
PROJECT DETAILS

Project Title

Automating Infrastructure using Terraform

Problem statement

Nowadays, infrastructure automation is critical. We tend to put the most emphasis on software development processes, but infrastructure deployment strategy is just as important. Infrastructure automation not only aids disaster recovery, but it also facilitates testing and development.

Your organization is adopting the DevOps methodology and in order to automate provisioning of infrastructure there's a need to setup a centralised server for Jenkins.

Terraform is a tool that allows you to provision various infrastructure components. Ansible is a platform for managing configurations and deploying applications. It means you'll use Terraform to build a virtual machine, for example, and then use Ansible to install the necessary applications on that machine.

Considering the Organizational requirement, you are asked to automate the infrastructure using Terraform first and install other required automation tools in it.

Tool required

- Terraform (installed on a controller machine)
- AWS account with security credentials)
- For SSH I have created in terraform command code put ssh-key with instance that will create in future

Need configuration on instance after created

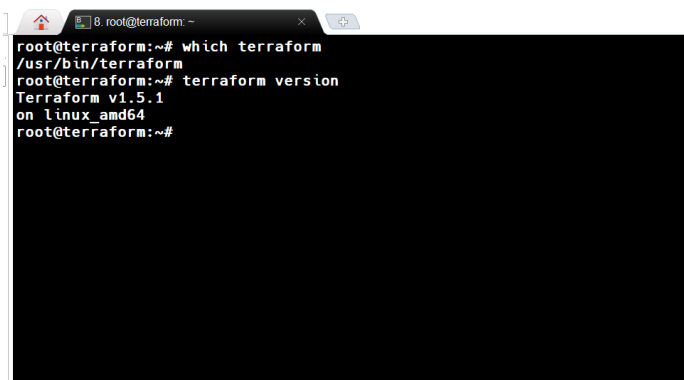
- Launch an EC2 instance using Terraform
- Connect to the instance
- Install Jenkins, Java and Python in the instance
- Validate that the packages are installed

Procedure

Step 1. Configure the controller node

Installation steps:

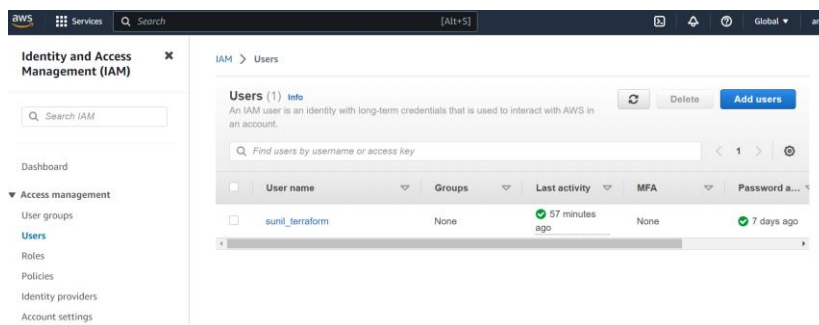
- Procedure to install terraform in controller:
 - First need to visit
https://developer.hashicorp.com/terraform/downloads?product_intent=terraform
 - Follow the below command to install terraform in linux os at controller node
 - `wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg`
 - `echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list`
 - `sudo apt update && sudo apt install terraform`
 - To validate:
 - Terraform version
 - Which terraform



```
root@terraform:~# which terraform
/usr/bin/terraform
root@terraform:~# terraform version
Terraform v1.5.1
on linux_amd64
root@terraform:~#
```

Step 2: Setup AWS “access key” & “secret key”

- Procedure to setup access key and secret key:
 - Visit AWS account > go to IAM service > user section > create user



- Click “Add users” in case it not created in my case it is already created

Specify user details

User details

User name
sunil1

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and +, -, @, _ (hyphen)

☒ **Provide user access to the AWS Management Console - optional**
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

Are you providing console access to a person?

User type

☒ **Specify a user in Identity Center - Recommended**
We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

☐ I want to create an IAM user

- Click “ I want to create an IAM user” and then then follow as below

Console password

☒ **Autogenerated password**
You can view the password after you create the user.

☐ **Custom password**
Enter a custom password for the user.

Must be at least 8 characters long

Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # \$ % ^ & * () _ + - (hyphen) = [] { } ' "

☐ Show password

☐ **Users must create a new password at next sign-in - Recommended**
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

Information If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel Next

- Click “next” & attach policy as “ AdministratorAccess”

Permissions options

☐ **Add user to group**
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ **Copy permissions**
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ **Attach policies directly**
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1106) [Refresh](#) [Create policy](#)

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

Filter by Type

Search: admin X All types 37 matches < 1 2 > ⚙

	Policy name	Type	Attached entities
<input checked="" type="checkbox"/>	AdministratorAccess	AWS managed - job function	1
<input type="checkbox"/>	AdministratorAccess-A...	AWS managed	0
<input type="checkbox"/>	AdministratorAccess-A...	AWS managed	0

- Now create user

sunil1 Autogenerated No

Permissions summary

< 1 >

Name	Type	Used as
AdministratorAccess	AWS managed - job function	Permissions policy

Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

Cancel Previous **Create user**

Retrieve password

You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.

Console sign-in details

Console sign-in URL
<https://763351537969.signin.aws.amazon.com/console>

User name
sunil1

Console password
***** [Show](#)

[Email sign-in instructions](#)

[Download .csv file](#)
[Return to users list](#)

- Now create access and secret key for above user

Summary

ARN

arn:aws:iam::763351537969:user:sunil1

Console access

Enabled without MFA

Access key 1

Not enabled

Created

July 01, 2023, 19:00 (UTC+05:30)

Last console sign-in

Never

Access key 2

Not enabled

Permissions

Groups

Tags

Security credentials

Access Advisor

Console sign-in

Manage console access

Console sign-in link

<https://763351537969.signin.aws.amazon.com/console>

Console password

Updated 1 minute ago (2023-07-01 19:00 GMT+5:30)

- Click create access key

Remove

Resync

Assign MFA device

Device type	Identifier	Certifications	Created on
No MFA devices. Assign an MFA device to improve the security of your AWS environment.			
<div>Assign MFA device</div>			

Access keys (0)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

Create access key

No access keys

As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. [Learn more](#)

Create access key

- Select for command line

Access key best practices & alternatives [info](#)

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

☒ Command Line Interface (CLI)
 You plan to use this access key to enable the AWS CLI to access your AWS account.

☐ Local code
 You plan to use this access key to enable application code in a local development environment to access your AWS account.

☐ Application running on an AWS compute service
 You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

☐ Third-party service
 You plan to use this access key to enable a third-party service to access your AWS account.

- Check policy and click next

- Access and secret key generated and need to save for future used

Step3 Write terraform file

Create aws instance using terraform

➤ Procedure to follow as mentioned below:

- First need to create directory as "terra" in my case
- Use command "mkdir terra" && "cd terra"

```

root@terraform:~# ls
install-docker.sh  main.tf  snap
root@terraform:~# mkdir terra
root@terraform:~# cd terra/
root@terraform:~/terra#

```

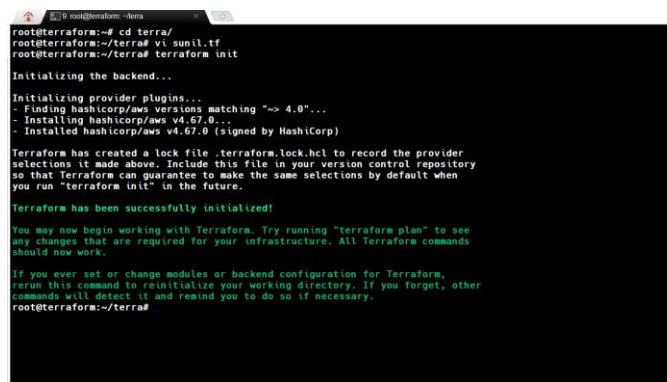
- Make “sunil.tf” file and write code to add cloud providers (like aws, azure, gcp etc)
- Vi sunil.tf >> press “i” insert for write then write code as mention below

```

• terraform {
•   required_providers {
•     aws = {
•       source  = "hashicorp/aws"
•       version = "~> 4.0"
•     }
•   }
• }

```

- Then press “:wq!” to save
- Now initiate with terraform
- Command “terraform init”



```

root@terraform:~# cd terra/
root@terraform:~/terra# vi sunil.tf
root@terraform:~/terra# terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 4.0"...
- Installing hashicorp/aws v4.67.0...
- Installed hashicorp/aws v4.67.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
root@terraform:~/terra#

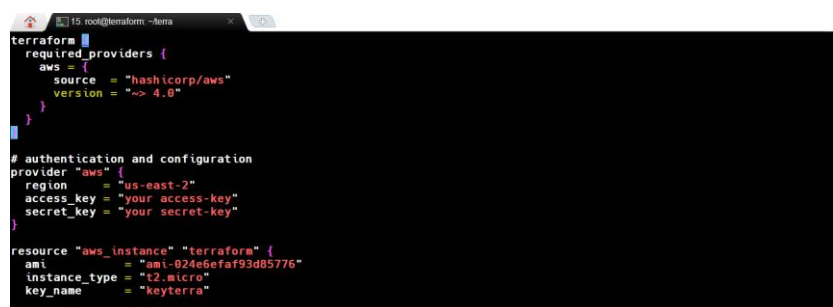
```

- Authentication and configuration
 - Vi sunil.tf
 - And write code for “aws authentication and configuration”

```

▪ # Authentication and configuration
▪ provider "aws" {
▪   region      = "ap-southeast-2"
▪   access_key  = "Your_access_key"
▪   secret_key  = "your_secure_key"
▪ }

```



```

terraform
required_providers {
  aws = {
    source  = "hashicorp/aws"
    version = "~> 4.0"
  }
}

# authentication and configuration
provider "aws" {
  region      = "us-east-2"
  access_key  = "your access-key"
  secret_key  = "your secret-key"
}

resource "aws_instance" "terraform" {
  ami          = "ami-024e6efaf93d85776"
  instance_type = "t2.micro"
  key_name     = "keyterra"
}

```

- Command “terraform plan” to check plan


```

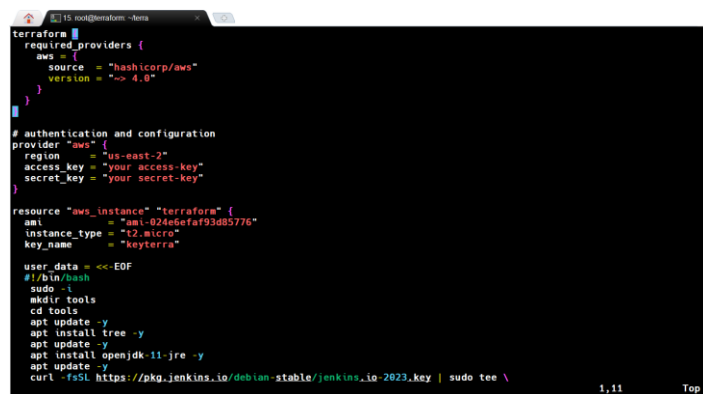
key_name      = "keyterra"

user_data = <<-EOF
#!/bin/bash
apt update -y
apt install tree -y
apt update -y
apt install openjdk-11-jre -y
apt update -y
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
apt-get update -y
apt-get install jenkins -y
add-apt-repository ppa:deadsnakes/ppa
apt update -y
apt list --upgradable
apt-get update -y
apt install python -y
EOF

tags = {
  Name = "terraform"
}

resource "aws_key_pair" "terraform" {
  key_name      = "keyterra"
  public_key    = "your ssh-key"
}

```



```

terraform
required_providers {
  aws = {
    source  = "hashicorp/aws"
    version = "~> 4.0"
  }
}

# authentication and configuration
provider "aws" {
  region     = "us-east-2"
  access_key = "your access-key"
  secret_key = "your secret-key"
}

resource "aws_instance" "terraform" {
  ami          = "ami-024e6efaf93d85776"
  instance_type = "t2.micro"
  key_name     = "keyterra"

  user_data = <<-EOF
#!/bin/bash
sudo -i
mkdir tools
cd tools
apt update -y
apt install tree -y
apt update -y
apt install openjdk-11-jre -y
apt update -y
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
1,11 Top

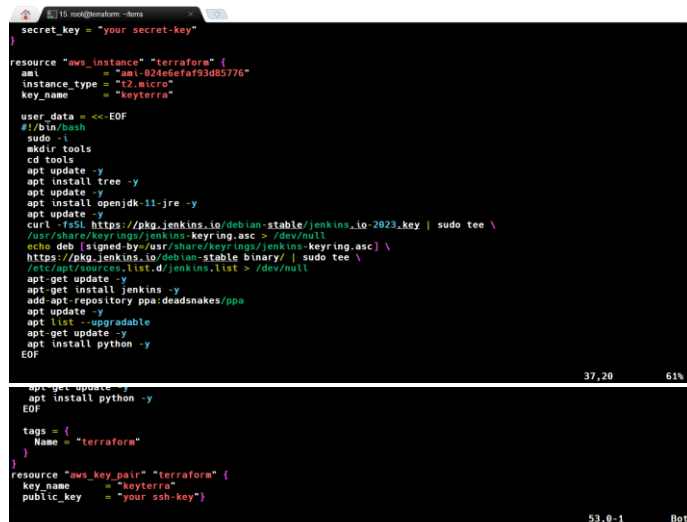
```

- User_data script for install tree, Java Python and Jenkins at created instance via terraform.


```

user_data = <<-EOF
#!/bin/bash
sudo -i
mkdir tools
cd tools
apt update -y
apt install tree -y
apt update -y
apt install openjdk-11-jre -y
apt update -y
curl -fsSL https://pkg.jenkins.io/debian-
stable/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-
keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
apt-get update -y
apt-get install jenkins -y
add-apt-repository ppa:deadsnakes/ppa
apt update -y
apt list --upgradable
apt-get update -y
apt install python -y
EOF

```



```

secret_key = "your secret-key"

resource "aws_instance" "terraform" {
  ami           = "ami-024e6ef93d85776"
  instance_type = "t2.micro"
  key_name      = "keyterra"

  user_data = <<-EOF
#!/bin/bash
sudo -i
mkdir tools
cd tools
apt update -y
apt install tree -y
apt update -y
apt install openjdk-11-jre -y
apt update -y
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
apt-get update -y
apt-get install jenkins -y
add-apt-repository ppa:deadsnakes/ppa
apt update -y
apt list --upgradable
apt-get update -y
apt install python -y
EOF

tags = {
  Name = "terraform"
}

resource "aws_key_pair" "terraform" {
  key_name      = "keyterra"
  public_key    = "your ssh-key"
}

```

- Code for terraform



sunil1.tf

- Now execute terraform plan command to check plan for execution

```

}
+ tenancy = (known after apply)
+ user_data = "0q9h0D3a04s1714cedf2a976f2a3d9c462e7496" = (known after apply)
+ user_data_base64 = (known after apply)
+ user_data_replace_on_change = false = (known after apply)
+ vpc_security_group_ids = (known after apply)
}

# aws_key_pair.terraform will be created
resource "aws_key_pair" "terraform" {
+ arn = (known after apply)
+ fingerprint = (known after apply)
+ id = (known after apply)
+ key_name = "keyterra"
+ key_name_prefix = (known after apply)
+ key_pair_id = (known after apply)
+ key_type = (known after apply)
+ public_key = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGCvDCDeEnUv5jKp4FESHT10sgW0hIWh8IqTkg0nz/Td0UpruW0
inF9wt0t1c2Jw0RLd40GBM1sTNzbyXTNRqgs2kNkYpLPCGbAhg5dez0ZMTjhuZ1qY0ruRKf10kM1WRWwVsluy+e3ycYKf8q1lrhvcqfNIZaeMUm0coG
vJYcZuWb/t0h0rWZD7YDXdYimR1ka3abzITqTkaYdGUm0NNNSRZemaazhXpV0Td+eYiGU01fMc tA3TAUHKJa2A1K4dzR1EPLnZU81zp8784k0C79Ca9
pPvPeficcy/280121c7oX4bpJ5AKSh16wz1cc/QU1U0b3JhuY67eUKS3x0hsdubcwFJ2aoupIF1AM8BP7DgEJJAzCCmrLX71UHM3UK/km/7bY4Zd4fxE7K
DJCbPhLasCykh5V55hDVMIRKYdpH+EclTookbpc9fbZubR0WIZjxkpOPkc4+fBWVTV6GQh0RTh++cFJy3KTeKu7paMuHj0Q5lTKXModqtId5FW-
root@terraform"
+ tags_all = (known after apply)
}

Plan: 2 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions
if you run "terraform apply" now.
root@terraform:~/terra#

```

- To validate run command “terraform validate”

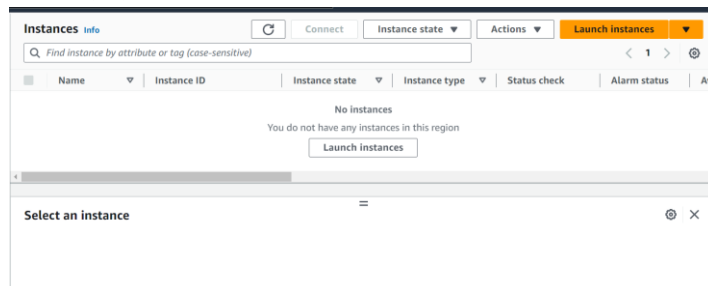
```

root@terraform:~/terra# terraform validate
Success! The configuration is valid.

root@terraform:~/terra#

```

- Before go proceed for apply we need to validate instance in AWS account



- Final step to create instance via terraform
 - To execute and create instance on AWS then run command “terraform apply”

```

root@terraform:~/terra# terraform apply

```


EC2 > Security Groups > sg-05c430564237954db - default > Edit outbound rules

Edit outbound rules [info](#)

Outbound rules control the outgoing traffic that's allowed to leave the instance.

Security group rule ID	Type	Protocol	Port range	Destination	Description - optional	
sg-0714a08b07c9d751d	SSH	TCP	22	Custom 0.0.0.0/0		Delete
sg-08b5f368af4c10d57	All traffic	All	All	Custom 0.0.0.0/0		Delete

[Add rule](#)

- **Login via controller**

- Take public ip of newly created instance and do ssh from controller node as mention below



Ssh -i ubuntu@public ip



ssh ubuntu@54.175.182.178 (in my case)

```
root@terraform:~/terra# ssh ubuntu@54.175.182.178
root@terraform:~/terra# vi sunil.tf
root@terraform:~/terra# ssh ubuntu@54.175.182.178
The authenticity of host '54.175.182.178 (54.175.182.178)' can't be established.
ED25519 key fingerprint is SHA256:e4Nq5RCunKIM4uyjnCv17MjGpa9CsQ1U8uvvWSLLJqo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.175.182.178' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Sat Jul 1 12:34:23 UTC 2023

System load:  0.0          Processes:    97
Usage of /:   31.8% of 7.57GB Users logged in:  0
Memory usage: 50%         IPv4 address for eth0: 172.31.81.55
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

73 updates can be applied immediately.
40 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
```

```
ED25519 key fingerprint is SHA256:e4Nq5RCunKIM4uyjnCv17MjGpa9CsQ1U8uvvWSLLJqo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.175.182.178' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Sat Jul 1 12:34:23 UTC 2023

System load:  0.0          Processes:    97
Usage of /:   31.8% of 7.57GB Users logged in:  0
Memory usage: 50%         IPv4 address for eth0: 172.31.81.55
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

73 updates can be applied immediately.
49 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sat Jul 1 12:34:25 2023 from 18.206.107.29
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-81-55:~$
```

- **validate**

- Now validate tools like java, Jenkins and python3 output

```
root@ip-172-31-81-55:~# which java
/usr/bin/java
root@ip-172-31-81-55:~# java --version
openjdk 11.0.19 2023-04-18
OpenJDK Runtime Environment (build 11.0.19+7-post-Ubuntu-0ubuntu122.04.1)
OpenJDK 64-Bit Server VM (build 11.0.19+7-post-Ubuntu-0ubuntu122.04.1, mixed mode, sharing)
root@ip-172-31-81-55:~# which jenkins
/usr/bin/jenkins
root@ip-172-31-81-55:~# jenkins --version
2.401.2
root@ip-172-31-81-55:~# which python3
/usr/bin/python3
root@ip-172-31-81-55:~# python3 --version
Python 3.10.6
root@ip-172-31-81-55:~#
```