

Managing Pods

- [Kubernetes Commands](#)
- [Add a New CronJob](#)
- [Managing CronJobs List](#)
- [Deleting Log Files inside other Pods Using CronJob](#)
- [Bash Script to Delete Log Files Inside Pods Based on Retention Days](#)
- [Bash Script to Delete Log Files Inside Pods Based on Retention Days with Different Target Folders](#)
- [Fix Error Agent Unavailable on Rancher](#)

Kubernetes Commands

Access inside a pod

```
kubectl exec -ti <pod_name> -n <namespace_name> -- /bin/bash
```

Find and stop pods

```
for i in `kubectl get deployment -n <namespace> | grep <pod-name-prefix> | awk '{print $1}'`  
do  
    echo $i  
    kubectl scale deployment $i -n <namespace> --replicas=0  
done
```

Find and start pods

```
for i in `kubectl get deployment -n <namespace> | grep <pod-name-prefix> | awk '{print $1}'`  
do  
    echo $i  
    kubectl scale deployment $i -n <namespace> --replicas=1  
done
```

Add a New CronJob

Option 1: Create a CronJob Using a YAML Manifest

1. Create a new `cronjob.yaml` file:

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: example-cronjob
spec:
  schedule: "*/5 * * * *" # Runs every 5 minutes
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: example
              image: busybox
              args:
                - /bin/sh
                - -c
                - date; echo Hello from the Kubernetes CronJob!
          restartPolicy: OnFailure
```

2. Apply the new CronJob:

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

Option 2: Use `kubectl create`

You can create a CronJob directly with the `kubectl create` command:

```
kubect! create cronjob <cronjob-name> \  
  --schedule="*/5 * * * *" \  
  --image=busybox \  
  -- /bin/sh -c "date; echo Hello from the Kubernetes CronJob!"
```

This creates a simple CronJob that runs every 5 minutes and prints the current date and a message.

Verifying Changes

- List CronJobs to ensure the changes or new CronJob were applied:

```
kubect! get cronjobs
```

- Describe CronJob to see detailed information:

```
kubect! describe cronjob <cronjob-name>
```

Let me know if you need help with any specific part!

Managing CronJobs List

1. List all CronJobs in the current namespace:

```
kubectl get cronjobs
```

This will display a list of all CronJobs in the namespace you are currently working in.

2. List CronJobs in a specific namespace:

```
kubectl get cronjobs -n <namespace>
```

Replace `<namespace>` with the name of the namespace.

3. View detailed information about a specific CronJob:

```
kubectl describe cronjob <cronjob-name>
```

This provides detailed information about the specified CronJob, including its schedule, concurrency policy, and job history.

4. List all CronJobs across all namespaces:

```
kubectl get cronjobs --all-namespaces
```

5. Patch CronJob to Update Image:

```
kubectl set image cronjob/<cronjob-name> <cronjob-name>=<image>:<new-tag>
```

6. Look for the created Job and Pod associated with the CronJob:

```
kubectl get jobs -n <your-namespace>
```

```
kubectl get pods -n <your-namespace> --selector=job-name=<job-name>
```

Deleting Log Files inside other Pods Using CronJob

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: delete-server-log
  namespace: <namespace>
spec:
  schedule: "*/5 * * * *" # Runs every 5 minutes
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: <pod-name>
              image: <image>:<tag>
              args:
                - /bin/sh
                - -c
                - date=$(date +%Y-%m-%d_%H-%M-%S); echo "Hello from the Kubernetes CronJob!" > /Log/delete-
$date.txt
          restartPolicy: OnFailure
          imagePullSecrets:
            - secret1
            - secret2
          securityContext:
            runAsUser: 0 # Run as root
```

Bash Script to Delete Log Files Inside Pods Based on Retention Days

```
#!/bin/bash

# Retention days for log files (delete files older than this many days)
RETENTION_DAYS=2 # Change this to your desired retention period in days

# Directory where the logs are mounted (should be the same as mountPath in Kubernetes)
LOG_DIR="/Log" # Change this to the path where logs are mounted on your system

# Namespace where the pods are located
NAMESPACE="<<namespace>" # Change this to your actual namespace

# Patterns to search for in pod names
PATTERNS=("<pattern-1", "<pattern-2", "<pattern-n")

# Find all pod names that match the multiple patterns in the specified namespace
PODS=$(kubectl get pods -n "$NAMESPACE" --no-headers -o custom-columns=":metadata.name" | grep -E "$PATTERN_REGEX")

# Check if there are any matching pods
if [ -z "$PODS" ]; then
    echo "No pods found matching the patterns in namespace $NAMESPACE."
    exit 1
fi

# Loop through each pod name in the array
for POD in "${PODS[@]}"; do
    echo "Cleaning logs in pod $POD..."
done
```

```
# Execute the command to delete log files in the pod older than retention days
kubectl exec -n "$NAMESPACE" "$POD" -- /bin/sh -c "find $LOG_DIR -type f -name 'town_*.log' -mtime
+$RETENTION_DAYS -exec rm -f {} \;"

kubectl exec -n "$NAMESPACE" "$POD" -- /bin/sh -c "find $LOG_DIR -type f -name 'town*.log' -mtime
+$RETENTION_DAYS -exec rm -f {} \;"

# Confirmation message
echo "Log cleanup complete for pod $POD."
done
```


Bash Script to Delete Log Files Inside Pods Based on Retention Days with Different Target Folders

```
#!/bin/bash

# Retention days for log files (delete files older than this many days)
RETENTION_DAYS=1 # Change this to your desired retention period in days

# Namespace where the pods are located
NAMESPACE="app5-ns" # Change this to your actual namespace

# Declare an associative array
declare -A config

# Key -> Patterns to search for in pod names
# Value -> Directory where the logs are mounted (should be the same as mountPath in Kubernetes)
config["x-server-"]="/app/Log"
config["y-server-"]="/Log"
config["z-server-"]="/Log"

# Add z-server entries from z-server02 to z-server10
for i in $(seq -w 2 10); do
    config["z-server${i}-"]="/Log"
done

# Generate PATTERN_REGEX from config keys
PATTERN_REGEX=$(printf "|%s" "${!config[@]}")
PATTERN_REGEX=${PATTERN_REGEX:1} # Remove the leading '|'
```

```

# Find all pod names that match the multiple patterns in the specified namespace
PODS=$(kubectl get pods -n "$NAMESPACE" --no-headers -o custom-columns=":metadata.name" | grep -E
"$PATTERN_REGEX")

# Check if there are any matching pods
if [ -z "$PODS" ]; then
    echo "No pods found matching the patterns in namespace $NAMESPACE."
    exit 1
fi

# Iterate over each pod
for POD in $PODS; do
    # Determine the log directory based on the matching pattern
    for key in "${!config[@]}"; do
        if [[ "$POD" =~ $key ]]; then
            LOG_DIR="${config[$key]}"
            echo "Entering pod: $POD (Log Directory: $LOG_DIR)"

            # Check if the log directory exists and list log files
            kubectl exec -n "$NAMESPACE" "$POD" -- /bin/sh -c "find $LOG_DIR -type f -name '*.log' -mtime
+$RETENTION_DAYS"

            # kubectl exec -n "$NAMESPACE" "$POD" -- /bin/sh -c "find $LOG_DIR -type f -name '*.log' -mtime
+$RETENTION_DAYS -exec rm -f {} \;"

            # Confirmation message
            echo "Log cleanup complete for pod $POD."
            echo "-----"
            break
        fi
    done
done

```

Fix Error Agent Unavailable on Rancher

Find agent pod for Rancher

```
kubectl get pods -n cattle-system
```

Show error logs

```
kubectl logs -n cattle-system cattle-cluster-agent-<id>
```

Result:

```
ERROR: The environment variable CATTLE_CA_CHECKSUM is set but there is no CA certificate configured at https://<domain>/v3/settings/cacerts
```

Find `CATTLE_CA_CHECKSUM` configuration

```
kubectl describe deployment -n cattle-system cattle-cluster-agent
kubectl get svc -n cattle-system cattle-cluster-agent
kubectl get svc -n cattle-system cattle-cluster-agent -o yaml
kubectl get deployment -n cattle-system cattle-cluster-agent -o yaml
```

Update value of `CATTLE_CA_CHECKSUM` if it is necessary

Open the URL and copy the CA certificate.

```
https://<domain>/v3/settings/cacerts
```

Generate checksum:

```
sha256sum rancher-ca.crt
```

Edit the value on deployment:

```
kubectrl edit deployment cattle-cluster-agent -n cattle-system
```

Restart the service

```
kubectrl rollout restart deployment cattle-cluster-agent -n cattle-system
```