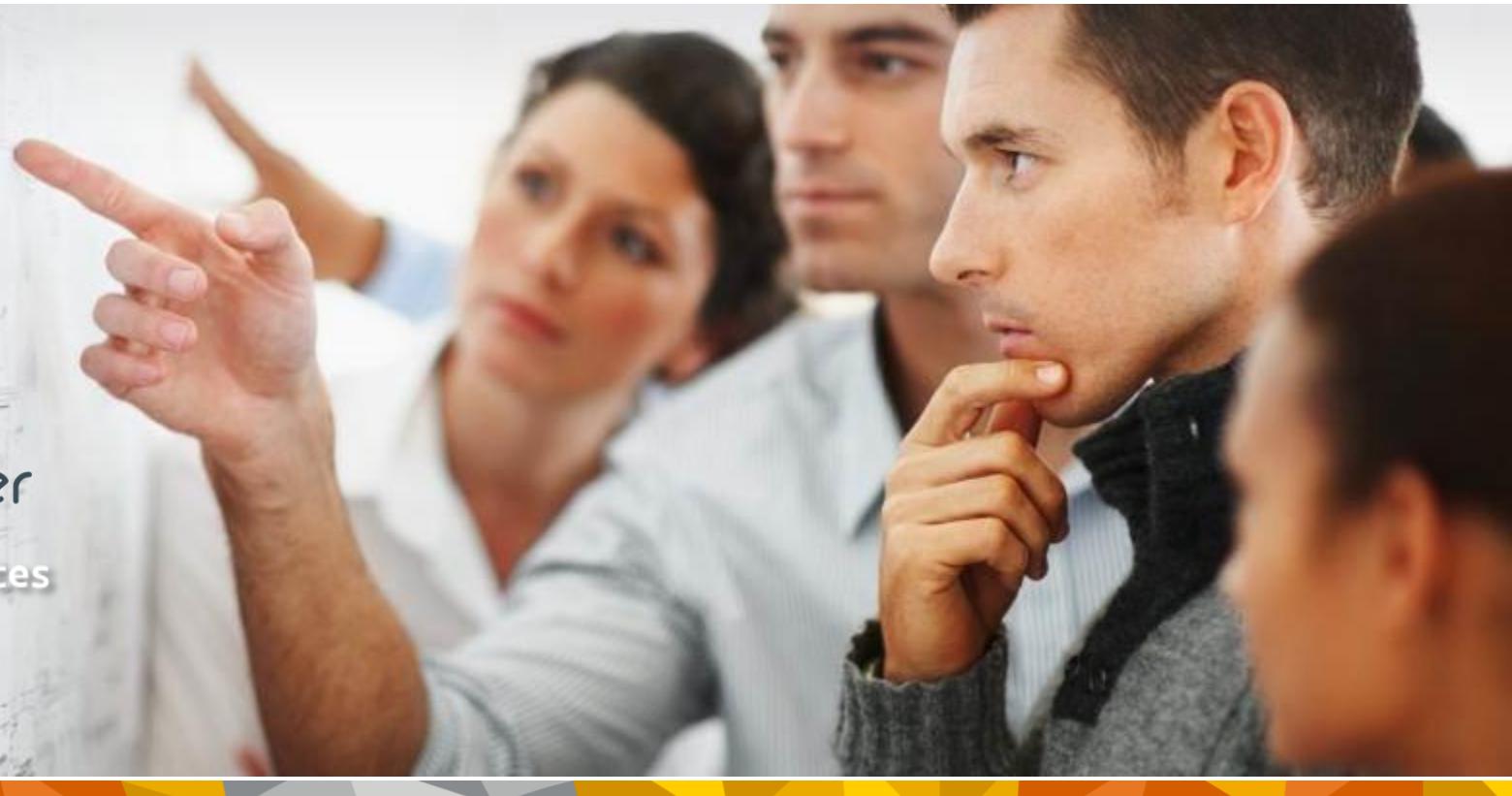




Optum Global Solutions



OPENSHIFT



<https://oneconnect.uhg.com/docs/DOC-53617>

## Open Shift Enterprise V3

Training Presentation for OpenShift Enterprise  
Sreejith Kaimal

Day 02

# More Terminology ..

---

## PROJECT

- A **project** allows a community of users to organize and manage their content in isolation from other communities.
- A project in Openshift contains multiple objects to make up a logical application.
- Can have separate name, display name and description.
- Each project scopes its own set of objects, policies , constraints & Service accounts.

## TEMPLATE

- A template describes a set of objects that can be parameterized and processed
- It produces a list of objects for creation by OpenShift.
- You can define new templates to make it easy to recreate all the objects of your application.

## ROUTE

- An route is a way to expose a service by giving it an externally-reachable hostname.
- Each route consists of a route name, service selector, and (optionally) security configuration.
- A defined route and the endpoints identified by its service can be consumed by a router.
- Router also provides load balancing features.

## VOLUME

- Volumes are mounted file systems available to pods and their containers
- Volumes may be backed by a number of host-local or network attached storage endpoints.
- Persistent volume can be attached to a pod.
- Similar to a mount point.

# Request CLI

Request Installation of OpenShift CLI tool via appstore.

1. The OpenShift CLI exposes commands for managing your applications and tools to interact with each component within OSE.
2. Authorization is handled in the OpenShift policy engine.
3. The developer CLI uses the **oc** command, which is used for project-level operations.
4. The administrator CLI uses the **oadm** command for more advanced, administrator operations.

Various object types that CLI supports is listed below

Object Type	Abbreviated Version
build	
buildConfig	bc
deploymentConfig	dc
event	ev
imageStream	is
imageStreamTag	istag
imageStreamImage	isimage
job	
LimitRange	limits
node	
pod	po
ResourceQuota	quota
replicationController	rc
secrets	
service	svc
ServiceAccount	serviceaccounts
persistentVolume	pv
persistentVolumeClaim	pvc

The screenshot shows a web browser window titled 'AppStore - Home' with the URL 'appstore.uhc.com'. The page header includes the 'UNITEDHEALTH GROUP' logo and navigation links for 'Home', 'Welcome, Kaimal, Sreejith S', and 'Help'. A search bar at the top right contains the text 'openshift' with a magnifying glass icon. Below the search bar, there are three tabs: 'Catalog', 'Requests', and 'Admin'. The main content area displays a table titled 'Search Result: openshift' with two items found. The table has columns for 'App Name', 'Vendor Name', 'Win 7', and 'Win 8'. The first item is 'OpenShift CLI 3.1.0.4' by RedHat, marked as Yes for both Win 7 and Win 8. The second item is 'OpenShift CLI 3.1.1.6' by RedHat, also marked as Yes for both Win 7 and Win 8. At the bottom of the table, it says '1 - 2 of 2 items'. A small note at the bottom right of the page states '© 2016 - AppStore Team (Version: 3.1.0.21727 Last Updated: 3/16/2016 12:04:14 PM)'.

The CLI is ideal in situations where you are:

1. Working directly with project source code.
2. Scripting OpenShift operations.
  - Using templates , automating custom environment creation etc.
  - Update environment variables
  - Update labels on objects etc.
3. Restricted by bandwidth resources and cannot use the web console.
4. View details about containers that are in a stuck state.
5. Capture build logs etc

# Setup CLI

Check if the OC client tool is installed on your machine.  
Launch only from 'cmd'

C:\Windows\system32\cmd.exe

```
C:\>Users\skaima5>oc login https://ose-ctc-core.optum.com
Authentication required for https://ose-ctc-core.optum.com:443 (openshift)
Username: skaima5
Password:
Login successful.

Using project 'skcore'.
Welcome! See 'oc help' to get started.

C:\>Users\skaima5>
```

- The command prompts for the Openshift server URL. (give Core or DMZ)
- The command prompts for login credentials: a user name and password.
- A session is established with the server, and a session token is received.
- If you do not have a project, information is given on how to create one.

Start oc from command prompt.  
Use Command  
• oc login  
• You should be within the network.  
• The session is not limited to the current cmd prompt.  
• Server name would be prompted if you have not created/deleted your kube config.

Computer > OS (C:) > Users > skaima5 > .kube

Organize Open Share with New folder

Name	Date
config	2/3/2018

TextPad - C:\Users\skaima5\.kube\config

```
apiVersion: v1
clusters:
- cluster:
  server: https://ose-ctc-core.optum.com
  name: ose-ctc-core-optum-com:443
contexts:
- context:
  cluster: ose-ctc-core-optum-com:443
  namespace: skcore
  user: skaima5/ose-ctc-core-optum-com:443
  name: skcore/ose-ctc-core-optum-com:443/skaima5
current-context: skcore/ose-ctc-core-optum-com:443/skaima5
kind: Config
preferences: {}
users:
- name: skaima5/ose-ctc-core-optum-com:443
  user:
    token: goIXZDt171JHUVjQfmslwM5y3wG7zdEh4Stx-hGa7FM
C:\>Users\skaima5>
```

Check the current configuration using  
• oc config view  
Check the projects that you have access to.  
• oc get projects

C:\users\<ntid>\.kube  
Contains the kube configuration file

[https://docs.openshift.com/enterprise/3.1/cli\\_reference/get\\_started\\_cli.html](https://docs.openshift.com/enterprise/3.1/cli_reference/get_started_cli.html)

# Manage CLI Profiles

```
apiVersion: v1
clusters:
- cluster:
  insecure-skip-tls-verify: true
  server: https://ose-core.optum.com:443
  name: ose-core-optum-com:443
- cluster:
  insecure-skip-tls-verify: true
  server: https://ose-dmz.optum.com:443
  name: ose-dmz-optum-com:443
- cluster:
  insecure-skip-tls-verify: true
  server: https://ose-core.optum.com
  name: v3core
- cluster:
  insecure-skip-tls-verify: true
  server: https://ose-dmz.optum.com
  name: v3dmz
contexts:
- context:
  cluster: ose-dmz-optum-com:443
  namespace: sk-dmz
  user: skaima5/ose-dmz-optum-com:443
  name: sk-dmz/ose-dmz-optum-com:443/skaima5
- context:
  cluster: ose-core-optum-com:443
  namespace: sk-project
  user: skaima5/ose-core-optum-com:443
  name: sk-project/ose-core-optum-com:443/skaima5
- context:
  cluster: v3core
  user: skaima5/ose-core-optum-com:443
  name: skv3core
- context:
  cluster: v3dmz
  namespace: sk-dmz
  user: skaima5/ose-dmz-optum-com:443
  name: skv3dmz
current-context: sk-dmz/ose-dmz-optum-com:443/skaima5
kind: Config
preferences: {}
users:
- name: skaima5/ose-core-optum-com:443
  user: {}
- name: skaima5/ose-dmz-optum-com:443
  user: {}
```

The **clusters** section defines connection details for Openshift clusters, including the address for their master server.  
A cluster is nothing but a server definition.

A **context** is a mapping of server to user & project.  
You could have multiple contexts within the same server.  
The namespace points to the project that you are using.

The **current context** parameter points to the current server/project/user that is being used.  
All oc commands will use this configuration.  
Ps: *if you are switching servers, you need to do a logout & explicit login.*

The **users** section holds the user credentials for the logged in or previously logged user.  
When the user is logged in, it will contain a token key.

OpenShift allows multiple CLI configurations  
Contexts allow you to easily switch between multiple users across multiple OpenShift servers, or clusters, when using issuing CLI operations. Nicknames make managing CLI configuration easier by providing short-hand references to contexts, user credentials, and cluster details.

Command	Usage
set-credentials	oc config set-credentials <username>
set-cluster	oc config set-cluster <cluster_nickname> --server=https://ose-core.optum.com --insecure-skip-tls-verify=true
set-context	oc config set-context <context_nickname> --cluster=<cluster_nickname> --user=<user_nickname> --namespace=<namespace>
use-context	oc config use-context <context_nickname>

# Selecting a context to connect to OSE

There are two environments for OSE, so its better to configure both on your CLI config if you want to switch between them , here is how to do it.

## 1. Set Cluster Settings

```
➤ oc config set-cluster oseDmz --server=https://ose-ctc-dmz.optum.com --insecure-skip-tls-verify=true
```

## 2. Set Context Settings

```
➤ oc config set-context oseDmzCtx --cluster=oseDmz --user= skaima5/ose-dmz-optum-com:443 --namespace=dmz-proj-name
```

## 3. Switch context

- If you are switching context within the same cluster, then you can do in the same session.
- If you are switching context across clusters, then you need to logout and then switch context and then login again.

## 4. Switch context , the context switch takes care of pointing to the right cluster.

```
➤ oc config use-context oseDmzCtx
```

## 5. You can repeat steps 1 & 2 for as many servers you want to add. (e.g, ose-dmz and ose-core)

### Generic OC command instruction format

```
➤ oc <action> <object_type> <object_name_or_id>
```

### Create a project

```
➤ oc new-project hello-world --description="This is an example project to demonstrate OpenShift v3" --display-name="Hello World"
```

### Switch Project

```
➤ oc project hello-world
```

### Create an Application

```
➤ oc new-app mysql
```

### Check some OpenShift object types via oc, this lists a generic help on object types.

```
➤ oc types
```

```
C:\Users\skaima5>oc config view
apiVersion: v1
clusters:
- cluster:
    insecure-skip-tls-verify: true
    server: https://ose-core.optum.com:443
    name: ose-core-optum-com:443
- cluster:
    insecure-skip-tls-verify: true
    server: https://ose-dmz.optum.com:443
    name: ose-dmz-optum-com:443
- cluster:
    insecure-skip-tls-verify: true
    server: https://ose-core.optum.com
    name: v3core
- cluster:
    insecure-skip-tls-verify: true
    server: https://ose-dmz.optum.com
    name: v3dmz
contexts:
- context:
    cluster: ose-dmz-optum-com:443
    namespace: sk-dmz
    user: skaima5/ose-dmz-optum-com:443
```

# OC CLI Commands reference

Operation	Syntax
Get	<code>oc get &lt;objecttype&gt; &lt;objectname&gt;</code> <code>oc get pods</code>
Describe	<code>oc describe &lt;objecttype&gt; &lt;objectid&gt;</code> <code>oc describe pod jbosstom-1-17v5h</code>
Edit	<code>oc edit &lt;objecttype&gt;/&lt;objectname&gt;</code> <code>oc edit pod/jbosstom-1-17v5h</code>
Env	<code>oc env &lt;objecttype&gt;/&lt;objectname&gt; &lt;envvar&gt;=&lt;value&gt;</code> Some objects do not allow adding env variables directly.
Label	<code>oc label &lt;objecttype&gt; &lt;objectname&gt; &lt;label&gt;</code> <code>oc label pods foo unhealthy=true</code>
Stop	<code>oc stop -f &lt;file_path&gt;</code> <code>oc stop &lt;object_type&gt; &lt;object_name&gt;</code> <code>oc stop &lt;object_type&gt; -l &lt;label&gt;</code> <code>oc stop all -l &lt;label&gt;</code>
Delete	<code>oc delete -f &lt;file_path&gt;</code> <code>oc delete &lt;object_type&gt; &lt;object_name_or_id&gt;</code> <code>oc delete &lt;object_type&gt; -l &lt;label&gt;</code> <code>oc delete all -l &lt;label&gt;</code>
Expose	<code>oc expose &lt;object_type&gt; &lt;objectname&gt;</code>
Project	<code>oc project &lt;project_name&gt;</code>
New Project	<code>oc new-project &lt;project_name&gt;</code>

Operation	Syntax
Logs	<code>oc logs -f &lt;podname&gt; &lt;containername&gt;</code>
Exec	<code>oc exec &lt;podname&gt; [-c &lt;containerID&gt;] &lt;command&gt;</code> <code>oc exec -it --pod=jbosstom-1-17v5h /bin/bash</code>
RSH	<code>oc rsh &lt;podname&gt;</code> <code>oc rsh jbosstom-1-17v5h</code>
port-forward	<code>oc port-forward &lt;pod_ID&gt; &lt;first_port_ID&gt; &lt;second_port_ID&gt;</code>
Run Proxy	<code>oc proxy --port=&lt;port_ID&gt; --www=&lt;static_dir&gt;</code>
Policy	Manage Policy Authorization. <code>oc policy [--options]</code>
Scale	<code>oc scale &lt;object_type&gt; &lt;object_id&gt; --replicas=&lt;Num_replicas&gt;</code>
Start Build	<code>oc start-build --from-build=&lt;build_name&gt;</code> <code>oc start-build &lt;buildConfig_name&gt;</code>
View Build Log	<code>oc build-logs &lt;build_name&gt;</code>
Deploy	<code>oc deploy &lt;deploymentconfig&gt;</code>
Login	<code>oc login [-u=&lt;username&gt;] [-p=&lt;password&gt;] [-s=&lt;server&gt;] [-n=&lt;project&gt;] [--certificate-authority=&lt;/path/to/file.crt&gt;] [--insecure-skip-tls-verify]</code>

# OSE Console

The image shows four windows from the OpenShift Web Console:

- Welcome to OpenShift:** Shows the main landing page with a "New Project" button.
- New Project:** A modal window for creating a new project named "skaimal" with a display name "My First Hello Project" and a description about testing the S2I template process.
- Select Image or Template:** A window showing options for adding components to a project. It highlights "Builder Image" and "Template". Two specific templates are circled with red circles: "cakephp-example" under Instant Apps and "fis-java Openshift:1.0" under xPaaS.
- Project Overview:** A dashboard for the "skaimal" project. It includes sections for Overview, Browse, and Settings. A callout box points to the "Add to Project" button, and another points to the "Get started with your project" section.

Annotations provide additional context:

- A callout box points to the "Name" field in the New Project modal: "Name is the qualifying name of your project which will be used in scripts."
- A callout box points to the "Template" section in the Select Image or Template window: "Template : Describes a set of resources that are intended to be together. Builder Image looks for git repositories to bundle the source & image together."
- A callout box points to the "Add to Project" button in the Project Overview: "Check various console options that are available to manage your projects. A project would need services / applications running on the pods."
- A URL at the bottom: [https://docs.openshift.com/enterprise/3.0/using\\_images/db\\_images/mysql.html](https://docs.openshift.com/enterprise/3.0/using_images/db_images/mysql.html)

# OSE Console

OpenShift Web Console https://ose-core.optum.com/console/project/skaimal/create/fromtemplate?name=mysl

Parameters

\* DATABASE\_SERVICE\_NAME  
mysql-db

Database service name

\* MYSQL\_USER  
sk

Username for MySQL user that will be used for accessing the database

\* MYSQL\_PASSWORD  
12345

Password for the MySQL user

\* MYSQL\_DATABASE  
skdb

Database name

\* VOLUME\_CAPACITY  
512Mi

Volume space available for data, e.g. 512Mi, 2Gi

Labels

Each label is applied to each created resource.

skdblavel1 mysql-sk-label template mysql-persistent-term

Create Cancel

Colored Rings show the status of the pods. Yellow is a warning. Blue is running.

Many features are not ready yet (as on Feb 2016), so errors could be environmental or specific to application. Please use the oneconnect forum, if in doubt.

Application created. Continue to overview.

Manage your app

The web console is convenient, but if you need deeper control you may want to try our command line tools.

Command line tools

Download and install the `oc` command line tool. After that, you can start by logging in, switching to this particular project, and displaying an overview of it, by doing:

```
oc login https://ose-core.optum.com:443
oc project skaimal
oc status
```

For more information about the command line tools, check the [CLI Reference](#) and [Basic CLI Operations](#).

While creating an application from a Template or an Image, the form fields are different.  
An Image build expects for a git repo which it can use to bundle with the image.  
A template build simply creates from a standard template file—Good for creating a DB or MQ or anything similar project specific.

MyFirstProject

OVERVIEW

BROWSE

SETTINGS

SERVICE mysql 3306/TCP → 3306 Create Route

DEPLOYMENT: MYSQL #2 20 minutes ago from config change

CONTAINER: MYSQL  
Image: rhsc1/mysql-56-rhel7 Ports: 3306/TCP

# OSE Console

OpenShift Web Console

https://ose-core.optum.com/console/project/skaimal/settings?main-tab=openshiftCc

Project Settings

General information

Name: skaimal  
Display name: MyFirstProject  
Description: Hello World

Quota

Resource type	Used	Max
cpu	250 millicores	4 cores
memory	256 MiB	2 GiB
persistentvolumeclaims	1	2
pods	1	8
replicationcontrollers	1	5
resourcequotas	1	10
secrets	9	20
services	1	5

Resource limits

Resource type	Min	Max	Default Request	Default Limit	Max Limit/Request Ratio
Pod cpu	250	2	—	—	—
	millicores	cores			
Pod memory	256 MiB	2 GiB	—	—	—
Container cpu	250	2	250 millicores	250 millicores	—
	millicores	cores			
Container memory	256 MiB	2 GiB	256 MiB	256 MiB	—

Assigned resource limits to the logged account.  
Ask for more if you need more.

https://ose-dmz.optum.com/console/project/sk-dmz/o

Apps Bookmarks UnitedHealth Citrix XenApp - Log... Whitehat Sentinel

OPENSHIFT ENTERPRISE

Projects skdmz Filter by labels Label key

Builds (highlighted)

Deployments

Events

Image Streams

Pods (highlighted)

Routes

Services

Storage

SERVICE : JBOSTOM  
jbostom-sk-dmz.ose-dmz.optum.com

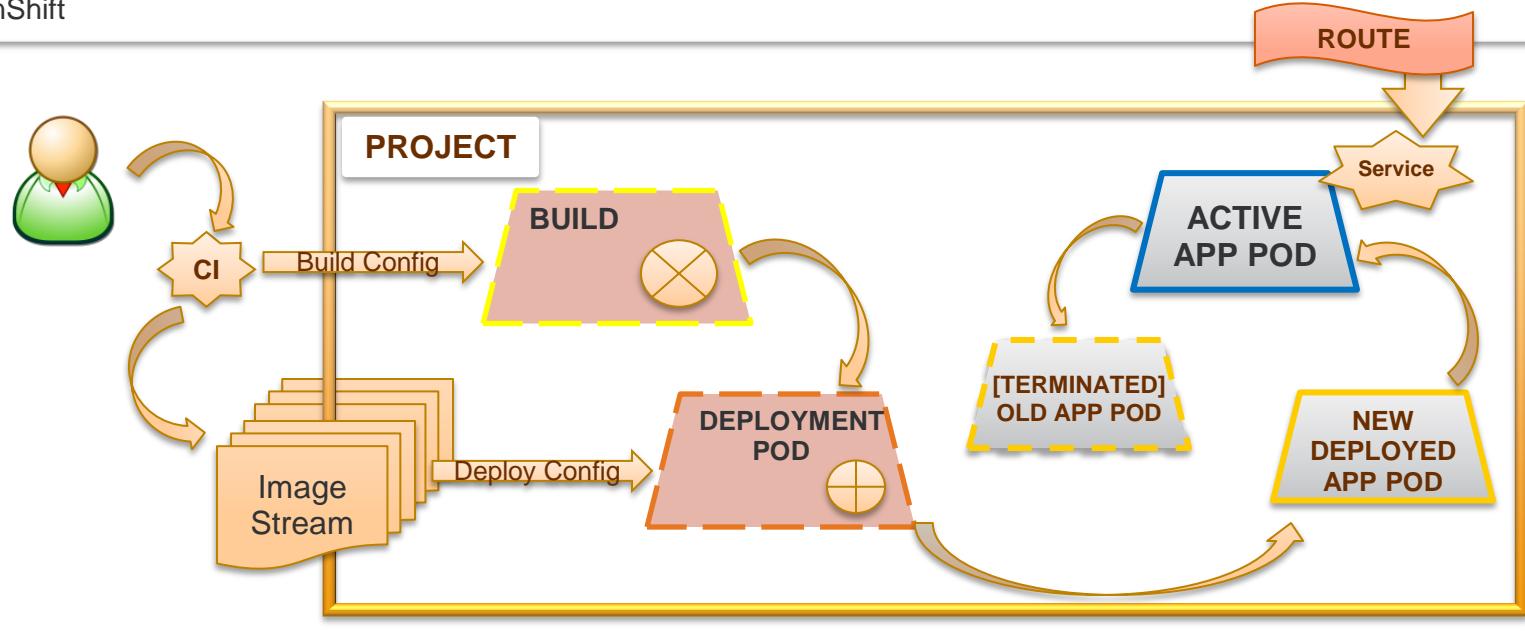
DEPLOYMENT: JBOSTOM, #1

CONTAINER: JBOSTOM  
Image: sk-dmz/jbostom (410)  
Build: #1 from <> Delete pod authored by sskaimal  
Ports: 8080/TCP, 8443/TCP

Each 'Object' that you add to the 'Project' creates its own configuration files. These configuration items trigger the changes.  
You can see BuildConfig, Deployment, Image Stream, Pods, Routes , Services in the 'Browse' menu.  
Deleting a config, triggers deletion of the object.

# Core Objects

1. **Containers & Images** : Linux container technologies are lightweight mechanisms for isolating running processes so that they are limited to interacting with only their designated resources
2. **Pods**: The smallest compute unit that can be defined, deployed, and managed.
3. **Services** :Service serves as an internal load balancer. It identifies a set of replicated pods in order to proxy the connections it receives to them. Backing pods can be added to or removed from a service arbitrarily while the service remains consistently available.
4. **Projects** :Kubernetes namespace provides a mechanism to scope resources in a cluster.
5. **Users** :Regular users, System Administrators, Service accounts
6. **Builds**: the process of transforming input parameters into a resulting object (Source to Image)
7. **Image streams**: can be used to automatically perform an action, such as updating a deployment, when a new image is made available.
8. **Deployment**: creates a new replication controller and lets it start up pods. Replication controller ensures that a specified number of replicas of a pod are running at all time
9. **Routes** : is a way to expose a service by giving it an externally-reachable hostname
10. **Templates** describes a set of objects that can be parameterized and processed to produce a list of objects for creation by OpenShift



# Build & Deploy

Process of 'not only' building 'code' + 'Applications' but also the environments.

It does the process of Transforming input parameters into a resulting object

A '*BuildConfig*' object is the definition of the entire build process.

OpenShift leverages Kubernetes by creating Docker containers from build images and pushing them to a Docker registry.

The OpenShift build system provides extensible support for build strategies that are based on selectable types specified in the build API. There are three build strategies available:

## 1. Docker build

- Builds Docker images from a Dockerfile and a 'context'. A build's context is the files located in the specified PATH or URL.
- The URL parameter can specify the location of a Git repository; the repository acts as the build context.

## 2. Source-to-Image (S2I) build

- Tool for building reproducible Docker images.
- It produces ready-to-run images by injecting application source into a Docker image and assembling a new Docker image.
- New image incorporates the base image (the builder) and built source and is ready to use with the `docker run` command.

## 3. Custom build

- The Custom build strategy allows developers to define a specific builder image responsible for the entire build process.
- <https://hub.docker.com/r/openshift/origin-custom-docker-builder/>

## Source To Image Build ( S2I )

'Framework' that makes it easy to write images that take application source code as an input and produce a new image that runs the assembled application as output.

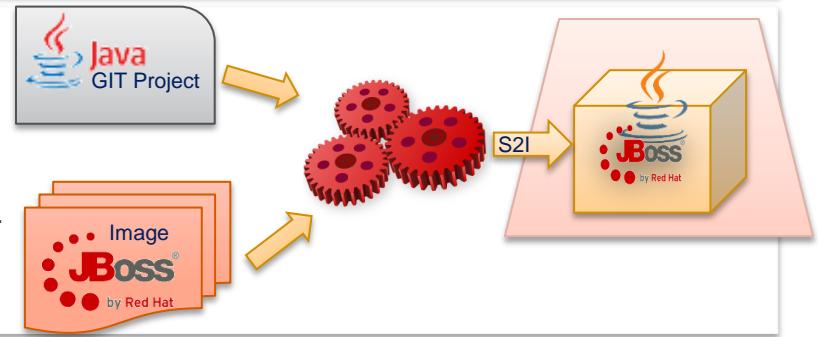
The build process consists of the following three fundamental elements, which are combined into a final Docker image:

- Sources
- S2I scripts
- Builder image

The builder places 'Sources' & 'Scripts' inside the 'Image'

## Build Process in S2I

1. Download the S2I Build Script.
2. Download the application source code that needs to be built (git)
3. Bundle source code as 'tar' file to be placed inside the Image(pom)
4. Assemble -> Run Build -> Install tar content into the Image sources.
5. Run script takes care of starting the application within the image/container.
6. Any errors in each step causes the build to fail.
7. Usage provides documentation on how to use the build.



# Build & Deploy

You can create a new OpenShift application using the web console **or** by running the `oc new-app` command from the CLI.

OpenShift creates a new application by specifying source code, images, or templates.

The new-app command looks for images on the local Docker installation (if available), in a Docker registry, or an OpenShift image stream.

If creating a **Source** build, **new-app** attempts to determine which language builder to use based on the presence of certain files in the root of the repository, e.g for a JEE project, it should have a *pom.xml* in the root folder.

The new-app creation triggers the following objects to be created.

Artifact	Description
BuildConfig	A BuildConfig is created for each source repository specified in the command line. The BuildConfig specifies the strategy to use, the source location, and the build output location.
ImageStream	An image stream can be used to automatically perform an action, such as updating a deployment, when a new image is made available. For BuildConfig, two Image Streams are usually created: one to represent the input image (the builder image in the case of Source builds or FROM image in case of Docker builds), and another one to represent the output image. If a Docker image was specified as input to new-app, then an image stream is created for that image as well.
DeploymentConfig	A DeploymentConfig is created either to deploy the output of a build, or a specified image.
Service	The new-app command attempts to detect exposed ports in input images. It uses the lowest numeric exposed port to generate a service that exposes that port. In order to expose a different port, after new-app has completed, simply use the <code>oc expose</code> command to generate additional services.

# Build & Deploy your Application using S2I

The image consists of four screenshots of the OpenShift Web Console, each with numbered callouts (1 through 4) and yellow boxes highlighting specific fields or instructions.

- Screenshot 1:** Shows the 'skaimal' project overview. A yellow box highlights the 'Add to Project' button, with the text 'Create a new Project. Click Add to Project'. A red arrow points from this box to the 'Add to Project' button. Another red arrow points from the 'Add to Project' button to the 'From Image' section of the next screenshot.
- Screenshot 2:** Shows the 'Instant Apps' catalog. A red circle with the number '2' is in the top right. A yellow box highlights the 'jboss-eap64-openshift:1.1' entry under the 'PHP' category, with the text 'Select the JBoss eap64 Image.' A red arrow points from this box to the 'jboss-eap64-openshift' configuration screen in the third screenshot.
- Screenshot 3:** Shows the 'jboss-eap64-openshift' configuration screen. A red circle with the number '3' is in the bottom right. A yellow box highlights the 'Name' field with the value 'sampleapp1', with the text 'Provide the name of your application.' A red arrow points from this box to the 'Git Repository URL' field in the fourth screenshot.
- Screenshot 4:** Shows the 'Build Configuration' and 'Deployment Configuration' sections. A red circle with the number '4' is in the bottom left. A yellow box highlights the 'MAVEN\_MIRROR\_URL' field with the value 'http://repo1sandbox.uhc.com/artifactory/repo', with the text 'Change the MAVEN Repo URL to point to UHG Internal Repo. Else builds will fail if you are doing a JBoss s2i maven build.' Another yellow box highlights the 'Git Repository URL' field with the value 'https://github.com/sreekaimal/hello2.git', with the text 'My code with pom.xml is located in this git repo. This is a simple web app with index.html and snoop.jsp'. A red arrow points from the 'Git Repository URL' field in this screenshot to the 'Git Repository URL' field in the third screenshot.

# Build & Deploy your Application using S2I

The image consists of three screenshots of the OpenShift Web Console, each showing a different step in the application creation and deployment process:

- Screenshot 1: Application created.** Shows the 'sampleapp1' application has been created successfully. A red arrow points from this screen to the 'Continue to overview.' link in Screenshot 2.
- Screenshot 2: Continue to overview.** Shows the 'sampleapp1' application in the 'sk' project. It displays a GitHub webhook trigger message and a pending build. A blue box contains text about clicking 'Continue to Overview' and using the CLI 'oc' command.
- Screenshot 3: Build progress.** Shows the 'Builds' page for 'sampleapp1'. It lists a completed build ('sampleapp1-3') and a pending build ('sampleapp1-1'). A red arrow points from the 'View Log' link in Screenshot 2 to the log viewer in this screenshot. A blue box contains text about viewing the build log and browsing individual resources.

**1. Once the Create command is issued, various configuration items are created inside your environment.**

**2. These configuration items trigger chain of events.**

**3. Build happens inside an available pod, uses spare memory from your quota.**

**4. The logs shown on console are updated in real-time.**

**5. Build do not impact or bring down your existing application. So a re-build would not cause your application to stop.**

**6. If the build is a success, it starts the deployment.**

**7. If deployment is success, it spins up a new pod**

**8. If the new pod started fine, it will scale down the old pod & remap.**

**Click on 'Continue to Overview'**  
Takes you to the build progress page.  
The CLI oc command can also track the build progress.  
Try oc get pods  
To see the build pod doing its work.

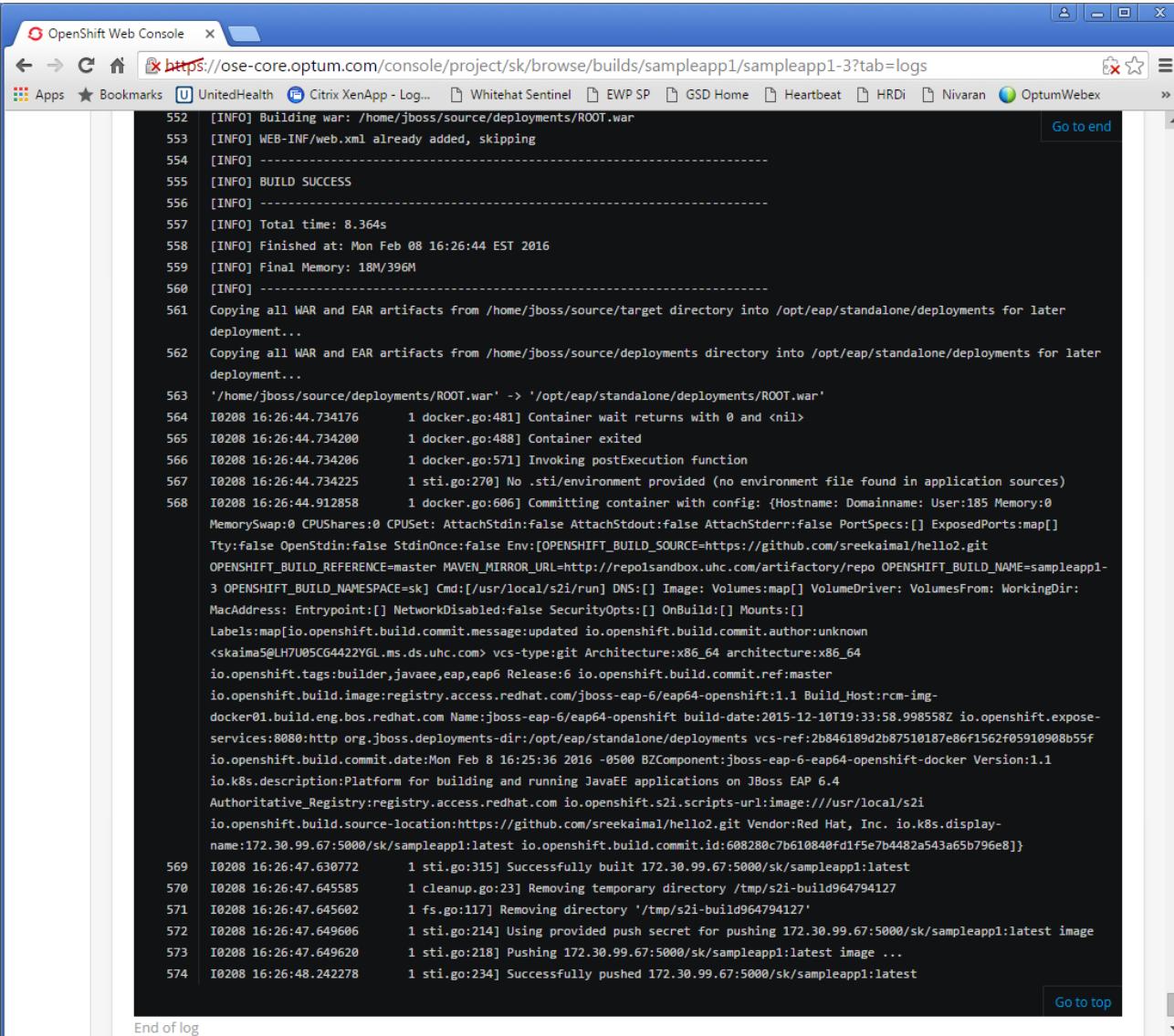
**Click on 'View Log' to see the build Progress.**  
**Click on Browse to see individual resources for this Application**

**The logs keep rolling.**  
At any point if it fails, the status of resources reflect this

# Build & Deploy your Application using S2I

- A successful build does not ensure a successful deployment.
- A deployment would request additional resources from your environment.
- A deployment starts another similar pod before it can bring down the existing pod.
- Lack of resources can cause the sequence of events to halt.
- All events are recorded in the events section. Some events would be very generic, like “*Back off restarting failed docker container*” and some might be specific like “*unable to admit pod without exceeding quota for resource memory*”
- If the container restarts often, it means that the image is not able to sustain within the pod.
- This could be an issue with the code or the server or lack of resources to run.
- By default each container gets 256MB of memory, which might not be enough for e.g., JBoss server. In this case, modifying the deployment configuration would be required.
- Check the settings section to know the current state of your OpenShift account.
- Check pod state & logs for more info..

C:/> oc logs <podname>



The screenshot shows a browser window titled "OpenShift Web Console" displaying the URL <https://ose-core.optum.com/console/project/sk/browse/builds/sampleapp1/sampleapp1-3?tab=logs>. The page content is a log viewer with numerous log entries. The log entries include:

```
552 [INFO] Building war: /home/jboss/source/deployments/ROOT.war
553 [INFO] WEB-INF/web.xml already added, skipping
554 [INFO] -----
555 [INFO] BUILD SUCCESS
556 [INFO] -----
557 [INFO] Total time: 8.364s
558 [INFO] Finished at: Mon Feb 08 16:26:44 EST 2016
559 [INFO] Final Memory: 18M/396M
560 [INFO] -----
561 Copying all WAR and EAR artifacts from /home/jboss/source/target directory into /opt/eap/standalone/deployments for later deployment...
562 Copying all WAR and EAR artifacts from /home/jboss/source/deployments directory into /opt/eap/standalone/deployments for later deployment...
563 '/home/jboss/source/deployments/ROOT.war' -> '/opt/eap/standalone/deployments/ROOT.war'
564 I0208 16:26:44.734176 1 docker.go:481] Container wait returns with 0 and <nil>
565 I0208 16:26:44.734200 1 docker.go:488] Container exited
566 I0208 16:26:44.734206 1 docker.go:571] Invoking postExecution function
567 I0208 16:26:44.734225 1 sti.go:270] No .sti/environment provided (no environment file found in application sources)
568 I0208 16:26:44.912858 1 docker.go:606] Committing container with config: {Hostname: Domainname: User:185 Memory:0 MemorySwap:0 CPUShares:0 CPUSet: AttachStdin:false AttachStdout:false AttachStderr:false PortSpecs:[] ExposedPorts:map[] Tty:false OpenStdin:false StdinOnce:false Env:[OPENSHIFT_BUILD_SOURCE=https://github.com/sreekaimal/hello2.git OPENSHIFT_BUILD_REFERENCE=master MAVEN_MIRROR_URL=http://repolsandbox.uhc.com/artifactory/repo OPENSHIFT_BUILD_NAME=sampleapp1-3 OPENSHIFT_BUILD_NAMESPACE=sk] Cmd:[/usr/local/s2i/run] DNS:[] Image: Volumes:map[] VolumeDriver: VolumesFrom: WorkingDir: MacAddress: Entrypoint:[] NetworkDisabled:false SecurityOpts:[] OnBuild:[] Mounts:[] Labels:map{io.openshift.build.commit.message:updated io.openshift.build.commit.author:unknown <skaima50LH7U05G64422YGL.ms.ds.uhc.com vcs-type:git Architecture:x86_64 architecture:x86_64 io.openshift.tags:builder,javabee,eap,eap6 Release:6 io.openshift.build.commit.ref:master io.openshift.build.image:registry.access.redhat.com/jboss-eap-6/eap64-openshift:1.1 Build Host:rcl-img-docker01.build.eng.bos.redhat.com Name:jboss-eap-6/eap64-openshift build-date:2015-12-10T19:33:58.998558Z io.openshift.expose-services:8080:http org.jboss.deployments-dir:/opt/eap/standalone/deployments vcs-ref:2b846189d2b87510187e86f1562f05910908b55f io.openshift.build.commit.date:Mon Feb 8 16:25:36 2016 -0500 BZComponent:jboss-eap-6-eap64-openshift-docker Version:1.1 io.k8s.description:Platform for building and running JavaEE applications on JBoss EAP 6.4 Authoritative_Registry:registry.access.redhat.com io.openshift.s2i.scripts-url:image:///usr/local/s2i io.openshift.build.source-location:https://github.com/sreekaimal/hello2.git Vendor:Red Hat, Inc. io.k8s.display-name:172.30.99.67:5000/sk/sampleapp1:latest io.openshift.build.commit.id:608280c7b610840fd1f5e7b4482a543a65b796e8}
569 I0208 16:26:47.630772 1 sti.go:315] Successfully built 172.30.99.67:5000/sk/sampleapp1:latest
570 I0208 16:26:47.645585 1 cleanup.go:23] Removing temporary directory '/tmp/s2i-build964794127'
571 I0208 16:26:47.645602 1 fs.go:117] Removing directory '/tmp/s2i-build964794127'
572 I0208 16:26:47.649606 1 sti.go:214] Using provided push secret for pushing 172.30.99.67:5000/sk/sampleapp1:latest image ...
573 I0208 16:26:47.649620 1 sti.go:218] Pushing 172.30.99.67:5000/sk/sampleapp1:latest image ...
574 I0208 16:26:48.242278 1 sti.go:234] Successfully pushed 172.30.99.67:5000/sk/sampleapp1:latest
```



# Edit Deployment Configuration

The screenshot shows the OpenShift Web Console interface. On the left, there's a sidebar with 'Projects' dropdown set to 'sk'. Under 'Deployments', it lists 'sampleapp1' (created 7 hours ago). A context menu is open over 'sampleapp1', with 'Edit (Raw)' highlighted. The main area shows deployment details: Strategy (Rolling), Update Period (1 sec), Interval (1 sec), Timeout (600 sec), Max Unavailable (25%), and Max Surge (25%). Below this is a 'Template' section with a link to its URL.

If the pod logs do not look something like below, it means server did not started successfully -

```
[org.jboss.as] (Controller Boot Thread) JBoss015961:  
Http management interface listening on  
http://0.0.0.0:9990/management  
[org.jboss.as] (Controller Boot Thread) JBoss015951:  
Admin console listening on http://0.0.0.0:9990  
[org.jboss.as] (Controller Boot Thread) JBoss015874:  
JBoss EAP 6.4.4.GA (AS 7.5.4.Final-redhat-4) started in  
12186ms - Started
```

Edit the deployment config to increase the container resource allowance as shown in the screenshot.  
Save and click the deploy button again.  
Multiple deploys exhaust the resources e.g,replication controller.  
Click on Browse->Deployments and next to your deployment, you will see a #X where x is number of deploy attempts. Click on that and delete old attempts.  
Don't delete the deployment configuration itself.

The screenshot shows the 'Editing sampleapp1' dialog. The JSON configuration is displayed, with the 'resources' section highlighted by a red box. It includes 'limits' (cpu: 1024m, memory: 1536Mi) and 'requests' (cpu: 500m, memory: 1536Mi). At the bottom right are 'Save' and 'Cancel' buttons.

```
39 | deploymentconfig: sampleapp1
40 | template:
41 |   metadata:
42 |     creationTimestamp: null
43 |     labels:
44 |       app: sampleapp1
45 |       deploymentconfig: sampleapp1
46 |     spec:
47 |       containers:
48 |
49 |         name: sampleapp1
50 |         image: 172.30.99.67:5000/sk/sampleapp1@sha256:cbd4336cfb53846a787ac68
51 |         ports:
52 |
53 |           containerPort: 8080
54 |           protocol: TCP
55 |
56 |           containerPort: 8443
57 |           protocol: TCP
58 |
59 |         resources:
60 |           limits:
61 |             cpu: 1024m
62 |             memory: 1536Mi
63 |           requests:
64 |             cpu: 500m
65 |             memory: 1536Mi
66 |
67 |         terminationMessagePath: /dev/termination-log
68 |         imagePullPolicy: Always
69 |
70 |       restartPolicy: Always
71 |       terminationGracePeriodSeconds: 30
72 |       dnsPolicy: ClusterFirst
73 |       securityContext: {}
```

If the pod logs do not look something like below, it means server did not started successfully -

# Checking Build , Routes, Deployment & Pods in project

The image contains four screenshots of the OpenShift Web Console interface, each with a yellow callout box and text annotations.

- Builds:** Shows the build history for sampleapp1. It includes a bar chart of build durations (33s, 24s, 16s, 8s) and a table of builds (#1, #2, #3). A callout box states: "Builds track the number of builds, the status and statistics regarding the build. Build Config changes can be made here. Historical build records allow us to rollback to a specific build if the current one does not work fine. It consumes resources. If you don't need rollback, delete old builds."
- Deployments:** Shows the deployment configuration for sampleapp1. It includes fields for Strategy (Rolling), Update Period (1 sec), Interval (1 sec), Timeout (600 sec), Max Unavailable (25%), and Max Surge (25%). A callout box states: "Tracks the deployments, shows configuration details. Edit deployment configuration here."
- Pods:** Shows the pod details for sampleapp1-8-nginx. It includes tabs for Details, Metrics, Logs, and Terminal. A callout box states: "Pods list all the pods in your account. Click on any pod and check the Terminal. You can actually view the contents of your pod. Same as `oc exec -it --pod=<podname>`. Check the logs for your container here."
- Routes:** Shows the route configuration for sampleapp1. It displays the YAML code for the route. A callout box states: "Routes are a way to expose your services externally. External=outside your OSE environment. Not External"

# Testing..

<http://sampleapp1-sk.ose-core.optum.com/>

The screenshot shows the OpenShift Web Console interface. On the left, there's a sidebar with 'Overview', 'Browse', and 'Settings'. The main area shows a project named 'sk'. Under 'DEPLOYMENT: SAMPLEAPP1, #8', it says '1 pod' with a blue circle around it. To the right, there's a 'Details' panel with text about pods, services, and deployments. A red arrow points from the '1 pod' text to the first step in the list below.

1. Check if the pod status is Blue - consistently.
2. Check events to see if there are no container restarts being logged.
3. Check settings to see if there are no warning icons or max limits.
4. Check if logs are being generated in pod.

1. Try modifying the source git repo & hit build again.
2. In 'Overview' See how the build & deploy spins a new pod.
3. Check resource usages during a build.
4. If you are out of resources, then delete a few unused configuration items / failed pods.
5. Resource quota can be increased if required.
6. Always keep the container memory at a moderate level. We don't need to spin up a container with 4GB when we can get optimum performance with 2GB.
7. Remember this is horizontally scalable.

The screenshot shows the deployment details for 'sampleapp1-sk.ose-core.optum.com'. It includes sections for 'S2I Build', 'Setting Resource Limits' (with a code snippet for a Deployment Config), 'Development Resources' (with links to blogs and training), and 'Environment Variables' (with a field for 'MAVEN\_MIRROR\_URL'). A red arrow points from the 'Deployment' section in the first screenshot to the 'S2I Build' section here.

## Next Steps

1. Create a Web Project in JBDS or eclipse.
2. Deploy locally on Jboss EAP 6.4 , test
3. Upload it to codehub
4. Build a S2I package
5. Deploy to OpenShift V3
6. Check what shows in overview when you build & deploy.

Help & Support available via Oneconnect  
<https://oneconnect.uhg.com/groups/openshift-enterprise-users-group>

Support team can be reached at  
[OpenShiftEnterpriseSupport\\_DL@ds.uhc.com](mailto:OpenShiftEnterpriseSupport_DL@ds.uhc.com)





Optum Global Solutions

Thank you.

Sreejith Kaimal  
Sr.Architect  
Provider Portal Applications