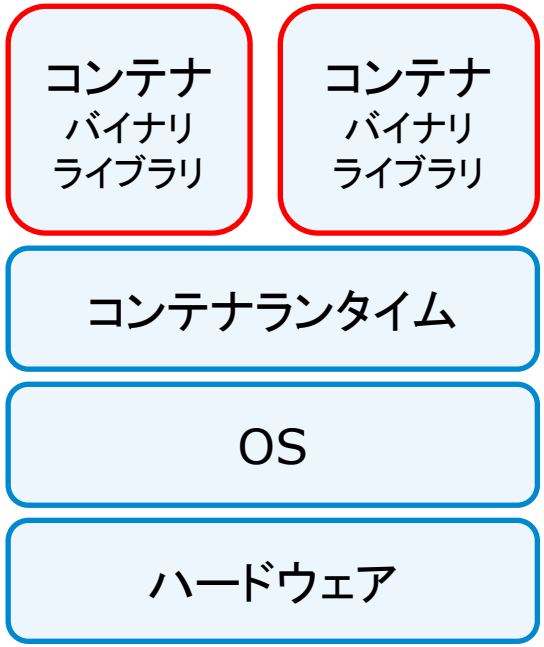
SRA OSS, INC.

OSS のコンテナオーケストレーションツール Kubernetes のご紹介

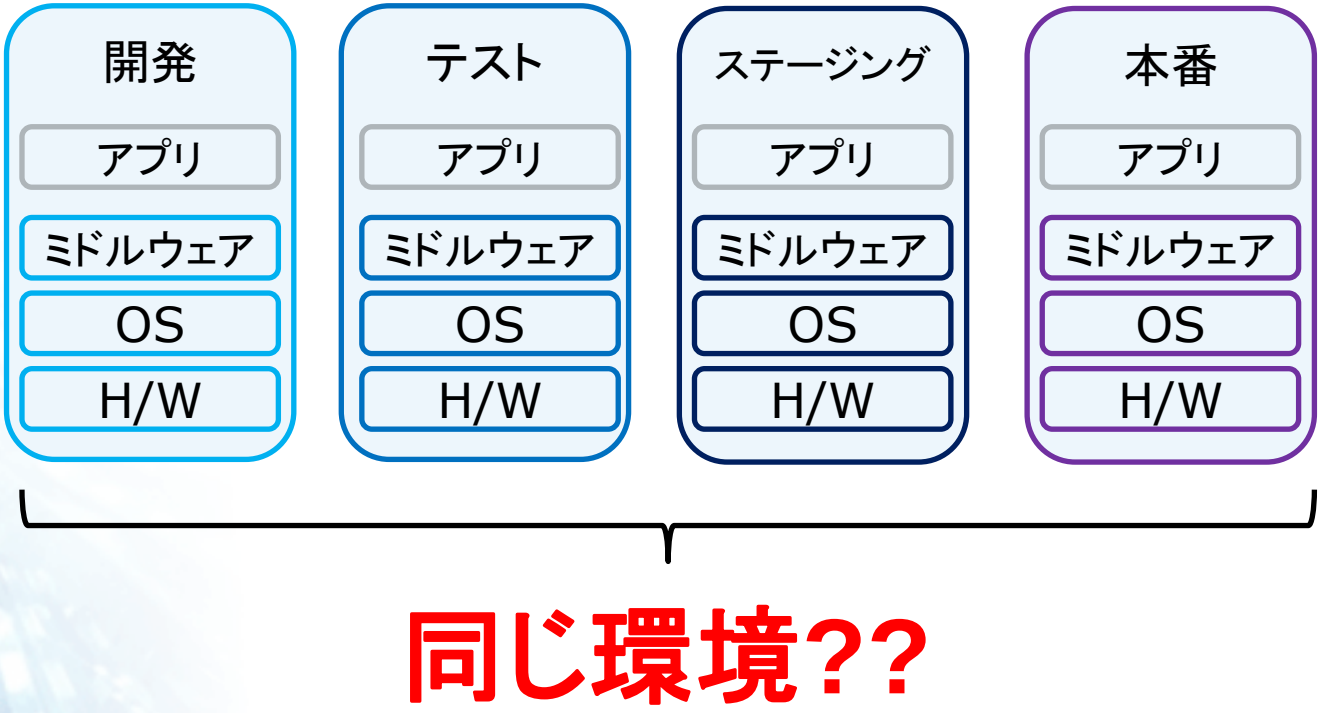
コンテナとは

- ホストOS上に論理的な区画(コンテナ)を作ってアプリケーションを動作させるために必要なプログラムなどをまとめたもの
- ホストOS上のリソースをコンテナで共有して利用
- サーバ仮想化に対してオーバヘッドが少なく
- 複数のコンテナを組み合わせて一つのアプリケーションを作るマイクロサービスとの相性がよい

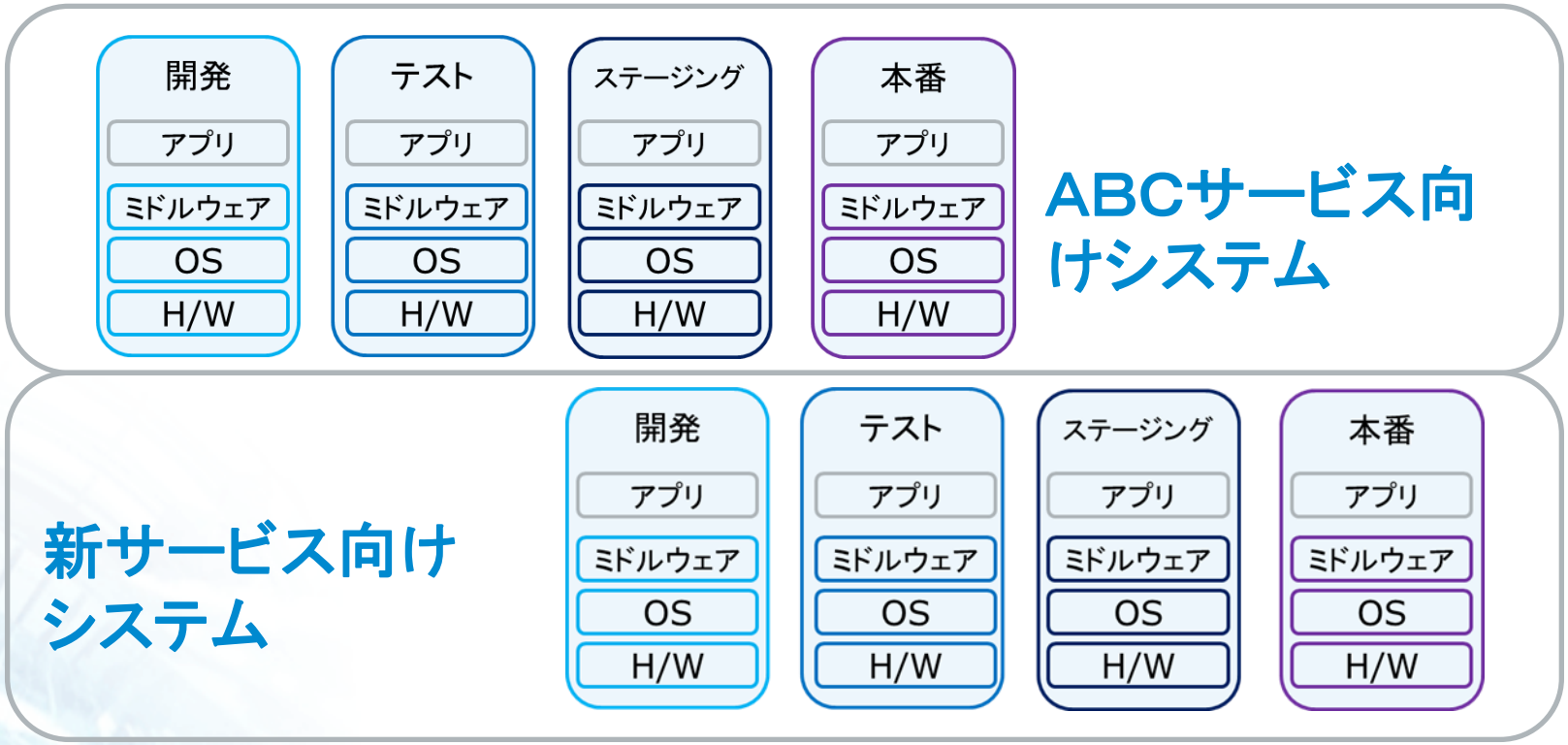


コンテナを利用しないアプリ開発の問題点

- 開発環境・テスト環境・ステージング環境・本番環境など、複数の環境でインフラ・ミドルウェアの違いに依存する問題が発生しやすい



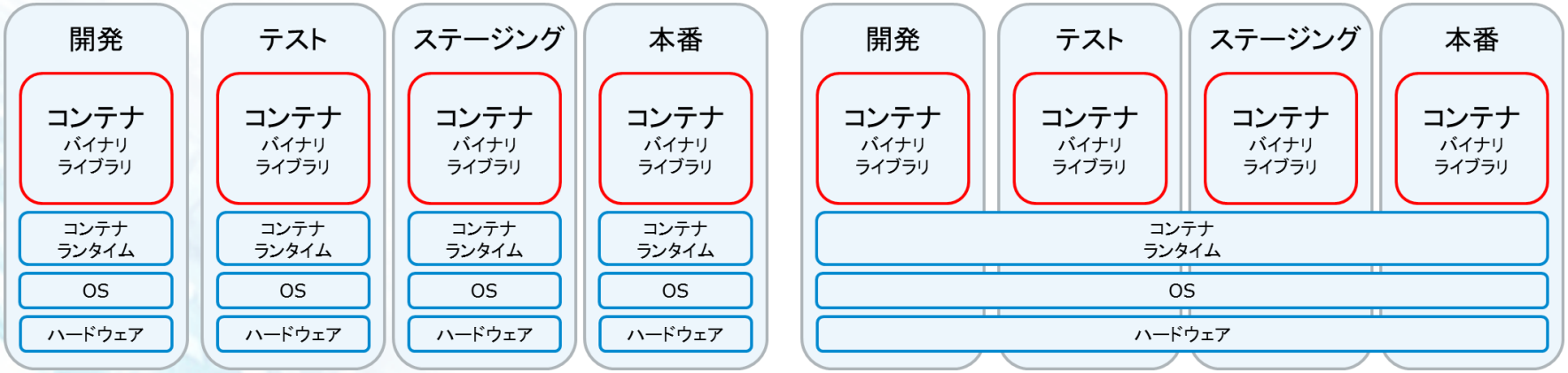
- サービスが増えると開発環境・テスト環境・ステージング環境・本番環境も増える・・



インフラの準備時間とテスト実施のための時間がかかってしまって迅速な開発・サービス提供ができなくなってしまう

コンテナを利用したアプリ開発

- コンテナランタイム上で同じイメージを動かすことで環境の違いのリスクを減らすことができる
- コンテナ間は独立しているので、同じホスト上でも動作させることができる
- サーバ仮想化に対してオーバヘッドが少ないため、サーバの利用効率が上昇する



- コンテナの構成ファイルをGitリポジトリで管理・共有できる
- コンテナのイメージをリポジトリで管理・共有できる
- 継続的インティグレーション／継続的デリバリー（CI・CD）を実施しやすく、アプリケーション開発が効率的にできる

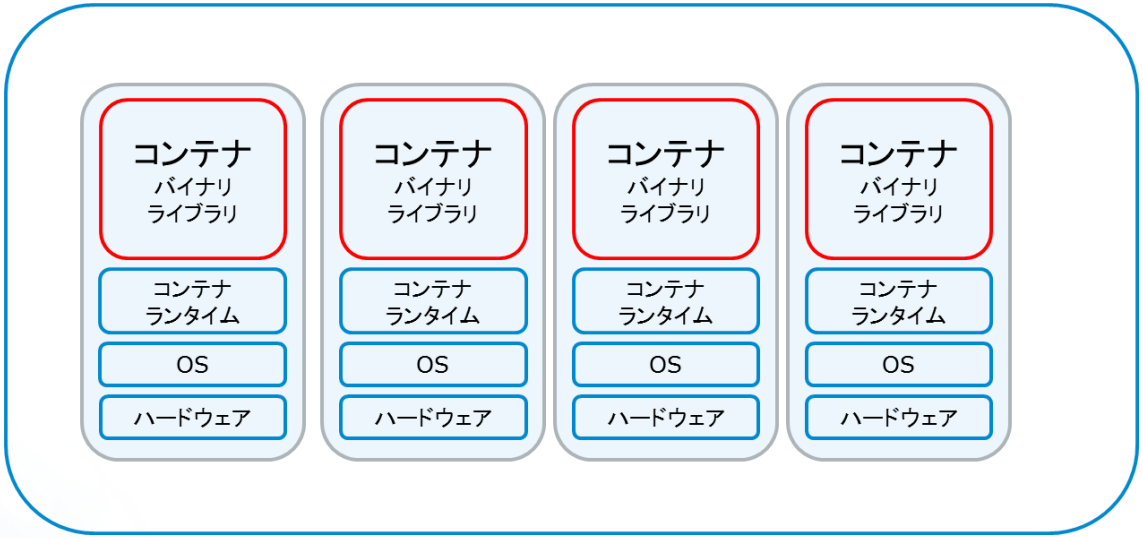
CI/CD

アプリケーションソースコードの変更がされたら即座にコンテナを作成して、テスト環境でテストして本番環境にリリース可能な状況にしておく

生産性の向上や環境管理の負荷軽減が期待できる

コンテナオーケストレーション

- コンテナを動かす際には、可用性・性能を考慮して複数のホストマシンから構成されるクラスタが必要



コンテナを管理するための クラスタプラットフォーム

2025年の崖(経済産業省より)

https://www.meti.go.jp/shingikai/mono_info_service/digital_transformation/20180907_report.html

kubernetesとは

- Googleでの10年以上の運用経験をもとに設計されたシステムをオープンソース化(2014~)
- CNCFにより開発・メンテナンス
  - ベンダーフリー・認定システム間で互換性が高い
  - kubernetesを中心としたエコシステムの整備
    - Prometheus – fluentd

https://www.cncf.io/certification/software-conformance/

- 大規模クラスタをサポート

https://kubernetes.io/docs/setup/best-practices/cluster-large/

項目	サポート対象
ノード数	5000台以下
Pod数	150,000以下
コンテナ数	300,000以下
1ノードあたりのpod数	100以下

Kubernetes リリースサイクル

- Kubernetes X.Y.Z
  - X メジャーバージョン
  - Y マイナーバージョン
  - Z パッチバージョン
  - 最新 1.16.2 (11/4時点)
- マイナーバージョンのリリースサイクル 約3カ月
- 3つのマイナーバージョンをメンテナンス

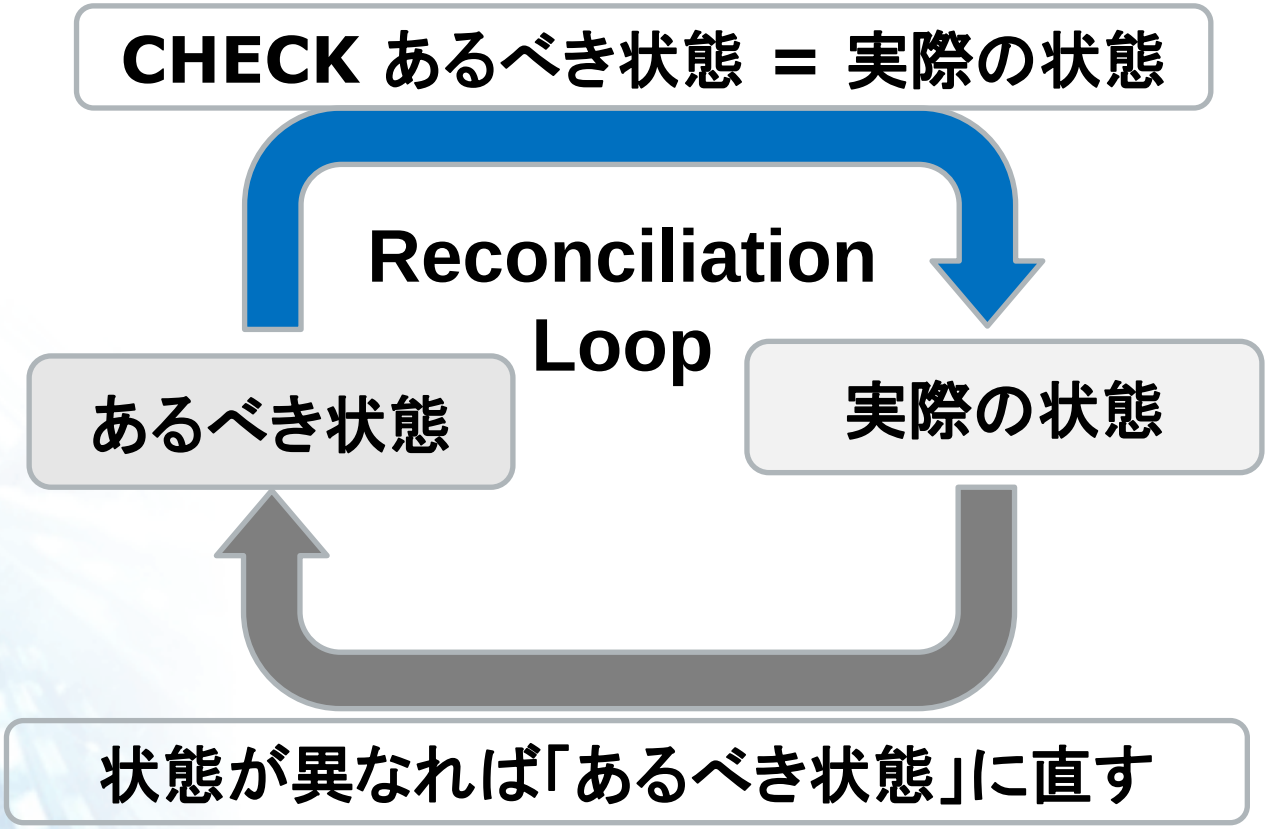
一つのマイナーバージョンは

約9か月間のみメンテナンスされる

https://kubernetes.io/docs/setup/release/version-skew-policy/

kubernetesの大きな特徴

- ユーザが宣言的に「あるべき状態」を定義
- Kubernetesは「あるべき状態」になるように「自律的」に働く



あるべき状態

- 宣言1：Nginx を3つ動かす

  1つのNginxに障害が発生して、合計2つしか動いていない場合には新規で起動

  ⇒ 障害の自動修復

- 宣言2: Nginxの平均CPU利用率が50%になるようにNginxを自動的に増やす

  ⇒ 水平オートスケール

- 宣言3: ノードのリソースが不足したらノードを追加

  ⇒ ノードの水平オートスケール

あるべき状態

- 宣言1：Nginx を3つ動かす

  Nginxが2つしか動いていない場合には新規で起動

  ⇒ 可用性

Kubernetesの特徴

「あるべき状態」になるように
「自律的」に働く

⇒ ノードの水平オートスケール

YAMLによる構成管理

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-master
  labels:
    app: redis
spec:
  selector:
    matchLabels:
      app: redis
      role: master
      tier: backend
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: master
        tier: backend
    spec:
      containers:
      - name: master
        image: redis:5.0.4
        resources:
          requests:
                cpu: 100m
                memory: 100Mi
        ports:
        - containerPort: 6379
```

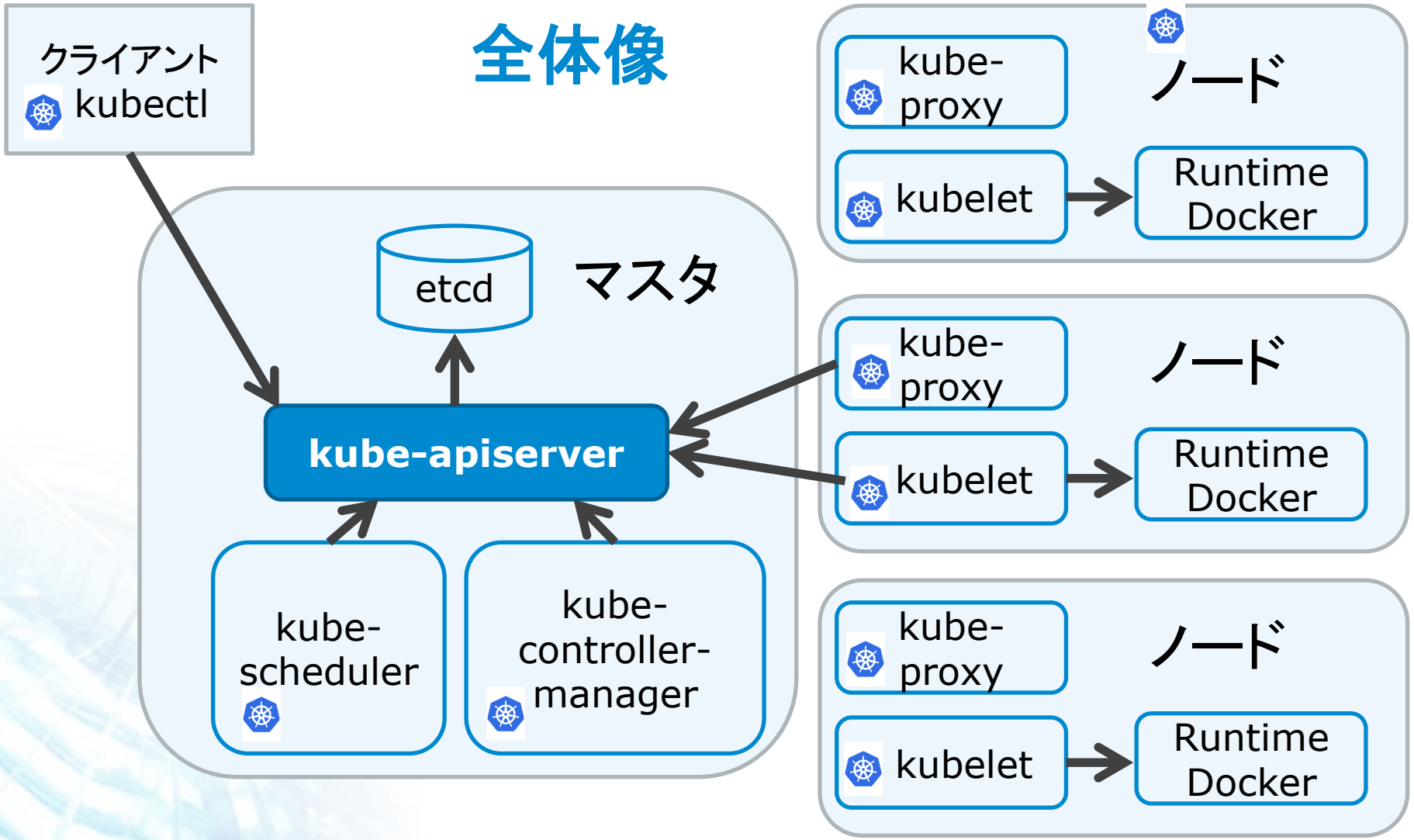
構成をファイルで定義できる
Githubなどで管理しやすい

複製数の指定

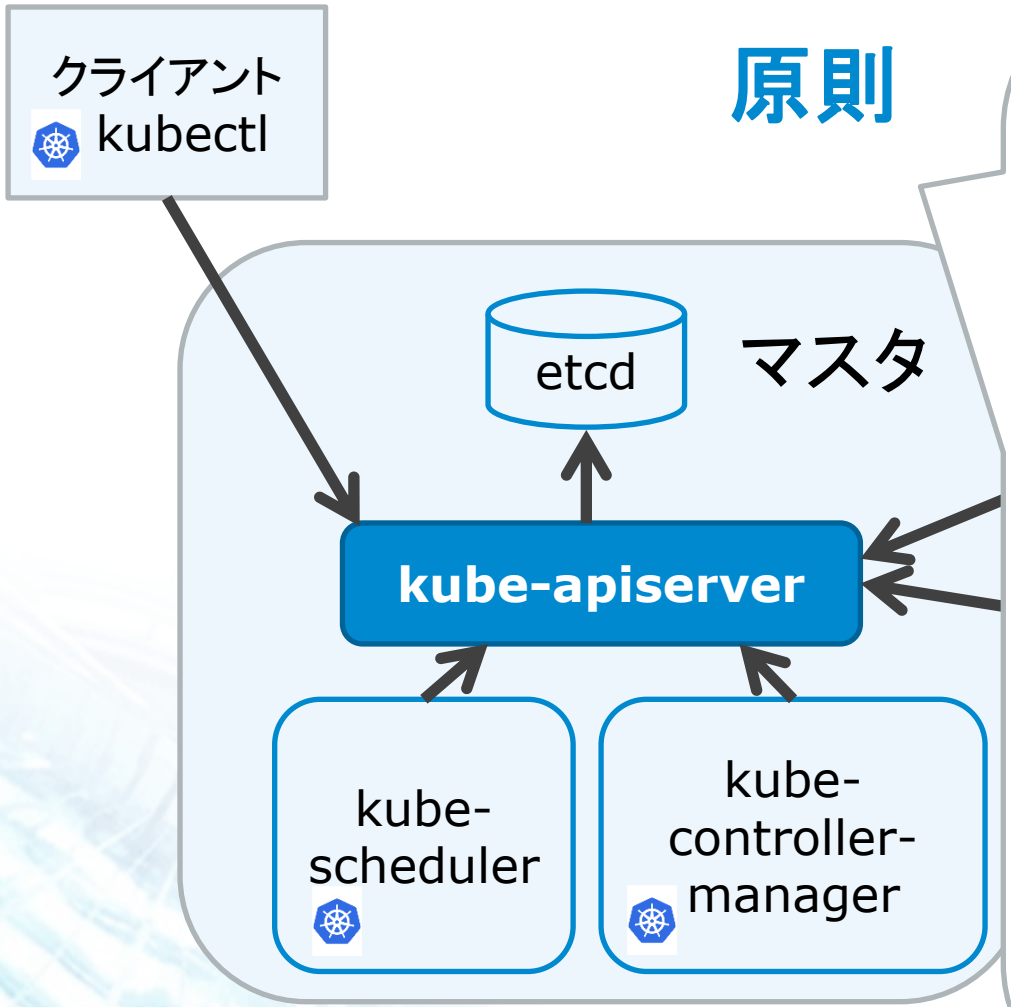
コンテナイメージの指定

H/Wリソースの設定

kubernetes アーキテクチャ



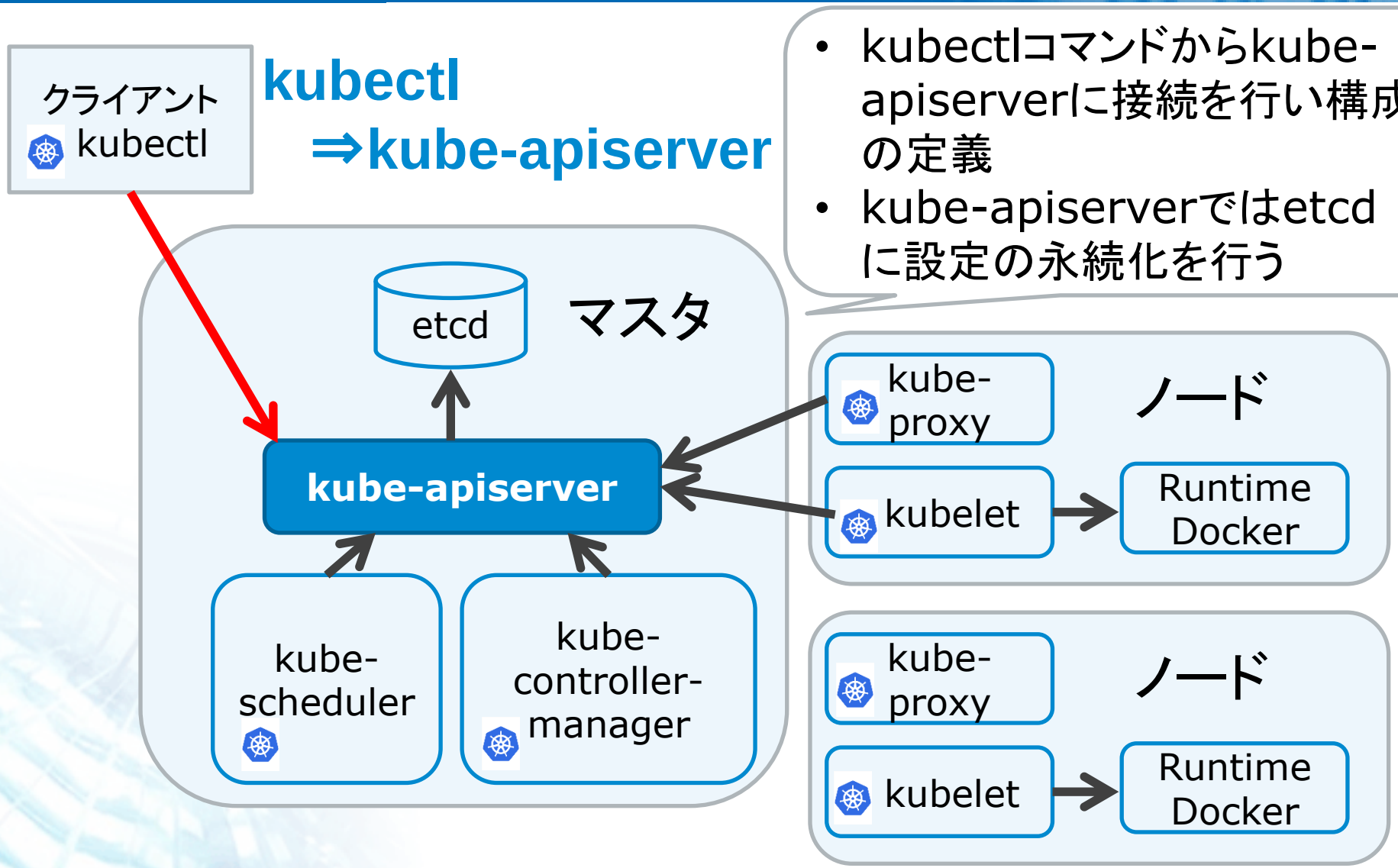
kubernetes アーキテクチャ



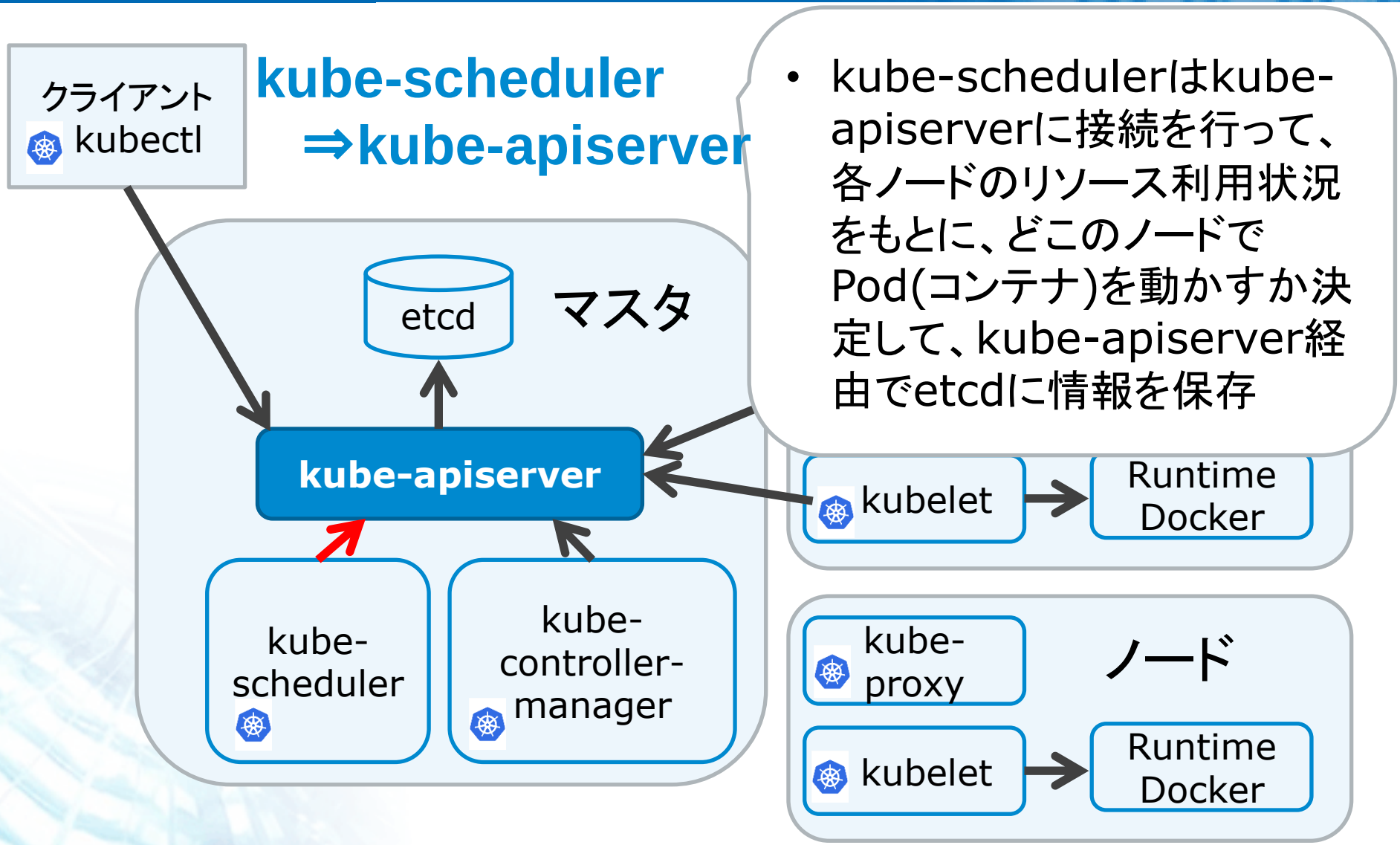
原則

- 各コンポーネント はkube-apiserverを通してやりとりを行う
- データはetcdに保存される
- kube-apiserverからのみetcdへアクセスできる
- 部分的な障害がサービス停止にならないようなアーキテクチャ
- 他のコンポーネントに接続ができなくても、各コンポーネントは直近の内容をもとに動作を続ける

kubernetes アーキテクチャ

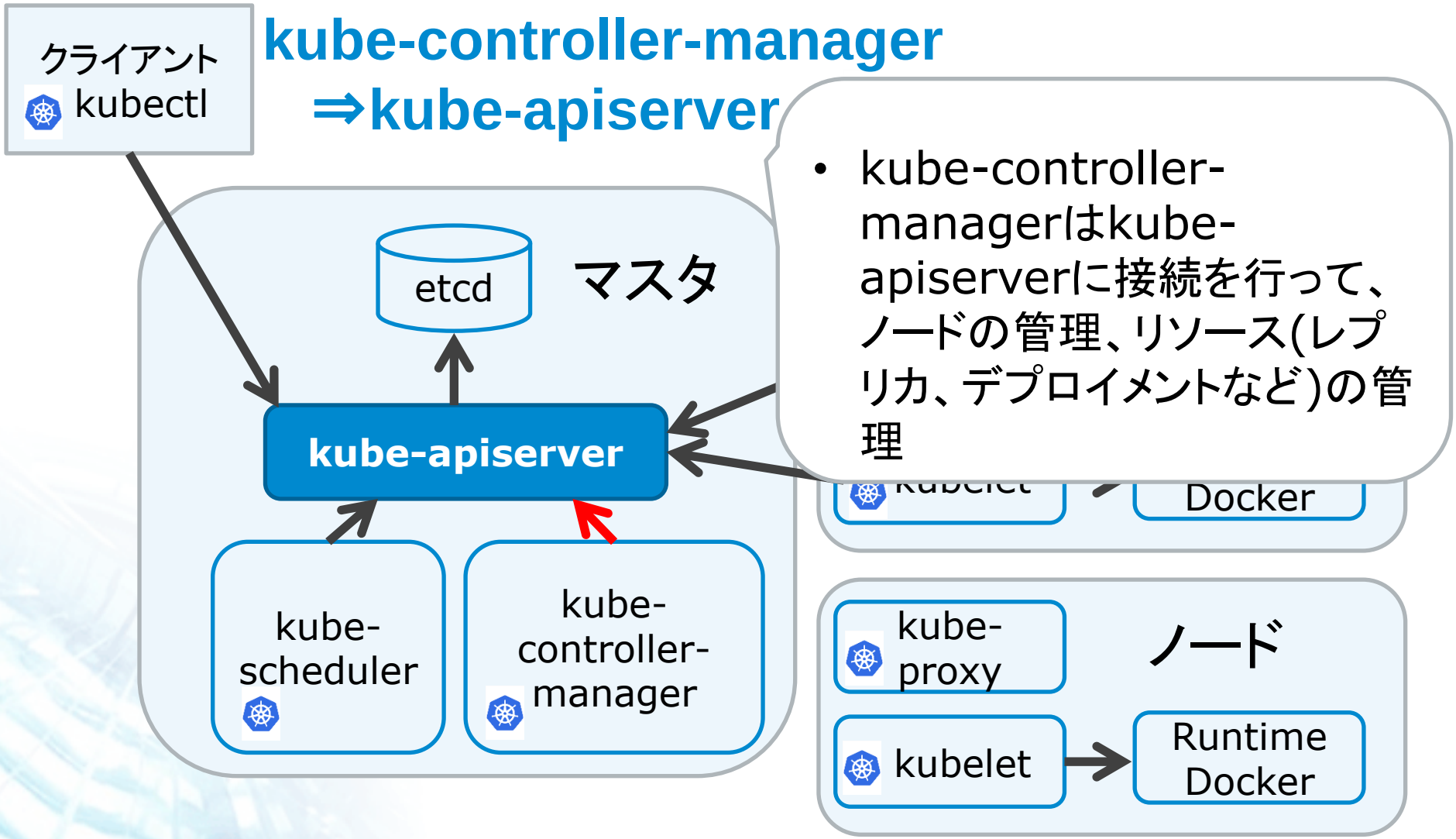


kubernetes アーキテクチャ

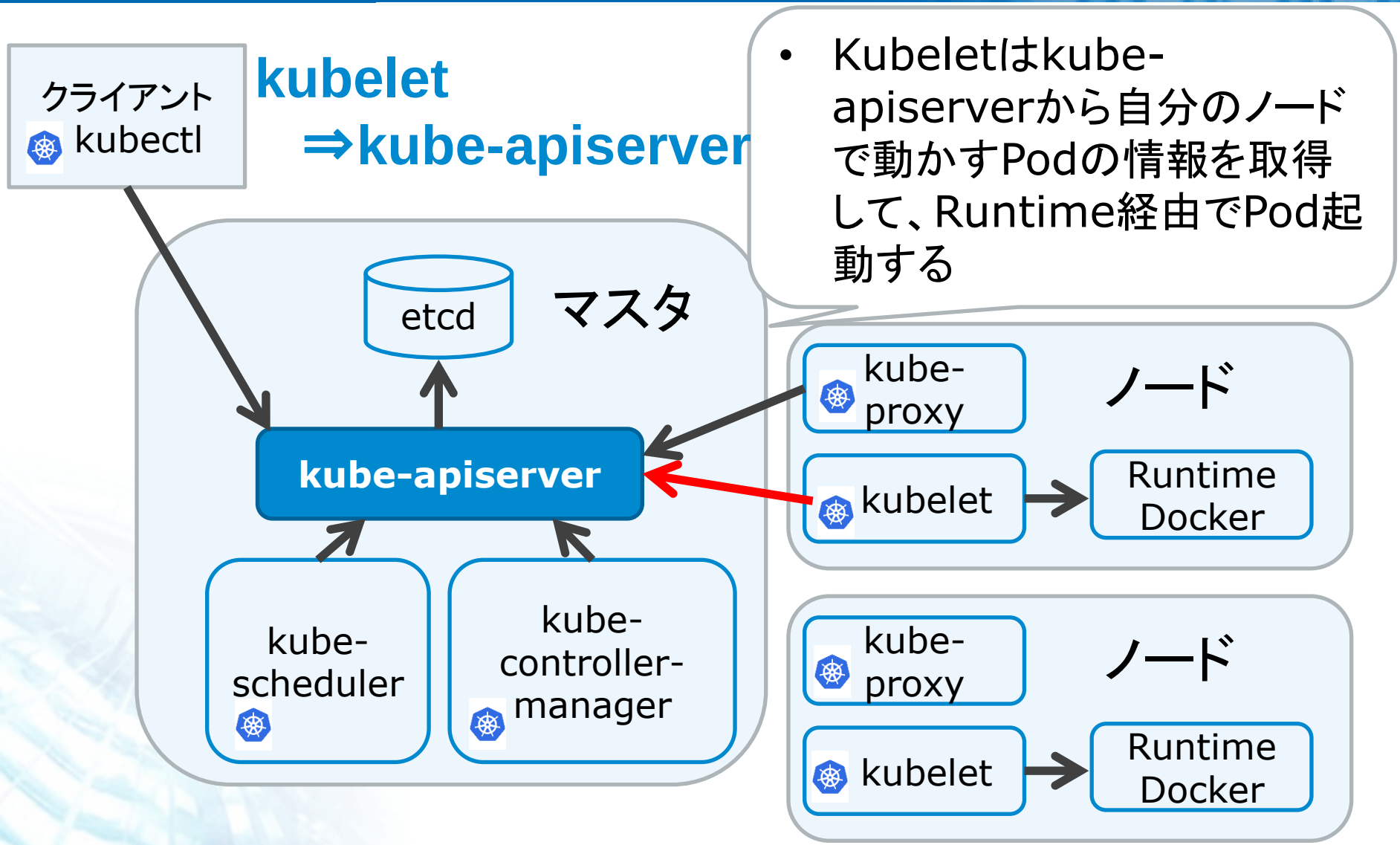


- kube-schedulerはkube-apiserverに接続を行って、各ノードのリソース利用状況をもとに、どこのノードでPod(コンテナ)を動かすか決定して、kube-apiserver経由でetcdに情報を保存

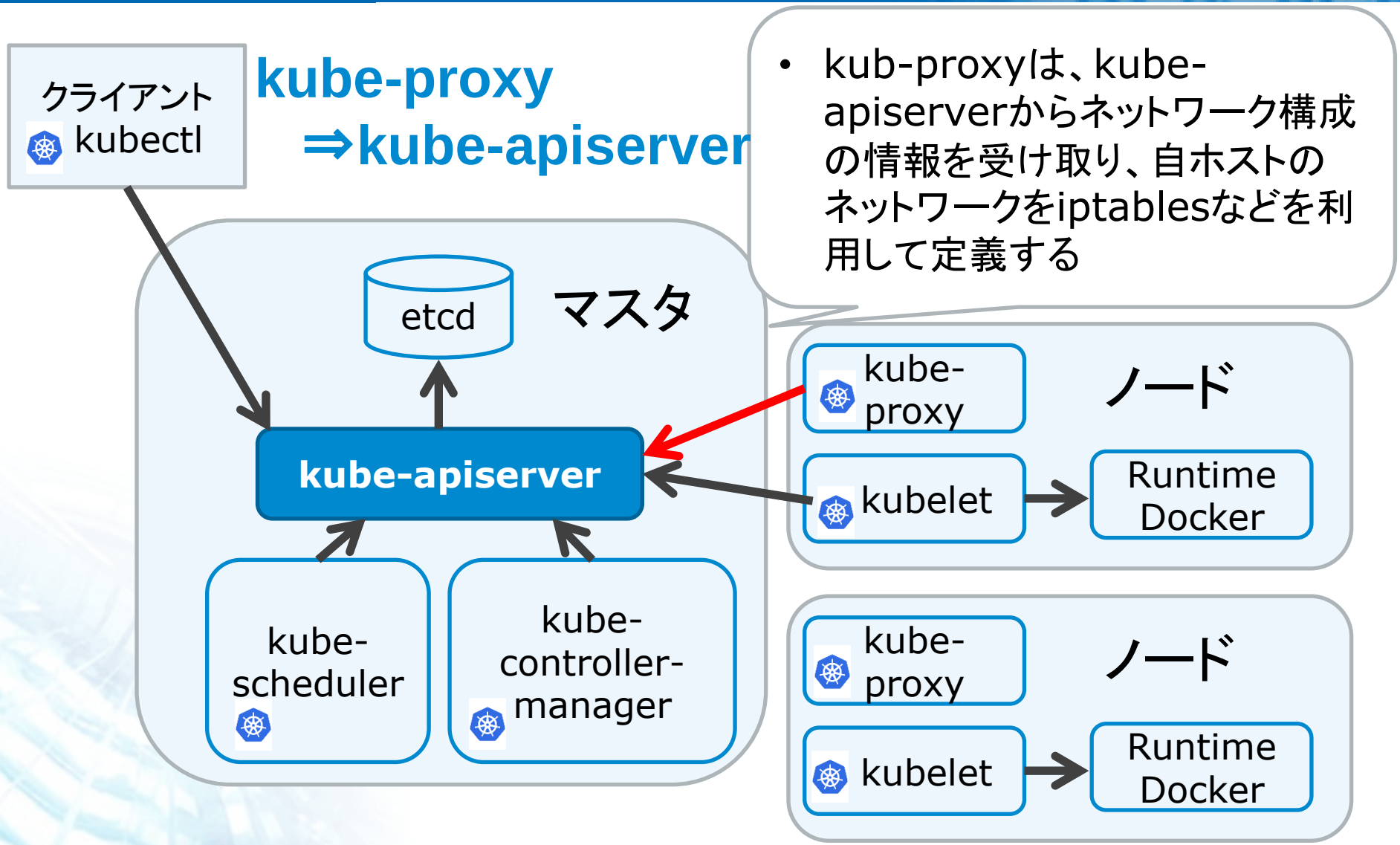
kubernetes アーキテクチャ



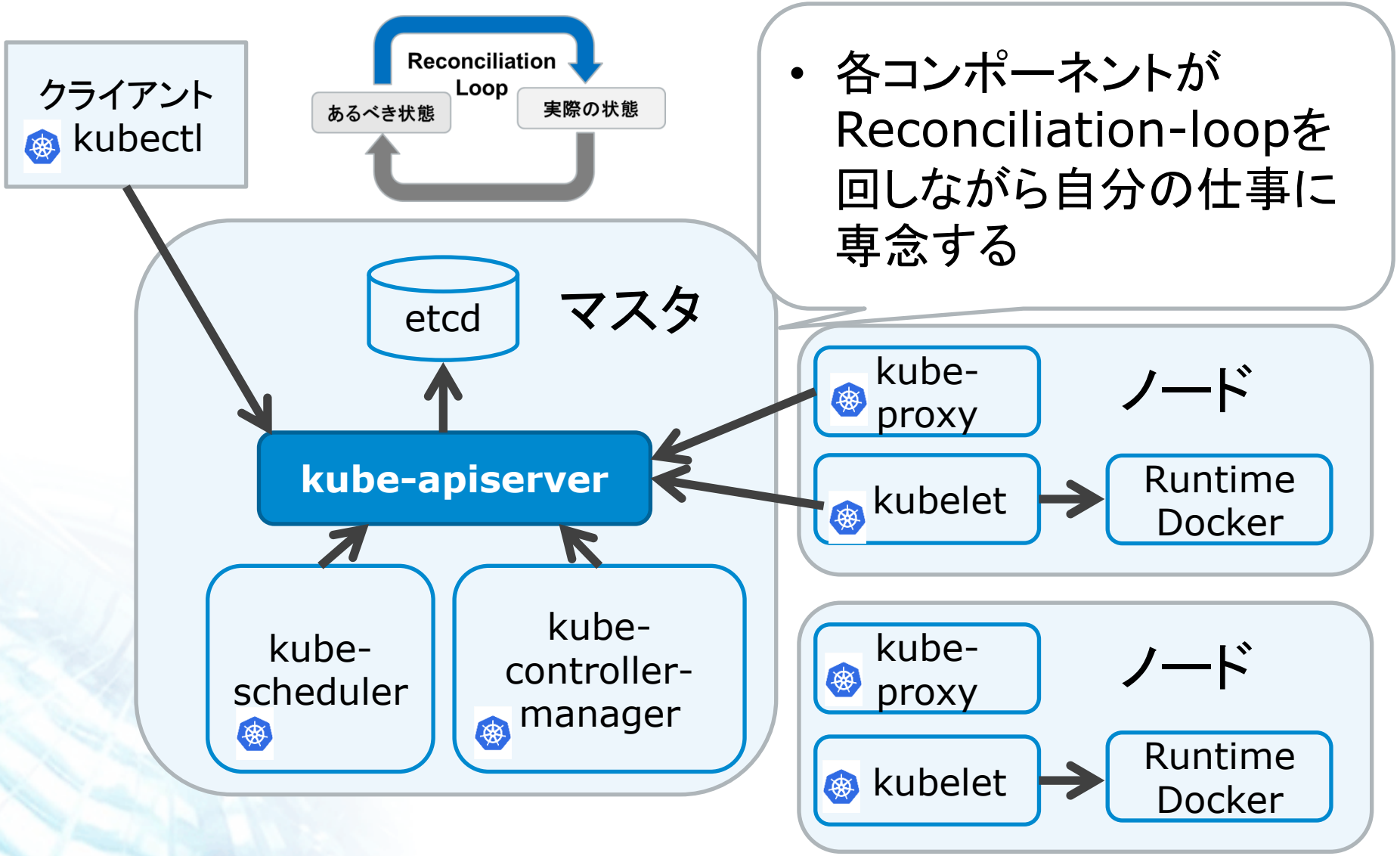
kubernetes アーキテクチャ



kubernetes アーキテクチャ



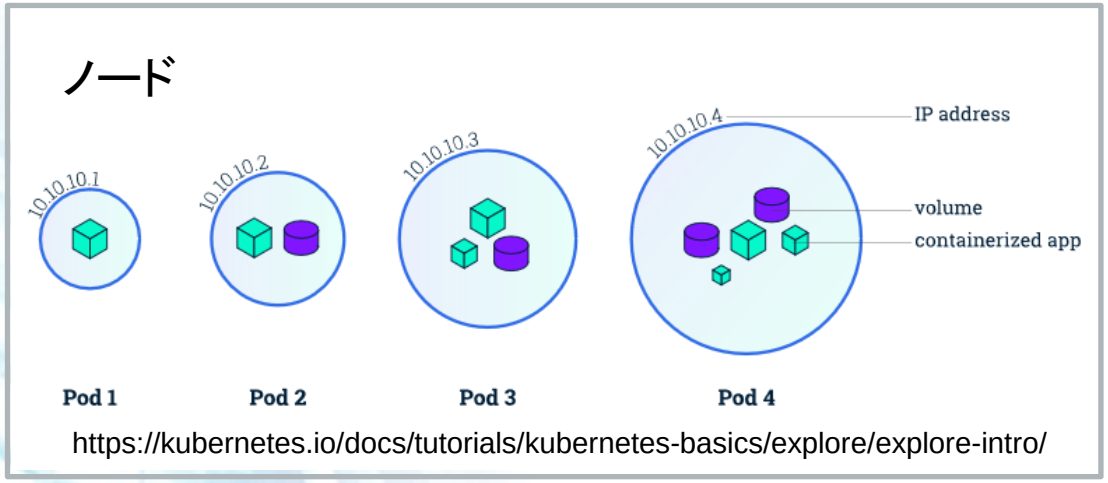
kubernetes アーキテクチャ



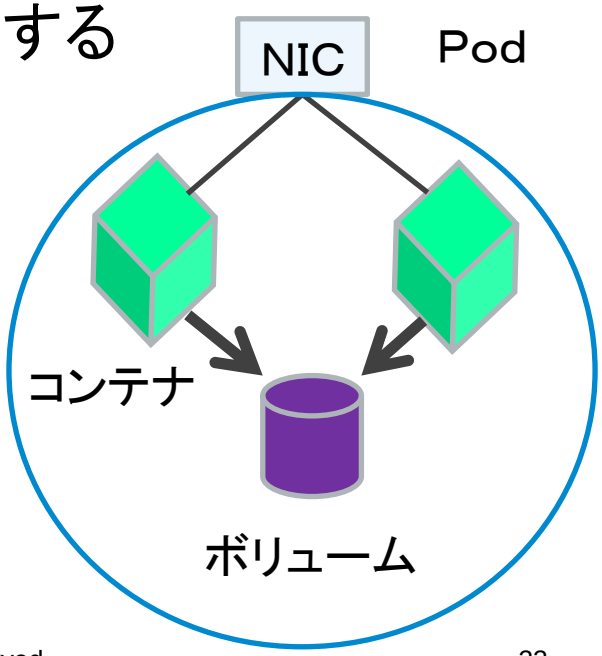
- 各コンポーネントがReconciliation-loopを回しながら自分の仕事に専念する

Podについて

- kubernetesにデプロイする「論理ホスト」
  - 1つ以上のコンテナ
  - 共有ネットワーク
  - 共有ボリューム
  - 同じPod内のコンテナは同一ノードにデプロイされる
  - Podはいずれかの Namespaceに属する



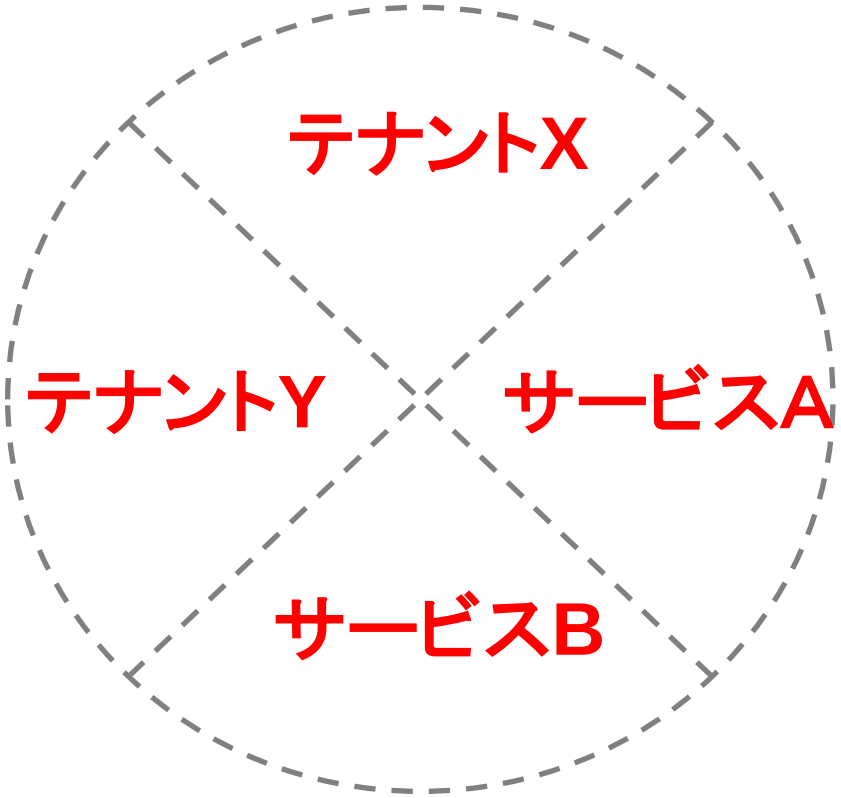
https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/



Namespace(名前空間)

- 一つのk8sクラスター上で複数の仮想クラスターの動作をサポート

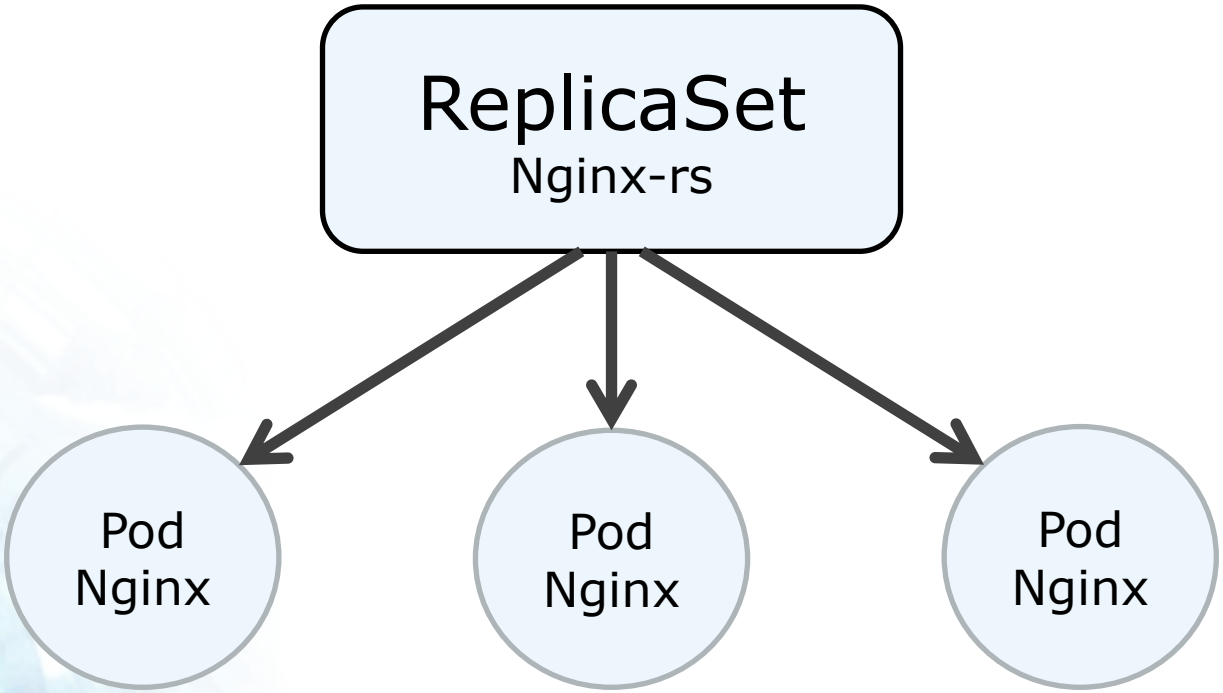
開発・テスト環境



本番環境

ReplicaSetについて

- Podの上位リソース
- 同じPodを指定数だけ起動して管理するリソース
  (コントローラが指定数動いているか確認する)



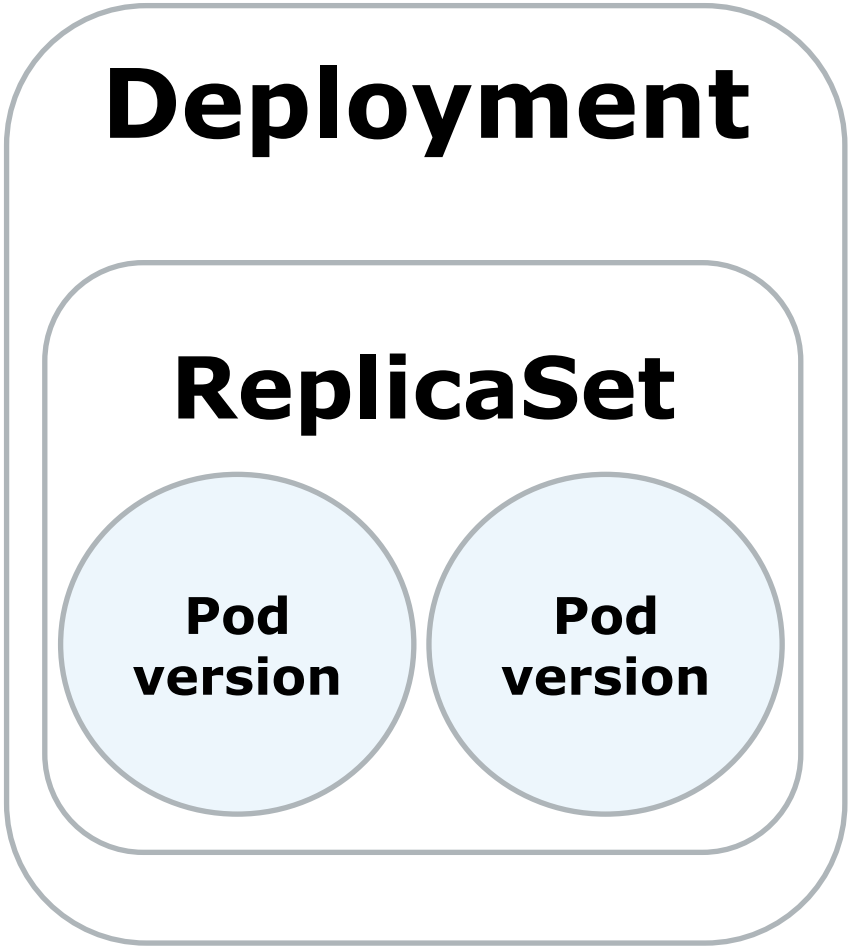
Deploymentについて

- ReplicaSetの上位リソース
- バージョン管理

  特定のバージョンにアップデート/ロールバック
- アップデート戦略
  - Recreate

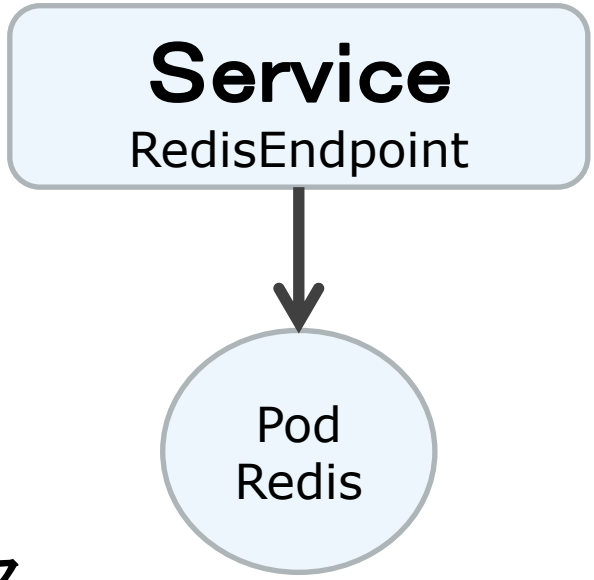
    Podをすべて停止、新しいバージョンで一斉に起動
  - RollingUpdate

    少数の Pod を順次アップデート



Serviceについて

- PodのIPアドレスは不定
- ServiceはPodへのエンドポイントを提供する
  - クラスタ内部向け
  - 外向け
  - クラスタ内外の負荷分散
- Kubernetes内部でもっている
  DNSを利用して名前解決
- アプリからはサービス名でアクセス

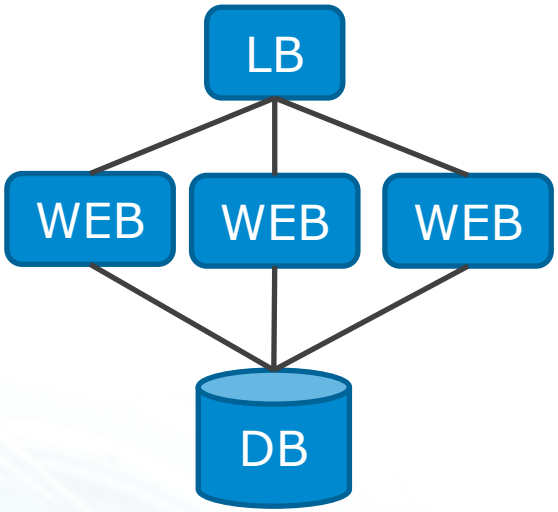


その他のリソース

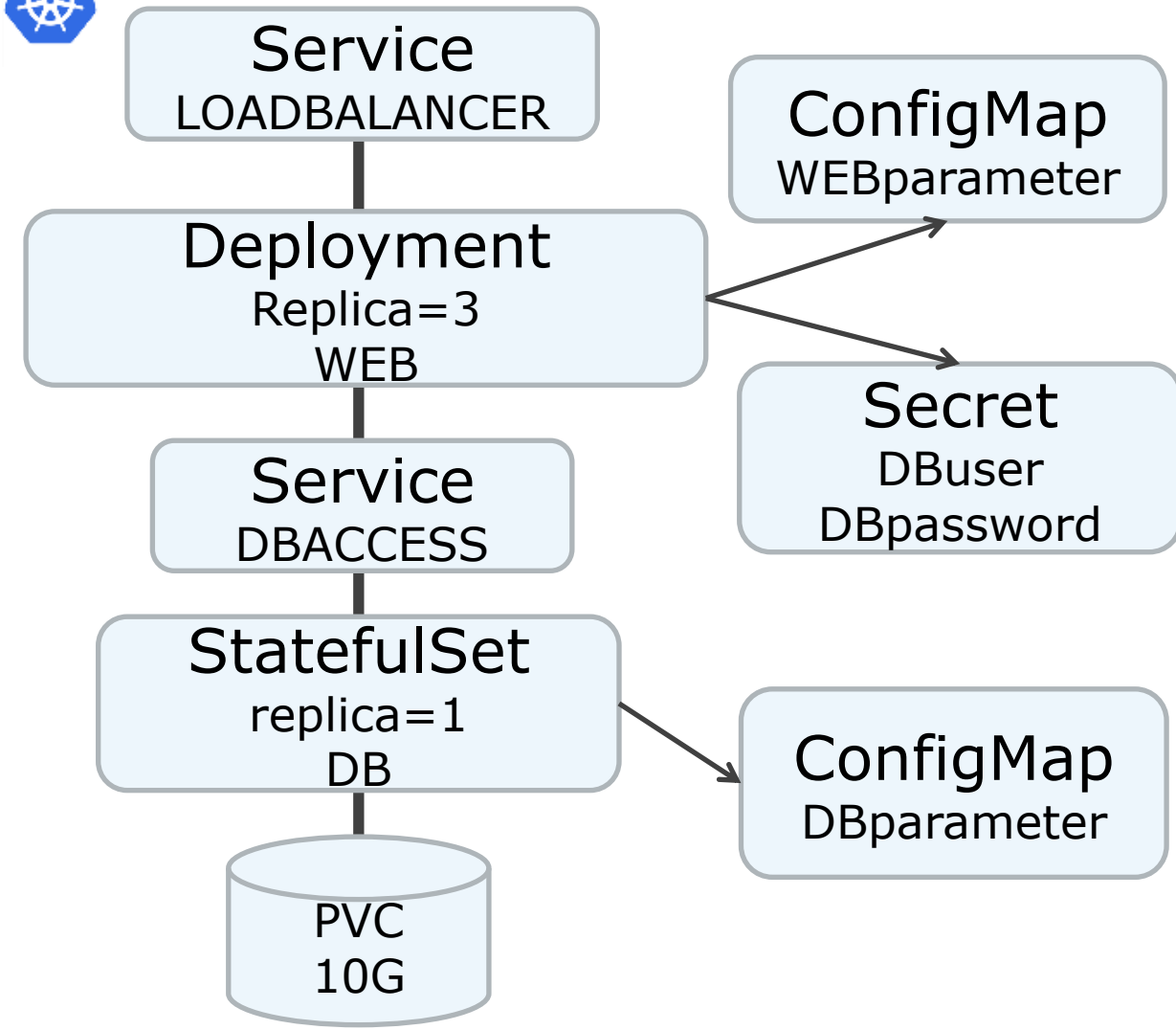
- DaemonSet
  - 各ノードで一つだけ動くPod
- Statefulset
  - 状態を保持するアプリケーション
  - ネットワーク識別子、永続的なストレージを利用できる
- Job/CronJob
  - 処理が終わったら終了するPod
- PersistentVolume/PesistentVolumeClaim
  - 永続的なボリューム
- ConfigMap/Secret
  - 設定・パスワード

構成図とkubernetesのリソース

構成図

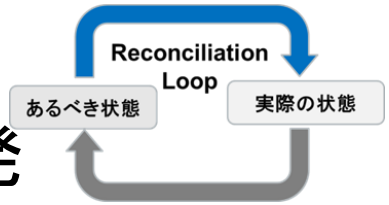


リソースを組み合わせてYAMLファイルで構成定義

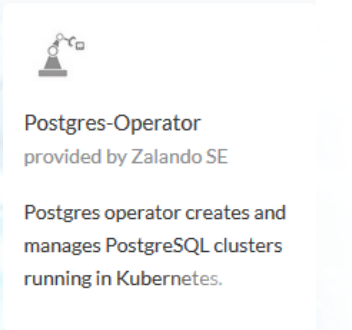


カスタムリソース

- リソースを自分で作ることもできる
  ⇒ 独自のReconclilation Loopの開発
- オペレータの運用ノウハウをコードで表現できる
- https://operatorhub.io/で公開されている



例: Zalando社：Postgres-Operator
PostgreSQL SR構成を管理してくれるリソース
Primary-Standby(マスタ-スレーブ)の状態管理



システム管理者と開発者の役割

VMによる運用

- システム管理者
  - VMの払い出し
  - ネットワークの定義
  - ホストの監視・管理
  - ミドルウェアの冗長化対策
- 開発者
  - sshなどでVMにアクセスして環境の構築

コンテナによる運用

- システム管理
  - kubernetesクラスタの監視・管理
- 開発者
  - ファイルベースで構成を定義
  - Gitなどでアプリケーションと構成を管理

kubernetes 長所と短所

Pros

- コンテナ開発による生産性の向上
- コンテナによる環境管理の負荷軽減
- コンテナによる効率的なサーバ運用
- 自動スケーリング
- 障害の自動修復
- ファイルベースによる構成、バージョン管理
- 特定のベンダに支配されない標準技術

Cons

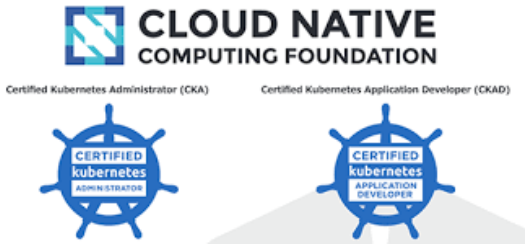
- 学習コストが高い
- Kubernetesのリリースが早くて追従が大変
- エコシステム(関連ソフトウェア)も進歩が速く追従が大変

kubernetesの学習

- オンラインコースの活用
  - ブラウザでkubernetesクラスタを操作できる

認定資格制度

- Certified Kubernetes Administrator (CKA) 試験
  - Kubernetesの管理者に必要なスキル、知識、および能力が問われる試験
  - ハンズオン形式の試験　試験時間3時間



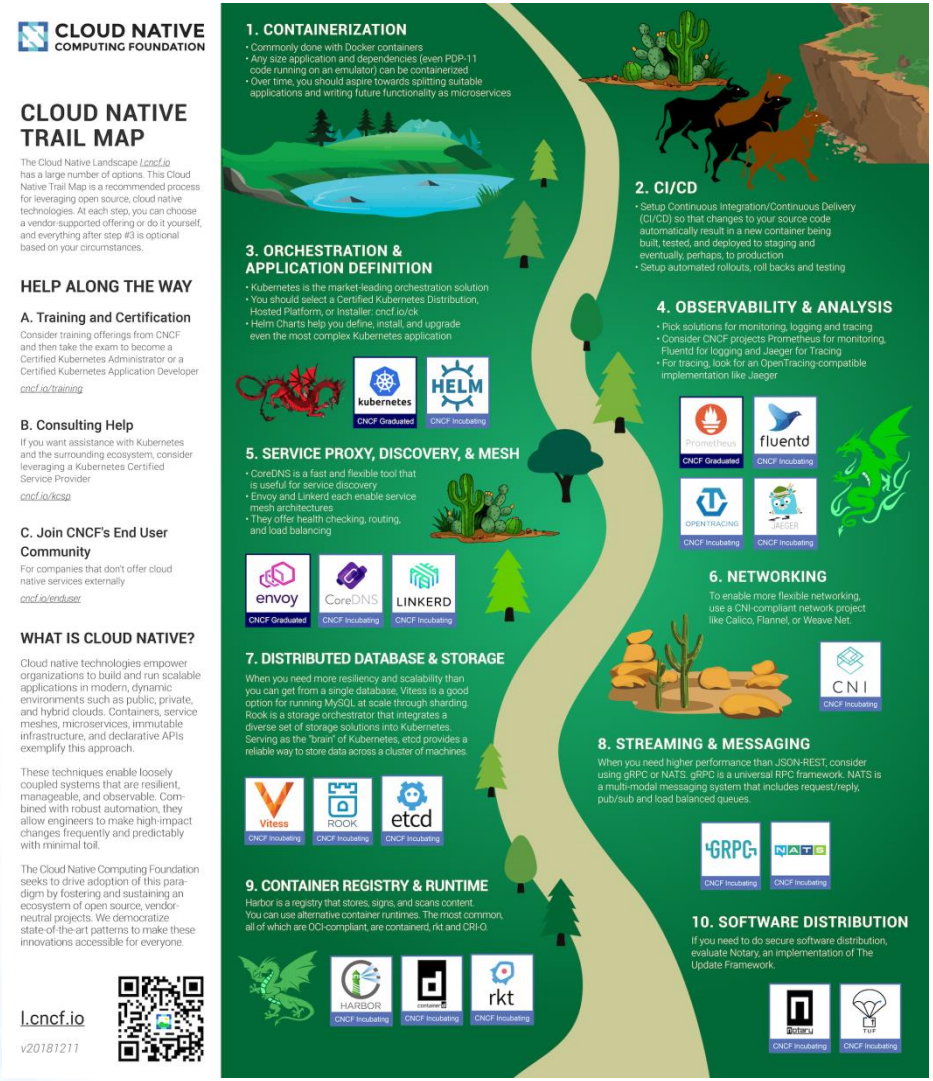
- Certified Kubernetes Application Developer（CKAD）試験
  - Kubernetes上でアプリケーションを設計・構築・設定・展開できることを証明する資格試験
  - ハンズオン形式の試験 試験時間 2時間

情報収集

- **kubernetes slack** https://slack.k8s.io/
  #jp-users #jp-events チャンネル

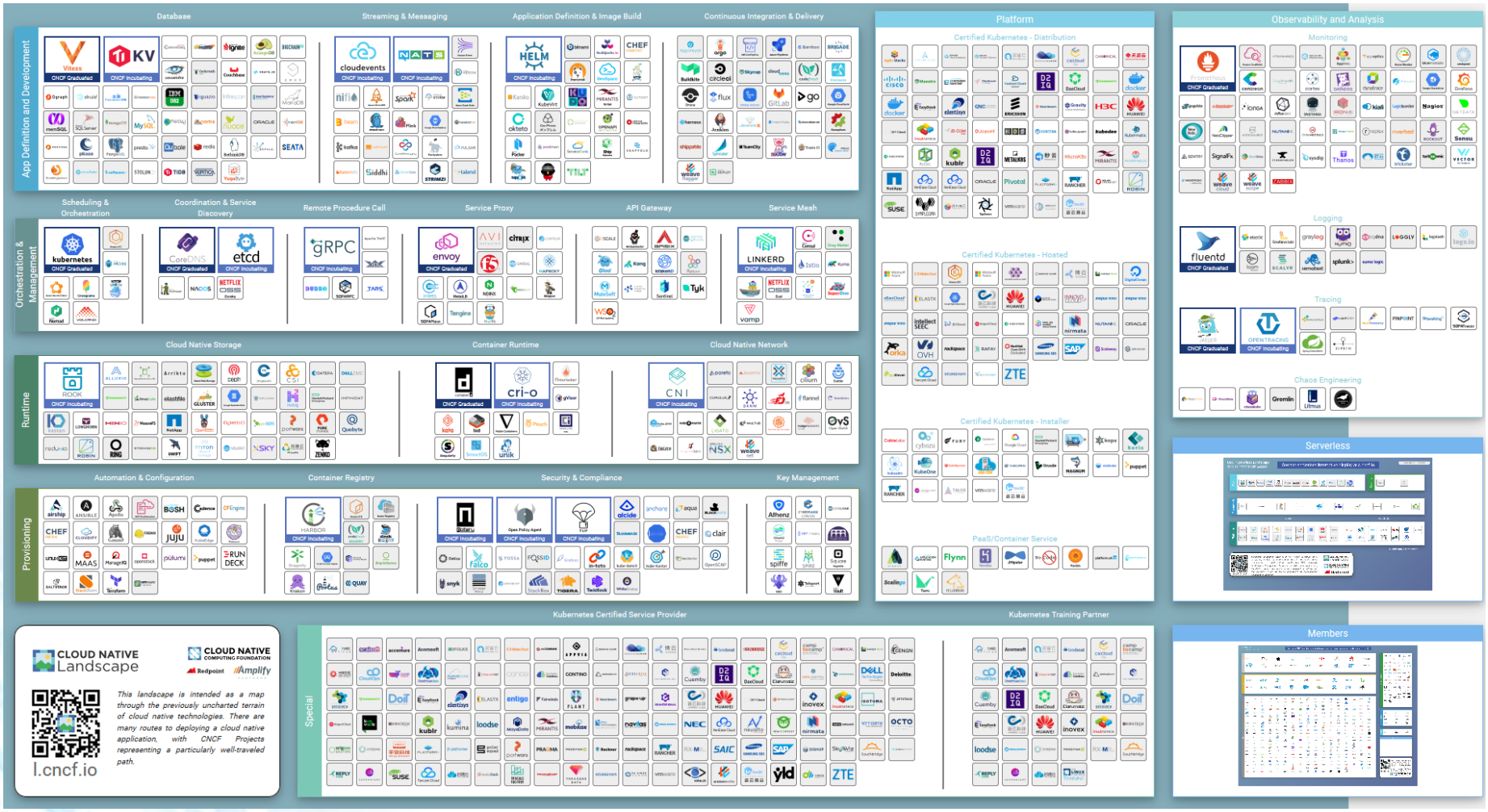
- **Kubernetes Meetup Tokyo**
  https://k8sjp.connpass.com/
  毎月開催 19:00~
  200~300人規模のイベント
  youtube配信あり

CNCF トレイルマップ



https://github.com/cncf/landscape/blob/master/README.md#trail-map

CNCF landscape



ご清聴ありがとうございました。