

Cap. 5 - Kubernetes - 4 Aulas

Objetivos

1. Entender os conceitos básicos sobre Contêineres e seus Orquestradores.

Pré-requisitos:

1. Terminar o capítulo anterior (Cloud Outro)
2. Realizar a leitura sobre o Kubernetes. [<https://kubernetes.io/docs/concepts/>].
3. Realizar a leitura sobre o Docker. [<https://docs.docker.com/engine/docker-overview/>].

Branch

No capítulo 3, foi realizada a implantação do Openstack no hardware disponível. Além de VMs, o Openstack consegue também manipular containers, tal qual o MaaS e o Juju. Contudo o grande problema é que não basta apenas criar e destruir um container, é preciso administrar uma complexa arquitetura, monitorando e controlando um conjunto de containers que representam uma implantação de um sistema.

Perguntas preliminares:

1. O que é um *Container*? Qual a principal diferença entre *Container* e VM?
2. O que é e o que faz Kubernetes? Quais as vantagens e desvantagens em relação ao Openstack.

3. Baseando-se nos capítulos anteriores, desenhe uma arquitetura que represente desde o Hardware até o Kubernetes rodando em cima do Openstack.

Docker

- Realizar a leitura: <https://docs.docker.com/get-started/>
- Criar uma instância pequena (1Gb de RAM) com IP público
- Vamos criar um *WebServer* em NodeJS
 - Criar o arquivo `~/hello-node/server.js`

```
var http = require('http');

var handleRequest = function(request, response) {
  console.log('Received request for URL: ' + request.url);
  response.writeHead(200);
  response.end('Hello World!');
};

var www = http.createServer(handleRequest);
www.listen(8080);
```

- **Testar** o *WebServer* rodando no terminal e abrindo no navegador.
- Instalar o Docker
 - `$ sudo apt install docker.io`
- Criar um arquivo `~/hello-node/Dockerfile` que dará origem ao *container*

```
FROM node:6.9.2
EXPOSE 8080
COPY server.js .
CMD node server.js
```

- Montar a imagem local
 - \$ docker build -t hello-node:v1 ./hello-node/
- Corrigir o problema
- Criar individualmente um login na <https://hub.docker.com/>
- Fazer o login no Docker
- Aplicar a *tag* na imagem e dar o *push* do Docker Hub
- Para mais detalhes
 - \$ docker --help

1. O que é Docker Swarm?

2. Quais as diferenças entre Docker e LXD?

3. O que é Docker Registry?

Kubernetes aka k8s

- Realizar a leitura: <https://kubernetes.io/docs/tutorials/kubernetes-basics/>
- **Atenção:** Não é necessário executar o tutorial
- Verificar a instalação do Kubernetes. Rever o R3.
- Instalar *kubectl* no cliente via snap.
- Verificar a configuração do Kubernetes via *kubectl*
 - \$ kubectl cluster-info
- Caso tenha problemas, ver Kubernetes-core no Juju Charms Store
- Configurar o *kubectl proxy* para utilização externa na porta 8080.

- Acessar o *Dashboard*: [ip público do cliente]:8080/ui

1. Qual o comando completo utilizado no *proxy*?

Fazendo um Deploy

- Fazer o *deploy* da imagem docker criada na sessão anterior
- \$ kubectl run hello-node --image=[user]/hello-node:v1 --port=8080
- Verificar o *deploy status* no *Dashboard*
- Expandir o *deploy* para 5 réplicas do *pod*

1. Explique o que é *pod* e *replicaset*?

Acessando o Deploy

- Criar um serviço que irá expor o *deploy*
 - Alternativa I - *Load Balancer* (AWS, Azure e GCE)¹
 - \$ kubectl expose deployment hello-node --type=LoadBalancer
 - Alternativa II - *ClusterIP* com *Ingress*
 - \$ kubectl expose deployment hello-node --type=ClusterIP
 - Criar o arquivo ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: hello-node-ingress
spec:
  rules:
  - host: hello-node.[ip do kubernetes-worker].xip.io
    http:
      paths:
      - path: /
        backend:
          serviceName: hello-node
          servicePort: 8080
```

- \$ kubectl create -f ingress.yaml
- Acompanhar o andamento no *Dashboard*
- Testar o serviço acessando o *deploy* no navegador

1. Qual a URL utilizada?

2. O que faz <http://xip.io?>

¹ É possível habilitar o serviço de *LoadBalancer* em algumas distribuições do Openstack.

Alterando o *deploy*

- Alterar o arquivo server.js
- Montar uma nova imagem com *tag* v2 e faça um *push*
- Fazer um *deploy* de um *pod* de testes
- Testar as alterações
- Fazer a implantação imediata em produção:
 - \$ kubectl set image deployment/hello-node hello-node=[user]/hello-node:v2

1. Você considera esse modelo de implantação imediata bom ou ruim? Justifique.

Para saber mais

1. <https://kubernetes.io/docs/tutorials/>
2. <https://kubernetes.io/docs/tasks/>
3. <https://jujucharms.com/kubernetes-core/>

Questões Complementares

1. Qual a diferença entre serviços *stateless* e *statefull*?
2. Como os recursos são distribuídos no uso de containers?
3. Descreva o princípio de *Horizontal Pod Autoscaling*.
4. É possível rodar o Openstack sobre Kubernetes? e Kubernetes sobre Openstack?
Assistir: <https://youtu.be/GsZMagSmQjY>.

Concluindo

1. O que é um *Container*? Qual a principal diferença entre *Container* e VM?

2. O que é e o que faz Kubernetes? Quais as vantagens e desvantagens em relação ao Openstack.

3. Baseando-se nos capítulos anteriores, desenhe uma arquitetura que represente desde o Hardware até o K8s rodando em cima do Openstack.

Conclusão: Descreva detalhadamente como seria uma arquitetura de *Continuous Integration/Continuous Delivery* (CI/CD) utilizando Kubernetes.