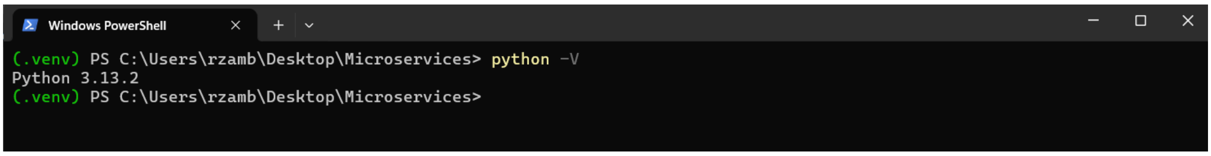
AIM4:

To develop and deploy microservices and serverless functions.

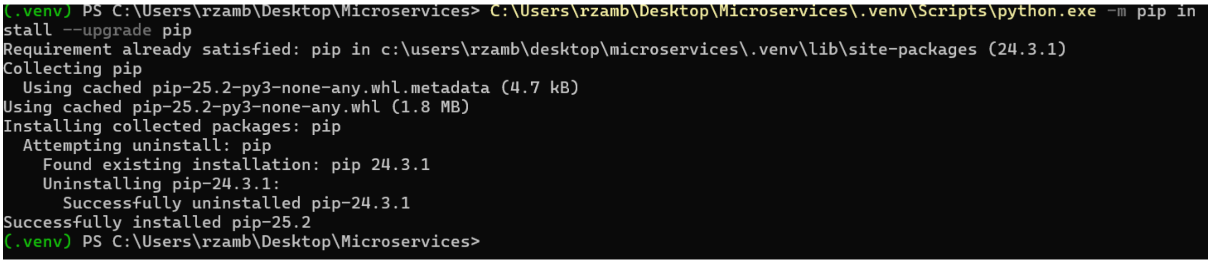
Step-by-Step Implementation with Supporting Screenshots:

Step 1: Check for latest Python version



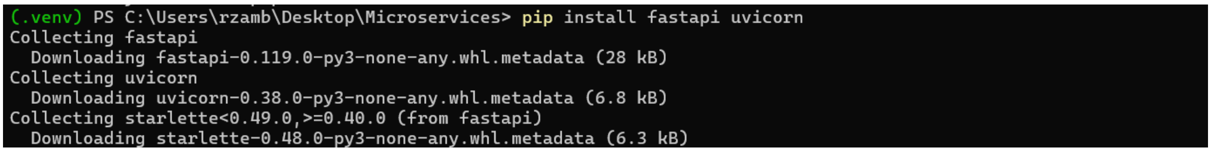
```
(.venv) PS C:\Users\rzamb\Desktop\Microservices> python -V
Python 3.13.2
(.venv) PS C:\Users\rzamb\Desktop\Microservices>
```

Step 2: Initialize .venv machine to avoid VPN issues and safe local execution.



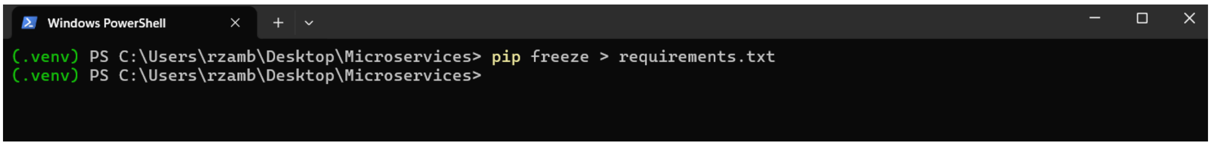
```
(.venv) PS C:\Users\rzamb\Desktop\Microservices> C:\Users\rzamb\Desktop\Microservices\.venv\Scripts\python.exe -m pip in
stall --upgrade pip
Requirement already satisfied: pip in c:\users\rzamb\desktop\microservices\.venv\lib\site-packages (24.3.1)
Collecting pip
  Using cached pip-25.2-py3-none-any.whl.metadata (4.7 kB)
Using cached pip-25.2-py3-none-any.whl (1.8 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.3.1
    Uninstalling pip-24.3.1:
      Successfully uninstalled pip-24.3.1
Successfully installed pip-25.2
(.venv) PS C:\Users\rzamb\Desktop\Microservices>
```

Step 3: Install Fastapi and uvicorn, to design endpoints easily and to run the fastapi.



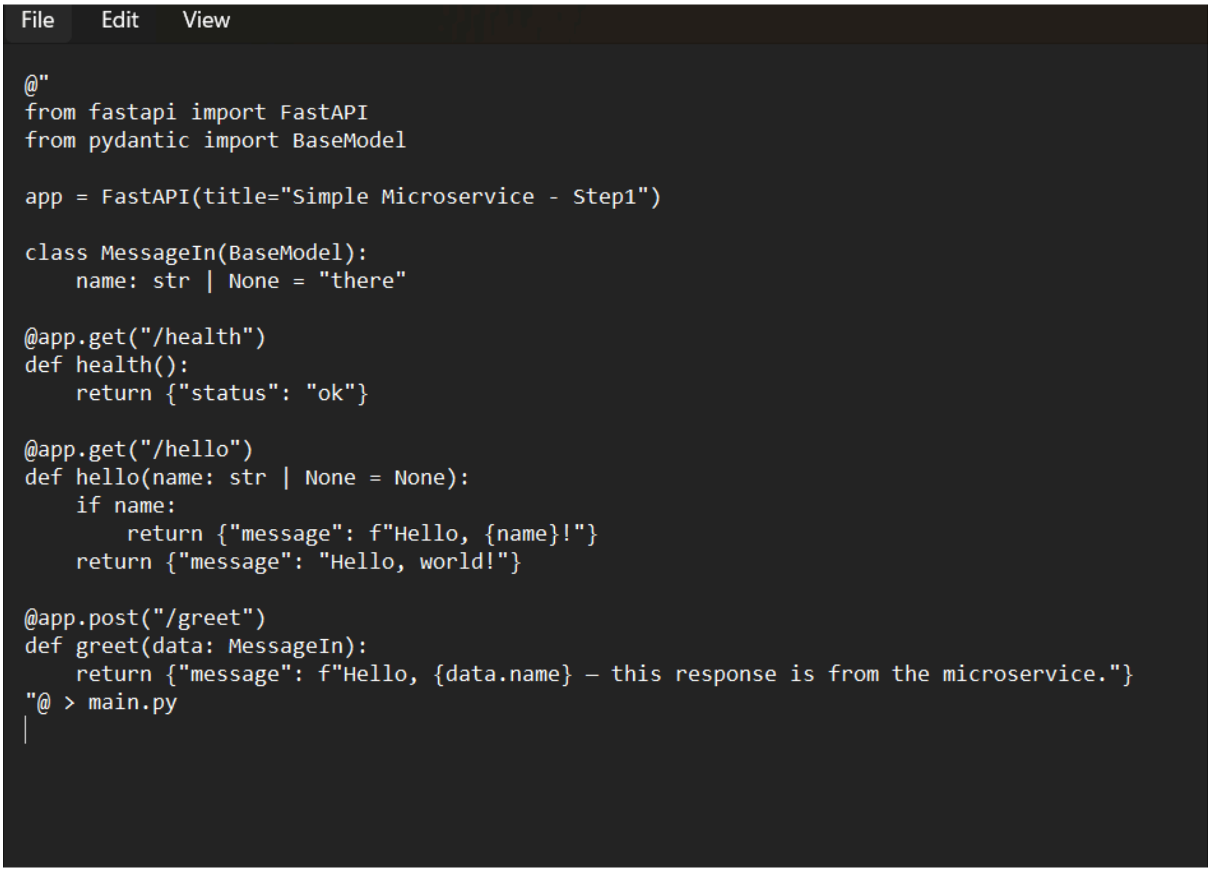
```
(.venv) PS C:\Users\rzamb\Desktop\Microservices> pip install fastapi uvicorn
Collecting fastapi
  Downloading fastapi-0.119.0-py3-none-any.whl.metadata (28 kB)
Collecting uvicorn
  Downloading uvicorn-0.38.0-py3-none-any.whl.metadata (6.8 kB)
Collecting starlette<0.49.0,>=0.40.0 (from fastapi)
  Downloading starlette-0.48.0-py3-none-any.whl.metadata (6.3 kB)
```

Step 4: Make a separate requirements file



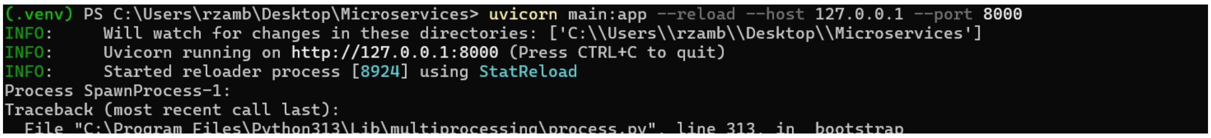
```
(.venv) PS C:\Users\rzamb\Desktop\Microservices> pip freeze > requirements.txt
(.venv) PS C:\Users\rzamb\Desktop\Microservices>
```

Step 5: Make a Basic Python application which works like a microservice for our assignement



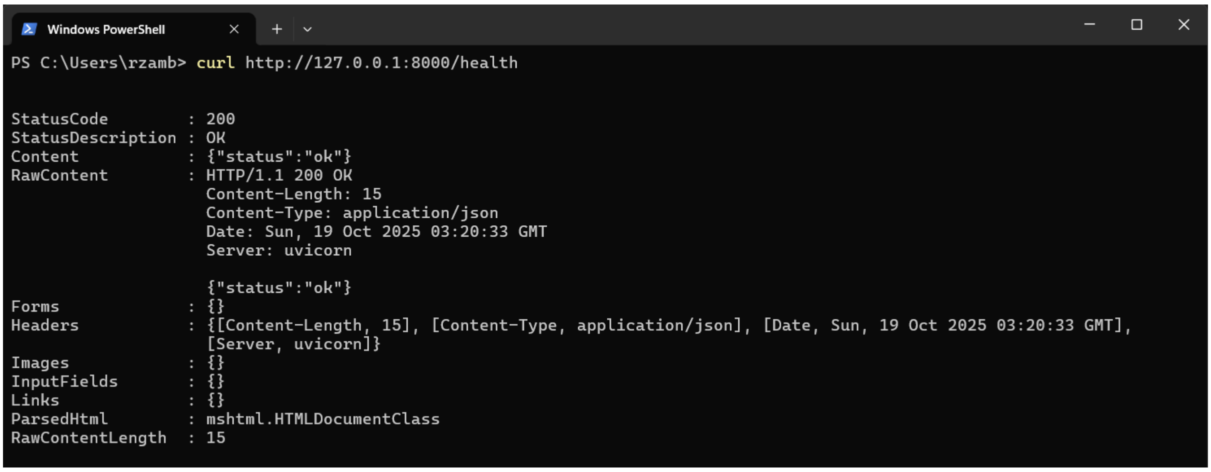
```
@"
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI(title="Simple Microservice - Step1")

class MessageIn(BaseModel):
    name: str | None = "there"

@app.get("/health")
def health():
    return {"status": "ok"}

@app.get("/hello")
def hello(name: str | None = None):
    if name:
        return {"message": f"Hello, {name}!"}
    return {"message": "Hello, world!"}

@app.post("/greet")
def greet(data: MessageIn):
    return {"message": f"Hello, {data.name} — this response is from the microservice."}
"@ > main.py
```

Step 6: Run and test locally



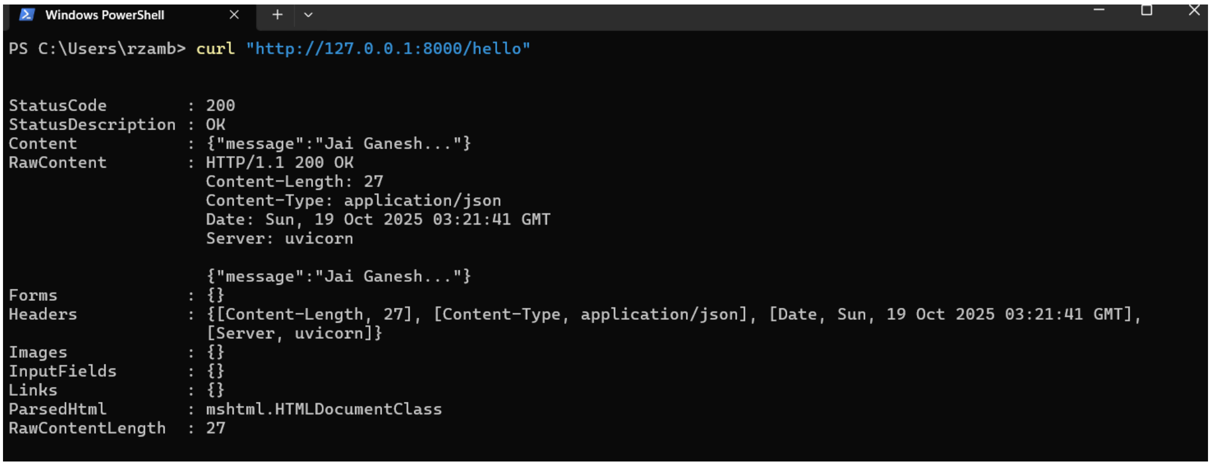
```
(.venv) PS C:\Users\rzamb\Desktop\Microservices> uvicorn main:app --reload --host 127.0.0.1 --port 8000
INFO:     Will watch for changes in these directories: ['C:\\Users\\rzamb\\Desktop\\Microservices']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [8924] using StatReload
Process SpawnProcess-1:
Traceback (most recent call last):
  File "C:\Program Files\Python313\Lib\multiprocessing\process.py", line 313, in _bootstrap
```

Step 7: Health Check for application

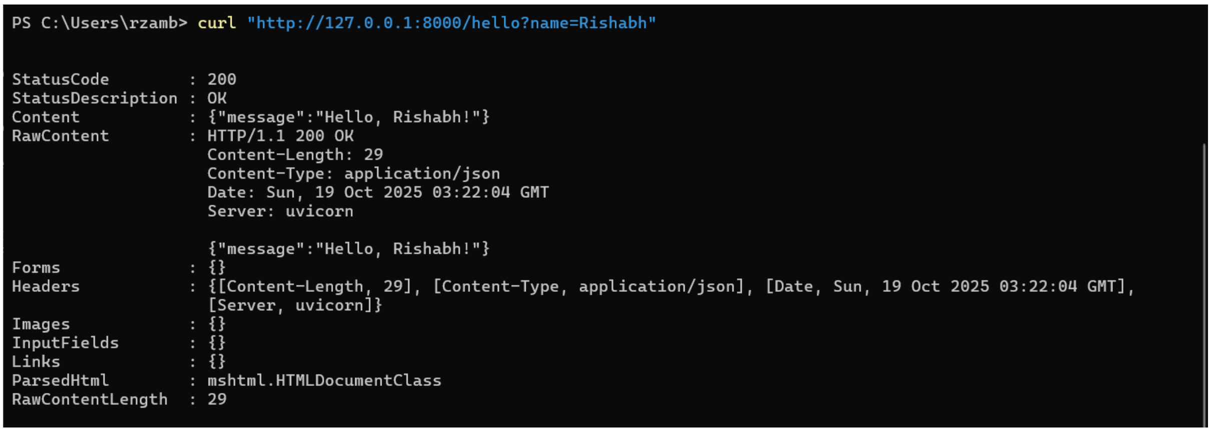


```
PS C:\Users\rzamb> curl http://127.0.0.1:8000/health


StatusCode        : 200
StatusDescription : OK
Content           : {"status":"ok"}
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 15
                    Content-Type: application/json
                    Date: Sun, 19 Oct 2025 03:20:33 GMT
                    Server: uvicorn

                    {"status":"ok"}
Forms             : {}
Headers           : {[Content-Length, 15], [Content-Type, application/json], [Date, Sun, 19 Oct 2025 03:20:33 GMT],
                    [Server, uvicorn]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 15
```

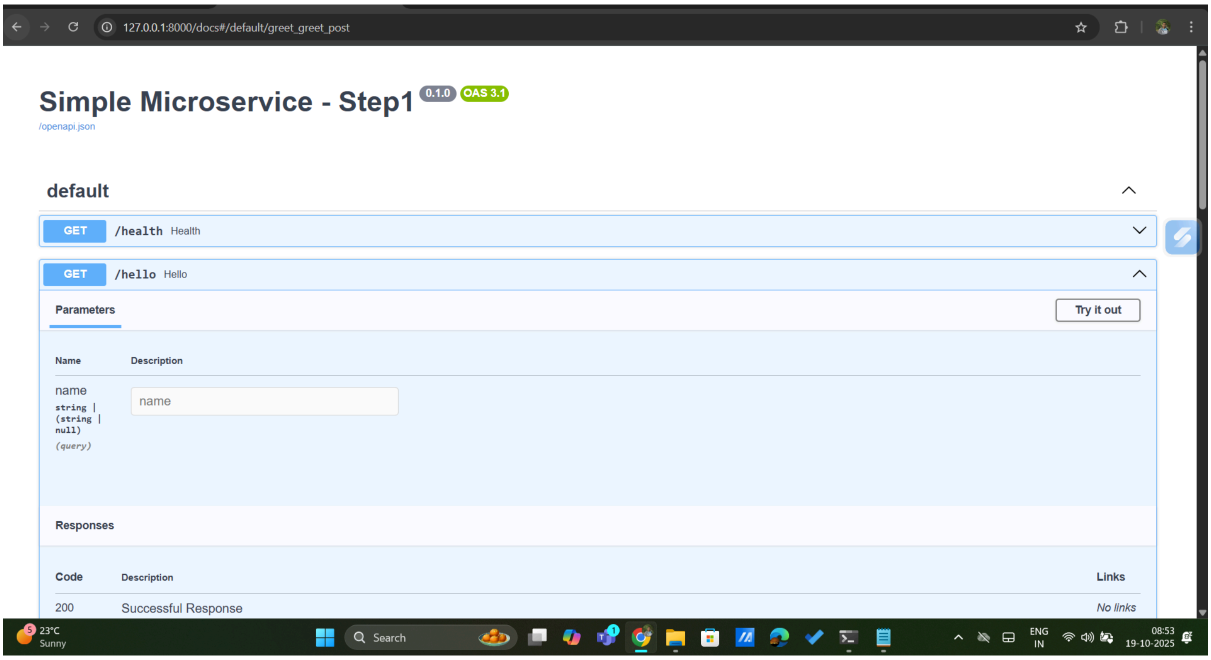
Step 8: Testing of Application



```
PS C:\Users\rzamb> curl "http://127.0.0.1:8000/hello"


StatusCode        : 200
StatusDescription : OK
Content           : {"message":"Jai Ganesh..."}
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 27
                    Content-Type: application/json
                    Date: Sun, 19 Oct 2025 03:21:41 GMT
                    Server: uvicorn

                    {"message":"Jai Ganesh..."}
Forms             : {}
Headers           : {[Content-Length, 27], [Content-Type, application/json], [Date, Sun, 19 Oct 2025 03:21:41 GMT],
                    [Server, uvicorn]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 27
```



```
PS C:\Users\rzamb> curl "http://127.0.0.1:8000/hello?name=Rishabh"


StatusCode        : 200
StatusDescription : OK
Content           : {"message":"Hello, Rishabh!"}
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 29
                    Content-Type: application/json
                    Date: Sun, 19 Oct 2025 03:22:04 GMT
                    Server: uvicorn

                    {"message":"Hello, Rishabh!"}
Forms             : {}
Headers           : {[Content-Length, 29], [Content-Type, application/json], [Date, Sun, 19 Oct 2025 03:22:04 GMT],
                    [Server, uvicorn]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 29
```



127.0.0.1:8000/docs#/default/greet_greet_post

Simple Microservice - Step1 0.1.0 OAS 3.1

/openapi.json

default

GET /health Health

GET /hello Hello

Parameters

Name Description

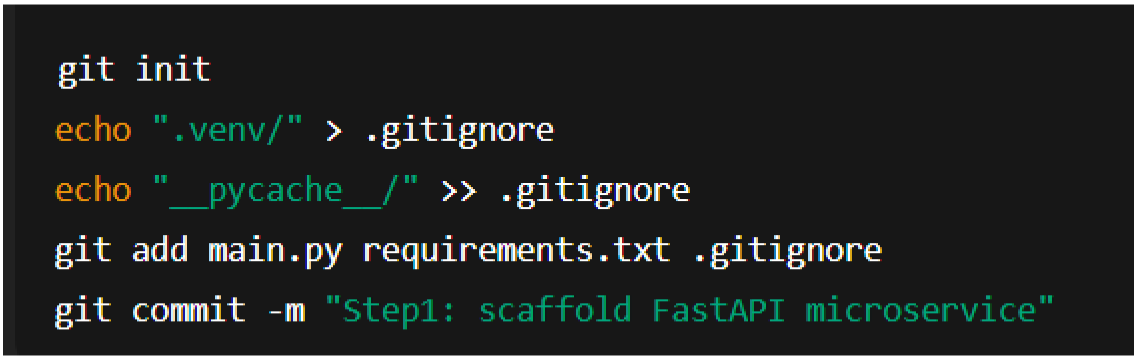
name string | (string | null) (query) — name

Responses

Code Description Links

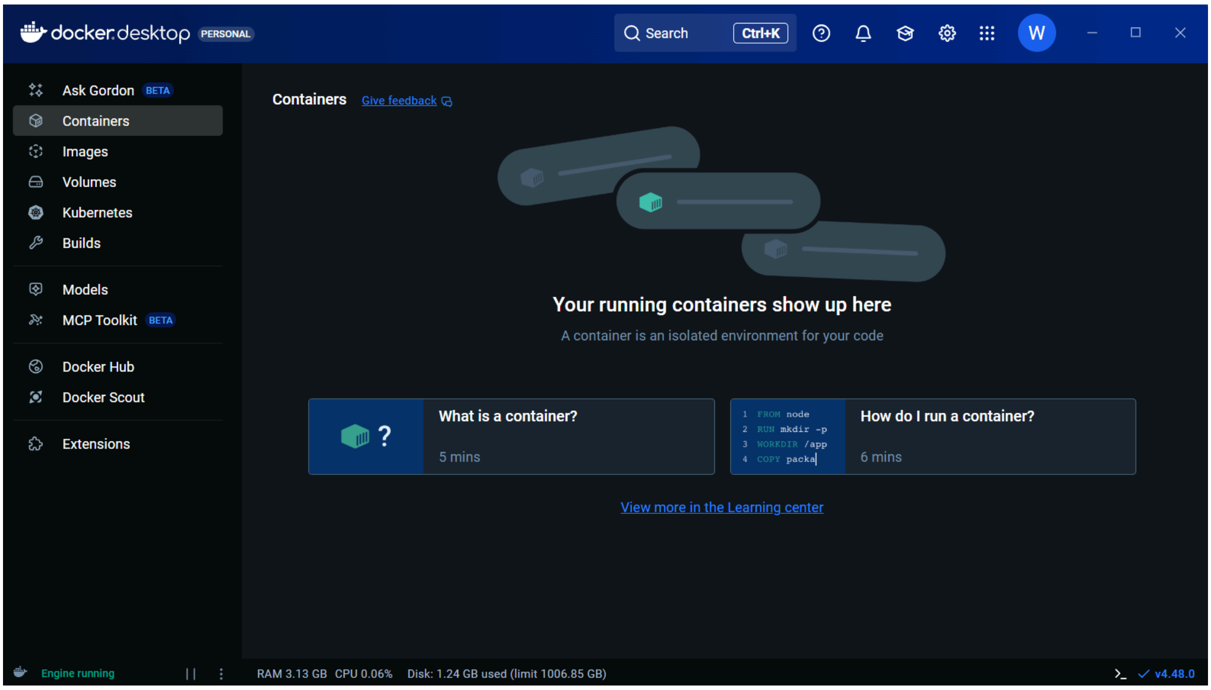
200 Successful Response No links

Step 9: (optional) uploading to local git for showing version control



```
git init
echo ".venv/" > .gitignore
echo "__pycache__/" >> .gitignore
git add main.py requirements.txt .gitignore
git commit -m "Step1: scaffold FastAPI microservice"
```

Step 10: Installing Docker Desktop and running Docker Engine



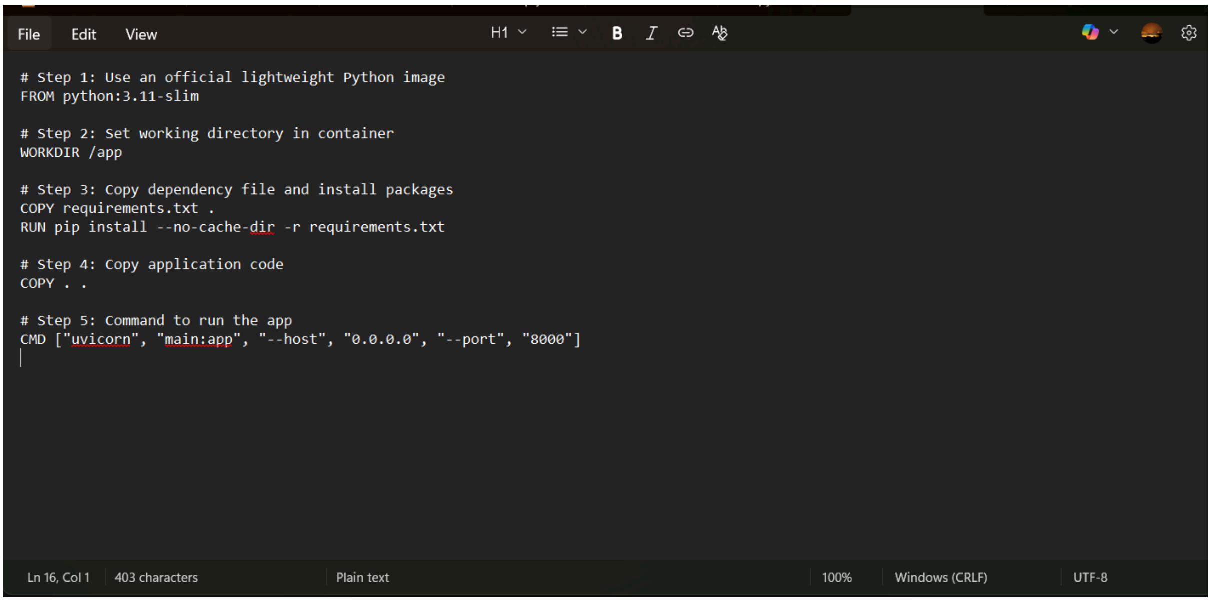
Your running containers show up here

A container is an isolated environment for your code

Engine running

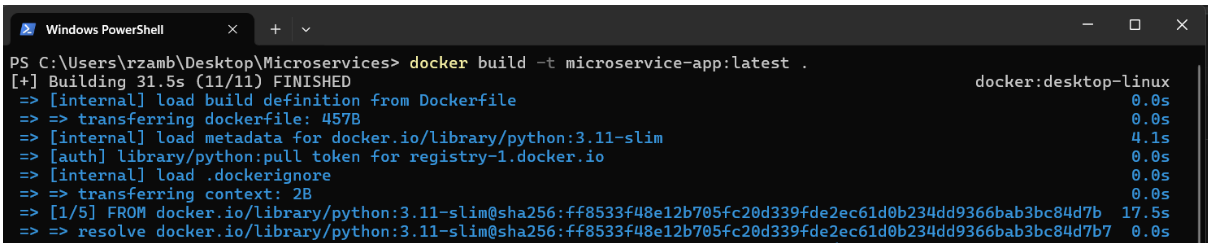
RAM 3.13 GB CPU 0.06% Disk: 1.24 GB used (limit 1006.85 GB)

Step 11: Create Dockerfile



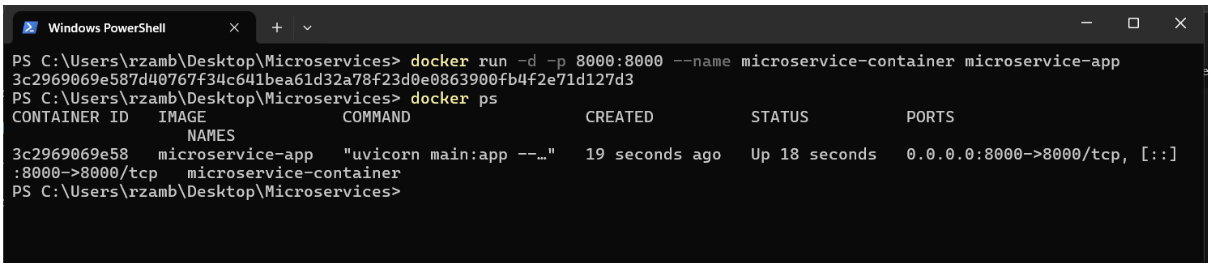
```
# Step 1: Use an official lightweight Python image
FROM python:3.11-slim

# Step 2: Set working directory in container
WORKDIR /app

# Step 3: Copy dependency file and install packages
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Step 4: Copy application code
COPY . .

# Step 5: Command to run the app
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Step 12: Convert Dockerfile to Docker image



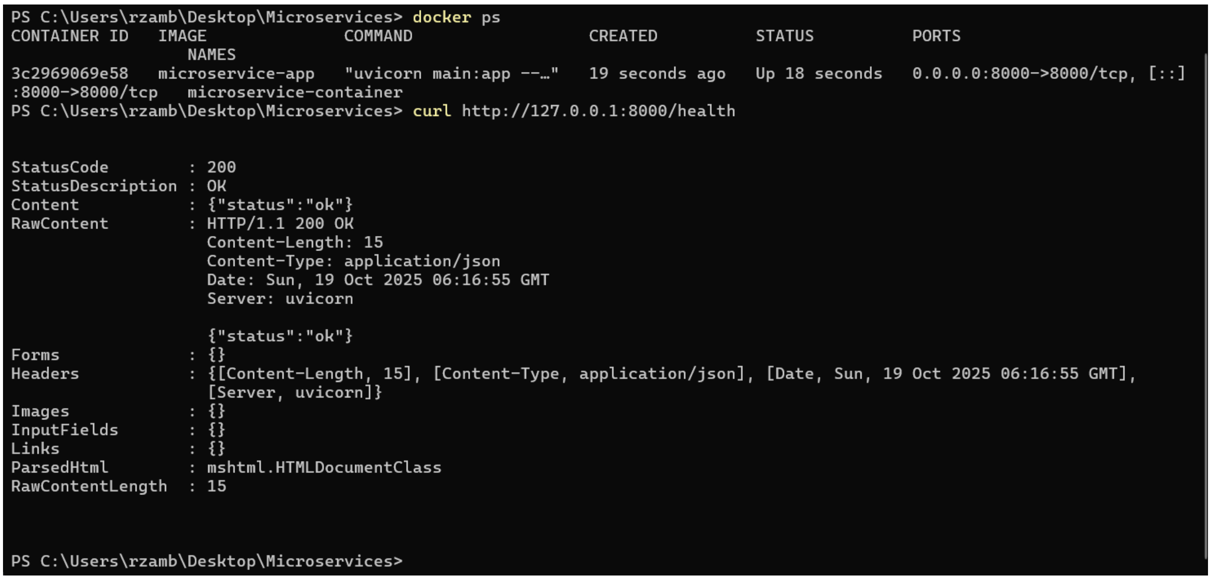
```
PS C:\Users\rzamb\Desktop\Microservices> docker build -t microservice-app:latest .
[+] Building 31.5s (11/11) FINISHED                                                     docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 457B                                                                    0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                                     4.1s
 => [auth] library/python:pull token for registry-1.docker.io                                           0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [1/5] FROM docker.io/library/python:3.11-slim@sha256:ff8533f48e12b705fc20d339fde2ec61d0b234dd9366bab3bc84d7b  17.5s
 => => resolve docker.io/library/python:3.11-slim@sha256:ff8533f48e12b705fc20d339fde2ec61d0b234dd9366bab3bc84d7b7  0.0s
```

Step 13: Run docker image on local:host :: 8000:8000 ports (trying to run docker application locally)

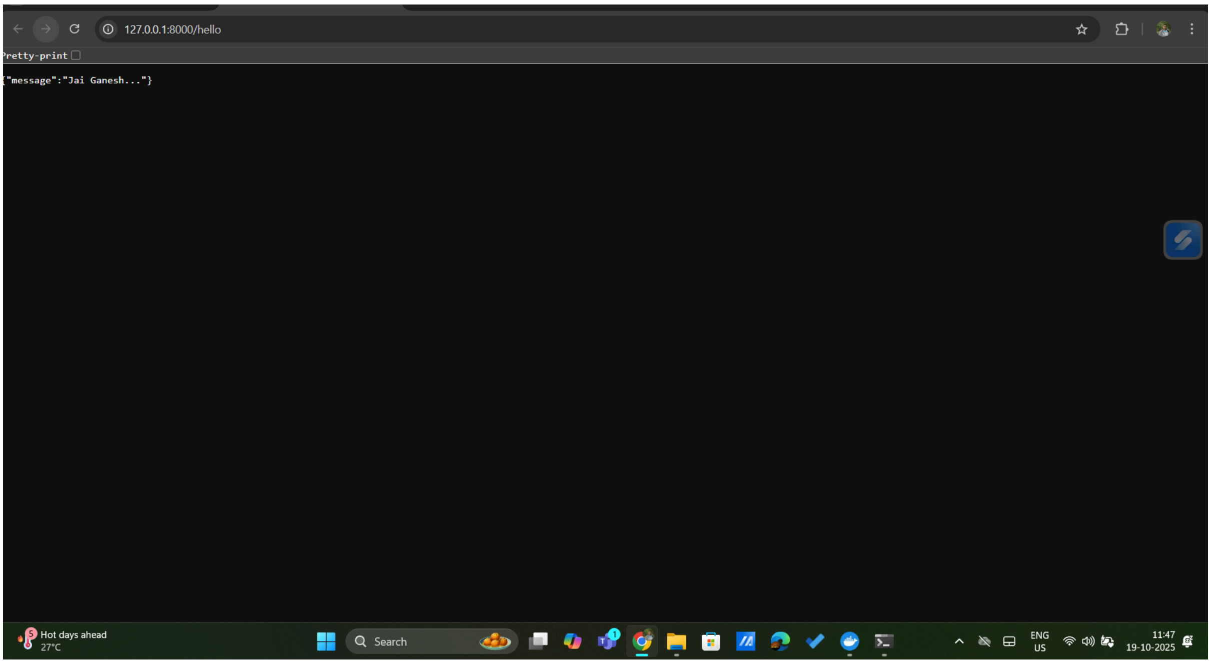


```
PS C:\Users\rzamb\Desktop\Microservices> docker run -d -p 8000:8000 --name microservice-container microservice-app
3c2969069e587d40767f34c641bea61d32a78f23d0e0863900fb4f2e71d127d3
PS C:\Users\rzamb\Desktop\Microservices> docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS
                 NAMES
3c2969069e58   microservice-app   "uvicorn main:app --…"   19 seconds ago   Up 18 seconds   0.0.0.0:8000->8000/tcp, [::]
:8000->8000/tcp   microservice-container
PS C:\Users\rzamb\Desktop\Microservices>
```

Step 14: Testing if Microservice running on local docker

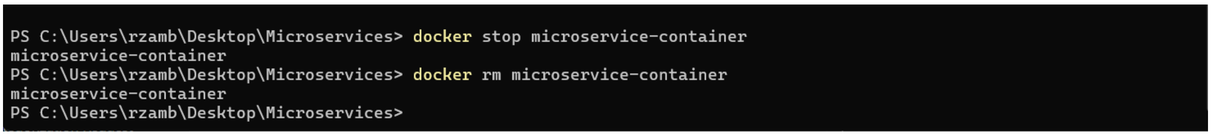


```
PS C:\Users\rzamb\Desktop\Microservices> docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS
                 NAMES
3c2969069e58   microservice-app   "uvicorn main:app --…"   19 seconds ago   Up 18 seconds   0.0.0.0:8000->8000/tcp, [::]
:8000->8000/tcp   microservice-container
PS C:\Users\rzamb\Desktop\Microservices> curl http://127.0.0.1:8000/health


StatusCode        : 200
StatusDescription : OK
Content           : {"status":"ok"}
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 15
                    Content-Type: application/json
                    Date: Sun, 19 Oct 2025 06:16:55 GMT
                    Server: uvicorn

                    {"status":"ok"}
Forms             : {}
Headers           : {[Content-Length, 15], [Content-Type, application/json], [Date, Sun, 19 Oct 2025 06:16:55 GMT],
                    [Server, uvicorn]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 15



PS C:\Users\rzamb\Desktop\Microservices>
```



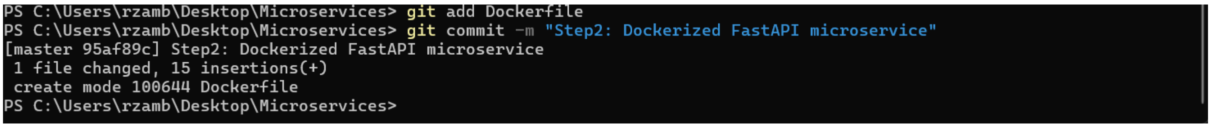
```
{"message":"Jai Ganesh..."}
```

Step 15: Stop and clear Docker (local)



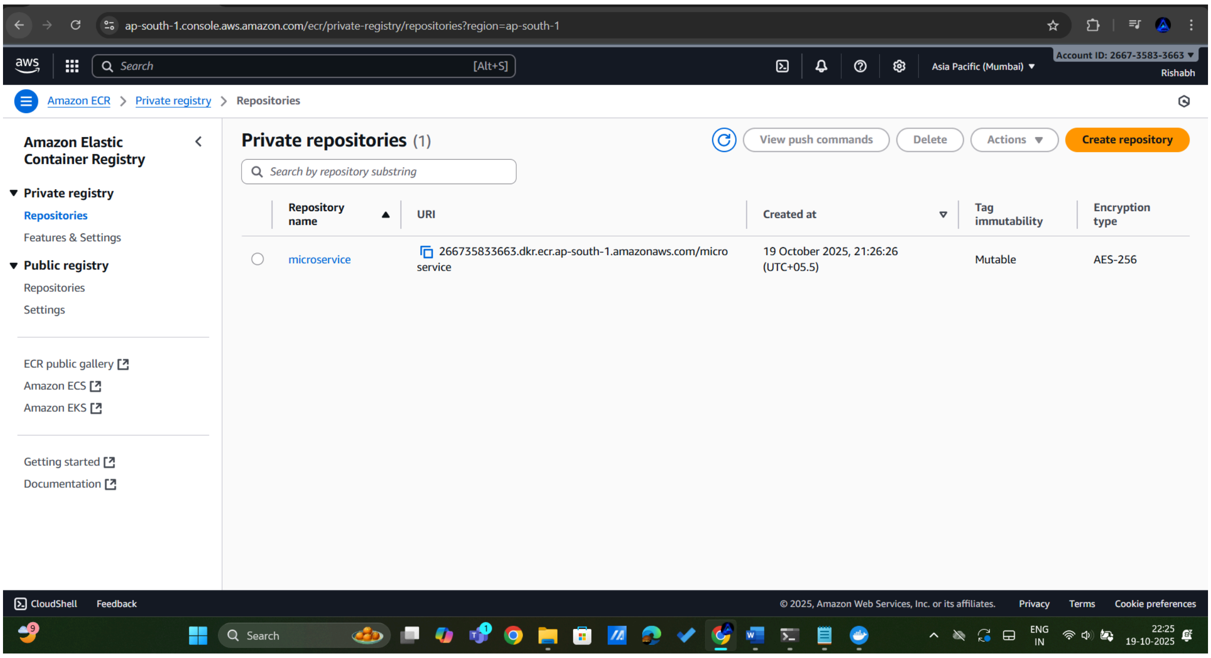
```
PS C:\Users\rzamb\Desktop\Microservices> docker stop microservice-container
microservice-container
PS C:\Users\rzamb\Desktop\Microservices> docker rm microservice-container
microservice-container
PS C:\Users\rzamb\Desktop\Microservices>
```

Step 16: Update git



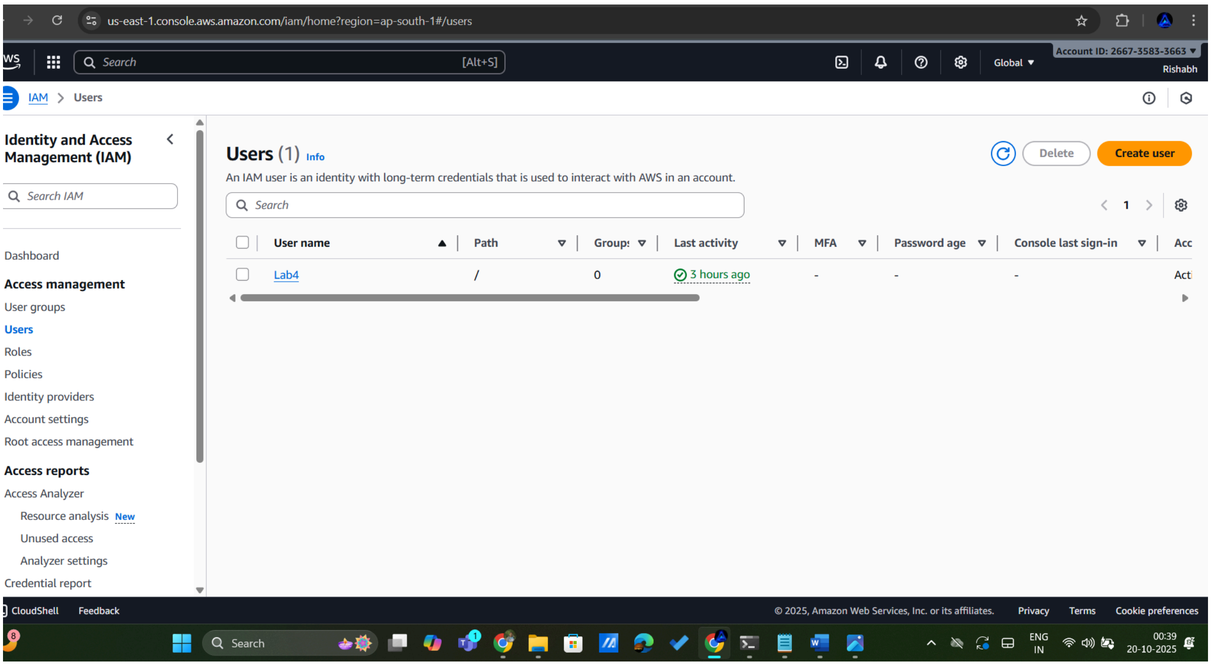
```
PS C:\Users\rzamb\Desktop\Microservices> git add Dockerfile
PS C:\Users\rzamb\Desktop\Microservices> git commit -m "Step2: Dockerized FastAPI microservice"
[master 95af89c] Step2: Dockerized FastAPI microservice
 1 file changed, 15 insertions(+)
 create mode 100644 Dockerfile
PS C:\Users\rzamb\Desktop\Microservices>
```

Step 17: Build private repository on ECR and upload docker image via IAM user



Private repositories (1)

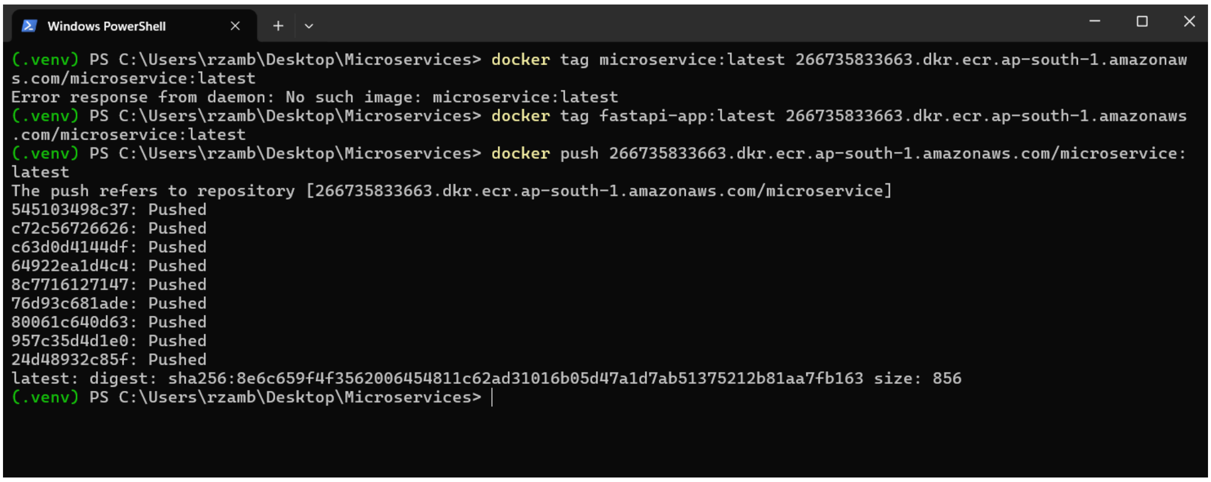
	Repository name	URI	Created at	Tag immutability	Encryption type
○	microservice	266735833663.dkr.ecr.ap-south-1.amazonaws.com/microservice	19 October 2025, 21:26:26 (UTC+05.5)	Mutable	AES-256



Users (1) Info

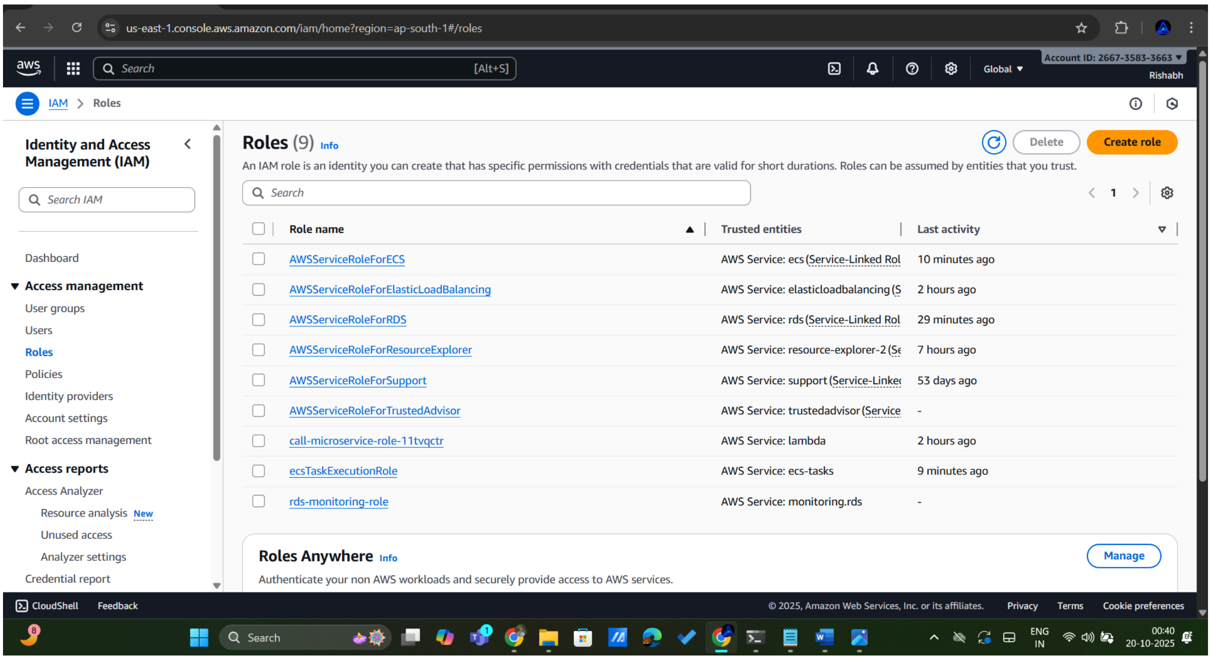
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

□	User name	Path	Group:	Last activity	MFA	Password age	Console last sign-in	Acc
□	Lab4	/	0	✓ 3 hours ago	-	-	-	Act



```
(.venv) PS C:\Users\rzamb\Desktop\Microservices> docker tag microservice:latest 266735833663.dkr.ecr.ap-south-1.amazonaw
s.com/microservice:latest
Error response from daemon: No such image: microservice:latest
(.venv) PS C:\Users\rzamb\Desktop\Microservices> docker tag fastapi-app:latest 266735833663.dkr.ecr.ap-south-1.amazonaws
.com/microservice:latest
(.venv) PS C:\Users\rzamb\Desktop\Microservices> docker push 266735833663.dkr.ecr.ap-south-1.amazonaws.com/microservice:
latest
The push refers to repository [266735833663.dkr.ecr.ap-south-1.amazonaws.com/microservice]
545103498c37: Pushed
c72c56726626: Pushed
c63d0d4144df: Pushed
64922ea1d4c4: Pushed
8c7716127147: Pushed
76d93c681ade: Pushed
80061c640d63: Pushed
957c35d4d1e0: Pushed
24d48932c85f: Pushed
latest: digest: sha256:8e6c659f4f3562006454811c62ad31016b05d47a1d7ab51375212b81aa7fb163 size: 856
(.venv) PS C:\Users\rzamb\Desktop\Microservices>
```

Step 18: Define roles for IAM user



us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/roles

IAM > Roles

Roles (9) Info

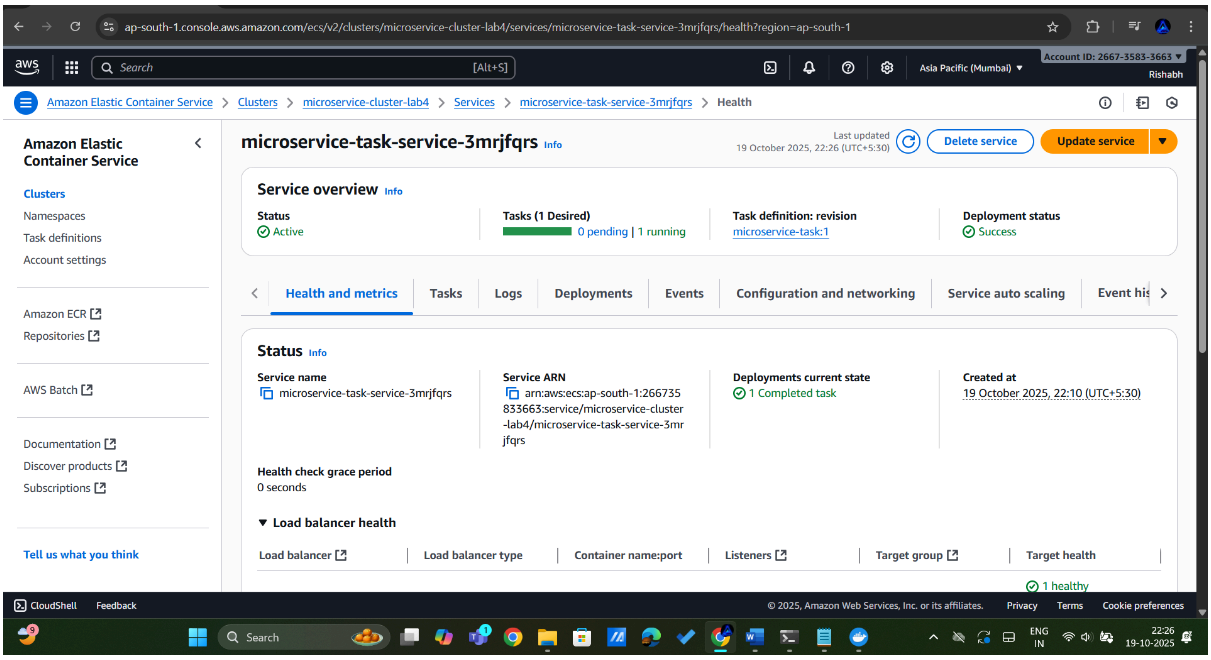
An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSServiceRoleForECS	AWS Service: ecs (Service-Linked Rol	10 minutes ago
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (S	2 hours ago
AWSServiceRoleForRDS	AWS Service: rds (Service-Linked Rol	29 minutes ago
AWSServiceRoleForResourceExplorer	AWS Service: resource-explorer-2 (Se	7 hours ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linke	53 days ago
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service	-
call-microservice-role-11tvqctr	AWS Service: lambda	2 hours ago
ecsTaskExecutionRole	AWS Service: ecs-tasks	9 minutes ago
rds-monitoring-role	AWS Service: monitoring.rds	-

Roles Anywhere Info

Authenticate your non AWS workloads and securely provide access to AWS services.

Step 19: Build ECS Cluster and health check for communication



ap-south-1.console.aws.amazon.com/ecs/v2/clusters/microservice-cluster-lab4/services/microservice-task-service-3mrjfqrs/health?region=ap-south-1

Amazon Elastic Container Service > Clusters > microservice-cluster-lab4 > Services > microservice-task-service-3mrjfqrs > Health

microservice-task-service-3mrjfqrs Info

Last updated 19 October 2025, 22:26 (UTC+5:30)

Service overview

Status	Tasks (1 Desired)	Task definition: revision	Deployment status
Active	0 pending | 1 running	microservice-task:1	Success

Status

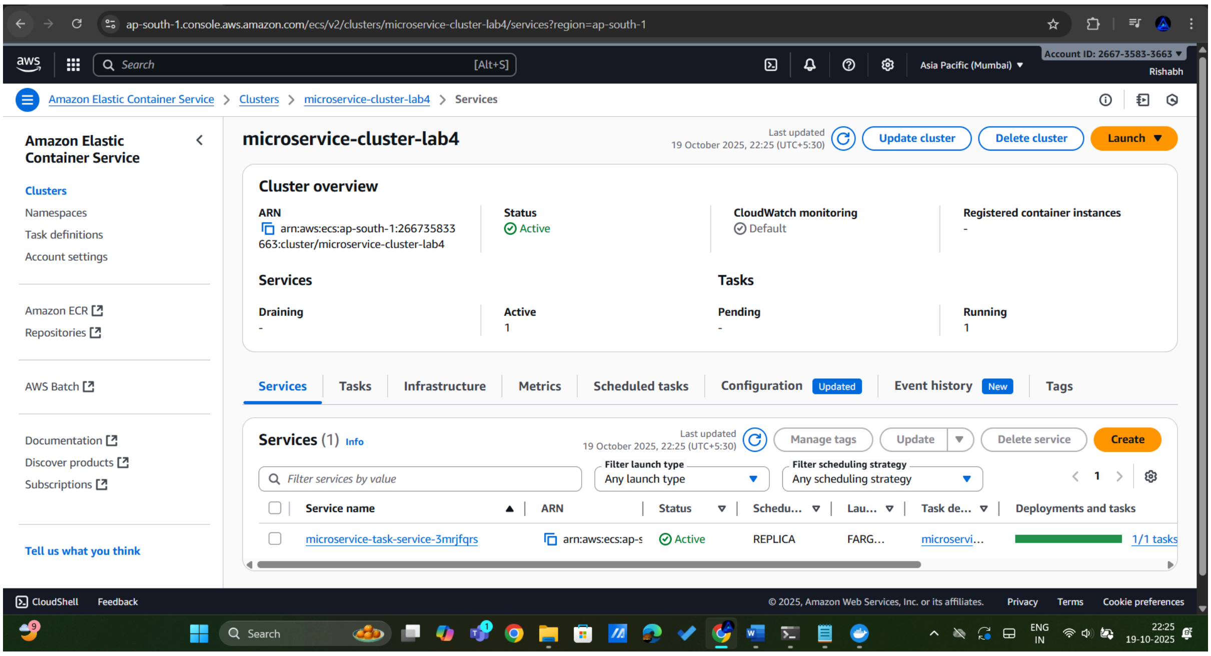
Service name	Service ARN	Deployments current state	Created at
microservice-task-service-3mrjfqrs	arn:aws:ecs:ap-south-1:266735833663:service/microservice-cluster-lab4/microservice-task-service-3mrjfqrs	1 Completed task	19 October 2025, 22:10 (UTC+5:30)

Health check grace period
0 seconds

Load balancer health

Load balancer	Load balancer type	Container name:port	Listeners	Target group	Target health
					1 healthy

Step 20: Define ECS service



microservice-cluster-lab4

Last updated 19 October 2025, 22:25 (UTC+5:30)

Cluster overview

ARN: arn:aws:ecs:ap-south-1:266735833663:cluster/microservice-cluster-lab4

Status: Active

CloudWatch monitoring: Default

Registered container instances: -

Services

Draining: -

Active: 1

Tasks

Pending: -

Running: 1

Services (1) Info

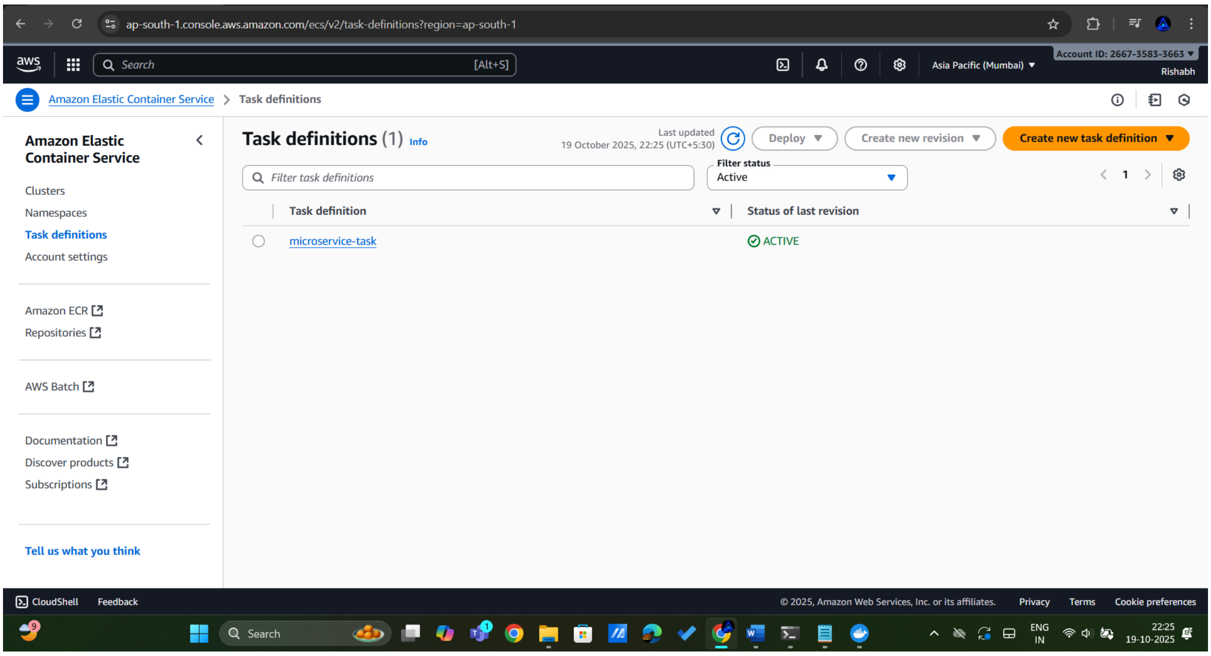
Last updated 19 October 2025, 22:25 (UTC+5:30)

Filter launch type: Any launch type

Filter scheduling strategy: Any scheduling strategy

Service name	ARN	Status	Schedu...	Lau...	Task de...	Deployments and tasks
microservice-task-service-3mrjfqrs	arn:aws:ecs:ap-s	Active	REPLICA	FARG...	microservi...	1/1 tasks

Step 21: Define ECS task



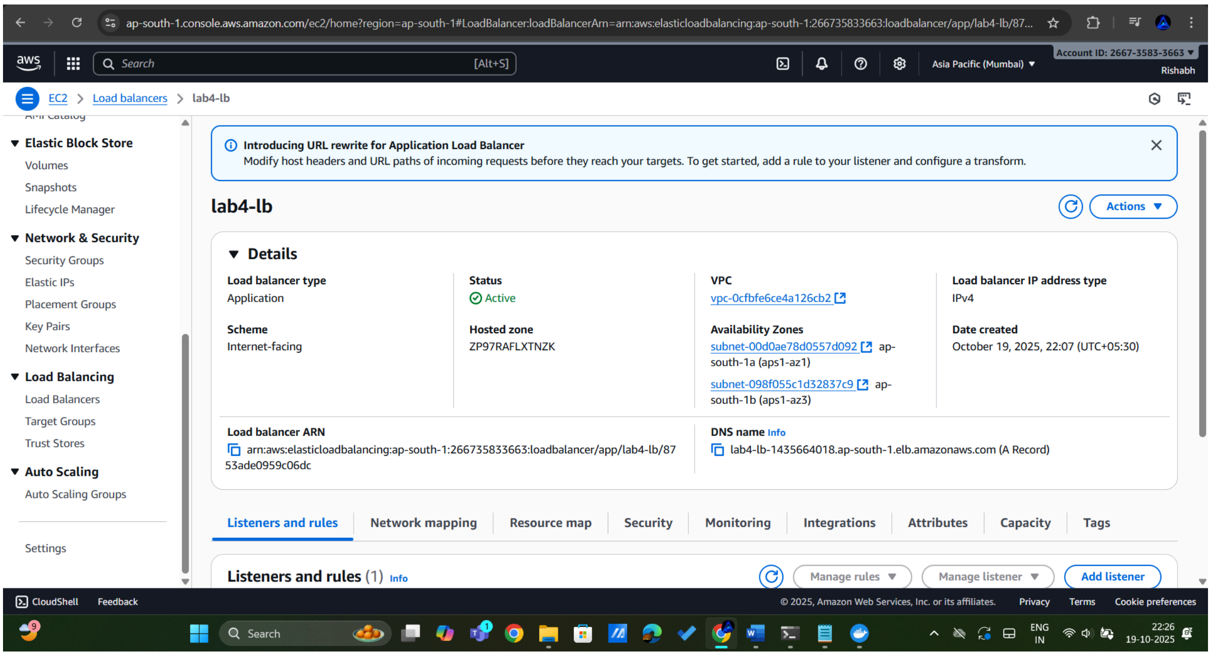
Task definitions (1) Info

Last updated 19 October 2025, 22:25 (UTC+5:30)

Filter status: Active

Task definition	Status of last revision
microservice-task	ACTIVE

Step 22: Create an ALB



ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LoadBalancer:loadBalancerArn=arn:aws:elasticloadbalancing:ap-south-1:266735833663:loadbalancer/app/lab4-lb/87...

EC2 > Load balancers > lab4-lb

Introducing URL rewrite for Application Load Balancer
Modify host headers and URL paths of incoming requests before they reach your targets. To get started, add a rule to your listener and configure a transform.

lab4-lb

Details

Load balancer type: Application

Status: Active

VPC: vpc-0cfbfe6ce4a126cb2

Load balancer IP address type: IPv4

Scheme: Internet-facing

Hosted zone: ZP97RAFLXTNZK

Availability Zones: subnet-00d0ae78d0557d092 ap-south-1a (aps1-az1); subnet-098f055c1d32837c9 ap-south-1b (aps1-az3)

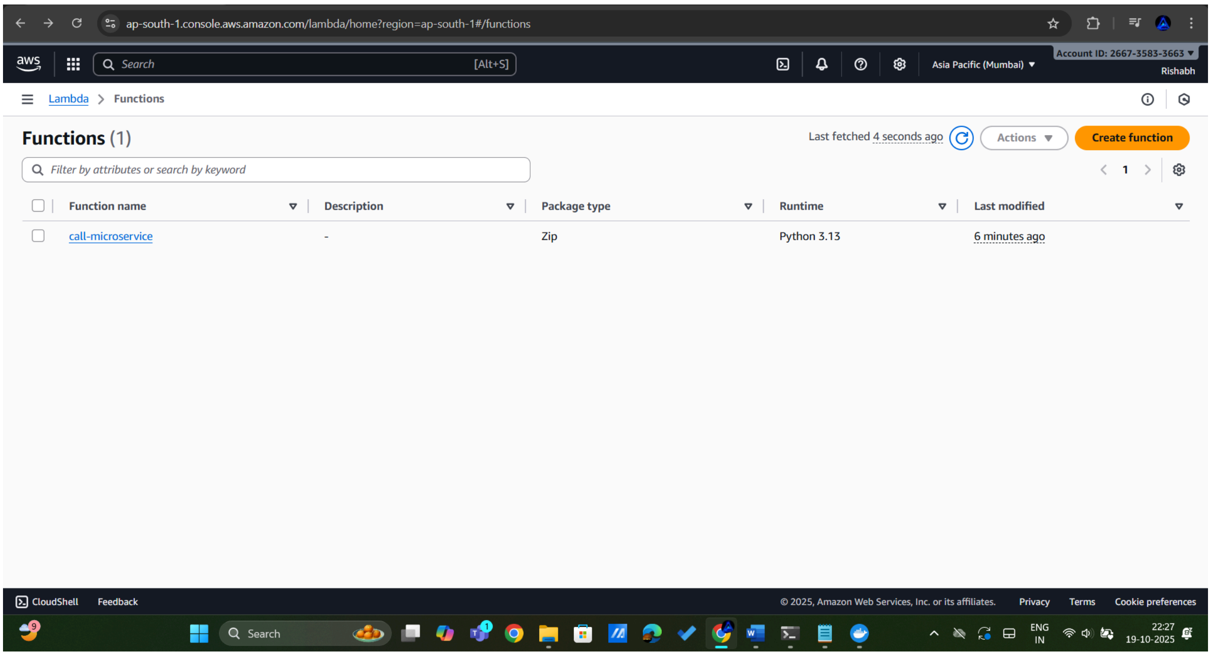
Date created: October 19, 2025, 22:07 (UTC+05:30)

Load balancer ARN: arn:aws:elasticloadbalancing:ap-south-1:266735833663:loadbalancer/app/lab4-lb/8753ade0959c06dc

DNS name: lab4-lb-1435664018.ap-south-1.elb.amazonaws.com (A Record)

Listeners and rules (1)

Step 23: Create lambda function



ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions

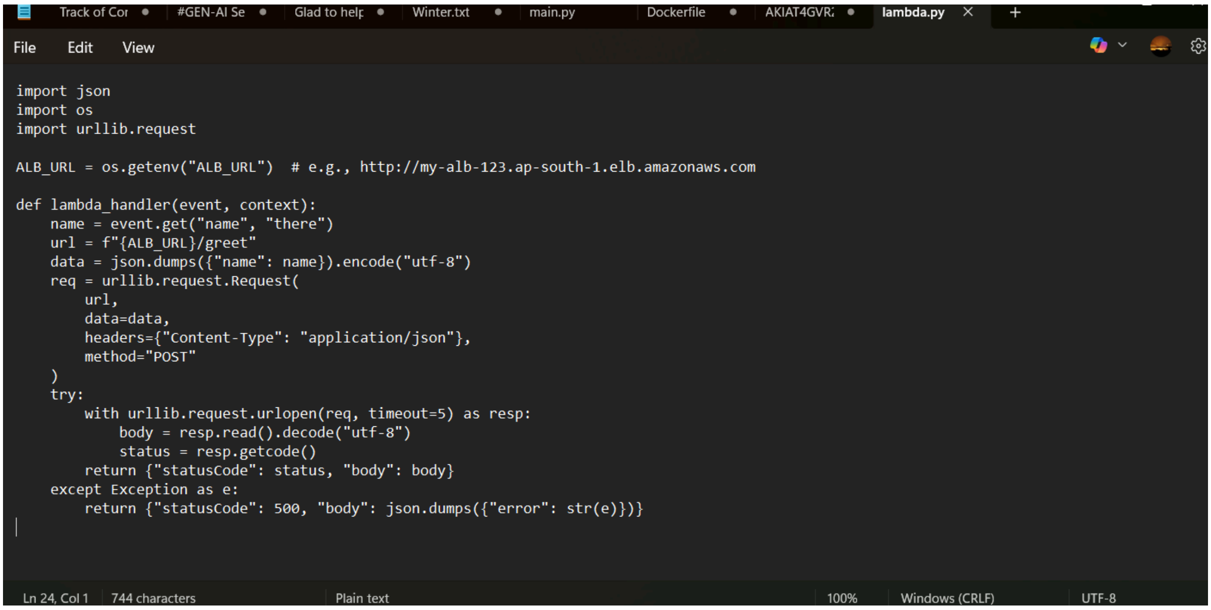
Lambda > Functions

Functions (1)

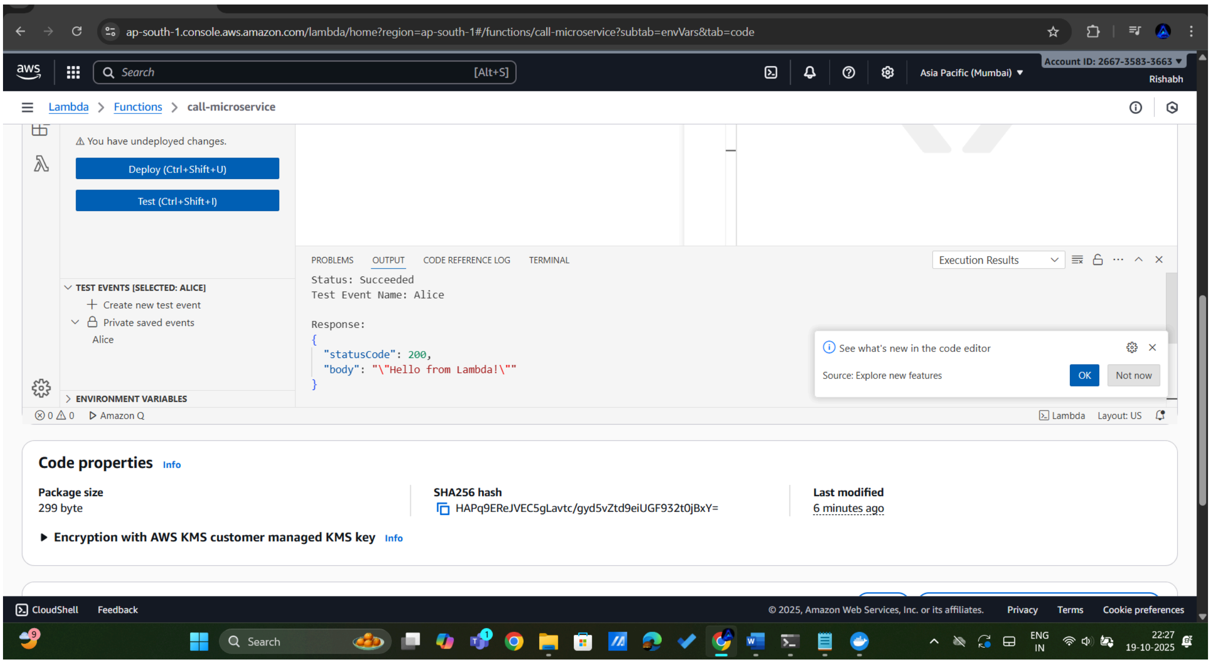
Last fetched 4 seconds ago

Function name	Description	Package type	Runtime	Last modified
call-microservice	-	Zip	Python 3.13	6 minutes ago

Step 24: Test using Lambda function and code



```
import json
import os
import urllib.request

ALB_URL = os.getenv("ALB_URL")  # e.g., http://my-alb-123.ap-south-1.elb.amazonaws.com

def lambda_handler(event, context):
    name = event.get("name", "there")
    url = f"{ALB_URL}/greet"
    data = json.dumps({"name": name}).encode("utf-8")
    req = urllib.request.Request(
        url,
        data=data,
        headers={"Content-Type": "application/json"},
        method="POST"
    )
    try:
        with urllib.request.urlopen(req, timeout=5) as resp:
            body = resp.read().decode("utf-8")
            status = resp.getcode()
        return {"statusCode": status, "body": body}
    except Exception as e:
        return {"statusCode": 500, "body": json.dumps({"error": str(e)})}
```



```
Status: Succeeded
Test Event Name: Alice

Response:
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}
```

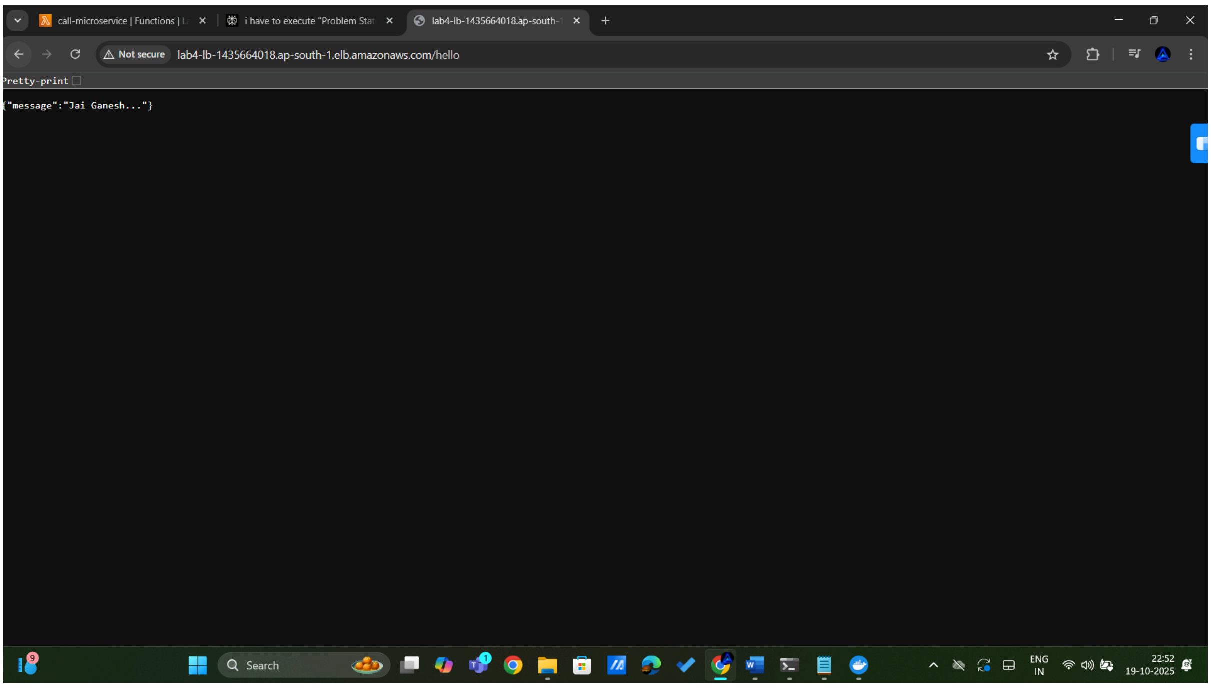
Code properties Info

Package size
299 byte

SHA256 hash
HAPq9EReJVEC5gLavtc/gyd5vZtd9eiUGF932t0jBxY=

Last modified
6 minutes ago

▶ Encryption with AWS KMS customer managed KMS key Info



call-microservice | Functions | L
i have to execute "Problem Stat
lab4-lb-1435664018.ap-south-
Not secure
lab4-lb-1435664018.ap-south-1.elb.amazonaws.com/hello
Pretty-print □

```
{"message":"Jai Ganesh..."}
```

Search
ENG
IN
22:52
19-10-2025