DEPLOYING MICROSOFT AZURE 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

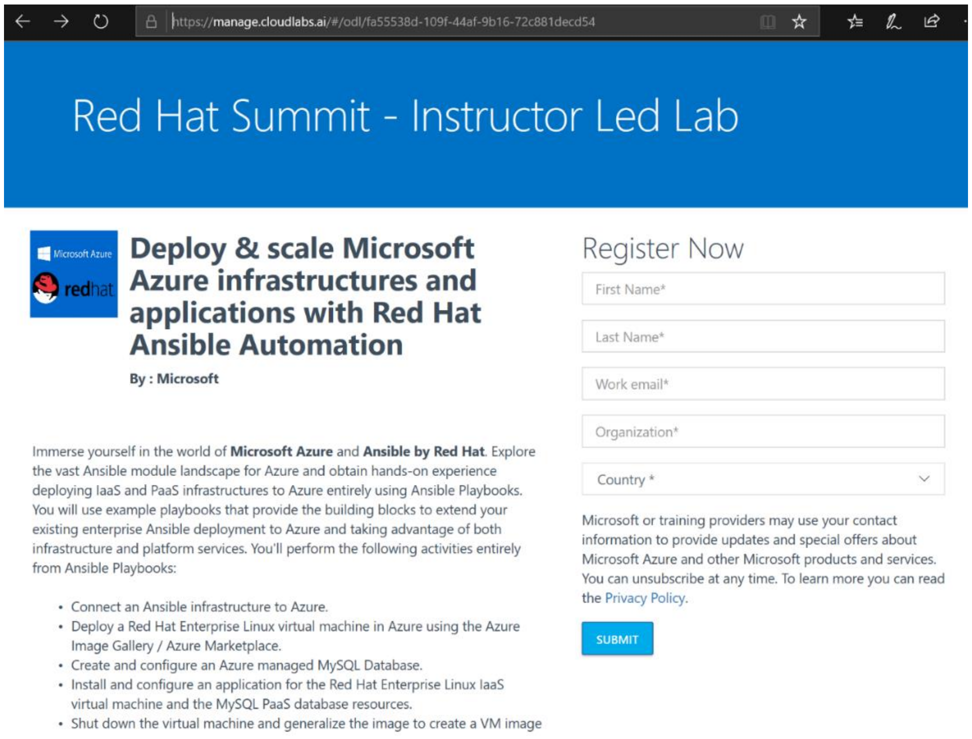
TABLE OF CONTENTS

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 7.6 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


https://manage.cloudlabs.ai/#/odl/fa55538d-109f-44af-9b16-72c881decd54

Red Hat Summit - Instructor Led Lab

Deploy & scale Microsoft Azure infrastructures and applications with Red Hat Ansible Automation

By : Microsoft

Immerse yourself in the world of **Microsoft Azure** and **Ansible by Red Hat**. Explore the vast Ansible module landscape for Azure and obtain hands-on experience deploying IaaS and PaaS infrastructures to Azure entirely using Ansible Playbooks. You will use example playbooks that provide the building blocks to extend your existing enterprise Ansible deployment to Azure and taking advantage of both infrastructure and platform services. You'll perform the following activities entirely from Ansible Playbooks:

- Connect an Ansible infrastructure to Azure.
- Deploy a Red Hat Enterprise Linux virtual machine in Azure using the Azure Image Gallery / Azure Marketplace.
- Create and configure an Azure managed MySQL Database.
- Install and configure an application for the Red Hat Enterprise Linux IaaS virtual machine and the MySQL PaaS database resources.
- Shut down the virtual machine and generalize the image to create a VM image

Register Now

First Name*

Last Name*

Work email*

Organization*

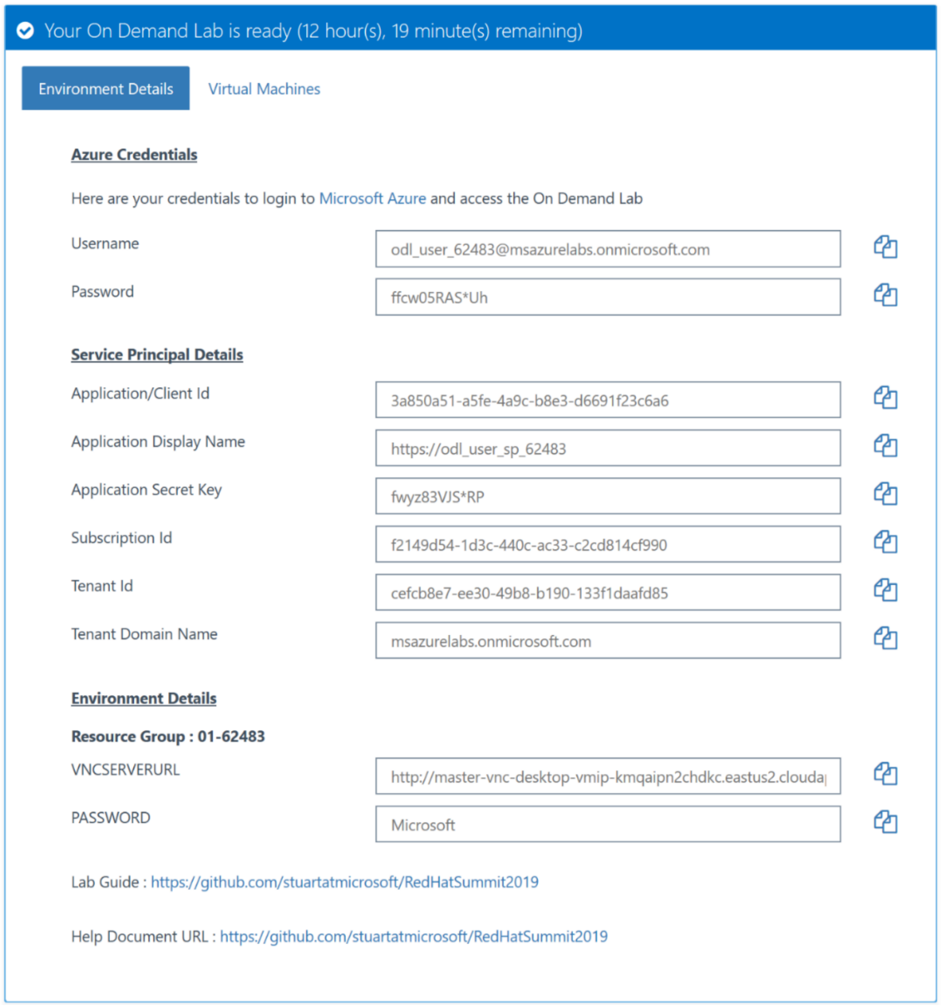
Country *

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.

Accessing your Lab VM

Access to the Lab VM is provided by noVNC, an HTML5-based VNC client; Any modern HTML5-capable web browser should be able to access it. Upon registration submission, you should receive the required credentials to access your Lab VM. 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


Your On Demand Lab is ready (12 hour(s), 19 minute(s) remaining)

Environment Details | Virtual Machines

Azure Credentials

Here are your credentials to login to Microsoft Azure and access the On Demand Lab

Username	odl_user_62483@msazurelabs.onmicrosoft.com
Password	ffcw05RAS*Uh

Service Principal Details

Application/Client Id	3a850a51-a5fe-4a9c-b8e3-d6691f23c6a6
Application Display Name	https://odl_user_sp_62483
Application Secret Key	fwyz83VJS*RP
Subscription Id	f2149d54-1d3c-440c-ac33-c2cd814cf990
Tenant Id	cefcb8e7-ee30-49b8-b190-133f1daafd85
Tenant Domain Name	msazurelabs.onmicrosoft.com

Environment Details

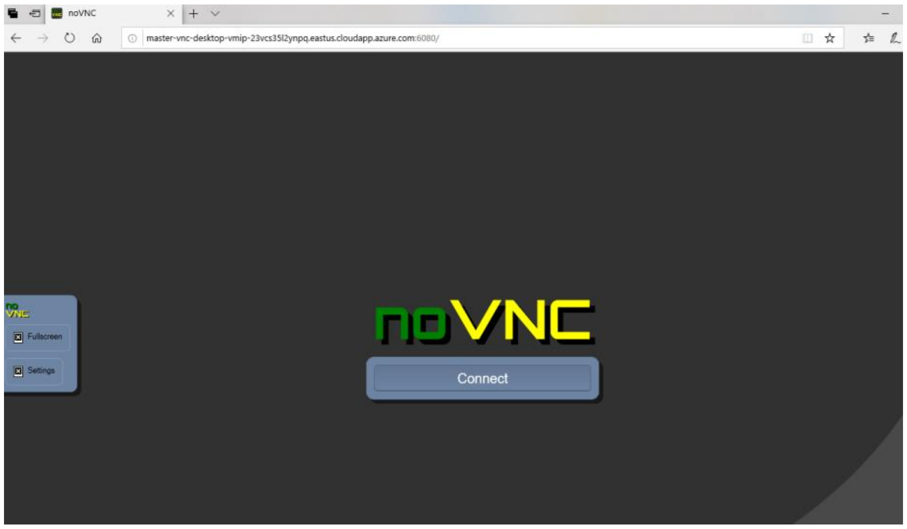
Resource Group : 01-62483

VNCSERVERURL	http://master-vnc-desktop-vmip-kmqaipn2chdkc.eastus2.clouda
PASSWORD	Microsoft

Lab Guide : https://github.com/stuartatmicrosoft/RedHatSummit2019

Help Document URL : https://github.com/stuartatmicrosoft/RedHatSummit2019

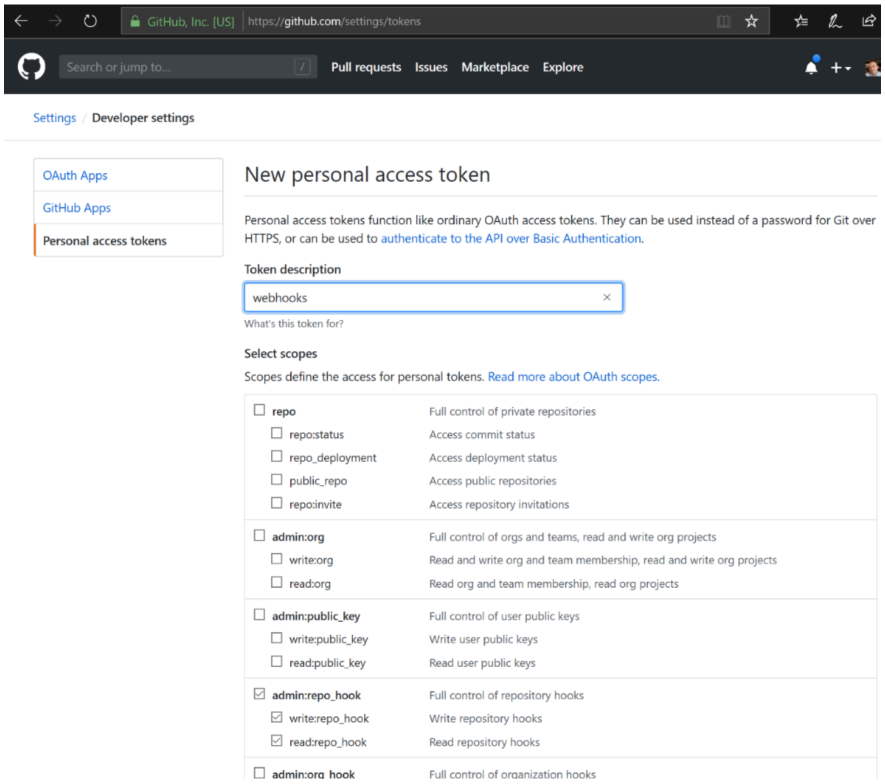
➢ Enter the VNCSERVERURL into your web browser and log in to the VNC Lab VM


master-vnc-desktop-vmip-23vcs35l2ynpq.eastus.cloudapp.azure.com:6080/

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.



GitHub, Inc. [US] https://github.com/settings/tokens

Settings / Developer settings

OAuth Apps
GitHub Apps
Personal access tokens

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Token description: webhooks

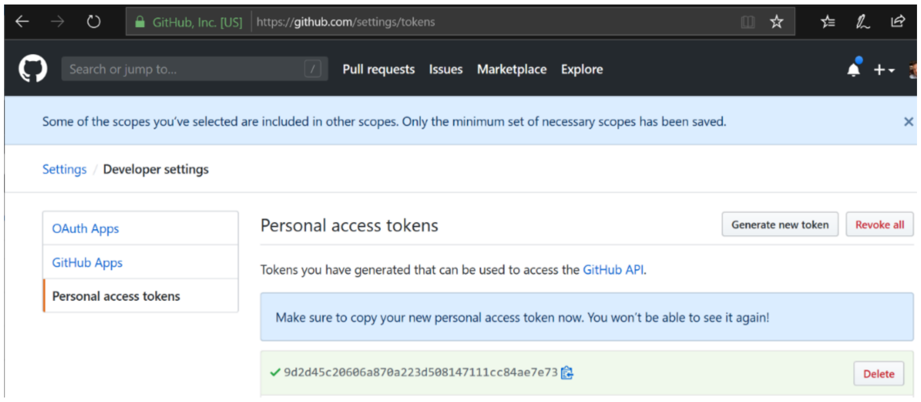
What's this token for?

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

☐ repo	Full control of private repositories
☐ repo:status	Access commit status
☐ repo_deployment	Access deployment status
☐ public_repo	Access public repositories
☐ repo:invite	Access repository invitations
☐ admin:org	Full control of orgs and teams, read and write org projects
☐ write:org	Read and write org and team membership, read and write org projects
☐ read:org	Read org and team membership, read org projects
☐ admin:public_key	Full control of user public keys
☐ write:public_key	Write user public keys
☐ read:public_key	Read user public keys
☑ admin:repo_hook	Full control of repository hooks
☑ write:repo_hook	Write repository hooks
☑ read:repo_hook	Read repository hooks
☐ admin:org_hook	Full control of organization hooks

- <u>Take note of the token after it is generated!</u>



GitHub, Inc. [US] https://github.com/settings/tokens

Some of the scopes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.

Settings / Developer settings

OAuth Apps
GitHub Apps
Personal access tokens

Personal access tokens

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your new personal access token now. You won't be able to see it again!

✓ 9d2d45c20606a870a223d508147111cc84ae7e73

Login to the Azure Linux CLI

- Open the credentials.txt file on your desktop
- AZ_USER_NAME is your Azure Linux CLI & Portal Username
- AZ_USER_PASSWORD is your Azure Linux CLI & Portal Password
- `az login`

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. 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

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

Obtaining & Preparing your Labs

- Open a Linux Shell
- `git clone https://github.com/stuartatmicrosoft/RedHatSummit2019`
- `cd RedHatSummit2019`

To begin working through the lab exercises, you will need to generate your own Ansible variables file. Open the "vars.yml" file located in the "playbooks" directory in your favorite editor and set values for the following:

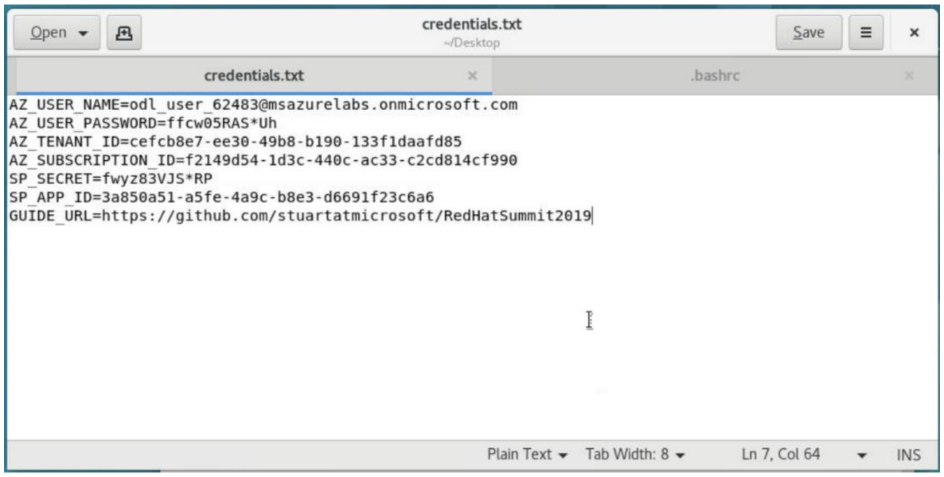
resource_group:	(Given on the credential screen or "`az group list`")
vm_name:	(A prefix for the Azure resources you create)
location:	(Your Azure DC eastus/northcentralus – Check VNC URL)
github_id:	(Your GitHub ID)
github:token:	(Your Personal Access Token just generated)

Save the file and run the random-number generator script. Your personal variables file, vars-myvars.yml has now been created.

- `./randomize.sh`

Environment variables in your ~/.bashrc need to be 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. Open your ~/.bashrc file and paste in the service principal credential information:

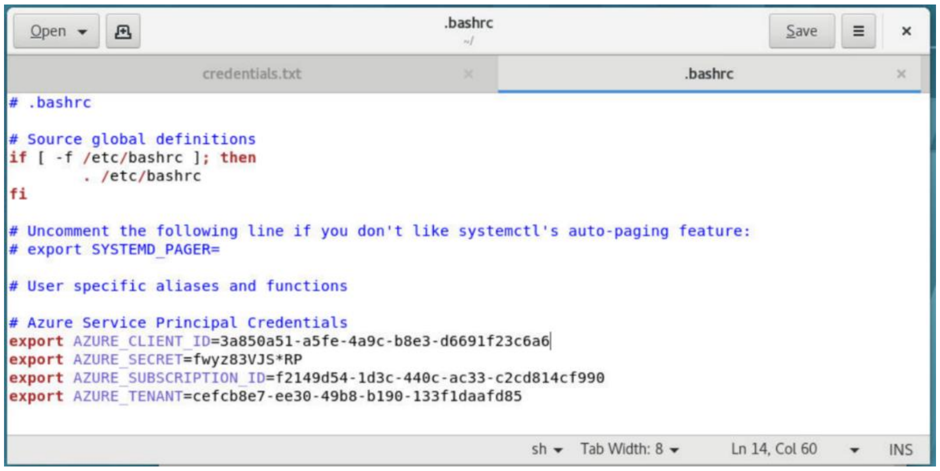
- Open the credentials.txt file on your desktop



```
AZ_USER_NAME=odl_user_62483@msazurelabs.onmicrosoft.com
AZ_USER_PASSWORD=ffcw05RAS*Uh
AZ_TENANT_ID=cefcb8e7-ee30-49b8-b190-133f1daafd85
AZ_SUBSCRIPTION_ID=f2149d54-1d3c-440c-ac33-c2cd814cf990
SP_SECRET=fwyz83VJS*RP
SP_APP_ID=3a850a51-a5fe-4a9c-b8e3-d6691f23c6a6
GUIDE_URL=https://github.com/stuartatmicrosoft/RedHatSummit2019
```

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

.bashrc	credentials.txt
AZURE_CLIENT_ID	SP_APP_ID
AZURE_SECRET	SP_SECRET
AZURE_SUBSCRIPTION_ID	AZ_SUBSCRIPTION_ID
AZURE_TENANT	AZ_TENANT_ID



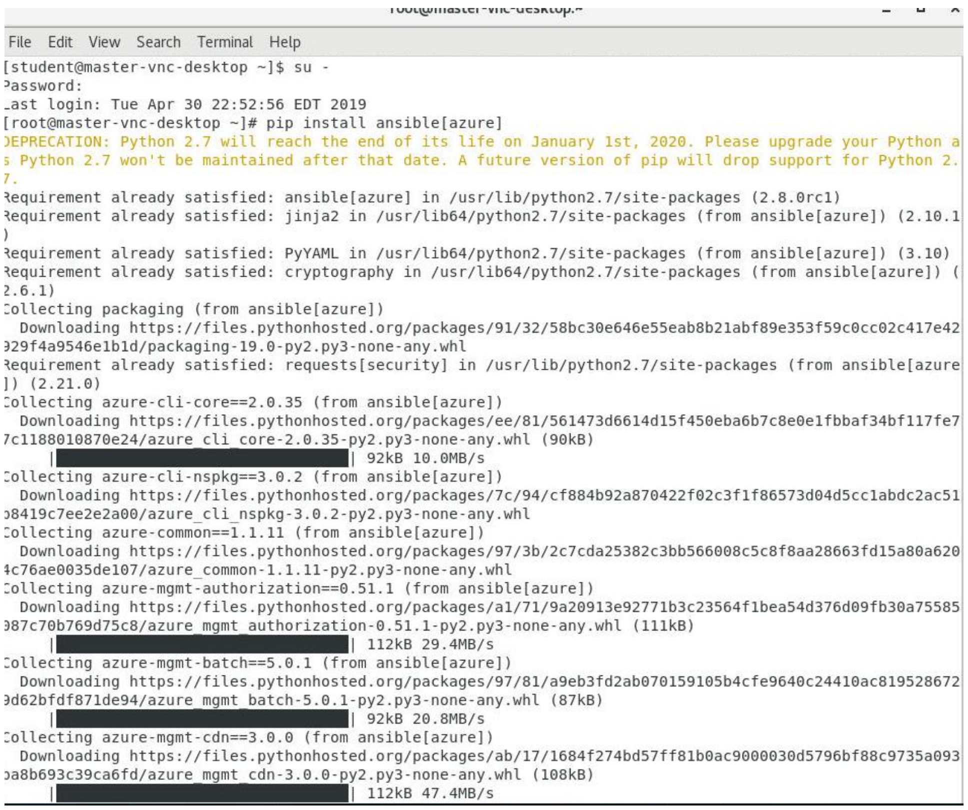
```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions

# Azure Service Principal Credentials
export AZURE_CLIENT_ID=3a850a51-a5fe-4a9c-b8e3-d6691f23c6a6
export AZURE_SECRET=fwyz83VJS*RP
export AZURE_SUBSCRIPTION_ID=f2149d54-1d3c-440c-ac33-c2cd814cf990
export AZURE_TENANT=cefcb8e7-ee30-49b8-b190-133f1daafd85
```

- Do not forget to `source .bashrc` or close and re-open your current shell window!

Install the Azure Modules for Ansible

Ansible 2.8.0rc1 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:

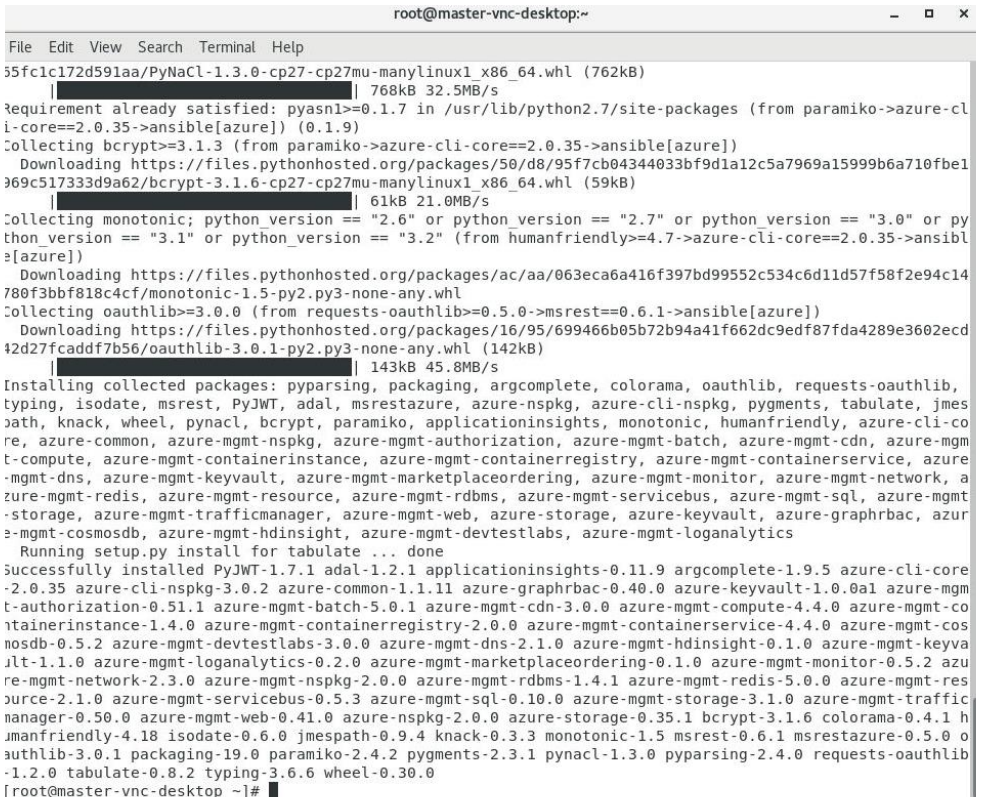
- Become root user: `su -` (The password for all accounts is Microsoft)
- `pip install ansible[azure]`



```
[student@master-vnc-desktop ~]$ su -
Password:
Last login: Tue Apr 30 22:52:56 EDT 2019
[root@master-vnc-desktop ~]# pip install ansible[azure]
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python a
s Python 2.7 won't be maintained after that date. A future version of pip will drop support for Python 2.
7.
Requirement already satisfied: ansible[azure] in /usr/lib/python2.7/site-packages (2.8.0rc1)
Requirement already satisfied: jinja2 in /usr/lib64/python2.7/site-packages (from ansible[azure]) (2.10.1
)
Requirement already satisfied: PyYAML in /usr/lib64/python2.7/site-packages (from ansible[azure]) (3.10)
Requirement already satisfied: cryptography in /usr/lib64/python2.7/site-packages (from ansible[azure]) (
2.6.1)
Collecting packaging (from ansible[azure])
  Downloading https://files.pythonhosted.org/packages/91/32/58bc30e646e55eab8b21abf89e353f59c0cc02c417e42
929f4a9546e1b1d/packaging-19.0-py2.py3-none-any.whl
Requirement already satisfied: requests[security] in /usr/lib/python2.7/site-packages (from ansible[azure
]) (2.21.0)
Collecting azure-cli-core==2.0.35 (from ansible[azure])
  Downloading https://files.pythonhosted.org/packages/ee/81/561473d6614d15f450eba6b7c8e0e1fbbaf34bf117fe7
7c1188010870e24/azure_cli_core-2.0.35-py2.py3-none-any.whl (90kB)
    |████████████████████████████████| 92kB 10.0MB/s
Collecting azure-cli-nspkg==3.0.2 (from ansible[azure])
  Downloading https://files.pythonhosted.org/packages/7c/94/cf884b92a870422f02c3f1f86573d04d5cc1abdc2ac51
b8419c7ee2e2a00/azure_cli_nspkg-3.0.2-py2.py3-none-any.whl
Collecting azure-common==1.1.11 (from ansible[azure])
  Downloading https://files.pythonhosted.org/packages/97/3b/2c7cda25382c3bb566008c5c8f8aa28663fd15a80a620
4c76ae0035de107/azure_common-1.1.11-py2.py3-none-any.whl
Collecting azure-mgmt-authorization==0.51.1 (from ansible[azure])
  Downloading https://files.pythonhosted.org/packages/a1/71/9a20913e92771b3c23564f1bea54d376d09fb30a75585
987c70b769d75c8/azure_mgmt_authorization-0.51.1-py2.py3-none-any.whl (111kB)
    |████████████████████████████████| 112kB 29.4MB/s
Collecting azure-mgmt-batch==5.0.1 (from ansible[azure])
  Downloading https://files.pythonhosted.org/packages/97/81/a9eb3fd2ab070159105b4cfe9640c24410ac819528672
9d62bfdf871de94/azure_mgmt_batch-5.0.1-py2.py3-none-any.whl (87kB)
    |████████████████████████████████| 92kB 20.8MB/s
Collecting azure-mgmt-cdn==3.0.0 (from ansible[azure])
  Downloading https://files.pythonhosted.org/packages/ab/17/1684f274bd57ff81b0ac9000030d5796bf88c9735a093
ba8b693c39ca6fd/azure_mgmt_cdn-3.0.0-py2.py3-none-any.whl (108kB)
    |████████████████████████████████| 112kB 47.4MB/s
```

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:


root@master-vnc-desktop:~
File Edit View Search Terminal Help

```
55fc1c172d591aa/PyNaCl-1.3.0-cp27-cp27mu-manylinux1_x86_64.whl (762kB)
     |████████████████████████████████| 768kB 32.5MB/s
Requirement already satisfied: pyasn1>=0.1.7 in /usr/lib/python2.7/site-packages (from paramiko->azure-cl
i-core==2.0.35->ansible[azure]) (0.1.9)
Collecting bcrypt>=3.1.3 (from paramiko->azure-cli-core==2.0.35->ansible[azure])
  Downloading https://files.pythonhosted.org/packages/50/d8/95f7cb04344033bf9d1a12c5a7969a15999b6a710fbe1
969c517333d9a62/bcrypt-3.1.6-cp27-cp27mu-manylinux1_x86_64.whl (59kB)
     |████████████████████████████████| 61kB 21.0MB/s
Collecting monotonic; python_version == "2.6" or python_version == "2.7" or python_version == "3.0" or py
thon_version == "3.1" or python_version == "3.2" (from humanfriendly>=4.7->azure-cli-core==2.0.35->ansibl
e[azure])
  Downloading https://files.pythonhosted.org/packages/ac/aa/063eca6a416f397bd99552c534c6d11d57f58f2e94c14
780f3bbf818c4cf/monotonic-1.5-py2.py3-none-any.whl
Collecting oauthlib>=3.0.0 (from requests-oauthlib>=0.5.0->msrest==0.6.1->ansible[azure])
  Downloading https://files.pythonhosted.org/packages/16/95/699466b05b72b94a41f662dc9edf87fda4289e3602ecd
42d27fcaddf7b56/oauthlib-3.0.1-py2.py3-none-any.whl (142kB)
     |████████████████████████████████| 143kB 45.8MB/s
Installing collected packages: pyparsing, packaging, argcomplete, colorama, oauthlib, requests-oauthlib,
typing, isodate, msrest, PyJWT, adal, msrestazure, azure-nspkg, azure-cli-nspkg, pygments, tabulate, jmes
path, knack, wheel, pynacl, bcrypt, paramiko, applicationinsights, monotonic, humanfriendly, azure-cli-co
re, azure-common, azure-mgmt-nspkg, azure-mgmt-authorization, azure-mgmt-batch, azure-mgmt-cdn, azure-mgm
t-compute, azure-mgmt-containerinstance, azure-mgmt-containerregistry, azure-mgmt-containerservice, azure
-mgmt-dns, azure-mgmt-keyvault, azure-mgmt-marketplaceordering, azure-mgmt-monitor, azure-mgmt-network, a
zure-mgmt-redis, azure-mgmt-resource, azure-mgmt-rdbms, azure-mgmt-servicebus, azure-mgmt-sql, azure-mgmt
-storage, azure-mgmt-trafficmanager, azure-mgmt-web, azure-storage, azure-keyvault, azure-graphrbac, azur
e-mgmt-cosmosdb, azure-mgmt-hdinsight, azure-mgmt-devtestlabs, azure-mgmt-loganalytics
  Running setup.py install for tabulate ... done
Successfully installed PyJWT-1.7.1 adal-1.2.1 applicationinsights-0.11.9 argcomplete-1.9.5 azure-cli-core
-2.0.35 azure-cli-nspkg-3.0.2 azure-common-1.1.11 azure-graphrbac-0.40.0 azure-keyvault-1.0.0a1 azure-mgm
t-authorization-0.51.1 azure-mgmt-batch-5.0.1 azure-mgmt-cdn-3.0.0 azure-mgmt-compute-4.4.0 azure-mgmt-co
ntainerinstance-1.4.0 azure-mgmt-containerregistry-2.0.0 azure-mgmt-containerservice-4.4.0 azure-mgmt-cos
mosdb-0.5.2 azure-mgmt-devtestlabs-3.0.0 azure-mgmt-dns-2.1.0 azure-mgmt-hdinsight-0.1.0 azure-mgmt-keyva
ult-1.1.0 azure-mgmt-loganalytics-0.2.0 azure-mgmt-marketplaceordering-0.1.0 azure-mgmt-monitor-0.5.2 azu
re-mgmt-network-2.3.0 azure-mgmt-nspkg-2.0.0 azure-mgmt-rdbms-1.4.1 azure-mgmt-redis-5.0.0 azure-mgmt-res
ource-2.1.0 azure-mgmt-servicebus-0.5.3 azure-mgmt-sql-0.10.0 azure-mgmt-storage-3.1.0 azure-mgmt-traffic
manager-0.50.0 azure-mgmt-web-0.41.0 azure-nspkg-2.0.0 azure-storage-0.35.1 bcrypt-3.1.6 colorama-0.4.1 h
umanfriendly-4.18 isodate-0.6.0 jmespath-0.9.4 knack-0.3.3 monotonic-1.5 msrest-0.6.1 msrestazure-0.5.0 o
authlib-3.0.1 packaging-19.0 paramiko-2.4.2 pygments-2.3.1 pynacl-1.3.0 pyparsing-2.4.0 requests-oauthlib
-1.2.0 tabulate-0.8.2 typing-3.6.6 wheel-0.30.0
[root@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

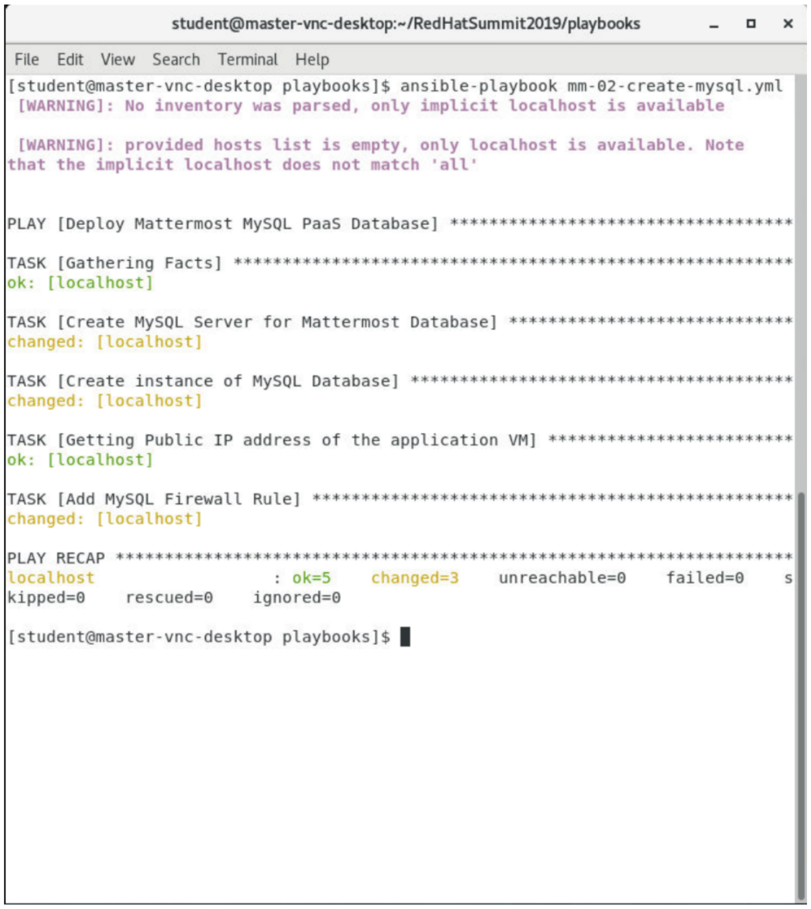
Estimated Playbook Runtime: 2m 53s

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

Playbook 2 – Deploying MySQL PaaS

Estimated Playbook Runtime: 3m 39s

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

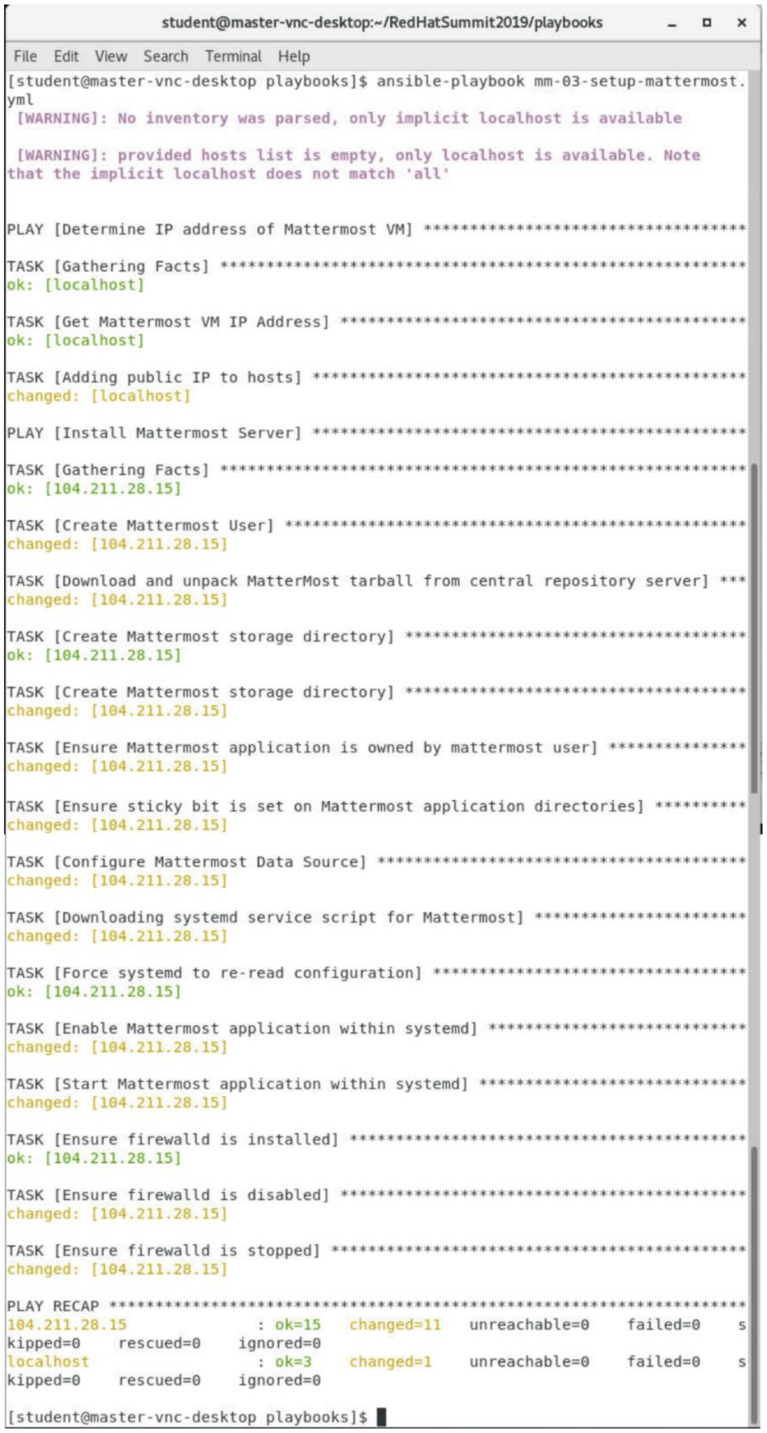

student@master-vnc-desktop:~/RedHatSummit2019/playbooks
File Edit View Search Terminal Help

```
[student@master-vnc-desktop playbooks]$ 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    s
kipped=0    rescued=0    ignored=0

[student@master-vnc-desktop playbooks]$
```

Playbook 3 – Deploying & Configuring Mattermost

Estimated Playbook Runtime: 0m 59s

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



student@master-vnc-desktop:~/RedHatSummit2019/playbooks

File Edit View Search Terminal Help

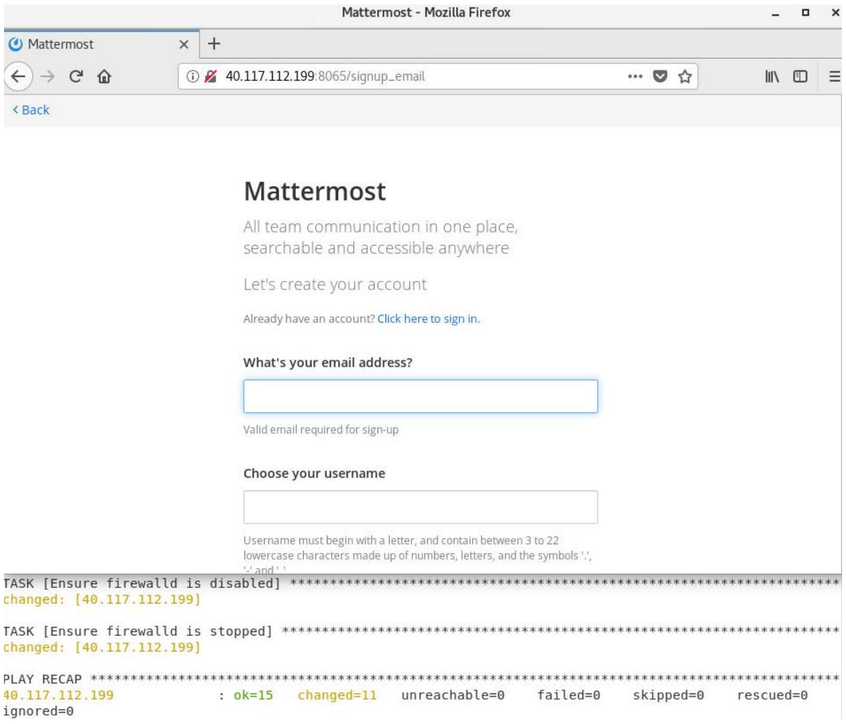
```
[student@master-vnc-desktop playbooks]$ ansible-playbook mm-03-setup-mattermost.
yml
 [WARNING]: No inventory was parsed, only implicit localhost is available

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


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

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

TASK [Get Mattermost VM IP Address] ********************************************
ok: [localhost]

TASK [Adding public IP to hosts] ***********************************************
changed: [localhost]

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

TASK [Gathering Facts] *********************************************************
ok: [104.211.28.15]

TASK [Create Mattermost User] **************************************************
changed: [104.211.28.15]

TASK [Download and unpack MatterMost tarball from central repository server] ***
changed: [104.211.28.15]

TASK [Create Mattermost storage directory] *************************************
ok: [104.211.28.15]

TASK [Create Mattermost storage directory] *************************************
changed: [104.211.28.15]

TASK [Ensure Mattermost application is owned by mattermost user] ***************
changed: [104.211.28.15]

TASK [Ensure sticky bit is set on Mattermost application directories] **********
changed: [104.211.28.15]

TASK [Configure Mattermost Data Source] ****************************************
changed: [104.211.28.15]

TASK [Downloading systemd service script for Mattermost] ***********************
changed: [104.211.28.15]

TASK [Force systemd to re-read configuration] **********************************
ok: [104.211.28.15]

TASK [Enable Mattermost application within systemd] ****************************
changed: [104.211.28.15]

TASK [Start Mattermost application within systemd] *****************************
changed: [104.211.28.15]

TASK [Ensure firewalld is installed] *******************************************
ok: [104.211.28.15]

TASK [Ensure firewalld is disabled] ********************************************
changed: [104.211.28.15]

TASK [Ensure firewalld is stopped] *********************************************
changed: [104.211.28.15]

PLAY RECAP *********************************************************************
104.211.28.15              : ok=15   changed=11   unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
localhost                  : ok=3    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

[student@master-vnc-desktop playbooks]$
```

Test the single node Mattermost Application

The Mattermost application should be available for you to test in a single-node configuration. Observing the IP address 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!

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

Initial Startup Screen:


Mattermost - Mozilla Firefox
Mattermost
40.117.112.199:8065/signup_email
< Back

Mattermost

All team communication in one place, searchable and accessible anywhere

Let's create your account

Already have an account? Click here to sign in.

What's your email address?

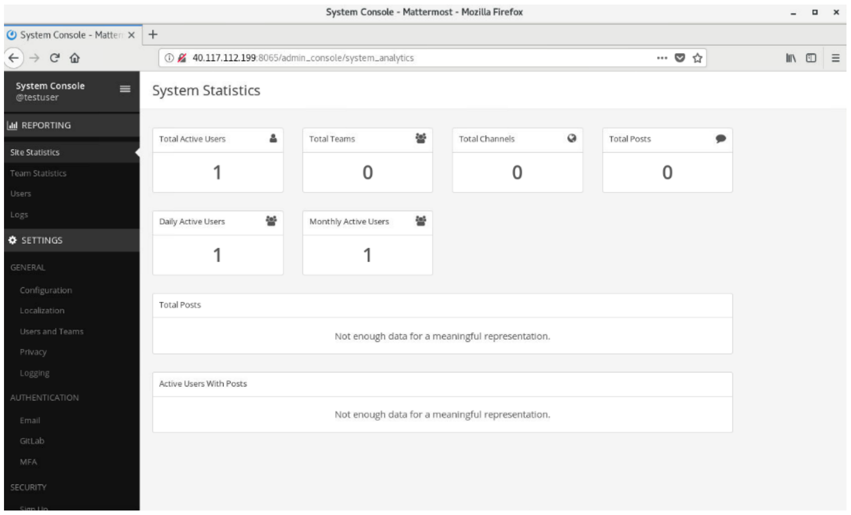
Valid email required for sign-up

Choose your username

Username must begin with a letter, and contain between 3 to 22 lowercase characters made up of numbers, letters, and the symbols '.', '-' and '_'

```
TASK [Ensure firewalld is disabled] *********************************************************************
changed: [40.117.112.199]

TASK [Ensure firewalld is stopped] **********************************************************************
changed: [40.117.112.199]

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

Administrative Console:



System Statistics

Total Active Users	Total Teams	Total Channels	Total Posts
1	0	0	0

Daily Active Users	Monthly Active Users
1	1

Total Posts

Not enough data for a meaningful representation.

Active Users With Posts

Not enough data for a meaningful representation.

Playbook 4 – Generalizing & Creating a VM Disk Image

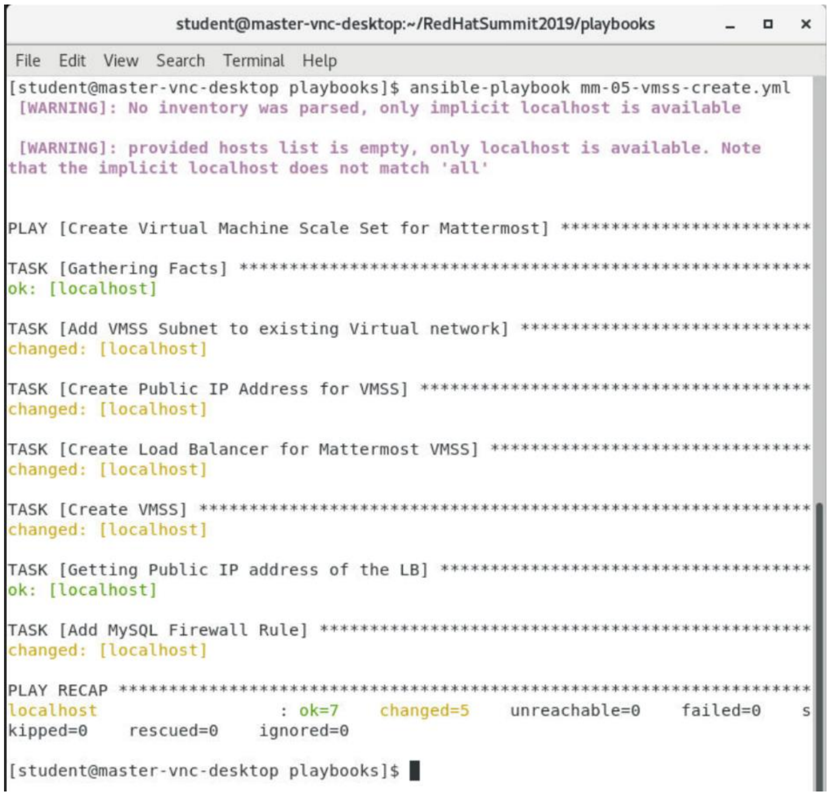
Estimated Playbook Runtime: 1m 8s

➢ `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`



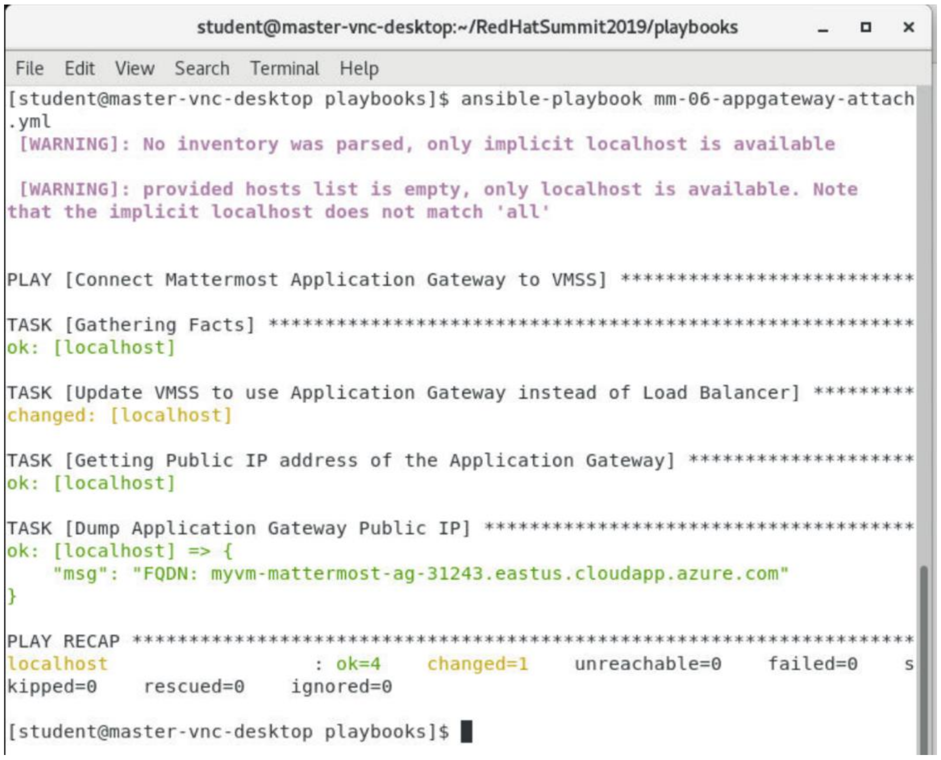
student@master-vnc-desktop:~/RedHatSummit2019/playbooks
File Edit View Search Terminal Help

```
[student@master-vnc-desktop playbooks]$ 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    s
kipped=0    rescued=0    ignored=0

[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`

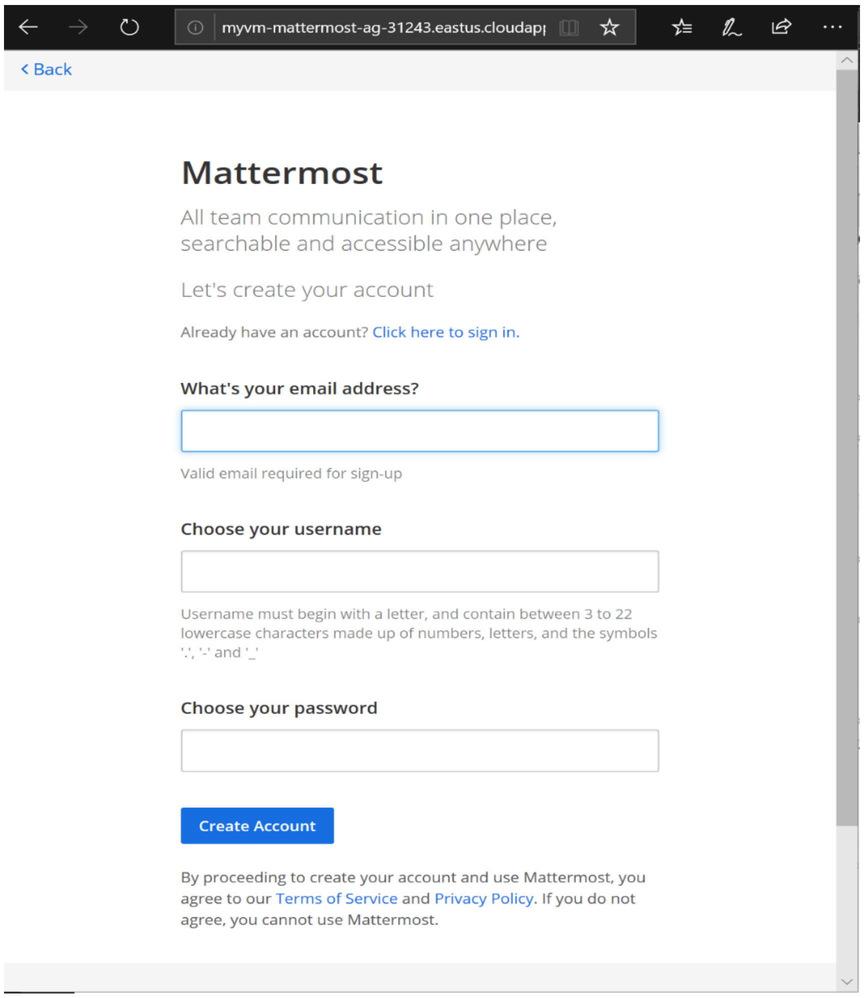

student@master-vnc-desktop:~/RedHatSummit2019/playbooks
File Edit View Search Terminal Help

```
[student@master-vnc-desktop playbooks]$ 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 Public IP address of the Application Gateway] ********************
ok: [localhost]

TASK [Dump Application Gateway Public IP] **************************************
ok: [localhost] => {
    "msg": "FQDN: myvm-mattermost-ag-31243.eastus.cloudapp.azure.com"
}

PLAY RECAP *********************************************************************
localhost                  : ok=4    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

[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/northcentralus.cloudapp.azure.com



Mattermost

All team communication in one place, searchable and accessible anywhere

Let's create your account

Already have an account? Click here to sign in.

What's your email address?

Valid email required for sign-up

Choose your username

Username must begin with a letter, and contain between 3 to 22 lowercase characters made up of numbers, letters, and the symbols '.', '-' and '_'

Choose your password

Create Account

By proceeding to create your account and use Mattermost, you agree to our Terms of Service and Privacy Policy. If you do not agree, you cannot use Mattermost.

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

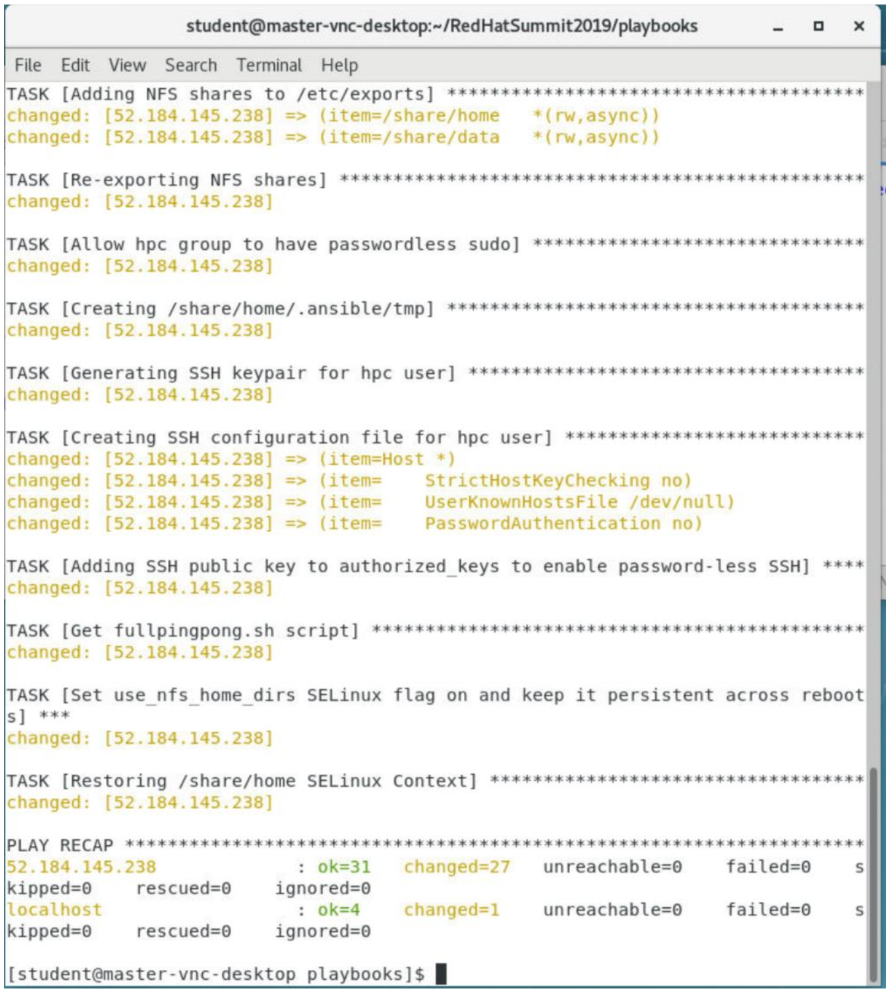
Estimated Playbook Runtime: 16m 25s

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

Playbook 1 – Configure the HPC Cluster Master NFS Share VM

Estimated Playbook Runtime: 4m 40s

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


student@master-vnc-desktop:~/RedHatSummit2019/playbooks
File Edit View Search Terminal Help

```
TASK [Adding NFS shares to /etc/exports] ****************************************
changed: [52.184.145.238] => (item=/share/home    *(rw,async))
changed: [52.184.145.238] => (item=/share/data    *(rw,async))

TASK [Re-exporting NFS shares] **************************************************
changed: [52.184.145.238]

TASK [Allow hpc group to have passwordless sudo] ********************************
changed: [52.184.145.238]

TASK [Creating /share/home/.ansible/tmp] ****************************************
changed: [52.184.145.238]

TASK [Generating SSH keypair for hpc user] **************************************
changed: [52.184.145.238]

TASK [Creating SSH configuration file for hpc user] *****************************
changed: [52.184.145.238] => (item=Host *)
changed: [52.184.145.238] => (item=    StrictHostKeyChecking no)
changed: [52.184.145.238] => (item=    UserKnownHostsFile /dev/null)
changed: [52.184.145.238] => (item=    PasswordAuthentication no)

TASK [Adding SSH public key to authorized_keys to enable password-less SSH] ****
changed: [52.184.145.238]

TASK [Get fullpingpong.sh script] ***********************************************
changed: [52.184.145.238]

TASK [Set use_nfs_home_dirs SELinux flag on and keep it persistent across reboot
s] ***
changed: [52.184.145.238]

TASK [Restoring /share/home SELinux Context] ************************************
changed: [52.184.145.238]

PLAY RECAP **********************************************************************
52.184.145.238             : ok=31   changed=27   unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
localhost                  : ok=4    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

[student@master-vnc-desktop playbooks]$ █
```

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

Estimated Playbook Runtime: 8m 1s

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

Playbook 3 – Configure the Infiniband HPC Worker Nodes

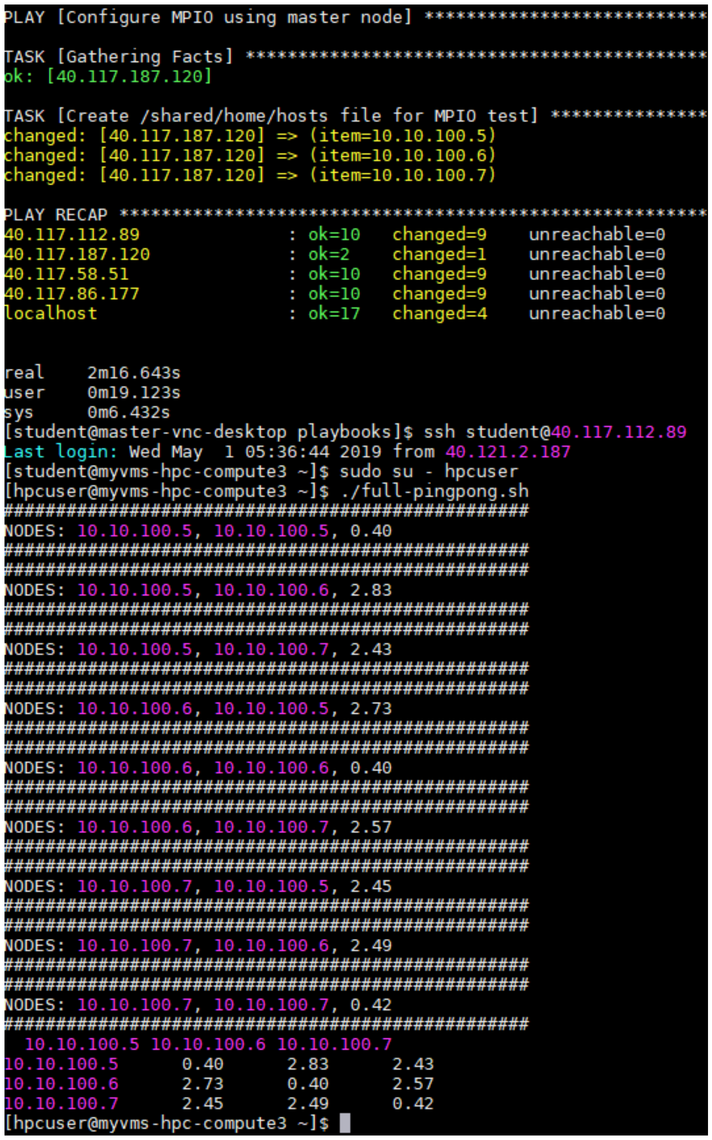
Estimated Playbook Runtime: 2m 37s

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

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`



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

TASK [Gathering Facts] *********************************************
ok: [40.117.187.120]

TASK [Create /shared/home/hosts file for MPIO test] ***************
changed: [40.117.187.120] => (item=10.10.100.5)
changed: [40.117.187.120] => (item=10.10.100.6)
changed: [40.117.187.120] => (item=10.10.100.7)

PLAY RECAP *********************************************************
40.117.112.89              : ok=10   changed=9    unreachable=0
40.117.187.120             : ok=2    changed=1    unreachable=0
40.117.58.51               : ok=10   changed=9    unreachable=0
40.117.86.177              : ok=10   changed=9    unreachable=0
localhost                  : ok=17   changed=4    unreachable=0


real    2m16.643s
user    0m19.123s
sys     0m6.432s
[student@master-vnc-desktop playbooks]$ ssh student@40.117.112.89
Last login: Wed May  1 05:36:44 2019 from 40.121.2.187
[student@myvms-hpc-compute3 ~]$ sudo su - hpcuser
[hpcuser@myvms-hpc-compute3 ~]$ ./full-pingpong.sh
##################################################
NODES: 10.10.100.5, 10.10.100.5, 0.40
##################################################
##################################################
NODES: 10.10.100.5, 10.10.100.6, 2.83
##################################################
##################################################
NODES: 10.10.100.5, 10.10.100.7, 2.43
##################################################
##################################################
NODES: 10.10.100.6, 10.10.100.5, 2.73
##################################################
##################################################
NODES: 10.10.100.6, 10.10.100.6, 0.40
##################################################
##################################################
NODES: 10.10.100.6, 10.10.100.7, 2.57
##################################################
##################################################
NODES: 10.10.100.7, 10.10.100.5, 2.45
##################################################
##################################################
NODES: 10.10.100.7, 10.10.100.6, 2.49
##################################################
##################################################
NODES: 10.10.100.7, 10.10.100.7, 0.42
##################################################
  10.10.100.5 10.10.100.6 10.10.100.7
10.10.100.5      0.40      2.83      2.43
10.10.100.6      2.73      0.40      2.57
10.10.100.7      2.45      2.49      0.42
[hpcuser@myvms-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* RPM

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. In this lab you can either deploy an HDInsight 3.6 or HDInsight 4.0 cluster. 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.

Playbook "36" – Deploy HDInsight 3.6

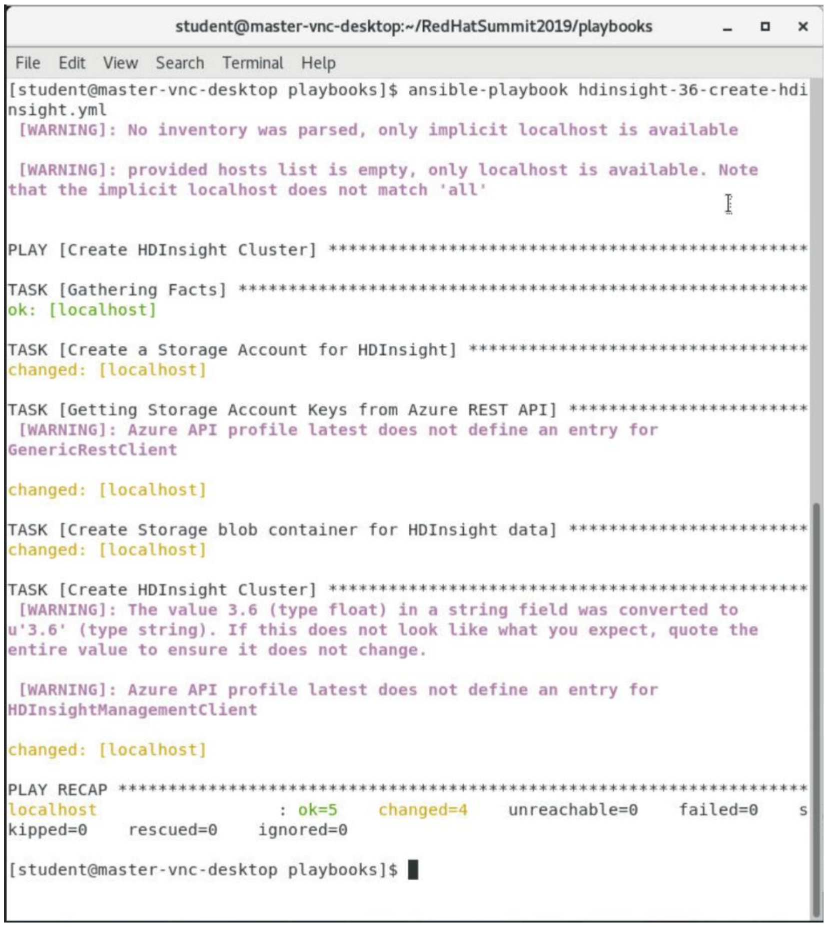
Estimated Playbook Runtime: 18m 19s

- `ansible-playbook hdinsight-36-create-hdinsight.yml`

Playbook "40" – Deploy HDInsight 4.0

Estimated Playbook Runtime: 17m 17s

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


student@master-vnc-desktop:~/RedHatSummit2019/playbooks
File Edit View Search Terminal Help

```
[student@master-vnc-desktop playbooks]$ ansible-playbook hdinsight-36-create-hdi
nsight.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]: The value 3.6 (type float) in a string field was converted to
u'3.6' (type string). If this does not look like what you expect, quote the
entire value to ensure it does not change.

 [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    s
kipped=0    rescued=0    ignored=0

[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 system. An organization can use AKS to deploy, scale and manage containers and container-based applications across a cluster of container hosts. As part of this lab, you will deploy an AKS cluster using Ansible. Using the Azure Linux CLI, you will then merge the K8S configuration into your local ~/.kube directory thus enabling cluster control with standard "kubectl" directives. Finally, you will execute another playbook which will deploy an Azure CosmosDB Database (or use the existing one from Lab 6) and apply a K8S definition file to the AKS cluster. You can then test the application, a small voting booth program, and cast your vote for favorite superhero!

Playbook 0 – Create the Managed AKS Cluster

Estimated Playbook Runtime: 7m 41s

➢ `ansible-playbook aks-00-create-aks-cluster.yml`

Merge the cluster configuration

Remind yourself of your resource group and merge the K8S credentials into your ~/.kube directory

➢ `az aks get-credentials -g <YOUR_RG> -n <YOUR_AKS_CLUSTER_NAME>`

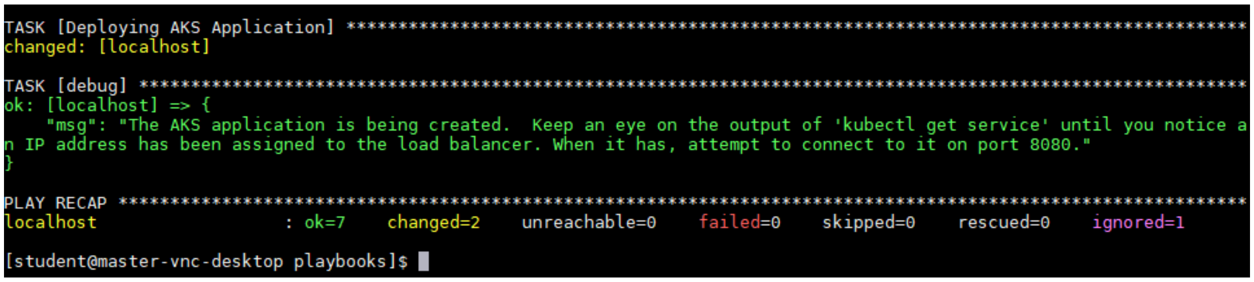
You will now be able to execute standard "kubectl" commands against the AKS cluster.

Playbook 1 – Deploy a NoSQL CosmosDB & the K8S Application

Estimated Playbook Runtime: 4m 41s

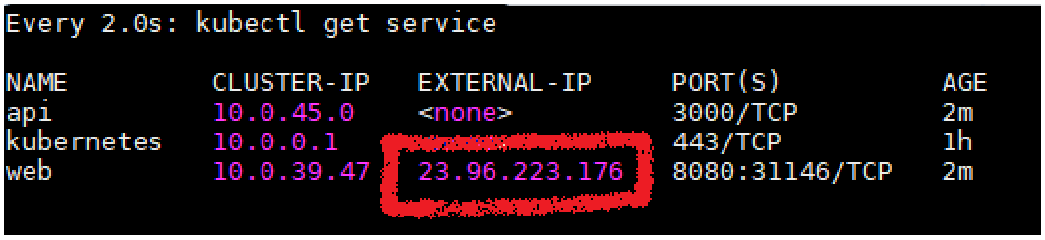
➢ `ansible-playbook aks-01-create-cosmosdb.yml`

After the playbook is complete, you will need to determine the IP address of the K8S load balancer which is in the process of being created. Use the watch command to repeatedly check for this to be available.



```
TASK [Deploying AKS Application] ****************************************************************************************
changed: [localhost]

TASK [debug] ************************************************************************************************************
ok: [localhost] => {
    "msg": "The AKS application is being created.  Keep an eye on the output of 'kubectl get service' until you notice a
n IP address has been assigned to the load balancer. When it has, attempt to connect to it on port 8080."
}

PLAY RECAP **************************************************************************************************************
localhost                  : ok=7    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=1

[student@master-vnc-desktop playbooks]$
```

➢ `watch kubectl get service`



```
Every 2.0s: kubectl get service

NAME         CLUSTER-IP    EXTERNAL-IP     PORT(S)          AGE
api          10.0.45.0     <none>          3000/TCP         2m
kubernetes   10.0.0.1                      443/TCP          1h
web          10.0.39.47    23.96.223.176   8080:31146/TCP   2m
```

Test the Voting 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 service`" using http. Feel free to rate and cast votes as you feel appropriate.

➢ Visit http://x.x.x.x:8080 (in the example http://23.96.223.176:8080)

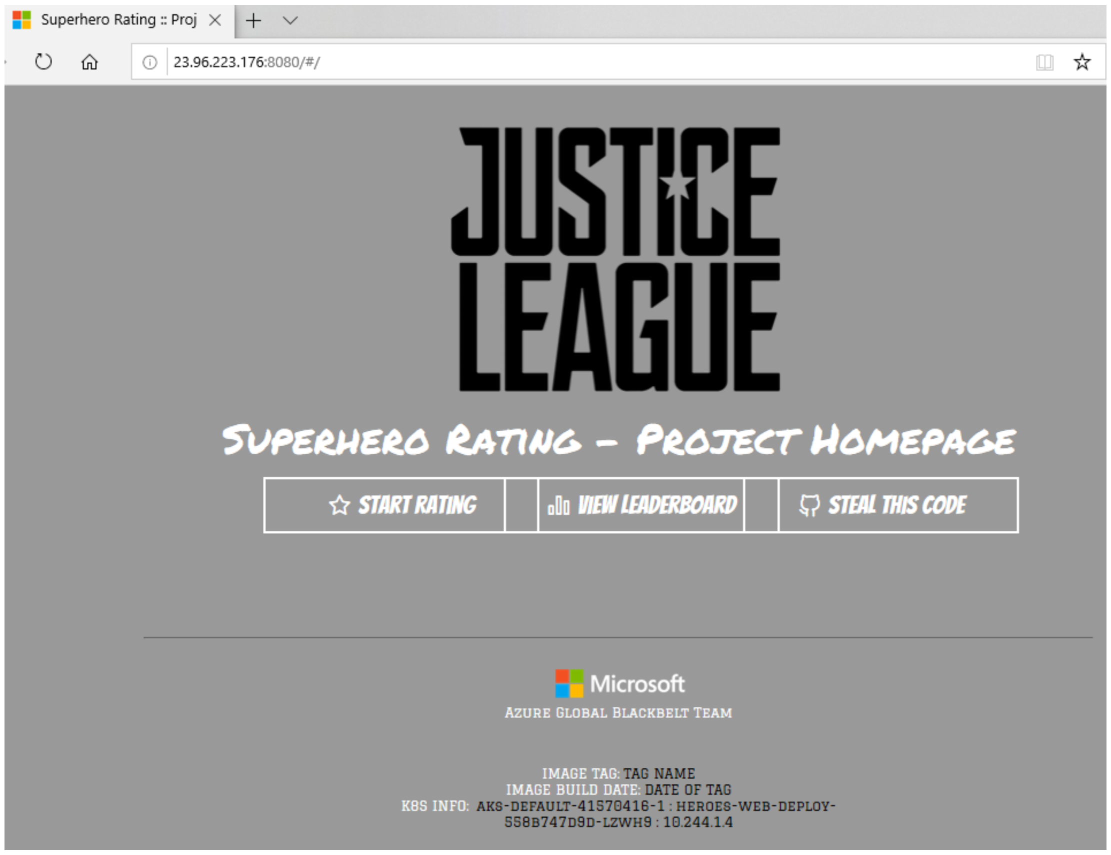


IMAGE TAG: TAG NAME
IMAGE BUILD DATE: DATE OF TAG
K8S INFO: AKS-DEFAULT-41570416-1 : HEROES-WEB-DEPLOY-558B747D9D-LZWH9 : 10.244.1.4

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/RedHatSummit2019

Playbook 0 – Create Azure Container Registry & Image

Estimated Playbook Runtime: 1m 31s

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

ACR Creation Task:

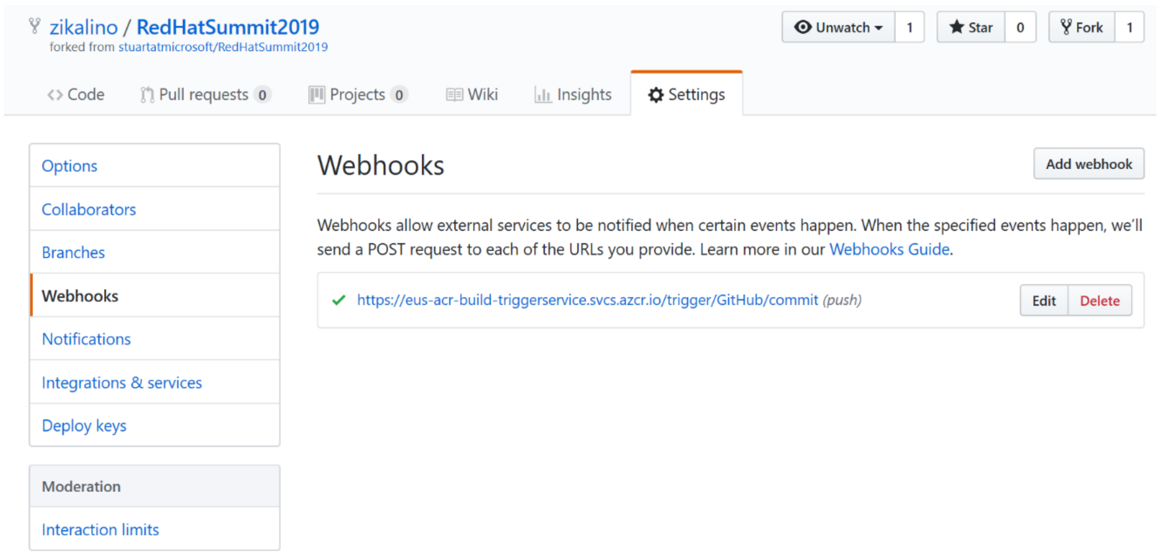
```
    - name: Create container registry
      azure_rm_containerregistry:
        resource_group: "{{ resource_group }}"
        name: "{{ registry_name }}"
        location: eastus
        admin_user_enabled: true
        sku: Premium
```

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
}}/RedHatSummit2019
                    branch: master
                    sourceControlAuthProperties:
                      tokenType: PAT
                      token:  "{{ github_token }}"
                  sourceTriggerEvents:
                    - commit
                  status: Enabled
              baseImageTrigger:
                name: myBaseImageTrigger
                baseImageTriggerType: Runtime
          location: eastus
```

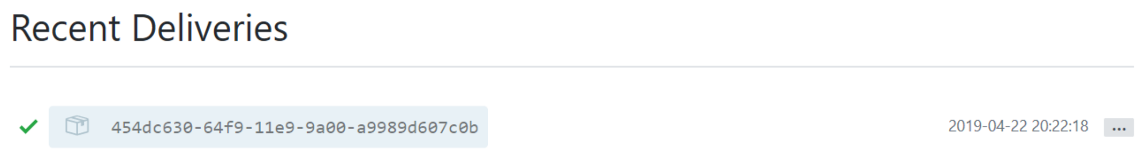
After this playbook completes, verify in your GitHub account that the webhook was created. You can view this by viewing the forked repository in your GitHub account and looking in the "Settings/Webhooks" screen.



Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

✓ https://eus-acr-build-triggerservice.svcs.azcr.io/trigger/GitHub/commit (push)

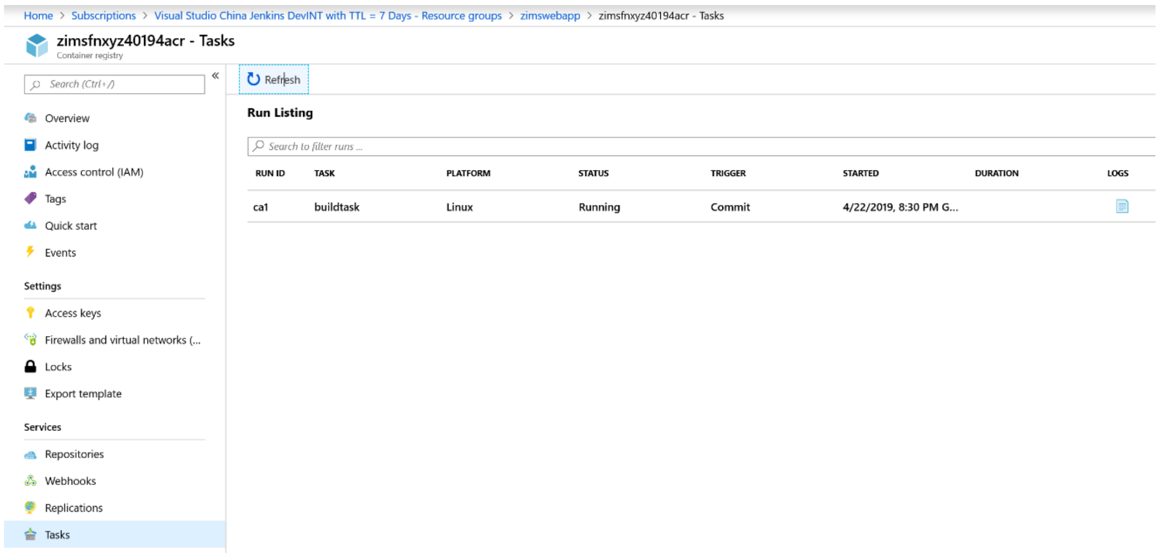
To verify that the webhook is operational, click the "Edit" button to verify that a delivery has been made:



✓ 454dc630-64f9-11e9-9a00-a9989d607c0b 2019-04-22 20:22:18

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:


Home > Subscriptions > Visual Studio China Jenkins DevINT with TTL = 7 Days - Resource groups > zimswebapp > zimsfnxyz40194acr - Tasks
zimsfnxyz40194acr - Tasks
Container registry
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
Quick start
Events
Settings
Access keys
Firewalls and virtual networks (...
Locks
Export template
Services
Repositories
Webhooks
Replications
Tasks
Refresh
Run Listing
Search to filter runs ...

RUN ID	TASK	PLATFORM	STATUS	TRIGGER	STARTED	DURATION	LOGS
ca1	buildtask	Linux	Running	Commit	4/22/2019, 8:30 PM G...		

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
```

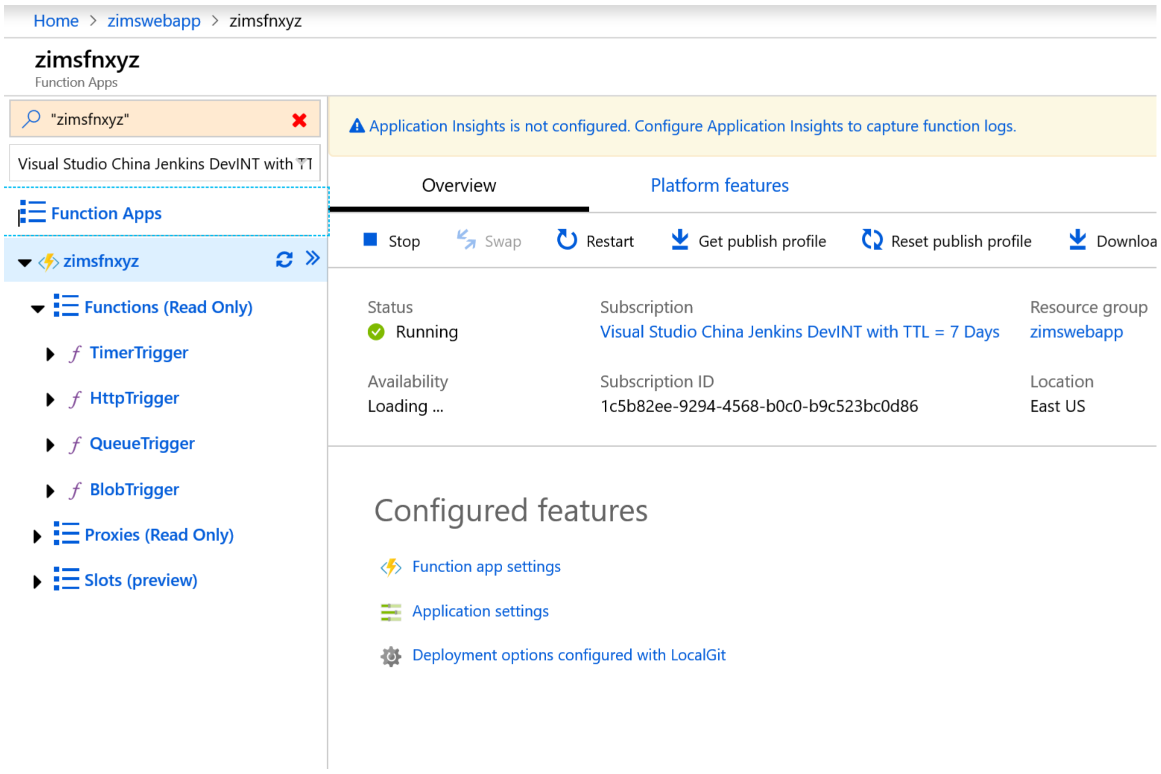
Obtain Azure Container Registry Credentials:

```
    - 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:


Home > zimswebapp > zimsfnxyz

zimsfnxyz
Function Apps

"zimsfnxyz"

Visual Studio China Jenkins DevINT with TT

Function Apps

zimsfnxyz

Functions (Read Only)

TimerTrigger

HttpTrigger

QueueTrigger

BlobTrigger

Proxies (Read Only)

Slots (preview)

Application Insights is not configured. Configure Application Insights to capture function logs.

Overview | Platform features

Stop | Swap | Restart | Get publish profile | Reset publish profile | Downloa

Status	Subscription	Resource group
Running	Visual Studio China Jenkins DevINT with TTL = 7 Days	zimswebapp

Availability	Subscription ID	Location
Loading ...	1c5b82ee-9294-4568-b0c0-b9c523bc0d86	East US

Configured features

Function app settings

Application settings

Deployment options configured with LocalGit

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`

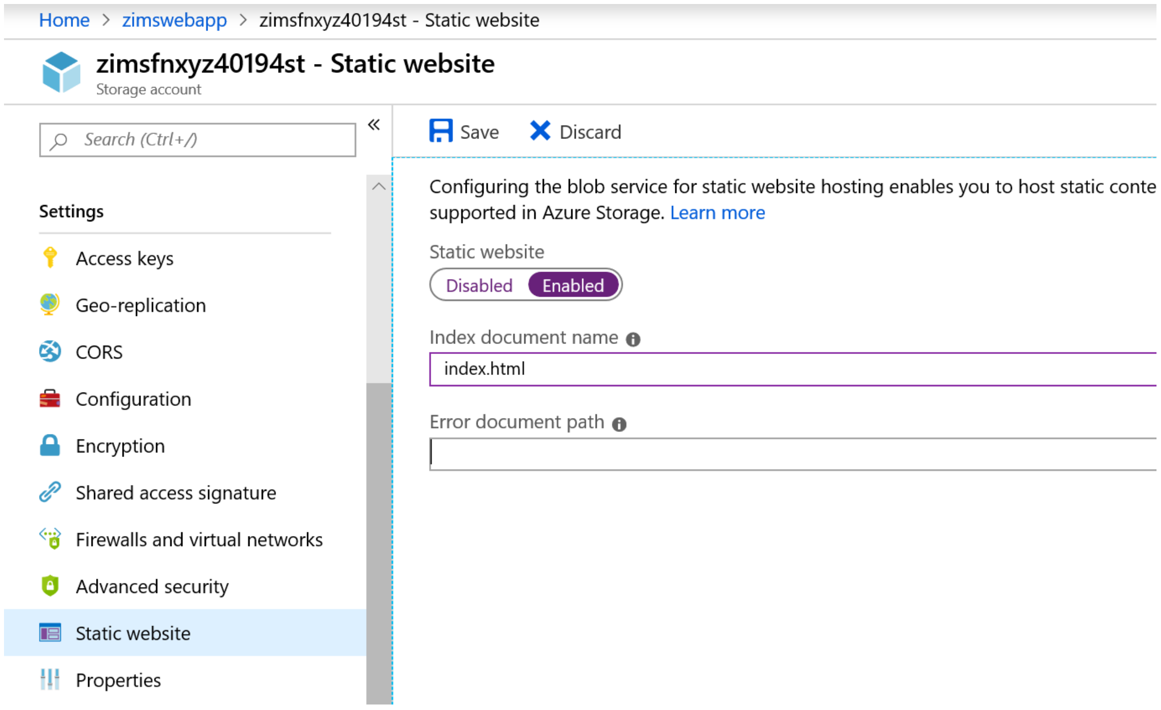
This playbook performs the following tasks:

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"


Home > zimswebapp > zimsfnxyz40194st - Static website

zimsfnxyz40194st - Static website
Storage account

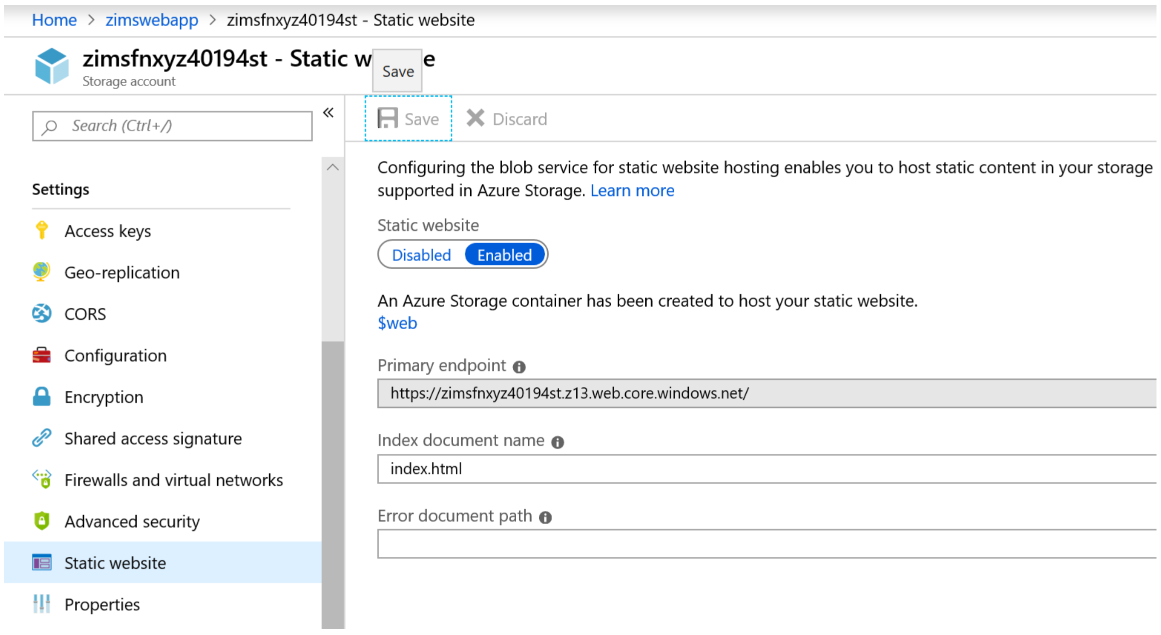
Configuring the blob service for static website hosting enables you to host static conte supported in Azure Storage. Learn more

Static website: Disabled [✓ Enabled]

Index document name: index.html

Error document path:

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



Configuring the blob service for static website hosting enables you to host static content in your storage supported in Azure Storage. Learn more

Static website: Disabled [✓ Enabled]

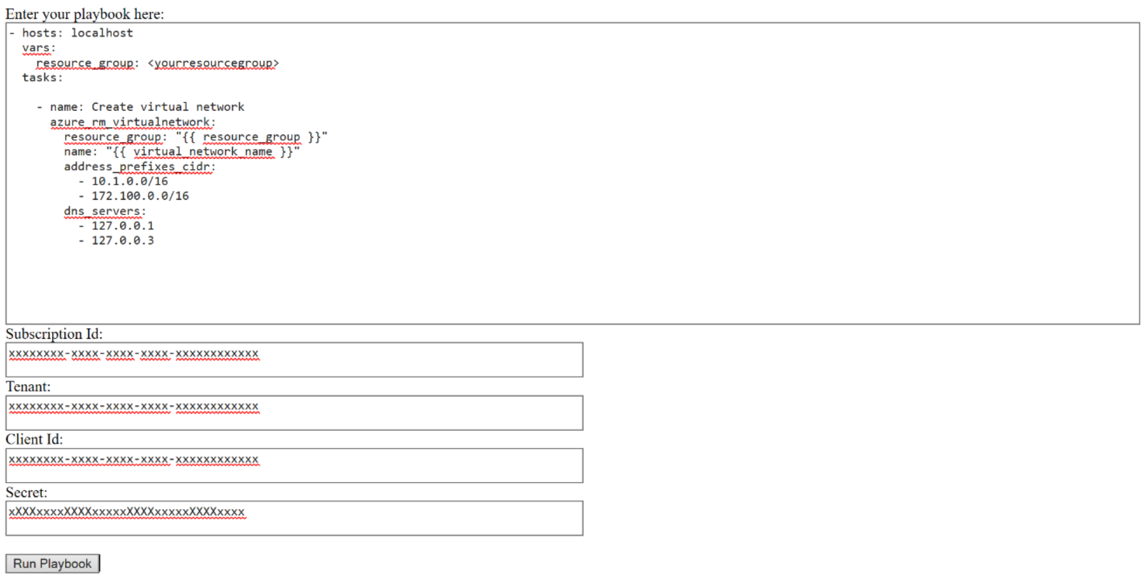
An Azure Storage container has been created to host your static website.
$web

Primary endpoint: https://zimsfnxyz40194st.z13.web.core.windows.net/

Index document name: index.html

Error document path:

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



Enter your playbook here:

```
- hosts: localhost
  vars:
    resource_group: <yourresourcegroup>
  tasks:

    - name: Create virtual network
      azure_rm_virtualnetwork:
        resource_group: "{{ resource_group }}"
        name: "{{ virtual_network_name }}"
        address_prefixes_cidr:
          - 10.1.0.0/16
          - 172.100.0.0/16
        dns_servers:
          - 127.0.0.1
          - 127.0.0.3
```

Subscription Id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Tenant: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Client Id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Secret: xXXXxxxxXXXXxxxxxXXXXxxxxxXXXXxxxx

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

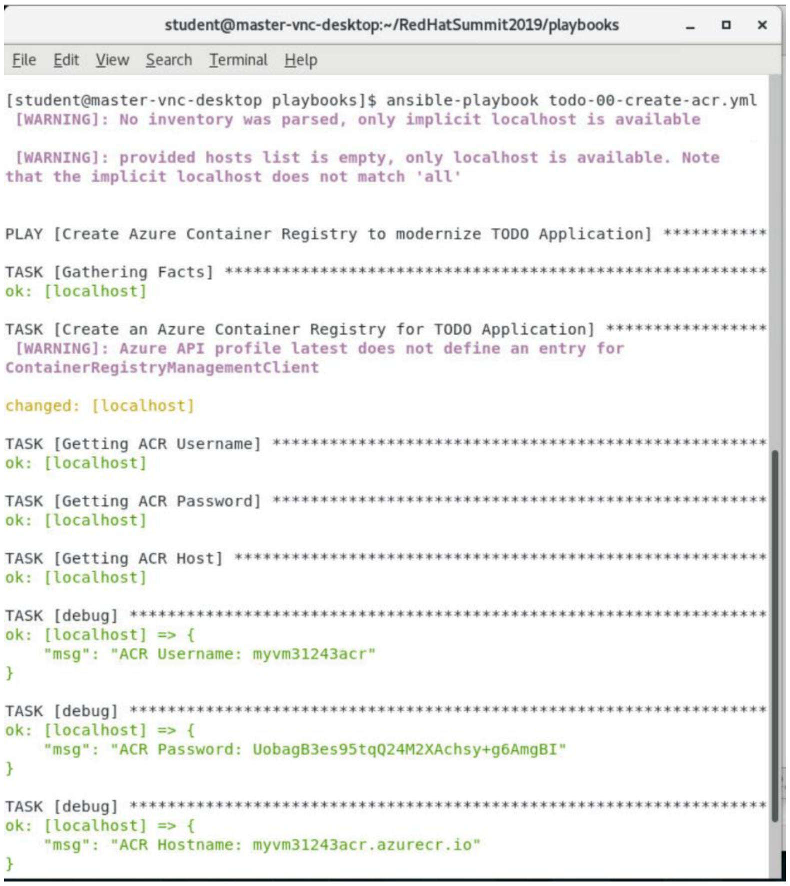
When embracing cloud, it is important to consider not only the migration of applications but also the opportunity to modernize them. This lab will allow you to take an existing NodeJS application currently running on your Lab VM (Try visiting http://localhost) which is backed by a locally running MongoDB and migrate it to Azure. Instead of a straight "Lift & Shift" migration, we will containerize the NodeJS application 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.

Playbook 0 – Create an Azure Container Registry (ACR)

Estimated Playbook Runtime: 0m 7s

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.

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


student@master-vnc-desktop:~/RedHatSummit2019/playbooks

File Edit View Search Terminal Help

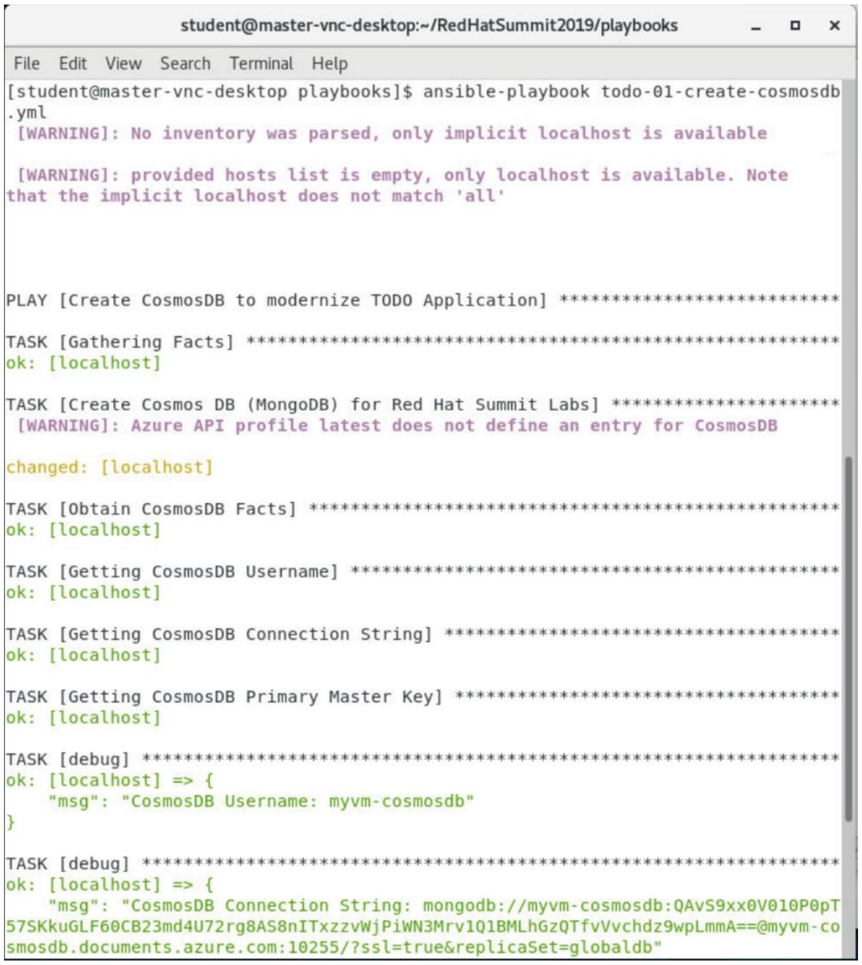
```
[student@master-vnc-desktop playbooks]$ ansible-playbook todo-00-create-acr.yml
 [WARNING]: No inventory was parsed, only implicit localhost is available

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


PLAY [Create Azure Container Registry to modernize TODO Application] ***********

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

TASK [Create an Azure Container Registry for TODO Application] *****************
 [WARNING]: Azure API profile latest does not define an entry for
ContainerRegistryManagementClient

changed: [localhost]

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

TASK [Getting ACR Password] ****************************************************
ok: [localhost]

TASK [Getting ACR Host] ********************************************************
ok: [localhost]

TASK [debug] *******************************************************************
ok: [localhost] => {
    "msg": "ACR Username: myvm31243acr"
}

TASK [debug] *******************************************************************
ok: [localhost] => {
    "msg": "ACR Password: UobagB3es95tqQ24M2XAchsy+g6AmgBI"
}

TASK [debug] *******************************************************************
ok: [localhost] => {
    "msg": "ACR Hostname: myvm31243acr.azurecr.io"
}
```

Playbook 1 – Deploy a NoSQL CosmosDB

Estimated Playbook Runtime: 4m 41s

Create an Azure CosmosDB database (or use the existing one from Lab 4) 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.

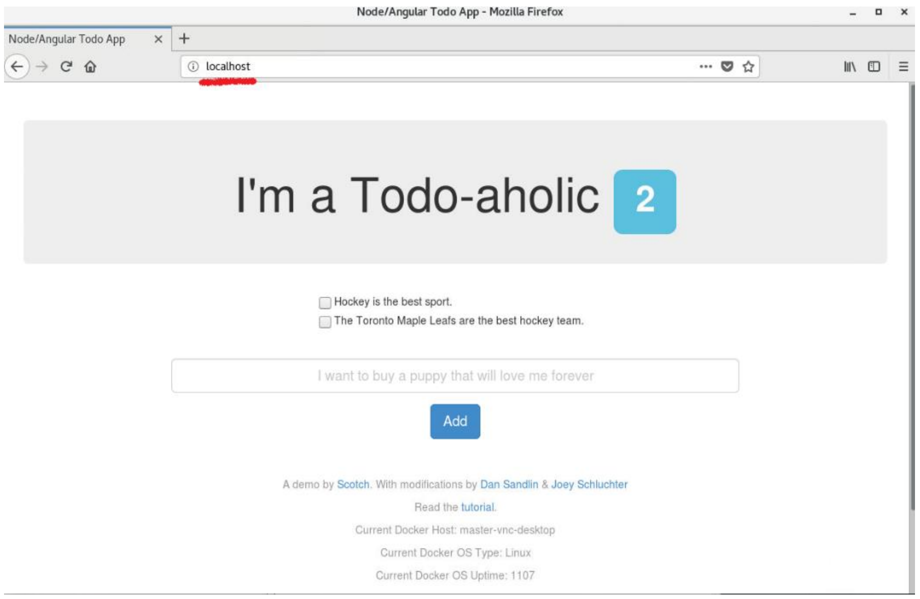
➢ `ansible-playbook todo-01-create-cosmosdb.yml`


student@master-vnc-desktop:~/RedHatSummit2019/playbooks
File Edit View Search Terminal Help

```
[student@master-vnc-desktop playbooks]$ 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: myvm-cosmosdb"
}

TASK [debug] *******************************************************************
ok: [localhost] => {
    "msg": "CosmosDB Connection String: mongodb://myvm-cosmosdb:QAvS9xx0V010P0pT
57SKkuGLF60CB23md4U72rg8AS8nITxzzvWjPiWN3Mrv1Q1BMLhGzQTfvVvchdz9wpLmmA==@myvm-co
smosdb.documents.azure.com:10255/?ssl=true&replicaSet=globaldb"
```

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


Node/Angular Todo App - Mozilla Firefox
Node/Angular Todo App
localhost
I'm a Todo-aholic 2
Hockey is the best sport.
The Toronto Maple Leafs are the best hockey team.
I want to buy a puppy that will love me forever
Add
A demo by Scotch. With modifications by Dan Sandlin & Joey Schluchter
Read the tutorial.
Current Docker Host: master-vnc-desktop
Current Docker OS Type: Linux
Current Docker OS Uptime: 1107

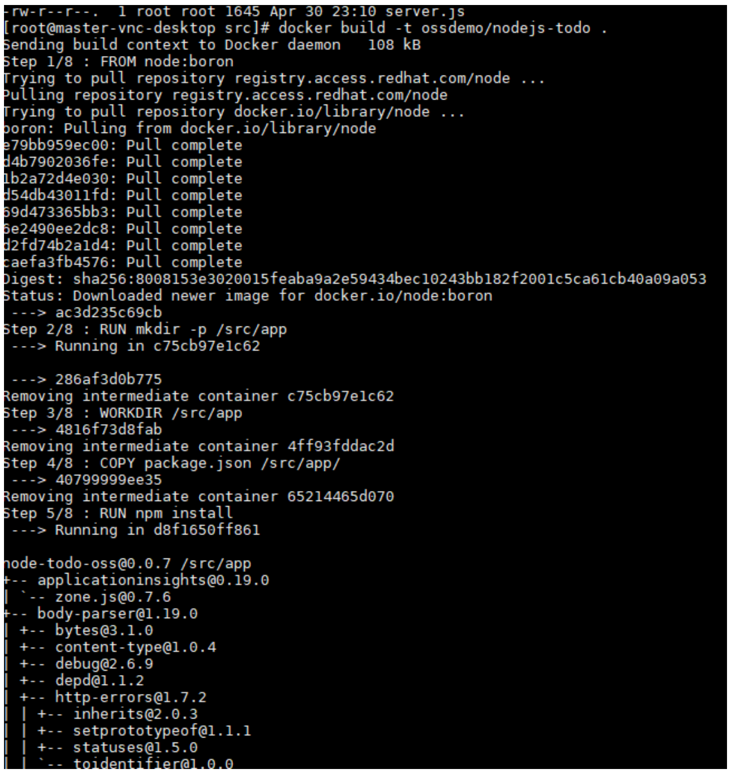
Prepare the Docker Container

Install, start and enable docker:

- `yum -y install docker; systemctl start docker; systemctl enable docker`

Switch into the /source/sample-apps/nodejs-todo/src directory and edit the Dockerfile to expose port 80 instead of 8080. Save the file. Build the Docker container:

- `docker build -t ossdemo/nodejs-todo .`



```
-rw-r--r--.  1 root root 1645 Apr 30 23:10 server.js
[root@master-vnc-desktop src]# docker build -t ossdemo/nodejs-todo .
Sending build context to Docker daemon   108 kB
Step 1/8 : FROM node:boron
Trying to pull repository registry.access.redhat.com/node ...
Pulling repository registry.access.redhat.com/node
Trying to pull repository docker.io/library/node ...
boron: Pulling from docker.io/library/node
e79bb959ec00: Pull complete
d4b7902036fe: Pull complete
1b2a72d4e030: Pull complete
d54db43011fd: Pull complete
69d473365bb3: Pull complete
6e2490ee2dc8: Pull complete
d2fd74b2a1d4: Pull complete
caefa3fb4576: Pull complete
Digest: sha256:8008153e3020015feaba9a2e59434bec10243bb182f2001c5ca61cb40a09a053
Status: Downloaded newer image for docker.io/node:boron
 ---> ac3d235c69cb
Step 2/8 : RUN mkdir -p /src/app
 ---> Running in c75cb97e1c62

 ---> 286af3d0b775
Removing intermediate container c75cb97e1c62
Step 3/8 : WORKDIR /src/app
 ---> 4816f73d8fab
Removing intermediate container 4ff93fddac2d
Step 4/8 : COPY package.json /src/app/
 ---> 40799999ee35
Removing intermediate container 65214465d070
Step 5/8 : RUN npm install
 ---> Running in d8f1650ff861

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
| +-- depd@1.1.2
| +-- http-errors@1.7.2
| | +-- inherits@2.0.3
| | +-- setprototypeof@1.1.1
| | +-- statuses@1.5.0
| | `-- toidentifier@1.0.0
```

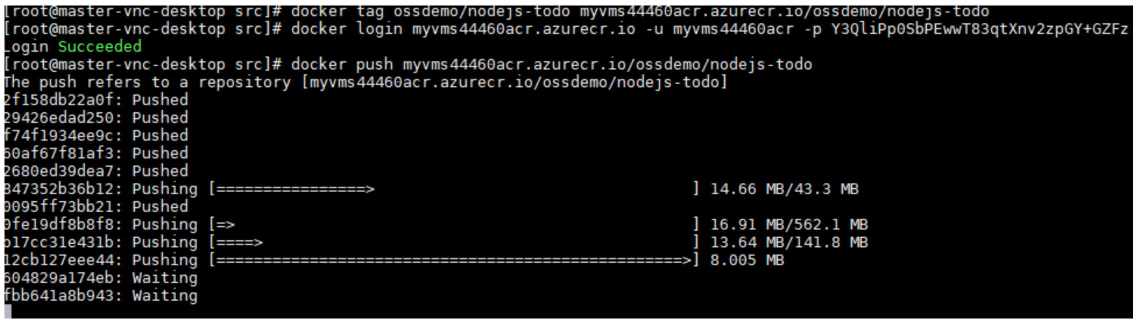
Tag the Docker container using the hostname of your ACR provided by the playbook execution:

- ➢ `docker tag ossdemo/nodejs-todo Xacr.azurecr.io/ossdemo/nodejs-todo`

Push the Docker Container to Azure Container Registry

Login to your ACR and push the container using the username and password of your ACR provided by the playbook:

- ➢ `docker login X.azurecr.io -u USERNAME -p PASSWORD`
- ➢ `docker push X.azurecr.io/ossdemo/nodejs-todo`

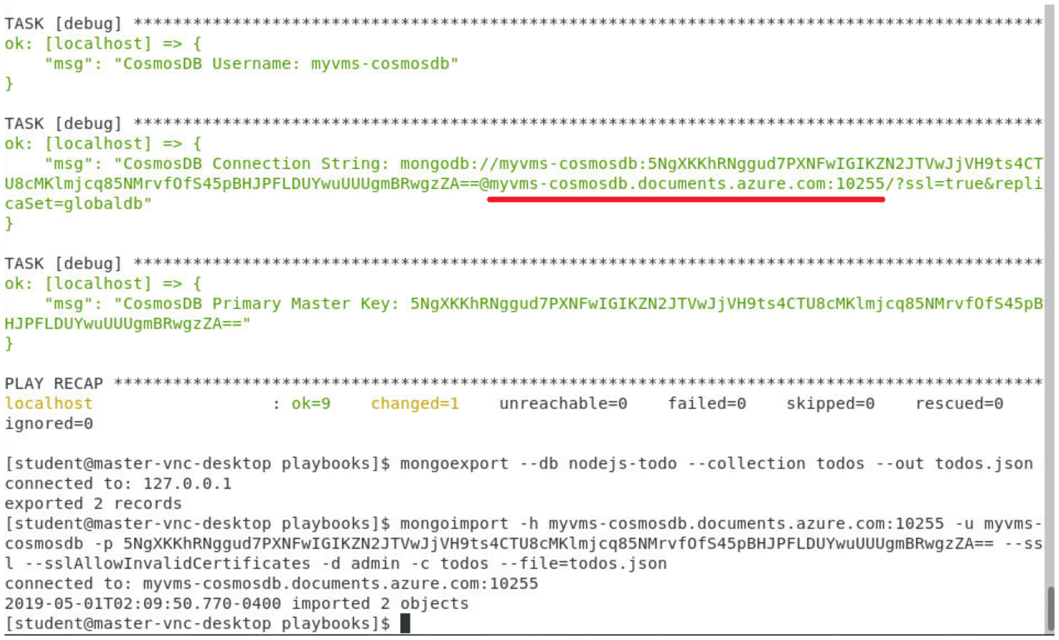


```
[root@master-vnc-desktop src]# docker tag ossdemo/nodejs-todo myvms44460acr.azurecr.io/ossdemo/nodejs-todo
[root@master-vnc-desktop src]# docker login myvms44460acr.azurecr.io -u myvms44460acr -p Y3QliPp0SbPEwwT83qtXnv2zpGY+GZFz
Login Succeeded
[root@master-vnc-desktop src]# docker push myvms44460acr.azurecr.io/ossdemo/nodejs-todo
The push refers to a repository [myvms44460acr.azurecr.io/ossdemo/nodejs-todo]
2f158db22a0f: Pushed
29426edad250: Pushed
f74f1934ee9c: Pushed
60af67f81af3: Pushed
2680ed39dea7: Pushed
847352b36b12: Pushing [================>                                  ] 14.66 MB/43.3 MB
0095ff73bb21: Pushed
0fe19df8b8f8: Pushing [=>                                                 ] 16.91 MB/562.1 MB
b17cc31e431b: Pushing [====>                                              ] 13.64 MB/141.8 MB
12cb127eee44: Pushing [==================================================>] 8.005 MB
604829a174eb: Waiting
fbb641a8b943: Waiting
```

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

TASK [debug] ***********************************************************************************
ok: [localhost] => {
    "msg": "CosmosDB Connection String: mongodb://myvms-cosmosdb:5NgXKKhRNggud7PXNFwIGIKZN2JTVwJjVH9ts4CT
U8cMKlmjcq85NMrvf0fS45pBHJPFLDUYwuUUUgmBRwgzZA==@myvms-cosmosdb.documents.azure.com:10255/?ssl=true&repli
caSet=globaldb"
}

TASK [debug] ***********************************************************************************
ok: [localhost] => {
    "msg": "CosmosDB Primary Master Key: 5NgXKKhRNggud7PXNFwIGIKZN2JTVwJjVH9ts4CTU8cMKlmjcq85NMrvf0fS45pB
HJPFLDUYwuUUUgmBRwgzZA=="
}

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

[student@master-vnc-desktop playbooks]$ mongoexport --db nodejs-todo --collection todos --out todos.json
connected to: 127.0.0.1
exported 2 records
[student@master-vnc-desktop playbooks]$ mongoimport -h myvms-cosmosdb.documents.azure.com:10255 -u myvms-
cosmosdb -p 5NgXKKhRNggud7PXNFwIGIKZN2JTVwJjVH9ts4CTU8cMKlmjcq85NMrvf0fS45pBHJPFLDUYwuUUUgmBRwgzZA== --ss
l --sslAllowInvalidCertificates -d admin -c todos --file=todos.json
connected to: myvms-cosmosdb.documents.azure.com:10255
2019-05-01T02:09:50.770-0400 imported 2 objects
[student@master-vnc-desktop playbooks]$ 
```

Playbook 2 – Create an Azure Application Service Plan

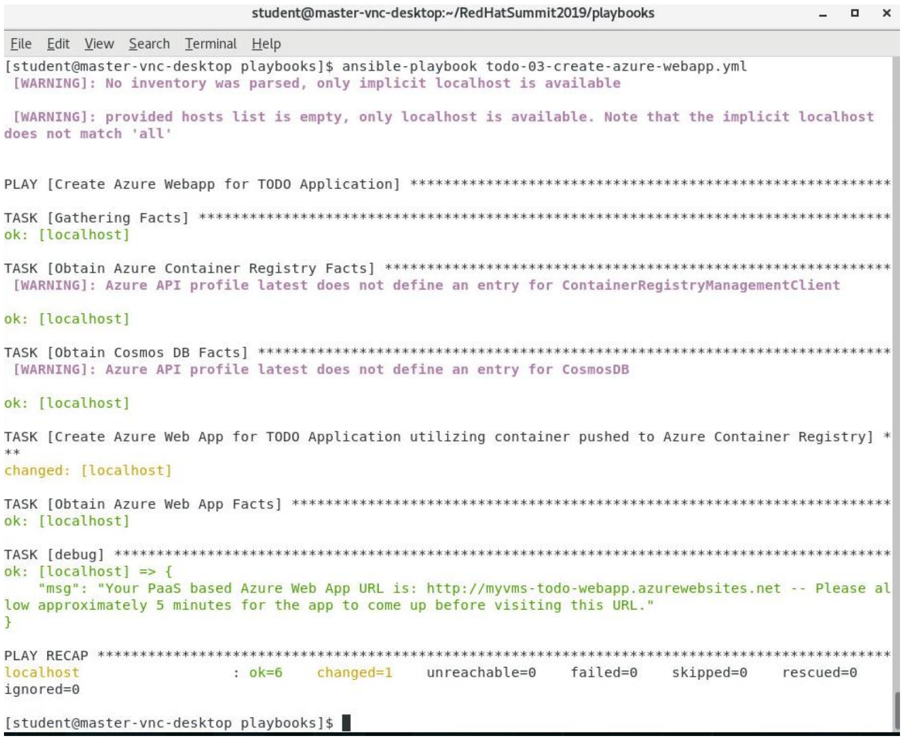
Estimated Playbook Runtime: 0m 14s

- `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`

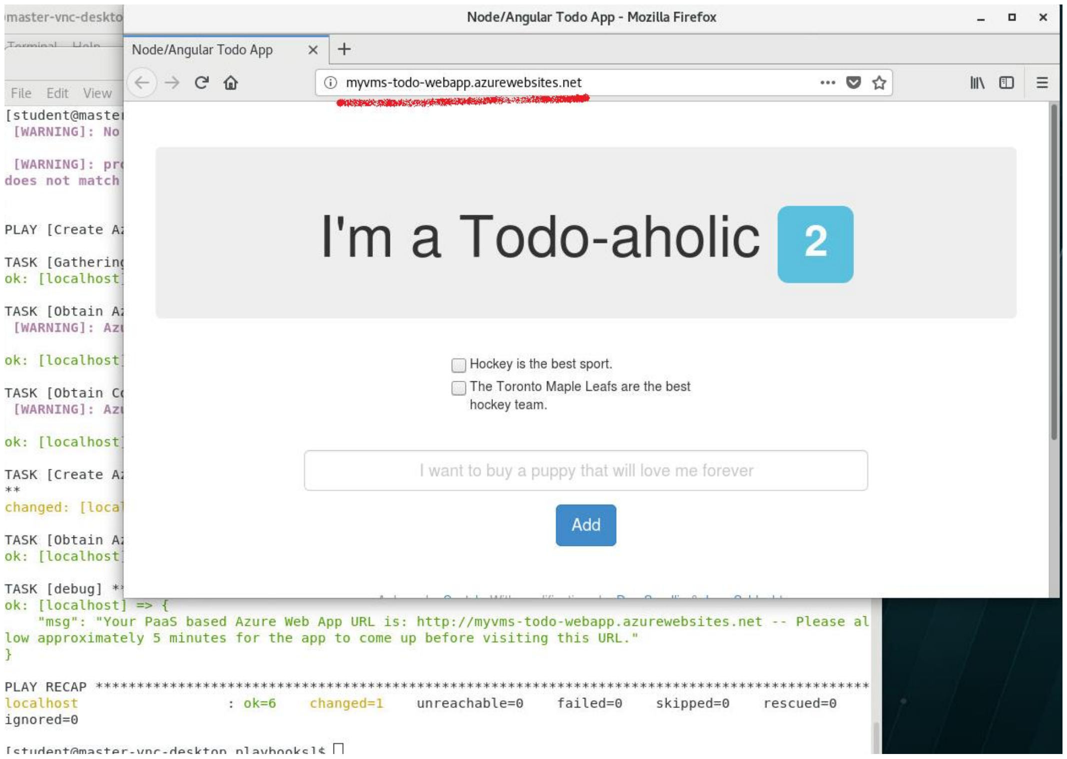

student@master-vnc-desktop:~/RedHatSummit2019/playbooks
File Edit View Search Terminal Help

```
[student@master-vnc-desktop playbooks]$ 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://myvms-todo-webapp.azurewebsites.net -- Please al
low 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

[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



```
[student@maste
 [WARNING]: No

 [WARNING]: pr
does not match

PLAY [Create A

TASK [Gathering
ok: [localhost]

TASK [Obtain A
 [WARNING]: Az

ok: [localhost]

TASK [Obtain C
 [WARNING]: Az

ok: [localhost]

TASK [Create A
**
changed: [loca

TASK [Obtain A
ok: [localhost]

TASK [debug] **
ok: [localhost] => {
    "msg": "Your PaaS based Azure Web App URL is: http://myvms-todo-webapp.azurewebsites.net -- Please al
low 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

[student@master-vnc-desktop playbooks]$
```

Node/Angular Todo App - Mozilla Firefox

myvms-todo-webapp.azurewebsites.net

I'm a Todo-aholic 2

- [ ] Hockey is the best sport.
- [ ] The Toronto Maple Leafs are the best hockey team.

You've reached the end!

Thank you for your participation!

Please take the survey for this lab!