Concepts

YAML File

myapp.yaml:

```
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo
  namespace: mem-example
spec:
  containers:
  - name: memory-demo-ctr
    image: polinux/stress
    resources:
      requests:
        memory: "100Mi"
      limits:
        memory: "200Mi"
    command: ["stress"]
    args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]
```

```
# Create a pod by applying a yaml file
kubectl apply -f myapp.yaml

# □□□□□□□□□□ YAML □□□□
kubectl create deploy nginx-deployment --image=nginx --dry-run=client -o yaml
```

Get the complete manifest of the deployed Kubernetes object.

```
kubectl get deploy nginx -n nginx-demo -o yaml
kubectl get svc <service-name> -n <name-space> -o yaml
kubectl get deployment <deployment-name> -n <name-space> -o yaml
```

Kubectl Configuration

```
kubectl version
kubectl cluster-info
kubectl config view
kubectl config view --minify --raw
```

Namespace (ns)

- Namespace ????????????? subdomain???????????????? 253 ?????????????? hyphen, dot ???

```
# Create a namespace mem-example
kubectl create namespace mem-example
kubectl get ns
```

Pod (po)

```
# Create a pod memory-demo
kubectl apply -f myapp.yaml

# Check the pods
kubectl get pods
kubectl get all -A
kubectl get pod memory-demo --namespace=mem-example
kubectl get pod memory-demo --output=yaml --namespace=mem-example
kubectl top pod memory-demo --namespace=mem-example
kubectl describe pod memory-demo --namespace=mem-example
# Get a list of pods and the node they run on
kubectl get po -o=custom-columns=NODE:.spec.nodeName,NAME:.metadata.name
# Get the name of the containers of a running pod
kubectl get pod MYPOD -o 'jsonpath={.spec.containers[*].name}'

# Delete a pod
kubectl delete pod memory-demo --namespace=mem-example

# Run Shell in the Pod
kubectl exec -it <pod-name> -n <namespace> -- bash
```

```
# Attach the pod
kubectl attach <pod-name> -n <namespace> -i

# Force to delete a pod
kubectl delete pod <pod-name> --force --grace-period=0

# Copy a file from a pod to a local file
kubectl cp <namespace>/<pod>:<file_path> <local_file_path>
```

Deployment (deploy)

```
kubectl get deployments
kubectl rollout status deployment/nginx-deployment
kubectl describe deployment <deployment-name>

# Update a new image
kubectl set image deployment/nginx-deployment nginx=nginx:sometag

# Scale a deployment
kubectl scale deployment deployment --replicas=X
```

Delete a deployment

> NOTE: ?? deployment ?????????????? container, PersistentVolumes, image, kubernetes secrets ???????????

```
# Delete the specified deployment
kubectl delete deployment <deployment name> --namespace <namespace name>

# Delete all deployments for the specified namespace
kubectl delete deployment --all --namespace=<namespace name>

# Using yaml file to delete the deployment
kubectl delete -f blog-deployment-1.yaml -f blog-deployment-2.yaml

# Using label selector
kubectl get deployment -l app=my-app
kubectl get all --selector app=[app-label]
```

```
kubectl delete deployment -l app=my-app
kubectl delete all --selector app=[app-label]

# Force-delete a hung deployment
kubectl delete deployment <deployment-name> --grace-period=0 --force
kubectl delete replicaset -l app=<label>
kubectl delete pods -l app=<label>
```

Deployment with health-check

- initialDelaySeconds:
  ??? service ?????????????? health check
- periodSeconds: ?????????????? 10?
- timeoutSeconds: ????
- successThreshold: ?????????????? service ?????
- failureThreshold: ???????

```
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-deployment
...
    spec:
      containers:
      - name: webapp
        image: zxcvbnius/docker-demo
        ports:
        - name: webapp-port
          containerPort: 3000
        livenessProbe:
          httpGet:
            path: /
            port: webapp-port
          initialDelaySeconds: 15
          periodSeconds: 15
          timeoutSeconds: 30
          successThreshold: 1
          failureThreshold: 3
```

Secret

- ????? yaml ???????? base64 ?????

```
# Create a secret as environment variables
kubectl create secret generic [secret-name] \
  --from-literal=APP_API_KEY='your-api-key' \
  --from-literal=EXTERNAL_SERVICE_API_KEY='your-external-key' \
  -n [your-namespace]
```

Pull Docker Images from Private Docker Registries

```
# Create
# Usage: kubectl create secret docker-registry [secret-name] \
#  --docker-server:[address] \
#  --docker-username=[username] \
#  --docker-password=[password] \
#  -n [namespace]

# For Docker Hub Registry
kubectl create secret docker-registry dockerhub-pull-secret \
  --docker-server=docker.io \
  --docker-username=yourusername \
  --docker-password='Your-Personal-Key' \
  -n my-devops-prod
```

Check the secret

```
kubectl get secret [secret-name] -n [namespace] -o yaml


# get the value of a secret (if you have the base64 command available)
kubectl -n mynamespace get secret MYSECRET -o 'jsonpath={.data.DB_PASSWORD}' | base64 -d
```

??? base64

```
# String to Base64
echo -n 'ThisIsSecretStrings' | base64
```

```
# Base64 to String
echo -n 'VGhpc0lzU2VjcmV0U3RyaW5ncw==' | base64 -d
```

secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret
type: Opaque
data:
  db-password: cm9vdHBhc3M=
```

deployment.yaml

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: mysql-server
...
    spec:
      containers:
      - name: mysql
        image: mysql:5.7
        args:
          - "--ignore-db-dir=lost+found"
        ports:
        - name: mysql-port
          containerPort: 3306
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-secret
              key: db-password
        volumeMounts:
...
```

secret2.yaml (???? base64)

```
apiVersion: v1
kind: Secret
metadata:
  name: tailscale-auth
stringData:
  TS_AUTHKEY: tskey-0123456789abcdef
```

basicauth-secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-basic-auth
type: kubernetes.io/basic-auth
stringData:
  username: admin # required field for kubernetes.io/basic-auth
  password: t0p-Secret # required field for kubernetes.io/basic-auth
```

Volume

emptyDir

- ???????`/var/lib/kubelet/pods/<pod-uid>`
- pod ???? containers ??????????
- ?? pod ??volume ?????????

```
apiVersion: v1
kind: Pod
metadata:
  name: empty-dir-pod
spec:
  containers:
    - name: alpine
      image: alpine
      args:
        - sleep
        - '120'
      volumeMounts:
        - name: pod-storage
          mountPath: /data/
```

```
  volumes:
    - name: pod-storage
      emptyDir:
        sizeLimit: 500Mi
```

PVC

PersistentVolumeClaim

- pod/deployment ??????? PVC/PV ??????
- ?????????? PVC ?? PV

```
kubectl get pvc -n <namespace>


kubectl delete pvc <pvc-name> -n <namespace>
```

?????? Plugin: hostpath-storage

```
microk8s enable hostpath-storage
microk8s status


❯ kubectl get storageclass
NAME                          PROVISIONER            RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
microk8s-hostpath (default)   microk8s.io/hostpath   Delete          WaitForFirstConsumer
false                  17m
```

pvc.yaml:

- PVC ????????? *Pending*??????????????? *Bound*?
- PersistentVolume ???????????????

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ollama-pvc
  namespace: ollama
spec:
  accessModes:
    - ReadWriteOnce
  resources:
```

```
    requests:
      storage: 3Gi
```

?????? node ??????

- ???? PV ? PVC ??????????????

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: ollama-pv
  labels:
    type: local
spec:
  storageClassName: localvol-ollama
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/k8svol/ollama"

---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ollama-pvc
  namespace: ollama
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
  storageClassName: localvol-ollama
```

Label

?? Labels?

- app: myapp
- release: stable, release: qa
- env: dev, env: prod
- tier: backend, tier: frontend
- depart: enginerring, depart: marketing

```
# □□□□□□
kubectl label pods <pod-name> <label-key>=<label-value>


# Show labels
kubectl get pods  --show-labels -n <namespace>


# Remove an existed label 'app=XXXX'
kubectl label pods <pod-name> app-
```

Annotations: ??????

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
  labels:
    app: webserver
    tier: backend
  annotations:
    version: latest
    release_date: 2017/12/28
    contact: zxcvbnius@gmail.com
spec:
  containers:
...
```

Service (svc)

- nodePort range: 3000 ~ 32767

```
# Check service
kubectl get svc -n <name-space>
```

```
# Create a service to expose the port of a pod
kubectl expose pod <pod-name> --port=<pod-port> --name=<service-name> -n <name-space>

# Create s service to expose a random port of a pod/deploy
kubectl expose pod <pod-name> --type=NodePort --name=<service-name> -n <name-space>
kubectl export deploy <deploy-name> --type=NodePort --name=<service-name> -n <name-space>

# Forward a port from a service to a local port
kubectl port-forward svc/<service-name> <local-port>:<remote-port>
```

ConfigMap

- ?? ConfigMap ???????? configuration
  ?? webserver config file, Nginx config file
- ???? container ????????????? Config
- ??????? configuration
- ConfigMap v.s Secret: ???????????? Secret ??? Secret ??????? Base64 ?????????? API ?? database ????????????????????? ConfigMap??????? port number ?? Redis ? config file?

```
# Create from a file
# kubectl create configmap <configmap-name> --from-file=/path/to/file
kubectl create configmap redis-config --from-file=my-redis.conf
kubectl create configmap nginx-conf --from-file=./my-nginx.conf

kubectl get configmap
kubectl describe configmap <configmap-name>

# Create with a command
# Usgae: kubectl create configmap <configmap-name> --from-literal=<key>=<value>
kubectl create configmap mysql-host --from-literal=ip=127.0.0.1
```

Apply to pod

```
apiVersion: v1
kind: Pod
metadata:
  name: apiserver
  labels:
    app: webserver
    tier: backend
```

```
spec:
  containers:
  - name: nodejs-app
    image: zxcvbnius/docker-demo
    ports:
    - containerPort: 3000
  - name: nginx
    image: nginx:1.13
    ports:
    - containerPort: 80
    volumeMounts:
    - name: nginx-conf-volume
      mountPath: /etc/nginx/conf.d
  volumes:
  - name: nginx-conf-volume
    configMap:
      name: nginx-conf
      items:
      - key: my-nginx.conf
        path: my-nginx.conf
```

ResourceQuota

??? namespace ?? CPU/RAM ??

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: quota-cpu-mem
spec:
  hard:
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
```

????????? secrets, services

```
apiVersion: v1
kind: ResourceQuota
```

```
metadata:
  name: object-quotas
  namespace: hellospace
spec:
  hard:
    services: "2"
    services.loadbalancers: "1"
    secrets: "1"
    configmaps: "1"
    replicationcontrollers: "10"
```

Apply to the namespace

```
kubectl apply -f quota-cpu-mem.yaml -n <namespace>
```

Check the resourcequota

```
kubectl get resourcequotas -n <namespace>

kubectl describe resourcequotas <resourcequotas-name> -n <namespace>
```

Node

```
# List nodes
kubectl get nodes
kubectl get nodes,pods,deploy -o wide

# Node □□
kubectl drain <node-name>
kubectl drain <node-name> --force

# Node □□
kubectl uncordon <node-name>
```

Logs

```
kubectl logs -f deployment/<deployment-name> -n <name-space>
kubectl logs -f pod/<pod-name> -n <name-space>
kubectl logs -l app=xyz
```

???? (env)

```
kubectl get deployment your-app-name -n your-namespace-prod -o yaml | grep
EXTERNAL_SERVICE_URL


kubectl set env <resource>/<resource-name> --list
```

Revision #11
Created 2026-02-06 16:09:31 CST by A-Lang (Admin)
Updated 2026-03-04 14:59:06 CST by A-Lang (Admin)