AIRFLOW SUMMIT

Enhancing Airflow REST API From Basic Integration to Enterprise Scale

Vishal Vijayvargiya
Sr. Software Engineer - AWS

AIRFLOW SUMMIT

What is the Airflow REST API?

- RESTful interface provided by Apache Airflow

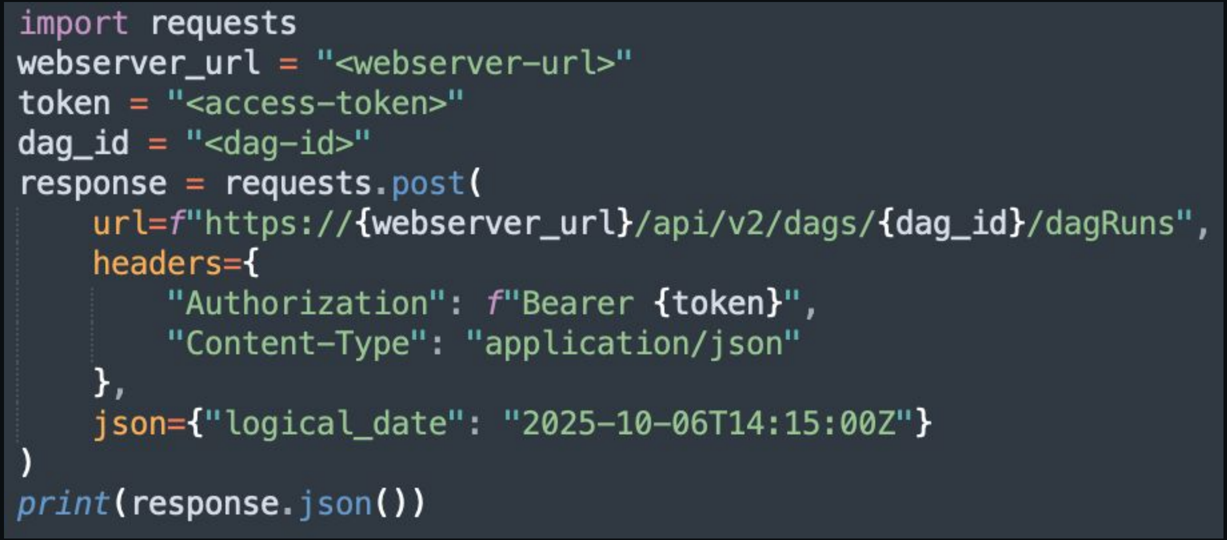
- Enables programmatic control of Airflow:
  - Trigger DAG runs
  - Pause/unpause DAGs
  - Manage connections, variables, and pools

- Why it matters:
  - Allows Airflow to integrate with external systems and automation tools
  - Forms the foundation for workflow orchestration at scale

AIRFLOW SUMMIT

Example

- Trigger a DAG run



```python
import requests
webserver_url = "<webserver-url>"
token = "<access-token>"
dag_id = "<dag-id>"
response = requests.post(
    url=f"https://{webserver_url}/api/v2/dags/{dag_id}/dagRuns",
    headers={
        "Authorization": f"Bearer {token}",
        "Content-Type": "application/json"
    },
    json={"logical_date": "2025-10-06T14:15:00Z"}
)
print(response.json())
```

AIRFLOW SUMMIT

Why Vanilla REST API Isn’t Enough for Enterprises

- Limited security → hard to integrate with IAM
- Exposed webserver → networking/security risks
- Hard to scale across multiple teams/orgs → inconsistent patterns
- Limited audit/compliance visibility → no centralized logs

AIRFLOW SUMMIT

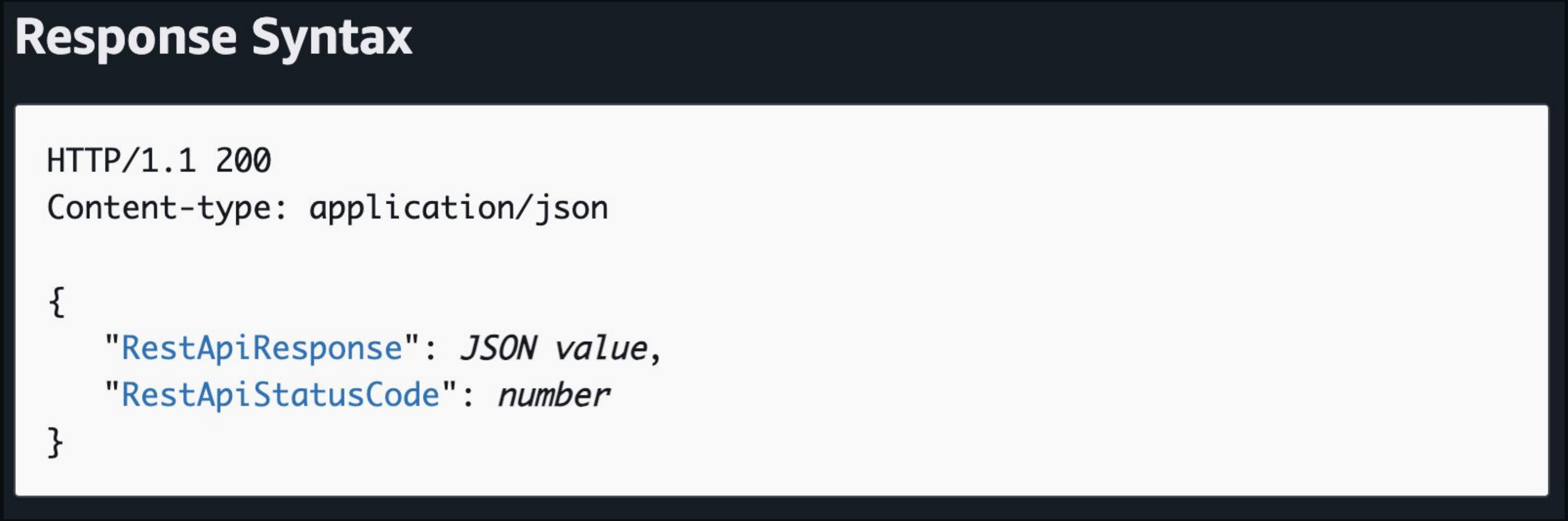
Amazon MWAA InvokeRestAPI

- Simplifies calling Airflow REST API endpoints securely
- Works across:
  - AWS CLI (aws mwaa invoke-rest-api)
  - SDKs (boto3, etc.)
  - Cloud integrations (Step Functions, Lambda, CI/CD pipelines)
  - MwaaDagRunSensor and MwaaTaskRunSensor (apache-airflow-providers-amazon)

Request Syntax

```
POST /restapi/Name HTTP/1.1
Content-type: application/json

{
   "Body": JSON value,
   "Method": "string",
   "Path": "string",
   "QueryParameters": JSON value
}
```

AIRFLOW SUMMIT



Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "RestApiResponse": JSON value,
   "RestApiStatusCode": number
}
```

AIRFLOW SUMMIT

Example

- Trigger a DAG run

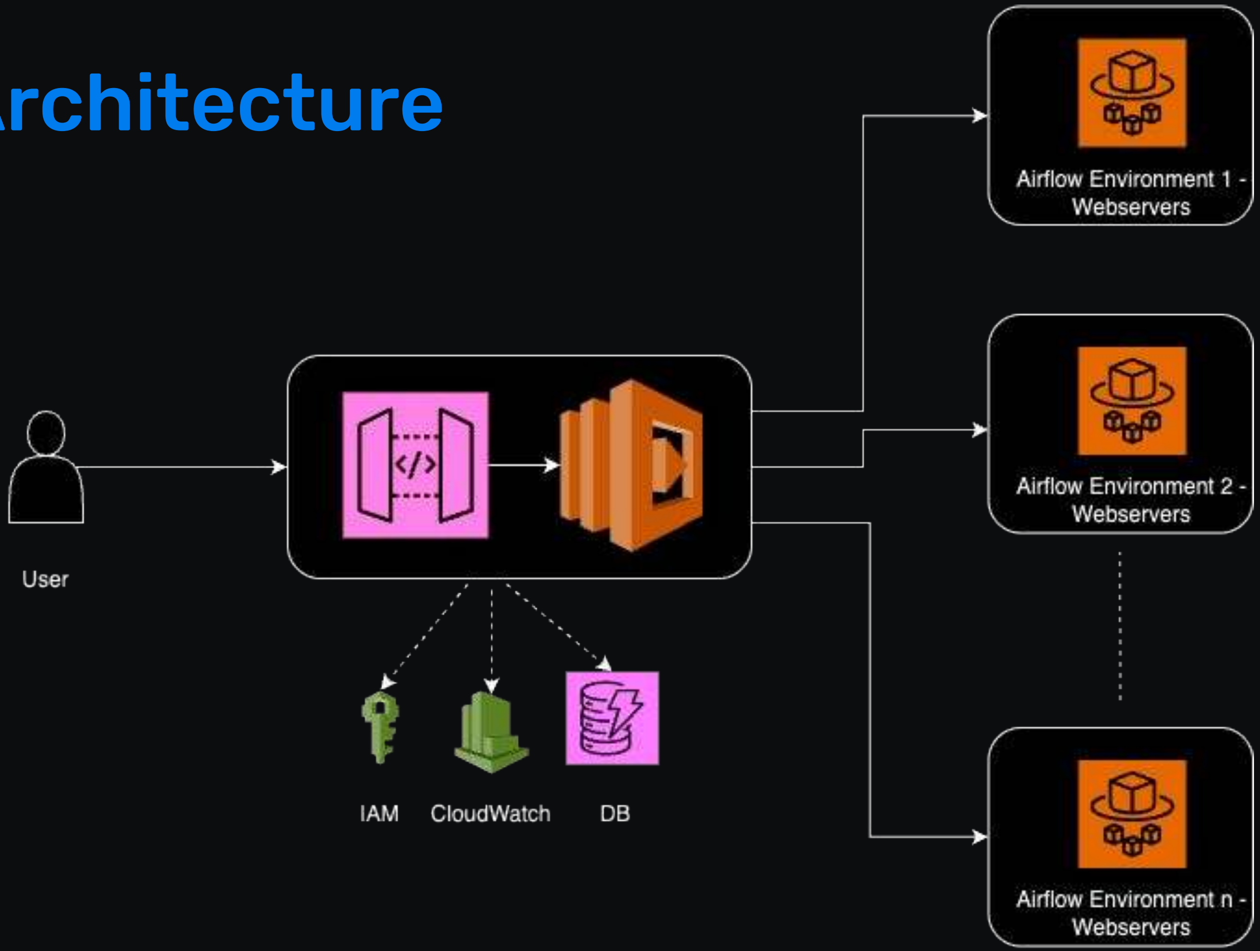
```python
import boto3
import json

client = boto3.client("mwaa")
mwaa_env = "mwaa-env-name"
dag_id = "<dag-id>"

response = client.invoke_rest_api(
    Name=mwaa_env,
    Path=f"/dags/{dag_id}/dagRuns",
    Method="POST",
    Body=json.dumps({"logical_date": "2025-10-06T10:00:00Z"})
)

print(response["RestApiResponse"])
```

IAM Permissions

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowMwaaRestApiAccess",
      "Effect": "Allow",
      "Action": "airflow:InvokeRestApi",
      "Resource": [
      "arn:aws:airflow:us-east-1:111122223333:role/{your-environment-name}/{airflow-role}"
      ]
    }
  ]
}
```

High Level Architecture

- Considerations:
  - Webserver Autoscaling
  - CPU Utilization and Active Connection Count
  - Error Handling



User
IAM
CloudWatch
DB
Airflow Environment 1 - Webservers
Airflow Environment 2 - Webservers
Airflow Environment n - Webservers

🔄 CI/CD for Data Pipelines

- Automate DAG validation & deployment via CLI/SDK

⚡ Event-Driven Workflows (Data Ingestion + ML Retraining)

- Trigger pipelines securely from SaaS apps
- Secure, auditable event-based triggers

🔗 Cross-Service Orchestration

- Step Functions + Airflow
- Lambda + Airflow
- Hybrid Orchestration (On-prem + Cloud)

🛡️ Audit & Compliance

- CloudTrail logging of all API calls
- Centralized monitoring dashboards

Example

3.0

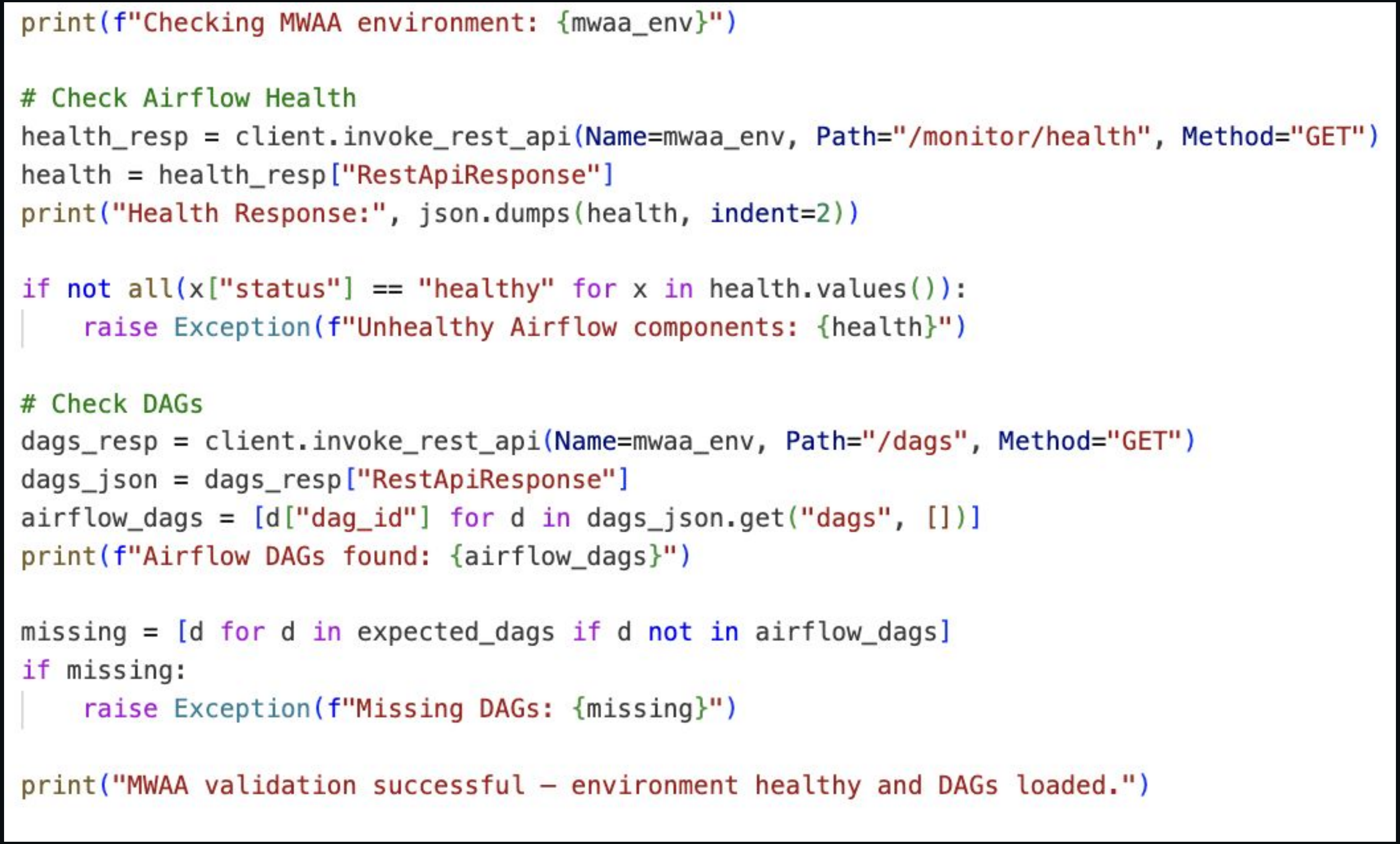
Automated Post-Deployment Health & DAG Validation

- An organization deploys Airflow DAGs frequently through automated CI/CD pipelines.
- After every DAG deployment to MWAA, teams need to ensure:
  - Airflow components are **healthy** (scheduler, triggerer, database).
  - All expected DAGs are **parsed and visible** in Airflow.
- There’s **no built-in mechanism** to detect silent DAG parsing failures.
- Teams need an **automated, secure, IAM-based validation** process that runs immediately after each deployment.

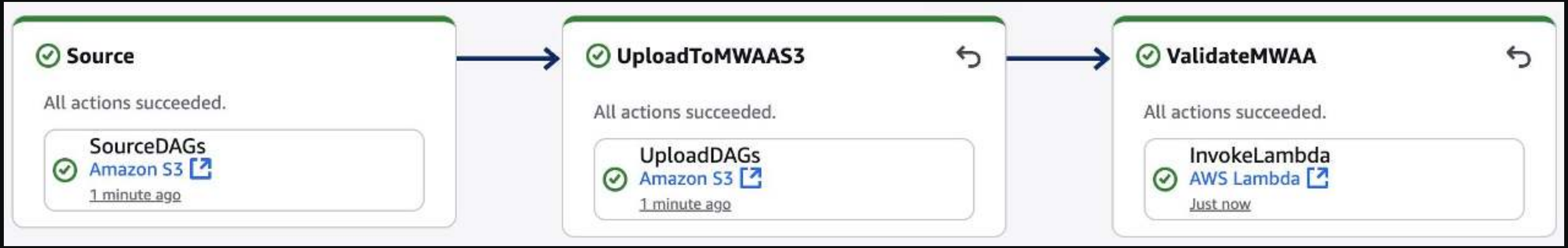
AIRFLOW SUMMIT

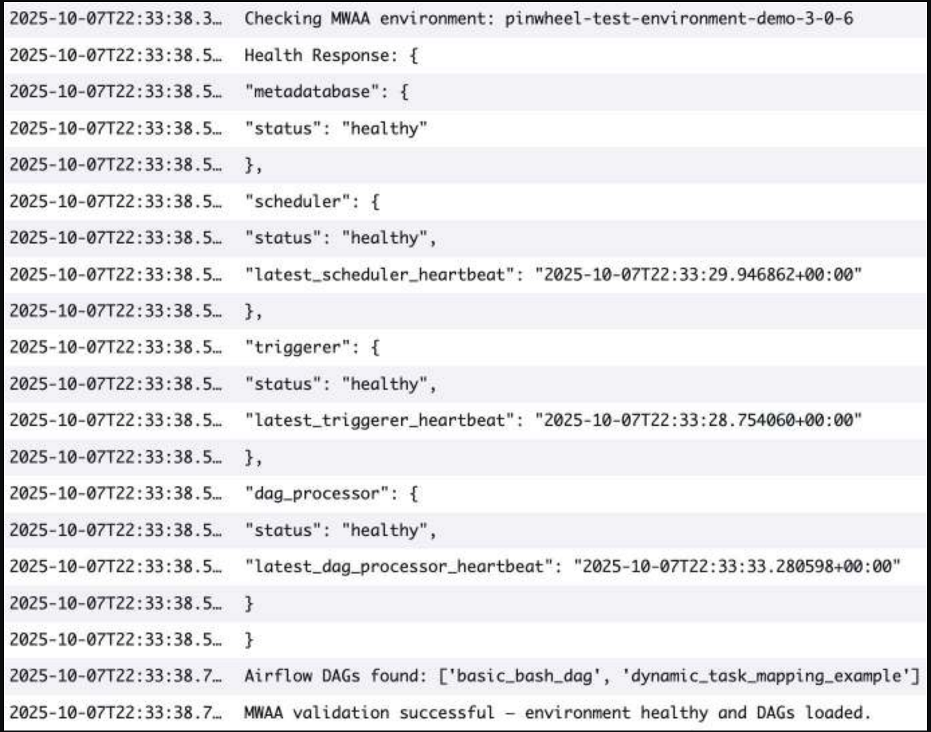
Automated Health & DAG Validation Using MWAA InvokeRestApi

- Use AWS Lambda triggered after each deployment to:
  - Call /monitor/health → ensure Airflow components are operational
  - Call /dags → verify all expected DAGs are successfully parsed
- Fails pipeline automatically if any validation step fails
- Fully IAM-authenticated, no public exposure
- CloudTrail-logged for audit and compliance

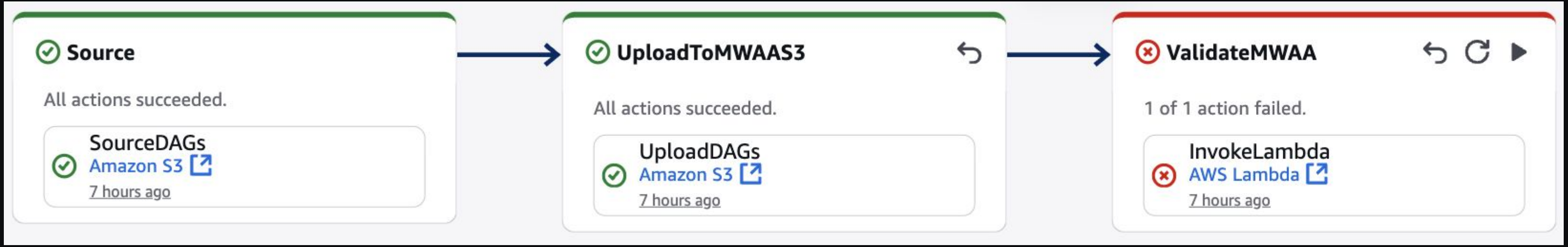


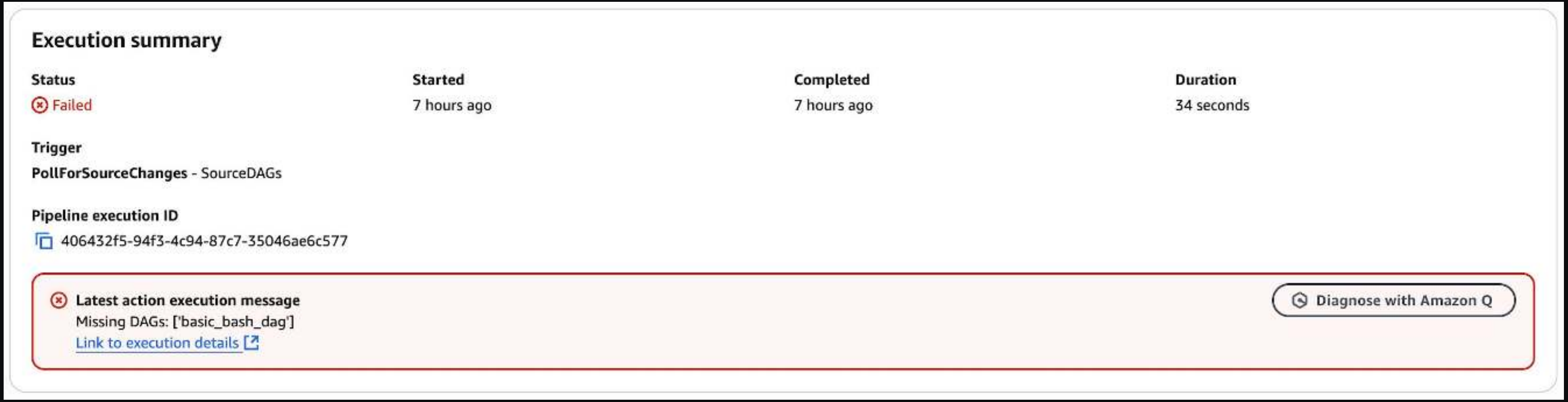
```python
print(f"Checking MWAA environment: {mwaa_env}")

# Check Airflow Health
health_resp = client.invoke_rest_api(Name=mwaa_env, Path="/monitor/health", Method="GET")
health = health_resp["RestApiResponse"]
print("Health Response:", json.dumps(health, indent=2))

if not all(x["status"] == "healthy" for x in health.values()):
    raise Exception(f"Unhealthy Airflow components: {health}")

# Check DAGs
dags_resp = client.invoke_rest_api(Name=mwaa_env, Path="/dags", Method="GET")
dags_json = dags_resp["RestApiResponse"]
airflow_dags = [d["dag_id"] for d in dags_json.get("dags", [])]
print(f"Airflow DAGs found: {airflow_dags}")

missing = [d for d in expected_dags if d not in airflow_dags]
if missing:
    raise Exception(f"Missing DAGs: {missing}")

print("MWAA validation successful – environment healthy and DAGs loaded.")
```





```
2025-10-07T22:33:38.3…  Checking MWAA environment: pinwheel-test-environment-demo-3-0-6
2025-10-07T22:33:38.5…  Health Response: {
2025-10-07T22:33:38.5…  "metadatabase": {
2025-10-07T22:33:38.5…  "status": "healthy"
2025-10-07T22:33:38.5…  },
2025-10-07T22:33:38.5…  "scheduler": {
2025-10-07T22:33:38.5…  "status": "healthy",
2025-10-07T22:33:38.5…  "latest_scheduler_heartbeat": "2025-10-07T22:33:29.946862+00:00"
2025-10-07T22:33:38.5…  },
2025-10-07T22:33:38.5…  "triggerer": {
2025-10-07T22:33:38.5…  "status": "healthy",
2025-10-07T22:33:38.5…  "latest_triggerer_heartbeat": "2025-10-07T22:33:28.754060+00:00"
2025-10-07T22:33:38.5…  },
2025-10-07T22:33:38.5…  "dag_processor": {
2025-10-07T22:33:38.5…  "status": "healthy",
2025-10-07T22:33:38.5…  "latest_dag_processor_heartbeat": "2025-10-07T22:33:33.280598+00:00"
2025-10-07T22:33:38.5…  }
2025-10-07T22:33:38.5…  }
2025-10-07T22:33:38.7…  Airflow DAGs found: ['basic_bash_dag', 'dynamic_task_mapping_example']
2025-10-07T22:33:38.7…  MWAA validation successful – environment healthy and DAGs loaded.
```





Execution summary

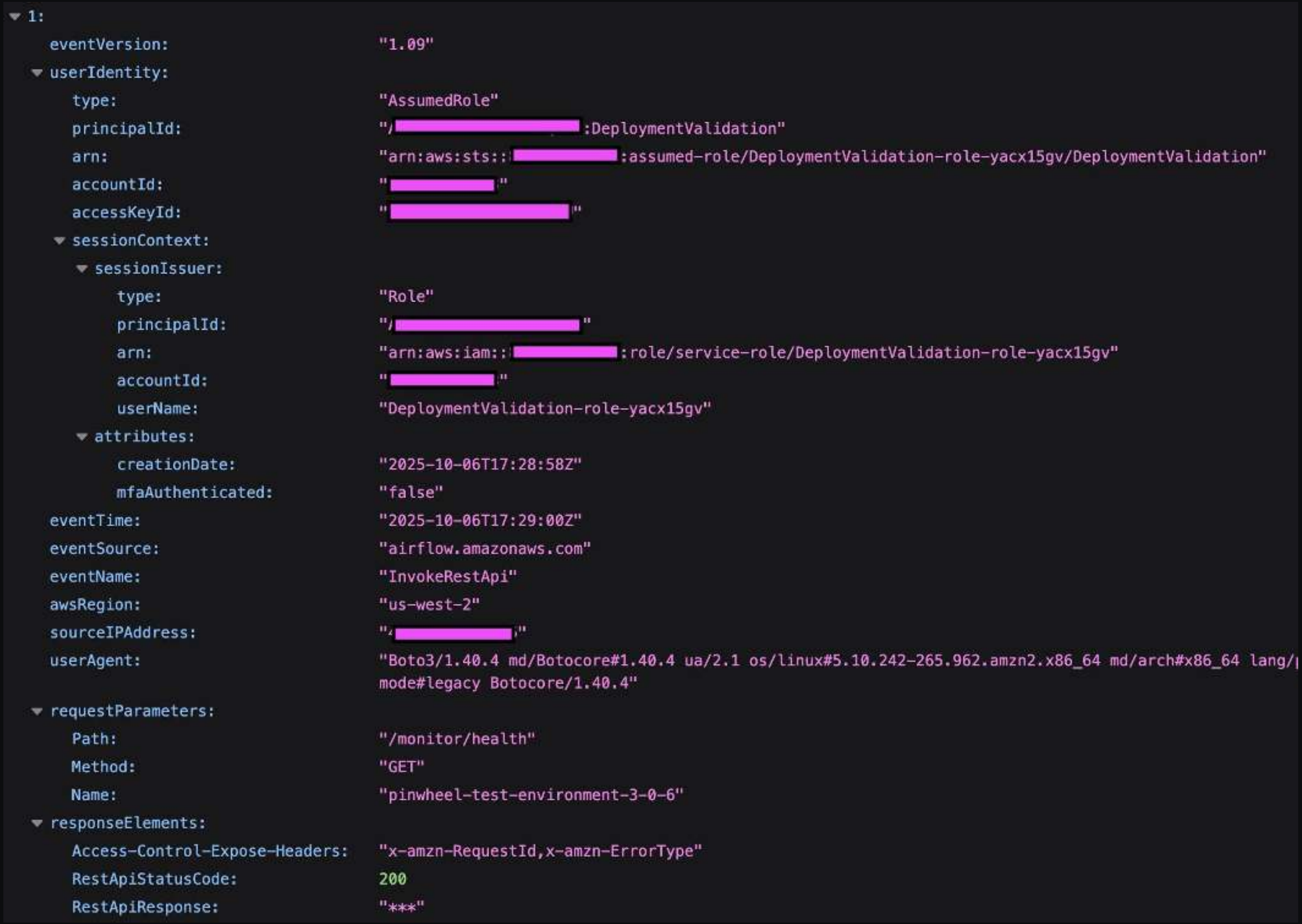
Status	Started	Completed	Duration
Failed	7 hours ago	7 hours ago	34 seconds

Trigger
PollForSourceChanges - SourceDAGs

Pipeline execution ID
406432f5-94f3-4c94-87c7-35046ae6c577

Latest action execution message
Missing DAGs: ['basic_bash_dag']
Link to execution details

CloudTrail Logs



```
▼ 1:
    eventVersion:                     "1.09"
  ▼ userIdentity:
      type:                           "AssumedRole"
      principalId:                    "[redacted]:DeploymentValidation"
      arn:                            "arn:aws:sts::[redacted]:assumed-role/DeploymentValidation-role-yacx15gv/DeploymentValidation"
      accountId:                      "[redacted]"
      accessKeyId:                    "[redacted]"
    ▼ sessionContext:
      ▼ sessionIssuer:
          type:                       "Role"
          principalId:                "[redacted]"
          arn:                        "arn:aws:iam::[redacted]:role/service-role/DeploymentValidation-role-yacx15gv"
          accountId:                  "[redacted]"
          userName:                   "DeploymentValidation-role-yacx15gv"
      ▼ attributes:
          creationDate:               "2025-10-06T17:28:58Z"
          mfaAuthenticated:           "false"
    eventTime:                        "2025-10-06T17:29:00Z"
    eventSource:                      "airflow.amazonaws.com"
    eventName:                        "InvokeRestApi"
    awsRegion:                        "us-west-2"
    sourceIPAddress:                  "[redacted]"
    userAgent:                        "Boto3/1.40.4 md/Botocore#1.40.4 ua/2.1 os/linux#5.10.242-265.962.amzn2.x86_64 md/arch#x86_64 lang/
                                      mode#legacy Botocore/1.40.4"
  ▼ requestParameters:
      Path:                           "/monitor/health"
      Method:                         "GET"
      Name:                           "pinwheel-test-environment-3-0-6"
  ▼ responseElements:
      Access-Control-Expose-Headers:  "x-amzn-RequestId,x-amzn-ErrorType"
      RestApiStatusCode:              200
      RestApiResponse:                "***"
```

AIRFLOW SUMMIT

Closing: Airflow 3

- InvokeRestAPI usage remains fully consistent across Airflow 2 and 3 — no breaking changes
- This abstraction layer shields enterprise workflows from backend changes.

Questions?

linkedin.com/in/vishalvijayvargiya/