📡 MQTT with MongoDB, Python and Terraform

This project demonstrates a simple telemetry API written in Python, using MongoDB as a database and various DevOps technologies.

Build coverage 74%

📑 Table of Contents

💻 Running Locally

With Python and MongoDB

1. Make sure you have Python and MongoDB installed.
2. Create and activate the virtual environment:

```
cd mqtt-mongo-python
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
```

3. Run the application:

```
make run
```

> To change MQTT broker or MongoDB credentials, edit the `.env` file in `app/env`.

With Docker and Compose

Use the provided `Makefile` to run the project with Docker:

```
make docker
```

To remove all containers, volumes, and images:

```
make docker-clear
```

🧪 Testing

- Run tests:

```
make test
```

- Run tests with coverage report:

```
make coverage
```

> Coverage HTML report will be saved in `html/` by default.

⚙️ Project Configuration with Terraform on AWS

🔐 SSH Keys

Generate the required keys:

```
ssh-keygen -t rsa -b 4096 -m PEM -f key_ec2_dev -N ""
ssh-keygen -t rsa -b 4096 -m PEM -f key_ec2_prod -N ""
ssh-keygen -t rsa -b 4096 -m PEM -f key_github -N ""
```

> You may add a `.pem` extension to the EC2 key if you prefer. You will find the keys in the `.ssh` folder in your user directory. At the end, you'll have:

- key_ec2_dev.pem - Private key to connect to the developer EC2 instance:

```
ssh -i "key_ec2_dev.pem" ubuntu@ec2-[YOUR_IP].sa-east-
1.compute.amazonaws.com
```

- key_ec2_prod.pem - Private key to connect to the production EC2 instance:

```
ssh -i "key_ec2_prod.pem" ubuntu@ec2-[YOUR_IP].sa-east-
1.compute.amazonaws.com
```

- key_ec2_dev.pub - Public key registered in AWS for the development EC2
- key_ec2_prod.pub - Public key registered in AWS for the production EC2
- key_github - Private key injected into the EC2 by Terraform to clone the GitHub repository
- key_github.pub - Public key registered in GitHub under Deploy Keys to allow clone access

🌐 Elastic IP

This project uses fixed IPs via AWS Elastic IP. Manually create two Elastic IPs in the AWS console, one for development and one for production. Save the Allocation IDs to use later.

📣 Slack Webhook

This project sends notifications to Slack for Dependabot, GitHub Actions, and CloudWatch SNS alerts. Slack's documentation explains how to create a channel and a webhook.

🔑 GitHub Secrets

Go to: **Settings > Secrets and variables > Actions > Secrets**, and add:

- `DEV_EC2_PUBLIC_IP`
  - Paste the Allocation ID of the development Elastic IP
- `PROD_EC2_PUBLIC_IP`
  - Paste the Allocation ID of the production Elastic IP
- `DEV_EC2_SSH_PRIVATE_KEY`
  - Paste the contents of the key_ec2_dev.pem private key `key_ec2_dev.pem`
- `PROD_EC2_SSH_PRIVATE_KEY`
  - Paste the contents of the key_ec2_prod.pem private key `key_ec2_prod.pem`
- `SLACK_WEBHOOK_URL`
  - Your Slack webhook URL

☁️ AWS IAM - Identity Providers

- Go to **IAM > Identity providers > Add provider [OpenID Connect]**
- Provider URL: `https://token.actions.githubusercontent.com`
- Audience: `sts.amazonaws.com`

🔐 AWS IAM - Assign Role to the Identity Provider

- Access the Identity Provider again, click on the previously registered provider, and click on **Assign Role**
- Keep "Create a new role" selected and click on "next"
- Keep Web Identity selected
- Select the previously configured Audience (`sts.amazonaws.com`)
- Enter your GitHub organization name (or your GitHub username)
- On the next screen, add the following policies:
  - `AmazonS3FullAccess`
  - `AmazonDynamoDBFullAccess`
  - `AmazonEC2FullAccess`
  - `CloudWatchAgentServerPolicy`
  - `IAMFullAccess`
  - `AmazonSNSFullAccess`
  - `AWSLambda_FullAccess`

🪣 AWS S3 - Statefile

Create a bucket with versioning enabled.

🧱 AWS DynamoDB

Create a table with Partition Key: `LockID`

⚙️ GitHub Actions Workflow Variables

For GitHub Actions to work properly, use these variables in the workflow files located at `.github/workflows`:

- `aws-assume-role-arn`: e.g., `arn:aws:iam::123456789012:role/your-role-name`
- `aws-region`: e.g., `sa-east-1`
- `aws-statefile-s3-bucket`: name of your S3 bucket
- `aws-lock-dynamodb-table`: name of your DynamoDB table

⚙️ Terraform Variables

For Terraform to work properly, use these variables in the configuration files located at `terraform/envs/`:

- `ami_id`
  - Set the AMI ID for the OS image to be used, e.g., `ami-0a174b8e659123575` # Ubuntu 22.04 Free Tier - sa-east-1
- `instance_type`
  - Set the instance type, e.g., `t2.micro`
- `eip_id`
  - Paste the Elastic IP allocation ID, usually starting with `eipalloc-...`
- `branch`
  - Set the name of the branch

📖 API Docs

🔌 Endpoints

- **Without Docker:**
  - http://localhost:3000/telemetry
  - http://localhost:3000/docs
- **With Docker:**
  - http://localhost:3001/telemetry
  - http://localhost:3001/docs
- **On AWS (Docker):**
  - http://[YOUR_IP]:3001/telemetry
  - http://[YOUR_IP]:3001/docs

📚 Generate HTML Documentation (with `pdoc`)

To generate developer-friendly HTML documentation from the source code using `pdoc`:

```
make doc
```

📄 Logs

```
# Cloud Init log (executed on instance creation)
sudo cat /var/log/cloud-init-output.log

# CloudWatch Agent log
sudo cat /opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log
```

☁️ Destroy Config

Edit the `destroy_config.json` file inside the `terraform/` folder to decide whether to destroy the environment:

```
{
  "dev": false,
  "prod": true
}
```