Microsoft

OPENSHIFT on Azure

Khaled Elbedri
Technology Solutions Professional - Global
Black Belt, Open Source Software
9/27/2017

TABLE OF CONTENTS

OBJECTIVES AND PRE-REQUISITES

This document describes the steps necessary to deploy and manage Red Hat OpenShift container platform, and CloudForms on Azure. The lab is based on OpenShift version 3.5 and CloudForms version 4.1.

You will need an active Red Hat subscription and a valid Azure account to perform the lab instructions.

- Create your free azure account (https://azure.microsoft.com/en-us/free/), today.
- Request a free 30 days evaluation from RedHat (https://access.redhat.com/products/red-hat-openshift-container-platform/evaluation), (https://access.redhat.com/solutions/411973)
- You need to have an account on GitHub, If you don't have one create a free account (https://github.com/).

NB: As an alternative to OpenShift enterprise, you can deploy the community version OpenShift.org. And as an alternative to CloudForms, you can deploy the community version ManageIQ. The community edition represents a test bed incubator and upstream to the enterprise one. All OpenShift container platform features are also available in OpenShift.org.

To perform the lab using OpenShift.org, follow the steps at https://github.com/Microsoft/openshift-origin and skip Exercise-1.

NB: If you want to use your corporate Azure account, request a resource group with owner access control from your Azure admin.

NB: Even though, you can bring your own Red Hat subscription for OpenShift, the deployment template in this lab will provision the master and compute nodes from RHEL images, pay as you go. You will incur extra charges not covered by the Azure free tier. Amount varies depending on the number and size of the vm nodes.

The Lab exercises cover:

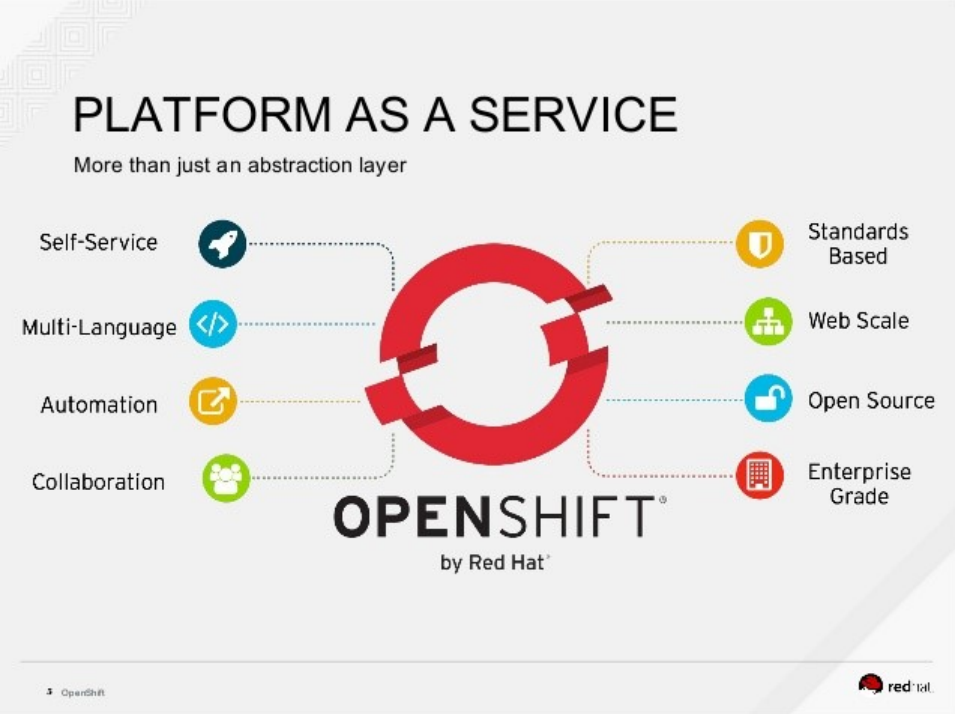
- Introduction to OpenShift3
- Deployment of OpenShit on Azure
- Creating and managing OpenShift projects on azure
- Creating and managing OpenShift applications on Azure
- Automating builds with Linux Containers on Azure.
- Mini project: Jboss EAP application
- Integration of OpenShift with Azure OMS
- Installation and introduction to CloudForms

INTRODUCTION TO OPENSHIFT

OpenShift containers platform is Red Hat's Platform-as-a-Service (PaaS) on top of Kubernetes. It allows developers to quickly develop, host, and scale applications in a cloud environment.

Microsoft and Red Hat have signed a partnership that includes support to run Red Hat OpenShift on Microsoft Azure and Azure Stack.

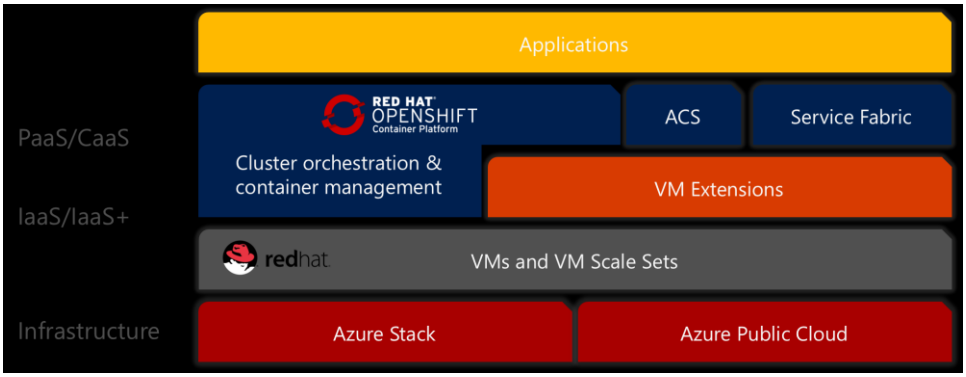
OpenShift offers multiple access modes including: developer CLI, admin CLI, web console and IDE plugins. Click2cloud is a plugin that allows Visual studio to deploy code to OpenShift, directly.



EXERCISE-1: DEPLOY OPENSHIFT ON AZURE

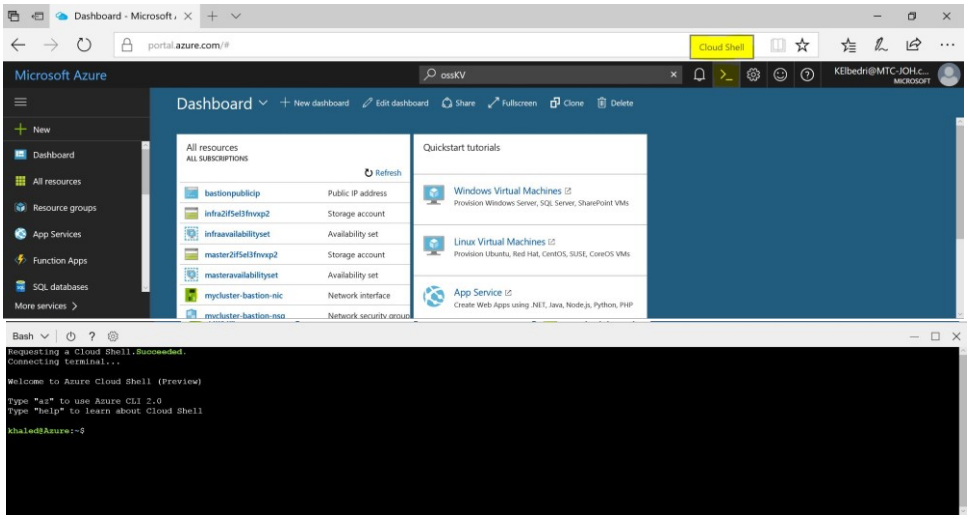
OpenShift, leverages multiple Azure services such as VM *extensions*, Azure disks, and *Key vaults*… to provide an Enterprise grade offering for customers who would like to containerize and manage their applications, without investing long time and hard effort configuring and integrating various tools.

OpenShift offers another alternative to multiple CaaS (container as a service) solutions available on Azure, such as *Azure container service, Azure service fabric* and *Pivotal* from *CloudFoundry*…



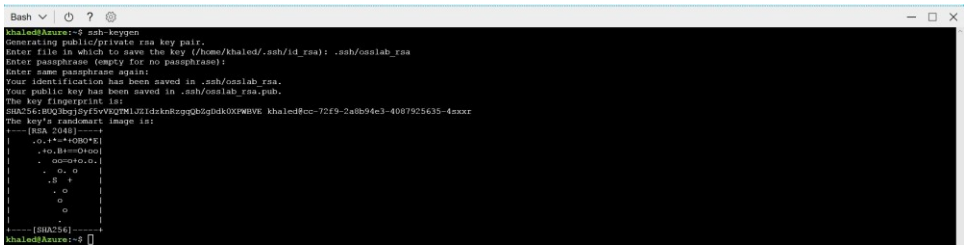
OpenShift container platform is available as an Azure Resource Manager solution at https://github.com/Microsoft/openshift-container-platform.

1. Login to Azure portal and start a new Bash Cloud shell session.



2. From the open terminal, create a new ssh key pair with the name “osslab_rsa” and save it under .ssh directory

```
$ ssh-keygen
```



3. Use the Azure CLI v2 to create a new resource group to host the lab resources

```
$ az group create -n ossdemo -l 'West Europe'
```

```
khaled@Azure:~$ az group create -n ossdemo -l 'West Europe'
{
  "id": "/subscriptions/f3a5dfdb-e863-40d9-b23c-752b886c0260/resourceGroups/ossdemo",
  "location": "westeurope",
  "managedBy": null,
  "name": "ossdemo",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

4. Create a Key Vault and add your *ssh* private key, created in the previous step.

```
$ az keyvault create -n ossKV -g ossdemo -l 'West
Europe' --enabled-for-template-deployment true
```


```
$ az keyvault secret set --vault-name ossKV -n ossSecret
--file ~/.ssh/osslab_rsa
```

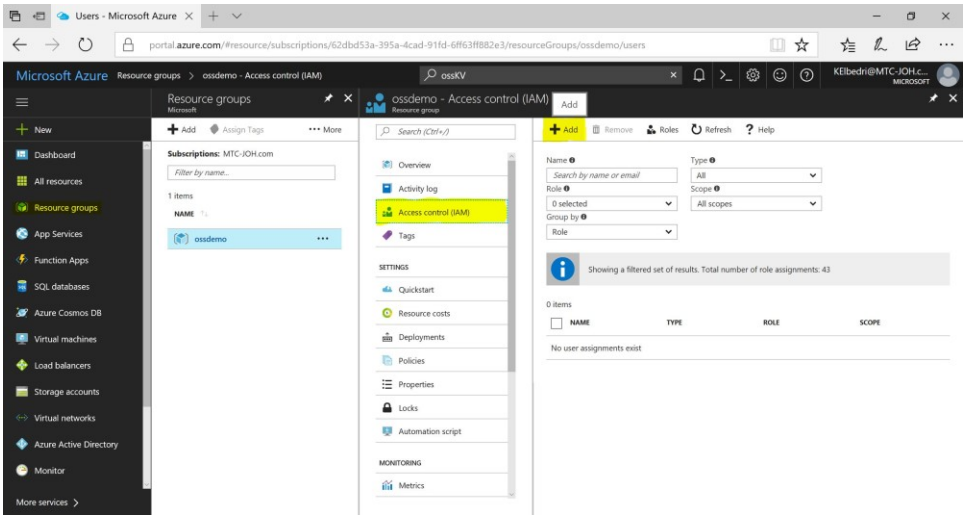
The deployment of OpenShift relies on Ansible scripts that should be configured for Azure as the cloud provider. During and post-installation, OpenShift requires access to some Azure resources, like provisioning an Azure managed disk for persistence storage backend. When you have an application that needs to access or modify resources, you must set up an Azure Active Directory (AD) application and assign the required permissions to it. The service principal object defines the policy and permissions for an application's use in a specific tenant, providing the basis for a security principal to represent the application at run-time.

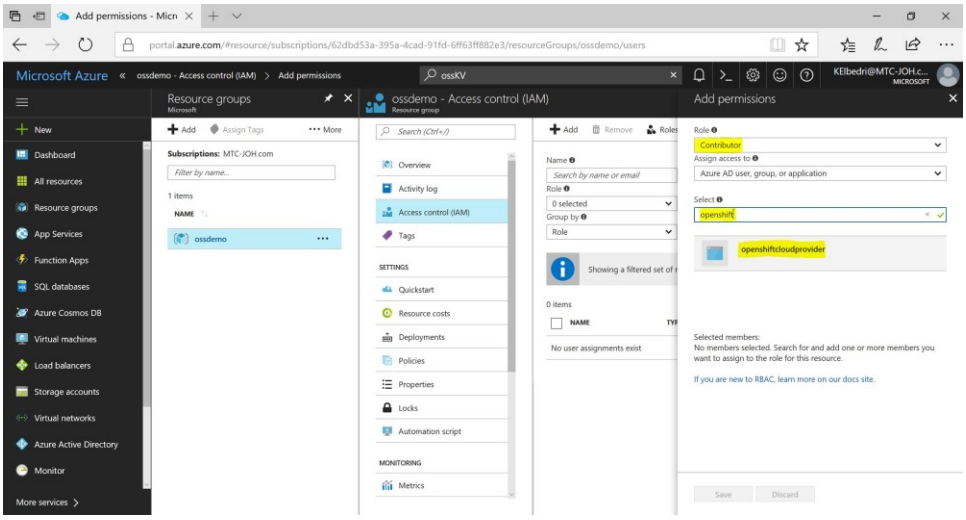
5. Create an Azure Active Directory Service Principal and choose a different password. Copy the resource group scope from step number 3.

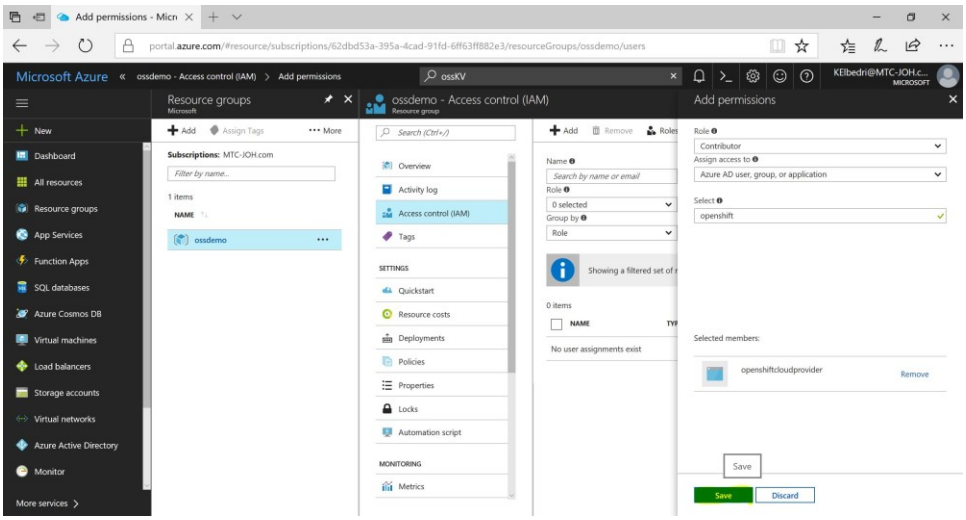
```
$ az ad sp create-for-rbac -n openshiftcloudprovider --
password changeMePassword --role contributor --scopes
/subscriptions/f3a5dfdb-e863-40d9-b23c-
752b886c0260/resourceGroups/ossdemo
```

```
khaled@Azure:~$ az ad sp create-for-rbac -n osscloudprovider --password changeMePassword --role contributor --scopes /subscriptions/f3a5dfdb-e863-40d9-b23c-752b886c0260/resourceGroups/
ossdemo
{
  "appId": "83fc34ee-5be2-4c89-92fd-496c620c6f6e",
  "displayName": "osscloudprovider",
  "name": "http://osscloudprovider",
  "password": "changeMePassword",
  "tenant": "72f988bf-86f1-41af-91ab-2d7cd011db47"
}
```

6. Now, go to the Azure portal and assign the required permissions to the service principal “osscloudprovider” on the resource group “ossdemo”.







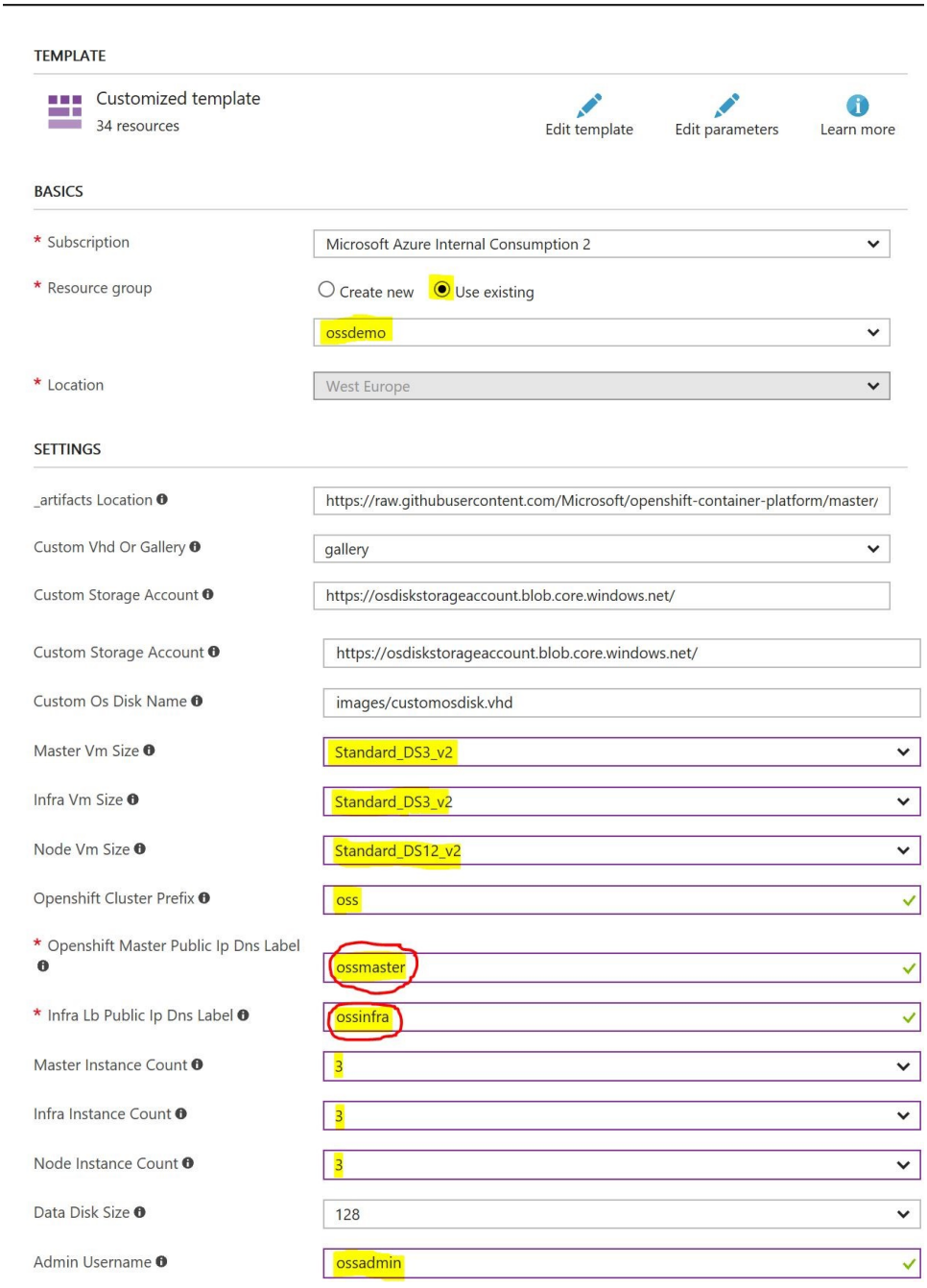
7. Note the application id of your service principal.

```
$ az ad sp show --id http://openshiftcloudprovider
```

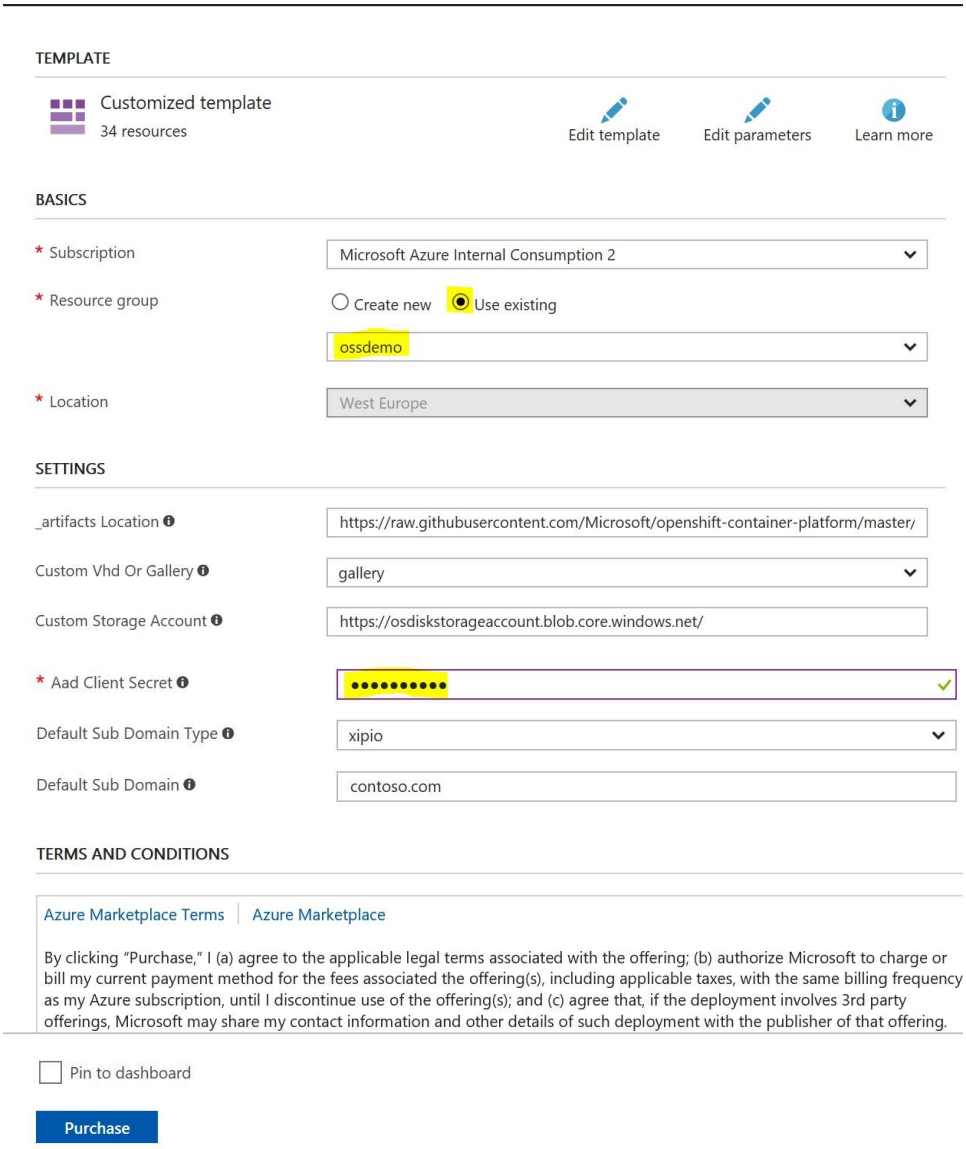
8. Use the following Azure resource manager solutions template to deploy your OpenShift environment: https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FMicrosoft%2Fopenshift-container-platform%2Fmaster%2Fazuredeploy.json

   - At this stage we will need Red Hat Network or Red Hat satellite credentials to register RHEL vms with RedHat and get access to the required software channels during the deployment. More specifically, we will need:
     - Red Hat Network/Satellite name
     - The associated password or activation key
   - Pool Id provides access to the required software channels for Openshift and Cloudforms. PoolId can be listed by invoking the command 'subscription-manager list -available' from a registered RHEL system. Contact your Red Hat admin or Red Hat support if you are missing information
   - For high availability consideration, we are deploying 3 vms for each type (master, infra and node). If you want to deploy the lab with minimal cost, you can reduce the number of vms to one per each. Also, you can scaledown the vm families, but preferably stick to vms with SSD disks for faster deployment.

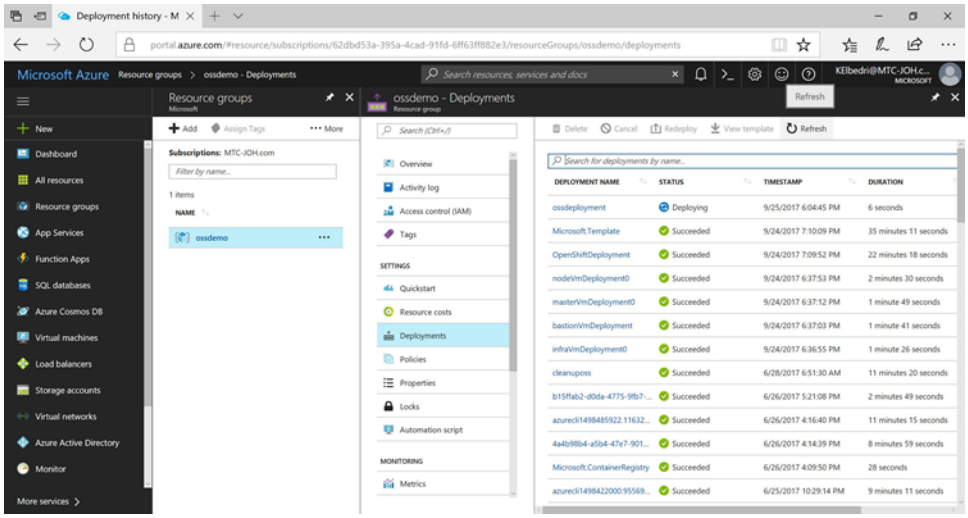
- The master and infra load balancer DNSs (in red circles) should be globally unique. Choose your own names.




* Openshift Password
Enable Metrics
true
Enable Logging
true
Enable Cockpit
true
Rhsm Username Password Or Activation Key
usernamepassword
* Rhsm Username Or Org Id
* Rhsm Password Or Activation Key
* Rhsm Pool Id
8a85f9819be87e78015be89906382ad7
* Ssh Public Key
* Key Vault Resource Group
ossdemo
* Key Vault Name
ossKV
* Key Vault Secret
* Aad Client Id
your service principal appId
* Openshift Password
Enable Metrics
true
Enable Logging
true
Enable Cockpit
true
Rhsm Username Password Or Activation Key
usernamepassword
* Rhsm Username Or Org Id
* Rhsm Password Or Activation Key
* Rhsm Pool Id
8a85f9819be87e78015be89906382ad7
* Ssh Public Key
* Key Vault Resource Group
ossdemo
* Key Vault Name
ossKV
* Key Vault Secret
* Aad Client Id
your service principal appId



9. From the Azure portal, go to your resource group "ossdemo", track the progress of the deployment and make sure the deployment finishes, successfully.



The following diagram explains the physical architecture of the deployed cluster.

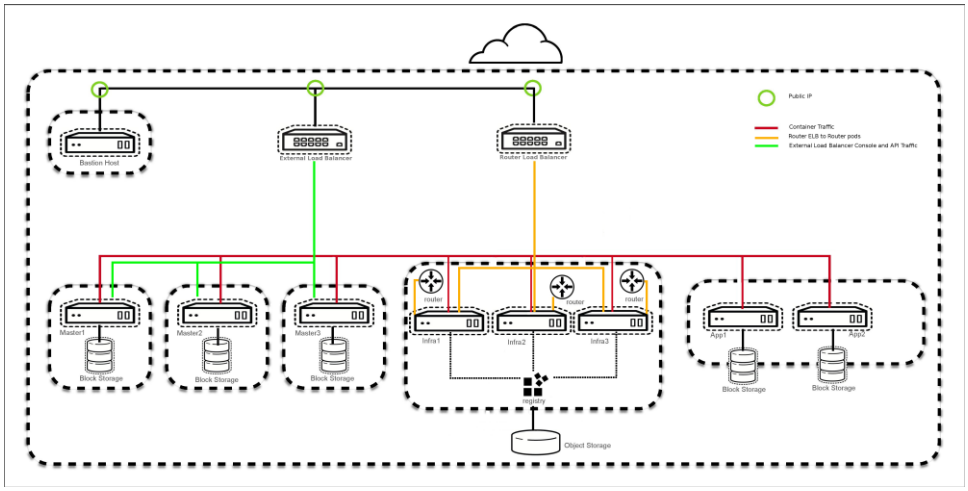


Figure 6: Deployment diagram

The bastion server implements mainly two distinct functions. One is that of a secure way to connect to all the nodes, and second that of the "installer" of the system. The bastion host is the only ingress point for SSH in the cluster from external entities. When connecting to the OpenShift Container Platform infrastructure, the bastion forwards the request to the appropriate server.

The next diagram, explains the role and tasks of the Openshift master/agents and the logical architecture of the solution.

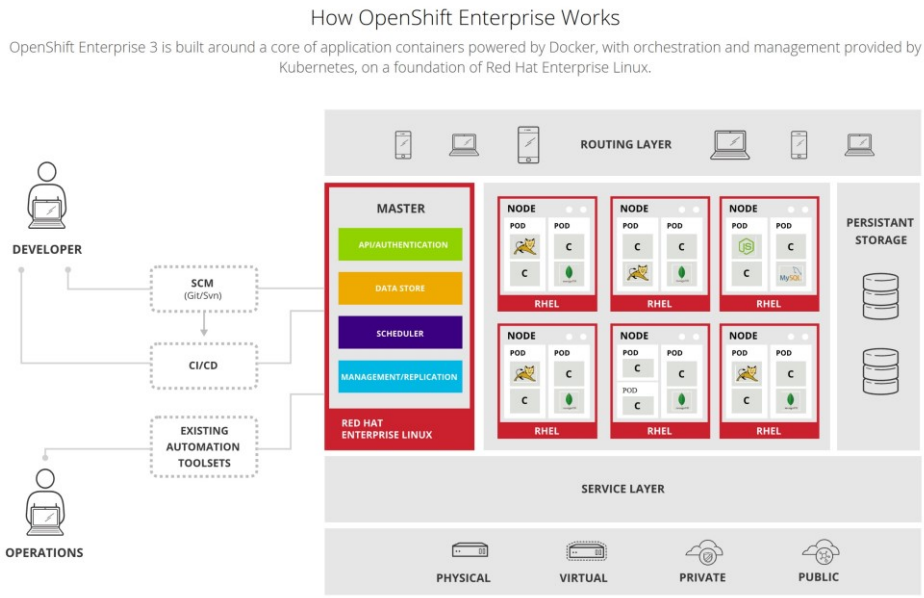


Figure 7: OpenShift logical architecture

EXERCISE-2: CREATE AND MANAGE PROJECTS

There are many ways to launch images within an OpenShift project. In this exercise we will focus on the graphical portal.

To create an *application*, you must first create a new *project*, then select an *InstantApp* template. From there, OpenShift begins the build process, and creates a new deployment.

1. Login to your *github* account, or create one if you didn't.
2. Browse to *openshift/ruby-ex* repository and fork it into your *github* account

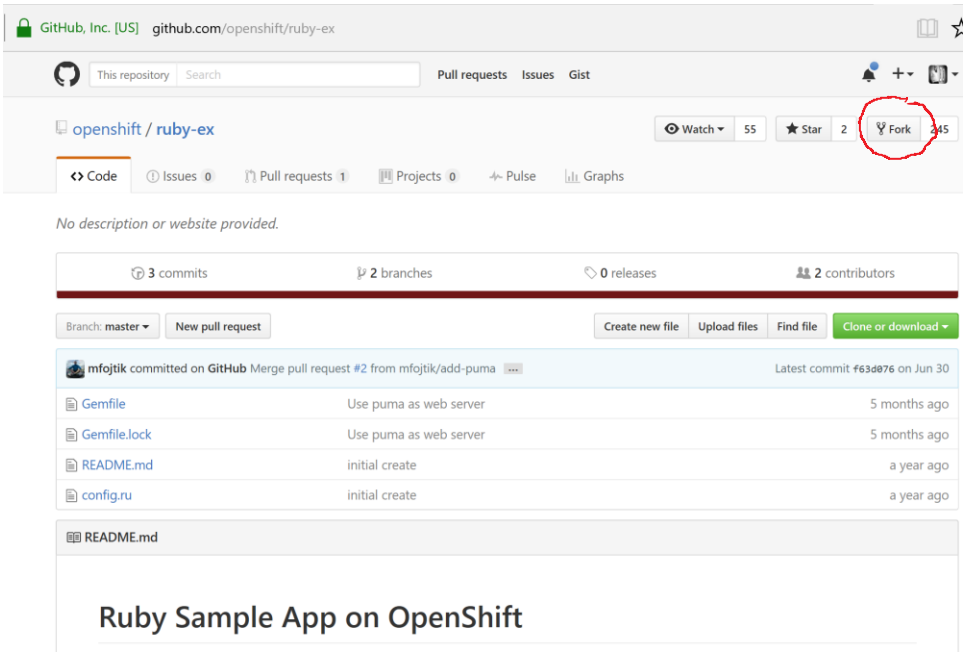
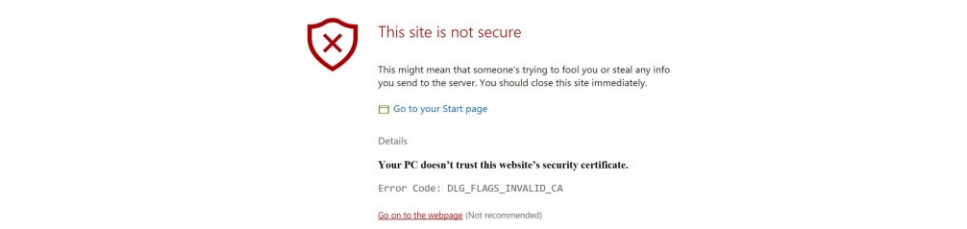
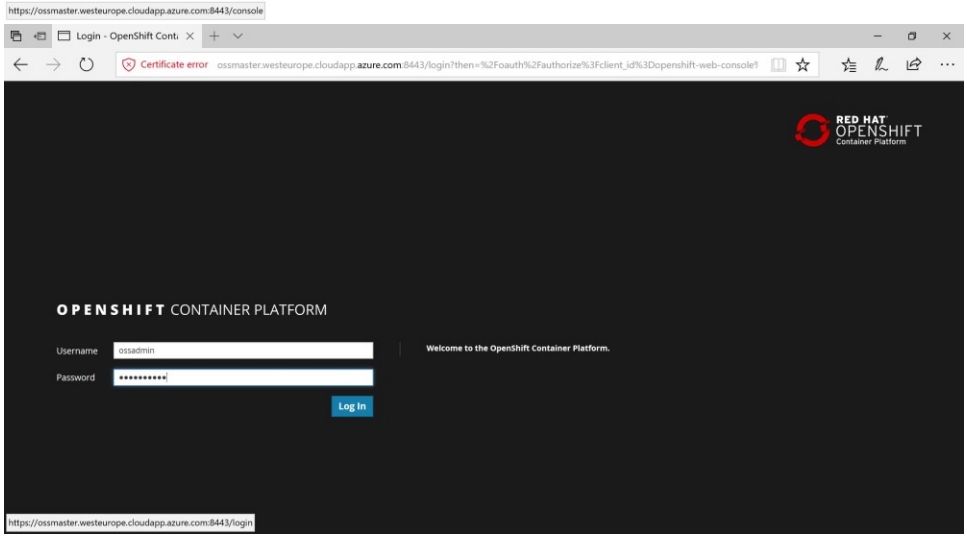


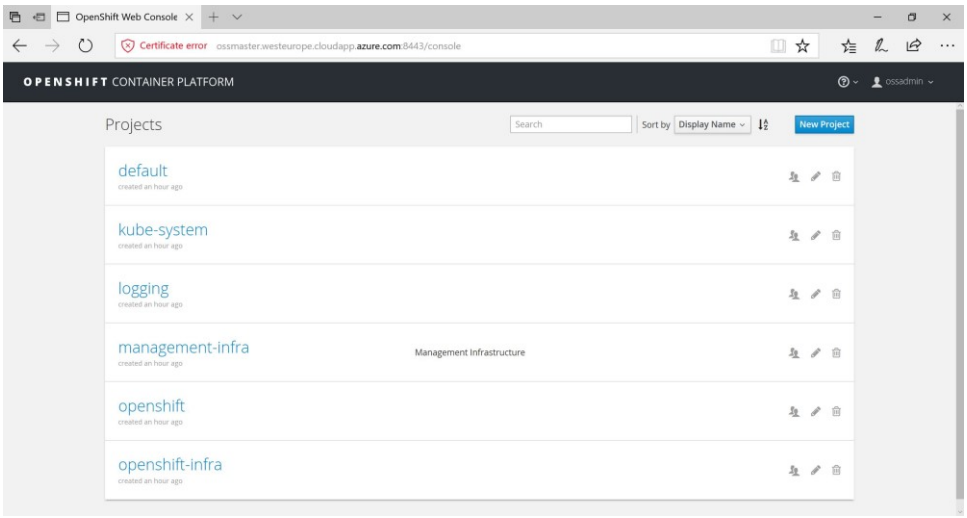
Figure 8: Fork ruby-ex sample application

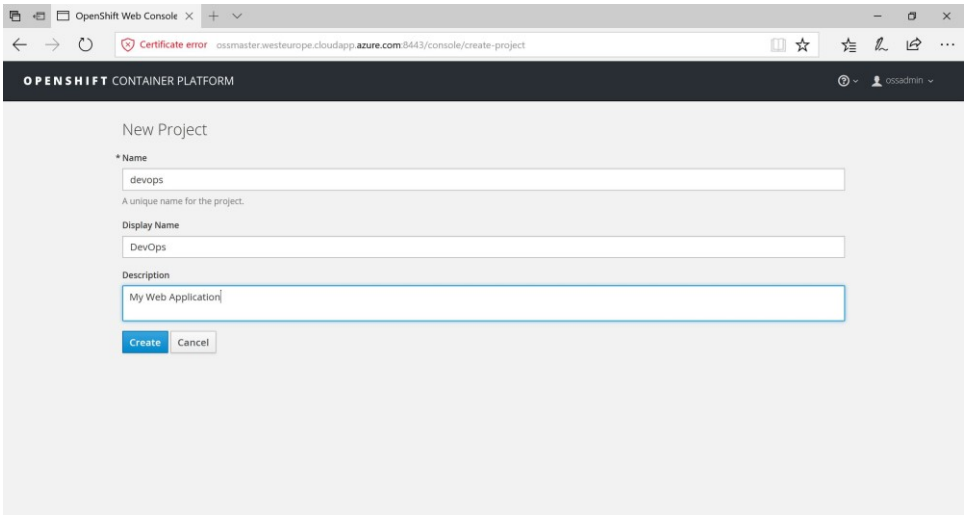
3. From your browser, visit the OpenShift web console at *https://FQDN-master-node:8443*. The web site, uses a self-signed certificate, so if prompted, continue and ignore the browser warning.
4. Log in using your username and password.





5. To create a new project, click **New Project**.
6. Type a unique name, display name, and description for the new project.
7. Click **Create**. The web console's welcome screen should start loading.

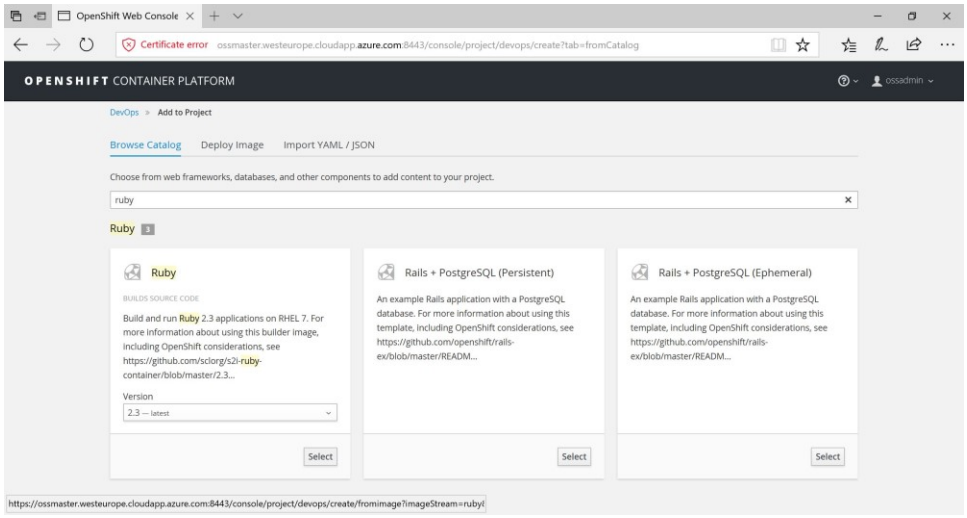

OpenShift Web Console
Certificate error
ossmaster.westeurope.cloudapp.azure.com:8443/console
OPENSHIFT CONTAINER PLATFORM
ossadmin
Projects
Search
Sort by
Display Name
New Project
default
created an hour ago
kube-system
created an hour ago
logging
created an hour ago
management-infra
created an hour ago
Management Infrastructure
openshift
created an hour ago
openshift-infra
created an hour ago


OpenShift Web Console
Certificate error
ossmaster.westeurope.cloudapp.azure.com:8443/console/create-project
OPENSHIFT CONTAINER PLATFORM
ossadmin
New Project
* Name
devops
A unique name for the project.
Display Name
DevOps
Description
My Web Application
Create
Cancel

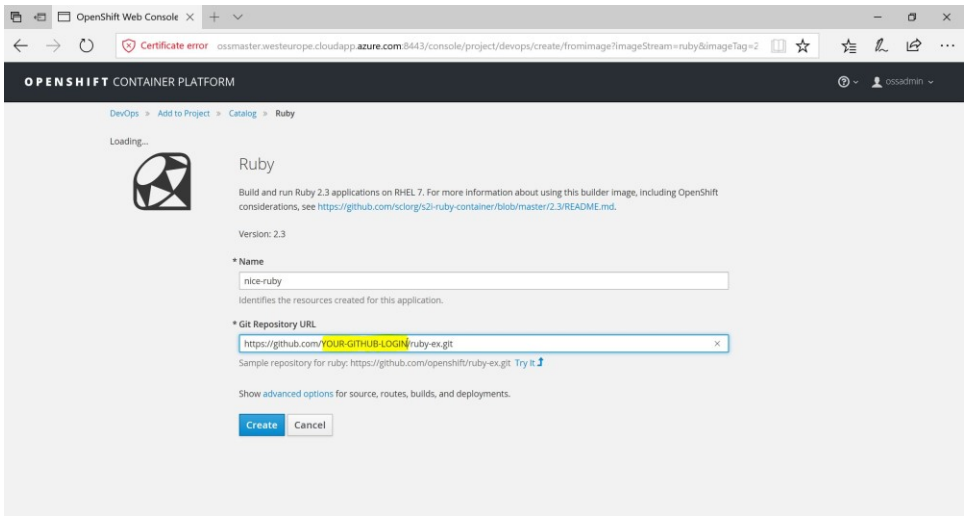
EXERCISE-3: CREATE AND MANAGE APPLICATIONS

The *Select Image* or *Template* page gives you the option to add an application to your project from a publicly accessible *Git* repository, or from a *template*:

1. If creating a new project did not automatically redirect you to the *Select Image or Template page*, you might need to click **Add to Project**.
2. Click **Browse**, then select
3. Click the **ruby:latest** builder image.

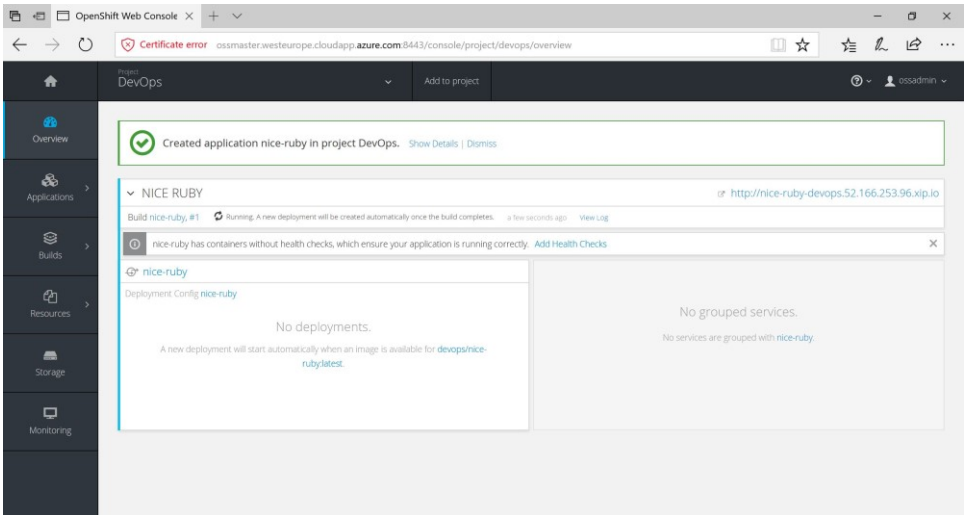


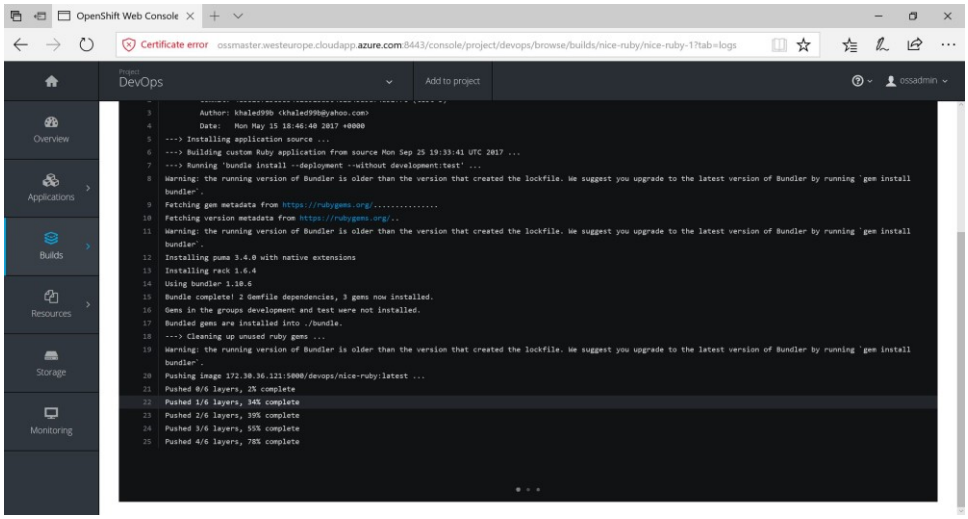
4. Type a **name** for your application, and specify the git repository URL you previously forked: https://github.com/<your_github_username>/ruby-ex.git.



5. Optionally, click **Show advanced routing, build, and deployment options**. Explore the build configuration and other options and note that this example application automatically creates a route, *webhook* trigger, and builds change triggers. A *webhook* is an HTTP call-back triggered by a specific event.
6. Click **Create**. Creating your application might take some time. note the *payload url,* we will use it later to set a *webhook* in your *github* repository.

7. You can follow along on the **Overview** page of the web console to see the new resources being created, and watch the progress of the build and deployment. Click on "view log", you will notice that Openshift is pulling the code of the application from Github and building the layers of the container image that will host the application. Once the build is complete, Openshift will start a new pod.





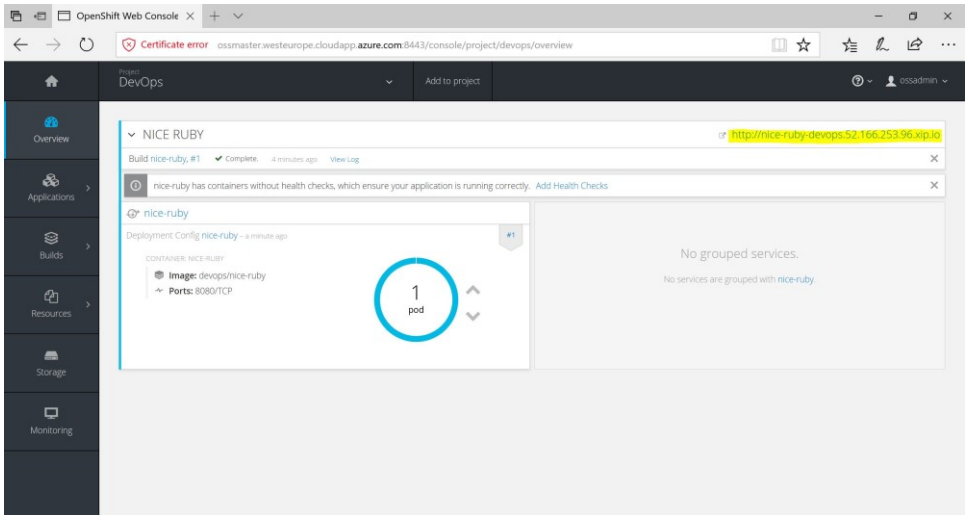
OpenShift leverages the Kubernetes concept of a pod, which is one or more containers deployed together on one host. A pod is the smallest compute unit that can be defined, deployed, and managed.

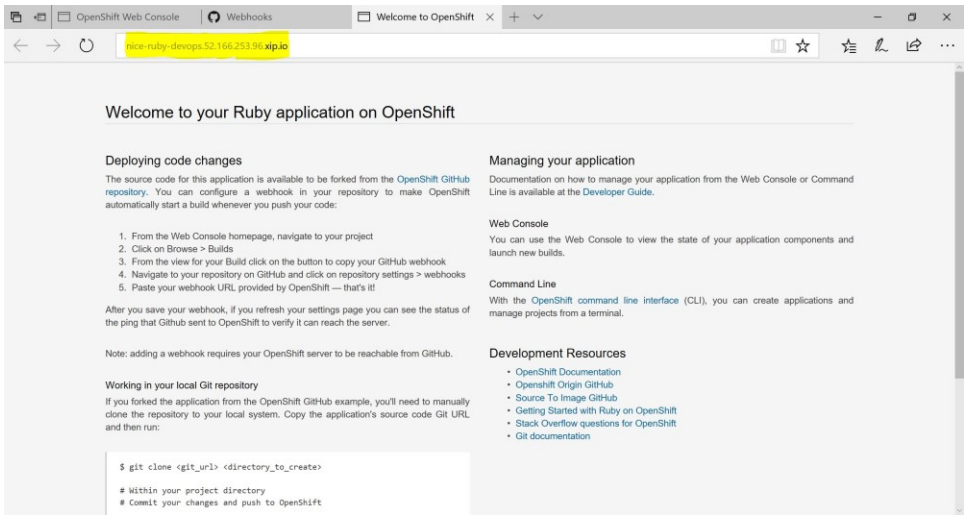
Pods are the rough equivalent of a machine instance (physical or virtual) to a container. Each pod is allocated its own internal IP address, therefore owning its entire port space. Containers within pods can share their local storage and networking.

While the Ruby *pod* is being created, its status is shown as pending. The Ruby *pod* then starts up and displays its newly-assigned IP address. When the Ruby *pod* is running, the build is complete.

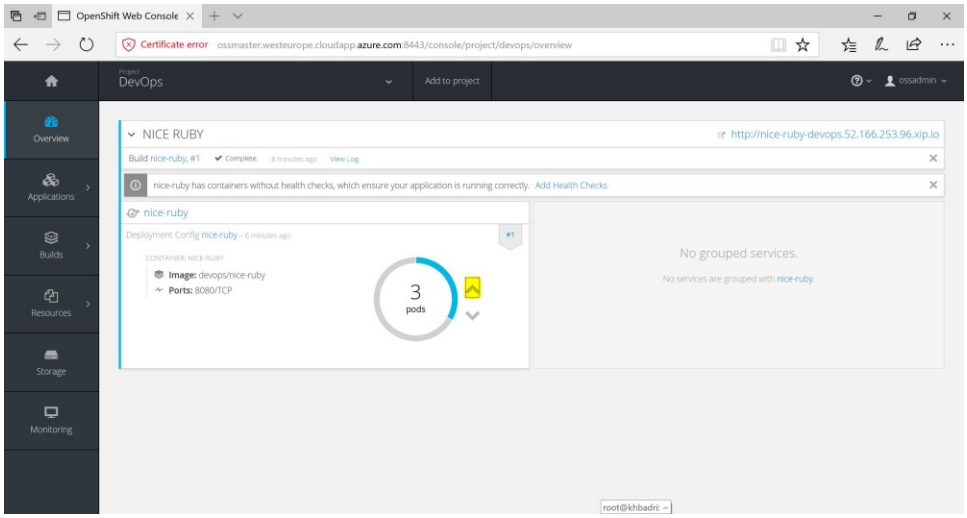
Pods have a lifecycle; they are defined, then they are assigned to run on a node, then they run until their container(s) exit or they are removed for some other reason. Pods, depending on policy and exit code, may be removed after exiting, or may be retained in order to enable access to the logs of their containers.

8. From the overview page, click the web address for the application in the upper right corner. Verify that the web application is up and available.

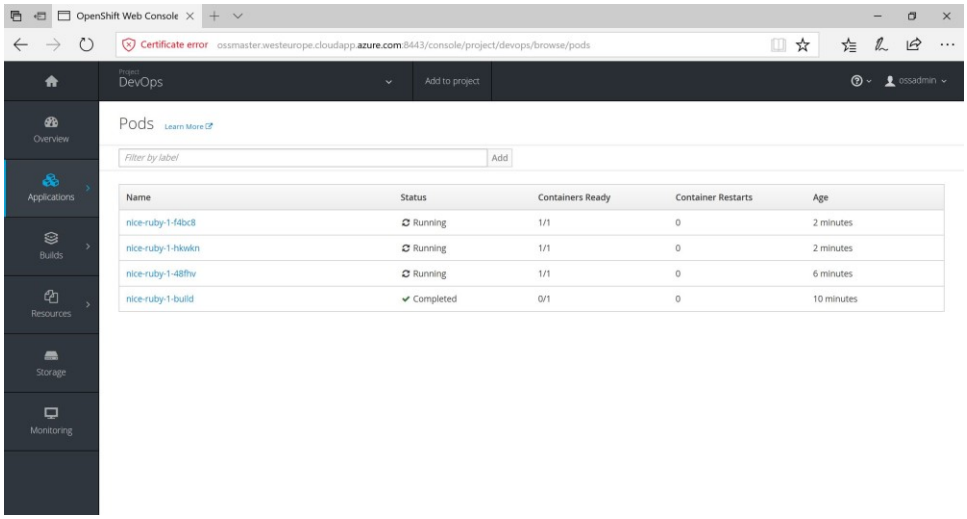




9. Return to the *OpenShift* admin console. Browse to the project's overview page, and test scaling out and in your application by increasing or decreasing the number of *pods*, using the up and down arrow signs on the web console.
   Scale out the app into 3 pods and watch the progress.



10. Browse to **Applications -> Pods**, and make sure 3 pods serving the same application are now up and running.

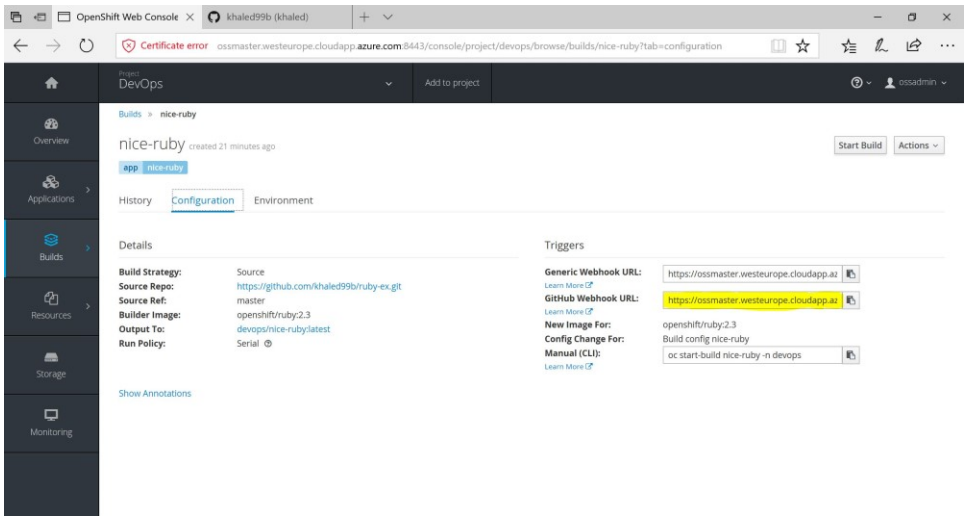

OpenShift Web Console
Certificate error
ossmaster.westeurope.cloudapp.azure.com:8443/console/project/devops/browse/pods
Project
DevOps
Add to project
ossadmin
Overview
Applications
Builds
Resources
Storage
Monitoring
Pods
Learn More
Filter by label
Add
Name
Status
Containers Ready
Container Restarts
Age
nice-ruby-1-f4bc8
Running
1/1
0
2 minutes
nice-ruby-1-hkwkn
Running
1/1
0
2 minutes
nice-ruby-1-48fhv
Running
1/1
0
6 minutes
nice-ruby-1-build
Completed
0/1
0
10 minutes

EXERCISE-4: CONFIGURING AUTOMATED BUILDS

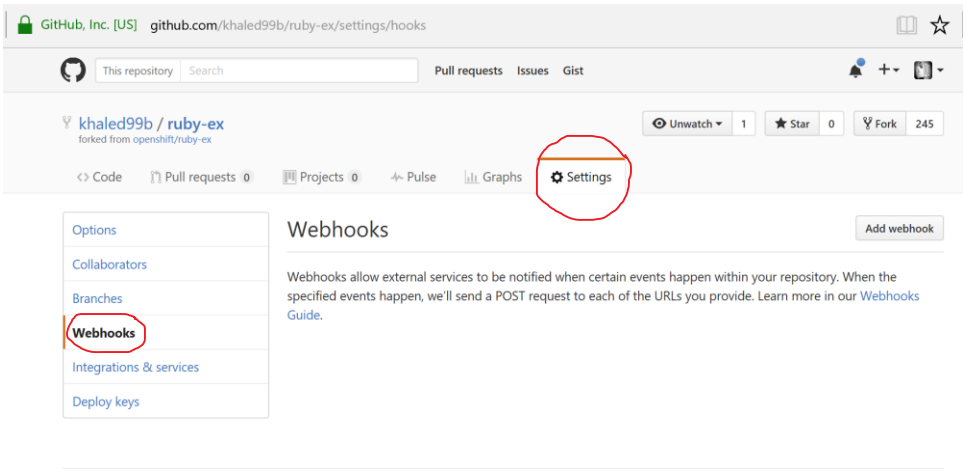
Since we forked the source code of the application from the OpenShift GitHub repository, we can use a *webhook* to automatically trigger a rebuild of the application whenever code changes are pushed to the forked repository.

To set up a *webhook* for your application:

1. From the Web Console, navigate to the project containing your application.
2. Click the **Browse** tab, then click **Builds**.
3. Click your build name, then click the **Configuration** tab.
4. Click next to **GitHub webhook URL** to copy your *webhook* payload URL.



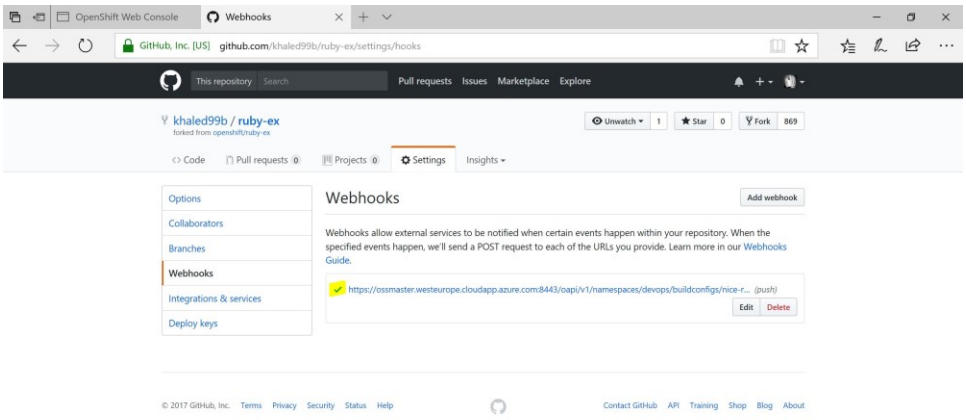
5. Navigate to your forked repository on GitHub, then click **Settings**.
6. Click **Webhooks & Services** and Click **Add webhook**.



7. Paste your *webhook* URL into the **Payload URL** field.
8. As Content Type choose application/json
9. Disable SSL verification and click **Add webhook** to save.

GitHub will now attempt to send a ping payload to your *OpenShift* server to ensure that communication is successful. If you see a green check mark appear next to your *webhook* URL, then it is correctly configured.

Hover your mouse over the check mark to see the status of the last delivery.



Next time you push a code change to your forked repository, your application will automatically rebuild.

EXERCISE-5: CONTINUOUS DEPLOYMENT

In this section, we demonstrate one of the most powerful features of *OpenShift*. We will see how we can trigger a continuous deployment pipeline, just by committing code change to Github.

Once there is a code change, the Github *webhook* will trigger the build of a new container image that combines a blueprint image from the registry with the updated code and generate a new image. This feature is called ***S2I***, or source to image. Once the build finishes, *OpenShift* will automatically deploy the new application based on the new image. This capability enables multiple deployment strategies such as A/B testing, Rolling upgrades…

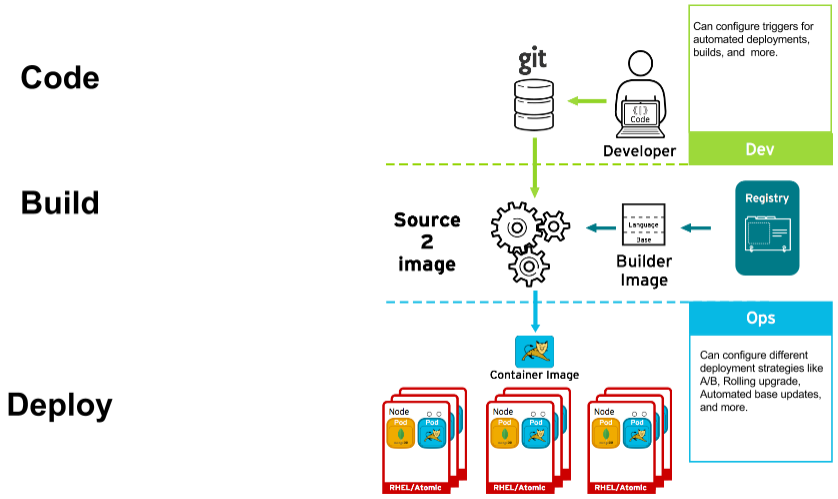


Figure 21: Continuous deployment pipeline

1. Use Azure cloud shell or install *Git* into your local machine

   PS: If you don't want to use *git*, you still can perform this exercise by moving to step-5 and editing the file *config.ru*, directly from the web interface of *github* and then go to step-8.

   Create a "dev" folder and change into.

```
$ mkdir dev && cd dev
```

2. Clone the forked repository to your local system

```
$ git clone
https://github.com/<YourGithubUsername>/ruby-ex.git
```

3. Make sure your local *git* repository is referencing to your *ruby-ex git*, on *github*:

```
$ cd ruby-ex
$ git remote -v
```

4. On your local machine, use your preferred text editor to change the sample application's source for the file ***config.ru***

   Make a code change that will be visible from within your application. For example: on line 229, change the title to "Welcome to your Ruby application on OpenShift POWERED BY AZURE!", then save your changes.

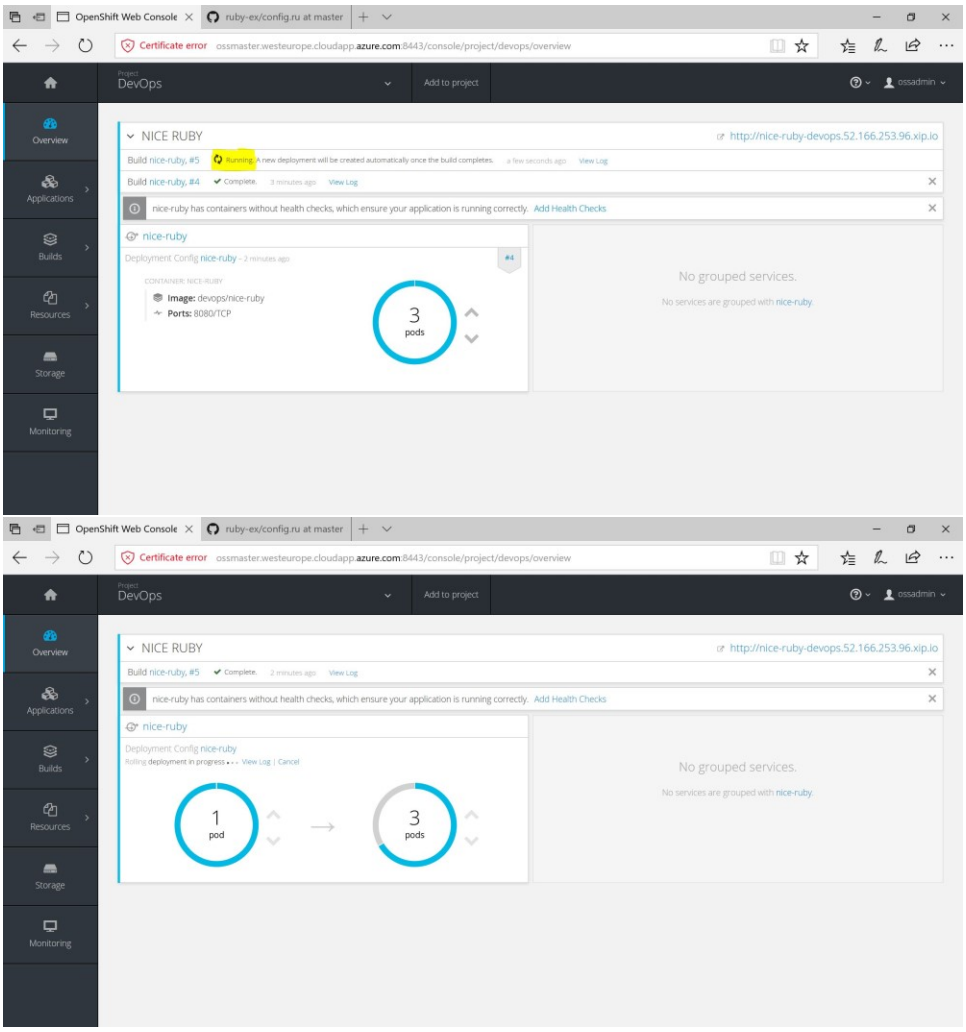
5. Verify the working tree status

```
$ git status
```

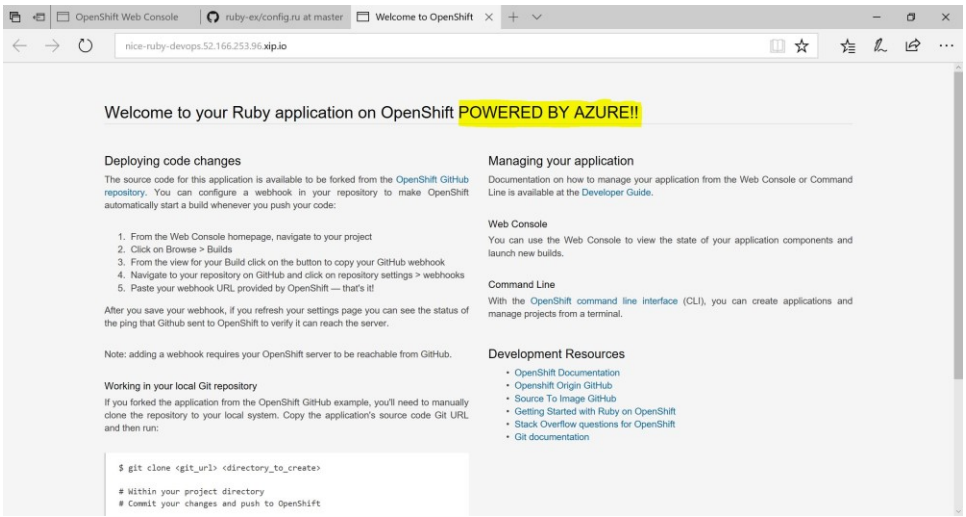
6. Add config.ru content to the index, Commit the change in *git*, and push the change to your fork. You will need to authenticate with your *github* credentials

```
$ git add config.ru
$ git commit -m "simple message"
$ git status
$ git push
```

7. If your *webhook* is correctly configured, your application will immediately rebuild itself, based on your changes. Monitor the build from the graphical console. Once the rebuild is successful, view your updated application using the route that was created earlier. Now going forward, all you need to do is push code updates and OpenShift handles the rest.



8. From the web browser refresh the page and note the new change



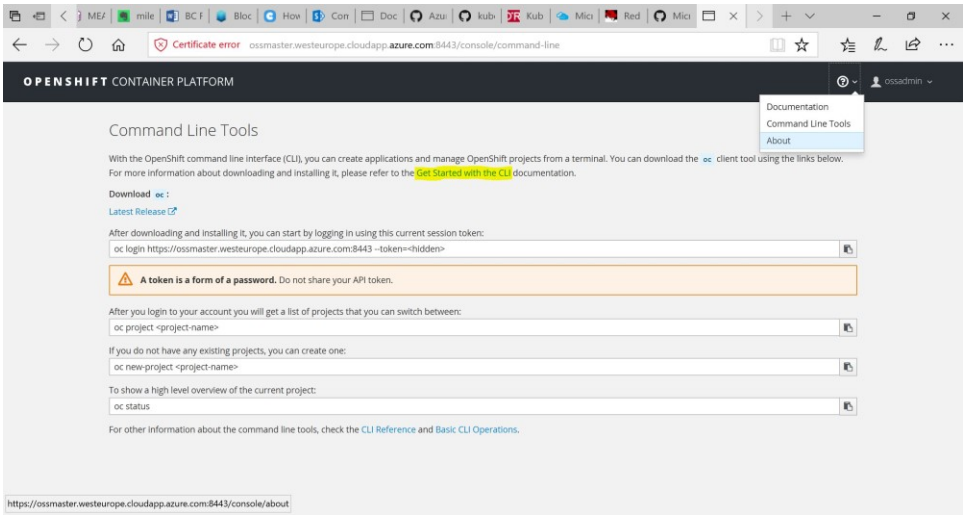
9. You may find it useful to manually rebuild an image if your *webhook* is not working, or if a build fails and you do not want to change the code before restarting the build. To manually rebuild the image based on your latest committed change to your forked repository:
    a. Click the **Browse** tab, then click **Builds**.
    b. Find your build, then click **Start Build**.

EXERCISE-6: MINI PROJECT: JBOSS EAP APPLICATION

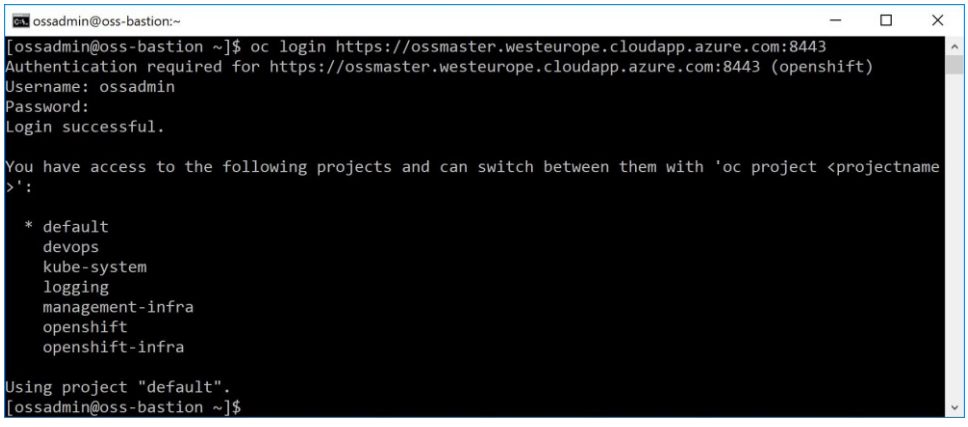
In this mini project, we will deploy a three tiers application consisting of Leaflet mapping front end with a JaxRS and MongoDB back end. The application, allows to visualize the locations of major National Parks and Historic Sites (https://github.com/OpenShiftDemos/restify-mongodb-parks).

During this exercise, we will leverage the CLI tool of OpenShift.

1. Download and install the OpenShift CLI related to your operating system. The easiest way to download the CLI is by accessing the About page on the web console if your cluster administrator has enabled the download links. Another alternative is to ssh into your bastion host that has the CLI tool installed already.



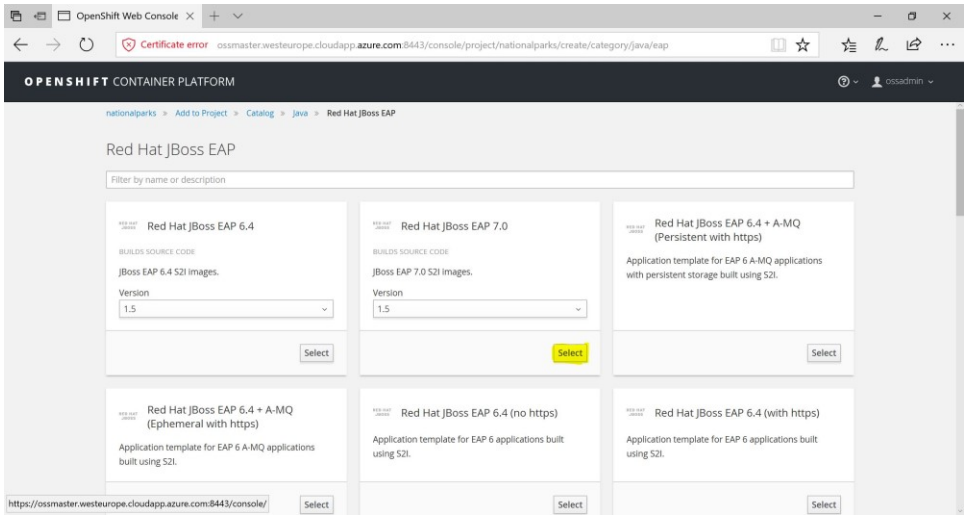
2. Use your openshift url endpoint to login to your environment from the CLI

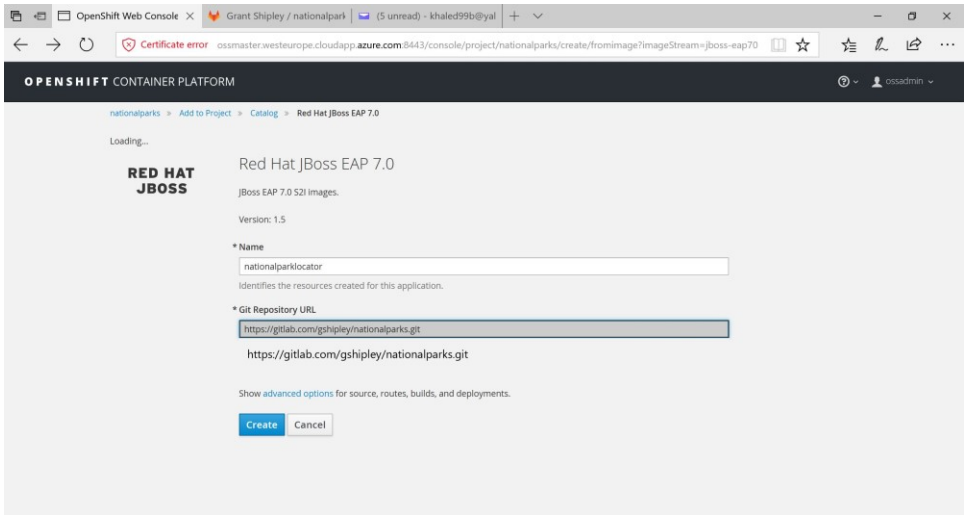


3. Create a new project “nationalparks”

```
ossadmin@oss-bastion:~
[ossadmin@oss-bastion ~]$ oc new-project nationalparks
Now using project "nationalparks" on server "https://ossmaster.westeurope.cloudapp.azure.com:8443".
You can add applications to this project with the 'new-app' command. For example, try:

    oc new-app centos/ruby-22-centos7~https://github.com/openshift/ruby-ex.git

to build a new example application in Ruby.
[ossadmin@oss-bastion ~]$
```

4. From the web console, add a new Java application using the following git lab repository https://gitlab.com/gshipley/nationalparks.git

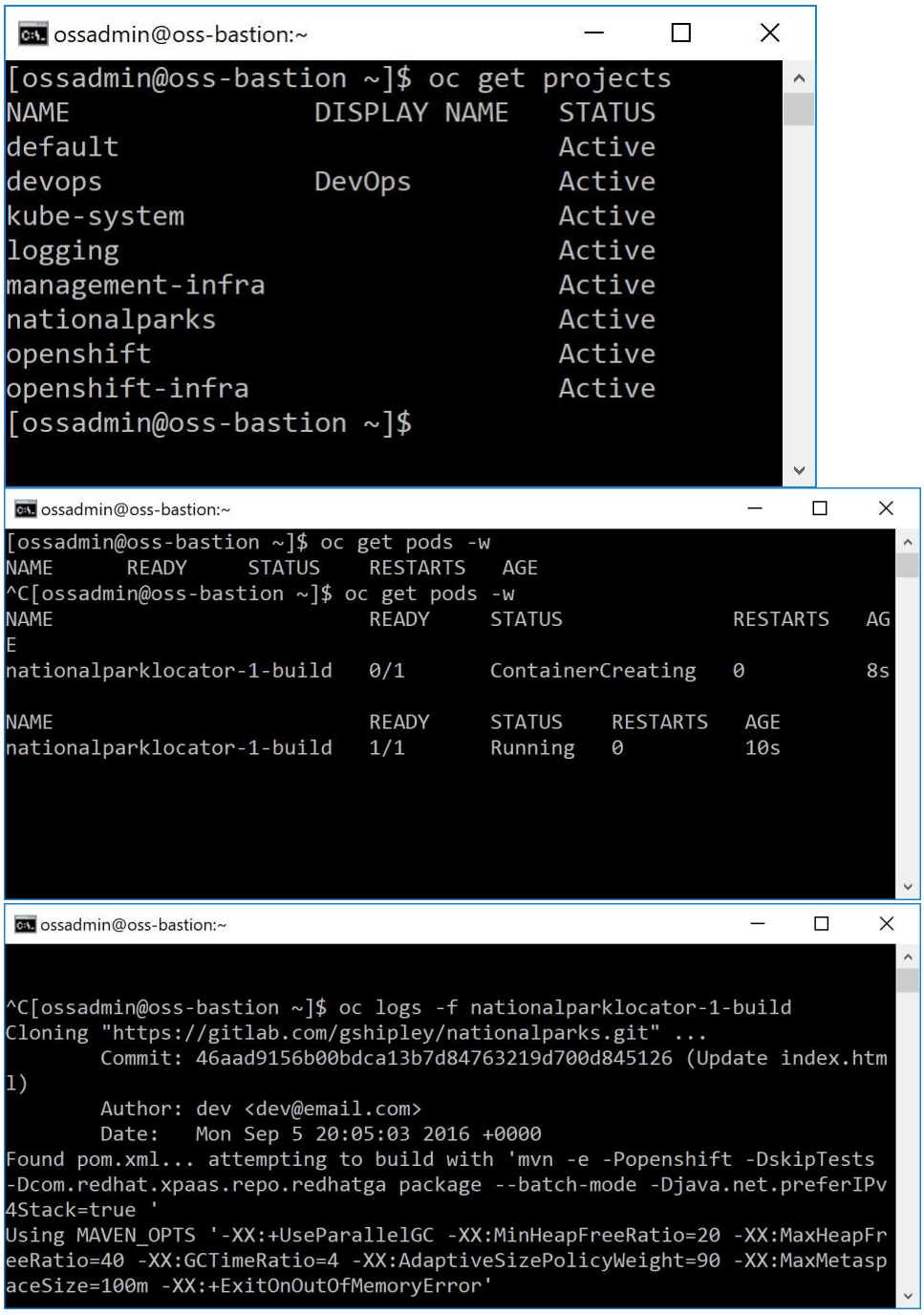




5. List builds operations:

```
ossadmin@oss-bastion:~
[ossadmin@oss-bastion ~]$ oc get builds
NAME                  TYPE      FROM          STATUS    STARTED              DURATION
nationalparklocator-1   Source    Git@46aad91   Running   About a minute ago
[ossadmin@oss-bastion ~]$
```

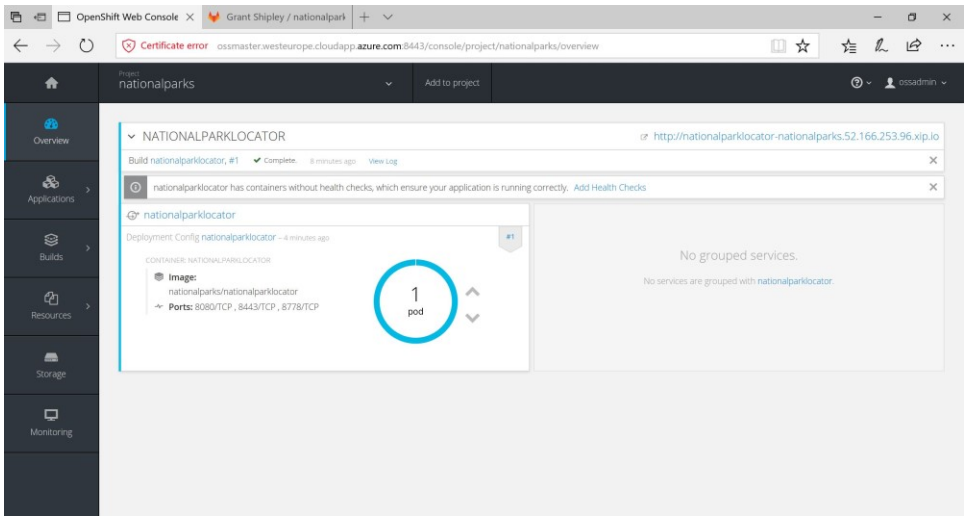
6. List existing projects, pods and view logs in real time:

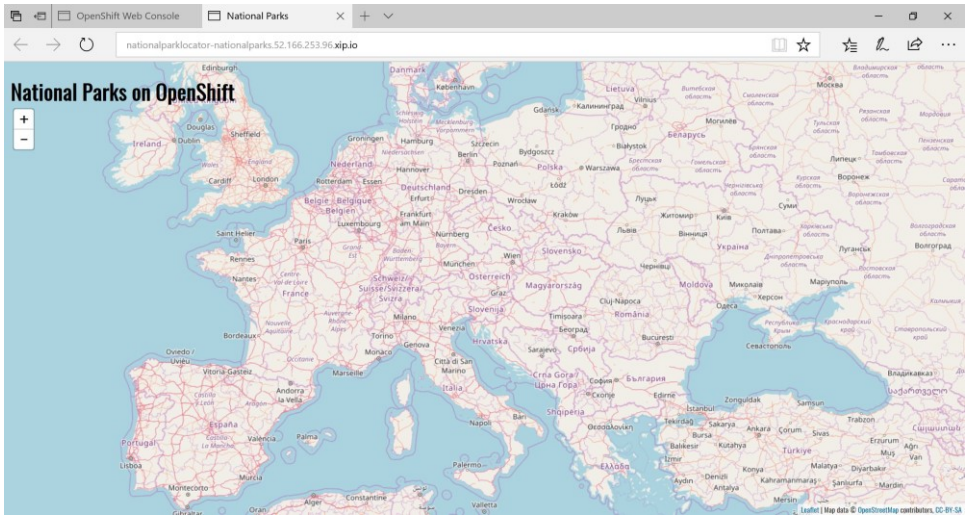


Output truncated ….

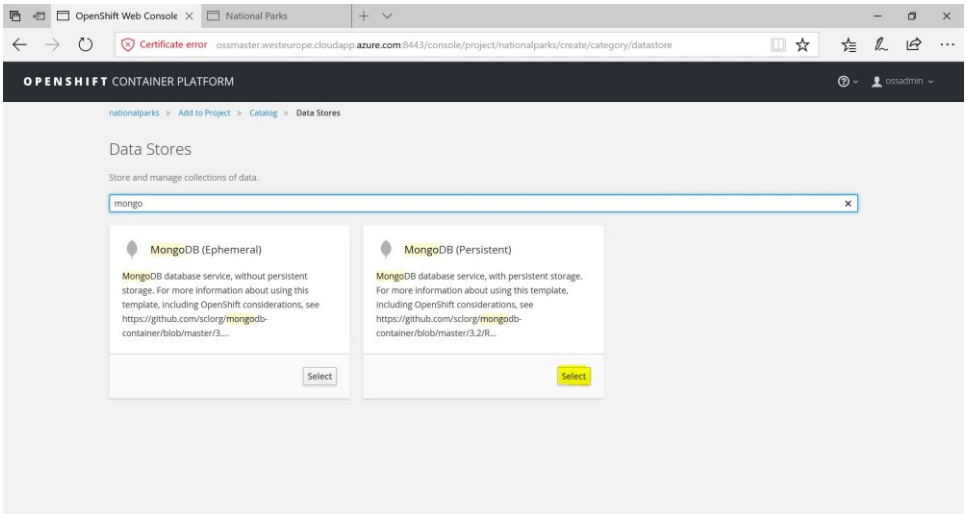
```
ossadmin@oss-bastion:~
ap/standalone/deployments for later deployment...
Pushing image 172.30.36.121:5000/nationalparks/nationalparklocator:latest
 ...
Pushed 0/7 layers, 1% complete
Pushed 1/7 layers, 27% complete
Pushed 2/7 layers, 48% complete
Pushed 3/7 layers, 58% complete
Pushed 4/7 layers, 84% complete
Pushed 5/7 layers, 88% complete
Pushed 6/7 layers, 88% complete
Pushed 6/7 layers, 98% complete
Pushed 7/7 layers, 100% complete
Push successful
[ossadmin@oss-bastion ~]$
```

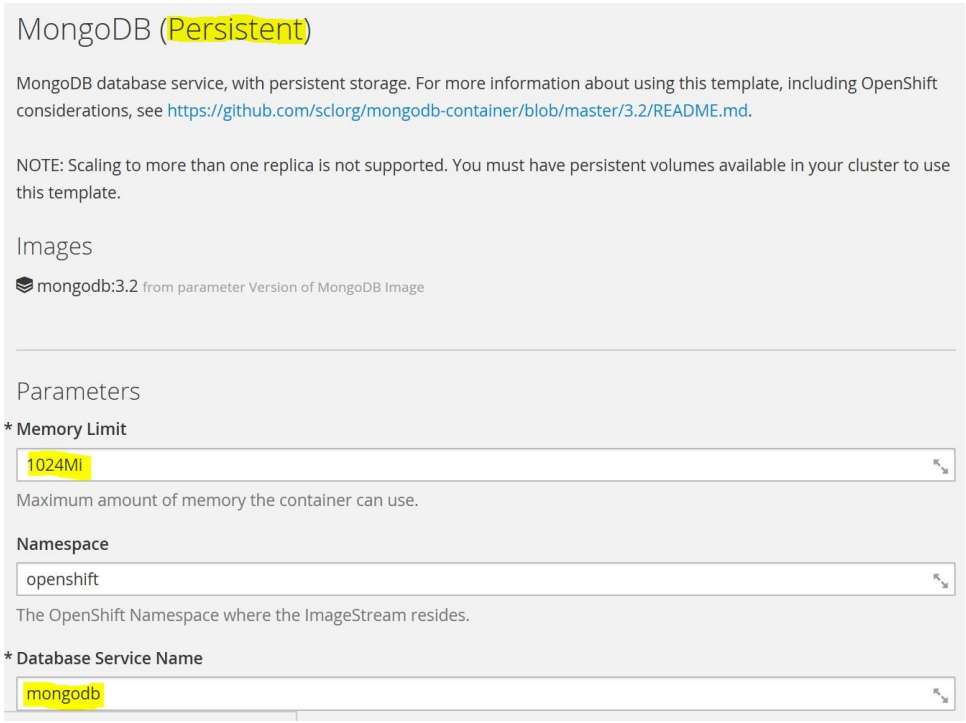
7. Browse the newly created application and verify it is available

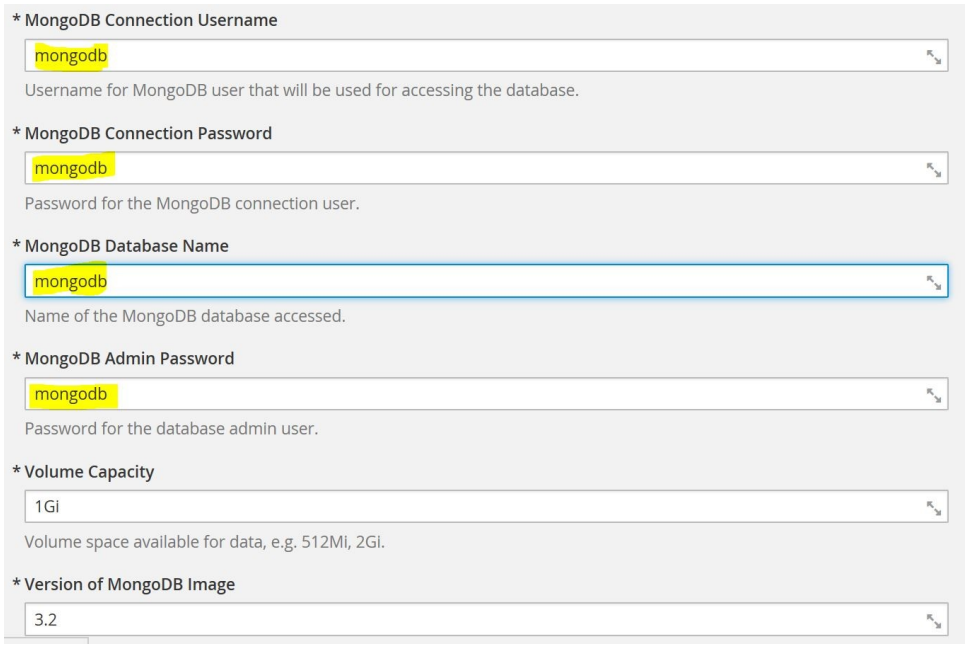




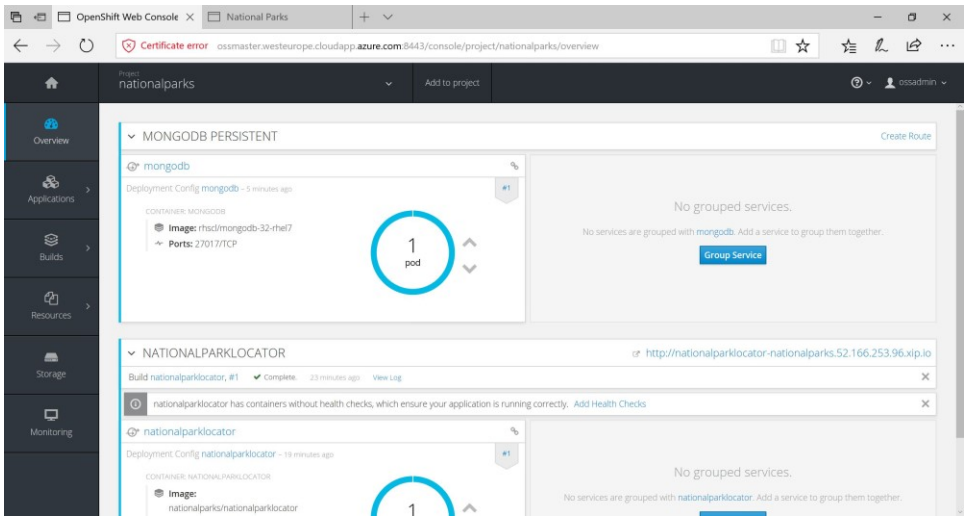
8. We can see the map but not the attraction points. The reason is that we only deployed the front-end application. What we will need now is to add a backend data base. From the web console, add a new persistent mongodb data store. And set the needed environment variables and specification as bellow:



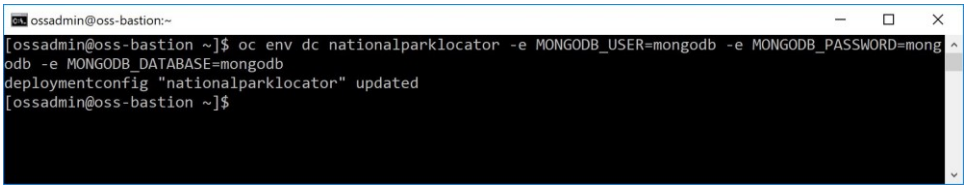




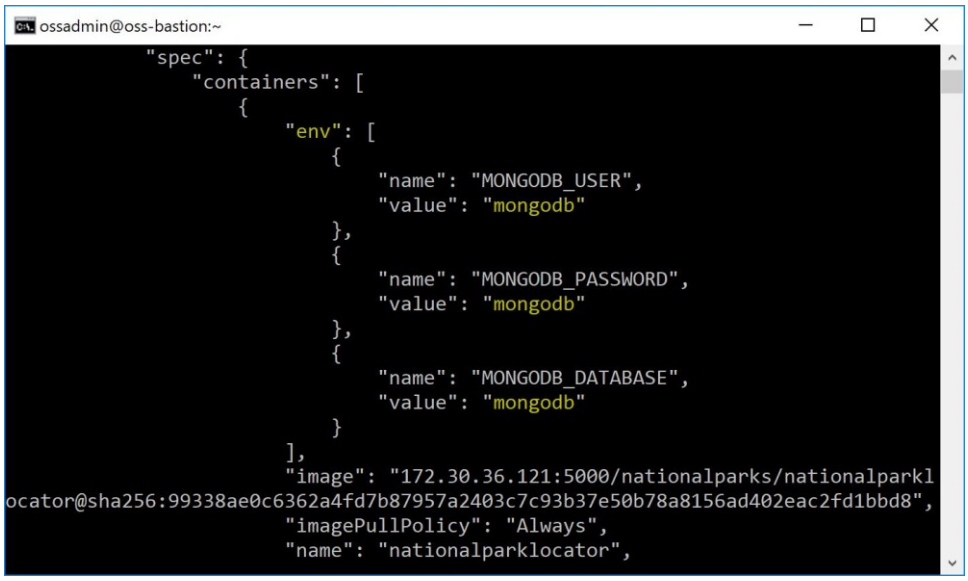
9. The graphical portal should show two applications in our project



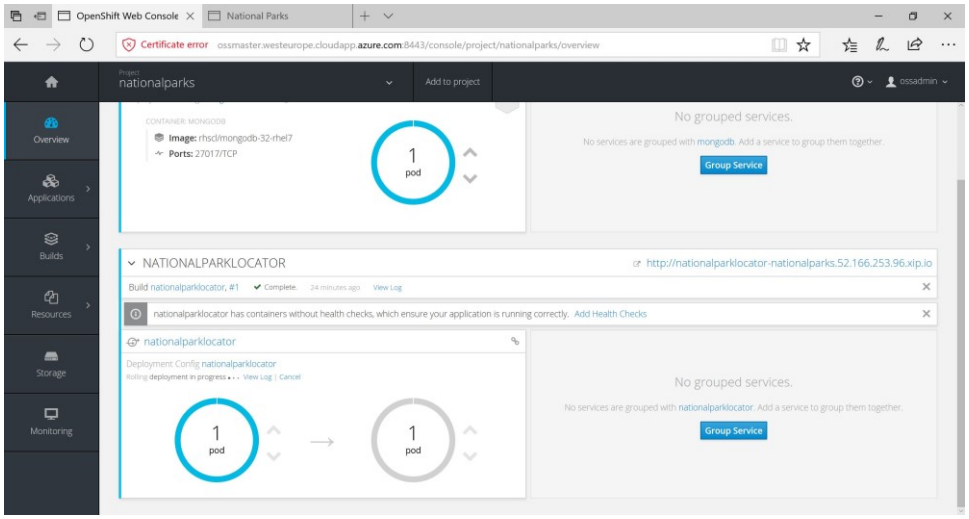
10. From the left menu in the web console, click on storage and verify that OpenShift created a persistent storage volume using Azure storage.
11. Change the deployment configuration of the front-end application to include the environment variables required to access the database



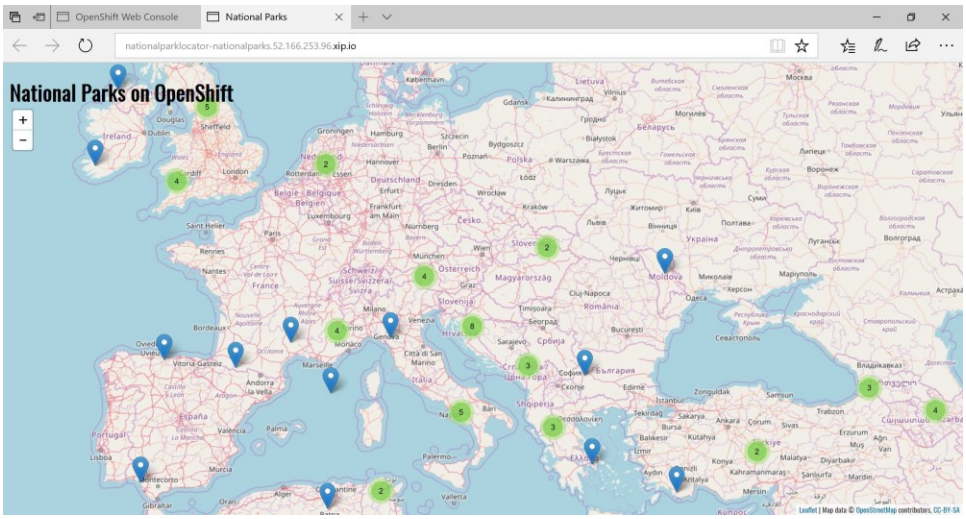
12. verify the last modification took place by running “oc get dc nationalparklocator -o json”



13. Back to the graphical console, note the automatic migration to a new pod based on the new configuration



13. Navigate to the application end-point and verify that the parks are now showing on the map



14. Our new application became very popular, and we need to scale out our front end to two pods.

```
ossadmin@oss-bastion:~
[ossadmin@oss-bastion ~]$ oc get dc
NAME                REVISION   DESIRED   CURRENT   TRIGGERED BY
mongodb             1          1         1         config,image(mongodb:3.2)
nationalparklocator   2          1         1         config,image(nationalparklocator:latest)
[ossadmin@oss-bastion ~]$
[ossadmin@oss-bastion ~]$ oc scale --replicas=2 dc/nationalparklocator
deploymentconfig "nationalparklocator" scaled
[ossadmin@oss-bastion ~]$
[ossadmin@oss-bastion ~]$ oc get dc
NAME                REVISION   DESIRED   CURRENT   TRIGGERED BY
mongodb             1          1         1         config,image(mongodb:3.2)
nationalparklocator   2          2         2         config,image(nationalparklocator:latest)
[ossadmin@oss-bastion ~]$
```

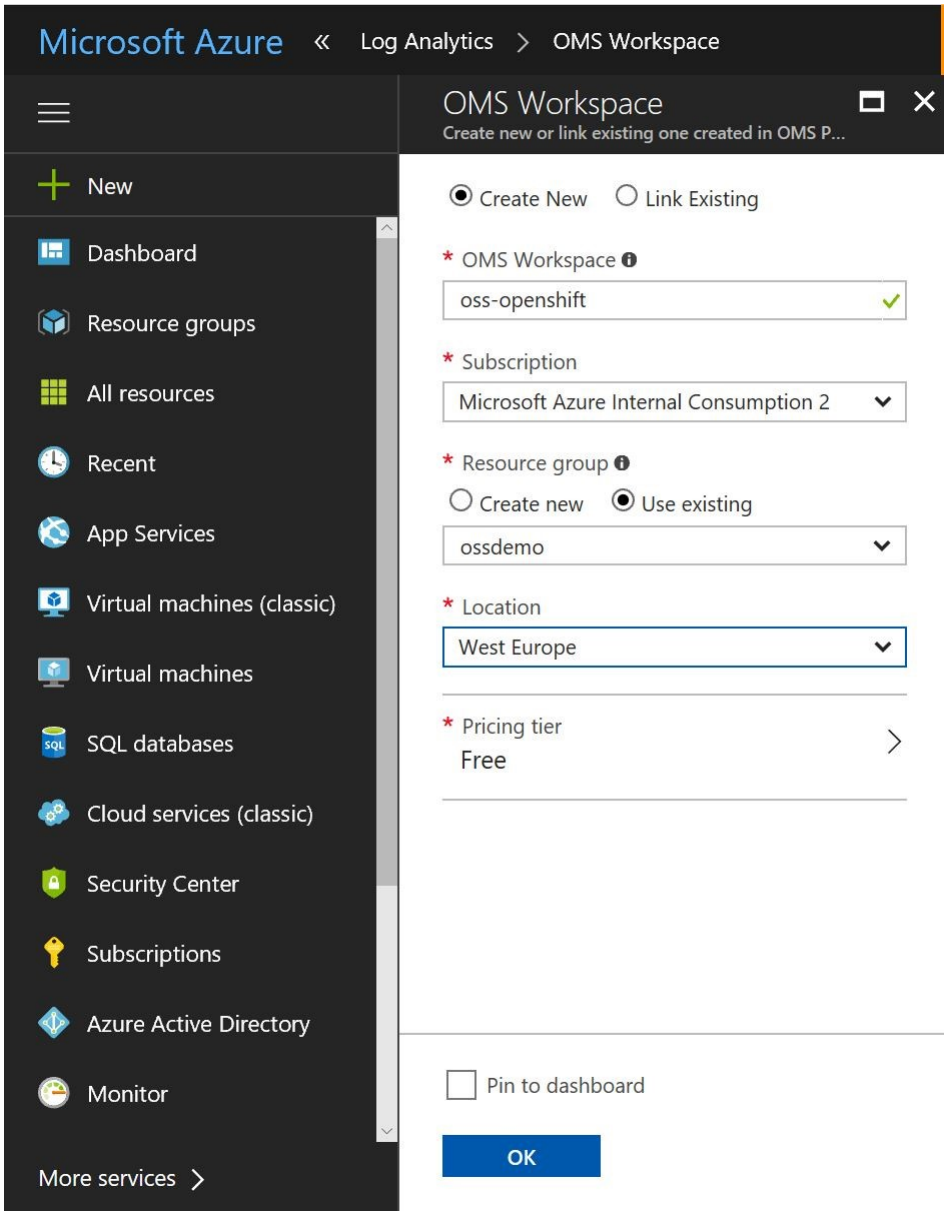
15. Now, let's test the self-healing, capabilities of OpenShift by deleting one of the running pods. Because, the desired state of the replication controller is 2 pods for the application "nationalparklocator", OpenShift will automatically and instantly trigger the deployment of a new pod.

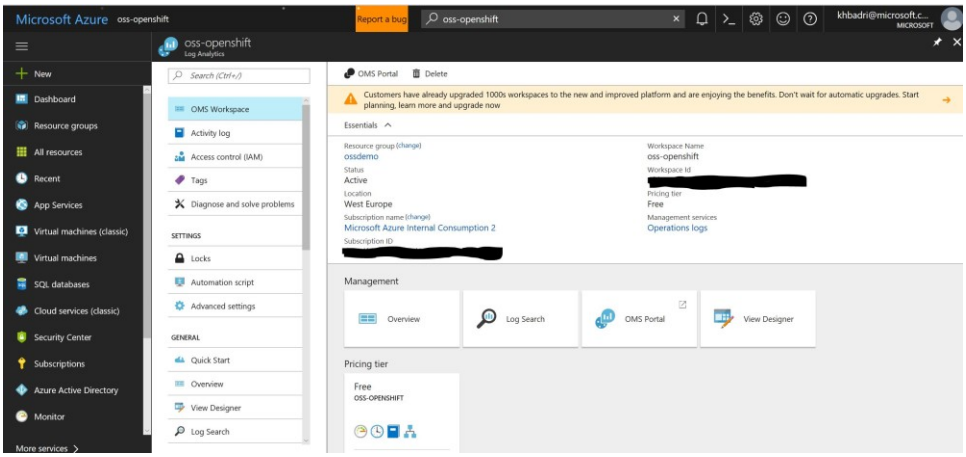
```
ossadmin@oss-bastion:~
[ossadmin@oss-bastion ~]$ oc get pods
NAME                        READY     STATUS              RESTARTS   AGE
mongodb-1-twrzb             0/1       ContainerCreating   0          1h
nationalparklocator-1-build   0/1       Completed           0          2h
nationalparklocator-2-2cqz2   1/1       Running             0          4m
nationalparklocator-2-nz14j   1/1       Running             0          1h
[ossadmin@oss-bastion ~]$
[ossadmin@oss-bastion ~]$ oc delete pod nationalparklocator-2-2cqz2
pod "nationalparklocator-2-2cqz2" deleted
[ossadmin@oss-bastion ~]$
[ossadmin@oss-bastion ~]$ oc get pods
NAME                        READY     STATUS              RESTARTS   AGE
mongodb-1-twrzb             0/1       ContainerCreating   0          1h
nationalparklocator-1-build   0/1       Completed           0          2h
nationalparklocator-2-nz14j   1/1       Running             0          1h
nationalparklocator-2-sqdq4   0/1       ContainerCreating   0          9s
[ossadmin@oss-bastion ~]$
[ossadmin@oss-bastion ~]$ oc get pods
NAME                        READY     STATUS              RESTARTS   AGE
mongodb-1-twrzb             0/1       ContainerCreating   0          1h
nationalparklocator-1-build   0/1       Completed           0          2h
nationalparklocator-2-nz14j   1/1       Running             0          1h
nationalparklocator-2-sqdq4   1/1       Running             0          1m
[ossadmin@oss-bastion ~]$
```

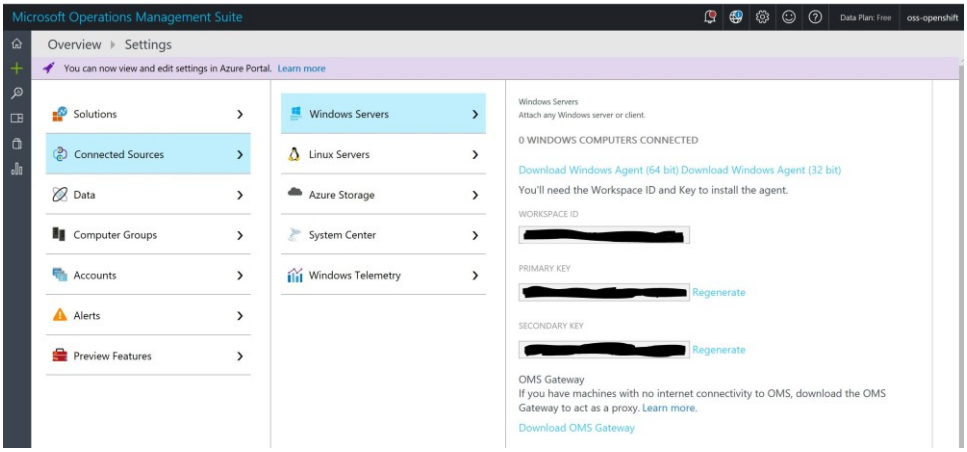
EXERCISE-7: MONITORING OEPNSHIFT WITH AZURE OMS

Azure Operations Management Suite (OMS) provides native support to OpenShift. In this exercise, we will walk through the steps of configuring OpenShift to export monitoring metrics directly to OMS

1. From the Azure portal create a new OMS workspace. Open the OMS portal and note the workspace id and one of the primary keys.







2. From Cloud Shell ssh into the bastion host, then ssh into one of the master node and create an OpenShift project and account for OMS.

```
[ossadmin@oss-bastion ~]$ ssh oss-master-0
[ossadmin@oss-master-0 ~]$ oadm new-project omslogging --node-selector='zone=default'
[ossadmin@oss-master-0 ~]$ oc project omslogging
[ossadmin@oss-master-0 ~]$ oc create serviceaccount omsagent
[ossadmin@oss-master-0 ~]$ oadm policy add-cluster-role-to-user cluster-reader system:serviceaccount:omslogging:omsagent
[ossadmin@oss-master-0 ~]$ oadm policy add-scc-to-user privileged system:serviceaccount:omslogging:omsagent
```

2. Use wget to download the ocp-* files from https://github.com/Microsoft/OMS-docker/tree/master/OpenShift

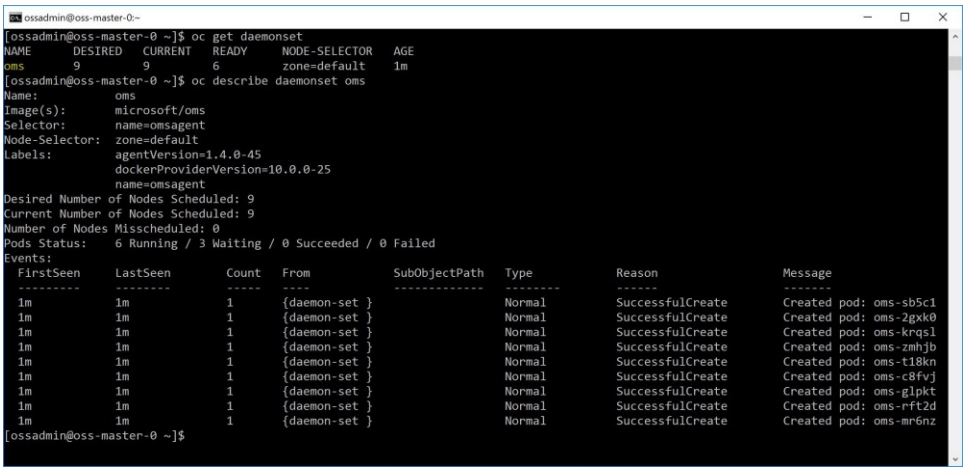
3. Make the file “secretgen.sh” executable. Run it, and provide our workspace id and key.

```
[ossadmin@oss-master-0 ~]$ chmod +x secretgen.sh
[ossadmin@oss-master-0 ~]$ ./secretgen.sh
```

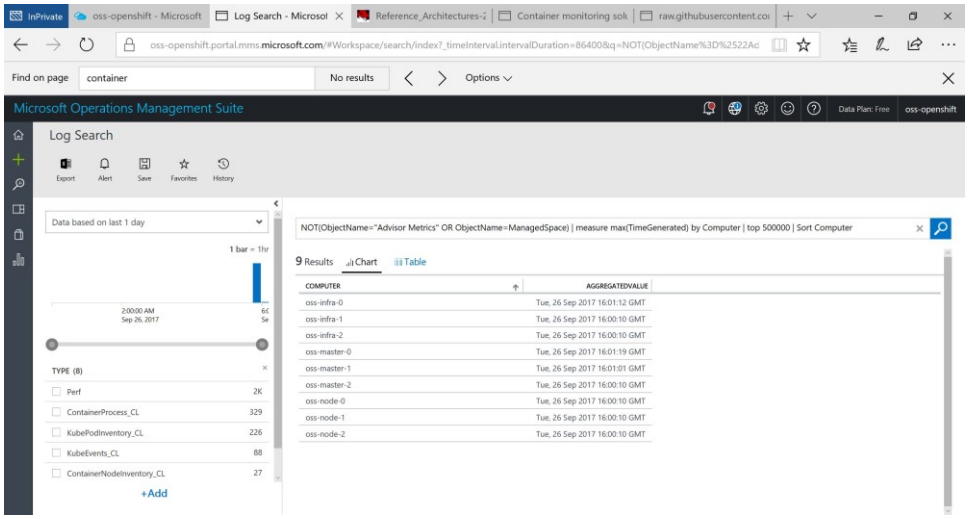
4. Create an OMS daemon set. A DaemonSet ensures that all the openshift cluster nodes run a copy of the oms pod.

```
[ossadmin@oss-master-0 ~]$ oc create -f ocp-secret.yaml
```

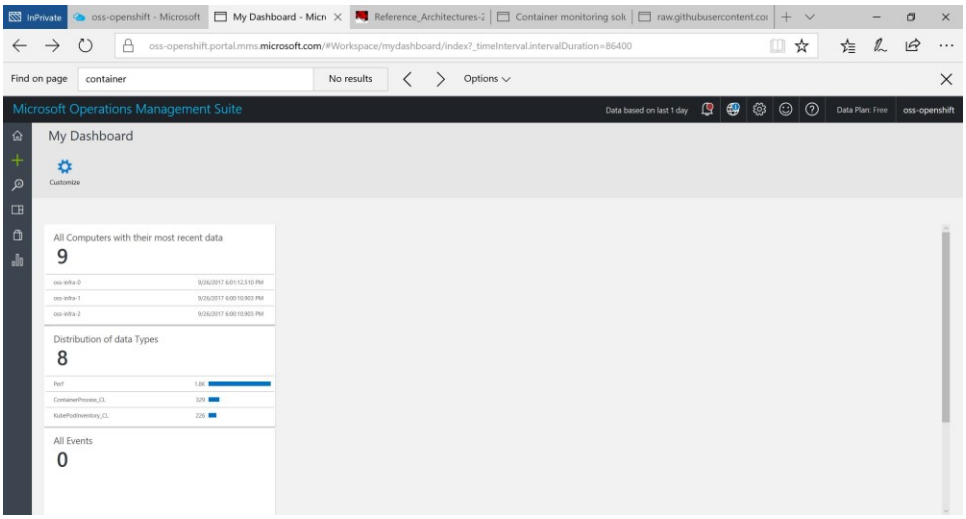
5. Validate that the daemon set is working properly

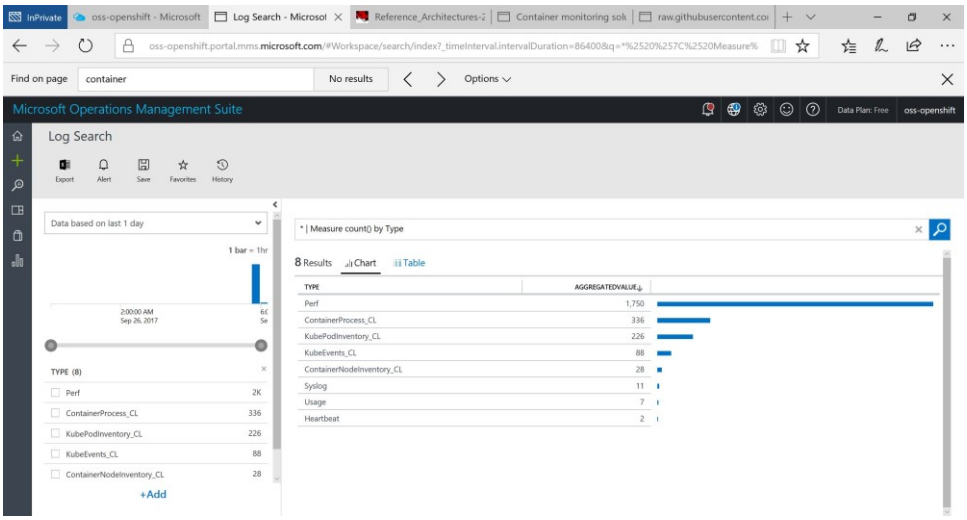


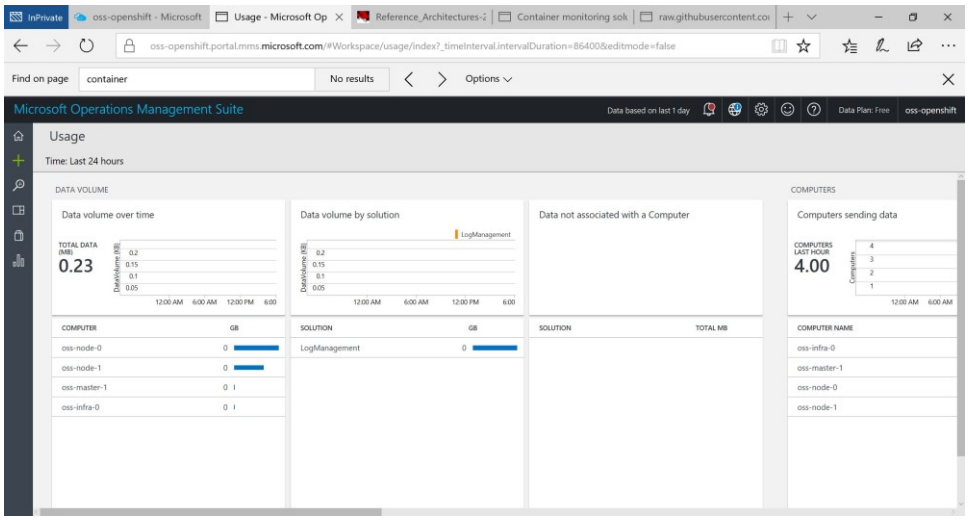
6. Back to OMS portal, you will find that there are new data sources exporting metrics.

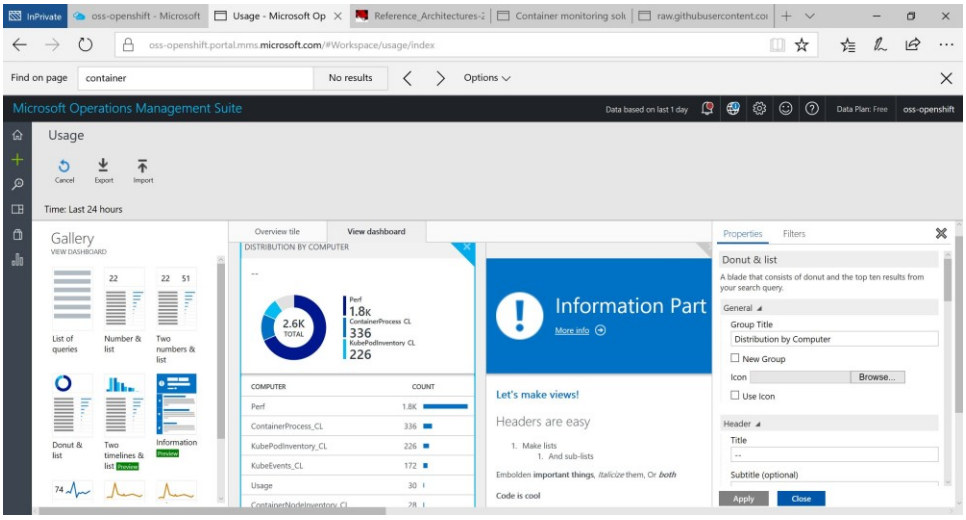


7. Create your custom dashboard and start exploring the data exported by OpenShift under different visualization formats


Microsoft Operations Management Suite
My Dashboard
Customize
All Computers with their most recent data
9
Distribution of data Types
8
All Events
0


Microsoft Operations Management Suite
Log Search
Export
Alert
Save
Favorites
History
Data based on last 1 day
* | Measure count() by Type
8 Results
Chart
Table
TYPE
AGGREGATEDVALUE
Perf 1,750
ContainerProcess_CL 336
KubePodInventory_CL 226
KubeEvents_CL 88
ContainerNodeInventory_CL 28
Syslog 11
Usage 7
Heartbeat 2
TYPE (8)
+Add


Microsoft Operations Management Suite
Usage
Time: Last 24 hours
DATA VOLUME
Data volume over time
0.23
Data volume by solution
Data not associated with a Computer
COMPUTERS
Computers sending data
4.00
COMPUTER
oss-node-0
oss-node-1
oss-master-1
oss-infra-0
SOLUTION
LogManagement
COMPUTER NAME


Microsoft Operations Management Suite
Usage
Cancel
Export
Import
Time: Last 24 hours
Gallery
VIEW DASHBOARD
List of queries
Number & list
Two numbers & list
Donut & list
Two timelines & list
Information
Overview tile
View dashboard
DISTRIBUTION BY COMPUTER
2.6K
TOTAL
Perf
1.8K
ContainerProcess_CL
336
KubePodInventory_CL
226
COMPUTER
COUNT
Information Part
More info
Let's make views!
Headers are easy
Code is cool
Properties
Filters
Donut & list
A blade that consists of donut and the top ten results from your search query.
General
Group Title
Distribution by Computer
New Group
Icon
Browse...
Use Icon
Header
Title
Subtitle (optional)
Apply
Close

EXERCISE-8: RED HAT CLOUD FORMS ON AZURE

Red Hat Cloud Forms is an open source cloud management solution providing a unified and consistent set of management capabilities across:

- Virtualization platforms like Red Hat Virtualization, VMware vCenter, and Microsoft Hyper-V.
- Private cloud platforms based on OpenStack.
- Public cloud platforms like Microsoft Azure and Amazon Web Services.
- Red Hat OpenShift

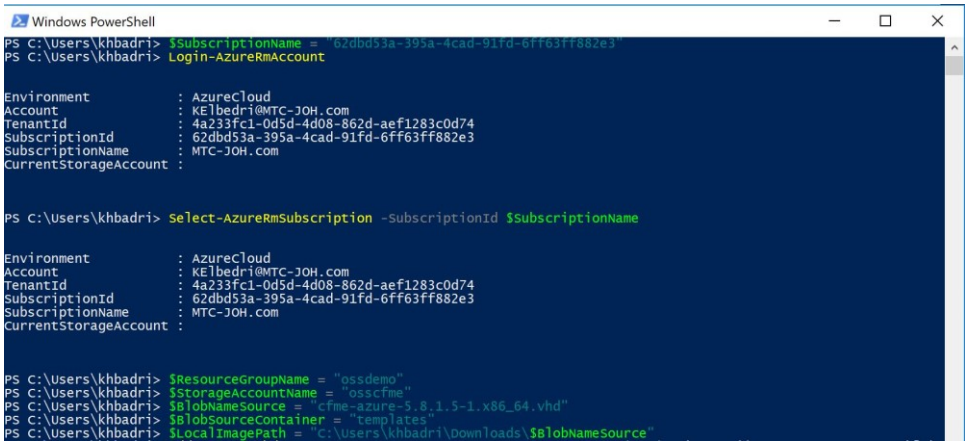
CloudForms can see and manage both the guest and host systems, allowing management of workloads and infrastructure within the same system.

CloudForms is a first-class citizen on Azure and provides a solid hybrid cloud management solution. During this exercise we will walk through the installation and configuration of CloudFroms on Azure and the integration with the OpenShift environment created in the previous steps. In this section we will leverage another powerful open source SDK to perform the steps: Powershell.

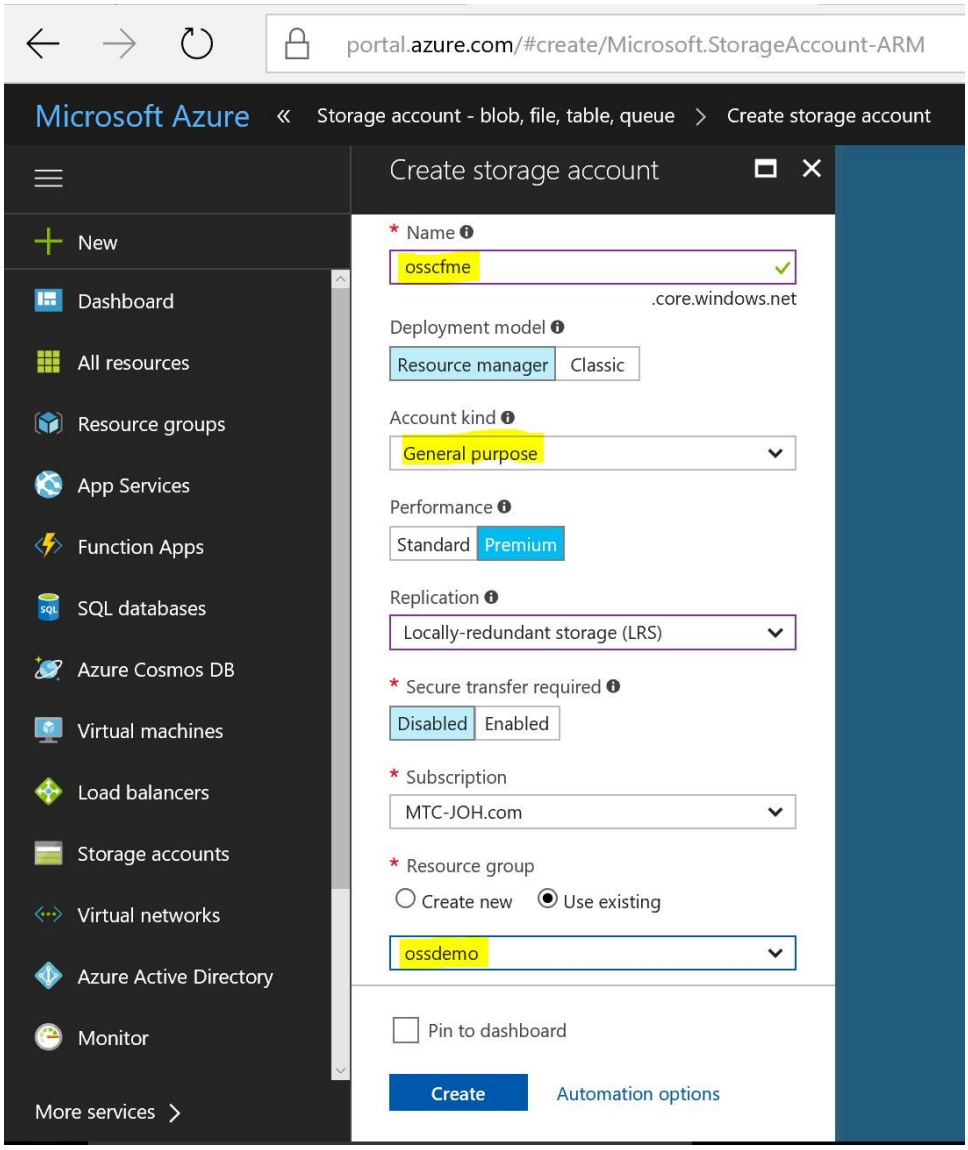
1. Login to Red Hat network portal and download the latest cloudforms vhd image for Azure. You will need an active Red Hat subscription. (https://www.redhat.com/en/technologies/management/cloudforms)
2. We will need to round up the size of the vhd image to a multiple number of MB, otherwise the provisioning will fail in the next steps. We can use powershell to perform this operation

```
> Resize-VHD -Path $LocalImagePath -SizeBytes 32770MB
```

3. Configure some variables to use during the deployment. Change $BlobNameSource and $LocalImagePath to reflect your environment.

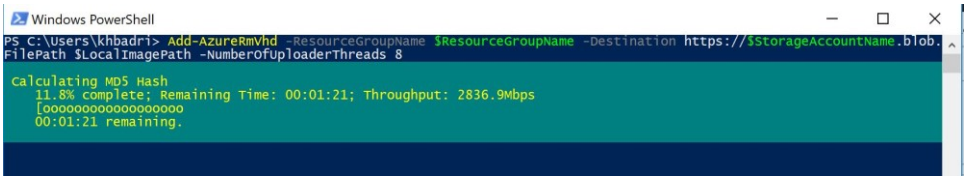


4. Create a new storage account “osscfme” in the “ossdemo” resource group to deploy the vhd image to.

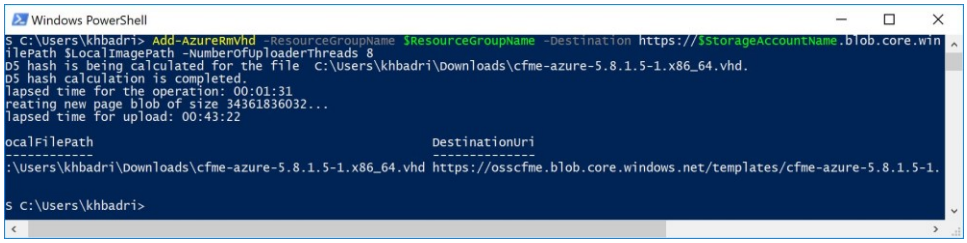


5. Upload the vhd image into the newly created storage account. This operation will take 15 minutes. Time for a break!

```
> Add-AzureRmVhd -ResourceGroupName $ResourceGroupName -
Destination
https://$StorageAccountName.blob.core.windows.net/$BlobSour
ceContainer/$BlobNameSource -LocalFilePath $LocalImagePath
-NumberOfUploaderThreads 8
```



.....

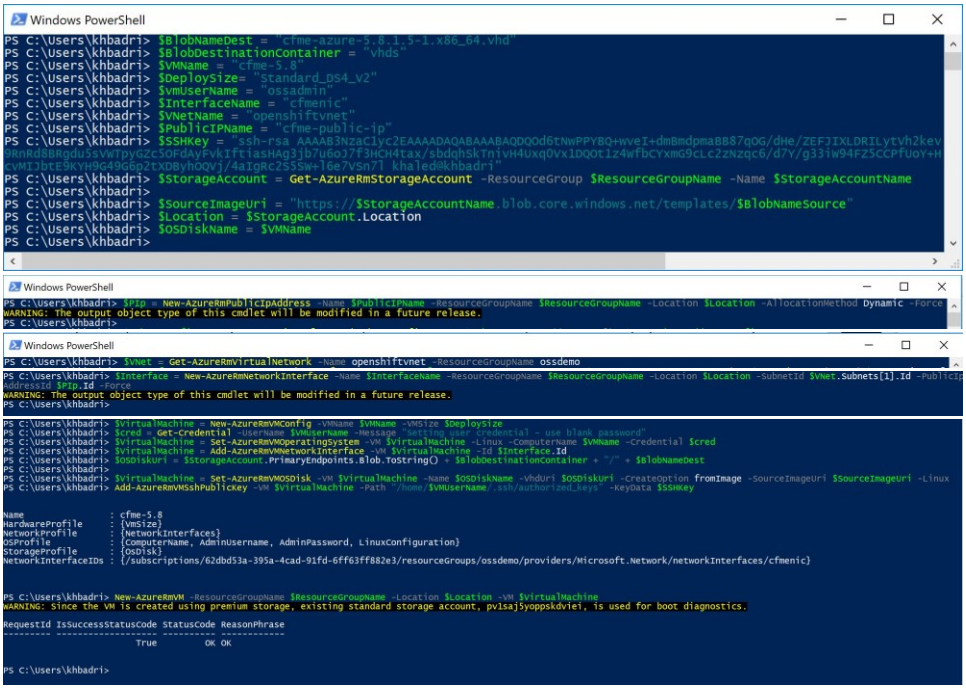


6. Customize the Azure environment and create the CloudForms vm

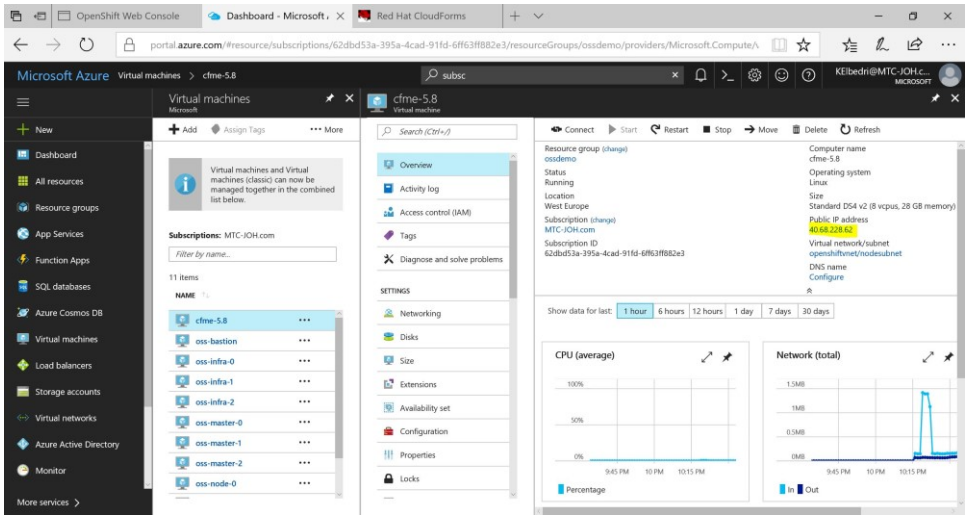
```
$BlobNameDest = "cfme-azure-5.8.1.5-1.x86_64.vhd"

$BlobDestinationContainer = "vhds"

$VMName = "cfme-5.8"

$DeploySize= "Standard_DS4_V2"

$vmUserName = "ossadmin"


$InterfaceName = "cfmenic"

$VNetName = "openshiftvnet"

$PublicIPName = "cfme-public-ip"


$SSHKey = " ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDQOd6tNwPPYBQ+wveI+dmBmdpmaBB
87qOG/dHe/ZEFJIXLDRILytVh2kevgeXn/SzbqL3DJ4qQWVGmsaZwELhlQK
dcc/AybwRn9tQ94H2WgPQ79RnRd8BRgdu5sVWTpyGZc5OFdAyFvkIftiasH
Ag3jb7u6oJ7f3HCH4tax/sbdqhSkTnivH4Uxq0Vx1DQOt1z4WfbCYxmG9cL
c2zNzqc6/d7Y/g33iW94FZ5CCPfUoY+HdyOSu5cy/rWtreCskWQZwNR8xkv
DOKIlc2bnBTCosN79FMzZYyiOcvMIJbtE9KYH9G49G6p2tXDByhOQVj/4aI
gRc2S5SW+l6e7VSn7l khaled@khbadri"


$StorageAccount = Get-AzureRmStorageAccount -ResourceGroup
$ResourceGroupName -Name $StorageAccountName


$SourceImageUri =
"https://$StorageAccountName.blob.core.windows.net/template
s/$BlobNameSource"

$Location = $StorageAccount.Location

$OSDiskName = $VMName
```

```
# Network
$PIp = New-AzureRmPublicIpAddress -Name $PublicIPName -
ResourceGroupName $ResourceGroupName -Location $Location -
AllocationMethod Dynamic -Force
$VNet = Get-AzureRmVirtualNetwork -Name openshiftVnet -
ResourceGroupName ocpkhaled


$Interface = New-AzureRmNetworkInterface -Name
$InterfaceName -ResourceGroupName $ResourceGroupName -
Location $Location -SubnetId $VNet.Subnets[1].Id -
PublicIpAddressId $PIp.Id -Force


# Specify the VM Name and Size
$VirtualMachine = New-AzureRmVMConfig -VMName $VMName -
VMSize $DeploySize


# Add User
$cred = Get-Credential -UserName $VMUserName -Message
"Setting user credential - use blank password"
$VirtualMachine = Set-AzureRmVMOperatingSystem -VM
$VirtualMachine -Linux -ComputerName $VMName -Credential
$cred


# Add NIC
$VirtualMachine = Add-AzureRmVMNetworkInterface -VM
$VirtualMachine -Id $Interface.Id


# Add Disk
$OSDiskUri =
$StorageAccount.PrimaryEndpoints.Blob.ToString() +
$BlobDestinationContainer + "/" + $BlobNameDest


$VirtualMachine = Set-AzureRmVMOSDisk -VM $VirtualMachine -
Name $OSDiskName -VhdUri $OSDiskUri -CreateOption fromImage
-SourceImageUri $SourceImageUri -Linux


# Set SSH key
Add-AzureRmVMSshPublicKey -VM $VirtualMachine -Path
"/home/$VMUserName/.ssh/authorized_keys" -KeyData $SSHKey


# Create the VM
```

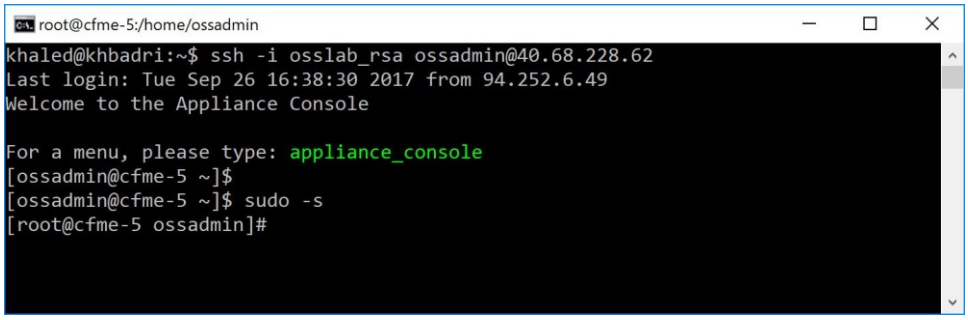
```
New-AzureRmVM -ResourceGroupName $ResourceGroupName -
Location $Location -VM $VirtualMachine
```



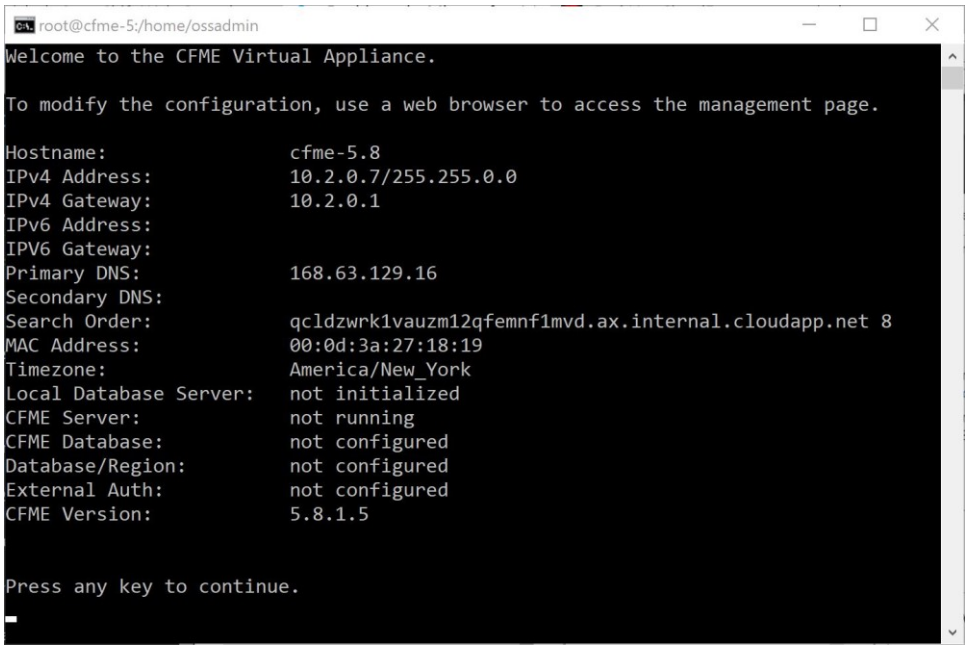
7. Navigate into the “cfme” vm from the Azure Portal, and note the IP address.



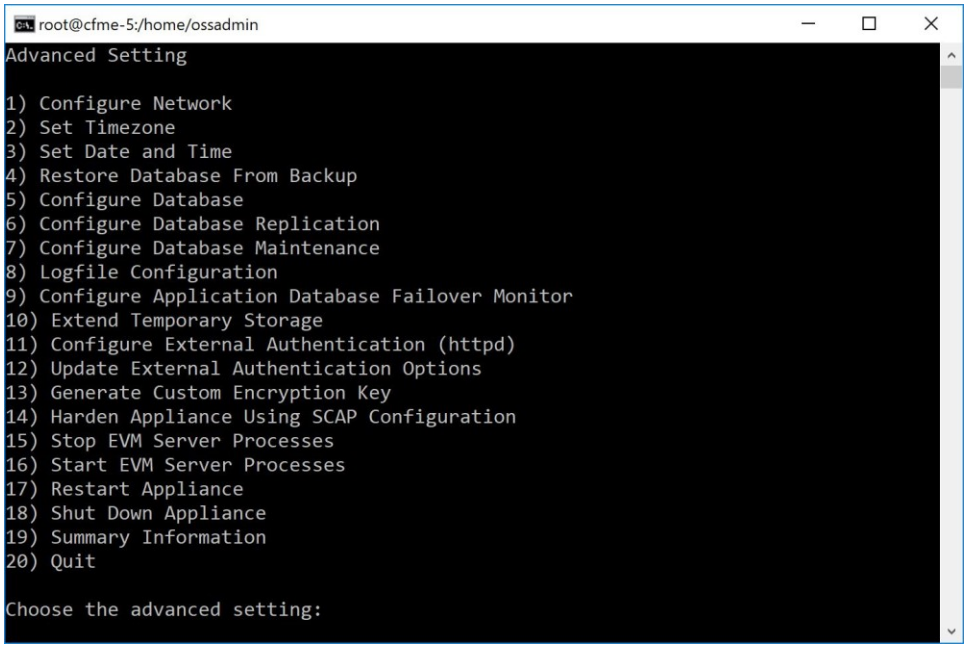
8. From the Cloud Shell, ssh into the appliance using the SSH key.



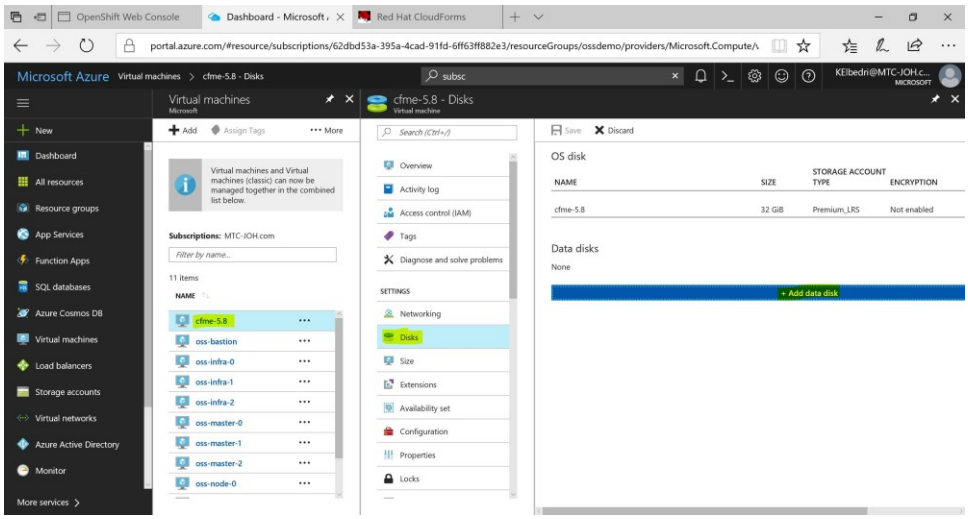
9. Switch to root “sudo -s” and enter “appliance_console” command. The Red Hat CloudForms appliance summary screen displays.

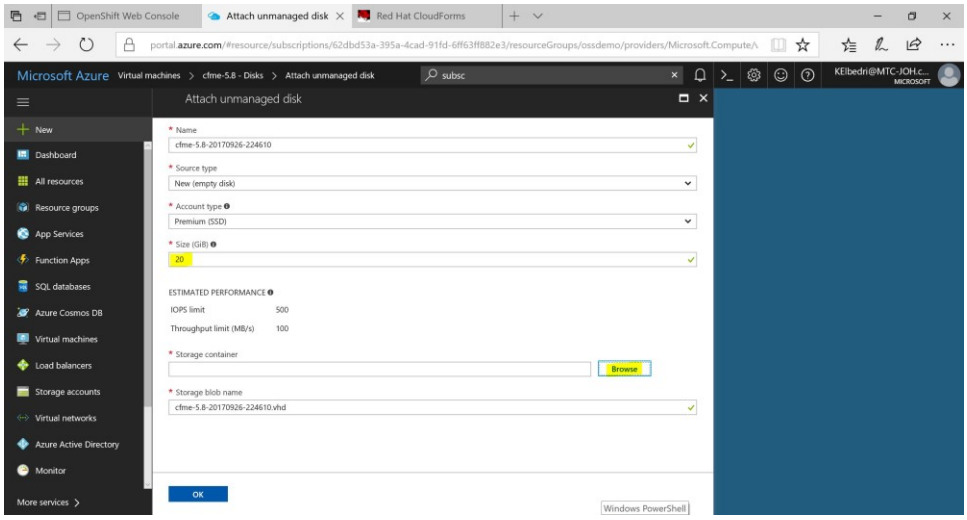


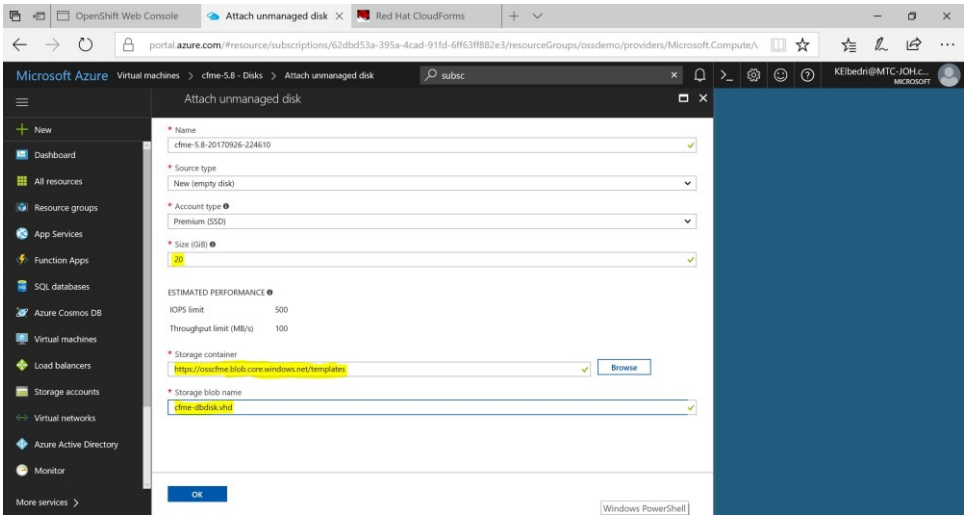
10. Press Enter to manually configure settings.



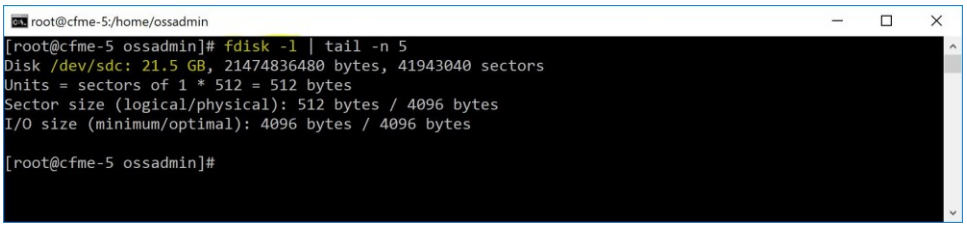
11. Press the number for the item you want to change, and press Enter. The options for your selection are displayed.
12. Follow the prompts to make the changes.
13. Press Enter to accept a setting where applicable. Configure the time zone (2) and quit (20)
14. Back to the Azure portal. We need to add a new disk for the CloudForms data base.



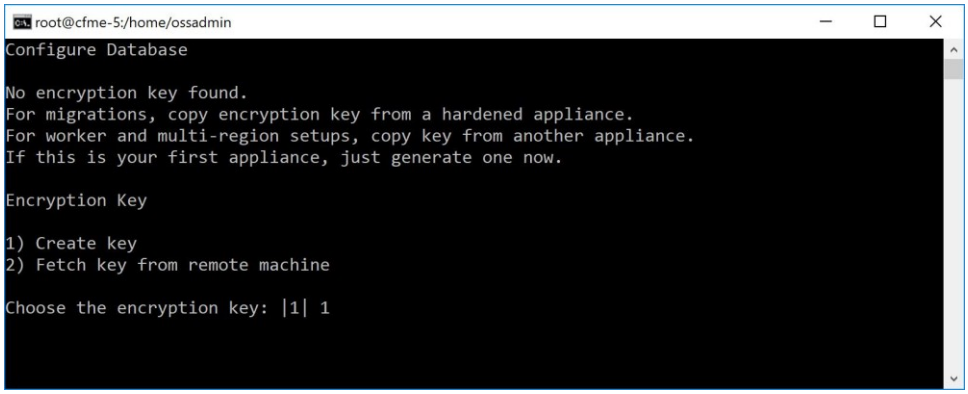




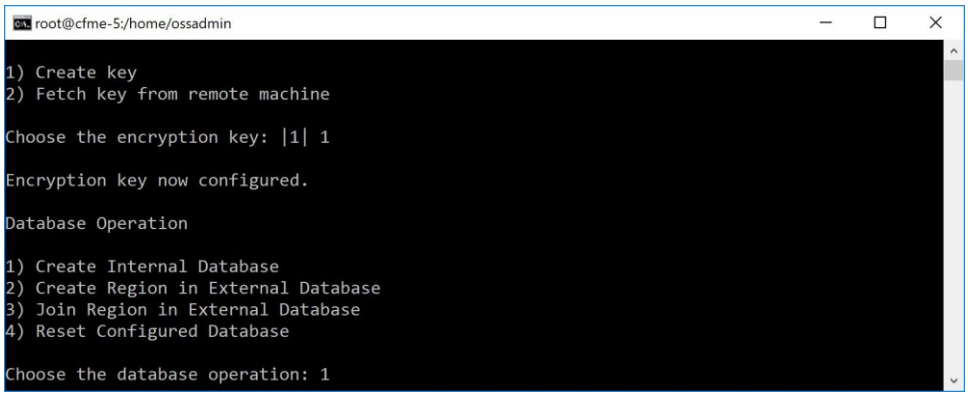
15. Use the command fdisk to verify the newly added disk



16. Run “appliance_console” again, hit enter and choose (5) to configure the database
17. Choose (1) to create a key

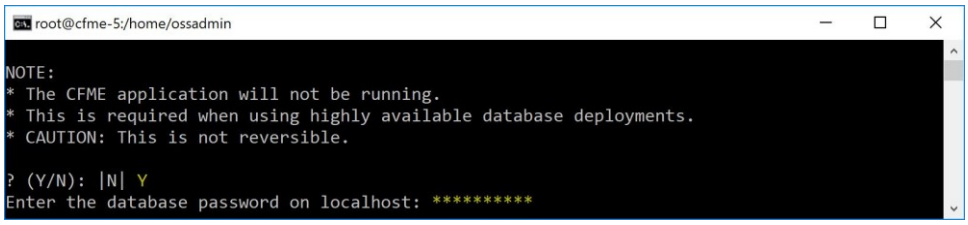


18. Choose (1) to create internal database for the database location.

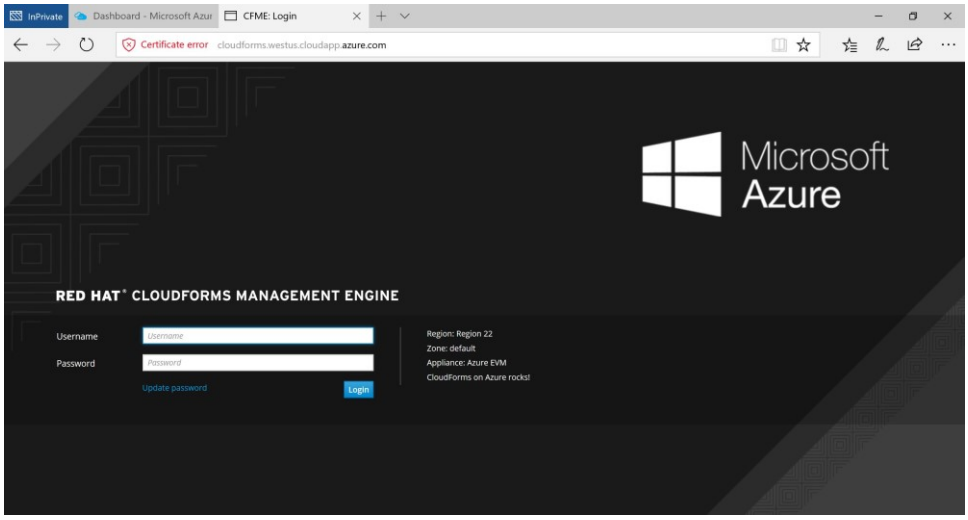


19. Choose (1) for the disk we attached previously

20. Select Y to configure the appliance as a database-only appliance and create and confirm a password for the database. As a result, the appliance is configured as a basic PostgreSQL server, without a user interface

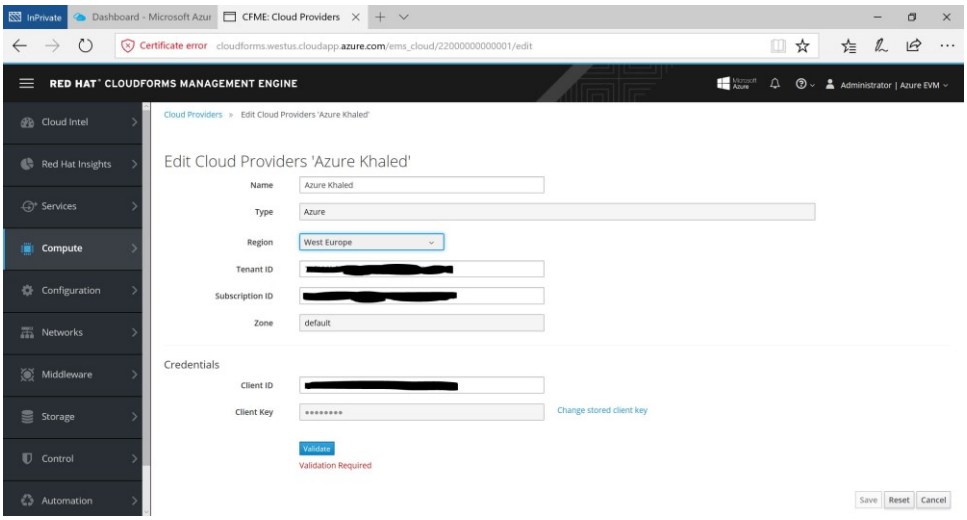


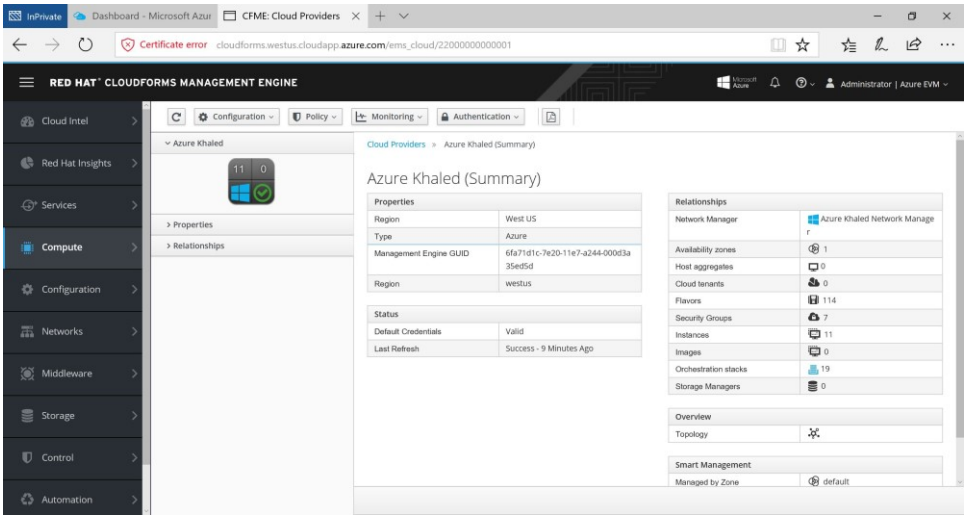
21. Choose (16) to start EVM processes. If it fails, choose () to restart the appliance
22. Once Red Hat CloudForms is installed, you can log in and perform administration tasks.
23. Log in to Red Hat CloudForms for the first time after installing by:
24. Navigate to the URL for the login screen. (https://xx.xx.xx.xx on the virtual machine instance)



25. Enter the default credentials (Username: admin | Password: smartvm) for the initial login.
26. Click Login.
27. Navigate to the URL for the login screen. (https://xx.xx.xx.xx on the virtual machine instance)
28. Click Update Password beneath the Username and Password text fields.
29. Enter your current Username and Password in the text fields.
30. Input a new password in the New Password field.
31. Repeat your new password in the Verify Password field.
32. Click Login.
33. To Add an Azure Cloud Provider:
    - Navigate to Compute → Clouds → Providers.
    - Click (Configuration), then click (Add a New Cloud Provider).
    - Enter a Name for the provider.

- From the Type list, select Azure.
- Select a region from the Region list. One provider will be created for the selected region.
- Enter Tenant ID.
- Enter Subscription ID.
- Enter Zone.
- In the Credentials section, enter the Client ID and Client Key; click Validate.
- Click Add.



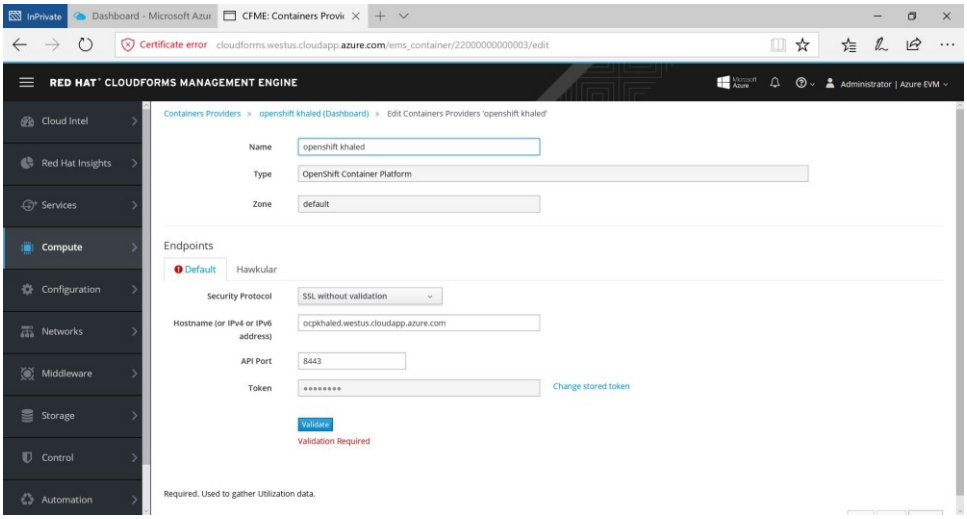


34. To add OpenShift provider
    - Navigate to Compute → Containers → Providers.
    - Click (Configuration), then click (Add Existing Containers Provider).

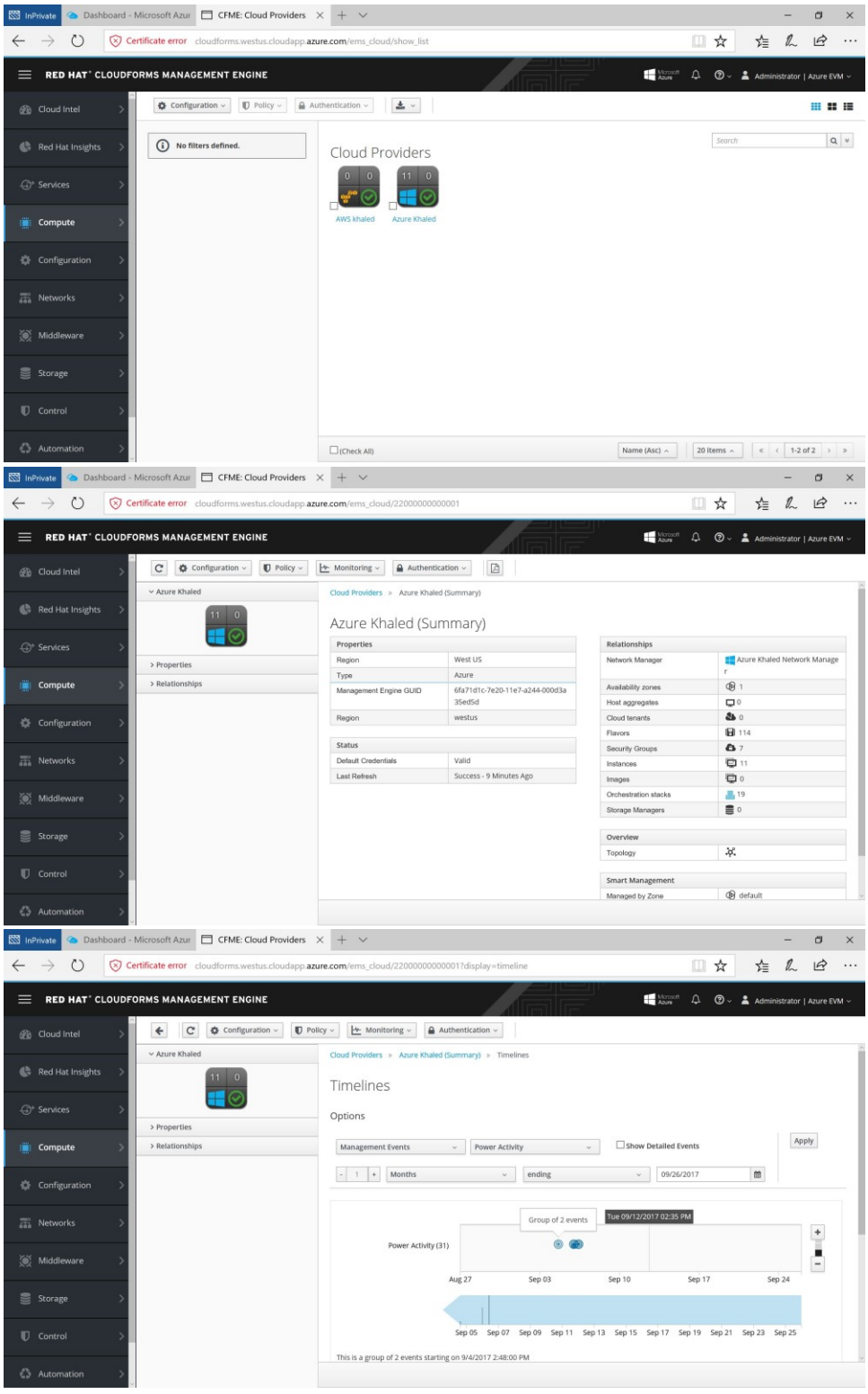
- Enter a Name for the provider.
- From the Type list, select OpenShift Container Platform.
- Enter the appropriate Zone for the provider. If you do not specify a zone, it is set to default.
- Under Endpoints in the Default tab, configure the following for the OpenShift provider:
    - Select a Security Protocol method to specify how to authenticate the provider: Choose SSL without validation
    - Enter the Hostname or IPv4 of your OpenShift environment and keep default port
    - Run the following to obtain the token needed to add an OpenShift Container Platform

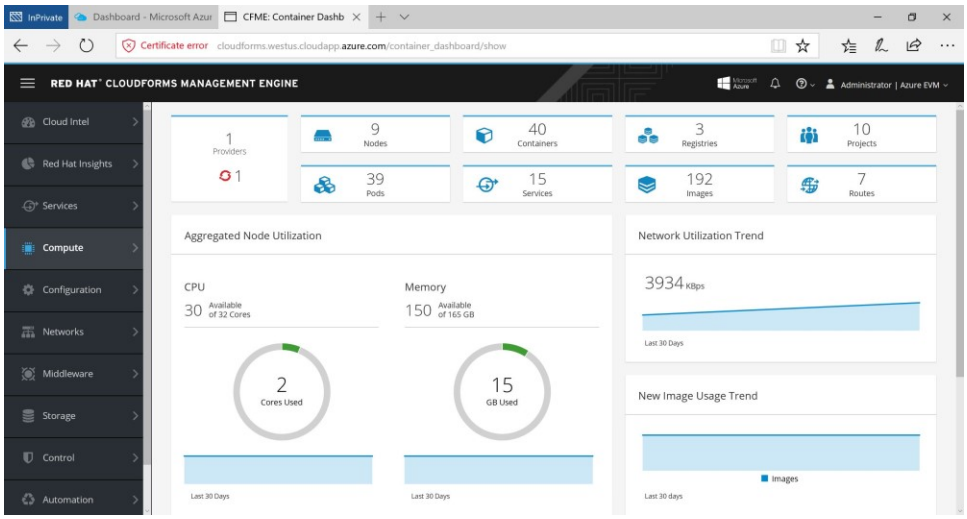
```
# oc sa get-token -n management-infra management-admin
eyJhbGciOiJSUzI1NiI...
```

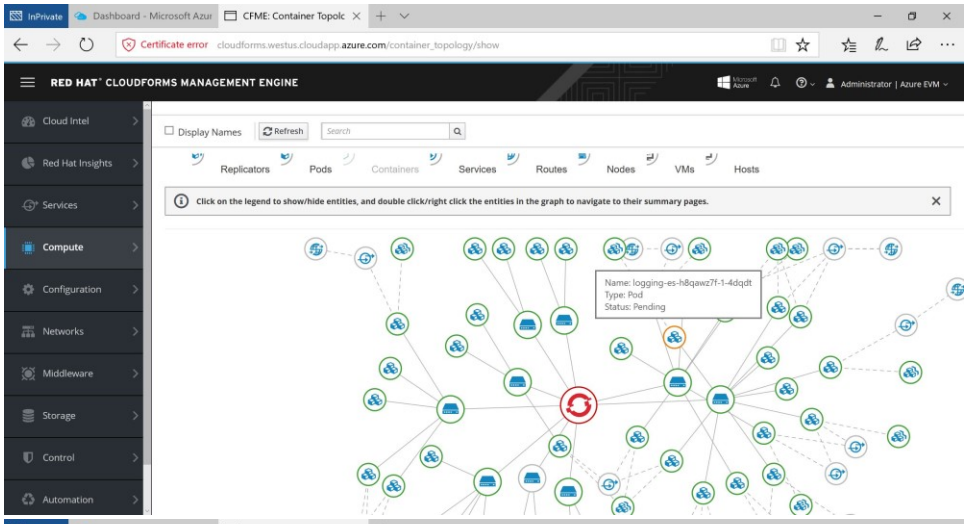
    - Enter the OpenShift management token in the Token field.
    - Enter the same token in the Confirm Token field.
    - Click Validate to confirm that Red Hat CloudForms can connect to the OpenShift Container Platform provider.

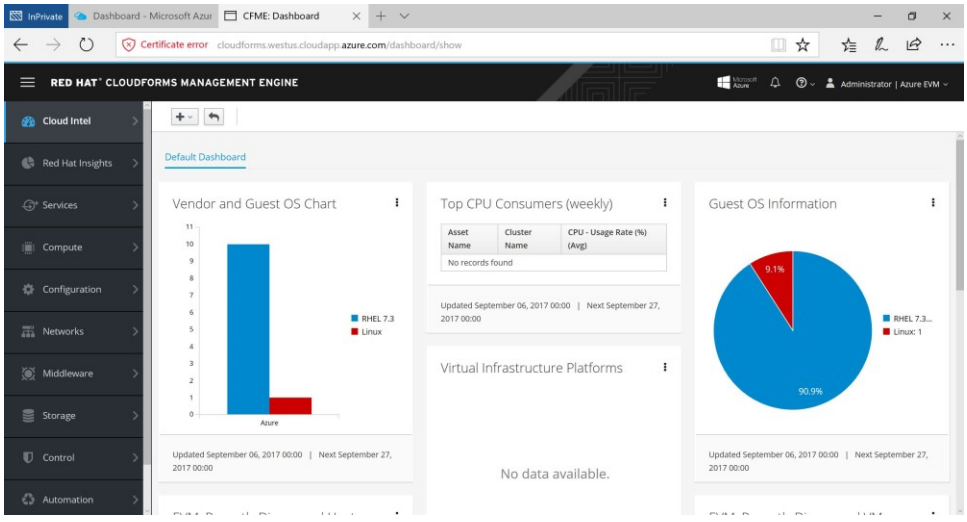


35. Explore CloudForms by navigating the left menu to get an idea on the insights and intelligence provided by the solution.


InPrivate
Dashboard - Microsoft Azur
CFME: Cloud Providers
Certificate error
cloudforms.westus.cloudapp.azure.com/ems_cloud/show_list
RED HAT CLOUDFORMS MANAGEMENT ENGINE
Administrator | Azure EVM
Cloud Intel
Red Hat Insights
Services
Compute
Configuration
Networks
Middleware
Storage
Control
Automation
Configuration
Policy
Authentication
No filters defined.
Cloud Providers
AWS khaled
Azure Khaled
(Check All)
Name (Asc)
20 Items
1-2 of 2
cloudforms.westus.cloudapp.azure.com/ems_cloud/220000000000001
Monitoring
Azure Khaled
Cloud Providers » Azure Khaled (Summary)
Azure Khaled (Summary)
Properties
Region
West US
Type
Azure
Management Engine GUID
6fa71d1c-7e20-11e7-a244-000d3a35ed5d
Region
westus
Status
Default Credentials
Valid
Last Refresh
Success - 9 Minutes Ago
Relationships
Network Manager
Azure Khaled Network Manager
Availability zones
Host aggregates
Cloud tenants
Flavors
Security Groups
Instances
Images
Orchestration stacks
Storage Managers
Overview
Topology
Smart Management
Managed by Zone
default
cloudforms.westus.cloudapp.azure.com/ems_cloud/220000000000001?display=timeline
Cloud Providers » Azure Khaled (Summary) » Timelines
Timelines
Options
Management Events
Power Activity
Show Detailed Events
Apply
Months
ending
09/26/2017
Group of 2 events
Tue 09/12/2017 02:35 PM
Power Activity (31)
Aug 27
Sep 03
Sep 10
Sep 17
Sep 24
Sep 05
Sep 07
Sep 09
Sep 11
Sep 13
Sep 15
Sep 17
Sep 19
Sep 21
Sep 23
Sep 25
This is a group of 2 events starting on 9/4/2017 2:48:00 PM


RED HAT CLOUDFORMS MANAGEMENT ENGINE
Cloud Intel
Red Hat Insights
Services
Compute
Configuration
Networks
Middleware
Storage
Control
Automation
1 Providers
1
9 Nodes
40 Containers
3 Registries
10 Projects
39 Pods
15 Services
192 Images
7 Routes
Aggregated Node Utilization
CPU
30 Available of 32 Cores
2 Cores Used
Memory
150 Available of 165 GB
15 GB Used
Network Utilization Trend
3934 KBps
Last 30 Days
New Image Usage Trend
Images
Last 30 days


RED HAT CLOUDFORMS MANAGEMENT ENGINE
Display Names
Refresh
Search
Replicators
Pods
Containers
Services
Routes
Nodes
VMs
Hosts
Click on the legend to show/hide entities, and double click/right click the entities in the graph to navigate to their summary pages.
Name: logging-es-h8qawz7f-1-4dqdt
Type: Pod
Status: Pending


RED HAT CLOUDFORMS MANAGEMENT ENGINE
Default Dashboard
Vendor and Guest OS Chart
RHEL 7.3
Linux
Azure
Top CPU Consumers (weekly)
Asset Name
Cluster Name
CPU - Usage Rate (%) (Avg)
No records found
Updated September 06, 2017 00:00 | Next September 27, 2017 00:00
Virtual Infrastructure Platforms
No data available.
Guest OS Information
9.1%
90.9%
RHEL 7.3...
Linux: 1

END THE LAB

To end the lab, simply delete the resource group *openshiftRG* from the Azure portal or from the Azure CLI. And delete the created *webhook* from your *git* repository.

```
$ az group delete ossdemo
```

REFERENCES

USEFUL LINKS

https://access.redhat.com/documentation/en-us/reference_architectures/2017/pdf/deploying_red_hat_openshift_container_platform_3.5_on_microsoft_azure/Reference_Architectures-2017-Deploying_Red_Hat_OpenShift_Container_Platform_3.5_on_Microsoft_Azure-en-US.pdf

https://blogs.technet.microsoft.com/msoms/2017/08/04/container-monitoring-solution-in-red-hat-openshift/

https://testdrive.azure.com/#/test-drive/redhat.openshift-test-drive

https://github.com/Microsoft/openshift-container-platform

https://github.com/Microsoft/openshift-origin

https://access.redhat.com/documentation/en-us/red_hat_cloudforms/4.5/html/installing_red_hat_cloudforms_on_microsoft_azure/

http://manageiq.org/

REDHAT AND MICROSOFT PARTNERSHIP

http://openness.microsoft.com/2016/04/15/microsoft-red-hat-partnership-accelerating-partner-opportunities/

https://www.redhat.com/en/microsoft