

Hello readers,

In this article, I am eager to talk about how to provision and manage cloud resources using code. There are various tools which support “infrastructure-as-code”, such as [Chef](#), [Puppet](#), [Ansible](#), [CloudFormation](#), [SaltStack](#) etc, but , in this blog I want to talk about Terraform. . Terraform is an open-source IaC software tool created by Hashicorp that provides a consistent CLI to manage various cloud resources. Indeed, we will be using it, to create AWS services (VPC, Subnets, Security group , Ec2 with user-data to launch Apache server with a sample website on it , internet gateway etc..). We will go through GitHub by creating a local environment which will hold our code. We will then, create a new GitHub repository to access our code remotely.

We will learn how to:

- Create a directory in your local environment and initialize a new git repository
- Create a new repository in GitHub and push our code in it.
- Use Terraform to automate a fully managed code template which have working sample website running on Apache.

I will drop my public repository which have all the codes at the end of this article.

Before we get started, make sure you have these resources ready.

First download Terraform, use google or this link <https://www.terraform.io/downloads>, download visual studio code <https://code.visualstudio.com/download> and GitHub <https://git-scm.com/downloads>

Let start with GitHub

I- On your Desktop make right click to open gitbash .You will see something similar to this

```
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop (main)
$

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop (main)
$

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop (main)
$ |
```

2- Run `mkdir keita-terraform-project`

```
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop (main)
$ mkdir keita-terraform-project

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop (main)
$ |
```

3- Run `cd keita-terraform-project` in order to navigate the folder we just created.

```
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop (main)
$ cd keita-terraform-project/

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project (main)
$ |
```

4 Run `mkdir project-p01` to create another folder inside our main project.

```
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project (main)
$ cd project-p01/

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project/project-p01 (main)
$ |
```

As you could see, we created two folders which hold our project locally, we then have to have a remote control to access and host them on GitHub. Let then create an empty github repository.

If you don't have a github account , you could have one via this link <https://github.com/>


After creating your github account, create a new repository , yourname-terraform-p02

🔒 https://github.com/new

Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * **Repository name ***

 I2BKeit / keita-terraform-p02 ✓

Great repository names are short and memorable. Need inspiration? How about [laughing-parakeet?](#)

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)



.gitignore template: None ▼

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▼

Voila! We created an empty repository which look like this :


Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** `https://github.com/I2BKeit/keita-terraform-p02.git` 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).


...or create a new repository on the command line

```
echo "# keita-terraform-p02" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/I2BKeit/keita-terraform-p02.git
git push -u origin main
```



...or push an existing repository from the command line

```
git remote add origin https://github.com/I2BKeit/keita-terraform-p02.git
git branch -M main
git push -u origin main
```



So far, we do have our project in our local environment(you laptop), now let configure it , to be accessible via a remote control. Let go back to gitbah within our project directory and run these commands.

- `echo "# keita-terraform-p02" >> README.md`
- `git init`
- `git add README.md`
- `git commit -m"github repository has been configured and we could access our project remotly"`

- `git branch -M main`
- `git remote add origin https://github.com/I2BKeit/keita-terraform-p02.git`
- `git push -u origin main`

```

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project/pr
ject-p01 (main)
$ git commit -m"github repository has been configured and we could
access our project remotly"^C

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project/pr
ject-p01 (main)
$ git commit -m"github repository has been configured and we could
access our project remotly"
[main (root-commit) be8ef2e] github repository has been configured
and we could access our project remotly
1 file changed, 1 insertion(+)
create mode 100644 README.md

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project/pr
ject-p01 (main)
$ git branch
* main

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project/pr
ject-p01 (main)
$ git remote add origin https://github.com/I2BKeit/keita-terraform
-p02.git

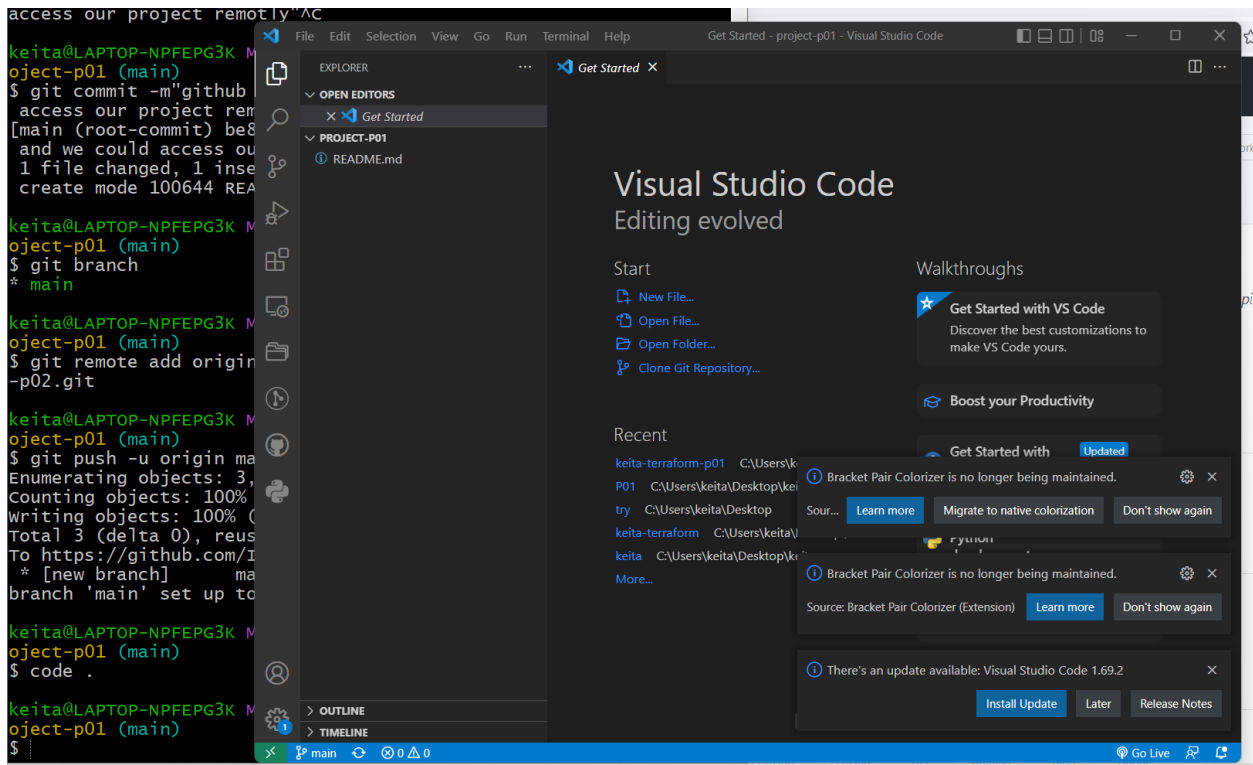
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project/pr
ject-p01 (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 277 bytes | 277.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/I2BKeit/keita-terraform-p02.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project/pr
ject-p01 (main)
$

```

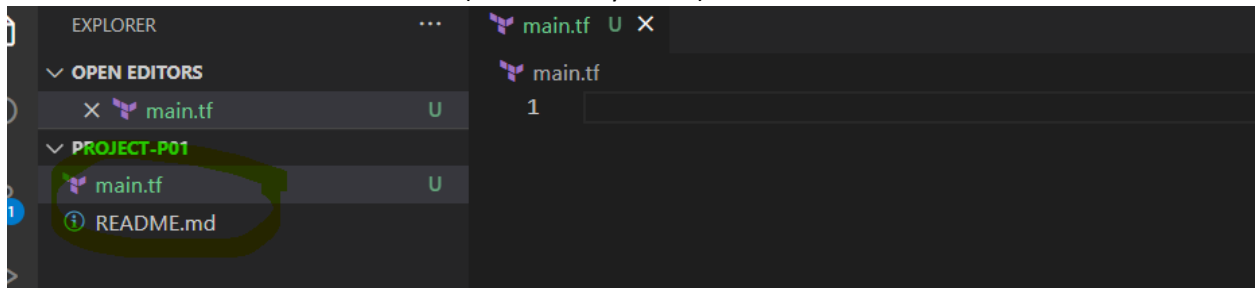
We now have our project pushed to github and it is a public repository anyone could access it anytime, anywhere.

We learnt how to create folders locally and access it remotely via github. Let now start working on our main subject: Terraform . Go back to gitbash and with our project directory run code `.` , which will open the project with visual code.

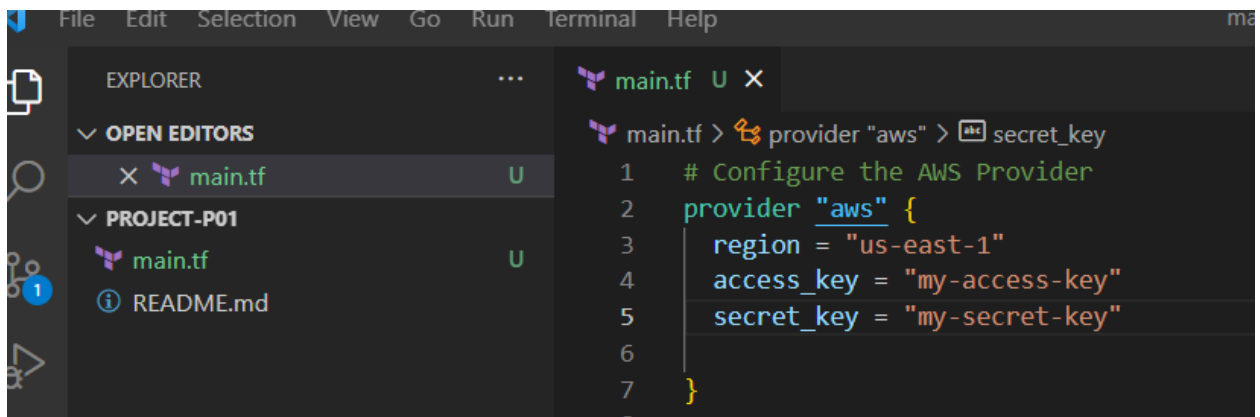


II Configure terraform main file.

- 1- Create a new file and name it main.tf (could be any name):



- 2- Enable communication between Terraform and AWS:



_Grab you access and secret key from IAM service under your security credentials.

Run terraform inti to initialize our project.

```
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project/pr
object-p01 (main)
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.22.0...
- Installed hashicorp/aws v4.22.0 (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 contro
l repository
so that Terraform can guarantee to make the same selections by def
ault when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform p
lan" to see
any changes that are required for your infrastructure. All Terrafo
rm commands
should now work.

If you ever set or change modules or backend configuration for Ter
raform,
rerun this command to reinitialize your working directory. If you
forget, other
commands will detect it and remind you to do so if necessary.

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project/pr
object-p01 (main)
$
```

Voila !! our project passed initialization let then implement our main.tf to create services

- 3- Automate AWS service via code as service.
 - a- Automate a VPC , name it keita-demo-VPC:

```
main.tf 6
README.md 7
8 resource "aws_vpc" "keita-demo-vpc" {
9     cidr_block = "10.0.0.0/16"
10     tags = {
11         "Name" = "keita-demo-vpc"
12     }
13 }
```

Run terraform plan to determine the desired state of above resource we just created. If everything works perfectly you will see something similar like this:

```
$ terraform plan

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_vpc.keita-demo-vpc will be created
+ resource "aws_vpc" "keita-demo-vpc" {
  + arn                               = (known after apply)
  + cidr_block                        = "10.0.0.0/16"
  + default_network_acl_id           = (known after apply)
  + default_route_table_id           = (known after apply)
  + default_security_group_id        = (known after apply)
  + dhcp_options_id                  = (known after apply)
  + enable_classiclink                = (known after apply)
  + enable_classiclink_dns_support    = (known after apply)
  + enable_dns_hostnames              = (known after apply)
  + enable_dns_support                = true
  + 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                             = {
    + "Name" = "keita-demo-vpc"
  }
  + tags_all                         = {
    + "Name" = "keita-demo-vpc"
  }
}

Plan: 1 to add, 0 to change, 0 to destroy.
```

Go ahead and launch terraform apply to automatically create our VPC. It will prompt your yes or No. Enter yes as value to launch. After running terraform apply , you should see something

similar like this:

```

+ default_route_table_id      = (known after apply)
+ default_security_group_id   = (known after apply)
+ dhcp_options_id             = (known after apply)
+ enable_classiclink           = (known after apply)
+ enable_classiclink_dns_support = (known after apply)
+ enable_dns_hostnames         = (known after apply)
+ enable_dns_support           = true
+ 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                         = {
+   + "Name" = "keita-demo-vpc"
+ }
+ tags_all                     = {
+   + "Name" = "keita-demo-vpc"
+ }
}

Plan: 1 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.

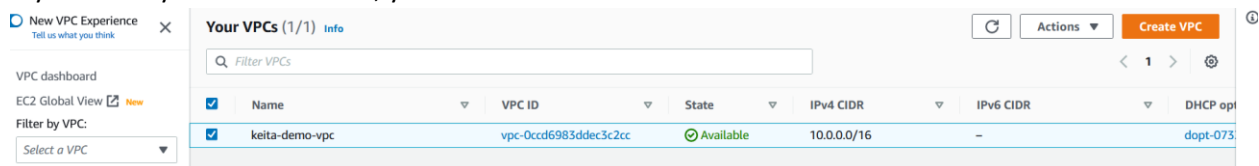
  Enter a value: yes

aws_vpc.keita-demo-vpc: Creating...
aws_vpc.keita-demo-vpc: Creation complete after 2s [id=vpc-0ccd6983ddec3c2cc]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

```

If you check your AWS console, you will see a new VPC created via terraform.



b- Create an Internet Gateway:

```

14 resource "aws_internet_gateway" "keita-gw" {
15     vpc_id = aws_vpc.keita-demo-vpc.id
16
17     tags = {
18         Name = "keita-gw"
19     }
20 }

```

c- Create Route Table

```

resource "aws_route_table" "keita-route-table" {
    vpc_id = aws_vpc.keita-demo-vpc.id

    route{
        cidr_block = "0.0.0.0/0"
        gateway_id = aws_internet_gateway.keita-gw.id
    }

    tags = {
        Name = "keita-route-table"
    }
}

```

d-

New VPC Experience
Tell us what you think

VPC dashboard

EC2 Global View 🔗 New

Filter by VPC:
Select a VPC

Route tables (1/2) info

Filter route tables

	Name	Route table ID	Explicit subnet associat...	Edge associations	Main	VPC	Ow...
<input type="checkbox"/>	-	rtb-0f6a4ae6e29bce1	-	-	Yes	vpc-0ccd6983ddec3c2cc keit...	32035..
<input checked="" type="checkbox"/>	keita-route-table	rtb-0a0ef02b5693d249	-	-	No	vpc-0ccd6983ddec3c2cc keit...	32035..

Create route table

e- Create two public Subnet:

```


resource "aws_subnet" "keita-public-subnet-A" {
    vpc_id = aws_vpc.keita-demo-vpc.id
    cidr_block = "10.0.0.0/24"
    availability_zone = "us-east-1a"
    tags = {
        "Name" = "keita-public-subnet-A"
    }
}

resource "aws_subnet" "keita-public-subnet-B" {
    vpc_id = aws_vpc.keita-demo-vpc.id
    cidr_block = "10.0.4.0/24"
    availability_zone = "us-east-1a"
    tags = {
        "Name" = "keita-public-subnet-B"
    }
}

```

Tell us what you think

VPC dashboard

EC2 Global View 

Filter by VPC:

Select a VPC

Virtual private cloud

Your VPCs

	Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR
<input type="checkbox"/>	keita-public-subnet-A	subnet-0d464187be4ca58a3	Available	vpc-0cccd6983ddec3c2cc keit...	10.0.0.0/24	-
<input type="checkbox"/>	keita-public-subnet-B	subnet-029ea49e7690f7b69	Available	vpc-0cccd6983ddec3c2cc keit...	10.0.4.0/24	-

f- Associate subnet with route table

```
# 5- Associate route to the public subnet
resource "aws_route_table_association" "Keita-public-RT-A" {
  subnet_id      = aws_subnet.keita-public-subnet-A.id
  route_table_id = aws_route_table.keita-route-table.id
}

resource "aws_route_table_association" "Keita-public-RT-B" {
  subnet_id      = aws_subnet.keita-public-subnet-B.id
  route_table_id = aws_route_table.keita-route-table.id
}
```

Edit subnet associations

Change which subnets are associated with this route table.

Available subnets (2/2)

Filter subnet associations

<input checked="" type="checkbox"/>	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
<input checked="" type="checkbox"/>	keita-public-subnet-A	subnet-0d464187be4ca58a3	10.0.0.0/24	-	rtb-0a0ef02b5693d2ff9 / keita-route-table
<input checked="" type="checkbox"/>	keita-public-subnet-B	subnet-029ea49e7690f7b69	10.0.4.0/24	-	rtb-0a0ef02b5693d2ff9 / keita-route-table

Selected subnets

subnet-0d464187be4ca58a3 / keita-public-subnet-A X

subnet-029ea49e7690f7b69 / keita-public-subnet-B X

g- Create security group

```
# 6- Security group which allows pport: 22, 80, 443
resource "aws_security_group" "keita-terraform-demo" {
  name           = "keita-terraform-demo"
  description    = "Allow TLS inbound traffic"
  vpc_id        = aws_vpc.keita-demo-vpc.id

  ingress {
    description = "HTTPS"
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"] // Anyone can access this
  }

  ingress {
    description = "HTTP"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"] // Anyone can access this
  }

  ingress {
    description = "SSH"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"] // Anyone can access this
  }

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

  tags = {
    Name = "keita-terraform-demo"
  }
}
```

gateways
Carrier gateways
DHCP Option Sets
Elastic IPs
Managed prefix lists
Endpoints

Security Groups (1/2) info						
Filter security groups						
Name	Security group ID	Security group name	VPC ID	Description	Owner	
<input checked="" type="checkbox"/> keita-terraform-demo	sg-07acae3a0451438c0	keita-terraform-demo	vpc-0cc6f983ddec3c2cc	Allow TLS inbound traf...	320352301616	

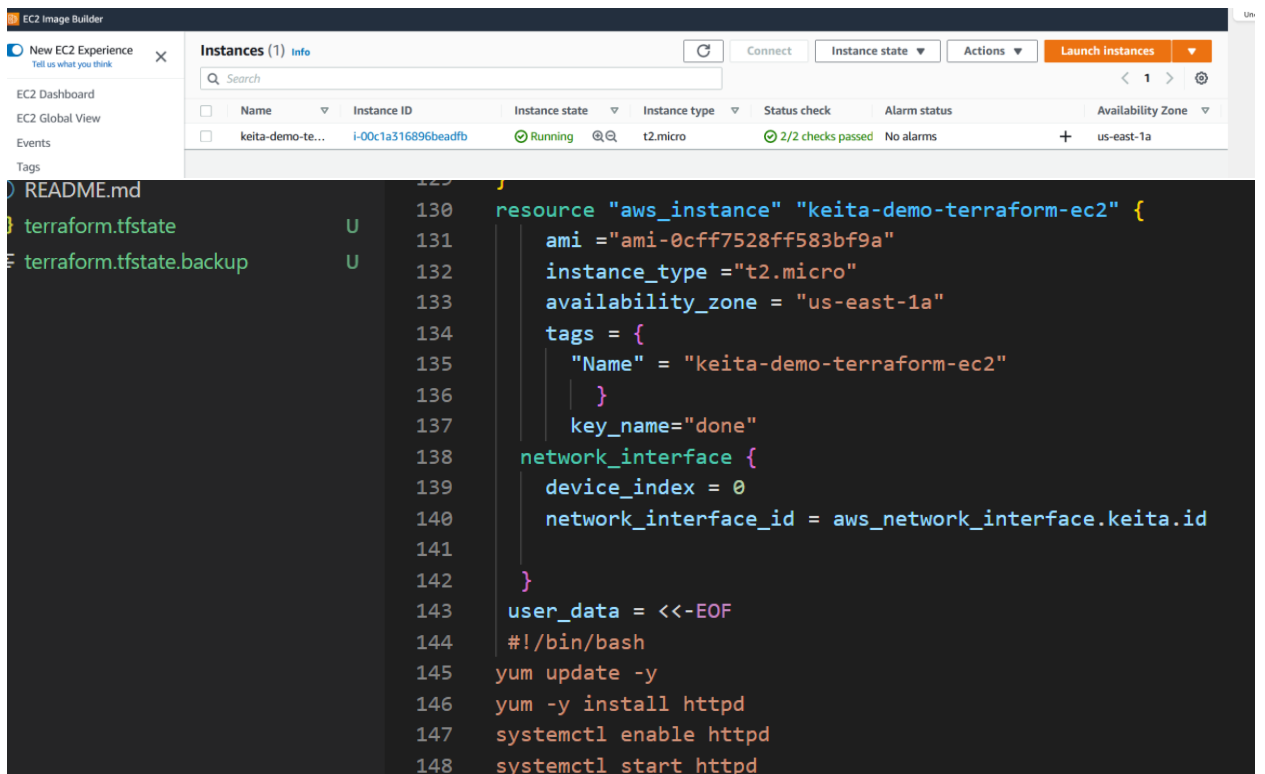
h- Create a network interface

```
# 7- Network interface
resource "aws_network_interface" "test" {
  subnet_id      = aws_subnet.keita-public-subnet-A
  private_ips    = ["10.0.0.50"]
  security_groups = aws-security_group.keita-terraform-demo
}
}
```

- i- Assign an elastic IP in the subnet that was created

```
resource "aws_eip" "keita-eip" {
  vpc                = true
  network_interface  = aws_network_interface.keita.id
  associate_with_private_ip = "10.0.0.50"
  depends_on = [
    aws_internet_gateway.keita-gw
  ]
}
```

- j- Create an ec2 instance and install Apache :



The screenshot displays the AWS Management Console's EC2 Instance Builder. The 'Instances (1)' table lists the following instance:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
keita-demo-te...	i-00c1a316896beadfb	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a

Below the console, the Terraform configuration for the EC2 instance is shown:

```
129 }
130 resource "aws_instance" "keita-demo-terraform-ec2" {
131     ami = "ami-0cff7528ff583bf9a"
132     instance_type = "t2.micro"
133     availability_zone = "us-east-1a"
134     tags = {
135         "Name" = "keita-demo-terraform-ec2"
136     }
137     key_name = "done"
138     network_interface {
139         device_index = 0
140         network_interface_id = aws_network_interface.keita.id
141     }
142 }
143 user_data = <<-EOF
144 #!/bin/bash
145 yum update -y
146 yum -y install httpd
147 systemctl enable httpd
148 systemctl start httpd
```

main.tf	U	149	echo '<html>
PROJECT-P01		150	<body>
.terraform	•	151	
.terraform.lock.hcl	U	152	<h2>Thank you dear readers </h2>
main.tf	U	153	<details>
README.md		154	<summary>Terraform commands used </summary>
terraform.tfstate	U	155	
terraform.tfstate.backup	U	156	Terraform init
		157	Terraform plan
		158	Terraform apply
		159	Terraform destroy
		160	
		161	</details>
		162	<details>
		163	<summary>Git commands used:</summary>
		164	
		165	git init
		166	git push
		167	git add .
		168	git commit -m"your messages"
		169	git remote add origin url
		170	
		171	
		172	</details>
		173	<p>Ibrehima keita </p>
		174	</body>
OUTLINE		175	</html>' > /var/www/html/index.html
MELINE		176	EOF
TERRAFORM PROVIDERS		177	}

Plan: 1 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.keita-demo-terraform-ec2: Creating...

aws_instance.keita-demo-terraform-ec2: Still creating... [10s elapsed]

aws_instance.keita-demo-terraform-ec2: still creating... [20s elapsed]

aws_instance.keita-demo-terraform-ec2: still creating... [30s elapsed]

aws_instance.keita-demo-terraform-ec2: Creation complete after 33s [id=i-00c1a316896beadfb]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project/project-p01 (main)

\$

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public
keita-demo-terraform-ec2	i-0d3572eeac8b7cb99	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	-

←	→	↺	🔒	50.17.236.28
---	---	---	---	--------------

Thank you dear readers

▼ Terraform commands used

-

Terraform init

Terraform plan

Terraform apply

Terraform destroy

▼ Git commands used:

-

git init

git push

git add .

git commit -m "your messages"

git remote add origin url

Ibrehima keita

EXPLORER

OPEN EDITORS

main.tf

PROJECT-P01

terraform

gitattributes

terraform.lock.hcl

main.tf

README.md

terraform.tfstate

terraform.tfstate.backup

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

GITLENS

JUPYTER

```

aws_internet_gateway.keita-gw: Still destroying... [id=igw-0381469ccea4473c4, 10s elapsed]
aws_internet_gateway.keita-gw: Destruction complete after 13s
aws_instance.keita-demo-terraform-ec2: Still destroying... [id=i-0d3572eeac8b7cb99, 20s elapsed]
aws_instance.keita-demo-terraform-ec2: Still destroying... [id=i-0d3572eeac8b7cb99, 30s elapsed]
aws_instance.keita-demo-terraform-ec2: Destruction complete after 30s
aws_network_interface.keita: Destroying... [id=eni-0e6eabd5980605c12]
aws_network_interface.keita: Destruction complete after 1s
aws_subnet.keita-public-subnet-A: Destroying... [id=subnet-05267086fc47ed1e7]
aws_security_group.keita-terraform-demo: Destroying... [id=sg-0774334f8d4a66eaf]
aws_security_group.keita-terraform-demo: Destruction complete after 0s
aws_subnet.keita-public-subnet-A: Destruction complete after 0s
aws_vpc.keita-demo-vpc: Destroying... [id=vpc-044f927ac74ccc0c8]
aws_vpc.keita-demo-vpc: Destruction complete after 0s

Destroy complete! Resources: 11 destroyed.
PS C:\Users\keita\Desktop\keita-terraform-project\project-p01>

```

```
keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project (main)
$ date
Wed Jul 20 04:13:11 EDT 2022

keita@LAPTOP-NPFEPG3K MINGW32 ~/Desktop/keita-terraform-project (main)
$ |
```