

Hands-on



Features shown during the demo

- Kubernetes plugin install
- CloudNativePG operator install
- Postgres cluster install
- Insert data in the cluster
- Failover
- Backup
- Recovery
- Scale out/down
- Fencing
- Hibernation
- Monitoring
- Rolling updates (minor and major)

Deployment

Administration

Backup and
Recovery

High Availability

Monitoring

Last CloudNativePG tested version is 1.25



This demo is in



<https://github.com/sergioenterprisedb/kubecon2022-demo>



VMs IP Addresses

vm	vm ip	public ip
vm1		
vm2		
vm3		
vm4		
vm5		
vm6		
vm7		
vm8		
vm9		

vm	vm ip	public ip
vm10		
vm11		
vm12		
vm13		
vm14		
vm15		
vm16		
vm17		
vm18		



Setup the demo env



Step 1: Connect to your demo environment

- Connect to the terminal 1:
 - Run in the browser `https://<your vm ip address>`
 - Connect to the terminal as user **workshop** with the password **workshop**
- Connect to the terminal 2:
 - Open the new browser tab.
 - Run `https://<your vm ip address>`
 - Connect to the terminal as user **workshop** with the password **workshop**



Step 2: Create k3d cluster

- In the terminal 1:
 - Go to the directory `/home/workshop/workshop/cnp_demo`:
`cd /home/workshop/workshop/cnp_demo`
 - Create the k3d cluster - run the script:
`./00_start_infra.sh`
 - Check the cluster:
`kubectl get nodes`
`kubectl get pods`



Step 3: Run Minio server

- In the terminal 1:
 - Go to the directory `/home/workshop/workshop/cnp_demo`:
`cd /home/workshop/workshop/cnp_demo`
 - Start MinIO:
`./start_minio_docker_server.sh &`



Install the operator



Step 4: Install CNPG Operator and CNPG plugin

- In the terminal 1:
 - Install CNPG plugin:
`./01_install_plugin.sh`
 - Install CNPG operator:
`./02_install_operator.sh`
 - Check the installation of the operator (try several times):
`./03_check_operator_installed.sh`



Create the postgres cluster



Step 5: Install Postgres cluster

- In the terminal 1:
 - Create the yaml file:
`./04_get_cluster_config_file.sh`
 - Create the postgres cluster:
`./05_install_cluster.sh`
- In the terminal **2**:
 - Go to the directory `/home/workshop/workshop/cnp_demo`:
`cd /home/workshop/workshop/cnp_demo`
 - Check the Postgres cluster status:
`./06_show_status.sh`



Step 6: Create table test with 1000 rows

- In the terminal 1:
 - Run the script:
`./07_insert_data.sh`
 - Check the data in the table test:
`./check_table_test.sh`

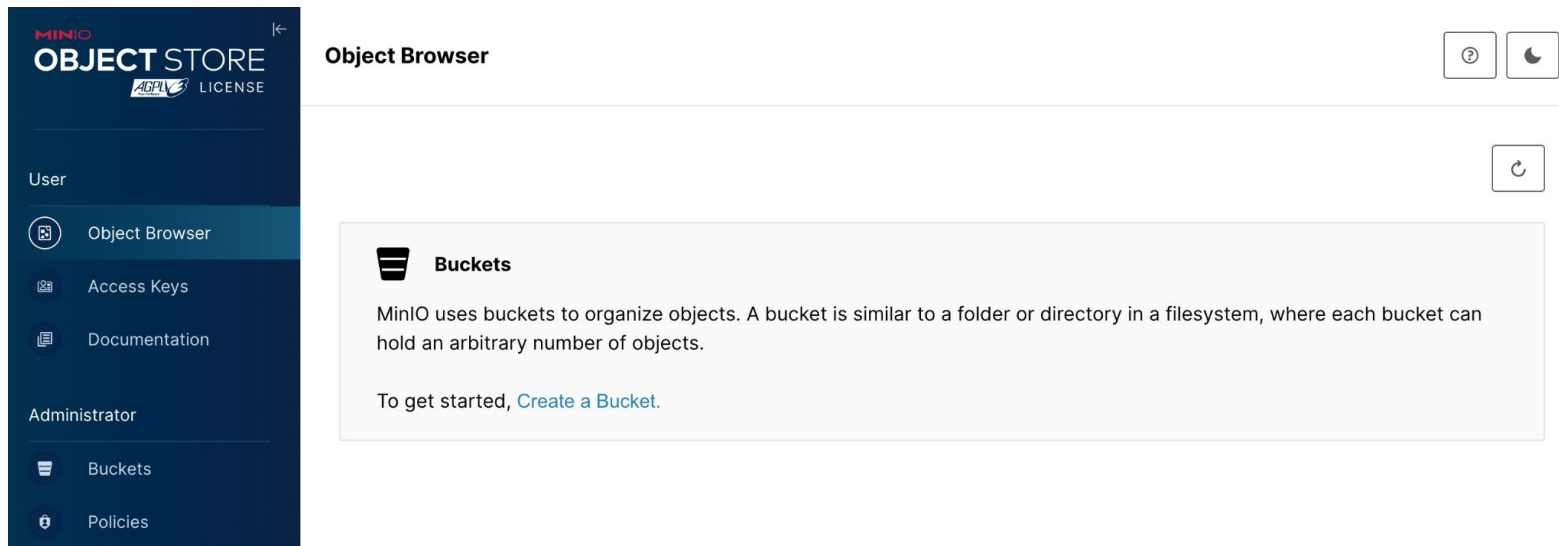


Upgrade the postgres cluster



Step 7: Connect to the MinIO server

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



Step 8: Check the cluster status

- In the terminal 1:
 - Run the command
`kubectl -cnpg status cluster-example`
 - In the output
 - check Postgres version: "PostgreSQL Image: ghcr.io/cloudnative-pg/postgresql:**16.1**"
 - check "Continuous Backup status": "**Not configured**"



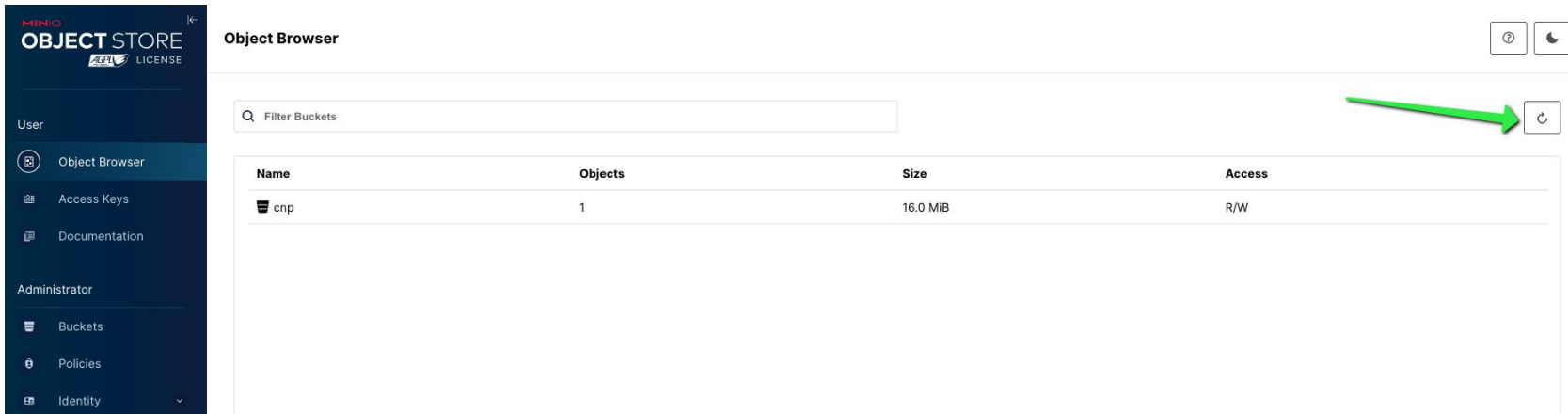
Step 9: Run the script 09_upgrade.sh

- With this step we will:
 - Run the postgres minor update from the version 16.1 to 16.4
 - We will configure the WAL files backup to the MinIO storage
- In the terminal 1:
 - Run the script:
`./09_upgrade.sh`
- In the terminal **2**:
 - Check the upgrade status:
`./06_show_status.sh`



Step 10: Check the WAL backup on the MinIO server:

- In the browser tab - MinIO server
 - Press “refresh” button:



The screenshot displays the MinIO Object Browser interface. On the left is a dark sidebar with the 'MINIO OBJECT STORE' logo and a navigation menu. The main area is titled 'Object Browser' and features a search bar labeled 'Filter Buckets'. Below the search bar is a table with the following data:

Name	Objects	Size	Access
cnp	1	16.0 MiB	R/W

In the top right corner of the main area, there are two icons: a help icon and a refresh icon. A green arrow points to the refresh icon, which is a circular arrow symbol.



Backup & Restore



Step 11: 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 backup in the MinIO GUI



Step 12: 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 script:
`./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:
`./check_restore_table_test.sh`



Failover



Step 13: Run failover test

- With this step we will:
 - Delete the primary database of the cluster cluster-example
 - Check the cluster status in the another terminal window
- In the terminal 1:
 - Run the script:
`./13_failover.sh`
- In the terminal **2**:
 - Check the failover cluster status:
`./06_show_status.sh`



Scale-out and scale-down



Step 14: Scale-out the postgres cluster

- With this step we will:
 - Add the 1 standby to the cluster
- In the terminal 1:
 - Run the script:
`./14_scale_out.sh`
- In the terminal **2**:
 - Check the cluster status:
`./06_show_status.sh`



Step 15: Scale-down the postgres cluster

- With this step we will:
 - Remove 2 standby pods from the cluster
- In the terminal 1:
 - Run the script:
`./15_scale_down.sh`
- In the terminal **2**:
 - Check the cluster status:
`./06_show_status.sh`



Fencing



Step 16: Stop postgres process on the pod

- In the terminal 1:
 - Run the script:
`./30_fencing_on.sh`
- In the terminal **2**:
 - Check the cluster status:
`./06_show_status.sh`



Step 17: Start the postgres process on the pod

- In the terminal 1:
 - Run the script:
`./31_fencing_off.sh`
- In the terminal **2**:
 - Check the cluster status:
`./06_show_status.sh`



Hibernation



Step 18: Stop the postgres cluster

- In the terminal 1:
 - Run the script:
`./32_hibernation_on.sh`
- In the terminal **2**:
 - Check the cluster status:
`./06_show_status.sh`



Step 19: Start the postgres cluster

- In the terminal 1:
 - Run the script:
`./33_hibernation_off.sh`
- In the terminal **2**:
 - Check the cluster status:
`./06_show_status.sh`



Major Upgrade



Step 20: Create the Postgres 16 Cluster

- In the terminal 1:
 - Change directory to /home/workshop/workshop/cnp-demo/major_upgrade_demo
`cd /home/workshop/workshop/cnp-demo/major_upgrade_demo`
 - Create the cluster v16:
`./04_create_cluster_v16.sh`
 - Check the cluster status:
`./05_show_status_v16.sh`
 - Insert the data:
`./06_insert_data_cluster_v16.sh`
 - Verify the inserted data:
`./07_verify_data_inserted.sh`



Step 21: Create the Postgres 17 Cluster and import data from PG 16

- Create the cluster postgres v17 and import the data from the postgres v16:
`./08_upgrade_v16_to_v17.sh`
- Check the cluster status:
`./09_show_status_v17.sh`
- Verify the data in the postgres v17:
`./10_verify_data_migrated_16_17.sh`



Monitoring



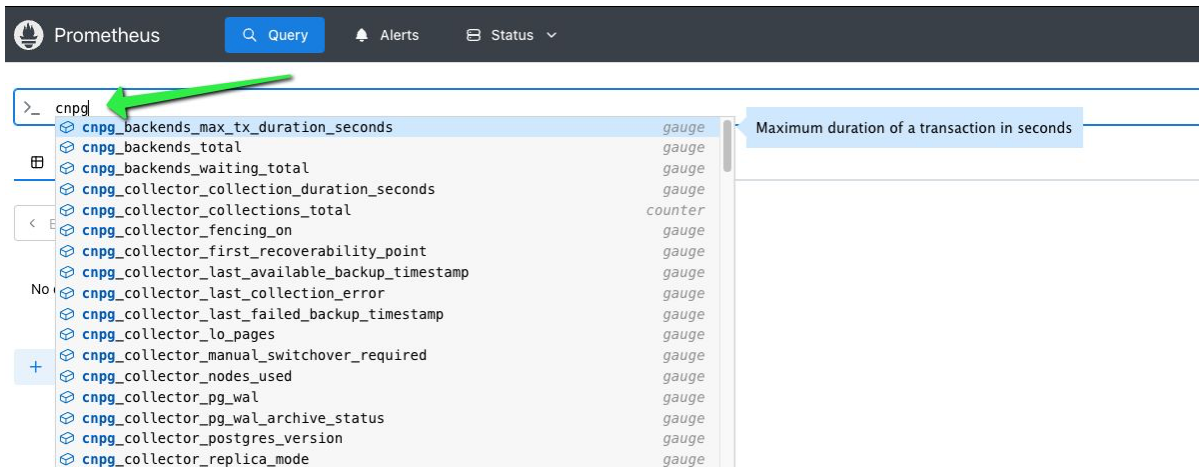
Setup monitoring

- In the terminal 1:
 - Run the command:
`kubectl get pods`
 - Change the directory::
`cd /home/workshop/workshop/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-dashboards/grafana-dashboard.json>



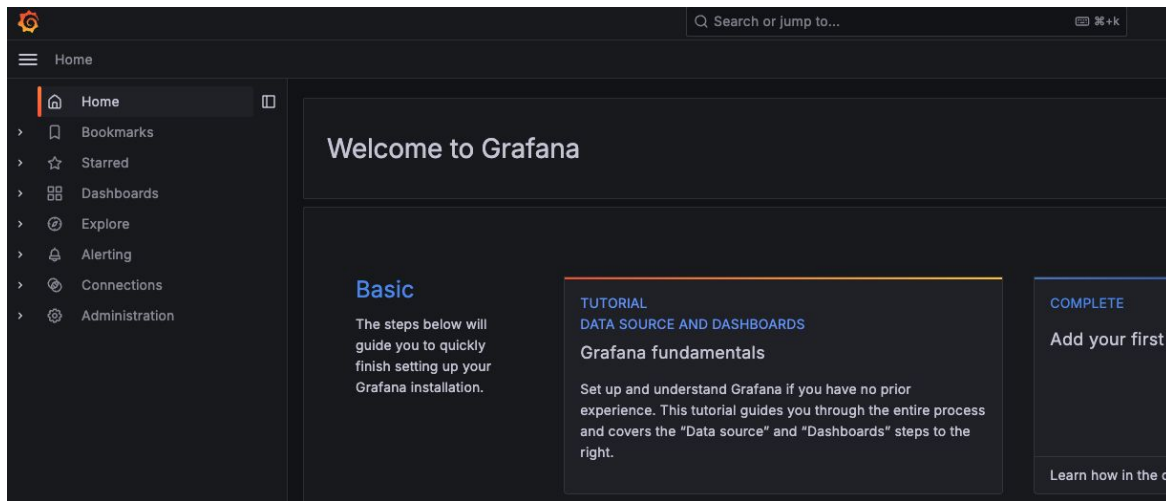
Explore Prometheus

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



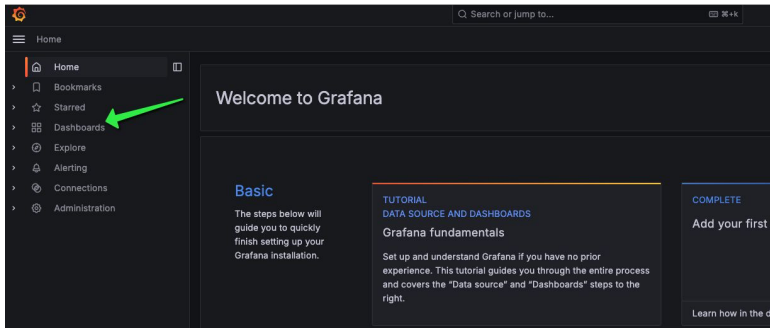
Access the Grafana page

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

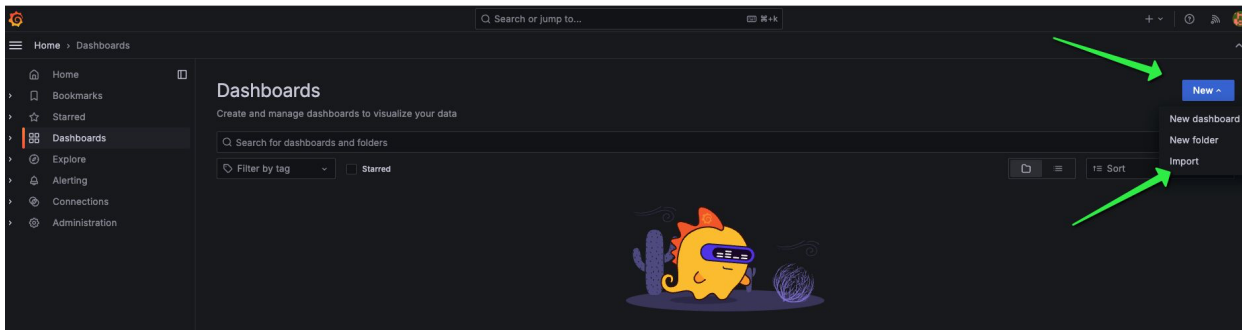


Configure Grafana

- Go to Dashboards

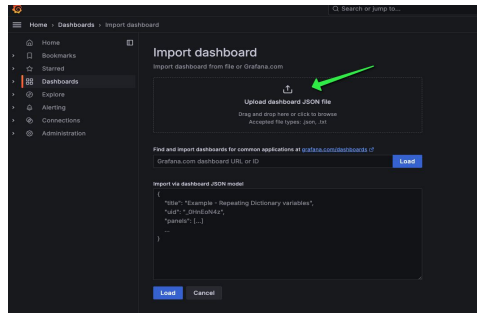


- Press “New”, then “Import”:

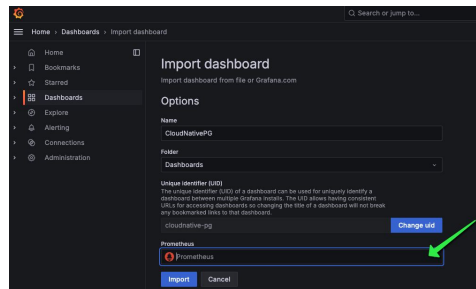


Configure Grafana - continued

- Upload the Dashboard json file:



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



Explore CNPG Dashboard- continued

- Explore the CNPG Dashboard:





EDB

Postgres® for the AI Generation

THANK YOU

