

Chapter 4:

Kubernetes OpenShift



Table of Content

- Benefits and features of OpenShift
- Interaction using Web, CLI, Rest
- Additional concepts
 - Project, Application
 - BuildConfig, ImageStream, Source to Image
 - Routes
 - Triggers
- Deploy application using image and source to image
- Application templates
- Deploy application using webhooks

OpenShift Introduction



What is OpenShift

- Kubernetes is an excellent at orchestrating and scheduling containers

But for “APPLICATION” deployment, there’s more needed

- OpenShift provides an excellent **platform** for developing, deploying and running applications.
- A **layer** on top of Docker and Kubernetes that makes it accessible and easy for developers to create applications



Ways to interact with OpenShift

- **Web Console**

Web based GUI for all aspects of developing a project and is full of features

- **Command Line Interface**

Written in “Go”, the tool is called “oc”, is a single binary executable provided for all major operating systems, and it can be used to perform any operation that can be accomplished via the web console

- **Rest APIs**

Web-console and CLI talks via Rest API, which can be used by users mostly to automate through code



Features of OpenShift

All features are same in Web console and the CLI

- Scale the application containers
- Create projects
- View log files
- View the memory and CPU utilization of a container
- Easy build docker images and deploy application with few clicks



Features of OpenShift

Source-to-Image (S2I) - Open source project by the OpenShift team

- No need to write dockerfiles or build Docker images
- Works directly with source code (e.g. github URL)
- Needs base image and CMD when container starts, rest is done by S2I

e.g. Scenario: Java app in Git, using maven build system, depends on Tomcat webserver.

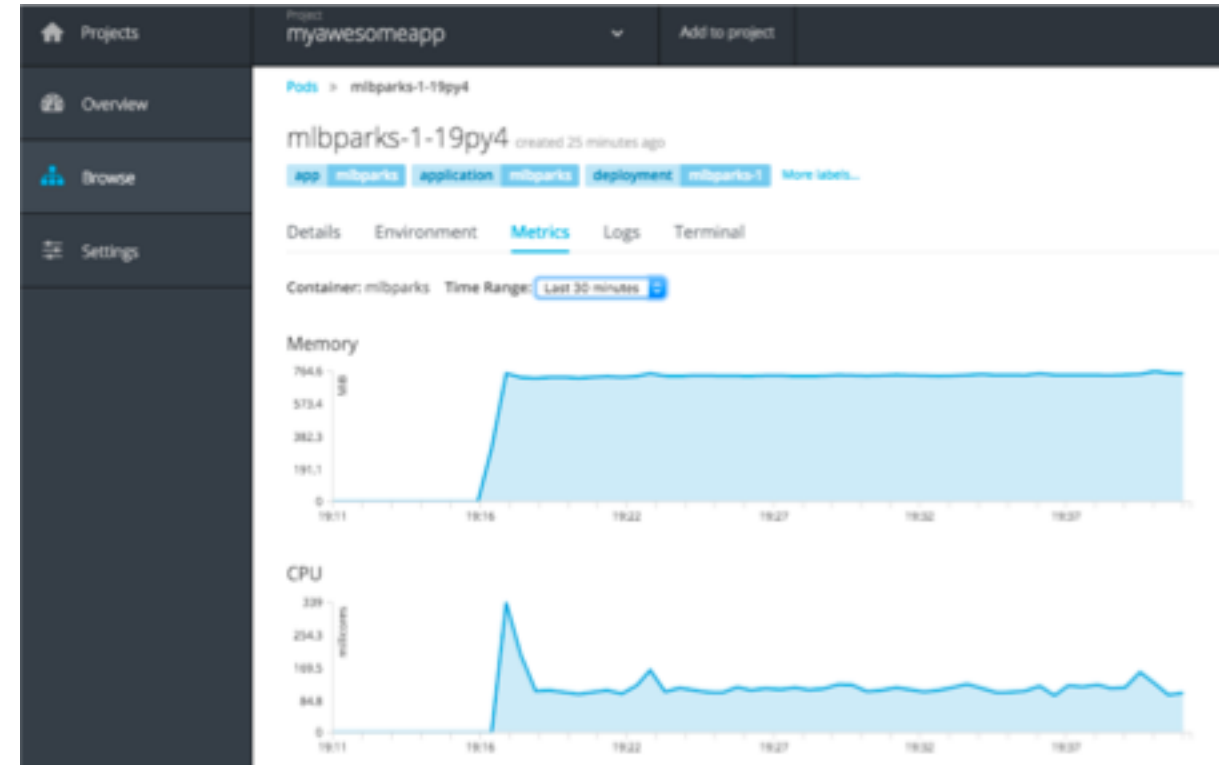
S2I will download the base tomcat image, clone the repo, identify it as a maven project, run a maven build, take the artifact, creates a new docker image containing tomcat webserver and project artifact, deploy the build artifact and start the container



Features of OpenShift

Integrate logs and metrics

- **Logs:-** OpenShift provides a comprehensive view of application logs – runtime logs, build logs, deployment logs – accessible using web-console and CLI
- **Metrics:-** Application metrics, utilization of system resources, memory and CPU utilization for all containers

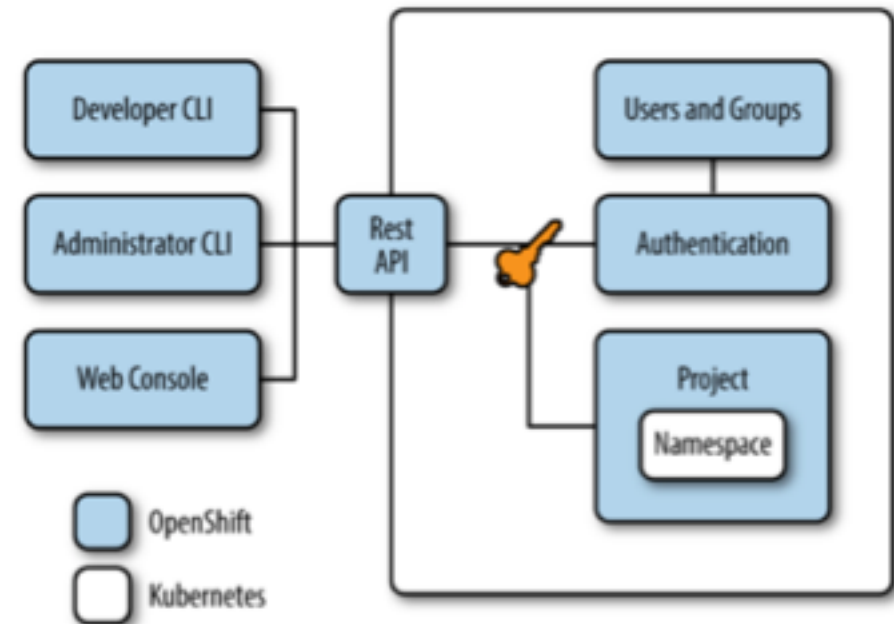


OpenShift “Projects”

Kubernetes use Namespace to divide cluster resources between multiple uses. There is no security between namespaces, Being a user in k8s cluster, you can see all namespaces.

Project

- Project wraps a namespace, with access controlled through an authentication model
- Users can only see and access what they are allowed to
- Enables multitenant



OpenShift “Applications”

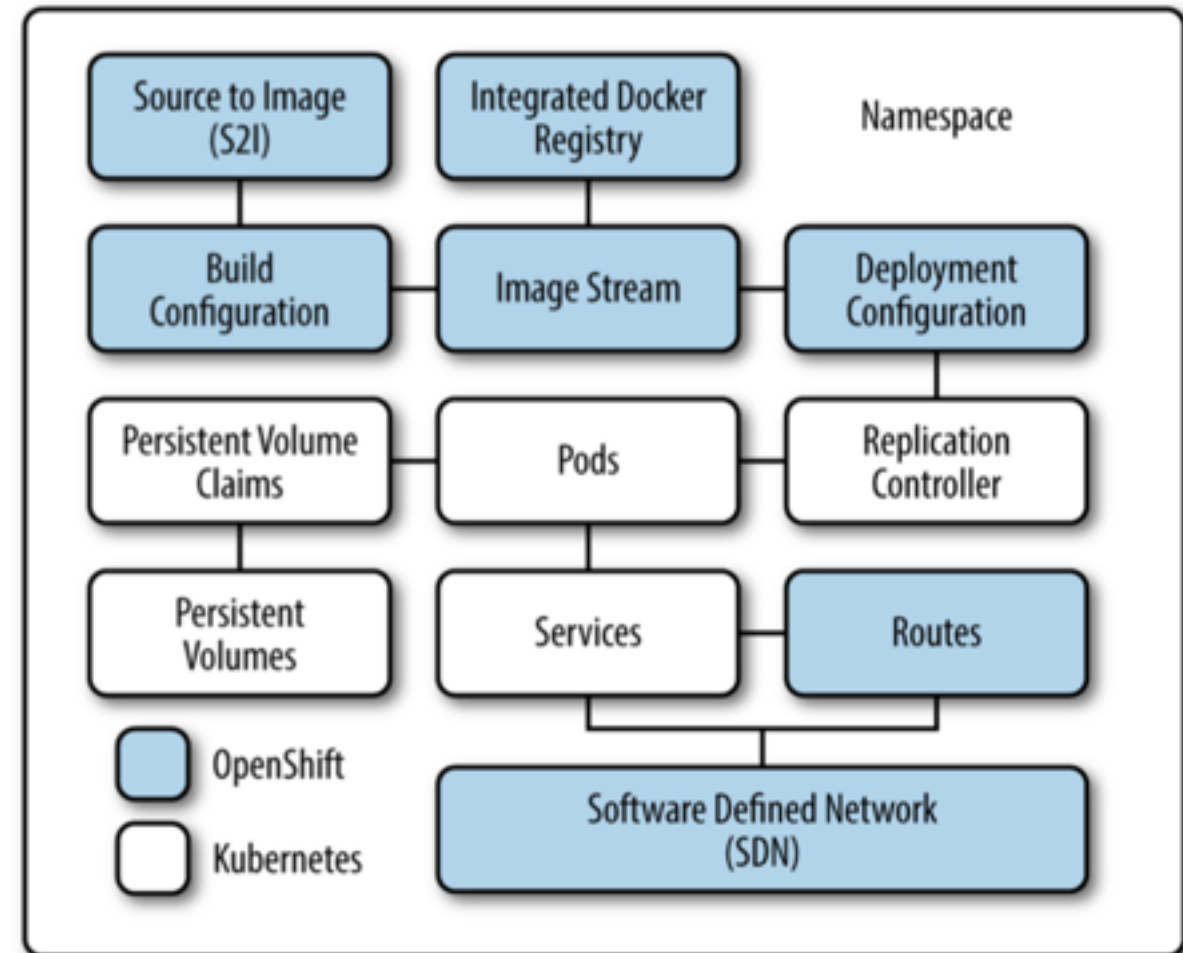
OpenShift has no formal concept of an Application.
It's flexible based on need.

Build configuration: description of how to build source code. It can be either S2I or directly a docker image

Image stream: Tracks Docker image and it's versions. You can refer to an existing Docker registry, if present

Deployment configuration: defines the template for a pod and manages deploying new images or configuration changes. All kinds of deployment strategy goes here and this creates replicationcontroller.

Fig. different parts goes in for an application



OpenShift Deployment and “DeploymentConfig”

Deployments provide management over common user applications, described using :

- **Deployment configuration**, describes the desired state of a particular component of the application.
- **Replication controllers**, which contain the state of a deployment configuration.
- **Pods**

What's different than Kubernetes Deployment ?

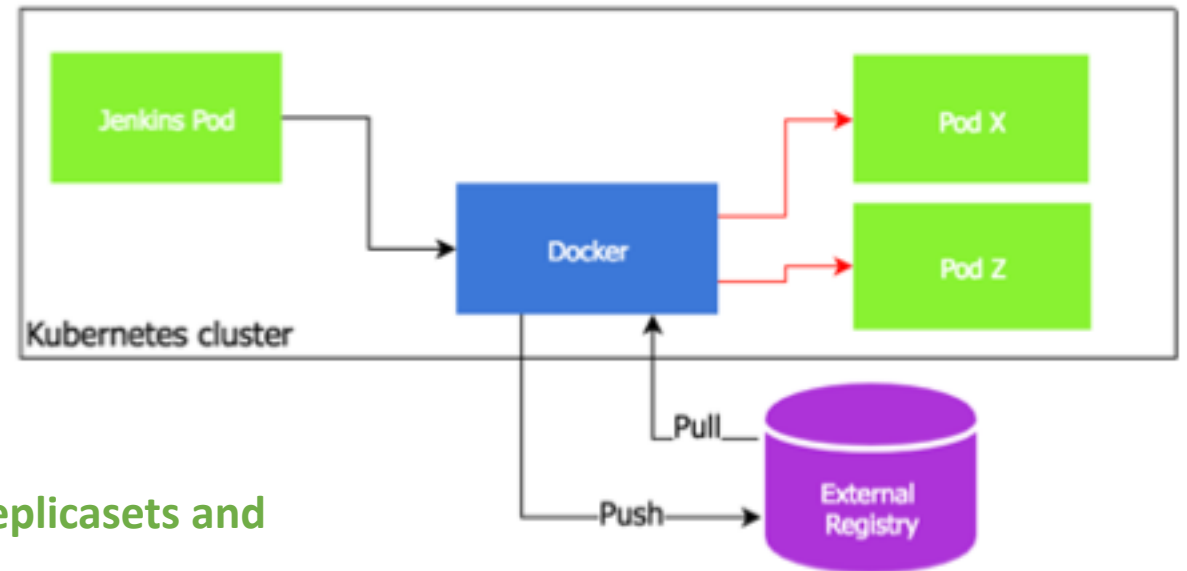
Triggers: drive the creation of new deployments in response to events, both inside and outside OpenShift.

- Config change trigger – changes in replication controller template
- Image change trigger
oc import-image foo --scheduled

OpenShift “ImageStream”

Kubernetes requires to have an image registry to store images.

Need to expose docker socket directly to build agent (e.g. Jenkins) which is a **security risk** - this agent will be able to do anything with other containers running on the same host

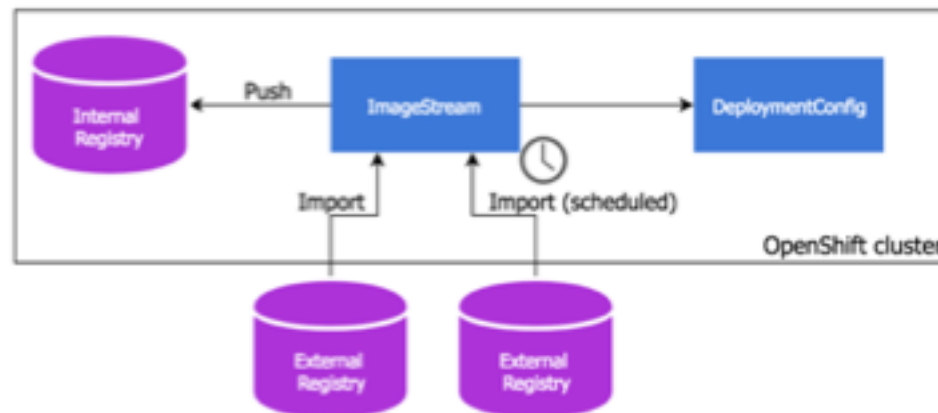


Can we create images the same way as we create pods, replicaset and other resources?

K8s doesn't have {image} resource unlike OpenShift.

Manage Images with BuildConfig and “ImageStream”

- fetch files from git repository
- build a container image based on *Dockerfile* definition from that repo
- push *foo:latest* built image to *ImageStreamTag* object
- Image pushed to OpenShift *internal registry*
- ImageStreamTags objects are references to individual container images kept in internal or external registries



```
apiVersion: v1
kind: BuildConfig
metadata:
  name: foo-build
spec:
  source:
    git:
      uri: "git@git.example.com:foo.git"
      type: Git
  strategy:
    dockerStrategy:
      noCache: true
  output:
    to:
      kind: ImageStreamTag
      name: foo:latest
```

OpenShift “Build” and “BuildConfig”

Build - Transform input parameters into a resulting object, mostly into a runnable container image.

BuildConfig - Config describing the strategy of a build. The build strategies are:

- Source-to-Image (S2I)
 - Force Pull
 - Incremental
- Pipeline - Jenkins pipeline for execution
 - Provide jenkinsfile in the build config
 - Provide jenkinsfile in a git repo and use git repo in the build config
- Docker – Invokes Docker build, thus expects dockerfile with all required artifact
 - Provide Dockerfile
 - Provide docker image location

OpenShift Build Triggers

In BuildConfig, triggers can be defined to control the circumstances in which the BuildConfig should be run. The following build triggers are available:

- **Webhook** - Trigger a new build by sending a request using push events from Github, GitLab, Bitbucket. In case of any change in Git repo, a new build will be created
- **Image Change** – when a new image is found
- **Configuration Change** – only when new BuildConfig is created. in future versions; updates in BuildConfig will also trigger a new build

Trigger can be set/removed

```
oc set triggers bc <name> --from-github
```

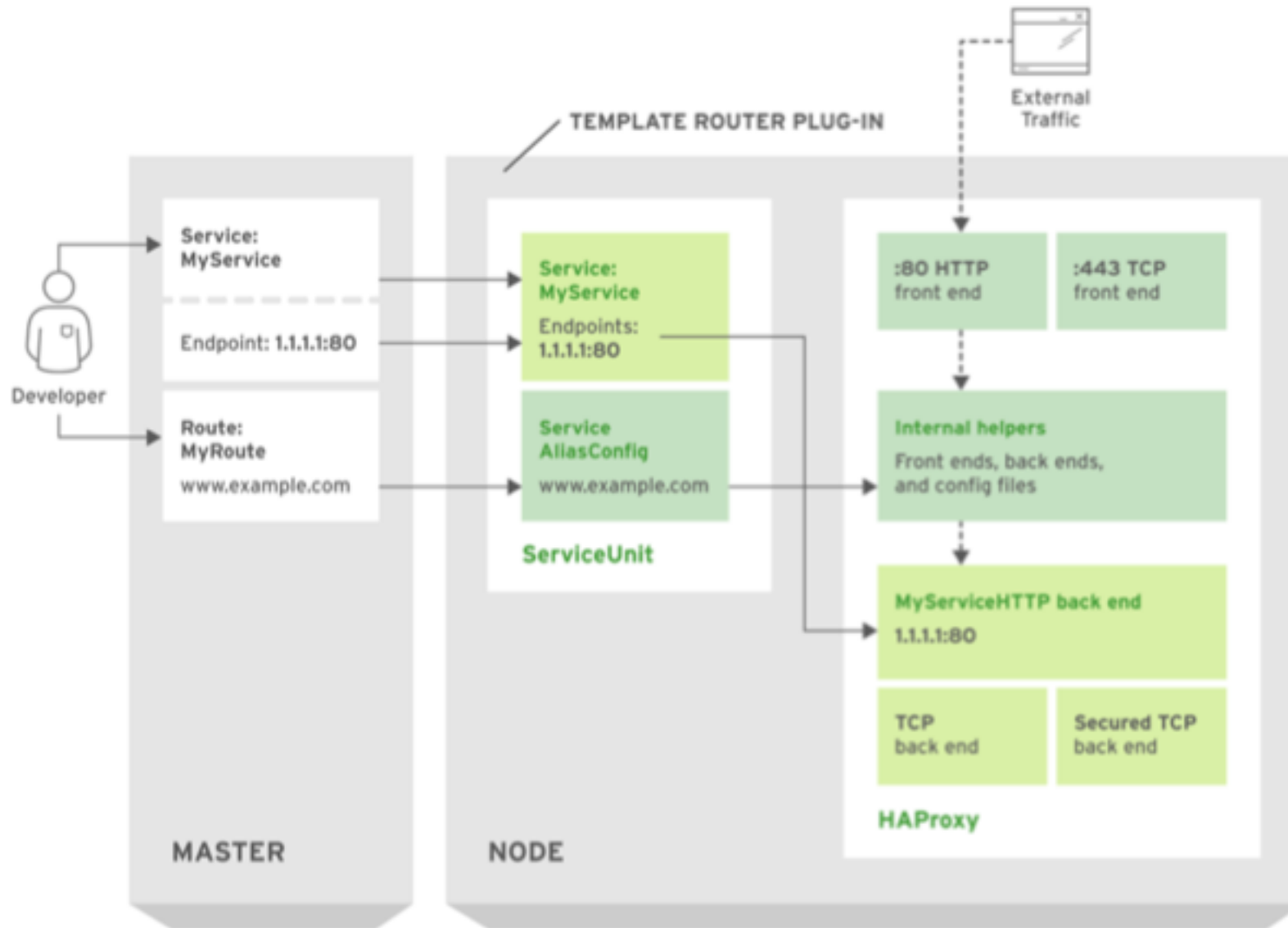
```
oc set triggers bc <name> --from-image='<image>'
```

OpenShift concept of “Routes”

- Expose a service by giving it an externally-reachable hostname - `www.example.com`
- Route and the endpoints is consumed by a **router** (HAProxy, F5 are supported by default) to provide named connectivity that allows external clients to reach applications
- Each route consists of a route name (max 63 characters), service selector, and (optional) security configuration.

```
oc expose svc/frontend --hostname=www.example.com --port 8080
```


HAProxy Router Data Flow

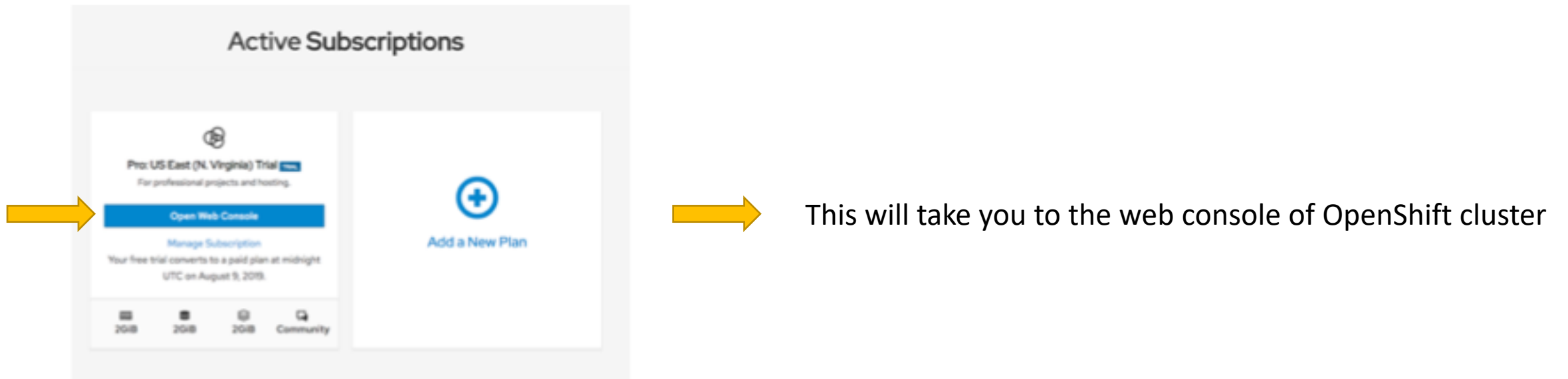


OpenShift Online – labs



OpenShift Online Login

- Login OpenShift Online at <https://manage.openshift.com/>
 - User id: osonline.test1@gmail.com
 - Password: JustM300
- Open Web Console



OpenShift

- Web Console labs



Ex 1: Deploy Application – Web Console (Using Image)

- Steps to execute

Steps:

- Create project **<your-name>-firstproject**
- Deploy from image – with image "**openshiftroadshow/parksmap-katacoda:1.2.0**"
- Scale up pod; Check logs; Check events
- Check Image Stream
- Create route to expose external service

Ex 1: Deploy Application – Web Console (Using Image)

- Steps in UI

My Projects + Create Project

Create Project

* Name
shekhar-firstproject
A unique name for the project.

Display Name
shekhar-firstproject

Description
shekhar firstproject

Cancel Create

Get started with your project.

Add content to your project from the catalog of web frameworks, databases, and other components. You may also deploy an existing image, create or replace resources from their YAML, or JSON definitions, or select an item shared from another project.

Browse Catalog

Deploy Image Import YAML / JSON Select from Project


Deploy image

Image Results

1 2

Image Name
openshiftroadshow/parksmmap-katacoda:1.2.0

To deploy an image from a private repository, you must [create an image pull secret](#) with your image registry credentials. [Learn More](#)

 openshiftroadshow/parksmmap-katacoda:1.2.0
20 days ago, 222.1 MiB, 12 layers

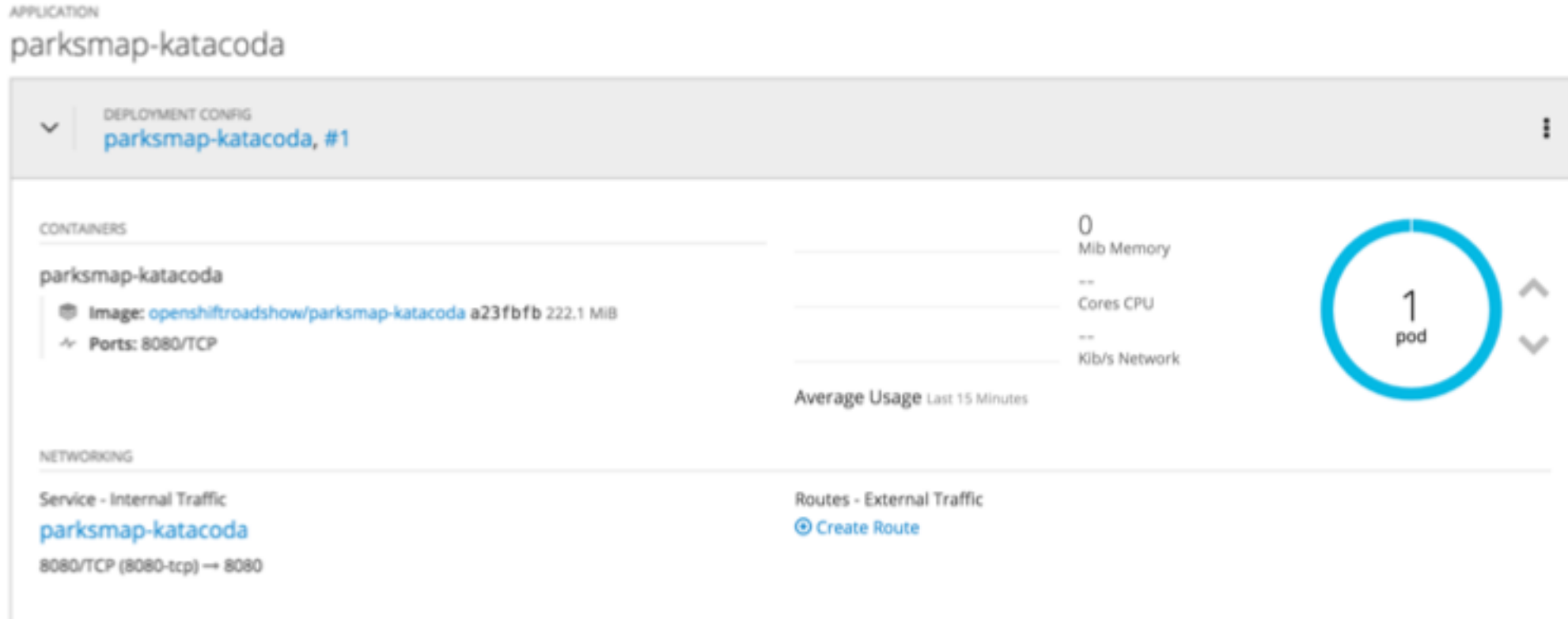
- Image Stream **parksmmap-katacoda:1.2.0** will track this image.
- This image will be deployed in Deployment Config **parksmmap-katacoda**.
- Port 8080/TCP will be load balanced by Service **parksmmap-katacoda**. Other containers can access this service through the hostname **parksmmap-katacoda**.

* Name
parksmmap-katacoda
Identifies the resources created for this image.

Cancel Deploy

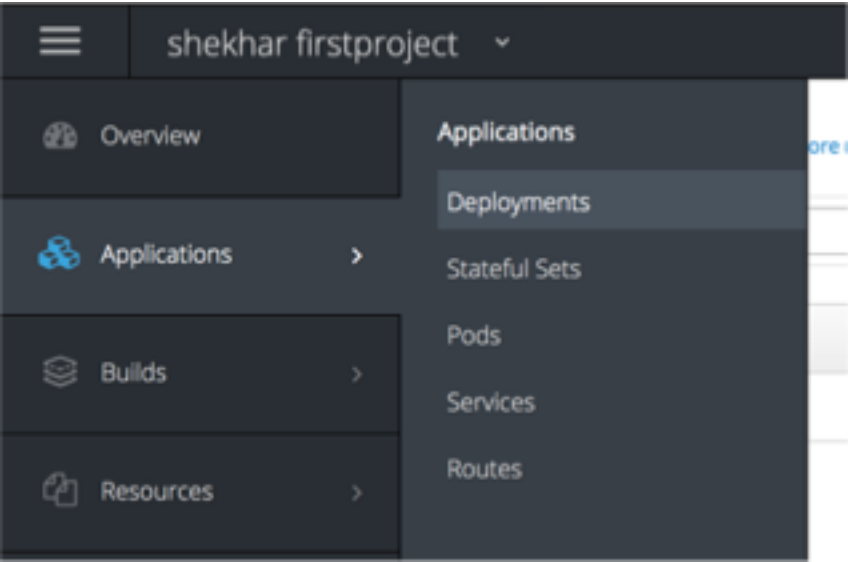
Ex 1: Deploy Application – Web Console (Using Image)

- Project overview



Ex 1: Deploy Application – Web Console (Using Image)

- Deployment



Deployments [Learn More](#)

Filter by label

Name	Last Version
parksmap-katacoda	#1

Deployments > parksmap-katacoda > #1

parksmap-katacoda-1 created 4 minutes ago

app parksmap-katacoda openshift.io/deployment-config.name parksmap-katacoda

Details Environment Metrics **Logs** Events

Status: Active
Deployment Config: [parksmap-katacoda](#)
Status Reason: config change
Selectors: app=parksmap-katacoda, deployment=parksmap-katacoda-1, deploymentconfig=parksmap-katacoda
Replicas: 1 current / 1 desired

1 pod

Scale up/down pods

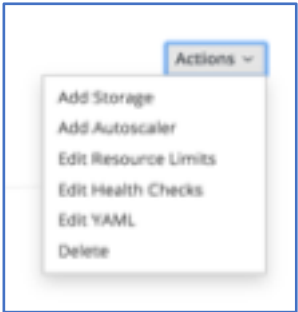
Events

Logs

Template

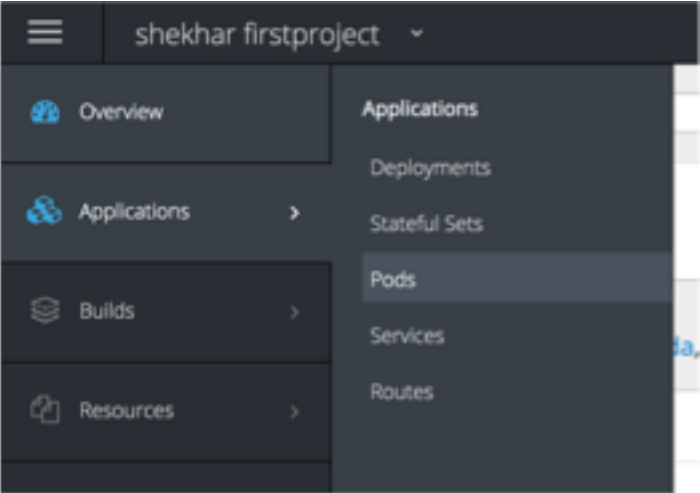
Container parksmap-katacoda does not have health checks to ensure your application is running correctly. [Add Health Checks](#)

Containers



Ex 1: Deploy Application – Web Console (Using Image)

- Pods



Pods [Learn More](#)

Filter by label

Name	Status
parksmap-katacoda-1-dwrbd	Running

[Pods](#) > [parksmap-katacoda-1-dwrbd](#)

parksmap-katacoda-1-dwrbd created 10 minutes ago

[app](#) [parksmap-katacoda](#) [deployment](#) [parksmap-katacoda-1](#) [deploymentconfig](#) [parksmap-katacoda](#)

[Details](#) [Environment](#) [Metrics](#) [Logs](#) [Terminal](#) [Events](#)

Status

Status: Running

Deployment: [parksmap-katacoda, #1](#)

IP: 10.131.89.27

Node: ip-172-31-51-217.ec2.internal (172.31.51.217)

Restart Policy: Always

Template

Containers

parksmap-katacoda

- Image: [openshiftroadshow/parksmap-katacoda](#)
- Ports: 8080/TCP
- Mount: default-token-4kgkw → /var/run/secrets/k
- CPU: 30 millicores to 1 core
- Memory: 409 MiB to 512 MiB

Volumes

default-token-4kgkw

Type: secret (populated by a se

Secret: [default-token-4kgkw](#)

Container parksmap-katacoda

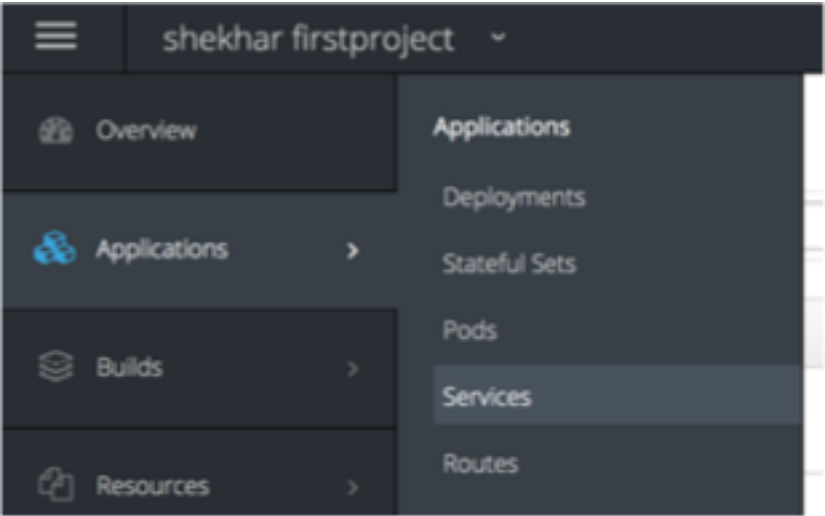
State: Running since Jul 10, 2019 10:40:20 PM

Ready: true

Restart Count: 0

Ex 1: Deploy Application – Web Console (Using Image)

- Service



Services [Learn More of](#)

Filter by label

Name	Cluster IP	External IP
parksmap-katacoda	172.30.200.202	none

Services > parksmap-katacoda

parksmap-katacoda created 12 minutes ago

app **parksmap-katacoda**

Details Events

Selectors: deploymentconfig=parksmap-katacoda

Type: ClusterIP ←

IP: 172.30.200.202

Hostname: parksmap-katacoda.shekhar-firstproject.svc ⓘ

Session affinity: None

Routes: [Create route](#) ←

Traffic

Route	Service Port	Target Port	Hostname
none →	8080/TCP (8080-tcp)	→ 8080	none

Learn more about [routes](#) and [services](#).

Pods

Pod	Status	Containers
		Ready

Note:- Check **routes**, there might be no routes

Ex 1: Deploy Application – Web Console (Using Image)

- Image Stream

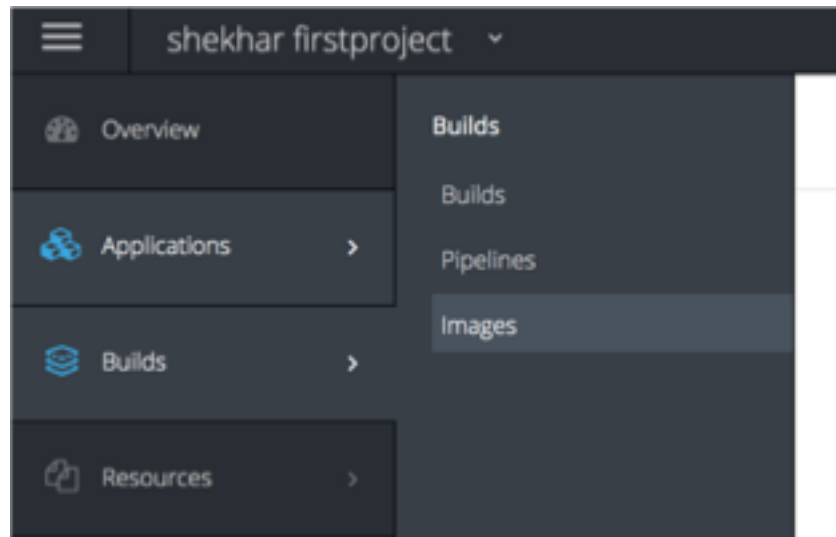


Image Streams [Learn More](#)

Filter by label

Name	Docker Repo
parksmap-katacoda	docker-registry.default.svc:5000/shekhar-firstproject/parksmap-katacoda

[Image Streams](#) > parksmap-katacoda

parksmap-katacoda created 15 minutes ago

app **parksmap-katacoda**

Pulling repository: docker-registry.default.svc:5000/shekhar-firstproject/parksmap-katacoda
Image count: 1

[Show Annotations](#)

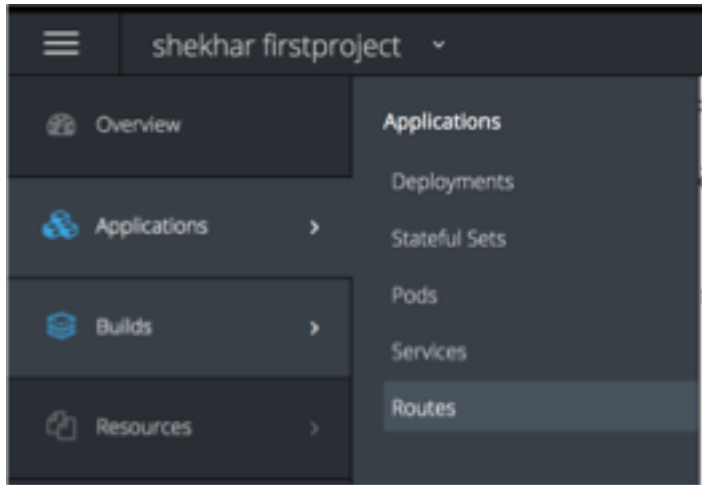
Tag	From
<input type="text" value="1.2.0"/>	openshiftroadshow/parksmap-katacoda:1.2.0

i To push an image to this image stream:

```
$ sudo docker tag myimage registry/shekhar-firstproject/parksmap-katacoda:tag  
$ sudo docker push registry/shekhar-firstproject/parksmap-katacoda:tag
```

Ex 1: Deploy Application – Web Console (Using Image)

- Create Route



The screenshot shows the 'Create Route' form in the OpenShift web console. The form has a title 'Create Route' and a subtitle 'Routing is a way to make your application publicly visible.' It contains several input fields: 'Name' (with the value 'parksmap'), 'Hostname' (with the value 'www.example.com'), 'Path' (with the value '/'), 'Service' (with the value 'parksmap-katacode'), and 'Target Port' (with the value '8080 -> 8080 (TCP)').

The screenshot shows the 'Routes' page in the OpenShift web console. It has a title 'Routes' and a link 'Learn More'. Below the title is a search bar with the placeholder text 'Filter by label'. Below the search bar is a table with two columns: 'Name' and 'Hostname'. The table contains one row with the name 'parksmap' and the hostname 'http://parksmap-shekhar-firstproject.b9ad.pro-us-east-1.openshiftapps.com'.

Name	Hostname
parksmap	http://parksmap-shekhar-firstproject.b9ad.pro-us-east-1.openshiftapps.com

URL to access

Ex 1: Deploy Application – Web Console (Using Image)

- Monitoring and Events

The screenshot displays the OpenShift web console interface for a project named "shekhar firstproject". The left sidebar contains navigation links: Overview, Applications, Builds, Resources, Storage, Monitoring (selected), and Catalog. The main content area is divided into two columns. The left column, titled "Monitoring", includes sections for Pods, Deployments, Stateful Sets, and Builds. The right column, titled "Events", shows a list of recent events.

Monitoring Section:

- Pods:** A table showing one pod named "parksmap-katacoda-1-dwrbd" in a "Running" state, created 19 minutes ago. The image used is "openshiftroadshow/parksmap-katacoda a23fbfb".
- Deployments:** A table showing one deployment named "parksmap-katacoda-1" with one pod, created 19 minutes ago. The image used is "openshiftroadshow/parksmap-katacoda a23fbfb".
- Stateful Sets:** A message stating "There are no stateful sets in this project."
- Builds:** A message stating "There are no builds in this project."

Events Section:

Event Name	Event Type	Time
Pods/parksmap-katacoda-1-deploy	Killing	19 minutes ago
Pods/parksmap-katacoda-1-dwrbd	Pulled	19 minutes ago
Pods/parksmap-katacoda-1-dwrbd	Created	19 minutes ago
Pods/parksmap-katacoda-1-dwrbd	Started	19 minutes ago
Pods/parksmap-katacoda-1-dwrbd	Pulling	19 minutes ago
Pods/parksmap-katacoda-1-dwrbd	Scheduled	19 minutes ago
rc/parksmap-katacoda-1	Successful Create	19 minutes ago
Pods/parksmap-katacoda-1-deploy	Pulling	19 minutes ago
Pods/parksmap-katacoda-1-deploy	Pulled	19 minutes ago

Ex 2: Deploy Application – Web Console (Using S2I)

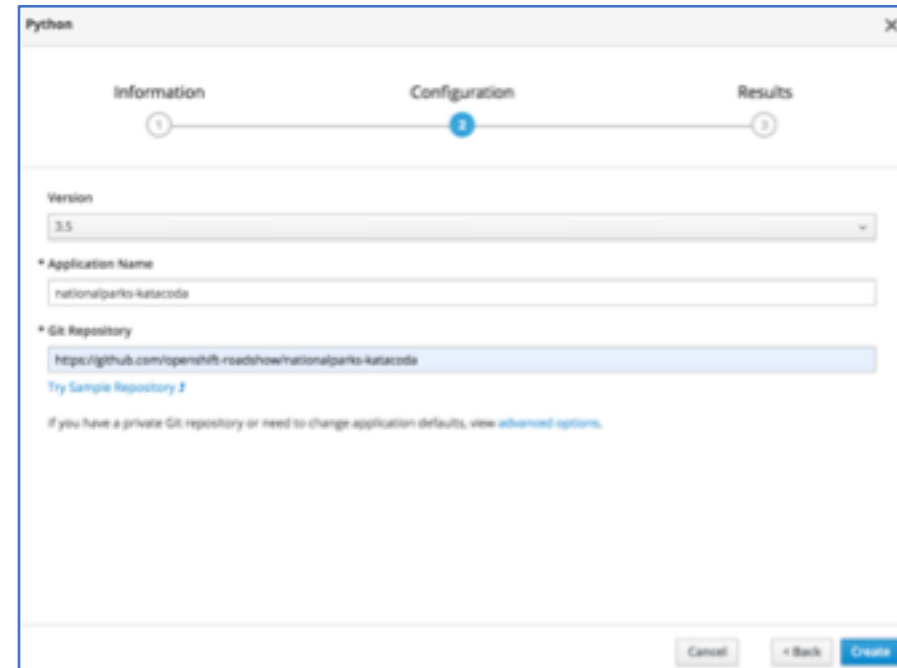
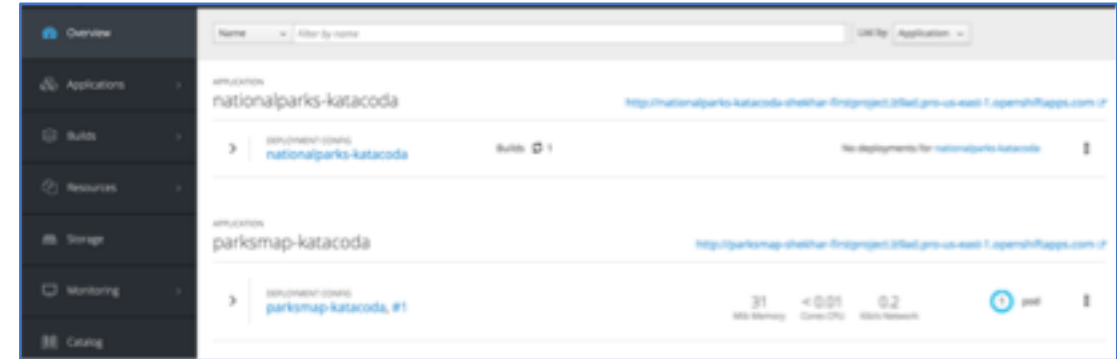
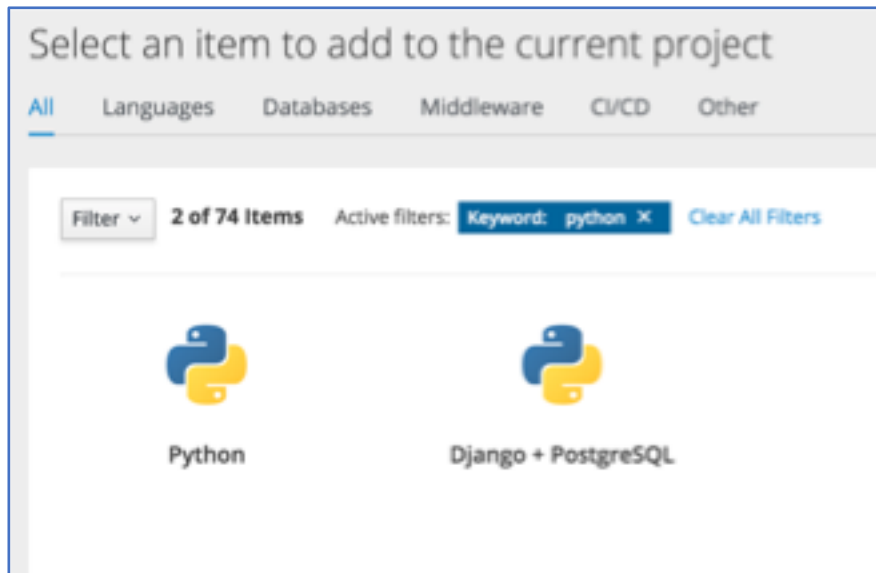
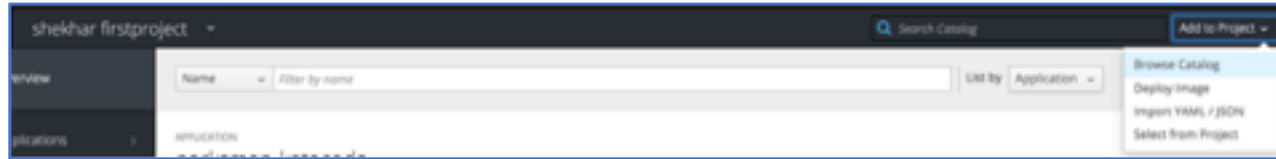
- Steps to execute

Steps:

- Use the same project **<your-name>-firstproject**
- Add --> **browse catalog** --> select Python --> create application
- **Change Python version to 3.5** ; Provide name “**nationalparks-katacoda**”
- Provide Git repo “**<https://github.com/openshift-roadshow/nationalparks-katacoda>**”
- Scale up pod; Check logs; Check events
- Check Build; Check build config
- Check Image Stream
- Create route to expose external service

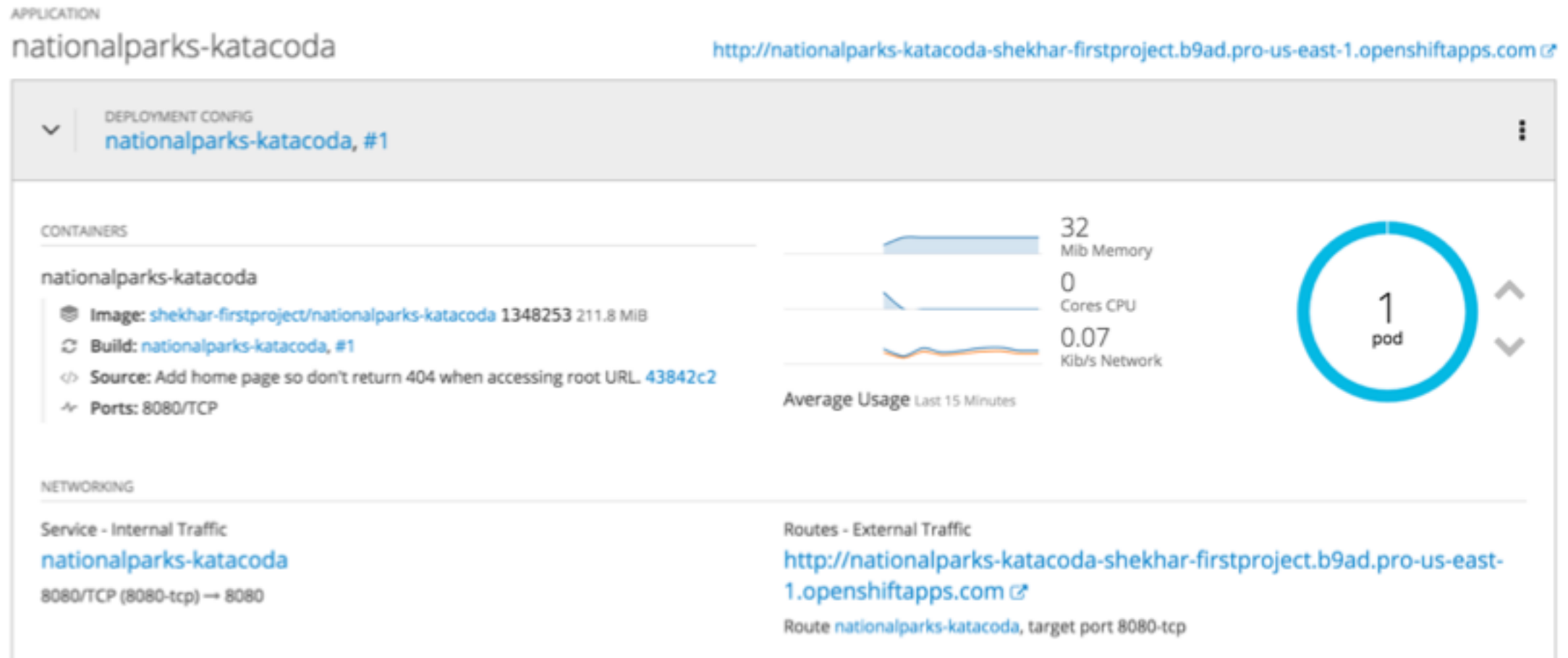
Ex 2: Deploy Application – Web Console (Using S2I)

- Steps in UI



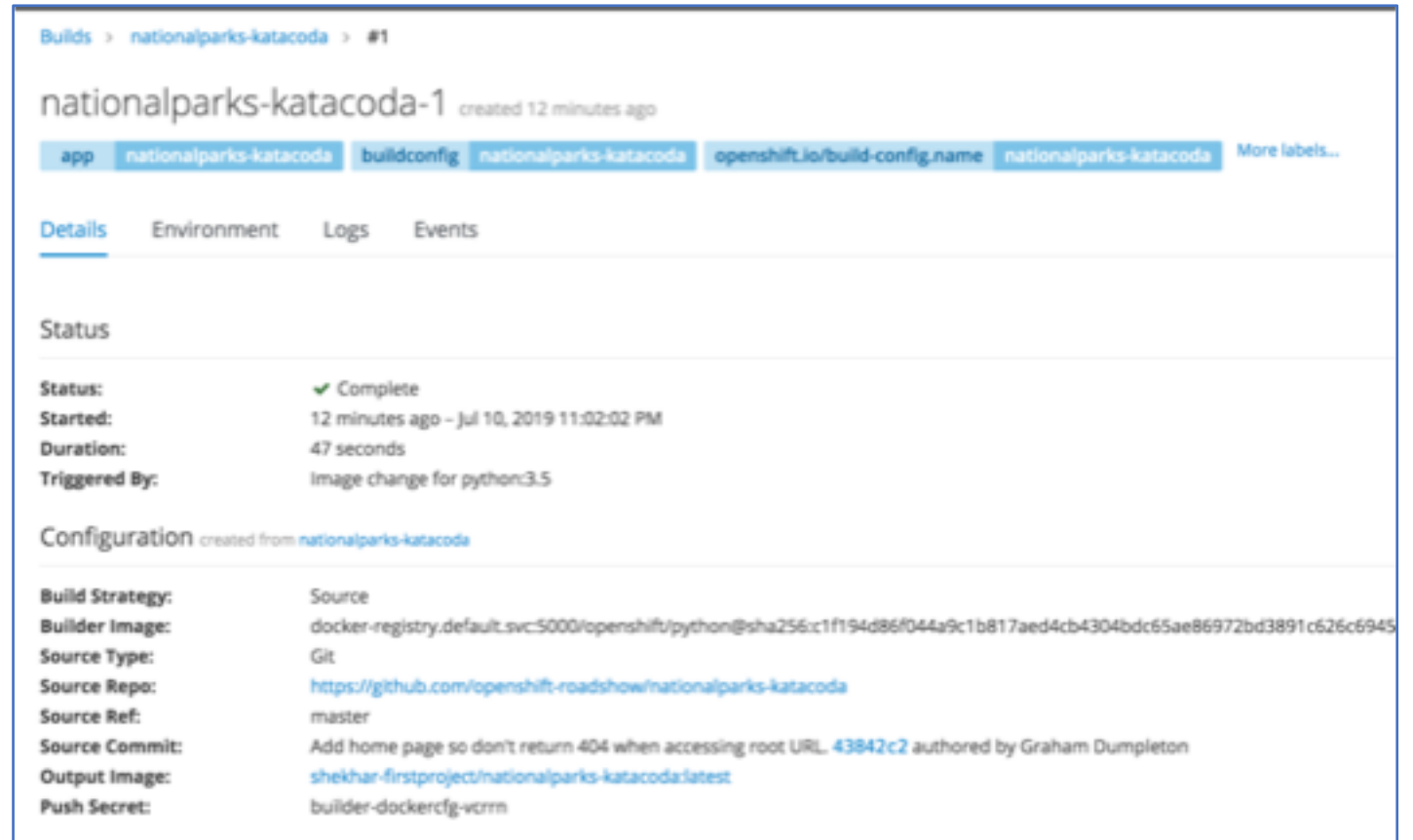
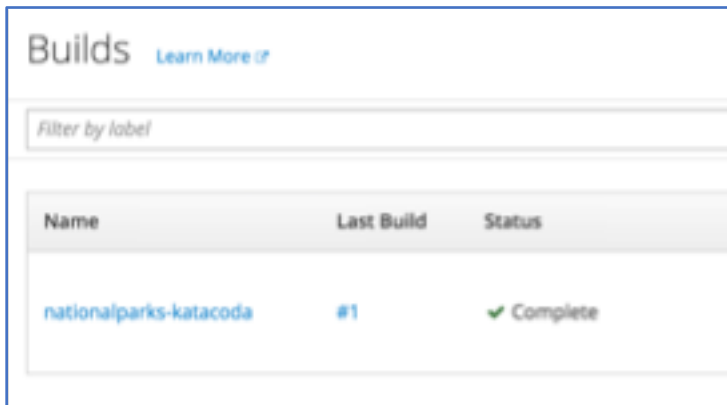
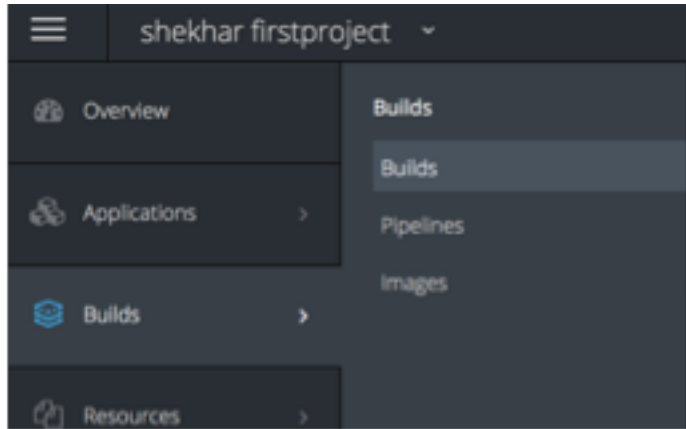
Ex 2: Deploy Application – Web Console (Using S2I)

- Project overview



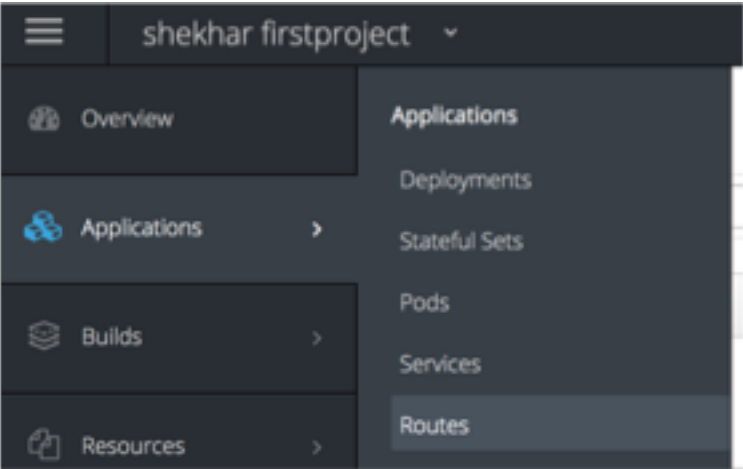
Ex 2: Deploy Application – Web Console (Using S2I)

- Build



Ex 2: Deploy Application – Web Console (Using S2I)

- Route



Route auto created

Routes Learn More			
<input type="text" value="Filter by label"/>			<input type="button" value="Add"/>
Name	Hostname	Service	Target Port
nationalparks-katacoda	http://nationalparks-katacoda-shekhar-firstproject.b9ad.pro-us-east-1.openshiftapps.com	nationalparks-katacoda	8080-tcp
parksmmap	http://parksmmap-shekhar-firstproject.b9ad.pro-us-east-1.openshiftapps.com	parksmmap-katacoda	8080-tcp

Clean up

Delete the projects to save on limited resources

OpenShift – OC CLI labs



OpenShift OC Install

- **OC CLI:** Download the {command-line tool}, un-tar it and add to the {path}

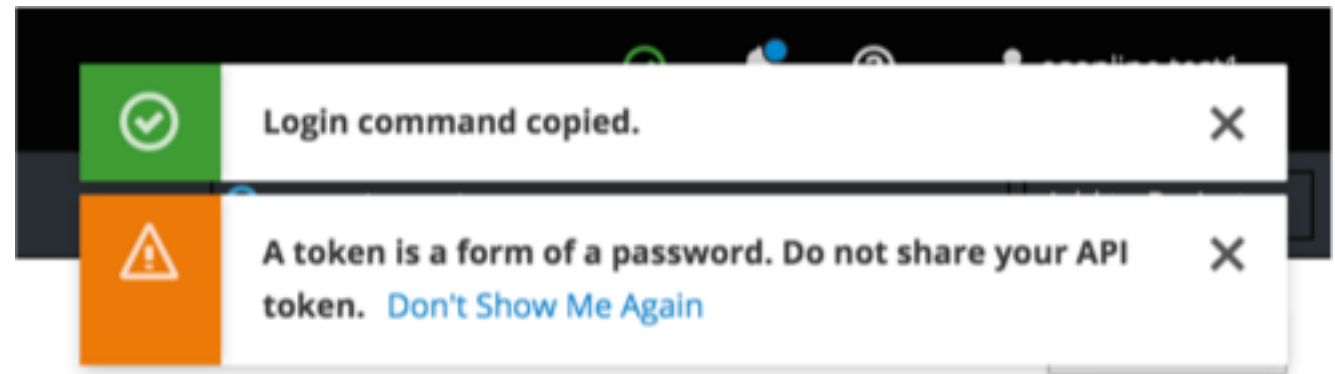
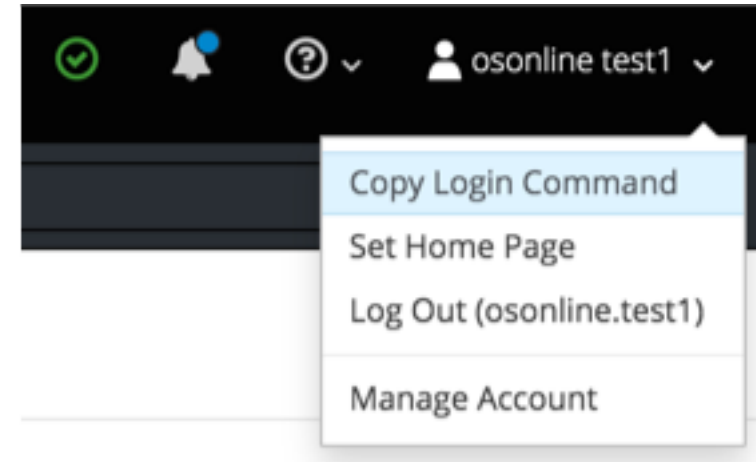
https://github.com/shekhar2010us/kubernetes_teach_git/blob/master/openshift/openshift_oc_cli_download.md

OpenShift OC Login

- **Login token:** In the top right corner, click “Copy Login Command”. This will bring the login token

Format: `oc login <url> --token=<token>`

- **CLI Login:** In a terminal, paste the above command to login to the OpenShift online Cluster



Ex 4: Deploy Application – CLI (Using S2I)

Steps:

- Create a new project **<your-name>-secondproject**
- Create an App using S2I --> using Wildfly 10.0
- Provide name “**openshift-helloworld**”
- Fork the git <https://github.com/shekhar2010us/openshift-helloworld.git>
- Provide Git repo “**<your forked repo>**”
- Scale up pod; Check logs; Check events
- Check Build; Check build config
- Check Image Stream
- Create route to expose external service

https://github.com/shekhar2010us/kubernetes_teach_git/blob/master/openshift/openshift_oc_ex4_s2i.md

Ex 5: Deployment using Webhook

Steps:

- Use the existing project <your-name>-secondproject
- Find the webhook URI for your particular application
- Add webhook in your git project
- Make changes, push to git, these changes will trigger a new build

https://github.com/shekhar2010us/kubernetes_teach_git/blob/master/openshift/openshift_oc_ex5_webhook.md

Clean up

Delete the projects to save on limited resources

Application Template

- Describes a **parameterized set of objects** to produce a list of objects for creation.
- The objects to create can include anything that users have permission to create.
- May also define a **set of labels** to apply to every object defined in the template.

What a templates contains:

- A set of resources, set of values and matching label

Scope of templates:

- Global or local to projects

Creating an Application Template

Important components

- **OpenShift Images:** Base images
 - S2I or Docker images with name and tag
 - ImageStream or private registries
- **Builds:** Generate an image from source code
 - BuildConfig – source; strategy (s2i, docker, custom); output description; list of triggers
- **Images:** Images produced by the builds, store it in ImageStream or private registry
- **Deployments:** This is the core, what images will be deployed and how
 - DeploymentConfig – images, replica, triggers to create deployment automatically, strategy, etc.
- **Abstractions:** Additional resources needed for our application, like networking, storage, security,...

Ex 6: Application Template

- The application template we will be using will create a full Java EE application and a MongoDB database that will perform geospatial queries to update a map via REST endpoints.

https://github.com/shekhar2010us/kubernetes_teach_git/blob/master/openshift/openshift_oc_ex6_templates.md

Create Templates from an existing Project

No filter – export all resources from the current project

```
oc export -o json --as-template=my_template > my_template.json
```

Limit resources

// export all services to a template

```
oc export -o json service --all --as-template=my_template > my_template.json
```

// export the services and deployment configurations labeled name=test

```
oc export -o json svc,dc -l name=test --as-template=my_template > my_template.json
```

Pointers when creating Templates

- Create templates on a **project/team basis**, this reduces lot of ongoing effort
- Provide meaningful names to resources and use **labels** to describe your resources. Be as verbose as possible, you can't add README
- **Parameterize everything** – a user might want to customize
- When the resources in a template are created, if there is a **BuildConfiguration** defined, it will only start an **automated build** if there is an **ImageChange** trigger defined. This will change in the next release and we will be able to launch a build on resource creation
- You should **constrain the CPU and memory** a container in a pod can use

Clean up

Delete the projects to save on limited resources

References



OpenShift CLI Reference:

https://docs.openshift.com/enterprise/3.0/cli_reference/basic_cli_operations.html#object-types

OpenShift Rest Reference:

https://docs.openshift.com/container-platform/3.6/rest_api/openshift_v1.html

OpenShift Rest JAVA Reference:

<https://github.com/openshift/openshift-restclient-java>

OpenShift Rest Python Reference:

<https://github.com/openshift/openshift-restclient-python>

https://docs.openshift.com/container-platform/3.7/rest_api/index.html#rest-api-example-python

OpenShift Online Trial

<https://www.openshift.com/trial/>

OpenShift Online Trial – Web Console

<https://manage.openshift.com/account/index>