



Docker & Kubernetes

ハンズオン

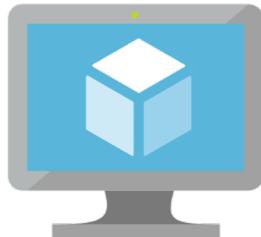


今回の実施内容

- 6. Docker イメージ動作確認
- 5. Docker イメージ作成
- 4. Dockerfile 作成
- 3. Javaのソース作成
- 2. az cli インストール
- 1. Docker インストール

- 7. Docker Login
- 8. Docker イメージPush

- 14. Ingress YAML 作成と適用
- 13. Service YAML 作成と適用
- 12. Kubernetes の基本操作
- 11. Deployment の適用
- 10. Deployment YAML 作成
- 9. ACR 接続情報の作成



踏み台 Linux VM



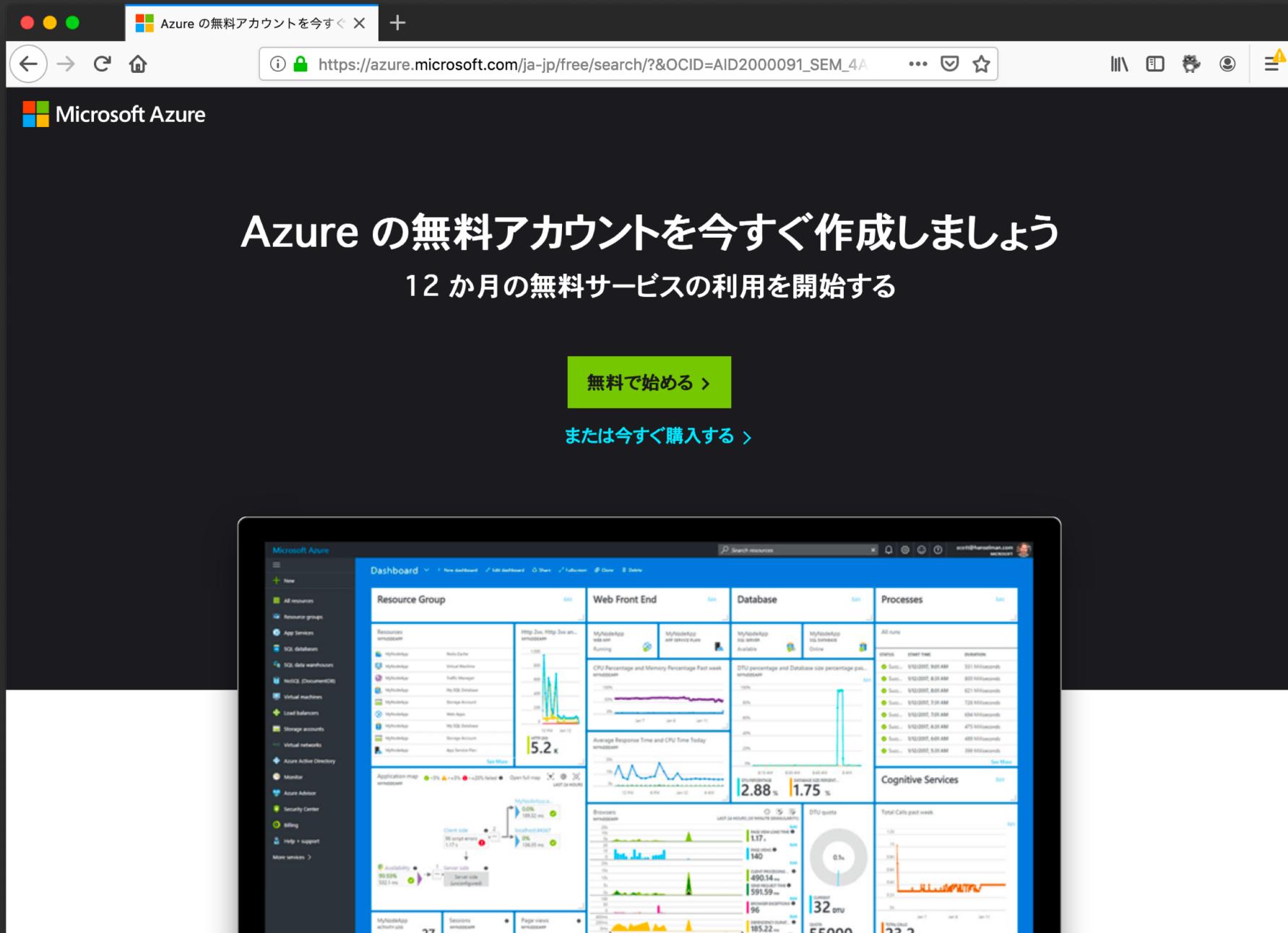
Azure Container
Registry



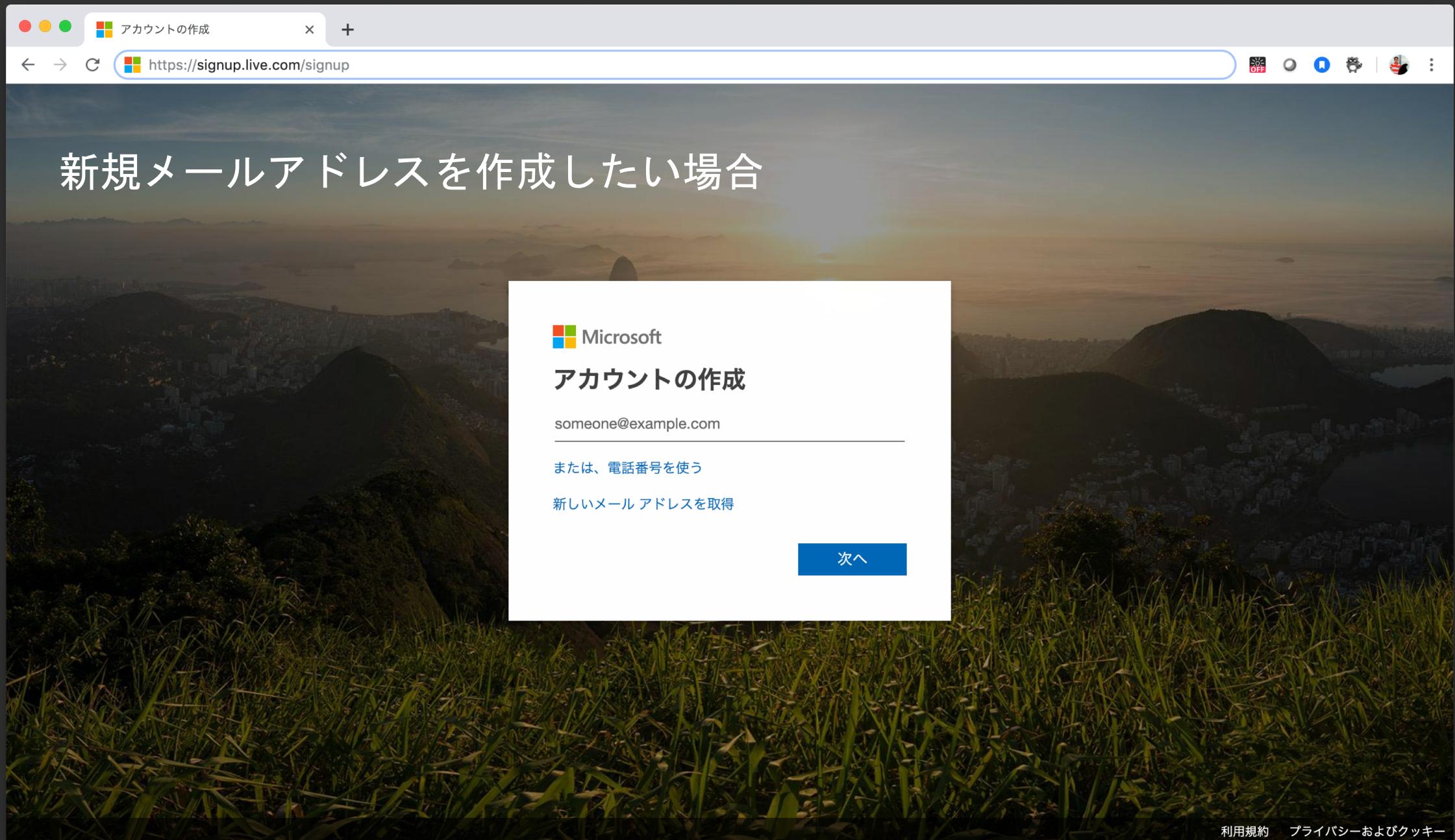
Azure Kubernetes
Service

Azure アカウント
ご作成のお願い

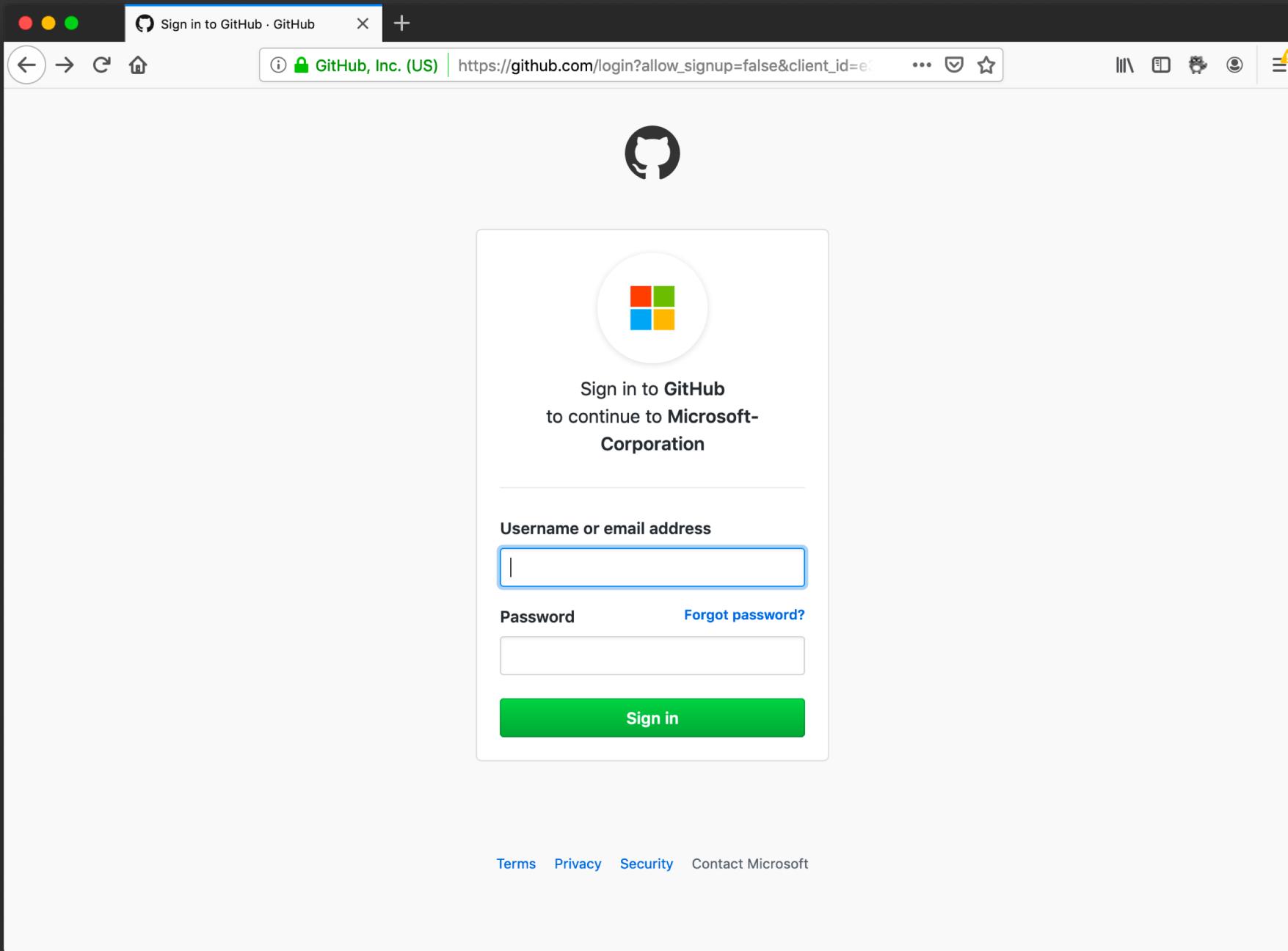




<http://aka.ms/free-account-create-jp>



<https://signup.live.com/signup>



The screenshot shows a web browser window with the GitHub logo in the top left corner. The title bar reads "GitHub - Where software is built X". The address bar shows "GitHub, Inc. (US) https://github.com/sessions/two-factor". The main content area is titled "Two-factor authentication". It features a large GitHub logo at the top, followed by the title "Two-factor authentication". Below this is a form with a yellow placeholder box labeled "Authentication code" and a blue "Verify" button. To the right of the yellow box is a link "What's this?". Below the form is a list item with a smartphone icon: "Open the two-factor authentication app on your device to view your authentication code and verify your identity." Further down, there is a section titled "Don't have your phone?" with a link "Enter a two-factor recovery code". At the bottom of the page are links for "Terms", "Privacy", "Security", and "Contact GitHub".

GitHub - Where software is built X

GitHub, Inc. (US) https://github.com/sessions/two-factor

Two-factor authentication

Authentication code [What's this?](#)

Verify

Open the two-factor authentication app on your device to view your authentication code and verify your identity.

Don't have your phone?
[Enter a two-factor recovery code](#)

[Terms](#) [Privacy](#) [Security](#) [Contact GitHub](#)

Authorize application

GitHub, Inc. (US) https://github.com/login/oauth/authorize?client_id=e37ff...

 ✓ 

Authorize Microsoft-Corporation

 Microsoft-Corporation by Microsoft-corp
wants to access your yoshioterada account

 Personal user data
Email addresses (read-only), profile information (read-only)

Authorize Microsoft-corp

Authorizing will redirect to
<https://login.live.com>

 Not owned or
operated by GitHub  Created
2 years ago  More than 1K
GitHub users

Learn more about OAuth

Azure - Sign up Microsoft Authenticator - Secur +

https://signup.azure.com/signup?offer=ms-azr-0044p&appId=102&ref=azur ... ☰ ☆

Microsoft Azure yosshi2007@gmail.com サインアウト

Azure の無料アカウントのサインアップ

30 日間の ¥ 22,500 のクレジットから開始し、引き続き無料でご利用いただけます



1 電話による本人確認

[テキスト メッセージを送信する] を選択されると、海外より本人確認の SMS をご案内します。海外から送信される SMS を拒否設定した状態では受信ができないため、ご注意ください。

「電話で確認コードを受け取る」を選択されると、海外より自動音声で確認コードをご案内します。

テキスト メッセージまたは電話により、お客様ご自身であることを確認できます。

国コード

日本 (+81)

電話番号

808 4346 852

テキスト メッセージを送信する 電話する

内容

12か月の無料製品
最初の 30 日以内、およびアカウントを従量課金制の価格にアップグレードした後の 12 か月間、仮想マシン、ストレージ、データベースなどの人気の製品を無料でご利用いただけます。

¥ 22,500 クレジット
最初の 30 日間、無料で利用できる製品以外の Azure サービスを試すには ¥ 22,500 をご使用ください。

25 以上の常に無料の製品
サーバーレス、コンテナー、人工知能を含む 25 を超える製品を、常に無料でご利用いただけます。これらは最初の 30 日間に (アップグレードを選択した場合はいつでも) 入手できます。

自動変更なし
アップグレードを選択しない限り課

今回実行する 環境のご選択



1. 自分のアカウントを利用し自分で進めたい方
2. チームを作りモブプロで実施したい方



Azure Cloud Shell をご利用ください

The screenshot shows the Microsoft Azure portal interface. A yellow box highlights the Cloud Shell icon in the top navigation bar. The main content area displays the Azure services dashboard, including links to Virtual machines, App Services, Storage accounts, SQL databases, Azure Database for PostgreSQL, Azure Cosmos DB, Kubernetes services, and Function App. Below this are four cards: Microsoft Learn, Azure Monitor, Security Center, and Cost Management. The 'Recent resources' section lists three items: yoshioAKSCluster1137 (Kubernetes service), MC-YOSHIO-AKS-1137 (Resource group), and voshio (Container registry). The bottom part of the screen shows the Cloud Shell terminal window with the following output:

```
Bash
Requesting a Cloud Shell. Succeeded.
Connecting terminal...
yoshio@Azure: ~ $
```

1. 自分のアカウントを利用し自分で進めたい方
2. チームを作りモブプロで実施したい方

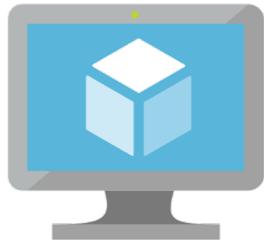


踏み台の Linux VM をご作成ください

Javaのソース作成

az cli インストール

Docker インストール



踏み台 Linux VM

新しい Linux VM (踏み台) をお作りください

The screenshot shows the Microsoft Azure portal interface. The left sidebar contains a 'FAVORITES' section with links to various Azure services: Resource groups, All resources, Recent, Virtual machines (classic), App Services, SQL databases, Virtual machines, Cloud services (classic), Subscriptions, Application Insights, Public IP addresses, SQL servers, Network security groups, Storage accounts, Network interfaces, Azure Active Directory, Monitor, Security Center, and Cost Management & Billing.

The main content area is titled 'New' and displays the 'Azure Marketplace'. A search bar at the top of the marketplace lists popular items:

| Category | Item | Action |
|----------|--------------------------------|-------------------------------------|
| Popular | Windows Server 2016 Datacenter | Quickstart tutorial |
| | Ubuntu Server 18.04 LTS | Learn more |
| | Web App | Quickstart tutorial |
| | SQL Database | Quickstart tutorial |
| | Function App | Quickstart tutorial |
| | Azure Cosmos DB | Quickstart tutorial |
| | Kubernetes Service | Quickstart tutorial |
| | DevOps Project | Quickstart tutorial |
| | Storage account | Quickstart tutorial |
| | Show recently created items | |

Create a virtual machine - Micr X +

https://ms.portal.azure.com/#create/Canonical.UbuntuServer1804LTS-ARM

Home > New > Create a virtual machine

Create a virtual machine

Basics Disks Networking Management Advanced Tags Review + create

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image.

Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization.

Looking for classic VMs? [Create VM from Azure Marketplace](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription Microsoft Azure Internal Billing-CDA

* Resource group Select existing... Create new

Instance details

* Virtual machine name

* Region (Asia Pacific) Japan East

Availability options No infrastructure redundancy required

* Image Ubuntu Server 18.04 LTS

Browse all public and private images

* Size Standard D2s v3
2 vcpus, 8 GiB memory
[Change size](#)

Review + create < Previous Next : Disks >

Linux VM インストール後すること

1. Ubuntu 18.04 に Docker をインストール

<https://qiita.com/myyasuda/items/cb8e076f4dba5c41afbc>

2. Linux に Azure CLI (az コマンドラインツールのインストール)

<http://aka.ms/az-cli-install-jp>

3. Git Clone

git clone <https://github.com/yoshioterada/k8s-Azure-Container-Service-AKS--on-Azure>

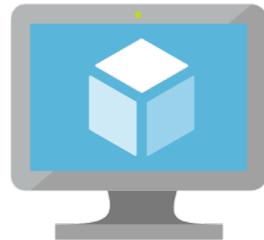
Dockerの操作



今回の実施内容

- 6. Docker イメージ動作確認
- 5. Docker イメージ作成
- 4. Dockerfile 作成

- 7. Docker Login
- 8. Docker イメージPush



踏み台 Linux VM



Azure Container
Registry

Java の Maven ビルド環境構築 – Docker 内

```
$ cd FrontService
```

```
$ cat 0-Dockerfile-for-Maven
```

```
FROM maven:3.6.1-jdk-8-slim
COPY src /usr/src/app/src
COPY pom.xml /usr/src/app
RUN mvn -f /usr/src/app/pom.xml package
RUN rm -rf src pom.xml
```

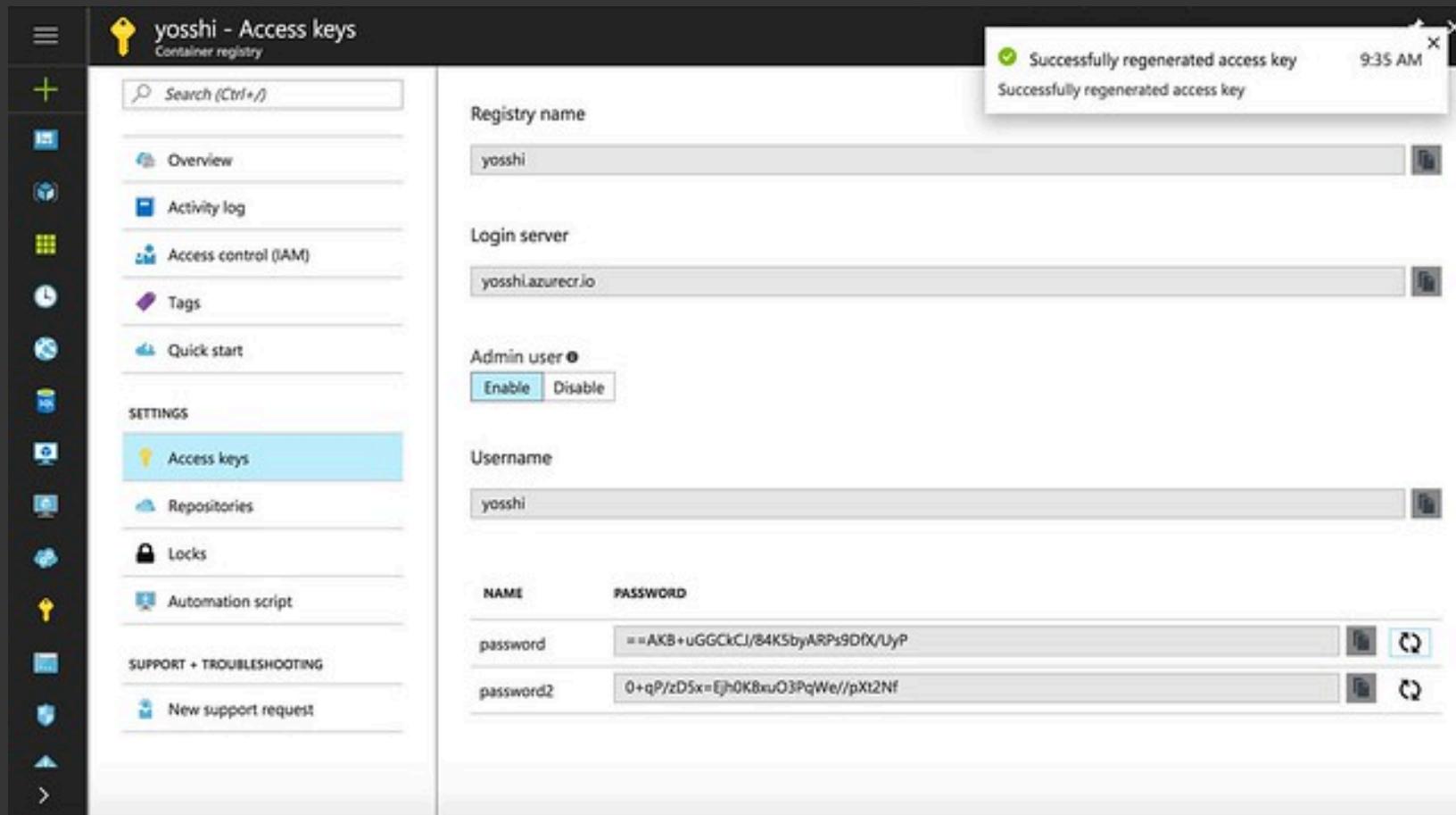
```
$ docker build -t maven-include-localrepo:1.1 . -f 0-Dockerfile-for-Maven
```

Java の Maven ビルド環境構築 – Docker 内

```
$ docker images | grep maven-include
```

| | | | | |
|-------------------------|-----|--------------|--------------|-------|
| maven-include-localrepo | 1.1 | 5d433a869411 | 15 hours ago | 451MB |
|-------------------------|-----|--------------|--------------|-------|

Azure Container Registry の作成



<https://github.com/yoshioterada/DEIS-k8s-ACS/blob/master/CreateAzureContainerRegistry.md>

ACR へログイン – docker login の実行

Connection Informations

After created the above Azure Container Registry, you can access to the Registry Server with following informations.

| Container Registry Connection Info | Value |
|------------------------------------|-------------------|
| HostName : | yosshi.azurecr.io |
| User Name : | yosshi |
| Password : | [Password] |

Push Docker image to Azure Container Registry

In this section, we will upload the Docker image from local environment to the Azure Container Service.

Login to the Azure Container Registry

At first, please login to the Azure Container Registry by using "docker login" command like follows.

```
local$ docker login -u yosshi -p "[password]" yosshi.azurecr.io
Login Succeeded
```

ソースコード・ビルドとイメージ作成

```
$ cat 1-Dockerfile-Multi
FROM maven-include-localrepo:1.1 AS build
COPY src /usr/src/app/src
COPY pom.xml /usr/src/app
RUN mvn -f /usr/src/app/pom.xml clean package

#####
# Build container image copying from BUILD artifact
#####
FROM mcr.microsoft.com/java/jdk:8u212-zulu-alpine
#FROM mcr.microsoft.com/java/jdk:11u3-zulu-alpine

# create directory for application
RUN mkdir /app
WORKDIR /app
# add user for application
RUN adduser -S java
USER java

COPY --from=build /usr/src/app/target/front-spring-0.0.1-SNAPSHOT.jar app.jar

# set entrypoint to execute spring boot application
ENV JAVA_OPTS=""
ENTRYPOINT [ "sh", "-c", "java $JAVA_OPTS -Djava.security.egd=file:/dev/.urandom -jar app.jar" ]
```

ビルド・スクリプトの編集 (2-build-create.sh) 橙色箇所を適宜編集

```
DOCKER_IMAGE=tyoshio2002/front-spring-service
DOCKER_REPOSITORY=yoshio.azurecr.io ← ACR のホスト名

# Build docker image
docker build -t $DOCKER_IMAGE:$VERSION . -f 1-
Dockerfile-Multi

docker tag $DOCKER_IMAGE:$VERSION
$DOCKER_REPOSITORY/$DOCKER_IMAGE:$VERSION
# Push the image to Private Docker Registry
docker push $DOCKER_REPOSITORY/$DOCKER_IMAGE:$VERSION
```

ビルド・スクリプトの実行

```
$ ./2-build-create.sh 1.0
```

ソースコードを編集したのち

```
$ ./2-build-create.sh 1.1
```

ソースコードを編集したのち

```
$ ./2-build-create.sh 1.2
```

Docker のイメージの起動

```
$ docker run -p 8080:8080 \
-it tyoshio2002/front-spring-service:1.0
```

動作確認

```
$ curl http://localhost:8080/sample/hello  
front-hello-v1
```

ソースコードの修正 – 新規バージョン作成

```
$ vi src/main/java/com/yoshio3/frontspring/  
FrontController.java
```

修正箇所

```
return "front-hello-v1";
```

修正後

```
return "front-hello-v2";
```

ビルド・スクリプトの実行と確認

```
$ ./2-build-create.sh 1.1
```

```
$ docker run -p 8080:8080 ¥  
-it tyoshio2002/front-spring-service:1.1
```

```
$ curl http://localhost:8080/sample/hello  
front-hello-v2
```

ACR にイメージが Push されているか確認

The screenshot shows the Azure portal interface for managing a Container Registry named 'yoshio'. The left sidebar contains a 'FAVORITES' section with various service icons, and the main area displays the 'yoshio - Repositories' page. The 'REPOSITORIES' section lists several repositories, with 'tyoshio2002/front-spring-service' selected. The right panel provides detailed information about this repository, including its tag count (9), manifest count (8), last updated date (8/2/2019, 12:36 PM GMT+9), and a list of tags (1.0, 1.8, 1.7, 1.6, 1.5, 1.4, 1.3, 1.2, 1.1).

Home > Resource groups > ContainerReg > yoshio - Repositories > tyoshio2002/front-spring-service

yoshio - Repositories
Container registry

Search (Ctrl+ /)

Refresh

Overview

Activity log

Access control (IAM)

Tags

Quick start

Events

Settings

Access keys

Firewalls and virtual networks (...)

Locks

Export template

Services

Repositories

Webhooks

Replications

Tasks

Policies

Content trust

Monitoring

tyoshio2002/front-spring-service
Repository

tyoshio2002/front-spring-service

Last updated date
8/2/2019, 12:36 PM GMT+9

Tag count
9

Manifest count
8

Search to filter tags ...

TAGS

1.0

1.8

1.7

1.6

1.5

1.4

1.3

1.2

1.1

AKS インストール



AKS のインストール方法

CLI によるインストール

<http://aka.ms/aks-install-cli-jp>

GUI によるインストール

<http://aka.ms/aks-install-gui-jp>

Kubectl のインストールと接続情報の取得

Kubectl コマンドのインストール

```
az aks install-cli
```

AKS 接続用の資格情報の取得

```
az aks get-credentials --resource-group  
myResourceGroup --name myAKScluster
```

Kubectl クラスタ情報、バージョン、ノード

k8s クラスタの情報

```
$ kubectl cluster-info
```

k8s のバージョン

```
$ kubectl version
```

k8s のノードの情報

```
$ kubectl get node
```

今回の実施内容

7. Docker Login

8. Docker イメージPush

14. Ingress YAML 作成と適用

13. Service YAML 作成と適用

12. Kubernetes の基本操作

11. Deployment の適用

10. Deployment YAML 作成

9. ACR 接続情報の作成



Azure Container
Registry



Azure Kubernetes
Service

Kubectl クラスタ情報、バージョン、ノード

ACR 接続情報の作成

```
$ kubectl create secret docker-registry ¥  
docker-reg-credential ¥  
--docker-server=yoshio.azurecr.io ¥  
--docker-username=yoshio ¥  
--docker-password="*****¥*****" ¥  
--docker-email=foo-bar@microsoft.com
```

Deployment YAML の作成 – (4-create-deployment-svc.yaml)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-front-service
spec:
  replicas: 2
  selector:
    matchLabels:
      app: spring-front-service
#  minReadySeconds: 120
#  progressDeadlineSeconds: 600
#  strategy:
#    type: RollingUpdate
#    rollingUpdate:
#      maxSurge: 50%
#      maxUnavailable: 50%
  template:
    metadata:
      labels:
        app: spring-front-service
        version: v1
        stage: staging
    spec:
      securityContext:
        runAsUser: 1000
      imagePullSecrets:
        - name: docker-reg-credential
      containers:
        - name: spring-front-service
          image: yoshio.azurecr.io/tyoshio2002/front-spring-service:1.7
```

修正箇所

```
image:
yoshio.azurecr.io/tyoshio2002/front-spring-service:1.7
```

ご自身で作成した ACR サーバ名、
コンテナのイメージ名に置き換えてください

Deployment YAML の適用

build スクリプトの実行

```
$ ./4-1-build-create.sh 1.1
```

内部的には sed でバージョンの書き換えと apply を実行

```
# Change the version for docker image inside of YAML file
sed -i -e "s|image: .*|image:
$DOCKER_REPOSITORY/$DOCKER_IMAGE:${VERSION}|g" 4-create-deployment-
svc.yaml

# Apply the new Image to the Service
kubectl apply --record -f 4-create-deployment-svc.yaml
```

※慣れてくると 2-build-create.sh の中で上記を実行し 1 回で apply まで可能

kubectl コマンド

```
$ kubectl get deployment
```

| NAME | READY | UP-TO-DATE | AVAILABLE | AGE |
|----------------------|-------|------------|-----------|-----|
| spring-front-service | 2/2 | 2 | 2 | 18h |

```
$ kubectl get po
```

| NAME | READY | STATUS | RESTARTS | AGE |
|--------------------------------------|-------|---------|----------|-----|
| spring-front-service-c65c977fd-68jkf | 1/1 | Running | 0 | 17h |
| spring-front-service-c65c977fd-kshzc | 1/1 | Running | 0 | 17h |

```
$ kubectl logs spring-front-service-c65c977fd-68jkf
```

アプリケーションの動作確認 – Port Forward

port-forward で k8s の pod のポートをローカルにフォワード

```
$ kubectl port-forward spring-front-service-c65c977fd-68jkf 8080:8080
Forwarding from 127.0.0.1:8080 -> 8080
```

ローカルホストに接続し動作確認

```
$ curl localhost:8080/sample/hello
front-hello-v1
```

Kubernetes における最小単位は Pod



Pod

Pod は複数のコンテナを束ねた物

- 依存関係の強いコンテナ同士を繋げローカル接続が可能
 - IP アドレス, ポート番号などを共有
- 自作サービスの起動前に初期化処理を実施 (Init Container)
- Proxy を経由したサービス呼び出し (Side Car: Envoy)
 - プログラミングで言う所の AOP をコンテナで実現可能
 - 実処理前にインターセプトして他の作業を実施可能

Pod の特徴

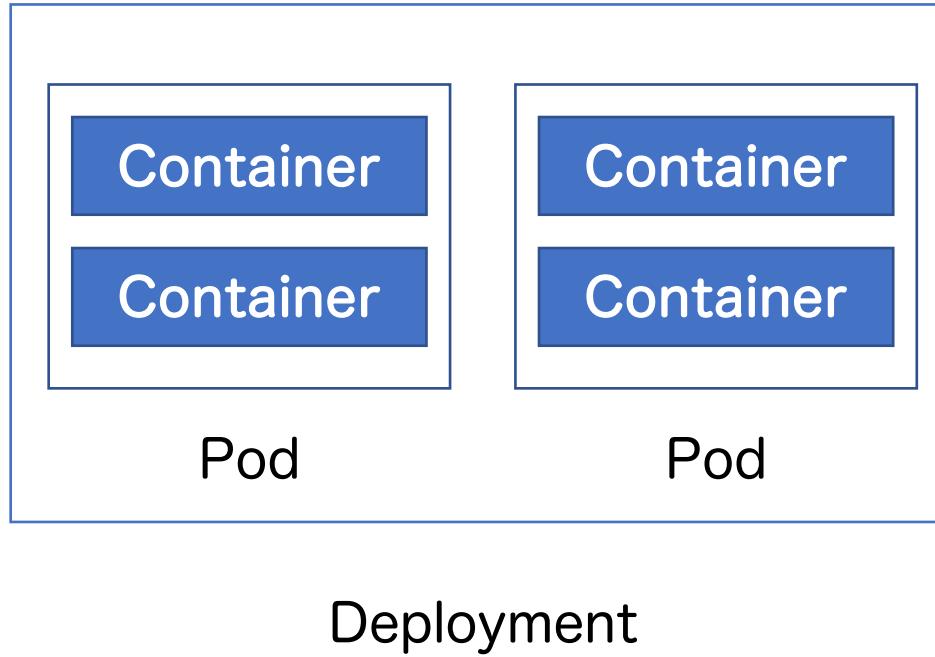


Pod

Pod は揮発性で単体で動作

- ローリング・アップグレード機能はなし
- ヘルス・チェック機能はなし
- 自動修復機能はなし
- スケール・オートスケール機能もなし

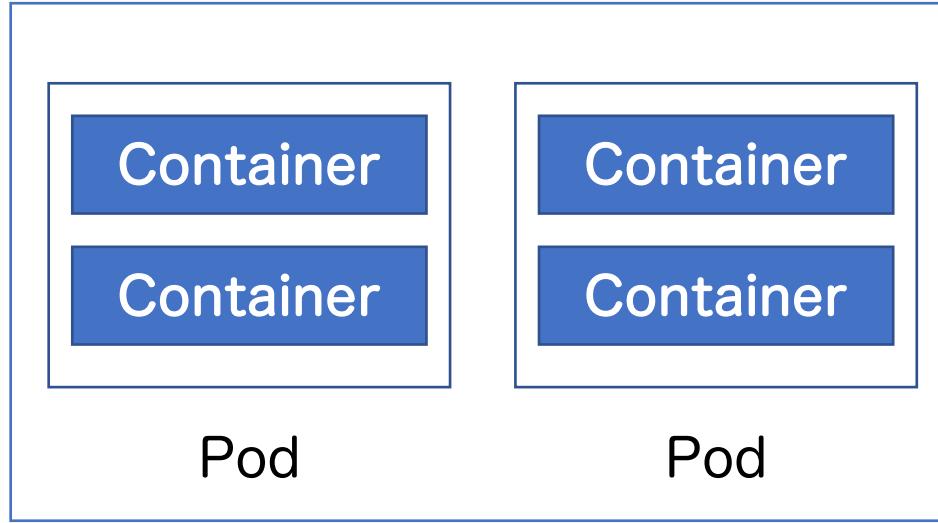
Deployment の利用



Deployment で提供

- ローリング・アップグレード機能
- ヘルス・チェック機能
- 自動修復機能
- スケール機能 (replicas)

Deployment Label によるフィルタリング

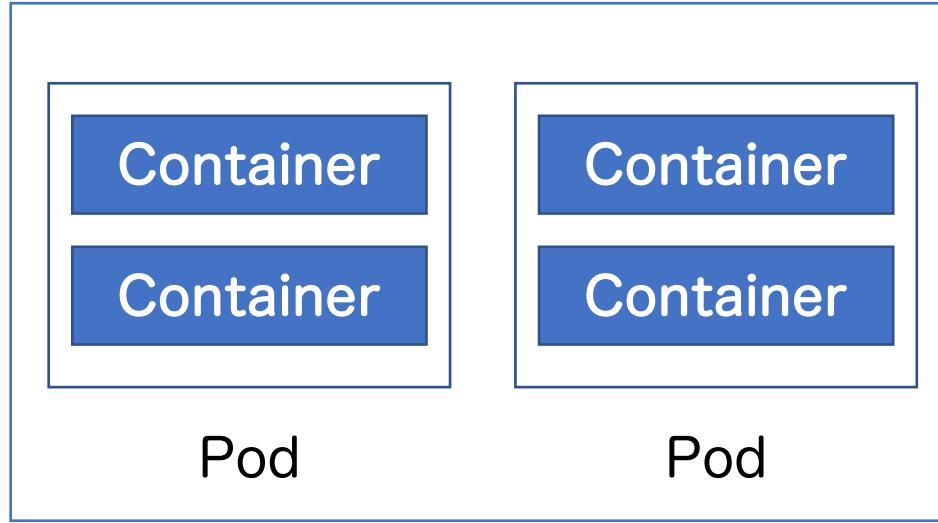


Deployment-v1

```
template:  
metadata:  
labels:  
  app: spring-front-service  
version: v1  
stage: staging
```

```
$ kubectl get po --selector app=spring-front-service
```

Deployment Label によるフィルタリング

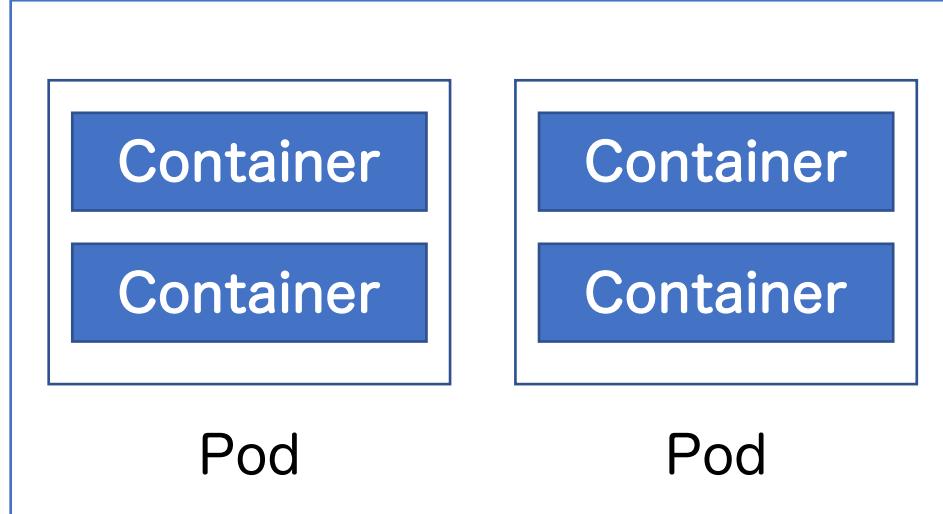


Deployment-v2

```
template:  
metadata:  
labels:  
  app: spring-front-service  
  version: v2  
  stage: staging
```

```
$ kubectl get po --selector ¥  
  app=spring-front-service,version=v2
```

サービスへの接続は？

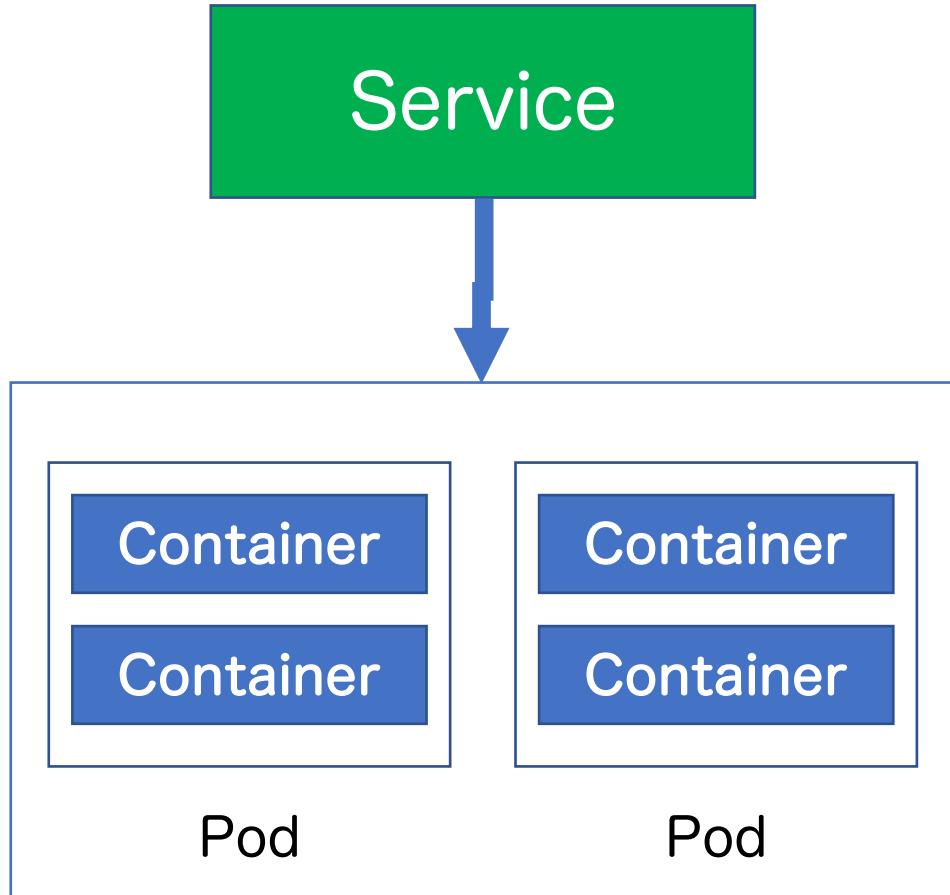


今まで、Kubectl get po でPodの一覧を取得し、port-forward コマンドでローカルにポートをフォワードし動作確認

Pod 每に IP アドレスが振られているが、接続するためには全ての IP アドレスを覚えてなければならない？

Pod は再起動すると IP アドレスが変更

Service を利用



Deployment-v1

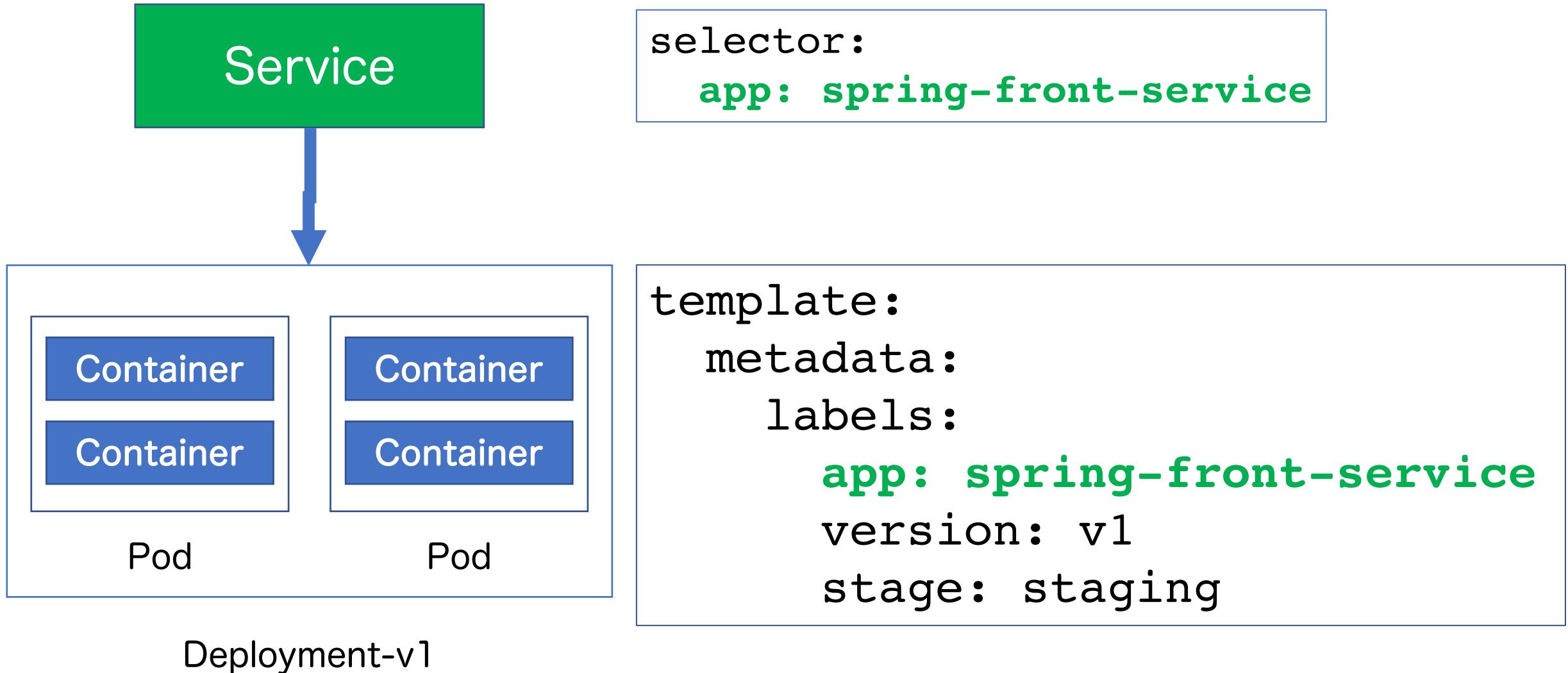
Service で利用する Pod 一覧を取得

Service は一意の IP アドレス, DNS 名を保持

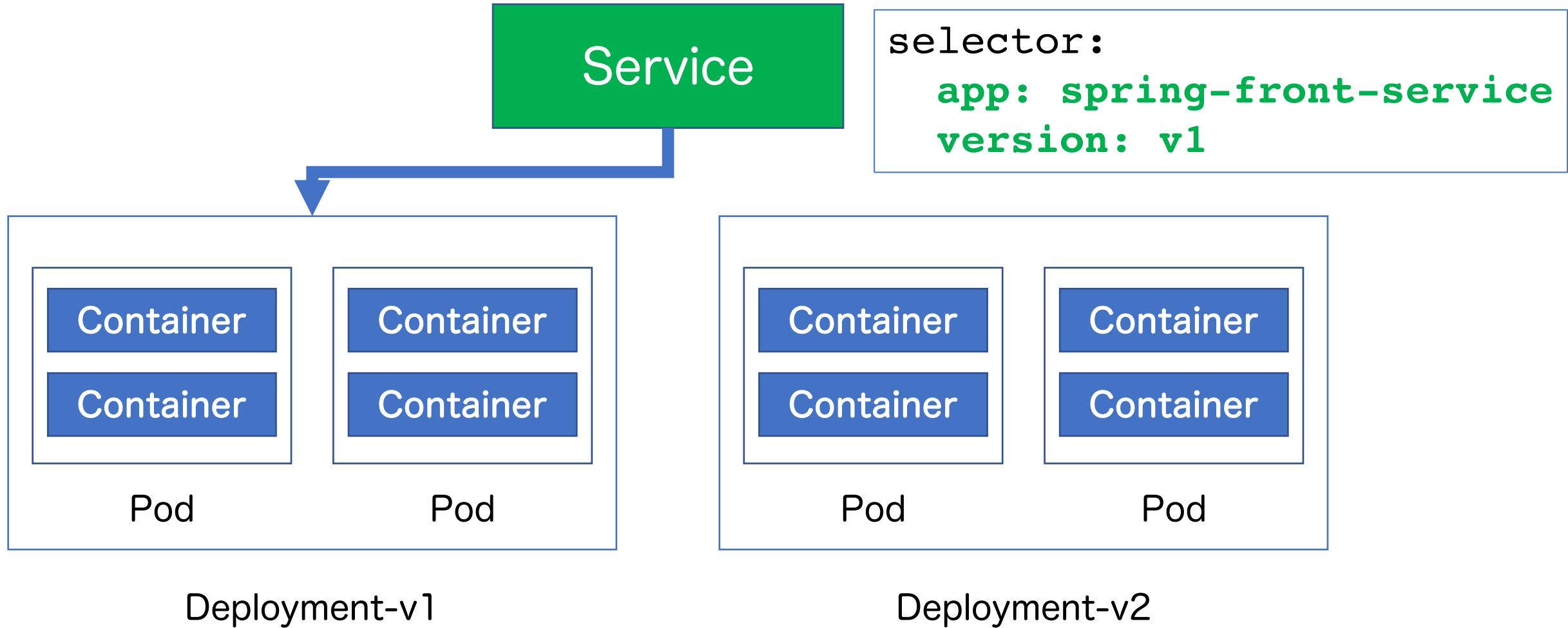
(DNS 名の例 : spring-front-service.default.svc.cluster.local)

設定により内部ネットワーク、外部ネットワークから接続可能

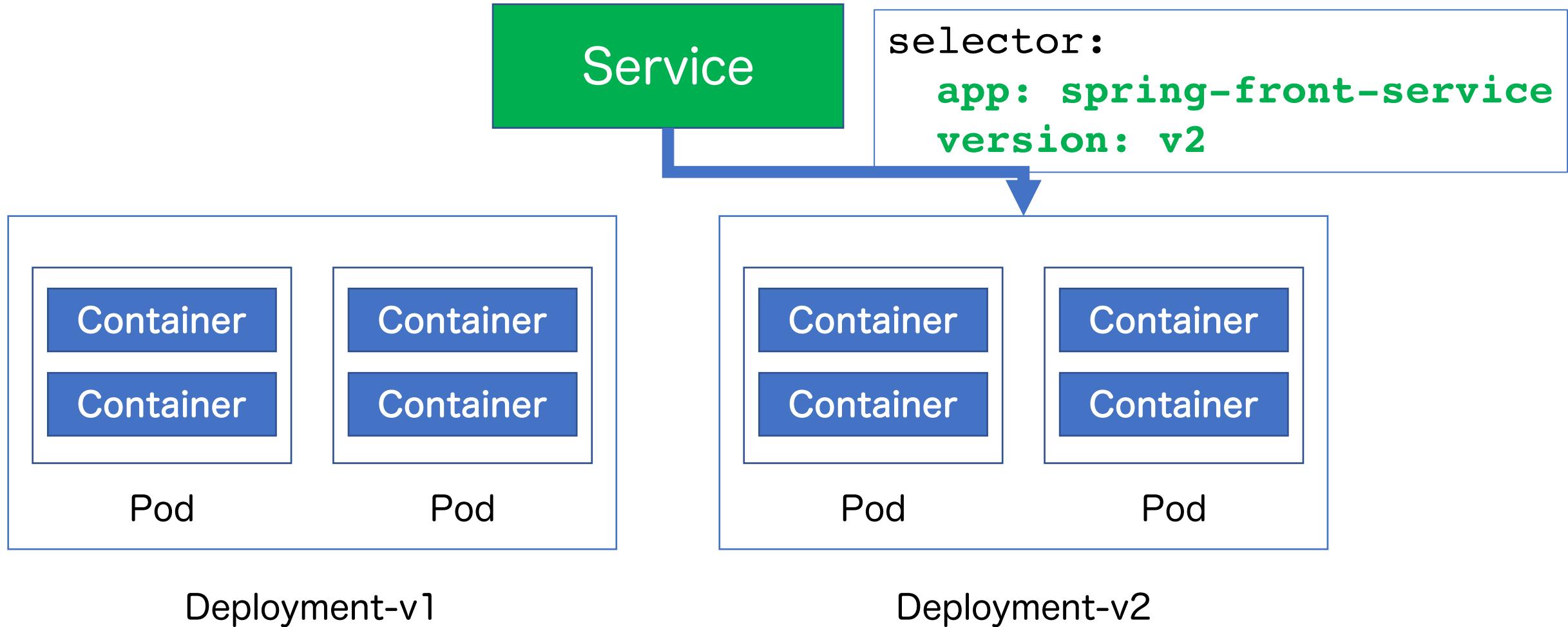
Service による Pod のフィルタリング



Service による切り替え Version 1へ



Service による切り替え Version 2へ



Service の作成

Service YAML を適用

```
$ kubectl apply 11-Service.yaml
```

Service 作成の確認

```
$ kubectl get svc
```

| NAME | TYPE | CLUSTER-IP | EXTERNAL-IP | PORT(S) | AGE |
|----------------------|-----------|------------|-------------|---------|-----|
| kubernetes | ClusterIP | 10.0.0.1 | <none> | 443/TCP | 16d |
| spring-front-service | ClusterIP | 10.0.74.52 | <none> | 80/TCP | 18h |

危険：一時的に外部 IP の割り当て

直接既存の設定を変更 (非推奨：通常はファイル編集後 apply を実施)

```
$ kubectl edit svc/spring-front-service
```

修正箇所

```
type: ClusterIP
```

修正後

```
type: LoadBalancer
```

危険：一時的に外部 IP の割り当て-続き

直接既存の設定を変更 (非推奨：通常はファイル編集後 apply を実施)

| \$ kubectl get svc -w | | | | | |
|-----------------------|--------------|------------|--------------|--------------|-----|
| NAME | TYPE | CLUSTER-IP | EXTERNAL-IP | PORT(S) | AGE |
| kubernetes | ClusterIP | 10.0.0.1 | <none> | 443/TCP | 16d |
| spring-front-service | LoadBalancer | 10.0.74.52 | <pending> | 80:32314/TCP | 18h |
| NAME | | | | | |
| spring-front-service | LoadBalancer | 10.0.74.52 | 20.***.*.153 | 80:32314/TCP | 18h |

パブリック IP の割り当てあと

```
$ curl http://20.188.1.153/sample/hello  
front-hello-v1
```

元の内部IPだけの割り当てに戻す

```
$ kubectl edit svc/spring-front-service
```

修正箇所

```
nodePort: 32314 (この行が自動的に追加されているので削除)  
type: LoadBalancer
```

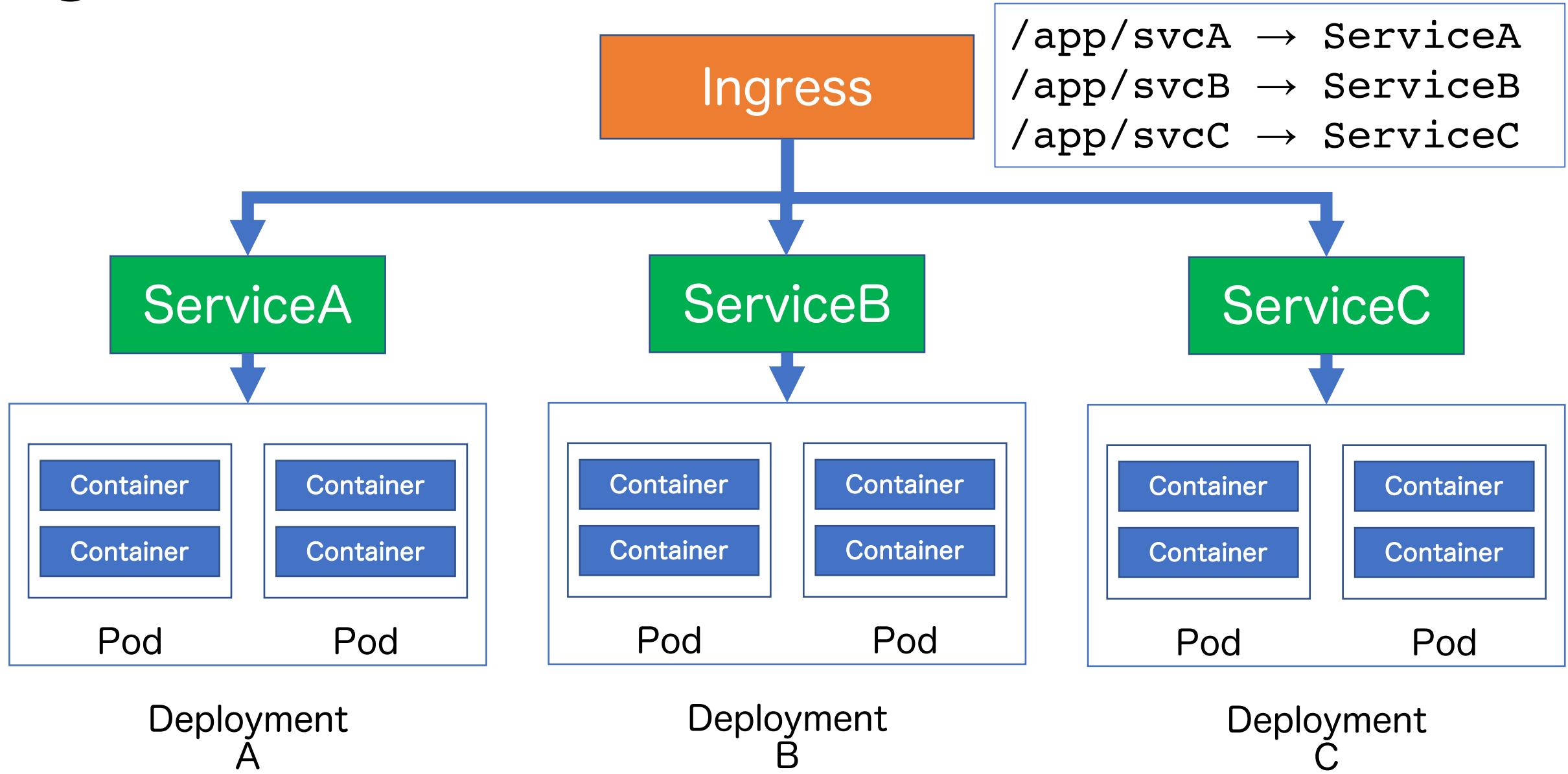
修正後

```
type: ClusterIP
```

Private IP ではサービスを外部に公開できない



Ingress を利用



HTTP アプリケーションルーティングを利用する場合

```
$ kubectl apply -f 11-Service.yaml
```

Ingress のホスト名、IPアドレスの確認

```
$ kubectl get ing -w
```

| NAME | HOSTS | ADDRESS | PORTS | AGE |
|---------------|--|-----------|-------|-----|
| front-service | front-service.****.japaneast.aksapp.io | 23.*.*.97 | 80 | 18h |

外部の IP アドレス、ホスト名経由でサービスにアクセス

```
$ curl front-service.****.japaneast.aksapp.io/sample/hello  
front-hello-v1
```

HTTP アプリケーションルーティングを利用する場合

```
$ kubectl apply -f 11-Service.yaml
```

確認

```
$ kubectl get ing
```

| NAME | HOSTS | ADDRESS | PORTS | AGE |
|---------------|---|-----------|-------|-----|
| front-service | front-service.*****.japaneast.aksapp.io | 23.*.*.97 | 80 | 18h |

終了！！
Great Jobs!!





Docker & Kubernetes

ハンズオン

