شرح Terraform

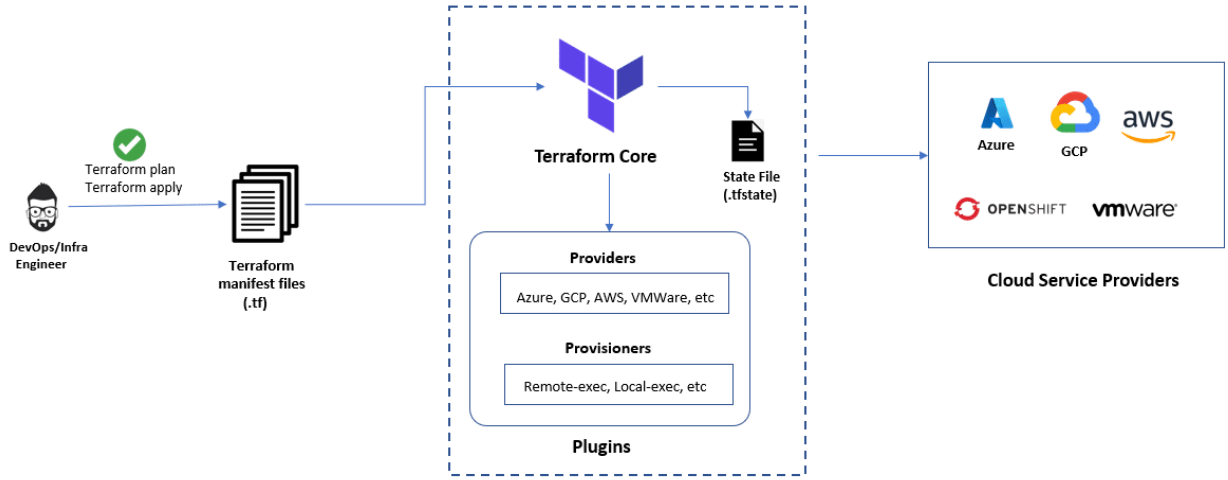
HashiCorp Terraform

BY: Mohamed Atef Elbitawy

https://www.linkedin.com/in/mohamedelbitawy/

لا تنسونا من صالح الدعاء

Terraform



- ال **Terraform** عباره عن Open Source Tool ودي Tool من ضمن ال Tools بتاعت ال Infrastructure as Code بنستخدمها علشان نبني بيها ال Infrastructure. الفكرة إنك تقدر تعمل أو تدير ال Infrastructure بتاعتك (زي السيرفرات، الشبكات، الـDNS، وغيره) عن طريق كتابة كود بدل ما تعمل الخطوات يدويًا
- اللي عملت ال Terraform شركه اسمها Hashicorp وال Terraform مكتوبه ب GO Language

Why Terraform?

- **ليه نستخدم Terraform تحديدا عن باقي ال Tools التانيه**

➢ ال Terraform عباره عن **Cloud Agnostic** يعني ايه Cloud Agnostic يعني Terraform بت Support ان انت تبني بيها Infrastructure علي Cloud Provider مختلفين زي AWS او Azure او GCP ف هي بتتعامل مع كل ال Infrastructure Providers

➢ ال Terraform فيها كل ال concept بتاعت ال Coding يعني فيها فكرة ال Conditions وال Loops وال Variables وهكذا

➢ الاستخدام الخاص بيها سهل جدا وسريع وبالنسبه للمشاكل الخاصه بيها بتكون واضحه

➢ وعلشان ال Terraform بيكون Open Source ف ال Project Active بيتطور دايما

➢ ال community الخاص بيها كبير جدا ف بتالي بيكون فيه Support لو فيه اي مشكله بتقدر ان انت تعمل Search وتلاقي حل للمشكله دي

➢ وكمان من المميزات اللي بتميز ال Terraform وهي ال Documentations الخاصه بيها

How Terraform Works?

- ال Terraformمنقسمه ل جزئين جزء اسمه **Terraform Core** وجزء اسمه **Terraform Plugins**

1- Terraform Core

- ال **Terraform Core** عباره عن statically-compiled binary مكتوبه بلغه Go وال Terraform Core هو ده ال component اللي بتنزلها وانت بتعمل Install لل Terraform by default
- **المسئوليه الرئيسيه لل Terraform Core**
  - انها تقرأ ال Code اللي انت هتكتبه وتعمله ال Interpolation وتت Check ال errors
  - هي المسئوله انها تعمل state management
  - وهي المسئوله انها تكون حاجه اسمها Graph
  - وهي اللي بتعمل عملية ال Plan
  - والمسئوله انها تكلم ال Plugins وت download ال Plugins اللي محتاجها ال code بتاعك وانها تبدء ت communicate معاها انها تسلمها الاوامر اللي هتبدء تنفذها علشان تبني ال Infrastructure علي ال Cloud Provider بتاعك

2- Terraform Plugins

- ال **Plugins**عباره عن component مكتوبه بلغه Go ولكن ال component دي مش بت get invoked directly هي بت get invoked عن طريق ال Terraform Core ال Terraform Core هو الوحيد اللي بيقدر يشغلها عن طريق ال Remote Procedure Calls (RPC)
- عباره عن Tools مكمله بتنزل حسب احتياجك ليها علشان متاخدش مساحه كبيره من الجهاز او السيرفر بمعني ان انت مثلا هتبني Infrastructure ف هتتعامل مثلا مع AWS ف انت هتكتب ال code ف بمجرد ان انت هتعمل Run للكود ف هيبدء ي dedicate ان انت عايز ال Plugins بتاعت ال AWS ف هيبدء انه يعمل Download لل Plugins بتاعت ال AWS اتوماتيك وهكذا
- ال Terraform Core وال Terraform Plugins بي communicate مع بعض من خلال حاجه اسمها remote procedure calls (RPC) ودي بتم علي مستوي ال Operating System
- **المسئوليه الرئيسيه لل Terraform Plugins**
  - هي المسئوله انها ت Handle عملية ال Calling مابينها ومابين ال Cloud Providers APIs
  - وهي المسئوله انها تعمل عمليه ال Authentication مع ال Providers
  - وبترتب ال Creations الخاصه ب ال resources

Terraform Concepts

- مبدئيا Project ال Terraform ببساطه بنبقي عاملين Folder فاضي ونبدء جواه نعمل create ل Files وال Files دي اهم حاجه يكون ال Extensions بتاعها يكون **.tf** ف ال files دي ملهاش اي تسميه معينه وملهاش اي ترتيب معين وليكن علشان هنتعامل مع اول حاجه وهي ال Providers ف هنعمل file اسمه provider.tf
- ف لما تعمل Run ل اي command Terraform وانت جوه ال Folder بيبدء يعدي في ال folder ويشوف كل ال Files اللي بيكون ال Extensions بتاعها **.tf** وبياخدها وبيبدء يرتب اتوماتيك ويبدء ينفذها

Providers

- ال Providers دا اول خطوه دايما بنعملها علشان نبدء نكتب Terraform Code
- ال Provider دا عباره عن Section او Template لكل resource او لكل جزء في ال Terraform بنستخدمها وبنكتبها بشكل معين في ال file الخاص بال provider
- وال Providers انا بحدد بيه code ال Terraform اللي انا هكتبه في ال folder اللي هبدء هتعامل فيه مع انهي Providers اذا كان AWS او Azure او GCP او OpenStack او Kubernetes والرابط ده https://registry.terraform.io/browse/providers عليه كل ال Providers اللي ممكن تستخدمهم
- ومن خلال ال Documentations هتلاقيه شارحلك ازاي تكتب ال sections وشارحلك ازاي ت Authentication مع كل Providers وهكذا

➢ ف انت تقدر ان انت تستخدم اي Cloud Providers ف انا هستخدم ال AWS ف احنا هنعمل file زي مشرحنا خاص بال provider وليكن هيكون اسمه provider.tf وهنكتب جواه ال section الاتي وال section بيكون بالشكل ده . ان انت بتحدد انهي provider انت هتستخدمه ف احنا هنستخدم ال AWS وبعدين حددنا ال region اللي هيشتغل عليها وهتبقي ال default علشان الكود بتاعك يبدء ي create فيها ال Resource

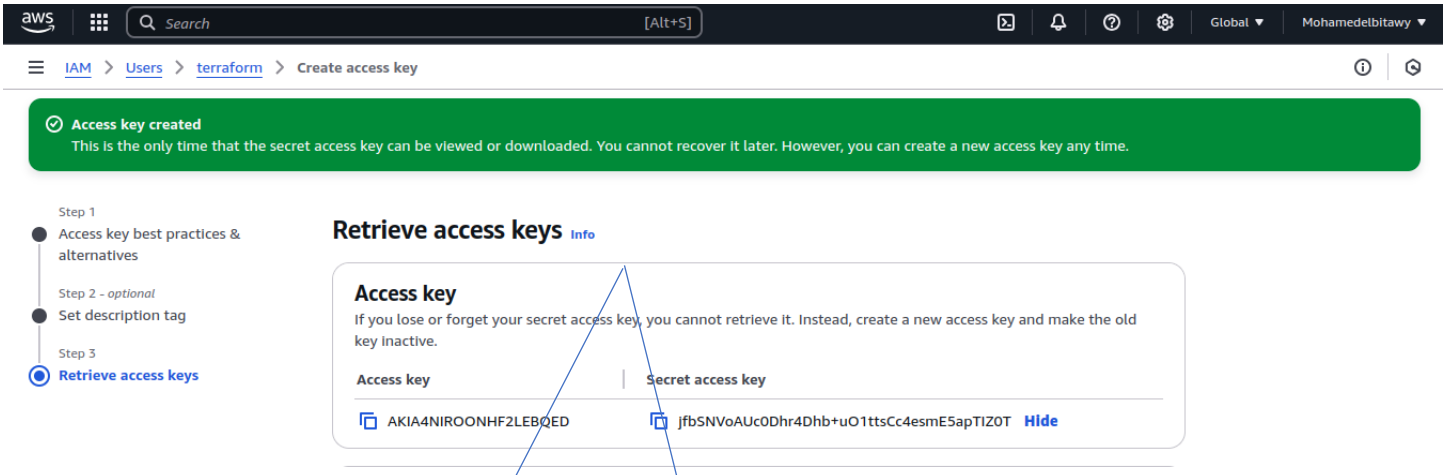
```
# Configure the AWS Provider
provider "aws" {
  region  = "us-east-1"
}
```

- بعد كده لازم انا اعرفه الاكونت بتاعي في AWS فين لان AWS Provider عندهم اكونتات كتيره جدا ف هو ازاي هيميز الاكونت بتاعي عن باقي الاكونتات علشان يعمل build علي الاكونت بتاعي ان فين ال credentials اللي توصله للاكونت بتاعي كل Provider من ال Providers بيحتاج ان انا اعمله Authentications مع ال Service اللي هتدير ال resources الخاصه بيها
- فيه عندي اكتر من طريقه علشان اعمل Authentications او انا ابصيله ال Credentials بتاعتي

1. اول طريقه وهي ال Static Credentials

```
# Configure the AWS Provider
provider "aws" {
 region  = "us-east-1"
 access_key = "my-access-key"
 secret_key = "my-secret-key"
}
```

- والطريقه دي عباره عن ان انت بتحط في الكود بتاعك Two Parameters وهما ال access_key وال secret_key بس الطريقه دي ليها عيب خطير جدا ان احنا بنكتب ال credentials في ال code وال code ده بيترفع علي ال Git repository ف ده معرض لمشاكل كبيره جدا ان ممكن اي شخص يعرف ال credentials دي ومعرض لأي Attack
- وال access_key وال secret_key في ال AWS بنجبهم عن طريق ان احنا بنعمل Create ل user من خلال ال IAM ونعمل create لل Access Key



- ف احنا بناخد ال Access key وال secret access key ونحطهم في ال file بالشكل ده

```
# Configure the AWS Provider
provider "aws" {
 region  = "us-east-1"
 access_key = "AKIA4NIROONHF2LEBQED"
 secret_key = "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"
}
```

2. تاني طريقه وهي Environment Variables

```
% export AWS_ACCESS_KEY_ID="anaccesskey"
% export AWS_SECRET_ACCESS_KEY="asecretkey"
```

- ودي فكرتها ان بدل اما تحط ال credentials في الكود بتعمل export environment variables عن طريق ال CLI ان انت بتديله ال AWS_ACCESS_KEY_ID وال AWS_SECRET_ACCESS_KEY عن طريق ان انت بتكتب في ال CLI الاتي
- بتديله ال access_key وبعدين ال secret_key بالشكل ده من خلال ال CLI

```
mohamed@MohamedAtef: ~/terraform$ export export AWS_ACCESS_KEY_ID=AKIA4NIRONHF2LEBQED
mohamed@MohamedAtef: ~/terraform$ export AWS_SECRET_ACCESS_KEY=jfbSNVoAUc0Dhr4hb+uO1ttsCc4
```

3. تالت طريق وهي ال Shared Configuration and Credentials Files

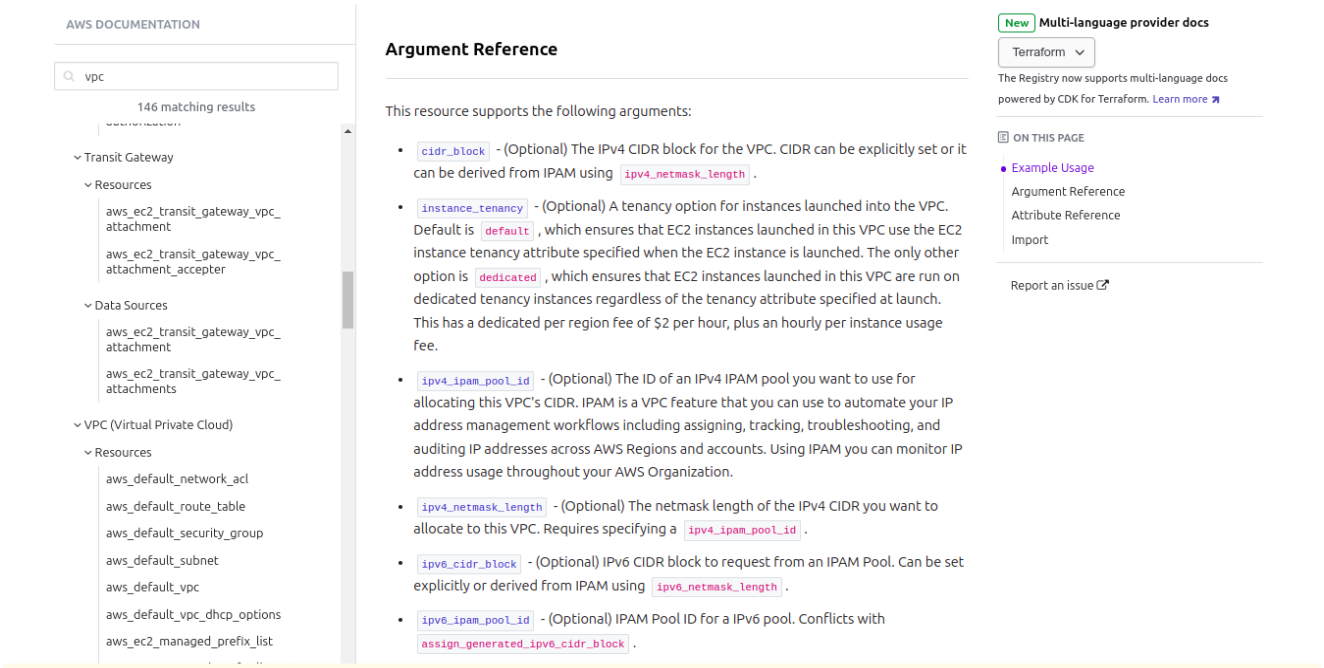
- والطريقه دي اغلب الناس بتستخدمها عن طريق ان احنا بنستغل ال Credentials files بتاعت ال AWS CLI
- الفكره ببساطه ان Terraform علشان تدور علي ال credentials بتبص اول حاجه لو انت كاتب ال credentials دي في file ولا لا لو كاتبهم في file هتستخدمهم لو مش كاتبهم في file هتبدء تدور في ال environment variables بتاعتك لو ملاقتهمش في ال environment variables هتبدء تدور في ال Path اللي تحت /.aws/~ علي file اسمه credentials لو ال file موجود ومكتوب فيه ال credentials بشكل معين هتستخدمه
- ف ال file اللي اسمه credentials بينكتب فيه الاتي

```
[default]
aws_access_key_id= AKIA4NIROONHF2LEQED
aws_secret_access_key= jfbSNVoAUc0Dhr4Dhb+uO1ttsc4esmE5apTIZ0T
```

- نقدر كمان في ال Terraform ان احنا نضيف Resource وليكن مثلا احنا عايزين نعمل Create ل resource معين وليكن ل VPC ف اللي احنا هنعمله هنعمل Create ل file وليكن هنسميه vpc وهيكون ال Extensions زي ماقولنا vpc.tf وهيكون جواه كل ال resources اللي احنا محتاجنها
- ف ال resource بتتكتب بالشكل ده ودايما استخدم ال Documentations وده مثال بسيط علي ال resource وده الشكل ال Standard لل resource اللي هنشتغل بيه

```
resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"
}
```

- دايما الجزء الخاص بال resource بيكون خاص بكل ال Providers ان هيكون بنفس ال Format وال Format بيكون عباره عن
  - ان اول جزء بنقول resource
  - وبعد كده بنبدء نحدد اسم ال resource اللي هنستخدمه زي aws_vpc وده بنجيبه من ال Documentations
  - وبعد كده بنكتب اي اسم من اختيارنا زي ال main وده زي ال Variable في ال Coding علشان نبدء بعد كده نستخدمه في اماكن مختلفه
  - وبعد كده بنفتح curl bracts{ } ونبدء نكتب جواه حاجه اسمها Arguments وال Argument Reference دي بيكون فيها كل ال Parameters اللي ممكن تباصيها علشان ت Create ال Resource وال Resources دي فيه جزء هو بيعتبره Mandatory لازم انت تحدده زي ال cidr_block وهتلاقيه مكتوب جنبه Required وجزء تاني بيكون Optional وOptional علشان هو بيكون حاططلهم Default Values لو انت مخترتلهمش حاجه



init

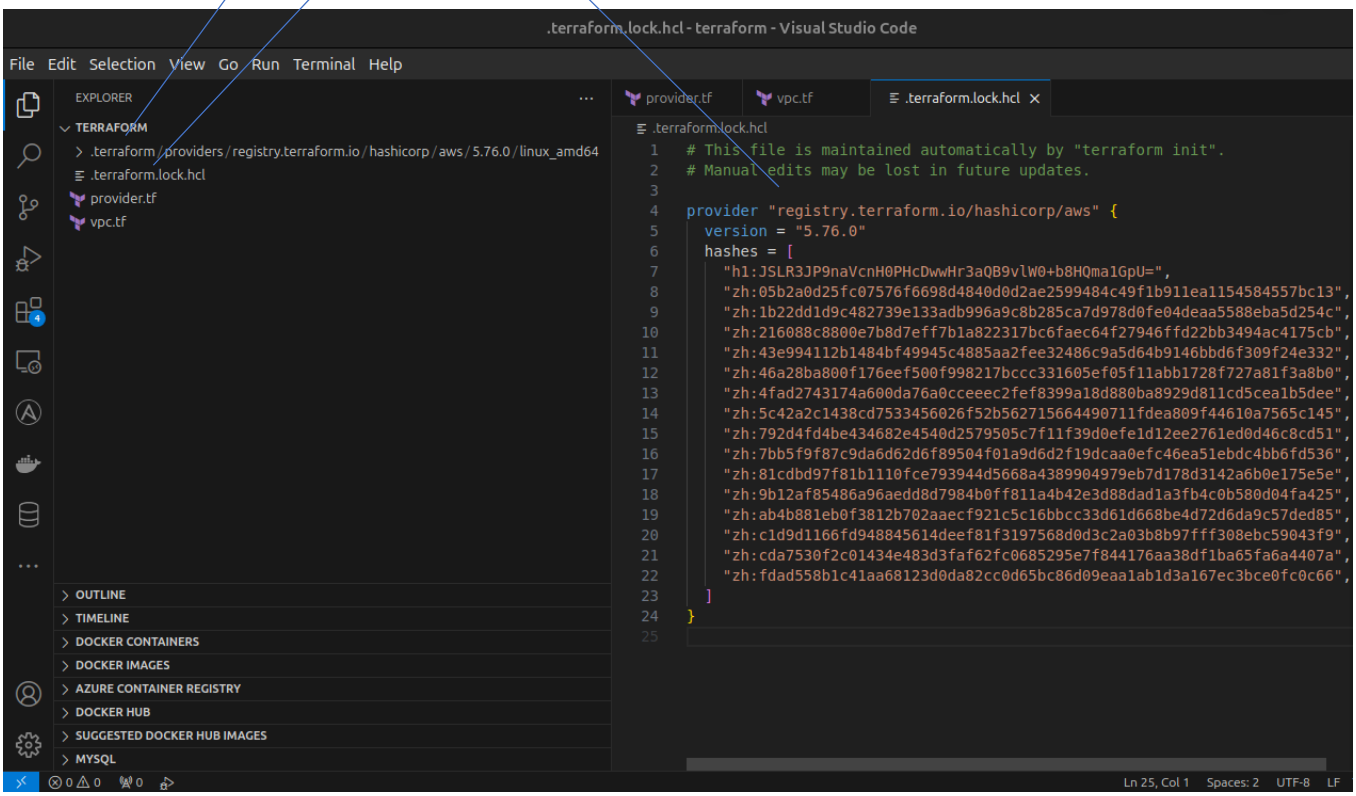
- ال **Terraform Init** عباره عن Command بنعمله Run لما نبدأ أي Project باستخدام Terraform. الغرض من ال Command ده ان هو بيعمل (Initialize) لل working directory اللي بيحتوي علي Terraform configuration files
- علشان نبدء ن Run ال Code الخاص بال Terraform محتاج تبقي واقف في ال Terminal في ال Root Directory اللي فيه ال Files الخاص بال Terraform ف اول خطوه هنعملها هنستخدم command ال init ف هنستخدم terraform init علشان نRun ال Code وال init زي ماقولنا بيعمل اول حاجه Initializing لل Backend وبعد كده بيبدء يعمل Download لل Provider Plugins اللي محتاجها الكود بتاعك

```
mohamed@MohamedAtef: ~/terraform$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.76.0...
- Installed hashicorp/aws v5.76.0 (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.

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.
```

هتلاقيه هنا بيقولك انه بيعمل اول حاجه Initializing لل backend

هتلاقيه هنا بيعمل Search علي ال Plugins وبعدين بيعملها Download

- و زي ماقولنا ان ال Plugins عباره عن Component بتكون منفصله ومكتوبه بال Go الكود بيدونها وبتبدء ال Terraform انها تمانجها ف بالتالي هو مش بيعملها Download في ال Operating System لان ال Component دي بتعتبر مؤقته related لكل Project ف ال Plugins بيتعملها Download علي مستوي ال Project بتاعك مش علي مستوي الجهاز بتاعك هو بيبدء يعمل Download لل Plugins دي في ال Folder اللي اسمها terrform. بيعمل download تحت terraform. علشان ال Plugins دي في منها Versions لان علي حسب ال Project بتاعك بتستخدم ال Plugins المناسبه ب Versions تانيه علشان ميحصلهاش clash ف كل ال Plugins بتنزل علي مستوي ال Project تحت ال Folder اللي اسمه terraform.
- وهنلاقيه كمان عمل Download ل File اسمه terraform.lock.hcl. ال file ده فايدته ان انت لو سبت ال file ده موجود في الكود بتاعك واخدت الكود ده علي جهاز تاني وعملت init هيبدأ يعمل install لنفس ال Plugins ونفس ال Versions ونفس كل حاجه



- احنا كده عملنا ال Configuration بتاعت الكود وبقت connected لل AWS Account وكتبنا كمان ال Resource جوه ال VPC وعملنا كمان Initializations كده كل حاجه جاهزه ان احنا نبدأ نستخدم ال Terraform علشان ن Run ال Code

Plan

- ولكن Terraform بتوفر ميزه جميله جدااا مش موجوده في معظم ال Tools التانيه وهي ان هي اديتك Options ان انت تبقي قادر تعرف ايه اللي هيحصل بالظبط لو انت عملت Run للكود بتاعك علي الاكونت قبل اما هي تبدأ ت Run الكود فعلياا والاوبشن ده اسمه ال **Plan Command**
- وال Plan Command بيبدأ يقارن الكود المكتوب بال state بيبدأ يشوف هل فعلا ال Resource ده موجود فعلا ومعمول ليه Create علي AWS Account ولا لا ولو موجود هيقولك انه مش هيقدر يعمل ليه Create ولو مش موجود هيبدأ يعمل ليه Create ولو فيه اي تغيير هيبدأ يغيرها
- علشان نعمل Run لل Plan هنستخدم Command اسمه Terraform Plan
- بمجرد ان انت بتعمل Run لل Plan هيطلعلك ال Output بالشكل ده هيقولك ان الخطه بتاعتي او ال Plan هتضيف Resource واحد وهيغير zero resource وهدمر zero resource والكلام ده بعد اما نعمل Apply هيبدأ يتغير بعد اما اعمل Create لل Resource

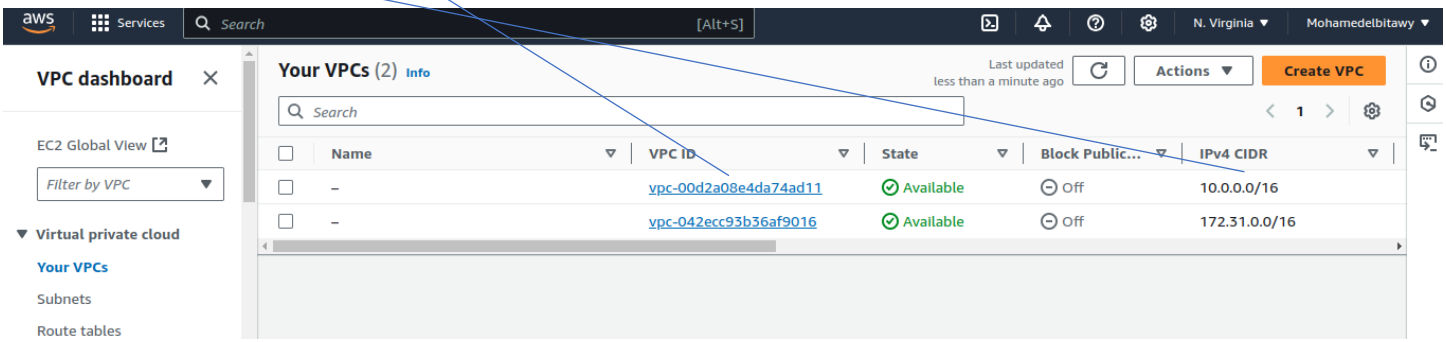
```
mohamed@MohamedAtef: ~/terraform$ terraform plan

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.main will be created
  + resource "aws_vpc" "main" {
      + 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_dns_hostnames                 = (known after apply)
      + 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: 1 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't
guarantee to take exactly these actions if you run "terraform apply" now.
```

هنلاقيه هنا قايلك انه عمل add واحد بس

- هنعمل Apply عن طريق Command اسمه Terraform apply ال Apply ده ال Command التاني اللي بنفذه بعد ال Plan وده اللي بيدأ ينفذ الحاجه فعليا
- ف لما ننفذ ال apply هيعرضلك ال Plan تاني ويسألك هل انت عايز تنفذ الكلام ده ولا لا ف لو قولتله yes هينفذلك الكلام ده وهيقولك انه عمل Apply complete وفيه Resource انضاف ومفيش اي تغيير ومفيش اي destroyed

```
mohamed@MohamedAtef: ~/terraform~$ terraform apply

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.main will be created
  + resource "aws_vpc" "main" {
      + 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_dns_hostnames                 = (known after apply)
      + 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: 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.main: Creating...
aws_vpc.main: Creation complete after 2s [id=vpc-00d2a08e4da74ad11]

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

- ف لو انت فتحت ال VPC علي ال AWS هتلاقيه عمل Create لل VPC وال CIDR اللي احنا حددناه



- لو احنا جينا فرضا ان احنا غيرنا في ال Configuration بتاعت ال resource في ال VPC ان احنا هنعمل false لل dns ف احنا هنعدل في ال Configuration وهنضيف الجزء الخاص بال dns
- ف لو عملنا plan تاني

```
resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"
  enable_dns_support = false
}
```

- ف لو جينا عملنا plan تاني للكود هنلاقيه قايلنا ان حصل تغيير واحد فقط

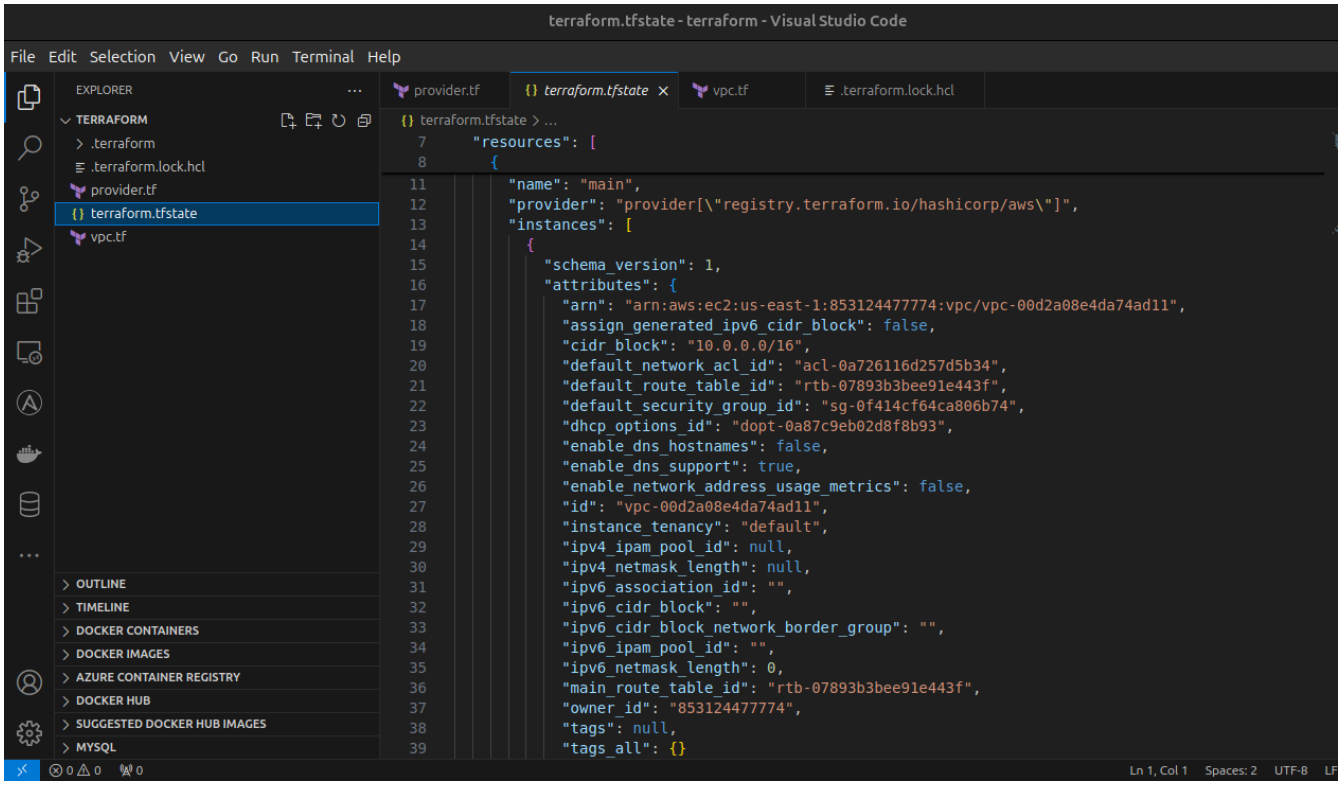
```
mohamed@MohamedAtef: ~/terraform$ terraform plan
aws_vpc.main: Refreshing state... [id=vpc-00d2a08e4da74ad11]

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_vpc.main will be updated in-place
  ~ resource "aws_vpc" "main" {
      ~ enable_dns_support                 = true -> false
        id                                 = "vpc-00d2a08e4da74ad11"
        tags                               = {}
        # (18 unchanged attributes hidden)
    }

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

Note: You didn't use the -out option to save this plan, so Terraform can't
guarantee to take exactly these actions if you run "terraform apply" now.
```

هتلاقيه هنا قايلك ان حصل Change واحد بس

- ازاي ال Terraform بتبدأ ت Handle الموضوع ده كله دا هياخدنا ف ان فيه حاجه اسمها state file
-

State File

- ال Terraform لما تبدأ ت Run وتعمل اول Apply بيتكريت file اسمه terraform.tfstate وده اسمه ال State File وده عباره عن انه بيحفظ فيه ال Resources اللي هو عمل ليها Create علي ال AWS Account بتفاصيلها وال Values وبكل ال Attributes اللي بيحصلها Generations بعد اما ال Resource بيتعمله Create



- عندي اي Resource في الكلاود بيكون فيه معلومات كتير فيه بتتحدد بعد اما يتعمله Create فعليا مش قبلها زي ال ID بتاع ال VPC ف دي حاجه مش انت اللي بتحددها دي حاجه بتتحدد بعد اما يتعملها Create
- وكل مره انت بتعمل Run بيبدأ يقارن ويشوف ال Parameters اللي انت مباصيها في الكود ومابين الموجوده في ال terraform.tfstate هي ولا وبناءا عليه هياخد القرار ي اما هيعمل change ولا لا
- ف ال State File هو يعتبر هو حلقة الوصل اللي بتحقق ال checkاللي مابين الكود ومابين الواقع في ال AWS Account ف لو انت شلت حلقة الوصل دي اللي هي ال State File ف هيكون بالنسبه لل Terraform ان انت هتعمل Run للكود ل اول مره ف هتكون مشكله لو ال State File اتمسح ف لازم ناخد بالنا كويس جدا

State Management

- احنا لحد دلوقتي عرفنا ايه هو ال State File وعرفنا ان ال Terraform بتستخدمها علشان ت Manage موضوع ال State وتفهم ايه ال Resources اللي في الكود المعمول ليها Create وايه اللي مش معمول ليها Create وبعدين تبدأ تت check علي ال AWS Account
- ولكن زي ماحنا عارفين ان ال State File بيكون معمول ليه Save علي الجهاز بتاعك طب لو فيه اكتر من حد في التيم بيشتغل وكل واحد بيعدل فيه الكود ويرفعه علي GitHub ف هنا هيكون لكل واحد State File ف دي طبعا مشكله بالاضافه لو ال Folder اللي فيه ال Terraform اتمسح ف ساعتها قالوا ان ال State File ده لازم يكون محفوظ في مكان أمن ومن هنا جات فكره ال BackEnd ولو اخدت بالك وانت بتعمل init كان اول خطوه بتظهرلك انه بيقولك انه بيعمل Initialization لل backend وي By Default ال Backend بتاعت ال Terraform بتكون موجوده Local في Folder ال Terraform ولكن بنقدر نغير ده في اننا بنضيف Section ال Backend وسكشن ال Backend حاجه شبه ال Providers كده منفصله ملهاش علاقه ب Provider معين ولكن كل Provider بي Support الباك اند الخاص بيه
- لو انت دخلت علي الجزء الخاص بال Backend علي ال Documentation هتلاقي فيه انواع كتيره جدا تقدر تعملها علي Kubernetes او علي كذا Service مختلفه او Azure وهكذا
- ف احنا هنستخدم السكشن الخاص بال S3 bucket في ال AWS ف ده شكل ال Configuration

```
terraform {
  backend "s3" {
    bucket = "example-bucket"
    key    = "path/to/state"
    region = "us-east-1"
  }
}
```

- ف اللي احنا هنعمله هنعمل Create ل S3 Bucket علي ال AWS ف انا عملته باسم sprints-bootcamp-state-bucket1 وفي ال key في المكان اللي هيكون موجود فيه هحطه في ال terraform.tfstate ف اللي هيحصل ان ال state file بدل اما كان Local عندي علي الجهاز بقي موجود علي ال AWS

```
terraform {
  backend "s3" {
    bucket = "sprints-bootcamp-state-bucket1"
    key    = "terraform.tfstate"
    region = "us-east-1"
  }
}
```

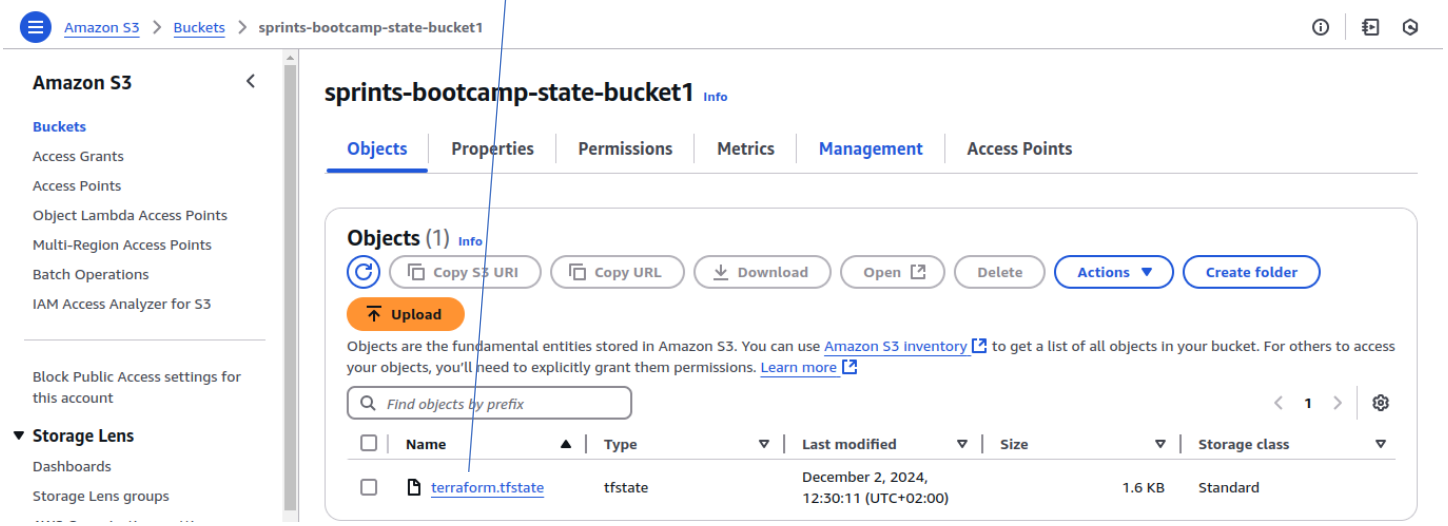
- لو جينا عملنا Run لل Plan هيطلعلنا Error بالشكل ده ليه طلعلنا Error علشان اي تغيير في ال Backend بيحتاج ان انت تغير ال Initializations لان زي ماحنا عارفين ان اول حاجه كان بيعملها كان بيعمل Initialize لل Backend

```
mohamed@MohamedAtef: ~/terraform$ terraform plan

Error: Backend initialization required, please run "terraform init"

Reason: Initial configuration of the requested backend "s3"

The "backend" is the interface that Terraform uses to store state,
perform operations, etc. If this message is showing up, it means that the
Terraform configuration you're using is using a custom configuration for
the Terraform backend.

Changes to backend configurations require reinitialization. This allows
Terraform to set up the new configuration, copy existing state, etc. Please run
"terraform init" with either the "-reconfigure" or "-migrate-state" flags to
use the current configuration.

If the change reason above is incorrect, please verify your configuration
hasn't changed and try again. At this point, no changes to your existing
configuration or state have been made.
```

- لو جينا عملنا تاني init هنلاقيه بيقولنا ان فيه State File كان في Backend قديمه عايز تنقله في Backend الجديده ولالا لو قولتله yes هياخده وينقله هناك في ال AWS علي ال S3

```
mohamed@MohamedAtef: ~/terraform$ terraform init
Initializing the backend...
Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the previous "local" backend to the
  newly configured "s3" backend. No existing state was found in the newly
  configured "s3" backend. Do you want to copy this state to the new "s3"
  backend? Enter "yes" to copy and "no" to start with an empty state.

  Enter a value: yes
Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.76.0

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.
```

- لو جينا دخلنا علي ال AWS علي ال S3 هنلاقيه ضاف ال State File

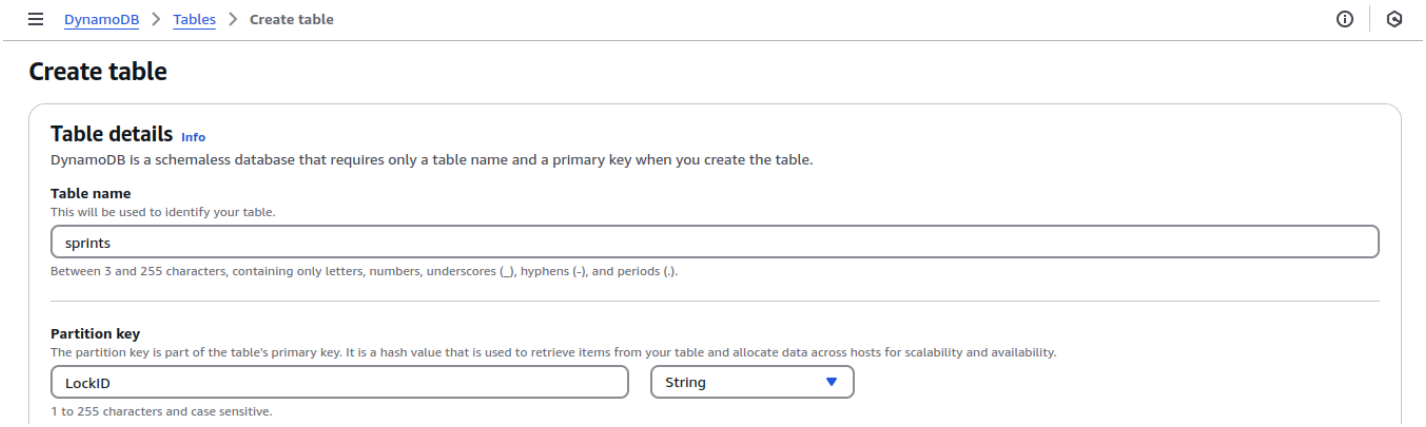


- ف انت بقي تقدر ان انت تمسح ال state file الموجود عندك Local بدون اي مشكله ف كل مره هتعمل Run للكود هيعمل Update علي ال state file الموجود علي ال AWS في ال S3

- فيه مشكله بسيطه ليها علاقه بالأ State وهي ان ممكن يكون شخص معاك في التيم يكون شغال معاك علي نفس الكود في نفس الوقت وشغالين من نفس ال State file وانتوا الاتنين بتعملوا Run في نفس الوقت ف ممكن انتوا الاتنين بتعملوا Update في نفس ال State File ف انت ممكن بتكون بتضيف Resource وهو مشفهوش فيحصل مشكله ف الحل ان احنا نشوف طريقه ان لو واحد فينا بيعمل Run مينفعش التاني ي Run في نفس الوقت
- ف ال Terraform اديتنا وسيله او طريقه نقدر ان احنا نحل المشكله دي وهي ال State Lock ان هي ت Lock ال State يعني بمجرد ان حد بيعمل Run لل State File يبدأ ال Terraform يمنع اي حد تاني انه يتعامل مع نفس ال State File لحد ما الشخص التاني يخلص
- عن طريق ان انت بتديله اوبشن dynamodb_table وده بيتحط في ال Section الخاص بال Backend وده اللي انا هستخدمه علشان ال State Lock والاوبشن ده لازم يكون جواه Primary Key وبيكون اسمه LockID وال Type بتاعه String
- وال dynamodb دي عباره عن Service موجوده في ال AWS نوع من انواع الداتا بيز بيتخزن فيها Key و Value

- `dynamodb_table` - (Optional) Name of DynamoDB Table to use for state locking and consistency. The table must have a partition key named `LockID` with type of `String`.

- ف احنا هنعمل Create ل DynamoDB Table علي AWS باسم sprints ولازم تحدد ال Partition key ف احنا هنستخدم ال LockID ولازم تكون بنفس الشكل



- بعد اما عملنا Create لل Table الخاص بال dynamodb_table وضفنا ال Partition key ف اللي هنعمله دلوقتي ان احنا هنضيف dynamodb_table في ال Section بتاع ال Backend بالشكل ده

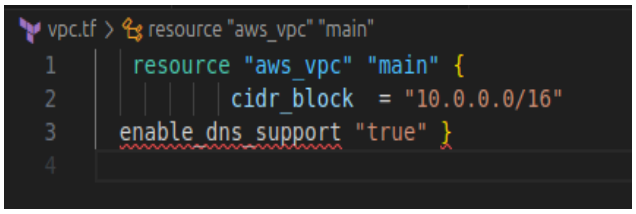
```
terraform {
 backend "s3" {
  bucket = "sprints-bootcamp-state-bucket1"
  key    = "terraform.tfstate"
  region = "us-east-1"
  dynamodb_table = "sprints"
 }
}
```

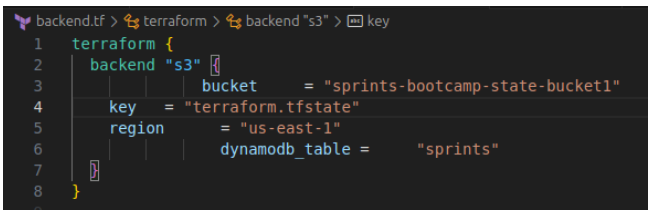
- لو جينا عملنا Run علي طول لل Terraform هيطلعلنا Error لان زي ماقولنا ان اي تغيير في ال Backend لازم تعمل terraform init تاني قبل اما تعمل Terraform apply
- ف دلوقتي لو فيه اي شخص بيعمل Run للكود وفيه حد بيعمل في نفس الوقت Run هيعمل Lock ليه وهيظهرلك بالشكل ده وهيقولك انك مش هينفع تعمل Run لان الشخص كذا بيعمل Run

```
mohamed@MohamedAtef: ~/terraform$ terraform apply
Acquiring state lock. This may take a few moments...

Error: Error acquiring the state lock
Error message: operation error DynamoDB: PutItem, https response error StatusCode: 400, RequestID:
VECM3QNBGON8BRBKB7VV4KQNSO5MVJF66Q9ASUAAJG, ConditionalCheckFailedException: The conditional request failed
Lock Info:
  ID:        f132b51-ec2557-f20-854a41-061564547952a
  Path:      sprints-bootcamp-state-bucket1/terraform.tfstate
  Operation: OperationTypeApply
  Who:       mohamed@MohamedAtef
  Version:   1.9.8
  Created:   2024-12-02 12:59:27.248983342 +0000 UTC
  Info:
Terraform acquires a state lock to protect the state from being written
by multiple users at the same time. Please resolve the issue above and try
again. For most commands, you can disable locking with the "-lock=false"
```

- **من ضمن ال Commands اللي بنستخدمها**

1- Command اسمه **fmt command** وال Command ده كل فايدته ببساطه انه بيخلي شكل الكود بتاعك منظم بمعني انت ممكن تكتب الكود بالشكل ده بدون اي تنظيم وفيه Spaces مابين ال Lines وبعضها مثلا او مابين اليساوي وهكذا ف ال fmt command هينظم الكود بتاعنا بالكامل

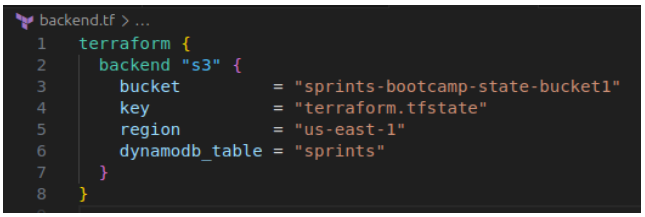




- ف لو عملنا Run لل fmt command هتلاقيه قايلك انه عمل تنظيم ل انهي file

```
mohamed@MohamedAtef: ~/terraform$ terraform fmt
backend.tf
vpc.tf
```

- ف لو جينا شوفنا شكل الكود تاني هنلاقيه بقي منظم جدااا



2- فيه عندي Command كمان اسمه **Destroy** ده احنا بنستخدمه لو احنا عايزين نمسح ال Infrastructure كلها عن طريق ان احنا بنستخدم terraform destroy اول اما تنفذ الامر ده هتلاقيه بيديك خطه كده ب ايه الحاجات اللي هتتمسح

```
mohamed@MohamedAtef: ~/terraform$ terraform destroy
# aws_vpc.main will be destroyed
Plan: 0 to add, 0 to change, 1 to destroy.

Do you 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

aws_vpc.main: Destroying... [id=vpc-0b349ad41fa8639de]
aws_vpc.main: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
```

Variables & Workspaces

- في ال Terraform او في ال Infrastructure as a code بشكل عام بيحاول يدينا نفس الافكار الموجوده في ال Coding افكار زي ال Variables
- ف احنا بنستخدم ال Variables علشان اكتر من شخص يقدر ياخده ويغير ال Parameters بشكل سهل منغير مايدخل يدور في الكود كله وده بنعمله باستخدام ال Variables
- ف علشان نستخدم ال Variable هنعمل file لل Variables باسم مثلا variables.tf وبنكتب جوه الملف ده ال Section الخاص بال Variables وليكن مثلا هنعمل variable لل cidr وبيكون بالشكل ده

```
variable "cidr" {
 type = string
}
```

- ف علشان نقدر نستخدم ال Variable ده هنعمل reference عن طريق ان احنا هنروح علي ال file الخاص بال VPC وبدل ما احنا كنا كاتبين في ال cidr_block ال 10.0.0.1 هنكتب var.cidr

```
resource "aws_vpc" "main" {
 cidr_block = var.cidr
 enable_dns_support = false
}
```

- احنا مدناش قيمه لل variable وفي عندنا اكتر من طريقه علشان نديله قيمه ال cidr

1- اول طريقه لو احنا عملنا Run للكود ومدنلهوش قيمه ال Variable هيطلب مننا ان احنا نديله قيمه ال Variable اللي اسمه cidr ف هو هيعتبره ك Input

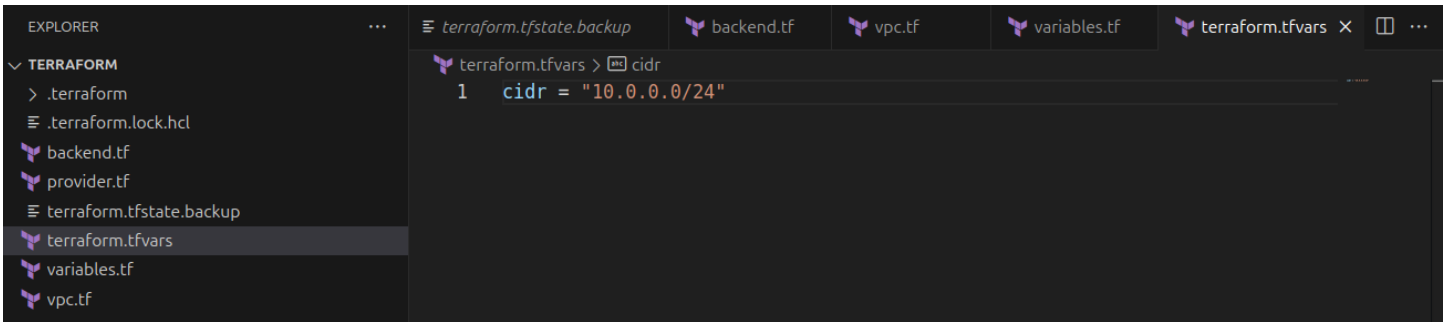
```
mohamed@MohamedAtef: ~/terraform$ terraform apply
var.cidr
  Enter a value: بكتب هنا بقي قيمه المتغير
```

- ودي مش افضل طريقه لان ممكن الكود يكون فيه اكتر من Variable ف لو استخدمت الطريقه دي هتدخل كل القيم بتاعت ال Variable ك Input والموضع هيكون صعب

2- فيه طريقه تانيه ممكن نستخدمها وهي ان احنا نعمل export ل environmental variable بالشكل ده ان احنا نقوله TF_VAR_ وبيكونوا Capital زي مامكتوب كده وبعدين نديله اسم ال Variable وبعدين يساوي وبعدين نديله القيمه اللي احنا عايزنها ف هيكون بالشكل ده TF_VAR_cidr=10.0.0.0/16

```
mohamed@MohamedAtef: ~/terraform$  export TF_VAR_cidr=10.0.0.0/16
```

3- فيه طريقه ابسط من كده عن طريق ان احنا بنستخدم حاجه اسمها ال Variables definition files او ال .tfvars وده بيكون عباره عن file ممكن تكتبه بالاسم ال Standard بتاعه وهو terraform.tfvars وبتيجي جوه ال file ده تكتب ال Variables بتاعتك زي مثلا cidr="10.0.0.0/16"



- ف لو عملت Run هيشتغل معاك بدون اي مشكله ومش شرط ان يكون اسم ال file يكون terraform.tfvars ولكن لو سميته اي اسم تاني وليكن مثلا dev.tfvars هتحتاج ان انت تكتبله الجزء ده --var-file وبعدين اسم ال file بالشكل ده

```
mohamed@MohamedAtef: ~/terraform$ terraform apply --var-file dev.tfvars
```

- فيه عندي مشكله هتحصل لو انا عندي اكتر من environment وعايز اعمل apply ليهم يعني لو عندي dev.tfvars بيحتوي علي cidr = "10.0.0.0/16" وعندي كمان prod.tfvars بيحتوي علي cidr="192.0.0.0/16" وكل واحد ليه القيم بتاعته ف لو احنا جينا عملنا Apply وادينالله var-file dev.tfvars -- هيروح يبني VPC بال cidr اللي هو 10.0.0.0 وهيعمل save لل State File في ال Path بتاع ال Backend ف لو انت جيت كمان شويه وعايز تعمل Create لل Production Environment ف هتديله var-file prod.tfvars -- ساعتها هو هيشوف ان انت شغال علي نفس ال State File ف كده هيعتبر كأنك رايح تعمل Destroy لل dev environment علشان ت build ال Production environment ف هو مش هيكون فاهم ان ان بتعمل Production لان انت شغال علي نفس ال State File
- ف علشان ن Manage موضوع ال Environment دي ونبقي قادرين نعمل Create ل اكتر من Environment ف هنستخدم حاجه اسمها Workspaces

Workspaces & Environments

- في ال Terraform الـ Workspaces هي طريقة لإدارة بيئات متعددة لمشروع واحد باستخدام نفس الكود بمعنى، بدل ما تكتب كود مختلف لكل environment زي "Development" و"Production"، ممكن تستخدم Workspaces لتفصل ال state لكل environment مع الاحتفاظ بنفس الكود. حسب انا واقف فين يعني لو انا واقف علي ال Workspace بتاعت ال Dev يبقي انا ب Manage ال State File بتاعت ال Dev ولو انا واقف علي ال Workspace بتاعت ال Prod يبقي ب Manage ال State File بتاعت ال Prod وهكذا
- لو انا عايز اعرف ال Workspace اللي انا واقف عليها حاليا هستخدم

```
mohamed@MohamedAtef: ~/terraform$ terraform workspace show
default
```

- لو انت عايز تعمل Create ل Workspace جديده وليكن هتعمل لل dev هتستخدم

```
mohamed@MohamedAtef: ~/terraform$ terraform workspace new dev
Created and switched to workspace "dev"!
You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
```

- وهنعمل كمان Workspace لل prod هنستخدم

```
mohamed@MohamedAtef: ~/terraform$ terraform workspace new prod
Created and switched to workspace "prod"!
You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
```

- لو احنا عايزين نعرض ال Workspaces اللي عندنا هنستخدم terraform workspace list وده هيعملي List وهيقولك انت واقف علي انهي Workspace هتلاقي جنب ال Prod علامه ال * star معناها انه انت كده علي ال Production

```
mohamed@MohamedAtef: ~/terraform$ terraform workspace list
  default
  dev
* prod
```

- ف لو عملنا apply لل Production هيعمل من غير اي مشكله لان انا واقف او مختار ال Prod في ال Workspace

```
mohamed@MohamedAtef: ~/terraform~$ terraform apply --var-file prod.tfvars
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.main will be created
  + resource "aws_vpc" "main" {
      + arn                                  = (known after apply)
      + cidr_block                           = "192.0.0.0/16"
      + 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_dns_hostnames                 = (known after apply)
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.
Do you want to perform these actions in workspace "prod"?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
  Enter a value: yes
aws_vpc.main: Creating...
aws_vpc.main: Creation complete after 3s [id=vpc-095094ff4905b9f3b]
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

- لو احنا عملنا apply لل dev من غير ماحنا نختار ال Workspace بتاعته هيطلعلنا error وهيقولنا انه هيضيف وهيعمل Destroy وده مش اللي احنا عايزينه

```
mohamed@MohamedAtef: ~/terraform~$ terraform apply --var-file dev.tfvars
aws_vpc.main: Refreshing state... [id=vpc-095094ff4905b9f3b]
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
with the following symbols:
-/+ destroy and then create replacement
Terraform will perform the following actions:
 # aws_vpc.main must be replaced
-/+ resource "aws_vpc" "main" {
    ~ arn                              = "arn:aws:ec2:us-east-1:853124477774:vpc/vpc-095094ff4905b9f3b" -> (known after apply)
    - assign_generated_ipv6_cidr_block = false -> null
    ~ cidr_block                       = "192.0.0.0/16" -> "10.0.0.0/16" # forces replacement
    ~ default_network_acl_id           = "acl-05a3da1323c57d08b" -> (known after apply)
    ~ default_route_table_id           = "rtb-0d7646e6b96db0717" -> (known after apply)
    ~ default_security_group_id        = "sg-09b0103eda77112e1" -> (known after apply)
    ~ dhcp_options_id                  = "dopt-0a87c9eb02d8f8b93" -> (known after apply)
    ~ enable_dns_hostnames             = false -> (known after apply)
    ~ enable_network_address_usage_metrics = false -> (known after apply)
    ~ id                               = "vpc-095094ff4905b9f3b" -> (known after apply)
```

- ف علشان نبدل لل Workspace اللي احنا عايزنها هنستخدم

```
mohamed@MohamedAtef: ~/terraform~$ terraform workspace select dev
Switched to workspace "dev".
```

output

- في Terraformالـ output بيستخدم عشان تعرض قيم معينة من الـ configuration بعد ما تنفذ أوامر زي apply. الفكرة منه إنه يسهل عليك الوصول لمعلومات مفيدة بدون الحاجة للبحث في تفاصيل الـ state file.
- ملف ال output بينكتب بالطريقه دي

```
output "<name>" {
  value = <expression>
}
```

- وليكن مثلا عايزين ال id بتاع ال vpc ف احنا هنقوله output وبعدين نديله اي اسم وليكن هسميه vpc_id وبعدين في ال value هنكتب ال resource اللي احنا عايزينه ف هنقوله aws_vpc.main.id

```
output " vpc_id "{
  value = aws_vpc.main.id
}
```

- ف لو جينا عملنا apply هنلاقيه في الاخر عارضلنا ال Output وعارض ال vpc_id

```
mohamed@MohamedAtef: ~/terraform~$ terraform apply --var-file prod.tfvars
aws_vpc.main: Refreshing state... [id=vpc-0f0ffacae9c83c2d3]
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.
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:
vpc_id = "vpc-0f0ffacae9c83c2d3"
```

- ال output بتبقي معمول ليها Save في Section جوه ال State file ف ده بيديك اماكنيه ان انت تقدر تقرئهم في اي وقت
- ف لو انت عايز تعرض ال output بعد التنفيذ هتستخدم

```
mohamed@MohamedAtef: ~/terraform~$ terraform output
vpc_id = "vpc-0f0ffacae9c83c2d3"
```

- ولو انت عايز تعرض output معين هتستخدم terraform output <output_name>

Provisioners

- في Terraform الـ Provisioners بتستخدم لتنفيذ أوامر أو سكربتات على resources بعد إنشائها سواء الاوامر دي او الاسكريبتات هنفذها Local او Remotely .
- يعني مثلا انا عايز بعد اما الكود بتاعي يخلص او بعد اما ال resource يتعمل ليه Create هيروح يقرأ معلومات منه ويكتبها في File
- أنواع الـ Provisioners:
  - local-exec: يشغل أوامر على الجهاز ال Local اللي بتنفذ منه Terraform.
  - remote-exec: بيشغل أوامر على ال resource نفسه باستخدام SSH أو WinRM.
  - file: ينقل ملفات من جهازك لل Resource.
- ال Section الخاص بال Provisioning بينكتب جوه ال resource وال Section ده مبيتنفذش الا لما ال resource بتتنفذ اول مره
- مثال مثلا علي ال local-exec Provisioner ان كل اللي بيعمله ان بعد اما يعمل Create لل instance يطبع رساله

```
resource "aws_instance" "example" {
  ami           = "ami-12345678"
  instance_type = "t2.micro"

  provisioner "local-exec" {
    command = "echo Instance created with ID ${self.id}"
  }
}
```

- مثال علي remote-exec Provisioner

```
resource "aws_instance" "example" {
  ami           = "ami-12345678"
  instance_type = "t2.micro"
  connection {
    type     = "ssh"
    user     = "ubuntu"
    private_key = file("~/.ssh/id_rsa")
    host     = self.public_ip
  }
  provisioner "remote-exec" {
    inline = [
      "sudo apt update",
      "sudo apt install -y nginx"
    ]
  }
}
```

- مثال علي ال file Provisioner ده بينقل ملف من جهازك للسيرفر

```
resource "aws_instance" "example" {
  ami           = "ami-12345678"
  instance_type = "t2.micro"

  connection {
    type     = "ssh"
    user     = "ubuntu"
    private_key = file("~/.ssh/id_rsa")
    host     = self.public_ip
  }

  provisioner "file" {
    source      = "my-script.sh"
    destination = "/tmp/my-script.sh"
  }
}
```

لا تنسونا من صالح الدعاء

HashiCorp

Terraform

BY: Mohamed Atef Elbitawy

https://www.linkedin.com/in/mohamedelbitawy/