3-Tier Cloud Gaming Deployment Configuration Manual

Edge-Fog-Cloud • LSTM (Edge) • Q-learning (Cloud) • Flask APIs
Local & AWS EC2 • Amazon Linux 2023 • Windows/WSL friendly

1. Introduction

This manual shows how to deploy and verify a three-tier Cloud Gaming system. Edge runs a lightweight LSTM to predict imminent combat; Fog aggregates and caches; Cloud uses a trained Q-learning policy to pick a network slice (Basic/Medium/Premium).

Repo: github.com/muzakkir908/MSc_RIC

2. Quick Start

Local (single machine)

Bash (Linux/macOS/WSL):

```
git clone https://github.com/muzakkir908/MSc_RIC.git
cd MSc_RIC
python3 -m pip install --upgrade pip setuptools
python3 -m pip install -r requirements.txt
python3 edge_server.py   # :5000
python3 correct_cloud_server.py   # :5001
python3 fog_cache_server.py   # :5002
```

PowerShell (Windows):

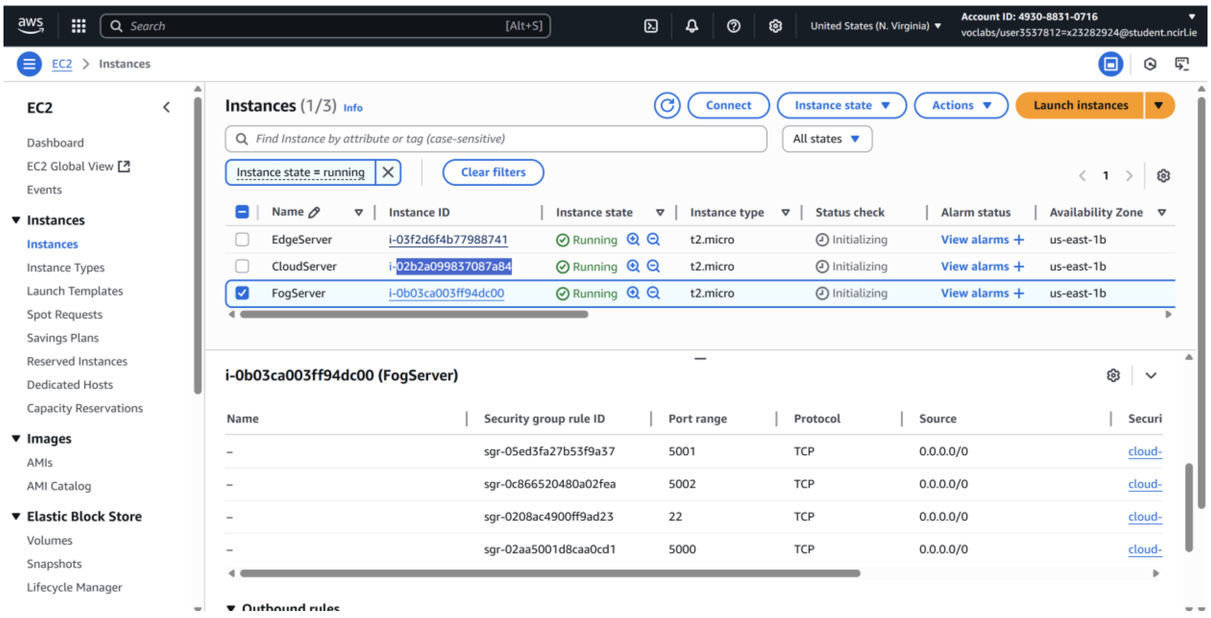
```
git clone https://github.com/muzakkir908/MSc_RIC.git
cd MSc_RIC
py -m pip install --upgrade pip setuptools
py -m pip install -r requirements.txt
py edge_server.py   # :5000
py correct_cloud_server.py   # :5001
py fog_cache_server.py   # :5002
```

AWS (three EC2 instances)

On each instance (Amazon Linux 2023):

```
sudo dnf update -y
sudo dnf install -y git python3-pip
git clone https://github.com/muzakkir908/MSc_RIC.git
cd MSc_RIC
pip3 install -r requirements.txt
```

Open inbound ports in the Security Group: 22 (SSH), 5000 (Edge), 5001 (Cloud), 5002 (Fog). Prefer private IPs within the VPC for service-to-service traffic.



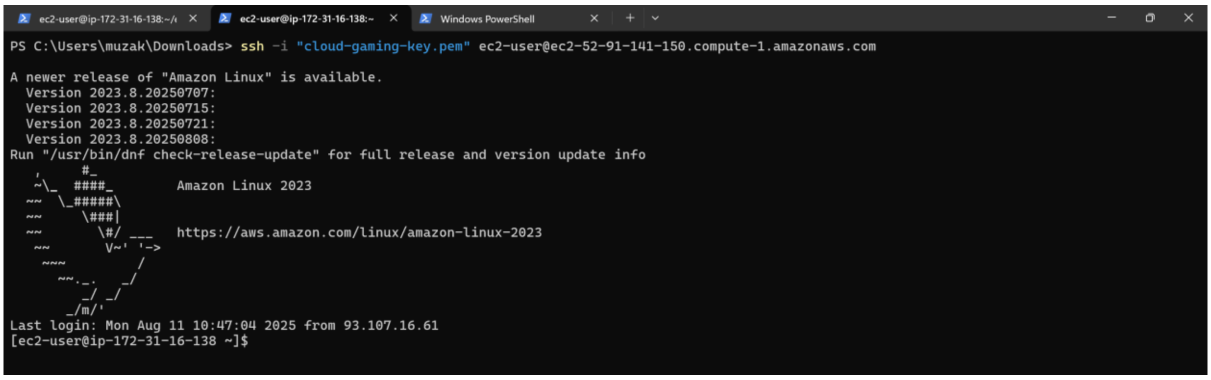
Instances (1/3) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
EdgeServer	i-03f2d6f4b77988741	Running	t2.micro	Initializing	View alarms +	us-east-1b
CloudServer	i-02b2a099837087a84	Running	t2.micro	Initializing	View alarms +	us-east-1b
FogServer	i-0b03ca003ff94dc00	Running	t2.micro	Initializing	View alarms +	us-east-1b

i-0b03ca003ff94dc00 (FogServer)

Name	Security group rule ID	Port range	Protocol	Source	Securi
–	sgr-05ed3fa27b53f9a37	5001	TCP	0.0.0.0/0	cloud-
–	sgr-0c866520480a02fea	5002	TCP	0.0.0.0/0	cloud-
–	sgr-0208ac4900ff9ad23	22	TCP	0.0.0.0/0	cloud-
–	sgr-02aa5001d8caa0cd1	5000	TCP	0.0.0.0/0	cloud-

Figure 1. EC2 instances and Security Group inbound rules (ports 22, 5000, 5001, 5002) for Edge, Cloud, and Fog.



```
PS C:\Users\muzak\Downloads> ssh -i "cloud-gaming-key.pem" ec2-user@ec2-52-91-141-150.compute-1.amazonaws.com

A newer release of "Amazon Linux" is available.
  Version 2023.8.20250707:
  Version 2023.8.20250715:
  Version 2023.8.20250721:
  Version 2023.8.20250808:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Mon Aug 11 10:47:04 2025 from 93.107.16.61
[ec2-user@ip-172-31-16-138 ~]$
```

Figure 2. PowerShell SSH connection to Amazon Linux 2023 (initial host key fingerprint confirmation).

3. Prerequisites

• AWS account (for cloud runs) • SSH key (.pem) • Python 3.9+ • Git • curl • (optional) tmux/screen

• Deep learning stack: PyTorch 2.x (preferred) or TensorFlow 2.x

• Amazon Linux 2023 on EC2; Windows/macOS/Linux locally

Network & Ports

Port	**Protocol**	**Source (SG)**	**Used by**
5000	TCP	Your IP / VPC CIDR	Edge /predict, /health
5001	TCP	Your IP / VPC CIDR	Cloud /decide, /health
5002	TCP	Your IP / VPC CIDR	Fog /process, /stats, /health
22	TCP	Your IP only	SSH management

4. Architecture Overview

Edge: LSTM predicts combat probability in real time (target sub-5 ms).
Fog: Aggregates N=10 recent probabilities; LRU cache with TTL (e.g., 5 minutes) to avoid repeat Cloud calls.
Cloud: Q-learning agent selects slice 0/1/2 based on state and predicted intensity.

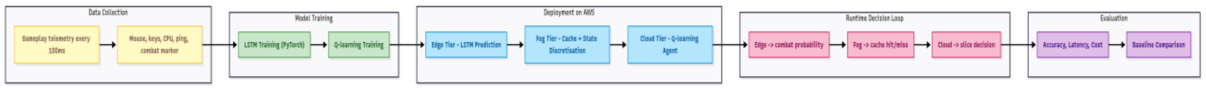


Figure 3. End-to-end workflow: Data Collection → Model Training (LSTM, Q-learning) → AWS Deployment (Edge, Fog, Cloud) → Runtime Decision Loop → Evaluation (accuracy, latency, cost, baseline comparison).

5. Configuration

5.1 Environment variables (.env)

```
# .env example (place in project root)
EDGE_URL=http://127.0.0.1:5000
FOG_URL=http://127.0.0.1:5002
CLOUD_URL=http://127.0.0.1:5001
CACHE_TTL_SECONDS=300
CACHE_SIZE=1000
```

If your scripts read from environment variables, export before running.

Bash:

```
export EDGE_URL=http://<EDGE_PRIVATE_IP>:5000
export CLOUD_URL=http://<CLOUD_PRIVATE_IP>:5001
export FOG_URL=http://<FOG_PRIVATE_IP>:5002
```

PowerShell:

```
$env:EDGE_URL="http://<EDGE_PRIVATE_IP>:5000"
$env:CLOUD_URL="http://<CLOUD_PRIVATE_IP>:5001"
$env:FOG_URL="http://<FOG_PRIVATE_IP>:5002"
```

5.2 Model files

Edge: place trained_lstm_model.pth (or .h5) in ./models if required.
Cloud: ensure ./models/trained_q_learning_model.pkl exists.

5.3 Bind addresses

Ensure Flask apps run with host='0.0.0.0' for remote access, e.g.:

```
app.run(host='0.0.0.0', port=5000)
```

6. Running the Services

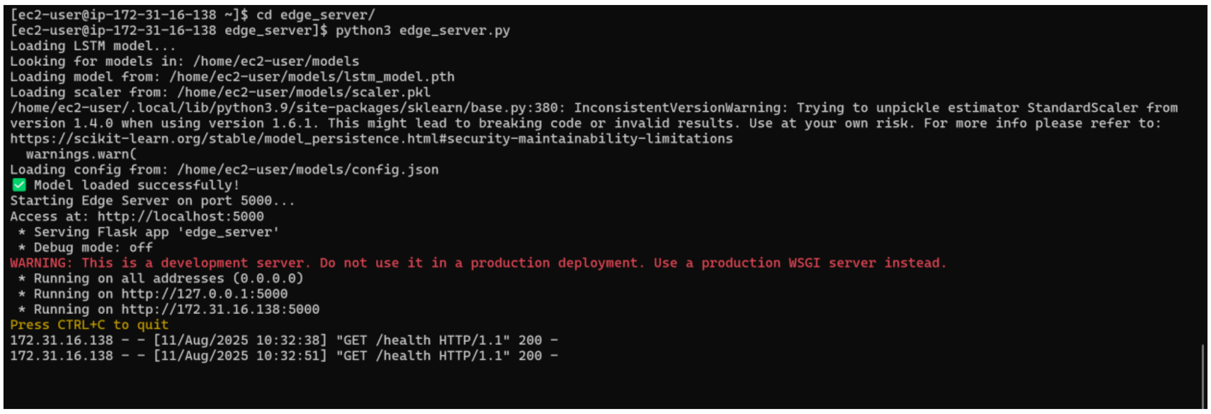
6.1 Foreground (simple)

Bash:

```
python3 edge_server.py
python3 correct_cloud_server.py
python3 fog_cache_server.py
```

PowerShell:

```
py edge_server.py
py correct_cloud_server.py
py fog_cache_server.py
```



```
[ec2-user@ip-172-31-16-138 ~]$ cd edge_server/
[ec2-user@ip-172-31-16-138 edge_server]$ python3 edge_server.py
Loading LSTM model...
Looking for models in: /home/ec2-user/models
Loading model from: /home/ec2-user/models/lstm_model.pth
Loading scaler from: /home/ec2-user/models/scaler.pkl
/home/ec2-user/.local/lib/python3.9/site-packages/sklearn/base.py:380: InconsistentVersionWarning: Trying to unpickle estimator StandardScaler from
version 1.4.0 when using version 1.6.1. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
Loading config from: /home/ec2-user/models/config.json
✅ Model loaded successfully!
Starting Edge Server on port 5000...
Access at: http://localhost:5000
 * Serving Flask app 'edge_server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.16.138:5000
Press CTRL+C to quit
172.31.16.138 - - [11/Aug/2025 10:32:38] "GET /health HTTP/1.1" 200 -
172.31.16.138 - - [11/Aug/2025 10:32:51] "GET /health HTTP/1.1" 200 -
```

Figure 4. Edge Flask server running on port 5000 (development server notice visible; binds on 0.0.0.0).

6.2 Background with tmux (EC2)

```
tmux new -s edge -d 'python3 edge_server.py'
tmux new -s cloud -d 'python3 correct_cloud_server.py'
tmux new -s fog -d 'python3 fog_cache_server.py'
tmux ls
```

6.3 As systemd services (auto-start on boot)

Create /etc/systemd/system/edge.service:

```
[Unit]
Description=Edge Server
After=network.target

[Service]
User=ec2-user
WorkingDirectory=/home/ec2-user/MSc_RIC
ExecStart=/usr/bin/python3 edge_server.py
Restart=always

[Install]
WantedBy=multi-user.target
```

Enable & start:

```
sudo systemctl daemon-reload
sudo systemctl enable edge
sudo systemctl start edge
sudo systemctl status edge
```

Repeat for cloud.service (correct_cloud_server.py) and fog.service (fog_cache_server.py).

7. Verification & Tests

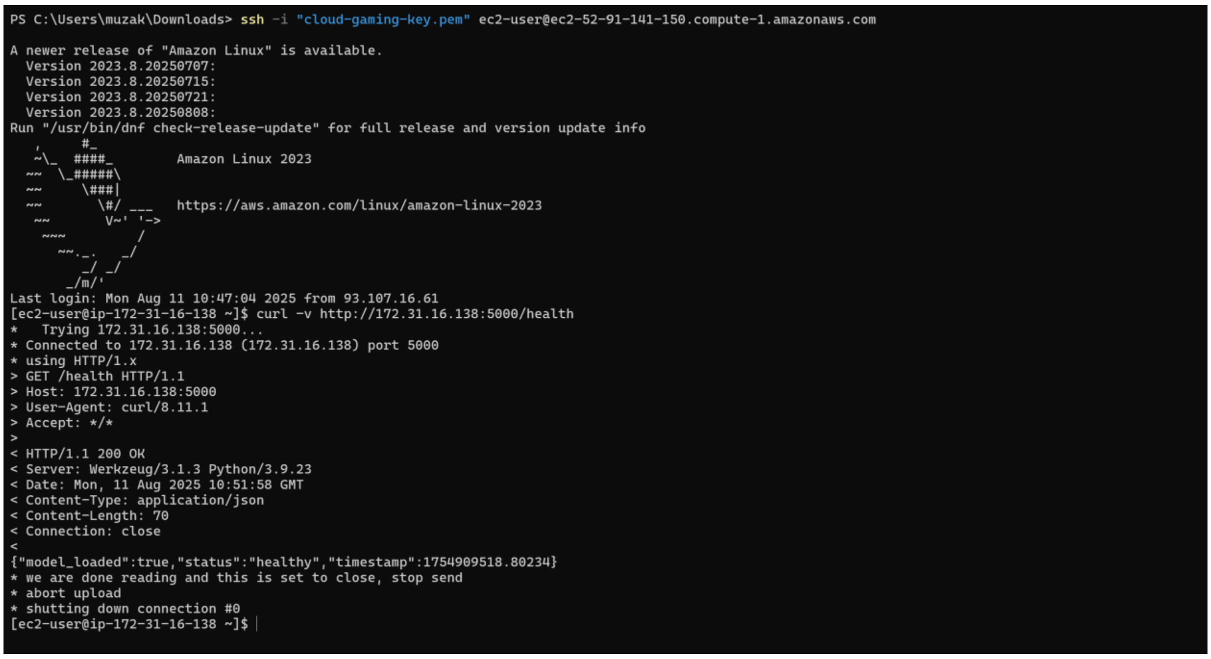
7.1 Health checks

Bash:

```
curl http://<EDGE_HOST>:5000/health
curl http://<CLOUD_HOST>:5001/health
curl http://<FOG_HOST>:5002/health
```

PowerShell note:

```
In PowerShell, "curl" is an alias for Invoke-WebRequest. Use either:
iwr http://<EDGE_HOST>:5000/health -UseBasicParsing
or explicitly call curl.exe:
cmd /c curl http://<EDGE_HOST>:5000/health
```



```
PS C:\Users\muzak\Downloads> ssh -i "cloud-gaming-key.pem" ec2-user@ec2-52-91-141-150.compute-1.amazonaws.com

A newer release of "Amazon Linux" is available.
  Version 2023.8.20250707:
  Version 2023.8.20250715:
  Version 2023.8.20250721:
  Version 2023.8.20250808:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Mon Aug 11 10:47:04 2025 from 93.107.16.61
[ec2-user@ip-172-31-16-138 ~]$ curl -v http://172.31.16.138:5000/health
*   Trying 172.31.16.138:5000...
* Connected to 172.31.16.138 (172.31.16.138) port 5000
* using HTTP/1.x
> GET /health HTTP/1.1
> Host: 172.31.16.138:5000
> User-Agent: curl/8.11.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Werkzeug/3.1.3 Python/3.9.23
< Date: Mon, 11 Aug 2025 10:51:58 GMT
< Content-Type: application/json
< Content-Length: 70
< Connection: close
<
{"model_loaded":true,"status":"healthy","timestamp":1754909518.80234}
* we are done reading and this is set to close, stop send
* abort upload
* shutting down connection #0
[ec2-user@ip-172-31-16-138 ~]$
```

Figure 5. Successful /health response (HTTP 200) confirming model_loaded=true and service status=healthy.

7.2 End-to-end tests

Update URLs in 05_tests/integration_tests/test_client.py & test_3tier.py, then run:

```
cd 05_tests/integration_tests
python3 test_client.py
python3 test_3tier.py
```

Outputs may include CSVs and a plot (3tier_performance.png).

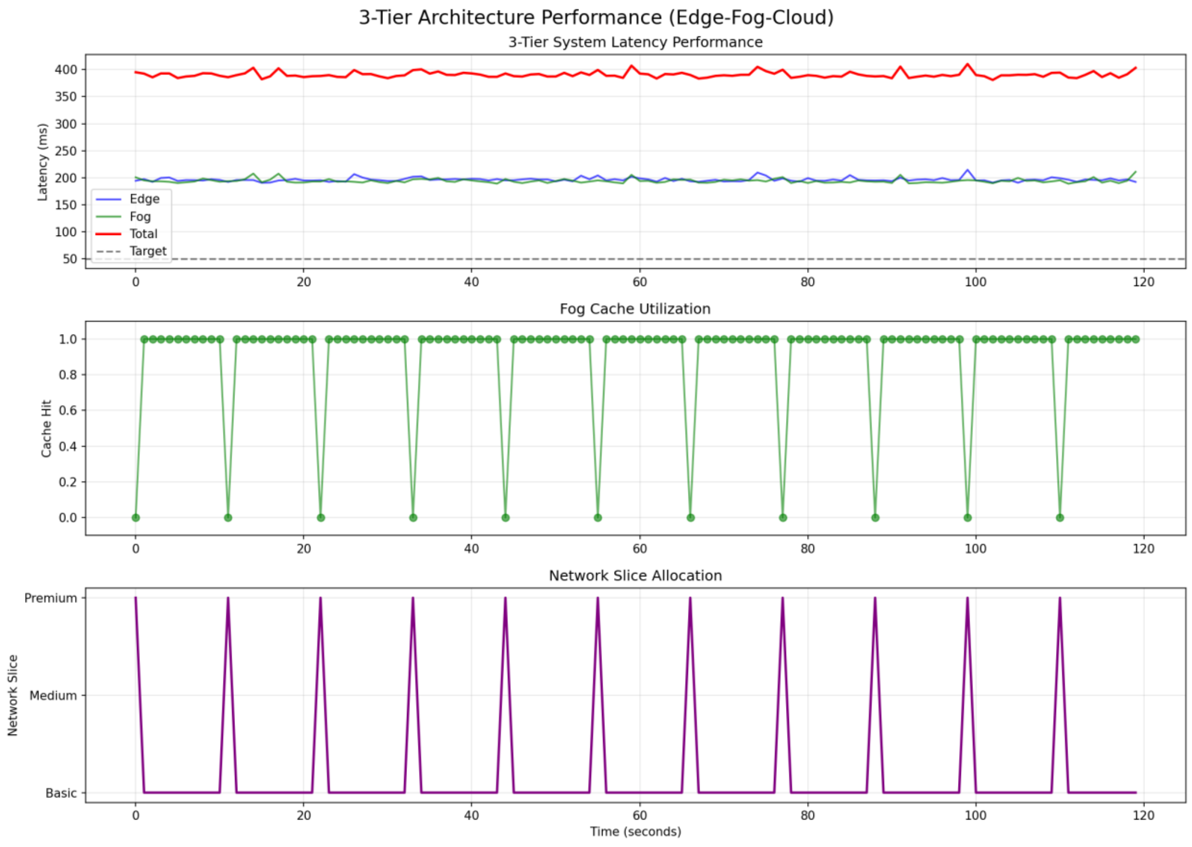
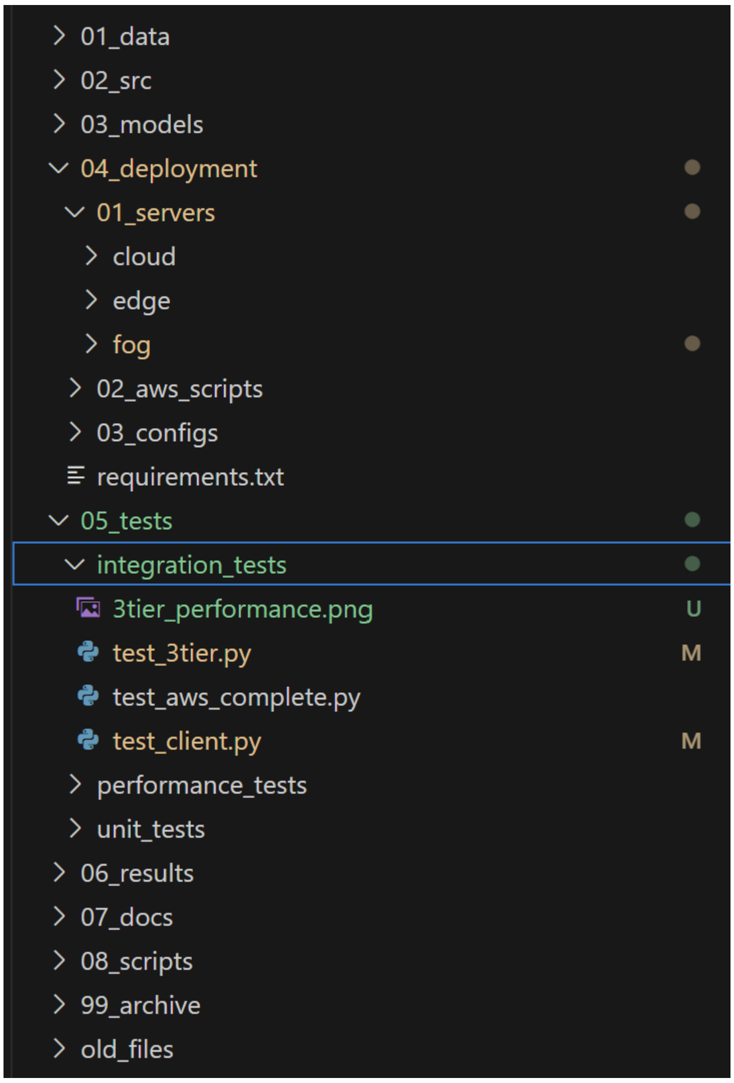


Figure 6. test_3tier.py results: latency, Fog cache hits, and network slice allocation (0/1/2).

8. Repository Map

01_edge/: edge_server.py
02_fog/: fog_cache_server.py
03_cloud/: correct_cloud_server.py
models/: trained_q_learning_model.pkl (+ LSTM)
05_tests/: integration & performance tests



```
> 01_data
> 02_src
> 03_models
v 04_deployment
  v 01_servers
    > cloud
    > edge
    > fog
  > 02_aws_scripts
  > 03_configs
  requirements.txt
v 05_tests
  v integration_tests
    3tier_performance.png          U
    test_3tier.py                  M
    test_aws_complete.py
    test_client.py                 M
  > performance_tests
  > unit_tests
> 06_results
> 07_docs
> 08_scripts
> 99_archive
> old_files
```

Figure 7. Project structure highlighting Edge, Fog, Cloud,models, and tests.

9. Security & Hardening (Concise)

• Restrict SG sources: open 5000–5002 only to your IP or VPC CIDR.
• Never commit secrets. If using .env, avoid pushing to Git.
• Use private IPs for Fog↔Edge/Cloud; keep public access minimal.
• Consider HTTPS (nginx reverse proxy + cert) for public endpoints.
• Keep instances patched (dnf update).

10. Troubleshooting

Connectivity (timeout/refused):

```
ss -ltnp | grep 500[0-2]
# If Flask binds to 127.0.0.1, change to 0.0.0.0
# Verify SG rules and correct IP/port
```

Missing models / ImportError:

```
ls -lah models/
pip3 install -r requirements.txt --upgrade
py -m pip install -r requirements.txt  # PowerShell
```

High latency:

Upgrade to t3.medium+; run tests from nearby region; prefer private IPs.

Fog cache not hitting:

Send identical requests within TTL; confirm cache size/TTL; check /stats.

11. Packaging for Submission (Optional)

7-Zip (Windows CMD/PowerShell):

```
"%ProgramFiles%\7-Zip\7z.exe" a -xr!venv "C:\path\to\final.zip"
"C:\path\to\MSc_RIC\*"
```

PowerShell native:

```
Compress-Archive -Path "C:\path\to\MSc_RIC\*" -DestinationPath
"C:\path\to\final.zip"
```