DevOps and Cloud

March 2025

Terraform Fundamentals

Homework (M4)

Vasil Atanasov

@VasAtanasov

Environment Setup

A single VM will be set up with Vagrant. All tasks will be performed inside Debian VM.

Docker and Terraform installed on the Debian VM. This is convenient because we can leverage auto completion and generally, I feel better in the Linux Bash terminal than PowerShell.

Terraform and Docker

Remote Image with Local Mount

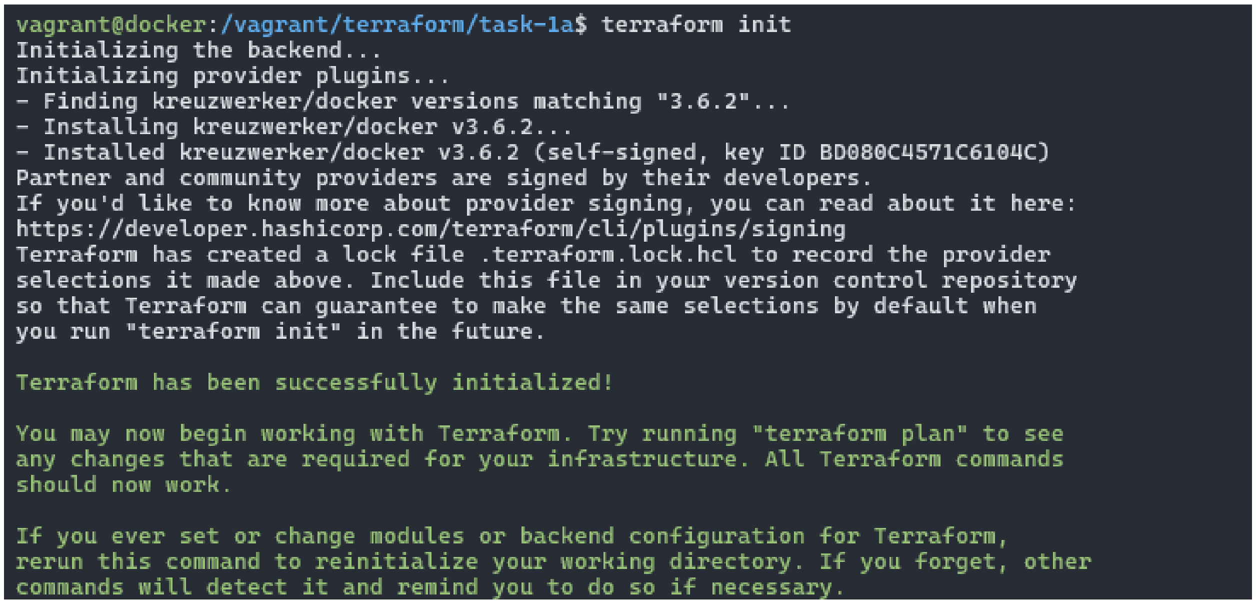
The files for the task are located inside terraform/task-1a folder. If we explore the folder we will see terraform code split into separate files: main.tf, variables.tf and terraform.tfvars.

First, we need to get the project. Clone the repo in terraform/task-1a. Later the a vagrant trigger will clean it up on destroy.

```
git clone https://github.com/shekeriev/bgapp.git
```

We will be using docker provider and to download it we need to initialize the project with:

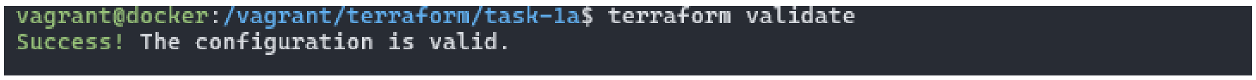
```
terraform init
```



```
vagrant@docker:/vagrant/terraform/task-1a$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "3.6.2"...
- Installing kreuzwerker/docker v3.6.2...
- Installed kreuzwerker/docker v3.6.2 (self-signed, key ID BD080C4571C6104C)
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://developer.hashicorp.com/terraform/cli/plugins/signing
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.
```

To validate that our configuration is correct we use:

```
terraform validate
```



```
vagrant@docker:/vagrant/terraform/task-1a$ terraform validate
Success! The configuration is valid.
```

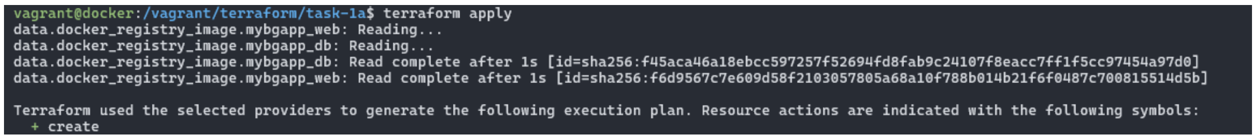
When we explore the main.tf file we will see the following definition:

```
data "docker_registry_image" "mybgapp_db" {
  name = "${var.mybgapp_db_image_repo}:${var.mybgapp_db_image_tag}"
}

resource "docker_image" "mybgapp_db" {
  name = data.docker_registry_image.mybgapp_db.name
  pull_triggers = [data.docker_registry_image.mybgapp_db.sha256_digest]
}
```

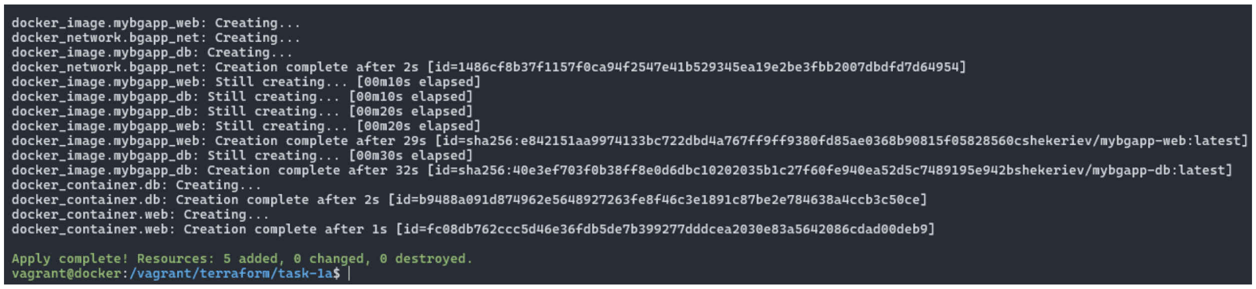
This gives the ability to update the image dynamically when there is a sha256 sum change. So, to work we need both docker_image resource and docker_image_registry data. The same applies for the mybgaap_web image.

To provision docker execute

```
terraform apply
```

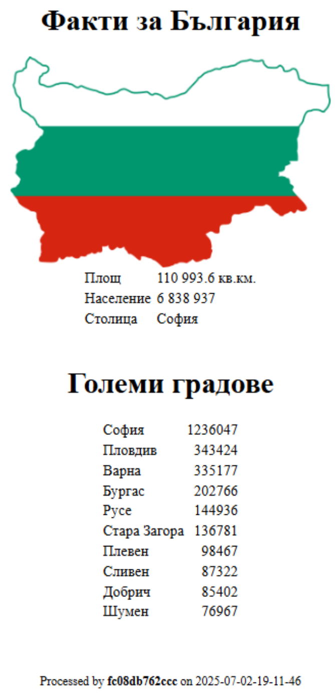


```
vagrant@docker:/vagrant/terraform/task-1a$ terraform apply
data.docker_registry_image.mybgapp_web: Reading...
data.docker_registry_image.mybgapp_db: Reading...
data.docker_registry_image.mybgapp_db: Read complete after 1s [id=sha256:f45aca46a18ebcc597257f52694fd8fab9c24107f8eacc7ff1f5cc97454a97d0]
data.docker_registry_image.mybgapp_web: Read complete after 1s [id=sha256:f6d9567c7e609d58f2103057805a68a10f788b014b21f6f0487c700815514d5b]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create
```



```
docker_image.mybgapp_web: Creating...
docker_network.bgapp_net: Creating...
docker_image.mybgapp_db: Creating...
docker_network.bgapp_net: Creation complete after 2s [id=1486cf8b37f1157f0ca94f2547e41b529345ea19e2be3fbb2007dbdfd7d64954]
docker_image.mybgapp_web: Still creating... [00m10s elapsed]
docker_image.mybgapp_db: Still creating... [00m10s elapsed]
docker_image.mybgapp_db: Still creating... [00m20s elapsed]
docker_image.mybgapp_web: Still creating... [00m20s elapsed]
docker_image.mybgapp_web: Creation complete after 29s [id=sha256:e842151aa9974133bc722dbd4a767ff9ff9380fd85ae0368b90815f05828560cshekeriev/mybgapp-web:latest]
docker_image.mybgapp_db: Still creating... [00m30s elapsed]
docker_image.mybgapp_db: Creation complete after 32s [id=sha256:40e3ef703f0b38ff8e0d6dbc10202035b1c27f60fe940ea52d5c7489195e942bshekeriev/mybgapp-db:latest]
docker_container.db: Creating...
docker_container.db: Creation complete after 2s [id=b9488a091d874962e5648927263fe8f46c3e1891c87be2e784638a4ccb3c50ce]
docker_container.web: Creating...
docker_container.web: Creation complete after 1s [id=fc08db762ccc5d46e36fdb5de7b399277dddcea2030e83a5642086cdad00deb9]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
vagrant@docker:/vagrant/terraform/task-1a$ |
```

To check if all is working go to http://localhost:8000 on the Host OS.



Факти за България

Площ	110 993.6 кв.км.
Население	6 838 937
Столица	София

Големи градове

София	1236047
Пловдив	343424
Варна	335177
Бургас	202766
Русе	144936
Стара Загора	136781
Плевен	98467
Сливен	87322
Добрич	85402
Шумен	76967

Processed by **fc08db762ccc** on 2025-07-02-19-11-46

Local Image Build

The files for the task are located inside terraform/task-1b folder. If we explore the folder, we will see similar terraform code split into separate files: main.tf, variables.tf and terraform.tfvars like in the first task.

First, we need to get the project. While in the folder task-1b clone the repo. Later it will be cleaned up by a vagrant trigger on destroy.

```
git clone https://github.com/shekeriev/bgapp.git
```

We will perform the same steps as the previous task.

```
terraform init
```

```
terraform validate
```

Let's explore the docker image setup

```
resource "docker_image" "bgapp_db" {
  name = "${var.mybgapp_db_image_repo}:${var.mybgapp_db_image_tag}"
  keep_locally = true

  build {
    context    = "${path.cwd}/bgapp"
    dockerfile = "${path.cwd}/bgapp/Dockerfile.db"
  }
}
```

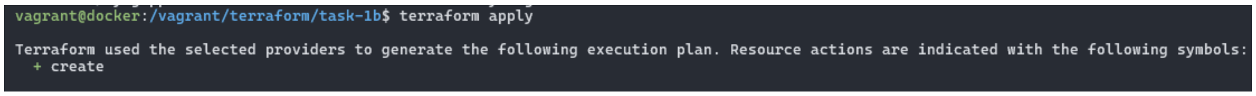
The file terraform.tfvars was also changed.

- Changed the db and web image names
- Changed the volume host path

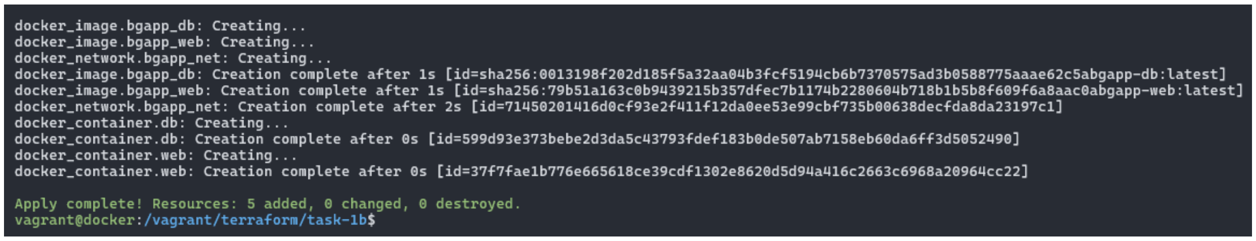
Here we can see that the image needs to be built locally first before we can use it. The same applies for the bgapp_web image.

We then provision docker

```
terraform apply
```

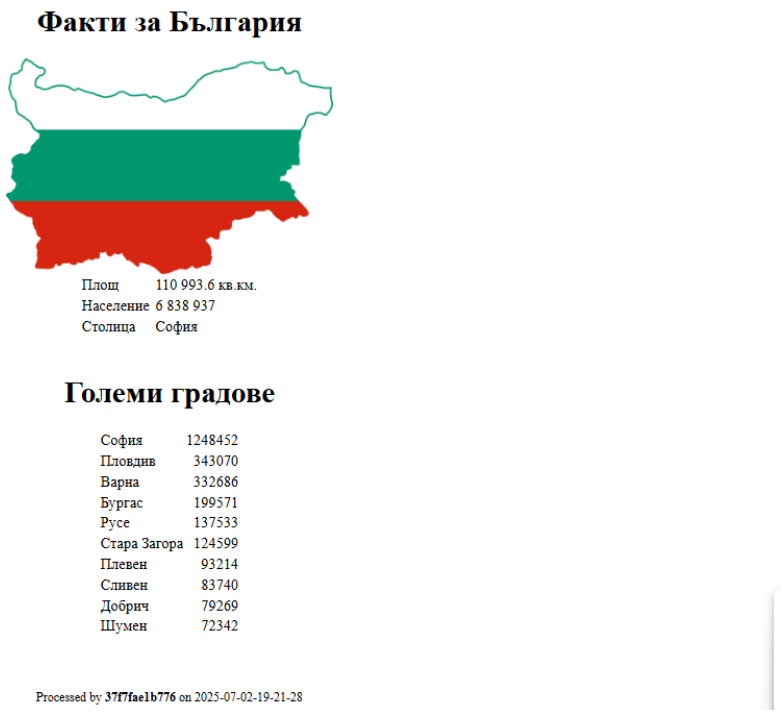


```
vagrant@docker:/vagrant/terraform/task-1b$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create
```



```
docker_image.bgapp_db: Creating...
docker_image.bgapp_web: Creating...
docker_network.bgapp_net: Creating...
docker_image.bgapp_db: Creation complete after 1s [id=sha256:0013198f202d185f5a32aa04b3fcf5194cb6b7370575ad3b0588775aaae62c5abgapp-db:latest]
docker_image.bgapp_web: Creation complete after 1s [id=sha256:79b51a163c0b9439215b357dfec7b1174b2280604b718b1b5b8f609f6a8aac0abgapp-web:latest]
docker_network.bgapp_net: Creation complete after 2s [id=71450201416d0cf93e2f411f12da0ee53e99cbf735b00638decfda8da23197c1]
docker_container.db: Creating...
docker_container.db: Creation complete after 0s [id=599d93e373bebe2d3da5c43793fdef183b0de507ab7158eb60da6ff3d5052490]
docker_container.web: Creating...
docker_container.web: Creation complete after 0s [id=37f7fae1b776e665618ce39cdf1302e8620d5d94a416c2663c6968a20964cc22]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
vagrant@docker:/vagrant/terraform/task-1b$
```

If we brows http://localhost:8000 on the Host OS, we will see the site loading.



Факти за България

Площ	110 993.6 кв.км.
Население	6 838 937
Столица	София

Големи градове

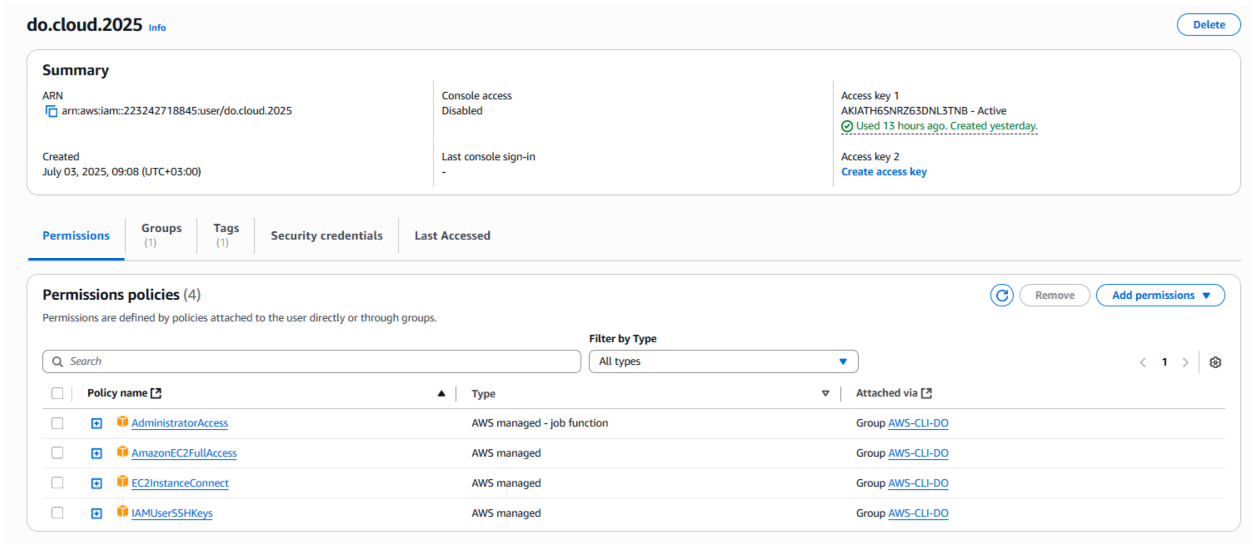
София	1248452
Пловдив	343070
Варна	332686
Бургас	199571
Русе	137533
Стара Загора	124599
Плевен	93214
Сливен	83740
Добрич	79269
Шумен	72342

Processed by **37f7fae1b776** on 2025-07-02-19-21-28

Terraform and AWS

Setup AWS User

To not use the root user, I created an additional and more limited use account for the purpose of the task through the Identity and Access Management service. The user will be used only with the CLI it has no access to the console but an access key which will be saved to the environment.



do.cloud.2025 Info

Summary

ARN
arn:aws:iam::223242718845:user/do.cloud.2025

Created
July 03, 2025, 09:08 (UTC+03:00)

Console access
Disabled

Last console sign-in
-

Access key 1
AKIATH6SNRZ63DNL3TNB - Active
Used 13 hours ago. Created yesterday.

Access key 2
Create access key

Permissions policies (4)

Permissions are defined by policies attached to the user directly or through groups.

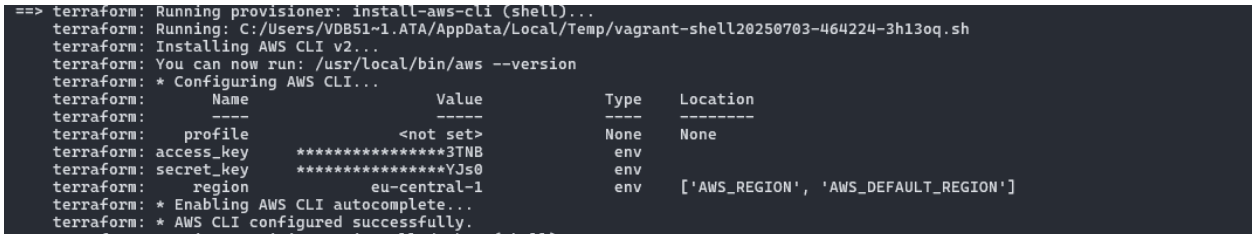
Filter by Type: All types

Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Group AWS-CLI-DO
AmazonEC2FullAccess	AWS managed	Group AWS-CLI-DO
EC2InstanceConnect	AWS managed	Group AWS-CLI-DO
IAMUserSSHKeys	AWS managed	Group AWS-CLI-DO

Setup AWS V2 CLI

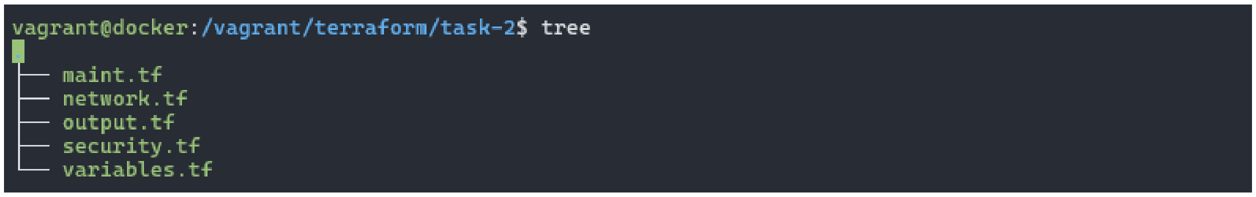
To set up the aws cli tool and configure it with the access key I have saved the access key id and access key secret to environment variables on my Host OS (Windows 11) and load them in the Vagrantfile configuration which passes the variables to the aws-cli-setup.sh script. The script installs and configures the aws cli for the vagrant user if both AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY are present.

```
terraform.vm.provision 'install-aws-cli', type: :shell, privileged: false do |shell|
  shell.env = {
    'AWS_ACCESS_KEY_ID' => "#{ENV['AWS_ACCESS_KEY_ID']}",
    'AWS_SECRET_ACCESS_KEY' => "#{ENV['AWS_SECRET_ACCESS_KEY']}",
    'AWS_DEFAULT_REGION' => "#{ENV.fetch('AWS_DEFAULT_REGION', 'eu-central-1')}",
    'AWS_OUTPUT_FORMAT' => "#{ENV.fetch('AWS_OUTPUT_FORMAT', 'json')}"
  }
  shell.path = "#{SCRIPTS_DIR}/aws-cli-setup.sh"
end
```



```
==> terraform: Running provisioner: install-aws-cli (shell)...
    terraform: Running: C:/Users/VDB51~1.ATA/AppData/Local/Temp/vagrant-shell20250703-464224-3h13oq.sh
    terraform: Installing AWS CLI v2...
    terraform: You can now run: /usr/local/bin/aws --version
    terraform: * Configuring AWS CLI...
    terraform:       Name                    Value             Type    Location
    terraform:       ----                    -----             ----    --------
    terraform:    profile                <not set>             None    None
    terraform: access_key     ****************3TNB              env
    terraform: secret_key     ****************YJs0              env
    terraform:     region             eu-central-1              env    ['AWS_REGION', 'AWS_DEFAULT_REGION']
    terraform: * Enabling AWS CLI autocomplete...
    terraform: * AWS CLI configured successfully.
```

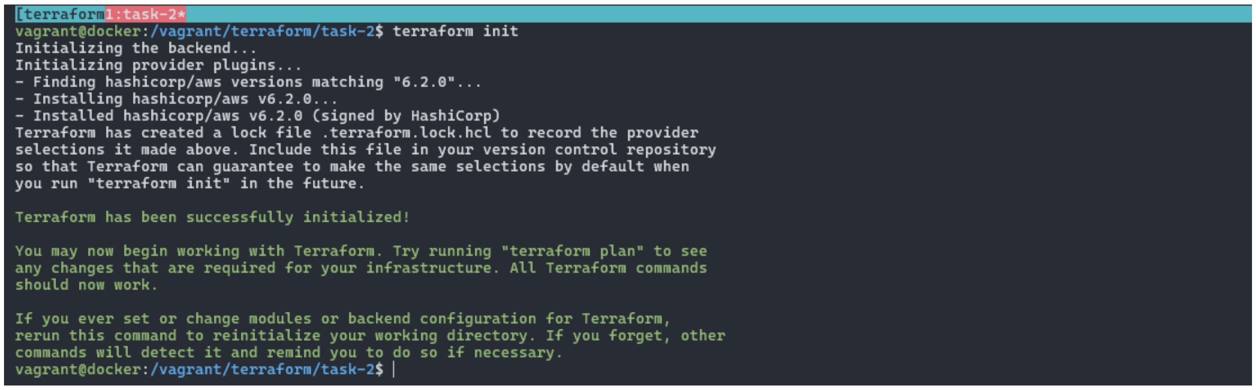
Explore Terraform Project Structure



```
vagrant@docker:/vagrant/terraform/task-2$ tree
.
├── maint.tf
├── network.tf
├── output.tf
├── security.tf
└── variables.tf
```

Download the AWS Provider

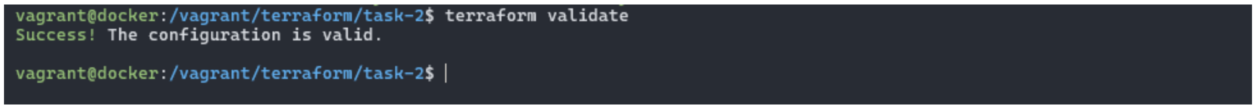
```
terraform init
```



```
[terraform] 1:task-2*
vagrant@docker:/vagrant/terraform/task-2$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "6.2.0"...
- Installing hashicorp/aws v6.2.0...
- Installed hashicorp/aws v6.2.0 (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.
vagrant@docker:/vagrant/terraform/task-2$ |
```

Validate Terraform Code

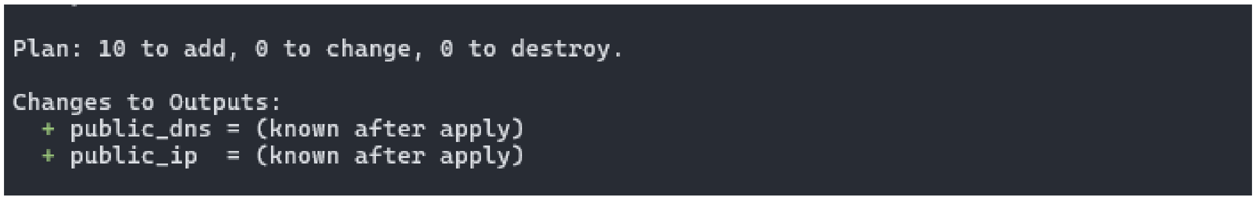
```
terraform validate
```


```
vagrant@docker:/vagrant/terraform/task-2$ terraform validate
Success! The configuration is valid.

vagrant@docker:/vagrant/terraform/task-2$ |
```

Terraform Plan

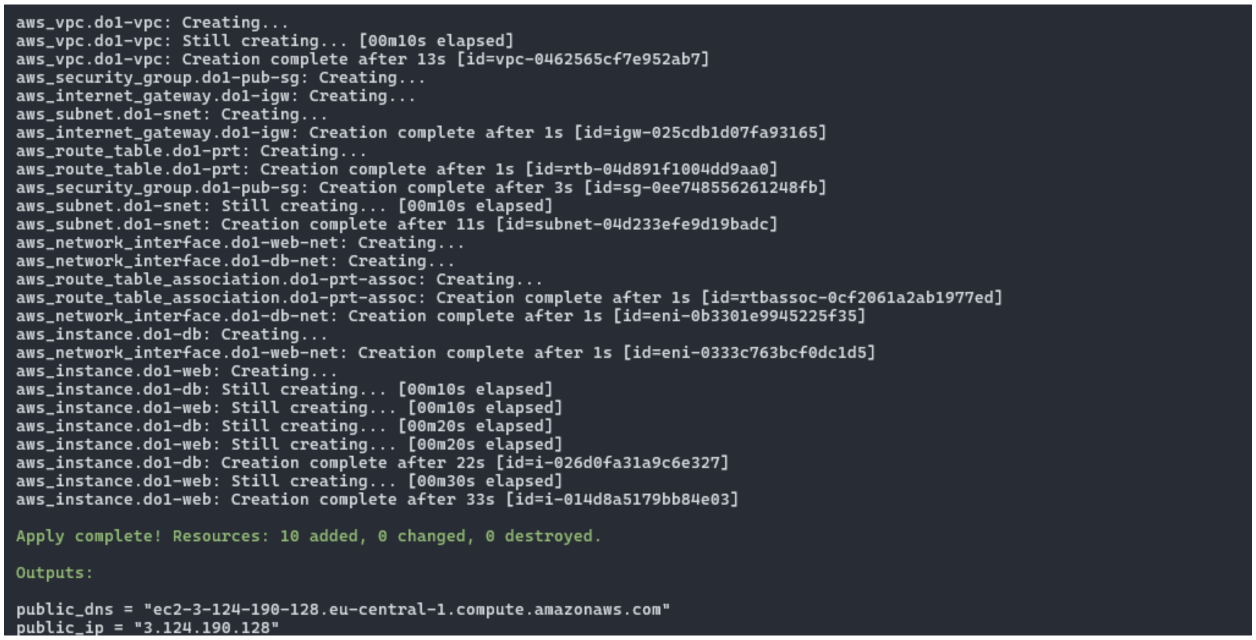
To visually verify what will be created on AWS we execute.

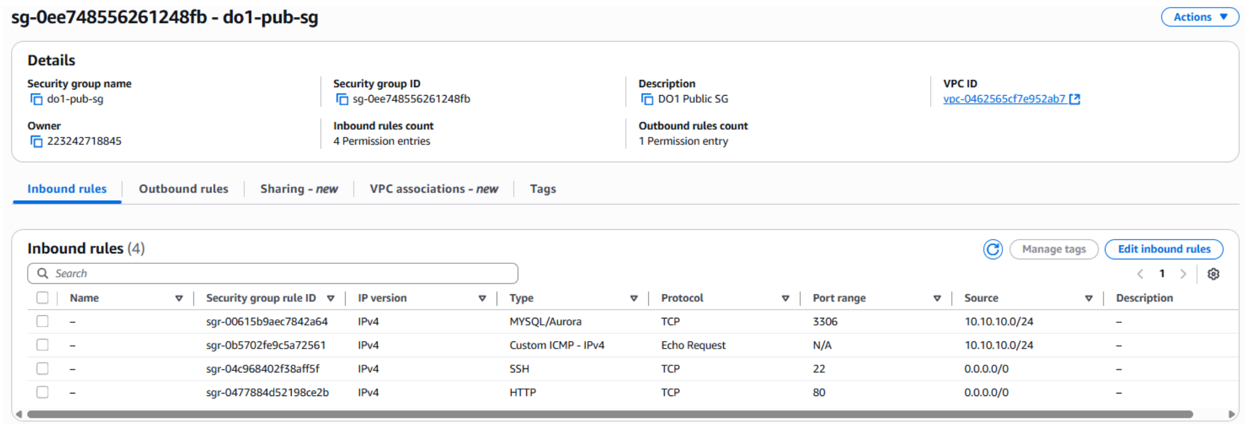
```
terraform plan
```


```
Plan: 10 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + public_dns = (known after apply)
  + public_ip  = (known after apply)
```

Provision of the Infrastructure

```
terraform apply
```


```
aws_vpc.do1-vpc: Creating...
aws_vpc.do1-vpc: Still creating... [00m10s elapsed]
aws_vpc.do1-vpc: Creation complete after 13s [id=vpc-0462565cf7e952ab7]
aws_security_group.do1-pub-sg: Creating...
aws_internet_gateway.do1-igw: Creating...
aws_subnet.do1-snet: Creating...
aws_internet_gateway.do1-igw: Creation complete after 1s [id=igw-025cdb1d07fa93165]
aws_route_table.do1-prt: Creating...
aws_route_table.do1-prt: Creation complete after 1s [id=rtb-04d891f1004dd9aa0]
aws_security_group.do1-pub-sg: Creation complete after 3s [id=sg-0ee748556261248fb]
aws_subnet.do1-snet: Still creating... [00m10s elapsed]
aws_subnet.do1-snet: Creation complete after 11s [id=subnet-04d233efe9d19badc]
aws_network_interface.do1-web-net: Creating...
aws_network_interface.do1-db-net: Creating...
aws_route_table_association.do1-prt-assoc: Creating...
aws_route_table_association.do1-prt-assoc: Creation complete after 1s [id=rtbassoc-0cf2061a2ab1977ed]
aws_network_interface.do1-db-net: Creation complete after 1s [id=eni-0b3301e9945225f35]
aws_instance.do1-db: Creating...
aws_network_interface.do1-web-net: Creation complete after 1s [id=eni-0333c763bcf0dc1d5]
aws_instance.do1-web: Creating...
aws_instance.do1-db: Still creating... [00m10s elapsed]
aws_instance.do1-web: Still creating... [00m10s elapsed]
aws_instance.do1-db: Still creating... [00m20s elapsed]
aws_instance.do1-web: Still creating... [00m20s elapsed]
aws_instance.do1-db: Creation complete after 22s [id=i-026d0fa31a9c6e327]
aws_instance.do1-web: Still creating... [00m30s elapsed]
aws_instance.do1-web: Creation complete after 33s [id=i-014d8a5179bb84e03]

Apply complete! Resources: 10 added, 0 changed, 0 destroyed.

Outputs:

public_dns = "ec2-3-124-190-128.eu-central-1.compute.amazonaws.com"
public_ip = "3.124.190.128"
```

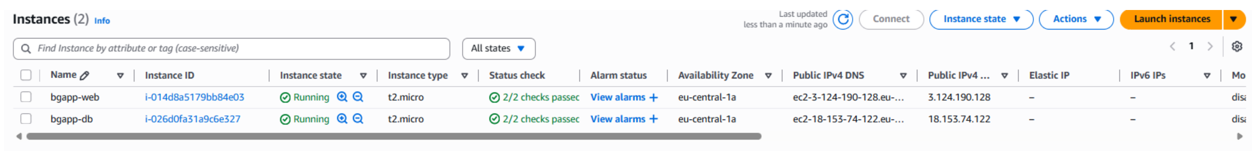

sg-0ee748556261248fb - do1-pub-sg

Details

Security group name do1-pub-sg	**Security group ID** sg-0ee748556261248fb	**Description** DO1 Public SG	**VPC ID** vpc-0462565cf7e952ab7
Owner 223242718845	**Inbound rules count** 4 Permission entries	**Outbound rules count** 1 Permission entry	

Inbound rules (4)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
–	sgr-00615b9aec7842a64	IPv4	MYSQL/Aurora	TCP	3306	10.10.10.0/24	–
–	sgr-0b5702fe9c5a72561	IPv4	Custom ICMP - IPv4	Echo Request	N/A	10.10.10.0/24	–
–	sgr-04c968402f38aff5f	IPv4	SSH	TCP	22	0.0.0.0/0	–
–	sgr-0477884d52198ce2b	IPv4	HTTP	TCP	80	0.0.0.0/0	–


Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states

Instances (2) Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Mo
☐	bgapp-web	i-014d8a5179bb84e03	Running	t2.micro	2/2 checks passec	View alarms +	eu-central-1a	ec2-3-124-190-128.eu-...	3.124.190.128	–	–	dis
☐	bgapp-db	i-026d0fa31a9c6e327	Running	t2.micro	2/2 checks passec	View alarms +	eu-central-1a	ec2-18-153-74-122.eu-...	18.153.74.122	–	–	dis

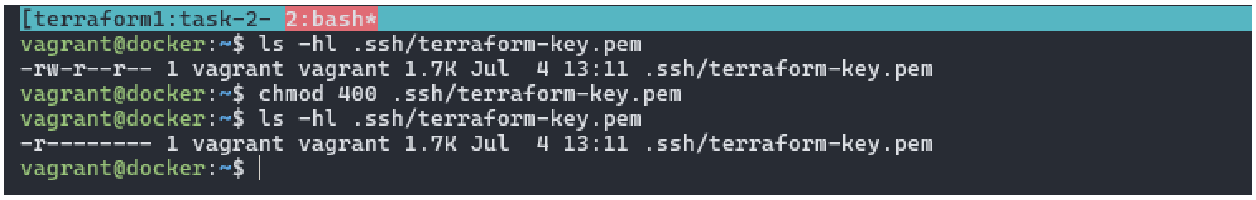
From the screenshots it is visible we can access the machines with SSH and make ping request in the private network.

SSH Setup

To be able to login into the machines we need a keypair which I generated beforehand and stored on my Host machine. I will transfer it to the VM with the following command:

```
scp -P 2222 C:\Users\v.atanasov\.ssh\terraform-key.pem vagrant@127.0.0.1:/home/vagrant/.ssh/terraform-key.pem
```

Then restrict the key usage only to the owner user (vagrant)



```
[terraform1:task-2- 2:bash*
vagrant@docker:~$ ls -hl .ssh/terraform-key.pem
-rw-r--r-- 1 vagrant vagrant 1.7K Jul  4 13:11 .ssh/terraform-key.pem
vagrant@docker:~$ chmod 400 .ssh/terraform-key.pem
vagrant@docker:~$ ls -hl .ssh/terraform-key.pem
-r-------- 1 vagrant vagrant 1.7K Jul  4 13:11 .ssh/terraform-key.pem
vagrant@docker:~$ |
```

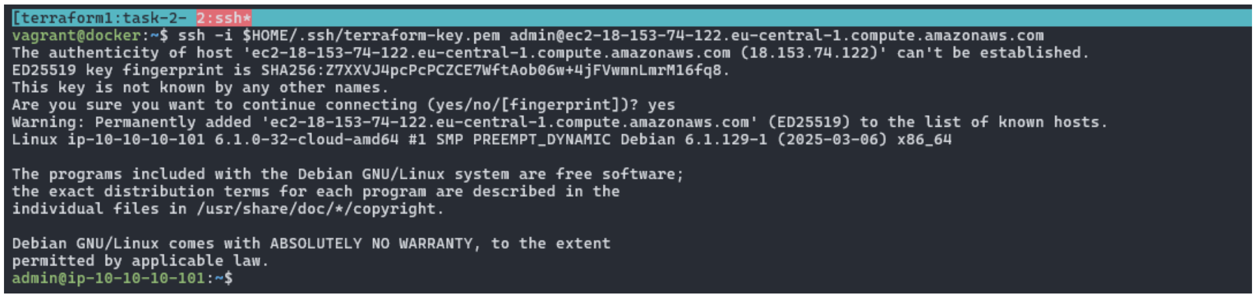
DB Server Setup

I will ssh into the server machine and manually set up the database

SSH into DB Server

To be able to log in we need to specify the location of the key

```
ssh -i $HOME/.ssh/terraform-key.pem admin@ec2-18-153-74-122.eu-central-1.compute.amazonaws.com
```



```
[terraform1:task-2- 2:ssh*
vagrant@docker:~$ ssh -i $HOME/.ssh/terraform-key.pem admin@ec2-18-153-74-122.eu-central-1.compute.amazonaws.com
The authenticity of host 'ec2-18-153-74-122.eu-central-1.compute.amazonaws.com (18.153.74.122)' can't be established.
ED25519 key fingerprint is SHA256:Z7XXVJ4pcPcPCZCE7WftAob06w+4jFVwmnLmrM16fq8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-153-74-122.eu-central-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
Linux ip-10-10-10-101 6.1.0-32-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.129-1 (2025-03-06) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
admin@ip-10-10-10-101:~$
```

Install MariaDB Server

```
sudo apt update && sudo apt install -y mariadb-server git
```

Make the Database Accessible

By default, the server will start listening for connections on 127.0.0.1:3306. To be able to connect to the database from the web server we need to modify the configuration to accept connections from the outside. First remove the bind-address line and replace it with the new one.

```
sudo sed -i '/^\s*bind-address\s*=/d' /etc/mysql/mariadb.conf.d/50-server.cnf
sudo sed -i '/^\[mysqld\]/a bind-address = 0.0.0.0' /etc/mysql/mariadb.conf.d/50-server.cnf
```

Enable and start the MariaDB service

```
sudo systemctl enable --now mariadb
```

Download the project

```
git clone https://github.com/shekeriev/bgapp ~/bgapp
```

Install the database required for the web application

```
sudo mysql -u root < ~/bgapp/db/db_setup.sql
```

At this point we should have our db running and ready to accept connections from web server.

Web Server Setup

I will ssh into the server machine and manually set up the web server

SSH into Web Server

```
ssh -i $HOME/.ssh/terraform-key.pem admin@ec2-3-124-190-128.eu-central-1.compute.amazonaws.com
```

Install the required packages

```
sudo apt update && sudo apt install -y apache2 php php-mysql git
```

Enable and start the Apache2 service

```
sudo systemctl enable --now apache2
```

Download the project

```
git clone https://github.com/shekeriev/bgapp ~/bgapp
```

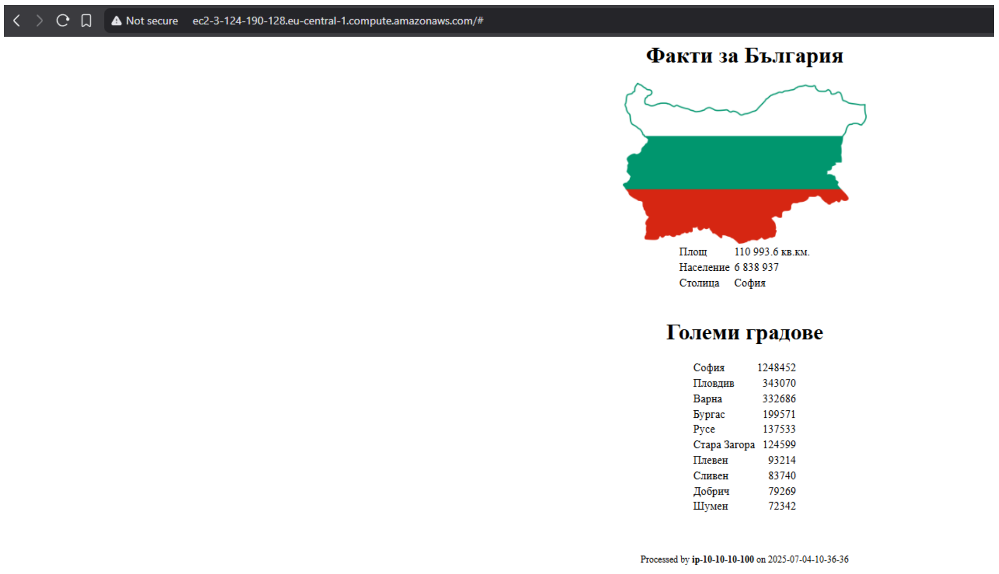
Copy the files related to the web application

```
sudo cp ~/bgapp/web/* /var/www/html/
```

Substitute MariaDB related connection parameters

We don't have name resolution so we need to change the host from DB to the IP address of the database server.

```
sudo sed -i 's/db/10.10.10.101/g' /var/www/html/config.php
```



Факти за България

Площ	110 993.6 кв.км.
Население	6 838 937
Столица	София

Големи градове

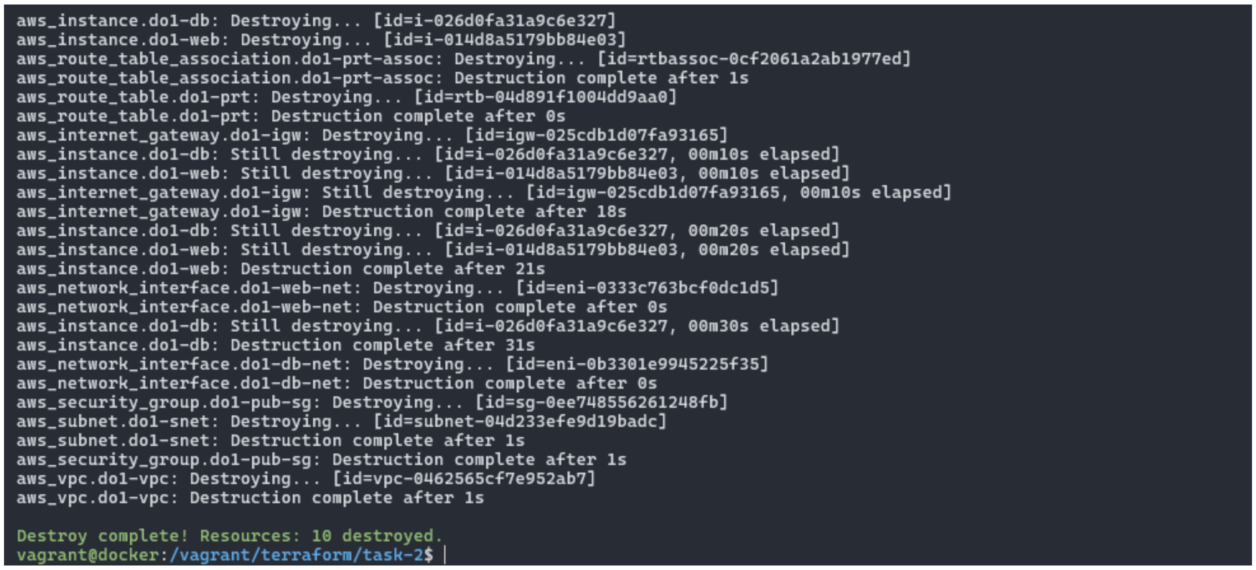
София	1248452
Пловдив	343070
Варна	332686
Бургас	199571
Русе	137533
Стара Загора	124599
Плевен	93214
Сливен	83740
Добрич	79269
Шумен	72342

Processed by **ip-10-10-10-100** on 2025-07-04-10-36-36

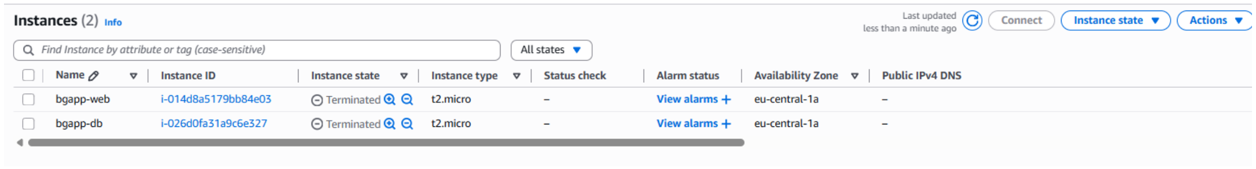
Destroy the Infrastructure

To destroy all configurations related to the infrastructure we execute

```
terraform destroy
```



```
aws_instance.do1-db: Destroying... [id=i-026d0fa31a9c6e327]
aws_instance.do1-web: Destroying... [id=i-014d8a5179bb84e03]
aws_route_table_association.do1-prt-assoc: Destroying... [id=rtbassoc-0cf2061a2ab1977ed]
aws_route_table_association.do1-prt-assoc: Destruction complete after 1s
aws_route_table.do1-prt: Destroying... [id=rtb-04d891f1004dd9aa0]
aws_route_table.do1-prt: Destruction complete after 0s
aws_internet_gateway.do1-igw: Destroying... [id=igw-025cdb1d07fa93165]
aws_instance.do1-db: Still destroying... [id=i-026d0fa31a9c6e327, 00m10s elapsed]
aws_instance.do1-web: Still destroying... [id=i-014d8a5179bb84e03, 00m10s elapsed]
aws_internet_gateway.do1-igw: Still destroying... [id=igw-025cdb1d07fa93165, 00m10s elapsed]
aws_internet_gateway.do1-igw: Destruction complete after 18s
aws_instance.do1-db: Still destroying... [id=i-026d0fa31a9c6e327, 00m20s elapsed]
aws_instance.do1-web: Still destroying... [id=i-014d8a5179bb84e03, 00m20s elapsed]
aws_instance.do1-web: Destruction complete after 21s
aws_network_interface.do1-web-net: Destroying... [id=eni-0333c763bcf0dc1d5]
aws_network_interface.do1-web-net: Destruction complete after 0s
aws_instance.do1-db: Still destroying... [id=i-026d0fa31a9c6e327, 00m30s elapsed]
aws_instance.do1-db: Destruction complete after 31s
aws_network_interface.do1-db-net: Destroying... [id=eni-0b3301e9945225f35]
aws_network_interface.do1-db-net: Destruction complete after 0s
aws_security_group.do1-pub-sg: Destroying... [id=sg-0ee748556261248fb]
aws_subnet.do1-snet: Destroying... [id=subnet-04d233efe9d19badc]
aws_subnet.do1-snet: Destruction complete after 1s
aws_security_group.do1-pub-sg: Destruction complete after 1s
aws_vpc.do1-vpc: Destroying... [id=vpc-0462565cf7e952ab7]
aws_vpc.do1-vpc: Destruction complete after 1s

Destroy complete! Resources: 10 destroyed.
vagrant@docker:/vagrant/terraform/task-2$
```



Instances (2) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Find Instance by attribute or tag (case-sensitive)
All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
bgapp-web	i-014d8a5179bb84e03	Terminated	t2.micro	–	View alarms +	eu-central-1a	–
bgapp-db	i-026d0fa31a9c6e327	Terminated	t2.micro	–	View alarms +	eu-central-1a	–