Shoreline.io

Runbook

# Kubernetes Cronjob Failure

← Back to Runbooks

## Overview

Kubernetes 52

Get the Terraform

A Kubernetes Cronjob Failure incident occurs when a scheduled task, or cronjob, in a Kubernetes cluster fails to execute as expected. This may be due to a variety of reasons, such as misconfiguration, resource constraints, or software bugs. The incident requires investigation and debugging to identify the root cause of the failure and resolve the issue to restore normal operation.There is also a kubernetes limitation that permanently stops a cronjob after too many (e.g. 100) execution errors or failures to schedule.

## Parameters

```
export CRONJOB_NAME="PLACEHOLDER"

export POD_NAME="PLACEHOLDER"

export NAMESPACE="PLACEHOLDER"

export EXPECTED_SCHEDULE="PLACEHOLDER"

export PATH_TO_CRONJOB_YAML="PLACEHOLDER"
```

## Debug

### Check if the cronjob is still active

```
kubectl get cronjobs
```

## Check if the pods created by the cronjob are still running

```
kubectl get pods -l job-name=${CRONJOB_NAME}
```

## Check the logs of the pods created by the cronjob

```
kubectl logs ${POD_NAME}
```

## Check if the cronjob schedule is correct

```
kubectl describe cronjobs/${CRONJOB_NAME}
```

## Check if there are any errors in the cronjob events

```
kubectl describe events --field-selector involvedObject.name=${CRONJOB_NAME}
```

## Check if the cronjob image exists in the container registry

```
kubectl describe cronjobs/${CRONJOB_NAME} | grep Image:
```

## Check the status of the last cronjob run

```
kubectl describe cronjobs/${CRONJOB_NAME} | grep Last Schedule Time:
```

## Check if the cronjob is running on the expected node

```
kubectl describe pods ${POD_NAME} | grep Node:
```

Check if the pod has sufficient resources

```
kubectl describe pods ${POD_NAME} | grep -i resource
```

Check if there are any errors in the pod events

```
kubectl describe pods ${POD_NAME} | grep -i events
```

## Repair

Check the cronjob configuration to ensure that it is correctly defined and scheduled to run at the intended time.

```
#!/bin/bash



# Set the namespace and cronjob name

NAMESPACE=${NAMESPACE}

CRONJOB_NAME=${CRONJOB_NAME}



# Get the cronjob object

CRONJOB=$(kubectl get cronjob $CRONJOB_NAME -n $NAMESPACE -o json)



# Check if the cronjob schedule is correct

EXPECTED_SCHEDULE=${EXPECTED_SCHEDULE}

CURRENT_SCHEDULE=$(echo $CRONJOB | jq -r .spec.schedule)



```

```
if [ "$CURRENT_SCHEDULE" != "$EXPECTED_SCHEDULE" ]; then

    # Update the cronjob schedule

    kubectl patch cronjob $CRONJOB_NAME -n $NAMESPACE --type='json' -p='[{"op

fi
```

## Check for "100 missed start times" error. Recreate the cronjob if found.

```
#!/bin/bash



# Set the namespace and cronjob name

NAMESPACE=${NAMESPACE}

CRONJOB=${CRONJOB_NAME}



# Check for "100 missed start times" error

if kubectl get cronjob $CRONJOB -n $NAMESPACE | grep -q "100 missed"; then

  # Delete the cronjob

  kubectl delete cronjob $CRONJOB -n $NAMESPACE



  # Recreate the cronjob

  kubectl apply -f ${PATH_TO_CRONJOB_YAML} -n $NAMESPACE



  # Verify the new cronjob is scheduled and running
```

```

    kubectl get cronjob $CRONJOB -n $NAMESPACE

    kubectl get pods -l job-name=$CRONJOB -n $NAMESPACE

fi
```

**Learn more**

## Related Runbooks

Check out these related runbooks to help you debug and resolve similar issues.