Deliverables (Students must submit)
Deliverable A — “Audit Evidence Pack” (one folder)

学生は audit-pack/ フォルダを提出。

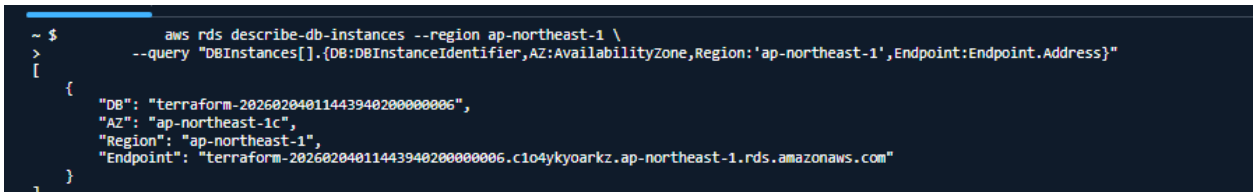
audit-pack/
├── 00_architecture-summary.md
├── 01_data-residency-proof.txt
├── 02_edge-proof-cloudfront.txt
├── 03_waf-proof.txt
├── 04_cloudtrail-change-proof.txt
├── 05_network-corridor-proof.txt
└── evidence.json (Malgus scripts output)

Deliverable B — One paragraph “auditor narrative”
“この設計が APPI 的に安全で、なぜ DB を海外に置けないか”を 8〜12 行で説明。

Verification Commands (CLI proof students can paste)
1) Data residency proof (RDS only in Tokyo)

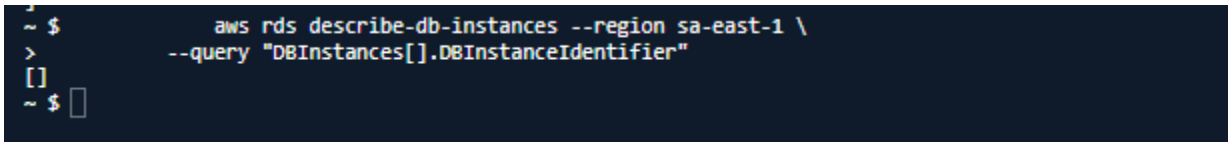
Tokyo: RDS exists

```
aws rds describe-db-instances --region ap-northeast-1 \
--query "DBInstances[].{DB:DBInstanceIdentifier,AZ:AvailabilityZone,Region:'ap-northeast-1',Endpoint:Endpoint.Address}"
```



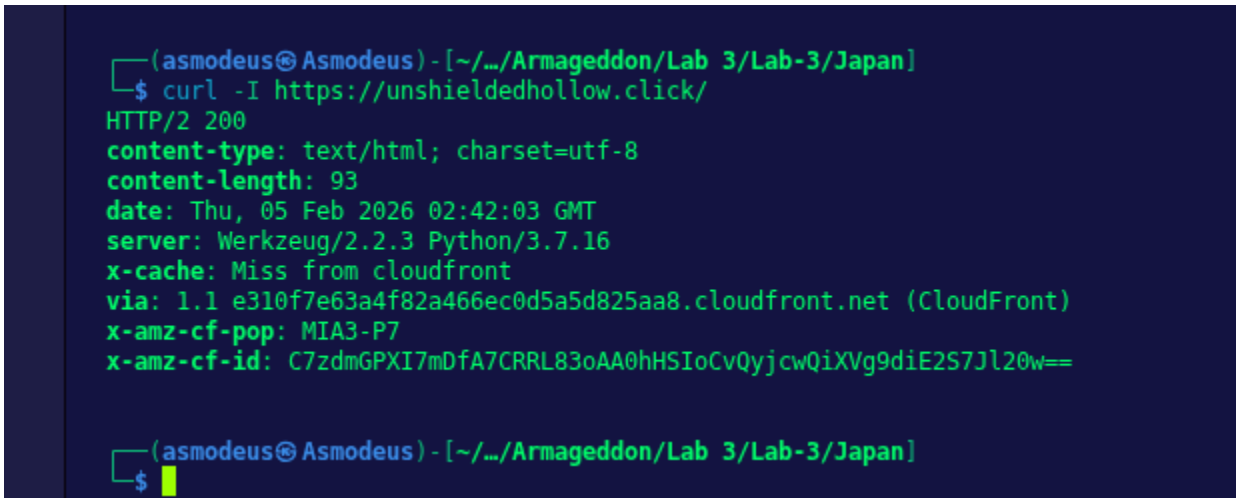
São Paulo: No RDS

```
aws rds describe-db-instances --region sa-east-1 \
--query "DBInstances[].DBInstanceIdentifier"
```



2) Edge proof (CloudFront logs show cache + access)
Students capture request headers:

curl -I https://chewbacca-growls.com/api/public-feed

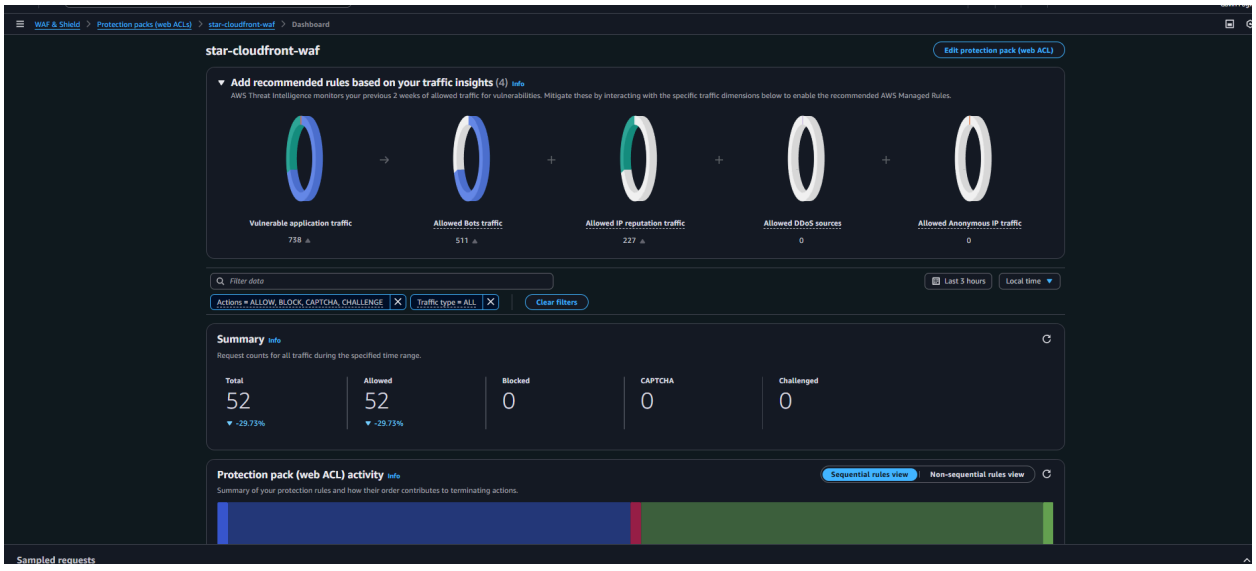


And/or submit CloudFront standard log evidence (Hit/Miss/RefreshHit)

3) WAF proof

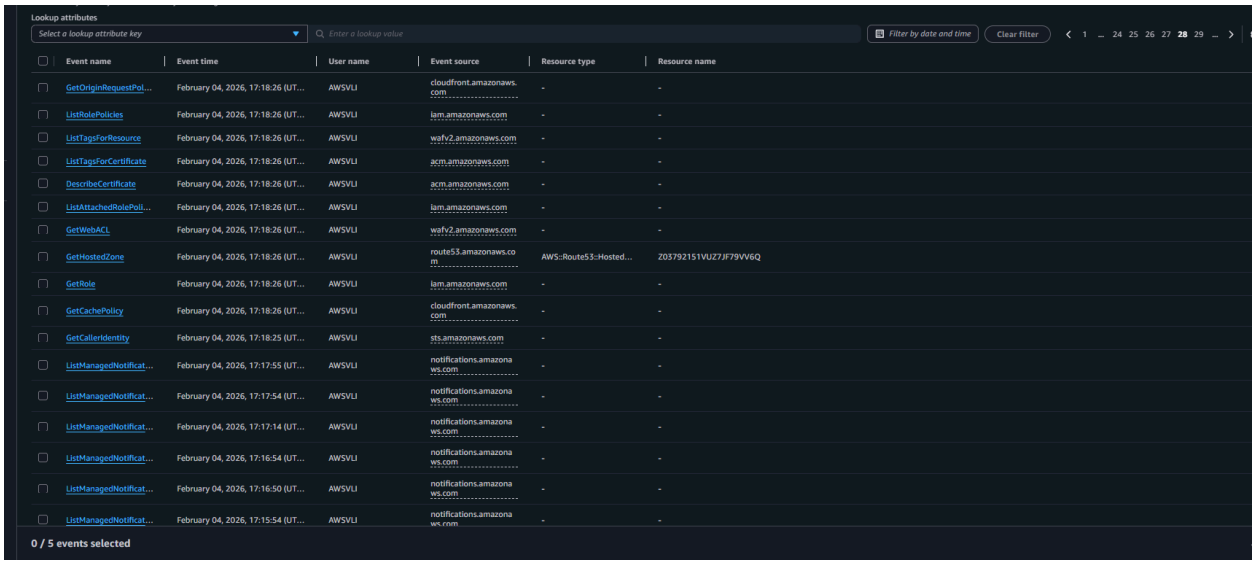
Provide:

- WAF log snippet or Insights summary
- WAF logging destination options are documented

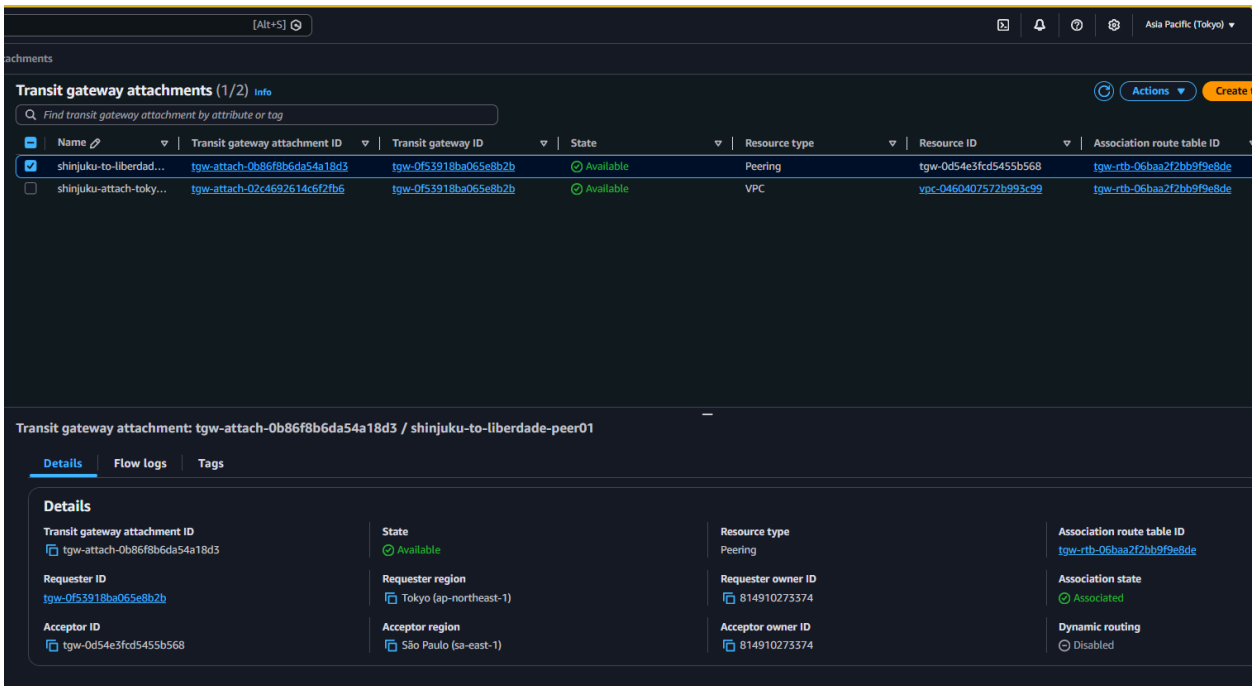


4) Change proof (CloudTrail)
CloudTrail has event history with a 90-day immutable record of management events

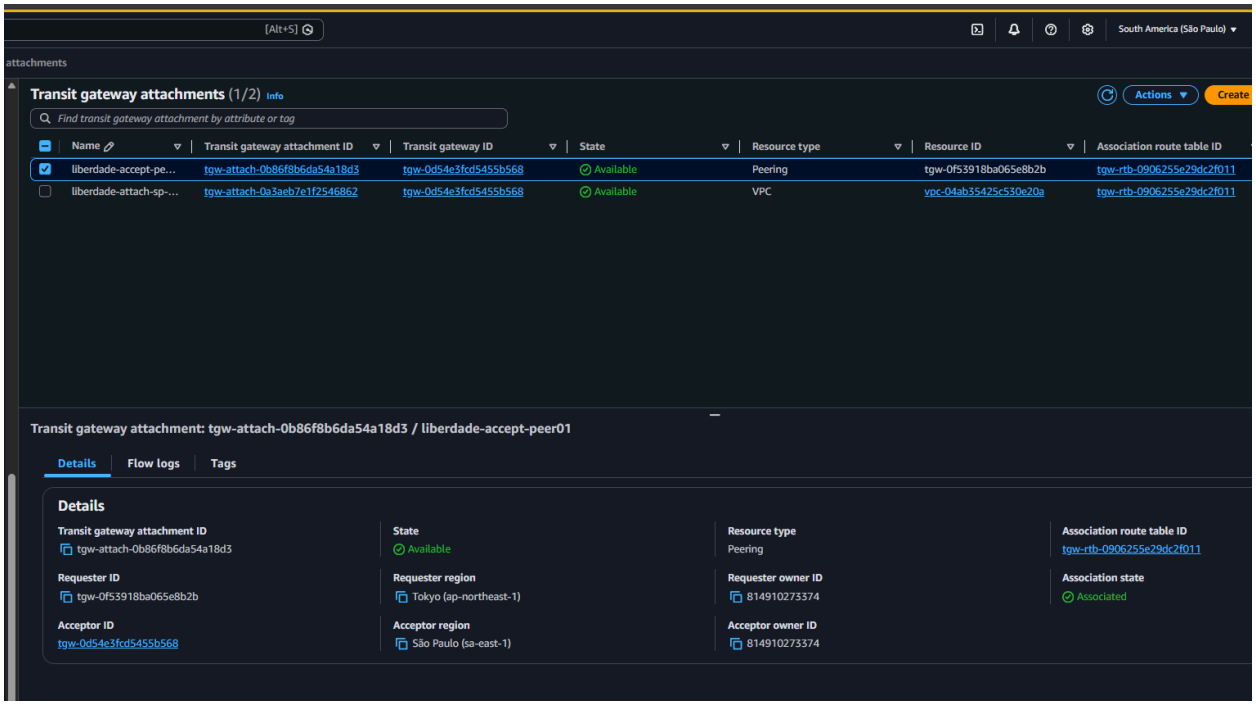
Students capture:
--> “who changed SG / TGW route / WAF / CloudFront config”



5) Network corridor proof (TGW)
Students prove:
TGW attachments exist in both regions
routes point cross-region CIDRs to TGW
Tokyo



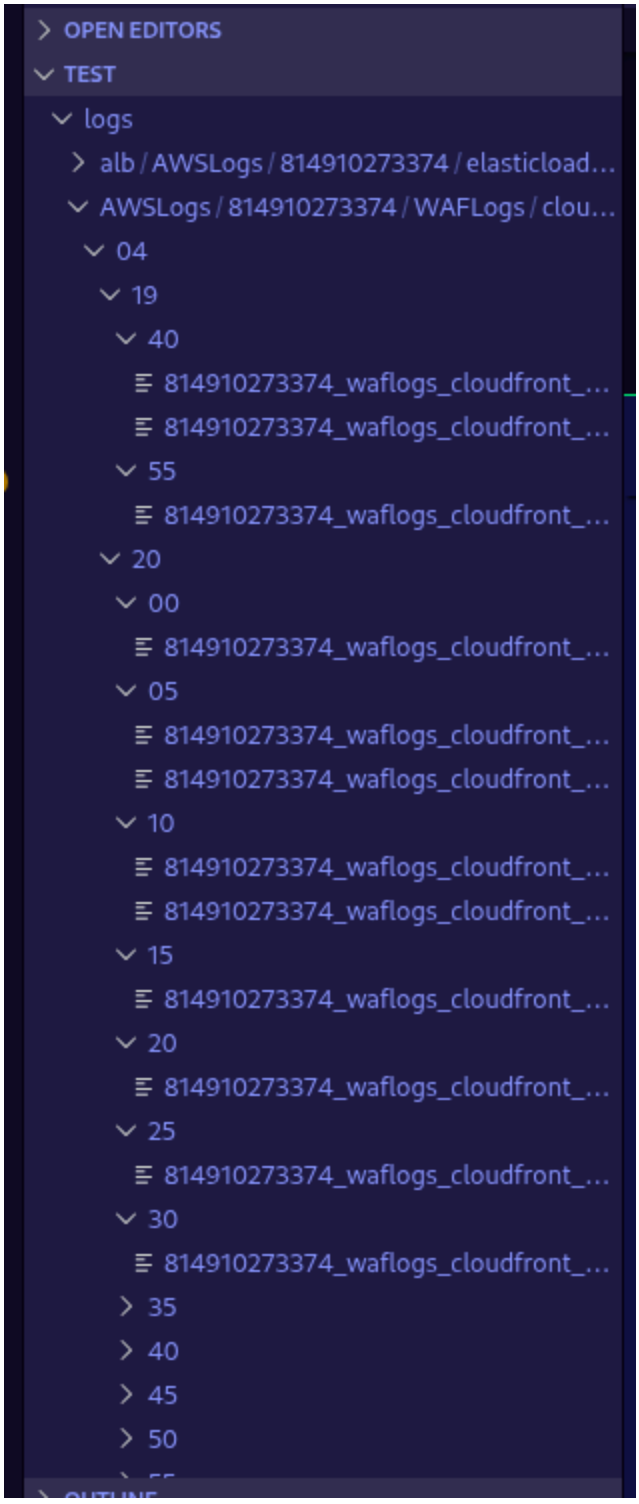
Sao-Paulo



6) AWS CLI verification (students can prove the bucket/logs exist)

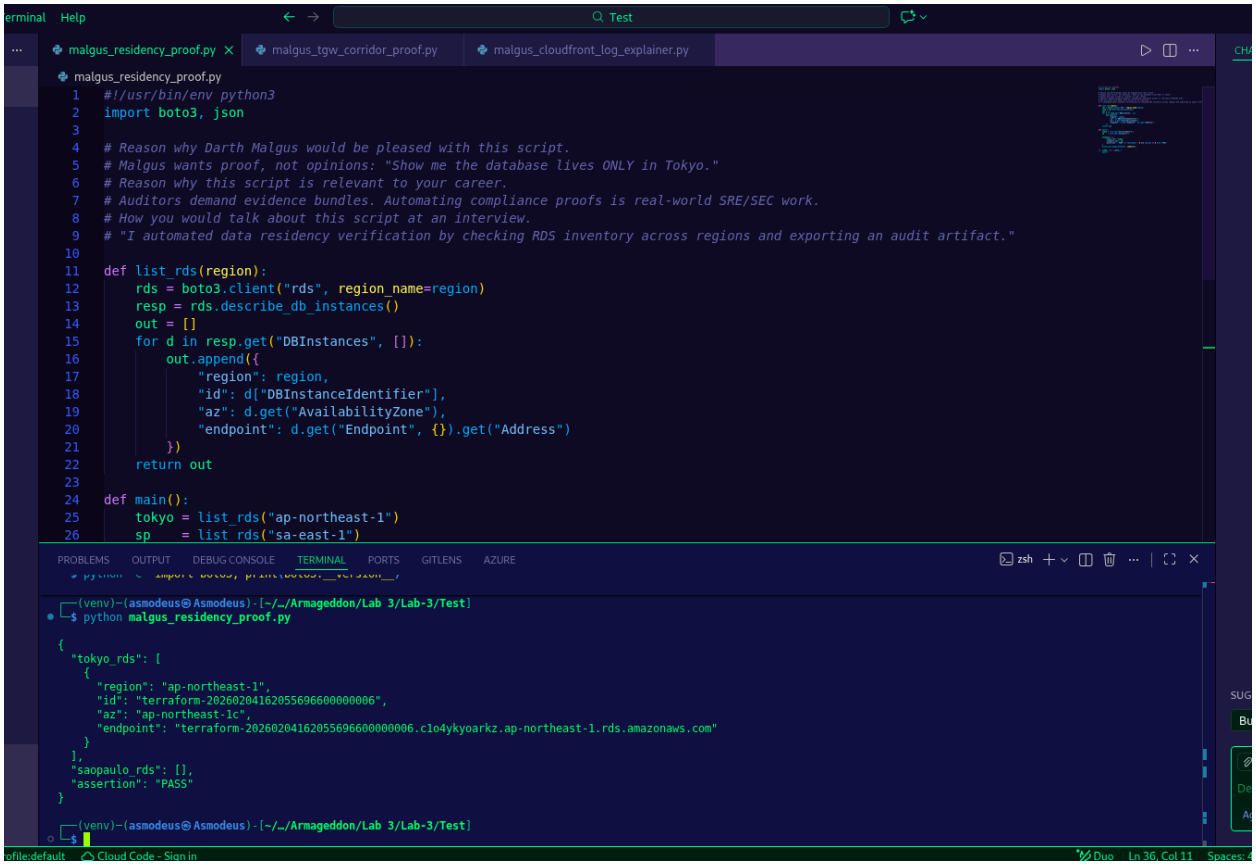
```
aws s3 ls s3://Class_Lab3/
# If logs are under a folder/prefix:
aws s3 ls s3://aws-alb-logs-sa-east-1-star1-814910273374/ --recursive | tail -n 20
```

Download one file manually (sanity check):



aws s3 cp s3://aws-waf-logs-sa-east-1-star1-814910273374/ logs.gz .

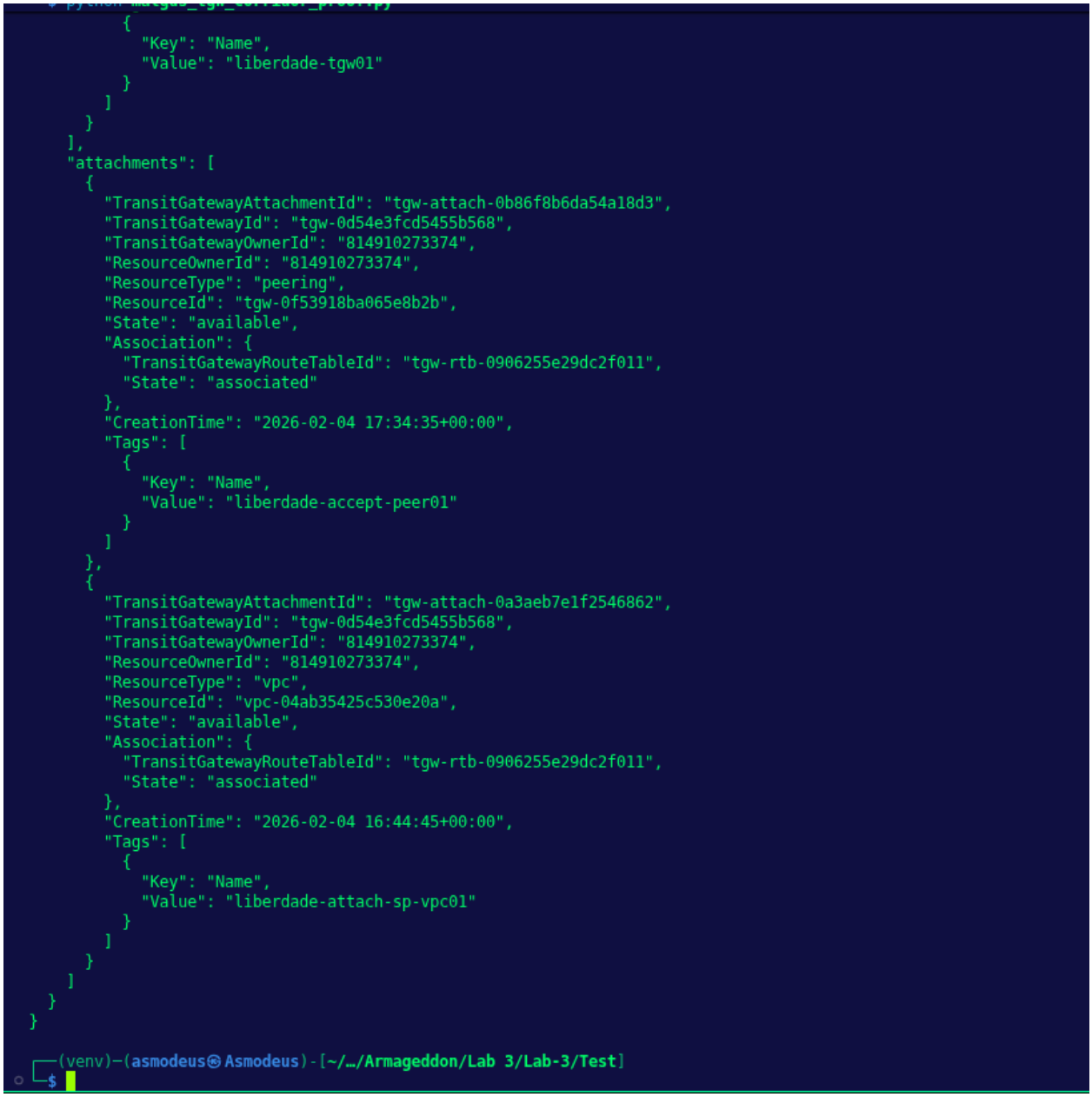
Script 1 — malgus_residency_proof.py
Creates a “DB only in Tokyo” proof file.



Script 2 — malgus_tgw_corridor_proof.py

```
┌──(venv)─(asmodeus㉿Asmodeus)-[~/…/Armageddon/Lab 3/Lab-3/Test]
└─$ python malgus_tgw_corridor_proof.py
{
  "tokyo": {
    "region": "ap-northeast-1",
    "transit_gateways": [
      {
        "TransitGatewayId": "tgw-0f53918ba065e8b2b",
        "TransitGatewayArn": "arn:aws:ec2:ap-northeast-1:814910273374:transit-gateway/tgw-0f53918ba065e8b2b",
        "State": "available",
        "OwnerId": "814910273374",
        "Description": "shinjuku-tgw01 (Tokyo hub)",
        "CreationTime": "2026-02-04 16:20:36+00:00",
        "Options": {
          "AmazonSideAsn": 64512,
          "AutoAcceptSharedAttachments": "disable",
          "DefaultRouteTableAssociation": "disable",
          "DefaultRouteTablePropagation": "disable",
          "VpnEcmpSupport": "enable",
          "DnsSupport": "enable",
          "SecurityGroupReferencingSupport": "disable",
          "MulticastSupport": "disable",
          "EncryptionSupport": {
            "EncryptionState": "disabled"
          }
        },
        "Tags": [
          {
            "Key": "Name",
            "Value": "shinjuku-tgw01"
          }
        ]
      }
    ],
    "attachments": [
      {
        "TransitGatewayAttachmentId": "tgw-attach-0b86f8b6da54a18d3",
        "TransitGatewayId": "tgw-0f53918ba065e8b2b",
        "TransitGatewayOwnerId": "814910273374",
        "ResourceOwnerId": "814910273374",
        "ResourceType": "peering",
        "ResourceId": "tgw-0d54e3fcd5455b568",
        "State": "available",
        "Association": {
          "TransitGatewayRouteTableId": "tgw-rtb-06baa2f2bb9f9e8de",
          "State": "associated"
        },
        "CreationTime": "2026-02-04 17:34:11+00:00",
        "Tags": [
          {
            "Key": "Name",
            "Value": "shinjuku-to-liberdade-peer01"
```

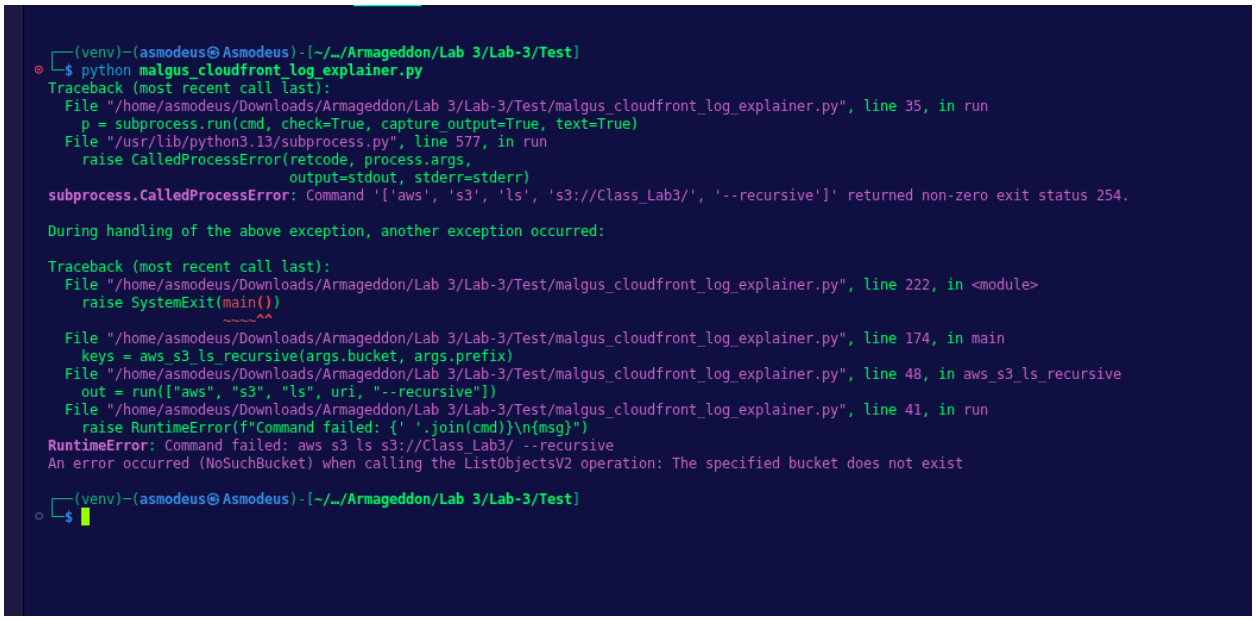
```
            "Value": "shinjuku-to-liberdade-peer01"
          }
        ]
      },
      {
        "TransitGatewayAttachmentId": "tgw-attach-02c4692614c6f2fb6",
        "TransitGatewayId": "tgw-0f53918ba065e8b2b",
        "TransitGatewayOwnerId": "814910273374",
        "ResourceOwnerId": "814910273374",
        "ResourceType": "vpc",
        "ResourceId": "vpc-0460407572b993c99",
        "State": "available",
        "Association": {
          "TransitGatewayRouteTableId": "tgw-rtb-06baa2f2bb9f9e8de",
          "State": "associated"
        },
        "CreationTime": "2026-02-04 16:20:53+00:00",
        "Tags": [
          {
            "Key": "Name",
            "Value": "shinjuku-attach-tokyo-vpc01"
          }
        ]
      }
    ]
  },
  "saopaulo": {
    "region": "sa-east-1",
    "transit_gateways": [
      {
        "TransitGatewayId": "tgw-0d54e3fcd5455b568",
        "TransitGatewayArn": "arn:aws:ec2:sa-east-1:814910273374:transit-gateway/tgw-0d54e3fcd5455b568",
        "State": "available",
        "OwnerId": "814910273374",
        "Description": "liberdade-tgw01 (Sao Paulo spoke)",
        "CreationTime": "2026-02-04 16:44:17+00:00",
        "Options": {
          "AmazonSideAsn": 64512,
          "AutoAcceptSharedAttachments": "disable",
          "DefaultRouteTableAssociation": "disable",
          "DefaultRouteTablePropagation": "disable",
          "VpnEcmpSupport": "enable",
          "DnsSupport": "enable",
          "SecurityGroupReferencingSupport": "disable",
          "MulticastSupport": "disable",
          "EncryptionSupport": {
            "EncryptionState": "disabled"
          }
        },
        "Tags": [
          {
            "Key": "Name",
            "Value": "liberdade-tgw01"
          }
```



Shows TGW attachments + routes that form the “legal corridor”.

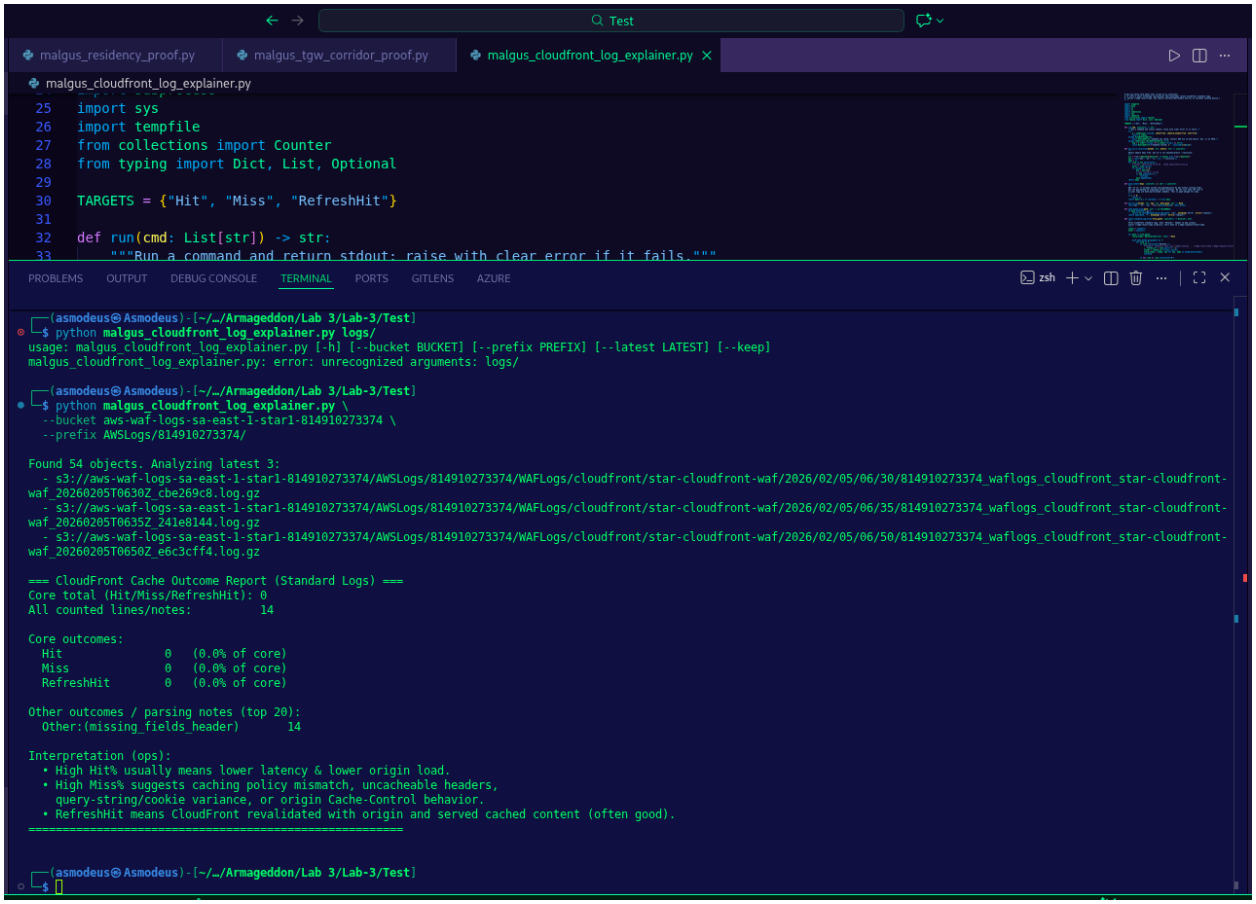
Script 3 — malgus_cloudtrail_last_changes.py
Pulls recent CloudTrail events for “who changed what”.
--> Event history is available by default; it provides a 90-day record of management events.
Not Created Yet
Script 4 — malgus_waf_summary.py
Summarizes WAF logs (Allow vs Block) from CloudWatch Logs destination.
WAF logging destinations: CloudWatch Logs, S3, Firehose.
Not Created Yet

Script 5 — malgus_cloudfront_log_explainer.py (optional)
If you ingest CloudFront standard logs into S3, this script reads a log file and counts Hit/Miss/RefreshHit.
Not Working



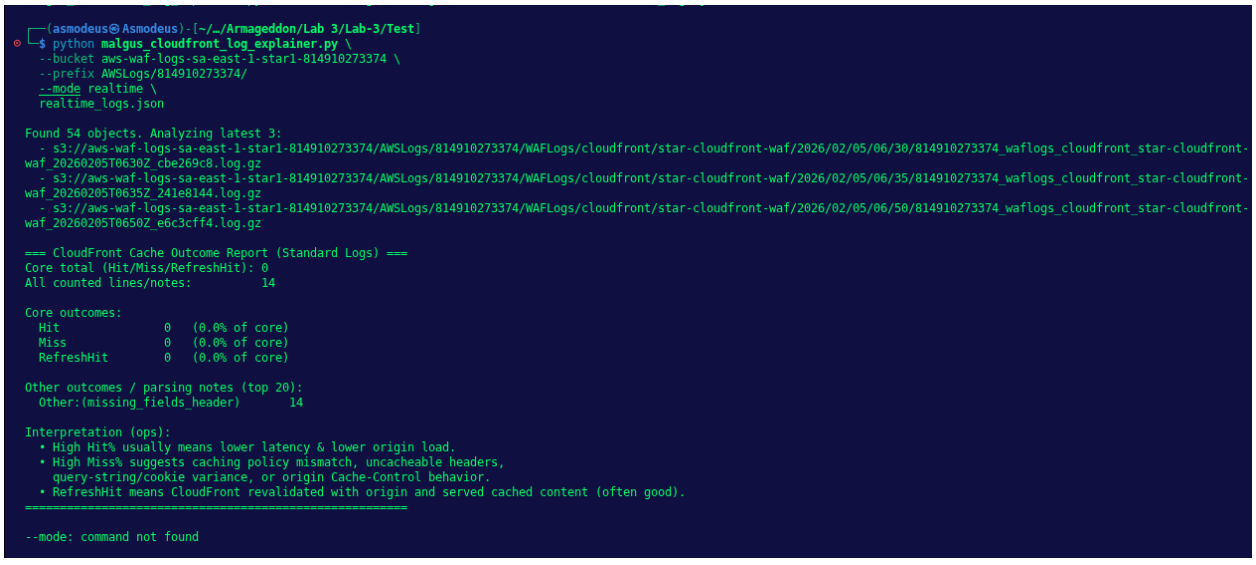
CloudFront standard logs reference Hit / RefreshHit semantics.
A) Standard logs in S3 (downloaded locally)

python3 malgus_cloudfront_log_explainer.py --mode standard cloudfront.log.gz
python3 malgus_cloudfront_log_explainer.py --mode standard cloudfront_part1.log cloudfront_part2.log



B) Real-time logs as JSON lines

python3 malgus_cloudfront_log_explainer.py --mode realtime realtime_logs.jsonl



Final Lab Assumptions (Locked)

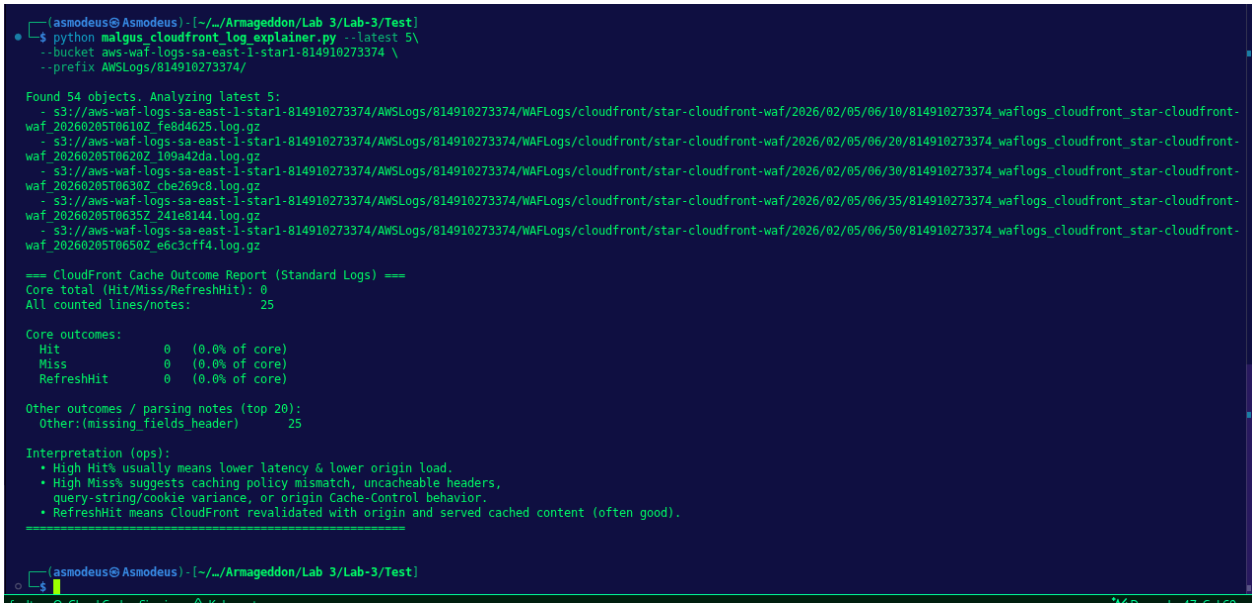
S3 Bucket: Class_Lab3

CloudFront Logs Prefix: Chwebacca-logs/ ← intentionally misspelled

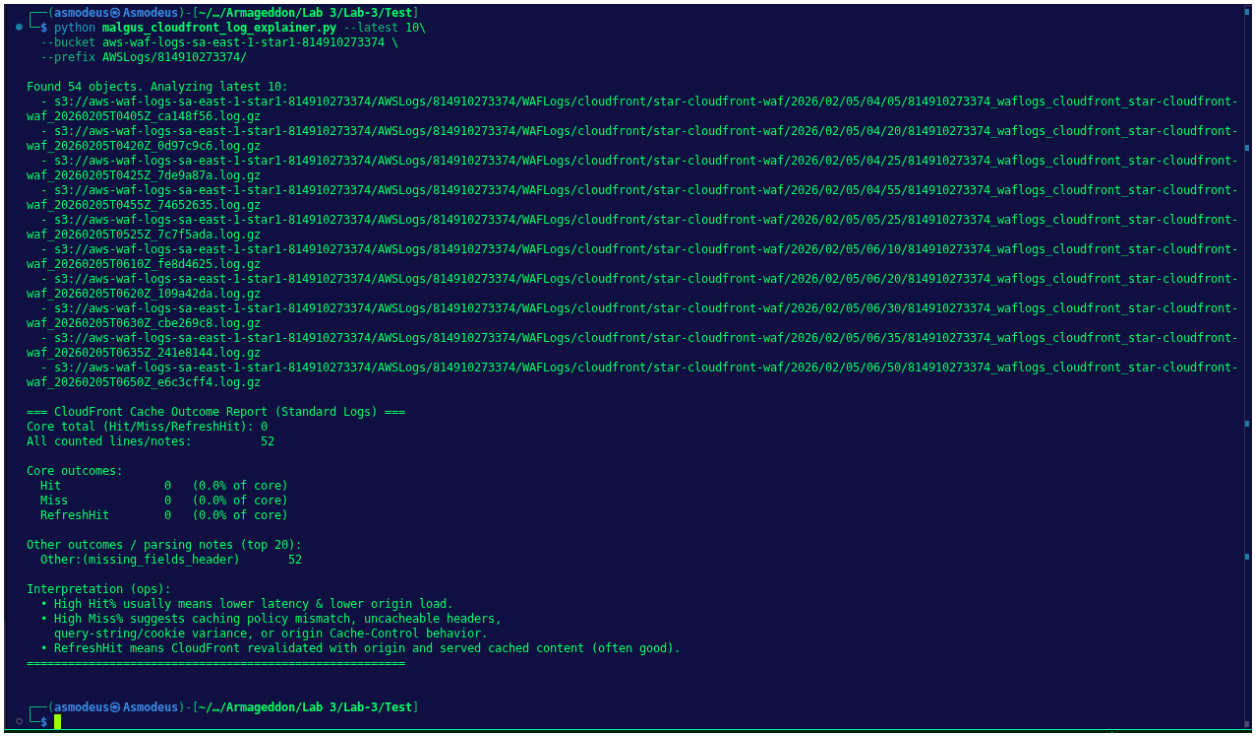
AWS Account ID: 200819971986

Running Scripts:

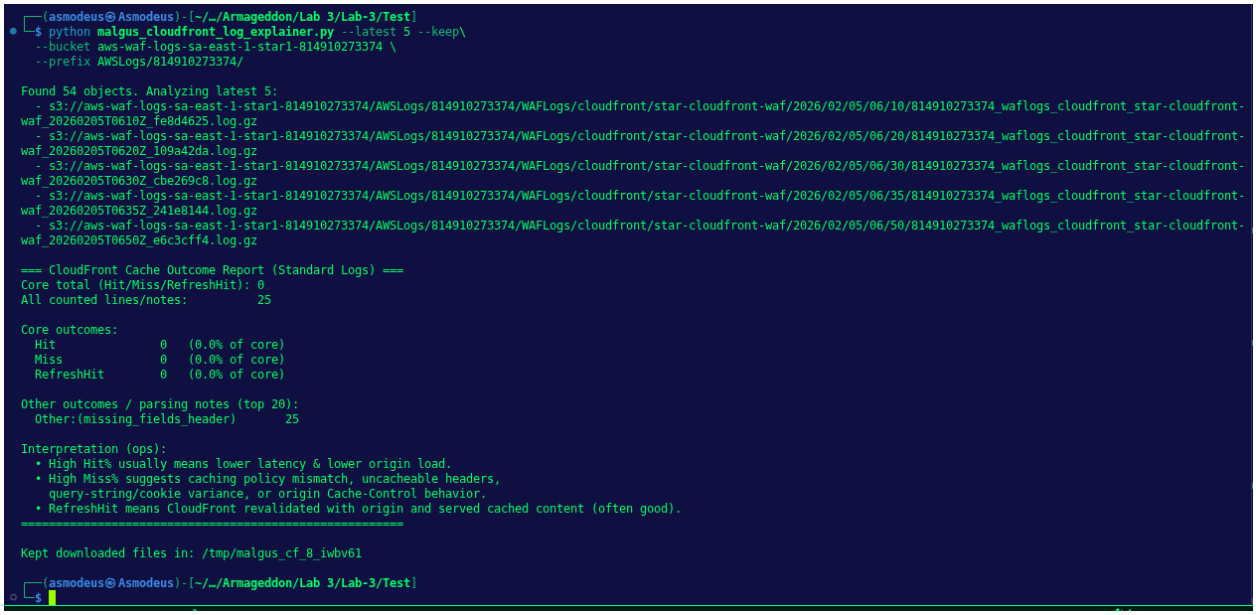
python3 malgus_cloudfront_log_explainer.py --latest 5



python3 malgus_cloudfront_log_explainer.py --prefix cloudfront-logs/ --latest 10



python3 malgus_cloudfront_log_explainer.py --prefix cloudfront-logs/ --latest 5 --keep



From stdin (nice for pipelines)

zcat cloudfront.log.gz | python3 malgus_cloudfront_log_explainer.py --mode standard -

Where “Hit / Miss / RefreshHit” come from (student-facing truth)

In standard CloudFront logs, you usually read the field:

x-edge-result-type (primary)

sometimes also x-edge-response-result-type

Values commonly include: Hit, Miss, RefreshHit, plus other states like Error, LimitExceeded, etc.

That’s why the script reports “Other:*” — so students don’t blindly ignore unusual outcomes.