Insper

Computação em Nuvem

Cap. 5 - Container Orchestration - 4 Aulas

Raul Ikeda - rauligs@insper.edu.br

Grupo:

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.

Docker

- Realizar a leitura: https://docs.docker.com/get-started/
- Criar uma instância pequena (1Gb de RAM) com IP público no Openstack
- Vamos criar novamente um *WebServer* em NodeJS
  - Criar o arquivo ~/**hello-node/server.js**

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

var handleRequest = function(request, response) {
  console.log('Received request for URL: ' + request.url);
  response.writeHead(200);
  response.end('{ "host": ' + os.hostname() + ', "message": "Hello World!"}');
};
var www = http.createServer(handleRequest);
www.listen(8080);
```

  - **Testar** o *WebServer* rodando localmente no terminal e abrindo no navegador.
- Instalar o Docker
  - $ sudo apt install docker.io
- Criar uma 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 de permissão sem usar *sudo*.

- 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 técnicas 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*.

4. Qual foi o processo para acessar o Dashboard?

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 no terminal: $ kubectl get all
- Verificar o *deploy status* no *Dashboard*
- Expandir o *deploy* para 5 réplicas do *pod*

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

Acessando o Deploy

- Criar um serviço que irá expor o *deploy*
- **Alternativa I** - *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

- **Alternativa II** - *NodePort*
  - $ kubectl expose deployment hello-node --type=NodePort
  - Verificar qual porta foi designada e liberar o security group do *worker*
  - $ juju run --unit kubernetes-worker/[# worker] "open-port [porta]"

- **Alternativa III** - *Load Balancer* (você instalou o Octavia)
  - $ kubectl expose deployment hello-node --type=LoadBalancer

- Acompanhar o andamento no terminal e no *Dashboard*
- Testar o serviço acessando o *deploy* no navegador

6. Qual você considera a melhor alternativa? Justifique.

7. O que faz http://xip.io?

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

8. 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. Como é criado um serviço *stateful* no Kubernetes?

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? Assistir: https://youtu.be/GsZMagSmQjY. Quais as vantagens dessa arquitetura?

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: Você desempenhou uma implantação em tempo real usando Kubernetes e DockerHub. Como seria uma arquitetura de *Continuous Integration/Continuous Delivery* (CI/CD) real utilizando Kubernetes?