DEPLOY AND SCALE MICROSOFT AZURE CLOUD NATIVE INFRASTRUCTURES AND APPLICATIONS WITH RED HAT ANSIBLE AUTOMATION

Stuart Kirk & Zim Kalinowski
MICROSOFT CORPORATION One Microsoft Way, Redmond, WA, 98052

THIS PAGE INTENTIONALLY LEFT BLANK

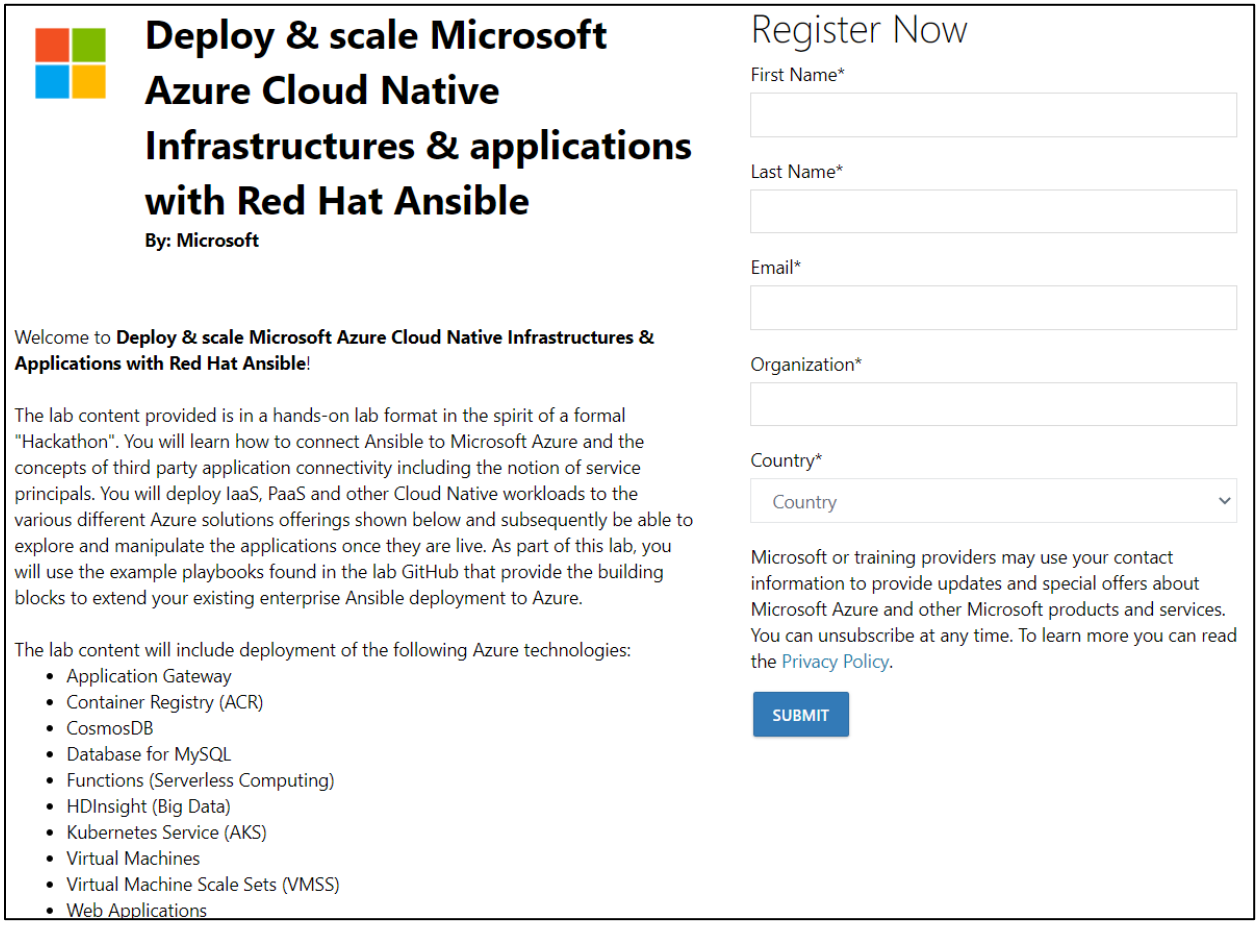
TABLE OF CONTENTS

OBTAINING & PREPARING YOUR LAB VM

Obtaining Your Lab Environment

Your Lab VM is provided by a content delivery system managed by Spektra Systems. To obtain your credentials, you must register for one of the pre-provisioned Lab VMs. Each Lab VM RHEL 8.1 with GUI pre-installed.

- Visit **https://aka.ms/azuresummitlab** in your web browser
- Complete the registration form using your CORPORATE email credentials and select the "SUBMIT" button



Deploy & scale Microsoft Azure Cloud Native Infrastructures & applications with Red Hat Ansible

By: Microsoft

Welcome to **Deploy & scale Microsoft Azure Cloud Native Infrastructures & Applications with Red Hat Ansible**!

The lab content provided is in a hands-on lab format in the spirit of a formal "Hackathon". You will learn how to connect Ansible to Microsoft Azure and the concepts of third party application connectivity including the notion of service principals. You will deploy IaaS, PaaS and other Cloud Native workloads to the various different Azure solutions offerings shown below and subsequently be able to explore and manipulate the applications once they are live. As part of this lab, you will use the example playbooks found in the lab GitHub that provide the building blocks to extend your existing enterprise Ansible deployment to Azure.

The lab content will include deployment of the following Azure technologies:

- Application Gateway
- Container Registry (ACR)
- CosmosDB
- Database for MySQL
- Functions (Serverless Computing)
- HDInsight (Big Data)
- Kubernetes Service (AKS)
- Virtual Machines
- Virtual Machine Scale Sets (VMSS)
- Web Applications

Register Now

First Name*

Last Name*

Email*

Organization*

Country*

Country

Microsoft or training providers may use your contact information to provide updates and special offers about Microsoft Azure and other Microsoft products and services. You can unsubscribe at any time. To learn more you can read the Privacy Policy.

SUBMIT

Accessing your Lab VM via VNC

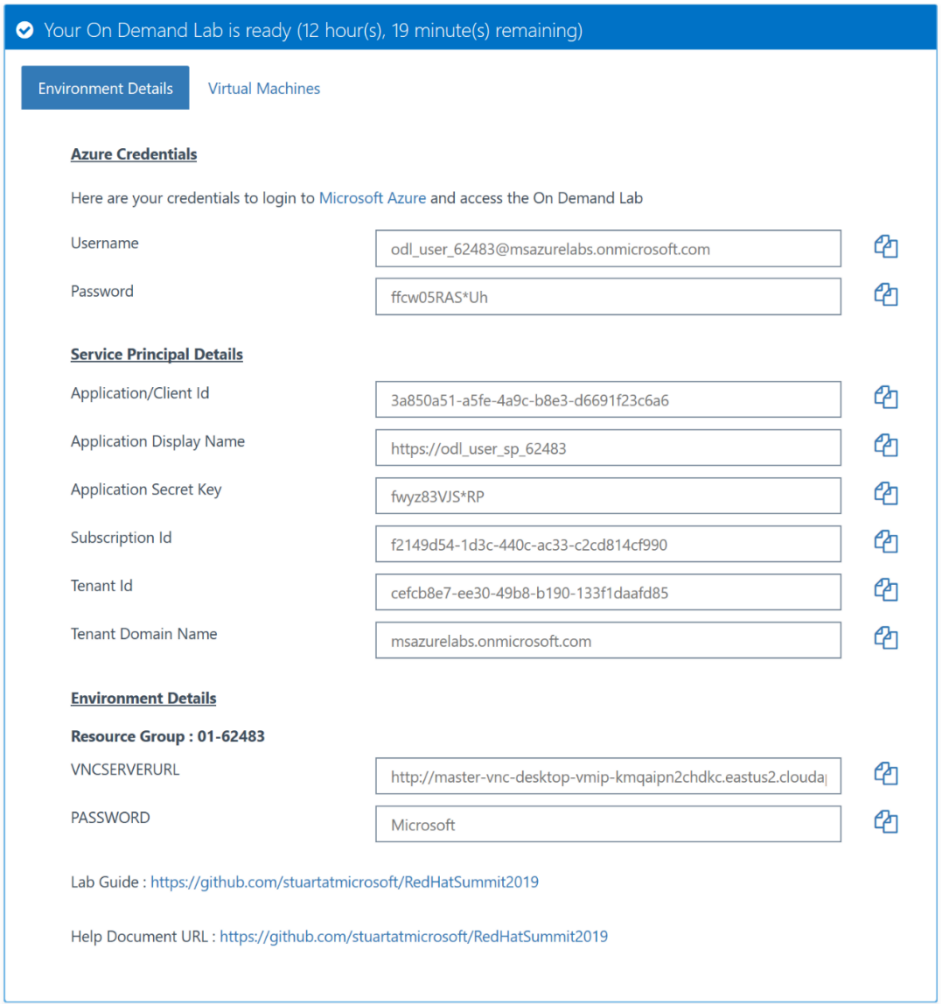
Access to the Lab VM is provided by **two** methods:

- **noVNC** - an HTML5-based VNC client
- **SSH** – connecting directly to the VM via SSH on port 2112

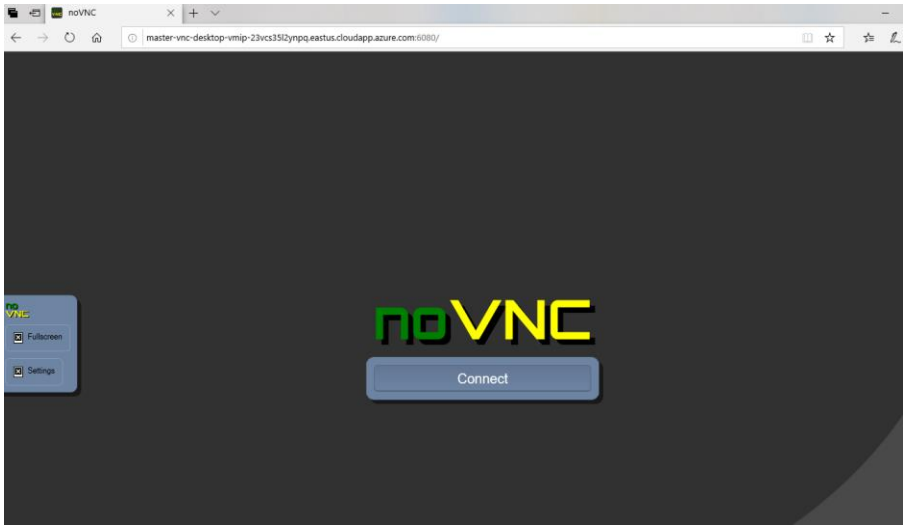
If you wish to connect using **VNC** to obtain a virtual desktop, continue with the instructions below in this section. If you wish you connect using **SSH**, **please jump to the next section**.

Upon registration submission, you should receive the required credentials to access your Lab VM via VNC. All credential information is present in a text file on your desktop, so only record the three values below.

- Obtain the VNCSERVERURL web URL
- Obtain the password; The password for all accounts is: **Microsoft**
- Take note of the resource-group which is assigned to you

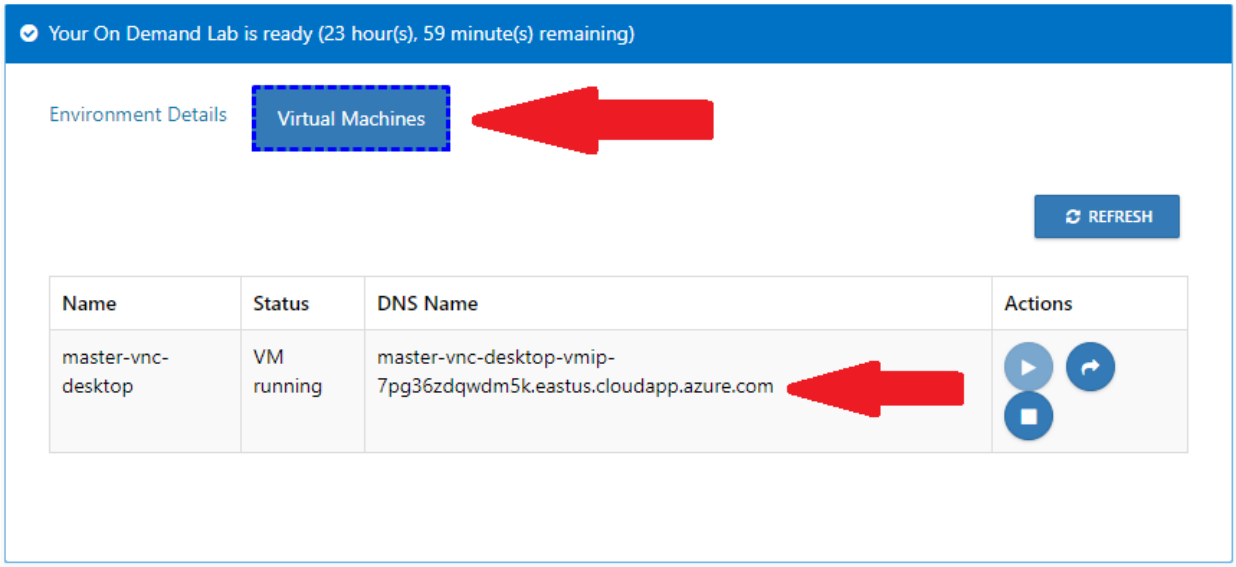


- Enter the VNCSERVERURL into your web browser and log in to the VNC Lab VM



Accessing your Lab VM via SSH

Determine the FQDN of your host to SSH to by clicking the "Virtual Machines" tab on the Spektra Systems registration page. SSH to this host **ON PORT 2112** as follows:

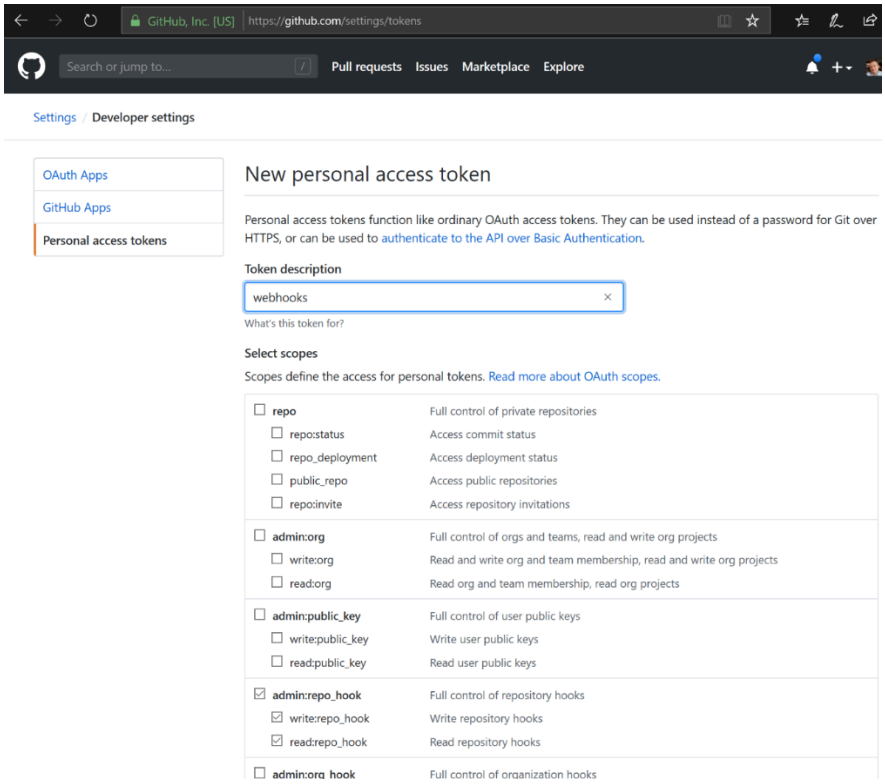


- `ssh student@master-vnc-desktop-vmip-7pg36zdqwdm5k.eastus.cloudapp.azure.com -p 2112`
- The password for SSH and all other accounts is: **Microsoft**

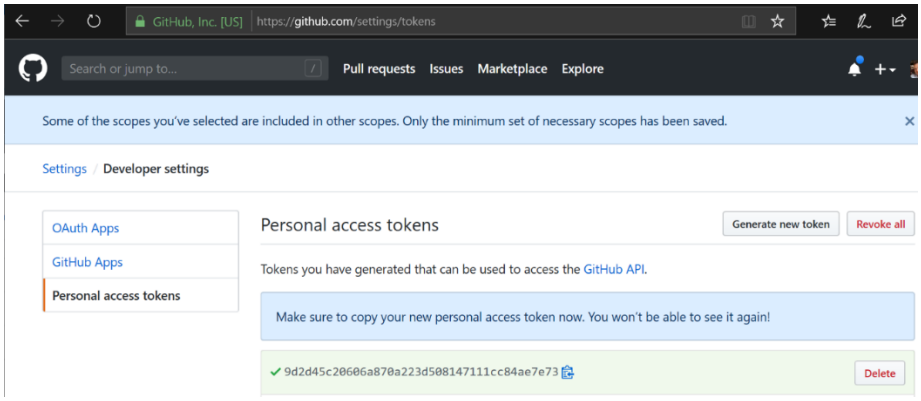
Generate a GitHub Personal Access Token

For Lab #5 (Azure Functions / Serverless) you will need to have a GitHub account and a personal access token.

- Log in to your GitHub account
- Visit: https://github.com/settings/tokens
- Click on "**Generate new token**"
- Give the token any description you wish and select "**admin:repo_hook**" as the scope for the token.



- Take note of the token after it is generated!



Login to the Azure Linux CLI when using VNC and SSH

VNC:

- Open the credentials.txt file on your desktop
- AZURE_USER_NAME is your Azure Linux CLI & Portal Username
- AZURE_USER_PASSWORD is your Azure Linux CLI & Portal Password
- **az login**

SSH:

- View the credentials file on your desktop: **cat Desktop/credentials.txt**
- AZURE_USER_NAME is your Azure Linux CLI & Portal Username
- AZURE_USER_PASSWORD is your Azure Linux CLI & Portal Password
- **az login -u odl_user_1234@something.onmicrosoft.com -p 5om3-cr4zyPa$$w0rd!**

```
[student@master-vnc-desktop ~]$ cat Desktop/credentials.txt
AZURE_USER_NAME=odl_user_175872@azurehol1119.onmicrosoft.com
AZURE_USER_PASSWORD=iswa46ZVL*XP
AZURE_CLIENT_ID=f2c2b32e-e8cd-4da9-b619-5d488ec2867a
AZURE_SECRET=kqzn26QXQ*MT
AZURE_SUBSCRIPTION_ID=c4a52ec2-51a8-437b-9965-61708549d2d4
AZURE_TENANT_ID=dc079525-eacb-4739-80c4-30481e7543a8
GUIDE_URL=https://github.com/stuartatmicrosoft/RedHatSummit2020
[student@master-vnc-desktop ~]$ az login -u odl_user_175872@azurehol1119.onmicrosoft.com -p iswa46ZVL*XP
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "dc079525-eacb-4739-80c4-30481e7543a8",
    "id": "c4a52ec2-51a8-437b-9965-61708549d2d4",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Azure HOL 1119",
    "state": "Enabled",
    "tenantId": "dc079525-eacb-4739-80c4-30481e7543a8",
    "user": {
      "name": "odl_user_175872@azurehol1119.onmicrosoft.com",
      "type": "user"
    }
  }
]
[student@master-vnc-desktop ~]$
```

In VNC, the login process will likely open a web browser which will prompt you to enter your username/password credentials. Close the web browser when prompted to do so. For SSH, you should be logged in immediately as shown above.

The default output format of the Azure Linux CLI is JSON. It is recommended that you change your default output to "table" format.

- **az configure**
- Choose "y"es to change options
- Choose "3" – Table Format
- Configure other options as you wish

Login to the Azure Portal

The Azure Portal provides a GUI-based environment to access the entire Azure platform. During your lab exercises, it is recommended that while Ansible playbooks are running that you view what activity is transpiring in the Azure Portal as resources are configured. This can be done by accessing your assigned "Resource Group" and clicking the "Refresh" button. We would suggest always keeping a browser window open to the Azure Portal.

- Visit https://portal.azure.com
- Login to the Azure portal using the **AZURE_USER_NAME** and **AZURE_USER_PASSWORD**

Obtaining & Preparing your Labs

- Open a shell on your RHEL 8.1 terminal or connect to your VM via SSH
- **`git clone https://github.com/stuartatmicrosoft/RedHatSummit2020`**
- **`cd RedHatSummit2020/playbooks`**

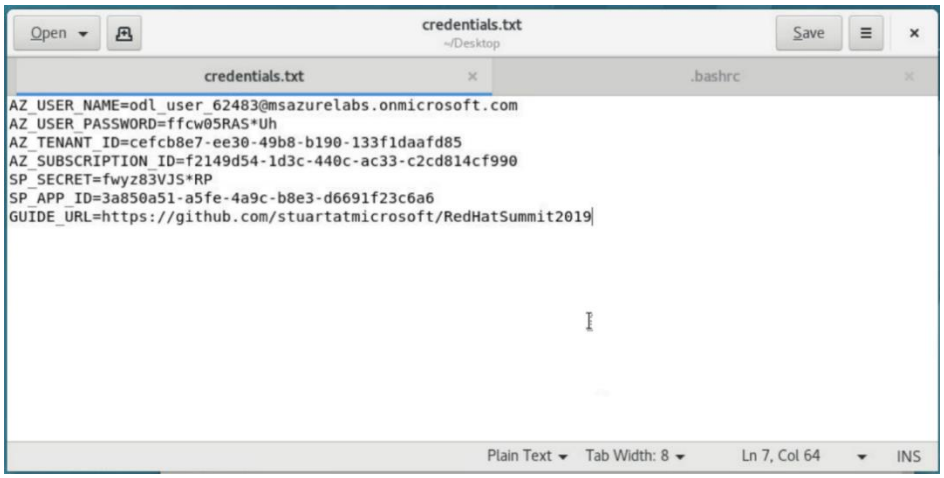
To begin working through the lab exercises, you will need to generate your own Ansible variables file. Perform the following to generate your variables file:

- **`chmod 755 lab-build.sh`**
- **`./lab-build.sh`**

```
What is your first name:  > Stuart
What is your last name:  > Kirk
What year were you born:  > 1975
What is your GitHub ID:  > stuartatmicrosoft
What is your GitHub Personal Access Token:  > asdf
What is the Azure Red Hat OpenShift (ARO) API URL:  > https://api.xvoqh9s9.eastus.aroapp.io:6443

To Recap:
Your first name is: Stuart
Your last name is: Kirk
You were born in: 1975
Your GitHub ID is: stuartatmicrosoft
Your GitHub PAT is: asdf
The ARO API URL is: https://api.xvoqh9s9.eastus.aroapp.io:6443

Is this correct?
1) Yes
2) No
Select a numbered option >> 1

**********************************************************************************
**********************************************************************************
**********************************************************************************
 For the Azure Red Hat OpenShift (ARO) lab:
 Your username to log in to ARO is: skirk44703
 Your password to log in to ARO is: Microsoft
**********************************************************************************
**********************************************************************************
**********************************************************************************
>>>>> The ARO cluster has received your login credentials

Your custom variables file, vars-myvars.yml, has been created.  Go forth and conquer!
[student@master-vnc-desktop playbooks]$
```

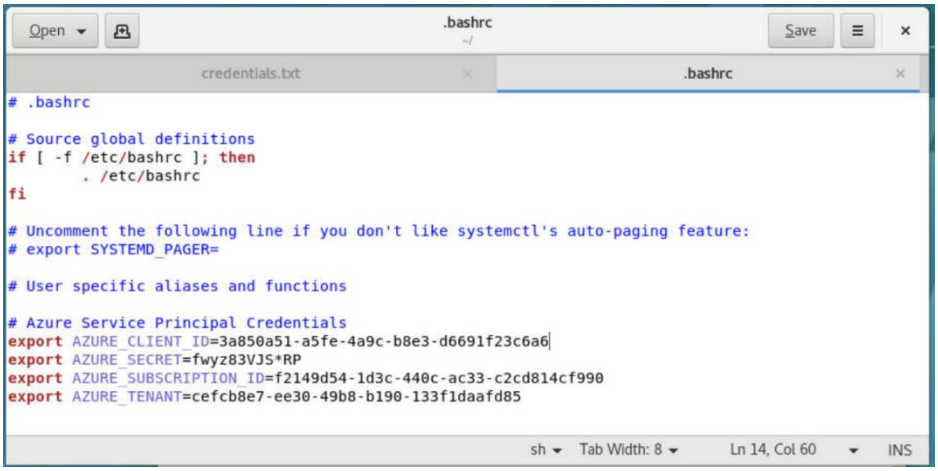
Environment variables in your ~/.bashrc are set with the data for the service principal which you have been assigned. A service principal is a login mechanism for external applications to access a specific set of Microsoft Azure resources within a subscription. It is akin to a "service account" on a Linux host. View your ~/.bashrc file and verify the service principal credential information:

- Open the credentials.txt file on your desktop



- Verify the following values into your ~/.bashrc from credentials.txt

.bashrc & credentials.txt
AZURE_CLIENT_ID
AZURE_SECRET
AZURE_SUBSCRIPTION_ID
AZURE_TENANT



- Do not forget to `source .bashrc` or close and re-open your current shell window in the event you made any changes to these values!

Install the Azure Modules for Ansible

Ansible 2.9.6 is already pre-installed on your Lab VM. We are assuming that in most cases, you know how to install Ansible and have already done so in your enterprises. To provide connectivity for Ansible to Microsoft Azure, you must install the Azure modules for Ansible:

- **pip-2.7 install --user ansible[azure]**

```
Activate the web console with: systemctl enable --now cockpit.socket

This system is not registered to Red Hat Insights. See https://cloud.redhat.com/
To register this system, run: insights-client --register

Last login: Sun Apr 12 17:34:39 2020
[student@master-vnc-desktop ~]$ clear
[student@master-vnc-desktop ~]$ pip2.7 install --user ansible[azure]
Requirement already satisfied: ansible[azure] in ./.local/lib/python2.7/site-packages
Requirement already satisfied: PyYAML in ./.local/lib/python2.7/site-packages (from ansible[az
Requirement already satisfied: jinja2 in ./.local/lib/python2.7/site-packages (from ansible[az
Requirement already satisfied: cryptography in ./.local/lib/python2.7/site-packages (from ansi
Collecting azure-mgmt-storage==3.1.0; extra == "azure" (from ansible[azure])
  Downloading https://files.pythonhosted.org/packages/e8/d9/496b29857a252bc3fcc4bbda069c0eb64b
0-py2.py3-none-any.whl (696kB)
    100% |████████████████████████████████| 706kB 1.6MB/s
Collecting azure-common==1.1.11; extra == "azure" (from ansible[azure])
  Downloading https://files.pythonhosted.org/packages/97/3b/2c7cda25382c3bb566008c5c8f8aa28663
.py3-none-any.whl
Collecting azure-graphrbac==0.40.0; extra == "azure" (from ansible[azure])
  Downloading https://files.pythonhosted.org/packages/89/0a/29f7e2914033e2536026b8f0d7f8deb1ed
py2.py3-none-any.whl (63kB)
    100% |████████████████████████████████| 71kB 10.7MB/s
Collecting packaging; extra == "azure" (from ansible[azure])
  Downloading https://files.pythonhosted.org/packages/62/0a/34641d2bf5c917c96db0ded85ae4da25b6
none-any.whl
Collecting azure-mgmt-iothub==0.7.0; extra == "azure" (from ansible[azure])
  Downloading https://files.pythonhosted.org/packages/9c/c8/333e4f03eef95832f90534c2aea3b6809c
```

This set of python modules includes all the current Azure modules for Ansible. For modules under development, there is a set of preview modules which can be installed. They are on GitHub: https://github.com/Azure/azure_preview_modules

Upon successful installation of all the modules, you should expect output like:

```
Successfully built tabulate scandir
Installing collected packages: azure-nspkg, azure-mgmt-nspkg, typing, isodate, msrest, PyJWT,
restazure, azure-common, azure-mgmt-storage, azure-graphrbac, pyparsing, packaging, azure-mgmt
 azure-mgmt-hdinsight, azure-mgmt-containerregistry, azure-mgmt-devtestlabs, azure-mgmt-batch,
gmt-loganalytics, azure-mgmt-web, azure-storage, azure-mgmt-redis, azure-mgmt-cdn, azure-mgmt-
zure-keyvault, azure-mgmt-marketplaceordering, azure-mgmt-compute, azure-mgmt-dns, azure-mgmt-
 azure-mgmt-resource, azure-mgmt-keyvault, azure-mgmt-authorization, azure-mgmt-servicebus, co
er, contextlib2, zipp, scandir, pathlib2, importlib-metadata, argcomplete, colorama, azure-cli
pygments, tabulate, jmespath, knack, wheel, pynacl, bcrypt, paramiko, applicationinsights, mon
humanfriendly, azure-cli-core, azure-mgmt-containerinstance, azure-mgmt-cosmosdb, azure-mgmt-c
service, azure-mgmt-automation, azure-mgmt-sql, azure-mgmt-trafficmanager, azure-mgmt-monitor
  Found existing installation: wheel 0.34.2
    Uninstalling wheel-0.34.2:
      Successfully uninstalled wheel-0.34.2
Successfully installed PyJWT-1.7.1 adal-1.2.2 applicationinsights-0.11.9 argcomplete-1.11.1 az
core-2.0.35 azure-cli-nspkg-3.0.2 azure-common-1.1.11 azure-graphrbac-0.40.0 azure-keyvault-1.
ure-mgmt-authorization-0.51.1 azure-mgmt-automation-0.1.1 azure-mgmt-batch-5.0.1 azure-mgmt-cd
azure-mgmt-compute-4.4.0 azure-mgmt-containerinstance-1.4.0 azure-mgmt-containerregistry-2.0.0
gmt-containerservice-4.4.0 azure-mgmt-cosmosdb-0.5.2 azure-mgmt-devtestlabs-3.0.0 azure-mgmt-d
 azure-mgmt-hdinsight-0.1.0 azure-mgmt-iothub-0.7.0 azure-mgmt-keyvault-1.1.0 azure-mgmt-logan
0.2.0 azure-mgmt-marketplaceordering-0.1.0 azure-mgmt-monitor-0.5.2 azure-mgmt-network-2.3.0 a
t-nspkg-2.0.0 azure-mgmt-rdbms-1.4.1 azure-mgmt-redis-5.0.0 azure-mgmt-resource-2.1.0 azure-mg
cebus-0.5.3 azure-mgmt-sql-0.10.0 azure-mgmt-storage-3.1.0 azure-mgmt-trafficmanager-0.50.0 az
-web-0.41.0 azure-nspkg-2.0.0 azure-storage-0.35.1 bcrypt-3.1.7 colorama-0.4.3 configparser-4.
extlib2-0.6.0.post1 humanfriendly-8.1 importlib-metadata-1.6.0 isodate-0.6.0 jmespath-0.9.5 kn
3 monotonic-1.5 msrest-0.6.1 msrestazure-0.5.0 packaging-20.3 paramiko-2.7.1 pathlib2-2.3.5 py
.5.2 pynacl-1.3.0 pyparsing-2.4.7 scandir-1.10.0 tabulate-0.8.2 typing-3.7.4.1 wheel-0.30.0 zi
[student@master-vnc-desktop ~]$
```

You are now ready to go!!!

LAB 1 – AZURE INFRASTRUCTURE & PLATFORM SERVICES

Summary of Lab

This lab is intended to provide infrastructure administrators with end-to-end provisioning skills for deploying scalable IaaS and PaaS resources in Microsoft Azure. The following Ansible playbooks/instructions will deploy and IaaS infrastructure and a MySQL-based PaaS database. A separate playbook will be used to install Mattermost (comparable to Slack) on the infrastructure node at which time you will test the operation of the service and verify it is functioning as expected. After verification and to enable scaling, we will shut down the infrastructure node, generalize it and deploy an Azure Virtual Machine Scale Set (VMSS). This service allows infrastructure nodes to automatically be rapidly allocated/deallocated as required for demand. As a VMSS requires a disk image to deploy from, we will use the disk image which was created on the single IaaS node to create the VMSS. To front-end the application we will implement Azure Application Gateway (AG). AG supports URL-based routing, multi-site routing, cookie-based session affinity and a web application firewall.

Playbook 0 – Preparing the Application Gateway

Estimated Playbook Runtime: 16m 25s

- `ansible-playbook mm-00-prerequisites.yml`

Playbook 1 – Deploying the Infrastructure Node

Estimated Playbook Runtime: 2m 53s

- `ansible-playbook mm-01-vm-deploy.yml`

Playbook 2 – Deploying MySQL PaaS

Estimated Playbook Runtime: 3m 39s

- `ansible-playbook mm-02-create-mysql.yml`

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook mm-02-create-mysql.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit
localhost does not match 'all'

PLAY [Deploy Mattermost MySQL PaaS Database] ************************************************

TASK [Gathering Facts] **********************************************************************
ok: [localhost]

TASK [Create MySQL Server for Mattermost Database] ******************************************
changed: [localhost]

TASK [Create instance of MySQL Database] ****************************************************
changed: [localhost]

TASK [Getting Public IP address of the application VM] **************************************
ok: [localhost]

TASK [Add MySQL Firewall Rule] **************************************************************
changed: [localhost]

PLAY RECAP **********************************************************************************
localhost                  : ok=5    changed=3    unreachable=0    failed=0    skipped=0    re
scued=0    ignored=0


real    3m42.988s
user    0m23.152s
sys     0m2.124s
[student@master-vnc-desktop playbooks]$
```

Now would be a good time to look at the Azure Portal to see the resources which have been created!

Playbook 3 – Deploying & Configuring Mattermost

Estimated Playbook Runtime: 0m 59s

- `ansible-playbook mm-03-setup-mattermost.yml`

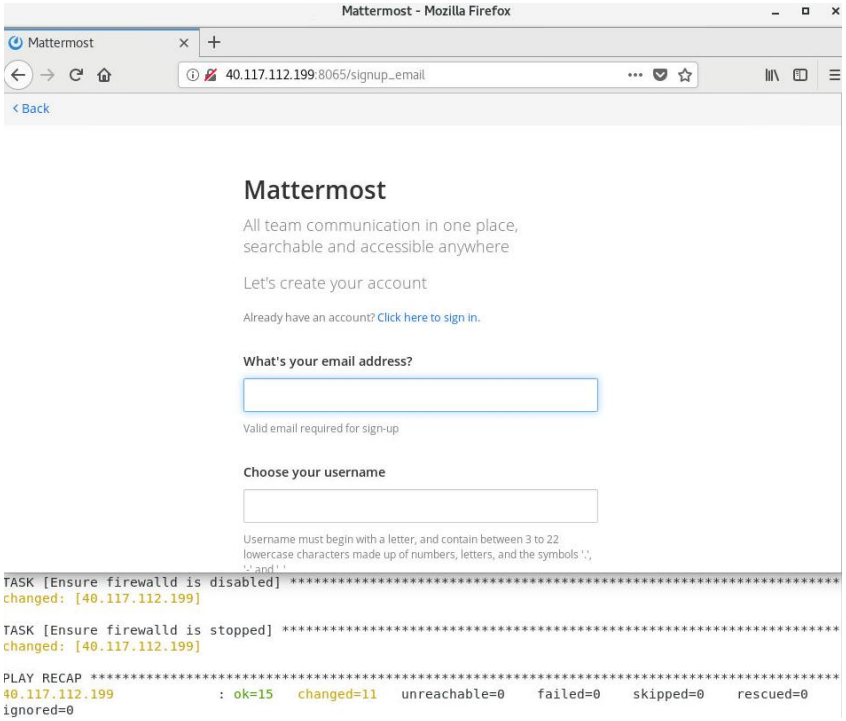
```
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit
localhost does not match 'all'

PLAY [Determine IP address of Mattermost VM] ***********************************************

TASK [Gathering Facts] *********************************************************************
ok: [localhost]

TASK [Get Mattermost Public IP Address Information] ****************************************
ok: [localhost]

TASK [Adding public IP to playbook group] **************************************************
changed: [localhost]

TASK [Saving FQDN of virtual machine] ******************************************************
ok: [localhost]

PLAY [Install Mattermost Server] ***********************************************************

TASK [Gathering Facts] *********************************************************************
ok: [13.68.227.103]

TASK [Create Mattermost User] **************************************************************
changed: [13.68.227.103]

TASK [Download and unpack MatterMost tarball from central repository server] ***************
changed: [13.68.227.103]

TASK [Create Mattermost storage directory] *************************************************
ok: [13.68.227.103]

TASK [Create Mattermost data directory] ****************************************************
changed: [13.68.227.103]

TASK [Ensure Mattermost application is owned by mattermost user] ***************************
changed: [13.68.227.103]

TASK [Ensure sticky bit is set on Mattermost application directories] **********************
changed: [13.68.227.103]

TASK [Configure Mattermost Data Source] ****************************************************
changed: [13.68.227.103]

TASK [Downloading systemd service script for Mattermost] ***********************************
changed: [13.68.227.103]

TASK [Force systemd to re-read configuration] **********************************************
ok: [13.68.227.103]

TASK [Enable Mattermost application within systemd] ****************************************
changed: [13.68.227.103]

TASK [Start Mattermost application within systemd] *****************************************
changed: [13.68.227.103]

TASK [Ensure firewalld is installed] *******************************************************
ok: [13.68.227.103]

TASK [Ensure firewalld is disabled] ********************************************************
changed: [13.68.227.103]

TASK [Ensure firewalld is stopped] *********************************************************
changed: [13.68.227.103]

TASK [Dump Mattermost FQDN] ****************************************************************
ok: [13.68.227.103] => {
    "msg": "Mattermost Server FQDN: http://stuartkirk1975-mattermost-47428.eastus.cloudapp.azu
re.com:8065"
}
```

Test the single node Mattermost Application

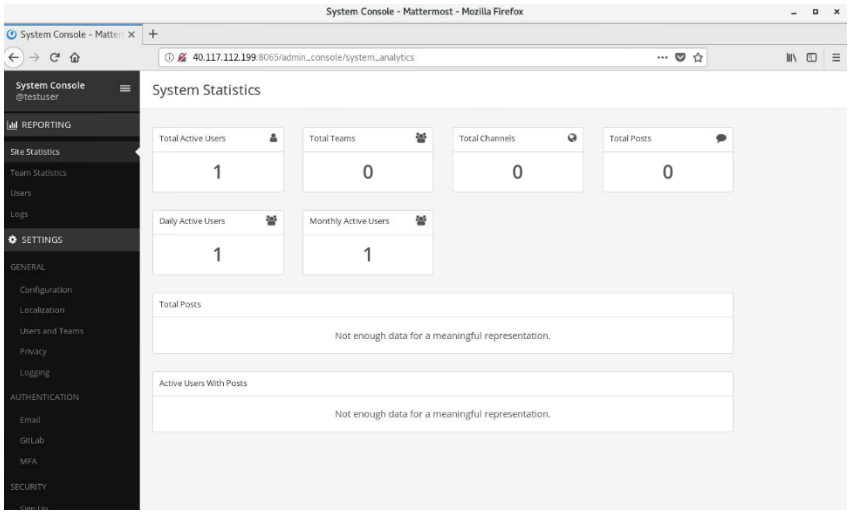
The Mattermost application should be available for you to test in a single-node configuration. Observing the FQDN from the playbook just executed, attempt to gain access to the Mattermost application, create a user account and view the administrative portal. Remember to access the service on port 8065! The URL should be displayed on your screen after the playbook is completed.

- Visit http://x.x.x.x:8065

Initial Startup Screen:



Administrative Console:



Playbook 4 – Generalizing & Creating a VM Disk Image

Estimated Playbook Runtime: 1m 8s

- `ansible-playbook mm-04-create-vm-image.yml`

Playbook 5 – Creating a Virtual Machine Scale Set (VMSS)

Estimated Playbook Runtime: 3m 52s

- `ansible-playbook mm-05-vmss-create.yml`

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook mm-05-vmss-create.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does
not match 'all'

PLAY [Create Virtual Machine Scale Set for Mattermost] *************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [localhost]

TASK [Add VMSS Subnet to existing Virtual network] *****************************************************
changed: [localhost]

TASK [Create Public IP Address for VMSS] ***************************************************************
changed: [localhost]

TASK [Create Load Balancer for Mattermost VMSS] ********************************************************
changed: [localhost]

TASK [Create VMSS] *************************************************************************************
changed: [localhost]

TASK [Getting Public IP address of the LB] *************************************************************
ok: [localhost]

TASK [Add MySQL Firewall Rule] *************************************************************************
changed: [localhost]

PLAY RECAP *********************************************************************************************
localhost                  : ok=7    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    igno
red=0


real    2m51.051s
user    0m21.616s
sys     0m2.331s
[student@master-vnc-desktop playbooks]$
```

Playbook 6 – Attaching the AG to the VMSS

Estimated Playbook Runtime: 5m 56s

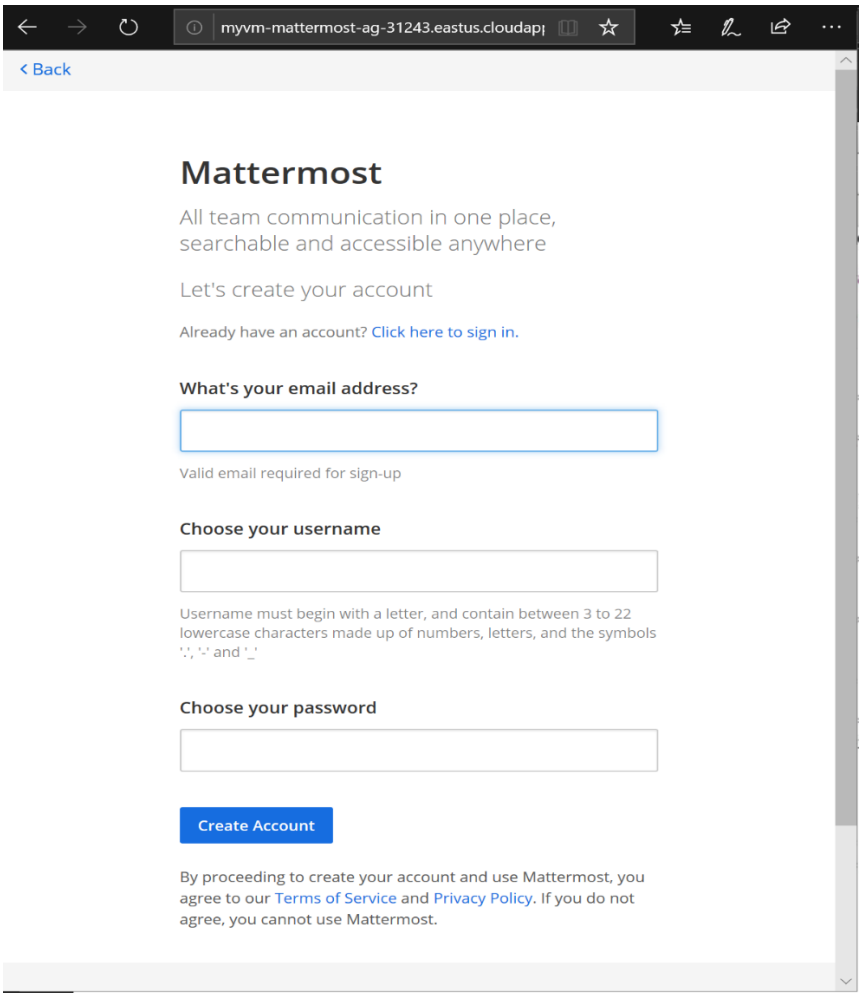
- `ansible-playbook mm-06-appgateway-attach.yml`

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook mm-06-appgateway-attach.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does
not match 'all'

PLAY [Connect Mattermost Application Gateway to VMSS] ****************************************************

TASK [Gathering Facts] ***********************************************************************************
ok: [localhost]

TASK [Update VMSS to use Application Gateway instead of Load Balancer] ***********************************
changed: [localhost]

TASK [Getting FQDN of the Application Gateway] ***********************************************************
ok: [localhost]

TASK [Dump Application Gateway Public IP] ****************************************************************
ok: [localhost] => {
    "msg": "Mattermost App Gateway FQDN: http://stuartkirk1975-mattermost-ag-47428.eastus.cloudapp.azure.com"
}

PLAY RECAP ***********************************************************************************************
localhost                  : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    igno
red=0


real    4m55.235s
user    0m28.112s
sys     0m2.097s
[student@master-vnc-desktop playbooks]$
```

Test the Mattermost Application using the Azure AG

The AG will take a few minutes to fully connect to the VMSS. After waiting a few minutes, visit the URL of the AG as provided by the final playbook which connected it to the VMSS. Notice that in this case you no longer need to specify port 8065 when connecting since the AG will provide the port mapping. Why not also refresh the list of Azure services which you have deployed in your resource group in the Azure Portal?

- Visit http://<X>-mattermost-ag-<number>.eastus/southcentralus.cloudapp.azure.com



LAB 2 – INFINIBAND & HIGH-PERFORMANCE COMPUTING ON AZURE

Summary of Lab

This lab demonstrates the ability of Azure Virtual Machines to support HPC applications and workloads that require parallel processing environments / low latency interconnects. The lab will deploy a single master NFS server that will act as the common storage repository for all HPC nodes. For the worker nodes, three Azure "Standard_A8" virtual machines will be deployed with Infiniband interconnects. Each worker node will have an NFS mount back to the master server and have home directories and a common workspace shared across all nodes. You will be able to perform latency tests (measured in microseconds) using both TCP and Infiniband connections

Playbook 0 – Deploy the HPC Cluster Master NFS Share VM

Estimated Playbook Runtime: 3m 13s

- `ansible-playbook hpc-00-cluster-master-deploy.yml`

Playbook 1 – Configure the HPC Cluster Master NFS Share VM

Estimated Playbook Runtime: 2m 53s

- `ansible-playbook hpc-01-cluster-master-configure.yml`

```
TASK [Ensure mdmonitor is enabled] *********************************************************************
ok: [40.121.59.41]

TASK [Ensure mdmonitor is started] *********************************************************************
changed: [40.121.59.41]

TASK [Mounting /share/data and creating required /etc/fstab entry] *************************************
changed: [40.121.59.41]

TASK [Creating hpc group] ******************************************************************************
changed: [40.121.59.41]

TASK [Creating hpc user] *******************************************************************************
changed: [40.121.59.41]

TASK [Setting /share/home SELinux Context] *************************************************************
changed: [40.121.59.41]

TASK [Adding NFS shares to /etc/exports] ***************************************************************
changed: [40.121.59.41] => (item=None)
changed: [40.121.59.41] => (item=None)

TASK [Re-exporting NFS shares] *************************************************************************
changed: [40.121.59.41]

TASK [Allow hpc group to have passwordless sudo] *******************************************************
changed: [40.121.59.41]

TASK [Creating /share/home/.ansible/tmp] ***************************************************************
changed: [40.121.59.41]

TASK [Generating SSH keypair for hpc user] *************************************************************
changed: [40.121.59.41]

TASK [Creating SSH configuration file for hpc user] ****************************************************
changed: [40.121.59.41] => (item=None)
changed: [40.121.59.41] => (item=None)
changed: [40.121.59.41] => (item=None)
changed: [40.121.59.41] => (item=None)

TASK [Adding SSH public key to authorized_keys to enable password-less SSH] ****************************
changed: [40.121.59.41]

TASK [Get fullpingpong.sh script] **********************************************************************
changed: [40.121.59.41]

TASK [Set use_nfs_home_dirs SELinux flag on and keep it persistent across reboots] *********************
changed: [40.121.59.41]

TASK [Restoring /share/home SELinux Context] ***********************************************************
changed: [40.121.59.41]

PLAY RECAP *********************************************************************************************
40.121.59.41               : ok=31   changed=27   unreachable=0    failed=0    skipped=0    rescued=0    igno
red=0
localhost                  : ok=5    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    igno
red=0


real    2m50.052s
user    0m29.549s
sys     0m5.229s
[student@master-vnc-desktop playbooks]$
```

Playbook 2 – Deploy a 3-Node Infiniband-capable VM Cluster

Estimated Playbook Runtime: 12m 6s

- `ansible-playbook hpc-02-cluster-compute-deploy.yml`

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook hpc-02-cluster-compute-deploy.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not
match 'all'

PLAY [Deploy HPC Compute Node and Network Infrastructure] ***********************************************

TASK [Gathering Facts] ***********************************************************************************
ok: [localhost]

TASK [Create Compute Node 1 NIC] *************************************************************************
changed: [localhost]

TASK [Create Compute Node 2 NIC] *************************************************************************
changed: [localhost]

TASK [Create Compute Node 3 NIC] *************************************************************************
changed: [localhost]

TASK [Create an availability set with default options] ***************************************************
changed: [localhost]

TASK [Create HPC Compute Node 1 VM] **********************************************************************
changed: [localhost]

TASK [Create HPC Compute Node 2 VM] **********************************************************************
changed: [localhost]

TASK [Create HPC Compute Node 3 VM] **********************************************************************
changed: [localhost]

PLAY RECAP ***********************************************************************************************
localhost                  : ok=8    changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored
=0


real    8m13.531s
user    0m47.319s
sys     0m3.720s
[student@master-vnc-desktop playbooks]$
```

Playbook 3 – Configure the Infiniband HPC Worker Nodes

Estimated Playbook Runtime: 2m 37s

- `ansible-playbook hpc-03-cluster-compute-configure.yml`

```
TASK [Ensure deltarpm package is installed] ************************************************************************************
changed: [168.61.50.216]
changed: [40.121.40.146]
changed: [23.96.46.162]

TASK [Install the EPEL repository] *********************************************************************************************
changed: [168.61.50.216]
changed: [23.96.46.162]
changed: [40.121.40.146]

TASK [Ensure required RPM packages are installed] ******************************************************************************
changed: [168.61.50.216]
changed: [40.121.40.146]
changed: [23.96.46.162]

TASK [Set use_nfs_home_dirs SELinux flag on and keep it persistent across reboots] *********************************************
changed: [168.61.50.216]
changed: [40.121.40.146]
changed: [23.96.46.162]

TASK [Mounting /share/data and creating required /etc/fstab entry] *************************************************************
changed: [168.61.50.216]
changed: [23.96.46.162]
changed: [40.121.40.146]

TASK [Mounting /share/home and creating required /etc/fstab entry] *************************************************************
changed: [168.61.50.216]
changed: [23.96.46.162]
changed: [40.121.40.146]

TASK [Creating hpc group] ******************************************************************************************************
changed: [23.96.46.162]
changed: [168.61.50.216]
changed: [40.121.40.146]

TASK [Creating hpc user] *******************************************************************************************************
changed: [168.61.50.216]
changed: [23.96.46.162]
changed: [40.121.40.146]

TASK [Allow hpc group to have passwordless sudo] *******************************************************************************
changed: [23.96.46.162]
changed: [168.61.50.216]
changed: [40.121.40.146]

PLAY [Configure MPIO using master node] ****************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [40.121.59.41]

TASK [Create /shared/home/hosts file for MPIO test] ****************************************************************************
changed: [40.121.59.41] => (item=None)
changed: [40.121.59.41] => (item=None)
changed: [40.121.59.41] => (item=None)

PLAY RECAP *********************************************************************************************************************
168.61.50.216              : ok=10   changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
23.96.46.162               : ok=10   changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
40.121.40.146              : ok=10   changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
40.121.59.41               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
localhost                  : ok=17   changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


real    2m29.280s
user    0m36.898s
sys     0m7.134s
[student@master-vnc-desktop playbooks]$
```

Perform Latency Testing Using Infiniband & TCP Connections

During the creation of the HPC Worker Nodes, you will see Ansible connecting with the three worker nodes by IP address. Connect to one of these nodes via ssh and login as the user "student". You should not be asked for a password as the Worker Nodes were created using the SSH key which was already generated when your Lab VM was built. Perform latency tests using the fullpingpong.sh script and observe the inter-node communication of eth1 using dapl (Infiniband) vs tcp

- **ssh student@x.x.x.x**
- **sudo su - hpcuser**
- **./fullpingpong.sh**

```
[student@master-vnc-desktop playbooks]$ ssh student@23.96.46.162
Last login: Mon Apr 13 01:16:44 2020 from 13.82.196.191
[student@stuartkirk1975-hpc-compute3 ~]$ sudo su - hpcuser
[hpcuser@stuartkirk1975-hpc-compute3 ~]$ ./full-pingpong.sh
################################################
NODES: 10.10.100.5, 10.10.100.5, 0.41
################################################
################################################
NODES: 10.10.100.5, 10.10.100.6, 2.53
################################################
################################################
NODES: 10.10.100.5, 10.10.100.7, 3.12
################################################
################################################
NODES: 10.10.100.6, 10.10.100.5, 2.76
################################################
################################################
NODES: 10.10.100.6, 10.10.100.6, 0.42
################################################
################################################
NODES: 10.10.100.6, 10.10.100.7, 3.15
################################################
################################################
NODES: 10.10.100.7, 10.10.100.5, 3.08
################################################
################################################
NODES: 10.10.100.7, 10.10.100.6, 3.20
################################################
################################################
NODES: 10.10.100.7, 10.10.100.7, 0.46
################################################
  10.10.100.5 10.10.100.6 10.10.100.7
10.10.100.5      0.41      2.53      3.12
10.10.100.6      2.76      0.42      3.15
10.10.100.7      3.08      3.20      0.46
[hpcuser@stuartkirk1975-hpc-compute3 ~]$
```

Switch fullpingpong to TCP and observe the results:

- `sed -i "s/dapl/tcp/g" full-pingpong.sh`
- `./fullpingpong.sh`

Observe the latency (measured in microseconds) between nodes for Intel MPI communications. The worker node IP addresses are listed on the horizontal and vertical axis; Their intersection indicates the latency between nodes. On some Microsoft Azure virtual machines to achieve ever faster Infiniband communications, the Infiniband interface is presented directly to the operating system and can be manipulated by contents in the ib_utils* RPMs

LAB 3 – AZURE BIG DATA SOLUTIONS USING HDINSIGHT

Summary of Lab

HDInsight in Microsoft's Platform-based Big Data solution; It is one of the most popular services among enterprise customers for open-source Apache Hadoop and Apache Spark analytics. HDInsight is a cloud distribution of the Apache Hadoop

components from Hortonworks Data Platform. During the cluster deploy, a storage account is created in your resource group where several sample data sets will be placed. Follow the tutorial listed below to begin performing Big Data queries against HDInsight.

Deploy HDInsight 4.0

Estimated Playbook Runtime: 22m 17s

- `ansible-playbook hdinsight-40-create-hdinsight.yml`

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook hdinsight-40-create-hdinsight.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Create HDInsight Cluster] ************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [localhost]

TASK [Create a Storage Account for HDInsight] **********************************************************************************
changed: [localhost]

TASK [Getting Storage Account Keys from Azure REST API] ************************************************************************
[WARNING]: Azure API profile latest does not define an entry for GenericRestClient
changed: [localhost]

TASK [Create Storage blob container for HDInsight data] ************************************************************************
changed: [localhost]

TASK [Create HDInsight Cluster] ************************************************************************************************
[WARNING]: Azure API profile latest does not define an entry for HDInsightManagementClient
changed: [localhost]

PLAY RECAP *********************************************************************************************************************
localhost                  : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


real    15m42.051s
user    1m21.122s
sys     0m5.266s
[student@master-vnc-desktop playbooks]$
```

Big Data Sample Exercise

Visit and complete the following tutorial:

- Visit https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-load-data-run-query

---

LAB 4 – AZURE KUBERNETES SERVICE (AKS)

Summary of Lab

Azure Kubernetes Service (AKS) is a managed container orchestration service based on the open source Kubernetes project. An organization can use the AKS service free-of-charge to deploy, scale and manage containers and container-based applications across a cluster of hosts of electable size. As part of this lab, you will deploy an AKS cluster and application using Ansible. Using the Azure Linux CLI, you will also merge the K8S configuration into your local ~/.kube directory thus enabling cluster control with standard "kubectl" directives.

Playbook 0 – Create the Managed AKS Cluster

Estimated Playbook Runtime: 7m 41s

This playbook will deploy an Azure Kubernetes Service using Ansible and will provide you with the name of the cluster upon completion. This service will host the "Service Tracker" application which is deployed in the playbooks to follow.

- **ansible-playbook aks-00-create-aks-cluster.yml**

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook aks-00-create-aks-cluster.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Deploy AKS Infrastructure to support K8S Service Tracker Application] ****************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [localhost]

TASK [Create a managed Azure Container Services (AKS) instance to support K8S Service Tracker Application] ********************
[WARNING]: Azure API profile latest does not define an entry for ContainerServiceClient
changed: [localhost]

TASK [debug] *******************************************************************************************************************
ok: [localhost] => {
    "msg": "The name of your AKS cluster is: stuartkirk1975-aks. Please execute: 'az aks get-credentials -n stuartkirk1975-aks -g 01'"
}

PLAY RECAP *********************************************************************************************************************
localhost                  : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


real    7m44.150s
user    0m40.258s
sys     0m2.512s
[student@master-vnc-desktop playbooks]$
```

Merge the cluster configuration

Remind yourself of your resource group and merge the K8S credentials into your ~/.kube directory or copy & paste the output from the playbook which was just run to allow kubectl to manage your AKS cluster

- **az aks get-credentials -g <YOUR_RG> -n <YOUR_AKS_CLUSTER_NAME>**

You will now be able to execute standard "kubectl" commands against the AKS cluster. Feel free to explore the cluster or continue with the execution of playbooks.

Playbook 1 – Create an Azure Container Registry

Estimated Playbook Runtime: 0m 12s

- **ansible-playbook aks-01-create-acr.yml**

Azure has a container repository that can manage, build, and securely store your container workloads. This playbook will create a new Azure Container Registry and provide the credentials for it. While you may not need the credentials as part of this lab, as the Ansible playbooks accommodate for it, there may be other labs in which you will. Please take note of them.

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook aks-01-create-acr.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Create Azure Container Registry] *****************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [localhost]

TASK [Create an Azure Container Registry for AKS and Modernization Labs] *******************************************
[WARNING]: Azure API profile latest does not define an entry for ContainerRegistryManagementClient
changed: [localhost]

TASK [Query Azure Container Registry for credential information] ***************************************************
ok: [localhost]

TASK [debug] *******************************************************************************************************
ok: [localhost] => {
    "msg": "ACR Username: stuartkirk197547428acr"
}

TASK [debug] *******************************************************************************************************
ok: [localhost] => {
    "msg": "ACR Password: QP+nG0BKnB7b3hnkdh0nf98gjTBkHoCC"
}

TASK [debug] *******************************************************************************************************
ok: [localhost] => {
    "msg": "ACR Hostname: stuartkirk197547428acr.azurecr.io"
}

PLAY RECAP *********************************************************************************************************
localhost                  : ok=6    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


real    0m9.140s
user    0m5.264s
sys     0m0.845s
[student@master-vnc-desktop playbooks]$
```

Playbook 2 – Build / Tag / Push containers to ACR using Podman

Estimated Playbook Runtime: 2m 40s

- **ansible-playbook aks-02-build-tag-push.yml**

The previous version of this instructor-led-lab made use of Docker to build and manage containers. Docker has been removed in favor of Buildah and Podman to manage the container workloads you will be deploying. Using the "podman_image" module for Ansible, this playbook will build all the containers required to form this application and its associated microservices. It will subsequently push them to your Azure Container Registry.

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook aks-02-build-tag-push.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not mat

PLAY [Use Podman to Build and Tag Service Tracker Containers] ******************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [localhost]

TASK [Query Azure Container Registry for credential information] ****************************************************
[WARNING]: Azure API profile latest does not define an entry for ContainerRegistryManagementClient
ok: [localhost]

TASK [Build Service Tracker DATA API container with Podman] ********************************************************
ok: [localhost]

TASK [Build Service Tracker FLIGHTS API container with Podman] *****************************************************
changed: [localhost]

TASK [Build Service Tracker QUAKES API container with Podman] ******************************************************
changed: [localhost]

TASK [Build Service Tracker WEATHER API container with Podman] *****************************************************
changed: [localhost]

TASK [Build Service Tracker User Interface container with Podman] **************************************************
changed: [localhost]

TASK [Push DATA API container to Azure Container Registry] *********************************************************
changed: [localhost]

TASK [Push FLIGHTS API container to Azure Container Registry] ******************************************************
changed: [localhost]

TASK [Push QUAKES API container to Azure Container Registry] *******************************************************
changed: [localhost]

TASK [Push WEATHER API container to Azure Container Registry] ******************************************************
changed: [localhost]

TASK [Push Service Tracker User Interface container to Azure Container Registry] ***********************************
changed: [localhost]

PLAY RECAP *********************************************************************************************************
localhost                  : ok=12   changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


real    2m31.318s
user    2m10.196s
sys     0m36.073s
[student@master-vnc-desktop playbooks]$
```

Playbook 3 – Deploy the Kubernetes configuration files

Estimated Playbook Runtime: 0m 50s

- **`ansible-playbook aks-03-deploy-k8s.yml`**

This playbook will perform two tasks:

- Change the k8s configuration files to reference your Azure Container Registry
- Apply the k8s configuration files to the AKS cluster

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook aks-03-deploy-k8s.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Deploy Service Tracker Containers into Azure Kubernetes Service] *********************************************************************

TASK [Gathering Facts] *********************************************************************************************************************
ok: [localhost]

TASK [Query Azure Container Registry for credential information] ***************************************************************************
[WARNING]: Azure API profile latest does not define an entry for ContainerRegistryManagementClient
ok: [localhost]

TASK [Merging Azure Container Registry hostname into DATA API k8s configuration] ***********************************************************
changed: [localhost]

TASK [Merging Azure Container Registry hostname into FLIGHTS API k8s configuration] ********************************************************
changed: [localhost]

TASK [Merging Azure Container Registry hostname into QUAKES API k8s configuration] *********************************************************
changed: [localhost]

TASK [Merging Azure Container Registry hostname into WEATHER API k8s configuration] ********************************************************
changed: [localhost]

TASK [Merging Azure Container Registry hostname into Service Tracker UI k8s configuration] *************************************************
changed: [localhost]

TASK [Create a k8s namespace in AKS for Service Tracker] ***********************************************************************************
ok: [localhost]

TASK [Apply MongoDB configuration to AKS and wait for it to come alive] ********************************************************************
changed: [localhost]

TASK [Apply DATA API configuration to AKS] *************************************************************************************************
changed: [localhost]

TASK [Apply FLIGHTS API configuration to AKS] **********************************************************************************************
changed: [localhost]

TASK [Apply QUAKES API configuration to AKS] ***********************************************************************************************
changed: [localhost]

TASK [Apply WEATHER API configuration to AKS] **********************************************************************************************
changed: [localhost]

TASK [Apply Service Tracker User Interface configuration to AKS] ***************************************************************************
changed: [localhost]

TASK [debug] *******************************************************************************************************************************
ok: [localhost] => {
    "msg": "Don't forget to run 'kubectl get svc' until you see the IP address of the cluster load balancer. Then connect to it on port 8080 via http."
}

PLAY RECAP *********************************************************************************************************************************
localhost                  : ok=15   changed=11   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


real    0m42.304s
user    0m11.422s
sys     0m2.292s
[student@master-vnc-desktop playbooks]$
```

After the playbook has completed, you will need to wait until the cluster LoadBalancer IP is provisioned for the application As shown below. You can determine the IP using "kubectl" however using the "watch" command is more efficient as you will be alerted after it is provisioned – it may take 5-6 minutes.

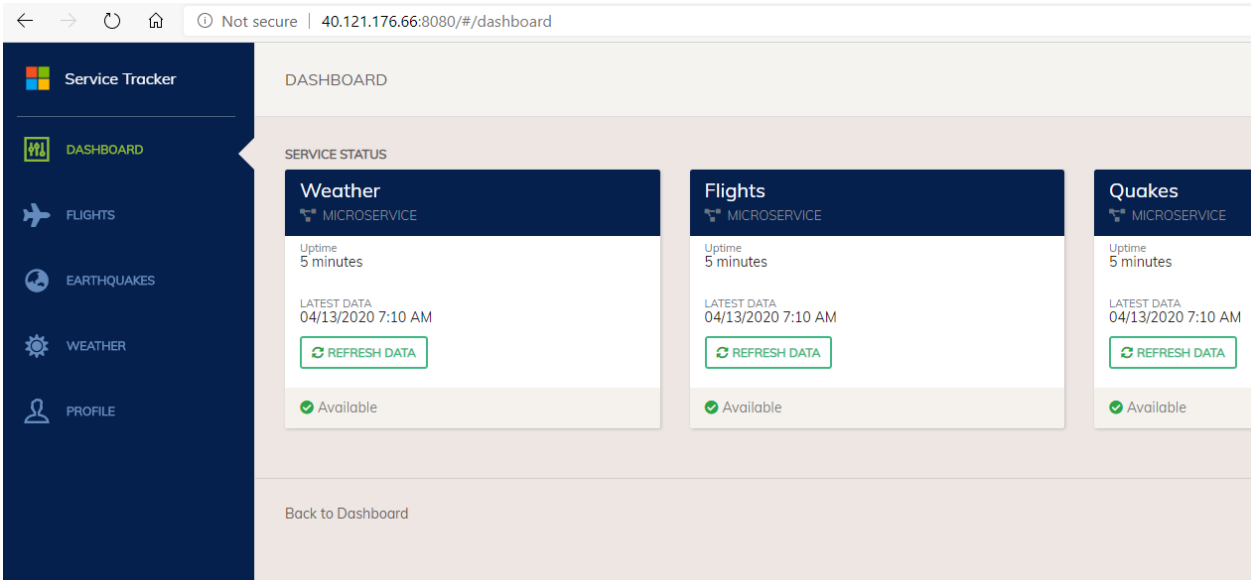
- **watch kubectl get service**

```
Every 2.0s: kubectl get svc

NAME                TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)          AGE
data-api            ClusterIP      10.0.192.251    <none>          3009/TCP         2m23s
flights-api         ClusterIP      10.0.142.161    <none>          3003/TCP         2m22s
kubernetes          ClusterIP      10.0.0.1        <none>          443/TCP          5h25m
mongodb             ClusterIP      10.0.177.160    <none>          27017/TCP        2m53s
quakes-api          ClusterIP      10.0.252.45     <none>          3012/TCP         2m21s
service-tracker-ui  LoadBalancer   10.0.108.191    40.121.176.66   8080:32120/TCP   2m19s
weather-api         ClusterIP      10.0.5.221      <none>          3015/TCP         2m20s
```

Test the Service Tracker Application in the Managed K8S Cluster

Open your web browser and visit the IP address on port 8080 which was provided to you by "`kubectl get svc`" using http. If there is no connection, please wait and try again. The services can take several minutes to become live. Feel free to explore the different micro services offered and the profile in which you are connected as.

- Visit http://x.x.x.x:8080 (in the example http://40.121.176.66:8080)
- Be sure to click the buttons to "REFRESH DATA" to make each microservice application obtain its initial dataset and/or refresh the page.



LAB 5 – SERVERLESS COMPUTING USING AZURE FUNCTIONS

Summary of Lab

In this exercise you will run Ansible as an Azure Function Application; We will create an entire end-to-end scenario. Initially, we will start by creating an Azure Container Registry. Using the Azure REST API, we will then create task in ACR that will build our Function Application image. Leveraging new functionality in Ansible 2.8 we will then deploy a container-based Azure Function Application. Finally, to test the Function Application we will deploy a static website using a v2 Azure Storage Account.

Pre-Requisites

This Lab Exercise will assume that:

- You have added your GitHub ID and Personal Access Token to the vars.yml file prior to executing this lab

- That the contents of vars-myvars.yml displays both your GitHub ID and Personal Access Token
- You have forked the master branch of this lab repository located at: https://github.com/stuartatmicrosoft/RedHatSummit2020

Playbook 0 – Create Azure Container Registry & Image

Estimated Playbook Runtime: 1m 31s

- `ansible-playbook fa-00-create-image.yml`

ACR Creation Task:

```
    - name: Create container registry
      azure_rm_containerregistry:
        resource_group: "{{ resource_group }}"
        name: "{{ registry_name }}"
        location: eastus
        admin_user_enabled: true
        sku: Premium
```

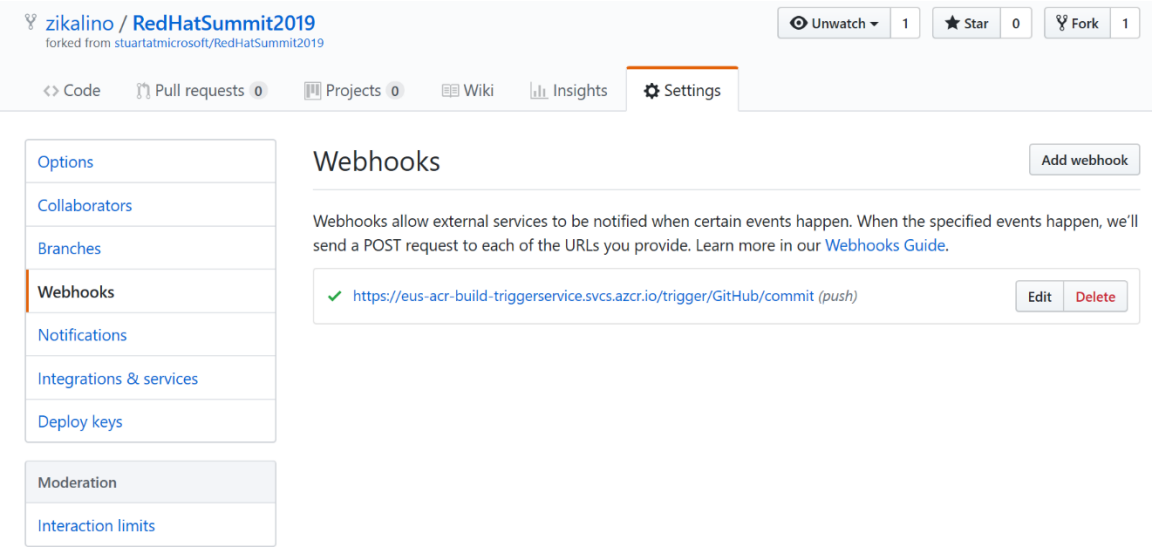
Image Creation Task:

In this task we will be using the azure_rm_resource module. This module allows Ansible playbooks to call the Azure REST API directly in the event there is not a pre-existing module for the task you wish to complete. This will provide you with virtually complete access to the Azure platform.

```
    - name: Build Image using Azure Container Registry
      azure_rm_resource:
        api_version: '2018-09-01'
        resource_group: "{{ resource_group }}"
        provider: containerregistry
        resource_type: registries
        resource_name: "{{ registry_name }}"
        subresource:
          - type: tasks
            name: "{{ task_name }}"
        body:
          properties:
            status: Enabled
            platform:
              os: Linux
              architecture: amd64
            agentConfiguration:
              cpu: 2
            step:
              type: Docker
              imageNames:
                - functionapp
              dockerFilePath: Dockerfile
              contextPath: function-app-container
              isPushEnabled: true
              noCache: false
```

```
            trigger:
              sourceTriggers:
                - name: mySourceTrigger
                  sourceRepository:
                    sourceControlType: Github
                    repositoryUrl: https://github.com/{{ github_id
}}/RedHatSummit2020
                    branch: master
                    sourceControlAuthProperties:
                      tokenType: PAT
                      token:  "{{ github_token }}"
                  sourceTriggerEvents:
                    - commit
                  status: Enabled
              baseImageTrigger:
                name: myBaseImageTrigger
                baseImageTriggerType: Runtime
          location: eastus
```

After this playbook completes, verify in your GitHub account that the webhook was created. You can view this by viewing the forked repository in your GitHub account and looking in the "Settings/Webhooks" screen.

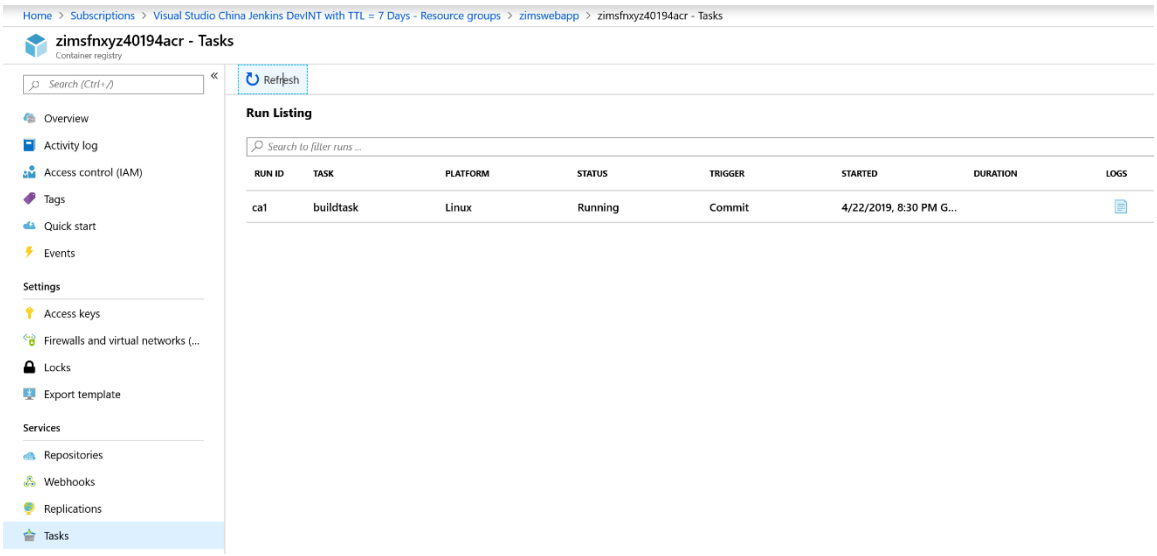


To verify that the webhook is operational, click the "Edit" button to verify that a delivery has been made:

This initial webhook delivery is not related to any recent commit, so to trigger your image build you will need to commit a change to the forked repository. To accomplish this, edit README.md file, add any new content you wish and commit

the change to your forked repository. If you re-check the "Recent Deliveries" list, it should now contain two items.

In addition, if you visit the Azure Portal (https://portal.azure.com) and find your ACR in your resource group, select "Tasks" in the blade on the left, you should see following task created:



Before proceeding to the next playbook, wait until the Azure Portal displays the status as "Succeeded". This means that your image has been updated and is ready.

Playbook 1 – Create an Azure Function application

Estimated Playbook Runtime: 1m 48s

- `ansible-playbook fa-01-create-function-app-from-acr.yml`

This playbook performs the following tasks:

Create a Linux-based Azure Application Service Plan

```
    - name: Create a linux app service plan
      azure_rm_appserviceplan:
        resource_group: "{{ resource_group }}"
        name: "{{ function_name }}plan"
        sku: S1
        is_linux: true
        number_of_workers: 1
```

Create a Storage Account

```
    - name: create storage account for function apps
      azure_rm_storageaccount:
        resource_group: '{{ resource_group }}'
        name: "{{ storage_name }}"
        account_type: Standard_LRS
        kind: StorageV2
```

```
      register: output
```

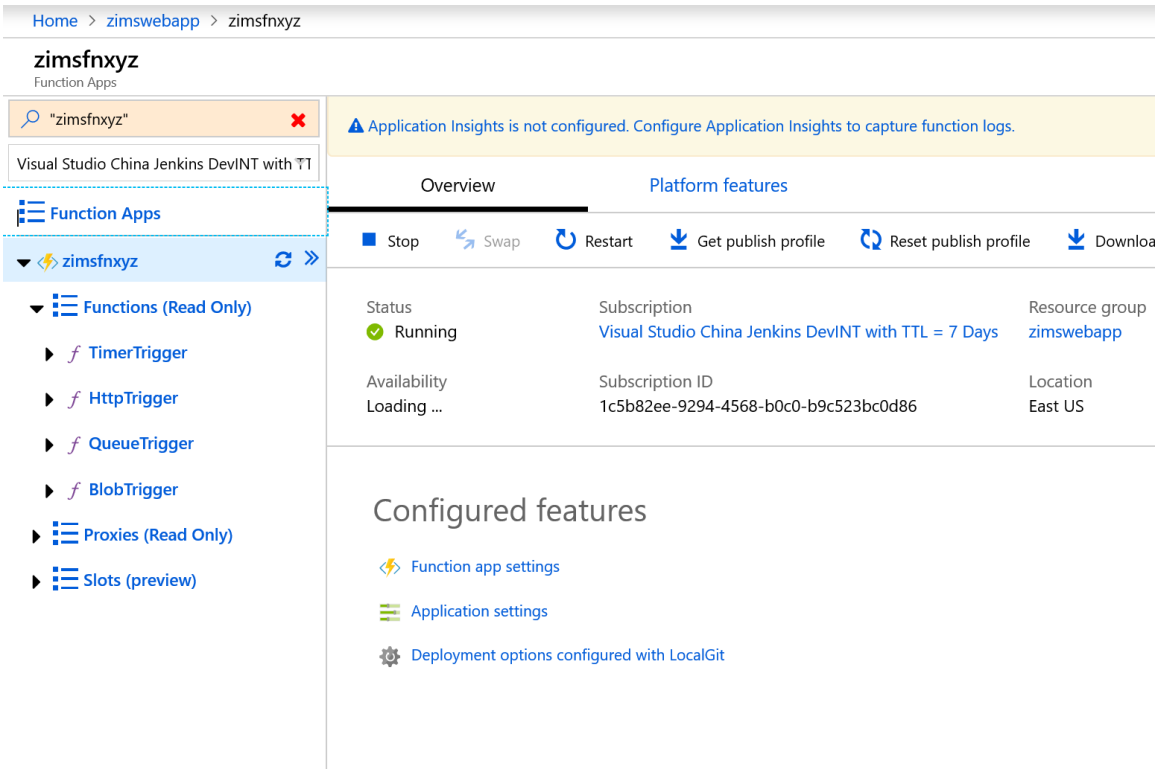
Obtain Azure Container Registry Credentials:

```
    - name: Obtain Azure Container Registry Facts
      azure_rm_containerregistry_facts:
        resource_group: "{{ resource_group }}"
        name: "{{ registry_name }}"
        retrieve_credentials: true
      register: acr_output
```

Create the Azure Function Application

```
    - name: Create container based function app
      azure_rm_functionapp:
        resource_group: "{{ resource_group }}"
        name: "{{ function_name }}"
        storage_account: "{{ storage_name }}"
        plan:
          resource_group: "{{ resource_group }}"
          name: "{{ function_name }}plan"
        container_settings:
          name: functionapp
          registry_server_url: "{{ acr_output.registries[0].login_server
}}"
          registry_server_user: "{{ acr_output.registries[0].name }}"
          registry_server_password: "{{
acr_output.registries[0].credentials.password }}"
```

Once again, visit the Azure Portal (https://portal.azure.com) and in your resource group you should see a new resource named "Application Service". If the Function Application was created correctly, you should see the following output after the blade loads:



The following contents should be visible in the screen above:

- TimerTrigger
- HttpTrigger
- QueueTrigger
- BlobTrigger

If these functions do not appear after several seconds your function application was not created correctly. Please notify one of the proctors.

Playbook 2 – Create a static web app using an Azure Function

Estimated Playbook Runtime: 0m 25s

- `ansible-playbook fa-02-create-website.yml`

This playbook performs the following tasks:

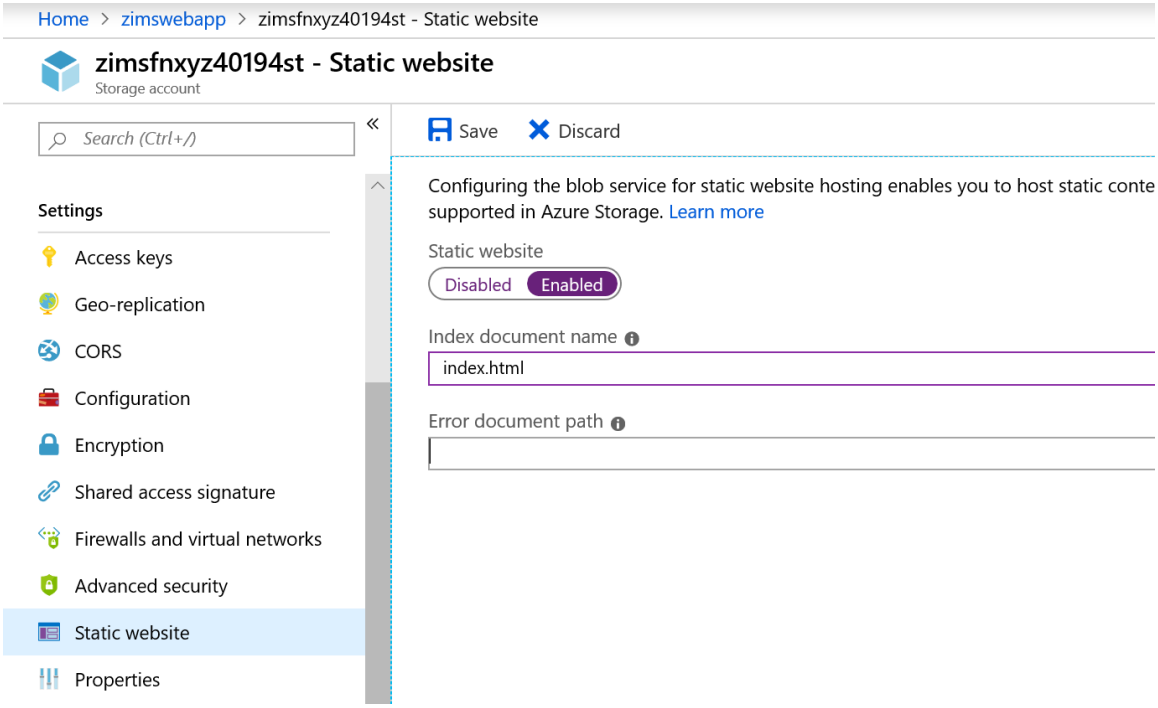
Create a new index.html / Populate / Upload to Storage Account

```
    - name: Create index.html file from template
      copy:
        src: ../function-app-container/index-template.html
        dest: ../function-app-container/index.html
    - name: Adjust function app URL
      replace:
        path: ../function-app-container/index.html
        regexp: FUNCTIONNAME
        replace: "{{ function_name }}"
    - name: Create container $web and upload index.html
      azure_rm_storageblob:
```

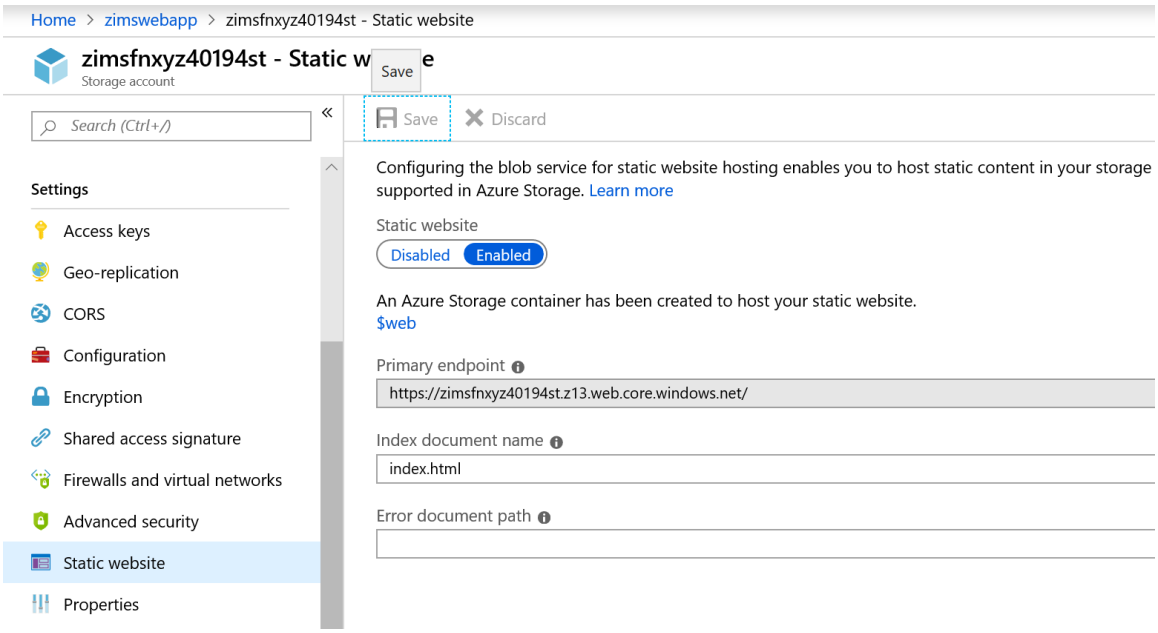
```
        resource_group: "{{ resource_group }}"
        storage_account_name: "{{ storage_name }}"
        container: $web
        blob: index.html
        src: ../function-app-container/index.html
        public_access: container
        content_type: 'text/html'
        force: yes
```

To enable the website, once again visit the Azure Portal (https://portal.azure.com) and select the Storage Account that was created. On the blade that appears:

- Click "Settings"
- Click "Static Website"
- Choose "Enabled"
- Enter "index.html" as the Index Document Name
- Click "Save"



After the data is saved, you can use the primary endpoint now visible on the blade and visit this site using your web browser:



In your web browser, you should see the following very simple webpage created:

Here you can enter any playbook and your Azure credentials contained in "credentials.txt".

After pressing "Run Playbook" the data will be sent to and executed by the function application. The results should be displayed after processing completes.

LAB 6 – MODERNIZE NODEJS & MONGODB WITH WEB APPS & COSMOSDB

Summary of Lab

When embracing cloud, it is important to consider not only the migration of applications but also the opportunity to modernize them. This lab will allow you to take an existing NodeJS application currently running on your Lab VM (Try visiting http://localhost) which is backed by a locally running MongoDB and migrate it to Azure. Instead of a straight "Lift & Shift" migration, we will containerize the NodeJS application using podman and push it to Azure Container Registry. You will then create an Azure Web Application to launch the application as a platform service. We will then deploy an Azure CosmosDB Database (or use the existing one from Lab 4) which will provide the database back-end for the existing MongoDB. As part of this lab, all existing data will be migrated from MongoDB to Azure CosmosDB and you will be able to test the resulting application which will be deployed entirely as a platform-based service using Ansible.

Playbook 0 – Create an Azure Container Registry (ACR)

Estimated Playbook Runtime: 0m 9s

Create an Azure Container Registry to store the containerized NodeJS application. Upon completion of the playbook, take note of the ACR Username, ACR Password and ACR hostname. They will be required to push the container to ACR. Your ACR may have already been created from the AKS lab if you chose to complete it.

- `ansible-playbook todo-00-create-acr.yml`

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook todo-00-create-acr.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not mat

PLAY [Create Azure Container Registry] *****************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [localhost]

TASK [Create an Azure Container Registry for AKS and Modernization Labs] *******************************
[WARNING]: Azure API profile latest does not define an entry for ContainerRegistryManagementClient
changed: [localhost]

TASK [Query Azure Container Registry for credential information] ***************************************
ok: [localhost]

TASK [debug] *******************************************************************************************
ok: [localhost] => {
    "msg": "ACR Username: stuartkirk197547428acr"
}

TASK [debug] *******************************************************************************************
ok: [localhost] => {
    "msg": "ACR Password: QP+nG0BKnB7b3hnkdh0nf98gjTBkHoCC"
}

TASK [debug] *******************************************************************************************
ok: [localhost] => {
    "msg": "ACR Hostname: stuartkirk197547428acr.azurecr.io"
}

PLAY RECAP *********************************************************************************************
localhost                  : ok=6    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


real    0m7.743s
user    0m5.276s
sys     0m0.844s
[student@master-vnc-desktop playbooks]$
```

Playbook 1 – Deploy an Azure PaaS NoSQL CosmosDB

Estimated Playbook Runtime: 4m 41s

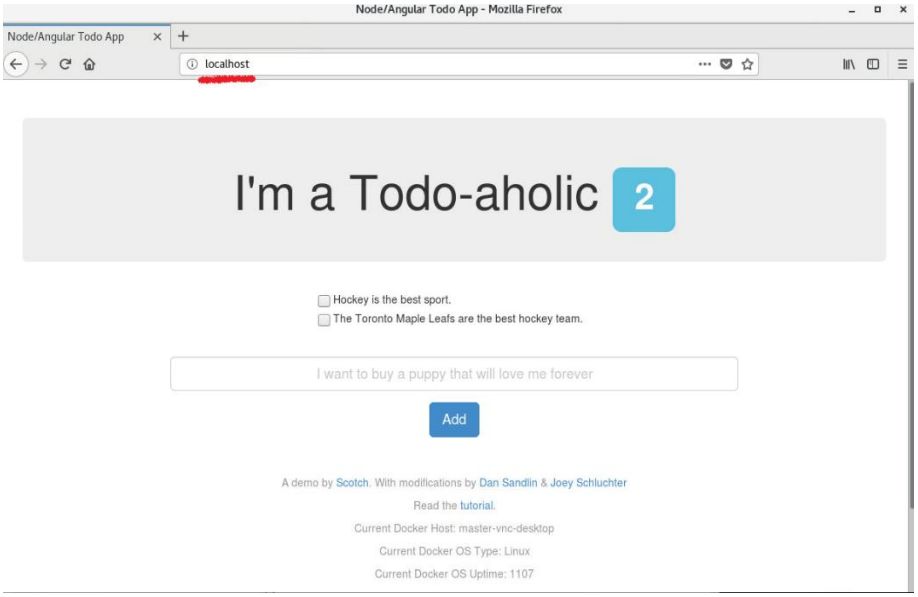
Create an Azure CosmosDB database to store the data from the containerized NodeJS application. Upon completion of the playbook, take note of the CosmosDB Username, CosmosDB Connection String and CosmosDB Primary Master Key.

- **ansible-playbook todo-01-create-cosmosdb.yml**

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook todo-01-create-cosmosdb.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Create CosmosDB to modernize TODO Application] ***************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [localhost]

TASK [Create Cosmos DB (MongoDB) for Red Hat Summit Labs] **********************************************************
[WARNING]: Azure API profile latest does not define an entry for CosmosDB
changed: [localhost]

TASK [Obtain CosmosDB Facts] ***************************************************************************************
ok: [localhost]

TASK [Getting CosmosDB Username] ***********************************************************************************
ok: [localhost]

TASK [Getting CosmosDB Connection String] **************************************************************************
ok: [localhost]

TASK [Getting CosmosDB Primary Master Key] *************************************************************************
ok: [localhost]

TASK [debug] *******************************************************************************************************
ok: [localhost] => {
    "msg": "CosmosDB Username: stuartkirk1975-cosmosdb"
}

TASK [debug] *******************************************************************************************************
ok: [localhost] => {
    "msg": "CosmosDB Connection String: mongodb://stuartkirk1975-cosmosdb:vLif0QYL895GCoFIejRlrm3sD4DcuB3cQPg5qNVKg7MiWZ5oUwBMP5VX5Z3S
Eq7H4ztXb8zB2zjV7yVgRtQM8g==@stuartkirk1975-cosmosdb.documents.azure.com:10255/?ssl=true&replicaSet=globaldb"
}

TASK [debug] *******************************************************************************************************
ok: [localhost] => {
    "msg": "CosmosDB Primary Master Key: vLif0QYL895GCoFIejRlrm3sD4DcuB3cQPg5qNVKg7MiWZ5oUwBMP5VX5Z3SEq7H4ztXb8zB2zjV7yVgRtQM8g=="
}

PLAY RECAP *********************************************************************************************************
localhost                  : ok=9    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


real    3m44.959s
user    0m29.538s
sys     0m2.634s
[student@master-vnc-desktop playbooks]$
```

Populate data in the "To-Do" application

- Visit http://localhost and wait for the NodeJS app to appear
- Enter a few statements and select "Add" after each one
- You may enter as many statements as you wish



Prepare the Container using podman

- Switch into the **/source/sample-apps/nodejs-todo/src** directory
- Edit (vi/nano) the **Dockerfile** to expose **port 80** instead of **port 8080**.
- Save the file.

Build the container with **podman**:

  - `sudo podman build --format docker -t ossdemo/nodejs-todo .`

```
[student@master-vnc-desktop src]$ sudo podman build --format docker -t ossdemo/nodejs-todo .
STEP 1: FROM node:boron
Getting image source signatures
Copying blob 69df12c70287 done
Copying blob 3b7ca19181b2 done
Copying blob 425d7b2a5bcc done
Copying blob 221d80d00ae9 done
Copying blob c5e155d5a1d1 done
Copying blob 4250b3117dca done
Copying blob ea2f5386a42d done
Copying blob d421d2b3c5eb done
Copying config ab290b8530 done
Writing manifest to image destination
Storing signatures
STEP 2: RUN mkdir -p /src/app
0302ef564188d3c6e527c1702dc2c8d86a6187a07ca1d681d5b5032de88e2d2c
STEP 3: WORKDIR /src/app
9d5523ba10fe76dfccb9373d9f0d736cae680ffc6905161433d9cbf55c0363f2
STEP 4: COPY package.json /src/app/
e8070b3022c1108ab040ab003fe34be4d50a0eb251231846b58f1688596bad63
STEP 5: RUN npm install
node-todo-oss@0.0.7 /src/app
+-- applicationinsights@0.19.0
| `-- zone.js@0.7.6
+-- body-parser@1.19.0
| +-- bytes@3.1.0
| +-- content-type@1.0.4
| +-- debug@2.6.9
```

Tag the container using the hostname of your ACR provided by the playbook execution:

- **sudo podman tag ossdemo/nodejs-todo Xacr.azurecr.io/ossdemo/nodejs-todo**

Push the Container to Azure Container Registry using podman

Login to your ACR and push the container using the username and password of your ACR provided by the first playbook you ran in this lab (scroll up):

- **sudo podman login X.azurecr.io -u USERNAME -p PASSWORD**
- **sudo podman push X.azurecr.io/ossdemo/nodejs-todo**

```
[student@master-vnc-desktop src]$ sudo podman tag ossdemo/nodejs-todo stuartkirk197527382acr.azurecr.io/ossdemo/nodejs-todo
[student@master-vnc-desktop src]$ sudo podman login stuartkirk197527382acr.azurecr.io -u stuartkirk197527382acr -p =kAzUYGI2iXhMgJUZPD
chejQpoj=qiTe
Login Succeeded!
[student@master-vnc-desktop src]$ sudo podman push stuartkirk197527382acr.azurecr.io/ossdemo/nodejs-todo
Getting image source signatures
Copying blob a27518e43e49 done
Copying blob ec62f19bb3aa done
Copying blob 4230ff7f2288 [==============>-----------------------] 54.4MiB / 139.6MiB
Copying blob f94641f1fe1f [===================>------------------] 53.6MiB / 100.7MiB
Copying blob 2c719774c1e1 done
Copying blob 910d7fd9e23e [==>---------------------------------] 44.4MiB / 549.7MiB
Copying blob f1965d3c206f [============================>-------] 35.2MiB / 43.8MiB
Copying blob f39151891503 done
Copying blob b82f7790d5fd [======================================] 4.5KiB / 4.5KiB
```

Migrate MongoDB to Azure CosmosDB

Export your MongoDB data into a JSON file and import it into Azure CosmosDB using the CosmosDB Username, Hostname (in red) and Primary Master Key:

- `mongoexport --db nodejs-todo --collection todos --out todos.json`
- `mongoimport -h X.azure.com:10255 -u USERNAME -p PRIMARYMASTERKEY --ssl --sslAllowInvalidCertificates -d admin -c todos --file=todos.json`

```
TASK [Create Cosmos DB (MongoDB) for Red Hat Summit Labs] ***********************************************************************
[WARNING]: Azure API profile latest does not define an entry for CosmosDB
ok: [localhost]

TASK [Obtain CosmosDB Facts] ****************************************************************************************************
ok: [localhost]

TASK [Getting CosmosDB Username] ************************************************************************************************
ok: [localhost]

TASK [Getting CosmosDB Connection String] ***************************************************************************************
ok: [localhost]

TASK [Getting CosmosDB Primary Master Key] **************************************************************************************
ok: [localhost]

TASK [debug] ********************************************************************************************************************
ok: [localhost] => {
    "msg": "CosmosDB Username: stuartkirk1975-cosmosdb"
}

TASK [debug] ********************************************************************************************************************
ok: [localhost] => {
    "msg": "CosmosDB Connection String: mongodb://stuartkirk1975-cosmosdb:vLif0QYL895GCoFIejRlrm3sD4DcuB3cQPg5qNVKg7MiWZ5oUwBMP5VX5Z3S
Eq7H4ztXb8zB2zjV7yVgRtQM8g==@stuartkirk1975-cosmosdb.documents.azure.com:10255/?ssl=true&replicaSet=globaldb"
}

TASK [debug] ********************************************************************************************************************
ok: [localhost] => {
    "msg": "CosmosDB Primary Master Key: vLif0QYL895GCoFIejRlrm3sD4DcuB3cQPg5qNVKg7MiWZ5oUwBMP5VX5Z3SEq7H4ztXb8zB2zjV7yVgRtQM8g=="
}

PLAY RECAP **********************************************************************************************************************
localhost                  : ok=9    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


real    0m10.270s
user    0m6.961s
sys     0m1.041s
[student@master-vnc-desktop playbooks]$ cd /source/sample-apps/nodejs-todo/src/
[student@master-vnc-desktop src]$ mongoimport -h stuartkirk1975-cosmosdb.documents.azure.com:10255 -u stuartkirk1975-cosmosdb -p vLif0
QYL895GCoFIejRlrm3sD4DcuB3cQPg5qNVKg7MiWZ5oUwBMP5VX5Z3SEq7H4ztXb8zB2zjV7yVgRtQM8g== --ssl --sslAllowInvalidCertificates -d admin -c to
dos --file=todos.json
2020-04-13T18:55:56.762-0400    connected to: mongodb://stuartkirk1975-cosmosdb.documents.azure.com:10255/
2020-04-13T18:55:56.768-0400    0 document(s) imported successfully. 0 document(s) failed to import.
[student@master-vnc-desktop src]$
```

Playbook 2 – Create an Azure Application Service Plan

Estimated Playbook Runtime: 0m 14s

- `ansible-playbook todo-02-create-appservice-plan.yml`

Playbook 3 – Create an Azure Web Application

Estimated Playbook Runtime: 0m 31s

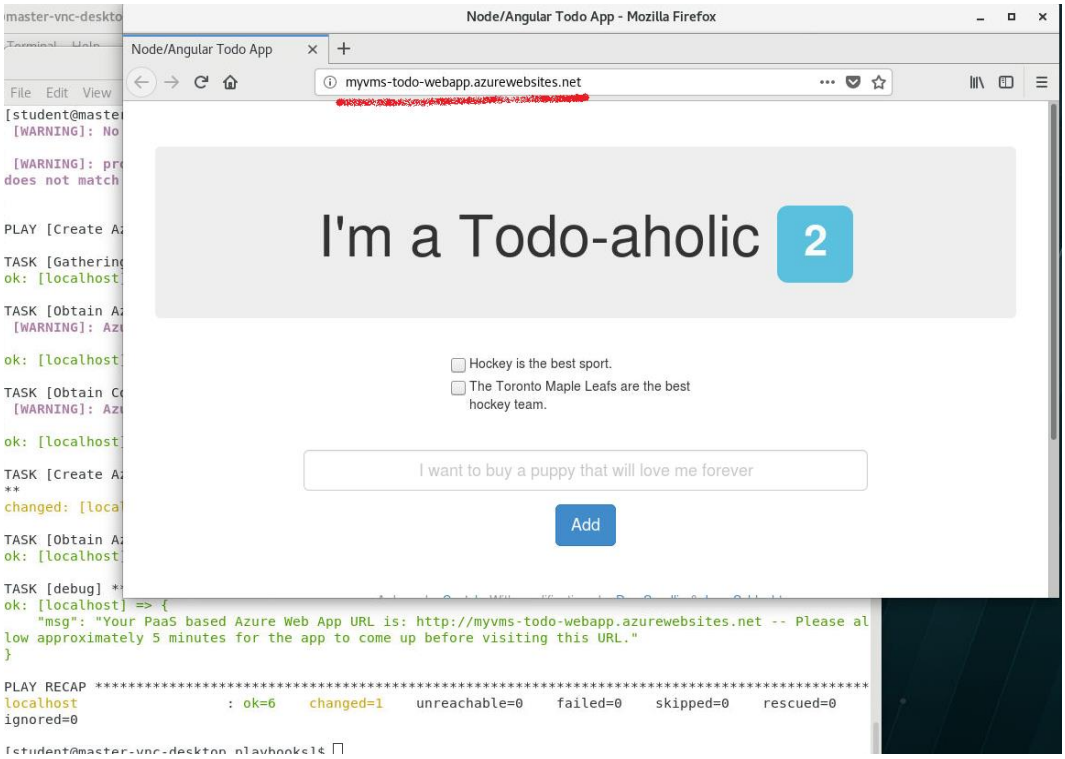
- `ansible-playbook todo-03-create-azure-webapp.yml`

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook todo-03-create-azure-webapp.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Create Azure Webapp for TODO Application] ********************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [localhost]

TASK [Obtain Azure Container Registry Facts] ***********************************************************************************
[WARNING]: Azure API profile latest does not define an entry for ContainerRegistryManagementClient
ok: [localhost]

TASK [Obtain Cosmos DB Facts] **************************************************************************************************
[WARNING]: Azure API profile latest does not define an entry for CosmosDB
ok: [localhost]

TASK [Create Azure Web App for TODO Application utilizing container pushed to Azure Container Registry] ************************
changed: [localhost]

TASK [Obtain Azure Web App Facts] **********************************************************************************************
ok: [localhost]

TASK [debug] *******************************************************************************************************************
ok: [localhost] => {
    "msg": "Your PaaS based Azure Web App URL is: http://stuartkirk1975-todo-webapp.azurewebsites.net -- Please allow approximately 5
minutes for the app to come up before visiting this URL."
}

PLAY RECAP *********************************************************************************************************************
localhost                  : ok=6    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


real    1m14.716s
user    0m14.438s
sys     0m1.763s
[student@master-vnc-desktop playbooks]$
```

Test the Migrated Application

Now that the application has been created you will be accessing the NodeJS application as a containerized Azure Web Application connected to an Azure CosmosDB hosting the MongoDB application. You have, in effect, modernized an infrastructure service into a platform service. Congratulations!

- Visit http://<X>-todo-webapp.azurewebsites.net



LAB 7 – AZURE RED HAT OPENSHIFT

Summary of Lab

This lab provides exposure to Azure Red Hat OpenShift and allows you to interact with it via Ansible, the command line and the Web UI. Labs include logging into Azure Red Hat OpenShift and creating namespaces using Ansible, creating an application using source-to-image, and deployment of Microsoft SQL Server 2017 to OpenShift using Ansible. While the playbooks and applications are being deployed, you can choose to log into the Azure Red Hat OpenShift console and view the logs & builds as they progress or view from the command line with the "oc logs" command. It is recommended that you read this lab in its entirety prior to executing as some steps require quickly switching from the terminal to the Web UI.

Playbook 0 – Login to Azure Red Hat OpenShift / Create Namespaces

Estimated Playbook Runtime: 0m 12s

Login to Azure Red Hat OpenShift and create the projects (namespaces) required for the two deployments in this lab exercise.

- `ansible-playbook aro-00-login-namespace.yml`

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook aro-00-login-namespace.yml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the
implicit localhost does not match 'all'

PLAY [Log in to Azure Red Hat OpenShift (ARO) and create projects] ***********************

TASK [Gathering Facts] *********************************************************************
ok: [localhost]

TASK [Log in to Azure Red Hat OpenShift (ARO)] *********************************************
ok: [localhost]

TASK [Save API key variable] ***************************************************************
ok: [localhost]

TASK [Create an Azure Red Hat Openshift (ARO) project for Source-To-Image Deploy] ********
changed: [localhost]

TASK [Create an Azure Red Hat Openshift (ARO) project for Microsoft SQL Server] **********
changed: [localhost]

PLAY RECAP *********************************************************************************
localhost                  : ok=5    changed=2    unreachable=0    failed=0    skipped=0
  rescued=0    ignored=0


real    0m10.491s
user    0m6.074s
sys     0m1.003s
[student@master-vnc-desktop playbooks]$
```

Create a new Azure Red Hat OpenShift application using S2I

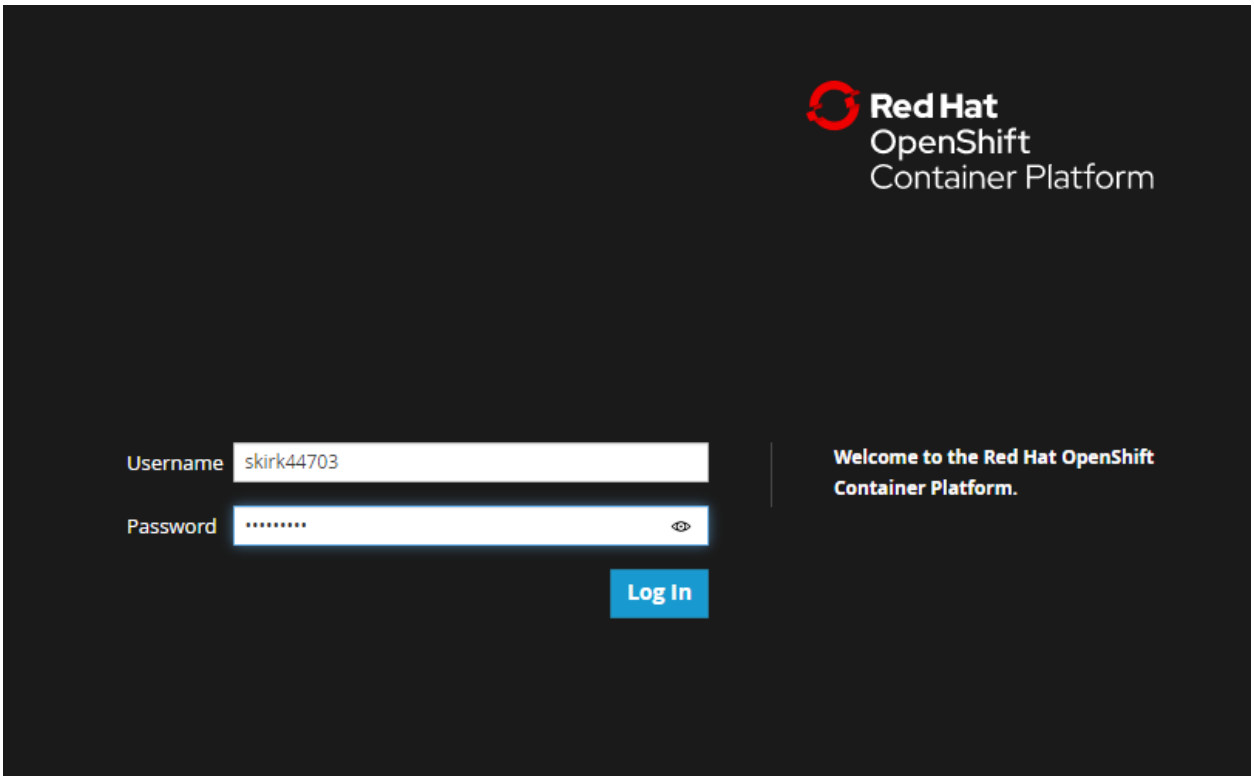
Source-to-Image (S2I) provides an alternative to using Dockerfiles to create new container images and can be used either as a feature from OpenShift or as the standalone s2i utility. S2i allows developers to work using their usual tools, instead of learning Dockerfile syntax and using operating system commands such as yum, and usually creates slimmer images, with fewer layers. S2I uses the following process to build a custom container image for an application:

1. Start a container from a base container image called the builder image, which includes a programming language runtime and essential development tools such as compilers and package managers.
2. Fetch the application source code, usually from a Git server, and send it to the container.
3. Build the application binary files inside the container.
4. Save the container, after some clean up, as a new container image, which includes the programming language runtime and the application binaries.

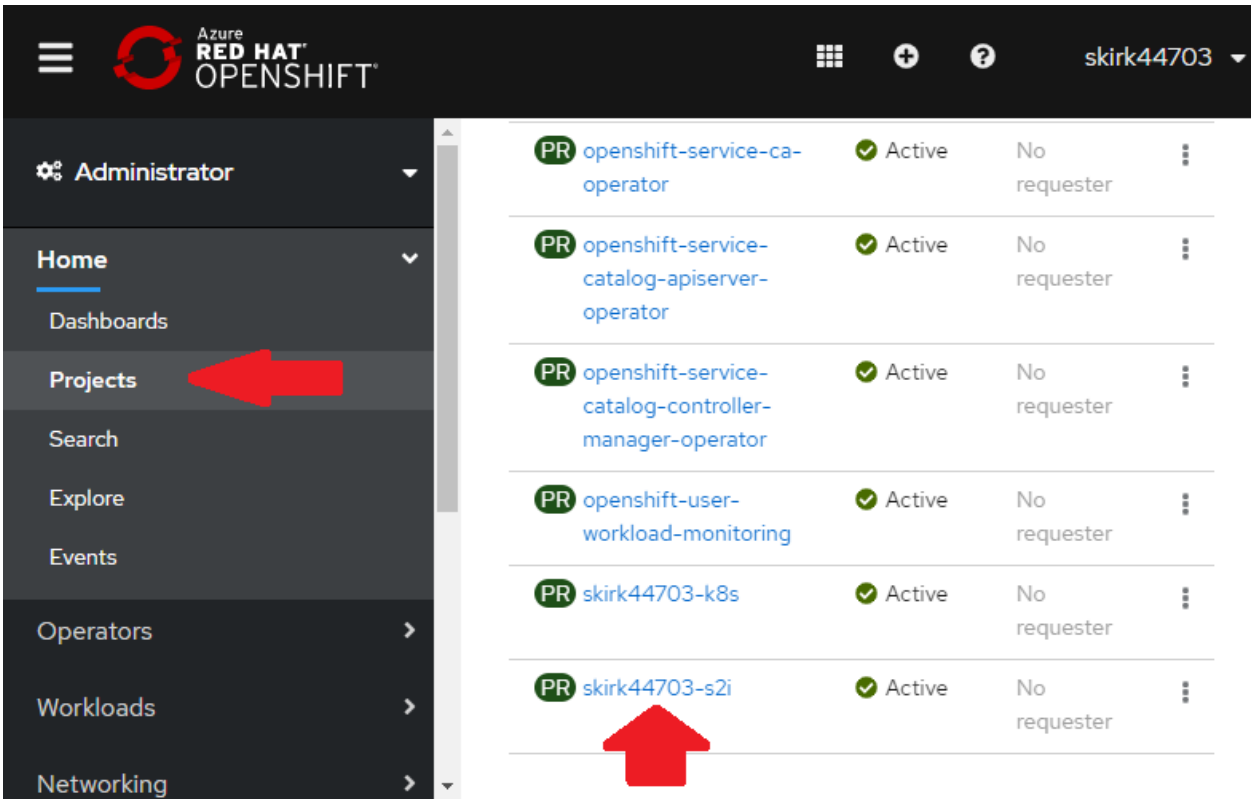
The builder image is a regular container image following a standard directory structure and providing scripts that are called during the S2I process. Most of these builder images can also be used as base images for Dockerfiles, outside the S2I process. The s2i command is used to run the S2I process outside of OpenShift, in a

Docker-only environment. It can be installed on a RHEL system from the source-to-image RPM package, and on other platforms, including Windows and Mac OS, from the installers available in the S2I project on GitHub.

Using a web browser, log in to the Azure Red Hat OpenShift console using the URL which has been provided for you and the login name which was assigned to you out of the lab-build.sh script. Your password, as for everything else, is "Microsoft".



Once inside the console, navigate into the "Projects" link on the left side of the screen and click on the project with your username with an "s2i" extension. For example: "skirk12345-s2i"



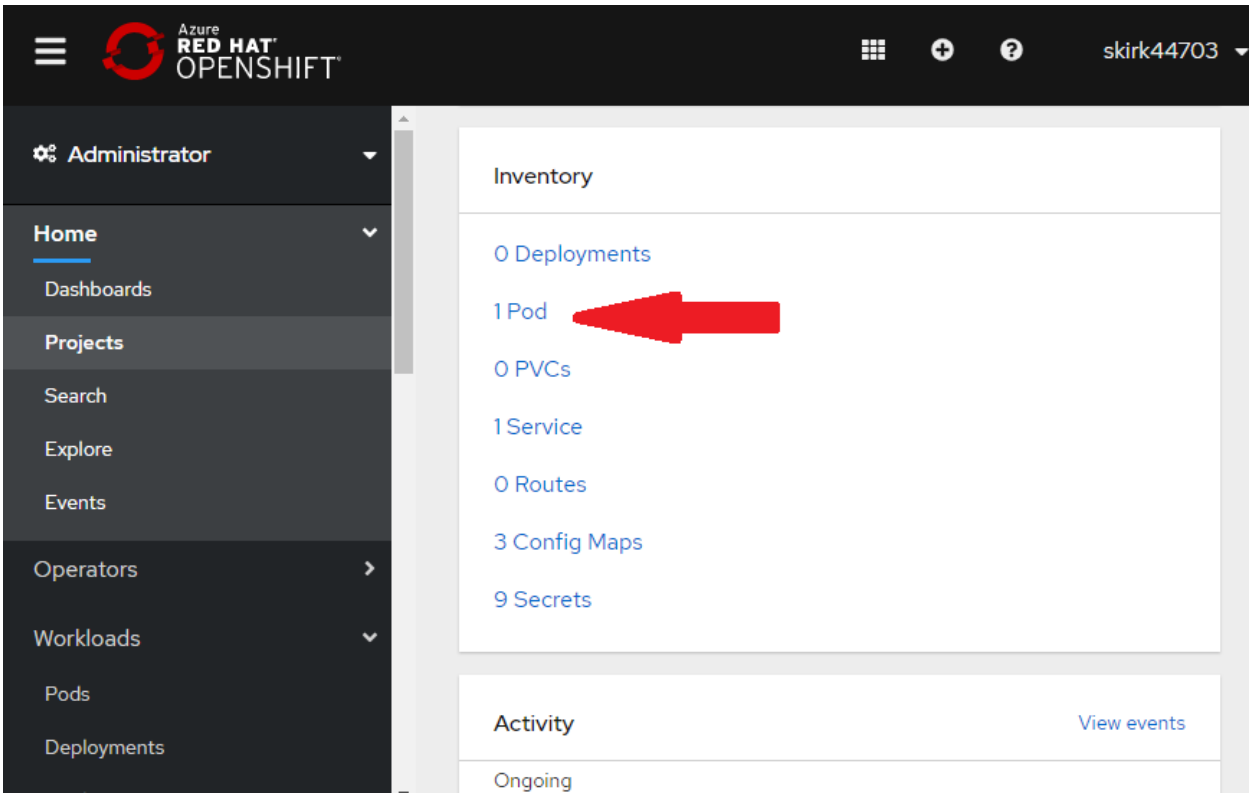
Switch back to your terminal window, and execute the following commands to log in to the OpenShift cluster, enter the project created by Ansible, and start the Source-To-Image build:

- `oc login <ADDRESS-OF-ARO-API:6443>`
- `oc project <YOUR-ARO-USERNAME-s2i>`
- `oc new-app \`
- `php:7.1~https://github.com/stuartatmicrosoft/RedHatSummit2020 \`
- `--context-dir php-helloworld --name php-helloworld`

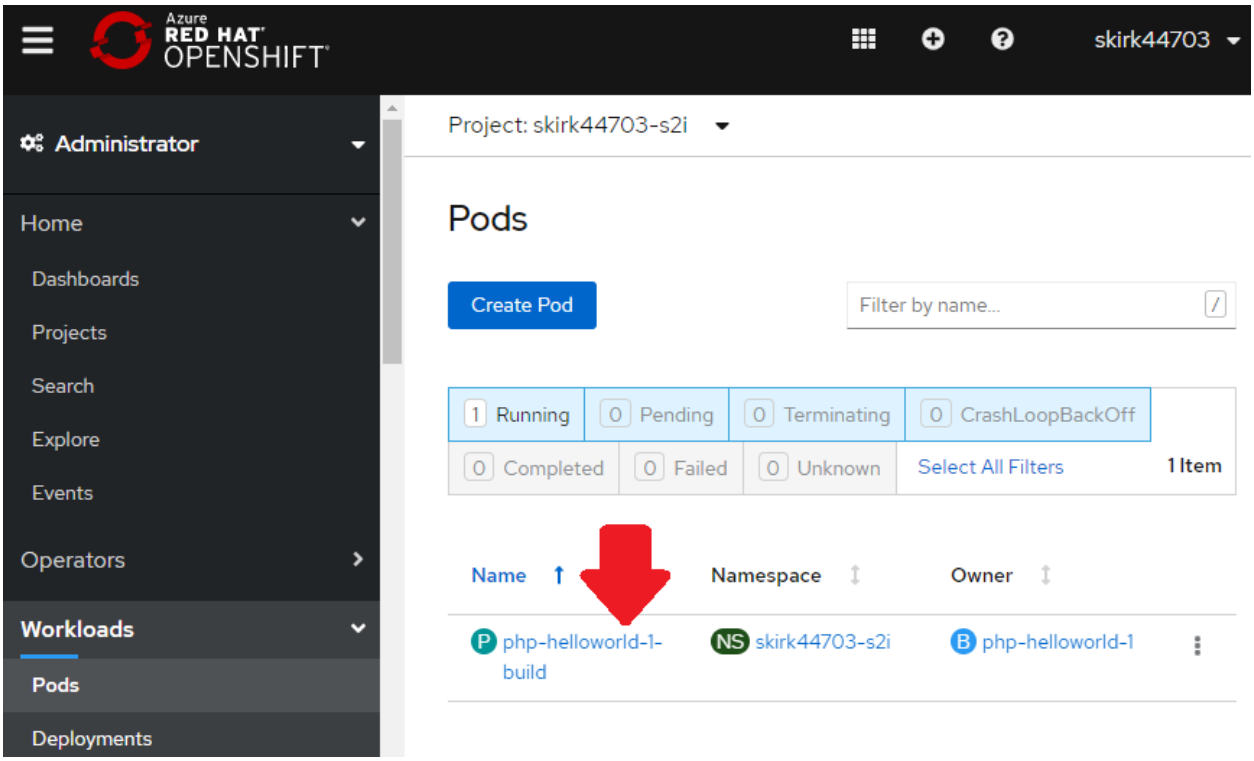
You should see output that the source-to-image build has begun as follows:

```
You have access to 56 projects, the list has been suppressed. You can list all projects wi
th 'oc projects'

Using project "default".
[student@master-vnc-desktop playbooks]$ oc project skirk44703-s2i
Now using project "skirk44703-s2i" on server "https://api.xvoqh9s9.eastus.aroapp.io:6443".
[student@master-vnc-desktop playbooks]$ oc new-app \
> php:7.1~https://github.com/stuartatmicrosoft/RedHatSummit2020 \
> --context-dir php-helloworld --name php-helloworld
--> Found image 8e01e80 (4 months old) in image stream "openshift/php" under tag "7.1" for
 "php:7.1"

    Apache 2.4 with PHP 7.1
    -----------------------
    PHP 7.1 available as container is a base platform for building and running various PHP
 7.1 applications and frameworks. PHP is an HTML-embedded scripting language. PHP attempts
 to make it easy for developers to write dynamically generated web pages. PHP also offers
built-in database integration for several commercial and non-commercial database managemen
t systems, so writing a database-enabled webpage with PHP is fairly simple. The most commo
n use of PHP coding is probably as a replacement for CGI scripts.

    Tags: builder, php, php71, rh-php71

    * A source build using source code from https://github.com/stuartatmicrosoft/RedHatSum
mit2020 will be created
      * The resulting image will be pushed to image stream tag "php-helloworld:latest"
      * Use 'oc start-build' to trigger a new build
    * This image will be deployed in deployment config "php-helloworld"
    * Ports 8080/tcp, 8443/tcp will be load balanced by service "php-helloworld"
      * Other containers can access this service through the hostname "php-helloworld"

--> Creating resources ...
    imagestream.image.openshift.io "php-helloworld" created
    buildconfig.build.openshift.io "php-helloworld" created
    deploymentconfig.apps.openshift.io "php-helloworld" created
    service "php-helloworld" created
--> Success
    Build scheduled, use 'oc logs -f bc/php-helloworld' to track its progress.
    Application is not exposed. You can expose services to the outside world by executing
one or more of the commands below:
     'oc expose svc/php-helloworld'
    Run 'oc status' to view your app.
```

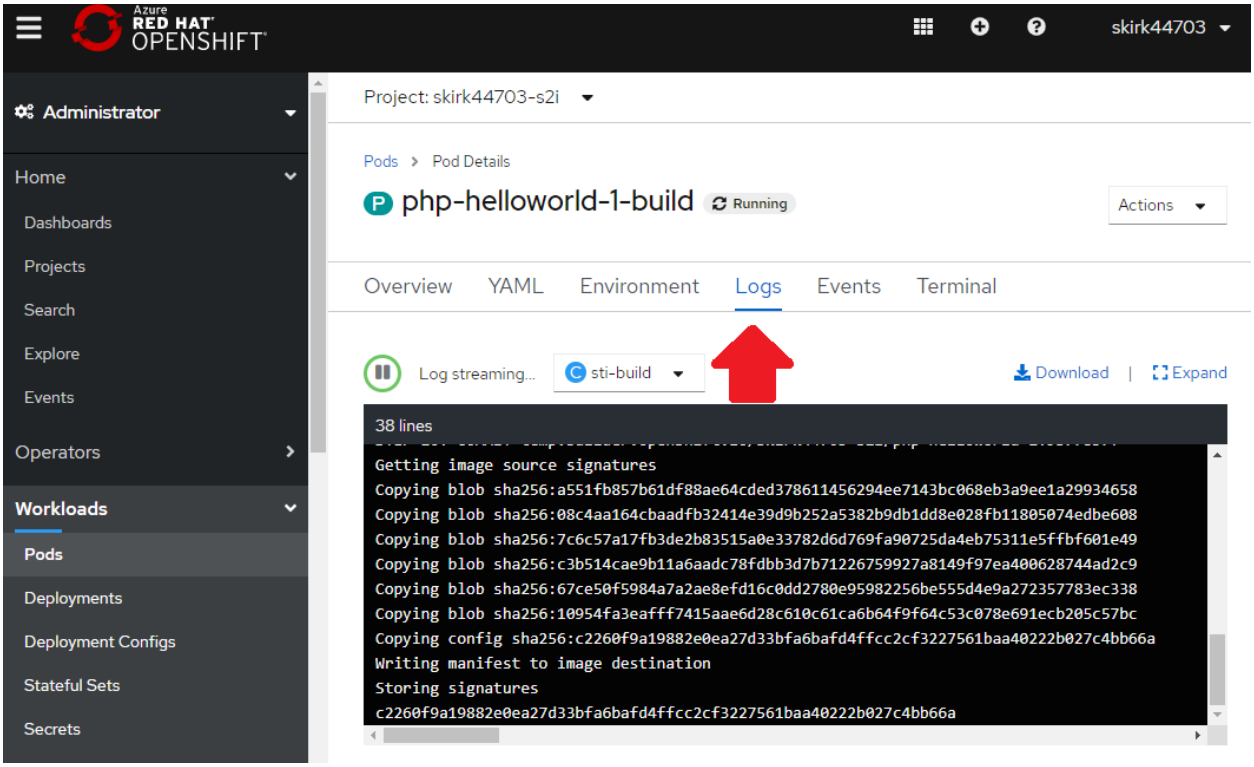
After you verify that the source-to-image build is underway, quickly switch back to the Azure Red Hat OpenShift console, and click-on the "1 Pod" reference on the Project Inventory screen:



Click the name of the pod in which is the build currently underway:



Click on the "Logs" lab on the "Pod Details" screen and observe the build log as it progresses in real-time and provisions the container with the required supporting infrastructure:



You can also watch the logs of the build from the terminal by using the command:

- `oc logs -f bc/php-helloworld`

Switch back to your terminal window and expose the service to create an external route. Subsequently query for the route which was just created:

- `oc expose svc/php-helloworld`
- `oc get route`

```
[student@master-vnc-desktop playbooks]$ oc expose svc/php-helloworld
route.route.openshift.io/php-helloworld exposed
[student@master-vnc-desktop playbooks]$ oc get route
NAME              HOST/PORT                                                  PATH   SER
VICES         PORT       TERMINATION   WILDCARD
php-helloworld    php-helloworld-skirk44703-s2i.apps.xvoqh9s9.eastus.aroapp.io          php
-helloworld   8080-tcp                 None
[student@master-vnc-desktop playbooks]$
```

Visit the URL which was provided. In this case:

http://php-helloworld-skirk44703-s2i.apps.xvoqh9s9.eastus.aroapp.io/

Playbook 1 – Deploy Microsoft SQL Server to Azure Red Hat OpenShift

Estimated Playbook Runtime: 0m 10s

Perform the following pre-deployment tasks:

Verify that you are logged in to the OpenShift CLI as your username:

- **oc whoami**

```
[student@master-vnc-desktop playbooks]$ oc whoami
skirk49172
```

Switch to the project (namespace) created for Microsoft SQL Server in Playbook #0

- **oc project <YOUR-USERNAME>-mssql-01**

```
[student@master-vnc-desktop playbooks]$ oc project skirk49172-mssql-01
Now using project "skirk49172-mssql-01" on server "https://api.bfz8cqn4.
eastus.aroapp.io:6443".
```

Create the administrator password for Microsoft SQL Server as a Kubernetes secret

- **oc create secret generic mssql --from-literal=SA_PASSWORD="Ansible123456$$"**

```
[student@master-vnc-desktop playbooks]$ oc create secret generic mssql --from-literal=SA_P
ASSWORD="Ansible123456$$"
secret/mssql created
```

Run the Ansible Playbook

- **ansible-playbook aro-01-deploy-mssql-server.yaml**

```
[student@master-vnc-desktop playbooks]$ time ansible-playbook aro-01-deploy-mssql-
server.yaml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'

PLAY [Deploy Microsoft SQL Server to Azure Red Hat OpenShift] ********************

TASK [Gathering Facts] ***********************************************************
ok: [localhost]

TASK [Log in to Azure Red Hat OpenShift (ARO)] ***********************************
ok: [localhost]

TASK [Save API key variable] *****************************************************
ok: [localhost]

TASK [Create storage for Microsoft SQL Server] ***********************************
changed: [localhost]

TASK [Deploy Microsoft SQL Server to Azure Red Hat OpenShift] ********************
changed: [localhost]

TASK [debug] *********************************************************************
ok: [localhost] => {
    "msg": "Don't forget to run 'oc get svc' until you see the external IP address
."
}

PLAY RECAP ***********************************************************************
localhost                  : ok=6    changed=2    unreachable=0    failed=0    ski
pped=0    rescued=0    ignored=0


real    0m10.506s
user    0m6.221s
sys     0m0.999s
[student@master-vnc-desktop playbooks]$
```

Determine the EXTERNAL-IP address which is assigned to Microsoft SQL Server

- **watch oc get svc**

```
Every 2.0s: oc get svc                          master-vnc-desktop: Sat Apr 18 04:43:03 2020

NAME               TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
mssql-deployment   LoadBalancer   172.30.53.170   20.185.101.158   1433:32670/TCP   12m
```

View the deployment logs and wait for the Microsoft SQL Server container to be created and made ready for use:

- **oc logs -f deployments/mssql-deployment**

```
[student@master-vnc-desktop playbooks]$ oc logs -f deployments/mssql-deployment
SQL Server 2019 will run as non-root by default.
This container is running as user 1000550000.
To learn more visit https://go.microsoft.com/fwlink/?linkid=2099216.
2020-04-18 20:39:54.20 Server      Setup step is copying system data file 'C:\temp
latedata\master.mdf' to '/var/opt/mssql/data/master.mdf'.
2020-04-18 20:39:54.31 Server      Did not find an existing master data file /var/
opt/mssql/data/master.mdf, copying the missing default master and other system dat
abase files. If you have moved the database location, but not moved the database f
iles, startup may fail. To repair: shutdown SQL Server, move the master database t
o configured location, and restart.
2020-04-18 20:39:54.33 Server      Setup step is copying system data file 'C:\temp
latedata\mastlog.ldf' to '/var/opt/mssql/data/mastlog.ldf'.
2020-04-18 20:39:54.35 Server      Setup step is copying system data file 'C:\temp
latedata\model.mdf' to '/var/opt/mssql/data/model.mdf'.
2020-04-18 20:39:54.37 Server      Setup step is copying system data file 'C:\temp
latedata\modellog.ldf' to '/var/opt/mssql/data/modellog.ldf'.
2020-04-18 20:39:54.40 Server      Setup step is copying system data file 'C:\temp
latedata\msdbdata.mdf' to '/var/opt/mssql/data/msdbdata.mdf'.
2020-04-18 20:39:54.42 Server      Setup step is copying system data file 'C:\temp
latedata\msdblog.ldf' to '/var/opt/mssql/data/msdblog.ldf'.
2020-04-18 20:39:54.55 Server      Microsoft SQL Server 2017 (RTM-CU20) (KB4541283
) - 14.0.3294.2 (X64)
        Mar 13 2020 14:53:45
        Copyright (C) 2017 Microsoft Corporation
        Developer Edition (64-bit) on Linux (Ubuntu 16.04.6 LTS)
2020-04-18 20:39:54.56 Server      UTC adjustment: 0:00
2020-04-18 20:39:54.56 Server      (c) Microsoft Corporation.
2020-04-18 20:39:54.56 Server      All rights reserved.
2020-04-18 20:39:54.57 Server      Server process ID is 40.
2020-04-18 20:39:54.57 Server      Logging SQL Server messages in file '/var/opt/m
ssql/log/errorlog'.
2020-04-18 20:39:54.57 Server      Registry startup parameters:
         -d /var/opt/mssql/data/master.mdf
         -l /var/opt/mssql/data/mastlog.ldf
```

Connect to Microsoft SQL Server 2017 and verify connectivity by listing the preset static variable in the database. Subsequently exit sqlcmd.

- **sqlcmd -S <YOUR-EXTERNAL-IP> -U sa -P "Ansible123456$$"**
- **:Listvar**
- **:Quit**

```
[student@master-vnc-desktop playbooks]$ sqlcmd -S 20.185.101.158 -U sa -P "Ansible
123456$$"
1> :Listvar
SQLCMDCOLSEP = " "
SQLCMDCOLWIDTH = "0"
SQLCMDDBNAME = ""
SQLCMDEDITOR = "edit.com"
SQLCMDERRORLEVEL = "0"
SQLCMDHEADERS = "0"
SQLCMDINI = ""
SQLCMDLOGINTIMEOUT = "8"
SQLCMDMAXFIXEDTYPEWIDTH = "0"
SQLCMDMAXVARTYPEWIDTH = "256"
SQLCMDPACKETSIZE = "4096"
SQLCMDSERVER = "20.185.101.158"
SQLCMDSTATTIMEOUT = "0"
SQLCMDUSER = "sa"
SQLCMDWORKSTATION = "master-vnc-desktop"
1> :QUIT
[student@master-vnc-desktop playbooks]$
```

You have just deployed Microsoft SQL Server 2017, using Ansible, inside Azure Red Hat OpenShift, running on Microsoft Azure. Congratulations!

You've reached the end!

Thank you for your participation!

Please take the survey for this lab!

UPDATE FOR 2020