

kubernetes勉強会

(基本編)

kubernetes(k8s)とは？

- Kubernetes は**コンテナ化されたアプリケーション**のデプロイ、スケーリングなどの管理を自動化するためのプラットフォーム（コンテナオーケストレーションエンジン）です。
-

コンテナとは？

- OS上に「独立したサーバーと同様の振る舞いをする区画」のこと。
カーネルなどの機能を使って、OSのプロセスとして起動する。

@boxbg-orange text-white rounded demo-box-pad

Dockerコンテナ

 Dockerコンテナ <http://image.itmedia.co.jp/ait/articles/1701/30/wi-docker01002.png>

コンテナ ≡ 仮想ホスト (VM)

※ コンテナは仮想ホスト (VM) のプロセス単位に近

+++

参考

DockreFileの書き方を学ぶ

- 改めてDockerfileのベストプラクティスを振り返ろう
<https://www.slideshare.net/ssuser1f3c12/introduce-that-best-practices-for-writing-dockerfiles>
-

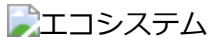
k8sで何ができるか？

- 複数のKubernetes Node の管理 |
- コンテナのスケジューリング |
- ローリングアップデート |
- スケーリング/オートスケーリング |
- コンテナの死活監視 |
- 障害時のセルフヒーリング |
- サービスディスカバリ |
- ロードバランシング |
- データの管理 |

- ワークロードの管理 |
- ログの管理 |
- Infrastructure as Code |
- その他エコシステムとの連携や拡張 (下ページ) |

+++

エコシステム



アーキテクチャ



kubernetes リソース

- **Workloads** リソースコンテナの実行に関するリソース
- **Discovery & LB** リソースコンテナを外部公開するようなエンドポイントを提供するリソース
- **Config & Storage** リソース設定/機密情報/永続化ボリュームなどに関するリソース
- **Cluster** リソースセキュリティやクォータなどに関するリソース
- **Metadata** リソースクラスタ内の他のリソースを操作するためのリソース

Workloads リソース

- Pod
- ~~ReplicationController~~ (廃止)
- ReplicaSet
- Deployment
- DaemonSet
- StatefulSet
- Job
- CronJob

Workloadsの階層構造

- Pod ---> ReplicaSet ---> Deployment
- Pod ---> DemonSet
- Pod ---> StatefulSet
- Pod ---> Job ---> CronJob

@box[bg-orange text-white rounded demo-box-pad](Pod が 1 サーバに相当。コンテナが 1 プロセスに相当。)

Pods-overview



- <https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/#pods-overview>

Node-overview



<https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/#node-overview>

Discovery & LB リソース

- Service
 - ClusterIP ★
 - ExternalIP (ClusterIP の一種)
 - NodePort ★
 - LoadBalancer
 - Headless (None)
 - ExternalName
 - None-Selector
- Ingress ★

Service の役割

- L4 LoadBalancing
 - クラスタ内DNSによる名前解決
 - ラベルを利用したPodのサービスディスカバリ
-

Ingress の役割

- L7 LoadBalancing
 - HTTPS終端
 - パスベースルーティング
-

ClusterIP



kind: Service の type: ClusterIP

```
apiVersion: v1
kind: Service
metadata:
  name: sample-clusterip
spec:
  type: ClusterIP
  ports:
    - name: "http-port"
      protocol: "TCP"
      port: 8080
      targetPort: 80
  selector:
    app: sample-app

# ClusterIP Serviceを作成
$ kubectl apply -f clusterip_sample.yml
```

@2 @6

NodePort



kind: Service の type: NodePort

```
apiVersion: v1
kind: Service
metadata:
  name: sample-nodeport
spec:
  type: NodePort
  ports:
    - name: "http-port"
      protocol: "TCP"
      port: 8080
      targetPort: 80
      nodePort: 30080
  selector:
    app: sample-app

# NodePort Serviceの作成
kubectl apply -f nodeport_sample.yml
```

@2 @6

Ingress



Config & Storage リソース

- Config
 - Secret 機密情報などを管理する
 - ConfigMap 単純なKey-Value値や設定ファイルなどは、ConfigMapで管理する
 - Storage
 - PersistentVolumeClaim PersistentVolumeリソースの中から「xxxGBの領域ちょうだい！」と要求するためのリソース。
-
- volume k8sノードのstorage相当。（抽象化されておらず、直接ノードのディレクトリを指定する）
 - EmptyDir
 - HostPath
 - nfs などのvolumeプラグインがある。

+++

volume は POD定義で直接指定する。

```
apiVersion: v1
kind: Pod
metadata:
  name: sample-hostpath
```

```
spec:
  containers:
  - image: nginx:1.12
    name: nginx-container
    volumeMounts:
    - mountPath: /srv
      name: hostpath-sample
  volumes:
  - name: hostpath-sample
    hostPath:
      path: /data
      type: DirectoryOrCreate

$ kubectl apply -f hostpath-sample.yml
```

@2 @12-16 @9-11

- persistentVolume
k8sで抽象化された永続化Volumeです。

+++

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: sample-pv
  labels:
    type: nfs
    environment: stg
spec:
  capacity:
    storage: 10G
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: slow
  mountOptions:
  - hard
  nfs:
    server: xxx.xxx.xxx.xxx
    path: /nfs/sample

$ kubectl create -f pv_sample.yml
```

@2

- persistentVolumeClaim
PersistentVolumeリソースの中から「xxxGBの領域ちょうだい！」と要求するためのリソース。



Cluster リソース

未稿

Metadata リソース |

未稿

小ネタ

- kubectlコマンドのパラメタ補完（必須！） <https://kubernetes.io/docs/tasks/tools/install-kubectl/#enabling-shell-autocompletion>
-

終わり