Compass

Atividade Final Prática
Documentação

Versão 1.2

Histórico da Revisão

Data	Versão	Descrição	Autor
05/03/2022	1.0	Documentação inicial	Erico Ferreira V. Broock
07/03/2022	1.1	Integrando Jenkins com GitHub	Erico Ferreira V. Broock
10/03/2020	1.2	Criando ssh para Jenkins e pipeline	Erico Ferreira V. Broock

Índice Analítico

Documentação da Atividade Final Prática

1. Introdução

1.1 Finalidade

Este documento descreve como foi realizado a atividade prática de forma sequencial. O objetivo é ter uma definição onde pode-se replicar o projeto e detectar possíveis problemas.

1.2 Escopo

Instruções do projeto:

- Instalem uma Virtual Machine utilizando a S.O Oracle Linux - last stable version;

- Instalem o Docker/Kubernetes nessa máquina virtual;

- Instalem o Jenkins no Kubernetes dessa máquina virtual;

- Construa os manifests no Kubernetes para um site Wordpress e banco MySQL 5.6, faça o deploy manual de todos os manifests na namespace projetocompass, com exceção do manifest de deployment do Wordpress;

- Criem um pipeline no jenkins, onde ao disparar esse job, fará a criação de um site WordPress dentro do Kubernetes (dica: ele precisa interagir somente com o manifest de deployment do Wordpress);

- Validem o funcionamento do site Wordpress através do browser Chrome ou Firefox, através da URI projetofinal.compass.uol

Observação: Nenhuma outra distro Linux está autorizada para essa atividade, utilizem a Oracle Linux - last stable version

Importante: "Cada dupla deverá montar a documentação do que fizeram"

1.3 Referências

Obtenção de vagrantfile para iniciar a configuração: https://yum.oracle.com/boxes/

Obtenção do vagrant na versão 2.2.29, para Windows: https://www.vagrantup.com/

Obtenção do VirtualBox na versão 6.1.32, para Windows: https://www.virtualbox.org/wiki/Downloads

Instalação do Jenkins com arquivos yaml: https://www.jenkins.io/doc/book/installing/kubernetes/

Para configurar o iptables e acessar Jenkins: https://jensd.be/343/linux/forward-a-tcp-port-to-another-ip-or-port-using-nat-with-iptables

2. Visão Lógica

Esta sessão descreve os procedimentos adotados para a execução da atividade final prática.

2.1 Vagrantfile

O vagrantfile foi configurado conforme dados abaixo. Para ser utilizado, na pasta onde está o vagrantfile é preciso haver a pasta scripts com o arquivo docker.sh. Ainda, é necessário atualizar o *ip: "192.168.18.10"* para um ip existente em nova rede.

```
Vagrant.configure("2") do |config|

  config.vm.box = "oraclelinux/8"
  config.vm.box_url = "https://oracle.github.io/vagrant-
projects/boxes/oraclelinux/8.json"

  # Disksize
  config.disksize.size = '30GB'

  config.vm.network "forwarded_port", guest: 81, host: 81
  config.vm.network "forwarded_port", guest: 80, host: 80
  config.vm.network "forwarded_port", guest: 3306, host: 3306
  config.vm.network "forwarded_port", guest: 8080, host: 8080
  config.vm.network "forwarded_port", guest: 30000, host: 30000
  config.vm.network "forwarded_port", guest: 19999, host: 19999
  config.vm.network "forwarded_port", guest: 9001, host:9001

  config.vm.network "public_network", ip: "192.168.15.15"

  config.vm.provider "virtualbox" do |vb|

    vb.memory = "3548"
    vb.name = "compasso"
  end

# Extra
  config.vm.provision "shell",
    inline: "chmod +x /vagrant/scripts/*"
  config.vm.provision "shell",
  inline: "sudo /vagrant/scripts/docker.sh"
end
```

Script 1: Vagrantfile

2.2 Docker

Para instalação do Docker foi utilizado o seguinte script localizado na pasta scripts. O vagrant deve executá-lo na primeira execução do comando "vagrant up". O script em questão atualiza o linux, adiciona o repositório do Docker, baixa e executa a instalação do Docker e finaliza registrando o usuário vagrant para utilização do Docker.

Assim que a primeira vez que o comando “vagrant up” for executada com sucesso, é possível se conectar com “vagrant ssh” e executar o comando “docker ps” para verificar a instalação.

```
# Instalando o docker
echo "Iniciando a instalação do docker."
echo "Realizando update."
sudo yum update -y
echo "Instalando yum-utils."
sudo yum install -y yum-utils
echo "Inserindo caminho para repositório do docker."
sudo yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
sudo yum update -y
echo "Instalando o docker."
sudo yum -y install docker-ce docker-ce-cli containerd.io
sudo systemctl start docker
sudo systemctl enable docker
echo "Registrando o usuário vagrant para utilizar o docker."
sudo groupadd docker
sudo usermod -aG docker vagrant
newgrp docker
echo "Finalização da instalação do docker."
```

Script 2: Docker.sh

2.3 Kubectl e Minikube

Para instalação do Kubernetes foi adotado os seguintes scripts executados, após logar na vm, com os comandos:

[vagrant@localhost vagrant]$ sh ./scripts/kubectl.sh

[vagrant@localhost vagrant]$ sh ./scripts/minikube.sh

A pasta /vagrant é a pasta de compartilhamento de arquivos.

```
echo 'Baixando binário do kubectl.'
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

echo 'Validando binário baixado'
curl -LO "https://dl.k8s.io/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
echo "$(<kubectl.sha256)  kubectl" | sha256sum --check

echo 'Instalando o kubectl'
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

echo 'Mostrando versão instalada:'
kubectl version --client
```

Script 3: kubectl.sh

```
echo "Instalando o minikube."

curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-
amd64

sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

Script 4: minikube.sh

2.4 Jenkins

Para instalação do Jenkins no Kubernetes foram adotados os seguintes comandos no Minikube:

[vagrant@localhost vagrant]$ minikube start

[vagrant@localhost vagrant]$ kubectl create namespace jenkins

[vagrant@localhost vagrant]$ kubectl create -f jenkins-deploy.yaml

[vagrant@localhost vagrant]$ kubectl create -f jenkins-service.yaml

O comando “kubectl get svc -n jenkins” deve mostrar o serviço configurado para a porta 30000.

O comando “kubectl get pods -n jenkins” deve mostrar o pod criado. Algumas vezes demorou cerca de 7 minutos para o pod ficar com o status “running”.

O comando “curl 192.168.49.2:30000” deve retorna um código html, sinal de que o serviço do Jenkins está funcionando. O ip 192.168.49.2 foi gerado para o minikube.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkins
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jenkins
  template:
    metadata:
      labels:
        app: jenkins
    spec:
      containers:
      - name: jenkins
        image: jenkins/jenkins:lts-jdk11
        ports:
        - containerPort: 8080
        volumeMounts:
        - name: jenkins-home
          mountPath: /var/jenkins_home
      volumes:
      - name: jenkins-home
        emptyDir: { }
```

Script 5: jenkins-deploy.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: jenkins
spec:
  type: NodePort
  ports:
  - port: 8080
    #targetPort: 8080
    nodePort: 30000
  selector:
    app: jenkins
```

Script 6: jenkins-service.yaml

- Configurando as portas para o Jenkins ser acessado pelo navegador do computador local:

$ sudo iptables -t nat -A PREROUTING -p tcp --dport 30000 -j DNAT --to-destination 192.168.49.2:30000

$ sudo iptables -t nat -A POSTROUTING -p tcp -d 192.168.49.2 --dport 30000 -j SNAT --to-source 192.168.15.15

$ sudo iptables -A FORWARD -p tcp -d 192.168.49.2 --dport 30000 -j ACCEPT

$ sudo iptables-save|sudo tee /etc/sysconfig/iptables

Onde, 192.168.49.2 é o ip do minikube e 192.168.15.15 foi definido no vagrantfile. Assim é possível acessar o Jenkins pelo navegador utilizando a url: 192.168.15.15:30000.

Para recuperar a senha inicial do Jenkins é preciso verificar o nome do pod com “kubectl get pods -n jenkins” e executar “kubectl logs <pod_name> -n jenkins”.

Ao abrir o Jenkins, acessando por 192.168.15.15:30000, foi adotado instalação de plugins recomendados.

- Na sessão ‘Create First Admin User’ foram adotados os seguintes critérios:

Nome de usuário: adm

Senha: mestre

Nome completo: administrador

Email: ericobroock@gmail.com

- Na sessão ‘Instance Configuration’ foi deixada a url padrão.

2.5 Manifests para Wordpress e banco MySQL 5.6

Para a aplicação dos manifests deve-se ir para a pasta /vagrant/manifests e aplicar os seguintes comandos:

$ minikube addons enable ingress

$ kubectl create -f namespace.yaml

$ kubectl apply -f mysql-secret.yaml

$ kubectl apply -f storage-class.yaml

$ kubectl apply -f mysql-pv.yaml

$ kubectl apply -f wordpress-pv.yaml

$ kubectl apply -f mysql-pvc.yaml

$ kubectl apply -f wordpress-pvc.yaml

$ kubectl apply -f mysql-deploy.yaml

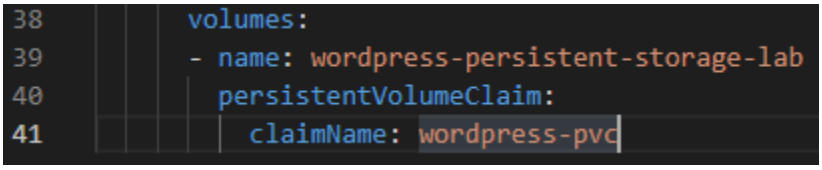
$ kubectl apply -f wordpress-deployment.yaml

$ kubectl apply -f ingress-mysql-wordpress.yaml

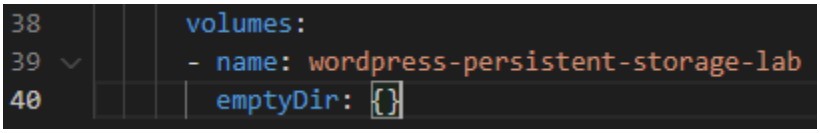
Com exceção do manifest wordpress-deployment.yaml, todos os demais são configurados na namespace ‘projetocompass’.

ATUALIZAÇÃO DEPOIS DA ENTREGA

Como houve erro de aplicação, foram desinstalados os arquivos: storage-class.yaml, mysql-pv.yaml, wordpress-pv.yaml, mysql-pvc.yaml e wordpress-pvc.yaml. E nos dois arquivos de deployment foram alterados os trechos finais que referenciam os volumes, a linha ‘persistentVolumeClaim’ e ‘claimName:’ foram substituídos por ‘emptyDir: {}



Script 7: wordpress-deployment.yaml anterior.



Script 8: wordpress-deployment.yaml atualizado.

Para teste foi executado os seguintes comandos:

$ minikube addons enable ingress

$ kubectl create -f namespace.yaml

$ kubectl apply -f mysql-secret.yaml

$ kubectl apply -f mysql-deploy.yaml

$ kubectl apply -f wordpress-deployment.yaml

$ kubectl apply -f ingress-mysql-wordpress.yaml

Editado o arquivos hosts no Windows:

- Executar o notepad como administrador e abrir arquivo: C:/Windows/System32/drivers/etc/hosts

- Inserir uma linha com a informação do ip da vm e url: 192.168.15.15 projetofinal.compass.uol

Tal como foi feito com a porta 30000, é preciso liberar a porta 80 com os comandos:

$ sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.49.2:80

$ sudo iptables -t nat -A POSTROUTING -p tcp -d 192.168.49.2 --dport 80 -j SNAT --to-source 192.168.15.15

$ sudo iptables -A FORWARD -p tcp -d 192.168.49.2 --dport 80 -j ACCEPT

Agora, acessando pelo navegador:

- Ao colocar o ip 192.168.15.15:80 – Abre a página Nginx Erro 404.

- Ao colocar a url projetofinal.compass.uol – Abre o Wordpress.

2.6 Comandos para gerar a chave ssh e estabelecer confiançao do pod com a vm

Para acessar o pod do Jenkins:

$ kubectl get pods -n jenkins

$ kubectl exec -it <pod-name> -n jenkins -- bash

jenkins@jenkins-789c9b6b84-j9lw6:/$ ssh-keygen -b 1024 -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (/var/jenkins_home/.ssh/id_rsa):
/var/jenkins_home/.ssh/jenkins

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /var/jenkins_home/.ssh/jenkins

Your public key has been saved in /var/jenkins_home/.ssh/jenkins.pub

$ Cat /var/jenkins_home/.ssh/jenkins.pub

Copiar a chave pública

$ sudo vi ~/.ssh/authorized_keys

Colar a chave pública abaixo de outra chave se houver.

Para testar a conexão basta acessar novamente o pod do Jenkins e realizar o comando:

$ ssh vagrant@192.168.15.15

O pod deverá ter acessado a vm.

Comando $ exit, para sair da vm, e novamente para sair do pod.

2.7 Plugins no Jenkins e criação da pipeline

Aqui no exemplo abaixo, o ip da vm é 192.168.18.18,

Instalar o plugin Kubernetes CLI.

Criar um job do tipo pipeline.

Para efeito de teste não há necessidade de configurar um trigger.

Em pipeline scripts, adotar:

```
pipeline {
  agent any
    stages {
      stage('Build') {
        steps {
          sh 'ssh vagrant@192.168.18.18 kubectl apply -f /vagrant/manifests/wordpress-
deployment.yaml -n projetocompass'
        }
    }
  }
}
```

Figura 1: Saída do console da pipeline criando o deployment do wordpress

2.8 Jenkins acessando o GitHub

No vagrant, comando:

$ ssh-keygen -t rsa -b 4096 -C "<seu-usuario>@gmail.com"

Assim cria-se o par de chaves ssh, considerando que foi aceita no padrão, sem alteração de nome e inserção de senha, executa-se:

$ cat ~/.ssh/id_rsa

A chave privada é impressa na tela. Copiamos toda a chave e acessamos o Jenkins.

No Jenkins, Gerenciar Jenkins – Manage Credentials – Add Global Credentials

ID – nome para indicar a credencial

Username – git

Private Key – Enter directly: colar a chave privada copiada anteriormente.

Save.

No vagrant, comando:

$ cat ~/.ssh/id_rsa.pub

Copiamos agora a chave pública.

No GitHub, Settings – SSH and GPG Keys – New SSH key. Dar um nome a chave e colar o código da chave pública

-- Primeiro Job no Jenkins:

Selecionar opção de versionamento de git e inserir a url ssh do repositório. Vai mostrar uma mensagem de erro e ao selecionar a chave criada anteriormente, a conexão deve ser estabelecida.