

Launch EC2 Instance using Ansible on AWS



In the previous lab, we've used AWS Systems Manager Service to Install Ansible on the Target Instance and ran Ansible Playbook. 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 in our environment.

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.

The screenshot shows the 'Create role' page in the AWS IAM console. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information 'Prasad_SMU'. A 'Console Home' button is on the left. The main heading is 'Create role' with a progress indicator showing four steps, with the first step being active. Below the heading is the section 'Select type of trusted entity' with four options: 'AWS service' (selected), 'Another AWS account', 'Web identity', and 'SAML 2.0 federation'. The 'AWS service' option is described as 'EC2, Lambda and others' and 'Allows AWS services to perform actions on your behalf'. Below this is the 'Choose a use case' section, which lists 'Common use cases' with 'EC2' selected (described as 'Allows EC2 instances to call AWS services on your behalf') and 'Lambda' (described as 'Allows Lambda functions to call AWS services on your behalf'). There is also a section 'Or select a service to view its use cases' with a grid of services including API Gateway, CodeDeploy, EMR, KMS, RoboMaker, AWS Backup, CodeGuru, ElastiCache, Kinesis, and S3. At the bottom, there is a '* Required' label, a 'Cancel' button, and a 'Next: Permissions' button.

Select the below three Default IAM Policies:

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

 Services ▾ Resource Groups ▾ ⚙

Prasad_SMU ▾ Global ▾ Support ▾

1 2 3 4

Filter policies ▾


Q SSM

Showing 13 results

Feedback  English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

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

 Services ▾ Resource Groups ▾ ⚙

Prasad_SMU ▾

1 2 3 4

Task 2: Launch & Configure EC2 Instances with SSM Agent

Navigate to EC2 Service and click on Launch Instance.

Select the **Amazon Linux AMI**.

The screenshot shows the AWS Management Console interface for the 'Launch Instance' wizard, specifically the 'Choose AMI' step. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information (Prasad_SMU, N. Virginia, Support). The wizard progress bar shows seven steps: 1. Choose AMI (active), 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group, and 7. Review. Below the progress bar, the title 'Step 1: Choose an Amazon Machine Image (AMI)' is displayed, followed by a brief explanation of AMIs. A search bar is present with the placeholder text 'Search for an AMI by entering a search term e.g. "Windows"'. On the left, a 'Quick Start' sidebar lists 'My AMIs', 'AWS Marketplace', and 'Community AMIs', with a 'Free tier only' filter. The main content area displays two AMIs: 'Amazon Linux 2 AMI (HVM), SSD Volume Type' (ami-0323c3dd2da7fb37d) and 'Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type' (ami-0915e09cc7ceee3ab). Both are marked as 'Free tier eligible'. The first AMI is selected, and its details, including supported architectures (64-bit x86 and 64-bit Arm), are shown on the right.

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.

The screenshot shows the 'Configure Instance Details' step of the EC2 instance launch wizard. The top navigation bar and wizard progress bar are consistent with the previous screenshot. The title 'Step 3: Configure Instance Details' is displayed, followed by a brief explanation of the step. The configuration options are as follows: 'Number of instances' is set to 1, with a 'Launch into Auto Scaling Group' link; 'Purchasing option' has 'Request Spot instances' unchecked; 'Network' is set to 'vpc-062814d035612343e | Custom VPC', with a 'Create new VPC' link; 'Subnet' is set to 'subnet-01ee44283bcd09e5c | Public Subnet 1 | us-e', with a 'Create new subnet' link and a note '245 IP Addresses available'; 'Auto-assign Public IP' is set to 'Use subnet setting (Enable)'; 'Placement group' has 'Add instance to placement group' unchecked; 'Capacity Reservation' is set to 'Open', with a 'Create new Capacity Reservation' link; and 'IAM role' is set to 'EC2-Role-SSM', with a 'Create new IAM role' link.

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.

▼ Advanced Details

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.

aws Services ▼ Resource Groups ▼ ★

Prasad_SMU ▼ N. Virginia ▼ Support ▼

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

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	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	✕
Env	Production	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	✕
OS	Linux	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	✕

[Add another tag](#) (Up to 50 tags maximum)

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:

Description:

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

Warning

Click on Review and Launch.

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

Click on Launch Instances.

Step 7: Review Instance Launch

Please review your instance launch details. You can't edit details after you launch your instance.

Improve your instances' security
 Your instances may be accessible from the Internet. You can also open additional ports in your security groups.

AMI Details
Red Hat Enterprise Linux 8 (HVM)
 Free tier eligible
 Root Device Type: ebs
 Virtualization type: x86_64

Instance Type

Instance Type	ECUs
t2.micro	Variable

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.

Network Performance
 Low to Moderate

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

The screenshot shows the AWS Management Console interface for EC2 instances. The left sidebar contains navigation links for EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES, and IMAGES. The main content area displays a table of instances. The 'Ansible Controller' instance is highlighted. Below the table, the details for the selected instance are shown, including its ID, state, type, DNS, and network configuration.

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**

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	Elastic IPs	-	Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Availability zone	us-east-1a	Private DNS	ip-10-192-10-197.ec2.internal	Security groups	RHEL8-SG , view inbound rules , view outbound rules
Private IPs	10.192.10.197	Scheduled events	No scheduled events	Private IPs	10.192.10.197		
Secondary private IPs							

Task 3: Create a S3 Bucket to store SSM Logs

Navigate to S3 Service.

Click on Create Bucket.

Give a unique Bucket Name as per your choice.

The screenshot shows the 'Create bucket' page in the AWS Management Console. The 'General configuration' section is visible, showing the bucket name 'prasadbhavsarlambogod' and the region 'US East (N. Virginia) us-east-1'. A note indicates that the bucket name must be unique and cannot contain spaces or uppercase letters.

Amazon S3 > Create bucket

Create bucket

General configuration

Bucket name
prasadbhavsarlambogod

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.

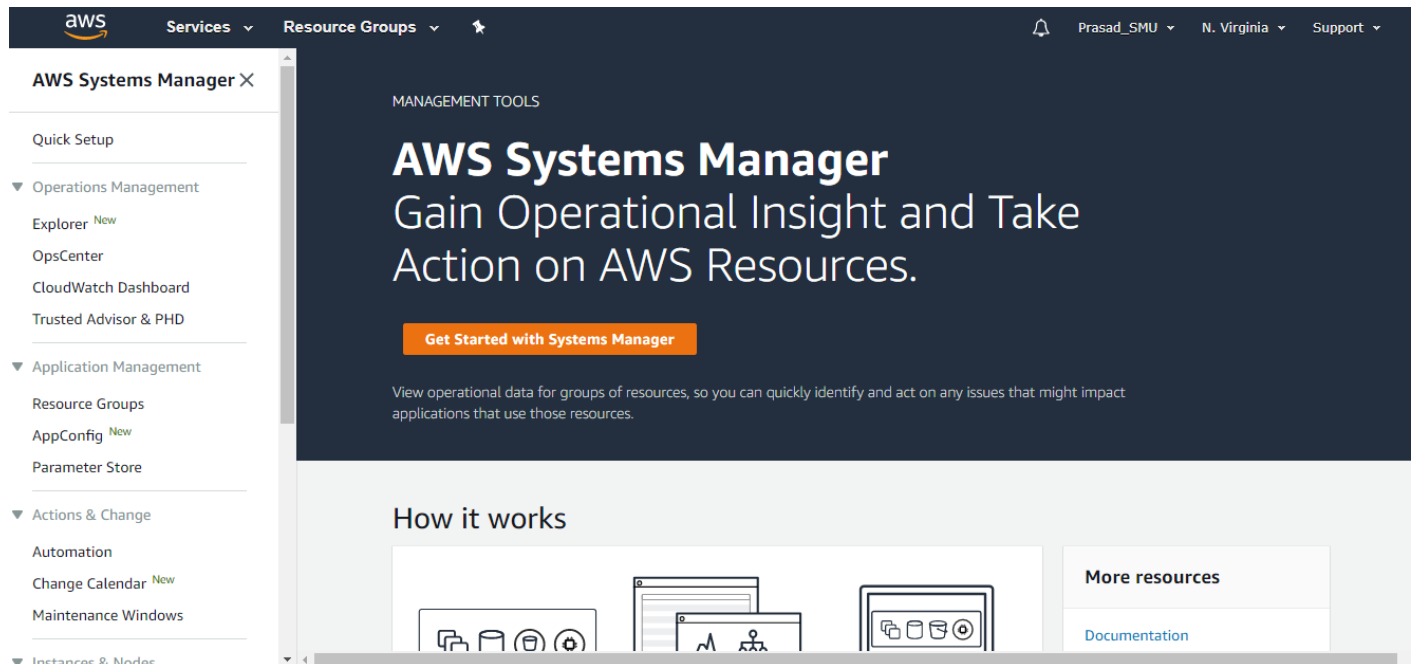
Click on Create. S3 Bucket has been successfully created to store Systems Manager (SSM) logs.

The screenshot shows the AWS Management Console interface for Amazon S3. On the left, there's a sidebar with 'Amazon S3' selected. The main content area shows 'Buckets (2)' with a search bar and a table of buckets. The table has columns: Name, Region, Access, and Bucket created. Two buckets are listed: 'cf-templates-umsgutrdp0mt-us-east-1' and 'prasadbhavsarlambofod'. The second bucket is selected. Above the table, there are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'.

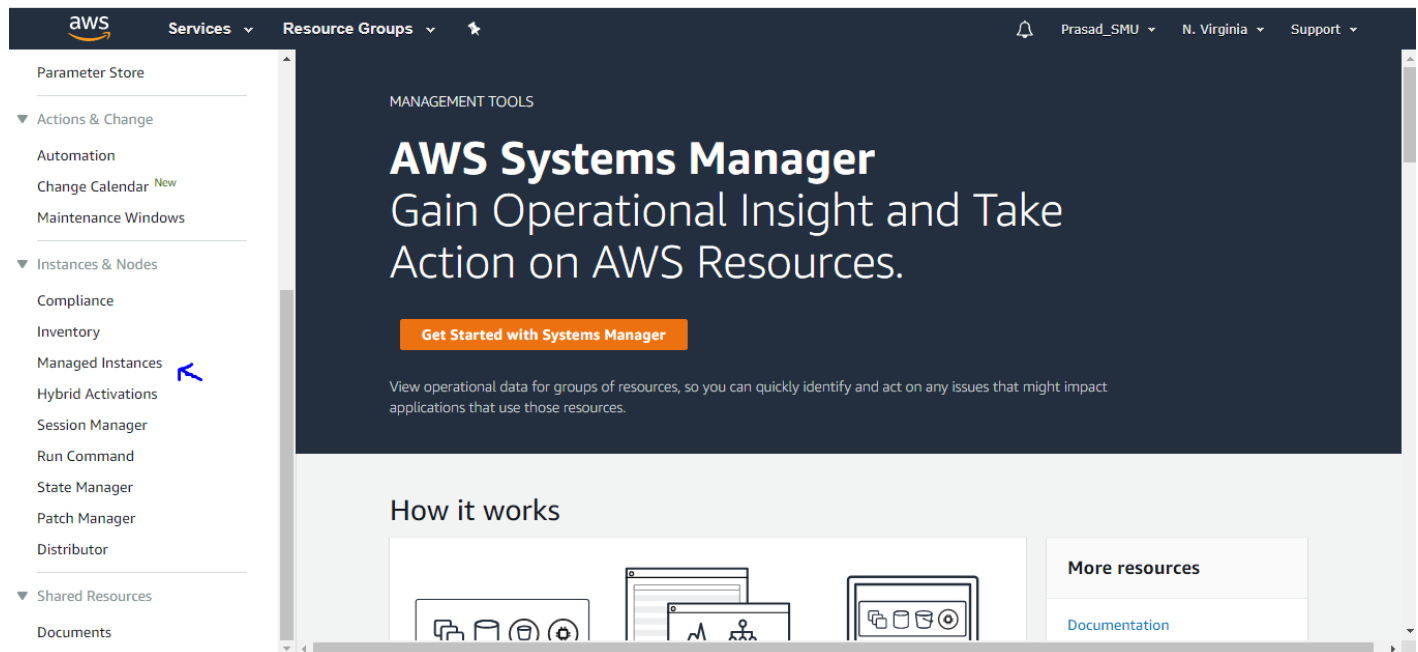
Name	Region	Access	Bucket created
cf-templates-umsgutrdp0mt-us-east-1	US East (N. Virginia) us-east-1	Objects can be public	2020-04-19T08:18:12.000Z
prasadbhavsarlambofod	US East (N. Virginia) us-east-1	Objects can be public	2020-05-02T04:43:40.000Z

Task 4: AWS Systems Manager: Managed Instances

Navigate to AWS Systems Manager Service.



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



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.

AWS Systems Manager X

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*

Maintenance Windows

Instances & Nodes

Managed Instances

View details Agent auto update Configure Inventory Actions

Q

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

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.

Launch Instance Connect Actions

Filter by tags and attributes or search by keyword

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

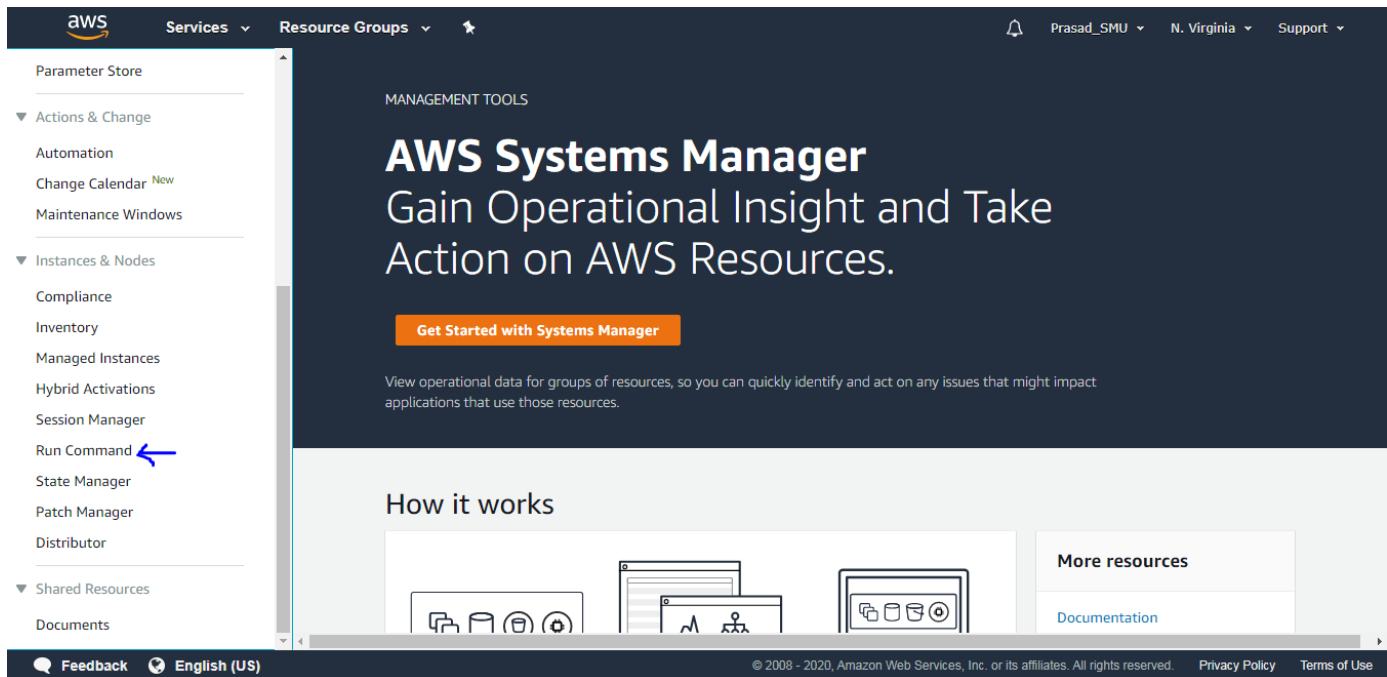
Instance: **i-0a25233378569135c (Ansible Controller)** Public DNS: **ec2-54-165-239-253.compute-1.amazonaws.com**

Description Status Checks Monitoring Tags

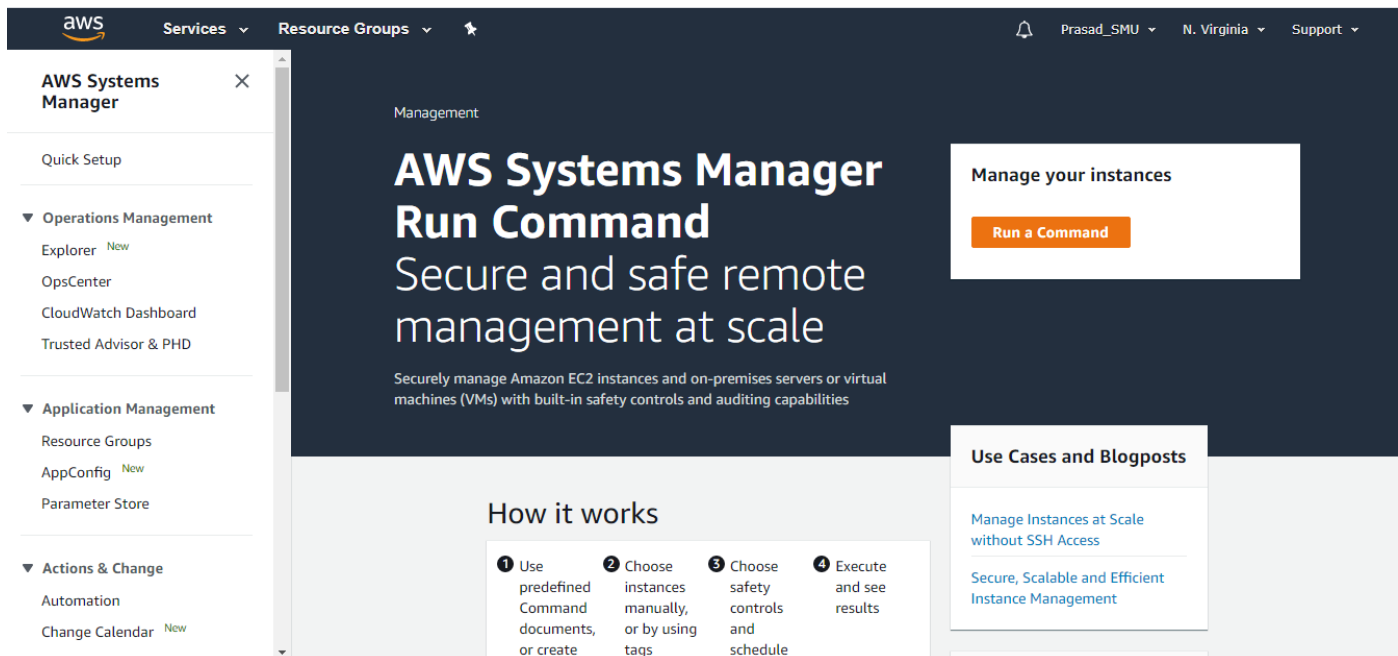
Instance ID	i-0a25233378569135c	Public DNS (IPv4)	ec2-54-165-239-253.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	54.165.239.253
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	
Private DNS	ip-10-192-10-197.ec2.internal	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.

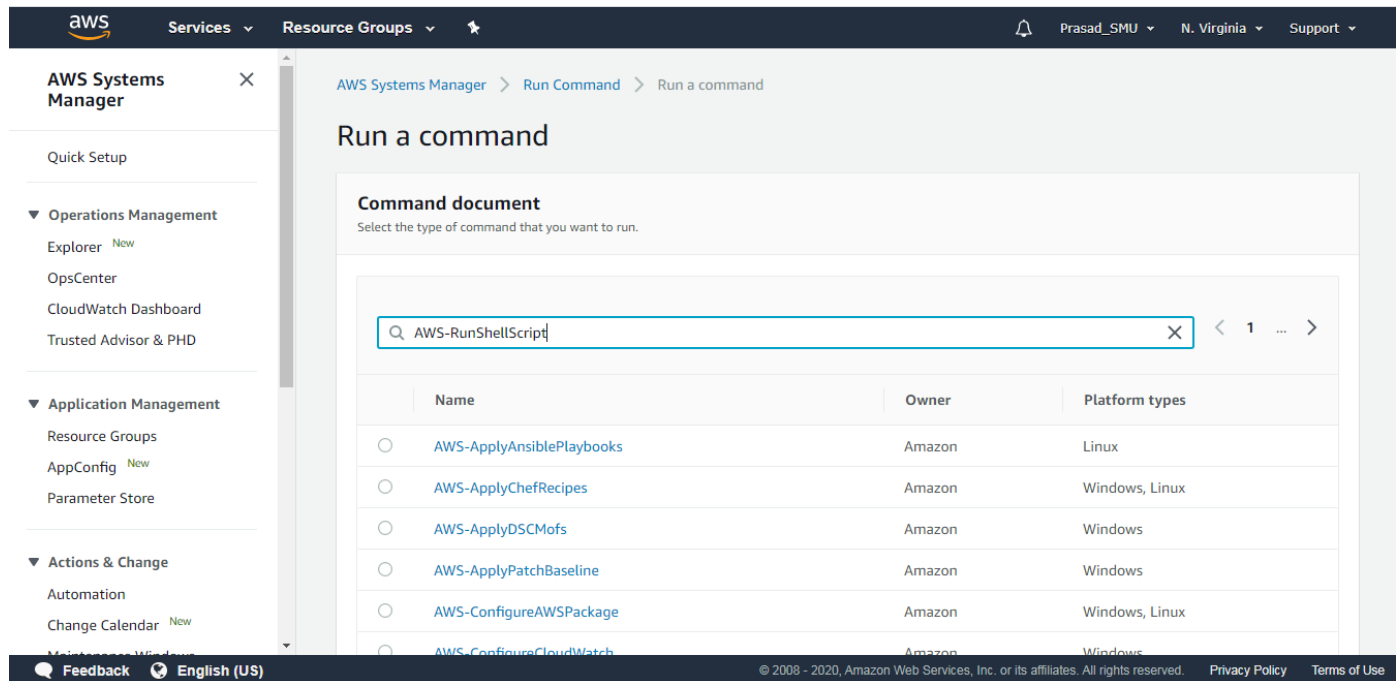


Click on Run a Command.



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

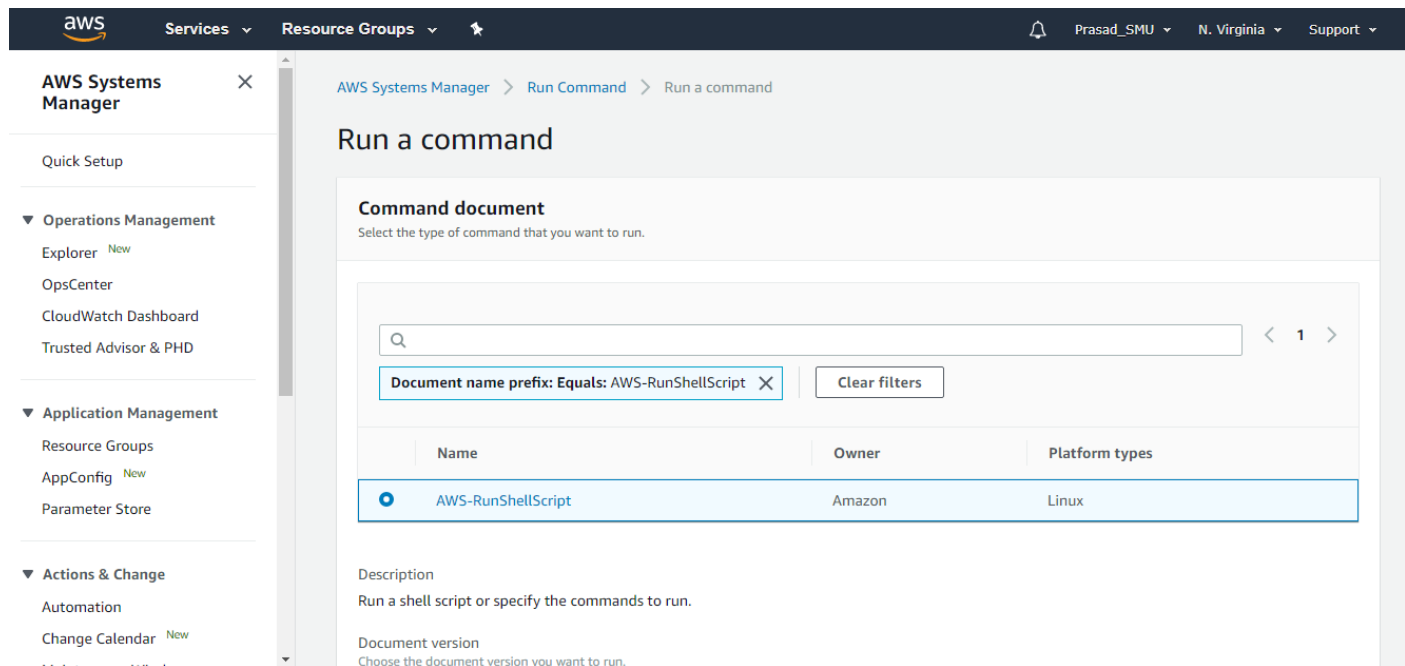
AWS-RunShellScript



The screenshot shows the AWS Systems Manager console. The left sidebar contains navigation links for AWS Systems Manager, Quick Setup, Operations Management (Explorer, OpsCenter, CloudWatch Dashboard, Trusted Advisor & PHD), Application Management (Resource Groups, AppConfig, Parameter Store), and Actions & Change (Automation, Change Calendar). The main content area is titled 'Run a command' and shows a 'Command document' section. A search bar contains the text 'AWS-RunShellScript'. Below the search bar is a table with the following data:

	Name	Owner	Platform types
<input type="radio"/>	AWS-ApplyAnsiblePlaybooks	Amazon	Linux
<input type="radio"/>	AWS-ApplyChefRecipes	Amazon	Windows, Linux
<input type="radio"/>	AWS-ApplyDSCMofs	Amazon	Windows
<input type="radio"/>	AWS-ApplyPatchBaseline	Amazon	Windows
<input type="radio"/>	AWS-ConfigureAWSPackage	Amazon	Windows, Linux
<input type="radio"/>	AWS-ConfigureCloudWatch	Amazon	Windows

Select the Command Document.



The screenshot shows the AWS Systems Manager console. The left sidebar is the same as the previous screenshot. The main content area is titled 'Run a command' and shows a 'Command document' section. The search bar is empty. Below the search bar is a table with the following data:

	Name	Owner	Platform types
<input checked="" type="radio"/>	AWS-RunShellScript	Amazon	Linux

Below the table, the 'Description' section is visible, stating: 'Run a shell script or specify the commands to run.' The 'Document version' section is also visible, stating: 'Choose the document version you want to run.'

Read the highlighted Description.

Command document

Select the type of command that you want to run.

< 1 >

Document name prefix: Equals: AWS-RunShellScript X

Clear filters

Name	Owner	Platform types
<input checked="" type="radio"/> 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.

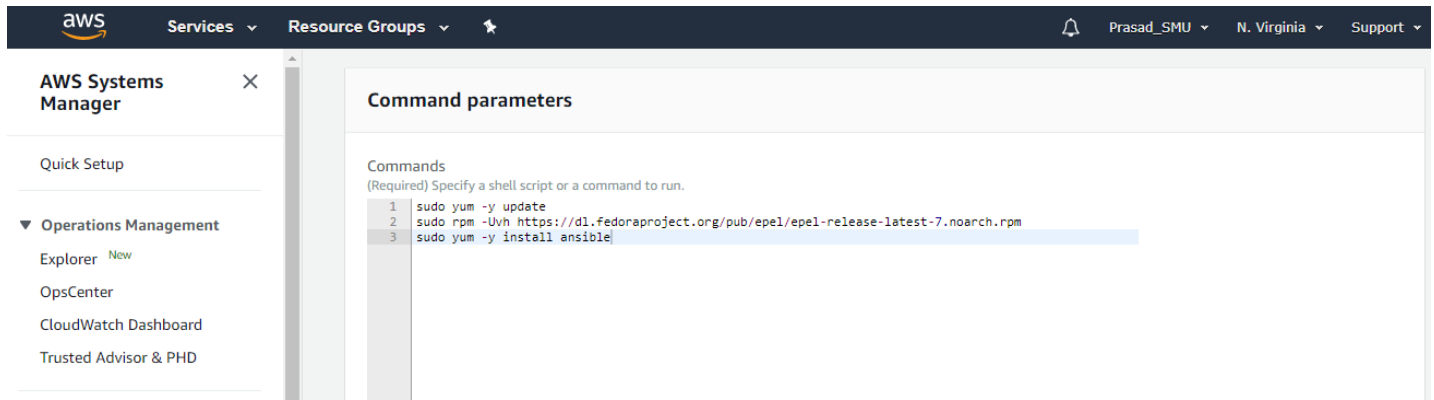
Instances

< 1 >

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Availability zone	Ping status	Last pin
<input checked="" type="checkbox"/>	SSM	i-0a25233378569135c	running	us-east-1a	Online	02/05/2023 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.



aws Services Resource Groups

Prasad_SMU N. Virginia Support

AWS Systems Manager

Quick Setup

▼ Operations Management

Explorer New

OpsCenter

CloudWatch Dashboard

Trusted Advisor & PHD

Command parameters

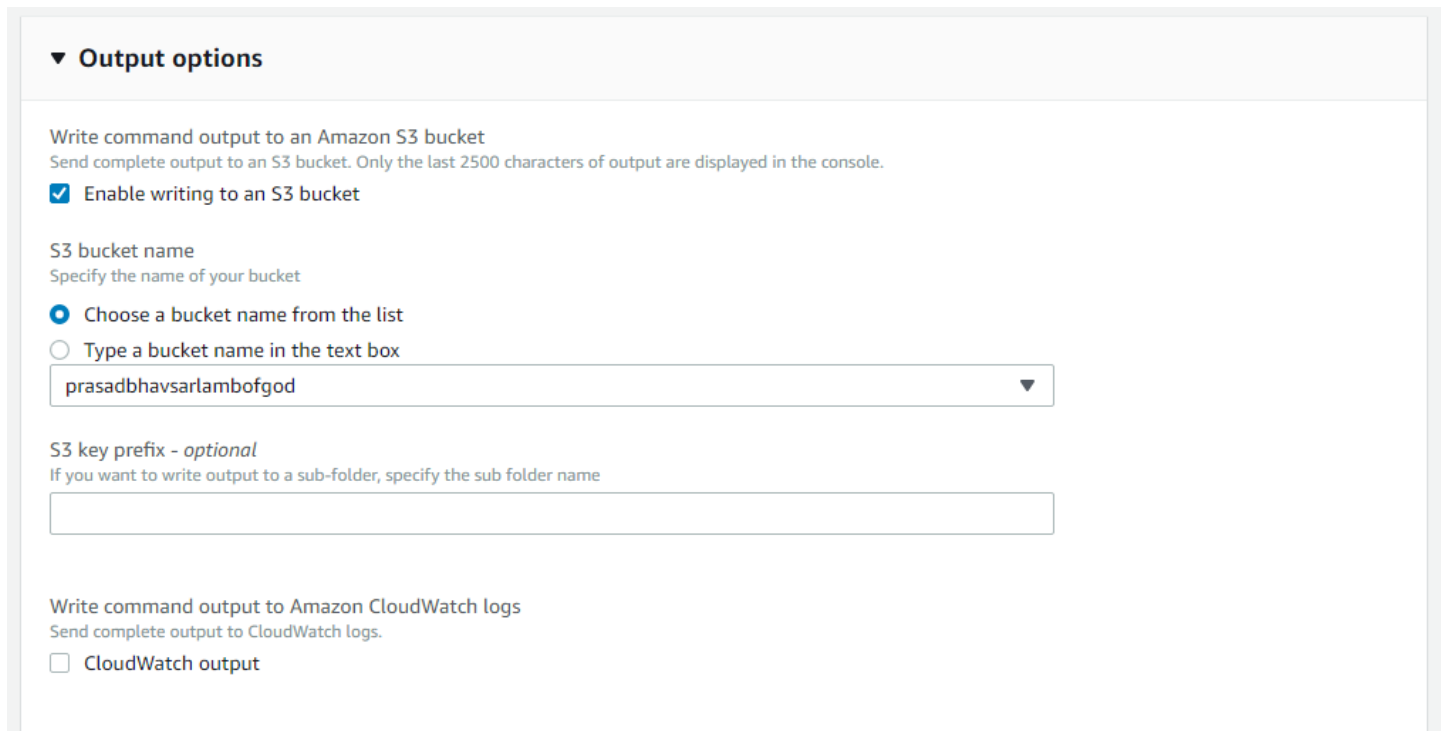
Commands
(Required) Specify a shell script or a command to run.

```
1 sudo yum -y update
2 sudo rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
3 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

prasadbhavsarlambogod ▼

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.

The screenshot shows the AWS Systems Manager console. A green notification bar at the top states: "Command ID: 86cbd007-4d7e-49d1-8b39-79948e92d377 was successfully sent!". The breadcrumb trail is "AWS Systems Manager > Run Command > Command ID: 86cbd007-4d7e-49d1-8b39-79948e92d377". The command title is "Command ID: 86cbd007-4d7e-49d1-8b39-79948e92d377". Below the title are buttons: "Cancel command", "Rerun" (with a 'New' tag), and "Copy to new" (with a 'New' tag). The "Command status" section shows: Overall status: In Progress (with a red circle icon), Detailed status: In Progress (with a red circle icon), # targets: 1, # completed: 0, # error: 0, and # delivery timed out: 0. The "Targets and outputs" section has a search bar and a "View output" button. Below is a table with 7 columns: Instance ID, Instance name, Status, Detailed Status, Start time, and Finish time. One target is listed with Instance ID "i-0a25233378569135c", Instance name "ip-10-192-10-197.ec2.internal", Status "In Progress" (with a red circle icon), and Detailed Status "In Progress" (with a red circle icon).

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**.

The screenshot shows the AWS Systems Manager console with the same command ID. The green notification bar remains. The breadcrumb trail is "AWS Systems Manager > Run Command > Command ID: 86cbd007-4d7e-49d1-8b39-79948e92d377". The command title is "Command ID: 86cbd007-4d7e-49d1-8b39-79948e92d377". Below the title are buttons: "Cancel command", "Rerun" (with a 'New' tag), and "Copy to new" (with a 'New' tag'). The "Command status" section shows: Overall status: Success (with a green checkmark icon), Detailed status: Success (with a green checkmark icon), # targets: 1, # completed: 1, # error: 0, and # delivery timed out: 0. The "Targets and outputs" section has a search bar and a "View output" button. Below is a table with 7 columns: Instance ID, Instance name, Status, Detailed Status, Start time, and Finish time. One target is listed with Instance ID "i-0a25233378569135c", Instance name "ip-10-192-10-197.ec2.internal", Status "Success" (with a green checkmark icon), and Detailed Status "Success" (with a green checkmark icon). The Start time is "Sat, 02 May 2020 07:38:06 GMT" and the Finish time is "Sat, 02 May 2020 07:38:18 GMT".

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

The screenshot shows the AWS Management Console. On the left, the 'INSTANCES' section is expanded, showing a list of instances. The instance 'i-0a25233378569135c' (SSM) is selected. The 'Description' tab is active, showing details like Instance ID, Instance state (running), Instance type (t2.micro), Private DNS (ip-10-192-10-197.ec2.internal), and Private IPs (10.192.10.197). A terminal window is open, showing the command 'which ansible' being executed, and the output is '/usr/bin/ansible', indicating that Ansible is successfully installed on the instance.

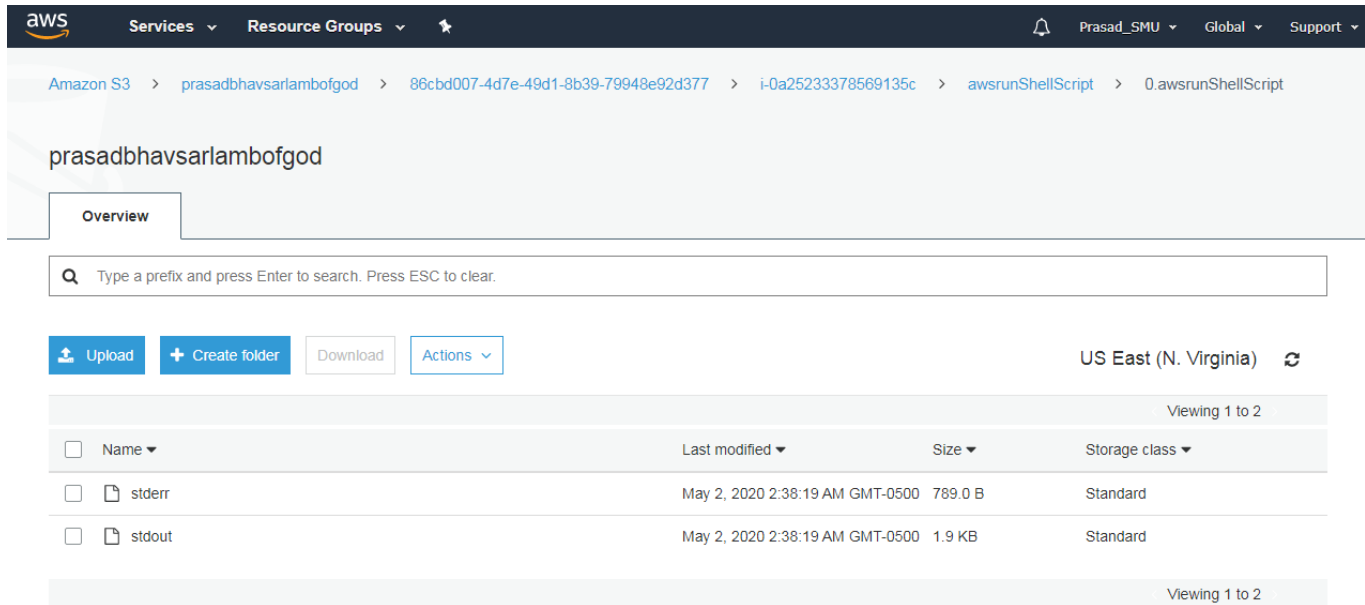
Ansible has been successfully installed on the Target Instance.

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

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

The screenshot shows the AWS S3 console. The bucket 'prasadbhavsarlambogod' is selected. The 'Overview' tab is active, showing the bucket's details. The 'Properties' tab is also visible. The bucket is located in the 'US East (N. Virginia)' region. The bucket contains one object, '86cbd007-4d7e-49d1-8b39-79948e92d377', which is the SSM log file.

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



Amazon S3 > prasadbhavsarlambofgod > 86cbd007-4d7e-49d1-8b39-79948e92d377 > i-0a25233378569135c > awsrunShellScript > 0.awsrunShellScript

prasadbhavsarlambofgod

Overview

Q Type a prefix and press Enter to search. Press ESC to clear.

Upload Create folder Download Actions

US East (N. Virginia)

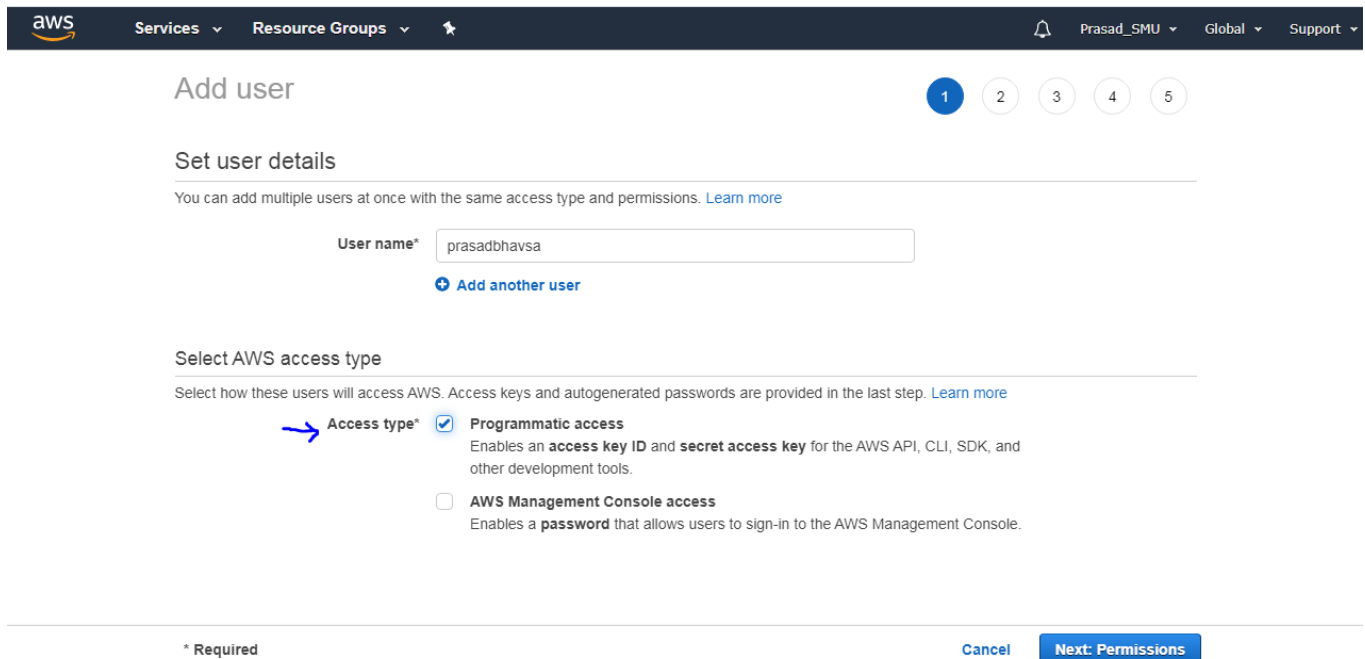
Name	Last modified	Size	Storage class
stderr	May 2, 2020 2:38:19 AM GMT-0500	789.0 B	Standard
stdout	May 2, 2020 2:38:19 AM GMT-0500	1.9 KB	Standard

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**.



aws Services Resource Groups

Add user

1 2 3 4 5

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name* prasadbhavsar

+ Add another user

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

Cancel Next: Permissions

Click on Next: Permissions.

Click on **Attach existing policies directly**.

Search and select **AmazonEC2FullAccess** Policy.

Add user 1 2 3 4 5

▼ Set permissions

Add user to group Copy permissions from existing user **Attach existing policies directly**

Create policy

Filter policies ▼ Showing 1 result

	Policy name ▼	Type	Used as
<input checked="" type="checkbox"/>	AmazonEC2FullAccess	AWS managed	Permissions policy (3)

Cancel Previous **Next: Tags**

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.

Identity and Access Management (IAM)

Dashboard

▼ Access management

- Groups
- Users**
- Roles
- Policies

Add user **Delete user**

Showing 3 results

	User name ▼	Groups	Access key age	Password age	Last activity	MFA
<input type="checkbox"/>	admin1	admin	None	200 days	200 days	Not enabled
<input type="checkbox"/>	admin2	admin	200 days	200 days	200 days	Not enabled
<input type="checkbox"/>	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

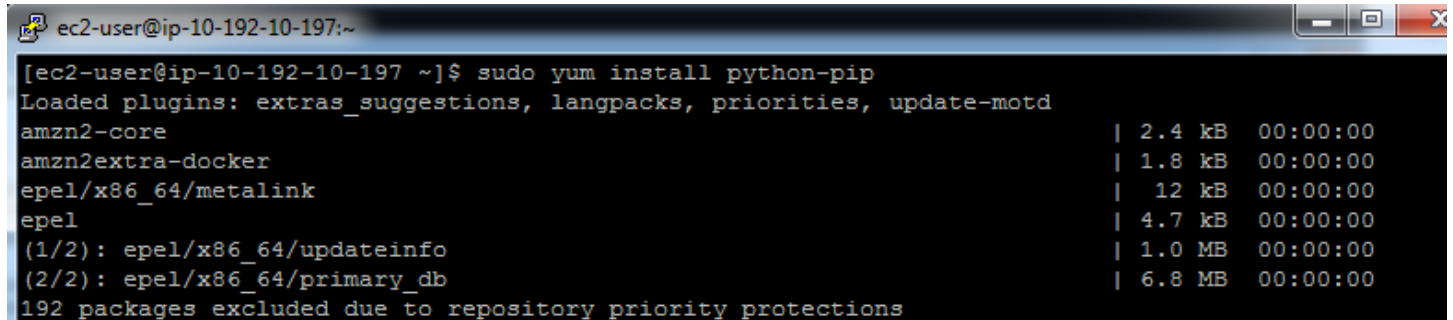
  _ | _ | _ )
  _ | ( _ /   Amazon Linux 2 AMI
  __| \__|__|

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:~  
[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`

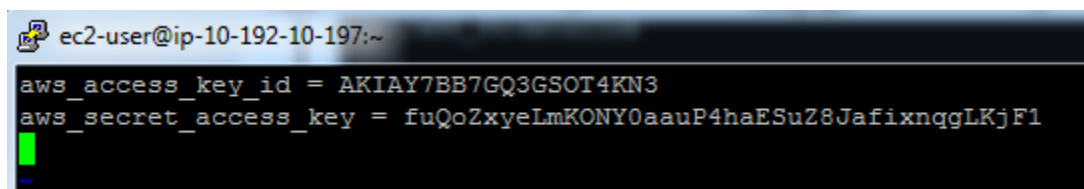
```
~]$ sudo pip install boto
```

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.



```
ec2-user@ip-10-192-10-197:~  
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

Switch to Root User.

```
[root@ip-10-192-10-197 ec2-user]# sudo su -
Last login: Wed May  6 05:56:21 UTC 2020 on pts/0
```

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

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

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

```
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
```

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.

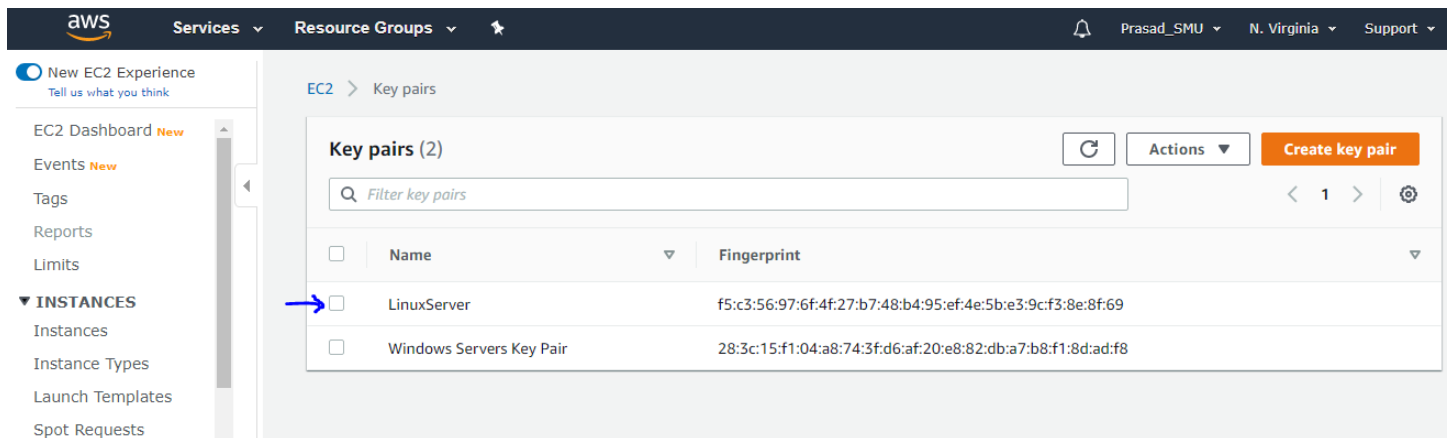
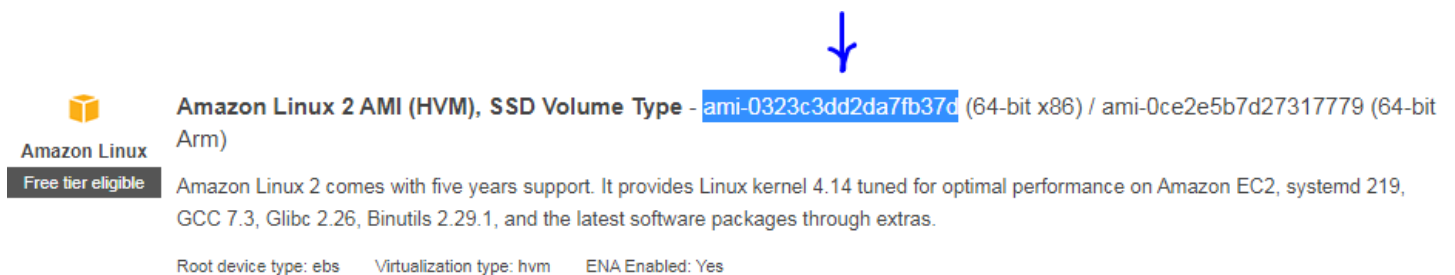


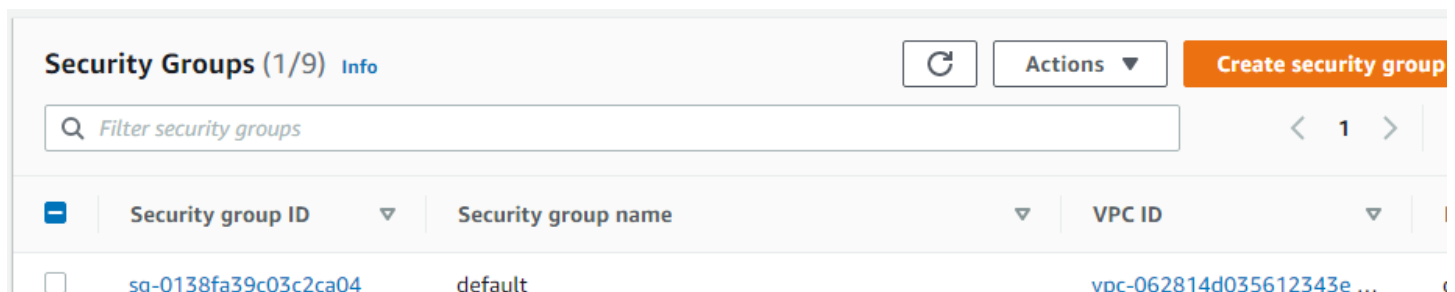
Image:

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



Group:

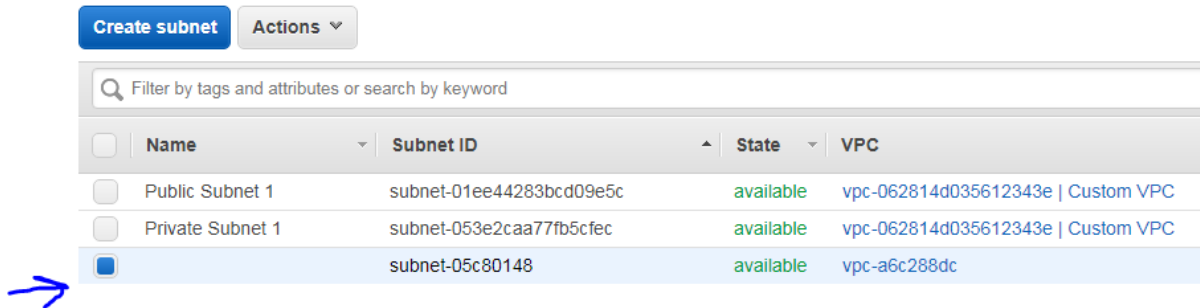
Security Group information can be found on the EC2 Service Dashboard. Select the Security Group as per your choice. I've selected the Default Security Group.



vpc_subnet_id:

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.



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 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`

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

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

Task 10: Execute Ansible Playbook on EC2 Instance

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    ignored=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.

The screenshot displays the AWS Management Console interface for the EC2 service. The left sidebar shows the navigation menu with 'INSTANCES' selected. The main content area shows a table of EC2 instances. The instance 'Ansible Controller' (ID: i-02a89e653909f7a66) is highlighted in blue. It is a t2.micro instance in the us-east-1a Availability Zone, currently in the 'running' state. The console shows a list of 11 instances, with the new one at the bottom. The 'Status Checks' column shows '2/2 checks passed' for most instances, and 'Initializing' for the new one. The 'Alarm Status' column shows 'None' for all instances.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
	i-0cb4711711c5a18d8	t2.micro	us-east-1a	stopped		None
	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
	i-06fc6c6f2400143e8	t2.micro	us-east-1a	running	2/2 checks passed	None
	i-056010c787acae7db	t2.micro	us-east-1a	running	2/2 checks passed	None
	i-05308de4d1fe34b24	t2.micro	us-east-1a	running	Initializing	None
	i-04d960bbddffc81ee	t2.micro	us-east-1d	running	2/2 checks passed	None
	i-037b36f4a23b67a3b	t2.micro	us-east-1a	running	2/2 checks passed	None
	i-02f01417acefac668	t2.micro	us-east-1a	running	2/2 checks passed	None
	i-02a89e653909f7a66	t2.micro	us-east-1a	running	2/2 checks passed	None

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

		i-05308de4d1fe34b24	t2.micro	us-east-1a		running		2/2 checks passed	None
		i-04d960bbddffc81ee	t2.micro	us-east-1d		running		2/2 checks passed	None
Private IPs		10.192.10.4					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	

This completes the Lab on Launching EC2 Instance using Ansible on AWS.

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