



OpenShift Container Platform 4.4 on AWS with Microsoft Windows Server 2019/1809 Nodes

Developer Preview

Pre-requisites

The document assumes that a Fedora Linux host was used for running all the Linux commands.

Please note that a command preceded by `>` is to be run in a PowerShell window on a Windows instance, and a command preceded by `$` is to be run on a Linux console (localhost)

Python and pip installation

Python3 and pip are required to follow this guide. To check if python3 is installed on the system, run the following command:

```
$ python3 --version
```

To install python3, run the following command:

```
$ sudo dnf install python3
```

To check if pip is installed on the system, run the following command:

```
$ python3 -m pip --version
```

```
pip 19.0.3 from /usr/lib/python3.7/site-packages/pip (python 3.7)
```

To install pip refer [pip installation](#)



Bring up the OpenShift cluster with ovn-kubernetes

Download the latest dev-preview 4.4 [OpenShift Installer and Client](#) and [pull secret for AWS](#).

Note: Download both the openshift-install and openshift-client.

Unzip the files with the following commands: (Reference commands given below)

```
$ tar -xzf openshift-install-linux-<version>.tar.gz
$ tar -xzf openshift-client-linux-<version>.tar.gz
```

After extracting openshift client, move the **oc** and **kubectl** binaries to `/usr/local/bin`

```
$ sudo mv {kubectl,oc} /usr/local/bin
```

Configure AWS CLI

See [Configuring the AWS CLI](#)

Create the install-config

```
$ ./openshift-install create install-config --dir=<cluster_directory>
> SSH Public Key <path-to-your-rsa>/id_rsa.pub
> Platform <i.e. aws>
> Region <region close by i.e. us-east-1>
> Base Domain <Your Domain>
> Cluster Name <cluster_name>
> Pull Secret <content of pull-secrets>
```

[Official documentation](#) should be consulted for credentials and other cloud provider related instructions.

The previous step will result in an `install-config.yaml` in your cluster directory. Edit the `install-config.yaml` to switch networkType from OpenShiftSDN to OVNKubernetes inside the cluster directory:

```
$ sed -i 's/OpenShiftSDN/OVNKubernetes/g' install-config.yaml
```

Create manifests

We want to enable hybrid networking on the cluster. For that we need to create the manifests:

```
$ ./openshift-install create manifests --dir=<cluster_directory>
```

Above step will create a **manifests** and **openshift** folder in your `<cluster_directory>`



Configuring OVNKubernetes on a Hybrid cluster

Inside the `cluster_directory`, create a copy of `manifests/cluster-network-02-config.yml` as `manifests/cluster-network-03-config.yml`.

```
$ cp manifests/cluster-network-02-config.yml manifests/cluster-network-03-config.yml
```

Edit `manifests/cluster-network-03-config.yml` as shown below:

1. Modify the `api version` to `operator.openshift.io/v1`
2. Add the following to the ``spec:`` section of `manifests/cluster-network-03-config.yml`

```
spec:
  defaultNetwork:
    type: OVNKubernetes
    ovnKubernetesConfig:
      hybridOverlayConfig:
        hybridClusterNetwork:
          - cidr: 10.132.0.0/14
            hostPrefix: 23
```

Here is an example of `manifests/cluster-network-03-config.yml`

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  creationTimestamp: null
  name: cluster
spec:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  externalIP:
    policy: {}
  networkType: OVNKubernetes
  serviceNetwork:
    - 172.30.0.0/16
  defaultNetwork:
    type: OVNKubernetes
    ovnKubernetesConfig:
      hybridOverlayConfig:
        hybridClusterNetwork:
          - cidr: 10.132.0.0/14
            hostPrefix: 23
status: {}
```

The highlighted portion above is the required modification for the file.

NOTE: the `hybridClusterNetwork` CIDR cannot overlap `clusterNetwork` CIDR



Create the cluster

Execute the following command to create the cluster and wait for it come up:

```
$ ./openshift-install create cluster --dir=<cluster_directory>
```

Export the kubeconfig so that oc can communicate with the cluster:

```
$ export KUBECONFIG=$(pwd)/<cluster_directory>/auth/kubeconfig
```

Make sure you can interact with the cluster:

```
$ oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-133-44.ec2.internal	Ready	worker	4h5m	v1.17.1
ip-10-0-142-29.ec2.internal	Ready	master	4h14m	v1.17.1
ip-10-0-152-168.ec2.internal	Ready	master	4h14m	v1.17.1
ip-10-0-153-185.ec2.internal	Ready	worker	4h5m	v1.17.1
ip-10-0-164-237.ec2.internal	Ready	worker	4h5m	v1.17.1
ip-10-0-174-28.ec2.internal	Ready	master	4h13m	v1.17.1

Verify Hybrid networking

The network.operator cluster CR spec should look like the example below:

```
$ oc get network.operator cluster -o yaml
```

```
...
spec:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  defaultNetwork:
    ovnKubernetesConfig:
      hybridOverlayConfig:
        hybridClusterNetwork:
        - cidr: 10.132.0.0/14
          hostPrefix: 23
        type: OVNKubernetes
    serviceNetwork:
    - 172.30.0.0/16
  status: {}
...
```



Bring up the Windows node

Launch a *Windows 2019 Server Datacenter with Containers* instance and add it to the cluster.

Note: The [Windows Node Installer](#) tool can be used to create the Windows instance instead, however it is primarily for developer use only and is not supported.

Using AWS console

Login into the [AWS console](#). Make sure you are in the same region as your cluster and click on **Running Instances**

Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

394 Running Instances

Search for the name of your cluster to see the masters and workers running.

Select Launch instance to create a new instance:

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Choose the image for **Microsoft Windows Server 2019 Base with Containers**

**Microsoft Windows Server 2019 Base with Containers** - ami-03fbd9bcb97e9cabb

Windows Microsoft Windows 2019 Datacenter edition with Containers. [English]

Free tier eligible

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select **m5a.large** Instance Type:

Note: We need an AMD based instance

Step 2: Choose an Instance Type

<input checked="" type="checkbox"/>	General purpose	m5a.large	2	8	EBS only	Yes	Up to 10 Gigabit
-------------------------------------	-----------------	-----------	---	---	----------	-----	------------------



Hit **Next: Configure Instance Details** button to Configure the details

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Configure Instance Details](#)

Select the same vpc as your openshift cluster: (match vpc id or cluster name)

Network ⓘ [Create new VPC](#)
No default VPC found. [Create a new default VPC.](#)

Make sure you have selected a **public subnet** or you will not be able to RDP to the instance.

Subnet ⓘ [Create new subnet](#)

Auto-assign Public IP ⓘ

Enable auto-assigning of the Public IP

Auto-assign Public IP ⓘ

In the **Tags** section, the cluster tag needs to be added with the value as owned. You can also name your instance by adding the tag for the same.

Cluster tag is something like **kubernetes.io/cluster/<cluster_name>** and can be verified in the Tags section of any node in the cluster

sumehta-winc-cluster1-fhb8-master-0 owned i-0dd78179e392c4af1 m4.xlarge us-east-1a running 2/2 checks ... None

Instance: **i-0dd78179e392c4af1** (sumehta-winc-cluster1-fhb8-master-0) Private IP: 10.0.135.94

Description Status Checks Monitoring **Tags**

[Add/Edit Tags](#)

Key	Value	
Name	sumehta-winc-cluster1-fhb8-master-0	Hide Column
kubernetes.io/cluster/sumehta-winc-cluster1-fhb8	owned	Hide Column
openshift_creationDate	2019-10-22T15:05:05.041046+00:00	Show Column



The resultant tags looks like the following:

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances ⓘ	Volumes ⓘ	
Name	sumehta-winc-cluster1-winnode	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="X"/>
kubernetes.io/cluster/sumehta-winc-cluster1-fhbk8	owned	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="X"/>

Go to the next step *Configure Security Group*.

Select **MyIP** under source for **RDP** Rule and add **All traffic** rule and change source to **Custom 10.0.0.0/16** for allowing cluster communication. Open **WinRM-HTTPS** port for remote management, enabled for **MyIP**. The configurations should look like:

Edit inbound rules

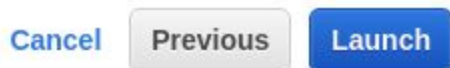
Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ	Description ⓘ	
All TCP ▾	TCP	0 - 65535	Custom ▾ 10.0.0.0/16	kubes	<input type="button" value="X"/>
WinRM-HT ▾	TCP	5986	Custom ▾ 144.121.20.162/32	ansible	<input type="button" value="X"/>
RDP ▾	TCP	3389	Custom ▾ 144.121.20.162/32	rdp	<input type="button" value="X"/>

Add Rule

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Cancel Save

Then hit the **Launch** Button in the review page.





It will then show a prompt window to ask for the key pair.

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair

Select a key pair

default

☒ I acknowledge that I have access to the selected private key file (default.pem), and that without this file, I won't be able to log into my instance.

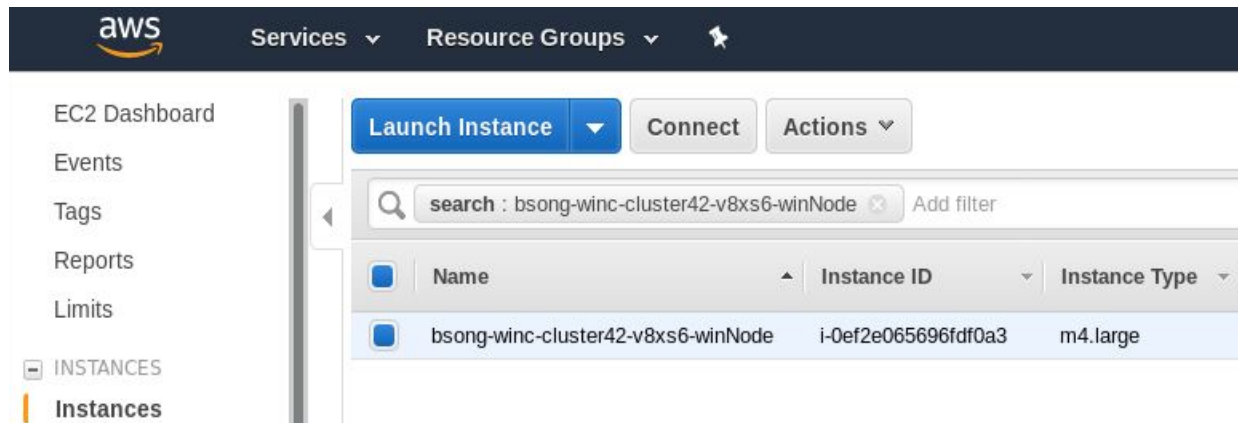
CancelLaunch Instances

Select the key and launch the instance.

If you already have one key pair on AWS, you can select the one you have. Or you can generate a new key pair by giving it a name and store the *.pem file on your local machine by hitting the Download key Pair button. This is for you to connect to your instance later.

Note: If you are unable to launch an instance due to unavailability of instance type within a zone, simply repeat the process, but choose a different public subnet.

Navigate to your aws **ec2 instance** through web browser and find your **instance**
<https://console.aws.amazon.com/ec2/>



Attach cluster worker **IAM** and **Security Group** to windows node:

- 1) Select windows node instance, choose **Actions > Instance Settings > Attach/Replace IAM role**.
- 2) Select the IAM role *for your cluster's worker profile* to attach to your instance, and choose **Apply**.

Attach/Replace IAM Role

Select an IAM role to attach to your instance. If you don't have any IAM roles, choose Create new IAM role to create a role in the IAM console. If an IAM role is already attached to your instance, the IAM role you choose will replace the existing role.

Instance ID i-0ef2e065696fdf0a3 (bsong-winc-cluster42-v8xs6-winNode) ⓘ

IAM role* ⓘ [Create new IAM role](#) ⓘ

- 3) Select a openshift cluster **worker node** and check the id of cluster **security group**

Security groups	terraform-20190820211544288200000003 view inbound rules view outbound rules
Scheduled events	No scheduled events
AMI ID	rhcos-42.80.20190725.1-hvm (ami-089b38aa83694cdcd)
Platform	-
IAM role	bsong-winc-cluster42-v8xs6-worker-role



4) Again select windows node instance and select **Actions > Networking > Change Security Groups**

Change Security Groups

Instance ID:i-0ef2e065696fdf0a3

Interface ID:eni-0d8404764ac6b0656

Select Security Group(s) to associate with your instance

	Security Group ID	Security Group Name	Description
<input type="checkbox"/>	sg-0c6f0f0273f51e0a3	default	default VPC security group
<input type="checkbox"/>	sg-0b874aac22c56887e	k8s-elb-a957623a1c39211e9b1ac02...	Security group for Kubernetes ELB a957623a1c39211e9b1a...
<input checked="" type="checkbox"/>	sg-0d4bd58158fbd770e	launch-wizard-3	launch-wizard-3 created 2019-08-21T12:08:07.334-04:00
<input type="checkbox"/>	sg-015d582d0b90940e3	terraform-201908202115434596000...	Managed by Terraform
<input checked="" type="checkbox"/>	sg-00af21a85bb99f590	terraform-201908202115442882000...	Managed by Terraform

Cancel

Assign Security Groups

Now you can [RDP](#) into the node, [setup your Windows instance](#) and [setup Ansible](#). You can skip the [Windows Node Installer](#) section as that section deals with automating the instance creation.

RDP Setup

You can now RDP into the instance
Hit the **Connect** button.





Click on the [Download Remote Desktop File](#) to download the remote desktop application <filename>.rdp:

Connect To Your Instance

You can connect to your Windows instance using a remote desktop client of your choice, and by downloading and running the RDP shortcut file below:

[Download Remote Desktop File](#)

When prompted, connect to your instance using the following details:

Public DNS	ec2-18-232-61-30.compute-1.amazonaws.com
User name	Administrator
Password	Get Password

If you've joined your instance to a directory, you can use your directory credentials to connect to your instance.

If you need any assistance connecting to your instance, please see our [connection documentation](#).

[Close](#)

Then click on the [Get Password](#) button and select your secret.pem file (private key of the key-pair you provided to launch the instance) or paste its content to [decrypt the password](#) and use it for the next step:

Connect to your instance > Get Password

Connection method ☒ A standalone RDP client [i](#)
☐ Session Manager [i](#)

The following Key Pair was associated with this instance when it was created.

Key Name pmahajan.pem

In order to retrieve your password you will need to specify the path of this Key Pair on your local machine:

Key Pair Path [Choose File](#) pmahajan.pem

Or you can copy and paste the contents of the Key Pair below:

```
-----BEGIN RSA PRIVATE KEY-----  
YOUR KEY HERE  
-----END RSA PRIVATE KEY-----
```

[Decrypt Password](#)



Install a RDP tool:

For Fedora, we use freerdp:

```
$ sudo dnf install freerdp
```

RDP into the windows node:

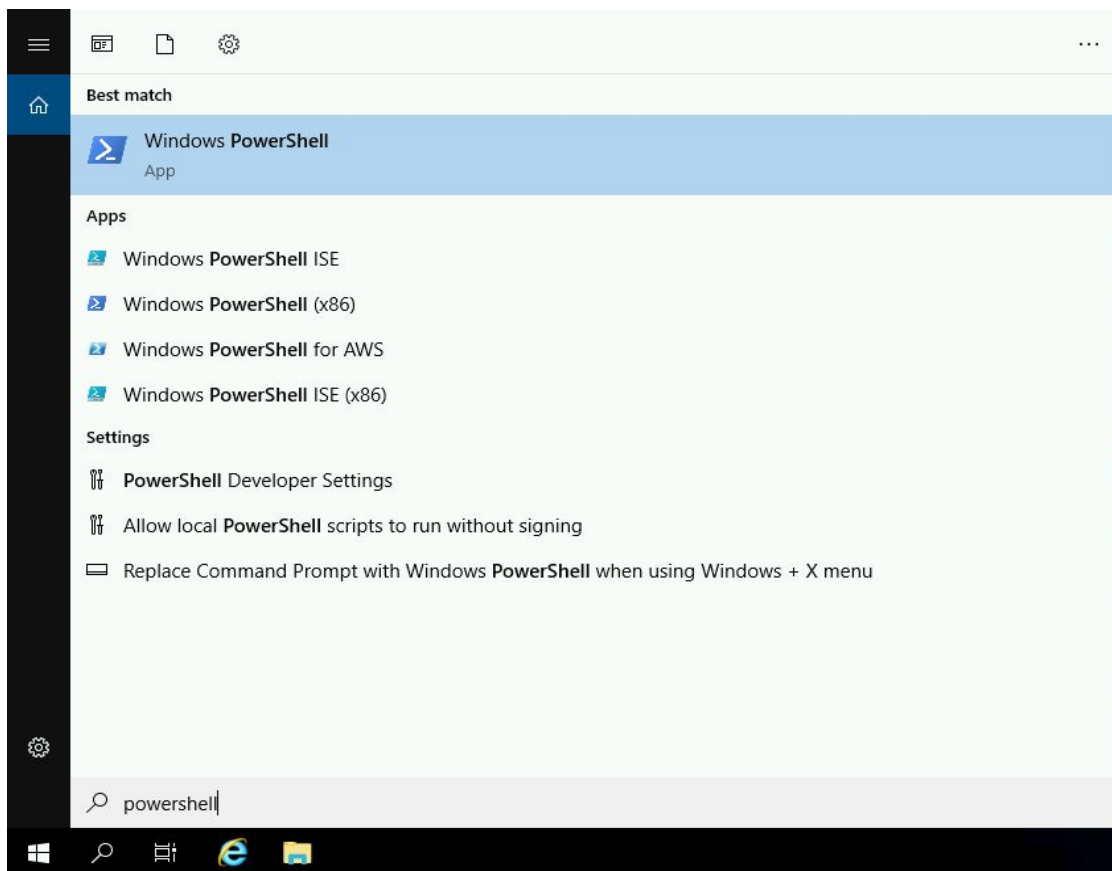
```
$ xfreerdp <path/to/rdpfile/filename.rdp> /p:'<password>' /f
Or
$ xfreerdp /u:Administrator /v:<Public DNS i.e.
ec2-3-80-190-235.compute-1.amazonaws.com> /h:1080 /w:1920 /p:'<password>'
```

You can now see a Windows screen come up.

Setup on Windows instance

Enable WinRM

Bring up powershell and execute the following commands:





```
> $url =  
"https://raw.githubusercontent.com/ansible/ansible/devel/examples/scripts/ConfigureRemotingForAnsible.ps1"  
$file = "$env:temp\ConfigureRemotingForAnsible.ps1"  
(New-Object -TypeName System.Net.WebClient).DownloadFile($url, $file)  
powershell.exe -ExecutionPolicy ByPass -File $file
```

Enable Console logs

Open TCP port 10250 in the Windows Firewall so that logs can be viewed in console and using *oc logs*. On your Windows instance, execute the following in a powershell window:

```
> New-NetFirewallRule -DisplayName "Enable console logs" -Direction Inbound -Action  
Allow -Protocol TCP -LocalPort 10250 -EdgeTraversalPolicy Allow
```

Setup Ansible connection

Now we can use Ansible to configure the Windows host. On the Linux host, install ansible and pywinrm, as well as selinux-python bindings:

Note: This step assumes that python3 is installed on the system. Ansible will not work without python.

```
$ sudo dnf install python3-libselinux  
$ pip install ansible==2.9 pywinrm selinux --user
```

Create a hosts file with the following information:

```
[win]  
<node_ip> ansible_password=<password>  
  
[win:vars]  
ansible_user=<username>  
cluster_address=<cluster_address>  
ansible_connection=winrm  
ansible_ssh_port=5986  
ansible_winrm_server_cert_validation=ignore
```

<cluster_address> is the cluster endpoint. It is the combination of your cluster name and the base domain for your cluster.

```
$ oc cluster-info | head -n1 | sed 's/.*\\//api.//g' | sed 's/:.*//g'
```



<node_ip> is the IPv4 public ip of the Windows instance from the [Amazon console](#). Do not use the DNS address of the instance here.

<username> and <password> are the login credentials for the Windows instance. Note the username listed here must have administrative privileges. In default case, <username> is 'Administrator'

Here is an example hosts file:

```
[win]
40.69.185.26 ansible_password='mypassword'

[win:vars]
ansible_user=Administrator
cluster_address=winc-cluster.devcluster.openshift.com
ansible_connection=winrm
ansible_ssh_port=5986
ansible_winrm_server_cert_validation=ignore
```

Test if Ansible is able to communicate with the Windows instance with the following command:

```
$ ansible win -i <name_of_the_hosts_file> -m win_ping -v
```

Note: If you do not want to provide the password in the hosts file, you can provide the same as an extra-variable to any ansible command. For example, the above command could be executed as:

```
$ ansible win -i <name_of_the_hosts_file> -m win_ping -v --extra-vars
"ansible_password=<password>"
```

Install jq library to parse json files:

```
$ sudo dnf install jq
```

Windows Node Installer

Windows Node Installer (WNI) is an unsupported tool that can automate the process of launching a Windows instance and prepare it for bootstrapping. For more details, please refer to the [README](#).

Download WNI v4.4.2-alpha on your Linux host and ensure you can execute it:

```
$ wget
https://github.com/openshift/windows-machine-config-bootstrapper/releases/download/v
4.4.2-alpha/wni
$ chmod +x wni
```



The following command creates the instance with your AWS account and outputs the credentials. It expects your credentials to be stored on the system and have the `aws_access_key_id` and `aws_secret_access_key`.

```
$ ./wni aws create --kubeconfig <path to OpenShift cluster>/auth/kubeconfig
--credentials <path to aws>/credentials --credential-account <credential-account in
the credentials file, ex: default> --instance-type m5a.large --ssh-key <name of the
existing ssh key> --private-key <path to private key> --dir <directory>
```

For eg.

```
$ ./wni aws create --kubeconfig ~/<cluster_dir>/auth/kubeconfig --credentials
~/aws/credentials --credential-account dev --instance-type m5a.large --ssh-key
aws-key-name --private-key ~/.ssh/key.pem --dir ./windowsnodeinstaller/
```

Once the instance is launched, the console would provide the credentials for login as well as the node-ip. You can now [setup Ansible](#) before moving on to the next steps.

Bootstrap the Windows instance

On a Linux host, run the Ansible Playbook that transfers the necessary files onto the Windows instance and bootstraps it so that it can join the cluster as a worker node.

Note: Playbook assumes you have jq installed. Your active RDP connection might be disrupted during the execution of the playbook. Make sure you have at least 600mb of free space in `/tmp` directory.

Install git to clone github repository:

```
$ sudo dnf install git
```

Run the ansible playbook to bootstrap the windows worker node. Make sure you have at least 6GB of free space in `/tmp` directory.

```
$ git clone https://github.com/openshift/windows-machine-config-bootstrapper.git
$ ansible-playbook -i <path_to_hosts_file>
windows-machine-config-bootstrapper/tools/ansible/tasks/wsutil/main.yml -v
```

```
$ oc get nodes
```

You can now see the Windows instance has joined the cluster

NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-133-44.ec2.internal	Ready	worker	4h5m	v1.17.1



ip-10-0-142-29.ec2.internal	Ready	master	4h14m	v1.17.1
ip-10-0-152-168.ec2.internal	Ready	master	4h14m	v1.17.1
ip-10-0-153-185.ec2.internal	Ready	worker	4h5m	v1.17.1
ip-10-0-164-237.ec2.internal	Ready	worker	4h5m	v1.17.1
ip-10-0-174-28.ec2.internal	Ready	master	4h13m	v1.17.1
ip-10-0-71-15.ec2.internal	Ready	worker	3h30m	v1.17.1

API rate limit exceeded error when running WSU

WSU playbook uses GitHub API to fetch releases for WMCB. You might encounter an API rate limit exceeded error while running WSU playbook in TASK [Get release]. The issue occurs due to GitHub rate-limiting unauthenticated requests at 60 requests per hour. As a workaround, wait for the rate-limit to reset (at most 1 hour) before running the playbook again.

Test Windows workload

We can now create a pod that can be deployed on a Windows instance. We used an example [WebServer](#) deployment to create a pod:

Note: Given the size of Windows images, it is recommended to pull the Docker image mcr.microsoft.com/windows/servercore:ltsc2019 on the instance first, before creating the pods.

On Windows instance, run the following command in a powershell window:

```
> docker pull mcr.microsoft.com/windows/servercore:ltsc2019
```

Note: Refer [RDP section](#) to set up and RDP into your windows node

On the Linux host, deploy the pods:

```
$ oc create -f
https://gist.github.com/suhanime/683ee7b5a2f55c11e3a26a4223170582/raw/d89
3db98944bf615fccfe73e6e4fb19549a362a5/WinWebServer.yaml -n default
```

Once the deployment has been created, we can check the status of the pods:

```
$ oc get pods -n default
```

```
$ oc get pods -n default
NAME                                READY   STATUS    RESTARTS   AGE
win-webserver-6f5bdc5b95-x65tq     1/1     Running   0           14m
```




We have created a service of [LoadBalancer](#) type:

```
$ oc get service win-webserver -n default
NAME                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
win-webserver       LoadBalancer       172.30.156.18    a22c5338b80f3431bb6428f8a43a4877-890285700.us-east-2.elb.amazonaws.com
80:30897/TCP        33s
```

```
$ curl a22c5338b80f3431bb6428f8a43a4877-890285700.us-east-2.elb.amazonaws.com
<html><body><H1>Windows Container Web Server</H1><p>IP 10.132.1.2 callerCount 4
</body></html>
```

Deploying in a namespace other than default

In order to deploy into a different namespace SCC must be disabled in that namespace. This should never be used in production, and any namespace that this has been done to should not be used to run Linux pods.

To skip SCC for a namespace the label `openshift.io/run-level = 1` should be applied to the namespace. This will apply to both Linux and windows pods, and thus Linux pods should not be deployed into this namespace.

For example, to create a new project and apply the label:

```
$ oc new-project <project_name>
$ oc label namespace <project_name> "openshift.io/run-level=1"
```