EDB
Postgres® for the AI Generation

CloudNativePG Training

Hands-on

Section A

First steps

Exercise #A1 - Playground setup

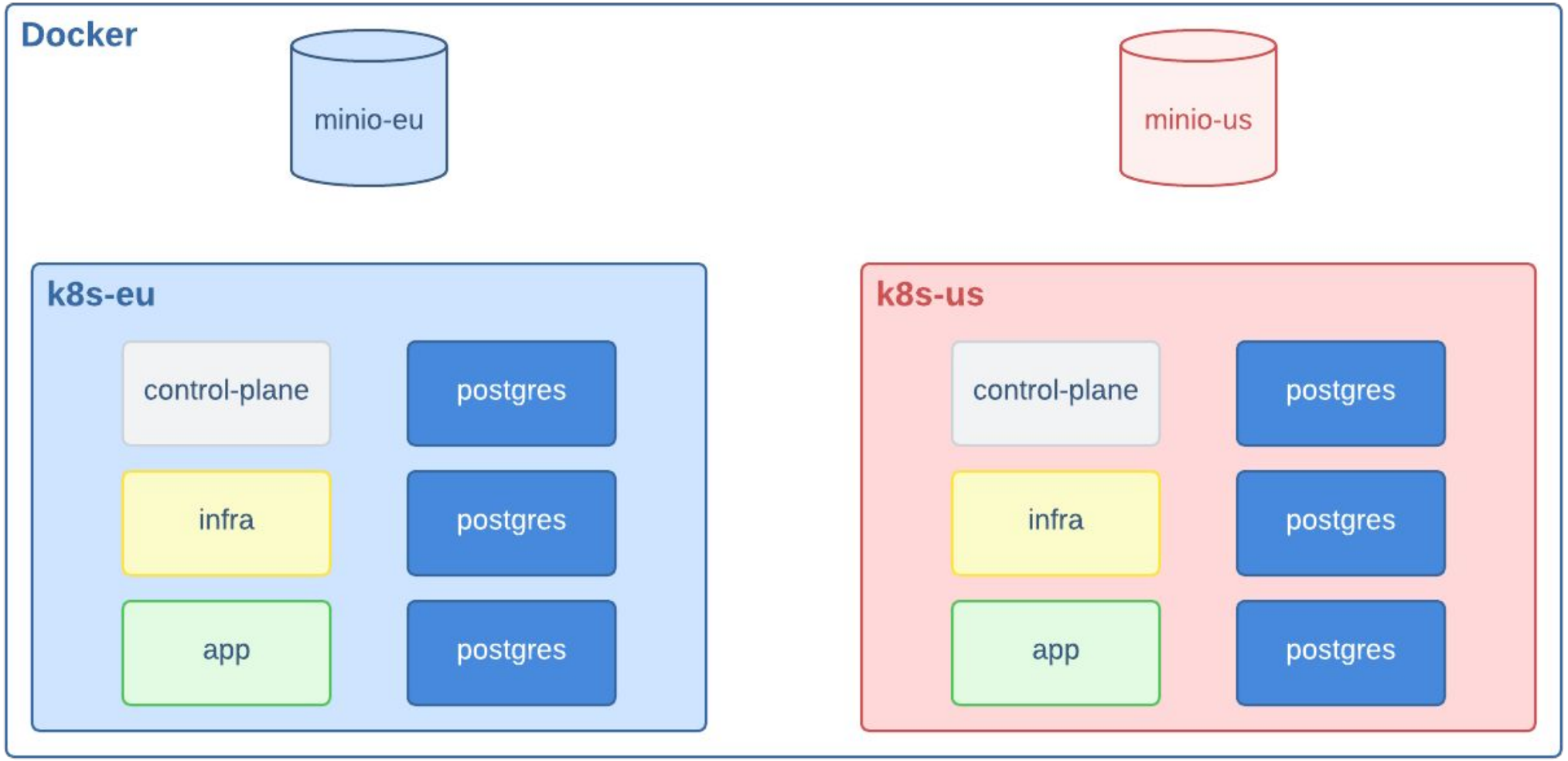
To complete this exercise, you'll need to install CNPG Playground on your computer. This setup is essential for hands-on exercises throughout the session.

Follow the installation steps provided on the **CNPG Playground** page.

Key Tasks:

- Verify access to both the EU and US contexts by running **`kubectl config use-context <context-name>`**
- List the nodes in your Kubernetes cluster with **`kubectl get nodes`**
- Ensure you have the latest CNPG plugin installed by checking its version with **`kubectl cnpg version`**

Local Environment Overview



Exercise #A2 - CloudNativePG 1.26.0-rc3 installation

To continue with the setup, you need to **deploy the CNPG operator**. Familiarise yourself with the documentation.

Key Tasks:

- Install the operator in the EU cluster with:
  ```
  kubectl apply --server-side -f \
  https://raw.githubusercontent.com/cloudnative-pg/cloudnative-pg/main/releases/cnpg-1.26.0-rc3.yaml
  ```
- Verify it runs on the control plane with:
  ```
  kubectl get pods -o wide -n cnpg-system
  ```
- Examine the **affinity** section of the **cnpg-controller-manager** deployment
- Repeat for the **US** cluster

Exercise #B1 - Bootstrap a new cluster

Open the "Quickstart" page from the documentation, download the `**cluster-example.yaml**` file from the website and save it locally.

For example, you can use `` `curl -LO <URL>` ``.

Key Tasks:

- Inspect the **cluster.yaml** file content
- Deploy the cluster by applying the manifest:
  `kubectl apply -f cluster-example.yaml`
- Use the CloudNativePG plugin to view the status of your cluster:
  `kubectl cnpg status cluster-example`
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Exercise #B2 - Convention over configuration

Visit the **CloudNativePG API Reference** in the documentation, focusing on the Cluster resource. Explore the default conventions used in CloudNativePG by comparing the configuration manifest with the actual cluster resource specifications.

Key Tasks:

- Retrieve the specifications of the deployed cluster
  `kubectl get -o yaml cluster cluster-example`
- Compare them with the initial manifest in **cluster-example.yaml**
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Exercise #B3 - Explore the generated resources

CloudNativePG generates several Kubernetes resources to manage PostgreSQL clusters, such as **PVCs**, **pods**, **services**, **configmaps**, **secrets**, **clusters**, and **PDBs**.

Key Tasks:

- List all resources of a specific type with "**`kubectl get KIND`**". For example:
  `kubectl get pvc`

- Get details on a specific object with "**`kubectl get KIND NAME`**". For example:
  `kubectl get pvc cluster-example-1`
  `kubectl get pvc -o yaml cluster-example-1`
  `kubectl describe pvc cluster-example-1`

- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Exercise #B4 - Editing a cluster

Although resources should normally be changed via a source control manager like Git (for GitOps), sometimes you might be asked to change a resource live, through the "**kubectl edit**" command.

Key Tasks:

- Familiarise with the "**`kubectl edit cluster`**" command. For example: **`kubectl edit cluster cluster-example`**
- Set "**shared_buffers**" to "*whatever*". What happens? Why?
- Set "**log_destination**" from "**csvlog**" to "**stderr**". What happens? Why?
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Recommendation

While it's useful to be aware of the “kubectl edit” command, the recommended approach is to always modify the cluster's manifest file directly. After making changes, apply the updated manifest using the following command:

```
kubectl apply -f cluster.yaml
```

Please use this method from now on whenever you're asked to edit a resource.

Exercise #B5 - Connections from inside a pod

While applications typically connect to your PostgreSQL database over the network, sometimes it can be useful to directly access your PostgreSQL cluster by opening a shell to a running container—whether it's a primary or a replica—and using the "**psql**" command-line tool.

Key Tasks:

- Familiarise with the "**`kubectl exec -ti`**" command.
- Open a shell and start exploring as if it was a system container. Comments?
  ```
  kubectl exec -ti cluster-example-1 -c postgres -- bash
  ```

- Connect to PostgreSQL using "psql". List databases and users. Comments?
  ```
  kubectl exec -ti cluster-example-1 -c postgres -- psql -c '\du' -c '\l'
  kubectl cnpg psql cluster-example -- -c '\du' -c '\l'
  ```

- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Exercise #B6 - Superuser access

By default, CloudNativePG restricts superuser access to PostgreSQL over the network and disables the use of `**ALTER SYSTEM**`. Why do you think these precautions are in place? What's your take on this approach?

Key Tasks:

- How do we prevent access from the network for the "**postgres**" user?
- Then, try running:
  ```
  kubectl cnpg psql cluster-example -- -c 'ALTER SYSTEM SET archive_mode TO off'
  ```
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Exercise #B7 - Superuser access

To understand how superuser access is restricted, follow these steps and then participate in a group discussion.

Key Tasks:

1. Take note of the existing secrets
2. Run **psql** with: "`SELECT usename FROM pg_shadow WHERE passwd IS NULL`"
3. Edit the cluster and set `enableSuperuserAccess` to `true`
4. Repeat steps 1 and 2. What has changed?
5. Disable superuser access again, and repeat 1 and 2. What has changed now?

Engage in exploration, enquire, and foster collaborative learning with your peers.

Exercise #B8 - Where are my Postgres logs?

All resources created and managed by CloudNativePG follow Kubernetes conventions by logging to standard output in JSON format. This includes PostgreSQL logs, which are never stored in the container's filesystem.

Key Tasks:

- Familiarise with the "**`kubectl logs <POD>`**" command. For example:
  `kubectl logs cluster-example-1`
- Now practice with the "**logs cluster/pretty**" commands of the plugin:
  `kubectl cnpg logs cluster cluster-example -f | kubectl cnpg logs pretty`
- What changed? Check the options of the commands with '-h'.
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Recommendation

For each exercise, keep a log window open to monitor the cluster in real time. For example, use the following command:

```
kubectl cnpg logs cluster cluster-example -f | kubectl cnpg logs pretty
```

Exercise #C1 - Testing the controller

The controller's role is to ensure that the observed state aligns with the desired state. Conduct your own experiments to observe this behavior in action.

Key Tasks:

- Get the list of available services
  `kubectl get services`
- Delete the **read-write** service
  `kubectl delete service cluster-example-rw`
- List the available services again. What has changed?
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Exercise #C2 - Configuring PostgreSQL

To modify PostgreSQL configurations, always use the cluster manifest and review the `.spec.postgresql.parameters` section. Keep in mind that CloudNativePG maintains exclusive control over certain parameters, so be sure to check which ones are restricted. Additionally, some parameters require special handling in a high-availability (HA) cluster.

Key Tasks:

- Check the current value of **max_connections** ("`SHOW max_connections`")
- Edit the cluster and increase its value. What happens? Why?
- Edit the cluster and decrease it value. What happens? Why?
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Exercise #C3 - Our first disruptions

Let's continue exploring the controller's capabilities by introducing disruptions and testing its self-healing features firsthand.

Key Tasks:

- Get the list of available pods, services and endpoints
  ```
  kubectl get pods -o wide
  kubectl get services,endpoints
  ```

- Delete a replica pod. What happened?
  ```
  kubectl delete pod cluster-example-2
  ```

- Delete a the primary's pod. What happened? Where's the primary?
  ```
  kubectl delete pod cluster-example-1
  ```

- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Exercise #C4 - Running pgbench

During POCs and demonstrations to customers, pgbench will be vital. You need to learn how to run it with CloudNativePG, using Kubernetes jobs. The cnpg plugin helps you by providing a way for you to create jobs for both initialization and run.

Key Tasks:

- Run the initialization job with a small scale
- Run pgbench for a minute.
- Run it again. What happens?
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

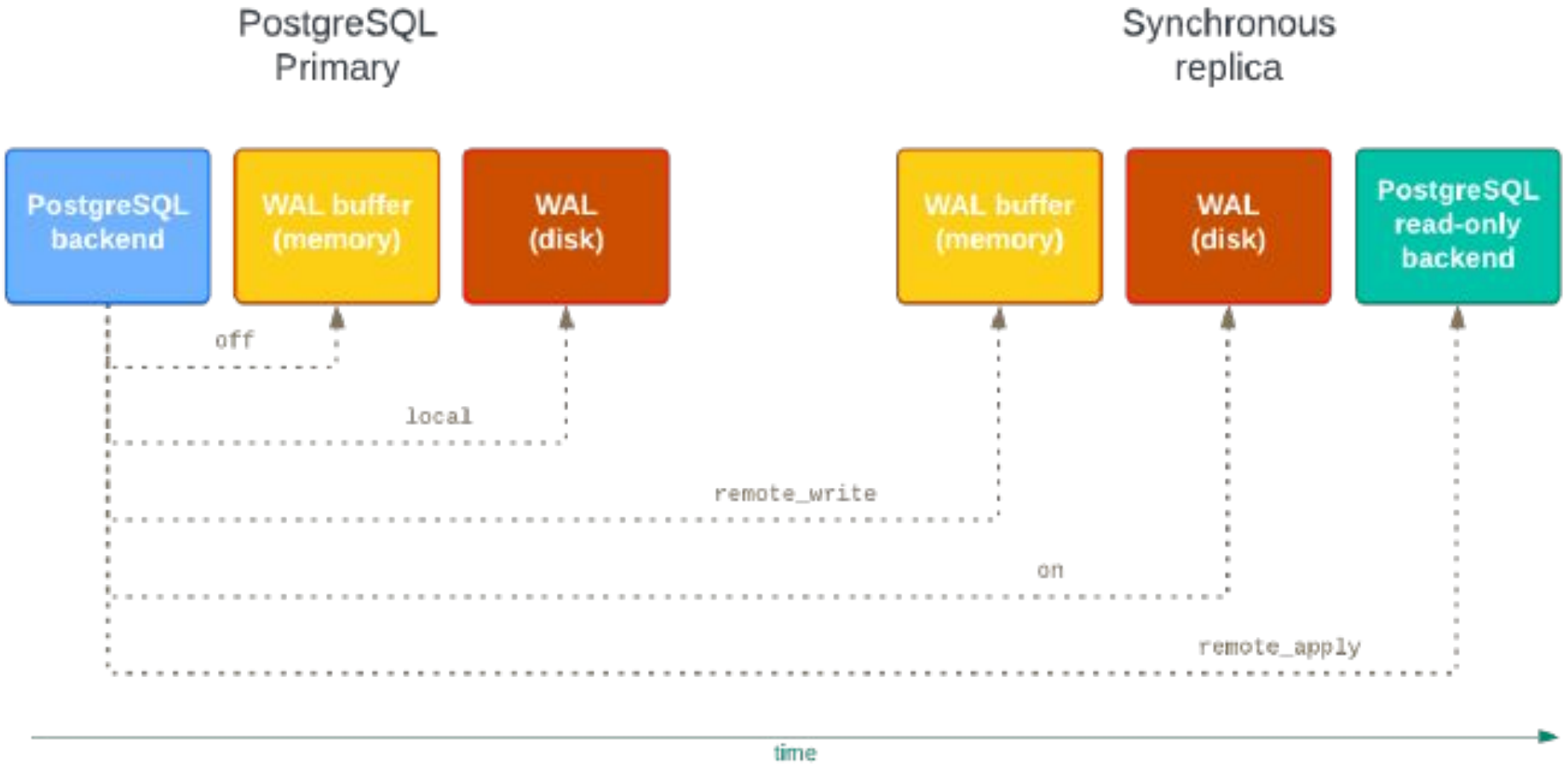
Exercise #C5 - Synchronous replication

The current recommendation for production environments is to setup a 3 instance cluster with at least one synchronous replica. This will reduce the risk of data loss and the time to promote a replica.

Key Tasks:

- Configure synchronous replication with quorum of at least 1 replica
- Simulate a failure on both replicas. Commit a transaction. What happens?
- Run pgbench trying different values of synchronous_commit (see next slide)
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

The “synchronous_commit” GUC



Exercise #C6 - Replication slots for High Availability

CloudNativePG automatically manages physical replication slots for each hot standby replica in the High Availability cluster, both in the primary and the standby. Read the documentation and learn more about primary and standby HA slots.

Key Tasks:

- Check the slots in the primary by querying the “pg_replication_slots” view:

```
SELECT slot_name, restart_lsn, active FROM pg_replication_slots WHERE NOT temporary AND slot_type = 'physical'"
```

- Check the slots in the standby instances
- Try a switchover. What happens?
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Cleanup

Destroy the “cluster-example” Postgres cluster from your environment:

```
kubectl delete cluster cluster-example
```

Verify it has been removed:

```
kubectl get cluster
```

Or … destroy the entire playground and recreate it! and don’t forget the operator!

```
./scripts/teardown.sh
./scripts/setup.sh
```

Exercise #D1 - Exploring the "pg-eu" Cluster

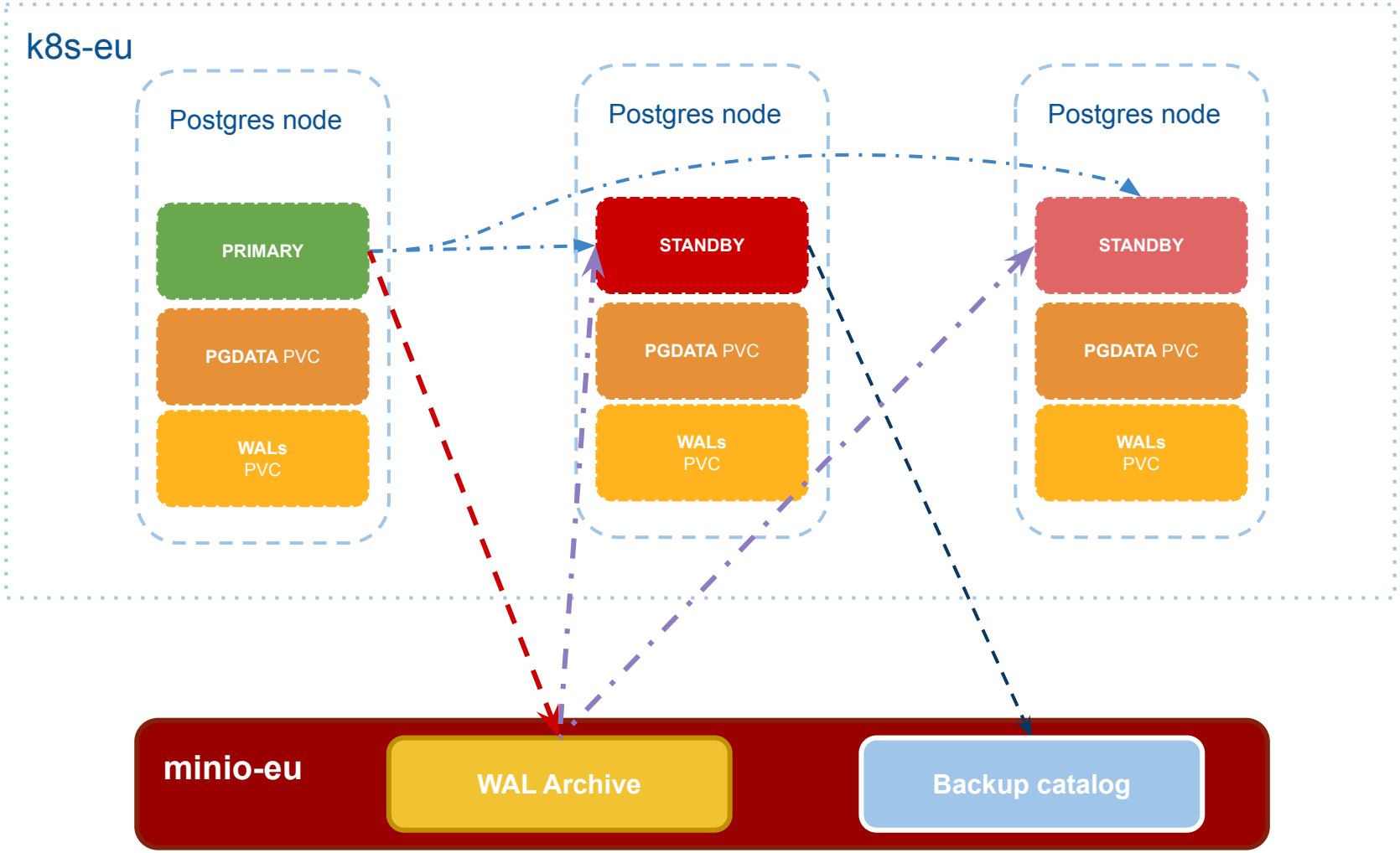
Take a moment to review the manifest for the "pg-eu" cluster. Explore the provided links to the documentation for a deeper understanding. This example offers valuable insights, so feel free to engage with your classmates in discussions.

Key Tasks:

- Deploy the cluster and verify where the instances are running:
  `kubectl --context kind-k8s-eu apply -f examples/eu/pg-eu.yaml`

- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

The “pg-eu” Postgres cluster in our playground



Exercise #D2 - WAL archiving

WAL archiving is directly controlled by CloudNativePG, with currently little room for customization and object store only support with Barman Cloud.

Key Tasks:

- Connect to the MinIO container and check WAL files are being shipped:
  `docker exec minio-eu ls /data/backups/pg-eu/wals/0000000100000000`
- What other ways do you know to check WAL archiving is working?
- Are replicas using the WAL archive at all?
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

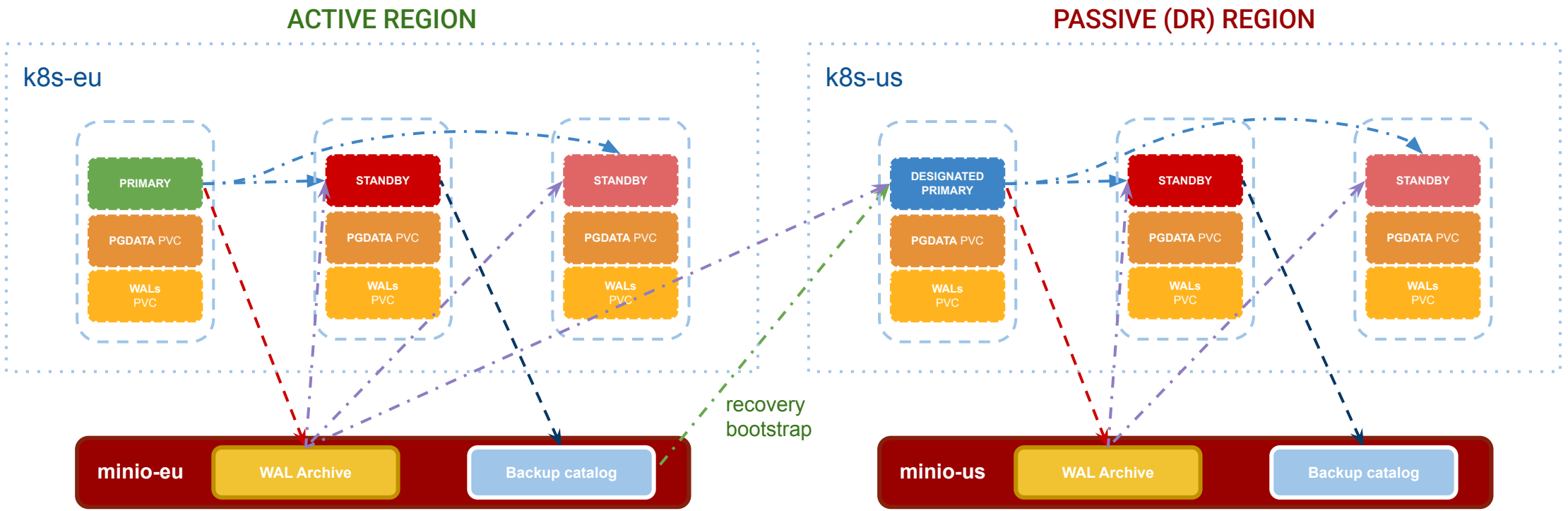
Exercise #D3 - Backups

CloudNativePG offers flexible backup solutions, supporting both scheduled and on-demand backups. These backups can be performed from either the primary or standby instances. Currently, CloudNativePG supports base backups using Barman Cloud for object storage and Kubernetes Volume Snapshots for persistent volumes.

Key Tasks:

- Verify if the immediate base backup has been taken:
  `docker exec minio-eu ls /data/backups/pg-eu/base`
- Take an on-demand backup using the plugin:
  `kubectl cnpg backup pg-eu`
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Distributed topology



Exercise #D4 - Setup the “pg-us” Cluster

A distinctive feature of CloudNativePG is its support for a distributed topology, achieved through replica clusters. In this setup, the "pg-us" cluster functions as a replica cluster, working in tandem with the "pg-eu" cluster to form a distributed PostgreSQL architecture.

Take a closer look at the "replica" stanza within the configuration of both clusters to understand how replication is structured across the distributed environment.

Key Tasks:

- Deploy the cluster and verify where the instances are running:
  `kubectl --context kind-k8s-us apply -f examples/us/pg-us.yaml`

- Consider how replication operates in this distributed setup and whether cascading replication is in use.

- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Exercise #D5 - Demote the “pg-eu” Cluster

In a typical active/passive data center setup, business continuity plans often require switching primary operations to the secondary data center every few months. CloudNativePG simplifies this process by providing a declarative approach. The first step in this procedure is called demotion, which transitions the current primary to a standby role, allowing seamless failover to the other data center.

Key Tasks:

- Modify the “pg-eu” cluster by setting “pg-us” as the primary cluster.
- Get the demotion token:
  `kubectl get cluster pg-eu -o jsonpath='{.status.demotionToken}'`
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Exercise #D6 - Promote the “pg-us” Cluster

Now that the "pg-eu" cluster has been demoted to a replica and is awaiting WAL files in the "pg-us" bucket, it's time to promote "pg-us" as the primary cluster. Seamless *promotion* requires using the promotion token (which corresponds to the demotion token of "pg-eu"). This operation is commonly referred to as a *data center switchover*.

Key Tasks:

- Update the configuration to promote "pg-us" to the primary role. Ensure that both the primary status and promotion token are set simultaneously to enable a smooth switchover.
- What happens on “pg-us”? and on “pg-eu”?
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*