DOCUMENTAÇÃO DE INSTALAÇÃO DO CLUSTER

Utilizando Kops

- Baseado em -> https://aws.amazon.com/pt/blogs/compute/kubernetes-clusters-aws-kops/

1. Instalar Kops e Kubectl
2. Adicionar IEM na AWS com as seguintes permissões:

- AmazonEC2FullAccess
- AmazonRoute53FullAccess
- IAMFullAccess
- AmazonS3FullAccess
- AmazonVPCFullAccess

3. Criar um bucket no S3 para salvar estado do Cluster (nome = kubernetes-aws-io-marcelo)

- $ aws s3api create-bucket --bucket kubernetes-aws-io-marcelo
- $ aws s3api put-bucket-versioning --bucket kubernetes-aws-io-marcelo --versioning-configuration Status=Enabled
- Exportar variavel de ambiente para uso no Kops, $ export KOPS_STATE_STORE=s3://kubernetes-aws-io-marcelo

4. Configurar DNS

- Primeiramente, baixar o jq para rodar interpretar o JSON
- Criar a zona no route53 (nome = cluster.kubernetes-aws.io-marcelo)

```
$ ID=$(uuidgen) && \
aws route53 create-hosted-zone \
```

```
--name cluster.kubernetes-aws.io-marcelo \
--caller-reference $ID \
| jq .DelegationSet.NameServers
```

- O output deve ser o seguinte:

```
[
"ns-94.awsdns-11.com",
"ns-1962.awsdns-53.co.uk",
"ns-838.awsdns-40.net",
"ns-1107.awsdns-10.org"
]
```

-Não foi utilizado um domínio, e sim o conceito experimental de gossip-based cluster, para isso, mudar nome do cluster com .k8s.local no final.

5. Criar o cluster (nome = cluster.kubernetes-aws.io-marcelo)

```
$ kops create cluster \
--name cluster.kubernetes-aws.io-marcelo.k8s.local \
--zones us-west-2a \
--state s3://kubernetes-aws-io-marcelo \
--yes
```

- Para maiores configurações (numero de nodes, regiao, etc), olhar o documento base.

6. Deletar o cluster

- $ kops delete cluster --state=s3://kubernetes-aws-io-marcelo --yes
- Para criar novamente o cluster após deletar, começar a partir do "Criar Cluster"

Adicionando Dashboard no Kops

- Baseado em -> https://github.com/kubernetes/kops/blob/master/docs/addons.md

1. $ kubectl create -f https://raw.githubusercontent.com/kubernetes/kops/master/addons/kubernetes-dashboard/v1.7.1.yaml
2. Para acessar o dashboard, $ kubectl proxy --port=8080 &
3. Acesse pela url localhost:8080/ui

Exemplo de deploy de uma aplicação

- Imagem exemplo: marcelogdeandrade/node-web-app

1. $ kubectl run node-web-app --image=docker.io/marcelogdeandrade/node-web-app --port=8081
2. Para fazer um port-forward de uma porta local para a porta 8081 do pod (porta que o servidor está rodando)

- nome do pod = node-web-app-2024651897-7w2dj
- $ kubectl port-forward node-web-app-2024651897-7w2dj 8083:8081

3. Acesse o serviço pelo url localhost:8083

Configurando o Horizontal Auto Scaler

1. Primeiro, é necessario o uso do serviço Heapster para monitoramento do clustes:

- $ git clone https://github.com/kubernetes/heapster.git
- $ cd heapster
- $ kubectl create -f deploy/kube-config/influxdb/
- $ kubectl create -f deploy/kube-config/rbac/heapster-rbac.yaml

2. Dar deploy em uma aplicação com expose e definição de recursos da CPU

- $ kubectl run node-web-app --image=docker.io/marcelogdeandrade/node-web-app --port=8081 --requests=cpu=200m

3. Ativar autoscale no deployment desejado

- $ kubectl autoscale deployment node-web-app --cpu-percent=50 --min=1 --max=5
- Para verificar se tudo está funcionando, rode $ kubectl get hpa

```
NAME           REFERENCE                     TARGETS
MINPODS   MAXPODS   REPLICAS   AGE
node-web-app   Deployment/node-web-app   0% / 50%
1         5         0          3s
```

4. Dar deploy no seriço de stress utilizado

- $ kubectl run -i --tty load-generator --image=busybox /bin/sh
- Descobrir o endpoint da aplicação rodada anteriormente com $ kubectl get endpoints, no caso, será 100.96.2.7:8081
- $ while true; do wget -q -O- http://100.96.2.7:8081; done

5. Observar comportamento

- Rode $ kubectl get hpa e veja se a utilização do CPU aumenta e são feitos novos pods

Configurando o LoadBalancer

1. Dar deploy de uma aplicação com 3 replicas.

- $ kubectl run node-web-app --image=docker.io/marcelogdeandrade/node-web-app --port=8081 --

replicas=3

2. Dar expose no deployment com tipo LoadBalancer

- $ kubectl expose deployment node-web-app --type=LoadBalancer --name=node-web-app

3. Acesse a aplicação pelo IP externo criado.

- Minikube = $ minikube service node-web-app --url
- Outros = $ kubectl describe services

Juntando LoadBalancer com Horizontal Pod Autoscaler

1. Dar deploy de uma aplicação com 3 replicas e definição de recursos

- $ kubectl run node-web-app --image=docker.io/marcelogdeandrade/node-web-app --port=8081 --requests=cpu=200m --replicas=3

2. Dar expose no deployment com tipo LoadBalancer

- $ kubectl expose deployment node-web-app --type=LoadBalancer --name=node-web-app

3. Ativar autoscaler no deployment feito

- $ kubectl autoscale deployment node-web-app --cpu-percent=50 --min=3 --max=5

Deploy do MongoDB no Cluster

1. Uma alternativa simples é usar o serviço mLab.

2. Criar volume na AWS para ser usado no kubernetes

- $ aws ec2 create-volume --size 200 --region us-west-2 --availability-

zone us-west-2a

3. Criar o RC do Mongo a partir do arquivo yaml.

- $ kubectl create -f mongo.yaml

4. Criar serviço para expor o mongo a todo o cluster com o nome de mongo

- $ kubectl expose rc mongo-controller --port=27017 --name=mongo

Deploy da API no Cluster

1. Fazer o deploy após criar a instancia do Mongo
2. Dar deploy da imagem

- $ kubectl run api --image=docker.io/marcelogdeandrade/projeto-cloud-api --port=3000

3. Criar serviço de LoadBalancer para acesso externo da API

- $ kubectl expose deployment api --type=LoadBalancer --name=api

4. Para acessar a API, veja o valor de LoadBalancer Ingress no seguinte comando:

- $ kubectl describe service api

5. Trocar endpoint to cliente para o endpoint visto acima.

Passos futuros

1. Configurar deploys de Pods e Serviços através de um arquivo .yaml

RODAR PROJETO A PARTIR DOS SCRIPTS

Pré-Requisitos

- AWS CLI Configurada
- Kops
- Kubectl
- jq

Ordem dos Scripts

- full_install_cluster.sh (Esperar alguns minutos até o cluster iniciar)
- install_cluster_dependencies.sh
- install_mongodb.sh
- install_api.sh
- install_client.sh

1. Caso ja tenha feito a primeira instalação do cluster e queira iniciá-lo novamente, substitua o script de full_install_cluster.sh por start_cluster.sh.

- start_cluster.sh (Esperar alguns minutos até o cluster iniciar)
- install_cluster_dependencies.sh
- install_mongodb.sh
- install_api.sh
- install_client.sh