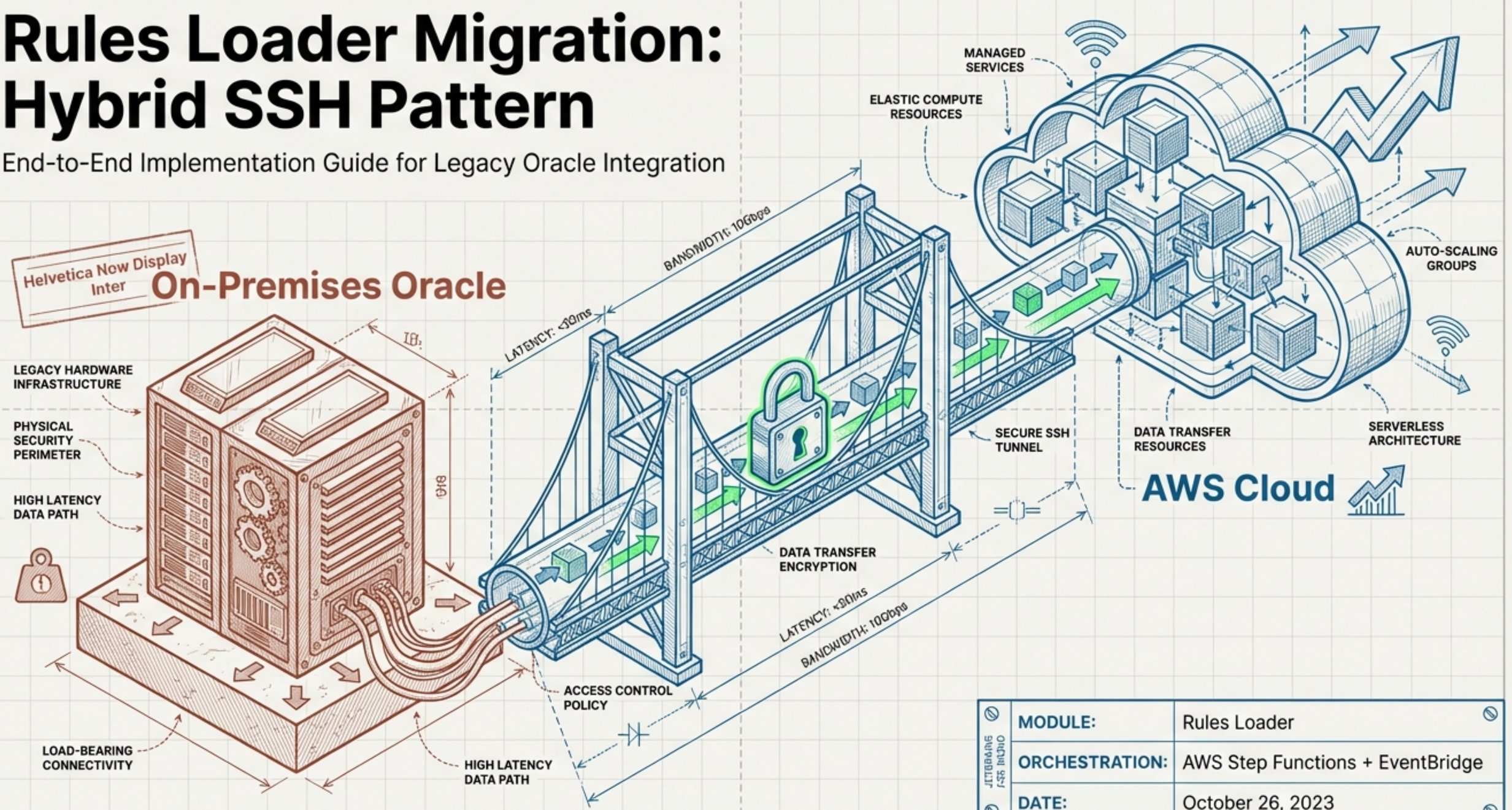
# Rules Loader Migration: Hybrid SSH Pattern

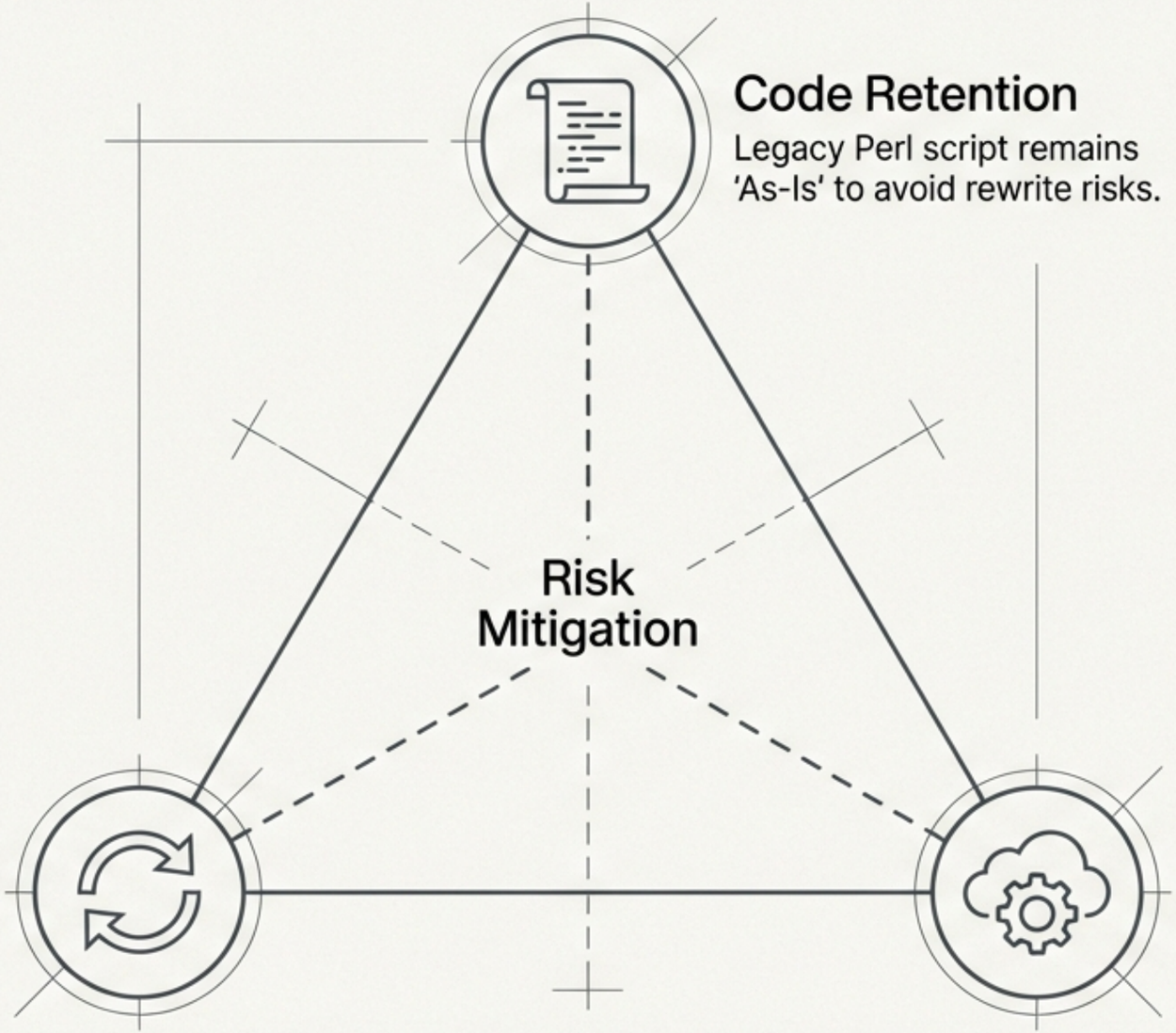
End-to-End Implementation Guide for Legacy Oracle Integration




| | |
|---|---|
| MODULE: | Rules Loader |
| ORCHESTRATION: | AWS Step Functions + EventBridge |
| DATE: | October 26, 2023 |

# Strategic Approach: Orchestration Without Rewriting

**Core Context:** The Rules Loader fetches business rules from the On-Prem Atlas DB (Oracle) and consolidates them for the cloud pipeline.




## Code Retention

Legacy Perl script remains 'As-Is' to avoid rewrite risks.

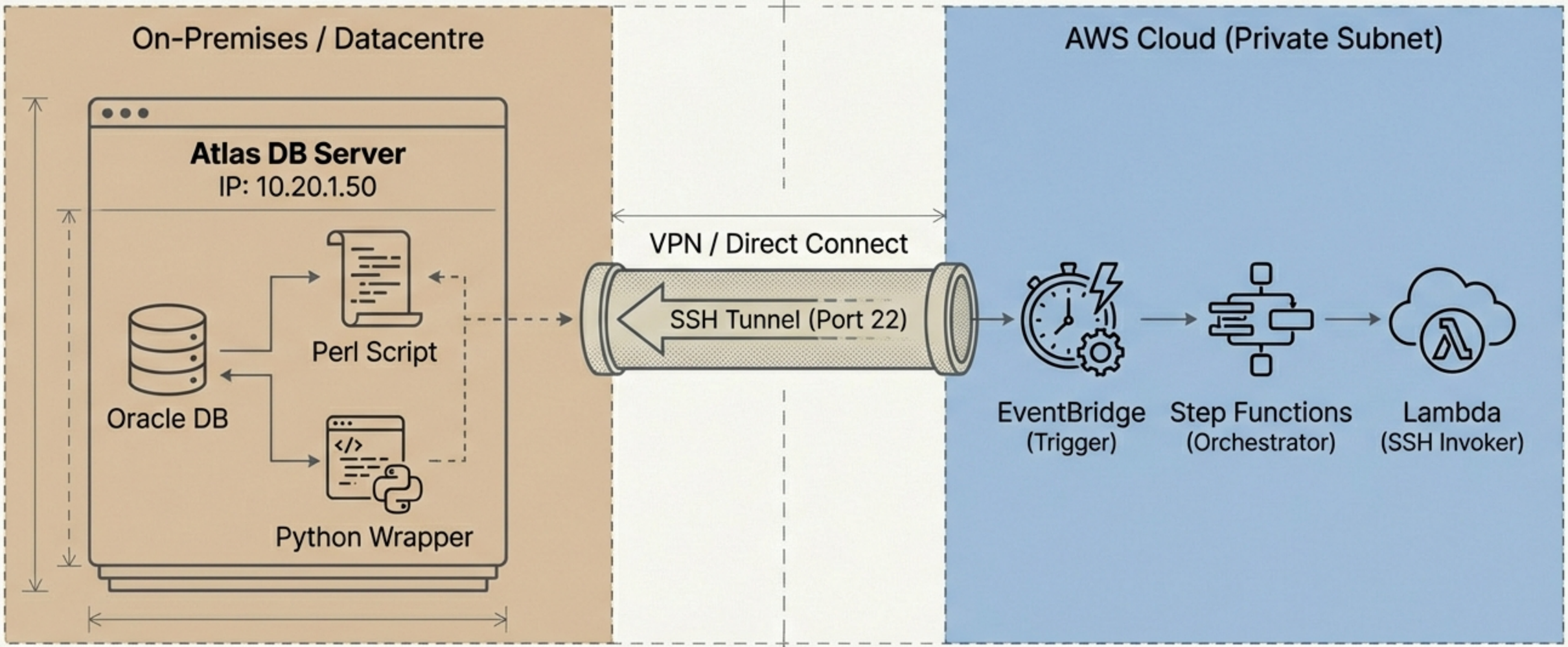
## Callback Pattern

Waits for asynchronous On-Prem completion.

## Hybrid Orchestration

AWS Step Functions triggers job via Secure SSH.

# High-Level Architecture



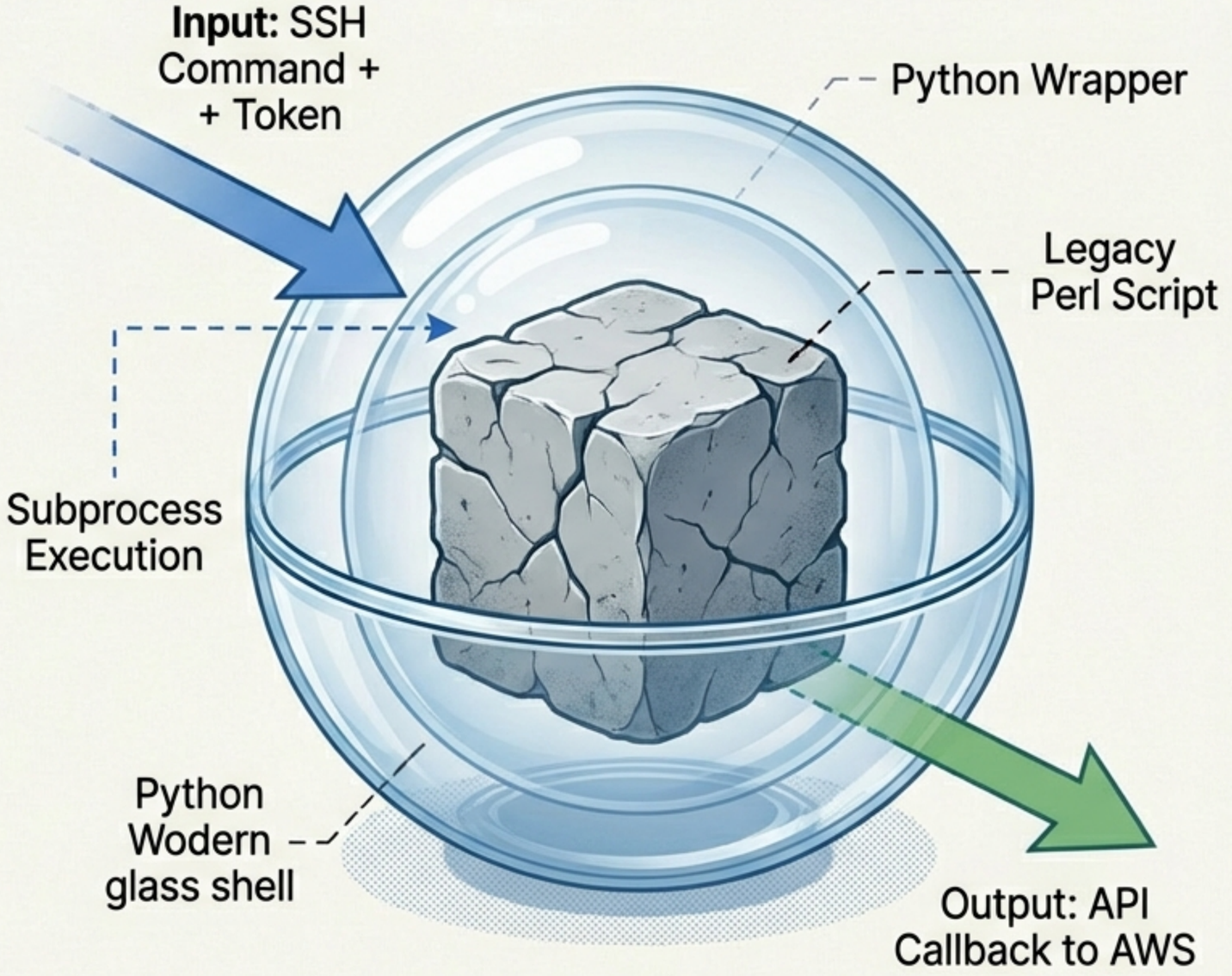

**Key Insight:** The architecture bridges a private subnet in AWS with the specific On-Prem server IP, utilizing a designated Runner Service Account (svc_aws_runner).

# Phase 1: The On-Premises Wrapper Strategy

**Concept:** We deploy a Python wrapper (rules_wrapper.py) alongside the legacy script. This acts as a translator, handling AWS communication while the Perl script remains dumb.

**Wrapper Responsibilities:**

- **Input**: Accepts taskToken & script_path.
- **Execution**: Runs legacy rules_loader.pl via subprocess.
- **Output**: Captures exit codes, STDOUT, STDERR.
- **Callback**: Calls AWS API (SendTaskSuccess/Failure).



# Code Spotlight: rules_wrapper.py

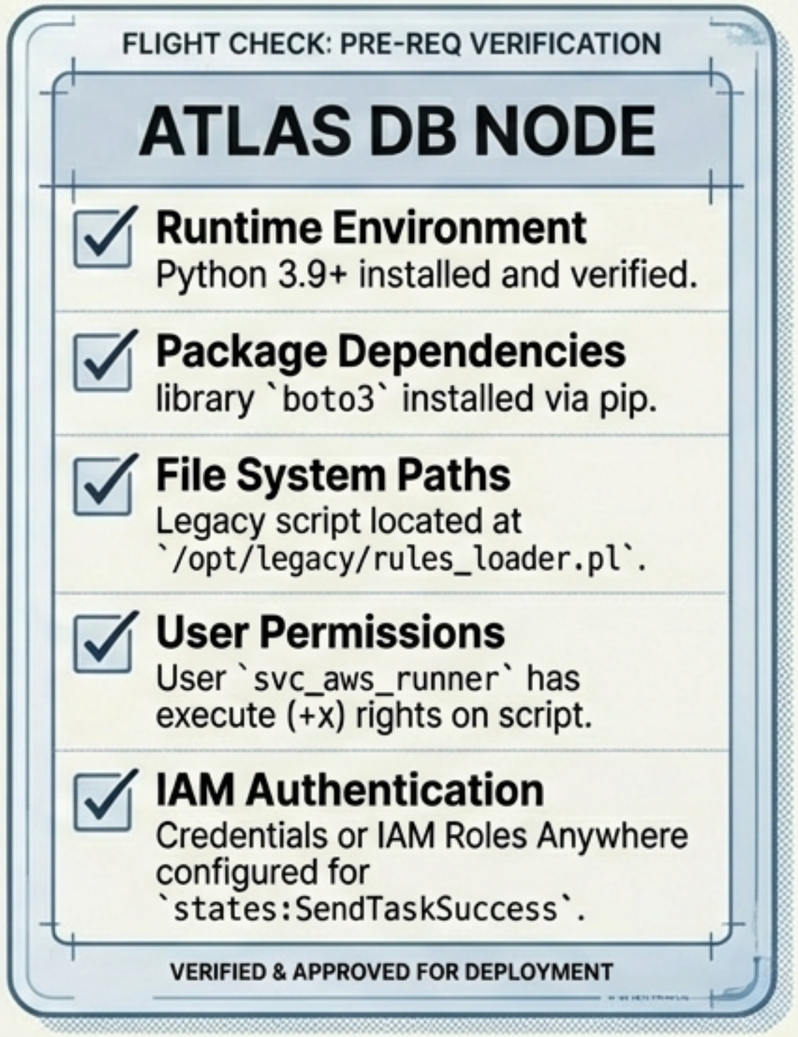
```python
# 1. Run the Legacy Perl Script
result = subprocess.run(["perl", args.script], check=True,
                        capture_output=True, text=True)

# 2. On Success: Send Callback
client.send_task_success(
    taskToken=args.token,
    output=''{"status": "success", "message": "Rules Consolidated to S3"}'
)

# 3. On Script Failure: Send Failure Callback
except subprocess.CalledProcessError as e:
    client.send_task_failure(
        taskToken=args.token,
        error="LegacyScriptError",
        cause=f"Perl Script Failed. Exit Code: {e.returncode}"
    )
```

**Execution:** Runs local binary

**The Handshake:** Returns token to AWS

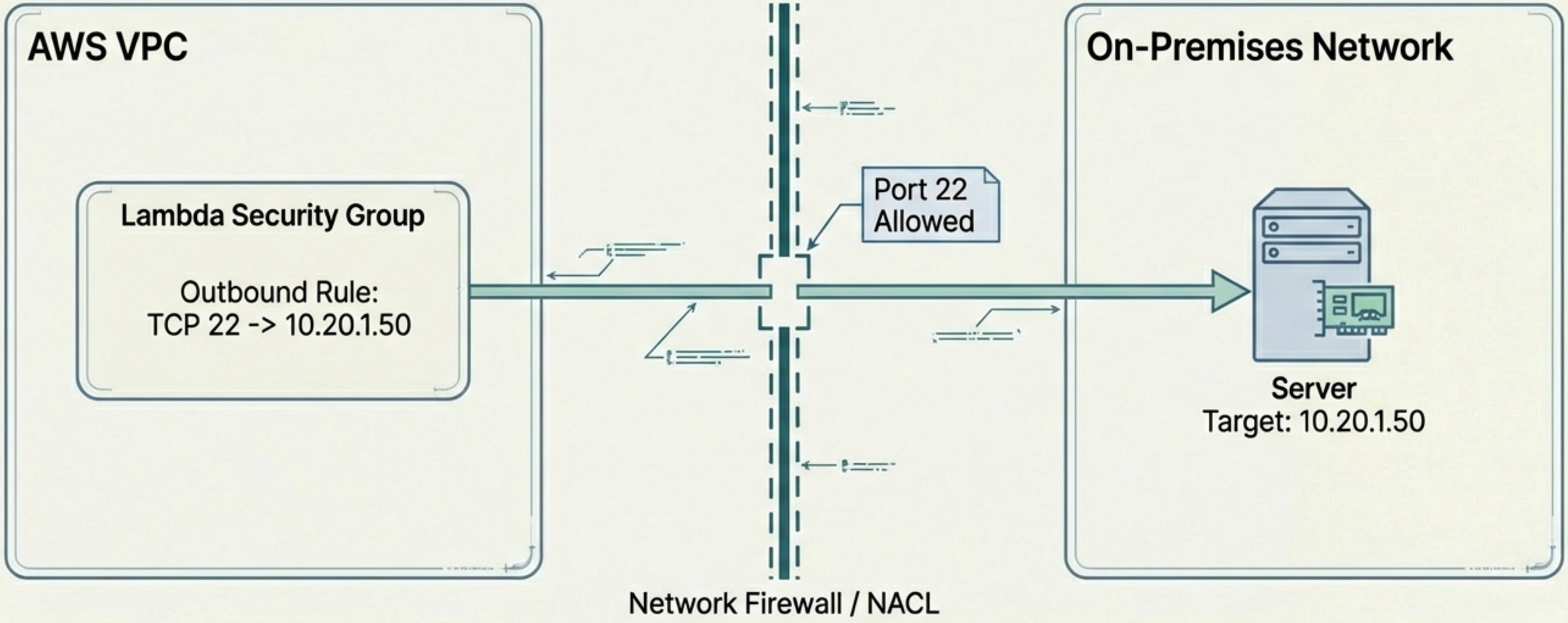
# On-Premises Server Prerequisites

## Target Server: Atlas DB (10.20.1.50)



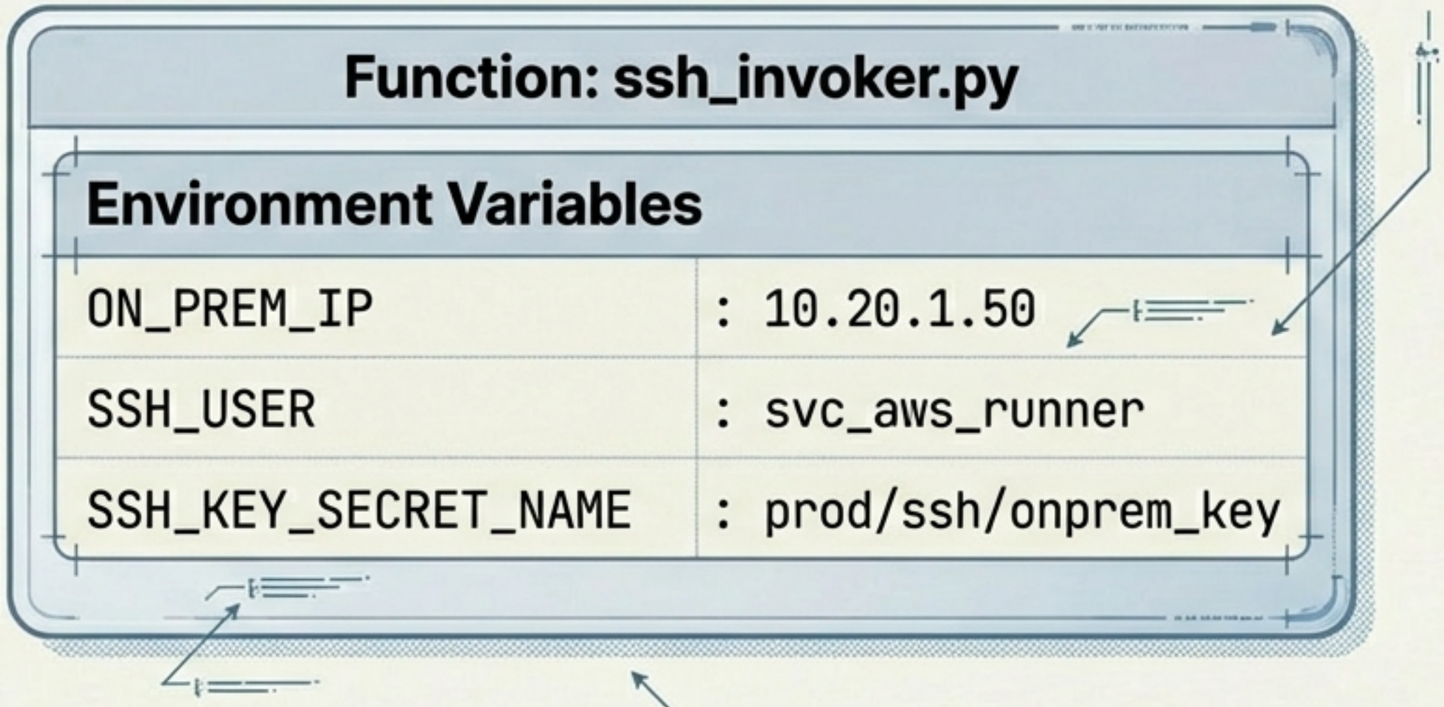
# Phase 2: AWS Infrastructure & Connectivity

**Network Path:** Verified connectivity between AWS VPC Private Subnet and On-Prem Server 10.20.1.50.



# The ‘SSH Invoker’ Lambda

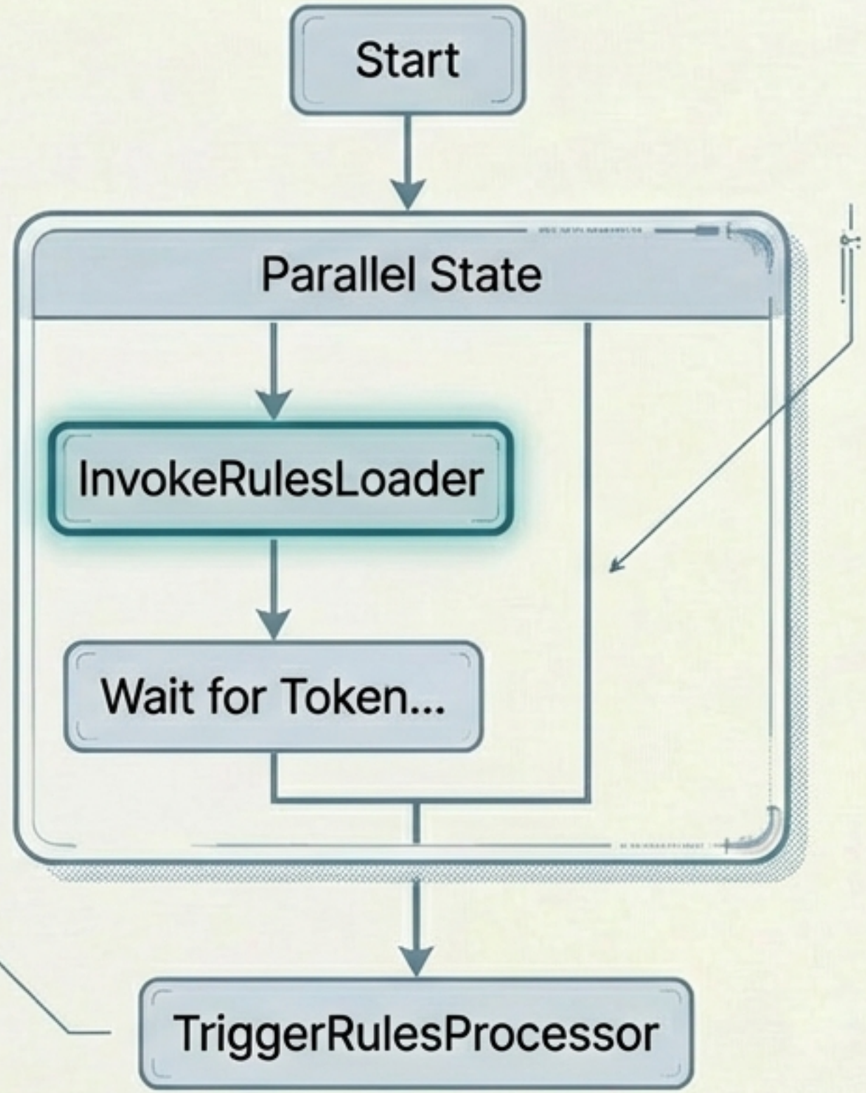
This function acts as the arm that reaches across the gap. It is a generic invoker that reuses shared logic to execute shell commands remotely.



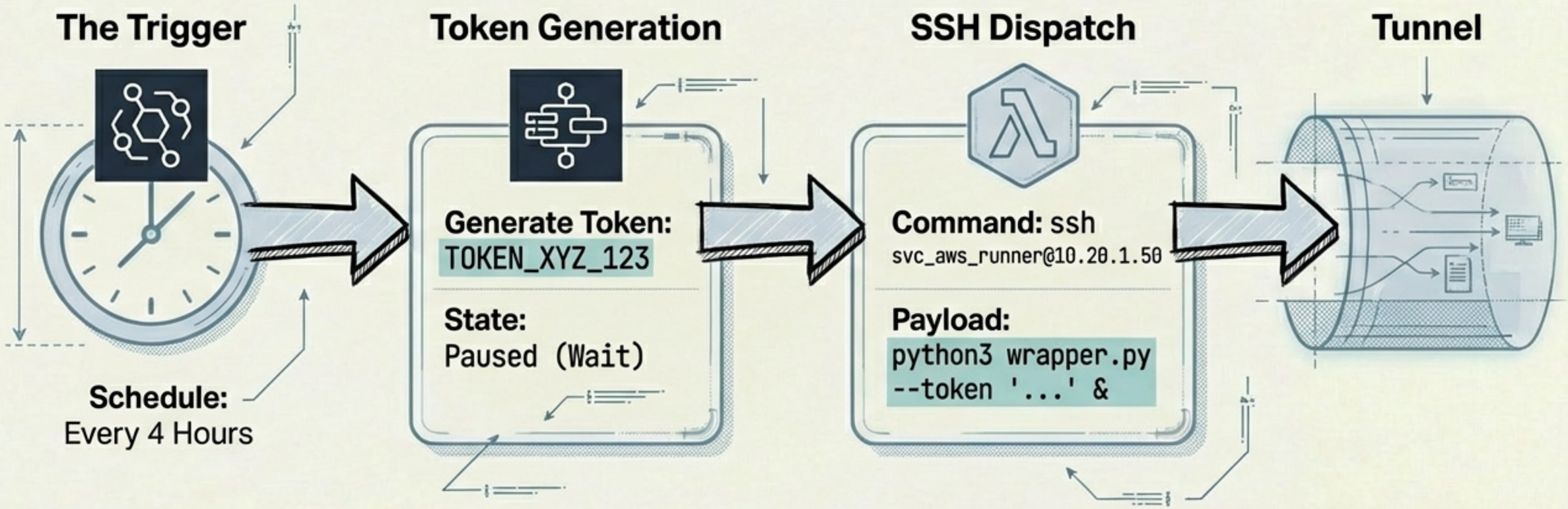

**Security Note:** Private key is never hardcoded. It is retrieved at runtime from AWS Secrets Manager.

# Orchestration Logic: The Step Function

```
"InvokeRulesLoader": {
  "Type": "Task",
  "Resource": "arn:aws:states:::lambda:invoke
    .waitForTaskToken",
  "Parameters": {
    "FunctionName": "ssh_invoker_function",
    "Payload": {
      "token.$": "$$.Task.Token",
      "command": "nohup python3",
      "command": "nohup python3 /opt/scripts/
     rules_wrapper.py --token '$$.Task.Token' &"
    }
  },
  "TimeoutSeconds": 7200
}
```



# Workflow Part I: Trigger & Handshake



# Workflow Part II: Execution & Callback

## Step 3: On-Prem Execution

Wrapper launches Perl Script

Duration: ~30 Minutes (Uploads to S3)

## Step 4: The Callback

Script completes (Exit 0)

```
client.send_task_success(token)
```

## Step 5: Continuation

Workflow Resumes

Next Step: TriggerRulesProcessor

# Failure Scenarios & Recovery Protocols

| Scenario | System Behaviour | Automated Response |
| --- | --- | --- |
| VPN Down | Lambda SSH Exception | Retry 3x, then Alert SNS |
| Script Crash | Perl exits with error code | Wrapper sends TaskFailure -> Pipeline Stop -> Alert SNS |
| Server Hang | No callback received | Step Function Timeout (7200s) -> Alert SNS |
| Bad Token | Auth Failure | Log to local /var/log/rules_loader_wrapper.log |

# Go-Live Prerequisites Checklist

- [ ] **Legacy Script:** Verified path `/opt/legacy/` exists.
- [ ] **Python Env:** Python 3 + `boto3` installed on Oracle server.
- [ ] **SSH Key:** Public key in `authorized_keys`; Private key in AWS Secrets Manager.
- [ ] **Firewall:** AWS Lambda subnet CIDR whitelisted.
- [ ] **IAM:** On-Prem credentials have permission to talk to the specific Step Function ARN.

# Migration Summary

## Confirmed migration path. Automated, monitored, and resilient.

Successfully migrated Rules Loader execution to a modern orchestration layer without destabilising legacy code.

### Latency

Asynchronous processing (~30 mins) handled via Token.

### Security

Fully encrypted SSH tunnel.

### Observability

Full visibility in AWS Console; legacy logs retained locally.