

ゴール

Kubernetes環境をローカルで構築し、アプリケーションのデプロイ方法の一種である「ローリングアップデート」と「Blue/Greenデプロイ」を簡単な例で体験する。

(参考) [アプリケーションのデプロイとテストの戦略](#)

目次

- [ゴール](#)
- [目次](#)
- [kubernetes環境構築](#)
 - [kubernetesのインストール](#)
 - [必ず PROXY設定を初期化](#)
 - [Kubernetes有効化](#)
 - [完了状態](#)
 - [Kubernetes Dashboardインストール](#)
 - [インストール](#)
 - [確認](#)
 - [Dashboard管理ユーザー登録とroleのバインド](#)
 - [Ingress-nginxインストール](#)
 - [インストール](#)
 - [確認](#)
 - [イメージファイルのビルド](#)
 - [Dashboard画面を起動](#)
 - [DashboardにログインするためのTOKENを取得](#)
 - [dashboardにアクセスするためのPROXYを起動する](#)
 - [dashboard起動](#)
- [kubernetesの概念](#)
 - [kubernetesクラスタとNode](#)
 - [kubernetesクラスタとNode 概念図](#)
 - [Master Nodeを構成する管理コンポーネント](#)
 - [Dashbord画面で確認](#)
 - [Namespace](#)
 - [Pod](#)
 - [Pod単体](#)
 - [PodのNodeへの配置](#)
 - [Podをデプロイ～動作確認～削除まで](#)
 - [デプロイ](#)
 - [デバッグ用Podのデプロイ](#)
 - [Pod仮想IPについて](#)
 - [SWTESTの情報](#)
 - [DebugコンテナからSWTESTの動作確認](#)
 - [curlコマンドで確認](#)

- Podの削除
- ReplicaSet
 - マニフェストファイルの確認
 - ReplicaSet生成
 - ReplicaSet削除
- Deployment
 - Deployment/ReplicaSet/Pod関連図
 - 世代管理付のデプロイ
 - コマンドラインからデプロイする
 - リビジョンの確認
- ローリングアップデートとロールバック
 - ローリングアップデート
 - マニフェストファイル
 - アップデートの実施
 - 履歴の確認
 - ロールバックの実施
 - 切り替え中の状態
 - 履歴の確認
 - 最後に削除する
- Service(blue/green)作成
 - blue/greenデプロイメント
 - blue/greenサービス
 - デプロイ
 - 状態確認
 - debugコンテナからアクセスし動作確認
- IngressによるBlue/Greenデプロイ
 - IngressによるBlue/Greenデプロイイメージ
 - Ingress-nginxの状況確認
 - Ingress追加
 - 動作確認
 - 現行バージョンへのアクセス
 - 次期バージョンへのアクセス
- Blue -> Green への切り替え
 - path切り替え
 - 新バージョンへのアクセス
 - 旧バージョンへのアクセス
- 最後に実習環境の削除方法(できればKubernetes環境は破棄願います)
 - Kubertenesリソース削除
 - ingress-nginx, dashboard削除
 - Docker Dashboardで無効化
 - 不要データの削除

kubernetes環境構築

kubernetesのインストール

DockerDesktopからKubernetesを有効化します。

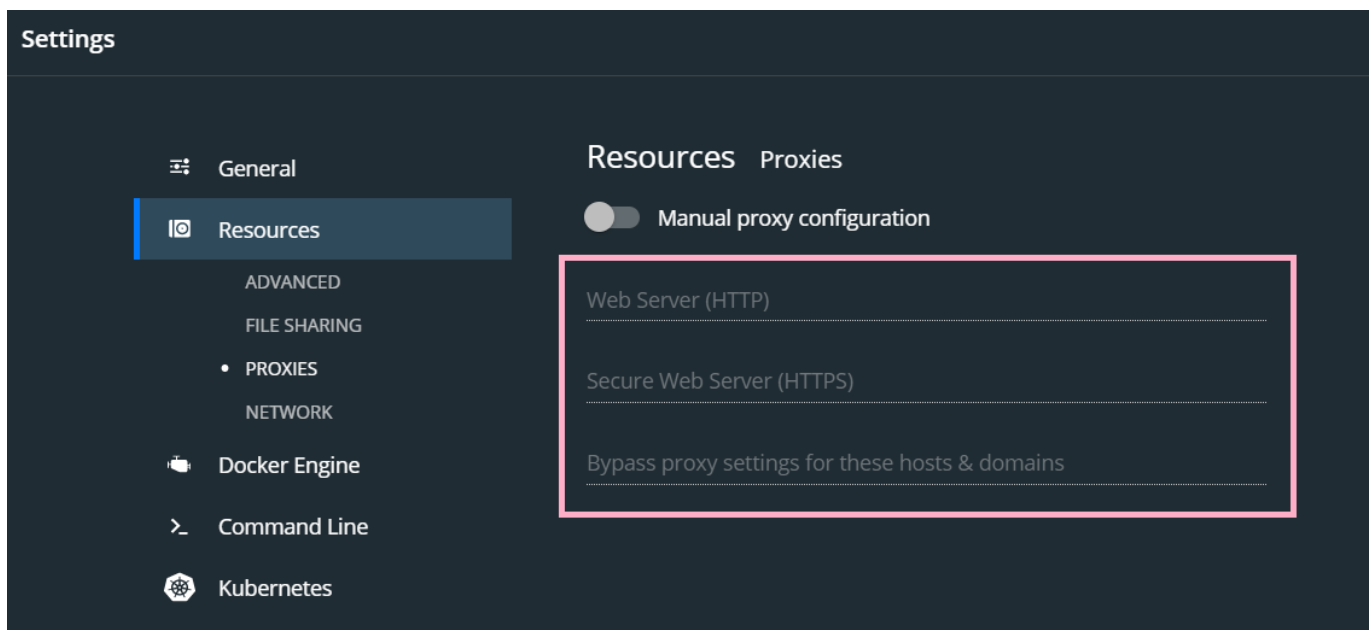
【重要】

社内LAN、VPN環境ではうまく有効化できないので、DockerDesktopのPROXY設定は必ず空にすること。

Kubernetes有効化はiphoneのテザリング環境下、または、自宅のインターネット環境下で実施します。

Kubernetes関連イメージのダウンロード部分で止まり、進退窮まり初期状態にリセットせざるを得なくなります。

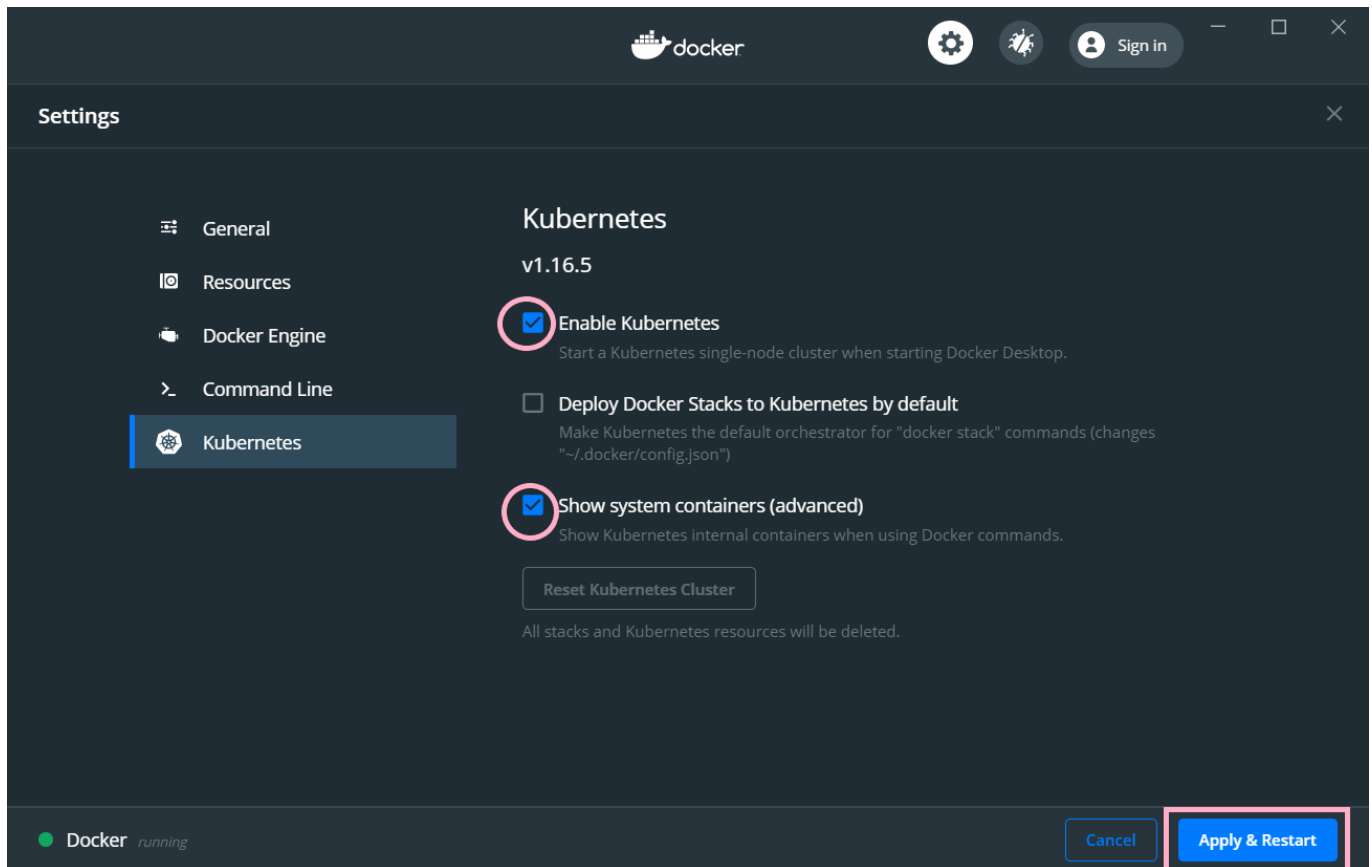
必ず PROXY設定を初期化



SettingsのResourcesでPROXIESは空白にして「Apply&Restart」ボタンを押す

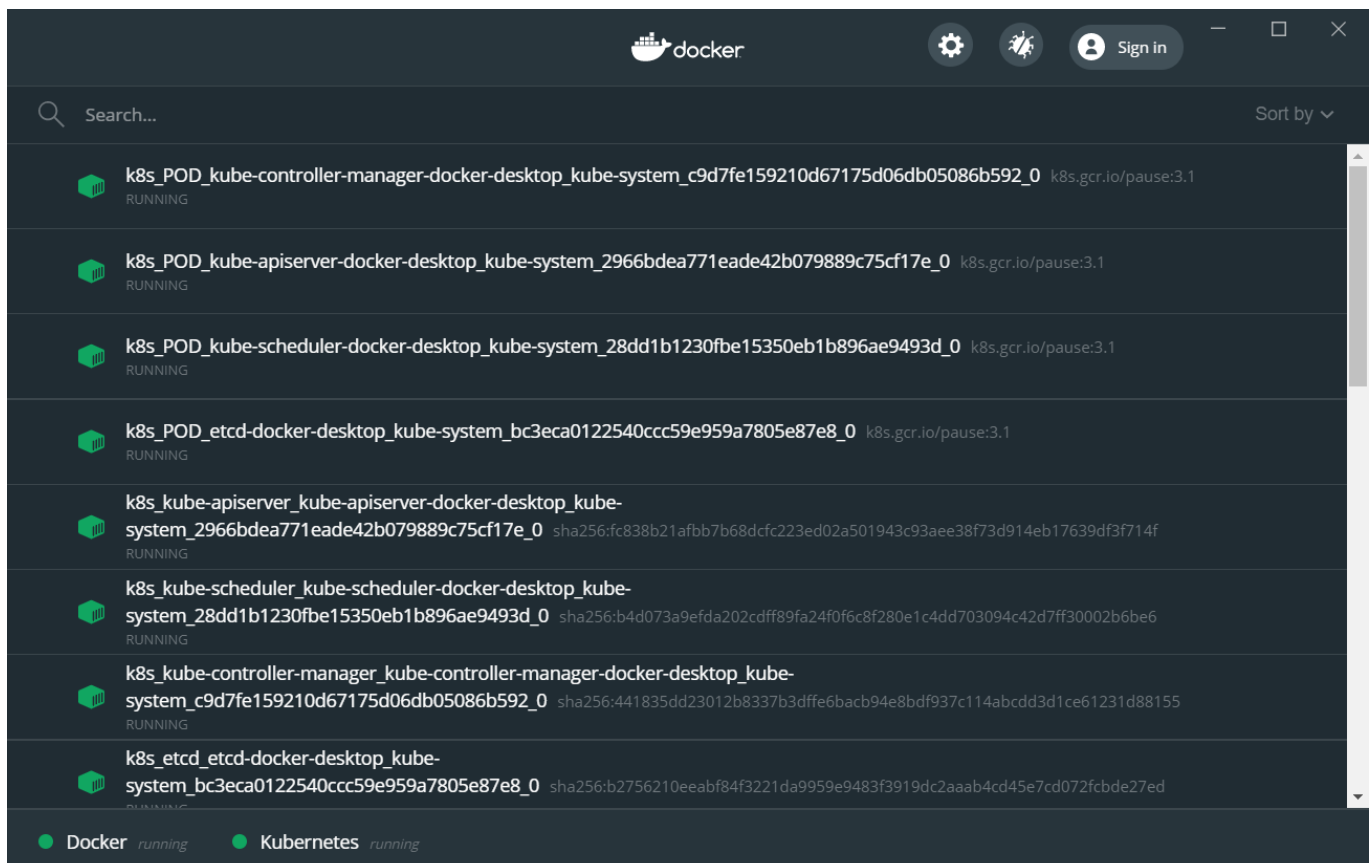
Kubernetes有効化

- Enable Kubernetesにチェックを入れる
- Show system containers(advanced)にチェックを入れる → インストール進捗が分かりやすい
- 「Apply&Reset」ボタンを押す



完了状態

コンテナが全てグリーンになり、画面左下のKubernetesがグリーンになりrunningと表示されれば有効化完了です。



Kubernetes Dashboardインストール

Kubernetesの状況把握や管理のためのダッシュボードソフトをインストールします。

インストール

パワーシェルを起動し、作業フォルダにて次のコマンドを投入する。

```
> kubectl apply -f recommended.yaml
```

確認

ひきつづきパワーシェルにて起動を確認

```
> kubectl get svc -n kubernetes-dashboard
-----
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)
AGE
dashboard-metrics-scraper          ClusterIP           10.111.105.244  <none>           8000/TCP
9h
kubernetes-dashboard               ClusterIP           10.108.122.103  <none>           443/TCP
9h
-----
```

Dashboard管理ユーザー登録とroleのバインド

パワーシェルで次のコマンドを投入する。

```
> kubectl apply -f create_account.yaml

> kubectl apply -f role_binding.yaml
```

Ingress-nginxインストール

2日目に必要なIngress-nginxをあらかじめインストールします。

インストール

パワーシェルを起動し、作業フォルダにて次のコマンドを投入する。

```
> kubectl apply -f mandatory.yaml
-----
namespace/ingress-nginx created
deployment.apps/default-http-backend created
```

```

service/default-http-backend created
configmap/nginx-configuration created
configmap/tcp-services created
configmap/udp-services created
serviceaccount/nginx-ingress-serviceaccount created
clusterrole.rbac.authorization.k8s.io/nginx-ingress-clusterrole created
role.rbac.authorization.k8s.io/nginx-ingress-role created
rolebinding.rbac.authorization.k8s.io/nginx-ingress-role-nisa-binding created
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress-clusterrole-nisa-binding created
deployment.apps/nginx-ingress-controller created
-----

> kubectl apply -f cloud-generic.yaml
-----
service/ingress-nginx created
-----

```

確認

ひきつづきパワーシェルにて起動を確認

```

> kubectl get svc -n ingress-nginx
-----
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)
AGE
default-http-backend               ClusterIP           10.105.175.142  <none>           80/TCP
2m27s
ingress-nginx                      LoadBalancer       10.105.219.79   localhost
9000:31923/TCP,443:31166/TCP      101s
-----

```

イメージファイルのビルド

予め今回使用するイメージファイルをビルド・PULLしておきます。

パワーシェルにて下記バッチを起動

```

> .\build_image.bat
-----
:
:
REPOSITORY      TAG                IMAGE ID          CREATED
SIZE
swtest/nginx    latest            8d71d4b00e5f     28 hours ago
170MB

```

REPOSITORY SIZE	TAG	IMAGE ID	CREATED
swtest/api 121MB	v2	51e06a4c2b89	5 hours ago
swtest/api 121MB	latest	6380831691ba	28 hours ago
REPOSITORY SIZE	TAG	IMAGE ID	CREATED
swtest/debug 8.67MB	latest	421bf1f5871d	About a minute ago

Dashboard画面を起動

DashboardにログインするためのTOKENを取得

【重要】 次のコマンドは必ずパワースhellで実行すること。

```
> kubectl -n kubernetes-dashboard describe secret $(kubectl -n kubernetes-
dashboard get secret | sls admin-user | ForEach-Object { $_ -Split '\s+' } |
Select -First 1)

-----
Name:          admin-user-token-8db5m
Namespace:     kubernetes-dashboard
Labels:        <none>
Annotations:   kubernetes.io/service-account.name: admin-user
               kubernetes.io/service-account.uid: 7b982cb1-97a4-4c83-ae58-
c0514d77d656

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:      1025 bytes
namespace:   20 bytes
token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IiA1SDBab2ozYlZ4cDY4Ry1lMXE4WGc5SW5PcmgxY19fYUVFY3NyZl
pCTVUifQ.....
-----
```

dashboardにアクセスするためのPROXYを起動する

```
>kubectl proxy
```

```
-----  
Starting to serve on 127.0.0.1:8001  
-----
```

dashboard起動

下記URLをブラウザで開く

```
http://localhost:8001/api/v1/namespaces/kubernetes-  
dashboard/services/https:kubernetes-dashboard:/proxy/
```

Kubernetes Dashboard

☐ Kubeconfig

クラスターにアクセスするために作成した kubeconfig ファイルを選択してください。kubeconfig ファイルの設定方法や使用方法についてもっと知るためには、[Configure Access to Multiple Clusters](#) セクションを参照してください。

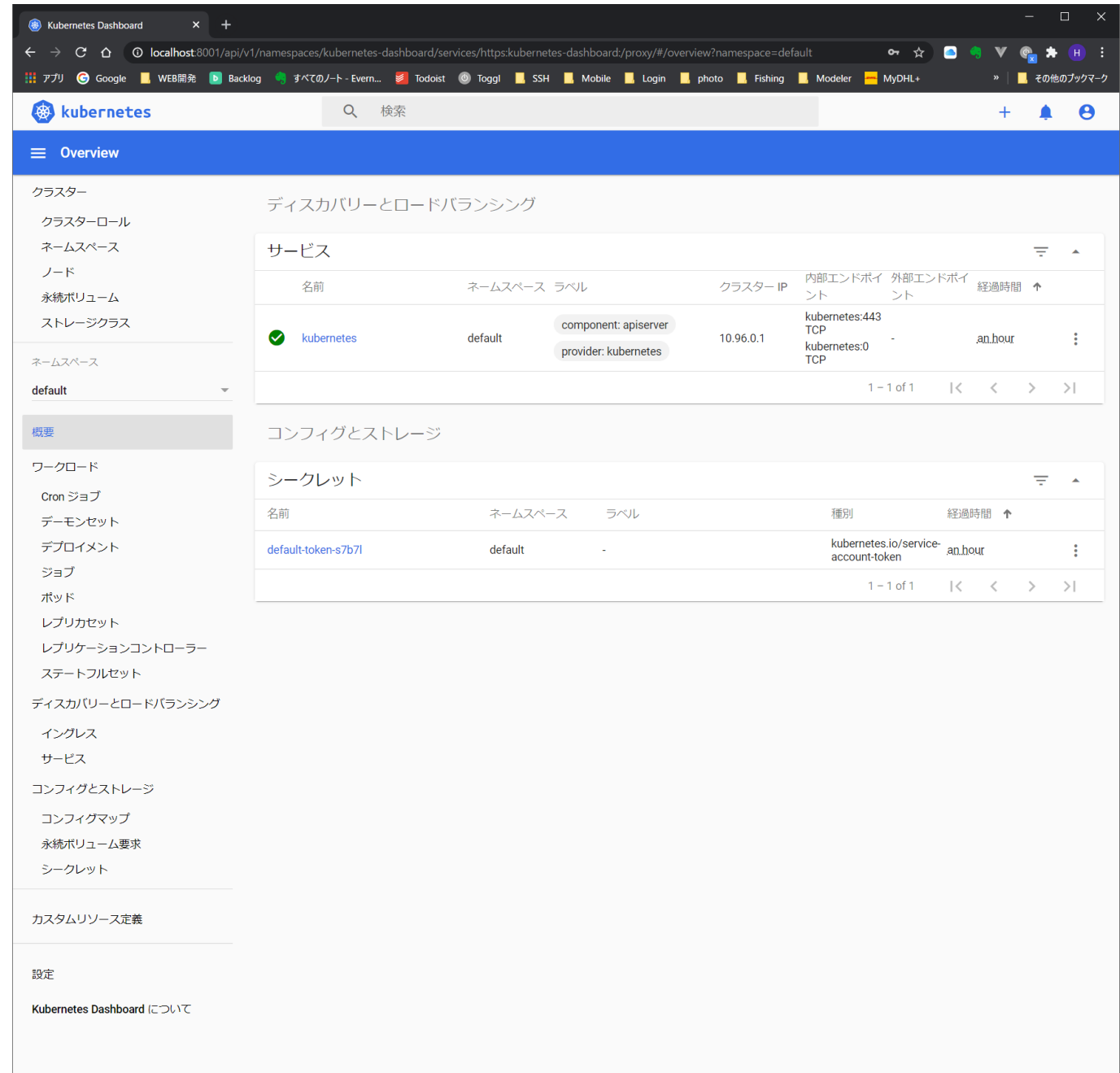
☒ トークン

すべてのサービスアカウントには、ダッシュボードのログインに使用できる有効なベアータークンを持つシークレットがあります。ベアータークンの設定方法や使用方法についてもっと知るためには、[Authentication](#) セクションを参照してください。

トークンを入力 *

サインイン

- トークンを選択する
- パワーシェル画面に表示されているTOKENをコピーし、WEB画面のトークン部分にペーストする
- 「サインイン」ボタンをクリックする



kubernetesの概念

kubernetesクラスタで実行されるアプリケーションは様々なリソースと協調して動作することで成立している。(コンテナ・オーケストレーションと呼ばれる)

以下にkubernetesのリソース一覧をまとめる。*付は今回実習で体験する。

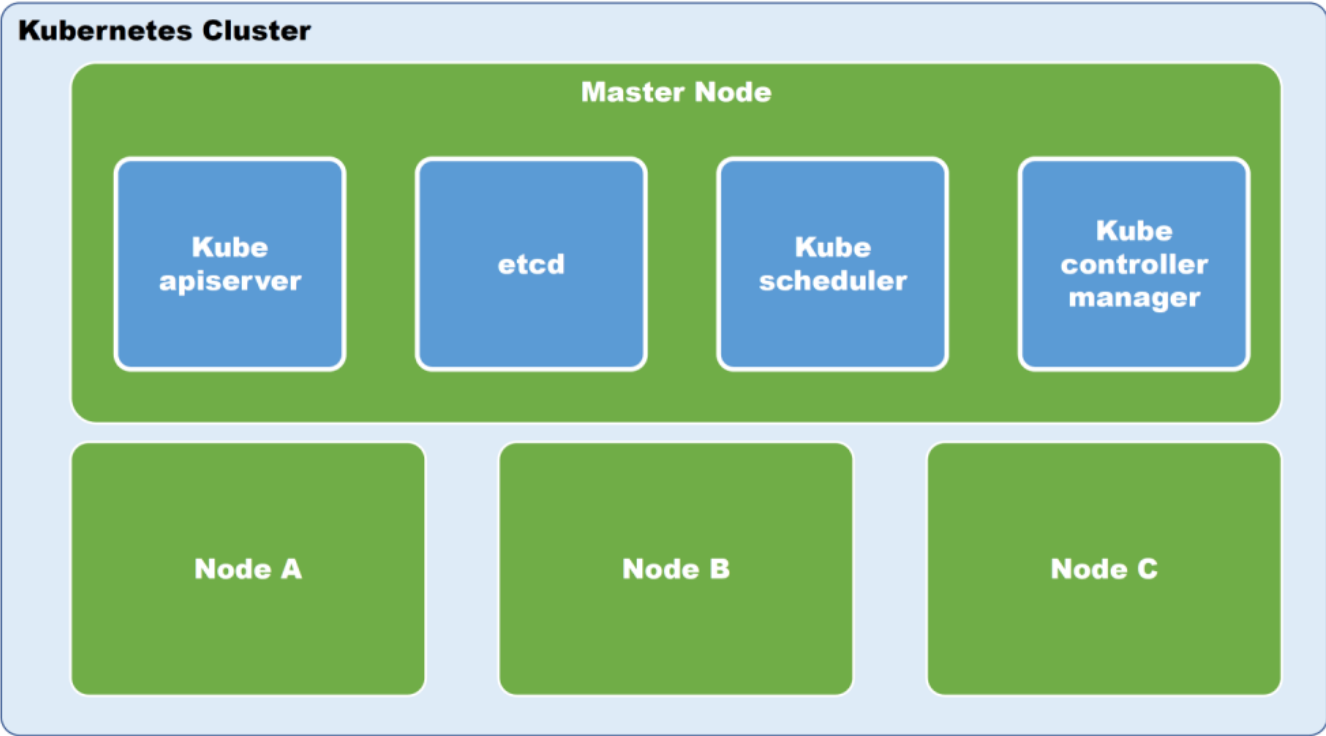
リソース名	用途
Node*	Kubernetesクラスタで実行するコンテナを配置するためのサーバー
Namespace*	Kuberneteクラスタ内で作る仮想的なクラスタ
Pod*	コンテナの集合体の単位。コンテナを実行する方法を定義する
ReplicaSet*	同じ仕様のPodを複数生成・管理する

リソース名	用途
Deployment*	ReplicaSetの世代を管理する
Service*	Podの集合にアクセスするための経路を定義する
Ingress*	ServiceをKubernetesの外部に公開する
ConfigMap	設定情報を定義し、Podに供給する
PersistentVolume	Podが利用するストレージのサイズや種別を定義する
PersistentVolumeClaim	PersistentVolumeを動的に確保する
StorageClass	PersistentVolumeが確保するストレージの種類を定義する
StatefulSet	同じ仕様で一貫性のあるPosを複数生成・管理する
Job	常駐目的ではない複数のPodを作成し、正常終了することを保証する
CronJob	cron記法でスケジューリングして実行されるJob
Secret	認証情報等の機密データを定義する
Role	Namespace内で操作可能なKubertenesリソースのルールを定義する
RoleBinding	RoleとKubernetesリソースを利用するユーザーを紐付ける
ClusterRole	Cluster全体で操作可能なKubertenesリソースのルールを定義する
ClusterRoleBinding	ClusterRoleとKubernetesリソースを利用するユーザーを紐付ける
ServiceAccont	PodにKubernetesリソースを操作させる際に利用するユーザー

kubernetesクラスタとNode

- KubernetesクラスタはKubernetesの様々なリソースを管理する集合体のこと。
- その中で最も大きい概念がNode（ホストのこと）となる。
- Kubernetesクラスタには全体を管理する役割のMaster Nodeが少なくとも一つは必要。

kubernetesクラスタとNode 概念図



Master Nodeを構成する管理コンポーネント

コンポーネント名	役割
kube-apiserver	KubernetesのAPIを公開するコンポーネント。kubectlからのリソース操作を受け付ける
etcd	高可用性を備えた分散キーバリューストア。Kubernetesのバックングストアとして利用される
kube-scheduler	Nodeを監視し、コンテナを配置する最適なNodeを選択する
kube-controller-manager	リソースを制御するコントローラを実行

Dashbord画面で確認

- ネームスペース : kube-system を選択
- 概要をクリック

The screenshot displays the Kubernetes Dashboard interface. On the left sidebar, the 'kube-system' namespace is selected. The main content area shows the 'Overview' for this namespace. A blue box highlights the 'Pods' section, which lists the following pods:

名前	ラベル	ノード	状態	再起動	CPU 使用量 (コア数)	メモリー使用量 (バイト)	経過時間
etcd-docker-desktop	component: etcd tier: control-plane	docker-desktop	Running	0	-	-	a_day
vpnkit-controller	component: vpnkit-controller	docker-desktop	Running	0	-	-	a_day
storage-provisioner	component: storage-provisioner	docker-desktop	Running	1	-	-	a_day
kube-apiserver-docker-desktop	component: kube-apiserver tier: control-plane	docker-desktop	Running	0	-	-	a_day
kube-scheduler-docker-desktop	component: kube-scheduler tier: control-plane	docker-desktop	Running	0	-	-	a_day
kube-controller-manager-docker-desktop	component: kube-controller-manager tier: control-plane	docker-desktop	Running	0	-	-	a_day
kube-proxy-vvpjv	controller-revision-hash: 759676c746 k8s-app: kube-proxy	docker-desktop	Running	0	-	-	a_day
coredns-5644d7b6d9-h8dzt	k8s-app: kube-dns pod-template-hash: 5644d7b6d9	docker-desktop	Running	0	-	-	a_day
coredns-5644d7b6d9-kcv9r	k8s-app: kube-dns pod-template-hash: 5644d7b6d9	docker-desktop	Running	0	-	-	a_day

Below the pods list, the 'ReplicaSets' and 'Services' sections are visible. The 'Services' section shows the 'kube-dns' service with the following details:

名前	ラベル	クラスター IP	内部エンドポイント	外部エンドポイント	経過時間
kube-dns	k8s-app: kube-dns kubernetes.io/cluster-service: true	10.96.0.10	kube-dns.kube-system:53 UDP kube-dns.kube-system:53 TCP	-	a_day

Namespace

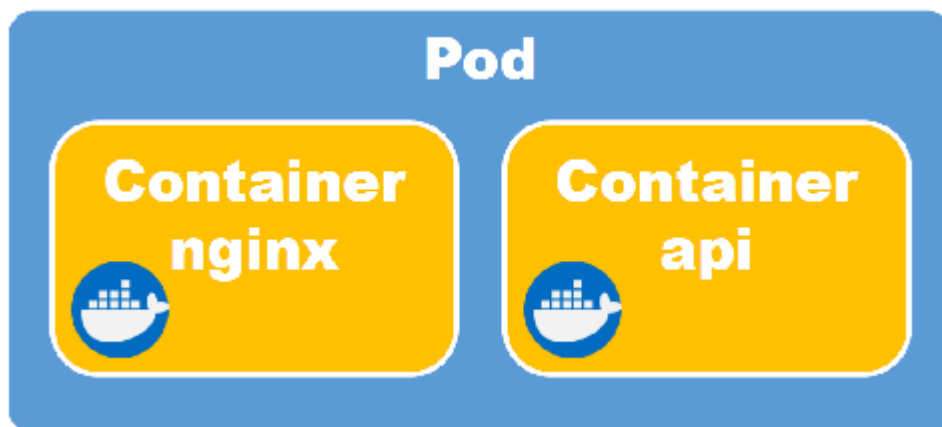
NamespaceとはKubernetesクラスタ内に入れ子となる仮想的なクラスタ

ネームスペース			
名前	ラベル	フェーズ	経過時間
✓ ingress-nginx	-	Active	.59 minutes
✓ kubernetes-dashboard	-	Active	a day
✓ docker	-	Active	a day
✓ default	-	Active	a day
✓ kube-node-lease	-	Active	a day
✓ kube-public	-	Active	a day
✓ kube-system	-	Active	a day

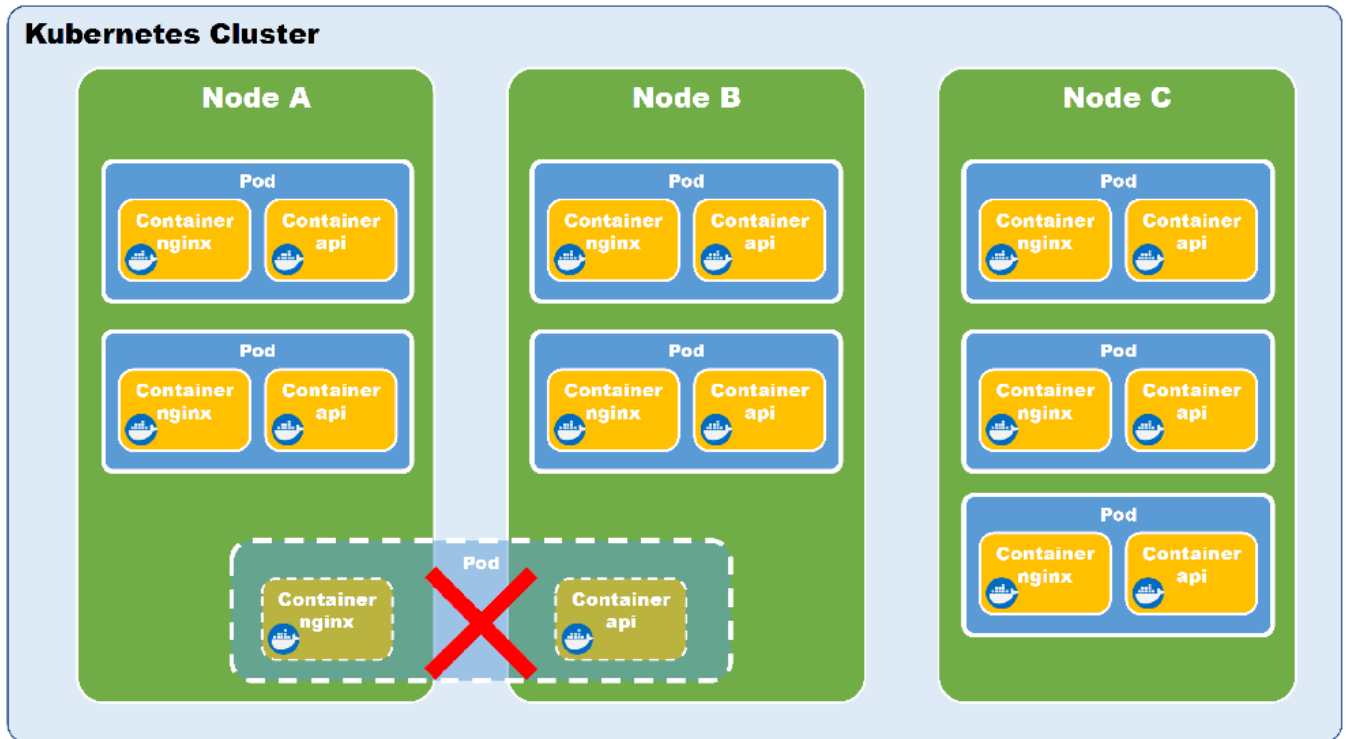
Pod

コンテナの集合体の単位で少なくともひとつのコンテナを持つ Dockerコンテナ単体、あるいはDockerコンテナの集合体

Pod単体



PodのNodeへの配置



※ひとつのPod内のコンテナが別々のノードに配置されることはない

Podをデプロイ～動作確認～削除まで

マニフェストファイル: `swtest-pod.yaml`

```

apiVersion: v1
kind: Pod                                # リソース種類を表す(Pod,ReplicaSet,Deployment...)
metadata:
  name: swtest
spec:
  containers:                             # Podなので配下にコンテナの仕様を記載する
  - name: nginx                           # 使用するイメージ
    image: swtest/nginx:latest            # 必ずローカルを使用
    imagePullPolicy: Never
    env:
    - name: BACKEND_HOST                   # 環境変数設定
      value: localhost:8080               # バックエンドのlocalhost:8080へフォワードする
    ports:
    - containerPort: 80                   # コンテナポートとしてExposeする番号を指定
  - name: api                             # 使用するイメージ
    image: swtest/api:latest              # 必ずローカルを使用
    ports:
    - containerPort: 8080                 # コンテナポートとしてExposeする番号を指定

```

デプロイ

Dashboard画面からデプロイします

入力して作成 **ファイルから作成** フォームから作成

現在選択されているネームスペースにデプロイする、リソースを指定した YAML または JSON ファイルを選択します。 [もっと詳しく](#)

YAML または JSON ファイルを選択してください

swtest-pod.yaml

アップロード キャンセル

1. 画面右上「+」マークをクリック
2. 「ファイルから作成」を選択
3. 「...」をクリックし `swtest-pod.yaml` を選択
4. 「アップロード」ボタンをクリック

下図のようになれば正常にデプロイできています。

ワークロード

ワークロードの状態

ポッド

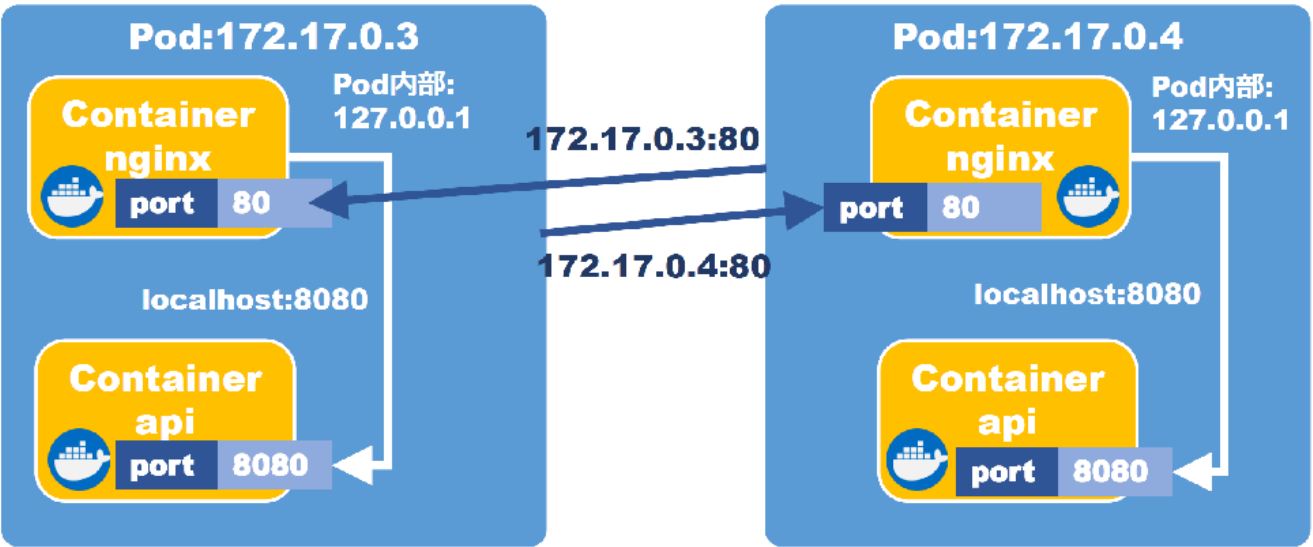
名前	ラベル	ノード	状態	再起動	CPU 使用量 (コア数)	メモリー使用量 (バイト)	経過時間
✓ swtest	-	docker-desktop	Running	0	-	-	31 seconds

1 - 1 of 1 < > >>

デバッグ用Podのデプロイ

`debug-pod.yaml` を同様にデプロイする

Pod仮想IPについて



SWTESTの情報

Workloads > Pods > swtest

クラスター
クラスターロール
ネームスペース
ノード
永続ボリューム
ストレージクラス

ネームスペース
default

概要
ワークロード
Cron ジョブ
デモンセット
デプロイメント
ジョブ
ポッド
レプリカセット
レプリケーションコントローラー
ステートフルセット
ディスカバリーとロードバランシング
インGRESS
サービス
コンフィグとストレージ
コンフィグマップ
永続ボリューム要求
シークレット

検索

PodのIPアドレス

名前	ネームスペース	作成時刻	経過時間	UID
swtest	default	2020/09/21	14 minutes	0c8a3610-88ab-48a9-894e-0449b21d88ab

リソース情報

ノード	状態	IP	QoS クラス	再起動
docker-desktop	Running	10.1.0.111	BestEffort	0

状況 項目:4

イベント 項目:7

コンテナ

nginx

イメージ
swtest/nginx:latest

BACKEND_HOST
localhost:8080

api

イメージ
swtest/api:latest

同一ホストの8080ポートで
アクセス可能

DebugコンテナからSWTESTの動作確認

Workloads > Pods > debug-pod

クラスター

クラスターロール

ネームスペース

ノード

永続ボリューム

ストレージクラス

ネームスペース

default

概要

ワークロード

Cron ジョブ

デーモンセット

デプロイメント

ジョブ

ポッド

レプリカセット

レプリケーションコントローラー

ステートフルセット

ディスクバリーとロードバランシング

インGRESS

debugコンテナに入る

メタデータ

名前

debug-pod

ネームスペース

default

作成時刻

2020/09/21

経過時間

17 minutes

UID

4c305e68-fc31-4e9b-bf24-2d467bc819fe

注釈

kubectl.kubernetes.io/last-applied-configuration

リソース情報

ノード

docker-desktop

状態

Running

IP

10.1.0.119

QoS クラス

BestEffort

再起動

0

状況

項目: 4

イベント

項目: 13

コンテナ

debug

イメージ

swtest/debug:latest

ポッドで実行する

curlコマンドで確認

Workloads > Pods > debug-pod > Shell

クラスター

クラスターロール

ネームスペース

ノード

永続ボリューム

ストレージクラス

debug-pod の debug のシェル

```
bash-5.0#  
bash-5.0#  
bash-5.0# curl http://10.1.0.111/  
The API service is running... bash-5.0#  
bash-5.0#
```

Podの削除

Dashboardを使ってswtestのPodを削除します。

メニューの「ワークロード」>「ポッド」を選択します。

ポッド

名前

ラベル

ノード

状態

再起動

CPU 使用量 (コア数)

メモリー使用量 (バイト)

経過時間

debug-pod

-

docker-desktop

Running

0

-

-

35 minutes

swtest

-

docker-desktop

Running

0

-

-

クリックしポップアップメニューの削除を選択

リソースの削除

namespace `default` の pod `swtest` を本当に削除しますか？

 この操作は次と同等です: `kubectl delete -n default pod swtest`

削除

キャンセル

ReplicaSet

同一仕様のPodの複製数を管理・制御する

PodのマニフェストファイルからはひとつのPodしか生成出来ない。

これを複数同時に生成可能なのが「ReplicaSet」。

マニフェストファイルの確認

マニフェストファイル: `swtest-replicaset.yaml`

```
apiVersion: apps/v1
kind: ReplicaSet          # 属性が「ReplicaSet」となる
metadata:
  name: swtest
  labels:
    app: swtest
spec:
  replicas: 3              # 作成するPodの数を指定
  selector:
    matchLabels:
      app: swtest
  template:                # template以下はPodのそれと同様
    metadata:
      labels:
        app: swtest
    spec:
      containers:
        - name: nginx
          image: swtest/nginx:latest
          imagePullPolicy: Never
          env:
            - name: BACKEND_HOST
              value: localhost:8080
          ports:
            - containerPort: 80
        - name: api
          image: swtest/api:latest
```

```
imagePullPolicy: Never
ports:
  - containerPort: 8080
```

ReplicaSet生成

Dashboardからマニフェストファイルを選択し、ReplicaSetを生成します。

入力して作成 ファイルから作成 フォームから作成

現在選択されているネームスペースにデプロイする、リソースを指定した YAML または JSON ファイルを選択します。 [もっと詳しく](#)

YAML または JSON ファイルを選択してください

swtest-replicaset.yaml

アップロード キャンセル

メタデータ				
名前	ネームスペース	作成時刻	経過時間	UID
swtest	default	2020/09/21	27 seconds	2955a37d-39e0-408a-9722-8ac85e849aa1
ラベル				
app: swtest				

リソース情報	
セレクトター	イメージ
app: swtest	swtest/nginx:latest swtest/api:latest

Pods status

稼働中
3

要求中
3

Podが3セット生成された

ポッド								
名前	ラベル	ノード	状態	再起動	CPU 使用量 (コア数)	メモリー使用量 (バイト)	経過時間	↑
✓ swtest-6j9lz	app: swtest	docker-desktop	Running	0	-	-	a minute	⋮
✓ swtest-8kzwk	app: swtest	docker-desktop	Running	0	-	-	a minute	⋮
✓ swtest-t7jtb	app: swtest	docker-desktop	Running	0	-	-	a minute	⋮

1 - 3 of 3 |< < > >|

レプリカセット					
名前	ラベル	ポッド	経過時間 ↑	イメージ	
✓ swtest	app: swtest	3 / 3	2 minutes	swtest/nginx:latest	swtest/api:latest

クリック後ポップアップからスケールを選択

リソースをスケールする

replicaset swtest は、目標のレプリカ数になるように更新されます。

目標のレプリカ数

5

現在のレプリカ数

3

この操作は次と同等です: `kubectl scale -n default replicaset swtest --replicas=5`

スケール

キャンセル

レプリカ数を5に変更して「スケール」ボタンを押す

Pods status								
稼働中	要求中							
5	5	Podが2つ追加されて合計5セットが動作している						

ポッド								
名前	ラベル	ノード	状態	再起動	CPU 使用量 (コア数)	メモリー使用量 (バイト)	経過時間	
✓ swtest-7dwpb	app: swtest	docker-desktop	Running	0	-	-	a minute	⋮
✓ swtest-n2dmh	app: swtest	docker-desktop	Running	0	-	-	a minute	⋮
✓ swtest-6j9lz	app: swtest	docker-desktop	Running	0	-	-	3 minutes	⋮
✓ swtest-8kzwk	app: swtest	docker-desktop	Running	0	-	-	3 minutes	⋮
✓ swtest-t7jtb	app: swtest	docker-desktop	Running	0	-	-	3 minutes	⋮

1 - 5 of 5 |< < > >|

ReplicaSet削除

Dashboardのレプリカセットのメニューから、先ほど生成したレプリカセットを削除する。

リソースの削除

ネームスペース `default` の replicaset `swtest` を本当に削除しますか？

i この操作は次と同等です: `kubectl delete -n default replicaset swtest`

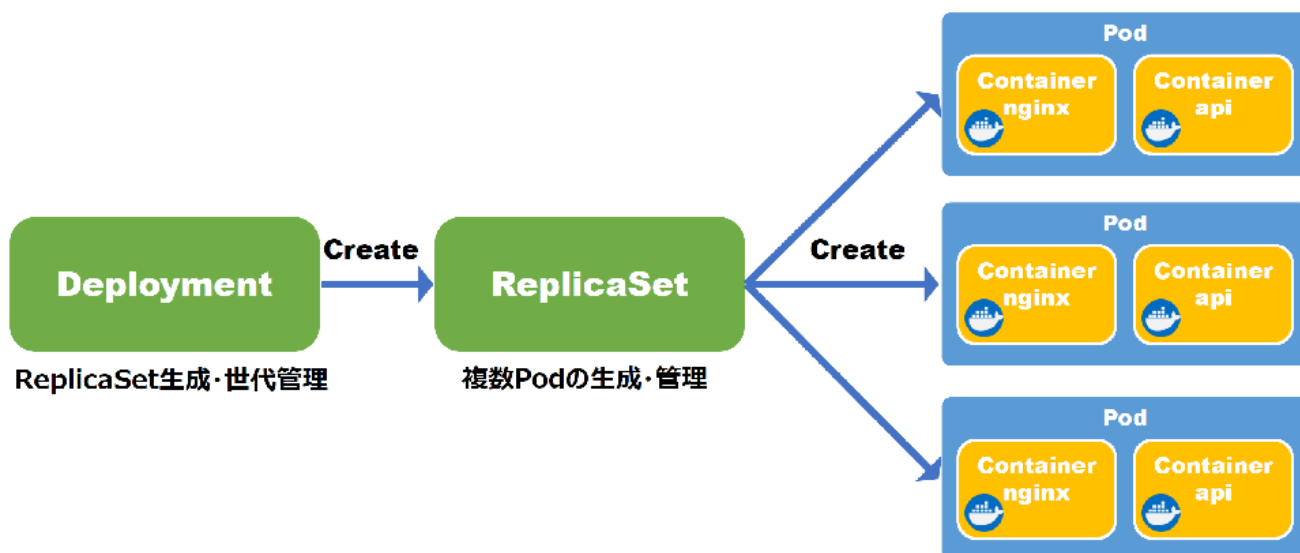
削除

キャンセル

Deployment

- **Deployment** とは、ReplicaSetの上位リソース。
- Deploymentはアプリケーションデプロイの基本単位となるリソース。
- ReplicaSetを管理・操作することができる。
- Deploymentは世代管理機能を持っておりデプロイのロールバックが可能。
- 実運用ではReplicaSetを直接用いる事は少なく、ほとんどがDeploymentのマニフェストファイルで運用している。

Deployment/ReplicaSet/Pod関連図



世代管理付のデプロイ

マニフェストファイル: `swtest-deployment.yaml`

```

apiVersion: apps/v1
kind: Deployment      # ReplicaSetと異なるのはkindだけ
metadata:
```

```
name: swtest
labels:
  app: swtest
spec:
  replicas: 3
  selector:
    matchLabels:
      app: swtest
  template:
    metadata:
      labels:
        app: swtest
    spec:
      containers:
        - name: nginx
          image: swtest/nginx:latest
          imagePullPolicy: Never
          env:
            - name: BACKEND_HOST
              value: localhost:8080
          ports:
            - containerPort: 80
        - name: api
          image: swtest/api:latest
          imagePullPolicy: Never
          ports:
            - containerPort: 8080
```

コマンドラインからデプロイする

作業フォルダより、下記コマンドを投入し、アプリケーションをデプロイする。

```
> kubectl apply -f swtest-deployment.yaml --record
```

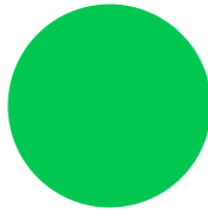
```
deployment.apps/swtest created
```

ワークロード

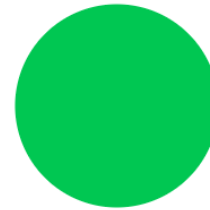
ワークロードの状態



デプロイメント



ポッド



レプリカセット

デプロイメント

名前	ラベル	ポッド	経過時間 ↑	イメージ	
✓ swtest	app: swtest	3 / 3	a minute	swtest/nginx:latest swtest/api:latest	⋮
1 - 1 of 1 < < > >					

ポッド

名前	ラベル	ノード	状態	再起動	CPU 使用量 (コア数)	メモリー使用量 (バイト)	経過時間 ↑	
✓ swtest-76b84cff4-6lviz	app: swtest pod-template-hash: 76b84cff4	docker-desktop	Running	0	-	-	a minute	⋮
✓ swtest-76b84cff4-nt6kn	app: swtest pod-template-hash: 76b84cff4	docker-desktop	Running	0	-	-	a minute	⋮
✓ swtest-76b84cff4-zhxxk	app: swtest pod-template-hash: 76b84cff4	docker-desktop	Running	0	-	-	a minute	⋮
✓ debug-pod	-	docker-desktop	Running	0	-	-	2 hours	⋮
1 - 4 of 4 < < > >								

レプリカセット

名前	ラベル	ポッド	経過時間 ↑	イメージ	
✓ swtest-76b84cff4	app: swtest pod-template-hash: 76b84cff4	3 / 3	a minute	swtest/nginx:latest swtest/api:latest	⋮
1 - 1 of 1 < < > >					

リビジョンの確認

```
> kubectl rollout history deployment swtest

deployment.apps/swtest
REVISION  CHANGE-CAUSE
1          kubectl.exe apply --filename=swtest-deployment.yaml --record=true
```

ローリングアップデートとロールバック

ローリングアップデート

アプリのダウンタイムをゼロで、新しいバージョンに順次切り替えることをローリングアップデートという。

今回swtest/apiのバージョンがv2に変更になった場合を想定したローリングアップデートを行う。

 ローリングアップデート

マニフェストファイル

新バージョンのマニフェストファイル: `swtest-deployment-v2.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: swtest
  labels:
    app: swtest
spec:
  replicas: 3
  selector:
    matchLabels:
      app: swtest
  template:
    metadata:
      labels:
        app: swtest
    spec:
      containers:
        - name: nginx
          image: swtest/nginx:latest
          imagePullPolicy: Never
          env:
            - name: BACKEND_HOST
              value: localhost:8080
          ports:
            - containerPort: 80
        - name: api
          image: swtest/api:v2 # apiアプリバージョンがlatestからv2になった
          imagePullPolicy: Never
          ports:
            - containerPort: 8080
```

アップデートの実施

作業フォルダで下記コマンドを投入する。


```
> kubectl apply -f swtest-deployment-v2.yaml --record  
  
deployment.apps/swtest configured
```

履歴の確認

```
> kubectl rollout history deployment swtest  
  
deployment.apps/swtest  
REVISION  CHANGE-CAUSE  
1          kubectl.exe apply --filename=swtest-deployment.yaml --record=true  
2          kubectl.exe apply --filename=swtest-deployment-v2.yaml --record=true
```

メタデータ

名前

swtest

ネームスペース

default

作成時刻

2020/09/21

経過時間

15 minutes

UID

40d13465-2cca-40f1-870f-e790e830d1a7

ラベル

app: swtest

注釈

deployment.kubernetes.io/revision: 2

revisionが2になっている

[kubectrl.kubernetes.io/last-applied-configuration](#)
[kubernetes.io/change-cause](#)

リソース情報

ストラテジー

RollingUpdate

最小準備秒数

0

改版履歴上限

10

セレクトター

app: swtest

ローリングアップデートストラテジー

最大サージ

25%

最大利用不可

25%

ポッド状態

更新済み

3

合計

3

利用可能

3

全て起動状態

状況

種別	状態	最終探査時刻	最終遷移時刻	理由	メッセージ
Available	True	17 minutes	17 minutes	MinimumReplicasAvaila	Deployment has minimum availability.
Progressing	True	2 minutes	18 minutes	NewReplicaSetAvailabl	ReplicaSet "swtest-64b67676ff" has successfully progressed.

新しいレプリカセット

名前

swtest-64b67676ff

ネームスペース

default

経過時間

a minute

ポッド

3 / 3

ラベル

app: swtest pod-template-hash: 64b67676ff

イメージ

swtest/nginx:latest swtest/api:v2

api:v2が適用されている

古いレプリカセット

表示するものはありません

リソースがありません。

ロールバックの実施

新バージョンに何らかの問題があった場合、旧バージョンへロールバックします。

作業フォルダで下記コマンドを投入する。

```
> kubectl rollout undo deployment swtest

deployment.apps/swtest rolled back
```

切り替え中の状態

ポッド状態

更新済み	合計	利用可能	利用不可
3	4	3	1

順次置き換わる

状況

種別	状態	最終探査時刻	最終遷移時刻	理由	メッセージ
Available	True	27 minutes	27 minutes	MinimumReplicasAvaila	Deployment has minimum availability.
Progressing	True	0 seconds	27 minutes	ReplicaSetUpdated	ReplicaSet "swtest-76b84cff4" is progressing.

新しいレプリカセット

名前

swtest-76b84cff4

ネームスペース

default

経過時間

27 minutes

ポッド

2 / 3

api:latest起動中

ラベル

app: swtest pod-template-hash: 76b84cff4

イメージ

swtest/nginx:latest swtest/api:latest

古いレプリカセット

api:v2削除中

名前	ネームスペース	ラベル	ポッド	経過時間	イメージ
swtest-64b67676ff	default	app: swtest pod-template-hash: 64b67676ff	2 / 1	.12 minutes	swtest/nginx:latest swtest/api:v2

履歴の確認

```
> kubectl rollout history deployment swtest

deployment.apps/swtest
REVISION  CHANGE-CAUSE
2          kubectl.exe apply --filename=swtest-deployment-v2.yaml --record=true
3          kubectl.exe apply --filename=swtest-deployment.yaml --record=true
```

メタデータ

名前

swtest

ネームスペース

default

作成時刻

2020/09/21

経過時間

27 minutes

UID

40d13465-2cca-40f1-870f-e790e830d1a7

ラベル

app: swtest

注釈

deployment.kubernetes.io/revision: 3

revisionは3

[kubectrl.kubernetes.io/last-applied-configuration](#)
[kubernetes.io/change-cause](#)

リソース情報

ストラテジー

RollingUpdate

最小準備秒数

0

改版履歴上限

10

セレクトー

app: swtest

ローリングアップデートストラテジー

最大サージ

25%

最大利用不可

25%

ポッド状態

更新済み

3

合計

3

利用可能

3

全て起動完了

状況

種別	状態	最終探査時刻	最終遷移時刻	理由	メッセージ
Available	True	27 minutes	27 minutes	MinimumReplicasAvail	Deployment has minimum availability.
Progressing	True	8 seconds	28 minutes	NewReplicaSetAvailab	ReplicaSet "swtest-76b84cff4" has successfully progressed.

新しいレプリカセット

名前

swtest-76b84cff4

ネームスペース

default

経過時間

27 minutes

ポッド

3 / 3

ラベル

app: swtest

pod-template-hash: 76b84cff4

イメージ

swtest/nginx:latest

swtest/api:latest

旧バージョンに戻った

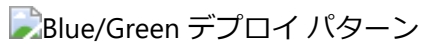
最後に削除する

```
> kubectl delete -f swtest-deployment.yaml

deployment.apps "swtest" deleted
```

Service(blue/green)作成

- **Service** とはPodの集合体(ReplicaSet)に対して経路やサービスディスカバリを提供
- Serviceのターゲットはラベルセレクトタにより決定される



Blue/Greenデプロイメントを実現するため、下記2つのサービス – ReplicaSetを作成します。

1. Blueサービス → Blueデプロイメント (swtest/api:latest)
2. Greenサービス → Greenデプロイメント (swtest/api:v2)

blue/greenデプロイメント

マニフェストファイル: `swtest-deployment-blue.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: swtest-blue
  labels:
    app: swtest
    release: blue          # releaseというラベルでblue/greenを切り替える
spec:
  replicas: 3
  selector:
    matchLabels:
      app: swtest
      release: blue
  template:
    metadata:
      labels:
        app: swtest
        release: blue
    spec:
      containers:
        - name: nginx
          image: swtest/nginx:latest
          imagePullPolicy: Never
          env:
            - name: BACKEND_HOST
              value: localhost:8080
          ports:
            - containerPort: 80
        - name: api
          image: swtest/api:latest
```

```
imagePullPolicy: Never
ports:
  - containerPort: 8080
```

マニフェストファイル: `swtest-deployment-green.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: swtest-green
  labels:
    app: swtest
    release: green
spec:
  replicas: 3
  selector:
    matchLabels:
      app: swtest
      release: green
  template:
    metadata:
      labels:
        app: swtest
        release: green
    spec:
      containers:
        - name: nginx
          image: swtest/nginx:latest
          imagePullPolicy: Never
          env:
            - name: BACKEND_HOST
              value: localhost:8080
          ports:
            - containerPort: 80
        - name: api
          image: swtest/api:v2
          imagePullPolicy: Never
          ports:
            - containerPort: 8080
```

新バージョンのAPI

blue/greenサービス

マニフェストファイル: `swtest-service.yaml`

```
apiVersion: v1
kind: Service
metadata:
  name: swtest-blue
spec:
```

```
selector:
  app: swtest
  release: blue      # release=blueのReplicaSetはswtest-blueサービス
ports:
  - name: http        # httpの80番ポートをswtestに振り当てる
    port: 80

---
apiVersion: v1
kind: Service
metadata:
  name: swtest-green
spec:
  selector:
    app: swtest
    release: green    # release=greenのReplicaSetはswtest-greenサービス
  ports:
    - name: http      # httpの80番ポートをswtestに振り当てる
      port: 80
```

デプロイ

作業フォルダで下記コマンドを投入する。

```
> kubectl apply -f swtest-deployment-blue.yaml --record
> kubectl apply -f swtest-deployment-green.yaml --record
> kubectl apply -f swtest-service.yaml
```

状態確認

デプロイメント

名前	ネームスペース	ラベル	ポッド	経過時間 ↑	イメージ
✓ swtest-green	default	app: swtest release: green	1 / 1	46 minutes	swtest/nginx:latest swtest/api:v2
✓ swtest-blue	default	app: swtest release: blue	1 / 1	46 minutes	swtest/nginx:latest swtest/api:latest

1 - 2 of 2

ポッド

名前	ネームスペース	ラベル	ノード	状態	再起動	CPU 使用量 (コア数)	メモリー使用量 (バイト)	経過時間 ↑
✓ swtest-green-5f5d97d75-47clp	default	app: swtest pod-template-hash: 5f5d97d75 すべて表示	docker-desktop	Running	0	-	-	46 minutes
✓ swtest-blue-84c6dc6c-987hs	default	app: swtest pod-template-hash: 84c6dc6c すべて表示	docker-desktop	Running	0	-	-	46 minutes
✓ debug-pod	default	-	docker-desktop	Running	0	-	-	5 hours

1 - 3 of 3

レプリカセット

名前	ネームスペース	ラベル	ポッド	経過時間 ↑	イメージ
✓ swtest-green-5f5d97d75	default	app: swtest pod-template-hash: 5f5d97d75 すべて表示	1 / 1	46 minutes	swtest/nginx:latest swtest/api:v2
✓ swtest-blue-84c6dc6c	default	app: swtest pod-template-hash: 84c6dc6c すべて表示	1 / 1	46 minutes	swtest/nginx:latest swtest/api:latest

1 - 2 of 2

ディスカバリーとロードバランシング

サービス

名前	ネームスペース	ラベル	クラスター IP	内部エンドポイント	外部エンドポイント	経過時間 ↑
✓ swtest-green	default		10.108.38.241	swtest-green:80 TCP swtest-green:0 TCP	-	46 minutes
✓ swtest-blue	default		10.103.25.7	swtest-blue:80 TCP swtest-blue:0 TCP	-	46 minutes
✓ kubernetes	default	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	a day

debugコンテナからアクセスし動作確認

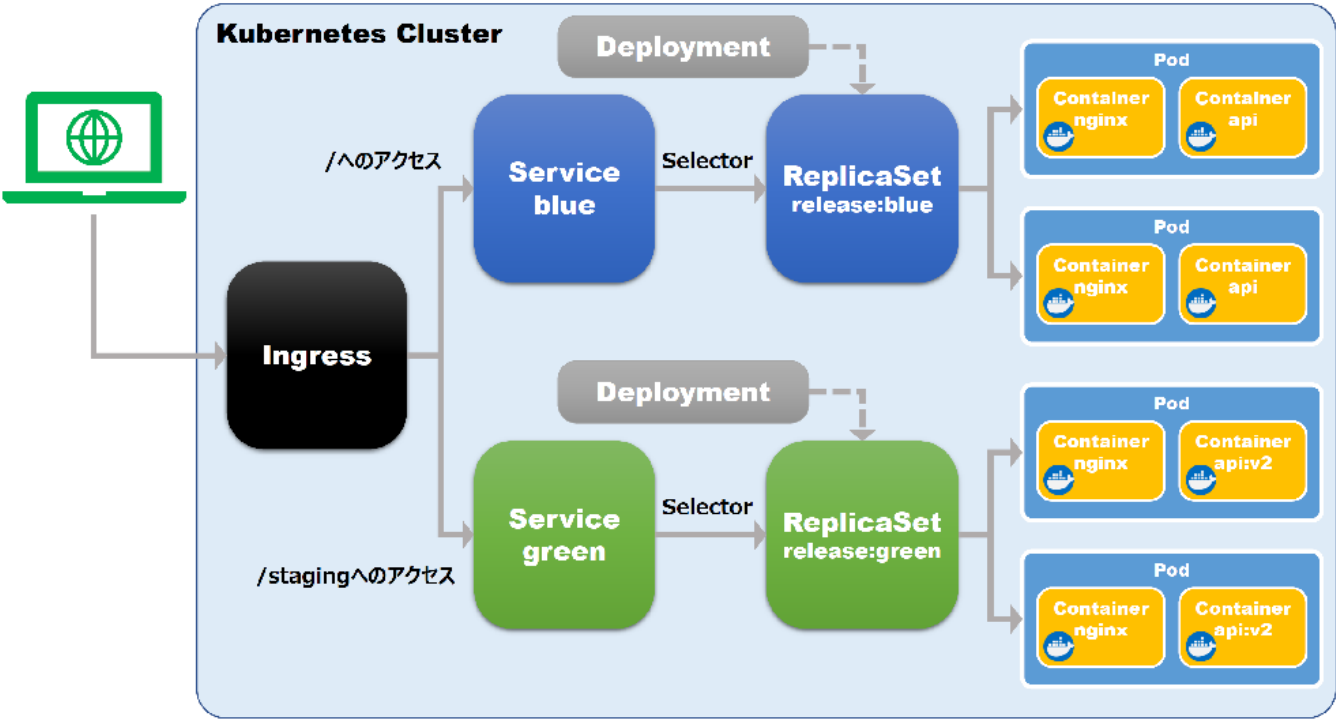
debug-pod の debug

のシェル

```
bash-5.0#  
bash-5.0# curl -I http://swtest-green/  
HTTP/1.1 200  
Server: nginx/1.16.1  
Date: Mon, 21 Sep 2020 14:05:43 GMT  
Content-Type: text/plain; charset=UTF-8  
Content-Length: 40  
Connection: keep-alive  
  
bash-5.0# curl http://swtest-green/  
The API Version 2 service is running... bash-5.0#  
bash-5.0# 新バージョンが応答  
bash-5.0#  
bash-5.0# curl -I http://swtest-blue/  
HTTP/1.1 200  
Server: nginx/1.16.1  
Date: Mon, 21 Sep 2020 14:06:22 GMT  
Content-Type: text/plain; charset=UTF-8  
Content-Length: 30  
Connection: keep-alive  
  
bash-5.0# curl http://swtest-blue/  
The API service is running... bash-5.0#  
bash-5.0#  
bash-5.0#  
bash-5.0# 現行バージョンが応答
```

IngressによるBlue/Greenデプロイ

IngressによるBlue/Greenデプロイイメージ



Ingress-nginxの状況確認

メニューのネームスペースから **ingress-nginx** を選択し、メニューの「概要」をクリックすると下記画面表示となる。

Overview

クラスタークラスターロールネームスペースノード永続ボリュームストレージクラス

ネームスペースingress-nginx

概要

ワークロードCron ジョブデーモンセットデプロイメントジョブポッドレプリカセットレプリケーションコントローラーステートフルセット

ディスカバリーとロードバランシング

イングレスサービス

コンフィグとストレージ

ワークロード

ワークロードの状態

デプロイメント項目: 2

ポッド項目: 2

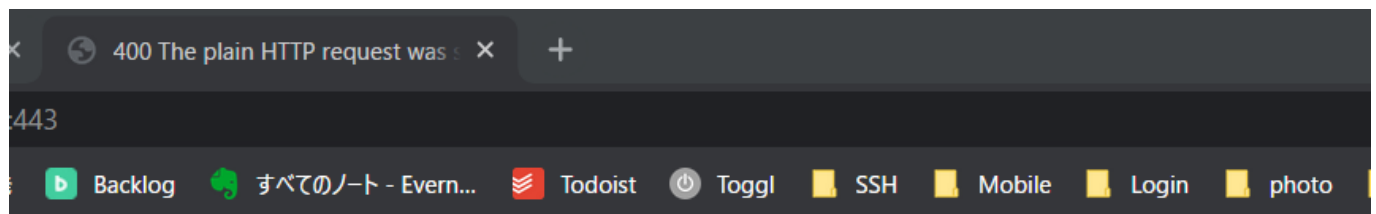
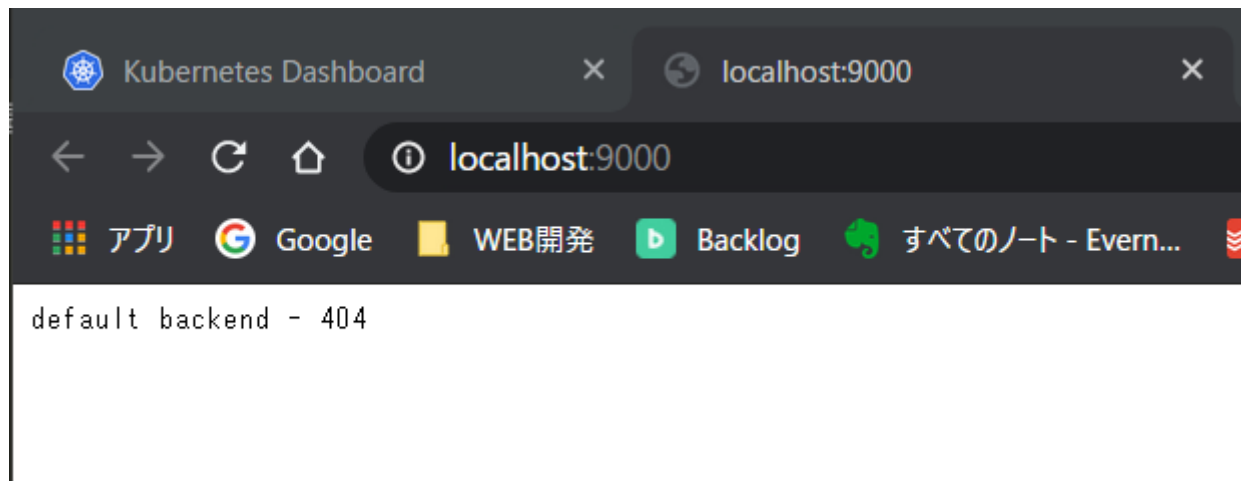
レプリカセット項目: 2

ディスカバリーとロードバランシング

サービス

名前	ラベル	クラスタ IP	内部エンドポイント	外部エンドポイント	経過時間	
ingress-nginx	app: ingress-nginx	10.105.219.79	ingress-nginx.ingress-nginx:9000 TCP ingress-nginx.ingress-nginx:31923 TCP ingress-nginx.ingress-nginx:443 TCP ingress-nginx.ingress-nginx:31166 TCP	localhost:9000 localhost:443	9 hours	
default-http-backend	app: default-http-backend	10.105.175.142	default-http-backend.ingress-nginx:80 TCP default-http-backend.ingress-nginx:0 TCP	-	9 hours	

画面中の「外部エンドポイント」のURLリンクをクリックすると、下記画面が新たに開けば正常に動作している。



400 Bad Request

The plain HTTP request was sent to HTTPS port

nginx/1.13.12

Ingress追加

マニフェストファイル: `swtest-ingress.yaml`

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: swtest
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
    - host: swtest.api.local
      http:
        paths:
          - path: /
            backend:
              serviceName: swtest-blue
```

現行バージョンは/でアクセス

```
    servicePort: 80
  - path: /staging                    # 新バージョンは/stagingでアクセス
    backend:
      serviceName: swtest-green
      servicePort: 80
```

作業フォルダで下記コマンドを投入する

```
> kubectl apply -f swtest-ingress.yaml

ingress.networking.k8s.io/swtest configured
```

下図のようにイングレスが追加される

ディスカバリーとロードバランシング				
イングレス				
名前	ラベル	エンドポイント	経過時間 ↑	
swtest	-	localhost 🔗	56 seconds	⋮
1 - 1 of 1 < < > >				

動作確認

現行バージョンへのアクセス

Postmanで動作確認を行います

- URL: `http://localhost:9000/`
- Headers: Host -> `swtest.api.local`
- 「Send」 ボタンクリック

現行バージョンの応答が返る

GET http://localhost:9000/ Send Save

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies Code

Headers 7 hidden

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Host	swtest.api.local				
	Key	Value	Description			

Body Cookies Headers (5) Test Results 200 OK 57 ms 193 B Save Response

Pretty Raw Preview Visualize Text

1 The API service is running...

次期バージョンへのアクセス

- URL: http://localhost:9000/staging/
- Headers: Host -> swtest.api.local
- 「Send」 ボタンクリック

次期バージョンの応答が返る

GET http://localhost:9000/staging/ Send Save

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies Code

Headers 7 hidden

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Host	swtest.api.local				
	Key	Value	Description			

Body Cookies Headers (5) Test Results 200 OK 14 ms 203 B Save Response

Pretty Raw Preview Visualize Text

1 The API Version 2 service is running...

Blue -> Green への切り替え

path切り替え

マニフェストファイルの「path」を入れ替えることによりGreenをリリースする

イングレス

名前	ネームスペース	ラベル	エンドポイント	経過時間 ↑
swtest	default	-	localhost 🔗	an hour

ポップアップメニューから編集を選択する

オンラインからpathを書き換えると即時にBlueからGreenにアプリケーションが切り替わる

リソースの編集

YAML

JSON

```

8  resourceVersion: '168890'
9  generation: 13
10 creationTimestamp: '2020-09-21T15:23:15Z'
11 annotations:
12   kubernetes.io/last-applied-configuration: >
13   [{"apiVersion": "networking.k8s.io/v1beta1", "kind": "Ingress", "metadata": {"annotations": {"nginx.ingress.kubernetes.io/rewrite-target": "/"}, "name": "swtest", "namespace": "default"}, "spec": {"rules": [{"host": "swtest.api.local", "http": {"paths": [{"backend": {"serviceName": "swtest-blue", "servicePort": 80}, "path": "/"}, {"backend": {"serviceName": "swtest-green", "servicePort": 80}, "path": "/staging"}]}}]}]}]
14   nginx.ingress.kubernetes.io/rewrite-target: /
15 spec:
16   rules:
17   - host: swtest.api.local
18     http:
19       paths:
20       - path: /
21         backend:
22           serviceName: swtest-blue
23           servicePort: 80
24       - path: /staging
25         backend:
26           serviceName: swtest-green
27           servicePort: 80
28 status:
29   loadBalancer:
30     ingress:
31     - hostname: localhost
32

```

/ -> /stagingに書き換える

/staging -> / に書き換える

この操作は次と同等です: `kubectl apply -f <spec.yaml>`

更新 キャンセル 更新ボタンをクリックすると即時にBlueとGreenが入れ替わる

新バージョンへのアクセス

Postmanで動作確認を行います

- URL: `http://localhost:9000/`
- Headers: Host -> `swtest.api.local`
- 「Send」 ボタンクリック

新バージョンの応答が返る

GET

http://localhost:9000/

Send

Save

ParamsAuthHeaders (8)BodyPre-req.TestsSettingsCookiesCode

Headers7 hidden

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Host	swtest.api.local				
	Key	Value	Description			

BodyCookiesHeaders (5)Test Results200 OK14 ms203 BSave Response

PrettyRawPreviewVisualizeText

1The API Version 2 service is running...

旧バージョンへのアクセス

- URL: `http://localhost:9000/staging/`
- Headers: Host -> `swtest.api.local`
- 「Send」 ボタンクリック

旧バージョンの応答が返る

GET

http://localhost:9000/staging/

Send

Save

ParamsAuthHeaders (8)BodyPre-req.TestsSettingsCookiesCode

Headers7 hidden

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Host	swtest.api.local				
	Key	Value	Description			

BodyCookiesHeaders (5)Test Results200 OK28 ms193 BSave Response

PrettyRawPreviewVisualizeText

1The API service is running...

最後に実習環境の削除方法(できればKubernetes環境は破棄願います)

次回予定の、**Apache Kafka** はDocker環境で実施します。Kubertenes環境が残っていると、動作が重くなります。

出来れば次回実習までにKubertenes環境を破棄願います。

PROXYのプロセスをCtrl+Cで終了させた後、構築の逆順でリソースを削除します。

Kubertenesリソース削除

1. Ingress: `kubectl delete -f swtest-ingress.yaml`
2. Service: `kubectl delete -f swtest-service.yaml`
3. Deployment: `kubectl delete -f swtest-deployment-green.yaml`
4. Deployment: `kubectl delete -f swtest-deployment-blue.yaml`

ingress-nginx, dashboard削除

1. ingress-nginx: `kubectl delete -f cloud-generic.yaml`
2. ingress-nginx: `kubectl delete -f mandatory.yaml`
3. admin-user: `kubectl delete -f role_binding.yaml`
4. admin-user: `kubectl delete -f create_account.yaml`
5. dashboard: `kubectl delete -f recommended.yaml`

Docker Dashboardで無効化

1. Docker DashboardのSettingでKubertenesの有効化ボタンを元に戻す
2. 「Apply&Restart」 ボタンを押す

不要データの削除

1. Docker DashboardのTroubleshootボタンを押す
2. 「Clean / Purge data」 ボタンを押す
3. 削除対象のチェックを全て付ける
4. 「Delete」 ボタンを押す

