

Interactive session
It's time to go hands-on!



Hand-on documentation



Download this presentation

<https://tinyurl.com/mukp4nux>



Features shown during the demo

- Installation operator and CNPG plugin
- Check the CloudNativePG operator status
- Installation Backup (Barman) Operator
- Setup PostgreSQL Cluster on Kubernetes
- Create the backup of the PostgreSQL Cluster
- Run out-of-the-place recovery
- Run switchover
- Simulate failover szenario
- Run minor upgrade using rolling upgrade approach
- Scale out/down
- Test fencing
- Test hibernation
- Database migration
- Major upgrade (in-place)
- Monitoring

Deployment

Administration

Backup and
Recovery

Fencing

High Availability

Monitoring

Patching

Hibernation

Last CloudNativePG tested version is 1.26



This demo is in 

<https://github.com/EnterpriseDB/cnpg-hands-on>

<https://tinyurl.com/4dnhp98j>



Setup the Environment



Prerequisites

- Install kubectl, helm, git, docker, k3d

- Download the github repository:

```
git clone https://github.com/EnterpriseDB/cnpg-hands-on
```

- Change the directory to cnpg-hands-on:

```
cd cnpg-hands-on
```



Create k3d cluster

- In the terminal 1:
 - Create the k3d cluster - run the script:
`./00_start_infra.sh`
 - Check the cluster:
`kubectl get nodes`
`kubectl get pods`



Create k3d cluster

- If you are using other OS as Mac or Linux, or if you are using other k8s distribution:
 - Start your k8s cluster
 - Install Prometheus and Grafana:

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

```
helm upgrade --install -f
```

```
https://raw.githubusercontent.com/cloudnative-pg/cloudnative-pg/main/docs/src/samples/monitoring/kube-stack-config.yaml prometheus-community prometheus-community/kube-prometheus-stack
```



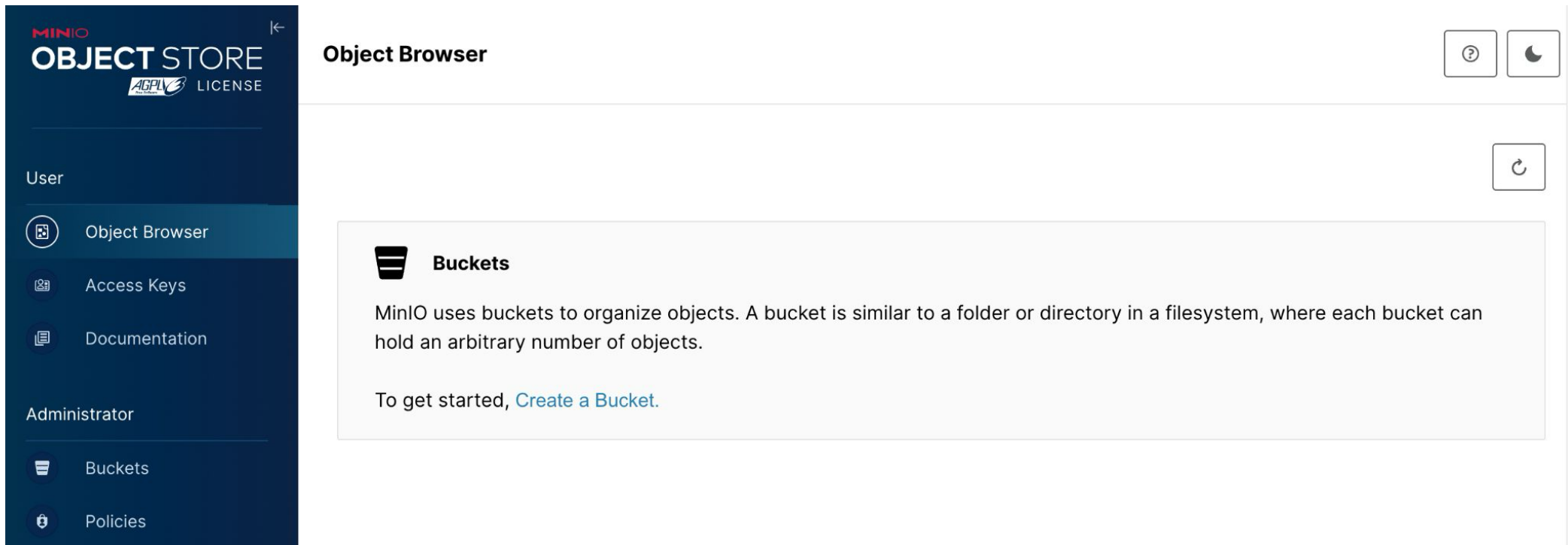
Run Minio server

- In the terminal 1:
 - Start MinIO:
`./start_minio_docker_server.sh &`



Call to action: Connect to the MinIO server

- In the browser open the new tab and go to
 - `http://localhost:9001`
 - Connect as user **admin** with the password: **password**
- The page will appear:



Use case Plug-in installation



Install CNPG plugin

- In the terminal run the script 01_install_plugin.sh:
`./01_install_plugin.sh`
- Call the help for the CNPG Plugin, run:
`kubectl-cnp help`



Use case

Operator installation



Install the CNPG Operator and check the it in the terminal

- In the terminal install the operator:

```
./02_install_operator.sh
```

In the terminal check the installation of the operator:

```
./03_check_operator_installed.sh
```



Use case

Backup Configuration



Barman Cloud Operator configuration

- Installing cert-manager
- Installing Barman Cloud operator
- Configuration the storage in MinIO
- In the terminal 1:
 - Install Barman Operator:
`./04_install_barman_plugin.sh`
 - Check the installation of the barman operator:
`./05_check_barman_plugin.sh`



Use case

Create the postgres cluster



Configure and Install the Postgres cluster

- In the terminal 1:
 - Create the yaml file:
`./06_get_cluster_config_file.sh`
 - Create the postgres cluster:
`./07_install_cluster.sh`
- In the terminal **2** - Check the Postgres cluster status:
`./08_show_status.sh`



Create table test with 1000 rows

- Once cluster is running ... (minimum the primary) insert the data - run the script:

```
cd /home/workshop/workshop/cnp-demo
```

```
./09_insert_data.sh
```



Use case Backup & Restore



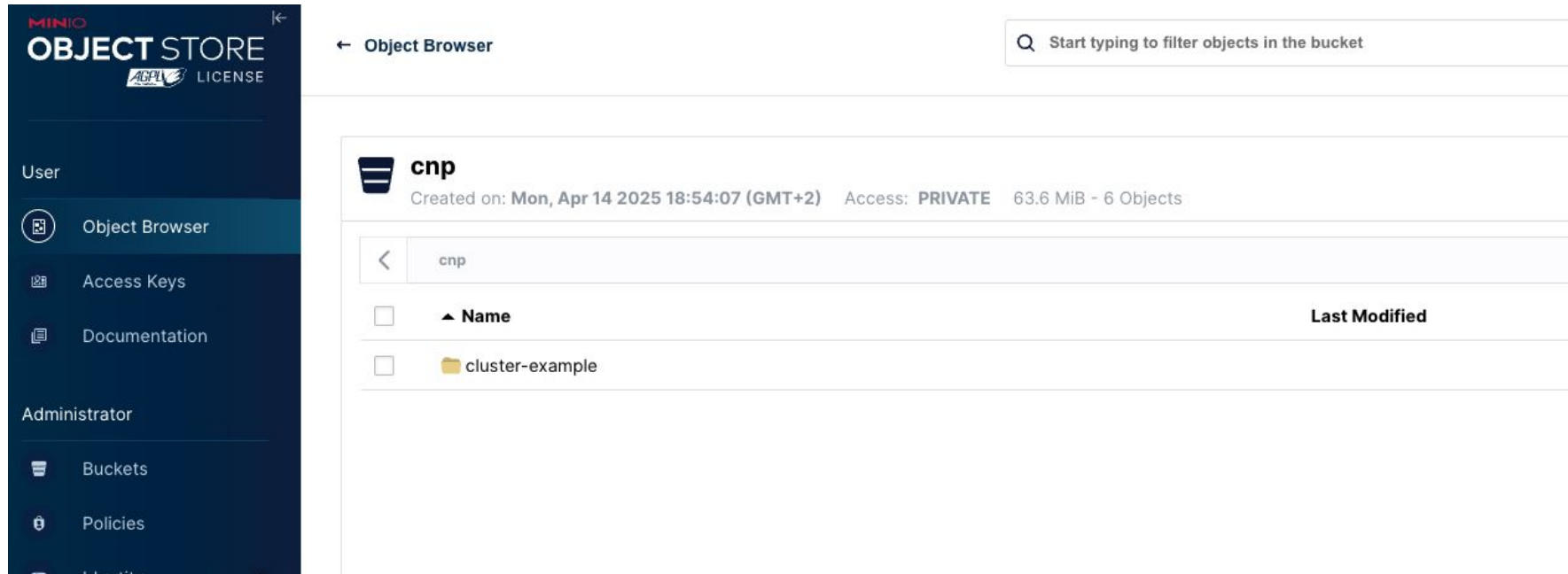
Backup demonstration

- Create the full backup
- Check Backup in MinIO UI
- Restore the database from the backup



Connect to the MinIO server

- In the browser open the new tab and go to
 - `http://localhost:9001`
 - Connect as user **admin** with the password: **password**
- The page will appear:



- Click on cluster-example and check the backup of WAL files



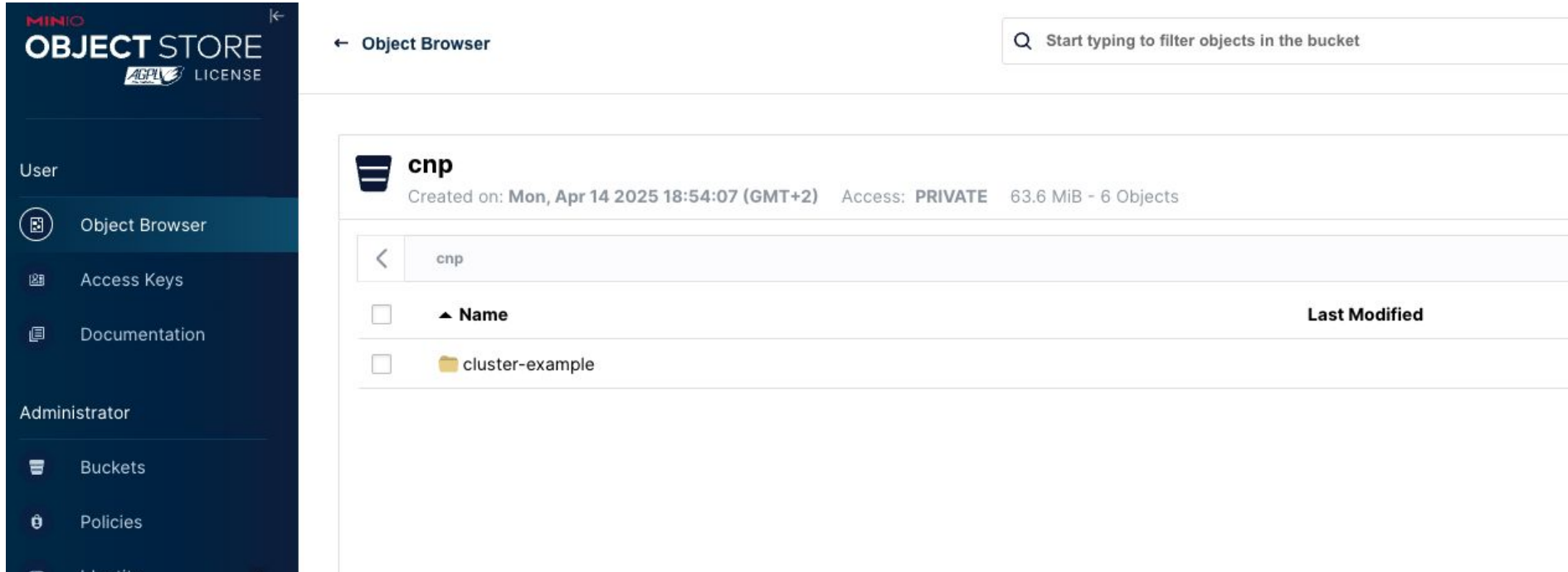
Create the full backup

- With this step we will:
 - Create the full backup of the postgres cluster in the MinIO storage:
- In the terminal 1:
 - Run the script:
`./10_backup_cluster.sh`
 - Check the backup status:
`./11_backup_describe.sh`
 - Check the created backup in the MinIO GUI



Call to action: Check Backup in MinIO UI

- Check the created backup in the MinIO GUI



The screenshot displays the MinIO Object Store interface. On the left is a dark sidebar with the 'MINIO OBJECT STORE' logo and a list of navigation items: 'Object Browser' (selected), 'Access Keys', 'Documentation', 'Buckets', and 'Policies'. The main area shows the 'Object Browser' for a bucket named 'cnp'. It includes a search bar at the top right with the placeholder text 'Start typing to filter objects in the bucket'. Below the bucket name, it states 'Created on: Mon, Apr 14 2025 18:54:07 (GMT+2)', 'Access: PRIVATE', and '63.6 MiB - 6 Objects'. A table lists the objects in the bucket:

<input type="checkbox"/>	Name	Last Modified
<input type="checkbox"/>	cluster-example	



Restore the database from the backup

- With this step we will:
 - Create the new cluster cluster-restore
 - Restore the full backup created in the previous step in the new cluster:
- In the terminal 1:
 - Run the restore:
`./12_restore_cluster.sh`
 - Check the creation status:
`kubectl get pods -w` # after creation stop the execution with <ctrl>+c
 - Check the table test in the cluster-restore, run the script:
`./13_check_restore.sh`
 - Delete the cluster-restore to avoid resource problems during the workshop:
`kubectl delete cluster cluster-restore`



Use case Promote & Failover



Promote standby to the primary

- With this step we will:
 - promote a pod in the cluster to primary, so you can start with maintenance work or test a switch-over situation in your cluster
 - Promote is the option of the cnpg plugin
- In the terminal 1:
 - Run the script:
`./14_promote.sh`
- In the terminal **2**:
 - Check the failover cluster status:
`./08_show_status.sh`



Run failover test

- With this step we will:
 - Delete the primary database of the cluster cluster-example
 - The operator will:
 - detect failure of the primary
 - promote standby to the new primary and create the new standby according to the configuration (3 instances)
- In the terminal 1:
 - Run the script:
`./15_failover.sh`
- In the terminal **2**:
 - Check the failover cluster status:
`./08_show_status.sh`



Use case

Minor Upgrade



Run the Promote and Upgrade

- With this step we will:
 - Run the postgres minor update from the version 16.4 to 16.5
- In the web terminal **2** (prepare a terminal for status - and one to run the admin-commands):
 - Check the upgrade status:
`./08_show_status.sh`
- In the terminal **1**:
 - Run the script:
`./16_minor_upgrade.sh`
- In terminal **2**: (prepare a terminal for status - and one to run the admin-commands):
 - check Postgres version: PostgreSQL Image: quay.io/enterprisedb/postgresql:16.5



Use case

Scale-out and scale-down



Scale-out the postgres cluster

- With this step we will:
 - Add the 1 standby to the cluster
- In the web terminal 1:
 - Run the script:
./18_scale_out.sh (using `-replicas=X...` another way would be to update the YAML)
- In the web terminal **2**:
 - Check the cluster status:
./08_show_status.sh



Scale-down the postgres cluster

- With this step we will:
 - Remove 2 standby pods from the cluster
- In the web terminal 1:
 - Run the script:
`./19_scale_down.sh`
- In the web terminal **2**:
 - Check the cluster status:
`./08_show_status.sh`



Use Case Fencing



Stop postgres process on the pod

- In the web terminal 1:
 - Run the script:
`./20_fencing.sh` on
- In the web terminal **2**:
 - Check the cluster status:
`./08_show_status.sh`



Start the postgres process on the pod

- In the terminal 1:
 - Run the script:
`./20_fencing.sh off`
- In the terminal **2**:
 - Check the cluster status:
`./08_show_status.sh`



Use case Hibernation



Stop the postgres cluster

- In the terminal 1:
 - Run the script:
`./21_hibernation.sh` on
- In the terminal **2**:
 - Check the cluster status:
`./08_show_status.sh`



Start the postgres cluster

- In the terminal 1:
 - Run the script:
`./21_hibernation.sh off`
- In the terminal **2**:
 - Check the cluster status:
`./08_show_status.sh`



Use case Database Migration

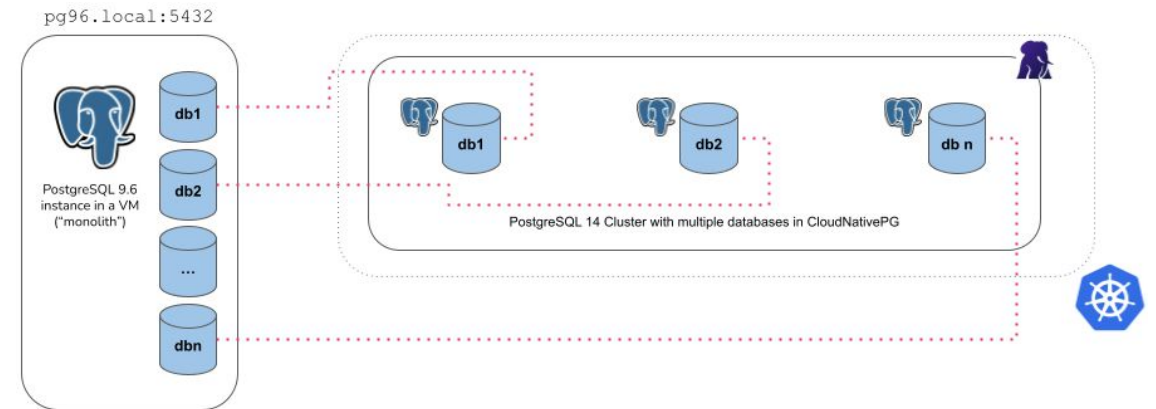
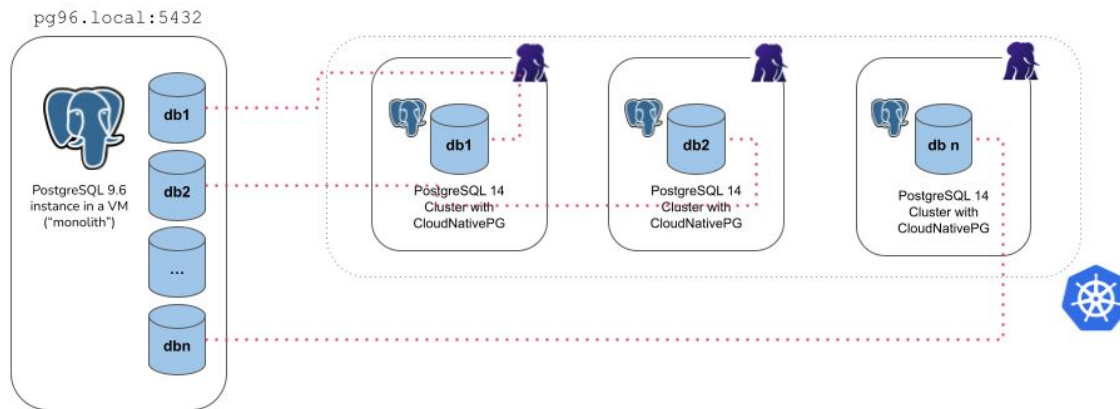


Database Migration

- In this step we will migrate the app database from our existing cluster to the new cluster
- We will create the yaml file with the setting “import” in the bootstrap section
- The operator uses internally postgres tools pg_dump and pg_restore
- This method can be used to **migrate** another database or to run **out-of-the-place upgrade**
- Possible settings:

Microservices:

Monolith:



Migrate the cluster example to the new cluster

- In the terminal 1:
 - Create the new cluster v17 and migrate the database app from the cluster cluster-example::
`./22_major_upgrade_out_of_the_place.sh`
 - Connect to the migrated cluster and check version and data::
`./23_verify_data_migrated_16_17.sh`



Use case

Major Upgrade



Database Major Upgrade

- CNPG supports 3 types of major upgrade:
 - **In-place**: you should test carefully this method before you migrate your production database. You should create the fallback scenario in case of failure
 - **Out-of-the-place**: we discussed this topic in “Database migration”
 - **Logical replication**: you can create the new database in the new cluster and replicate the data between two clusters.
 - CloudNativePG enhances this capability by providing declarative support for key PostgreSQL logical replication objects:
 - Publications via the Publication resource
 - Subscriptions via the Subscription resource



In-place major upgrade

- In this step we will:
 - upgrade the existing cluster cluster-example from the Postgres version 16 to 17
 - choose in-place method

Note: For this action the downtime should be planned: during the upgrade process postgres instances are not available!

- In the terminal 1:
 - Run in-place major upgrade for the cluster cluster-example:
`./24_major_upgrade_in_place.sh`
 - Connect to the upgraded cluster and check version:
`./25_verify_major_upgrade_16_17.sh`



Use case Monitoring



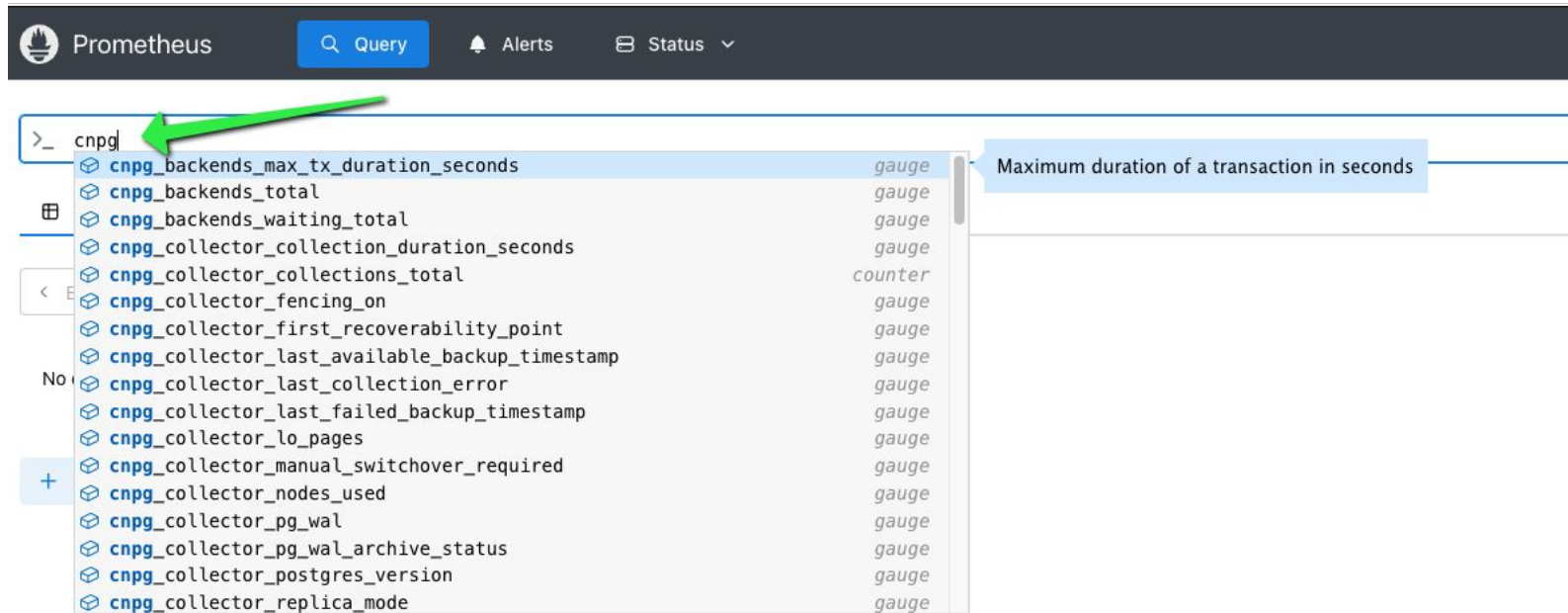
Setup monitoring

- In the terminal 1:
 - Change the directory::
`cd <your github dir>/cnp-demo/monitoring`
 - Install the prometheus rules:
`./01_prometheus_rules.sh`
 - Start port forwarding for prometheus and grafana:
`./02_port_forwarding_prometheus_grafana.sh`
- Download the Grafana Dashboard to your laptop:
 - <https://github.com/cloudnative-pg/grafana-dashboards/blob/main/charts/cluster/grafana-dashboard.json>



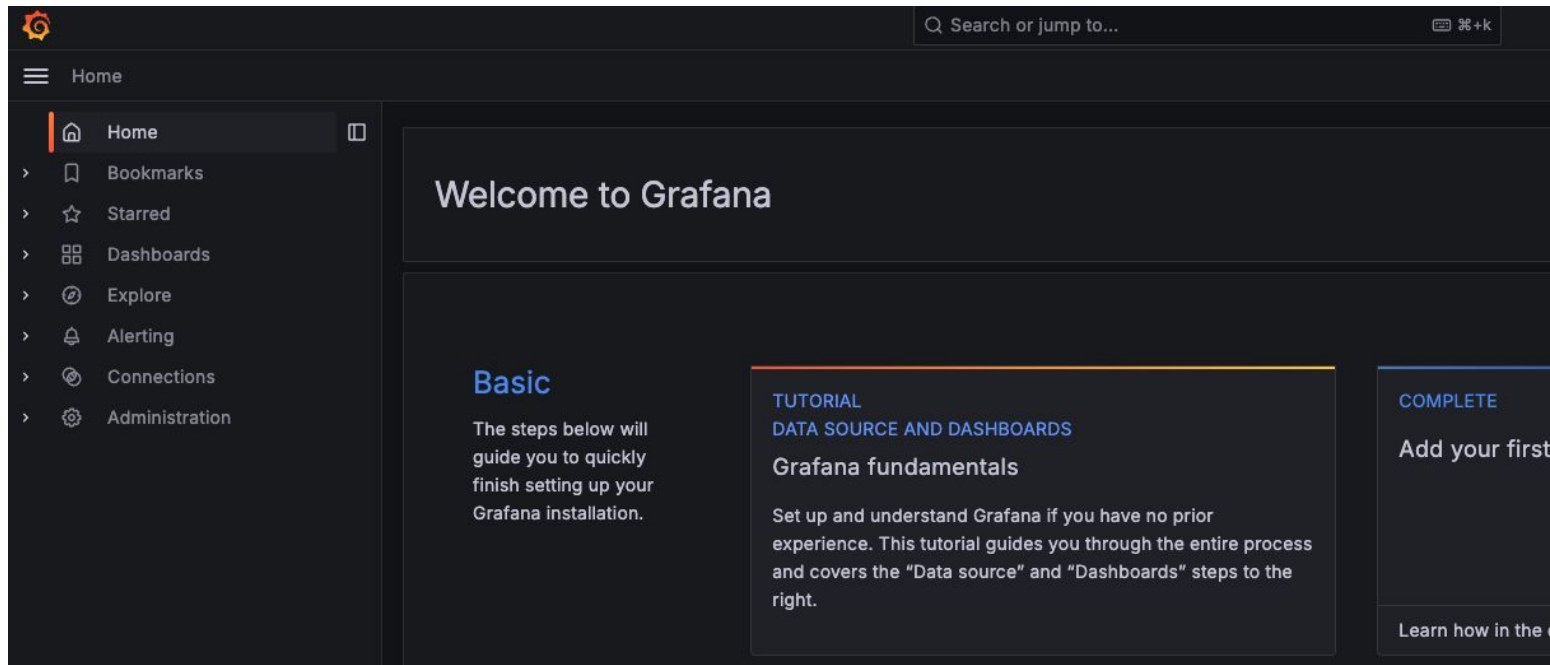
Explore Prometheus

- In the browser open the new tab and go to
 - <http://localhost:9090>
- The prometheus page will appear - search for “cnpg” metrics:



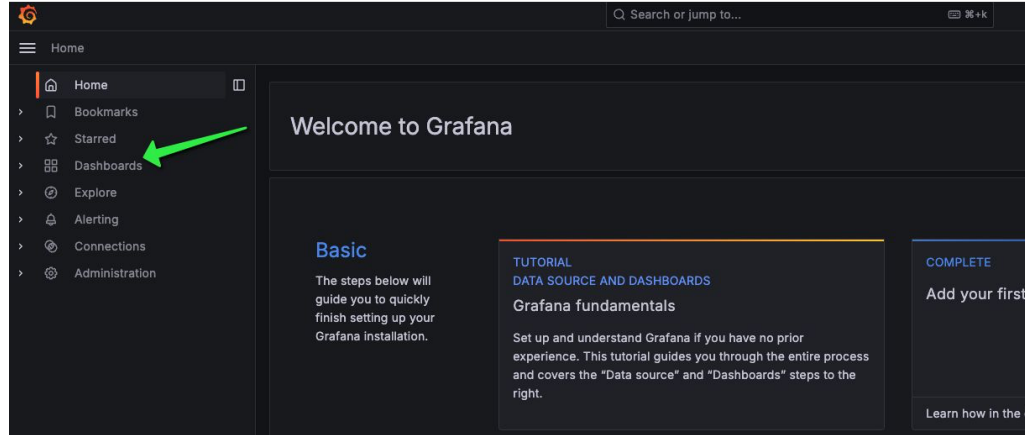
Access the Grafana page

- In the browser open the new tab and go to
 - <http://localhost:3000>
 - Connect as user **admin** with the password: **prom-operator**
- The grafana page will appear

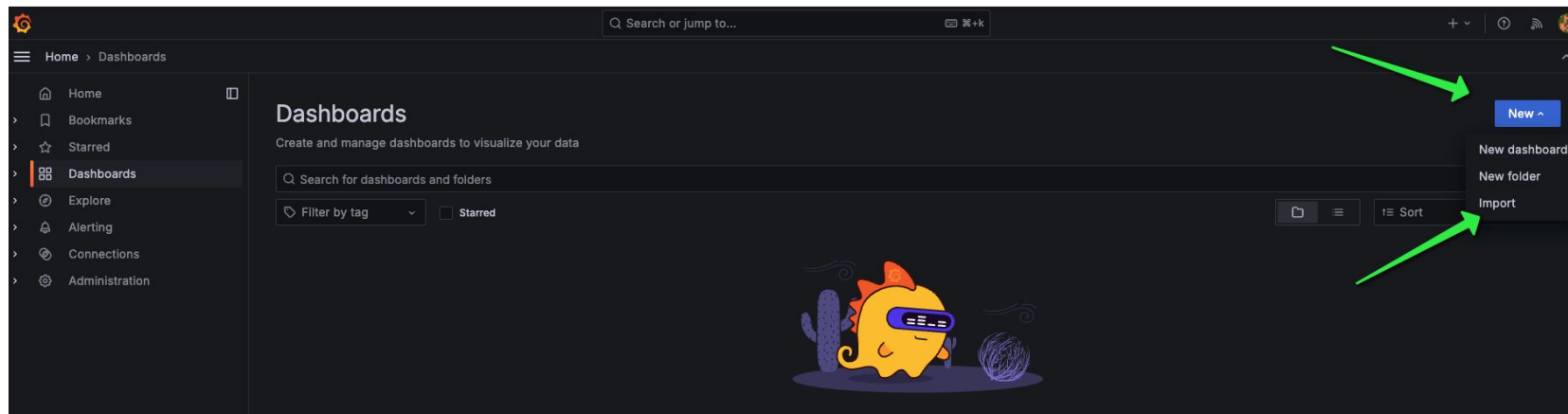


Configure Grafana

- Go to Dashboards

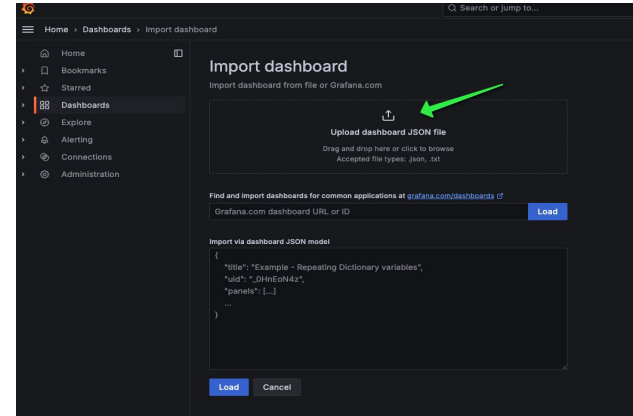


- Press “New”, then “Import”:

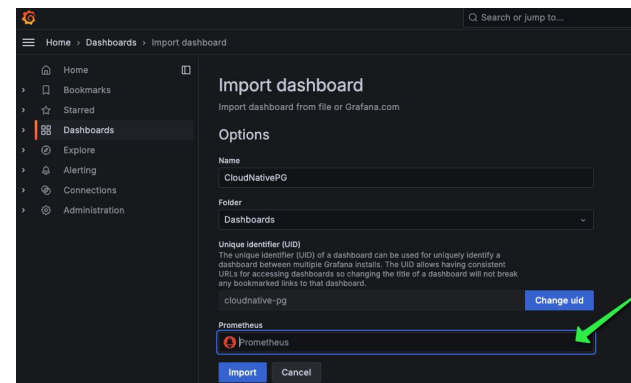


Configure Grafana - continued

- Upload the Dashboard json file:



- Upload grafana-dashboard.json file
- Select Prometheus as the data source:



Explore CNPG Dashboard- continued

- Explore the CNPG Dashboard:

