CAP2 Kubernates Architecture

Bash Completion

> sudo apt-get install bash-completion vim -y

> source <(kubectl completion bash)

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

UnTaint

> kubectl describe nodes | grep -i taint

Taints: node-role.kubernetes.io/control-plane:NoSchedule

Taints: <none>

> kubectl taint nodes --all node-role.kubernetes.io/control-planenode/

|  |  |  |
| --- | --- | --- |
| Basic.yaml | Basicservice.yaml | MultiContainer Pod |
| apiVersion: v1  kind: Pod  metadata:  name: basicpod  labels:  type: webserver  spec:  containers:  - name: webcont  image: nginx  ports:  - containerPort: 80 | apiVersion: v1  kind: Service  metadata:  name: basicservice  spec:  selector:  type: webserver  type: NodePort  ports:  - protocol: TCP  port: 80 | apiVersion: v1  kind: Pod  metadata:  name: basicpod  labels:  type: webserver  spec:  containers:  - name: webcont  image: nginx  ports:  - containerPort: 80  - name: fdlogger  image: fluentd |

Crear un deploy simple:

> kubectl create deployment firstpod --image=nginx

CAP3 Build

> sudo apt-get -y install python3

> which python3

/usr/bin/python3

|  |  |
| --- | --- |
| Simple.py | Dockerfile |
| #!/usr/bin/python3  import time  import socket  while True :  host = socket.gethostname()  date = time.strftime("%Y-%m-%d %H:%M:%S")  now = str(date)  f = open("date.out", "a" )  f.write(now + " " + host + " \n")  f.close()  time.sleep(5) | FROM docker.io/library/python:3  ADD simple.py /  ## RUN pip install pystrich  CMD [ "python", "./simple.py" ] |

> sudo podman build -t simpleapp .

> sudo podman images

> sudo podman run localhost/simpleapp

Configurar repo local

> find $HOME -name easyregistry.yaml

> create -f easyregistry.yaml

> kubectl get svc | grep registry

> curl 10.97.40.62:5000/v2/\_catalog

> find $HOME -name local-repo-setup.sh

> . $HOME/local-repo-setup.sh

> sudo podman pull docker.io/library/alpine

> sudo podman tag alpine $repo/tagtest

> sudo podman push $repo/tagtest

> curl $repo/v2/\_catalog

Taguear el build python

> sudo podman tag simpleapp $repo/simpleapp

> sudo podman push $repo/simpleapp

> curl $repo/v2/\_catalog

> kubectl create deployment try1 --image=$repo/simpleapp

> kubectl scale deployment try1 --replicas=6

> sudo crictl ps | grep simple

> kubectl get deployment try1 -o yaml > simpleapp.yaml

> kubectl delete deployment try1

Probes:

|  |  |
| --- | --- |
| Simpleapp.yaml | Simpleapp.yaml (readiness y liveness probe) |
| apiVersion: apps/v1  kind: Deployment  metadata:  labels:  app: try  name: try  namespace: default  spec:  progressDeadlineSeconds: 600  replicas: 6  selector:  matchLabels:  app: try  spec:  containers:  - image: 10.97.40.62:5000/simpleapp  name: simpleapp  readinessProbe:  periodSeconds: 5  exec:  command:  - cat  - /tmp/healthy  resources: {}  restartPolicy: Always  securityContext: {} | apiVersion: apps/v1  kind: Deployment  metadata:  labels:  app: try  name: try  namespace: default  spec:  progressDeadlineSeconds: 600  replicas: 6  selector:  matchLabels:  app: try  spec:  containers:  - name: simpleapp  image: 10.97.40.62:5000/simpleapp  **readinessProbe:**  periodSeconds: 5  exec:  command:  - cat  - /tmp/healthy  resources: {}  - name: goproxy  image: registry.k8s.io/goproxy:0.1  ports:  - containerPort: 8080  **readinessProbe:**  tcpSocket:  port: 8080  initialDelaySeconds: 5  periodSeconds: 10  **livenessProbe:** #<-----  tcpSocket:  port: 8080  initialDelaySeconds: 15  periodSeconds: 20 #<-- ....to here  dnsPolicy: ClusterFirst  restartPolicy: Always  securityContext: {} |

> for name in $(kubectl get pod -l app=try1 -o name)

> do

> kubectl exec $name -c simpleapp -- touch /tmp/healthy

> done

> kubectl describe pod try1-76cc5ffcc6-tx4dz | grep -E 'State|Ready'

CAP 4 Design

*Container Network Interface (CNI) (kubenet, Cilium)*

Job and Cronjob

|  |  |
| --- | --- |
| Job (CAP) | Cronjob |
| apiVersion: batch/v1  kind: Job  metadata:  name: sleepy  spec:  completions: 5  parallelism: 2  activeDeadlineSeconds: 15  template:  spec:  containers:  - name: resting  image: busybox  command: ["/bin/sleep"]  args: ["3"]  restartPolicy: Never | apiVersion: batch/v1  kind: CronJob #<-- Change this line  metadata:  name: sleepy  spec:  schedule: "\*/2 \* \* \* \*" #<-- Add Linux style cronjob syntax  jobTemplate: #<-- New jobTemplate and spec  spec:  template:  spec:  activeDeadlineSeconds: 10  containers:  - name: resting  image: busybox  command: ["/bin/sleep"]  args: ["30"]  restartPolicy: Never |

Using Labels

> kubectl create deployment design2 --image=nginx

> kubectl get deployments.apps design2 -o wide

> kubectl get -l app=design2 pod

> kubectl get --selector app=design2 pod -o yaml

Resource Limits

|  |  |
| --- | --- |
| *stress.yaml* | Simple Init Container |
| apiVersion: apps/v1  kind: Deployment  metadata:  annotations:  deployment.kubernetes.io/revision: "1"  generation: 1  name: stressmeout  spec:  replicas: 1  selector:  matchLabels:  run: stressmeout  template:  metadata:  labels:  run: stressmeout  spec:  nodeSelector:  kubernetes.io/hostname: worker #<<-- EDIT to be YOUR worker hostname  containers:  - image: vish/stress  imagePullPolicy: Always  name: stressmeout  resources:  limits:  cpu: “1”  memory: “500Mi”  args:  - -cpus  - "2"  - -mem-total  - "1950Mi"  - -mem-alloc-size  - "100Mi"  - -mem-alloc-sleep  - "1s"  restartPolicy: Always  schedulerName: default-scheduler  securityContext: {}  terminationGracePeriodSeconds: 30 | init-tester.yaml  apiVersion: v1  kind: Pod  metadata:  name: init-tester  labels:  app: inittest  spec:  containers:  - name: webservice  image: nginx  initContainers:  - name: failed  image: busybox  command: [/bin/true] |

Crd: custom resource definitions: kubectl get crd

CAP 5 Deployment configuration (ConfigMap) Crear el coonfigmap por línea comando.

kubectl create configmap colors --from-literal=text=black --from-file=./favorite --from-file=./primary/

|  |  |
| --- | --- |
| *simpleapp.yaml* |  |
| apiVersion: apps/v1  kind: Deployment  metadata:  labels:  app: try  name: try  namespace: default  spec:  replicas: 6  selector:  matchLabels:  app: try  spec:  containers:  - image: 10.97.40.62:5000/simpleapp  name: simpleapp  env:  - name: ilike  valueFrom  configMapKeyRef:  name: colors  key: favorite  envFrom:  - configMapRef  name: colors  imagePullPolicy: Always | kubectl exec -it try1-d4fbf76fd-46pkb -- /bin/bash -c 'env'  <- obtener configmap con una key  <- obtener configmap completo |

|  |  |
| --- | --- |
| Volume mount from configmap | nfs |
| apiVersion: apps/v1  kind: Deployment  metadata:  labels:  app: try  name: try  namespace: default  spec:  replicas: 6  selector:  matchLabels:  app: try  spec:  containers:  - image: 10.97.40.62:5000/simpleapp  name: simpleapp  volumeMounts:  - mountPath: /etc/cars  name: car-vol  env:  - name: ilike  valueFrom  configMapKeyRef:  name: colors  key: favorite  envFrom:  - configMapRef  name: colors  imagePullPolicy: Always  volumes:  - name: car-vol  configMap:  defaultMode: 420  name: fast-car | Crear el FS  cp> bash $HOME/CreateNFS.sh  wk> sudo apt-get -y install nfs-common nfs-kernel-server  wk> showmount -e cp #<-- Edit to be first node’s name or IP  wk> sudo mount cp:/opt/sfw /mnt  wk> ls -l /mnt  car-map.yaml  apiVersion: v1  kind: ConfigMap  metadata:  name: fast-car  namespace: default  data:  car.make: Ford  car.model: Mustang  car.trim: Shelby |

Persistant volume

|  |  |
| --- | --- |
| Simpleapp.yaml con pvc | PVol.yaml |
| apiVersion: apps/v1  kind: Deployment  metadata:  labels:  app: try  name: try  namespace: default  spec:  replicas: 6  selector:  matchLabels:  app: try  spec:  containers:  - image: 10.97.40.62:5000/simpleapp  name: simpleapp  volumeMounts:  - name: car-vol  mountPath: /etc/cars  - name: nfs-vol  mountPath: /opt  volumes:  - name: car-vol  configMap:  defaultMode: 420  name: fast-car  - name: nfs-vol  persistentVolumeClaim:  claimName: pvc-one | apiVersion: v1  kind: PersistentVolume  metadata:  name: pvvol-1  spec:  capacity:  storage: 1Gi  accessModes:  - ReadWriteMany  persistentVolumeReclaimPolicy: Retain  nfs:  path: /opt/sfw  server: cp #<-- Edit to match IP  readOnly: false  --------------------------------------------------------------------  Pvc.yaml  apiVersion: v1  kind: PersistentVolumeClaim  metadata:  name: pvc-one  spec:  accessModes:  - ReadWriteMany  resources:  requests:  storage: 200Mi |

Usando ConfigMaps Configurations

|  |  |  |
| --- | --- | --- |
| Basic.yaml. (POD) | Weblog-pvc.yaml. (PVC). SPEC-SAR | Weblog-pv.yaml. (PV). SPEC-SACH |
| apiVersion: v1  kind: Pod  metadata:  name: basicpod  labels:  type: webserver  spec:  containers:  - name: webcont  image: nginx  ports:  - containerPort: 80  volumeMounts:  - mountPath: "/var/log/nginx/"  name: weblog-pv-storage  - name: fdlogger  image: fluentd  volumeMounts:  - mountPath: "/var/log"  name: weblog-pv-storage  volumes:  - name: weblog-pv-storage  presistentVolumeClaim:  claimName: **weblog-pv-claim** | apiVersion: v1  kind: PersistentVolumeClaim  metadata:  name: **weblog-pv-claim**  spec:  storageClassName: manual  accessModes:  - ReadWriteOnce  resources:  requests:  storage: 100Mi | apiVersion: v1  kind: PersistentVolume  metadata:  name: weblog-pv-volume  spec:  storageClassName: manual  accessModes:  - ReadWriteOnce  capacity:  storage: 100Mi  hostPath:  path: "/tmp/weblog" |

> kubectl exec -c webcont -it basicpod -- /bin/bash

> ls -l /var/log/nginx/access.log

> tail -f /var/log/nginx/access.log

|  |  |  |
| --- | --- | --- |
| Basic.yaml | Weblog-configmap.yaml |  |
| apiVersion: v1  kind: Pod  metadata:  name: basicpod  labels:  type: webserver  spec:  containers:  - name: webcont  image: nginx  ports:  - containerPort: 80  volumeMounts:  - mountPath: "/var/log/nginx/"  name: weblog-pv-storage  - name: fdlogger  image: fluentd  env:  - name: FLUENTD\_OPT  value: -c /etc/fluentd-config/fluent.conf  volumeMounts:  - name: weblog-pv-storage  mountPath: "/var/log"  - name: log-config  mountPath: "/etc/fluentd-config"  volumes:  - name: weblog-pv-storage  presistentVolumeClaim:  claimName: weblog-pv-claim  - name: log-config  configMap:  name: fluentd-config | apiVersion: v1  kind: ConfigMap  metadata:  name: fluentd-config  namespace: default  data:  fluent.conf: |  <source>  @type tail  format none  path /var/log/access.log  tag count.format1  </source>  <match \*.\*\*>  @type stdout  id stdout\_output  </match> |  |

> kubectl logs basicpod webcont

> kubectl logs basicpod fdlogger

Rolling updates and RollBack

> sudo podman build -t simpleapp .

> sudo podman images | grep simple

> sudo podman tag simpleapp $repo/simpleapp:v2

> sudo podman push $repo/simpleapp:v2

> sudo podman pull $repo/simpleapp

> sudo podman pull $repo/simpleapp:v2

> kubectl edit deployment try1

containers:

- image: 10.105.119:236:5000/simpleapp:v2 #<--edit

> kubectl get events

> kubectl describe pod try...

> kubectl rollout history deployment try1

> kubectl rollout history deployment try1 --revision=1 > one.out

> kubectl rollout history deployment try1 --revision=2 > one.out

> diff one.out two.out

> kubectl rollout undo --dry-run=client deployment/try1 //Para ver la plantilla del deploy antes de ejecutar.

> kubectl get pods

> kubectl rollout undo deployment try1 --to-revision=1 //Ejecuta la reversa a version 1

Para actualizar una imagen a version 2

kubectl set image deployments misimplez misimplez=$repo/misimplez:v2

CAP 6 Entendiendo seguridad

|  |  |
| --- | --- |
| Second.yaml |  |
| apiVersion: v1  kind: Pod  metadata:  name: secondapp  spec:  securityContext:  runAsUser: 1000  containers:  - name: busy  image: busybox  command:  - sleep  - "3600"  securityContext:  runAsUser: 2000  allowPrivilegeEscalation: false  capabilities:  add: ["NET\_ADMIN","SYS\_TIME"] | > kubectl exec -it secondapp -- sh  > ps aux  > grep Cap /proc/1/status  > exit  > capsh --decode=00000000000005fb |

Crear y consumir secrets

> echo LFTr@1n | base64

|  |  |
| --- | --- |
| Second.yaml | Secret.yaml |
| apiVersion: v1  kind: Pod  metadata:  name: secondapp  spec:  securityContext:  runAsUser: 1000  containers:  - name: busy  image: busybox  command:  - sleep  - "3600"  securityContext:  runAsUser: 2000  allowPrivilegeEscalation: false  capabilities:  add: ["NET\_ADMIN","SYS\_TIME"]  volumeMounts:  - name: mysql  mountPath: /mysqlpassword  volumes:  - name: mysql  secret:  secretName: lfsecret | apiVersion: v1  kind: Secret  metadata:  name: lfsecret  data:  password: TEZUckAxbgo= |

> kubectl exec -ti secondapp -- /bin/sh

> cat /mysqlpassword/password

> cd /mysqlpassword/

> ls -ltra

Trabajando con serviceaccount

|  |  |  |
| --- | --- | --- |
|  | Secod.yaml |  |
|  | apiVersion: v1  kind: Pod  metadata:  name: secondapp  spec:  serviceAccountName: secret-access-sa  securityContext:  runAsUser: 1000  containers:  - name: busy  image: busybox  command:  - sleep  - "3600"  securityContext:  runAsUser: 2000  allowPrivilegeEscalation: false |  |
| ServiceAccount.yaml | ClusterRole.yaml | RoleBinding.yaml |
| apiVersion: v1  kind: ServiceAccount  metadata:  name: secret-access-sa | apiVersion: rbac.authorization.k8s.io/v1  kind: ClusterRole  metadata:  name: secret-access-cr  rules:  - apiGroups:  - ""  resources:  - secrets  verbs:  - get  - list | apiVersion: rbac.authorization.k8s.io/v1  kind: RoleBinding  metadata:  name: secret-rb  subjects:  - kind: ServiceAccount  name: secret-access-sa  roleRef:  kind: ClusterRole  name: secret-access-cr  apiGroup: rbac.authorization.k8s.io |

Implementando Política de Seguridad

|  |  |
| --- | --- |
| allclosed.yaml (todo denegado) | allclosed.yaml (solo aceptar ingress por puerto 80) |
| apiVersion: networking.k8s.io/v1  kind: NetworkPolicy  metadata:  name: deny-default  spec:  podSelector: {}  policyTypes:  - Ingress  - Egress | apiVersion: networking.k8s.io/v1  kind: NetworkPolicy  metadata:  name: deny-default  spec:  podSelector: {}  policyTypes:  - Ingress  ingress:  - from:  - podSelector: {}  ports:  - port: 80  protocol: TCP  # - Egress |

|  |  |
| --- | --- |
| Second.yaml |  |
| apiVersion: v1  kind: Pod  metadata:  name: secondapp  labels:  example: second  spec:  serviceAccountName: secret-access-sa  containers:  - name: webserver  image: nginx  - name: busy  image: busybox  command:  - sleep  - "3600"  securityContext:  runAsUser: 2000  allowPrivilegeEscalation: false | > kubectl create service nodeport secondapp --tcp=80  > kubectl edit svc secondapp  apiVersion: v1  kind: Service  metadata:  name: secondapp  spec:  ports:  - name: "80"  nodePort: 32000  port: 80  protocol: TCP  targetPort: 80  selector:  example: second  sessionAffinity: None  status:  loadBalancer: {} |

kubectl replace -f $HOME/app2/allclosed.yaml

kubectl run -it test --rm=true --image alpine -- ping -c5 192.168.55.91

CAP7 Exposing Applications

|  |  |  |
| --- | --- | --- |
| newservice.yaml |  |  |
| apiVersion: v1  kind: Service  metadata:  name: secondapp  spec:  ports:  - protocol: TCP  port: 80  nodePort: 32000  type: LoadBalancer  selector:  example: second  sessionAffinity: None  status:  loadBalancer: {} | > kubectl create deployment newserver --image=httpd  > curl http://35.184.219.5:32000  > kubectl exec -it -c busy secondapp -- sh  > nc -vz 127.0.0.1 80  127.0.0.1 (127.0.0.1:80) open  > nc -vz www.linux.com 80  www.linux.com (151.101.185.5:80) open  > exit  > kubectl exec -it secondapp -c busy -- sh  > nslookup secondapp  > nslookup registry  > nslookup 10.96.0.10  > nslookup kube-dns  > nslookup kube-dns.kube-system.svc.cluster.local  > exit  Eje  > kubectl create ns multitenant  > kubectl -n multitenant create deployment mainapp --image=nginx  > kubectl -n multitenant expose deployment mainapp --name=shopping \  --type=NodePort --port=80  > kubectl exec -it secondapp -c busy -- sh  > nslookup shopping  > nslookup shopping.multitenant  > nslookup shopping.multitenant.svc.cluster.local  > wget shopping  > wget shopping.multitenant  > wget -O - shopping.multitenant |  |

Service Mesh

* Inyectar un deploy a serviceMesh

kubectl -n multitenant get deploy mainapp -o yaml | linkerd inject - | kubectl apply -f -

Helm (install ingx ingress controller)

> helm search hub ingress

> helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

> helm repo update

> helm fetch ingress-nginx/ingress-nginx –untar

> cd ingress-nginx

> ls

> vim values.yaml

> editar values.yaml (*kind: DaemonSet)*

> helm install myingress .

> kubectl get ingress --all-namespaces

> kubectl --namespace default get services -o wide myingress-ingress-nginx-controller

> kubectl get pod --all-namespaces |grep nginx

> vim ingress.yaml

Comandos helm de la prueba:

* helm repo add bitnami <https://charts.bitnami.com/bitnami> /agrega un repo
* Helm repo list //Ver los repos seteados
* Helm repo update //Actualiza los repos
* Helm search repo nginx //Busca nginx en el repo
* Helm -n mercury install mideployhelm2 bitnami/apache –set replicaCount=2 //nuevo deploy
* helm -n mercury ls -a //listar lo instalado en namespace mercury
* Helm -n mercury upgrade mideployhelm bitnami/nginx
* helm -n mercury uninstall mideployhelm

|  |  |
| --- | --- |
| Ingress.yaml (ahora Podemos agregar reglas al ingress) |  |
| apiVersion: networking.k8s.io/v1  kind: Ingress  metadata:  name: ingress-test  annotations:  nginx.ingress.kubernetes.io/service-upstream: "true"  namespace: default  spec:  ingressClassName: nginx  rules:  - host: www.example.com  http:  paths:  - backend:  service:  name: secondapp  port:  number: 80  path: /  pathType: ImplementationSpecific |  |

> kubectl get svc |grep ingress

> curl -H "Host: www.example.com" http://10.104.227.79

> kubectl create -f ingress.yaml

> kubectl get ingress

> kubectl get pod -o wide |grep myingress

> curl 192.168.219.118 //404

> kubectl get svc |grep ingress

> curl 10.104.227.79 //404

> curl -H "Host: www.example.com" http://10.104.227.79

> kubectl get ds myingress-ingress-nginx-controller -o yaml | linkerd inject --ingress - | kubectl apply -f -

Got to dashboard and select top> default namespace> daemonset/myingress-nginx...

> kubectl create deployment thirdpage --image=nginx

> kubectl label pod thirdpage-<tab> example=third

> expose deployment thirdpage --port=80 --type=NodePort

> kubectl exec -it thirdpage-<Tab> -- /bin/bash

> vim /usr/share/nginx/html/index.html //<title>Third Page</title> #<-- Edit this line

> exit

> kubectl edit ingress ingress-test

|  |  |
| --- | --- |
| Ingress-test.yaml |  |
| apiVersion: networking.k8s.io/v1  kind: Ingress  metadata:  name: ingress-test  annotations:  nginx.ingress.kubernetes.io/service-upstream: "true"  namespace: default  spec:  ingressClassName: nginx  rules:  - host: thirdpage.org  http:  paths:  - backend:  service:  name: thirdpage  port:  number: 80  path: /  pathType: ImplementationSpecific  - host: www.example.com  http:  paths:  - backend:  service:  name: secondapp  port:  number: 80  path: /  pathType: ImplementationSpecific |  |

[09:56] Carlos Eduardo Campuzano Pacheco

nodepool: npmicroservicios01

Revisar link antes de la prueba (requisitos del sistema)

<https://docs.linuxfoundation.org/tc-docs/certification/lf-handbook2/candidate-requirements>

<https://syscheck.bridge.psiexams.com/>

<https://docs.linuxfoundation.org/tc-docs/certification/lf-handbook2/taking-the-exam>