Openshift Basics

POD:

A set of one or more containers that are deployed onto a Node and share a unique IP and Volume (persistent Storage). Pods also define the security and runtime policy for each container

LABEL:

Labels are key-value pairs that can be assigned to any resource in the system for grouping and selection. Many resources use lables to identify sets of other resources.

Volume:

Container are not persistent by default; their contents are cleared when they are restarted.

Volumes are mounted file-systems available to pods and their container which may be backed by a number of host-local or network attached storage endpoints.

The simplest volume type is EmptyDir, Which is a temporary directory on a single machine. Administrators may also

Allow you to request a **Persistent Volume** that is automatically attached to your pods.

Node:

Machines set up in the cluster to run containers. Nodes are usually managed by administrators and not by end users.

**Replica Set:**

\* A replica set is a core Kubernetes object called **ReplicaSet**.

a replica set ensures that a specified number of pod replicas are running at any given time.

The difference between a replica set and a replication controller is that a replica set supports set-based selector requirements whereas a replication controller only supports equality-based selector requirements.

NOTE:

* Only use replica sets if you require custom update orchestration or do not require updates at all, otherwise, use [Deployments](https://docs.openshift.com/container-platform/3.9/architecture/core_concepts/deployments.html#deployments-and-deployment-configurations).
* Replica sets can be used independently, but are used by deployments to orchestrate pod creation, deletion, and updates.
* Deployments manage their replica sets automatically, provide declarative updates to pods, and do not have to manually manage the replica sets that they create.

**Replication controllers:**

A replication controller is a core Kubernetes object called **ReplicationController**.

A [replication controller](https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller/) ensures that a specified number of replicas of a pod are running at all times. If pods exit or are deleted, the replication controller acts to instantiate more up to the defined number.

A replication controller configuration consists of:

1. The number of replicas desired (which can be adjusted at runtime).
2. A pod definition to use when creating a replicated pod.
3. A selector for identifying managed pods.

NOTE:

* A selector is a set of labels assigned to the pods that are managed by the replication controller. These labels are included in the pod definition that the replication controller instantiates.
* The replication controller uses the selector to determine how many instances of the pod are already running in order to adjust as needed.
* The replication controller does not perform auto-scaling based on load or traffic, as it does not track either. Rather, this would require its replica count to be adjusted by an external auto-scaler.

**Jobs:**

* A job is similar to a replication controller, in that its purpose is to create pods for specified reasons.
* The difference is that replication controllers are designed for pods that will be continuously running, whereas jobs are for one-time pods.
* A job tracks any successful completions and when the specified amount of completions have been reached, the job itself is completed

**Deployments and Deployment Configurations:**

Building on replication controllers, OpenShift Container Platform adds expanded support for the software development and deployment lifecycle with the concept of deployments.

In the simplest case, a deployment just creates a new replication controller and lets it start up pods.

However, OpenShift Container Platform deployments also provide the ability to transition from an existing deployment of an image to a new one and also define hooks to be run before or after creating the replication controller.

The OpenShift Container Platform **DeploymentConfig** object defines the following details of a deployment:

1. The elements of a **ReplicationController** definition.
2. Triggers for creating a new deployment automatically.
3. The strategy for transitioning between deployments.
4. Life cycle hooks.

 Openshift Commands

* GET OPEN\_SHIFT\_REGISTRY

**minishift openshift registry**

**(or)**

**oc --as system:admin get -o "jsonpath={.spec.clusterIP}:{.spec.ports[\*].port}" service/docker-registry -n default**

* Login Command From terminal

**To login as administrator:**

**oc login -u system:admin**

* TOKEN Command

**oc whoami -t**

* Verify the correct user:

**\* oc whoami**

**Or**

**oc whoami --config=/etc/origin/master/admin.kubeconfig**

* **CLUSTER ROLE TO the USER**

**oc adm policy add-cluster-role-to-user cluster-admin <user-name (ex: developer)>**

* **Get the information about the Pods:**

**oc get pods**

* **Get the nodes**

**oc get nodes**

* Create Openshift username and password

**htpasswd -b /etc/origin/openshift-passwd student redhat**

**\* Logs, Builds and route commands**

**oc get builds**

**oc logs buil/app-name**

**oc get pods**

**\* Delete project**

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

* **Create the project**

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

* Check status

**oc status**

* Describe the resources in openshift container

**oc describe <resource resource-name>**

* **Export the definition of the resource:: oc export**
* **Create the resources from a resource definition::: oc create**
* **Remove the resource from OCP cluster::: oc delete resource\_type name**