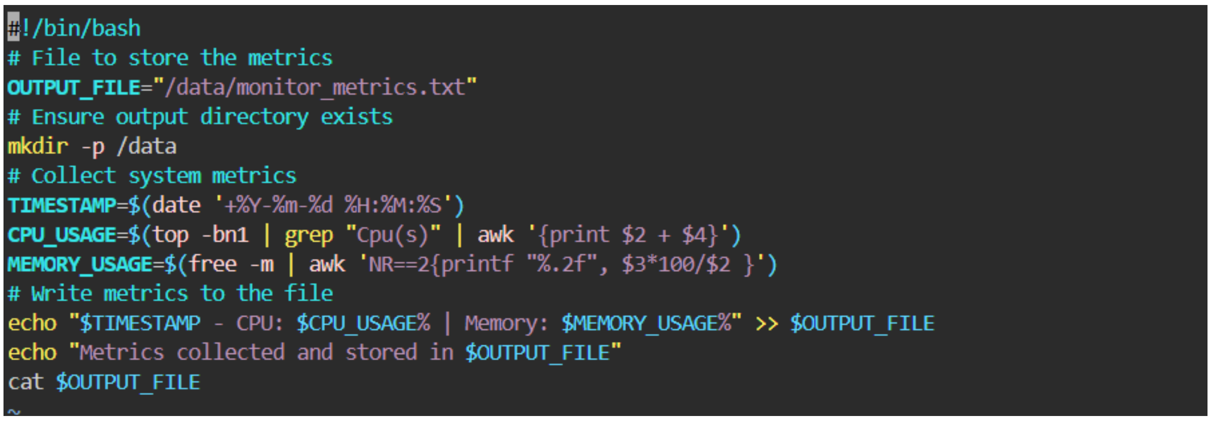
Automating System Metrics Monitoring in Kubernetes Using Jobs and CronJobs

simple CPU and memory monitoring script in Bash. It collects system statistics and stores them in a file, which can be executed using Kubernetes **Jobs** and **CronJobs**.

Step 1: Bash Script

Create a script `monitor.sh` that captures CPU and memory usage.

```
#!/bin/bash

# File to store the metrics

OUTPUT_FILE="/data/monitor_metrics.txt"

# Ensure output directory exists

mkdir -p /data

# Collect system metrics

TIMESTAMP=$(date '+%Y-%m-%d %H:%M:%S')

CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | awk '{print $2 + $4}')

MEMORY_USAGE=$(free -m | awk 'NR==2{printf "%.2f", $3*100/$2 }')

# Write metrics to the file

echo "$TIMESTAMP - CPU: $CPU_USAGE% | Memory: $MEMORY_USAGE%" >> $OUTPUT_FILE

echo "Metrics collected and stored in $OUTPUT_FILE"

cat $OUTPUT_FILE
```

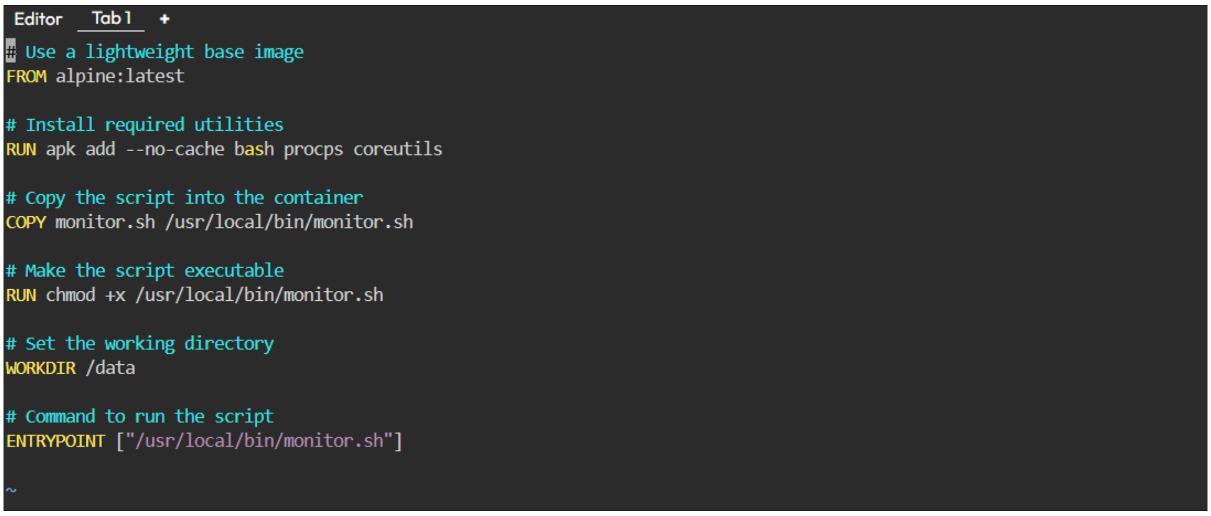


```
#!/bin/bash
# File to store the metrics
OUTPUT_FILE="/data/monitor_metrics.txt"
# Ensure output directory exists
mkdir -p /data
# Collect system metrics
TIMESTAMP=$(date '+%Y-%m-%d %H:%M:%S')
CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | awk '{print $2 + $4}')
MEMORY_USAGE=$(free -m | awk 'NR==2{printf "%.2f", $3*100/$2 }')
# Write metrics to the file
echo "$TIMESTAMP - CPU: $CPU_USAGE% | Memory: $MEMORY_USAGE%" >> $OUTPUT_FILE
echo "Metrics collected and stored in $OUTPUT_FILE"
cat $OUTPUT_FILE
~
```

Step 2: Dockerfile

To run the script in Kubernetes, package it in a Docker container. Create a `Dockerfile`.

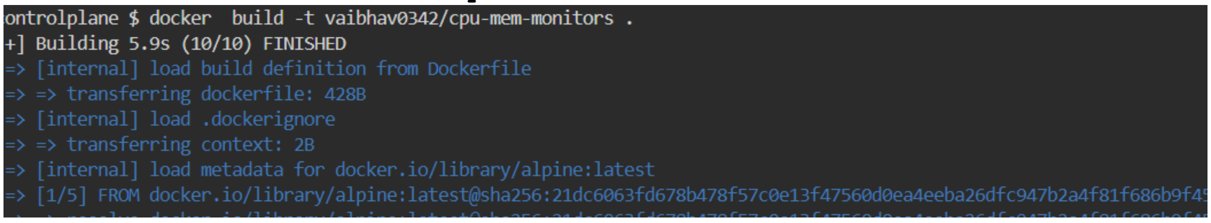
```
# Use a lightweight base image
FROM alpine:latest

# Install required utilities
RUN apk add --no-cache bash procps coreutils

# Copy the script into the container
COPY monitor.sh /usr/local/bin/monitor.sh

# Make the script executable
RUN chmod +x /usr/local/bin/monitor.sh

# Set the working directory
WORKDIR /data

# Command to run the script
ENTRYPOINT ["/usr/local/bin/monitor.sh"]
```



```
# Use a lightweight base image
FROM alpine:latest

# Install required utilities
RUN apk add --no-cache bash procps coreutils

# Copy the script into the container
COPY monitor.sh /usr/local/bin/monitor.sh

# Make the script executable
RUN chmod +x /usr/local/bin/monitor.sh

# Set the working directory
WORKDIR /data

# Command to run the script
ENTRYPOINT ["/usr/local/bin/monitor.sh"]

~
```

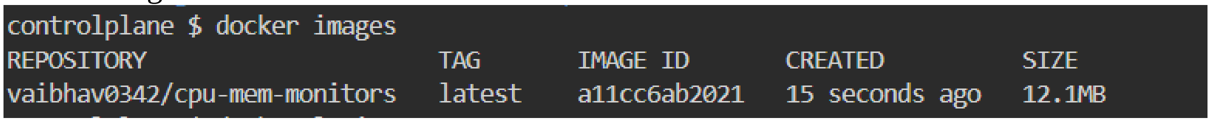
Build and push the Docker image:

- **`docker build -t vaibhav0342/cpu-mem-monitors .`**



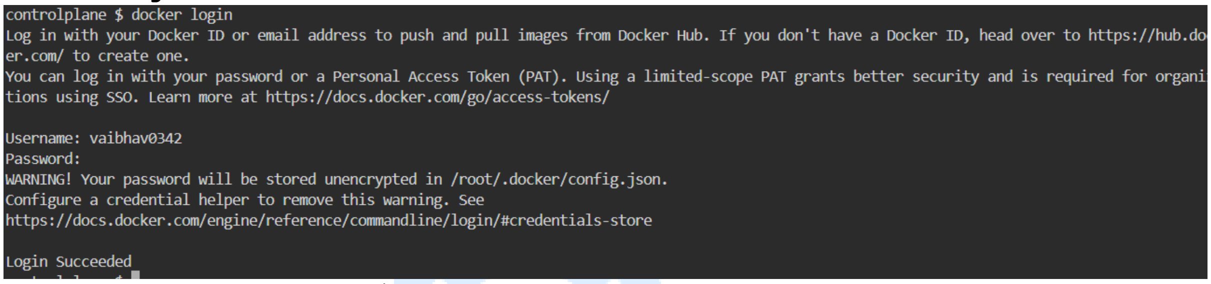
```
ontrolplane $ docker  build -t vaibhav0342/cpu-mem-monitors .
+] Building 5.9s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 428B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/alpine:latest
=> [1/5] FROM docker.io/library/alpine:latest@sha256:21dc6063fd678b478f57c0e13f47560d0ea4eeba26dfc947b2a4f81f686b9f45
```

- docker images



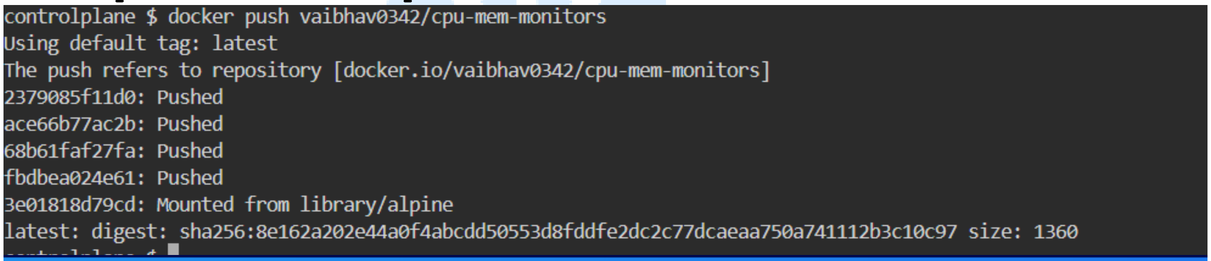
```
controlplane $ docker images
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
vaibhav0342/cpu-mem-monitors  latest    a11cc6ab2021   15 seconds ago   12.1MB
```

- **`docker login`**

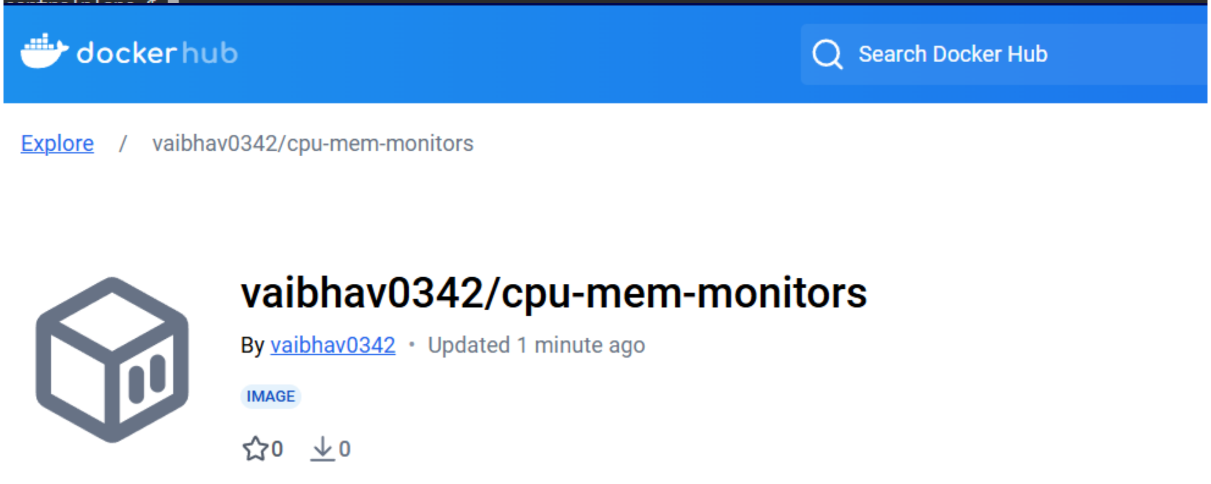


```
controlplane $ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.do
er.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required for organi
tions using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: vaibhav0342
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

- **`docker push vaibhav0342/cpu-mem-monitor`**



```
controlplane $ docker push vaibhav0342/cpu-mem-monitors
Using default tag: latest
The push refers to repository [docker.io/vaibhav0342/cpu-mem-monitors]
2379085f11d0: Pushed
ace66b77ac2b: Pushed
68b61faf27fa: Pushed
fbdbea024e61: Pushed
3e01818d79cd: Mounted from library/alpine
latest: digest: sha256:8e162a202e44a0f4abcdd50553d8fddfe2dc2c77dcaeaa750a741112b3c10c97 size: 1360
```

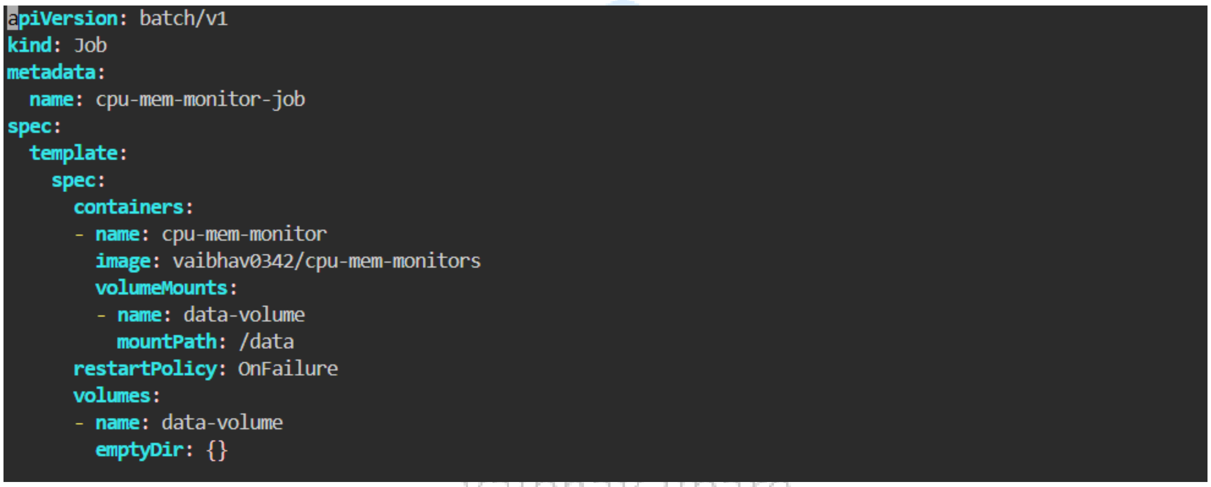


vaibhav0342/cpu-mem-monitors

By vaibhav0342 · Updated 1 minute ago

Step 3: Kubernetes Job

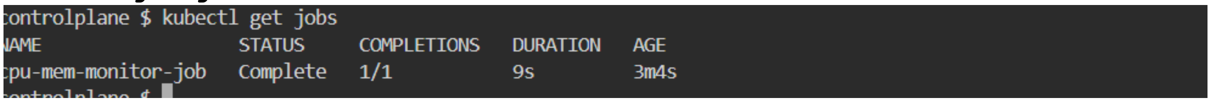
```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: cpu-mem-monitor-job
spec:
  template:
    spec:
      containers:
      - name: cpu-mem-monitor
        image: vaibhav0342/cpu-mem-monitors
        volumeMounts:
        - name: data-volume
          mountPath: /data
      restartPolicy: OnFailure
      volumes:
      - name: data-volume
        emptyDir: {}
```



```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: cpu-mem-monitor-job
spec:
  template:
    spec:
      containers:
      - name: cpu-mem-monitor
        image: vaibhav0342/cpu-mem-monitors
        volumeMounts:
        - name: data-volume
          mountPath: /data
      restartPolicy: OnFailure
      volumes:
      - name: data-volume
        emptyDir: {}
```

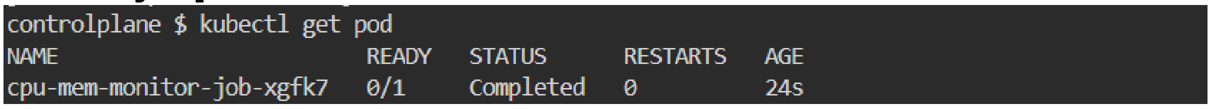
Apply the manifest:

- `kubectl apply -f job.yaml`
- `kubectl get jobs`



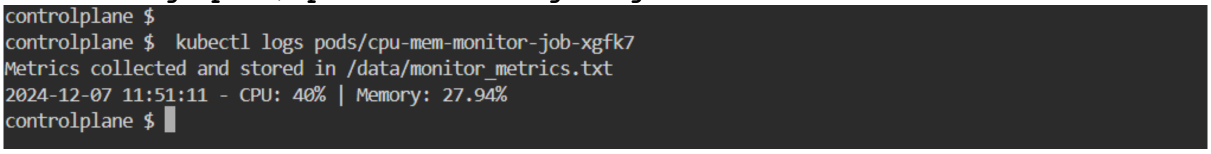
```
ontrolplane $ kubectl get jobs
AME                  STATUS     COMPLETIONS   DURATION   AGE
pu-mem-monitor-job   Complete   1/1           9s         3m4s
```

- `kubectl get pod`



```
controlplane $ kubectl get pod
NAME                        READY   STATUS      RESTARTS   AGE
cpu-mem-monitor-job-xgfk7   0/1     Completed   0          24s
```

- `kubectl logs pods/cpu-mem-monitor-job-xgfk7`



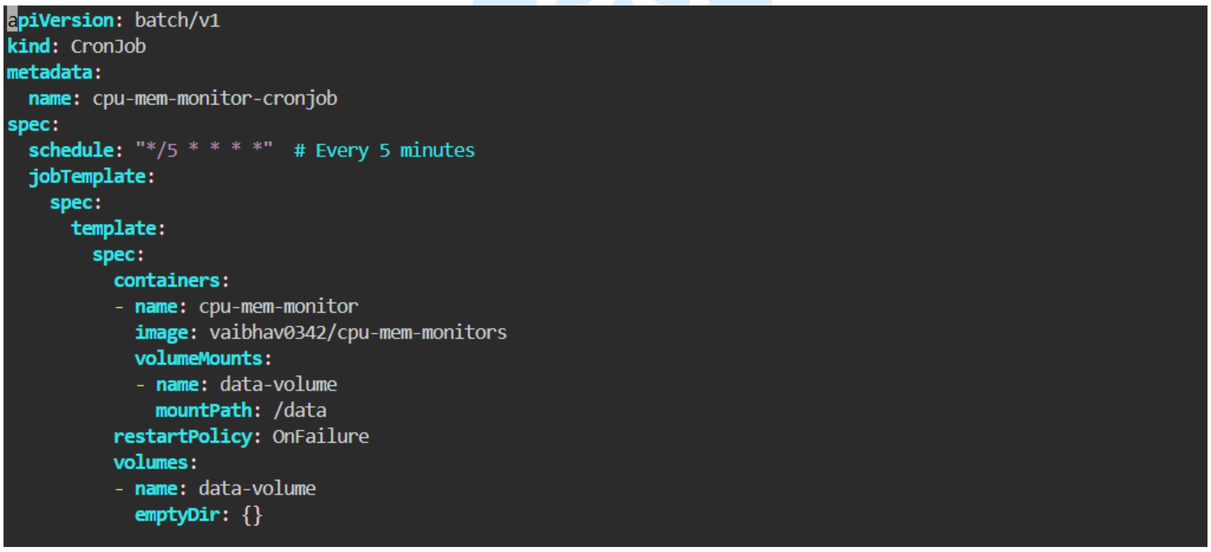
```
controlplane $
controlplane $  kubectl logs pods/cpu-mem-monitor-job-xgfk7
Metrics collected and stored in /data/monitor_metrics.txt
2024-12-07 11:51:11 - CPU: 40% | Memory: 27.94%
controlplane $
```

```
Metrics collected and stored in /data/monitor_metrics.txt
2024-12-07 11:51:11 - CPU: 40% | Memory: 27.94%
```

Step 4: Kubernetes CronJob

This is a Kubernetes CronJob configuration that schedules a job to ***run every 5 minutes.***

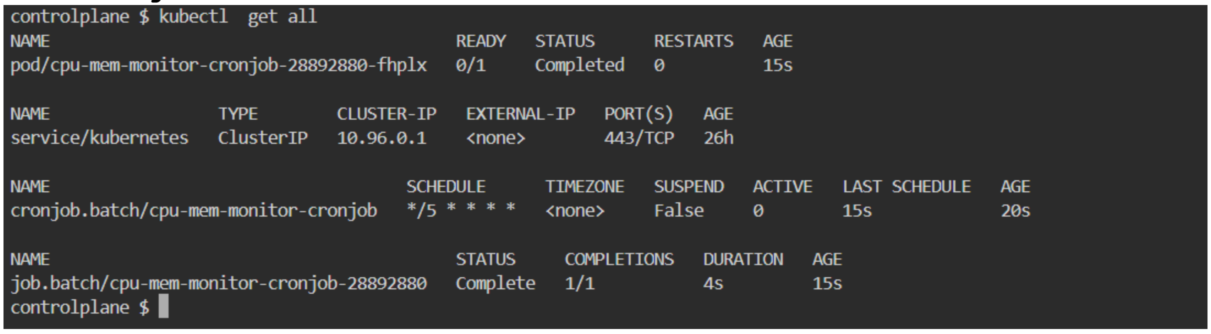
```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: cpu-mem-monitor-cronjob
spec:
  schedule: "*/5 * * * *"  # Every 5 minutes
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: cpu-mem-monitor
            image: vaibhav0342/cpu-mem-monitors
            volumeMounts:
            - name: data-volume
              mountPath: /data
          restartPolicy: OnFailure
          volumes:
          - name: data-volume
            emptyDir: {}
```



```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: cpu-mem-monitor-cronjob
spec:
  schedule: "*/5 * * * *"  # Every 5 minutes
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: cpu-mem-monitor
            image: vaibhav0342/cpu-mem-monitors
            volumeMounts:
            - name: data-volume
              mountPath: /data
          restartPolicy: OnFailure
          volumes:
          - name: data-volume
            emptyDir: {}
```

Apply the manifest:

- `kubectl apply -f cronjob.yaml`
- `kubectl  get all`

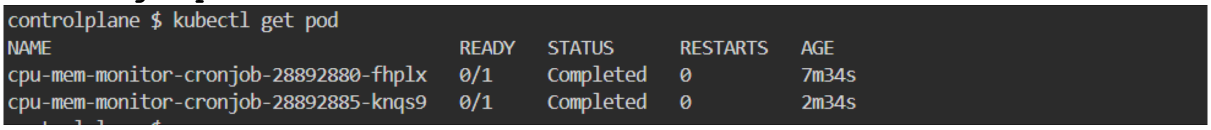


```
controlplane $ kubectl  get all
NAME                                          READY   STATUS      RESTARTS   AGE
pod/cpu-mem-monitor-cronjob-28892880-fhplx    0/1     Completed   0          15s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   26h

NAME                                    SCHEDULE      TIMEZONE   SUSPEND   ACTIVE   LAST SCHEDULE   AGE
cronjob.batch/cpu-mem-monitor-cronjob   */5 * * * *   <none>     False     0        15s             20s

NAME                                          STATUS     COMPLETIONS   DURATION   AGE
job.batch/cpu-mem-monitor-cronjob-28892880    Complete   1/1           4s         15s
controlplane $ 
```

Step 5: Accessing the Metrics

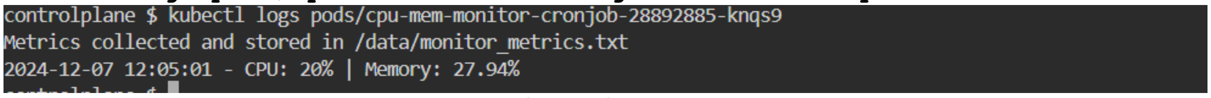
- The metrics will be stored in the file `/data/monitor_metrics.txt` within the container.
- You can view the logs by getting the pod associated with the Job or CronJob:

➢ **`kubectl get pod`**



```
controlplane $ kubectl get pod
NAME                                 READY   STATUS      RESTARTS   AGE
cpu-mem-monitor-cronjob-28892880-fhplx   0/1     Completed   0          7m34s
cpu-mem-monitor-cronjob-28892885-knqs9   0/1     Completed   0          2m34s
```

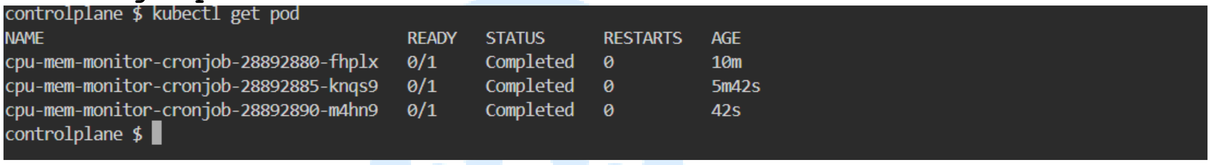
➢ **`kubectl logs pods/cpu-mem-monitor-cronjob-28892885-knqs9`**



```
controlplane $ kubectl logs pods/cpu-mem-monitor-cronjob-28892885-knqs9
Metrics collected and stored in /data/monitor_metrics.txt
2024-12-07 12:05:01 - CPU: 20% | Memory: 27.94%
```

```
Metrics collected and stored in /data/monitor_metrics.txt
2024-12-07 11:51:22 - CPU: 20% | Memory: 27.94%
```

➢ **`kubectl get pod`**



```
controlplane $ kubectl get pod
NAME                                     READY   STATUS      RESTARTS   AGE
cpu-mem-monitor-cronjob-28892880-fhplx   0/1     Completed   0          10m
cpu-mem-monitor-cronjob-28892885-knqs9   0/1     Completed   0          5m42s
cpu-mem-monitor-cronjob-28892890-m4hn9   0/1     Completed   0          42s
controlplane $
```
