**Automated YouTube Analytics Dashboard**

This documentation walks you through building a **fully automated YouTube Analytics Dashboard** using:

* **Terraform** (IaC for provisioning AWS services)
* **AWS**: Lambda, S3, EC2 (Grafana), CloudWatch
* **Python**: Boto3, requests
* **YouTube Data API v3**

Your project will:

* Automatically fetch analytics data from YouTube
* Store JSON in S3 bucket
* Visualize data via Grafana on EC2
* Deploy everything via Terraform

**Prerequisites**

✅ You must have:

* AWS CLI configured with IAM user (aws configure)
* Admin access to AWS account
* Visual Studio Code with extensions: Python, Terraform, AWS
* Git & GitHub setup

**1. Project Folder Structure**

youtube-dashboard/

├── lambda/

│ ├── lambda\_function.py

│ ├── requirements.txt

│ ├── .env # Will be excluded from GitHub

│ ├── .env.example # For GitHub reference

│ └── zip\_me.sh # ZIP packager script

│

├── terraform/

│ ├── main.tf

│ ├── variables.tf

│ ├── terraform.tfvars

│ ├── outputs.tf

│ ├── iam-role.json

│

├── .gitignore

└── README.md

**2. Lambda Function Code**

**lambda/lambda\_function.py**

import boto3

import requests

import os

from datetime import datetime

from dotenv import load\_dotenv

load\_dotenv()

def lambda\_handler(event=None, context=None):

api\_key = os.getenv("YOUTUBE\_API\_KEY")

channel\_id = os.getenv("CHANNEL\_ID")

s3\_bucket = os.getenv("S3\_BUCKET\_NAME")

url = f"https://www.googleapis.com/youtube/v3/channels?part=statistics&id={channel\_id}&key={api\_key}"

response = requests.get(url)

data = response.json()

filename = f"youtube-stats-{datetime.utcnow().isoformat()}.json"

s3 = boto3.client("s3")

s3.put\_object(Bucket=s3\_bucket, Key=filename, Body=str(data))

return {

"statusCode": 200,

"body": f"Uploaded: {filename}"

}

**lambda/requirements.txt**

boto3

requests

python-dotenv

**lambda/.env.example (DO NOT commit .env)**

YOUTUBE\_API\_KEY=your\_key\_here

CHANNEL\_ID=your\_channel\_id\_here

S3\_BUCKET\_NAME=your\_s3\_bucket\_name

**lambda/zip\_me.sh**

#!/bin/bash

rm -f lambda.zip

cd lambda

python3 -m venv .venv

source .venv/bin/activate

pip install -r requirements.txt

deactivate

cd .venv/lib/python3.\*/site-packages

zip -r9 ../../../../lambda.zip .

cd ../../../../

zip -g lambda.zip lambda/lambda\_function.py

Make it executable:

chmod +x lambda/zip\_me.sh

Run it with:

./lambda/zip\_me.sh

**3. Terraform Setup**

**terraform/variables.tf**

variable "s3\_bucket" {}

variable "youtube\_api\_key" {}

variable "channel\_id" {}

variable "key\_pair\_name" {}

**terraform/terraform.tfvars**

s3\_bucket = "your-s3-bucket-name"

youtube\_api\_key = "from-.env"

channel\_id = "from-.env"

key\_pair\_name = "your-ec2-keypair"

**terraform/iam-role.json**

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "lambda.amazonaws.com"

},

"Action": "sts:AssumeRole"

}

]

}

**terraform/main.tf**

provider "aws" {

region = "us-east-1"

}

resource "aws\_s3\_bucket" "youtube" {

bucket = var.s3\_bucket

}

resource "aws\_iam\_role" "lambda\_exec" {

name = "lambda-exec-role"

assume\_role\_policy = file("iam-role.json")

}

resource "aws\_iam\_role\_policy\_attachment" "attach\_policy" {

role = aws\_iam\_role.lambda\_exec.name

policy\_arn = "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"

}

resource "aws\_lambda\_function" "youtube\_fetch" {

function\_name = "youtube-data-fetcher"

role = aws\_iam\_role.lambda\_exec.arn

handler = "lambda\_function.lambda\_handler"

runtime = "python3.9"

filename = "../lambda/lambda.zip"

environment {

variables = {

YOUTUBE\_API\_KEY = var.youtube\_api\_key

CHANNEL\_ID = var.channel\_id

S3\_BUCKET\_NAME = var.s3\_bucket

}

}

}

resource "aws\_cloudwatch\_event\_rule" "trigger" {

name = "every-6h"

schedule\_expression = "rate(6 hours)"

}

resource "aws\_cloudwatch\_event\_target" "target" {

rule = aws\_cloudwatch\_event\_rule.trigger.name

arn = aws\_lambda\_function.youtube\_fetch.arn

}

resource "aws\_lambda\_permission" "allow" {

statement\_id = "AllowExecutionFromCloudWatch"

action = "lambda:InvokeFunction"

function\_name = aws\_lambda\_function.youtube\_fetch.function\_name

principal = "events.amazonaws.com"

source\_arn = aws\_cloudwatch\_event\_rule.trigger.arn

}

resource "aws\_security\_group" "grafana\_sg" {

name = "grafana-sg"

ingress {

from\_port = 3000

to\_port = 3000

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

}

resource "aws\_instance" "grafana" {

ami = "ami-0c2b8ca1dad447f8a" # Amazon Linux 2

instance\_type = "t2.micro"

key\_name = var.key\_pair\_name

security\_groups = [aws\_security\_group.grafana\_sg.name]

user\_data = <<-EOF

#!/bin/bash

sudo yum install -y amazon-linux-extras

sudo amazon-linux-extras enable epel

sudo yum install -y grafana

sudo systemctl enable grafana-server

sudo systemctl start grafana-server

EOF

}

**terraform/outputs.tf**

output "grafana\_public\_ip" {

value = aws\_instance.grafana.public\_ip

}

**4. Deploy with Terraform**

cd terraform

terraform init

terraform apply -auto-approve

Note the grafana\_public\_ip from output. Open http://<ip>:3000 in browser. Default login:

* user: admin
* pass: admin

**5. Final GitHub Setup**

**.gitignore**

lambda/.venv/

lambda/.env

lambda/lambda.zip

.terraform/

terraform.tfstate\*

**README.md**

Add:

* Project purpose
* Architecture diagram
* Tools used
* Setup instructions

Let me know if you want me to generate a ZIP folder or GitHub-ready push with commit structure!

yes genearte a zip folder