Infrastructure as Code IaC bir servistir.

IaC nedir? Bir kod yazarak resource ları, infrastructure ı oluşturmadır.

Infrastructure as Code 
• lac is the process of managing 
infrastructure in a file or files rather 
than 
manually 
configuring 
resources in a user interface. 

IaC kişiye bağımlılığı da kurtarıyor şirketleri.

TerraForm da hızlı bir şekilde versiyonlar ve değişiklikler geliyor. Çok hızlı gelişiyor.

HashiCorp şirketi tarafından çıkarılmıştır.

Go dilinden üretilmiş bir dil olan HashiCorp Configuration Language – HCL ile oluşturulan bir configurasyon ya da terraform dosyası oluşturularak kullanılır. .tf uzantılıdır dosyalar.

Daha sonra HCL dilinde yazılan bu dosya çalıştırılıyor, daha sonra terraform bunun çalışması için bazı eklentiler indiriyor. Daha sonra bir exceution plan oluşturuyor. Bunu onaylarsak apply diyoruz. Son kez planı getirip emin misin diye soruyor ve confirm edersek kodu çalıştırıyor.

Terraform Architecture 
TF-Config 
State 
CORE 
2 main components 
PROVIDERS : 
ms I Azure I [laaS] 
Kubernetes I [PaaS] 
Fastiy I 
over 100 providers 
to over 1000 resources 

Terraform un iki ana bileşeni vardır. Biri state diğeri de core dur. Terraform eklentilerle çalışan bir tooldur. Core da extensionlarla çalışan bir arabirim, onlarla iletişime geçen ve anlayan yapıdır.

Conf dosyalarını çalıştırdığımızı düşünelim, bunların metadataları state te tutulur. Daha sonra conf dosyasında değişiklik yapılacaksa, terraform state bakar ve o değişiklikleri yapacağını bildirir bize.

State sayesinde de en başından beri neler yapılmış resource larla ilgili bunu görebiliyoruz.

TERRAFORM PROJECT 
Write: Define infrastructure 
configuration files 
in 
Terraforrn 
Configuration 
Terraform 
State File 
Initialize: Download the correct 
provider plug-ins for the project. 
Plan: Review the 
changes 
Terraform will make to your 
infrastructure 
Apply: Terraform provisions your 
infrastructure and updates the state 
file 

İş akışı olarak önce ne isteniyorsa onlar conf doyasına yazılıyor.

Sonra conf dosyasının olduğu yerde terraform başlatılır.

Daha sonra exceution plan çıkarılır ve altyapıda yapacağı değişiklikleri bize / operatöre bildirir. Bu da hata ihtimalini düşürür

Apply dediğimizde ise hangi platformda hangi resource ların oluşmasını istediysek onları oluşturur.

Terraform da state ten bahsetmiştik az önce, bilgileri tutan yapıdır. State olmazsa terraform çalışmaz. Değişiklik yapıtığımızda da state e bakarak değişiklikleri bize döndürür.

Deklaratif yapı şudur: bizim hali hazırda 3 tane instane ımız var, 5 e çıkarılmasını istiyoruz, state e bakar 3 instance var iki tane daha ayağa kaldırır. 5 yeni instance kaldırmaz.

Providers bileşeni, platformların API ları ile iletişime geçerek o platformlarda işlemler yapılmasını sağlar.

Providers 
A provider is responsible for 
understanding 
API 
Google 
ilii 
amazon 
Cloud Platform 
web services 
Azure 
interactions and exposing 
resources. Every Terraform 
provider 
has 
its own 
documentation, 
describing 
its resource types and their 
arguments. 
kubernetes 
docker 
openstack PostgreSQL Bitbucket 
Nomad 
CONSUL 
DATADOG 
New 
GitHub Relic 
vmware 
vSphere 

Modules, operatörlerin işini kolaylaştırır. Aslında her yazdığımız dosya birer module dir.

Modules 
A Terraform module is a set of Terraform 
configuration files in a single directory. Even a 
simple configuration consisting of a single 
directory with one or more «.tf» files is a 
module. When you run Terraform commands 
directly from such a directory, it is considered 
the root module. So in this sense, every 
Terraform configuration is part of a module. 
main ff 
outputs tf 
Root Module 
- plan 
- a pply 
main.tf 
variaolesff 
outputs ff 
Child Module 
• main_tf 
van If 
Child Module 
variables.tf 
outputs.tf 
Child Module 
MODULES FOLDER 

Başka birinin yazdığı dosyaları kullan diyerek aynı resource ları değişkenleri değiştirerek kaldırmayı sağlayan bileşendir module.

Bunun dışında terraform registry de hazır module ler var. Biz de hazırlayıp koyabiliriz.

Backends bileşeni state i nerede tuttuğumuzla alakalıdır. Default olarak lokalde tutar.

Backends 
A "backend" in Terraform determines how state is loaded and how an 
operation such as is executed. By default, Terraform uses the "local" 
backend, which is the normal behavior of Terraform you're used to. Backends 
are completely optional. You can successfully use Terraform without ever 
having to learn or use backends. Backends are used for keeping sensitive 
information off disk. 
Terraform 
Terraform 
Using AWS S3 
Remote Backend 
Amazon S3 

Şirketin kullandığı platforma göre nerede tutulduğu belirlenir. Örneğin AWS yi kullananan bir şirket S3 te tutar state leri. Bu sayede S3 ün kriptolama ve versiyonlama gibi özelliklerini de kullanabilir.

Resource Blocks:

Resource Blocks 
<BLOCK TYPE> "GLOCK LABEL>" 
# Block body 
"<BLOCK LABEL>" 
# Argument 
resource 
ami 
"aws instance" "web" { 
= "ami-a1b2c3d4 " 
instance type = "t2 . micro" 
* Resources are the most important element in the Terraform language. Each resource block describes 
one or more infrastructure objects, such as virtual networks, compute instances, or higher-level components 
such as DNS records. 
* A resource block declares a resource of a given type ("aws_instance") with a given local name ("web"). The 
name is used to refer to this resource from elsewhere in the same Terraform module, but has no 
significance outside that module's scope. 
* The resource type and name together serve as an identifier for a given resource and so must be unique 
within a module. 

Resource lar oluştururken yukarıdaki şekilde kod blokları ile yazılıyor. HCL dili demiştik. Her resource bloğu bir ya da daha fazla altyapı objelerini tanımlar, yani resource oluşturur. "aws\_instance" burada sabittir ve type tır. "web" local name ya da block label dır. web burada aslında terraform un bu instance ı nasıl tanıdığıdır.

Biz bir resource oluştururken terraform dokumantasyonuna gidip bakıyoruz.

google da "terraform instance aws" yazarsak sonuçlarda çıkacaktır.

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance>

ya da terraform aws provider yazarız.

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs>

Burada sol kısımda resource ları gösterir.

AWS DOCUMENTATION 
aws provider 
) Guides 
) ACM (Certificate Manager) 
) ACM PCA (Certificate Manager 
Private Certificate Authority) 
AMP (Managed Prometheus) 
) API Gateway 
) API Gateway V2 
Account Management 
) Amplify 
) App Mesh 
App Runner 
) AppConfig 
) AppFlow 
i 
AWS Provi er 
Use the Amazon Web Services (AWS) provider to interact with the many 
resources supported by AWS. You must configure the provider with the proper 
credentials before you can use it. 
Use the navigation to the left to read about the available resources. 
To learn the basics of Terraform using this provider, follow the hands-on get 
started tutorials. Interact with AWS services, including Lambda, RDS, and IAM 
by following the AWS services tutorials. 
Example Usage 
Terraform 0.13 and later: 
terraform { 
required _ providers { 
aws 

Terraform registry :

i&rraform I Registry 
Browse v 
Terraform Registry 
Publish v 
Use Terraform Cloud for 
Discover Terraform providers that power all of Terraform's resource types, 
or find modules for quickly deploying common infrastructure 
configurations. 
O Brmvse Providers 
Browse Modules 
O Browse Policy Libraries 
12634 & 
Browse Run Tasks 

Buradan da provider lara erişebiliriz.

Policy libraries , terraformu ücretli kullananların şirketlerindeki policylerine göre policy ekleyip kullanabildiği kısımdır.

AWS çok zengin olduğu için onun provider sayfasında çok fazla resource görülür.

GITHUB oocuMENTAT10N 
github provider 
subjects-la i m n_ternplate 
permissions 
repositories 
actions 
claim 
Iss.ons 
zit 
; it h ub.acticns.secret 
github repository 
github_branch 
t h protection 
github branch_protection v3 
secret_repostori es 
g t h ub_d 
githut_ enterpnse_crgamizatjcn 
Jithub_issue 
• You m 
reauired_oruiders 
block to every module that will create resources 
With this provider. If do not explicitly in g submodule. 
your terraform run may break in hard•tctroubleshoot ways. 
Terraform O. 12 and earlier: 
the GitHub Provider 
-othub" 
resource " 
A Note: 
Vuhen upgrading from to 
user 
. Otherwise, Tsraform stil require the old provider to interact hith the state 
Authentication 
The GitHub provider offers multiple ways to authenticate with GitHub API. 
OAuth / Personal Access Token 
To authenticate using OAuth tokens. ensure that the 
environment variable is set. 
• GIT"uo 
argument or the 

Örneğin GitHub resource ları daha azdır.

Terraform avantajları:

Plaatfor Agnostic tir. yani heterojen bir environment i tek dosyada yönetebiliriz.

State Management:

State Management 
Terraform creates a state file when a project is first initialized. 
Terraform uses this local state to create plans and make changes to 
your infrastructure. 
• Prior to any operation, Terraform does a refresh to update the state 
with the real infrastructure. This means that Terraform state is the 
source of truth by which configuration changes are measured. 
• If a change is made or a resource is appended to a configuration, 
Terraform compares those changes with the state file to determine 
what changes result in a new resource or resource modifications. 

Dilin yazması kolaydır, exceution plan çıkarması ise çok büyük bir kolaylıktır.

Operator Confidence 
• The workflow built into Terraform aims to instill confidence in users 
by promoting easily repeatable operations and a planning phase to 
allow users to ensure the actions taken by Terraform will not cause 
disruption in their environment. 
Upon terraform apply, the user will be prompted to review the 
proposed changes and must affirm the changes or else Terraform 
will not apply the proposed plan. 

You can use both 
ANSI BLE 
Better: for configuring 
that infrastructure 
HashiCorp 
Terraform 
Better: for infrastructure 

Ansible ile kısa bir karşılaştırması:

Terraform resource ları oluştururken daha iyiyken Ansible oluşmuş resource ları daha iyi yönetir. İkisi birbirinin alanına girer ancak birlikte kullanılırlar.

AN SIBLE 
Terraform 
Networking, Load Balancers, Databases, Users, Permissions, etc. 

Uygulama terraform ile oluşturulurken Ansible a deploy ediliyor günümüzde.

Şimdi handson için bir Amazon Linux 2 AMI t2.micro makine kaldırıyoruz.

Remote-SSh ile bağlandık ve update ettik.

sudo yum install -y yum-utils

komutuyla config manager indiriyoruz.

sudo yum-config-manager --add-repo <https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo>

komutuyla HashiCorp resmi reposunu yüklüyoruz.

sudo yum -y install terraform

komutuyla terraform u kurduk.

terraform version

komutuyla version check ediyoruz.

Çok hızlı güncelleniyor demiştik, version güncel değil diyor, bunu görmüş olduk:

rap-172-31-22-176# 
• ec2-use 
Terraform VI. 3.9 
on linux amd64 
:$ terraform version 
Your version of Terraform is out of date! The latest version 
is 1.4.ø. You can update by downloading from https://www.terraform.io/downloads.html 

DevOps ta komutlar tool un ismiyle başlıyor. burada terraform ile başlayacak. terraform --help bize bununla ilgili komutları getirecek.

terraform -help 
• ec2-use 
lesson 
Usage: terraform [global options] Cares] 
The available convnands for execution are listed below. 
The primary workflow convnands are given first, followed by 
less connon or more advanced connands. 
Main connands: 
init 
validate 
plan 
apply 
destroy 
Prepare your working directory for other connands 
Check whether the configuration is valid 
Show changes required by the current configuration 
Create or update infrastructure 
Destroy previously-created infrastructure 
All other commands: 
console 
Try Terraform expressions at an interactive connand prompt 
Reformat your configuration in the standard style 
force—unlock Release a stuck lock on the current workspace 
get 
graph 
import 
login 
logout 
output 
providers 
refresh 
s how 
state 
Install or upgrade remote Terraform modules 
Generate a Graphviz graph of the steps in an operation 
Associate existing infrastructure with a Terraforrn resource 
Obtain and save credentials for a remote host 
Remove locally-stored credentials for a remote host 
Show output values fron your root module 
Show the providers required for this configuration 
Update the state to match remote systems 
Show the current state or a saved plan 
Advanced state mana 

üsttekiler ana komutlar diyebiliriz.

ne demiştik, terraformu çalıştırmak istiyorsak .tf uzantılı dosya neredeyse orada çalıştıracaz.

örneğin apply komutlarıyla ne yapacağımızı görmek istiyorsak terraform apply --help yazıyoruz:

r.p-172-31-28-1# lesson:$ terraforyn -help apply 
• ec2-use 
Usage: terraform [global options] apply [options] [PLAN] 
Creates or updates infrastructure according to Terraforrn configuration 
files in the current directory. 
By default, Terraform will generate a new plan and present it for your 
approval before taking any action. You can optionally provide a plan 
file created by a previous call to fterraform plan% in which case 
Terraform will take the actions described in that plan without any 
confirmation prompt. 
Options : 
-auto -approve 
-backup-path 
-compact-warnings 
Skip interactive approval Of plan before applying. 
Path to backup the existing state file before 
nodifying. Defaults to the &state-out" path with 
n. backup" extension. Set to 
" to disable backup. 
If Terraforrn produces any warnings that are not 
accompanied by errors, show them in a more compact 
form that includes only the sunnary messages. 

bizim burada yazacağımız .tf uzantılı dosyanın resource oluşturabilmesi için;

terraform u linux makinede çalıştırıyoruz. Bu makinenin resource ları ayağa kaldırmak için yetkileri olması gerekir.

Bu nedenle terraform un çalışması için authentication and configuration gerekiyor.

Dokumantasyona baktığımızda bunu görebiliyoruz:

Authentication and Configuration 
Configuration for the AWS Provider can be derived from several sources, which 
are applied in the following order: 
1. Parameters in the provider configuration 
2. Environment variables 
3. Shared credentials files 
4. Shared configuration files 
5. Container credentials 
6. Instance profile credentials and region 
@ ON THIS PAGE 
Example usage 
• Authentication and 
Configuration 
AWS Configuration Reference 
Custom User-Agent Information 
Argument Reference 
Getting the Account ID 
Report an issue ? 

Burada yer alan seçeneklerden birisini kullanabiliriz.

Biz şu anda linux 2 kullanıyoruz ve CLI yüklü geliyor. CLI credentials ları girerek bunu yapabiliriz.

r€ip-172-31-28-1# lesson:$ aws configure 
ec2-use 
AIS Access Key ID L None]: 
AWS Secret Access Key [None]: 
Default region name [None]: 
Default output format ['Ione] : 
lesson:$ expo 
o ec2-use 

Ya da .tf dosyasında aşağıdaki şekilde yazıp çekmesini sağlayabiliriz.

access 
_ key 
" 3'- secret-k 

Ancak bu iki yöntem de güvenli değildir. Amazon bu instance a IAM role vermemizi sağlıyor. Kaldırdığımız instance a ec2fullaccess policy ile role verecez:

Permissions policies (Selected 1/824) Info 
Choose one or more policies to attach to your new role. 
Q 
Filter policies by property or policy name and press en 
"ec2fullaccess" X 
Policy name t? 
Clear filters 
AmazonEC2FullAcc 
Type 
AWS,m 

Daha sonra instance sayfasında instance ı seçip security / modify IAM role diyerek oluşturduğumuz role u seçerek update IAM role diyoruz.

Terraform da çlaışırken dizin önemli, her conf kendi dizininde oluşturulmalıdır:

mkdir terraform-aws && cd terraform-aws && touch main.tf

komutuyla dizini oluşturup main.tf dosyamızı oluşturduk. bu dizin içinde ne kadar .tf uzantılı dosya oluşsa da terraform hepsini tek bir dosya olarak görür.

Bu nedenle sonradan geri dönüp bakması kolay olsun diye resource lar farklı dosyalarda oluşturuluyor.

Şimdi main.tf dosyamızı oluşturacaz.

Biz burada gelip tek tek yazmayacaz. task olarak bir tane aws instance oluşturmamız istendi. Gidip google da aratıp

e ntati on 
O PROVIDER 
Resource: aws instance 
Provides an EC2 instance æsource. This allows instances to be created, updated. and deleted. 
Instances also support provisioning. 
Example usage 
Basic example using AMI lookup 
recent 
filter 
vaL•ÆS 
es/h•.-ssd,-' ubu 
ntu-f,ca1-2ø.aa-e 
- ("hum"] 
How to use this provider 
10 install this this 
into Terraform Then. 
terrafxm init. 
Terraform 0.134 
4 
optiov, 

sağ üstteki use provide butonuna tıklayıp orayı kopyalıyoruz.

Bu kod ile kullanacağımız platform (burada aws) un versionunu belirtmiş oluyoruz. Provider ne demiştik, Terraform un platformlar ın API ile iletişime geçen bileşenidir.

terraform-lesson > terraform-aws > main.tf > 
terraform { 
required _ providers { 
source 
version = 
"hashicorp/aws" 
"4.57.1" 
provider "aws" { 
# Configuration options 

dosyamıza yapıştırdık. ezberlenecek bir şey yok, nereye bakacağımız bilmemiz gerekiyor.

VS code da bazı extension ları indiriyoruz:

Hashicorp Terraform u indiriyoruz. Bu bize yazım kolaylığı sağlar, yazıları renklendiriyor.

HashiCorp Terraform 
v2.25.4 
HashiCorp hashicorp.com 2,551,172 | 
Syntax highlighting and autocompletion for Terraform 
This extension is disabled in this workspace because it is 
defined to run in the Remote Extension Host. Please install the 
extension in 'SSH: my-instance-Inx' to enable. Learn More 

Terraform doc snippe ileride kullanabileceğimiz bir eklenti, şu an gerek yok:

Terraform doc snippe 
Run at scale 077848 
Terraform code snippets pulle... 
19ninstå19 
This extension is disabled globally by the 
DETAILS 
user. 
FEATURE CONTRIBUTIONS 
CHANGELOG 

Şu an değil ancak ileride terraform ile çalışan bir şirkette işe girersek intellij IDEA nın extension ları terraform için daha iyi çalışır:

IntelliJ IDEA 
Coming in 2023.1 What's New Features Resou 
Upcoming Webinar: OAuth for Java Developers Thursday, March SO, 2023 15:00 — 16:00 UT 
Register 
IntelliJ IDEA - the Leadl 
Java and Kotlin IDE 
The IDE that makes development a more productive and 
enjoyable experience 
Our website uses s 
the purposes of ac 
access to the tele 
collection and coo 
it may affect how 
With your consent, 
address to collect 
personalized offer 
the Terms of Use. 
this purpose. You 
time by visiting t 
[A]ccept All U 

AWS de instance için önce region seçmemiz gerekiyor, eklediğimiz extension sayesinde region yazdığımızda bloğu getirir:

provider aws 
# Configuration options 
region 

resource 
"name" { 
aws 
aws 
accessanalyzer analyzer 
Tv'S 
er@ip-172- 
aws 
e*ip-172-31 
er@ip-172-31 aws_ 
E. md tennafc aws 
accessanalyzer archive rule 
account alternate contact 
aws 
acm certificate 
aws 
acm certificate validation 
aws 
acmpca certificate 
aws 
OUTPUT 
aws_acmpca_certificate_authority 
aws_acmpca_certificate_authority_certificate 
acmpca_permlsslon 
acmpca_policy 
alb 

resource yazdıktan sonra type için bize önerileri getirdi. biz aws instance yazacaz.

name kısmına biz istediğimizi yazacaz.

bunu dokümantasyondan da kopyalayıp yapıştırabilriz.

instance için olmazsa olmaz neler var;

ami,instance type:

v resource "aws instance" "tf-ec2" 
"ami-oe6dcf34cegesoe22" 
instance_type 
"t2. micro" 
tags 
" Name" 
"created-by-tf" 

burada role, key-name vs girebiliriz. biz şu anda basit bir instance oluşturacağımız için girmiyoruz.

terraform dosyamız hazır çalışması için öncelikle terraform init ile initialization yapacaz.

rap-172-31-28-1# lesson:$ terraform init 
• ec2-use 
Terraform initialized in an empty directory! 
The directory has no Terraform configuration files. You may begin working 
With Terraforsn irmnediately by creating Terraforyn configuration files. 
O ec2-us 
—172-31-28-1# lesson: $ terra 

terraform dosyası olmayan bir dosyada terraform apply dediğimizde burada bir terraform dosyanız yok diye uyarı verir. burada yapmanın yolu da -chdir=DIR komutunu eklemektir:

Global options (use these before the subconnand, if any) : 
-help 
- version 
Switch to a different working directory before executing the 
given subcomand_ 
Show this help output, or the help for a specified subconvnand. 
An alias for the "version" subcomand. 

terraform init komutunu girdik, yukarıda belirttiğimiz source tan bazı eklentileri indirecek.:

terraform { 
required _ providers { 
source 
version 
"hashicorp/aws" 
"4.57.1" 

r@ip-172-31-22-176# terrafom-a.s:$ terraform init 
• ec2-use 
Initializing the backend... 
Initializing provider plugins... 
- Finding hashicorp/aws versions matching "4.57.1"... 
- Installing hashicorp/aws v4.57.1. 
- Installed hashicorp/aws v4.57.1 (signed by HashiCorp) 
Terraform has created a lock file .terrafom.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 selections by default when 
you run "terraform init" in the future. 
Terrafomn has been successfully initialized! 
You may now begin working With Terraform. Try running 
any changes that are required for your infrastructure. 
should now work. 
"terraform plan" to see 
All Terraform commands 
If you ever set or change modules or backend configuration for Terraform, 
rerun this command to reinitialize your working directory. If you forget, 
commands Will detect it and remind you to do so if necessary. 
p-172-31-22-176# terrafomn-a.s : 
o ec2-use 
other 

bizim başka provider larımız olsaydı onları da source tan indirecekti. İndirdiği plug-in leri görüyoruz:

v .terraform/providers r.„ 
terraform-provider-a... 
.terraform.lock.hcl 

terraform.lock.hcl dosyası ile bizim için versiyon kilitliyor:

provider "registry. io/hashicorp/awsn 
- "4.57 1 
vers Lon 
constraints - 
"4.51 
hashes = 
"hl : REIxUibBFYIKyeQcgOov4DUQ-" , 
" zh : 469e6fe6a9e9ge€ecb168d32fe5e2e9a83cf161f3916øde75ff96f7674c51ee1" , 
" Zh : 611Øba2c15a22686S2ec9ea379 /dde216de84ece42ge5S c49e af9c aa 2be1 ed62 , 
" uh : 7a8øe6fa9b35be178bb566eg3f7984dd6ffb7 , 
"zh : eaaa8e9ebaø9ada41e13116d52d4baeceø4fead8fcf3eab68cca3a67ed738e18" , 
" Zh : ec 169c 89decS4d54B1f1 Sca aeØ869a dg182c 6b6db6Sb" , 
" zh : ff6Ø53942c4øa999ø4bd4a7f3cø82c1fa8f927ecceø374566eb7e8ee8145e582" , 

şu an ki version 4.57.1 . ileride bir başkası değişiklik yapmak istediğinde güncel versiyona göre bir girdi yapacak ancak o eski versiyonda çalışmayacak. bu nedenle terraform versiyonu kilitliyor.

versiyonu eski bir version girip terraform plan dediğimizde çıkan uyarı:

's terraform-aws;$ terraform plan 
Inconsistent dependency lock file 
The following dependency selections recorded in the lock file are inconsistent with the current configuration: 
provider registry.terraform.io/hashicorp/aws: locked version selection 4.57.1 doesn't match the updated version constraints 
To update the locked dependency selections to match a changed configuration, run: 
terraform init -upgrade 
"4.55.1" 

bu durumda komutun terraform init -upgrade şeklinde girilmesigerekir:

terraform init 
ec2 -use 
Initializing the backend... 
Initializing provider plugins... 
- Finding hashicorp/aws versions matching "4 8 
- Installing hashicorp/aws v4.8.ø... 
-upgrade 

tabi bu durumda yeni versiyon (4.8.0) ın da plug-in dosyasını indirir:

v linux_amd64 
v 4.57.1 /linux_amd64 
terraform-provider-aws_v4.57.1 _x5 
.terraform]ockhd 

bu kez de (4.8.0) versiyona kilitler:

provider "registry. io/hashicorp/aws 
version 
constraints = . 
hashes = L 
"zh:16cbdbc03ad13358d12d33e645e2ab5a615e3a3662a7de3c317267c9377713d8' 
" z h : Id813c5e6c21fe37Ø652495e29f783db4e65e37f913ffød53d28515c36fbb7aa' 
" zh : 31 
" zh: sagga9ecggnacabb5686d7cb57cag10fgcß7eg77c65aecedd1g6e838387f,n• 
" zh: 5758 cbb813ø91db8573f27bba37c48f63bag5f2W4f3bc,19f13131e3c3Ø5b848' 
z h : 67ea 
" zh : gb12df85d86d96aedd8d7g8dbøff811adbOe3d88ddd1a3fbdcab589dMfad25 
" z h: f9b1e42ebc62fea7103ø8ef5c31e45b94c8ea1eb8f27fa1ef3' 
" z h : f11bb34e SdbS1e7 e29g a4fe3Øc5egf6b 608Bd611791 cc99deebaaa ' 

Bunun amacı versiyon değişikliklerinde hata alınmamasıdır.

peki eğer ilk bloğu yazmasaydık çalışır mıydı?

terraform { 
required _ providers { 
source 
version 
"hashicorp/aws" 
"4.57.1" 

AWS ve diğer official provider lara özel olarak çalışır:

aws 
AWS 
O 
Kubernetes 
Azure 
Alibaba Cloud 
Google Cloud Platform 
Oracle Cloud 
Infrastructure 

Diğer provider larda çalışmaz. Bu platformların provider ları resmi olarak veriyor.

Şimdi terraform plan komutunu girebiliriz:

Terraform will perform the following actions: 
# aws instance.tf-ec2 will be created 
+ resource ••aws instance" "tf-ec2" 
+ id 
" ami - øe6dc+34cøge5ØØ22 
arn 
availabi lity_zone 
cpu_core_count 
dis 
disable_api_termination 
-e ebs_optirnized 
eet_pa s swo rd_d at a 
host id 
host_resource_group_arn 
instance initiated shutdown 
instance state 
instance_type 
ipv6_address_count 
ipv6_addresses 
key_name 
monitoring 
outpost_arn 
passwrd_data 
placement_eroup 
(known 
(known 
— (known 
— ( knobm 
— ( knovm 
= (known 
- (known 
false 
— (known 
— (knovm 
( knol•m 
— (known 
(known 
( knot•m 
behavior = 
— (known 
- (known 
= ( known 
- (known 
— (known 
( knom 
= (knom 
— (knowtl 
. (known 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
after 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 
apply) 

en sona geldiğimizde vermediğimiz bilgiler için know after apply yani oluşturulduktan sonra bilinecek diyor diğerleri de girdiğimiz bilgileri bize gösteriyor.

terraform apply komutunu girecez. terraform plan demeden direk bu komutu da girebilirdik ancak burada da bize exceution planı gösterir. son kararın mı gibi bir soru olarak düşünebiliriz.

bizden teyit istiyor:

Do want to perfom these actions? 
Terraform will perform the actions described above. 
Only •yes • will be accepted to approve. 
Enter a value: 

sadece yes girebiliriz, no seçeneği yok. yes dedikten sonra resourcu oluşturur:

Enter a value: yes 
æs instance.tf-ec2: Creating... 
instance.tf-ec2: 
Still creating... [IBS elapsed] 
instance.tf-ec2: 
Still creating... [20s elapsed] 
instance.tf-ec2: 
Still creating... [30s elapsed] 
instance.tf-ec2: Creation complete after 32s [id=i-ø6f4e5ø1565cb959ø] 
Apply canplete! Resources: 1 added, changed, destroyed. 

Konsoldan da görebiliriz:

Instances (2) Info 
Q Find instance by attribute or tog (case-sensitive) 
c 
Connect 
Name 
Terraform-Server 
created-by-tf 
V Instance ID 
i-02dbcd12f73d19f09 
i-06f4e501565cb9590 
Instance state v 
@ Running 
@ Running QA 
Instance type 
t2.micro 
t2.micro 
Instance state 
Status check 
@ 2/2 checks passe 
@ Initializing 

terraform apply dediğimizde terraform.tfstate dosyamız oluştu:

main.tf X {l 
terraform ffstate X 
terraform-lesson > terraform-aws > terraform.tfstate > 
"ami": "ami -øe6dcf34cøge5øe22" , 
"arn": 
"associate_public_ip_address" : 
true, 
"availability_zone": "us-east-Id" , 
"capacity_reservation_preference": "op 
"capacity_reservation target" • 
"cpu_core count" : 
"cpu_threads_per core : 
"credit_specification" : 
"cpu 
credits": "standard" 
"disable_api_stop" 
false, 

Terraform da terraform yüklü bir makineyi kaldırmak challenge ımız olsun

terraform state list ile state leri görebiliriz:



terraform state show aws\_instance.tf-ec2 komutu ile state i terminalde görebiliriz:

[ec2-user@ip-172-31-22 
-176 terraform-aws]$ terraform state show aws_instance.tf-ec2 
# aws instance.tf-ec2: 
resource "aws instance" "tf-ec2" 
arn 
associate public_ip_address 
availability _ zone 
cpu_core_count 
cpu_threads_per_core 
disable_api_stop 
disable_api_termination 
ebs_optimized 
get _ password_data 
hibernation 
id 
instance initiated 
instance state 
instance_type 
ipv6_address_count 
ipv6_addresses 
monitoring 
Dlacenent Dartition 
shutdown 
rumber 
behavior 
" arn : aws : ec2 : us-east- 1 : 995714465218 : instance/i-ø6f4e5ø1565cb9591 
true 
"us-east-Id" 
false 
false 
false 
false 
false 
"i -86f4e5ß1565cb959ß" 
" stop" 
" running" 
"t2.micro" 
false 

şimdi main.tf te bir değişiklik yapalım ve key name ekleyelim ve tag i değiştirelim:

v resource "aws instance" "tf-ec2" 
" ami-øe6dcf34cege50e22" 
instance_type 
"t2 . micro" 
key_name = 
tags 
" Name" 
"first-key" 
"tf-ecd" 

yeni bir resource oluşturalım:

resource "aws_s3 bucket" "tf-s3" { 
bucket 
" geovanni -tf -lesson" 

terraform apply dediğimizde çalıştıracak mı? çalıştırmaz çünkü biz sadece ec2fullaccess yetkisi verdik. s3 yetkisi vermedik. terraform plan dediğimizde bu komut çalışır çünkü bize sadece planı gösterecek.

Terraform will perform the actions: 
instance.tf-eci%ust be replaced 
- / + resource aws_instance" "tf-ec2" 
id 
arn 
associate_public_ip_address 
availability_zone 
cpu_threads_per_core 
disable_api_stop 
ebs_optimized 
hibernation 
host id 
host_resource_eroup_arn 
iam_instance_profile 
instance initiated shutdown 
instance state 
ipv6_address_count 
ipv6 addresses 
key_name 
" arn : aws : ec2 : us -east -1 : : instance/i-Ø5e3561aa91a54eaf" 
- true -i (known after apply) 
"us-east-la" (knot•m after apply) 
= I (known after apply) 
- 1 -Y (known after apply) 
(known after apply) 
false 
false 
- false 
false 
(known 
• (known 
(known 
-> (known after apply) 
(known after apply) 
(known after apply) 
-s null 
after apply) 
after apply) 
after apply) 
"i -ø5e3561aa91a5aeaf" (knom after apply) 
behavior = "stop" (knovm after apply) 
"running" (known after apply) 
- (known after apply) 
[J -> (known after apply) 
"oliver" # forces replacement 

planda ec2 nun replace eidleceğini söylüyor neden key pair ekledik. bunu sonradan ekleyemeyeceği için ec2 yu silip başka bir ec2 ayağa kaldıracak.

+ işretliler eklenecek, - işaretliler destroy edilecek, ~ işaretliler değişecekler.

en altta da 2 resource un ekleneceğini 1 resource un da detroy edileceğini söylüyor:

website { 
error document 
index document 
redirect all_requests_to 
routing_rules 
(known after apply) 
(known after apply) 
(known after apply) 
(known after apply) 
Plan: 2 to 
add, 
to change, 
1 to destroy. 

terraform apply dediğimizde bize s3 için yetkisi olmadığı yönünde uyarı verecek:

was instance. tf-ec2: 
instance.tf-ec2: 
aws_instance.tf-ec2: Still creating 
aws instance.tf-ec2: Still creating 
"s_instance.tf-ec2: Still creating... 
aws_instance.tf-ec2: Still creating 
aws instance.tf-ec2•. Still cr-eating... 
Creating... 
Still creating... 
[2es 
[30s 
[40s 
[ses 
elapsed] 
elapsed] 
elapsed] 
[Imøs elapsed] 
Creation ca•nplete after Im2s [id=i 075635aaØf52e38eI] 
creating S3 (Sirvle Storage) Bucket (Oliver-tf-test-bucke±-lessm): AccessDenied: Access Denied 
Error : 
status code: aø3, request id: 
S8NUIM28VJTNMPQ, host id: glbXzoR8LOeX18tML"3/3yfKD2P1tzgay5s21Joefvxnwp1eyh 
with aws s3 bucket. tf-s3, 
on main -tf line 26, in resource 
26: resource "aws s3 bucket" "tf-s3 { 
"aws s3 bucket" 
"tf-s3": 

IAM user dan role s3fullaccess policy de ekliyoruz ve terraform apply diyoruz tekrar.

instance.tf-ec2: Destroying... [id 
s3 bucket.tf-s3: Creating... 
s3 bucket.tf-s3: Creation complete 
instance.tf-ec2: 
Still destroying 
instance.tf-ec2: 
Still destroying. 
instance.tf-ec2: 
Still destroying. 
after Is [id=geovanni-tf-lesson] 
les elapsed] 
2øs elapsed] 
elapsed] 
instance.tf-ec2: Destruction complete after 40s 
instance.tf-ec2: Creating... 
instance.tf-ec2: Still creating... [IBS elapsed] 
instance.tf-ec2: Still creating... [20s elapsed] 
instance.tf-ec2: Still creating... [30s elapsed] 
instance.tf-ec2: Creation complete after 32s [id=i-ø172843øcaføf38f3] 
Apply canplete! Resources: 2 added, changed, 1 destroyed. 

önceki instance ı sildi, yeni instance ı ve s3 bucket u oluşturdu.

eğer apply dedikten sonra bize sormasın istersek "terraform apply -auto-approve" komutunu kullanıyoruz.

gerçek ortamda pek kullanılmasa da farklı bir plan alma seçeneği var. plan - out=justs3 komutu ile.

bir resource u silince ne olur düşüncesiyle plan ile sadece bu değişikliği görebiliyoruz, alternatif bir plan. Bu sayede ana dosyaya dokunulmuyor yeni bir plan oluşturup kaydediyor. daha sonra istediğimizde o dosyayı çalıştırabiliyoruz.

komutu girdiğimizde yeni planı bir dosya olarak kaydetti:

.terraforrnlock.hcl 
justs3 

bu planı çalıştırmak için de aşağıdaki komutu giriyoruz:

-31-28-1# terraform-aws:$ 
ec2-use 
reip-172 
aws_instance.tf-ec2: Destroying... [id=i 
aws instance.tf-ec2: Still destroying. 
"s_instance.tf-ec2: destroying. 
terraform apply justs3-tfpIdn 
instance. tf-ec2: 
Still destroying... 
-Ø75635aaØf52e38e1] 
[id=i-e75635aaøf52e38e1. 
[id=i-e75635aaøf52e38e1, 
IOS elapsed] 
elapsed 
30s elapsed] 
aws_instance.tf-ec2: Destruction after 40s 

.terraformlock.hcl 
justs3 
main.tf 
{l terraform.tfstate 
terraform.tfstate.backup 

yeni bir state de oluştur, güncel state te ec2 yok anca bu backup ta var.

şimdi terraform apply -auto-approve komutu girdiğimizde bize sormadan main.tf yi çalıştıracak ve ec2 ayağa kaldıracak.

Plan: 1 to add, ø to 
change, to destroy. 
. t-f -ec2 
Creating... 
aws_instance.tf-ec2: Still creating... [les elapsed] 
"s instance.tf-ec2 
: Still creating... 
[30s elaps&i] 
was instance.tf-ec2: Still creating... 
aws_instance.tf-ec2: Creation canplete after 32s [id=i-Ø5b1cabØ3f1192998] 
Apply canplete! Resources: I added. 
e changed, destroyed. 
aws 
aws 
O ec2- 
terra-form-aws;$ terraform state list 
instance.tf-ec2 
s3 bucket. tf-s3 
terraform-aws $ 

bu kez de backup state te ec2 olmayacak güncel state te olacak. çünkü bir önceki durumda ec2 yoktu.

şimdi terraform destroy komutuyla resource ları silecek edecek. bu komutta da -auto-approve komutu eklenebilir.

Do 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 
s3 bucket.tf-s3: 
Destroying... [id=geovanni-tf-lesson] 
instance.tf-ec2: 
Destroying... [id=i-ø172843øcaføf38f3] 
s3 bucket.tf-s3: Destruction complete after øs 
instance.tf-ec2: Still destroying... [id=i-ø172843øcaføf38f3, 
instance.tf-ec2: Still destroying... [id=i-ø172843øcaføf38f3, 
instance.tf-ec2: Still destroying... [id=i-ø172843øcaføf38f3, 
instance.tf-ec2: Still destroying... [id=i-ø172843øcaføf38f3, 
instance.tf-ec2: Destruction complete after 40s 
Destroy canplete! Resources: 2 destroyed. 
[ec2-user@ip-172-31-22-176 terraform-aws]$ 
IBS elapsed] 
20s elapsed] 
30s elapsed] 
40s elapsed] 

şimdi state bakıyor ve resource ları görüyor. bu resource ları siliyor. silinmiş haliyle yeni state oluşturacak. backup state te ise resource ların var olduğu yer hali yer alacak.

güncel state:

terraform-lesson > terraform-aws > terraform.tfstate > 
"version": 4, 
"terraform version : 
" "1.3.g", 
"serial": 
"lineage" • 
"outputs" : 
8, 
. "4c216e94-ßdd8-cefd-b3fe-e7a7dß4edd69", 
o, 
"resources" 
"check results": 
null 

backupstate:

terraform-lesson > terraform-aws > terraform.tfstate.backup 
"version": 4, 
"terraform version": 
"serial": 
5, 
"lineage • 
"outputs 
"resources" : 
"1.3.9", 
". "4c216e94-ødd8-cefd-b3fe-e7a7duedd69", 
mode" "managed" , 
"type": "aws instance", 
"name": "tf-ec2", 
"provider" : 
"instances": 
" " regi stry. terraform. io/hashicorp/ aws\"] " , 
"schema version": 1 
"attributes : 
"ami": "ami-øe6dcf34cøgesøe22" , 
arn": "arn 
"associate_public_ip_address": true, 
"availability _ zone": "us-east-Id" , 
"capacity_reservation specification" : 
"capacity_reservation_preference" : 
"capacity_reservation target 
" open" 
"cpu 
"cpu 
core count": 1, 
threads_per core": 1 

devamı da var resource lar yer alıyor.

bu ec2 stop edilecek, bir sonraki derse bu ec2 ya remote ssh ile bağlanıp gelecez.