

Cloud Native PostgreSQL Kubernetes Ready!

セールスエンジニア 吉野孝太郎

2021年11月17日



目次

1. Cloud Native PostgreSQL

Kubernetesをフル活用したPostgreSQLのご紹介

2. Cloud Native PostgreSQL デモ

AWSとAzureでKubernetesクラスター作成

- Backup/Recovery
- CNP-Bench
- アプリケーションデモ (BFF)

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)



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, **14**
 - EDB Postgres Advanced 10, 11, 12, 13, **14**
 - BDR (3.6 on 2ndQPostgres in Q4/2021)

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、プログラマ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 1.17+**
 - OpenShift 4.6+
- Public Cloud
 - Microsoft Azure (AKS) – リリース済！
 - Amazon Web Services (EKS) - 2021/Q4 (10-12月期)
 - Google Cloud (GKS) - 2021/Q4 (10-12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>

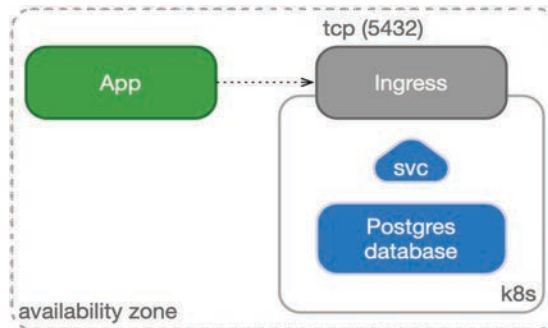
ユースケース

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



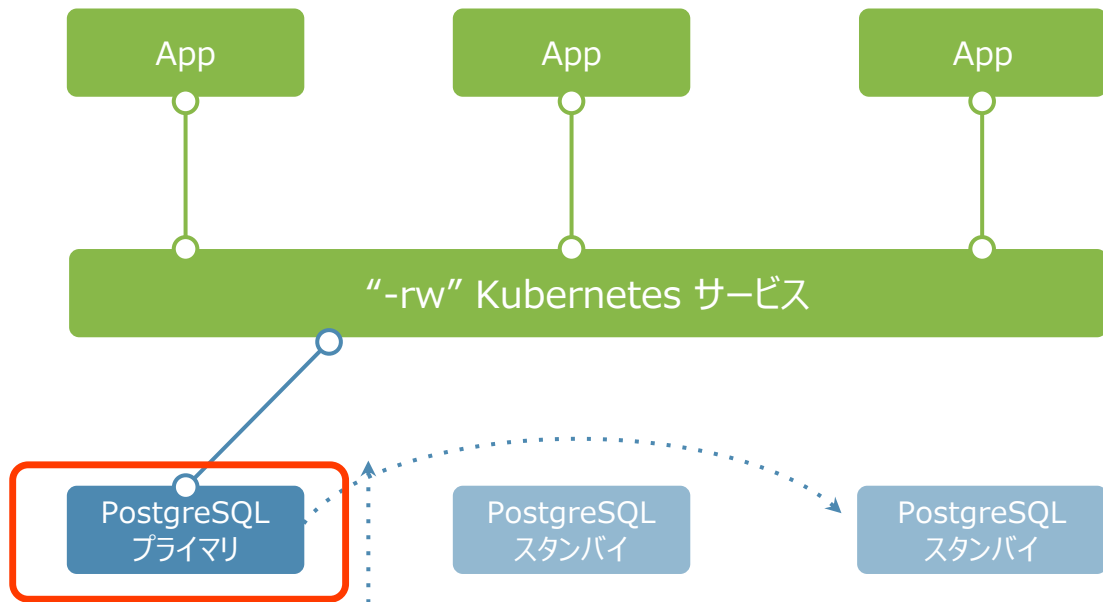
ケース2: レガシーシステムがKubernetesの外



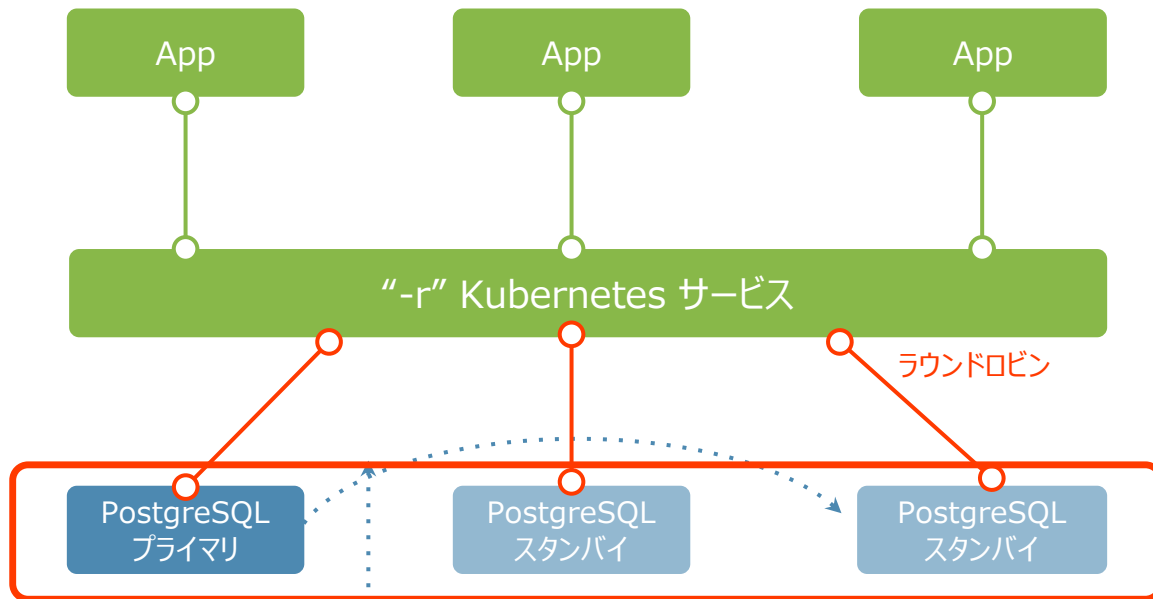
PostgreSQLクラスタ アーキテクチャ

- 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)

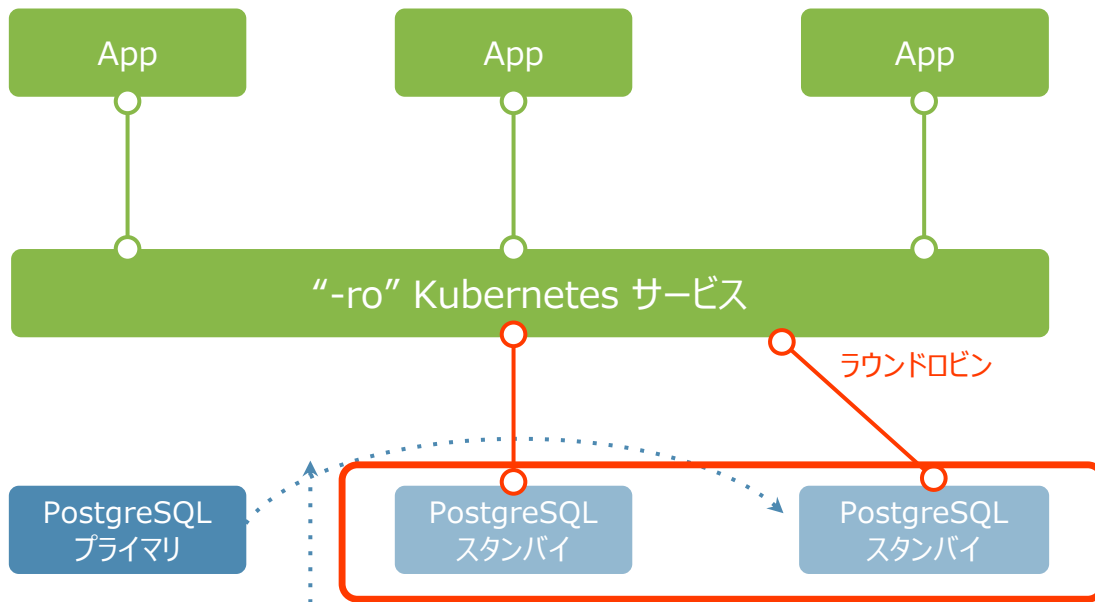
Read-writeワークロード(“-rw” service)



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
- ロールングアップデート
- 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(AWS S3、Azure Blob Storage)
 - Public clouds (AWS, Azure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

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最小（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
  :
```

※ $0 \leq \text{minSyncReplicas} \leq \text{maxSyncReplicas} < \text{instances}$

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>



Cloud Native PostgreSQL デモ



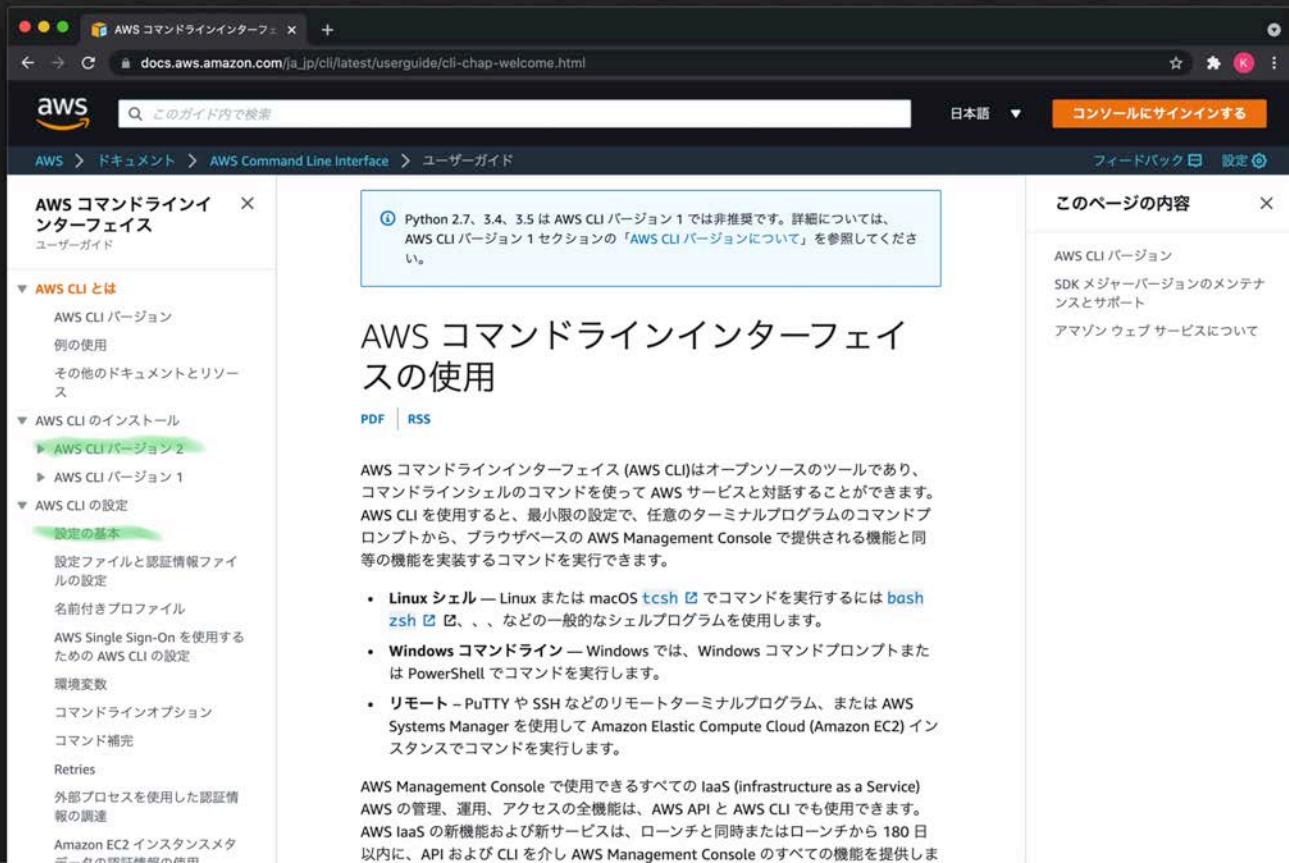
Quickstart

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、AWSと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



👉 AWS CLIのインストール
https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-chap-welcome.html



← Back

Cloud Native PostgreSQL

Before You Start

Free evaluation

Use cases

Architecture

Installation and upgrades

Quickstart

Install, Configure, Deploy Open

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:

minikube start

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

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

パート2 Operatorのインストール

EDB docs BETA

← Back

 Cloud Native PostgreSQL

Before You Start

Free evaluation

Use cases

Architecture

Installation and upgrades

Quickstart

Install, Configure, Deploy Open

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

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

↩ ⏪ 🔍 ⚙ ⚠ ⌵

ON THIS PAGE

Part 1 - Setup the local Kubernetes/OpenShift playground

Part 2 - Install Cloud Native PostgreSQL

Part 3 - Deploy a PostgreSQL cluster

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
```


パート3 PostgreSQLのデプロイ

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:

```
# 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
```

① There's more

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

ON THIS PAGE

Part 1 - Setup the local Kubernetes/OpenShift playground

Part 2 - Install Cloud Native PostgreSQL

Part 3 - Deploy a PostgreSQL cluster

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
```

~/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`

```
~/D/c/w/yaml >>> kubectl cnp status cluster-example
```

```
~/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	✓	x	x		OK
cluster-example-2		0/17000060	0/17000060	6965230531272454161	x	✓	x		OK
cluster-example-3		0/17000060	0/17000060	6965230531272454161	x	✓	x		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
```

```
~/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	✓	x	x		OK
cluster-example-2		0/16000000	0/16000000	6965230531272454161	x	✓	x		OK
cluster-example-3		0/16000000	0/16000000	6965230531272454161	x	✓	x		OK

```
~/D/c/w/yaml >>> ll
```

```
Permissions Size User      Date Modified Name
-rw-r--r--  551 kotaroyoshino 23 5 06:26 cluster-example.yaml
```

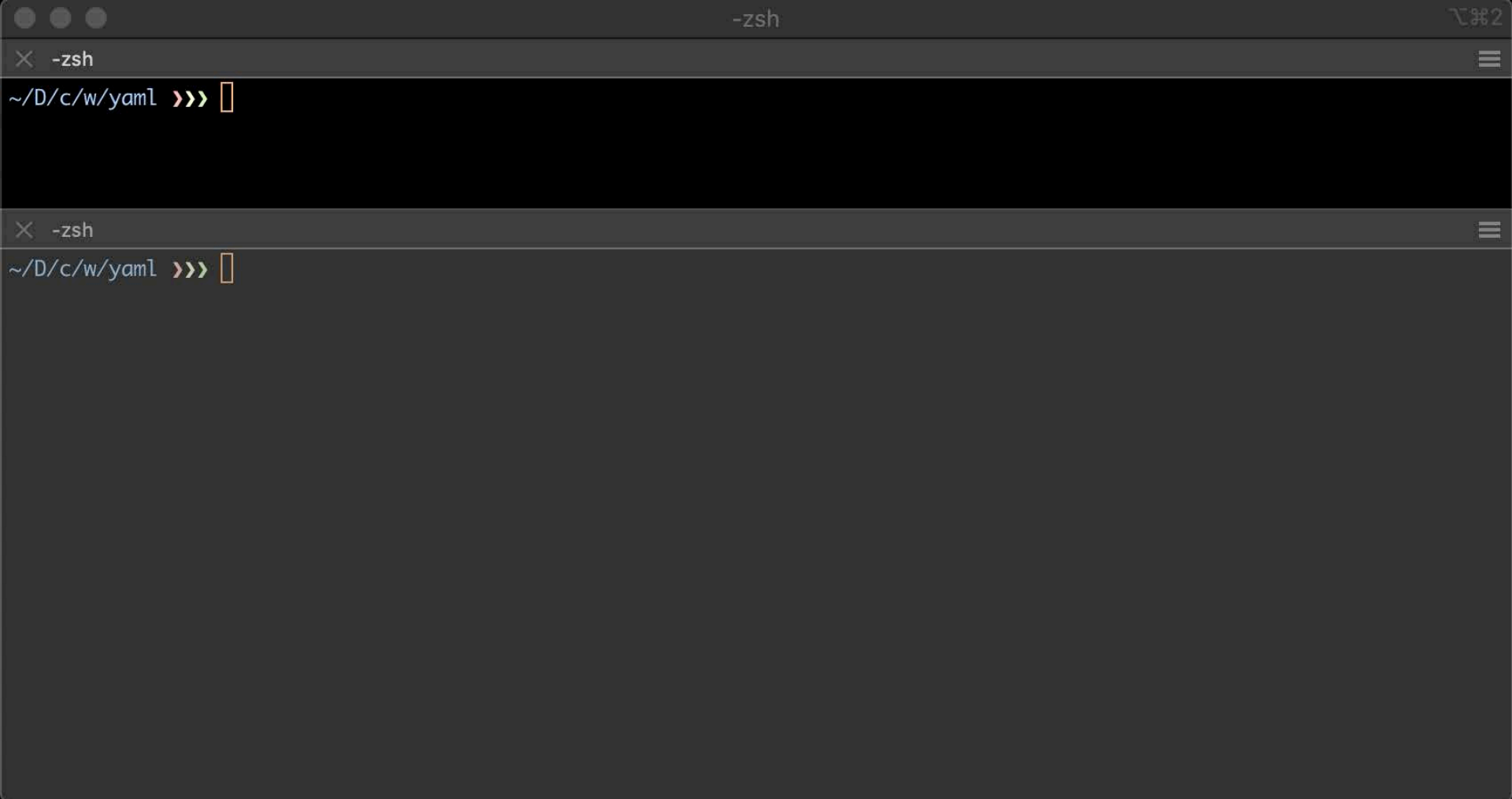
```
~/D/c/w/yaml >>> vi cluster-example.yaml
```

PsqI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）
`kubectI port-forward service/cluster-example-rw 5454:5432`
- PsqIによる接続
`psqI -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

`kubectI get secret cluster-example-superuser -oyaml -o=jsonpath={.data.password}}|base64 -d`



Podの中のpgdata

- execを使用して、Podの中のpgdata配下を参照する
kubect 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/*

```
~/D/c/w/yaml >>> kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NO
cluster-example-1	1/1	Running	1	6h14m	10.240.0.86	aks-agentpool-28437543-vmss000000	<none>
cluster-example-2	1/1	Running	2	6h16m	10.240.0.118	aks-agentpool-28437543-vmss000001	<none>
cluster-example-3	1/1	Running	0	6h17m	10.240.0.236	aks-agentpool-28437543-vmss000002	<none>

```
~/D/c/w/yaml >>> 
```

Backup & Recovery

- secretにAWSの認証情報を登録する
kubectrl create secret generic aws-creds ¥
--from-literal=ACCESS_KEY_ID=<access key here> ¥
--from-literal=ACCESS_SECRET_KEY=<secret key here>
--from-literal=ACCESS_SESSION_TOKEN=<session token here> # if required

※S3へのアクセス権限を持つユーザーであること

- Secretの確認
kubectrl describe secret/aws-creds

クラスタ (yaml) にBackupを設定

```
spec:
  :
  backup:
    barmanObjectStore:
      destinationPath: "s3://<BUCKET_NAME>/backup/"
      s3Credentials:
        accessKeyId:
          name: aws-creds
          key: ACCESS_KEY_ID
        secretAccessKey:
          name: aws-creds
          key: ACCESS_SECRET_KEY
```

Backupを実行（オンデマンド）

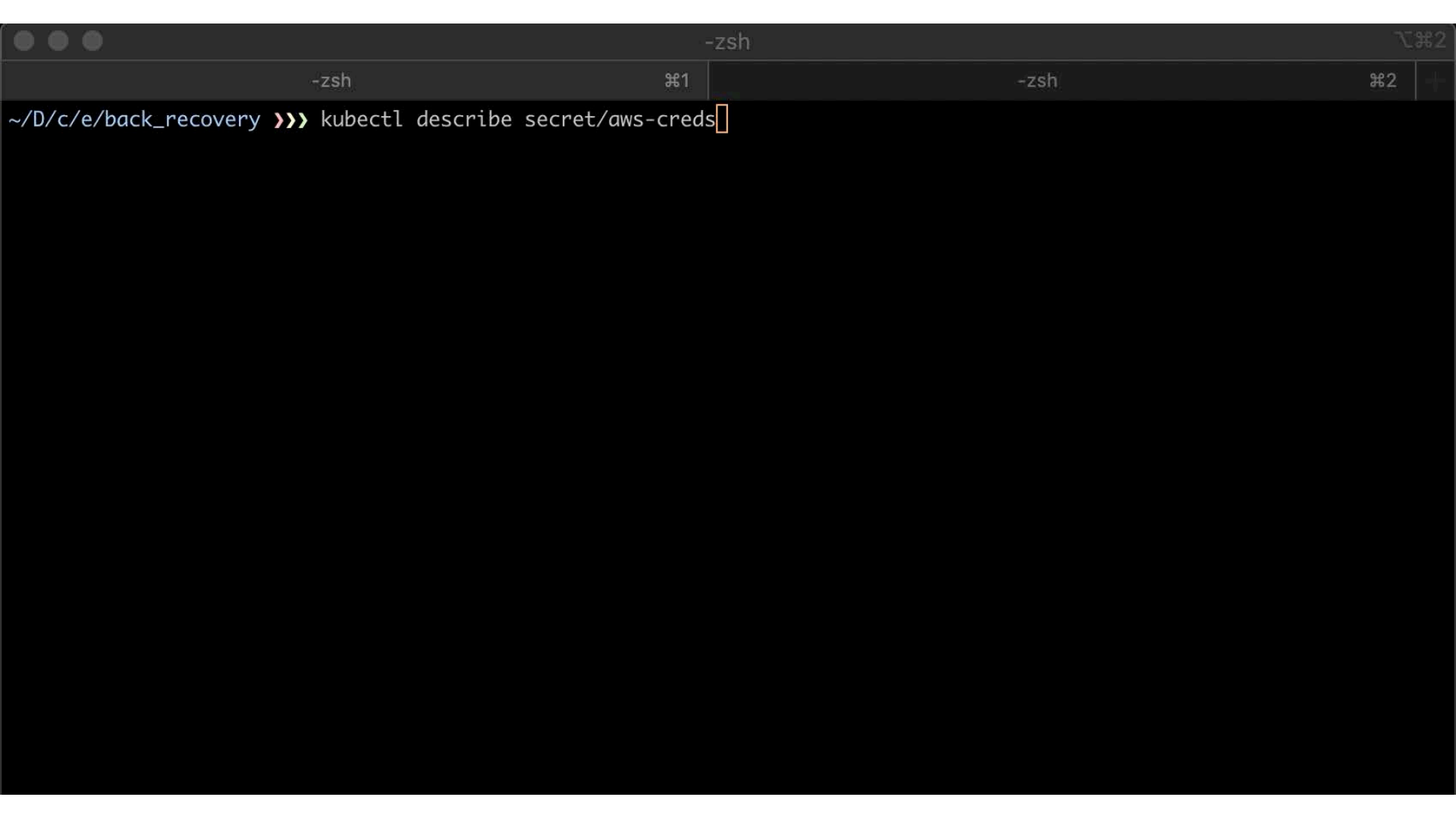
```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Backup
metadata:
  name: backup-example
spec:
  cluster:
    name: cluster-example
```

- Backupの実行と確認
 - kubectl apply -f backup-example.yaml
 - kubectl describe backup backup-example

Recoveryでクラスタを新規作成

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: cluster-recovery
spec:
  :
  bootstrap:
  recovery:
  backup:
    name: backup-example
```

- Recoveryの実行
kubect apply -f cluster-recovery.yaml



-zsh

zsh2

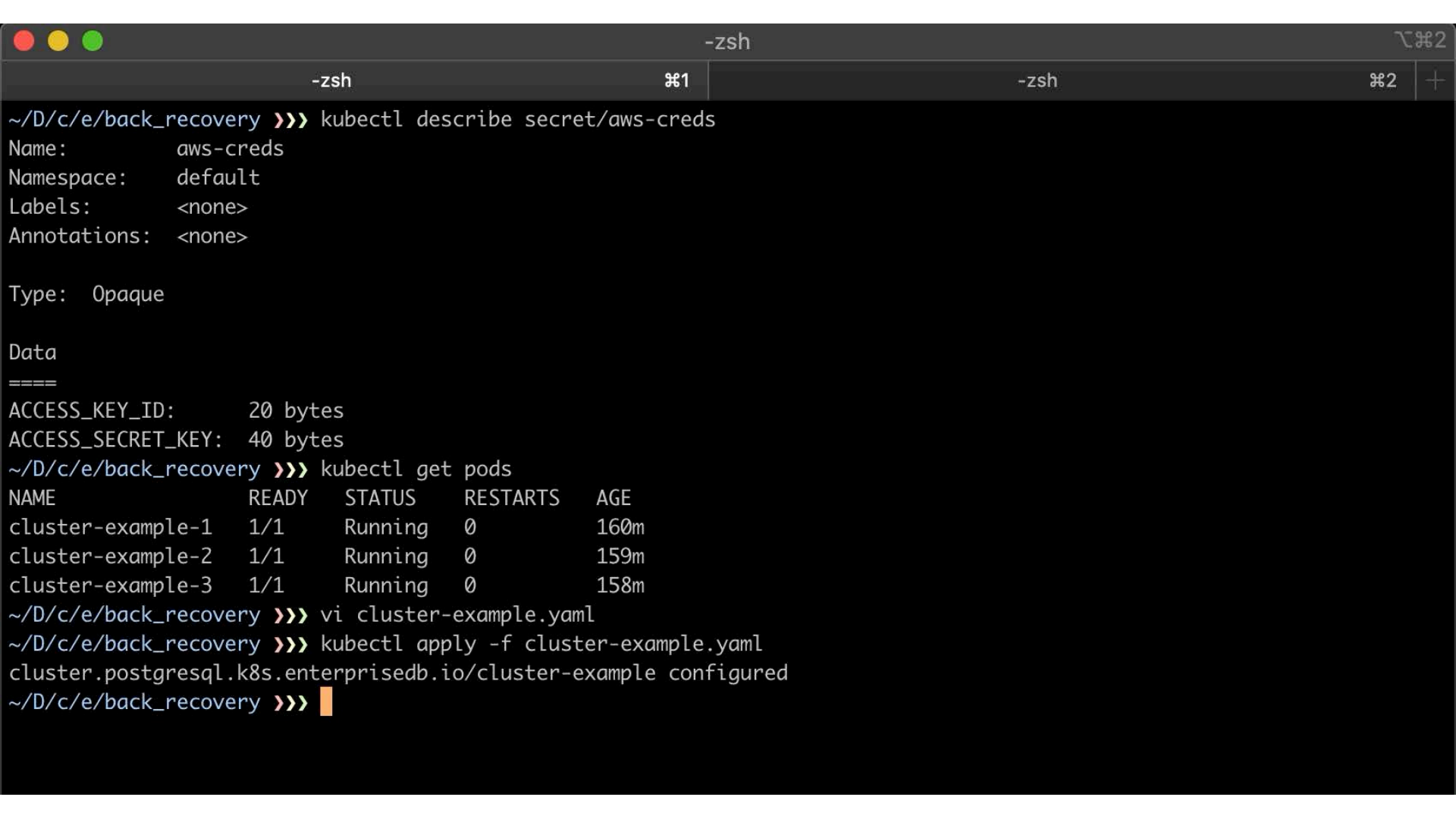
-zsh

zsh1

-zsh

zsh2

~/D/c/e/back_recovery >>> kubectl describe secret/aws-creds



-zsh

⌘2

-zsh

⌘1

-zsh

⌘2

+

~/D/c/e/back_recovery >>> kubectl describe secret/aws-creds

Name: aws-creds

Namespace: default

Labels: <none>

Annotations: <none>

Type: Opaque

Data

====

ACCESS_KEY_ID: 20 bytes

ACCESS_SECRET_KEY: 40 bytes

~/D/c/e/back_recovery >>> kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
cluster-example-1	1/1	Running	0	160m
cluster-example-2	1/1	Running	0	159m
cluster-example-3	1/1	Running	0	158m

~/D/c/e/back_recovery >>> vi cluster-example.yaml

~/D/c/e/back_recovery >>> kubectl apply -f cluster-example.yaml

cluster.postgresql.k8s.enterprisedb.io/cluster-example configured

~/D/c/e/back_recovery >>> █

(1 row)

```
postgres=# select * from countries;
```

```
country_code |      country_name
```

```
-----+-----
```

```
jpn          | Japan
```

```
usa          | United States of America
```

(2 rows)

```
postgres=# exit
```

```
~/D/c/e/back_recovery >>> cat backup-example.yaml
```

File: **backup-example.yaml**

```
1  apiVersion: postgresql.k8s.enterprisedb.io/v1
```

```
2  kind: Backup
```

```
3  metadata:
```

```
4    name: backup-example
```

```
5  spec:
```

```
6    cluster:
```

```
7      name: cluster-example
```

```
~/D/c/e/back_recovery >>> kubectl apply -f backup-example.yaml
```

```
backup.postgresql.k8s.enterprisedb.io/backup-example created
```

```
~/D/c/e/back_recovery >>> 
```





-zsh

⌘2

-zsh

⌘1

-zsh

⌘2

+

~/D/c/e/back_recovery >>> kubectl get pods

Pg_basebackupによるコピー作成

- bootstrapでpg_basebackupを使用して、DBクラスタをコピー作成
 - 同じメジャーバージョンのPostgresqlから、コピー可（Kubernetes内／外どちらでも）
 - ストリーミングレプリケーション経由（要、REPLICATION LOGIN privileges）
 - コピー元DBにログイン可能であること（md5、クライアント認証をサポート）
- bootstrapを設定して、kubectl apply
kubectl apply -f cluster-copy.yaml

```
# Require 1Gi of space
```

```
storage:
```

```
  size: 10G
```

```
bootstrap:
```

```
  pg_basebackup:
```

```
    source: cluster-example-db
```

```
#右へ続き
```

```
#左から続き
```

```
externalClusters:
```

```
- name: cluster-example-db
```

```
  connectionParameters:
```

```
    host: cluster-example-rw.default.svc
```

```
    user: streaming_replica
```

```
    sslmode: verify-full
```

```
  sslKey:
```

```
    name: cluster-example-replication
```

```
    key: tls.key
```

```
  sslCert:
```

```
    name: cluster-example-replication
```

```
    key: tls.crt
```

```
  sslRootCert:
```

```
    name: cluster-example-ca
```

```
    key: ca.crt
```

~/D/c/e/pg_basebackup >>> ll

<u>Permissions</u>	<u>Size</u>	<u>User</u>	<u>Date Modified</u>	<u>Name</u>
.rw-r--r--@	696	kotaroyoshino	18 7 03:28	cluster-copy.yaml

~/D/c/e/pg_basebackup >>>



-zsh

2

-zsh

1

-zsh

2

+

~/D/c/e/pg_basebackup >>> kubectl get pods

Cnp-bench

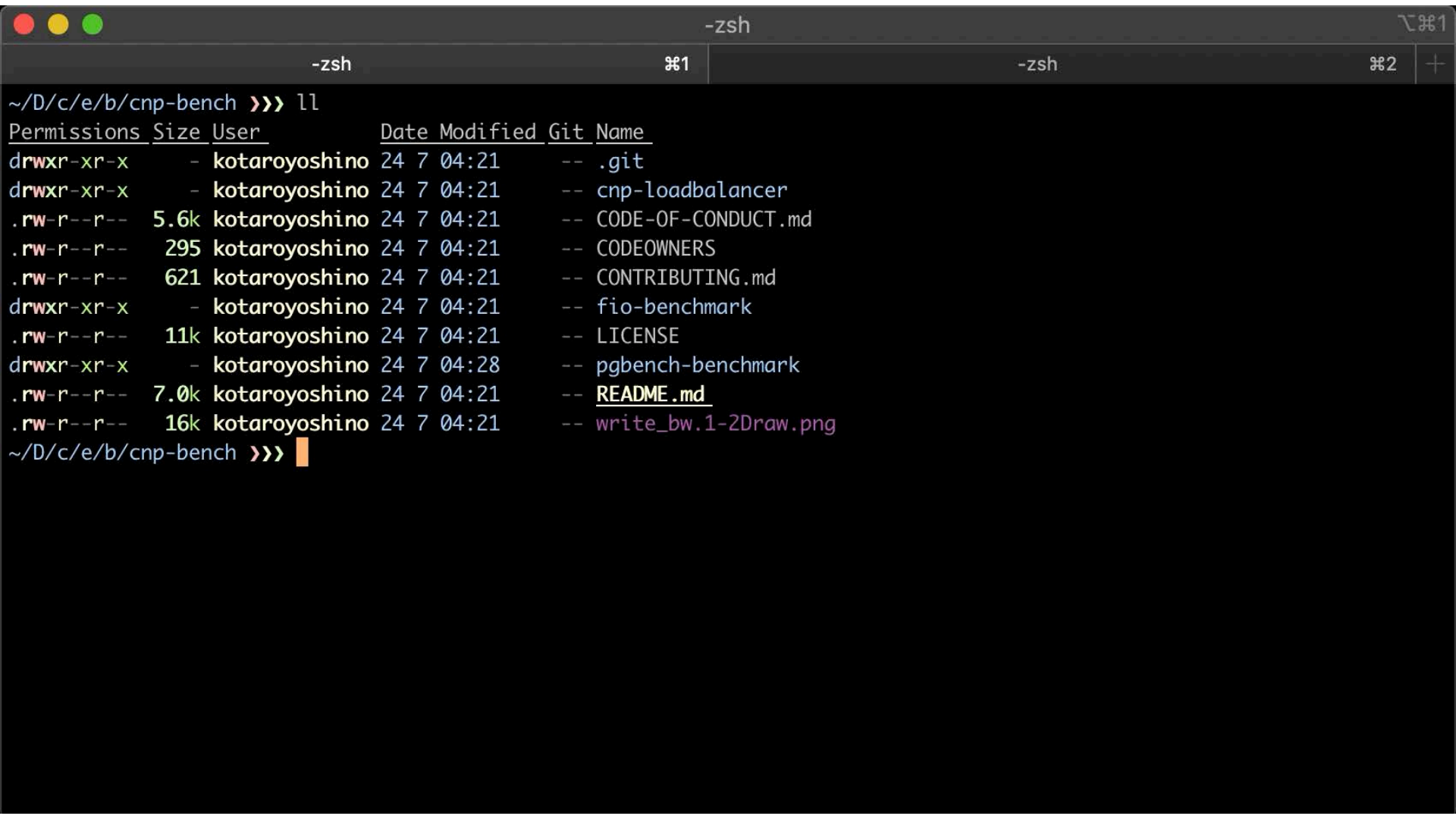
- CNP向け、ベンチマークのガイドライン
 - ストレージベンチ : fio
 - データベースベンチ : pgbench
- Helm Charts使用
Helmのインストール
`brew install helm` (Mac用)
`helm version`
- Open Source : Apache 2.0 license
<https://github.com/EnterpriseDB/cnp-bench>



-zsh

⌘⌘1

~/D/c/e/bench >>> git clone https://github.com/EnterpriseDB/cnp-bench.git



-zsh

⌘1

-zsh

⌘1

-zsh

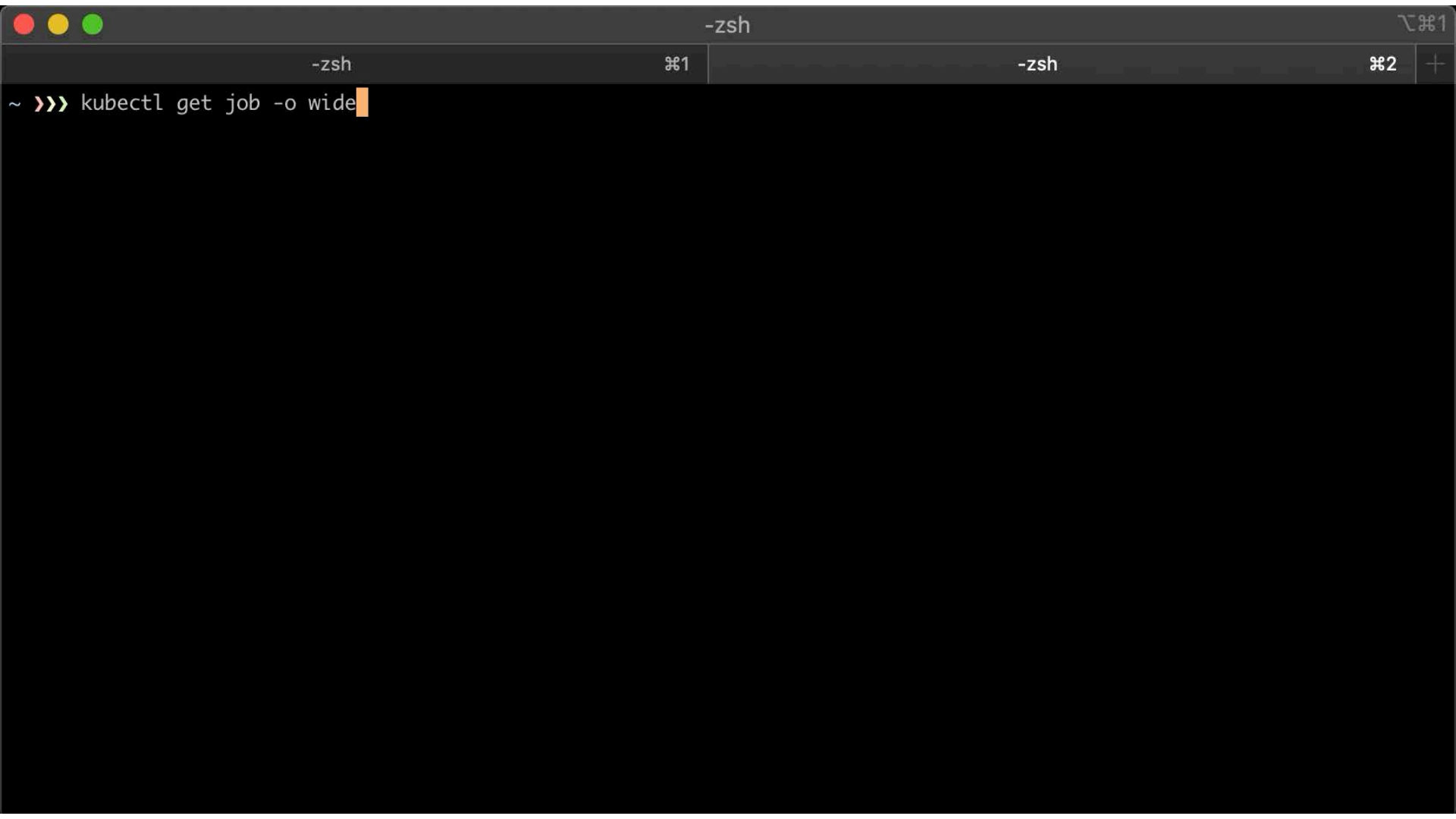
⌘2

+

~/D/c/e/b/cnp-bench >>> ll

Permissions	Size	User	Date	Modified	Git	Name
drwxr-xr-x	-	kotaroyoshino	24 7	04:21	--	.git
drwxr-xr-x	-	kotaroyoshino	24 7	04:21	--	cnp-loadbalancer
.rw-r--r--	5.6k	kotaroyoshino	24 7	04:21	--	CODE-OF-CONDUCT.md
.rw-r--r--	295	kotaroyoshino	24 7	04:21	--	CODEOWNERS
.rw-r--r--	621	kotaroyoshino	24 7	04:21	--	CONTRIBUTING.md
drwxr-xr-x	-	kotaroyoshino	24 7	04:21	--	fio-benchmark
.rw-r--r--	11k	kotaroyoshino	24 7	04:21	--	LICENSE
drwxr-xr-x	-	kotaroyoshino	24 7	04:28	--	pgbench-benchmark
.rw-r--r--	7.0k	kotaroyoshino	24 7	04:21	--	<u>README.md</u>
.rw-r--r--	16k	kotaroyoshino	24 7	04:21	--	write_bw.1-2Draw.png

~/D/c/e/b/cnp-bench >>> █



-zsh

1

-zsh

1

-zsh

2

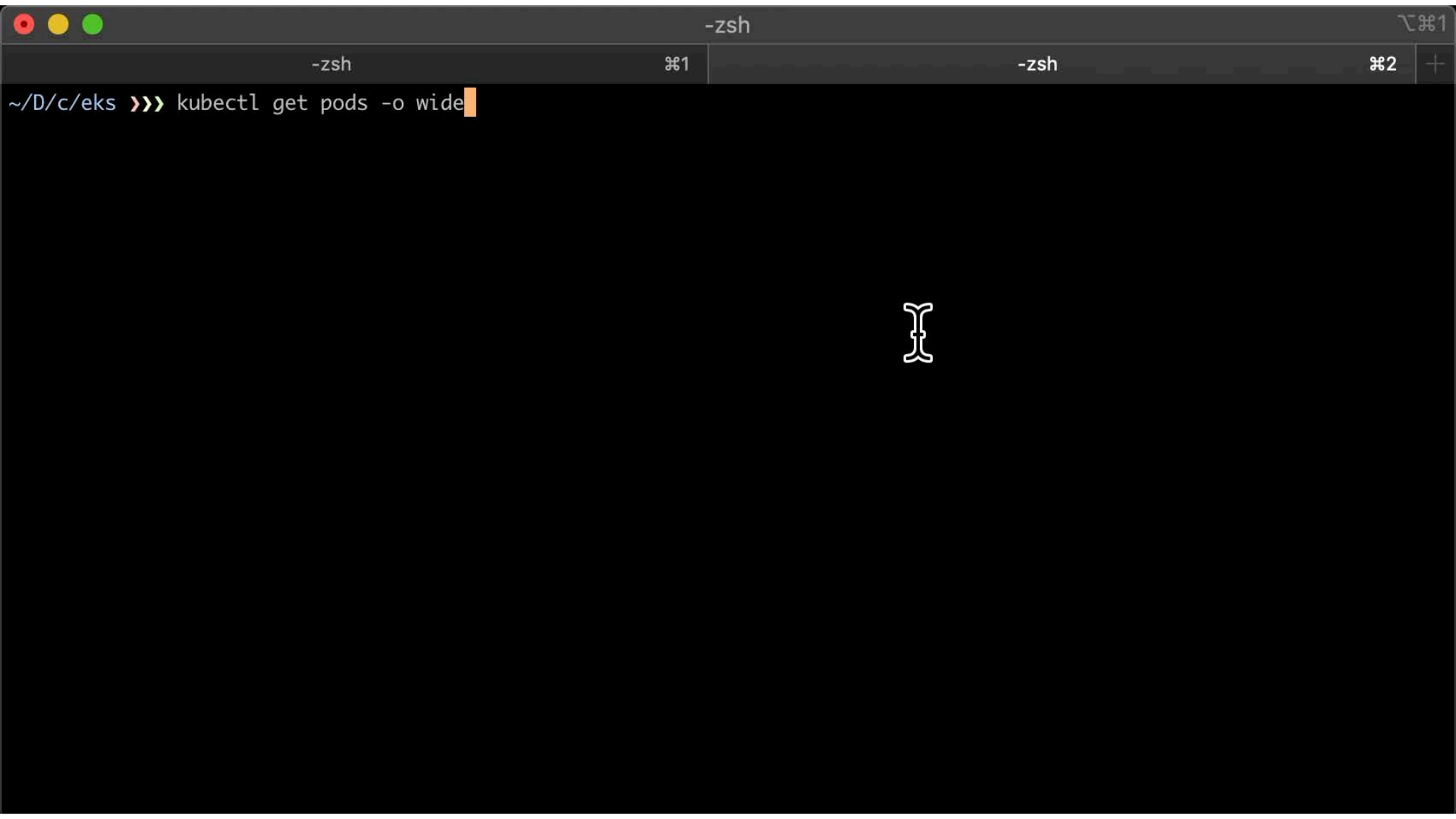
+

~ >>> kubectl get job -o wide

~/D/c/e/b/cnp-bench >>> ll

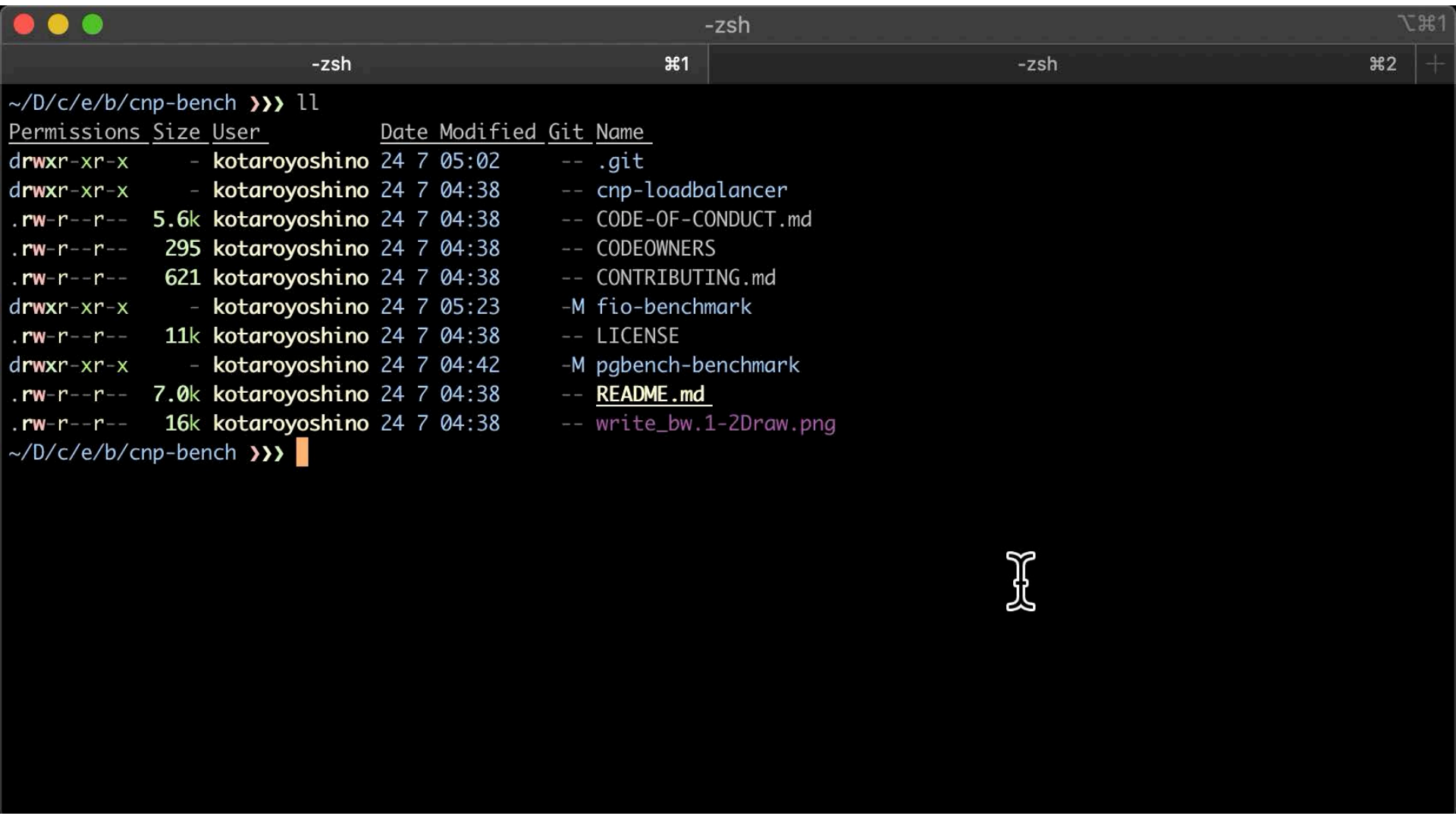
Permissions	Size	User	Date	Modified	Git	Name
drwxr-xr-x	-	kotaroyoshino	24 7	05:02	--	.git
drwxr-xr-x	-	kotaroyoshino	24 7	04:38	--	cnp-loadbalancer
.rw-r--r--	5.6k	kotaroyoshino	24 7	04:38	--	CODE-OF-CONDUCT.md
.rw-r--r--	295	kotaroyoshino	24 7	04:38	--	CODEOWNERS
.rw-r--r--	621	kotaroyoshino	24 7	04:38	--	CONTRIBUTING.md
drwxr-xr-x	-	kotaroyoshino	24 7	05:02	--	fio-benchmark
.rw-r--r--	11k	kotaroyoshino	24 7	04:38	--	LICENSE
drwxr-xr-x	-	kotaroyoshino	24 7	04:42	-M	pgbench-benchmark
.rw-r--r--	7.0k	kotaroyoshino	24 7	04:38	--	<u>README.md</u>
.rw-r--r--	16k	kotaroyoshino	24 7	04:38	--	write_bw.1-2Draw.png

~/D/c/e/b/cnp-bench >>> □



アプリケーションデモ（BFF）

- PostgreSQL on Kubernetes × サーバーレス（Lambda）で、マイクロサービス実装検証
- APIゲートウェイとして、AWS APIゲートウェイ
- 実行環境としてAWS Lambda
 - サーバーレスで突然のアクセス激増にも、自動的にスケール
- データストアとして、PostgreSQL on Kubernetes
 - KubernetesのPodリソース管理で、フレキシブルにCPU、メモリを割り当て可能



~/D/c/e/b/cnp-bench >>> ll

Permissions	Size	User	Date Modified	Git	Name
drwxr-xr-x	-	kotaroyoshino	24 7 05:02	--	.git
drwxr-xr-x	-	kotaroyoshino	24 7 04:38	--	cnp-loadbalancer
.rw-r--r--	5.6k	kotaroyoshino	24 7 04:38	--	CODE-OF-CONDUCT.md
.rw-r--r--	295	kotaroyoshino	24 7 04:38	--	CODEOWNERS
.rw-r--r--	621	kotaroyoshino	24 7 04:38	--	CONTRIBUTING.md
drwxr-xr-x	-	kotaroyoshino	24 7 05:23	-M	fio-benchmark
.rw-r--r--	11k	kotaroyoshino	24 7 04:38	--	LICENSE
drwxr-xr-x	-	kotaroyoshino	24 7 04:42	-M	pgbench-benchmark
.rw-r--r--	7.0k	kotaroyoshino	24 7 04:38	--	<u>README.md</u>
.rw-r--r--	16k	kotaroyoshino	24 7 04:38	--	write_bw.1-2Draw.png

~/D/c/e/b/cnp-bench >>>



```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
~ /D/c/eks >>> kubectl get services
NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP
cluster-example-any                 ClusterIP    10.100.126.15    <none>
cluster-example-loadbalancer         LoadBalancer 10.100.41.69     pe649a21eac2e43b0a6eb66315d0bcd7-cb2a931d4aad788.elb.ap-n
cluster-example-r                    ClusterIP    10.100.250.7     <none>
cluster-example-ro                   ClusterIP    10.100.165.92    <none>
cluster-example-rw                   ClusterIP    10.100.26.175    <none>
kubernetes                           ClusterIP    10.100.0.1       <none>
```

- 専用ホスト
- キャパシティの予約
- イメージ
- AMI
- ELASTIC BLOCK STORE
- ボリューム
- スナップショット
- ライフサイクルマネージャー
- ネットワーク & セキュリティ
- セキュリティグループ
- Elastic IP
- プレースメントグループ
- キーペア
- ネットワークインターフェイス
- ロードバランシング
- ロードバランサー

① AWS Launch Wizard for SQL Server を使用すると、Microsoft SQL Server Always On 可用性グループのサイズ調整、設定、デプロイを簡単に行うことができます。詳細はこちら

インスタンスを起動

開始するには、クラウド内の仮想サーバーである Amazon EC2 インスタンスを起動します。

インスタンスを起動

注意: インスタンスは アジアパシフィック (東京) リージョンで起動されます

予定されているイベント

アジアパシフィック (東京)

予定されているイベントはありません

アカウントの属性

サポートされているプラットフォーム

VPC

デフォルト VPC

vpc-ba9c93dd

設定

EBS 暗号化

ゾーン

EC2 シリアルコンソール

デフォルトのクレジット仕様

コンソールの実験

追加情報

入門ガイド

ドキュメント

すべての EC2 リソース

フォーラム

料金

お問い合わせ



AWS マネジメントコンソール

AWS のサービス

▼ 最近アクセスしたサービス

- | | | |
|----------------------------|----------------|---------------------------|
| API Gateway | S3 | Certificate Manager |
| Lambda | CloudFormation | Elastic Container Service |
| EC2 | CloudWatch | Billing |
| VPC | RDS | AWS Organizations |
| Elastic Kubernetes Service | IAM | Route 53 |

▶ すべてのサービス

外出先でも AWS リソースへの接続を維持



AWS コンソールモバイルアプリで 4 つの追加リージョンがサポートされるようになりました。AWS コンソールモバイルアプリを iOS または Android モバイルデバイスにダウンロードしてください。

[詳細はこちら](#)

AWS の詳細

Amazon Redshift

費用効果が高く、高速でシンプルなデータウェアハウスで、クエリをデータレイクに拡張できます。

[詳細はこちら](#)

AWS Fargate でのサーバーレスコンテナの実行

AWS Fargate を使用して、サーバーやクラスターを管理することなくコンテナを実行およびスケールできます。

[詳細はこちら](#)

Amazon S3 によるスケーラブル、堅牢、安全なバックアップと復元

コスト削減を実現するバックアップおよび復元ソリューションを AWS で構築しているお客様の事例をご覧ください。

[詳細はこちら](#)

AWS Marketplace

ソリューションの構築

シンプルなウィザードと自動化されたワークフローで開始する。

仮想マシンの起動

EC2
2 ~ 3 分



ウェブアプリケーションの構築

仮想サーバーを使用した構築
Elastic Beanstalk
6 分



Lightsail
1 ~ 2 分





ロードマップ

Q4

コネクションプーリングのサポート

ロジカルレプリケーションのサポート

双方向レプリケーションのサポート

オペレーターによるメジャーバージョンアップ

追加でサポートされるプラットフォーム:

- GKE
- Linux on Power
- Linux on Z

マーケットプレイスの可用性

- RedHat
- 米国政府-国防総省

2022年以降

異なるデータセンターのk8sクラスタ
一間での双方向レプリケーション

すでに利用可能なデフォルトのメトリックに基づくサンプルのGrafanaダッシュボード

オペランドのCPU / メモリ使用量 (CPU / メモリ) を報告するオペレーター

スマートなデフォルトと自動チューニング

pgAdminオペレーター

【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
英語版
https://www.enterprisedb.com/docs/kubernetes/cloud_native_postgresql/
日本語版（ユーザ登録の必要あり）
<https://edbjapan.com/manual/cnp-210509/>