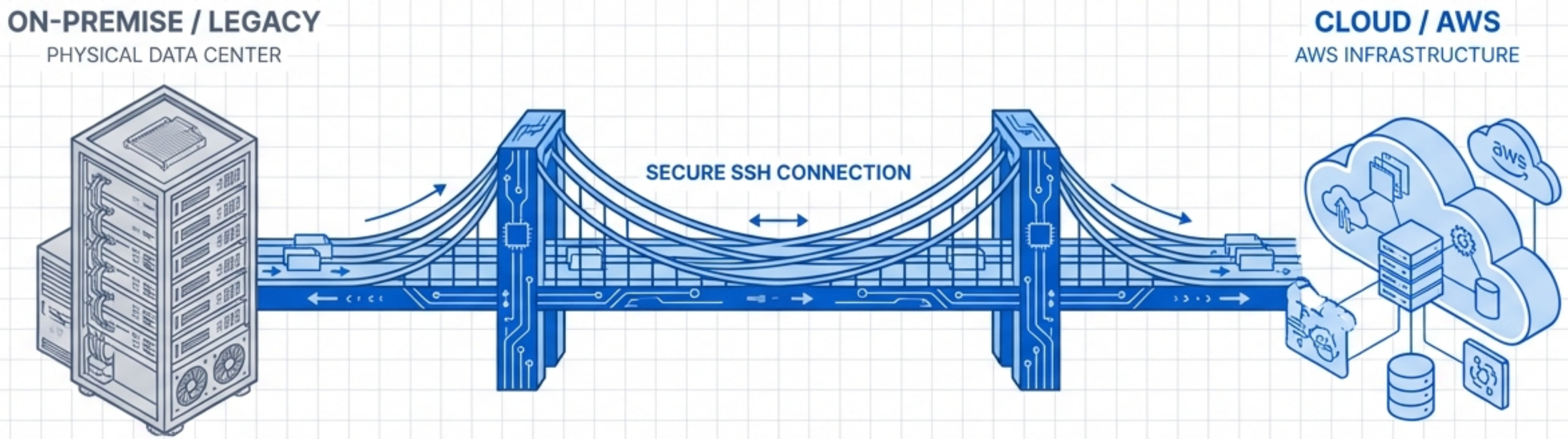
MIGRATION ARCHITECTURE SERIES // VOL. 1

Policy Loader Migration: Hybrid SSH Pattern

Architectural Blueprint and Implementation Guide



MODULE	MIGRATION TYPE	ORCHESTRATION
Policy Loader (Scraper)	Hybrid - SSH Invocation	AWS Step Functions + EventBridge

STRATEGIC OVERVIEW: CENTRALISED CONTROL, LOCAL EXECUTION

THE CHALLENGE (LEGACY)

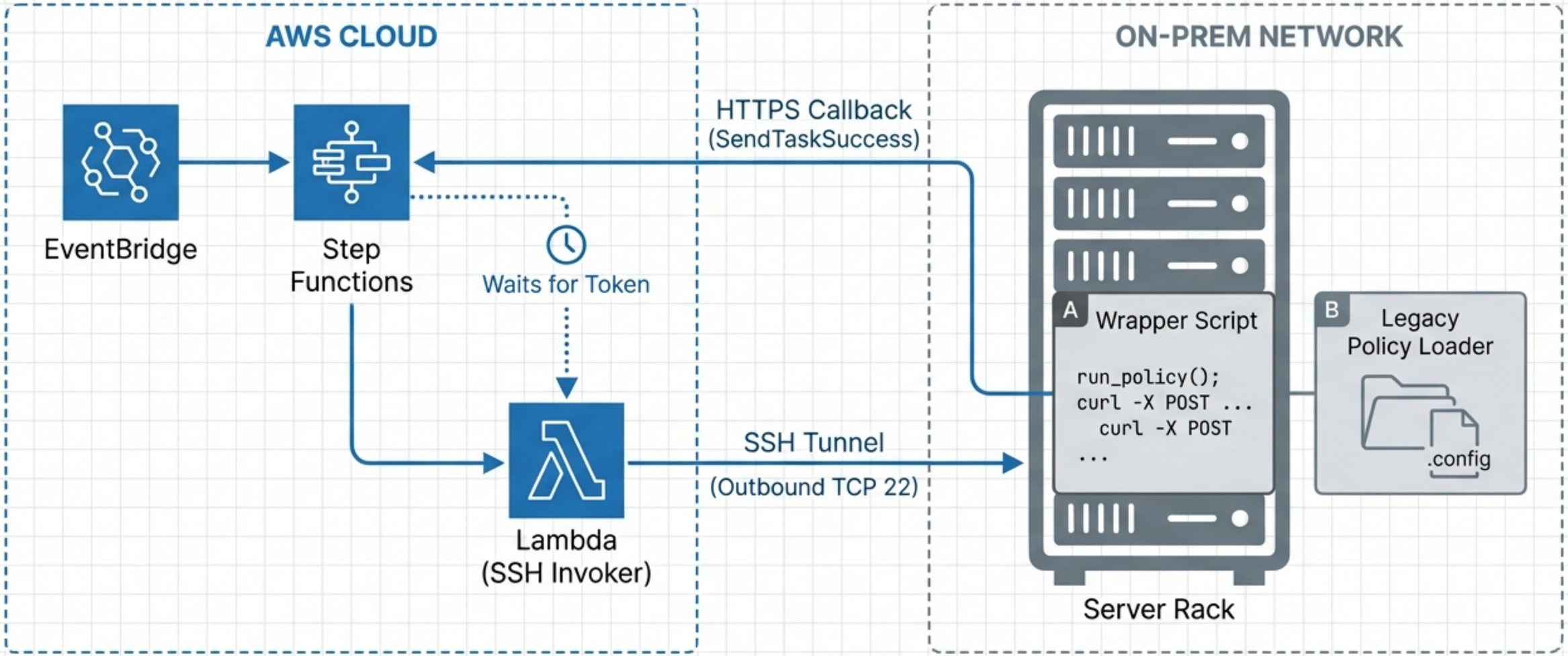
- **Trigger:** Local Cron Job (Isolated)
- **Visibility:** Logs trapped on server
- **IP Reputation:** Trusted On-Prem IP required
- **Risk:** No centralised error handling

THE STRATEGY (TARGET)

- **Code:** Remains ‘As-Is’ on-premise
- **Trigger:** AWS EventBridge (Reliable Scheduling)
- **Orchestration:** AWS Step Functions (Remote Control)
- **Monitoring:** Task Token Callback Pattern

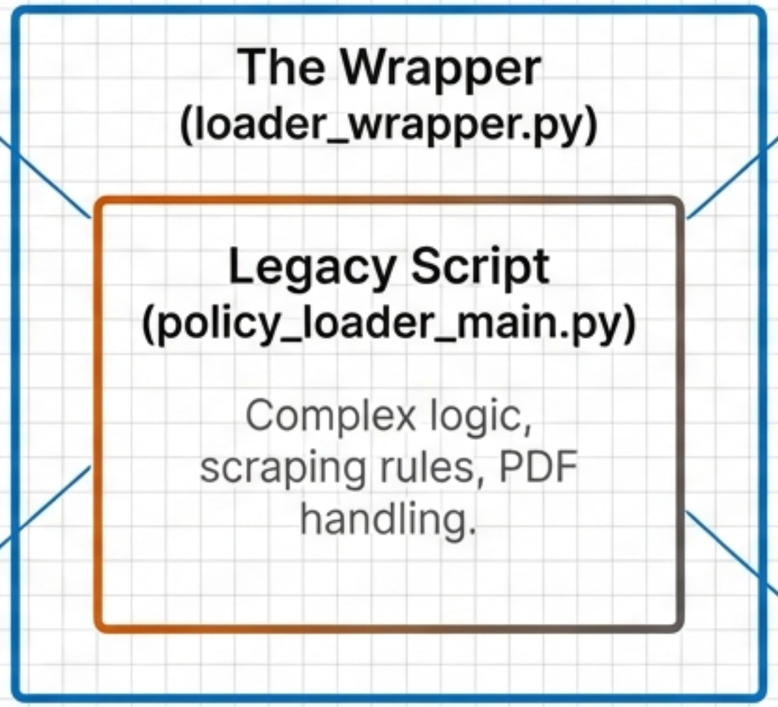
```
return token;
```

HIGH-LEVEL ARCHITECTURE



The Hybrid Gap: Step Functions pauses workflow execution, waiting for the On-Prem wrapper to report status via API.

Phase 1: On-Prem Preparation – The Wrapper Pattern



ACCEPT TOKEN

Receives ‘taskToken’ from AWS via command line args.

EXECUTE

Runs the legacy script via subprocess calls.

CALLBACK

Calls AWS API ‘SendTaskSuccess’ or ‘SendTaskFailure’.

CAPTURE

Intercepts Exit Codes (0 or 1) and Stderr output.

Deep Dive: loader_wrapper.py implementation

```python
try:
    # 1. Run Legacy Scraper
    result = subprocess.run(
        ["python3", "policy_loader_main.py"],
        check=True, capture_output=True
    )

    # 2. Success Callback
    client.send_task_success(
        taskToken=args.token,
        output='{"status": "success"}'
    )

except subprocess.CalledProcessError as e:
    # 3. Failure Callback
    client.send_task_failure(
        taskToken=args.token,
        error="ScraperError",
        cause=str(e.stderr)[:250]
    )
```

Subprocess Execution

Executes legacy code locally. check=True ensures we catch non-zero exit codes immediately.

The Happy Path

If exit code is 0, we return a success status to AWS.

Error Handling

Catches CalledProcessError. Truncates error messages to 250 chars to fit API limits.

On-Prem Security & Identity Management

Authorizing the server to speak to the cloud

Option A: Standard IAM User

Create IAM User: `"svc_policy_loader'`

Config: stored in `~/.aws/credentials`

Pros: Simple setup.

Cons: Long-lived static credentials require rotation.

Option B: IAM Roles Anywhere (Recommended)

Authentication: `x.509` **Certificates** + Trust Anchor

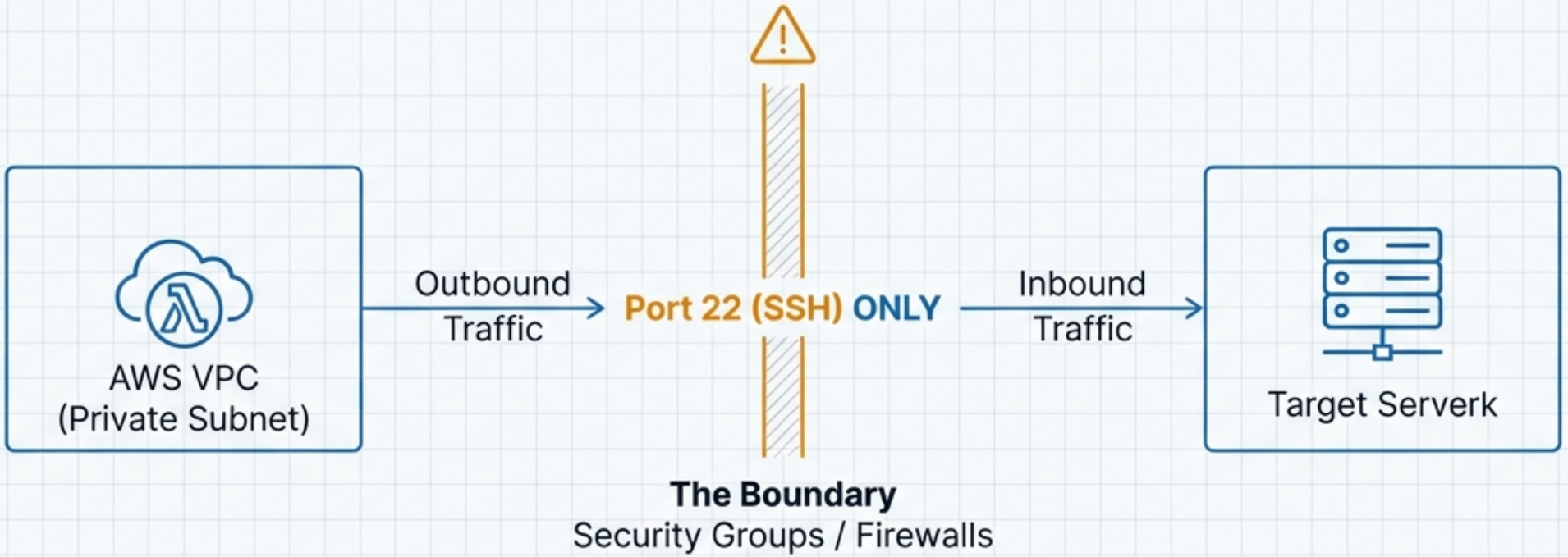
Config: `AWS IAM Roles Anywhere agent`

Pros: Temporary credentials, no static keys.

Security: Enterprise-grade standard.

Policy Scope Requirement: strictly scoped to `'states:SendTaskSuccess'` and `'states:SendTaskFailure'`.

Phase 2: AWS Controller Setup – Network Plumbing



Component	Setting	Value
VPC Connectivity	Link Type	VPN or Direct Connect
Lambda Security Group	Outbound Rule	`Protocol: TCP, Port: 22, Dest: On-Prem IP`
On-Prem Firewall	Inbound Rule	`Protocol: TCP, Port: 22, Source: Lambda CIDR`

Deep Dive: The SSH Invoker Lambda

ssh_invoker.py implementation

```python
# 1. Establish Secure Connection
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh.connect(host, username=user, pkey=pkey)

# 2. Construct Async Command
cmd = f"nohup python3 /opt/scripts/loader_wrapper.py \
      --token '{token}' > /tmp/loader.log 2>&1 &"

# 3. Execute and Disconnect
stdin, stdout, stderr = ssh.exec_command(cmd)
```

Paramiko Connection

Uses Python Paramiko library. Retrieves SSH Private Key securely from AWS Secrets Manager at runtime.

The Injection Command

Injects the unique Step Functions ‘token’ into the wrapper arguments.

Background Execution (The Magic)

Uses ‘nohup’ and ‘&’ to run the process in the background. This allows the Lambda to disconnect immediately (millisecs) while the job runs for hours.

Modern Architectural Blueprint

Phase 3: Orchestration Logic – The "Wait" Pattern

Start
(Point 0)

Step Functions generates Token.

Action
(Point 1s)

Lambda delivers token via SSH.

The Gap

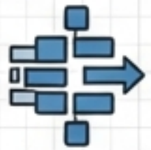
Step Functions Pauses (Zero Cost)

Can wait up to 1 Year.

Callback
(Point 45 mins)

On-Prem Wrapper calls API with Token.

Resume
(Point 45m 1s)



Step Functions resumes execution.

Why this pattern?

Standard Lambda timeout is 15 minutes.
Our Policy Loader takes 45+ minutes.

By using `.waitForTaskToken`, we decouple the orchestration time from the execution time.

The State Machine Definition (ASL)

```
"InvokePolicyLoader": {
  "Type": "Task",
  "Resource": "arn:aws:states:::lambda:invoke.waitForTaskToken",
  "Parameters": {
    "FunctionName": "ssh_invoker_function",
    "Payload": {
      "token.$": "$$.Task.Token"
    }
  },
  "TimeoutSeconds": 14400,
  "Catch": [ { "ErrorEquals": ["States.Timeout"],
    "Next": "NotifyFailure"
  } ],
  "Next": "SuccessState"
}
```

[A] [B] [C]

Legend

[A] Resource Type: ".waitForTaskToken" suffix tells Step Functions to pause immediately after invoking the Lambda.

[B] Payload Injection: "$$.Task.Token" is the intrinsic function that generates the unique ID.

[C] Safety Timeout: "14400 seconds" (4 hours). Hard limit to prevent 'zombie' jobs if the server crashes silently.

Modern Architectural Blueprint

Operations: Life of a Job – The Handoff (08:00 AM)

Trigger Fired

BLUE (Running)

Generates Token: AAAA-1234

SSH Login to On-Prem

```
python3 loader_wrapper.py --token AAAA-1234 &
```

Command Received. Process Started.

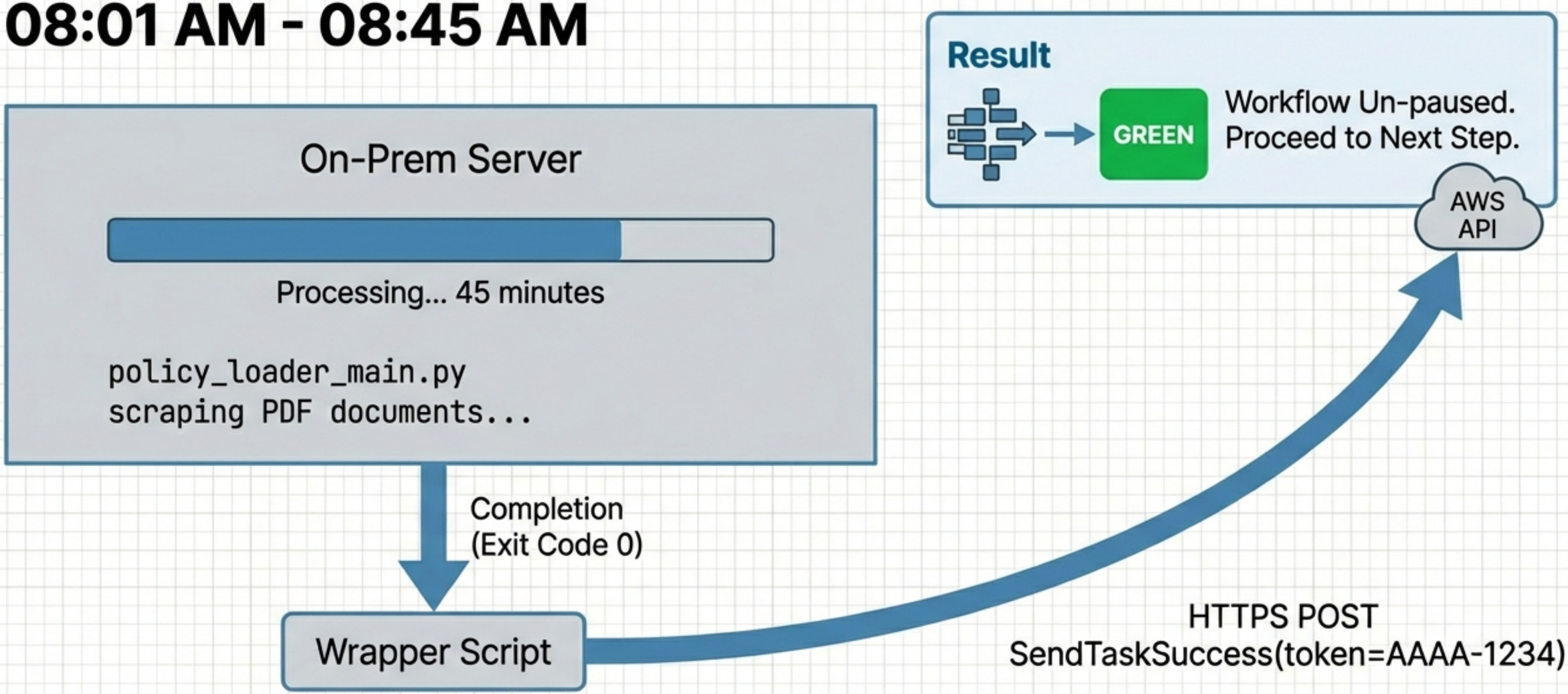
Side Note

Lambda finishes execution here. Step Functions remains in 'Blue' (Running) state.

Modern Architectural Blueprint

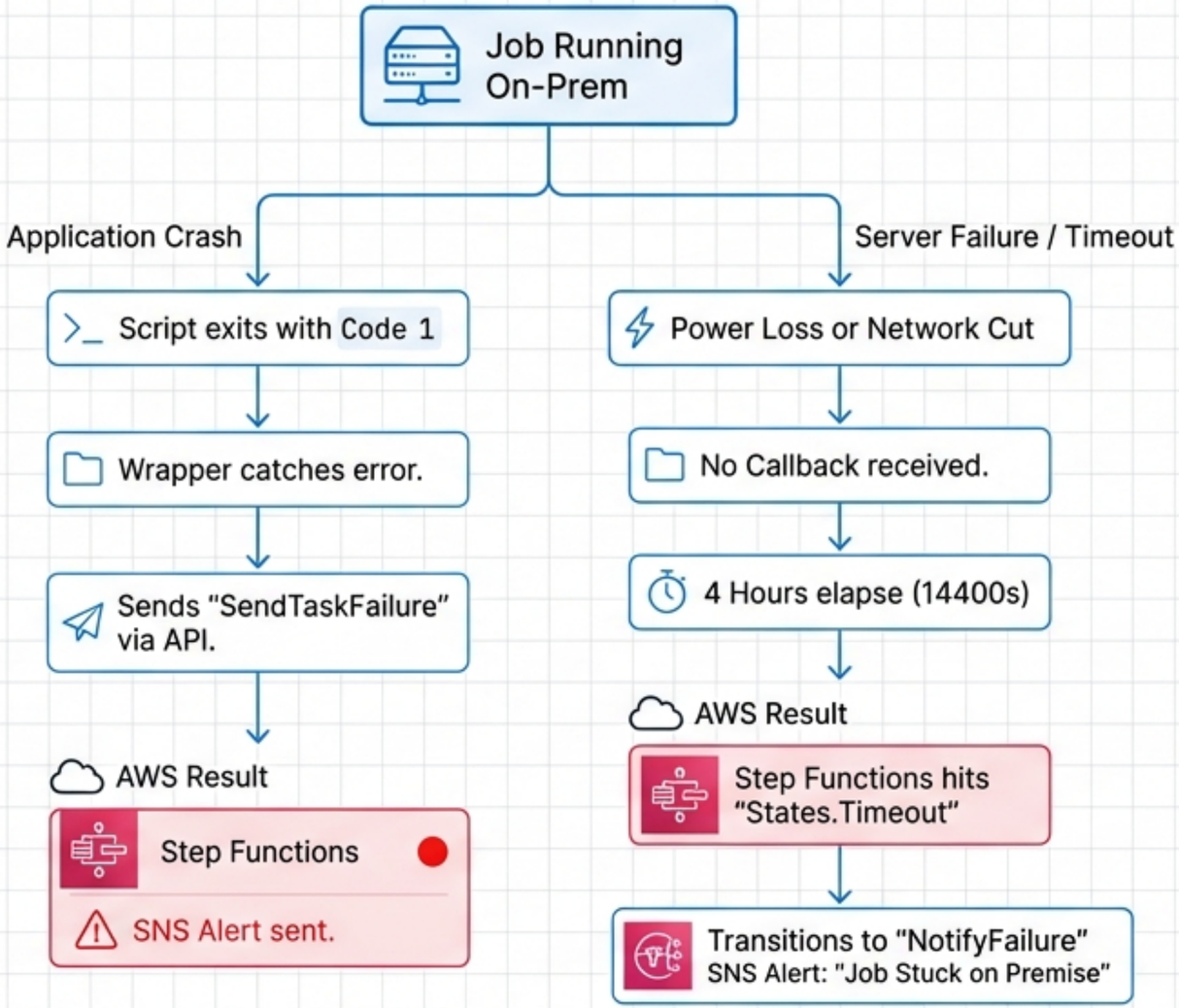
Operations: Life of a Job – Execution & Callback

08:01 AM - 08:45 AM



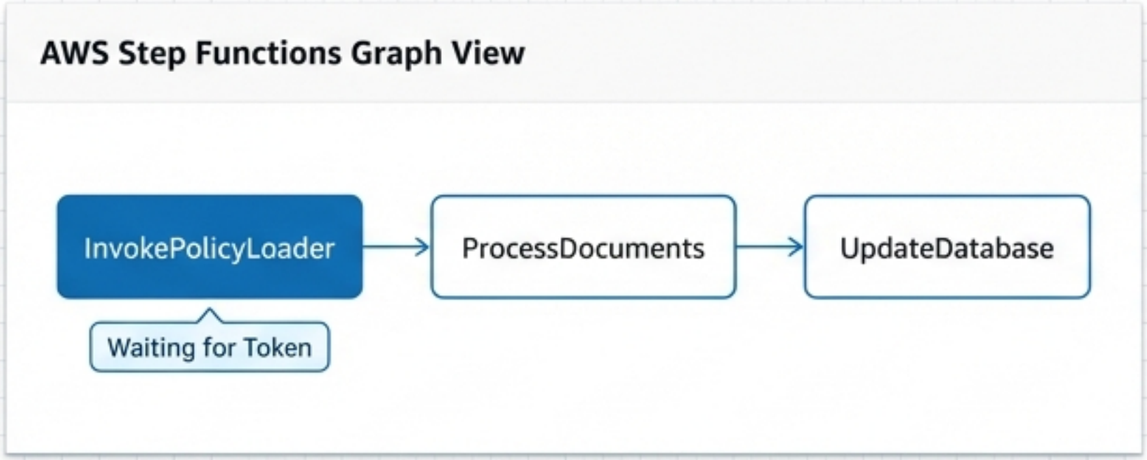
Modern Architectural Blueprint

Operations: Failure Scenarios & Recovery



Operational Visibility & Monitoring

The Console Experience



- In Progress (On-Prem)
- Success
- Failed / Timed Out

Logging Strategy

Connectivity Logs

- CloudWatch Logs (Lambda Group). Tracks SSH connection success/fail.

Application Logs

- Stdout/Stderr captured by Wrapper.
- Stored locally: `/tmp/loader.log`
- Optional: Pushed via CloudWatch Agent.

Deployment Prerequisites Checklist

NETWORK

- [ ] VPN / Direct Connect verified active.
- [ ] Lambda Security Group allows Outbound TCP 22 to On-Prem IP.
- [ ] On-Prem Firewall allows Inbound TCP 22.

SECURITY

- [ ] SSH Private Key stored in AWS Secrets Manager.
- [ ] Public Key added to `~/.ssh/authorized_keys` on target host.
- [ ] IAM Role created with '`states:SendTaskSuccess`' permissions.

CODE

- [ ] `loader_wrapper.py` deployed to `/opt/scripts/` directory.
- [ ] Python 3 environment verified on host.
- [ ] AWS CLI or Boto3 installed on host.