0から始める
OpenStack on Kubernetesの導入と運用

日本電信電話株式会社
伊藤 広樹

NTTコムウェア株式会社
平井 普

自己紹介

- 伊藤 広樹　(所属: 日本電信電話株式会社)
  2015年～2016年　　社内OpenStack基盤の運用
  2017年～(現在)　　Blazarにコントリビュート開始

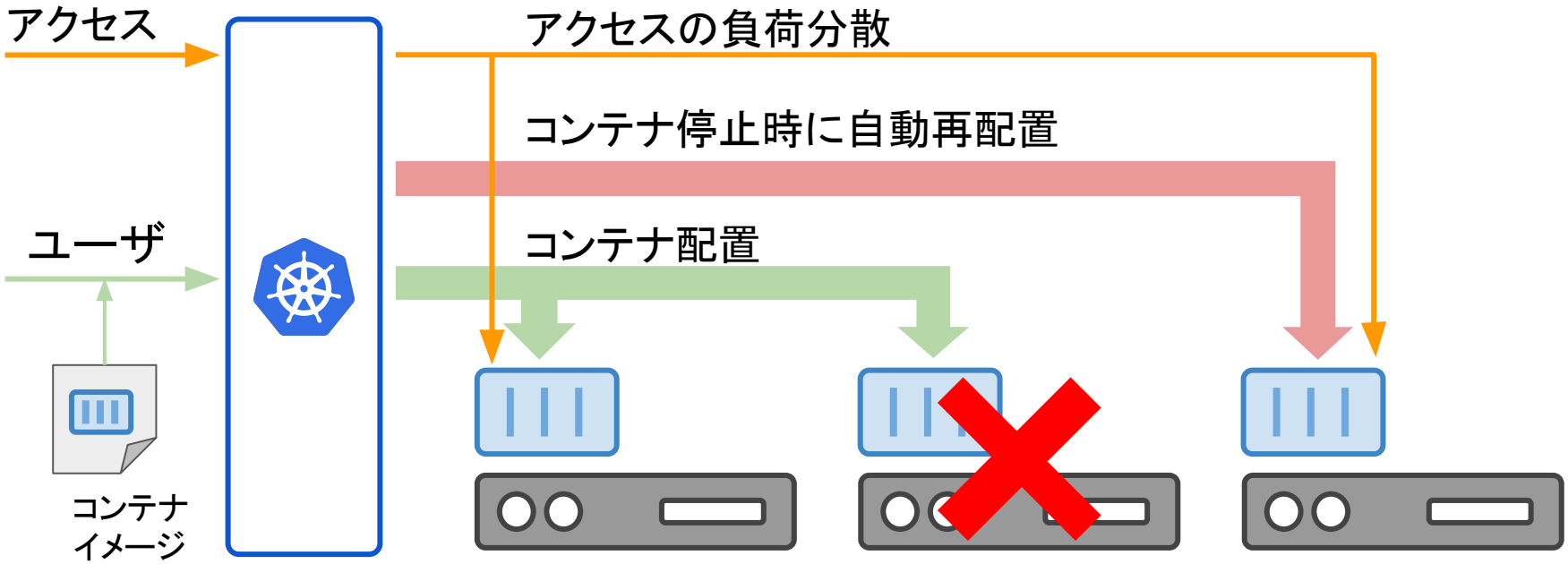
- 平井 普　(所属：NTTコムウェア)
  2011年～2016年　　NTT通信網NW機器の運用設定
  2016年～(現在)　　OpenStack基盤の運用

最近, 名前はよく聞くけど

Kubernetesって何がいいの??

Kubernetes

- **複数サーバ上に跨ったコンテナ管理のための機能を実現するOSS**

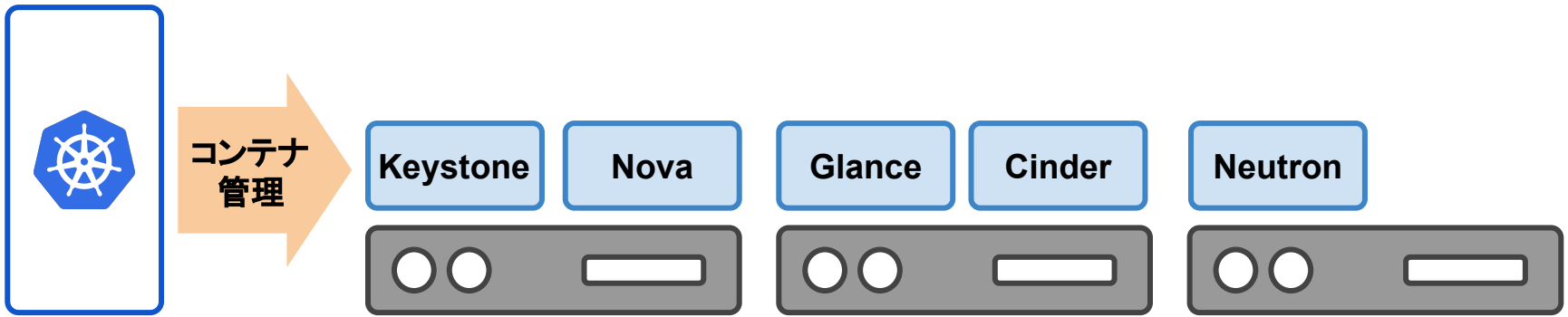


じゃあ, Kubernetesを使えば

OpenStackの構築/運用は楽になる??

OpenStack on Kubernetes

- Kubernetesを使うことで, PacemakerとHAproxyを使った場合と比較して何か良いことがある??



その効果を明らかにするため,

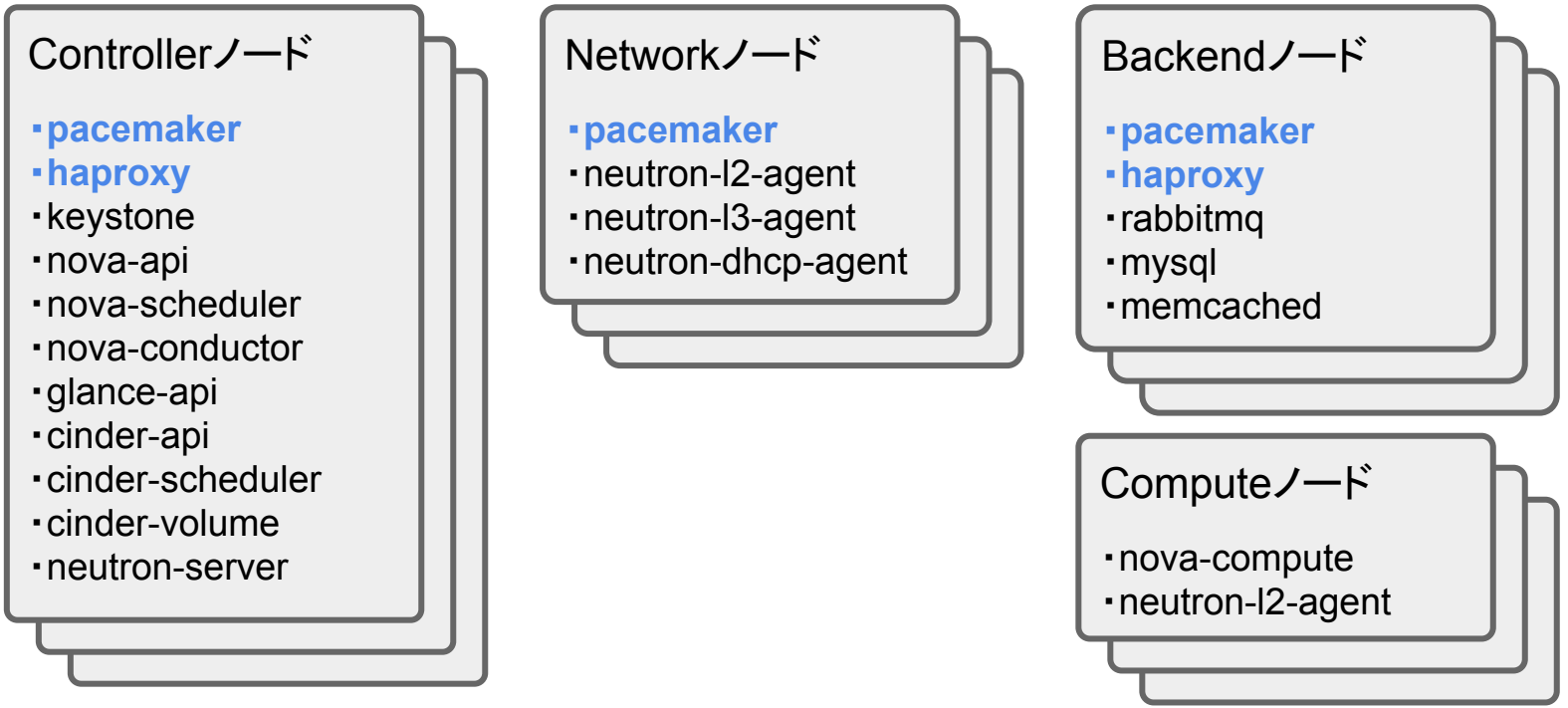
コンテナ&Kubernetes初心者 の私たちが OpenStack on Kubernetes を 0から始めてみた

アジェンダ

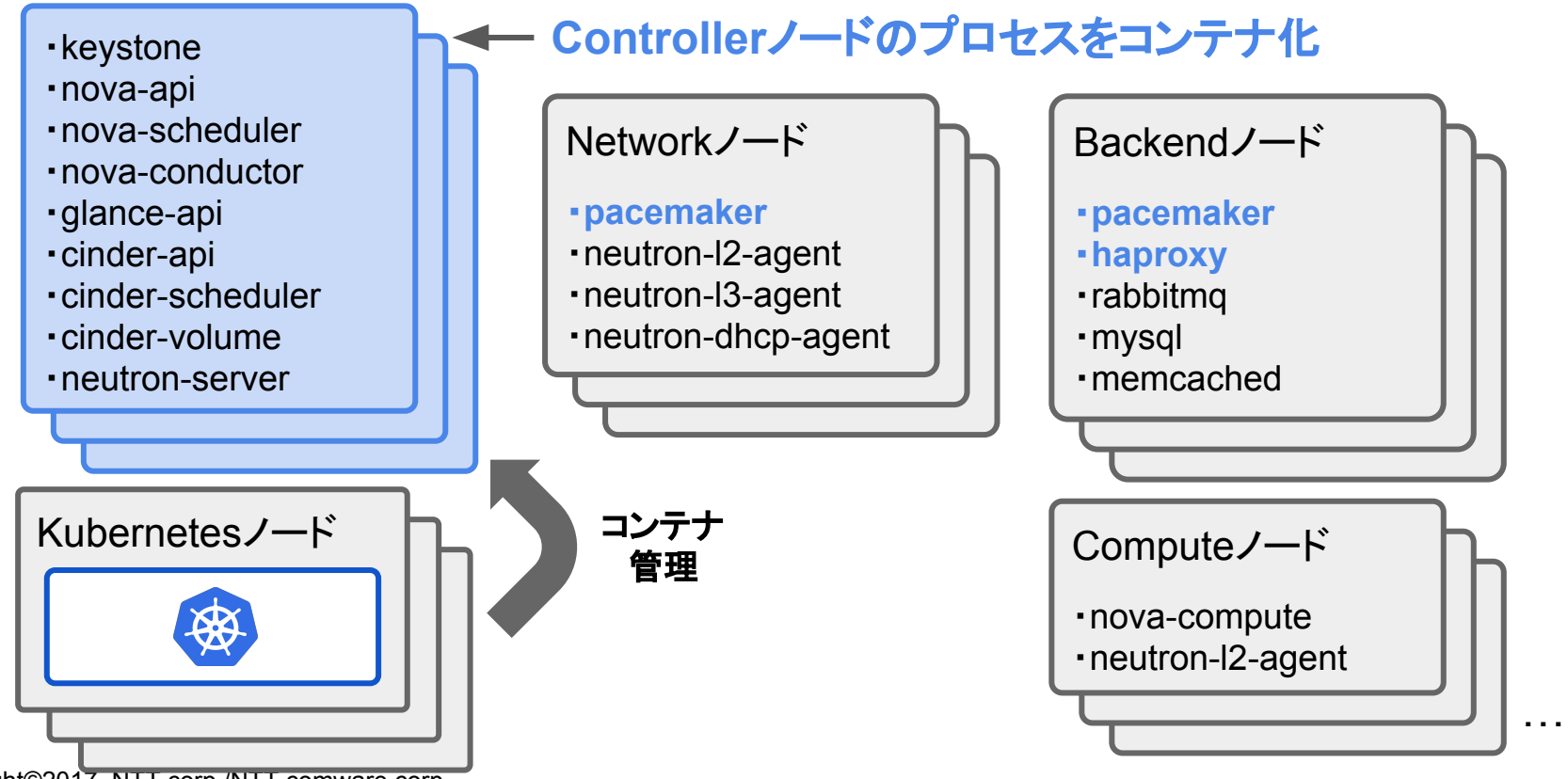
- 0から始める前に
- 0から始めるOpenStack on Kubernetesの導入
- 0から始めるOpenStack on Kubernetesの運用
- 0から始めたまとめ

0から始める前に

現状の標準構成 (コミュニティ推奨)



目標とする構成 (Kubernetes利用)



Kubernetes適用に向けた方針

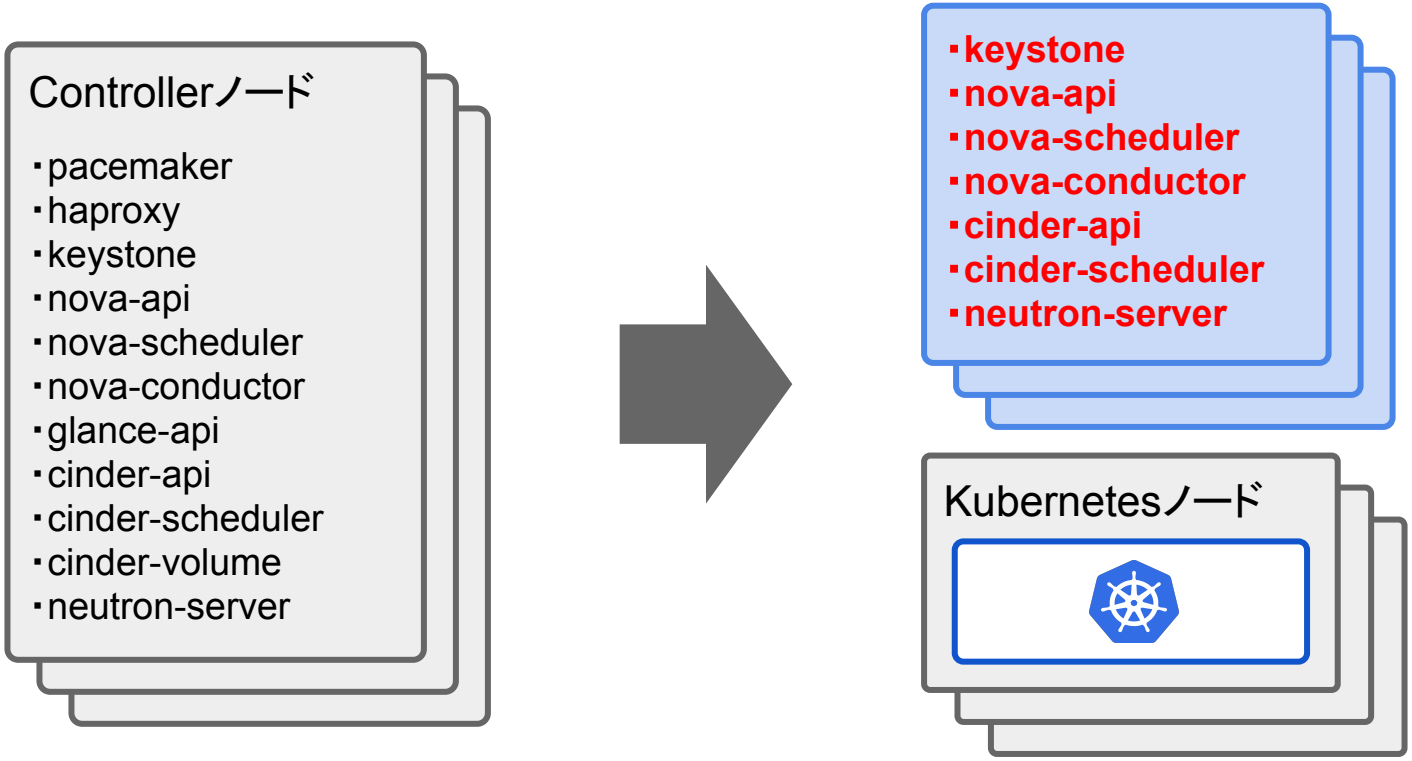
- ControllerノードのOpenStackプロセスをKubernetes上で管理する
- KubernetesおよびOpenStackの各プロセスはできる限り冗長化する
- できるだけ一般的かつ簡単な構成にする
- 必要なもののみ自作する

Kubernetes化する環境

- **OpenStackコミュニティで稼働中の環境をKubernetes化**
  - MasakariプロジェクトのCIシステムが対象
    - CI試験がOpenStack上で行われている
    - コンポーネント: Keystone, Nova, Glance, Cinder, Neutron
    - バージョン: Newton版

[1] https://wiki.openstack.org/wiki/Masakari
[2] http://openstackdays.com/program-detail/#d1p2s9

達成状況



0から始めるOpenStack on Kubernetesの導入

既存のプロジェクトで実現できる?

- コミュニティの既存プロジェクト
  - **Kolla:** Dockerイメージの作成＋Kubernetes上への配布
  - **OpenStack-helm:** Helmを使ったOpenStackの構築/バージョン管理

- **既存プロジェクトは新規構築のみをサポートしている**
  - 今回は構築済みの環境をKubernetes化したい
- **既存プロジェクトではKubernetesクラスタの構築はサポート外**
  - Kubernetesクラスタは独自に設計/構築する必要がある

OpenStack on Kubernetes導入までの道のり

STEP1: Kubernetesクラスタの設計

- Kubernetesのプロセス配置と冗長化の設計

STEP2: Kubernetesクラスタ上のOpenStackの設計

- Kubernetesのリソース作成

STEP3: 実際に環境構築

- Kubernetes環境構築/OpenStackデプロイ/構築試験

OpenStack on Kubernetes導入までの道のり

STEP1: Kubernetesクラスタの設計

- Kubernetesのプロセス配置と冗長化の設計

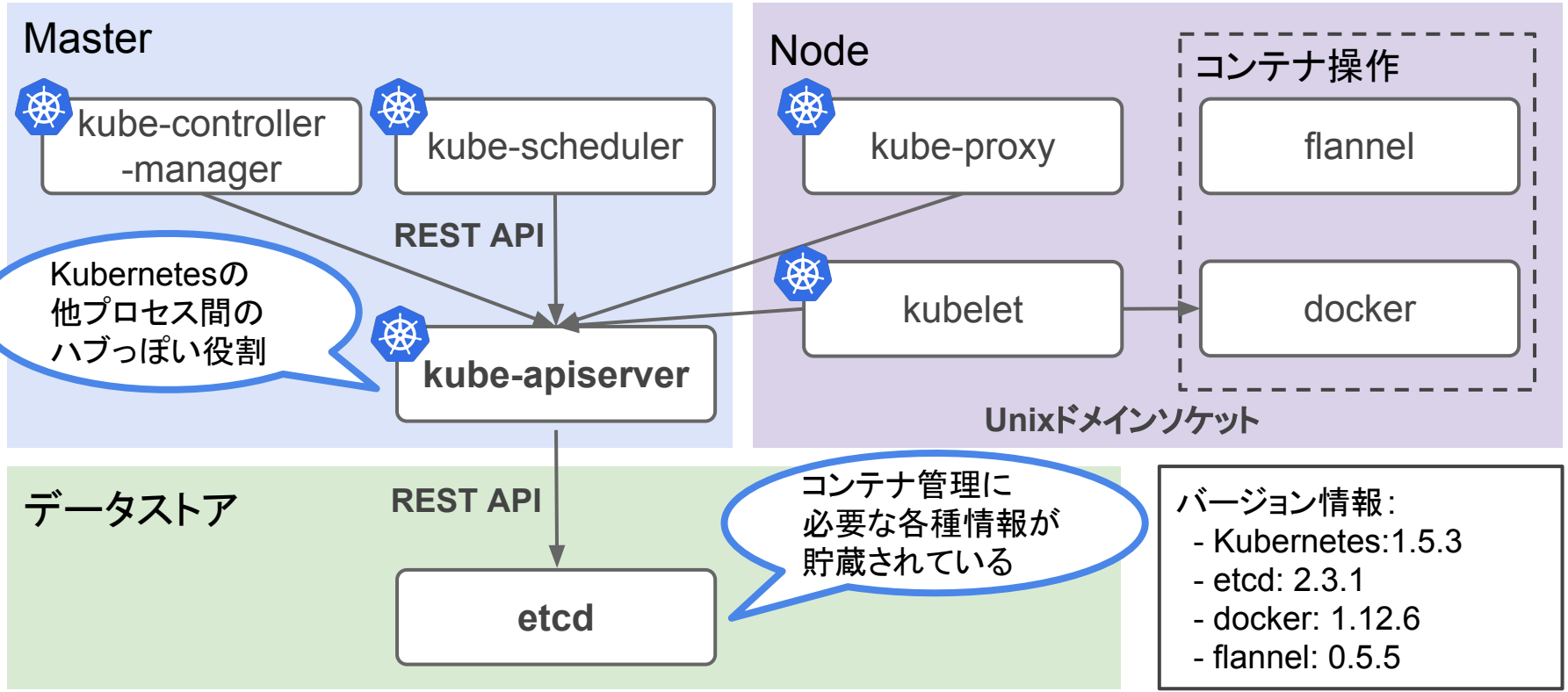
STEP2: Kubernetesクラスタ上のOpenStackの設計

- Kubernetesのリソース作成

STEP3: 実際に環境構築

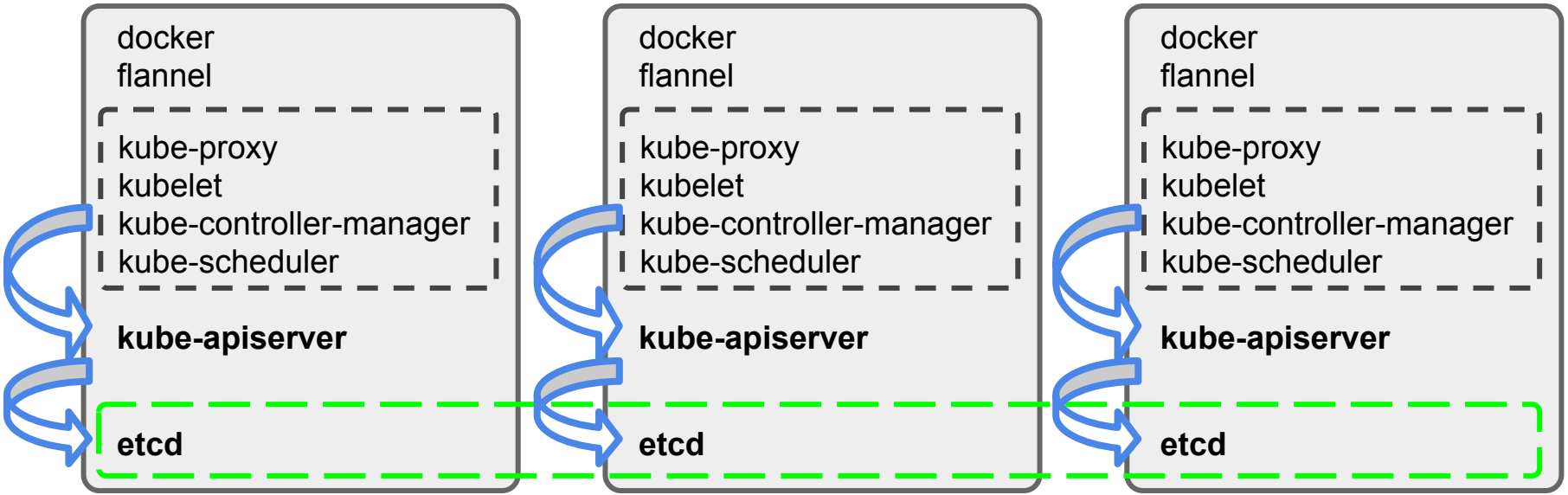
- Kubernetes環境構築/OpenStackデプロイ/構築試験

Kubernetesを構成するプロセス群



今回のKubernetesクラスタ設計

- Kubernetesの各プロセスはローカルのkube-apiserverを参照
- kube-apiserverはローカルのetcdを参照, etcdは3ノードでクラスタ化



OpenStack on Kubernetes導入までの道のり

STEP1: Kubernetesクラスタの設計

- Kubernetesのプロセス配置と冗長化の設計

STEP2: Kubernetesクラスタ上のOpenStackの設計

- Kubernetesのリソース作成

STEP3: 実際に環境構築

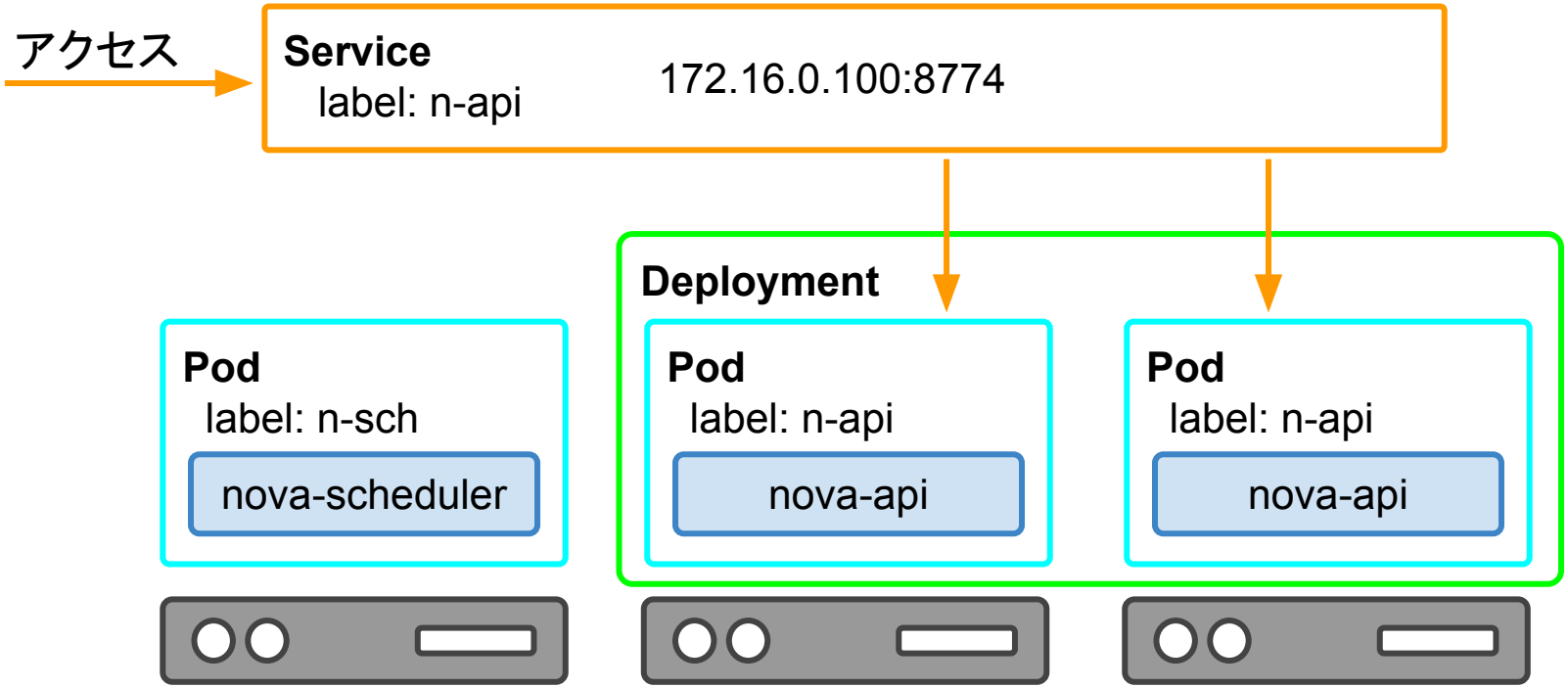
- Kubernetes環境構築/OpenStackデプロイ/構築試験

Kubernetesの用語説明

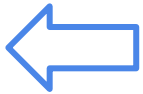
Kubernetesの仕組みの中でも基本的なものを利用してみた

- Pod
- Deployment
- Service

Kubernetesの用語説明



OpenStackデプロイのための準備物

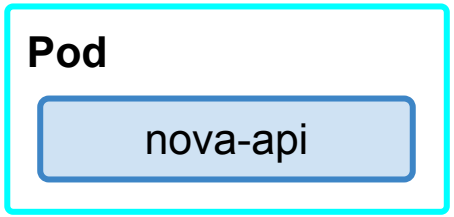
- Dockerfile
  - Kubernetes化するOpenStackプロセス分作成
- Pod/DeploymentとServiceの設定  今回の説明範囲
  - yamlファイルを作成
- OpenStackの各種設定ファイル

Pod/Deploymentの利用

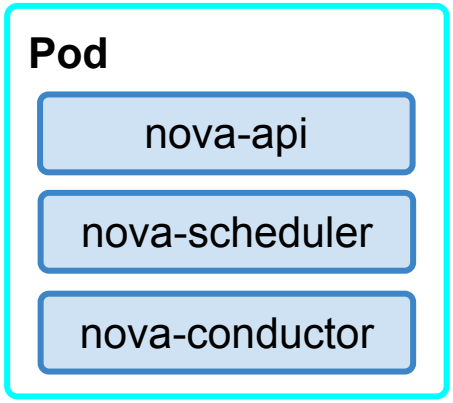
どのプロセスをPodとしてまとめればいい?

今回の選択

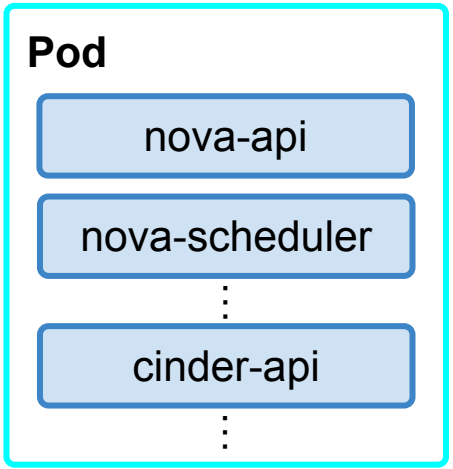
- 1Pod - 1プロセス



- 1Pod - 1コンポーネント



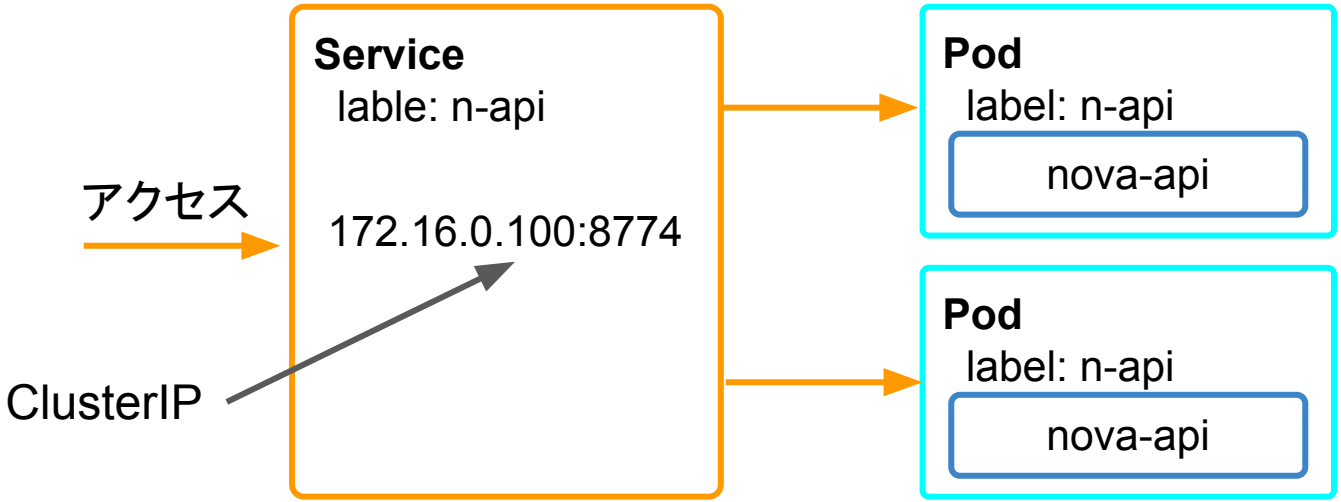
- 1Pod - 全プロセス



高	増設/減設の柔軟性	低
多	設定ファイルの量	少

Serviceの利用

- hoge-apiプロセスへの負荷分散に利用できる

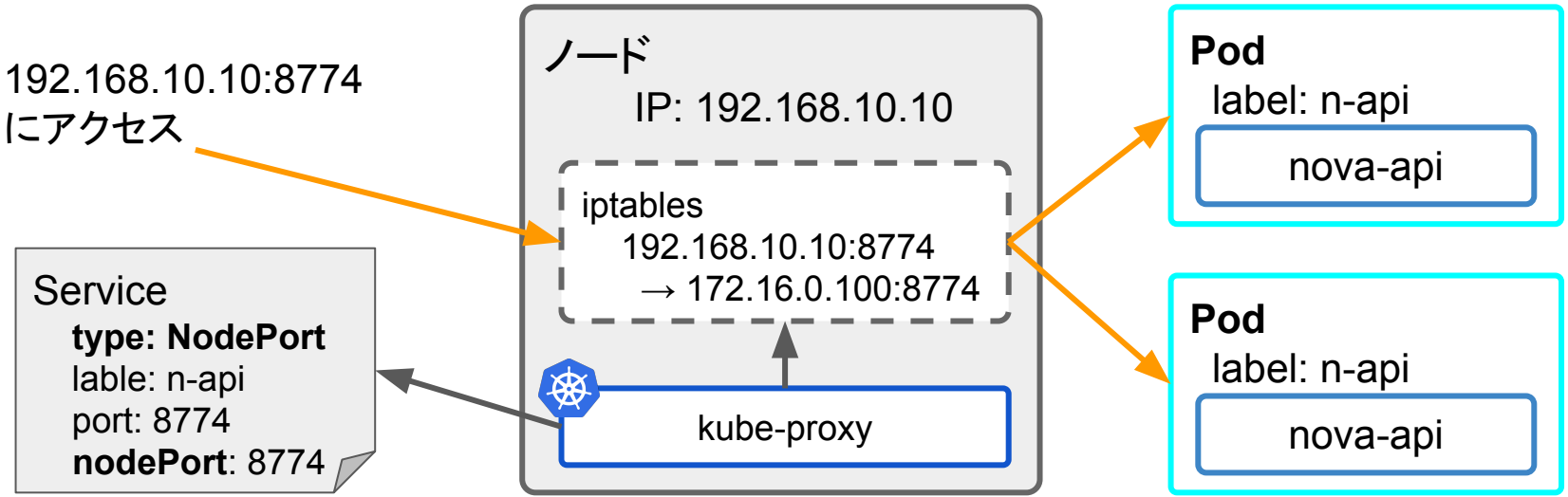


しかし, デフォルトの設定(ClusterIP利用)だとKubernetesクラスタの外部からアクセスできない!

Serviceの設計ポイント

外部からServiceのエンドポイントにアクセス可能にするためには??

- **NodePortタイプのServiceを利用**
  - クラスタを構成するノードのIPとportを利用する



OpenStack on Kubernetes導入までの道のり

STEP1: Kubernetesクラスタの設計

- Kubernetesのプロセス配置設計, 冗長化設計

STEP2: Kubernetesクラスタ上のOpenStackの設計

- Kubernetesのリソース作成

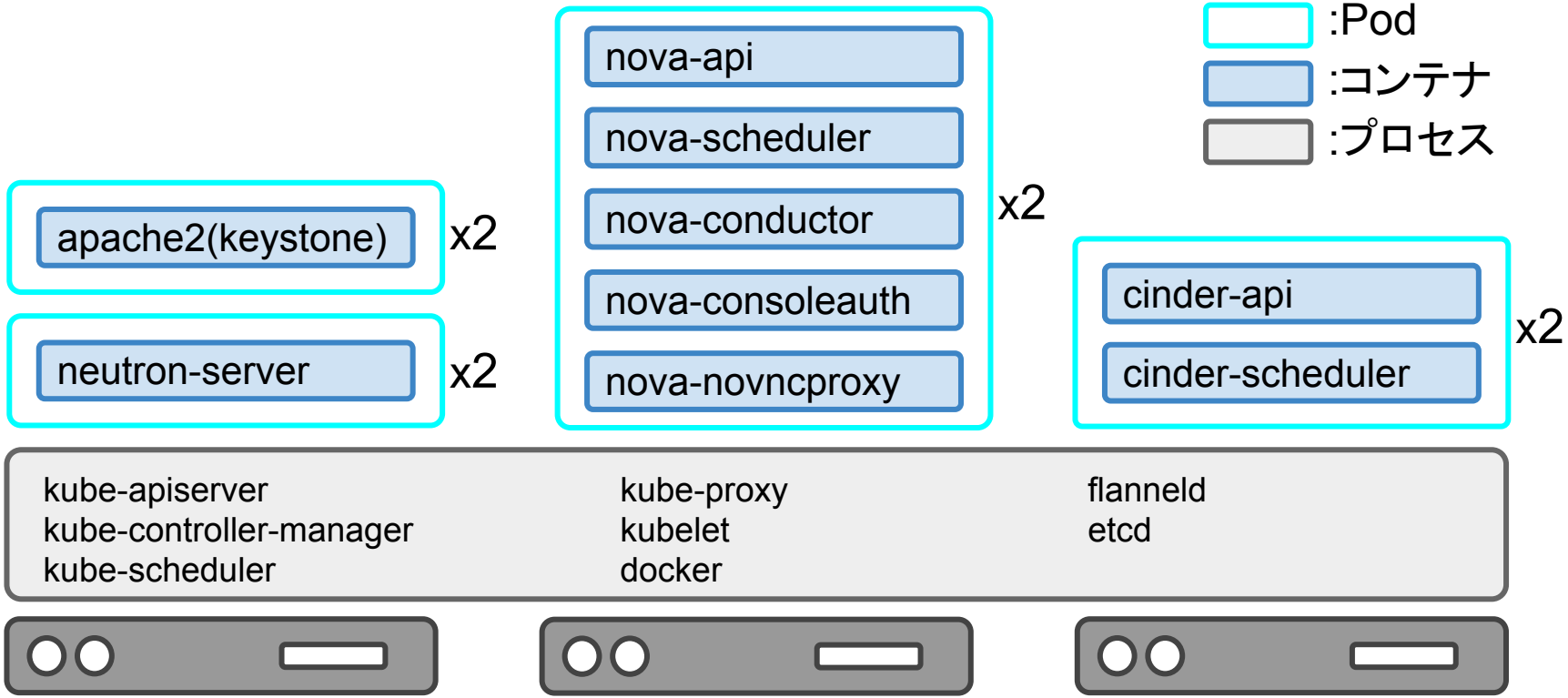
STEP3: 実際に環境構築

- Kubernetes環境構築, OpenStackデプロイ, 構築試験

環境構築の流れ

- Kubernetesクラスタ構築
- OpenStackの設定ファイル設置
- Kubernetes上にOpenStackをデプロイ
  - コンテナイメージのビルド
  - DeploymentとServiceの作成
- 構築試験
  - OpenStack APIへのアクセスが通ることを確認
  - Masakariにパッチを投げて正常にCI試験が実行されることを確認

構築したOpenStack on Kubernetes



結局, 導入してみてどうなった?

- Controllerノードの標準構成(相当)をKubernetesのみで実現できた
  - 従来のHAproxyやPacemakerの役割をKubernetesが担う
  - 系全体としてはHAproxyとPacemakerに加えてKubernetes管理が増える
  - Kubernetesの学習コストはもちろん必要
- 結局, 構築は楽になったか？
  - HAproxyやPacemakerの設定を実装するのに比較すると楽になった
    - DeploymentやServiceのyamlファイルの作成は
      HAproxyやPacemakerのconfファイル作成より容易

0から始めるOpenStack on Kubernetesの運用

運用を始めるために

ユーザからの
問い合わせ対応

機器、サービスの
状態監視

説明対象

トラブル時の
解析、復旧

必要に応じた
メンテナンス

OpenStack on Kubernetes監視までの道のり

STEP1: 監視項目の決定

- 故障を早期検知、異変を早期察知するための項目を精査

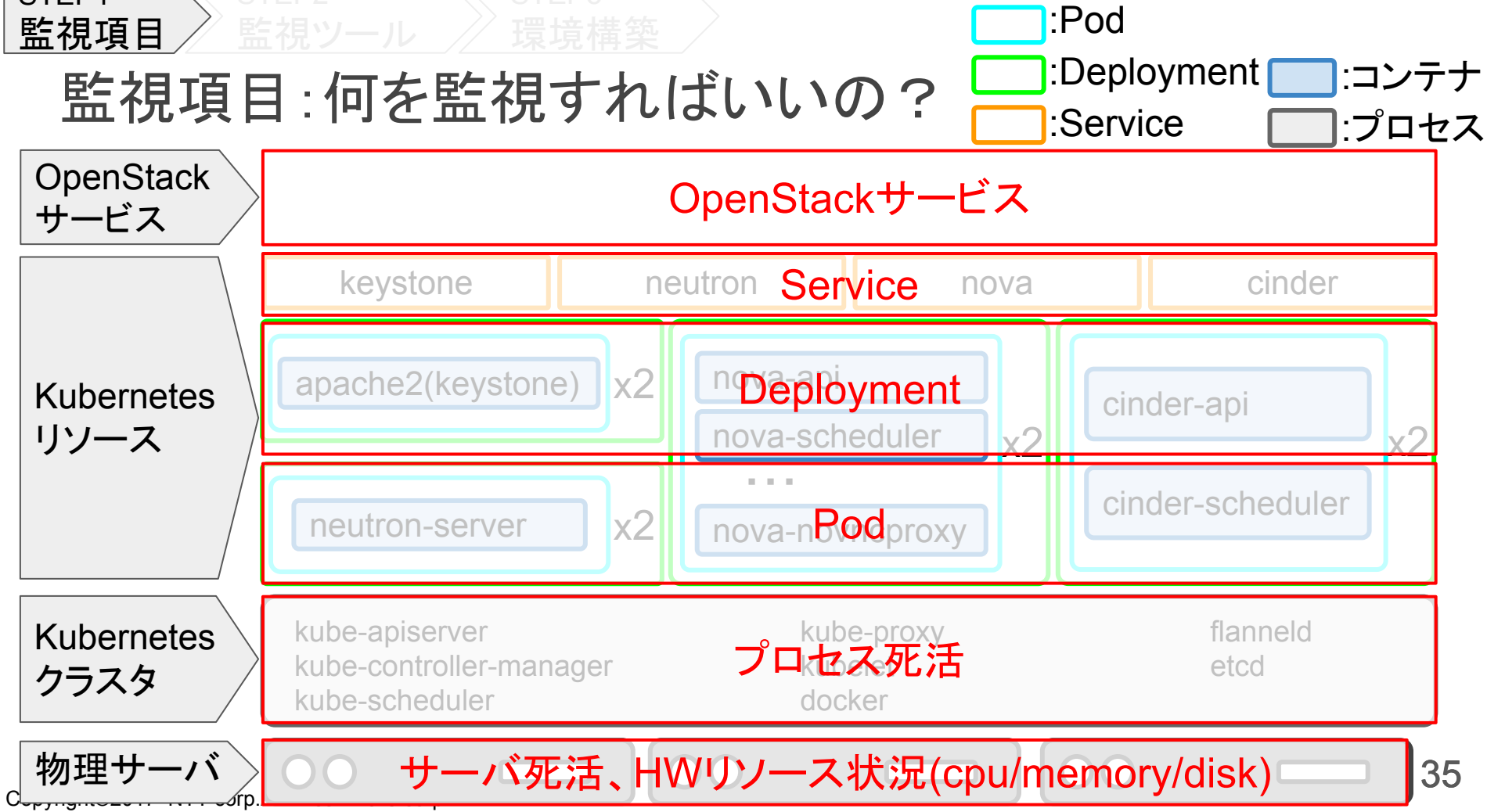
STEP2: 監視ツールの設計

- 監視項目をカバーする監視ツール設定を設計

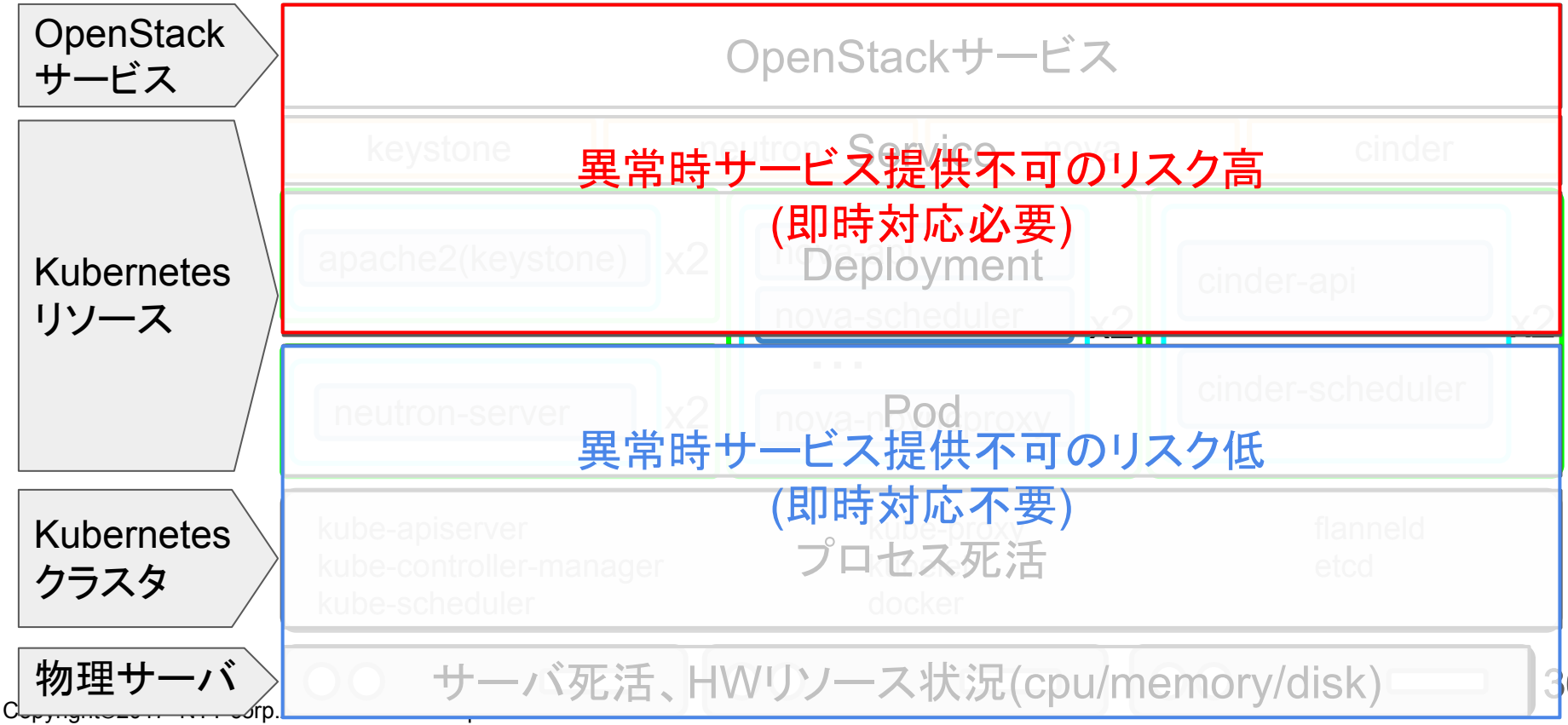
STEP3: 実際に環境構築

- 監視環境を構築

監視項目：何を監視すればいいの？



監視レベル：何が重要になるの？



監視ツール：何を使えばいいの？

Kubernetesの監視に特別なツールは必須でない

→OSSの**Zabbix**で簡単に実現可能

ZABBIX

提供層	監視項目	Zabbixでの実現
サービス	OpenStackサービス状態	従来監視と同一
Kubernetes リソース	Service状態	従来監視と異なる
	Deployment状態	
	Pod状態	
Kubernetes クラスタ	プロセス死活	従来監視と同様
物理サーバ	サーバ死活、HWリソース	従来監視と同一

外部スクリプトで監視って、複雑なんでしょ？

いいえ、たった数十行のshellでできます

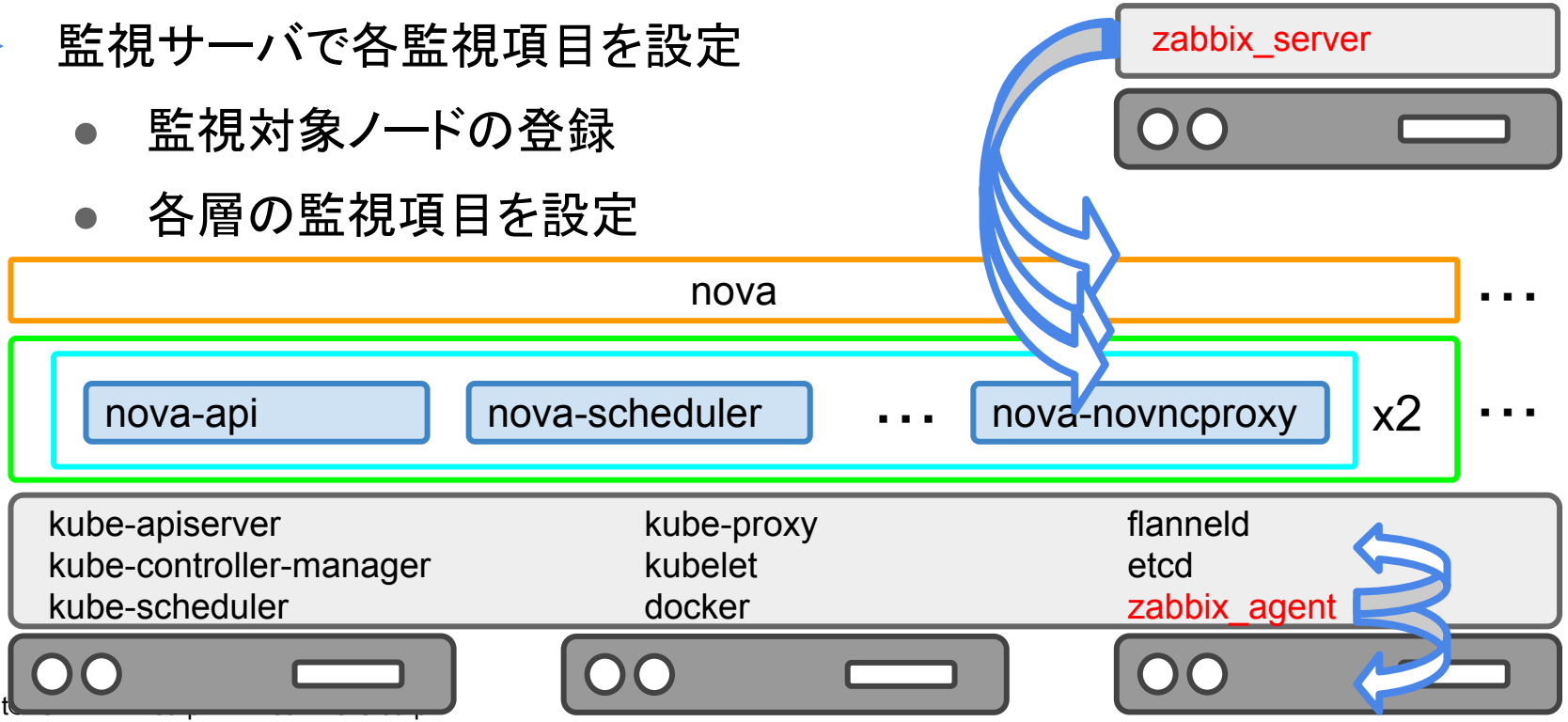
- KubernetesのREST APIにHTTPアクセス
- レスポンスjsonから該当リソースの状態を抽出し、状態を出力するだけ

PODリソースの状態確認shellサンプル(一部抜粋)

```
1 PODS_DATA=`curl -s $endpoint/api/v1/pods`
2 POD_LIST=`echo $PODS_DATA | jq '.items[].metadata.name'`
3 for pod_name in $POD_LIST
4 do
5     STATUS=`echo $PODS_DATA | \
6                 jq '.items[] | select(.metadata.name == '$pod_name')' | jq '.status.phase'`
7     echo $pod_name,$STATUS
8 done
```

環境構築：構築の流れ

- 監視対象ノードにzabbix_agentを導入
- 監視サーバで各監視項目を設定
  - 監視対象ノードの登録
  - 各層の監視項目を設定



「OpenStack」on Kubernetes監視のポイントは？

- 従来と同様の方法で監視を実現可能
- KubernetesKubernetes化したOpenStackプロセスのプロセス監視は不要
  (コンテナ内のプロセス死活はPOD状態に反映)
- Kubernetesクラスタのノード監視は一律の監視設定で良い
- 事前の検証は必要

結局，運用してみて楽になった？

- まだ分からない・・・
  - Kubernetes化の後、故障が発生していない…
  - Kubernetes化の後、増減設やupdateのメンテナンスが発生していない…

0から始めたまとめ

まとめ

結果:

- Kubernetesを使ってOpenStackを構築してみて「楽になった」か？
  - 楽になった
- Kubernetesを使ってOpenStackを運用してみて「楽になった」か？
  - まだ分からない・・・

考察:

- どんな人(環境)に向いている？
  - プロセス配置まで厳密に管理したい運用者には, 向いていないかも... (大規模チームでの運用?)
  - プロセス配置の管理を気にしない運用者には, 向いているかも… (少数精鋭チームでの運用者?)

ご清聴ありがとうございました

参考

OpenStackに適用する際の注意点

- 追加で必要な運用作業も発生
  - 再deploy発生で、コンテナ上OpenStackプロセスのホスト名が変わる
    →定期的にNovaなどのDBから不要なホスト名を削除する作業が発生

```
openstack@openstack:~$ openstack compute service list
+----+------------------+-----------------------------------+---------+
| ID | Binary           | Host                              | ~snip~ |
+----+------------------+-----------------------------------+---------+
| 65 | nova-scheduler   | nova-deployment-3169709093-5r5k2  | ~snip~ |
| 66 | nova-conductor   | nova-deployment-3169709093-tjll5  | ~snip~ |
| 67 | nova-consoleauth | nova-deployment-3169709093-chqw8  | ~snip~ |
~snip~
```

PodごとにKubernetesが付与

Dockerfile ~ Keystone

- **keystone**

```
FROM ubuntu:16.04
RUN apt update && apt install -y software-properties-common \
       && add-apt-repository cloud-archive:newton
RUN apt update && apt install -y keystone apache2 libapache2-mod-wsgi
ENV APACHE_RUN_USER=www-data
ENV APACHE_RUN_GROUP=www-data
ENV APACHE_PID_FILE=/var/run/apache2/apache2.pid
ENV APACHE_RUN_DIR=/var/run/apache2
ENV APACHE_LOCK_DIR=/var/lock/apache2
ENV APACHE_LOG_DIR=/var/log/apache2
EXPOSE 5000 35357
CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
```

Dockerfile ~ Nova 1/2

- **nova-api**

```
FROM ubuntu:16.04
RUN apt update && apt install -y software-properties-common \
       && add-apt-repository cloud-archive:newton
RUN apt update && apt install -y nova-api
EXPOSE 8774
CMD ["/usr/bin/nova-api", "--config-file", "/etc/nova/nova.conf", "--log-file", "/var/log/nova/nova-api.log"]
```

- **nova-scheduler**

```
FROM ubuntu:16.04
RUN apt update && apt install -y software-properties-common \
       && add-apt-repository cloud-archive:newton
RUN apt update && apt install -y nova-scheduler
CMD ["/usr/bin/nova-scheduler", "--config-file", "/etc/nova/nova.conf", "--log-file", "/var/log/nova/nova-scheduler.log"]
```

- **nova-conductor**

```
FROM ubuntu:16.04
RUN apt update && apt install -y software-properties-common \
       && add-apt-repository cloud-archive:newton
RUN apt update && apt install -y nova-conductor
CMD ["/usr/bin/nova-conductor", "--config-file", "/etc/nova/nova.conf", "--log-file", "/var/log/nova/nova-conductor.log"]
```

Dockerfile ~ Nova 2/2

- **nova-novcnproxy**

```
FROM ubuntu:16.04
RUN apt update && apt install -y software-properties-common \
       && add-apt-repository cloud-archive:newton
RUN apt update && apt install -y nova-novncproxy
EXPOSE 6080
CMD ["/usr/bin/nova-novncproxy", "--config-file", "/etc/nova/nova.conf", "--log-file", "/var/log/nova/nova-novncproxy.log"]
```

- **nova-consoleauth**

```
FROM ubuntu:16.04
RUN apt update && apt install -y software-properties-common \
       && add-apt-repository cloud-archive:newton
RUN apt update && apt install -y nova-consoleauth
CMD ["/usr/bin/nova-consoleauth", "--config-file", "/etc/nova/nova.conf", "--log-file", "/var/log/nova/nova-consoleauth.log"]
```

Dockerfile ~ Cinder

- **cinder-api**

```
FROM ubuntu:16.04
RUN apt update && apt install -y software-properties-common \
       && add-apt-repository cloud-archive:newton
RUN apt update && apt install -y cinder-api python-memcache
EXPOSE 8776
CMD ["/usr/bin/cinder-api", "--config-file", "/etc/cinder/cinder.conf", "--log-file", "/var/log/cinder/cinder-api.log"]
```

- **cinder-scheduler**

```
FROM ubuntu:16.04
RUN apt update && apt install -y software-properties-common \
       && add-apt-repository cloud-archive:newton
RUN apt update && apt install -y cinder-scheduler
CMD ["/usr/bin/cinder-scheduler", "--config-file", "/etc/cinder/cinder.conf", "--log-file",
"/var/log/cinder/cinder-scheduler.log"]
```

Dockerfile ~ Neutron

- **neutron-server**

```
FROM ubuntu:16.04
RUN apt update && apt install -y software-properties-common \
       && add-apt-repository cloud-archive:newton
RUN apt update && apt install -y neutron-server
EXPOSE 9696
CMD ["/usr/bin/neutron-server", "--config-file", "/etc/neutron/neutron.conf", "--config-file",
"/etc/neutron/plugins/ml2/ml2_conf.ini", "--log-file", "/var/log/neutron/neutron-server.log"]
```

Deployment ~ Keystone

- **keystone-deployment.yaml**

```
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: keystone-deployment
spec:
  replicas: 2
  template:
        metadata:
        labels:
        app: keystone
        spec:
        containers:
        - name: keystone
        image: <your repo IP>:<port>/keystone:newton
        volumeMounts:
        - name: keystone
        mountPath: /etc/keystone
        - name: apache2
        mountPath: /etc/apache2
        ports:
        - containerPort: 5000
        name: public
        - containerPort: 35357
        name: admin
        volumes:
        - name: keystone
```

```
        hostPath:
        path: /etc/k8s/keystone
        - name: apache2
        hostPath:
        path: /etc/k8s/apache2
```

Deployment ~ Nova

- **nova-deployment.yaml**

```
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nova-deployment
spec:
  replicas: 2
  template:
        metadata:
        labels:
        app: nova
        spec:
        hostNetwork: true
        containers:
        - name: nova-api
        image: <your repo IP>:<port>/nova-api:newton
        volumeMounts:
        - name: nova
        mountPath: /etc/nova
        ports:
        - containerPort: 8774
        securityContext:
        privileged: true
        - name: nova-scheduler
        image: <your repo IP>:<port>/nova-scheduler:newton
        volumeMounts:
        - name: nova
        mountPath: /etc/nova
        - name: nova-conductor
        image: <your repo IP>:<port>/nova-conductor:newton
        volumeMounts:
        - name: nova
        mountPath: /etc/nova
        - name: nova-novncproxy
        image: <your repo IP>:<port>/nova-novncproxy:newton
        volumeMounts:
        - name: nova
        mountPath: /etc/nova
        ports:
        - containerPort: 6080
        - name: nova-consoleauth
        image: <your repo
IP>:<port>/nova-consoleauth:newton
        volumeMounts:
        - name: nova
        mountPath: /etc/nova
        volumes:
        - name: nova
        hostPath:
        path: /etc/k8s/nova
```

Deployment ~ Cinder

- **cinder-deployment.yaml**

```
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: cinder-deployment
spec:
  replicas: 2
  template:
        metadata:
        labels:
        app: cinder
        spec:
        containers:
        - name: cinder-api
        image: <your repo IP>:<port>/cinder-api:newton
        volumeMounts:
        - name: cinder
        mountPath: /etc/cinder
        ports:
        - containerPort: 8776
        - name: cinder-scheduler
        image: <your repo
IP>:<port>/cinder-scheduler:newton
        volumeMounts:
```

```
        - name: cinder
        mountPath: /etc/cinder
        volumes:
        - name: cinder
        hostPath:
        path: /etc/k8s/cinder
```

Deployment ~ Neutron

- **neutron-deployment.yaml**

```
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: neutron-deployment
spec:
  replicas: 2
  template:
        metadata:
        labels:
        app: neutron
        spec:
        containers:
        - name: neutron-server
        image: <your repo IP>:<port>/neutron-server:newton
        volumeMounts:
        - name: neutron
        mountPath: /etc/neutron
        ports:
        - containerPort: 9696
        volumes:
        - name: neutron
        hostPath:
        path: /etc/k8s/neutron
```

Service ~ Keystone / Nova

- **keystone-service.yaml**

```
---
apiVersion: v1
kind: Service
metadata:
  name: keystone-service
spec:
  type: NodePort
  ports:
  - name: public
        port: 5000
        targetPort: 5000
        nodePort: 5000
        protocol: TCP
  - name: admin
        port: 35357
        targetPort: 35357
        nodePort: 35357
        protocol: TCP
  selector:
        app: keystone
```

- **nova-service.yaml**

```
---
apiVersion: v1
kind: Service
metadata:
  name: nova-service
spec:
  type: NodePort
  ports:
  - name: nova-api
        port: 8774
        targetPort: 8774
        nodePort: 8774
        protocol: TCP
  - name: nova-novncproxy
        port: 6080
        targetPort: 6080
        nodePort: 6080
        protocol: TCP
  selector:
        app: nova
```

Service ~ Cinder / Neutron

- **cinder-service.yaml**

```
---
apiVersion: v1
kind: Service
metadata:
  name: cinder-service
spec:
  type: NodePort
  ports:
  - name: cinder-api
       port: 8776
       targetPort: 8776
       nodePort: 8776
       protocol: TCP
  selector:
       app: cinder
```

- **neutron-service.yaml**

```
---
apiVersion: v1
kind: Service
metadata:
  name: neutron-service
spec:
  type: NodePort
  ports:
  - name: neutron-server
       port: 9696
       targetPort: 9696
       nodePort: 9696
       protocol: TCP
  selector:
       app: neutron
```

Kubernetesリソース監視スクリプト ~ Pod

```
#!/bin/bash

# Set args.
ENDPOINT=$1

# Get Pod json data.
PODS_DATA=`curl -s $ENDPOINT/api/v1/pods`

if [ -z "$PODS_DATA" ]
then
        echo "ERROR: Can't access to Kubernetes API endpoint."
        exit 2
fi

# Get Pod name list
POD_LIST=`echo $PODS_DATA | jq '.items[].metadata.name'`

# Get Pod status
for pod_name in $POD_LIST
do
        STATUS=`echo $PODS_DATA | jq '.items[] | select(.metadata.name == '$pod_name')' | jq '.status.phase'`
        echo $pod_name,$STATUS
done
```

Kubernetesリソース監視スクリプト ~ Deployment

```
#!/bin/bash

# Set args.
ENDPOINT=$1
DEPLOYMENT_NAME=$2
THRESHOLD=$3

# Get Deployment json data.
DEPLOYMENT_DATA=`curl -s
$ENDPOINT/apis/extensions/v1beta1/deployments`

if [ -z "$DEPLOYMENT_DATA" ]
then
        echo "ERROR: Can't access to Kubernetes API
endpoint."
        exit 2
fi

# Get Deployment status.
AVAILABLE_REPLICAS=`echo $DEPLOYMENT_DATA | jq '.items[] |
select(.metadata.name == "'$DEPLOYMENT_NAME'")' | jq -r
'.status.availableReplicas'`
```

```
# Check Deployment status.
STATUS=""
if test $AVAILABLE_REPLICAS -eq 0
then
        STATUS=0 # Error: No Pods are available.
elif test $AVAILABLE_REPLICAS -le $THRESHOLD
then
        STATUS=1 # Warning: Available Pods are less than or
equal to threshold.
else
        STATUS=2 # Healthy: Available Pods are more than
threshold.
fi

# Return Deployment status.
echo $STATUS
```

Kubernetesリソース監視スクリプト ~ Service

```
#!/bin/bash

# Set args.
ENDPOINT=$3
SERVICE_NAME=$1
THRESHOLD=$2

# Get Service json data.
SERVICE_DATA=`curl -s
$ENDPOINT/api/v1/namespaces/default/endpoints/$SERVICE_NAME
`

if [ -z "$SERVICE_DATA" ]
then
        echo "ERROR: Can't access to Kubernetes API
endpoint."
        exit 2
fi

# Get Available Service Endpoints.
AVAILABLE_ENDPOINTS=`echo $SERVICE_DATA | jq
'.subsets[].addresses[].targetRef.name' | wc -l`
```

```
# Check Service status.
STATUS=""
if test $AVAILABLE_ENDPOINTS -eq 0
then
        STATUS=0 # Error: No Service Endpoints are
available.
elif test $AVAILABLE_ENDPOINTS -le $THRESHOLD
then
        STATUS=1 # Warning: Available Service Endpoints are
less than or equal to threshold.
else
        STATUS=2 # Healthy: Available Service Endpoints are
more than threshold.
fi

# Return Service status.
echo $STATUS
```