AWS re:Invent

DECEMBER 2 – 6, 2024 | LAS VEGAS, NV

SVS218-NEW

Accelerate Python Lambda functions with SnapStart

Samrat Karak
(he/him)
Principal Product Manager,
AWS Lambda
AWS

Leandro Damascena
(he/him)
Sr. Engineer SA,
Powertools for AWS Lambda
AWS

Initialization time

Java +

AWS Lambda

Initialization time

.NET

+

AWS Lambda

aws

Initialization time

Python +

AWS Lambda

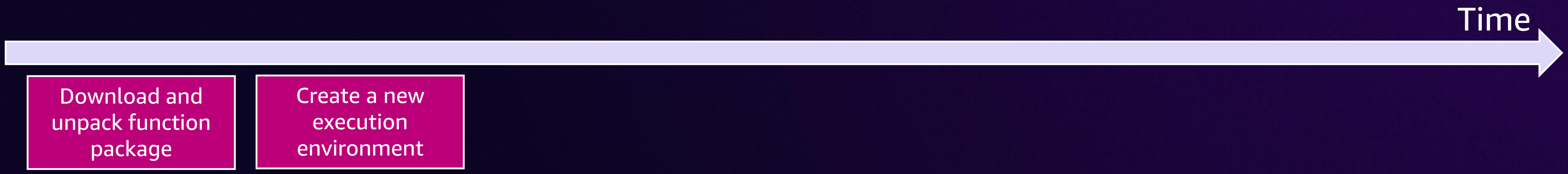
Agenda

On-demand invocation model

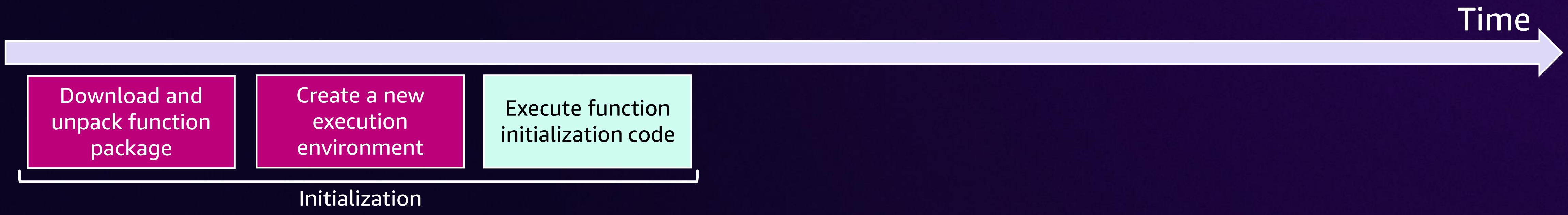
On-demand invocation model

Time
Download and unpack function package

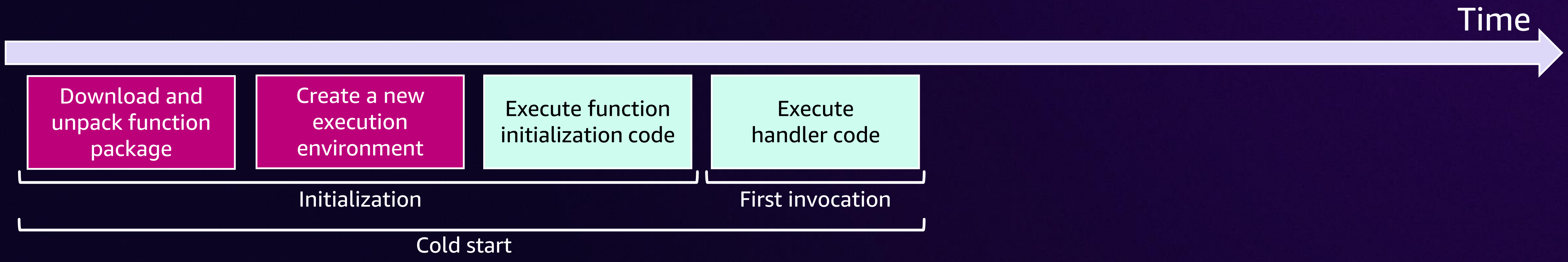
On-demand invocation model


Time
Download and unpack function package
Create a new execution environment

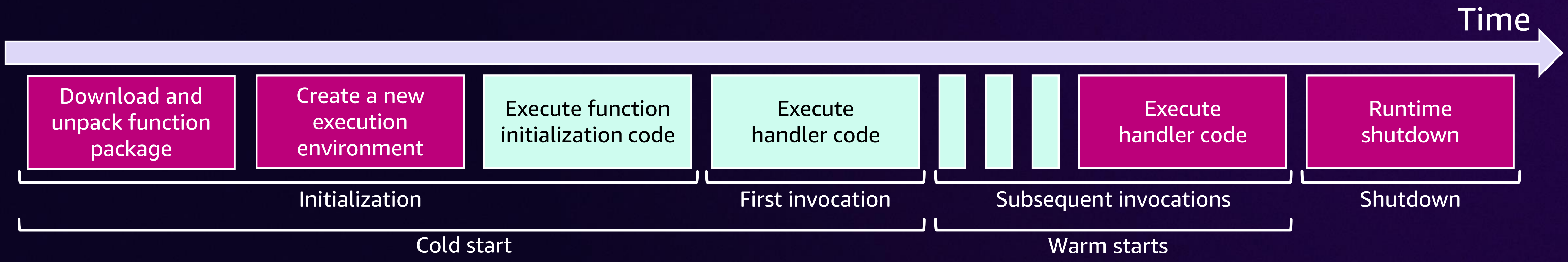
On-demand invocation model


Time
Download and unpack function package
Create a new execution environment
Execute function initialization code
Initialization

On-demand invocation model


Time
Download and unpack function package
Create a new execution environment
Execute function initialization code
Execute handler code
Initialization
First invocation
Cold start

On-demand invocation model



NEW

AWS Lambda SnapStart for Python and .NET

Delivers faster startup performance as low as sub-second

AWS Lambda SnapStart

Benefit

Delivers **faster startup performance**, from several seconds to as low as sub-second, with minimal or no changes to your function code

AWS Lambda SnapStart

Benefit

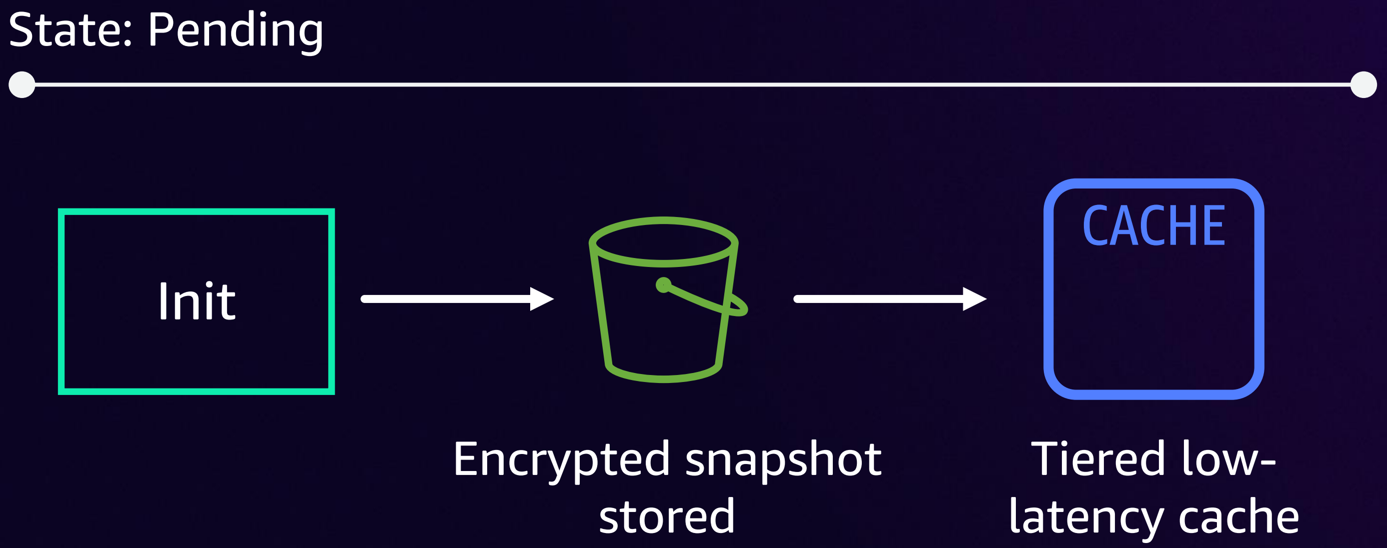
Delivers **faster startup performance**, from several seconds to as low as sub-second, with minimal or no changes to your function code

Supported on

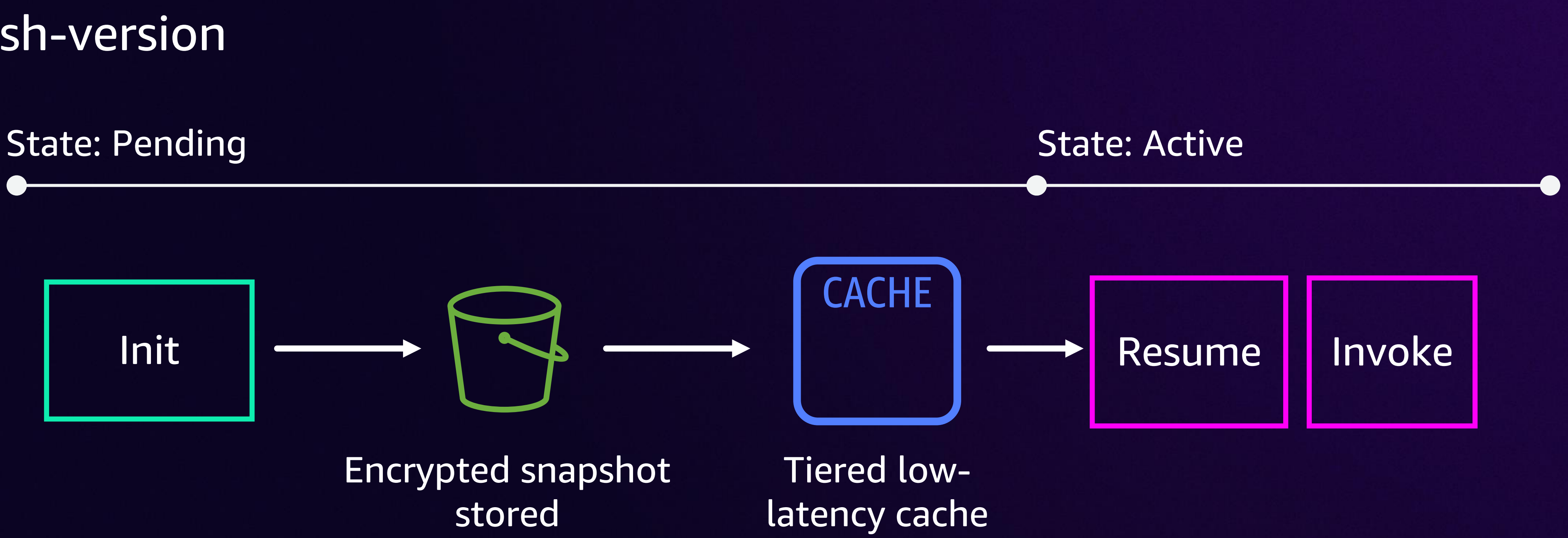
- Python runtime versions 3.12 and later
- .NET 8 and later
- Java 11 and later

SnapStart overview

publish-version

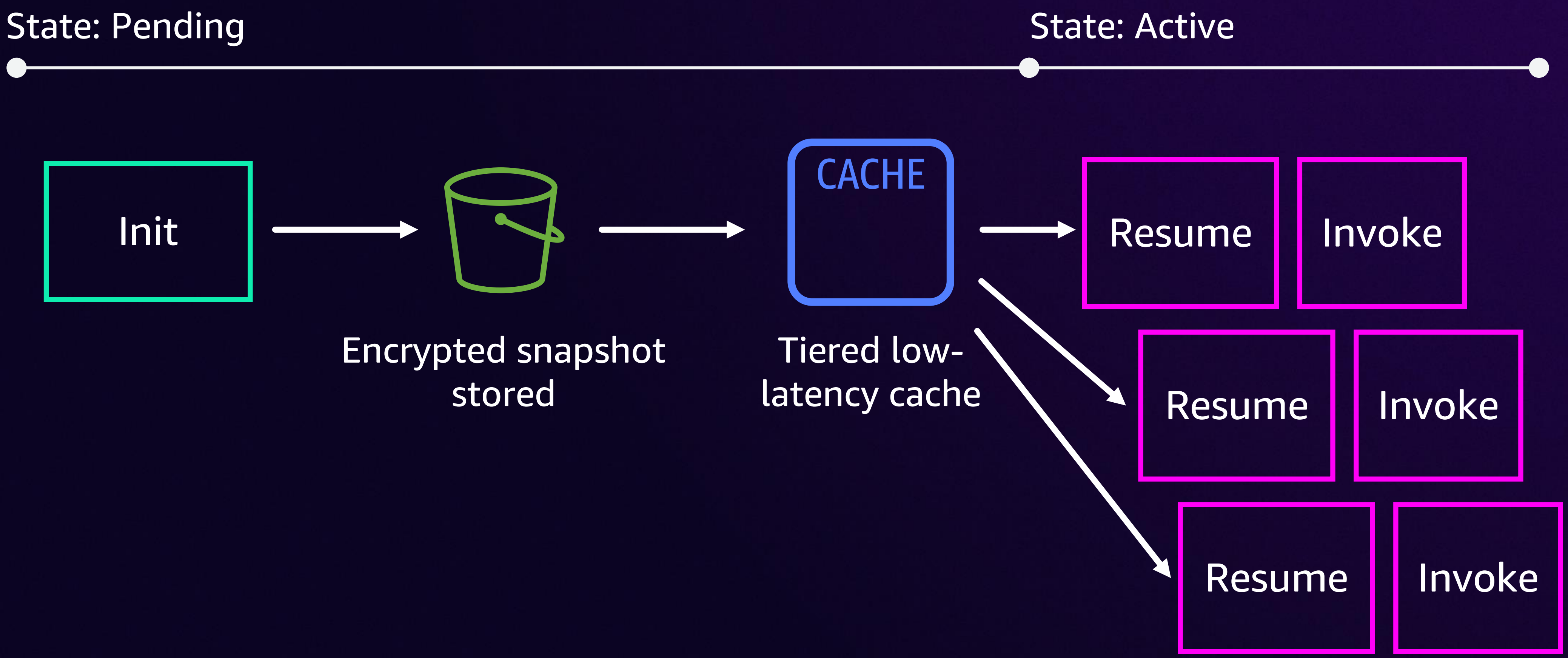


SnapStart overview

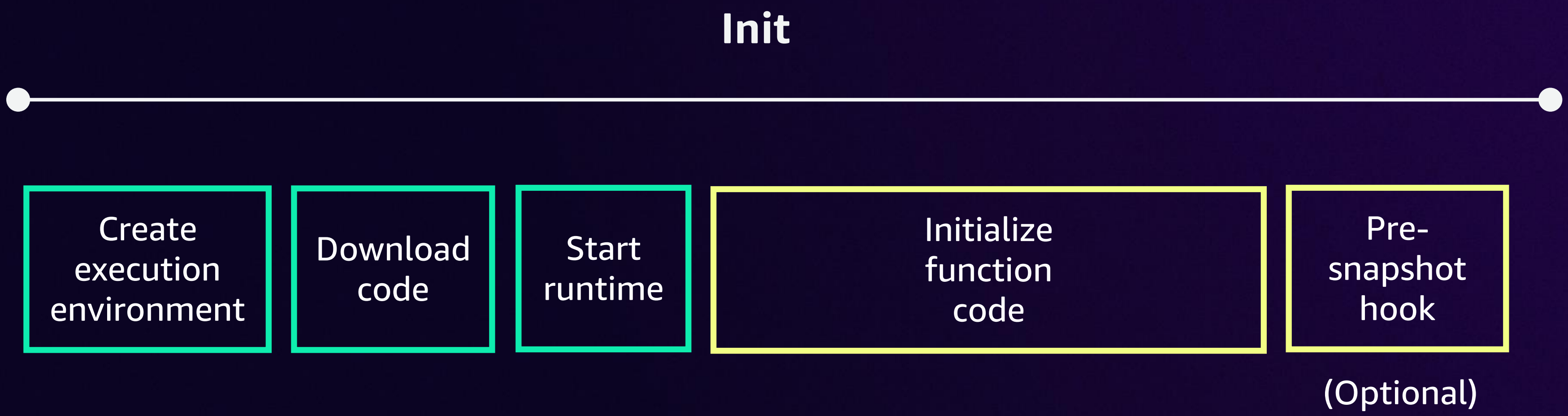


SnapStart overview

publish-version



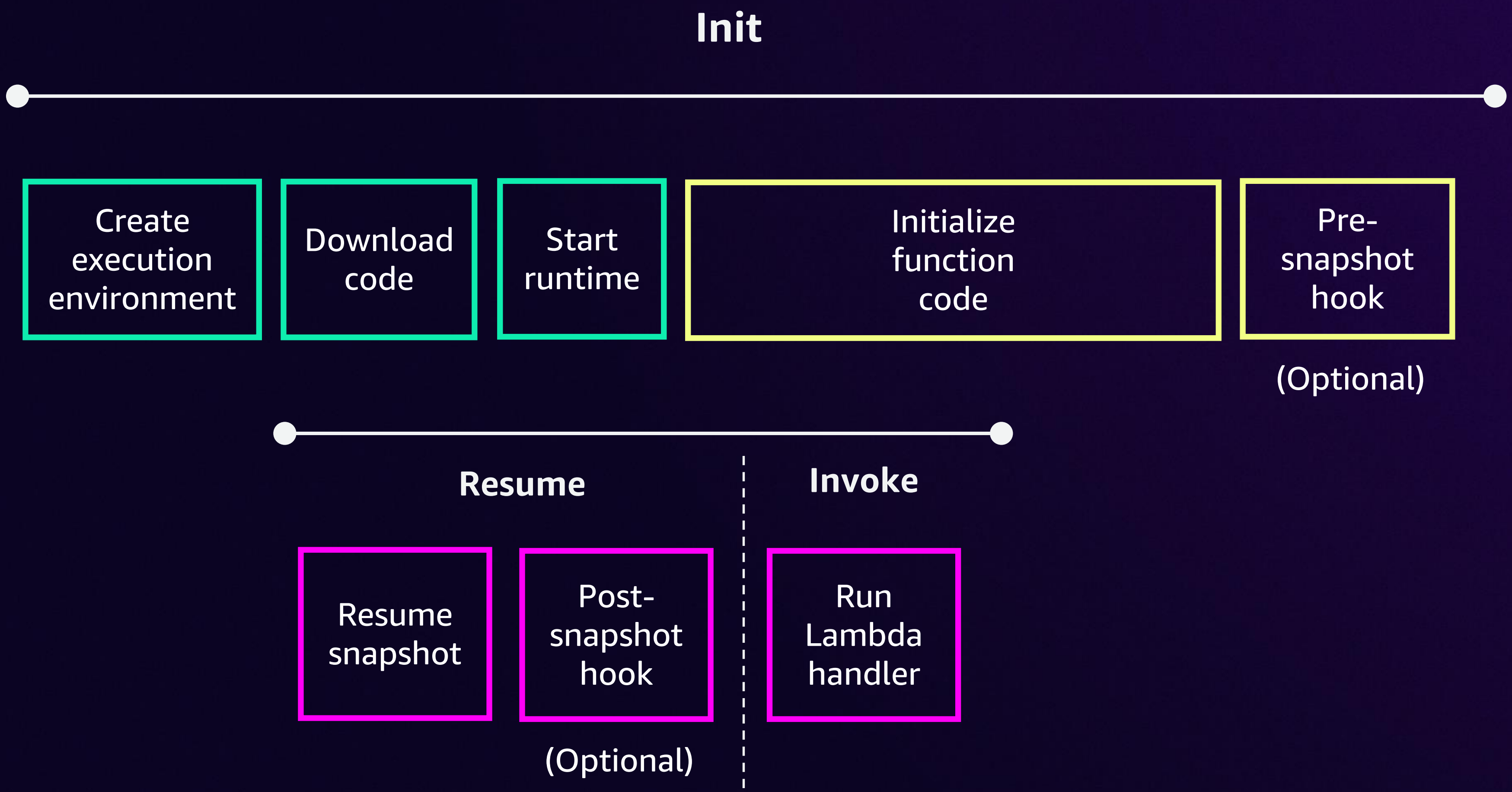
Invocation model



Firecracker

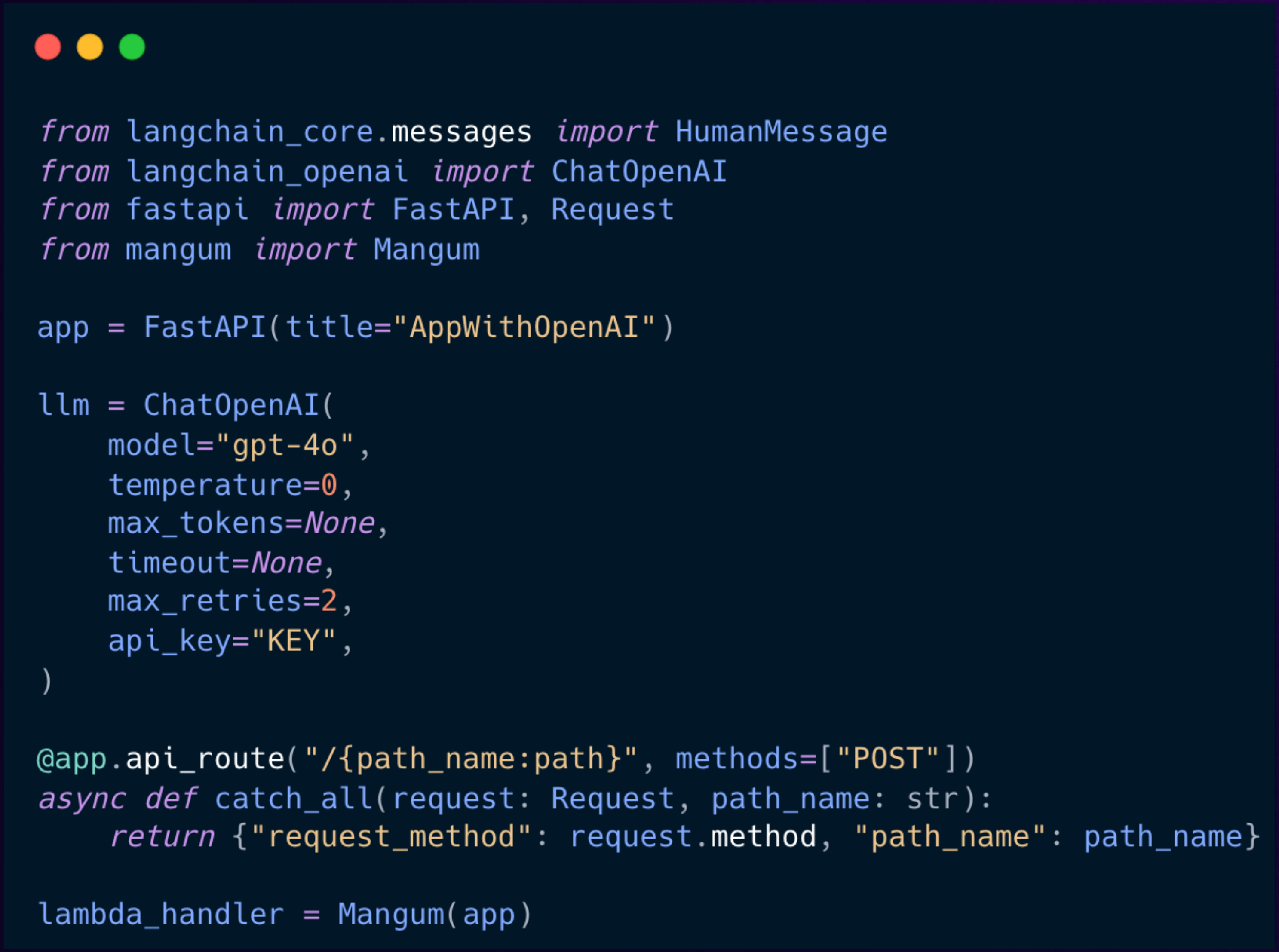
microVM snapshot technology

Invocation model



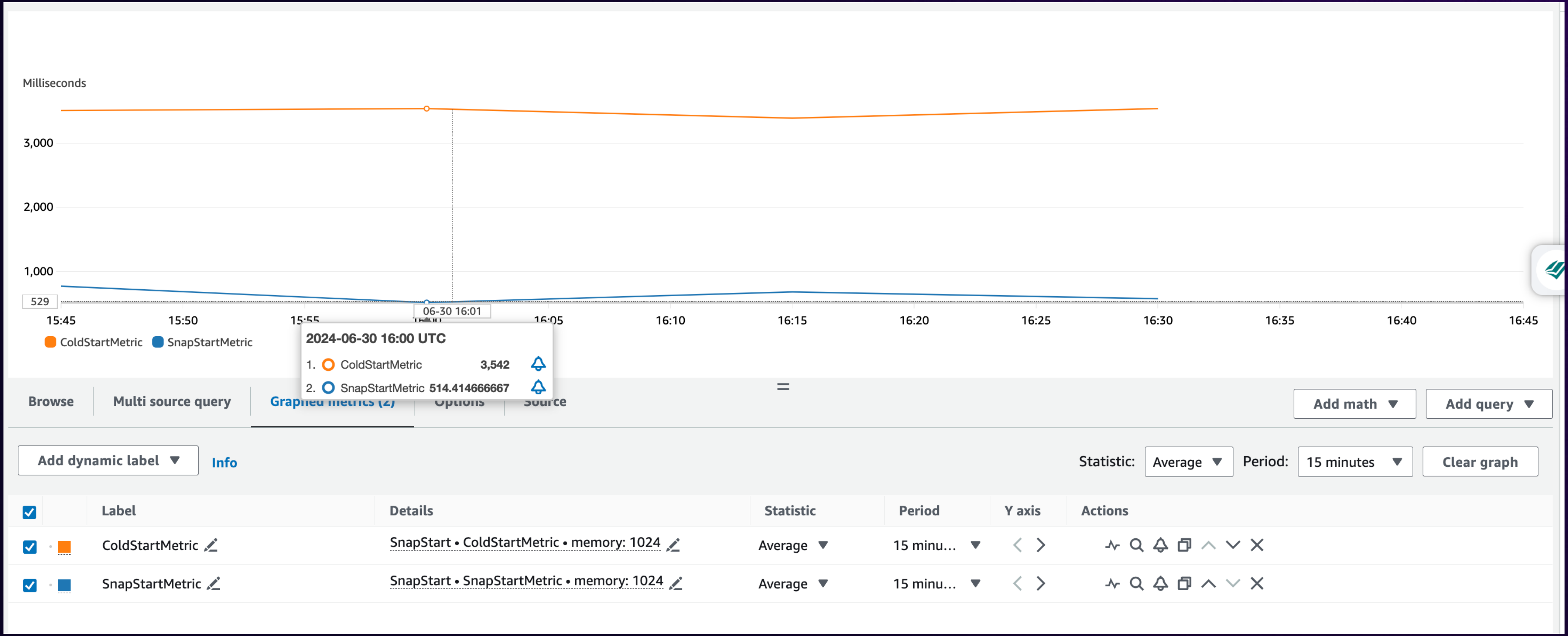
Use cases

Chatbot with gen AI



```python
from langchain_core.messages import HumanMessage
from langchain_openai import ChatOpenAI
from fastapi import FastAPI, Request
from mangum import Mangum

app = FastAPI(title="AppWithOpenAI")

llm = ChatOpenAI(
    model="gpt-4o",
    temperature=0,
    max_tokens=None,
    timeout=None,
    max_retries=2,
    api_key="KEY",
)

@app.api_route("/{path_name:path}", methods=["POST"])
async def catch_all(request: Request, path_name: str):
    return {"request_method": request.method, "path_name": path_name}

lambda_handler = Mangum(app)
```

Chatbot with gen AI – Results



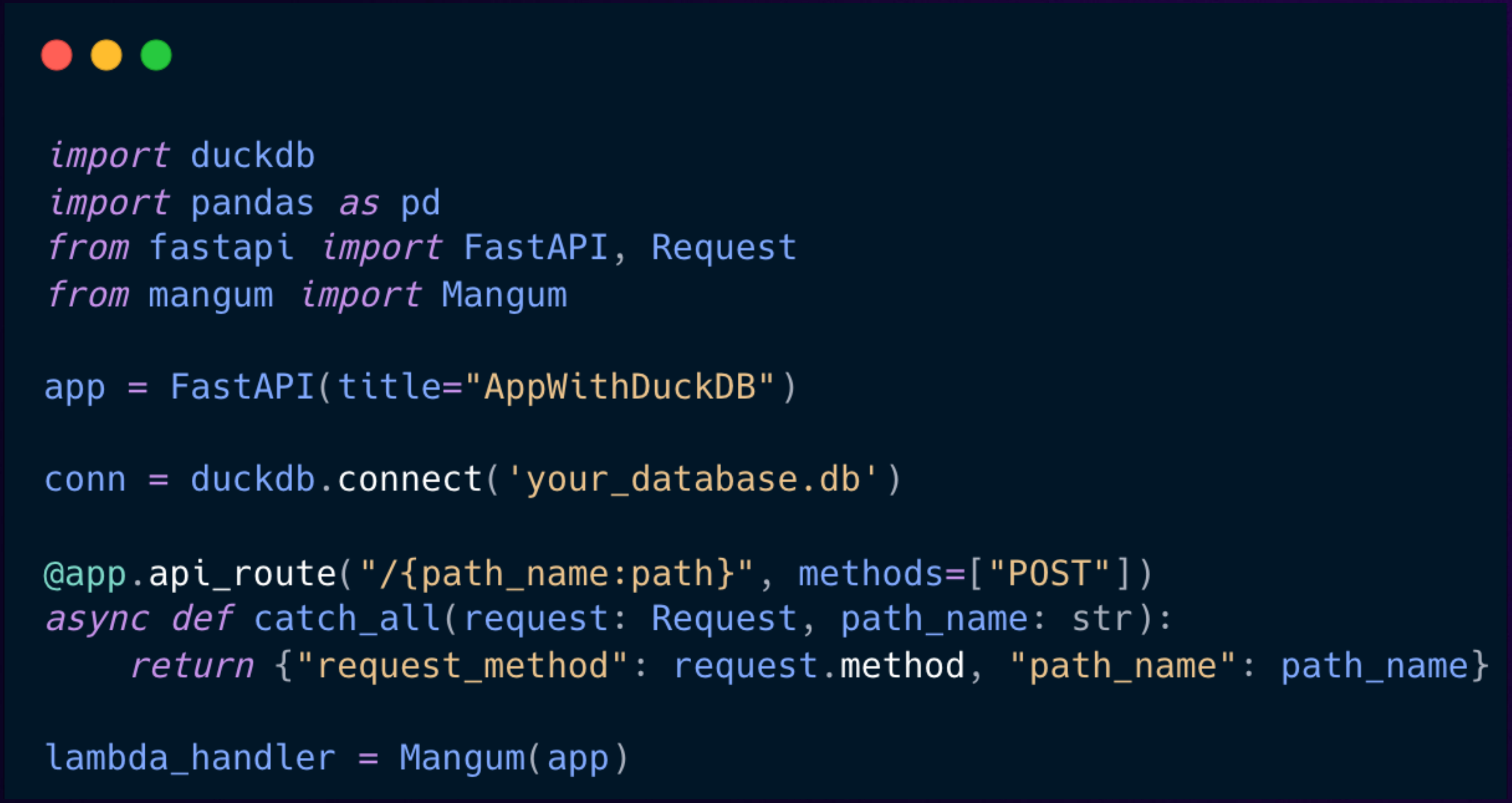
2024-06-30 16:00 UTC

1. ColdStartMetric 3,542
2. SnapStartMetric 514.414666667

Statistic: Average Period: 15 minutes

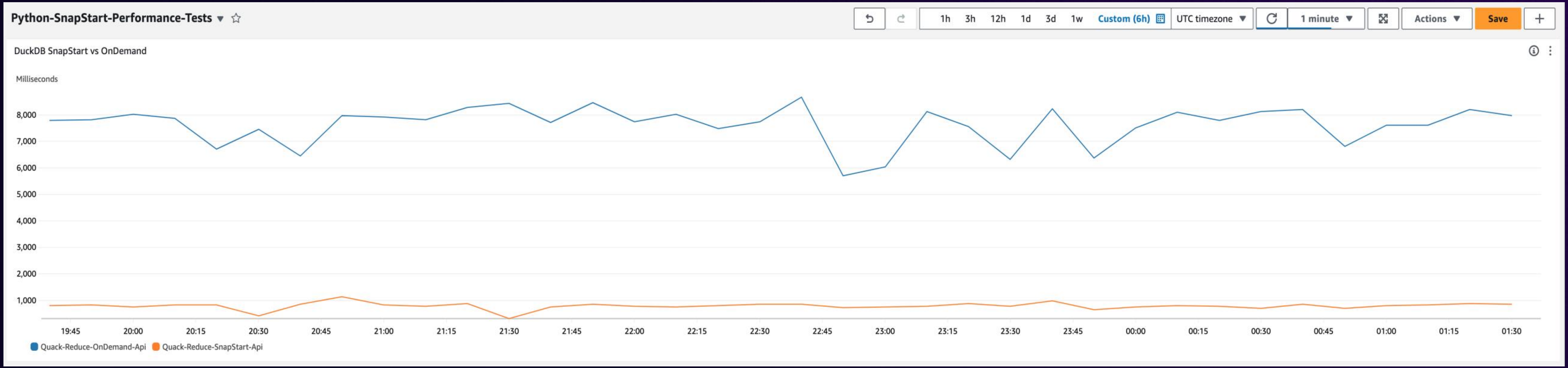
✓		Label	Details	Statistic	Period	Y axis	Actions
✓	🟧	ColdStartMetric	SnapStart • ColdStartMetric • memory: 1024	Average	15 minu…		
✓	🟦	SnapStartMetric	SnapStart • SnapStartMetric • memory: 1024	Average	15 minu…		

Data analysis



```python
import duckdb
import pandas as pd
from fastapi import FastAPI, Request
from mangum import Mangum

app = FastAPI(title="AppWithDuckDB")

conn = duckdb.connect('your_database.db')

@app.api_route("/{path_name:path}", methods=["POST"])
async def catch_all(request: Request, path_name: str):
    return {"request_method": request.method, "path_name": path_name}

lambda_handler = Mangum(app)
```

Data analysis – Results

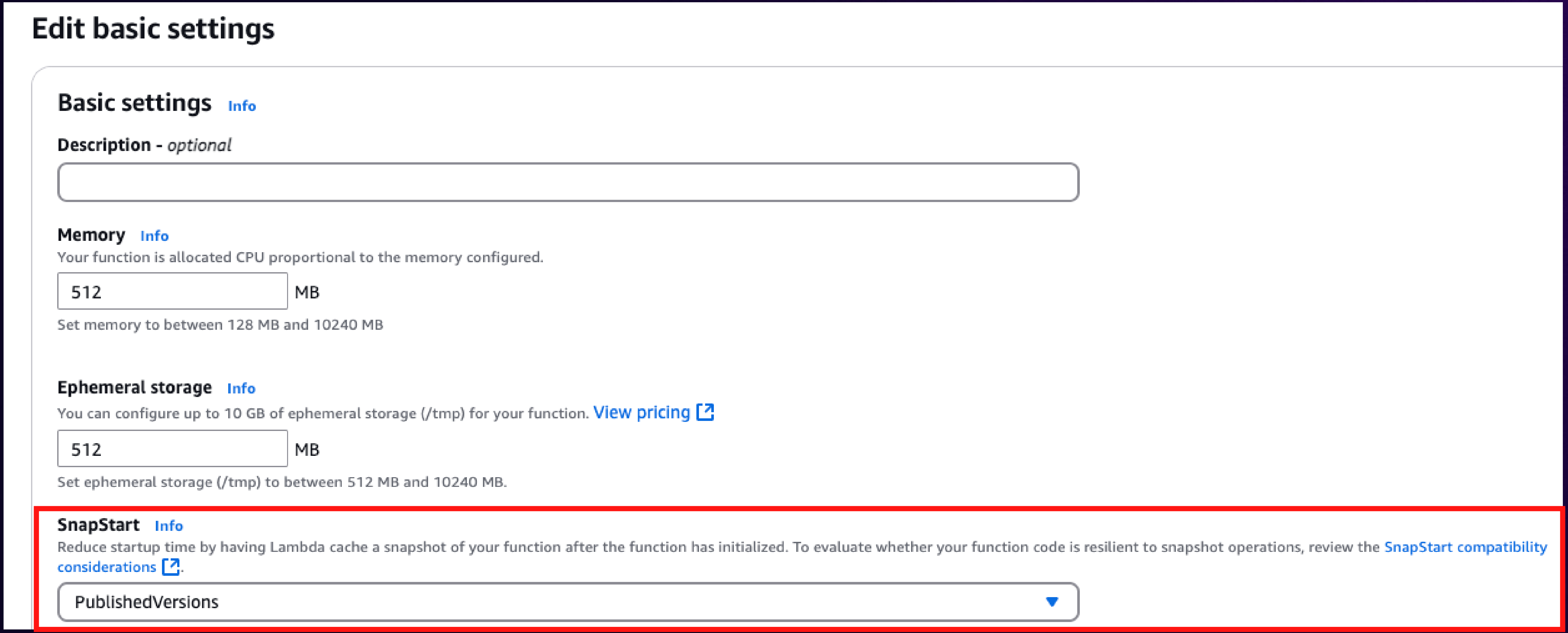


Configuring SnapStart

Enabling SnapStart

- You can enable SnapStart for new or existing functions
- SnapStart only works with published versions or alias of your Lambda function
- You can use Lambda console, AWS CDK, AWS SAM, AWS SDK, AWS CLI, or other IaC tools to enable SnapStart

Configuring SnapStart – AWS Management Console



Edit basic settings

Basic settings Info

Description - *optional*

Memory Info

Your function is allocated CPU proportional to the memory configured.

512 MB

Set memory to between 128 MB and 10240 MB

Ephemeral storage Info

You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing

512 MB

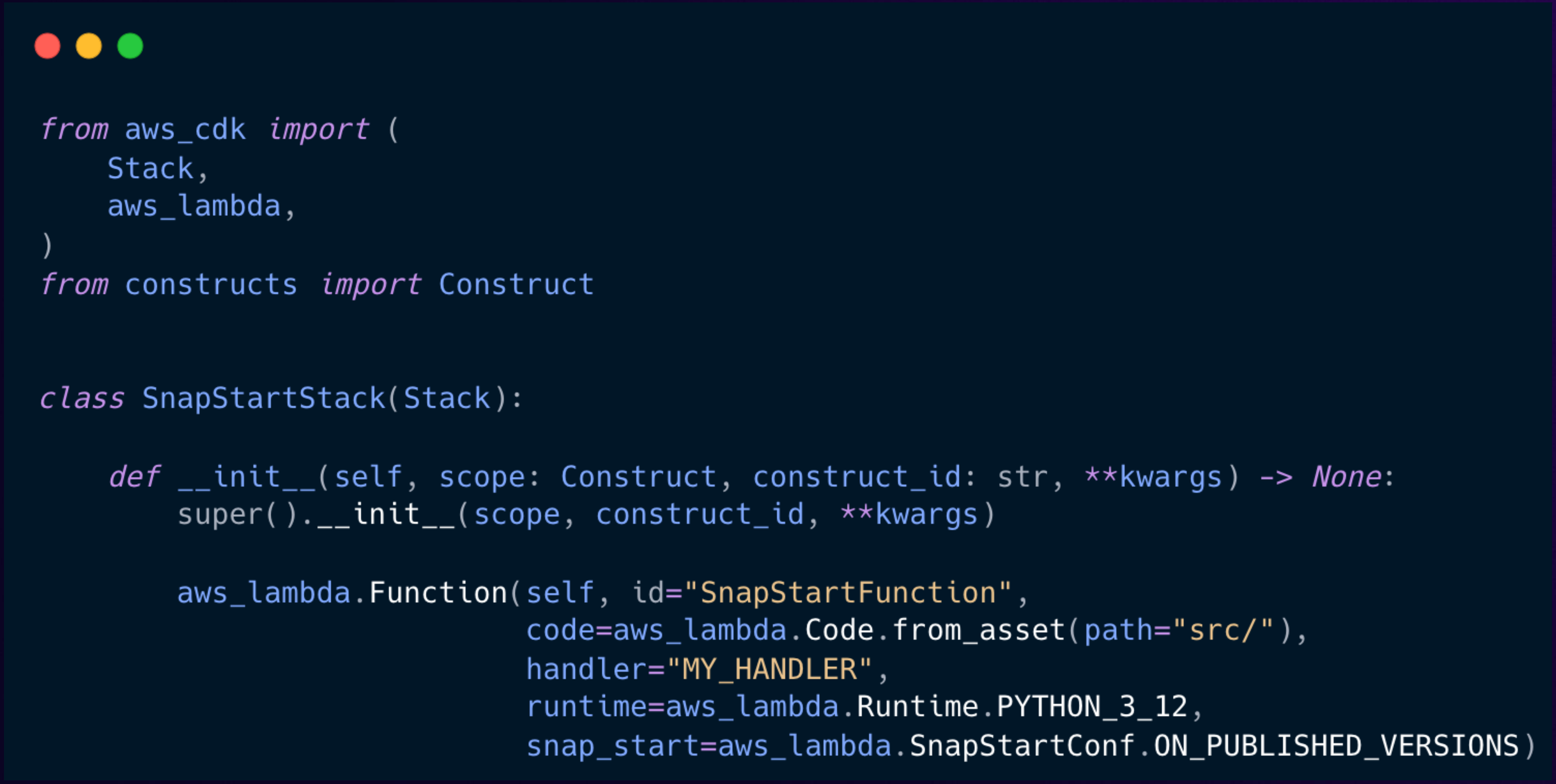
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart Info

Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations.

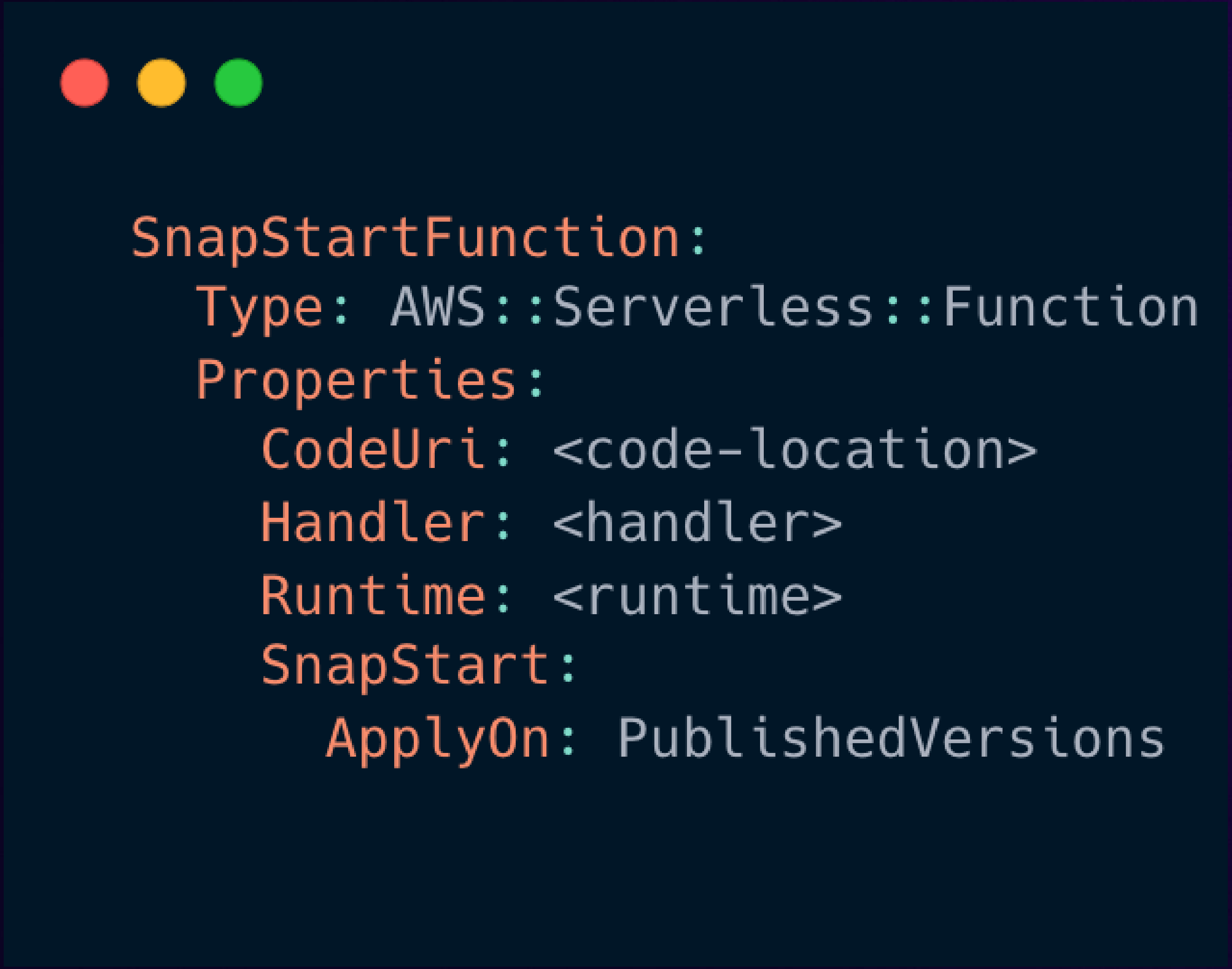
PublishedVersions

Configuring SnapStart – AWS CDK



```python
from aws_cdk import (
    Stack,
    aws_lambda,
)
from constructs import Construct


class SnapStartStack(Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        aws_lambda.Function(self, id="SnapStartFunction",
                            code=aws_lambda.Code.from_asset(path="src/"),
                            handler="MY_HANDLER",
                            runtime=aws_lambda.Runtime.PYTHON_3_12,
                            snap_start=aws_lambda.SnapStartConf.ON_PUBLISHED_VERSIONS)
```

Configuring SnapStart – AWS SAM



```
SnapStartFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: <code-location>
    Handler: <handler>
    Runtime: <runtime>
    SnapStart:
      ApplyOn: PublishedVersions
```

SnapStart runtime hooks for Python and .NET

Implement code before or after snapshots

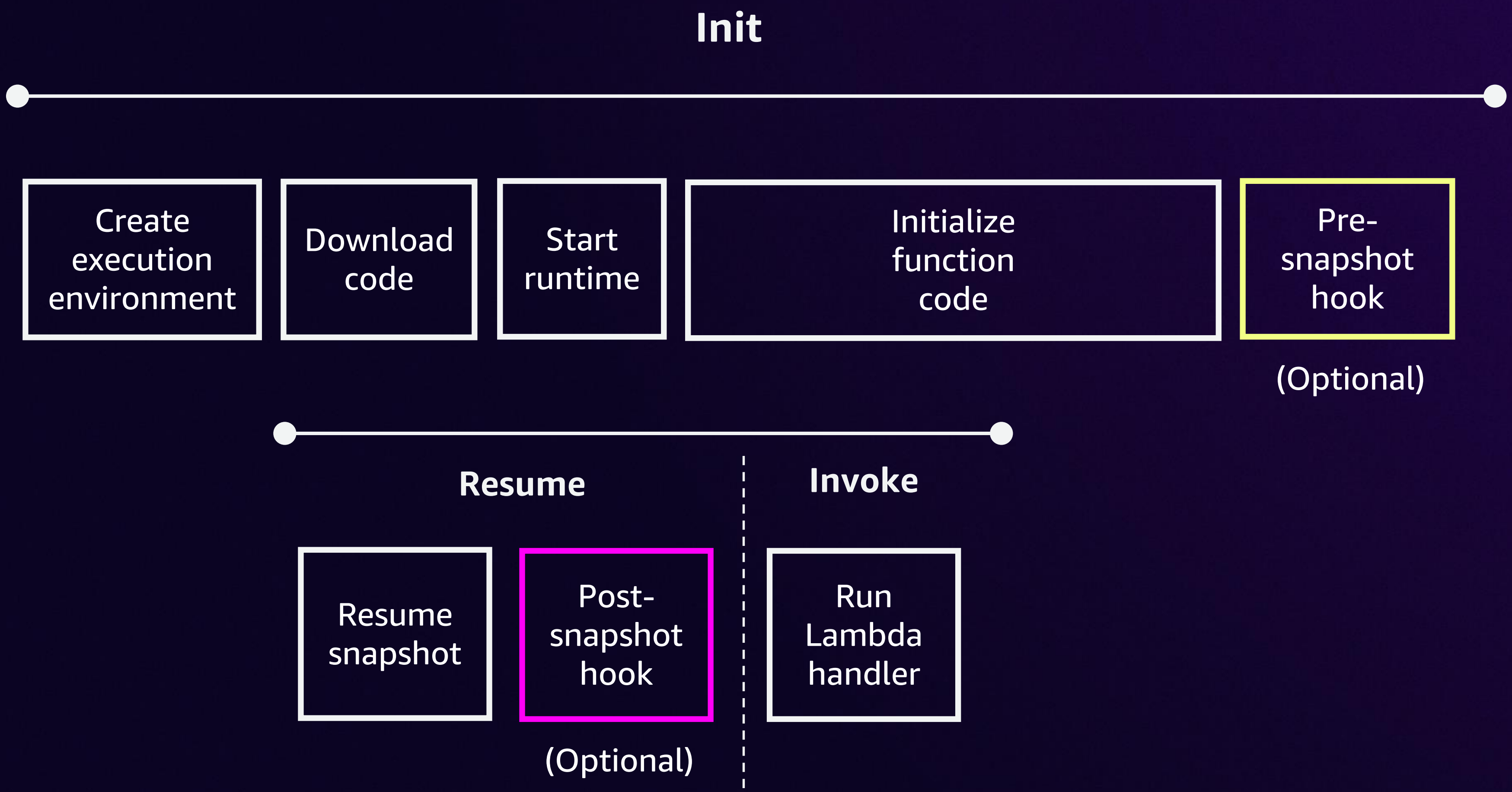
Run code before Lambda creates a snapshot

- Download external files
- Computation-intensive tasks

Run code after Lambda resumes a function from a snapshot

- Connecting to a database
- Refresh data
- Retrieve secrets

Hooks detail



Runtime hooks for Python

Runtime hooks are part of the open source Snapshot Restore for Python project

@register_before_snapshot

- Run code before the snapshot creation

@register_after_restore

- Run code after Lambda resumes a function from a snapshot

SnapStart runtime hooks in Python

```python
from py_snapshot_restore import register_before_snapshot, register_after_restore

@register_before_snapshot
def beforeLambdaCheckpoint():
    # your logic here
    pass

@register_after_restore
def after_restore():
    # your logic here
    pass

def lambda_handler(event, context):
    pass
```

SnapStart runtime hooks in Python

```python
from snapshot_restore_py import register_before_snapshot, register_after_restore

def lambda_handler(event, context):
    # lambda handler code
    pass

def fn_before_snapshot():
    # your logic here
    pass

def fn_after_restore():
    # your logic here
    pass

register_before_snapshot(fn_before_snapshot)
register_after_restore(fn_after_restore)
```

Runtime hooks for .NET

Runtime hooks are part of the open source Snapshot Restore for .NET project

RegisterBeforeSnapshot

- Run code before the snapshot creation

RegisterAfterRestore

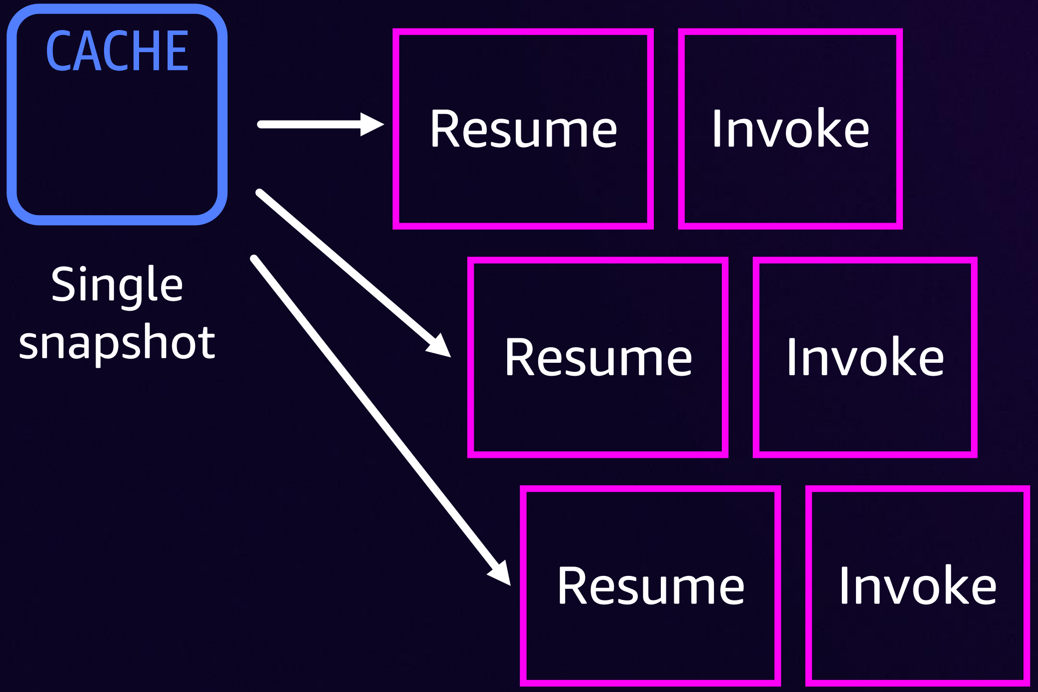
- Run code after Lambda resumes a function from a snapshot

Example for .NET

```csharp
public class SampleClass
{
    public SampleClass()
    {
        Amazon.Lambda.Core.SnapshotRestore.RegisterBeforeSnapshot(BeforeCheckpoint);
        Amazon.Lambda.Core.SnapshotRestore.RegisterAfterRestore(AfterCheckpoint);
    }

    private ValueTask BeforeCheckpoint()
    {
        // Add logic to be executed before taking the snapshot
        return ValueTask.CompletedTask;
    }

    private ValueTask AfterCheckpoint()
    {
        // Add logic to be executed after restoring the snapshot
        return ValueTask.CompletedTask;
    }
```

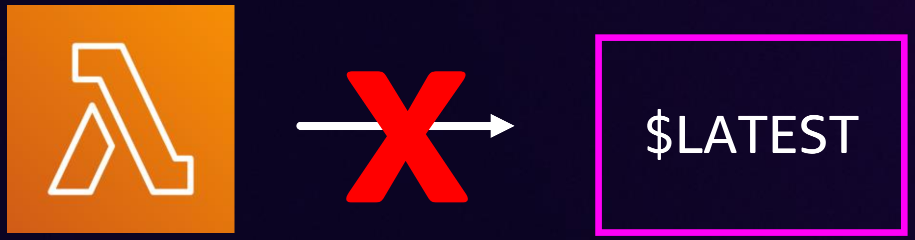
Considerations

Uniqueness



Ensure to generate unique content after initialization

Version and alias



Ensure to publish a version of your function

DNS cache in Python and .NET

- DNS cache in Python
  - Requests library and urllib3 don’t cache DNS

- DNS cache in .NET
  - Automatically reestablish socket connection after restore

SnapStart vs. provisioned concurrency

SnapStart helps reduce cold start latency for 99% of requests

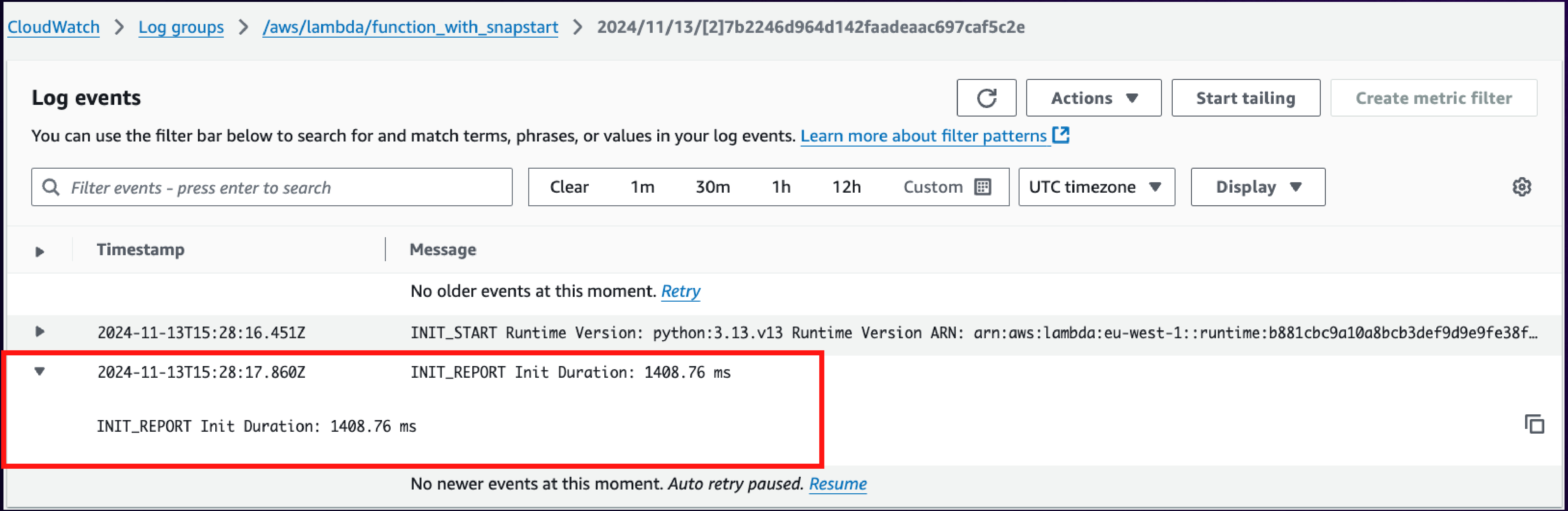
- SnapStart is ideal for workloads with unpredictable traffic spikes

Provisioned concurrency enables warm starts for 99.999% of requests

- Ideal for strict low-latency (double-digit milliseconds) applications

Lambda SnapStart – CloudWatch logs

Initialization logs



CloudWatch > Log groups > /aws/lambda/function_with_snapstart > 2024/11/13/[2]7b2246d964d142faadeaac697caf5c2e
Log events
Actions
Start tailing
Create metric filter
Filter events - press enter to search
Clear
1m
30m
1h
12h
Custom
UTC timezone
Display
Timestamp
Message

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

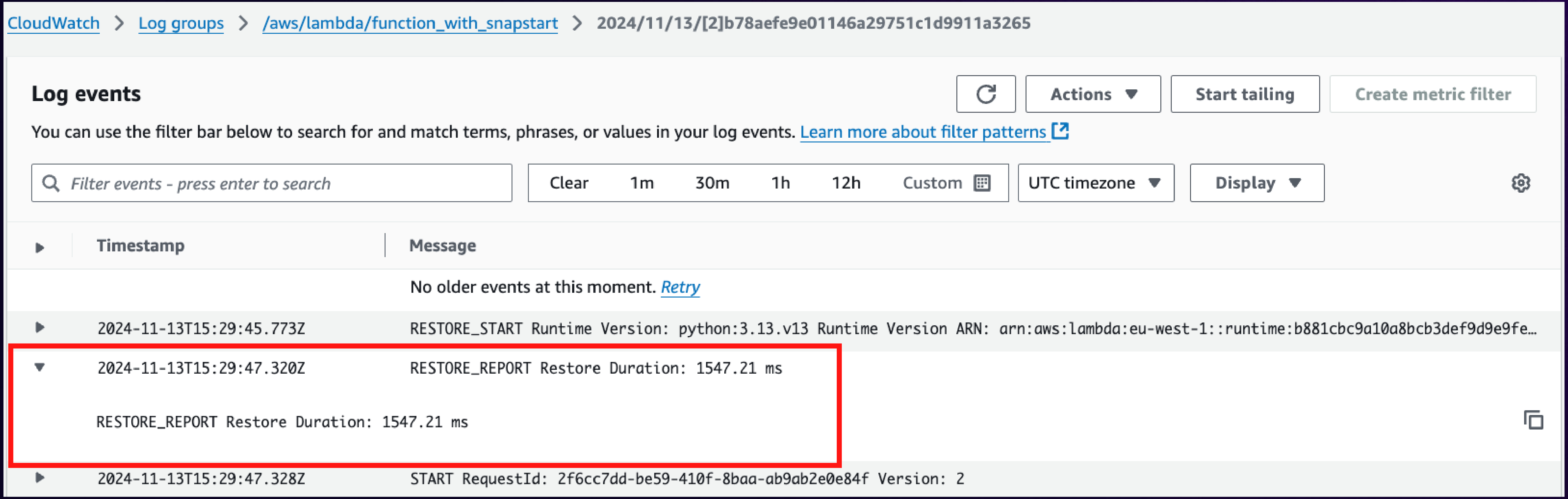
No older events at this moment. *Retry*

```
2024-11-13T15:28:16.451Z    INIT_START Runtime Version: python:3.13.v13 Runtime Version ARN: arn:aws:lambda:eu-west-1::runtime:b881cbc9a10a8bcb3def9d9e9fe38f…
2024-11-13T15:28:17.860Z    INIT_REPORT Init Duration: 1408.76 ms

INIT_REPORT Init Duration: 1408.76 ms
```

No newer events at this moment. *Auto retry paused.* Resume

Lambda SnapStart – CloudWatch logs

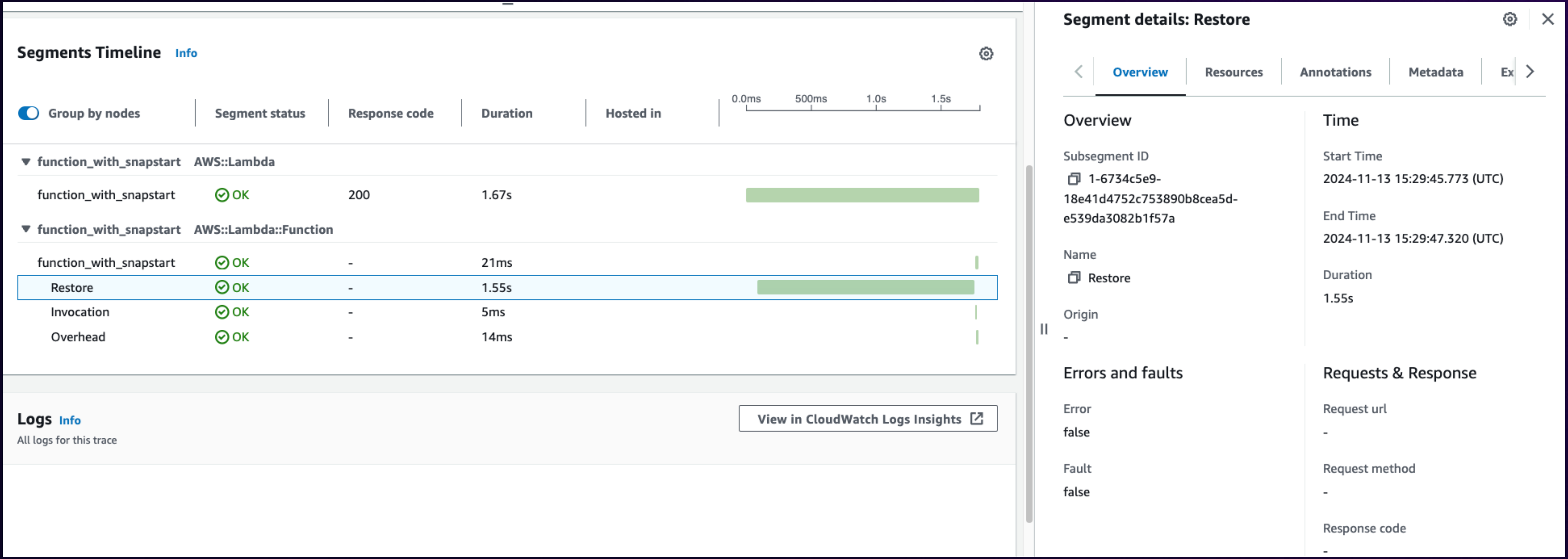
Invocation logs


CloudWatch > Log groups > /aws/lambda/function_with_snapstart > 2024/11/13/[2]b78aefe9e01146a29751c1d9911a3265
Log events
Actions
Start tailing
Create metric filter
Filter events - press enter to search
Clear
1m
30m
1h
12h
Custom
UTC timezone
Display

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                   Message
                            No older events at this moment. Retry
2024-11-13T15:29:45.773Z    RESTORE_START Runtime Version: python:3.13.v13 Runtime Version ARN: arn:aws:lambda:eu-west-1::runtime:b881cbc9a10a8bcb3def9d9e9fe…
2024-11-13T15:29:47.320Z    RESTORE_REPORT Restore Duration: 1547.21 ms

RESTORE_REPORT Restore Duration: 1547.21 ms

2024-11-13T15:29:47.328Z    START RequestId: 2f6cc7dd-be59-410f-8baa-ab9ab2e0e84f Version: 2
```

Lambda SnapStart – AWS X-Ray



Segments Timeline Info

Group by nodes	Segment status	Response code	Duration	Hosted in
▼ function_with_snapstart AWS::Lambda				
function_with_snapstart	✓ OK	200	1.67s	
▼ function_with_snapstart AWS::Lambda::Function				
function_with_snapstart	✓ OK	-	21ms	
Restore	✓ OK	-	1.55s	
Invocation	✓ OK	-	5ms	
Overhead	✓ OK	-	14ms	

Logs Info
All logs for this trace
View in CloudWatch Logs Insights

Segment details: Restore

Overview | Resources | Annotations | Metadata | Ex

Overview

Subsegment ID
1-6734c5e9-18e41d4752c753890b8cea5d-e539da3082b1f57a

Name
Restore

Origin
-

Time

Start Time
2024-11-13 15:29:45.773 (UTC)

End Time
2024-11-13 15:29:47.320 (UTC)

Duration
1.55s

Errors and faults

Error
false

Fault
false

Requests & Response

Request url
-

Request method
-

Response code
-

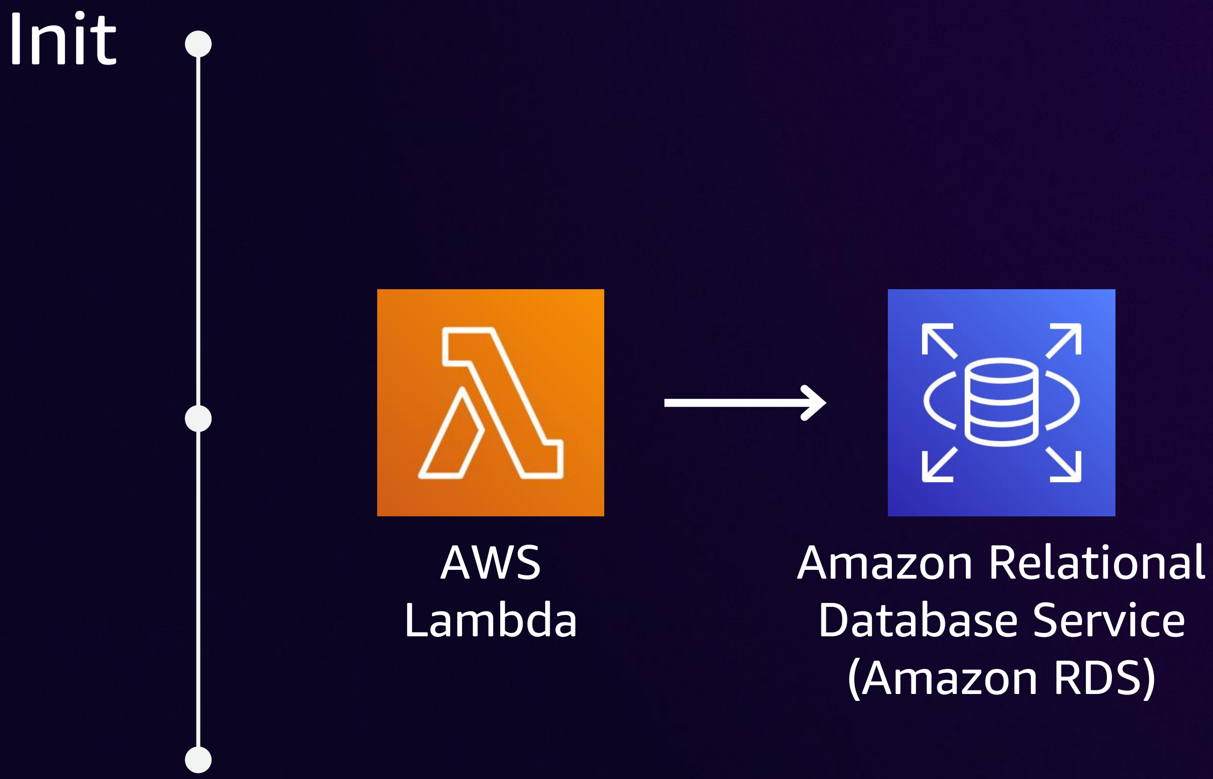
Best practices

Best practices

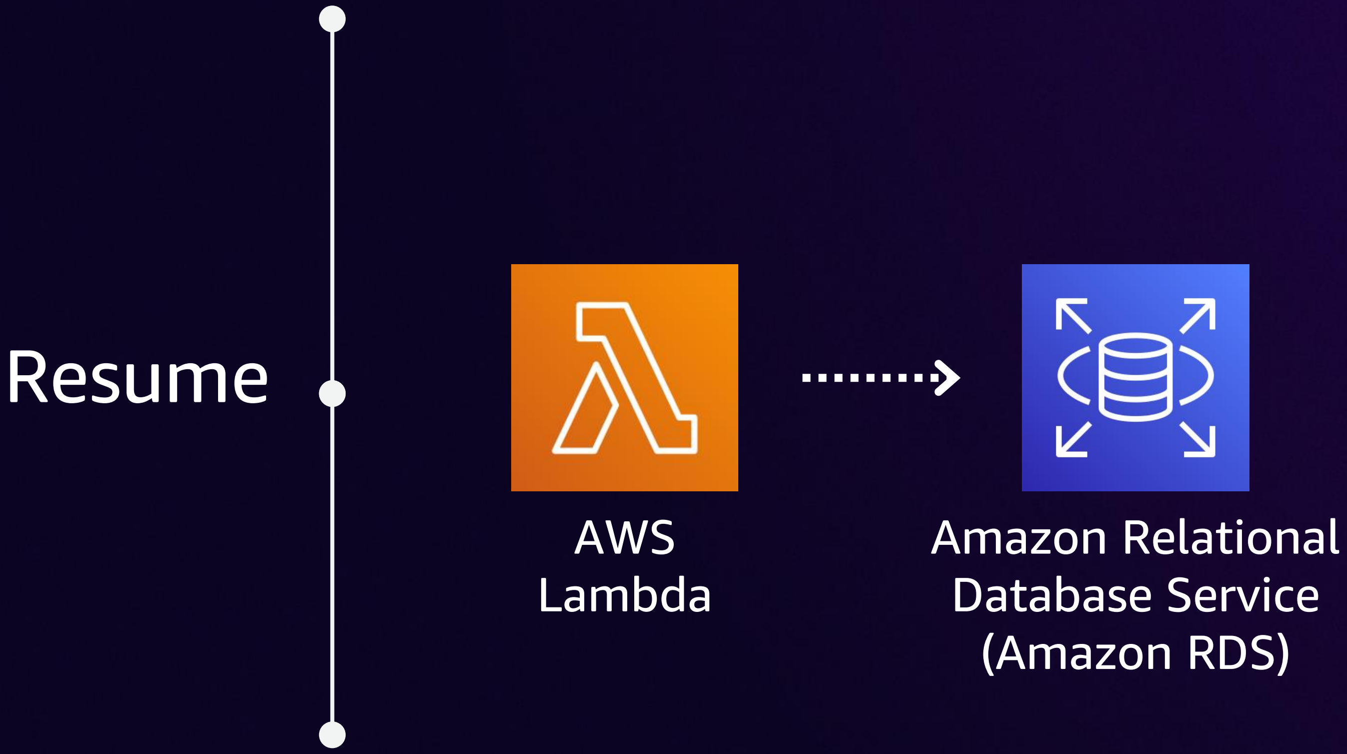
Network connections

Ephemeral data

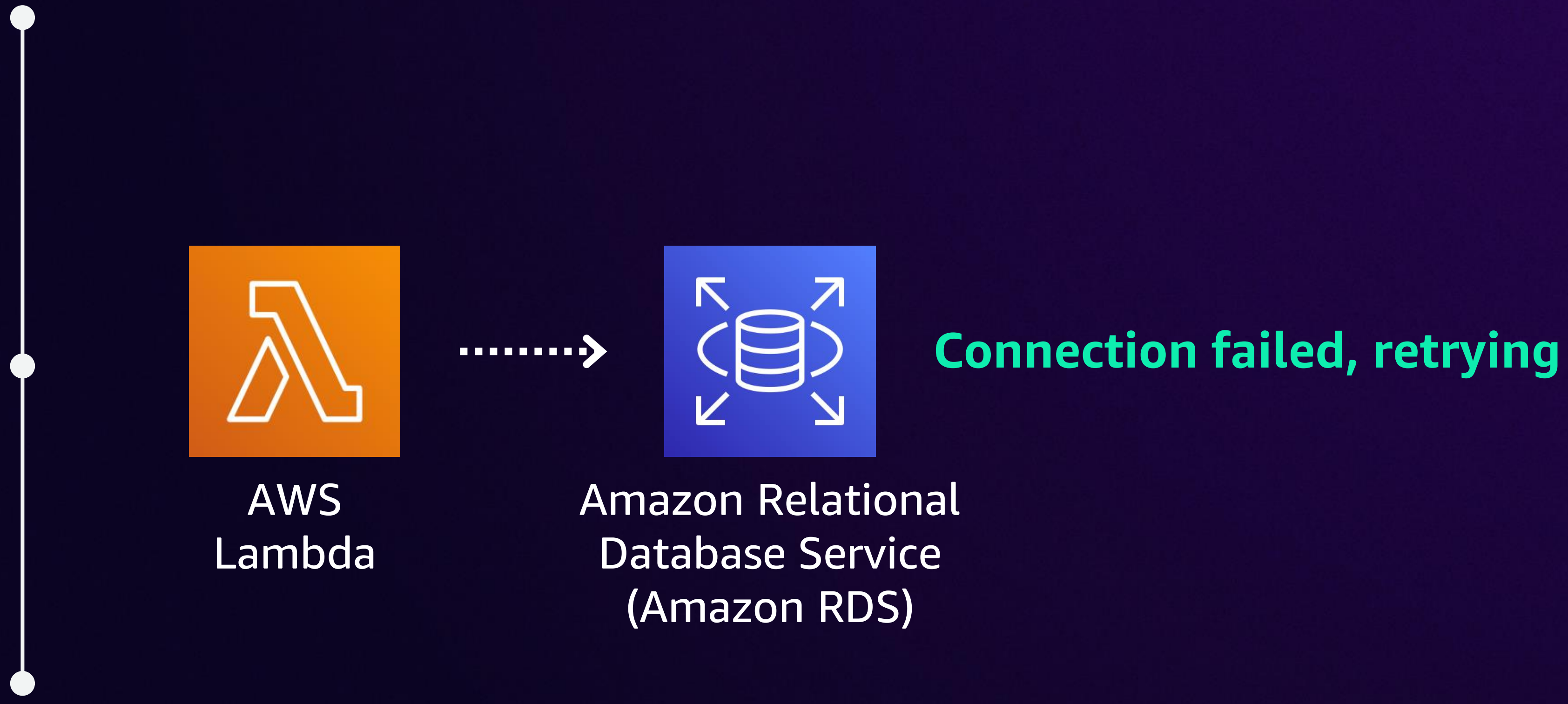
Network connections



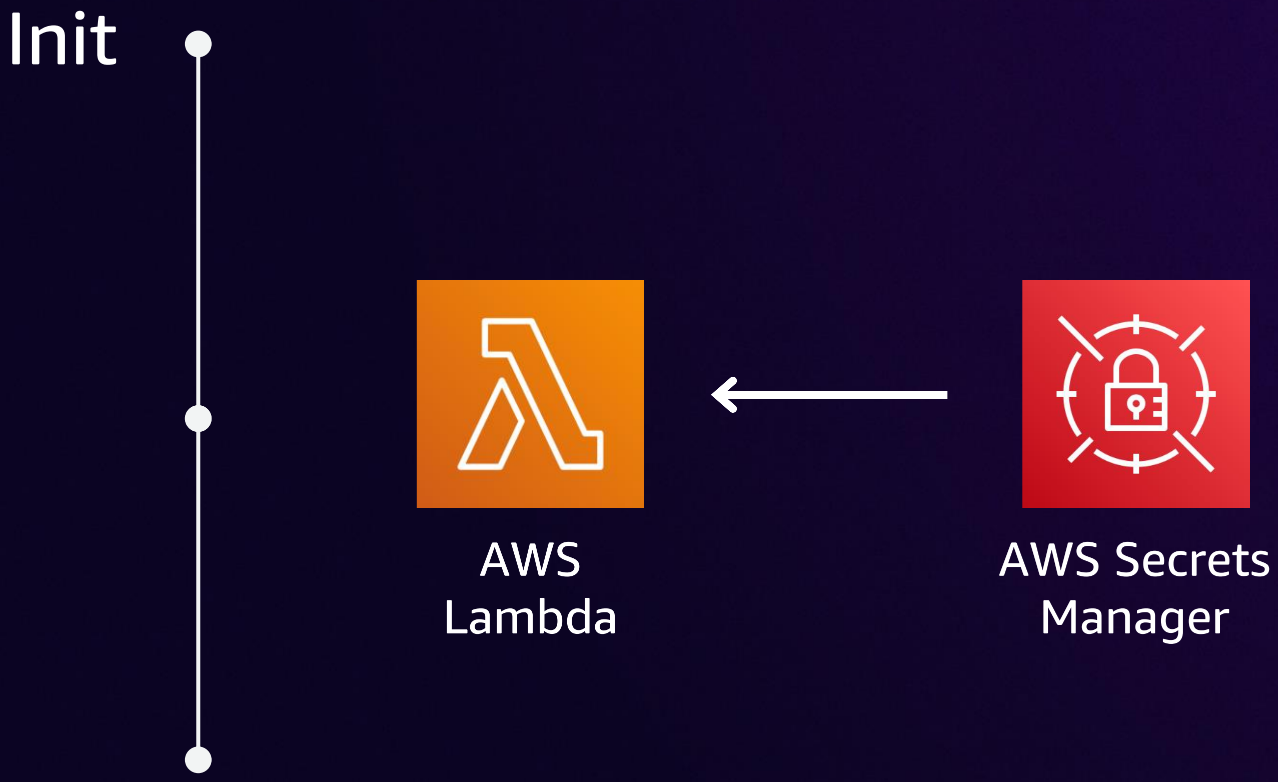
Network connections



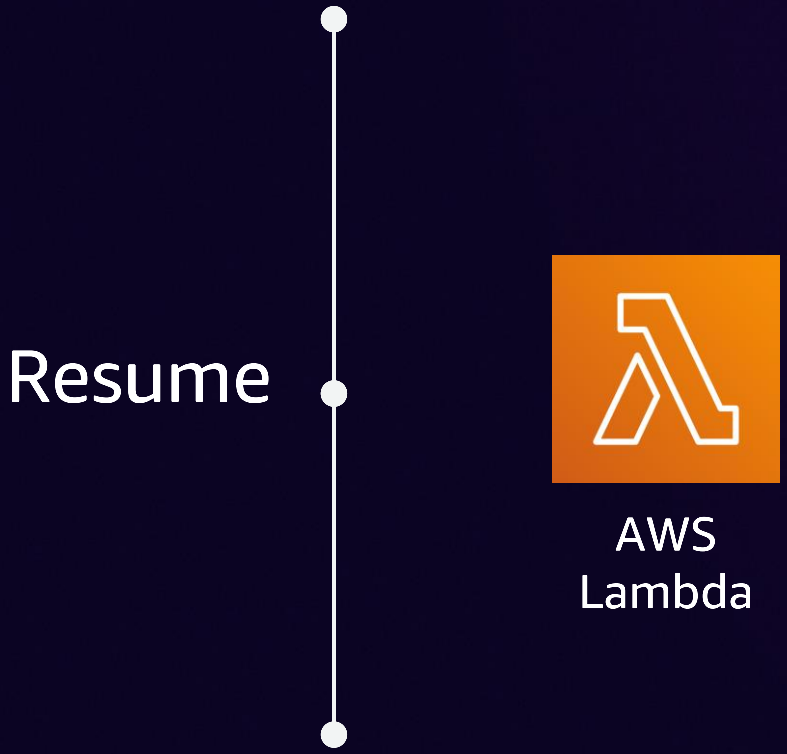
Network connections


Invoke
AWS Lambda
Amazon Relational Database Service (Amazon RDS)
Connection failed, retrying

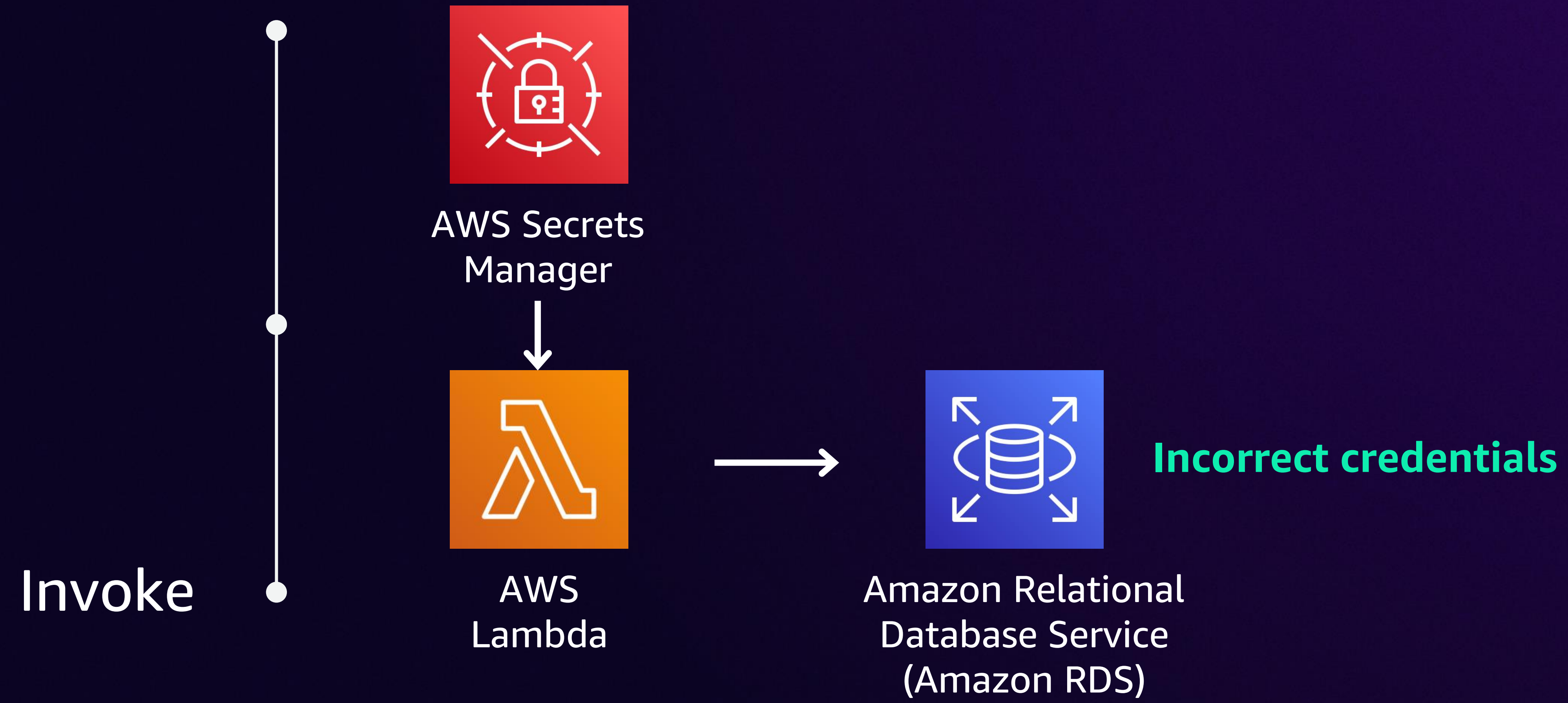
Ephemeral data



Ephemeral data



Ephemeral data



Pricing

SnapStart pricing

Usage priced along two dimensions – represents a nominal added charge for typical use cases

- Cache – $3.9 per GB-month
  - Charged over active duration of a function version ($0.0000015046 per GB-second)
  - Lower costs by deleting unused versions
- Restore – $1.4 per GB restored with 10K restores
  - Charged per GB restored ($0.0001397998 per GB restored)

Pricing example (monthly)

- Let's assume a 1 GB function, 300 ms execution duration

- 100M invokes, **250K restores** (i.e., cold starts)
  - Total charges: $558.8
    - Compute charges: $500; request charges: $20 (***no change***)
    - SnapStart cache charges: $3.9 ($3.9 x 1 GB)
    - SnapStart restore charges: $34.9 ($0.0001397998 x 1 GB x 250K restores)

Takeaways

- Lambda SnapStart is now available for Python and .NET
- SnapStart only works with published versions or alias of your Lambda function
- Lower SnapStart costs by deleting unused versions

Thank you!

Please complete the session survey in the mobile app

Samrat Karak

(he/him)
Principal Product Manager,
AWS Lambda

AWS

Leandro Damascena

(he/him)
Sr. Engineer SA,
Powertools for AWS Lambda

AWS