

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

The screenshot shows the Terraform website at [https://www.terraform.io/downloads.html](#). The top navigation bar includes a search icon and a menu icon. Below the header, the URL is shown as "Developer / Terraform / Install v1.3.5". The main content area features the Terraform logo and the heading "Install Terraform". A sub-header says "Install or update to v1.3.5 (latest version) of Terraform to get started." Below this, there's a section titled "Operating System" with tabs for macOS, Windows (which is highlighted with a green oval), Linux, FreeBSD, OpenBSD, and Solaris. Under the Windows tab, two download options are listed: "Binary download for Windows" (386) and "Binary download for Windows" (AMD64). Each option shows the file size (386 is 386 KB, AMD64 is 1.35 MB), the version (1.3.5), and a "Download" button. The AMD64 download button is also highlighted with a green oval.

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

The screenshot shows the Windows Control Panel search results for "Edit environment variables for your account". The result "Edit environment variables for your account" is selected and highlighted with a blue oval. This leads to the "Edit environment variables for your account" Control Panel page, which has an "Open" button highlighted with a blue oval. From this page, the "Edit environment variables for your account" link is selected and highlighted with a blue oval, leading to the "Environment Variables" dialog. This dialog shows the "User variables" and "System variables" sections. In the "System variables" section, there is a new entry for "PATH" with the value "C:\Python39\Scripts;C:\Python39;C:\ProgramData\Anaconda3\;C...\;CMD;EXE;BAT;CMD;VBS;VBE;JS;JSE;WSH;MSC;PY;PW...". The "Edit..." button for this entry is highlighted with a blue oval. A separate "Edit environment variable" dialog is also visible on the right, showing the "Edit environment variable" title and a list of environment variables with their paths. The "OK" button in this dialog is highlighted with a blue oval.

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)

Developer / Terraform / Install v1.3.5

The screenshot shows the Terraform website's "Install" page for version 1.3.5. It features a "Install Terraform" button with a purple logo. Below it, a note says "Install or update to v1.3.5 (latest version) of Terraform to get started." The "Operating System" section has tabs for macOS, Windows, Linux (which is selected), FreeBSD, OpenBSD, and Solaris. A "Copy" button is available for the Linux package manager instructions. The "Binary download for Linux" section offers 386 and AMD64 versions, each with a "Download" button.

Operating System

macOS Windows **Linux** FreeBSD OpenBSD Solaris

1.3.5 (latest) ⚙

Package manager for Linux

Ubuntu/Debian CentOS/RHEL Fedora Amazon Linux Homebrew

```
$ sudo yum install -y yum-utils shadow-utils
$ sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
$ sudo yum -y install terraform
```

Copy ⚙

Binary download for Linux

386 Version: 1.3.5 Download ⚙

AMD64 Version: 1.3.5 Download ⚙

```
[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 |
|-------------|--------|---------|------------|------|
| <hr/>       |        |         |            |      |
| 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

The screenshot shows the 'Add user' wizard step 1: Set user details. The 'User name*' field contains 'terraform' and is highlighted with a green box. Below it is a link to 'Add another user'. The 'Select AWS access type' section follows, with the 'Select AWS credential type*' dropdown set to 'Access key - Programmatic access' (which is also highlighted with a green box). This option enables programmatic access via access key ID and secret access key.

The screenshot shows the 'Add user' wizard step 2: Set permissions. It includes options to 'Add user to group', 'Copy permissions from existing user', and 'Attach existing policies directly' (which is highlighted with a green box). Below is a table of policies, with 'AdministratorAccess' selected (indicated by a checked checkbox and highlighted with a red box).

| Policy name | Type | Used as |
|---|--------------|------------------------|
| <input checked="" type="checkbox"/> AdministratorAccess | Job function | Permissions policy (1) |
| <input type="checkbox"/> AdministratorAccess-Amplify | AWS managed | None |

The screenshot shows the 'Add user' wizard step 3: Add tags (optional). It displays a table where a tag named 'name' with the value 'terraform' is listed. There is a 'Remove' link next to the tag entry. A note at the bottom states: 'You can add 49 more tags.'

AWS Services Search [Alt+S] Global ▾ Yevhen Yakymov ▾

Console Home AWS Budgets IAM CloudWatch RDS EC2 S3 Lambda CloudShell VPC CloudFormation Route 53

Add user

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

| | |
|----------------------|--|
| User name | terraform |
| AWS access type | Programmatic access - with an access key |
| Permissions boundary | Permissions boundary is not set |

Permissions summary

The following policies will be attached to the user shown above.

| Type | Name |
|----------------|---------------------|
| Managed policy | AdministratorAccess |

Tags

The new user will receive the following tag

| Key | Value |
|------|-----------|
| name | terraform |

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)

AWS Services Search [Alt+S] Global ▾ Yevhen Yakymov ▾

Console Home AWS Budgets IAM CloudWatch RDS EC2 S3 Lambda CloudShell VPC CloudFormation Route 53

Add user

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://389890997503.signin.aws.amazon.com/console>

[Download .csv](#)

| | User | Access key ID | Secret access key |
|---|-----------|----------------------|-------------------|
| ▼ | terraform | AKIAVVR2QPT74XR7XHWI | ***** Show |

Created user terraform
Attached policy AdministratorAccess to user terraform
Created access key for user terraform

3. Creating infrastructure.

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

```
[ec2-user@ip-172-31-14-46 MyFirstInfrastructure]$ touch myinfrastructure.tf
[ec2-user@ip-172-31-14-46 MyFirstInfrastructure]$
[ec2-user@ip-172-31-14-46 MyFirstInfrastructure]$ vim myinfrastructure.tf
provider "aws" {
    access_key = "AKIA"
    secret_key = "vHMhh"
    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_ACES_KEY_ID="XXXXXXXXXXXX"
export AWS_SECRET_ACES_KEY="XXXXXXXXXXXX"
export AWS_DEFAULT_REGION="your-region"
```

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

Go

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

- c) in the credentials file:

~/path/to/file/credentials

Text

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

~/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-06206bcf6d0...]
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-078de88ee6...]
```

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

AWS Services Search Frankfurt Yevhen Yakymov

Console Home AWS Budgets IAM CloudWatch RDS EC2 S3 Lambda CloudShell VPC CloudFormation Ro >

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

| Name | Instance ID | Instance state | Instance type | Status |
|---------------|-------------|----------------|---------------|--------|
| JenkinsServer | i-0655a902 | Running | t2.micro | 2, - |
| MyWebServer-1 | i-078de88e | Pending | t2.micro | - |
| MyWebServer | i-0e49692a | Stopped | t2.micro | - |

Select an instance

AWS Services Search Frankfurt Yevhen Yakymov

Console Home AWS Budgets IAM CloudWatch RDS EC2 S3 Lambda CloudShell VPC CloudFormation Ro >

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

| Name | Security group ID | Security group name | VPC ID |
|--------------|-------------------|--------------------------|-----------|
| - | sg-0e367306d | default | vpc-08d25 |
| - | sg-0b0fcdf75 | WebServerGroup | vpc-08d25 |
| allow_tcp | sg-06206bcf6 | Web Server Security G... | vpc-08d25 |
| JenkinsGroup | sg-0e79cbba8 | JenkinsGroup | vpc-08d25 |

Inbound rules (3)

| 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 Ro >

New EC2 Experience Tell us what you think X

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 | Status |
|---------------|----------------------|----------------|---------------|--------|
| JenkinsServer | i-0655a902[REDACTED] | Running | t2.micro | 2 |
| MyWebServer-1 | i-078de88e[REDACTED] | Terminated | t2.micro | - |
| MyWebServer | i-0e49692a[REDACTED] | Stopped | t2.micro | - |

Instance: i-078de88e[REDACTED] (MyWebServer-1)

Details Security Networking Storage Status checks Monitoring Tags

Instance summary

| | | |
|--|------------------------------|-------------------------------|
| Instance ID i-078de88e[REDACTED] (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 described. **user_data** is done via nested Heredoc in the **resource / user_data** section.

```
provider "aws" {
    access_key = "XXXXXXXXXXXXXX"
    secret_key = "XXXXXXXXXXXXXXXXXXXXXX"
    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 must 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\Programming\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  x
ov > Desktop > Programming > DevOps > Terraform > infrastructure > learn-terraform-dependencies > main.tf >
1  terraform {
2    required_providers {
3      aws = [
4        {
5          source = "hashicorp/aws"
6          version = "4.44.0"
7        }
8      }
9    }
10
11   provider "aws" {
12     region = "eu-central-1"
13   }
14
15
16   data "aws_ami" "amazon_linux" {
17     most_recent = true
18     owners = ["amazon"]
19     filter {
20       name = "name"
21       values = ["amzn2-ami-kernel-5.10-hvm-*-x86_64-gp2"]
22     }
23   }
24
25
26   resource "aws_instance" "example_a" {
27     ami = data.aws_ami.amazon_linux.id
28     instance_type = "t2.micro"
29   }
30
31
32   resource "aws_instance" "example_b" {
33     ami = data.aws_ami.amazon_linux.id
34     instance_type = "t2.micro"
35   }
36
37   resource "aws_eip" "static_ip"{
38     vpc = true
39     instance = aws_instance.example_a.id
40   }
```

```
> 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.
```

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with links like 'EC2 Dashboard', 'Events', 'Tags', 'Limits', 'Instances' (which is expanded), 'Images', and 'Elastic Block Store'. Under 'Instances', 'Instances New' is highlighted. The main pane displays a table of instances with columns: Name, Instance ID, Instance state, Instance type, and Status check. Three instances are listed: 'JenkinsServer' (Stopped, t2.micro), 'MyWebServer' (Stopped, t2.micro), and 'i-0eadaa93c20edce40' (Running, t2.micro). The last instance is selected, indicated by a checked checkbox and a green border around the row. Below the table, a detailed view for 'Instance: i-0eadaa93c20edce40' is shown. The 'Details' tab is active, displaying the 'Instance summary' section. It shows the Instance ID (i-0eadaa93c20edce40), Public IPv4 address (18.194.172.183), Private IPv4 addresses (172.31.35.88), IPv6 address (-), Instance state (Running), and Public IPv4 DNS (ec2-18-194-172-183.eu-central-1.compute.amazonaws.com).

The screenshot shows the AWS Management Console with the search bar set to "Elastic IP addresses". The left sidebar is expanded to show "Elastic IPs" under "Network & Security". A table lists one elastic IP address:

| Name | Allocated IPv4 add... | Type |
|------|-----------------------|-----------|
| - | 18.194.172.183 | Public IP |

Below the table, a summary card provides details for the selected IP address:

| Allocated IPv4 address | Type | Allocation ID | Reverse DNS record |
|------------------------|-----------|----------------------------|--------------------|
| 18.194.172.183 | Public IP | eipalloc-015da0a9d8f7a1af5 | - |

```
> 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 accepted 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 explicitly 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\Programming\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.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-1:389890997503:terraform-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\Programming\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.

Screenshot of the AWS EC2 Instances page showing a list of running t2.micro instances. One instance, 'i-0d16696507c6f190e', is highlighted with a green box.

Instances (1/5) Info

| 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

| | | |
|------------------------------------|---|---|
| Instance ID i-0d16696507c6f190e | Public IPv4 address 18.156.89.50 | Private IPv4 addresses 172.31.33.135 |
|------------------------------------|---|---|

Screenshot of the AWS Elastic IP addresses page showing a list of allocated public IPs. One IP, '18.156.89.50', is highlighted with a green box.

Elastic IP addresses (1/1)

| 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-1:389890997503: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/389890997503/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
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "4.44.0"
6     }
7   }
8 }
9
10 provider "aws" {
11   profile = "default"
12   region = "eu-central-1"
13 }
14
15
16
17 data "aws_ami" "amazon_linux" {
18   most_recent = true
19   owners = ["amazon"]
20   filter {
21     name = "name"
22     values = ["amzn2-ami-kernel-5.10-hvm-*-x86_64-gp2"]
23   }
24 }
25
26
27 resource "aws_instance" "example" {
28   ami = data.aws_ami.amazon_linux.id
29   instance_type = "t2.micro"
30   tags = {
31     Name = "ExampleInstance"
32   }
33 }
```

Set the instance name with a variable.

The configuration includes a number of “hard-coded” values. Terraform variables 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"
}

main.tf
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 }
  ]
}
```

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



```
 1  terraform {
 2      required_providers {
 3          aws = {
 4              source = "hashicorp/aws"
 5              version = "4.44.0"
 6          }
 7      }
 8  }
 9
10
11 provider "aws" {
12     profile = "default"
13     region = "eu-central-1"
14 }
15
16
17 data "aws_ami" "amazon_linux" {
18     most_recent = true
19     owners = ["amazon"]
20     filter {
21         name = "name"
22         values = ["amzn2-ami-kernel-5.10-hvm-*-x86_64-gp2"]
23     }
24 }
25
26
27 resource "aws_instance" "example" {
28     ami = data.aws_ami.amazon_linux.id
29     instance_type = "t2.micro"
30     tags = {
31         Name = var.instance_name
32     }
33 }
```

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

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



```
output.tf • variable.tf main.tf
output.tf
1 output "instance_id" {
2   value      = "aws_instance.example.id"
3   description = "ID of the EC2 instance"
4 }
5
6
7 output "instance_public_ip" {
8   value      = "aws_instance.example.public_ip"
9   description = "Public IP address of the EC2 instance"
10 }
```

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 Terraform 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 configuration files 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 separate dev/staging/production environments, which will cause an increasing burden when updating those parts of your configuration
- You may wish to share parts of your configuration between projects and teams, 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 environment using a VPC and two EC2 instances. You can create it by manually building the directory structure and files using the following commands to clone this GitHub repo.

Clone the GitHub repository.

```
Yevhen Yakymov@DESKTOP-9L1MFHN MINGW64 ~/Desktop/Programming/DevOps/Terraform/infrastructure
$ git clone https://github.com/hashicorp/learn-terraform-modules.git
Cloning into 'learn-terraform-modules'...
remote: Enumerating objects: 121, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (11/11), done.
Receiving objects: 38% (46/121) used 33 (delta 31), pack-reused 79
Receiving objects: 100% (121/121), 17.77 KiB | 2.54 MiB/s, done.
Resolving deltas: 100% (54/54), done.

Yevhen Yakymov@DESKTOP-9L1MFHN MINGW64 ~/Desktop/Programming/DevOps/Terraform/infrastructure
$ ls
learn-terraform-aws-outputs/ learn-terraform-dependencies/ MyFirstInfrastructure/
learn-terraform-aws-variables/ learn-terraform-modules/ MyWebServer/
```

Change into that directory in your terminal

```
Yevhen Yakymov@DESKTOP-9L1MFHN MINGW64 ~/Desktop/Programming/DevOps/Terraform/infrastructure
$ cd learn-terraform-modules/

Yevhen Yakymov@DESKTOP-9L1MFHN MINGW64 ~/Desktop/Programming/DevOps/Terraform/infrastructure/learn-terraform-modules (main)
$ ls
LICENSE main.tf modules/ outputs.tf README.md variables.tf
```

Check out the ec2-instances tag into a local branch

```
Yevhen Yakymov@DESKTOP-9L1MFHN MINGW64 ~/Desktop/Programming/DevOps/Terraform/infrastructure/learn-terraform-modules (main)
$ git checkout tags/ec2-instances -b ec2-instances

Yevhen Yakymov@DESKTOP-9L1MFHN MINGW64 ~/Desktop/Programming/DevOps/Terraform/infrastructure/learn-terraform-modules (ec2-instances)
```

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



The screenshot shows a code editor interface with three tabs at the top: **main.tf**, **variables.tf**, and **outputs.tf**. The **main.tf** tab is active and displays the following Terraform configuration code:

```
1 # Terraform configuration
2
3 terraform {
4   required_providers {
5     aws = {
6       source = "hashicorp/aws"
7     }
8   }
9 }
10
11 provider "aws" {
12   region = "us-west-2"
13 }
14
15 module "vpc" {
16   source  = "terraform-aws-modules/vpc/aws"
17   version = "2.21.0"
18
19   name = var.vpc_name
20   cidr = var.vpc_cidr
21
22   azs           = var.vpc_azs
23   private_subnets = var.vpc_private_subnets
24   public_subnets = var.vpc_public_subnets
25
26   enable_nat_gateway = var.vpc_enable_nat_gateway
27
28   tags = var.vpc_tags
29 }
30
31 module "ec2_instances" {
32   source  = "terraform-aws-modules/ec2-instance/aws"
33   version = "2.12.0"
34
35   name           = "my-ec2-cluster"
36   instance_count = 2
37
38   ami             = "ami-0c5204531f799e0c6"
39   instance_type    = "t2.micro"
40   vpc_security_group_ids = [module.vpc.default_security_group_id]
41   subnet_id       = module.vpc.public_subnets[0]
42
43   tags = {
44     Terraform  = "true"
45     Environment = "dev"
46   }
47 }
```