Public Registry terraform-github

AWS console uzerinden Terraform makineye connect ile baglandik . ssh I koplyaladik ve vscode gittiik burda remote baglanti ssh balanti yerine yapistirdik ve bu makineye uzak baglanti ile baglandik.

Terraform makine icinde “terraform-module”adinda klasor olusuturk “bu klasor altinda main.tf -version.tf -variable.tf-output.tf-userdata.sh -.gitignore ve README dosyalarini olusturduk.

Main.tf

data "aws\_ami" "amazon-linux-2" { ##data kullanrak mevcut olan kaynklari okurhangi kaynaklari kullancaksak onu yazariz buraya

owners = ["amazon"] ##kuullanicagimiz ami nin sahibidir yada baska yada kendi olusutursugumuz ami buraya yazabiliriz

most\_recent = true ## en guncel ami cekilecektir

filter { ##filrte adi verilir tipi ebs secilir

name = "root-device-type"

values = ["ebs"]

}

filter { ## filter olarak hvm sanallastirma olarak hvm kullanilicagi filtrenelir

name = "virtualization-type"

values = ["hvm"]

}

filter { ##64 bitlik makine secmis oluyoruz bu yazilmazsa en son cikan 256 lik en guncel makine uygulanacaktir.

name = "architecture"

values = ["x86\_64"]

}

filter { ##owner olarak amazon secilmisti owner-alias ile takma adlar ile bizim ami ler cikabilir

name = "owner-alias"

values = ["amazon"]

}

filter {

name = "name"

values = ["amzn2-ami-kernel-5.10-hvm\*"] ##bu kod blogu asil onemli olan amazon2 ami kernel 5 secilmesi yani secilecek iso dosyasir.

} ##bu tanimli filtreler ani owner-alias degistilemez tanimli filtreler kullnailarak is kolaylasir.Filtre istenirse kullanilmayabilir.

} ##amazon ec2 consolden makine yaga kaldirilirken bu filtreler zaten kar;imiz ciker

data "template\_file" "userdata" { ##userdata adresi verilmis olunur.userdata local addir

template = file("${abspath(path.module)}/userdata.sh") ## file fonksiyonu bir dosyanin icerigini okumak icin kullanilir yani burda userdata icerigini okumaya yararr/burdaki path.module modulun adresini cagitmis olur/module yolu degisebilir absulete o path yolunu degismeyen sabit hale getirmek icin kullanilirbu daresten userdatayi almis olur

vars = {

server-name = var.server-name

}

}

resource "aws\_instance" "tfmyec2" { ##burdaki tfmyec2 ismi degistirilebilir.

ami = data.aws\_ami.amazon-linux-2.id ##yukarida olusutulan data bilgisinden cekilir

instance\_type = var.instance\_type ## var ile varible dosyasibdan degiskenler veriler islenmis olunur

count = var.num\_of\_instance

key\_name = var.key\_name

vpc\_security\_group\_ids = [aws\_security\_group.tf-sec-gr.id]

user\_data = data.template\_file.userdata.rendered ##template-file gibi script dosyalarinin cagrilip okunamasi icin rendered kullanilir ornegin ami icin id kullnilarak ami dosyaari cagrilmisti.

tags = {

Name = var.tag

}

}

resource "aws\_security\_group" "tf-sec-gr" { ##security group olustururken aws consolde bir tag var bir de name var onlari buraya giriyoruz

name = "${var.tag}-terraform-sec-grp" ##variable dosyasindan tag alacak ve terraform-sec-grp bunu da ekleyecek

tags = {

Name = var.tag

}

dynamic "ingress" { ##tek tek butun portlari asagidaki content icerisindeki isleri donduracektir

for\_each = var.docker-instance-ports ##burda variable dosyasinda 3 adet port var her biri icin ayri ayri ingress yani inbi=ound olusturucak

iterator = port

content {

from\_port = port.value

to\_port = port.value

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

}

egress { ##engress yani output

from\_port =0

protocol = "-1" #burdaki "-1" butun portlara acik demekir

to\_port =0

cidr\_blocks = ["0.0.0.0/0"]

}

}

Variable.tf

variable "instance\_type" {

type = string

default = "t2.micro"

}

variable "key\_name" { ##key name ozeldir buraya girilimesi gerekir herkes kendir keyini.

type = string

}

variable "num\_of\_instance" { # ayaga kaldirilan instance sayisi yazilir

type = number

default = 1

}

variable "tag" {

type = string

default = "Docker-Instance"

}

variable "server-name" {

type = string

default = "docker-instance"

}

variable "docker-instance-ports" {

type = list(number)

description = "docker-instance-sec-gr-inbound-rules"

default = [22, 80, 8080] ##security group olusturuldugunda burdan cekecek hangi portlsrin acik olmasini istiyorsak burda belirtip maintf te kullanilacak

}

Version.tf

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "5.23.1"

}

}

}

Bunlari terraform provider siteside USE PROVIDER kismindan aliyoruz. <https://registry.terraform.io/providers/hashicorp/aws/latest>

userdata.sh

#!/bin/bash

hostnamectl set-hostname ${server-name}     ##makine hostname  olarak server-name verilir

yum update -y

amazon-linux-extras install docker -y    ##docker kurulumu yapiliyor

systemctl start docker

systemctl enable docker      # makine kapatilip acildiginda icerinde hazir bir sekilde docker kurulu olacaktir

usermod -a -G docker ec2-user    #usermod ile yetkiler docker ec2-user grubuna eklenmis olur cunku docker in bazi yetkilere ihtiyaci var/

# install docker-compose

curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" \              ##docker compose u burdan indirip docker kurulumunu yapiyor

-o /usr/local/bin/docker-compose

chmod +x /usr/local/bin/docker-compose     #makinye yetki veriliyor chmod ile

.gitignore

# Terraform lockfile          ##.gitignore dosyasinda hassa verile bulunur bunlari saklmak gitignore dosyasi kullanilir.

.terraform.lock.hcl

# .tfstate files

\*.tfstate

\*.tfstate.\*

# Crash log files

crash.log

# Exclude all .tfvars files, which are likely to contain sentitive data, such as

# password, private keys, and other secrets. These should not be part of version

# control as they are data points which are potentially sensitive and subject

# to change depending on the environment.

\*.tfvars

output.tf

output "instance\_public\_ip" {

value = aws\_instance.tfmyec2.\*.public\_ip ## "\*" butun instancelerin public ip lerini dondurcektir

}

output "sec\_gr\_id" {

value = aws\_security\_group.tf-sec-gr.id

}

output "instance\_id" {

value = aws\_instance.tfmyec2.\*.id

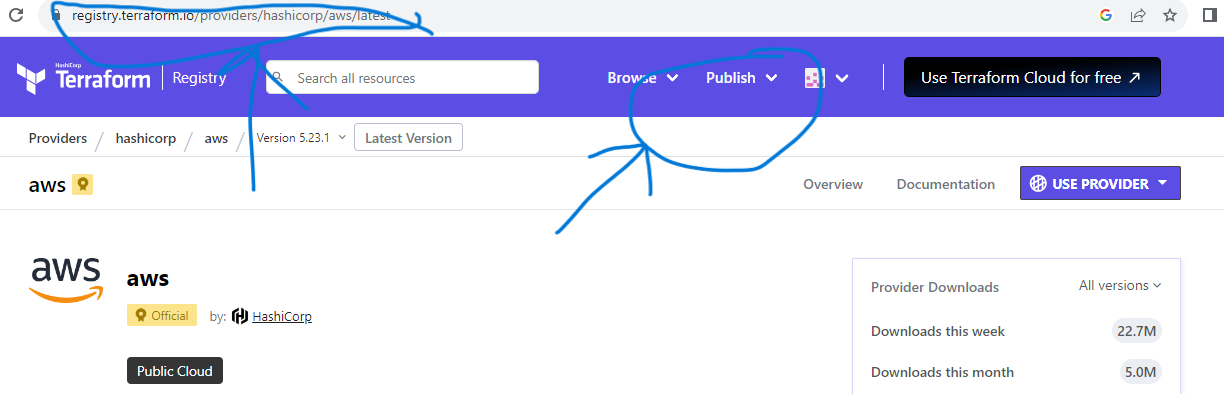
}

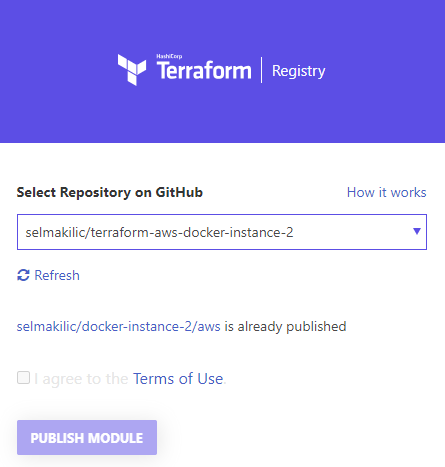
Bu dosyalari olusturdak sonra github repomuza giderzi ve “terraform-aws” bas kisminda olacak sekilde bir public repository olusururuz. Tam ismini biz terraform makine docker kurucagimiz icin repo adini “terraform-aws-docker-instance” diye oluturduk

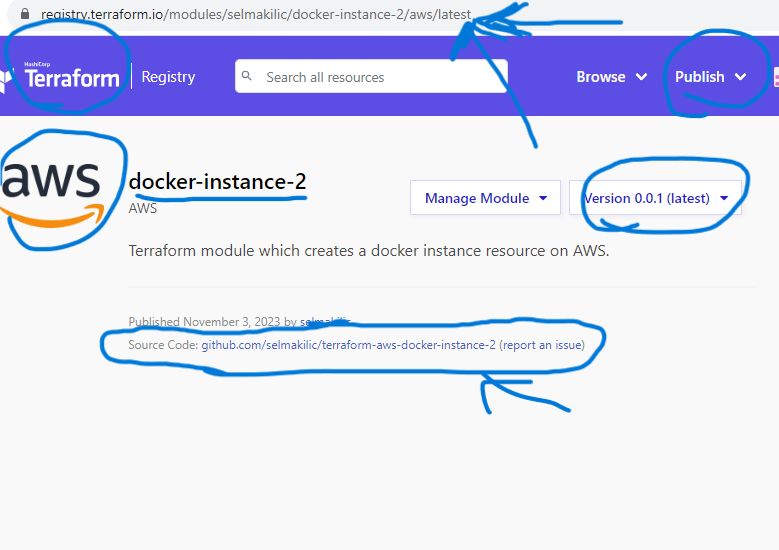
Sonra vscode terraform makinde yine terraform-moduls yani yukarida olusturdugumuz dosyali iceren terraform-moduls klasoru disinda ana yerde bir tane “github” klasoru olusutruk ve terminalde bu klasore sag tiklayik “open integrated in terminal”dedik ve bu klasore girdikten sonra komut olarak “git clone” ve “repo adresini yapistirdik” boylece repo ile locakdeki klasor birbiri ile baglantili oldu “ git clone “ komutu repodan butun bilgileri local bilgilasaya yada uzaktan baglanti kurdugumuz bunun gibi insatance makinelere cekilmis olunur.

Sonra “github” klasorumuz repo ile eslestikten sonra terraform makinde digger lasor altinda olsturdugumuz digger tum main.tf -.gitignore -userdat.sh ve digerlerini kopyalayip “github” klasoru icine yapistirdik.Vscode terminal de hala “github” klasoru icindeyken bu klasorun icindeki tum dosyalari github repoya gondermek icin komut satirina “git add . “ ve “git commit -m “istenilen bir yorum yazilir “ ve “ git push “ ile git reposuna localden dosyalar gonderilmis olunur. Ayrica gonderilen calismalarin tumu icin versiyonlama yapilir istersek github repodayken ust sol tarafta “tag” kismina “create tag” denilir ve versiyonlama icin “v0.0.1” yazilir yada terminalde komut satirina “git tag v0.0.1 “ yazilarak repoya gondeilen doayalara version verilmis olunur.

Github reposundaki dosyalari public registry ile paylasmak ve burdaki dosyalari genel kullanima sunmak icin : <https://registry.terraform.io/providers/hashicorp/aws/latest> bu siteye gidilir ve hesap acilir ve bu sayfada “ Publish” kisminda ” module” kismina bu git reponing ismi dusmus olunur taglari ile beraber devam edilir .



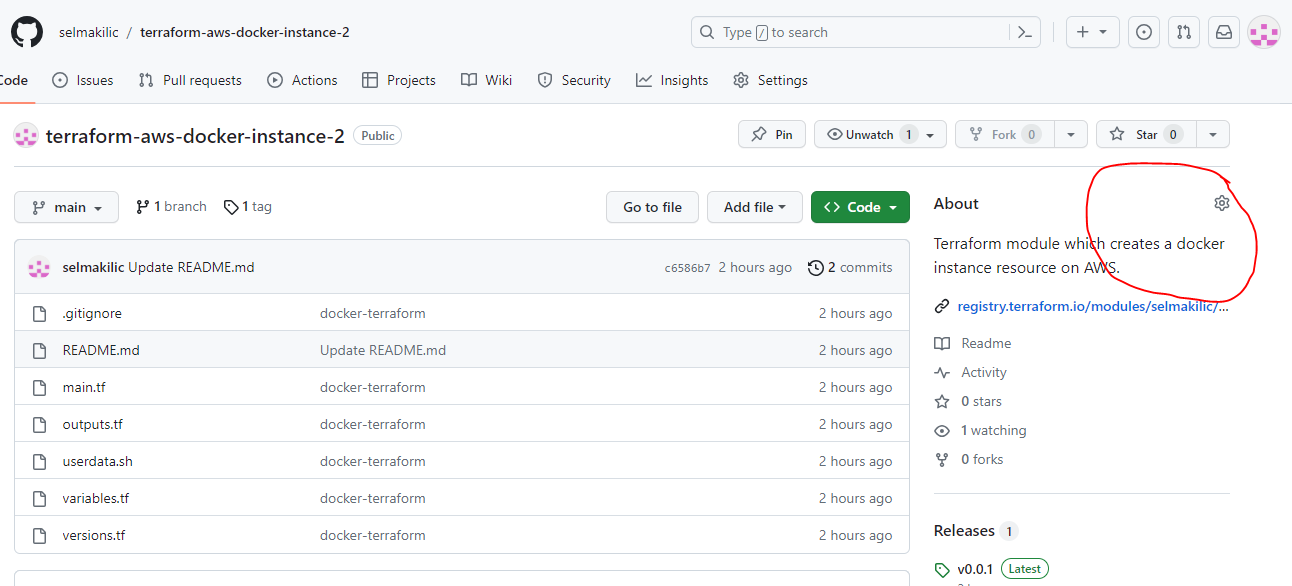


Publis kismina basinca “module” gelir burda bosluga tiklayinca otamatik olarak git reposunun ismi gelir.ilerlettikten sonra ise ;

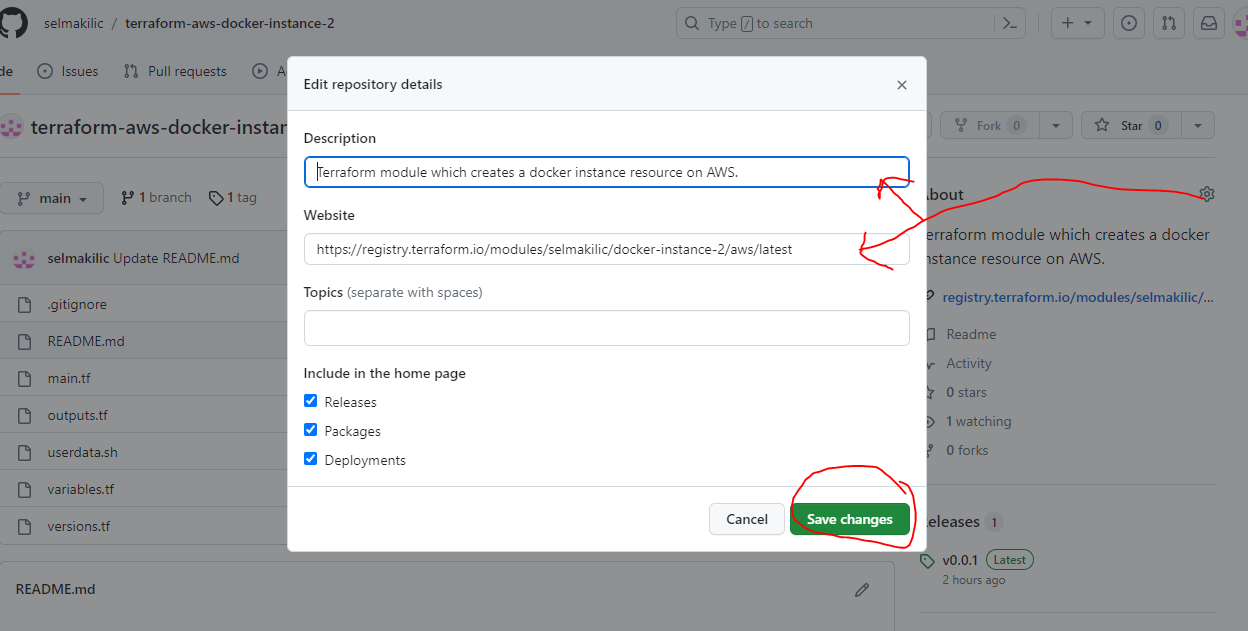
Yukaridaki sayfadaen ust kisimda <https://registry.terraform.io/modules/selmakilic/docker-instance-2/aws/latest> bu sayfada git repodan geldigi gosterilir ayni zamanda

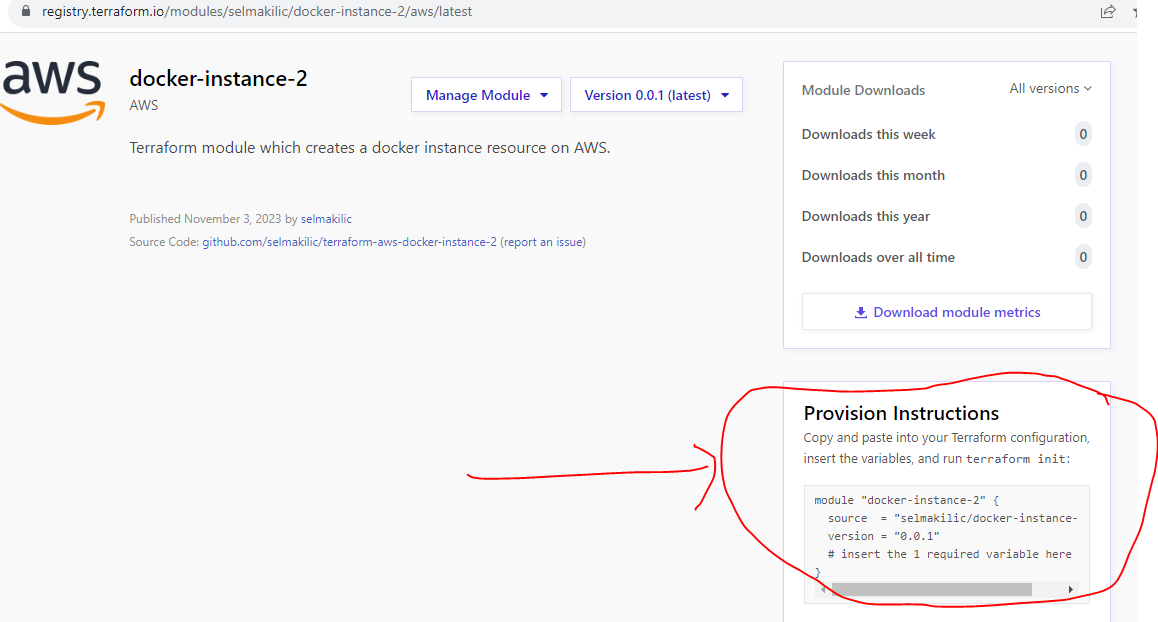
“Source Code: [github.com/selmakilic/terraform-aws-docker-instance-2](https://github.com/selmakilic/terraform-aws-docker-instance-2)([report an issue](https://github.com/selmakilic/terraform-aws-docker-instance-2/issues))”

Bu kisma tiklayinca github repository sayfasina yonledirir.

Github repository sayfasinda ise “

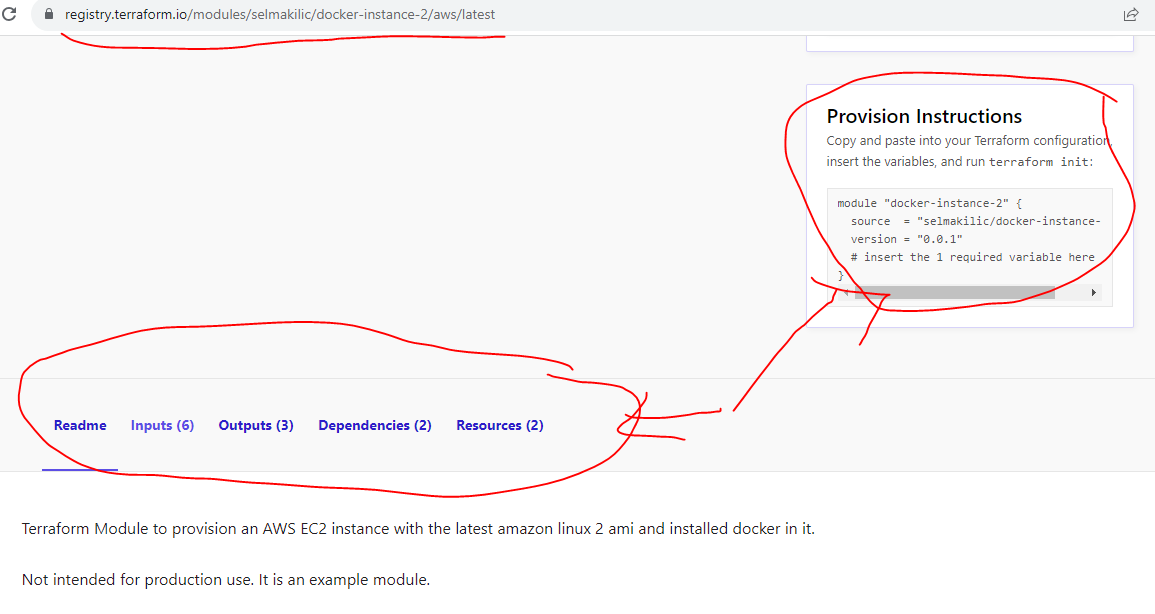
Burda iasaretlenmi sayfanin sag tarafinda bulunan ayarlar kismina gidilip burad acilan yere “Registry” sayafasinin linki yapistilir ve boylece de Github repo sayafasindan da registry sayafasina yonlendiriliecek link baglantisi yapilmis olur.





Terraform Registry sayfasinda son olarak ti=um bu getilen calismalari heresin kullaniminina acmak icin “Provision Instructions” kismi olusur ve burasi kopyalanarak hazir elinizde calistiracaginiz terraform -docker-instanece burdaki gibi bir calisma elde etmis olursunuz.

Bunlar hazir kod olarak kullanilir.



Sayfa sonunda bulunan kissimda ise github repodan cekilen kaynaklar dahil butun bilgiler gorulur .