Basics in OpenShift

[**Openshift 1**](#_iiviljltep74)

[**Projects 2**](#_l9huqokx55g7)

[Creating project using cli command 2](#_dfgdf6vhomk3)

[Creating project using web console 3](#_ui0v0io5sxt2)

[Creating a project using yaml file 4](#_bmicn18kcgbq)

[**Deployments 4**](#_y73s5uxauf6)

[Creating deployment using cli command: 4](#_i7abqvs1vuhb)

[Creating deployment using web console: 5](#_2b88ne2xdvg0)

[Creating deployment using a yaml file: 7](#_idf9k2cnv5ka)

[**Services 9**](#_v23ipq2nwzwz)

[Creating a Service with cli command: 9](#_9jqv29m199ag)

[Creating a Service using web console: 10](#_yvm03kf3g52d)

[Creating a Service with yaml file: 12](#_gv1gay6avrbp)

[**Routes 13**](#_miibrhmw1my)

[Creating a Route with cli command: 13](#_eagttgf4tmjy)

[Creating a route using web console: 14](#_iv4bkmvtdbv3)

[Creating a Route with yaml file: 15](#_ax4fzk1f2rlj)

[**ConfigMaps 16**](#_1i6ufj9lyyw7)

[Creating a ConfigMap with cli command: 16](#_8ceu4fcuqiyh)

[Creating a ConfigMap using web console: 17](#_8gfgiwjsas9z)

[Creating a ConfigMap with yaml file: 18](#_xqie4r5pxdtm)

[**Secrets 19**](#_1hedo8tpzx6c)

[Creating a secret using cli command 19](#_d69y4n12529h)

[Creating a Secret using web console: 20](#_aw388y4g216g)

[Creating a secret using yaml file 22](#_b7rctmlnhnkf)

[**Application - End to end deployment 23**](#_a2tabnbbaapu)

[Creating application using web console: 23](#_6v3201w4t1c2)

[Creating application using yaml file: 27](#_33g1i91gygqz)

# OpenShift

Openshift is an open-source container orchestration platform that is to automate the management of containerized applications. Openshift provides a complete ecosystem for building, deploying and scaling containerized applications.

# Projects

Project is a top-level organizational unit that provides isolation and a controlled environment for managing and deploying applications.

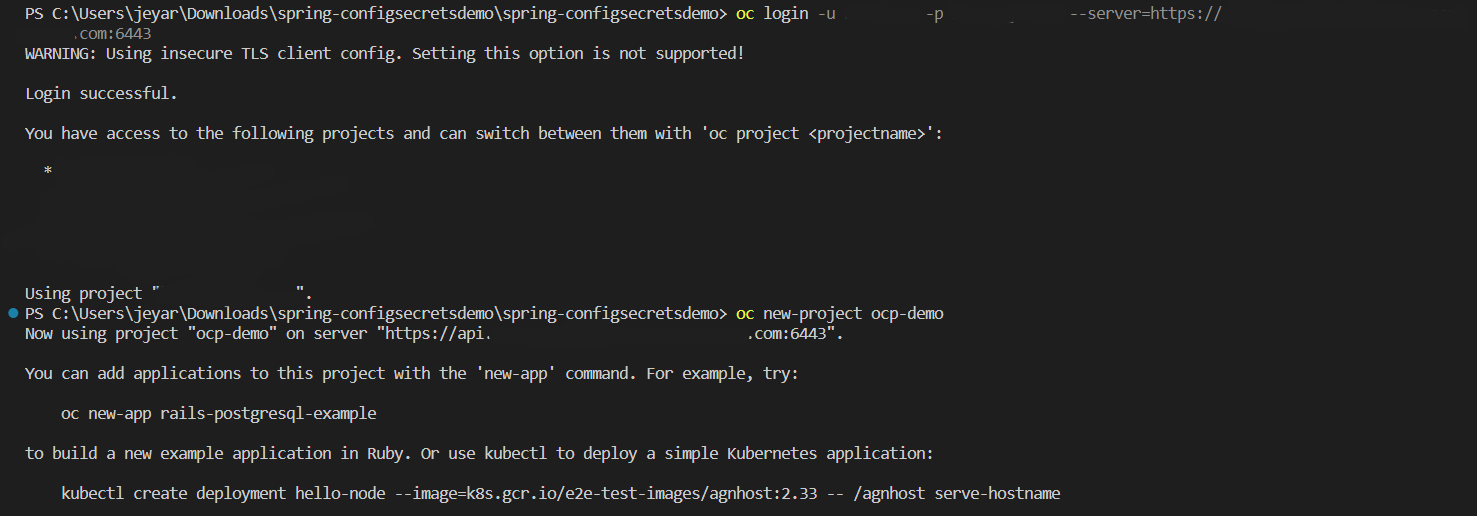
## Creating project using cli command

**Step1:** Login to the ocp

|  |
| --- |
| **oc login -u <username> -p <password> –server=<hostname>:6443** |

**Step2:** To create a new project

|  |
| --- |
| **oc new-project <project-name>** |



You can view the projects using the following commands

|  |
| --- |
| **oc projects (or) oc get projects** |

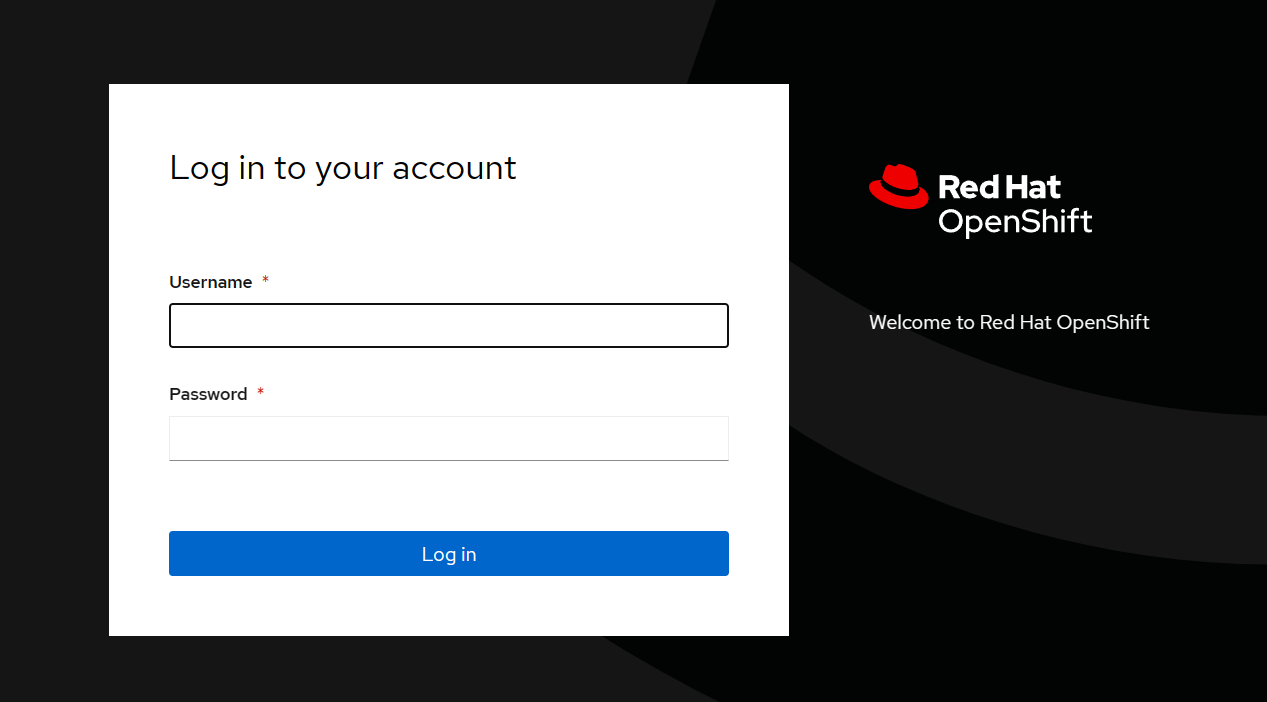
You can delete the project using the following command

|  |
| --- |
| **oc delete project <project-name>** |

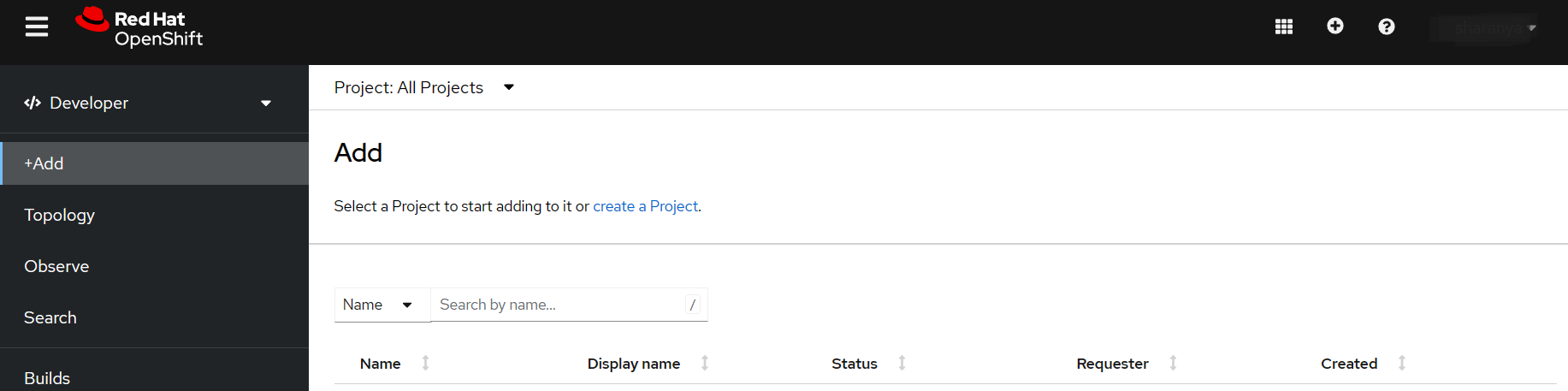


## Creating project using web console

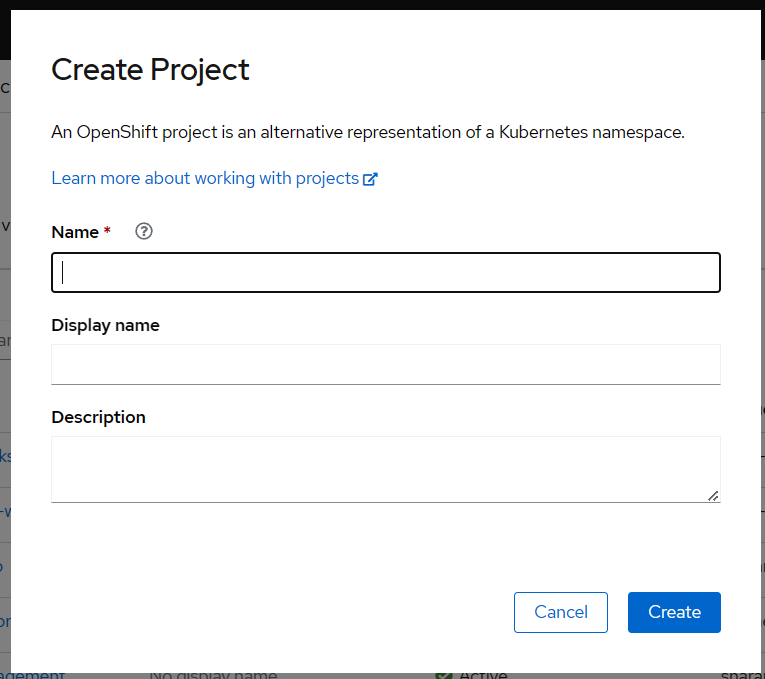
**Step1:** Login to the openshift web console



**Step2:** Click on the create a project



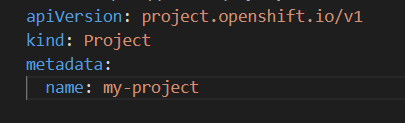
**Step3:** A dialogue box will open and then add the necessary fields and then click on Create



A project will be created

## Creating a project using yaml file

**Step1:** Create a yaml file for the project with the name project.yaml



**Step2:** To create a project

|  |
| --- |
| **oc apply -f <project.yaml>** |

# Deployments

Deployment is a k8s workload that manages the lifecycle of pods. It is defined using YAML files. Deployment and application are related but deployment is a specific type of Kubernetes resource focused on managing replicas of pods, an application in OpenShift encompasses a broader set of resources that make up the software or service you are deploying and running.

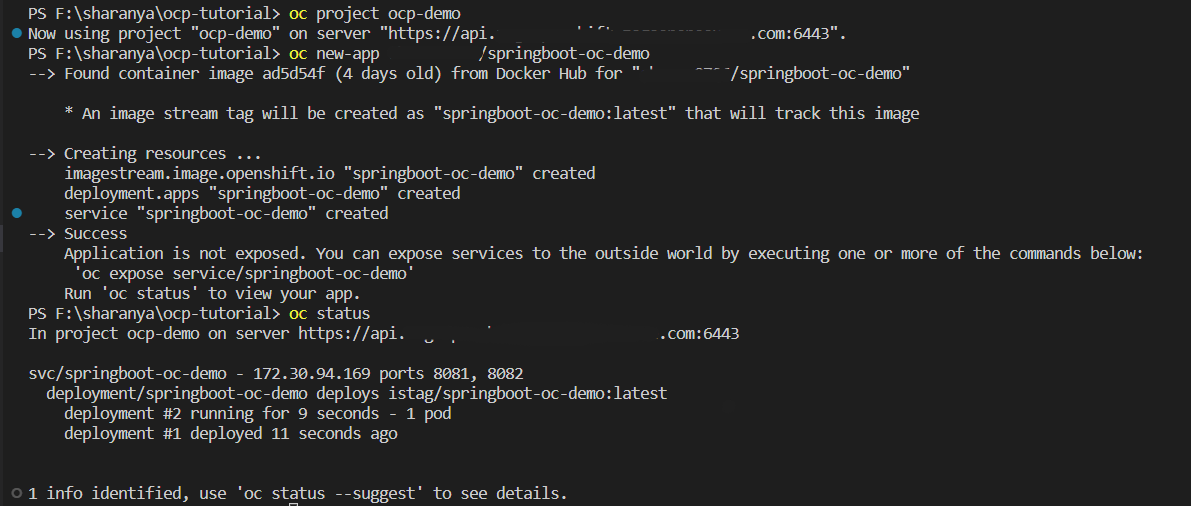
## Creating deployment using cli command:

**Step1:** Switch to the project you want to create the deployment using the following command:

|  |
| --- |
| oc project <project-name> |

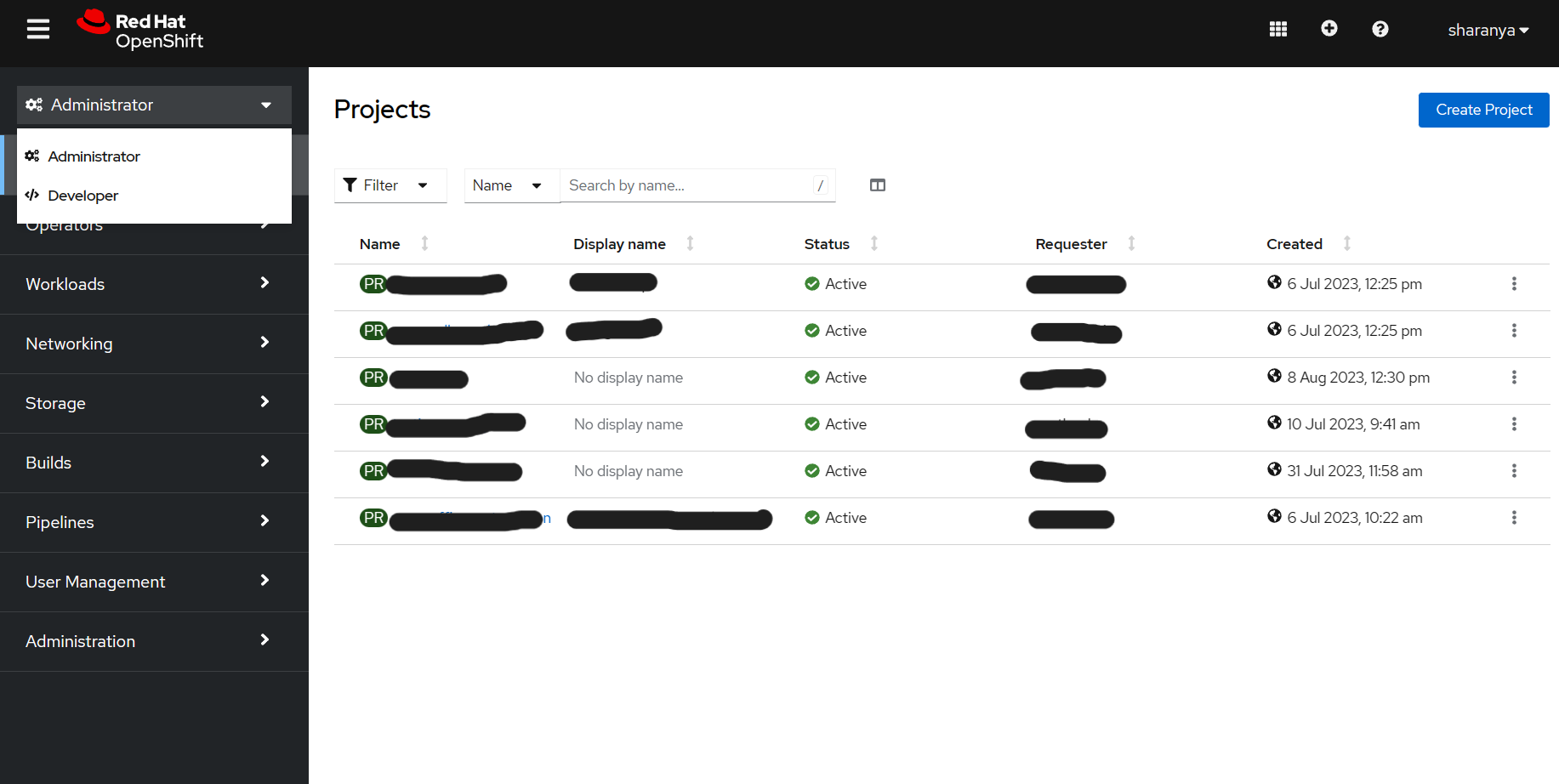
**Step2:** Create a deployment using the following command:

|  |
| --- |
| oc new-app <container-image-name> |

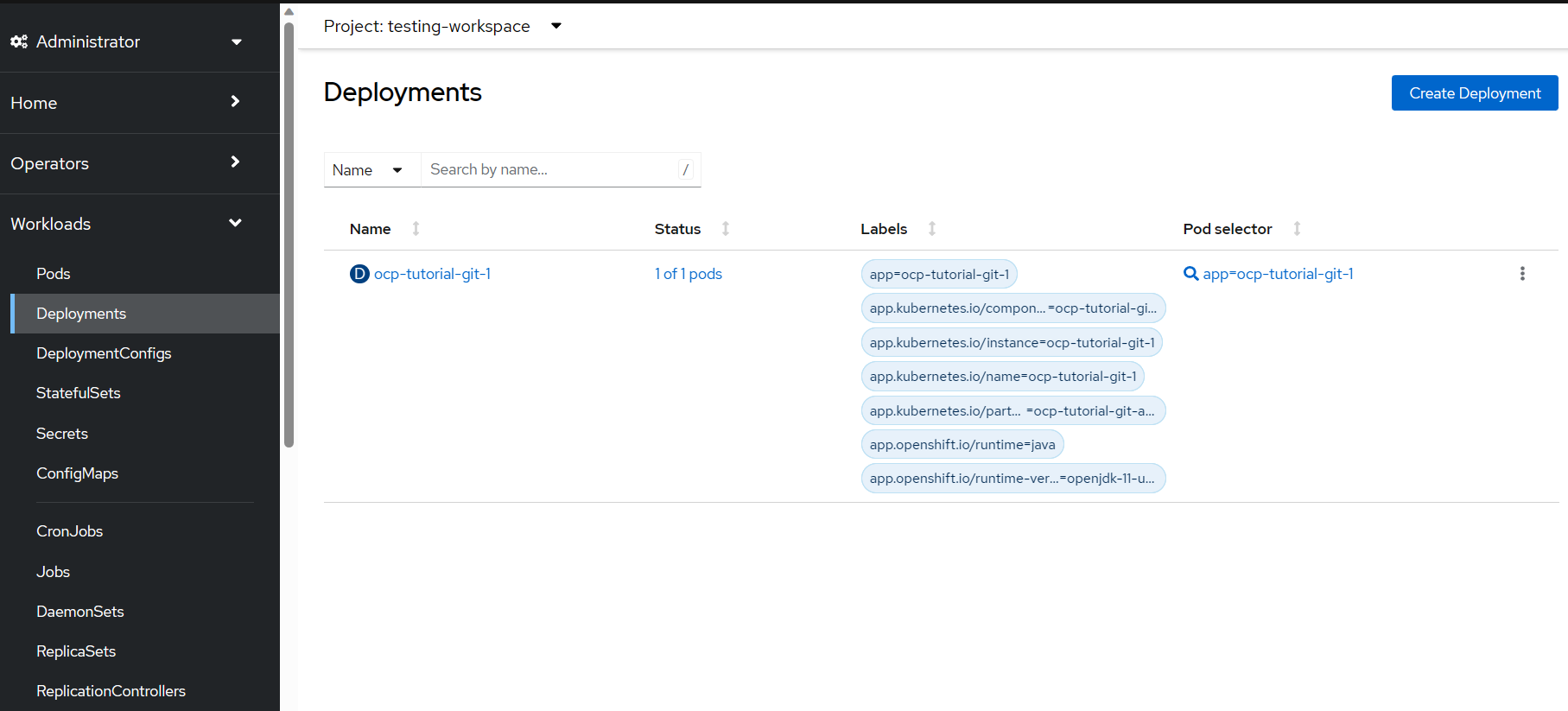


## Creating deployment using web console:

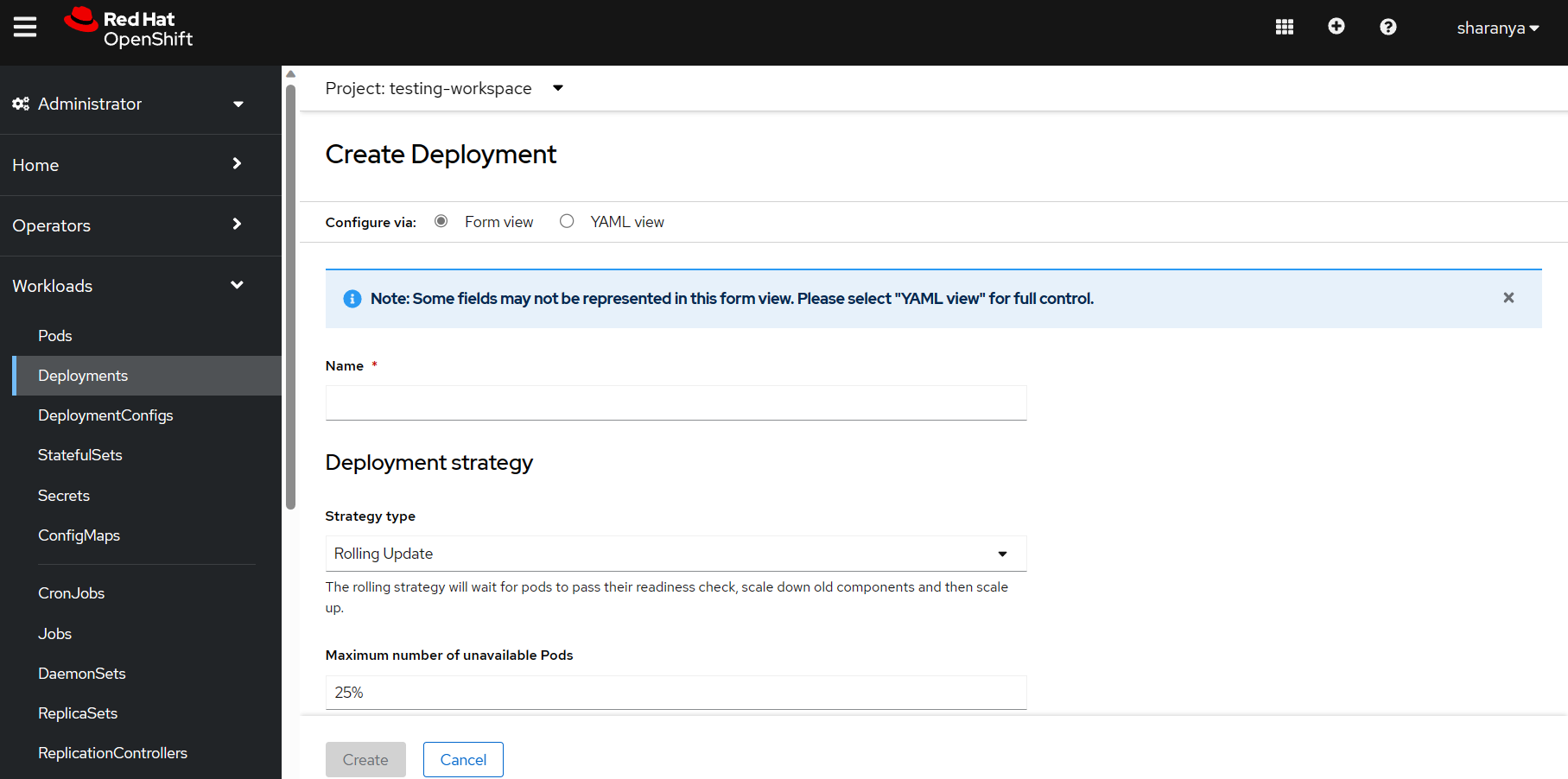
**Step1:** Select the Administrator from the top left drop-down



Step2: Then click on the Workloads and then on Deployment

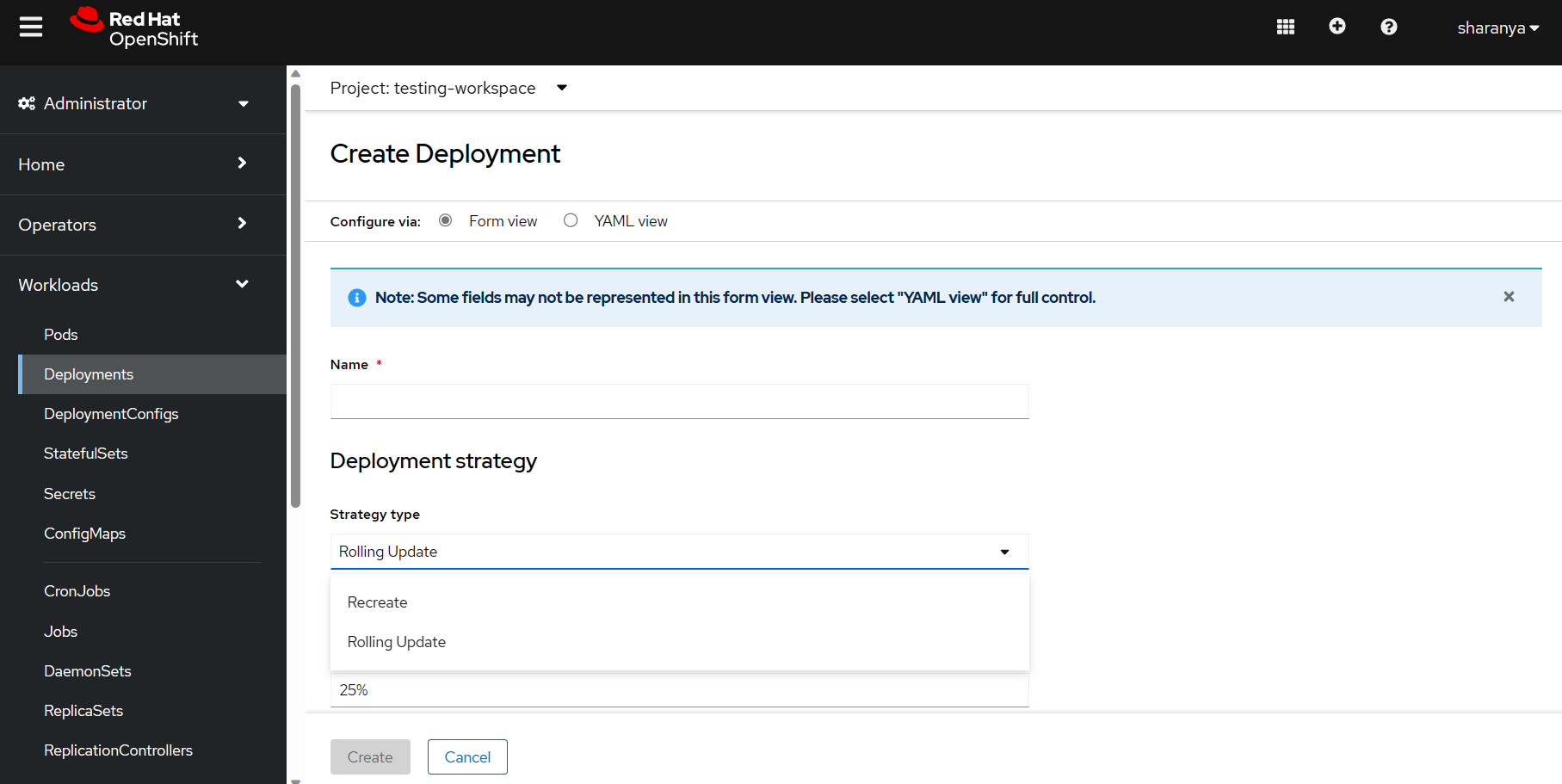


Step3: Click on Create Deployment



Deployment can be created either in Form or YAML view

Step4: Fill in the name and the default strategy type is Rolling Update and any environmental variables need to be included can also be included and then click on Create



The deployment will be created

## Creating deployment using a yaml file:

* **Step 1:**

Create a deployment file as required



Explanation:

**Kind:** defines the resource type

**apiVersion:** Specifies the Kubernetes API version for the resource.

**Metadata:** contains the deployment metadata information

**Name:** defines the name of the deployment. It should be unique within the namespace

**Namespace:** defines the name of the namespace

**Spec:** The specification section defines the desired state and behavior of the deployment. It defines how the Deployment should create and maintain replicas of the pods and how it should handle updates and rollbacks.

**Replicas:** defines the number of pods to be created

**Template:** defines the template for the pods to be created. It includes the container specifications and any other settings needed for the pods.

**Spec:** defines the specification of the pods

**Containers:**

**Name:** defines the container name

**Image:** defines the name of the image to be containerized

**Ports:** defines the ports for running the container

**Resources:** defines the CPU and memory requirements and limits for the container to use.

**Limits:** the maximum limit(hard limit) for a container

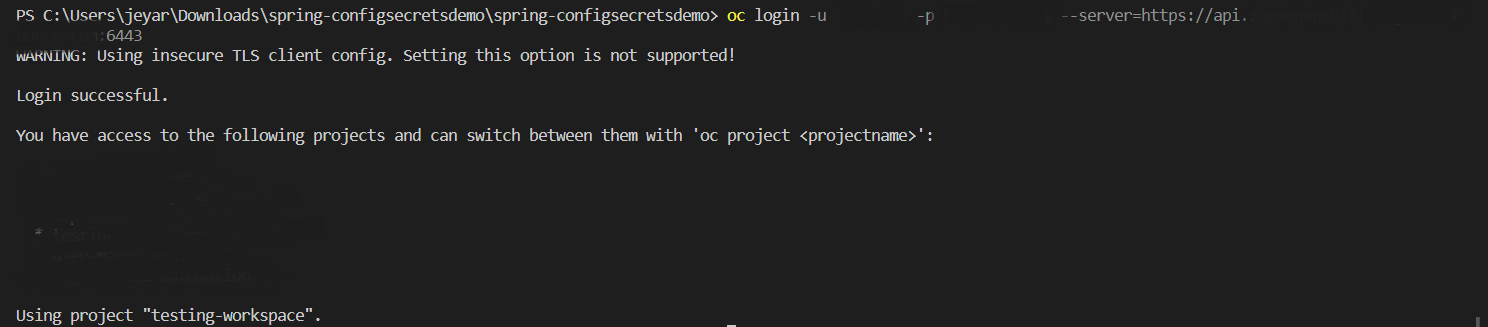
**Requests:** the minimum requirement(soft limit) for a container  **RestartPolicy:** Specifies the containers within the pod should be restarted incase of a container termination.

* **Step 2:**

Login to the OCP

Command:

|  |
| --- |
| oc login -u <username> -p <password> –server=<hostname>:6443 |



* **Step 3:**

Go to the desired project

Command:

|  |
| --- |
| oc project <desired project name> |



* **Step 4:**

Apply the deployment to the project

Command:

|  |
| --- |
| oc apply -f <deployment file name> |



# 

This is how the deployment is created.

# Services

A Service is an abstraction that defines a set of pods and a policy to access them. It provides a consistent way to access and discover pods, even as they are created, deleted, or replaced.

It is like a front door that helps different parts of your application talk to each other easily and reliably.

## Creating a Service with cli command:

* **Step 1:**

Go to the desired project

Command:

|  |
| --- |
| oc project <desired project name> |

* **Step 2:**

Create a service using the following command:

|  |
| --- |
| oc create service <service-type> <service-name> <service-description> |

Example:

|  |
| --- |
| oc create service clusterip myapp-service –tcp=8080:80 |

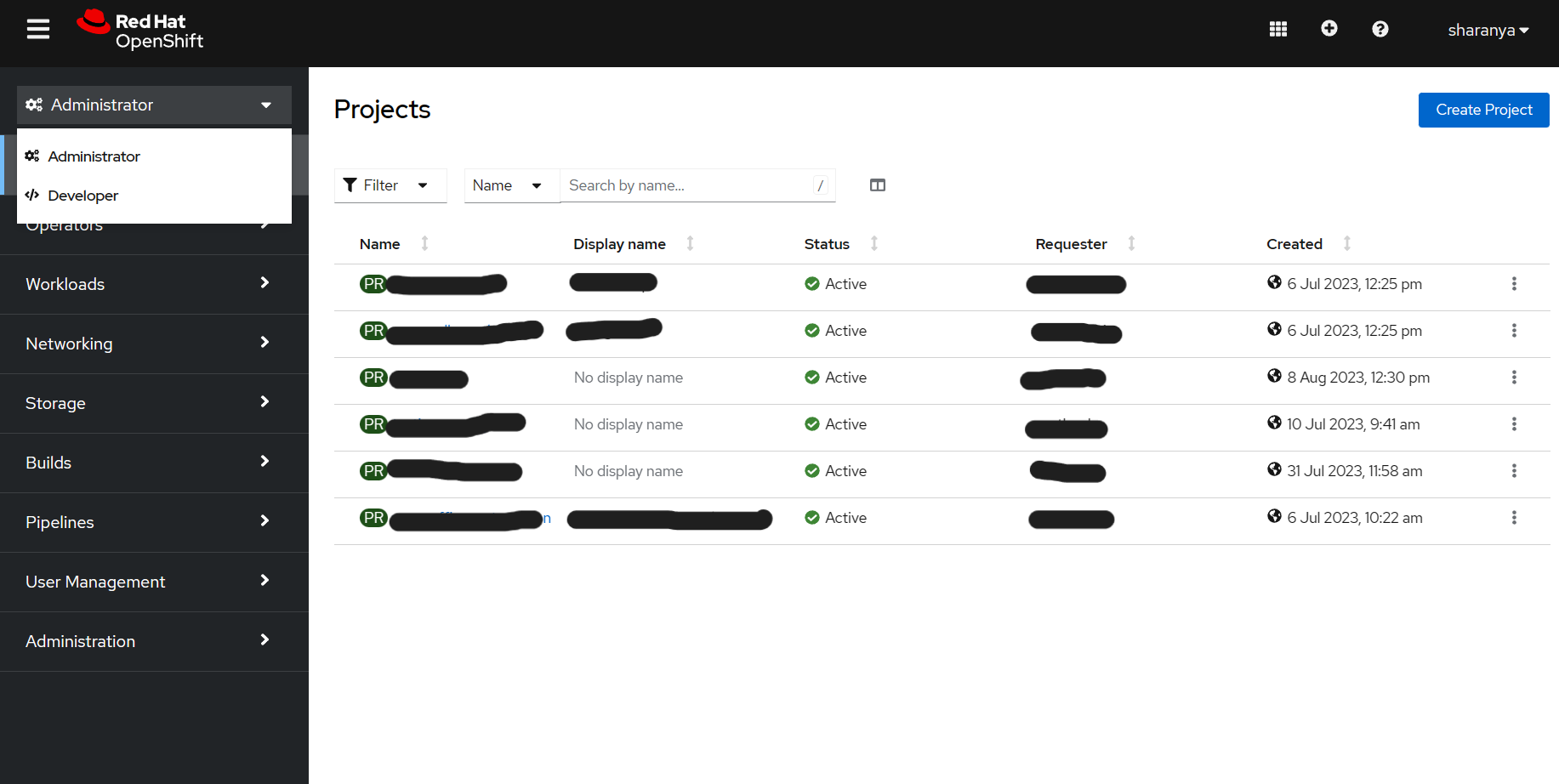
The output will be



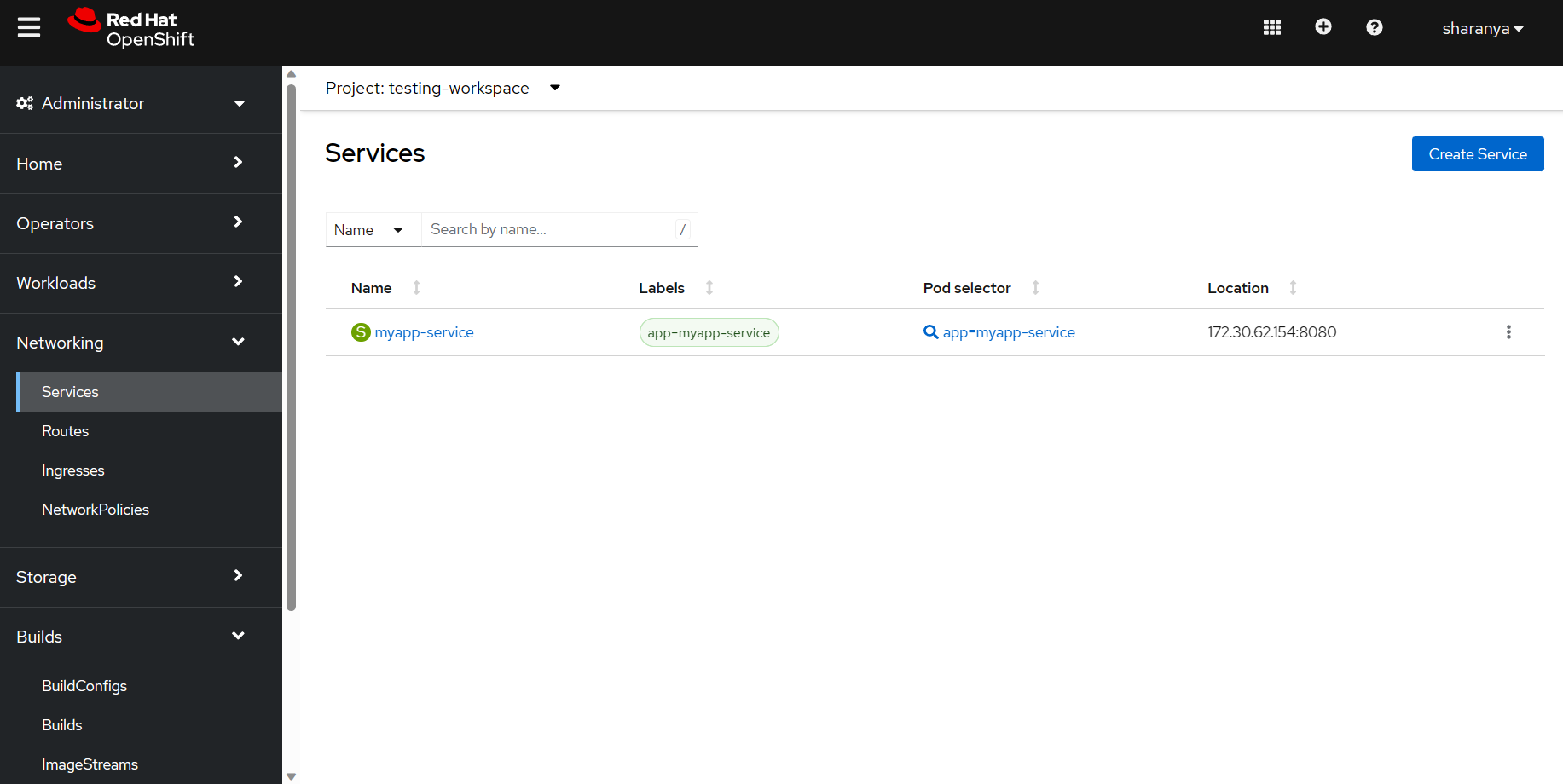
# 

## Creating a Service using web console:

**Step1:** Select the Administrator from the top left drop-down



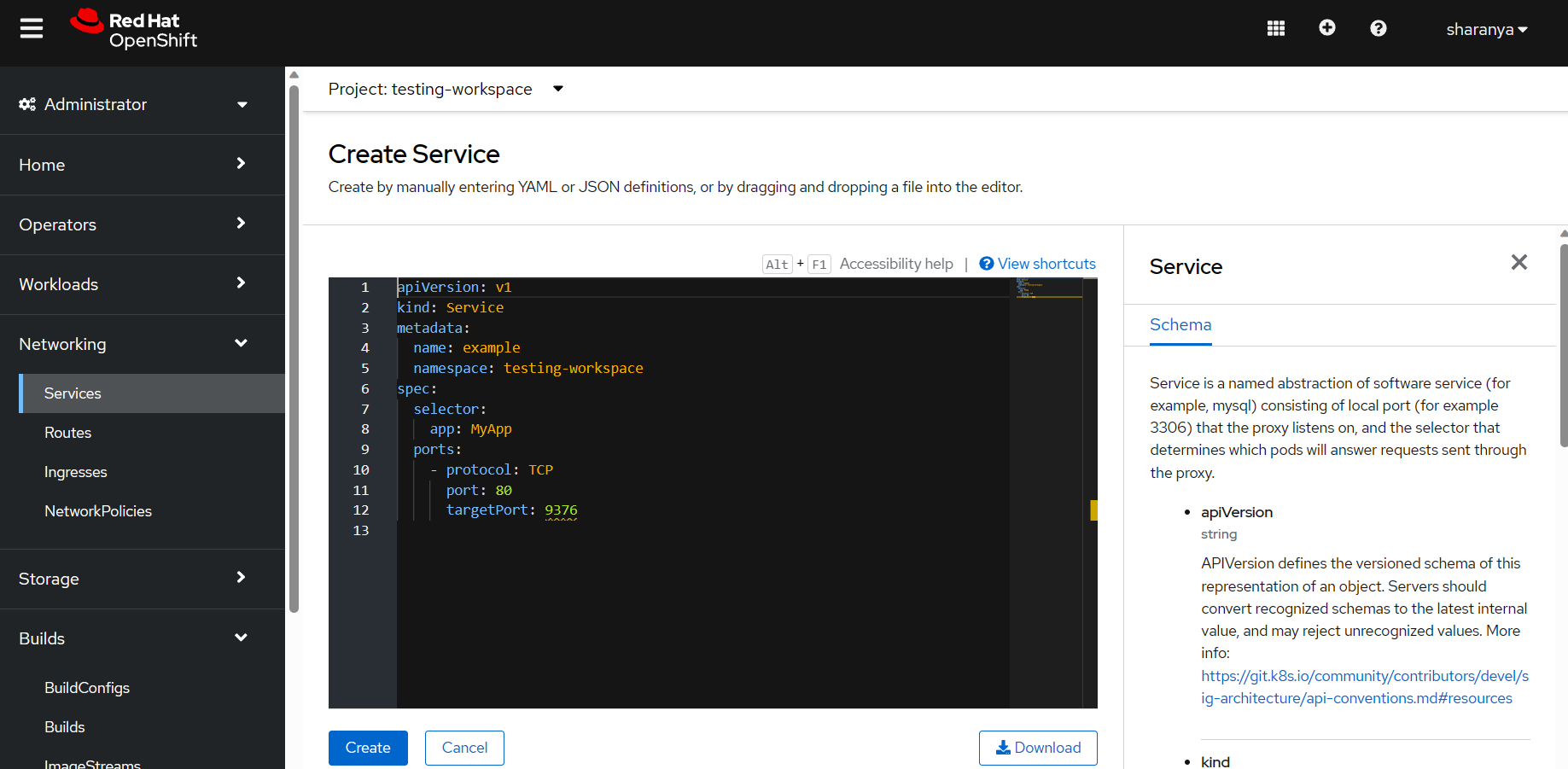
Step2: Then click on the Networking and then on Services



# 

Click on the Create Service

Step3: Write a service yaml file and then click on create.



This is how service will be created from the web console

## Creating a Service with yaml file:

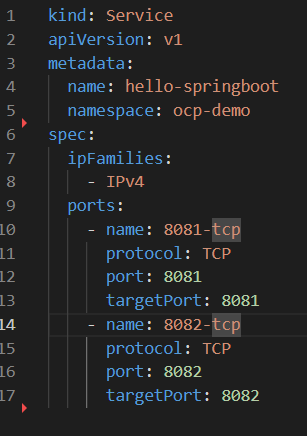
* **Step1:**

Switch to the desire project using the following command:

|  |
| --- |
| **oc project <desired project name>** |

* **Step2:**

Create a service yaml file and apply it.



**Command:**

|  |
| --- |
| **oc apply -f <service>.yaml** |

The output will be



# 

# Routes

Route in OpenShift is a way to expose your application's services to external network traffic, making your application accessible from the internet using a defined hostname and handling secure connections. "Route" is like a signpost or a map that helps people from outside your OpenShift cluster find and access your application. It's like giving your app a public-facing web address that anyone can use to reach it.

## Creating a Route with cli command:

* **Step1:**

Switch to the desire project using the following command:

|  |
| --- |
| **oc project <desired project name>** |

Before creating a Route, make sure you have an OpenShift Service that you want to expose. If you don't have one, you can create it first.

Expose the service

|  |
| --- |
| **oc expose service <service-name> --hostname=<hostname>** |



The output here is for the service we have created before

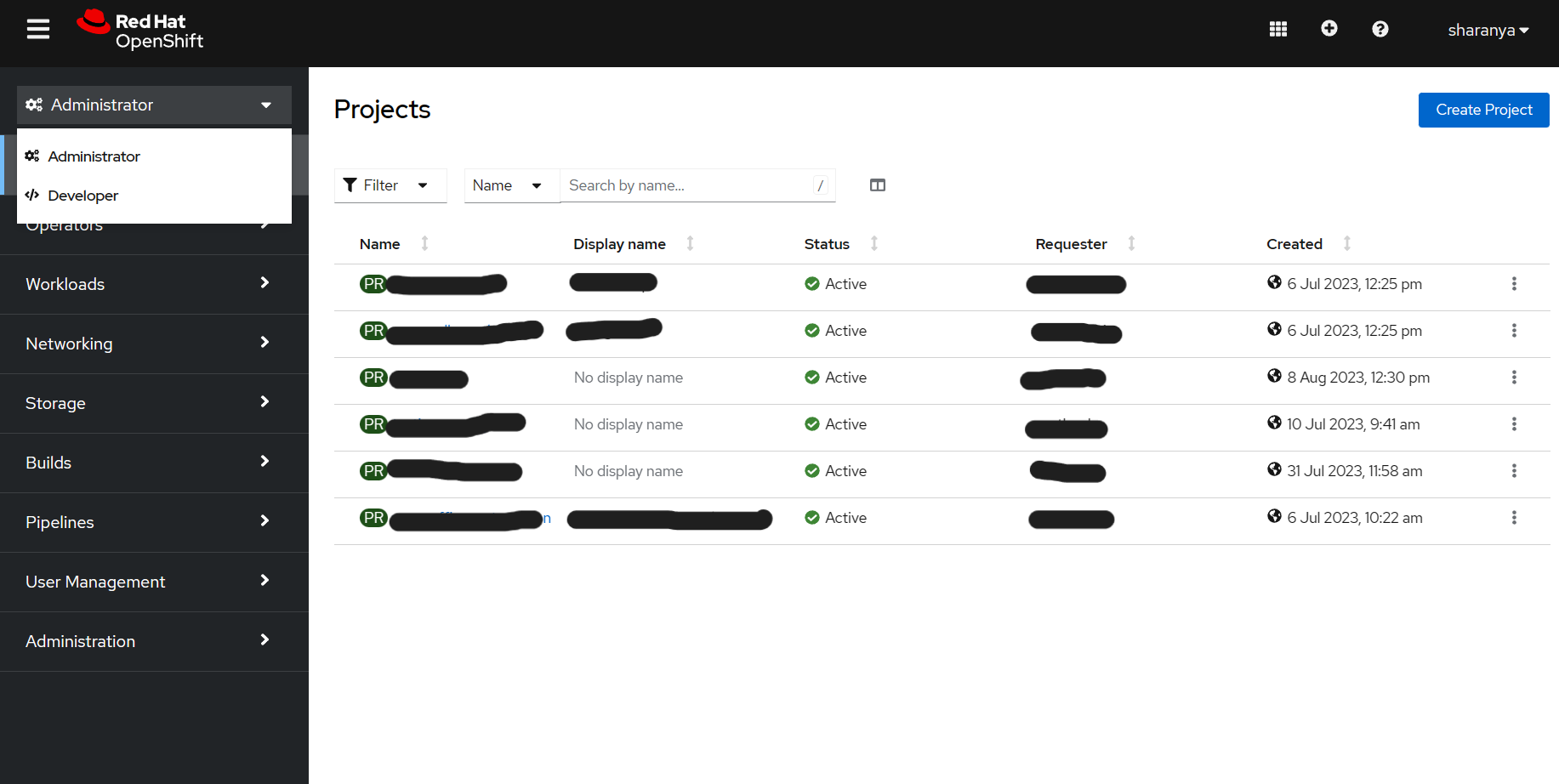
## 

## 

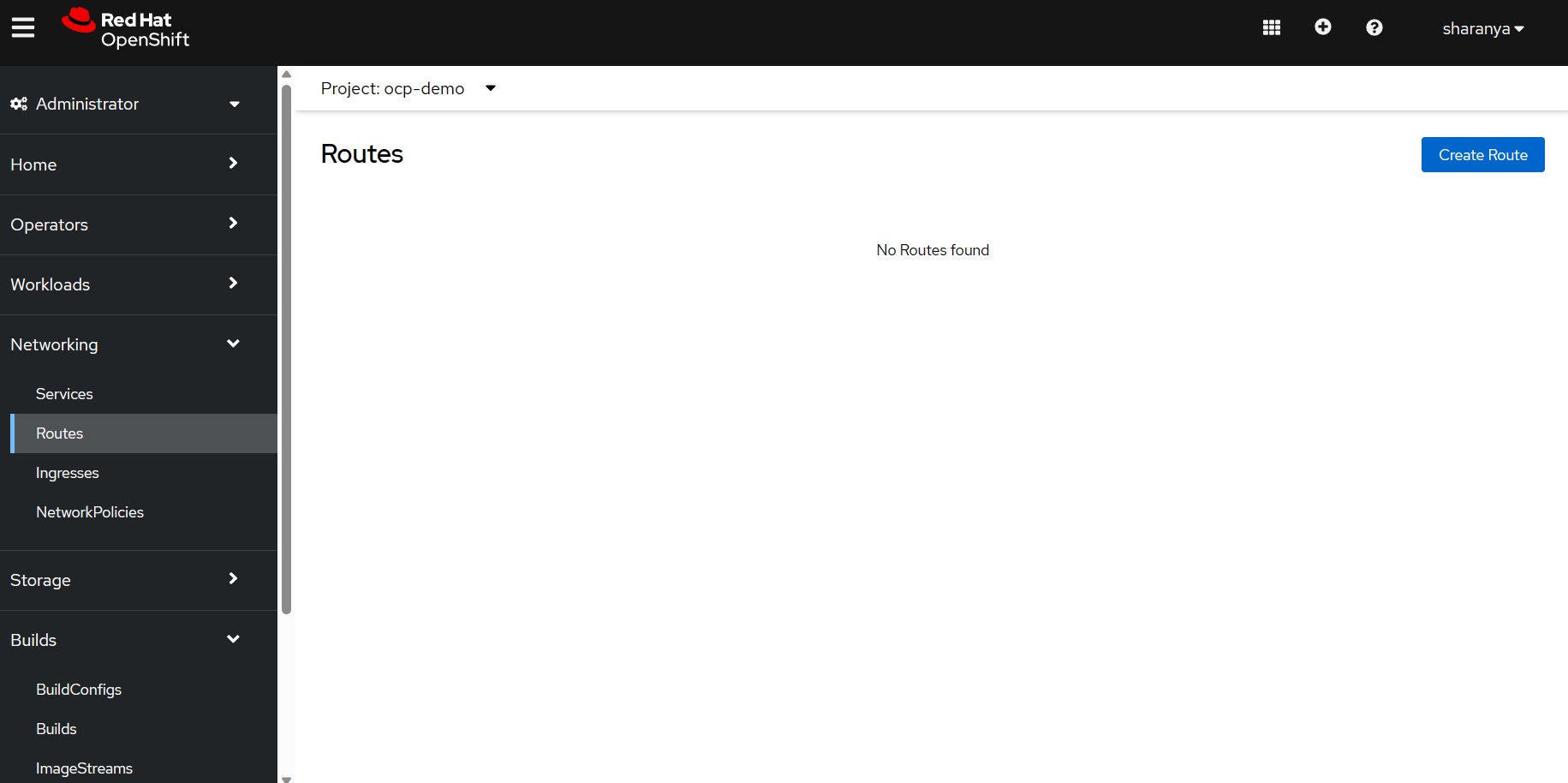
## 

## Creating a route using web console:

**Step1:** Select the Administrator from the top left drop-down



Step2: Then click on the Networking and then on Routes



# 

Then click on Create Route

# 

Fill the necessary fields and select the service you want to expose on the Service drop-down and then click on Create

## Creating a Route with yaml file:

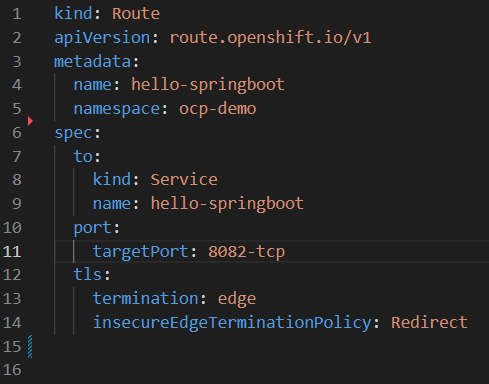
* **Step1:**

Switch to the desire project using the following command:

|  |
| --- |
| **oc project <desired project name>** |

* **Step2:**

Create a yaml file for the route:



**Explanation:**

spec.host: Defines the hostname that users will use to access your application.

spec.to.kind: Specifies that the Route should point to a Service.

spec.to.name: Specifies the name of the Service to expose.

spec.port.targetPort: Specifies the port on the pods that the Service is listening on.

spec.tls.termination: Specifies how TLS/SSL should be terminated. Here, it's set to edge, meaning HTTPS will be terminated at the edge (Route level).

spec.tls.insecureEdgeTerminationPolicy: Redirects insecure HTTP requests to HTTPS.

Command:

|  |
| --- |
| oc apply -f <route>.yaml |

The output will be



# ConfigMaps

ConfigMap is a resource that is used to store configuration data in key-value pairs used by pods, Deployments and other Kubernetes objects. It allows keeping the configurations separate from the application code that can be useful when modifying the configuration without changing the application’s container image or code.

## Creating a ConfigMap with cli command:

* **Step1:**

Switch to the desire project using the following command:

|  |
| --- |
| **oc project <desired project name>** |

* **Step2:**

Create a configmap using the following command:

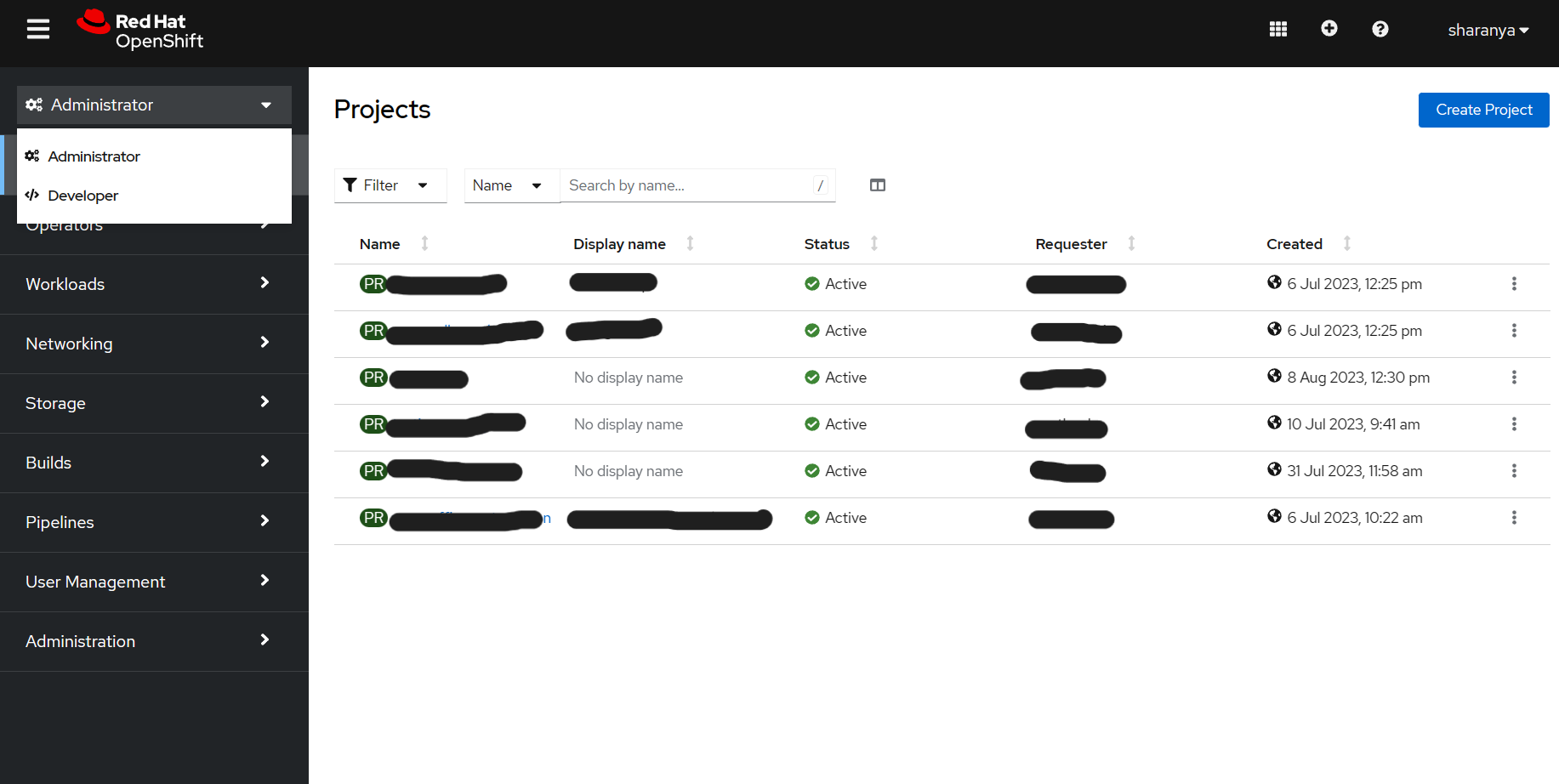
|  |
| --- |
| **oc create configmap <configmap-name> –from-literal=<key1>=<value1> –from-literal=<key1>=<value1>** |

The output will be

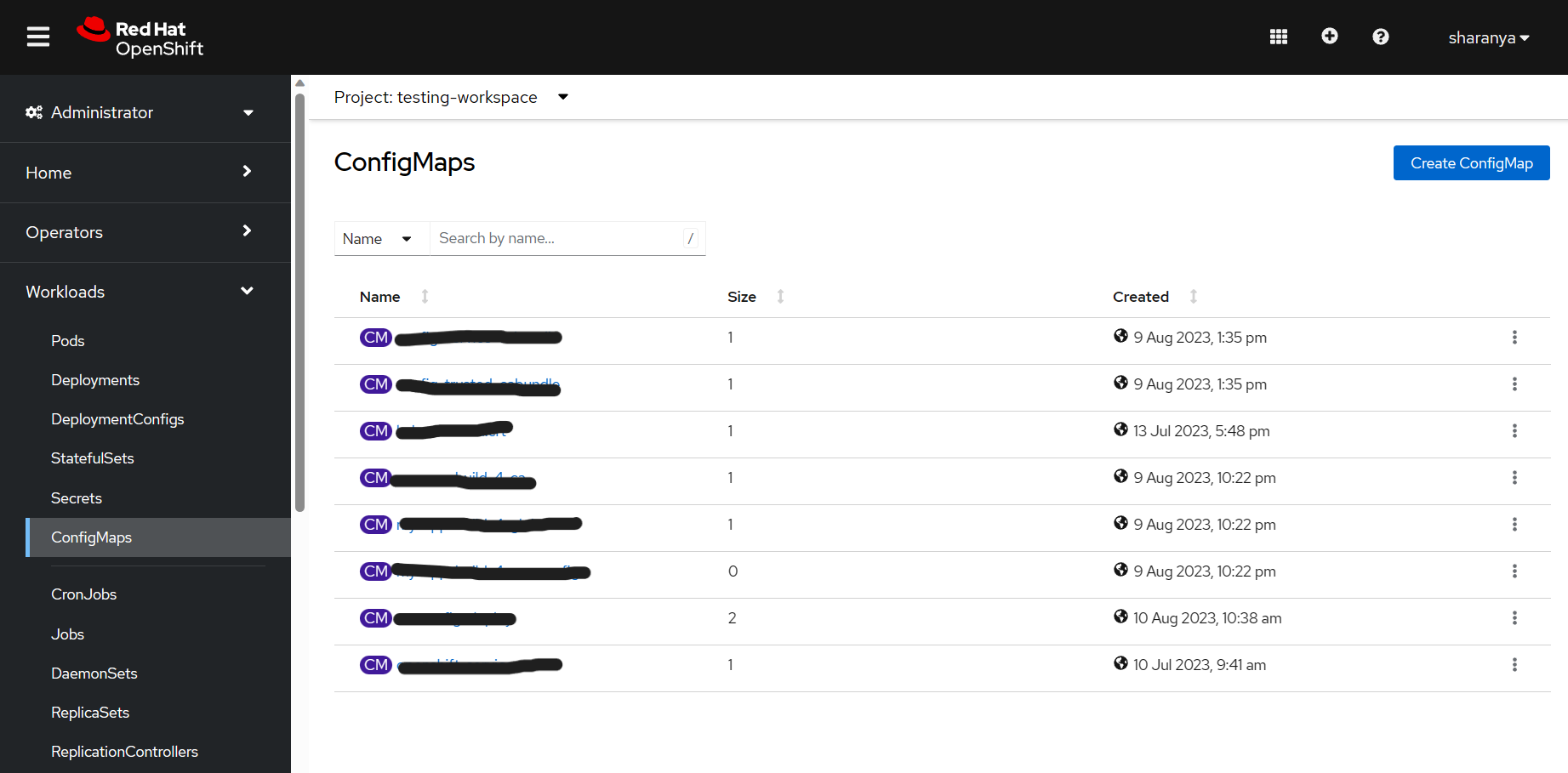


## Creating a ConfigMap using web console:

**Step1:** Select the Administrator from the top left drop-down

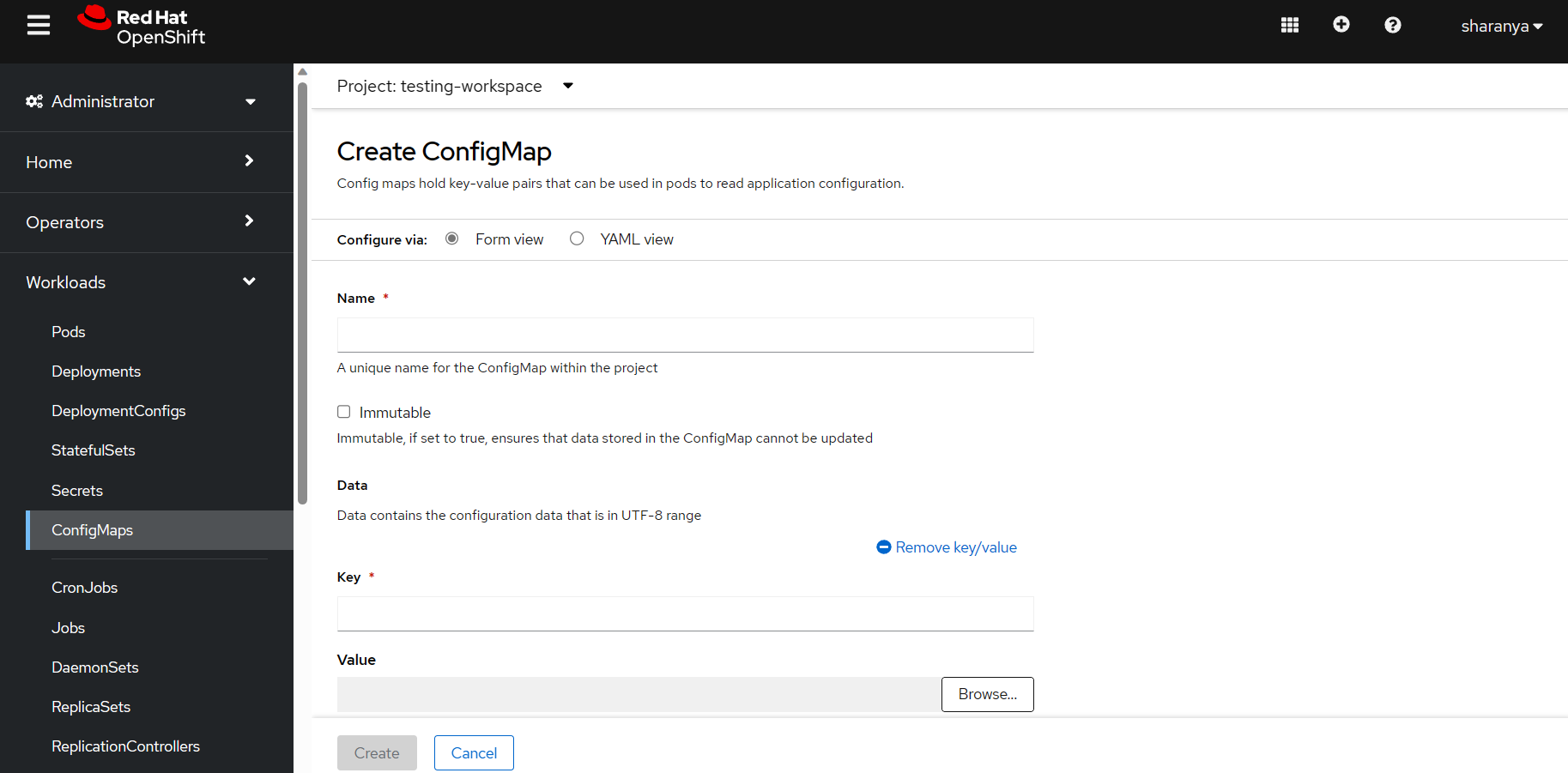


Step2: Then click on the Workloads and then on ConfigMap



Select the project from the Project drop-down on the top and then create a configMap by clicking on the Create ConfigMap button.

**Step3:** Fill up the required fields and then click on Create



This will create a ConfigMap in the project you want to use.

## Creating a ConfigMap with yaml file:

* **Step1:**

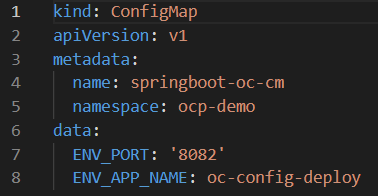
Switch to the desire project using the following command:

|  |
| --- |
| **oc project <desired project name>** |

* **Step2:**

Create a configMap yaml file and apply it.

**Example:**



**Command:**

|  |
| --- |
| oc apply -f <configmap file>.yaml |

**Explanation:**

data :

ENV\_PORT : an environment variable for defining the port

ENV\_APP\_NAME: an environment variable for defining the app name

# Secrets

Secrets are resources used to manage sensitive data and separate from the application or configuration. It is also a YAML file and data is encrypted and used in application as an environmental variable.

## Creating a secret using cli command

* **Step1:**

Switch to the desire project using the following command:

|  |
| --- |
| **oc project <desired project name>** |

* **Step2:**

Create a secret using the following command:

|  |
| --- |
| oc create secret generic <secret name> –from-literal=username=<user name> –from-literal=password=<password> |

The Output will be

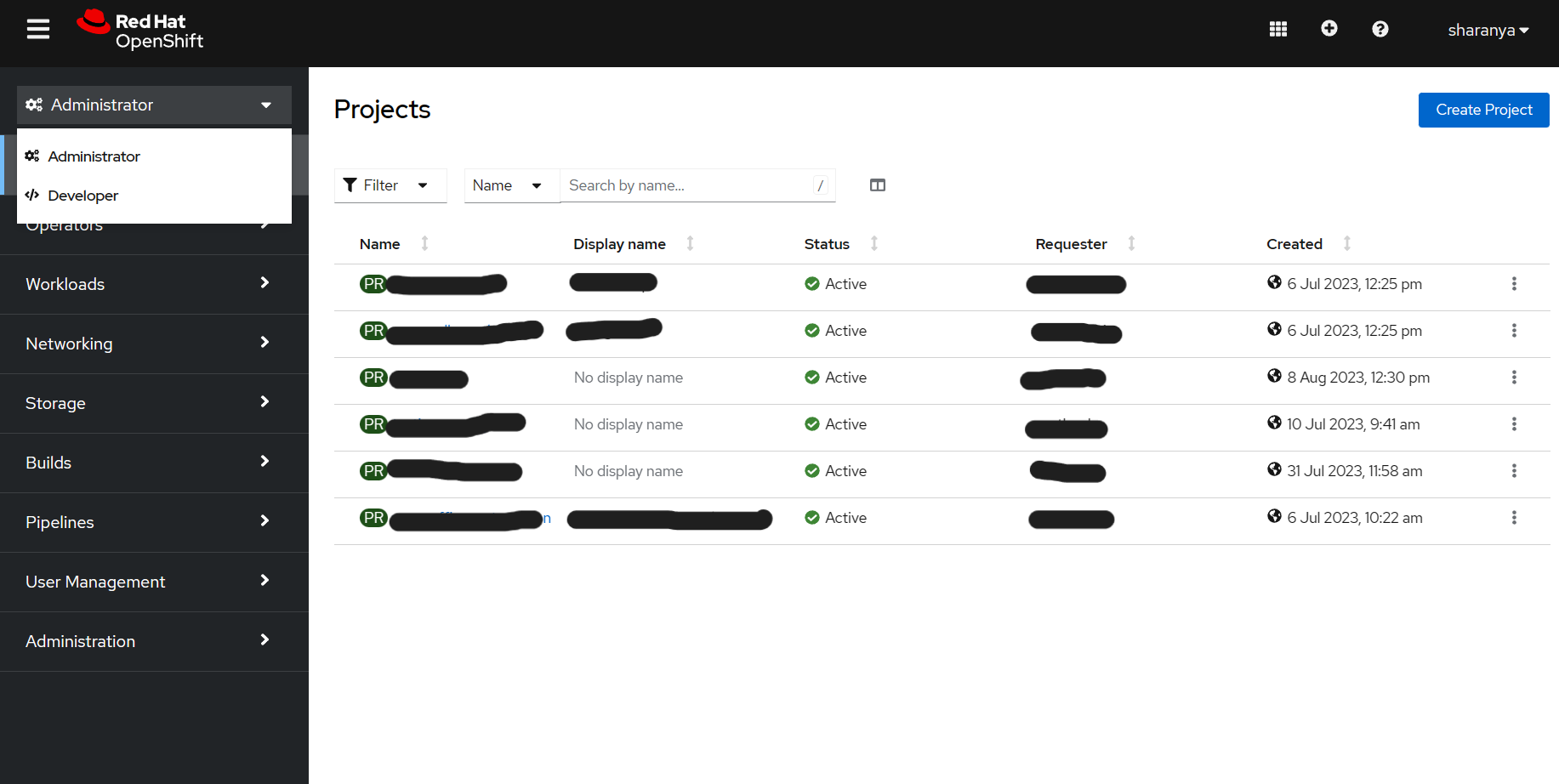


Here the username and password are secrets

## 

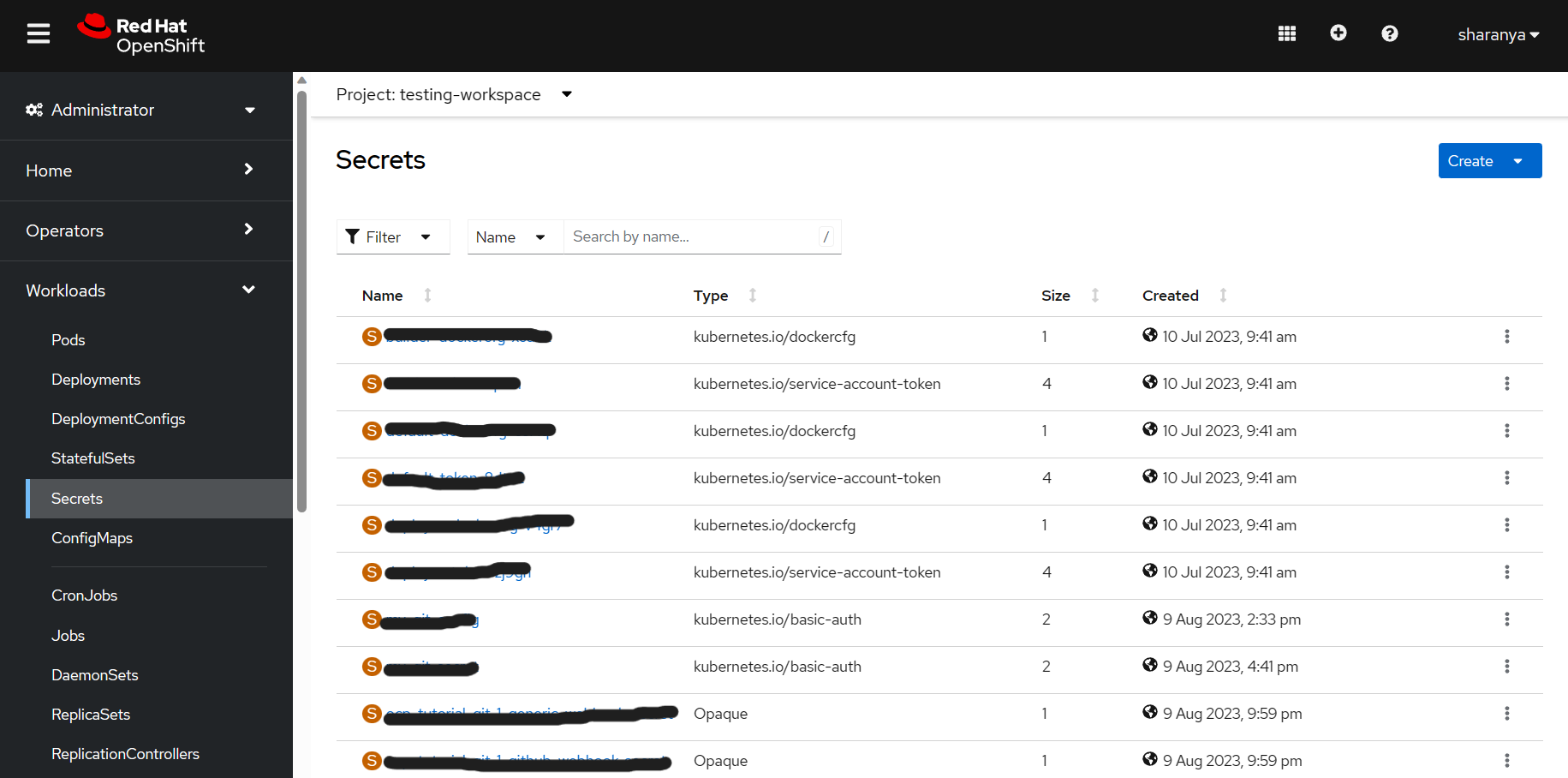
## Creating a Secret using web console:

**Step1:** Select the Administrator from the top left drop-down

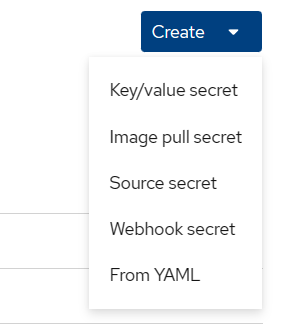


## 

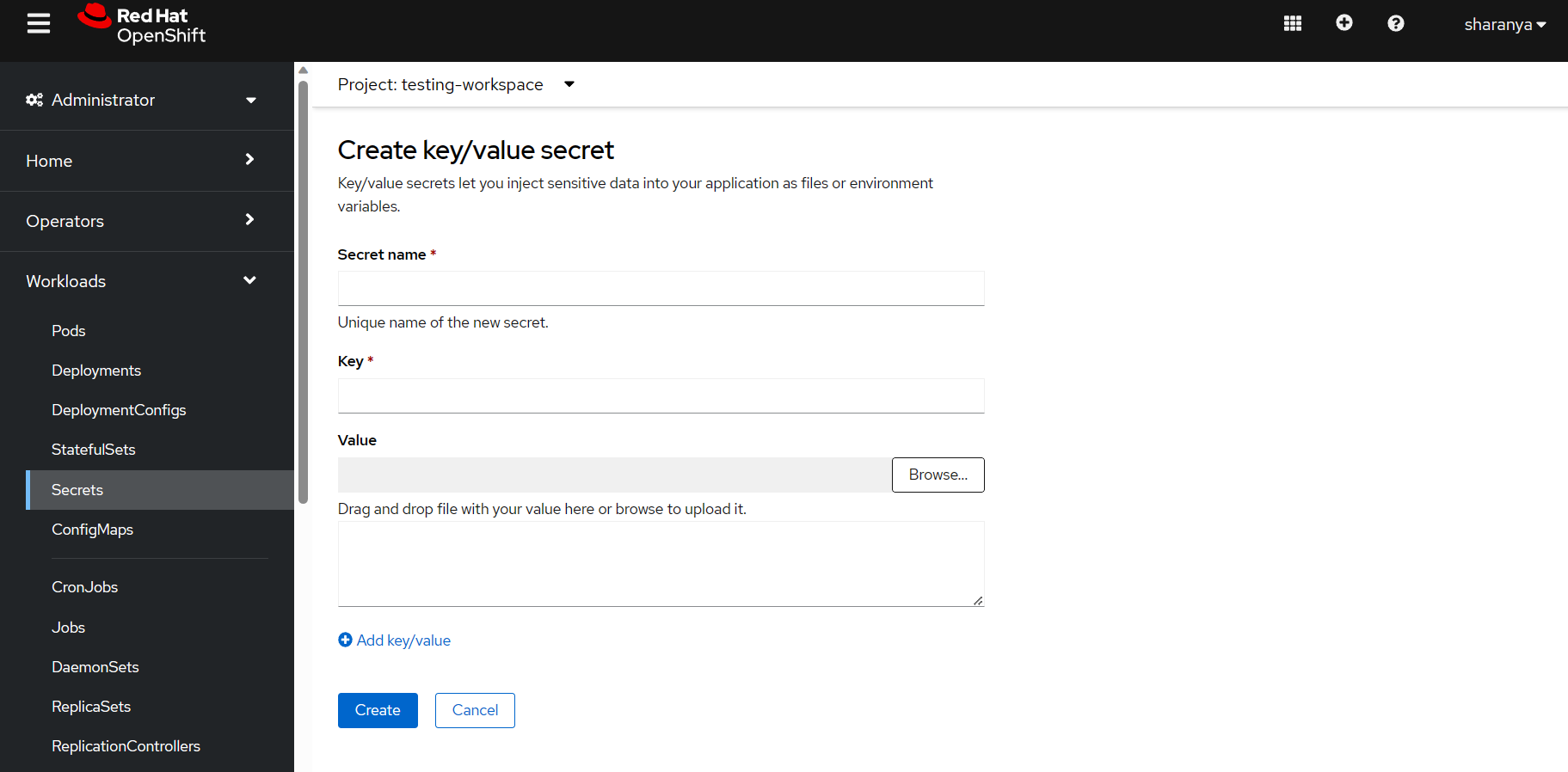
Step2: Then click on the Workloads and then on Secrets



Select the project from the Project drop-down on the top and then create a secret by clicking on the Create button. It will show options as shown below



Step3: Click on the secret you want to create. Here we are going to create a key/value secret



Fill the necessary fields and then click on Create to create a Secret. This is how secrets are created.

## 

## Creating a secret using yaml file

* **Step1:**

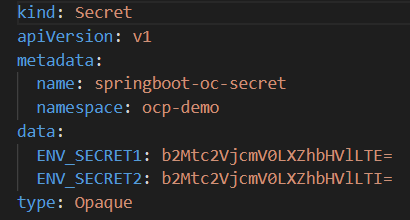
Switch to the desire project using the following command:

|  |
| --- |
| **oc project <desired project name>** |

* **Step2:**

Create a secret yaml file

**Example:**



Command:

|  |
| --- |
| oc apply -f <secret file>.yaml |



Explanation:

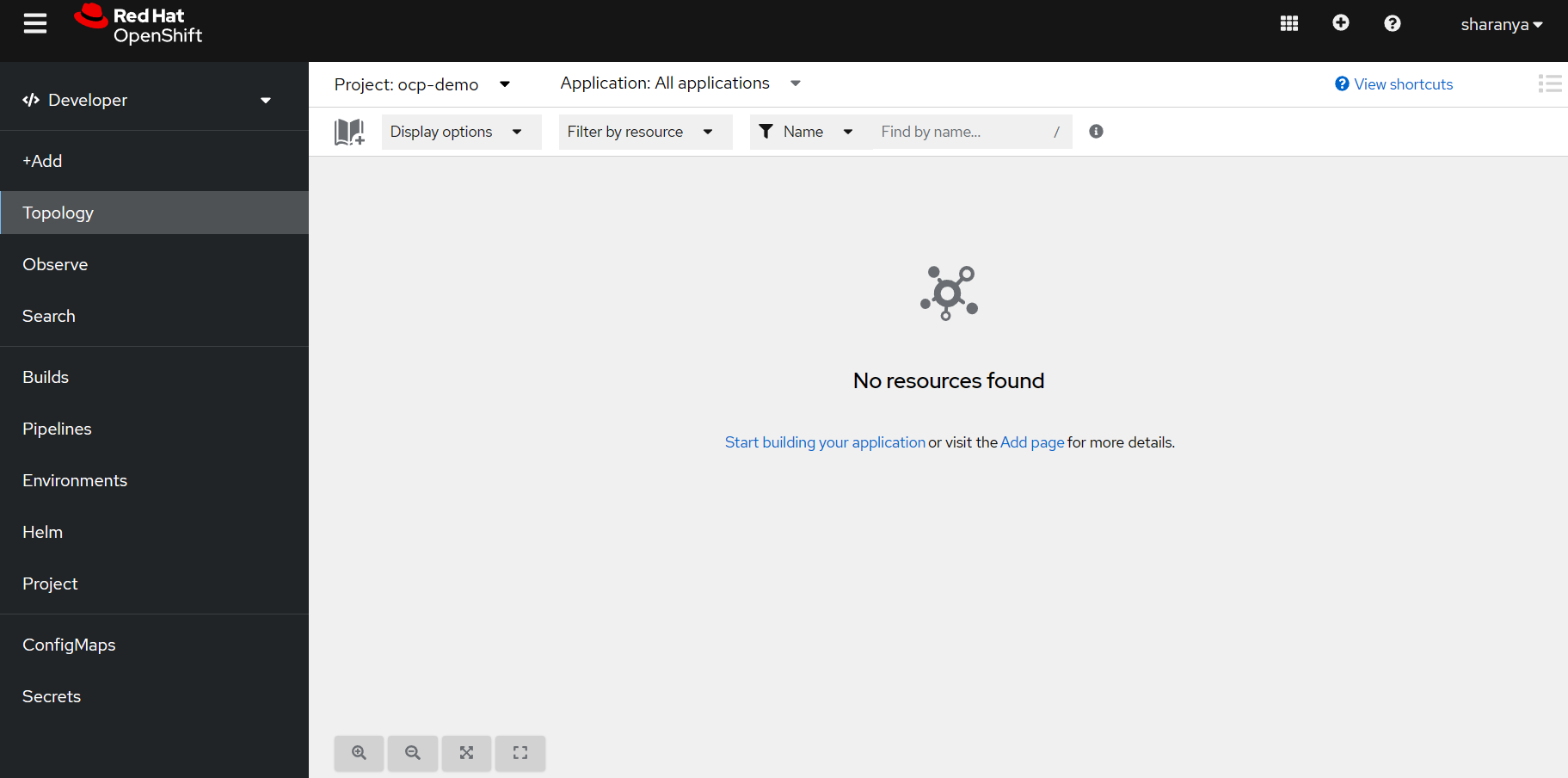
ENV\_SECRET1, ENV\_SECRET2: are environmental variables which have base64 values

# Application - End to end deployment

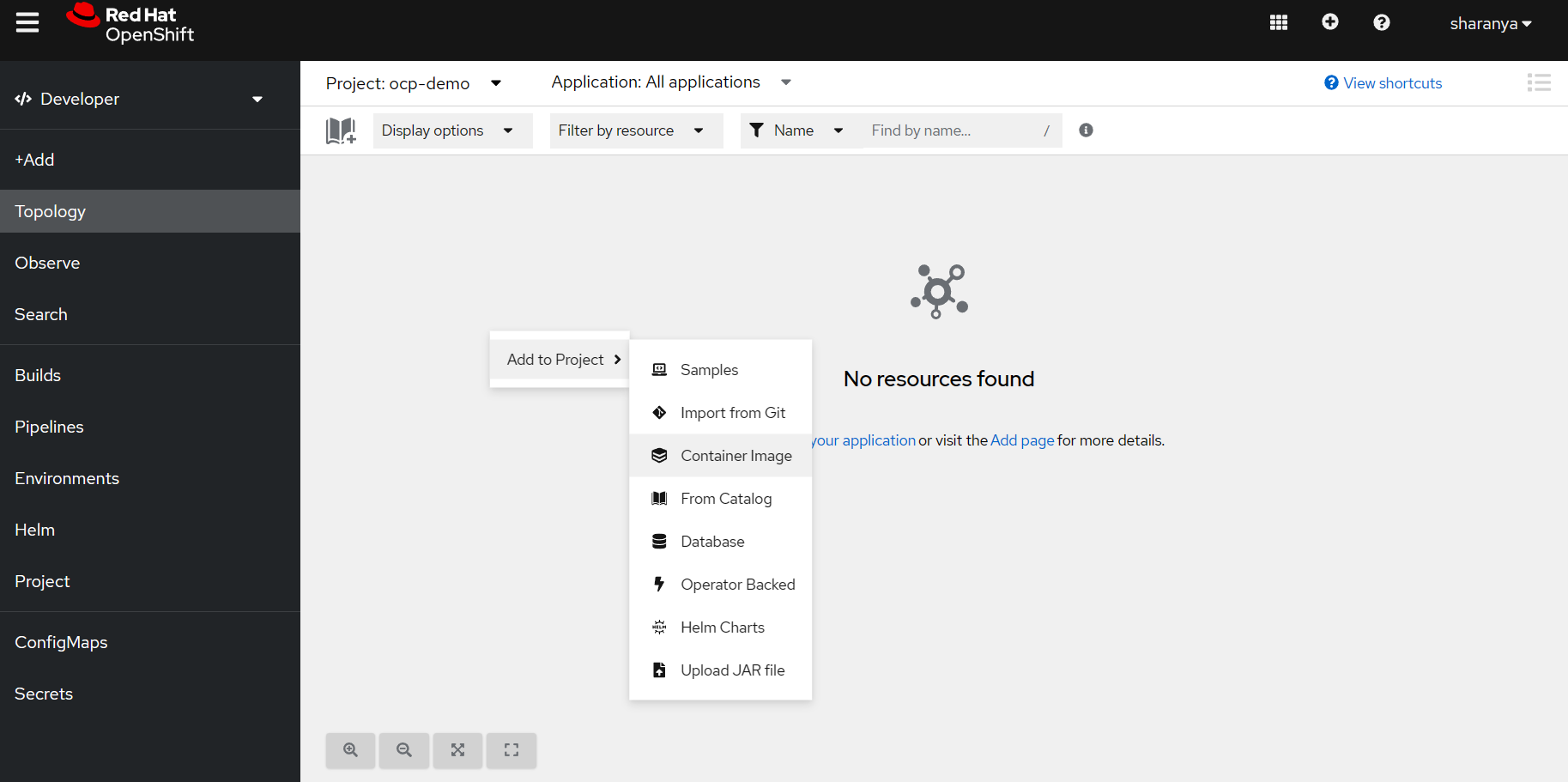
Application refers to the software or service that you are deploying and managing within the platform. An application can consist of one or more components (containers) and often includes additional resources like services, routes, config maps, and more.

## Creating application using web console:

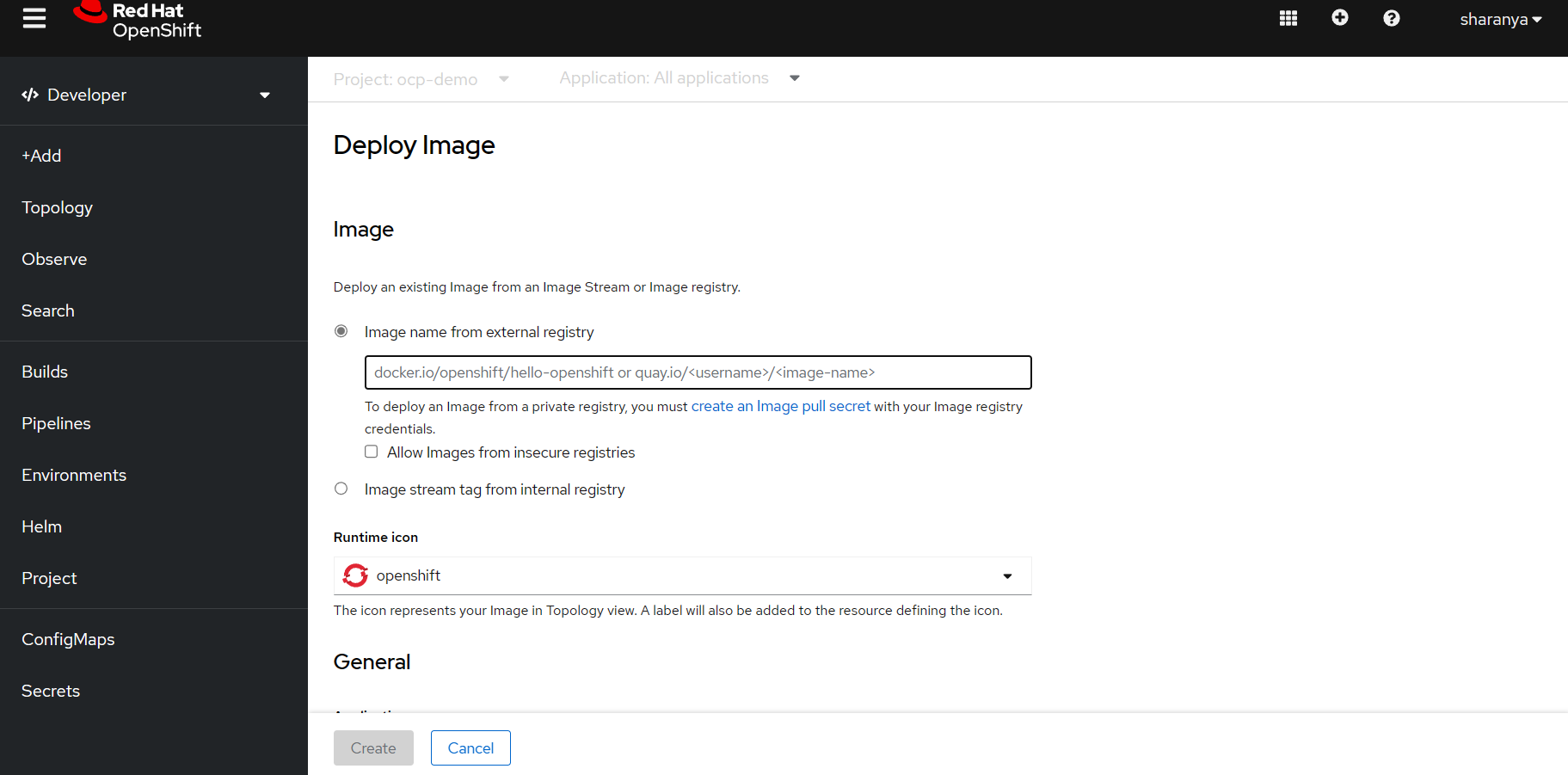
**Step1:** In the topology, select the project you want to create the application by clicking the drop-down of the project option



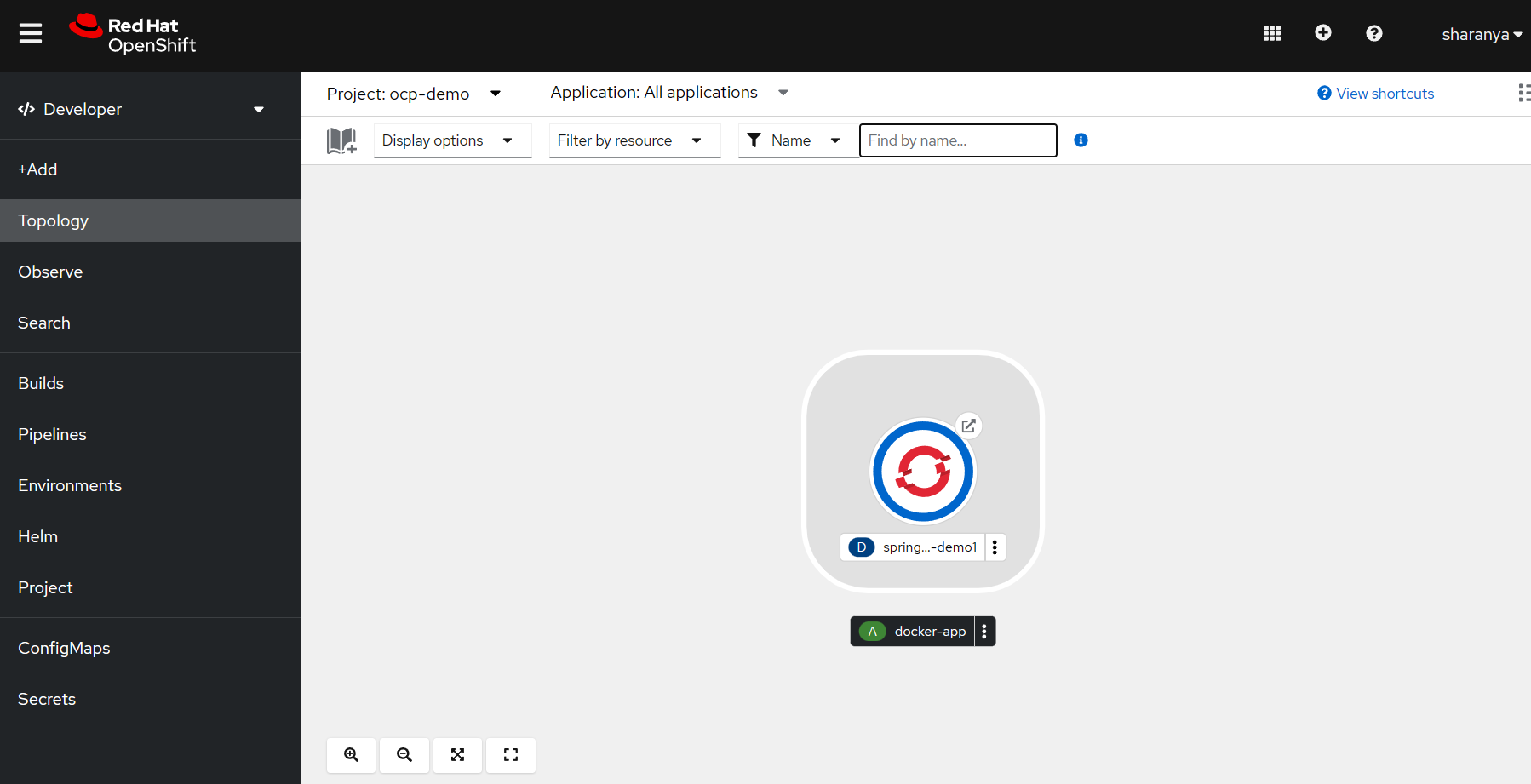
**Step2:** Right click on the empty space, you will see Add to Project option, on that select the source for the application as Container image



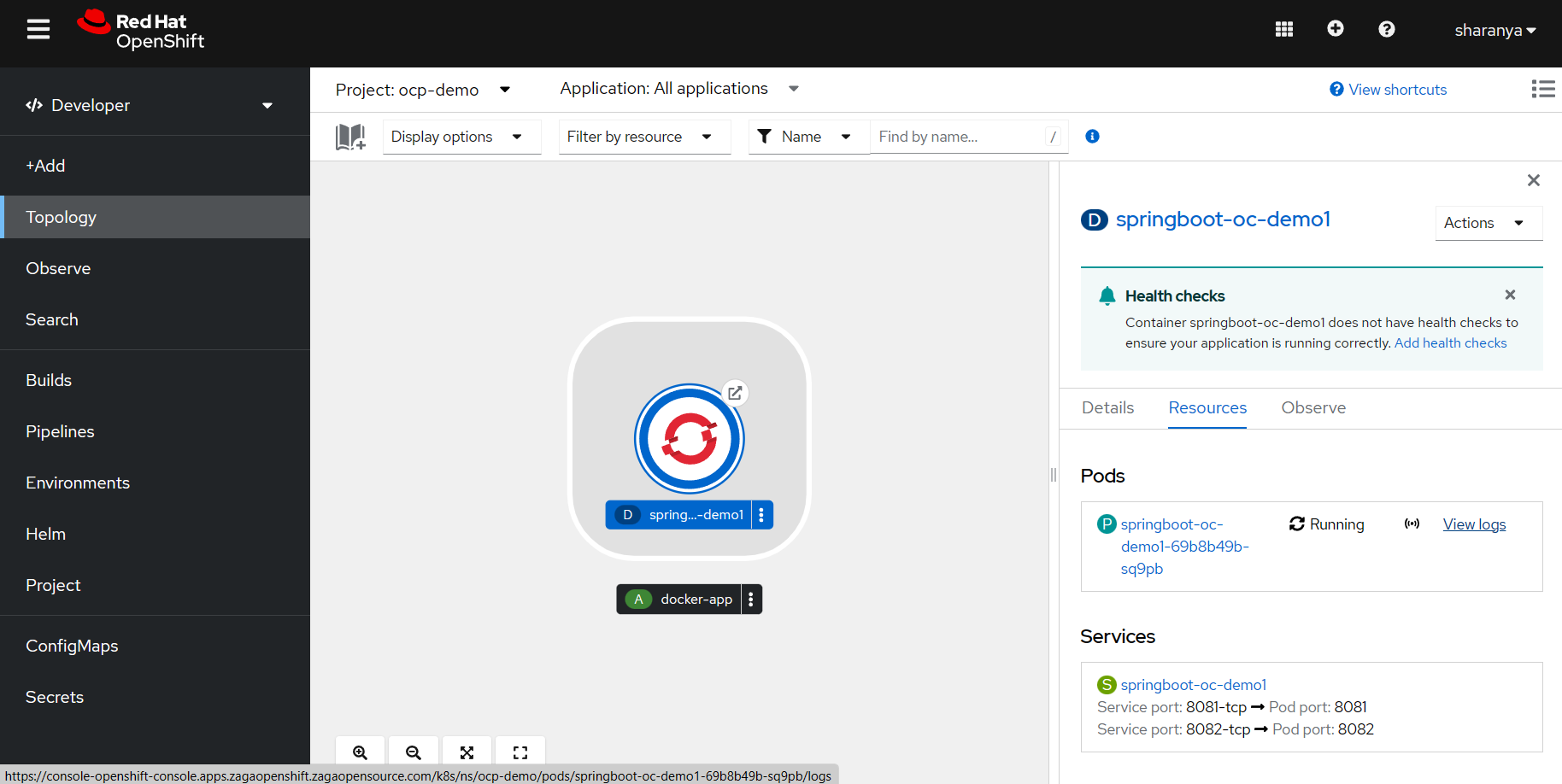
**Step3:** Enter the required fields and then click on the Create



The application will be created



You can view the logs by clicking on the pod



# 

## 

## 

## 

## 

## 

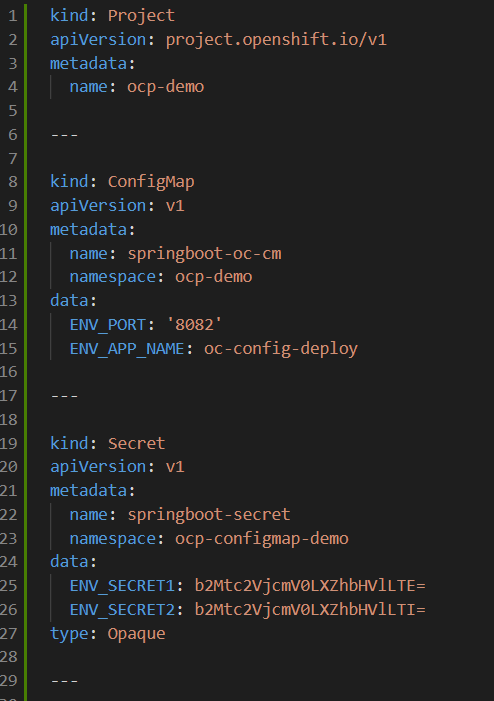
## 

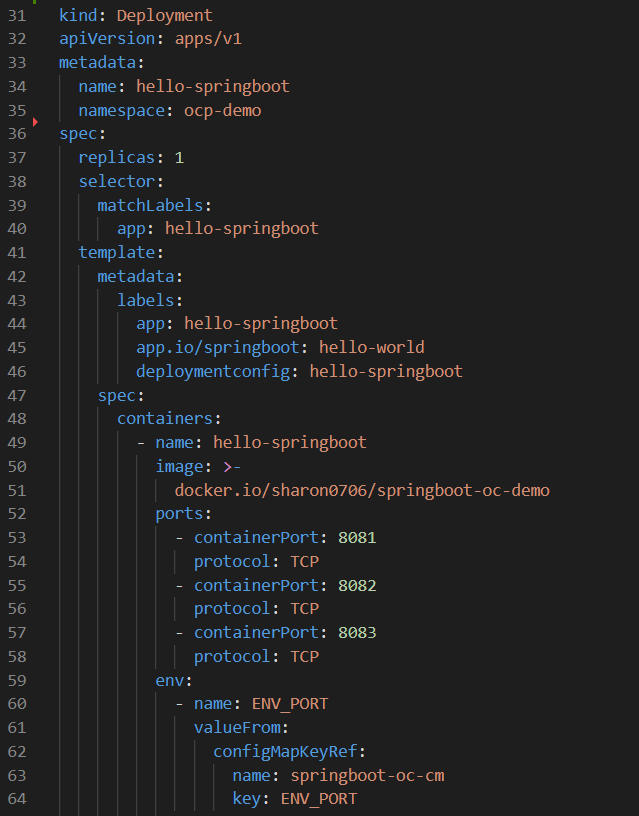
## 

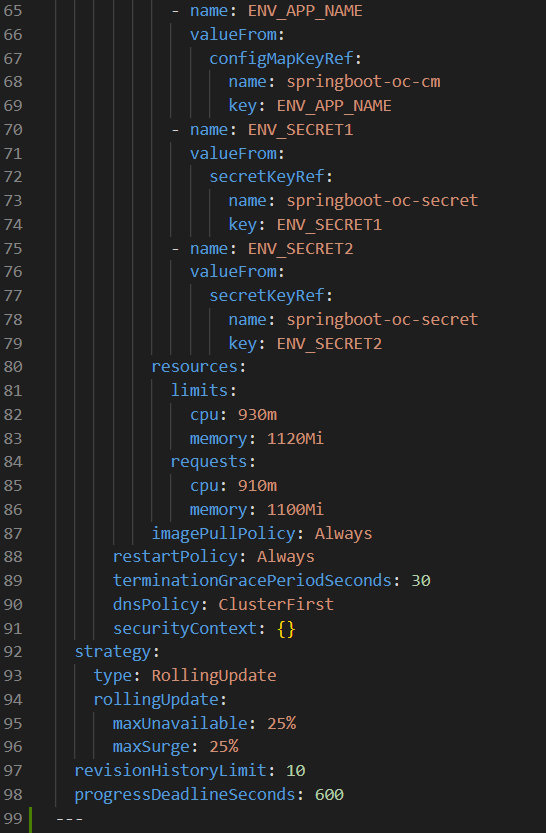
## Creating application using yaml file:

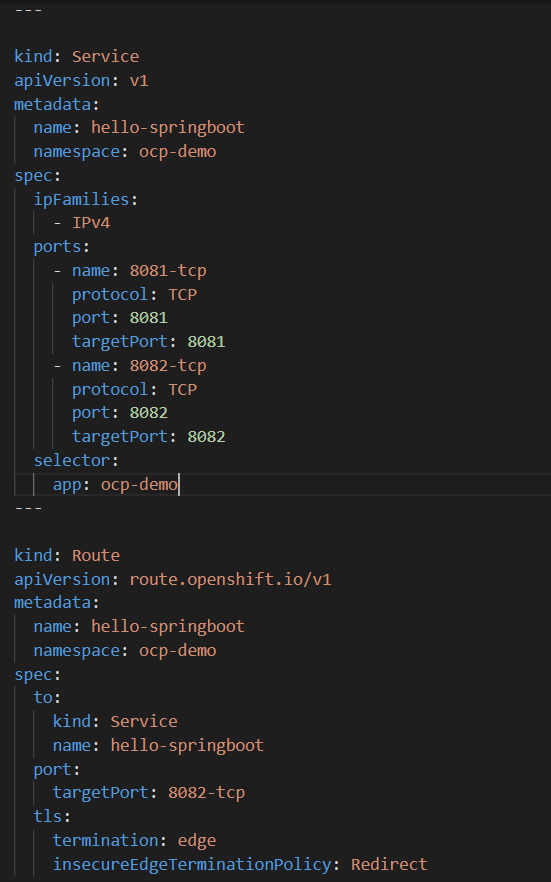
An application can be created using Deployments or DeploymentConfig

Here the application is created using Deployment. This can be done by including all the yaml files and to separate each file with — (three hyphens)



****

****

****

**Command:**

|  |
| --- |
| **oc apply -f <deployment>.yaml** |

The output will be

