Kubernetes GitOps

## Prerequisites

# For this tutorial we will use continuous delivery tool Harness to do GitOps. In order to use Harness, we will need to create a Harness free account. It uses Argo CD-as-a-Service.

# Also, we will need a GitHub account and a sample repository. For this exercise purpose we forked the repository from the example.

# In order to proceed with the assignment, we need to create a Harness account. After our account Is created, we will proceed with the presented deployment options. In our case **Kubernetes with GitOps.** Here Harness takes care of everything. When the agent gets installed successfully, we go to the next configuration. There we provide the Git details, we add the Git repo link, the one we forked at the beginning.

# To test the GitOps workflow we select “Harness Hosted”. After this we connect to the cluster and make sure the connection is successful. If it is successful, we will get this screen:

# Graphical user interface, text, application Description automatically generated

# Next, we go to ‘App Details’ tab and there we configure the sync policy from ‘manual’ to ‘automated’. This means, when any changes happen to our GitHub repo, the GitOps agent will pick those changes automatically and syncing will happen in an automated fashion.

# Graphical user interface, text, application, chat or text message Description automatically generated Graphical user interface, text, application, chat or text message Description automatically generated

# 

# The next tabs, the ‘Sync Status’ tab, shows us the sync status of the application, and the ‘Manifest’ tab, shows us the manifest details of the application.

# The ‘GitOps’ tab from the main dashboard represents the entry to our GitOps workflow.

# Graphical user interface, application Description automatically generated

# When we click on the application, we see our application.

# 

# Next, we click on our application and go to the Resource view where we see 5 replicas running as mentioned in the values.yaml file.

# 

# Next, we will try to change the replicaCount to 4 and 6. For this purpose we go to AppDetailes, and there we change it from 5 to 4.

# Graphical user interface, text, application Description automatically generated

# Here we can see our second deployment as ‘Automated’.

# 

# Graphical user interface, text, application, email Description automatically generated

# After a short time the configuration state shows healthy and successful sync state.

# 

# In this resource view we see that the number of replicas is 4, as we changed it from 5 to 4.

# If we want to change the replicaCount to 6 we repeat the process. We go to AppDetailes, and there we change it replicaCount to 6 instead of 4.

# Graphical user interface, application Description automatically generated

# In this resource view we see that the number of replicas now is 6, as we changed it from 4 to 6.

# Graphical user interface Description automatically generated