가상화 클라우드 K8S

LAB

Lab 1_Pre-Setting

실습 주의 사항

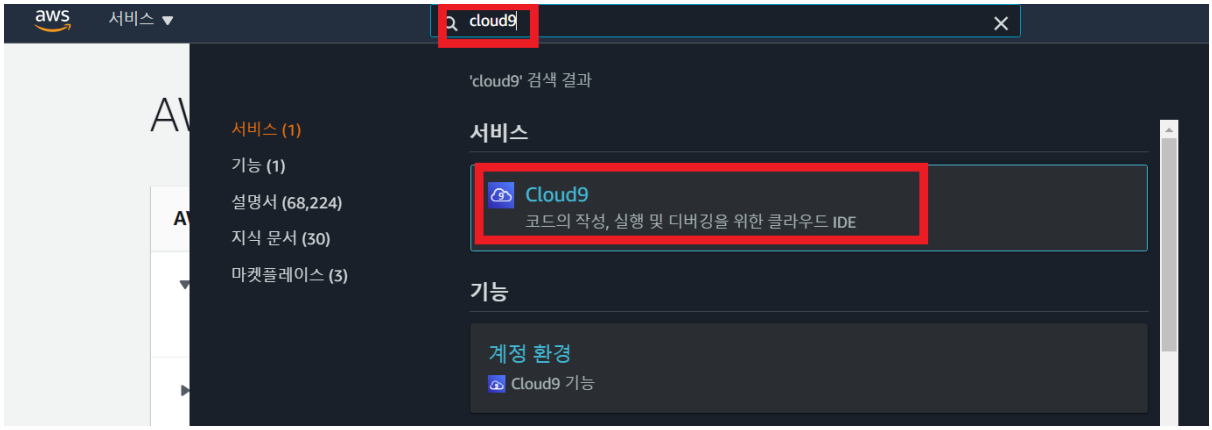
*a****** 는 계정명입니다. 본인의 계정명으로 변경하여 진행합니다.*

<> 괄호에는 각 수강생별 고유한 값들을 의미합니다.

제공되는 IP나 실습진행시 출력되는 고유한 값들로 변경하여 진행합니다.

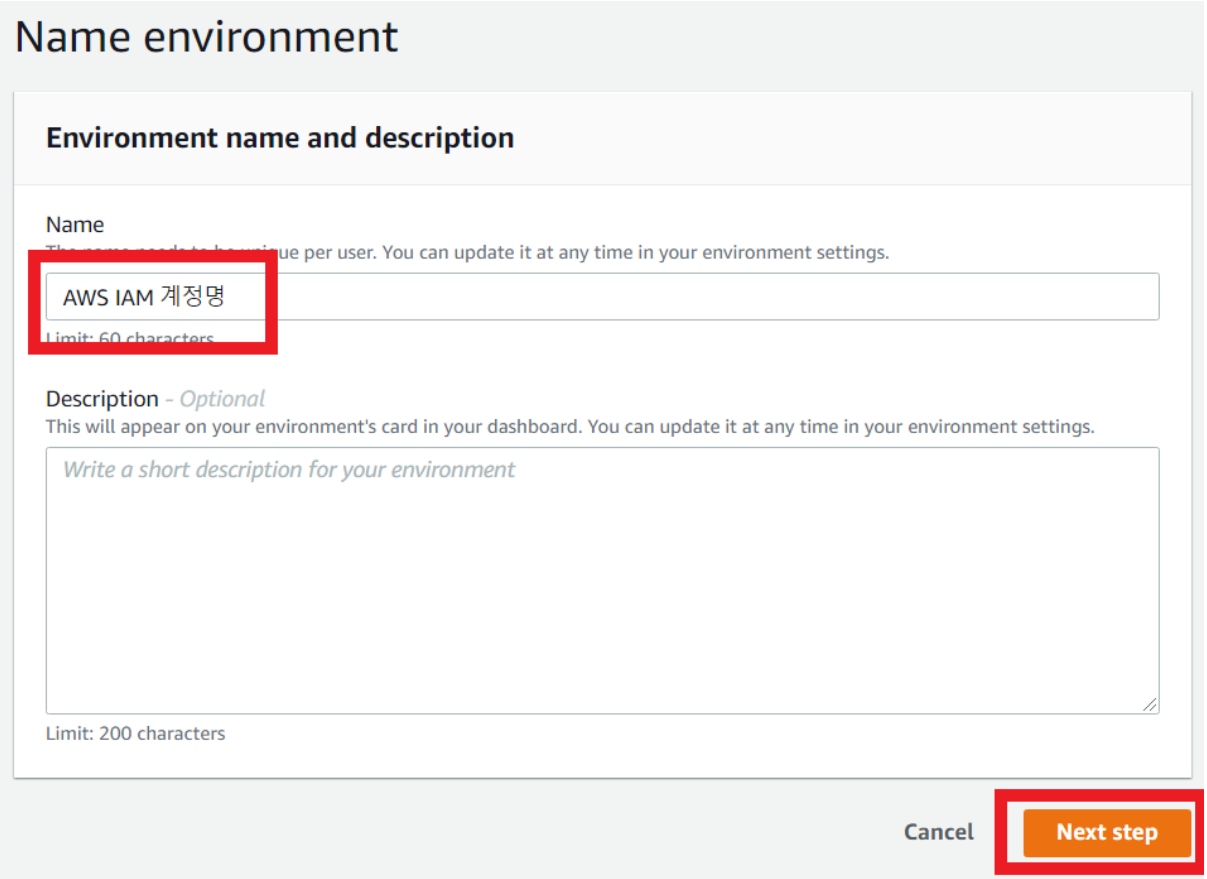
이때 <> 괄호도 지워주며 진행합니다.

1. 제공 받은 계정으로 AWS에 로그인
2. Cloud9 서비스 검색 및 선택



3. Create environment 클릭

4. Name : a****** 입력 후 Next step 클릭



5. 아래와 같이 선택한 뒤 하단의 Next step 클릭

Environment type : Create a new EC2 instance for environment (direct access)

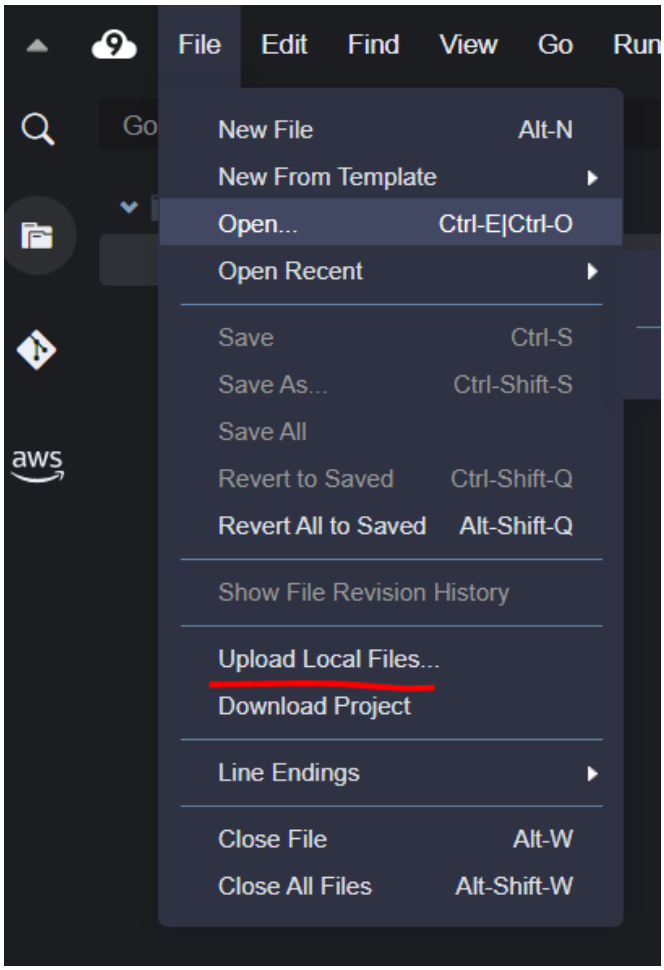
Instance type : t3.small (2 GiB RAM + 2 vCPU)

Platform : Ubuntu Server 18.04 LTS

6. 제공 받은 키페어 파일을 Cloud9 환경에 업로드

좌측 상단 File 클릭, Upload Local Files 클릭하여

키페어 파일을 드래그 앤 드랍 또는 파일을 열어 업로드



7. 업로드된 키페어 파일의 권한을 수정

chmod 600 k8skey.pem

8. ssh 명령을 사용하여 Master에 접속

ssh -i k8skey.pem <master ip>

9. hostname 을 변경

sudo -i

sudo hostnamectl set-hostname k8s-master

sudo -i

10. kubernetes 클러스터 설정

kubeadm init --pod-network-cidr=172.16.0.0/16 --apiserver-advertise-address=<Master IP>

명령 완료 후 출력되는 kubeadm join 명령어를 메모장에 저장

11. Master 노드 설정

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

export KUBECONFIG=/etc/kubernetes/admin.conf

kubectl create -f ₩

https://raw.githubusercontent.com/wsjang619/k8s_course/master/lab1/yaml/flannel.yaml

12. kubectl 설정

source <(kubectl completion bash)

echo "source <(kubectl completion bash)" >> ~/.bashrc

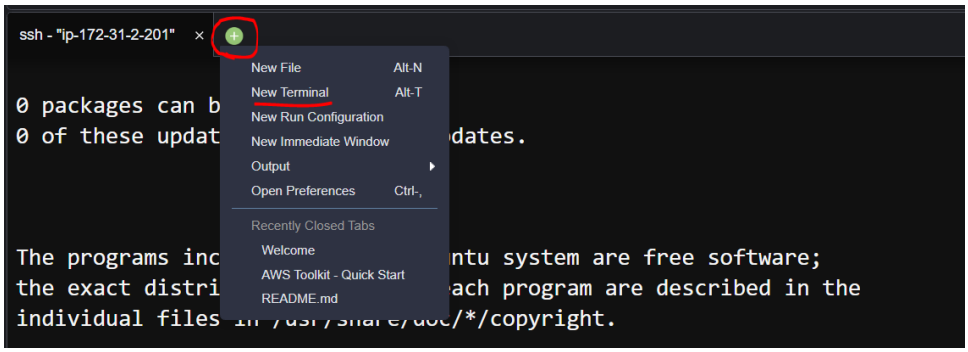
source /etc/bash_completion

alias k=kubectl

complete -F __start_kubectl k

13. Cloud9에서 아래 +버튼을 클릭, 터미널을 추가하여 Worker 1 에 접속

ssh -i k8skey.pem <worker 1 ip>



14. hostname 변경

sudo -i

sudo hostnamectl set-hostname k8s-worker1

sudo -i

15. Master와 연동

<kubeadm join ~~~~~~ 10번 과정에서 메모장에 저장해둔 명령어>

16. 13~15번을 참고하여 worker2 도 Master와 연동 작업 수행

17. Master 에서 연동된 Worker 확인

kubectl get nodes

```
root@k8s-master:~# kubectl get nodes
NAME          STATUS   ROLES                  AGE   VERSION
k8s-master    Ready    control-plane,master   21m   v1.22.2
k8s-worker1   Ready    <none>                 39s   v1.22.2
k8s-worker2   Ready    <none>                 37s   v1.22.2
root@k8s-master:~#
```

18. 실습 파일 다운로드

git clone https://github.com/education-labs/k8s

Lab 2_Pod

실습 디렉토리 이동

cd ~/k8s/lab2-pod

1. Pod 조회

kubectl get pods

kubectl get pod

kubectl get po

2. yaml 확인

cat pod.yaml

3. yaml 을 활용한 Pod 생성

kubectl create -f pod.yaml

4. Pod 조회 및 세부정보 확인

kubectl get po

kubectl describe po

5. Pod 삭제

kubectl delete po <Pod명>

kubectl delete po --all

6. Pod 조회

kubectl get pod

7. run 명령을 사용한 pod 생성

조건 pod name : lab2-pod container image : nginx container port : 80

kubectl run lab2-pod --image=nginx --port=80

8. 7과정 에서 생성한 pod 조회

kubectl get po

kubectl describe po

9. 7에서 생성한 pod 삭제

kubectl delete pod lab2-pod

kubectl delete pod --all

Lab 3_Namespace

실습 디렉토리 이동

cd ~/k8s/lab3-ns

1. Namespace 확인

kubectl get namespace

kubectl get ns

2. yaml 확인

cat ns.yaml

3. yaml 을 활용한 namespace 생성

kubectl create -f ns.yaml

4. Namespace 재확인

kubectl get ns

5. n1 namespace 안에 pod 생성

kubectl create -f pod.yaml -n lab3-ns

6. pod 확인

kubectl get po

kubectl get po -n lab3-ns

7. pod2.yaml 확인

cat pod2.yaml

8. pod2.yaml 생성

kubectl create -f pod2.yaml

9. 5에서 생성한 Pod 삭제

kubectl delete pod lab3-pod -n lab3-ns

10. pod 재확인

kubectl get pod -n lab3-ns

11. 3에서 생성한 Namespace 삭제

kubectl delete ns lab3-ns

12. create, run 명령으로 namespace 와 pod 생성

-namespace 조건-

namespace name : lab3-ns2

-pod 조건-

pod name : lab3-pod3

namespace : lab3-ns2

container image : nginx

container port : 80

kubectl create ns lab3-ns2

kubectl run lab3-pod3 --image=nginx --port=80 -n lab3-ns2

13. 10에서 생성한 namespace와 pod 조회

kubectl get ns

kubectl get pod -n lab3-ns2

14. namespace 삭제로 해당 namespace에 있던 리소스 모두와 함께 삭제

kubectl delete ns lab3-ns2

15. 삭제 확인

kubectl get ns

Lab 4_Replicaset

실습 디렉토리 이동

cd ~/k8s/lab4-rs

1. Replicaset 확인

kubectl get replicaset

kubectl get rs

2. yaml 확인

cat rs.yaml

3. yaml 을 활용한 replicaset 생성

kubectl create -f rs.yaml

4. replicaset, pod 확인

kubectl get rs

kubectl get pod

5. pod 1개 삭제

kubectl delete pod <pod명>

6. pod 재배포 확인

kubectl get po

7. replicaset 삭제

kubectl delete rs --all

8. replicaset 삭제 확인

kubectl get rs

kubectl get pod

Lab 5_Deployment

실습 디렉토리 이동

cd ~/k8s/lab5-deploy

1. Deployment 확인

kubectl get deployment

kubectl get deploy

2. yaml 확인

cat deploy.yaml

3. yaml 을 활용한 Deployment 생성

kubectl create -f deploy.yaml

4. deployment, replicaset, pod 확인

kubectl get deploy

kubectl get rs

kubectl get pod

5. pod 1개 삭제

kubectl delete pod <pod명>

6. pod 재배포 확인

kubectl get po

7. Deployment 삭제

kubectl delete deploy --all

8. Deployment 삭제 확인

kubectl get deploy

kubectl get pod

9. 터미널을 하나 더 오픈하여 모니터링용 터미널을 생성합니다.

watch -n 0.5 kubectl get pod

10. create 명령으로 deployment 생성

-deployment 조건-

deployment name : lab5-deploy2

container image : nginx:1.14.0

container port : 80

replicas : 3

kubectl create deploy lab5-deploy2 --image=nginx:1.14.0 --port=80 --replicas=3

11. 컨테이너 이미지 1.15.0으로 버전 업데이트

kubectl set image deployment/lab5-deploy2 nginx=nginx:1.15.0 --record=true

이때 명령어 수행 직후 모니터링 터미널로 동작 확인 record=true 값으로 해야 히스토리 확인시 어떤 내용인지 확인 가능

12. 업데이트 내역 확인

kubectl describe pod

kubectl describe deploy

13. 업데이트 방식 변경

kubectl edit deploy lab5-deploy2

strategy:

rollingUpdate:

maxSurge: 25%

maxUnavailable: 25%

type: RollingUpdate

를

strategy:

type: Recreate

로 수정합니다.

vi 편집기 사용법과 동일합니다.

14. 컨테이너 이미지 1.16.0 으로 버전 업데이트

kubectl set image deployment/lab5-deploy2 nginx=nginx:1.16.0 --record=true

이때 명령어 수행 직후 모니터링 터미널로 동작 확인

15. 업데이트 내역 확인

kubectl describe pod

kubectl describe deploy

16. 롤 아웃 기록 확인

kubectl rollout history deploy/lab5-deploy2

17. 직전 버전으로 롤백

kubectl rollout undo deploy/lab5-deploy2

18. 버전 확인

kubectl describe deploy

19. 리비전 지정하여 롤백

kubectl rollout undo deploy/lab5-deploy2 --to-revision=1

20. 버전 확인

kubectl describe deploy

21. 스케일링

kubectl scale deploy/lab5-deploy2 --replicas=5

22. 결과 확인

kubectl get pod

kubectl describe deploy

23. deployment 삭제

kubectl delete deploy --all

Lab 6_ClusterIP Service

실습 디렉토리 이동

cd ~/k8s/lab6-clusterip

1. Service 조회

kubectl get service

kubectl get svc

2. yaml 확인

cat pod.yaml

cat svc-clusterip.yaml

3. yaml 을 활용한 ClusterIP 유형의 Service및 pod 생성

kubectl create -f svc-clusterip.yaml

kubectl create -f pod.yaml

4. 위 3에서 만든 리소스 확인

kubectl get pod

kubectl get svc

5. curl 명령을 통한 통신 확인

curl <lab6-svc의 ClusterIP>:9090

4에서 확인한 IP를 입력합니다.

6. 리소스 삭제

kubectl delete pod,svc --all

Lab 7_NodePort Service

실습 디렉토리 이동

cd ~/k8s/lab7-nodeport

1. yaml 확인

cat pod.yaml

cat svc-nodeport.yaml

2. yaml 을 활용한 NodePort 유형의 Service및 pod 생성

kubectl create -f svc-nodeport.yaml

kubectl create -f pod.yaml

3. 생성된 Service 확인

kubectl get svc

4. Worker1 로의 접근을 통해 통신 확인

curl <Worker 1 노드의 IP>:30000

5. Worker2 로의 접근을 통해 통신 확인

curl <Worker 2 노드의 IP>:30000

6. 리소스 삭제

kubectl delete pod,svc --all

Lab 8_Loadbalancer Service

실습 디렉토리 이동

cd ~/k8s/lab8-loadbalancer

1. yaml 확인

cat pod.yaml

cat svc-loadbalancer.yaml

2. yaml 을 활용한 Loadbalancer 유형의 Service및 pod 생성

kubectl create -f svc-loadbalancer.yaml

kubectl create -f pod.yaml

3. 생성한 Service 확인

kubectl get svc

ExternalIP 를 발급받지 못하는 Pending 상태 확인

4. Loadbalancer 타입의 서비스가 생성한 ClusterIP 를 활용하여 내부에서 접근 시도

curl <Loadbalancer 타입의 서비스의 ClusterIP>:9090

5. Loadbalancer 타입의 서비스가 생성한 NodePort 를 활용하여 외부에서 접근 시도

curl <아무 Worker IP>:<Loadbalancer 타입의 서비스의 NodePort>

Lab 9_EmptyDir Volume

실습 디렉토리 이동

cd ~/k8s/lab9-emptydir

1. yaml 확인

cat pod.yaml

2. yaml 을 활용한 emptydir 유형의 Volume을 사용하는 Pod 생성

kubectl create -f pod.yaml

3. 위 2에서 만든 Pod 내부의 컨테이너 redis로 접속합니다.

kubectl exec -it emptydir-pod --container redis -- /bin/bash

4. 마운트 된 디렉토리로 이동 후 파일생성

cd /mount1

echo hello emptydir >> test.txt

cat test.txt

exit

5. 위 2에서 만든 Pod 내부의 컨테이너 nginx로 접속합니다.

kubectl exec -it emptydir-pod --container nginx -- /bin/bash

6. 디렉토리 이동 후 4에서 생성한 파일 확인

cd /mount2

ls

cat test.txt

exit

7. 리소스 삭제

kubectl delete pod emptydir-pod

Lab 10_HostPath Volume

실습 디렉토리 이동

cd ~/k8s/lab10-hostpath

1. yaml 확인

cat pod.yaml

2. yaml 을 활용한 hostpath 유형의 Volume을 사용하는 Pod 생성

kubectl create -f pod.yaml

3. 위 2에서 생성한 redis 컨테이너로 접속

kubectl exec -it hostpath-pod --container redis -- /bin/bash

4. 마운트 된 디렉토리로 이동 후 파일생성

cd /mount1

echo hello hostpath >> test.txt

cat test.txt

exit

5. 위 2에서 생성한 Pod 가 어떤 노드에 생성되었는지 확인

kubectl get pod -o wide

6. 위 5에서 확인한 노드의 터미널로 이동하여 파일 생성 확인

cd /tmp

ls

cat test.txt

7. 리소스 삭제

kubectl delete pod hostpath-pod

Lab 11_Persistent Volume

실습 디렉토리 이동

cd ~/k8s/lab11-pv

1. PV, PVC, Pod yaml 확인

cat pv.yaml pvc.yaml pod.yaml

2. 위 13에서 확인한 yaml로 리소스 생성

kubectl create -f pv.yaml

kubectl create -f pvc.yaml

kubectl create -f pod.yaml

3. 생성된 리소스 확인

kubectl get pv,pvc,po

4. persistentvolume 안에 있는 컨테이너로 접속

kubectl exec -it pv-pod --container container -- /bin/bash

5. 마운트 된 디렉토리로 이동 후 파일생성

cd /mount1

echo hello pv >> pv.txt

cat pv.txt

exit

6. persistentvolume라는 pod 가 어떤 노드에 생성되었는지 확인

kubectl get po pv-pod -o wide

7. 위 18에서 확인한 노드의 터미널로 이동하여 파일 생성 확인

cd /tmp

ls

cat pv.txt

8. 클리어

kubectl delete pod,pv,pvc --all

Appendix_Autoscaling HPA

실습 디렉토리 이동

cd ~/k8s/appendix-hpa

1. Metic-server 설치

kubectl create -f metric-server.yaml

2. Deployment 편집

kubectl edit deployment metrics-server -n kube-system

/kubelet-use-node-status-port

- --kubelet-use-node-status-port

- --metric-resolution=15s

- --kubelet-insecure-tls

맨 아랫줄을 추가해줍니다.(- --kubelet-insecure-tls)

3. 동작 확인

kubectl top node

4. 실습에 필요한 Deployment와 Service 생성

kubectl create -f php-apache.yaml

5. HPA 생성

kubectl autoscale deployment php-apache ₩

--cpu-percent=50 --min=1 --max=10

6. 생성한 HPA 확인

kubectl get hpa

7. php-apache 서비스의 ClusterIP 확인

kubectl get svc

8. 부하 생성(다른 터미널을 추가하여 진행)

kubectl run -i --tty load-generator --rm ₩

--image=busybox --restart=Never -- ₩

/bin/sh -c "while sleep 0.01; do ₩

wget -q -O- http://<위에서확인한 IP>; done"

9. 1분정도 뒤 Pod 증가 확인

kubectl get hpa

kubectl get deployment php-apache

10. 부하 중지(부하 생성 시 터미널에서 진행)

Ctrl + C

11. 약 5~7분 뒤 Pod 감소 확인

kubectl get hpa

kubectl get deployment php-apache

12. clear

kubectl delete pod,hpa,deploy --all