|  |  |  |
| --- | --- | --- |
|  | earn DevOps: Kubetnetes - Edward Vaaene    kubernetes.io    <https://kubernetes.io/docs/reference/kubectl/cheatsheet/>    <https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands> |  |
|  | ı III III  ııııııı  ııııııı  ııııııı o  ııııııı  ııııııı  ııııııı  ııııııı  2 |  |
|  |  |  |
|  | <https://547286019343.signin.aws.amazon.com/console>    ravi  Ravi@123$ |  |
|  | ssh -i "ravikishorek.pem" ubuntu@ec2-3-109-201-172.ap-south-1.compute.amazonaws.com    *From <*[*https://ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#ConnectToInstance:instanceId=i-0610eba3f1d5ca693*](https://ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#ConnectToInstance:instanceId=i-0610eba3f1d5ca693)*>* |  |
| Kubernetis installation | # Install Kubernetes on Ubuntu 18.04    Quick guide to install Kubernetes via Minikube on Ubuntu 18.04.    ## 1. Install VirtualBox  Install VirtualBox to be used as the hypervisor.    ```bash  sudo apt-get install -y virtualbox virtualbox-ext-pack  ```    ## 2. Install kubectl  Install kubectl, the kubernetes command line tool.    ```bash  sudo apt-get update && \  sudo apt-get install -y apt-transport-https && \  curl -s <https://packages.cloud.google.com/apt/doc/apt-key.gpg> | sudo apt-key add - && \  sudo touch /etc/apt/sources.list.d/kubernetes.list && \  echo "deb <http://apt.kubernetes.io/> kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes.list && \  sudo apt-get update && \  sudo apt-get install -y kubectl  ```    ## 3. Install Minikube  Install Minikube to run your single node kubernetes cluster.    ```bash  curl -Lo minikube <https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64> \  && chmod +x minikube && \  sudo cp minikube /usr/local/bin && rm minikube  ```    ## 4. Done!  You can run Minikube and start your cluster by running:    ```bash  minikube start  ```    You can then see that it is running:    ```bash  kubectl get nodes  ```  ```enable autocomplete  source <(kubectl completion bash)  echo "source <(kubectl completion bash)" >> ~/.bashrc  alias k=kubectl  complete -F \_\_start\_kubectl k  ```      Yay! |  |
|  | Z OPON  a ON |  |
|  | Pod.yml  apiVersion: v1  kind: Pod  metadata:    name: pod1  spec:    containers:      - name: nginx-ravi        image: nginx        ports:          - containerPort: 80    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl create -f pod.yml  pod/pod1 created    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get pod  NAME READY STATUS RESTARTS AGE  pod1 1/1 Running 0 2m8s    kubectl describe pod <pod-name>    kubectl describe pod pod1    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get pod -n kube-system  NAME READY STATUS RESTARTS AGE  coredns-64897985d-rrcgg 1/1 Running 0 174m  etcd-minikube 1/1 Running 0 174m  kube-apiserver-minikube 1/1 Running 0 174m  kube-controller-manager-minikube 1/1 Running 0 174m  kube-proxy-dwxm9 1/1 Running 0 174m  kube-scheduler-minikube 1/1 Running 0 174m  storage-provisioner 1/1 Running 1 (174m ago) 174m |  |
| Creating a pod with 2 containers in it | apiVersion: v1  kind: Pod  metadata:    name: pod1  spec:    containers:      - name: nginx-ravi        image: nginx        ports:          - containerPort: 81      - name: watcher-ravi        image: afakharany/watcher |  |
|  | **Pod3.yml**  apiVersion: v1  kind: Pod  metadata:    name: pod3    labels:      app: helloworld  spec:    containers:    - name: k8s-demo      image: arjunachari12/k8s-demo      resources:        limits:          memory: "128Mi"          cpu: "500m"      ports:        - containerPort: 3000    **Pod3service.yml**  apiVersion: v1  kind: service  metadata:    name: helloworld-service  spec:    ports:      - port: 31001        nodePort: 31002        targetPort: 3000        protocol: TCP    selector:      app: helloworld    type: NodePort | NOd•Pu p 31″ |
|  | **Kubectl --help**  **Kubectl explain service**  kubectl explain service --recursive  kubectl explain pod --recursive | K describe svc helloworld-service  ubuntu@ip-172-31-11-243:~/kubernetes-training$ k describe svc helloworld-service  Name: helloworld-service  Namespace: default  Labels: <none>  Annotations: <none>  Selector: app=helloworld  Type: NodePort  IP Family Policy: SingleStack  IP Families: IPv4  IP: 10.103.61.1  IPs: 10.103.61.1  Port: <unset> 31001/TCP  TargetPort: 3000/TCP  NodePort: <unset> 31002/TCP  Endpoints: <none>  Session Affinity: None  External Traffic Policy: Cluster  Events: <none> |
|  | minikube service helloworld-service --url    curl <http://192.168.49.2:31002> ==> runs the particular docker with port 3000 as per service written |  |
|  | Scaling  • Scaling in Kubernetes can be done using the Replication Controller  • The replication controller will ensure a specified number of pod replicas will  run at all time  A pods created with the replica controller will automatically be replaced if they  fail, get deleted, or are terminated  • Using the replication controller is also recommended if you just want to make  1 pod is always running, even after reboots  sure  • You can then run a replication controller with just 1 replica  • This makes sure that the pod is always running  (Guest) |  |
|  | **Replicaset.yml**  apiVersion: apps/v1  kind: ReplicaSet  metadata:    name: helloworld-controller    labels:      app: helloworld  spec:    replicas: 5    selector:      matchLabels:        app: helloworld    template:      metadata:        name: pod3        labels:          app: helloworld      spec:        containers:        - name: k8s-demo          image: arjunachari12/k8s-demo          ports:          - containerPort: 3000 | ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl create -f replicaset.yml  replicaset.apps/helloworld-controller created    ubuntu@ip-172-31-11-243:~/kubernetes-training$ k get pod  NAME READY STATUS RESTARTS AGE  helloworld-controller-gfj4l 1/1 Running 0 22s  helloworld-controller-lbvbc 1/1 Running 0 22s  helloworld-controller-nw4jv 1/1 Running 0 22s  helloworld-controller-v4t78 1/1 Running 0 22s  pod1 2/2 Running 0 84m  pod3 1/1 Running 0 38m    If 1 pod is deleted, automaticaaly new pod gets created by replica-controller    ubuntu@ip-172-31-11-243:~/kubernetes-training$ k get replicasets  NAME DESIRED CURRENT READY AGE  helloworld-controller 5 5 5 10m  ubuntu@ip-172-31-11-243:~/kubernetes-training$ k get rs  NAME DESIRED CURRENT READY AGE  helloworld-controller 5 5 5 10m  ubuntu@ip-172-31-11-243:~/kubernetes-training$ k delete rs helloworld-controller  replicaset.apps "helloworld-controller" deleted  ubuntu@ip-172-31-11-243:~/kubernetes-training$ |
|  | **Deployments**  Helloworld-deployment.yml    apiVersion: apps/v1  kind: Deployment  metadata:    name: helloworld-deployment  spec:    replicas: 10    selector:      matchLabels:        app: helloworld    template:      metadata:        name: pod3        labels:          app: helloworld      spec:        containers:        - name: k8s-demo          image: arjunachari12/k8s-demo          ports:          - name: nodejs-port            containerPort: 3000 | kubectl create -f helloworld-deployment.yml    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get pod  NAME READY STATUS RESTARTS AGE  helloworld-deployment-5456d65445-54nvc 1/1 Running 0 35s  helloworld-deployment-5456d65445-6lq9x 1/1 Running 0 35s  helloworld-deployment-5456d65445-6xh2g 1/1 Running 0 35s  helloworld-deployment-5456d65445-82cg2 1/1 Running 0 35s  helloworld-deployment-5456d65445-94sbp 1/1 Running 0 35s  helloworld-deployment-5456d65445-gffgn 1/1 Running 0 35s  helloworld-deployment-5456d65445-qg2v8 1/1 Running 0 35s  helloworld-deployment-5456d65445-qt58v 1/1 Running 0 35s  helloworld-deployment-5456d65445-sn5pb 1/1 Running 0 35s  helloworld-deployment-5456d65445-wg647 1/1 Running 0 35s      Directly create deployment without creating yml file  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl create deployment my-dep --image=busybox  deployment.apps/my-dep created |
|  | **Deployment & Service in single file**    apiVersion: apps/v1  kind: Deployment  metadata:    name: helloworld-deployment  spec:    replicas: 10    selector:      matchLabels:        app: helloworld    template:      metadata:        name: pod3        labels:          app: helloworld      spec:        containers:        - name: k8s-demo          image: arjunachari12/k8s-demo          ports:          - name: nodejs-port            containerPort: 3000  ---  apiVersion: v1  kind: Service  metadata:    name: helloworld-service  spec:    selector:      app: helloworld    ports:    - port: 31001      nodePort: 31002      targetPort: 3000      protocol: TCP    type: NodePort | kubectl create -f helloworld-deployment.yml  deployment.apps/helloworld-deployment created  service/helloworld-service created    kubectl get all  NAME READY STATUS RESTARTS AGE  pod/helloworld-deployment-5456d65445-475zs 1/1 Running 0 76s  pod/helloworld-deployment-5456d65445-6tzcw 1/1 Running 0 76s  pod/helloworld-deployment-5456d65445-7plnp 1/1 Running 0 76s  pod/helloworld-deployment-5456d65445-bfqhh 1/1 Running 0 76s  pod/helloworld-deployment-5456d65445-g2lbn 1/1 Running 0 76s  pod/helloworld-deployment-5456d65445-knxx8 1/1 Running 0 76s  pod/helloworld-deployment-5456d65445-qs5tt 1/1 Running 0 76s  pod/helloworld-deployment-5456d65445-slxct 1/1 Running 0 76s  pod/helloworld-deployment-5456d65445-tqg92 1/1 Running 0 76s  pod/helloworld-deployment-5456d65445-xn6w9 1/1 Running 0 76s    NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  service/helloworld-service NodePort 10.110.28.34 <none> 31001:31002/TCP 76s  service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 22h    NAME READY UP-TO-DATE AVAILABLE AGE  deployment.apps/helloworld-deployment 10/10 10 10 76s    NAME DESIRED CURRENT READY AGE  replicaset.apps/helloworld-deployment-5456d65445 10 10 10 76s      ubuntu@ip-172-31-11-243:~/kubernetes-training$ minikube service helloworld-service --url  <http://192.168.49.2:31002>  ubuntu@ip-172-31-11-243:~/kubernetes-training$ curl <http://192.168.49.2:31002>  Hello World!ubuntu@ip-172-31-11-243:~/kubernetes-training$ |
| Challenge | ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl set image deployment/helloworld-deployment k8s-demo=arjunachari12/k8s-demo:2  deployment.apps/helloworld-deployment image updated  ubuntu@ip-172-31-11-243:~/kubernetes-training$ minikube service helloworld-service --url  <http://192.168.49.2:31002>  ubuntu@ip-172-31-11-243:~/kubernetes-training$ curl <http://192.168.49.2:31002>  Hello World v2!ubuntu@ip-172-31-11-243:~/kubernetes-training$    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl rollout undo deployment/helloworld-deployment  deployment.apps/helloworld-deployment rolled back  ubuntu@ip-172-31-11-243:~/kubernetes-training$ curl <http://192.168.49.2:31002>  Hello World!ubuntu@ip-172-31-11-243:~/kubernetes-training$    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl edit deployments.apps helloworld-deployment  #updated image: arjunachari12/k8s-demo to image: arjunachari12/k8s-demo:2  deployment.apps/helloworld-deployment edited  ubuntu@ip-172-31-11-243:~/kubernetes-training$ curl <http://192.168.49.2:31002>  Hello World v2!ubuntu@ip-172-31-11-243:~/kubernetes-training$ | <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#rolling-back-a-deployment>    challenge: deploy new version on the image and test the new deployment. Revert back to old version |
|  | Keep watching: Kubectl get rs -w |  |
|  |  |  |
|  | Liveliness probe  apiVersion: apps/v1  kind: Deployment  metadata:    name: helloworld-deployment  spec:    replicas: 10    selector:      matchLabels:        app: helloworld    template:      metadata:        name: pod3        labels:          app: helloworld      spec:        containers:        - name: k8s-demo          image: arjunachari12/k8s-demo          ports:          - name: nodejs-port            containerPort: 3000          livenessProbe:              httpGet:                path: /                port: nodejs-port              initialDelaySeconds: 15              timeoutSeconds: 30          readinessProbe:              httpGet:                path: /                port: nodejs-port              initialDelaySeconds: 15              timeoutSeconds: 30  ---  apiVersion: v1  kind: Service  metadata:    name: helloworld-service  spec:    selector:      app: helloworld    ports:    - port: 31001      nodePort: 31002      targetPort: 3000      protocol: TCP    type: NodePort |  |
|  | apiVersion: v1  kind: Pod  metadata:  labels:  test: liveness  name: liveness-exec  spec:  containers:  - name: liveness  image: k8s.gcr.io/busybox  args:  - /bin/sh  - -c  - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600  livenessProbe:  exec:  command:  - cat  - /tmp/healthy  initialDelaySeconds: 5  periodSeconds: 5 | <https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/>    After 30 sec:  Events:  Type Reason Age From Message  ---- ------ ---- ---- -------  Normal Scheduled 41s default-scheduler Successfully assigned default/liveness-exec to minikube  Normal Pulling 40s kubelet Pulling image "k8s.gcr.io/busybox"  Normal Pulled 38s kubelet Successfully pulled image "k8s.gcr.io/busybox" in 2.010947417s  Normal Created 38s kubelet Created container liveness  Normal Started 38s kubelet Started container liveness  Warning Unhealthy 0s (x2 over 5s) kubelet Liveness probe failed: cat: can't open '/tmp/healthy': No such file or directory    kubectl get pod liveness-exec  NAME READY STATUS RESTARTS AGE  liveness-exec 1/1 Running 0 57s |
|  | Use resources in deployment file  Under containers: filed  resources:            requests:              memory: "64Mi"              cpu: "250m"            limits:              memory: "128Mi"              cpu: "500m" |  |
| Environment variables | apiVersion: v1  kind: Pod  metadata:    name: pod3    labels:      app: helloworld  spec:    containers:    - name: myapp      image: gcr.io/google-samples/node-hello:1.0      env:      - name: DEMO\_GREETING        value: "Hello from Ravi"      resources:        limits:          memory: "128Mi"          cpu: "500m" | ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl create -f variables.yml  Pod/pod3 created  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get pod  NAME READY STATUS RESTARTS AGE  pod3 1/1 Running 0 74s    kubectl exec pod3 -- printenv ===> shows all environment variables inside the container  DEMO\_GREETING=Hello from Ravi |
| Config Map to pass env vars | Configmap.yml  apiVersion: v1  kind: ConfigMap  metadata:    name: myappconfigmap  data:    DEMO\_GREETING: "Hello from Ravi"    DEMO\_GREETING1: "Hello from Kishore"    DEMO\_GREETING2: "Hello from Koppuravuri"    configmapPod.yml  apiVersion: v1  kind: Pod  metadata:    name: pod3    labels:      app: helloworld  spec:    containers:    - name: myapp      image: gcr.io/google-samples/node-hello:1.0      env:      - name: DEMO\_GREETING        valueFrom:          configMapKeyRef:            name: myappconfigmap            key: DEMO\_GREETING      resources:        limits:          memory: "128Mi"          cpu: "500m" | ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl create -f configmap.yml  configmap/myappconfigmap created    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl create -f configmapPod.yml  pod/pod3 created    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get all  NAME READY STATUS RESTARTS AGE  pod/pod3 1/1 Running 0 9s    NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 24h    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl exec pod3 -- printenv  PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  HOSTNAME=pod3  DEMO\_GREETING=Hello from Ravi |
| Secrets to pass env vars | Secret.yml  apiVersion: v1  kind: Secret  metadata:    name: helloworld-secrets  type: Opaque  data:    MYSQL\_ROOT\_PASSWORD: bXktc2VjcmV0LXB3    MYSQL\_USER: cmF2aQ==    MYSQL\_PASSWORD: cmF2aTEyMw==    MYSQL\_DATABASE: ZGIxMjM=    secretPod.yml  apiVersion: v1  kind: Pod  metadata:    name: mysql    labels:      name: myapp  spec:    containers:    - name: myapp      image: mysql      ports:        - containerPort: 3306      env:      - name: MYSQL\_ROOT\_PASSWORD        valueFrom:          secretKeyRef:            name: helloworld-secrets            key: MYSQL\_ROOT\_PASSWORD      - name: MYSQL\_USER        valueFrom:          secretKeyRef:            name: helloworld-secrets            key: MYSQL\_USER      - name: MYSQL\_PASSWORD        valueFrom:          secretKeyRef:            name: helloworld-secrets            key: MYSQL\_PASSWORD      - name: MYSQL\_DATABASE        valueFrom:          secretKeyRef:            name: helloworld-secrets            key: MYSQL\_DATABASE | kubectl create -f secret.yml  kubectl create -f secretPod.yml    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get all  NAME READY STATUS RESTARTS AGE  pod/myapp 0/1 CrashLoopBackOff 7 (4m16s ago) 15m  **pod/mysql 1/1 Running 0 2m9s**  pod/pod3 1/1 Running 0 133m    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl describe pod mysql  Environment:  MYSQL\_ROOT\_PASSWORD: <set to the key 'MYSQL\_ROOT\_PASSWORD' in secret 'helloworld-secrets'> Optional: false  MYSQL\_USER: <set to the key 'MYSQL\_USER' in secret 'helloworld-secrets'> Optional: false  MYSQL\_PASSWORD: <set to the key 'MYSQL\_PASSWORD' in secret 'helloworld-secrets'> Optional: false  MYSQL\_DATABASE: <set to the key 'MYSQL\_DATABASE' in secret 'helloworld-secrets'> Optional: false  Mounts:  /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-prmq6 (ro)  Conditions: |
|  | Kubernetes Volumes  • Volumes can be attached using different volume plugins.  node 1  kubelet  kube-proxy  Pod 1  Local  iptables  Docker  Pod2 PodN  AWS Cloud  EBS Storage  Network storage  NFS  Google Cloud  Google Disk  Microsoft Cloud  Azure Disk |  |
| Volumes | **PersistentVolume.yml**  apiVersion: v1  kind: PersistentVolume  metadata:    name: task-pv-volume    labels:      type: local  spec:    storageClassName: manual    capacity:      storage: 10Gi    accessModes:      - ReadWriteOnce    hostPath:      path: "/mnt/data" | ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl create -f persistentVolume.yml  persistentvolume/task-pv-volume created  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get pv  NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE  task-pv-volume 10Gi RWO Retain Available manual 42s  ubuntu@ip-172-31-11-243:~/kubernetes-training$ |
| PersistentVolumeClaim.yml | **PersistentVolumeClaim.yml**  apiVersion: v1  kind: PersistentVolumeClaim  metadata:    name: task-pv-claim  spec:    resources:      requests:        storage: 3Gi    storageClassName: manual    accessModes:      - ReadWriteOnce |  |
| Volume-pod.yml | **Volume-pod.yml**  apiVersion: v1  kind: Pod  metadata:    name: task-pv-pod  spec:    volumes:      - name: task-pv-storage        persistentVolumeClaim:          claimName: task-pv-claim    containers:    - name: task-pv-container      image: nginx      resources:        limits:          memory: "128Mi"          cpu: "500m"      ports:        - containerPort: 80      volumeMounts:        - mountPath: "/usr/share/nginx/html"          name: task-pv-storage | pers.  local directry  volume I  o ume  pod  containe  ece  lume laim  th usr/share/nginx |
| Ingress | Rule:  arjun.com/product  arjun.com/  O  ent  ngre  contr  arj  .com  O | minikube addons enable ingress    kubectl get ns    kubectl get ns -n ingress-nginx      ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl run nginx-pod --image=nginx  pod/nginx-pod created    Creates a pod automatically without writing any yml files  Use k edit pod nginx-pod to see the yml file of this created pod    kubectl run nginx-pod1 --image=nginx --dry-run=client -o yaml > primitive-pod.yml  --> creates a yaml file but will not run the pod |
| Logs | kubectl logs <pod-name>  If a pod has more than 1 container in it, use as below  kubectl logs pod1 -c <container-name>    kubectl logs pod1 --all-containers=true ==> shows logs of all containers |  |
| Exec commnd | ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl exec -it nginx-pod -- /bin/bash  root@nginx-pod:/# |  |
| Cluster details | kubectl config view  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl config get-contexts  CURRENT NAME CLUSTER AUTHINFO NAMESPACE  \* minikube minikube minikube default |  |
| Create all pods in a folder | kubectl create -f ../kubernetes-training/    Delete all pods in a folder kubectl delete -f ../kubernetes-training/ |  |
| Creates yaml file without running anything | kubectl create deployment my-dep --image=nginx --dry-run=client -o yaml > nginx-pod.yaml |  |
| Challenge  Imperative commands | <https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands>  Challenge:   1. 1.Run redis image 2. create service for the above pod  3. create a sample secret 4. create a deployment for redis image with replicas=3       Kubectl create pod redis-pod --image=redis --dry-run=client -o yaml > redis-pod.yaml | k run pod1 --image=redis replicas=3  k create secret generic secr1 --type=Opaque  k create deployment pod1 --image=redis |
| ingress | [10:24] arjun (Guest)  minikube addons enable ingress    [10:25] arjun (Guest)  kubectl get pods -n ingress-nginx    [10:28] arjun (Guest)  kubectl create deployment web --image=gcr.io/google-samples/hello-app:1.0        ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl expose deployment web --type=NodePort --port=8080  service/web exposed  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get svc  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 46h  web NodePort 10.106.172.226 <none> 8080:30044/TCP 9s  ubuntu@ip-172-31-11-243:~/kubernetes-training$ minikube service web --url  <http://192.168.49.2:30044>  ubuntu@ip-172-31-11-243:~/kubernetes-training$ curl <http://192.168.49.2:30044>  Hello, world!  Version: 1.0.0  Hostname: web-746c8679d4-sd7jb  ubuntu@ip-172-31-11-243:~/kubernetes-training$        ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl create deployment web2 --image=gcr.io/google-samples/hello-app:2.0  deployment.apps/web2 created  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl expose deployment web2 --type=NodePort --port=8080  service/web2 exposed  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get svc  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 46h  web NodePort 10.106.172.226 <none> 8080:30044/TCP 6m13s  web2 NodePort 10.109.67.177 <none> 8080:31141/TCP 19s  ubuntu@ip-172-31-11-243:~/kubernetes-training$ minikube service web2 --url  <http://192.168.49.2:31141>  ubuntu@ip-172-31-11-243:~/kubernetes-training$ curl <http://192.168.49.2:31141>  Hello, world!  Version: 2.0.0  Hostname: web2-5858b4c7c5-45bxg  ubuntu@ip-172-31-11-243:~/kubernetes-training$      Ingress  Nginx Ingress controller  hello-world.info /  Ingress  Rule  fhello-world.infoi  service  hello-app:I.O  web 2  service  hello-app :2.0    apiVersion: networking.k8s.io/v1  kind: Ingress  metadata:    name: myingress    annotations:      nginx.ingress.kubernetes.io/rewrite-target: /$1    labels:      name: myingress  spec:    rules:      - host: hello-world.info        http:          paths:            - pathType: Prefix              path: /              backend:                service:                  name: web                  port:                    number: 8080            - pathType: Prefix              path: /v2              backend:                service:                  name: web2                  port:                    number: 8080  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl create -f pod-ingress.yml  ingress.networking.k8s.io/myingress created    ubuntu@ip-172-31-11-243:~/kubernetes-training$ minikube ip  192.168.49.2  ubuntu@ip-172-31-11-243:~/kubernetes-training$ curl 192.168.49.2 -H 'Host: hello-world.info'  Hello, world!  Version: 1.0.0  Hostname: web-746c8679d4-sd7jb    ubuntu@ip-172-31-11-243:~/kubernetes-training$ curl 192.168.49.2/v2 -H 'Host: hello-world.info'  Hello, world!  Version: 2.0.0  Hostname: web2-5858b4c7c5-45bxg |  |
| Namespaces | ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get ns  NAME STATUS AGE  default Active 47h  ingress-nginx Active 18h  kube-node-lease Active 47h  kube-public Active 47h  kube-system Active 47h  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get po -n ingress-nginx  NAME READY STATUS RESTARTS AGE  ingress-nginx-admission-create-8gsqv 0/1 Completed 0 18h  ingress-nginx-admission-patch-q8c69 0/1 Completed 0 18h  ingress-nginx-controller-cc8496874-7d958 1/1 Running 1 (71m ago) 18h  ubuntu@ip-172-31-11-243:~/kubernetes-training$      Ns.yml  apiVersion: v1  kind: Namespace  metadata:  name: myspace  ---  apiVersion: v1  kind: ResourceQuota  metadata:  name: compute-quota  namespace: myspace  spec:  hard:  requests.cpu: "1"  requests.memory: 1Gi  limits.cpu: "2"  limits.memory: 2Gi | ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl create -f ns.yml  namespace/myspace created  resourcequota/compute-quota created    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get ns  NAME STATUS AGE  default Active 47h  ingress-nginx Active 18h  kube-node-lease Active 47h  kube-public Active 47h  kube-system Active 47h  myspace Active 81s  ravi-ns Active 8m43s  ubuntu@ip-172-31-11-243:~/kubernetes-training$    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl describe ns myspace  Name: myspace  Labels: kubernetes.io/metadata.name=myspace  Annotations: <none>  Status: Active    Resource Quotas  Name: compute-quota  Resource Used Hard  -------- --- ---  limits.cpu 0 2  limits.memory 0 2Gi  requests.cpu 0 1  requests.memory 0 1Gi    No LimitRange resource.      ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl create -f pod3.yml -n myspace  pod/pod3 created  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get pod -n myspace  NAME READY STATUS RESTARTS AGE  pod3 1/1 Running 0 35s  ubuntu@ip-172-31-11-243:~/kubernetes-training$ |
| Challenge | [11:44] arjun (Guest)  challenge: Create a new ns, attach resource quota with maximum configmap 10, secrets 20, services 10, volume 5 |  |
| Auto scaling | If cpu utilization goes beyond 60%, then create a new pod automatically  [12:06] arjun (Guest)  <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/>    [12:07] arjun (Guest)  <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/> |  |
| Big picture | Cloud  Infra  AWS  Write Ku  yaml files  Deployement - run container as pod  services - expose pod as service  Security layer  Networking layer  Config u ration  Automation script  Gatewa  SSL termi  SSL  Backend  WAF  Security layer  EKS  Kubernetes Cluster  amespace Intel-ns  Isti0  In  node 4 - GPI-J  label; ML  nodel, node 2, node3  o  fo  secrets  config map  hpa - autoscaling pod  monitoring - FluendD or  Prometheus must run  in all nodes, so we run  it as DaemonSet  Statefulset for  cassandra  Volumes to store  container data  use nodeselector to run  data science workload  on node4  Also use Affinity/Anti-  affinity for complex  placement  service  service  irtoring  UI  pod o  cassand  Taint" ole ration  Default  namespa  intel.com  CL NT  Se  serv  Ice  service  S3 bukcet  store  -mages udiO  docs  Storage class  EBS  Volu  Secret Mana  SQL se  AWS region  - US West  1  On-premises  API  DB |  |
| Jobs & Cronjobs | <https://kubernetes.io/docs/concepts/workloads/controllers/job/>  <https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/>    Cronjob.yml  apiVersion: batch/v1  kind: CronJob  metadata:    name: hello  spec:    schedule: "37 9 \* \* \*"    jobTemplate:      spec:        template:          spec:            containers:            - name: pi              image: perl              command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]            restartPolicy: OnFailure | ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl create -f cronjob.yml  cronjob.batch/hello created    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get cronjob  NAME SCHEDULE SUSPEND ACTIVE LAST SCHEDULE AGE  hello 37 9 \* \* \* False 0 3m11s 5m38s    ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get pods -w  NAME READY STATUS RESTARTS AGE  php-apache-7656945b6b-kxsmc 1/1 Running 0 174m  web-746c8679d4-sd7jb 1/1 Running 0 4h38m  web2-5858b4c7c5-45bxg 1/1 Running 0 4h31m  hello-27510337-vsr5f 0/1 Pending 0 0s  hello-27510337-vsr5f 0/1 Pending 0 0s  hello-27510337-vsr5f 0/1 ContainerCreating 0 0s  hello-27510337-vsr5f 1/1 Running 0 23s  hello-27510337-vsr5f 0/1 Completed 0 30s  hello-27510337-vsr5f 0/1 Completed 0 31s |
| Roles | Role.yaml  apiVersion: rbac.authorization.k8s.io/v1  kind: Role  metadata:    namespace: default    name: pod-writer  rules:  - apiGroups: [""]    resources: ["pods"]    verbs: ["get", "watch", "list", "create", "update", "patch", "delete"]  ---  apiVersion: rbac.authorization.k8s.io/v1  kind: RoleBinding  metadata:    name: write-pods    namespace: default  subjects:  - kind: User    name: user1    apiGroup: rbac.authorization.k8s.io  roleRef:    kind: Role    name: pod-writer    apiGroup: rbac.authorization.k8s.io  ---  apiVersion: rbac.authorization.k8s.io/v1  kind: ClusterRoleBinding  metadata:    name: admin-user  subjects:  - kind: User    name: user1    apiGroup: rbac.authorization.k8s.io  roleRef:    apiGroup: rbac.authorization.k8s.io    kind: ClusterRole    name: cluster-admin | ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl apply -f role.yaml  role.rbac.authorization.k8s.io/pod-writer unchanged  rolebinding.rbac.authorization.k8s.io/write-pods created  clusterrolebinding.rbac.authorization.k8s.io/admin-user unchanged  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get role  NAME CREATED AT  pod-writer 2022-04-22T09:59:20Z  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get rolebindings.rbac.authorization.k8s.io  NAME ROLE AGE  write-pods Role/pod-writer 33s  ubuntu@ip-172-31-11-243:~/kubernetes-training$ kubectl get clusterrolebinding  NAME ROLE AGE  admin-user ClusterRole/cluster-admin 107s  cluster-admin ClusterRole/cluster-admin |
| Katacoda | [Launch Single Node Kubernetes Cluster | Kubernetes | Katacoda](https://www.katacoda.com/courses/kubernetes/launch-single-node-cluster) |  |
| Alerts |  | Grafana is good UI tool |
| Prometheus for alerts | <https://www.katacoda.com/courses/prometheus/getting-started> |  |
| Github | [ravikishorek1/kubernetes (github.com)](https://github.com/ravikishorek1/kubernetes)    ravikishorek1  Ravi@Marks |  |
|  |  |  |