Setup 2 EC2 Servers with Terraform and AWS AMI - Project report

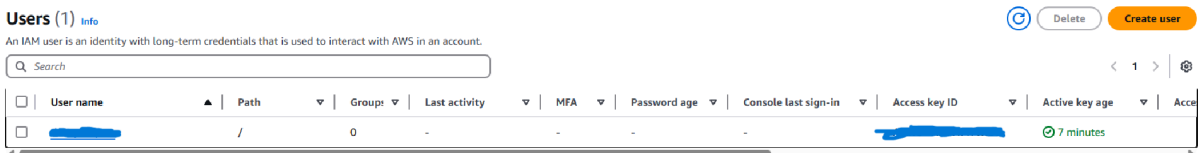
Terraform structure overview

Terraform files used in project:

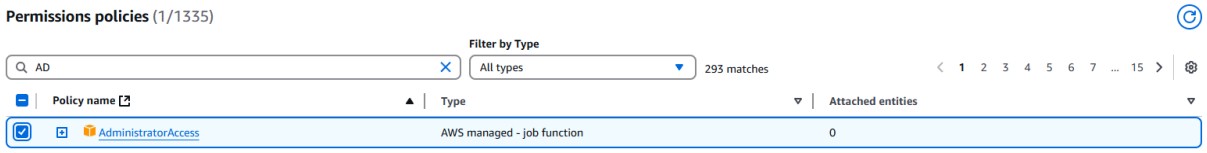
- *main.tf* – two EC2 creations based on AMI image, authorizing two public keys for SSH access stored in the */keys* folder and setting the firewall. For incoming connections available: 22 ports from all IPs, 5432 - 5435 only from EC2 servers created by a script, for outgoing - no restrictions.
- *variables.tf* – variables used in main script.
- *outputs.tf* – configuration responsible for printing newly created EC2 IPv4 addresses after successful script run.

AWS Prerequisites:

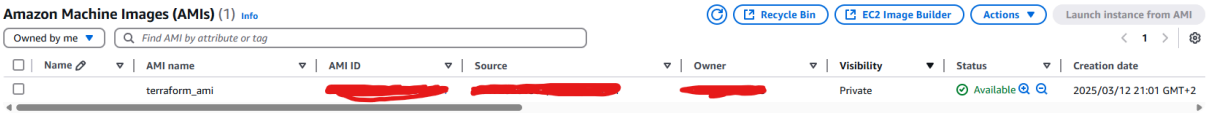
Step 1: Creating a separate IAM user, exporting his AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY for access:



Step 2: Giving user permission for creating EC2s from AMI image (giving an administrator access may not be best due to security reasons, might be better to fine-tune permissions to an exact task scope):



Step 3: Creating AMI instance with access granted to a user created in the *Step 1*:



Running the project:

After setting the correct parameters in the *variables.tf*, run the project with these commands:

```
terraform init

set AWS_ACCESS_KEY_ID=LEGITKEYID
set AWS_SECRET_ACCESS_KEY=LEGITTOPSECRETNUCLEARWARHEADLAUNCHKEY
set AWS_DEFAULT_REGION=eu-north-1
```

```
terraform apply -auto-approve
```

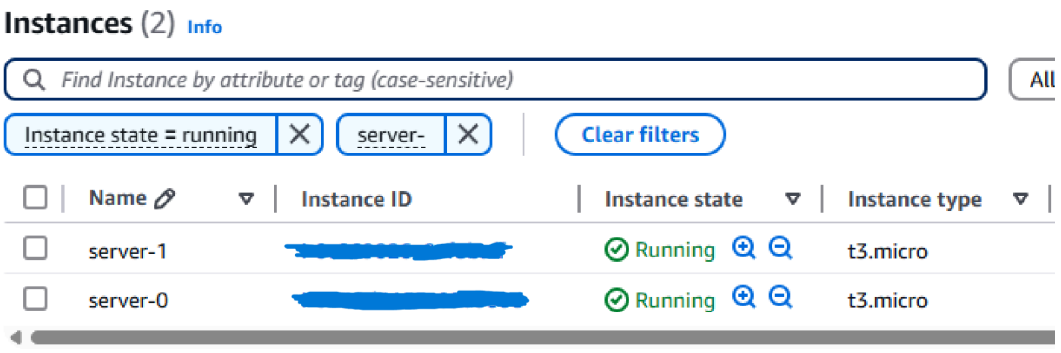
Successful run result - *terraform init*:

```
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.90.1...
- Installed hashicorp/aws v5.90.1 (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.
```

Successful run result - *terraform apply*:

```
Plan: 2 to add, 0 to change, 0 to destroy.
aws_security_group_rule.db_access[0]: Creating...
aws_security_group_rule.db_access[1]: Creating...
aws_security_group_rule.db_access[1]: Creation complete after 1s [id=sgrule-250479133]
aws_security_group_rule.db_access[0]: Creation complete after 1s [id=sgrule-1092762407]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

server_ips = [
  "16.16.213.199",
  "51.21.128.122",
]

C:\Users\Maksym\GitHub\terraform_ami>
```

Created EC2s are displayed in AWS:



Troubleshooting:

While creating instances using the script, I encountered issues with properly setting the credentials. These, and other types of issues can be more easily troubleshooted by storing debug logs in the file with these commands (create file first):

```
set TF_LOG=DEBUG
set TF_LOG_PATH=terraform-debug.log
```