

Automation Terraform

Detail Materi



Konsep dan Cara Kerja Terraform

Indikator :
Melakukan Automation Infrastructure As a
Code menggunakan Terraform



Komponen dan Konfigurasi
Pada Terraform



Desain dan Build Infrastructure
dengan Terraform



Provisioning Pada Instance
yang di Build dengan Terraform

Modul Sekolah DevOps Cilsy

Hak Cipta © 2019 **PT. Cilsy Fiolution Indonesia**

Hak Cipta dilindungi Undang-Undang. Dilarang memperbanyak atau memindahkan sebagian atau seluruh isi buku ini dalam bentuk apapun, baik secara elektronik maupun mekanis, termasuk mecropy, merekam atau dengan sistem penyimpanan lainnya, tanpa izin tertulis dari Penulis dan Penerbit.

Penulis : Adi Saputra, Irfan Herfiandana & Tresna Widiyaman

Editor: Rizal Rahman & Tresna Widiyaman

Revisi Batch 2

Penerbit : **PT. Cilsy Fiolution Indonesia**

Web Site : <https://cilsyfiolution.com> , <https://devops.cilsy.id>

Sanksi Pelanggaran Pasal 113 Undang-undang Nomor 28 Tahun 2014 tentang Hak Cipta

1. Setiap orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam pasal 9 ayat (1) huruf i untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan atau pidana denda paling banyak Rp100.000.000,00 (seratus juta rupiah).
2. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak cipta melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf c, huruf d, huruf f, dan atau huruf h, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah)
3. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf a, huruf b, huruf e, dan atau huruf g, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan atau pidana denda paling banyak Rp1.000.000.000,00 (satu miliar rupiah)
4. Setiap orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan atau pidana denda paling banyak Rp4.000.000.000,00 (empat miliar rupiah)



Daftar Isi

Cover.....	1
12. Automation Terraform.....	5
Learning Outcomes.....	5
Outline Materi.....	5
12.1. Terraform.....	6
12.1.1. Pengenalan Terraform.....	6
12.1.2. Fungsi dan Kegunaan Terraform.....	7
12.2. Roadmap Pembelajaran.....	8
12.3. Setup dan Instalasi Terraform.....	9
12.3.1. Instalasi Terraform.....	9
12.3.1.1. Linux.....	9
12.3.1.2. Binary Package.....	9
12.3.1.3. Windows.....	10
12.3.1.4. Mac.....	10
12.3.2. Verify Installation.....	10
12.3.3. Download Konfigurasi Terraform.....	11
12.3.4. Setup Credential AWS.....	11
12.3.5. Membuat Key Pair.....	12
12.3.6. Membuat Security Group (optional).....	13
12.4. Struktur File folder pada Terraform.....	14
12.4.1. Struktur Folder Terraform.....	14
12.4.2. File dan Penjelasan fungsinya.....	15
12.4.2.1. File main.tf.....	15
12.4.2.2. File Variable.....	20
12.4.3. Exercise.....	28
12.5. Konfigurasi Terraform.....	28
12.5.1. Konfigurasi EC2.....	28
12.5.1.1. Membuat EC2.....	28
12.5.1.2. <i>Manage</i> EC2.....	32
12.5.1.3. Menghapus EC2.....	35



12.5.2. Konfigurasi S3.....	36
12.5.2.1. Membuat Bucket S3.....	36
12.5.2.2. Manage Bucket S3.....	39
12.5.2.3. Menghapus Bucket S3.....	40
12.5.3. Konfigurasi Amazon RDS.....	41
12.5.3.1. Membuat Database RDS.....	41
12.5.3.2. Menghapus database RDS.....	44
12.5.4. Exercise.....	44
12.6. Create EC2 dengan terraform (Multiple Server).....	44
12.6.1. Exercise.....	46
12.7. Summary.....	47

12.

Automation Terraform

Learning Outcomes

Setelah selesai mempelajari bab ini, peserta mampu :

1. Memahami Konsep dan Cara Kerja Terraform
2. Memahami Komponen dan Konfigurasi Pada Terraform
3. Melakukan Desain dan Build Infrastructure dengan Terraform
4. Melakukan Provisioning Pada Instance yang di Build dengan Terraform

Outline Materi

1. Pengenalan Terraform
2. Setup dan Instalasi
3. Struktur File dan Folder
4. Konfigurasi Terraform
5. Terraform Multi Server
6. Provisioning Terraform



12.1. Terraform

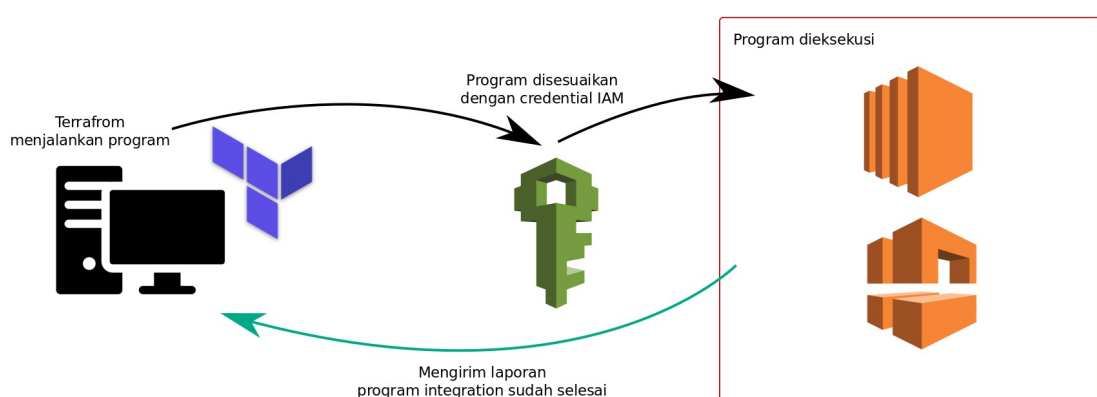
12.1.1. Pengenalan Terraform

Terraform adalah **Infrastructure as Code** dan salah satu tool Automation pada AWS. Infrastructure as Code adalah proses penyediaan IT infrastruktur dimana sistem dibangun dan dikelola melalui kode secara automasi (otomatis), bukan secara manual. Atau bisa disebut juga bahasa kerennya **Programmable Infrastructure**.



Logo Terraform

Dengan menggunakan kode dan meng-automasi. Proses setting dan konfigurasi baremetal, virtual mesin, cloud computing baik itu instalasi baru atau perubahan konfigurasi dapat dilakukan secara cepat, mudah, dan berulang. Selain itu bermanfaat sebagai dokumentasi juga, jadi siapapun akan tahu konfigurasi server, kebutuhan aplikasi server dan sebagainya.



Ilustrasi alur kerja terraform

Gambar diatas merupakan salah satu ilustrasi dari sistem automasi terrafrom, disana ketika program yang sudah kita buat dijalankan, dia akan mencoba



untuk mengecek identitas credential IAM yang kita masukan. Setelah tervalidasi, baru program akan mengeksekusi untuk membuat sebuah EC2 baru di AWS. Setelah proses pembuatan selesai, maka reportnya akan dikirimkan kembali kepada alamat awal si program, sehingga kita bisa tau proses tersebut error atau berhasil.

Selain AWS, ada juga beberapa provider yang support untuk menggunakan automasi dari terraform. Provider yang disupport Terraform adalah sebagai berikut :

- AWS, GCE, Azure, OpenStack, DigitalOcean, Docker, CloudStack, Heroku, vSphere, vCloud (Cloud Infrastructure)
- Chef, Rundeck (Configuration Management)
- CloudFlare, DNSMadeEasy, Dyn, DNSimple (DNS provider)
- Mailgun (Email)
- Atlas (Hashicorp workflow engine)
- Consul, PowerDNS (DNS and service registry)
- MySQL, PostgreSQL (Database)
- StatusCake (Monitoring)

12.1.2. Fungsi dan Kegunaan Terraform

1. Infrastruktur sebagai Kode

Infrastruktur dijelaskan menggunakan sintaks konfigurasi tingkat tinggi. Hal ini memungkinkan cetak biru pusat data kita akan diversi dan diperlakukan seperti yang kita lakukan pada kode lainnya. Selain itu, infrastruktur dapat dibagi dan digunakan kembali.

2. Automation Plan

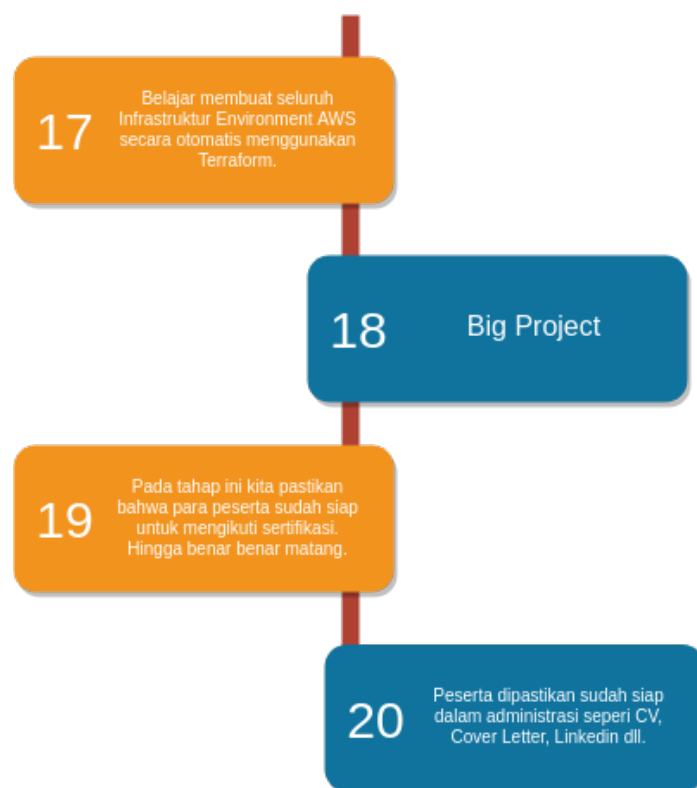
Terraform memiliki langkah "perencanaan" di mana ia menghasilkan rencana eksekusi. Rencana eksekusi menunjukkan apa yang akan dilakukan Terraform saat kita memanggil program. Ini memungkinkan kita menghindari kejutan ketika Terraform memanipulasi infrastruktur.



3. Automation Change

Perubahan yang rumit dapat diterapkan pada infrastruktur kita dengan interaksi manusia yang sangat minim. Dengan rencana eksekusi dan grafik sumber daya yang disebutkan sebelumnya, kita akan tahu persis apa yang akan diubah dan menghindari banyak kemungkinan *human error*.

12.2. Roadmap Pembelajaran



Roadmap Pembelajaran

Pada pembahasan ini kita sudah memasuki pada tahapan ke 17, dimana kita akan membuat infrastruktur yang ada di AWS dibuat secara otomatis menggunakan terraform. Jadi apabila kita ingin membuat infrastruktur, kita tidak perlu ribet untuk melakukan konfigurasi step by step. Kita hanya perlu menjalankan sebuah script untuk membuild nya.



Tahap ini merupakan tahap terakhir pada materi devops ini, tahap dimana kita membuild semua bahan yang kita siapkan untuk sebuah aplikasi secara otomatis.

12.3. Setup dan Instalasi Terraform

12.3.1. Instalasi Terraform

Terraform tersedia pada berbagai platform mulai dari Windows, Linux dan Mac. Setiap platform juga memiliki caranya sendiri untuk menginstallkan Terraform. Untungnya, semua sudah ada dokumentasinya yang bisa di lihat [di sini](#).

Pada saat modul ini dibuat, Terraform memiliki versi stable di versi 1.0.1. Berikut beberapa cara untuk menginstallkan Terraform pada mesin kita.

12.3.1.1. Linux

Untuk menginstall Terraform pada Linux, kita cukup menambahkan repository Terraform dan menginstallkannya menggunakan package manager pada distro yang dipakai.

```
sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $
(lsb_release -cs) main"
sudo apt-get update && sudo apt-get install terraform
```

12.3.1.2. Binary Package

Pada tahap ini kita akan menginstallkan terraform pada komputer Linux kita, cara installasi terraform agak berbeda dengan installasi pada biasanya yang menggunakan **apt-get**. Aplikasi terraform berbentuk file program yang bisa kita eksekusi.

Hal pertama yang harus dilakukan adalah update repo dan install aplikasi zip, unzip dan wget.

```
$ sudo apt-get update
$ sudo apt-get install zip unzip wget
```



Setelah itu pindah directory ke root, lalu download terraform dan extrack dengan unzip.

```
$ cd ~
$ wget
https://releases.hashicorp.com/terraform/1.0.1/terraform_1.0.1_linux_386.zip
unzip terraform_1.0.1_linux_386.zip
```

Berikutnya pindahkan file terraform tersebut ke /usr/local/bin agar bisa kita akses.

```
sudo mv terraform /usr/local/bin
```

12.3.1.3. Windows

[Chocolatey](#) adalah sistem package management open source dan gratis untuk Windows. Instal paket Terraform dari baris perintah.

```
choco install terraform
```

12.3.1.4. Mac

Homebrew adalah sistem package management open source dan gratis untuk Mac OS X. Instal formula resmi Terraform dari terminal.

Pertama, instal tap HashiCorp, tempat penyimpanan semua paket Homebrew kami.

```
brew tap hashicorp/tap
```

Sekarang, instal Terraform dengan hashicorp/tap/terraform.

```
brew install hashicorp/tap/terraform
```

12.3.2. Verify Installation

Ketikan **terraform** di terminal, maka akan muncul seperi dibawah ini.



```
taufik@hewlettpackard:~$ terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init      Prepare your working directory for other commands
  validate  Check whether the configuration is valid
  plan      Show changes required by the current configuration
  apply     Create or update infrastructure
  destroy   Destroy previously-created infrastructure

All other commands:
  console    Try Terraform expressions at an interactive command prompt
  fmt        Reformat your configuration in the standard style
  force-unlock Release a stuck lock on the current workspace
  get        Install or upgrade remote Terraform modules
  graph      Generate a Graphviz graph of the steps in an operation
  import     Associate existing infrastructure with a Terraform resource
  login      Obtain and save credentials for a remote host
  logout     Remove locally-stored credentials for a remote host
```

Hasil Instalasi Terraform

12.3.3. Download Konfigurasi Terraform

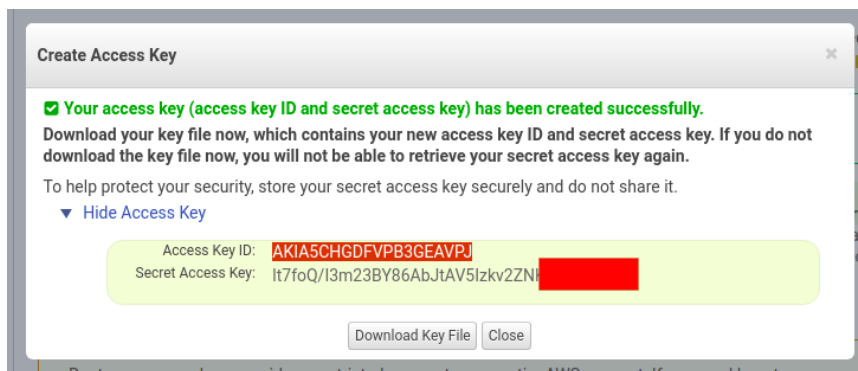
Ada beberapa konfigurasi terraform yang sudah didesain secara best practice dan akan dibahas dibagian selanjutnya, maka dari itu kita harus mendownloadnya terlebih dahulu. Berikut merupakan link nya.

```
git clone https://github.com/sdcilisy/terraform-aws.git
```

12.3.4. Setup Credential AWS

Pada bagian ini kita akan mencoba untuk mengakses AWS menggunakan terraform , hal yang harus di siapkan adalah credential yang berguna untuk autentikasi ke AWS. Untuk membuat credential sudah kita bahas pada bab sebelumnya, harusnya kalian sudah dapat membuat sebuah credential yang siap untuk digunakan.





Berikut adalah **Access key ID** dan **Secret access key** seperti gambar di atas, selanjutnya kita simpan atau catat key tersebut yang nantinya akan kita gunakan pada terraform untuk mengakses akun AWS.

Kita juga bisa menggunakan **AWS CLI** sebagai pengganti input `access_key` dan `secret_key` pada block **provider aws**, jadi kita tidak menggunakan variable **aws_access_key** pada `variable.tf` begitu juga dengan private key.

```
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform$ aws configure
AWS Access Key ID [*****AVPJ]:
AWS Secret Access Key [*****7sYG]:
Default region name [us-east-2]:
Default output format [json]:
```

12.3.5. Membuat Key Pair

Kita membutuhkan key pair untuk mengakses instance yang akan dibuat. Jika kita tidak mendeklarasikan **key_name** pada **resource aws intance**, kita tidak akan dibuatkan key pair oleh Terraform.

Maka dari itu, kita bisa membuat private key terlebih dahulu secara manual atau dengan resource [aws_key_pair](#).

Pada praktikum ini, saya menggunakan key-pair dengan nama **taufik_moduldo** yang sudah disimpan di local machine.

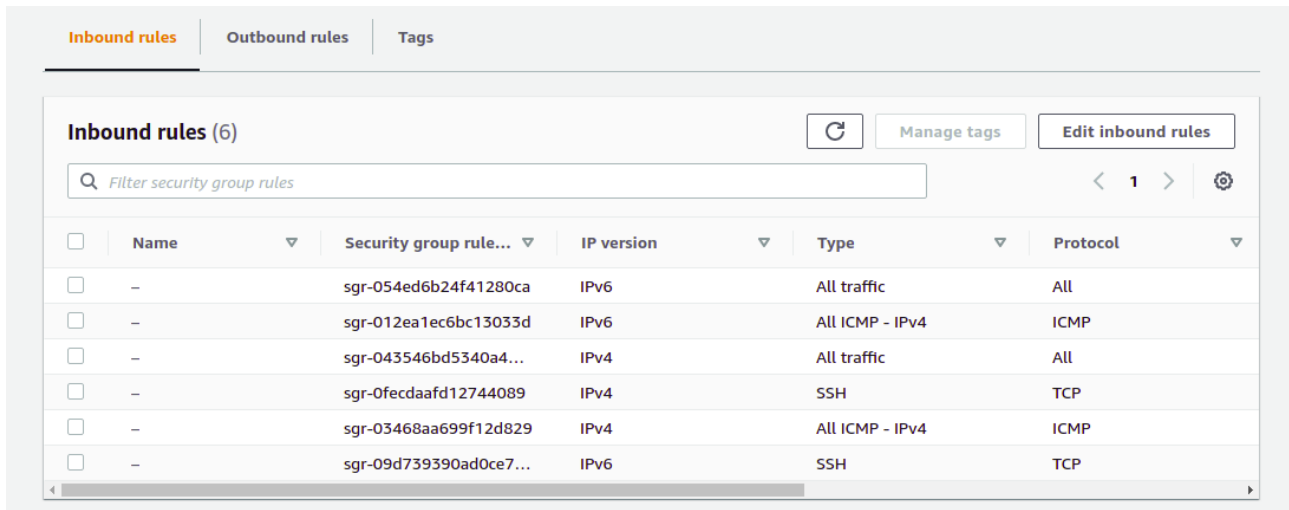


12.3.6. Membuat Security Group (optional)

Pada kasus tertentu, kita membutuhkan instance kita untuk dapat menerima traffic custom. Oleh karena itu, kita bisa saja membuat Security Group untuk instance kita.

Jika kita tidak membuat Security Group, maka terraform akan menggunakan default security group yang ada pada vpc.

Pada praktikum ini, saya menggunakan Security Group dengan id **sg-0d08bb22687b02c67** dengan rule sebagai berikut.



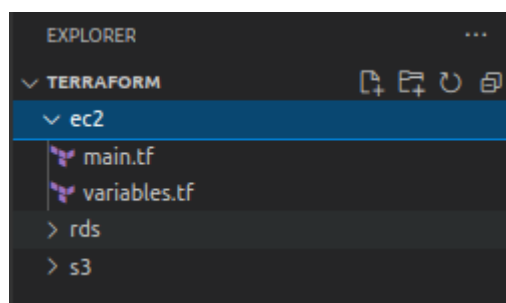
	Name	Security group rule...	IP version	Type	Protocol
<input type="checkbox"/>	-	sg-054ed6b24f41280ca	IPv6	All traffic	All
<input type="checkbox"/>	-	sg-012ea1ec6bc13033d	IPv6	All ICMP - IPv4	ICMP
<input type="checkbox"/>	-	sg-043546bd5340a4...	IPv4	All traffic	All
<input type="checkbox"/>	-	sg-0fecdaafd12744089	IPv4	SSH	TCP
<input type="checkbox"/>	-	sg-03468aa699f12d829	IPv4	All ICMP - IPv4	ICMP
<input type="checkbox"/>	-	sg-09d739390ad0ce7...	IPv6	SSH	TCP



12.4. Struktur File folder pada Terraform

12.4.1. Struktur Folder Terraform

Untuk memahami terraform sebelumnya, maka kita harus memahami struktur dari file atau terraform itu sendiri. Untuk kali ini struktur terraform di buat sebagai modul-modul yang dapat di gunakan berkali-kali dalam terraform. Berikut contoh atau gambar struktur terraform.



Ilustrasi Struktur folder terraform beserta isinya

Pada gambar di atas dapat kita lihat ada folder utama yaitu terraform, selanjutnya ada sub folder **ec2** dengan file **main.tf** dan **variables.tf** dengan fungsi sebagai berikut :

1. **main.tf** : konfigurasi utama yang berisikan parameter untuk membuat ec2. Disini ada beberapa parameter yang dibiarkan default. Untuk lebih lengkapnya, silahkan baca dokumentasi aws intance provider aws [disini](#).
2. **variables.tf** : berisi variable variable yang digunakan pada main.tf. Kita bisa memanggilnya dengan syntax **var.(nama variable)**.

Pada terraform, ada yang dinamakan module. Module ini merupakan kumpulan file dengan ekstensi .tf yang berada dalam 1 directory. Berarti **main.tf** dan **variables.tf** merupakan 1 module didalam directory **ec2**.

Dengan adanya module ini, kita tidak perlu mendefinisikan mana **variables.tf** pada **main.tf** karena Terraform mengevaluasi semua file konfigurasi dalam sebuah module, secara efektif memperlakukan seluruh module sebagai satu



dokumen. Memisahkan berbagai blok ke dalam file yang berbeda adalah murni untuk kenyamanan pembaca dan pengelola, dan tidak berpengaruh pada perilaku modul.

12.4.2. File dan Penjelasan fungsinya

12.4.2.1. File *main.tf*

Berikut akan kita jabarkan isi dari file **main.tf** yang berada pada direktori **module/ec2/**

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
    }
  }
  required_version = ">= 0.14.9"
}

provider "aws" {
  access_key = var.aws_access_key
  secret_key = var.aws_secret_key
  profile = "default"
  region = var.region
}

resource "aws_instance" "devops" {
  ami = var.ami
  instance_type = var.instance_type
  key_name = var.key_name
  vpc_security_group_ids = var.vpc_security_group_ids
  associate_public_ip_address = var.associate_public_ip_address
  count = var.instance_count

  root_block_device {
```



```

    volume_type          = var.volume_type
    volume_size          = var.root_volume_size
    delete_on_termination = var.delete_on_termination
  }

  ebs_block_device {
    device_name = "/dev/sdb"
    volume_size = var.volume_size
    volume_type = var.volume_type
  }

  tags = {
    Name = var.tags["name"]
    Purpose = var.tags["purpose"]
    Env = var.tags["env"]
  }

  volume_tags = {
    Name = var.tags["name"]
    Purpose = var.tags["purpose"]
    Env = var.tags["env"]
  }

  provisioner "remote-exec" {
    inline = [
      "sudo apt update && sudo apt upgrade -y",
      "sudo apt install -y nginx"
    ]

    connection {
      host = self.public_ip
      type = "ssh"
      user = "ubuntu"

      private_key = "${file("/home/taufik/Documents/kerja/cilsy/modul/Sekolah
Devops/key-pair/taufik_moduldo.pem")}"
    }
  }

```




```
}
}
}
```

File di atas merupakan file lengkap dari **main.tf**, selanjutnya akan kita coba jabarkan secara detail isi dari file tersebut agar kita memahami isi dari script yang ada pada file tersebut.

- Block dibawah mendefinisikan provider yang kita akan gunakan yaitu AWS dari hashicorp sendiri dan versi Terraform yang digunakan. Selengkapnya, provider bisa dilihat [disini](#).

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
    }
  }
  required_version = ">= 0.14.9"
}
```

- Block provider aws ini mendeklarasikan access key untuk membuat resource melalui Terraform pada region yang telah ditentukan

```
provider "aws" {
  access_key = var.aws_access_key
  secret_key = var.aws_secret_key
  profile = "default"
  region = var.region
}
```

- Perintah dibawah berguna untuk memanggil fungsi untuk membuat atau create, mengubah atau change , dan juga hapus atau destroy **EC2**. Ada banyak resource yang bisa kita manage, dokumentasi lengkapnya [disini](#).

```
resource "aws_instance" "devops" {
```

- Perintah dibawah berguna untuk memanggil atau deklarasi dari ami yang akan di gunakan. Misal dengan ubuntu dan centos dan lain-lain. Pada



praktikum ini, AMI yang digunakan adalah Ubuntu Server 18.04 LTS (HVM).

```
ami = var.ami
```

- Perintah dibawah berguna untuk memilih type instance yang akan di gunakan.

```
instance_type = var.instance_type
```

- Perintah dibawah berguna untuk file key-pair yang akan di gunakan. Pada bagian setup, sudah disebutkan disini saya menggunakan key-pair yang sudah dibuat sebelumnya.

```
key_name = var.key_name
```

- Perintah dibawah berguna untuk memanggil dan memilih security group yang akan di gunakan. Pada bagian setup, sudah disebutkan disini saya menggunakan security group yang sudah dibuat sebelumnya.

```
vpc_security_group_ids = var.vpc_security_group_ids
```

- Perintah dibawah berguna untuk memanggil dan memilih ip publik akan di gunakan.

```
associate_public_ip_address = var.associate_public_ip_address
```

- Perintah dibawah berguna untuk memanggil dan membuat EC2 sebanyak kita ingin kan.

```
count = var.instance_count
```

- Pada perintah dibawah berguna untuk mengatur hardisk yang ingin digunakan, untuk hardisk ada 2 macam, pertama adalah **root block** seperti dibawah, selanjutnya adalah **EBS**.

```
root_block_device {
  volume_type      = var.volume_type
  volume_size      = var.root_volume_size
  delete_on_termination = var.delete_on_termination
}
```



- Seperti **root block**, **EBS** berguna sebagai hardisk yang akan di gunakan oleh EC2. Beda nya dengan root block, EBS adalah sebagai tempat untuk folder-folder home, opt dan lain-lain dari linux di dalam EC2.

```
ebs_block_device {
  device_name = "/dev/sdb"
  volume_size = var.volume_size
  volume_type = var.volume_type
}
```

- Selanjutnya adalah perintah tags. Seperti di perintah dibawah, perintah ini berguna untuk mengatur Nama yang akan di gunakan, hal yang di perlukan sebenarnya adalah Name saja, namun yang lain seperti Purpose atau Env merupakan tambahan saja.

```
tags = {
  Name = var.tags["name"]
  Purpose = var.tags["purpose"]
  Env = var.tags["env"]
}
```

- Selanjutnya selain pada EC2 kita juga dapat mengatur untuk volume atau EBS yang kita buat, sama seperti dengan perintah tags pada EC2 di atas.

```
volume_tags = {
  Name = var.tags["name"]
  Purpose = var.tags["purpose"]
  Env = var.tags["env"]
}
```

- Terdapat perintah yang terakhir pada file **main.tf** yaitu adalah **provisioner**, fungsi provisioner berguna untuk provisioning, atau untuk menginstall paket-paket atau aplikasi pada linux yang berada di EC2.

Pada perintah dibawah kita dapat mengatur paket installasi yang akan kita installkan seperti contoh dibawah yaitu update repository, install **nginx**,



Pada connection kita menggunakan ip public instance via **ssh** dengan user **ubuntu**, sedangkan private key dapat kita sesuaikan dengan lokasi file tersebut berada.

```
provisioner "remote-exec" {
  inline = [
    "sudo apt update && sudo apt upgrade -y",
    "sudo apt install -y nginx"
  ]

  connection {
    host = self.public_ip
    type = "ssh"
    user = "ubuntu"
    private_key = "${file("/home/taufik/Documents/kerja/cilasy/modul/Sekolah
Devops/key-pair/taufik_moduldo.pem")}"
  }
}
```

12.4.2.2. File Variable

Berikut akan kita jabarkan isi dari file variable.tf yang berada pada direktori **module/ec2/**

```
variable "aws_access_key" {
  default = "AKIA5CHGDFVPB3GEAVPJ"
}

variable "aws_secret_key" {
  default = "lt7foQ/I3m23BY86AbJtAV5Izkv2ZN*****"
}

variable "region" {
  default = "us-east-2"
}
```



```
variable "availability_zone" {
    default = "us-east-2a"
}

variable "ami" {
    default = "ami-0b9064170e32bde34"
}

variable "instance_type" {
    default = "t2.micro"
}

variable "root_volume_size" {
    default = 8
}

variable "instance_count" {
    default = 1
}

variable "delete_on_termination" {
    default = true
}

variable "volume_size" {
    default = 20
}

variable "volume_type" {
    default = "gp2"
}

variable "key_name" {
    default = "taufik_moduldo"
}
```



```
variable "vpc_security_group_ids" {
    default = ["sg-0d08bb22687b02c67"]
}

variable "associate_public_ip_address" {
    default = true
}

variable "tags" {
    type = map(string)
    default = {
        "name" = "sekolah-devops-instance"
        "purpose" = "praktikum-sekolah-devops"
        "env" = "dev"
    }
}
```

Perintah di atas merupakan file yang berasal dari **variable.tf**. File ini berisikan variable atau parameter yang akan di gunakan. Pada file ini berguna untuk mendefinisikan variable secara default. Berikut akan kita jelaskan setiap bagian dari isi file **variable.tf** tersebut.

- Pada perintah dibawah berguna untuk mendefinisikan aws access key dan secret key yang akan di gunakan pada **main.tf**. Pada bagian sebelumnya sudah dijelaskan bagaimana cara mendapatkannya.

```
variable "aws_access_key" {
    default = "AKIA5CHGDFVPB3GEAVPJ"
}

variable "aws_secret_key" {
    default = "lt7foQ/I3m23BY86AbJtAV5Izkv2ZN*****"
}
```

- Selanjutnya adalah variable untuk mengatur **region**. Pada praktikum ini, region yang digunakan adalah **us-east-2** atau region Ohio.



```
variable "region" {
  default = "us-east-2"
}
```

- Untuk daftar code region sendiri dapat kita lihat seperti tabel dibawah ini atau pada [link berikut](#).

Code	Name
us-east-1	US East (N. Virginia)
us-east-2	US East (Ohio)
us-west-1	US West (N. California)
us-west-2	US West (Oregon)
ca-central-1	Canada (Central)
eu-central-1	EU (Frankfurt)
eu-west-1	EU (Ireland)
eu-west-2	EU (London)
eu-west-3	EU (Paris)
ap-northeast-1	Asia Pacific (Tokyo)
ap-northeast-2	Asia Pacific (Seoul)
ap-northeast-3	Asia Pacific (Osaka-Local)
ap-southeast-1	Asia Pacific (Singapore)
ap-southeast-2	Asia Pacific (Sydney)
ap-south-1	Asia Pacific (Mumbai)
sa-east-1	South America (São Paulo)

Daftar code region AWS

- Untuk zone Ohio kita dapat menggunakan us-east-2a atau us-east-2b seperti perintah dibawah. Availability zone dapat dilihat [disini](#).

```
variable "availability_zone" {
  default = "us-east-2a"
}
```

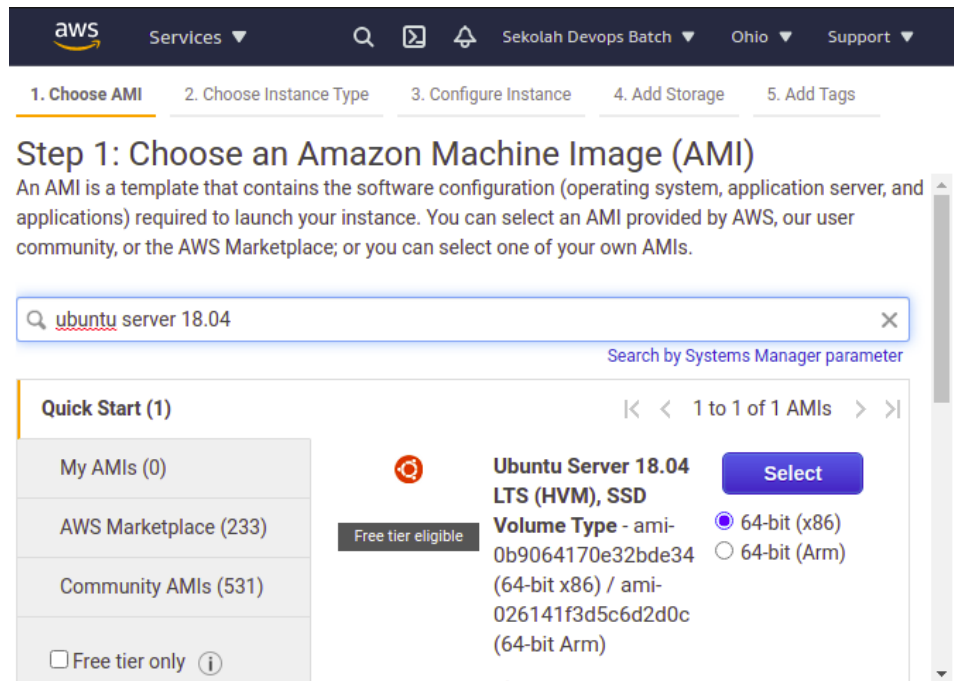
Perintah dibawah untuk mendefinisikan ami atau distro iso yang di gunakan, kali ini kita dapat menggunakan ami dari distro ubuntu 18.04.

```
variable "ami" {
  default = "ami-0b9064170e32bde34"
}
```

- Untuk melihat ami-ami yang lain kita dapat melihat saat kita lauch EC2 seperti di bawah ini. Bisa kita lihat code AMI pada Ubuntu Server 18.04



LTS adalah **ami-0b9064170e32bde34**. Perlu diingat, ami pada setiap region berbeda meskipun dengan nama yang sama.



Code AMI Image AWS

- Perintah dibawah berguna untuk memberikan nilai default variable yaitu t2.micro, t2.micro merupakan type pada EC2 AWS, semakin tinggi type maka spesifikasi server instance akan semakin tinggi

```
variable "instance_type" {
    default = "t2.micro"
}
```

- Untuk pilihan type instance yang lainnya, kita dapat melihat pada gambar dibawah ini maupun pada link berikut <https://aws.amazon.com/id/ec2/instance-types/>
- Pada perintah dibawah memberikan atau mendefinisikan variable default untuk hardisk root storage adalah 8 GB.

```
variable "root_volume_size" {
    default = 8
}
```




```
}
```

- Pada perintah dibawah memberikan atau mendefinisikan variable default jumlah instance yang ingin di buat adalah 1 server instance.

```
variable "instance_count" {
    default = 1
}
```

- Pada perintah dibawah memberikan atau mendefinisikan variable default true saat menghapus EC2, maka hardisk yang sebelumnya telah di buat akan di hapus juga.

```
variable "delete_on_termination" {
    default = true
}
```

- Pada perintah dibawah memberikan atau mendefinisikan variable default untuk hardisk EBS adalah 20 GB.

```
variable "volume_size" {
    default = 20
}
```

- Pada perintah dibawah memberikan atau mendefinisikan variable default untuk hardisk type adalah gp2.

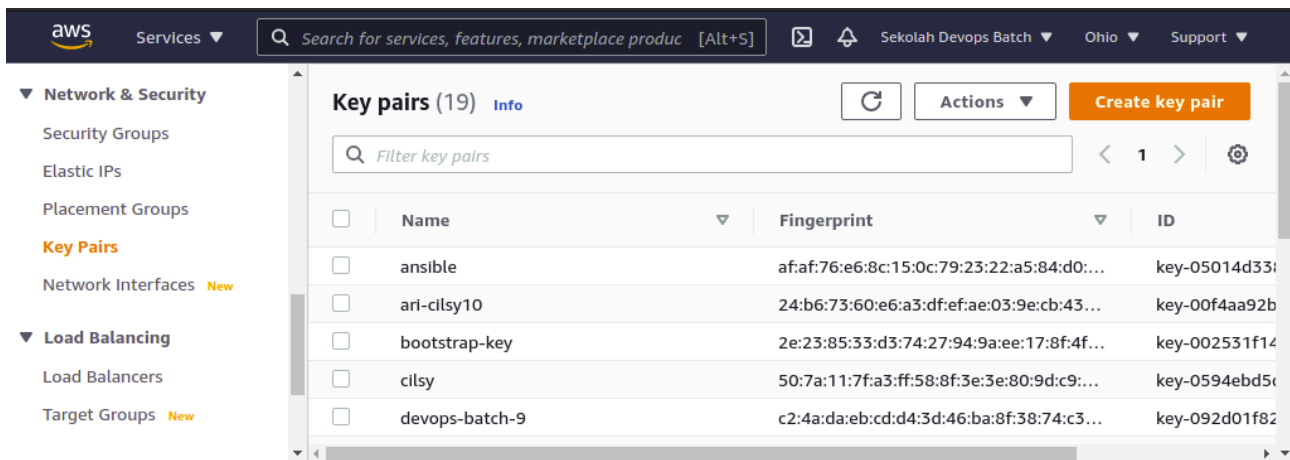
```
variable "volume_type" {
    default = "gp2"
}
```

- Pada perintah dibawah memberikan atau mendefinisikan variable default untuk key_name adalah "taufik_moduldo".

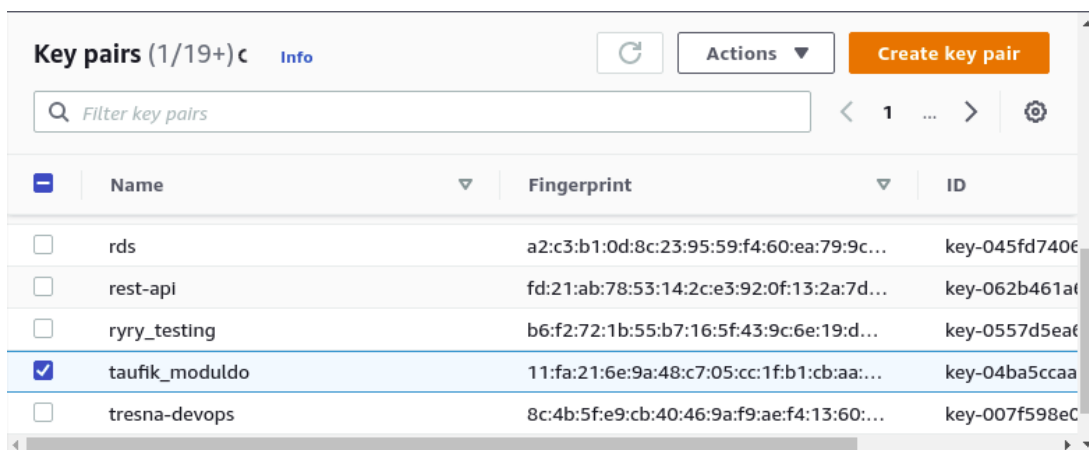
```
variable "key_name" {
    default = "taufik_moduldo"
}
```

key_name dapat di buat pada menu key pairs pada halaman Dashboard EC2.





Ketika kita sudah membuat key pair, akan muncul di dashboard ini.



- Daftar key pair yang ada dan dapat kita gunakan adalah key yang kita miliki file .pem nya. File .pem di gunakan sebagai password untuk remote ssh pada instance yang sudah kita buat.

```
ssh -i <key_pair> ubuntu@ip_public
```

Kita dapat juga untuk membuat key baru dengan cara **create keypair** pada menu tersebut.

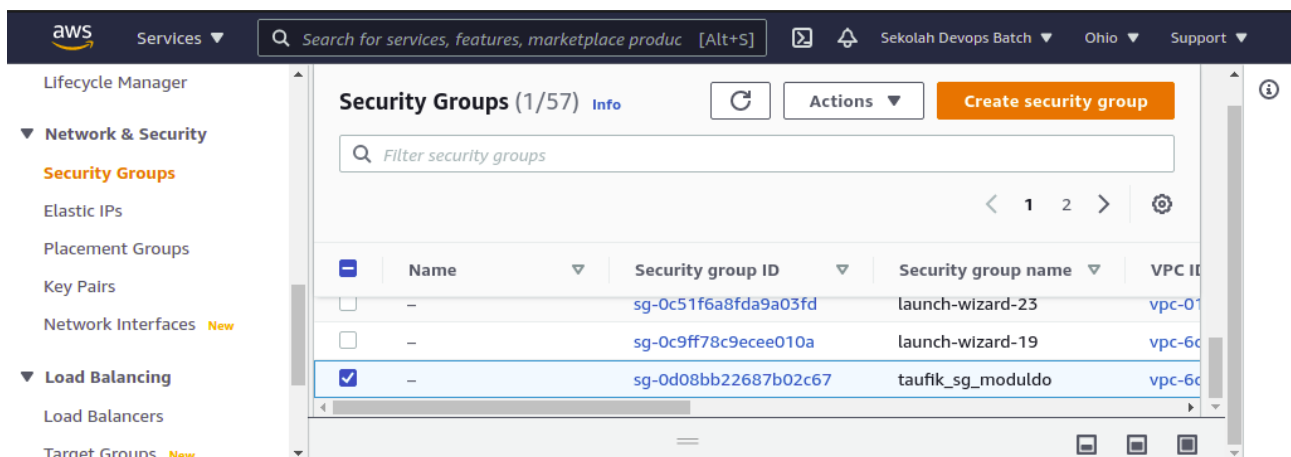
- Selanjutnya pada perintah dibawah, kita dapat mengatur default dari security group yang kita miliki, security group berguna sebagai firewall



karena pada security group port-port atau ip di atur baik allow acces maupun deny access.

```
variable "vpc_security_group_ids" {
  default = ["sg-0d08bb22687b02c67"]
}
```

- Kita dapat melihat security group pada dashboard EC2 seperti di bawah ini.



- Perintah dibawah gunanya untuk memberikan nilai default **ip public**, secara default kita dapat membuat status true, sehingga setiap terraform di jalankan maka akan mendapatkan **ip public**.

```
variable "associate_public_ip_address" {
  default = true
}
```

- Pada perintah dibawah memberikan atau mendefinisikan variable default tags, tags di gunakan untuk memberikan nama atau deskripsi dari EC2 yang kita buat.

```
variable "tags" {
  type = map(string)
  default = {
    "name" = "sekolah-devops-instance"
    "purpose" = "praktikum-sekolah-devops"
```



```
"env" = "dev"

}

}
```

- Pada perintah dibawah berguna untuk memberikan atau mendefinisikan variable default jika ingin menginstall aplikasi-aplikasi pada linux, misal install python dan lain-lain.

```
variable "install_ec2" {
    default = "apt-get -y python"
}
```

12.4.3. Exercise

Soal Praktek :

1. Desain Sebuah infrastructure EC2 yang akan kalian buat.
2. Sesuaikan file-file konfigurasi terraform yang sudah dipelajari tadi sesuai dengan desain yang akan kalian buat, mulai dari Security Group, AMI, dan Subnet yang ada.

12.5. Konfigurasi Terraform

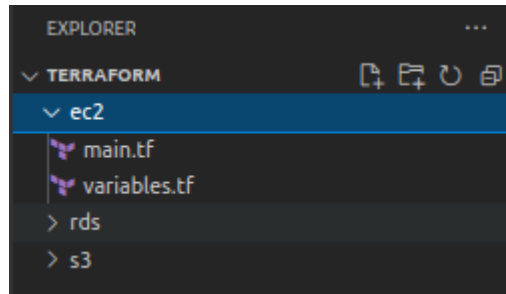
12.5.1. Konfigurasi EC2

12.5.1.1. Membuat EC2

Selanjutnya adalah membuat EC2 dengan menggunakan terraform, file pada terraform sendiri sudah kita jelaskan sebelumnya. Jika ada yang bingung tentang sintak-sintak atau perintah dari terraform maka dapat membaca penjelasan kembali di atas,

Untuk membuat EC2 dengan menggunakan EC2 terraform maka kita harus membuat struktur file seperti di bawah ini. Sebenarnya kita sudah membuatnya tadi, jadi kita hanya perlu mengikuti ke proses selanjutnya.





Ilustrasi Struktur File Terraform

Selanjutnya masuk ke terminal pada folder terraform/ec2 lalu ketik perintah **terraform init**. Pada proses ini, terraform menyiapkan dependensi seperti backend, provider yang digunakan. Setelah selesai, terraform akan membuat file **state** pada directory tersebut.

```
cd ec2
terraform init
```

Maka hasilnya akan seperti dibawah ini.

```
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform/ec2$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v3.48.0

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.
```

Setelah itu ketikan perintah plan seperti dibawah ini.

```
terraform plan
```



Setelah menjalankan perintah di atas maka, terraform akan mengecek semua perintah yang ada di file .tf apakah sudah tepat, jika masih ada yang belum tepat, maka lihatlah kembali pada sub bab yang di atas.

```
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform/ec2$ 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_instance.devops[0] will be created
+ resource "aws_instance" "devops" {
  + ami                  = "ami-0b9064170e32bde34"
  + arn                  = (known after apply)
  + associate_public_ip_address = true
  + availability_zone     = (known after apply)
  + cpu_core_count       = (known after apply)
  + cpu_threads_per_core  = (known after apply)
  + get_password_data     = false
  + host_id              = (known after apply)
  + id                   = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_state        = (known after apply)
  + instance_type         = "t2.micro"
  + ipv6_address_count    = (known after apply)
  + ipv6_addresses       = (known after apply)
  + key_name              = "taufik_moduldo"
  + outpost_arn           = (known after apply)
}
```

Pada akhir output, ada keterangan seperti berikut yang artinya, kita akan membuat resource sesuai dengan konfigurasi di **main.tf**.

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

Langkah selanjutnya adalah menjalankan program terraform dengan menggunakan perintah berikut ini.

terraform apply

```
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform/ec2$ 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_instance.devops[0] will be created
+ resource "aws_instance" "devops" {
  + ami                  = "ami-0b9064170e32bde34"
  + arn                  = (known after apply)
  + associate_public_ip_address = true
  + availability_zone     = (known after apply)
  + cpu_core_count       = (known after apply)
  + cpu_threads_per_core  = (known after apply)
  + get_password_data     = false
  + host_id              = (known after apply)
  + id                   = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_state        = (known after apply)
  + instance_type         = "t2.micro"
  + ipv6_address_count    = (known after apply)
  + ipv6_addresses       = (known after apply)
}
```

Untuk mengkonfirmasi apakah konfigurasi sudah dirasa benar, kita masukan yes untuk konfirmasi.

```
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.

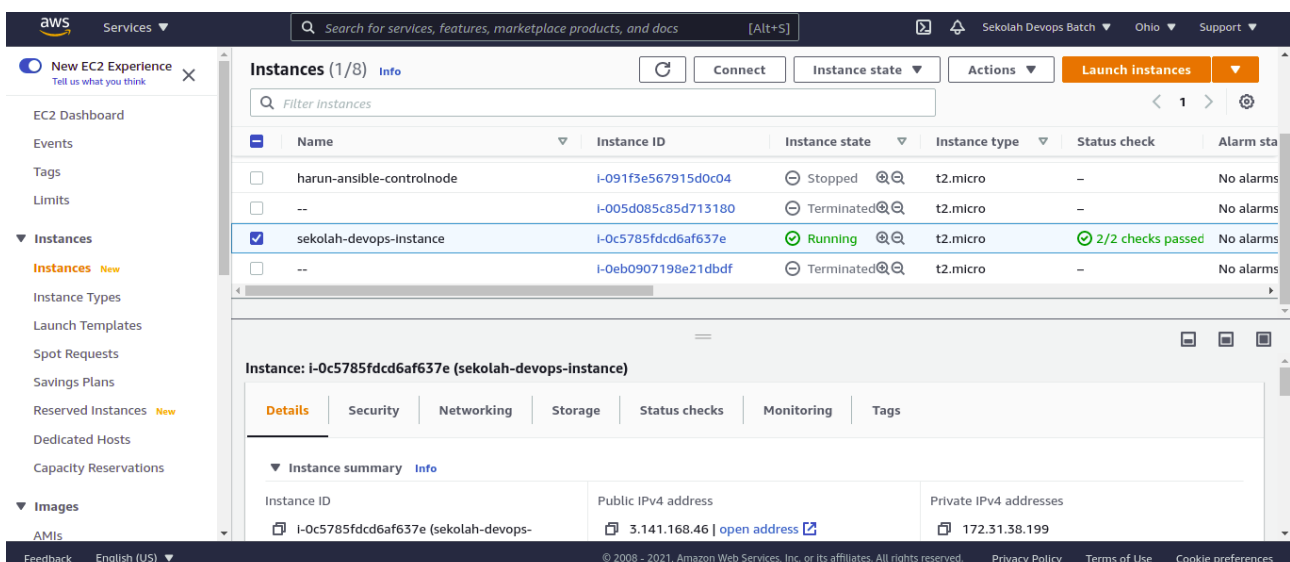
Enter a value: yes
```

Tunggu sampai tampilan seperti di atas, maka terraform yang kita jalankan telah berhasil membuat 1 instance EC2 pada AWS .

```
aws_instance.devops[0]: Still creating... [1m30s elapsed]
aws_instance.devops[0]: Creation complete after 1m31s [id=i-0c5785fdcd6af637e]

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

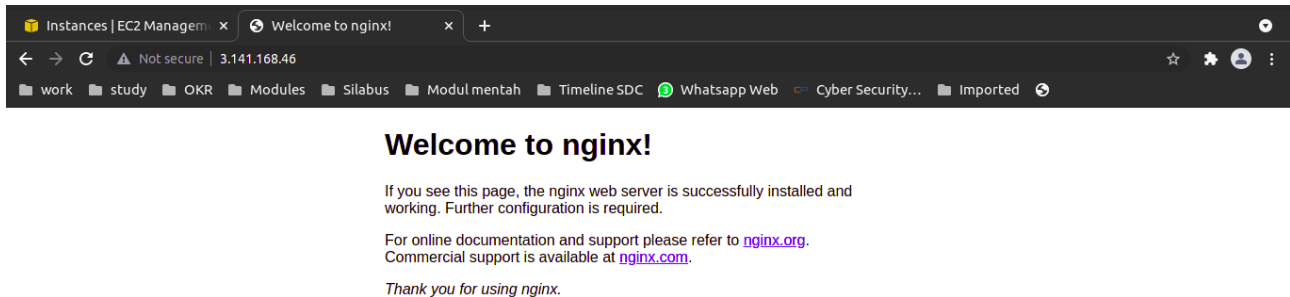
Jika berhasil maka akan muncul pada dashboard EC2 seperti dibawah ini.



Hasil EC2 dari Terraform



kita bisa mengecek service nginx pada instance tersebut dengan mengakses ip public dari instance tersebut.



12.5.1.2. Manage EC2

Setelah kita membuat EC2, selanjutnya kita manage EC2 dengan terraform. Cukup dengan mengubah **variable.tf** yang ada pada folder **ec2/**.

```
variable "tags" {
  type = map(string)
  default = {
    "name" = "sekolah-devops-instance"
    "purpose" = "praktikum-sekolah-devops"
    "env" = "dev"
  }
}
```

Pada bagian name dan env kita ubah menjadi staging.

```
variable "tags" {
  type = map(string)
  default = {
    "name" = "sekolah-devops-staging"
    "purpose" = "praktikum-sekolah-devops"
    "env" = "staging"
  }
}
```




```
}
}
```

Selanjutnya jalankan kembali perintah plan untuk melihat perubahan.

```
terraform plan
```

```
taufik@hewlett-packard:~/Documents/kerja/cilsy/praktikum/terraform/terraform/ec2$ terraform plan
aws_instance.devops[0]: Refreshing state... [id=i-0c5785fdcd6af637e]

Note: Objects have changed outside of Terraform

Terraform detected the following changes made outside of Terraform since the last "terraform apply":

# aws_instance.devops[0] has been changed
~ resource "aws_instance" "devops" {
  id          = "i-0c5785fdcd6af637e"
  tags       = {
    "Env"      = "dev"
    "Name"     = "sekolah-devops-instance"
    "Purpose"  = "praktikum-sekolah-devops"
  }
  # (30 unchanged attributes hidden)

  ~ root_block_device {
    + tags = {}
    # (8 unchanged attributes hidden)
  }
  # (5 unchanged blocks hidden)
}

Unless you have made equivalent changes to your configuration, or ignored the relevant attributes using ignore_changes, the following plan may include actions to undo or respond to these changes.
```

Terraform will perform the following actions:

```
# aws_instance.devops[0] will be updated in-place
~ resource "aws_instance" "devops" {
  id          = "i-0c5785fdcd6af637e"
  ~ tags      = {
    ~ "Env"      = "dev" -> "staging"
    ~ "Name"     = "sekolah-devops-instance" -> "sekolah-devops-staging"
    # (1 unchanged element hidden)
  }
  ~ tags_all  = {
    ~ "Env"      = "dev" -> "staging"
    ~ "Name"     = "sekolah-devops-instance" -> "sekolah-devops-staging"
    # (1 unchanged element hidden)
  }
  ~ volume_tags = {
    ~ "Env"      = "dev" -> "staging"
    ~ "Name"     = "sekolah-devops-instance" -> "sekolah-devops-staging"
    # (1 unchanged element hidden)
  }
  # (28 unchanged attributes hidden)
```

Pada gambar di atas kita dapat melihat perubahan yang terjadi, setelah perubahan sudah benar maka selanjutnya adalah menjalankan perintah apply.

```
terraform apply
```

Selanjutnya akan muncul gambar seperti dibawah, maka dengan ini terraform kita telah berhasil menjalankan file main.tf dengan baik.

```
aws_instance.devops[0]: Modifying... [id=i-0c5785fdcd6af637e]
aws_instance.devops[0]: Still modifying... [id=i-0c5785fdcd6af637e, 10s elapsed]
aws_instance.devops[0]: Modifications complete after 16s [id=i-0c5785fdcd6af637e]

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

Untuk melihat hasilnya pada dashboard EC2, kita akan melihat perubahan nama yang sudah kita lakukan sebelumnya.

The screenshot shows the AWS Management Console interface. At the top, there's a search bar and navigation links. Below, the 'Instances (1/8)' section is visible. A table lists several instances, with 'sekolah-devops-staging' (Instance ID: i-0c5785fdcd6af637e) highlighted in blue. This instance is in a 'Running' state with '2/2 checks passed'. Below the table, the details for the selected instance are shown, including its Public IPv4 address (3.141.168.46) and Private IPv4 addresses (172.31.38.199).

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
harun-ansible-controlnode	i-091f3e567915d0c04	Stopped	t2.micro	-	No alarms
--	i-005d085c85d713180	Terminated	t2.micro	-	No alarms
sekolah-devops-staging	i-0c5785fdcd6af637e	Running	t2.micro	2/2 checks passed	No alarms
--	i-0eb0907198e21dbdf	Terminated	t2.micro	-	No alarms

Instance: i-0c5785fdcd6af637e (sekolah-devops-staging)

Instance summary		
Instance ID	Public IPv4 address	Private IPv4 addresses
i-0c5785fdcd6af637e (sekolah-devops-)	3.141.168.46 open address	172.31.38.199

Hasil Manage EC2 Terraform



12.5.1.3. Menghapus EC2

Untuk mendelete atau destroy EC2 yang sudah kita buat tadi, sekarang cukup dengan menjalankan perintah seperti di bawah ini.

terraform destroy

```
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform/ec2$ terraform destroy
aws_instance.devops[0]: Refreshing state... [id=i-0c5785fdcd6af637e]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
- destroy

Terraform will perform the following actions:

# aws_instance.devops[0] will be destroyed
- resource "aws_instance" "devops" {
  - ami                  = "ami-0b9064170e32bde34" -> null
  - arn                  = "arn:aws:ec2:us-east-2:898130718046:instance/i-0c5785fdcd6af637e" -> null
  - associate_public_ip_address = true -> null
  - availability_zone     = "us-east-2c" -> null
  - cpu_core_count        = 1 -> null
  - cpu_threads_per_core   = 1 -> null
  - disable_api_termination = false -> null
  - ebs_optimized         = false -> null
  - get_password_data      = false -> null
  - hibernation           = false -> null
  - id                    = "i-0c5785fdcd6af637e" -> null
  - instance_initiated_shutdown_behavior = "stop" -> null
  - instance_state        = "running" -> null
  - instance_type         = "t2.micro" -> null
}
```

Untuk konfirmasi menghapus resource, ketikan yes

```
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
```

Tunggu beberapa saat agar Terraform menghapus resource.

```
aws_instance.devops[0]: Destroying... [id=i-0c5785fdcd6af637e]
aws_instance.devops[0]: Still destroying... [id=i-0c5785fdcd6af637e, 10s elapsed]
aws_instance.devops[0]: Still destroying... [id=i-0c5785fdcd6af637e, 20s elapsed]
aws_instance.devops[0]: Still destroying... [id=i-0c5785fdcd6af637e, 30s elapsed]
aws_instance.devops[0]: Still destroying... [id=i-0c5785fdcd6af637e, 40s elapsed]
aws_instance.devops[0]: Destruction complete after 42s

Destroy complete! Resources: 1 destroyed.
```

Cek **dashboard EC2** kita dapat melihat seperti di atas ini. Apakah resource berubah menjadi **terminated** ?



12.5.2. Konfigurasi S3

12.5.2.1. Membuat Bucket S3

Selanjutnya pada bagian ini kita akan coba membuat Bucket S3 baru menggunakan Terraform, pada pembahasan di atas kita tidak membahas mengenai script yang ada pada pembuatan S3 dikarenakan tidak banyak script yang akan kita gunakan, maka kita akan jelaskan disini sembari mencobanya.

Untuk buat sebuah direktori baru didalam direktori terraform yaitu S3 seperti dibawah ini.

```
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform$ ls
ec2  rds  s3
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform$ cd s3
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform/s3$
```

Setelah itu masuk kedalam folder s3 tersebut, kita lihat file main.tf. Konfigurasinya tidak jauh sama dengan EC2, namun kali ini ada beberapa input/argument khusus resource s3 bucket seperti bucket, acl, versioning dll. Seperti biasa, dokumentasi nya ada [disini](#).

```
provider "aws" {
  region = "us-east-2"
}

resource "aws_s3_bucket" "example" {
  bucket = "devops-cilsy-bucket"
  acl    = "private"
  versioning {
    enabled = true
  }

  tags {
    Name = "devops-cilsy-bucket"
  }
}
```



Setelah itu simpan konfigurasi tersebut. Jika kita lihat pada script diatas tidak berbeda jauh dengan konfigurasi pembuatan EC2 pada bagian sebelumnya, dan jika kalian memahami cara pembuatan S3 sendiri mungkin tidak akan terlalu sulit pada saat melihat maksud dari script diatas.

Untuk beberapa opsional script lanjutan pada S3 ini dapat kalian akses melalui web dokumentasi official dari terraform sendiri, yaitu sebagai berikut :

https://www.terraform.io/docs/providers/aws/r/s3_bucket.html

Lanjut pada pembuatan, selanjutnya kita jalankan init, dan plan pada folder S3 tersebut.

terraform init

```
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform/s3$ terraform init
Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.48.0...
- Installed hashicorp/aws v3.48.0 (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.
```

Lalu kita lakukan perintah berikut untuk melihat konfigurasi yang akan dibangun.

terraform plan



```
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform/s3$ 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_s3_bucket.devops_bucket will be created
+ resource "aws_s3_bucket" "devops_bucket" {
+   acceleration_status      = (known after apply)
+   acl                     = "private"
+   arn                     = (known after apply)
+   bucket                  = "devops-cilsy-bucket"
+   bucket_domain_name      = (known after apply)
+   bucket_regional_domain_name = (known after apply)
+   force_destroy           = false
+   hosted_zone_id          = (known after apply)
+   id                     = (known after apply)
}
```

Setelah berhasil melakukan terraform plan, selanjutnya kita jalankan apply untuk membuild bucket S3 menggunakan terraform.

terraform apply

```
aws_s3_bucket.example: Creating...
  acceleration_status: "" => "<computed>"
  acl: "" => "private"
  arn: "" => "<computed>"
  bucket: "" => "devops-cilsy-bucket"
  bucket_domain_name: "" => "<computed>"
  bucket_regional_domain_name: "" => "<computed>"
  force_destroy: "" => "false"
  hosted_zone_id: "" => "<computed>"
  region: "" => "<computed>"
  request_payer: "" => "<computed>"
  tags.%: "" => "1"
  tags.Name: "" => "devops-cilsy-bucket"
  versioning.#: "" => "1"
  versioning.0.enabled: "" => "true"
  versioning.0.mfa_delete: "" => "false"
  website_domain: "" => "<computed>"
  website_endpoint: "" => "<computed>"
aws_s3_bucket.example: Still creating... (10s elapsed)
aws_s3_bucket.example: Still creating... (20s elapsed)
aws_s3_bucket.example: Creation complete after 29s (ID: devops-cilsy-bucket)

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

jangan lupa untuk konfirmasi dengan mengetik yes

```
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_s3_bucket.devops_bucket: Creating...
aws_s3_bucket.devops_bucket: Still creating... [10s elapsed]
aws_s3_bucket.devops_bucket: Still creating... [20s elapsed]
aws_s3_bucket.devops_bucket: Still creating... [30s elapsed]
aws_s3_bucket.devops_bucket: Creation complete after 37s [id=devops-cilsy-bucket]

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

Setelah berhasil, kita bisa lihat hasilnya didalam console AWS untuk S3 seperti berikut

Name	AWS Region	Access	Creation date
devops-cilsy-bucket	US East (Ohio) us-east-2	Objects can be public	July 8, 2021, 14:51:33 (UTC+07:00)
devops-cilsy.s3.ap-southeast-1	Asia Pacific (Singapore) ap-southeast-1	Objects can be public	March 13, 2021, 10:43:56 (UTC+07:00)

Dengan begini pembuatan S3 menggunakan terraform sudah berhasil dilakukan, kalian dapat mencoba untuk memodifikasi script yang ada dengan tambahan komponen yang lainnya.

12.5.2.2. Manage Bucket S3

Sama seperti EC2, segala perubahan pada file main.tf bisa dimaintain untuk di apply. Misalnya mengubah nama bucket dan lain lain.

Setelah itu, jalankan perintah berikut

```
terraform plan
```

```
terraform apply
```

Terakhir, konfirmasi dengan mengetik yes.



12.5.2.3. Menghapus Bucket S3

Untuk menghapusnya sendiri perintahnya tidak berbeda dengan penghapusan EC2 sebelumnya, kita bisa menggunakan perintah berikut.

```
terraform destroy
```

Lalu konfirmasi dengan mengetik yes.



12.5.3. Konfigurasi Amazon RDS

12.5.3.1. Membuat Database RDS

Selanjutnya pada bagian ini kita akan coba melakukan konfigurasi pada Amazon RDS, sehingga kita dapat membuat sebuah database RDS baru dengan menggunakan terraform. Konfigurasi tersebut bisa kalian dapatkan di link yang sebelumnya sudah diberikan, disana akan terdapat dua file yaitu **main.tf** dan **variable.tf**

```
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform$ ls
ec2  rds  s3
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform$ cd rds/
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform/rds$
```

Kita dapat coba untuk menyesuaikan isi yang ada pada file **variable.tf** seperti dibawah ini, sebagai contoh yang kita akan coba buat disini adalah database mysql.

```
variable "aws_access_key" {
    default = "AKIA5CHGDFVPB3GEAVPJ"
}

variable "aws_secret_key" {
    default = "lt7foQ/I3m23BY86AbJtAV5Izkv2ZN*****"
}

variable "db_instance" {
    default = "db.t2.micro"
}

variable "rds_engine" {
    default = "mysql"
}

variable "rds_engine_version" {
```



```

    default = "5.7"
}

variable "rds_identifier" {
    default = "terraformrds"
}

variable "rds_db_name" {
    default = "terraformrds"
}

variable "rds_db_username" {
    default = "devopscilsy"
}

variable "rds_db_password" {
    default = "1234567890"
}

variable "rds_parameter_group_name" {
    default = "default.mysql5.7"
}

```

Disana kalian dapat melakukan set pada database yang akan digunakan, database name, password dan juga identifier pada RDS yang akan kalian buat. Untuk parameter lebih detailnya kalian dapat mengakses pada situs resminya berikut. https://www.terraform.io/docs/providers/aws/r/db_instance.html

Lanjut pada pembuatan, selanjutnya kita jalankan init, dan plan pada folder rds tersebut.

```

terraform init
terraform plan

```



```
taufik@hewlettpackard:~/Documents/kerja/cilsy/praktikum/terraform/terraform/rds$ 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_db_instance.devops_mysql will be created
+ resource "aws_db_instance" "devops_mysql" {
  + address                        = (known after apply)
  + allocated_storage              = 20
  + allow_major_version_upgrade   = true
  + apply_immediately              = (known after apply)
  + arn                           = (known after apply)
  + auto_minor_version_upgrade    = true
  + availability_zone              = (known after apply)
  + backup_retention_period        = 35
  + backup_window                  = "22:00-23:00"
  + ca_cert_identifier             = (known after apply)
  + character_set_name             = (known after apply)
  + copy_tags_to_snapshot         = false
  + db_subnet_group_name          = (known after apply)
  + delete_automated_backups      = true
  + endpoint                      = (known after apply)
  + engine                        = "mysql"
  + engine_version                 = "5.7"
```

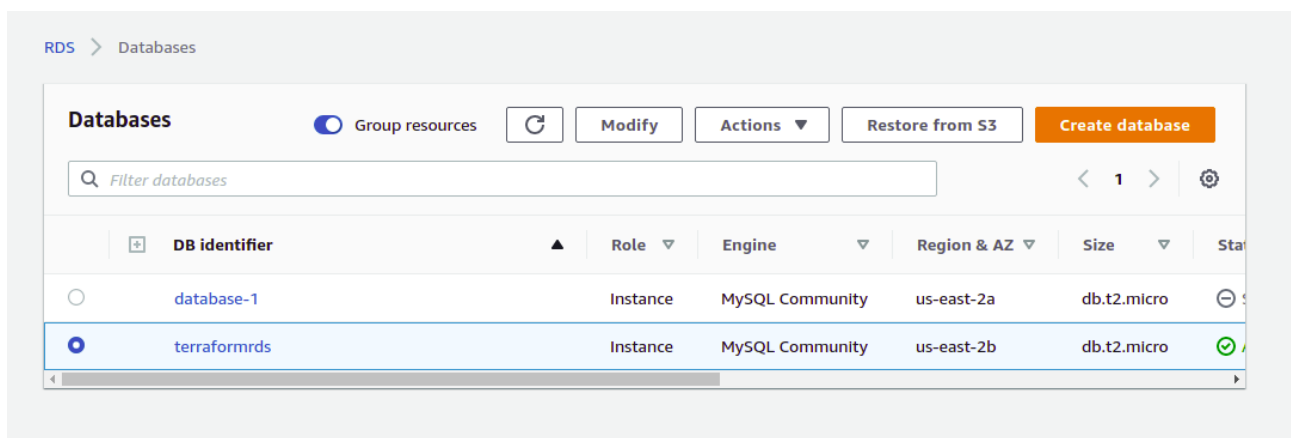
Setelah berhasil melakukan terraform plan, selanjutnya kita jalankan apply untuk membuild database RDS menggunakan terraform. Memang proses ini membutuhkan waktu sekitar 17 menit. Jadi siapkan kopi untuk santai sejenak.

terraform apply

```
aws_db_instance.devops_mysql: Still creating... [15m40s elapsed]
aws_db_instance.devops_mysql: Still creating... [15m50s elapsed]
aws_db_instance.devops_mysql: Still creating... [16m0s elapsed]
aws_db_instance.devops_mysql: Still creating... [16m10s elapsed]
aws_db_instance.devops_mysql: Still creating... [16m20s elapsed]
aws_db_instance.devops_mysql: Still creating... [16m30s elapsed]
aws_db_instance.devops_mysql: Still creating... [16m40s elapsed]
aws_db_instance.devops_mysql: Still creating... [16m50s elapsed]
aws_db_instance.devops_mysql: Still creating... [17m0s elapsed]
aws_db_instance.devops_mysql: Still creating... [17m10s elapsed]
aws_db_instance.devops_mysql: Still creating... [17m20s elapsed]
aws_db_instance.devops_mysql: Creation complete after 17m27s [id=terraformrds]

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

Setelah berhasil, kita bisa lihat hasilnya didalam console AWS untuk melihat database RDS tersebut seperti berikut.



Dengan begini kita sudah berhasil membuat sebuah database mysql baru pada amazon RDS menggunakan terraform, kita dapat melakukan modifikasi pada database sesuai yang akan kita gunakan.

12.5.3.2. Menghapus database RDS

Untuk menghapusnya sendiri perintahnya tidak berbeda dengan sebelumnya, kita bisa menggunakan perintah berikut.

```
terraform destroy
```

12.5.4. Exercise

Soal Praktek :

1. Build sebuah Instance baru menggunakan sistem operasi Amazon Linux dan Juga CentOS.
2. Build sebuah bucket S3 dan juga database mysql di RDS
3. Pastikan Semua Sistem Operasi , bucket dan database dapat diakses
4. Setelah semua dapat di akses, screenshot hasilnya.
5. Destroy semua instance, S3 dan database yang sudah dibuat

12.6. Create EC2 dengan terraform (Multiple Server)

Pada bagian ini kita akan mencoba untuk membuat beberapa server sekaligus dengan menggunakan terraform seperti di bawah ini. Tahap pertama yang harus kita lakukan adalah membuka file variables.tf pada **ec2/**.

```
variable "instance_count" {  
    default = 1  
}
```



Lalu ubah menjadi seperti dibawah ini dengan menambahkan instance_count dan nilainya.

```
variable "instance_count" {
    default = 3
}
```

Instance count, berguna untuk mengatur jumlah server yang ingin kita deploy pada AWS. Untuk menjalankan program tersebut maka jalankan perintah berikut.

terraform plan

Hasil perintah di atas akan menghasilkan seperti gambar dibawah ini.

```
+ kms_key_id      = (known after apply)
+ throughput      = (known after apply)
+ volume_id       = (known after apply)
+ volume_size     = 8
+ volume_type     = "gp2"
}
Plan: 3 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.
```

Selanjutnya apply terraform yang sudah kita setting tadi.

terraform apply

```
+ kms_key_id      = (known after apply)
+ throughput      = (known after apply)
+ volume_id       = (known after apply)
+ volume_size     = 8
+ volume_type     = "gp2"
}
}
Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

Enter a value: yes
```

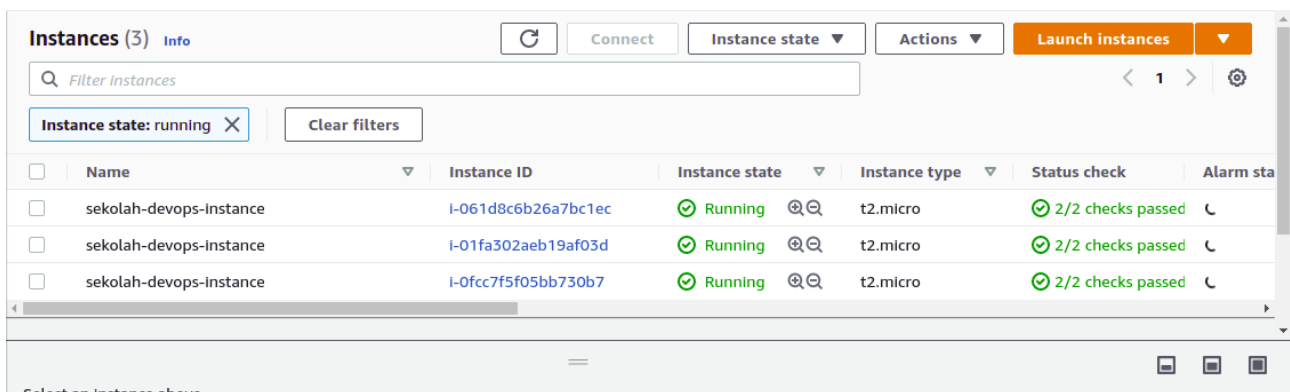


```
aws_instance.devops[0] (remote-exec): Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
aws_instance.devops[0] (remote-exec): Processing triggers for ufw (0.36-0ubuntu0.18.04.1) ...
aws_instance.devops[0] (remote-exec): Processing triggers for ureadahead (0.100.0-21) ...
aws_instance.devops[0] (remote-exec): Processing triggers for libc-bin (2.27-3ubuntu1.4) ...

aws_instance.devops[0]: Creation complete after 2m58s [id=i-061d8c6b26a7bc1ec]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

Jika muncul gambar seperti di atas maka proses berjalan dengan baik, kita dapat melihat dashboard EC2 maka akan muncul 3 instance yang telah kita buat.



The screenshot shows the AWS Management Console 'Instances' page. At the top, it says 'Instances (3)' and 'Info'. There are buttons for 'Connect', 'Instance state', 'Actions', and 'Launch instances'. A search bar 'Filter Instances' is present. Below the filters, there is a table with 3 instances. Each instance is named 'sekolah-devops-Instance' and is in a 'Running' state. The instance types are 't2.micro' and all status checks have passed (2/2 checks passed).

Name	Instance ID	Instance state	Instance type	Status check	Alarm sta
sekolah-devops-Instance	i-061d8c6b26a7bc1ec	Running	t2.micro	2/2 checks passed	
sekolah-devops-Instance	i-01fa302aeb19af03d	Running	t2.micro	2/2 checks passed	
sekolah-devops-Instance	i-0fcc7f5f05bb730b7	Running	t2.micro	2/2 checks passed	

Hasil Multiple Server

12.6.1. Exercise

Soal Praktek :

1. Build 3 buah instance baru menggunakan Sistem Operasi Ubuntu Server 18.04.
2. Pastikan ketiga instance tersebut sudah terinstallkan webserver dan database server yang dapat diakses.
3. Semua tesebut dibuild menggunakan terrafrom.



12.7. Summary

1. Terraform adalah Infrastructure as Code dan salah satu tool Automation dimana sistem dibangun dan dikelola melalui kode secara automasi (otomatis), bukan secara manual.
2. Untuk melakukan konfigurasi Terraform kita membutuhkan credential untuk mengatur dan menghubungkan tool dengan akun AWS
3. File konfigurasi terraform terdapat beberapa folder yang memiliki fungsi yang berbeda beda, ada untuk membuat EC2, S3 dan RDS.
4. File file pada suatu folder dengan ekstensi .tf, akan diperlakukan sebagai 1 dokumen oleh Terraform. Namun tidak untuk file .tf yang berbeda directory.
5. Kita dapat melakukan build pada bucket S3 dan database mysql pada RDS dengan menggunakan terraform menggunakan beberapa parameter yang ada.
6. Kita dapat melakukan Build Multiple Server, dimana kita membuat beberapa buah server secara sekaligus menggunakan satu script saja.
7. Dengan menggunakan provisioning, kita dapat menginstall aplikasi pada Instance yang kita build, sehingga instance tersebut pada saat standby sudah terinstallkan aplikasi yang kita inginkan.

