OPTUM™

Optum Global Solutions

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

Open Shift Enterprise V3

Day 04

Training Presentation for OpenShift Enterprise

Sreejith Kaimal

February 6, 2016

Templates

A template describes a set of objects that can be parameterized and processed to produce a list of objects for creation by OpenShift. A template will contain

- A set of resources/objects that will be created as part of 'creating/deploying' the template
- A set of values for the parameters defined in the template
- A set of labels to describe the generated resources
- A template will be defined in JSON or YAML format, and will be loaded into OpenShift for application creation.
- The templates can have global visibility scope (visible for every OpenShift project) or project visibility scope (visible only for a specific project).

```yaml
apiVersion: v1
kind: Template
metadata:
  name: templateName
  annotations:
    description: "Description"
    iconClass: "icon-redis"
    tags: "database,nosql"
objects:
- apiVersion: v1
  kind: Pod
  metadata:
    name: redis-master
  spec:
    containers:
    - env:
      - name: REDIS_PASSWORD
        value: ${REDIS_PASSWORD}
      image: dockerfile/redis
      name: master
      ports:
      - containerPort: 6379
        protocol: TCP
parameters:
- description: Password used for Redis Auth
  from: '[A-Z0-9]{8}'
  generate: expression
  name: REDIS_PASSWORD
labels:
  redis: master
```

Template Name, Description, Icon , search tags etc. This is what displays on the console when the template is made available.

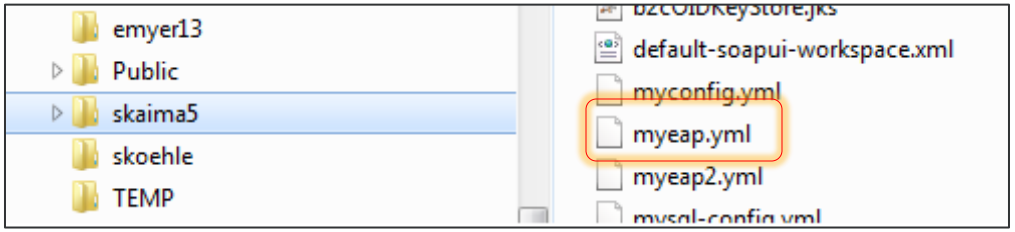
The list of objects the template is expected to create like pod.

Can include ${Parameters} which will be substituted with user provided values while creating the template.

Can include rules on creating the parameters. Like passwords, usernames etc.

The process command can transform a project template into a project configuration file. Try this

```
C:\> oc process openshift//eap64-basic-s2i > myeap.yml
```



Templates

Operation	Syntax	Description
oc create	oc create -f <*file_or_dir_path*>	Parse a configuration file and create one or more OpenShift objects based on the file contents. The -f flag can be passed multiple times with different file or directory paths. When the flag is passed multiple times, oc create iterates through each one, creating the objects described in all of the indicated files. Any existing resources are ignored.
oc update	oc update -f <*file_or_dir_path*>	Attempt to modify an existing object based on the contents of the specified configuration file. The -f flag can be passed multiple times with different file or directory paths. When the flag is passed multiple times, oc update iterates through each one, updating the objects described in all of the indicated files.
oc process	oc process -f <*template_file_path*>	Transform a project template into a project configuration file.
oc get	templates –n openshift	Lists all templates available with OpenShift namespace.

1. Check all the existing templates that you can use-reuse
   - **`oc get template -n openshift`**
2. Process an existing template to create a yml or json output
   - **`oc process openshift//eap64-basic-s2i > myeap.yml`**
   - **`oc process openshift//eap64-basic-s2i --output=yaml > myeapx.yml`**
3. Verify if the yaml file was created.
4. Try running **`oc process –h`** for more help options
5. Update yaml file to project specific settings. Edit the container memory etc.
6. Use create command to create OpenShift objects , the yml file should be in the same local where is cmd is pointing to.
   - `oc create -f myeapx.yml`
7. The yaml then becomes a script for creating standard project specific templates.
8. Use the yaml to do project specific builds.
9. Check on console if the result was successful.

https://docs.openshift.com/enterprise/3.0/dev_guide/templates.html#dev-guide-templates

Templates (eap64-basic-s2i) - example

```
apiVersion: v1
items:
- apiVersion: v1
  kind: Service
  metadata:
    annotations:
      description: The web server's http port.
    labels:
      application: eap-app
      template: eap64-basic-s2i
      xpaas: 1.2.0
    name: eap-app
  spec:
    ports:
    - port: 8080
      targetPort: 8080
    selector:
      deploymentConfig: eap-app
- apiVersion: v1
  id: eap-app-http
  kind: Route
  metadata:
    annotations:
      description: Route for application's http service.
    labels:
      application: eap-app
      template: eap64-basic-s2i
      xpaas: 1.2.0
    name: eap-app
  spec:
    host: ""
    to:
      name: eap-app
- apiVersion: v1
  kind: ImageStream
  metadata:
    labels:
      application: eap-app
      template: eap64-basic-s2i
      xpaas: 1.2.0
    name: eap-app
```

```
- apiVersion: v1
  kind: BuildConfig
  metadata:
    labels:
      application: eap-app
      template: eap64-basic-s2i
      xpaas: 1.2.0
    name: eap-app
  spec:
    output:
      to:
        kind: ImageStreamTag
        name: eap-app:latest
    source:
      contextDir: kitchensink
      git:
        ref:
        uri:
https://codehub.optum.com/osev3hello/eapqs.git
      type: Git
    strategy:
      sourceStrategy:
        forcePull: true
        from:
          kind: ImageStreamTag
          name: jboss-eap64-openshift:1.2
          namespace: openshift
      type: Source
    triggers:
    - github:
        secret: yogduwxA
      type: GitHub
    - generic:
        secret: i66EAtIH
      type: Generic
    - imageChange: {}
      type: ImageChange
    - type: ConfigChange
- apiVersion: v1
  kind: DeploymentConfig
  metadata:
    labels:
      application: eap-app
      template: eap64-basic-s2i
      xpaas: 1.2.0
```

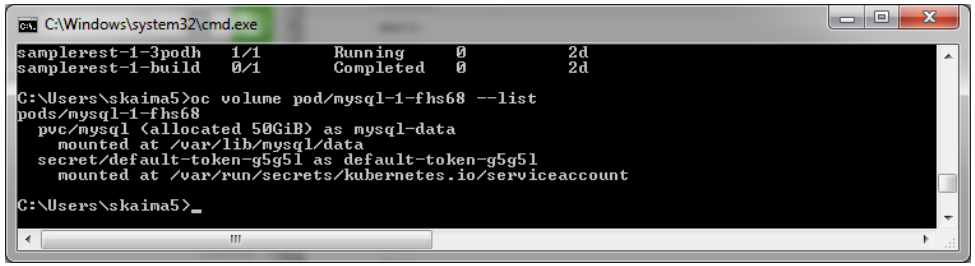
```
name: eap-app
  spec:
    replicas: 1
selector:
      deploymentConfig: eap-app
    strategy:
type: Recreate
    template:
      metadata:
        labels:
          application: eap-app
          deploymentConfig: eap-app
        name: eap-app
      spec:
        containers:
        - env:
          - name: OPENSHIFT_KUBE_PING_LABELS
            value: application=eap-app
          - name: OPENSHIFT_KUBE_PING_NAMESPACE
            valueFrom:
              fieldRef:
                fieldPath: metadata.namespace
          - name: HORNETQ_CLUSTER_PASSWORD
            value: x4hC4I7Q
          - name: HORNETQ_QUEUES
            value: ""
          - name: HORNETQ_TOPICS
            value: ""
          - name: JGROUPS_CLUSTER_PASSWORD
            value: 55NtnAPv
          image: eap-app
          imagePullPolicy: Always
          livenessProbe:
            exec:
              command:
              - /bin/bash
              - -c
              - /opt/eap/bin/livenessProbe.sh
          name: eap-app
          ports:
          - containerPort: 8778
            name: jolokia
            protocol: TCP
          - containerPort: 8080
            name: http
            protocol: TCP
…
```

Volume Claims

- Containers are not persistent by default; on restart, their contents are cleared.
- Volumes are mounted file systems available to pods and their containers which may be backed by a number of host-local or network attached storage endpoints.
- A PersistentVolume object is a storage resource in an OpenShift Enterprise cluster. Storage is provisioned by your cluster administrator by creating PersistentVolume objects from sources such as NFS mounts.
- A **PersistentVolume** is a specific resource. A **PersistentVolumeClaim** is a request for a resource with specific attributes, such as storage size. In between the two is a process that matches a claim to an available volume and binds them together. This allows the claim to be used as a volume in a pod. OSE finds the volume backing the claim and mounts it into the pod.
- You can use the CLI command oc volume to add, update, or remove volumes and volume mounts for any object that has a pod template like replication controllers or deployment configurations.

oc volume <object_selection> <operation> <mandatory_parameters> <optional_parameters>

oc volume <object_type>/<name> --add [options]	To add a volume to an existing object, e.g., pod. oc volume rc/r1 --add --name=v1 --type=secret --secret-name='$ecret' --mount-path=/data
oc volume <object_type>/<name> --remove [options]	Remove/Unmount the volume from the specified objects oc volume dc/d1 --remove --name=v1 --containers=c1
oc volume pod/p1 –list	List all volumes that is mounted to the pod type object with name p1
oc volume dc --all --name=v1	List volumes on all deployment configs.



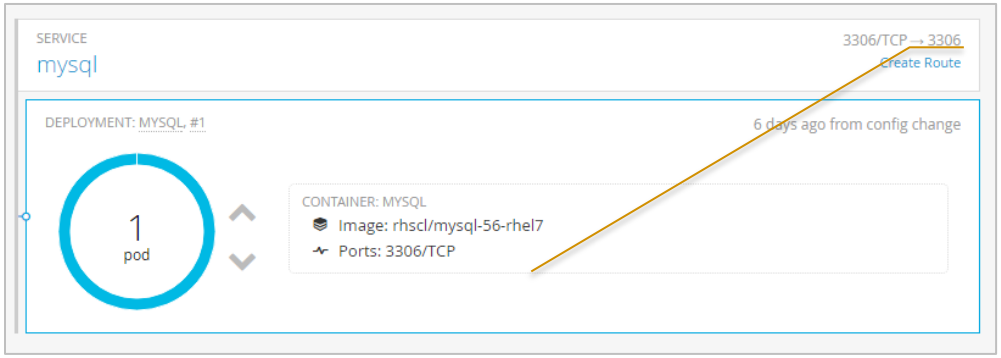
Access Mode	Description
ReadWriteOnce RWO	The volume can be mounted as read-write by a single node.
ReadOnlyMany ROX	The volume can be mounted read-only by many nodes.
ReadWriteMany RWX	The volume can be mounted as read-write by many nodes

https://docs.openshift.com/enterprise/3.2/dev_guide/volumes.html

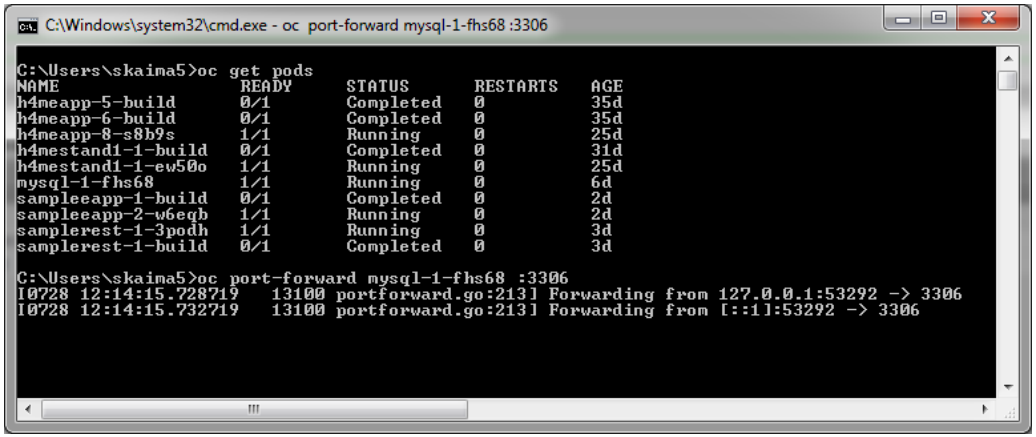
Port Forwarding

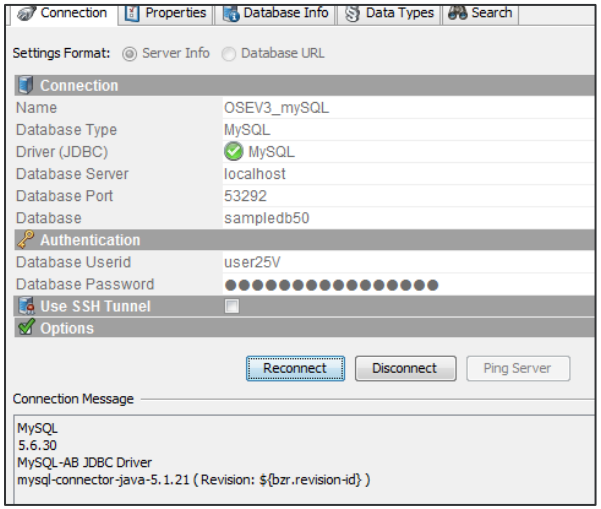
How do I connect to a database running on OSEv3 from my local

You can use the CLI to forward one or more local ports to a pod. This allows you to listen on a given or random port locally, and have data forwarded to and from given ports in the pod.



This is a MySQL Persistent DB running on OSE.
This exposes 3306 port for apps running within OSE.
This is not accessible from your local machine.
Create a port-forward from your local machine to pod.
Use that local port to get access to your pod DB .
This works only as long as oc cmd is running
This is only for local, for OSE-OSE it should work on existing ports specified.





Templates

Hands On 1

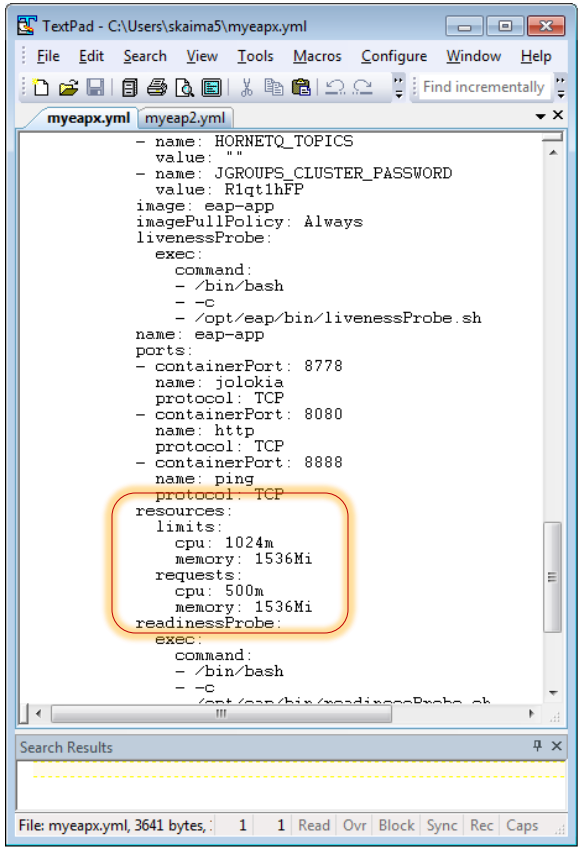
Create a default template for a tomcat app

1. Create a template file on your local.
2. Use the process command to get the yaml file (use yaml directive)
3. Edit the yaml file using your text editor. Keep an eye on indentation if its yaml.
4. Change the deployment section to have more resource allocations.
5. Edit application & label names.
6. Add / customize additional parameters for your template to run without defaults.
7. Execute the yaml file to get the environment setup.
8. Verify on console.

Hands On 2

Create a MySQL Persistent storage DB

1. Use the console to locate the MySQL Persistent template.
2. Complete the template parameters like username/dbname etc.
3. Use the volume capacity (1Gi) as input.
4. Create the MySQL app.
5. Tail logs to see if the server started & listening on 3306
6. Try connecting to the DB from your local. Use service URL to connect.
7. Create a port forward from your local and connect to the DB.



Customize Assemble Scripts

Source-to-Image (S2I) is a 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. The new image incorporates the base image (the builder) and built source and is ready to use with the docker run command. S2I supports incremental builds, which re-use previously downloaded dependencies, previously built artifacts, etc.

The assemble script can also be part of your GIT repository. If the build encounters the assemble inside /.sti/bin folder in your source code. It would use that assemble script instead of the existing S2I build script.

The S2I script prepares the container by creating folders / setting path and required resources. The STI assemble can be used to customize and initialize application resources as well. App startup routines should not be part of STI. It should be initialized via web.xml or likewise.

branch: master — h4meapiservices / .sti / bin

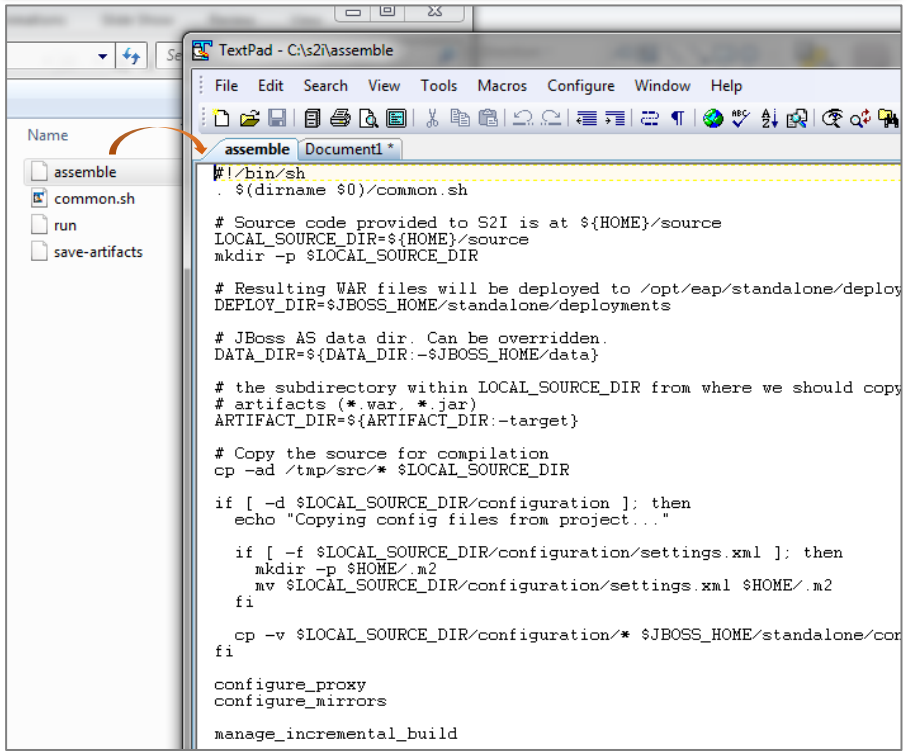
S2I Try10

Kaimal authored 29 days ago

..	
assemble	S2I Try10
common.sh	STI Assemble try3
monitoring-install.sh	STI Path test2
run	STI Assemble try3
save-artifacts	STI Assemble try3

Download the S2I scripts to your local machine. This needs an existing application.

oc rsync apiservices-8-81u4s:/usr/local/s2i C:/




```
12  I0728 14:18:17.928001   1 docker.go:344] Image contains io.openshift.s2i.scripts-url set to "image:///usr/local/s2i"
13  I0728 14:18:17.928024   1 download.go:57] Using image internal scripts from: image:///usr/local/s2i/save-artifacts
14  I0728 14:18:17.928085   1 sti.go:221] Using assemble from upload/src/.sti/bin
15  I0728 14:18:17.928106   1 sti.go:221] Using run from upload/src/.sti/bin
16  I0728 14:18:17.928112   1 sti.go:221] Using save-artifacts from upload/src/.sti/bin
17  I0728 14:18:17.928133   1 sti.go:148] Clean build will be performed
18  I0728 14:18:17.928138   1 sti.go:151] Performing source build from file:///tmp/s2i-build776325274/upload/src#master
19  I0728 14:18:17.928143   1 sti.go:164] Running "assemble" in "172.30.51.116:5000/h4me-stg/apiservices:latest"
```

https://docs.openshift.com/enterprise/3.2/dev_guide/builds.html#dev-guide-builds

Opening a remote shell to your container

- The oc rsh command allows you to locally access and manage tools that are on the system.
- While in the remote shell, you can issue commands as if you are inside the container and perform local operations like monitoring, debugging, and using CLI commands specific to what is running in the container.

```
C:\Windows\system32\cmd.exe - oc  rsh mysql-1-fhs68
h4mestand1-1-ew50o    1/1       Running     0          25d
mysql-1-fhs68         1/1       Running     0          6d
sampleeapp-1-build    0/1       Completed   0          2d
sampleeapp-2-w6eqb    1/1       Running     0          2d
samplerest-1-3podh    1/1       Running     0          3d
samplerest-1-build    0/1       Completed   0          3d

C:\Users\skaima5>oc rsh mysql-1-fhs68
bash-4.2$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.30 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

- oc exec can be used to execute a command remotely. However, the oc rsh command provides an easier way to keep a remote shell open persistently.
- The session would be terminated if inactive.
- Your changes are not persistent. If you make changes directly within the container and that container is destroyed and rebuilt, your changes will no longer exist.
- rsh will work only if the pod is in Running state with no issues reported.
- If rsh does not work, try the oc describe to see the current state of the pod.

Service Accounts

- Service accounts provide a flexible way to control API access without sharing a regular user's credentials. Service accounts are API objects that exist within each project.
- They can be created or deleted like any other API object. It is recommended to use service account for setting up build & deploy.
- Every service account has an associated user name that can be granted roles, just like a regular user. The user name is derived from its project and name
- Create a service account using the json file in the example

C:\Windows\system32\cmd.exe

```
C:\Users\skaima5>oc create -f service.json
serviceaccount "jrobot" created

C:\Users\skaima5>oc describe serviceaccount
Name:           builder
Namespace:      coectccr
Labels:         <none>

Image pull secrets:     builder-dockercfg-itkwc

Mountable secrets:      builder-token-z9yvk
                        builder-dockercfg-itkwc

Tokens:                 builder-token-icizf
                        builder-token-z9yvk


Name:           default
Namespace:      coectccr
```

```
{
  "apiVersion": "v1",
  "kind": "ServiceAccount",
  "metadata": {
    "name": "jrobot"
  }
}
```

Every service account is also a member of two groups:

1. **system:serviceaccounts**, which includes all service accounts in the system
2. **system:serviceaccounts**:<project>, which includes all service accounts in the specified project

C:\Windows\system32\cmd.exe

```
C:\Users\skaima5>oc describe secret jrobot-token-2vegt -n coectccr
Name:           jrobot-token-2vegt
Namespace:      coectccr
Labels:         <none>
Annotations:    kubernetes.io/service-account.name=jrobot,kubernetes.io/service-account.uid=5647fa

Type:   kubernetes.io/service-account-token

Data
====
ca.crt: 1066 bytes
token:  eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZ
vc2VjcmV0Lm5hbWUiOiJqcm9ib3QtdG9rZW4tMnZlZ3QiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtY
0N2ZhMjUtNTRmZi0xMWU2LWE4YzItMDA1MDU2ODcwNGRkIiwic3UiIjoic3ldGVtOnNlcnZpY2VhY2NvdW500mNvZWN0Y2NyO
Yoy8T_2VJRczUz7e39jsB1AV-z91UUVhuWId6r8XqdW6FU88pp8YX4taelnQBg8dXQaOW99rsV71Pv-DUNdwXwjPJCczI3zi24
Kzqo4SmPQsMwG6bEHR4_W7eEsDnQk3UkiK6YG2AaIrhQUtuW0I8w


C:\Users\skaima5>oc describe secret jrobot-token-2vegt -n coectccr > secret.txt

C:\Users\skaima5>oc logout
Logged "cloudcoe" out on "https://ose-ctc-core.optum.com:443"

C:\Users\skaima5>
```

1. Store your secret token into a text file.
2. Create a batch file or a command to use the token instead of your MS\userid and password.
3. Use the token as a parameter while doing oc login

TextPad - C:\Users\skaima5\token.bat

File Edit Search View Tools Macros Configure Window Help

Find incrementally

token.bat | secret.txt

```
oc login --token=
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJrdWJlcm5
ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWN
lYWNjb3VudC9uYW1lc3BhY2UiOiJjb2VjdGNjciIsImt1YmVybmV0ZXM
uaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJqcm9ib3QtdG9
rZW4tMnZlZ3QiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3N
```

1. Recommended approach is to use **token** for jenkins and build automation, it resolves the username and password prompt and gives a one click login.
2. Its easy to manage secrets and service accounts and grant specific permissions so that audit is easy.
3. The tokens are non-expiring and would not cause id's to be locked out.
4. These tokens can also be used for externalizing the build & deploy lifecycle.

Build from Private Repository

How to build and deploy from a private repository.

- Basic authentication requires either a combination of username and password, or a token to authenticate against the SCM server.
- There are these secrets that can be created in the environment for authentication
  - new : Create a new secret based on a key file or on files within a directory
  - new-dockercfg : Create a new dockercfg secret
  - new-basicauth : Create a new secret for basic authentication
  - new-sshauth : Create a new secret for SSH authentication
  - add : Add secrets to a ServiceAccount

```
{
  "apiVersion": "v1",
  "kind": "Secret",
  "metadata": {
    "name": "gitcoesecret"
  },
  "namespace": "coectccr",
  "data": {
    "username": "cloudcoe",
    "password": "MyPass123="
    }
}
```

C:\Windows\system32\cmd.exe

```
C:\Users\skaima5>oc secrets new-basicauth gitcoesecret --username=cloudcoe --password=MyPass1123x
secret/gitcoesecret

C:\Users\skaima5>
```

- You can also add the secret using the json specification
- Add the secret to the builder service account. Each build is run with serviceaccount/builder role, so you need to give it access your secret

```
oc secrets add serviceaccount/builder secret/gitcoesecret
```

- Add a sourceSecret field into the source section inside the BuildConfig and set it to the name of the secret that you created

Editing sampleex2p

```
26      imageChange:
27        lastTriggeredImageID: 'registry.access.redhat.com/jboss-eap-6/eap64-openshift:1.2'
28    -
29      type: ConfigChange
30  source:
31    type: Git
32    git:
33      uri: 'https://codehub.optum.com/osev3hello/samplepriv.git'
34      ref: master
35    sourceSecret:
36      name: "gitcoesecret"
37  strategy:
38    type: Source
```

Uses the basic secret that was created in the environment.
This is not available via console on OSE3.0
This has to be done via CLI.

Terminal Access / Troubleshooting & Debugging

oc get <object_type> [<object_name>] *oc get pods*	Return a list of objects for the specified object type. E.g. to get all pods in your project.
oc describe <object_type> <object_name> *oc describe svc*	Returns information about the specific object returned by the query. Useful to get extended details of any specific object. Especially pods status.
oc env <object_type>/<object_name> <var_name>=<value> *oc env bc/bcapp-name -e MAVEN_MIRROR_URL=http://repo1sandbox.uhc.com/artifactory/repo*	Update the desired object type with a new environment variable:
oc label <object_type> <object_name> <label>	Update a specific label on the given object.
oc expose <object_type> <object_name>	Look up a service and expose it as a route. There is also the ability to expose a deployment configuration, replication controller, service, or pod as a new service on a specified port.
oc delete <object_type> <object_name> *oc delete pod mysql-1-fhs68*	Delete the specified object. An object configuration can also be passed in through STDIN as a file.
oc start-build <buildconfig_name>	Start a build from your existing build configuration.
oc deploy <deploymentconfig>	Start a deployment using a deployment config.
oc scale <object_type> <object_name> --replicas=<#_of_replicas>	Set the number of desired replicas for a replication controller or a deployment configuration to the number of specified replicas: Equivalent to clicking the scale icon on the console.
oc logs -f <pod>	Retrieve the log output for a specific build, deployment, or pod. This command works for builds, build configurations etc..
oc rsh <pod>	Open a remote shell session to a container:
oc rsync <local_dir> <pod>:<pod_dir> -c <container> *oc rsync apiservices-8-81u4s:/usr/local/s2i C:/*	Copy contents of local directory to a directory in an already-running pod container. It will default to the first container if none is specified.

More Hands On..

1. The same steps work for any spring / hibernate / jsf web-app.
2. Try adding a Data Source in your web-app.
3. Try consuming a web-service & build a rest service around it.
4. Here are some useful links
   - JBoss Tools - Evaluate OpenShift 3 with the IDE
   - JBoss Tools - Create and Build OpenShift 3 Applications in the IDE
   - JBoss Tools - Set up and Remotely Monitor an OpenShift Database Server
   - JBoss Tools - Deploy a Docker Image to OpenShift 3

OPTUM™

Optum Global Solutions

Thank you.

Sreejith Kaimal
Sr.Architect
Cloud Migration COE