

Membuat dan men-destroy Resources AWS

menggunakan Terraform - aku@widiyanto.org

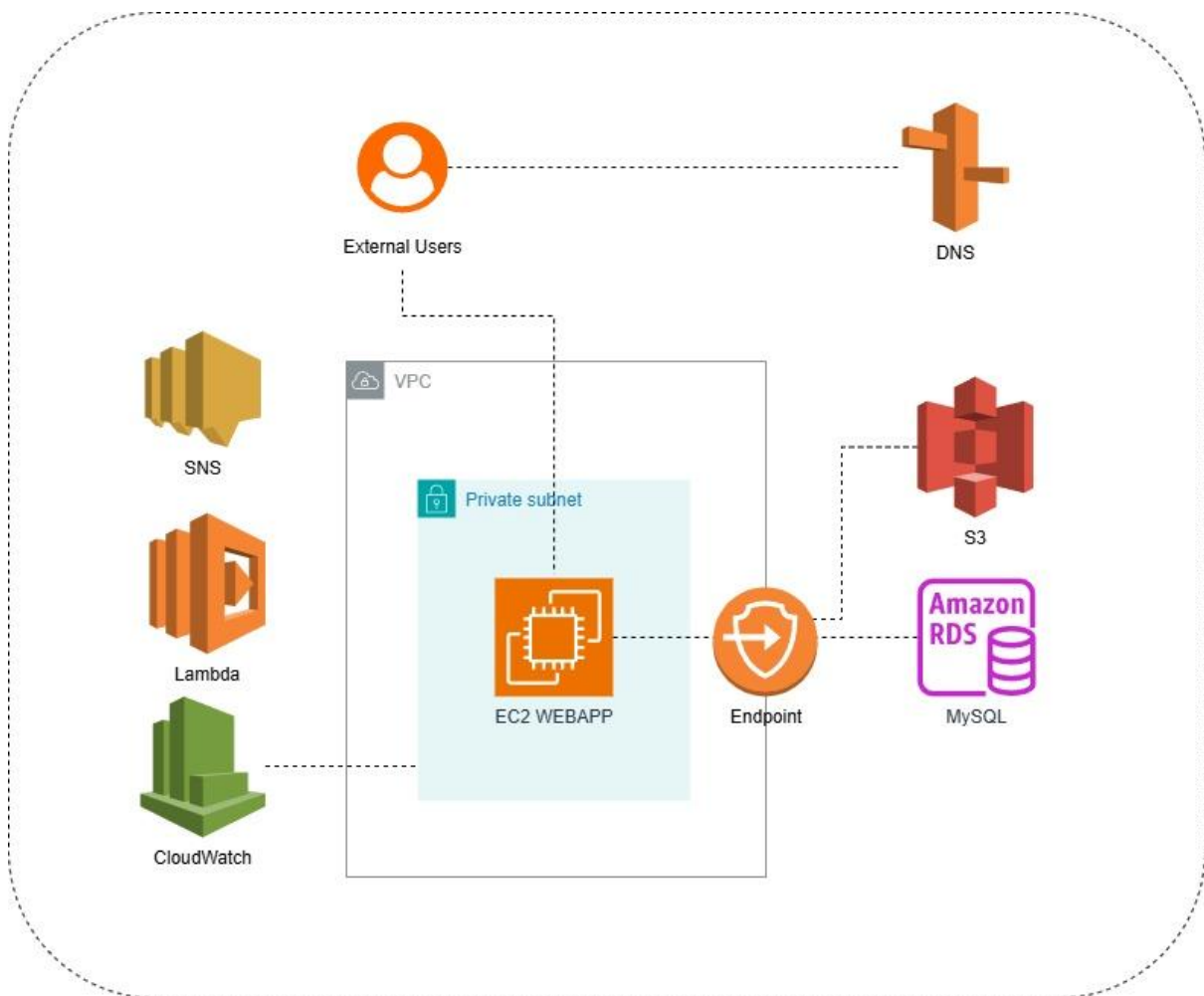
Membuat resources yang dibuat secara otomatis menggunakan Infrastructur as a Code (IaC) Terraform

Resources yang akan dibuat

1. **Amazon EC2 Instance** – Menggunakan Ubuntu 22.04 LTS - ap-southeast-1.
2. **Amazon RDS** – Menggunakan db.t3micro engine version = "8.0.35"
3. **VPC** – Menggunakan vpc defaultnya AWS
4. **S3 Bucket** – Digunakan untuk tempat penyimpanan file asset
5. **Security Group** – Allow HTTP, HTTPS, SSH, MySQL
6. **Region** - Menggunakan ap-southeast-1

🔧 Langkah Step-by-Step Setup Terraform

✓ Diagram Arsitekturnya



✓ Step 1: Persiapan Install Terraform di WSL

1. `sudo apt update && sudo apt install -y software-properties-common gnupg curl`
2. `curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg`
3. `echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com`
4. `$(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list`
5. `sudo apt update && sudo apt install -y terraform`
6. `terraform -v`

✓ Step 2: Mengatur Kredensial AWS

Terraform menggunakan salah satu dari cara berikut untuk mengakses akun AWS:

(a) File ~/.aws/credentials (paling umum)

Gunakan AWS CLI (kalau install) atau buat file ini secara manual:

```
mkdir -p ~/.aws
vi ~/.aws/credentials

[default]
aws_access_key_id = YOUR_ACCESS_KEY
aws_secret_access_key = YOUR_SECRET_KEY

vi ~/.aws/config

[default]
region = ap-southeast-1

bisa dapatkan akses key dari: IAM > Users > Security credentials
```

✓ Step 3: Koneksi Internet

Karena Terraform akan:

- Menghubungi AWS API
- Mendownload provider plugin

Pastikan WSL **punya akses internet** (cek ping google.com atau curl aws.amazon.com).

✓ Step 4: Struktur Terraform sederhana

Kita akan buat struktur Terraform sederhana untuk:

- 1 VPC
- 1 EC2 (Ubuntu t2.micro)
- 1 RDS (MySQL)
- 1 S3 Bucket

Struktur direktori:

css

Copy

Edit

```
terraform/
├─ main.tf
├─ variables.tf
└─ outputs.tf
```

variables.tf

```
variable "region" {  
    default = "ap-southeast-1"  
}  
  
variable "db_username" {  
    default = "tempAdmin"  
}  
  
variable "db_password" {  
    default = "tempAdmin954*"  
}
```

outputs.tf

```
output "ec2_public_ip" {  
    value = aws_instance.web.public_ip  
}  
  
output "rds_endpoint" {  
    value = aws_db_instance.mysql.endpoint  
}  
  
output "s3_bucket_name" {  
    value = aws_s3_bucket.app_bucket.bucket  
}
```

```
provider "aws" {

  data "aws_vpc" "default" {

    default = true

  }

  # Perubahan di sini - menggunakan aws_subnets sebagai ganti aws_subnet_ids

  data "aws_subnets" "default" {

    filter {

      name = "vpc-id"

      values = [data.aws_vpc.default.id]

    }

  }

  # Security Group

  resource "aws_security_group" "web_sg" {

    name = "web-sg"

    description = "Allow HTTP, HTTPS, SSH, MySQL"

    vpc_id = data.aws_vpc.default.id

    ingress {

      from_port = 22

      to_port = 22

      protocol = "tcp"

      cidr_blocks = ["0.0.0.0/0"]

    }

    ingress {

      from_port = 80

      to_port = 80

      protocol = "tcp"

      cidr_blocks = ["0.0.0.0/0"]

    }

    ingress {

      from_port = 443

      to_port = 443

      protocol = "tcp"

      cidr_blocks = ["0.0.0.0/0"]

    }

    ingress {

      from_port = 3306

      to_port = 3306

      protocol = "tcp"

      self = true # Mengganti security_groups dengan self

    }

  }
```

```
egress {

  from_port = 0

  to_port = 0

  protocol = "-1"

  cidr_blocks = ["0.0.0.0/0"]

}

# EC2 Instance

resource "aws_instance" "web" {

  ami = "ami-0c1907b6d738188e5" # Ubuntu 22.04 LTS - ap-southeast-1

  instance_type = "t2.micro"

  subnet_id = data.aws_subnets.default.ids[0] # Perubahan di sini

  # Ganti ini:

  #security_groups = [aws_security_group.web_sg.name]

  # Menjadi ini:

  vpc_security_group_ids = [aws_security_group.web_sg.id]

  key_name = "wid" # ganti dengan nama key pair

  tags = {

    Name = "WebAppInstance"

  }

}

# RDS Instance

resource "aws_db_instance" "mysql" {

  allocated_storage = 20

  engine = "mysql"

  engine_version = "8.0.35"

  instance_class = "db.t3.micro"

  db_name = "webappdb" # Ganti 'name' dengan 'db_name' untuk MySQL

  username = var.db_username

  password = var.db_password

  db_subnet_group_name = aws_db_subnet_group.default.name

  vpc_security_group_ids = [aws_security_group.web_sg.id]

  skip_final_snapshot = true

  backup_retention_period = 0

  monitoring_interval = 0

  publicly_accessible = false # Disarankan untuk keamanan

}

resource "aws_db_subnet_group" "default" {

  name = "default-subnet-group"

  subnet_ids = data.aws_subnets.default.ids # Perubahan di sini
```

Penjelasan lengkap main.tf

Berikut penjelasan lengkap untuk konfigurasi main.tf bagian per bagian:

1. Provider AWS

Provider AWS

```
provider "aws" {  
  region = var.region  
}
```

- Mendefinisikan provider AWS dan region yang akan digunakan
- var.region berarti nilai diambil dari variabel (biasa didefinisikan di variables.tf)

2. S3 Bucket

```
resource "aws_s3_bucket" "app_bucket" {  
  bucket = "my-webapp-bucket-${random_id.bucket_id.hex}"  
  force_destroy = true  
}  
  
resource "random_id" "bucket_id" {  
  byte_length = 4  
}
```

- Membuat bucket S3 dengan nama unik (mengandung random hex)
- force_destroy = true memungkinkan bucket dihapus meski berisi data
- random_id menghasilkan string random 4 byte (8 karakter hex) untuk keunikan nama bucket

3. Jaringan (VPC & Subnet)

```
data "aws_vpc" "default" {  
  default = true  
}  
  
data "aws_subnets" "default" {  
  filter {  
    name = "vpc-id"  
    values = [data.aws_vpc.default.id]  
  }  
}
```

- Menggunakan VPC default di akun AWS
- Mengambil semua subnet dalam VPC default tersebut
- Data source (data) digunakan untuk membaca infrastruktur yang sudah ada

4. Security Group

```
# Security Group

resource "aws_security_group" "web_sg" {

  name      = "web-sg"

  description = "Allow HTTP, HTTPS, SSH, MySQL"

  vpc_id    = data.aws_vpc.default.id


  ingress {

    from_port = 22

    to_port   = 22

    protocol  = "tcp"

    cidr_blocks = ["0.0.0.0/0"]

  }


  ingress {

    from_port = 80

    to_port   = 80

    protocol  = "tcp"

    cidr_blocks = ["0.0.0.0/0"]

  }


  ingress {

    from_port = 443

    to_port   = 443

    protocol  = "tcp"

    cidr_blocks = ["0.0.0.0/0"]

  }


  ingress {

    from_port = 3306

    to_port   = 3306

    protocol  = "tcp"

    self      = true # Mengganti security_groups dengan self

  }


  egress {

    from_port = 0

    to_port   = 0

    protocol  = "-1"

    cidr_blocks = ["0.0.0.0/0"]

  }

}
```


Membuka akses untuk:

- **SSH** (port 22) - dari mana saja (0.0.0.0/0)
- **HTTP** (port 80) - dari mana saja
- **HTTPS** (port 443) - dari mana saja
- **MySQL** (port 3306) - hanya dari instance dalam SG yang sama (self = true)
- **Egress** - mengizinkan semua traffic keluar

5. EC2 Instance

```
# EC2 Instance
resource "aws_instance" "web" {
  ami      = "ami-0c1907b6d738188e5" # Ubuntu 22.04 LTS - ap-southeast-1
  instance_type = "t2.micro"
  subnet_id   = data.aws_subnets.default.ids[0] # Perubahan di sini
  # Ganti ini:
  #security_groups = [aws_security_group.web_sg.name]
  # Menjadi ini:
  vpc_security_group_ids = [aws_security_group.web_sg.id]
  key_name    = "wid" # ganti dengan nama key pair
  tags = {
    Name = "WebAppInstance"
  }
}
```

- Membuat EC2 instance Ubuntu 22.04 tipe t2.micro
- Ditempatkan di subnet pertama dari VPC default
- Menggunakan security group yang sudah dibuat
- key_name harus diganti dengan key pair SSH yang sudah ada di AWS

6. RDS MySQL

```
resource "aws_db_instance" "mysql" {
  allocated_storage = 20
  engine            = "mysql"
  ...
  vpc_security_group_ids = [aws_security_group.web_sg.id]
  publicly_accessible = false
}
```

- Database MySQL 8.0 dengan storage 20GB
- Menggunakan security group yang sama dengan EC2 (bisa akses MySQL)
- publicly_accessible = false berarti hanya bisa diakses dari dalam VPC
- skip_final_snapshot = true (hati-hati, ini menghilangkan backup saat RDS dihapus)

7. DB Subnet Group

```
resource "aws_db_subnet_group" "default" {
  name      = "default-subnet-group"
  subnet_ids = data.aws_subnets.default.ids
}
```

- Grup subnet untuk penempatan RDS
- Menggunakan semua subnet default

8. Alur Kerja Infrastruktur

Alur Kerja Infrastruktur:

1. **Bucket S3** dibuat untuk menyimpan assets webapp
2. **EC2 Instance** akan:
 - Berjalan di subnet default
 - Bisa diakses via SSH/HTTP/HTTPS dari internet
 - Bisa akses database MySQL
3. **RDS MySQL:**
 - Hanya bisa diakses dari EC2 instance
 - Tidak terbuka ke internet
 - Berada di subnet private (asumsi subnet default termasuk private)

Yang Perlu Diperhatikan:

1. **Security:**

- Port 22 terbuka untuk semua (sebaiknya dibatasi ke IP tertentu)
- Database menggunakan SG yang sama dengan web server (idealnya dipisah)

2. **Variabel:**

- var.region, var.db_username, var.db_password harus didefinisikan

3. **Key Pair:**

- Ganti your-key-name dengan nama key pair yang sudah ada di AWS

✓ Step 5: Perintah Terraform untuk membuat resources

Untuk uji coba:

```
# 0. Inisialisasi Terraform
terraform init

# 1. Buat dan simpan plan
terraform plan -out=tfplan

# 2. Review isi plan (opsional)
terraform show tfplan

# 3. Apply plan yang sudah disimpan
terraform apply tfplan
```

Arti pesan tersebut:

1. **Tanpa opsi -out:** Ketika hanya menjalankan terraform plan (tanpa menyimpan hasilnya), Terraform hanya menunjukkan *preview* perubahan yang akan dilakukan.
2. **Tidak ada jaminan:** Jika langsung menjalankan terraform apply setelahnya, mungkin ada perbedaan antara:
 - Apa yang ditampilkan di plan
 - Apa yang benar-benar di-apply

Mengapa ini penting?

- Jika infrastruktur berubah antara waktu plan dan apply (misalnya ada perubahan manual di AWS Console), hasil apply bisa berbeda dari yang di-plan.
- Terraform tidak bisa menjamin konsistensi karena tidak ada "snapshot" rencana yang disimpan

✓ Step 6: Persiapan Terraform untuk menghapus/destroy

Untuk menghapus/destroy semua resources AWS yang telah dibuat melalui Terraform, bisa menggunakan perintah berikut:

Cara Destroy Resources Terraform:

1. **Pertama, lihat dulu apa yang akan di-destroy** (opsional tapi disarankan):

```
terraform plan -destroy -out=tfdestroyplan
```

Ini akan menunjukkan semua resources yang akan dihapus.

2. **Eksekusi destroy:**

```
terraform destroy tfdestroyplan
```

Terraform akan menampilkan daftar resources yang akan dihapus dan meminta konfirmasi.

3. **Jika ingin langsung destroy tanpa konfirmasi:**

```
terraform destroy -auto-approve
```

Beberapa catatan penting:

1. **S3 Bucket:**

- Karena mengatur `force_destroy = true`, bucket akan dihapus meskipun berisi file
- Tanpa setting ini, destroy akan gagal jika bucket tidak kosong

2. **RDS Instance:**

- Karena ada `skip_final_snapshot = true`, database akan dihapus tanpa backup terakhir
- Jika ini di-production, sebaiknya buat snapshot manual dulu

3. **Resources yang akan dihapus:**

- EC2 Instance
- Security Group
- RDS MySQL Instance
- DB Subnet Group
- S3 Bucket
- Random ID resource

Jika mengalami error saat destroy:

1. **Resources manual:**
Pastikan tidak ada resources yang dibuat manual (di luar Terraform) yang bergantung pada resources yang dikelola Terraform
2. **Dependency error:**
Terkadang perlu menjalankan destroy beberapa kali jika ada dependency issues
3. **Force destroy:**
Untuk kasus tertentu bisa tambahkan flag -force, tapi hati-hati:

```
terraform destroy -auto-approve
```

Best Practice Destroy:

1. **Backup data penting** dulu jika ada
2. **Verifikasi environment** yang akan di-destroy sudah benar
3. **Gunakan workspace** jika ingin memisahkan environment (dev/staging/prod)
4. **Set timeout** jika resources besar:

```
terraform destroy -timeout=30m
```

Setelah destroy selesai, bisa verifikasi di AWS Console bahwa semua resources sudah terhapus. Terraform juga akan menghapus file status (terraform.tfstate) yang menyimpan informasi tentang infrastruktur yang dikelola.

Evidence create & Destroy AWS Resources

```
terraform plan -out=tfplan
```

```

1 widianto@ID-LPT-073:~/terraform$ terraform plan -out=tfplan
2 data.aws_vpc.default: Reading...
3 data.aws_vpc.default: Read complete after 1s [id=vpc-018916e773db7d4bc]
4 data.aws_subnets.default: Reading...
5 data.aws_subnets.default: Read complete after 0s [id=ap-southeast-1]
6
7 Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
8   + create
9
10 Terraform will perform the following actions:
11
12   # aws_db_instance.mysql will be created
13   + resource "aws_db_instance" "mysql" {
14       + address                               = (known after apply)
15       + allocated_storage                     = 20
16       + apply_immediately                     = false
17       + arn                                   = (known after apply)
18       + auto_minor_version_upgrade           = true
19       + availability_zone                     = (known after apply)
20       + backup_retention_period               = 0
21       + backup_target                         = (known after apply)
22       + backup_window                         = (known after apply)
23       + ca_cert_identifier                    = (known after apply)
24       + character_set_name                    = (known after apply)
25       + copy_tags_to_snapshot                 = false
26       + database_insights_mode                = (known after apply)
27       + db_name                               = "webappdb"
28       + db_subnet_group_name                  = "default-subnet-group"
29       + dedicated_log_volume                  = false
30       + delete_automated_backups              = true
31       + domain_fqdn                           = (known after apply)
32       + endpoint                             = (known after apply)
33       + engine                               = "mysql"
34       + engine_lifecycle_support              = (known after apply)
35       + engine_version                       = "8.0.35"
36       + engine_version_actual                 = (known after apply)
37       + hosted_zone_id                       = (known after apply)
38       + id                                    = (known after apply)
39       + identifier                           = (known after apply)
40       + identifier_prefix                     = (known after apply)
41       + instance_class                       = "db.t3.micro"
42       + iops                                  = (known after apply)
43       + kms_key_id                           = (known after apply)

```

terraform apply tfplan

```

1 widianto@ID-LPT-073:~/terraform$ terraform apply tfplan
2 random_id.bucket_id: Creating...
3 random_id.bucket_id: Creation complete after 0s [id=kaGVJW]
4 aws_db_subnet_group.default: Creating...
5 aws_s3_bucket.app_bucket: Creating...
6 aws_security_group.web_sg: Creating...
7 aws_db_subnet_group.default: Creation complete after 1s [id=default-subnet-group]
8 aws_s3_bucket.app_bucket: Creation complete after 3s [id=my-webapp-bucket-91a19527]
9 aws_security_group.web_sg: Creation complete after 3s [id=sg-0e3968c482a1254cd]
10 aws_db_instance.mysql: Creating...
11 aws_instance.web: Creating...
12 aws_db_instance.mysql: Still creating... [10s elapsed]
13 aws_instance.web: Still creating... [10s elapsed]
14 aws_instance.web: Creation complete after 13s [id=i-0dcc7f7c60e08f3bc]
15 aws_db_instance.mysql: Still creating... [20s elapsed]
16 aws_db_instance.mysql: Still creating... [30s elapsed]
17 aws_db_instance.mysql: Still creating... [40s elapsed]
18 aws_db_instance.mysql: Still creating... [50s elapsed]
19 aws_db_instance.mysql: Still creating... [1m0s elapsed]
20 aws_db_instance.mysql: Still creating... [1m10s elapsed]
21 aws_db_instance.mysql: Still creating... [1m20s elapsed]
22 aws_db_instance.mysql: Still creating... [1m30s elapsed]
23 aws_db_instance.mysql: Still creating... [1m40s elapsed]
24 aws_db_instance.mysql: Still creating... [1m50s elapsed]
25 aws_db_instance.mysql: Still creating... [2m0s elapsed]
26 aws_db_instance.mysql: Still creating... [2m10s elapsed]
27 aws_db_instance.mysql: Still creating... [2m20s elapsed]
28 aws_db_instance.mysql: Still creating... [2m30s elapsed]
29 aws_db_instance.mysql: Still creating... [2m40s elapsed]
30 aws_db_instance.mysql: Still creating... [2m50s elapsed]
31 aws_db_instance.mysql: Still creating... [3m0s elapsed]
32 aws_db_instance.mysql: Creation complete after 3m3s [id=db-T6BOPBNQ5QZW2YROKIMEFWKA7Q]
33
34 Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
35
36 Outputs:
37
38 ec2_public_ip = "13.212.122.63"
39 rds_endpoint = "terraform-20250429021147512700000001.c5ya80m00geq.ap-southeast-1.rds.amazonaws.com:3306"
40 s3_bucket_name = "my-webapp-bucket-91a19527"

```

EC2 Instance

Instances (1/1) [Info](#)

Find Instance by attribute or tag (case-sensitive) All states ▾

Last updated less than a minute ago [Connect](#) [Instance state ▾](#) [Actions ▾](#) [Launch instances](#)

<input checked="" type="checkbox"/>	Name ↗	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability Zone ▾	Public IPv4 DNS ▾	Public IPv4 ... ▾	Elastic IP
<input checked="" type="checkbox"/>	WebAppInstance	i-0dcc7f7c60e08f3bc	Running 🔍 🔍	t2.micro	2/2 checks passed View alarms +		ap-southeast-1c	ec2-13-212-122-63.ap-...	13.212.122.63	-

i-0dcc7f7c60e08f3bc (WebAppInstance)

[Details](#) [Status and alarms](#) [Monitoring](#) [Security](#) [Networking](#) [Storage](#) [Tags](#)

▼ Instance summary [Info](#)

Instance ID
[🔗](#) i-0dcc7f7c60e08f3bc

IPv6 address
-

Hostname type
IP name: ip-172-31-7-34.ap-southeast-1.compute.internal

Answer private resource DNS name
-

Auto-assigned IP address
[🔗](#) 13.212.122.63 [Public IP]

IAM Role

Public IPv4 address
[🔗](#) 13.212.122.63 | [open address 🔗](#)

Instance state
Running

Private IP DNS name (IPv4 only)
[🔗](#) ip-172-31-7-34.ap-southeast-1.compute.internal

Instance type
t2.micro

VPC ID
[🔗](#) vpc-018916e773db7d4bc [🔗](#)

Subnet ID
[🔗](#) subnet-0e3968c482a1254cd [🔗](#)

Private IPv4 addresses
[🔗](#) 172.31.7.34

Public IPv4 DNS
[🔗](#) ec2-13-212-122-63.ap-southeast-1.compute.amazonaws.com | [open address 🔗](#)

Elastic IP addresses
-

AWS Compute Optimizer finding
[🔗](#) Opt-in to AWS Compute Optimizer for recommendations. | [Learn more 🔗](#)

Auto Scaling Group name

S3-Bucket

Amazon S3

General purpose buckets

Directory buckets

Table buckets

Access Grants

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

Storage Lens groups

AWS Organizations settings

Feature spotlight

Account snapshot - updated every 24 hours

View Storage Lens dashboard

General purpose buckets

Directory buckets

General purpose buckets (2)

Buckets are containers for data stored in S3.

Find buckets by name

Name	AWS Region	IAM Access Analyzer	Creation date
cf-templates-1cunugus3jgm1-ap-southeast-1	Asia Pacific (Singapore) ap-southeast-1	View analyzer for ap-southeast-1	April 26, 2025, 11:10:22 (UTC+07:00)
my-webapp-bucket-91a19527	Asia Pacific (Singapore) ap-southeast-1	View analyzer for ap-southeast-1	April 29, 2025, 09:11:46 (UTC+07:00)

RDS

Aurora and RDS

Databases

Query Editor

Performance Insights

Snapshots

Exports in Amazon S3

Automated backups

Reserved instances

Proxies

Subnet groups

Parameter groups

Option groups

Custom engine versions

Zero-ETL integrations

Events

Event subscriptions

Recommendations

Certificate update

terraform-20250429021147512700000001

Summary

DB identifier

terraform-20250429021147512700000001

Status

Available

Class

db.t3.micro

Role

Instance

Engine

MySQL Community

Region & AZ

ap-southeast-1a

Connectivity & security

Monitoring

Logs & events

Configuration

Zero-ETL integrations

Maintenance & backups

Data migrations - new

Tags

Recommendations

Endpoint & port

Endpoint

terraform-20250429021147512700000001.c5ya8

Port

3306

Networking

Availability Zone

ap-southeast-1a

VPC

vpc-018916e773db7d4bc

Subnet group

default-subnet-group

Subnets

subnet-0627e628bcb79ad97

subnet-0b4aeebd98aa4c1d4

subnet-0771971a020b2903f

Network type

IPv4

Security

VPC security groups

web-sg (sg-0e3968c482a1254cd)

Publicly accessible

No

Certificate authority

rds-ca-rsa2048-g1

Certificate authority date

May 22, 2061, 05:39 (UTC+07:00)

DB instance certificate expiration date

April 29, 2026, 09:13 (UTC+07:00)

Deployment Script

Untuk deployment saya membuat simple script menggunakan bash script yang bisa di cek di repositorynya,dengan nama file setup.sh yang mana :

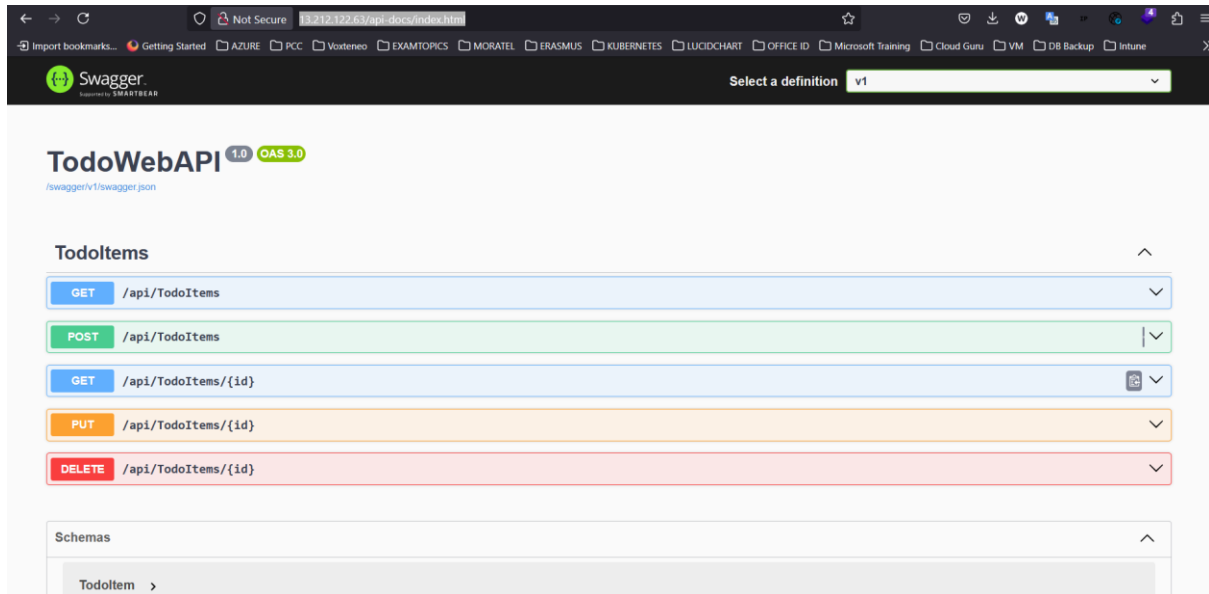
- Menyiapkan semua kebutuhan untuk menjalankan aplikasi .NET Core Web API.
- Otomatis mengkonfigurasi database, reverse proxy, systemd service, dan men-deploy aplikasi.

Manfaat:

- Menghemat waktu provisioning manual.
- Bisa diulang berkali-kali
- Memudahkan integrasi dengan automation lain (seperti Ansible, CI/CD).

Hasil deployment

Hasil deployment bisa diakses melalui URL Berikut <http://13.212.122.63/api-docs/index.html>



Cloud infra budgeting control and limit

Dalam mengontrol budgeting saya telah membuat script menggunakan cloudformation berada di repository di folder “Control Budget”

Cara Menjalankan CloudFormation Template (di AWS Console)

1. Simpan template ke file

Misalnya simpan dengan nama: zero-spend-budget.yaml

2. Buka AWS Console:

 [CloudFormation Console](#)

3. Klik “**Create stack**” → “**With new resources (strd)**”

4. Pilih template:

- Upload file zero-spend-budget.yaml

5. Isi parameter:

- **NotificationEmail:** info@widianto.org

6. Klik Next → Next → Centang “I acknowledge” → **Create stack**

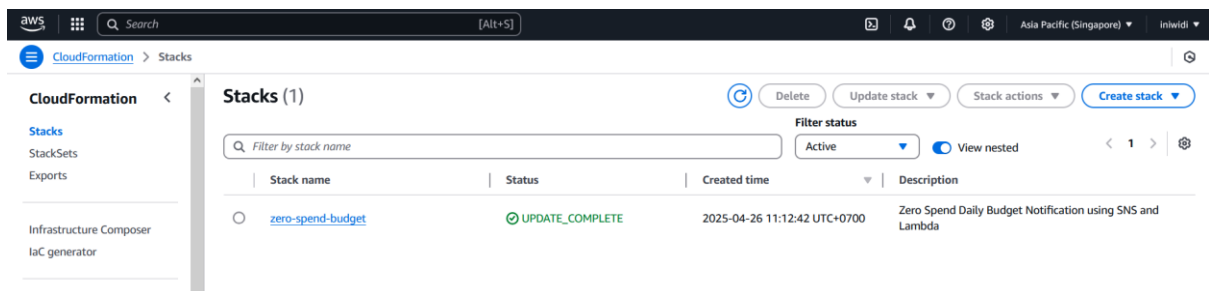
Setelah Stack Berhasil Dibuat:

1. Cek email masuk dan **konfirmasi subscription** SNS.

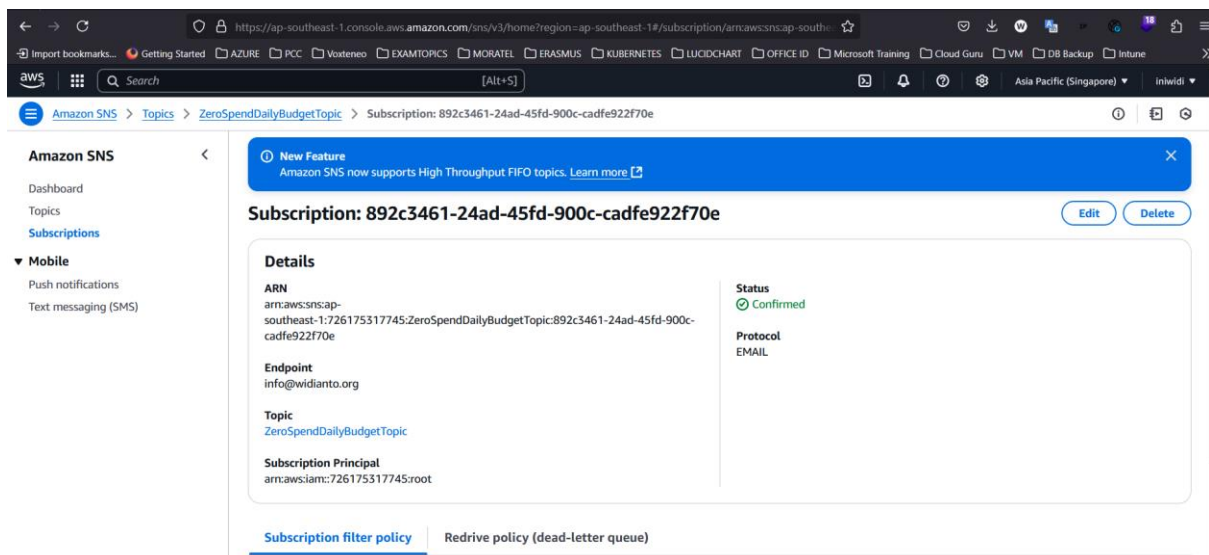
2. Budget akan aktif dan memantau limit \$0.01 setiap bulan.
3. Jika melampaui, SNS akan:
 - Kirim email
 - Trigger Lambda (saat ini hanya log event, tapi bisa dikembangkan nanti)

Evidence Infra budgeting control dan limit

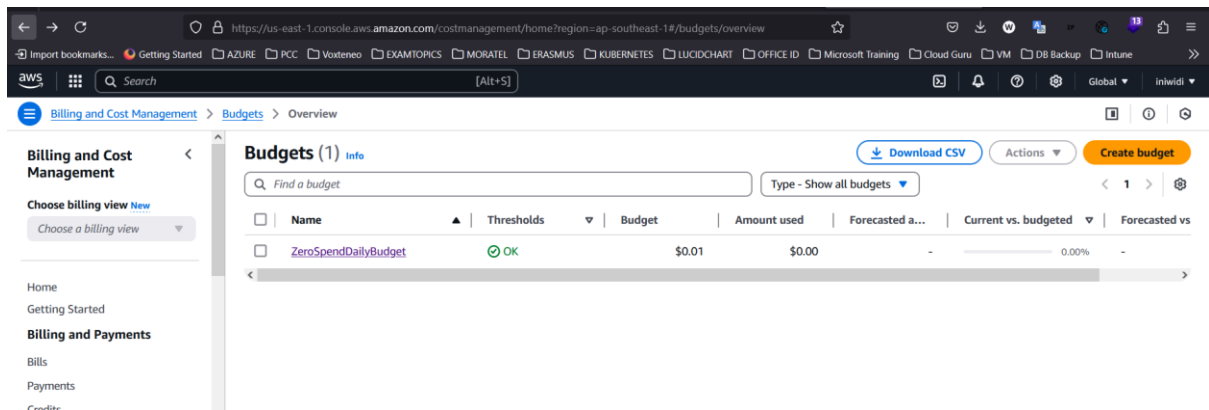
Cloudformation



SNS



Budget



Membuat centralized log dari EC2

Summary

Agar aplikasi (misal: file log nginx, php, atau dotnet app) otomatis dikirim ke CloudWatch Logs, perlu:

- Install agent **CloudWatch Logs Agent** atau lebih modern **CloudWatch Unified Agent** di EC2.
- Konfigurasi agent-nya untuk **push file log** ke CloudWatch Logs.

Syarat Agar Bisa Mengirim ke CloudWatch:

- EC2 harus memiliki **IAM Role** dengan policy: CloudWatchAgentServerPolicy
- Jika belum ada, bisa:
 - Buat IAM Role
 - Attach role ke EC2
 - Tambahkan policy berikut:

```
1 {
2   "Effect": "Allow",
3   "Action": [
4     "logs:CreateLogGroup",
5     "logs:CreateLogStream",
6     "logs:PutLogEvents"
7   ],
8   "Resource": "*"
9 }
```

◆ Kalau belum secara eksplisit membuat dan meng-attach IAM Role ke EC2, maka:


- ❌ EC2 tidak bisa mengirim log ke CloudWatch
- ❌ EC2 tidak bisa menjalankan CloudWatch Agent dengan sukses

✓ Solusinya

perlu:

1. **Membuat IAM Role** dengan policy CloudWatchAgentServerPolicy
 2. **Attach role tersebut ke EC2 instance**
-

Langkah 1: Buat IAM Role dengan Policy

1. Masuk ke **AWS Console**
 2. Buka layanan **IAM**
 3. Di sidebar kiri, pilih **Roles**
 4. Klik **[Create role]**
 5. Di bagian **Trusted entity type**, pilih: AWS service
 6. Di **Use case**, pilih: EC2 → Klik **Next**
 7. Di bagian **Permissions**, cari dan pilih:
 -  CloudWatchAgentServerPolicy
 8. Klik **Next** lagi
 9. Beri nama, misalnya: EC2CloudWatchAgentRole
 10. Klik **Create role**
-

Langkah 2: Attach IAM Role ke EC2 Instance

1. Masuk ke **EC2 Console**
 2. Pilih instance
 3. Klik tab **Actions** → pilih:
 - Security → Modify IAM role
 4. Di dropdown **IAM Role**, pilih:
 - EC2CloudWatchAgentRole (atau nama role yang buat)
 5. Klik **Update IAM Role**
-

✓ Hasil

Setelah langkah di atas selesai:

- Instance sekarang bisa mengakses AWS CloudWatch.

- Script CloudWatch Agent akan bisa membuat log group, log stream, dan push log dengan sukses.

Install CloudWatch Agent

Setelah syarat agar EC2 bisa mengirimkan data ke cloudwatch terpenuhi mari kita lanjut install AWS Agent di EC2 nya

install-cloudwatch-agent.sh yang:

- Menginstal CloudWatch Agent
- Mengatur log dari:
 - /var/log/syslog
 - Semua file di /var/log/nginx/*.log

Install-cloudwatch-agent.sh (script ada di repository)

```

1 #!/bin/bash
2
3 # Update & install CloudWatch Agent
4 sudo apt update -y
5 sudo apt install -y amazon-cloudwatch-agent
6
7 # Buat direktori config jika belum ada
8 sudo mkdir -p /opt/aws/amazon-cloudwatch-agent/etc
9
10 # Buat file konfigurasi CloudWatch Agent
11 cat <<EOF | sudo tee /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json
12 {
13     "logs": {
14         "logs_collected": {
15             "files": {
16                 "collect_list": [
17                     {
18                         "file_path": "/var/log/syslog",
19                         "log_group_name": "ec2-syslog",
20                         "log_stream_name": "{instance_id}-syslog",
21                         "timestamp_format": "%b %d %H:%M:%S"
22                     },
23                     {
24                         "file_path": "/var/log/nginx/access.log",
25                         "log_group_name": "ec2-nginx-access",
26                         "log_stream_name": "{instance_id}-nginx-access",
27                         "timestamp_format": "%Y-%m-%d %H:%M:%S"
28                     },
29                     {
30                         "file_path": "/var/log/nginx/error.log",
31                         "log_group_name": "ec2-nginx-error",
32                         "log_stream_name": "{instance_id}-nginx-error",
33                         "timestamp_format": "%Y/%m/%d %H:%M:%S"
34                     }
35                 ]
36             }
37         },
38         "log_stream_name": "default"
39     }
40 }
41 EOF
42
43 # Jalankan CloudWatch Agent dengan config
44 sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \
45     -a fetch-config \
46     -m ec2 \
47     -c file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json \
48     -s
49
50 echo "✅ CloudWatch Agent configured and started."

```

Evidence centralize log dari EC2

The screenshot displays the AWS CloudWatch console. The top navigation bar shows the AWS logo, a search bar, and various service icons. The main content area is divided into two sections: 'Log groups (3)' and 'Log events'.

Log groups (3)

By default, we only load up to 10000 log groups.

Search: Filter log groups or try prefix search

Exact match ☐

Log group	Log class	Anomaly d...	Data protection	Sensitive data co...	Retention	Metric filters	Contributor Inst...
nginx-access	Standard	Configure	-	-	Never expire	-	-
nginx-error	Standard	Configure	-	-	Never expire	-	-
syslog	Standard	Configure	-	-	Never expire	-	-

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

Search: Filter events - press enter to search

Clear 1m 30m 1h 12h Custom UTC timezone Display

Timestamp	Message
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:17 +0000] "GET /assets/123.php HTTP/1.1" 404 0 "-" "curl/7.81.0"
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:17 +0000] "GET /assets/3.php HTTP/1.1" 404 0 "-" "curl/7.81.0"
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:17 +0000] "\x16\x03\x01\x02\x00\x01\x00\x01\xFC\x03\x08\xB2\x09/\x01d:au\xF8U\xBF\x11\x02*1\xE7" \xFE\xAD\xF8/\x08;\x021\xF\xD6 1' \x00\x04M;J\x1D\x...
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:17 +0000] "GET /assets/333.php HTTP/1.1" 404 0 "-" "curl/7.81.0"
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:17 +0000] "GET /assets/333.php HTTP/1.1" 404 0 "-" "curl/7.81.0"
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:17 +0000] "\x16\x03\x01\x02\x00\x01\x00\x01\xFC\x03\x05\xC7\x14\xE78\xE1\x9A4\x09\xB2\xED\x07\xF6V4\xA8\xA6\x0E\x22R\x08\xBC\xDC\xD9\xB8\xC0\x9...
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:17 +0000] "\x16\x03\x01\x02\x00\x01\x00\x01\xFC\x03\x03\xD1E;\x0C+\xAEU1\x09\x16\x09\xC0\xE8\xA7\x06\xBD\xB5gV\xB4\xC5\xDC\xC3\x15u\x1A\x0F\xEFA\x19\x03...
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:17 +0000] "\x16\x03\x01\x02\x00\x01\x00\x01\xFC\x03\x09Y\xC8\x09\x06\x02K\x1E1' \xEEn(\x7\xB7o\xA3\xF4\xDF\xBE\x91\xEF\x11Y\xE7\x0C\x08\x0B\x0C\x...
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:18 +0000] "GET /124443.php HTTP/1.1" 404 0 "-" "curl/7.81.0"
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:18 +0000] "GET /22919811.php HTTP/1.1" 404 0 "-" "curl/7.81.0"
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:18 +0000] "GET /.1.php HTTP/1.1" 404 0 "-" "curl/7.81.0"
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:18 +0000] "\x16\x03\x01\x02\x00\x01\x00\x01\xFC\x03\x0D\x00\x91.\xB8A\x0F\x07\xCA\xF8\xB3\xA3c\xEC/\xF07] \x09/\xCEo\xB8\xC0\xD5\x07\xF0U\x07\xB6 (...)
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:18 +0000] "\x16\x03\x01\x02\x00\x01\x00\x01\xFC\x03\x03\x097;2; \xC9j\x02\x07\xA8\x11\x01\xCA\xF9g\x02\xFAG\x08\x11\x18\xA2\x1F' \x00\xF8\xCF.\xA6\x147\xB...
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:18 +0000] "GET /.3.php HTTP/1.1" 404 0 "-" "curl/7.81.0"
2025-04-30T09:27:24.873Z	64.23.201.216 - - [30/Apr/2025:09:06:18 +0000] "\x16\x03\x01\x02\x00\x01\x00\x01\xFC\x03\x03\x03\xF0b\xA7y-\xBF\x1E] \xB8\xFE\x04\xF3\x03\x09(\x0FV\x08H\x0C\x93\x03]p1\x08z\x03! 'M\x06\$K\xFE\...
2025-04-30T09:27:29.328Z	140.213.47.204 - - [30/Apr/2025:09:12:52 +0000] "GET /swagger/v1/swagger.json HTTP/1.1" 200 4961 "http://13.212.122.63/api-docs/index.html" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:137...

No newer events at this moment. Auto retry paused. [Resume](#)

[Back to top](#)

Monitoring Resources menggunakan CloudWatch

membuat **CloudWatch Alarm** untuk **CPU Usage > 75%** dan mengirim notifikasi ke email dapat dilakukan dengan membuat monitoring.tf + sns.tf

Berikut adalah struktur dan isi **monitoring.tf** dan **sns.tf** lengkap untuk membuat:

- **SNS Topic** dan **email subscription** (info@widianto.org)
- **CloudWatch Alarm** jika CPU EC2 > 75% selama 2 menit

✓ sns.tf – SNS Topic dan Email Subscription

```
# SNS Topic
resource "aws_sns_topic" "cpu_alerts" {
  name = "ec2-cpu-alerts"
}

# SNS Email Subscription
resource "aws_sns_topic_subscription" "email_alert" {
  topic_arn = aws_sns_topic.cpu_alerts.arn
  protocol = "email"
  endpoint = "info@widianto.org"
}
```

⚠ Setelah apply, AWS akan kirim email konfirmasi ke info@widianto.org. Harus diklik "Confirm Subscription".

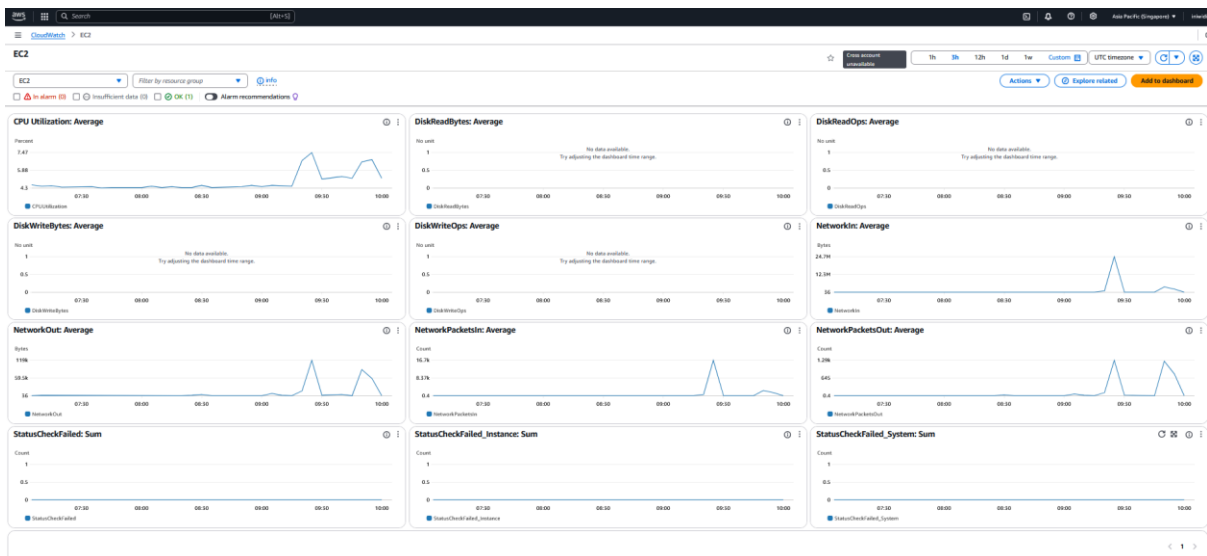
✓ monitoring.tf – CloudWatch Alarm untuk EC2

```
# CloudWatch Alarm for EC2 CPU > 75%
resource "aws_cloudwatch_metric_alarm" "high_cpu_alarm" {
  alarm_name      = "high-cpu-ec2"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods = 2
  metric_name     = "CPUUtilization"
  namespace      = "AWS/EC2"
  period         = 60
  statistic       = "Average"
  threshold       = 75

  alarm_description = "This alarm triggers when EC2 CPU > 75% for 2 minutes"
  alarm_actions     = [aws_sns_topic.cpu_alerts.arn]

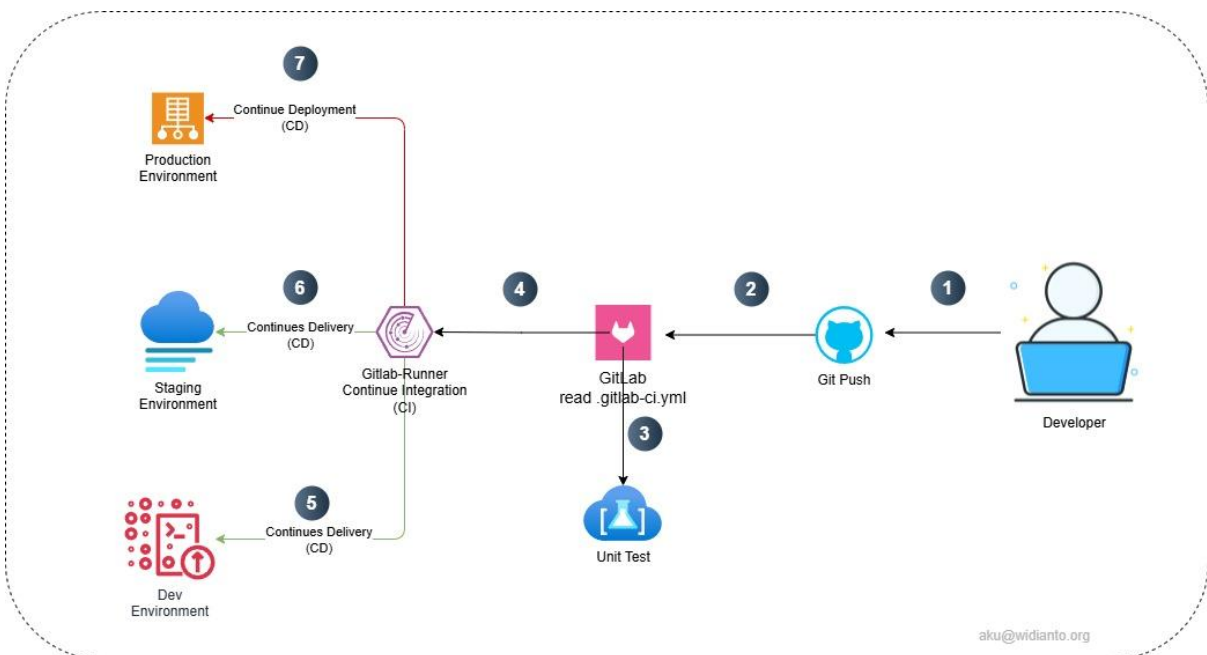
  dimensions = {
    InstanceId = aws_instance.web.id
  }
}
```


✓ Evidence Monitoring EC2



✖ Understanding of CI/CD concepts

Design Architecture



Gambar aliran proses CI/CD/CD, termasuk tahapan penting **Unit Test** di tahap ke-3. Berikut penjelasannya:

🔄 Penjelasan Alur CI/CD


1. Developer Menulis Kode

- Developer melakukan coding atau perubahan fitur pada project di lokal mesin.

2. Git Push

- Developer melakukan git push ke repository Git (misalnya GitLab).
- Ini akan memicu pipeline CI/CD secara otomatis.

3. Unit Test

- Setelah GitLab menerima perubahan, pipeline menjalankan **Unit Test** terlebih dahulu.
-  **Tujuan Unit Test:**
 - Untuk **memverifikasi bahwa setiap fungsi/unit bekerja sesuai harapan**.
 - Mendeteksi bug lebih awal sebelum kode masuk tahap berikutnya (CI/CD).
 - Menjamin bahwa perubahan kode **tidak merusak fitur yang sudah ada** (regression).

4. GitLab Baca .gitlab-ci.yml dan Panggil GitLab Runner

- File .gitlab-ci.yml memuat konfigurasi pipeline seperti:
 - Tahap build, test, deploy
 - Environment target (dev, staging, prod)
- GitLab kemudian menjalankan perintah-perintah tersebut melalui **GitLab Runner**.

5. Continuous Delivery ke Dev Environment

- Jika Unit Test berhasil, kode dideploy ke **Dev Environment** untuk pengujian internal lebih lanjut (UI/UX, integrasi, dll).

6. Continuous Delivery ke Staging Environment

- Setelah Dev lolos, kode dilanjutkan ke **Staging Environment** untuk UAT (User Acceptance Testing).
- Ini mensimulasikan kondisi production untuk validasi akhir.

7. Continuous Deployment ke Production

- Setelah semua tahapan sukses dan (bila perlu) mendapat persetujuan manual, kode **dideploy ke production secara otomatis**.

Kenapa Unit Test Penting di Tahap ke-3?

1. Deteksi Dini Masalah Kode:

- Menemukan bug sebelum kode lanjut ke proses build atau deploy.

2. Menghindari Kerusakan Berantai:

- Gagal di unit test = pipeline berhenti → mencegah error sampai ke production.

3. Mengurangi Biaya Perbaikan Bug:

- Bug yang ditemukan lebih awal akan jauh lebih murah diperbaiki daripada saat sudah di production.

4. Menjaga Kepercayaan Tim dan DevOps:

- Pipeline yang dilengkapi test lebih dapat dilkan untuk deployment otomatis.

Kesimpulan

Dengan penambahan **Unit Test di tahap ke-3**, proses CI/CD menjadi lebih robust, aman, dan siap untuk praktik DevOps modern. Tahapan ini membantu memastikan kualitas kode tetap terjaga secara otomatis sebelum dilanjutkan ke tahap pengiriman (*delivery*) dan penerapan (*deployment*).