SmartPension

Wednesday, September 30, 2020 7:20 PM

Environment Setup:

Step 1: Create a minimal centos 7 virtual machine to act as a Terraform control Node using VMware Workstation.

Step 2: Configure the Terraform control node VM by executing the following commands.

- Install git version control, and other goodies.

  # sudo yum -y install epel-release
  # sudo yum -y install wget unzip vim git -y

- Validate the proper installation of git. If no error output is produced then, git is properly installed.

  # git --version

Step 3: Setup Terraform.

- Download and unzip the 64 bit Terraform binary version 0.12.29 from HashiCorp.

  # wget https://releases.hashicorp.com/terraform/0.12.29/terraform_0.12.29_linux_amd64.zip
  # unzip terraform_0.12.29_linux_amd64.zip

- Move the 64 bit Terraform binary to /usr/local/bin/

  # sudo mv terraform /usr/local/bin/

- Validate the proper installation of Terraform. If no error output is produced then, Terraform is properly installed.

  # terraform --version

Step 4: Setup Ansible and AWS CLI

- Download and install pip3 (python's package installer) using yum for centos 7:

  # sudo yum -y install python3-pip

- Use pip3 to install AWS CLI. (pip install [package_name].

  # pip3 install awscli --user

- Use pip3 to install Ansible (pip install [package_name] and validate.

  # pip3 install ansible --user

- Download a preconfigured Ansible config file from cgpeanut/smartPension repo:

  # cd $HOME
  # mkdir code
  # git clone https://github.com/cgpeanut/smartPension.git
  # cd smartPension
  # cat ansible.cfg

- Verify that AWS CLI and Ansible are installed properly.

  # aws --version
  # ansible --version

```
[rroxas@geopoiesis smartPension]$ ansible --version
ansible 2.9.13
  config file = /home/rroxas/code/smartPension/ansible.cfg
  configured module search path = ['/home/rroxas/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/rroxas/.local/lib/python3.6/site-packages/ansible
  executable location = /home/rroxas/.local/bin/ansible
  python version = 3.6.8 (default, Apr  2 2020, 13:34:55) [GCC 4.8.5 20150623 (Red Hat 4.8.5-39)]
[rroxas@geopoiesis smartPension]$
```

- Configure AWS CLI with our free AWS account credentials and test whether the integration works properly.

  # aws configure

```
[rroxas@geopoiesis smartPension]$ aws configure
AWS Access Key ID [****************ILXY]:                                    LXY
AWS Secret Access Key [****************KOKD]:                                    KD
Default region name [us-east-1]: us-east-1
Default output format [json]: json
[rroxas@geopoiesis smartPension]$ aws ec2 describe-instances
{
    "Reservations": []
}
[rroxas@geopoiesis smartPension]$
```

Step 5: Setup AWS IAM permissions for Terraform:

- Grant Terrafrom permissions to create, update and delete various AWS resources either by:
  - Create a separate IAM user with the required permissions.

Let's use strict granular permissions:

- Copy https://github.com/cgpeanut/smartPension/blob/master/strict_terraform_deployment_iam_policy.json to buffer.
- Login to your AWS console and choose the IAM console.
- Cut and paste the strict_terraform_deployment_iam_policy.json to AWS IAM -> Policies -> Create Policy (JSON) - Review and name it SmartPensionUserPolicy.json

Create policy

1 2

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

Visual editor | JSON | Import managed policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CustomPolicyForACGAWSTFCourse",
      "Action": [
        "ec2:Describe*",
        "ec2:Get*",
        "ec2:AcceptVpcPeeringConnection",
        "ec2:AttachInternetGateway",
        "ec2:AssociateRouteTable",
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateInternetGateway",
        "ec2:CreateNetworkAcl",
        "ec2:CreateNetworkAclEntry",
        "ec2:CreateRoute",
        "ec2:CreateRouteTable",
        "ec2:CreateSecurityGroup",
        "ec2:CreateSubnet",
        "ec2:CreateTags",
        "ec2:CreateVpc",
        "ec2:CreateVpcPeeringConnection",
        "ec2:DeleteNetworkAcl",
        "ec2:DeleteNetworkAclEntry",
        "ec2:DeleteRoute",
        "ec2:DeleteRouteTable",
        "ec2:DeleteSecurityGroup",
        "ec2:DeleteSubnet",
        "ec2:DeleteTags",
        "ec2:DeleteVpc",
        "ec2:DeleteVpcPeeringConnection",
        "ec2:DetachInternetGateway",
        "ec2:DisassociateRouteTable",
        "ec2:DisassociateSubnetCidrBlock",
        "ec2:CreateKeyPair",
```

Create policy

1 2

Review policy

Name* smartPensionUserPolicy

Use alphanumeric and '+=,.@-_' characters. Maximum 128 characters.

Description smartPension strict User Policy

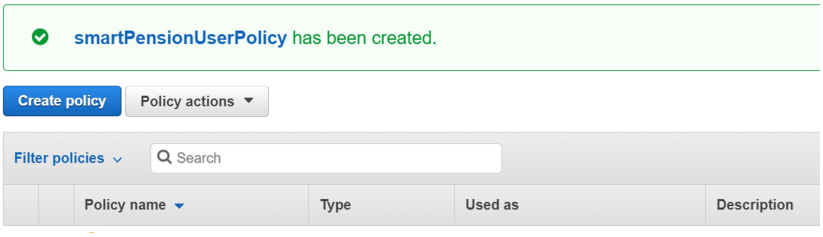
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Summary

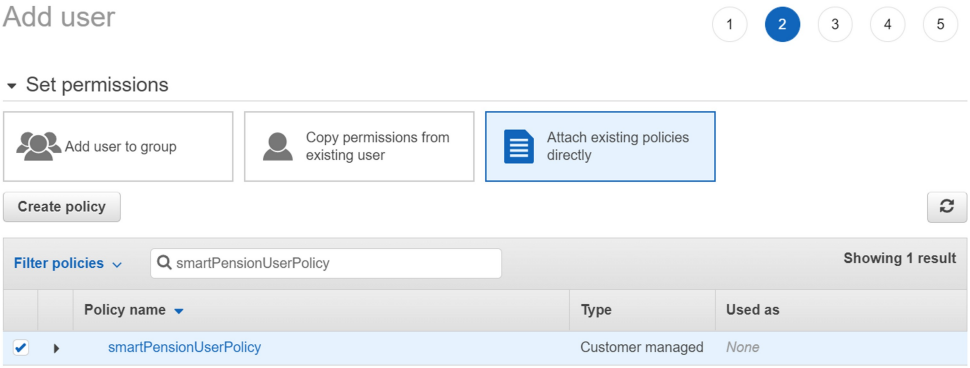
Filter

Service	Access level	Resource	Request condition
Allow (7 of 240 services) Show remaining 233			
Certificate Manager	Full access	All resources	None
EC2	**Full**: Read, Tagging **Limited**: List, Write	All resources	None
ELB	**Full**: List, Tagging **Limited**: Read, Write	All resources	None
ELB v2	**Full**: Tagging **Limited**: Read, Write	All resources	None
Route 53	**Full**: List **Limited**: Read, Write	All resources	None
S3	**Limited**: List, Read, Write	All resources	None
Systems Manager	**Limited**: List, Read	All resources	None

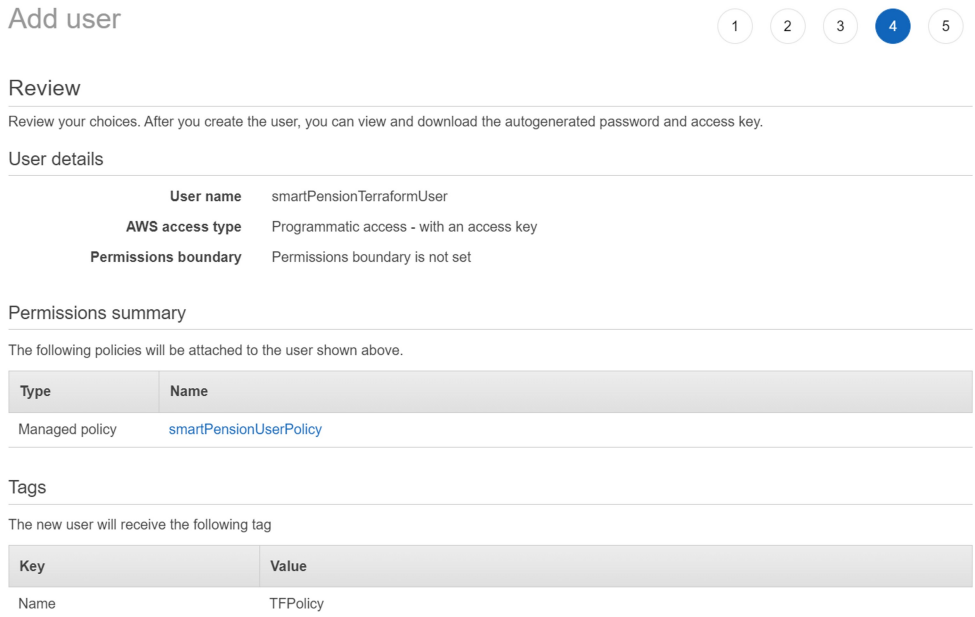
- Click Create Policy :)



- Now Create the smartPensionTerraformUser and attach the smartPensionUserPolicy to it.
  - Users -> Add user name: smartPensionTerraformUser
  - Click Programatic Access (no need for console access)
  - Click Next Permissions
  - Select Attach existing policies directly and locate smartPensionUserPolicy
  -



  - Next Tags: Best practice, Key: Name Value: TFPolicy
  - Next Review: and Click Create User.



User: smarrtPensionTerraformUser
Access key ID: super secret access ID
Secret access key: super secret access key

Step 6: Persisting Terraform State in S3 backend (requirement #3).

- Create the s3 bucket using AWS CLI

```
# cd $HOME/code/smartPension
# aws configure (input your credentials)
```

aws s3api create-bucket --bucket smartpensionpersistantbucket6315
place in backend.tf file bucket = "smartpensionpersistantbucket6315"

```
[rroxas@geopoiesis smartPension]$ aws s3api create-bucket --bucket smartpensionpersistantbucket6315
{
    "Location": "/smartpensionpersistantbucket6315"
}
```

```
# Set S3 backend for persisting TF state file remotely, ensure bucket already exits
# And that AWS user being used by Terraform has read/write permissions.

terraform {
  required_version = ">=0.12.0"
  required_providers {
    aws = ">=3.0.0"
  }
  backend "s3" {
    region  = "us-east-1"
    profile = "default"
    key     = "terraformstatefile"
    bucket  = "smartpensionpersistantbucket6315"
  }
}
```

- Validate s3 bucket by issuing the terraform init and fmt command to initialize and beautify the .tf files.

  # terraform init
  # terraform fmt

```
[rroxas@geopoiesis smartPension]$ terraform init

Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (hashicorp/aws) 3.8.0...

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.
```

We have successfully configured our persistent Terraform S3 backend.

Step 7: Create the variables.tf and providers.tf files

```
jenkins.tlp - rroxas@192.168.214.138:22 - Bitvise xterm - rroxas@geopoiesis:~/c
variable "profile" {
  type    = string
  default = "default"
}

variable "region-master" {
  type    = string
  default = "us-east-1"
}

variable "region-worker" {
  type    = string
  default = "us-west-2"
}

variable "external_ip" {
  type    = string
  default = "0.0.0.0/0"
}
~
~
~
```

Note: variables.tf has the regions defined in this case, us-east-1, providers.tf can use the values in variables.tf as alias to the variables "profile", "region-master", etc.

terraform init to validate and make sure there's no syntax errors.

Step 8: VPC Network setup with Terraform

- Create the networks.tf file located:

  https://github.com/cgpeanut/smartPension/blob/master/networks.tf

- Initialize, Validate, Beautify and check your terraform codes.
  - # terraform init
  - # terraform fmt
  - # terraform validate
  - # terraform plan
  - # terraform apply

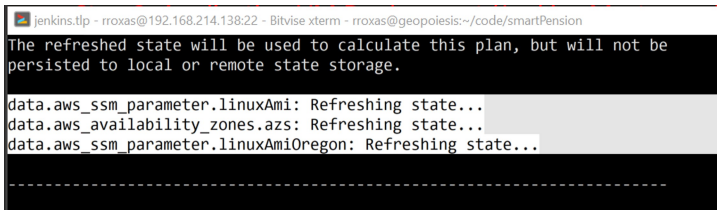
```
Plan: 13 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_vpc.vpc_master: Creating...
aws_vpc.vpc_master_oregon: Creating...
aws_vpc.vpc_master: Creation complete after 4s [id=vpc-0c1ce917f55a8bb6d]
aws_internet_gateway.igw: Creating...
aws_subnet.subnet_1: Creating...
aws_subnet.subnet_2: Creating...
aws_subnet.subnet_2: Creation complete after 2s [id=subnet-0b91fec8e2ad2912d]
aws_subnet.subnet_1: Creation complete after 2s [id=subnet-08a006cbf59d9850b]
aws_internet_gateway.igw: Creation complete after 2s [id=igw-016f606cc14a911cd]
aws_vpc.vpc_master_oregon: Creation complete after 7s [id=vpc-09ffb83063807231a]
aws_internet_gateway.igw-oregon: Creating...
aws_subnet.subnet_1_oregon: Creating...
aws_vpc_peering_connection.useast1-uswest2: Creating...
aws_subnet.subnet_1_oregon: Creation complete after 2s [id=subnet-067d6ed808164b31c]
aws_internet_gateway.igw-oregon: Creation complete after 3s [id=igw-05084fca6312f655a]
aws_vpc_peering_connection.useast1-uswest2: Creation complete after 10s [id=pcx-0949667827dc7d5df]
aws_vpc_peering_connection_accepter.accept_peering: Creating...
aws_route_table.internet_route: Creating...
aws_route_table.internet_route_oregon: Creating...
aws_route_table.internet_route: Creation complete after 2s [id=rtb-09a02c399c762b413]
aws_main_route_table_association.set-master-default-rt-assoc: Creating...
aws_main_route_table_association.set-master-default-rt-assoc: Creation complete after 1s [id=rtbassoc-02a32c4f514ea8654]
aws_route_table.internet_route_oregon: Creation complete after 3s [id=rtb-0df2014b75f46ce99]
aws_main_route_table_association.set-worker-default-rt-assoc: Creating...
aws_vpc_peering_connection_accepter.accept_peering: Creation complete after 4s [id=pcx-0949667827dc7d5df]
aws_main_route_table_association.set-worker-default-rt-assoc: Creation complete after 1s [id=rtbassoc-02fdf5d2cc12e8993]

Apply complete! Resources: 13 added, 0 changed, 0 destroyed.

Outputs:

PEERING-CONNECTION-ID = pcx-0949667827dc7d5df
VPC-ID-US-EAST-1 = vpc-0c1ce917f55a8bb6d
VPC-ID-US-WEST-2 = vpc-09ffb83063807231a
[rroxas@geopoiesis smartPension]$
```

Step 9: Application VM Deployment (Jenkins Master and Worker Nodes + HTTP Requirements)

Fetch AMI IDs Using SSM (AWS Systems Manager) Parameter Store Here's the documentation link. This will install Jenkins in a CI/CD pipeline we sill use Aws Systems Manager (SSM) and terraform

http://docs.aws.amazon.com/systems-manager/latest/userguide/parameter-store-public-parameter.html

- Create a file called instances.tf



- Deploy EC2 Key Pairs for the App Nodes so we can get ssh access to the application nodes.

Note: EC2 Keypair need to be created & attached to an EC2 instance before the EC2 instances themselves are launched, because they are baked into the instance at boot time!

We will do this using Terraform.

ssh-keygen - t rsa

```
rroxas@geopoiesis smartPension]$ ssh-keygen -t rsa
enerating public/private rsa key pair.
nter file in which to save the key (/home/rroxas/.ssh/id_rsa):
reated directory '/home/rroxas/.ssh'.
nter passphrase (empty for no passphrase):
nter same passphrase again:
our identification has been saved in /home/rroxas/.ssh/id_rsa.
our public key has been saved in /home/rroxas/.ssh/id_rsa.pub.
he key fingerprint is:
HA256:2pi0NHbIbuxr9hHF1T8R9Dx6JWjD0MG77sc0mTCbQQs rroxas@geopoiesis.rcubed.com
he key's randomart image is:
---[RSA 2048]----+
         .ooo.o.|
        .E++..o.|
         oo=o.o=|
     . . . .*..o+|
      B S    O +.|
     * O .  + *  |
      O o  . o .|
     oo  .  . o  |
     ooo.  ...   |
----[SHA256]-----+
rroxas@geopoiesis smartPension]$
```

- Add the newly created keys to instances.tf

jenkins.tlp - rroxas@192.168.214.138:22 - Bitvise xterm - rroxas@geopoiesis:~/code/smartPension

```
#Get Linux AMI ID using SSM Parameter endpoint in us-east-1
data "aws_ssm_parameter" "linuxAmi" {
  provider = aws.region-master
  name     = "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
}

#Get Linux AMI ID using SSM Parameter endpoint in us-west-2
data "aws_ssm_parameter" "linuxAmiOregon" {
  provider = aws.region-worker
  name     = "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
}

#Create key-pair for logging into EC2 in us-east-1
resource "aws_key_pair" "master-key" {
  provider   = aws.region-master
  key_name   = "jenkins"
  public_key = file("~/.ssh/id_rsa.pub")
}

#Create key-pair for logging into EC2 in us-west-2
resource "aws_key_pair" "worker-key" {
  provider   = aws.region-worker
  key_name   = "jenkins"
  public_key = file("~/.ssh/id_rsa.pub")
}
~
```

- Validate by executing:

terraform fmt
terraform validate
terraform plan
terraform apply

```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_key_pair.master-key will be created
  + resource "aws_key_pair" "master-key" {
      + arn         = (known after apply)
      + fingerprint = (known after apply)
      + id          = (known after apply)
      + key_name    = "jenkins"
      + key_pair_id = (known after apply)
      + public_key  = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDmM3eNopr/3Dn/agX4BuGIj8KFa5VpPJV0or43Bt3FXPlB7dTpTRyjqco9nUCSQk6506klb3mkQAGB+p7jy2M00
mlaPakkVo49kJsUCajSGVIBmv9Km3adbjSCyvkrhfIM3r/c7p9HopG3UtabGJ2V4g9GmIZ/Boll6gKYlh96/RikSa2ZaUS9lODYaH5orrd8y6SLXdzscPgg9qHagtqW3Sg4MLpKiQCQGdQ+wyl7Y
6eHiQ32BcdM3f0RgaZOp+L/ozY985uroyXZHrOqTIJxGY+TKxIqMJd/5avLY/ee1v+i347pnzle7cxKHJo+IEryvVTGegLe8/rRv25Gm88l rroxas@geopoiesis.rcubed.com"
    }

  # aws_key_pair.worker-key will be created
  + resource "aws_key_pair" "worker-key" {
      + arn         = (known after apply)
      + fingerprint = (known after apply)
      + id          = (known after apply)
      + key_name    = "jenkins"
      + key_pair_id = (known after apply)
      + public_key  = "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDmM3eNopr/3Dn/agX4BuGIj8KFa5VpPJV0or43Bt3FXPlB7dTpTRyjqco9nUCSQk6506klb3mkQAGB+p7jy2M00
mlaPakkVo49kJsUCajSGVIBmv9Km3adbjSCyvkrhfIM3r/c7p9HopG3UtabGJ2V4g9GmIZ/Boll6gKYlh96/RikSa2ZaUS9lODYaH5orrd8y6SLXdzscPgg9qHagtqW3Sg4MLpKiQCQGdQ+wyl7Y
6eHiQ32BcdM3f0RgaZOp+L/ozY985uroyXZHrOqTIJxGY+TKxIqMJd/5avLY/ee1v+i347pnzle7cxKHJo+IEryvVTGegLe8/rRv25Gm88l rroxas@geopoiesis.rcubed.com"
    }

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

- Let's cleanup so we don't get charged to much, wifey will dis-approve I'm using her credit card! :0

  # terraform destroy --auto-approve

Step 10: Application deployment: Deploying the Jenkins Master and Worker EC2 Instances. We need this to be able to deploy ansible via ssh (better than creating a puppetmaster)

  # vim instances.tf

- Add the code that creates the EC2 instances in us-east-1 and us-west-2

```
# Get Linux AMI ID using SSM Parameter endpoint in us-east-1
data "aws_ssm_parameter" "linuxAmi" {
  provider = aws.region-master
  name     = "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
}

# Get Linux AMI ID using SSM Parameter endpoint in us-west-2
data "aws_ssm_parameter" "linuxAmiOregon" {
  provider = aws.region-worker
  name     = "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
}

# Note: that this code will expect SSH key pair to exist in the user default directory
# otherwise it will Fail!

# Create key-pair for logging into EC2 in us-east-1
resource "aws_key_pair" "master-key" {
  provider   = aws.region-master
  key_name   = "jenkins"
  public_key = file("~/.ssh/id_rsa.pub")
}

# Create key-pair for logging into EC2 in us-west-2
resource "aws_key_pair" "worker-key" {
  provider   = aws.region-worker
  key_name   = "jenkins"
  public_key = file("~/.ssh/id_rsa.pub")
}

# Create and bootstrap EC2 in us-east-1
resource "aws_instance" "jenkins-master" {
  provider                    = aws.region-master
  ami                         = data.aws_ssm_parameter.linuxAmi.value
  instance_type               = var.instance-type
  key_name                    = aws_key_pair.master-key.key_name
  associate_public_ip_address = true
  vpc_security_group_ids      = [aws_security_group.jenkins-sg.id]
  subnet_id                   = aws_subnet.subnet_1.id

  tags = {
    Name = "jenkins_master_tf"
  }

  depends_on = [aws_main_route_table_association.set-master-default-rt-assoc]
}

# Create EC2 in us-west-2
resource "aws_instance" "jenkins-worker-oregon" {
  provider                    = aws.region-worker
  count                       = var.workers-count
  ami                         = data.aws_ssm_parameter.linuxAmiOregon.value
  instance_type               = var.instance-type
  key_name                    = aws_key_pair.worker-key.key_name
  associate_public_ip_address = true
  vpc_security_group_ids      = [aws_security_group.jenkins-sg-oregon.id]
  subnet_id                   = aws_subnet.subnet_1_oregon.id

  tags = {
    Name = join("_", ["jenkins_worker_tf", count.index + 1])
  }
  depends_on = [aws_main_route_table_association.set-worker-default-rt-assoc, aws_instance.jenkins-master]
}
~
~
~
~
~
"instances.tf" 62L, 2258C                                                    62,1          All
```

- Add these code to the variables.tf file for better segregation

```
variable "workers-count" {
  type    = number
  default = 1
}

variable "instance-type" {
  type    = string
  default = "t3.micro"
}

###This chunk of template can also be put inside outputs.tf for better segregation
output "Jenkins-Main-Node-Public-IP" {
  value = aws_instance.jenkins-master.public_ip
}

output "Jenkins-Worker-Public-IPs" {
  value = {
    for instance in aws_instance.jenkins-worker-oregon :
    instance.id => instance.public_ip
  }
}
```

- Create a file called outputs.tf
  # vim outputs.tf

```
output "VPC-ID-US-EAST-1" {
  value = aws_vpc.vpc_master.id
}

output "VPC-ID-US-WEST-2" {
  value = aws_vpc.vpc_master_oregon.id
}

output "PEERING-CONNECTION-ID" {
  value = aws_vpc_peering_connection.useast1-uswest2.id
}

output "Jenkins-Main-Node-Public-IP" {
  value = aws_instance.jenkins-master.public_ip
}

output "Jenkins-Main-Node-Private-IP" {
  value = aws_instance.jenkins-master.private_ip
}

output "Jenkins-Worker-Public-IPs" {
  value = {
    for instance in aws_instance.jenkins-worker-oregon :
    instance.id => instance.public_ip
  }
}

output "Jenkins-Worker-Private-IPs" {
  value = {
    for instance in aws_instance.jenkins-worker-oregon :
    instance.id => instance.private_ip
  }
}
```

- Rinse and Repeat
  # terraform fmt
  # terraform validate
  # terraform plan
  # terraform apply

```
aws_instance.jenkins-worker-oregon[0]: Creation complete after 21s [id=i-0acc5e1aca363b22c]

Apply complete! Resources: 20 added, 0 changed, 0 destroyed.

Outputs:

Jenkins-Main-Node-Private-IP = 10.0.1.136
Jenkins-Main-Node-Public-IP = 100.24.58.184
Jenkins-Worker-Private-IPs = {
  "i-0acc5e1aca363b22c" = "192.168.1.11"
}
Jenkins-Worker-Public-IPs = {
  "i-0acc5e1aca363b22c" = "34.209.220.168"
}
PEERING-CONNECTION-ID = pcx-0e64956f336d3692e
VPC-ID-US-EAST-1 = vpc-0c671cb9a305ec27e
VPC-ID-US-WEST-2 = vpc-0c047aaddb72d35a4
[rroxas@geopoiesis smartPension]$
```

- Validate that we can login to the Jenkins-Main-Node-Public-ip of 100.24.58.184 using the ssh keys we generated.

```
[rroxas@geopoiesis smartPension]$ ls
ansible.cfg                instances.tf  providers.tf        strict_terraform_deployment_iam_policy.json
Architectural_Diagram.pdf  networks.tf   README.md          terraformstatefile
backend.tf                 outputs.tf    security_groups.tf  variables.tf
[rroxas@geopoiesis smartPension]$ ssh ec2-user@100.24.58.184
The authenticity of host '100.24.58.184 (100.24.58.184)' can't be established.
ECDSA key fingerprint is SHA256:7hJSZVGL8P87DLFpADQF2sDBacBa/KrPekFRbnrrvDo.
ECDSA key fingerprint is MD5:c1:6e:18:99:a5:69:c2:08:58:26:f9:b7:86:be:9f:ba.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '100.24.58.184' (ECDSA) to the list of known hosts.

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-1-136 ~]$
```

- Now, ssh to the public IP of our worker instance to validate our ssh keys that we coded in terraform.

```
[rroxas@geopoiesis smartPension]$ ssh ec2-user@34.209.220.168
The authenticity of host '34.209.220.168 (34.209.220.168)' can't be established.
ECDSA key fingerprint is SHA256:IELuMb4fxrDJ7kzrZLGrETBazVoTA/WsbS2UgSARrNM.
ECDSA key fingerprint is MD5:b2:dc:89:76:ce:72:ec:20:06:6f:4e:74:4a:b4:92:dd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '34.209.220.168' (ECDSA) to the list of known hosts.

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-192-168-1-11 ~]$
```

Step 11: Bootstrapping infrastructure via Terraform Provisioners
ansible playbook creation sample for now, will install jenkins later.

- Create a directory to house our ansible templates
  # cd $HOME/code/smartPension
  # mkdir ansible_templates
  # cd ansible_templates

- Create a jenkins-master.yaml-sample and jenins-worker-sample.yml file to install git client and jq as root

  # vim jenkins-master-sample.yml - install git for now,

```
#Ansible Jenkins Master, sample playbook - jenkins-master-sample.yml
---
- hosts: "{{ passed_in_hosts }}"
  become: yes
  remote_user: ec2-user
  become_user: root
  tasks:
    - name: install Git client
      yum:
        name: git
        state: present
~
```

```
#Ansible Jenkins Worker, sample playbook - jenkins-worker-sample.yml
---
- hosts: "{{ passed_in_hosts }}"
  become: yes
  remote_user: ec2-user
  become_user: root
  tasks:
    - name: install jq, JSON parser
      yum:
        name: jq
        state: present
~
```

- Edit the $HOME/code/smartPension/ansible.cfg file to enable dynamic inventory querying

```
[defaults]
inventory       = ./ansible_templates/inventory_aws/tf_aws_ec2.yml
enable_plugins = aws_ec2
interpreter_python = auto_silent
#library         = ~/.ansible/plugins/modules:/usr/share/ansible/plugins/modules
#module_utils    = ~/.ansible/plugins/module_utils:/usr/share/ansible/plugins/module_utils
#remote_tmp      = ~/.ansible/tmp
#local_tmp       = ~/.ansible/tmp
#forks           = 5
```

- create a separate directory in $HOME/code/smartPension/ansible_templates named inventory_aws

  # mkdir $HOME/code/smartPension/ansible_templates/inventory_aws
  # cd $HOME/code/smartPension/ansible_templates/inventory_aws

- Cut and paster the tf_aws.ec2.yml file the github repository

  # https://github.com/cgpeanut/smartPension/blob/master/ansible_templates/inventory_aws/tf_aws_ec2.yml

  # vim $HOME/code/smartPension/ansible_templates/inventory_aws/tf_aws_ec2.yml

- Install boto3 using pip3 in our jenkins control node VM.

  # pip3 install boto3 --user

```
[rroxas@geopoiesis inventory_aws]$ pip3 install boto3 --user
Collecting boto3
  Downloading https://files.pythonhosted.org/packages/33/73/e90df0e0a5c55291273697d8eaa33f920b10590b5f632e971bb68ec55988/
boto3-1.15.10-py2.py3-none-any.whl (129kB)
    100% |████████████████████████████████| 133kB 3.0MB/s
Collecting botocore<1.19.0,>=1.18.10 (from boto3)
  Downloading https://files.pythonhosted.org/packages/cb/3f/e55f26983b7315265066f1cdc614e8d7e2c6a7a798ac6be88f3fdff90533/
botocore-1.18.10-py2.py3-none-any.whl (6.7MB)
    100% |████████████████████████████████| 6.7MB 268kB/s
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /home/rroxas/.local/lib/python3.6/site-packages (from boto3)
Requirement already satisfied: s3transfer<0.4.0,>=0.3.0 in /home/rroxas/.local/lib/python3.6/site-packages (from boto3)
Requirement already satisfied: urllib3<1.26,>=1.20; python_version != "3.4" in /home/rroxas/.local/lib/python3.6/site-pac
kages (from botocore<1.19.0,>=1.18.10->boto3)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /home/rroxas/.local/lib/python3.6/site-packages (from botoc
ore<1.19.0,>=1.18.10->boto3)
Requirement already satisfied: six>=1.5 in /home/rroxas/.local/lib/python3.6/site-packages (from python-dateutil<3.0.0,>=
2.1->botocore<1.19.0,>=1.18.10->boto3)
Installing collected packages: botocore, boto3
  Found existing installation: botocore 1.17.52
    Uninstalling botocore-1.17.52:
      Successfully uninstalled botocore-1.17.52
Successfully installed boto3-1.15.10 botocore-1.18.10
```

- cd to the instances.tf file
  # cd $HOME/code/smartPension
  # vim instances.tf

  https://github.com/cgpeanut/smartPension/blob/master/instances.tf

- Test the newly provisioned code (rinse and repeat)

  # terraform fmt
  # terraform validate
  # terraform plan
  # terraform apply

```
aws_main_route_table_association.set-worker-default-rt-assoc: Creation complete after 1s [id=rtbassoc-0f2da824417c624c0]
aws_vpc_peering_connection_accepter.accept_peering: Creation complete after 4s [id=pcx-0770ee8ce8883f528]
aws_instance.jenkins-master: Still creating... [10s elapsed]
aws_instance.jenkins-master: Provisioning with 'local-exec'...
aws_instance.jenkins-master (local-exec): Executing: ["/bin/sh" "-c" "aws --profile default ec2 wait instance-status-ok --region us-east-1 --instance-ids i-
0dbaa177ad2ecf846 \\\n&& ansible-playbook --extra-vars 'passed_in_hosts=tag_Name_jenkins_master_tf' ansible_templates/jenkins-master-sample.yml\n"]
aws_instance.jenkins-master: Still creating... [20s elapsed]
aws_instance.jenkins-master: Still creating... [30s elapsed]
aws_instance.jenkins-master: Still creating... [40s elapsed]
```

Magic of ec2 dynamic inventory (boto3)

Step 12: Creating and Application Load Balancer (ALB) and routing the traffic to an EC2 App Node via an Apache webserver deploying and EC2 instance behind a load balancer via terraform.

- Use ansible playbooks to customized the EC2 application nodes to run the Jenkins application.