	Bansilal Ramnath Agarwal Charitable Trust's Vishwakarma Institute of Information Technology Department of Artificial Intelligence and Data Science	
Name: Siddhesh Dilip Khairnar		
Class: TY	Division: B	Roll No: 372028
Semester: 5th		Academic Year: 2022-2023
Subject Name & Code: Cloud Computing & Analytics ADUA31203		
Title of Assignment: Deploy Web app using Kubernetes		

ASSIGNMENT NO. 8

Theory/ Writeup:

Introduction: Kubernetes is an open-source platform for automating the deployment, scaling, and management of containerized applications. It is a popular tool for container orchestration and provides a way to manage large numbers of containers as a single unit rather than having to manage each container individually.

Kubernetes has become an essential tool for managing and deploying modern applications, and its importance lies in its ability to provide a unified platform for automating and scaling the deployment, management, and scaling of applications. With Kubernetes, organizations can achieve increased efficiency and agility in their development and deployment processes, resulting in faster time to market and reduced operational costs. Kubernetes also provides a high degree of scalability, allowing organizations to scale their applications as their business grows and evolves easily.

Additionally, Kubernetes offers robust security features, ensuring that applications are protected against potential threats and vulnerabilities. With its active community and extensive ecosystem, Kubernetes provides organizations with access to a wealth of resources, tools, and services that can help them to improve and enhance their applications continuously. Overall, the importance of using Kubernetes lies in its ability to provide a flexible, scalable, and secure platform for managing modern applications and enabling organizations to stay ahead in a rapidly evolving digital landscape.

Here's a basic overview of how to use Kubernetes: ☐ **Set up a cluster:**

To use Kubernetes, you need to set up a cluster, which is a set of machines that run the Kubernetes control plane and the containers. You can set up a cluster on your own infrastructure or use a cloud

provider such as Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure. □
Package your application into containers:

To run your application on Kubernetes, you need to package it into one or more containers. A container is a standalone executable package that includes everything needed to run your application, including the code, runtime, system tools, libraries, and settings.

- **Define the desired state of your application using manifests:**

Kubernetes uses manifests, which are files that describe the desired state of your application, to manage the deployment and scaling of your containers. The manifests specify the number of replicas of each container, how they should be updated, and how they should communicate with each other.

- **Push your code to an SCM platform:**

Push your application code to an SCM platform such as GitHub.

- **Use a CI/CD tool to automate:**

Use a specialised CI/CD platform such as Harness to automate the deployment of your application. Once you set it up, done; you can easily and often deploy your application code in chunks whenever a new code gets pushed to the project repository.

- **Expose the application:**

Once you deploy your application, you need to expose the application to the outside world by creating a Service with a type of LoadBalancer or ExternalName.

This allows users to access the application through a stable IP address or hostname.

- **Monitor and manage your application:**

After your application is deployed, you can use the kubectl tool to monitor the status of your containers, make changes to the desired state, and scale your application up or down.

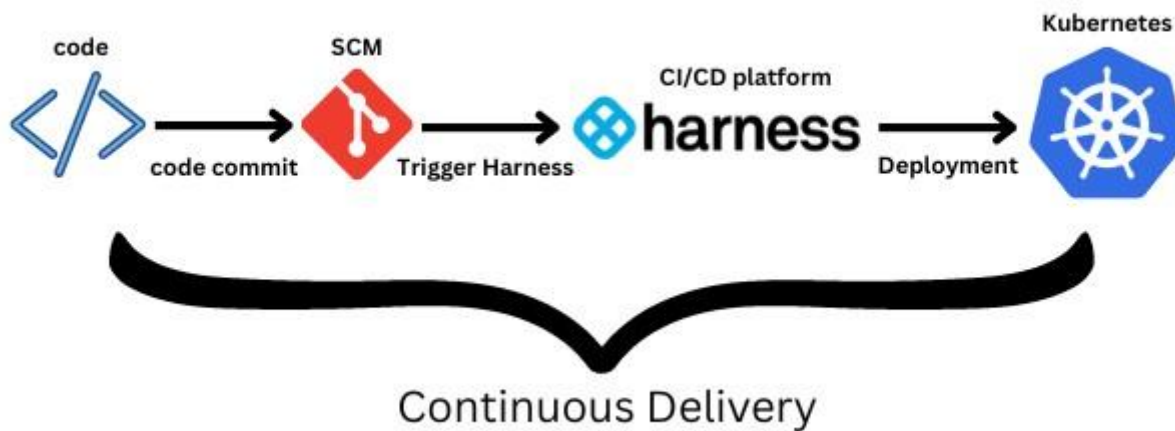
These are the general steps to deploy an application on Kubernetes. Depending on the application's complexity, additional steps may be required, such as configuring storage, network policies, or security. However, this should give you a good starting point for deploying your application on Kubernetes.

Today, we will see how to automate simple application deployment on Kubernetes using Harness.

Prerequisites

- Free [Harness cloud](#) account
- Download and install [Node.js and npm](#)
- GitHub account, we will be using our [sample notes application](#)
- Kubernetes cluster access, you can use [Minikube](#) or [Kind](#) to create a singlenode cluster

Tutorial



We will use our sample application that is already in the GitHub repository. We will use a Kubernetes cluster to deploy our application. Next, we will use a CI/CD platform, Harness, in this tutorial to show how we can automate the software delivery process easily.

Step 1: Test the sample application locally

Fork and clone the [sample notes application](#)

Go to the application folder with the following command

```
cd notes-app-cicd
```

Install dependencies with the following command

```
npm install
```

Run the application locally to see if the application works perfectly well

```
node app.js
```

Step 2: Containerize the application

You can see the Dockerfile in the sample application repository.

```
FROM node:14-alpine
```

```
WORKDIR /app
```

```
COPY package*.json ./
```

```
RUN npm install
```

```
COPY . .
```

```
EXPOSE 3000
```

```
CMD [ "npm", "start" ]
```

Use the following command to build, tag and push the image to any container registry of your choice. We will push it to Docker Hub in this tutorial. For Mac M1, use the following command

```
docker buildx build --platform=linux/arm64 --platform=linux/amd64 -t docker.io/$your docker hub user name/$image name:$tag name --push -f ./Dockerfile .
```

For other than Mac M1, use the below commands to build and push the image,

```
docker build -t $your docker hub user name/$image name .
docker push $your docker hub user name/$image name .
```

Step 3: Create or get access to a Kubernetes cluster

Make sure to have access to a Kubernetes cluster from any cloud provider. You can even use Minikube or Kind to create a cluster. In this tutorial, we are going to make use of a Kubernetes cluster from Google Cloud (GCP)

I already have an account on Google Cloud, so creating a cluster will be easy.

Step 4: Make sure the Kubernetes manifest files are neat and clean

You need deployment yaml and service yaml files to deploy and expose your application. Make sure both files are configured properly.

You can see that we have deployment.yaml and service.yaml file already present in the sample application repository.

Below is our deployment.yaml file.

```
apiVersion: apps/v1 kind:
Deployment metadata: name:
notes-app-deployment labels:
app: notes-app spec:
replicas: 2
selector:
matchLabels: app:
notes-app
template:
metadata: labels:
app: notes-app
spec: containers:
- name: notes-app-
deployment image:
pavansa/notes-app
resources: requests: cpu:
"100m" imagePullPolicy:
IfNotPresent ports:
- containerPort: 3000
```

Below is our service.yaml file

```
apiVersion: v1
# Indicates this as a service
kind: Service metadata: #
Service name name: notes-
```

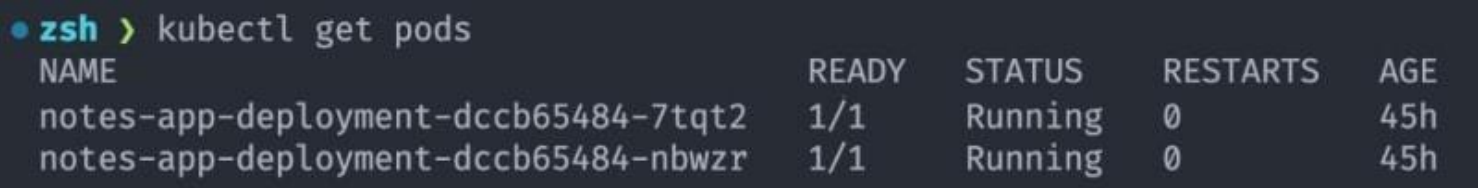
app-deployment spec:
selector:
Selector for Pods
app: notes-app ports:
Port Map - port:
80 targetPort:
3000 protocol:
TCP type:
LoadBalancer

Apply the manifest files with the following commands. Starting with deployment and then service yaml file.

```
kubectl apply -f deployment.yaml kubectl  
apply -f service.yaml
```

Verify the pods are running properly as expected after applying the kubectl apply commands.

```
kubectl get pods
```



NAME	READY	STATUS	RESTARTS	AGE
notes-app-deployment-dccb65484-7tqt2	1/1	Running	0	45h
notes-app-deployment-dccb65484-nbwzr	1/1	Running	0	45h

Step 5: Let's automate the deployment using Harness

You need a CI/CD tool to automate your continuous integration and deployment process. Harness is known for its innovation and simplicity in the CI/CD space. Hence, we will use this platform to set up automated continuous deployment of our application.

Once you sign up and verify your account, you will be presented with a welcome message and project creation set up. Proceed to create a project.



Welcome [REDACTED], let's get you started!

Take your software delivery processes to the next level using our Harness modules

[+ Project](#)

Add the name to the project, save and continue.

1 About the Project

2 Invite Collaborators (Optional)

About the Project

Name ⓘ Id ⓘ : notesappexa...

notes-app-example

Color ⓘ Organization ⓘ

default

Description (optional)

Tags (optional)

Save and Continue >

notes-app-example

Id: notesappexample

Admin (1)

Collaborators

+

+

Modules

Select the 'Continuous Delivery' module and start your free plan.

Which Harness module would you like to start using for this project

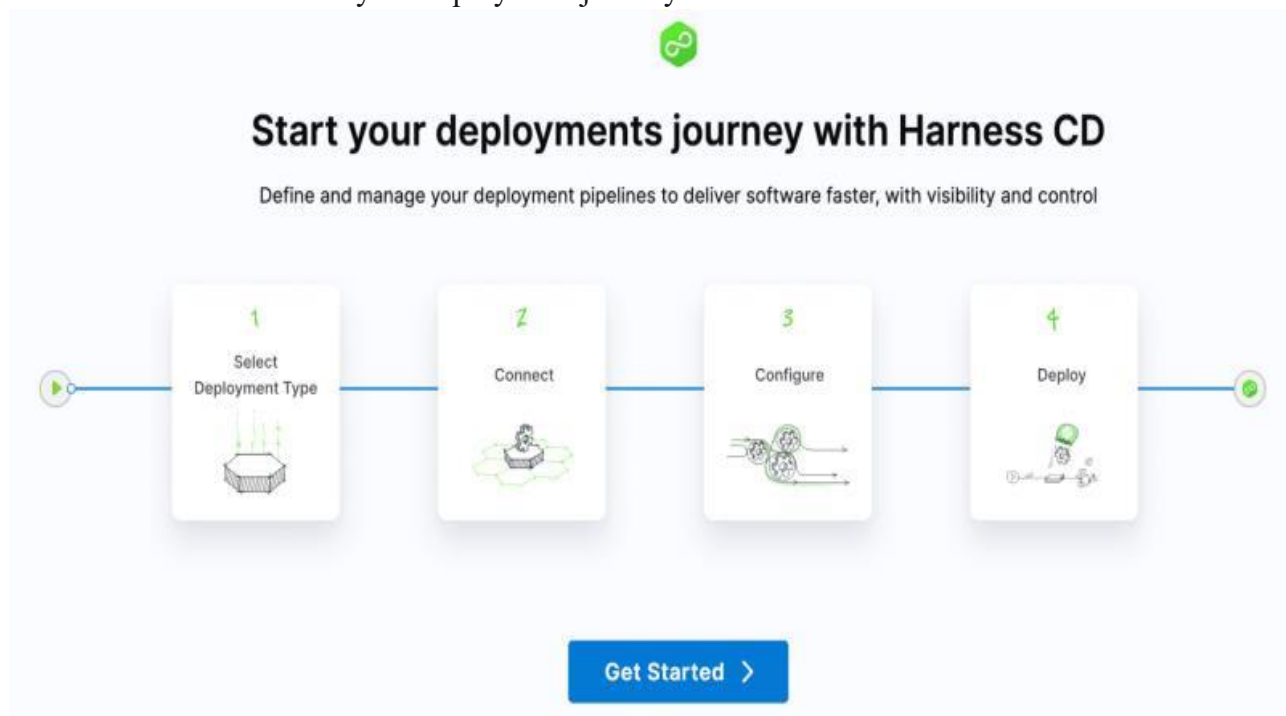
**Continuous Delivery**
Define pipelines that enable you to improve your deployment times, manage your integrations and view your services all in one place.

**Continuous Integration**
Take your software build processes to the next level with Harness Continuous Integration.

**Feature Flags**
Take your feature rollout processes to the next level using our Harness Feature Flags

Continuous Delivery
[Go to Module](#)

Go to the module and start your deployment journey.



The set up is very straightforward, as shown in the above image; you can deploy your application in just four simple steps.

Select your deployment type i, e Kubernetes and click 'Connect to Environment'.

Select deploym...

Connect to Environment

Configure Service

Run Pipeline

Select deployment type



Kubernetes

[Click here for other deployment types](#)

Next: Connect to Environment >

Connect to your Kubernetes environment with Delegate. A Delegate is a service that runs on your infrastructure to execute tasks on behalf of the Harness platform.



Connect Harness to your environment

i A Delegate is a service that runs on your infrastructure to execute tasks on behalf of the Harness platform.

How do you want to install the Delegate?

You can install the Delegate in your target environment.

Kubernetes

Docker

Install the Delegate

1 Download YAML

↓ Download YAML file

Preview Yaml

2 Install on your Cluster

Set up and connect to your target cluster.

Install the Delegate with the following command:

```
$ kubectl apply -f harness-delegate.yml
```

< Back

Next: Configure Service >

Download the Delegate YAML file and install it on your Kubernetes cluster by applying the `kubectl apply` command as stated in the above step.

Make sure to execute the command `kubectl apply -f harness-delegate.yaml` in the right path where you downloaded your delegate YAML file.

Ensure your Delegate installation is successful.

Next, configure the service and add the manifest details.

Select deploym...

Connect to Environment

Configure Service

Run Pipeline

Configure Service

sample_service 

Manifest type

Kubernetes

Helm

Manifest location

Harness Filestore

GitHub

GitLab

Bitbucket



Connect to a GitHub

1

Select your Authentication method

GitHub Account URL

https://github.com/pavanbelagatti/notes-app-cicd

Please provide a repository to test the credentials. This is required just for checking connectivity. The connector will still be created at account level.

Test Repository

notes app cicd

OAuth

Access Token

< Back

Next: Create a Pipeline >

After adding all the details, click 'Create a Pipeline'.

Select deploym...

Connect to Environment

Configure Service

Run Pipeline

Your pipeline is ready to go!



Deployment type

Kubernetes

Connect to Environment

✓ Success

Delegate run as

K8sCluster

✓ Success

Environment details

Connector: K8s Cluster

Environment: dev

Infrastructure: dev-cluster

Namespace: default

Service Configuration

✓ Success

Service name: sample_service

Manifest type: K8sManifest

Manifest storage: Harness

Artifact storage: -

Run Pipeline

Check if all the connections are successful. Once everything looks fine, click on 'Run Pipeline'.

Run Pipeline ⓘ

All Stages ▾

VISUAL

YAML

✕

No runtime inputs are required for the execution of this Pipeline

☐ Skip preflight check

☐ Notify only me about execution status

Run Pipeline

Cancel

Click on 'Run Pipeline' to see the successful deployment.

notes-app-cicd | main | First commit | 35b422a

Pipeline | Inputs | Policy Evaluations | Artifacts | Commits | Tests | Security Tests | Error Tracking | Resilience

2 stages

- Build Node App
- Deploy
 - Service 1s
 - Infrastructure Section
 - Resource Constraint
 - Rollout Deployment 14s

Console Logs

Wrap Up

```

4 INFO 07/02/2023 13:59:11
5 INFO 07/02/2023 13:59:11 Name: notes-app-deployment
6 INFO 07/02/2023 13:59:11 Namespace: default
7 INFO 07/02/2023 13:59:11 CreationTimestamp: Tue, 07 Feb 2023 08:11:37 +0000
8 INFO 07/02/2023 13:59:11 Labels: app=notes-app
9 INFO 07/02/2023 13:59:11 Annotations: deployment.kubernetes.io/revision: 2
10 INFO 07/02/2023 13:59:11 kubectl.kubernetes.io/last-applied-configuration:
11 INFO 07/02/2023 13:59:11 ('apiVersion': 'apps/v1', 'kind': 'Deployment', 'metadata': {'name': 'notes-app-deployment', 'namespace': 'default'}, 'spec': {'replicas': 2, 'selector': {'matchLabels': {'app': 'notes-app'}}, 'strategy': {'rollingUpdate': {'maxUnavailable': '25%', 'maxSurge': '25%'}}})
12 INFO 07/02/2023 13:59:11 Selector: app=notes-app
13 INFO 07/02/2023 13:59:11 Replicas: 2 desired | 2 updated | 2 total | 2 available | 0 unavailable
14 INFO 07/02/2023 13:59:11 StrategyType: RollingUpdate
15 INFO 07/02/2023 13:59:11 MinReadySeconds: 0
16 INFO 07/02/2023 13:59:11 RollingUpdateStrategy: 25% max unavailable, 25% max surge
17 INFO 07/02/2023 13:59:11 Pod Template:
18 INFO 07/02/2023 13:59:11 Labels: app=notes-app
19 INFO 07/02/2023 13:59:11 harness.io/release-name=release-5d8f92efe466a8e331f8b80e9185efdb8993a5a7c
20 INFO 07/02/2023 13:59:11 Containers:
21 INFO 07/02/2023 13:59:11 notes-app-deployment:
22 INFO 07/02/2023 13:59:11 Image: pavansa/notes-app
23 INFO 07/02/2023 13:59:11 Port: 3000/TCP
24 INFO 07/02/2023 13:59:11 Host Port: 0/TCP
25 INFO 07/02/2023 13:59:11 Requests:
26 INFO 07/02/2023 13:59:11 cpu: 100m
27 INFO 07/02/2023 13:59:11 Environment: <none>
28 INFO 07/02/2023 13:59:11 Mounts: <none>
29 INFO 07/02/2023 13:59:11 Volumes: <none>
30 INFO 07/02/2023 13:59:11 Conditions:
31 INFO 07/02/2023 13:59:11 Type Status Reason
32 INFO 07/02/2023 13:59:11 ----
33 INFO 07/02/2023 13:59:11 Available True MinimumReplicasAvailable
34 INFO 07/02/2023 13:59:11 Progressing True NewReplicaSetAvailable
35 INFO 07/02/2023 13:59:11 OldReplicaSets: <none>
36 INFO 07/02/2023 13:59:11 NewReplicaSet: notes-app-deployment-dccb65484 (2/2 replicas created)
37 INFO 07/02/2023 13:59:11
38 INFO 07/02/2023 13:59:11 Done.

```

Congratulations! We successfully deployed our application successfully on Kubernetes using Harness. Now, we can easily automate the deployment using the Harness CD module.

You can automate your CD process by adding Triggers. When any authorised person pushes any new code to your repository, your pipeline should get triggered and do CD. Let's see how to do that.

In the pipeline studio, you can click the 'Triggers' tab and add your desired trigger.

Pipeline Studio | Input Sets | Triggers | Execution History

Save | Discard | Run



Triggers are used to automate the execution of pipelines based on some event like new artifact/manifest, or run on a schedule or an external webhook.

Add New Trigger

Click on 'Add New Trigger' and select 'GitHub'.

Triggers

Webhook



GitHub



GitLab



BitBucket



Azure Repos



Custom

Add the required details and continue. As you can see, we are selecting 'Push' as our event. So whenever any authorised push happens to our repository, the pipeline should trigger and run.

Trigger: Push Trigger Enabled

VIS

✓ Configuration > ✓ Conditions > ✓ Pipeline Input

Trigger Configuration ⓘ

Name ⓘ Id ⓘ : Push_Trigger

Push Trigger

Description (optional) ⓘ ✎

Tags (optional) ⓘ ✎

Listen on New Webhook ⓘ

Payload Type ⓘ

GitHub

Connector ⓘ

Github

ACCOUNT

+ Add

Repository URL

https://github.com/pavanbelagatti/notes-app-exa

Event ⓘ

Push

☐ Auto-abort Previous Execution ⓘ

If a branch is updated, Harness will automatically cancel active builds for the same branch if invoked by the same Trigger.

Configure Secret (Optional)

Continue >

Cancel

On New Webhook Enabled

✓ Configuration > ✓ Conditions > ③ Pipeline Input

Pipeline Input ⓘ

+ Select Input Set(s)

CI Codebase

Build Type

☒ Git Branch ☐ Git Tag ☐ Git Pull Request

Branch Name

<+trigger.branch>

f(x)

You can see your newly created trigger in the ‘Triggers’ tab.



@ notes-app-example > Pipelines >

Build NodeJS

Pipeline Studio Input Sets **Triggers** Execution History

+ New Trigger

Search

TRIGGER	STATUS	ACTIVITY	LAST ACTIVATION	WEBHOOK	ENABLED
 Push Trigger Id: Push_Trigger		1 Activations in Last 7 days	1 hour ago ●		<input checked="" type="checkbox"/>

Now, whenever any authorised person pushes any code changes to your main/master repository, the pipeline automatically gets triggered.