

Postgres Pro Enterprise Manager Documentation

Postgres Pro Enterprise Manager Documentation

1. Postgres Pro Enterprise Manager Overview	1
1.1. Features	1
1.2. Architecture	2
1.2.1. Description of Components	2
1.2.2. High-Level Architecture	4
1.2.3. Manager and Agent Architecture	5
1.2.4. Monitoring Architecture	5
1.2.5. Backup Architecture	7
1.2.6. Instance Data Collection and Post-Processing	8
1.3. Hardware and Software Requirements	11
1.3.1. Minimum Hardware and Software Requirements	11
1.3.2. Resource Calculation	12
1.4. Supported Platforms and Operating Systems	13
1.5. Compatibility	13
1.6. Security	14
1.6.1. Manager and Agent	14
1.6.2. Networking	15
1.6.3. User-to-Manager Authentication	15
1.6.4. Manager-to-User Authentication	15
1.6.5. Authentication and Authorization Operation Scheme	15
1.6.6. Role-Based Access Control (RBAC) Model	17
1.7. Considerations and Limitations	22
2. What's New	23
2.1. What's New in PPEM 2.3	23
2.2. What's New in PPEM 2.2	24
2.3. What's New in PPEM 2.1	25
2.4. What's New in PPEM 2.0	26
3. Quick Start	28
4. Installation and Setup	31
4.1. Installation in Secure Environments	31
4.2. Installation in the High Availability Mode	36
4.3. Installation and Configuration of Backup and Restore Tools	39
4.4. Installation and Configuration of Logging and Monitoring Tools	40
4.5. Integration with External Data Sources	41
4.5.1. Integration with Prometheus	41
4.5.2. Integration with Elasticsearch and APM	45
4.6. Integration with OpenLDAP and Active Directory	52
4.7. Connecting the Manager and Agents to Instances via SSL	58
4.8. Integration with Tracing Tools	58
4.8.1. Tracing Architecture	58
4.8.2. Integration with Grafana Tempo	59
4.9. Configuration Behind a Reverse Proxy	60
5. Using PPEM	63
5.1. Logging in to the Web Application	63
5.2. Web Application Interface	63
5.3. System Configuration	64
5.3.1. Viewing PPEM Summary Information	65
5.3.2. Agents	65
5.3.3. Tags	67
5.4. User Management	68
5.4.1. Users	68
5.4.2. User Groups	70
5.4.3. User Roles and Privileges	71
5.5. Instances	75
5.6. Clusters	84
5.7. Configuration Presets	92
5.8. Tablespace	94
5.9. Databases	96

5.10. Schemas	102
5.11. Tables	104
5.12. Indexes	113
5.13. Functions	116
5.14. Sequences	120
5.15. Control Center	122
5.16. Metrics	123
5.17. Alerts	125
5.18. Activity	133
5.18.1. User Sessions	133
5.18.2. Vacuum Processes	136
5.18.3. Statistics Collection Processes	138
5.18.4. Clustering Processes	139
5.18.5. Reindexing or Index Creation Processes	140
5.18.6. Base Backup Processes	142
5.18.7. Copying Processes	144
5.19. SQL Statistics	145
5.20. Repository Maintenance	147
5.21. Instance Parameters	152
5.22. Extensions	154
5.23. Profiler	155
5.23.1. Profiler Servers	155
5.23.2. Samples	158
5.23.3. Viewing Profiler Graphs	159
5.23.4. Sampling Schedules	163
5.23.5. Reports	166
5.24. Authentication	170
5.25. Instance Roles	172
5.26. Message Log	173
5.27. Task Console	174
5.27.1. Viewing Tasks	174
5.27.2. Managing Schedules	174
5.27.3. Managing Mass Instance Operations	175
5.28. Backup	178
5.28.1. Storages	178
5.28.2. Instance Binds	183
5.28.3. Backups	186
5.28.4. Backup Schedules	193
5.28.5. Configuring Backup Storage Parameters for an Instance	197
5.29. Query Plan Visualization	198
6. Upgrade and Migration	200
6.1. Upgrading to PPEM 2.3	200
6.2. Upgrading to PPEM 2.2	201
6.3. Upgrading to PPEM 2.1	202
6.4. Upgrading to PPEM 2.0	203
7. Troubleshooting	205
8. Terms and Abbreviations	207

Chapter 1. Postgres Pro Enterprise Manager Overview

Postgres Pro Enterprise Manager (PPEM) is your all-in-one solution for streamlined database infrastructure administration. PPEM delivers an intuitive interface for managing and monitoring the PostgreSQL, Postgres Pro Standard, and Postgres Pro Enterprise DBMS.

PPEM tools allow you to perform a wide range of routine DBMS maintenance tasks, ensuring seamless management of your ecosystem across all operational areas — from backup and recovery to performance tracking, monitoring, troubleshooting, and beyond.

To get started with PPEM, refer to [Quick Start](#).

For more information about PPEM components and functionality, refer to [Architecture](#) and [Features](#).

For more information on working with PPEM, refer to:

- [Hardware and Software Requirements](#): Server specifications for PPEM deployment and resource calculation guidelines.
- [Supported Platforms and Operating Systems](#): The list of supported platforms and operating systems.
- [Compatibility](#): The list of supported Postgres Pro Standard and Postgres Pro Enterprise versions, as well as browsers.
- [Security](#): Access rights, networking, authentication, and Role-Based Access Control (RBAC).
- [Considerations and Limitations](#): Important aspects to consider when using PPEM.
- [Installation and Setup](#): Various installation and configuration options.
- [Using PPEM](#): Detailed step-by-step operating instructions.
- [Upgrade and Migration](#): Instructions for upgrading to new versions.
- [Troubleshooting](#): Instructions for self-diagnosis and troubleshooting.

You can find information about new releases and functionality upgrades in [What's New](#).

1.1. Features

PPEM enables you to perform the full cycle of Postgres Pro database deployment, operation, and maintenance tasks:

- Automatically discovering DBMS instances

PPEM discovers active DBMS instances and automatically adds them to the web application, determines the backup tools used, and synchronizes the backup configuration with the web application. Automatic discovery features minimize the time required to reconfigure DBMS instances to operate with PPEM and help to quickly start using PPEM, as well as to facilitate migration from other management tools.

- Installing and starting DBMS instances

PPEM allows you to deploy new DBMS instances using a browser. You can deploy single and cluster DBMS configurations, as well as deploy from backups.

- Configuring DBMS instances

For new and existing DBMS instances, PPEM provides tools to view and set configuration parameters. This eliminates the need to connect to the instance and directly edit configuration files, accelerating the setup process.

- Viewing and managing internal objects of DBMS schemas

PPEM allows you to view the internal structure of DBMS instances (tablespaces, databases, tables, indexes, and other objects) and also provides tools for creating new objects.

- Tracking internal activity of DBMS instances

PPEM interacts with the statistics subsystem and provides information about the DBMS internal activity: active and background processes, query execution statistics, wait events, and locks. PPEM can also be used to cancel or forcefully terminate DBMS processes.

- Visualizing running processes and analyzing query plans

PPEM integrates with [pgpro-otel-collector](#) that collects DBMS statistics and activity logs and provides metrics. PPEM acts as a consumer of this telemetry and provides graphs for evaluating and analyzing the DBMS performance.

- Profiling and integrating with `pgpro_pwr`

PPEM integrates with the `pg_profile` catalog and `pgpro_pwr` module. It also provides a convenient interface for working with snapshots and reports.

- Planning, starting, and monitoring routine operations

PPEM includes tools for service tasks such as vacuuming, collecting scheduler statistics, reindexing, and others. These operations can be performed either manually or automatically on a regular basis using the task scheduler.

- Backup and restore tasks

PPEM can automatically discover backup tools (`pg_probackup`), integrate with them, and provide viewing and management tools. PPEM allows you to perform backups both manually and on schedule, as well as create new DBMS instances from backups. Additionally, the point-in-time recovery (PITR) allows for more flexible backup restoration tasks.

- Directly accessing the DBMS instance console

PPEM allows you to connect to the instance console via the web version of `psql`. This tool is designed for experienced administrators and allows them to quickly connect to the DBMS and interact with it directly.

1.2. Architecture

1.2.1. Description of Components

The Postgres Pro Enterprise Manager architecture comprises the components below.

Web Application

Web application provides a graphical user interface accessible through a browser. The web application is installed on the same server as the [manager](#). Users perform various actions using the browser, and the web application transforms these actions into requests and sends them to the manager. The manager processes requests and returns them to the web application. The web application transforms the response into various representations and then displays it in the browser.

Manager

PPEM manager (hereinafter referred to as the manager) is a service that runs in the background. It accepts requests for DBMS infrastructure maintenance and has its own scheduler for performing regular service tasks. The manager also accepts requests from the [web application](#) and transforms them into operations according to the designed logic.

To perform operations, you need the following:

- various service data, which is usually (but not necessarily) stored in the repository

- executing instructions on the agent side

The manager reports the operation results to the web application upon completion.

Note

The operations performed by the manager can be synchronous or asynchronous. In case of synchronous operations, the manager is forced to wait until the operation completes to obtain the response. In case of asynchronous operations, the manager immediately receives a response that the operation has been queued for execution. In this case, the manager needs to periodically check the operation status, but, in most cases, the user receives a notification upon the operation completion.

Repository

PPEM repository (hereinafter referred to as the repository) is a database on a dedicated DBMS instance with a set of schemas and tables where the [manager](#) stores service data necessary for operation. When the manager starts, it establishes a connection with the repository and reads and writes data from and to it during operations. Repository availability is critical, since the manager cannot continue operating if the repository is unavailable.

Agents

PPEM agents (hereinafter referred to as agents) are services that run on managed DBMS servers. They receive control instructions from the [manager](#) and, depending on the instruction type, perform various actions both in the operating system and in the DBMS instance, for example:

- running commands
- creating directories and files
- executing queries

During or after execution, the execution result is sent to the manager to be recorded in the repository and/or subsequently processed according to the operation logic.

Agents also have an internal service task scheduler that regularly performs background tasks and sends their results to the manager.

One agent is enough on a standalone server to serve one or more DBMS instances.

DBMS Instance

DBMS instance is a PPEM-managed object, i.e., the Postgres Pro DBMS in various editions (Postgres Pro Standard, Postgres Pro Enterprise). Several DBMS instances can be combined into clusters, typically streaming replication clusters.

Only a single [agent](#) interacts with a single DBMS instance. Avoid scenarios where multiple agents interact with the same DBMS instance, as this can lead to confusion and ambiguous behavior. From the DBMS instance standpoint, the agent is a regular application software that connects to the instance via the SQL interface using a predetermined DBMS account.

External Services

PPEM can use various *external services* to extend its features and capabilities. All these services are optional, and interaction with them is configured separately.

Note

PPEM does not include tools for administrative management of external services (e.g., resource and configuration control).

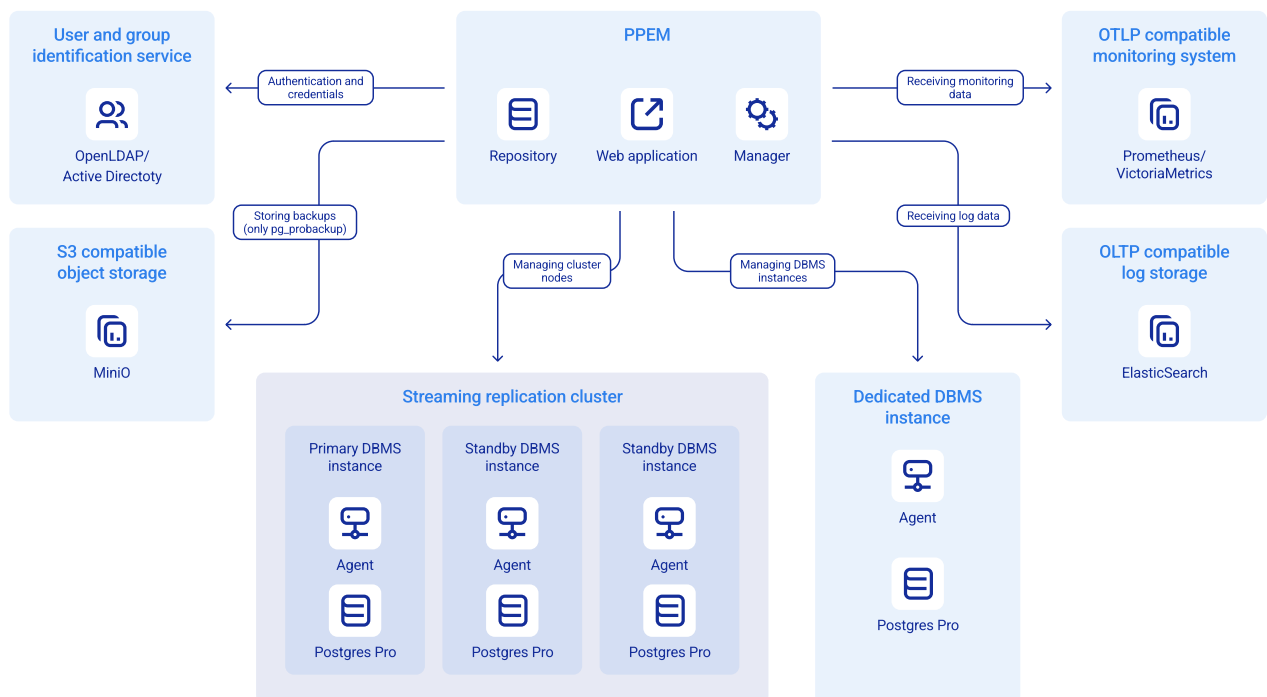
The following external services are supported:

- LDAP directory is a directory of users and groups for authenticating users in PPEM. PPEM supports OpenLDAP and Microsoft Active Directory.
- S3-compatible object storage is used by PPEM to store backups made with the `pg_probackup` utility.
- OTLP-compatible metrics storage system is a metrics storage system that supports the push model with the ability to record metrics via the OTLP protocol, for example, VictoriaMetrics, and/or the pull model capable of fetching metrics via the HTTP protocol, for example, Prometheus. PPEM can use such a system to receive metrics that are written there by the `pg-pro-otel-collector`. PPEM supports both VictoriaMetrics and Prometheus.
- OTLP-compatible log storage system is a metrics storage system capable of receiving logs via the OTLP protocol. PPEM can use such a system to receive DBMS logs that are written there by `pgpro-otel-collector`. PPEM supports Elasticsearch.

1.2.2. High-Level Architecture

The diagram below illustrates an example of a high-level architecture and interaction between components.

Figure 1.1. PPEM architecture



Where:

- PPEM is a DBMS infrastructure management service comprising **web application**, **manager**, **repository**, and **agents**.
- A **dedicated DBMS instance** is a PPEM-managed object that represents the PostgreSQL, Postgres Pro Standard, or Postgres Pro Enterprise DBMS.
- The replication cluster is several DBMS instances combined into a cluster, typically a streaming replication cluster.

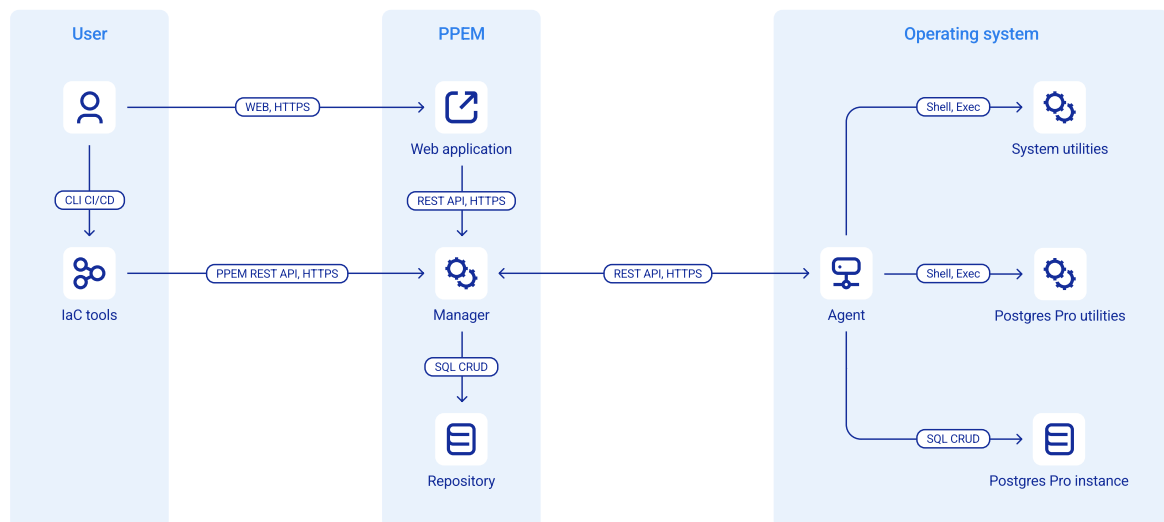
- The user and group identification service is a user and group directory that can be used to authenticate users in PPEM.
- The S3-compatible object storage is a system for storing backups made with `pg_probackup`.
- The OTLP-compatible monitoring system is a monitoring system capable of receiving metrics via the OTLP protocol or fetching them via HTTP. PPEM can use this system to receive DBMS performance metrics.

The user and group identification service, S3-compatible object storage, and OTLP-compatible monitoring system are optional [external services](#).

1.2.3. Manager and Agent Architecture

The diagram below illustrates an example of an architecture and interaction between the [manager](#) and [agent](#).

Figure 1.2. Manager and Agent Architecture



Where:

- Users can work with PPEM both through the [web application](#) in a browser and through automation tools (IaC), which can interact with PPEM via the REST API.
- In PPEM, the main graphical user interface is the web application. The web application is closely associated with the manager, from which it receives data and sends user control instructions. The manager provides an API for obtaining data and managing the DBMS infrastructure. The manager stores the intermediate state of the infrastructure in the [repository](#) database and interacts with the agent, which manages the [DBMS instance](#).
- The operating system is the working environment in which the DBMS instance and the agent are started. The agent interacts with the manager: sends environment data (OS and DBMS instance information) and receives control instructions.

1.2.4. Monitoring Architecture

To provide monitoring functions related to metrics and logs (hereinafter telemetry), PPEM uses [pg-pro-otel-collector](#) from Postgres Pro.

Working with telemetry can be organized in two ways:

- The **repository** databases are used to store telemetry: metrics are stored in the separate `monitoring` schema, and logs are stored in the `logs` schema.
- External data storages (in relation to PPEM) can be used to store telemetry.

In both cases `pgpro-otel-collector` is the required component. The collector gathers statistics and logs, generates telemetry data, and, depending on the selected mode, exports data to the manager or to an external storage system.

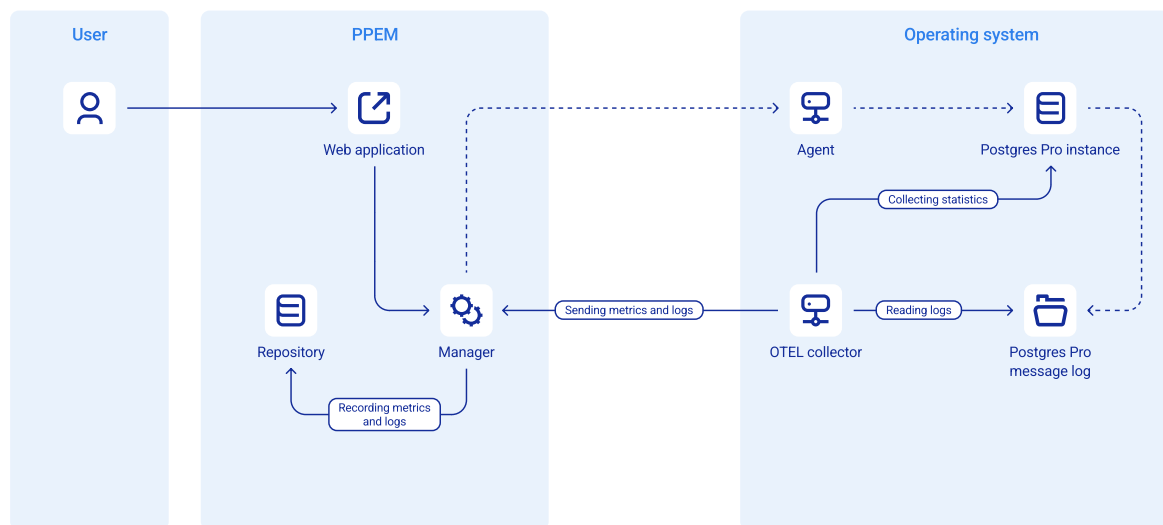
Note

Using the internal storage is not recommended in production environments, as it has limited scalability and potential performance issues if data volumes grow. Writing and reading large amounts of telemetry can negatively impact PPEM overall performance when accessing the repository database.

1.2.4.1. Internal Storage

The diagram below illustrates an example of a monitoring architecture that uses an internal storage.

Figure 1.3. Monitoring Using Internal Storage



Where:

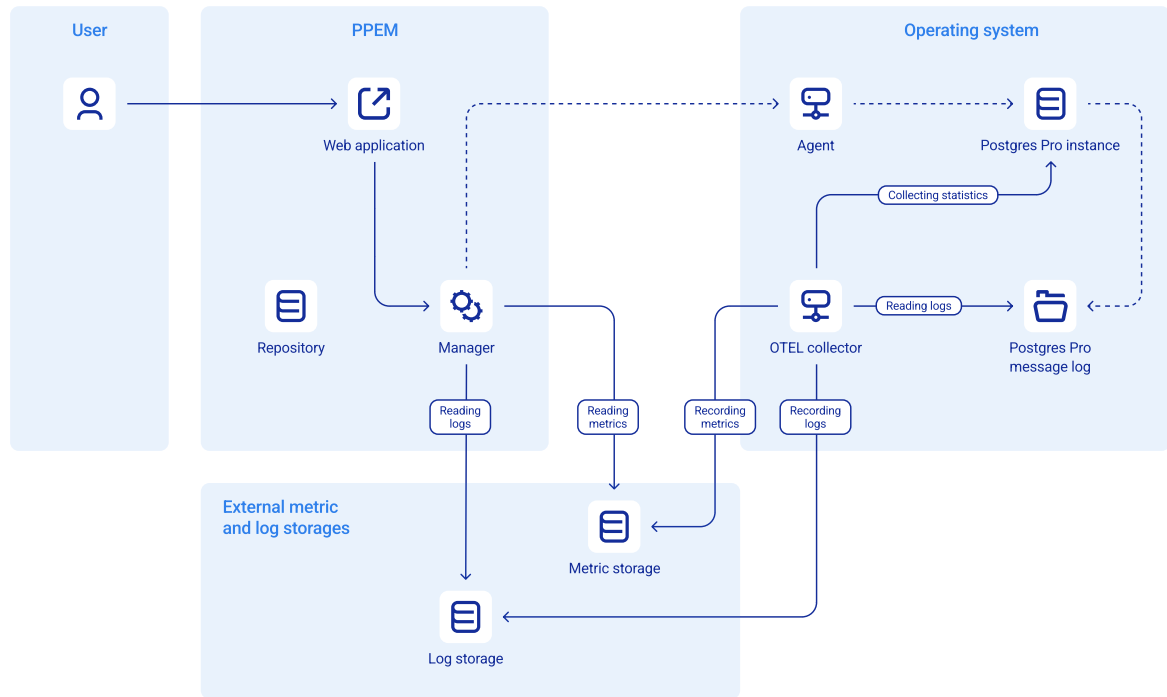
- Users work with PPEM through the web application in a browser where they view graphs of metrics and logs.
- In PPEM, metrics and logs are stored in the **repository**. The **manager** receives metrics and logs from the collector and stores them in repository databases. It also requests metrics and logs from the repository databases and returns them to the consumer via the REST API. The **web application** queries the manager for metrics and logs and then visualizes the received data.

- In the operating system, pgpro-otel-collector connects to the DBMS instance to receive metrics, reads the DBMS log files, processes the received data, and sends it to the manager via the REST API. The [agent](#) is a standalone component that provides control functionality and does not collect metrics or logs.

1.2.4.2. External Storage

The diagram below illustrates an example of a monitoring architecture that uses an external storage.

Figure 1.4. Monitoring Using External Storage



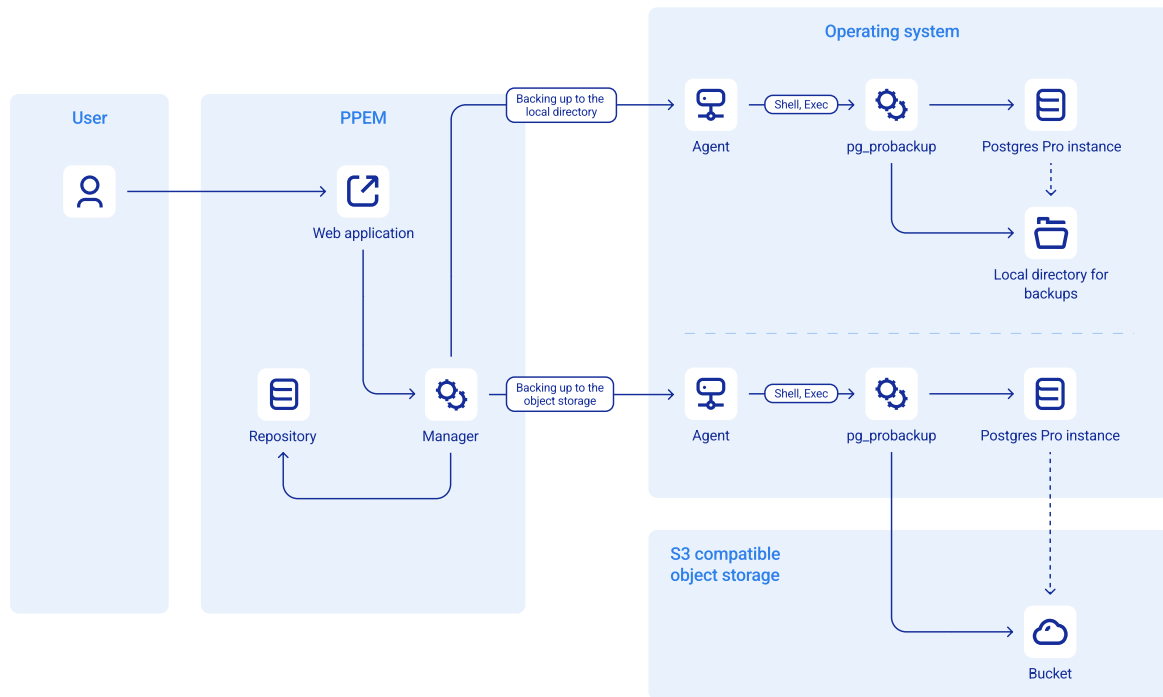
Where:

- Users work with PPEM through the [web application](#) in a browser where they view graphs of metrics and logs.
- In PPEM, the [manager](#) requests metrics and logs from external storages and returns them to the consumer via the REST API. The web application queries the manager for metrics and logs and then visualizes the received data.
- In the operating system, pgpro-otel-collector connects to the DBMS instance to receive metrics, reads the DBMS log files, processes the received data, and sends it to external storages via the REST API. The [agent](#) is a standalone component that provides control functionality and does not collect metrics or logs.
- [External storages for metrics and logs](#):
 - A dedicated monitoring system can act as a metrics storage. Prometheus, Victorimetrics, and OTLP-compatible storages are supported.
 - A separate system with OTLP-compatible storage can serve as a log storage. Elasticsearch is supported.

1.2.5. Backup Architecture

The diagram below illustrates an example of a backup architecture.

Figure 1.5. Backup Architecture



Where:

- Users utilize backup and restore functionality via the [web application](#).
- In PPEM, the web application sends requests to the [manager](#). The manager sends backup and restore requests to [agents](#).
- In the operating system, agents perform backup or restoration using the `pg_probackup` utility. A backup can be saved either to a local directory or to an S3-compatible object storage. The backup directory must be preconfigured to work with `pg_probackup`. In case of recovery, the backup can be obtained from either the local directory or from the object storage.
- The S3-compatible object storage is an [external service](#). Specific object storages are supported by means of `pg_probackup`, the support does not depend on PPEM.

1.2.6. Instance Data Collection and Post-Processing

This section explains how agents [collect instance data](#) and send it to the manager for [post-processing](#).

1.2.6.1. Instance Data Collection

Instance data collection by agents includes the following steps:

1. Connecting to the default instance database.
2. Collecting global objects:
 - roles
 - tablespaces
 - databases
3. Collecting local objects for each database:
 - extensions

- schemas
 - tables
 - indexes
 - sequences
 - functions
 - languages
4. Sending the collected objects to the manager.
 5. Sending the `DELETE /instances/objects` request to the manager with the specified `instance_id` attribute and data collection time to delete obsolete objects, for example, deleted objects.
 6. Sending the `POST /instances/objects/post_processing` request to the manager to start [post-processing](#).

If the manager already performs post-processing, for example, if it was not finished after the previous data collection cycle, `429 Too Many Requests` is returned. In this case, the agent finishes the current data collection cycle and postpones post-processing to the next cycle.

Important Considerations

- Every instance can potentially contain hundreds of thousands of objects, so agents collect data in batches.

You can specify the size of batches in the `ppem-agent.yml` agent configuration file using the `collectors.instance_objects.batch_size: number_of_objects_in_each_batch;` parameter.

- Agents send extended requests for collecting the following table and index data:

- For tables:
 - the table sizes per `main`, `vm`, and `fsm` layers, in bytes
 - the bloat size, in bytes
 - the TOAST size, in bytes
 - the number of tuples
 - the number of pages
 - the total index size, in bytes
 - the storage parameters
 - the table file path
- For indexes:
 - the index size, in bytes
 - the bloat size, in bytes
 - the storage parameters
 - the index file path

Extended requests can be resource-intensive and are sent once per 5 data collection cycles to avoid overloading the instance. You can specify how often extended requests are sent in the `ppem-agent.yml` agent configuration file using the following parameters:

- `collectors.instance_objects.extended.enabled`: Specifies whether extended requests are sent.

Possible values:

- `true`
- `false`
- `collectors.instance_objects.extended.interval`: The time interval for sending extended requests.

Alternatively, you can specify this time interval in the crontab format using the `collectors.instance_objects.extended.schedule` parameter. This parameter takes precedence over `collectors.instance_objects.extended.interval`.

- For composite objects, i.e., databases, schemas, and tables, agents collect information on all their dependencies. Based on this information, the manager generates summary information after the data collection cycle is completed.
- Agents automatically reconnect to the Postgres Pro DBMS instance once per 10 requests to avoid bloating the cache of the corresponding backend.

1.2.6.2. Post-Processing

Post-processing by the manager includes the following steps:

1. Updating the `size_bytes` field of the collected objects:

- For tables, the value is calculated as:

`relation_size + visibility_map_size + free_space_map_size + TOAST_size`

- For indexes, the value equals *`relation_size`*.
- For schemas and databases, the value is the total size of all their dependencies.

Note

The size of the `pg_toast` schema's tables is not considered since it is included in the size of database tables.

2. Re-generating the following summary information for all composite objects:

- For databases:
 - the total sum of table sizes, in bytes
 - the total sum of index sizes, in bytes
 - the number of tables
 - the number of indexes
 - the total bloat, in bytes
 - `available_xid_total` and `available_xid_percent`
- For schemas:
 - the number of tables
 - the number of indexes
 - the table size and bloat
 - the index size and bloat
- For tables:
 - the number of indexes

- the total sum of index sizes
- the total sum of bloat sizes

The summary information is stored in the `instance_objects` table as a JSONB column. The structure of the summary information depends on the composite object type:

```
database:
tables_count: INT
tables_all_size_bytes: BIGINT
tables_all_bloat_size_bytes: BIGINT
indexes_count: INT
indexes_all_size_bytes: BIGINT
indexes_all_bloat_size_bytes: BIGINT
available_xid_total: BIGINT
available_xid_percent: BIGINT

schema:
tables_count: INT
tables_all_size_bytes: BIGINT
tables_all_bloat_size_bytes: BIGINT
indexes_count: INT
indexes_all_size_bytes: BIGINT
indexes_all_bloat_size_bytes: BIGINT

table:
indexes_count: INT
indexes_all_size_bytes: BIGINT
indexes_all_bloat_size_bytes: BIGINT
```

1.3. Hardware and Software Requirements

This section provides hardware and software requirements for PPEM, as well as information on resource calculation when scaling the DBMS.

1.3.1. Minimum Hardware and Software Requirements

Below are the minimum hardware and software requirements for servers where the [manager](#) and [agent](#) are deployed.

The following can act as a server:

- bare metal server
- Xen/KVM/VMWare-based virtual machine
- container environment in privileged mode

Important

Using PPEM at minimum requirements does not provide fault tolerance and is recommended only for pilot or experimental projects, low-load environments without high availability and fault tolerance requirements.

For the manager:

One server with the following minimum configuration:

- CPU: 2 pcs.
- RAM: 4 GB

- SSD: 40 GB
- Platform: x86_64 or any listed in [Supported Platforms and Operating Systems](#)
- Operating system: Linux or any listed in [Supported Platforms and Operating Systems](#)

For the agent:

One server with the following minimum configuration:

- CPU: 2 pcs.
- RAM: 2 GB
- SSD: 10 GB
- Platform: x86_64 or any listed in [Supported Platforms and Operating Systems](#)
- Operating system: Linux or any listed in [Supported Platforms and Operating Systems](#)

1.3.2. Resource Calculation

For the [manager](#):

The required resources for the manager for 10 DBMS instances with 10 thousand objects in a separate instance:

- CPU: 2 pcs.
- RAM: 2 GB
- SSD: 40 GB

The minimum configuration of the server where the manager is deployed (up to 10 instances with 10 thousand objects in a separate instance):

- CPU: 2 pcs.
- RAM: 4 GB
- SSD: 40 GB

Resource calculation table:

Number of Instances	CPU, pc.	RAM, GB	SSD, GB
10	2	4	40
30	4	12	80
70	8	28	160
150	16	60	320
230	24	92	480
310	32	124	640

For the [agent](#):

The required resources for the agent for 1 DBMS instance (up to 100 thousand instance objects):

- CPU: 1 pc.
- RAM: 512 MB
- SSD: 20 GB

The minimum configuration of the server with the DBMS instance where the agent is deployed (up to 100 thousand instance objects):

- CPU: 2 pcs.
- RAM: 2 GB
- SSD: 20 GB

Resource calculation table:

Number of DBMS Objects	CPU, pc.	RAM, GB	SSD, GB
100 thousand	2	2	10
200 thousand	3	4	20
400 thousand	4	8	20

1.4. Supported Platforms and Operating Systems

The [manager](#) and [agent](#) support x86_64-based Linux operating systems.

PPEM supports installation and operation on the following Linux distributions:

- Alt Linux 10
- Alt Linux 11
- Alt Linux SP 8.2
- Alt Linux SP 8.4
- Alt Linux SP 10
- Astra Linux Smolensk 1.7
- Astra Linux Smolensk 1.8
- Debian 11
- Debian 12
- Red OS 7.3
- Red OS 8.2
- Red Hat Enterprise Linux 8
- Red Hat Enterprise Linux 9
- ROSA 2021.1
- SUSE Linux Enterprise Server 15
- Ubuntu 20.04
- Ubuntu 22.04
- Ubuntu 24.04

Important

PPEM operation has not been tested with distributions and non-x86_64 platforms other than those listed above, and proper operation is not guaranteed.

1.5. Compatibility

PPEM is compatible with the following products:

- Compatibility with PostgreSQL, Postgres Pro Standard, and Postgres Pro Enterprise:
 - For the [repository](#) service instance, versions from 14 to 16 are supported.
 - [Agents](#) can manage versions from 12 to 17.
- Agents support version 2.8 of pg_probackup.
- Agents support version 0.3 of pgpro-otel-collector.
- Browser compatibility:

To get the most out of the [web application](#), consider the following:

- Use a screen with a minimum resolution of 1280×720. A resolution of 1440×900 or higher is recommended for best experience.
- Use the following web browsers:

- Chrome 87 or later
- Firefox 85 or later
- Opera 73 or later
- Safari 14 or later

Important

Backward compatibility can be unavailable. For more information about issues with backward compatibility, refer to [What's New](#).

Direct backward compatibility is not guaranteed.

1.6. Security

1.6.1. Manager and Agent

The [manager](#) is a standard application software and does not require privileged access to operating system features. The manager service can operate fully when running under a non-privileged operating system user.

To work with the [repository](#), the manager needs a separate database where service information is stored. Also, the DBMS user with the following rights is required:

- The right to `LOGIN` to the instance.
- The repository database rights:
 - the ownership of the database
 - the right to connect to the database
 - no restrictions on access rights within the database (to perform migrations in the data schema)

The [agent](#) is a standard application software that requires the following for full operation:

- access to operating system features
- access to the managed DBMS instance

To implement most features, the agent only requires the access level of a non-privileged operating system user. There is a small number of features that require privileged access. To maintain this functionality, additional system configuration and granting of the necessary rights are required. Without the configuration and privileges, the agent cannot perform operations, which adversely affects PPEM functionality. It is recommended to complete all necessary configurations before running the agent.

Access to the managed DBMS instance can be divided into the following parts:

- Access to files and directories of the DBMS instance, which is provided using operating system access levels. The user on whose behalf the agent is running must have access to the main data directory.

Note

By default, the main data directory is initialized by the `postgres` owner with `0600` rights, so most DBMS installations restrict access to this configuration. Therefore, the optimal operational approach is to run the agent under the `postgres` system user.

- Access to the SQL interface of the DBMS instance, for which the agent requires the DBMS user with the following rights:
 - the right to `LOGIN` to the instance

- the right to connect to all instance databases
- the membership of the `pg_monitor` and `pg_signal_backend` roles

1.6.2. Networking

Networking between the [manager](#) and [agent](#) can be initiated by either side:

- The manager can send control instructions to agents, and agents return the execution results in response.
- Agents can send to the manager requests to register themselves and DBMS instances as well as to update the state of instances. The manager, in turn, sends a response to the agents' requests.

The manager and agent use the HTTPS protocol for communication. By default, the manager uses the `tcp/8080` port, and the agent uses the `tcp/8081` port. This traffic direction must be taken into account when configuring network access rules. The address and port are specified in the `ppem-manager.yml` manager and `ppem-agent.yml` agent configuration files.

For secure data transfer, TLS configuration is recommended.

1.6.3. User-to-Manager Authentication

To work with the [manager](#), user authentication and authorization are required. Authentication can be performed using:

- Built-in PPEM tools — user and group data is stored in the [repository](#) and managed by the PPEM administrator.
- An external LDAP directory (OpenLDAP or Active Directory). If you use the [external directory](#), the manager must be configured to work with this directory. All user and group data is stored in an external directory and managed by a dedicated LDAP directory administrator.

1.6.4. Manager-to-User Authentication

The [manager](#) and [agent](#) API is secured by authorization. To execute API requests, the manager and agent perform mutual authentication and issue access tokens for subsequent authorization. The tokens have a limited lifetime — the manager and agents autonomously track token expiration and renew them when necessary.

1.6.5. Authentication and Authorization Operation Scheme

There are three interaction types that require authentication and authorization during the [manager's](#) operation:

- [User → Manager](#)
- [Manager → Agent](#)
- [Agent → Manager](#)

All interactions are carried out via the HTTP or HTTPS protocol depending on the PPEM parameters.

1.6.5.1. User → Manager

In this interaction type, users work in the [web application](#).

1.6.5.1.1. User → Manager: Authentication

The user sends a `POST /v1/sessions` API request to the manager's endpoint to obtain access and refresh tokens for subsequent operations. The API request contains user credentials.

In the basic scenario, credentials are checked against the [repository](#). When using LDAP authentication, credentials are first checked against the directory service, then if they are not found, they are checked against the repository.

1.6.5.1.2. User → Manager: Authorization

Upon successful authentication, subsequent HTTP/HTTPS requests from users to the manager include the `Authorization` header, where the `Bearer` keyword is followed by a space and the access token text. For example:

```
headers:
Content-Type: application/json
Authorization: "Bearer eyJhbG..."
```

The granted access is determined using the [role-based access control \(RBAC\) model](#) in accordance with the roles assigned to the user:

- You can assign roles to [PPEM users](#) directly or through [PPEM groups](#) they are members of.
- You can assign roles to LDAP users only through PPEM groups they are members of.

To add an LDAP user to a PPEM group, the administrator must map the distinguished name (DN) of the user's LDAP group to the PPEM group name. The user is then automatically added to the PPEM group when [logging in to the web application](#).

The user group composition is periodically checked against the LDAP server and updated if necessary.

For more information about mapping LDAP groups to PPEM groups, refer to [Integration with OpenLDAP and Active Directory](#).

1.6.5.2. Manager → Agent

In this interaction type, the [manager](#) sends API requests to [agents](#) to perform various operations.

1.6.5.2.1. Manager → Agent: Authentication

The manager sends a `POST /v1/sessions` API request to the agent's endpoint to obtain access and refresh tokens for subsequent operations. The API request is made via the URL for connecting the agent to the manager from the repository and contains two parameters:

- `name`: The agent name from the repository.
- `api_key`: The API key for connecting the agent to the manager from the repository.

1.6.5.2.2. Manager → Agent: Authorization

Upon successful authentication, subsequent HTTP/HTTPS requests from the manager to agents include the `Authorization` header, where the `Bearer` keyword is followed by a space and the access token text. For example:

```
headers:
Content-Type: application/json
Authorization: "Bearer eyJhbG..."
```

The agent verifies the authenticity of the manager's access token. To do this, the agent generates a token based on the known values of the `name` and `api_key` parameters. If the generated and received tokens match, the authentication is successful. If the authenticity is confirmed and the requested endpoint is found, further work is allowed.

1.6.5.3. Agent → Manager

In this interaction type, [agents](#) send reports on the API command execution and updates about serviced instance objects to the [manager](#).

1.6.5.3.1. Agent → Manager: Authentication

The agent sends a `POST /v1/sessions` API request to the manager's endpoint to obtain access and refresh tokens for subsequent operations. The API request is made via the URL for connecting the agent

to the manager, which is specified in the `ppem-agent.yml` agent configuration file using the `agent.manager.url` parameter. The API request contains two parameters:

- `name`: The agent name, which is specified in the `ppem-agent.yml` agent configuration file using the `agent.name` parameter.
- `api_key`: The API key for connecting the agent to the manager, which is specified in the `ppem-agent.yml` agent configuration file using the `agent.manager.api_key` parameter.

1.6.5.3.2. Agent → Manager: Authorization

Upon successful authentication, subsequent HTTP/HTTPS requests from the agent to the manager include the `Authorization` header, where the `Bearer` keyword is followed by a space and the access token text. For example:

```
headers:
Content-Type: application/json
Authorization: "Bearer eyJhbG..."
```

The manager verifies the authenticity of the agent's access token. To do this, the manager generates a token based on the known values of the `name` and `api_key` parameters. If the generated and received tokens match, the authentication is successful. If the authenticity is confirmed and the requested endpoint is found, further work is allowed.

The granted access is determined by the rules specified in the PPEM code for how the agent should access manager resources.

1.6.5.4. Lifetime of Access and Refresh Tokens

The lifetime of access and refresh tokens is limited. It can be specified in the `ppem-manager.yml` manager or `ppem-agent.yml` agent configuration file using the `jwt.lifetime.access` and `jwt.lifetime.refresh` parameters.

When the access token expires, the manager or agent starts responding to the token owner with the "401 Unauthorized" error and the "ERR_AUTH_TOKEN_EXPIRED" code, for example:

```
{
  "error": {
    "code": "ERR_AUTH_TOKEN_EXPIRED",
    "title": "token is expired"
  }
}
```

To get a new access token, the token owner must send a `PUT /v1/sessions` API request to the endpoint with the refresh token that was received along with the expired access token. This results in new access and refresh tokens that can be used for further work.

If the refresh token has also expired, the token owner must re-authenticate.

1.6.6. Role-Based Access Control (RBAC) Model

1.6.6.1. Key Concepts

Access control in PPEM is implemented via the RBAC (Role-Based Access Control) model, which defines access restriction rules through roles and privileges. The model establishes the following basic conventions:

- *Object* is a resource to which access should be granted or restricted.
- *Subject* is a person (user) or an automated agent.
- *Privilege* is a permission granting access to perform an operation with the object.
- *Role* is a job function or title defined at the authorization level.

The basic conventions define the following extended conventions:

- *Access rule* is a combination of a role, privileges, and their relationships.
- *Session* is an association between a subject and a role.
- One subject can have several roles.
- One role can belong to several subjects.
- One role can have several privileges.
- One privilege can belong to several roles.

Note

Privileges are not assigned to subjects directly but are acquired by subjects through roles.

1.6.6.2. RBAC in PPEM

Fundamental principles:

- PPEM functionality is composed of *plugins*.
- Each plugin helps to manage a specific *resource*.
- Resources can be managed by basic *CRUD operations*: create, view, edit, and delete.
- Additionally, resource management may include various *RPC operations*. Their names depend on the resource management operation.
- *Privileges* allow subjects to perform certain operations. All resources are managed through privileges (basic minimum requirements):
 - `RESOURCE\_create`
 - `RESOURCE\_view`
 - `RESOURCE\_edit`
 - `RESOURCE\_delete`
- Privileges are included in *roles*.
- Roles can be assigned to a subject both at the time of subject creation and later.
- When the plugin receives a request from a subject to perform an operation, it checks the subject's permission to access the requested operation.
- Subjects (users) can create their own roles (if they have privileges) and assign these roles to other subjects.
- Privileges and roles may be object-specific. This means that you can specify a role and privilege that allows access to a limited set of objects.
- Binding to an object can be done at the role level (all subjects with this role have access to the object) or at the user level (only one user has access to the object).

1.6.6.2.1. Subjects

The following may act as subjects:

- Users. Roles are assigned to users at creation time. If roles are not explicitly specified, a default role may be assigned.
- [Agents](#). This role is assigned when creating or registering agents.

1.6.6.2.2. Objects

Objects can be either resources or their representations in the [repository](#), such as servers, agents, instances, users, and user groups.

1.6.6.2.3. Setting Privileges and Roles

On the first PPEM startup, the repository initialization is performed, during which system tables are populated (privileges and roles).

During initialization, each plugin is assigned its own set of privileges, roles, and “role-privilege” associations. For example, the accounts plugin has dedicated privileges and roles for managing access to its objects. Other plugins receive their own sets of privileges and roles to control access to their respective objects.

If users are granted necessary privileges, they can create custom roles, specify privilege sets, link roles to objects, and assign custom roles to subjects.

1.6.6.3. Implementation

Access privileges are implemented by the following set of tables, which are owned by the accounts plugin:

- `privileges`: Privileges with object classes they regulate access to.
- `roles`: Roles with target object classes.
- `role_privileges`: The “role-privilege” association defines relationships between roles and privileges included in them.
- `users`: Users of the system (subjects).
- `user_roles`: The “user-role” association that defines relationships between users and roles assigned to them.
- `user_privileges`: The view that displays the “user-privilege” associations.
- `groups`: The groups of the system.
- `group_roles`: The “group-role” association defines relationships between groups and roles assigned to them.
- `group_users`: The “group-user” association defines relationships between groups and users included in them.

1.6.6.3.1. The `privileges` Table

The table has the following fields:

- `id`: The unique privilege ID.
- `name`: The unique privilege name.
- `title`: The privilege description.
- `class`: The class of objects to which the privilege grants access.
- `source`: The name of the plugin that sets the privilege and then performs the access check.

The table is populated by the `manager` using migrations. Each plugin determines its own set of privileges.

An HTTP middleware handler is used to check privileges. Creation, modification, and deletion of privileges by the user is not supported, since the privilege check is implemented in the manager code.

1.6.6.3.2. The `roles` Table

The table has the following fields:

- `id`: The unique role ID.
- `name`: The unique role name.
- `title`: The role description.
- `class`: The class of objects to which the privilege grants access. The field value is used as a UI hint to get privileges of the corresponding class and the list of objects of this class.
- `source`: The name of the plugin that sets the role. For custom roles, the `user` value is set.

Roles are created by the manager using migrations. The basic role set is established for the accounts plugin.

With the manager API, users can create, modify, and delete custom roles, but they cannot modify or delete roles set by the manager (system roles).

1.6.6.3.3. The `role_privileges` Table

The table defines associations between roles and privileges and has the following fields:

- `id`: The unique association ID.
- `role`: The role ID.
- `privilege`: The privilege ID.
- `parametric`: Specifies whether object binding is used.
- `object`: The unique ID of an arbitrary class object.

The object class is determined by the `privileges.class` field value. The object ID declared here restricts the privilege specified in `role_privileges.privilege` to a single object and allows access to the object only by members of a single role specified in `role_privileges.role` (access is granted to users in `user_roles.user` who have `user_roles.role = role_privileges.role`).

The combination of `role`, `privilege`, and `object` is a unique key with the `object IS NOT NULL` condition.

Note

A parameterized object-specific “role-privilege” association determines access only to a single object (when specifying `object`). If access is required for N objects, N entries in the `role_privileges` table are created.

1.6.6.3.4. The `users` Table

The table has no RBAC-related fields. When creating a user via the manager API, you can specify a list of role IDs that will be assigned to the user.

1.6.6.3.5. The `user_roles` Table

The table defines associations between users and roles and has the following fields:

- `id`: The unique association ID.
- `user`: The user ID.
- `role`: The role ID.
- `object`: The unique ID of an arbitrary class object.

The object class is determined by the `privileges.class` field value. The object ID declared here restricts the privilege specified in `role_privileges.privilege` to a single object and allows access to the object by the user specified in `user_roles.user`.

Note

A parameterized object-specific “role-privilege” association determines access only to a single object (when specifying `object`). If access is required for N objects, N entries in the `role_privileges` table are created.

1.6.6.3.6. The `user_privileges` View

This view shows users with their roles and privileges and makes it easier to check privileges of a specific subject:

- `user`: `user_roles.user`
- `role`: `user_roles.role`
- `privilege`: `privilege.name`
- `object`: `user_roles.objects` or `role_privileges.object`

1.6.6.3.7. The `groups` Table

The table has no RBAC-related fields.

When creating a group, you can specify a list of role IDs that will be assigned to the group and, as a result, to all users included in it.

1.6.6.3.8. The `group_roles` Table

The table defines associations between groups and roles and has the following fields:

- `group_id`: The group ID.
- `role_id`: The role ID.
- `object`: The unique ID of an arbitrary class object.

The object class is determined by the `privileges.class` field value. The object ID declared here restricts the privilege specified in `role_privileges.privilege` to a single object and allows access to the object by the user specified in `group_roles.role_id`.

Note

A parameterized object-specific “group-role” association determines access only to a single object (when specifying `object`). If access is required for N objects, N entries in the `group_roles` table are created.

1.6.6.3.9. The `group_users` Table

The table defines associations between groups and users and has the following fields:

- `group_id`: The group ID.
- `user_id`: The user ID.

The “group-user” associations are not object-specific.

1.6.6.4. Object-Specific Roles

In general, it is assumed that roles can include privileges allowing access to all objects of any class.

For finer-grained access control, you can specify both the class and the object ID in `role_privileges.object`. This ensures that the role and privilege apply only to the object with the specified ID. The object can be specified in multiple locations:

- In `role_privileges.object` for a role. In this case, object access is granted to all role members.
- In `user_roles.object` for a user. In this case, access is granted only to a single user.
- In `group_roles.object` for a group. In this case, access is granted only to group users.

1.6.6.5. Checking Access Privileges of Subjects

During the subject authentication, a session itself and session JWT tokens are created. When an access token is created, it includes `user_id`. When a user makes requests to perform operations, the access token is attached to the request headers.

The [manager](#) performs authorization, extracts the user’s `user_id` from the access token and checks for the privilege in roles. If the privilege is available, access is granted. If the privilege is unavailable, the request is rejected.

The following is required to check that a subject has the permission to perform an operation with an object:

- `user_id` or `agent_id`: The user or agent ID.
- `class`: The class name for the object (resource type).
- `object`: The object ID (optional).

The client specifies the `Authorization: Bearer` header in the request and attaches the access token.

The server receives the request and extracts the data needed to verify access:

- The `user_id` (or `agent_id`) value is extracted from the access token.

- The `class` value is defined based on the privilege.
- The `object` values:
 - For GET requests of the `/resources` type, the values are extracted from the URL and request parameters (`ids=?`).
 - For GET requests of the `/resources/objectID` type, the values are extracted from the URL.
 - For PUT requests, the values are extracted from the request body.
 - For DELETE requests of the `/resources` type, the values are extracted from the request body (a separate object field).

To verify that a user has a specific privilege, the handler may need a mapping of operation IDs to their corresponding privileges. The verification is performed via the repository and the `user_privileges` view.

1.7. Considerations and Limitations

Using security-enhancing tools that add extra restriction layers may interfere with some PPEM features. PPEM's normal operation is neither guaranteed nor recommended when using such tools. These security-enhancing tools include:

- SELinux mandatory access control (MAC) systems. For more information, refer to the [official SELinux documentation](#).
- Information protection suites in Astra Linux Special Edition. For more information, refer to the [official Astra Linux documentation](#).

Chapter 2. What's New

This section contains key information about PPEM releases, along with functionality updates and improvements.

For migration instructions and recommendations, refer to [Upgrade and Migration](#).

2.1. What's New in PPEM 2.3

Release date: 14 November 2025

For migration instructions and recommendations, refer to [Upgrading to PPEM 2.3](#).

PPEM 2.3 introduces the following functional upgrades:

- Configuration presets management.

Added centralized management of configuration presets to meet your organization's internal standards and requirements on information security and information system workload.

This functionality is in the beta phase and has limitations. It will be expanded and updated in further releases.

For more information, refer to [Configuration Presets](#).

- Mass instance operations.

Added the ability to perform an operation on multiple instances at once.

This functionality is in the beta phase and has limitations. It will be expanded and updated in further releases.

For more information, refer to [Managing Mass Instance Operations](#).

- Alerts and triggers.

Added the ability to send alerts to users and user groups via email based on alert triggers. Alert triggers fire when values of metrics are below, equal to, or higher than the thresholds specified in alert trigger rules.

This functionality is in the beta phase and has limitations. It will be expanded and updated in further releases.

For more information, refer to [Alerts](#).

- Repository maintenance.

Implemented automatic cleaning of the repository database tables to keep them from overflowing using housekeeping rules. Housekeeping rules clean tables when data is stored longer than the specified time and when certain conditions are met. Tables can also be cleaned at the specified time interval.

Housekeeping rules are especially relevant if you store instance logs and metrics in the repository database.

This functionality is in the beta phase and has limitations. It will be expanded and updated in further releases.

For more information, refer to [Repository Maintenance](#).

- BiHA cluster management.

Added centralized management of BiHA clusters and their nodes.

For more information, refer to [Clusters](#).

- User role management.

Added centralized management of user roles based on the existing privileges according to your organization's requirements on information security and privilege hierarchy.

For more information, refer to [User Roles and Privileges](#).

- Automatic user blocking.

Implemented automatic blocking of users after the specified number of failed attempts to log in to the web application. Users blocked due to too many failed log-in attempts are not unblocked automatically.

For more information, refer to [Configuring the Automatic User Blocking](#).

- Request tracing.

Added the ability to trace incoming requests to the manager and agents. Tracing data is exported from PPEM using OTLP (OpenTelemetry protocol) and sent to a receiver supporting OTLP.

For more information, refer to [Integration with Tracing Tools](#).

- Connecting to instances via SSL.

Added the ability for the manager and agents to connect to instances via SSL.

For more information, refer to [Connecting the Manager and Agents to Instances via SSL](#).

- Request proxy.

Added the ability to proxy requests to PPEM based on URL prefixes.

For more information, refer to [Configuration Behind a Reverse Proxy](#).

- PostgreSQL 18 is supported.

The manager and agents can now manage PostgreSQL 18 DBMS.

- Metrics storage model optimization.

Switched the repository database to the flat metrics storage model. The new model is less resource-intensive and less susceptible to deadlocks.

- Updated [Compatibility](#).
- Added information on limitations to [Installation in Secure Environments](#).
- Added [Instance Data Collection and Post-Processing](#).

This section explains how agents collect instance data and send it to the manager for post-processing.

2.2. What's New in PPEM 2.2

Release date: 28 July 2025

For migration instructions and recommendations, refer to [Upgrading to PPEM 2.2](#).

PPEM 2.2 introduces the following functional upgrades:

- Control center.

The PPEM [web application](#) now includes the Control Center section, which provides quick status assessment for all managed systems. This tool offers a convenient visual overview of all PPEM-controlled instances and allows viewing historical data for key instance metrics in the graph format.

For more information, refer to [Control Center](#).

- Session wait profile visualization.

The interface for the [pg_wait_sampling](#) extension is added, providing tools to view and analyze session wait profiles and history. The new section helps to identify dependency issues for queries processing longer than expected.

For more information, refer to [Viewing Statistics on Wait Events](#).

- Detection of Patroni-managed clusters.

Implemented automatic [detection of clusters](#) managed by the Patroni failover manager, with subsequent display in the Clusters section of the web application.

2.3. What's New in PPEM 2.1

Release date: 15 April 2025

For migration instructions and recommendations, refer to [Upgrading to PPEM 2.1](#).

PPEM 2.1 introduces the following functional upgrades:

- Working with replication clusters and integration with the BiHA failover manager.

The ability to work with streaming replication clusters is added. The PPEM [web application](#) now supports creating a cluster from an existing instance and displaying it in the corresponding section.

PPEM 2.1 takes the first step toward full BiHA cluster support. The new version displays existing BiHA clusters and provides functionality to remove nodes from an existing cluster.

- Managing data sources.

In the Infrastructure → Data sources section of the PPEM web application, you can now create external storages for metrics and message logs collected via pgpro-otel-collector.

For metrics storage, PPEM supports integration with Prometheus, while for message log storage, it supports integration with Elasticsearch. Using external storage for metrics and logs is recommended because it reduces the load on the PPEM service repository by offloading additional data storage. It also eliminates the need for DBAs to perform additional scaling or closely monitor the PPEM service repository.

External storage systems are specifically designed to handle large volumes of specific data types and can be scaled and managed independently of PPEM.

For more information about configuring the integration, refer to [Integration with External Data Sources](#).

- Custom SQL metrics.

It is now possible to create and view [metrics](#) based on SQL statements planning and execution statistics collected by the [pgpro_stats](#) extension.

When creating or editing an SQL metric, the user can select a database, specify the user on behalf of which the query will be executed, and set the metric collection interval. The list of created metrics will be displayed on the page, and metric values can be viewed by clicking on the metric name.

- Navigation panel redesign.

The navigation panel is redesigned based on user experience feedback from infrastructure and application DBAs.

Pages primarily intended for infrastructure DBAs are now grouped under the Infrastructure section, while the Databases section mainly used by application DBAs is promoted to the top navigation level for convenient and quick access to the database list.

- Verifying the consistency of instance data directory

The system now supports instance data directory verification using the `checkdb` command included in the `pg_probackup` extension. This command performs a physical verification of all data files in the directory, validating page headers integrity and verifying block-level checksums.

- Adding the `Console` user role.

For flexible control over access to the `psql` web console, PPEM 2.1 introduces the `Console` user role. Now, only PPEM users with this role can access the `psql` console from the web application to perform database operations.

- Installing and working with extensions.

It is now possible to install and work with [extensions](#) in the web application.

The DBMS instance navigation panel now includes the new Extensions section that displays the list of installed extensions that can be deleted, as well as the Install extension block for adding extensions to the specified databases.

2.4. What's New in PPEM 2.0

Release date: 19 February 2025

For migration instructions and recommendations, refer to [Upgrading to PPEM 2.0](#).

PPEM 2.0 introduces the following functional upgrades:

- Migration to Golang.

The [manager](#) and [agents](#) migrated to Golang, enhancing performance, scalability, and extensibility capabilities to handle increasing workloads and growing numbers of managed systems.

The new manager can serve a larger number of agents while consuming less memory and CPU resources. New agents also use less memory and CPU resources during routine operations.

- Automatic resource detection.

Agents can now work in automatic detection mode. In this mode, an agent searches for DBMS instances, automatically registers them in PPEM, and continuously maintains up-to-date information about each DBMS instance.

Automatic detection and registration simplify PPEM deployment in environments with numerous instances and minimize manual operations required to add agents and database instances in PPEM.

- Integration with `pgpro-otel-collector`.

`pgpro-otel-collector` is designed to collect metrics and activity logs from DBMS instances and send them to monitoring systems.

PPEM now integrates with `pgpro-otel-collector`, enabling the use of collector-gathered metrics for monitoring purposes. Integration can be performed in two ways:

- PPEM receives metrics and logs from the collector via the OTLP protocol and stores them in the repository internal database.

- PPEM is integrated with external log and metrics storage systems, allowing to get monitoring data from these sources. External storage integration allows PPEM to incorporate into existing monitoring infrastructure and use it for internal monitoring functions.

For more information, refer to [Monitoring Architecture](#).

- API redesign.

The API was redesigned, with initial steps taken toward making it publicly available for broader use. The API is designed following REST standards, providing developers with a familiar framework to create, view, edit, and delete PPEM resources.

Future API publication will enable PPEM integration with automation systems, enhancing automated tools for DBMS infrastructure maintenance.

Chapter 3. Quick Start

This section explains how to install PPEM on a server according to the all-in-one deployment scheme. Following the provided instructions, you will test the installation process and get a minimal working version of PPEM suitable for demonstration purposes.

When executing the commands provided in this section, consider the following:

- The Debian Linux operating system commands are specified. For other operating system versions, use their corresponding commands.
- Standard object names are specified in the commands, for example, `ppem` for the [repository](#) database. If required, you can specify different names.

The installation process includes the following steps:

1. Ensure that [prerequisites](#) are met.
2. [Configure the manager](#).
3. [Configure an agent](#).

PPEM will be installed. You can update the browser page with the [web application](#) and start working.

Prerequisites

1. Prepare the server where PPEM will be installed according to [hardware and software requirements](#).
2. Install the Postgres Pro DBMS instance on the server.

For more information about installation, refer to the [official Postgres Pro documentation](#).

3. Start a new session under the superuser:

```
$ sudo -s
```

Configure the Manager

1. Install the [repository](#):

```
# wget -O pgpro-repo-add.sh https://repo.postgrespro.ru/ppem/ppem/keys/pgpro-repo-add.sh
# sh pgpro-repo-add.sh
```

2. Install the [manager](#):

```
# apt install ppem ppem-gui
```

The `ppem-manager.yml` manager configuration file will be downloaded to your local device.

3. Create the DBMS user on behalf of which the manager will connect to the repository database:

```
# sudo -u postgres createuser --pwprompt ppem
```

In this example, the `ppem` user is created. This user will be mentioned in other commands in this section.

When executing this command, specify the DBMS user password.

4. Create the repository database:

```
# sudo -u postgres createdb -O ppem ppem
```

5. Ensure that the DBMS user can connect to the database:

```
# psql -h localhost -U ppem -d ppem
```

In case of connection issues, ensure that the HBA rule allowing the DBMS user to connect to the database, as well as the authorization method, are specified in the `pg_hba.conf` configuration file.

For more information, refer to the official Postgres Pro documentation on the [pg_hba.conf](#) configuration file.

6. In the `ppem-manager.yml` manager configuration file:

- Specify the repository database name using the `repo.name` parameter:

```
repo:
  name: "ppem"
```

- Specify the DBMS user name and password using the `repo.user` and `repo.password` parameters:

```
repo:
  user: "ppem"
  password: "DBMS_user_password"
```

- Specify the URL for connecting the manager to the repository database using the `repo.url` parameter:

```
repo:
  url: "postgres://ppem:DBMS_user_password@localhost/ppem"
```

For more information about the URL format, refer to the official Postgres Pro documentation on [connection strings](#).

7. Start the manager service and add it to the server startup:

```
# systemctl start ppem
# systemctl enable ppem
```

The [web application](#) will be installed on the server.

Configure an Agent

1. Install the [agent](#):

```
# apt install ppem-agent
```

The `ppem-agent.yml` agent configuration file will be downloaded to your local device.

2. Create the DBMS user on behalf of which the agent will connect to the [repository](#) database:

```
# sudo -u postgres createuser -s --pwprompt ppem_agent
```

In this example, the `ppem_agent` user is created. This user will be mentioned in other commands in this section.

When executing this command, specify the DBMS user password.

3. Ensure that the DBMS user can connect to the repository database:

```
# psql -h localhost -U ppem_agent -d ppem
```

In case of connection issues, ensure that the HBA rule allowing the DBMS user to connect to the database, as well as the authorization method, are specified in the `pg_hba.conf` configuration file.

For more information, refer to the official Postgres Pro documentation on the [pg_hba.conf](#) configuration file.

4. Get an API key for configuring the agent:

- a. [Log in to the web application](#).
- b. Copy the API key from the displayed agent installation instruction and save this key.

5. In the `ppem-agent.yml` agent configuration file, specify the agent parameters:

```
agent:
  name: "agent_name"
  manager:
    url: "URL_for_connecting_to_manager"
    api_key: "API_key_for_connecting_to_manager"
  instance:
    connection_defaults:
      user: "DBMS_username"
      password: "DBMS_user_password"
http:
  server:
    address: "network_address_for_incoming_connections"
    port: "port_for_incoming_connections"
```

Where:

- `agent.name`: The unique agent name.
- `agent.manager.url`: The URL for connecting the agent to the [manager](#) in the `scheme://manager_network_address/path_to_API_version` format.
- `agent.manager.api_key`: The previously obtained API key for connecting the agent to the manager.
- `agent.instance.connection_defaults.user` and `agent.instance.connection_defaults.password`: The DBMS user name and password.
- `http.server.address` and `http.server.port`: The network address and port number for incoming network connections.

To enable listening of all network addresses and ports, do not specify any values for these parameters.

6. Start the agent service and add it to the server startup:

```
# systemctl start ppem-agent
# systemctl enable ppem-agent
```

Chapter 4. Installation and Setup

This section explains how to initially install and configure PPEM. It is recommended to see [Quick Start](#) first.

4.1. Installation in Secure Environments

In secure environments, the [manager](#) and [agent](#) services run on servers under separately created operating system users, not under the superuser. In this case, operating system users must be granted additional privileges so that the manager and agent services can function correctly.

Important

If the agent service is running on behalf of an operating system user lacking the required privileges, consider the following limitations:

- When creating [instances](#) and [clusters](#):
 - The main instance data catalog cannot be created. PPEM uses `sudo` and requires the privileges to run `chmod`, `chown`, `initdb`, and `pg_ctl`.
 - Instances cannot be configured. PPEM automatically specifies instance parameters in the `postgresql.auto.conf` configuration file and requires the privilege to access the main instance data catalog.
- When restoring instances from [backups](#):
 - The main instance data catalog cannot be restored. PPEM requires the privilege to access the catalog where the main instance data catalog will be created.
 - External catalogs cannot be restored. PPEM requires the privilege to access the catalogs where the external catalogs will be created.
- The PPEM service cannot be managed and the systemd unit cannot be created. PPEM requires the privilege to access the system path.

When executing the commands provided in this section, consider the following:

- The Debian Linux operating system commands are specified. For other operating system versions, use their corresponding commands.
- Standard object names are specified in the commands, for example, `ppem` for the [repository](#) database. If required, you can specify different names.

The installation process includes the following steps:

1. Ensure that [prerequisites](#) are met.
2. [Create operating system users](#).
3. [Configure the manager](#).
4. [Configure agents](#).

PPEM will be installed. You can update the browser page with the [web application](#) and start working.

Prerequisites

1. Prepare the servers for installing PPEM according to [hardware and software requirements](#).

You will need at least one server.

2. Install a Postgres Pro DBMS instance at least on one of the servers.

For more information about installation, refer to the [official Postgres Pro documentation](#).

Create Operating System Users

Create separate operating system users on all servers:

```
# useradd ppem
```

The [manager](#) and [agent](#) services will start under the created operating system users.

Configure the Manager

On the Postgres Pro DBMS instance server:

1. Install the [repository](#):

```
# wget -O pgpro-repo-add.sh https://repo.postgrespro.ru/ppem/ppem/keys/pgpro-repo-add.sh
# sh pgpro-repo-add.sh
```

2. Install the [manager](#):

```
# apt install ppem ppem-gui
```

The `ppem-manager.yml` manager configuration file will be downloaded to your local device.

3. Create the DBMS user on behalf of which the manager will connect to the repository database:

```
# sudo -u postgres createuser --pwprompt ppem
```

In this example, the `ppem` user is created. This user will be mentioned in other commands in this section.

When executing this command, specify the DBMS user password.

4. Create the repository database:

```
# sudo -u postgres createdb -O ppem ppem
```

5. Ensure that the DBMS user can connect to the database:

```
# psql -h localhost -U ppem -d ppem
```

In case of connection issues, ensure that the HBA rule allowing the DBMS user to connect to the database, as well as the authorization method, are specified in the `pg_hba.conf` configuration file.

For more information, refer to the official Postgres Pro documentation on the [pg_hba.conf](#) configuration file.

6. In the `ppem-manager.yml` manager configuration file:

- Specify the repository database name using the `repo.name` parameter:

```
repo:
  name: "ppem"
```

- Specify the DBMS user name and password using the `repo.user` and `repo.password` parameters:

```
repo:
  user: "ppem"
  password: "DBMS_user_password"
```

- Specify the URL for connecting the manager to the repository database using the `repo.url` parameter:

```
repo:
```

```
url: "postgres://ppem:DBMS_user_password@localhost/ppem"
```

For more information about the URL format, refer to the official Postgres Pro documentation on [connection strings](#).

7. Configure the manager service to start under the [created operating system user](#):

- a. Start editing the systemd unit:

```
# systemctl edit ppem
```

- b. In the `[Service]` section, specify the name of the operating system user:

```
[Service]
User=ppem
```

- c. Ensure that the operating system user is granted the privilege to read the `ppem-manager.yml` manager configuration file.

If the privilege is not granted, execute the following commands:

```
# chown ppem:ppem /etc/ppem-manager.yml
# chmod 400 /etc/ppem-manager.yml
```

- d. Save the systemd unit parameters, then reload it:

```
# systemctl daemon-reload
```

8. Start the manager service and add it to the server startup:

```
# systemctl start ppem
# systemctl enable ppem
```

The [web application](#) will be installed on the server.

Configure Agents

On all servers:

1. Install the [agents](#):

```
# apt install ppem-agent
```

The `ppem-agent.yml` agent configuration file will be downloaded to your local device.

2. Create the DBMS user on behalf of which the agent will connect to the [repository](#) database:

```
# sudo -u postgres createuser --pwprompt ppem_agent
```

In this example, the `ppem_agent` user is created. This user will be mentioned in other commands in this section.

When executing this command, specify the DBMS user password.

3. Grant `ppem_agent` the privilege to read the system catalog and run functions.

It is recommended to grant the following privileges:

```
GRANT pg_monitor, pg_maintain, pg_signal_backend, pg_read_all_settings TO
ppem_agent;
```

Grant `ppem_agent` the following privileges on every database in the instance:

```
GRANT EXECUTE ON FUNCTION pg_catalog.pg_stat_file(TEXT) TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_stat_file(TEXT, BOOLEAN) TO ppem_agent;
GRANT SELECT ON pg_catalog.pg_statistic TO ppem_agent;
GRANT SELECT ON pg_catalog.pg_config TO ppem_agent;
```

```
GRANT EXECUTE ON FUNCTION pg_catalog.pg_config() TO ppem_agent;
GRANT SELECT ON pg_catalog.pg_file_settings TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_show_all_file_settings() TO ppem_agent;
GRANT SELECT ON pg_catalog.pg_authid TO ppem_agent;
```

Grant `ppem_agent` the following privileges to manage backups:

```
ALTER ROLE ppem_agent WITH REPLICATION;
```

```
GRANT USAGE ON SCHEMA pg_catalog TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.current_setting(text) TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.set_config(text, text, boolean) TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_is_in_recovery() TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_backup_start(text, boolean) TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_backup_stop(boolean) TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_create_restore_point(text) TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_switch_wal() TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_last_wal_replay_lsn() TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current() TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current_snapshot() TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_snapshot_xmax(txid_snapshot) TO
  ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_control_checkpoint() TO ppem_agent;
```

You can grant privileges only for the database that will be used for connecting the user to the instance.

For more information about privileges, refer to the official Postgres Pro documentation on [pg_probackup](#).

4. Ensure that the DBMS user can connect to the repository database:

```
# psql -h localhost -U ppem_agent -d ppem
```

In case of connection issues, ensure that the HBA rule allowing the DBMS user to connect to the database, as well as the authorization method, are specified in the `pg_hba.conf` configuration file.

For more information, refer to the official Postgres Pro documentation on the [pg_hba.conf](#) configuration file.

5. Get an API key for configuring the agent:
 - a. [Log in to the web application](#).
 - b. Copy the API key from the displayed agent installation instruction and save this key.
6. In the `ppem-agent.yml` agent configuration file, specify the agent parameters:

```
agent:
  name: "agent_name"
  manager:
    url: "URL_for_connecting_to_manager"
    api_key: "API_key_for_connecting_to_manager"
instance:
  connection_defaults:
    user: "DBMS_username"
    password: "DBMS_user_password"
http:
  server:
    address: "network_address_for_incoming_connections"
    port: "port_for_incoming_connections"
```

Where:

- `agent.name`: The unique agent name.
- `agent.manager.url`: The URL for connecting the agent to the manager in the `scheme://manager_network_address/path_to_API_version` format.
- `agent.manager.api_key`: The previously obtained API key for connecting the agent to the manager.
- `agent.instance.connection_defaults.user` and `agent.instance.connection_defaults.password`: The DBMS user name and password.
- `http.server.address` and `http.server.port`: The network address of the server and the port number for incoming network connections.

To enable listening of all network addresses and ports, do not specify any values for these parameters.

7. Grant the **created operating system user** the superuser privileges to work with target directories using the `chown` and `chmod` commands.

Running, stopping, and reloading instances also require the operating system user to have the following privileges:

```
Cmnd_Alias PG_SYS = \  
    /usr/bin/systemctl status postgresql*.service, \  
    /usr/bin/systemctl stop postgresql*.service, \  
    /usr/bin/systemctl start postgresql*.service, \  
    /usr/bin/systemctl restart postgresql*.service, \  
    /usr/bin/systemctl reload postgresql*.service
```

```
Cmnd_Alias PG_CTL = \  
    /usr/lib/postgresql/17/bin/pg_ctl, \  
    /usr/lib/postgresql/16/bin/pg_ctl
```

```
ppem ALL = (root) NOPASSWD: PG_SYS  
ppem ALL = (postgres) NOPASSWD: PG_CTL
```

8. Configure the agent service to start on behalf of the operating system user:

- a. Start editing the systemd unit:

```
# systemctl edit ppem-agent
```

- b. In the `[Service]` section, specify the operating system user:

```
[Service]  
User=ppem
```

- c. Ensure that the operating system user is granted the privilege to read the `ppem-agent.yml` agent configuration file.

If the privilege is not granted:

```
# chown ppem:ppem /etc/ppem-agent.yml  
# chmod 400 /etc/ppem-agent.yml
```

- d. Save the systemd unit parameters, then reload it:

```
# systemctl daemon-reload
```

9. Start the agent service and add it to the server startup:

```
# systemctl start ppem-agent  
# systemctl enable ppem-agent
```

4.2. Installation in the High Availability Mode

This section provides an example of installing and setting up PPEM in the high availability mode. The following software is used in the example:

- Debian Linux 12 OS
- HAProxy 2.6.12 (included in the Debian repository)
- keepalived 2.2.7 (included in the Debian repository)
- Postgres Pro Enterprise 17.2.2 + BiHA (included in the `postgrespro-ent-17-contrib` package)

The recommended high availability PPEM architecture includes the following components:

- High availability cluster based on the [BiHA solution](#) available in Postgres Pro Enterprise 16 and higher.

BiHA cluster consists of three or more nodes. One of the nodes acts as the leader. The [manager](#) automatically connects to the leader. If the leader fails, one of the remaining nodes automatically becomes the new leader.

- Cluster of HAProxy servers + keepalived.

A virtual IP address is automatically activated on one of the servers using the keepalived service. Users and [agents](#) interact with managers using this virtual IP address. If the server with the virtual IP address fails, the virtual IP address is activated on one of the remaining servers.

HAProxy also balances client HTTP requests between available managers. For sending requests of a specific client to the same manager, the HAProxy IP-based stickiness or cookie session stickiness functionality is used.

You can deploy this component on separate servers and managers.

- Manager.

Two or more servers in the active mode.

The components required for implementation of the recommended architecture are described in the table below.

Component	Amount	Minimum Requirements
Manager server + HAProxy + keepalived	2	2 CPU, 4 GB RAM, 20 GB HDD.
Postgres Pro BiHA server	3	2 CPU, 4 GB RAM, 20 GB HDD.
Virtual static IP address	1	The IP address must be excluded from the DHCP pool. You can also create a DNS A record for this IP address.

To install PPEM in the high availability mode:

1. [Install Postgres Pro Enterprise with the BiHA solution.](#)
2. To provide PPEM servers with access to DBMS, edit the `pg_hba.conf` configuration file, for example:

```
# cat /var/lib/pgpro/ent-17/data/pg_hba.conf

host      all      all      192.168.1.0/24      scram-sha-256
```

3. [Configure the manager:](#)

```
# cat /etc/ppem-manager.yml
http:
  server:
```

```
    address: "network_address_for_incoming_connections"
    port: "port_for_incoming_connections"
repo:
  url: postgres://ppem:DBMS_user_password@biha-server-1/ppem
  fallback_addresses:
    - biha_server_2
    - biha_server_3
  target_session_attrs: read-write
```

Where:

- `http.server.address` and `http.server.port`: The network address of the server and the port number for incoming network connections.

In terms of HAProxy these are backend parameters.

- `repo.url`: The network address of the first BiHA cluster node.
- `fallback_addresses`: The network addresses of the remaining BiHA cluster nodes.
- `target_session_attrs`: The session attribute condition that determines when the manager can automatically connect to the BiHA cluster leader.

Specify `read-write`.

4. On HAProxy servers, install `haproxy`, `keepalived`, and the required tools:

```
sudo apt-get install haproxy keepalived psmisc
```

5. Configure HAProxy using the `haproxy.cfg` configuration file, for example:

```
global
  log /dev/log local0
  log /dev/log local1 notice
  stats socket /var/lib/haproxy/stats level admin
  chroot /var/lib/haproxy
  user haproxy
  group haproxy
  daemon

defaults
  log global
  mode http
  option httplog
  option dontlognull
  timeout connect 5000
  timeout client 50000
  timeout server 50000
  errorfile 400 /etc/haproxy/errors/400.http
  errorfile 403 /etc/haproxy/errors/403.http
  errorfile 408 /etc/haproxy/errors/408.http
  errorfile 500 /etc/haproxy/errors/500.http
  errorfile 502 /etc/haproxy/errors/502.http
  errorfile 503 /etc/haproxy/errors/503.http
  errorfile 504 /etc/haproxy/errors/504.http

frontend hafrontend
  bind *:80
  mode http
  default_backend habackend

backend habackend
```

```
mode http
balance roundrobin
option forwardfor
option httpchk
http-check send meth HEAD uri /
cookie SERVERID insert indirect
server ppem-server-1 PPEM_server_address-1:8080 cookie ppem-server-1 check
server ppem-server-2 PPEM_server_address-2:8080 cookie ppem-server-2 check
```

The cookie-based persistence method is used in the example. It assigns a cookie with the PPEM server name to users. This is required for sending all queries within the HTTP session to a single manager server.

6. Configure the HAProxy-1 server using the `keepalived.conf` configuration file:

```
global_defs {
    enable_script_security
}

vrrp_script chk_haproxy {
    script "/usr/bin/killall -0 haproxy"
    interval 3
    fall 2
    rise 3
    timeout 3
    user root
}

vrrp_instance internal {
    interface interface
    state MASTER
    virtual_router_id 124
    priority 100

    unicast_src_ip HAproxy-1_server_IP_address
    unicast_peer {
        HAproxy-2_server_IP_address
    }
    virtual_ipaddress {
        virtual_IP_address/subnet_class_(for_example_16_or_24) dev interface
    }
    track_script {
        chk_haproxy
    }
}
```

The configuration is the same for the HAProxy-2 server, but the IP addresses in the `unicast_src_ip` and `unicast_peer` parameters are switched.

7. On all BiHA servers, [configure agents](#).

The connection to the manager must be configured via a virtual IP address.

8. Perform healthcheck by ensuring that:

- The BiHA cluster database is available on all PPEM servers.
- The `ppem` system server is running on all PPEM servers:

```
systemctl status ppem
```

- The [web application](#) is available on all PPEM servers:

- on port 8080:

```
curl http://PPEM_server_IP_address:8080
```

- on port 80 via HAProxy:

```
curl http://PPEM_server_IP_address:80
```

- via a virtual IP address:

```
curl http://virtual_IP_address:80
```

- The HAProxy and keepalived services are running on all PPEM servers:

```
systemctl status haproxy
systemctl status keepalived
```

4.3. Installation and Configuration of Backup and Restore Tools

PPEM performs [backup](#) using `pg_probackup` manually installed on all servers. The version of `pg_probackup` must correspond to the version of DBMS instances. The backup features available in PPEM depend on the `pg_probackup` edition.

For more information about installation, refer to the official Postgres Pro documentation on [pg_probackup](#). It is also recommended to view the [example](#) of the `pg_probackup` installation using APT (for Debian-based operating systems).

After `pg_probackup` is installed, the [agent](#) automatically detects it and notifies the [manager](#). The server is now ready for backup operations.

You must manually configure the installed `pg_probackup`. The configuration process includes the following steps on all servers:

1. Create dedicated DBMS users and grant them backup privileges using `pg_probackup`.

For more information, refer to the official Postgres Pro documentation on [configuring the database cluster](#).

2. Configure a STREAM backup.

For more information, refer to the official Postgres Pro documentation on [setting up STREAM backups](#).

3. To provide point-in-time recovery (PITR), configure continuous WAL archiving in one of the following ways:

- When configuring an instance on a server as described in the [official Postgres Pro documentation](#).
- [When configuring an instance in the web application](#).

Example of the `pg_probackup` Installation Using APT

This section includes an example of `pg_probackup` installation on a server with a DBMS instance and PPEM.

The version of `pg_probackup` must correspond to the version of the instance. Version 16 is used in this example.

The installation process includes the following steps:

1. Start a new session under the superuser:

```
$ sudo -s
```

2. To add the GPG key for the `pg_probackup` repository, install additional utilities:

```
# apt install gpg wget
# wget -qO - https://repo.postgrespro.ru/pg_probackup/keys/GPG-KEY-PG-PROBACKUP | \
# tee /etc/apt/trusted.gpg.d/pg_probackup.asc
```

3. Configure the package repository:

```
# . /etc/os-release
# echo "deb [arch=amd64] https://repo.postgrespro.ru/pg_probackup/deb
$VERSION_CODENAME main-$VERSION_CODENAME " | tee /etc/apt/sources.list.d/
pg_probackup.list
```

4. To make `pg_probackup` packages available for viewing and installing, update the package manager metadata:

```
# apt update
# apt search pg_probackup
```

5. Install `pg_probackup`:

```
# apt install pg-probackup-16
```

After `pg_probackup` is installed, the [agent](#) automatically detects it and notifies the [manager](#). The server is now ready for backup operations.

4.4. Installation and Configuration of Logging and Monitoring Tools

Logging of DBMS instances and working with their metrics is done in PPEM using `pgpro-otel-collector` manually installed on all servers.

For more information, refer to the official Postgres Pro documentation on [installing pgpro-otel-collector](#).

You must manually configure the installed `pgpro-otel-collector`. The configuration process includes the following steps:

1. Configure instance logging.

Ensure that values are specified for the following configuration parameters:

- `logging_collector`
- `log_destination`
- `log_directory`
- `log_filename`

Instance logging must be done in the CSV or JSON format.

For more information, refer to the official Postgres Pro documentation on [server configuration](#).

2. Configure `pgpro-otel-collector` to collect instance logs and metrics.

For more information, refer to the official Postgres Pro documentation on configuring [logs](#) and [metrics](#) for `pgpro-otel-collector`.

3. Configure `pgpro-otel-collector` to send instance logs and metrics to PPEM or an external storage.

For more information, refer to the official Postgres Pro documentation on [integrating pgpro-otel-collector with PPEM](#).

Important

Local storages are intended for introductory use with PPEM. For production environments with a large number of instances and metrics, it is recommended to use external storages.

4.5. Integration with External Data Sources

This section explains how to integrate PPEM with Prometheus, as well as with Elasticsearch and Application Performance Monitoring (APM).

Fictional service names are used in the configuration examples:

- `example.org` is the main domain.
- `prometheus.example.org` is the Prometheus monitoring system service.
- `elasticsearch.example.org` is the Elasticsearch monitoring system service.
- `elasticsearch-apm.example.org` is the APM Elasticsearch monitoring system service.
- `postgresql-01.example.org` is the PostgreSQL DBMS service.

For more information about the monitoring architecture, refer to [Monitoring Architecture](#).

4.5.1. Integration with Prometheus

Integration with external Prometheus data sources is used for reading metrics written by [pgpro-otel-collector](#).

Note

You can use VictoriaMetrics instead of Prometheus, since these solutions use similar interfaces for reading and writing metrics.

The components below are required for integration.

pgpro-otel-collector

The monitoring agent performing the following functions:

- collecting statistics from Postgres Pro DBMS instances and converting them to metrics
- publishing metrics for further collection by the Prometheus monitoring system

Prometheus

The monitoring system performing the following functions:

- collecting metrics from pgpro-otel-collector monitoring agents
- storing metrics from monitoring agents according to the internal configuration parameters
- providing the HTTP interface for receiving metrics

PPEM

The Postgres Pro Enterprise Manager system performing the following functions:

- accessing the Prometheus monitoring system for receiving DBMS instance metrics
- providing the user with the monitoring interface in the form of graphs

The integration process includes the following steps:

1. [Configure pgpro-otel-collector for Prometheus.](#)
2. [Configure Prometheus.](#)
3. [Check for metrics in Prometheus.](#)
4. [Configure a metrics data source.](#)
5. [Check the operation of a metrics storage.](#)

Additional configuration of the [agent](#) is not required.

Configure pgpro-otel-collector for Prometheus

1. Enable and configure the `postgrespro` and `hostmetrics` receivers:

```
receivers:
  hostmetrics:
    initial_delay: 1s
    collection_interval: 60s
    scrapers:
      cpu:
        metrics:
          system.cpu.utilization:
            enabled: true
      disk: null
      filesystem: null
      load: null
      memory: null
      network: null
      paging: null
      processes: null
  postgrespro:
    max_threads: 3
    initial_delay: 1s
    collection_interval: 60s
    transport: tcp
    endpoint: localhost:5432
    database: postgres
    username: postgres
    password: ${env:POSTGRES_PASSWORD}
    plugins:
      activity:
        enabled: true
      archiver:
        enabled: true
      bgwriter:
        enabled: true
      cache:
        enabled: true
      databases:
        enabled: true
      io:
        enabled: true
      locks:
        enabled: true
      version:
        enabled: true
      wal:
```

```
enabled: true
```

2. Configure metrics publishing using `prometheusexporter` and the pipeline.

PPeM expects the metrics sent by `pgpro-otel-collector` to have the instance label with the node FQDN and DBMS instance port number separated by a colon, such as `postgresql_activity_connections{instance="postgresql-01.example.org:5432"}`.

Configuration example:

```
exporters:
  prometheus:
    const_labels:
      instance: postgresql-01.example.org:5432
    endpoint: :8889
    send_timestamps: true

service:
  extensions: []
  pipelines:
    metrics:
      exporters:
        - prometheus
      receivers:
        - postgrespro
        - hostmetrics
```

3. Start the collector and ensure that metrics are published on its side:

```
# systemctl status pgpro-otel-collector

# systemctl status pgpro-otel-collector
● pgpro-otel-collector.service - PostgresPro OpenTelemetry Collector
   Loaded: loaded (/lib/systemd/system/pgpro-otel-collector.service; enabled;
   preset: enabled)
   Active: active (running) since Thu 2025-03-20 01:18:08 MSK; 4h 13min ago
   Main PID: 6991 (pgpro-otel-coll)
     Tasks: 8 (limit: 3512)
    Memory: 119.3M
       CPU: 2min 49.311s
    CGroup: /system.slice/pgpro-otel-collector.service
            └─6991 /usr/bin/pgpro-otel-collector --config /etc/pgpro-otel-
collector/basic.yml

Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.366656,"msg":"Setting up own telemetry..."}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.367178,"msg":"Skipped telemetry setup."}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.3679142,"msg":"Development component. May change
in the future.","kind":"receiver","name":"postgrespro","data_type":"metrics"}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"warn","ts":1742422688.3494158,"caller":"envprovider@v1.16.0/
provider.go:59","msg":"Configuration references unset environment
variable","name":"POSTGRESQL_P"}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.4481084,"msg":"Starting pgpro-otel-
collector...","Version":"v0.3.1","NumCPU":1}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.4481149,"msg":"Starting extensions..."}
```

```
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"warn","ts":1742422688.4483361,"msg":"Using the 0.0.0.0 address exposes
this server to every network interface, which may facilitate Denial of Service
attack>}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.4515307,"msg":"Starting stanza
receiver","kind":"receiver","name":"filelog","data_type":"logs"}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.451749,"msg":"Everything is ready. Begin running
and processing data."}

curl -s 127.0.0.1:8889/metrics |grep -c postgres
4254
```

Configure Prometheus

You can configure collection of metrics using pgpro-otel-collector on the side of Prometheus in different ways. One of the ways is collecting using the static configuration:

```
- job_name: pgpro-otel-collector
  static_configs:
    targets:
      - postgresql-01.example.org:8889/metrics
```

For more information about other collection methods, refer to the [official Prometheus documentation](#).

PPEM does not require additional Prometheus configuration.

Check for Metrics in Prometheus

After configuring metrics collection using pgpro-otel-collector, ensure that metrics are received by the monitoring system.

For this check, you can use the built-in [expression browser](#) graphical tool or the [promtool](#) utility from Prometheus.

Example check using the promtool utility:

```
promtool query instant https://prometheus.example.org
'postgresql_activity_connections{instance="postgresql-01.example.org:5432"}'
```

Where:

- `https://prometheus.example.org`: The URL of the monitoring service.
- `postgresql_activity_connections{instance="postgresql-01.example.org:5432"}`: The name of the metric.

Response example:

```
postgresql_activity_connections{database="postgres",
instance="postgresql-01.example.org:5432", job="pgpro-otel-collector", state="active",
user="postgres"} 5
postgresql_activity_connections{database="postgres",
instance="postgresql-01.example.org:5432", job="pgpro-otel-collector", state="idle",
user="postgres"} 10
```

Configure a Metrics Data Source

1. In the navigation panel, go to Infrastructure → Data sources → Metrics storages.
2. In the top-right corner of the page, click Create storage.
3. Specify the metrics storage parameters (parameters marked with an asterisk are required):
 - Name: The unique name of the metrics storage. For example, Prometheus.

- **URL:** The network address for connecting to the metrics storage. For example, `https://prometheus.example.org/select/0/prometheus`.
- **User:** The unique name of the user if authorization is used.
- **Password:** The password of the user if the authorization is enabled.
- **Description:** The description of the metrics storage.
- **Make default datasource:** Specifies whether the metrics storage is used by default for all metric queries.

4. Click Save.

Check the Operation of a Metrics Storage

1. In the navigation panel, go to Monitoring → Metrics.
2. In the top-right corner of the page, select the instance for which there are metrics in the storage.
3. Change the default data source to the internal one and ensure that graphs are displayed without errors.

4.5.2. Integration with Elasticsearch and APM

Integration with external Elasticsearch data sources is used for reading logs written by [pgpro-otel-collector](#).

The components below are required for integration.

pgpro-otel-collector

The monitoring agent performing the following functions:

- collecting activity logs from Postgres Pro DBMS instances
- sending activity logs to APM Elasticsearch

APM Elasticsearch

The application performance monitoring system based on Elastic Stack performing the following functions:

- receiving data from the monitoring agent and converts it to the ES document format
- sending converted data to Elasticsearch

Elasticsearch

The activity log storage system performing the following functions:

- receiving activity logs from the application performance monitoring system
- storing activity logs according to internal storage parameters
- providing the interface for receiving activity logs

PPEM

The Postgres Pro Enterprise Manager system performing the following functions:

- accessing Elasticsearch for receiving DBMS instance activity logs
- providing the user with the monitoring interface based on activity logs in the form of text data

The integration process includes the following steps:

1. [Configure Elastisearch](#).
2. [Configure pgpro-otel-collector for Elastisearch](#).
3. [Check logs in Elastisearch](#).
4. [Configure a log data source](#).
5. [Check the operation of the log storage](#).

Additional configuration of the [agent](#) is not required.

Configure Elasticsearch

1. Install the Elasticsearch APM server using the standard documentation.
2. Integrate the Elasticsearch APM server with Elasticsearch using the standard documentation.
3. Configure the pgpro-otel-collector ingest pipeline.

This is required for compatibility between document (log) fields and the Elasticsearch Common Schema (ECS) field naming schema.

Pipeline configuration example (both queries are executed sequentially in Kibana Developer Tools):

```
PUT _ingest/pipeline/postgrespro-otelcol-enrich-logs
{
  "description": "Enrich PostgresPro Otel collector logs",
  "processors": [
    {
      "rename": {
        "if": "ctx?.labels?.message != null",
        "field": "labels.message",
        "target_field": "message",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "rename": {
        "if": "ctx?.labels?.pid != null",
        "field": "labels.pid",
        "target_field": "process.pid",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "rename": {
        "if": "ctx?.labels?.error_severity != null",
        "field": "labels.error_severity",
        "target_field": "log.level",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "rename": {
        "if": "ctx?.labels?.user != null",
        "field": "labels.user",
        "target_field": "user.name",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "rename": {
        "if": "ctx?.labels?.session_start != null",
```

```

    "field": "labels.session_start",
    "target_field": "session.start_time",
    "ignore_failure": true,
    "ignore_missing": false,
    "override": true
  }
},
{
  "rename": {
    "if": "ctx?.labels?.session_id != null",
    "field": "labels.session_id",
    "target_field": "session.id",
    "ignore_failure": true,
    "ignore_missing": false,
    "override": true
  }
},
{
  "rename": {
    "if": "ctx?.numeric_labels?.tx_id != null",
    "field": "numeric_labels.tx_id",
    "target_field": "transaction.id",
    "ignore_failure": true,
    "ignore_missing": false,
    "override": true
  }
},
{
  "rename": {
    "if": "ctx?.labels?.log_file_name != null",
    "field": "labels.log_file_name",
    "target_field": "log.file.path",
    "ignore_failure": true,
    "ignore_missing": false,
    "override": true
  }
},
{
  "rename": {
    "if": "ctx?.labels?.dbname != null",
    "field": "labels.dbname",
    "target_field": "db.name",
    "ignore_failure": true,
    "ignore_missing": false,
    "override": true
  }
},
{
  "gsub": {
    "if": "ctx?.service?.node?.name != null",
    "field": "service.node.name",
    "target_field": "host.name",
    "pattern": ":+$",
    "replacement": "",
    "ignore_failure": true,
    "ignore_missing": false
  }
},

```

```

    {
      "remove": {
        "field": [
          "observer.version",
          "observer.hostname",
          "service.language.name"
        ],
        "ignore_failure": true
      }
    },
    {
      "remove": {
        "field": "agent.version",
        "if": "ctx?.agent?.version == \"unknown\"",
        "ignore_failure": true
      }
    }
  ]
}
PUT _ingest/pipeline/logs-apm.app@custom
{
  "processors": [
    {
      "pipeline": {
        "name": "postgrespro-otelcol-enrich-logs"
      }
    }
  ]
}

```

Configure pgpro-otel-collector for Elasticsearch

1. Enable and configure the filelog receiver.

Receiver configuration example for the scenario when PostgreSQL logs are generated in the JSON format:

```

receivers:
  filelog:
    include:
      - /var/log/postgresql/*.json
    operators:
      - parse_ints: true
        timestamp:
          layout: '%Y-%m-%d %H:%M:%S.%L %Z'
          layout_type: strptime
          parse_from: attributes.timestamp
        type: json_parser
      - field: attributes.timestamp
        type: remove
    retry_on_failure:
      enabled: true
      initial_interval: 1s
      max_elapsed_time: 5m
      max_interval: 30s
    start_at: end

```

2. Configure processors:

```

processors:

```

```
attributes/convert:
  actions:
    - action: convert
      converted_type: string
      key: query_id
    - action: convert
      converted_type: string
      key: pid
resource:
  attributes:
    - action: upsert
      key: service.name
      value: postgresql
    - action: upsert
      key: service.instance.id
      value: postgresql-01.example.org:5432
```

Where:

- `service.name` is the key for naming the data stream (data stream) and, consequently, indexes.
- `service.instance.id` is the key for identifying the instance.
- For logs in the JSON format, converting `query_id` to a string is required because integers are displayed incorrectly in ES.

Important

Data streams are used for storing data. The target stream is selected automatically and has the `logs-apm.app.service.name-namespace` format.

The `service.name` value is specified in the collector configuration, in the `processors.resource.attributes` list, by the `key: service.name` element.

The `namespace` value is defined by the element with the `service.environment` key. It is not sent in this configuration so the default value is entered by default. If this configuration is used, activity logs will be stored in the stream named `logs-apm.app.postgresql-default`.

3. Configure logs sending using `otlhttpexporter` and the pipeline:

```
exporters:
  otlhttp/elastic_logs:
    compression: gzip
    endpoint: https://elasticsearch-apm.example.org
    tls:
      insecure_skip_verify: false

service:
  extensions: []
  pipelines:
    logs:
      receivers:
        - filelog
      processors:
        - resource
        - attributes/convert
      exporters:
        - otlhttp/elastic_logs
```

4. Start the collector and ensure that metrics are published on its side:

```
# systemctl status pgpro-otel-collector

# systemctl status pgpro-otel-collector
● pgpro-otel-collector.service - PostgresPro OpenTelemetry Collector
   Loaded: loaded (/lib/systemd/system/pgpro-otel-collector.service; enabled;
   preset: enabled)
   Active: active (running) since Thu 2025-03-20 01:18:08 MSK; 4h 13min ago
   Main PID: 6991 (pgpro-otel-coll)
     Tasks: 8 (limit: 3512)
    Memory: 119.3M
       CPU: 2min 49.311s
    CGroup: /system.slice/pgpro-otel-collector.service
            └─6991 /usr/bin/pgpro-otel-collector --config /etc/pgpro-otel-
collector/basic.yml

Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.366656,"msg":"Setting up own telemetry..."}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.367178,"msg":"Skipped telemetry setup."}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.3679142,"msg":"Development component. May change
in the future.","kind":"receiver","name":"postgrespro","data_type":"metrics"}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"warn","ts":1742422688.3494158,"caller":"envprovider@v1.16.0/
provider.go:59","msg":"Configuration references unset environment
variable","name":"POSTGRESQL_P"}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.4481084,"msg":"Starting pgpro-otel-
collector...","Version":"v0.3.1","NumCPU":1}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.4481149,"msg":"Starting extensions..."}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"warn","ts":1742422688.4483361,"msg":"Using the 0.0.0.0 address exposes
this server to every network interface, which may facilitate Denial of Service
attack"}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.4515307,"msg":"Starting stanza
receiver","kind":"receiver","name":"filelog","data_type":"logs"}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.451749,"msg":"Everything is ready. Begin running
and processing data."}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]:
{"level":"info","ts":1742422688.6523068,"msg":"Started watching
file","kind":"receiver","name":"filelog","data_type":"logs","component":"fileconsumer",
var/log/postgresql/postgresql-2025-03-20.json"}
```

Check Logs in Elasticsearch

After configuring logs sending from pgpro-otel-collector in Elasticsearch, ensure that metrics are received by the log storage system.

For this check, you can execute a query to the storage using the curl utility.

Query example:

```
curl -s -XGET "https://elasticsearch.example.org:9200/logs-apm.app.postgresql-default/_search?size=10" -H 'Content-Type: application/json' -d'
{
```

```
"_source": ["message", "service.node.name", "@timestamp"],
"sort": [
  { "@timestamp": "desc" }
],
"query": {
  "bool": {
    "filter": [
      { "term": {"service.node.name": "postgresql-01.example.org:5432" } } ]
  }
}
}'
```

Where:

- `https://elasticsearch.example.org:9200` is the URL of the log storage system.
- `logs-apm.app.postgresql-default` is the name of the data stream for the search.
- `size=10` is the size of the sample.
- `"_source": ["message", "service.node.name", "@timestamp"]` are requested fields.

Response example:

```
{
  "took": 18,
  "timed_out": false,
  "_shards": {
    "total": 11,
    "successful": 11,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 10000,
      "relation": "gte"
    },
    "max_score": null,
    "hits": [
      {
        "_index": ".ds-logs-apm.app.postgresql-default-2025.03.19-000379",
        "_id": "qmuArJUB2PKtie47Rffa",
        "_score": null,
        "_source": {
          "message": "checkpoint complete: wrote 2038 buffers (16.6%); 0 WAL file(s)
added, 0 removed, 10 recycled; write=269.563 s, sync=1.192 s, total=270.962 s; sync
files=246, longest=0.677 s, average=0.005 s; distance=162419 kB, estimate=174180 kB;
lsn=6/62000850, redo lsn=6/583C4DD8",
          "@timestamp": "2025-03-19T03:44:01.336Z",
          "service": {
            "node": {
              "name": "postgresql-01.example.org:5432"
            }
          }
        },
        "sort": [
          1742355841336
        ]
      }
    ]
  }
}
```

```
}  
}
```

Configure a Log Data Source

1. In the navigation panel, go to Infrastructure → Data sources → Message storages.
2. In the top-right corner of the page, click Create storage.
3. Specify the log storage parameters (parameters marked with an asterisk are required):
 - Name: The unique name of the log storage. For example, `Elasticsearch`.
 - URL: The network address for connecting to the log storage. For example, `https://elastic-search.example.org`.
 - Elasticsearch index: The name of the index (stream) for search queries.
Specify `logs-apm.app.postgresql-default`.
 - User: The unique name of the user if authorization is used.
 - Password: The password of the user if the authorization is enabled.
 - Description: The description of the log storage.
 - Make default datasource: Specifies whether the log storage is used by default for all metric queries.

Check the Operation of the Log Storage

In the navigation panel, go to Monitoring → Message journal.

4.6. Integration with OpenLDAP and Active Directory

PPEM supports authentication using the directory services OpenLDAP and Active Directory. Within the integration, roles are assigned to LDAP users in the [web application](#) using directory service user groups.

For more information about authentication and authorization, refer to [Authentication and Authorization Operation Scheme](#).

The integration process includes the following steps:

1. [Create a user group in a directory service](#).
2. [Configure integration with a directory service](#).
3. [Configure authorization in PPEM](#).
4. [Check authentication and authorization in PPEM](#) and [troubleshoot authentication](#) if required.

Create a User Group in a Directory Service

[Create a user group in OpenLDAP](#) or Active Directory. For Active Directory, two ways of creating a user group are supported:

- [Create a User Group in Active Directory using ADUC](#)
- [Creating a User Group in Active Directory Using PowerShell](#)

Important

Users added to groups must be created in OpenLDAP or Active Directory with the following configuration parameters required for PPEM:

- `first_name`: The first name of the user.
- `last_name`: The last name of the user.
- `email`: The email address of the user.
- `login`: The login of the user.
- `password`: The password of the user.

You must specify a value in the `string` format for each parameter.

For more information about creating user groups, refer to the [official OpenLDAP documentation](#) or to the official Microsoft documentation on [Active Directory](#).

Create a User Group in OpenLDAP

1. Create a user group configuration file in the LDIF format and specify the following parameters:

```
dn: user_group_DN
objectClass: posixGroup
cn: user_group_name
gidNumber: user_group_ID
memberUid: user_ID
```

Where:

- `dn`: The distinguished name (DN) of the user group.

You can specify the following attributes:

- `cn`: The unique name of the user group.
- `ou`: The organizational unit where the user group will be located.
- `dc`: The components of the domain that will be associated with the user group. For example, `example.com`.
- `cn`: The unique name of the user group.
- `gidNumber`: The unique ID of the user group.
- `memberUid`: The unique IDs of users that will be added to the group.

2. Create a user group in OpenLDAP:

```
ldapadd -x -D "cn=admin,dc=example,dc=com" -W -f example-group.ldif
```

Where:

- `-D`: The distinguished name of the OpenLDAP administrator.
- `-W`: The password request.
- `-f`: The name of the user group configuration file.

3. Ensure that the user group was created successfully:

```
ldapsearch -x -b "ou=groups,dc=example,dc=com" "(cn=examplegroup)"
```

Create a User Group in Active Directory using ADUC

1. In the Active Directory GUI, go to the organizational unit (OU) where the user group will be located.
2. Click Create → Group.
3. In the opened window, specify the user group parameters.
4. Click OK.
5. Add users to the group:
 - a. Go to the user group properties by double-clicking it.
 - b. Select Member Of and add users to the group.

Create a User Group in Active Directory Using PowerShell

1. Create a user group:

```
New-ADGroup -Path "OU=Groups,OU=Example,DC=example,DC=com" -Name "GROUPMSAD" -
GroupScope Global -GroupCategory Distribution
```

Where:

- `-Path`: The distinguished name (DN) of the user group.

You can specify the following attributes:

- `OU`: The organizational unit where the user group will be located.
- `DC`: The components of the domain that will be associated with the user group.
- `-Name`: The unique name of the user group.

2. Add users to the group:

```
Add -ADGroupMember GROUPMSAD -Members user1,user2,user3
```

Where:

- `-ADGroupMember`: The unique name of the group to which users will be added.
- `-Members`: The unique IDs of users that will be added to the group.

Configure Integration with a Directory Service

1. In the `ppem-manager.yml` manager configuration file, add the `ldap` section and specify integration parameters:

- **OpenLDAP:**

```
ldap:
  type: directory_service_type
  url: directory_service_network_address
  use_ssl: true or false
  base_dn: directory_service_base_DN
  bind_username: directory_service_username
  bind_password: directory_service_user_password
  group_class: user_group_object_class
  group_members_attr: user_group_member_attribute
  group_name_attr: user_group_name_attribute
  prefix_group_dn: DN_prefix_for_user_groups
  prefix_user_dn: DN_prefix_for_users
  user_class: user_object_class
  user_display_name_attr: user_display_name_attribute
  user_email_attr: user_email_attribute
  user_name_attr: user_login_attribute
  user_first_name_attr: user_first_name_attribute
  user_last_name_attr: user_last_name_attribute
  user_job_title_attr: user_job_title_attribute
  user_membership_attr: user_group_membership_attribute
  user_phone_attr: user_phone_attribute
  user_sync_interval: manager_and_directory_service_synchronization_time
```

- **Active Directory:**

```
ldap:
  type: directory_service_type
  url: directory_service_network_address
  base_dn: directory_service_base_DN
  bind_username: directory_service_username
  bind_password: directory_service_user_password
  user_sync_interval: manager_and_directory_service_synchronization_time
```

Where:

- `type`: The type of the directory service.

Possible values:

- `openldap`
- `ms_active_directory`
- `url`: The network address of the directory service.
- `bind_username`: The name of the directory service user for integration with PPEM.

The value format depends on the directory service:

- For OpenLDAP, a complete distinguished name (DN) is usually specified. For example, `cn=admin,ou=users,dc=example,dc=com`.
- For Active Directory, a value in the `username@domain` format is usually specified. For example, `admin@example.com`.
- `bind_password`: The password of the directory service user for integrating with PPEM.
- `base_dn`: The base distinguished name of the directory service.
- `prefix_user_dn`: The distinguished name prefix for users.

If this parameter is specified, users are searched using the `prefix_user_dn,base_dn` distinguished name. To search for users in the entire directory, specify `"`.

Optional parameter.

- `prefix_group_dn`: The distinguished name prefix for user groups.

If this parameter is specified, user groups are searched using the `prefix_group_dn,base_dn` distinguished name. To search for user groups in the entire directory, specify `"`.

Optional parameter.

- `user_class`: The name of the user object class.

Optional parameter for Active Directory.

- `user_name_attr`: The name of the user login attribute.

Default value: `cn` for OpenLDAP, `sAMAccountName` for Active Directory.

Optional parameter for Active Directory.

- `user_first_name_attr`: The name of the user first name attribute.

Default value: `givenName`.

Optional parameter.

- `user_last_name_attr`: The name of the user last name attribute.

Default value: `sn`.

Optional parameter.

- `user_display_name_attr`: The name of the user display name attribute.

Default value: `displayName`.

Optional parameter.

- `user_email_attr`: The name of the user email address attribute.

Default value: mail.

Optional parameter.

- `user_phone_attr`: The name of the user phone number attribute.

Default value: `telephoneNumber`.

Optional parameter.

- `user_job_title_attr`: The name of the user job title attribute.

Default value: `title`.

Optional parameter.

- `user_membership_attr`: The name of the user group membership attribute.

Default value for Active Directory: `memberOf`.

Optional parameter.

- `group_class`: The name of the user group object class.

Default value for Active Directory: `group`.

- `group_name_attr`: The name of the user group name attribute.

Default value: `cn`.

Optional parameter.

- `group_members_attr`: The name of the group member attribute.

- `group_filter`: The filter for searching user groups. For example, `(&(objectClass=group)(cn=*PPEM*))`.

Optional parameter.

- `group_membership_filter`: The filter for searching groups of which the specified user is a member. For example, `(&(objectClass=group)(uniqueMember=%USER_DN%))`.

Optional parameter.

- `group_list_size_limit`: The maximum number of user groups that can be received from the directory service.

Optional parameter.

- `user_sync_interval`: Synchronization time between the manager and directory service.

Default value: 5m.

Optional parameter.

- `ssl_cert_skip_verify`: Specifies whether verification of the directory service server certificate is skipped.

Possible values:

- `true`
- `false`

Optional parameter.

- `ssl_root_ca`: The path to the file in the PEM format with the CA certificate on the directory service server.

Optional parameter.

2. Restart the PPEM service:

```
systemctl restart ppem.service
```

Configure Authorization in PPEM

1. [Log in to the web application](#) under a user with the `System administrator` role.

For more information about roles, refer to [User Roles and Privileges](#).

2. [Create a user group](#).

In doing so, from LDAP group, select the LDAP group for which you want to configure authorization. If the LDAP group is not displayed, ensure that you [configured integration with the directory service](#) correctly.

Roles assigned to a user group at its creation will be automatically assigned to users from the specified LDAP group when they log in to the [web application](#).

Checking Authentication and Authorization in PPEM

1. [Log in to the web application](#) under a directory service user. The user must be a member of the LDAP group specified when [configuring authorization in PPEM](#).

The login format depends on the directory service:

- For OpenLDAP, a short user login is usually specified. For example, `j.doe`.
- For Active Directory, a login in the `username@domain` format is usually specified. For example, `j.doe@example.com`.

2. If authentication is successful, check authorization in one of the following ways:

- In the top-right corner of the page, click the username.

The personal account will open. It displays the email address, job title, and assigned roles of the user.

- If you logged in to the web application under a user with the `System administrator` role, in the navigation panel, go to Users.

The table of users will be displayed. Distinguished names (DNs) of LDAP users are displayed in the Login column.

Troubleshoot Authentication

The [manager](#) log is used for troubleshooting authentication.

You can specify logging parameters in the `ppem-manager.yml` manager configuration file. Logging can be done in a separate file or in the system journal (`journalctl`).

To troubleshoot authentication:

1. Connect to the server where the manager is installed.
2. View the manager log:
 - If logging is done in a separate file:

```
-- Output recent PPEM errors --
```

```
tail -n 1000 "path_to_the_PPEM_log_file" | grep ERROR
```

- If logging is done in the system journal:

```
-- Output PPEM errors for the last 5 minutes --  
journalctl --since "5m ago" -u ppem.service -g ERROR
```

3. Perform the actions required for troubleshooting authentication.

4.7. Connecting the Manager and Agents to Instances via SSL

During or after installing PPEM, you can enable the [manager](#) and [agents](#) to connect to [instances](#) via SSL. To do this, specify the following parameters in the `ppem-manager.yml` manager or `ppem-agent.yml` agent configuration file:

- Manager parameters:
 - `repo.sslmode`
 - `repo.sslrootcert`
 - `repo.sslcert`
 - `repo.sslkey`
- Agent parameters:
 - `agent.instance.connection_defaults.sslmode`
 - `agent.instance.connection_defaults.sslrootcert`
 - `agent.instance.connection_defaults.sslcert`
 - `agent.instance.connection_defaults.sslkey`

Important

PPEM does not support encrypted SSL keys configured using the `sslpassword` parameter.

For the parameter description, refer to the [official Postgres Pro documentation](#).

4.8. Integration with Tracing Tools

PPEM supports *distributed tracing* for requests. Tracing data is exported from PPEM using OTLP (OpenTelemetry protocol) and sent to a receiver that supports OTLP, such as `pgpro-otel-collector`, OpenTelemetry Collector, Grafana Tempo, or Jaeger.

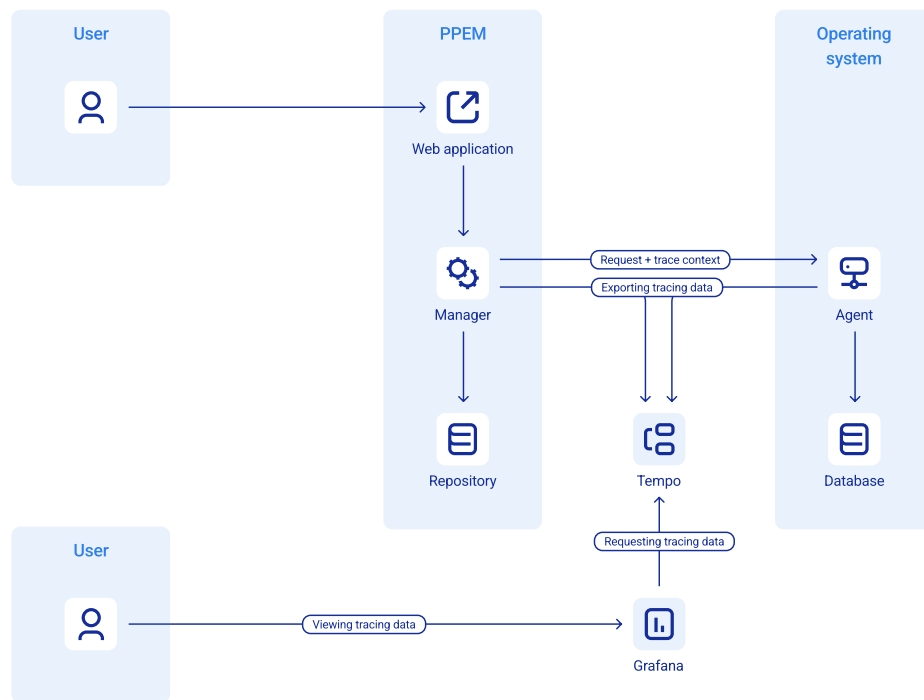
Note

Distributed tracing applies to incoming requests to the [manager](#) and [agents](#), as well as to their databases. It also applies to requests required for internal manager and agent operations.

However, distributed tracing does not apply to requests to other services.

4.8.1. Tracing Architecture

Below is an example of a distributed tracing architecture.

Figure 4.1. Distributed tracing architecture

The user interacts with PPEM through the [web application](#). To perform the required user actions, the web application sends requests to the manager.

Upon receiving a request from the web-application or API, the manager creates a root span, generates the trace context consisting of the trace and span ID, and saves the span start timestamp. The trace context is then associated with the request and used throughout the entire time this request is processed. After processing is finished, the span end timestamp is saved.

For each request to the repository or agents, the manager creates a child span. The child span has a unique ID and start/end timestamp, but it reuses the trace ID of the root span. The child span also stores the parent span ID. Once the child span starts or ends, its trace context is updated.

With each request to an agent, the manager passes the trace context. Similarly to the manager, upon receiving a request, the agent creates a child span reusing the received trace ID. The agent also creates child spans when sending requests to databases.

The manager and agents periodically send tracing data to the receiver, which saves it to the storage. Tracing data with the same trace ID represents a single trace that you can view using Jaeger or Grafana.

4.8.2. Integration with Grafana Tempo

This section explains how to provide tracing by integrating PPEM with Grafana Tempo.

Important

Grafana Tempo is used as an example to demonstrate the integration process. You can select any other supported trace storage based on your organization's requirements.

Within this integration, PPEM sends tracing data using OTLP via HTTP to Grafana Tempo for storing and processing. If required, tracing data is then sent to the Grafana visualization system to be represented in the form of graphs.

Before performing this instruction, install and set up Grafana and Grafana Tempo. For more information, refer to the official [Grafana](#) and [Grafana Tempo](#) documentation.

To integrate PPEM with Grafana Tempo:

1. Configure Grafana Tempo to receive tracing data.

In the Grafana Tempo configuration file, specify:

```
distributor:
  receivers:
    otlp:
      protocols:
        http:
          endpoint: "receiver_endpoint_URL"
```

Where `endpoint` is the endpoint URL of the tracing data receiver, for example, `0.0.0.0:4318`.

2. Configure the manager and agents to export tracing data.

In the `ppem-manager.yml` manager and `ppem-agent.yml` agent configuration files, specify:

```
otlp:
  traces:
    exporter:
      protocol: "http"
      endpoint_url: "exporter_endpoint_URL"
```

Where `endpoint_url` is the endpoint URL of the tracing data exporter, for example, `http://tempo.example.org:4318/v1/traces`.

3. Configure Grafana to work with Grafana Tempo:
 - a. Go to Home → Connections → Data sources.
 - b. Click Add new data source.
 - c. From Data source type, select Tempo.
 - d. Enter the data source name, for example, `tempo-1`.
 - e. In Connection URL, enter the endpoint URL of the tracing data exporter, for example, `http://tempo.example.org`.
 - f. (Optional) To view the list of traces:
 - i. Go to Home → Explore.
 - ii. Select the previously created data source.

4.9. Configuration Behind a Reverse Proxy

You can proxy requests to [installed PPEM](#) based on required URL prefixes.

When performing this instruction, consider the following:

- `nginx` is used as a reverse proxy example
- `/ppem` is used as a URL prefix example

To configure PPEM behind a reverse proxy:

1. On the server where `nginx` is installed:
 - a. In the `http` block of the `nginx.conf` configuration file, specify:

```
map $http_upgrade $connection_upgrade {
```

```

        default upgrade;
        ''         close;
    }

```

- b. Configure redirection to the `/ppem` URL prefix.

For example, in `/etc/nginx/sites-enabled/default`, specify:

```

server {
    listen 80 default_server;
    root /var/www/html;
    server_name _;
    location /ppem/ {
        rewrite ^/ppem/(.*)/\$1 break;
        proxy_pass http://127.0.0.1:8080/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
        proxy_set_header Host $http_host;
    }
    location / {
        return 404;
    }
}

```

- c. Reload nginx:

```
systemctl reload nginx
```

For more information, refer to the [official nginx documentation](#).

2. On the server where the [manager](#) is installed:

- a. In the `ppem-manager.yml` manager configuration file, specify:

```

frontend:
  PPEM_API_PREFIX: /ppem
  PPEM_FRONTEND_BASENAME: /ppem

```

- b. In `usr/share/ppem/web-app/index.html`, replace `<base href="/" />` with `<base href="/ppem/" />`.

- c. Restart PPEM:

```
restart ppem
```

3. On the servers where [agents](#) are installed:

- a. In the `ppem-agent.yml` agent configuration file, add the `/ppem` URL prefix to the `agent.manager.url` parameter value.

This value is specified in the `scheme://manager_network_address/path_to_API_version` format. You must add the URL prefix between `/manager_network_address/` and `/path_to_API_version`.

For example, if the current value is `https://example.postgrespro.com/v1`, the updated value must be `https://example.postgrespro.com/ppem/v1`.

- b. Restart the agent:

```
systemctl restart ppem-agent
```

4. (Optional) To enable nginx to download reports of the required size, in the `http`, `server`, or `location` directive, specify the `client_max_body_size: report_size_in_MBs;` parameter.

For example, in the `server` directive, specify:

```
server {  
    listen 80 default_server;  
    root /var/www/html;  
    server_name _;  
    client_max_body_size 100M;  
    ....  
}
```

In this case, the maximum report size is 100 MB.

Chapter 5. Using PPEM

This section explains how to manage PPEM using the [web application](#).

5.1. Logging in to the Web Application

You can log in to the [web application](#) after [installing PPEM](#).

To log in to the web application:

1. In the browser address bar, enter the network address of the server where the [manager](#) is installed.

You can specify the port number on which the server listens for incoming connections. If a DNS service is configured on the network, enter the network name of the server.

The server network address and port number can be specified in the `ppem-manager.yml` manager configuration file using the `http.server.address` and `http.server.port` parameters.

2. On the authorization page, enter your login and password, then click Log in.

When logging in to the web application for the first time, use the default user account with the `admin` login and the `admin` password.

It is recommended to perform the following steps after logging in to the web application for the first time:

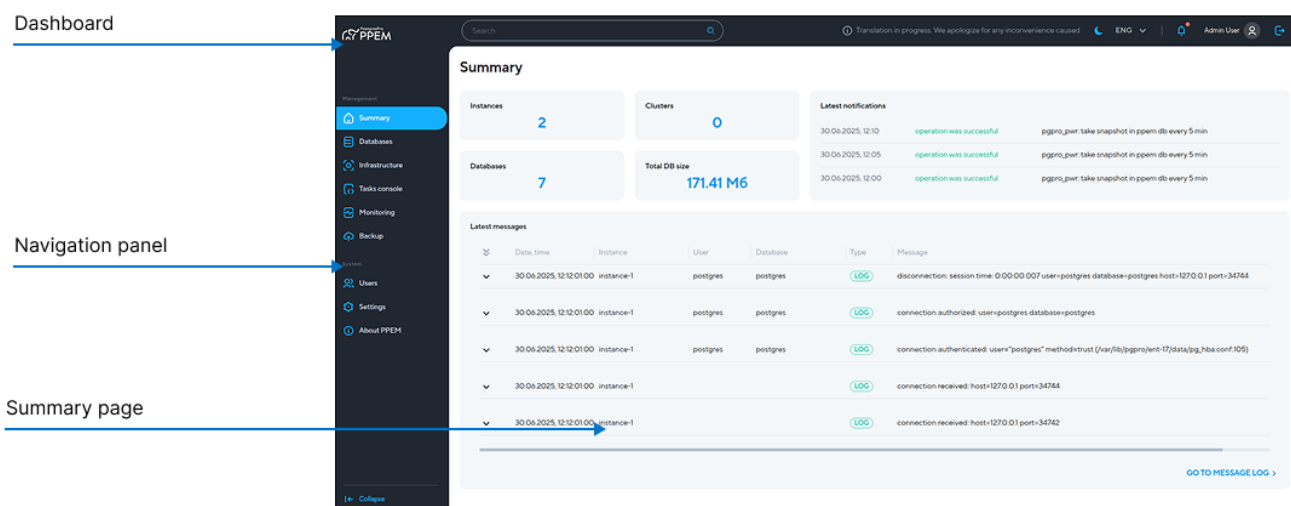
- [Edit](#) the default user with the `admin` login and change its password.
- [Create](#) a separate user for yourself.

5.2. Web Application Interface

This section describes the following elements of the [web application](#) interface:

- [Summary page](#)
- [Navigation panel](#)
- [Toolbar](#)

Figure 5.1. Web application interface



Summary Page

After you [log in](#) to the [web application](#), the Summary page with the main PPEM information is automatically displayed.

The following blocks are displayed on the summary page:

- Instances: Allows you to go to managing [instances](#).
- Clusters: Allows you to go to managing [clusters](#).
- Databases: Allows you to go to managing [databases](#).
- Total DB size: Displays the total amount of memory occupied by all databases.
- Recent Notifications: displays the three latest notifications received from PPEM.
- Latest messages: Displays the latest [log](#) messages.

You can view the full message log by clicking Go to message log.

Navigation Panel

Navigation through the [web application](#) pages is done using the navigation panel. When you go to a page, its contents are displayed on the right.

Certain pages have nested pages. If you go to a page containing nested pages, they are also displayed in the navigation panel.

To collapse or expand the navigation panel, at its bottom, click Collapse or  .

You can go to the Summary page from any page by clicking the PPEM logo at the top of the navigation panel.

Toolbar

When going to any [web application](#) page, the dashboard is displayed at the top of the page.

You can perform the following actions using the dashboard:

- Find a PPEM component by entering its name in the search field.
- Switch between web application themes:
 - To switch to the dark theme, click  .
 - To switch to the light theme, click  .
- Change the web application language:
 - To switch the language to English, select ENG: English.
 - To switch the language to Russian, select РУС: Русский.
- View notifications received from PPEM by clicking  .

You can perform the following actions in the opened window:

- To mark all notifications as read, click Read all.
- To view [tasks](#), click Show all.
- Log out of the web application by clicking  .

5.3. System Configuration

This section explains how to view PPEM summary information, as well as how to manage agents and tags.

5.3.1. Viewing PPEM Summary Information

In the navigation panel, go to About PPEM.

The following blocks will be displayed:

- Platform: Information about the server where the [manager](#) is installed.

The following parameters are displayed:

- Platform: The architecture of the server CPU.
- Kernel: The Linux kernel version.
- System: The version of the operating system installed on the server.
- Nodename: The FQDN of the server.
- Manager: Information about the manager.

The following parameters are displayed:

- Version: The manager version.
- Configuration: The path to the server catalog where the `ppem-manager.yml` configuration file is located.
- Repository: Information about the [repository](#).

The following parameters are displayed:

- Version: The repository version.
- Connection: The connection string for connecting to the repository database.
- Useful links: Links to the Postgres Pro official documentation.

5.3.2. Agents

This section explains how to manage [agents](#). It includes the following instructions:

- [Adding an Agent](#)
- [Viewing Agents](#)
- [Editing an Agent](#)
- [Deleting an Agent](#)

For more information about agents, refer to [Architecture](#).

Adding an Agent

Once installed on the server, the agent is automatically created in the [web application](#). If this does not happen, you can add the installed agent manually.

To add an agent:

1. In the navigation panel, go to Infrastructure → Agents.
2. In the top-right corner of the page, click Add agent.
3. Enter parameters of the new agent (parameters marked with an asterisk are required):
 - Name.
 - Host: The network address of the server where the agent is installed.
 - Port: The port number that the server uses to listen for incoming connections.
 - Protocol: The protocol used by the agent.

Possible values:

- http
- https

Note

The URL for connecting the agent to the [manager](#) is generated automatically based on the Host, Port, and Protocol parameters.

4. Click Save.

Viewing Agents

In the navigation panel, go to Infrastructure → Agents.

The table of agents with the following columns will be displayed:

- Agent: The unique name of the agent.

If the agent is automatically created in the web application, the name is generated based on the host name of the server where the agent is installed.

The agent status indicator is displayed to the left of the agent name. You can view the status by hovering over the agent name.

Possible statuses:

- `online`: The agent is working normally.
- `stopped`: The agent service was stopped.
- `unknown`: The agent status is unknown.
- `not_responding`: Failed to access the agent service.
- URL: The URL for connecting the agent to the manager.
- Authentication key: The key that the agent uses to authenticate when connecting to the manager.
- Version.

If the agent version differs from the manager version, the corresponding warning is displayed.

- Actions.

For more information about available actions, refer to other instructions in this section.

Editing an Agent

By default, agents automatically update their information, so the changes you make get overwritten over time.

You can disable the automatic update by setting the `collectors.instances.disabled` parameter to `true` in the `ppem-agent.yml` agent configuration file.

To edit an agent:

1. In the navigation panel, go to Infrastructure → Agents.
2. Click  next to the agent.
3. Edit agent parameters.

4. Click Save.

Deleting an Agent

Important

After deleting the agent from the web application, you must also delete it from the server. Otherwise, the agent will be automatically recreated in the web application.

To delete an agent:

1. In the navigation panel, go to Infrastructure → Agents.
2. Click  next to the agent.
3. Click Delete.

5.3.3. Tags

Tags are unique labels that can be assigned to [instances](#) when creating and editing instances. You can filter instances by assigned tags.

This section explains how to manage tags. It includes the following instructions:

- [Creating a Tag](#)
- [Viewing Tags](#)
- [Editing a Tag](#)
- [Deleting a Tag](#)

Creating a Tag

1. In the navigation panel, go to Settings.
2. In the top-right corner of the page, click Add tag.
3. Enter parameters of the new tag (parameters marked with an asterisk are required):
 - Description.
 - Display name.
4. Click Add.

Viewing Tags

In the navigation panel, go to Settings.

The table of tags with the following columns will be displayed:

- Description.
- Display name.
- Actions.

For more information about available actions, refer to other instructions in this section.

Editing a Tag

1. In the navigation panel, go to Settings.
2. Click  next to the tag.

3. Edit tag parameters.
4. Click Save.

Deleting a Tag

Important

Deleted tags cannot be restored.

To delete a tag:

1. In the navigation panel, go to Settings.
2. Click  next to the tag.
3. Click Delete.

5.4. User Management

Users manage PPEM through the [web application](#). If required, you can group them for centralized management.

User access levels to different operations are determined by user roles. Each user role has a predefined set of privileges. User roles can be assigned to both individual users and groups.

For more information about user roles and privileges, refer to [Role-Based Access Control \(RBAC\) Model](#).

This section explains how to manage users and user groups. It also includes information about existing user roles and privileges.

5.4.1. Users

This section explains how to manage users. It includes the following instructions:

- [Creating a User](#)
- [Viewing Users](#)
- [Editing a User](#)
- [Configuring the Automatic User Blocking](#)
- [Deleting a User](#)

Creating a User

1. In the navigation panel, go to Users.
2. In the top-right corner of the page, click Add user.
3. Enter parameters of the new user (parameters marked with an asterisk are required):
 - Name.
 - Last name.
 - Email.
 - Login and Password: The user login and password for [logging in to the web application](#).
The minimum password length is 8 characters.
 - Job title.
 - Phone.
 - Group: The groups to which the user will be added.

You can also add a user to a group when [creating](#) or [editing](#) the group.

- Permissions: The [user roles](#) that will be assigned to the user.

To assign a role to a user, click Add role + and select the role. For certain user roles, select the objects to which these roles will grant access.

- PPEM access: Specifies whether the user can log in to the web application.

Possible values:

- Active: The user can log in to the web application.
- Blocked: The user is blocked from logging in to the web application.

To unblock a user, select Active from PPEM access when [editing the user](#).

4. Click Save.

Viewing Users

In the navigation panel, go to Users.

The table of users with the following columns will be displayed:

- Full name: The name and last name of the user.
- Login: The user login for [logging in to the web application](#).
- Email.
- PPEM access: Specifies whether the user can log in to the web application.

Possible values:

- Active.
- Auto-blocked: The user was automatically blocked from logging in to the web application due to too many failed log-in attempts.
- Blocked: The user was blocked from logging in to the web application by an administrator.
- Personal roles: The [user roles](#) assigned to the user.
- Group: The [PPEM groups](#) and LDAP groups to which the user is added.

Editing a User

1. In the navigation panel, go to Users.

2.

Click  next to the user.

3. Edit user parameters.

4. Click Save.

Configuring the Automatic User Blocking

You can specify the maximum number of failed attempts to [log in to the web application](#), after which the user is automatically blocked from logging in again. To do this, use the `lifetime.max_failed_login_attempts: maximum_number_of_failed_attempts` parameter of the `ppem-manager.yml` manager configuration file.

Note

Users blocked due to too many failed log-in attempts are not unblocked automatically.

To unblock a user, select Active from PPEM access when [editing the user](#).

Deleting a User

Important

Deleted users cannot be restored.

To delete a user:

1. In the navigation panel, go to Users.
2. Click  next to the user.
3. Click Yes, delete.

5.4.2. User Groups

This section explains how to manage user groups. It includes the following instructions:

- [Creating a User Group](#)
- [Viewing User Groups](#)
- [Viewing User Group Information](#)
- [Editing a User Group](#)
- [Deleting a User Group](#)

Creating a User Group

1. In the navigation panel, go to Users → Groups.
2. In the top-right corner of the page, click Create group.
3. Enter parameters of the new user group (parameters marked with an asterisk are required):
 - Name.
 - Description.
 - LDAP group: The LDAP group that will be associated with the user group.
 - Users: The users that will be added to the group.

You can also add a user to a group when [creating](#) or [editing](#) the user.

- Permissions: The [user roles](#) that will be assigned to the user group.

To assign a role to a user group, click Add role + and select the role. For certain user roles, select the objects to which these roles will grant access.

4. Click Save.

Viewing User Groups

In the navigation panel, go to Users → Groups.

The table of user groups with the following columns will be displayed:

- Group: The name of the user group.
- Description.
- LDAP group: The LDAP group associated with the user group.
- Actions.

For more information about available actions, refer to other instructions in this section.

Viewing User Group Information

You can view which users are added and which user roles are assigned to the group.

To view information about a user group:

1. In the navigation panel, go to Users → Groups.
2. Click  → Show details next to the user group.

Editing a User Group

1. In the navigation panel, go to Users → Groups.
2. Click  → Edit next to the user group.
3. Edit user group parameters.
4. Click Save.

Deleting a User Group

Important

Deleted user groups cannot be restored.

Deleting a group does not delete its users but revokes the user roles assigned to them through the group. If required, you can [delete](#) users individually or [edit](#) them to reassign user roles.

To delete a user group:

1. In the navigation panel, go to Users → Groups.
2. Click  → Delete next to the user group.
3. Click Delete.

5.4.3. User Roles and Privileges

This section describes [user roles](#) and [privileges](#). It also includes the following instructions:

- [Creating a User Role](#)
- [Viewing User Roles and Privileges](#)
- [Editing a User Role](#)
- [Deleting a User Role](#)

For more information about roles and privileges, refer to [Role-Based Access Control \(RBAC\) Model](#).

User Roles Description

The following user roles are available in PPEM:

- System administrator role has a full set of privileges.
- Guest role can view a limited number of objects.
- Instance objects administrator role can manage instance objects.

- Instance objects viewer role can view instance objects.
- Instance administrator role can manage an instance.
- Instance PSQL user role can run psql within an instance.
- Access administrator role can manage user roles and group roles.
- Repositories and packages administrator can manage repositories and packages.

Privileges Description

User roles can have the following privileges:

- `privilege_view`: Viewing privileges and their descriptions.
- `role_create`: Creating user roles.
- `role_view`: Viewing user roles and their parameters.
- `role_edit`: Editing user roles.
- `role_delete`: Deleting user roles.
- `user_create`: Creating users.
- `user_view_all`: Viewing any users and their parameters.
- `user_edit_all`: Editing any users.
- `user_delete`: Deleting users.
- `project_create`: Creating projects.
- `project_view`: Viewing projects and their parameters.
- `project_edit`: Editing projects.
- `project_delete`: Deleting projects.
- `notification_create`: Creating notifications.
- `notification_view`: Viewing notifications and their parameters.
- `notification_edit`: Editing notifications.
- `notification_delete`: Deleting notifications.
- `group_create`: Creating user groups.
- `group_view`: Viewing user groups and their parameters.
- `group_edit`: Editing user groups.
- `group_delete`: Deleting user groups.
- `host_create`: Creating servers.
- `host_view`: Viewing servers and their parameters.
- `host_edit`: Editing servers.
- `host_delete`: Deleting servers.
- `agent_create`: Creating [agents](#).
- `agent_view`: Viewing agents and their parameters.
- `agent_edit`: Editing agents.
- `agent_delete`: Deleting agents.
- `instance_create`: Creating instances.
- `instance_view`: Viewing instances.
- `instance_edit`: Editing instances.
- `instance_delete`: Deleting instances.
- `session_view_all`: Viewing any user sessions.
- `session_delete_all`: Editing any user sessions.
- `session_update`: Updating user sessions.
- `command_create`: Creating commands.
- `command_view_all`: Viewing any commands.
- `command_edit_all`: Editing any commands.
- `command_delete_all`: Canceling any commands.
- `instance_object_view`: Viewing instance objects and their parameters.
- `metrics_view`: Viewing metrics.
- `job_create`: Creating jobs.
- `job_view_all`: Viewing any jobs.
- `job_edit_all`: Editing any jobs.

- `job_delete_all`: Deleting any jobs.
- `backup_create`: Creating backups.
- `backup_view`: Viewing backups.
- `backup_edit`: Editing backups.
- `backup_delete`: Deleting backups.
- `datasource_create`: Creating data storages.
- `datasource_view`: Viewing data storages.
- `datasource_edit`: Editing data storages.
- `datasource_delete`: Deleting data storages.
- `maintenance_create`: Executing maintenance commands.
- `instance_service_control`: Executing utility commands.
- `instance_settings_create`: Creating instance parameters.

This service privilege allows agents to add instance parameters to the repository database.

- `instance_settings_view`: Viewing instance parameters.
- `instance_settings_edit`: Editing instance parameters.
- `query_state_read`: Executing the `pg_query_state` command.
- `logs_view`: Viewing logs.
- `chart_create`: Creating graphs.
- `chart_view`: Viewing graphs.
- `chart_edit`: Editing graphs.
- `chart_delete`: Deleting graphs.
- `chart_group_create`: Creating graph groups.
- `chart_group_view`: Viewing graph groups.
- `chart_group_edit`: Editing graph groups.
- `chart_group_delete`: Deleting graph groups.
- `stat_activity_view`: Viewing the `pg_stat_activity` view statistics.
- `stat_statements_view`: Viewing any SQL statements executed by the server.
- `overview_view`: Viewing the system overview.
- `tag_create`: Creating tags.
- `tag_view`: Viewing tags and their parameters.
- `tag_edit`: Editing tags.
- `tag_delete`: Deleting tags.
- `progress_stats_view`: Viewing `pg_stat_progress_*` views statistics.
- `about_view`: Viewing the system information.
- `pgpro_pwr_databases_view`: Viewing `pgpro_pwr` extensions.
- `pgpro_pwr_servers_delete`: Deleting `pgpro_pwr` servers.
- `pgpro_pwr_servers_view`: Viewing `pgpro_pwr` servers.
- `replication_node_create`: Creating replication nodes.
- `pgpro_pwr_servers_add`: Adding `pgpro_pwr` servers.
- `pgpro_pwr_servers_patch`: Installing patches for `pgpro_pwr` extensions.
- `stat_locktree_view`: Viewing a locktree.
- `pgpro_pwr_samples_create`: Creating `pgpro_pwr` samples.
- `pgpro_pwr_samples_get`: Viewing `pgpro_pwr` samples.
- `pgpro_pwr_samples_delete`: Deleting `pgpro_pwr` samples.
- `pgpro_pwr_report_create`: Creating `pgpro_pwr` reports.
- `pgpro_pwr_report_delete`: Deleting `pgpro_pwr` reports.
- `replication_node_view`: Viewing replication nodes.
- `pgpro_pwr_report_view`: Viewing `pgpro_pwr` reports.
- `settings_preset_view`: Viewing presets.
- `pgpro_pwr_overview`: Viewing the contents of `pgpro_pwr` reports.
- `user_roles_edit`: Assigning and removing user roles.
- `group_roles_edit`: Assigning and removing user group user roles.
- `user_groups_edit`: Adding and deleting users from groups.
- `job_run_all`: Running any jobs.

Creating a User Role

1. In the navigation panel, go to Users → Roles and privileges.
2. In the top-right corner of the page, click Add role.
3. Enter parameters of the new user role (parameters marked with an asterisk are required):
 - System name: The system identifier of the role.
 - Name.
 - Description.
 - Privileges: The privileges that will be associated with the user role.

For more information about privileges, refer to [Privileges Description](#).

4. Click Next, and then verify the list of privileges.
5. Click Add.

Viewing User Roles and Privileges

In the navigation panel, go to Users → Roles and privileges.

The table of user roles with the following columns will be displayed:

- Role: The name and system identifier of the user role.

To display the list of names and system identifiers of the privileges associated with the user role,

click  next to the user role name.

- Privileges: The number of privileges associated with the user role.
- Description.
- Actions.

For more information about available actions, refer to other instructions in this section.

Editing a User Role

1. In the navigation panel, go to Users → Roles and privileges.
2. Click  next to the user role.
3. Edit user role parameters.
4. Click Next, and then verify the list of privileges.
5. Click Save.

Deleting a User Role

Important

Deleted user roles cannot be restored.

To delete a user role:

1. In the navigation panel, go to Users → Roles and privileges.
2. Click  next to the user role.

3. Confirm the operation and click Delete.

5.5. Instances

This section explains how to manage instances. It includes the following instructions:

- [Creating an Instance](#)
- [Viewing Instances](#)
- [Checking the Consistency of an Instance Catalog](#)
- [Stopping and Starting an Instance](#)
- [Restarting an Instance](#)
- [Creating a Backup](#)
- [Applying a Configuration Preset to an Instance](#)
- [Editing an Instance](#)
- [Selecting a Repository Instance](#)
- [Deleting an Instance](#)

Creating an Instance

[Agents](#) automatically discover hosts, instances, instance objects, and other components installed on the server and create them in the [web application](#). You can configure the automatic discovery in the `ppem-agent.yml` agent configuration file.

Note

The automatic discovery and a number of other features are not supported when [PPEM is installed in a secure environment](#).

PPEM provides the following ways of creating instances:

- [Creating a New Instance](#)
- [Adding an Existing Instance](#)
- [Creating an Instance from a Backup](#)

Creating a New Instance

Creating a new instance includes creating a new main data catalog and starting the instance service on the server.

To create a new instance:

1. In the navigation panel, go to Infrastructure → Instances.
2. In the top-right corner of the page, click Add instance.
3. Select Create a new instance, and then click Next.
4. Enter parameters of the new instance (parameters marked with an asterisk are required):
 - Name.
 - Server: The server where the instance is installed.
 - System user: The OS user who will own the instance files and catalogs and on behalf of which the instance service will run. In most cases, it is the `postgres` user.

Ensure that the specified user exists in the OS.

- Main data directory: The path to the server catalog where the main instance catalogs and files will be located.
- Connection address and Connection port: The network address and port number that the instance will use for receiving client connections.

- **Authentication method:** The authentication method that the instance will use to verify users when receiving client connections.

Possible values:

- `scram-sha-256`.
- `md5`.
- `trust`: Do not perform authentication.

It is recommended to select this value only for test environments.

- **DB super user and Super user password:** The name and password of the DBMS user that will be created and on behalf of which the agent will connect to the instance.

The default DBMS user name is `postgres`.

- **Tags:** The [tags](#) that will be assigned to the instance.
- **Configuration preset:** The [configuration preset](#) that will be applied to the instance.

Possible values:

- `—`: Do not apply a configuration preset to the instance.
- **Settings for 1C:** Apply the 1C configuration preset to the instance.
- **Settings for OLTP:** Apply the OLTP (Online Transaction Processing) configuration preset to the instance.

You can [apply](#) another configuration preset to the instance later.

- **Start instance after creation:** Specifies whether the instance service starts after instance creation.

5. Click Execute.

Adding an Existing Instance

You may need to add an existing instance if the automatic discovery of instances installed on the server is disabled.

Before performing this instruction, ensure that the instance is running on the server and is ready to receive client connections.

To add an existing instance:

1. In the navigation panel, go to Infrastructure → Instances.
2. In the top-right corner of the page, click Add instance.
3. Select Add an existing instance, and then click Next.
4. In Instance parameters, enter parameters of the instance (parameters marked with an asterisk are required):
 - **Name.**
 - **Server:** The server where the instance is installed.
 - **Main data directory:** The path to the server catalog where the main instance catalogs and files will be located.
 - **Connection address and Connection port:** The network address and port number that the instance uses for receiving client connections.
 - **DB super user and Super user password:** The name and password of the DBMS user on behalf of which the agent will connect to the instance.

The default DBMS user name is `postgres`.

- Tags: The [tags](#) that will be assigned to the instance.

5. Click Execute.

Creating an Instance from a Backup

For more information about backups, refer to [Backup](#).

Before performing this instruction, [create a backup](#).

To create an instance from a backup:

1. In the navigation panel, go to Infrastructure → Instances.
2. In the top-right corner of the page, click Add instance.
3. Select Create from backup, and then click Next.
4. Enter parameters of the backup from which the instance will be created (parameters marked with an asterisk are required):
 - Catalog: The storage catalog where the backup is located.
 - Instance: The instance for which the backup is created.
 - Period: The time period when the backup was created.
 - Backup: The backup from which the instance will be created.
5. Click Next, and then enter parameters of the new instance (parameters marked with an asterisk are required):
 - Name.
 - Server: The server where the instance is installed.
 - System user: The OS user who will own the instance files and catalogs and on behalf of which the instance service will run. In most cases, it is the `postgres` user.

It is recommended to ensure that the specified user exists in the OS.

- Main data directory: The path to the server catalog where the main instance catalogs and files will be located.
- Connection address and Connection port: The network address and port number that the instance will use for receiving client connections.
- Tags: The [tags](#) that will be assigned to the instance.
- Backup: The backup from which the instance will be created.

The value is entered automatically.

- Backup size: The size of the backup from which the instance will be created.

The value is entered automatically.

- Restore point: The state to which the instance will be restored.

Possible values:

- —: Restore the instance to the last state covered by the backup.
- Time: Restore the instance state to the specified date and time covered by the backup.

For this value, in Time, specify the date and time.

- LSN: Restore the instance to the state corresponding to the specified WAL LSN.

For this value, in LSN, enter the WAL LSN.

- Transaction: Restore the instance state to the specified transaction number.

For this value, in Transaction, enter the transaction number.

For Time, LSN, and Transaction, specify the following parameters:

- Restore including the specified value: Specifies whether the instance state is restored including the specified value.

For example, if you enter 123456 in Transaction and enable Restore including the specified value, the instance state is restored to the transaction 123456. Otherwise, the instance state is restored to the 123455 transaction.

- Action after restore: The action to perform on the server after restoring the instance to the state.

Possible values:

- Pause after restore: Pause the creation of the instance from the backup. It allows ensuring that the correct state was restored for the instance before creating it.
 - Promote after restore: Create the instance from the backup and start receiving client connections.
 - Shutdown instance after restore: Create the instance from the backup, and then stop the server.
- Partial recovery: Specifies the instance databases that will be restored or excluded from the restoration process.

Possible values:

- Do not use: Restore all instance databases.
- Exclude some databases: Exclude the specified databases from restoration.
- Restore some databases: Restore the specified instance databases.

For Exclude some databases and Restore some database, specify the unique name of the database using Databases, and then click Add database.

- Checking available space: Allows checking whether there is enough disk space on the server for creating the instance from the backup.

To start the check, click Check.

6. Click Execute.

Viewing Instances

In the navigation panel, go to Infrastructure → Instances.

The table of instances with the following columns will be displayed:

- Name:
 - The unique name of the instance.
 - PostgreSQL version: The version and edition of Postgres Pro on the instance server.
 - Data directory: The path to the server catalog where the main instance catalogs and files are located.
 - Port: The port number that the instance uses for receiving client connections.
- Server:

- The FQDN of the instance server.
- The IP address of the instance server.
- The status of the instance.

Possible values:

- Unknown: The status of the instance is unknown.
 - Initialization scheduled: The instance initialization was scheduled.
 - Initializing: The instance is being initialized.
 - Initialized: The instance is initialized.
 - Restoring: The instance is being created from a backup.
 - Restored: The instance is created from the backup.
 - Restore canceled: The creation of an instance from a backup was canceled.
 - Starting: The instance is being started.
 - Started: The instance is started.
 - Stopping: The instance is being stopped.
 - Stopped: The instance is stopped.
 - Restarting: The instance is being restarted.
 - Reloading: The cluster is being reloaded.
 - Failed: There is an error with the instance.
 - Scheduled for removal: The instance is being deleted.
 - Base backup: A backup is being created for the instance.
 - Agent not responding: The agent installed on the instance server is not responding.
- Role.

Possible values:

- primary: The instance is the primary node of the cluster and replicates data to standby nodes.
- standby: The instance is a standby node of a replication cluster and receives replicated data from the primary node.
- cascade: The instance is a standby node of a streaming replication cluster. It receives replicated data from the main node and provides data replication to other standby nodes simultaneously.
- maintenance: The instance is a separate node and is not added to a replication cluster.

For more information about data replication, refer to the [official Postgres Pro documentation](#).

- DB:
 - Database: Unique names of databases.
 - Transactions per second: The number of transactions per second in the database.
 - Connections: The number of client database connections.
- Tags: The [tags](#) assigned to the instance.

Checking the Consistency of an Instance Catalog

The catalog consistency check ensures that the main instance files were not damaged during storage.

To check the consistency of an instance catalog:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click  next to the instance.
3. Click Run.

To view the result, in the navigation panel, go to Tasks console and click Show log next to the task that was automatically created for checking the consistency of the instance catalog.

Stopping and Starting an Instance

1. In the navigation panel, go to Infrastructure → Instances.
2. Click  or  next to the instance.
3. To stop an instance, click Stop.

Restarting an Instance

1. In the navigation panel, go to Infrastructure → Instances.
2. Click  next to the instance.
3. Click Restart.

Creating a Backup

For more information about backups, refer to [Backup](#).

To create a backup:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click  next to the instance.
3. Enter parameters of the new backup (parameters marked with an asterisk are required):

- Instance: The instance for which the backup will be created.

The value is entered automatically.

- Copy storage: The storage where the backup will be located.

You can select a local or S3 storage. The local storage must be located on the same server as the instance for which you are creating the backup.

- User and Password: The name and password of the DBMS user under which the backup will be performed.
- Database: The database for connecting to the instance.
- Backup mode: The backup creation mode.

Possible values:

- full
- page
- ptrack
- delta

For more information about backup modes, refer to the official Postgres Pro documentation on [pg_probackup](#).

- Threads count: The number of parallel threads that will be started at backup creation.

- **Waiting time (sec):** The waiting timeout in seconds for WAL segment archiving and streaming.
- **Create a stand-alone backup:** Specifies whether to create a streaming backup that includes the WAL records required for restoring the instance later.
- **Replication slot:** The replication slot that will be used for transferring WAL records.
- **Create temporary replication slot:** Specifies whether to create a temporary replication slot for transferring WAL records of the instance for which you are creating the backup.

If you enable this checkbox, WAL segments are available even when they are switched at backup creation.

4. Click Next, and then specify additional parameters if required:

- **External catalogs:** The path to the instance catalog that will also be included in the backup.
- **include log catalog:** Specifies whether the backup includes the catalog with the instance activity logs.
- **Don't check copy:** Specifies whether to skip the automatic verification of the created backup.

If you enable this checkbox, the backup is created faster.

- **Smooth execution of the checkpoint:** Specifies whether backup creation starts only after the scheduled checkpoint.
- **Disable block-level verification:** Specifies whether to disable the block-level checksum verification for faster consistency checking at backup creation.
- **Compression level:** The file compression level at backup creation.

You can enter a value from 0 to 9, where 0 disables file compression, and 9 enables the highest file compression level.

- **Compression algorithm:** The algorithm used for compressing files.

Possible values:

- zlib
- lz4
- zstd
- pglz

This parameter is available only if you enter a value greater than 0 in Compression level.

- **Pinning:** The pinning parameters of the backup.

Possible values:

- **Do not pin:** Do not pin the backup.

If you select this value, the parameters specified in Parameters of storing are used.

- **ttl:** After the backup is created, it cannot be deleted from a storage during the specified number of days.

For this value, in Retention period, days, enter the number of days.

- **expire-time:** The backup cannot be deleted from a storage until the specified date and time.

For this value, in Retention period until, specify the date and time.

- **Storage parameters:** The backup storage parameters of the storage catalog created for the instance.

- Retention redundancy: The maximum number of full backups.

For example, if you specify 3, the catalog can contain a maximum of three full backups.

To disable this limitation, specify 0. In this case, the number of backups in the catalog is not limited.

- Retention window: The number of days (24 hours) covered by backups. For example, if you specify 7, the catalog must always contain backups required for restoring the data for the last seven days, including today.

To disable this limitation, specify 0. In this case, backups can be deleted from the catalog at any moment.

- WAL depth: The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR).

For example, if you specify 3, the catalog must always contain at least three backups on each timeline.

To disable this limitation, specify 0. In this case, point-in-time recovery is not available.

- Expired copies: The management policy for expired backups.

Possible values:

- Merge: Merge expired backups with new ones if possible.
- Delete: Delete expired backups from the catalog.
- Remove expired WAL: Delete WAL of expired backups from the catalog.

You can enable all checkboxes simultaneously.

Retention redundancy, Retention window, and WAL depth are applied only if you enable Merge and/or Delete for Expired copies.

The values of Retention redundancy and Retention window are considered simultaneously when deleting expired backups from the catalog. For example, if you enter 3 in Retention redundancy and 7 in Retention window, a maximum of three full backups are saved, as well as all backups required for restoring the data for the past 7 days, including today.

You can also configure storage parameters for an [instance](#), as well as for a storage when [creating](#) or [editing](#) it.

The following priority is applied:

- backup parameters are applied first
- instance parameters are applied second
- storage parameters are applied third

For more information about storage parameters, refer to the official Postgres Pro documentation on [pg_probackup](#).

5. Click Execute backup.

Applying a Configuration Preset to an Instance

You can apply another [configuration preset](#) to an instance.

Important

When you apply a configuration preset, the instance can be automatically restarted if it is required for changes to take effect.

To apply a configuration preset to an instance:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click  next to the instance.
3. Select a new configuration preset.
4. Click Apply.

Editing an Instance

You can [select the repository instance](#) when editing an instance.

To edit an instance:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click  next to the instance.
3. Edit instance parameters.
4. Click Save.

Selecting a Repository Instance

The repository database is located in an instance. Similarly to other instances, you can [create](#) this instance in the web application.

In the web application, you must manually specify the instance where the repository database is located.

To select the repository instance:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click  next to the instance.
3. Enable Instance of EM repository.
4. Click Save.

Deleting an Instance

Important

After deleting the instance from the web application, you must also delete it from the server. Otherwise, the instance will be automatically created in the web application again.

When [PPEM is installed in a secure environment](#), if you delete an automatically created instance, it will not be automatically created anymore. In this case, to restore the instance, you must [add it manually](#).

To delete an instance:

1. In the navigation panel, go to Infrastructure → Instances.

2. Click  next to the instance.
3. (Optional) To delete the main instance data catalog on the server, enable Delete with data directory.
4. Confirm the operation and click Delete.

5.6. Clusters

PPEM supports both standard primary-standby and BiHA (Built-in High Availability) clusters. If you install an [agent](#) on a server with a created cluster, the cluster is automatically created in the [web application](#).

This section explains how to manage clusters. It includes the following instructions:

- [Creating a Cluster](#)
- [Viewing Clusters](#)
- [Editing a Cluster](#)
- [Deleting a Cluster](#)
- [Managing Nodes of BiHA Clusters](#)

For more information, refer to the official Postgres Pro documentation on [replication](#) and [built-in high availability](#).

Creating a Cluster

PPEM provides the following ways of creating clusters:

- [Creating a New Standard Primary-Standby Cluster](#)
- [Creating a New BiHA Cluster](#)
- [Creating a Standard Primary-Standby Cluster from an Existing Instance](#)
- [Creating a BiHA Cluster from an Existing Instance](#)

Creating a New Standard Primary-Standby Cluster

When creating a new standard primary-standby cluster, you must specify the parameters of the instance that will be created and used as the primary node.

To create a new standard primary-standby cluster:

1. In the navigation panel, go to Infrastructure → Clusters.
2. In the top-right corner of the page, click Create cluster, and then Next.
3. Enter parameters of the new standard primary-standby cluster (parameters marked with an asterisk are required):
 - Node name: The unique name of the instance.
 - Server: The server where the instance is installed.
 - System user: The OS user who will own the instance files and catalogs and on behalf of which the instance service will run. In most cases, it is the `postgres` user.

Ensure that the specified user exists in the OS.

- Data catalog: The path to the server catalog where the main instance catalogs and files will be located.
- Connection address and Connection port: The network address and port number that the instance will use for receiving client connections.
- Authentication method: The authentication method that the instance will use to verify users when receiving client connections.

Possible values:

- scram-sha-256.
- md5.
- trust: Do not perform authentication.

It is recommended to select this value only for test environments.

- Database user and Database password: The name and password of the DBMS user that will be created and on behalf of which the agent will connect to the instance.

The default DBMS user name is `postgres`.

- Preset: The [configuration preset](#) that will be applied to the instance.

Possible values:

- Do not use preset: Do not apply a configuration preset to the instance.
- Settings for 1C: Apply the 1C configuration preset to the instance.
- Settings for OLTP: Apply the OLTP (Online Transaction Processing) configuration preset to the instance.

You can [apply a new configuration preset](#) later.

- Servers of standby nodes: Allows adding standby nodes to the standard primary-standby cluster.

It is recommended to have an odd number of nodes to ensure a quorum.

To add a standby node:

- a. Click Add servers.
- b. From Server, select the server where the instance is installed.

When the server is selected, the following instance information is displayed:

- Data catalog: The path to the server directory where the main instance directories and files are located.
- Network address and Port: The network address and port number that the instance uses to accept client connections.

- c. In Replication mode, select one of the following values:

- Synchronous

For this value, in Node identifier, enter the unique name of the node for the [synchronous_standby_names](#) configuration parameter.

- Asynchronous

- d. (Optional) To add more standby nodes, click Add server.
- e. Click Save.

4. Click Create cluster.

Creating a New BiHA Cluster

When creating a new BiHA cluster, you must specify the parameters of the instance that will be created and used as the leader node.

Before performing this instruction, satisfy the [required prerequisites](#) for setting up a BiHA cluster.

To create a new BiHA cluster:

1. In the navigation panel, go to Infrastructure → Clusters.

2. In the top-right corner of the page, click Create cluster.
3. In Failover manager, select BiHA, and then click Next.
4. Enter parameters of the new BiHA cluster (parameters marked with an asterisk are required):

- Cluster name.
- Node name: The unique name of the instance.
- Server: The server where the instance is installed.
- System user: The OS user who will own the instance files and catalogs and on behalf of which the instance service will run. In most cases, it is the `postgres` user.

Ensure that the specified user exists in the OS.

- Data catalog: The path to the server catalog where the main instance catalogs and files will be located.
- Connection address and Connection port: The network address and port number that the instance will use for receiving client connections.
- Authentication method: The authentication method that the instance will use to verify users when receiving client connections.

Possible values:

- `scram-sha-256`.
- `md5`.
- `trust`: Do not perform authentication.

It is recommended to select this value only for test environments.

- Database user and Database password: The name and password of the DBMS user that will be created and on behalf of which the agent will connect to the instance.

The default DBMS user name is `postgres`.

- Password for connection: The password for the `biha_replication_user` role. This role is used for connecting follower nodes to the leader node.

For more information, refer to the official Postgres Pro documentation on [BiHA roles](#).

- Preset: The [configuration preset](#) that will be applied to the instance.

Possible values:

- Do not use preset: Do not apply a configuration preset to the instance.
- Settings for 1C: Apply the 1C configuration preset to the instance.
- Settings for OLTP: Apply the OLTP (Online Transaction Processing) configuration preset to the instance.

You can [apply a new configuration preset](#) later.

- Servers of followers: Allows adding follower nodes to the BiHA cluster.

It is recommended to have an odd number of nodes to ensure a quorum.

To add a follower node:

- a. Click Add servers.
- b. From Server, select the server where the instance is installed.

When the server is selected, the following instance information is displayed:

- Data catalog: The path to the server directory where the main instance directories and files are located.
 - Network address and Port: The network address and port number that the instance uses to accept client connections.
 - c. (Optional) To add more follower nodes, click Add server.
 - d. Click Save.
5. Click Create cluster.

Creating a Standard Primary-Standby Cluster from an Existing Instance

Before performing this instruction, [create an instance](#).

To create a standard primary-standby cluster from an existing instance:

1. In the navigation panel, go to Infrastructure → Clusters.
2. In the top-right corner of the page, click Create cluster.
3. In Creation mode, select From instance, and then click Next.
4. Enter parameters of the new standard primary-standby cluster (parameters marked with an asterisk are required):
 - Primary node: The instance that will be used as the primary node of the standard primary-standby cluster.

When the instance is selected, the following information is displayed:

- Version and edition: The Postgres Pro version and edition installed on the instance server.
- Data catalog: The path to the server directory where the main instance directories and files are located.
- Network address and Port: The network address and port number that the instance uses to accept client connections.
- User: The name of the DBMS superuser on behalf of which the agent connects to the instance.
- Servers of standby nodes: Allows adding standby nodes to the standard primary-standby cluster.

It is recommended to have an odd number of nodes to ensure a quorum.

To add a standby node:

- a. Click Add servers.
- b. From Server, select the server where the instance is installed.

When the server is selected, the following instance information is displayed:

- Data catalog: The path to the server directory where the main instance directories and files are located.
- Network address and Port: The network address and port number that the instance uses to accept client connections.
- c. From Replication mode, select one of the following values:
 - Synchronous

For this value, in Node identifier, enter the unique name of the standby node for the [synchronous_standby_names](#) configuration parameter.

- Asynchronous
- d. (Optional) To add more standby nodes, click Add server.
- e. Click Save.

5. Click Create cluster.

Creating a BiHA Cluster from an Existing Instance

Before performing this instruction, satisfy the *required prerequisites* for setting up a BiHA cluster.

To create a BiHA cluster from an existing instance:

1. In the navigation panel, go to Infrastructure → Clusters.
2. In the top-right corner of the page, click Create cluster.
3. In Creation mode, select From instance.
4. In Failover manager, select BiHA, and then click Next.
5. Enter parameters of the new BiHA cluster (parameters marked with an asterisk are required):

- Leader: The instance that will be used as the leader node of the cluster.

When the instance is selected, the following information is displayed:

- Version and edition: The Postgres Pro version and edition installed on the instance server.
- Data catalog: The path to the server directory where the main instance directories and files are located.
- Network address and Port: The network address and port number that the instance uses to accept client connections.
- Password for connection: The password for the `biha_replication_user` role. This role is used for connecting follower nodes to the leader node.

For more information, refer to the official Postgres Pro documentation on *BiHA roles*.

- Servers of followers: Allows adding follower nodes to the BiHA cluster.

It is recommended to have an odd number of nodes to ensure a quorum.

To add a follower node:

- a. Click Add servers.
- b. From Server, select the server where the instance is installed.

When the server is selected, the following instance information is displayed:

- Data catalog: The path to the server directory where the main instance directories and files are located.
 - Network address and Port: The network address and port number that the instance uses to accept client connections.
- c. (Optional) To add more follower nodes, click Add server.
 - d. Click Save.

6. Click Create cluster.

Viewing Clusters

In the navigation panel, go to Infrastructure → Clusters.

The table of clusters with the following columns will be displayed:

- Cluster: The unique name and identifier of the cluster.
- Failover manager.

Possible values:

- Missing
- Patroni
- Biha
- Nodes, pcs.: The number of the cluster nodes.
- Version: The Postgres Pro version and edition of the cluster nodes.
- State.

Possible values:

- Active: All cluster nodes are started.
- Stale: Replication is stopped in the cluster.
- Promotion: The primary/leader node of the cluster is being changed.
- Network Address: The network address of the primary/leader cluster node.
- Last update: The date and time when the cluster state information was last updated.

Note

The [manager](#) receives cluster state information from agents with a delay. Due to the delay, the web application can display outdated information.

- Actions.

For more information about available actions, refer to other instructions in this section.

Editing a Cluster

1. In the navigation panel, go to Infrastructure → Clusters.
2. Click  next to the cluster.
3. Edit cluster parameters.
4. Click Save.

Deleting a Cluster

Important

Deleted clusters cannot be restored.

1. In the navigation panel, go to Infrastructure → Clusters.
2. Click  next to the cluster.
3. Confirm the operation and click Delete.

Managing Nodes of BiHA Clusters

This section explains how to manage nodes of BiHA clusters. It includes the following instructions:

- [Viewing Nodes](#)
- [Selecting a New Leader Node](#)
- [Adding a Follower Node](#)
- [Deleting a Follower Node](#)

Before performing these instructions, [create a BiHA cluster](#).

Viewing Nodes

1. In the navigation panel, go to Infrastructure → Clusters.
2. Click the name of the BiHA cluster.

The table of nodes with the following columns will be displayed:

- Instance: The unique name of the instance.

This column includes additional information:

Type: The type of the node.

Possible values:

- Leader
- Follower
- Parent node: The leader of the BiHA cluster.
- Node status: The status of the instance.

Possible values:

- Unknown: The status of the instance is unknown.
- Initialization scheduled: The instance initialization was scheduled.
- Initializing: The instance is being initialized.
- Initialized: The instance is initialized.
- Restoring: The instance is being created from a backup.
- Restored: The instance is created from the backup.
- Restore canceled: The creation of an instance from a backup was canceled.
- Starting: The instance is being started.
- Started: The instance is started.
- Stopping: The instance is being stopped.
- Stopped: The instance is stopped.
- Restarting: The instance is being restarted.
- Reloading: The cluster is being reloaded.
- Failed: There is an error with the instance.
- Scheduled for removal: The instance is being deleted.
- Base backup: A backup is being created for the instance.
- Agent not responding: The agent installed on the instance server is not responding.
- Replication mode.

Possible values:

- Synchronous
- Asynchronous

This column includes additional information:

Status: The status of replication.

Possible values:

- Active
- Inactive
- Network Address: The network address of the node.

This column includes additional information:

Server: The server where the instance is installed.

- User: The role used for connecting the follower node to the leader role. Usually, this is the `bi-ha_replication_user` role.
- Application: The application used for connecting the follower node to the leader node.
- Connection: The date and time when the follower node connected to the leader node.
- Horizon Visibility: The transaction visibility horizon of the node.
- Lag: The data lag between the leader and the follower node, in bytes.

This column includes additional information:

- Send: The amount of WAL generated by the leader node, but not yet sent to the follower node.
- Write: The amount of WAL sent to the follower node, but not yet written to the memory.
- Flush: The amount of WAL written to the memory of the follower node, but not yet saved to the disk.
- Replay: The amount of WAL saved to the disk of the follower node, but not yet applied to the database.
- Lag time: The time lag between the leader and the follower node, in seconds.

This column includes additional information:

- Write: The time passed between the leader node sending WAL to the follower node and the follower node writing it to the memory.
- Flush: The time passed between the follower node writing WAL to the memory and saving it to the disk.
- Replay: The time passed between the follower node saving WAL to the disk and applying it to the database.
- Reply time: The date and time when the leader node last received information from the follower node.
- Actions.

For more information about available actions, refer to other instructions in this section.

Selecting a New Leader Node

1. In the navigation panel, go to Infrastructure → Clusters.
2. Click the name of the BiHA cluster.

3. Click  next to the follower node.
4. Confirm the operation and click Confirm.

Adding a Follower Node

1. In the navigation panel, go to Infrastructure → Clusters.
2. Click the name of the BiHA cluster.
3. In the top-right corner of the page, click Edit topology.
4. Click Add server.
5. From Server, select the server where the instance is installed.

When the server is selected, the following instance information is displayed:

- Data catalog: The path to the server directory where the main instance directories and files are located.
- Network address and Port: The network address and port number that the instance uses to accept client connections.

6. Click Save.

Deleting a Follower Node

Important

- You can only delete follower nodes. To delete the leader node, [select a new leader node](#) first.
- Deleted follower nodes cannot be restored.

1. In the navigation panel, go to Infrastructure → Clusters.
2. Click the name of the BiHA cluster.
3. Click  next to the follower node.
4. Select one of the following values:
 - Update configuration and remove node from the cluster: Delete the follower node, but keep the instance in the `Started` state.
 - Stop and remove from the cluster: Delete the follower node, but keep the instance in the `Stopped` state.
 - Remove with all data: Delete the follower node and instance.
5. Confirm the operation and click Delete.

You can also delete follower nodes when editing the topology of the BiHA cluster. To do this, in the top-right corner of the page, click Edit topology.

5.7. Configuration Presets

Configuration presets are sets of Postgres Pro configuration parameters that apply to [instances](#). Once a configuration preset is applied to an instance, the corresponding values are automatically specified for its parameters.

You can apply a configuration preset to an instance when [creating the instance](#) or apply a new configuration preset to the [already existing instance](#). A configuration preset can also be [applied to multiple instances at once](#).

The following system presets exist in PPEM:

- Settings for 1C
- Settings for OLTP

This section explains how to manage configuration presets. It includes the following instructions:

- [Creating a Configuration Preset](#)
- [Viewing Configuration Presets](#)
- [Editing a Configuration Preset](#)
- [Deleting a Configuration Preset](#)

Creating a Configuration Preset

1. In the navigation panel, go to Settings → Configuration presets.
2. In the top-right corner of the page, click Create preset.
3. Enter parameters of the new configuration preset (parameters marked with an asterisk are required):
 - Name.
 - Description: The short description of the configuration preset.
The maximum recommended length is 20–30 characters.
 - Note: The full description of the configuration preset.
 - Configuration parameters: The configuration parameters that will be added to the preset.
Enter the configuration parameter and its value.
You can add multiple configuration parameters by clicking Add.
4. Click Create preset.

Viewing Configuration Presets

In the navigation panel, go to Settings → Configuration presets.

The table of configuration presets with the following columns will be displayed:

- Name: The name and full description of the configuration preset.
- Configuration parameters: The number of configuration parameters in the preset.

You can view the full list of configuration parameters and their short descriptions by clicking  next to the configuration preset name.

- Title: The short description of the configuration preset.
- Actions.

For more information about available actions, refer to other instructions in this section.

Editing a Configuration Preset

Important

System configuration presets cannot be edited.

To edit a configuration preset:

1. In the navigation panel, go to Settings → Configuration presets.

2. Click  next to the configuration preset.
3. Edit configuration preset parameters.
4. Click Apply changes.

Deleting a Configuration Preset

Important

- System configuration presets cannot be deleted.
- Deleted user configuration presets cannot be restored.

To delete a configuration preset:

1. In the navigation panel, go to Settings → Configuration presets.
2. Click  next to the configuration preset.
3. Confirm the operation and click Delete.

5.8. Tablespaces

Tablespaces allow you to manage how database object files are located within the file system. A tablespace can store individual tables, indexes, and entire databases.

For more information about tablespaces, refer to the [official Postgres Pro documentation](#).

This section explains how to manage tablespaces. It includes the following instructions:

- [Creating a Tablespace](#)
- [Viewing Tablespaces](#)
- [Renaming a Tablespace](#)
- [Deleting a Tablespace](#)

Creating a Tablespace

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Tablespaces.
4. In the right-top corner of the page, click Create space.
5. Enter parameters of the new tablespace (parameters marked with an asterisk are required):
 - Instance: The instance to which the tablespace will be assigned.
The value is entered automatically.
 - Name.
 - Catalog: The path to the directory where the tablespace will be located.
 - Owner: The DBMS user who will own the tablespace.
 - seq_page_cost: The value of the `seq_page_cost` parameter that will be assigned to the tablespace.
 - random_page_cost: The value of the `random_page_cost` parameter that will be assigned to the tablespace.
 - effective_io_concurrency: The value of the `effective_io_concurrency` parameter that will be assigned to the tablespace.

- `maintenance_io_concurrency`: The value of the `maintenance_io_concurrency` parameter that will be assigned to the tablespace.
- `compression`: The use of the *Compressed File System*.

Possible values:

- `zstd`
- `lz4`
- `pglz`
- `zlib`

This functionality is available only in the Postgres Pro Enterprise edition.

When creating a tablespace, a directory is created in the file system, and the tablespace is then created using the `CREATE TABLESPACE` command. You can also create a tablespace in an existing directory, provided the directory is empty.

6. Click Save.

Viewing Tablespaces

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Tablespaces.

The table of tablespaces with the following columns will be displayed:

- Name.
- Instance: The instance where the tablespace is created.
- Owner: The DBMS user who owns the tablespace.
- Path: The directory of the tablespace in the file system.
- Configuration parameters: Additional configuration parameters specific to tablespaces.
- Size: The total size of objects within the tablespace.
- Actions.

For more information about available actions, refer to other instructions in this section.

Renaming a Tablespace

Tablespaces can be renamed. When renaming a tablespace, the `ALTER TABLESPACE` SQL command is executed.

Important

System tablespaces cannot be renamed.

To rename a tablespace:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Tablespaces.
4. Click  next to the tablespace.
5. Specify the new tablespace name.

6. Click Save.

Deleting a Tablespace

When deleting a tablespace, the `DROP TABLESPACE` SQL command is executed.

Important

- Deleted tablespaces cannot be restored.
- System tablespaces cannot be deleted.
- You can only delete empty tablespaces.

To delete a tablespace:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Tablespaces.
4. Click  next to the tablespace.
5. Click Delete.

5.9. Databases

This section explains how to manage databases. It contains the following instructions:

- [Creating a Database](#)
- [Viewing Databases](#)
- [Starting the psql Terminal](#)
- [Collecting the Planner Statistics for a Database](#)
- [Reindexing a Database](#)
- [Vacuuming a Database](#)
- [Renaming a Database](#)
- [Deleting a Database](#)

Creating a Database

1. In the navigation panel, go to Databases.
2. In the top-right corner of the page, click Create database.
3. Enter parameters of the new database (parameters marked with an asterisk are required):
 - Instance: The instance where the database will be created.
 - Database name.
 - Database owner.
 - Encoding (ENCODING): The character encoding in the database.
 - Sorting category (LC_COLLATE): Specifies the `LC_COLLATE` value in the operating system environment of the database server.
 - Category of symbol types (LC_CTYPE): Specifies the `LC_CTYPE` value in the operating system environment of the database server.
 - Tablespace: The [tablespace](#) where the database will be created.
 - Allow connections: Specifies whether the database will be available for connections.
 - Limiting the number of connections: The maximum number of simultaneous database sessions.
 - Is template: Specifies whether the created database will be a template.

- Template: The template of the database.

4. Click Save.

Viewing Databases

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.

The table of databases includes the following columns:

- DB name.
- Total size.

The horizontal indicator displays the size of tables, bloat, and indexes.

- Wraparound: The percentage of released transaction IDs in the database.
- Instance: The instance where the database is created.
- Data collecting: The amount of the instance service information collected by [agents](#).
- Tables:
 - the number of tables
 - the size of tables
 - the table bloat percentage
- Indexes:
 - the number of indexes
 - the size of indexes
 - the index bloat percentage
- Actions.

For more information about available actions, refer to other instructions in this section.

The database size is calculated as the total sum of all its relational objects. This allows avoiding redundant calls of the `pg_database_size` function that uses the recursive traversal of directories and files and can negatively impact the performance of the main instance workload.

Important

If a database is created outside PPEM, the information about it can be displayed in the [web application](#) with a delay. The delay duration depends on the agent configuration. By default, it is one minute.

Starting the psql Terminal

The psql terminal provides low-level access to the DBMS instance and can be used when PPEM features are not sufficient. Although providing most of the standard `psql` functionality, it has certain limitations. For example, you cannot use the `\!` meta-command to run shell commands.

The DBMS instance connection parameters are used when the terminal is started.

To start the psql terminal:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click PSQL next to the database.

Collecting the Planner Statistics for a Database

The `ANALYZE` SQL command is executed when collecting the planner statistics.

To collect the planner statistics for a database:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click  → Analyze next to the database.
4. Specify the statistics collection parameters (parameters marked with an asterisk are required):

- Set cron-string execution: Allows specifying the time interval for collecting statistics in the crontab format.

If you turn on this toggle, in Execution, enter the string.

- Task planning: The type of the statistics collection task.

Possible values:

- Execute now: The statistics will be collected immediately after you perform this instruction.
- Time-delayed: The statistics will be collected at the specified date and time.
- On schedule: The statistics will be collected at the specified time interval.

For this value, specify the following parameters:

- Interval: The unit of measurement of the time interval.

Possible values:

- Minutes
- Hours
- Days
- Repeat every: The time interval for collecting statistics by minutes or hours.

This parameter is available only if you select Minutes or Hours from Interval.

- Execution days: The days when statistics will be collected.
- Cron total line: The string in the crontab format that specifies the time interval for collecting statistics.

The value is entered automatically.

This parameter is available only if you turn off Set cron-string execution.

- Name: The unique name of the statistics collection task.

This parameter is available only if you turn on Set cron-string execution or select Time-delayed / On schedule from Task planning.

- Time: The date and/or time when statistics will be collected.

This parameter is available only if you select Time-delayed from Task planning or Days from Interval.

- Start and Repeat untill: The start and end date and time for the statistics collection.

These parameters are available only if you turn on Set cron-string execution or select On schedule from Task planning.

5. Click Apply.
6. Confirm the operation.

The statistics collection will start in the asynchronous mode.

Note

Depending on the number of tables and columns, this operation may require significant resources so it is recommended to [track](#) it.

Reindexing a Database

The `REINDEX` SQL command is executed when reindexing.

To reindex a database:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3.  Click  → Reindex next to the database.
4. Specify the reindexing parameters (parameters marked with an asterisk are required):
 - Set cron-string execution: Allows specifying the time interval for reindexing in the crontab format.

If you turn on this toggle, in Execution, enter the string.

- Task planning: The type of the reindexing task.

Possible values:

- Execute now: Reindexing will be performed immediately after you perform this instruction.
- Time-delayed: Reindexing will be performed at the specified date and time.
- On schedule: Reindexing will be performed at the specified time interval.

For this value, specify the following parameters:

- Interval: The unit of measurement of the time interval.

Possible values:

- Minutes
- Hours
- Days
- Repeat every: The time interval for reindexing by minutes or hours.

This parameter is available only if you select Minutes or Hours from Interval.

- Execution days: The days when reindexing will be performed.
- Cron total line: The string in the crontab format that specifies the time interval for reindexing.

The value is entered automatically.

This parameter is available only if you turn off Set cron-string execution.

- Name: The unique name of the reindexing task.

This parameter is available only if you turn on Set cron-string execution or select Time-delayed / On schedule from Task planning.

- Time: The date and/or time when reindexing will be performed.

This parameter is available only if you select Time-delayed from Task planning or Days from Interval.

- Start and Repeat until: The start and end date and time for reindexing.

These parameters are available only if you turn on Set cron-string execution or select On schedule from Task planning.

5. Confirm the operation.

Note

Depending on the number and size of indexes, this operation may require significant resources so it is recommended to [track](#) it.

Vacuuming a Database

The `VACUUM` SQL command is executed when vacuuming.

To vacuum a database:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click  → Vacuum next to the database.
4. Specify the vacuum parameters (parameters marked with an asterisk are required):

- Modes: The vacuum modes.

Possible values:

- Analyze: Updates the statistics that the planner uses for selecting the most efficient way of query execution.
- Full: Performs a full vacuum via reindexing all table files.

Note

While the vacuum is being performed in this mode, all sessions working with tables that are being reindexed will be blocked.

- Freeze: Performs an aggressive tuple freeze to expand freezing and release transaction IDs for further reuse.

The aggressive freeze is always performed when overwriting the table, so do not select Freeze if Full is already selected.

- Set cron-string execution: Allows specifying the time interval for vacuuming in the crontab format.

If you turn on this toggle, in Execution, enter the string.

- Task planning: The type of the vacuuming task.

Possible values:

- Execute now: Vacuuming will be performed immediately after you perform this instruction.

- Time-delayed: Vacuuming will be performed at the specified date and time.
- On schedule: Vacuuming will be performed at the specified time interval.

For this value, specify the following parameters:

- Interval: The unit of measurement of the time interval.

Possible values:

- Minutes
- Hours
- Days
- Repeat every: The time interval for vacuuming by minutes or hours.

This parameter is available only if you select Minutes or Hours from Interval.

- Execution days: The days when vacuuming will be performed.
- Cron total line: The string in the crontab format that specifies the time interval for vacuuming.

The value is entered automatically.

This parameter is available only if you turn off Set cron-string execution.

- Name: The unique name of the vacuuming task.

This parameter is available only if you turn on Set cron-string execution or select Time-delayed / On schedule from Task planning.

- Time: The date and/or time when vacuuming will be performed.

This parameter is available only if you select Time-delayed from Task planning or Days from Interval.

- Start and Repeat untill: The start and end date and time for vacuuming.

These parameters are available only if you turn on Set cron-string execution or select On schedule from Task planning.

5. Click Apply.
6. Confirm the operation.

Note

Depending on the number and size of tables, this operation may require significant resources so it is recommended to [track](#) it.

Renaming a Database

The `ALTER DATABASE` SQL command is executed with the `RENAME TO` clause when renaming a database.

To rename a database:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click  → Rename next to the database.

4. Specify the new database name.
5. Click Save.

Deleting a Database

The `DROP DATABASE` SQL command is executed when deleting a database.

Important

Deleted databases cannot be restored.

To delete a database:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3.  → Drop next to the database.
4. (Optional) To force delete the database, enable Forced deletion.

If you enable this checkbox, all database connections will be terminated before deleting the database.

The database will not be deleted if there are prepared transactions, logical replication slots, or subscriptions.

5. Click Delete.
6. Confirm the operation.

5.10. Schemas

This section explains how to manage schemas. It contains the following instructions:

- [Creating a Schema](#)
- [Viewing Schemas](#)
- [Editing a Schema](#)
- [Deleting a Schema](#)

Creating a Schema

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. In the top-right corner of the page, click Create schema.
5. Enter parameters of the new schema (parameters marked with an asterisk are required):
 - Name.
 - Owner.
 - `lock_timeout, s`: Abort any statement that waits longer than the specified time while attempting to acquire a lock on a table, index, row, or other database object. The time limit applies separately to each lock acquisition attempt.

For more information about this parameter, refer to the [official Postgres Pro documentation](#).

6. Click Create.

Viewing Schemas

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.

The table of schemas with the following columns will be displayed:

- Schemas: The unique name of the schema.
- Total size.

The horizontal indicator displays the size of tables, bloat, and indexes.

- Owner.
- Tables:
 - the number of tables
 - the size of tables
 - the table bloat percentage
- Indexes:
 - the number of indexes
 - the size of indexes
 - the index bloat percentage
- Actions.

For more information about available actions, refer to other instructions in this section.

The size of the schema objects is calculated as the total sum of all its relational objects.

Important

If a schema is created outside PPEM, the information about it can be displayed in the [web application](#) with a delay. The delay duration depends on the agent configuration. By default, it is one minute.

Editing a Schema

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.

4. Click  next to the schema.

5. Edit schema parameters.
6. Click Save.

Deleting a Schema

The `DROP SCHEMA` SQL command is executed when deleting a schema.

Important

Deleted schemas cannot be restored.

To delete a schema:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click  next to the schema.
5. (Optional) To delete all dependent objects of the schema, turn on Cascade deletion.
6. Click Delete.
7. Confirm the operation.

5.11. Tables

This section explains how to manage tables. It contains the following instructions:

- [Creating a Table](#)
- [Viewing Tables](#)
- [Collecting the Planner Statistics for a Table](#)
- [Reindexing a Table](#)
- [Vacuuming a Table](#)
- [Viewing Table Constraints](#)
- [Viewing Table Storage Layers](#)
- [Editing a Table](#)
- [Viewing Table Columns](#)
- [Editing a Table Column](#)
- [Deleting a Table Column](#)
- [Deleting a Table](#)

Creating a Table

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. In the top-right corner of the page, click Create table.
6. Enter parameters of the new table (parameters marked with an asterisk are required):
 - Name.
 - Tablespace: The tablespace where the table will be created.
 - Columns: Columns of the created table.

To add a column:

- a. Click Add +.
- b. Enter parameters of the new column (parameters marked with an asterisk are required):
 - Name.
 - Type: The type of the column data.
 - NOT NULL: Specifies whether the column accepts `NULL` values.
 - Default value: The default value of the column.

- **Primary key:** Specifies whether the column can contain only unique (non-repeating) values different from `NULL`.

To add a primary key *storage parameter*, click Add + and specify the required parameters:

- **Name:** The name of the primary key storage parameter.

Possible values:

- *fillfactor*
- *deduplicate_items*

- **Value:** The value of the primary key storage parameter.

- **Unique key:** Specifies whether the column can contain only unique values.

To add a unique key *storage parameter*, click Add + and specify the required parameters:

- **Name:** The name of the unique key storage parameter.

Possible values:

- *fillfactor*
- *deduplicate_items*

- **Value:** The value of the unique key storage parameter.

- **Check expression:** The constraint of the column.
- **Compression method:** The compression method for the column.

Possible values:

- *pglz*
- *lz4*

c. Click Save.

- **Storage parameters:** *Storage parameters* of the table.

To add a storage parameter:

- a. Click Add +.
- b. Select the parameter.
- c. Specify the value.

- **UNLOGGED:** Specifies whether the created table is unlogged.
- **lock_timeout, s:** Abort any statement that waits longer than the specified time while attempting to acquire a lock on a table, index, row, or other database object. The time limit applies separately to each lock acquisition attempt.

For more information about this parameter, refer to the *official Postgres Pro documentation*.

7. Click Save.

Viewing Tables

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.

4. Click the name of the schema.

The schema page with the selected Tables tab will be displayed. This tab includes the following columns:

- Tables: The unique name of the table within the schema.
- Total size: The size of the table in bytes.

The horizontal indicator displays the following:

- the size of the table user data and bloating
- the total size of all table indexes, including the index bloat
- the size of the service TOAST storage without bloating
- CFS: Specifies whether the *CFS* compression is used for the table. Applicable only for the Postgres Pro Enterprise edition.
- Data: The size of data in bytes, with the bloat percentage indicated separately.
- Indexes: The total number of indexes, size of indexes in bytes, and bloating percentage.
- TOAST: The size of the service TOAST storage without bloating.
- Actions.

For more information about available actions, refer to other instructions in this section.

Collecting the Planner Statistics for a Table

The *ANALYZE* SQL command is executed when collecting the planner statistics.

To collect the planner statistics for a table:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5.  → Analyze next to the table.
6. Specify the statistics collection parameters (parameters marked with an asterisk are required):

- Set cron-string execution: Allows specifying the time interval for collecting statistics in the crontab format.

If you turn on this toggle, in Execution, enter the string.

- Task planning: The type of the statistics collection task.

Possible values:

- Execute now: The statistics will be collected immediately after you perform this instruction.
- Time-delayed: The statistics will be collected at the specified date and time.
- On schedule: The statistics will be collected at the specified time interval.

For this value, specify the following parameters:

- Interval: The unit of measurement of the time interval.

Possible values:

- Minutes

- Hours
- Days
- Repeat every: The time interval for collecting statistics by minutes or hours.
This parameter is available only if you select Minutes or Hours from Interval.
- Execution days: The days when statistics will be collected.
- Cron total line: The string in the crontab format that specifies the time interval for collecting statistics.

The value is entered automatically.

This parameter is available only if you turn off Set cron-string execution.

- Name: The unique name of the statistics collection task.

This parameter is available only if you turn on Set cron-string execution or select Time-delayed / On schedule from Task planning.

- Time: The date and/or time when statistics will be collected.

This parameter is available only if you select Time-delayed from Task planning or Days from Interval.

- Start and Repeat untill: The start and end date and time for the statistics collection.

These parameters are available only if you turn on Set cron-string execution or select On schedule from Task planning.

7. Click Apply.

8. Confirm the operation.

The statistics collection will start in the asynchronous mode.

Note

Depending on the number of tables and columns, this operation may require significant resources so it is recommended to [track](#) it.

Reindexing a Table

The `REINDEX` SQL command is executed when reindexing.

To reindex a table:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Click  → Reindex next to the table.
6. Specify the reindexing parameters (parameters marked with an asterisk are required):
 - Set cron-string execution: Allows specifying the time interval for reindexing in the crontab format.

If you turn on this toggle, in Execution, enter the string.

- Task planning: The type of the reindexing task.

Possible values:

- Execute now: Reindexing will be performed immediately after you perform this instruction.
- Time-delayed: Reindexing will be performed at the specified date and time.
- On schedule: Reindexing will be performed at the specified time interval.

For this value, specify the following parameters:

- Interval: The unit of measurement of the time interval.

Possible values:

- Minutes
- Hours
- Days
- Repeat every: The time interval for reindexing by minutes or hours.

This parameter is available only if you select Minutes or Hours from Interval.

- Execution days: The days when reindexing will be performed.
- Cron total line: The string in the crontab format that specifies the time interval for reindexing.

The value is entered automatically.

This parameter is available only if you turn off Set cron-string execution.

- Name: The unique name of the reindexing task.

This parameter is available only if you turn on Set cron-string execution or select Time-delayed / On schedule from Task planning.

- Time: The date and/or time when reindexing will be performed.

This parameter is available only if you select Time-delayed from Task planning or Days from Interval.

- Start and Repeat until: The start and end date and time for reindexing.

These parameters are available only if you turn on Set cron-string execution or select On schedule from Task planning.

7. Confirm the operation.

The reindexing task will start.

Note

Depending on the number and size of indexes, this operation may require significant resources so it is recommended to [track](#) it.

Vacuuming a Table

The `VACUUM` SQL command is executed when vacuuming.

To vacuum a table:

1. In the navigation panel, go to Databases.

2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Click  → Vacuum next to the table.
6. Specify the vacuum parameters (parameters marked with an asterisk are required):

- Modes: The vacuum modes.

Possible values:

- Analyze: Updates the statistics that the planner uses for selecting the most efficient way of query execution.
- Full: Performs a full vacuum via reindexing all table files.

Note

While the vacuum is being performed in this mode, all sessions working with tables that are being reindexed will be blocked.

- Freeze: Performs an aggressive tuple freeze to expand freezing and release transaction IDs for further reuse.

The aggressive freeze is always performed when overwriting the table, so do not select Freeze if Full is already selected.

- Set cron-string execution: Allows specifying the time interval for vacuuming in the crontab format.

If you turn on this toggle, in Execution, enter the string.

- Task planning: The type of the vacuuming task.

Possible values:

- Execute now: Vacuuming will be performed immediately after you perform this instruction.
- Time-delayed: Vacuuming will be performed at the specified date and time.
- On schedule: Vacuuming will be performed at the specified time interval.

For this value, specify the following parameters:

- Interval: The unit of measurement of the time interval.

Possible values:

- Minutes
- Hours
- Days
- Repeat every: The time interval for vacuuming by minutes or hours.

This parameter is available only if you select Minutes or Hours from Interval.

- Execution days: The days when vacuuming will be performed.
- Cron total line: The string in the crontab format that specifies the time interval for vacuuming.

The value is entered automatically.

This parameter is available only if you turn off Set cron-string execution.

- Name: The unique name of the vacuuming task.

This parameter is available only if you turn on Set cron-string execution or select Time-delayed / On schedule from Task planning.

- Time: The date and/or time when vacuuming will be performed.

This parameter is available only if you select Time-delayed from Task planning or Days from Interval.

- Start and Repeat until: The start and end date and time for vacuuming.

These parameters are available only if you turn on Set cron-string execution or select On schedule from Task planning.

7. Click Apply.

8. Confirm the operation.

Note

Depending on the number and size of tables, this operation may require significant resources so it is recommended to [track](#) it.

Viewing Table Constraints

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Click the name of the table.
6. Select Constraints.

The table of constraints with the following columns will be displayed:

- Name.
- Definition.

Viewing Table Storage Layers

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Click the name of the table.
6. Select Storage.

The table of storage layers with the following columns will be displayed:

- Size.
- Tablespace: The tablespace where the storage layer is located.

- File path: The path to the storage layer file.

The following storage layers are displayed in the table:

- Main data: The layer for storing the main user data.
- Free space map: The service layer for storing information about free space segments of the main storage layer.
- Visibility map: The service layer for storing information about visible rows of the main storage layer.
- CFM: The service layer for the [CFS](#) compression.
- TOAST: The service layer for storing large values exceeding the standard page data storage restrictions.

Editing a Table

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Click  next to the table.
6. Edit table parameters.

Available storage parameters are listed in [Table Storage Parameters](#).

7. Click Save.

Viewing Table Columns

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Click the name of the table.

The table page with the selected Structure tab will be displayed. This tab includes the following columns:

- Number: The ordinal number of the table column.
- Name.
- Type: The type of the table column data.
- Nullable: Specifies whether the table column can contain `NULL` values.
- Default value: The default value of the table column.
- Description.
- Actions.

For more information about available actions, refer to other instructions in this section.

Editing a Table Column

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.

4. Click the name of the schema.
5. Click the name of the table.
6. Click  next to the table column.
7. Edit table column parameters.
8. Click Save.

Deleting a Table Column

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Click the name of the table.
6. Click  next to the table column.
7. Click Delete.

Deleting a Table

Important

Deleted tables cannot be restored.

To delete a table:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Click  next to the table.
6. Click Delete.

Table Storage Parameters

You can specify the following *storage parameters* when *creating* or *editing* tables:

- *fillfactor* (integer)
- *toast_tuple_target* (integer)
- *parallel_workers* (integer)
- *autovacuum_enabled* (boolean)
- *toast.autovacuum_enabled* (boolean)
- *vacuum_index_cleanup* (enum)
- *toast.vacuum_index_cleanup* (enum)
- *vacuum_truncate* (boolean)
- *toast.vacuum_truncate* (boolean)
- *autovacuum_vacuum_threshold* (integer)
- *toast.autovacuum_vacuum_threshold* (integer)

- `autovacuum_vacuum_scale_factor` (floating point)
- `toast.autovacuum_vacuum_scale_factor` (floating point)
- `autovacuum_analyze_threshold` (integer)
- `autovacuum_analyze_scale_factor` (floating point)
- `autovacuum_vacuum_cost_delay` (floating point)
- `toast.autovacuum_vacuum_cost_delay` (floating point)
- `autovacuum_vacuum_cost_limit` (integer)
- `toast.autovacuum_vacuum_cost_limit` (integer)
- `autovacuum_freeze_min_age` (integer)
- `toast.autovacuum_freeze_min_age` (integer)
- `autovacuum_freeze_max_age` (integer)
- `toast.autovacuum_freeze_max_age` (integer)
- `autovacuum_freeze_table_age` (integer)
- `toast.autovacuum_freeze_table_age` (integer)
- `autovacuum_multixact_freeze_min_age` (integer)
- `toast.autovacuum_multixact_freeze_min_age` (integer)
- `autovacuum_multixact_freeze_max_age` (integer)
- `toast.autovacuum_multixact_freeze_max_age` (integer)
- `autovacuum_multixact_freeze_table_age` (integer)
- `toast.autovacuum_multixact_freeze_table_age` (integer)
- `log_autovacuum_min_duration` (integer)
- `toast.log_autovacuum_min_duration` (integer)
- `user_catalog_table` (boolean)

5.12. Indexes

This section explains how to manage indexes. It contains the following instructions:

- [Creating an Index](#)
- [Viewing Indexes](#)
- [Reindexing an Index](#)
- [Editing an Index](#)
- [Deleting an Index](#)

Creating an Index

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Click the name of the table.
6. In the top-right corner of the page, click Create index.
7. Enter parameters of the new index (parameters marked with an asterisk are required):
 - Name.
 - Concurrently building: Specifies whether the index will be created in a special mode that minimizes the number of locks and reduces the risk of their occurrence when there are concurrent workloads.
 - Unique index: Specifies whether to control duplicate table values when creating the index and adding new values to it.
 - Columns: Columns that will be included in the index.

To add a column:

- a. Click Add column +.
- b. Enter parameters of the new column (parameters marked with an asterisk are required):
 - Column: The name of the column.
 - Operator class: The operators that will be used by the index for the column.
 - Collation: The collation for the column.
 - Order: The sorting order for the column.

Possible values:

- Default
- Ascending
- Descending
- NULL's order: The sorting order of the `NULL` values.

Possible values:

- Default
- NULL first
- NULL last

The Operator class, Collation, Order, and NULL's order parameters are available only if you turn on Extended settings.

8. (Optional) In the top-right corner of the window, turn on Extended settings and specify additional parameters:
 - Tablespace: The tablespace where the index will be located.
 - Using method: The access method.

Possible values:

- btree
- gin
- gist
- brin
- hash
- Where predicate: The condition of the index.
- lock_timeout, s: Abort any statement that waits longer than the specified time while attempting to acquire a lock on a table, index, row, or other database object. The time limit applies separately to each lock acquisition attempt.

For more information about this parameter, refer to the [official Postgres Pro documentation](#).

9. Click Create.

Viewing Indexes

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.

5. Click the name of the table.
6. Select Indexes.

The table of indexes with the following columns will be displayed:

- Indexes: The unique name of the index.
- Total size.
- CFS: Specifies whether the *CFS* compression is used for the index.
- Status.

Possible values:

- valid: The standard state in which the index can be used in queries.
- invalid: The index cannot be used in queries. This status indicates that the index is being created or was damaged.
- Actions.

For more information about available actions, refer to other instructions in this section.

Important

If an index is created outside PPEM, the information about it can be displayed in the [web application](#) with a delay. The delay duration depends on the [agent](#) configuration. By default, it is one minute.

Reindexing an Index

The *REINDEX* SQL command is executed when reindexing.

To reindex an index:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Click the name of the table.
6. Select Indexes.
7. Click Reindex next to the index.
8. Specify the reindexing parameters (parameters marked with an asterisk are required):
 - Concurrently: Specifies whether the operation will be performed in the non-locking mode. This is the safest mode that allows avoiding locking other sessions by slowing down the process of reindexing.

Possible values:

- On
 - Off
 - Tablespace: The tablespace where the index will be reindexed.
9. Click Execute.

The reindexing task will start.

Note

Depending on the number and size of indexes, this operation may require significant resources so it is recommended to [track](#) it.

Editing an Index

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Click the name of the table.
6. Select Indexes.
7. Click  next to the index.
8. Edit index parameters.
9. Click Save.

Deleting an Index**Important**

Deleted indexes cannot be restored.

To delete an index:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Click the name of the table.
6. Select Indexes.
7. Click  next to the index.
8. Click Delete.

5.13. Functions

This section explains how to manage functions. It contains the following instructions:

- [Creating a Function](#)
- [Viewing Functions](#)
- [Editing a Function](#)
- [Deleting a Function](#)

Creating a Function

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.

3. Click the name of the database.
4. Click the name of the schema.
5. Select Functions.
6. In the top-right corner of the page, click Create function.
7. Enter parameters of the new function (parameters marked with an asterisk are required):

- Name.
- Language: The procedural language of the function.
- Arguments.

To add an argument:

- a. Click Add argument +.
- b. Enter parameters of the new argument (parameters marked with an asterisk are required):

- Name.
- Argument mode.

Possible values:

- IN
 - OUT
 - INOUT
 - VARIADIC
 - Type: The argument data type.
 - Default value: The default value of the argument.
- Function return table: Specifies whether the function returns a table.

If turn on this toggle, add a return value:

- a. Click Add return value +.
- b. Enter parameters of the new return value (parameters marked with an asterisk are required):

- Name.
- Type.

- Return value type: The data type of the value that the function will return.

This parameter is available only if you turn off Function return table.

- Function body: The body of the function in the selected procedural language.
- Window: Specifies whether the function is a window function.
- Optimizer: The attribute that informs the optimizer about the behavior of the function.

Possible values:

- Default: The default value is Volatile.
- Immutable: The function is *immutable* and it cannot modify the database and always returns the same result for specific arguments. It means that the function does not interact with the database and does not use the information that was not passed to the argument list.

If a function is immutable, any function call with constant arguments can be immediately replaced with the function value.

- **Stable:** The function is *stable* and it cannot modify the database and always returns the same result for specific arguments within one table scan, but the result can differ for different SQL operators.

This is applicable to functions whose results depend on the database contents and parameters, such as time zone. However, this is not applicable to `AFTER` triggers that are trying to read the rows changed by the current command.

Note

The `current_timestamp` functions are also considered stable, since their results do not change in the transaction.

- **Volatile:** The function is *volatile* and its result can change even within one table scan, so the function calls cannot be optimized.

Only a limited number of functions are volatile, for example, `random()`, `curval()`, and `timeofday()`.

Note

Any function that has side effects must be considered volatile even if its results are predictable so that its calls are not optimized. An example of such a function is `set-val()`.

- **Strict:** Specifies whether the function will always return `NULL` if `NULL` is passed in one of the arguments.
- **Leakproof:** Specifies whether the function does not have any side effects and it does not reveal any information about its arguments and only returns the result.
- **Security:** The privileges that will be used for calling and executing the function.

Possible values:

- **Default:** The default value is `Invoker`.
- **Invoker:** The function will be executed with the privileges of the user who called it.
- **Definer:** The function will be executed with the privileges of the user who defined it.
- **Parallel:** The parameters for calling the function in the parallel mode.

Possible values:

- **Default:** The default value is `Unsafe`.
- **Unsafe:** The function cannot be executed in the parallel mode and having such a function in an SQL operator will lead to selecting a sequential query plan.
- **Restricted:** The function can be executed in the parallel mode, but only in the parallel group leader process.
- **Safe:** The function can be safely executed in the parallel mode with no restrictions, including parallel worker processes.

- `lock_timeout, s`: Abort any statement that waits longer than the specified time while attempting to acquire a lock on a table, index, row, or other database object. The time limit applies separately to each lock acquisition attempt.

For more information about this parameter, refer to the [official Postgres Pro documentation](#).

- **Show SQL**: Displays the SQL query for creating the function with the specified parameters.

8. Click Create.

Viewing Functions

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Select Functions.

The table of functions with the following columns will be displayed:

- **Name**: The unique name of the function or procedure.
- **Arguments**: Arguments with their data types passed to the function.
- **Return values**: Values with their data types returned by the function.
- **Access control list**: The function access privileges.
- **Actions**.

For more information about available actions, refer to other instructions in this section.

Editing a Function

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.
4. Click the name of the schema.
5. Select Functions.
6. Click  next to the function.
7. Edit function parameters.
8. Click Save.

Deleting a Function

Important

Deleted functions cannot be restored.

To delete a function:

1. In the navigation panel, go to Databases.
2. (Optional) To display system databases, turn on Show system databases.
3. Click the name of the database.

4. Click the name of the schema.
5. Select Functions.
6. Click  next to the function.
7. Click Delete.

5.14. Sequences

This section explains how to manage sequences. It contains the following instructions:

- [Creating a Sequence](#)
- [Viewing Sequences](#)
- [Editing a Sequence](#)
- [Deleting a Sequence](#)

Creating a Sequence

1. In the navigation panel, go to Databases.
2. Click the name of the database.
3. Click the name of the schema.
4. Select Sequences.
5. In the top-right corner of the page, click Create sequence.
6. In Name, enter the unique name of the sequence.
7. (Optional) Turn on Extended settings and specify additional parameters:
 - No logging: Specifies whether the changes of the created sequence will not be saved to WAL.
 - Cycle: Specifies whether the created sequence is cycled.
 - Data type.
Possible values:
 - Default: The default value is Bigint.
 - Smallint.
 - Integer.
 - Bigint.
 - Min. value: The minimum value that will be generated by the sequence.
 - Max. value: The maximum value that will be generated by the sequence.
 - Start: The start value of the sequence.
 - Step: The number that will be added to the current sequence value to receive a new value.
 - Cache: The number of sequence values that will be allocated and stored in memory for faster access.
 - lock_timeout, s: Abort any statement that waits longer than the specified time while attempting to acquire a lock on a table, index, row, or other database object. The time limit applies separately to each lock acquisition attempt.

For more information about this parameter, refer to the [official Postgres Pro documentation](#).

8. Click Create.

Viewing Sequences

1. In the navigation panel, go to Databases.

2. Click the name of the database.
3. Click the name of the schema.
4. Select Sequences.

The table of sequences with the following columns will be displayed:

- Name.
- Owner.
- Data type: The data type of the sequence values.
- Last_value: The last value of the sequence.
- Min. value and Max. value: The minimum and maximum possible sequence values.
- Step: The step for changing the sequence values.
- Cycle: Specifies whether the sequence is cycled and restarts upon reaching its limit.
- Cache: The number of sequence values that will be allocated and stored in memory for faster access.
- Last_value: The last value of the sequence.
- Actions.

For more information about available actions, refer to other instructions in this section.

Editing a Sequence

1. In the navigation panel, go to Databases.
2. Click the name of the database.
3. Click the name of the schema.
4. Select Sequences.
5. Click  next to the sequence.
6. Edit sequence parameters.
7. Click Save.

Deleting a Sequence

Important

Deleted sequences cannot be restored.

To delete a sequence:

1. In the navigation panel, go to Databases.
2. Click the name of the database.
3. Click the name of the schema.
4. Select Sequences.
5. Click  next to the sequence.
6. (Optional) To delete all sequence objects, turn on Cascade deletion.

7. Click Delete.

5.15. Control Center

PPEM supports the *control center* that you can use for monitoring the state of instances and troubleshooting them if required.

To view the state of instances using control center, in the navigation panel, go to Monitoring → Operation control center.

The control center with the following elements of the [web application](#) interface will be displayed:

- [List of all instances](#) (the left block)
- [Instance tiles](#) (upper block)
- [Instance performance indicators](#) (lower block)

List of All Instances

The list of installed instances is displayed in the left block. The color of the dot located next to each instance name indicates the instance status:

-  Green (): The instance is operating normally.
-  Red (): An error occurred during instance operation.
-  Gray (): Failed to determine the instance state.

Instance Tiles

Installed instances are displayed as tiles in the upper block. Each tile displays the instance uptime, name, and the server where it is installed.

The tile color indicates the instance status. The correlation between the instance status and tile color is the same as for the dots in the [list of all instances](#).

You can perform the following actions using an instance tile:

- Go to [instance management](#) by clicking  → Go to instance.
- Start the psql terminal of an instance by clicking  → Open PSQL terminal.
- Start a [stopped](#) instance by clicking  → Start instance.

Instance Performance Indicators

The lower block allows monitoring key instance performance indicators using graphs based on metrics collected by [pgpro-otel-collector](#). For the graphs to work properly, you must first [install and configure logging and monitoring tools](#).

To view a graph, in the [upper block](#) click the instance tile, and then in the top-right corner of the lower block, select the graph.

For a description of the graphs, refer to [Viewing Main Metrics](#).

You can perform the following actions using the icons in the top-right corner of graphs:

- To display the specified time period covered by a graph, click Select interval, and then select the time period on the graph.
- To reset the time period, click Reset.
- To download a graph in the PNG or CSV format, click Download .png or Download .csv.

5.16. Metrics

Key performance indicators of the system can be monitored using graphs based on [metrics](#) collected by [pgpro-otel-collector](#). For the graphs to work properly, you must first [install and configure logging and monitoring tools](#).

You can also access [SQL metrics](#) that are provided based on planning statistics and SQL statement execution statistics collected by the [pgpro_stats](#) extension.

Important

When [creating SQL metrics](#) use queries with aggregate functions, such as `COUNT`, `SUM`, and `AVG`. The instance can fail when using metrics based on queries returning more than one row or separate values, for example, `SELECT 1`.

Viewing Main Metrics

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Metrics.

The following graphs will be displayed:

- WAL Archiver: The number of archived WAL segments.
- Vacuum workers: The number of vacuum operations.
- Background Writes: Buffers: The data volume written on the background from shared cache to disk.
- Background Writes: Maxwritten/Fsync:
 - maxwritten: The number of times the background writer had to stop writing because the limit was reached.
 - fsync: The number of forced fsync calls.
- Background Writes: Checkpoints: The number of checkpoints.
- Background Writes: Checkpoints Write/Sync: The time spent writing and synchronizing blocks during execution of checkpoints.
- Instance: Connections: Connections established with the DBMS instance and their state.
- Instance: Blocks rate: The number of cache hits and misses that resulted in the need to read data from disk.
- Instance: Transactions rate: Transactional activity in a DBMS instance.
- Instance: Events: Deadlocks, replication conflicts, and checksum validation errors in the DBMS instance.
- Instance: Tuples: The number of rows read, inserted, updated, and deleted.
- Instance: Cache hit ratio: The proportion of cache hits relative to all shared cache accesses.
- Instance: Temp bytes written: The amount of data written to temporary files.

- Instance: Temp files: The number of temporary files written by the DBMS instance.
- Instance: Locks: Lock dynamics.
- WAL: Written bytes: Bytes written to WAL.
- System: Load Average: Average server load.
- System: Memory Usage: Usage of server memory.
- System: Swap Usage: Usage of swap space in the system.
- System: Processes: Usage of processes in the system.

SQL Metrics

This section explains how to manage SQL metrics. It contains the following instructions:

- [Creating an SQL Metric](#)
- [Viewing SQL Metrics](#)
- [Editing an SQL Metric](#)
- [Deleting an SQL Metric](#)

Creating an SQL Metric

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Metrics → SQL metrics.
4. In the top-right corner of the page, click Add SQL metric.
5. Enter parameters of the new SQL metric (parameters marked with an asterisk are required):
 - Name.
 - Database: The database where the query will be executed.
 - User: The system role on behalf of which the query will be executed.
 - Collection interval: The frequency of data collection.
 - Query: The SQL query based on which the metric is computed.

Important

Use queries with aggregate functions, such as `COUNT`, `SUM`, and `AVG`. The instance can fail when using metrics based on queries returning more than one row or separate values, for example, `SELECT 1`.

- Instance restart is required: Specifies whether to restart the instance.
6. Click Add.

Viewing SQL Metrics

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Metrics → SQL metrics.

The table of SQL metrics with the following columns will be displayed:

- ID.

- Name.
- Database: The database where the query is executed.
- User: The system role on behalf of which the query will be executed.
- Interval: The frequency of data collection.
- Last value: The result of the last query in the `JSON` format.
- Updated at: The time the data was last updated.
- Query: The SQL query based on which the metric is computed.
- Actions.

For more information about available actions, refer to other instructions in this section.

To view the history of collected data, click the metric name.

Editing an SQL Metric

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Metrics → SQL metrics.
4.  Click next to the SQL metric.
5. Edit SQL metric parameters.
6. Click Save.

Deleting an SQL Metric

Important

Deleted SQL metrics cannot be restored.

To delete an SQL metric:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Metrics → SQL metrics.
4.  Click next to the SQL metric.
5. Click Delete.

5.17. Alerts

PPEM supports sending alerts to [users](#) and [user groups](#) via email to monitor agents, hosts, and instances. To send alerts, you must configure an SMTP server.

For an alert to be sent, an alert trigger must fire. An *alert trigger* fires when the value of a metric is below, equal to, or higher than the specified threshold. Metrics are stored in *data sources*, for example, in the [repository](#) database.

To determine the alert trigger threshold, alert trigger rules are used. An *alert trigger rule* is a set of one or multiple conditions containing logical operators and values. For example, an alert trigger rule

condition can include the `>` logical operator and the `0` value. With such a rule, the alert trigger fires if the value of the specified metric exceeds `0`.

Relations between multiple alert trigger rule conditions are determined by the `AND` and `OR` logical connectives.

For alerts to work properly, you must first [install and configure logging and monitoring tools](#).

This section explains how to manage alerts. It includes the following instructions:

- [Pre-Configuring Alerts](#)
- [Creating an Alert](#)
- [Viewing Alerts](#)
- [Disabling and Enabling Alerts](#)
- [Editing Alert Recipients](#)
- [Deleting an Alert](#)

Important

The alerts functionality is in the beta phase. Currently, only `pgpro-otel-collector` metrics can be used for alert triggers. Other existing limitations are specified in the corresponding instructions of this section.

Pre-Configuring Alerts

To work with alerts, you must first pre-configure them in the `ppem-manager.yml` manager configuration file.

You can specify the following parameters:

```
alerts:
  metrics:
    request_chunk_size: number_of_instance_IDs
  scheduler:
    interval: interval_for_checking_new_alerts
    initial_delay: delay_for_starting_aler_scheduler
    timeout: timeout_for_updating_alert_trigger_rules
  notifier:
    num_workers: number_of_concurrent_workers
    worker_batch_size: number_of_alerts_in_one_batch
    worker_interval: interval_for_checking_new_alerts
    backoff_base: exponential_backoff_calculation_duration
    max_retries: maximum_number_of_alert_attempts
    notification_timeout: alert_timeout
    janitor_interval: janitor_worker_polling_interval
    stale_processing_timeout: stale_alert_processing_timeout
  email:
    is_enabled: true or false
  smtp:
    host: SMTP_server_hostname_or_IP
    port: SMTP_server_port
    username: username_for_SMTP_server_authentication
    password: password_for_SMTP_server_authentication
    from: alert_sender_email
```

```
timeout: SMTP_server_connection_timeout
use_starttls: true or false
use_ssl: true or false
tls:
  insecure_skip_verify: true or false
  root_ca_path: path_to_root_CA
```

Where:

- **metrics:** The parameters of sending requests to the metrics plugin.
 - **request_chunk_size:** The maximum number of instance IDs within one request.
Default value: 100.
- **scheduler:** The parameters of the scheduler that updates alerts in the manager memory.
 - **interval:** The time interval for the scheduler to check for new alerts to process.
Default value: 50s.
 - **initial_delay:** The delay before starting the scheduler for the first time after the start of PPEM.
Default value: 10s.
 - **timeout:** The scheduler timeout for updating alert trigger rules.
Default value: 10m.
- **notifier:** The parameters of the notifier that sends alerts.
 - **num_workers:** The number of concurrent workers that will send alerts.
Default value: 5.
 - **worker_batch_size:** The number of alerts processed by workers in one batch.
Default value: 20.
 - **worker_interval:** The polling interval for workers to check for new alerts in the [repository database](#).
Default value: 30s.
 - **backoff_base:** The base duration for the exponential backoff calculation when resending a failed alert.
The delay for resending the alert is calculated as:
$$\text{backoff_base} \times (2^{\text{number_of_retry_attempts}}).$$

Default value: 10s.
 - **max_retries:** The maximum number of attempts to resend a failed alert.
Default value: 3.
 - **notification_timeout:** The maximum amount of time for the notifier to wait for an alert to be sent before considering it failed.
Default value: 20s.
 - **janitor_interval:** The polling interval for the janitor worker that cleans alerts stuck in the processing state.

Default value: 1m.

- `stale_processing_timeout`: The amount of time after which alerts stuck in the processing state are considered stale and must be reset by the janitor worker.

Default value: 10m.

- `email`: The parameters of sending alerts via email.
- `is_enabled`: Specifies whether alerts are sent via email.

Possible values:

- `true`
- `false`

If `false` is specified, alerts are logged instead of being sent via email.

Default value: `false`.

- `smtp`: The parameters of the SMTP server used for sending alerts.
- `host`: The hostname or IP address of the SMTP server.

Default value: `localhost`.

- `port`: The port number of the SMTP server.

Default value: 25.

- `username`: The username for authenticating in the SMTP server.

Default value: `""`.

- `password`: The password for authenticating in the SMTP server.

Default value: `""`.

- `from`: The email address of the alert sender.

Default value: `admin@localdomain.local`.

- `timeout`: The SMTP server connection timeout.

Default value: 10s.

- `use_starttls`: Specifies whether the STARTTLS extension is used for securing the SMTP server connection.

Possible values:

- `true`
- `false`

Default value: `false`.

- `use_ssl`: Specifies whether the SSL/TLS protocol is used for the SMTP server connection.

Possible values:

- `true`
- `false`

Default value: `false`.

- `tls`: The TLS protocol parameters.
- `insecure_skip_verify`: Specifies whether the client skips the verification of the certificate chain and hostname of the SMTP server.

Possible values:

- `true`
- `false`

Default value: `false`.

Important

Setting this parameter to `true` represents a security risk. Do it only for testing purposes or with trusted networks.

- `root_ca_path`: The path to the CA certificate used for verifying the certificate of the SMTP server.

Default value: `""`.

Creating an Alert

1. In the navigation panel, go to Monitoring → Alerts.
2. In the top-right corner of the page, click Create alert.
3. Enter parameters of the new alert (parameters marked with an asterisk are required):

- Name.
- Datasource type: The type of the metrics that will be used for the alert trigger.

Currently, you can only use `pgpro-otel-collector` metrics.

- Datasource.
- State: The state of the alert after creation.

Possible values:

- Disabled
- Enabled
- Check interval, sec.: The time interval in seconds for verifying the data source of the alert trigger.

Minimum value: 60.

- Flap check, count: The number of repeated triggers required for stopping the alert.

0 means that this limitation is disabled.

- Notify after, sec.: The time in seconds during which the trigger must continually fire for the alert to be sent.
- Cooldown period, sec.: The time in seconds during which the alert will not be sent after the last firing trigger.

0 means that this limitation is disabled. 129

Note

Currently, you cannot change the values of Flap check, count, Notify after, sec., and Cooldown period, sec..

4. Click Next, and then specify additional parameters (parameters marked with an asterisk are required):
 - **Metric name:** The name of the metric without any additional characters that will be used for the alert trigger.

You can use the following pgpro-otel-collector metrics from the `monitoring.metrics` table of the [repository](#) database:

- `postgresql.archiver.archived_count`
- `postgresql.archiver.failed_count`
- `postgresql.bgwriter.buffers_checkpoint`
- `postgresql.bgwriter.buffers_clean`
- `postgresql.bgwriter.buffers_backend`
- `postgresql.bgwriter.buffers_allocated`
- `postgresql.bgwriter.maxwritten_clean`
- `postgresql.bgwriter.buffers_backend_fsync`
- `postgresql.bgwriter.checkpoints_requested`
- `postgresql.bgwriter.checkpoints_timed`
- `postgresql.bgwriter.checkpoint_sync_time_milliseconds`
- `postgresql.bgwriter.checkpoint_write_time_milliseconds`
- `postgresql.databases.blocks_hit`
- `postgresql.databases.blocks_read`
- `postgresql.databases.conflicts`
- `postgresql.databases.deadlocks`
- `postgresql.databases.checksum_failures`
- `postgresql.databases.tuples_fetched`
- `postgresql.databases.tuples_returned`
- `postgresql.databases.tuples_inserted`
- `postgresql.databases.tuples_updated`
- `postgresql.databases.tuples_deleted`
- `postgresql.databases.temp_bytes`
- `postgresql.databases.temp_files`
- `postgresql.wal.bytes`
- `postgresql.databases.rollback`
- `system.cpu.utilization`
- `system.memory.usage`

- `system.paging.usage`
- `postgresql.wal.records`
- `postgresql.databases.commits`
- **Operator • Threshold value:** The alert trigger rule condition containing a logical operator and value.

Possible logical operators:

- `= (eq)`
- `> (gt)`
- `>= (gte)`
- `< (lt)`
- `<= (lte)`
- `!= (neq)`

For example, if you select `>` and specify `0`, the alert is sent when the value of the specified metric exceeds `0`.

You can add multiple alert trigger rule conditions by clicking **Add**.

- **Rule condition:** The logical connectives for the specified alert trigger rule conditions.

Possible values:

- **AND**
- **OR**

This parameter is available only if you added multiple alert trigger rule conditions.

- **Instances to check.**

Possible values:

- **Check all.**
- **Select instances.**

For this value, from **Instances**, select the instances.

- **Notify users:** The [users](#) that will receive alerts.
- **Notify groups:** The [user groups](#) that will receive alerts.
- **Alert template:** The template of the alert text.

You can use the following variables in the alert text:

- `{{.Title}}`: The name of the metric used for the alert trigger.
- `{{.Timestamp}}`: The date and time when the alert trigger fired.
- `{{.Status}}`: The status of the alert trigger.
- **Notify resolved:** Specifies whether the alert is sent once the trigger is resolved.

Possible values:

- **Enabled.**

You can use the same variables in this alert text as in Alert template.

- Disabled.

5. Click Save.

Viewing Alerts

In the navigation panel, go to Monitoring → Alerts.

The table of alerts with the following columns will be displayed:

- Name.
- State.

Possible values:

- Enabled
- Disabled
- Source name: The data source of the alert trigger.

This column includes additional information:

Type: The type of the metrics used for the alert trigger.

Possible values:

- Repositories: System metrics.
- Metrics: pgpro-otel-collector metrics.
- Logs: pgpro-otel-collector logs.

This type of metrics is temporarily not supported.

- Interval, sec.: The time interval in seconds for verifying the data source of the alert trigger.
- Flap check, count: The number of repeated triggers required for stopping the alert.

0 means that this limitation is disabled.

- Notify after, sec.: The time in seconds during which the trigger must continually fire for the alert to be sent.
- Cooldown period, sec.: The time in seconds during which the alert is not sent after the last trigger.

0 means that this limitation is disabled.

- Recipients: The [users](#) that receive alerts.
- Group recipients: The [user groups](#) that receive alerts.
- Rule: The alert trigger rule conditions.

For example, if an alert trigger rule condition is `postgresql.up > 0`, the alert is sent when the value of the `postgresql.up` metric exceeds 0.

- Actions.

For more information about available actions, refer to other instructions in this section.

Disabling and Enabling Alerts

1. In the navigation panel, go to Monitoring → Alerts.

2. Click  or  next to the alert.

Editing Alert Recipients

1. In the navigation panel, go to Monitoring → Alerts.
2.  Click  next to the alert.
3. Edit [users](#) and [user groups](#) that receive alerts.
4. Click Save.

Deleting an Alert

Important

- System alerts cannot be deleted.
- Deleted alerts cannot be restored.

To delete an alert:

1. In the navigation panel, go to Monitoring → Alerts.
2.  Click  next to the alert.
3. Confirm the operation and click Delete.

5.18. Activity

5.18.1. User Sessions

You can view information about user sessions and background worker processes of an instance. The information is based on the [pg_stat_activity](#) view.

This section explains how to manage user sessions. It includes the following instructions:

- [Viewing User Sessions](#)
- [Viewing Statistics on Wait Events](#)
- [Viewing the Current Backend Query Locks](#)
- [Viewing the Current Backend Query Plan](#)
- [Canceling the Current Backend Query](#)
- [Terminating a User Session](#)

Viewing User Sessions

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.

The table of user sessions with the following columns will be displayed:

- `pid`: The backend ID.
- `leader_pid`: The backend ID for the process in the parallel query execution group.
- `State`: The type of the backend.

For more information about possible values, refer to the official Postgres Pro documentation on the [pg_stat_activity](#) module (see the `backend_type` column).

- `backend_start`: The backend start date and time.
- `client_hostname`, `client_addr`, and `client_port`: The network name, address, and port number of the client that initiated the user session.

- `usesysid`: The ID of the DBMS user under which the session is initiated.
- `username`: The name of the DBMS user under which the session is initiated.
- `datid`: The ID of the database associated with the user session.
- `database`: The name of the database associated with the user session.
- `application_name`: The name of the source application of the user session.
- `State`: The state of the backend.

For more information about possible values, refer to the official Postgres Pro documentation on the [pg_stat_activity](#) view (see the `state` column).

- `wait_event_type`: The type of the wait event for the backend.

For more information, refer to the official Postgres Pro documentation on the [pg_stat_activity](#) module (see the `wait_event_type` and `wait_event` columns).

- `wait_event`: The name of the wait event for the backend.
- `transaction_duration_seconds`: The duration of the current backend transaction, in seconds.
- `xact_start`: The start date and time of the current backend transaction.
- `query_duration_seconds`: The duration of the current backend query in seconds.
- `query_start`: The start date and time of the current backend query.
- `state_change`: The date and time of the last backend state update (see the `state` column).
- `backend_xid`: The ID of the backend top-level transaction.
- `backend_xmin`: The current `xmin` bound of the backend.
- `query_id`: The ID of the current or last backend query.
- `query`: The text of the current or last backend query.
- Actions.

For more information about available actions, refer to other instructions in this section.

Viewing Statistics on Wait Events

PPEM supports integration with the [pg_wait_sampling](#) extension for viewing the wait history and profile in user sessions.

Before performing this instruction, [install](#) the `pg_wait_sampling` extension.

To view statistics on wait events:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. In `pid`, click the backend ID of the user session.

The Wait events profile page with the following [web application](#) interface elements will be displayed:

- The Session tile with the following parameters of the selected user session:
 - Duration: The duration of the user session.
 - Active transaction: The duration of the current transaction in the user session.
 - Active request: The duration of the current query in the user session.
 - Total time spent waiting: The total duration of wait events in the user session.
 - Observations: The number of sampled wait events in the user session.

- Unique events: The number of unique wait events sampled in the user session.
- The Sampling settings tile with the following wait profile configuration parameters:
 - `profile_period` (corresponds to `pg_wait_sampling.profile_period`)
 - `profile_pid` (corresponds to `pg_wait_sampling.profile_pid`)
 - `profile_queries` (corresponds to `pg_wait_sampling.profile_queries`)
 - `sample_cpu` (corresponds to `pg_wait_sampling.sample_cpu`)

For the description of the configuration parameters, refer to the [official Postgres Pro documentation](#). If required, you can [change the values of these parameters](#).

- The graph with statistics on wait events. You can display statistics for the last 5, 10, and 20 wait events using the corresponding buttons at the top of the graph. To scale up or down, use the slider at the bottom of the graph.
- The History tile with the following wait history parameters:
 - Duration: The duration of the wait history.
 - Start and End: The start and end date and time of the wait history.
- The History settings tile with the following wait history configuration parameters:
 - `history_size` (corresponds to `pg_wait_sampling.history_size`)
 - `history_period` (corresponds to `pg_wait_sampling.history_period`)

For the description of the configuration parameters, refer to the [official Postgres Pro documentation](#). If required, you can [change the values of these parameters](#).

- The table of events with the following columns:
 - Time: The date and time of the event.
 - Standby event type: The type of the wait event associated with the event.
- For more information about possible values, refer to the [official Postgres Pro documentation](#).
- Standby event: The wait event associated with the event.
 - Query id: The unique ID of the query associated with the event.
 - Query: The text of the query associated with the event.

To view the query text separately, click  next to it.

Viewing the Current Backend Query Locks

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4.  Click  next to the user session.

Viewing the Current Backend Query Plan

You can view the current backend query plan if the `pg_query_state` module is installed.

To view the current backend query plan:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.

4. Click  next to the user session.

Canceling the Current Backend Query

Canceling a current backend query does not terminate a user session.

To cancel the current backend query:

1. In navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Click  next to the user session.

5. Click Execute.

Terminating a User Session

When the user session is terminated, the current backend query is canceled automatically.

To terminate a user session:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Click  next to the user session.
5. Click Execute.

5.18.2. Vacuum Processes

You can view information about vacuum (*VACUUM*) and autovacuum (*autovacuum*) processes. The information is based on the *pg_stat_progress_vacuum* view.

This section explains how to manage vacuum processes. It includes the following instructions:

- [Viewing Vacuum Processes](#)
- [Canceling a Vacuum Process](#)
- [Terminating the User Session for a Vacuum Process](#)

Viewing Vacuum Processes

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Vacuum, and then in the top-right corner of the page, select the database.

The table of vacuum processes with the following columns will be displayed:

- PID: The ID of the backend.
- State: The state of the backend.

For more information about possible values, refer to the official Postgres Pro documentation on the *pg_stat_activity* module (see the *state* column).

- Wait event: The name and type of the wait event for the backend.

For more information, refer to the official Postgres Pro documentation on the *pg_stat_activity* module (see the *wait_event_type* and *wait_event* columns).

- Phase: The vacuum phase.

For more information about possible values, refer to the [official Postgres Pro documentation](#).

- Database: The name of the database being vacuumed.
- User: The name of the DBMS user on behalf of which vacuum is being performed.
- Table: The name or ID of the table being vacuumed.
- Query: The text of the current or last backend query.
- Query duration: The duration of the current or last backend query.
- Heap size: The size of the table for which vacuum is being performed.
- Total size: The total size of the table, including indexes, for which vacuum is being performed.
- Scanned, %: The percentage of data scanned in the table being vacuumed.

This column includes additional information:

Size: The size of scanned table data.

- Vacuumed, %: The percentage of vacuumed table data.

This column includes additional information:

Size: The size of vacuumed table data.

- Index vacuum: The number of vacuumed table indexes.
- Memory usage, %: Usage of the memory that stores pointers to expired versions of table rows.

For more information, refer to the official Postgres Pro documentation on the [autovacuum_work_mem](#) and [maintenance_work_mem](#) parameters.

- Actions.

For more information about available actions, refer to other instructions in this section.

Canceling a Vacuum Process

Canceling a vacuum process does not terminate a user session.

To cancel a vacuum process:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Vacuum, and then in the top-right corner of the page, select the database.
5. Click  next to the vacuum process.
6. Click Execute.

Terminating the User Session for a Vacuum Process

When the user session is terminated, the vacuum process is canceled automatically.

To terminate the user session for a vacuum process:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Vacuum, and then in the top-right corner of the page, select the database.

5. Click  next to the vacuum process.
6. Click Execute.

5.18.3. Statistics Collection Processes

You can view information about statistics collection processes for the planner ([ANALYZE](#)). The information is based on the `pg_stat_progress_analyze` view.

This section explains how to manage statistics collection processes. It includes the following instructions:

- [Viewing Statistics Collection Processes](#)
- [Canceling a Statistics Collection Process](#)
- [Terminating the User Session for a Statistics Collection Process](#)

Viewing Statistics Collection Processes

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Analyze, and then in the top-right corner of the page, select the database.

The table of statistics collection processes with the following columns will be displayed:

- PID: The ID of the backend.
- State: The state of the backend.

For more information about possible values, refer to the official Postgres Pro documentation on the `pg_stat_activity` module (see the `state` column).

- Wait event: The name and type of the wait event for the backend.

For more information, refer to the official Postgres Pro documentation on the `pg_stat_activity` module (see the `wait_event_type` and `wait_event` columns).

- Phase: The statistics collection phase.

For more information about possible values, refer to the [official Postgres Pro documentation](#).

- Database: The name of the database where the statistics collection is performed.
- User: The name of the DBMS user on behalf of which the statistics are being collected.
- Table: The name or ID of the table for which the statistics are being collected.
- Query: The text of the current or last backend query.
- Query duration: The duration of the current or last backend query.
- Current child table: The name of the child table for which the statistics are being collected.
- Child tables done: The number of child tables for which the statistics were collected.
- Child tables total: The total number of the child tables for which statistics must be collected.
- Scanned, %: The percentage of data scanned in the tables for which statistics are being collected.
- Sample size: The size of the sample used for statistics collection.
- Stat total: The total number of extended statistics objects.
- Stat computed: The number of computed extended statistics objects.
- Actions.

For more information about available actions, refer to other instructions in this section.

Canceling a Statistics Collection Process

Canceling a statistics collection process does not terminate a user session.

To cancel a statistics collection process:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Analyze, and then in the top-right corner of the page, select the database.
5. Click  next to the statistics collection process.
6. Click Execute.

Terminating the User Session for a Statistics Collection Process

When the user session is terminated, the statistics collection process is canceled automatically.

To terminate the user session for a statistics collection process:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Analyze, and then in the top-right corner of the page, select the database.
5. Click  next to the statistics collection process.
6. Click Execute.

5.18.4. Clustering Processes

You can view information about clustering processes — reindexing tables (*CLUSTER*) and about full vacuum processes (*VACUUM FULL*). The information is based on the *pg_stat_progress_cluster* view.

This section explains how to manage clustering processes. It includes the following instructions:

- [Viewing Clustering Processes](#)
- [Canceling a Clustering Process](#)
- [Terminating the User Session for a Clustering Process](#)

Viewing Clustering Processes

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Cluster, and then in the top-right corner of the page, select the database.

The table of clustering processes with the following columns will be displayed:

- PID: The ID of the backend.
- State: The state of the backend.

For more information about possible values, refer to the official Postgres Pro documentation on the *pg_stat_activity* module (see the *state* column).

- Wait event: The name and type of the wait event for the backend.

For more information, refer to the official Postgres Pro documentation on the *pg_stat_activity* module (see the *wait_event_type* and *wait_event* columns).

- Phase: The clustering phase.

For more information about possible values, refer to the [official Postgres Pro documentation](#).

- Database: The name of the database where clustering is being performed.
- User: The name of the DBMS user on behalf of which clustering is being performed.
- Table: The name or ID of the table for which clustering is being performed.
- Query: The text of the current or last backend query.
- Query duration: The duration of the current or last backend query.
- Blocked totals: The number of user sessions blocked by the clustering process.
- Heap scanned, %: The percentage of data scanned in the table for which clustering is being performed.
- Heap total: The total size of the table for which clustering is being performed.
- Scanned: The size of data scanned in the table for which clustering is being performed.
- Tuples scanned: The number of rows scanned in the table for which clustering is being performed.
- Tuples written: The number of rows written to the table for which clustering is being performed.
- Actions.

For more information about available actions, refer to other instructions in this section.

Canceling a Clustering Process

Canceling a clustering process does not terminate a user session.

To cancel a clustering process:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Cluster, and then in the top-right corner of the page, select the database.
5.  Click next to the clustering process.
6. Click Execute.

Terminating the User Session for a Clustering Process

When the user session is terminated, the clustering process is canceled automatically.

To terminate the user session for a clustering process:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Cluster, and then in the top-right corner of the page, select the database.
5.  Click next to the clustering process.
6. Click Execute.

5.18.5. Reindexing or Index Creation Processes

You can view information about reindexing ([REINDEX](#)) or index creation ([CREATE INDEX](#)) processes. The information is based on the [pg_stat_progress_create_index](#) view.

This section explains how to manage reindexing or index creation processes. It includes the following instructions:

- [Viewing Reindexing or Index Creation Processes](#)
- [Canceling a Reindexing or Index Creation Process](#)
- [Terminating the User Session for a Reindexing or Index Creation Process](#)

Viewing Reindexing or Index Creation Processes

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Reindex, and then in the top-right corner of the page, select the database.

The table of reindexing or index creation processes with the following columns will be displayed:

- PID: The ID of the backend.
- State: The state of the backend.

For more information about possible values, refer to the official Postgres Pro documentation on the [pg_stat_activity](#) module (see the `state` column).

- Wait event: The name and type of the wait event for the backend.

For more information, refer to the official Postgres Pro documentation on the [pg_stat_activity](#) module (see the `wait_event_type` and `wait_event` columns).

- Phase: The reindexing or index creation phase.

For more information about possible values, refer to the [official Postgres Pro documentation](#).

- Database: The name of the database where reindexing or index creation is being performed.
- User: The name of the DBMS user under which reindexation or index creation is being performed.
- Table: The name or ID of the table for which reindexing or index creation is being performed.
- Index: The name or ID of the index being reindexed or created.
- Query: The text of the current or last backend query.
- Query duration: The duration of the current or last backend query.
- Done, %: The percentage of table data processed in the current phase of reindexing or index creation.

This column includes additional information:

- Total size: The total size of the table data to be processed.
- Done size: The size of the processed table data.
- Tuples done, %: The percentage of table rows processed in the current phase of reindexing or index creation.

This column includes additional information:

- Total size: The total number of table rows to be processed.
- Done size: The number of processed table rows.
- Lockers: The processes locked when performing reindexing or index creation.

This column includes additional information:

- Done: The number of locked processes for which the waiting is finished.

- Total: The total number of locked processes.
- PID: The ID of the currently locked process.
- Partitions done, %: The percentage of partitioned tables processed when performing reindexing or index creation.

This column includes additional information:

- Total: The total number of partitioned tables to be processed.
- Done: The number of processed partitioned tables.
- Actions.

For more information about available actions, refer to other instructions in this section.

Canceling a Reindexing or Index Creation Process

Canceling a reindexing or index creation process does not terminate a user session.

To cancel a reindexing or index creation process:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Reindex, and then in the top-right corner of the page, select the database.
5. Click  next to the reindexing or index creation process.
6. Click Execute.

Terminating the User Session for a Reindexing or Index Creation Process

When the user session is terminated, the reindexing or index creation process is canceled automatically.

To terminate the user session for a reindexing or index creation process:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Reindex, and then in the top-right corner of the page, select the database.
5. Click  next to the reindexing or index creation process.
6. Click Execute.

5.18.6. Base Backup Processes

You can view information about base backup processes (usually these are standard backup processes). The information is based on the [pg_stat_progress_basebackup](#) view.

This section explains how to manage base backup processes. It includes the following instructions:

- [Viewing Base Backup Processes](#)
- [Canceling a Base Backup Process](#)
- [Terminating the User Session for a Base Backup Process](#)

Viewing Base Backup Processes

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.

3. In the navigation panel, go to Activity.
 4. Select Base backup, and then in the top-right corner of the page, select the database.
- The table of base backup processes with the following columns will be displayed:

- PID: The ID of the backend.
- State: The state of the backend.

For more information about possible values, refer to the official Postgres Pro documentation on the [pg_stat_activity](#) module (see the `state` column).

- Wait event: The name and type of the wait event for the backend.

For more information, refer to the official Postgres Pro documentation on the [pg_stat_activity](#) module (see the `wait_event_type` and `wait_event` columns).

- Phase: The base backup phase.

For more information about possible values, refer to the [official Postgres Pro documentation](#).

- Client address: The network address of the client that initiated the base backup.
- Backend duration: Backend duration (base backup duration).
- Started time: The backend start date and time (base backup start date and time).
- Sent, %: The percentage of data sent to the client when performing the base backup.

This column includes additional information:

- Total size: The total size of data to be sent to the client.
- Sent size: The size of data sent to the client.
- Tablespace total: The total number of tablespaces to be processed when performing a base backup.
- Tablespaces streamed: The number of tablespaces processed when performing base backup.
- Actions.

For more information about available actions, refer to other instructions in this section.

Canceling a Base Backup Process

Canceling a backup process does not terminate a user session.

To cancel a base backup process:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Base backup, and then in the top-right corner of the page, select the database.
5. Click  next to the base backup process.
6. Click Execute.

Terminating the User Session for a Base Backup Process

When the user session is terminated, the base backup process is canceled automatically.

To terminate the user session for a base backup process:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.

3. In the navigation panel, go to Activity.
4. Select Base backup, and then in the top-right corner of the page, select the database.
5. Click  next to the base backup process.
6. Click Execute.

5.18.7. Copying Processes

You can view information about copying processes (*COPY*). The information is based on the *pg_stat_progress_copy* view.

This section explains how to manage copying processes. It includes the following instructions:

- [Viewing Copying Processes](#)
- [Canceling a Copying Process](#)
- [Terminating the User Session for a Copying Process](#)

Viewing Copying Processes

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Copy, and then in the top-right corner of the page, select the database.

The table of copying processes with the following columns will be displayed:

- PID: The ID of the backend.
- State: The state of the backend.

For more information about possible values, refer to the official Postgres Pro documentation on the *pg_stat_activity* module (see the *state* column).

- Wait event: The name and type of the wait event for the backend.

For more information, refer to the official Postgres Pro documentation on the *pg_stat_activity* module (see the *wait_event_type* and *wait_event* columns).

- Database: The name of the database where copying is being performed.
- User: The name of the DBMS user under which copying is being performed.
- Table: The name or ID of the table being copied.
- Query: The text of the current or last backend query.
- Query duration: The duration of the current or last backend query.
- Table size: The size of the table being copied.
- Table tuples total: Rows of the table being copied.

This column includes additional information:

- Tuples processed: The number of processed table rows.
- Tuples excluded: The number of excluded table rows.
- Source file size: The size of the file being copied.
- Done, %: The completion percentage of copying.

This column includes additional information:

Processed: The size of the processed data.

- Command: The text of the copying command.
- Command type: The type of the copying command.
- Actions.

For more information about available actions, refer to other instructions in this section.

Canceling a Copying Process

Canceling a copying process does not terminate a user session.

To cancel a copying process:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Copy, and then in the top-right corner of the page, select the database.
5.  Click  next to the copying process.
6. Click Execute.

Terminating the User Session for a Copying Process

When the user session is terminated, the copying process is canceled automatically.

To terminate the user session for a copying process:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Activity.
4. Select Copy, and then in the top-right corner of the page, select the database.
5.  Click  next to the copying process.
6. Click Execute.

5.19. SQL Statistics

PPEM allows viewing execution statistics of SQL statements. This information is provided by the [pg_stat_statements](#) and [pgpro_stats](#) extensions. For SQL statistics collection to work correctly, one of these extensions must be installed and configured in the DBMS instance.

To view SQL statistics:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to SQL statistics.

The table with the following columns will be displayed:

- Query ID: The non-unique hash code of the statement. It can be the same as that of other statements executed in other databases or on behalf of other users.

This column includes additional information:

- Plan ID: The non-unique hash code of the plan. It can be the same as that of other plans for statements executed in other databases or on behalf of other users.

This column is displayed only for the Postgres Pro Enterprise edition.

- Top level: The statement execution level.

Possible values:

- true: The statement is executed at the top level.
- false: The statement is nested in a procedure or function.

To display this information, set the `pg_stat_statements.track` or `pgpro_stats.track` configuration parameter to `top`.

- Database: The instance database where the statement was executed.
- User: The name of the user who executed the statement.
- Calls: The total number of times the statement was executed.

This column includes additional information:

Rows: The total number of rows retrieved or affected by the statement.

- Total execution time, ms: The total time spent executing the statement, in milliseconds.

This column includes additional information:

- Max: The maximum time spent executing the statement.
- Min: The minimum time spent executing the statement.
- Mean: The mean time spent executing the statement.
- Stddev: The standard deviation of time spent executing the statement.
- Total planning time, ms: The total time spent planning the statement, in milliseconds.

To display this information, set the `pg_stat_statements.track_planning` or `pgpro_stats.track_planning` configuration parameter to `on`. Otherwise, 0 is displayed.

This column includes additional information:

- Max: The maximum time spent planning the statement.
- Min: The minimum time spent planning the statement.
- Mean: The mean time spent planning the statement.
- Stddev: The standard deviation of time spent planning the statement.
- Blocks time, ms: The total time the statement spent reading and writing data file blocks, in milliseconds.

To display this information, enable the `track_io_timing` configuration parameter. Otherwise, 0 is displayed.

This column includes additional information:

- Write: The time spent for writing blocks.
- Read: The time spent for reading blocks.
- Temp blocks, pc.: The total number of blocks affected by the statement when working with temporary files.

This column includes additional information:

- Written: The number of written blocks.
- Read: The number of read blocks.

- WAL bytes, B: The total amount of WAL bytes generated during the statement execution.

This column includes additional information:

- Records, pc.: The total number of WAL records generated during the statement execution.
- FPI, pc.: The total number of WAL full page images generated during the statement execution.
- Shared blocks • Hits, pc.: The total number of shared block cache hits by the statement.

This column includes additional information:

- Read: The total number of shared blocks read by the statement.
- Dirtied: The total number of shared blocks dirtied by the statement.
- Written: The total number of shared blocks written by the statement.
- Local blocks • Hits, pc.: The total number of local block cache hits by the statement.

This column includes additional information:

- Read: The total number of local blocks read by the statement.
- Dirtied: The total number of local blocks dirtied by the statement.
- Written: The total number of local blocks written by the statement.

To view information about the specified statement, click  next to it.

The displayed statistics are requested via the [agent](#) working with the instance. For this reason, the speed at which statistics are retrieved depends on two factors:

- network connectivity between the [manager](#) and agent that works with the instance
- the volume of transferred data that can also indirectly affect the instance performance

Given the cumulative nature of statistics, the resulting statistics snapshot can differ over time from the actual instance statistics.

5.20. Repository Maintenance

Certain tables of the [repository](#) database can overflow, for example, tables containing metrics and logs. To avoid table overflow, PPEM uses housekeeping rules. A *housekeeping rule* cleans the table by deleting the data stored longer than the specified amount of time. It can also be a set of one or multiple conditions containing a table field, logical operator, and value or function.

If a housekeeping rule is a set of multiple conditions, the `AND` logical connective is used.

For example, a housekeeping rule can delete the table data stored longer than 12 hours and include the condition containing the `oid` table field, the `>` logical operator, and the `10` value. With such a rule, the table data is deleted after 12 hours and if the value of the `oid` table field exceeds `10`.

You can mark housekeeping rules as auto-run. In this case, they clean the table at the specified time interval.

This section explains how to manage housekeeping rules. It contains the following instructions:

- [Creating a Housekeeping Rule](#)
- [Viewing Housekeeping Rules](#)
- [Viewing Detailed Housekeeping Rule Information](#)
- [Running a Housekeeping Rule](#)
- [Editing a Housekeeping Rule](#)
- [Deleting a Housekeeping Rule](#)

Important

The housekeeping rules functionality is in the beta phase. It will be updated and expanded later.

Creating a Housekeeping Rule

1. In the navigation panel, go to Settings → Repository.
2. In the top-right corner of the page, click Create rule.
3. Enter parameters of the new housekeeping rule (parameters marked with an asterisk are required):

- Instance: The instance where the [repository](#) database is located.

The value is entered automatically.

- Name.
- Schema: The schema where the table is located.
- Table: The table that will be cleaned.
- Timestamp column: The table column that contains the timestamp for determining the data age.
- Maximum storage time: The maximum time the table data can be stored before deletion.

Specify a value in one of the following formats:

- `<number_of_hours>h<number_of_minutes>m`, for example, `24h00m`
- `<number_of_hours>h`, for example, `24h`
- `<number_of_minutes>m`, for example, `24m`
- Auto-run schedule: Specifies whether the housekeeping rule is auto-run or not.

If this toggle is turned on, specify the time interval for cleaning the table in one of the following ways:

- In Crontab interval, enter the string in the crontab format.
- Click Configure schedule, and then specify the following parameters:
 - Interval: The unit of measurement of the time interval.

Possible values:

- Minutes
- Hours
- Days
- Repeat every: The time interval for cleaning the table by minutes or hours.

This parameter is available only if you select Minutes or Hours from Interval.
- Execution days: The days when the table will be cleaned.
- Cron total line: The string in the crontab format that specifies the time interval for cleaning the table.

The value is entered automatically.

After configuring the time interval, click Save.

- Description.

- Additional conditions: The housekeeping rule conditions.

To add a housekeeping rule condition, click Add condition, and then specify the following parameters:

- Field: The table field for the housekeeping rule condition.
- Condition: The logical operator of the housekeeping rule condition.

Possible values:

- =
 - $\neq$
 - IN
 - NOT IN
 - LIKE
 - NOT LIKE
 - >
 - <
 - $\geq$
 - $\leq$
 - REGEXP
 - NOT REGEXP
- Value type: The type of the value of the housekeeping rule condition.

Possible values:

- Literal
- Function

Currently, the only function you can use is `now`. This function takes the current time-stamp.

- Value: The value of the housekeeping rule condition.

For example, if you select `oid` from Field, `=` from Condition, Literal from Value type, and enter `10` in Value, the data is deleted if the value of the `oid` table field is equal to `10`.

After configuring the housekeeping rule condition, click Save.

You can add multiple housekeeping rule conditions by clicking Add condition. In this case, the `AND` logical connective is used.

- Action after cleaning: The action that will be performed after cleaning the table.

To add an action, click Add action, and then select the action. Currently, the only supported action is table vacuuming.

You can also enable the following checkboxes:

- Analyze: Update the table statistics.
- Full: Run the `VACUUM FULL` command.

In this case, more data is vacuumed, but the operation takes longer to complete and locks the table.

- Freeze: Perform an aggressive tuple freeze.

This is equivalent to running the `VACUUM` command with the `vacuum_freeze_min_age` parameter set to 0.

After adding the action, click Save.

You can add multiple actions by clicking Add action.

4. Click Create.

Viewing Housekeeping Rules

In the navigation panel, go to Settings → Repository.

The table of housekeeping rules with the following columns will be displayed:

- ID.
- Name.
- Maximum storage time: The maximum time the table data can be stored before deletion.
- Auto-run: Specifies whether the housekeeping rule is auto-run or not.

Possible values:

- On schedule: The housekeeping rule is auto-run.
- Disabled: The housekeeping rule is not auto-run.
- Schedule: The time interval in the crontab format for cleaning the table.
- Status.

Possible values:

- Completed: The table is cleaned successfully.
- Executing: The table is being cleaned.
- Error: An error occurred when cleaning the table.
- Canceled: The table cleaning was canceled.
- Last run: The date and time the table was last cleaned.
- Schema: The schema where the table is located.
- Table: The table that is being cleaned.
- Deleted: The amount of deleted table data.

You can view the following table parameters by hovering over the displayed value:

- Deleted.
- Current size.
- Previous size.
- Decrease: The table size decrease percentage.
- Actions.

For more information about available actions, refer to other instructions in this section.

Viewing Detailed Housekeeping Rule Information

1. In the navigation panel, go to Settings → Repository.
- 2.

Click  next to the housekeeping rule.

The window with detailed information about the housekeeping rule will open. On Last run, the following parameters related to the last table cleaning are displayed:

- Status.
- Start: The date and time the table cleaning started.
- End: The date and time the table cleaning ended.
- Duration: The duration of the table cleaning.
- Cleaning process ID.
- Table size before vacuuming.
- Table size after vacuuming.
- Deleted: The amount of deleted table data.

On Parameters, the following housekeeping rule parameters are displayed:

- Instance: The instance where the [repository](#) database is located.
- ID.
- Type: Specifies whether this is the system or user housekeeping rule.

Possible values:

- System
- User
- Schema: The schema where the table is located.
- Table: The table that is being cleaned.
- Timestamp column: The table column that contains the timestamp for determining the data age.
- Maximum storage time: The maximum time the table data can be stored before deletion.
- Auto-run: Specifies whether the housekeeping rule is auto-run or not.

Possible values:

- On schedule: The housekeeping rule is auto-run.
- Disabled: The housekeeping rule is not auto-run.
- Schedule in the crontab format: The time interval in the crontab format for cleaning the table.
- Description.
- Additional conditions: The table of housekeeping rule conditions.

This table includes the following columns:

- Field: The table field for the housekeeping rule condition.
- Condition: The logical operator of the housekeeping rule condition.

Possible values:

- =

- $\neq$
 - IN
 - NOT IN
 - LIKE
 - NOT LIKE
 - >
 - <
 - $\geq$
 - $\leq$
 - REGEXP
 - NOT REGEXP
- Value: The value of the housekeeping rule condition.

Running a Housekeeping Rule

You can manually run a housekeeping rule to clean the table immediately if this rule is auto-run.

To run a housekeeping rule:

1. In the navigation panel, go to Settings → Repository.
2.  Click  next to the housekeeping rule.
3. Click Run now.

Editing a Housekeeping Rule

1. In the navigation panel, go to Settings → Repository.
2.  Click  → Edit next to the housekeeping rule.
3. Edit housekeeping rule parameters.
4. Click Save.

Deleting a Housekeeping Rule

Important

- System housekeeping rules cannot be deleted.
- Deleted housekeeping rules cannot be restored.

1. In the navigation panel, go to Settings → Repository.
2.  Click  → Delete next to the housekeeping rule.
3. Click Delete.

5.21. Instance Parameters

PPEM allows [viewing](#) and [editing](#) parameters of the DBMS instance.

The `ALTER SYSTEM` command is used to modify instance parameters. It modifies the `postgresql.auto.conf` file.

If an instance parameter was initially specified in the main configuration file, such as `postgresql.conf`, but later changed via PPEM and written to `postgresql.auto.conf`, the [web application](#) will display this parameter in both files. This is normal because values from `postgresql.auto.conf` take precedence over other configuration files.

PPEM uses the `pg_reload_conf()` function to apply instance parameters without restarting the instance.

Viewing Instance Parameters

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Parameters.

The instance parameters will be displayed, grouped into tables by different semantic categories. The tables include the following columns:

- **Parameter:** The name and description of the instance parameter. If the instance parameter is specified in a configuration file, the path to this file and the row number will be displayed next to it.

This column also displays the instance parameter application status and errors, if they occurred.

To open a full description of the instance parameter in the official Postgres Pro documentation, click its name.

- **Value.**
- **Default value.**
- **Source:** The source of the instance parameter value.

Possible values:

- **default:** This is the default instance parameter value.
- **configuration file:** The instance parameter value is specified in the configuration file.
- **override:** The instance parameter value is overridden at the instance initialization stage.

Editing Instance Parameters

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Parameters.
4. In Value, perform one of the following actions:
 - Enable or disable the instance parameter by turning on or off the corresponding toggle.
 - Change the instance parameter value by specifying the new value in the corresponding field.

When an instance parameter is edited, an icon for restoring the previous value  is displayed next to it.

5. (Optional) To reset all edited but not yet saved instance parameters, in the bottom-right corner of the page, click Reset.
6. In the bottom-right corner of the page, click Save.

The action selection window will open if instance parameters, which require an instance restart, were edited.

7. Select one of the following actions:

- Apply changes and restart the instance: The instance parameter changes will be applied, the instance will restart automatically.
 - Apply changes, but restart instance later manually: The instance parameter changes will be applied, but you will need to restart the instance manually.
8. Click Apply.
 9. Confirm the operation.

Important

Restarting the DBMS instance terminates all its active sessions. The restart process may take a significant time due to the checkpoint execution. The restart duration depends on the configuration and the instance load.

It is recommended to pay special attention to restarts, especially in production environments with strict DBMS downtime requirements.

5.22. Extensions

This section explains how to manage instance extensions. It includes the following instructions:

- [Installing Extensions](#)
- [Viewing Extensions](#)
- [Editing an Extension](#)
- [Deleting an Extension](#)

Installing Extensions

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Extensions.
4. In the top-right corner of the page, click Install extension.
5. Enter parameters of the new extension (parameters marked with an asterisk are required):
 - Database: The database for which the extension will be installed.
 - Extension: The extension that will be installed.
 - Version: The version of the extension.
 - Schema: The schema into which the extension objects will be installed.

If no schema is specified, the current schema is used.

- Install extensions that the selected one depends on: Specifies whether to install all extensions that the selected extension depends on if they are not already installed.
6. Click Install.

Viewing Extensions

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Extensions.

The table of extensions with the following columns will be displayed:

- Name.
- Version.
- Schema: The schema into which the extension objects are installed.
- Database: The database for which the extension is installed.
- Owner.
- Description.

Editing an Extension

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Extensions.
4. Click  next to the extension.
5. Edit extension parameters.
6. Click Save.

Deleting an Extension

Important

Deleted extensions cannot be restored.

To delete an extension:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Extensions.
4. Click  next to the extension.
5. (Optional) To delete all related extensions, enable Delete related extensions.
6. Click Delete.

5.23. Profiler

Profiler is an interface to the [pgpro_pwr](#) module. With profiler, you can build detailed reports on database performance. Reports are based on samples of database data and cover the specified periods of time.

To build a report, take at least two samples. The samples are placed in the directory of the profiler server. Before taking samples, prepare profiler servers and create them in the [web application](#).

You can create schedules that allow taking samples automatically at the specified time interval. Such schedules also allow taking a sample for the specified date and time.

5.23.1. Profiler Servers

This section explains how to manage profiler servers. It includes the following instructions:

- [Creating a Profiler Server](#)
- [Viewing Profiler Servers](#)

- [Editing a Profiler Server](#)
- [Deleting a Profiler Server](#)

Creating a Profiler Server

Before performing this instruction:

- [Create an instance.](#)
- [Create a database.](#)
- Prepare a profiler server.

To create a profiler server:

1. Go to profiler servers in one of the following ways:
 - Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler → Servers.
 - b. From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler → Servers.
2. In the top-right corner of the page, click Create server.
3. Enter parameters of the new profiler server (parameters marked with an asterisk are required):
 - Database: The database for which the profiler server will take samples.
 - Server name.
 - Description.
 - Connection string: The connection string for the profiler server.
 - Enabled: Specifies whether the profiler server is enabled.

Possible values:

- yes
- no
- Retention period (days): The number of days during which samples can be stored on the profiler server.

Samples stored longer than the specified period are deleted.

4. Click Save.

Viewing Profiler Servers

1. Go to profiler servers in one of the following ways:
 - Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler → Servers.
 - b. From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.

- c. In the navigation panel, go to Profiler → Servers.
2. From Database, select the database.
3. Click Select.

The table of profiler servers with the following columns will be displayed:

- Name.
- Description.
- Connection string: The connection string for the profiler server.
- Enabled: Specifies whether the profiler server is enabled.

Possible values:

- yes
- no
- Retention period: The number of days during which samples are stored.

Samples stored longer than the specified period are deleted.

- Actions.

For more information about available actions, refer to other instructions in this section.

Editing a Profiler Server

1. Go to profiler servers in one of the following ways:
 - Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler → Servers.
 - b. From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler → Servers.
2. From Database, select the database.
3. Click Select.
4. Click  next to the profiler server.
5. Edit profiler server parameters.
6. Click Save.

Deleting a Profiler Server

Important

Deleted profiler servers cannot be restored.

Deleting a profiler server also removes all associated samples and disables sampling schedules that place samples to this server directory.

To delete a profiler server:

1. Go to profiler servers in one of the following ways:
 - Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler → Servers.
 - b. From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler → Servers.
2. From Database, select the database.
3. Click Select.
4. Click  next to the profiler server.
5. Click Delete.

5.23.2. Samples

This section explains how to [take](#) and [view](#) samples.

Taking a sample

Before performing this instruction:

- [Create an instance](#).
- [Create a database](#).
- [Create a profiler server](#).

To take a sample:

1. Go to samples in one of the following ways:
 - Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler → Samples.
 - b. From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler → Samples.
2. In the top-right corner of the page, click Take snapshot.
3. Enter parameters of the new sample (parameters marked with an asterisk are required):
 - Database: The database for which the sample will be taken.
 - Server: The profiler server whose directory will be used to store the sample.
 - skip_sizes: Specifies whether to skip the collection of relation sizes when taking the sample.

Possible values:

- `true`: The collection of relation sizes will be skipped when taking the sample.
- `false`: The collection of relation sizes will not be skipped when taking the sample.

4. Click Save.

Viewing Samples

1. Go to samples in one of the following ways:
 - Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler → Samples.
 - b. Select the instance.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler → Samples.
2. From Database, select the database.
3. From Server, select the profiler server.
4. Click Select.

The table of samples with the following columns will be displayed:

- **ID**: The unique identifier (ordinal number) of the sample.
- **Time**: The date and time when the sample was created.
- **Sizes were collected**: Specifies whether relation sizes were collected when taking the sample.

Possible values:

- `yes`
- `no`
- **Database stat reset**: Specifies whether database statistics were reset when taking the sample.

Possible values:

- `yes`
- `no`
- **Bgwriter stat reset**: Specifies whether `bgwriter` statistics were reset.

Possible values:

- `yes`
- `no`
- **Archiver stat reset**: Specifies whether archiver statistics were reset.

Possible values:

- `yes`
- `no`

5.23.3. Viewing Profiler Graphs

Before performing this instruction:

- [Create an instance.](#)
- [Create a database.](#)
- [Create a profiler server.](#)
- [Take at least two samples.](#)

To view profiler graphs:

1. Go to profiler graphs in one of the following ways:
 - Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler.
 - b. From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler.
2. From Database, select the database.
3. From Server, select the profiler server.
4. (Optional) In Period, specify the time period.

If you do not specify a value, the profiler graphs are displayed for the entire time range.

5. Click Select.

The following profiler graphs will be displayed:

- PostgreSQL Instance: tuples: Operations with rows per second in the database. Available metrics:

- Tuples returned: The number of live rows fetched by sequential scans and index entries returned by index scans.

This metric corresponds to the `tup_returned` column of the `pg_stat_database` view.

- Tuples fetched: The number of live rows fetched by index scans.

This metric corresponds to the `tup_fetched` column of the `pg_stat_database` view.

- Tuples inserted: The number of rows inserted by queries.

This metric corresponds to the `tup_inserted` column of the `pg_stat_database` view.

- Tuples updated: The number of rows updated by queries.

This metric corresponds to the `tup_updated` column of the `pg_stat_database` view.

- Tuples deleted: The number of rows deleted by queries.

This metric corresponds to the `tup_deleted` column of the `pg_stat_database` view.

For more information about these metrics, refer to the official Postgres Pro documentation on the [pg_stat_database](#) view.

- PostgreSQL bgwriter buffers: Operations with buffers per second in the database. Available metrics:

- Checkpoints buffers written: The number of buffers written during checkpoints and restart-points.

This metric corresponds to the `buffers_written` column of the `pg_stat_checkpoint` view.

- Background buffers written: The number of buffers written by the background writer.

This metric corresponds to the `buffers_clean` column of the `pg_stat_bgwriter` view.

- Backend buffers written: The number of buffers written directly by the backend.

In Postgres Pro 16, this metric corresponds to the `buffers_backend` column of the `pg_stat_bgwriter` view. In Postgres Pro 17, this metric is collected by the `pg_stat_io` view.

- Number of buffers allocated: The number of allocated buffers.

This metric corresponds to the `buffers_alloc` column of the `pg_stat_bgwriter` view.

For more information about these metrics, refer to the official Postgres Pro documentation on the [pg_stat_bgwriter](#), [pg_stat_io](#), and [pg_stat_checkpoint](#) views.

- PostgreSQL bgwriter write/sync: Synchronization and writing operations with buffers per second in the database. Available metrics:

- Bgwriter interrupts: The number of times the background writer stopped a cleaning scan due to writing too many buffers.

This metric corresponds to the `maxwritten_clean` column of the `pg_stat_bgwriter` view.

- Backend fsync count: The number of times the backend executed its own `fsync` call. Normally, these calls are executed by the background writer, even when the backend performs its own writes.

In Postgres Pro 16 or lower, this metric corresponds to the `buffers_backend_fsync` column of the `pg_stat_bgwriter` view. In Postgres Pro 17, this metric is collected by the `pg_stat_io` view.

For more information about these metrics, refer to the official Postgres Pro documentation on [pg_stat_bgwriter](#) and [pg_stat_io](#).

- PostgreSQL checkpoints count: Operations with checkpoints per second in the database. Available metrics:

- Scheduled checkpoints: The number of scheduled checkpoints that were completed due to timeout. Scheduled checkpoints can be skipped if the server has been idle since the last checkpoint. Both completed and skipped scheduled checkpoints are counted.

This metric corresponds to the `num_timed` column of the `pg_stat_checkpoint` view.

- Requested checkpoints: The number of requested checkpoints that were completed.

This metric corresponds to the `num_requested` column of the `pg_stat_checkpoint` view.

For more information about these metrics, refer to the official Postgres Pro documentation on the [pg_stat_checkpoint](#) view.

- PostgreSQL checkpoints write/sync: The time per second spent writing and synchronizing files during checkpoints in the database. Available metrics:

- Checkpoint write time (s): The time in seconds spent writing files to disk while completing checkpoints and restartpoints.

This metric corresponds to the `write_time` column of the `pg_stat_checkpoint` view.

- Checkpoint sync time (s): The time in seconds spent synchronizing files to disk while completing checkpoints and restartpoints.

This metric corresponds to the `sync_time` column of the `pg_stat_checkpoint` view.

For more information about these metrics, refer to the official Postgres Pro documentation on the [pg_stat_checkpoint](#) view.

- PostgreSQL Instance: events: Operations with events per second in the database. Available metrics:
 - Conflicts: The number of queries canceled due to conflicts with recovery. Conflicts can occur only on standby servers.

This metric corresponds to the `conflicts` column of the `pg_stat_database` view.

For more information about conflicts, refer to the official Postgres Pro documentation on the [pg_stat_database_conflicts](#) view.

- Deadlocks: The number of deadlocks.

This metric corresponds to the `deadlocks` column of the `pg_stat_database` view.

- Rollbacks: The number of rolled back transactions.

This metric corresponds to the `xact_rollback` column of the `pg_stat_database` view.

- Commits: The number of committed transactions.

This metric corresponds to the `xact_commit` column of the `pg_stat_database` view.

For more information about these metrics, refer to the official Postgres Pro documentation on the [pg_stat_database](#) view.

- PostgreSQL: cache hit ratio: This graph displays the Cache hit ratio metric that provides the percentage of the data received from the buffer cache per second in the database.

This metric is based on the `blks_hit` and `blks_read` columns of the `pg_stat_database` view and is calculated as follows:

$$\text{blks_hit} / (\text{blks_hit} + \text{blks_read})$$

For more information about the `blks_hit` and `blks_read` columns, refer to the official Postgres Pro documentation on the [pg_stat_database](#) view.

- PostgreSQL temp: bytes written: This graph displays the Bytes written metric that provides the amount of data written to temporary files by queries per second in the database. All temporary files are counted, regardless of why they were created and of the [log_temp_files](#) parameter value.

This metric corresponds to the `temp_bytes` column of the `pg_stat_database` view.

For more information about this metric, refer to the official Postgres Pro documentation on the [pg_stat_database](#) view.

- PostgreSQL temp: files created: This graph displays the Number of files metric that provides the number of temporary files created by queries per second in the database. All temporary files are counted, regardless of why they were created (sorting, hashing) and of the [log_temp_files](#) parameter value.

This metric corresponds to the `temp_files` column of the `pg_stat_database` view.

For more information about this metric, refer to the official Postgres Pro documentation on the [pg_stat_database](#) view.

- PostgreSQL archive command: Archiving operations with WAL files per second in the database. Available metrics:

- WAL segments archived: The number of successfully archived WAL files.

This metric corresponds to the `archived_count` column of the `pg_stat_archiver` view.

- WAL segments archive failed: The number of failed attempts to archive WAL files.

This metric corresponds to the `failed_count` column of the `pg_stat_archiver` view.

For more information about these metrics, refer to the official Postgres Pro documentation on the [pg_stat_archiver](#) view.

- PostgreSQL WAL write speed: This graph displays the WAL generated metric that provides the amount of WAL in bytes generated per second in the database.

This metric corresponds to the `wal_bytes` column of the `pg_stat_wal` view.

For more information about this metric, refer to the official Postgres Pro documentation on the [pg_stat_wal](#) view.

You can perform the following actions using the icons in the top-right corner of the planner graphs:

- To build a report using a graph, click Select the period for the report.

For more information about building a report, refer to [Building a Report](#).

- To reset the selected time period when building a report, click Reset.
- To download the graph in the PNG format, click Save as Image.

5.23.4. Sampling Schedules

This section explains how to manage sampling schedules. It includes the following instructions:

- [Creating a Sampling Schedule](#)
- [Viewing Sampling Schedules](#)
- [Editing a Sampling Schedule](#)
- [Executing a Sampling Schedule](#)
- [Activating and Deactivating a Sampling Schedule](#)
- [Deleting a Sampling Schedule](#)

Creating a Sampling Schedule

Before performing this instruction:

- [Create an instance](#).
- [Create a database](#).
- [Create a profiler server](#).

To create a sampling schedule:

1. Go to sampling schedules in one of the following ways:
 - Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler → Schedule.
 - b. From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler → Schedule.
2. In the top-right corner of the page, click Create task.
3. Enter parameters of the new sampling schedule (parameters marked with an asterisk are required):
 - Name.
 - Set cron-string execution: Allows specifying the time interval for taking samples in the crontab format.

If you turn on this toggle, in Execution, enter the string.

- Task planning: The type of the sampling schedule.

Possible values:

- Time-delayed: One sample will be created at the specified date and time.
- On schedule: Samples will be taken at the specified time interval.

For this value, specify the following parameters:

- Interval: The unit of measurement of the time interval.

Possible values:

- Minutes
- Hours
- Days
- Repeat every: The time interval for taking samples by minutes or hours.

This parameter is available only if you select Minutes or Hours from Interval.

- Execution days: The days when samples will be taken.
- Cron total line: The string in the crontab format that specifies the time interval for taking samples.

The value is entered automatically.

This parameter is available only if you turn off Set cron-string execution.

- Time: The date and/or time when samples will be taken.

This parameter is available only if you select Time-delayed from Task planning or Days from Interval.

- Start and Repeat until: The start and end date and time for taking samples.

These parameters are available only if you turn on Set cron-string execution or select On schedule from Task planning.

- Database: The database for which the sample will be taken.
- Server: The profiler server whose directory will be used to store the sample.
- skip_sizes: Specifies whether to skip the collection of relation sizes when taking the sample.

Possible values:

- true: The collection of relation sizes will be skipped when taking the sample.
- false: The collection of relation sizes will not be skipped when taking the sample.

4. Click Save.

Viewing Sampling Schedules

Go to sampling schedules in one of the following ways:

- Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler → Schedule.
 - b. From Instance, select the instance.

- Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler → Schedule.

The table of sampling schedules with the following columns will be displayed:

- Task: The unique name of the sampling schedule.
- Instance: The instance where the sampling schedule is created.
- Database: The database for which samples are taken.
- Schedule: The string in the crontab format that specifies the time interval for taking samples.
- User: The user that created the sampling schedule.
- Actions.

For more information about available actions, refer to other instructions in this section.

Editing a Sampling Schedule

1. Go to sampling schedules in one of the following ways:
 - Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler → Schedule.
 - b. From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler → Schedule.
2. Click  → Edit next to the sampling schedule.
3. Edit sampling schedule parameters.
4. Click Save.

Executing a Sampling Schedule

You can manually execute a sampling schedule to instantly start taking samples.

To execute a sampling schedule:

1. Go to sampling schedules in one of the following ways:
 - Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler → Schedule.
 - b. From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler → Schedule.

2.  → Execute next to the sampling schedule.

Activating and Deactivating a Sampling Schedule

You can deactivate a sampling schedule to temporarily stop taking samples. Sampling schedules are activated by default.

To deactivate or activate a sampling schedule:

- Go to sampling schedules in one of the following ways:
 - Using the monitoring section:
 - In the navigation panel, go to Monitoring → Profiler → Schedule.
 - From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - In the navigation panel, go to Infrastructure → Instances.
 - Click the name of the instance.
 - In the navigation panel, go to Profiler → Schedule.
2.  → Deactivate or Activate next to the sampling schedule.

Deleting a Sampling Schedule

Important

Deleted sampling schedules cannot be restored.

When you delete a sampling schedule, the associated samples are not deleted.

To delete a sampling schedule:

- Go to sampling schedules in one of the following ways:
 - Using the monitoring section:
 - In the navigation panel, go to Monitoring → Profiler → Schedule.
 - From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - In the navigation panel, go to Infrastructure → Instances.
 - Click the name of the instance.
 - In the navigation panel, go to Profiler → Schedule.
2.  → Delete next to the sampling schedule.
3. Click Delete.

5.23.5. Reports

This section explains how to manage reports. It includes the following instructions:

- [Building a Report](#)
- [Building a Report Using a Profiler Graph](#)

- [Viewing All Built Reports](#)
- [Viewing and Downloading a Report](#)
- [Deleting a Report](#)

It is recommended to view [available profiler graphs](#) first.

Building a Report

Before performing this instruction:

- [Create an instance.](#)
- [Create a database.](#)
- [Create a profiler server.](#)
- [Take at least two samples.](#)

To build a report:

1. Go to building a report in one of the following ways:
 - Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler → Reports generating.
 - b. From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler → Reports generating.
2. Enter parameters of the new report (parameters marked with an asterisk are required):
 - Database: The database for which the report will be built.
 - Server: The profiler server whose directory contains samples.
 - Report: The format of the time period that the report will cover.

Possible values:

- By snapshot: The report will cover the time period between two samples.
- By time: The report will cover the time period between two dates.
- Type.

Possible values:

- Standard: Provides the database load statistics for a specified time period.
- Differential: Provides comparative database load statistics for two specified time periods.
- Interval or Interval 1 and Interval 2: The time period covered by the report.

The period format depends on the value selected in Report:

- If you select By snapshot, specify the first and last samples.
- If you select By time, specify the start and end date and time.

3. Click Generate report.

Building a Report Using a Profiler Graph

You can also build a report when [viewing profiler graphs](#). In this case, the time period between two samples cannot be used as the time period that the report will cover.

Before performing this instruction:

- [Create an instance.](#)
- [Create a database.](#)
- [Create a profiler server.](#)
- [Take at least two samples.](#)

To build a report using a graph profiler:

1. Go to profiler graphs in one of the following ways:
 - Using the monitoring section:
 - a. In the navigation panel, go to Monitoring → Profiler.
 - b. From Instance, select the instance.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler.
2. From Database, select the database.
3. From Server, select the profiler server.
4. (Optional) In Period, specify the time period.

If you do not specify a value, the profiler graphs are displayed for the entire time range.

5. Click Select.
6. In the top-right corner of the profiler graph, click Select the period for the report.
7. Click and drag on the profiler graph to select the time period for the report.
8. (Optional) To include comparative database load statistics for two specified time periods in the report, click Select another period and repeat step 7.
9. Click Generate report.

Viewing All Built Reports

To view all built reports:

Go to reports in one of the following ways:

- Using the monitoring section:

In the navigation panel, go to Monitoring → Profiler → Reports.
- Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Profiler → Reports.

The table of reports with the following columns will be displayed:

- Instance: The instance where the report is built.
- Server: The profiler server whose directory contains samples.
- Database: The database for which the report is generated.

- Status.

Possible values:

- done: The report is built.
- pending: The report is being built.
- error: An error occurred while building a report.
- Period: The time period covered by the report.
- Execution start and Execution end: The start and end date and time of report building.
- User: The user that built the report.
- Actions.

For more information about available actions, refer to other instructions in this section.

Viewing and Downloading a Report

1. Go to reports in one of the following ways:

- Using the monitoring section:

In the navigation panel, go to Monitoring → Profiler → Reports.

- Using the configuration section of the specified instance:

a. In the navigation panel, go to Infrastructure → Instances.

b. Click the name of the instance.

c. In the navigation panel, go to Profiler → Reports.

2. Perform one of the following actions:

-

To view the report, click  next to it.

For more information about the report contents, refer to the [official Postgres Pro documentation](#).

- To download the report, in the top-right corner of the page, click Download report.

Deleting a Report

Important

Deleted reports cannot be restored.

To delete a report:

1. Go to reports in one of the following ways:

- Using the monitoring section:

In the navigation panel, go to Monitoring → Profiler → Reports.

- Using the configuration section of the specified instance:

a. In the navigation panel, go to Infrastructure → Instances.

b. Click the name of the instance.

c. In the navigation panel, go to Profiler → Reports.

2. Click  next to the report.
3. Click Delete.

5.24. Authentication

PPEM allows [viewing](#) authentication rules based on the host name of the instance (host-based authentication; HBA) and [editing](#) them. These rules are specified in the `pg_hba.conf` configuration file.

Viewing Authentication Rules

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Authentication.

The table with the `pg_hba.conf` configuration file information and the following columns will be displayed:

- Type: The type of the connection.

Possible values:

- local
- host
- hostssl
- hostnossl
- hostgssenc
- hostnogssenc
- Database: The databases to which the authentication rule applies.

The `all` value means that the rule applies to all databases.

- User: The users and groups to which the authentication rule is applied.

The `all` value means that the rule applies to all users.

- Address: The network address(es) of the client machine to which the authentication rule applies. It can include the name of the computer, IP range, or one of the keywords.

The `all` value means that the rule applies to all IP addresses.

This column is not used for rules with the `local` type.

- IP-mask: The mask of the IP address.

The value is displayed if only an IP address is specified in Address.

Specifying a mask in a separate column is an alternative for the `IP_address/mask_length` record.

This column is not used for rules with the `local` type.

- Method: The authentication method.

Possible values:

- trust
- reject
- scram-sha-256
- md5
- password

For the full list of possible authentication methods, refer to the [official Postgres Pro documentation](#).

- Options: The authentication method parameters in the `name=value` format.

For more information about available authentication method parameters, refer to the [official Postgres Pro documentation](#).

The rules in the table follow the same order as in the main `pg_hba.conf` configuration file. If the `include`, `include_if_exists`, and `include_dir` directives are used in this file, the rules listed in the included files are not displayed.

To view the full `pg_hba.conf` configuration file, in the top-right corner of the page, click View full file.

Editing Authentication Rules

PPEM provides the following ways of editing authentication rules:

- [Adding a line](#) to the `pg_hba.conf` configuration file
- Switching to the [edit mode](#) of the `pg_hba.conf` configuration file

Adding a Line to the Configuration File

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Authentication.
4. In the top-right corner of the page, click Add line.
5. Enter parameters of the new line in the `pg_hba.conf` configuration file (parameters marked with an asterisk are required):

- Type: The type of the connection.
- Users (search): The names of users and groups to which the authentication rule will apply.
- Users (will be saved to file): The names of users and groups to which the authentication rule will be applied.

You can specify a comma-separated list of users.

- Databases (search): The names of databases to which the authentication rule will apply.
- Databases (will be saved to file): The names of databases to which the authentication rule will apply.

You can specify a comma-separated list of databases.

- Address: The network address(es) of the client machine to which the authentication rule will apply.

This parameter is available only if you select any value except `local` from Type.

- IP-mask: The mask of the IP address.

This parameter is available only if you select any value except `local` from Type.

- Method: The authentication method.
- Options: The authentication method parameters in the `name=value` format.

6. Click Add.
7. (Optional) To reset all added but not yet saved lines, in the bottom-right corner of the page, click Reset.
8. In the bottom-right corner of the page, click Save.

Configuration File Editing Mode

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Authentication.
4. In the top-right corner of the page, turn on Edit mode.
5. Perform one of the following actions with the line:
 - To move the line one position up, click  next to it.
 - To move the line one position down, click  next to it.
 - To delete the line, click  next to it.
6. (Optional) To reset all edited but not yet saved lines, in the bottom-right corner of the page, click Reset.
7. In the bottom-right corner of the page, click Save.

Note

When rules are saved, the instance receives a configuration reload signal. In this case, all configuration files of the instance are reread.

5.25. Instance Roles

Instance roles provide information about database roles within an instance and are based on the `pg_roles` view.

To view instance roles:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Roles.
4. (Optional) To display system instance roles, turn on Show system roles.

The table of instance roles includes the following columns:

- Role: The name of the instance role.
- Login: Specifies whether the instance role has the `LOGIN` privilege.
- Create role: Specifies whether the instance role has the `CREATEROLE` privilege.
- Superuser: Specifies whether the instance role has the `SUPERUSER` privilege.
- Create DB: Specifies whether the instance role has the `CREATEDB` privilege.
- Inheritance of privileges: Specifies whether the instance role automatically inherits privileges of the instance roles it is a member of.
- Replication: Specifies whether the instance role is a replication instance role. Such roles can initiate replication connections and create and drop replication slots.
- Bypass RLS: Specifies whether the instance role bypasses row level security policies.

For more information, refer to the official Postgres Pro documentation on [row security policies](#).

- **Connection limit:** The maximum number of concurrent connections this instance role can make. Applies to roles that can connect to the server.
- **Valid until:** The password expiration time. Only used for password authentication.
- **Member_of:** The instance roles that include this role as a member.
- **Include roles:** The instance roles that are members of this role.

5.26. Message Log

PPEM logs DBMS instances. To provide logging, you must first [install and configure logging tools](#).

To view the message log of all instances, in the navigation panel, go to Monitoring → Message journal.

To view the message log of the specified instance:

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Message journal.

The table of messages with the following columns will be displayed:

- **Date, time:** The date and time when the message was created.
- **Instance:** The instance associated with the message.
- **Session:** The ID of the user session where the message was created.
- **Application:** The source application of the message.
- **User:** The user on behalf of which the session was initiated.
- **Database:** The database associated with the user session.
- **Type.**

Possible values:

- DEBUG1
- DEBUG2
- DEBUG3
- DEBUG4
- DEBUG5
- INFO
- NOTICE
- WARNING
- ERROR
- LOG
- FATAL
- PANIC

For more information about message types, refer to the [official Postgres Pro documentation](#).

- **SQLSTATE code:** The SQLSTATE code of the message.
- **Message.**
- **Details:** The detailed text of the message.
- **Hint:** The error hint.

Only values of the `ERROR`, `FATAL`, and `PANIC` messages are displayed in this column.

- Query ID: The ID of the query associated with the message.

You can use this ID to find statistics in the `pg_stat_activity`, `pg_stat_statements`, and `pgprostats` views.

- Query: The text of the query associated with the message.

5.27. Task Console

A user or the [manager](#) can initiate the execution of tasks on [agents](#) and instances, as well as create schedules for such tasks.

You can also use mass instance operations to perform an operation on multiple instances at once.

5.27.1. Viewing Tasks

In the navigation panel, go to Tasks console.

The table of tasks with the following columns will be displayed:

- Name: The unique name of the task with a description.
- Agent: The agent assigned to execute the task.

This column includes additional information:

Instance: The instance where the task was executed.

- Status.

Possible values:

- COMPLETED: The task was completed.
- FAILED: An error occurred while executing the task.
- Started at and Finished at: The start and end time of the task execution.
- Actions.

To view detailed information about the task, click  next to it.

The window with the following parameters will open:

- Status: The task execution status.
- ID: The task ID.
- Instance: The instance where the task was executed.
- Start and End: The task execution time.
- Log: The log entry about the task.

This information is available when the task is in the `COMPLETED` status.

- Errors: The errors occurred during the task execution.

This information is available when the task is in the `FAILED` status.

5.27.2. Managing Schedules

This section explains how to manage schedules. It includes the following instructions:

- [Viewing Schedules](#)

- [Deactivating and Activating a Task](#)
- [Deleting a Task](#)

Viewing Schedules

In the navigation panel, go to Tasks console → Schedule.

The table of schedules with the following columns will be displayed:

- Task: The unique name of the task with a description.

To view the task execution status, hover over its name.

Possible values:

- Active: The task is activated.
- Deactivated: The task is deactivated.
- Instances: The instance where the schedule is created.
- User: The user on whose behalf the task was created.
- Schedule: The string in the crontab format that specifies the schedule.
- Parameters: The task execution parameters.
- Actions.

For more information about available actions, refer to other instructions in this section.

Deactivating and Activating a Task

You can deactivate or activate the task to stop or continue its execution.

To deactivate or activate the task:

1. In the navigation panel, go to Tasks console → Schedule.

2. Click  or  next to the task.

Deleting a Task

Important

Deleted tasks cannot be restored.

To delete a task:

1. In the navigation panel, go to Tasks console → Schedule.

2. Click  next to the task.

3. Click Delete.

5.27.3. Managing Mass Instance Operations

Mass instance operations allow you to execute an operation on multiple instances at once instead of doing this on each instance separately.

This section explains how to manage mass instance operations. It includes the following instructions:

- [Creating a Mass Instance Operation](#)
- [Viewing Mass Instance Operations](#)
- [Viewing Detailed Mass Operation Information](#)
- [Canceling a Mass Instance Operation](#)

Important

The mass instance operations functionality is in the beta phase. Currently, the only supported mass instance operation is applying a [configuration preset](#). Other existing limitations are specified in the corresponding instructions of this section.

Creating a Mass Instance Operation

1. In the navigation panel, go to Tasks console → Operations.
2. In the top-right corner of the page, click Create operation.
3. Enter parameters of the new mass instance operation (parameters marked with an asterisk are required):

- Operation: The type of the mass instance operation.

Currently, the only supported mass instance operation is applying a [configuration preset](#), and the value is entered automatically.

- Configuration preset: The configuration preset that will be applied to the instances.
- Error limit: The amount of errors that can occur when executing the operation on an instance. If the error limit is exceeded, the operation is canceled.

Currently, the error limit cannot be set, and the -1 value is entered automatically.

4. Click Next, and then select the instances to which the configuration preset will be applied.

The configuration preset is applied to the instances in the order you select them.

Important

For nodes of a standard primary-standby or BiHA [cluster](#), consider the following:

- if you apply a configuration preset with more parameters than the previous one, first apply this preset to standby/follower nodes
- if you apply a configuration preset with fewer parameters than the previous one, first apply this preset to the primary/leader node

Applying a configuration preset in the wrong order may cause standby/follower nodes to shut down.

5. Click Next, and then, in Compatibility check, select whether to verify compatibility between the configuration preset and the instance before applying the preset. If the compatibility check fails, the configuration preset is not applied to the instance.

The compatibility check does not guarantee that no errors will occur when applying the configuration preset.

You can select one of the following values:

- For each instance

This is the recommended value.

- No check

6. Click Execute operation.

Viewing Mass Instance Operations

In the navigation panel, go to Tasks console → Operations.

- Name.
- Status.

Possible values:

- Scheduled: The mass instance operation was scheduled.
- Executing: The mass instance operation is being executed.
- Error: An error occurred when executing the mass instance operation.
- Canceling: The mass instance operation is being canceled.
- Canceled: The mass instance operation was canceled.
- Completed: The mass instance operation was executed.
- Owner: The user that executed the mass instance operation.
- Last execution: The date and time when the mass instance operation started executing.
- Duration: The duration of the mass instance operation.

This column includes additional information:

Completed: The date and time when the mass instance operation stopped executing.

- Actions.

For more information about available actions, refer to other instructions in this section.

Viewing Detailed Mass Operation Information

1. In the navigation panel, go to Tasks console → Operations.

2.

Click  next to the mass operation.

The window with the following detailed information about the mass instance operation will open:

- Operation: The unique name of the mass instance operation.

Status: The status of the mass instance operation and the number of performed steps.

Possible values:

- Scheduled: The mass instance operation was scheduled.
- Executing: The mass instance operation is being executed.
- Error: An error occurred when executing the mass instance operation.
- Canceling: The mass instance operation is being canceled.
- Canceled: The mass instance operation was canceled.
- Completed: The mass instance operation was executed.
- Configuration preset: The [configuration preset](#) being applied to the instances.

- **Compatibility check:** Specifies whether compatibility between the configuration preset and the instance is verified before applying the preset. If the compatibility check fails, the configuration preset is not applied to the instance.

The compatibility check does not guarantee that no errors will occur when applying the configuration preset.

Possible values:

- For each instance
- No check
- **Selected instances:** The instances to which the configuration preset is being applied.

Canceling a Mass Instance Operation

When you cancel a mass instance operation, its execution stops, but already executed operations are not reverted.

To cancel a mass instance operation:

1. In the navigation panel, go to Tasks console → Operations.
2. Click  next to the mass operation.
3. Click Cancel.

5.28. Backup

PPEM supports creating backups for instances. You can use a backup to create a new instance if the previous instance fails. To work with backups, you must first [install and configure the backup and restore tools](#).

Before creating backups, create storages where they will be located. You can create local and S3 storages.

Note

Local storages are intended for introductory use with PPEM. For production environments with a large number of instances and created backups, it is recommended to use S3 storages.

When creating a storage, you must bind an instance to it. When an instance is bound to a storage, a catalog for this instance's backups is created in the storage.

You cannot bind an instance to the same storage twice. If an instance is bound for the second time, it is unbound from the previous storage. You can manage instance binds separately.

Backup schedules can be created for automatically creating instance backups at the specified time interval. Backup schedules also allow for scheduling a single backup at the specified date and time.

5.28.1. Storages

This section explains how to manage storages. It includes the following instructions:

- [Creating a Storage](#)
- [Viewing Storages](#)
- [Editing a Storage](#)
- [Deleting a Storage](#)

Creating a Storage

You can create [local storages](#) and [S3 storages](#).

Creating a Local Storage

1. In the navigation panel, go to Backup → Backup storages.
2. In the top-right corner of the page, click Create backup storage.
3. Select Local storage, and then click Next.
4. Enter parameters of the new local storage (parameters marked with an asterisk are required):

- Storage name.
- Catalog of copies: The path to the local storage catalog where backups will be located.

Specify the path to an empty catalog.

- System user: The OS user who will own the local storage catalog.

It is recommended to specify the user on behalf of which the instance is installed, because this user must have read and write privileges in the local storage catalog.

- Instance: The instance that will be [bound](#) to the local storage. When an instance is bound to a storage, a catalog for this instance's backups is created in the storage.

You cannot bind an instance to the same storage twice. If an instance is bound for the second time, it is unbound from the previous storage.

- Storage parameters: The backup storage parameters of the storage catalog created for the instance.

Available parameters:

- Retention redundancy, pcs.: The maximum number of full backups.

For example, if you specify 3, the catalog can contain a maximum of three full backups.

To disable this limitation, specify 0. In this case, the number of backups in the catalog is not limited.

- Retention window, days: The number of days (24 hours) covered by backups.

For example, if you specify 7, the catalog must always contain backups required for restoring the data for the last seven days, including today.

To disable this limitation, specify 0. In this case, backups can be deleted from the catalog at any moment.

- WAL depth, pcs.: The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR).

For example, if you specify 3, the catalog must always contain at least three backups on each timeline.

To disable this limitation, specify 0. In this case, point-in-time recovery is not available.

- Expired copies: The management policy for expired backups.

Possible values:

- Merge: Merge expired backups with new ones if possible.
- Delete: Delete expired backups from the catalog.

You can enable all checkboxes simultaneously.

Retention redundancy, pcs, Retention window, days, and WAL depth, pcs are applied only if you enable Merge and/or Delete in Expired copies.

The values of Retention redundancy, pcs. and Retention window, days are considered simultaneously when deleting expired backups from the catalog.

For example, if you enter 3 in Retention redundancy, pcs. and 7 in Retention window, days, a maximum of three full backups are saved, as well as all backups required for restoring the data for the past 7 days, including today.

You can also configure storage parameters for an [instance](#), as well as for a backup when [creating](#) it.

The following priority is applied:

- backup parameters are applied first
- instance parameters are applied second
- storage parameters are applied third

For more information about storage parameters, refer to the official Postgres Pro documentation on [pg_probackup](#).

5. Click Save.

Creating an S3 Storage

1. In the navigation panel, go to Backup → Backup storages, and then select S3 storages.
2. In the top-right corner of the page, click Create backup storage.
3. Select S3 storage, and then click Next.
4. Enter parameters of the new S3 storage (parameters marked with an asterisk are required):

- Storage name.
- Type: The provider of the S3 storage.

Possible values:

- AWS
- Minio
- VK
- Host: The server where the commands for interactions between the [manager](#) and S3 storage will be executed.
- Server name: The network address of the S3 storage server.
- Protocol https: Specifies whether the HTTPS is used for interactions between the manager and S3 storage.
- Port: The number of the port for connecting the manager to the S3 storage.
- Access Key ID and Secret access key: The secure keys for connecting the manager to the S3 storage.
- Bucket: The name of the bucket on the S3 storage server where backups will be located.
- Region: The region where the S3 storage server is located.
- Catalog of copies: The path to the bucket catalog where backups will be located.
- Instance: The instance that will be [bound](#) to the S3 storage. When an instance is bound to a storage, a catalog for this instance's backups is created in the storage.

You cannot bind an instance to the same storage twice. If an instance is bound for the second time, it is unbound from the previous storage.

- **Storage parameters:** The backup storage parameters of the storage catalog created for the instance.

Available parameters:

- **Retention redundancy, pcs.:** The maximum number of full backups.

For example, if you specify 3, the catalog can contain a maximum of three full backups.

To disable this limitation, specify 0. In this case, the number of backups in the catalog is not limited.

- **Retention window, days:** The number of days (24 hours) covered by backups.

For example, if you specify 7, the catalog must always contain backups required for restoring the data for the last seven days, including today.

To disable this limitation, specify 0. In this case, backups can be deleted from the catalog at any moment.

- **WAL depth, pcs.:** The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR).

For example, if you specify 3, the catalog must always contain at least three backups on each timeline.

To disable this limitation, specify 0. In this case, point-in-time recovery is not available.

- **Expired copies:** The management policy for expired backups.

Possible values:

- **Merge:** Merge expired backups with new ones if possible.
- **Delete:** Delete expired backups from the catalog.

You can enable all checkboxes simultaneously.

Retention redundancy, pcs, Retention window, days, and WAL depth, pcs are applied only if you enable Merge and/or Delete in Expired copies.

The values of Retention redundancy, pcs. and Retention window, days are considered simultaneously when deleting expired backups from the catalog.

For example, if you enter 3 in Retention redundancy, pcs. and 7 in Retention window, days, a maximum of three full backups are saved, as well as all backups required for restoring the data for the past 7 days, including today.

You can also configure storage parameters for an [instance](#), as well as for a backup when [creating](#) it.

The following priority is applied:

- backup parameters are applied first
- instance parameters are applied second
- storage parameters are applied third

For more information about storage parameters, refer to the official Postgres Pro documentation on [pg_probackup](#).

5. Click Save.

Viewing Storages

To view storages, in the navigation panel, go to Backup → Backup storages.

To view S3 storages, select S3 storages.

The table of storages with the following columns will be displayed:

- Name.
- Instance: The instance where the catalog of the local storage is located.

This column is only displayed on the Local tab.

- Catalog of copies: The path to the catalog where backups are located.

This column is only displayed on the Local tab.

- Parameters:

- Type: The provider of the S3 storage.

Possible values:

- AWS
- Minio
- VK
- Host: The server where the commands for interactions between the manager and S3 storage are executed.
- Port: The number of the port for connecting the manager to the S3 storage.
- Bucket: The name of the bucket on the S3 storage server where backups are located.
- Region: The region where the S3 storage server is located.

This column is only displayed on the S3 storages tab.

- Retention redundancy, pcs.: The maximum number of full backups in a storage.
- Retention window, days: The number of days (24 hours) covered by backups in the storage.
- WAL depth, pcs.: The minimum number of backups on each timeline in a storage. Having backups on all timelines is required for point-in-time recovery (PITR).
- Expired copies: The management policy for expired backups.

Possible values:

- Disabled: Do not perform any actions with expired backups.
- Merge: Merge expired backups with new ones if possible.
- Delete: Delete expired backups from the catalog.

All values except Disabled can be displayed simultaneously.

Editing a Storage

1. In the navigation panel, go to Backup → Backup storages.

To edit an S3 storage, select S3 storages.

- 2.

Click  next to the storage.

3. Edit storage parameters.
4. Click Save.

Deleting a Storage

1. In the navigation panel, go to Backup → Backup storages.

To delete an S3 storage, select S3 storages.

2. Click  next to the storage.
3. Select one of the following values:
 - Remove from repository only: Delete the storage from the repository and [web application](#).
When the [agent](#) is restarted on the server, the storage is automatically recreated in the web application.
 - Recursive removing of all backups and storage: Delete the storage and its catalog from the repository, web application, and server.

Important

If you select this value, the deleted storage cannot be restored.

4. Confirm the operation and click Delete.

5.28.2. Instance Binds

This section explains how to manage instance binds. It includes the following instructions:

- [Binding an Instance to a Storage](#)
- [Viewing Instance Binds](#)
- [Editing an Instance Bind](#)
- [Deleting an Instance Bind](#)

Before performing these instructions, [create a storage](#).

Binding an Instance to a Storage

You cannot bind an instance to the same storage twice. If an instance is bound for the second time, it is unbound from the previous storage.

To bind an instance to a storage:

1. In the navigation panel, go to Backup → Backup storages.
To view instance binds to an S3 storage, select S3 storages.
2. Click the name of the storage.
3. In the top-right corner of the page, click Bind instance.

The following storage parameters will be displayed:

- Local storage or S3 storage: The unique name of the storage.
 - Catalog: The path to the storage catalog where backups are placed.
 - System user: The operating system user that owns the storage catalog.
4. Enter parameters of the new instance bind (parameters marked with an asterisk are required):
 - Identifier: The unique name of the instance bind.
 - Instance: The instance that will be bound to the storage.
 - Storage parameters: The backup storage parameters of the storage catalog created for the instance.

Available parameters:

- Retention redundancy, pcs.: The maximum number of full backups.

For example, if you specify 3, the catalog can contain a maximum of three full backups.

To disable this limitation, specify 0. In this case, the number of backups in the catalog is not limited.

- Retention window, days: The number of days (24 hours) covered by backups.

For example, if you specify 7, the catalog must always contain backups required for restoring the data for the last seven days, including today.

To disable this limitation, specify 0. In this case, backups can be deleted from the catalog at any moment.

- WAL depth, pcs.: The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR).

For example, if you specify 3, the catalog must always contain at least three backups on each timeline.

To disable this limitation, specify 0. In this case, point-in-time recovery is not available.

- Expired copies: The management policy for expired backups.

Possible values:

- Merge: Merge expired backups with new ones if possible.
- Delete: Delete expired backups from the catalog.

You can enable all checkboxes simultaneously.

Retention redundancy, pcs, Retention window, days, and WAL depth, pcs are applied only if you enable Merge and/or Delete in Expired copies.

The values of Retention redundancy, pcs. and Retention window, days are considered simultaneously when deleting expired backups from the catalog.

For example, if you enter 3 in Retention redundancy, pcs. and 7 in Retention window, days, a maximum of three full backups are saved, as well as all backups required for restoring the data for the past 7 days, including today.

You can also configure storage parameters for a backup when [creating](#) it, as well as for a storage when [creating](#) or [editing](#) it.

The following priority is applied:

- backup parameters are applied first
- instance parameters are applied second
- storage parameters are applied third

For more information about storage parameters, refer to the official Postgres Pro documentation on [pg_probackup](#).

5. Click Save.

Viewing Instance Binds

1. In the navigation panel, go to Backup → Backup storages.

To view instance binds to an S3 storage, select S3 storages.

2. Click the name of the storage.

The table of instance binds with the following columns will be displayed:

- Binding name.

- Instance: The bound instance.
- Retention redundancy, pcs.: The maximum number of full backups in a storage.
- Retention window, days: The number of days (24 hours) covered by backups in the storage.
- WAL depth, pcs.: The minimum number of backups on each timeline in a storage. Having backups on all timelines is required for point-in-time recovery (PITR).
- Actions.

For more information about available actions, refer to other instructions in this section.

The following storage parameters are displayed at the top of the table of instance binds:

- When viewing instance binds to a local storage:
 - Catalog: The path to the local storage catalog where backups are located.
 - System user: The operating system user that owns the local storage catalog.
- When viewing binds to an S3 storage:
 - Provider: The provider of the S3 storage.
 - Region: The region where the S3 storage server is located.
 - Protocol: The protocol used for interactions between the manager and S3 storage.
 - Host: The server where the commands for interactions between the manager and S3 storage are executed.
 - Port: The number of the port for connecting the manager to the S3 storage.
 - Catalog: The path to the bucket catalog where backups are located.
 - Bucket: The name of the bucket on the S3 storage server where backups are located.

Editing an Instance Bind

You cannot bind an instance to the same storage twice. If an instance is bound for the second time, it is unbound from the previous storage.

To edit an instance bind:

1. In the navigation panel, go to Backup → Backup storages.
To view instance binds to an S3 storage, select S3 storages.
2. Click the name of the storage.
3. Click  next to the instance bind.
4. Edit instance bind parameters.
5. Click Save.

Deleting an Instance Bind

Important

Deleted instance binds cannot be restored.

To delete an instance bind:

1. In the navigation panel, go to Backup → Backup storages.
To view instance binds to an S3 storage, select S3 storages.

2. Click the name of the storage.
3.  Click next to the instance bind.
4. Confirm the operation and click Delete.

5.28.3. Backups

This section explains how to manage backups. It includes the following instructions:

- [Creating a Backup](#)
- [Viewing Backups](#)
- [Checking the Consistency of a Backup](#)
- [Viewing the Log of a Backup](#)
- [Editing Pinning Parameters of a Backup](#)
- [Creating an Instance from a Backup](#)
- [Deleting a Backup](#)

Creating a Backup

1. Go to backups in one of the following ways:
 - Using the backup section:

In the navigation panel, go to Backup.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Backup.
2. In the top-right corner of the page, click Create backup.
3. Enter parameters of the new backup (parameters marked with an asterisk are required):
 - Instance: The instance for which the backup will be created.

The value is entered automatically when creating a backup using the table of backups of the specified instance.
 - Copy storage: The storage where the backup will be located.

You can select a local or S3 storage. The local storage must be located on the same server as the instance for which you are creating the backup.
 - User and Password: The name and password of the DBMS user under which the backup will be performed.
 - Database: The database for connecting to the instance.
 - Backup mode: The backup creation mode.

Possible values:

 - full
 - page
 - ptrack
 - delta

For more information about backup modes, refer to the official Postgres Pro documentation on [pg_probackup](#).
 - Threads count: The number of parallel threads that will be started at backup creation.

- **Waiting time (sec):** The waiting timeout in seconds for WAL segment archiving and streaming.
- **Create a stand-alone backup:** Specifies whether to create a streaming backup that includes the WAL records required for restoring the instance later.
- **Replication slot:** The replication slot that will be used for transferring WAL records.
- **Create temporary replication slot:** Specifies whether to create a temporary replication slot for transferring WAL records of the instance for which you are creating the backup.

If you enable this checkbox, WAL segments are available even when they are switched at backup creation.

4. Click Next, and then specify additional parameters if required:

- **External catalogs:** The path to the instance catalog that will also be included in the backup.
- **include log catalog:** Specifies whether the backup includes the catalog with the instance activity logs.
- **Don't check copy:** Specifies whether to skip the automatic verification of the created backup.

If you enable this checkbox, the backup is created faster.

- **Smooth execution of the checkpoint:** Specifies whether backup creation starts only after the scheduled checkpoint.
- **Disable block-level verification:** Specifies whether to disable the block-level checksum verification for faster consistency checking at backup creation.
- **Compression level:** The file compression level at backup creation.

You can enter a value from 0 to 9, where 0 disables file compression, and 9 enables the highest file compression level.

- **Compression algorithm:** The algorithm used for compressing files.

Possible values:

- zlib
- lz4
- zstd
- pglz

This parameter is available only if you enter a value greater than 0 in Compression level.

- **Pinning:** The pinning parameters of the backup.

Possible values:

- **Do not pin:** Do not pin the backup.

If you select this value, the parameters specified in Parameters of storing are used.

- **ttl:** After the backup is created, it cannot be deleted from a storage during the specified number of days.

For this value, in Retention period, days, enter the number of days.

- **expire-time:** The backup cannot be deleted from a storage until the specified date and time.

For this value, in Retention period until, specify the date and time.

- **Storage parameters:** The backup storage parameters of the storage catalog created for the instance.

- Retention redundancy: The maximum number of full backups.

For example, if you specify 3, the catalog can contain a maximum of three full backups.

To disable this limitation, specify 0. In this case, the number of backups in the catalog is not limited.

- Retention window: The number of days (24 hours) covered by backups.

For example, if you specify 7, the catalog must always contain backups required for restoring the data for the last seven days, including today.

To disable this limitation, specify 0. In this case, backups can be deleted from the catalog at any moment.

- WAL depth: The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR).

For example, if you specify 3, the catalog must always contain at least three backups on each timeline.

To disable this limitation, specify 0. In this case, point-in-time recovery is not available.

- Expired copies: The management policy for expired backups.

Possible values:

- Merge: Merge expired backups with new ones if possible.
- Delete: Delete expired backups from the catalog.
- Remove expired WAL: Delete WAL of expired backups from the catalog.

You can enable all checkboxes simultaneously.

Retention redundancy, Retention window, and WAL depth are applied only if you enable Merge and/or Delete in Expired copies.

The values of Retention redundancy and Retention window are considered simultaneously when deleting expired backups from the catalog.

For example, if you enter 3 in Retention redundancy and 7 in Retention window, a maximum of three full backups are saved, as well as all backups required for restoring the data for the past 7 days, including today.

You can also configure storage parameters for an [instance](#), as well as for a storage when [creating](#) or [editing](#) it.

The following priority is applied:

- backup parameters are applied first
- instance parameters are applied second
- storage parameters are applied third

For more information about storage parameters, refer to the official Postgres Pro documentation on [pg_probackup](#).

5. Click Execute backup.

Viewing Backups

Go to backups in one of the following ways:

- Using the backup section:

In the navigation panel, go to Backup.

- Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Backup.

The table of backups with the following columns will be displayed:

- Size.

This column includes additional information:

ID: The ID of the backup in `pg_probackup`.

- Mode: The backup creation mode.

Possible values:

- FULL
- DELTA
- PAGE
- PTRACK

For more information about backup modes, refer to the official Postgres Pro documentation on [pg_probackup](#).

- Storage: The storage where the backup is located.
- Instance: The instance for which the backup is created.
- Status.

Possible values:

- REQUESTED: Backup creation was requested.
- SCHEDULED: The backup creation was scheduled.
- RUNNING: The backup is being created.
- DONE: The backup was created successfully.
- DELETING_CANCEL: Backup deletion was canceled.
- DELETING: The backup is being deleted.
- VALIDATING: The backup consistency check is being performed.
- VALIDATED: The consistency check demonstrated that the backup was not damaged during storage.
- MERGING: An expired backup is being merged with a new one.
- MERGED: An expired backup was successfully merged with a new one.
- ERROR: An error occurred during backup.

This status is also displayed if the consistency check showed that the backup was damaged during storage.

- Started at and Finished at: The start and end date of backup creation.
- User: The user that created the backup.
- Validation status: The status of the backup consistency check.

Possible values:

- validated: The consistency check of the backup was performed successfully.
- validating: The backup consistency check is being performed.
- Start of validation and End of validation: The start and end date and time of the backup consistency check.
- Actions.

For more information about available actions, refer to other instructions in this section.

Checking the Consistency of a Backup

The consistency check ensures that a backup was not damaged during storage.

To check the consistency of a backup:

1. Go to backups in one of the following ways:
 - Using the backup section:
In the navigation panel, go to Backup.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Backup.

2. Click  next to the backup.

You can view the result of the consistency check in the Status column of the table of backups:

- If the backup was damaged during storage, the `ERROR` value will be displayed.
- If the backup was not damaged during storage, the `VALIDATED` value will be displayed.

Viewing the Log of a Backup

The backup log provides information about the backup creation progress.

To view the log of a backup:

1. Go to backups in one of the following ways:
 - Using the backup section:
In the navigation panel, go to Backup.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Backup.

2. Click  next to the backup.

Editing Pinning Parameters of a Backup

1. Go to backups in one of the following ways:
 - Using the backup section:
In the navigation panel, go to Backup.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Backup.

2. Click  → Edit pinning next to the backup.
3. Select one of the following values:
 - Do not pin: Apply backup storage parameters.
 - ttl: After the backup storage parameters are edited, the backup cannot be deleted during the specified number of days.

For this value, in Retention period, days, enter the number of days.
 - expire-time: The backup cannot be deleted until the specified date and time.

For this value, in Retention period until, specify the date and time.
4. Click Save.

Creating an Instance from a Backup

Before performing this instruction, [create a backup](#).

To create an instance from a backup:

1. Go to backups in one of the following ways:
 - Using the backup section:

In the navigation panel, go to Backup.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Backup.
2. Click  → Restore next to the backup.
3. Enter parameters of the new instance (parameters marked with an asterisk are required):
 - Name.
 - Server: The server where the instance is installed.
 - System user: The OS user who will own the instance files and catalogs and on behalf of which the instance service will run. In most cases, it is the `postgres` user.

It is recommended to ensure that the specified user exists in the OS.
 - Main data directory: The path to the server catalog where the main instance catalogs and files will be located.
 - Connection address and Connection port: The network address and port number that the instance will use for receiving client connections.
 - Tags: The [tags](#) that will be assigned to the instance.
 - Backup: The backup from which the instance will be created.

The value is entered automatically.
 - Backup size: The size of the backup from which the instance will be created.

The value is entered automatically.
 - Restore point: The state to which the instance must be restored.

Possible values:

- —: Restore the instance to the last state covered by the backup.
- Time: Restore the instance state to the specified date and time covered by the backup.

For this value, in Time, specify the date and time.

- LSN: Restore the instance to the state corresponding to the specified WAL LSN.

For this value, in LSN, enter the WAL LSN.

- Transaction: Restore the instance state to the specified transaction number.

For this value, in Transaction, enter the transaction number.

For Time, LSN, and Transaction, specify the following parameters:

- Restore including the specified value: Specifies whether the instance state is restored including the specified value.

For example, if you enter 123456 in Transaction and enable Restore including the specified value, the instance state is restored to the transaction 123456. If you do not enable Restore including the specified value, the instance state is restored to the 123455 transaction.

- Action after restore: The action to perform on the server after restoring the instance to the state.

Possible values:

- Pause after restore: Pause the creation of the instance from the backup.

It allows ensuring that the correct state was restored for the instance before creating it.

- Promote after restore: Create the instance from the backup and start receiving client connections.
- Shutdown instance after restore: Create the instance from the backup, and then stop the server.

- Partial recovery: Specifies the instance databases that will be restored or excluded from the restoration process.

Possible values:

- Do not use: Restore all instance databases.
- Exclude some databases: Exclude the specified databases from restoration.
- Restore some databases: Restore the specified instance databases.

For Exclude some databases and Restore some database, specify the unique name of the database using Databases, and then click Add database.

- Checking available space: Allows checking whether there is enough disk space on the server for creating the instance from the backup.

To start the check, click Check.

4. Click Execute.

Deleting a Backup

Important

Deleted backups cannot be restored.

To delete a backup:

1. Go to backups in one of the following ways:
 - Using the backup section:
In the navigation panel, go to Backup.
 - Using the configuration section of the specified instance:
 - a. In the navigation panel, go to Infrastructure → Instances.
 - b. Click the name of the instance.
 - c. In the navigation panel, go to Backup.
2. Click  → Delete next to the backup.
3. Click Delete.

5.28.4. Backup Schedules

This section explains how to manage backup schedules. It includes the following instructions:

- [Creating a Backup Schedule](#)
- [Viewing Backup Schedules](#)
- [Executing a Backup Schedule](#)
- [Deactivating and Activating a Backup Schedule](#)
- [Deleting a Backup Schedule](#)

Creating a Backup Schedule

1. In the navigation panel, go to Backup → Schedule.
2. In the top-right corner of the page, click Create task.
3. Enter parameters of the new backup schedule (parameters marked with an asterisk are required):
 - Name.
 - Set cron-string execution: Allows specifying the time interval for creating backups in the crontab format.

If you turn on this toggle, in Execution, enter the string.

- Task planning: The type of the backup schedule.

Possible values:

- Time-delayed: One backup will be created at the specified date and time.
- On schedule: Backups will be created at the specified time interval.

For this value, specify the following parameters:

- Interval: The unit of measurement of the time interval.

Possible values:

- Minutes
- Hours
- Days
- Repeat every: The time interval for creating backups by minutes or hours.

This parameter is available only if you select Minutes or Hours from Interval.

- Execution days: The days when backups will be created.
- Cron total line: The string in the crontab format that specifies the time interval for creating backups.

The value is entered automatically.

This parameter is available only if you turn off Set cron-string execution.

- Time: The date and/or time when the backup will be created.

This parameter is available only if you select Time-delayed from Task planning or Days from Interval.

- Start and Repeat until: The start and end date and time for creating backups on schedule.

These parameters are available only if you turn on Set cron-string execution or select On schedule from Task planning.

- Instance: The instance for which the backup will be created.
- Copy storage: The storage where the backup will be located.

You can select a local or S3 storage. The local storage must be located on the same server as the instance for which you are creating the backup.

- User and Password: The name and password of the DBMS user under which the backup will be performed.
- Database: The database for connecting to the instance.
- Backup mode: The backup creation mode.

Possible values:

- full
- page
- ptrack
- delta

For more information about backup modes, refer to the official Postgres Pro documentation on [pg_probackup](#).

- Threads count: The number of parallel threads that will be started at backup creation.
- Waiting time (sec): The waiting timeout in seconds for WAL segment archiving and streaming.
- Create a stand-alone backup: Specifies whether to create a streaming backup that includes the WAL records required for restoring the instance later.
- Replication slot: The replication slot that will be used for transferring WAL records.
- Create temporary replication slot: Specifies whether to create a temporary replication slot for transferring WAL records of the instance for which you are creating the backup.

If you enable this checkbox, WAL segments are available even when they are switched at backup creation.

4. Click Next, and then specify additional parameters if required:

- External catalogs: The path to the instance catalog that will also be included in the backup.
- include log catalog: Specifies whether the backup includes the catalog with the instance activity logs.
- Don't check copy: Specifies whether to skip the automatic verification of the created backup.

If you enable this checkbox, the backup is created faster.

- Smooth execution of the checkpoint: Specifies whether backup creation starts only after the scheduled checkpoint.
- Disable block-level verification: Specifies whether to disable the block-level checksum verification for faster consistency checking at backup creation.
- Compression level: The file compression level at backup creation.

You can enter a value from 0 to 9, where 0 disables file compression, and 9 enables the highest file compression level.

- Compression algorithm: The algorithm used for compressing files.

Possible values:

- zlib
- lz4
- zstd
- pglz

This parameter is available only if you enter a value greater than 0 in Compression level.

- Pinning: The pinning parameters of the backup.

Possible values:

- Do not pin: Do not pin the backup.

If you select this value, the parameters specified in Parameters of storing are used.

- ttl: After the backup is created, it cannot be deleted from a storage during the specified number of days.

For this value, in Retention period, days, enter the number of days.

- expire-time: The backup cannot be deleted from a storage until the specified date and time.

For this value, in Retention period until, specify the date and time.

- Storage parameters: The backup storage parameters of the storage catalog created for the instance.

Available parameters:

- Retention redundancy: The maximum number of full backups.

For example, if you specify 3, the catalog can contain a maximum of three full backups.

To disable this limitation, specify 0. In this case, the number of backups in the catalog is not limited.

- Retention window: The number of days (24 hours) covered by backups.

For example, if you specify 7, the catalog must always contain backups required for restoring the data for the last seven days, including today.

To disable this limitation, specify 0. In this case, backups can be deleted from the catalog at any moment.

- WAL depth: The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR).

For example, if you specify 3, the catalog must always contain at least three backups on each timeline.

To disable this limitation, specify 0. In this case, point-in-time recovery is not available.

- Expired copies: The management policy for expired backups.

Possible values:

- Merge: Merge expired backups with new ones if possible.
- Delete: Delete expired backups from the catalog.
- Remove expired WAL: Delete WAL of expired backups from the catalog.

You can enable all checkboxes simultaneously.

Retention redundancy, Retention window, and WAL depth are applied only if you enable Merge and/or Delete in Expired copies.

The values of Retention redundancy and Retention window are considered simultaneously when deleting expired backups from the catalog.

For example, if you enter 3 in Retention redundancy and 7 in Retention window, a maximum of three full backups are saved, as well as all backups required for restoring the data for the past 7 days, including today.

You can also configure storage parameters for an [instance](#), as well as for a storage when [creating](#) or [editing](#) it.

The following priority is applied:

- backup parameters are applied first
- instance parameters are applied second
- storage parameters are applied third

For more information about storage parameters, refer to the official Postgres Pro documentation on [pg_probackup](#).

5. Click Save.

Viewing Backup Schedules

In the navigation panel, go to Backup → Schedule.

The table of backup schedules with the following columns will be displayed:

- Task: The unique name of the backup schedule.
- Agent: The agent that creates backups.

This column includes additional information:

Instance: The instance for which backups will be created.

- Storage: The storage where backups are located.
- Compression: The file compression level at backup creation.

You can enter a value from 0 to 9, where 0 disables file compression, and 9 enables the highest file compression level.

- Last execution: The date and time of the last backup creation.
- Schedule: The string in the crontab format that specifies the time interval for creating backups.
- User: The user that created the backup schedule.

Executing a Backup Schedule

You can manually execute a backup schedule to instantly start the backup creation.

To execute a backup schedule:

1. In the navigation panel, go to Backup → Schedule.
2. Click  → Run next to the backup schedule.

Deactivating and Activating a Backup Schedule

You can deactivate a backup schedule to temporarily stop backup creation. Backup schedules are activated by default.

To deactivate or activate a backup schedule:

1. In the navigation panel, go to Backup → Schedule.
2. Click  → Deactivate or Activate next to the backup schedule.

Deleting a Backup Schedule

Important

Deleted backup schedules cannot be restored.

When you delete a backup schedule, the associated backups are not deleted.

To delete a backup schedule:

1. In the navigation panel, go to Backup → Schedule.
2. Click  → Delete next to the backup schedule.
3. Click Delete.

5.28.5. Configuring Backup Storage Parameters for an Instance

1. In the navigation panel, go to Infrastructure → Instances.
2. Click the name of the instance.
3. In the navigation panel, go to Backup → Retention policy.
4. Click  next to the storage.
5. Specify the backup storage parameters of the storage catalog created for the instance:

- Retention redundancy, pcs.: The maximum number of full backups.

For example, if you specify 3, the catalog can contain a maximum of three full backups.

To disable this limitation, specify 0. In this case, the number of backups in the catalog is not limited.

- Retention window, days: The number of days (24 hours) covered by backups.

For example, if you specify 7, the catalog must always contain backups required for restoring the data for the last seven days, including today.

To disable this limitation, specify 0. In this case, backups can be deleted from the catalog at any moment.

- WAL depth: The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR).

For example, if you specify 3, the catalog must always contain at least three backups on each timeline.

To disable this limitation, specify 0. In this case, point-in-time recovery is not available.

- Expired copies: The management policy for expired backups.

Possible values:

- Merge: Merge expired backups with new ones if possible.
- Delete: Delete expired backups from the catalog.

You can enable all checkboxes simultaneously.

Retention redundancy, pcs, Retention window, days, and WAL depth, pcs are applied only if you enable Merge and/or Delete in Expired copies.

The values of Retention redundancy, pcs. and Retention window, days are considered simultaneously when deleting expired backups from the catalog.

For example, if you enter 3 for Retention redundancy, pcs. and 7 for Retention window, days, a maximum of three full backups are saved, as well as all backups required for restoring the data for the past 7 days, including today.

You can also configure storage parameters for a backup when [creating](#) it, as well as for a storage when [creating](#) or [editing](#) it.

The following priority is applied:

- backup parameters are applied first
- instance parameters are applied second
- storage parameters are applied third

For more information about storage parameters, refer to the official Postgres Pro documentation on [pg_probackup](#).

6. Click Save.

5.29. Query Plan Visualization

PPEM supports visualization of query plans that can be retrieved using the `EXPLAIN` command. For more information about this command, refer to the [official Postgres Pro documentation](#).

The visualization contains a tree of query plan nodes and allows you to navigate between them. Each node has a unique name and corresponds to the specified stage of query execution. Clicking on a node name displays expanded information about this node.

This section explains how to manage query plan visualizations. It includes the following instructions:

- [Creating a Query Plan Visualization](#)
- [Viewing a Query Plan Visualization](#)
- [Editing a Query Plan Visualization](#)
- [Deleting a Query Plan Visualization](#)

Creating a Query Plan Visualization

Before performing this instruction:

1. Get the query plan:

```
EXPLAIN (ANALYZE, COSTS, VERBOSE, BUFFERS, FORMAT JSON)
```

2. Copy the query plan and save it.

To create a query plan visualization:

1. In the navigation panel, go to Monitoring → Visualization.
2. Enter parameters of the new query plan visualization (parameters marked with an asterisk are required):

- Name.

If you do not specify the name, it is generated automatically.

- Query plan: The query plan that will be visualized.
- Query: The text of the query whose plan will be visualized.

If you specify query text, it is added to the query plan visualization.

To create a visualization based on a query plan example, in Examples, click  next to the example. In this case, parameter values of the query plan visualization are filled in automatically.

3. Click Visualize.

Viewing a Query Plan Visualization

1. In the navigation panel, go to Monitoring → Visualization.
2. In Saved plans, click the name of the query plan visualization.

Editing a Query Plan Visualization

1. In the navigation panel, go to Monitoring → Visualization.
2. In Saved plans, click  next to the query plan visualization.
3. Edit query plan visualization parameters.
4. Click Visualize.

Deleting a Query Plan Visualization

Important

Deleted query plan visualizations cannot be restored.

To delete a query plan visualization:

1. In the navigation panel, go to Monitoring → Visualization.
2. In Saved plans, click  next to the query plan visualization.

Chapter 6. Upgrade and Migration

It is recommended to upgrade PPEM frequently to ensure access to recent fixes and improvements. You can upgrade to any supported version of any release starting with version 2.0.

Information about PPEM version upgrades can be found in [What's New](#).

6.1. Upgrading to PPEM 2.3

To upgrade PPEM to version 2.3, you must first upgrade the [manager](#), then [agents](#).

Before upgrading, view [backward compatibility features](#).

Considerations of Backward Compatibility in PPEM 2.3

In the PPEM 2.3, packaged API commands can now be sent by the manager. Typically, these are API commands required for executing various operations.

If the manager and agents operate on different PPEM versions, you cannot [create instances](#) and perform any actions with [databases](#) and their objects, such as [schemas](#).

To avoid backward compatibility issues, select the date and time when no operations with databases and their objects are scheduled, and then upgrade both the manager and agents to version 2.3.

Upgrading the Manager to Version 2.3

Note

Before upgrading the manager, wait for all routine tasks run in PPEM to complete (see active tasks in the [task console](#)).

To upgrade the manager:

1. Upgrade the repository and package manager metadata.

The upgrade procedure depends on the version of the used distributive and package manager.

2. Stop the manager 2.2 service:

```
systemctl stop ppm
```

3. Install the manager 2.3.

The installation procedure is performed using the package manager.

4. Upgrade the configuration files in the `/etc` catalog.

Place the parameters into a new configuration file if required.

5. Start the manager 2.3 service:

```
systemctl start ppm
```

6. Ensure that the manager service started successfully and is available:

```
systemctl status ppm
```

Upgrading Agents to Version 2.3

1. Upgrade the repository and package manager metadata.

The upgrade procedure depends on the version of the used distributive and package manager.

2. Stop the agent 2.2 service:

```
systemctl stop ppem-agent
```

3. Install the version 2.3 agent.

The installation procedure is performed using the package manager.

4. Upgrade the configuration files in the `/etc` catalog.

Place the parameters into a new configuration file if required.

5. Start the agent 2.3 service:

```
systemctl start ppem-agent
```

6. Ensure that the agent service started successfully:

```
systemctl status ppem-agent
```

6.2. Upgrading to PPEM 2.2

To upgrade PPEM to version 2.2, you must first upgrade the [manager](#), then [agents](#).

Before upgrading, view [backward compatibility features](#).

Considerations of Backward Compatibility in PPEM 2.2

In the PPEM 2.2, packaged API commands can now be sent by the manager. Typically, these are API commands required for executing various operations.

If the manager and agents operate on different PPEM versions, you cannot [create instances](#) and perform any actions with [databases](#) and their objects, such as [schemas](#).

To avoid backward compatibility issues, select the date and time when no operations with databases and their objects are scheduled, and then upgrade both the manager and agents to version 2.2.

Upgrading the Manager to Version 2.2

Note

Before upgrading the manager, wait for all routine tasks run in PPEM to complete (see active tasks in the [task console](#)).

To upgrade the manager:

1. Upgrade the repository and package manager metadata.

The upgrade procedure depends on the version of the used distributive and package manager.

2. Stop the manager 2.1 service:

```
systemctl stop ppem
```

3. Install the manager 2.2.

The installation procedure is performed using the package manager.

4. Upgrade the configuration files in the `/etc` catalog.

Place the parameters into a new configuration file if required.

5. Start the manager 2.2 service:

```
systemctl start ppem
```

6. Ensure that the manager service started successfully and is available:

```
systemctl status ppem
```

Upgrading Agents to Version 2.2

1. Upgrade the repository and package manager metadata.

The upgrade procedure depends on the version of the used distributive and package manager.

2. Stop the agent 2.1 service:

```
systemctl stop ppem-agent
```

3. Install the version 2.2 agent.

The installation procedure is performed using the package manager.

4. Upgrade the configuration files in the `/etc` catalog.

Place the parameters into a new configuration file if required.

5. Start the agent 2.2 service:

```
systemctl start ppem-agent
```

6. Ensure that the agent service started successfully:

```
systemctl status ppem-agent
```

6.3. Upgrading to PPEM 2.1

To upgrade PPEM to version 2.1, you must first upgrade the [manager](#), then [agents](#).

Upgrading the Manager to Version 2.1

Note

Before upgrading the manager, wait for all routine tasks run in PPEM to complete (see active tasks in the [task console](#)).

To upgrade the manager:

1. Upgrade the repository and package manager metadata.

The upgrade procedure depends on the version of the used distributive and package manager.

2. Stop the manager 2.0 service:

```
systemctl stop ppem
```

3. Install the manager 2.1.

The installation procedure is performed using the package manager.

4. Upgrade the configuration files in the `/etc` catalog.

Place the parameters into a new configuration file if required.

5. Start the manager 2.1 service:

```
systemctl start ppem
```

6. Ensure that the manager service started successfully and is available:

```
systemctl status ppem
```

Upgrading Agents to Version 2.1

1. Upgrade the repository and package manager metadata.

The upgrade procedure depends on the version of the used distributive and package manager.

2. Stop the agent 2.0 service:

```
systemctl stop ppem-agent
```

3. Install the agent 2.1.

The installation procedure is performed using the package manager.

4. Upgrade the configuration files in the `/etc` catalog.

Place the parameters into a new configuration file if required.

5. Start the agent 2.1 service:

```
systemctl start ppem-agent
```

6. Ensure that the agent service started successfully:

```
systemctl status ppem-agent
```

6.4. Upgrading to PPEM 2.0

This section contains [recommendations](#) and an [instruction](#) for migrating to PPEM 2.0.

Important

Version 2.0 is not backward compatible with version 1.0. No tools for smooth data migration are currently available.

Migration Recommendations

When migrating to PPEM 2.0, note the following changes implemented in this version:

- The manager and agent components are rewritten in Golang and implement the new API version. For this reason, some PPEM 2.0 components are incompatible with version 1.0 components:
 - The manager version 2.0 and agents version 1.0 are mutually incompatible.
 - The web application version 2.0 and the manager version 1.0 are mutually incompatible.
 - Configuration files of the manager version 1.0 and the manager version 2.0 are mutually incompatible.
 - Configuration files of agents version 1.0 and agents version 2.0 are mutually incompatible.
- Agents version 2.0 no longer regularly collect metrics and logs of DBMS instances. PPEM now uses pgpro-otel-collector to work with metrics and logs.

Therefore, the following migration recommendations for PPEM 2.0 are provided:

- It is recommended to install PPEM 2.0 on hardware separate from that used for PPEM 1.0, as well as to completely shut down all PPEM 1.0 components (including the manager and agents) before installing and starting the PPEM 2.0 manager and agents.

Important

Performing the same operations on the same DBMS instances simultaneously in two PPEM versions may lead to unpredictable consequences. Avoid using two PPEM versions simultaneously. Simultaneous operation of both versions was not tested and is not guaranteed.

- To use advanced DBMS instance health monitoring features in PPEM, it is recommended to install and configure `pgpro-otel-collector`.

For more information, refer to the official Postgres Pro documentation on [pgpro-otel-collector](#) and [integration with PPEM](#).

Migration

1. Completely shut down all PPEM 1.0 components.
2. On a separate server, [install the manager](#) version 2.0.

The manager 2.0 repository database can be located in the same DBMS instance as the manager 1.0 repository database.

Important

Do not use the same repository database for both versions, otherwise the manager version 2.0 will not start.

3. To verify that the manager version 2.0 is successfully installed, [log in to the web application](#) and get an API key for configuring the agent.
4. [Install an agent](#) version 2.0.
5. On all DBMS instance servers, delete agents first, and then delete the manager version 1.0.

Chapter 7. Troubleshooting

This section explains how to troubleshoot issues:

- In case of issues, [check the state of services](#).
- For more information, [check message logs](#).
- If the log information is insufficient, [change the verbosity of the log](#) and repeat the search.

Checking the State of Services

The [manager](#) and [agent](#) services are controlled by the systemd system manager:

- `ppem` is the manager service.
- `ppem-agent` is the agent service.

To view the state of services, use the `systemctl` utility:

```
# systemctl status ppem
● ppem.service - PostgresPro Enterprise Manager
   Loaded: loaded (/lib/systemd/system/ppem.service; enabled; preset: enabled)
   Active: active (running) since Sat 2024-11-16 15:43:01 MSK; 48min ago
 Main PID: 53582 (ppem-manager)
    Tasks: 7 (limit: 3512)
   Memory: 226.9M
      CPU: 8.445s
   CGroup: /system.slice/ppem.service
           └─53582 /usr/sbin/ppem-manager -config /etc/ppem-manager.yml
```

The status output contains the following:

- `ppem.service` is the name and description of the service.
- The dot indicator (●) uses colors to highlight the service state:
- White is the neutral inactive or deactivating state.
 - Red is the failed or error state.
 - Green is the normal active, reloading or activating state.
- `Loaded` is the service configuration state indicating that the configuration is uploaded to memory and that this is the normal state.
 - `Active` is the service running state, indicating that the service has successfully started and is currently operational.

The start time and running duration are also displayed.

- `Main PID` is the ID and name of the main process in the list of OS processes.
- `Tasks` is the total number of processes and streams generated by the main process.
- `Memory` is usage of memory.
- `CPU` is usage of CPU time.
- `CGroup` is the control group where service processes are placed.
- After the main section, the last service messages will also be displayed.

When operating correctly, the manager and agent services are in the `Active` state.

Checking Message Logs

The manager and agent services can send each other service messages during operation. By default, the manager and agent use the standard output (`stdout`) for sending service logs and messages. Messages are captured by the systemd tools and can be viewed using `journalctl`:

```
# journalctl -u ppem
...
```

You can use the `-f` option to output incoming messages:

```
# journalctl -fu ppem
...
```

Configuring Verbosity for the Log

Logging can have different verbosity:

- `error` is logging only errors.
- `warning` is logging errors and warnings.
- `info` is logging errors, warnings, and information messages.

This level is used by default.

- `debug` is logging errors, warnings, information messages, and debug messages.

You can configure logging in the `ppem-manager.yml` manager and `ppem-agent.yml` agent configuration files using the `log.level` configuration parameter.

Logging changes take effect after the service restart.

Chapter 8. Terms and Abbreviations

Key Terms

Web Application

Browser-based graphical user interface designed to manage and monitor DBMS infrastructure and resources.

Manager

Coordinates agent operation and serves requests from PPEM users.

Agent

Receives manager instructions and manages DBMS instances.

Resources and Objects

DBMS instance

Postgres Pro DBMS operating on a server or virtual machine.

Instance objects

Separate objects in the DBMS, such as tablespaces, databases, schemas, tables, and indexes.

Instance parameters

DBMS instance parameters, which are usually specified in the `postgresql.conf` configuration file.

Instance service

Method of starting a DBMS instance. It is recommended to use `systemd`.

Data source

Connectable source for storing resources of a certain type. PPEM classifies data sources by their deployment type (e.g., internal and external) and by the type of stored resources (e.g., metrics, logs, and backups).

Access Management

Privilege

Right to perform an action with a resource in PPEM.

Role

Privilege group in PPEM.

User

Account associated with a user in PPEM.

Group

User group in PPEM.