**Solution**

# TBistro

Terraform module which is used as a base to create EC2 instance(s) on AWS The module performs the following functionality:

* Looks-up AMI ID
* Creates default security group
* Creates instance IAM Role with policy ARN attachments
* Generates cloud-init user data
* Default tags
* Launches instance

**OS & Application Required**: VM Player with Ubuntu 18.04.3, AWS Account with Admin Access, RHEL 8.0 Ami(ami-087c2c50437d0b80d), Terraform, Ansible, Registered Domain for Route 53 and A Record, Putty and PuttyGen.

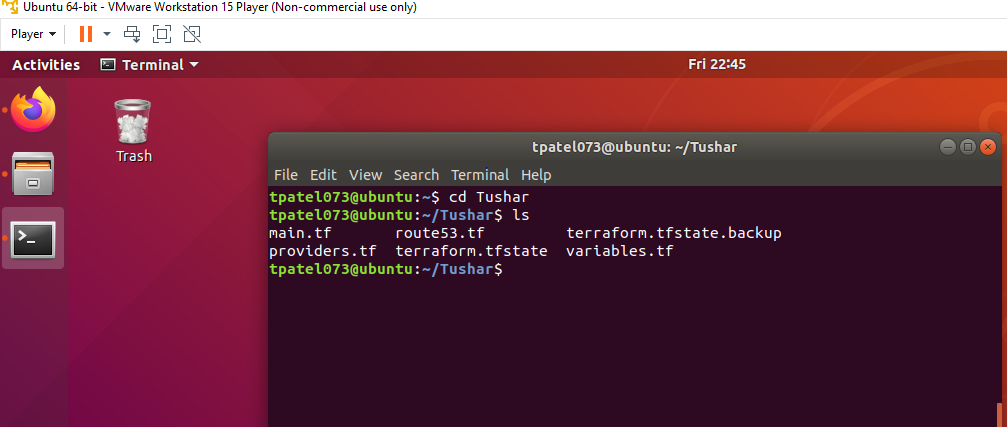
**1:)** Create Ec2 Instance using your Own Aws Account.

**2:)** Create the AWS Instance using Infrastructure as Code tool – Terraform

Created new IAM user \*\*1011 with Administrator Access and downloaded Access Key, Secret Access Key. Later **Access Key** Id and **Secret Access Key** are used for programmatic (API) **access** to AWS services.

From Network & Security in AWS Console, Created Key Pairs and download it. This key name will be embedded in main.tf for AWS instance creation. This key will be used later to access AWS instances via Putty (using puttygen convert \*. pem to \*.ppk)

For this Task, 2 RHEL 8.0 instances created in AWS via Terraform, First Ansible Controller (i-RHEL-Master) & Second Node (i-RHEL-Node). **Node will be managed by Master**. Terraform application downloaded, installed & Configured in Ubuntu 18.04.3 Desktop. Now Folder created for this project where all terraform files are kept and initialize Terraform.

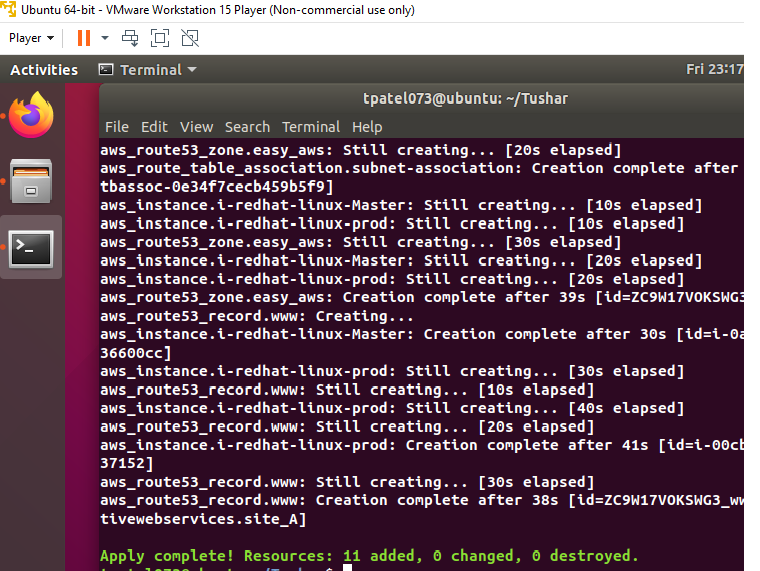


In Main terraform file VPC, Security groups & AWS instances are specified. Both Instance created in us-west-2 region. Region specified in **variable.tf** file along with Access Key ID and Secret Key Access Key ID.

**Providers Terraform file** is used to interact with the many resources supported by AWS. The AWS provider offers a flexible means of providing credentials for authentication. The following methods are supported (Static credentials, Environment variables, Shared credentials file, EC2 Role)

In Ubuntu Desktop, using terminal Run **Terraform init** command: Its used to initialize working directory containing terraform configuration files. Run **Plan**, terraform validate command, validates the configuration files in a directory, referring only to the configuration and not accessing any remote services, ex – Remote State, Provider APIs etc. **Run Terraform apply** command to apply the changes required to reach the desired state of configuration.

AWS Instances along with Security Group, VPC, Route53 and A Record created. For Security Purpose **Restrict access** to i**-RHEL-Node Instance** Inbound connection**(SSH)** from all IPs Except Public IP of Local PC, Public/Private IP of Master Node. Inbound HTTP connection open for all IPS until HTTPS Configured. SSH are Keep Open for Public IP of Desktop PC &Public/ Private IP of Master node for configuration & Testing Purpose and HTTP for nginx web page.



**3:)** Configure EC2 Instance (i-RHEL-Node) to be basic nginx server using Configuration Management Tool Ansible.

Install Ansible, Configured and established connectivity (create & Share SSH Key) between i-RHEL-Master (Ansible Controller) & i-RHEL-Node for passwordless authentication. Added Private IP of i-RHEL-Node in (Master – vi /etc/ansible/hosts) to run ansible playbook on i-RHEL-Node. In Master Node Created Folder TBistro and inside created File nginx.yml and run **ansible-playbook nginx.yml**. Nginx server installed successfully on i-RHEL-Node

-----------------------

Created Webserver as group of hosts in Master Node. Example vi /etc/ansible/hosts

[webserver]

Private IP of i-RHEL-Node

--------------------

**Tested Nginx web page using** public DNS IPV4 & public IP address of i-RHEL-Node Aws Instance

**4: ) Create internal Route 53 domain and A record using Terraform.**

This steps was completed with initial Terraform script in step1 (route53.tf) file created. Registered innovativewebservices.site in public domain.

From Aws console. Search for & Open route 53. **Public Hostedzone** innovativewebservices.site was already created with initial Terraform script. click on innovativewebservices.site. NS,SOA and A record will appear. Click on NS record 4 values will appear. Now open registered(innovativewebservices.site) public domain and open DNS Name Server, delete available Name server record on public domain and Add all 4 NS record from AWS Route 53.(it takes few hours to Sync)