コーステキスト補足資料

DO280v4.14

Red Hat OpenShift Administration II: Configuring a Production Cluster

2027年 01月 27日

レッドハット株式会社
トレーニングサービス部

Red Hat

ご注意

- 本資料はトレーニング資料につき個人での学習目的でのみ使用し、再配布、公開、加工はしないでください。

Red Hat

EX280 認定試験概要

Red Hat 認定試験概要

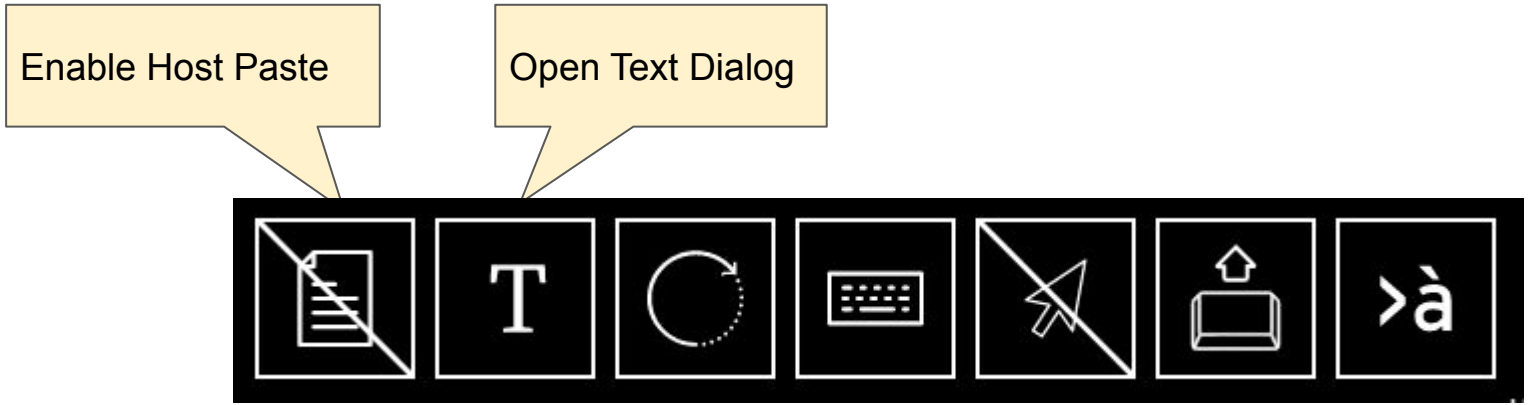
- 試験範囲
  - 試験範囲がEX280のページ上で公開されている
- 試験時間
  - 3 時間
- 試験形式
  - 実技試験
- 合否通知
  - 機械採点
  - 採点基準の詳細は非公開
  - メールによる通知（数日以内）

Red Hat

演習でのタイプ入力を減らすコツ

PDFテキストから仮想マシンへのコピー＆ペースト

- 仮想マシンの右上のボタンを使うことでコピー＆ペーストが可能
  - **Open Text Dialog**: ダイアログボックスにペーストしてSendボタン押下
  - **Enable Host Paste**: Enableにして**Ctrl-v**でペースト(MacOSはCommand-v)



Red Hat

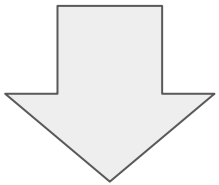
コマンド入力補完やヒストリを駆使してタイプを減らす

- ocコマンド名をタブで補完できる
  - oc adm policy add-<TAB>
- リソース名もタブで補完できる
  - oc get svc post<TAB>
- 一度入力したコマンドはシェルのヒストリから再度実行
  - oc login -u kubeadmin -p xxxxx
  - Ctrl-r oc login

参考:インストール後のoc completion設定

ocコマンドのタブ補完を有効にするには、事前にoc completionの設定をしておく

$ **oc completion bash** > bash_completion.sh
$ source bash_completion.sh



.bashrcに source bash_completion.sh を設定する

コマンドライン上でのパラメータの調べ方

- オプションスイッチを調べたい
  - oc get **-h**
- リソースの短縮名を調べたい
  - oc **api-resources**
- リソースの属性を調べたい
  - oc **explain** pod.spec.containers

Kubernetesのアーキテクチャー

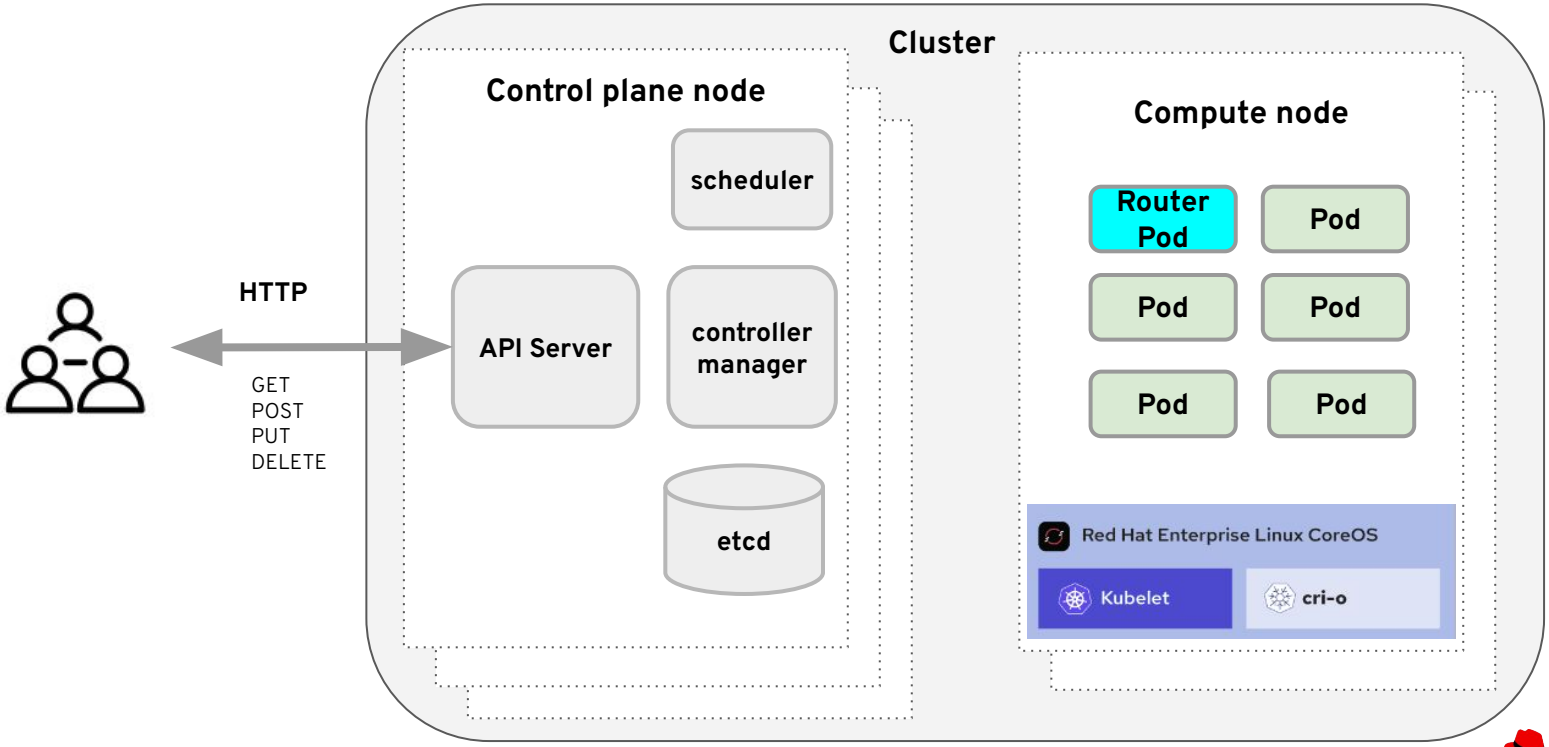
Red Hat

Kubernetesアーキテクチャー

Red Hat

ノード

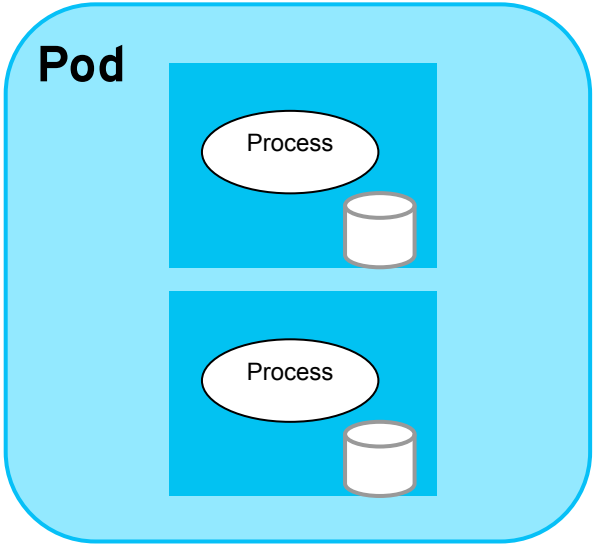
- コントロールプレーンノードには、APIサーバー、コントローラーマネージャー、etcdなどが含まれる。
- Computeノードには、Podを制御するKubletやコンテナーランタイムのCRI-Oが含まれる。



Red Hat

Pod

- Kubernetesにおける基本的な要素
  - Nodeへのデプロイは Pod 単位で行われる
- Podは複数のコンテナー を管理する
  - Podには通常は1つのコンテナーを含む
- Pod内のコンテナーはPod内リソースを共有する
  - IPアドレス
  - ストレージ



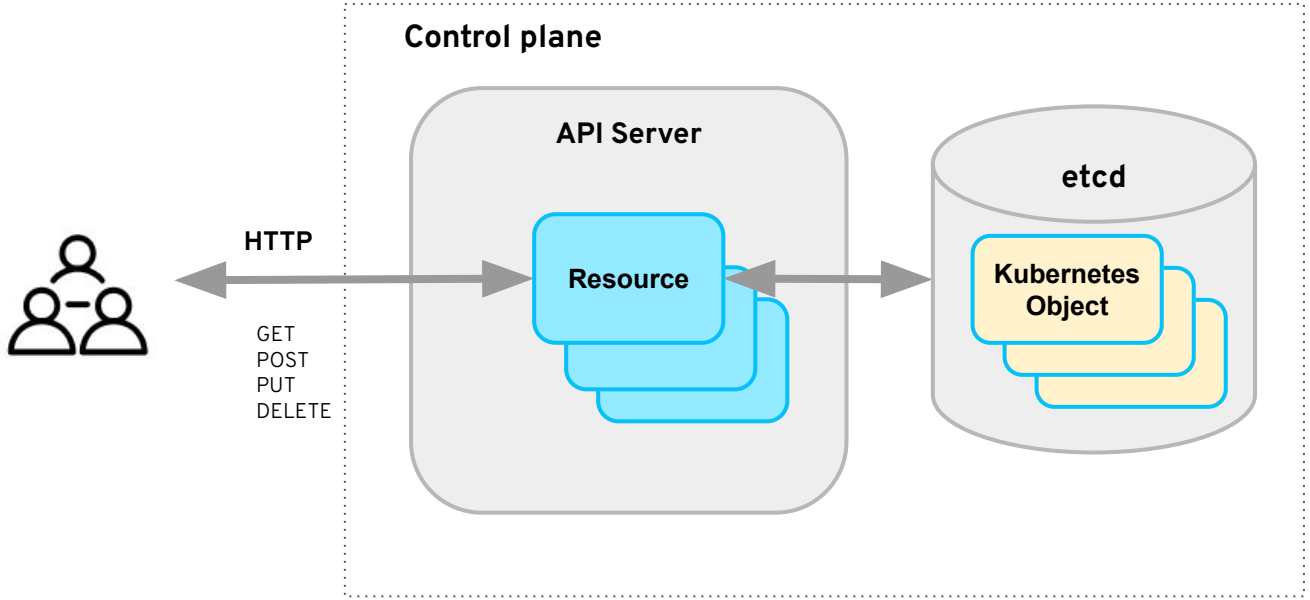
Red Hat

リソースとコントローラー

Red Hat

API、リソース、およびオブジェクト

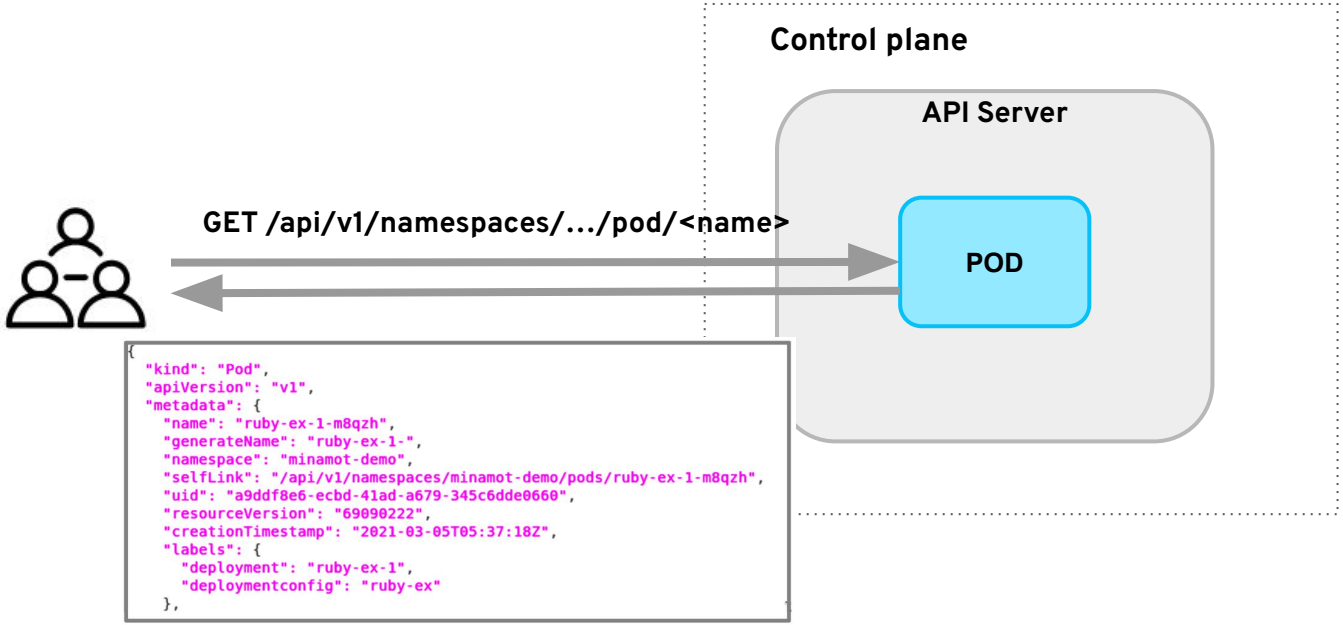
Kubernetesは、REST APIを公開する。このAPIはリソースへのアクセスポイントを提供する。APIによって作成されたリソースはetcd内でデータ(Kubernetesオブジェクト)として管理される。



Red Hat

Kubernetes API

ocコマンドやWebコンソールでリソースを操作するとき、HTTPリクエストがAPI Serverに送信される。
例えば、Podの詳細を調べるにはGETリクエストが送信され、JSONデータとして結果が返される。

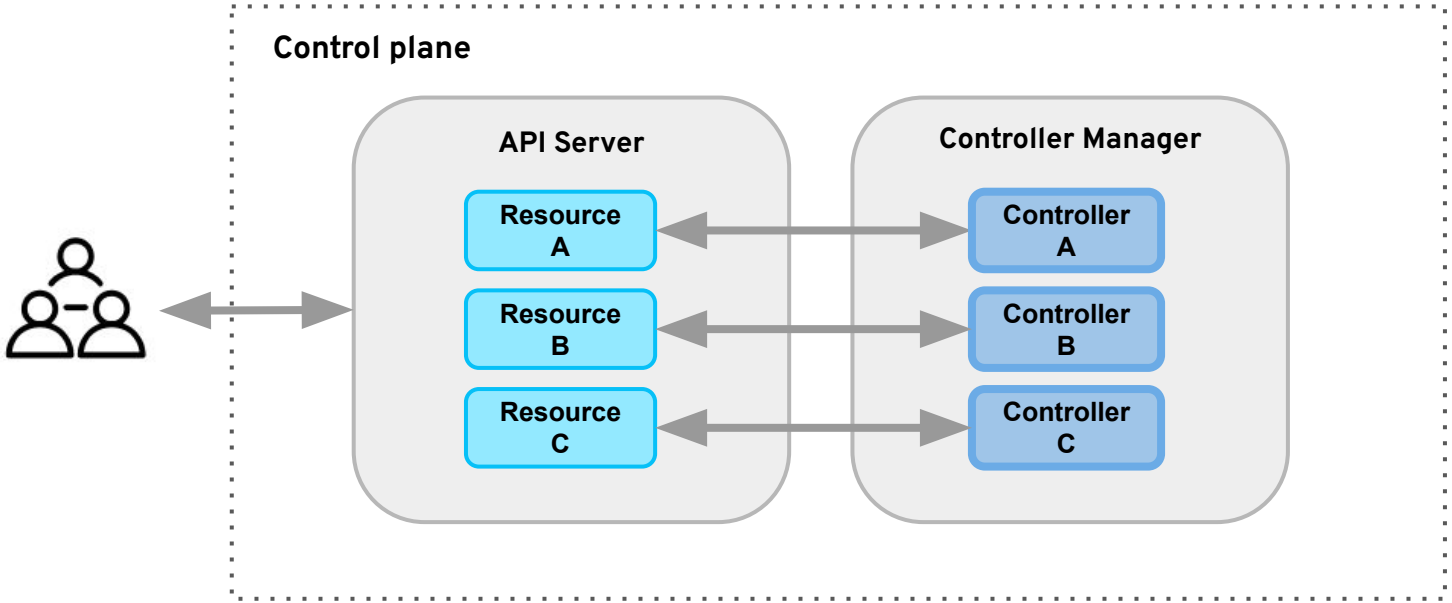


https://kubernetes.io/ja/docs/reference/

Red Hat

リソースとコントローラーの関係

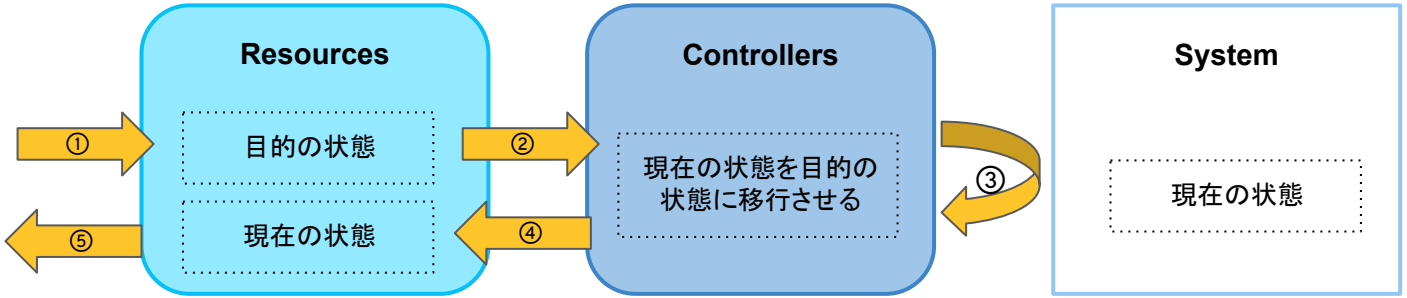
Kubernetes APIには多くの種類のリソースが定義されている。Kubernetesユーザーはリソースを作ることで、システムに対して指示を出す。リソースごとに対応するコントローラーが存在し、リソースに書かれていることを実行する。コントローラーの実行結果はリソースに書かれるので、Kubernetes利用者はリソースを調べることで実行結果を知ることができる。



Red Hat

Reconciliation Loop (調整ループ)

Kubernetesでは、リソースに書かれた「目的の状態」を維持するように動作する。コントローラーはシステムの「現在の状態」が「目的の状態」と一致しているかどうかを監視し、両者が異なる場合は、「現実の状態」を「目的の状態」に移行させる(これを調整ループと呼ぶ)。 Kubernetes内で障害が発生した場合、この仕組みによって自動的に修復をおこなう。これを **自己修復(セルフヒーリング)** という。



①APIサーバーによってリソースが作成される。リソースには「目的の状態」が書かれる。
⑤APIサーバーで「現在の状態」を調べることでコンテナーの状態(実行中、エラー)がわかる

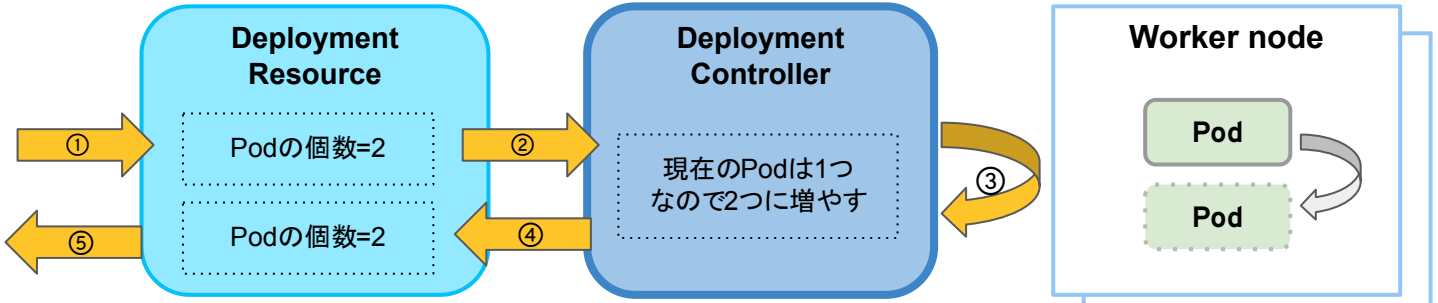
②コントローラーはリソースの状態を監視する
④実行結果をリソースの「現在の状態」に書く

③もし「目的の状態」と「現実の状態」が一致していなければ、コントローラーは「目的の状態」を満たすようにコンテナーを操作する

Red Hat

Reconciliation Loop (調整ループ)の例

Deploymentリソースは指定されたアプリケーションの Podデプロイするためのリソースである。このリソースの「目的の状態」の Podの個数(replicas)を修正することで、現実の Podの数を変更することができる。この処理はコントローラーによって行われるので、Kubernetes利用者は、実際の Podを直接に操作することはない。



①Deploymentリソースの「目的の状態の」のPodの数が1から2に修正された
⑤APIサーバーで「現在の状態」を調べることでPodの数が2に増えたことが確認できる

②コントローラーはDeploymentリソースの「目的の状態」のPodの個数の修正の通知を受ける
④実行結果をDeploymentリソースの「現在の状態」に書き込む

③Podの数を2つにするため、Podを一つ追加する

Red Hat

1章 宣言型のリソース管理

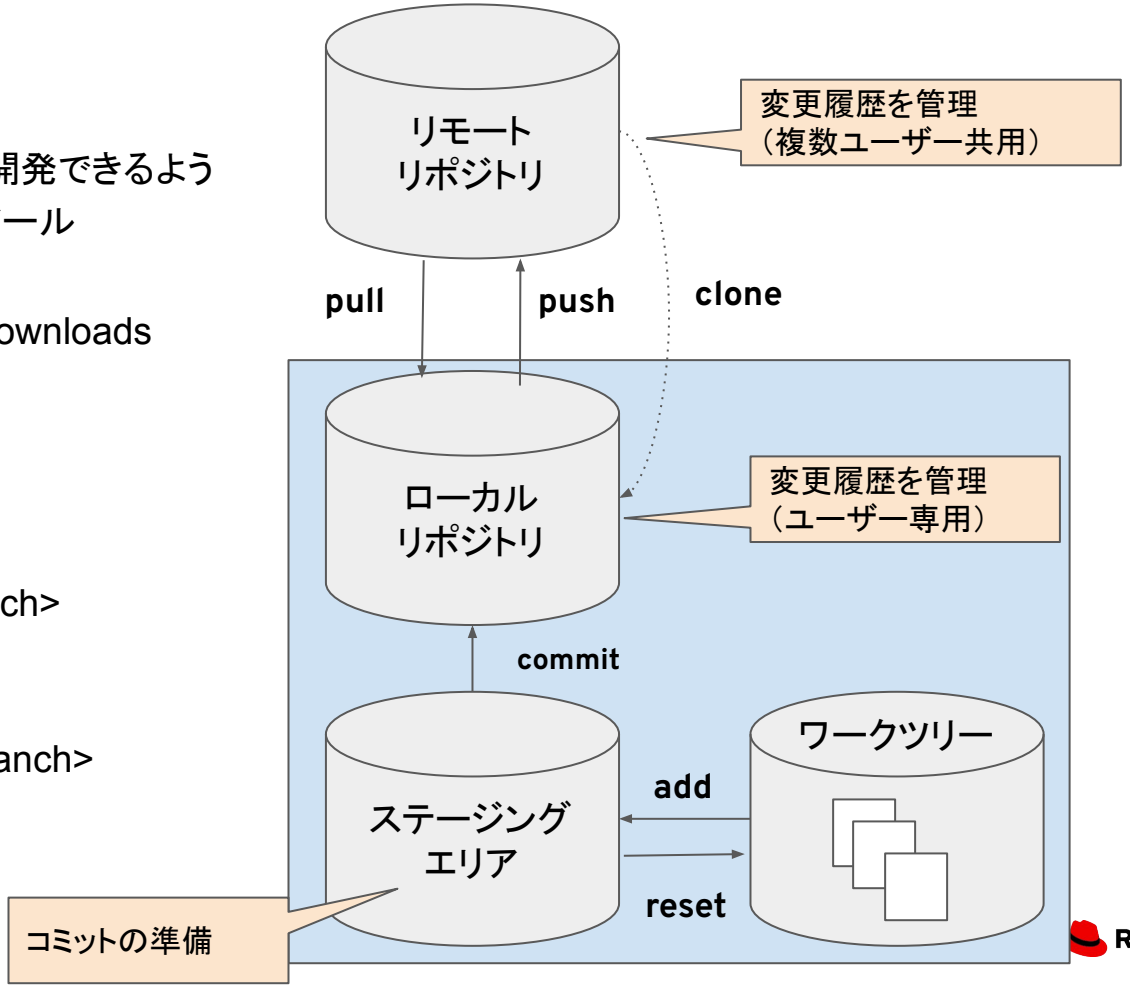
Red Hat

Git

Red Hat

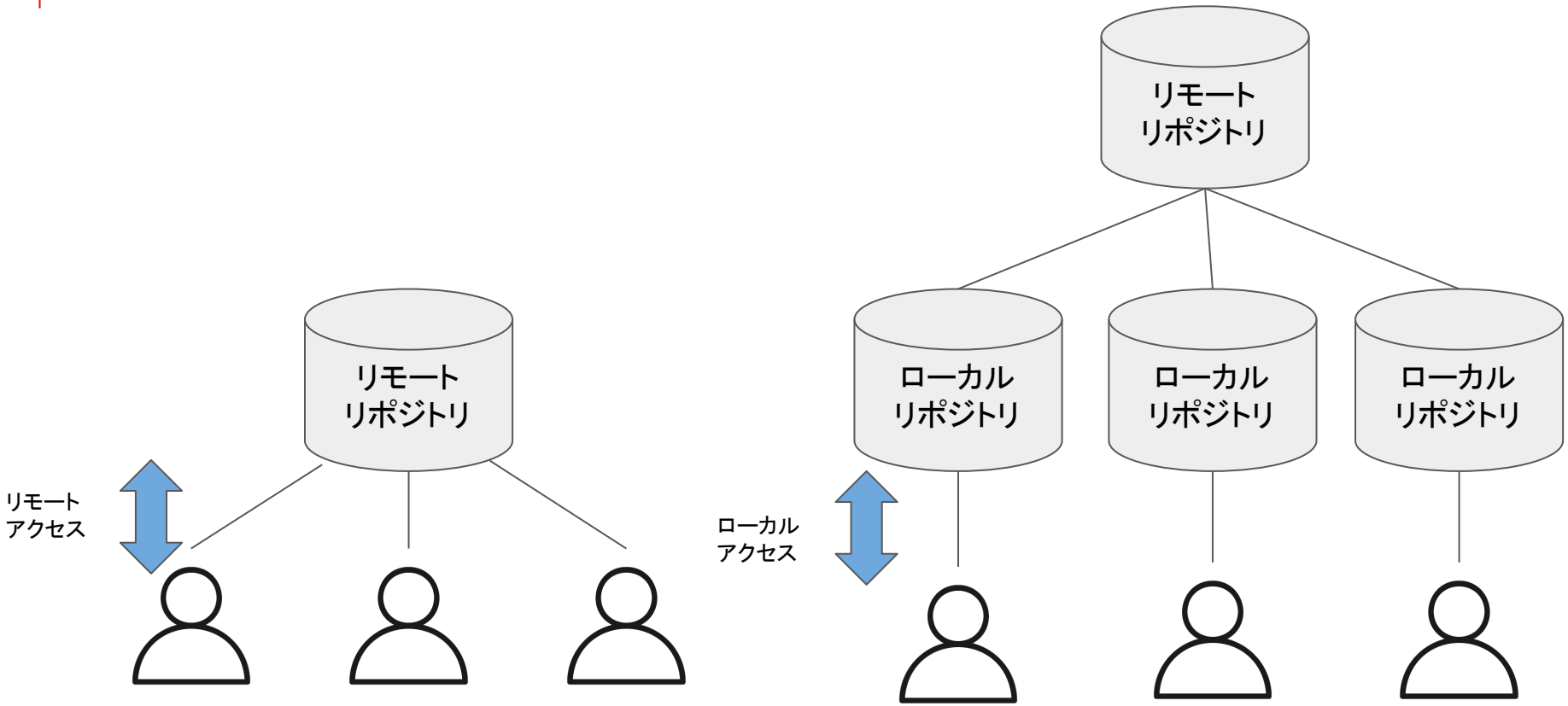
Gitの概要

- Gitとは
  - Linuxを分散して協同開発できるように考えられた分散型ツール
- Gitインストール
  - https://git-scm.com/downloads
- Git ワークフロー
  - git clone
  - git branch -a
  - git checkout master
  - git checkout -b <branch>
  - git add <files>
  - git commit
  - git push -u origin <branch>
  - git pull



Red Hat

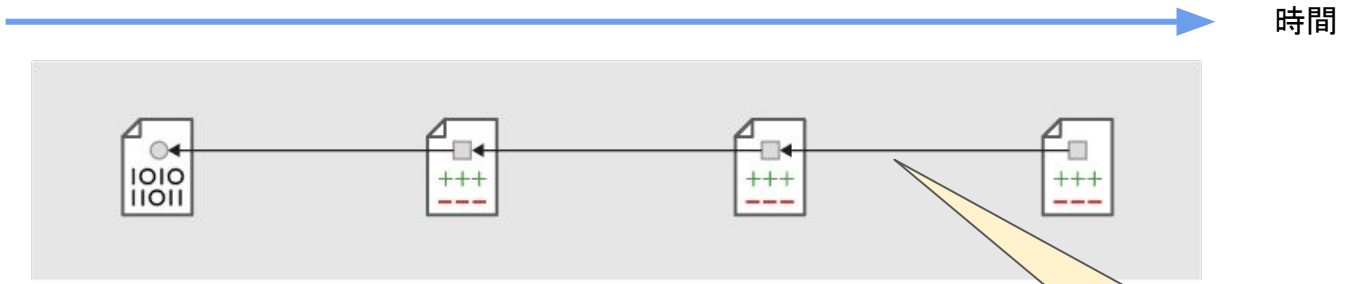
分散型と集中型の管理



Red Hat

コミットについて

リポジトリ内のファイルに変更を加える場合は、**コミット**でそれらの変更を保持できます。
Git は、各コミット内に親コミットへの参照を保存することで、コミットの順序も追跡します。
この方法でコミットを作成すると、チェーンが形成されます。



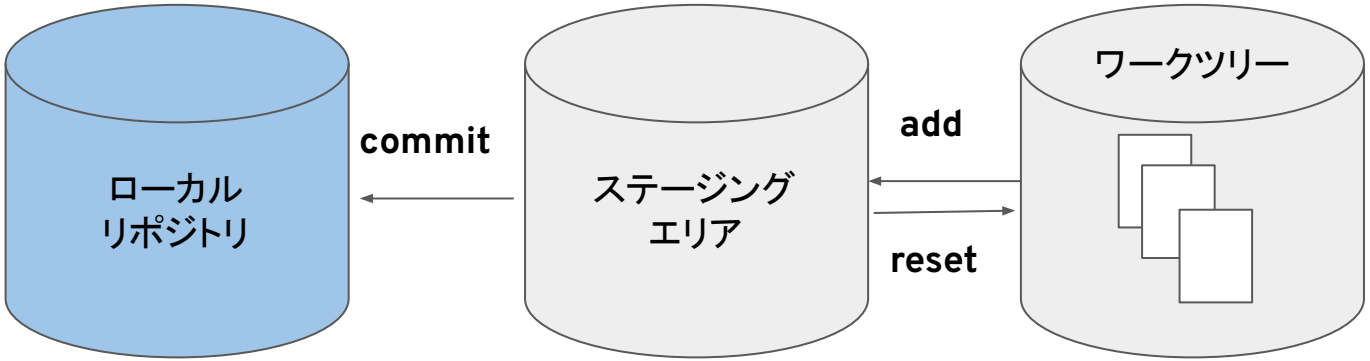
コミットの内容：

- 作成者
- タイムスタンプ
- メッセージ
- ハッシュ (コミット用に生成された一意の識別子)

変更のステージング

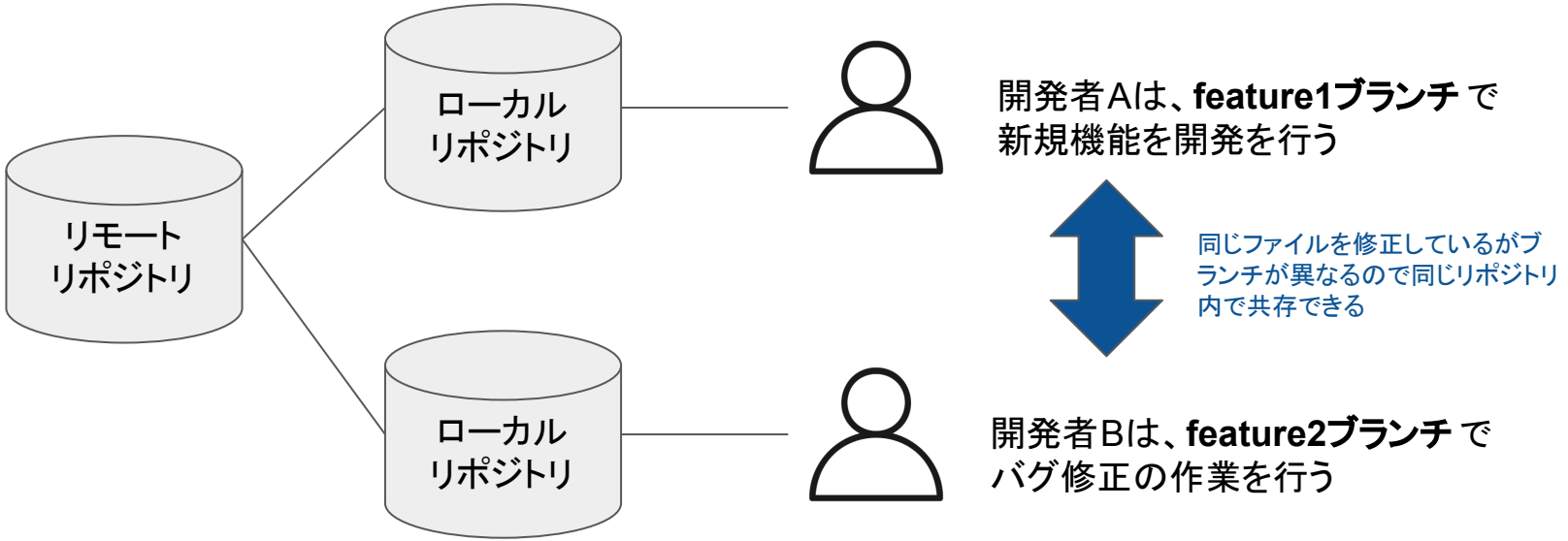
コミットを作成するためには、以下のステップを実行する。

1. プロジェクトのファイルを修正する
2. **git add** コマンドを使用して変更をステージングする
3. **git commit** コマンドを使用して変更をコミットする



ブランチを使用した変更の編成と組み込み

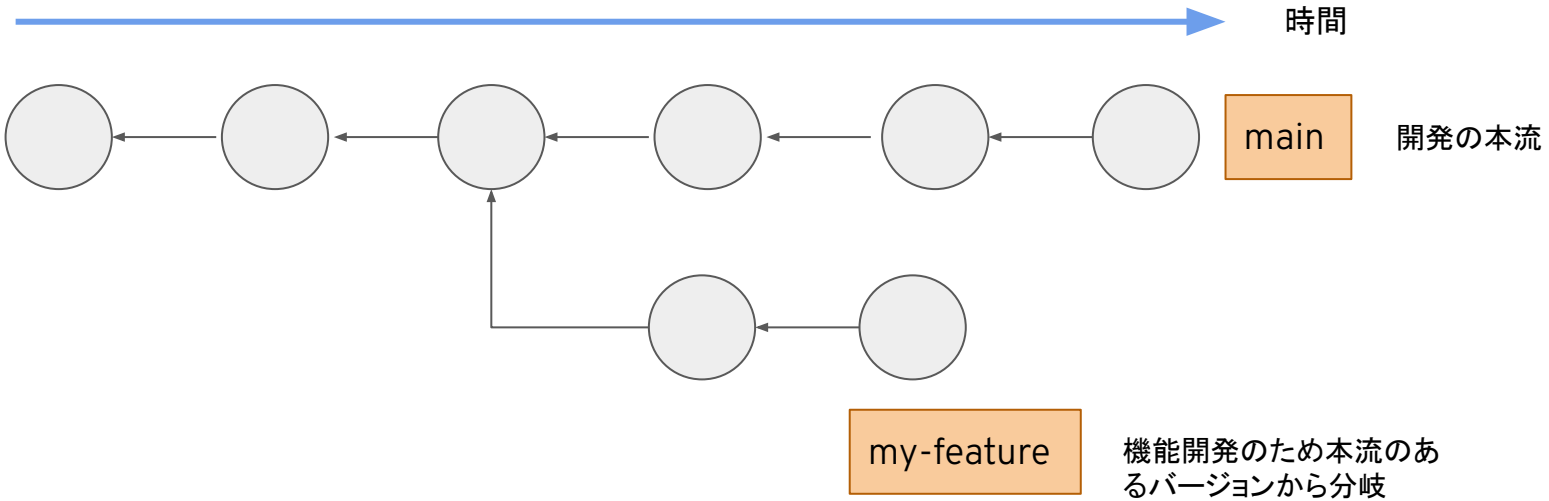
Git を使用すると、同じリポジトリに対する同時修正を**ブランチ**というコード変更の集合にまとめることができる。ブランチによって、開発の本流から分岐し、本流の開発を邪魔することなく作業を続けることが可能になる。



Red Hat

main ブランチの構築

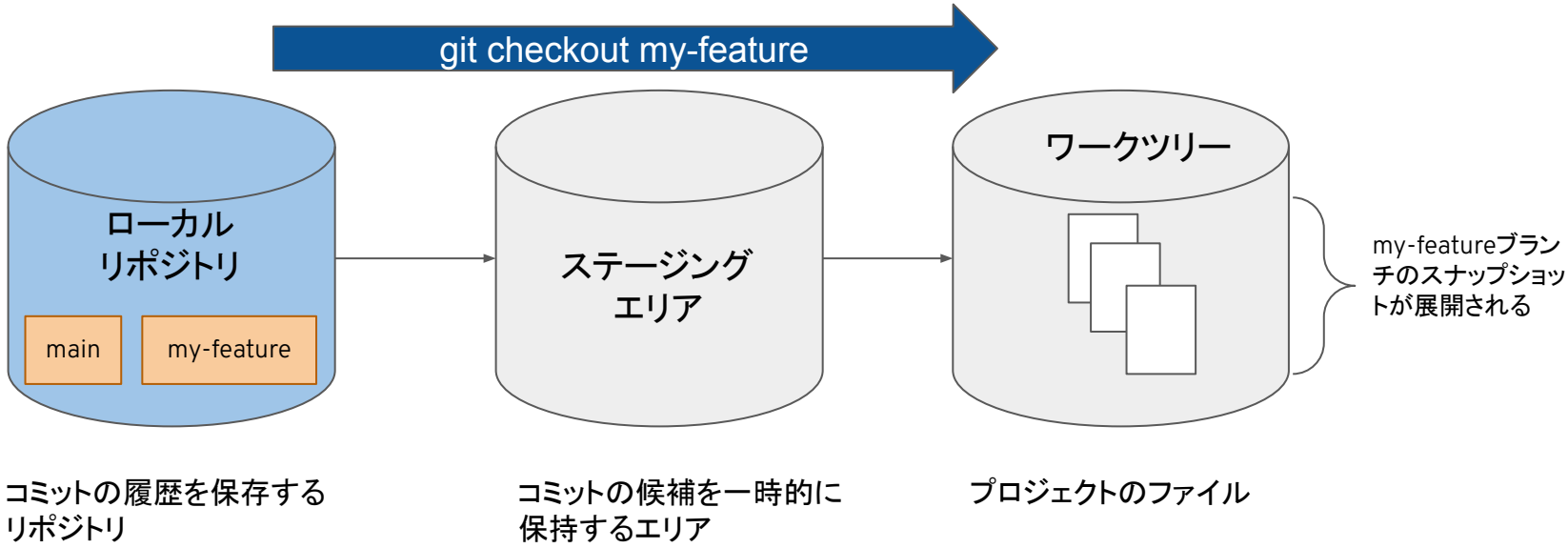
リポジトリの作業では、プライマリーブランチまたは**main ブランチ**を構築することが推奨される。



Red Hat

ブランチの保存方法

Git では、各ブランチのコンテンツが個別に保存されるのではなく、**ブランチの名前とコミットハッシュのペア**が保存される。複数のブランチ間で、現在のブランチを切り替えるには、このブランチの参照を切り替える。ブランチを切り替えると、ブランチが参照するコミットのスナップションが ワークツリーに展開される。



Red Hat

よく使われる Gitコマンド

git config --global user.name "Your User Name" **git config --global user.email your@user.email** **git config --list**	ユーザー名を設定 ユーザーの E メールを設定 ID 設定を確認
git init <ディレクトリ>	ディレクトリの Git リポジトリを初期化する
git status	作業領域とステージング領域のファイルの変更を確認する
git commit -m "コミットメッセージ "	ステージングされたファイルをコミットする
git branch -M main	現在のブランチ master (Git のデフォルト) の名前を mainに変更
git diff	最新の変更の相違点を表示
git add <ファイル> または <ディレクトリ>	変更をステージング領域に追加
git log	リポジトリのコミット履歴を表示
git show	最新のコミットとリポジトリファイルに加えられた変更を表示

ブランチ関連の Git コマンド

git branch <ブランチ名>	ブランチの新規作成
git branch -d <ブランチ名>	指定されたブランチの削除
git checkout <ブランチ名>	ブランチの切り替え
git checkout -b <ブランチ名>	ブランチの新規作成とチェックアウトを一度に実施
git branch	ブランチの一覧表示

GitOps

Red Hat

GitOpsとは

GitOps は、Git リポジトリを信頼できる唯一の情報源として使用し、インフラストラクチャをコードとして提供する。

- アプリケーション開発のための標準的なワークフロー
- アプリケーション要件を事前に設定することによるセキュリティの向上
- Git による可視化とバージョン管理で信頼性を向上
- あらゆるクラスタ、クラウド、オンプレミス環境における一貫性

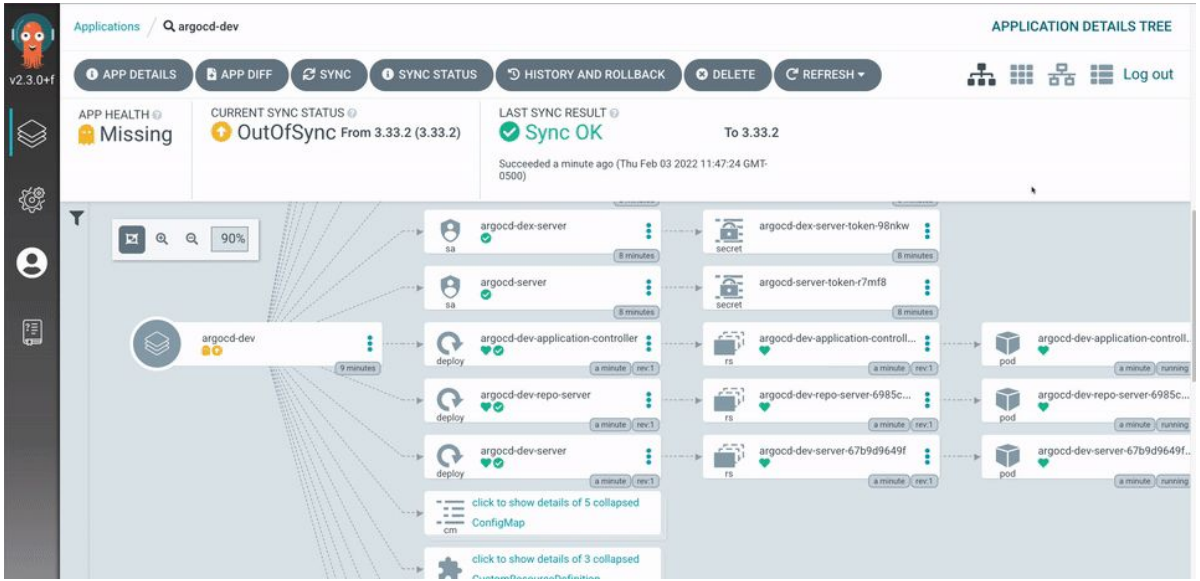
OpenShiftは、ArgoCDを導入するOpenShift GitOps Operatorをサポートする。

OpenShift GitOps
https://docs.redhat.com/ja/documentation/red_hat_openshift_gitops/1.14
OpenShift Operator のライフサイクル
https://access.redhat.com/ja/node/7048793

Red Hat

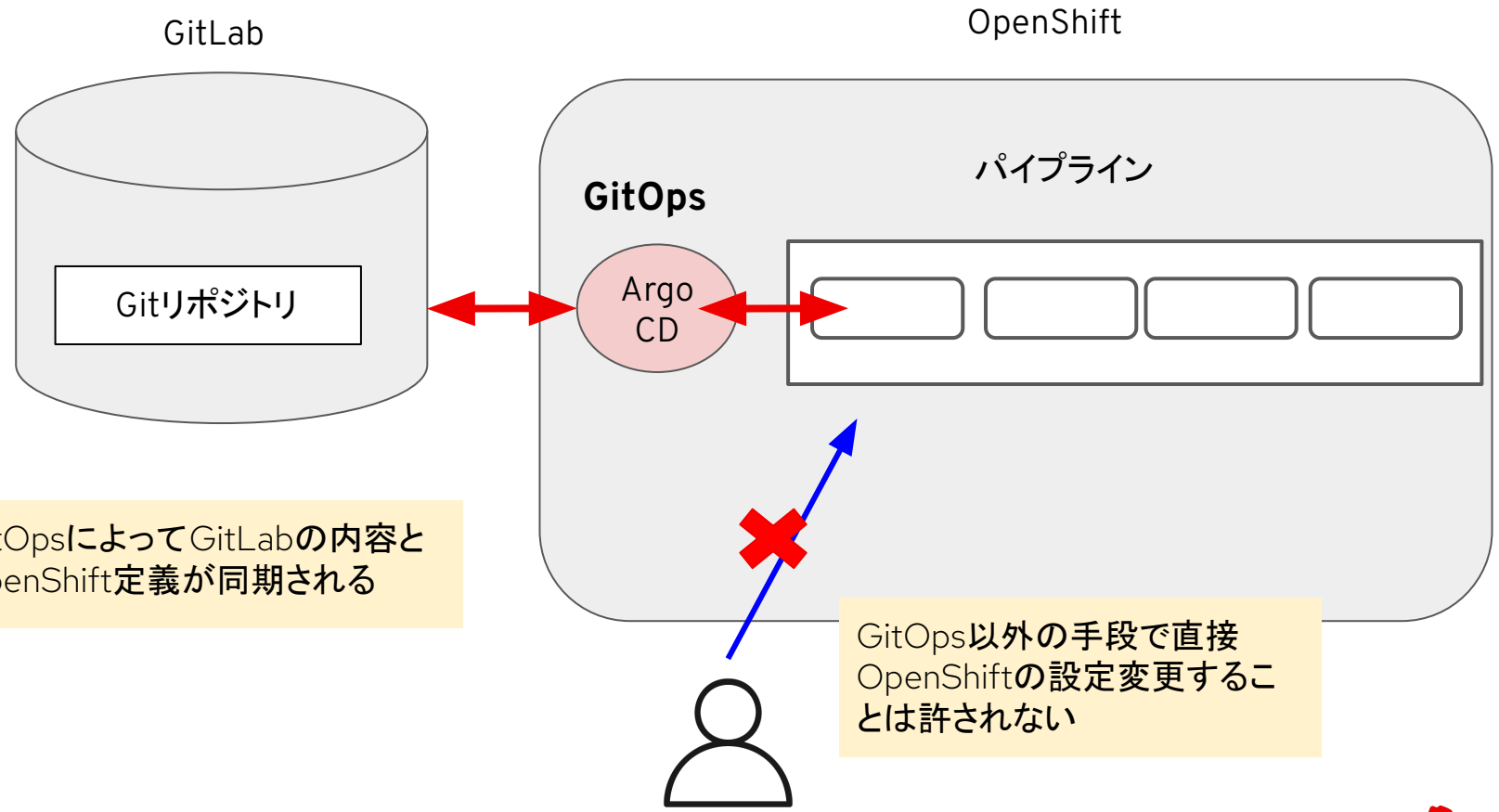
Argo CDとは

Argo CD とは、Kubernetes 向けの宣言型継続的デリバリーツールである。スタンドアロンのツールとしても、必要なリソースをクラスタにデリバリーする CI/CD ワークフローの一部としても使用できる。



Red Hat

OpenShift GitOpsの働き



GitOpsによってGitLabの内容とOpenShift定義が同期される

GitOps以外の手段で直接OpenShiftの設定変更することは許されない

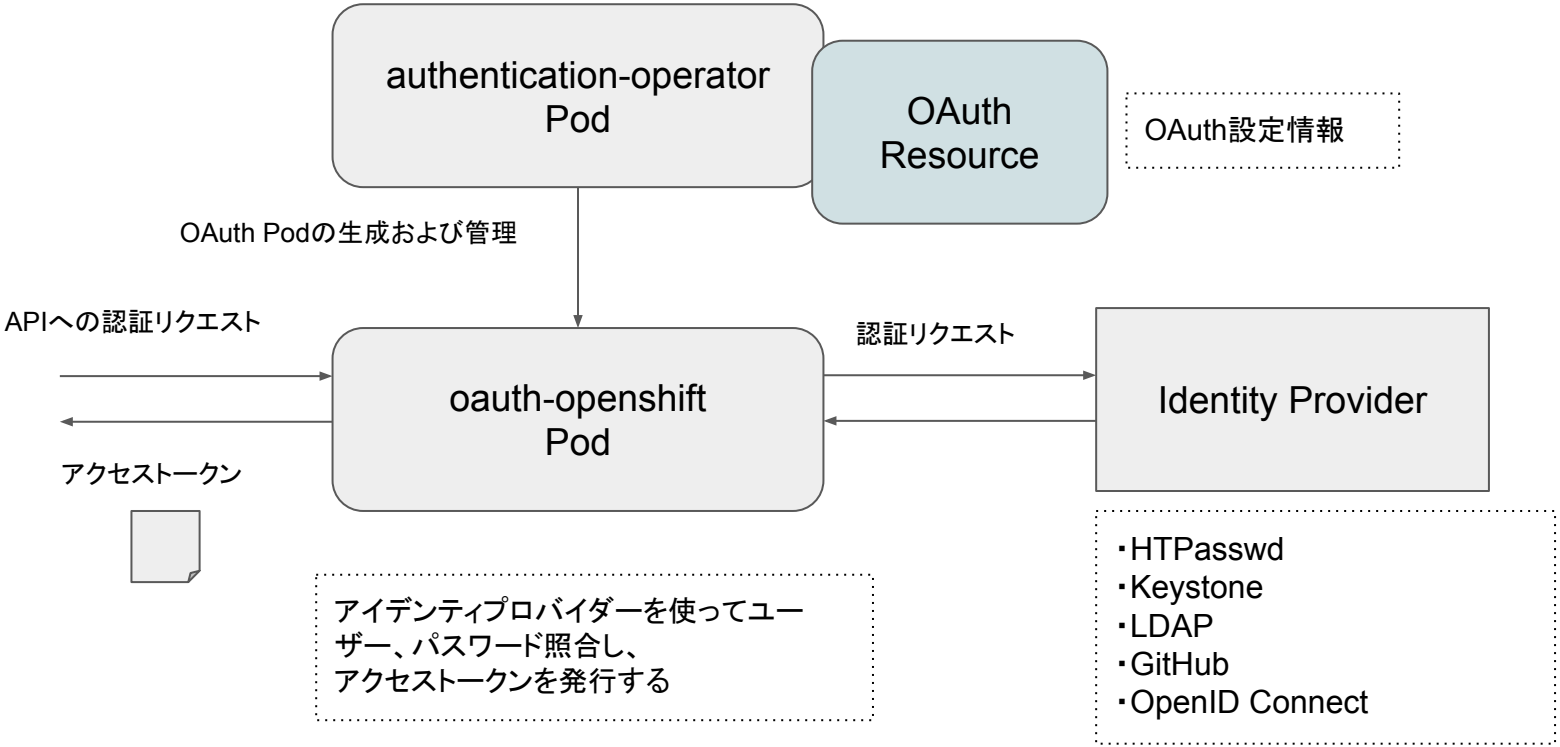
Red Hat

2章 パッケージ化されたアプリケーションのデプロイ

3章 認証と認可

Red Hat

認証Operatorの概要



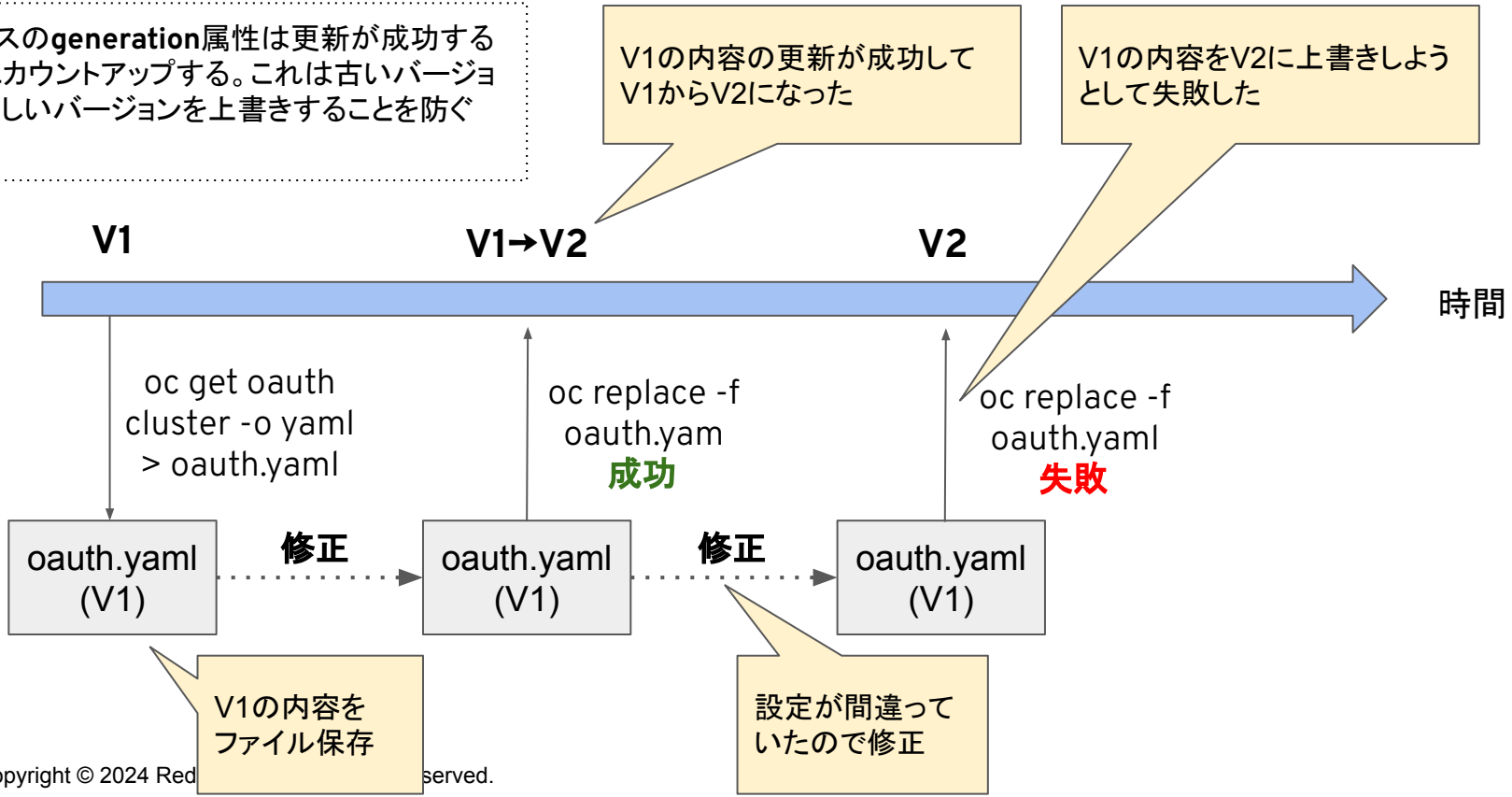
Red Hat

リソースを作成、修正するときにはバリデーションを指定

- YAMLファイルからリソースを作成するとき
  - oc create **--validate --dry-run=server** -f <ファイル名>
- リソースを修正するとき
  - oc apply **--validate --dry-run=server** -f <ファイル名>
- リソースを置き換えるとき
  - oc replace **--validate --dry-run=server** -f <ファイル名>

oauth.yamlのoc replaceに失敗する理由

リソースのgeneration属性は更新が成功するたびにカウントアップする。これは古いバージョンで新しいバージョンを上書きすることを防ぐ



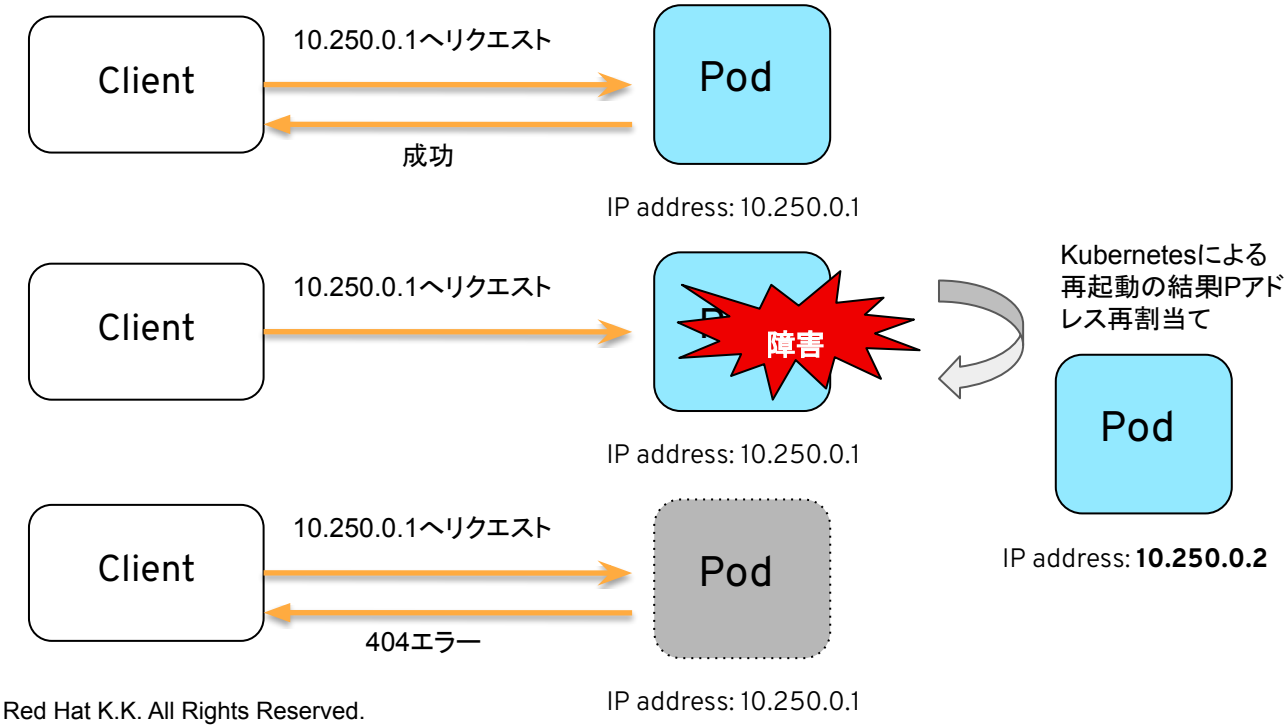
4章 ネットワークセキュリティ

Red Hat

TLSによる内部トラフィックの保護

Podへの通信の問題

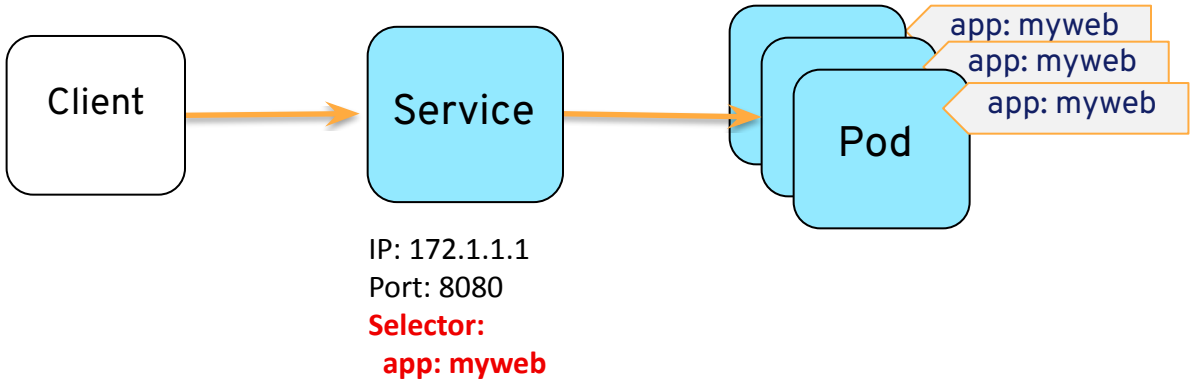
Podは異常終了やハングの場合はKubernetesによって自動的に再起動される。Podには再起動するとIPアドレスが変更されてしまうという特徴があるため、安定したアプリケーション間連携が難しいという問題がある。



Red Hat

Service

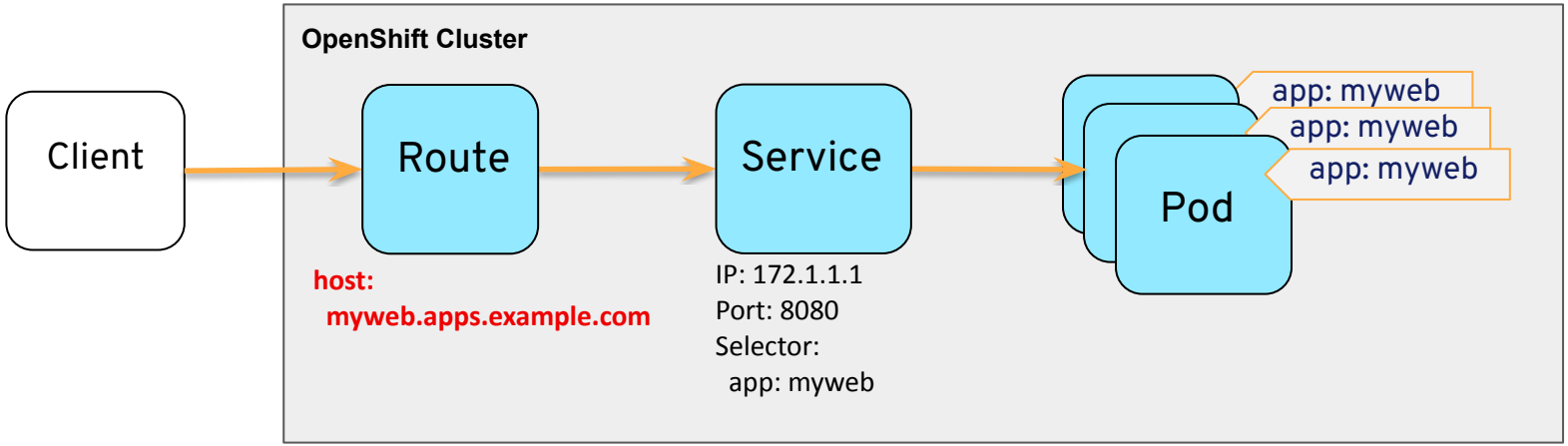
- Serviceは、安定したIPアドレスとポート番号を持ち、Podの負荷分散をおこなう
- Serviceにはセレクターを定義し、セレクターが一致するPodがService配下で管理される
- クライアントからServiceへのリクエストはそのServiceを支えるPodのひとつにルーティングされる
- Podが再起動して別のIPアドレスが割り当てられた場合も、ServiceとPodの間の関係は自動的に維持されるので、クライアントからService経由でPodに通信をすることで安定した通信が可能になる



Red Hat

Route

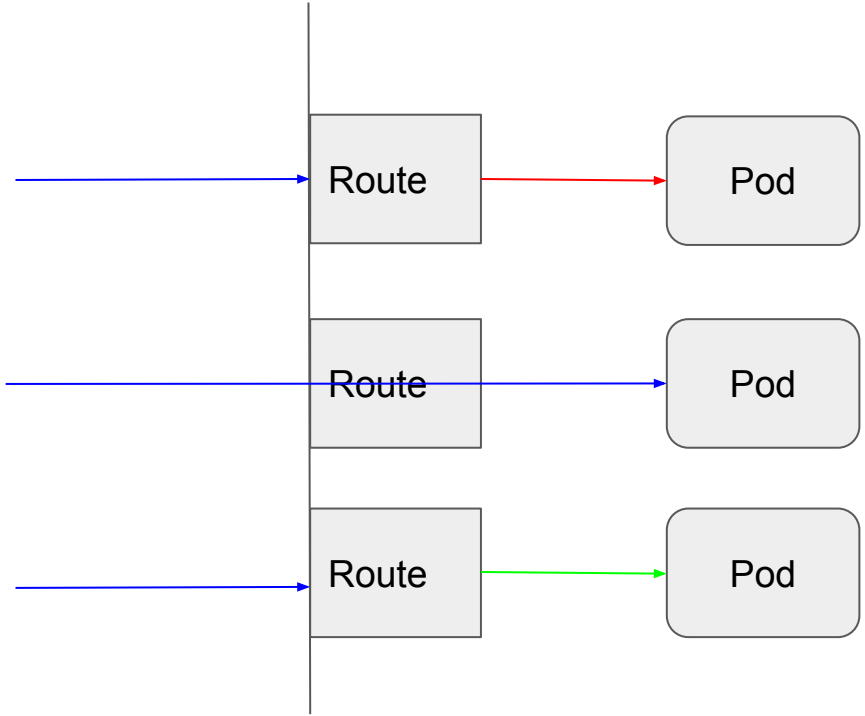
- RouteはOpenShift固有リソースである（対応するKubernetes標準リソースはIngress）。
- ServiceやPodが提供するIPアドレスはソフトウェア定義ネットワーク (SDN) で定義されたものであり、クラスター内のノードでのみ有効である
- Routeはクラスター外部からクラスター内のアプリケーションへの通信を可能にする仕組みである
  - Routeにはホスト名が設定されており、クラスター外からこのホスト名を呼び出す
  - RouteはServiceと関連付けられていて、Routeに送信されたリクエストはServiceに伝達する



Red Hat

OpenShiftの安全なルートの種類

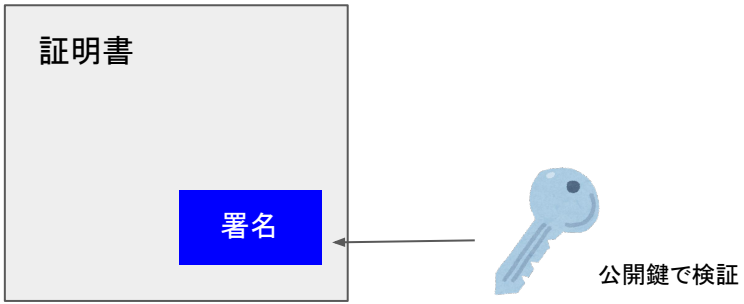
- エッジ
- パススルー
- 再暗号化



Red Hat

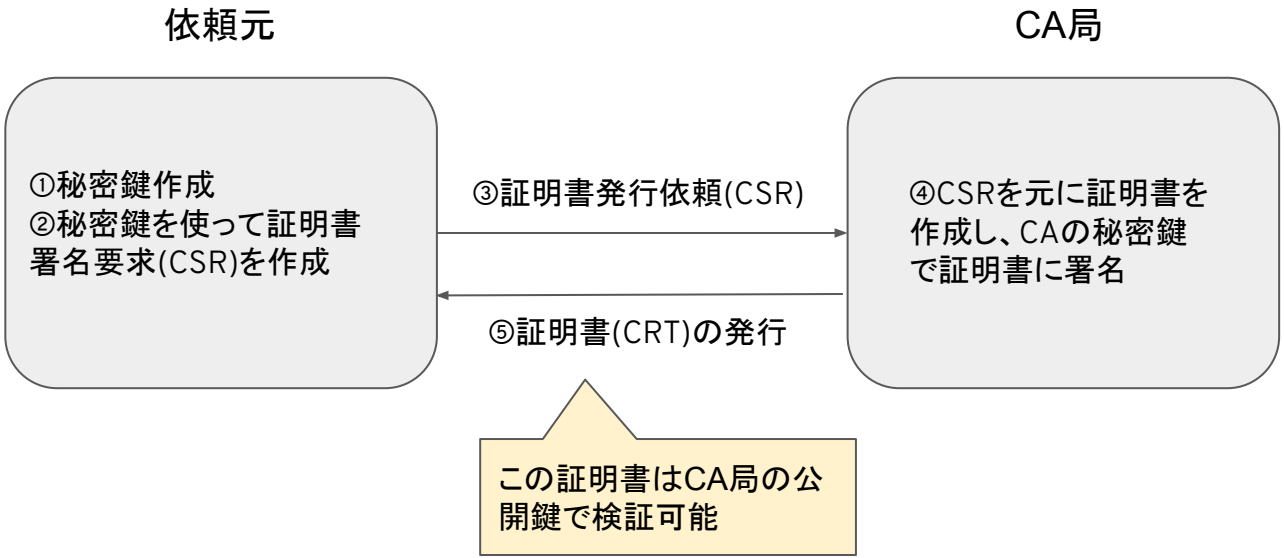
PKI (公開鍵インフラストラクチャ)

- 公開鍵と秘密鍵のペアで暗号化を実現
  - 公開鍵で暗号化したものは、秘密鍵で復号可能（その逆も可）
- 証明書を使うことで通信相手を検証できる
  - CA局が証明書を発行
  - 証明書にはCA局が（CA局の秘密鍵を使って暗号化した）署名を含む
  - 証明書を受け取ったものは、証明書の署名を（ CA局の公開鍵を使って）検証する



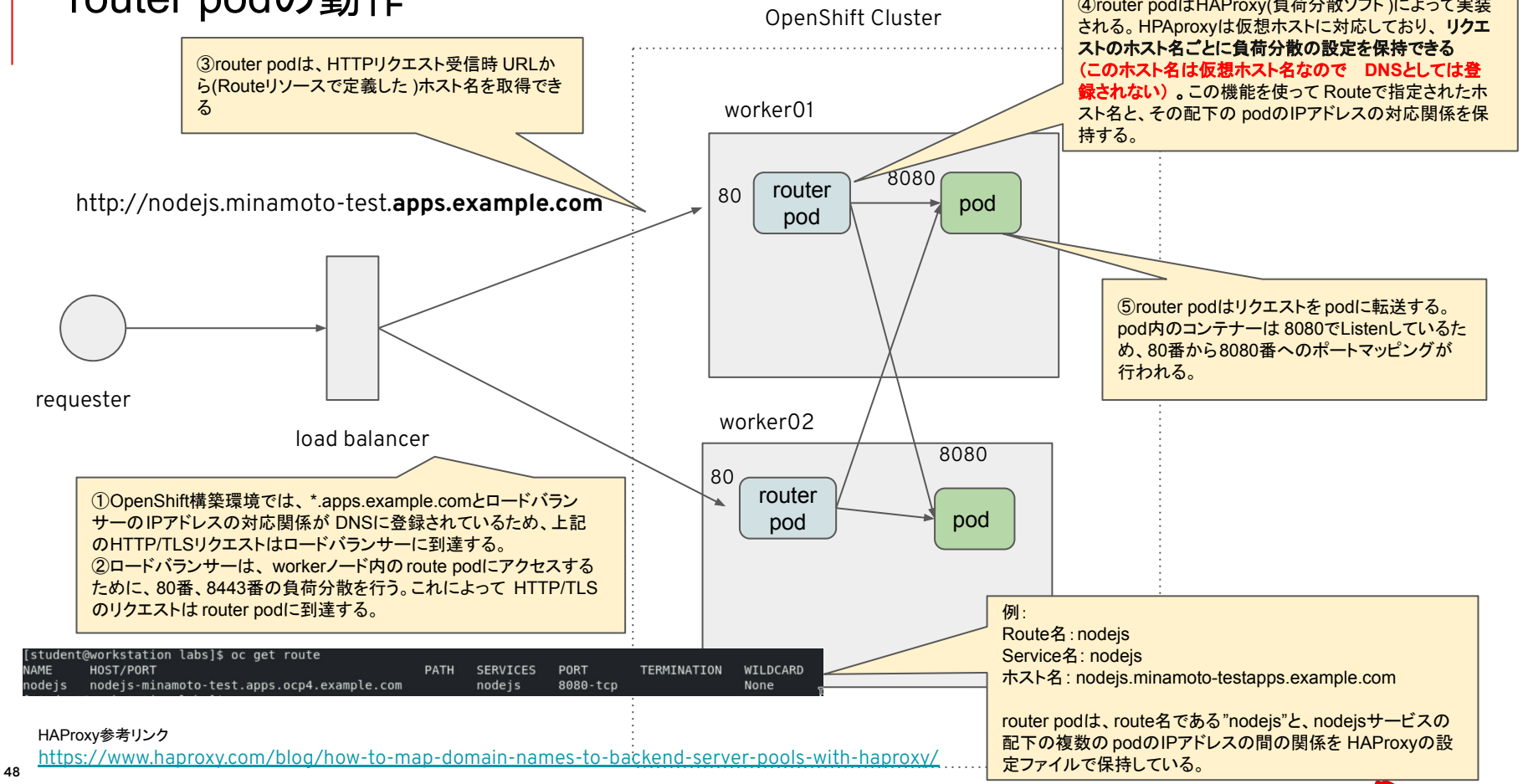
Red Hat

HTTPS/TLSサーバー証明書作成のプロセス



Red Hat

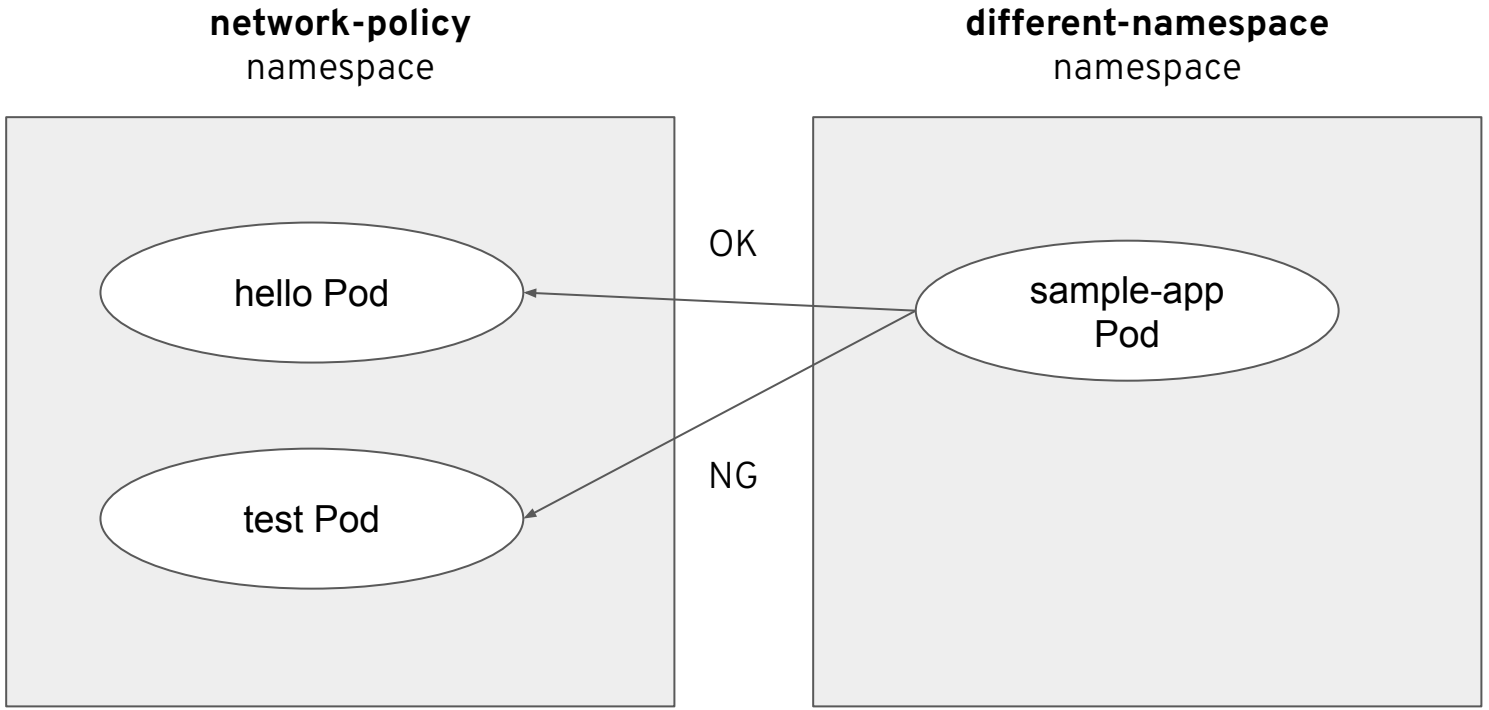
router podの動作



HAProxy参考リンク
https://www.haproxy.com/blog/how-to-map-domain-names-to-backend-server-pools-with-haproxy/

ネットワークポリシーの設定

ネットワークポリシーの設定



Red Hat

5章 非HTTP/SNIアプリケーションの公開

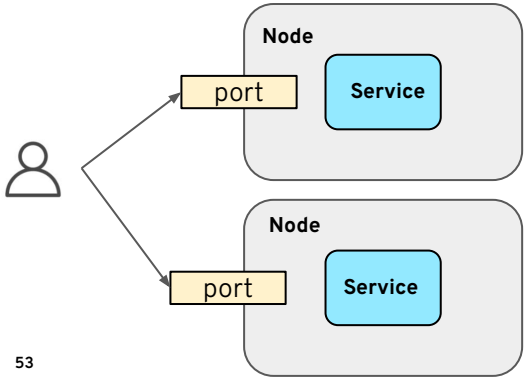
Red Hat

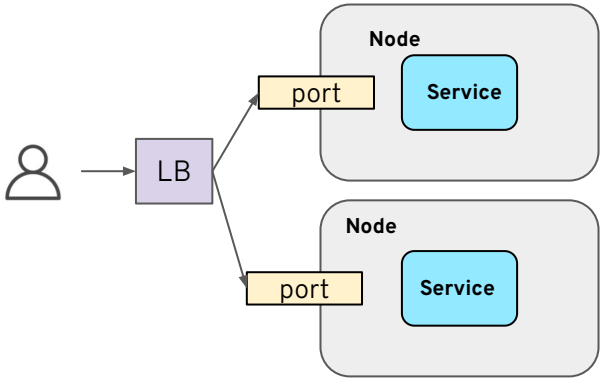
ロードバランサーサービス

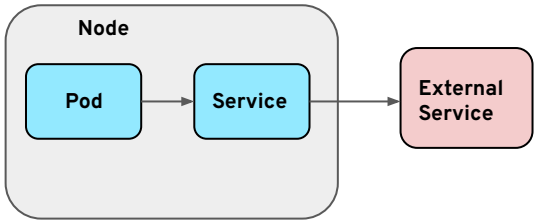
Red Hat

Serviceの種類

タイプ	説明
ClusterIP	サービスの内部Pアドレスを Podに公開
NodePort	ノードのIPアドレス/ポートにアクセスするとサービスに到達 すべてのノードで同じポート番号が公開され、どのノードからもサービスにアクセスできる
LoadBalancer	クラスター外部のロードバランサー経由でサービスに到達 アプリケーションごとにロードバランサーの設定が必要
ExternalName	クラスター外部のサービスにアクセスする







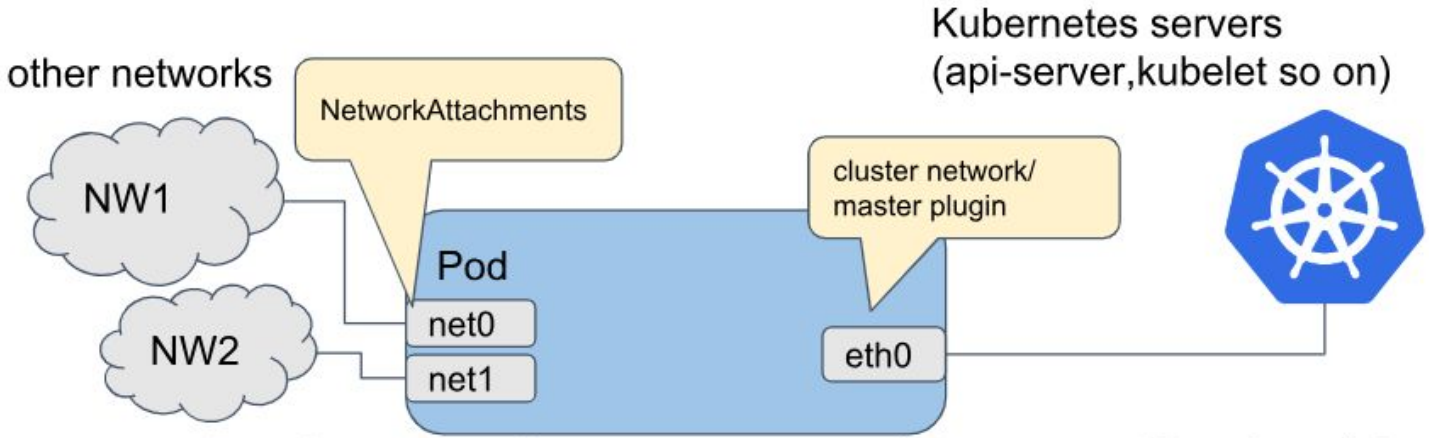
Red Hat

Multus

Red Hat

Multus CNI

- Multus CNIは、複数のネットワークインタフェースをPodにアタッチすることを可能にするコンテナーネットワークインタフェース (CNI) プラグイン
- Multus CNIを使うと、Podに追加のNICを設定して複数のネットワークに接続することができる



https://github.com/k8snetworkplumbingwg/multus-cni
https://www.redhat.com/en/blog/using-the-multus-cni-in-openshift

Red Hat

6章 開発者セルフサービスの有効化

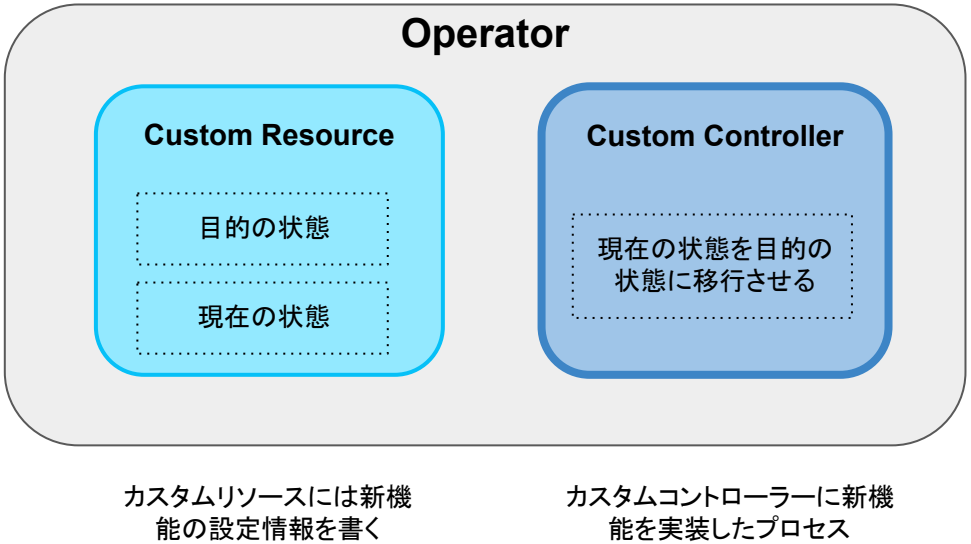
Red Hat

7章 Kubernetes Operatorの管理

Red Hat

Kubernetes Operator

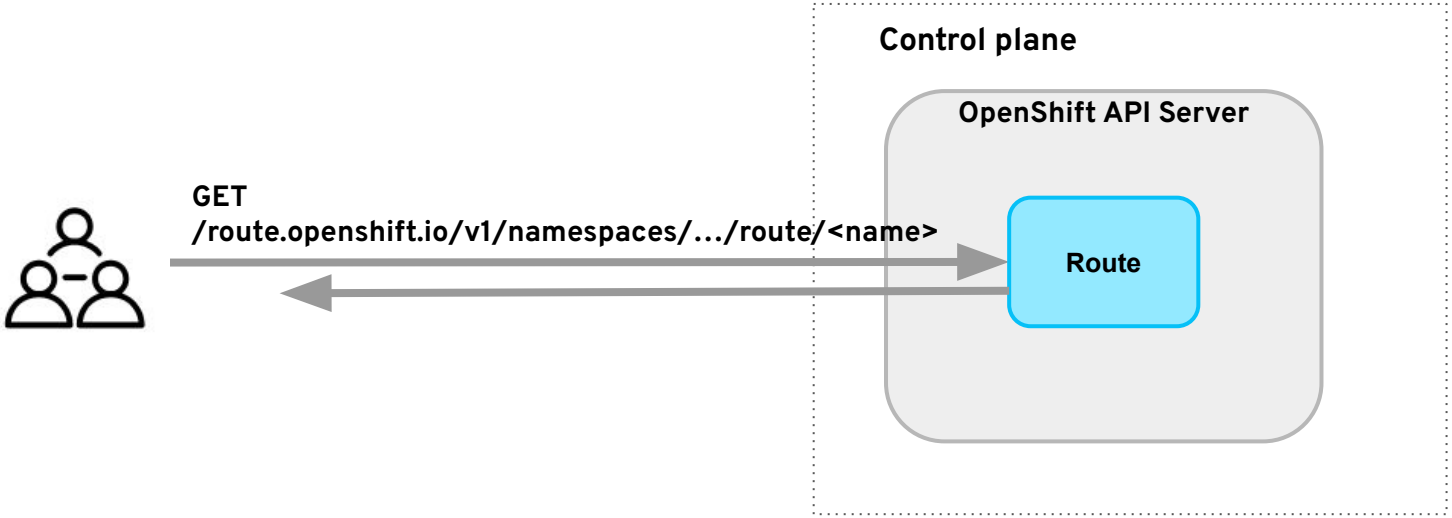
Kubernetesには、Kubernetesの機能を拡張する仕組みとして Kubernetes Operatorがある。Operatorは、カスタムリソースとカスタムコントローラーの組によって新規機能を実現する。カスタムリソースを作ることでAPIも拡張される。OpenShiftの提供機能は、OperatorによってKubernetesの機能を拡張することで実現している。



Red Hat

OperatorによるAPIの拡張

OpenShiftはオペレーターを導入することによって、Kubernetesの機能を拡張する。
OpenShiftが導入したカスタムリソースにアクセスするため、REST APIを拡張する



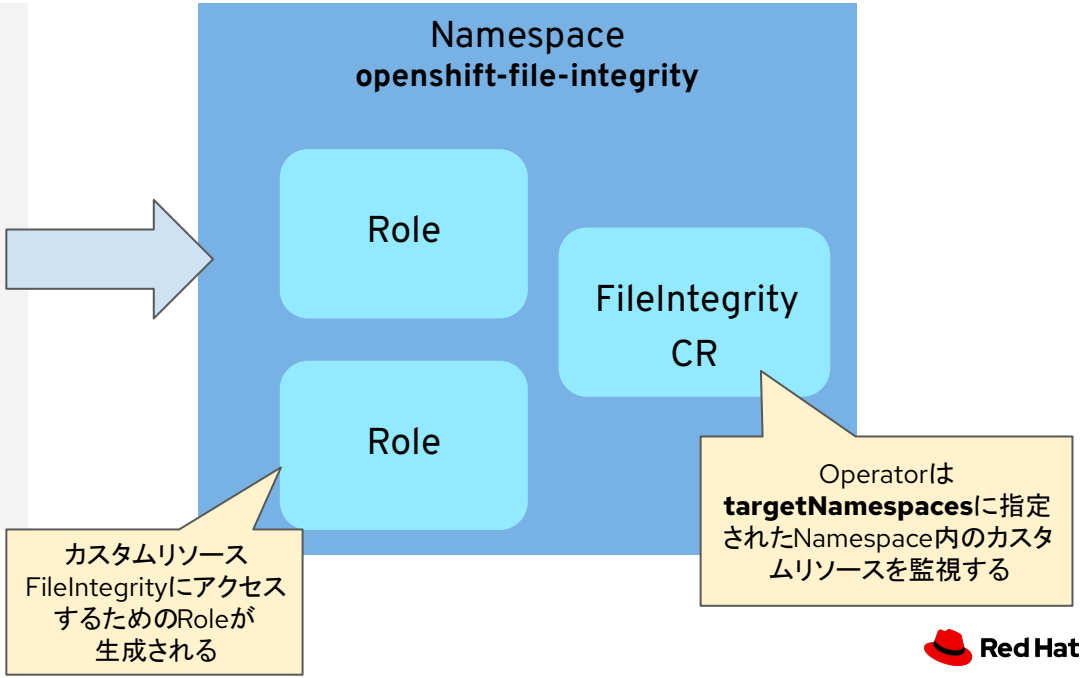
https://docs.openshift.com/container-platform/4.14/rest_api/overview/

Red Hat

OperatorGroup

Operator グループ は、OperatorGroup リソースによって定義され、マルチテナント設定をOLM でインストールされた Operator に提供する。Operator グループは、そのメンバーOperator に必要な RBAC アクセスを生成するために使用するターゲットnamespace を選択する。

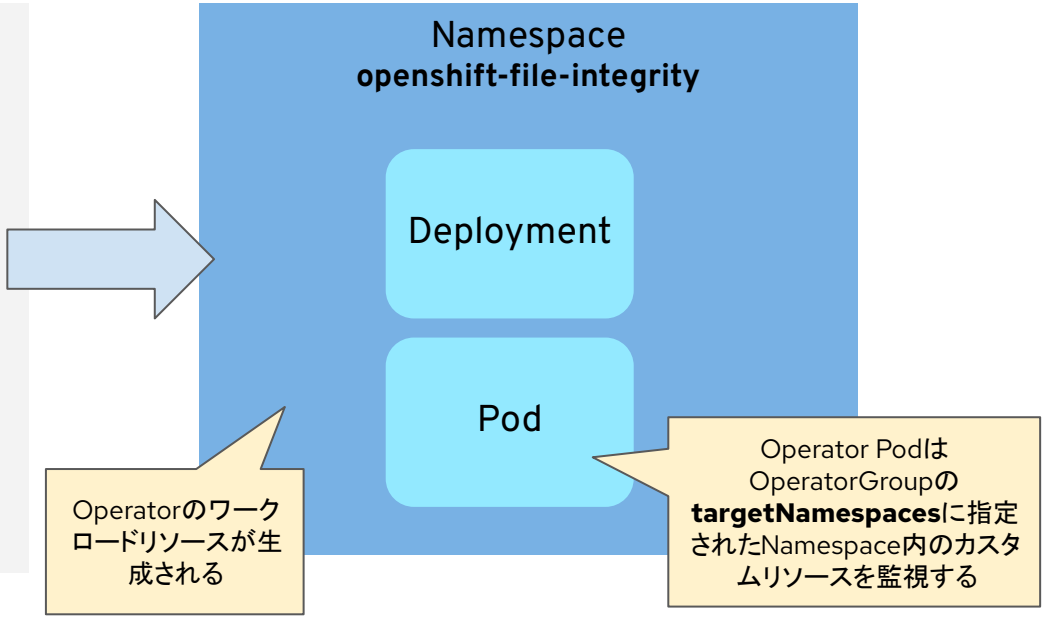
```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: file-integrity-operator
  namespace: openshift-file-integrity
spec:
  targetNamespaces:
  - openshift-file-integrity
```



Subscription

サブスクリプションは、Subscription オブジェクトによって定義され、Operator をインストールする意図を表す。
これは、Operator をカタログソースに関連付けるカスタムリソースである。
サブスクリプションは、サブスクライブするOperator パッケージのチャネルや、更新を自動または手動で実行するかどうかを記述する。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: file-integrity-operator
  namespace: openshift-file-integrity
spec:
  channel: "stable"
  installPlanApproval: Manual
  name: file-integrity-operator
  source: do280-catalog-cs
  sourceNamespace: openshift-marketplace
```



Red Hat

Operator関連のRole/RoleBinding

```
[student@workstation ~]$ oc get role
NAME                                                    CREATED AT
file-integrity-operator.v1.3.3                          2024-11-02T08:46:53Z
file-integrity-operator.v1.3.3-file-integrity-daemon-5c47bd6dd5   2024-11-02T08:46:56Z
file-integrity-operator.v1.3.3-file-integrity-operat-67b46b54df   2024-11-02T08:46:58Z

leader-election-role                                    2024-11-02T08:46:53Z
[student@workstation ~]$ oc get rolebinding
NAME                                                    ROLE                                                                AGE
file-integrity-operator-metrics                         ClusterRole/file-integrity-operator-metrics                         14m
file-integrity-operator.v1.3.3                          Role/file-integrity-operator.v1.3.3                                 14m
file-integrity-operator.v1.3.3-file-integrity-daemon-5c47bd6dd5   Role/file-integrity-operator.v1.3.3-file-integrity-daemon-5c47bd6dd5   13m
file-integrity-operator.v1.3.3-file-integrity-operat-67b46b54df   Role/file-integrity-operator.v1.3.3-file-integrity-operat-67b46b54df   13m
leader-election-rolebinding                             Role/leader-election-role                                           14m
system:deployers                                        ClusterRole/system:deployer                                         14m
system:image-builders                                   ClusterRole/system:image-builder                                    14m
system:image-pullers                                    ClusterRole/system:image-puller                                     14m

[student@workstation ~]$ oc describe rolebinding file-integrity-operator.v1.3.3
Name:         file-integrity-operator.v1.3.3
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  file-integrity-operator.v1.3.3
Subjects:
  Kind            Name                     Namespace
  ----            ----                     ---------
  ServiceAccount  file-integrity-operator
  ServiceAccount  file-integrity-daemon
  ServiceAccount  file-integrity-operator
```


Red Hat

8章 アプリケーションのセキュリティ

Red Hat

7章 クラスターの更新の説明

Red Hat

9章 学習内容の包括的な確認

Red Hat

付録

Red Hat

基本的な oc コマンド

主なリソースのタイプと用途（ネットワーク）

リソース名	意味	補足
Pod	複数のコンテナを含むことができるデプロイの最小単位	動的なIPアドレスを持つ
Service	Podのロードバランサー	静的なIPアドレスを持つ selectorにPodのラベルを指定することでPodと関連付けられる
Route	クラスター外部からPodへのアクセスを可能にする	DNS名を提供する

主なリソースのタイプと用途（デプロイメント）

oc new-appコマンドを使った場合

リソース名	意味	補足
DeploymentConfig	デプロイ設定	ReplicationControllerが自動的に生成される
ReplicationController	Podを指定された数に保つ	指定された個数のPodが自動的に生成される

```
$ oc new-app --name hello-limit --docker-image
quay.io/redhattraining/hello-world-nginx:v1.0
$ oc get dc
```

主なリソースのタイプと用途（デプロイメント）

oc create deployment コマンドを使った場合

リソース名	意味	補足
Deployment	デプロイ設定	ReplicaSetが自動的に生成される
ReplicaSet	Podを指定された数に保つ	指定された個数のPodが自動的に生成される

```
$ oc create deployment hello-limit --image
quay.io/redhattraining/hello-world-nginx:v1.0
$ oc get deploy
```

ocコマンド：ログイン

コマンド	意味	補足
oc login -u kubeadmin -p <password> <API server URL>	kubeadminとしてログインする	
oc login -u <user> -p <password> <API server URL>	一般ユーザとしてログインする	認証が成功するとトークンが発行される
oc whoami	ログイン済ユーザ名を表示	
oc whoami -t	ログイン済ユーザのトークンを表示	
oc logout	ログアウトする	トークンが無効になる

ocコマンド：プロジェクト

コマンド	意味	補足
oc new-project <name>	プロジェクト新規作成	すでに作成済の名前であればエラーになる
oc project	現在プロジェクトを表示	
oc project <name>	現在プロジェクトを指定されたプロジェクトに変更する	
oc projects	プロジェクト名をリスト	自分の権限で見えるものだけ
oc delete project <name1> <name2> ..	プロジェクトを削除する（複数指定可能）	プロジェクトに含まれるすべてのリソースが削除される

ocコマンド：プロジェクトの指定

コマンド	意味	補足

oc get <type> -n <project>	指定されたプロジェクト内で指定されたタイプのリソースを表示	例）
		$ oc get secret -n openshift-config
		$ oc get routes -n openshift-console
		$ oc get machinesets -n openshift-machine-api

oc get <type> <name> -n <project>	指定されたプロジェクト内で指定されたタイプ、名前のリソースを表示	例） $ oc get secret localusers -n openshift-config

ocコマンド：ラベルの指定

コマンド	意味	補足
oc get all -l <key=value>	現在プロジェクト内で指定されたラベルに一致するリソースをすべて表示する	例） $ oc get all -l app=myapp
oc delete all -l <key=value>	現在プロジェクト内で指定されたラベルに一致するリソースをすべて削除する	例） $ oc delete all -l app=myapp

ocコマンド：get

コマンド	意味	補足
oc get all	現在プロジェクト内のすべてのリソースを表示	
oc get <type>	現在プロジェクト内で指定されたタイプのリソースを表示	
oc get <type> <name>	現在プロジェクト内で指定されたタイプ、名前のリソースを表示	
oc get <type> <name> -o wide	現在プロジェクト内で指定されたタイプ、名前のリソースを表示 （表示されるカラムが増える）	Podの場合はスケジュールされたNodeの名前が表示される
oc get <type> <name> -o yaml	現在プロジェクト内で指定タイプ、名前のリソースをYAML形式で表示	この結果をファイルにリダイレクトして編集することが多い

ocコマンド：describe

コマンド	意味	補足
oc describe <type>	現在プロジェクト内で指定されたタイプのリソースを詳細表示する	指定されたタイプのリソースが複数存在する場合は連続表示
oc describe <type> <name>	現在プロジェクト内で指定されたタイプ、名前のリソースを詳細表示	
oc describe pod <name>	現在プロジェクト内で指定された名前のPod詳細情報を表示	IPアドレスの取得など
oc describe dc <name>	現在プロジェクト内で指定された名前のDeploymentConfig詳細情報を表示	レプリカ数、Pod情報（イメージURL, 環境変数, プローブ, リソースリクエスト）の確認

ocコマンド：describeのタイプ別使い方

コマンド	意味	補足
oc describe pod <name>	現在プロジェクト内で指定された名前のPod詳細情報を表示	IPアドレスの取得など
oc describe dc <name>	現在プロジェクト内で指定された名前のDeploymentConfig詳細情報を表示	レプリカ数、Pod情報（イメージURL, 環境変数, プローブ, リソースリクエスト）, アプリのイメージストリームの確認
oc describe svc <name>	現在プロジェクト内で指定された名前のService詳細情報を表示	Endpoints(対応するPod IPアドレスの集まり）の確認
oc describe node <name>	指定された名前の Nodeの詳細情報を表示	ノードラベル、テイント、使用可能な残りリソースの確認

ocコマンド：リソースの作成と編集

コマンド	説明
oc create -f file.yaml	ファイルからリソースを作成
oc create deployment loadtest --dry-run --image quay.io/redhattraining/loadtest:v1.0 -o yaml > file.yaml	コマンドからリソースを作成 (--dry-runはリソースを作成するが実行はしない)
oc edit <type> <name>	現在プロジェクト内で指定されたタイプ、名前のリソースを編集する
1. oc get <type> <name> -o yaml > file.yaml 2. vi file.yaml 3. oc apply -f file.yaml	1.リソースをファイルに保存 2.ファイルを修正 3.修正したファイルを適用

ocコマンド：create、apply、replaceの違い

コマンド	説明
oc create -f file.yaml	ファイルからリソースを新規作成
oc apply -f file.yaml	ファイルの内容をリソースに適用 該当リソースが存在しなければ新規作成
oc replace -f file.yaml	リソースを削除してから、新規作成

ocコマンド：デプロイの修正

コマンド	意味	補足
oc set env deploy/<deployname> -- from secret/<secname>	シークレットから環境変数をコンテナーに設定する	Pod内にコンテナーが複数ある場合は -c でコンテナ名を指定
oc set resources deploy/<deployname> --requests cpu=10m,memory=20Mi --limits cpu=80m,memory=100Mi	リソースリクエストとリミットをコンテナー設定する	Pod内にコンテナーが複数ある場合は -c でコンテナ名を指定
oc set serviceaccount deploy/<deployname> <serviceaccount name>	サービスアカウントを Pod に設定する	

注意
リソースがDeploymentConfigの場合は、タイプはdeploymentconfigまたはdc
リソースがDeploymentの場合は、タイプはdeploymentまたはdeploy

ocコマンド：routeの作成

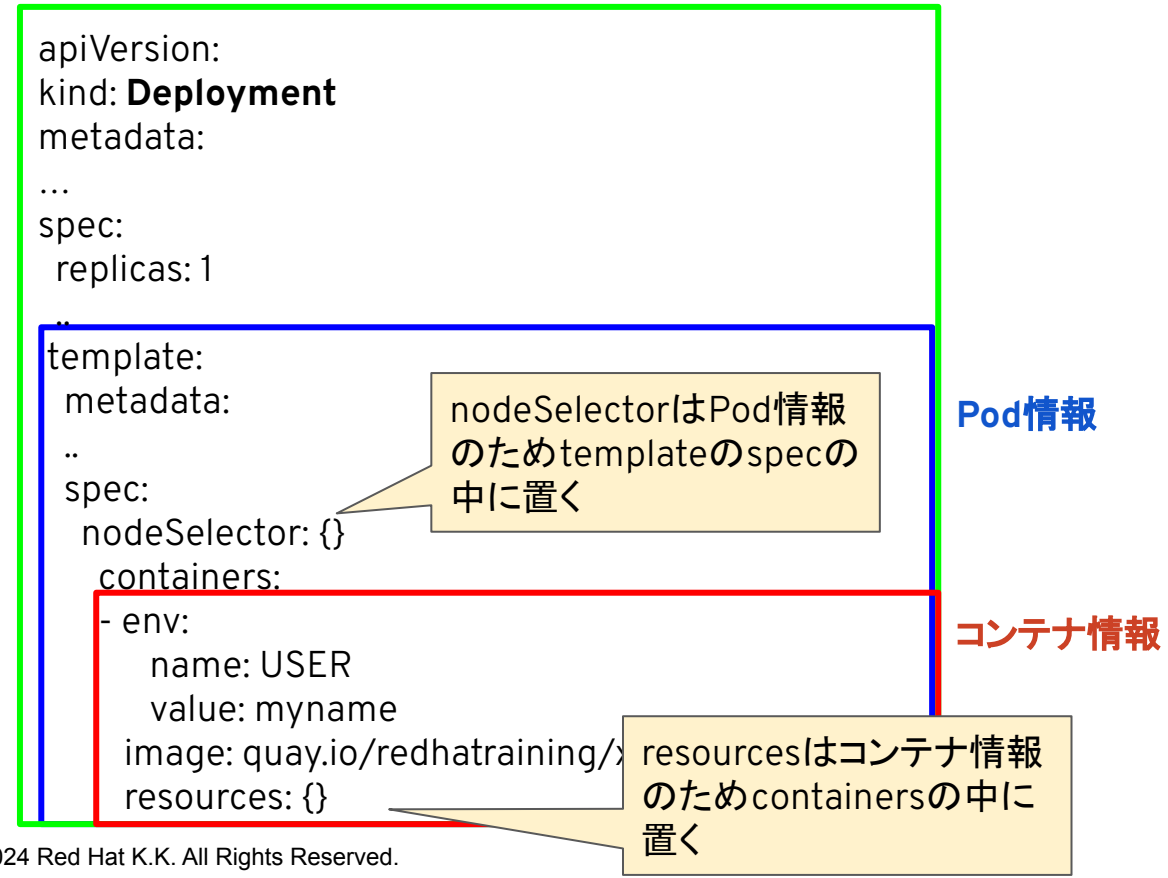
コマンド	意味	補足
oc expose svc <service>	サービス名からルートを作成する	

ocコマンド：delete

コマンド	意味	補足
oc delete <type> <name>	現在プロジェクト内で指定されたタイプ、名前のリソースを削除する	
oc delete all -l <key=value>	現在プロジェクト内で指定されたラベルに一致するリソースをすべて削除する	

Deploymentを修正する箇所

Deploymentを修正する箇所



Deploymentを修正する箇所とその理由

修正箇所	修正箇所	理由
nodeSelector	deploy.spec.template.spec.nodeSelector	スケジュールはPod単位
serviceAccountName	deploy.spec.template.spec.serviceAccountName	実行権限はPod単位
env	deploy.spec.template.spec.containers.env	環境変数はコンテナー単位
requests	deploy.spec.template.spec.containers.resources.requests	リソースリクエストはコンテナ単位

自己署名証明書

自己署名証明書の作成ステップ

コマンド	意味
openssl genrsa -out training.key 2048	秘密鍵作成

openssl req -new -key training.key -out training.csr	証明書署名要求 (CSR) を作成
	
	以下の質問に答える
	`Country Name (2 letter code) [XX]:`
	`State or Province Name (full name) []:`
	`Locality Name (eg, city) [Default City]:`
	`Organization Name (eg, company) [Default Company Ltd]:`
	`Organizational Unit Name (eg, section) []:`
	`Common Name (eg, your name or your server's hostname) []:`

openssl x509 -req -in training.csr -out training.crt -signkey training.key	認証局（CA）の代わりに、自分が作成した秘密鍵で署名して証明書を作成

トラブルシュート

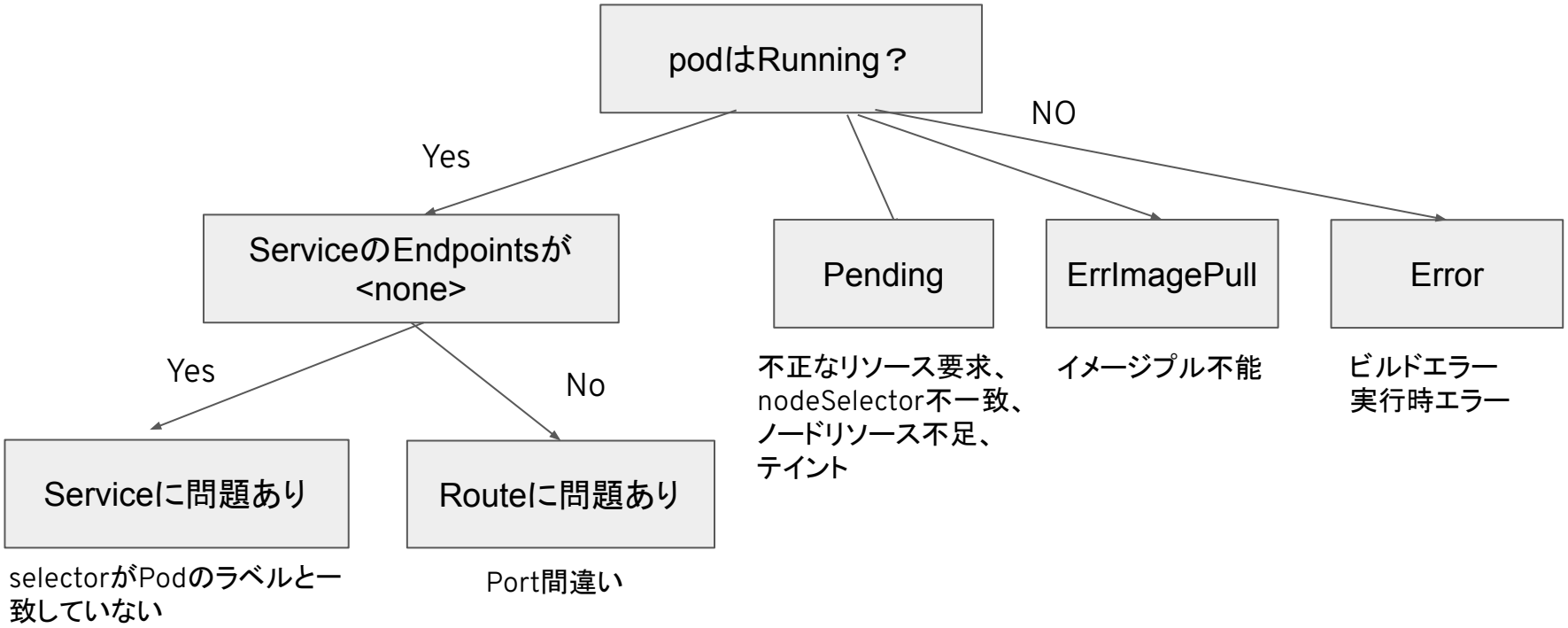
トラブルシュート基本コマンド

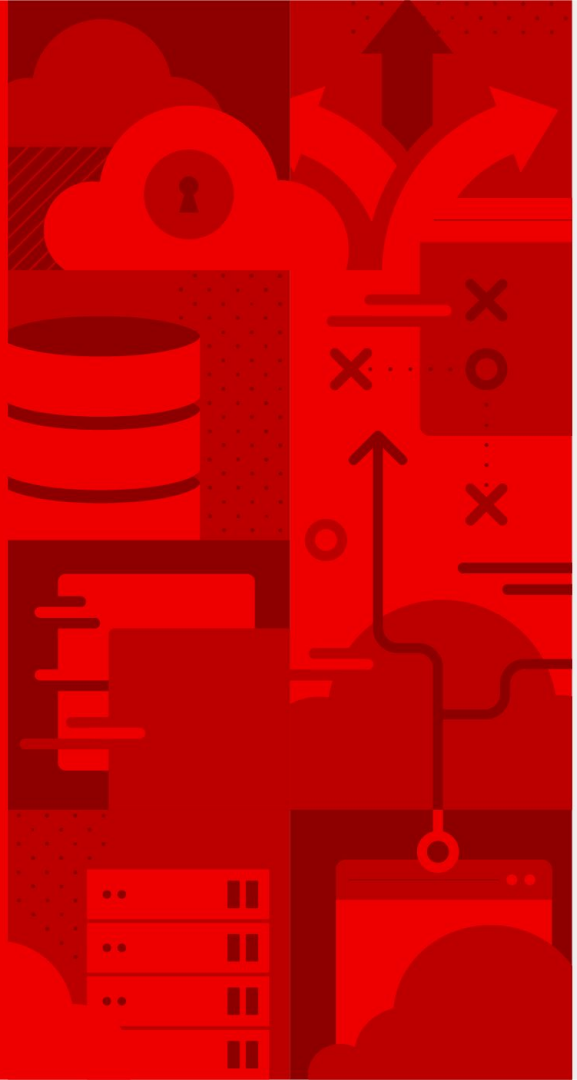
コマンド	意味	補足
oc logs -f <name>	Podのログを表示する	アプリケーションのエラーの原因がわかる
oc get events	イベントを表示する	エラーに至った経緯がわかる
oc describe pod <name>	Podの詳細情報を表示する	Podに関わるイベントを表示する
oc describe node <name>	Nodeの詳細情報を表示する	Nodeに関わるイベントを表示する
oc rsh <name>	コンテナーの中にシェルを開く	コンテナー内部の設定ファイルを確認できる

アプリケーションが起動しない

Podのステータス	意味	調査方法
Pending	スケジューリング失敗	oc describe pod <name> oc get events
ErrImagePull	イメージプル失敗	skopeo inspect
ImagePullBackoff	イメージプル失敗（繰り返し）	skopeo inspect
Error	実行時エラー	oc logs <name>
CrashLoopBackOff	実行時エラー（繰り返し）	oc logs <name>
OOMKilled	メモリ不足による強制終了	pod.spec.containers.resources. requests

Podのステートから原因を分析





Thank you.

https://www.redhat.com/ja/services/training

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHatLabs

Red Hat