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](https://www.chef.io/), [Puppet](https://puppet.com/), [Ansible](https://go.redhat.com/delivery-with-ansible-20181012?sc_cid=701f2000000tvWZAAY&gclid=EAIaIQobChMI4_CHlP7G4AIVkKqWCh0PKASVEAAYASAAEgIzmvD_BwE&gclsrc=aw.ds), [CloudFormation](https://aws.amazon.com/cloudformation/?sc_channel=PS&sc_campaign=acquisition_IN&sc_publisher=google&sc_medium=ACQ-P%7CPS-GO%7CBrand%7CDesktop%7CSU%7CManagement%20Tools%7CCloudFormation%7CIN%7CEN%7CText&sc_content=cloudformation_e&sc_detail=cloudformation&sc_category=Management%20Tools&sc_segment=293640258122&sc_matchtype=e&sc_country=IN&s_kwcid=AL!4422!3!293640258122!e!!g!!cloudformation&ef_id=EAIaIQobChMIpc2qnf7G4AIVBqyWCh07wwkwEAAYASAAEgIaRPD_BwE:G:s), [SaltStack](https://www.saltstack.com/) 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/downloaill d](https://code.visualstudio.com/downloaill%20d) 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

Text

Description automatically generated

2- Run mkdir keita-terraform-project

Text

Description automatically generated

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

Text

Description automatically generated with low confidence

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

Text

Description automatically generated

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, crame it eate a new repository , yourname-terraform-p02

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

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.

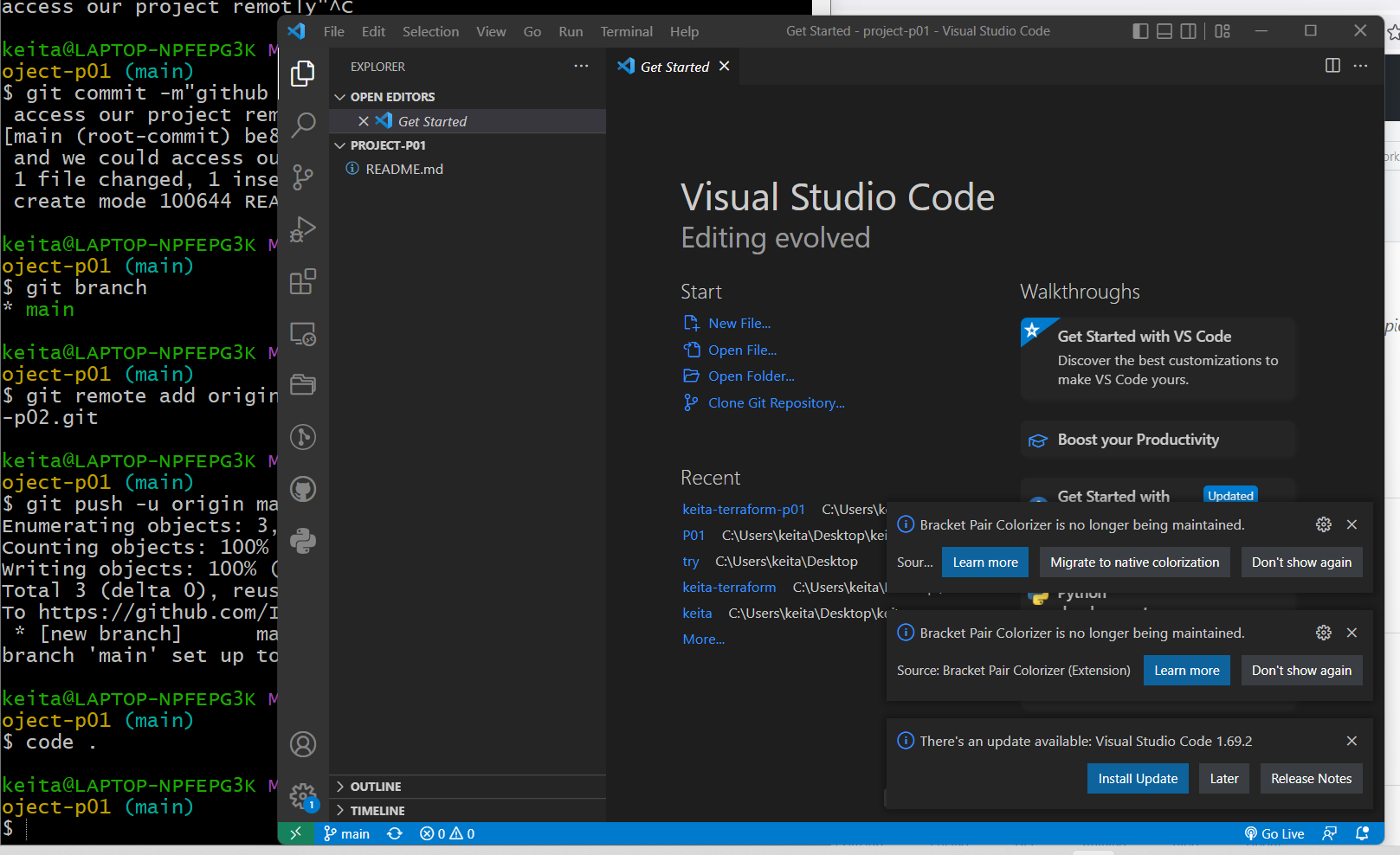
* echo "# keita-terraform-p02" >> README.md
* git init
* git add README.md
* git commit -m"github repository has been configured and we could access our project remotly"
* git branch -M main
* git remote add origin <https://github.com/I2BKeit/keita-terraform-p02.git>
* git push -u origin main

Text

Description automatically generated

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):

**Graphical user interface, text, application

Description automatically generated**

1. Enable communication between Terraform and AWS:

A screenshot of a computer

Description automatically generated with medium confidence

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

Run terraform inti to initialize our project.

Text

Description automatically generated

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

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

Text

Description automatically generated

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:

Text

Description automatically generated

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: Text

Description automatically generated

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

Graphical user interface

Description automatically generated with low confidence

1. Create an Internet Gateway:

Text

Description automatically generated

1. Create Route Table
2. Text

   Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Create two public Subnet:

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

1. Associate subnet with route table

Text

Description automatically generated

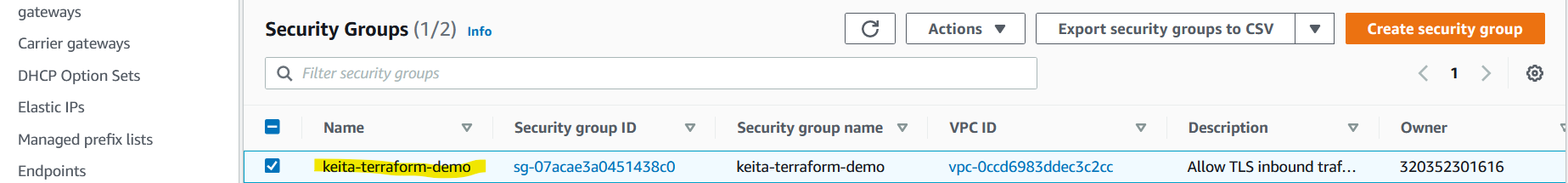
Graphical user interface, text, application, email

Description automatically generated

1. Create security group

Text

Description automatically generated



1. Create a network interface

Text

Description automatically generated

1. Assign an elastic IP in the subnet that was created

Text

Description automatically generated

1. Create an ec2 instance and install Apache :

Graphical user interface, application

Description automatically generated

Text

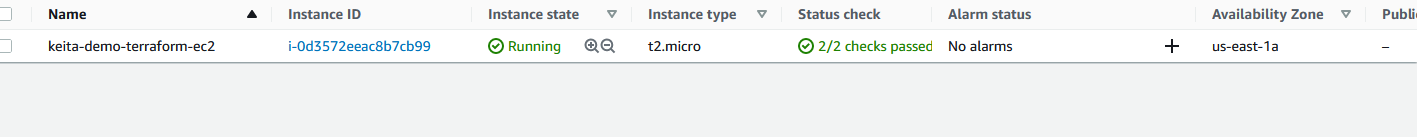
Description automatically generated

Text

Description automatically generated

Text

Description automatically generated



Graphical user interface, text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Text

Description automatically generated