Hw2 – Terraform and Docker

Part II:

Step1: Download main.tf and terraform.tfvars file

Step2: Update terraform.tfvars file with ssh_cidr , ssh_key_name

Step3: Configure AWS – session using aws configure

Step4: Set the session using aws configure set aws_session_token <YOUR SESSION TOKEN>

Deploying Infrastructure

Step1: Initialize terraform

Step2: terraform apply -auto-approve

Step3: Connect via ssh

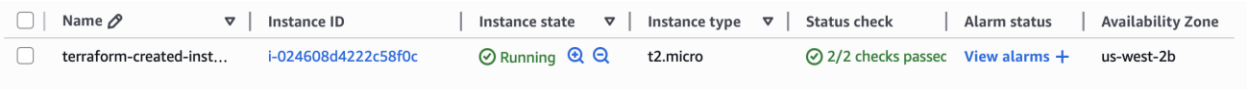


Fig1: Running EC2 instance created using terraform

Step4: Cleanup

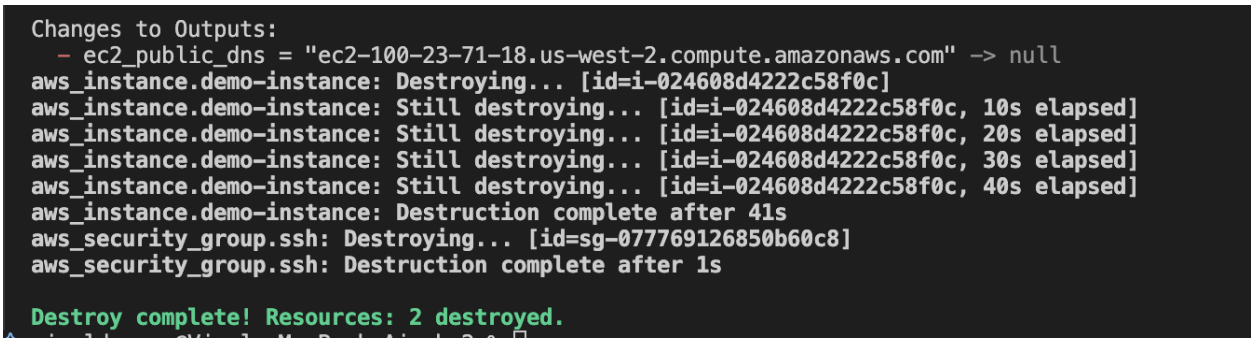


Fig2: Logs of successful resource destruction

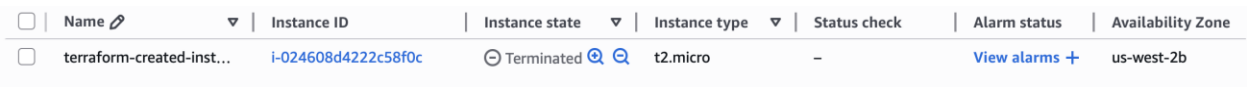


Fig3: Ec2 Terminated on AWS Console

Part III:

Build container and connect locally

Step1 : Create main.go file using the hw1 and update router.Run("localhost:8080") to router.Run(":8080")

Step2: Install Docker Desktop

Step3 : Create docker file

Step4: Build docker container

```
vinaldsouza@Vinals-MacBook-Air hw2 % docker build -t hw2 .
[+] Building 37.7s (13/13) FINISHED                                 docker:desktop-linux
```

Step5: Run the container

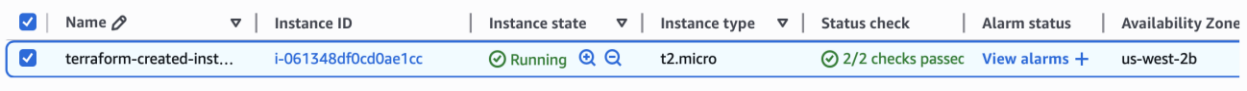
```
vinaldsouza@Vinals-MacBook-Air hw2 % docker run -p 8080:8080 hw2
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already att
ached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /albums                   --> main.getAlbums (3 handlers)
[GIN-debug] GET    /albums/:id               --> main.getAlbumByID (3 handlers)
[GIN-debug] POST   /albums                   --> main.postAlbums (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://github.com/gin-gonic/gin/blob/master/docs/doc.md#dont-trust-all-proxies for deta
ils.
[GIN-debug] Listening and serving HTTP on :8080
[GIN] 2026/01/23 - 23:41:07 | 200 |     563.125µs |    192.168.65.1 | GET      "/albums"
```

Step6: Test the connection to the application

```
vinaldsouza@Vinals-MacBook-Air hw2 % curl http://localhost:8080/albums
[
    {
        "id": "1",
        "title": "Blue Train",
        "artist": "John Coltrane",
        "price": 56.99
    },
    {
        "id": "2",
        "title": "Jeru",
        "artist": "Gerry Mulligan",
        "price": 17.99
    },
    {
        "id": "3",
        "title": "Sarah Vaughan and Clifford Brown",
        "artist": "Sarah Vaughan",
        "price": 39.99
    }
]%
```

Create EC2 and build docker remotely on cloud

Step1: Run the terraform commands as in part II.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
terraform-created-inst...	i-061348df0cd0ae1cc	Running	t2.micro	2/2 checks passec	View alarms +	us-west-2b

Fig4: Ec2 up and running

Step2: SSH into EC2

Step3: Install git

```
[ec2-user@ip-172-31-40-187 ~]$ sudo yum install git -y
Last metadata expiration check: 0:00:45 ago on Sat Jan 24 00:15:07 2026.
Dependencies resolved.
================================================================================
 Package              Architecture    Version                        Repository          Size
================================================================================
Installing:
 git                  x86_64          2.50.1-1.amzn2023.0.1          amazonlinux          53 k
Installing dependencies:
 git-core             x86_64          2.50.1-1.amzn2023.0.1          amazonlinux         4.9 M
 git-core-doc         noarch          2.50.1-1.amzn2023.0.1          amazonlinux         2.8 M
 perl-Error           noarch          1:0.17029-5.amzn2023.0.2       amazonlinux          41 k
 perl-Git             noarch          2.50.1-1.amzn2023.0.1          amazonlinux          41 k

Transaction Summary
================================================================================
Install  5 Packages

Total download size: 7.9 M
Installed size: 41 M
Downloading Packages:
(1/5): git-2.50.1-1.amzn2023.0.1.x86_64.rpm                  1.1 MB/s |  53 kB     00:00
(2/5): git-core-2.50.1-1.amzn2023.0.1.x86_64.rpm              47 MB/s | 4.9 MB     00:00
(3/5): perl-Error-0.17029-5.amzn2023.0.2.noarch.rpm          676 kB/s |  41 kB     00:00
(4/5): perl-Git-2.50.1-1.amzn2023.0.1.noarch.rpm             1.3 MB/s |  41 kB     00:00
(5/5): git-core-doc-2.50.1-1.amzn2023.0.1.noarch.rpm          18 MB/s | 2.8 MB     00:00
--------------------------------------------------------------------------------
Total                                                         38 MB/s | 7.9 MB     00:00
Running transaction check
```

Step4: Install docker

```
[ec2-user@ip-172-31-40-187 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:01:12 ago on Sat Jan 24 00:15:07 2026.
Package docker-25.0.14-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-40-187 ~]$ docker --version
Docker version 25.0.14, build 0bab007
```

Step5: Git clone

```
[ec2-user@ip-172-31-40-187 ~]$ git clone https://github.com/vinaldsz/scalable-distributed-systems.git
Cloning into 'scalable-distributed-systems'...
```

```
[ec2-user@ip-172-31-40-187 ~]$ ls
scalable-distributed-systems
```

Step6: Build docker container

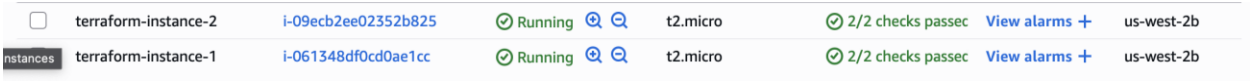
Step7: Run container

Step8: Test

```
vinaldsouza@Vinals-MacBook-Air hw2 % curl http://localhost:8080/albums
[
    {
        "id": "1",
        "title": "Blue Train",
        "artist": "John Coltrane",
        "price": 56.99
    },
    {
        "id": "2",
        "title": "Jeru",
        "artist": "Gerry Mulligan",
        "price": 17.99
    },
    {
        "id": "3",
        "title": "Sarah Vaughan and Clifford Brown",
        "artist": "Sarah Vaughan",
        "price": 39.99
    }
]%
```

Fig6: Accessing the ec2 and app within docker remotely

Part IV



Output of the test script

Starting data test...

Instance 1 response: [


```
  {
    "id": "1",
    "title": "Blue Train",
    "artist": "John Coltrane",
    "price": 56.99
  },
  {
    "id": "2",
    "title": "Jeru",
    "artist": "Gerry Mulligan",
    "price": 17.99
  },
  {
    "id": "3",
    "title": "Sarah Vaughan and Clifford Brown",
    "artist": "Sarah Vaughan",
    "price": 39.99
  }
]
```


and...

Instance 2 response: [


```
  {
    "id": "1",
    "title": "Blue Train",
    "artist": "John Coltrane",
    "price": 56.99
  },
  {
    "id": "2",
    "title": "Jeru",
    "artist": "Gerry Mulligan",
    "price": 17.99
  },
  {
    "id": "3",
    "title": "Sarah Vaughan and Clifford Brown",
    "artist": "Sarah Vaughan",
    "price": 39.99
  }
]
```


and adding...

Instance 2 response: {


```
  "id": "4",
  "title": "The Modern Sound of Betty Carter",
  "artist": "Betty Carter",
  "price": 49.99
}
```

Instance 1 response: [


```
  {
    "id": "1",
    "title": "Blue Train",
    "artist": "John Coltrane",
    "price": 56.99
  },
  {
    "id": "2",
    "title": "Jeru",
    "artist": "Gerry Mulligan",
    "price": 17.99
  },
  {
    "id": "3",
    "title": "Sarah Vaughan and Clifford Brown",
    "artist": "Sarah Vaughan",
    "price": 39.99
  }
]
```


and...

Instance 2 response: [


```
  {
    "id": "1",
    "title": "Blue Train",
    "artist": "John Coltrane",
    "price": 56.99
```

```
  },
  {
    "id": "2",
    "title": "Jeru",
    "artist": "Gerry Mulligan",
    "price": 17.99
  },
  {
    "id": "3",
    "title": "Sarah Vaughan and Clifford Brown",
    "artist": "Sarah Vaughan",
    "price": 39.99
  },
  {
    "id": "4",
    "title": "The Modern Sound of Betty Carter",
    "artist": "Betty Carter",
    "price": 49.99
  }
]
```


uhoh... what happened?

Comments:

1. Both start with same album
2. We add #4 in the Instance 2
3. Instance 2 shows 4 albums
4. Instance1 only has 3 albums

There is no shared state between the instances both applications run in isolation from each other and with no common shared data.

Part IV

1. Amount of time spent : 35 mins (Had not seen piazza and was not sure on why I was getting permission issues)
2. The mystery bug was with go_func() that was called within the for loop causing RACE condition. Some writes were missed since some other parallel processes read the count value while its being written by another process. I found the bug in the very first prompt when I asked to explain the project to me. However, I dug deeper on why this happened and how can we ensure that we can get a right count – it suggested using locks. A process writing would hold a exclusive lock which then will be released after completion.
3. For jsonl files refer to github