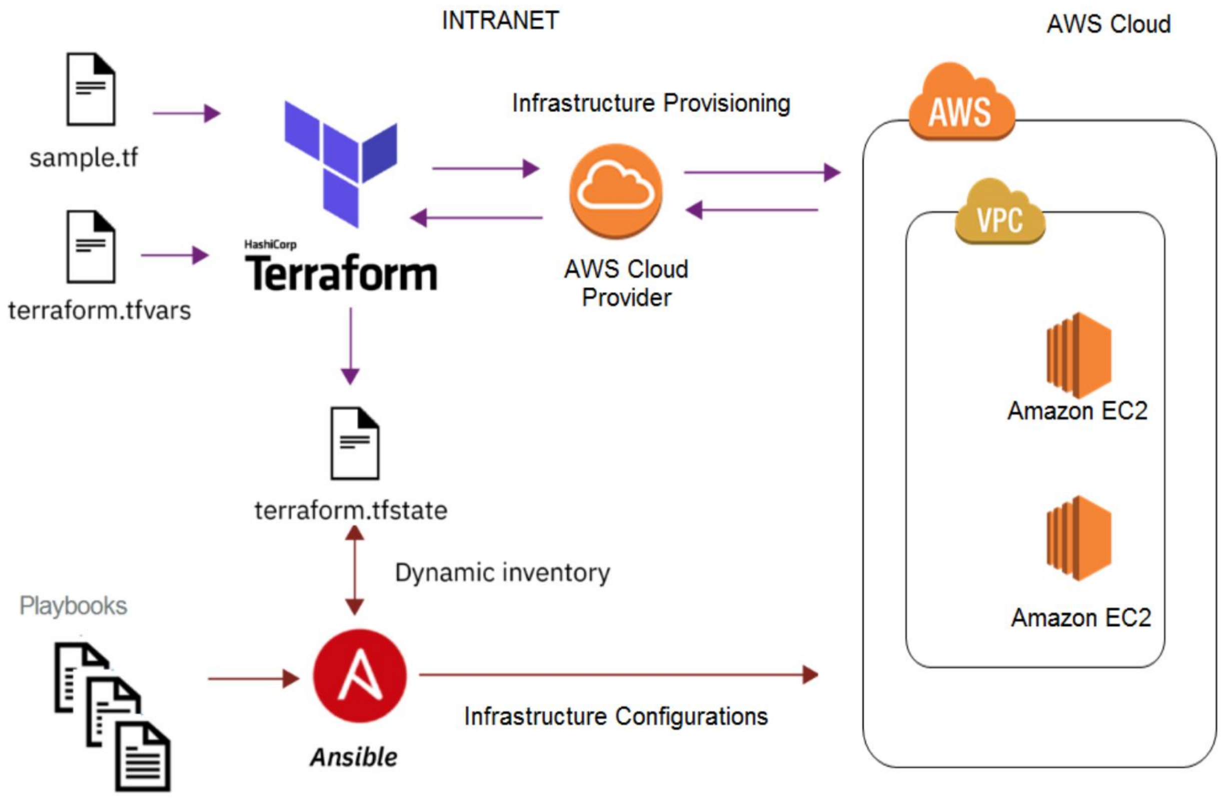
Tools used:

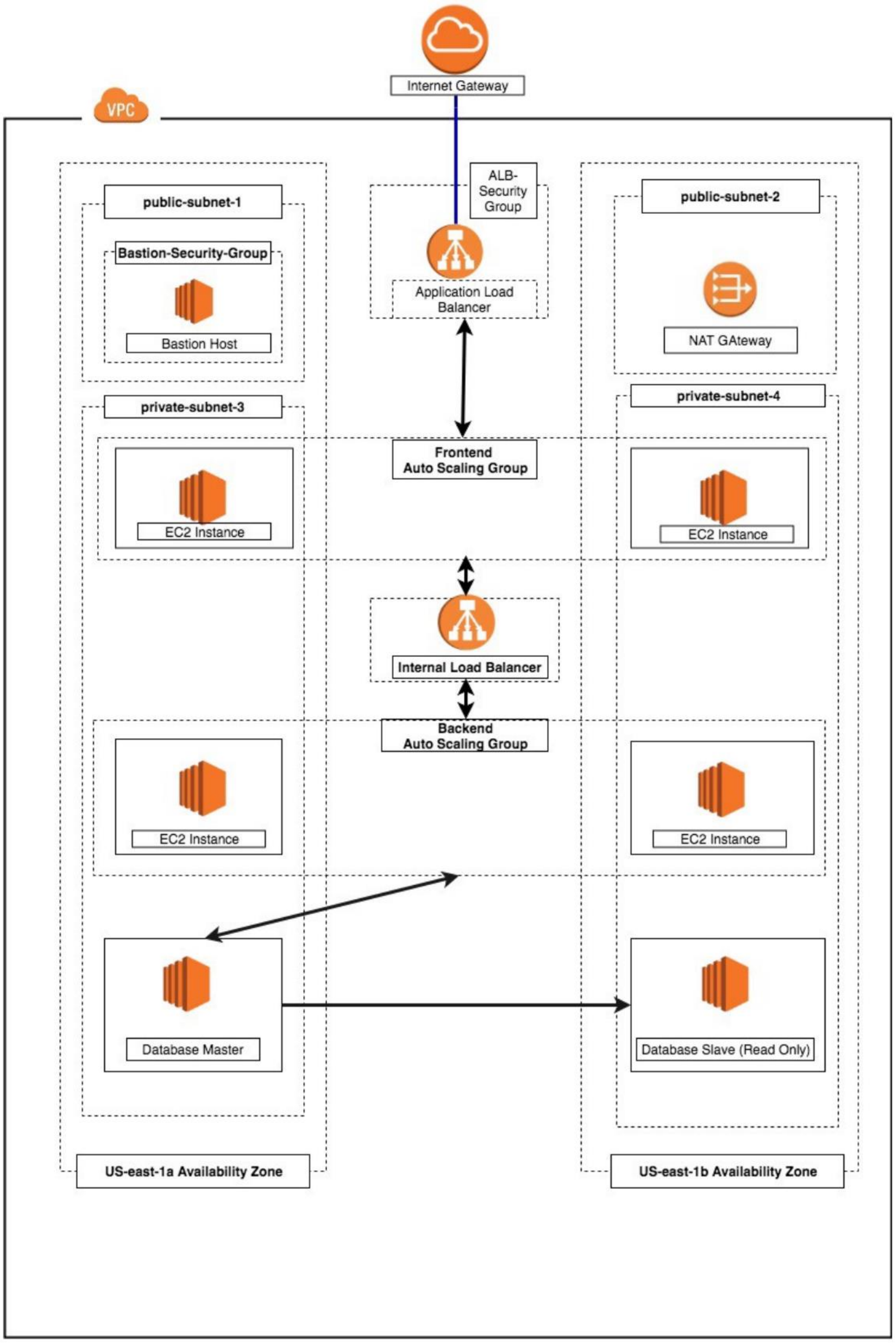
1. Ansible – To install and setup Jenkins master
2. Terraform – To form my VPC Resources
3. Jenkins – To implement CI/CD by accessing the VPC instances and deploying the node application

Architecture Diagrams

Server provisioning with Terraform and Ansible



3-Tier Application architecture on AWS


Internet Gateway
VPC
ALB-Security Group
Application Load Balancer
public-subnet-1
Bastion-Security-Group
Bastion Host
public-subnet-2
NAT GAteway
private-subnet-3
private-subnet-4
Frontend Auto Scaling Group
EC2 Instance
EC2 Instance
Internal Load Balancer
Backend Auto Scaling Group
EC2 Instance
EC2 Instance
Database Master
Database Slave (Read Only)
US-east-1a Availability Zone
US-east-1b Availability Zone

Project Resources

(1) **Repository for this project:** https://github.com/hani-hub/nodeApp-ansible-terraform-repo

Directory structure

```
.
├── artifacts
|   ├── config
|   |   └──config_multi-nodes.yaml
|   ├── playbooks
|   |   ├──install_gitlab.yaml
|   |   ├──install_java.yaml
|   |   └──install_jenkins.yaml
|   ├── scripts
|   |   ├──config_software.sh
|   |   ├──install_software.sh
|   |   └──ssh_pass.sh
|   ├── templates
|   |   ├──install_busybox.sh
|   |   ├──install_jenkins.sh
|   |   ├──install_nginx.sh
|   |   └──user_data.sh
|   └── terraform
|           ├── outputs.tf
|           ├── provider.tf
|           ├── resources.tf
|           ├── terraform.tf
|           └── variables.tf
├── images
|   ├── aws_configure.png
|   ├── aws_terraform_ans\v1.png
|   ├── aws_terraform_ans_v1.png
|   └── jenkins-ci.png
├── install.sh
├── README.md
├── screening
|   ├── api
|   |   ├── app.js
|   |   ├── bin
|   |   |   └── www
|   |   ├── package.json
|   |   ├── package-lock.json
|   |   └── README.md
|   ├── README.md
|   └── web
|       ├── app.js
|       ├── bin
|       |   └── www
|       ├── package.json
|       ├── package-lock.json
|       ├── public
|       |   └── stylesheets
|       |       └── style.css
|       ├── README.md
|       ├── routes
|       |   └── index.js
|       └── views
|           ├── error.jade
|           ├── index.jade
|           └── layout.jade
```

```
└── Vagrantfile
```

Thought Process

Use combination of IAC and CM

- Terraform will provision infrastructure like EC2 instances, Security Goups, ELB and VPC into AWS IaC
- Ansible will deploy/test application on EC2 instance as CM like Jenkins and GitLab

Setting up the environment:

This guide assumes that you already have some understanding of AWS and have a working account. The installation of Terraform and Ansible are straightforward, and the details are at this link.

Prerequisites:

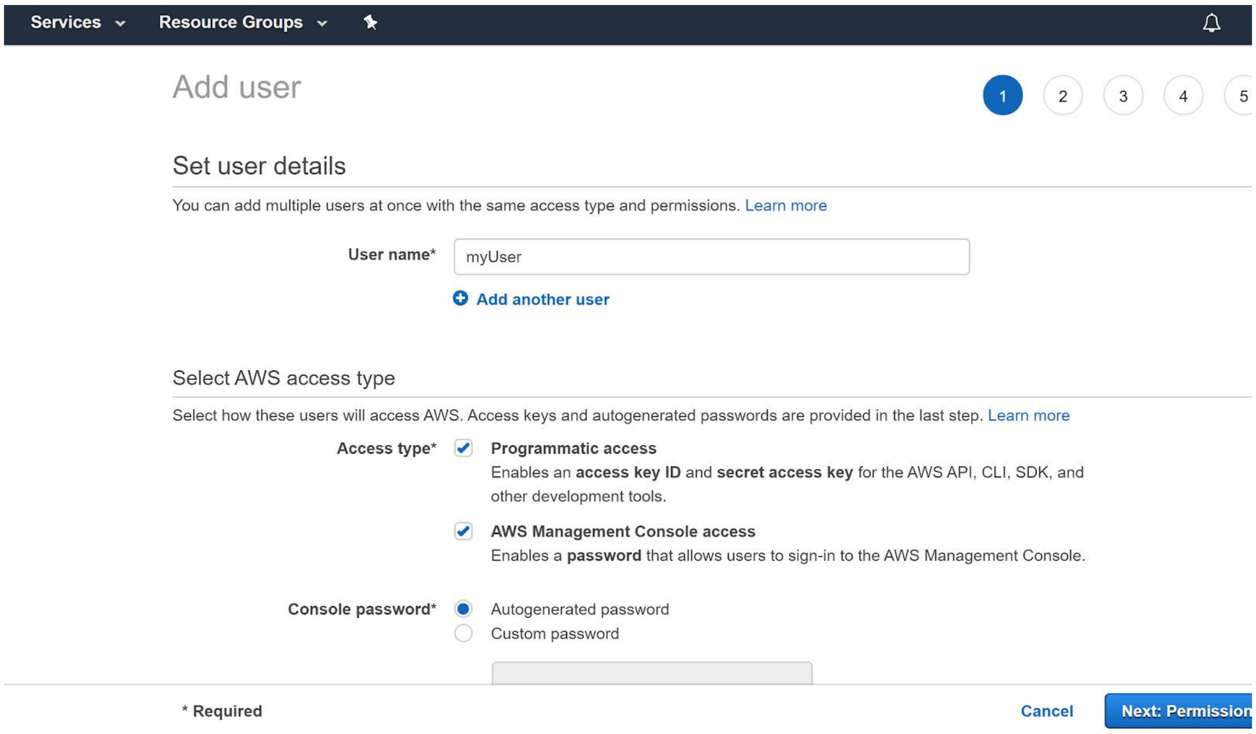
- AWS CLI (Install AWS CLI)
- Terraform (Install Terraform)

Step 1: AWS account setup and login

1. Setup AWS account if not already done
2. Login to your aws account

Step 2: AWS User creation, policy assignment and credentials setup

1. Go to services -> IAM -> Users -> Add user
2. Add user details



Add user

Set user details

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

User name* myUser

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

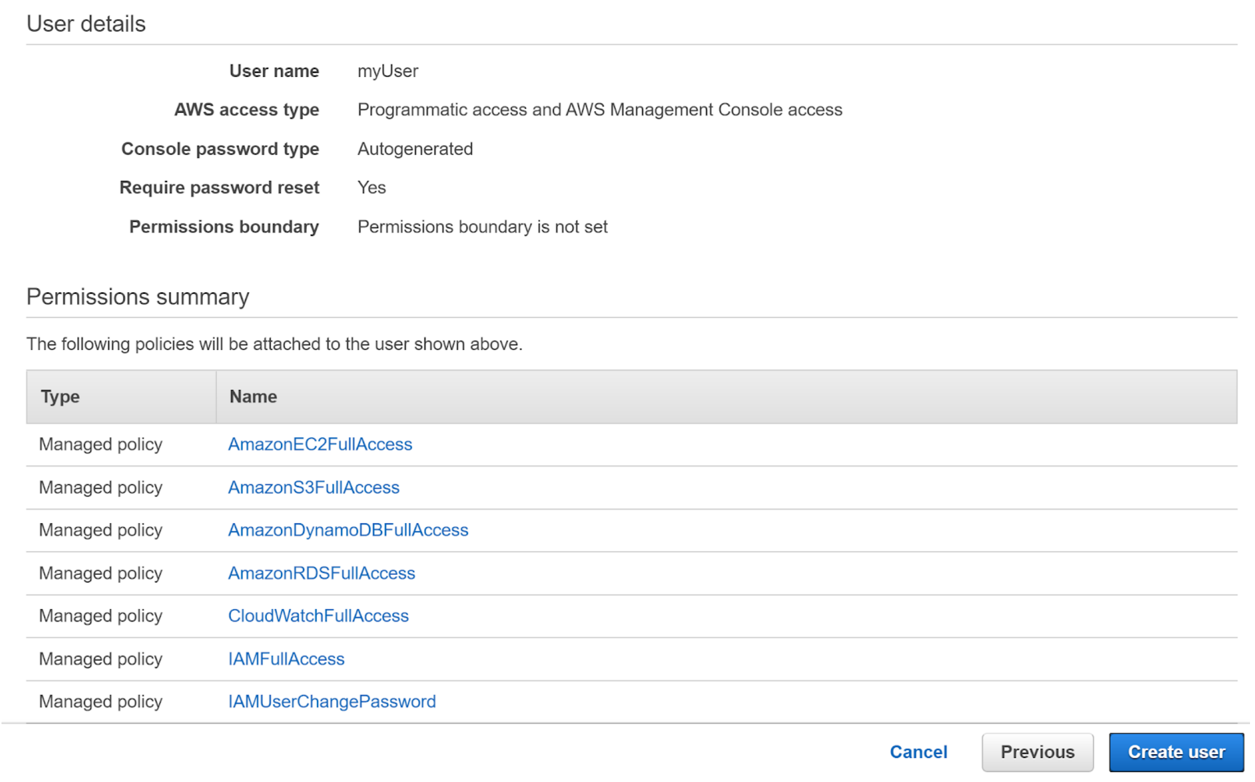
Access type* ✓ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

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

Console password* ● Autogenerated password
○ Custom password

3. Attach policies to this user

```
AmazonEC2FullAccess
AmazonS3FullAccess
AmazonDynamoDBFullAccess
AmazonRDSFullAccess
IAMFullAccess
CloudWatchFullAccess
```


User details

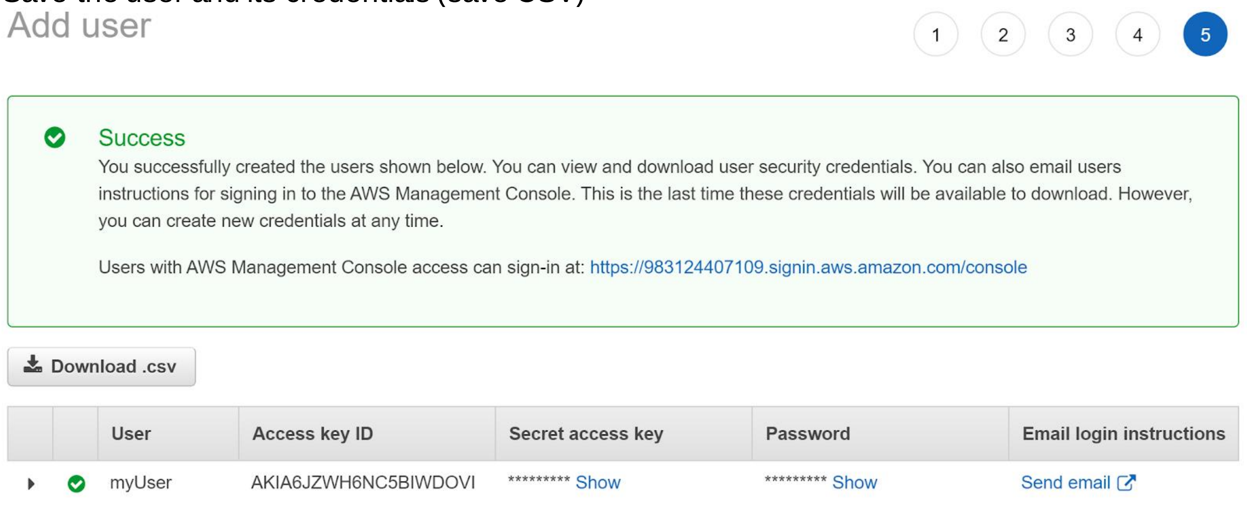
User name	myUser
AWS access type	Programmatic access and AWS Management Console access
Console password type	Autogenerated
Require password reset	Yes
Permissions boundary	Permissions boundary is not set

Permissions summary

The following policies will be attached to the user shown above.

Type	Name
Managed policy	AmazonEC2FullAccess
Managed policy	AmazonS3FullAccess
Managed policy	AmazonDynamoDBFullAccess
Managed policy	AmazonRDSFullAccess
Managed policy	CloudWatchFullAccess
Managed policy	IAMFullAccess
Managed policy	IAMUserChangePassword

4. Save the user and its credentials (save CSV)


Add user

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://983124407109.signin.aws.amazon.com/console

		User	Access key ID	Secret access key	Password	Email login instructions
▸	✓	myUser	AKIA6JZWH6NC5BIWDOVI	********* Show	********* Show	Send email

Step 3
Install Terraform (Manual Process)

1. Download the package in a location of your choice, from https://releases.hashicorp.com/terraform/0.12.26/terraform_0.12.26_linux_amd64.zip
2. Unzip this package
   unzip terraform_0.12.26_linux_amd64.zip
   3. Add the binary terraform path to PATH variable
   echo $PATH
   vi ~/.bashrc

Add line *export PATH = $PATH:<PATH_TO_YOURTERRAFORM_BINARY>*
source ~/.bashrc
4 Verify installation
Terraform -help

- Ansible (Install Ansible)

Defining SSH key-pair files

local-exec and remote-exec:

These two built in provisioners local-exec and remote-exec are required for Ansible to work in Terraform, as Terraform lacks the necessary native plug-ins. This is the workaround to invoke Ansible within the local-exec provisioner. That requires to **configure** the connection with the host, user, and private_key, see resource.tf for more details.

remote-exec

Python is required for Ansible to work, by using the "**remote-exec**" it makes sure that Python is installed before it's possible to invoke "**local-exec**"

local-exec

For Ansible, you can first run the Terraform, and output the IP addresses, then run ansible-playbook on those hosts

Description of various config files

Terraform

(1) Define Terraform version: terraform.tf

(2) Define AWS Provider: provider.tf

(3) Define AWS Resources: resources.tf

(4) Define Terraform Variables: variables.tf

(5) Define Terraform Outputs: outputs.tf

Ansible

(1) install_jenkins.yaml

(2) install_java.yaml

(3) install GitLab

Deploy Application

- terraform init
- terraform plan
- terraform apply
- terraform destroy