Cloud Native PostgreSQL
Kubernetes Ready!

セールスエンジニア 吉野孝太郎
2021年5月26日

EDB

本日のターゲット

1. Cloud Native PostgreSQL

   Kubernetesをフル活用したPostgreSQLのご紹介

2. Cloud Native PostgreSQL デモ

   Azure上でOperatorがいかにパワフルかご体験

3. 【Kubernetes×PostgreSQL】をその手に！

EDB と Kubernetes

- Kubernetes Certified Service Provider (KCSP)
  - このステータスに到達する最初のPostgreSQL会社

- Silver Member of CNCF & Linux Foundation

- Red Hat Certified Kubernetes Operators
  - Cloud Native PostgreSQL (PostgreSQL Operator)
  - Cloud Native BDR (BDR Operator)

EDB

Cloud Native PostgreSQL (CNP)

Cloud Native PostgreSQL と BDR

Kubernetesネイティブの EDB 製品

- Operators for Kubernetes
  - KubernetesのようにGO言語で書かれた
  - Kubernetes向けに設計
  - Kubernetes API serverと完全に統合
  - 従来の外部ツールは不要
    - Repmgr, Patroni, Stolon, Failover Manager
- コンテナイメージ
  - PostgreSQL 10, 11, 12, 13
  - EDB Postgres Advanced 10, 11, 12, 13
  - BDR (3.6 on 2ndQPostgres in Q1/2021)

Kubernetes オペレーター

Kubernetes コントローラを拡張し、複雑なアプリケーションの動作を定義します。

- Kubernetesオペレーターは、プログラム的な方法でオペレーターの作業を自動化します
- **PostgreSQL クラスターは複雑なアプリケーションです。**
  - デプロイとコンフィグ
  - 障害の検出とフェールオーバー
  - アップデートとスイッチオーバー
  - バックアップとリカバリ
- Kubernetesのネイティブコンポーネントと機能に依存:
  - セルフヒーリング, 高可用性, スケーラビリティ, リソース制御, アクセス制限, …
  - 宣言的かつ完全に自動化

コンテナイメージ

すべてのコンテナイメージは RedHat UBI 8 に基づいています

- Operator コンテナイメージ
  - EDB Limited Use License
- Application コンテナイメージ(PostgreSQL)
  - PostgreSQL:
    - The Postgres License (OSS)
    - GNU GPL 3 (Barman Cloud)
  - EDB Postgres Advanced Server:
    - EDB Limited Use License

ターゲット環境

Kubernetes環境ならどこでも実行可能！

- Private Cloud：
  - **Kubernetes 1.16+**
  - OpenShift 4.5+
- Public Cloud
  - Microsoft Azure (AKS) – リリース済！
  - Amazon Web Services (EKS) - 2021/Q2 (4-6月期)
  - Google Cloud (GKS) - 2021/Q2 (4-6月期)

評価／トライアル

Operator with PostgreSQL

- 試用ライセンスキーは不要
- 30日 以降の調整の試みなし
  - 各 Postgres クラスターの作成から

Operator with EDB Postgres Advanced

- 試用ライセンスキーが必要
- 有効期限 = ライセンス要求から 60日
- 有効期限後に調整の試みなし

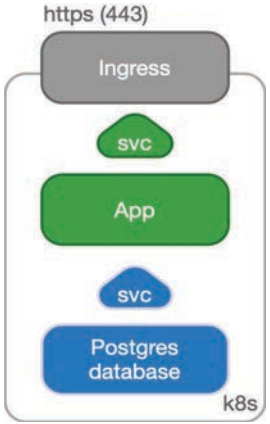
トライアルライセンスキーの取得

https://cloud-native.enterprisedb.com

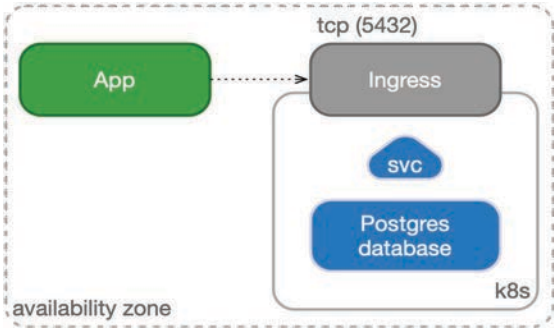
ユースケース

次の分類は、アプリケーションが存在する場所に基づいています。

ケース1: フルKubernetes化



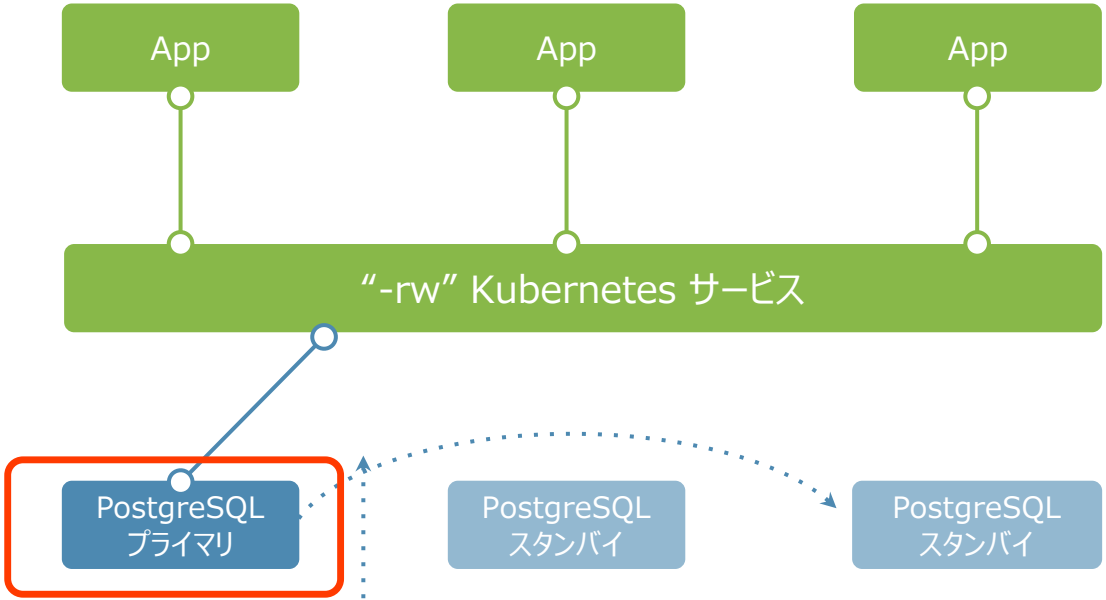
ケース2: レガシーシステムがKubernetesの外



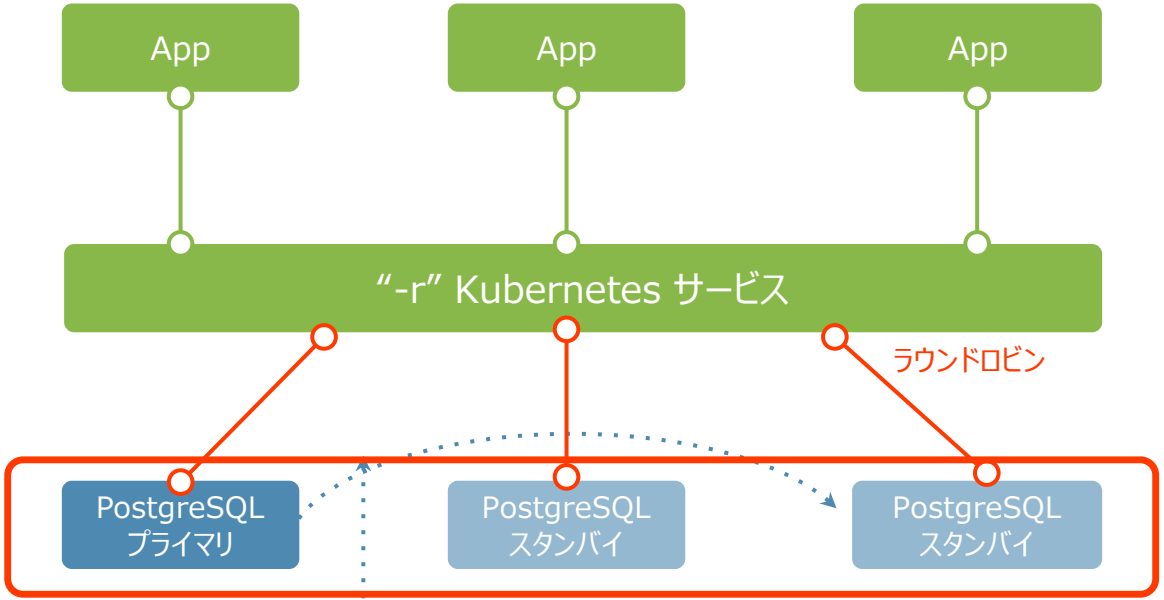
PostgreSQLクラスタ アーキテクチャ

- 1台の PostgreSQL プライマリ
- 任意の数のホット スタンバイ
  - PostgreSQLネイティブ ストリーミング レプリケーション
    - Async (default) and Sync (quorum-based)
  - HA構成には1台以上必要
  - pg_rewindの透過的なサポート
- 3 つの Kubernetes サービスをアプリに提供:
  - -rw suffix (読み取り/書き込みワークロード)
  - -r suffix (読み取りワークロード)
  - -ro suffix (プライマリを除く読み取りワークロード- in 1.1.0)

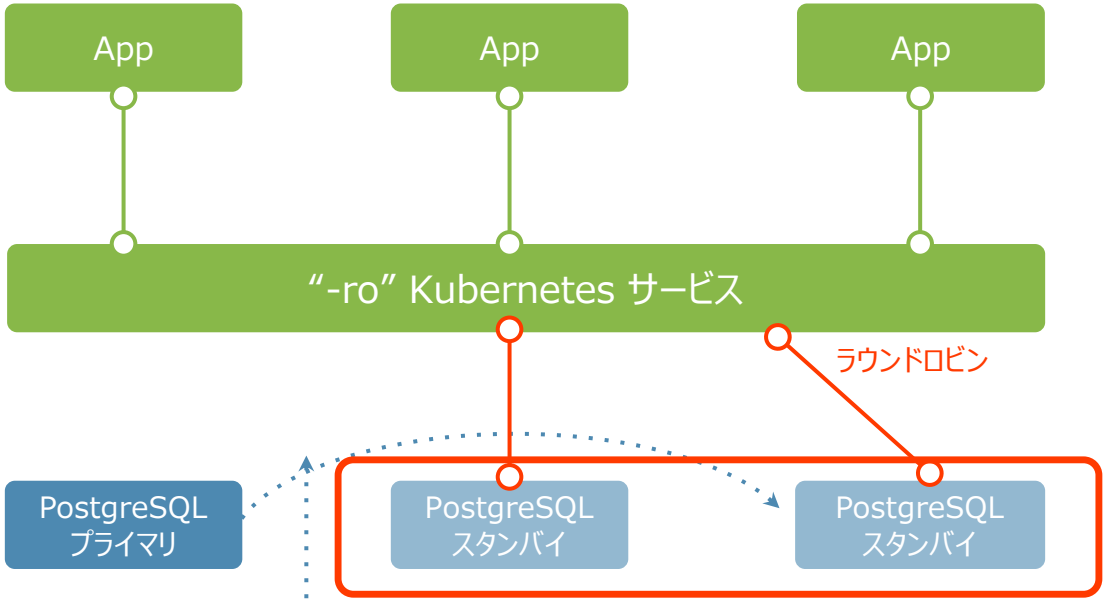
Read-writeワークロード(“-rw” service)



Readワークロード(“-r” service)



Readワークロード(“-ro” service)



クラスタ管理

- セルフヒーリング
  - 死活と読み取りの監視
  - プライマリのフェールオーバー
  - レプリカの自動作成
- プライマリの切り替え(スイッチオーバー)
  - 選択したスタンバイのプロモート
- ローリングアップデート
- スケールアップ/ダウン

ストレージ管理

- 永続ボリューム のサポート (PVC)
  - PVCの自動生成
  - PVC テンプレートのサポート
  - 同じクラスター内のポッドのストレージの再利用
- ストレージクラスの動的プロビジョニング
  - Azure (default, managed-premium), AWS(gp2, io1, st1, sc1), Google(standard)
- 選択の自由
  - Local storage（デフォルト）
  - Network storage

ローリングアップデート

- ダウンタイムのないローリングアップデート
  - 以下の順序で自動更新
    1. 最初に、スタンバイ群が1台ずつアップデートされる
    2. その後、プライマリがスイッチオーバーされる
    3. 最後に古いプライマリがアップデートされる
  - supervised / unsupervised
- トリガーされる時:
  - オペレータのアップデート
  - PostgreSQL のマイナーアップデート
  - 設定の変更で再起動が必要な場合にもトリガー
- **レプリカセットまたはステートフルセットを使用しなかった理由の一つ**

セキュリティ

- 4Cセキュリティモデル
  - Cloud, Cluster, Container, Code
- ポッドセキュリティポリシーとセキュリティコンテキスト
  - コンテナとボリュームアクセスにroot権限は不要
- TLS 暗号化コネクション
- PostgreSQLシークレットの作成
- 静的コード分析
  - Linters と カバレッジスキャン

TLS (SSL) サポート

- ネイティブで完全に自動化
  - オペレータの証明機関
  - 各 Postgres クラスターの証明機関
- TLS コネクションがデフォルトで有効
  - TLS 証明書に基づくクライアント認証

継続的なバックアップ

- スケジュールとオンデマンド
- オブジェクト ストアのサポート(現在S3互換のみ)
  - Public clouds (AWSでサポート)
  - Private clouds (e.g. MinIO)
- Barman Cloud に依存
  - barman-cloud-wal-archive
    - archive_commandを経由したWAL配送
  - barman-cloud-backup
    - スケジュール／オンデマンドのバックアップコマンド

リカバリー

- バックアップから新しいクラスターを作成する
- base backup レストア
  - Full or PITR
- バックアップWALファイルをプルおよび再生
- Barman Cloud に依存
  - barman-cloud-restore
    - base backup からのリストア
  - barman-cloud-wal-restore
    - オンデマンドのWALのプル

リソース管理

- Podリソース制御
  - CPU とメモリの要求と制限 (スケジュール時のみ適用)
- Podアフィニティーwith nodeSelector
  - Podを展開する場所
- Podアンチアフィニティーpolicies
  - Podを展開しない場所
- Pod Disruption Budget
  - PostgreSQL グループで同時中断を制限する
  - Kubernetes ノードの制御メンテナンスウィンドウ

カスタム リソース定義(CRD)

- “Cluster” リソース
  - Kubernetes コントローラの拡張
- “instances” パラメータ
  - クラスター内の PostgreSQL インスタンスの数
  - ホットスタンバイレプリカ数 = instances - 1
    - RPO=0 の非同期および同期レプリケーションをサポート

クラスタのデプロイメント

- 宣言型の構成
  - “kubectl” - Kubernetesの公式クライアントコマンドラインツール
  - YAML マニフェストファイル
  - Infrastructure as Code (IaC)
- クラスタの作成／更新と削除

  kubectl apply -f cluster-example.yaml

  kubectl delete -f cluster-example.yaml

マニフェストファイル（yamlファイル）

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: cluster-example
spec:
  instances: 3
  imageName: quay.io/enterprisedb/postgresql:13.2
storage:
    size: 1Gi
```

各種マニフェストファイルのサンプル

https://www.enterprisedb.com/docs/kubernetes/cloud_native_postgresql/samples/

同期レプリケーション

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: cluster-example
spec:
  instances: 3
  minSyncReplicas: 1
  maxSyncReplicas: 2
  :
```

※ 0 <= minSyncReplicas <= maxSyncReplicas < instances

PostgreSQLの設定

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: cluster-example
spec:
  instances: 3
  postgresql:
    parameters:
      max_worker_processes: "60"
    pg_hba:
      - host all all all md5
  :
```

EDB Postgres Advanced Server

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: cluster-example
spec:
  instances: 3
  imageName: quay.io/enterprisedb/edb-postgres-advanced:13.2
  licenseKey: <ここにライセンスキーを貼付け>
  :
```

トライアルライセンスキーの取得

https://cloud-native.enterprisedb.com

EDB

Cloud Native PostgreSQL

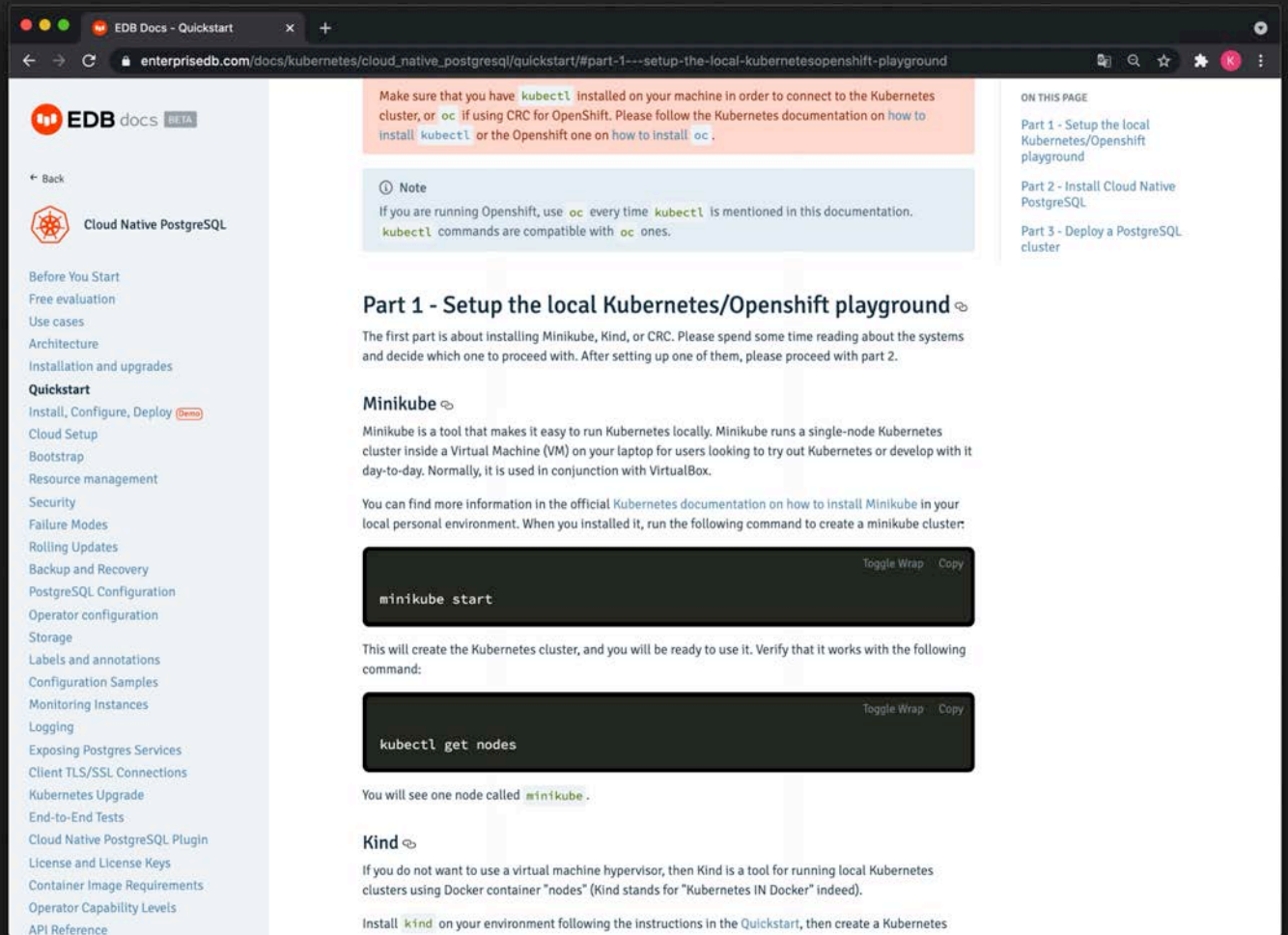
デモ

Quickstart

https://www.enterprisedb.com/docs/kubernetes/cloud_native_postgresql/quickstart/

- Quickstartの手順に沿って、デモを行います。
  - パート１　Kubernetes/Openshift 環境のセットアップ

    公式ドキュメントは、Minikubeを使用して
    ローカルにKubernetesクラスタを作成する方法です。
    今回のデモでは、Azureのクラウド上に作成する方法を説明します。
  - パート２　Operatorのインストール
  - パート３　PostgreSQLのデプロイ



EDB Docs - Quickstart

enterprisedb.com/docs/kubernetes/cloud_native_postgresql/quickstart/#part-1---setup-the-local-kubernetesopenshift-playground

EDB docs BETA

← Back

Cloud Native PostgreSQL

- Before You Start
- Free evaluation
- Use cases
- Architecture
- Installation and upgrades
- **Quickstart**
- Install, Configure, Deploy Demo
- Cloud Setup
- Bootstrap
- Resource management
- Security
- Failure Modes
- Rolling Updates
- Backup and Recovery
- PostgreSQL Configuration
- Operator configuration
- Storage
- Labels and annotations
- Configuration Samples
- Monitoring Instances
- Logging
- Exposing Postgres Services
- Client TLS/SSL Connections
- Kubernetes Upgrade
- End-to-End Tests
- Cloud Native PostgreSQL Plugin
- License and License Keys
- Container Image Requirements
- Operator Capability Levels
- API Reference

Make sure that you have `kubectl` installed on your machine in order to connect to the Kubernetes cluster, or `oc` if using CRC for OpenShift. Please follow the Kubernetes documentation on how to install `kubectl` or the Openshift one on how to install `oc`.

Note

If you are running Openshift, use `oc` every time `kubectl` is mentioned in this documentation. `kubectl` commands are compatible with `oc` ones.

ON THIS PAGE

- Part 1 - Setup the local Kubernetes/Openshift playground
- Part 2 - Install Cloud Native PostgreSQL
- Part 3 - Deploy a PostgreSQL cluster

Part 1 - Setup the local Kubernetes/Openshift playground

The first part is about installing Minikube, Kind, or CRC. Please spend some time reading about the systems and decide which one to proceed with. After setting up one of them, please proceed with part 2.

Minikube

Minikube is a tool that makes it easy to run Kubernetes locally. Minikube runs a single-node Kubernetes cluster inside a Virtual Machine (VM) on your laptop for users looking to try out Kubernetes or develop with it day-to-day. Normally, it is used in conjunction with VirtualBox.

You can find more information in the official Kubernetes documentation on how to install Minikube in your local personal environment. When you installed it, run the following command to create a minikube cluster:

Toggle Wrap Copy

```
minikube start
```

This will create the Kubernetes cluster, and you will be ready to use it. Verify that it works with the following command:

Toggle Wrap Copy

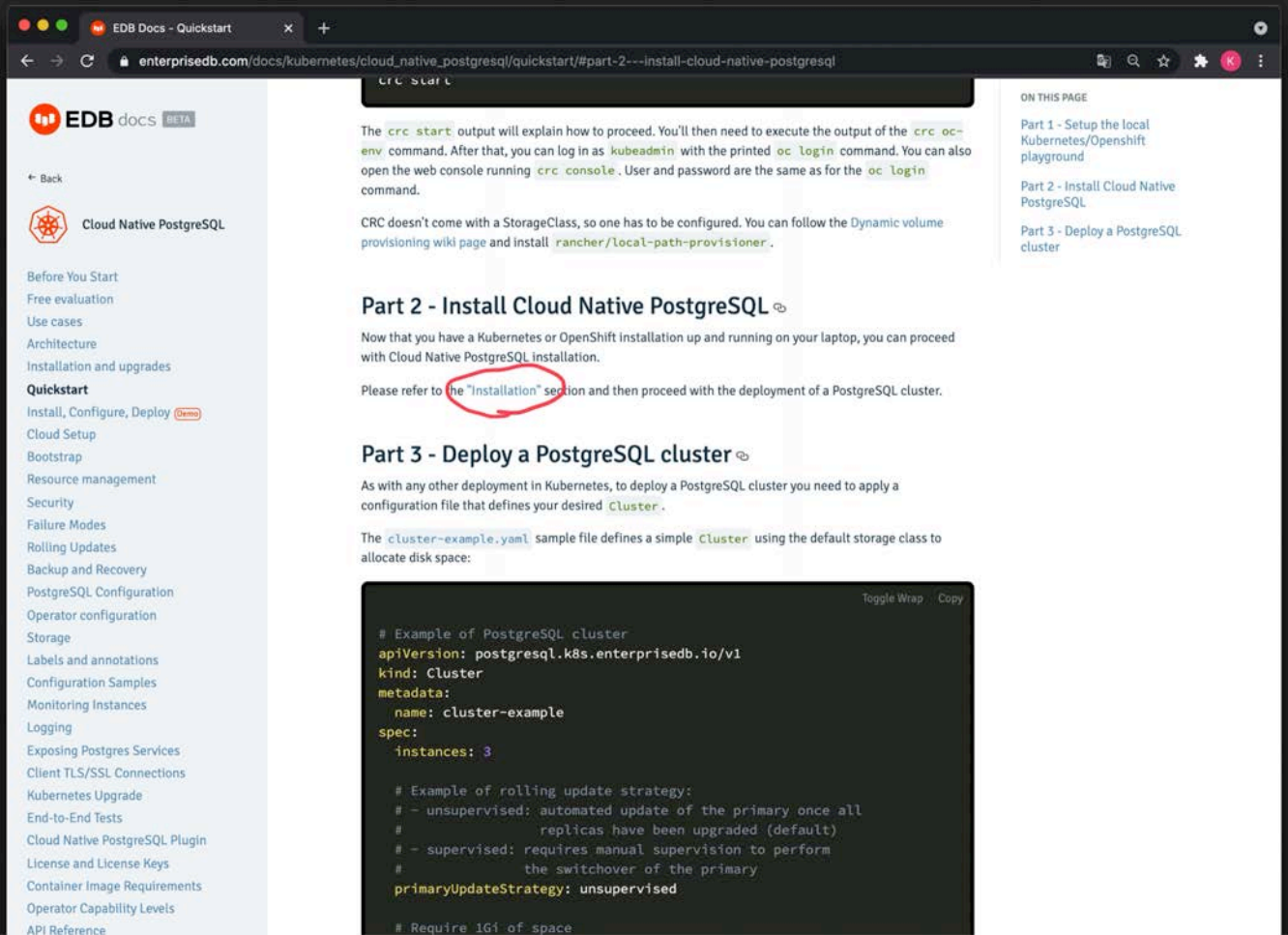
```
kubectl get nodes
```

You will see one node called `minikube`.

Kind

If you do not want to use a virtual machine hypervisor, then Kind is a tool for running local Kubernetes clusters using Docker container "nodes" (Kind stands for "Kubernetes IN Docker" indeed).

Install `kind` on your environment following the instructions in the Quickstart, then create a Kubernetes



EDB Docs - Quickstart

enterprisedb.com/docs/kubernetes/cloud_native_postgresql/quickstart/#part-2---install-cloud-native-postgresql

```
crc start
```

The `crc start` output will explain how to proceed. You'll then need to execute the output of the `crc oc-env` command. After that, you can log in as `kubeadmin` with the printed `oc login` command. You can also open the web console running `crc console`. User and password are the same as for the `oc login` command.

CRC doesn't come with a StorageClass, so one has to be configured. You can follow the Dynamic volume provisioning wiki page and install `rancher/local-path-provisioner`.

Part 2 - Install Cloud Native PostgreSQL

Now that you have a Kubernetes or OpenShift installation up and running on your laptop, you can proceed with Cloud Native PostgreSQL installation.

Please refer to the "Installation" section and then proceed with the deployment of a PostgreSQL cluster.

Part 3 - Deploy a PostgreSQL cluster

As with any other deployment in Kubernetes, to deploy a PostgreSQL cluster you need to apply a configuration file that defines your desired `Cluster`.

The `cluster-example.yaml` sample file defines a simple `Cluster` using the default storage class to allocate disk space:

Toggle Wrap　Copy

```
# Example of PostgreSQL cluster
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: cluster-example
spec:
  instances: 3

  # Example of rolling update strategy:
  # - unsupervised: automated update of the primary once all
  #                 replicas have been upgraded (default)
  # - supervised: requires manual supervision to perform
  #               the switchover of the primary
  primaryUpdateStrategy: unsupervised

  # Require 1Gi of space
```

ON THIS PAGE

- Part 1 - Setup the local Kubernetes/Openshift playground
- Part 2 - Install Cloud Native PostgreSQL
- Part 3 - Deploy a PostgreSQL cluster

EDB docs BETA

← Back

Cloud Native PostgreSQL

- Before You Start
- Free evaluation
- Use cases
- Architecture
- Installation and upgrades
- **Quickstart**
- Install, Configure, Deploy Demo
- Cloud Setup
- Bootstrap
- Resource management
- Security
- Failure Modes
- Rolling Updates
- Backup and Recovery
- PostgreSQL Configuration
- Operator configuration
- Storage
- Labels and annotations
- Configuration Samples
- Monitoring Instances
- Logging
- Exposing Postgres Services
- Client TLS/SSL Connections
- Kubernetes Upgrade
- End-to-End Tests
- Cloud Native PostgreSQL Plugin
- License and License Keys
- Container Image Requirements
- Operator Capability Levels
- API Reference

EDB Docs - Quickstart

enterprisedb.com/docs/kubernetes/cloud_native_postgresql/quickstart/#part-3---deploy-a-postgresql-cluster

EDB docs BETA

← Back

Cloud Native PostgreSQL

Before You Start
Free evaluation
Use cases
Architecture
Installation and upgrades
Quickstart
Install, Configure, Deploy Demo
Cloud Setup
Bootstrap
Resource management
Security
Failure Modes
Rolling Updates
Backup and Recovery
PostgreSQL Configuration
Operator configuration
Storage
Labels and annotations
Configuration Samples
Monitoring Instances
Logging
Exposing Postgres Services
Client TLS/SSL Connections
Kubernetes Upgrade
End-to-End Tests
Cloud Native PostgreSQL Plugin
License and License Keys
Container Image Requirements
Operator Capability Levels
API Reference

provisioning wiki page and install `rancher/local-path-provisioner`.

Part 2 - Install Cloud Native PostgreSQL

Now that you have a Kubernetes or OpenShift installation up and running on your laptop, you can proceed with Cloud Native PostgreSQL installation.

Please refer to the "Installation" section and then proceed with the deployment of a PostgreSQL cluster.

Part 3 - Deploy a PostgreSQL cluster

As with any other deployment in Kubernetes, to deploy a PostgreSQL cluster you need to apply a configuration file that defines your desired `Cluster`.

The `cluster-example.yaml` sample file defines a simple `Cluster` using the default storage class to allocate disk space:

Toggle Wrap　Copy

```yaml
# Example of PostgreSQL cluster
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: cluster-example
spec:
  instances: 3

  # Example of rolling update strategy:
  # - unsupervised: automated update of the primary once all
  #                 replicas have been upgraded (default)
  # - supervised: requires manual supervision to perform
  #               the switchover of the primary
  primaryUpdateStrategy: unsupervised

  # Require 1Gi of space
  storage:
    size: 1Gi
```

ⓘ There's more

For more detailed information about the available options, please refer to the "API Reference" section.

ON THIS PAGE

Cloud Native PostgreSQL Plugin

https://www.enterprisedb.com/docs/kubernetes/cloud_native_postgresql/cnp-plugin/

- PostgreSQLクラスタを管理するツールでPluginとして提供
  - インストールについて

    curl -sSfL ¥

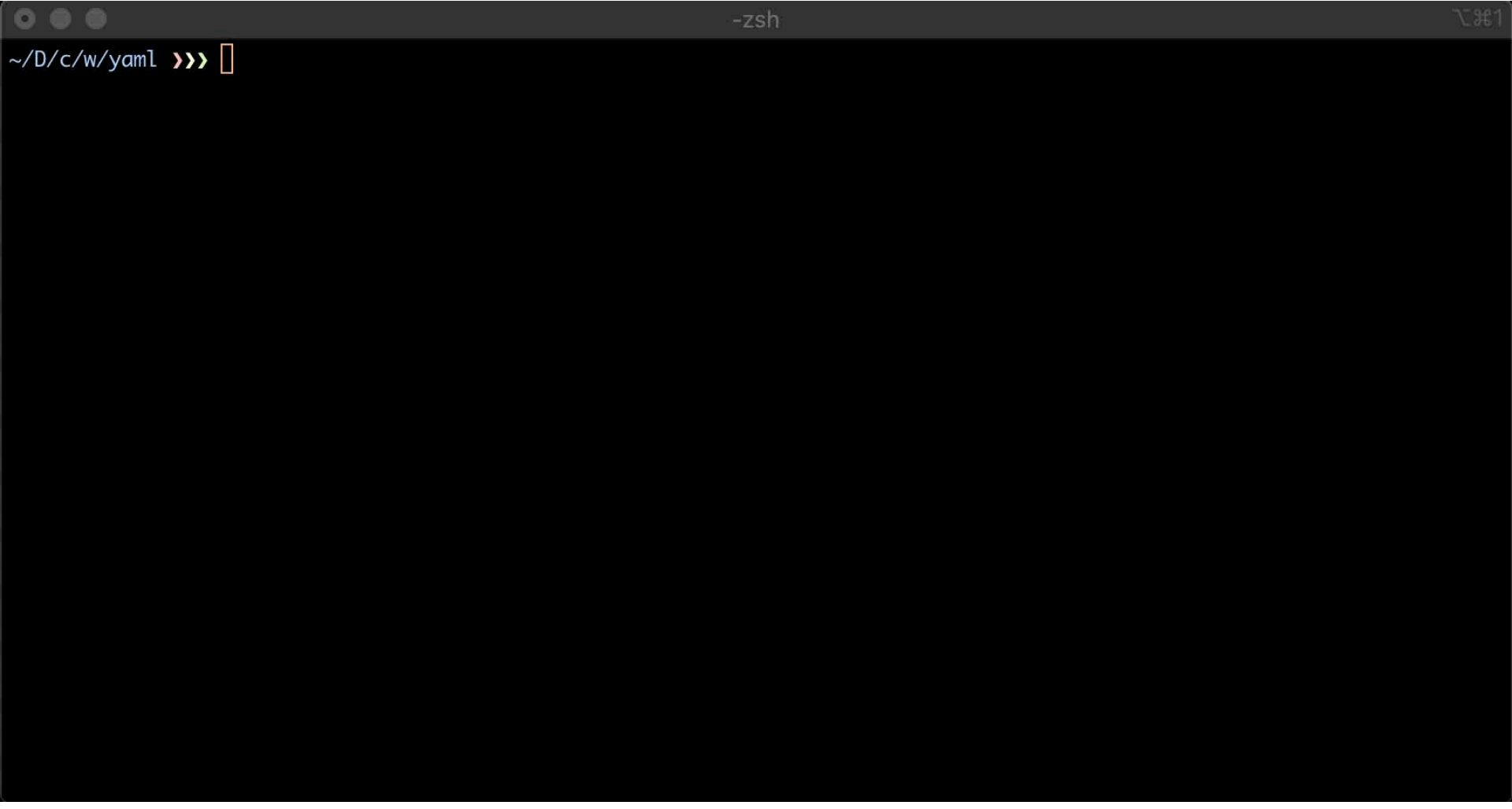
    https://github.com/EnterpriseDB/kubectl-cnp/raw/main/install.sh | ¥

    sudo sh -s -- -b /usr/local/bin
  - ステータスの確認

    kubectl cnp status cluster-example

    kubectl cnp status cluster-example --verbose（ステータスの詳細）
  - スイッチオーバー（手動）

    kubectl cnp promote cluster-example cluster-example-2


-zsh
~/D/c/w/yaml ❯❯❯

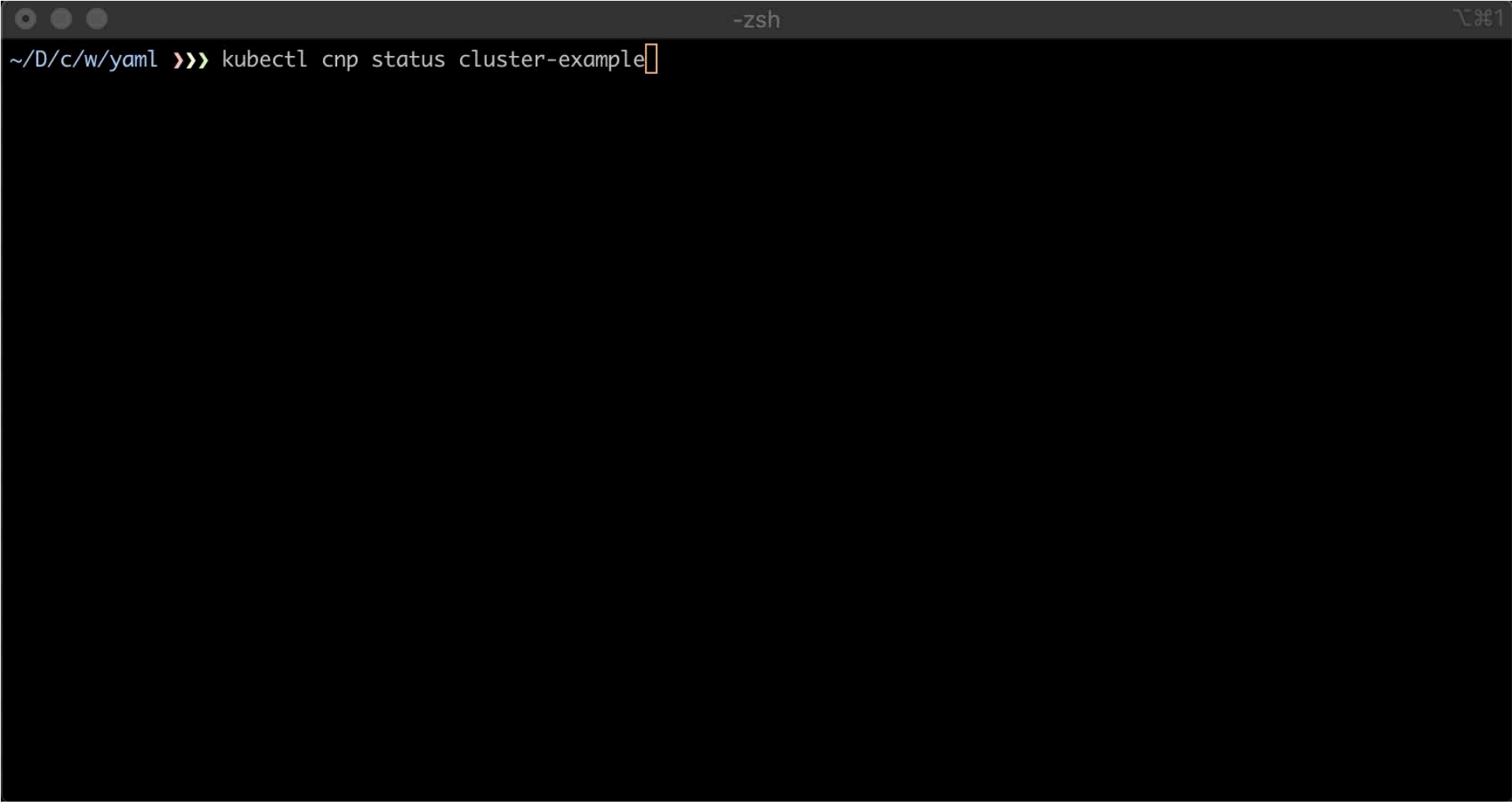
Operatorによる自動クラスタ管理

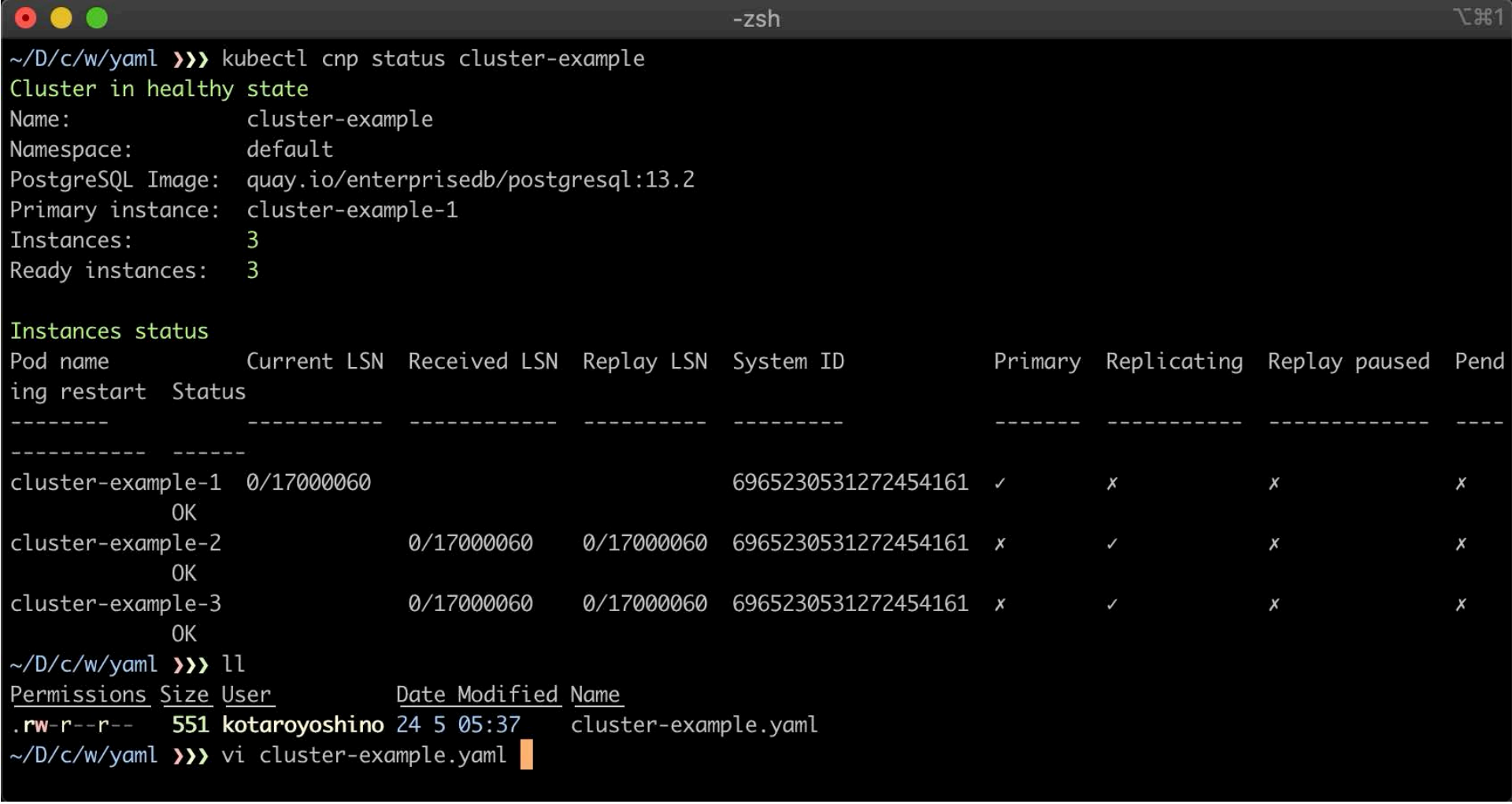
- セルフヒーリング
  - プライマリを削除してみよう！

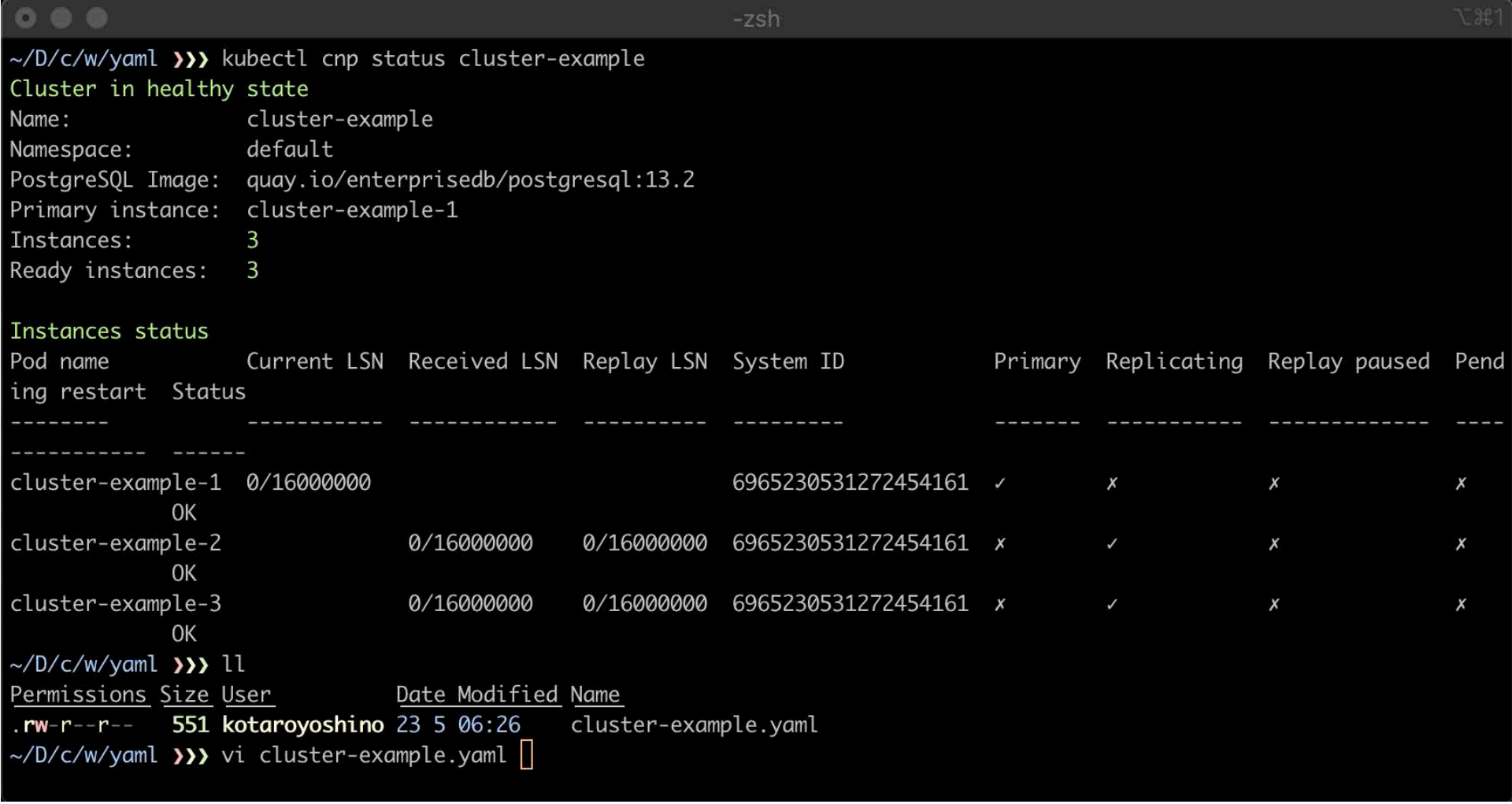
    kubectl delete pods cluster-example-1 --grace-period=0
- ローリングアップデート
  - マイナーアップデート

    yamlファイルで、バージョン指定を変更して
    kubectl apply -f cluster-example.yaml
- スケールアップ/ダウン機能
  - ホットスタンバイの増減

    yamlファイルで、 instancesの数を変更して
    kubectl apply -f cluster-example.yaml


-zsh
~/D/c/w/yaml ❯❯❯ kubectl cnp status cluster-example

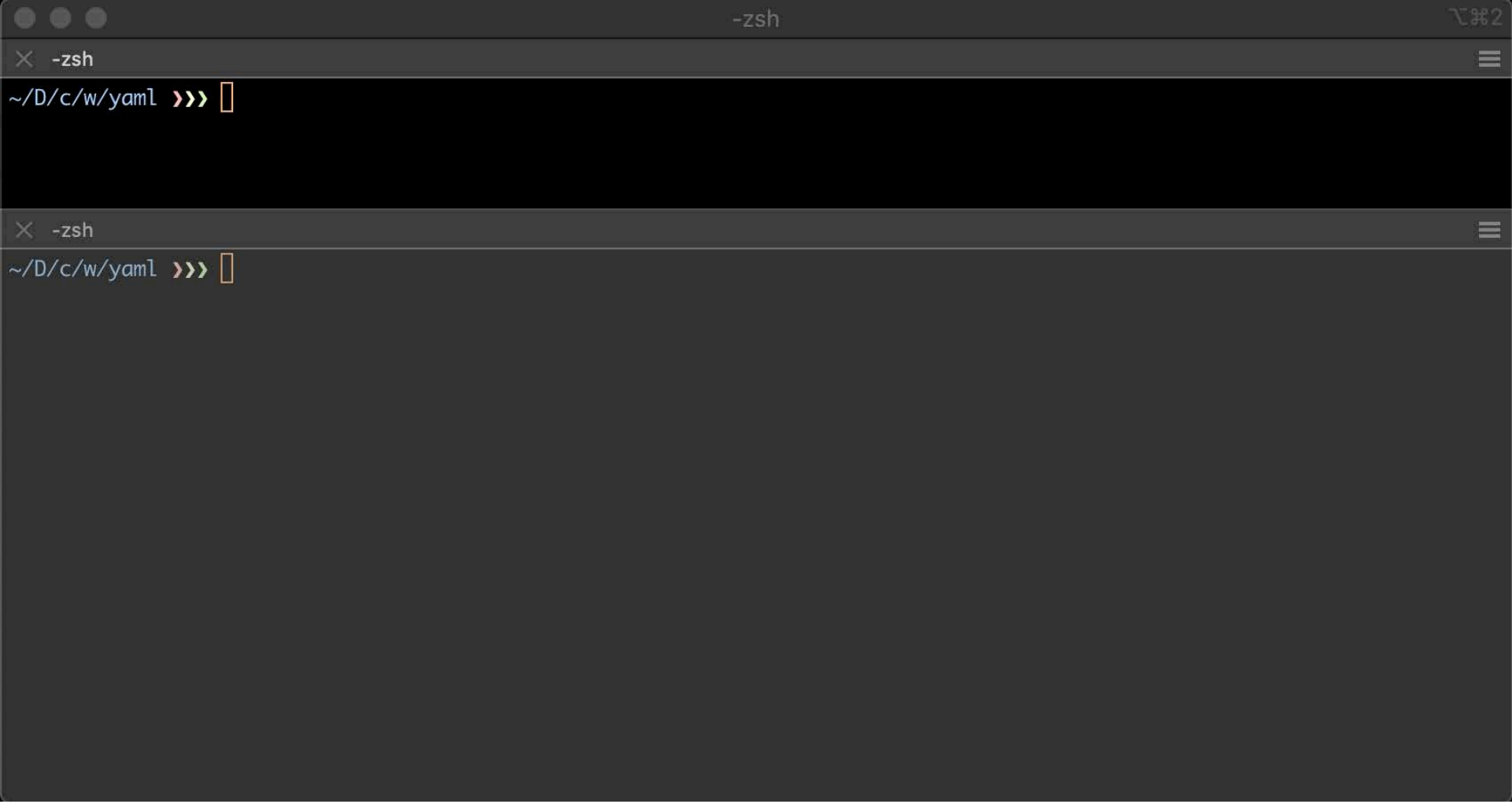

-zsh
~/D/c/w/yaml ❯❯❯ kubectl cnp status cluster-example
Cluster in healthy state
Name: cluster-example
Namespace: default
PostgreSQL Image: quay.io/enterprisedb/postgresql:13.2
Primary instance: cluster-example-1
Instances: 3
Ready instances: 3
Instances status
Pod name Current LSN Received LSN Replay LSN System ID Primary Replicating Replay paused Pending restart Status
cluster-example-1 0/17000060 6965230531272454161 ✓ ✗ ✗ ✗ OK
cluster-example-2 0/17000060 0/17000060 6965230531272454161 ✗ ✓ ✗ ✗ OK
cluster-example-3 0/17000060 0/17000060 6965230531272454161 ✗ ✓ ✗ ✗ OK
~/D/c/w/yaml ❯❯❯ ll
Permissions Size User Date Modified Name
.rw-r--r-- 551 kotaroyoshino 24 5 05:37 cluster-example.yaml
~/D/c/w/yaml ❯❯❯ vi cluster-example.yaml


-zsh
~/D/c/w/yaml ❯❯❯ kubectl cnp status cluster-example
Cluster in healthy state
Name: cluster-example
Namespace: default
PostgreSQL Image: quay.io/enterprisedb/postgresql:13.2
Primary instance: cluster-example-1
Instances: 3
Ready instances: 3
Instances status
Pod name Current LSN Received LSN Replay LSN System ID Primary Replicating Replay paused Pending restart Status
cluster-example-1 0/16000000 6965230531272454161 ✓ ✗ ✗ ✗ OK
cluster-example-2 0/16000000 0/16000000 6965230531272454161 ✗ ✓ ✗ ✗ OK
cluster-example-3 0/16000000 0/16000000 6965230531272454161 ✗ ✓ ✗ ✗ OK
~/D/c/w/yaml ❯❯❯ ll
Permissions Size User Date Modified Name
.rw-r--r-- 551 kotaroyoshino 23 5 06:26 cluster-example.yaml
~/D/c/w/yaml ❯❯❯ vi cluster-example.yaml

Psqlによる接続

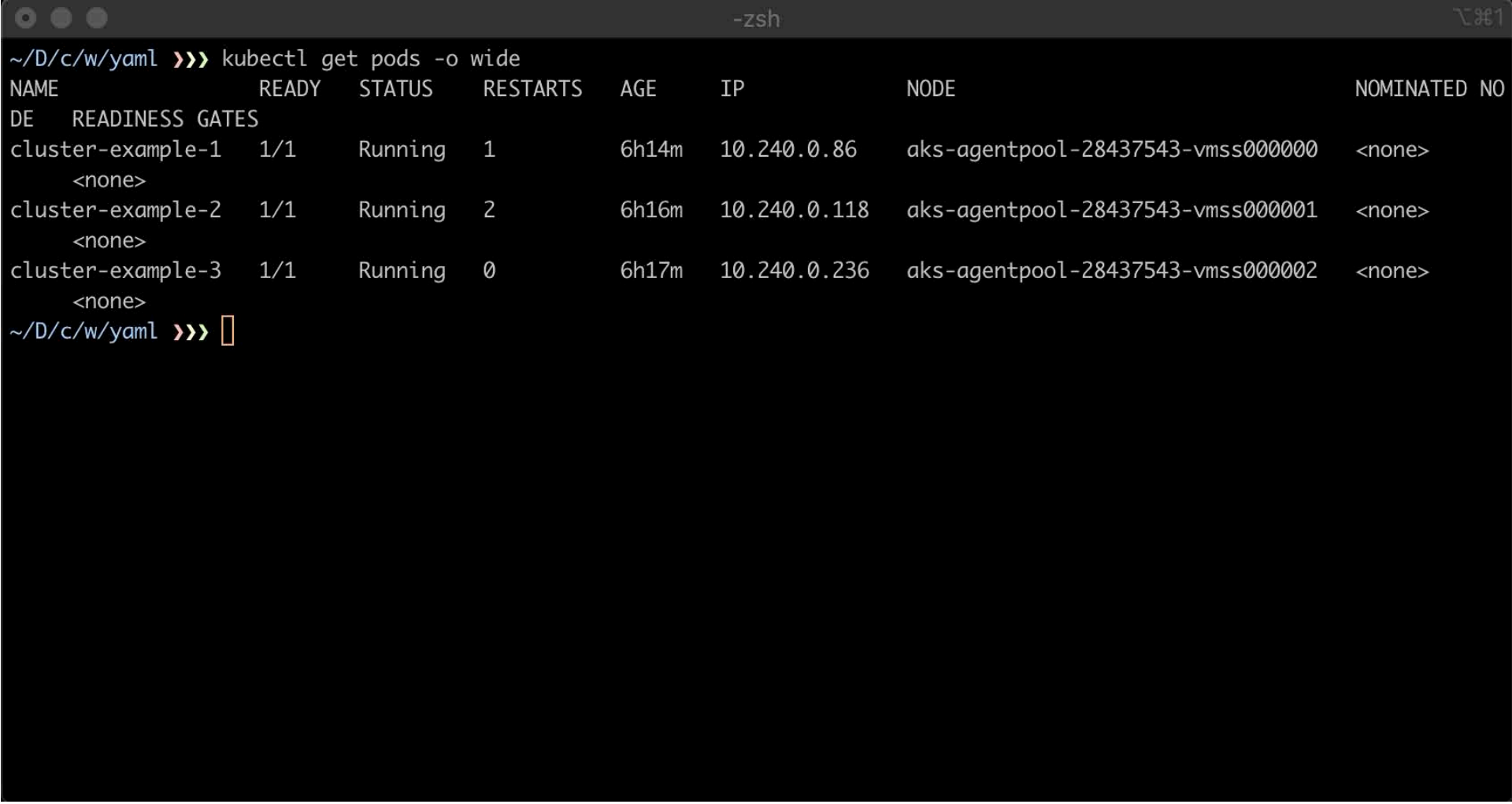
- Port-forwardを使用して、Kubernetesクラスタ内部IPへ接続（テスト用）
  kubectl port-forward service/cluster-example-rw 5454:5432
- Psqlによる接続
  psql -p 5454 -h 127.0.0.1 -U postgres

  ※パスワードはKubernetesが自動作成しているため、デコードする
  kubectl get secret cluster-example-superuser -oyaml -o=jsonpath={.data.password}|base64 -d


-zsh
⌥⌘2
-zsh
~/D/c/w/yaml ❯❯❯
-zsh
~/D/c/w/yaml ❯❯❯

Podの中のpgdata

- execを使用して、Podの中のpgdata配下を参照する
  kubectl exec --stdin --tty cluster-example-1 -- /bin/bash
  cd /var/lib/postgresql/data/pgdata
- 各種設定ファイル
  cat postgresql.conf
  cat pg_hba.conf
- ログファイル
  ls log/*


-zsh
~/D/c/w/yaml ❯❯❯ kubectl get pods -o wide
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NO
DE READINESS GATES
cluster-example-1 1/1 Running 1 6h14m 10.240.0.86 aks-agentpool-28437543-vmss000000 <none>
<none>
cluster-example-2 1/1 Running 2 6h16m 10.240.0.118 aks-agentpool-28437543-vmss000001 <none>
<none>
cluster-example-3 1/1 Running 0 6h17m 10.240.0.236 aks-agentpool-28437543-vmss000002 <none>
<none>
~/D/c/w/yaml ❯❯❯

【Kubernetes×PostgreSQL】をその手に！

最先端のPostgreSQLユーザーエクスペリエンスを
いち早くご体験ください！

- EDBのお問い合わせ先
  https://edbjapan.com/contact/
- Cloud Native PostgreSQLのマニュアル
  https://www.enterprisedb.com/docs/kubernetes/cloud_native_postgresql/