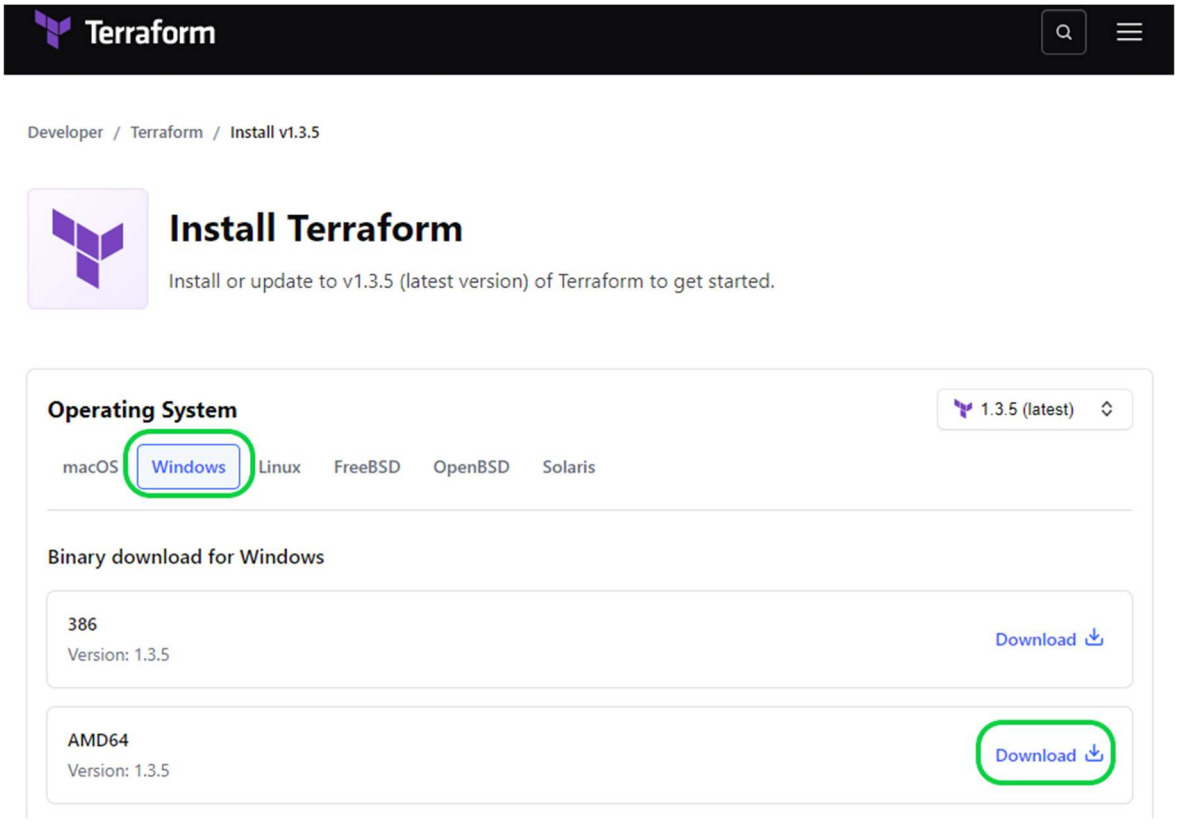
Terraform

1. Download and install Terraform on Windows x64 and Amazon Linux.
2. Creating “terraform” user IAM in AWS console with Admin access.
3. Creating infrasrtucture.
4. Project initialization and terraform plan.
5. Terraform apply and creating infrastructure.
6. Simple Terraform destroy.
7. Bootstrapping container instances with Amazon EC2 user data.
8. Terraform implicit dependencies example.
9. Terraform explicit dependencies example.
10. Terraform variables.
11. Terraform Output.
12. Terraform Modules.

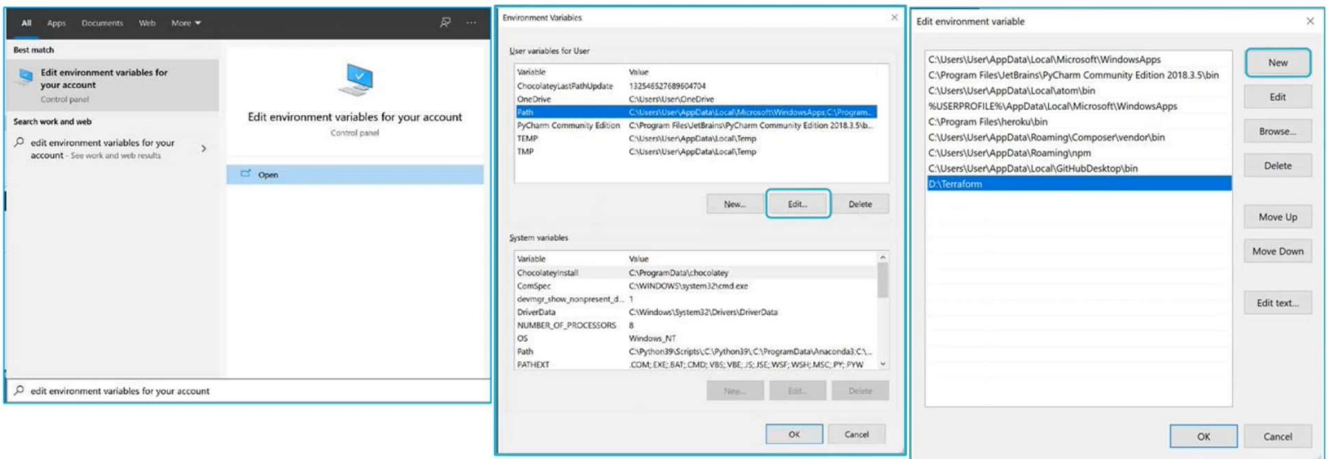
1. Download and install Terraform on Windows x64 and Amazon Linux.

- **Windows**

a. Download newest version Terraform 1.3.5 from website and unzip archive



b. Added path where are you save terraform.exe to system PATH

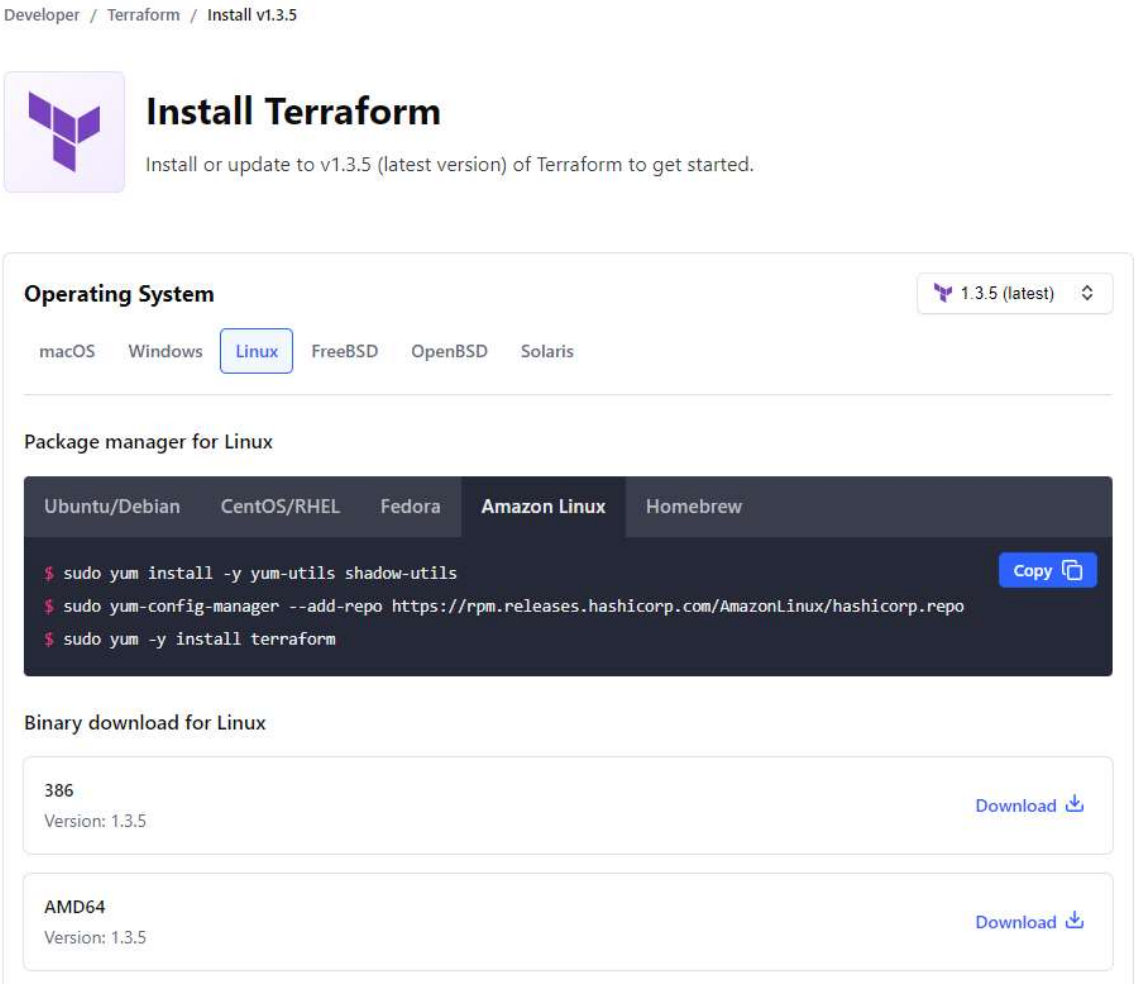


c. Verify the instalation

```
PS C:\Users\Yevhen Yakymov> terraform -version
Terraform v1.3.5
on windows_amd64
PS C:\Users\Yevhen Yakymov> terraform -help
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure
```

- **Amazon Linux**

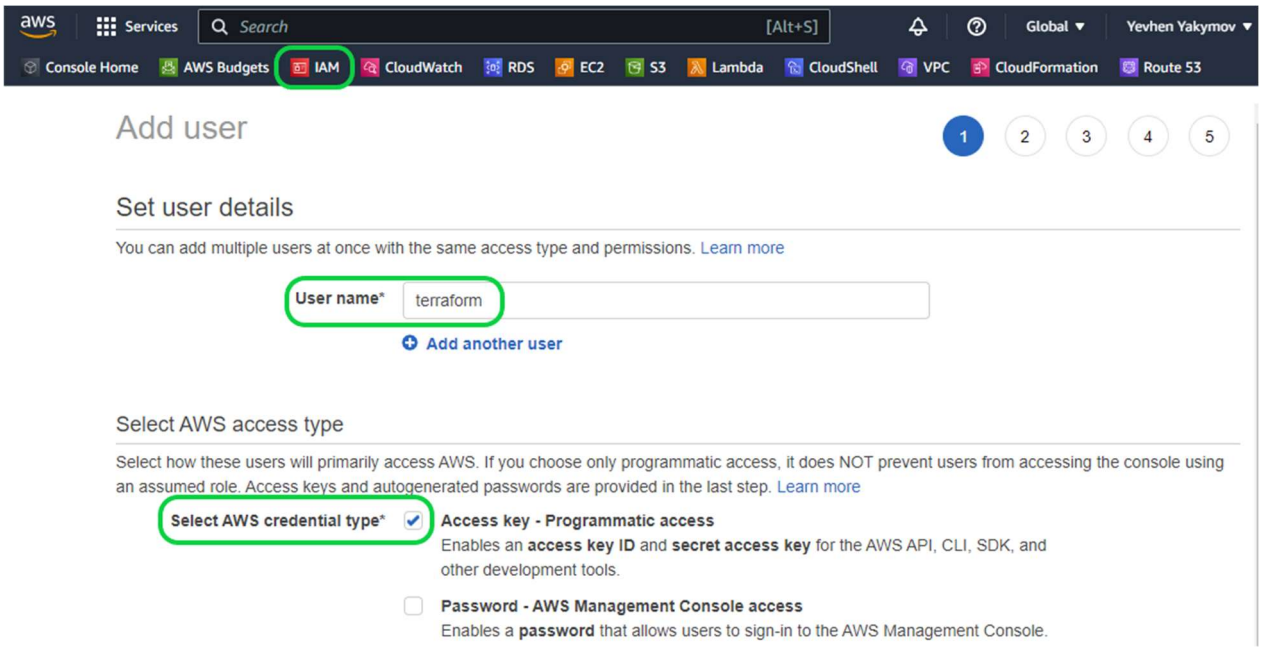
a. Download and install (package manager or binary download it is up to you)

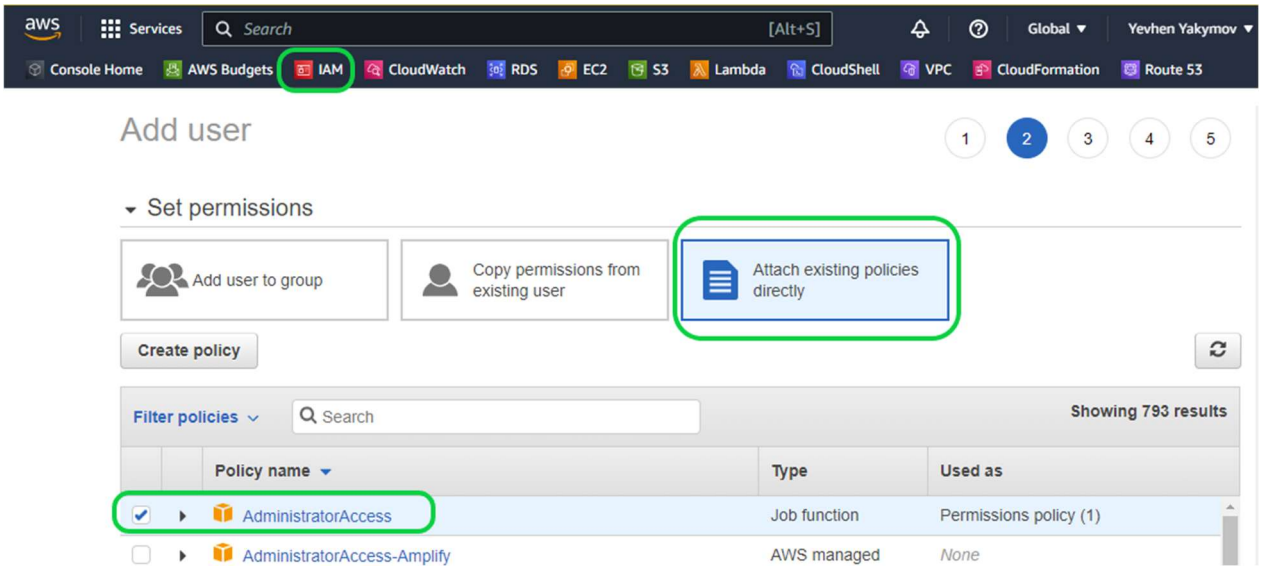


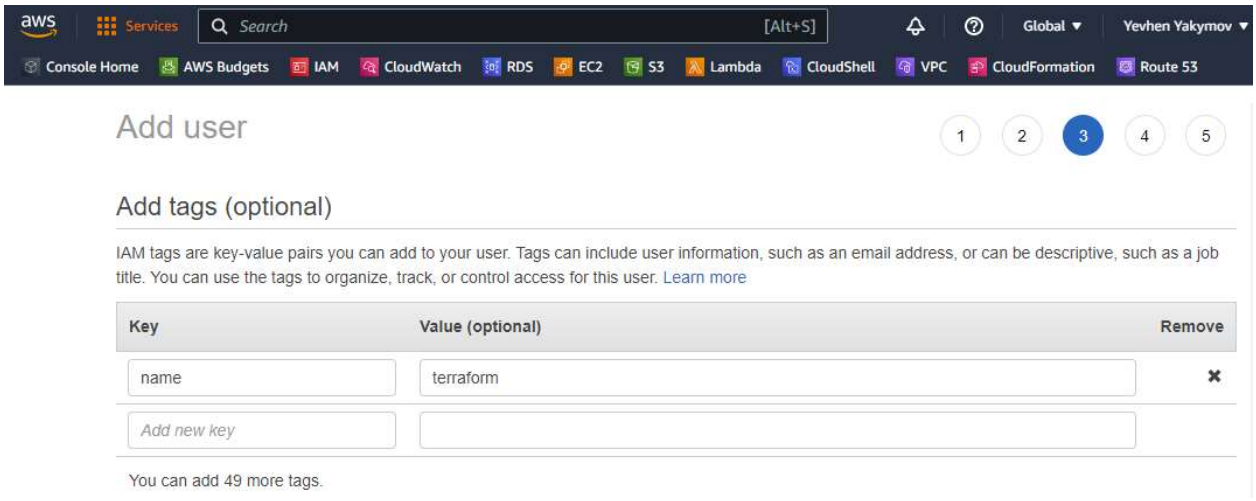
```
[ec2-user@ip-172-31-14-46 ~]$ sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
adding repo from: https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
grabbing file https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo to /etc/yum.repos.d/hashicorp.repo
repo saved to /etc/yum.repos.d/hashicorp.repo
[ec2-user@ip-172-31-14-46 ~]$
[ec2-user@ip-172-31-14-46 ~]$ sudo yum -y install terraform
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
hashicorp                                                | 1.4 kB  00:00:00
hashicorp/x86_64/primary                                 | 118 kB  00:00:00
hashicorp                                                             842/842
Resolving Dependencies
--> Running transaction check
---> Package terraform.x86_64 0:1.3.5-1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package            Arch            Version          Repository          Size
================================================================================
Installing:
 terraform          x86_64          1.3.5-1          hashicorp           13 M

Transaction Summary
================================================================================
Install  1 Package

Total download size: 13 M
Installed size: 58 M
Downloading packages:
warning: /var/cache/yum/x86_64/2/hashicorp/packages/terraform-1.3.5-1.x86_64.rpm: Header V4 RSA/SHA512 Signature, key ID a3219f7b: NOKEY
Public key for terraform-1.3.5-1.x86_64.rpm is not installed
terraform-1.3.5-1.x86_64.rpm                             |  13 MB  00:00:00
Retrieving key from https://rpm.releases.hashicorp.com/gpg
Importing GPG key 0xA3219F7B:
 Userid     : "HashiCorp Security (HashiCorp Package Signing) <security+packaging@hashicorp.com>"
 Fingerprint: e8a0 32e0 94d8 eb4e a189 d270 da41 8c88 a321 9f7b
 From       : https://rpm.releases.hashicorp.com/gpg
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : terraform-1.3.5-1.x86_64                                     1/1
  Verifying  : terraform-1.3.5-1.x86_64                                     1/1

Installed:
  terraform.x86_64 0:1.3.5-1

Complete!
```

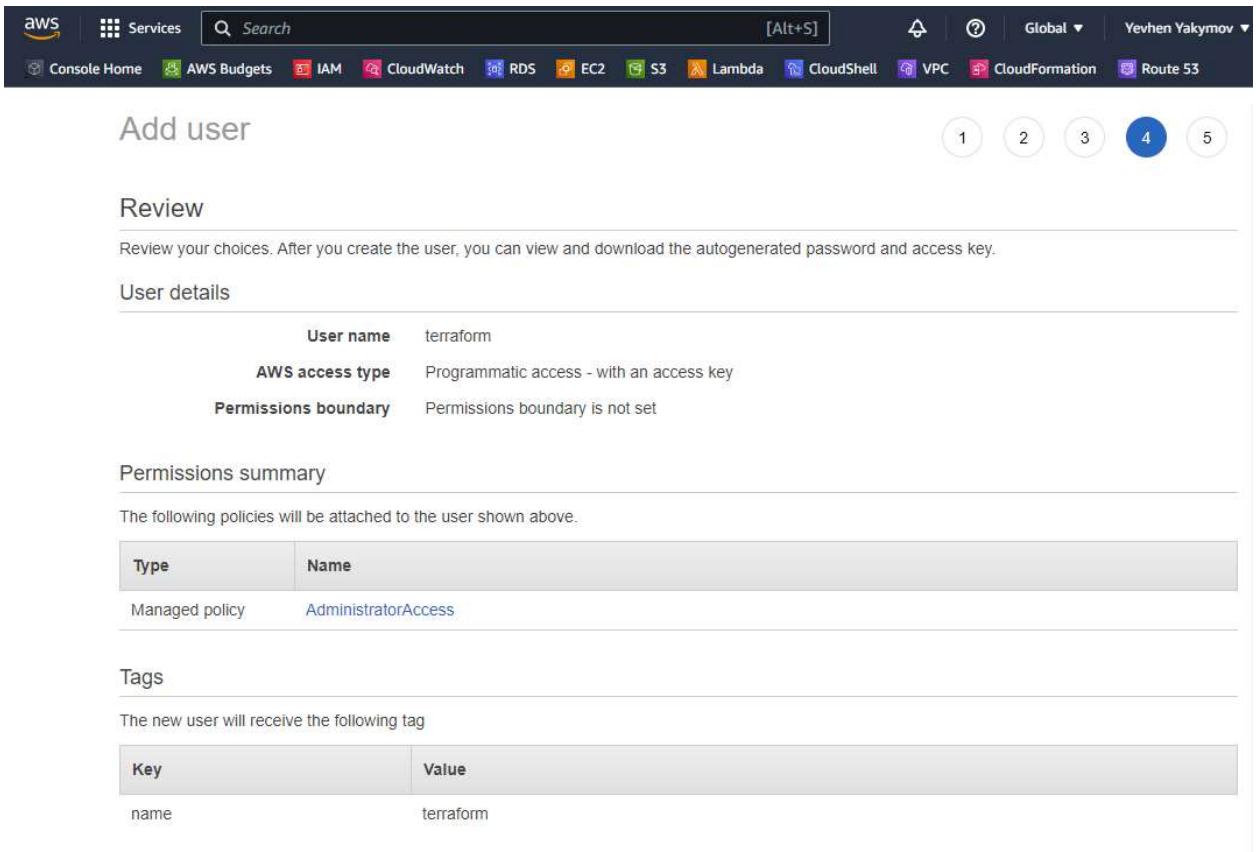
```
[ec2-user@ip-172-31-14-46 ~]$ terraform -version
Terraform v1.3.5
on linux_amd64
[ec2-user@ip-172-31-14-46 ~]$ terraform -help
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure
```

2. Creating “terraform” user IAM in AWS console with Admin access

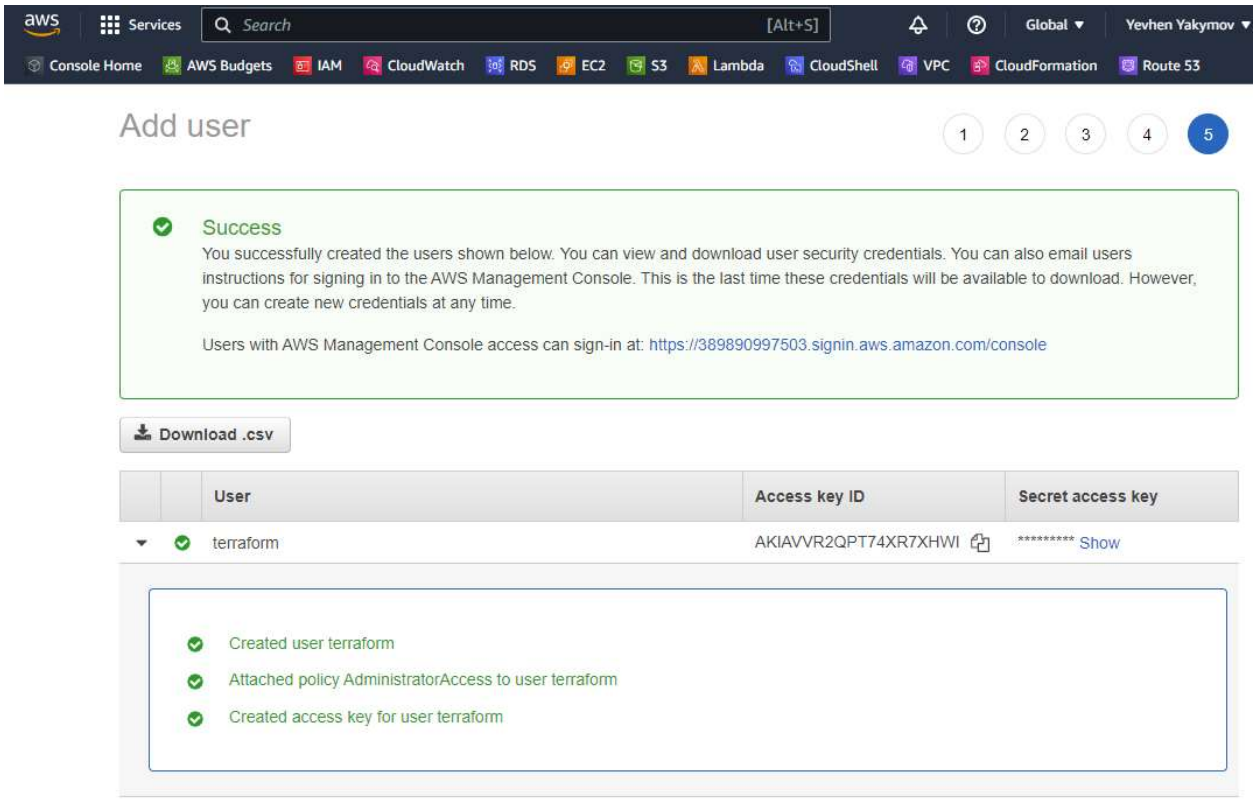








Save **Access key ID** and **Secret access key**. This credentials should be used in **"provider"** section of *.tf file(**access_key** and **secret_key** variable)



3. Creating infrasrtucture.

Create **myinfrasrtucture.tf** file with a description of the infrastructure to be deployed.

```
[ec2-user@ip-172-31-14-46 MyFirstInfrastructure]$ touch myinfrasrtucture.tf
[ec2-user@ip-172-31-14-46 MyFirstInfrastructure]$
[ec2-user@ip-172-31-14-46 MyFirstInfrastructure]$ vim myinfrasrtucture.tf
```

```
provider "aws" {
    access_key = "AKIA[redacted]"
    secret_key = "vHMhh[redacted]"
    region = "eu-central-1"
}

resource "aws_instance" "MyWebServer-1" {
        count = 1
        ami = "ami-076309742d466ad69"
        instance_type = "t2.micro"
        vpc_security_group_ids = [aws_security_group.MyWebServer-1.id]
        tags = {
        Name = "MyWebServer-1"
        Project = "MyFirstInfrastructure"
        Owner = "yakymov1yevhen@gmail.com"
        }
}

resource "aws_security_group" "MyWebServer-1" {
  name        = "Web Server Security Group"
  description = "Allow TCP inbound traffic"

  ingress {
    description      = "TCP from VPC (HTTP)"
    from_port        = 80
    to_port          = 80
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"]
  }

  ingress {
    description      = "TCP from VPC HTTPS"
    from_port        = 443
    to_port          = 443
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"]
  }

  ingress {
    description      = "SSH from VPC"
    from_port        = 22
    to_port          = 22
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"]
  }

  egress {
    from_port        = 0
    to_port          = 0
    protocol         = "-1"
    cidr_blocks      = ["0.0.0.0/0"]
  }

  tags = {
    Name = "allow_tcp"
  }
}
```

You have different ways to grant credentials, besides directly specifying access_key and secret_key it in the file *.tf.

AWS Credentials:

a) export before apply:

bash

```
export AWS_ACCES_KEY_ID="XXXXXXXXXXX"
export AWS_SECRET_ACCES_KEY="XXXXXXXXXXX"
export AWS_DEFAULT_REGION="your-region"
```

b) specify in the provider's section of ***.tf** file:

Go

```
provider "aws" {
    access_key="XXXXXXXXXXX"
    secret_key="XXXXXXXXXXX"
    region="your-region"
}
```

c) in the credentials file:

~/path/to/file/credentials

Text

```
[default/IAM user]
aws_access_key_id=XXXXXXXXXXX
aws_secret_access_key=XXXXXXXXXXX
```

~/path/to/file/config

Text

```
[default/IAM user]
region=us-west-2
```

Go

```
provider "aws" {
    shared_credentials_files = ["~/path/to/file/credentials"]
    shared_config_files      = ["~/path/to/file/config"]
}
```

4. Project initialization and terraform plan

terraform init – prepare your working directory for other commands(download providers)

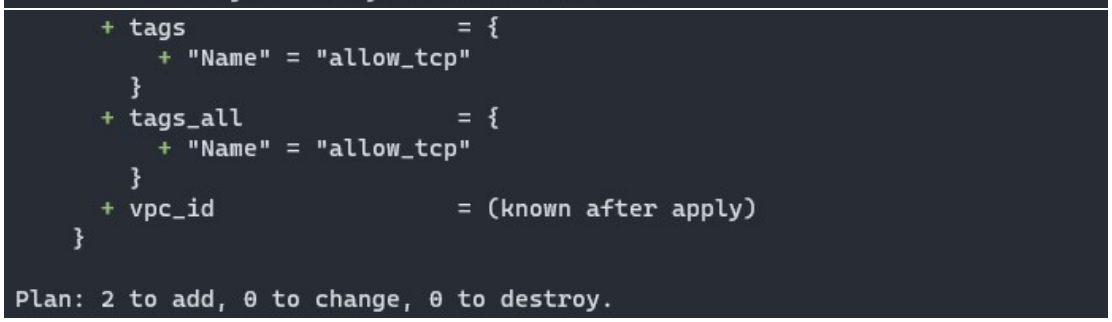
```
[ec2-user@ip-172-31-14-46 MyFirstInfrastructure]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.45.0...
- Installed hashicorp/aws v4.45.0 (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.
[ec2-user@ip-172-31-14-46 MyFirstInfrastructure]$ ll
total 4
-rw-rw-r-- 1 ec2-user ec2-user 1341 Dec  4 11:55 myinfrasrtucture.tf
[ec2-user@ip-172-31-14-46 MyFirstInfrastructure]$ ls -la
total 8
drwxrwxr-x 3 ec2-user ec2-user   78 Dec  4 12:02 .
drwxrwxr-x 3 ec2-user ec2-user   35 Dec  4 11:51 ..
-rw-rw-r-- 1 ec2-user ec2-user 1341 Dec  4 11:55 myinfrasrtucture.tf
drwxr-xr-x 3 ec2-user ec2-user   23 Dec  4 12:02 .terraform
-rw-r--r-- 1 ec2-user ec2-user 1377 Dec  4 12:02 .terraform.lock.hcl
```

terraform validate – check whether the configuration is valid

```
[ec2-user@ip-172-31-14-46 MyFirstInfrastructure]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-14-46 MyFirstInfrastructure]$
```

terraform plan – show changes required by the current configuration

```
terraform plan

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

Terraform will perform the following actions:

  # aws_instance.MyWebServer-1[0] will be created
  + resource "aws_instance" "MyWebServer-1" {
      + ami                                  = "ami-076309742d466ad69"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + subnet_id                            = (known after apply)
      + tags                                 = {
          + "Name"    = "MyWebServer-1"
          + "Owner"   = "yakymov1yevhen@gmail.com"
          + "Project" = "MyFirstInfrastructure"
        }
      + tags_all                             = {
          + "Name"    = "MyWebServer-1"
          + "Owner"   = "yakymov1yevhen@gmail.com"
          + "Project" = "MyFirstInfrastructure"
```



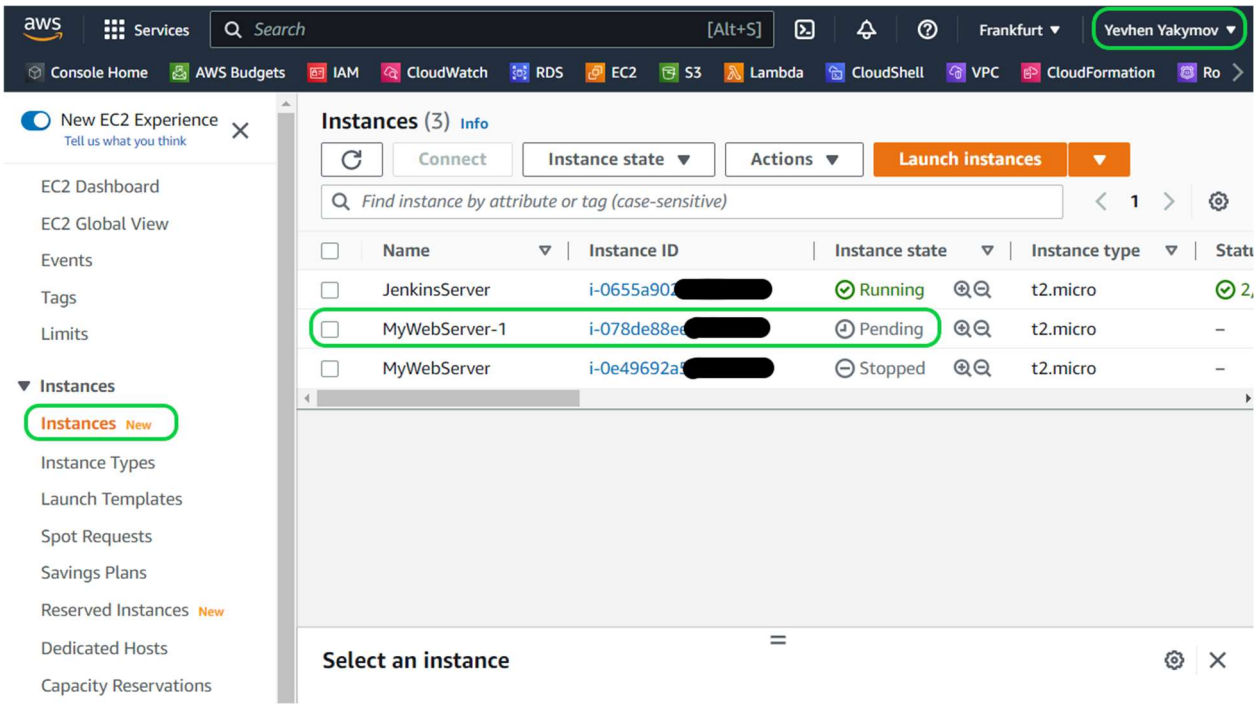
```
      + tags                   = {
          + "Name" = "allow_tcp"
        }
      + tags_all               = {
          + "Name" = "allow_tcp"
        }
      + vpc_id                 = (known after apply)
    }

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

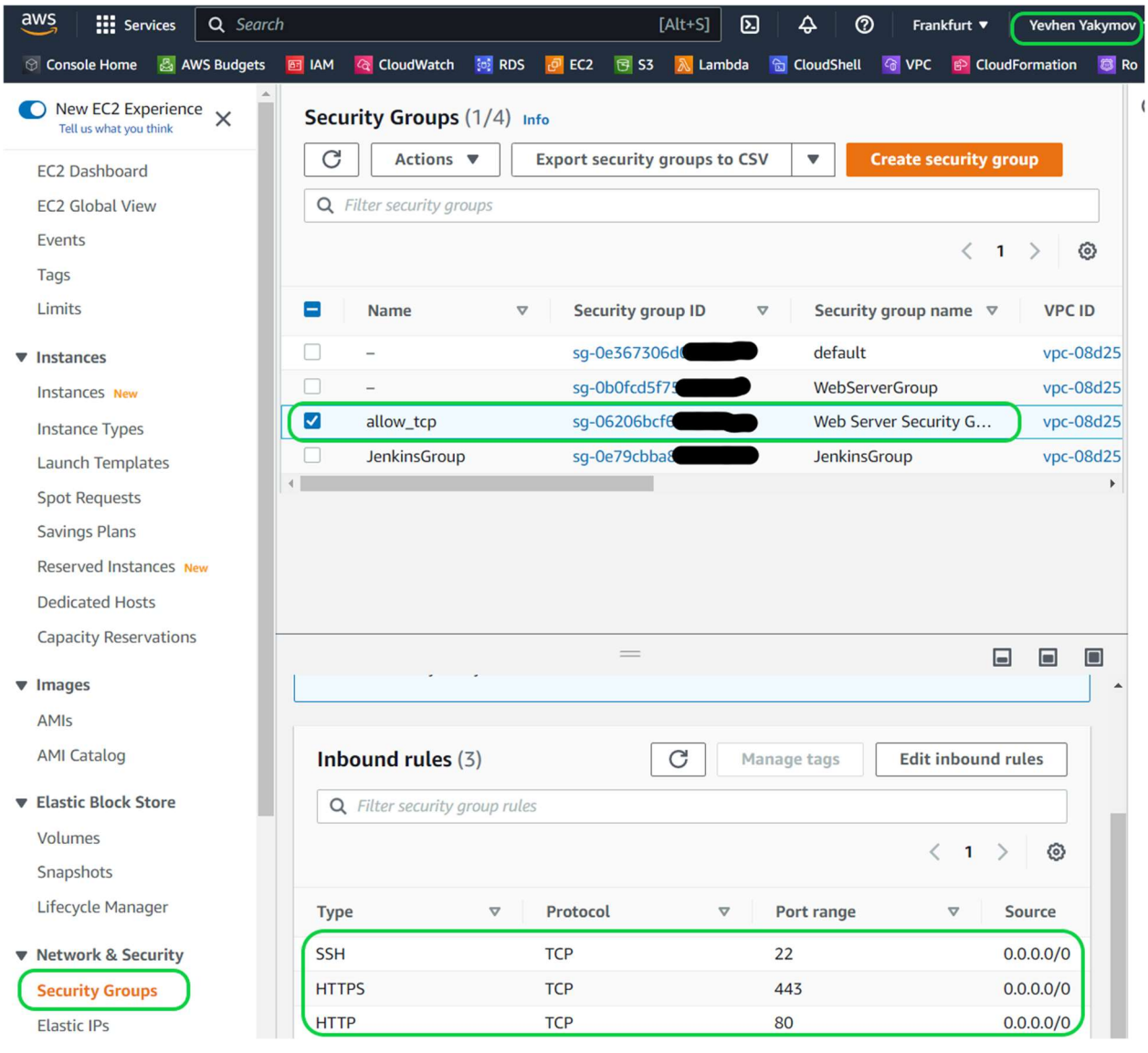
5. Terraform apply and creating infrastructure

terraform apply – create or update infrastructure

```
terraform apply

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

Terraform will perform the following actions:

  # aws_instance.MyWebServer-1[0] will be created
  + resource "aws_instance" "MyWebServer-1" {
      + ami                                  = "ami-076309742d466ad69"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
```

```
Plan: 2 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_security_group.MyWebServer-1: Creating...
aws_security_group.MyWebServer-1: Creation complete after 2s [id=sg-06206bcf6d[redacted]]
aws_instance.MyWebServer-1[0]: Creating...
aws_instance.MyWebServer-1[0]: Still creating... [10s elapsed]
aws_instance.MyWebServer-1[0]: Still creating... [20s elapsed]
aws_instance.MyWebServer-1[0]: Still creating... [30s elapsed]
aws_instance.MyWebServer-1[0]: Creation complete after 31s [id=i-078de88ee[redacted]]

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


aws
Services
Search
[Alt+S]
Frankfurt
Yevhen Yakymov
Console Home
AWS Budgets
IAM
CloudWatch
RDS
EC2
S3
Lambda
CloudShell
VPC
CloudFormation
New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Capacity Reservations
Instances (3) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
Name
Instance ID
Instance state
Instance type
JenkinsServer
Running
t2.micro
MyWebServer-1
Pending
t2.micro
MyWebServer
Stopped
t2.micro
Select an instance


aws
Services
Search
[Alt+S]
Frankfurt
Yevhen Yakymov
Console Home
AWS Budgets
IAM
CloudWatch
RDS
EC2
S3
Lambda
CloudShell
VPC
CloudFormation
New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Security Groups (1/4) Info
Actions
Export security groups to CSV
Create security group
Filter security groups
Name
Security group ID
Security group name
VPC ID
default
WebServerGroup
allow_tcp
Web Server Security G...
JenkinsGroup
Inbound rules (3)
Manage tags
Edit inbound rules
Filter security group rules
Type
Protocol
Port range
Source
SSH
TCP
22
0.0.0.0/0
HTTPS
TCP
443
0.0.0.0/0
HTTP
TCP
80
0.0.0.0/0

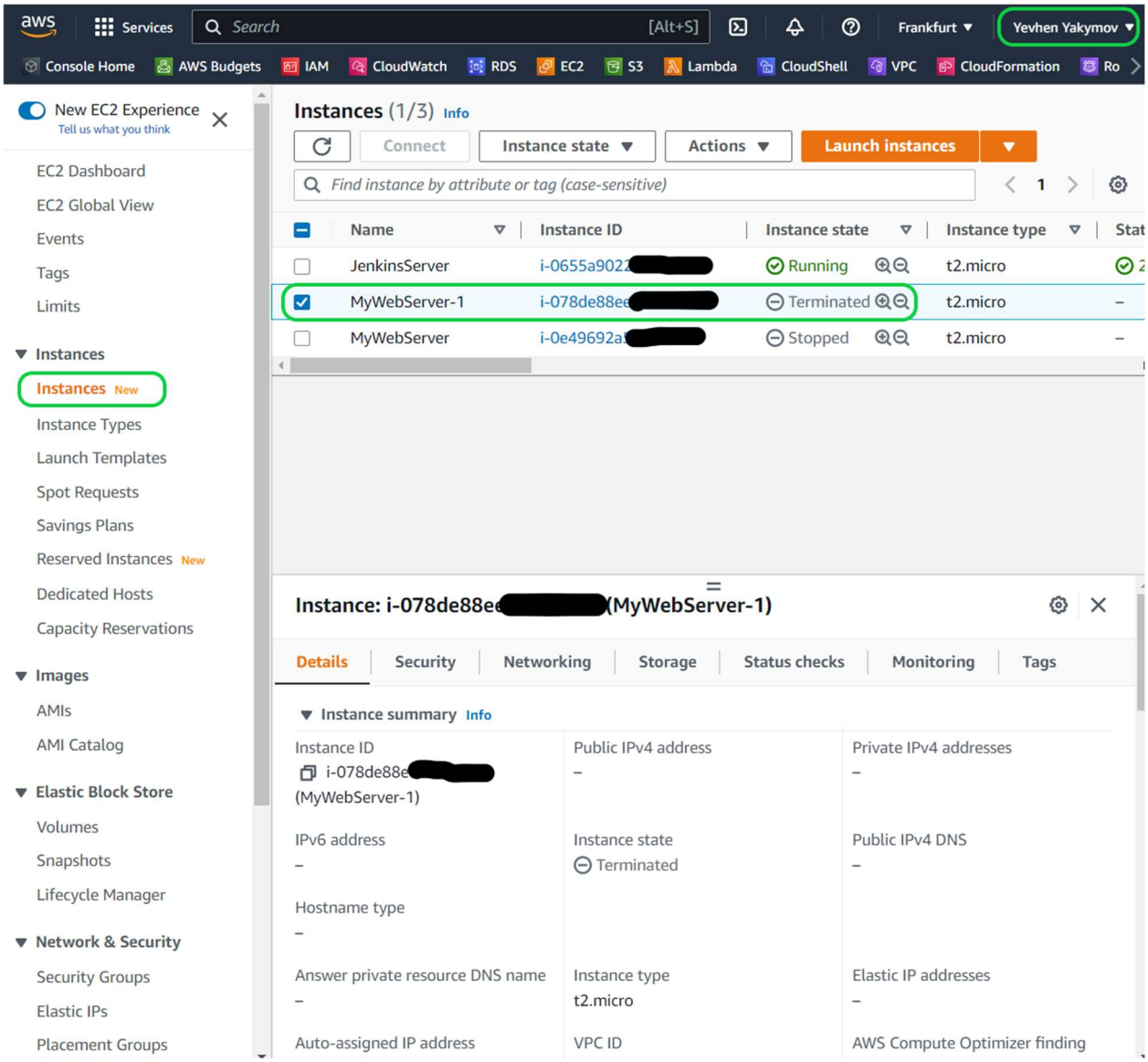
6. Terraform destroy.

terraform destroy – destroy previously-created infrastructure

```
 terraform destroy
aws_security_group.MyWebServer-1: Refreshing state... [id=sg-06206bcf6d05311a7]
aws_instance.MyWebServer-1[0]: Refreshing state... [id=i-078de88ee673b316e]

Terraform used the selected providers to generate the following execution plan. Resource actions
are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.MyWebServer-1[0] will be destroyed
  - resource "aws_instance" "MyWebServer-1" {
      - ami                                  = "ami-076309742d466ad69" -> null
      - arn                                  = "arn:aws:ec2:eu-central-1:389890997503:instance/i-078
de88ee673b316e" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "eu-central-1c" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-078de88ee673b316e" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
```

```
  # aws_security_group.MyWebServer-1 will be destroyed
  - resource "aws_security_group" "MyWebServer-1" {
      - arn                    = "arn:aws:ec2:eu-central-1:389890997503:security-group
05311a7" -> null
      - description            = "Allow TCP inbound traffic" -> null
      - egress                 = [
          - {
```

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

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.MyWebServer-1[0]: Destroying... [id=i-078de88ee673b316e]
aws_instance.MyWebServer-1[0]: Still destroying... [id=i-078de88ee673b316e, 10s elapsed]
aws_instance.MyWebServer-1[0]: Still destroying... [id=i-078de88ee673b316e, 20s elapsed]
aws_instance.MyWebServer-1[0]: Still destroying... [id=i-078de88ee673b316e, 30s elapsed]
aws_instance.MyWebServer-1[0]: Destruction complete after 40s
aws_security_group.MyWebServer-1: Destroying... [id=sg-06206bcf6d05311a7]
aws_security_group.MyWebServer-1: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```


aws
Services
Search
[Alt+S]
Frankfurt
Yevhen Yakymov
Console Home
AWS Budgets
IAM
CloudWatch
RDS
EC2
S3
Lambda
CloudShell
VPC
CloudFormation
New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Instances (1/3) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
Name
Instance ID
Instance state
Instance type
JenkinsServer
Running
t2.micro
MyWebServer-1
Terminated
t2.micro
MyWebServer
Stopped
t2.micro
Instance: i-078de88e (MyWebServer-1)
Details
Security
Networking
Storage
Status checks
Monitoring
Tags
Instance summary Info
Instance ID
(MyWebServer-1)
Public IPv4 address
Private IPv4 addresses
IPv6 address
Instance state
Terminated
Public IPv4 DNS
Hostname type
Answer private resource DNS name
Instance type
t2.micro
Elastic IP addresses
Auto-assigned IP address
VPC ID
AWS Compute Optimizer finding

7. Bootstrapping container instances with Amazon EC2 user data.

You can add in ***.tf** file **user_data** for the section where aws instance is discribed. **user_data** is done via nested Heredoc in the **resource / user_data section**.

```
provider "aws" {
    access_key = "[redacted]"
    secret_key = "[redacted]"
    region = "eu-central-1"
}

resource "aws_instance" "MyWebServer-1" {
        count = 1
        ami = "ami-076309742d466ad69"
        instance_type = "t2.micro"
        vpc_security_group_ids = [aws_security_group.MyWebServer-2.id]
        tags = {
        Name = "MyWebServer-1"
        Project = "MyFirstInfrastructure"
        Owner = "yakymov1yevhen@gmail.com"
        }
        user_data = <<EOF
#!/bin/bash
yum -y update
yum -y install httpd
myip = `curl http://169.254.169.254/latest/meta-data/local-ipv4`
echo "<h2>WebServer with IP: $myip</h2><br>Build by Terraform" > /var/www/html/index.html
sudo systemctl start httpd
sudo systemctl enable httpd
EOF
}

resource "aws_security_group" "MyWebServer-2" {
  name        = "Web Server Security Group"
  description = "Allow TCP inbound traffic"

  ingress {
    description      = "TCP from VPC (HTTP)"
    from_port        = 80
    to_port          = 80
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"]
  }
```

You can also make a shell script and add it to user_data.
user_data = file("script.sh")

It should remembered that the main purpose of Terraform is not a configure infrastructure, but creating infrastructure. So use user_data in specific cases.

8. Terraform implicit dependencies example.

Most common source of dependencies is an implicit dependency between two resources.
Create a directory named **learn-terraform-dependencies** and write configuration into **main.tf**
The **aws_eip** resource type allocate and associates an **Elastic IP** to **EC2** instance. Instance must exist before the **Elastic IP** can be created and attached, Terraform musr ensure that **aws_instance.example_a** was created before it creates **aws_eip.static_ip.** Meanwhile, **aws_instance.example_b** can be created in parallel to the other resources.

```
    Directory: C:\Users\Yevhen Yakymov\Desktop\Programing\DevOps\Terraform\infrastructure\learn-terraform
    -dependencies


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---        06.12.2022     18:38                .terraform
-a----        04.12.2022     22:07           1406 .terraform.lock.hcl
-a----        06.12.2022     18:46           1591 main.tf
-a----        04.12.2022     22:13            180 terraform.tfstate
-a----        04.12.2022     22:13          11495 terraform.tfstate.backup
```

main.tf

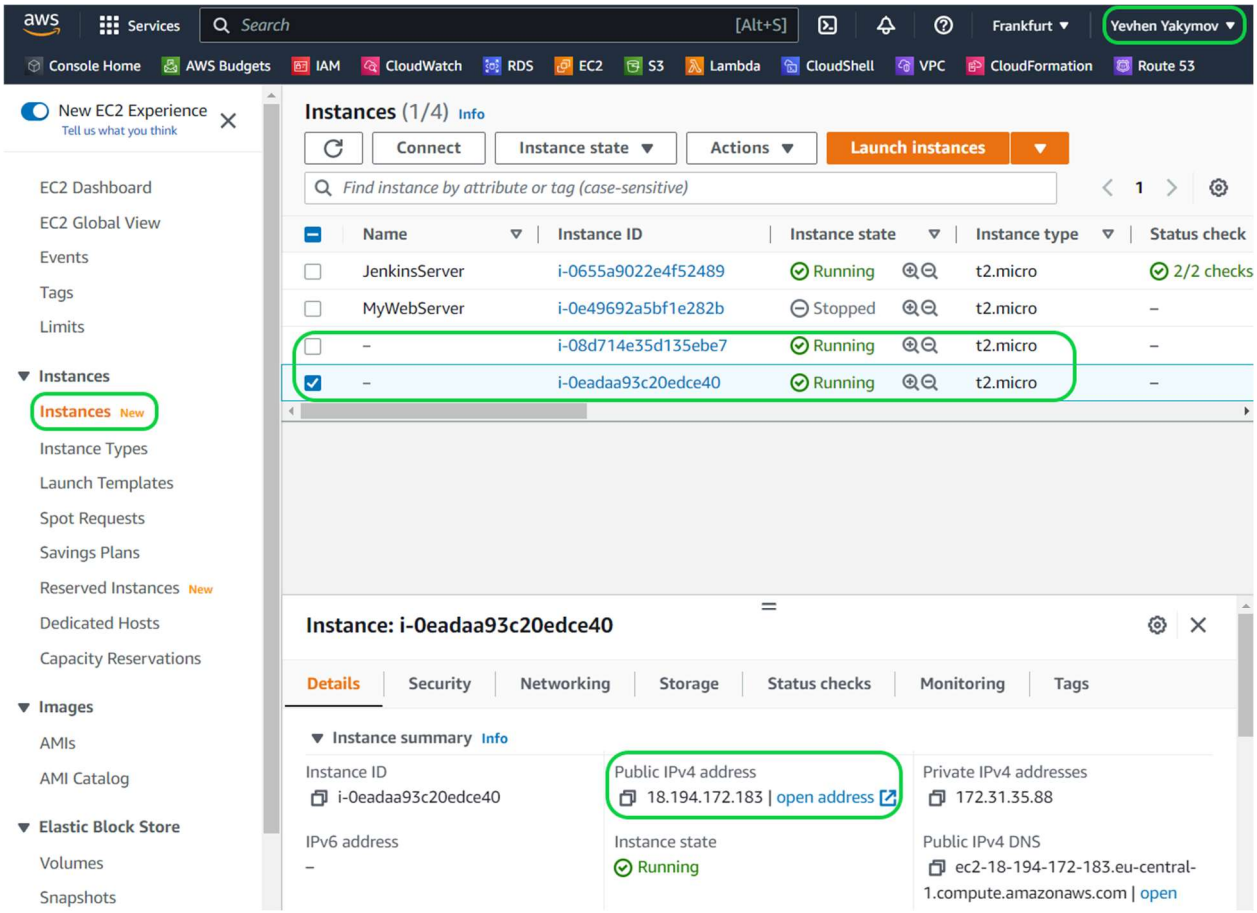
ov > Desktop > Programing > DevOps > Terraform > infrastructure > learn-terraform-dependencies > main.tf >

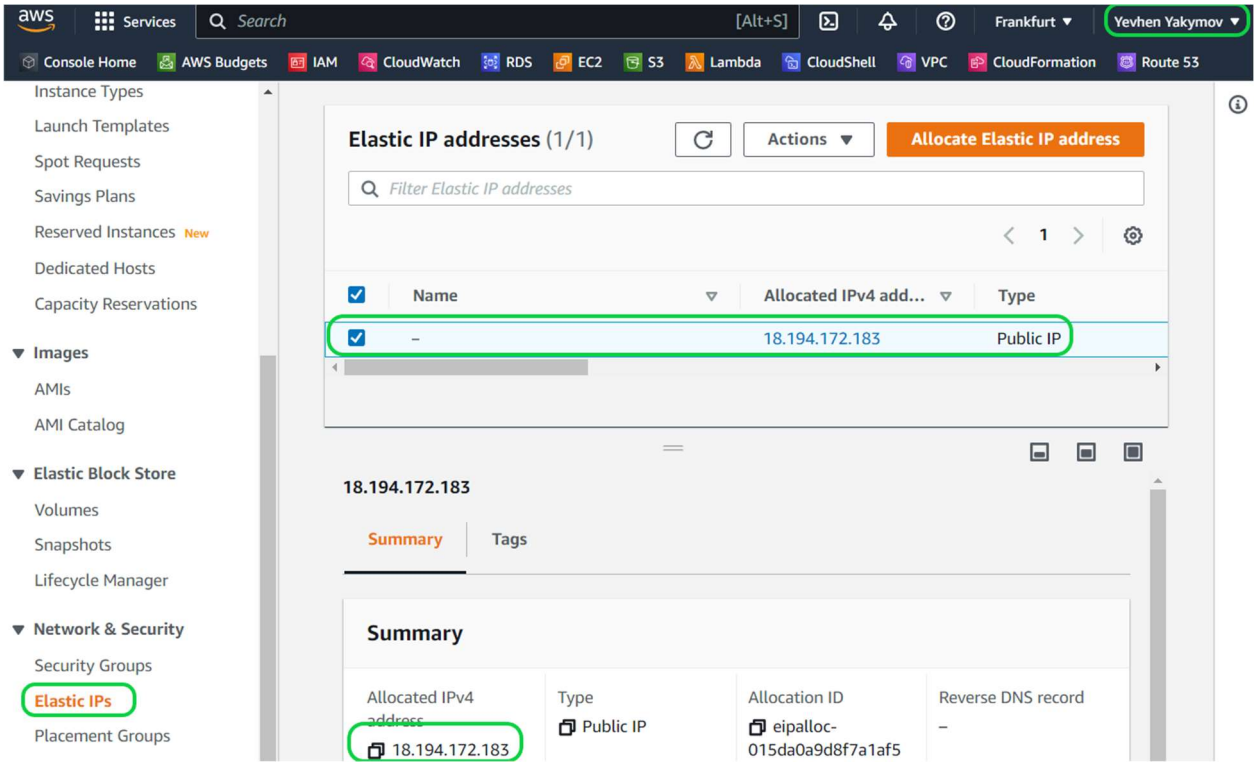
```hcl
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "4.44.0"
    }
  }
}


provider "aws" {
    region = "eu-central-1"
}


data "aws_ami" "amazon_linux" {
  most_recent = true
  owners = ["amazon"]
  filter {
    name = "name"
    values = ["amzn2-ami-kernel-5.10-hvm-*-x86_64-gp2"]
  }
}


resource "aws_instance" "example_a" {
        ami = data.aws_ami.amazon_linux.id
        instance_type = "t2.micro"
}


resource "aws_instance" "example_b" {
        ami = data.aws_ami.amazon_linux.id
        instance_type = "t2.micro"
}

resource "aws_eip" "static_ip"{
  vpc = true
  instance = aws_instance.example_a.id
}
```

```
> terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.44.0

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.
```

```
> terraform plan
data.aws_ami.amazon_linux: Reading...
data.aws_ami.amazon_linux: Read complete after 0s [id=ami-076309742d466ad69]

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

Terraform will perform the following actions:

  # aws_eip.static_ip will be created
  + resource "aws_eip" "static_ip" {
      + allocation_id        = (known after apply)
      + association_id       = (known after apply)
      + carrier_ip           = (known after apply)
      + customer_owned_ip    = (known after apply)
      + domain               = (known after apply)
      + id                   = (known after apply)
      + instance             = (known after apply)
      + network_border_group = (known after apply)
      + network_interface    = (known after apply)
      + private_dns          = (known after apply)
      + private_ip           = (known after apply)
      + public_dns           = (known after apply)
      + public_ip            = (known after apply)
      + public_ipv4_pool     = (known after apply)
      + tags_all             = (known after apply)
      + vpc                  = true
    }

  # aws_instance.example_a will be created
  + resource "aws_instance" "example_a" {
      + ami                                  = "ami-076309742d466ad69"
```

```
Plan: 3 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_instance.example_b: Creating...
aws_instance.example_a: Creating...
aws_instance.example_b: Still creating... [10s elapsed]
aws_instance.example_a: Still creating... [10s elapsed]
aws_instance.example_a: Still creating... [20s elapsed]
aws_instance.example_b: Still creating... [20s elapsed]
aws_instance.example_b: Still creating... [30s elapsed]
aws_instance.example_a: Still creating... [30s elapsed]
aws_instance.example_a: Creation complete after 32s [id=i-0eadaa93c20edce40]
aws_eip.static_ip: Creating...
aws_instance.example_b: Creation complete after 32s [id=i-08d714e35d135ebe7]
aws_eip.static_ip: Creation complete after 1s [id=eipalloc-015da0a9d8f7a1af5]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```






```
> terraform destroy
data.aws_ami.amazon_linux: Reading...
data.aws_ami.amazon_linux: Read complete after 1s [id=ami-076309742d466ad69]
aws_instance.example_b: Refreshing state... [id=i-08d714e35d135ebe7]
aws_instance.example_a: Refreshing state... [id=i-0eadaa93c20edce40]
aws_eip.static_ip: Refreshing state... [id=eipalloc-015da0a9d8f7a1af5]
```

```
Plan: 0 to add, 0 to change, 3 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_eip.static_ip: Destroying... [id=eipalloc-015da0a9d8f7a1af5]
aws_instance.example_b: Destroying... [id=i-08d714e35d135ebe7]
aws_eip.static_ip: Destruction complete after 2s
aws_instance.example_a: Destroying... [id=i-0eadaa93c20edce40]
aws_instance.example_b: Still destroying... [id=i-08d714e35d135ebe7, 10s elapsed]
aws_instance.example_a: Still destroying... [id=i-0eadaa93c20edce40, 10s elapsed]
aws_instance.example_b: Still destroying... [id=i-08d714e35d135ebe7, 20s elapsed]
aws_instance.example_a: Still destroying... [id=i-0eadaa93c20edce40, 20s elapsed]
aws_instance.example_b: Destruction complete after 30s
aws_instance.example_a: Destruction complete after 29s

Destroy complete! Resources: 3 destroyed.
```

9. Terraform explicit dependencies example.

The **depends_on** argument is acepted by any resource or module block and accepts a list of resources to create explicit dependencies for.

Assume we have an app running on your **EC2 instance** that expects to use a specific **Amazon S3 bucket**. This dependency is configured inside the application, and not visible to Terraform. You can use **depends_on** to explicity declare the dependency. You can also specify multiple resources in the **depends_on** argument, and Terraform will wait until all of them have been created before creating the target resources.

Add configuration into the file **main.tf.** First must be created **Amazon S3 bucket** than **EC2 instance** and at the end **module SQS.**

```
44  resource "aws_s3_bucket" "example" {
45    acl = "private"
46  }
47
48  resource "aws_instance" "example_c" {
49    ami = data.aws_ami.amazon_linux.id
50    instance_type = "t2.micro"
51
52    depends_on = [
53      aws_s3_bucket.example
54    ]
55  }
56
57
58  module "example_sqs_queue" {
59    source = "terraform-aws-modules/sqs/aws"
60    version = "2.1.0"
61
62    depends_on = [
63      aws_s3_bucket.example,
64      aws_instance.example_c
65    ]
66  }
```

This configuration includes a reference to a new module, **terraform-aws-modules/sqs/aws.** Modules must be installed before Terraform can use them.

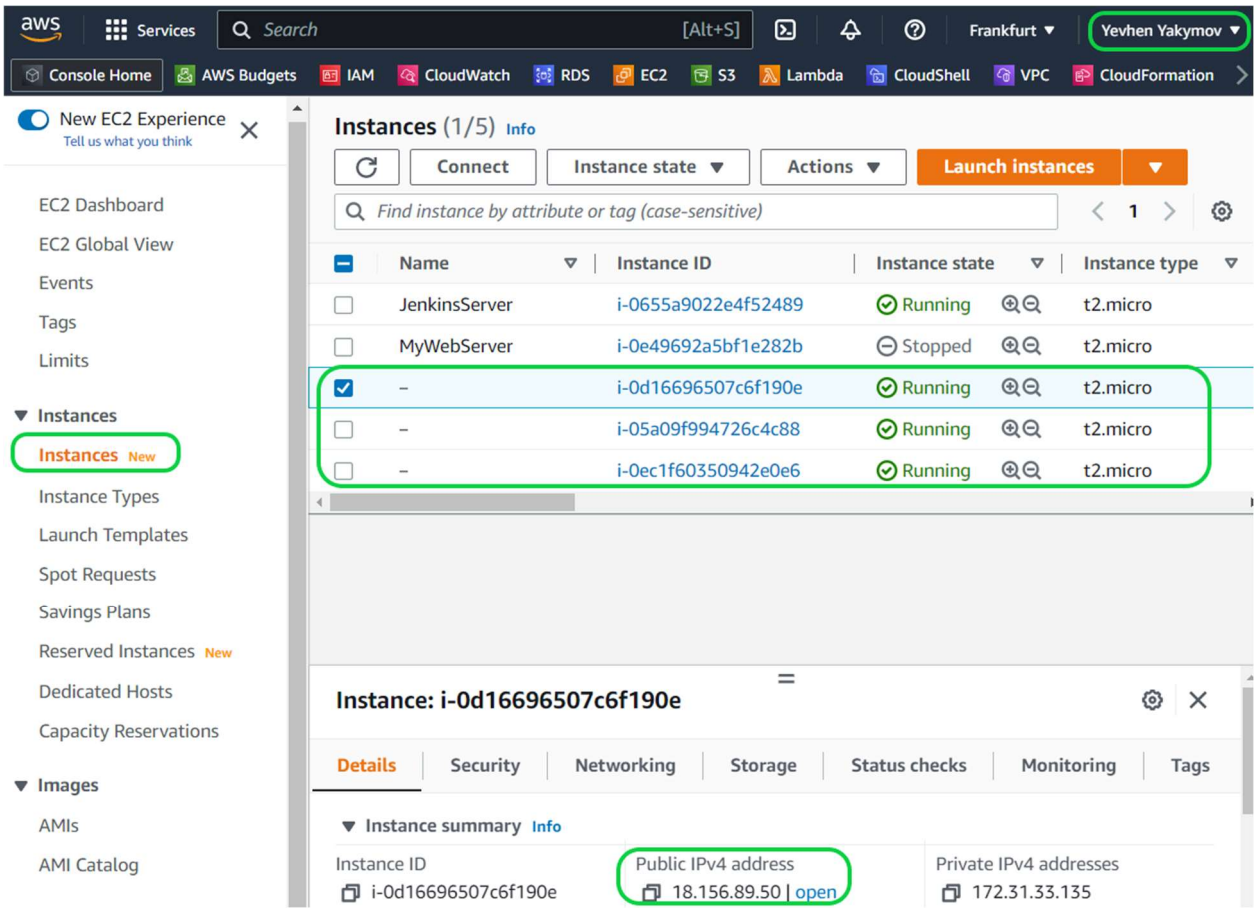
Run **terraform get** to install the module

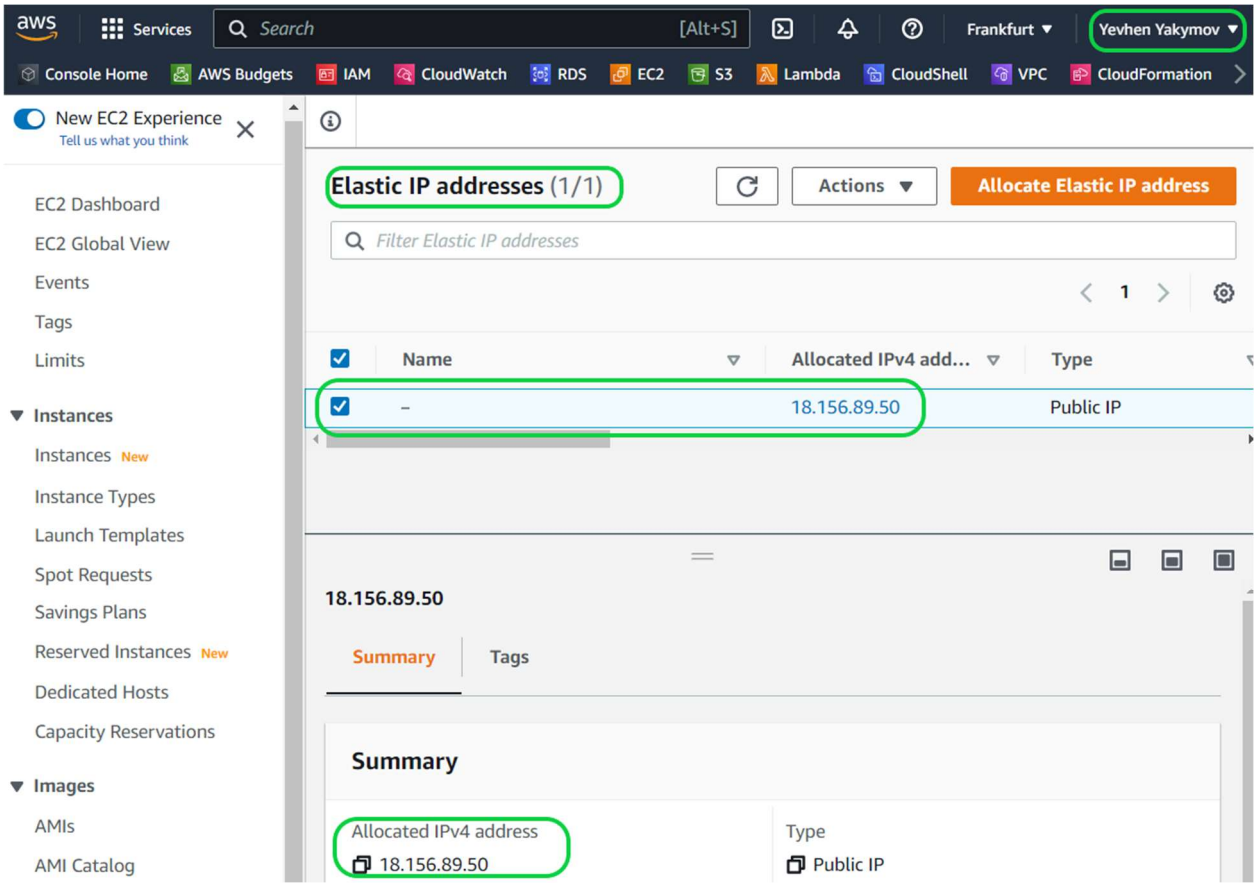
```
> terraform validate

Error: Module not installed

  on main.tf line 58:
  58: module "example_sqs_queue" {

This module is not yet installed. Run "terraform init" to install all modules required by this
configuration.

PS C:\Users\Yevhen Yakymov\Desktop\Programing\DevOps\Terraform\infrastructure\learn-terraform-dependencies
> terraform get
Downloading registry.terraform.io/terraform-aws-modules/sqs/aws 2.1.0 for example_sqs_queue...
- example_sqs_queue in .terraform\modules\example_sqs_queue
```

Run **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_s3_bucket.example: Creating...
aws_instance.example_b: Creating...
aws_instance.example_a: Creating...
aws_s3_bucket.example: Creation complete after 1s [id=terraform-20221206182610910000000001]
aws_instance.example_c: Creating...
aws_instance.example_b: Still creating... [10s elapsed]
aws_instance.example_a: Still creating... [10s elapsed]
aws_instance.example_c: Still creating... [10s elapsed]
aws_instance.example_a: Still creating... [20s elapsed]
aws_instance.example_b: Still creating... [20s elapsed]
aws_instance.example_c: Still creating... [20s elapsed]
aws_instance.example_a: Still creating... [30s elapsed]
aws_instance.example_b: Still creating... [30s elapsed]
aws_instance.example_c: Still creating... [30s elapsed]
aws_instance.example_a: Creation complete after 31s [id=i-0d16696507c6f190e]
aws_eip.static_ip: Creating...
aws_instance.example_b: Creation complete after 32s [id=i-0ec1f60350942e0e6]
aws_eip.static_ip: Creation complete after 2s [id=eipalloc-0b4ff1cf915b61d35]
aws_instance.example_c: Creation complete after 32s [id=i-05a09f994726c4c88]
module.example_sqs_queue.aws_sqs_queue.this[0]: Creating...
module.example_sqs_queue.aws_sqs_queue.this[0]: Still creating... [10s elapsed]
module.example_sqs_queue.aws_sqs_queue.this[0]: Still creating... [20s elapsed]
module.example_sqs_queue.aws_sqs_queue.this[0]: Creation complete after 26s [id=https://sqs.eu-central-1.amazonaws.c
om/389890997503/terraform-20221206182644067800000006]
module.example_sqs_queue.data.aws_arn.this[0]: Reading...
module.example_sqs_queue.data.aws_arn.this[0]: Read complete after 0s [id=arn:aws:sqs:eu-central-1:389890997503:terr
aform-20221206182644067800000006]

  Warning: Argument is deprecated

    with aws_s3_bucket.example,
    on main.tf line 45, in resource "aws_s3_bucket" "example":
    45:   acl = "private"

  Use the aws_s3_bucket_acl resource instead


Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
PS C:\Users\Yevhen Yakymov\Desktop\Programing\DevOps\Terraform\infrastructure\learn-terraform-dependencies>
```

Since both the instance "**example_c**" and **SQS Queue** are dependent upon the **S3 bucket**, Terraform waits until the bucket is created to begin creating the other two resources.


aws
Services
Search
[Alt+S]
Frankfurt
Yevhen Yakymov
Console Home
AWS Budgets
IAM
CloudWatch
RDS
EC2
S3
Lambda
CloudShell
VPC
CloudFormation
New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Instances (1/5) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
Name
Instance ID
Instance state
Instance type
JenkinsServer
i-0655a9022e4f52489
Running
t2.micro
MyWebServer
i-0e49692a5bf1e282b
Stopped
t2.micro
–
i-0d16696507c6f190e
Running
t2.micro
–
i-05a09f994726c4c88
Running
t2.micro
–
i-0ec1f60350942e0e6
Running
t2.micro
Instance: i-0d16696507c6f190e
Details
Security
Networking
Storage
Status checks
Monitoring
Tags
Instance summary Info
Instance ID
i-0d16696507c6f190e
Public IPv4 address
18.156.89.50 | open
Private IPv4 addresses
172.31.33.135


aws
Services
Search
[Alt+S]
Frankfurt
Yevhen Yakymov
Console Home
AWS Budgets
IAM
CloudWatch
RDS
EC2
S3
Lambda
CloudShell
VPC
CloudFormation
New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic IP addresses (1/1)
Actions
Allocate Elastic IP address
Filter Elastic IP addresses
Name
Allocated IPv4 add...
Type
–
18.156.89.50
Public IP
18.156.89.50
Summary
Tags
Summary
Allocated IPv4 address
18.156.89.50
Type
Public IP

Run **terraform destroy**

```
> terraform destroy
data.aws_ami.amazon_linux: Reading...
aws_s3_bucket.example: Refreshing state... [id=terraform-20221206182610910000000001]
data.aws_ami.amazon_linux: Read complete after 0s [id=ami-076309742d466ad69]
aws_instance.example_b: Refreshing state... [id=i-0ec1f60350942e0e6]
aws_instance.example_a: Refreshing state... [id=i-0d16696507c6f190e]
aws_instance.example_c: Refreshing state... [id=i-05a09f994726c4c88]
aws_eip.static_ip: Refreshing state... [id=eipalloc-0b4ff1cf915b61d35]
module.example_sqs_queue.aws_sqs_queue.this[0]: Refreshing state... [id=https://sqs.eu-central-1.amazonaws
.com/389890997503/terraform-20221206182644067800000006]
module.example_sqs_queue.data.aws_arn.this[0]: Reading...
module.example_sqs_queue.data.aws_arn.this[0]: Read complete after 0s [id=arn:aws:sqs:eu-central-1:3898909
97503:terraform-20221206182644067800000006]

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

```
aws_eip.static_ip: Destroying... [id=eipalloc-0b4ff1cf915b61d35]
module.example_sqs_queue.aws_sqs_queue.this[0]: Destroying... [id=https://sqs.eu-central-1.amazonaws.com/3
89890997503/terraform-20221206182644067800000006]
aws_instance.example_b: Destroying... [id=i-0ec1f60350942e0e6]
aws_eip.static_ip: Destruction complete after 1s
aws_instance.example_a: Destroying... [id=i-0d16696507c6f190e]
module.example_sqs_queue.aws_sqs_queue.this[0]: Destruction complete after 2s
aws_instance.example_c: Destroying... [id=i-05a09f994726c4c88]
aws_instance.example_b: Still destroying... [id=i-0ec1f60350942e0e6, 10s elapsed]
aws_instance.example_a: Still destroying... [id=i-0d16696507c6f190e, 10s elapsed]
aws_instance.example_c: Still destroying... [id=i-05a09f994726c4c88, 10s elapsed]
aws_instance.example_b: Still destroying... [id=i-0ec1f60350942e0e6, 20s elapsed]
aws_instance.example_a: Still destroying... [id=i-0d16696507c6f190e, 20s elapsed]
aws_instance.example_c: Still destroying... [id=i-05a09f994726c4c88, 20s elapsed]
aws_instance.example_b: Still destroying... [id=i-0ec1f60350942e0e6, 30s elapsed]
aws_instance.example_a: Still destroying... [id=i-0d16696507c6f190e, 30s elapsed]
aws_instance.example_c: Destruction complete after 30s
aws_s3_bucket.example: Destroying... [id=terraform-20221206182610910000000001]
aws_s3_bucket.example: Destruction complete after 0s
aws_instance.example_b: Still destroying... [id=i-0ec1f60350942e0e6, 40s elapsed]
aws_instance.example_b: Destruction complete after 40s
aws_instance.example_a: Destruction complete after 40s

Destroy complete! Resources: 6 destroyed.
```

Both implicit and explicit dependencies affect the order in which resources are destroyed as well as created.

10. Terraform Variables.

According our goal, you should have a directory named **learn-terraform-aws-variables** with the following configuration in a file **main.tf.**

variable.tf main.tf

main.tf

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "4.44.0"
    }
  }
}


provider "aws" {
    profile = "default"
    region = "eu-central-1"
}


data "aws_ami" "amazon_linux" {
  most_recent = true
  owners = ["amazon"]
  filter {
    name = "name"
    values = ["amzn2-ami-kernel-5.10-hvm-*-x86_64-gp2"]
  }
}


resource "aws_instance" "example" {
        ami = data.aws_ami.amazon_linux.id
        instance_type = "t2.micro"
        tags = {
          Name = "ExampleInstance"
  }
}
```

Set the instance name with a variable.

The configuration includes a number of "hard-coded" values. Terraform variebles allow you to write configuration that is flexible and easier to re-use.

Add a variable to define the instance name.

Creating a new file called **variables.tf** with a block defining a new **instance_name** variable.

variable.tf

```
variable "instance_name" {
  type        = string
  default     = "ExampleInstance"
  description = "Value of the Name tag for the EC2 instance"
}

```

Note: Terraform loads all files in the current directory ending in **.tf** , so you can name your configuration files however you choose.

In **main.tf,** update the **aws_instance** resource block to use the new variable.

```
data "aws_ami" "amazon_linux" {
  most_recent = true
  owners = ["amazon"]
  filter {
    name = "name"
    values = ["amzn2-ami-kernel-5.10-hvm-*-x86_64-gp2"]
  }
}


resource "aws_instance" "example" {
        ami = data.aws_ami.amazon_linux.id
        instance_type = "t2.micro"
        tags = {
          Name = var.instance_name
  }
}
```

Apply the configuration. Respond to the confirmation promt with a "yes".

Another one way to set the variable is put in command line with option **-var:**

```
terraform apply -var "instance_name=YetAnotherName"
```

11. Terraform Output.

Output EC2 instance configuration

According our goal, you should have a directory named **learn-terraform-aws-outputs** with the following configuration in a file **main.tf.**

output.tf ● variable.tf main.tf ×

main.tf

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "4.44.0"
    }
  }
}


provider "aws" {
    profile = "default"
    region = "eu-central-1"
}


data "aws_ami" "amazon_linux" {
  most_recent = true
  owners = ["amazon"]
  filter {
    name = "name"
    values = ["amzn2-ami-kernel-5.10-hvm-*-x86_64-gp2"]
  }
}


resource "aws_instance" "example" {
        ami = data.aws_ami.amazon_linux.id
        instance_type = "t2.micro"
        tags = {
          Name = var.instance_name
  }
}
```

Create a file called **output.tf** in your **learn-terraform-aws-outputs** directory.

Add outputs to the new file for your EC2 inastance's ID and IP address.

Terraform prints output values to the screen when you apply your configuration. Query the **outputs** with the **terraform output** command.

```
$ terraform output
instance_id = "i-0bf954919ed765de1"
instance_public_ip = "54.186.202.254"
```

You can use Terraform **outputs** to connect your Terraform projects with other parts of your infrastructure, or with other Terrafirm projects.

12. Terraform Modules.

As you manage your infrastructure with Terraform, you will create increasingly complex configurations. There is no limit to the complexity of a single Terraform configuration file or directory, so it is possible to continue writing and updating your configurationfiles in a single directory. However, if you do, you may encounter one or more problems:

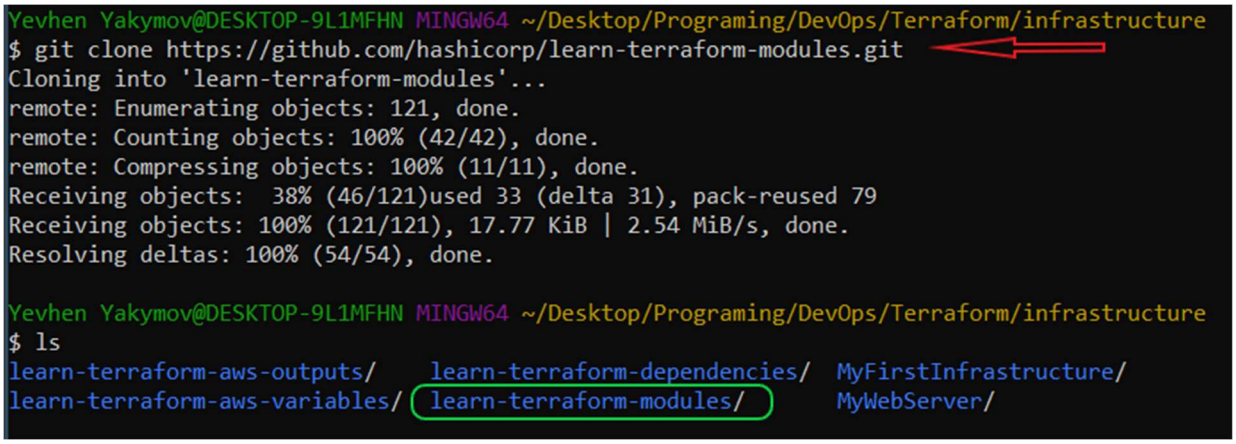
- Understanding and navigating the configuration files will become increasingly difficult
- Updating the configuration will become more risky, as an update to one section may cause unintended consequences to other parts of your configuration
- There will be an increasing amount of duplication of similar blocks of configuration, for instance when configuration separete dev/staging/production environments, which will cause an incresing burdenwhen updating those parts of your configuration
- You may wish to share parts of your configuration between projects and tems, and will quickly find that cutting and pasting blocks of configuration between projects is error prone and hard to maintain.

So main goal of creating and using Terraform modules is to simplify your current workflow.

Create Terraform configuration

For example, you will use **modules** to create an example AWS envirinment using a VPC and two EC2 instances. You can create it by manualy building the directory structure and files using the following commands to clone this GitHub repo.

Clone the GitHub repository.



Change into that directory in your terminal

Check out the ec2-instances tag into a local branch

Configuration in a file **main.tf.**

main.tf variables.tf outputs.tf

main.tf > ...

```
# Terraform configuration

terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
    }
  }
}

provider "aws" {
  region = "us-west-2"
}

module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "2.21.0"

  name = var.vpc_name
  cidr = var.vpc_cidr

  azs             = var.vpc_azs
  private_subnets = var.vpc_private_subnets
  public_subnets  = var.vpc_public_subnets

  enable_nat_gateway = var.vpc_enable_nat_gateway

  tags = var.vpc_tags
}

module "ec2_instances" {
  source  = "terraform-aws-modules/ec2-instance/aws"
  version = "2.12.0"

  name           = "my-ec2-cluster"
  instance_count = 2

  ami                    = "ami-0c5204531f799e0c6"
  instance_type          = "t2.micro"
  vpc_security_group_ids = [module.vpc.default_security_group_id]
  subnet_id              = module.vpc.public_subnets[0]

  tags = {
    Terraform   = "true"
    Environment = "dev"
  }
}
```