OpenShift@Ops

http://bit.ly/testdrive-openshift-ops

Basic Concepts

Test Drive: OpenShift@Ops

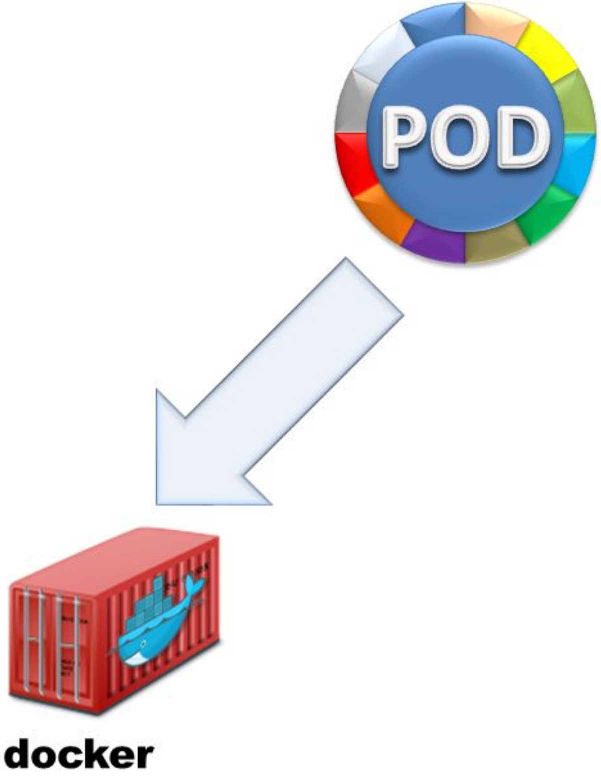
Basic Concepts

The basic element on OpenShift Container Platform, it's **the Pod**.

Two things must be remembered about the Pod:

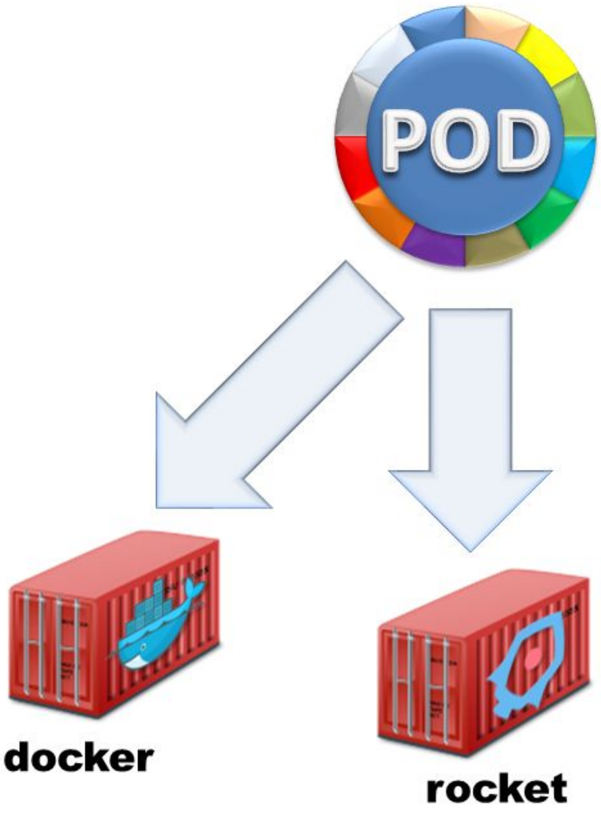
1. **A Pod is an abstraction of a Container**
2. **Pod has resources associated.**

Basic Concepts



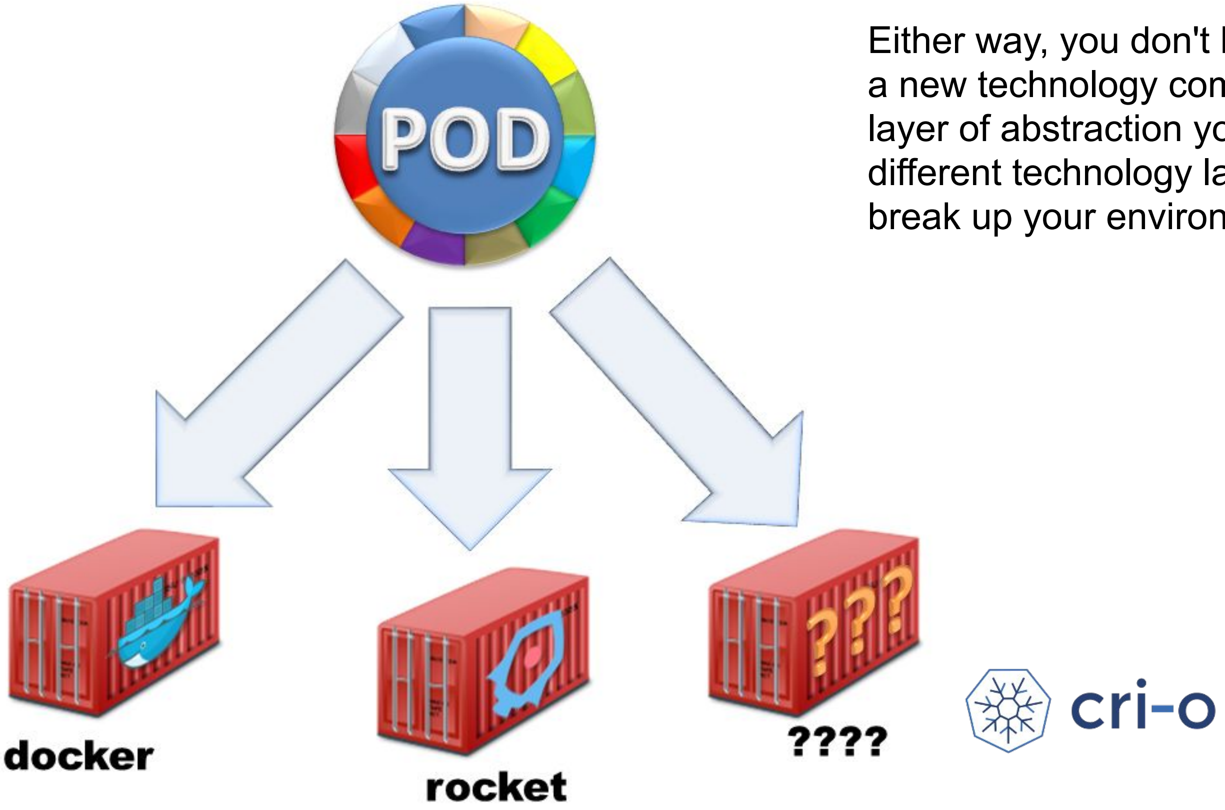
Currently, we're using Pod as an abstraction of Docker containers.

Basic Concepts



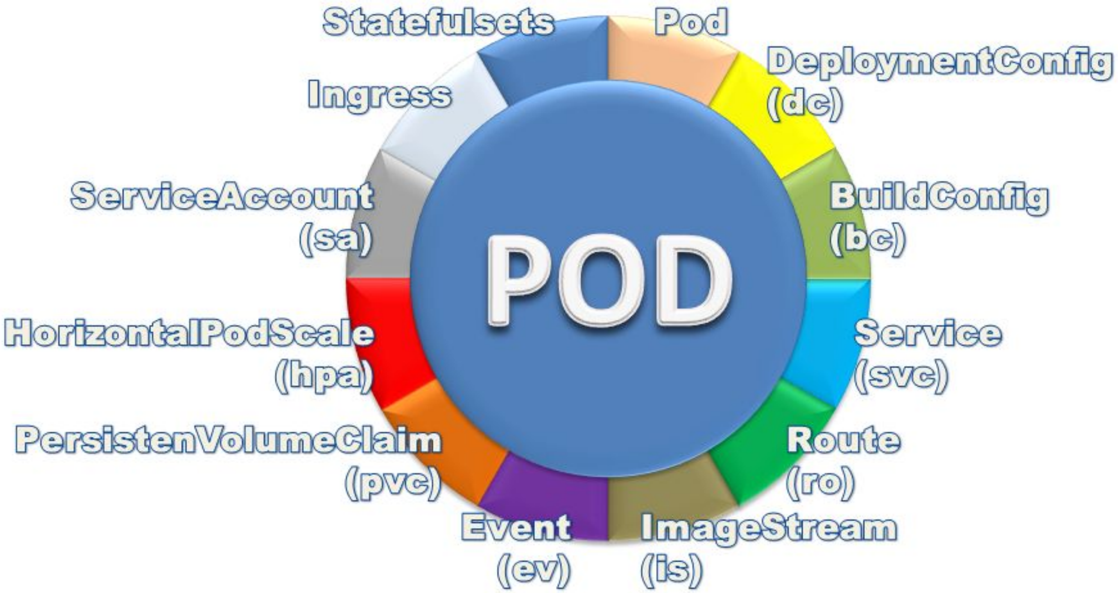
Eventually, you might have a choice to switch to a different Container Technology like Rocket.

Basic Concepts



Either way, you don't have to worry tomorrow a new technology comes up. By handling this layer of abstraction you can switch to a different technology layer without concern of break up your environment.

Basic Concepts



Here are some examples of Resources available for the Pod, each responsible for dealing with in a given moment.

Basic Concepts

```
apiVersion: v1
kind: Pod
metadata:
  name: appserver
spec:
  containers:
  - image: jboss/wildfly:latest
    imagePullPolicy: IfNotPresent
    name: appserver
    ports:
    - containerPort: 8080
      protocol: TCP
  dnsPolicy: ClusterFirst
  restartPolicy: Always
```

A very simple Resource available is called: Pod (also the name of a Resource).

This Resource will pull a container named "jboss/wildfly" and make it available on port 8080.

Please note other information like "restartPolicy"

Basic Concepts

```
apiVersion: v1
kind: DeploymentConfig
metadata:
  name: appserver
  labels:
    name: appserver
spec:
  replicas: 1
  selector:
    app: appserver
    deploymentconfig: appserver
  template:
    metadata:
      labels:
        app: appserver
        deploymentconfig: appserver
    spec:
      containers:
      - image: jboss/wildfly:latest
        name: appserver
        ports:
        - containerPort: 8080
          protocol: TCP
```

DeploymentConfig is another example of a Resource is very frequently used.

It's responsible for pulling a container and it's responsible in deal with changes of versions of the same container.

For example: What happens when we face a new version of JBoss ? How are we going to change the old for a new one ?

Notice a property named "**replicas**" which says the number of instances this container must have at all times.

Basic Concepts

```
apiVersion: v1
kind: Service
metadata:
  labels:
    name: appserver
  name: appserver
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    app: appserver
    deploymentconfig: appserver
  sessionAffinity: None
  type: ClusterIP
```

Service Resource is another very important one and very common often found.

It provides a Pod with a new IP and a hostname, that it can be accessed through the entire OpenShift's Cluster.

Basic Concepts

```
apiVersion: v1
kind: Route
metadata:
  labels:
    name: appserver
  name: appserver
spec:
  host:
myserver2.cloudapps.testdrive.com
  port:
    targetPort: 8080
  to:
    kind: Service
    name: appserver
    weight: 100
```

Route Resource is always important, specially if you're looking to expose your Pod to the outside world as an Web application.

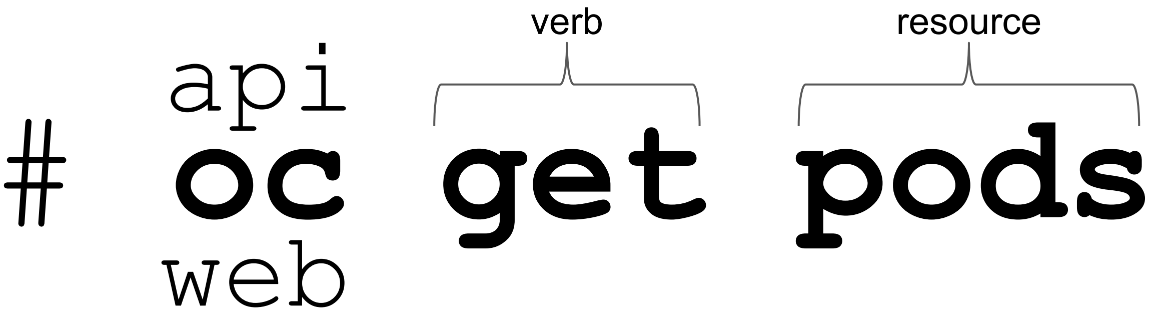
This resource binds the information provided by the Service Resource and offers an external way to get access through a valid **http/https** address.

In our example, this Pod can be accessed by typing:

`http://myserver2.cloudapps.testdrive.com`

The https protocol along with a valid certificate is also available.

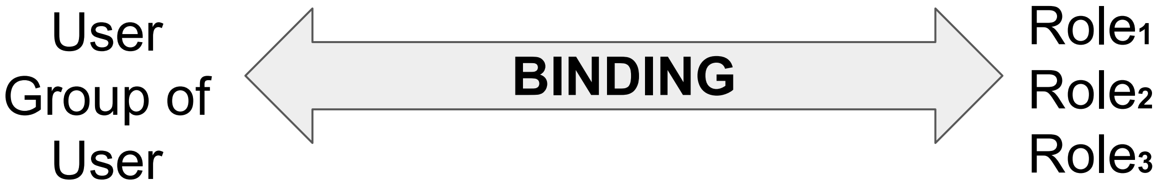
Basic Concepts



```
# oc get pods
```

Rule = Verb + Resource

Role = Rule$_1$ + Rule$_2$ + Rule$_3$ + ...



Basic Concepts

```
NAME        READY STATUS    RESTARTS AGE
jboss-1-xyz 1/1   Running   0        23m
```

Basic Concepts: Logical Perspective

Once a Pod concept is understood, we must find a way to group together several Pods in order to offer a concise solution.

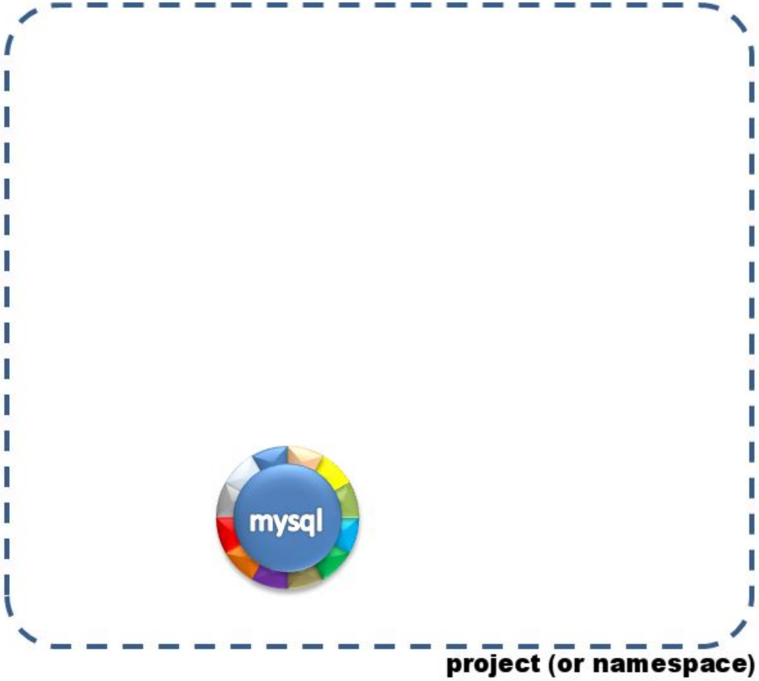
Basic Concepts: Logical Perspective

We do that by group in a concept called **Project** (or **Namespace**).

project (or namespace)

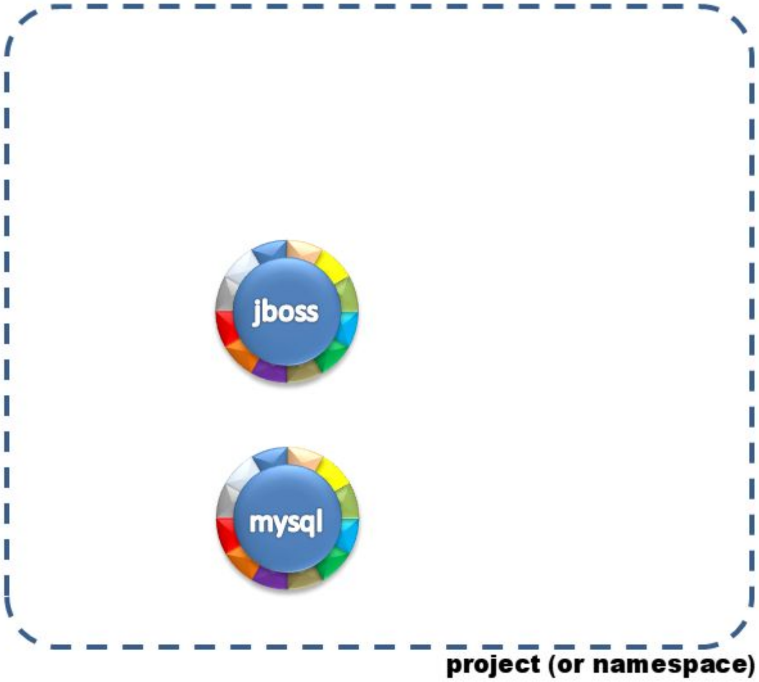
Basic Concepts: Logical Perspective

Inside our Project, we can add a Database Pod (for example) like MySQL.

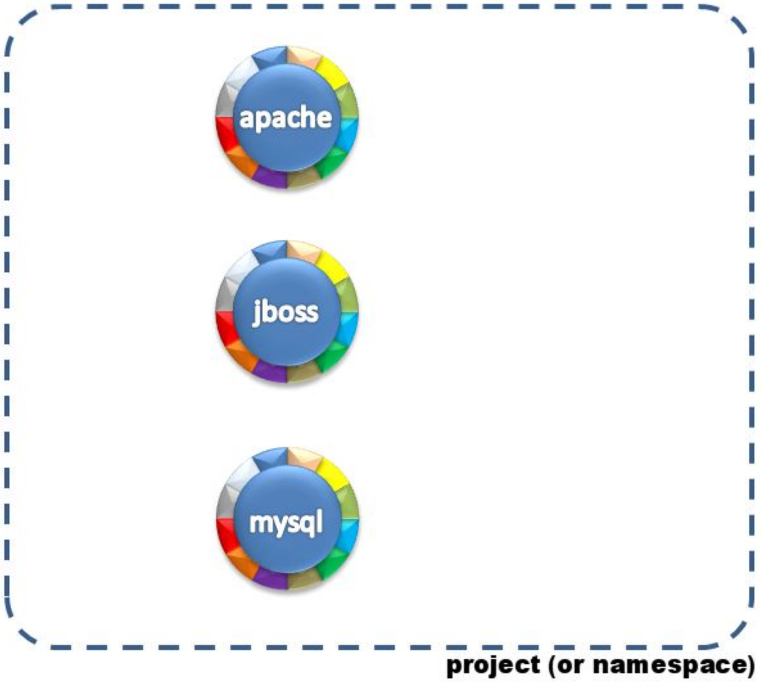


Basic Concepts: Logical Perspective

Then add a second Pod like a JBoss Application Server.



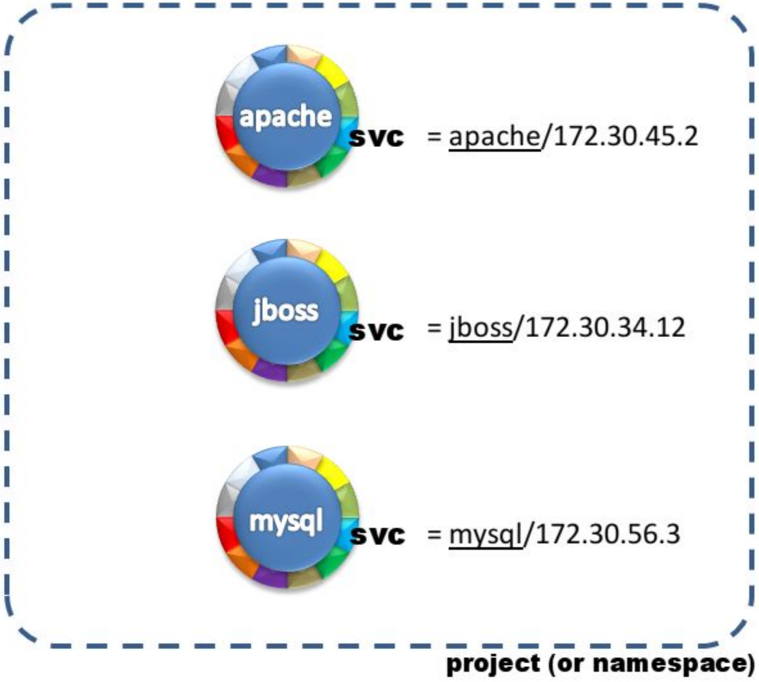
Basic Concepts: Logical Perspective



And finally, a third Pod for handling static content like a Apache.

There is no limit in how many Pods a Project can handle. Our stack can be as big as we might want it.

Basic Concepts: Logical Perspective

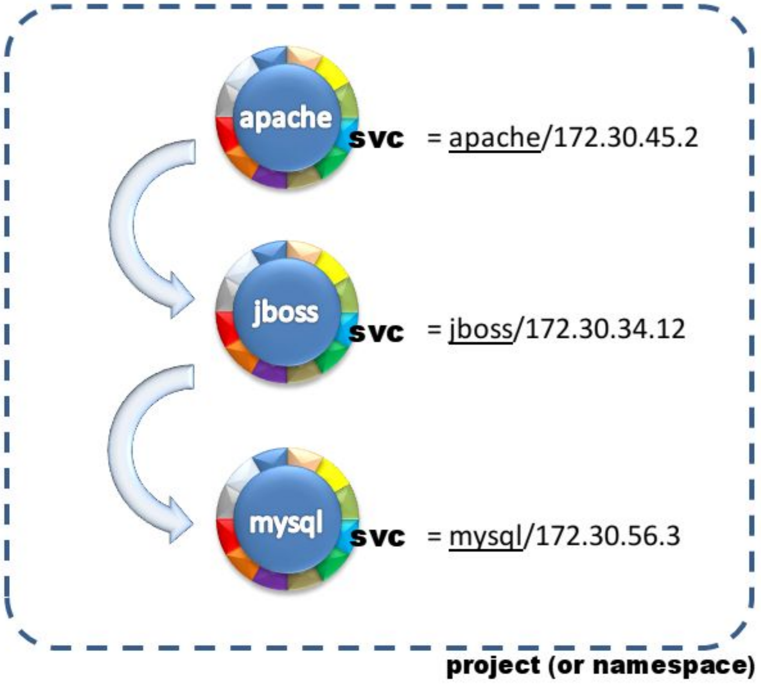


However, our Pods are working pretty much isolated. In order to each other be able to communicate, we need to add a Service Resource in each Pod.

OpenShift will assign a valid IP and a hostname that be accessed through the entire Cluster.

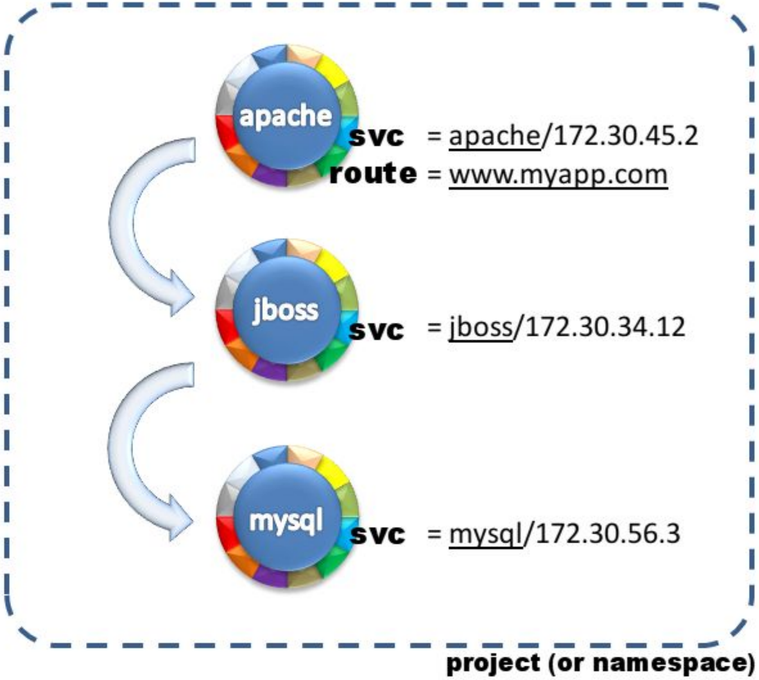
(it's possible to isolate the communication of each Pod only inside the Project. In order to do that, we need to perform a Multi Tenant installation).

Basic Concepts: Logical Perspective



Now, JBoss can connect to a Database named "mysql" a fetch all the necessary tables.

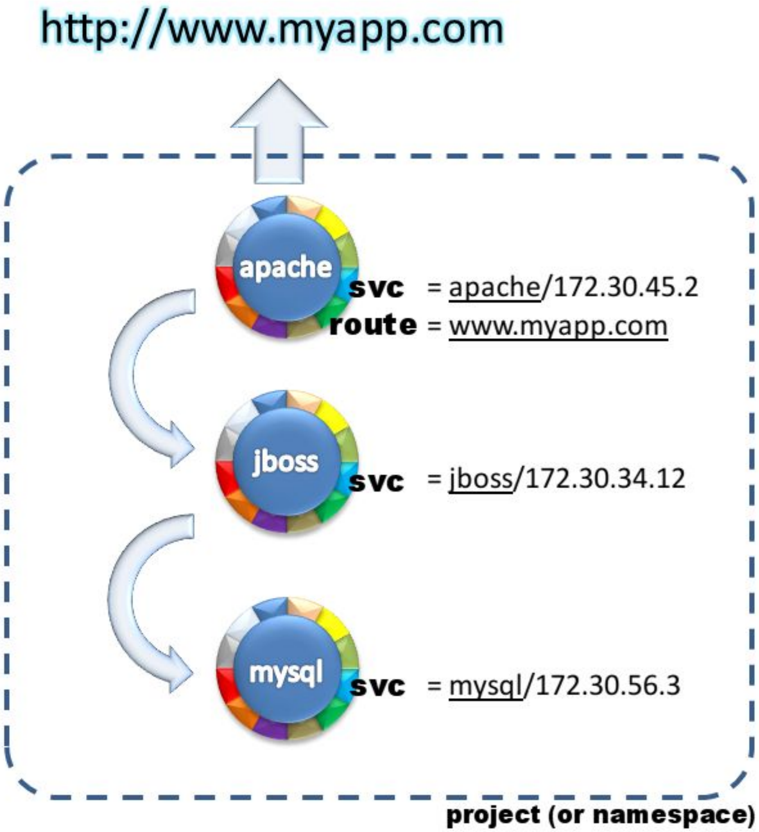
Basic Concepts: Logical Perspective



Finally, we need to expose one of the Pods to the outside world.

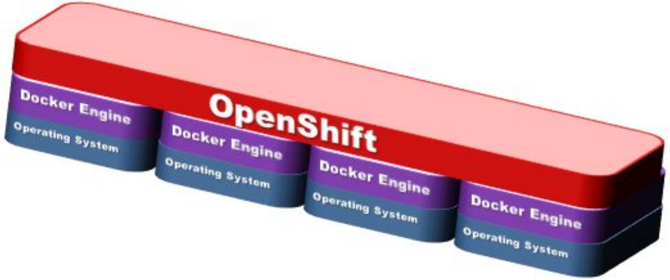
We do this by creating another resource called "Route", which gives an valid URL to our Pod.

Basic Concepts: Logical Perspective



Our solution is ready to be accessed by a WebBrowser by typing the URL indicated.

Basic Concepts: Mechanics

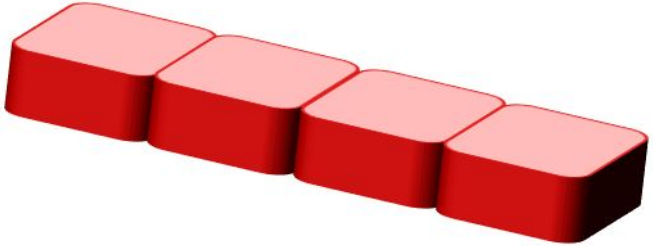


OpenShift creates another layer on top of several hosts that has some kind of Container technology.

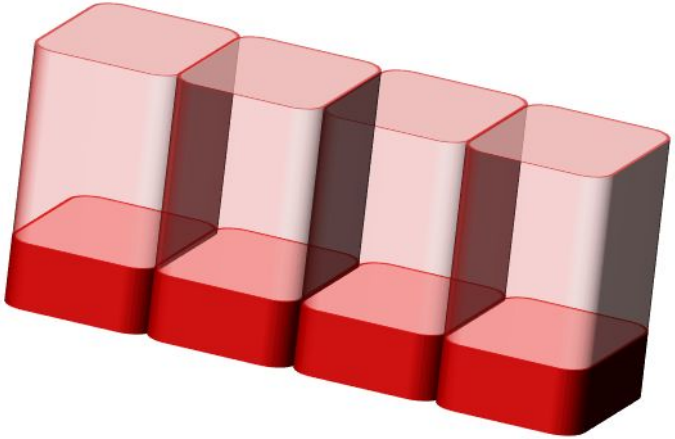
In our example, we're using Docker Container Technology.

Basic Concepts: Mechanics

By adding new new layer, we're going to turn each host and represent as a red box for simplification purposes.

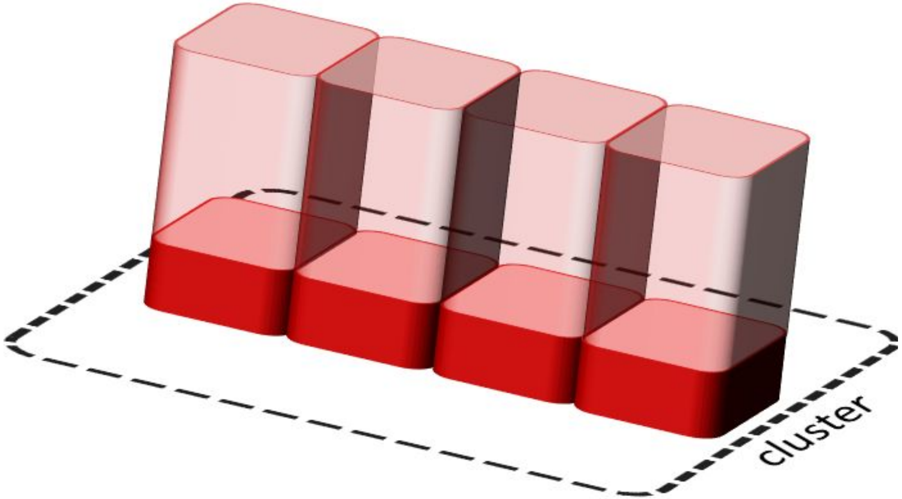


Basic Concepts: Mechanics



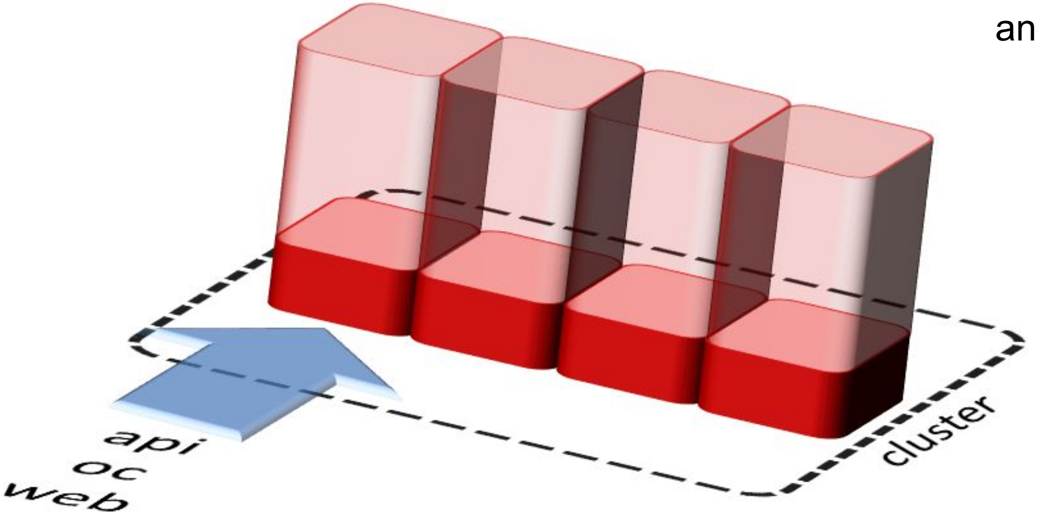
Once each host is ready to run Docker Containers, it means we're ready to host Pods in each of those hosts.

Basic Concepts: Mechanics



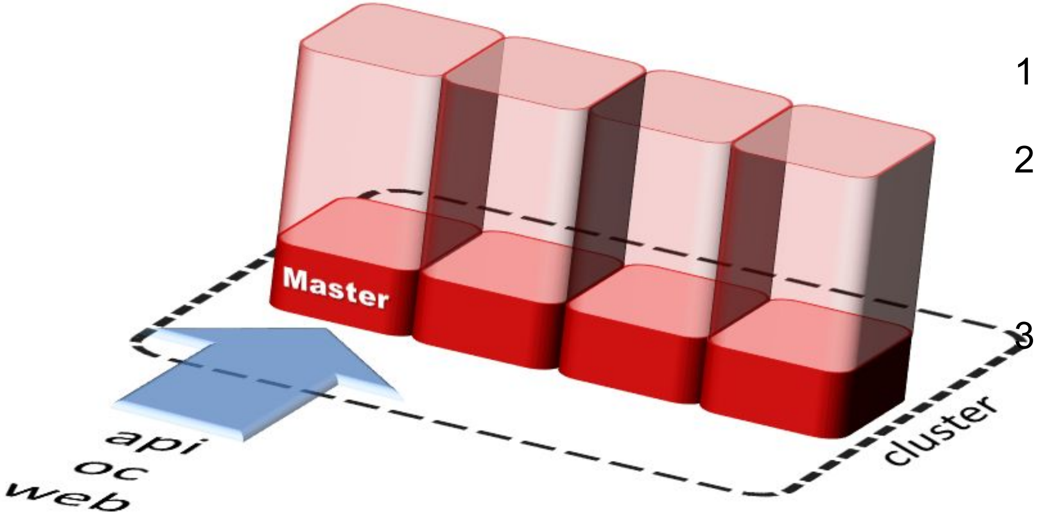
The set of those hosts we're going to call:
Cluster

Basic Concepts: Mechanics



In one of those hosts within the Cluster, I'm going to talk to someone who has the "brains" and can tell me where are the resources, how many Pods I can create and so on.

Basic Concepts: Mechanics

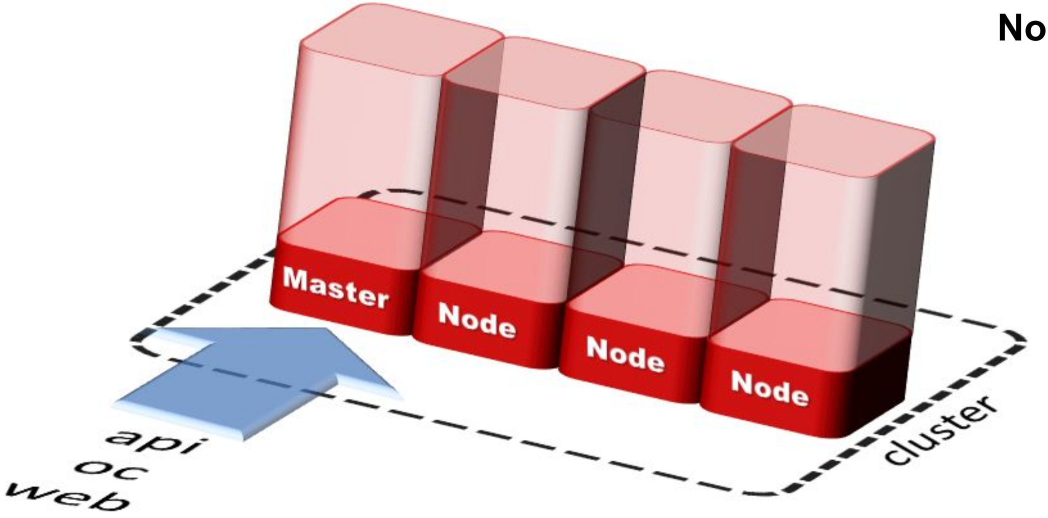


This special host is called: **The Master**

We can use 3 different ways to communicate with the Master:

1. Using an **API calls** using a valid and sign certificate.
2. Using an **Client Application** available for major Operating Systems (named OpenShift Client or OC).
3. Using a **Web Console** which give us a Visual Way to handle the Pods.

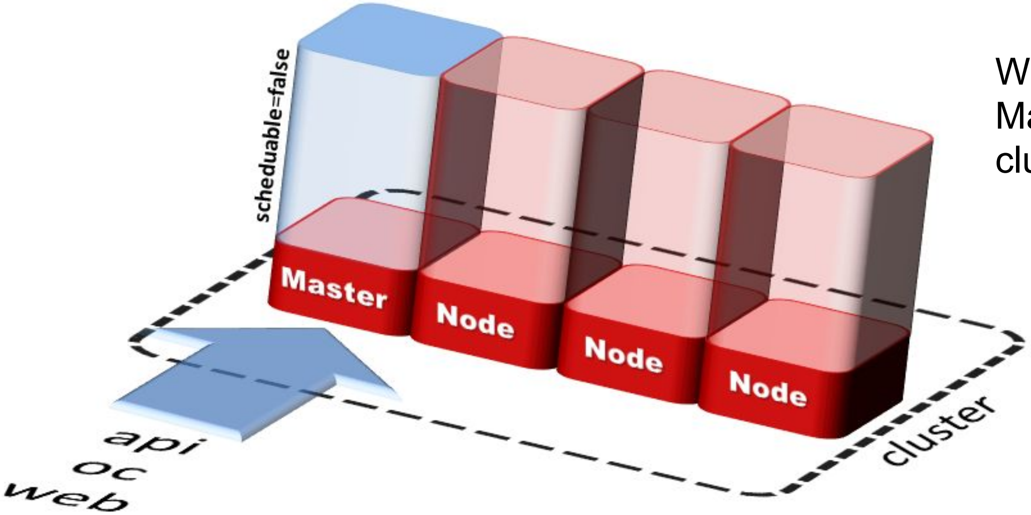
Basic Concepts: Mechanics



Every remaining host available will hold our Pods.

Those hosts are called:
Nodes

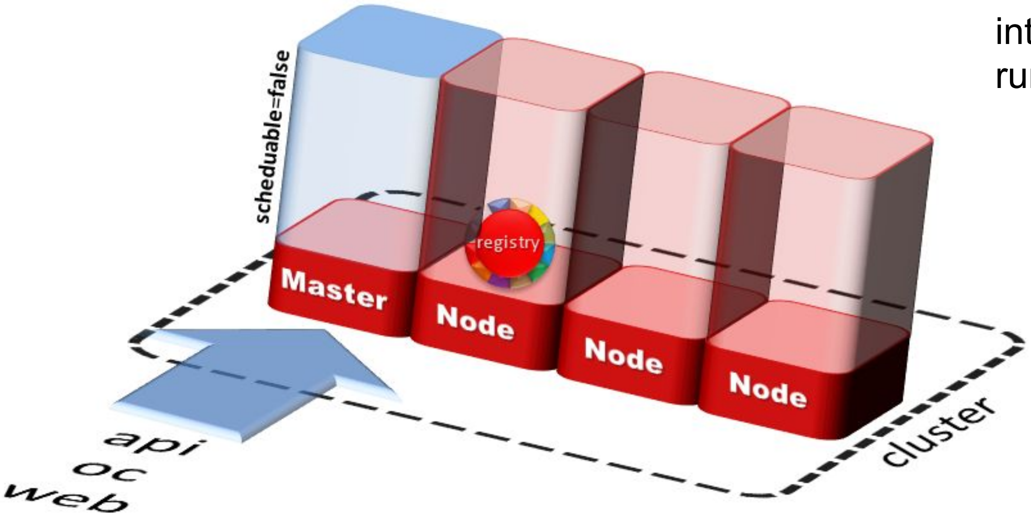
Basic Concepts: Mechanics



Although is possible to have Pods running in the Master, we need to avoid that in order to provide the Master with the maximum performance possible.

We don't want anything interfere with the Master's capability in managing the entire cluster.

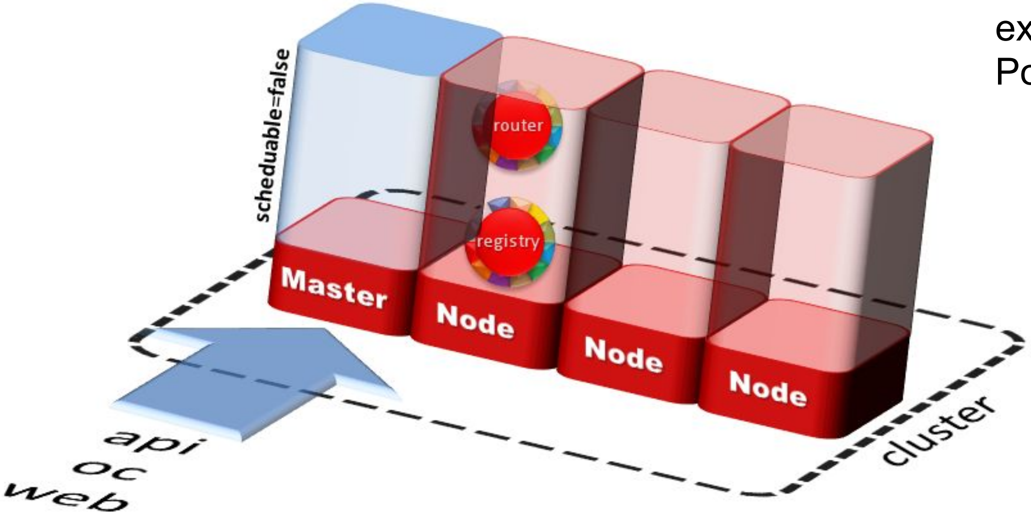
Basic Concepts: Mechanics



One very important Pod that comes in a OpenShift's installation it's the **Registry**.

The Registry is responsible to serve internal requests to get a Pod up and running within the Cluster.

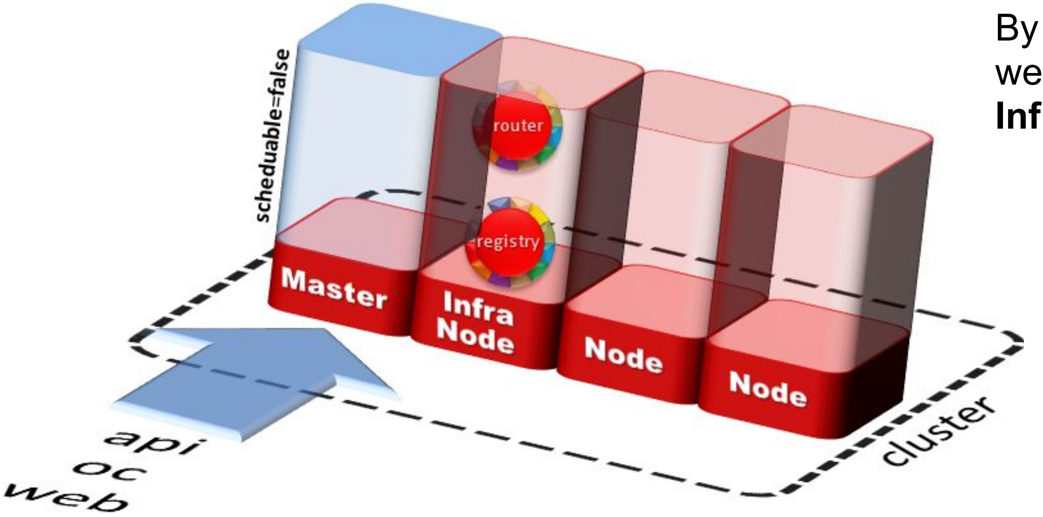
Basic Concepts: Mechanics



Another very important Pod that comes with OpenShift is the **Router**.

The Router is responsible for handling external requests and forward to a specific Pod.

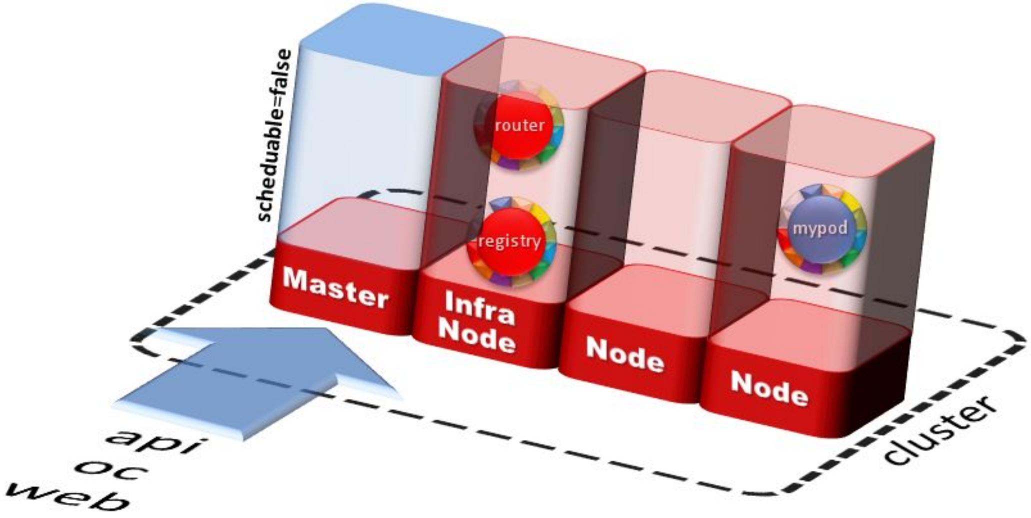
Basic Concepts: Mechanics



We usually keep both of those important Pods (among others) located in a single Host.

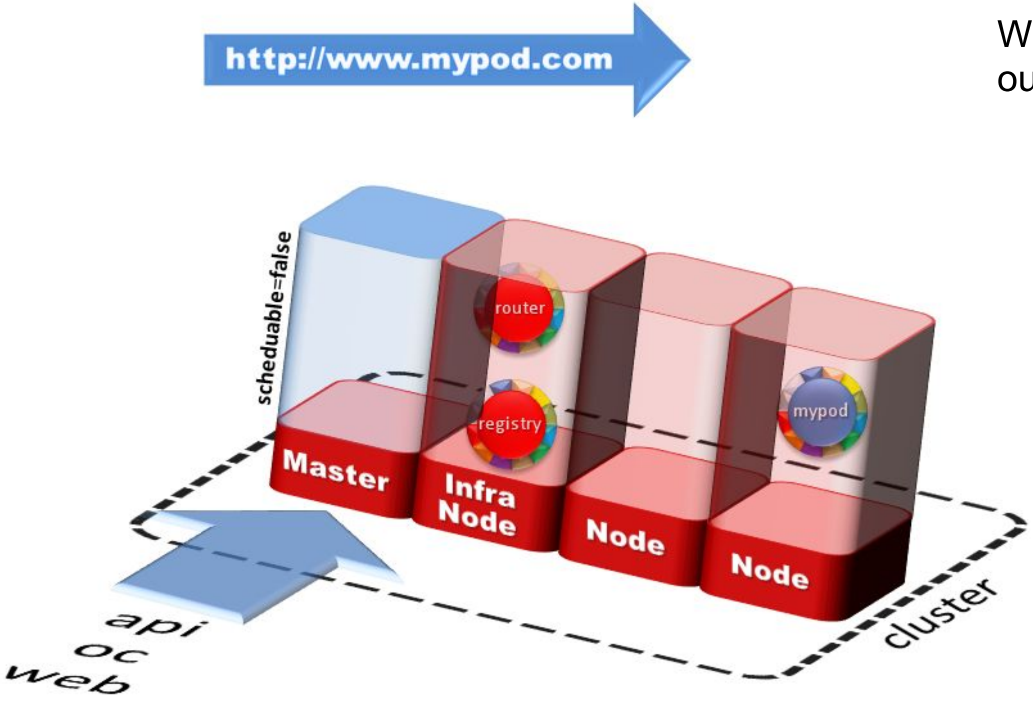
By hosting Pods of a Infrastructure nature, we usually call this Host:
Infra Node

Basic Concepts: Mechanics



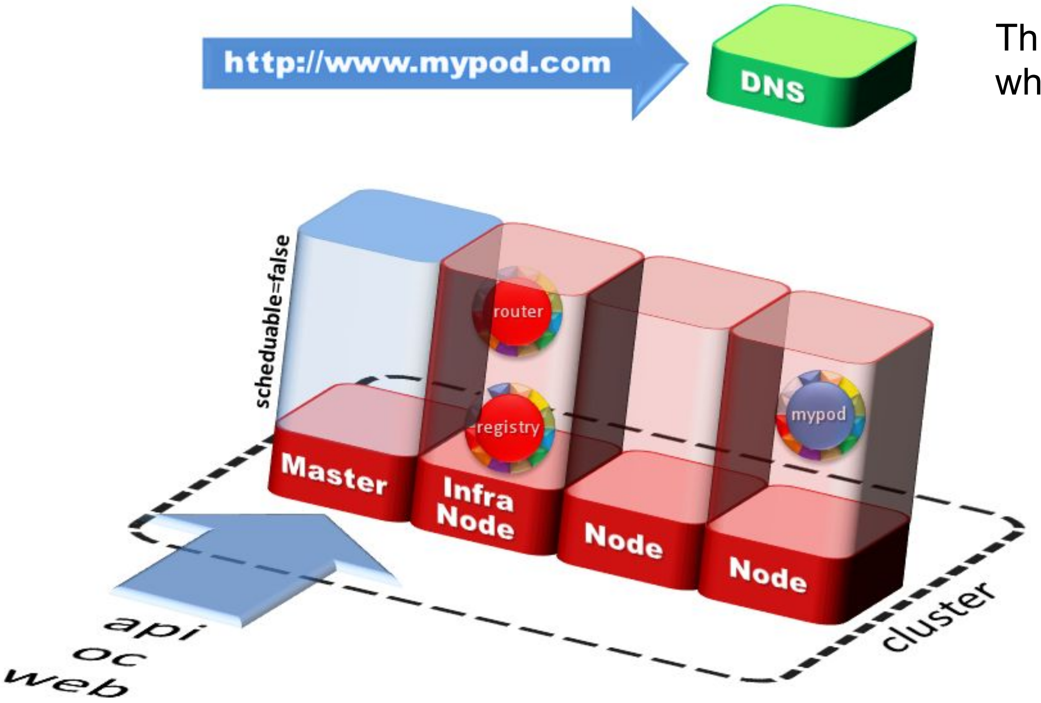
Now that we've got a fully working OpenShift, we're ready to create a Pod called "MyPod".

Basic Concepts: Mechanics



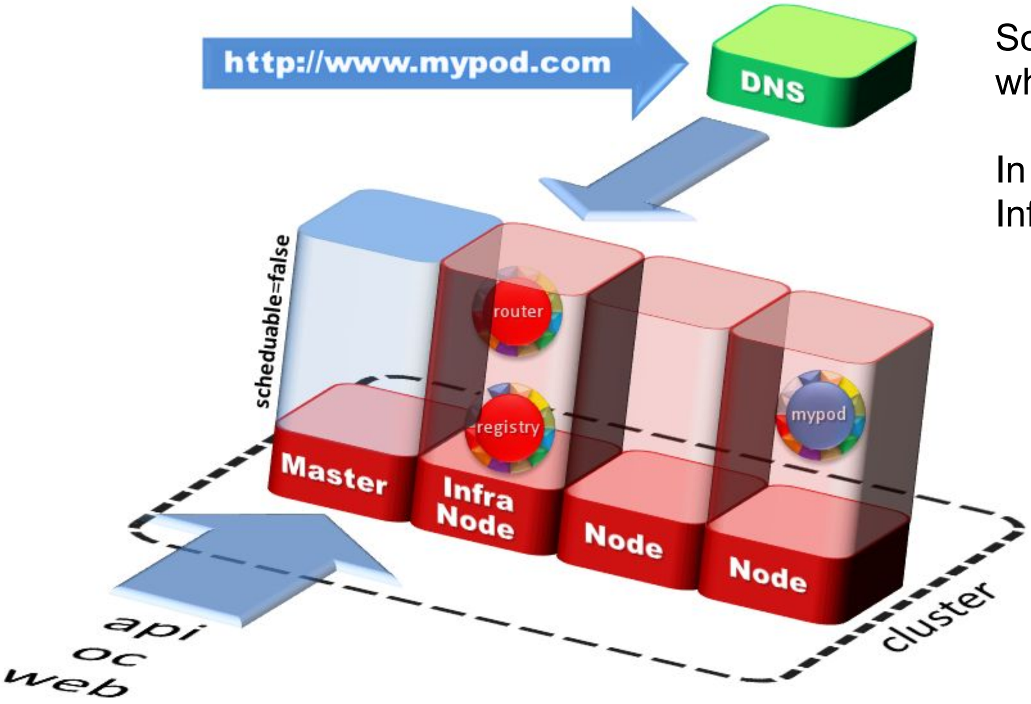
When someone needs to get access to our Pod, by typing an URL....

Basic Concepts: Mechanics



This request must first ask a DNS Server where I can find this particular resource.

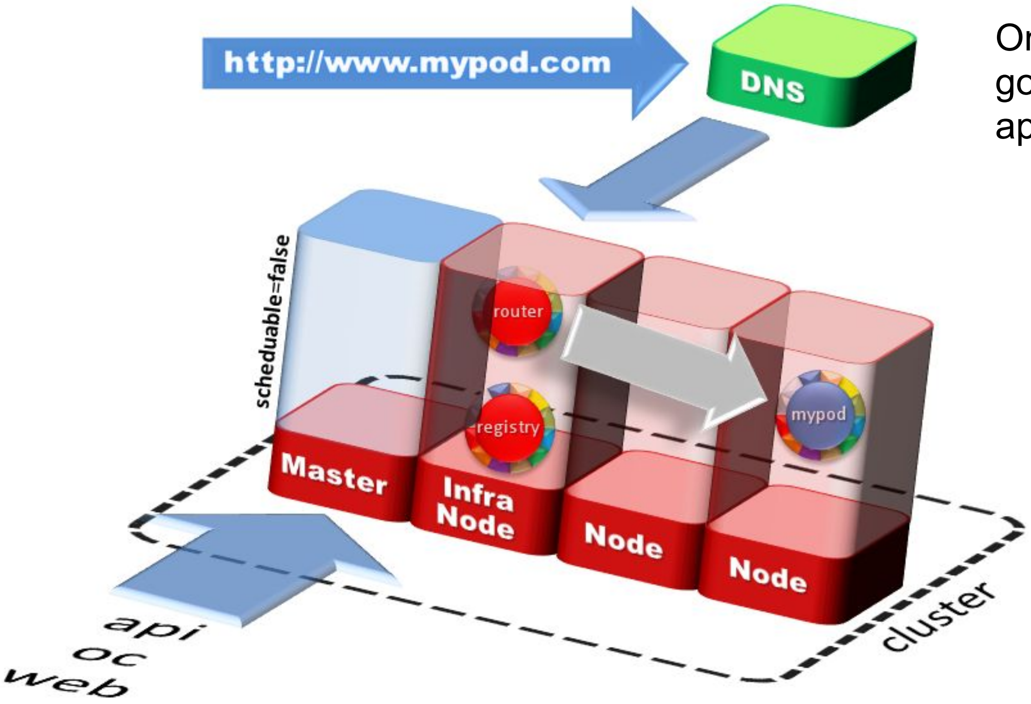
Basic Concepts: Mechanics



So the DNS, it's going to point to a Host where the Router is running.

In our example, the Pod is located at the Infra Node.

Basic Concepts: Mechanics



Once the request arrives at our Router, it's going to forward the request to the appropriate Pod in the cluster.

OpenShift Installation

Test Drive: OpenShift@Ops

OpenShift Installation

First the Master host need to be able to access each server in the cluster through SSH (including its own). For that, we’re going to create a SSH key to do that.

```
[root@bastion ~]# for host in {master,infranode,node}1; do
  echo -e "\n\n${host}"
  ssh ${host} "ssh-keygen -t rsa -b 4096 -q -N '' -f /root/.ssh/id_rsa"
done
```

Because we’re trying to do for the very first time, each host will ask for a password:

```
 /usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host master1 (192.168.0.4)' can't be established.
ECDSA key fingerprint is SHA256:REAM3DVAfxJcAB1oCzPc+sLlw9ezGIIYYs4+ombDtYM.
ECDSA key fingerprint is MD5:78:da:3d:2e:9f:7a:5d:46:87:7a:ce:88:f9:32:de:0f.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already
installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the
new keys
root@leandro01-infra's password:    r3dh4t1!

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh master1'"
and check to make sure that only the key(s) you wanted were added.
```

OpenShift Installation

First the Master host need to be able to access each server in the cluster through SSH (including its own). For that, we’re going to create a SSH key to do that.

```
[root@bastion ~]# for host in {master,infranode,node}1; do
  echo -e "\n\n${host}"
  ssh ${host} "ssh-keygen -t rsa -b 4096 -q -N '' -f /root/.ssh/id_rsa"
done
```

Because we’re trying to do for the very first time, each host will ask for a password:

```
 /usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host master1 (192.168.0.4)' can't be established.
ECDSA key fingerprint is SHA256:REAM3DVAfxJcAB1oCzPc+sLlw9ezGIIYYs4+ombDtYM.
ECDSA key fingerprint is MD5:78:da:3d:2e:9f:7a:5d:46:87:7a:ce:88:f9:32:de:0f.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already
installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the
new keys
root@leandro01-infra's password:   r3dh4t1!

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh master1'"
and check to make sure that only the key(s) you wanted were added.
```

OpenShift Installation

Now copy the generated public key to each host that we're going to use. Be sure to have your hosts right, because it's different for each attendee (in my example, my hosts are named: **master1**, **infranode1** and **node1**):

```
[root@bastion ~]# for host in {master,infranode,node}1; do
  echo -e "\n\n${host}"
  ssh-copy-id -i /root/.ssh/id_rsa.pub ${host}
done

[root@bastion ~]# ssh infranode1
[root@infranode1 ~]# for host in {master,infranode,node}1; do
  echo -e "\n\n${host}"
  ssh-copy-id -i /root/.ssh/id_rsa.pub ${host}
done
[root@infranode1 ~]# exit

[root@bastion ~]# ssh node1
[root@node1 ~]# for host in {master,infranode,node}1; do
  echo -e "\n\n${host}"
  ssh-copy-id -i /root/.ssh/id_rsa.pub ${host}
done
[root@node1 ~]# exit
```

OpenShift Installation

You may test if everything worked, by typing

```
[root@bastion ~]# ssh master1
Last login: Sun May 21 17:38:32 2017 from 177.73.70.108
[root@master1 ~]# exit
logout
Connection to master1 closed.
```

Now, try to log into infra’s and node’s host like

```
[root@bastion ~]# ssh infranode1
Last login: Sun May 21 17:38:32 2017 from 177.73.70.108
[root@infranode1 ~]# exit
logout
Connection to infra1 closed.

[root@bastion ~]# ssh node1
Last login: Sun May 21 17:38:32 2017 from 177.73.70.108
[root@node1 ~]# exit
logout
Connection to node1 closed.
```

Every single attempt to access each host must happen without asking for any password at all.

OpenShift Installation

Now, it’s important to install some tools in **Master / Infra / Node**, which it’s going to play a big part during the installation process:

```
[root@bastion ~]# for host in {master,infranode,node}1; do
  echo -e "\n\n${host}"
  ssh ${host} "/usr/bin/yum install -y wget git net-tools bind-utils yum-utils
iptables-services bridge-utils bash-completion kexec-tools sos psacct openshift-ansible
docker vim screen httpd-tools glusterfs glusterfs-client-xlators glusterfs-libs
glusterfs-fuse heketi-client"
done
```

… and update it, so everything will be up-to-date:

```
[root@bastion ~]# for host in {master,infranode,node}1; do
  echo -e "\n\n${host}"
  ssh ${host} "/usr/bin/yum -y update"
done
```

OpenShift Installation

Now, we need to prepare each host to be able to run Docker and for that, we need to ssh into each one of them and create a Docker storage data.
First, let’s make sure that on the Master, by understand which disks are available for us:

```
[root@master1 ~]# fdisk -l
Disk /dev/sda: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
...
...

Disk /dev/sdb: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
```

We’re going to realize that **`/dev/sdb`** is available for us to use as partition.

OpenShift Installation

Hence, we’re going to create a new partition on this disk to use as our Docker storage data, and setup a script named “docker-storage-setup” and running this script to get our docker storage data right in place:

```
[root@master1 ~]# vi /etc/sysconfig/docker-storage-setup

#STORAGE_DRIVER="devicemapper"
DEVS="sdb"
VG="docker-vg"
#DATA_SIZE=100%FREE
WIPE_SIGNATURES=true


[root@master1 ~]# docker-storage-setup
```

OpenShift Installation

Finally, we’re going to start it and enable it:

```
[root@master1 ~]# systemctl start docker
[root@master1 ~]# systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to
/usr/lib/systemd/system/docker.service.
```

TEST

Now, repeat the same tasks for your **infra** and **node** hosts. Starting on SSH key creation:

```
[root@bastion ~]# for host in {master,infranode,node}1; do
  echo -e "\n\n${host}"
  scp /etc/sysconfig/docker-storage-setup ${host}:/etc/sysconfig/docker-storage-setup
done

[root@bastion ~]# for host in {master,infranode,node}1; do
  echo -e "\n\n${host}"
  ssh ${host} "/usr/bin/docker-storage-setup; /usr/bin/systemctl start docker;
/usr/bin/systemctl enable docker"
done
```

OpenShift Installation

This is what an inventory looks like when you're editing:

```
[root@bastion ~]# cat /etc/ansible/hosts
...
[OSEv3:children]
...
timeout=60
ansible_user=ec2-user
ansible_become=yes
log_path=/root
...

masters
nodes
etcd

[masters]
master1.company-GUID.internal

[etcd]
master1.company-GUID.internal

[nodes]
## These are the masters
master1.company-GUID.internal openshift_node_group_name='node-config-master'

## These are infranodes
infranode1.company-GUID.internal openshift_node_group_name='node-config-infra'

## These are regular nodes
node1.company-GUID.internal openshift_node_group_name='node-config-compute'
node2.company-GUID.internal openshift_node_group_name='node-config-compute'
node3.company-GUID.internal openshift_node_group_name='node-config-compute'
node4.company-GUID.internal openshift_node_group_name='node-config-compute'
```

OpenShift Installation

We're going to run a playbook to perform all necessary check-ups

```
[root@bastion ~]# cd /usr/share/ansible/openshift-ansible/playbooks/
[root@bastion ~]# ansible-playbook prerequisites.yml

PLAY [Initialization Checkpoint Start]
**************************************************************************
*
...


PLAY RECAP
****************************************************************
localhost                  : ok=11   changed=0    unreachable=0
failed=0
one-infra                  : ok=63   changed=17   unreachable=0
failed=0
one-master                 : ok=74   changed=18   unreachable=0
failed=0
one-node1                  : ok=63   changed=17   unreachable=0
failed=0
```

OpenShift Installation

Finally, in order to start the installation, we’re going to run an Ansible’s playbook:

```
[root@master1 ~]# ansible-playbook deploy_cluster.yml

PLAY [Initialization Checkpoint Start]
**************************************************************************
*

TASK [Set install initialization 'In Progress']
**************************************************************************
*
ok: [one-master]

...
```

That should take on average 40 minutes to complete

OpenShift Installation

After the installation is done, you should see the following message:

```
INSTALLER STATUS
**********************************************************************************
Health Check               : Complete (0:00:34)
etcd Install               : Complete (0:01:11)
Master Install             : Complete (0:02:44)
Master Additional Install  : Complete (0:01:29)
Node Install               : Complete (0:07:26)
Hosted Install             : Complete (0:01:51)
Web Console Install        : Complete (0:00:57)
Metrics Install            : Complete (0:01:58)
Service Catalog Install    : Complete (0:02:33)
```

OpenShift Installation

Now, we’re going to install the tool responsible for OpenShift’s installation. This should only be installed in the **Master**:

```
[root@bastion ~]# yum -y atomic-openshift-clients

[root@bastion ~]# mkdir ~/.kube
[root@bastion ~]# scp master1.<GUID>.internal:/root/.kube/config /root/.kube/.

[root@bastion ~]# oc login -u system:admin
Logged into "https://master1.accenture-484e.internal:443" as "system:admin" using existing credentials.

You have access to the following projects and can switch between them with 'oc project <projectname>':

  * default
    kube-public
    kube-service-catalog
    kube-system
    management-infra
    openshift
    openshift-ansible-service-broker
    openshift-console
    openshift-infra
    openshift-logging
    openshift-metrics-server
    openshift-monitoring
    openshift-node
    openshift-sdn
    openshift-template-service-broker
    openshift-web-console
    operator-lifecycle-manager

Using project "default".
```

OpenShift Installation

Use the OC (OpenShift's Client) command and check if the nodes are Ready. It should be look like this:

```
[root@bastion ~]# oc get nodes
NAME                          STATUS    ROLES     AGE       VERSION
infranode1.<GUID>.internal    Ready     infra     10h       v1.11.0+d4cacc0
master1.<GUID>.internal       Ready     master    10h       v1.11.0+d4cacc0
node1.<GUID>.internal         Ready     compute   10h       v1.11.0+d4cacc0
```

and we can check if all the necessary pods are up and running:

```
[root@bastion ~]# oc get pods --all-namespaces
NAMESPACE                          NAME                            READY     STATUS    RESTARTS   AGE
default                            docker-registry-1-vn6nr         1/1       Running   0          22m
default                            registry-console-1-9hd84        1/1       Running   0          20m
default                            router-1-z4tlp                  1/1       Running   0          23m
kube-service-catalog               apiserver-hkxfw                 1/1       Running   0          17m
kube-service-catalog               controller-manager-h9jgd        1/1       Running   1          17m
openshift-ansible-service-broker   asb-1-deploy                    0/1       Pending   0          15m
openshift-ansible-service-broker   asb-etcd-1-deploy               0/1       Pending   0          15m
openshift-infra                    hawkular-cassandra-1-gj2gd      1/1       Running   0          18m
openshift-infra                    hawkular-metrics-hjh8s          1/1       Running   0          18m
openshift-infra                    heapster-9nf9j                  1/1       Running   0          18m
openshift-template-service-broker  apiserver-94m7g                 1/1       Running   0          14m
openshift-web-console              webconsole-68b848cb77-dqhhf     1/1       Running   1          22m
```

OpenShift Installation

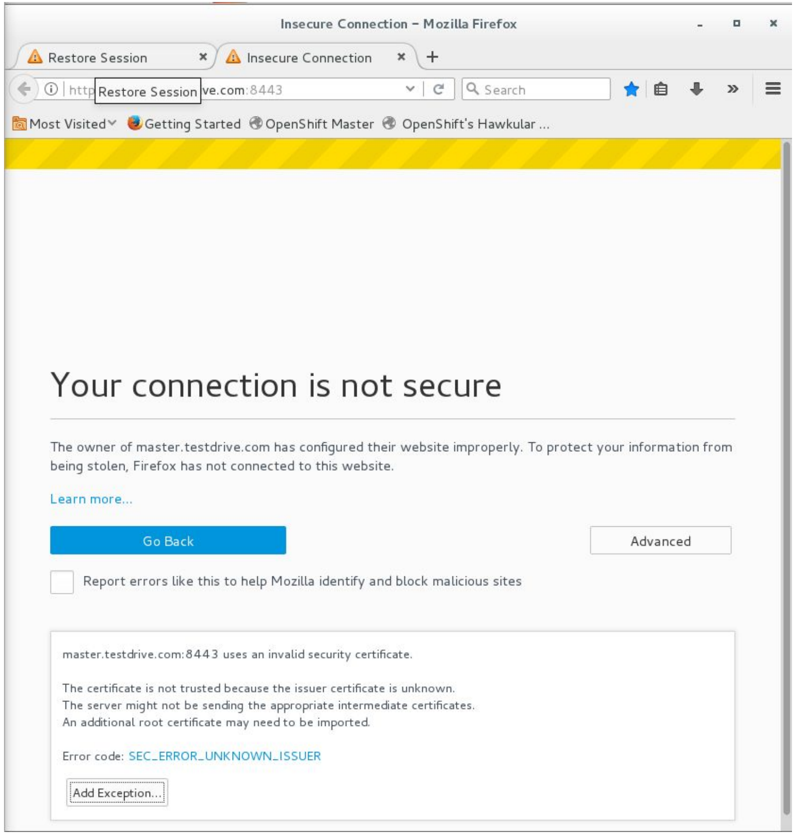
For this part, which we're telling the Master where the authentication should happen:

```
[root@bastion 0 ~]# ssh master1.floriapa-30a1.internal

[ec2-user@master1 ~]$ sudo less /etc/origin/master/master-config.yaml
...
oauthConfig:
  assetPublicURL: https://master.testdrive.com:8443/console/
  grantConfig:
    method: auto
  identityProviders:
  - challenge: true
    login: true
    mappingMethod: claim
    name: htpasswd
    provider:
      apiVersion: v1
      kind: HTPasswdPasswordIdentityProvider
      file: /etc/origin/master/htpasswd
  masterCA: ca-bundle.crt
  masterPublicURL: https://master.company-GUID.openshiftworkshop.com:443
  masterURL: https://master1.company-GUID.internal:443
  sessionConfig:
    sessionMaxAgeSeconds: 3600
    sessionName: ssn
    sessionSecretsFile: /etc/origin/master/session-secrets.yaml
  tokenConfig:
    accessTokenMaxAgeSeconds: 86400
    authorizeTokenMaxAgeSeconds: 500
...
```

OpenShift Access

`https://master.<GUID>.open.redhat.com`

OpenShift Installation



Your connection is not secure

The owner of master.testdrive.com has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.

Learn more...

□ Report errors like this to help Mozilla identify and block malicious sites

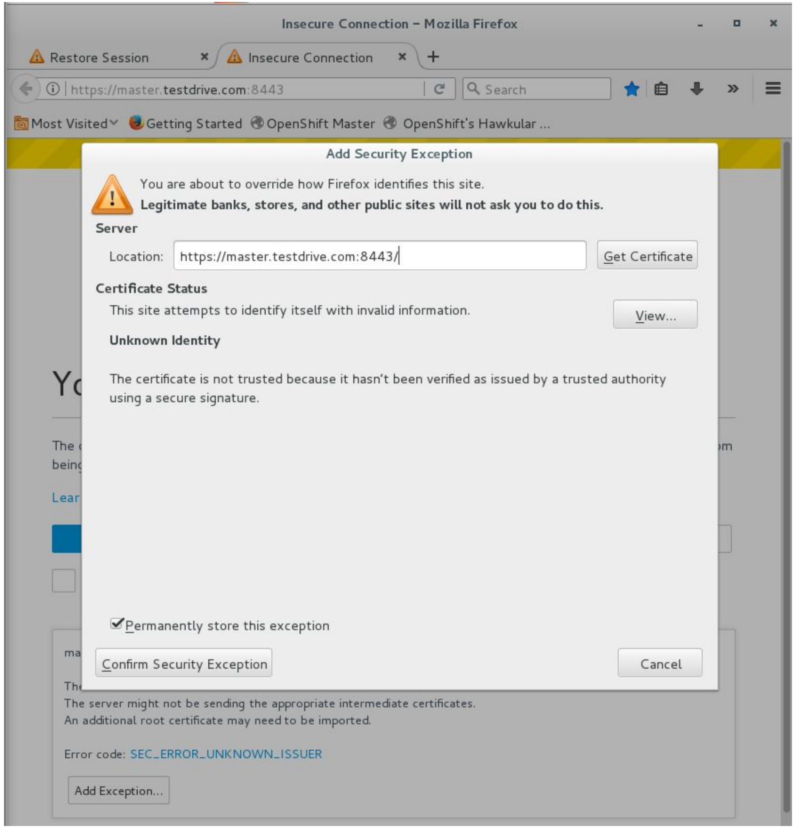
master.testdrive.com:8443 uses an invalid security certificate.

The certificate is not trusted because the issuer certificate is unknown.
The server might not be sending the appropriate intermediate certificates.
An additional root certificate may need to be imported.

Error code: SEC_ERROR_UNKNOWN_ISSUER

Because you’re accessing for the very first time OpenShift’s WebConsole, which works in a secure endpoint https, you need to accept the certificate:

OpenShift Installation



https://master.testdrive.com:8443

Add Security Exception

You are about to override how Firefox identifies this site.
Legitimate banks, stores, and other public sites will not ask you to do this.

Server

Location: https://master.testdrive.com:8443/

Certificate Status

This site attempts to identify itself with invalid information.

Unknown Identity

The certificate is not trusted because it hasn't been verified as issued by a trusted authority using a secure signature.

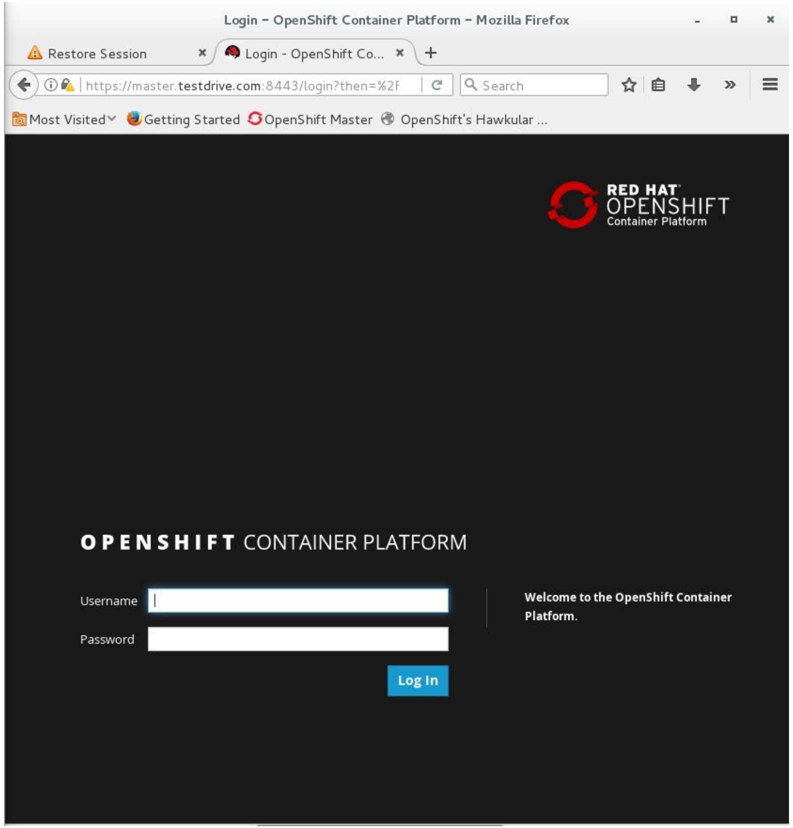
☑ Permanently store this exception

The server might not be sending the appropriate intermediate certificates.
An additional root certificate may need to be imported.

Error code: SEC_ERROR_UNKNOWN_ISSUER

Be sure to make the address “master.<GUID>.open.redhat.com” trustable:

OpenShift Installation



Login – OpenShift Container Platform – Mozilla Firefox

https://master.testdrive.com:8443/login?then=%2f

OPENSHIFT CONTAINER PLATFORM

Username

Password

Log In

Welcome to the OpenShift Container Platform.

Now, you can get access to OpenShift’s WebConsole:

Test Drive: OpenShift@Ops

First Steps

Test Drive: OpenShift@Ops

First Steps

Let's create a new project to get started (Although we're creating this project as OpenShift's Administrator, **`system:admin`**, we're going to create a project for user **`opentlc-mgr`**, so you can watch the resources also at the WebConsole).

```
[root@bastion ~]# oc login master.<GUID>.open.redhat.com:443 -u opentlc-mgr
Authentication required for https://master.<GUID>.open.redhat.com:443 (openshift)
Username: opentlc-mgr
Password: r3dh4t1!
Login successful.

[root@bastion ~]# oc adm new-project first-project-userID --display-name="My First
Project - userID" --admin=userID
Created project first-project-userID

[root@bastion ~]# oc login master.<GUID>.open.redhat.com:443 -u userID
Authentication required for https://master.<GUID>.open.redhat.com:443 (openshift)
Username: userID
Password: openshift
Login successful.

[root@bastion ~]# oc project first-project-userID
Now using project "first-project-userID" on server "https://master.<GUID>.open.redhat.com:443".
```

First Steps

```
[root@bastion ~]# oc get projects
NAME                     DISPLAY NAME                  STATUS
first-project-userID           My First Project - userID     Active


[root@bastion ~]# oc project first-project-userID
Using project "first-project-userID" on server "https://master.<GUID>.open.redhat.com:443".
```

First Steps

In our first project, we're going to create our first Pod.

```
[root@bastion ~]# oc create -f -<<EOF
apiVersion: v1
kind: Pod
metadata:
  name: first-pod-userID
spec:
  containers:
  - image: jboss/wildfly:latest
    imagePullPolicy: IfNotPresent
    name: first-pod-userID
    ports:
    - containerPort: 8080
      protocol: TCP
  dnsPolicy: ClusterFirst
  restartPolicy: Always
EOF
pod "appserver" created
```

First Steps

You can type this command to see the available Pods:

```
[root@bastion ~]# oc get pods
NAME              READY       STATUSRESTARTS    AGE
first-pod-userID  1/1         Running     0           1m
```

Or you can see how that evolves by type:

```
[root@bastion ~]# oc get pods --watch
NAME              READY STATUS                  RESTARTS    AGE
first-pod-userID        0/1         ContainerCreating       0           9s
first-pod-userID        1/1         Running                 0           19s
```

First Steps

You can type this command to see the Pods content:

```
[root@bastion ~]# oc get pods/first-pod-userID -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
      openshift.io/scc: restricted
  creationTimestamp: 2019-09-16T21:32:07Z
  name: first-pod-userID
  namespace: first-project-userID
  resourceVersion: "43381"
  selfLink: /api/v1/namespaces/first-project-userID/pods/first-pod-userID
  uid: 6f036ca3-d8c9-11e9-9abd-02c173f05590
spec:
  containers:
  - image: jboss/wildfly:latest
      imagePullPolicy: IfNotPresent
      name: first-pod-userID
      ports:
      - containerPort: 8080
      protocol: TCP
      resources: {}
      securityContext:
      capabilities:
    ...
```

First Steps

You can see also how each of the Pods developed over time by see each state

```
[root@bastion ~]# oc get events
LAST SEEN   FIRST SEEN   COUNT      NAME                          KIND  SUBOBJECT
TYPE  REASON      SOURCE                              MESSAGE
4m          4m          1     first-pod-userID.15c507d9944f1903   Pod
NormalScheduled   default-scheduler                   Successfully assigned
first-project-user1/first-pod-userID to node4.<GUID>.internal
4m          4m          1     first-pod-userID.15c507d9f3f8a2e4   Pod   spec.containers{first-pod-userID}
NormalPulling     kubelet, node4.<GUID>.internal   pulling image "jboss/wildfly:latest"
3m          3m          1     first-pod-userID.15c507dde4a31168   Pod   spec.containers{first-pod-userID}
NormalPulled      kubelet, node4.<GUID>.internal   Successfully pulled image "jboss/wildfly:latest"
3m          3m          1     first-pod-userID.15c507dde66bfaee   Pod   spec.containers{first-pod-userID}
NormalCreated     kubelet, node4.<GUID>.internal   Created container
3m          3m          1     first-pod-userID.15c507ddeefec660   Pod   spec.containers{first-pod-userID}
NormalStarted     kubelet, node4.<GUID>.internal   Started container
```

First Steps

And once the Pod is actually running, we can see the logs.

```
[root@bastion ~]# oc logs first-pod-userID
=========================================================================

  JBoss Bootstrap Environment

  JBOSS_HOME: /opt/jboss/wildfly

  JAVA: /usr/lib/jvm/java/bin/java

  JAVA_OPTS:  -server -Xms64m -Xmx512m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m
-Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true
--add-exports=java.base/sun.nio.ch=ALL-UNNAMED --add-exports=jdk.unsupported/sun.misc=ALL-UNNAMED
--add-exports=jdk.unsupported/sun.reflect=ALL-UNNAMED

=========================================================================

21:18:15,699 INFO  [org.jboss.modules] (main) JBoss Modules version 1.9.1.Final
21:18:16,143 INFO  [org.jboss.msc] (main) JBoss MSC version 1.4.8.Final
21:18:16,152 INFO  [org.jboss.threads] (main) JBoss Threads version 2.3.3.Final
21:18:16,273 INFO  [org.jboss.as] (MSC service thread 1-2) WFLYSRV0049: WildFly Full 17.0.1.Final
(WildFly Core 9.0.2.Final) starting
21:18:17,023 INFO  [org.wildfly.security] (ServerService Thread Pool -- 28) ELY00001: WildFly Elytron
version 1.9.1.Final
...
```

First Steps

So, what happens if I want to know which server is running my Pod. I can use "oc describe"

```
[root@bastion ~]# oc describe pods/first-pod-userID
Name:             first-pod-userID
Namespace:        first-project-userID
Priority:         0
PriorityClassName:  <none>
Node:             node4.<GUID>.internal/192.168.0.202
Start Time:       Mon, 16 Sep 2019 21:17:56 +0000
Labels:           <none>
Annotations:      openshift.io/scc=restricted
Status:           Running
IP:               10.1.4.4
Containers:
  appserver:
      Container ID:   docker://f45358c666d2493a9b5cc2e7a43f63456f1adff72046054a6f7a9c8fd4c9e452
      Image:      jboss/wildfly:latest
      Image ID:
docker-pullable://docker.io/jboss/wildfly@sha256:c3fe28079103ca8c70d73f3d93626f2f862179875779ea2b9bab70ee
502531df
      Port:       8080/TCP
      Host Port:  0/TCP
      State:      Running
      Started:    Mon, 16 Sep 2019 21:18:14 +0000
      Ready:      True
...
```

Red Hat

First Steps

So, let's see our Pods available

```
[root@bastion ~]# oc get pods
NAME                  READY        STATUSRESTARTS     AGE
first-pod-userID   1/1           Running      0            13m
```

...and then, try to delete our running Pod.

```
[root@bastion ~]# oc delete pod/first-pod-userID
pod "first-pod-userID" deleted
```

...by checking again, you will see there isn't any more pods available.

```
[root@bastion ~]# oc get pods
No resources found.
```

First Steps

Now, instead of using a Pod Resource, we're going to create a DeploymentConfig instead.

```
[root@bastion ~]# oc create -f -<<EOF
apiVersion: v1
kind: DeploymentConfig
metadata:
  name: first-dc-userID
  labels:
    name: first-dc-userID
spec:
  replicas: 1
  selector:
    app: first-dc-userID
    deploymentconfig: first-dc-userID
  template:
    metadata:
      labels:
        app: first-dc-userID
        deploymentconfig: first-dc-userID
    spec:
      containers:
      - image: jboss/wildfly:latest
        name: first-dc-userID
        ports:
        - containerPort: 8080
          protocol: TCP
EOF
deploymentconfig "first-dc-userID" created
```

First Steps

Let's see how this particular Resource develops into a Pod

```
[root@bastion ~]# oc get pods -w
NAME                    READY       STATUS            RESTARTS    AGE
first-dc-userID-1-sjvts 0/1         ContainerCreating       0           4s
first-dc-userID-1-sjvts 1/1         Running                 0           4s
first-dc-userID-1-sjvts   0/1       Pending                 0           0s
first-dc-userID-1-sjvts   0/1       Pending                 0           0s
first-dc-userID-1-sjvts   0/1       ContainerCreating       0           0s
first-dc-userID-1-sjvts   1/1       Running                 0           5s
first-dc-userID-1-sjvts   0/1       Completed               0           10s
first-dc-userID-1-sjvts   0/1       Terminating             0           10s
first-dc-userID-1-sjvts   0/1       Terminating             0           10s
```

First Steps

And now that we've got our first DeploymentConfig (or DC) available, we can list it

```
[root@bastion ~]# oc get dc
NAME               REVISION     DESIRED      CURRENT      TRIGGERED BY
first-dc-userID    1            1            1            config


[root@bastion ~]# oc get dc -o yaml
apiVersion: v1
items:
- apiVersion: apps.openshift.io/v1
  kind: DeploymentConfig
  metadata:
      creationTimestamp: 2019-09-16T21:59:28Z
      generation: 1
      labels:
      name: first-dc-userID
      name: first-dc-userID
      namespace: first-project-userID
      resourceVersion: "47757"
      selfLink:
/apis/apps.openshift.io/v1/namespaces/first-project-userID/deploymentconfigs/first-dc-userID
      uid: 413a5a0b-d8cd-11e9-9abd-02c173f05590
  spec:
      replicas: 1
      revisionHistoryLimit: 10
      ...
```

First Steps

Like the previous example, let's list our pods

```
[root@bastion ~]# oc get pods
NAME                     READY        STATUSRESTARTS    AGE
first-dc-userID-1-sjvts 1/1          Running      0           6m
```

...and then, try to delete it

```
[root@bastion ~]# oc delete pod first-dc-userID-1-sjvts
pod "first-dc-userID-1-sjvts" deleted
```

Even though the Pod was deleted, see what happens next.

```
[root@bastion ~]# oc get pods -w
NAME                           READY        STATUS                 RESTARTS    AGE
first-dc-userID-1-m69gp        0/1          ContainerCreating           0            1s
first-dc-userID-1-sjvts        1/1          Terminating                 0            31s
first-dc-userID-1-sjvts        0/1          Terminating                 0            32s
first-dc-userID-1-m69gp        1/1          Running                     0            7s
```

First Steps

Next, we're going to create a Service Resource

```
[root@bastion ~]# oc create -f -<<EOF
apiVersion: v1
kind: Service
metadata:
  labels:
    name: first-svc-userID
  name: first-svc-userID
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    app: first-dc-userID
    deploymentconfig: first-dc-userID
  sessionAffinity: None
  type: ClusterIP
EOF
service "appserver" created
```

First Steps

Like DC, we can always list all Service Resources available

```
[root@bastion ~]# oc get services
NAME              TYPE        CLUSTER-IP        EXTERNAL-IP       PORT(S)     AGE
first-svc-userID  ClusterIP   172.30.136.16     <none>      8080/TCP    9s


[root@bastion ~]# oc get services -o wide
NAME        TYPE        CLUSTER-IP  EXTERNAL-IP   PORT(S)   AGE   SELECTOR
first-svc-userID   ClusterIP   172.30.136.16   <none>       8080/TCP   1m
app=first-dc-userID,deploymentconfig=first-dc-userID


[root@bastion 0 ~]# oc get service -o yaml
apiVersion: v1
items:
- apiVersion: v1
  kind: Service
  metadata:
      creationTimestamp: 2019-09-16T22:17:52Z
      labels:
      name: first-svc-userID
      name: first-svc-userID
      namespace: first-project-userID
      resourceVersion: "50678"
...
```

First Steps

And finally, we're going to create a Route that allows us to access this Pod externally

```
[root@bastion ~]# oc create -f -<<EOF
apiVersion: v1
kind: Route
metadata:
  labels:
    name: first-route-userID
  name: first-route-userID
spec:
  host: first-app-userID.apps.<GUID>.open.redhat.com
  port:
    targetPort: 8080
  to:
    kind: Service
    name: first-svc-userID
    weight: 100
EOF
route "appserver" created
```

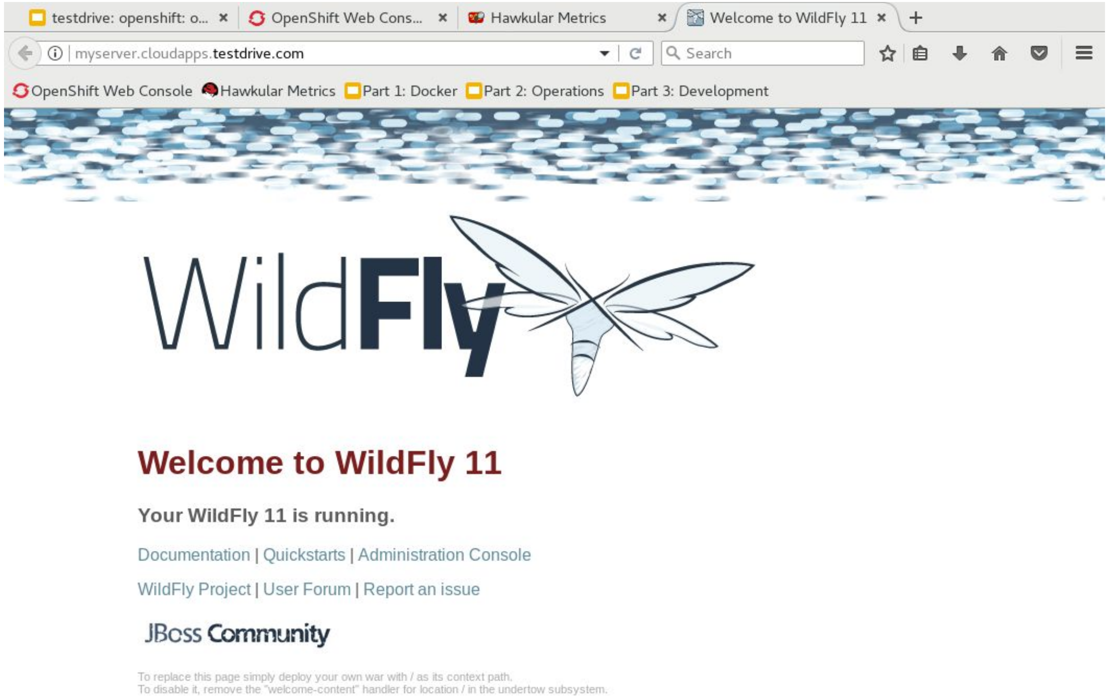
First Steps

Next, we make sure the route is available for us

```
[root@bastion ~]# oc get routes
NAME                  HOST/PORT                                                PATH   SERVICES     PORT   TERMINATION
WILDCARD
first-route-userID    first-app-userID.apps.<GUID>.open.redhat.com                          first-svc-userID   8080
      None
```

```
[root@bastion ~]# oc get routes -o yaml
apiVersion: v1
items:
- apiVersion: route.openshift.io/v1
  kind: Route
  metadata:
      creationTimestamp: 2019-09-16T22:25:45Z
      labels:
      name: first-route-useid
      name: first-route-userID
      namespace: first-project-userID
      resourceVersion: "51941"
      selfLink: /apis/route.openshift.io/v1/namespaces/first-project-userID/routes/first-route-userID
      uid: ed4ade11-d8d0-11e9-9abd-02c173f05590
  spec:
      ...
```

First Steps



Welcome to WildFly 11

Your WildFly 11 is running.

Documentation | Quickstarts | Administration Console

WildFly Project | User Forum | Report an issue

To replace this page simply deploy your own war with / as its context path.
To disable it, remove the "welcome-content" handler for location / in the undertow subsystem.

By opening our WebBrowser and typing our URL created in a Route, we should see this:

http://first-app-userID.apps.<GUID>.open.redhat.com

First Steps

Before using, let's first delete and old project:

```
[root@bastion ~]# oc delete project first-project-userID
project.project.openshift.io "first-project-userID" deleted
```

...and create a new one

```
[root@bastion ~]# oc new-project templates-project-userID --display-name="Templates
Project - userID"
Now using project "templates-project-userID" on server "https://master.<GUID>.open.redhat.com:443".

[root@bastion 0 ~]# oc get projects
NAME                         DISPLAY NAME                STATUS
first-project-userID         My First Project - UserID        Active
templates-project-userIDTemplates Project - UserID       Active

[root@bastion ~]# oc project templates-project-userID
Now using project "templates-project-userID" on server "master.<GUID>.open.redhat.com:443".

[root@bastion 0 ~]# oc project
Using project "templates-project-userID" on server "https://master.<GUID>.open.redhat.com:443".
```

Access the Github and create the file template-userID.yaml with the correct parameters for **userID** and **<GUID>.**

template-userID.yaml

```
apiVersion: v1
kind: Template
metadata:
   name: template-userID
parameters:
- name: APPLICATION_NAME
  displayName: Application's Name
  description: How you want to call your application's name
  required: true
  value: app-userID
- name: GUID
  displayName: GUID's Name
  description: GUID for Test Drive Access
  required: true
  value: <GUID>
objects:
- apiVersion: v1
  kind: DeploymentConfig
  metadata:
          name: ${APPLICATION_NAME}
          labels:
          name: ${APPLICATION_NAME}
  spec:
          replicas: 1
          selector:
          app: ${APPLICATION_NAME}
          deploymentconfig: ${APPLICATION_NAME}
          template:
          metadata:
          labels:
          app: ${APPLICATION_NAME}
          deploymentconfig: ${APPLICATION_NAME}
          spec:
          containers:
          - image: jboss/wildfly:latest
          name: ${APPLICATION_NAME}
          ports:
          - containerPort: 8080
          protocol: TCP
- apiVersion: v1
  kind: Service
  metadata:
          labels:
          name: ${APPLICATION_NAME}
          name: ${APPLICATION_NAME}
  spec:
          ports:
```

First Steps

Now, we're going to get all the previous resources into a single file called Template. By running the command "oc new-app" we're able to create all the resources into a single shot:

```
[root@bastion ~]# oc new-app
https://raw.githubusercontent.com/git-user/git-project/master/template-userID.yaml

--> Deploying template "templates-project-userID/template-userID" for
"https://raw.githubusercontent.com/git-user/git-project/master/template-userID.yaml" to project
templates-project-userID

     * With parameters:
        * Application's Name=app-userID
        * GUID's Name=<GUID>

--> Creating resources ...
      deploymentconfig.apps.openshift.io "app-userID" created
      service "app-userID" created
      route.route.openshift.io "app-userID" created
--> Success
      Access your application via route 'app-userID.apps.<GUID>.open.redhat.com'
      Run 'oc status' to view your app.
```

First Steps

Verify all resources configured:

```
[root@bastion ~]# oc get events
LAST SEEN   FIRST SEEN   COUNT      NAME                                NAME                    SUBOBJECT
      TYPE  REASON            SOURCE                              MESSAGE
8m          8m          1     app-userID-1-deploy.15c50d9cb5db5920   Pod
      NormalScheduled         default-scheduler                   Successfully assigned
templates-project-userID/app-userID-1-deploy to node3.<GUID>.internal
8m          8m          1     app-userID.15c50d9cb4d6d330         DeploymentConfig
      NormalDeploymentCreated   deploymentconfig-controller       Created new replication controller
"app-userID-1" for version 1
8m          8m          1     app-userID-1-deploy.15c50d9d409a8914   Pod
spec.containers{deployment}   NormalPulling           kubelet, node3.<GUID>.internal   pulling image
"registry.redhat.io/openshift3/ose-deployer:v3.11.104"
8m          8m          1     app-userID-1-deploy.15c50d9d8bdbc1f9   Pod
spec.containers{deployment}   NormalPulled            kubelet, node3.<GUID>.internal   Successfully
pulled image "registry.redhat.io/openshift3/ose-deployer:v3.11.104"
8m          8m          1     app-userID-1-deploy.15c50d9d8dd7ead6   Pod
spec.containers{deployment}   NormalCreated           kubelet, node3.<GUID>.internal   Created container
8m          8m          1     app-userID-1-deploy.15c50d9d968ca405   Pod
spec.containers{deployment}   NormalStarted           kubelet, node3.<GUID>.internal   Started container
8m          8m          1     app-userID-1.15c50d9da20c3743       ReplicationController
      NormalSuccessfulCreate  replication-controller              Created pod: app-userID-1-m7bgc
...
```

First Steps

Verify all resources configured:

```
[root@bastion ~]# oc get routes
NAME         HOST/PORT                                  PATH   SERVICES     PORT   TERMINATION   WILDCARD
app-userID    app-userID.apps.<GUID>.open.redhat.com           app-userID    8080                     None

[root@bastion ~]# oc get services
NAME         TYPE         CLUSTER-IP   EXTERNAL-IP    PORT(S)    AGE
app-userID    ClusterIP    172.30.1.98    <none>          8080/TCP    11m

[root@bastion ~]# oc get pods -o wide
NAME                 READY STATUSRESTARTS   AGE     IP           NODE                         NOMINATED NODE
app-userID-1-m7bgc    1/1       Running    0        12m    10.1.12.5    node2.<GUID>.internal    <none>

[root@bastion ~]# oc logs app-userID-1-m7bgc

[root@bastion ~]# oc get dc
NAME         REVISION    DESIRED    CURRENT    TRIGGERED BY
app-userID    1           1           1           config
```

First Steps

Now, we're going to download a template that creates everything

```
[root@bastion ~]# curl -O
https://raw.githubusercontent.com/ldsanches/ocp-test-drive/master/mytemplate.yaml

  % Total    % Received % Xferd  Average Speed   Time    Time    Time   Current
                                 Dload  Upload   Total   Spent   Left   Speed
100  1536  100  1536    0     0   8057      0 --:--:-- --:--:-- --:--:--  8084

[root@bastion ~]# sed -i 's/<GUID>/testdrive-1234/g' mytemplate.yaml
[root@bastion ~]# sed -i 's/-userID/-user5678/g' mytemplate.yaml
```

...and then, create our template directly into our namespace:

```
[root@bastion ~]# oc create -f mytemplate.yaml
template.template.openshift.io/template-userID created


[root@bastion ~]# oc get template
NAME                    DESCRIPTION       PARAMETERS        OBJECTS
template-user2                            2 (all set)       3
```

Note that this template only exists on project **`templates-project-userID`**

First Steps

Now, we're going to use this template, which in turns it's going to create all the necessary resources for us, but before, we need to remove the lats resources created.

```
[root@bastion ~]# oc delete routes app-userID
route "app-userID" deleted

[root@bastion ~]# oc delete services app-userID
service "app-userID" deleted

[root@bastion ~]# oc delete dc app-userID
deploymentconfig "app-userID" deleted

[root@bastion ~]# oc new-app template-userID -p APPLICATION_NAME= sanches-app
--> Deploying template "templates-project-userID/template-userID" to project templates-project-userID

      * With parameters:
      * Application's Name=sanches-app
      * GUID's Name=orizon-3505

--> Creating resources ...
      deploymentconfig.apps.openshift.io "sanches-app" created
      service "sanches-app" created
      route.route.openshift.io "sanches-app" created
--> Success
      Access your application via route 'sanches-app.apps.<GUID>.open.redhat.com'
      Run 'oc status' to view your app.
```

First Steps

Let's check our Pod being created

```
[root@bastion ~]# oc get pods -w
NAME                 READY     STATUS     RESTARTS    AGE
sanches-app-1-9bz16    1/1        Running    0           32s
```

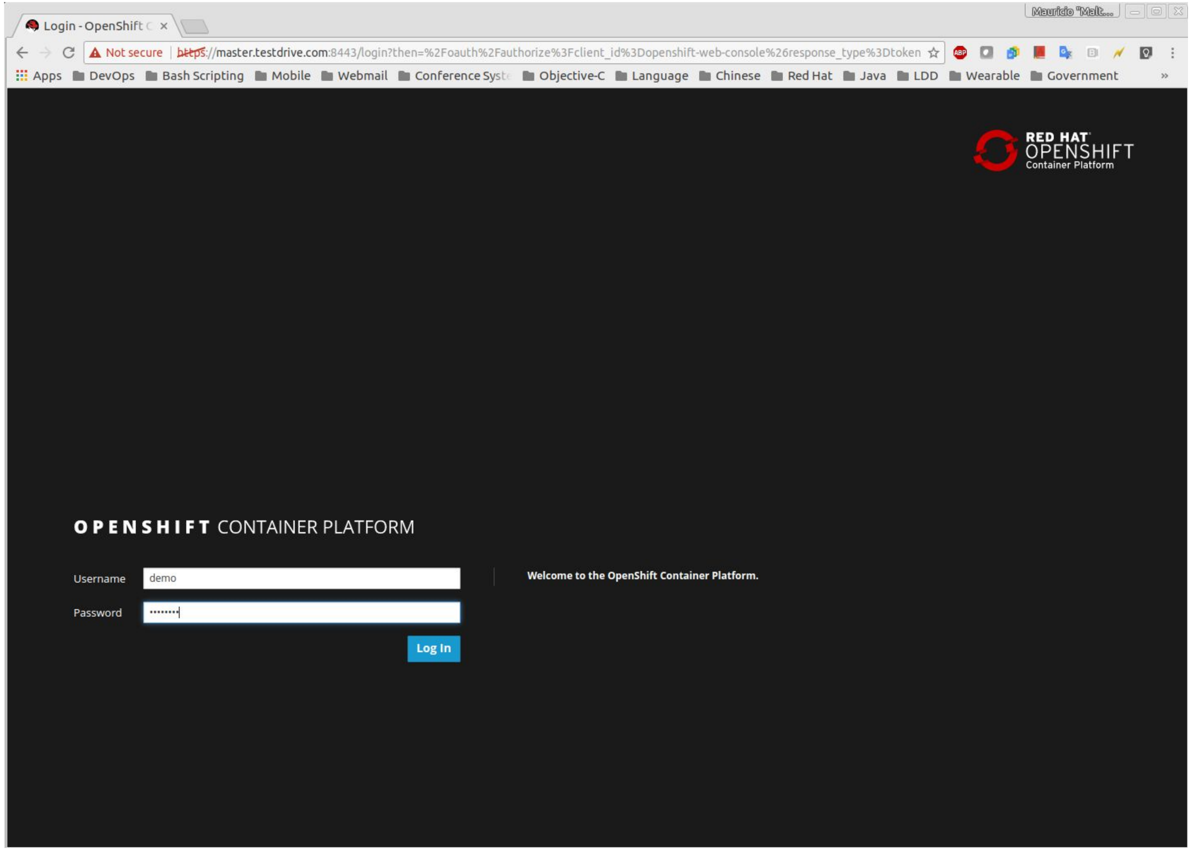
Let's see if there is any Service Resource available

```
[root@bastion ~]# oc get services
NAME        CLUSTER-IP        EXTERNAL-IP    PORT(S)     AGE
sanches-app    172.30.17.123    <none>        8080/TCP    45s
```

And some of routes

```
[root@bastion ~]# oc get routes
NAME                 HOST/PORT                                         PATH         SERVICES          PORT
TERMINATION          WILDCARD
Sanches-app          sanches.apps.<GUID>.openshiftworkshop.com                      sanches-app       8080
None
```

First Steps



Not secure | ~~https~~://master.testdrive.com:8443/login?then=%2Foauth%2Fauthorize%3Fclient_id%3Dopenshift-web-console%26response_type%3Dtoken

OPENSHIFT CONTAINER PLATFORM

Username: demo

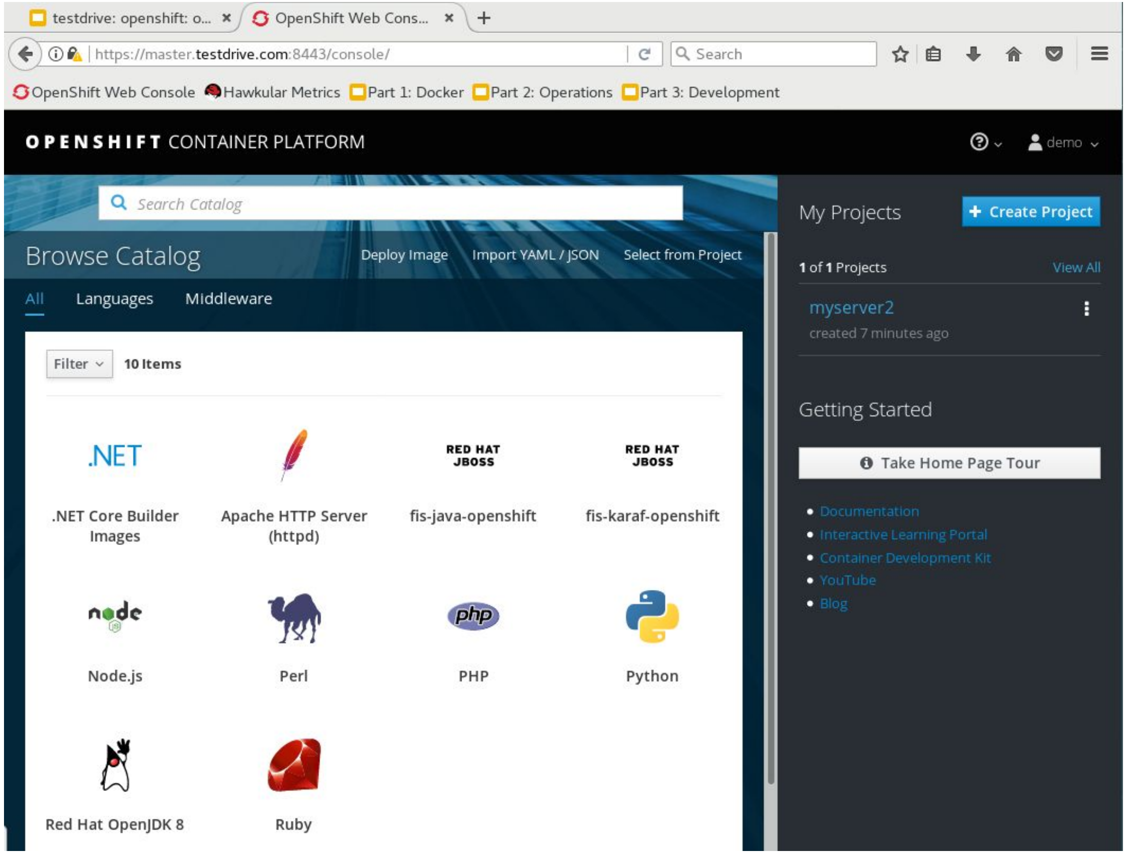
Password: ••••••••

Welcome to the OpenShift Container Platform.

Go to OpenShift's Console, and log as user **UserID** and password **openshift**

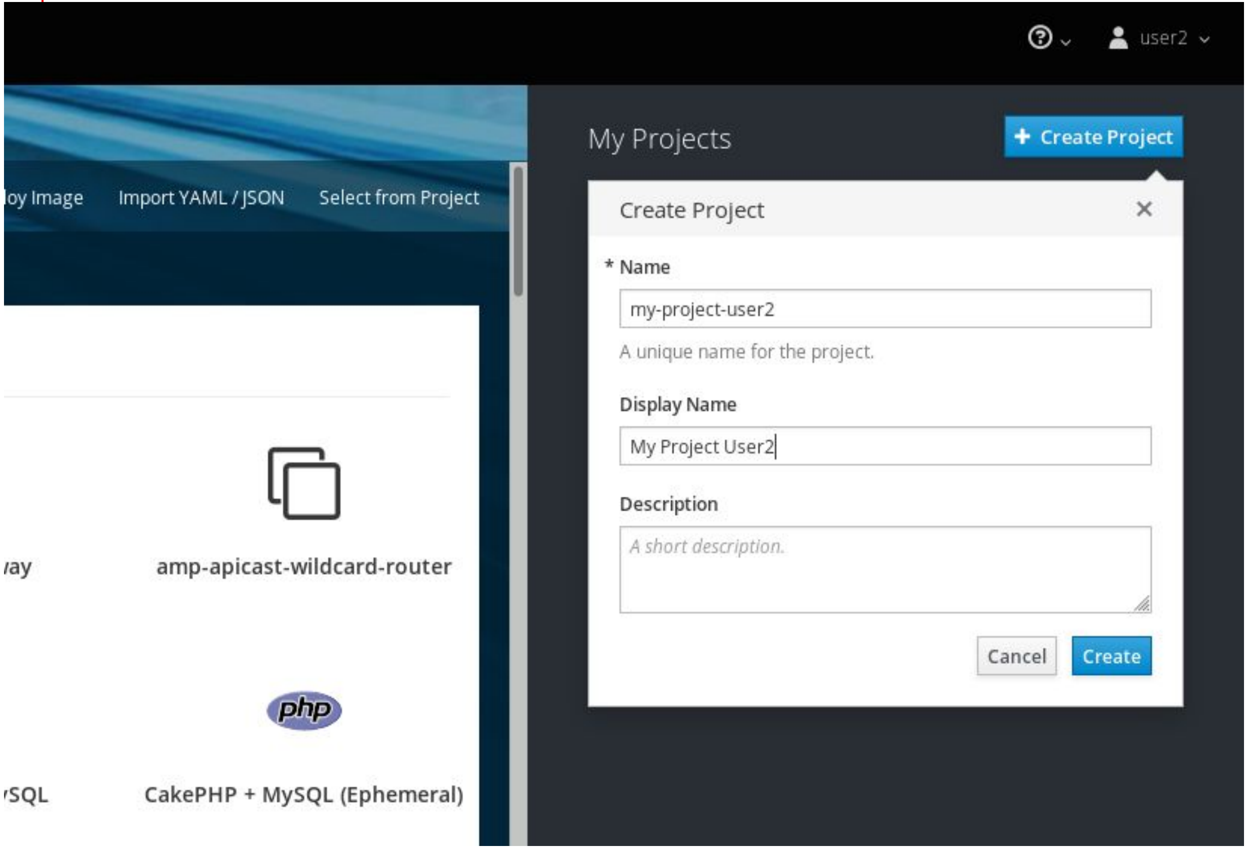
https://master.<GUID>.open.redhat.com

First Steps



Create a new project called **my-project-userID** with display name **“My Project UserID”**

First Steps



* Name

my-project-user2

A unique name for the project.

Display Name

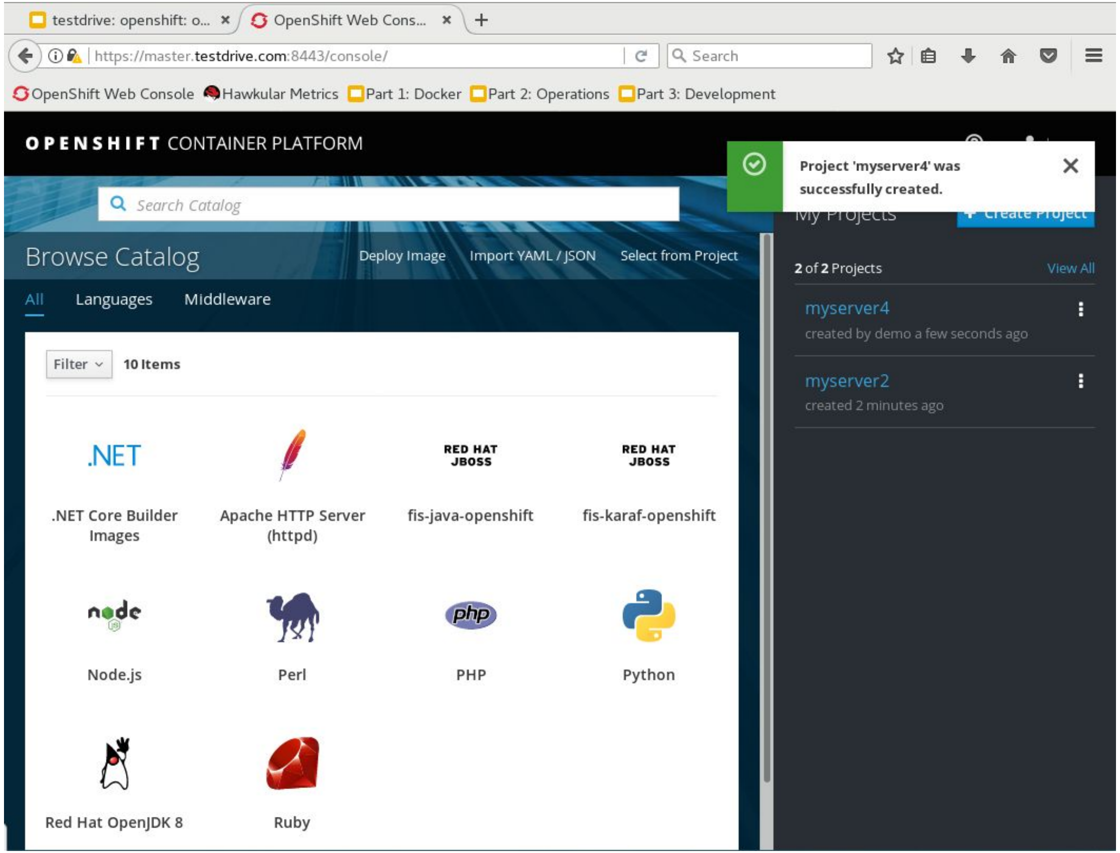
My Project User2

Description

A short description.

Create a new project called **my-project-userID** with display name **“My Project UserID”**

First Steps



testdrive: openshift: o... × OpenShift Web Cons... ×

https://master.testdrive.com:8443/console/

OpenShift Web Console Hawkular Metrics Part 1: Docker Part 2: Operations Part 3: Development

OPENSHIFT CONTAINER PLATFORM

Project 'myserver4' was successfully created.

Search Catalog

Browse Catalog Deploy Image Import YAML / JSON Select from Project

All Languages Middleware

Filter 10 Items

.NET Core Builder Images

Apache HTTP Server (httpd)

fis-java-openshift

fis-karaf-openshift

Node.js

Perl

PHP

Python

Red Hat OpenJDK 8

Ruby

My Projects Create Project

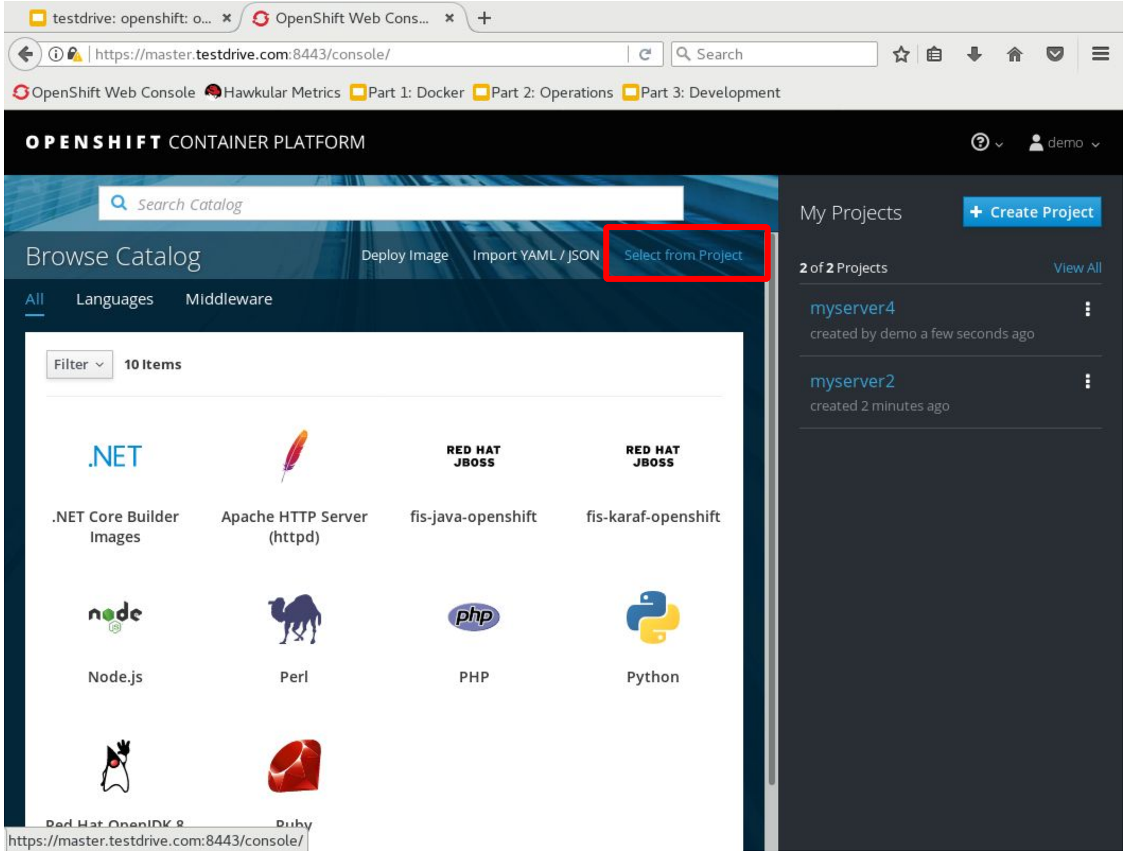
2 of 2 Projects View All

myserver4
created by demo a few seconds ago

myserver2
created 2 minutes ago

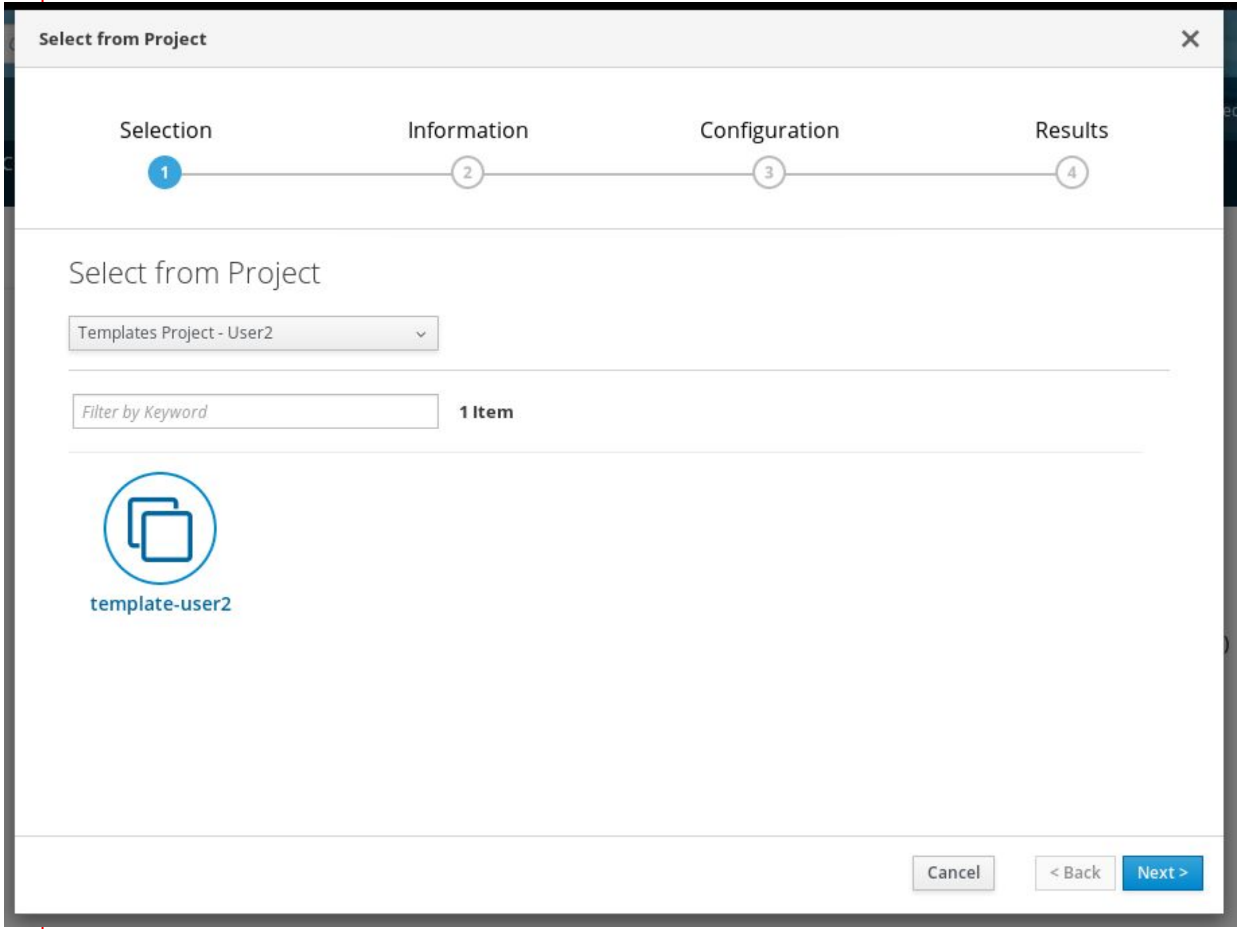
The Web Console will indicate the project was successfully created.

First Steps



Now select the "`Select from Project`" option

First Steps



Select from Project

Select from Project

Templates Project - User2

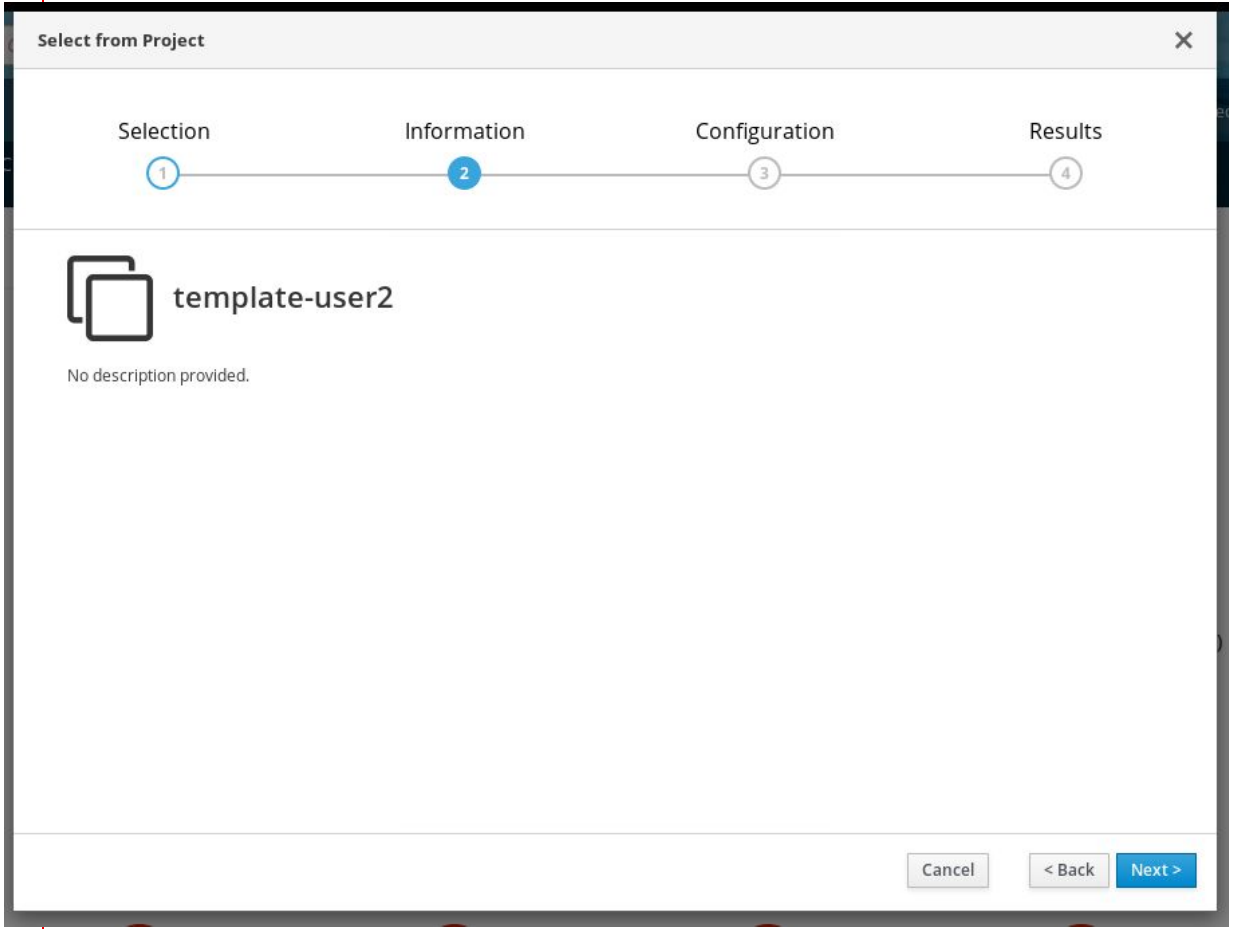
Filter by Keyword

1 Item

template-user2

And we're going to create a new application on Project **`my-project-userID`**, based on previously created **`templates-project-userID`**

First Steps



Select from Project

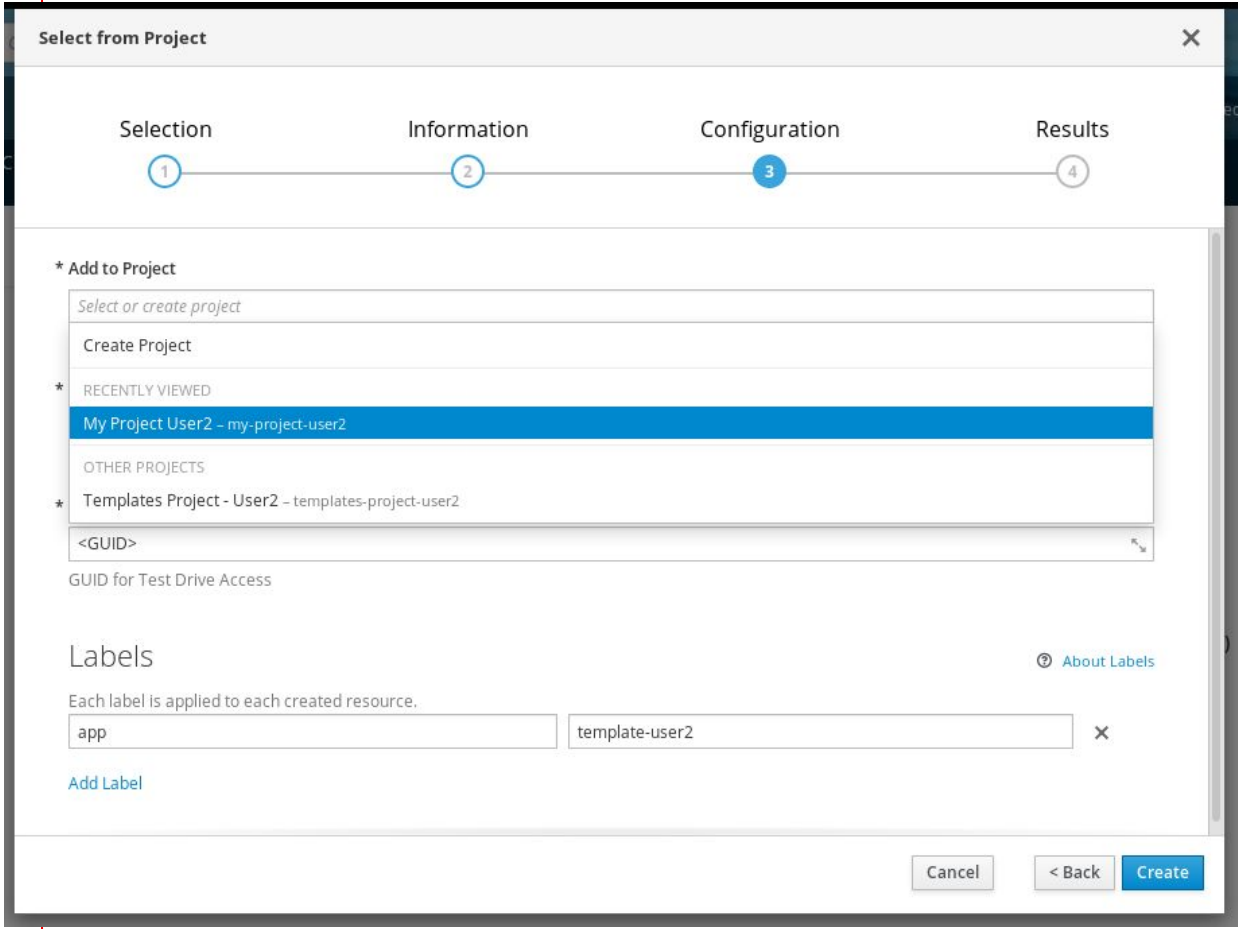
Selection · Information · Configuration · Results

template-user2

No description provided.

It's indicated that on Project `template_userID`, there is this template available.

First Steps



* Add to Project

Select or create project

Create Project

RECENTLY VIEWED

My Project User2 – my-project-user2

OTHER PROJECTS

Templates Project - User2 – templates-project-user2

<GUID>

GUID for Test Drive Access

Labels

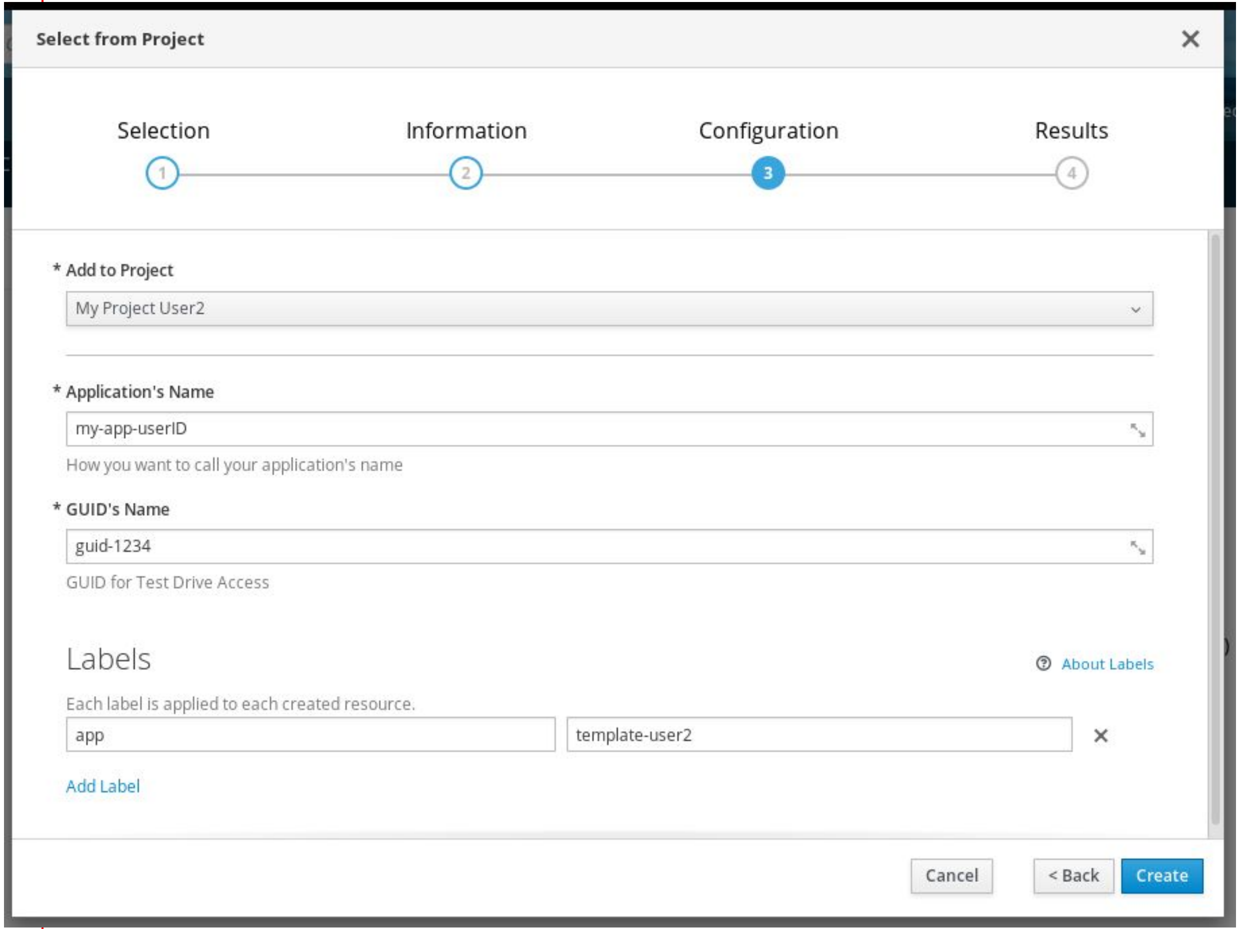
About Labels

Each label is applied to each created resource.

app: template-user2

Now, we're going to select the project we want to insert our template into: **`my-project-userID`**

First Steps



Select from Project

* Add to Project

My Project User2

* Application's Name

my-app-userID

How you want to call your application's name

* GUID's Name

guid-1234

GUID for Test Drive Access

Labels

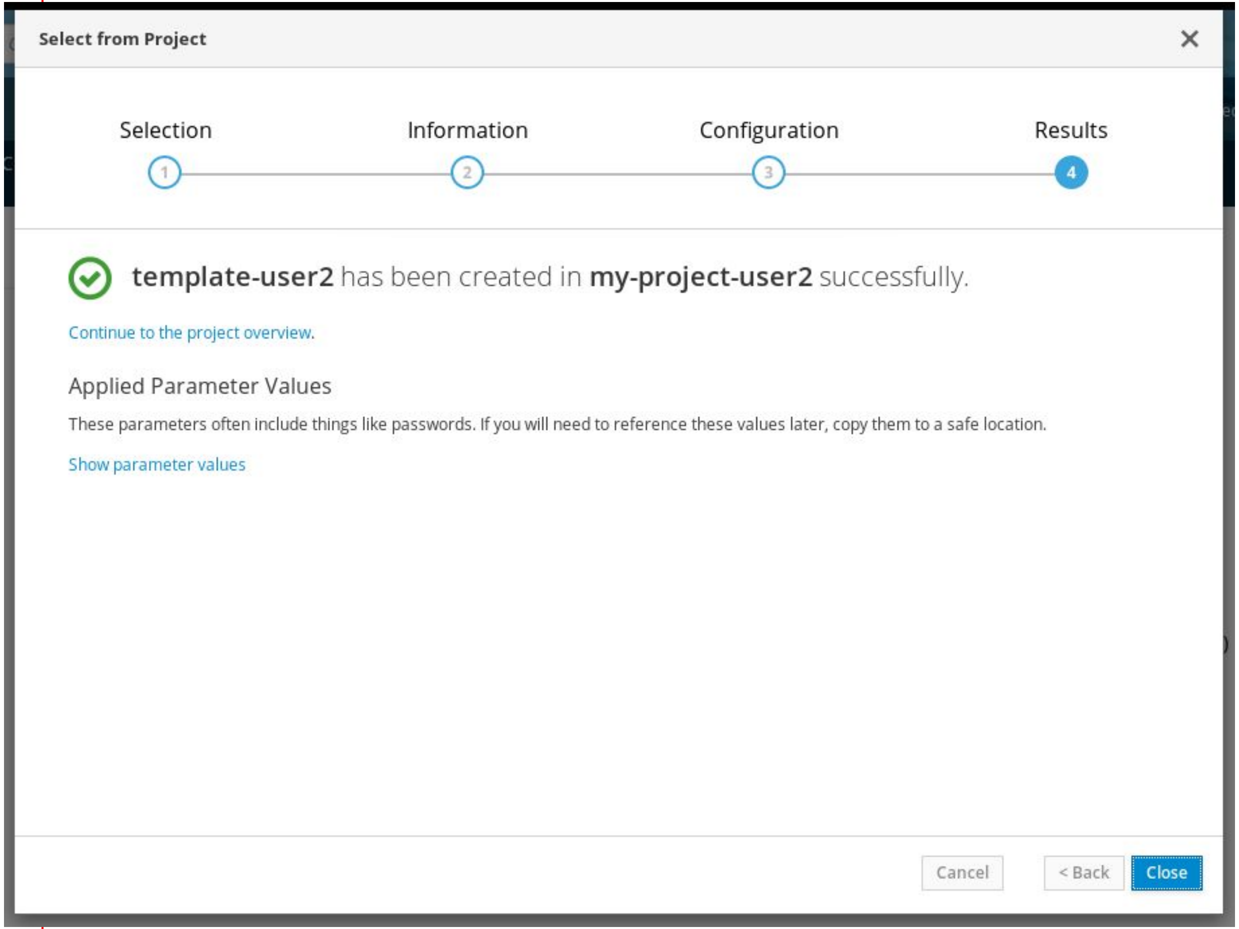
Each label is applied to each created resource.

app	template-user2

Because our template there is somes **PARAMETERS** named **APPLICATION_NAME**, and **GUID** we're going to insert some values: `my-app-userID`, and validate the **`GUID`**.

Red Hat

First Steps



template-user2 has been created in **my-project-user2** successfully.

Continue to the project overview.

Applied Parameter Values

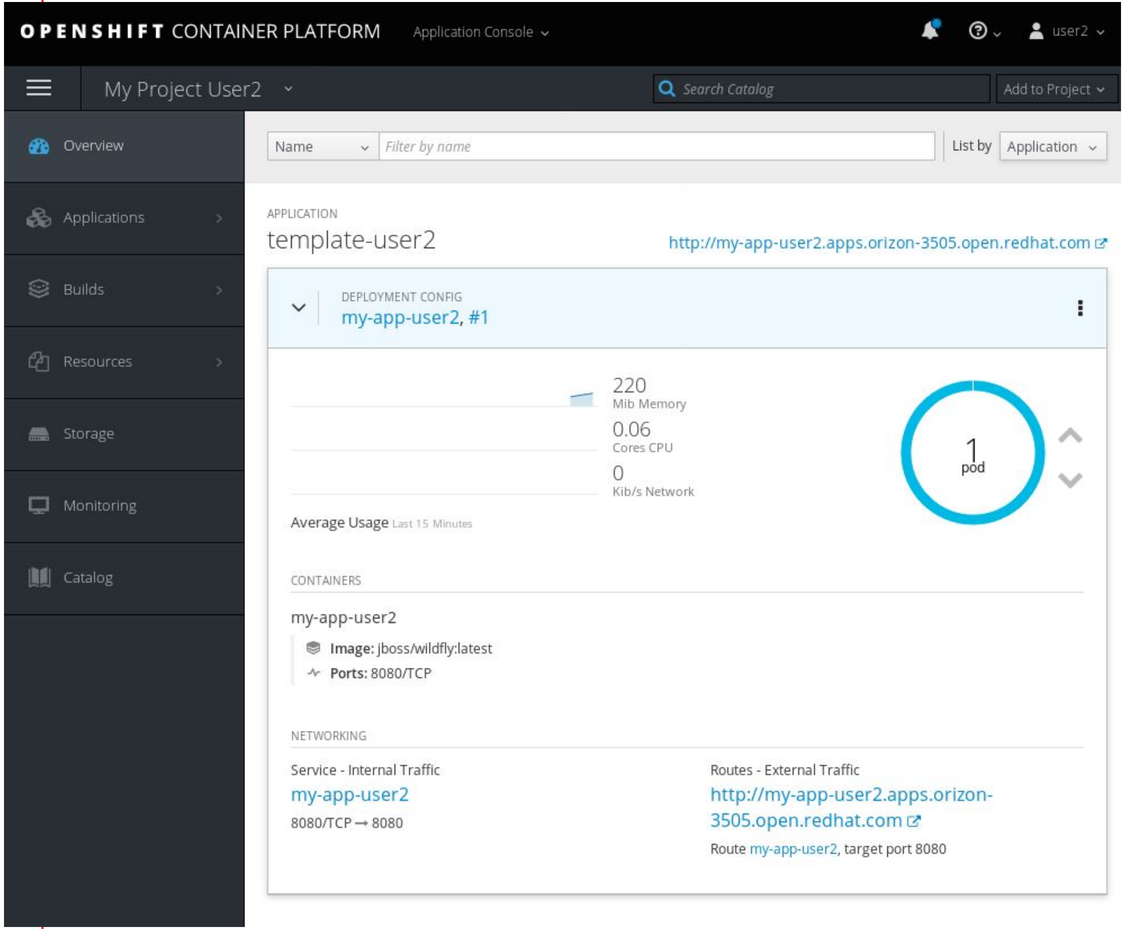
These parameters often include things like passwords. If you will need to reference these values later, copy them to a safe location.

Show parameter values

Finally, we were able to create the whole set of resources at once using OpenShift's WebConsole.

Click: **Continue to the project overview**

First Steps



OPENSHIFT CONTAINER PLATFORM Application Console ⌄ user2 ⌄

My Project User2 ⌄ Search Catalog Add to Project ⌄

- Overview
- Applications ›
- Builds ›
- Resources ›
- Storage
- Monitoring
- Catalog

Name ⌄ Filter by name List by Application ⌄

APPLICATION

template-user2 http://my-app-user2.apps.orizon-3505.open.redhat.com

DEPLOYMENT CONFIG

my-app-user2, #1

220 Mib Memory

0.06 Cores CPU

0 Kib/s Network

Average Usage Last 15 Minutes

1 pod

CONTAINERS

my-app-user2

Image: jboss/wildfly:latest

Ports: 8080/TCP

NETWORKING

Service - Internal Traffic

my-app-user2

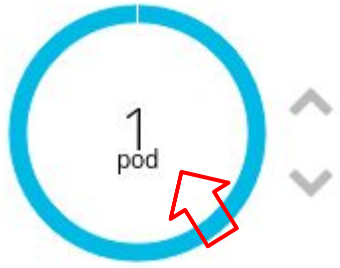
8080/TCP → 8080

Routes - External Traffic

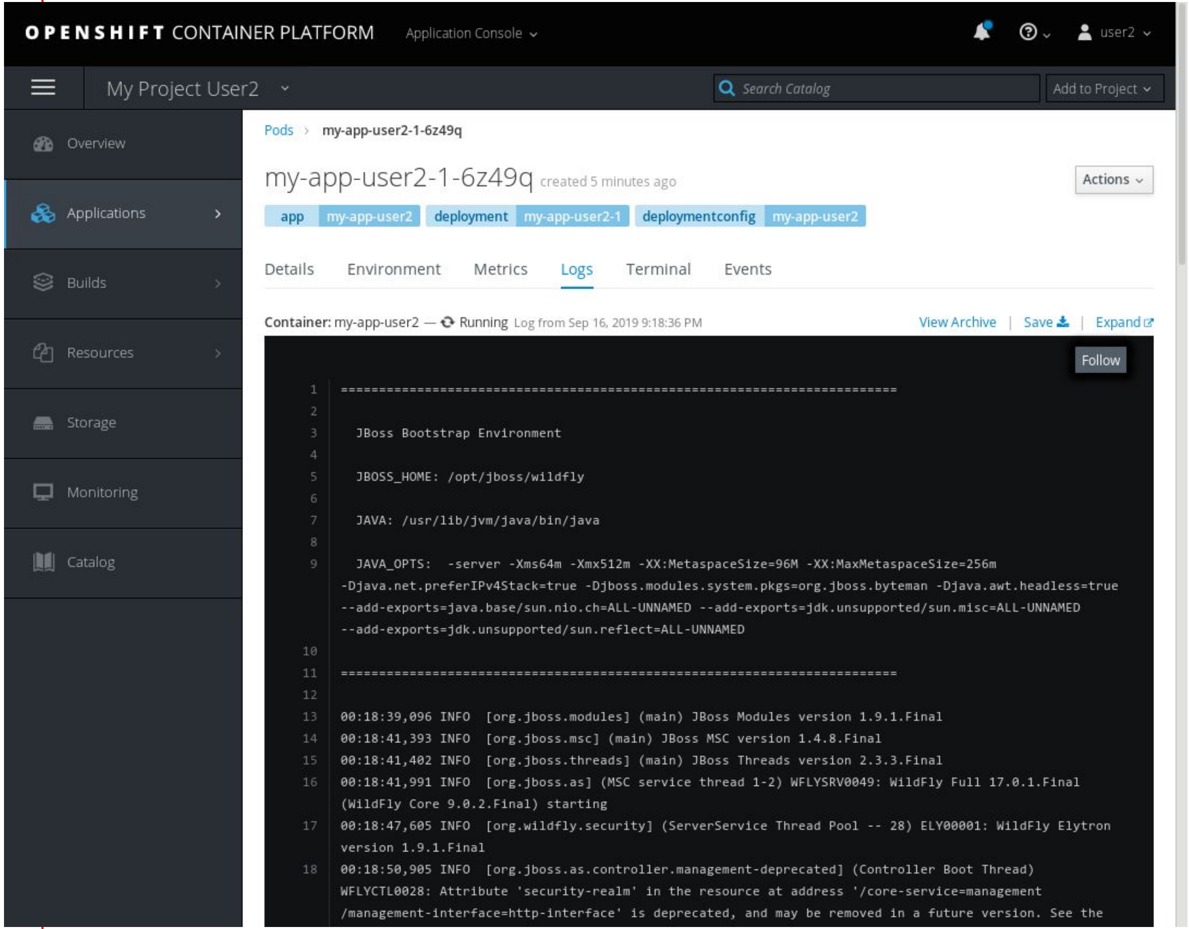
http://my-app-user2.apps.orizon-3505.open.redhat.com

Route my-app-user2, target port 8080

Let's see the logs of this application, by clicking into the middle of the pod.



First Steps



my-app-user2-1-6z49q created 5 minutes ago

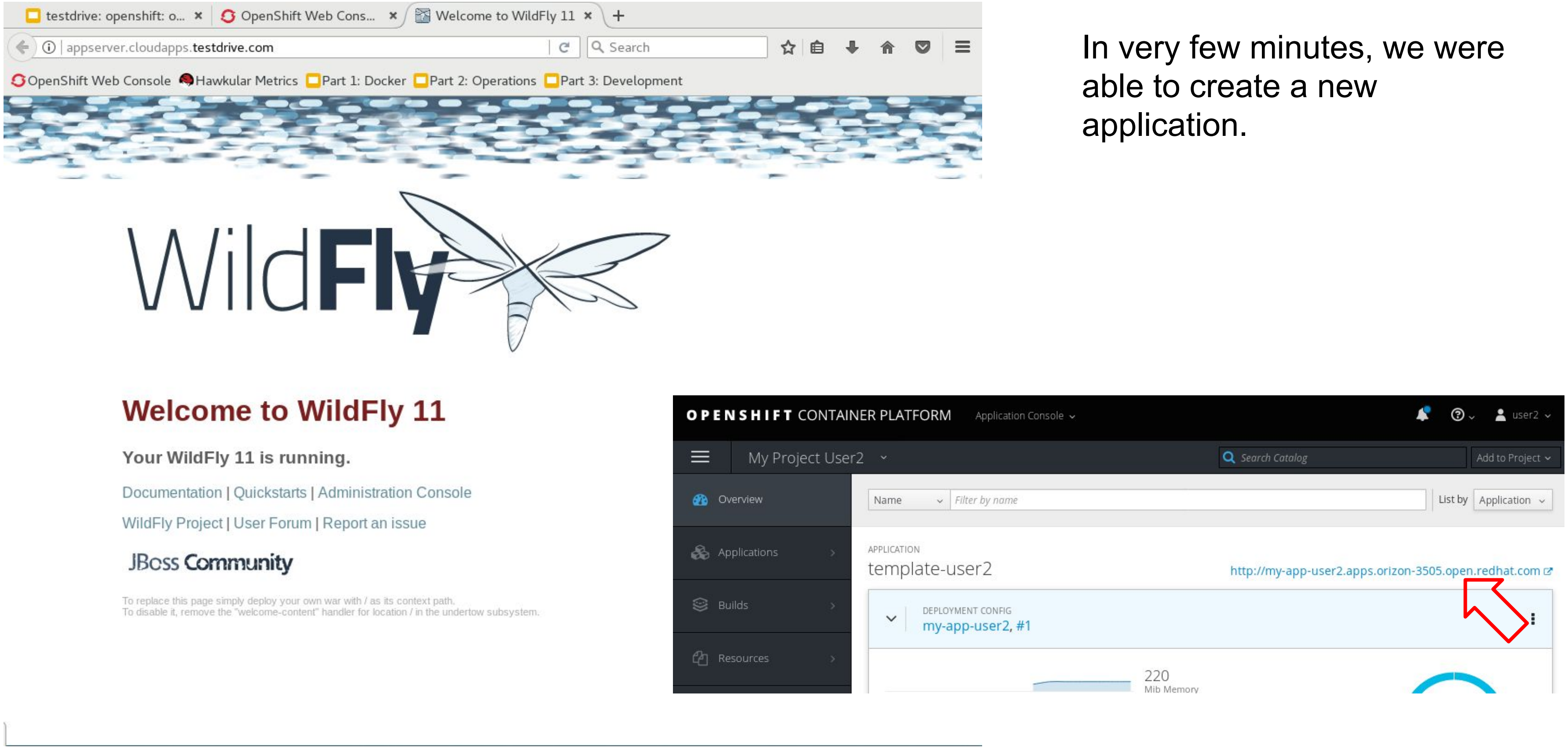
Container: my-app-user2 — Running Log from Sep 16, 2019 9:18:36 PM

```
=========================================================================

  JBoss Bootstrap Environment

  JBOSS_HOME: /opt/jboss/wildfly

  JAVA: /usr/lib/jvm/java/bin/java

  JAVA_OPTS:  -server -Xms64m -Xmx512m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m
-Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true
--add-exports=java.base/sun.nio.ch=ALL-UNNAMED --add-exports=jdk.unsupported/sun.misc=ALL-UNNAMED
--add-exports=jdk.unsupported/sun.reflect=ALL-UNNAMED

=========================================================================

00:18:39,096 INFO  [org.jboss.modules] (main) JBoss Modules version 1.9.1.Final
00:18:41,393 INFO  [org.jboss.msc] (main) JBoss MSC version 1.4.8.Final
00:18:41,402 INFO  [org.jboss.threads] (main) JBoss Threads version 2.3.3.Final
00:18:41,991 INFO  [org.jboss.as] (MSC service thread 1-2) WFLYSRV0049: WildFly Full 17.0.1.Final
(WildFly Core 9.0.2.Final) starting
00:18:47,605 INFO  [org.wildfly.security] (ServerService Thread Pool -- 28) ELY00001: WildFly Elytron
version 1.9.1.Final
00:18:50,905 INFO  [org.jboss.as.controller.management-deprecated] (Controller Boot Thread)
WFLYCTL0028: Attribute 'security-realm' in the resource at address '/core-service=management
/management-interface=http-interface' is deprecated, and may be removed in a future version. See the
```

And selecting the option "Logs"

First Steps

In very few minutes, we were able to create a new application.



testdrive: openshift: o... | OpenShift Web Cons... | Welcome to WildFly 11

appserver.cloudapps.testdrive.com

Welcome to WildFly 11

Your WildFly 11 is running.

Documentation | Quickstarts | Administration Console

WildFly Project | User Forum | Report an issue

JBoss Community

To replace this page simply deploy your own war with / as its context path.
To disable it, remove the "welcome-content" handler for location / in the undertow subsystem.

APPLICATION
template-user2

http://my-app-user2.apps.orizon-3505.open.redhat.com

DEPLOYMENT CONFIG
my-app-user2, #1

220
Mib Memory

Test Drive: OpenShift@Ops

Persistent Volumes

Test Drive: OpenShift@Ops

Persistent Volumes

at the end of process, those are the available persistence storage available for our cluster.

```
[root@bastion ~]# oc get pv
NAME      CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS      CLAIM     REASON    AGE
local1    1Gi        RWO,RWX       Recycle         Available                       2m
local10   1Gi        RWO,RWX       Recycle         Available                       2m
local11   5Gi        RWO,RWX       Recycle         Available                       1m
local12   5Gi        RWO,RWX       Recycle         Available                       1m
local13   5Gi        RWO,RWX       Recycle         Available                       1m
local14   5Gi        RWO,RWX       Recycle         Available                       1m
local15   5Gi        RWO,RWX       Recycle         Available                       1m
local16   5Gi        RWO,RWX       Recycle         Available                       1m
...
local6    1Gi        RWO,RWX       Recycle         Available                       2m
local7    1Gi        RWO,RWX       Recycle         Available                       2m
local8    1Gi        RWO,RWX       Recycle         Available                       2m
local9    1Gi        RWO,RWX       Recycle         Available                       2m
```

TEST

Please create an application that needs persistence (such as database) and then check the situation of Persistent Volumes by typing

```
[root@bastion ~]# oc get pv --all-namespaces

[root@bastion ~]# oc get pvc --all-namespaces
```

Thank You!

Test Drive: OpenShift@Ops

in linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

f facebook.com/redhatinc

twitter.com/RedHat

Red Hat

Feedback

in linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

f facebook.com/redhatinc

twitter.com/RedHat

http://bit.ly/redhat-testdrive-feedback

Red Hat