**DevOps Helm/Helmfile Kubernetes deployment**

**Delete/destroy all the AWS resources every time you do not use them**

**Note**: I assume that if you are going through this course during several days -**You always destroy all resources in AWS** It means that you **stop you Kubernetes** cluster every time you are **not working on it**. The easiest way is to do it via terraform cd /.../.../.../terraform\_code ; terraform destroy # hit yes

**destroy/delete**manually if**terraform**can't do that**:**

           - VOLUMES  
           - LoadBalancer/s (if exists)  
           - RecordSet/s (custom RecordSet/s)  
           - EC2 instances  
           - network resources  
           -  ...

**except:**  
           - S3 bucket        (delete once you do not want to use this free 1 YEAR account anymore, or you are done with this course.)  
           - Hosted Zone   (delete once you do not want to use this

Please **do not forget** redeploy **tiller**pod by using of this commands every time you are starting your Kubernetes cluster.

1. # Start your Kubernetes cluster
2. cd /.../.../.../terraform\_code
3. terraform apply
5. # Crete service account && initiate tiller pod in your Kubernetes cluster
6. kubectl create serviceaccount --namespace kube-system tiller
7. kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --serviceaccount=kube-system:tiller
8. # kubectl patch deploy --namespace kube-system tiller-deploy -p '{"spec":{"template":{"spec":{"serviceAccount":"tiller"}}}}'
9. helm init --service-account tiller --upgrade

**How to install KOPS binary**

**Simple shell function for** kops **installation**

Kubernetes documentation: <https://kubernetes.io/docs/getting-started-guides/kops/>

Copy and paste this code:

1. function install\_kops {
2. if [ -z $(which kops) ]
3. then
4. curl -LO https://github.com/kubernetes/kops/releases/download/$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag\_name | cut -d '"' -f 4)/kops-linux-amd64
5. chmod +x kops-linux-amd64
6. mv kops-linux-amd64 /usr/local/bin/kops
7. else
8. echo "kops is most likely installed"
9. fi
10. }
12. install\_kops

Hit enter and **kops binary** should be automatically installed to your Linux machine.

**Install kops on MacOS:**

1. curl -OL https://github.com/kubernetes/kops/releases/download/1.8.0/kops-darwin-amd64
2. chmod +x kops-darwin-amd64
3. mv kops-darwin-amd64 /usr/local/bin/kops
4. # you can also install using Homebrew
5. brew update && brew install kops

**How to install TERRAFORM binary**

Here is the **bash function** to install terrafrom:

1. TERRAFORM\_ZIP\_FILE=terraform\_0.11.7\_linux\_amd64.zip
2. TERRAFORM=https://releases.hashicorp.com/terraform/0.11.7
3. TERRAFORM\_BIN=terraform
5. function install\_terraform {
6. if [ -z $(which $TERRAFORM\_BIN) ]
7. then
8. wget ${TERRAFORM}/${TERRAFORM\_ZIP\_FILE}
9. unzip ${TERRAFORM\_ZIP\_FILE}
10. sudo mv ${TERRAFORM\_BIN} /usr/local/bin/${TERRAFORM\_BIN}
11. rm -rf ${TERRAFORM\_ZIP\_FILE}
12. else
13. echo "Terraform is most likely installed"
14. fi
16. }
18. install\_terraform

Alternatively:

**Install terraform on MacOS :**

1. 1) Download ZIP file
2. wget https://releases.hashicorp.com/terraform/0.11.7/terraform\_0.11.7\_darwin\_amd64.zip
3. 2) unzip this ZIP package
4. 3) copy it to your executable path

**Install terraform on Windows:**

1. 1) Download ZIP file
2. wget https://releases.hashicorp.com/terraform/0.11.7/terraform\_0.11.7\_windows\_amd64.zip
3. 2) unzip this ZIP package
4. 3) copy it to your executable path

**How to install KUBECTL binary**

**How to install kubectl binary to Linux like OS**

Copy and paste this code to your command line:

1. KUBECTL\_BIN=kubectl
3. function install\_kubectl {
4. if [ -z $(which $KUBECTL\_BIN) ]
5. then
6. curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/$KUBECTL\_BIN
7. chmod +x ${KUBECTL\_BIN}
8. sudo mv ${KUBECTL\_BIN} /usr/local/bin/${KUBECTL\_BIN}
9. else
10. echo "Kubectl is most likely installed"
11. fi
13. }

**Run**this command:

1. install\_kubectl

By now you should be able to use kubectl command.

**How to start Kubernetes cluster**

**How to start Kubernetes cluster by using Kops and Terraform**

1. SSH\_KEYS=~/.ssh/udemy\_devopsinuse
3. if [ ! -f "$SSH\_KEYS" ]
4. then
5. echo -e "\nCreating SSH keys ..."
6. ssh-keygen -t rsa -C "udemy.course" -N '' -f ~/.ssh/udemy\_devopsinuse
7. else
8. echo -e "\nSSH keys are already in place!"
9. fi
11. echo -e "\nCreating kubernetes cluster ...\n"
13. kops create cluster \
14. --name=course.<example>.com \
15. --state=s3://course.<example>.com \
16. --authorization RBAC \
17. --zones=<define-zone> \
18. --node-count=2 \
19. --node-size=t2.micro \
20. --master-size=t2.micro \
21. --master-count=1 \
22. --dns-zone=course.<example>.com \
23. --out=terraform\_code \
24. --target=terraform \
25. --ssh-public-key=~/.ssh/udemy\_devopsinuse.pub

**How to run Jupyter Notebooks locally as Docker image**

**Run this code to start up your personal Jupyter Notebook locally from Docker image**

1. docker ps
2. docker run --name djupyter -d -p 8888:8888 jupyter/scipy-notebook:2c80cf3537ca
3. 6f1d5c03efced84f7e9502649c1618e8304f304a69ce3f6100d2ef11111
5. docker logs 6f1d5c03efced84f7e9502649c1618e8304f304a69ce3f6100d2ef11111 -f
6. ...
7. ...
8. Copy/paste this URL into your browser when you connect for the first time,
9. to login with a token:
10. http://localhost:8888/?token=<some\_long\_token>
11. ...
12. ...
13. docker stop djupyter

**How to deploy Juypyter Notebooks to Kubernetes via YAML file**

**Simple Kubernetes YAML specification for Jupyter Notebooks**

Create Save these line to a file: jupyter\_notebook.yaml

Execute deployment: kubectl create -f jupyter\_notebook.yaml

1. apiVersion: apps/v1
2. kind: Deployment
3. metadata:
4. name: jupyter-k8s-udemy
5. labels:
6. app: jupyter-k8s-udemy
7. spec:
8. replicas: 1
9. selector:
10. matchLabels:
11. app: jupyter-k8s-udemy
12. template:
13. metadata:
14. labels:
15. app: jupyter-k8s-udemy
16. spec:
17. containers:
18. - name: minimal-notebook
19. image: jupyter/minimal-notebook:latest
20. ports:
21. - containerPort: 8888
22. command: ["start-notebook.sh"]
23. args: ["--NotebookApp.token=''"]
24. ---
25. kind: Service
26. apiVersion: v1
27. metadata:
28. name: jupyter-k8s-udemy
29. spec:
30. type: NodePort
31. selector:
32. app: jupyter-k8s-udemy
33. ports:
34. - protocol: TCP
35. nodePort: 30040
36. port: 8888
37. targetPort: 8888

**How to SSH to the physical servers in AWS**

**How to SSH to physical EC2 instances in AWS**

1. ssh -i ~/.ssh/<your\_public\_key>.pub admin@<public\_ip\_address\_of\_node\_1>
2. ssh -i ~/.ssh/<your\_public\_key>.pub admin@<public\_ip\_address\_of\_node\_2>
3. ssh -i ~/.ssh/<your\_public\_key>.pub admin@<public\_ip\_address\_of\_master>

These publicly accessible IP addresses can be retrieved even from your command line

1. aws ec2 describe-instances \
2. --query "Reservations[\*].Instances[\*].PublicIpAddress" \
3. --output=text

Install HELM binary and activate HELM user account in your cluster

**Install HELM binary to your PC/server**

1. HELM\_TAR\_FILE=helm-v2.9.1-linux-amd64.tar.gz
2. HELM\_URL=https://storage.googleapis.com/kubernetes-helm
3. HELM\_BIN=helm
5. function install\_helm {
7. if [ -z $(which $HELM\_BIN) ]
8. then
9. wget ${HELM\_URL}/${HELM\_TAR\_FILE}
10. tar -xvzf ${HELM\_TAR\_FILE}
11. chmod +x linux-amd64/${HELM\_BIN}
12. sudo cp linux-amd64/${HELM\_BIN} /usr/local/bin/${HELM\_BIN}
13. rm -rf ${HELM\_TAR\_FILE} linux-amd64
14. echo -e "\nwhich ${HELM\_BIN}"
15. which ${HELM\_BIN}
16. else
17. echo "Helm is most likely installed"
18. fi
20. }

**Setup tiller user account in your Kubernetes cluster in AWS**

1. kubectl create serviceaccount --namespace kube-system tiller
2. kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --serviceaccount=kube-system:tiller
3. # kubectl patch deploy --namespace kube-system tiller-deploy -p '{"spec":{"template":{"spec":{"serviceAccount":"tiller"}}}}'
4. helm init --service-account tiller --upgrade

**You can learn more about HELM at this web page**  
  
https://docs.helm.sh/

How to use HELM to deploy GOGS from locally downloaded HELM CHARTS

**How to create local file/directory structure for GOGS installation from local files**

I would suggest to follow the video and create it on your own.

1. ssh username@your\_server
3. # this will clone all the HELM charts
4. username@your\_server:~$ git clone https://github.com/helm/charts.git
6. username@your\_server:~$ ls charts/
7. code-of-conduct.md incubator OWNERS README.md stable
8. CONTRIBUTING.md LICENSE PROCESSES.md REVIEW\_GUIDELINES.md test
10. # create some folder
11. username@your\_server:~$ mkdir local\_installation
12. username@your\_server:~$ cd local\_installation
14. # copy gogs/ folder from charts/ folder you have cloned from github.com
15. username@your\_server:~/local\_installation$ rsync -avhx ../charts/incubator/gogs .
16. username@your\_server:~/local\_installation$ ls
17. gogs
19. # create an empty folder called charts/ inside your gogs/ folder
20. username@your\_server:~/local\_installation$ mkdir charts; cd charts
21. username@your\_server:~/local\_installation/gogs/charts$ rsync -avhx ../../../charts/stable/postgresql .

24. # final structure
26. username@your\_server:~/local\_installation/gogs/charts$ cd ..
27. username@your\_server:~/local\_installation/gogs$ tree -L 2
29. username@your\_server:~/local\_installation/gogs$ tree -L 2
30. .
31. ├── charts
32. │   └── postgresql
33. ├── Chart.yaml
34. ├── README.md
35. ├── requirements.lock
36. ├── requirements.yaml
37. ├── templates
38. │   ├── configmap-tcp.yaml
39. │   ├── configmap.yaml
40. │   ├── deployment.yaml
41. │   ├── \_helpers.tpl
42. │   ├── ingress.yaml
43. │   ├── NOTES.txt
44. │   ├── pvc.yaml
45. │   ├── secrets.yaml
46. │   └── service.yaml
47. └── values.yaml