



Kubernetes
技術者認定

Kubernetes技術者認定 技術解説セミナー (@OSC京都オンライン) 2023/07/29 開催

45分で理解するKubernetesの世界

鯨井貴博

ZEUS
enterprise

株式会社ゼウス・エンタープライズ
LinuCエヴァンジェリスト

LPI-JAPAN



■ 鯨井貴博(くじらいたかひろ)

LPI-Japanプラチナスポンサー 株式会社ゼウス・エンタープライズ

大学時代 Unixの存在を知り、日経Linuxを読み始める。
2000年にVine Linux 2.0で一度挫折を経験。
その悔しさを忘れきれず、2007年 他業種からIT業界に転職しLinuxに再チャレンジ。

SE・商用製品サポート・インストラクター・プロジェクト管理などを経験し、現在に至る。
自分自身が学習で苦労した経験から、初心者を含む受講者に分かりやすい講義を行うように心がけている。

また、興味の向くIT技術・オープンソースソフトウェアなどについて、
Opensourcetehtブログ (<https://www.opensourceteht.tokyo/>) で執筆中。
実際に自分でやってみる/使ってみる・開発者本人から話を聞いてみることを大切にしています。

2018年頃にDockerの延長で触り始めたのが、kubernetesを使うきっかけ。



Opensourcetehtブログ

すべての人に公開

<https://www.opensourceteht.tokyo/>

メインブログ

記事を書く

投稿数

709

読者数

54





IT業界におけるキャリアアップ例

■ 私の場合



- https://www.cisco.com/c/ja_ip/training-events/training-certifications/certifications/associate/ccna.html
- <https://linuc.org/>
- <https://www.ipa.go.jp/shiken/index.html>
- <https://www.jdla.org/certificate/general/>
- <https://www.accel-exam.jp/>
- <https://html5exam.jp/>
- <https://training.linuxfoundation.org/ja/certification/certified-kubernetes-administrator-cka/>



■ Kubernetes技術者認定とは

Kubernetesの管理・運用やKubernetes用のアプリケーションを開発・構築することができるスキルの証明です。LPI-JapanはAuthorized Certification Partnerとして、「CKA」「CKAD」「CKS」に加えて、クラウドネイティブの概念や基礎が学べる「KCNA」の4つの試験と、それぞれのトレーニング教材を提供しています。



CKA

基本的なインストールのほかにKubernetesクラスタを構成および管理するスキルを備えていること証明します。



CKAD

Kubernetes用のクラウドネイティブアプリケーションを設計、構築、公開できるスキルを備えていること証明します。



KCNA

Kubernetesとクラウドネイティブエコシステムの基本知識があることを証明します。



CKS

コンテナベースのアプリケーションやKubernetesプラットフォームの環境などのセキュリティを確保できるスキルの証明。





45分で理解するkubernetesの世界

- コンテナの特徴
- Kubernetesとは
- Kubernetesのリソース
- Appendix



コンテナの特徴



コンテナを起動してみる！

kubernetesを使った アプリケーションの起動と公開

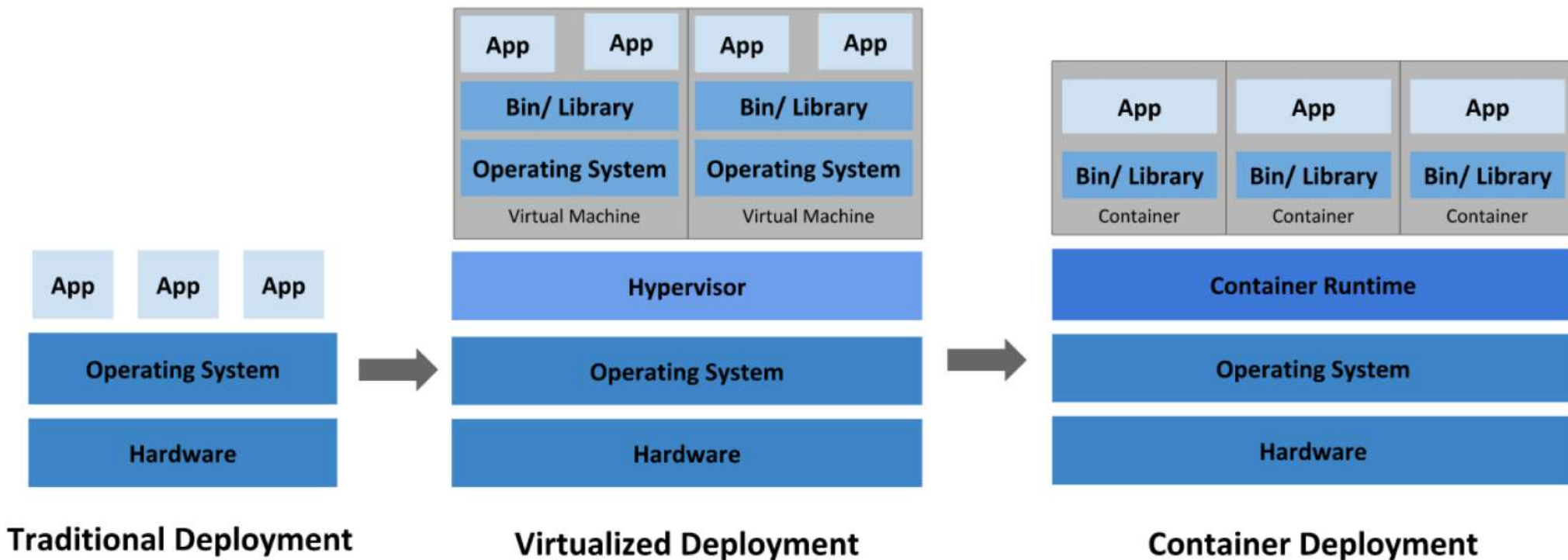
<https://youtu.be/MAbY02dwbgw>



コンテナの特徴(物理マシン・仮想マシンとの比較)

- 軽量である
- 起動が高速(早ければ数秒)
- 移植性に優れる

仕組みは違えど、
いずれもOS/プログラム/NW/セキュリティ(アクセス制限)などは
共通した機能を持っていることを認識することが重要！



<https://kubernetes.io/ja/docs/concepts/overview/>



コンテナの特徴

- 軽量である ⇒ホストOSのLinuxカーネルを共有するため
- 起動が高速(早ければ数秒) ⇒軽量であるため
- 移植性に優れる ⇒コンテナ基盤さえあれば起動可能

The screenshot shows the Docker Hub page for the nginx official image. The 'Tags' section is active, displaying a table of image tags and their compressed sizes. A red circle highlights the 'COMPRESSED SIZE' column, and a blue callout bubble points to it with the text 'コンテナの元となるデータ自体が小さいサイズである'.

TAG	DIGEST	OS/ARCH	VULNERABILITIES	COMPRESSED SIZE
latest	7c12ed2bf80d	linux/386	0 H 0 M 35 L	65.47 MB
	1bb5c4b86cb7	linux/amd64	0 H 0 M 35 L	67.32 MB
	0a7fa5f030de	linux/arm/v5	0 H 0 M 35 L	60.26 MB
+5 more...				

https://hub.docker.com/_/nginx/tags



コンテナが解決する課題

- 検証環境と商用環境で全く同じデータ(コンテナイメージ)を使うことができる

移植性と名前空間(Namespace)の恩恵

<https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/>

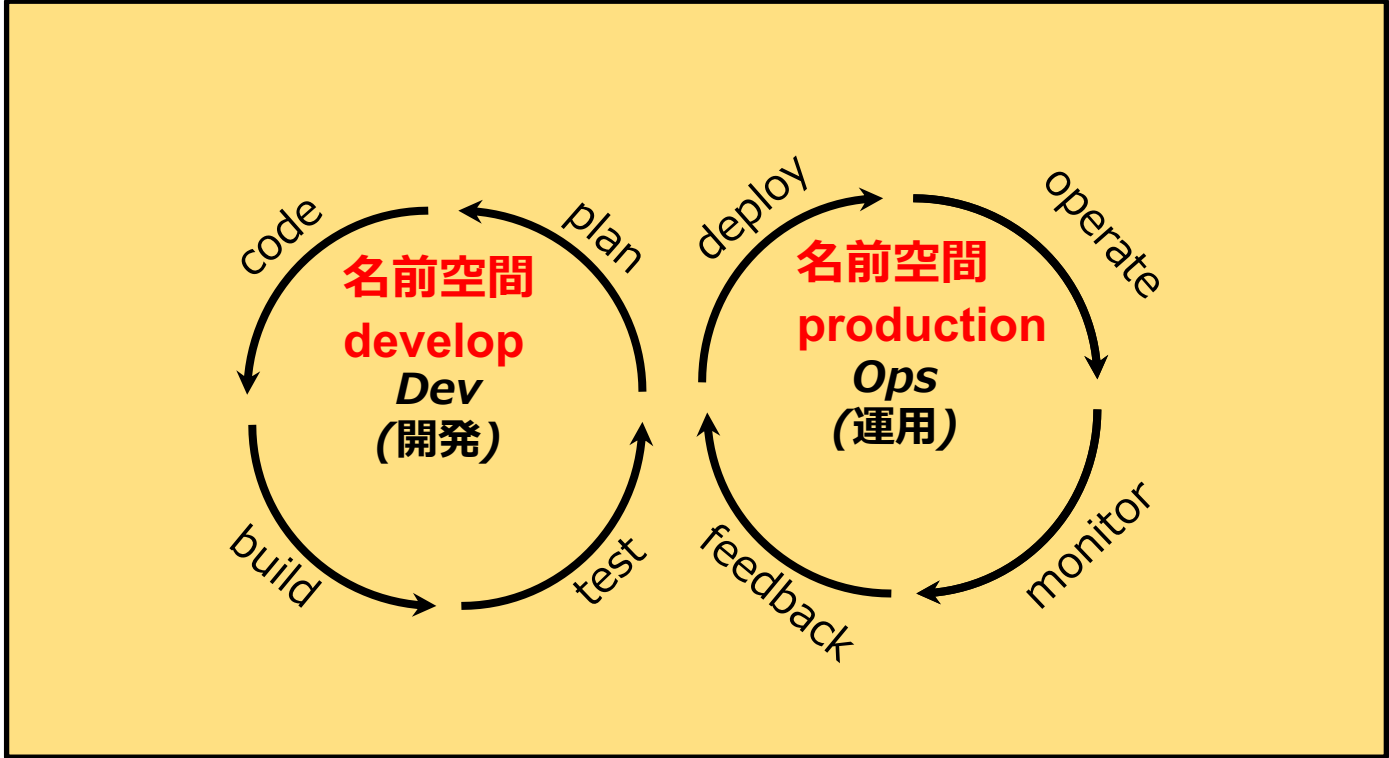
```
kubeuser@master01:~$ kubectl get namespaces
```

NAME	STATUS	AGE	
production	Active	111d	・・・商用環境
develop	Active	113d	・・・検証環境
cadvisor	Active	69d	
default	Active	129d	
ingress-nginx	Active	127d	
kube-node-lease	Active	129d	
kube-public	Active	129d	
kube-system	Active	129d	
kubernetes-dashboard	Active	116d	
metallb-system	Active	128d	



コンテナが解決する課題

- DevOps(開発・運用)との相性がいい





Kubernetesとは



- ベースにあるのは、Google社内で使われていたコンテナクラスタマネージャ「Borg」
- 2014年6月にオープンソースで公開
- 2015年7月にバージョン1.0となり、Linux Foundation傘下の組織であるCNCF(Cloud Native Computing Foundation)に移管

Releases

Release History

The Kubernetes project maintains release branches for the most recent three minor releases (1.27, 1.26, 1.25). Kubernetes 1.19 and newer receive [approximately 1 year of patch support](#). Kubernetes 1.18 and older received approximately 9 months of patch support.

Kubernetes versions are expressed as **x.y.z**, where **x** is the major version, **y** is the minor version, and **z** is the patch version, following [Semantic Versioning](#) terminology.

More information in the [version skew policy](#) document.

Release History

1.27

Latest Release: 1.27.3 (released: 2023-06-14)

End of Life: 2024-06-28

Patch Releases: [1.27.0](#), [1.27.1](#), [1.27.2](#), [1.27.3](#)

Complete 1.27 [Schedule](#) and [Changelog](#)

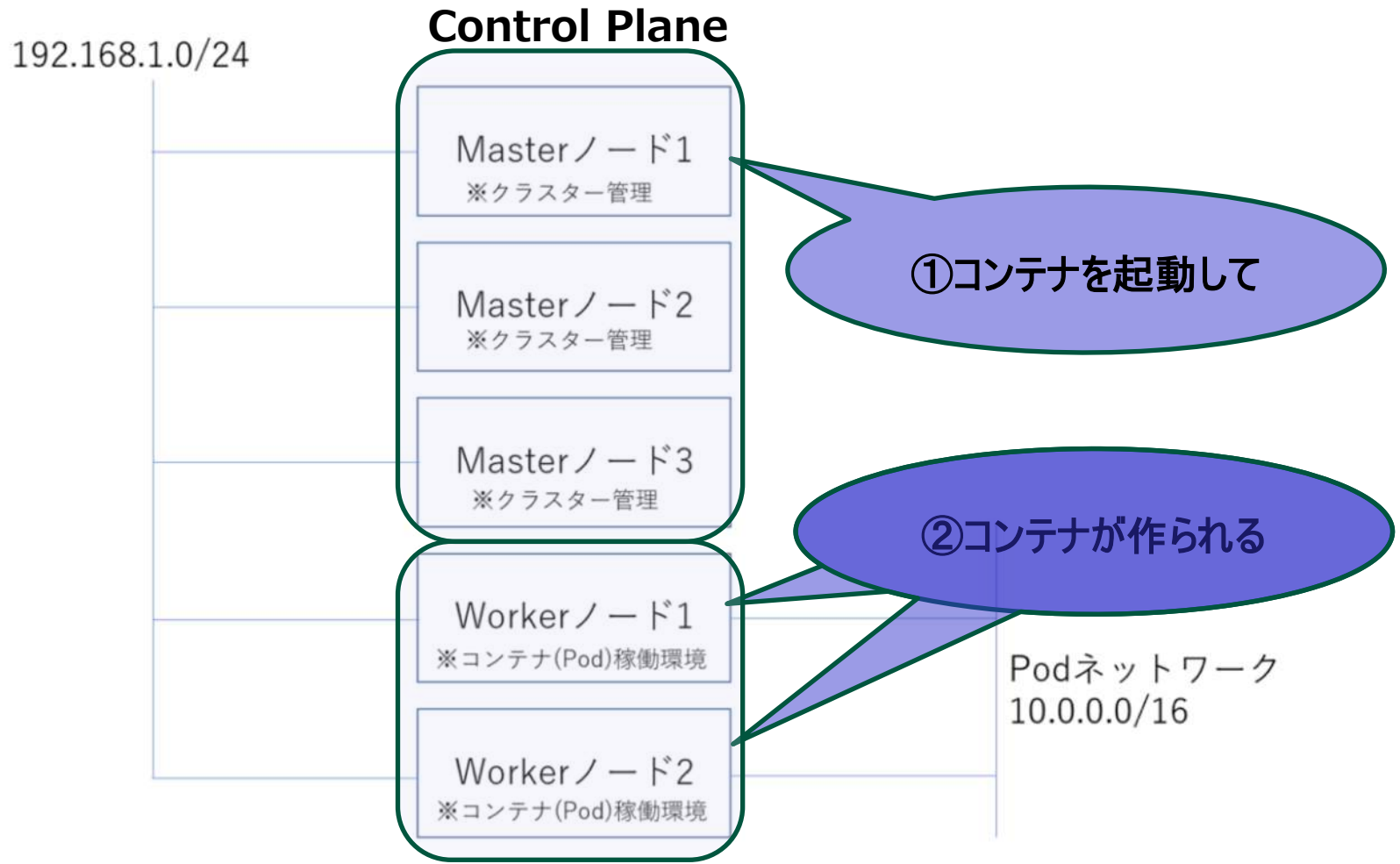
<https://kubernetes.io/releases/>



kubernetesのcluster構成

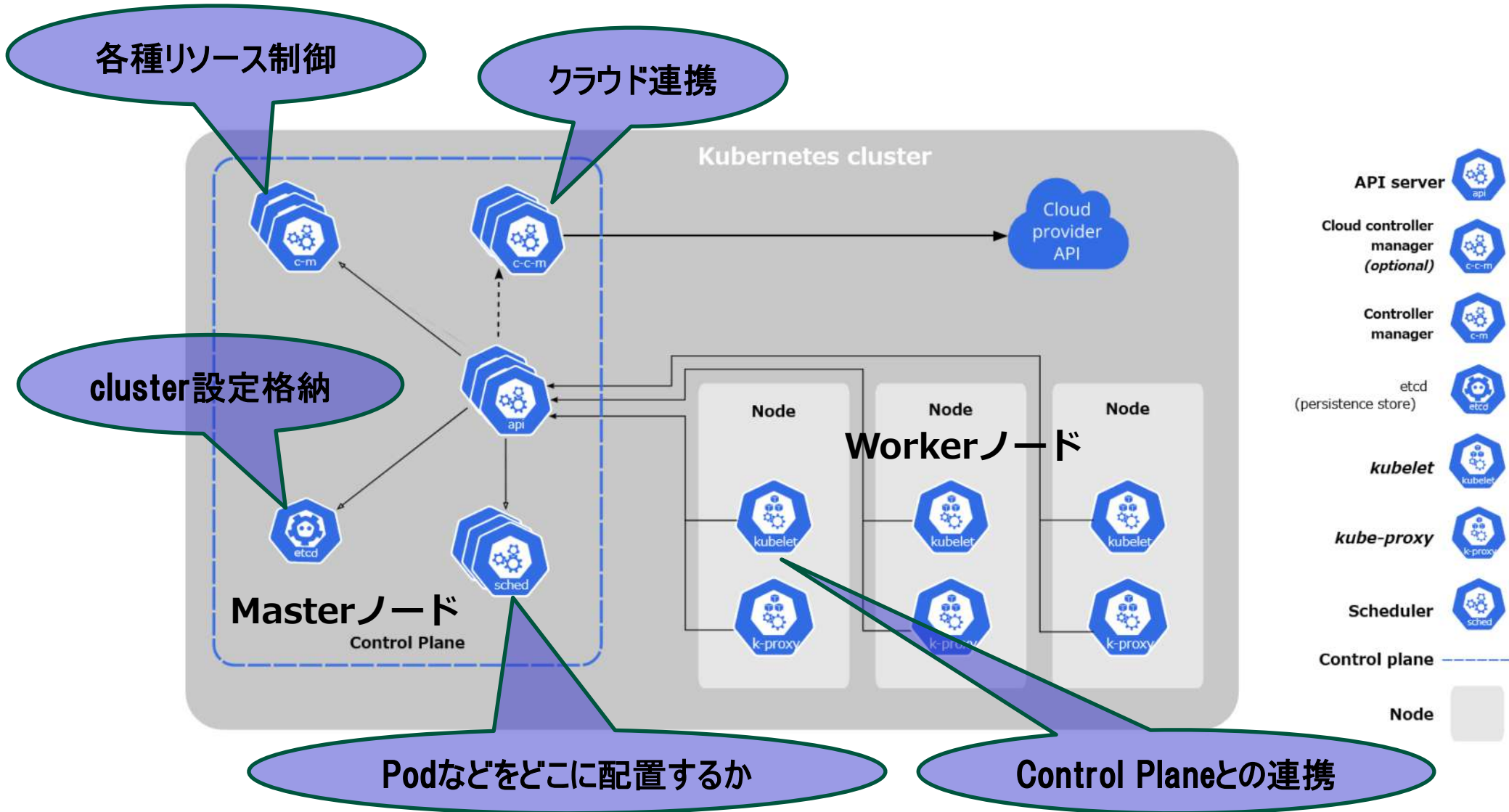
- Masterノード：クラスター全体の管理する役割
- Workerノード：コンテナ(Pod)が稼働する環境

検証用など最小構成としたい場合、
両方の役割を1台のマシンで兼用可能
※当然だが、冗長性なし





kubernetesのcluster構成





kubernetesの種類(distribution)

- Red Hat OpenShift
Red Hat社製Kubernetes

<https://www.redhat.com/ja/technologies/cloud-computing/openshift>

- Tanzu Kubernetes Grid
Vmware社製kubernetes

<https://tanzu.vmware.com/jp/kubernetes-grid>

- Rancher
Suse社製Kubernetes

<https://www.rancher.com/>

Linuxディストリビューションに種類が多いように、
kubernetesも様々な種類がある！



kubernetesの種類(distribution)

- クラウドサービス(AKS、EKS、GCPなど)

Kubernetes Cluster(基盤)の管理が不要

AKS(Azure Kubernetes Service)

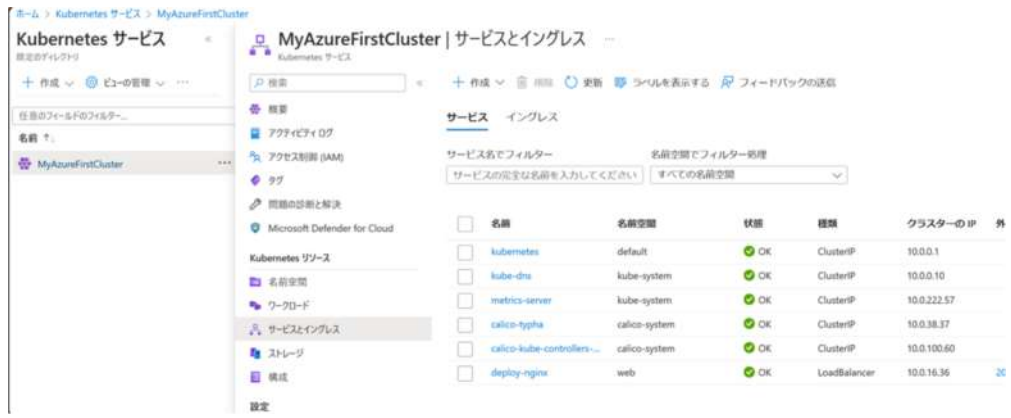
<https://azure.microsoft.com/ja-jp/products/kubernetes-service>

EKS(Amazon Elastic Kubernetes Service)

<https://aws.amazon.com/jp/eks/>

GKE(Google Kubernetes Engine)

<https://cloud.google.com/kubernetes-engine?hl=ja>



初めてのAKS(Azure Kubernetes Service) on Microsoft Azure

<https://www.opensourcetechnology.com/entry/20221223/1671721620>



kubernetesのインストール方法

■ kubeadm

拡張性のあるkubernetes構成を構成

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/>

■ kind

“kubernetes in docker”の略で、Docker上にkubernetesを構成

<https://kind.sigs.k8s.io/docs/user/quick-start/>

■ minikube

PC上にVMとしてkubernetes(シングルノード)を構成

<https://kubernetes.io/ja/docs/setup/learning-environment/minikube/>



kubernetesクラスター(v1.22.0)の構築(kubeadm) on Ubuntu Server 20.04.3 LTS

<https://www.opensourcetech.tokyo/entry/20220215/1644853476>

DualStack(IPv4 & IPv6)のkubernetesクラスター構築(v1.26.00・ubuntu22.04)

<https://www.opensourcetech.tokyo/entry/20230314/1678782139>

KubernetesクラスターのUpgrade(1.26 to 1.27)

<https://www.opensourcetech.tokyo/entry/20230513/1683966484>



Kubernetesのリソース



kubernetesのリソース

- Pod(コンテナ、最小の単位)
- ConfigMap(設定ファイルや変数などを定義)
- Volume(コンテナのデータ保存場所)
- Secret(パスワードなど秘匿性の高いデータを定義)
- ReplicaSet(コンテナを冗長化など)
- Deployment(コンテナの自動復旧・自動増減などの機能を付与)
- Service(コンテナを外部に公開する)
- Ingress(L7ロードバランサー機能)
- NetworkPolicy(ACLのようなもの)
- DaemonSet(コンテナを各ノードに1台ずつ配置する機能)
- Job(一度だけ処理を実行するためのコンテナ)
- CronJob(定期的に処理を実行するためのコンテナ)
- Role/ClusterRole(権限)

など

この中から必要なものを組み合わせて使う！



kubernetesにおけるリソースの定義

■ yamlという方法で表記する

```
kubeuser@master01:~$ cat pods.yaml
```

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
  labels:
```

```
    run: test-webserver
```

```
  name: test-webserver
```

```
spec:
```

```
  containers:
```

```
  - image: nginx:latest
```

```
    name: test-webserver
```

```
  resources:
```

```
    limits:
```

```
      cpu: 100m
```

```
      memory: 5Mi
```

```
    requests:
```

```
      cpu: 50m
```

```
      memory: 3Mi
```

APIバージョンとリソース(kind)の指定

コンテナに付与するラベル

使用するコンテナイメージの指定と名前

Pod(コンテナ)に割り当てるリソース指定

yamlに対応したEditorやツールを使うと便利♪



■ チートシート

<https://kubernetes.io/ja/docs/reference/kubectl/cheatsheet/>

kubectl Cheat Sheet

This page contains a list of commonly used `kubectl` commands and flags.

Note: These instructions are for Kubernetes v1.27. To check the version, use the `kubectl version` command.

Kubectl autocomplete

BASH

```
source <(kubectl completion bash) # set up autocomplete in bash into the current shell, bash-completion p  
echo "source <(kubectl completion bash)" >> ~/.bashrc # add autocomplete permanently to your bash shell.
```



Pod

- コンテナの実態 ≡ アプリケーション

- コンテナの起動

```
kubectl run nginx --image=nginx
```

- コンテナの一覧表示

```
kubectl get pod
```

- コンテナのひな形(yamlファイル)作成

```
kubectl run nginx --image=nginx --dry-run=client -o yaml > pod.yaml
```

- コンテナの起動(yamlファイルから)

```
kubectl apply -f pod.yaml
```

※Podネットワーク用(内部)のIPアドレスが付与されているので、
外部公開設定(Service)を使わないと外部と通信が出来ない

※リソースの制限(limits/requests)を制限しておく



■ Static Pod

指定したノードでPodを起動させる

kubelet.confに設定ファイル(yamlファイル)を配置するパスが書いてある

```
staticPodPath: /etc/kubernetes/manifests
```

```
kubeuser@master01:~$ ls /etc/kubernetes/manifests/  
etcd.yaml kube-apiserver.yaml kube-controller-manager.yaml kube-scheduler.yaml webserver.yaml
```

```
kubeuser@master01:~$ kubectl get pods  
NAME                READY  STATUS   RESTARTS  AGE  
nginx-6cbc9bb4f5-jm8mr    1/1    Running  0         120d  
nginx-6cbc9bb4f5-pfhfz    1/1    Running  0         121d  
nginx2-f96cfc57b-vnsfm    1/1    Running  0         121d  
test-webserver        1/1    Running  2 (69d ago)  105d  
webserver-master01   1/1    Running  0          71d
```

kubernetesでstatic podを起動させる

<https://www.opensourcetech.tokyo/entry/20230327/1679909358>



ConfigMap/Secret/ストレージ領域

- **ConfigMap : サーバ設定ファイルなどをPodへ渡す手段**

nginxコンテナ(Pod)のコンテンツ(index.html)をConfigMapで提供・更新する(kubernetes)

<https://www.opensourcetech.tokyo/entry/20230319/1679224123>

- **Secret**

パスワードなど機密情報をPodへ渡す手段

- **ストレージ領域(volume)**

Podにストレージ領域(データ保存場所)を提供する方法

hostPath : ノード上の指定パスにデータ保存

PersistentVolume(永続的な領域)/PersistentVolumeClaim(領域要求)

※NFSやクラウドストレージをデータ保存先として利用



ReplicaSet/Deployment

■ Podに、自動復旧機能やアップデートなどの機能を付与するもの

■ 自動復旧の例

```
kubeuser@master01:~/autoscaling$ kubectl get pods -w
```

NAME	READY	STATUS	RESTARTS	AGE
hpa-nginx-668b69fd79-9qqpb	1/1	Running	0	4m9s
hpa-nginx-668b69fd79-k7d2q	1/1	Running	0	4m9s
hpa-nginx-668b69fd79-pzxdl	1/1	Running	0	4m9s
hpa-nginx-668b69fd79-v5bjk	1/1	Running	0	4m9s

コンテナの削除

```
kubeuser@master01:~/autoscaling$ kubectl delete pods hpa-nginx-668b69fd79-9qqpb --force
```

hpa-nginx-668b69fd79-9qqpb	1/1	Terminating	0	4m45s
hpa-nginx-668b69fd79-xnqfx	0/1	Pending	0	0s
hpa-nginx-668b69fd79-xnqfx	0/1	ContainerCreating	0	1s
hpa-nginx-668b69fd79-xnqfx	1/1	Running	0	8s

不足となった1台を自動復旧



ReplicaSet/Deployment(Autoscaling)

■ オートスケーリングの例

```
kubeuser@master01:~/autoscaling$ cat hpa.yaml
```

```
apiVersion: autoscaling/v2
```

```
kind: HorizontalPodAutoscaler
```

```
metadata:
```

```
  name: hpa-nginx
```

```
spec:
```

```
  scaleTargetRef:
```

```
    apiVersion: apps/v1
```

```
    kind: Deployment
```

```
    name: hpa-nginx
```

```
minReplicas: 1
```

```
maxReplicas: 10
```

```
metrics:
```

```
- type: Resource
```

```
  resource:
```

```
    name: cpu
```

```
    target:
```

```
      type: Utilization
```

```
      averageUtilization: 50
```

負荷状況によって、
最小1台～最大10台の間で自動変動する

Pod(コンテナ)のCPU使用率が50%を超過
⇒増設



Deploymentの耐障害性(自動復旧)とオートスケーリング(kubernetes)

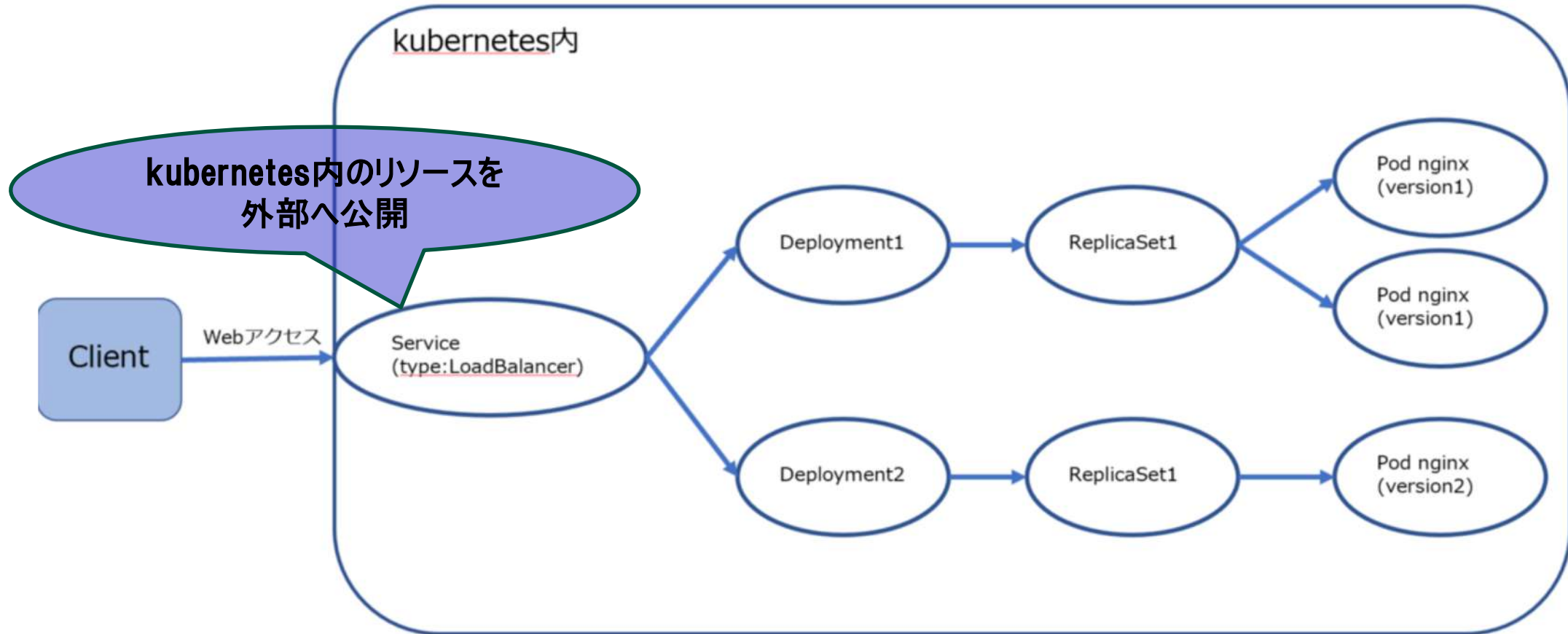
<https://www.opensourcetech.tokyo/entry/20230405/1680698675>

kubernetesにおけるPodへの負荷分散状況の確認(Service/Deployment経由)

<https://www.opensourcetech.tokyo/entry/20230326/1679837870>



- Podなどのリソースを外部からアクセスできるようにするもの(ルーティング処理)



MetalLBとService(type:LoadBalancer)による外部クライアントへのサービス公開 (kubernetes v1.26.00 on ubuntu22.04)

<https://www.opensourcetechnology.com/entry/20230316/1678966960>



■ Serviceの種類

NodePort :

kubernetesクラスターを構成するノードが持つIPアドレスの特定ポート(デフォルト設定だと、30000-32767)を公開する

LoadBalancer :

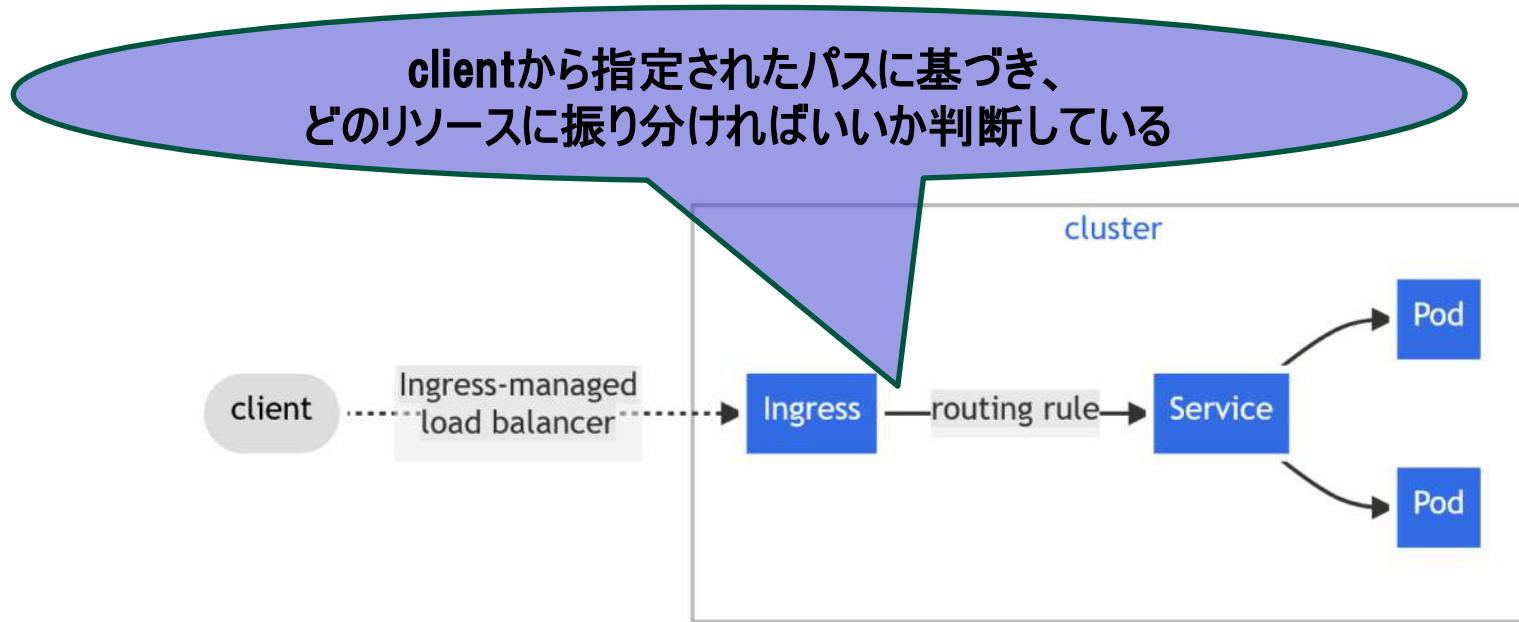
連携するLBから払い出された外部からのアクセスが可能なIPアドレスの指定ポートを公開する

ClusterIP :

kubernetesクラスター内部のPodなどからアクセスする



- 外部からのアクセスをパスに基づいて振り分ける機能(L7ロードバランサ)



<https://kubernetes.io/docs/concepts/services-networking/ingress/>



Ingress

```
kubeuser@master01:~$ cat ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-localhost
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
```

```
- http:
  paths:
  - backend:
    service:
      name: service1
    port:
      number: 80
    path: /red
    pathType: Prefix
```

```
- http:
  paths:
  - backend:
    service:
      name: service2
    port:
      number: 80
    path: /blue
    pathType: Prefix
```

/blueというパスが指定された場合、
“service2”というServiceにアクセスを割り振る

/redというパスが指定された場合、
“service1”というServiceにアクセスを割り振る

NGINX Ingress Controller + Ingressによるサービス公開(kubernetes v1.26)

<https://www.opensourcetech.tokyo/entry/20230317/1679054672>



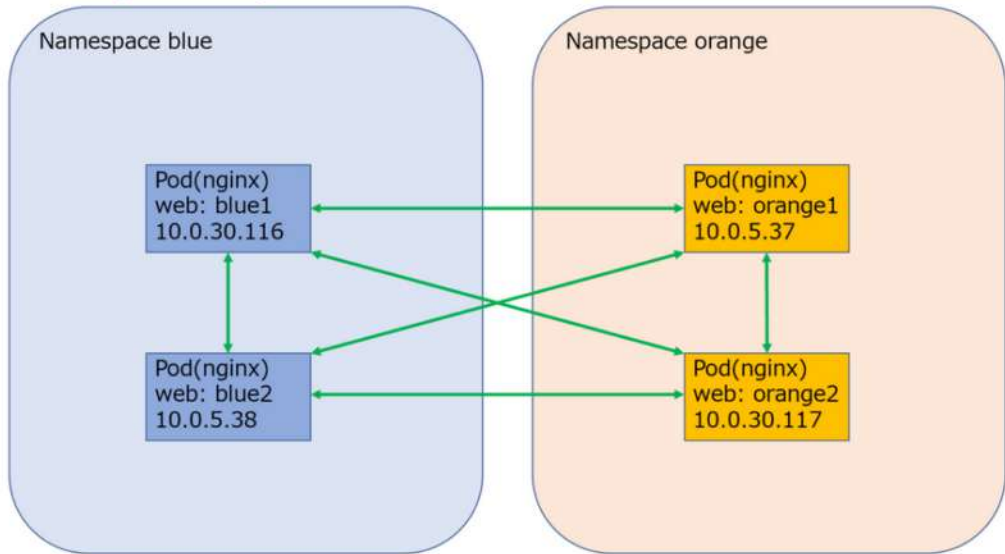
NetworkPolicy

■ クラスター内のPod間のアクセス制御するルール

デフォルト状態のPod間通信

Kubernetesクラスターを構築したデフォルト状態では、Pod間の通信は自由に行えるようになっているかと思えます。

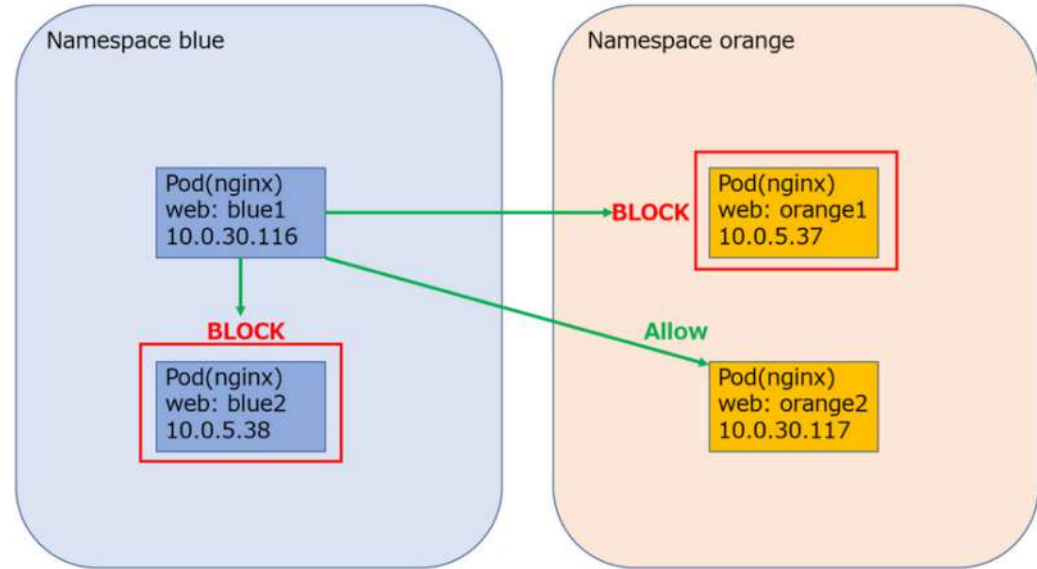
All Allow(デフォルト)



一部の通信を許可するNetwork Policyの適用

以下のようにblue1からorange2への通信を許可します。

All Deny(Ingress) + Allow(orange2 from blue1)



Network Policyによるトラフィック制御(kubernetes)

<https://www.opensourcetechnology.com/entry/20230331/1680263776>



NetworkPolicy

```
kubeuser@master01:~/networkpolicy$ cat np-all-deny.yaml
```

```
apiVersion: networking.k8s.io/v1
```

```
kind: NetworkPolicy
```

```
metadata:
```

```
  name: all-deny
```

```
  namespace: blue
```

```
spec:
```

```
  podSelector: {}
```

```
  egress:
```

```
  - {}
```

```
  policyTypes:
```

```
  - Ingress
```

```
  - Egress
```

```
---
```

```
apiVersion: networking.k8s.io/v1
```

```
kind: NetworkPolicy
```

```
metadata:
```

```
  name: all-deny
```

```
  namespace: orange
```

```
spec:
```

```
  podSelector: {}
```

```
  egress:
```

```
  - {}
```

```
  policyTypes:
```

```
  - Ingress
```

```
  - Egress
```

①全てのPod間通信を拒否するルール

```
kubeuser@master01:~/networkpolicy$ cat np-allow.yaml
```

```
apiVersion: networking.k8s.io/v1
```

```
kind: NetworkPolicy
```

```
metadata:
```

```
  name: orange2-from-blue1
```

```
  namespace: orange
```

```
spec:
```

```
  podSelector:
```

```
    matchLabels:
```

```
      web: orange2 . . . Network Policyの対象となるPodのラベル
```

```
  policyTypes:
```

```
  - Ingress . . . 制御するトラフィックの方向
```

```
  ingress:
```

```
  - from: . . . トラフィックの送信元情報
```

```
  - ipBlock: . . . IPアドレス+プレフィックスを使った送信元の制御
```

```
    cidr: 10.0.30.116/32
```

```
  - namespaceSelector: . . . 名前空間を使った送信元の制御
```

```
    matchLabels:
```

```
      name: blue
```

```
  - podSelector:
```

```
    matchLabels:
```

```
      web: blue1 . . . Podのラベルを使った送信元の制御
```

```
  ports: . . . Network Policyの対象となるPodの受信ポート & プロトコル
```

```
  - protocol: TCP
```

```
    port: 80
```

②一部のPod間通信を許可するルール



Appendix

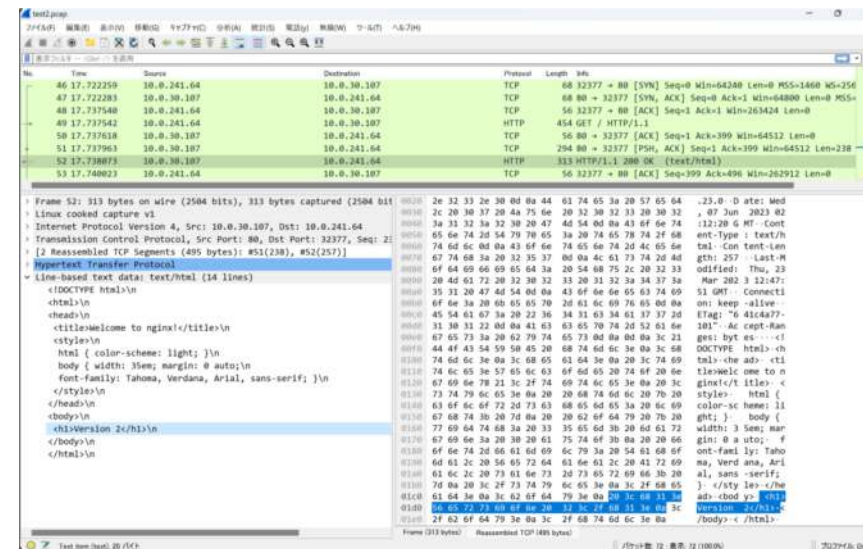


便利なツール

■ パケットキャプチャ(ksniff)

kubernetesのPod(コンテナ)のパケットキャプチャを取得する方法

<https://www.opensourcetech.tokyo/entry/20230608/1686209551>



■ stern

sternを使った複数Podのログ出力(kubernetes)

<https://www.opensourcetech.tokyo/entry/20230425/1682427182>

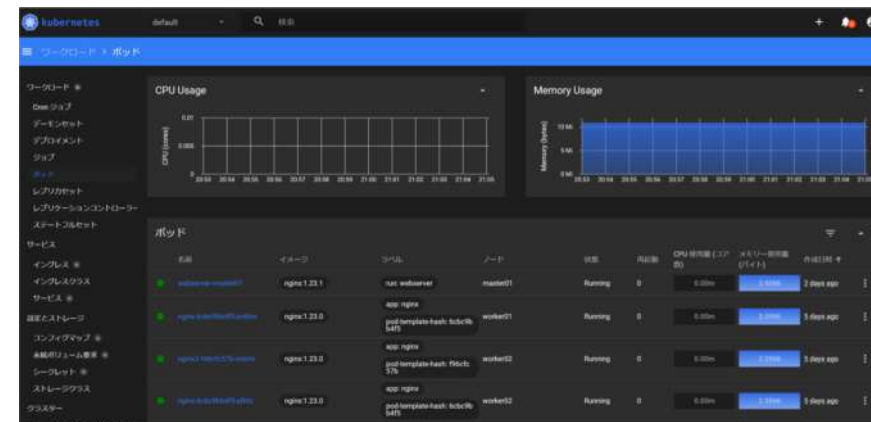
```
nginx-6cbc9bb4f5-jm8mr nginx 2023/04/25 12:46:14 [error] 31#31: *4335 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory),
client: 10.0.241.64, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "192.168.1.51", referer: "http://192.168.1.51/"
nginx-6cbc9bb4f5-pfhfz nginx 10.0.241.64 -- [25/Apr/2023:12:46:14 +0000] "GET / HTTP/1.1" 200 256 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/112.0" "-"
nginx-6cbc9bb4f5-pfhfz nginx 2023/04/25 12:46:14 [error] 32#32: *48661 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory),
client: 10.0.241.64, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "192.168.1.51", referer: "http://192.168.1.51/"
nginx-6cbc9bb4f5-pfhfz nginx 10.0.241.64 -- [25/Apr/2023:12:46:14 +0000] "GET /favicon.ico HTTP/1.1" 404 153 "http://192.168.1.51/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/112.0" "-"
nginx-6cbc9bb4f5-jm8mr nginx 10.0.241.64 -- [25/Apr/2023:12:46:15 +0000] "GET / HTTP/1.1" 200 256 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/112.0" "-"
```



■ dashboard

kubernetes dashboardを使う

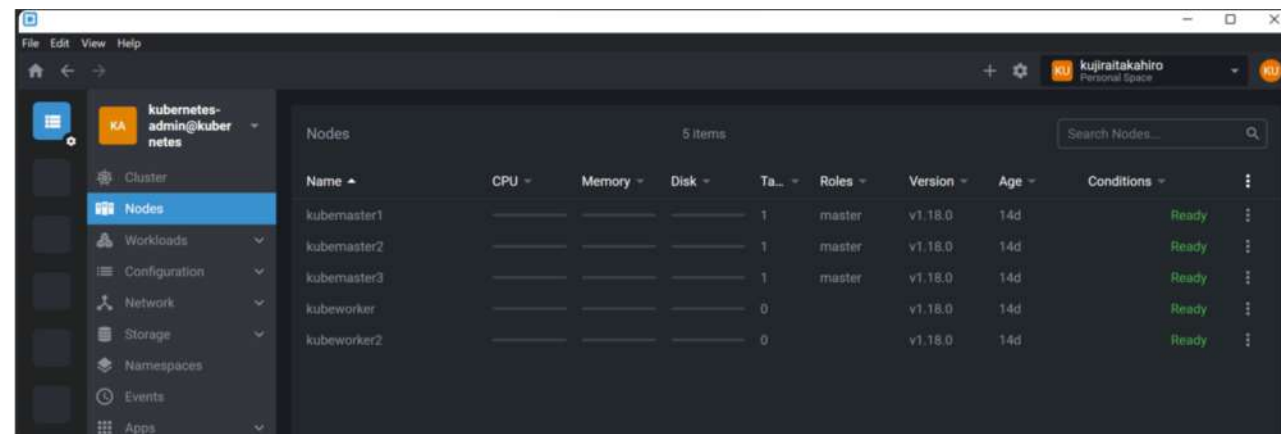
<https://www.opensourcetech.tokyo/entry/20230329/1680093328>



■ LENS

LENS(開発統合環境)でkubernetesクラスターを操作する

<https://www.opensourcetech.tokyo/entry/20211216/1639659801>



■ NewRelic

<https://newrelic.com/jp>





■ Kubernetesのベータ機能・アルファ機能を使ってみる

Feature Gates

<https://kubernetes.io/docs/reference/command-line-tools-reference/feature-gates/>

Feature gates for Alpha or Beta features

Feature	Default	Stage	Since	Until
APIListChunking	false	Alpha	1.8	1.8
APIListChunking	true	Beta	1.9	
APIPriorityAndFairness	false	Alpha	1.18	1.19
APIPriorityAndFairness	true	Beta	1.20	
APIResponseCompression	false	Alpha	1.7	1.15
APIResponseCompression	true	Beta	1.16	
APIServerDefaultPod...	...	Alpha	1.26	1.26

Kubernetes 1.27の新機能(Feature Gate)を有効化し、InPlacePodVerticalScalingを試してみる

<https://www.opensourcetech.tokyo/entry/20230514/1684069660>



コンテナを起動してみる！(改めて見てみましょう♪)

kubernetesを使った アプリケーションの起動と公開

<https://youtu.be/MAbY02dwbgw>



■ Kubernetes研修(LFS458-JP)やっています！

※講師主導型CKA(Certified Kubernetes Administrator)対策コース

ZEUS IT TRAINING CENTER

ゼウスITトレーニングセンター

LFS458-JP

kubernetes 研修

Kubernetes クラスタを 構築・管理するためのコースです。
また、Kubernetesを 管理するために
必要なスキルを 身に付けることができます。

スキルを身につけ
情報の荒波を
かきわけよう！

大規模 どこでも実行 柔軟

ご興味のある方はこ
ちらから



<https://www.it-training.tokyo/kubernetes/>

イベント運営と一緒にやってみませんか？



ボランティアスタッフって、何やるの？



<https://events.linuxfoundation.org/open-source-summit-japan/>



ご清聴ありがとうございました♪

