

PluralSight Terraform report

Installing Terraform

```
C:\Users\Asus>choco install terraform
Chocolatey v2.3.0
3 validations performed. 2 success(es), 1 warning(s), and 0 error(s).

Validation Warnings:
- A pending system reboot request has been detected, however, this is
  being ignored due to the current Chocolatey configuration. If you
  want to halt when this occurs, then either set the global feature
  using:
    choco feature enable --name="exitOnRebootDetected"
  or pass the option --exit-when-reboot-detected.

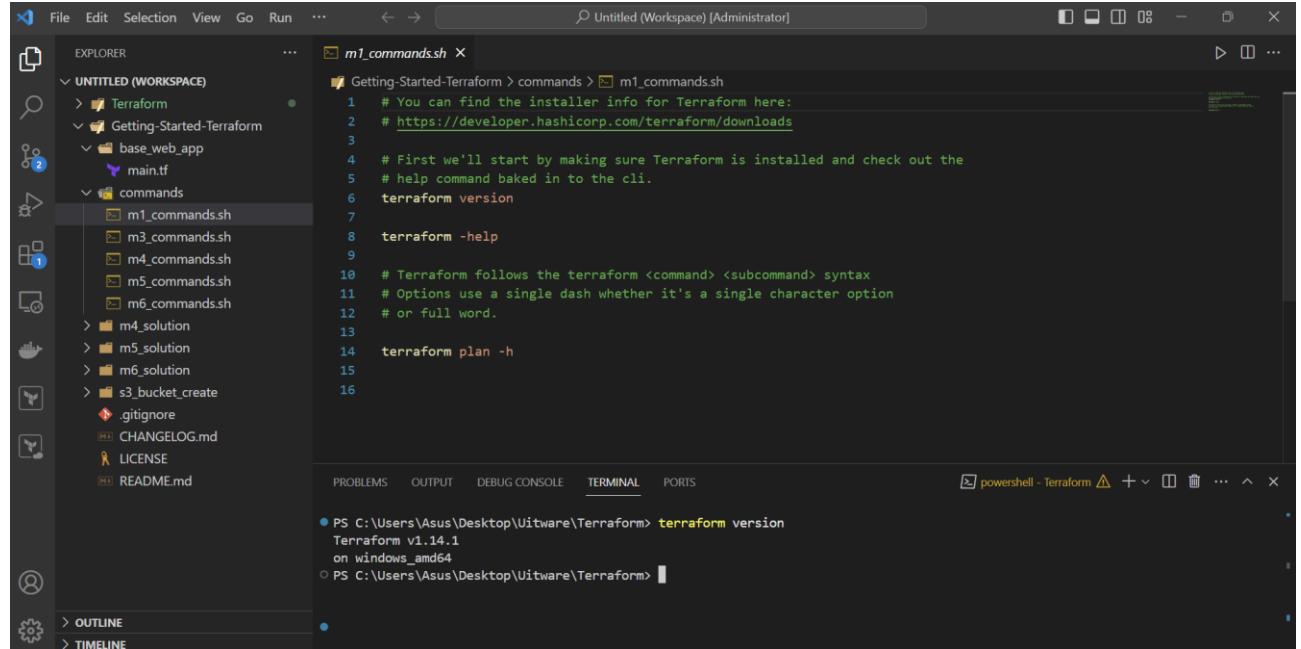
Installing the following packages:
terraform
By installing, you accept licenses for the packages.
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading terraform 1.14.1... 100%

terraform v1.14.1 [Approved]
terraform package files install completed. Performing other installation steps.
The package terraform wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N)o/[P]rint): Y

Removing old terraform plugins
```

```
Removing old terraform plugins
Downloading terraform 64 bit
  from 'https://releases.hashicorp.com/terraform/1.14.1/terraform_1.14.1_windows_amd64.zip'
Progress: 100% - Completed download of C:\Users\Asus\AppData\Local\Temp\chocolatey\terraform\1.14.1\terraform_1.14.1_windows_amd64.zip (29.78 MB).
Download of terraform_1.14.1_windows_amd64.zip (29.78 MB) completed.
Hashes match.
Extracting C:\Users\Asus\AppData\Local\Temp\chocolatey\terraform\1.14.1\terraform_1.14.1_windows_amd64.zip to C:\ProgramData\chocolatey\lib\terraform\tools...
C:\ProgramData\chocolatey\lib\terraform\tools
  ShimGen has successfully created a shim for terraform.exe
  The install of terraform was successful.
  Deployed to 'C:\ProgramData\chocolatey\lib\terraform\tools'
```

Cloned a repo for practice



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under "UNTITLED (WORKSPACE)". It includes a "Terraform" folder, a "Getting-Started-Terraform" folder containing "base_web_app" (with "main.tf") and "commands" (containing "m1_commands.sh", "m3_commands.sh", "m4_commands.sh", "m5_commands.sh", "m6_commands.sh"). Other files like "m4_solution", "m5_solution", "m6_solution", "s3_bucket_create", ".gitignore", "CHANGELOG.md", "LICENSE", and "README.md" are also listed.
- Terminal View:** Shows PowerShell output related to Terraform version:

```
PS C:\Users\Asus\Desktop\Uitware\Terraform> terraform version
Terraform v1.14.1
on windows_amd64

```

Terraform version

```

● PS C:\Users\Asus\Desktop\Uitware\Terraform> terraform version
Terraform v1.14.1
on windows_amd64
○ PS C:\Users\Asus\Desktop\Uitware\Terraform>

```

Running terraform init

Creating a folder and copying main file

```

#!/bin/sh
# In this module, we are simply trying to get the configuration deployed.
# First we'll copy our file from the base_web_app to a working directory
mkdir globo_web_app
cp ./base_web_app/main.tf ./globo_web_app/main.tf

# Now we can work with the main.tf file in globo_web_app
cd globo_web_app

# Now we can initialize the Terraform working directory.
# This will download the AWS provider and set up the backend.
terraform init

## We need AWS credentials to plan and apply the configuration.
# You can set them in the environment variables AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY
# or use the AWS CLI to configure them.
# If you have the AWS CLI installed, you can run:
aws configure

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powershell - Getting-Started-Terraform

Directory: C:\Users\Asus\Documents\Git repos\Getting-Started-Terraform

Mode LastWriteTime Length Name

Ln 4 Col 50 (70 selected) Spaces: 4 UTF-8 CRLF Shell Script Prettier

Running terraform init

```

provider "aws" {
  source = "hashicorp/aws"
  version = "~> 5.0"
}

# PROVIDERS

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powershell - globo_web_app

>>>

```

● PS C:\Users\Asus\Documents\Git repos\Getting-Started-Terraform\globo_web_app> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.100.0...
- Installed hashicorp/aws v5.100.0 (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.

```

```

    /*
1 This Terraform configuration sets up a basic web application on AWS using an EC2 instance running Nginx.
2 It includes the necessary networking components such as a VPC, subnet, internet gateway, and security group.
3 AWS credentials are required to apply this configuration and can be set using environment variables or the
4 AWS CLI.
5 */

6
7 terraform {
8   required_providers {
9     aws = {
10      source  = "hashicorp/aws"
11      version = "~> 5.0"
12    }
13  }
14 }
15
16 #####
17 # PROVIDERS

```

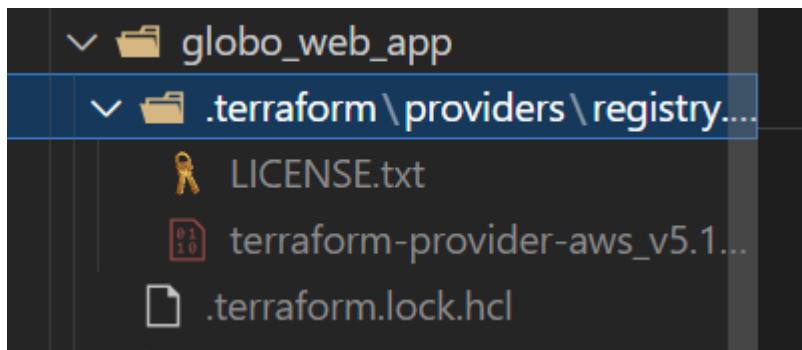
Terraform has been successfully initialized!

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.

PS C:\Users\Asus\Documents\Git repos\Getting-Started-Terraform\globo_web_app>

Now we have a file to talk to our provider



Initialization successful

Also created AWS account :)

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Creating access keys

Screenshot of the AWS IAM Users page showing no users.

The page title is "Identity and Access Management (IAM) > Users". The search bar shows "Search IAM". The main content area displays "Users (0)" with a note: "An IAM user is an identity with long-term credentials that is used to interact with AWS in an account." A search bar and a table header with columns "User name", "Path", "Groups", "Last activity", "MFA", and "Password age" are shown. The message "No resources to display" is centered below the table.

Screenshot of the "Create user" wizard Step 1: Specify user details.

The title is "Step 1 Specify user details". The sub-section title is "Specify user details". The "User name" field contains "dev-user". A note says: "The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = . @ _ - (hyphen)". An optional checkbox "Provide user access to the AWS Management Console" is checked. A note says: "In addition to console access, users with SigninLocalDevelopmentAccess permissions can use the same console credentials for programmatic access without the need for access keys." A callout box says: "If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keypairs, you can generate them after you create this IAM user. [Learn more](#)". Buttons at the bottom are "Cancel" and "Next".



User created

Screenshot of the AWS IAM Users page showing one user named "dev-user".

A green success message box says: "User created successfully. You can view and download the user's password and email instructions for signing in to the AWS Management Console. [View user](#)". The main content area shows "Users (1)" with a note: "An IAM user is an identity with long-term credentials that is used to interact with AWS in an account." A search bar and a table header with columns "User name", "Path", "Groups", "Last activity", "MFA", and "Password age" are shown. The table row for "dev-user" shows the "User name" as "dev-user", "Path" as "/", "Groups" as "0", "Last activity" as "-", "MFA" as "-", and "Password age" as "-".

Adding full access

Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

Copy permissions
Copy all group memberships, attached managed policies, inline policies, and any existing permissions boundaries from an existing user.

Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1434)

Filter by Type: s3

Policy name	Type	Attached entities
AmazonDMSRedshiftS3Role	AWS managed	0
AmazonS3FullAccess	AWS managed	0
AmazonS3ObjectLambda...	AWS managed	0
AmazonS3OutpostsFullAc...	AWS managed	0

Creating access key

1 policy added

Access key best practices & alternatives

Step 2 - optional

Set description tag

Step 3

Retrieve access keys

Access key best practices & alternatives

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.

Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

1 policy added

Step 1

Access key best practices & alternatives

Step 2 - optional

Set description tag

Step 3

Retrieve access keys

Set description tag - optional

The description for this access key will be attached to this user as a tag and shown alongside the access key.

Description tag value

Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

my-access-keys

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

Cancel Previous Create access key

Configured AWS for VScode

```

File Edit Selection View Go Run ...
Getting-Started-Terraform [Administrator]
EXPLORER
GETTING-STARTED-TERRAFORM
base_web_app
commands
m1_commands.sh
m3_commands.sh
m4_commands.sh
m5_commands.sh
m6_commands.sh
globo_web_app
m4_solution
m5_solution
m6_solution
s3_bucket_create
.gitignore
CHANGELOG.md
LICENSE
README.md
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell + □ □ □ ... ^ x
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform> aws configure
AWS Access Key ID [None]: AKIAVJYQ727FKUSWUZ
AWS Secret Access Key [None]: aHTBiNpQ/CytGBNi6Rijzh3n/QeKJR2TtpLhLSw7
Default region name [None]: eu-north-1
Default output format [None]:
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform> aws configure
AWS Access Key ID [*****USZ]: AKIAVJYQ727FKUSWUZ
AWS Secret Access Key [*****LSw7]: aHTBiNpQ/CytG8Ni6Rijzh3n/QeKJR2TtpLhLSw7
Default region name [eu-north-1]:
Default output format [None]: json
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform>
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Shell Script Prettier
main □ 0 □ 0 ✓ AWS: profiled dev-user

```

Added SSM permission because of blocking

Identity and Access Management (IAM)

1 policy added

Permissions Groups Roles Security credentials Last processed

Permissions policies (2)

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

Filter by Type

Policy name	Type	Attached via
AmazonS3FullAccess	AWS managed	Directly
AmazonSSMFullAccess	AWS managed	Directly

Permissions boundary (not set)

Created

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell - globo_web_app + □ □ □ ... ^ x
+ enable_dns_support = true
+ enable_network_address_usage_metrics = (known after apply)
+ id = (known after apply)
+ instance_tenancy = "default"
+ ipv6_association_id = (known after apply)
+ ipv6_cidr_block = (known after apply)
+ ipv6_cidr_block_network_border_group = (known after apply)
+ main_route_table_id = (known after apply)
+ owner_id = (known after apply)
+ tags_all = (known after apply)
}
Plan: 7 to add, 0 to change, 0 to destroy.

```

Applying our terraform plan

The screenshot shows the Visual Studio Code interface with the title bar "Getting-Started-Terraform [Administrator]". The Explorer sidebar on the left displays the project structure, including "base_web_app", "commands" (with files m1_commands.sh, m3_commands.sh, m4_commands.sh, m5_commands.sh, m6_commands.sh), "globo_web_app" (with files .tfplan, main.tf, m4_solution, m5_solution, m6_solution, s3_bucket_create, .gitignore, CHANGELOG.md, LICENSE, README.md), and AWS-related files (.tflock, .tfstate, .tfstate.backup). The main editor area shows the contents of "m3_commands.sh". The terminal tab at the bottom shows the command "terraform apply ./m3.tfplan" being run, resulting in the message "Plan: 7 to add, 0 to change, 0 to destroy." and "Saved the plan to: m3.tfplan". It also displays the command "terraform apply" being run again, with the message "Apply complete! Resources: 1 added, 0 changed, 0 destroyed." and the output of the AWS CLI showing the creation of an AWS instance.

```
main.tf
commands > m3_commands.sh
2 # First we'll copy our file from the base_web_app to a working directory
3 mkdir globo_web_app
4 cp ./base_web_app/main.tf ./globo_web_app/main.tf
5
6 # Now we can work with the main.tf file in globo_web_app
7 cd globo_web_app
8
9 # Now we can initialize the Terraform working directory.
10 # This will download the AWS provider and set up the backend.
11 terraform init
12
13 ## We need AWS credentials to plan and apply the configuration.
14 # You can set them in the environment variables AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY
15 # or use the AWS CLI to configure them

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell - globo_web_app + ... ^ x
+ ipv6_cidr_block_network_border_group = (known after apply)
+ main_route_table_id = (known after apply)
+ owner_id = (known after apply)
+ tags_all = (known after apply)

Plan: 7 to add, 0 to change, 0 to destroy.

Saved the plan to: m3.tfplan

To perform exactly these actions, run the following command to apply:
  terraform apply "m3.tfplan"
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform apply ./m3.tfplan

```

Applied, deployment completed

The screenshot shows the Visual Studio Code interface with the title bar "Getting-Started-Terraform [Administrator]". The Explorer sidebar on the left displays the project structure, including "base_web_app", "commands" (with files m1_commands.sh, m3_commands.sh, m4_commands.sh, m5_commands.sh, m6_commands.sh), "globo_web_app" (with files .tfplan, main.tf, m4_solution, m5_solution, m6_solution, s3_bucket_create, .gitignore, CHANGELOG.md, LICENSE, README.md), and AWS-related files (.tflock, .tfstate, .tfstate.backup). The main editor area shows the contents of "m3_commands.sh". The terminal tab at the bottom shows the command "terraform apply ./m3.tfplan" being run, resulting in the message "Plan: 1 to add, 0 to change, 0 to destroy." and "Saved the plan to: m3.tfplan". It also displays the command "terraform apply" being run again, with the message "Apply complete! Resources: 1 added, 0 changed, 0 destroyed." and the output of the AWS CLI showing the creation of an AWS instance.

```
main.tf
commands > m3_commands.sh
2 # First we'll copy our file from the base_web_app to a working directory
3 mkdir globo_web_app
4 cp ./base_web_app/main.tf ./globo_web_app/main.tf
5
6 # Now we can work with the main.tf file in globo_web_app
7 cd globo_web_app
8
9 # Now we can initialize the Terraform working directory.
10 # This will download the AWS provider and set up the backend.
11 terraform init
12
13 ## We need AWS credentials to plan and apply the configuration.
14 # You can set them in the environment variables AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY
15 # or use the AWS CLI to configure them

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell - globo_web_app + ... ^ x
Plan: 1 to add, 0 to change, 0 to destroy.

Saved the plan to: m3.tfplan

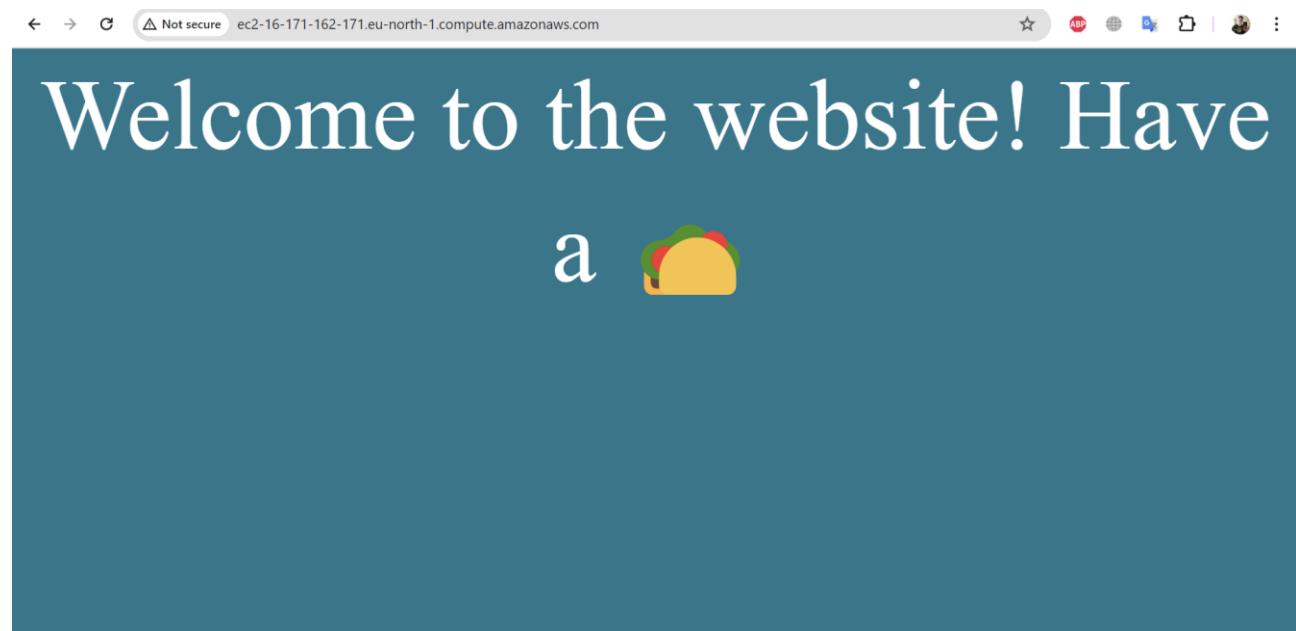
To perform exactly these actions, run the following command to apply:
  terraform apply "m3.tfplan"
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform apply ./m3.tfplan
aws_instance.nginx1: Creating...
aws_instance.nginx1: Still creating... [00m10s elapsed]
aws_instance.nginx1: Creation complete after 15s [id=i-0af789377fab08be1]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app>
```

Added

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like Dashboard, AWS Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, and Capacity Manager. Below that is a section for Images. At the bottom of the sidebar are links for CloudShell, Feedback, and Console Mobile App. The main content area has a header 'Instances (1/1) Info' with tabs for Connect, Instance state, Actions, and Launch instances. A search bar at the top says 'Find Instance by attribute or tag (case-sensitive)' with a dropdown for 'All states'. Below the search is a table with one row for an instance named 'i-0c7781cdcf841c056'. The instance is listed as 'Running' with an 't3.micro' instance type and '3/3 checks passed'. The 'Details' tab is selected, showing the Instance ID 'i-0c7781cdcf841c056', Public IPv4 address '16.171.162.171' with a link to 'open address', and Private IPv4 address '10.0.0.108'. Below the details are tabs for Status and alarms, Monitoring, Security, Networking, Storage, and Tags.

Website:



Destroyed

A screenshot of a code editor (Visual Studio Code) showing a Terraform workspace titled 'Getting-Started-Terraform [Administrator]'. The left sidebar shows a file tree with files like 'main.tf', 'terraform.tfstate', 'm3_commands.sh', 'LICENSE.txt', '.tflock.hcl', 'm3.tfplan', 'main.tf', 'terraform.tfstate', and 'terraform.tfstate.backup'. The main editor pane displays the 'main.tf' file with the following code:

```

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

# DATA

```

Below the code, the terminal tab shows the output of a 'terraform destroy' command:

```

aws_internet_gateway.app: Still destroying... [id=igw-072a95a0e1473dbbd, 00m30s elapsed]
aws_instance.nginx1: Still destroying... [id=i-0c7781cdcf841c056, 00m40s elapsed]
aws_internet_gateway.app: Still destroying... [id=igw-072a95a0e1473dbbd, 00m40s elapsed]
aws_instance.nginx1: Still destroying... [id=i-0c7781cdcf841c056, 00m50s elapsed]
aws_instance.nginx1: Destruction complete after 5s
aws_subnet.public_subneti: Destroying... [id=subnet-0c8f2a2232e28bf67]
aws_security_group.nginx_sg: Destroying... [id=sg-0c413a3b0182aaef27]
aws_subnet.public_subneti: Destruction complete after 0s
aws_security_group.nginx_sg: Destruction complete after 1s
aws_vpc.app: Destroying... [id=vpc-044e5d31c205bb426]
aws_vpc.app: Destruction complete after 1s

```

The status bar at the bottom indicates the file is 'AWS.profile-dev-user' and shows other system information like 'In 17. Col 12' and 'Prettier'.

The screenshot shows the AWS Management Console interface for the EC2 service. The left sidebar has a navigation tree with 'Instances' selected under 'EC2'. The main content area is titled 'Instances (1) Info' and shows a table with one row. The row contains the Instance ID 'i-0c7781cdcf841c056', the Instance state 'Terminated', the Instance type 't3.micro', and a 'View alarms +' button. A search bar at the top says 'Find Instance by attribute or tag (case-sensitive)'.

Lets use variables in terraform

Creating file variables.tf

The screenshot shows the VS Code code editor with the 'variables.tf' file open. The file contains the following code:

```
variable "region" {
```

The code editor interface includes a sidebar with file navigation, a central workspace with multiple tabs, and a bottom status bar showing file paths and terminal output.

Creating a variable for aws region

```

variable "aws_region" {
  type = string
  description = "AWS region to use for resources"
  default = "eu-north-1"
}

```

The screenshot shows the VS Code interface with the main.tf file open. The code editor displays the following Terraform configuration:

```

variable "aws_region" {
  type = string
  description = "AWS region to use for resources"
  default = "eu-north-1"
}

```

The terminal tab shows the output of a destroy command:

```

aws_internet_gateway.app: Still destroying... [id=igw-072a95a0e1473dbbd, 0m30s elapsed]
aws_instance.nginx1: Still destroying... [id=i-0c7781cdcf841c056, 0m40s elapsed]
aws_internet_gateway.app: Still destroying... [id=igw-072a95a0e1473dbbd, 0m40s elapsed]
aws_instance.nginx1: Still destroying... [id=i-0c7781cdcf841c056, 0m50s elapsed]
aws_instance.nginx1: Destruction complete after 51s
aws_subnet.public_subnet1: Destroying... [id=subnet-0c8f2a2232e28bf67]
aws_security_group.nginx_sg: Destroying... [id=sg-0c413a3b0182aaf27]
aws_subnet.public_subnet1: Destruction complete after 0s
aws_security_group.nginx_sg: Destruction complete after 1s
aws_vpc.app: Destroying... [id=vpc-044e5d31c205bb426]
aws_vpc.app: Destruction complete after 1s

```

The status bar at the bottom indicates "Destroy complete! Resources: 7 destroyed."

Replacing region with variable:

```

provider "aws" {
  region = var.aws_region
}

```

The screenshot shows the VS Code interface with the main.tf file open. The code editor displays the following Terraform configuration:

```

provider "aws" {
  region = var.aws_region
}

```

The terminal tab shows the output of a destroy command:

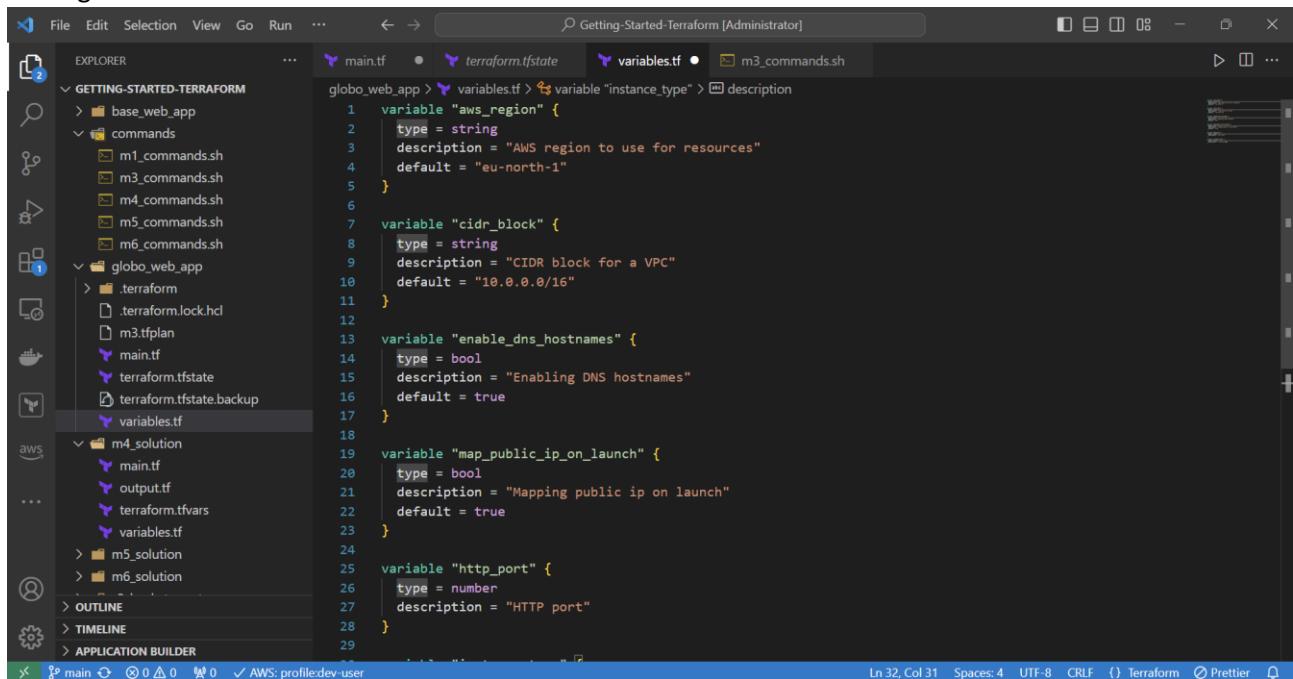
```

aws_internet_gateway.app: Still destroying... [id=igw-072a95a0e1473dbbd, 0m30s elapsed]
aws_instance.nginx1: Still destroying... [id=i-0c7781cdcf841c056, 0m40s elapsed]
aws_internet_gateway.app: Still destroying... [id=igw-072a95a0e1473dbbd, 0m40s elapsed]
aws_instance.nginx1: Still destroying... [id=i-0c7781cdcf841c056, 0m50s elapsed]
aws_instance.nginx1: Destruction complete after 51s
aws_subnet.public_subnet1: Destroying... [id=subnet-0c8f2a2232e28bf67]
aws_security_group.nginx_sg: Destroying... [id=sg-0c413a3b0182aaf27]
aws_subnet.public_subnet1: Destruction complete after 0s
aws_security_group.nginx_sg: Destruction complete after 1s
aws_vpc.app: Destroying... [id=vpc-044e5d31c205bb426]
aws_vpc.app: Destruction complete after 1s

```

The status bar at the bottom indicates "Destroy complete! Resources: 7 destroyed."

Adding variables to our main.tf file:



The screenshot shows the VS Code interface with the 'Getting-Started-Terraform' workspace open. The Explorer sidebar shows a tree structure of Terraform files and commands. The main editor tab displays the 'variables.tf' file, which contains the following code:

```
variable "aws_region" {
  type = string
  description = "AWS region to use for resources"
  default = "eu-north-1"
}

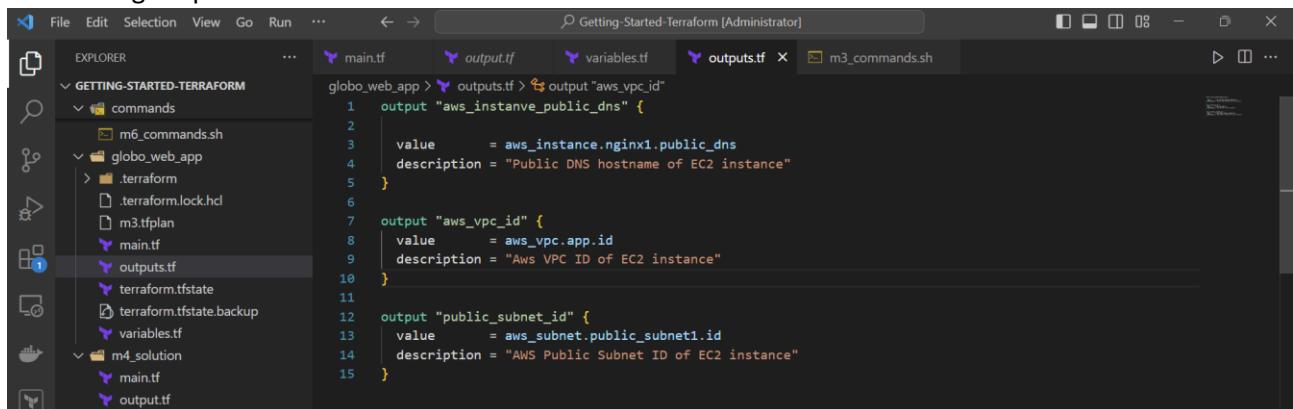
variable "cidr_block" {
  type = string
  description = "CIDR block for a VPC"
  default = "10.0.0.0/16"
}

variable "enable_dns_hostnames" {
  type = bool
  description = "Enabling DNS hostnames"
  default = true
}

variable "map_public_ip_on_launch" {
  type = bool
  description = "Mapping public ip on launch"
  default = true
}

variable "http_port" {
  type = number
  description = "HTTP port"
}
```

Also adding outputs:



The screenshot shows the VS Code interface with the 'Getting-Started-Terraform' workspace open. The Explorer sidebar shows a tree structure of Terraform files and commands. The main editor tab displays the 'outputs.tf' file, which contains the following code:

```
output "aws_instance_public_dns" {
  value     = aws_instance.nginx1.public_dns
  description = "Public DNS hostname of EC2 instance"
}

output "aws_vpc_id" {
  value     = aws_vpc.app.id
  description = "Aws VPC ID of EC2 instance"
}

output "public_subnet_id" {
  value     = aws_subnet.public_subnet1.id
  description = "AWS Public Subnet ID of EC2 instance"
}
```

Checking if formatting and syntax and logic is good

```
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform fmt
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform validate
Success! The configuration is valid.

PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app>
```

Applying new plan with outputs

```

main.tf
variables.tf
outputs.tf
terrafom.tvars
variables.tf m4_solution
m3_comma

```

```

1 http_port = 80
2 instance_type = "t3.micro"

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

aws instance.nginx1: Still creating... [00m10s elapsed]
aws_instance.nginx1: Creation complete after 13s [id=i-0e6913789893f0bef]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

aws_instancve_public_dns = "ec2-13-62-99-83.eu-north-1.compute.amazonaws.com"
aws_vpc_id = "vpc-047674d4ea6dc6969"
public_subnet_id = "subnet-00b6070f8bb6937c4"

PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app>

Adding 4 variables

```

main.tf
locals.tf
variables.tf
outputs.tf
terrafom.tvars
variables.tf m4_solution
m3_comma

```

```

variable "company_name" {
  type = string
  description = "Company name"
}

variable "project_name" {
  type = string
  description = "Project name"
}

variable "environment" {
  type = string
  description = "Environment"
}

variable "billing_code" {
  type = number
  description = "Billing code"
}

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

aws_instancve_public_dns = "ec2-13-62-99-83.eu-north-1.compute.amazonaws.com"
aws_vpc_id = "vpc-047674d4ea6dc6969"
public_subnet_id = "subnet-00b6070f8bb6937c4"

PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app>

Then add them as locals

EXPLORER

... main.tf

globo_web_app > locals.tf > variables.tf > output.tf > outputs.tf > terraform.tfvars > m3_commands.sh > ...

GETTING-STARTED-TERR... [+] ⏪ ⏴ ⏵ ⏷

commands

- m1_commands.sh
- m3_commands.sh
- m4_commands.sh
- m5_commands.sh
- m6_commands.sh

globo_web_app

- .terraform
- .terraform.lock.hcl

locals.tf

- m3.tfplan
- m4.tfplan
- main.tf
- outputs.tf
- terraform.tfstate
- terraform.tfstate.backup

variables.tf

aws

m4_solution

- main.tf
- output.tf
- terraform.tfvars

OUTLINE

TIMELINE

APPLICATION BUILDER

```
1 locals {
2     common_tags = {
3         Project = var.project
4         Company = var.company_name
5         Environment = var.environment
6         BillingCode = var.billing_code
7     }
8     prefix = "${var.project}-${var.environment}"
9 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powerShell - globo_web_app

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

```
aws_instance_public_dns = "ec2-13-62-99-83.eu-north-1.compute.amazonaws.com"
aws_vpc_id = "vpc-047574d4e6dc6969"
public_subnet_id = "subnet-00b6070f8bb6937c4"
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app
```

Also added tags to all blocks where I can add tags

```
main.tf globo_web_app x main.tf m4_solution locals.tf variables.tf outputs.tf terraform.tfvars
globo_web_app > main.tf > resource "aws_instance" "nginx1" [●] ami
34 ##### NETWORKING #####
35
36 # NETWORKING #
37 resource "aws_vpc" "app" {
38   cidr_block      = var.cidr_block
39   enable_dns_hostnames = var.enable_dns_hostnames
40
41   tags = local.common_tags
42 }
43
44 resource "aws_internet_gateway" "app" {
45   vpc_id = aws_vpc.app.id
46
47   tags = local.common_tags
48 }
49
50 resource "aws_subnet" "public_subnet1" {
51   cidr_block      = "10.0.0.0/24"
52   ...
53 }
54
55
aws_instance_public_dns = "ec2-13-62-99-83.eu-north-1.compute.amazonaws.com"
aws_vpc_id = "vpc-047674d4e6dc6969"
public_subnet_id = "subnet-00b6070f8bb6937c4"
● PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform console
Error: Unexpected "common_tags" block
  on locals.tf line 2, in locals:
  2:   common_tags {
```

The screenshot shows the Visual Studio Code interface with the title bar "Getting-Started-Terraform [Administrator]". The Explorer sidebar on the left lists files and folders related to a "globo_web_app" project, including "main.tf", "locals.tf", "variables.tf", and "outputs.tf". The main editor area displays the "main.tf" file with the following code:

```
49
50 resource "aws_subnet" "public_subnet1" {
51   cidr_block      = "10.0.0.0/24"
52   vpc_id          = aws_vpc.app.id
53   map_public_ip_on_launch = var.map_public_ip_on_launch
54
55   tags = local.common_tags
56 }
57
58 # ROUTING #
59 resource "aws_route_table" "app" {
60   vpc_id = aws_vpc.app.id
61
62   route {
63     cidr_block = "0.0.0.0/0"
64     gateway_id = aws_internet_gateway.app.id
65   }
66 }
```

A tooltip in the center of the screen displays the error message: "Error: Unexpected \"common_tags\" Від акумулятора on locals.tf line 2, in locals:"

At the bottom right, status bar information includes "Ln 102, Col 31", "Spaces:2", "UTF-8", "CRLF", "Terraform", and "Prettier".

This screenshot shows the same VS Code environment after the error has been resolved. The "main.tf" file now contains the corrected code:

```
59   resource "aws_route_table" "app" {
60     tags = local.common_tags
61   }
62
63   resource "aws_route_table_association" "app_subnet1" {
64     subnet_id    = aws_subnet.public_subnet1.id
65     route_table_id = aws_route_table.app.id
66   }
67
68   # SECURITY GROUPS #
69   # Nginx security group
70   resource "aws_security_group" "nginx_sg" {
71     name        = "nginx_sg"
72     vpc_id      = aws_vpc.app.id
73
74     # HTTP access from anywhere
75     ingress {
76       from_port    = var.http_port
77       to_port      = var.http_port
78       protocol    = "tcp"
79       cidr_blocks = ["0.0.0.0/0"]
80     }
81
82     # outbound internet access
83     egress {
84       from_port    = 0
85       to_port      = 0
86       protocol    = "tcp"
87     }
88
89   }
```

The status bar at the bottom right remains the same as in the previous screenshot.

```

resource "aws_instance" "nginx1" {
  ami                               = nonsensitive(data.aws_ssm_parameter.amzn2_linux.value)
  instance_type                     = var.instance_type
  subnet_id                         = aws_subnet.public_subnet1.id
  vpc_security_group_ids            = [aws_security_group.nginx_sg.id]
  user_data_replace_on_change       = true
  tags                             = local.common_tags
}

# INSTANCES #
resource "aws_instance" "nginxx1" {
  ami                               = nonsensitive(data.aws_ssm_parameter.amzn2_linux.value)
  instance_type                     = var.instance_type
  subnet_id                         = aws_subnet.public_subnet1.id
  vpc_security_group_ids            = [aws_security_group.nginx_sg.id]
  user_data_replace_on_change       = true
  tags                             = local.common_tags
}

```

In 102, Col 31 Spaces:2 UTF-8 CRLF {} Terraform ⚡ Prettier

Testing terraform console

```

PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform test

```

```

> min(4,2,18)
2
> lower("UITWARE")
"uitware"
> local.common_tags
(known after apply)
> exit

```

Ln 9, Col 2 Spaces:4 UTF-8 CRLF {} Terraform ⚡ Prettier

Adding a file startup_script.tpl with interpolation \${environment}

```

#!/bin/bash
sudo amazon-linux-extras install -y nginx1
sudo service nginx start
sudo rm /usr/share/nginx/html/index.html
sudo cat > /usr/share/nginx/html/index.html << 'WEBSITE'
<html>
<head>
<title>Taco Team Server - ${environment}</title>
</head>
<body style="background-color:#1F778D">
<p style="text-align: center;">
<span style="color:#FFFFFF;">
<span style="font-size:100px;">Welcome to the ${environment} website! Have a &#127790;</span>
</span>
</p>
</body>
</html>
WEBSITE

```

Ln 9, Col 2 Spaces:4 UTF-8 CRLF {} Terraform ⚡ Prettier

```

110     user_data = templatefile("./templates/startup_script.tpl")
111
112 }

```

Also added function merge to tags and tested in terraform console

```

resource "aws_vpc" "app" {
  cidr_block      = var.cidr_block
  enable_dns_hostnames = var.enable_dns_hostnames
  tags = merge(local.common_tags,{Name = "${local.prefix}-vpc"})
}

resource "aws_internet_gateway" "app" {
  vpc_id = aws_vpc.app.id
  tags = local.common_tags
}

resource "aws_subnet" "public_subnet1" {
  ...
}

```

Also added custom merge function with billing code in map

```

resource "aws_instance" "nginx1" {
  ami           = nonsensitive(data.aws_ssm_parameter.amzn2_linux.value)
  instance_type = var.instance_type
  subnet_id    = aws_subnet.public_subnet1.id
  vpc_security_group_ids = [aws_security_group.nginx_sg.id]
  user_data_replace_on_change = true

  tags = merge(local.common_tags,{Name = "${local.prefix}-nginx1", Billing = "${var.billing_code}"})

  user_data = templatefile("./templates/startup_script.tpl", {environment = var.environment})
}

```

Also added clickable DNS output

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Title Bar:** Getting-Started-Terraform [Administrator]
- Explorer:** Shows the project structure:
 - GETTING-STARTED-TERR... (includes files: m3_commands.sh, m4_commands.sh, m5_commands.sh, m6_commands.sh)
 - globo_web_app (includes files: .terraform, templates (with startup_script.tpl), locals.tf, m3.tftplan, m4.tftplan)
 - outputs.tf (selected, highlighted in blue)
 - main.tf
 - terraform.tfstate
 - terraform.tfstate.backup
- Editor:** The contents of the outputs.tf file are displayed:

```
1 output "aws_instance_public_dns" {
2     value      = "http://${aws_instance.nginx1.public_dns}:${var.http_port}"
3     description = "Public DNS hostname of EC2 instance"
4 }
5
6
7 output "aws_vpc_id" {
8     value      = aws_vpc.app_id
9     description = "Aws VPC ID of EC2 instance"
10 }
11
12 output "public_subnet_id" {
13     value      = aws_subnet.public_subnet1.id
14     description = "AWS Public Subnet ID of EC2 instance"
15 }
```
- Status Bar:** Shows the status bar with file paths and other information.

Formatting

```
● PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform fmt  
locals.tf  
main.tf  
terraform.tfvars  
variables.tf  
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app>
```

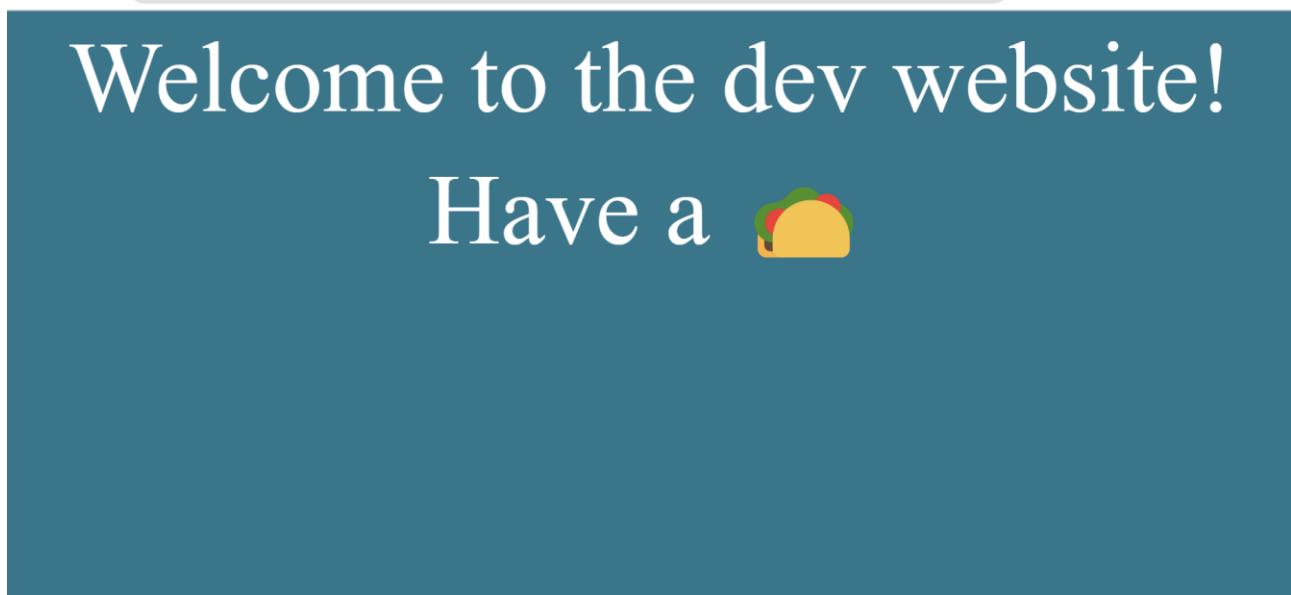
Applied with a clickable dns

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  powershell - globo_web_app +   ... ^ x

Apply complete! Resources: 1 added, 5 changed, 1 destroyed.

Outputs:

aws_instance_public_dns = "http://ec2-16-171-181-225.eu-north-1.compute.amazonaws.com:80"
aws_vpc_id = "vpc-047674d4ea6dc6969"
public_subnet_id = "subnet-00b6070f8bb6937c4"
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> 
```



Using terraform state inspection commands

File Edit Selection View Go Run ... ← → Getting-Started-Terraform [Administrator]

EXPLORER ... main.tf globo_web_app main.tf s3_bucket_create outputs.tf locals.tf terraform.tfvars startup_script.t...

GETTING-STARTED-TERR... [+] Type to search output.tf

m6_solution templates locals.tf main.tf output.tf terraform.tf variables.tf

s3_bucket_create [+] Type to search main.tf

main.tf s3_bucket_create outputs.tf main.tf

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powershell - globo_web_app + ...

```
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform show
# data.aws_ssm_parameter.amzn2_linux:
data "aws_ssm_parameter" "amzn2_linux" {
  arn      = "arn:aws:ssm:eu-north-1::parameter/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
  id       = "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
  insecure_value = "ami-00c4bcf1a0fe68ee2"
  name     = "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
  type     = "String"
  value    = (sensitive value)
  version  = 163
  with_decryption = true
}

# aws_instance.nginx1:
resource "aws_instance" "nginx1" {
  ami           = "ami-00c4bcf1a0fe68ee2"
  arn          = "arn:aws:ec2:eu-north-1:570729420030:instance/i-0840706ecf959f4cf"
  associate_public_ip_address = true
  availability_zone = "eu-north-1b"
  cpu_core_count = 1
  cpu_threads_per_core = 2
  disable_api_stop = false
  disable_api_termination = false
  ebs_optimized = false
  get_password_data = false
  hibernation = false
  host_id      = null
}
```

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF {} Terraform ⚙️ Prettier

▶ main ✘ 0 0 0 0 ✓ AWS: profile:dev-user

```
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform state list
data.aws_ssm_parameter.amzn2_linux
aws_instance.nginx1
aws_internet_gateway.app
aws_route_table.app
aws_route_table_association.app_subnet1
aws_security_group.nginx_sg
aws_subnet.public_subnet1
aws_vpc.app
○ PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app>
```

▶ PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform state show aws_instance.nginx1
aws_instance.nginx1:
resource "aws_instance" "nginx1" {
 ami = "ami-00c4bcf1a0fe68ee2"
 arn = "arn:aws:ec2:eu-north-1:570729420030:instance/i-0840706ecf959f4cf"
 associate_public_ip_address = true
 availability_zone = "eu-north-1b"
 cpu_core_count = 1
 cpu_threads_per_core = 2
 disable_api_stop = false
 disable_api_termination = false
 ebs_optimized = false
 get_password_data = false
 hibernation = false
 host_id = null
 iam_instance_profile = null
 id = "i-0840706ecf959f4cf"
 instance_initiated_shutdown_behavior = "stop"
 instance.lifecycle = null
 instance_state = "running"
 instance_type = "t3.micro"

▶ PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform output
aws_instanve_public_dns = "http://ec2-16-171-181-225.eu-north-1.compute.amazonaws.com:80"
aws_vpc_id = "vpc-047674d4ea6dc6969"
public_subnet_id = "subnet-00b6070f8bb6937c4"
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app>

Creating s3 bucket

```

● PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\s3_bucket_create> terraform init
  Initializing the backend...
  Initializing provider plugins...
    - Finding hashicorp/aws versions matching "~> 5.0"...
    - Installing hashicorp/aws v5.100.0...
    - Installed hashicorp/aws v5.100.0 (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.

```

Creating plan

```

● PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\s3_bucket_create> terraform plan -out s3.tfplan

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

Terraform will perform the following actions:

# aws_s3_bucket.taco_wagon will be created
+ resource "aws_s3_bucket" "taco_wagon" {
    + acceleration_status      = (known after apply)
    + acl                      = (known after apply)
    + arn                      = (known after apply)
    + bucket                   = (known after apply)
    + bucket_domain_name       = (known after apply)
    + bucket_prefix             = "taco-wagon"
    + bucketRegionalDomainName = (known after apply)
    + force_destroy              = true
    + hostedZoneId              = (known after apply)
    + id                        = (known after apply)
    + objectLockEnabled          = (known after apply)
    + policy                    = (known after apply)
    + region                    = (known after apply)
    + requestPayer               = (known after apply)
    + tags                      = {
        + "Environment" = "terraform-demo"
        + "Purpose"     = "State storage"
    }
    + tags_all                  = {
        + "Environment" = "terraform-demo"
        + "Purpose"     = "State storage"
    }
}
```

dev-user In 1 Col 1 Spaces:2 LITE-8 CRLE {} Terraform Prettier

Applying

```

● PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\s3_bucket_create> terraform apply ./s3.tfplan
aws_s3_bucket.taco_wagon: Creating...
aws_s3_bucket.taco_wagon: Creation complete after 3s [id=taco-wagon20251210225442285800000001]
aws_s3_bucket_public_access_block.taco_wagon_pab: Creating...
aws_s3_bucket_server_side_encryption_configuration.taco_wagon_encryption: Creating...
aws_s3_bucket_versioning.taco_wagon_versioning: Creating...
aws_s3_bucket_public_access_block.taco_wagon_pab: Creation complete after 1s [id=taco-wagon20251210225442285800000001]
aws_s3_bucket_server_side_encryption_configuration.taco_wagon_encryption: Creation complete after 1s [id=taco-wagon20251210225442285800000001]
aws_s3_bucket_versioning.taco_wagon_versioning: Creation complete after 2s [id=taco-wagon20251210225442285800000001]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

bucket_name = "taco-wagon20251210225442285800000001"
bucket_region = "us-east-1"

```

Creating s3 bucket for saving tfstate in cloud

```

1 terraform {
2   required_providers {
3     aws = {
4       source  = "hashicorp/aws"
5       version = "~> 5.0"
6     }
7   }
8   backend "s3" {
9     bucket = "taco-wagon20251210225442285800000001"
10    region = "eu-north-1"
11  } */
12 }

```

To see all of Terraform's top-level commands, run:
terraform -help

Reconfiguring bucket for region eu-north-1

```

1 terraform {
2   required_providers {
3     aws = {
4       source  = "hashicorp/aws"
5       version = "~> 5.0"
6     }
7   }
8   backend "s3" [
9     bucket = "my-terraform-state-yuri-20251211"
10    key    = "state/terraform.tfstate"
11    region = "eu-north-1"
12  ]
13 }

```

Creating plan

```

● PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform plan
data.aws_ssm_parameter.amzn2_linux: Reading...
aws_vpc.app: Refreshing state... [id=vpc-047674d4ea6dc6969]
data.aws_ssm_parameter.amzn2_linux: Read complete after 0s [id=/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2]
aws_internet_gateway.app: Refreshing state... [id=igw-063115e94197dd6ba]
aws_subnet.public_subnet1: Refreshing state... [id=subnet-00b6070f8bb6937c4]
aws_security_group.nginx_sg: Refreshing state... [id=sg-0a4b2bddf82245189]
aws_route_table.app: Refreshing state... [id=rtb-02f77884f28180c5d]
aws_route_table_association.app_subnet1: Refreshing state... [id=rtbassoc-052beacb04ae59d76]
aws_instance.nginx1: Refreshing state... [id=i-0840706ecf959f4cf]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
○ PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app>

```

Terraform access to state data is fine

Running terraform destroy

```
PS C:\Users\Asus\Documents\GitRepos\Getting-Started-Terraform\globo_web_app> terraform destroy
data.aws_ssm_parameter.amzn2_linux: Reading...
aws_vpc.app: Refreshing state... [id=vpc-047674d4ea6dc6969]
data.aws_ssm_parameter.amzn2_linux: Read complete after 1s [id=/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2]
aws_internet_gateway.app: Refreshing state... [id=igw-063115e94197dd6ba]
aws_subnet.public_subnet1: Refreshing state... [id=subnet-00b6070f8bb6937c4]
aws_security_group.nginx_sg: Refreshing state... [id=sg-0a4b2bddf82245189]
aws_route_table.app: Refreshing state... [id=rtb-02f77884f28180c5d]
aws_instance.nginx1: Refreshing state... [id=i-0840706ecf959f4cf]
aws_route_table_association.app_subnet1: Refreshing state... [id=rtbassoc-052beacb04ae59d76]

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.nginx1 will be destroyed
- resource "aws_instance" "nginx1" {
    - ami                      = "ami-00c4bcf1a0fe68ee2" -> null
    - arn                      = "arn:aws:ec2:eu-north-1:570729420030:instance/i-0840706ecf959f4cf" -> null
    - associate_public_ip_address = true -> null
    - availability_zone        = "eu-north-1b" -> null
    - cpu_core_count           = 1 -> null
    - cpu_threads_per_core     = 2 -> null
    - disable_api_stop          = false -> null
    - disable_api_termination   = false -> null
    - ebs_optimized             = false -> null
}

aws_route_table.app: Destroying... [id=rtb-02f77884f28180c5d]
aws_route_table.app: Destruction complete after 1s
aws_internet_gateway.app: Destroying... [id=igw-063115e94197dd6ba]
aws_instance.nginx1: Still destroying... [id=i-0840706ecf959f4cf, 00m10s elapsed]
aws_internet_gateway.app: Still destroying... [id=igw-063115e94197dd6ba, 00m10s elapsed]
aws_instance.nginx1: Still destroying... [id=i-0840706ecf959f4cf, 00m20s elapsed]
aws_internet_gateway.app: Still destroying... [id=igw-063115e94197dd6ba, 00m20s elapsed]
aws_instance.nginx1: Still destroying... [id=i-0840706ecf959f4cf, 00m30s elapsed]
aws_internet_gateway.app: Still destroying... [id=igw-063115e94197dd6ba, 00m30s elapsed]
aws_instance.nginx1: Still destroying... [id=i-0840706ecf959f4cf, 00m40s elapsed]
aws_internet_gateway.app: Still destroying... [id=igw-063115e94197dd6ba, 00m40s elapsed]
aws_instance.nginx1: Still destroying... [id=i-0840706ecf959f4cf, 00m50s elapsed]
aws_internet_gateway.app: Still destroying... [id=igw-063115e94197dd6ba, 00m50s elapsed]
aws_instance.nginx1: Still destroying... [id=i-0840706ecf959f4cf, 01m00s elapsed]
aws_internet_gateway.app: Still destroying... [id=igw-063115e94197dd6ba, 01m00s elapsed]
aws_internet_gateway.app: Destruction complete after 1m8s
aws_instance.nginx1: Still destroying... [id=i-0840706ecf959f4cf, 01m10s elapsed]
aws_instance.nginx1: Destruction complete after 1m12s
aws_subnet.public_subnet1: Destroying... [id=subnet-00b6070f8bb6937c4]
aws_security_group.nginx_sg: Destroying... [id=sg-0a4b2bddf82245189]
aws_subnet.public_subnet1: Destruction complete after 0s
aws_security_group.nginx_sg: Destruction complete after 0s
aws_vpc.app: Destroying... [id=vpc-047674d4ea6dc6969]
aws_vpc.app: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
```