- Preface:

- All YAML files do not require IP information, and there is no distinction between different satellite nodes;
- Send the Future folder to /home/user, then execute `sudo ./monitor-deploy.sh`;

I. Monitoring System Deployment:

Complete the deployment of kube-state-metrics, metrics-server, and cAdvisor.

```
# monitor-deploy.sh
#!/bin/bash
set -e
set -x
DIR="/home/user/Future"
USER_PATH="/home/user"

# Image import
cd $DIR
sudo ctr -n k8s.io image import kube-state-metrics.tar.gz --platform=linux/arm64
sudo ctr -n k8s.io image import metrics-server.tar --platform=linux/arm64

# Prometheus external deployment
sudo mv $DIR/prometheus /usr/local/prometheus
sudo mv $DIR/prometheus.service /usr/lib/systemd/system
sudo systemctl daemon-reload
sudo systemctl start prometheus
sudo systemctl enable prometheus --now

sudo mv $DIR/node_exporter /usr/local/node_exporter
sudo mv $DIR/node_exporter.service /etc/systemd/system/node_exporter.service
sudo systemctl daemon-reload
sudo systemctl start node_exporter
sudo systemctl enable node_exporter --now

# metrics-server deployment
sudo -u user kubectl apply -f metrics-server.yaml
sudo -u user kubectl rollout status deployment/metrics-server -n kube-system --timeout=1m

# kube-state-metrics deployment
cd kube-state-metrics
sudo -u user kubectl apply -f .
sudo -u user kubectl rollout status deployment/kube-state-metrics -n kube-system --timeout=1m

# cAdvisor exposed to Prometheus
sudo -u user kubectl create ns cadvisor
sudo -u user kubectl create serviceaccount monitor -n cadvisor
sudo -u user kubectl create clusterrolebinding monitor-clusterrolebinding -n cadvisor --
clusterrole=cluster-admin --serviceaccount=cadvisor:monitor
sudo -u user kubectl create token monitor -n cadvisor --duration=8760h > $USER_PATH/monitor-token

# Set go script to start on boot
sudo mv $DIR/future-promql.service /usr/lib/systemd/system
sudo systemctl daemon-reload
sudo systemctl enable --now future-promql
sudo systemctl start future-promql
```

II. Metrics Collection

2.1

Execute the `./promql.sh`.

The go program completes all pod-level:

- CPU (cores)/memory (Bytes) usage;
- Usage to request quota ratio (if requested); network I/O rate.
- All rate calculations are precise to a 1-minute span.
- kube-state-metrics/docs/metrics/workload/pod-metrics.md · GitHub

2.2

Execute promql.go, hosted by systemd, which scrapes metrics every 10s, with each metric written to a separate CSV;

Specify the number of execution rounds at the end (1~128), default is 5 rounds, the following specifies a duration of 5 minutes.

```
# promql.sh
#!/bin/bash
set -e
export GOPATH=/home/user/go
export PATH=$PATH:/usr/local/go/bin:$GOPATH/bin
export HOME=/home/user
cd /home/user/Future
sudo -u user kubectl rollout status deployment/metrics-server -n kube-system --timeout=1m
sudo -u user kubectl rollout status deployment/kube-state-metrics -n kube-system --timeout=1m
go run promql.go 30
```

2.3

The data storage path is specified in promql.go's STORE_PATH, overall as STORE_PATH/Future/yyyy-mm-dd/****.csv

```
```