OPENSHIFT

**OpenShift** is a cloud development Platform as a Service (PaaS) developed by Red Hat which is used to build, test, deploy, and run their applications.

The architecture of OpenShift is designed in such a way that it can support and manage Docker containers, which are hosted on top of all the layers using Kubernetes.

Openshift has inbuilt registry(Openshift Registry), Continuous integration(CI/CD), pipelines, Source Code Management(SCM).

**Types of Openshift Platforms:**

Openshift platform is available in 4 types for developers.

1. Openshift Origin: Opensource application container platform
2. Openshift Online: Public Application Development hosting service
3. Openshift Dedicated: Managed private cluster on AWS/Google Clouds
4. Openshift Enterprise: On-premise private PaaS

**Openshift SETUP :**

* **Modes of Deployment:**

1. All-in-one:

* Master node & Worker node will be deployed in same system.
* Used in Development environment.

1. Single Master:

* In this, One single master will manage all the worker nodes.

1. Multiple Master:

* Multiple master nodes will manage multiple worker nodes.
* Used in Production environment.
* **Target Environments for Deployment:**
* On-Premise (Virtual Machine)
* On cloud

**Management Tools:**

* Openshift Webconsole – direct way to interact with openshift to manage our applications. By default, it runs on 8443 port.

https://<openshift-ip>:8443/console

* Openshift CLI – Using commands we interact with Openshift
* Openshift Rest API – to interact with third-part applications.

**Openshift Vs Kubernetes:**

Although plain Kubernetes and OpenShift provide the same core orchestration features for containers, there are several differences between plain Kubernetes and the Kubernetes implementation that is built into OpenShift.

<https://www.whizlabs.com/blog/openshift-vs-kubernetes/>

**Builds in Openshift:**

* A [**build**](https://docs.openshift.com/container-platform/3.9/architecture/core_concepts/builds_and_image_streams.html#builds) in OpenShift is the process of transforming input parameters into a resulting object. Most often, builds are used to transform source code into a runnable container image.
* A build configuration, or **BuildConfig**, is characterized by a build strategy and one or more sources. The strategy determines the aforementioned process, while the sources provide its input.

**Build strategies:**

* **Source-to-Image (S2I):**  Produces ready-to-run images by injecting application source into a container image and assembling a new image.This is the most popular build strategy used in Openshift.
* Docker:  Invokes the “[docker build](https://docs.docker.com/engine/reference/commandline/build/)” command taking **Dockerfile** as input to produce a runnable image.
* Pipeline: Defines a Jenkins pipeline for execution
* Custom: Define a specific builder image. Can use our own builder image to customize our build process.

**Openshift Online:**

To use Openshift online plaform, take subscription to Redhat community and then take subscription to Openshift online platform.

* Default user will be “developer”. Any password can be used to login.
* By default, admin rights will be provided to only “system:admin”. But this admin cannot access webconsole.
* Without admin rights, “developer” will get several git and security related authorization issues.
* To give admin privileges to “developer”, login as system:admin using oc CLI.

Note: Use compatible version of CLI tool with online openshift platform, to avoid compatible issues. Used oc CLI version: 4.2

<https://mirror.openshift.com/pub/openshift-v4/clients/oc/4.2/>

* Using oc CLI,

$ oc adm policy add-cluster-role-to-user cluster-admin developer.

NOTE: If developer is unable to add as admin, create a new user with any name and add the same admin policies for that user.

* Now, login as “developer” and create new project.
* Then select specific option from catalog which consists of different languages, databases, pipelines and CI/CD tools.
* Then select required docker image and provide any Git url as source (Github/ Gitlab/ GitBucket). This will create deploymentconfig, buildconfig and pods.
* Now create a service for specific deployment and then create a route for accessing service externally.

**Docker Build:**

Deployed Python Application in Openshift Online platform using source code and Dockerfile from github as:

1. Used python:3.6 image
2. Used github url: <https://github.com/mmumshad/simple-webapp-flask>
3. Got DeploymentConfig, BuildConfig and pods created.
4. Created Service and Route.

NOTE: Please refer yaml files created in openshift online platform at

<https://github.com/AmruthaInnamuri/PythonAppConfigFiles>

**DeploymentConfig using image:**

1. Take source code(Local or Git) and create docker image. Used docker image --gcr.io/fleet-resolver-237016/spring-boot-example:v3
2. Create DeploymentConfig using that docker image.



1. Then create service for that deploymentconfig and then create a route for that service to access externally.

NOTE: please refer this specific yaml files at:

<https://github.com/AmruthaInnamuri/DeploymentConfigWithImage>

**DeploymentConfig with imagestream:**

1. Take source code and create docker image.
2. Push that docker image into internal openshift registry.
3. Create a new imagestream.
4. Now tag the docker image with the imagestream.
5. Now use this imagestream in deploymentconfig, then create service and route.

**Deploying java application in openshift online platform:**

Openshift Online platform’s web console will be OKD(origin community distribution).By default, in okd’s catalog there is no option to deploy Java application.

* To deploy java application used openjdk:8-jdk image and strategy as source.

$ oc new-project SampleJava

$ oc new-app openjdk:8-jdk~<java-github-url> --name=myjava –strategy=source

NOTE: As openshift online platform provides limited resources, could not run java application due memory insufficiency issue. Then shifted to Minishift.

**Online platforms used for hands-on practice:**

1. Katacoda playground
2. KodeKloud playground

**Fabric8 -maven-plugin:**

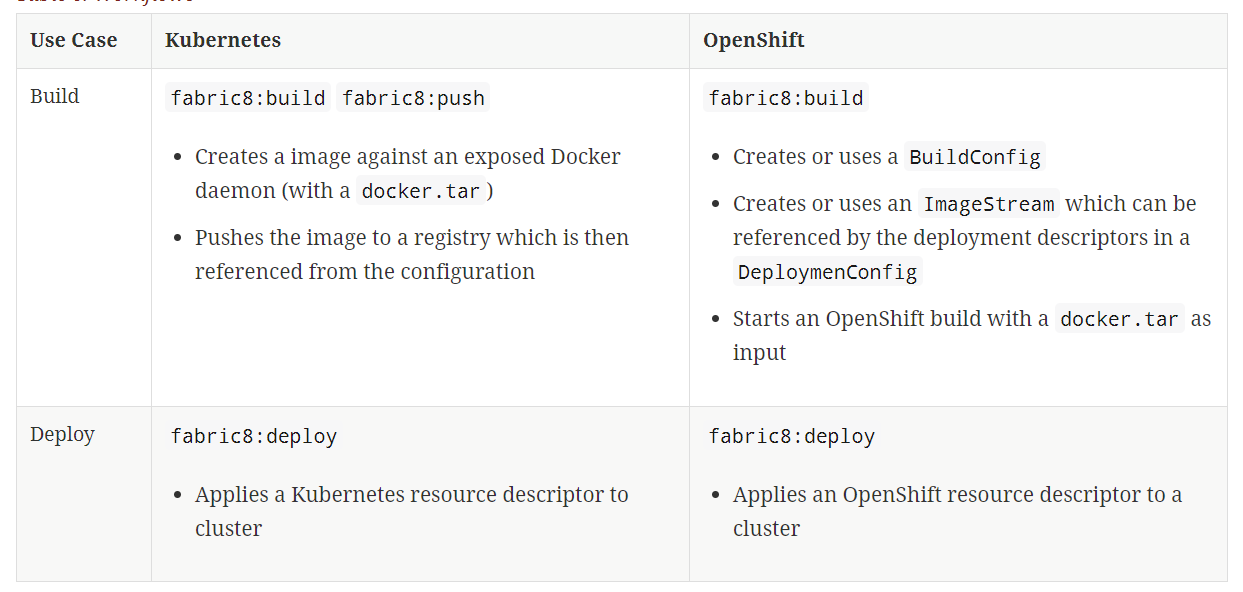
**fabric8-maven-plugin helps in deploying Java applications on Openshift.** OpenShift resource descriptors can be created or generated from [**fabric8:resource**](https://maven.fabric8.io/#fabric8:resource).

$ mvn fabric8:resource

* This will create following files in target/classes/META-INF/fabric8/openshift:

**deploymentconfig.yml, service.yml, route.yml**

* The generated yaml files are packaged within the Maven artifacts and can be deployed to a running orchestration platform with [**fabric8:appl**](https://maven.fabric8.io/#fabric8:apply)y and fabric8:deploy.



**Basic Tooling Changes in FEE:**

1. Need to create DeploymentConfig instead of Deployment.
2. Need to create service for DeploymentConfig.
3. Need to create route along with service.
4. Need to create imagestreams from images.
5. Need to create BuildConfigs.

NOTE: Further tooling changes need to be identified.

**MINISHIFT:**

**Minishift** is a **tool** that helps you run OpenShift locally by launching a single-node OpenShift Origin cluster inside a virtual machine. You can run Minishift on Windows, Mac OS, and GNU/Linux operating system.

**Installation on Windows:**

1. Install VirtualBox (Used version-- 6.1.4)
2. Install latest minishift (Used version—1.34.1)
3. Start Minishift using virtualbox

Goto cmd and change directory where minishift is downloaded and type:

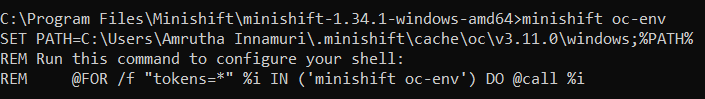


1. Once minishift gets started, you can see webconsole url

$ minishift console –url

1. To interact with minishift through commands, use openshift CLI.

$ minishift oc-env



Use the first line of output SET command, copy and paste it in cmd. Then you can use oc commands.

1. Login as “developer” using oc CLI

$ oc login https://<minishift-ip>:8443 --token=<token-id>

NOTE: token-id will be user specific and is unique one.

1. Provide admin-privileges to “developer” inorder to avoid git connectivity and authorization issues.
2. To get docker related info which is inbuilt in minishift

$ minishift docker-env

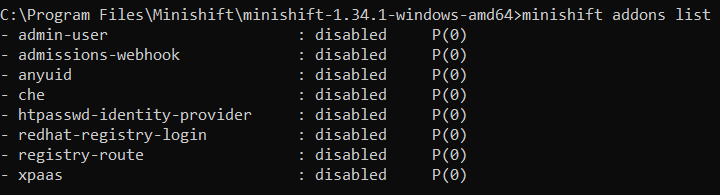
1. To use inbuilt docker of minishift

$ minishift ssh



**Adding registry-route in Minishift:**

1. In minishift cmd, check addons list



1. As registry-route is disabled, apply registry-route to minishift

$ minishift addon apply registry-route

1. Now login as user with admin-privileges into webconsole. There in “default” project, you will get new deploymentconfig with “docker-registry” name.
2. Now create route to the deploymentconfig created above. Make it as a secured route and add TLS termination type as Edge.
3. The routing URL which is formed there will now be the default docker-registry URL for deploying purpose.

**Adding openshift docker-registry as insecure-registry:**

 Insecure registries should be added using the **–insecure-registry** option to allow for the dockerdaemon to pull images from the repository.

* From docker prompt, open

$ vi  /etc/sysconfig/docker

* Then add the following line:

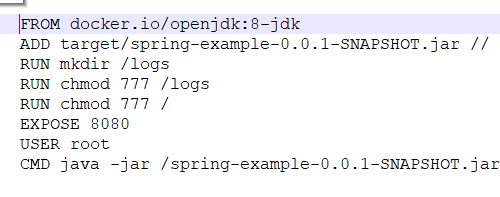
**INSECURE\_REGISTRY="--insecure-registry=<docker-registry-url>"**

* Then restart docker daemon.

**$ sudo systemctl restart docker.service**

**Deploying Java application on Minishift:**

1. Pull openjdk:8-jdk image into openshift docker registry.
2. Take a sample java application and create a Dockerfile as:



3.Use fabric8-maven-plugin in pom.xml



Here in pom.xml,

* + Used docker host as https://<minishift-ip>:2376
  + Used default certs prvided by minishift in certpath.

1. Create a new project through oc CLI

**$ oc new-project sampleproject**

1. Create a new imagestream

**$ oc create new is sampleimg**

1. Now tag docker image created with newly created imagestream in docker prompt as:

**$ sudo docker tag <docker-image>:tag <docker-registry-url>/<project-name>/<imagestream-name>:tag**

1. Now push the imagestream into openshift docker registry, to create deployments and pods in minishift.

**$ sudo docker push <docker-registry-url>/<project-name>/<imagestream-name>:tag**

NOTE:

* While pushing imagestream, getting invalid certificates issue and authentication issue for default-docker registry.Exact solution is not yet identified.