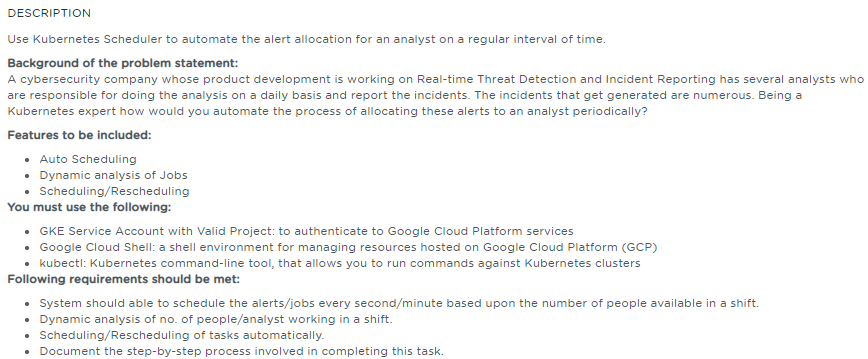
**Job Scheduling Using Kubernetes Cluster - Assessment**



**Table of Contents**

[**1.** **Project Abstract** 1](#_Toc44370662)

[**2.** **Work Environment** 1](#_Toc44370663)

[**2.1.** **Prerequisites** 1](#_Toc44370664)

[**2.2.** **Setting up Kubernetes Cluster** 2](#_Toc44370665)

[**3.** **Creation of Job/Tasks using Cronjob** 2](#_Toc44370666)

[**3.1.** **Create a Cronjob** 2](#_Toc44370667)

[**3.2.** **Run a Cronjob** 3](#_Toc44370668)

[**3.3.** **Status of Cronjob** 3](#_Toc44370669)

[**4.** **Scheduling of Cronjob** 3](#_Toc44370670)

[**4.1.** **Specify when the Cronjob runs** 3](#_Toc44370671)

[**4.2.** **Specify what the Cronjob runs** 4](#_Toc44370672)

[**4.3.** **Specify Deadline (optional)** 4](#_Toc44370673)

[**4.4.** **Inspecting a Cronjob** 4](#_Toc44370674)

[**5.** **References** 4](#_Toc44370675)

# **Project Abstract**

The intent of this project is to automate the alert allocation for an analyst on a regular interval of time using the Kubernetes Scheduler.

# **Work Environment**

## **Prerequisites**

Kubernetes cluster, and the kubectl command-line tool must be configured to communicate with your cluster.

## **Setting up Kubernetes Cluster**

Using the SimpliLearn Practice Lab session, Kubernetes Cluster is being setup with one master and two worker nodes

Run the below command on the master

|  |
| --- |
| $ sudo –i  $ kubeadm reset |

**kubeadm reset** is responsible for cleaning up a node local file system from files that were created using the kubeadm init or kubeadm join commands

Run the below command in order to set up the Kubernetes control plane

|  |
| --- |
| $ kubeadm init |

After executing the init command, Run the below commands to initialize the configuration and set the correct permissions

|  |
| --- |
| $ mkdir -p $HOME/.kube  $ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  $ sudo chown $(id -u):$(id -g) $HOME/.kube/config |

On Master, Run the below command to verify the master has all the required components running successfully

|  |
| --- |
| $ kubectl get pods –n kube-system |

Run the below command on the master to create the join-token

|  |
| --- |
| $ kubeadm token create --print-join-command |

On the slave nodes(kslave1 and kslave2), Run the command

|  |
| --- |
| $ kubeadm join 172.31.17.255:6443 --token wrfgen.elfhgxgoeqd7bnxn --discovery-token-ca-cert-hash sha256:655827d00fec5fa80d71caf58fa67b9efbc8824f1224952a4ad9fb5574fc6eaa |

On Master node, Run the following command to verify the worker nodes are joined the cluster

|  |
| --- |
| $ kubectl get nodes |

# **Creation of Job/Tasks using Cronjob**

Cronjobs is mainly used to run the tasks at a specific time or interval and to execute or assign the tasks automatically at regular interval of time

Good choice for automatic tasks such as backups, reporting, sending emails, or cleanup tasks (job scheduling).

CronJobs use job objects to complete their tasks. A CronJob creates a job object each time it runs. CronJobs are created, managed, scaled, and deleted in the same way as jobs.

## **Create a Cronjob**

Create a Cronjob from the manifest file as shown below and save the YAML manifest to a file (cronjob.yml)

|  |
| --- |
| apiVersion: batch/v1  kind: CronJob  metadata:  name: hello  spec:  schedule: "\*/1 \* \* \* \*"  jobTemplate:  spec:  template:  spec:  containers:  - name: hello  image: busybox  args:  - /bin/sh  - -c  - date; echo Hello from the Kubernetes cluster  restartPolicy: OnFailure |

## **Run a Cronjob**

Run the CronJob by using this command:

|  |
| --- |
| $ kubectl create –f cronjob.yml |

The output of the command results in

**cronjob.batch/hello created**

## **Status of Cronjob**

After creating the Cronjob, get its status using this command:

|  |
| --- |
| $ kubectl get cronjob hello |

The Last schedule status is set to None as the scheduler is set to run for every 1 minute

As you can see from the results of the command, the Cronjob has not scheduled or run any jobs yet. Watch for the job to be created in around few minutes:

|  |
| --- |
| $ kubectl get jobs -w |

Now you've seen five running job scheduled by the "hello" cron job. You can stop watching the job and view the cron job again to see that it scheduled the job:

You should see that the Cronjob **hello** successfully scheduled a job at the time specified in LAST SCHEDULE. There are currently 0 active jobs, meaning that the job has completed or failed.

Now, find the pods that the last scheduled job created and view the standard output of one of the pods.

# **Scheduling of Cronjob**

## **Specify when the Cronjob runs**

The **spec.schedule** field defines when and how often the Cronjob runs, using Unix standard crontab format. All Cronjob times are in UTC. There are 5 fields separated by spaces. These fields represent the following:

* Minutes (between 0 and 59)
* Hours (between 0 and 23)
* Day of the month (between 1 and 31)
* Month (between 1 and 12)
* Day of the week (between 0 and 6)

You can use the following special characters in any of the **spec.schedule** fields:

* ? is a wildcard value that matches a single character.
* \* is a wildcard value that matches zero or more characters.
* /, allows you to specify an interval for a field. For instance, if the first field (the minutes field) has a value of \*/5, it means "every 5 minutes". If the fifth field (the day-of-week field) is set to 0/5, it means "every fifth Sunday."

The **spec.schedule** value defined in the Cronjob.yml is \*/1 i.e scheduled for every 1 minute

The **spec.schedule** created in the Cronjob.yml is shown below mentioned

## **Specify what the Cronjob runs**

The **spec.jobTemplate** describes what the CronJob does, including its container images, the commands the containers execute, and the restart policy for the Cronjob.

## **Specify Deadline (optional)**

The **spec.startingDeadlineSeconds** field is optional. It stands for the deadline in seconds for starting the job if it misses its scheduled time for any reason.

After the deadline, the cron job does not start the job. Jobs that do not meet their deadline in this way count as failed jobs. If this field is not specified, the jobs have no deadline.

## **Inspecting a Cronjob**

To check a Cronjob's configuration, use kubectl describe:

|  |
| --- |
| $ kubectl describe cronjob hello |

# **References**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Components** | **Reference** |
| 1 | Screenshots Document | 1.Job\_Scheduling\_using\_k8s\_Cluster\_screenshots.docx |
| 2 | Sources | 1.Job\_Scheduling\_using\_k8s\_Cluster\_sources.docx |