Serverless updates from AWS re:Invent 2025

👋 Show of hands

The biggest secret of Serverless

(and many other serverless services)

Hundreds of thousands of
EC2 instances under-the-hood

Usman Khalid

He/His/Him

Head of Serverless Compute

linkedin.com/in/ahmedusmankhalid/

hundreds of thousands of servers.

What actually is serverless

Managed by AWS

- management of servers
- management of runtimes
- scaling
- load balancing
- request routing
- patching
- …

Pricing model

- pay per request
- pay per usage
- pay per duration
- scaling to 0
- on demand pricing

“I don’t want to manage my own cluster - that’s what AWS is for.”

AWS serverless

Today's focus:

- AWS Lambda
- AWS ECS
- DynamoDB & DSQL

a full list of
AWS serverless services

AWS Lambda

- FaaS (Function as a Service)
- various runtimes (official and unofficial)
- Docker lambda, Lambda Web Adapter
- **128MB - 10 240MB RAM, 1 - 6 vCPU**
- **15 minutes timeout**
- rapid auto scaling
- fully managed
- Event Source Mapping (ESM)

Lambda Managed Instances (LMI)

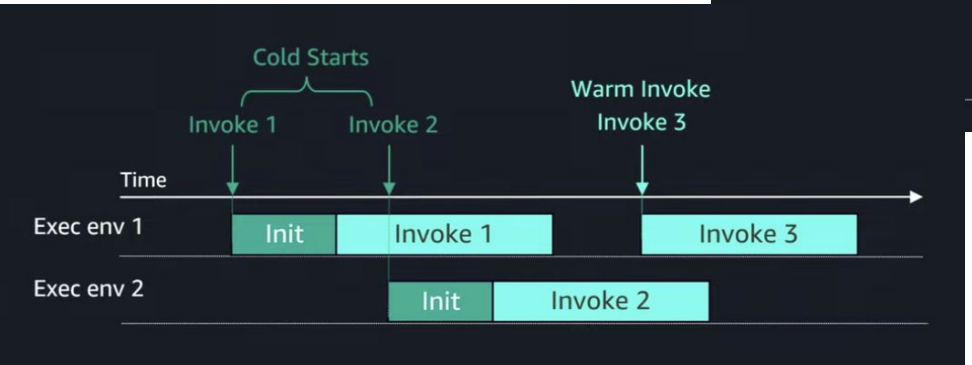
- pick underlying EC2 instance type
- fully managed
- same model as Lambda (no need to refactor/re-architect)
- great for steady and CPU heavy workloads (for unpredictable use standard Lambda)
- multi-concurrency with less cold starts
- you pay for EC2 + 15% management overhead
- supports Saving plans and Reserved Instances

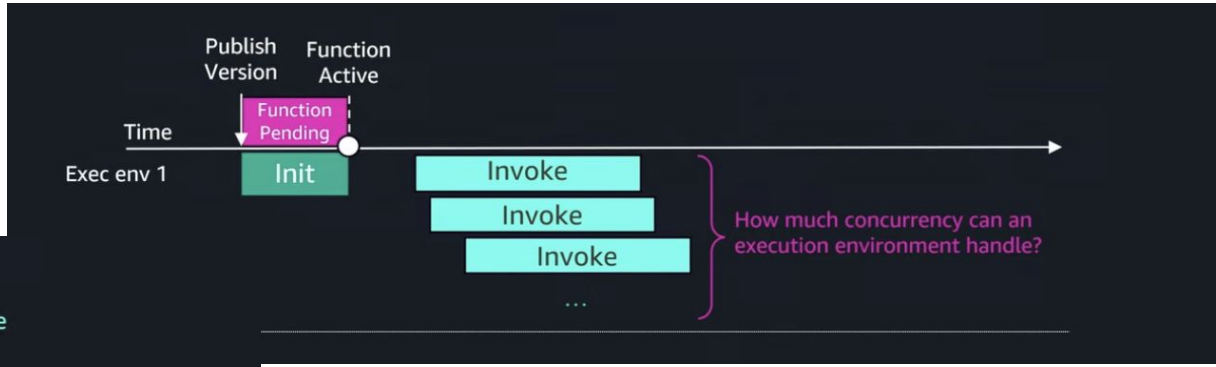
address compute choice, cost and cold starts limitations

Lambda Managed Instances (LMI)

1. create capacity provider (optionally select instance types)
2. associate your lambda function with capacity provider (adjust memory/cpu ratio)
3. deploy your function

Standard Lambda 👇





LMI multi-concurrency 👆

```typescript
import { App, Stack } from "aws-cdk-lib";
import { SecurityGroup, Vpc } from "aws-cdk-lib/aws-ec2";
import {
  CapacityProvider,
  ScalingOptions,
  TargetTrackingScalingPolicy,
} from "aws-cdk-lib/aws-lambda";

const stack = new Stack(new App(), "MyStack");
const vpc = new Vpc(stack, "MyVpc");
const securityGroup = new SecurityGroup(stack, "SecurityGroup", { vpc });

const capacityProvider = new CapacityProvider(stack, "MyCapacityProvider", {
  subnets: vpc.privateSubnets,
  securityGroups: [securityGroup],
  scalingOptions: ScalingOptions.manual([
    TargetTrackingScalingPolicy.cpuUtilization(70),
  ]),
  
});
```

```
architectures? (property) CapacityProviderProps.architectures?
capacityProviderName? (property) CapacityProviderProps.capacit
instanceTypeFilter? (property) CapacityProviderProps.instanceT
kmsKey? (property) CapacityProviderProps.kmsKey?: IKey | undef
maxVCpuCount? (property) CapacityProviderProps.maxVCpuCount?:
operatorRole? (property) CapacityProviderProps.operatorRole?:
```

The instruction set architecture required for compute instances. Only one architecture can be specified per capacity provider.

@default

- No architecture constraints specified

```
24
25  const lambda = new NodejsFunction(stack, "MyLambda", {
26    entry: "handler.js",
27    runtime: Runtime.NODEJS_24_X,
28    architecture: Architecture.ARM_64,
29    memorySize: 1024,
30  });
31  capacityProvider.addFunction(lambda, {
32    executionEnvironmentMemoryGiBPerVCpu: 4,
33    perExecutionEnvironmentMaxConcurrency: 20,
34
35  })
36
```

```
latestPublishedScalingConfig? (property) CapacityProviderFunct
publishToLatestPublished? (property) CapacityProviderFunctionC
```

The scaling options that are applied to the $LATEST.PUBLISHED version.

@default

- No scaling limitations are applied to the $LATEST.PUBLISHED version.

Lambda: Durable Functions

- workflow orchestration within single AWS Lambda
- powerful SDK (Node.js and Python, more are coming soon)
- you don’t pay for waiting (up to 1 year execution time)
- checkpoints and replays
- similar to Step Functions

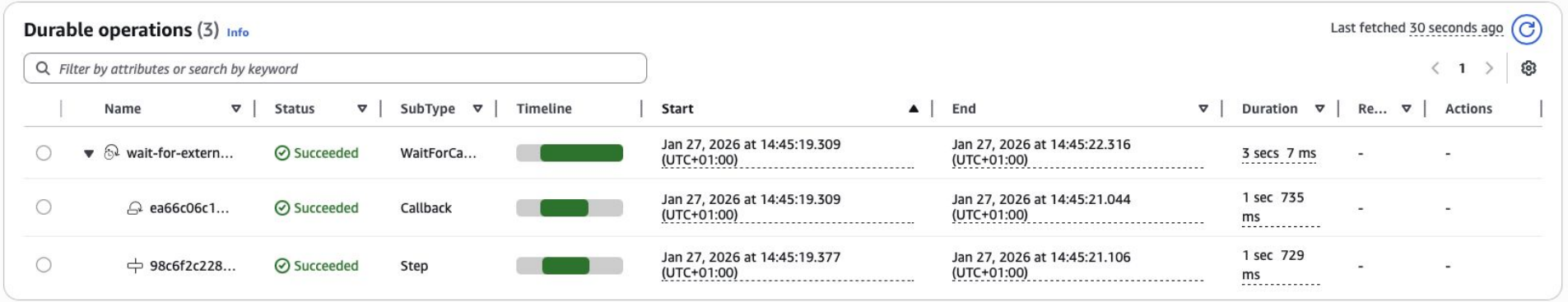
address timeout and orchestration limitations

```typescript
import { Duration } from '../../core';
/**
 * Configuration for durable functions.
 *
 * Lambda durable functions allow for long-running executions with persistent state.
 */
export interface DurableConfig {
    /**
     * The amount of time that Lambda allows a durable function to run before stopping it.
     *
     * Must be between 1 and 31,622,400 seconds (366 days).
     */
    readonly executionTimeout: Duration;
    /**
     * The number of days after a durable execution is closed that Lambda retains its history.
     *
     * Must be between 1 and 90 days.
     *
     * The underlying configuration is expressed in whole numbers of days. Providing a Duration that
     * does not represent a whole number of days will result in a runtime or deployment error.
     *
     * @default Duration.days(14)
     */
    readonly retentionPeriod?: Duration;
}
```

Durable Functions demo - what I've learned

1. problems with permissions
2. old SDK in Node.js runtime
3. permissions setup

public gist
with demo code



Durable operations (3) Info

Last fetched 30 seconds ago

Filter by attributes or search by keyword

1

Name	Status	SubType	Timeline	Start	End	Duration	Re...	Actions
wait-for-extern...	Succeeded	WaitForCa...		Jan 27, 2026 at 14:45:19.309 (UTC+01:00)	Jan 27, 2026 at 14:45:22.316 (UTC+01:00)	3 secs 7 ms	-	-
ea66c06c1...	Succeeded	Callback		Jan 27, 2026 at 14:45:19.309 (UTC+01:00)	Jan 27, 2026 at 14:45:21.044 (UTC+01:00)	1 sec 735 ms	-	-
98c6f2c228...	Succeeded	Step		Jan 27, 2026 at 14:45:19.377 (UTC+01:00)	Jan 27, 2026 at 14:45:21.106 (UTC+01:00)	1 sec 729 ms	-	-

Step-Functions vs. Durable functions

Choosing a service

Use to orchestrate across AWS services

If you want a visual builder experience

Never want to manage or patch

Use when orchestrating at the application level

If you want to use your favorite programming language

Don’t mind updating your functions when needed

Step-Functions vs. Durable functions - Eric Johnson

CNS380

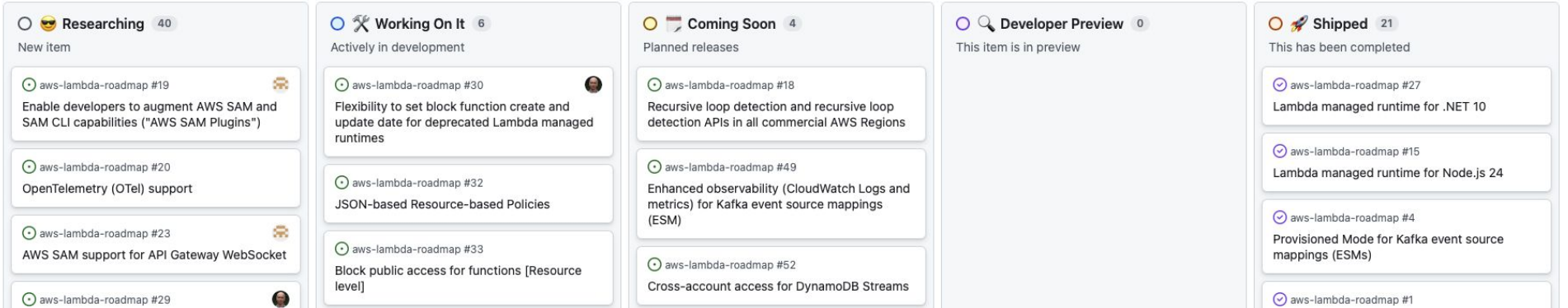
Durable Functions

API313

StepFunctions

AWS Lambda - worth mentioning

- public Lambda Roadmap on Github
- Tenant Isolation (pre:Invent)
- support for Egress-only Internet Gateway (IPv6 only)
- StepFunctions JSONata support (lightweight transformations without Lambda)



Elastic Container Service (ECS)

- native container orchestration service
- **compute service - Fargate (serverless), EC2**
- no control plane to manage
- blue/green deployments
- services and tasks
- pay as go model for used resources
- used for long running (15m+) serverless tasks before Durable Functions
- various deployment options (CDK, CloudFormation, AWS Copilot)

ECS Managed Instances (CNS342)

- simplicity of Fargate with control of EC2
- optional selection of instance types
- instances deployed to customer account
- fully managed by AWS
- ~2w retention lifecycle of instances
- cost optimized
- new capacity provider for EC2 Managed Instance

address Fargate flexibility

DynamoDB

- schemaless NoSQL key-value store
- highly available, scalable,
  fully managed, serverless storage
- strong consistency and eventual consistency modes
  (both regional and multi-regional)

“With DynamoDB, you are the query planner.”
- Alex DeBrie at DynamoDB Day 2025

DynamoDB - multi-attribute composite keys

- global secondary index (GSI) support
- up to 8 attributes in key
- up to 20 GSI per table (+ 5 LSI)

- no more “country#state#province#city” nonsense

- still works in hierarchy (eg. you need to query from left to right without skipping)
- no table primary key support yet

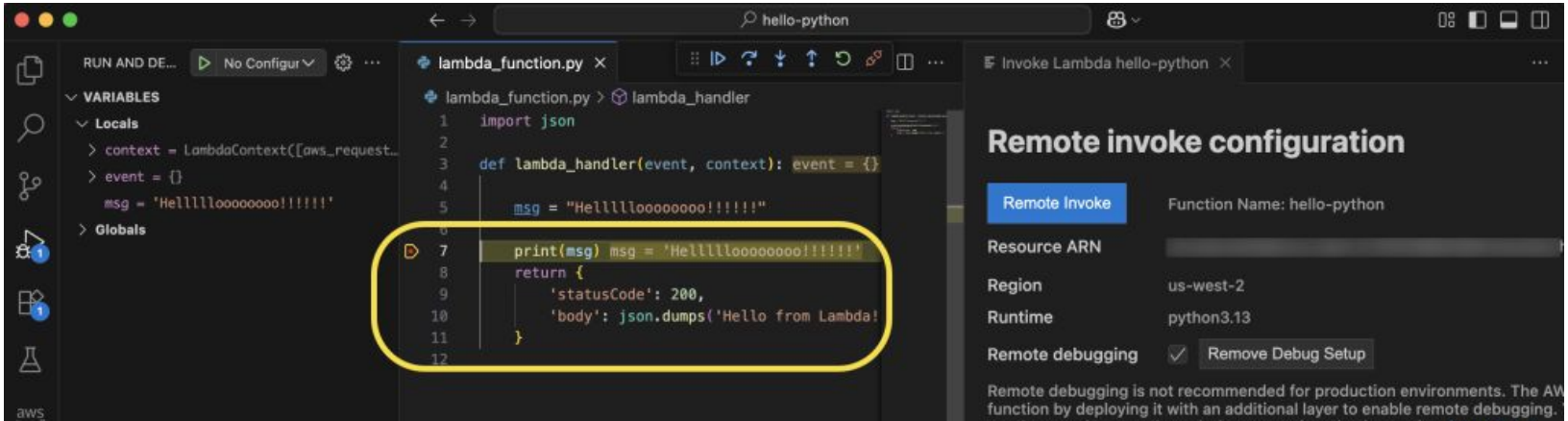
Aurora DSQL

- **distributed** SQL database (postgres compatible)
- fully managed, scalable, serverless (super simple to create and use)
- for those who love their SQL
- no foreign keys constraints support
- cross-region strong consistency by default (everything runs in transaction)
- optimistic concurrency control (OCC) (DAT455 - A Tale of Two Transactions)

if you have time watch deep dive on architecture

Local development

- AWS toolkit for Visual Studio Code
- LocalStack support (dockerized AWS simulator)
- remote debugging support
- Console to IDE (start in console move to VSCode)



re:Watch unofficial - AWS re:Invent Video Library

big thanks to
Martin Damovsky

re:Watch playlist
Serverless

Ivan Barlog

AWS Solutions Architect

Github ivanbarlog

Email ivan@barlog.sk

Web barlog.sk
beesolve.com

LinkedIn ivan-barlog

Go build something!