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

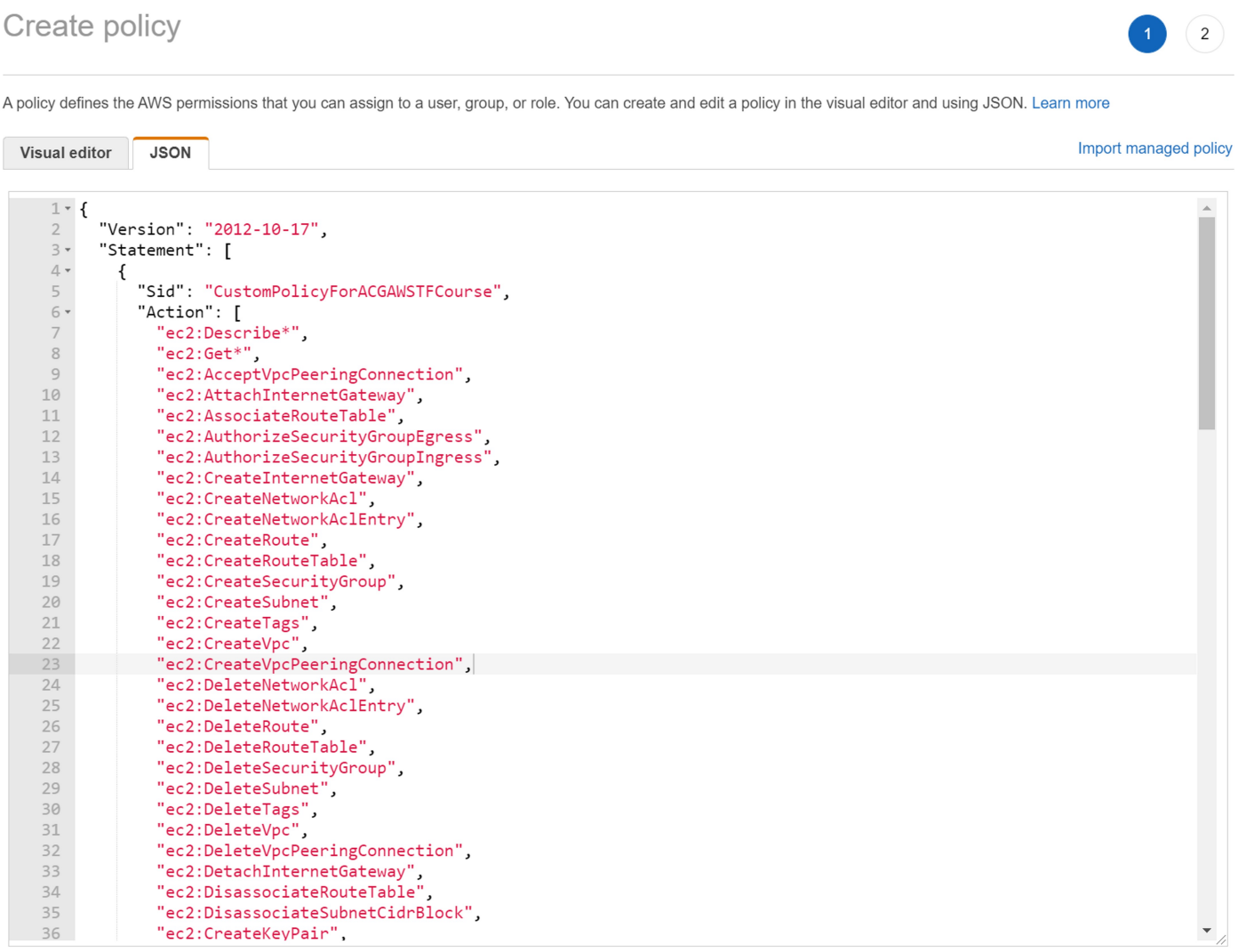
```
[rroxas@geopoiesis smartPension]$ aws configure
AWS Access Key ID [****************ILXY]:                    LXY
AWS Secret Access Key [****************KOKD]:                                    KD
Default region name [us-east-1]: us-east-1
Default output format [json]: json
[rroxas@geopoiesis smartPension]$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

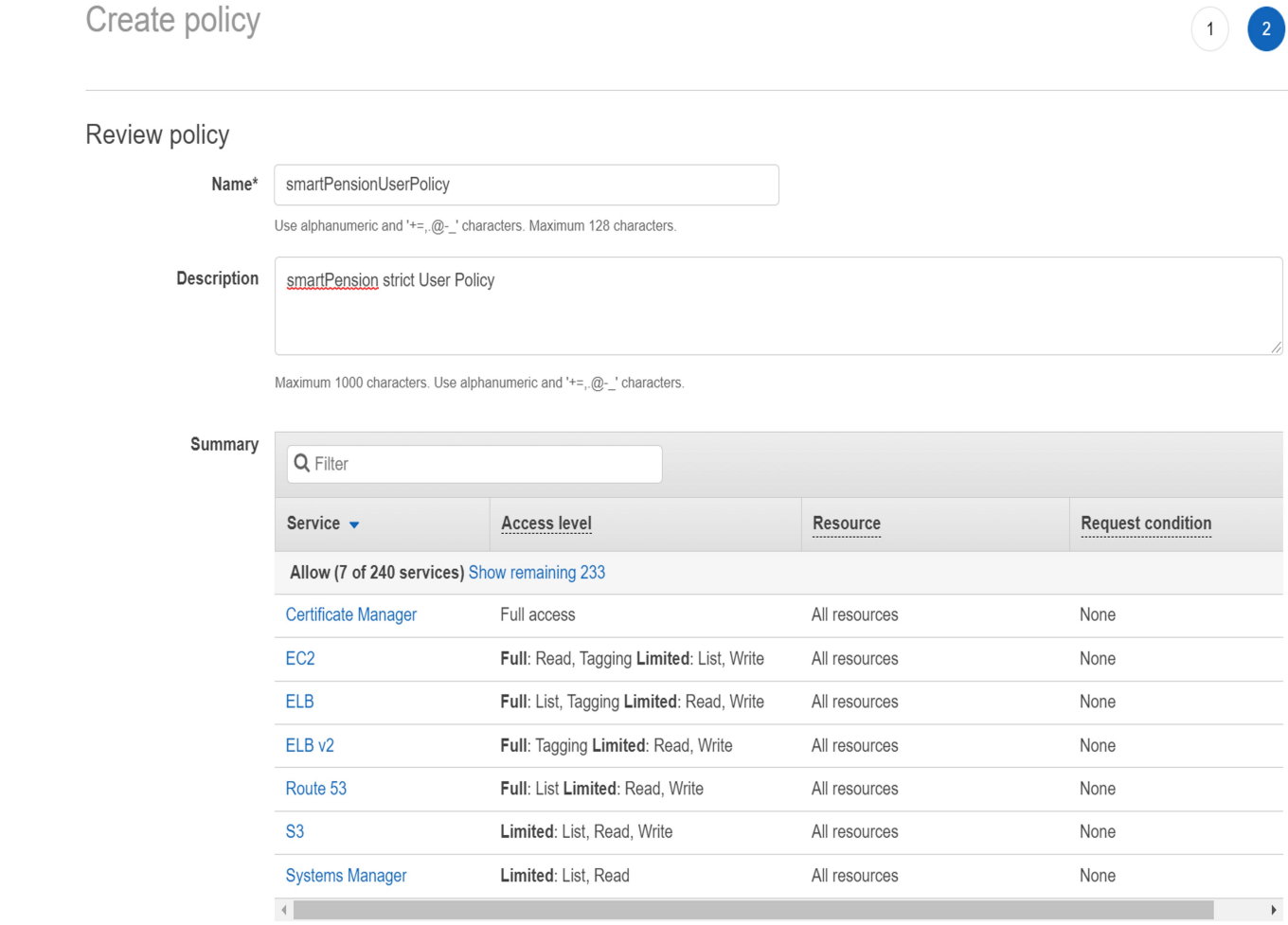
Step 5: Setup AWS IAM permissions for Terraform:

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

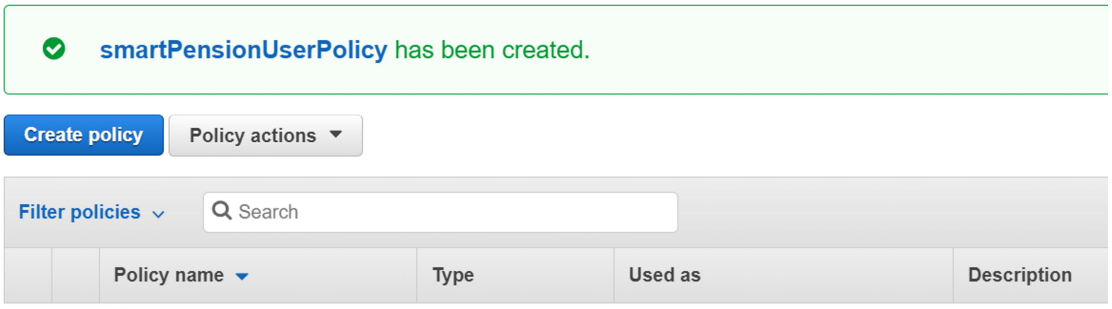
Let's use strict granular permissions:

- 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

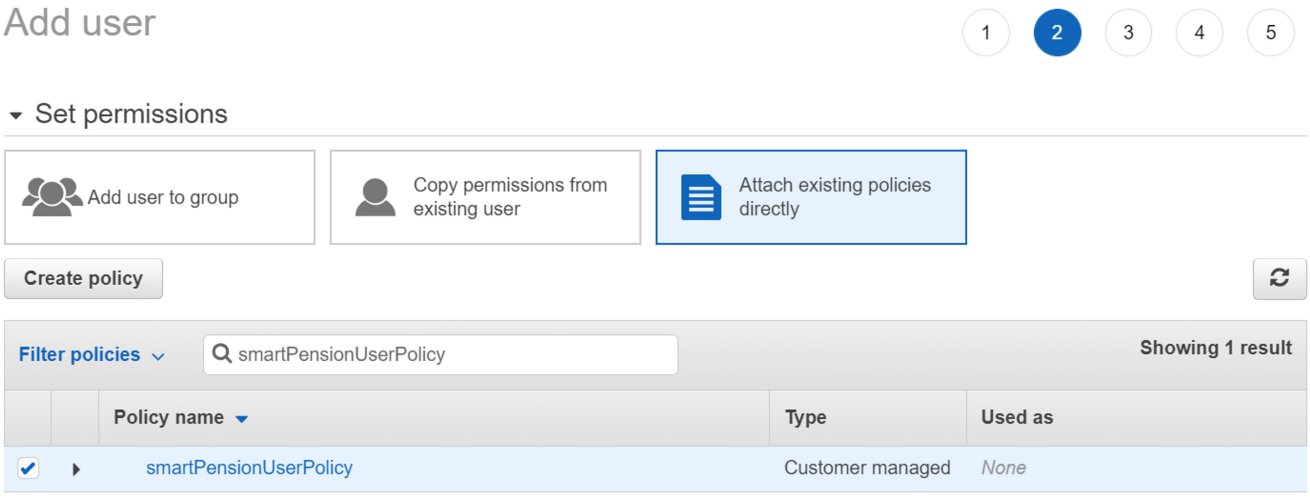




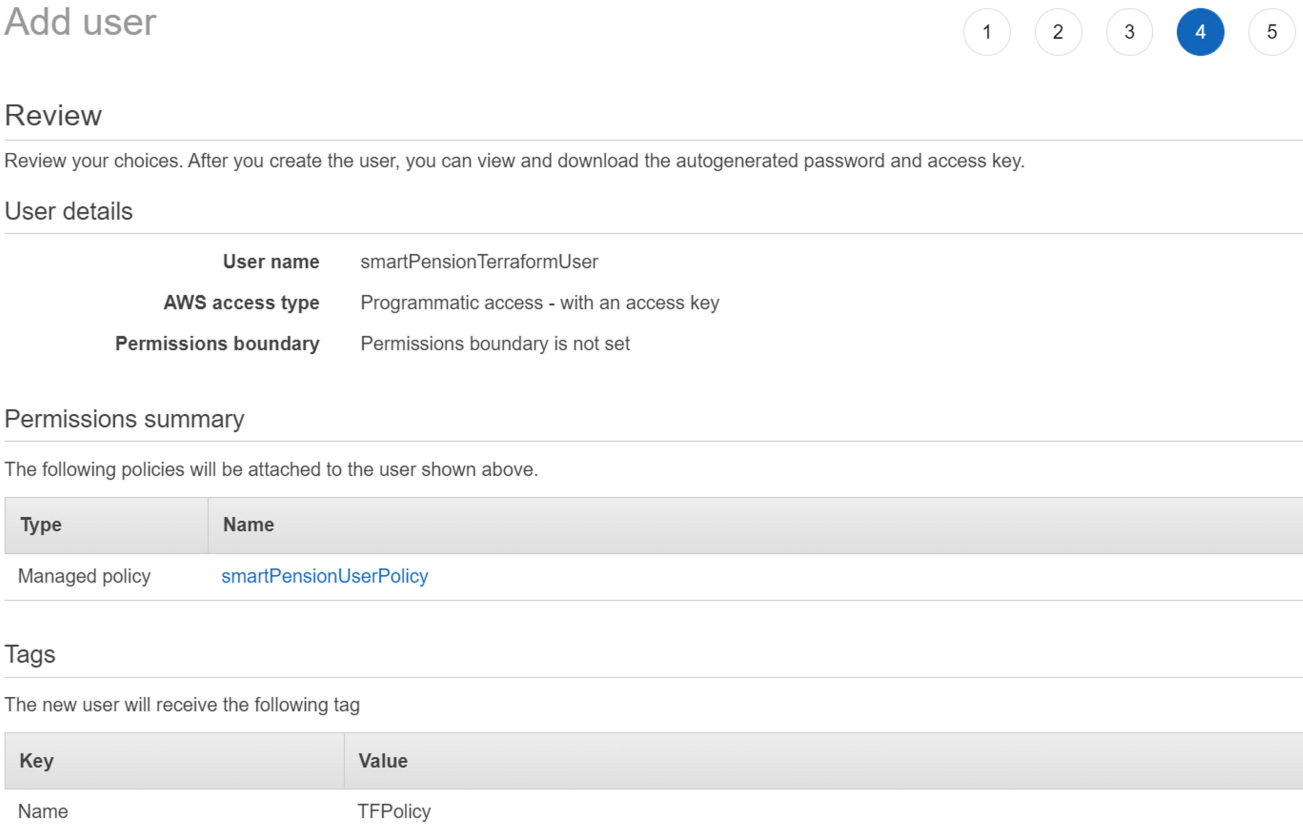
- Click Create Policy :)



- 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

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-0be89ffd1b6b1152d]
aws_internet_gateway.igw: Creating...
aws_subnet.subnet_2: Creating...
aws_subnet.subnet_1: Creating...
aws_subnet.subnet_1: Creation complete after 2s [id=subnet-0f2c14d3b28a264c1]
aws_subnet.subnet_2: Creation complete after 2s [id=subnet-02c11f57fc5aaf9fa]
aws_internet_gateway.igw: Creation complete after 2s [id=igw-0082e58ec1486aa57]
aws_vpc.vpc_master_oregon: Creation complete after 7s [id=vpc-0e1ad5547dcd95cd3]
aws_vpc_peering_connection.useast1-uswest2: Creating...
aws_internet_gateway.igw-oregon: Creating...
aws_subnet.subnet_1_oregon: Creating...
aws_subnet.subnet_1_oregon: Creation complete after 2s [id=subnet-0c6fb06c850c472dd]
aws_internet_gateway.igw-oregon: Creation complete after 3s [id=igw-07fd27efba060de68]
aws_vpc_peering_connection.useast1-uswest2: Creation complete after 8s [id=pcx-0c45099ee2557753e]
aws_vpc_peering_connection_accepter.accept_peering: Creating...
aws_route_table.internet_route_oregon: Creating...
aws_route_table.internet_route: Creating...
aws_route_table.internet_route: Creation complete after 2s [id=rtb-024839e3388eb425f]
aws_main_route_table_association.set-master-default-rt-assoc: Creating...
aws_main_route_table_association.set-master-default-rt-assoc: Creation complete after 0s [id=rtbassoc-07871ffbd0d1bab0b]
aws_route_table.internet_route_oregon: Creation complete after 3s [id=rtb-06efd990b34bb509e]
aws_main_route_table_association.set-worker-default-rt-assoc: Creating...
aws_vpc_peering_connection_accepter.accept_peering: Creation complete after 3s [id=pcx-0c45099ee2557753e]
aws_main_route_table_association.set-worker-default-rt-assoc: Creation complete after 1s [id=rtbassoc-0fa09d8de42a98cc5]

Apply complete! Resources: 13 added, 0 changed, 0 destroyed.
[rroxas@geopoiesis smartPension]$
```

terraform apply

```
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: Creation complete after 4s [id=vpc-094e33dbc8772f418]
aws_subnet.subnet_1: Creating...
aws_subnet.subnet_1: Creation complete after 2s [id=subnet-0059595937b3cc440]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[rroxas@geopoiesis smartPension]$
```