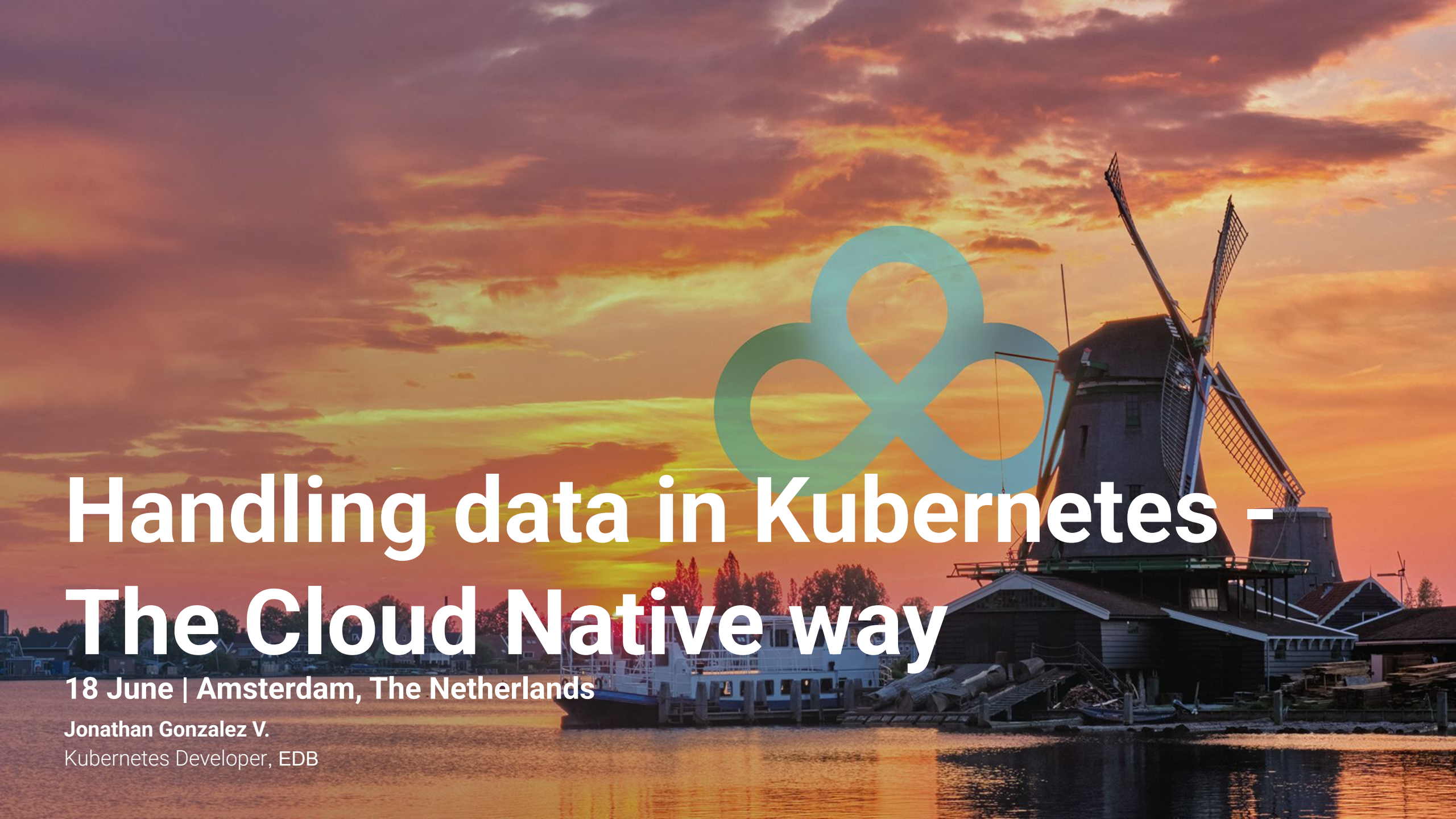


Handling data in Kubernetes - The Cloud Native way

18 June | Amsterdam, The Netherlands

Jonathan Gonzalez V.
Kubernetes Developer, EDB

Agenda

- Introduction to CloudNativePG
- CloudNativePG Playground Deployment
- Documentation is the key
- Moving around with the PostgreSQL Cluster
- Knowing the internal controller
- Using ArgoCD

CloudNativePG a bit of historical events

History

- **Aug 2019**: the Cloud Native journey at 2ndQuadrant starts
- **Sep 2020**: EDB acquires 2ndQuadrant
- **Feb 2022**: EDB decides to open source Cloud Native PostgreSQL
- **Apr 2022**: CloudNativePG is open sourced and submitted for CNCF Sandbox
- **Sep 2024**: CloudNativePG second submission for CNCF Sandbox
- **Jan 2025**: CloudNativePG is accepted in the CNCF Sandbox

The main features of CloudNativePG

- High Availability and Self-Healing
- Continuous backup (including snapshots)
- Point In Time Recovery (incl. snapshots)
- Managed services for rw and r/ro workloads
- Support for local PVCs
- “Security by default”, including mTLS
- Rolling updates, incl. minor PG releases
- Scale up/down of read-only replicas
- Native Prometheus exporter
- Logging to stdout in JSON format
- Synchronous replication
- Online import of Postgres databases
- Separate volume for WALs
- Postgres tablespaces, including temporary
- Replica clusters and distributed topologies
- Declarative management of roles, dbs, subs+pubs
- Declarative hibernation and fencing
- Connection pooling
- Postgres extensions (pgvector, PostGIS, …)
- **Major PostgreSQL upgrade (1.26.0)**

Section A

Setting up the environment

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:

```
kind create cluster --config k8s/kind-cluster.yaml
```

Key Tasks:

- 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`**

Exercise #A2 - CloudNativePG installation

To continue with the setup, you need to **deploy the CNPG operator** and ensure it is installed on the **control plane node**. The **cnpg plugin** can assist you throughout the deployment process. Familiarise yourself with the documentation.

Key Tasks:

- Install the operator in the cluster with:
```
kubectl cnpg install generate --control-plane \
  | kubectl apply -f - --server-side
```

- 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

CloudNativePG Documentation

- https://cloudnative-pg.io/docs/
  - Always use the current latest stable
  - Or go to the specific version
- There's a search bar, please use it
- Quickstart is always a good starting point
- Take time to read the full list of topics

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-example.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 #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 #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.

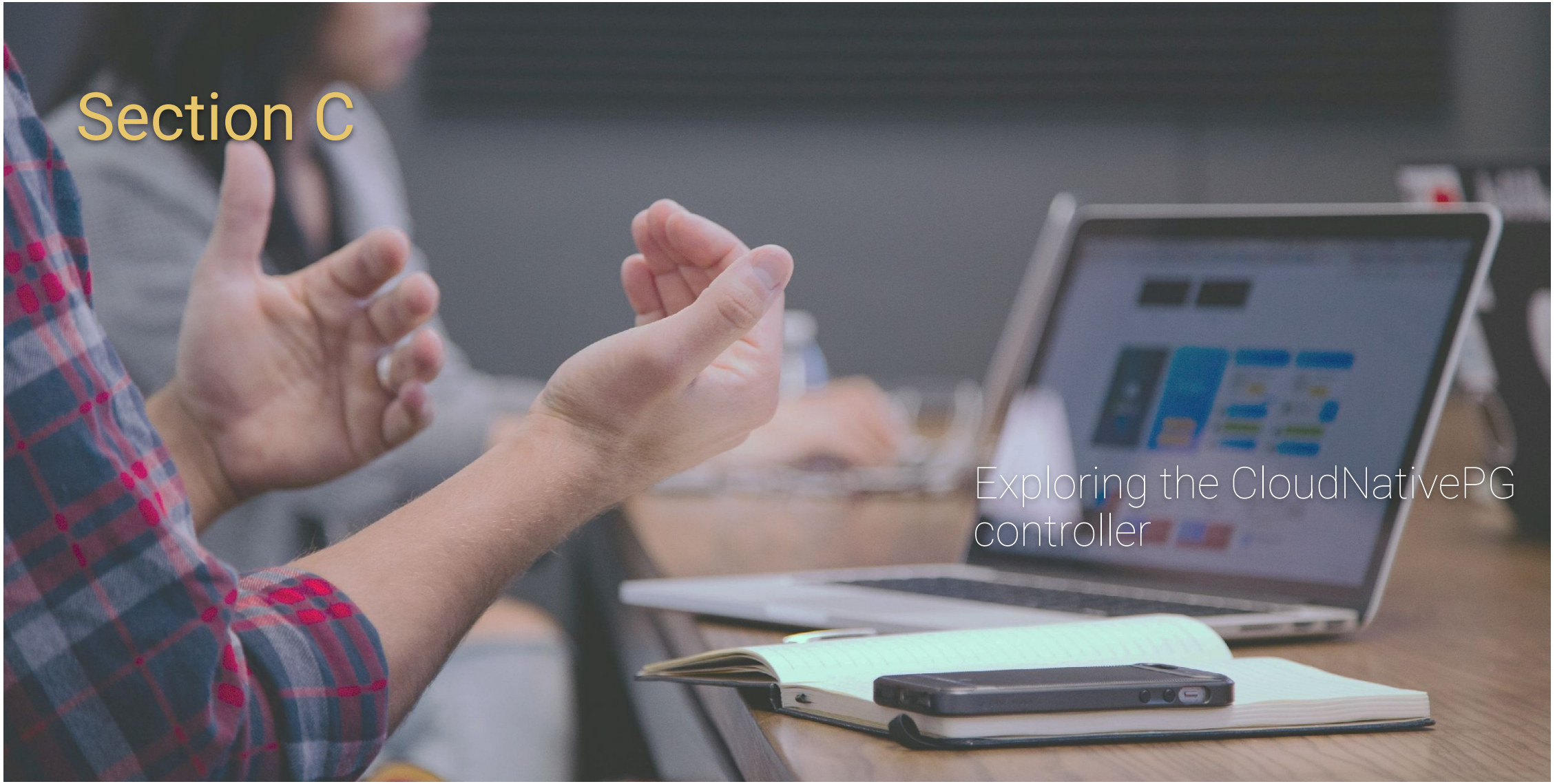
Key Tasks:

- 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 - Scaling

Scaling up and down is a common operation in Kubernetes. With PostgreSQL, scaling typically involves adding or removing replicas while maintaining a single primary. Let's explore how to perform this.

Key Tasks:

- Use the **`kubectl scale`** command to scale up, and observe.
  `kubectl scale cluster cluster-example --replicas 4`
- Use the **`kubectl scale`** command to scale down, and observe.
  `kubectl scale cluster cluster-example --replicas 3`
- What alternative approach could you have used for scaling?
- *Engage in exploration, enquire, and foster collaborative learning with your peers.*

Exercise #C4 - Our first disruptions

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

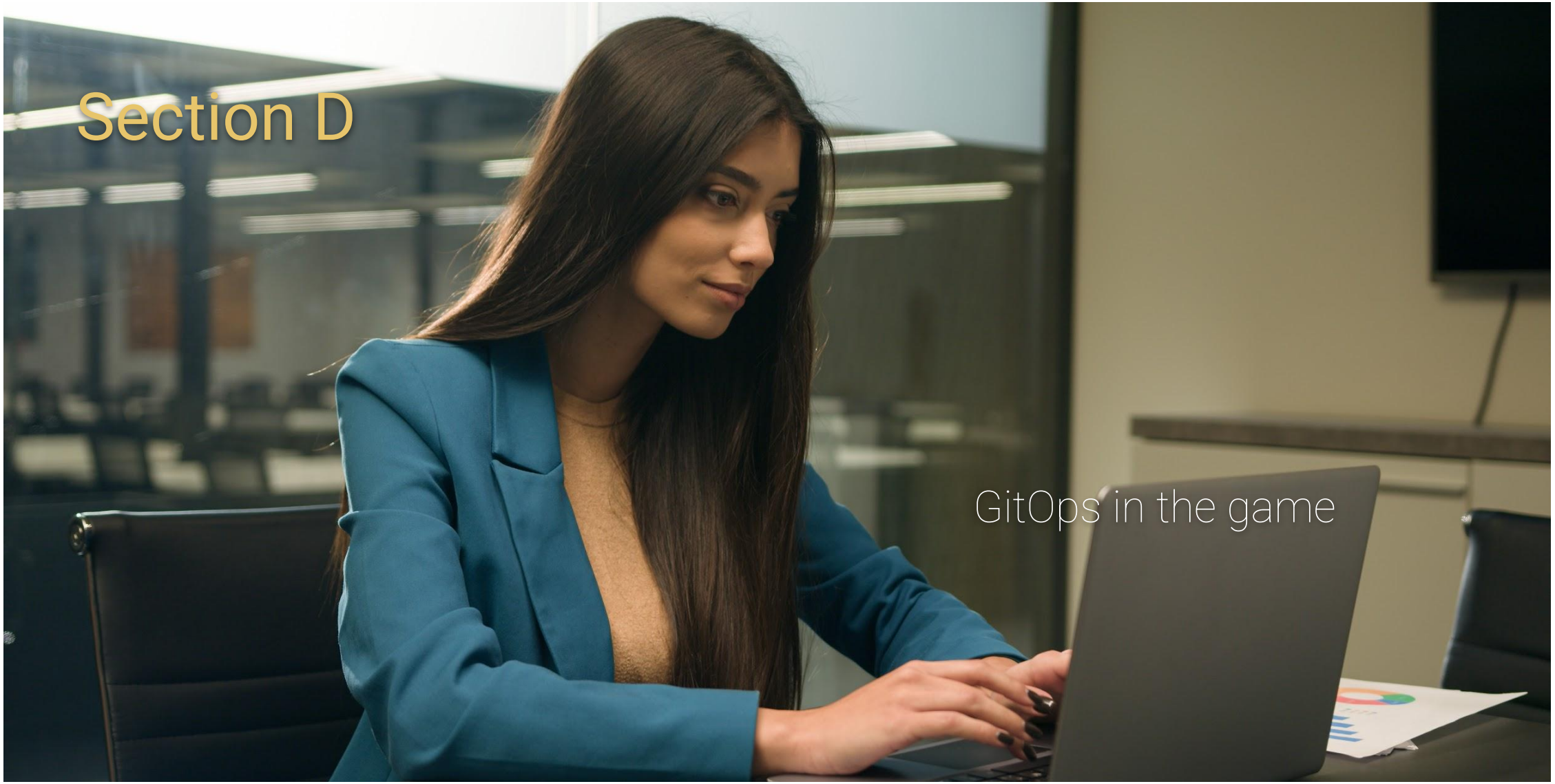
Key Tasks:

- 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.*



ArgoCD - Install

- Basic install
  - `kubectl create namespace argocd`
  - `kubectl apply -n argocd -f \ https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml`

- Forward port
  - `kubectl port-forward svc/argocd-server \ -n argocd 8080:443`

- Admin password
  - `argocd admin initial-password -n argocd`

- Login!
  - `argocd login --username admin localhost:8080`

ArgoCD - Installing Operator

Deploying CloudNativePG Operator with ArgoCD

- `argocd app create cloudnativepg \`
  `--repo https://github.com/sxd/cloudnative-pg-argocd.git \`
  `--path install-operator-argocdapp \`
  `--dest-server https://kubernetes.default.svc \`
  `--dest-namespace default`
- `argocd app sync cloudnativepg`
- `argocd app list`

ArgoCD - A simple cluster

Creating and managing the PostgreSQL cluster

- `argocd app create cnpg-clusters \`
  `--repo https://github.com/sxd/cloudnative-pg-argocd.git \`
  `--path clusters \`
  `--dest-server https://kubernetes.default.svc \`
  `--dest-namespace default`
- `argocd app sync cnpg-clusters`
- `argocd app sync cloudnative-pg-clusters`
- `kubectl -n cnpg-samples get pods -w`

ArgoCD - Web Interface

- Go to https://localhost:8080
- Login with admin user
- Navigate the resources and enjoy!

There’s always more!

- ArgoCD, FluxCD, KubeFleet, etc
- Every PR counts
  - https://github.com/argoproj/argo-cd/pull/22802
- Add more custom checks!

Thank you!

Connect with us:

Website: cloudnative-pg.io

Blog: cloudnative-pg.io/blog/

Github Discussions: github.com/cloudnative-pg/cloudnative-pg/discussions

Slack: communityinviter.com/apps/cloud-native/cncf

LinkedIn: linkedin.com/company/cloudnative-pg/

Mastodon: @CloudNativePG@mastodon.social

Bluesky: @CloudNativePG.bsky.social