**Group B Assignment No. 10(b)**

**Title:** Install and use OpenShift

**Aim:** Install and demonstrate the use of PaaS tool - OpenShift to design a web application.

**Objective:** To study and demonstrate the use of Openshift for designing a web applications.

**Theory:**

**OpenShift**

* OpenShift is RedHat’s cloud development Platform as a Service (PaaS). The free and open source cloud-based platform allows developers to create, test and run their applications and deploy them to the cloud.
* With OpenShift you have a choice of offerings, including online, on-premise, and open source project options.
* OpenShift provides support for Node.js, Ruby, Python, PHP, Perl, and Java and is extensible so that users can add support for other languages. Resources allocated for applications can be automatically or manually scaled so that as demand increases there is no degradation of performance.
* OpenShift provides portability through the DeltaCloud API so customers can migrate deployments to other cloud computing vendor environments.

**OpenShift Architecture**

Docker provides the abstraction for packaging and creating Linux-based, lightweight containers. Kubernetes provides the cluster management and orchestrates Docker containers on multiple hosts.

OpenShift adds:

* Source code management, builds, and deployments for developers
* Managing and promoting [images](https://docs.openshift.org/latest/architecture/core_concepts/containers_and_images.html#docker-images) at scale as they flow through your system
* Application management at scale
* Team and user tracking for organizing a large developer organization

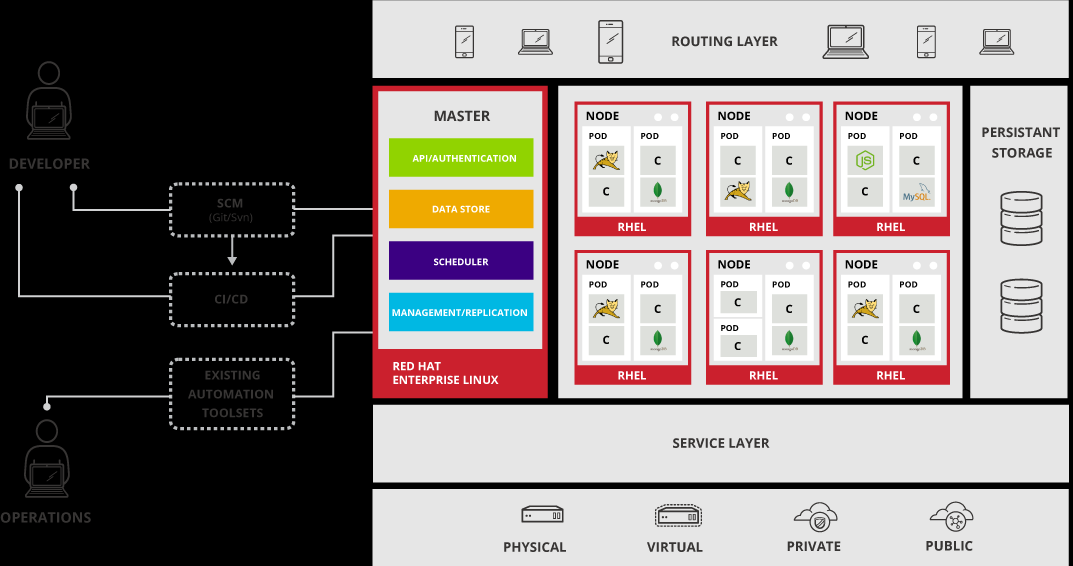
OpenShift has a microservices-based architecture of smaller, decoupled units that work together. It can run on top of (or alongside) a [Kubernetes cluster](https://docs.openshift.org/latest/architecture/infrastructure_components/kubernetes_infrastructure.html), with data about the objects stored in [etcd](https://docs.openshift.org/latest/architecture/infrastructure_components/kubernetes_infrastructure.html#master), a reliable clustered key-value store. Those services are broken down by function:

* REST APIs, which expose each of the core objects.
* Controllers, which read those APIs, apply changes to other objects, and report status or write back to the object.

Users make calls to the REST API to change the state of the system. Controllers use the REST API to read the user’s desired state, and then try to bring the other parts of the system into sync. For example, when a user requests a build they create a "build" object. The build controller sees that a new build has been created, and runs a process on the cluster to perform that build. When the build completes, the controller updates the build object via the REST API and the user sees that their build is complete.

The controller pattern means that much of the functionality in OpenShift is extensible. The way that builds are run and launched can be customized independently of how images are managed, or how deployments happen. The controllers are performing the "business logic" of the system, taking user actions and transforming them into reality. By customizing those controllers or replacing them with your own logic, different behaviors can be implemented. From a system administration perspective, this also means the API can be used to script common administrative actions on a repeating schedule. Those scripts are also controllers that watch for changes and take action. OpenShift makes the ability to customize the cluster in this way a first-class behavior.

To make this possible, controllers leverage a reliable stream of changes to the system to sync their view of the system with what users are doing. This event stream pushes changes from etcd to the REST API and then to the controllers as soon as changes occur, so changes can ripple out through the system very quickly and efficiently. However, since failures can occur at any time, the controllers must also be able to get the latest state of the system at startup, and confirm that everything is in the right state. This resynchronization is important, because it means that even if something goes wrong, then the operator can restart the affected components, and the system double checks everything before continuing. The system should eventually converge to the user’s intent, since the controllers can always bring the system into sync.



**Features of Openshift:** OpenShift Enterprise by Red Hat provides Application Development and IT Operations teams the ability to accelerate application delivery with the speed and consistency that business demands. Using the automation technologies and cloud architecture of OpenShift Enterprise to standardize and streamline developer workflows, IT organizations can provision quickly, build efficiently, and get apps to market faster. Following are the features of OpenShift:

1. **Self-Service Platform :** Developers can quickly and easily create applications on demand directly from the tools they use most. Operators can leverage placement and policy to orchestrate environments that meet their best practices.

### Polyglot, Multi-Language Support : Developers have the choice and the ability to run multiple languages, frameworks, and databases on the same platform. Allows customers to more easily take advantage of the Docker eco-system.

### Container-Based : OpenShift provides an immutable, container-based platform based on Docker to deploy and run applications and microservices.

### Automation : OpenShift automates application builds, deployments, scaling, health management and more leveraging integrated components from Kubernetes.

### Persistence : OpenShift allows platform architects the choice to incorporate persistence into their application component while still be able to offer stateless cloud native design.

### Application Centric Networking : Software defined networking found in OpenShift allow operators to leverage OpenSwitch or plug into an existing investment. In either case, OpenShift offers real IP services to dynamic end points across the PaaS platform.

### Multiple Interaction Models : Developers can create and manage applications utilizing a rich set of command-line tools, a powerful multi-device web console, or an Eclipse-based Integrated Development Environment such as JBoss Developer Studio.

**Installation Steps :**

1. Create an account on OpenShift.

**Explanation** – To be able to deploy our application on OpenShift, it is necessary that we must have an Openshift account.

Open the following website in the browser –

www.openshift.com

Enter your credentials. You will be asked to verify your details through your email account.

1. Now open Eclipse EE.
2. Goto Help-> Eclipse Market place
3. Search for JBoss
4. Select JBoss Openshift 2 tools and click install now
5. Complete the installation.
6. Now go to New-> Project
7. Select ‘Openshift Application’ in ‘Other’.
8. Give the project name and domain name and log in through your Openshift account.
9. Generate the ‘RSA key’ using ‘Key Management’ and add it to your project through ‘key wizard’. Select workspace as the cloning destination for Git.
10. Click on Finish
11. In project explorer window right click on the project you created and select New-> HTML file.
12. Give a name to html file and click on finish
13. Code the html file.
14. Now open : window -> perspective -> other -> git. Go into the Git Staging perspective and drag all the unstaged files to the ‘staged section’. Click on ‘commit’.
15. Go to Java EE server perspective. Right click on application and select ‘publish’.
16. Right click on the project, select Run as -> Run on Server.

**Mathematical Modeling**:

**Input**

**Output**

**Process**

Let ‘S’ be the system such that,

S = {I, O, Fn, Sc, Fc}

Where,

I -> {I1, I2, . . . , In} : set of inputs

O -> {O1, O2, . . . , On} : set of outputs

Fn -> {Fn1, Fn2, . . . ,Fnn} : set of functions

Sc -> {Sc1, Sc2, . . . ,Scn} : set of success cases

Fc -> {Fc1, Fc2, . . . ,Fcn} : set of failure cases

**I: Set of Inputs**

I1: Openshift credentials.

I2: Active internet connection.

I3: numbers for sorting

**O: Set of Outputs**

O1: sorted list

O2: Deployed web app

**Fn: Set of Functions**

Fn1: merge sort (recursive)

1. partitioning the list
2. comparing values
3. appropriate swapping

Fn2: Display

**Sc: Success Cases**

Sc1: valid credentials entered.

Sc2: internet connection is active

Sc3: sorted array displayed.

**Fc: Failure Cases**

Fc1: Internet connection lost

Fc2: Array not sorted properly due to wrong logic.

Fc3:Not a valid subscription for the OpenShift account .

**Input** : Set of numbers to be sorted

**Output**: Sorted List .

**Platform :** Ubuntu 14.04 , Eclipse EE Mars , jdk 1.8+

**Conclusion** : Thus a webapp is created and deployed on cloud using Openshift.

**Faqs**:

1) What is PaaS?

2) What are some other examples of PaaS?

3) List the languages, databases and servers that are supported by Openshift.