WHY ANSIBLE ON AWS IS MORE POWERFUL THAN AWS CLOUDFORMATION?

In the Lab 1-Deployement of CloudFormation Template, we deployed entire Network Design using CloudFormation Template. Our CloudFormation Stack includes Custom VPC, Subnets, Internet Gateway, NAT Gateways, Security Groups etc. CloudFormation is a recommended solution when you want to launch **Resources** in your AWS Infrastructure. But when it comes to deployment of **Applications** at Operating System level, it is not a recommended solution at all. We can deploy Applications using CloudFormation Templates either by using Pre-Configured AMIs or LAMP Configurations. But this is a tedious & time-consuming task. We can overcome this problem by running Ansible Playbooks on AWS. By running Ansible Playbooks on AWS, we can deploy pre-configured EC2 Instances within Minutes. In this lab, we are going to configure an Ansible Controller EC2 Instance on which we will run a Playbook to deploy a new EC2 Instance which will have all the required packages for Apache Server.

Below is the list of Tasks:

Task 1: Create IAM Role

Task 2: Launch & Configure EC2 Instances with SSM Agent

Task 3: Create a S3 Bucket to store SSM Logs

Task 4: AWS Systems Manager: Managed Instances

Task 5: AWS-Systems Manager: Run Command (Ansible Installation)

Task 6: Ansible Installation Check

Task 7: Create an IAM User

Task 8: PIP, BOTO Configurations

Task 9: Writing an Ansible Playbook

Task 10: Execute Ansible Playbook on EC2 Instance

Task 1: Create IAM Role

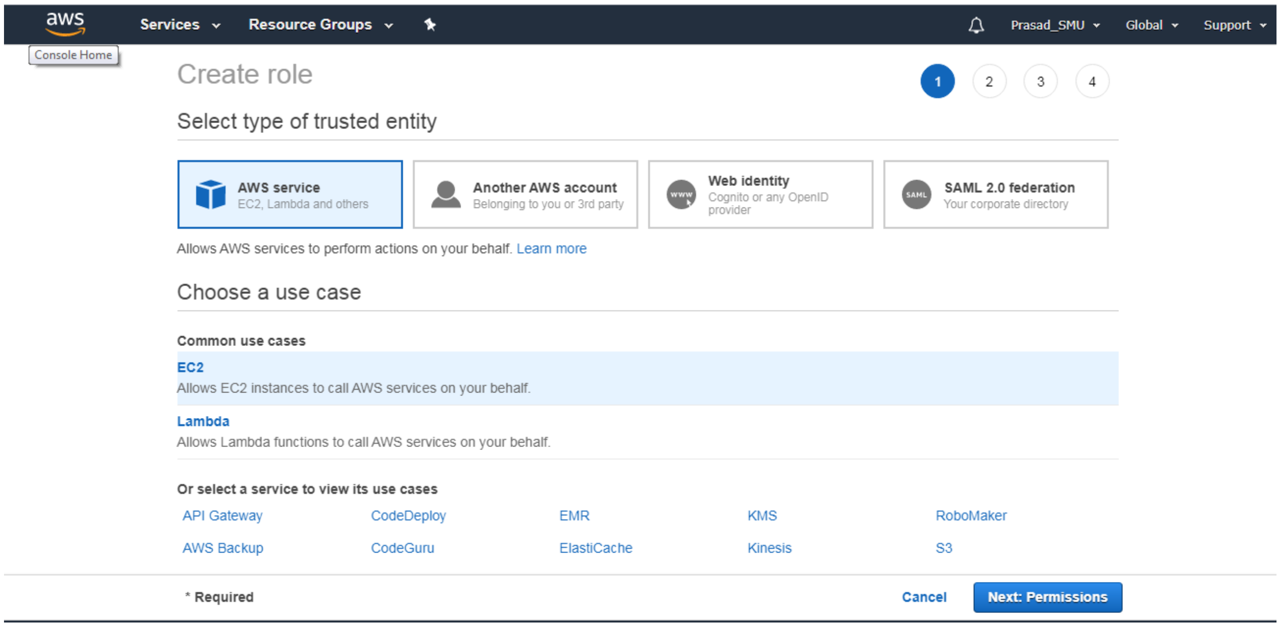
Login to the AWS Management Console.

Navigate to IAM Service and click on Roles.

Click on Create Role.

Make sure to select the Use Case as **EC2**.

Click Next: Permissions.



Create role

Select type of trusted entity

Allows AWS services to perform actions on your behalf.

Choose a use case

Common use cases

EC2
Allows EC2 instances to call AWS services on your behalf.

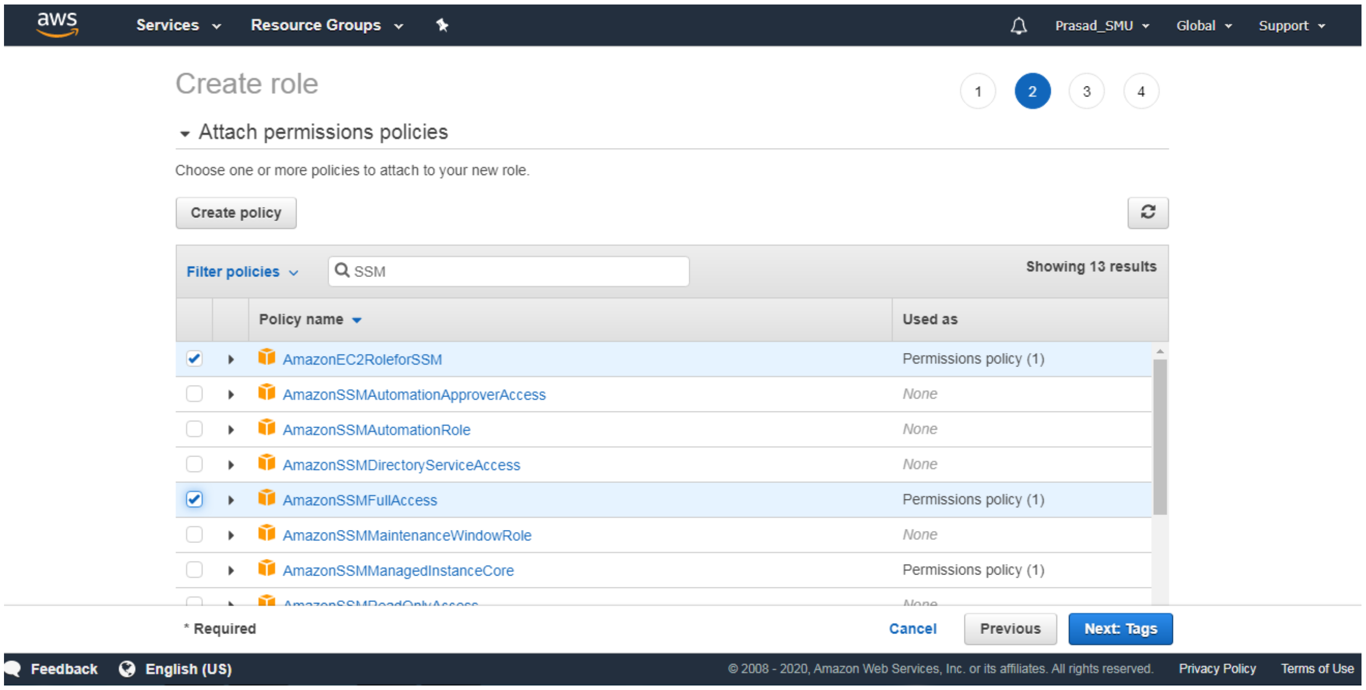
Lambda
Allows Lambda functions to call AWS services on your behalf.

Or select a service to view its use cases

* Required

Select the below three Default IAM Policies:

1. **AmazonEC2RoleforSSM**
2. **AmazonSSMFullAccess**
3. **AmazonEC2FullAccess**



Create role

Attach permissions policies

Choose one or more policies to attach to your new role.

SSM

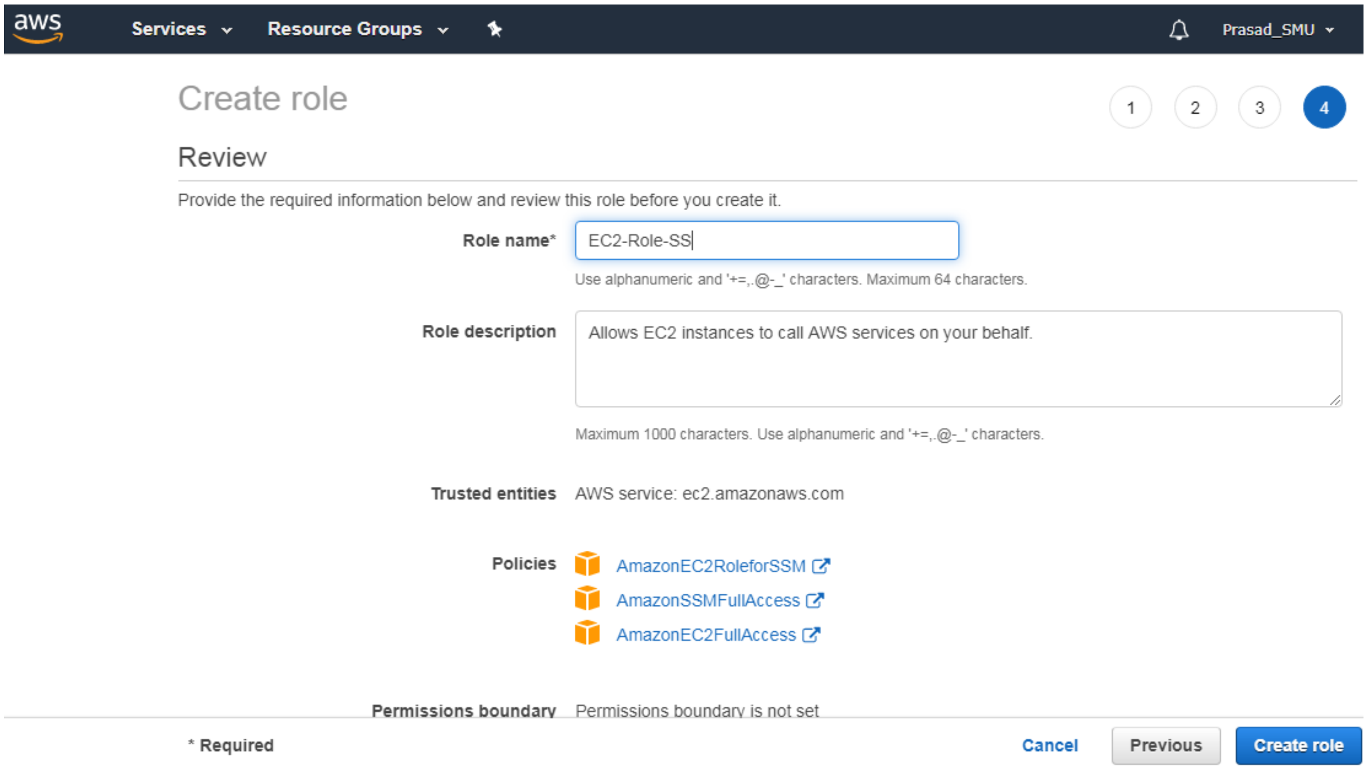
Showing 13 results

	Policy name	Used as
✓	AmazonEC2RoleforSSM	Permissions policy (1)
	AmazonSSMAutomationApproverAccess	None
	AmazonSSMAutomationRole	None
	AmazonSSMDirectoryServiceAccess	None
✓	AmazonSSMFullAccess	Permissions policy (1)
	AmazonSSMMaintenanceWindowRole	None
	AmazonSSMManagedInstanceCore	Permissions policy (1)
	AmazonSSMReadOnlyAccess	None

* Required

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Give the Role Name as per your Choice and click on Create Role.



Create role

Review

Provide the required information below and review this role before you create it.

Role name* EC2-Role-SS

Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.

Role description Allows EC2 instances to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Trusted entities AWS service: ec2.amazonaws.com

Policies
- AmazonEC2RoleforSSM
- AmazonSSMFullAccess
- AmazonEC2FullAccess

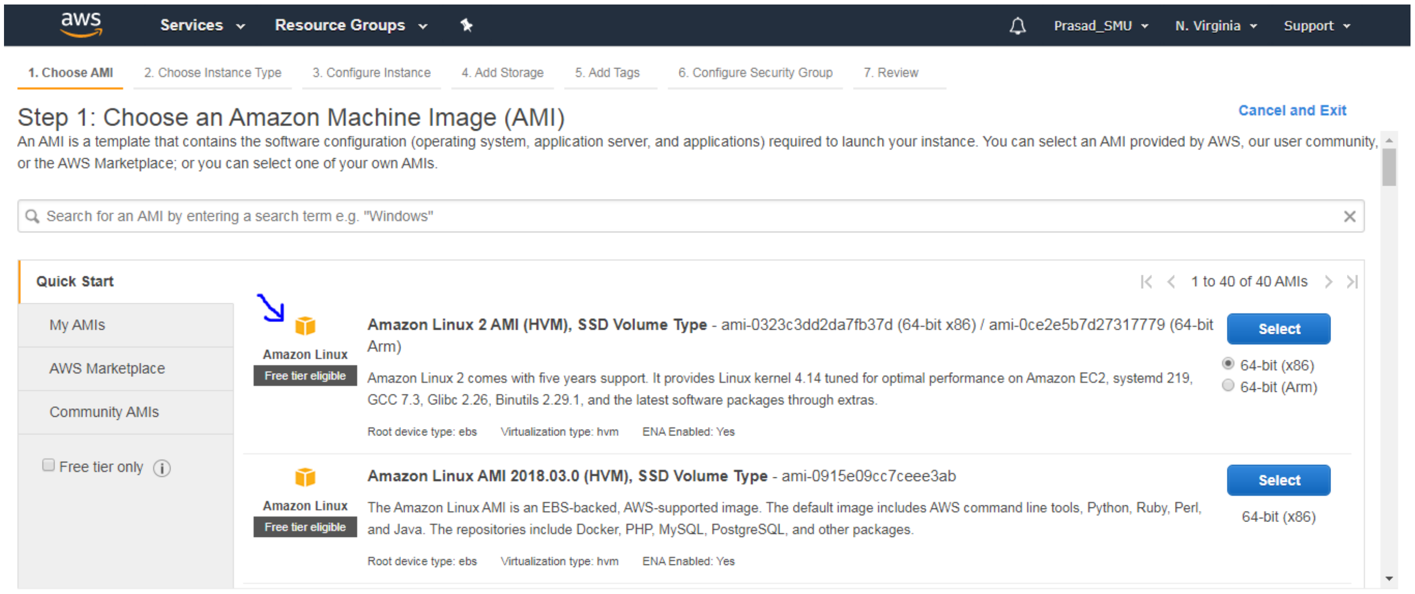
Permissions boundary Permissions boundary is not set

* Required

Task 2: Launch & Configure EC2 Instances with SSM Agent

Navigate to EC2 Service and click on Launch Instance.

Select the **Amazon Linux AMI**.



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0323c3dd2da7fb37d (64-bit x86) / ami-0ce2e5b7d27317779 (64-bit Arm)

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.

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

● 64-bit (x86) ○ 64-bit (Arm)

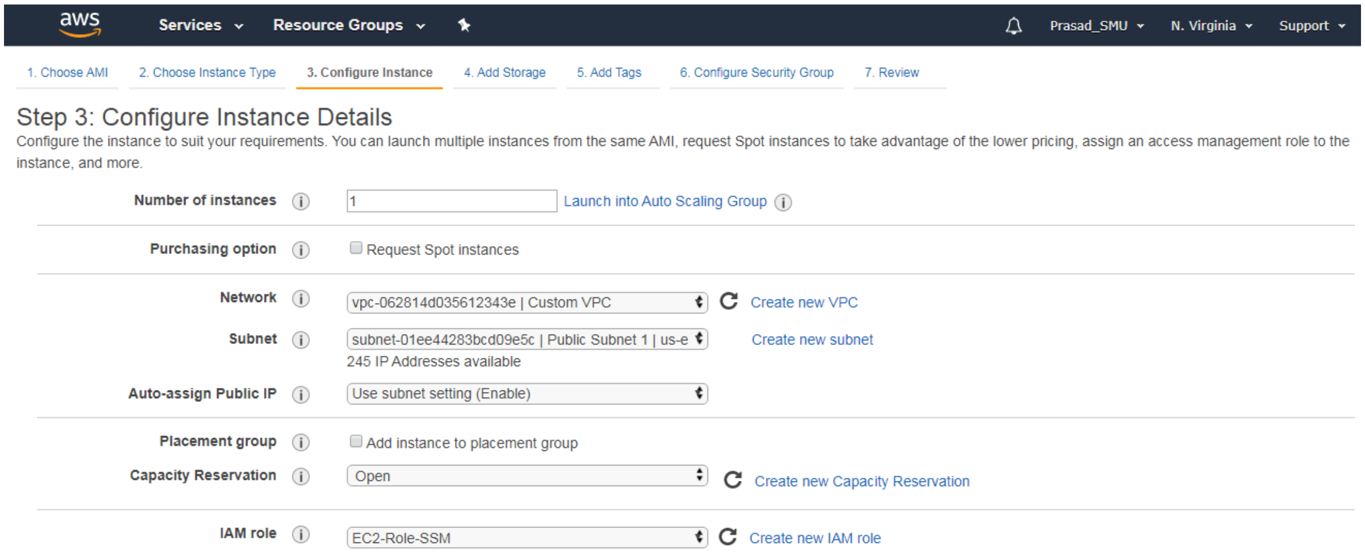
Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-0915e09cc7ceee3ab

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

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

64-bit (x86)

Select the Number of Instances as 1, select the Network as our Custom VPC, Select Subnet as Public Subnet 1 and select the IAM Role which you configured in the Task 1.



Step 3: Configure Instance Details

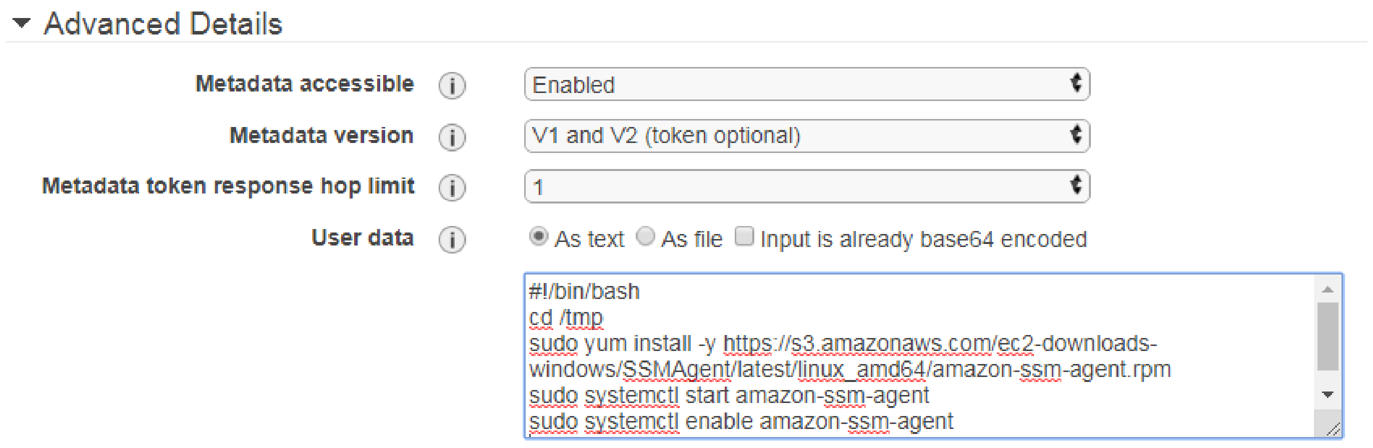
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1
Purchasing option	☐ Request Spot instances
Network	vpc-062814d035612343e | Custom VPC
Subnet	subnet-01ee44283bcd09e5c | Public Subnet 1 | us-e 245 IP Addresses available
Auto-assign Public IP	Use subnet setting (Enable)
Placement group	☐ Add instance to placement group
Capacity Reservation	Open
IAM role	EC2-Role-SSM

Since AWS Systems Manager is AGENTLESS, we need to install Packages for Systems Manager (SSM) to connect with Target Instances.

Scroll down on the same page, click on Advanced Details and in User Data field bootstrap the below commands.

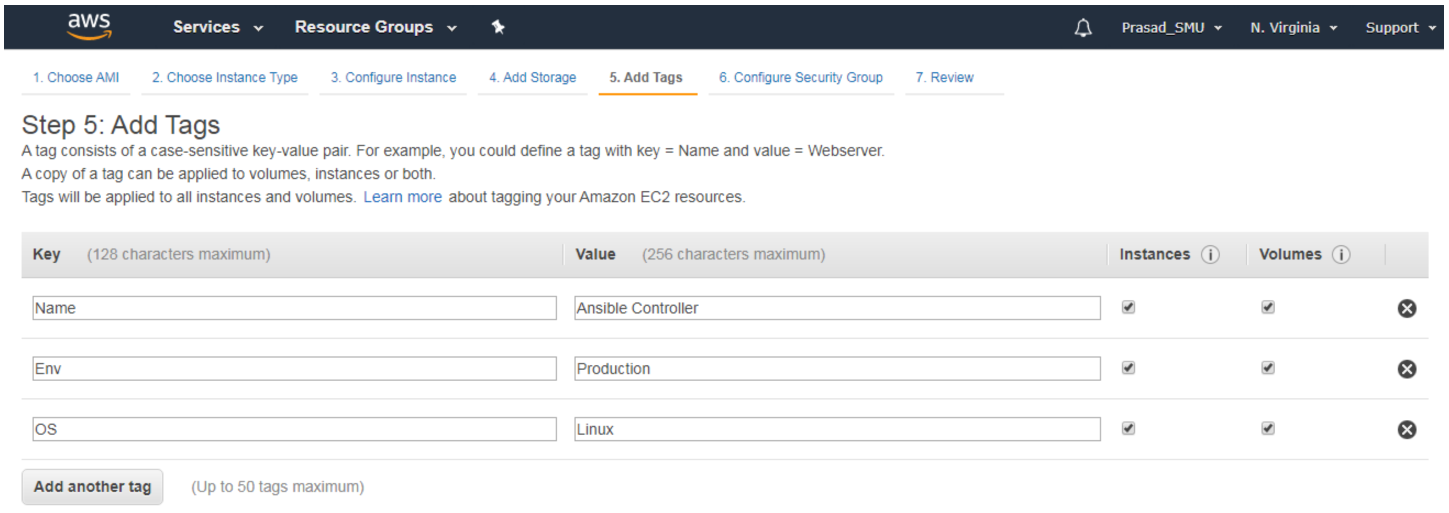
I've provided the Commands in text file.



Metadata accessible	Enabled
Metadata version	V1 and V2 (token optional)
Metadata token response hop limit	1
User data	[✓ As text] As file Input is already base64 encoded

```
#!/bin/bash
cd /tmp
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
sudo systemctl start amazon-ssm-agent
sudo systemctl enable amazon-ssm-agent
```

You can mention Tags as per your choice.



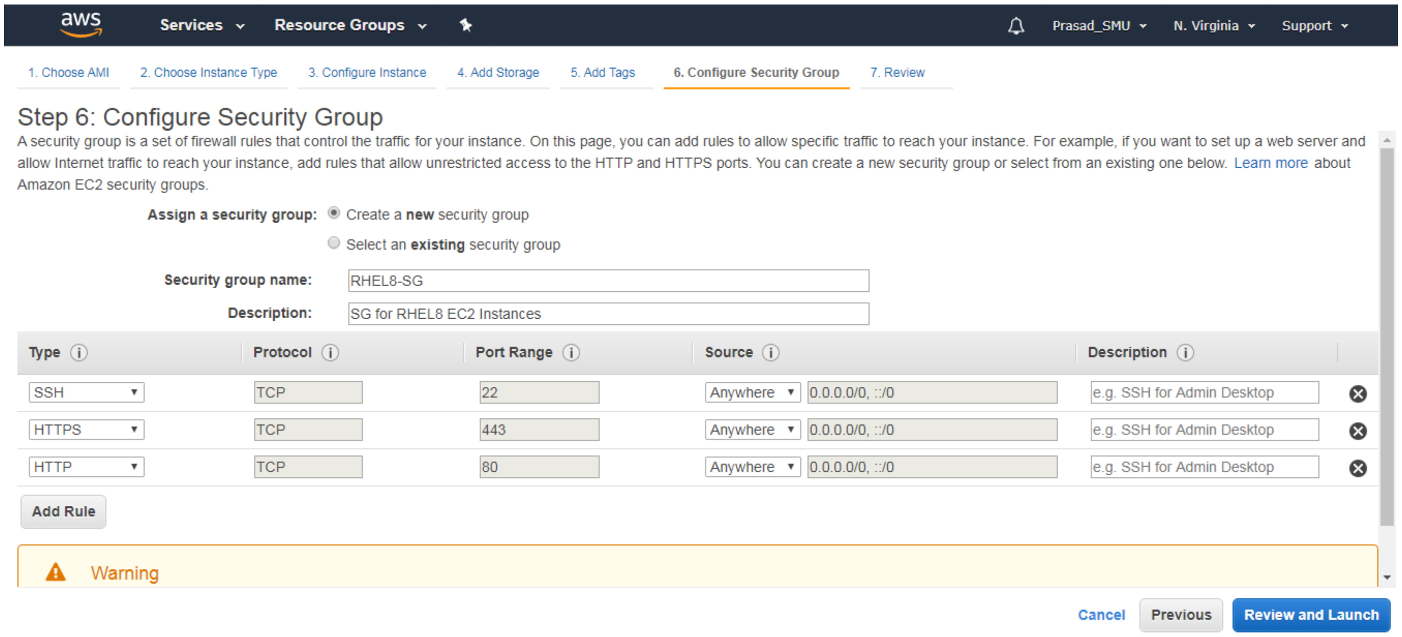
Step 5: Add Tags

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	Ansible Controller	✓	✓
Env	Production	✓	✓
OS	Linux	✓	✓

Click Next: Security Groups.

Create a new Security Group. Give the Name & Discription as per your choice. Allow SSH, HTTPS, HTTP Inbound traffic from Anywhere.



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: [✓ Create a **new** security group] Select an **existing** security group

Security group name: RHEL8-SG

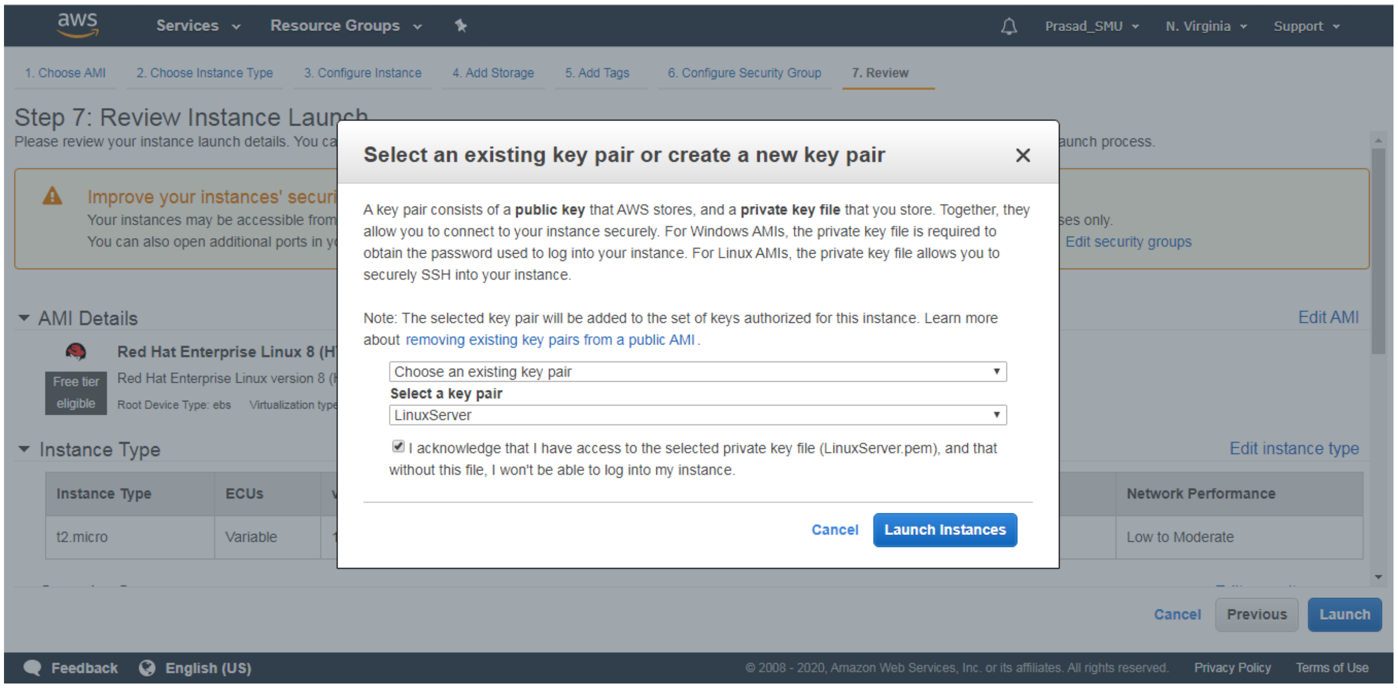
Description: SG for RHEL8 EC2 Instances

Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Anywhere	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Anywhere	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Anywhere	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Click on Review and Launch.

Select the existing Key Pair which you've using for previous labs.

Click on Launch Instances.



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

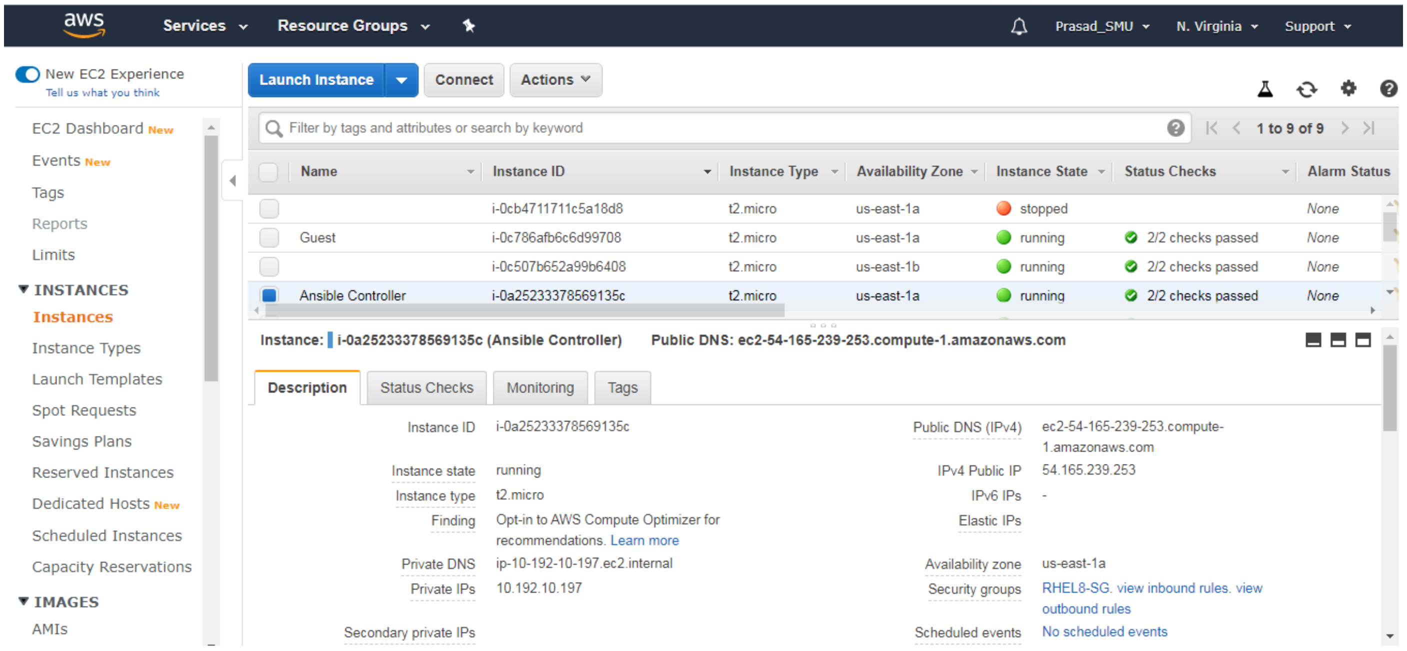
LinuxServer

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

Cancel | Launch Instances

Instance Type	ECUs		Network Performance
t2.micro	Variable		Low to Moderate

You can see that the Highlighted Instance has been launched Successfully!!!!!



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
	i-0cb4711711c5a18d8	t2.micro	us-east-1a	stopped		None
Guest	i-0c786afb6c6d99708	t2.micro	us-east-1a	running	2/2 checks passed	None
	i-0c507b652a99b6408	t2.micro	us-east-1b	running	2/2 checks passed	None
Ansible Controller	i-0a25233378569135c	t2.micro	us-east-1a	running	2/2 checks passed	None

Instance: i-0a25233378569135c (Ansible Controller) Public DNS: ec2-54-165-239-253.compute-1.amazonaws.com

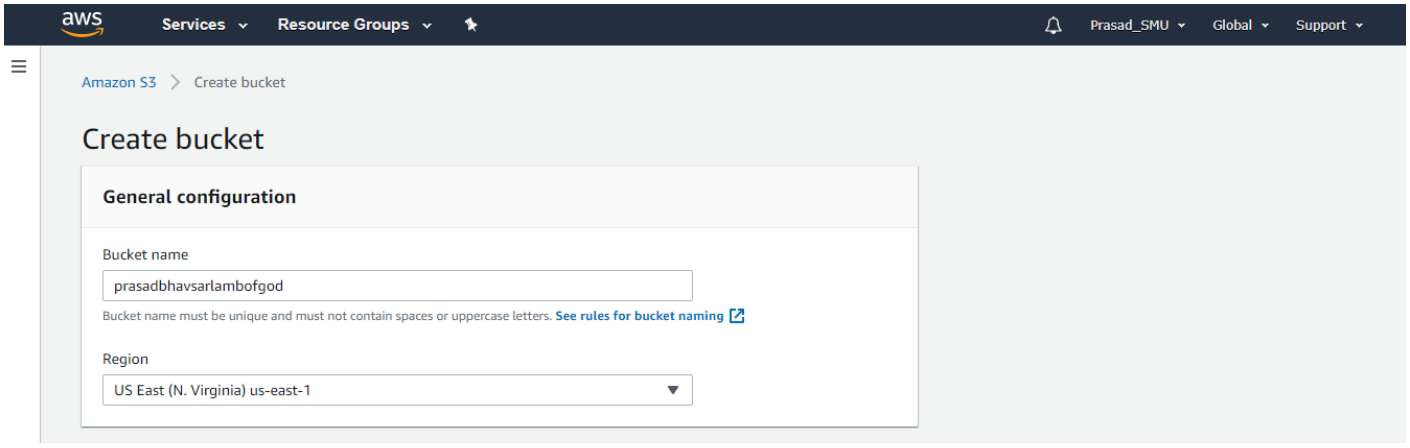
Instance ID	i-0a25233378569135c	Public DNS (IPv4)	ec2-54-165-239-253.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	54.165.239.253
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	
Private DNS	ip-10-192-10-197.ec2.internal	Availability zone	us-east-1a
Private IPs	10.192.10.197	Security groups	RHEL8-SG. view inbound rules. view outbound rules
Secondary private IPs		Scheduled events	No scheduled events

<u>Task 3: Create a S3 Bucket to store SSM Logs</u>

Navigate to S3 Service.

Click on Create Bucket.

Give a unique Bucket Name as per your choice.



Create bucket

General configuration

Bucket name

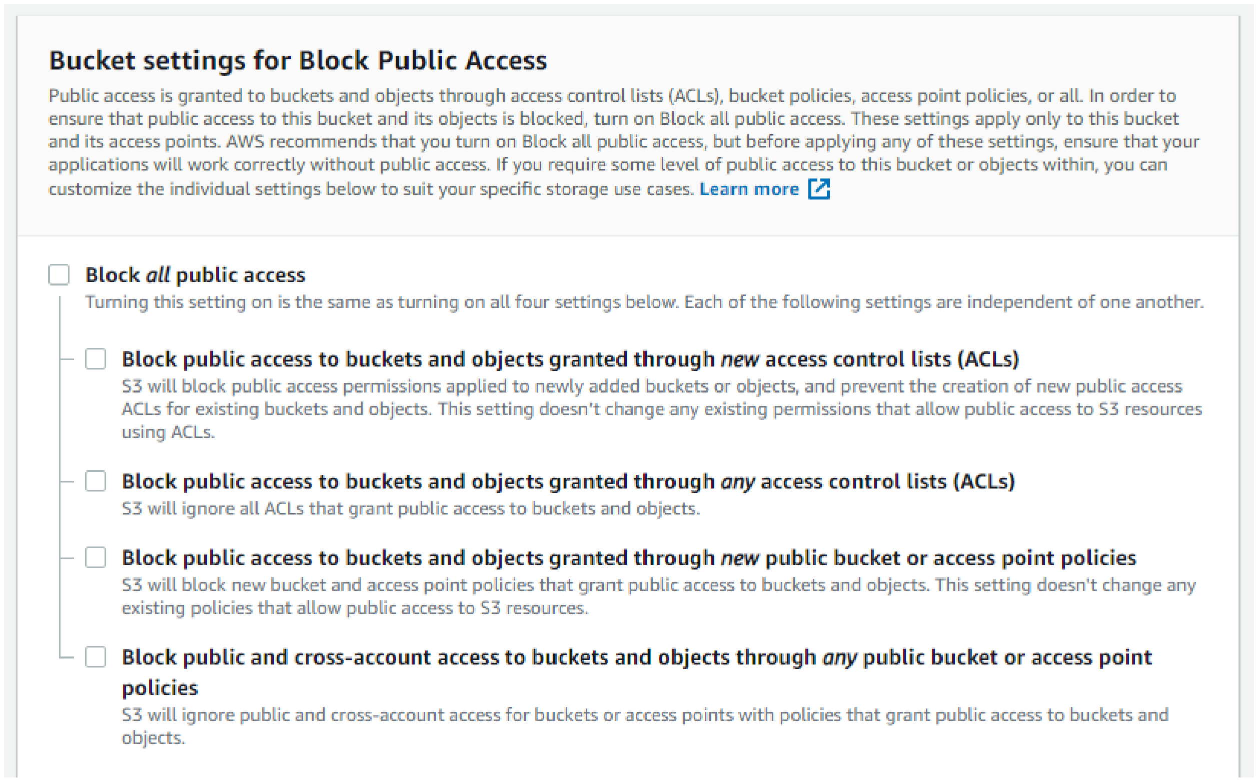
prasadbhavsarlambofgod

Bucket name must be unique and must not contain spaces or uppercase letters. See rules for bucket naming

Region

US East (N. Virginia) us-east-1

Make Bucket publicly Available by unchecking the Block all public access.



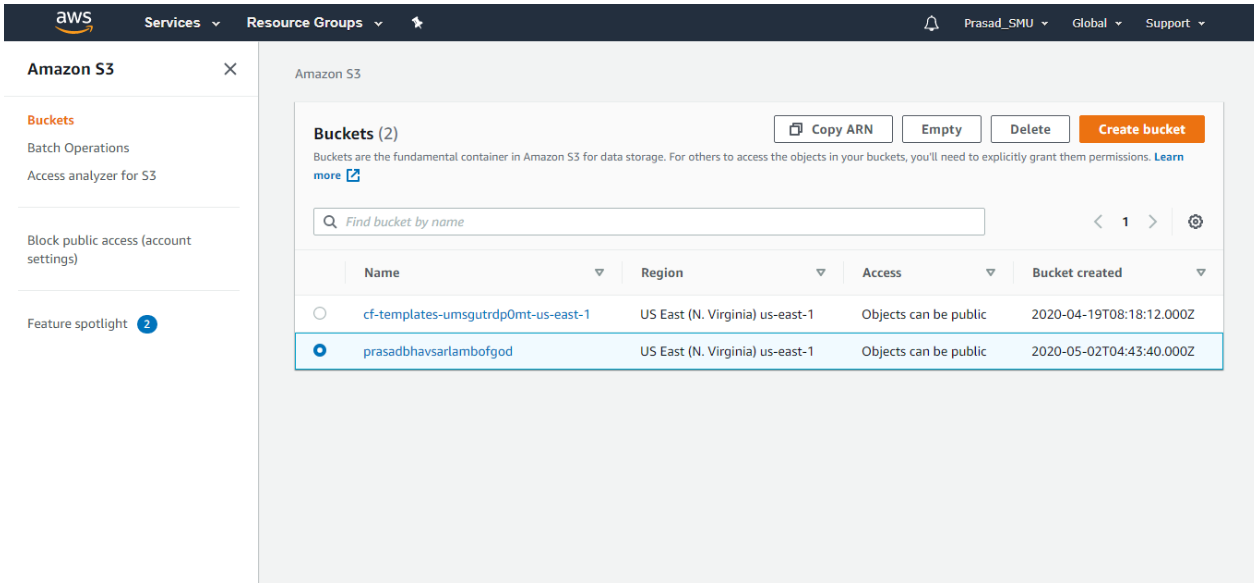
Bucket settings for Block Public Access

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

☐ **Block *all* public access**
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- ☐ **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
  S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- ☐ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
  S3 will ignore all ACLs that grant public access to buckets and objects.
- ☐ **Block public access to buckets and objects granted through *new* public bucket or access point policies**
  S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- ☐ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
  S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Click on Create. S3 Bucket has been successfully created to store Systems Manager (SSM) logs.



Amazon S3

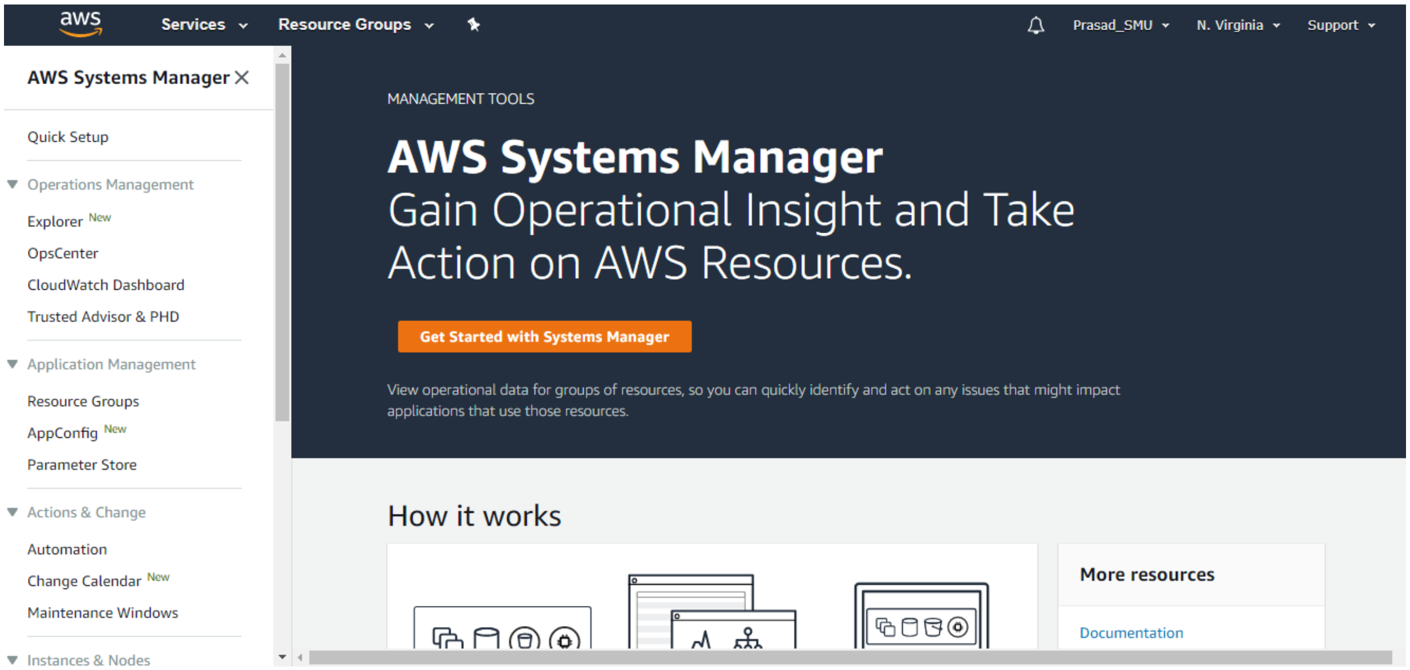
Buckets (2)

Buckets are the fundamental container in Amazon S3 for data storage. For others to access the objects in your buckets, you'll need to explicitly grant them permissions. Learn more

	Name	Region	Access	Bucket created
○	cf-templates-umsgutrdp0mt-us-east-1	US East (N. Virginia) us-east-1	Objects can be public	2020-04-19T08:18:12.000Z
●	prasadbhavsarlambofgod	US East (N. Virginia) us-east-1	Objects can be public	2020-05-02T04:43:40.000Z

Task 4: AWS Systems Manager: Managed Instances

Navigate to AWS Systems Manager Service.

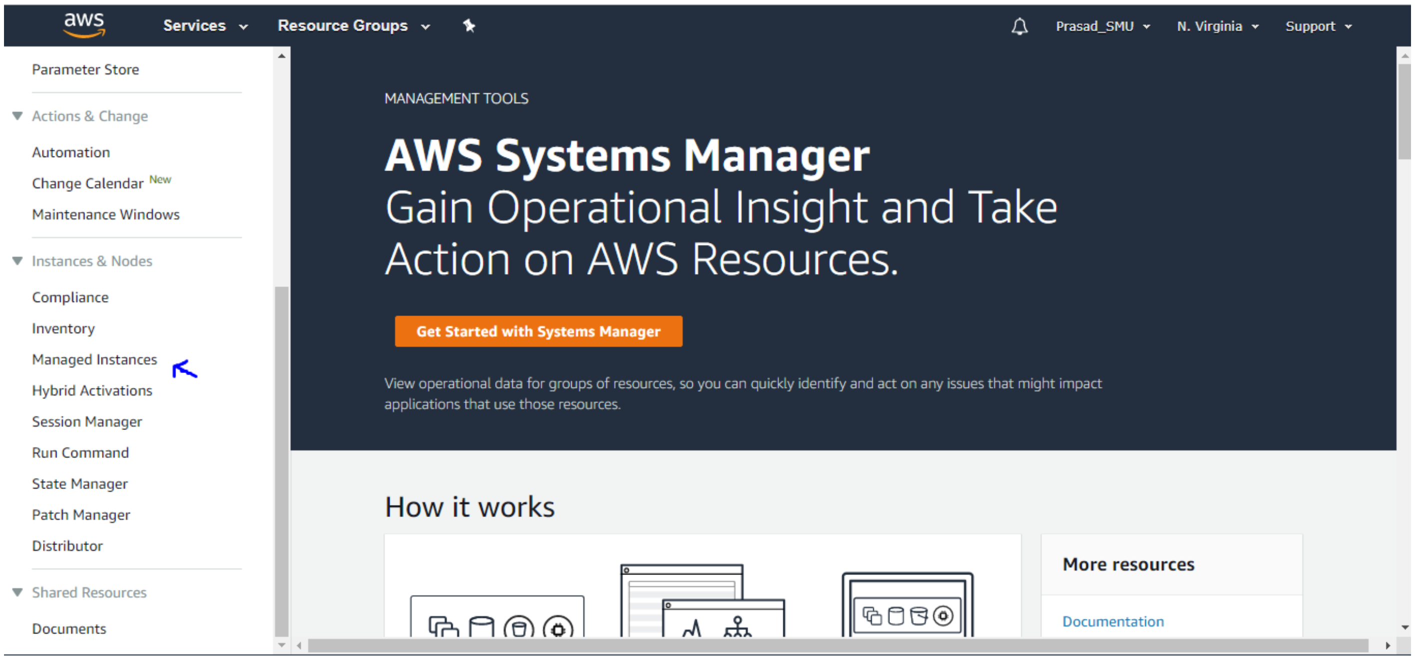


MANAGEMENT TOOLS

AWS Systems Manager
Gain Operational Insight and Take Action on AWS Resources.

View operational data for groups of resources, so you can quickly identify and act on any issues that might impact applications that use those resources.

One the left-hand side, click on Managed Instances.



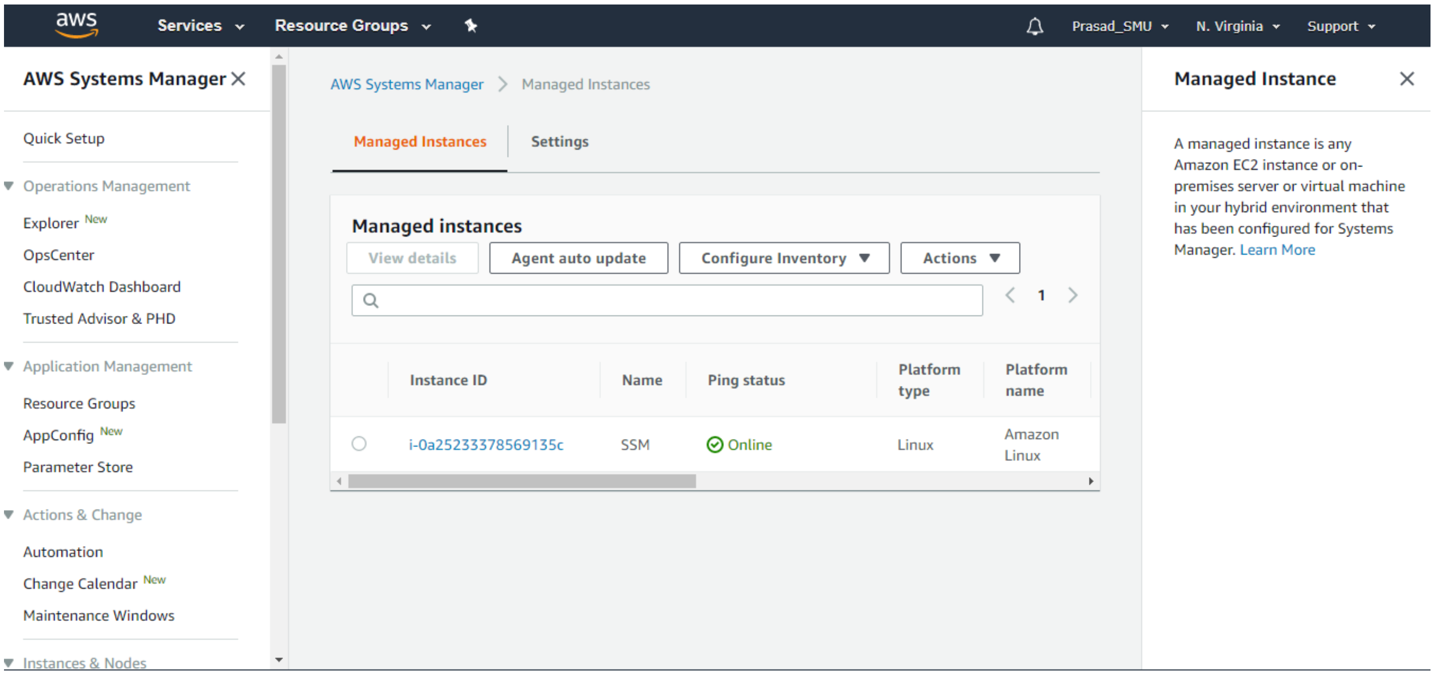
MANAGEMENT TOOLS

AWS Systems Manager
Gain Operational Insight and Take Action on AWS Resources.

View operational data for groups of resources, so you can quickly identify and act on any issues that might impact applications that use those resources.

You should see Instance which we launched in Task 2.

If you do not see any Instance in Managed Instances tab, it means Systems Manager Agent is not Installed on the EC2 Instance.



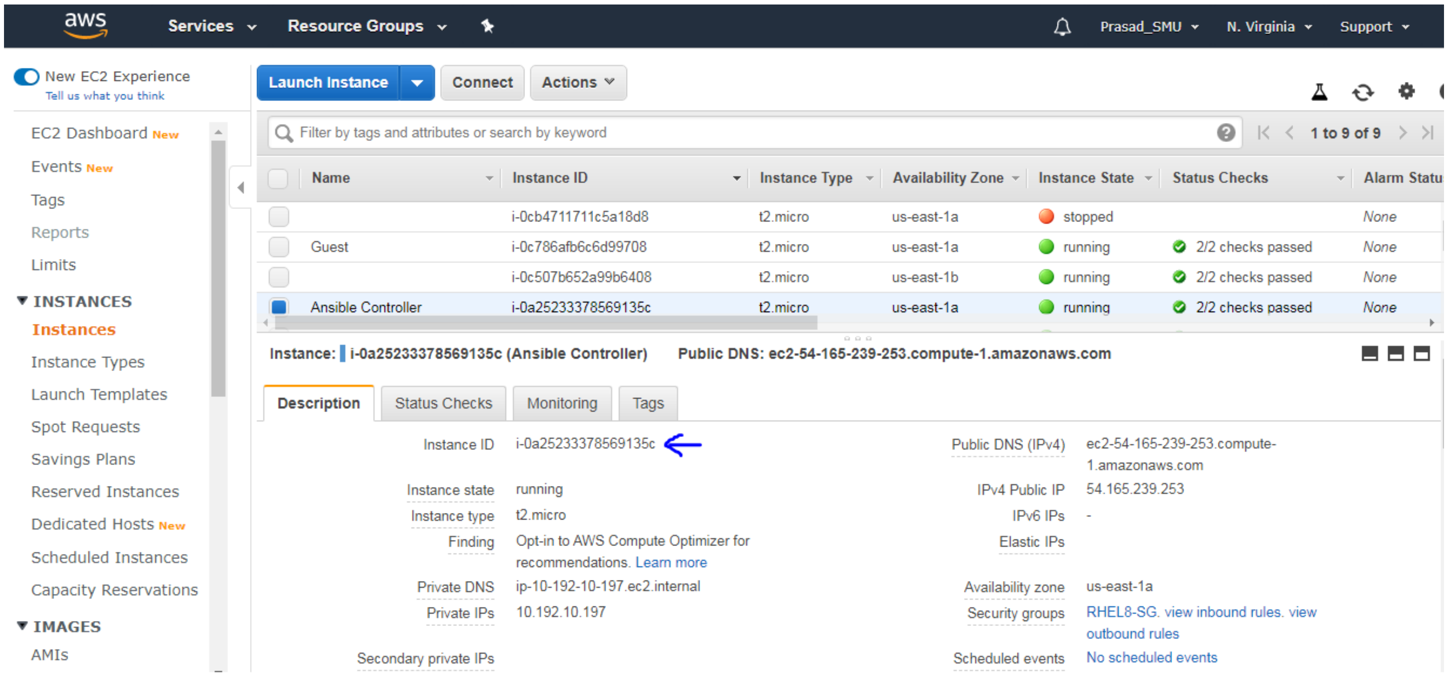
Managed instances

Instance ID	Name	Ping status	Platform type	Platform name
i-0a25233378569135c	SSM	Online	Linux	Amazon Linux

Managed Instance

A managed instance is any Amazon EC2 instance or on-premises server or virtual machine in your hybrid environment that has been configured for Systems Manager. Learn More

You can also verify the Instance IDs from EC2 Service Dashboard.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Statu
	i-0cb4711711c5a18d8	t2.micro	us-east-1a	stopped		None
Guest	i-0c786afb6c6d99708	t2.micro	us-east-1a	running	2/2 checks passed	None
	i-0c507b652a99b6408	t2.micro	us-east-1b	running	2/2 checks passed	None
Ansible Controller	i-0a25233378569135c	t2.micro	us-east-1a	running	2/2 checks passed	None

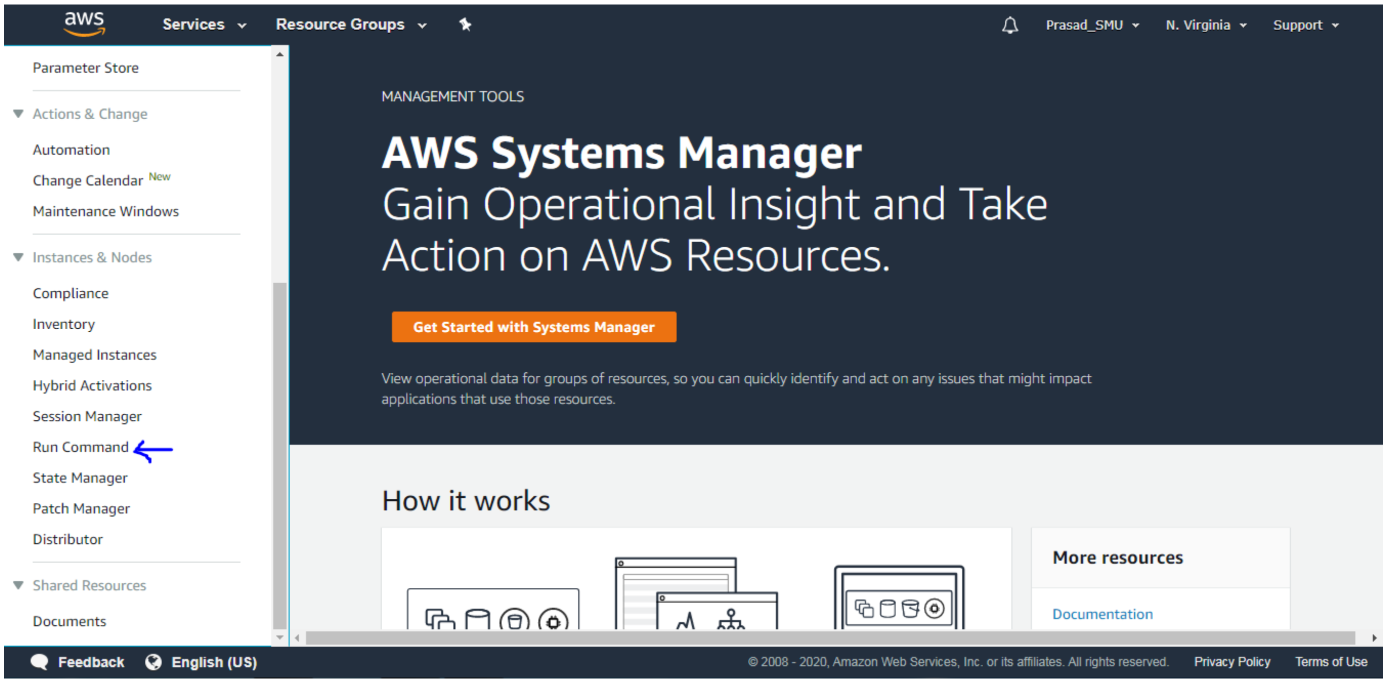
Instance: i-0a25233378569135c (Ansible Controller) Public DNS: ec2-54-165-239-253.compute-1.amazonaws.com

Description | Status Checks | Monitoring | Tags

Instance ID	i-0a25233378569135c	Public DNS (IPv4)	ec2-54-165-239-253.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	54.165.239.253
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	
Private DNS	ip-10-192-10-197.ec2.internal	Availability zone	us-east-1a
Private IPs	10.192.10.197	Security groups	RHEL8-SG. view inbound rules. view outbound rules
Secondary private IPs		Scheduled events	No scheduled events

Task 5: AWS-Systems Manager: Run Command (Ansible Installation)

Now under the same Service, on the left-hand side, click on Run Command.



Parameter Store

Actions & Change
- Automation
- Change Calendar New
- Maintenance Windows

Instances & Nodes
- Compliance
- Inventory
- Managed Instances
- Hybrid Activations
- Session Manager
- Run Command
- State Manager
- Patch Manager
- Distributor

Shared Resources
- Documents

MANAGEMENT TOOLS

AWS Systems Manager
Gain Operational Insight and Take Action on AWS Resources.

Get Started with Systems Manager

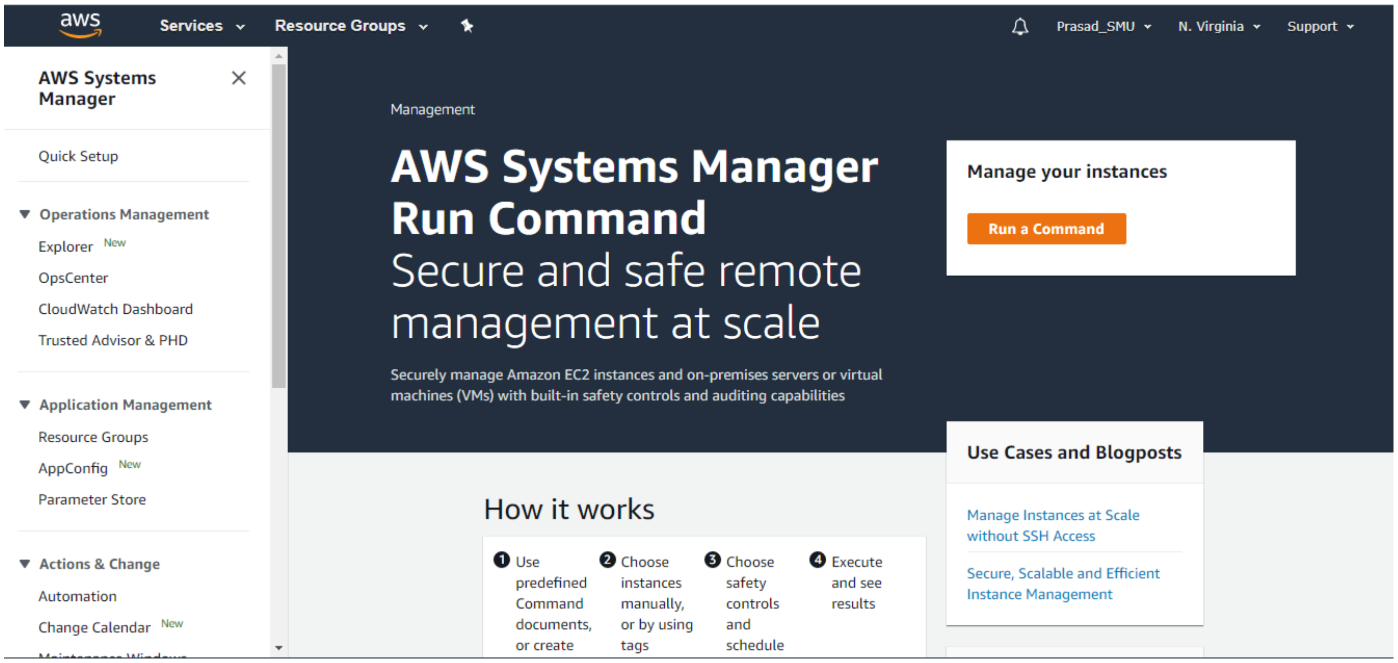
View operational data for groups of resources, so you can quickly identify and act on any issues that might impact applications that use those resources.

How it works

More resources
Documentation

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Click on Run a Command.



AWS Systems Manager

Quick Setup

Operations Management
- Explorer New
- OpsCenter
- CloudWatch Dashboard
- Trusted Advisor & PHD

Application Management
- Resource Groups
- AppConfig New
- Parameter Store

Actions & Change
- Automation
- Change Calendar New

Management

AWS Systems Manager Run Command
Secure and safe remote management at scale

Securely manage Amazon EC2 instances and on-premises servers or virtual machines (VMs) with built-in safety controls and auditing capabilities

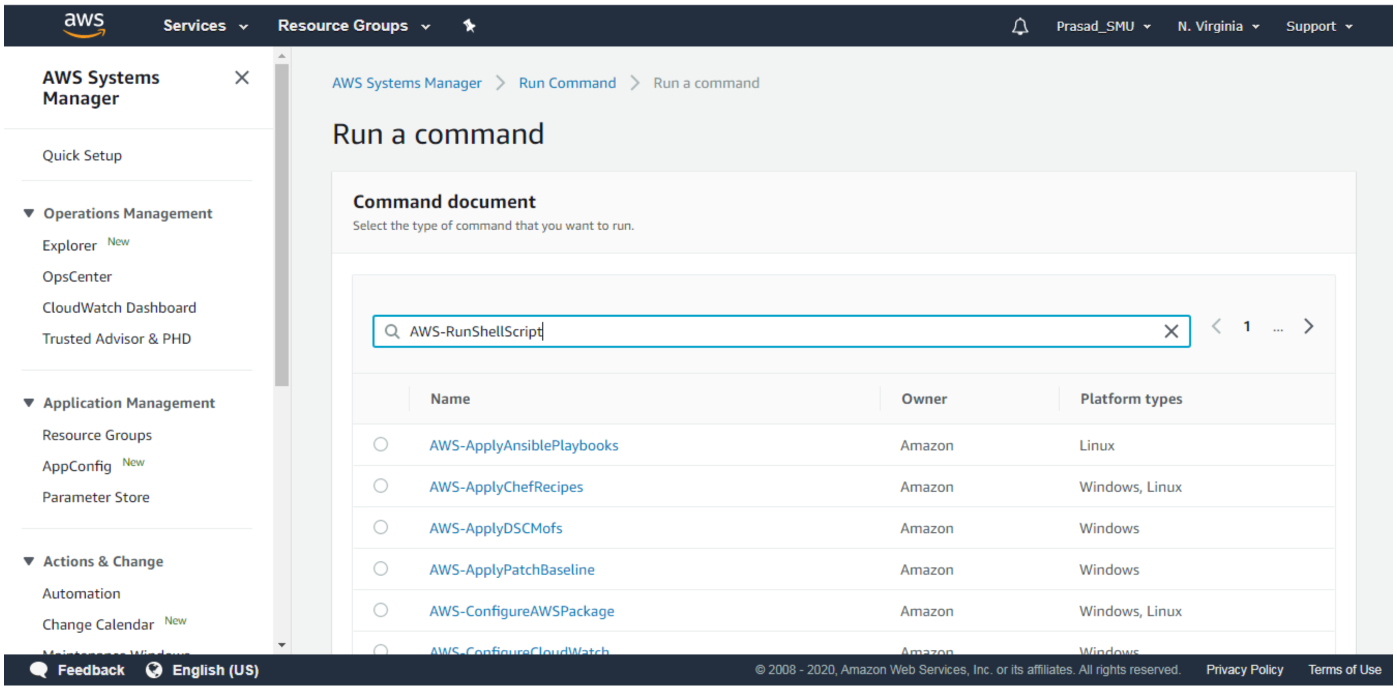
Manage your instances
Run a Command

How it works
1. Use predefined Command documents, or create
2. Choose instances manually, or by using tags
3. Choose safety controls and schedule
4. Execute and see results

Use Cases and Blogposts
- Manage Instances at Scale without SSH Access
- Secure, Scalable and Efficient Instance Management

Under Command Document, search for the below AWS Managed Document.

AWS-RunShellScript



Run a command

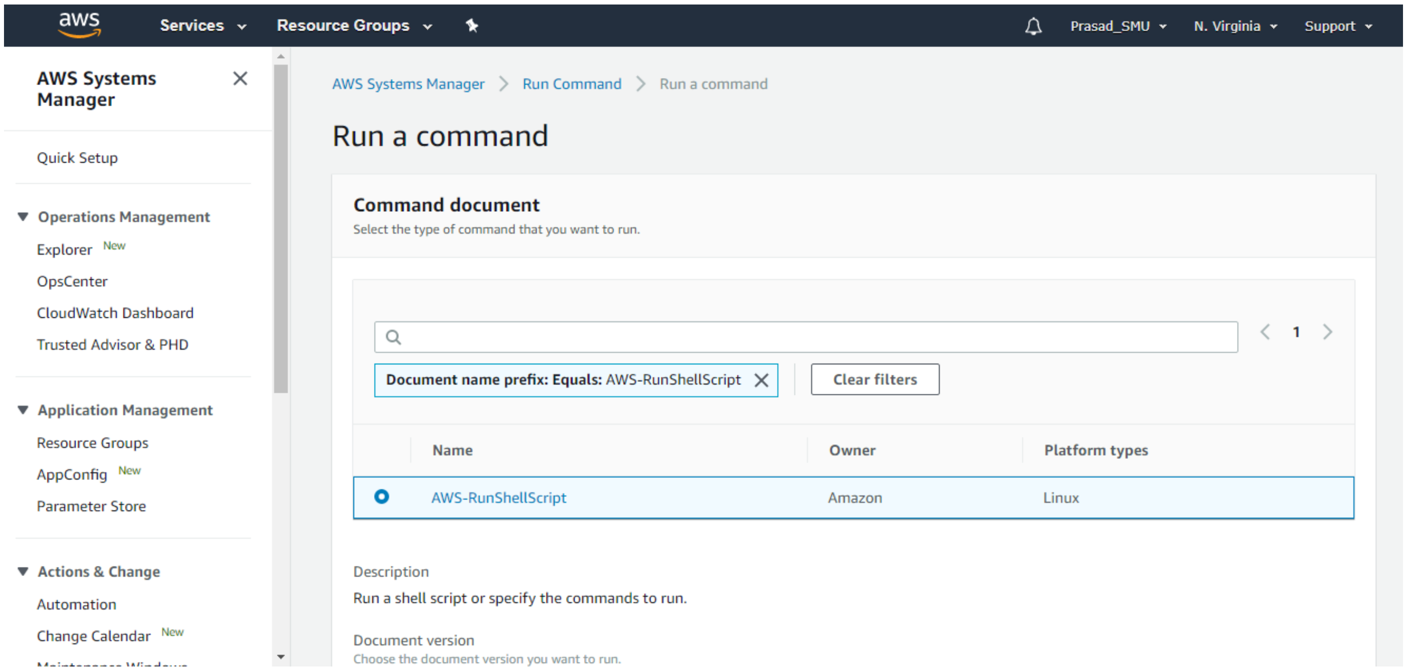
Command document

Select the type of command that you want to run.

AWS-RunShellScript

Name	Owner	Platform types
AWS-ApplyAnsiblePlaybooks	Amazon	Linux
AWS-ApplyChefRecipes	Amazon	Windows, Linux
AWS-ApplyDSCMofs	Amazon	Windows
AWS-ApplyPatchBaseline	Amazon	Windows
AWS-ConfigureAWSPackage	Amazon	Windows, Linux
AWS-ConfigureCloudWatch	Amazon	Windows

Select the Command Document.



Run a command

Command document

Select the type of command that you want to run.

Document name prefix: Equals: AWS-RunShellScript

Name	Owner	Platform types
● AWS-RunShellScript	Amazon	Linux

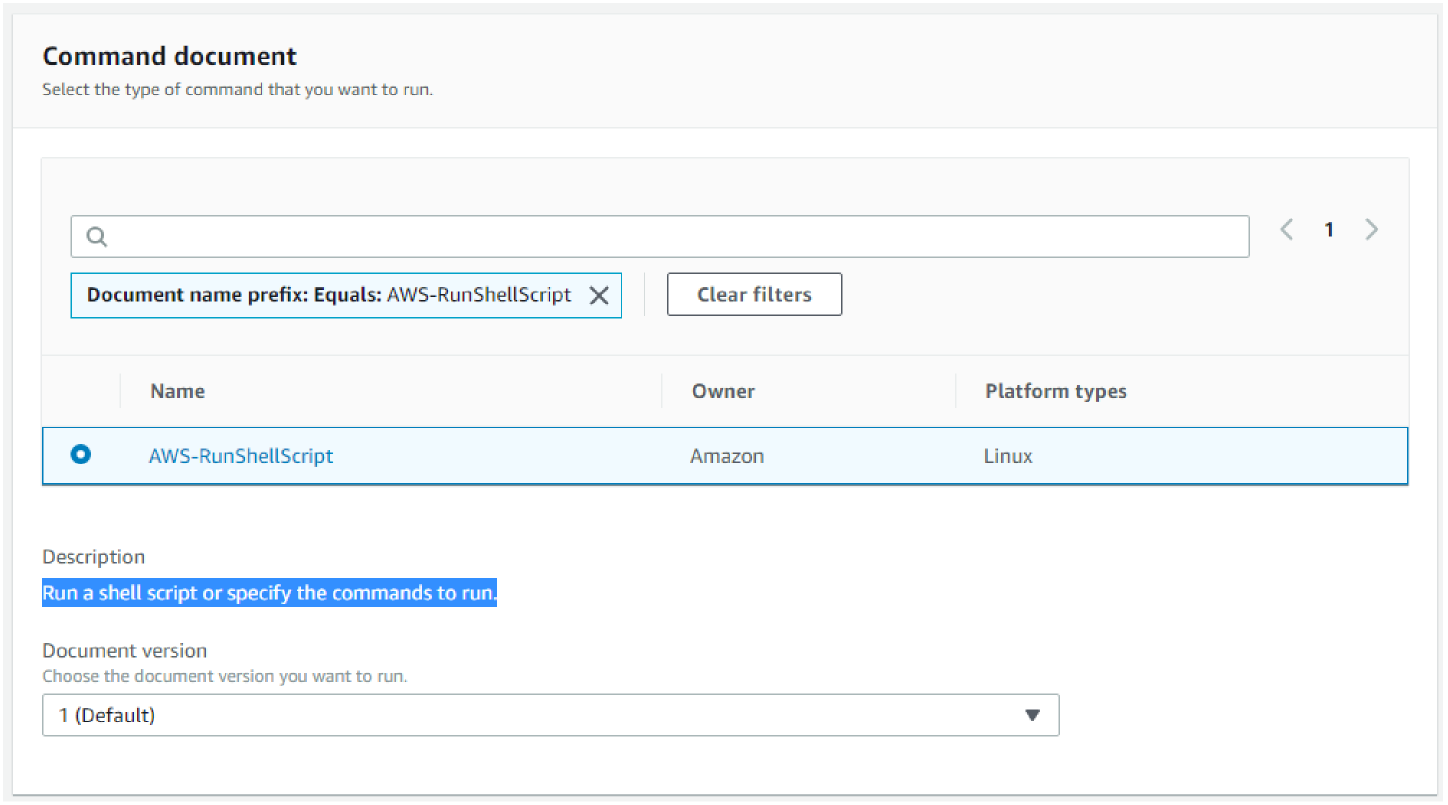
Description

Run a shell script or specify the commands to run.

Document version

Choose the document version you want to run.

Read the highlighted Description.



Command document

Select the type of command that you want to run.

Document name prefix: Equals: AWS-RunShellScript

	Name	Owner	Platform types
●	AWS-RunShellScript	Amazon	Linux

Description

Run a shell script or specify the commands to run.

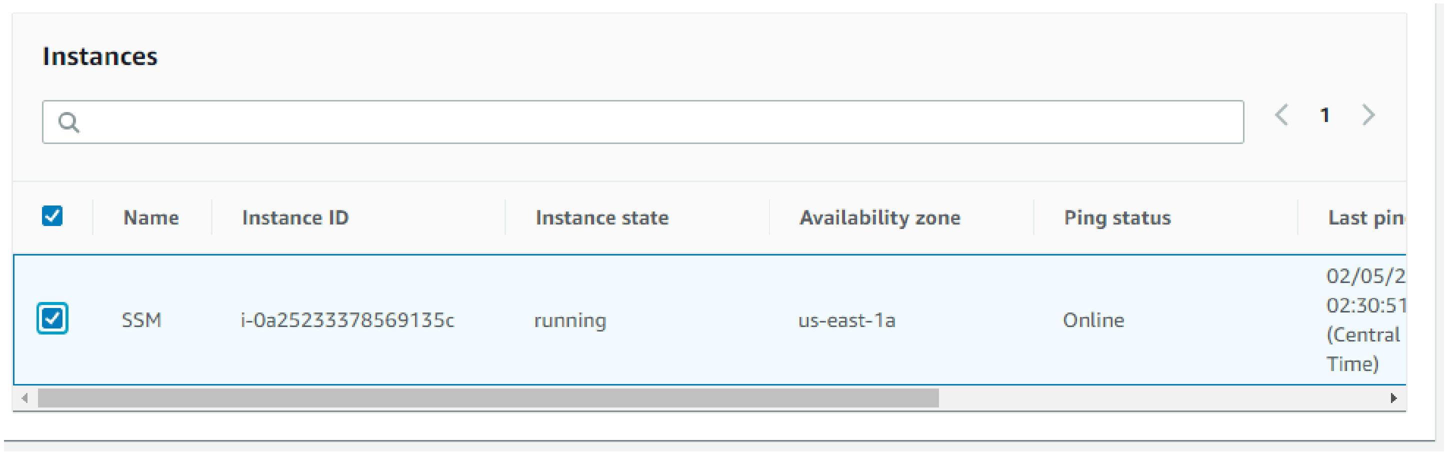
Document version

Choose the document version you want to run.

1 (Default)

Under Targets, click on Choose Instances Manually and select both the EC2 Instances.

You can also select Instances using Tags.

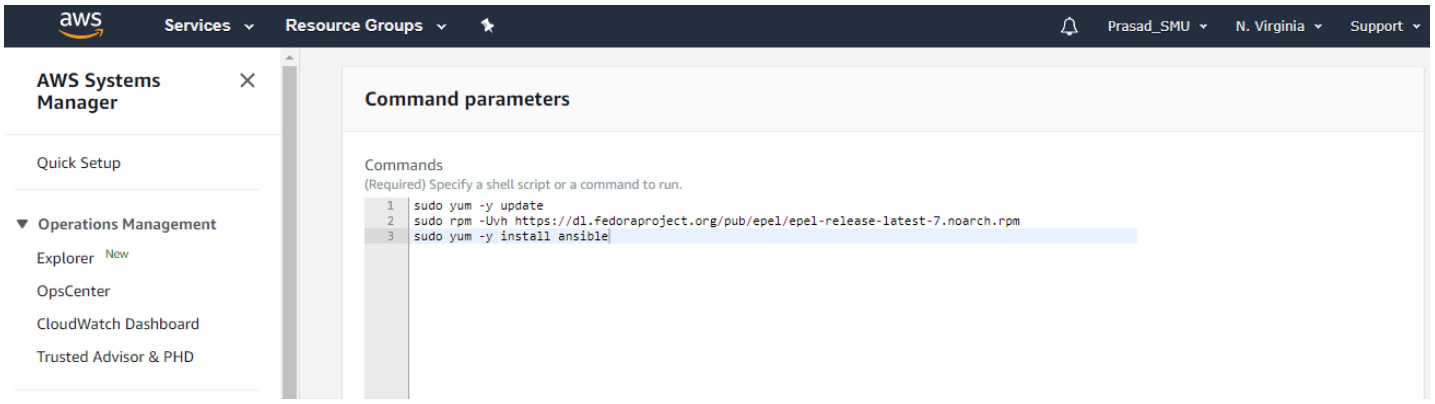


Instances

✓	Name	Instance ID	Instance state	Availability zone	Ping status	Last pin
✓	SSM	i-0a25233378569135c	running	us-east-1a	Online	02/05/2 02:30:51 (Central Time)

Type the below Script/Commands under Command Parameters. I've provided the Script in Text File.

This script does the Ansible Installation on the Target Instance.



Command parameters

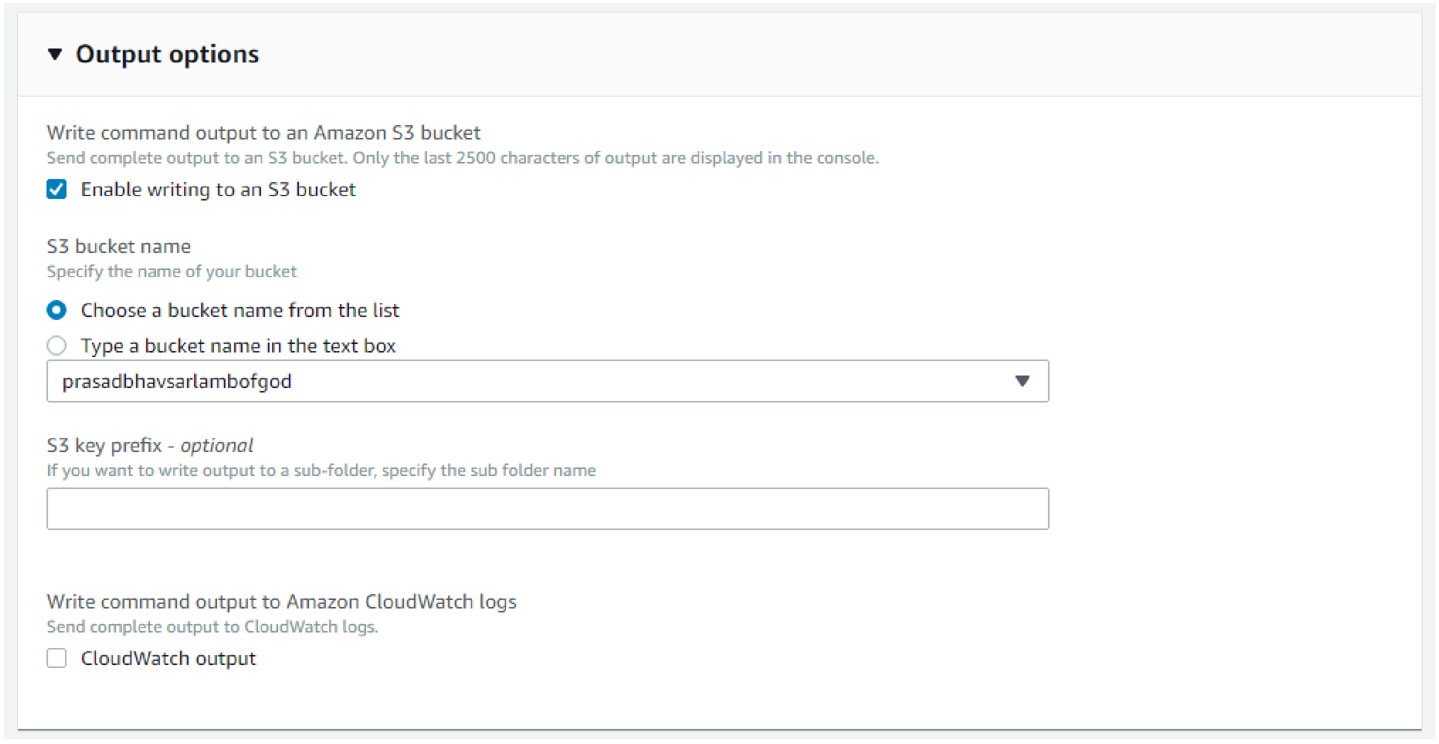
Commands
(Required) Specify a shell script or a command to run.

```
sudo yum -y update
sudo rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
sudo yum -y install ansible
```

You can now specify the S3 Bucket Name wherein Systems Manager logs will be saved.

Logs in the S3 Bucket will be saved in stdout.txt and stderr.txt format.

Stderr.txt file is quite useful if the Ansible Installation fails.



Output options

Write command output to an Amazon S3 bucket
Send complete output to an S3 bucket. Only the last 2500 characters of output are displayed in the console.
✓ Enable writing to an S3 bucket

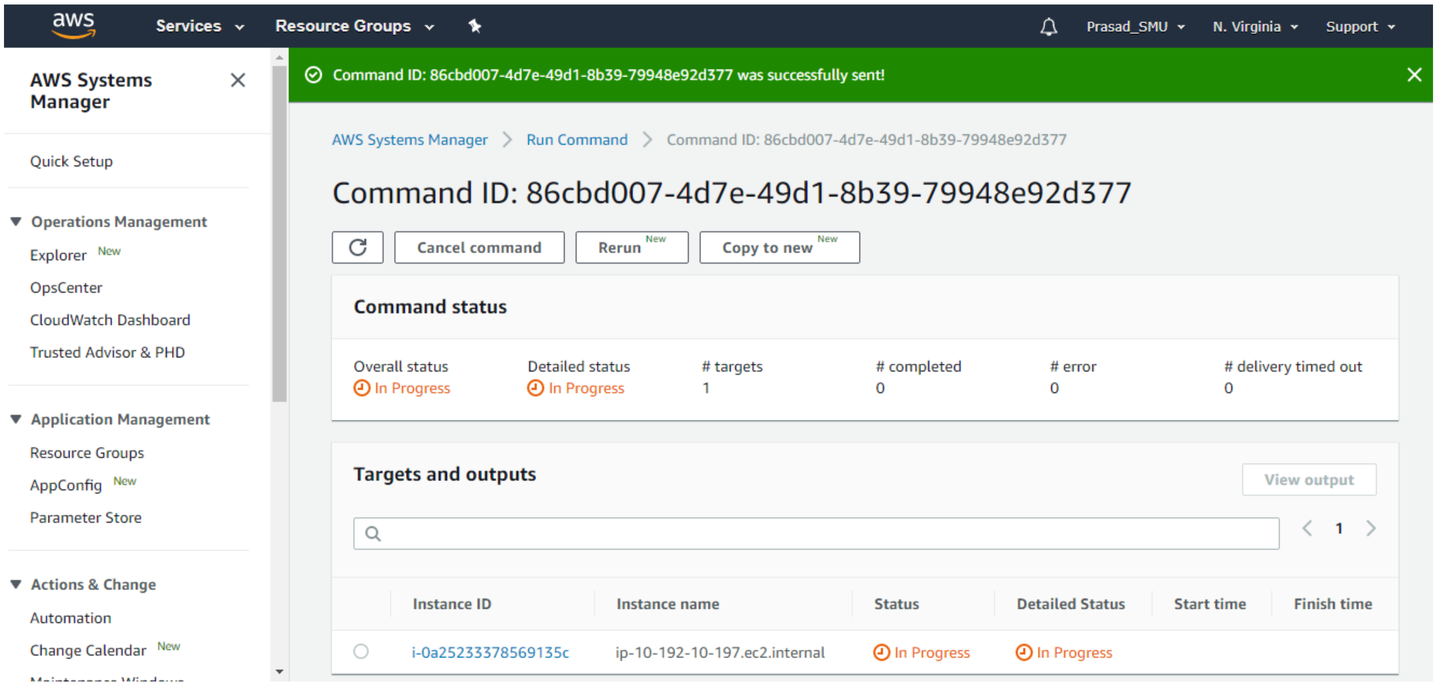
S3 bucket name
Specify the name of your bucket
● Choose a bucket name from the list
○ Type a bucket name in the text box
prasadbhavsarlambofgod

S3 key prefix - *optional*
If you want to write output to a sub-folder, specify the sub folder name

Write command output to Amazon CloudWatch logs
Send complete output to CloudWatch logs.
☐ CloudWatch output

Task 6: Ansible Installation Check

Make a note of Command ID and keep observing Overall Status.



Command ID: 86cbd007-4d7e-49d1-8b39-79948e92d377 was successfully sent!

Command ID: 86cbd007-4d7e-49d1-8b39-79948e92d377

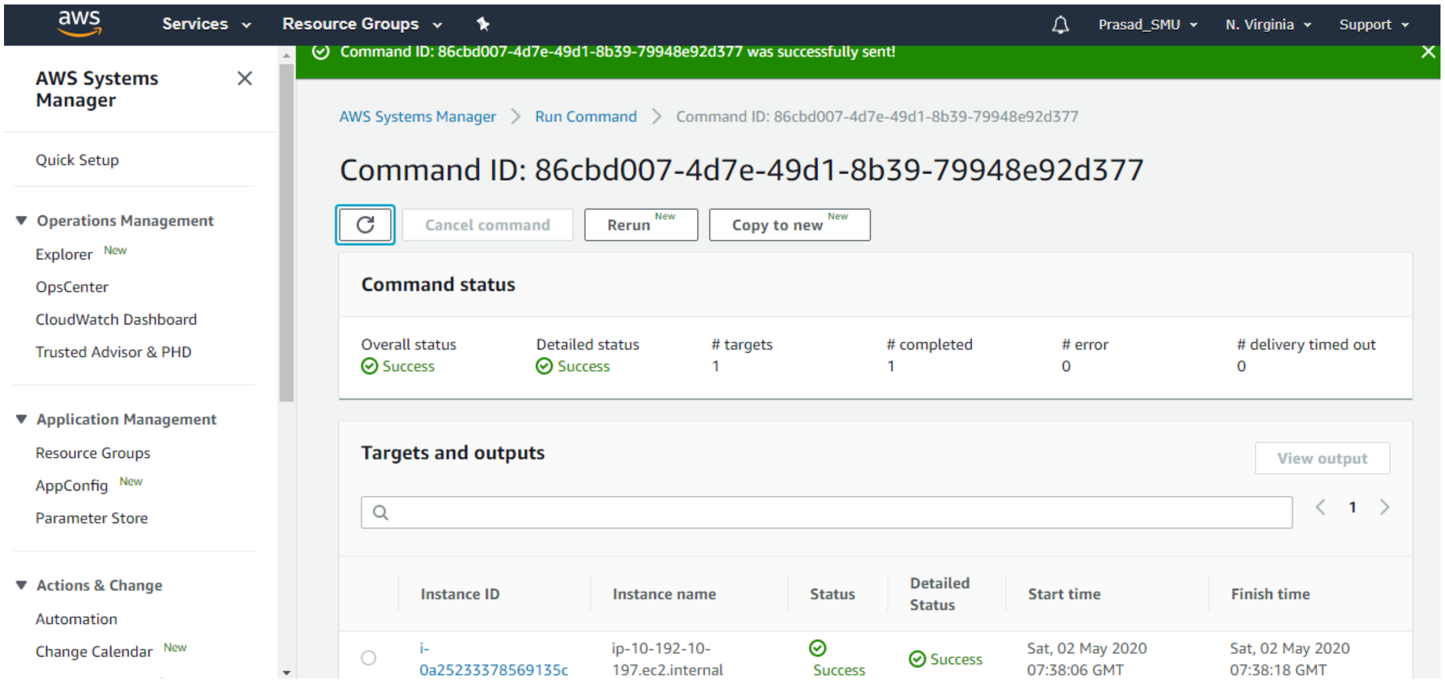
Command status

Overall status	Detailed status	# targets	# completed	# error	# delivery timed out
In Progress	In Progress	1	0	0	0

Targets and outputs

Instance ID	Instance name	Status	Detailed Status	Start time	Finish time
i-0a25233378569135c	ip-10-192-10-197.ec2.internal	In Progress	In Progress		

Status changes to **SUCCESS.**



Command ID: 86cbd007-4d7e-49d1-8b39-79948e92d377 was successfully sent!

Command ID: 86cbd007-4d7e-49d1-8b39-79948e92d377

Command status

Overall status	Detailed status	# targets	# completed	# error	# delivery timed out
Success	Success	1	1	0	0

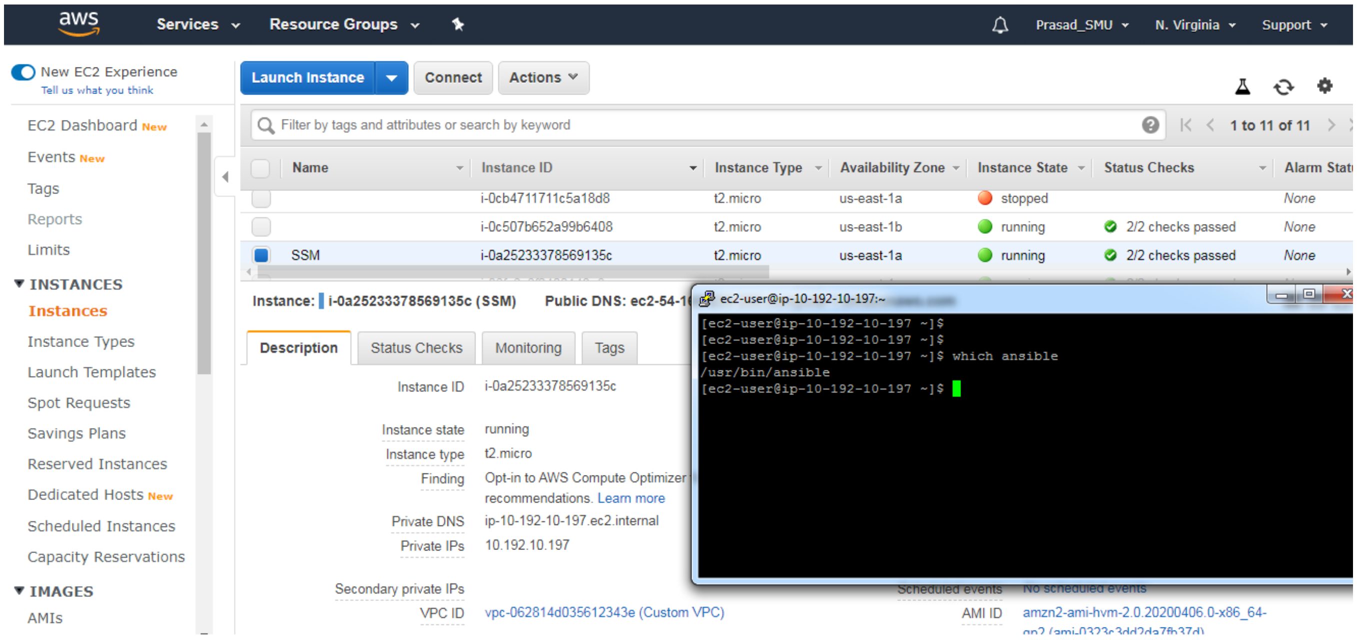
Targets and outputs

Instance ID	Instance name	Status	Detailed Status	Start time	Finish time
i-0a25233378569135c	ip-10-192-10-197.ec2.internal	Success	Success	Sat, 02 May 2020 07:38:06 GMT	Sat, 02 May 2020 07:38:18 GMT

Take SSH session of Target Instance.

Issue the below commands.

Command: which ansible



	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
☐		i-0cb4711711c5a18d8	t2.micro	us-east-1a	stopped		None
☐		i-0c507b652a99b6408	t2.micro	us-east-1b	running	2/2 checks passed	None
■	SSM	i-0a25233378569135c	t2.micro	us-east-1a	running	2/2 checks passed	None

Instance: i-0a25233378569135c (SSM)

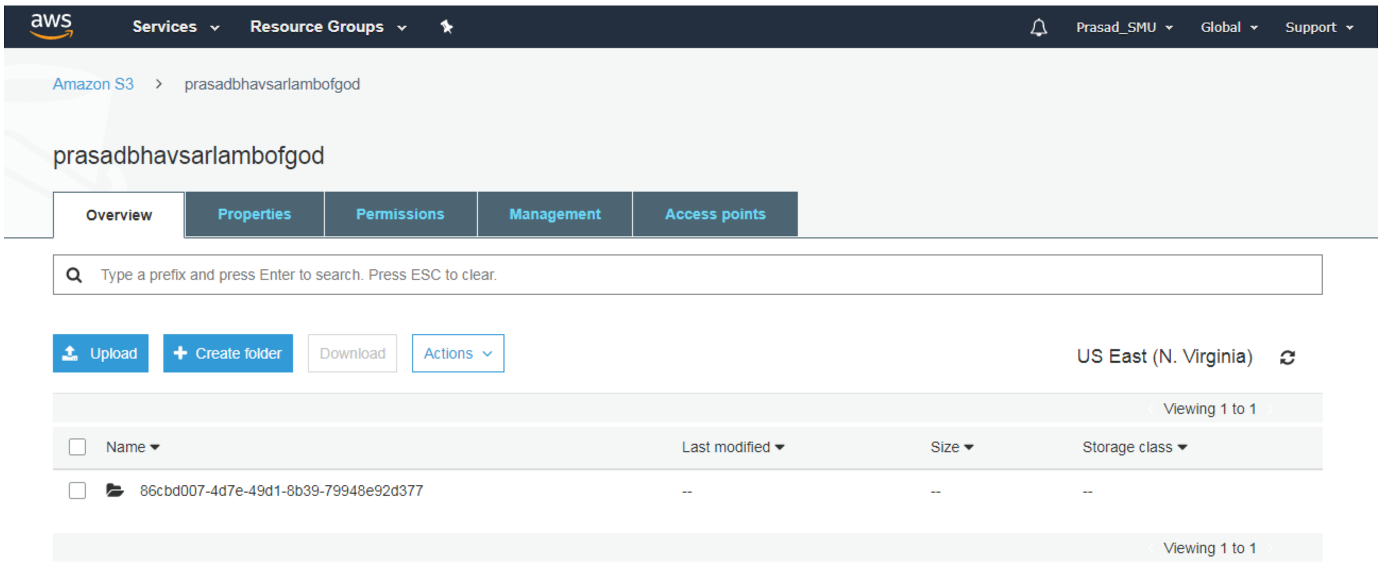
Instance ID	i-0a25233378569135c
Instance state	running
Instance type	t2.micro
Finding	Opt-in to AWS Compute Optimizer recommendations. Learn more
Private DNS	ip-10-192-10-197.ec2.internal
Private IPs	10.192.10.197
Secondary private IPs	
VPC ID	vpc-062814d035612343e (Custom VPC)
AMI ID	amzn2-ami-hvm-2.0.20200406.0-x86_64-gp2 (ami-0323c3dd2da7fb37d)

```
[ec2-user@ip-10-192-10-197 ~]$
[ec2-user@ip-10-192-10-197 ~]$
[ec2-user@ip-10-192-10-197 ~]$ which ansible
/usr/bin/ansible
[ec2-user@ip-10-192-10-197 ~]$
```

Ansible has been successfully installed on the Target Instance.

Now navigate to S3 Service and click on your S3 Bucket.

You'll notice that a new object for SSM logs have been created.



Amazon S3 > prasadbhavsarlambofgod

prasadbhavsarlambofgod

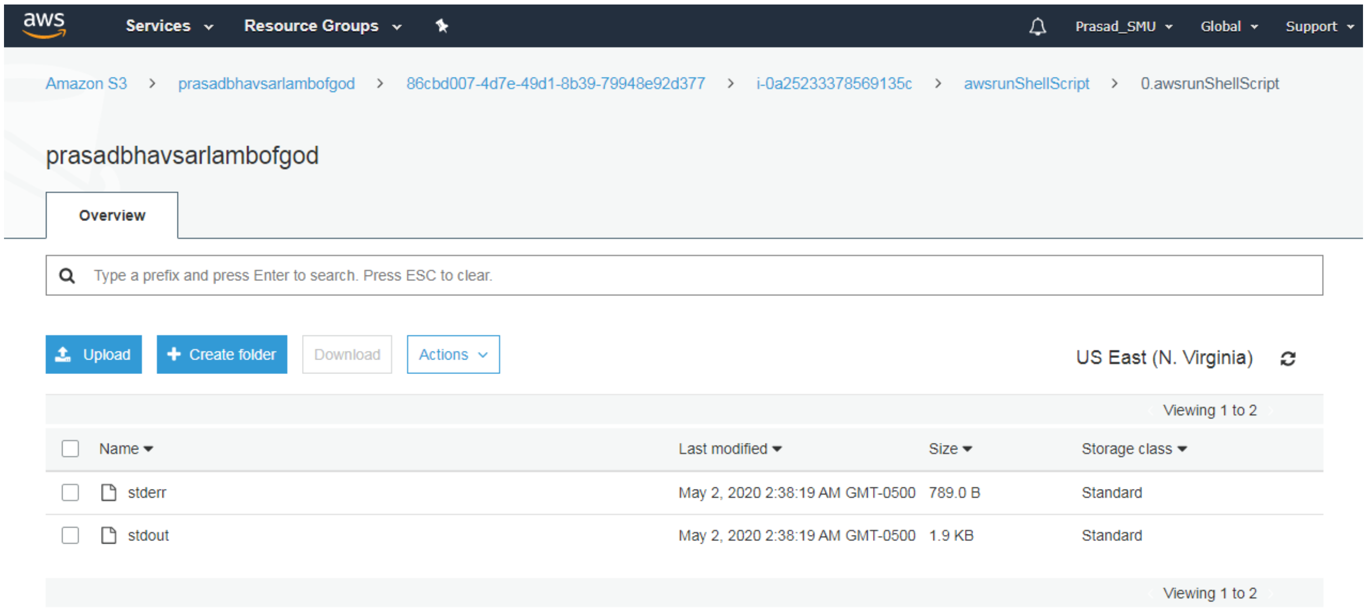
US East (N. Virginia)

Viewing 1 to 1

	Name	Last modified	Size	Storage class
☐	86cbd007-4d7e-49d1-8b39-79948e92d377	--	--	--

Viewing 1 to 1

Download the stderr.txt and stdout.txt if you want to check the SSM Logs.


aws
Services
Resource Groups
Prasad_SMU
Global
Support
Amazon S3 > prasadbhavsarlambofgod > 86cbd007-4d7e-49d1-8b39-79948e92d377 > i-0a25233378569135c > awsrunShellScript > 0.awsrunShellScript
Overview
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
Download
Actions
US East (N. Virginia)
Viewing 1 to 2

prasadbhavsarlambofgod

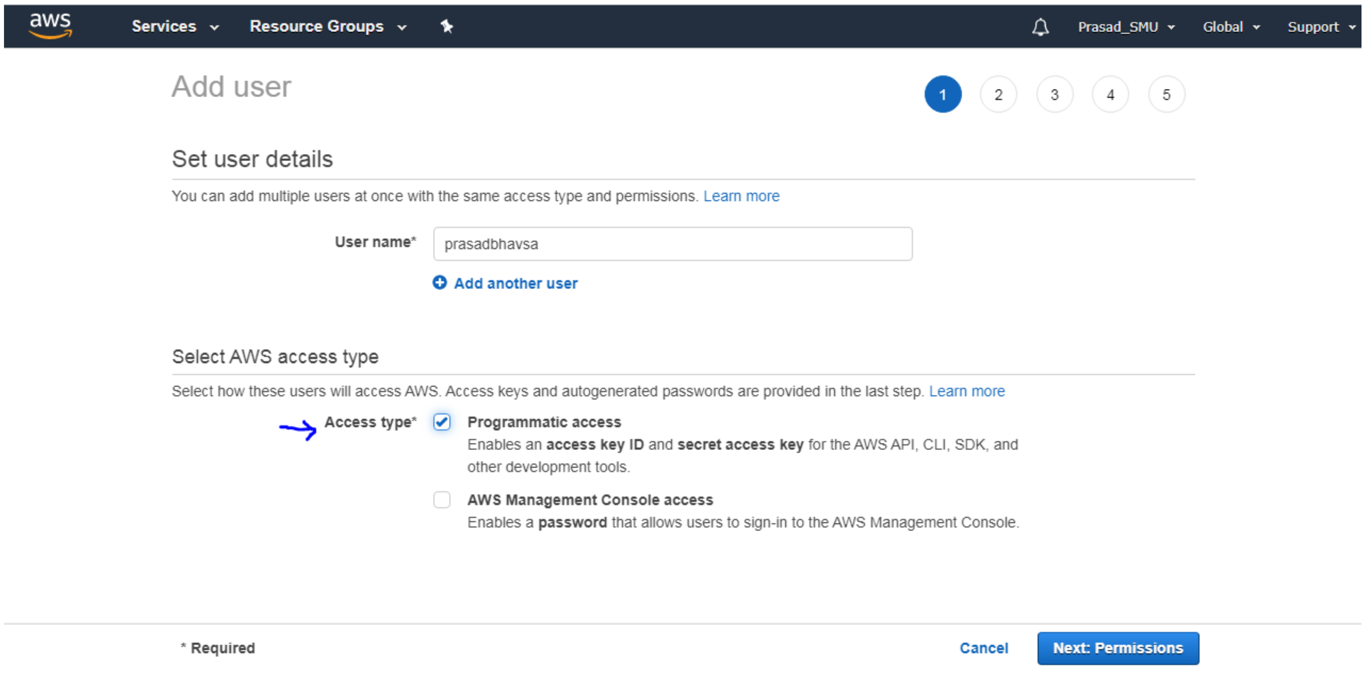
Name	Last modified	Size	Storage class
stderr	May 2, 2020 2:38:19 AM GMT-0500	789.0 B	Standard
stdout	May 2, 2020 2:38:19 AM GMT-0500	1.9 KB	Standard

Task 7: Create an IAM User

Navigate to IAM Service.

On left-hand side, click on Users and click on Add User.

Give the User name as per your choice and select the Access Type as **Programmatic Access**.



Add user

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* prasadbhavsa

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type* [✓] Programmatic access
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

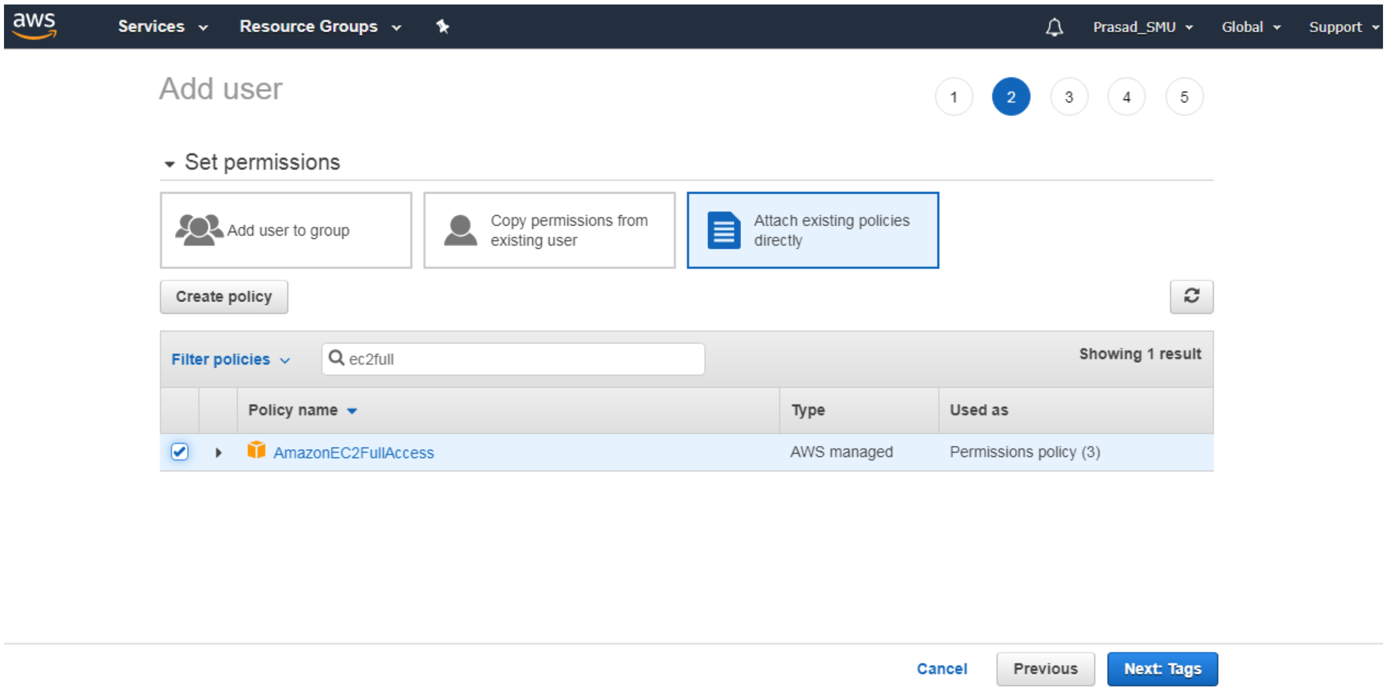
[ ] AWS Management Console access
Enables a **password** that allows users to sign-in to the AWS Management Console.

* Required

Click on Next: Permissions.

Click on **Attach existing policies directly**.

Search and select **AmazonEC2FullAccess** Policy.



ec2full

Showing 1 result

		Policy name	Type	Used as
✓	▸	AmazonEC2FullAccess	AWS managed	Permissions policy (3)

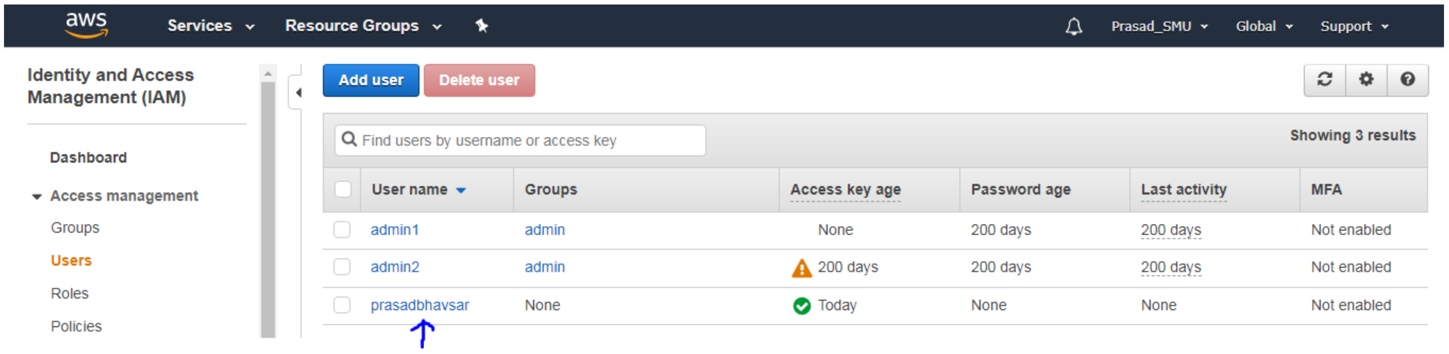
Click on Next: Tags.

You can add Tags if you wish else click Next: Review.

Review the configurations and click on Create User.

Make sure to note down the Access Key ID and Secret Access Key. You can also download the .csv file for a safe side. We will need these keys while doing boto configurations.

Click on Create User. User has been created successfully.



Showing 3 results

	User name	Groups	Access key age	Password age	Last activity	MFA
	admin1	admin	None	200 days	200 days	Not enabled
	admin2	admin	⚠ 200 days	200 days	200 days	Not enabled
	prasadbhavsar	None	✓ Today	None	None	Not enabled

Task 8: PIP, BOTO Configurations

Take SSH Session of the Ansible Controller EC2 Instance.

Python will be pre-installed on the Linux EC2 Instances.

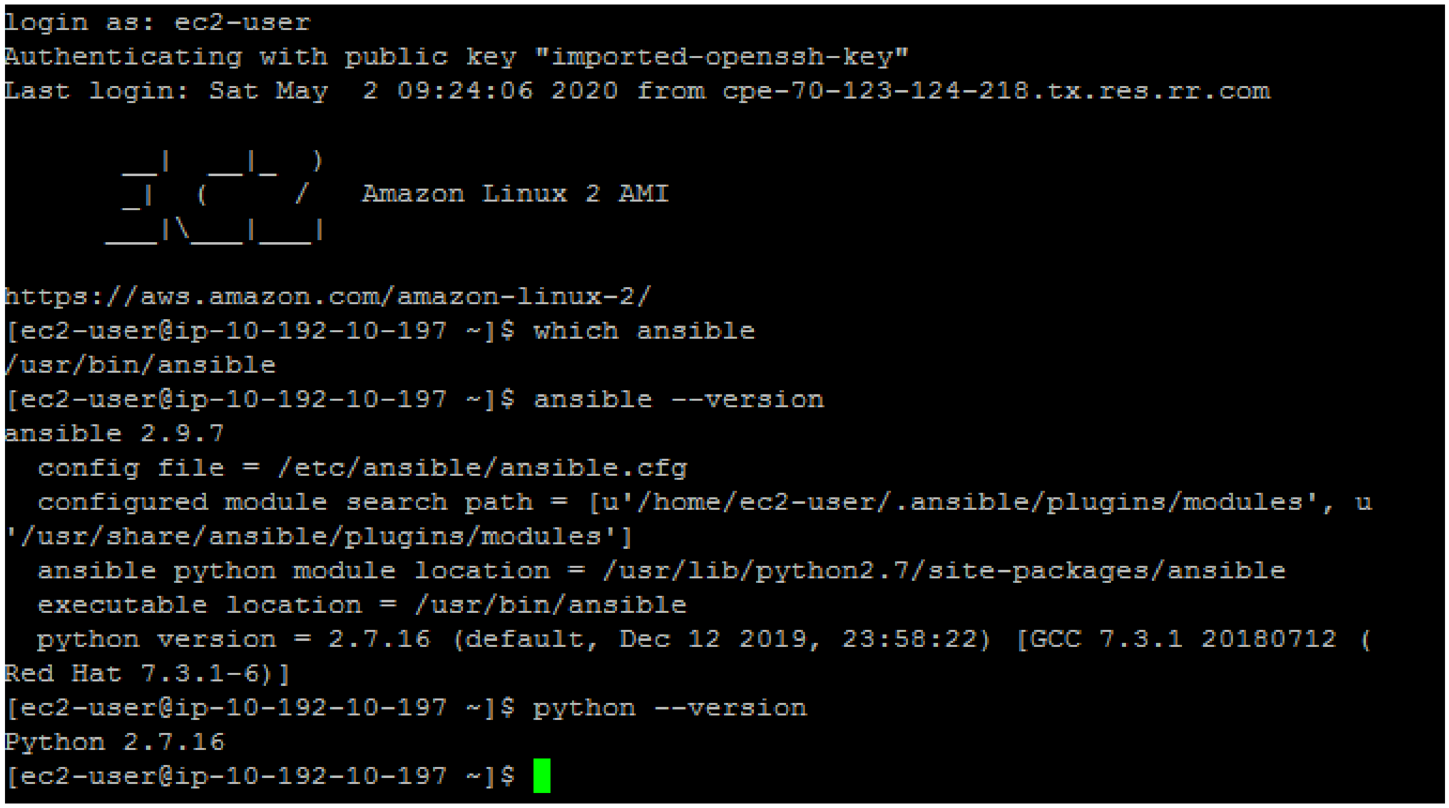
Make sure the Ansible and Python is Installed on the EC2 Instance. Run the below commands.

Commands:

which ansible

ansible –version

python --version

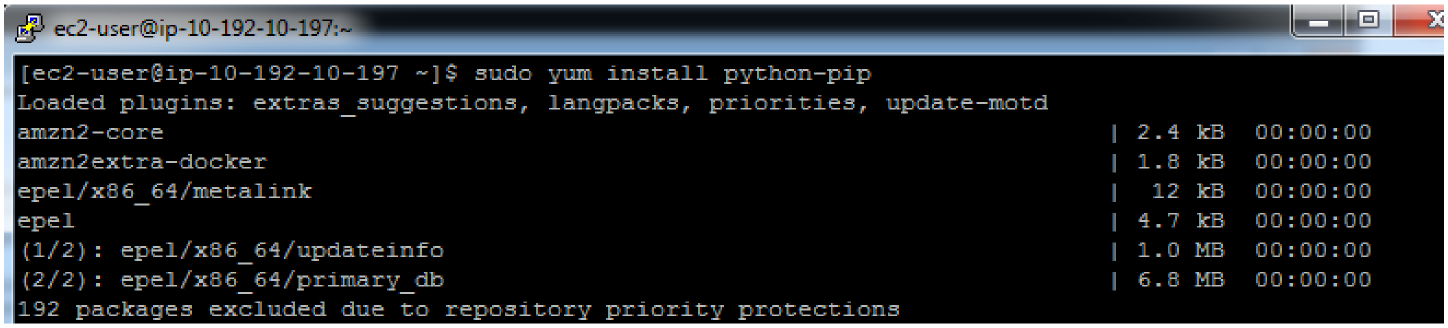


```
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Last login: Sat May  2 09:24:06 2020 from cpe-70-123-124-218.tx.res.rr.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-192-10-197 ~]$ which ansible
/usr/bin/ansible
[ec2-user@ip-10-192-10-197 ~]$ ansible --version
ansible 2.9.7
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/ec2-user/.ansible/plugins/modules', u
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.16 (default, Dec 12 2019, 23:58:22) [GCC 7.3.1 20180712 (
Red Hat 7.3.1-6)]
[ec2-user@ip-10-192-10-197 ~]$ python --version
Python 2.7.16
[ec2-user@ip-10-192-10-197 ~]$
```

To install BOTO, we would need a Python Module "PIP".

To install Python Module "PIP", run the below command.

Command: sudo yum install python-pip

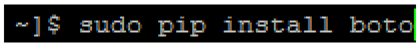


```
[ec2-user@ip-10-192-10-197 ~]$ sudo yum install python-pip
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                               | 2.4 kB  00:00:00
amzn2extra-docker                                        | 1.8 kB  00:00:00
epel/x86_64/metalink                                     |  12 kB  00:00:00
epel                                                     | 4.7 kB  00:00:00
(1/2): epel/x86_64/updateinfo                            | 1.0 MB  00:00:00
(2/2): epel/x86_64/primary_db                            | 6.8 MB  00:00:00
192 packages excluded due to repository priority protections
```

Once PIP is installed, we will now install BOTO.

To install BOTO, run the below command.

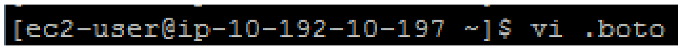
Command: sudo pip install boto



```
~]$ sudo pip install boto
```

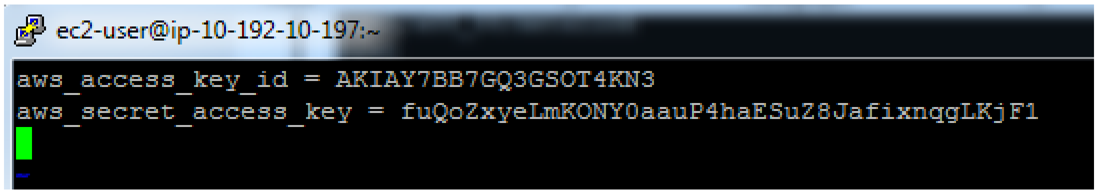
Now create a ".boto" file in your Home Directory.

Command: vi .boto



```
[ec2-user@ip-10-192-10-197 ~]$ vi .boto
```

Put the **aws_access_key_id** and **aws_secret_access_key** which you copied in Task 7 as follows.



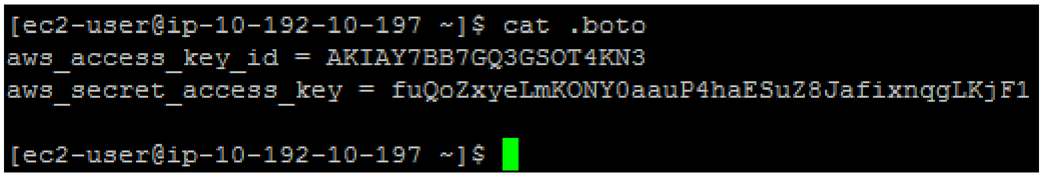
```
aws_access_key_id = AKIAY7BB7GQ3GSOT4KN3
aws_secret_access_key = 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
```

Save the vi editor by issuing below command.

Command: :wq!

Review the BOTO file again, issue the following command.

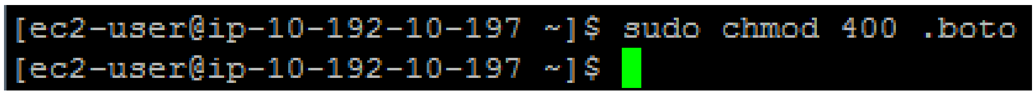
Command: cat .boto



```
[ec2-user@ip-10-192-10-197 ~]$ cat .boto
aws_access_key_id = AKIAY7BB7GQ3GSOT4KN3
aws_secret_access_key = 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

[ec2-user@ip-10-192-10-197 ~]$
```

Now save the .boto file with the permission 400.

Command: sudo chmod 400 .boto



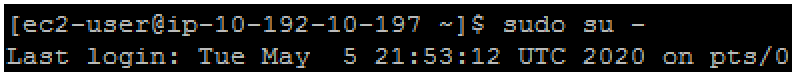
```
[ec2-user@ip-10-192-10-197 ~]$ sudo chmod 400 .boto
[ec2-user@ip-10-192-10-197 ~]$
```

PIP and BOTO configurations are now completed.

Task 9: Writing an Ansible Playbook

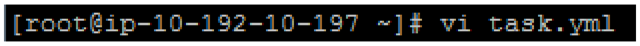
On the Ansible Controller EC2 Instance, open the vi editor to write an Ansible Playbook.

Switch User to ROOT.

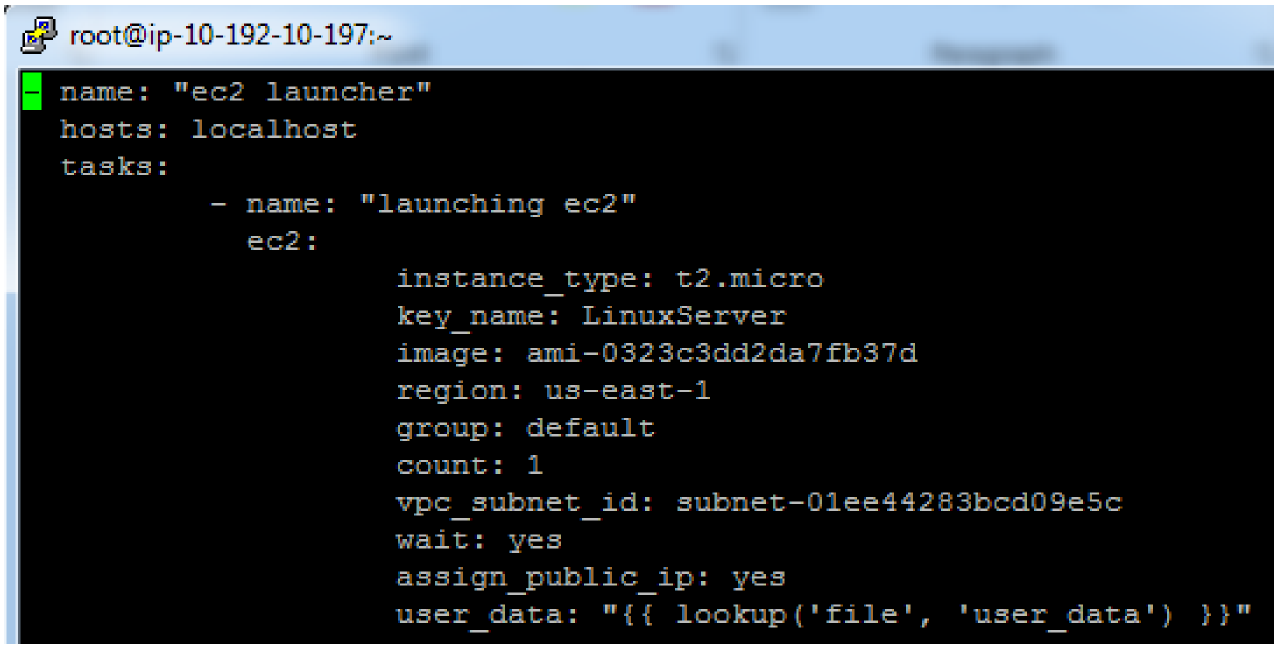


```
[ec2-user@ip-10-192-10-197 ~]$ sudo su -
Last login: Tue May  5 21:53:12 UTC 2020 on pts/0
```

Write the Ansible Playbook as below. Make sure of Indentation.



```
[root@ip-10-192-10-197 ~]# vi task.yml
```



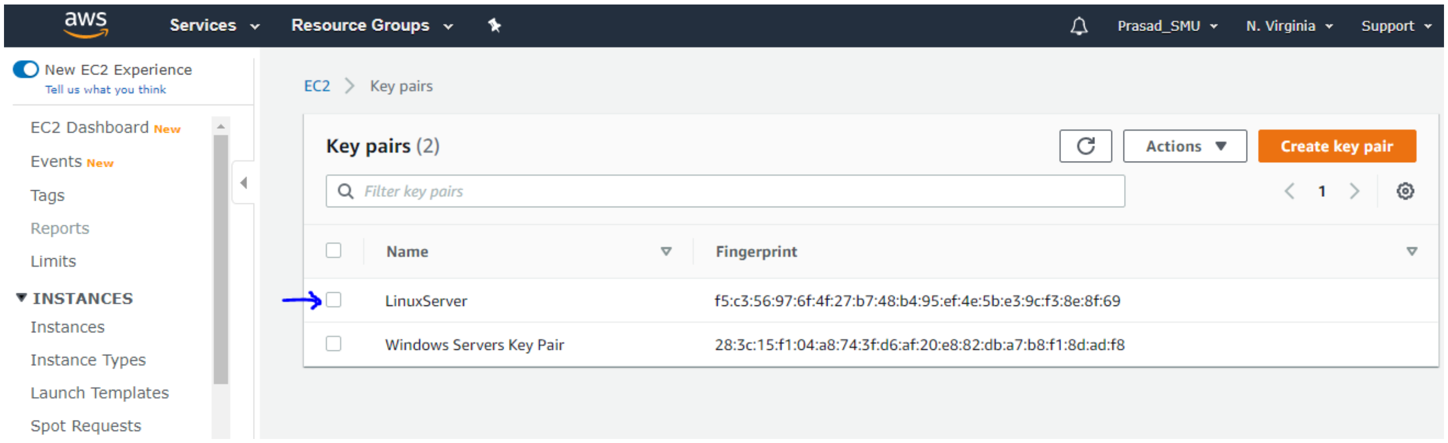
root@ip-10-192-10-197:~

```
- name: "ec2 launcher"
  hosts: localhost
  tasks:
          - name: "launching ec2"
            ec2:
                    instance_type: t2.micro
                    key_name: LinuxServer
                    image: ami-0323c3dd2da7fb37d
                    region: us-east-1
                    group: default
                    count: 1
                    vpc_subnet_id: subnet-01ee44283bcd09e5c
                    wait: yes
                    assign_public_ip: yes
                    user_data: "{{ lookup('file', 'user_data') }}"
```

Question: How to get key_name, image, group, vpc_subnet_id parameters?

key_name:

Since I've my entire architecture deployed on N. Virginia (US-EAST-1) region, I've created few Key Pairs for the previous labs. You can find the Key Pair information on the EC2 Service Dashboard. You can also create Key Pair if you wish. I've selected LinuxServer Key Pair.

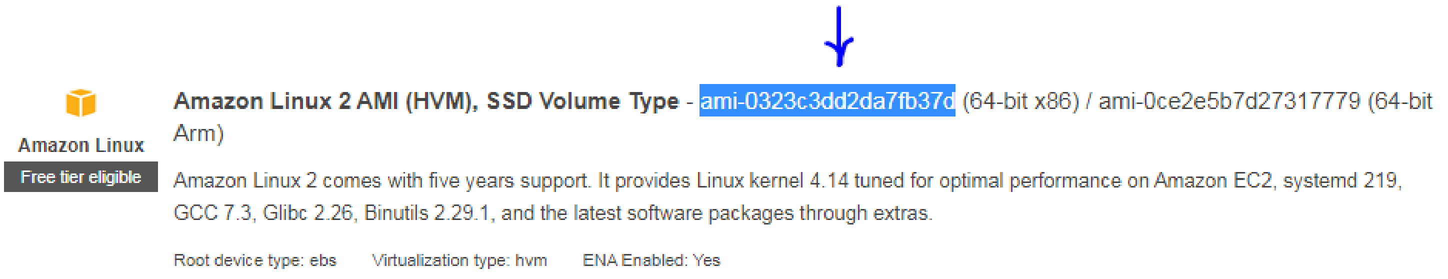


Key pairs (2)

Name	Fingerprint
LinuxServer	f5:c3:56:97:6f:4f:27:b7:48:b4:95:ef:4e:5b:e3:9c:f3:8e:8f:69
Windows Servers Key Pair	28:3c:15:f1:04:a8:74:3f:d6:af:20:e8:82:db:a7:b8:f1:8d:ad:f8

Image:

AMI Image information can be found while launching an EC2 Instance.



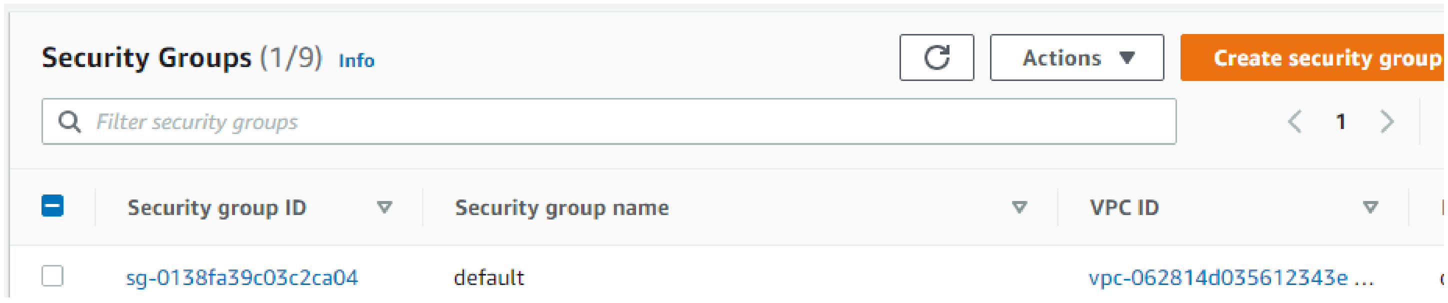
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0323c3dd2da7fb37d (64-bit x86) / ami-0ce2e5b7d27317779 (64-bit Arm)

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.

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

Group:

Security Group information can be found on the EC2 Service Dashboard. Select the Security Group as per your choice. I've selected the Default Security Group.



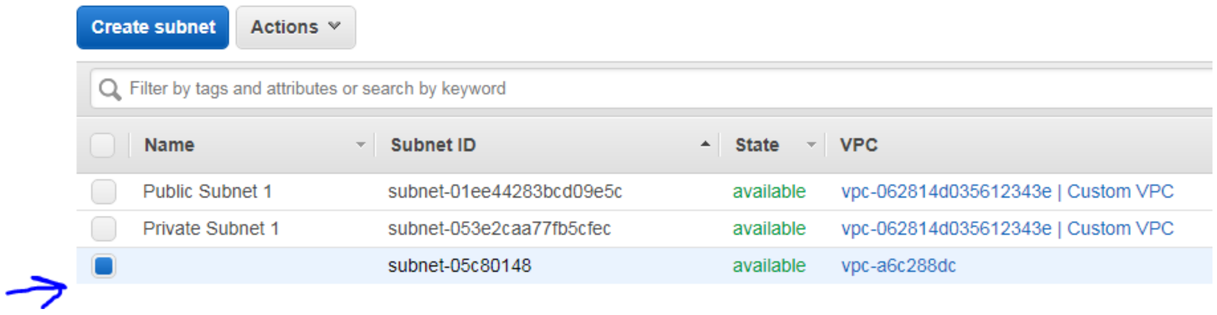
Security Groups (1/9)

Security group ID	Security group name	VPC ID
sg-0138fa39c03c2ca04	default	vpc-062814d035612343e ...

<u>vpc_subnet_id:</u>

Navigate to VPC Service and click on Subnet.

Copy the Subnet ID in which you want to deploy an EC2 Instance. I've selected the Default Subnet of Default VPC.



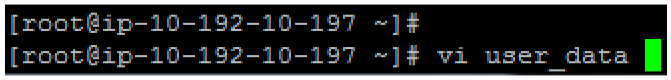
Name	Subnet ID	State	VPC
Public Subnet 1	subnet-01ee44283bcd09e5c	available	vpc-062814d035612343e | Custom VPC
Private Subnet 1	subnet-053e2caa77fb5cfec	available	vpc-062814d035612343e | Custom VPC
	subnet-05c80148	available	vpc-a6c288dc

Keep in mind that, these parameters key_name, image, group, vpc_subnet_id changes per Region. If you want to deploy EC2 Instance in another Region, make sure to associate parameters of that particular Region.

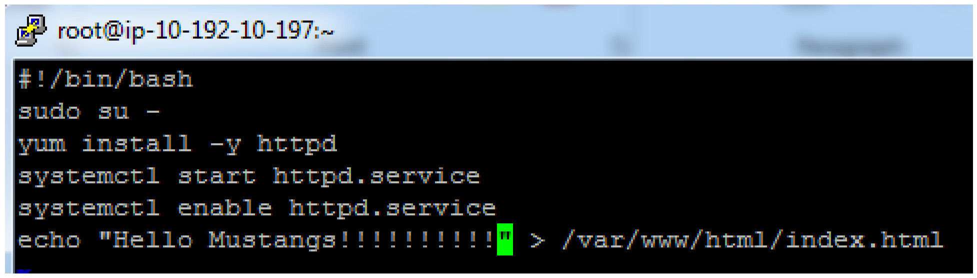
Now let's save our Playbook.

Command: : wq!

Let's create a User Data file at /root directory which will bootstrap the specified commands while launching an EC2 Instance.



```
[root@ip-10-192-10-197 ~]#
[root@ip-10-192-10-197 ~]# vi user_data
```



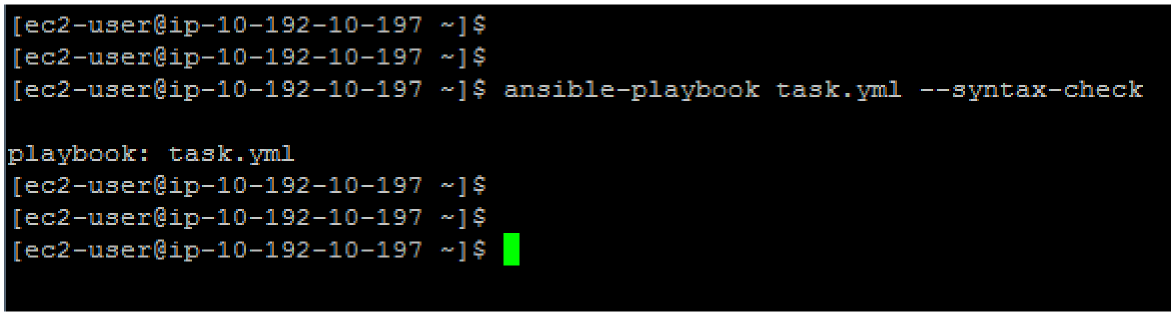
```
#!/bin/bash
sudo su -
yum install -y httpd
systemctl start httpd.service
systemctl enable httpd.service
echo "Hello Mustangs!!!!!!!!!!" > /var/www/html/index.html
```

Now let's save the User Data file.

Command: : wq!

Let's do the Syntax check of the written Ansible Playbook. If the Playbook Syntax is correct, it will return the same Playbook name.

Command: ansible-playbook task.yml --syntax-check



```
[ec2-user@ip-10-192-10-197 ~]$
[ec2-user@ip-10-192-10-197 ~]$
[ec2-user@ip-10-192-10-197 ~]$ ansible-playbook task.yml --syntax-check

playbook: task.yml
[ec2-user@ip-10-192-10-197 ~]$
[ec2-user@ip-10-192-10-197 ~]$
[ec2-user@ip-10-192-10-197 ~]$
```

Task 10: Execute Ansible Playbook on EC2 Instance

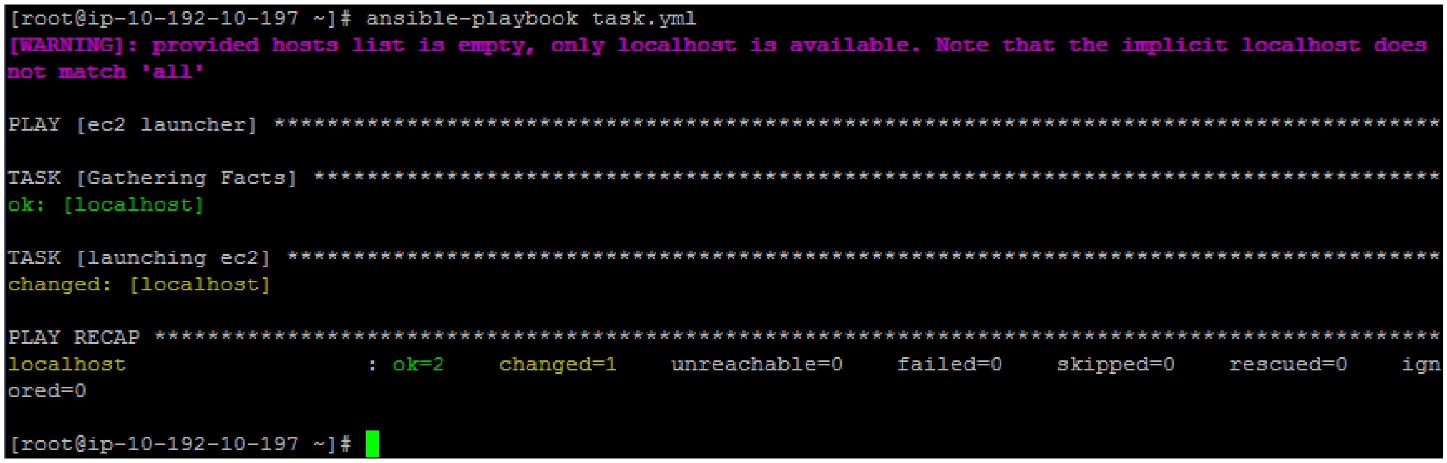
Now switch to ROOT User.

Command: sudo su -

Now, to execute the Playbook, run below command.

Command:

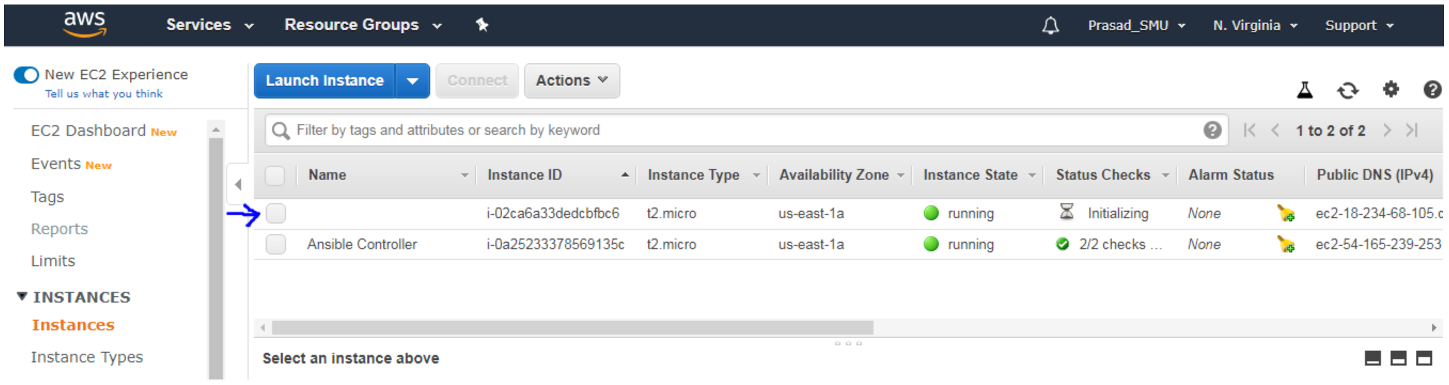
ansible-playbook task.yml



```
[root@ip-10-192-10-197 ~]# ansible-playbook task.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does
not match 'all'

PLAY [ec2 launcher] ***************************************************************************************

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

TASK [launching ec2] **************************************************************************************
changed: [localhost]

PLAY RECAP ************************************************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ign
ored=0

[root@ip-10-192-10-197 ~]#
```

You can see that the Ansible Playbook has been executed successfully.

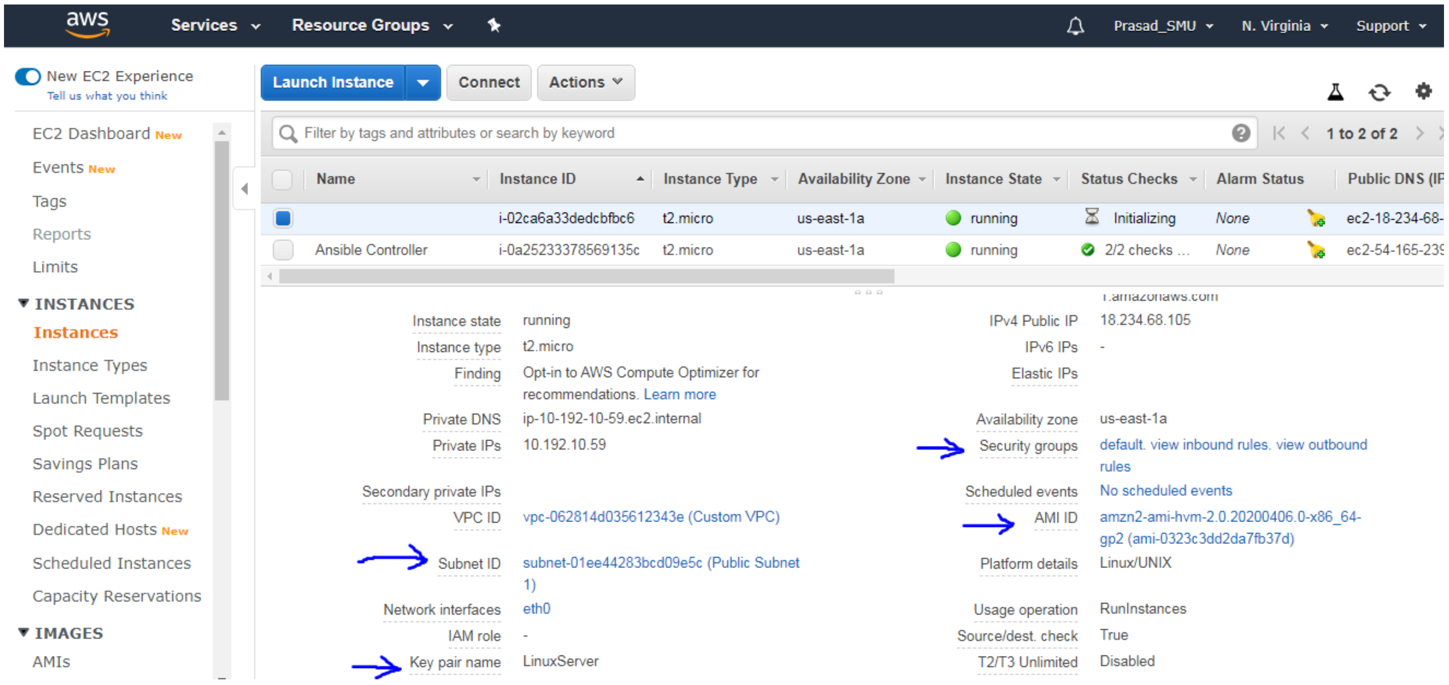
Navigate to the EC2 Service, you'll observe that a new Instance has been launched.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
	i-02ca6a33dedcbfbc6	t2.micro	us-east-1a	running	Initializing	None	ec2-18-234-68-105.c
Ansible Controller	i-0a25233378569135c	t2.micro	us-east-1a	running	2/2 checks ...	None	ec2-54-165-239-253

Select an instance above

You can now verify its parameters such as key_name, image, group, vpc_subnet_id which we defined in our Playbook.

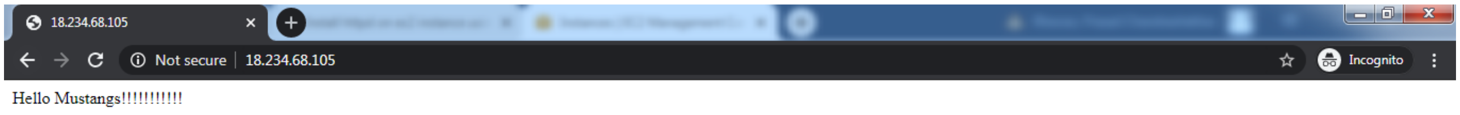


Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IF
	i-02ca6a33dedcbfbc6	t2.micro	us-east-1a	running	Initializing	None	ec2-18-234-68-
Ansible Controller	i-0a25233378569135c	t2.micro	us-east-1a	running	2/2 checks ...	None	ec2-54-165-23

Instance state	running	IPv4 Public IP	18.234.68.105
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	
Private DNS	ip-10-192-10-59.ec2.internal	Availability zone	us-east-1a
Private IPs	10.192.10.59	Security groups	default. view inbound rules. view outbound rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-062814d035612343e (Custom VPC)	AMI ID	amzn2-ami-hvm-2.0.20200406.0-x86_64-gp2 (ami-0323c3dd2da7fb37d)
Subnet ID	subnet-01ee44283bcd09e5c (Public Subnet 1)	Platform details	Linux/UNIX
Network interfaces	eth0	Usage operation	RunInstances
IAM role	-	Source/dest. check	True
Key pair name	LinuxServer	T2/T3 Unlimited	Disabled

Now copy the Public IP of the EC2 Instance and Paste it in your Browser.

If the Apache Packages has been successfully installed in your EC2 Instance then it should return the Web Page with Header as “Hello Mustangs!!!!!!”.



Hello Mustangs!!!!!!!!!!

This completes the Lab on Why Ansible on AWS is more powerful than AWS CloudFormation?

For Questions, contact me on pbhavsar@smu.edu .