AWS re:Invent

GPSTEC411

Bringing serverless assets across projects for productivity gain

Abhijit Vaidya
Solutions Architect, Partner
Amazon Web Services

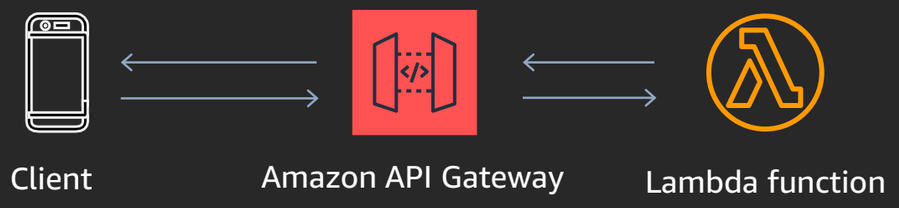
Ronald Widha
Mgr. Solutions Architect, Partner
Amazon Web Services

AWS re:Invent

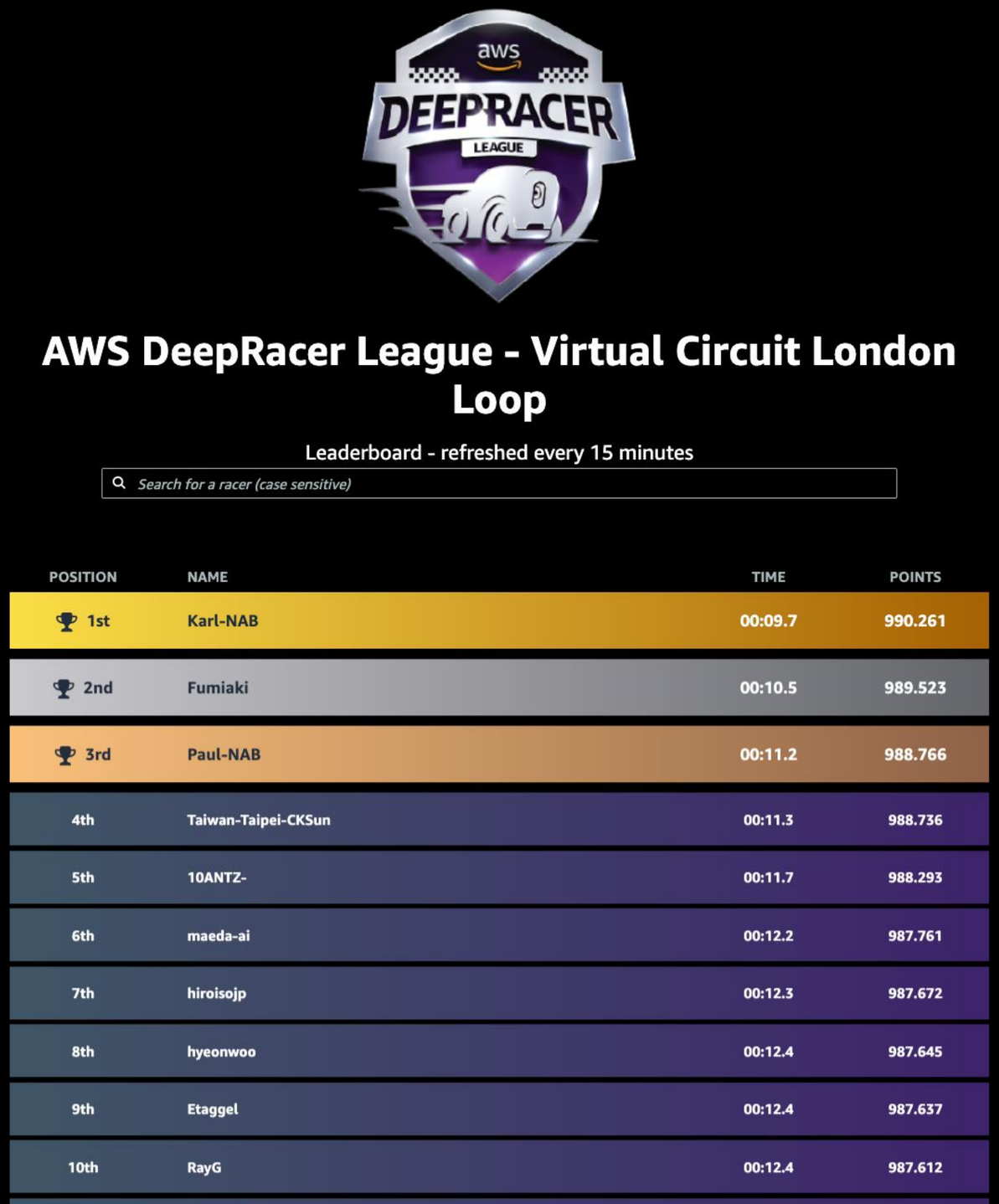
Agenda

- Code reuse
- Libraries and packages
- Lambda layers
- Serverless applications and repository
- Externalizing configuration:
  Environment variables, parameter stores, and step functions

The most awesome
Leaderboard microservice

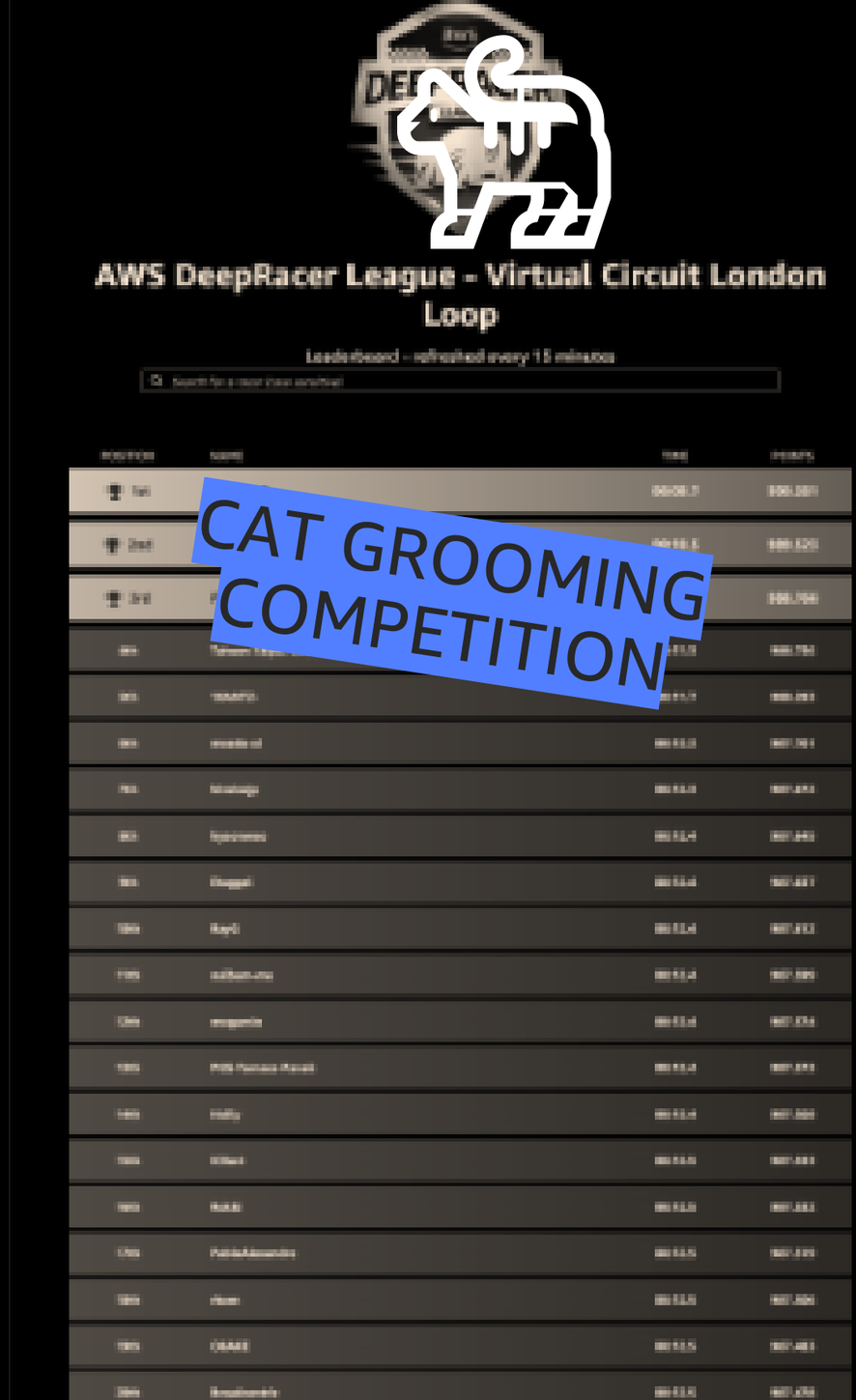


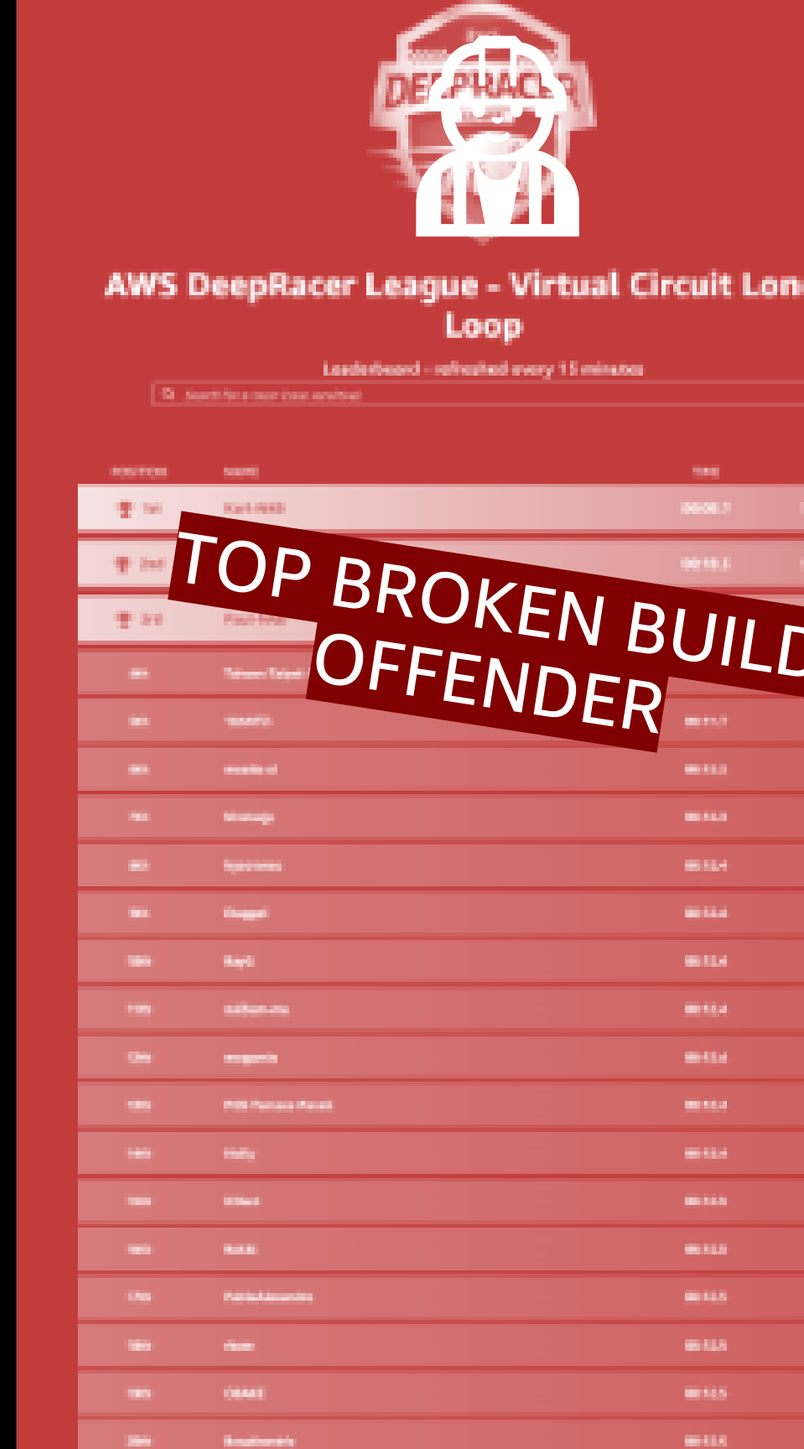
- Highly scalable
- Highly available
- Highly secure
- Highly performant



POSITION	NAME	TIME	POINTS
1st	Karl-NAB	00:09.7	990.261
2nd	Fumiaki	00:10.5	989.523
3rd	Paul-NAB	00:11.2	988.766
4th	Taiwan-Taipei-CKSun	00:11.3	988.736
5th	10ANTZ-	00:11.7	988.293
6th	maeda-ai	00:12.2	987.761
7th	hiroisojp	00:12.3	987.672
8th	hyeonwoo	00:12.4	987.645
9th	Etaggel	00:12.4	987.637
10th	RayG	00:12.4	987.612


AWS DeepRacer League - Virtual Circuit London Loop
Leaderboard - refreshed every 15 minutes
HOTDOG EATING LEAGUE


AWS DeepRacer League - Virtual Circuit London Loop
Leaderboard - refreshed every 15 minutes
CAT GROOMING COMPETITION


AWS DeepRacer League - Virtual Circuit Lon
Loop
Leaderboard - refreshed every 15 minutes
TOP BROKEN BUILD
OFFENDER

Anatomy of an AWS Lambda function

```
Function myhandler(event, context) {
    <Event handling logic> {
          result = SubfunctionA()
      } else {
            result = SubfunctionB()

    return result;
}

Function subFunctionA(thing){
 ## logic here
}

Function subFunctionB(thing){
 ## logic here
}
```

Code reuse

Q: How would you reuse this code across projects?

(This is going to be on the whiteboard)

Externalize common code

NPM

Maven

Externalize configuration

Git repo

Multi deployment

????@#%$@#!!

Multi tenant

Anatomy of a Lambda function

```
Import sdk
Import http-lib
Import ham-sandwich
Pre-handler-secret-getter()
Pre-handler-db-connect()
```

Dependencies, configuration information, common helper functions

```
Function myhandler(event, context) {
    <Event handling logic> {
                result = SubfunctionA()
            }else {
                        result = SubfunctionB()

    return result;
}
```

Common helper functions

Business logic sub-functions

Externalizing common code

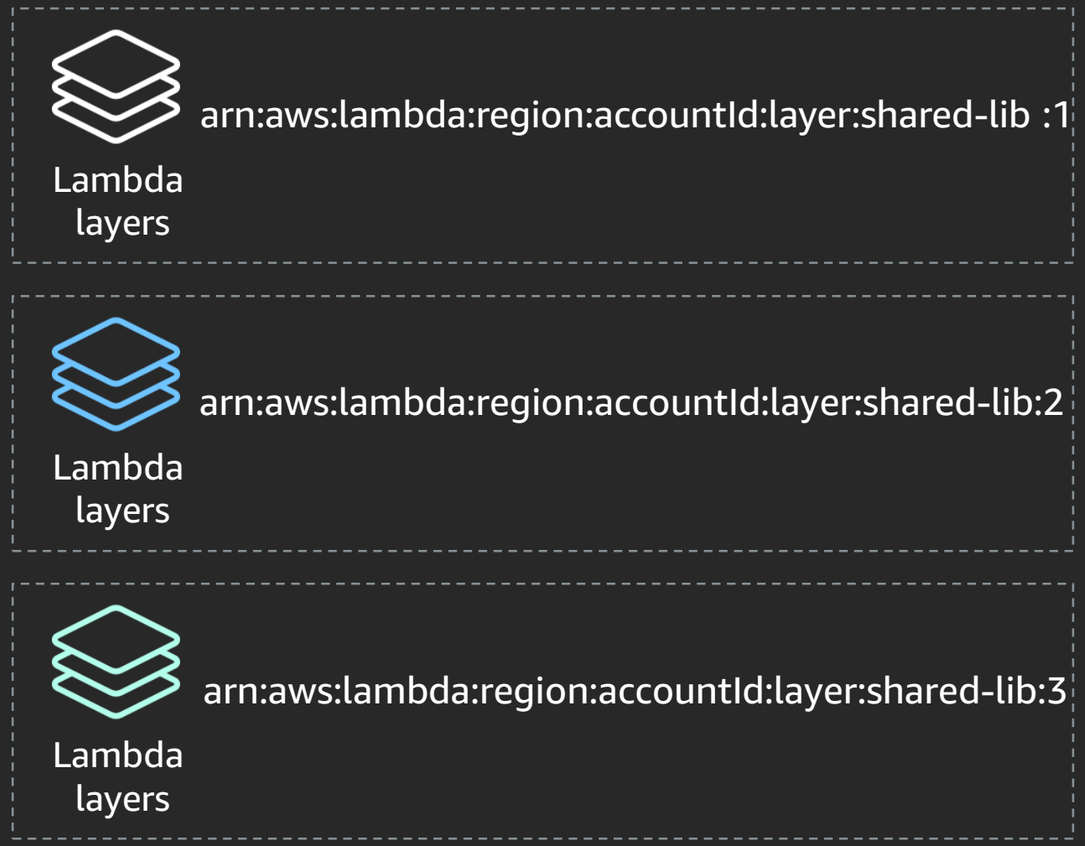
- Refactor out horizontal concern
- Package as a library (node modules, jar, etc.)

Characteristics

- Simple to do
- Great for open source
- Runtime dependent
- Difficult to version
- Difficult to update
- Large dependencies = cold starts

Using Lambda layers

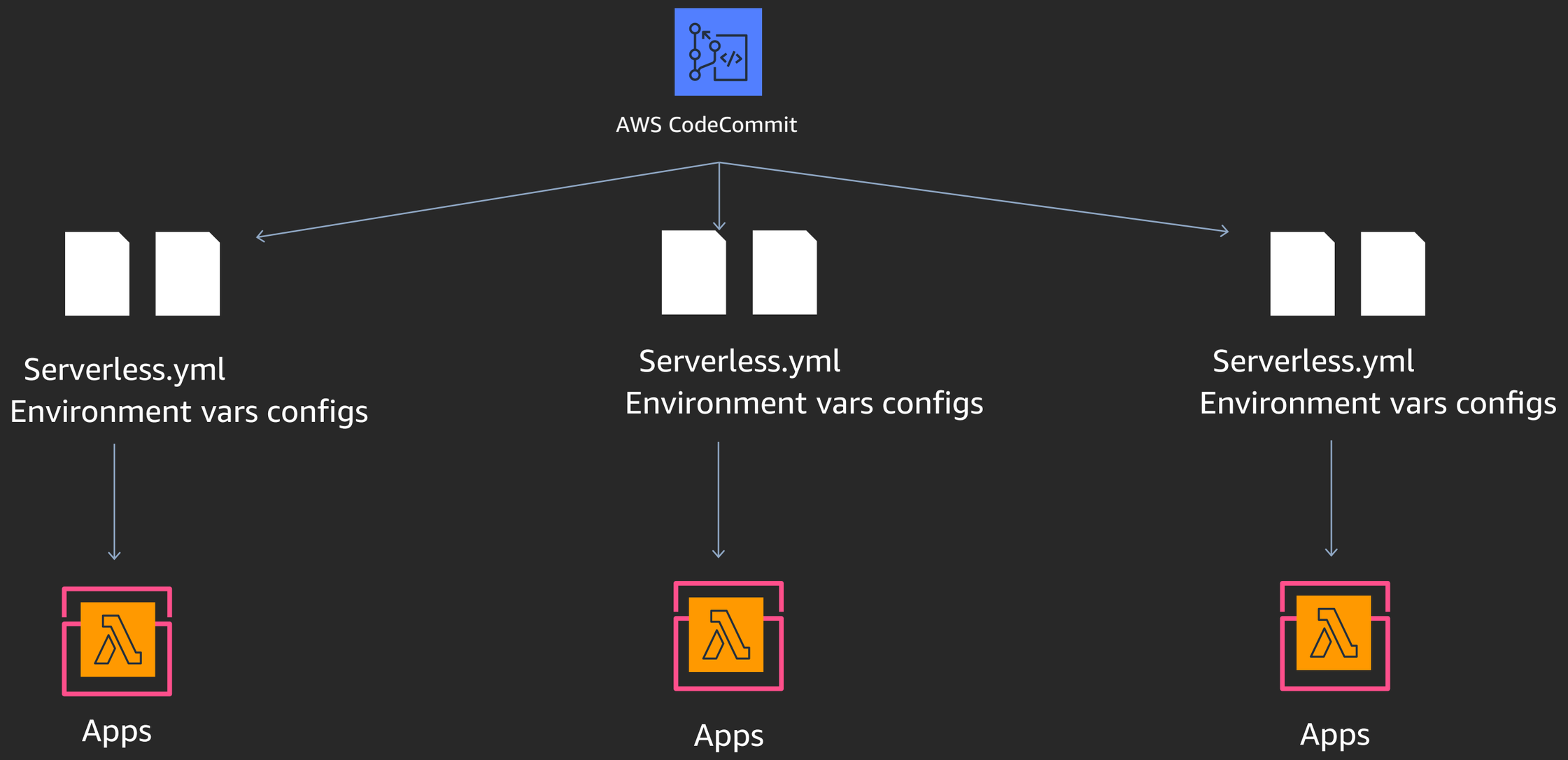
- Put common components in a .zip file and upload it as a Lambda layer
- Layers are immutable and can be versioned to manage updates
- When a version is deleted or permissions to use it are revoked, functions that used it previously will continue to work, but you won't be able to create new ones
- You can reference up to five layers, one of which can optionally be a custom runtime



Demo

AWS re:Invent

Externalizing configuration



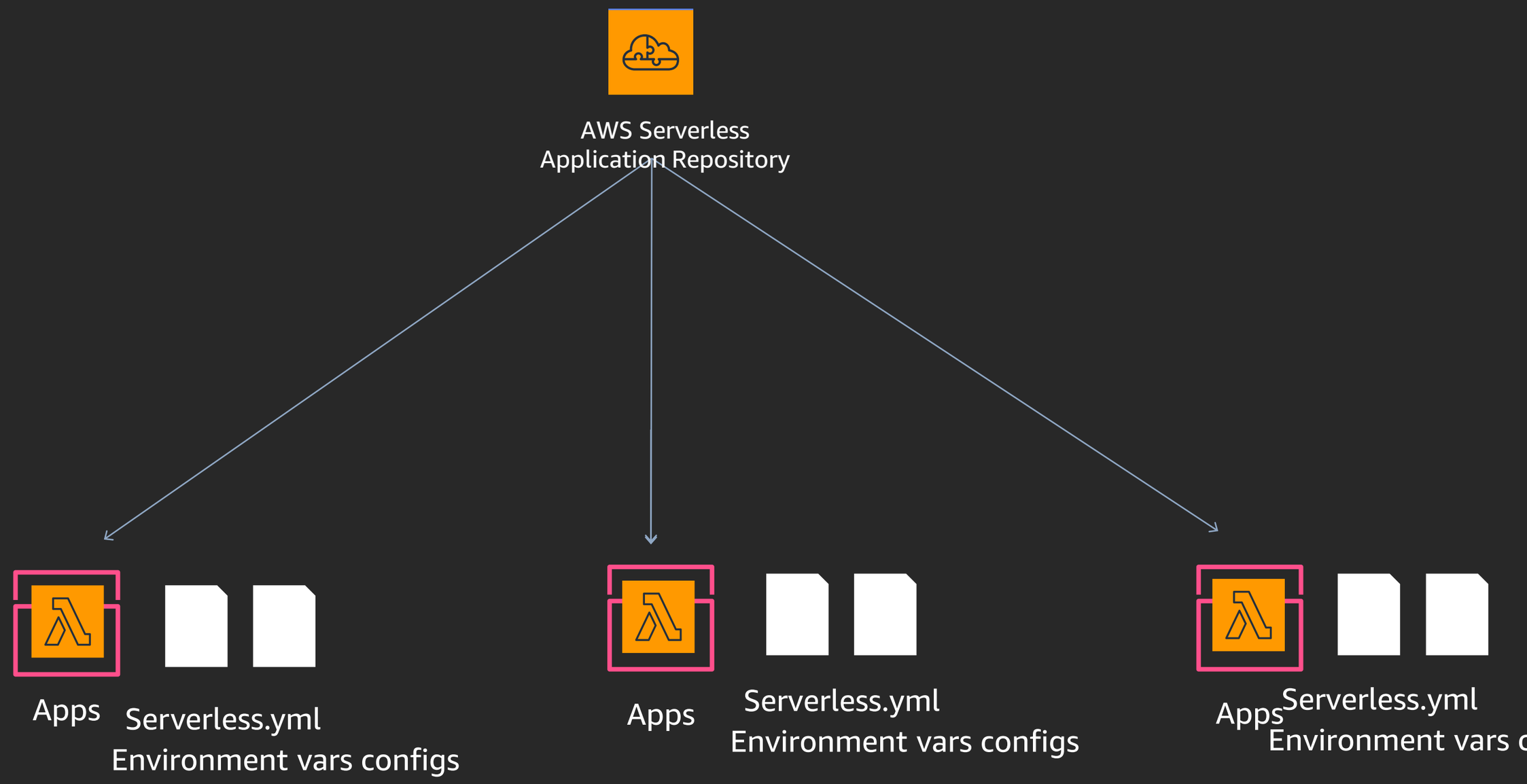
Externalizing configuration

- Taking out all the configs
- 12 Factor principles: using environment variable

Characteristics

- More difficult to onboard new ”customers”
- Require development environment (pre-compile, npm install, etc.)

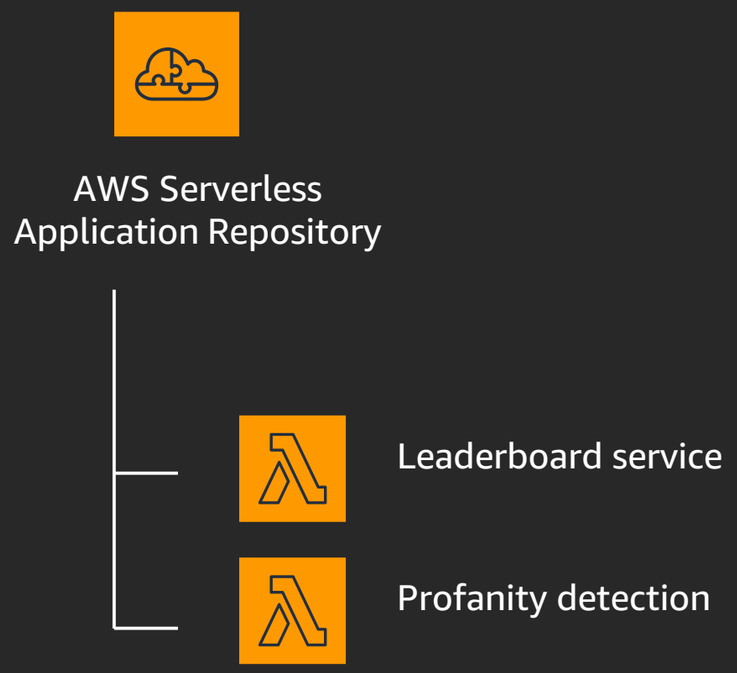
Externalizing configuration



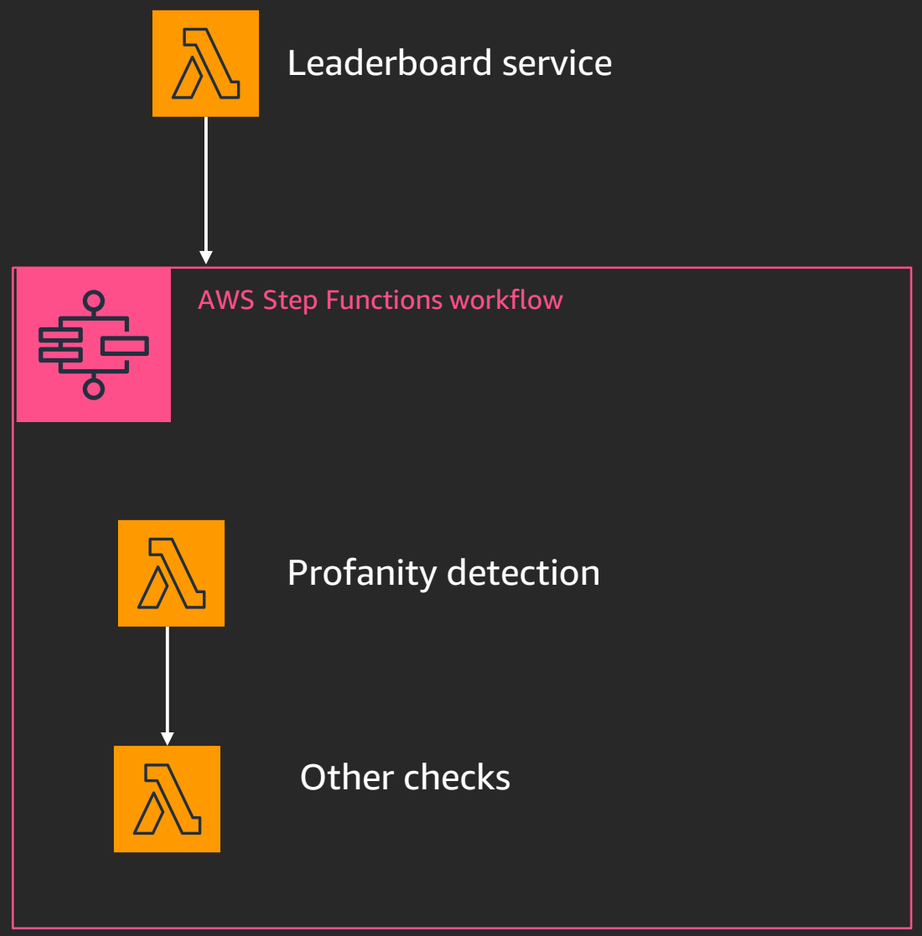
Demo

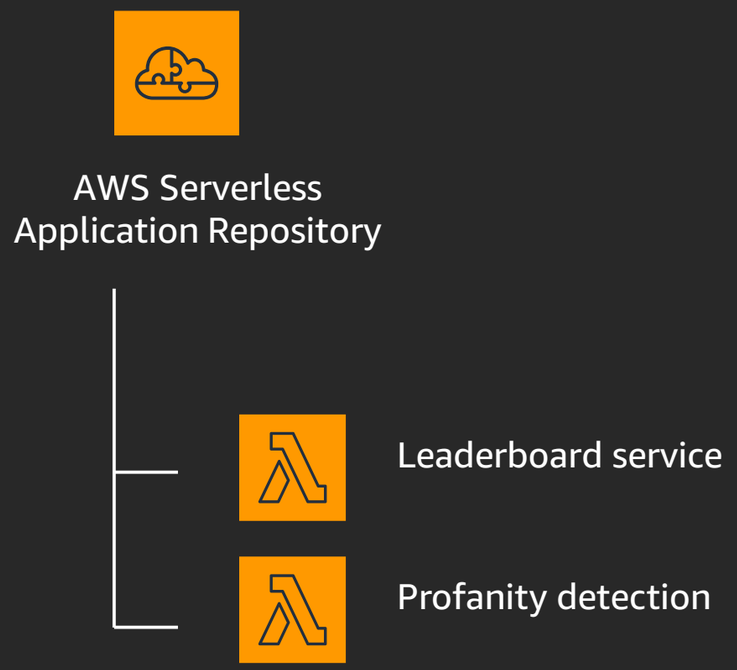
AWS re:Invent

New functionality request: Profanity detection



New functionality request: Profanity detection



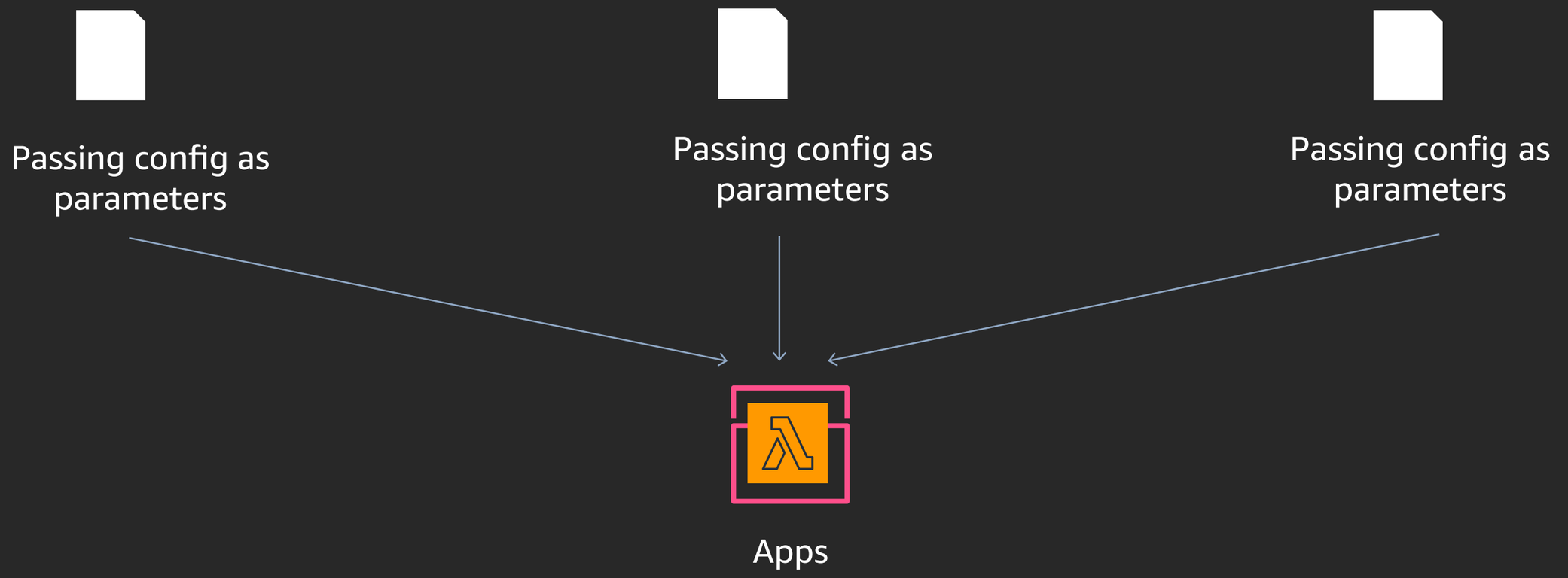


Demo

AWS re:Invent

Configuration as data flow

- App behavior changes based as instructed by the caller
- Suitable for "horizontal" microservices



```
{
        1: pen-pineapple
        2; apple-pen
}
```

```
{
        x: $.1
        y: $.2
}
```

```
X = event["…"]
Y = event["…"]
```

```
Sns.message={1:pen, 2:pineapple}
```

```
{
        x: $..records[0].Sns.message(@.1)
        y: $..records[0].Sns.message(@.2)
}
```

The config is a JSONPath, which can be injected as part of the args, or env var

Event-source-adapter

Node

```
var adapter =
require('EventSourceAdapter')
var {x,y} = adapter.map(event, …)
```

Python

```
import eventsourceadapter
p =
eventsourceadapter(config).map(event)
```

Java

```
class Parameters { String param1; Integer param2; }
EventSourceAdapter<Parameters> adapter = new EventSourceAdapter<Parameters>(config);
Parameters p = adapter.create(event, Parameters.class);
```

Thank you!

AWS re:Invent

Please complete the session survey in the mobile app.

AWS re:Invent