Management Tips

Free Up Disk Space

- Free Up Disk Space in your Kubernetes Cluster - Virtualization Howto

Port Forwarding

- Usage: `kubectl port forward TYPE/NAME [LOCAL_PORT:]REMOTE_PORT`
- LOCAL_PORT is the port on your local machine running kubectl
- REMOTE_PORT is the port on the target pod or service in the Kubernetes cluster

```
# For a service
kubectl -n <namespace> port-forward svc/my-service 8080:80


# For a deployment
kubectl -n <namespace> port-forward deploy/my-deployment 8080:80
```

K8s Cluster Monitoring

- LFK is a lightning-fast, keyboard-focused, yazi-inspired terminal user interface for navigating and managing Kubernetes clusters.

Kube-Prometheus-Stack

A set of Kubernetes manifests, Grafana dashboards, and Prometheus rules for monitoring Kubernetes clusters

- https://github.com/prometheus-operator/kube-prometheus
- https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack
- Kube-Prometheus-Stack installation and configuration - Virtualization Howto
- Kubernetes Monitoring with Prometheus & Grafana: Real-World Scenarios, Custom Metrics, and Proactive Alerts | Medium
- How to Monitor Kubernetes Using Prometheus and Grafana

```
kubectl create ns monitoring
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo list
helm repo update
helm install prometheus-stack prometheus-community/kube-prometheus-stack -n monitoring
```

Get Grafana 'admin' user password by running:

```
kubectl --namespace monitoring get secrets prometheus-stack-grafana -o jsonpath="{.data.admin-password}" | base64 -d ; echo
```

Access Grafana local instance:

```
export POD_NAME=$(kubectl --namespace monitoring get pod -l "app.kubernetes.io/name=grafana,app.kubernetes.io/instance=prometheus-stack" -oname)
kubectl --namespace monitoring port-forward $POD_NAME 3000
```

Get your grafana admin user password by running:

```
kubectl get secret --namespace monitoring -l app.kubernetes.io/component=admin-secret -o jsonpath="{.items[0].data.admin-password}" | base64 --decode ; echo
```

Access Grafana from external network

```
kubectl expose deploy prometheus-stack-grafana --type=NodePort --name=prometheus-stack-grafana-nport --port=3000 -n monitoring
```

Uninstall

```
helm uninstall prometheus-stack -n monitoring
```

Install with the custom values

values.yaml:

```
prometheus:
  prometheusSpec:
    retention: 15d
    serviceMonitorSelector: {}
alertmanager:
  alertmanagerSpec:
```

```
    replicas: 2
grafana:
  adminPassword: my-secure-password
  service:
    type: LoadBalancer
```

Install with helm

```
helm install prometheus-stack prometheus-community/kube-prometheus-stack -n monitoring -f values.yaml
```

FAQ

> [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: Missing Authority Key Identifier

- https://github.com/canonical/microk8s/issues/4864
- https://github.com/prometheus-community/helm-charts/issues/5232

Solution:

- ?????????????? Cluster Platform??? AKS, Microk8s?
- ???? Microk8s ???? node ???

  ```
  mkdir cadir
  openssl genrsa -out cadir/ca.key 2048
  openssl req -x509 -new -nodes -key ca.key -sha256 -days 360 -out cadir/ca.crt -addext "keyUsage=critical,digitalSignature,keyCertSign"
  microk8s.refresh-certs cadir
  ```

  ?? node ??

Rollout & Rollback

```
# 將deployment管理的pod升級到特定image版本
kubectl set image deploy/<deployment-name> <pod-name>=<image-path>:<version>
kubectl set image deploy/hello-deployment my-pod=zxcvbnius/docker-demo:v2.0.0
# 指令結尾加上 --record, 歷史紀錄會有完整升級指令內容
kubectl set image deploy/hello-deployment my-pod=zxcvbnius/docker-demo --record

# 查詢目前某deployment升級狀況
```

```
kubectl rollout status deploy <deployment-name>
kubectl rollout status deploy hello-deployment

# 查詢目前某deployment升級的歷史紀錄
kubectl rollout history deploy <deployment-name>

# 回滾Pod到先前一個版本
kubectl rollout undo deploy <deployment-name>
kubectl rollout undo deployment hello-deployment

# 回滾Pod到某個特定版本
kubectl rollout undo deploy <deployment-name> --to-revision=n
kubectl rollout undo deploy hello-deployment --to-revision=3
```

CPU & RAM Limitation

- cpu: 200m ?? core ? 20%

deployment.yaml

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: helloworld-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: helloworld-pod
  template:
    metadata:
      labels:
        app: helloworld-pod
    spec:
      containers:
      - name: my-pod
        image: zxcvbnius/docker-demo:latest
        ports:
        - containerPort: 3000
        resources:
```

```
          requests:
            cpu: "200m"
            memory: "100Mi"
          limits:
            cpu: "400m"
            memory: "200Mi"
```

TroubleShooting

Use alpine

```
# Get the ClusterIP of the pod
kubectl describe pod <pod-name> -n <namespace>

# Enter the alpine
kubectl run -i --tty alpine --image=alpine --restart=Never -- sh
apk add --no-cache curl
curl http://<ClusterIP>:<port>
```

Check pods/events

```
# Get a list of pods sorted by memory usage
kubectl top pods -A --sort-by='memory'

# Watch all warnings across the namespaces
kubectl get events -w --field-selector=type=Warning -A
# Get a list of events sorted by lastTimestamp
kubectl get events --sort-by=".lastTimestamp"
```

Revision #27
Created 2026-01-27 17:23:27 CST by A-Lang (Admin)
Updated 2026-04-24 20:55:36 CST by A-Lang (Admin)