
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

SVS311

Practical experience with a serverless-first strategy at Capital One

George Mao

Sr. Distinguished Engineer
Capital One

Heeki Park

Principal Solutions Architect
AWS

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Cloud Computing

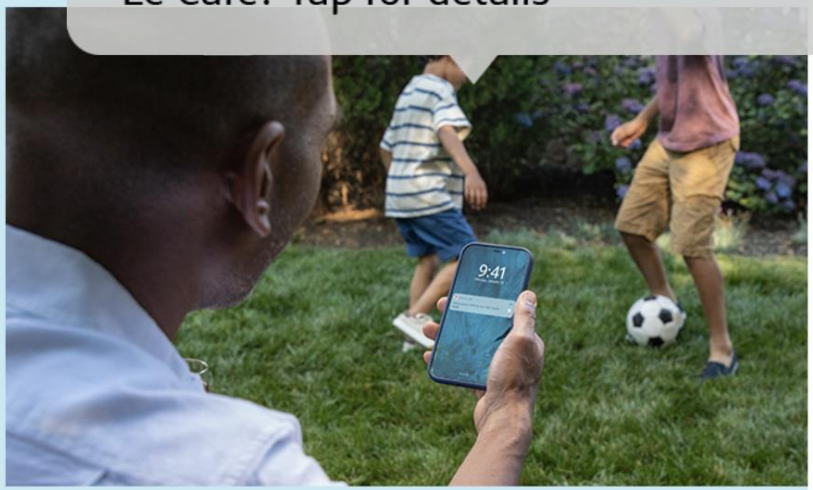
WE'RE ALL-IN

In 2020, we left our data centers to create exceptional banking experiences for our customers, becoming the first bank to go all-in on the public cloud

- Going all in on the cloud—and embracing cloud-native services like serverless computing—has enabled instant provisioning of infrastructure and rapid innovation

- Today, we're using real-time, streaming data at scale, machine learning, and the power of the cloud to solve unique, challenging technology problems and deliver intelligent, personalized solutions that benefit millions of customers







- Capital One Shopping, built in the cloud with microservices architecture and streaming data, helps customers save money shopping online by automatically finding lower prices, coupons and online credits

Capital One

A BANK BUILT LIKE A TECH COMPANY









Capital One



- **Tech is central to everything we do** at Capital One: for nearly a decade, we have invested and invented like the very best technology companies
- Our world-class, in-house technology team now numbers 11,000 people, **85% of which are engineers**
- We're **all in on the cloud like no other bank out there**, which enables us to create exceptional, innovative experiences for our customers
- Advanced, automated DevOps and CI/CD approaches mean engineers spend **more time at the top of the tech stack** building new and unique digital banking experiences

Build	Local developer experience
Deploy	Secrets management Traffic shifting with AWS CodeDeploy
Run	API resiliency patterns Resolving errors with provisioned concurrency Speeding up init with burst CPU SnapStart overview and testing results
Wrap-up	What to avoid/do with AWS Lambda Continue your AWS serverless learning

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Build	Local developer experience
Deploy	Secrets management Traffic shifting with AWS CodeDeploy
Run	API resiliency patterns Resolving errors with provisioned concurrency Speeding up init with burst CPU SnapStart overview and testing results
Wrap-up	What to avoid/do with AWS Lambda Continue your AWS serverless learning

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What is the AWS Serverless Application Model?

An open-source framework, optimized for building serverless applications

Model with YAML, deploy with AWS CloudFormation

Has shorthand syntax to express functions, APIs, databases, and event source mappings



https://aws.amazon.com/serverless/sam/
https://github.com/aws/serverless-application-model

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How do I use AWS SAM?



SAM CLI is a tool for local development, testing, deploying, and monitoring of serverless applications

sam init for creating new serverless applications

sam build for generating deployment artifacts

sam package for packaging artifacts and staging artifacts in an Amazon S3 bucket

sam deploy for deploying or updating serverless applications in your AWS account

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Here’s a sample template

```
# APIGW
Api:
  Type: AWS::Serverless::Api
  Properties:
    AccessLogSetting:
      DestinationArn: !GetAtt ApiLogGroup.Arn
      Format: '{ "requestId":"$context.requestId", "ip": "$context.identity.sourceIp", "requestTime":"$context.requestTime",
"httpMethod":"$context.httpMethod","routeKey":"$context.routeKey", "status":"$context.status","protocol":"$context.protocol"}'
    StageName: !Ref pApiStage
    DefinitionBody:
      Fn::Transform:
        Name: AWS::Include
        Parameters:
          Location: openapi.yaml
ApiLogGroup:
  Type: AWS::Logs::LogGroup
  Properties:
    LogGroupName: !Sub "/aws/apigateway/spring-${Api}”
    RetentionInDays: 7

# Lambda
Fn:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: ../target/spring-0.0.1-SNAPSHOT.jar
    Handler: cloud.heeki.spring.SpringHandler::handleRequest
    Role: !GetAtt FnRole.Arn
    AutoPublishAlias: live
    DeploymentPreference:
      Type: AllAtOnce
```

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

And running that function locally

```
sam local invoke -t iac/lambda.yaml --parameter-overrides "ParameterKey=pApiStage,ParameterValue=dev
ParameterKey=pFnMemory,ParameterValue=1024 ParameterKey=pFnTimeout,ParameterValue=60
ParameterKey=pFnVersion,ParameterValue=live ParameterKey=pTableArn,ParameterValue=arn:aws:dynamodb:us-east-
1:123456789012:table/spring-dynamodb-Table-O35Y4TCMLRC6" --env-vars etc/envvars.json -e etc/event.json Fn | jq -r '.body' | jq
```

```
Invoking cloud.heeki.spring.SpringHandler::handleRequest (java11)
Decompressing /Users/heeki/Documents/Isengard/code_spring/lambda_spring/target/spring-0.0.1-SNAPSHOT.jar
arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:21 is already cached. Skipping download
Skip pulling image and use local one: samcli/lambda:java11-x86_64-6158915438a92f4848696f285.

Mounting /private/var/folders/jw/cnlbf_fn65957c9pr7qw492c0000gr/T/tmprcyg4jip as /var/task:ro,delegated inside runtime container
START RequestId: d8703eea-5d61-4e3d-8974-6f51c7553404 Version: $LATEST
20:38:46.405 [main] INFO com.amazonaws.serverless.proxy.internal.LambdaContainerHandler - Starting Lambda Container Handler

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.3)

2022-10-06 20:38:49.818  INFO 18 --- [           main] lambdainternal.AWSLambda                 : Starting AWSLambda using Java 11.0.14.1 on 34da6aab0c61 with PID 18 (/var/runtime/lib/aws-
lambda-java-runtime-0.2.0.jar started by root in /var/task)
2022-10-06 20:38:49.821  INFO 18 --- [           main] lambdainternal.AWSLambda                 : No active profile set, falling back to 1 default profile: "default"
2022-10-06 20:38:52.923  INFO 18 --- [           main] c.a.s.p.i.servlet.AwsServletContext      : Initializing Spring embedded WebApplicationContext
2022-10-06 20:38:52.923  INFO 18 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2909 ms
2022-10-06 20:38:57.644  INFO 18 --- [           main] lambdainternal.AWSLambda                 : Started AWSLambda in 10.082 seconds (JVM running for 13.073)
2022-10-06 20:38:57.656  INFO 18 --- [           main] c.a.s.p.i.servlet.AwsServletContext      : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-10-06 20:38:57.656  INFO 18 --- [           main] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2022-10-06 20:38:57.658  INFO 18 --- [           main] o.s.web.servlet.DispatcherServlet        : Completed initialization in 2 ms
2022-10-06 20:38:57.889  INFO 18 --- [           main] c.a.s.p.internal.LambdaContainerHandler  : 127.0.0.1 null- null [09/04/2015:12:34:56Z] "GET /customer HTTP/1.1" 200 620 "-" "Custom
User Agent String" combined
END RequestId: d8703eea-5d61-4e3d-8974-6f51c7553404
REPORT RequestId: d8703eea-5d61-4e3d-8974-6f51c7553404     Init Duration: 0.60 ms       Duration: 13448.92 ms        Billed Duration: 13449 ms     Memory Size: 1024 MB          Max Memory
Used: 1024 MB
```

Problem: Pulls from public.ecr.aws/sam/emulation-java11

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

That image is pulled from Amazon ECR

```
Image was not found.
Building image.....................................................................
Skip pulling image and use local one: samcli/lambda:java11-x86_64-6158915438a92f4848696f285.
```

But if your organization does not allow public Amazon ECR and only allows internal repositories, like Artifactory:

Solution: Specify alternate repository

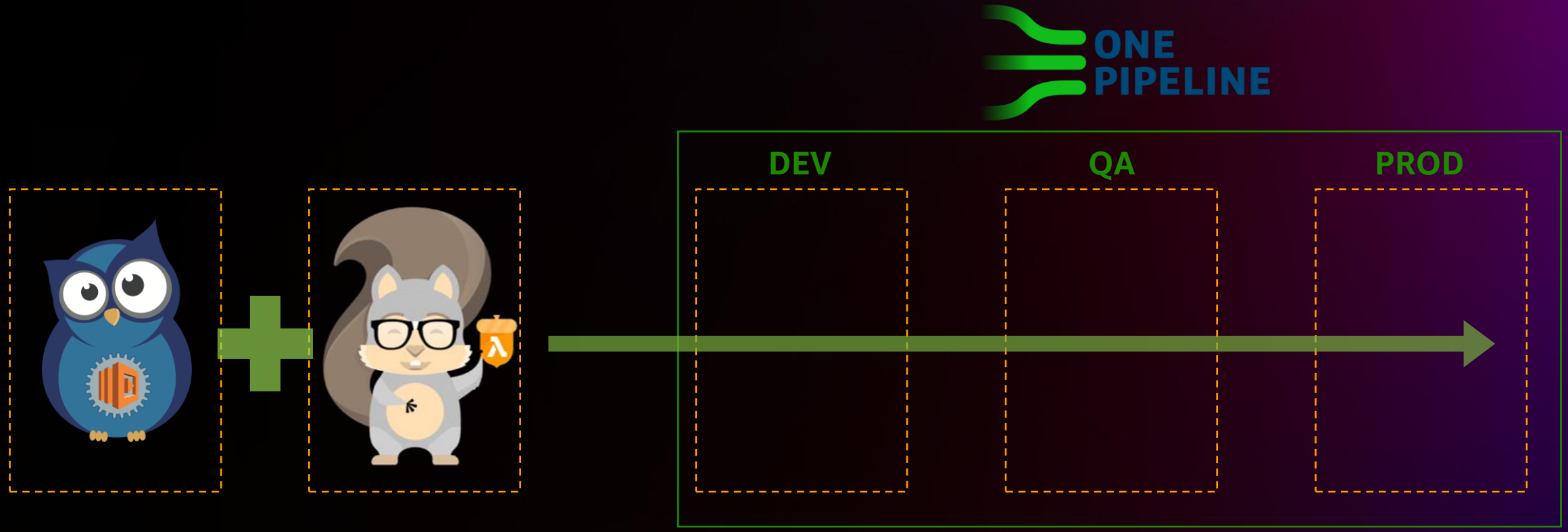
```
sam local invoke -t iac/lambda.yaml
–invoke-image "artifactory.example.com/amazon/lambda:java11-x86_64"
--parameter-overrides "ParameterKey=pApiStage,ParameterValue=dev
ParameterKey=pFnMemory,ParameterValue=1024 ParameterKey=pFnTimeout,ParameterValue=60
ParameterKey=pFnVersion,ParameterValue=live ParameterKey=pTableArn,ParameterValue=arn:aws:dynamodb:us-
east-1:123456789012:table/spring-dynamodb-Table-O35Y4TCMLRC6"
--env-vars etc/envvars.json -e etc/event.json Fn | jq -r '.body' | jq
```

https://s12d.com/aZ6C-RpP

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Capital One Serverless developer experience



C1 Bogie

AWS SAM/CFT

Open Policy Agent Rules (OPA)
Cloud Custodian

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Build

Local developer experience

Deploy

Secrets management

Traffic shifting with AWS CodeDeploy

Run

API resiliency patterns

Resolving errors with provisioned concurrency

Speeding up init with burst CPU

SnapStart overview and testing results

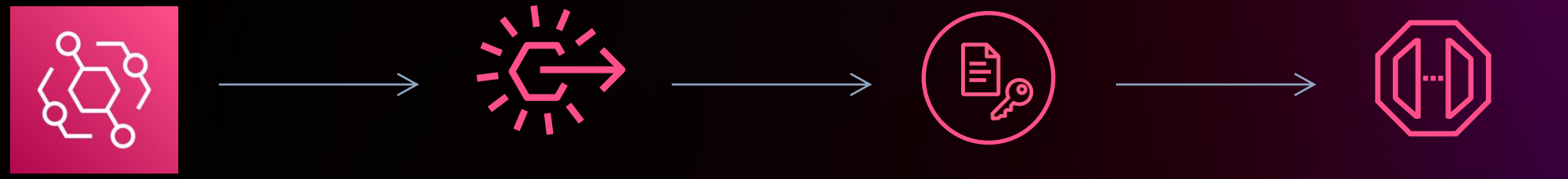
Wrap-up

What to avoid/do with AWS Lambda

Continue your AWS serverless learning

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s look at an example



Problem: Need to safely inject API keys when deploying this resource

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Injecting parameters or secrets at deploy time

Solution: Can use dynamic references

```
{{resolve:ssm:parameter-name:version}}

{{resolve:ssm-secure:parameter-
name:version}}

{{resolve:secretsmanager:secret-id:secret-
string:json-key:version-stage:version-id}}

{{resolve:secretsmanager:arn-for-secret}}
```

https://s12d.com/ATVGE8xn

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How would I implement the connection?

```
SlackConnection:
  Type: AWS::Events::Connection
  Properties:
    AuthorizationType: API_KEY
    AuthParameters:
      ApiKeyAuthParameters:
      ApiKeyName: Authorization
      ApiKeyValue: !Sub
        '{{resolve:secretsmanager:arn:aws:
         secretsmanager:${AWS::Region}:
        ${AWS::AccountId}:secret:${pSecretId}}}
```

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Injecting parameters or secrets at run time

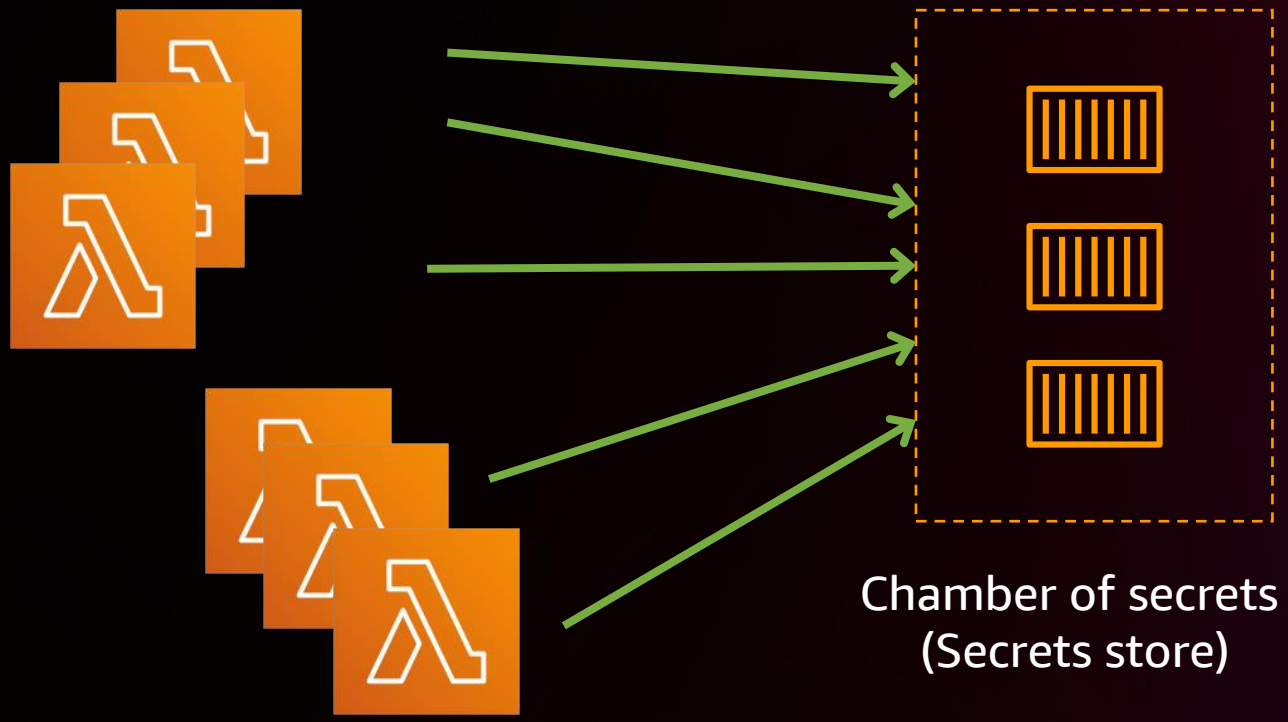
```
Fn:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: src
    Handler: fn.handler
    Role: !GetAtt FnRole.Arn
    Environment:
      Variables:
        SECRETS_MANAGER_TIMEOUT_MILLIS: 1000
        SECRETS_MANAGER_TTL: 60
    Layers:
      - 'arn:aws:lambda:us-east-1:177933569100:layer:
         AWS-Parameters-and-Secrets-Lambda-Extension:2'
```

https://s12d.com/u9YH0uTQ
https://s12d.com/XhoRNohO
https://s12d.com/VSTDv2Rz

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Capital One: Lambda extension for secrets

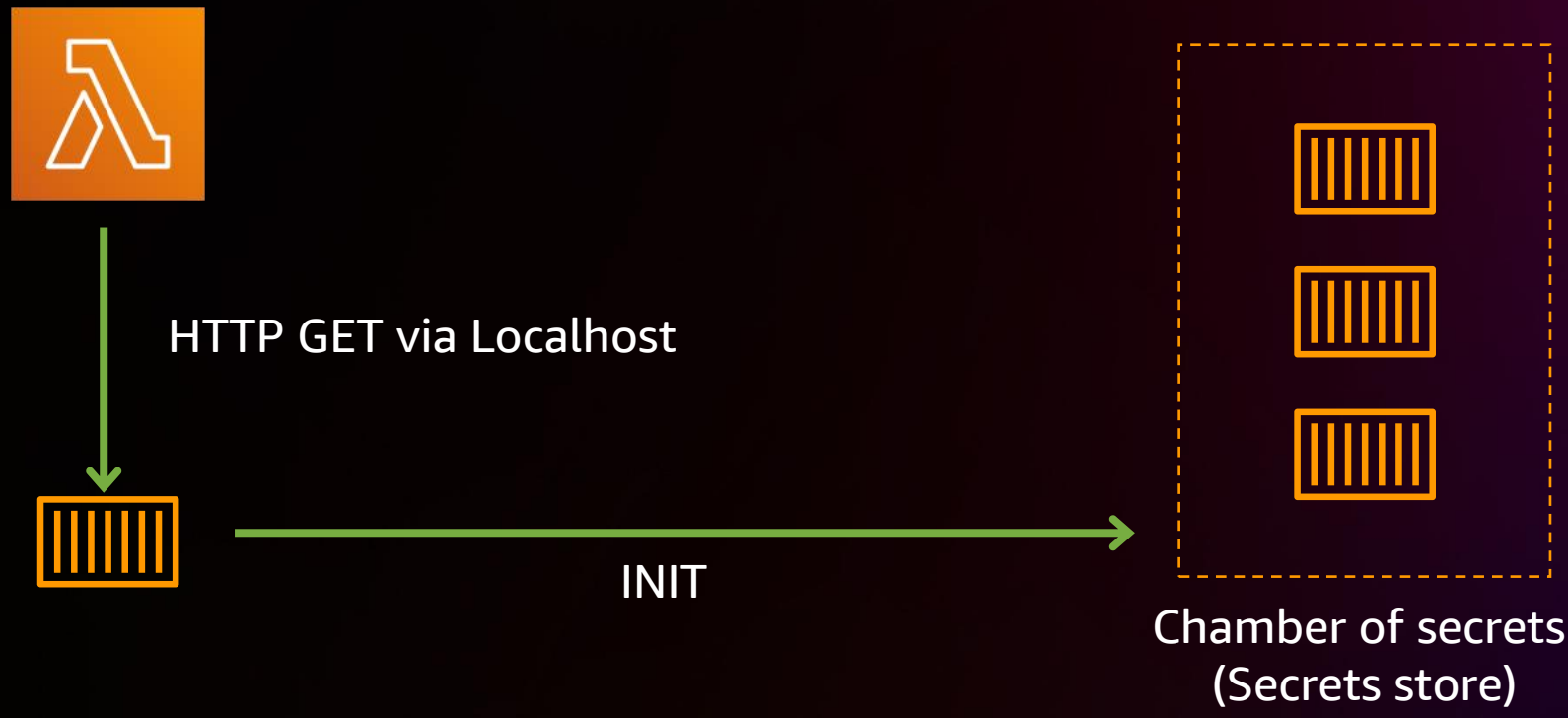
Problem: Lambda can cause massive parallel concurrency on our custom secrets store API



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Capital One: Lambda extension for secrets

Solution: We've created an **extension** that retrieves secrets at INIT and caches. Secrets are served up in a local HTTP server available to the Lambda EE only



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Build

Local developer experience

Deploy

Secrets management

Traffic shifting with AWS CodeDeploy

Run

API resiliency patterns

Resolving errors with provisioned concurrency

Speeding up init with burst CPU

SnapStart overview and testing results

Wrap-up

What to avoid/do with AWS Lambda

Continue your AWS serverless learning

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using CodeDeploy for traffic shifting

```
Fn:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: src
    Handler: fn.handler
    Role: !GetAtt FnRole.Arn
    AutoPublishAlias: LIVE_TRAFFIC
    DeploymentPreference:
      Type: Linear10PercentEvery1Minute
      Hooks:
        PreTraffic: !Ref PreTrafficFn
```

Problem: Performing fine-grained and safe deployments to Lambda

→ Publish an alias for each deployment

→ Shift 10% of the traffic, every minute

→ Reference to a function to validate that traffic shifting activity is ok to proceed

https://s12d.com/qxQTajaX

aws

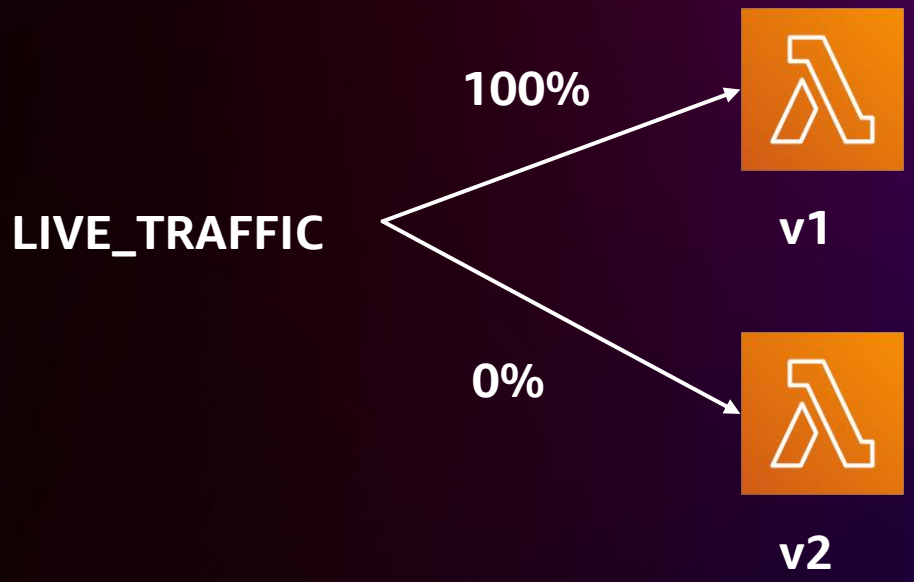
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Roll out code changes safely with aliases

Use AWS CodeDeploy to move traffic from your old version to the new one
Rollback to previous version if errors are detected during the rollout

	Description
LambdaLinear10PercentEvery1Minute	Move 10% every 1 minute
LambdaCanary10Percent5Minutes	Move 10%, then the remaining 90% after 5 minutes
LambdaAllAtOnce	Move all traffic

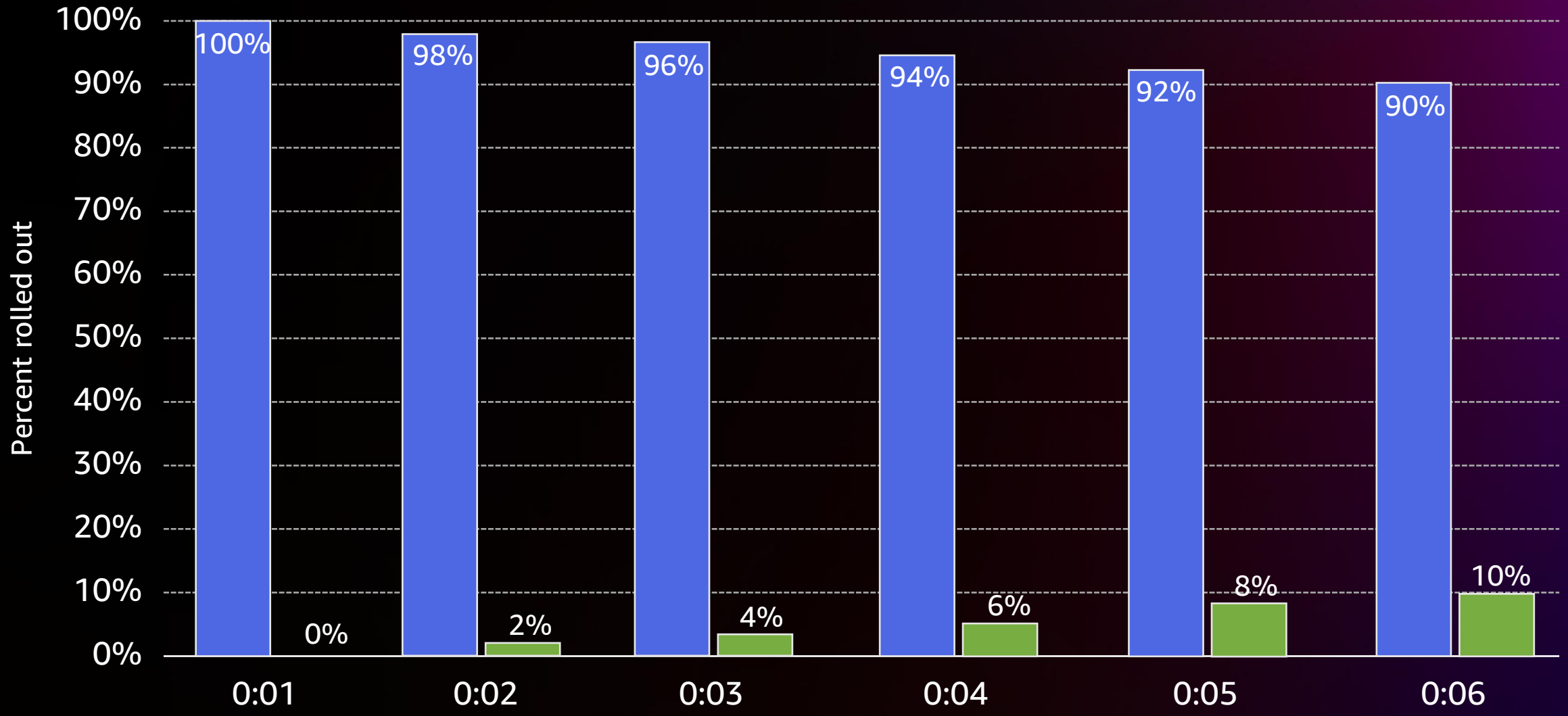
Move **X%** of traffic every **Y** minutes to the new version, until **100%** rolled out



Solution: Use linear or canary rollouts. However . . .

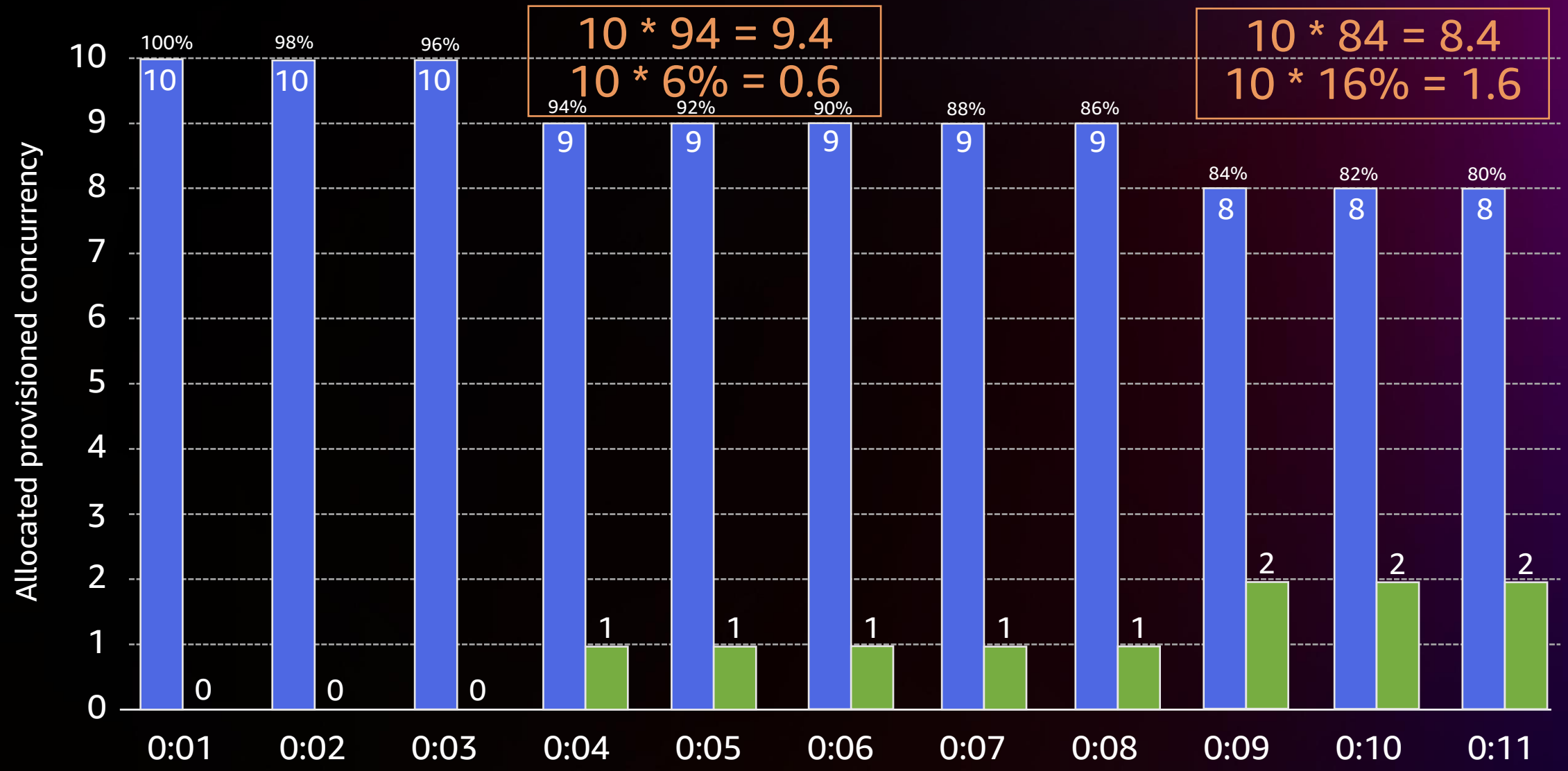
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Capital One: Linear rollout over 45 minutes



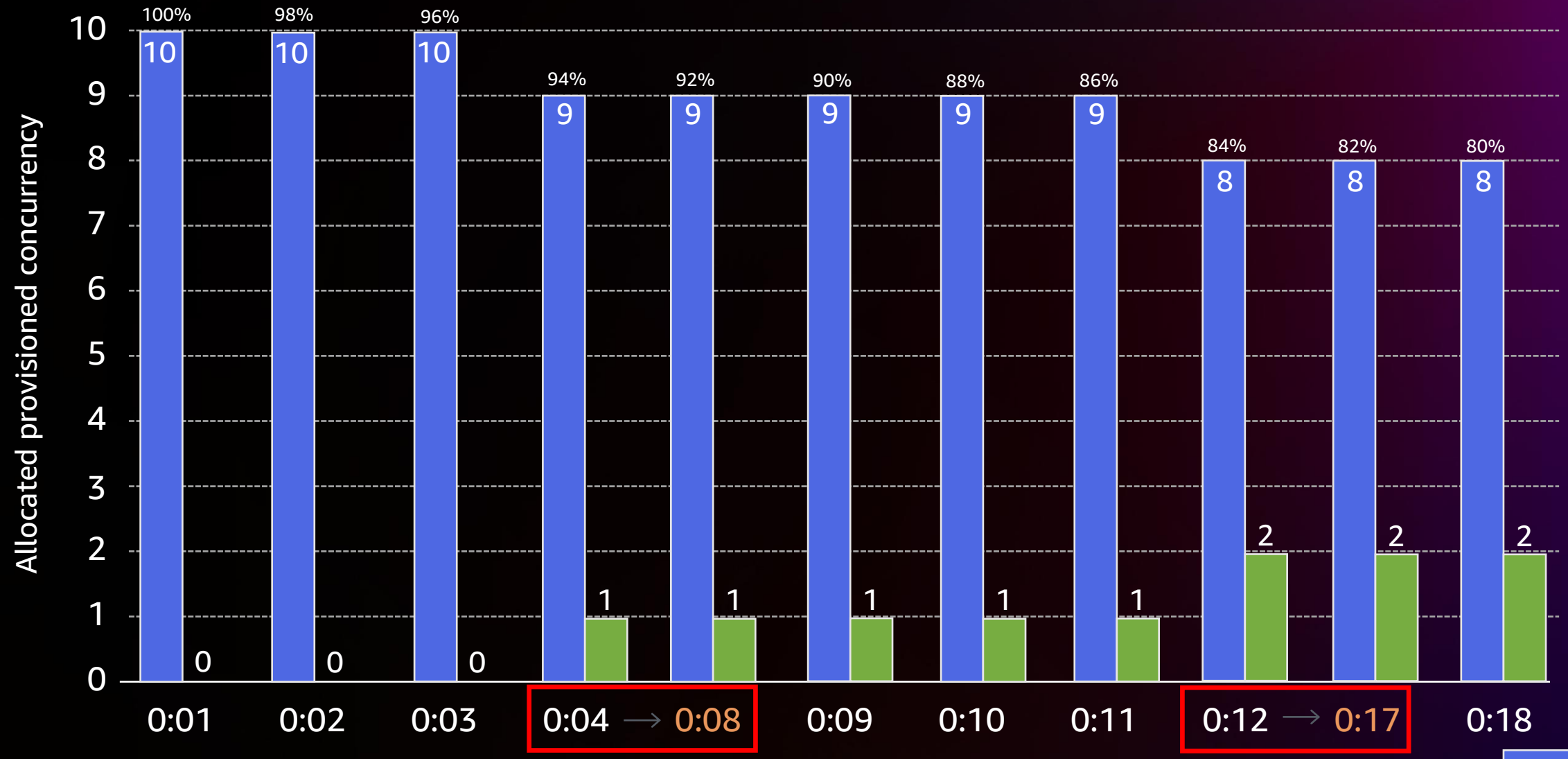
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Capital One: Linear rollout with PC (maybe)



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Capital One: Linear rollout with PC (actual)

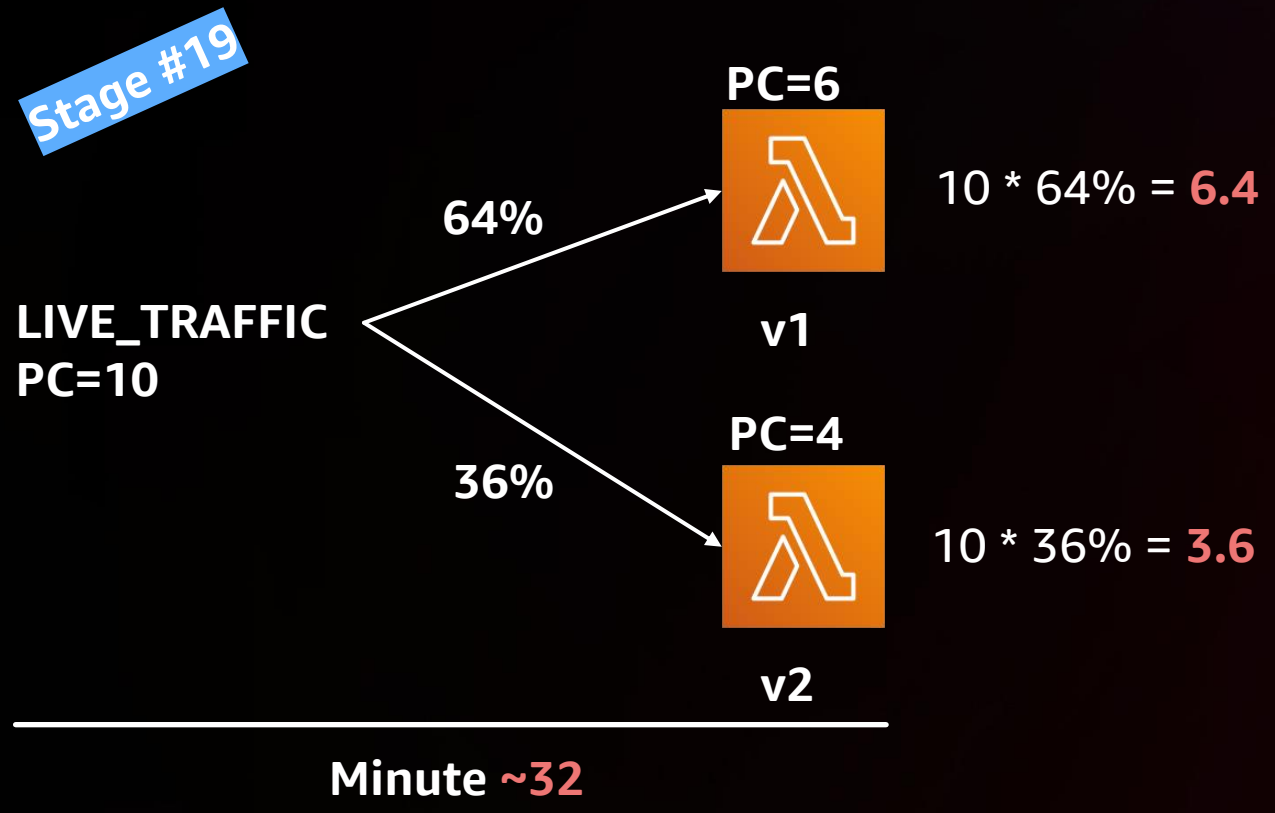


PC takes time to provision

v1 v2

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Capital One: Linear rollout with PC



Why does this matter?

1. Developer confusion
2. Unexpected cold starts
3. You may miss your rollout target!

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Build

Local developer experience

Deploy

Secrets management
Traffic shifting with AWS CodeDeploy

Run

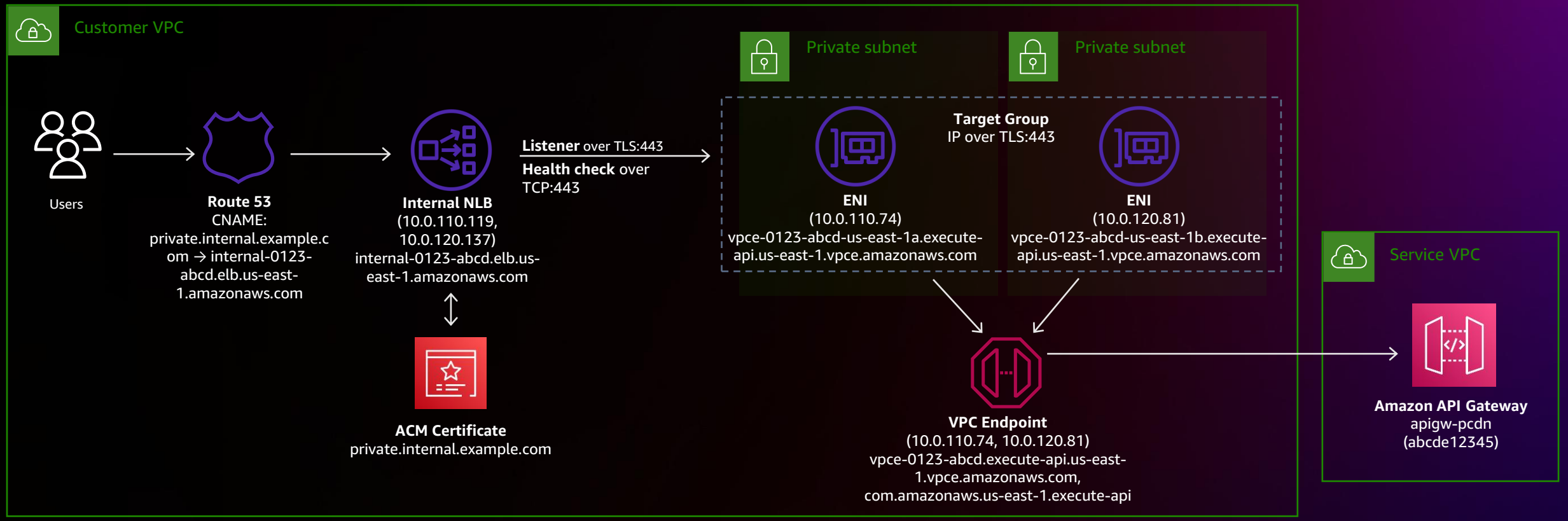
API resiliency patterns
Resolving errors with provisioned concurrency
Speeding up init with burst CPU
SnapStart overview and testing results

Wrap-up

What to avoid/do with AWS Lambda
Continue your AWS serverless learning

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

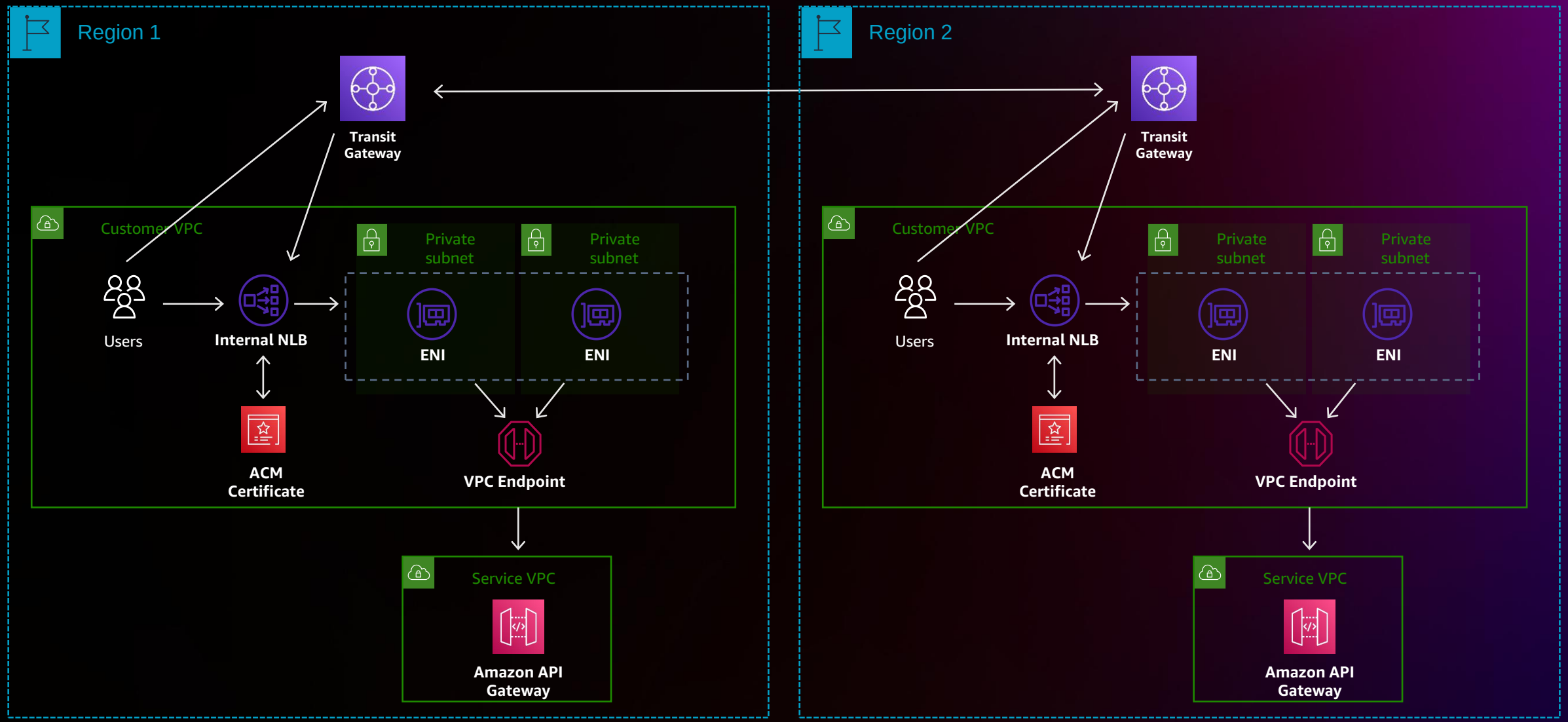
Enabling private API endpoints with custom domain names



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

https://s12d.com/IgbMy9Zq

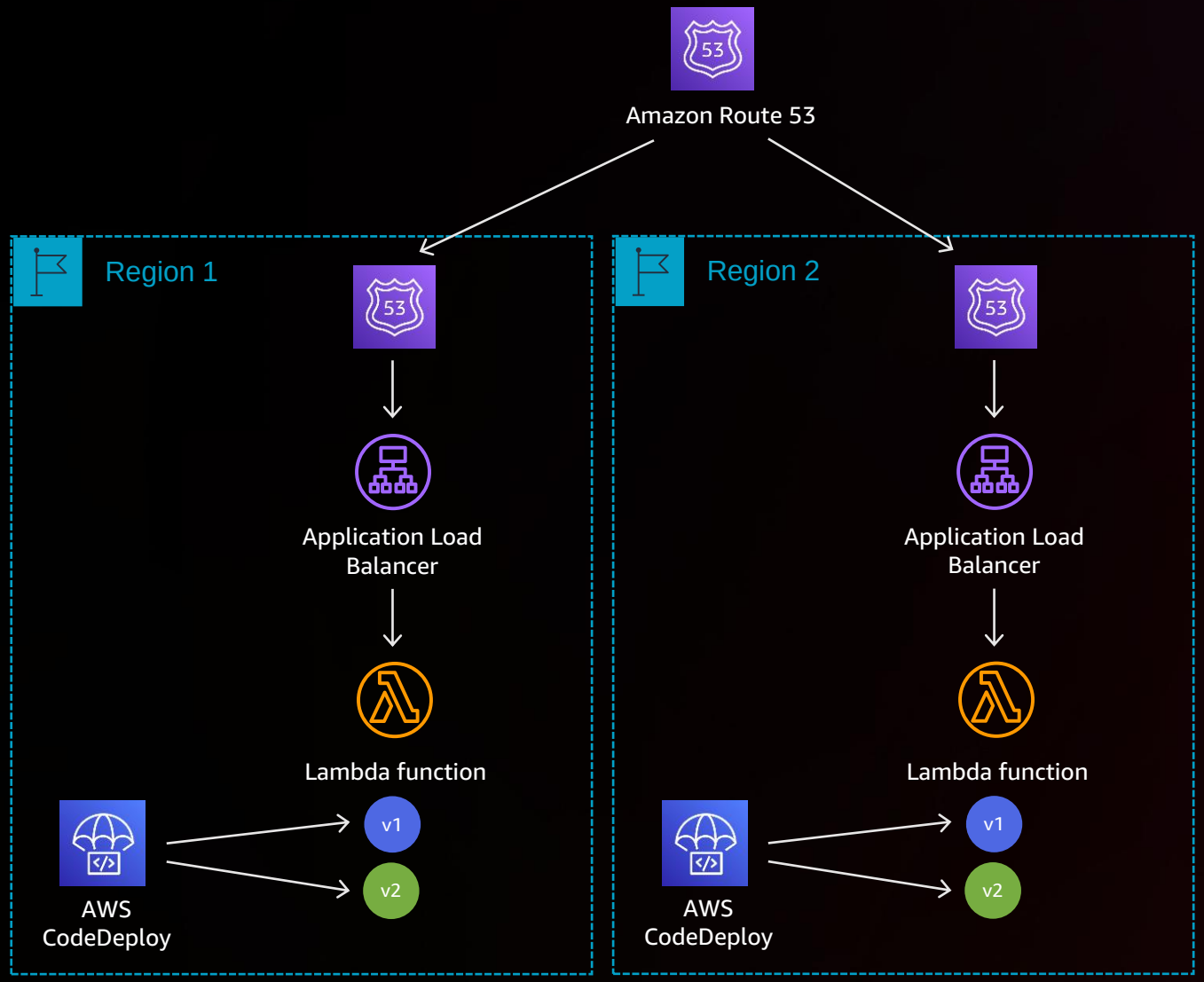
Building multi-Region private APIs



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

https://s12d.com/wPi0xWHL

Capital One API architecture



- Active/active **dual** Region resiliency
- Auto failover
- Phased rollouts with rollback ability
- Route 53 CNames
- Application Load Balancer
- Lambda
- CodeDeploy
- Standard Lambda Alias: LIVE_TRAFFIC

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Build

Local developer experience

Deploy

Secrets management
Traffic shifting with AWS CodeDeploy

Run

API resiliency patterns
Resolving errors with provisioned concurrency
Speeding up init with burst CPU
SnapStart overview and testing results

Wrap-up

What to avoid/do with AWS Lambda
Continue your AWS serverless learning

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Errors in your INIT phase == Stuck PC

index.js | Execution results

```
1  // Simulate an error in the Lambda INIT Phase
2  throw new Error("This is an INIT ERROR");
3
4  exports.handler =  async function(event, context) {
5    console.log("EVENT: \n" + JSON.stringify(event, null, 2))
6
7    console.log("Entering via ALB!!");
8
9    let albResponse = {
10     statusCode: 200,
11     statusDescription: "200 OK",
12     isBase64Encoded: false,
13     headers: {
14         "Content-Type": "text/html"
15     },
16     body: "<h1>Hello from Lambda!</h1>"
17   }
18
19   return albResponse;
20 }
```

→ This results in **0 PC** allocated!
→ All requests will be served on demand
→ AWS will NOT retry
→ Release a new version to fix this

Provisioned concurrency | Edit | Remove

Provisioned concurrency	Status
0	Failed FUNCTION_ERROR_INIT_FAILURE(0/1)

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Build	Local developer experience
Deploy	Secrets management Traffic shifting with AWS CodeDeploy
Run	API resiliency patterns Resolving errors with provisioned concurrency Speeding up init with burst CPU SnapStart overview and testing results
Wrap-up	What to avoid/do with AWS Lambda Continue your AWS serverless learning

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How does CPU burst work?

On-demand (First attempt)	 832 MB (memory)	 1 vCPU (unthrottled)	 10 seconds
On-demand (Second attempt)	 832 MB (memory)	 0.5 vCPU (proportional)	 (configured timeout)
PC (First attempt)	 832 MB (memory)	 0.5 vCPU (proportional)	 (configured timeout)

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Lambda INIT phase aka “cold start” is special

Lambda gives you CPU power in proportion to the **memory** allocation you select

Double your memory = Double your CPU power



TLDR: You get full unthrottled CPU during INIT . . . SHHhhhhhhh

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Compute-intensive sample code

```
m = hashlib.sha512()
# Generate a random long string
rstr = ''.join(random.choice(string.ascii_uppercase + string.digits)
                                                    for _ in range(N))


t1 = datetime.now()
print("Start time: " + str(t1))

# repeatedly compute the digest
for i in range(10000):
    m.update(rstr.encode())
    m.digest()

t2 = datetime.now()
print("End time: " + str(t2))
print("Difference: " + str(t2 - t1))
```

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Test #1: Memory at 512 MB

```
Function Logs
INIT Start time: 2022-10-20 19:34:52.043569
INIT End time: 2022-10-20 19:34:52.978530
Difference: 0:00:00.934961
START RequestId: 3b32aeb6-bd10-4f18-8082-86d89eb77f71 Version: $LATEST
INVOKE Start time: 2022-10-20 19:34:53.111385
INVOKE End time: 2022-10-20 19:34:56.232900
Difference: 0:00:03.121515
Handler complete
END RequestId: 3b32aeb6-bd10-4f18-8082-86d89eb77f71
REPORT RequestId: 3b32aeb6-bd10-4f18-8082-86d89eb77f71  Duration: 3253.51 ms
```

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Test #2: Memory at 1024 MB

```
Function Logs
INIT Start time: 2022-10-20 19:39:20.882747
INIT End time: 2022-10-20 19:39:21.764456
Difference: 0:00:00.881709
START RequestId: 13719048-e658-4366-ba70-6984e1f6659f Version: $LATEST
INVOKE Start time: 2022-10-20 19:39:21.828651
INVOKE End time: 2022-10-20 19:39:23.355386
Difference: 0:00:01.526735
Handler complete
END RequestId: 13719048-e658-4366-ba70-6984e1f6659f
REPORT RequestId: 13719048-e658-4366-ba70-6984e1f6659f  Duration: 1589.06 ms
```

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Test #3: Memory at 2048 MB

```
Function Logs
INIT Start time: 2022-10-20 19:41:15.023281
INIT End time: 2022-10-20 19:41:15.907825
Difference: 0:00:00.884544
START RequestId: 8c7d2bee-07ff-44b5-8fcd-66cb9c7d29fa Version: $LATEST
INVOKE Start time: 2022-10-20 19:41:15.951886
INVOKE End time: 2022-10-20 19:41:16.833448
Difference: 0:00:00.881562
Handler complete
END RequestId: 8c7d2bee-07ff-44b5-8fcd-66cb9c7d29fa
REPORT RequestId: 8c7d2bee-07ff-44b5-8fcd-66cb9c7d29fa  Duration: 924.62 ms
```

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Test #4: Memory at 3000 MB

```
Function Logs
INIT Start time: 2022-10-20 19:42:15.987440
INIT End time: 2022-10-20 19:42:16.883435
Difference: 0:00:00.895995
START RequestId: ba130731-096e-42de-969f-81b6d34505ae Version: $LATEST
INVOKE Start time: 2022-10-20 19:42:16.939871
INVOKE End time: 2022-10-20 19:42:17.859299
Difference: 0:00:00.919428
Handler complete
END RequestId: ba130731-096e-42de-969f-81b6d34505ae
REPORT RequestId: ba130731-096e-42de-969f-81b6d34505ae  Duration: 978.33 ms
```

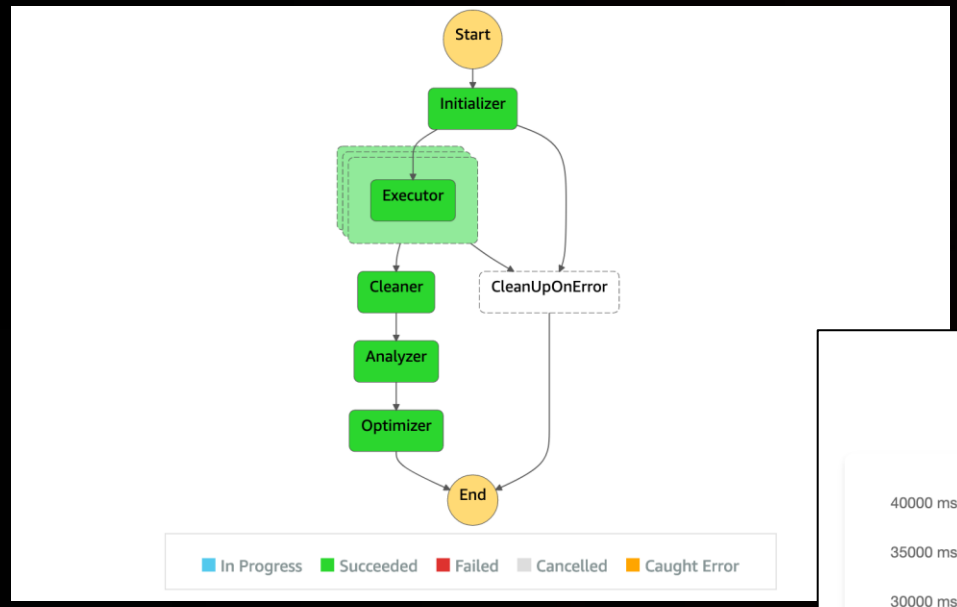
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

INIT summary

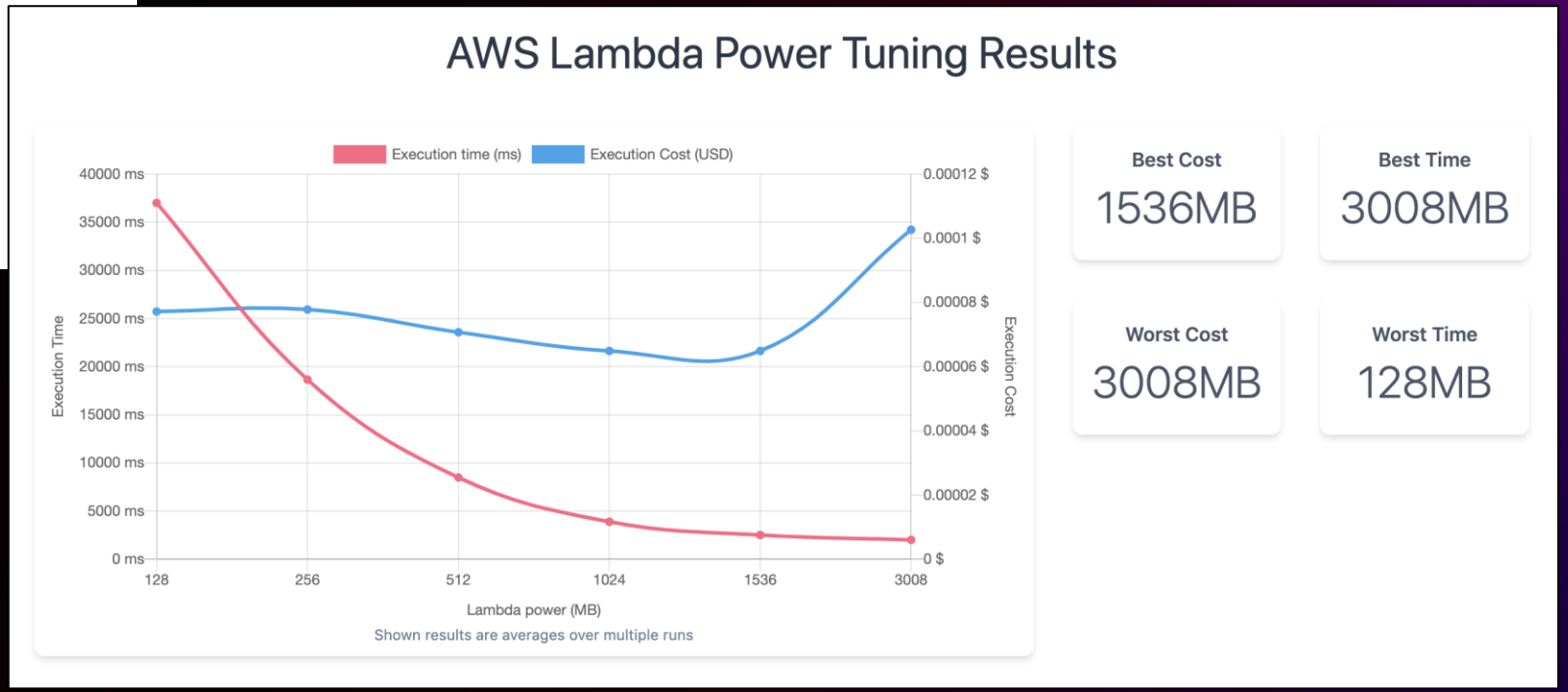
1. The INIT phase provides you with full, CPU unthrottled
2. Do your heavy lifting there, if you can
3. If your INIT exceeds 10 seconds, AWS will terminate it and retry inline with the INVOKE
4. Provisioned concurrency does NOT provide INIT with unthrottled CPU

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Lambda Power Tuning



A state machine powered by AWS Step Functions that helps you optimize your Lambda functions for cost and/or performance in a data-driven way



https://github.com/alexcasalboni/aws-lambda-power-tuning

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

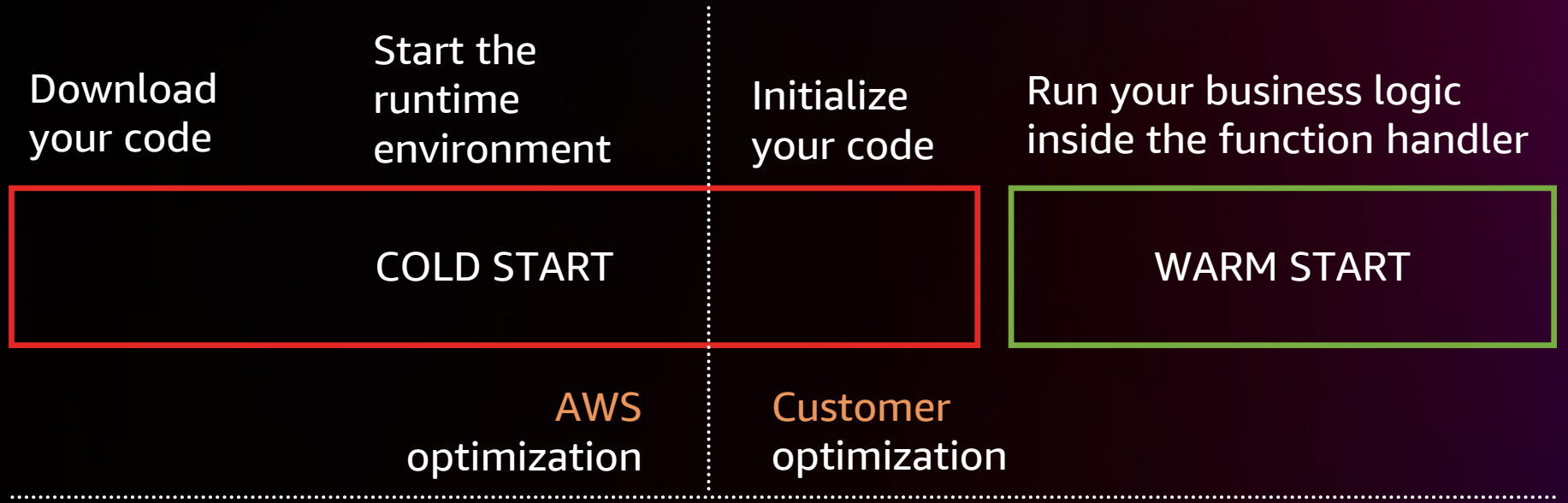
Build	Local developer experience
Deploy	Secrets management Traffic shifting with AWS CodeDeploy
Run	API resiliency patterns Resolving errors with provisioned concurrency Speeding up init with burst CPU SnapStart overview and testing results
Wrap-up	What to avoid/do with AWS Lambda Continue your AWS serverless learning

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

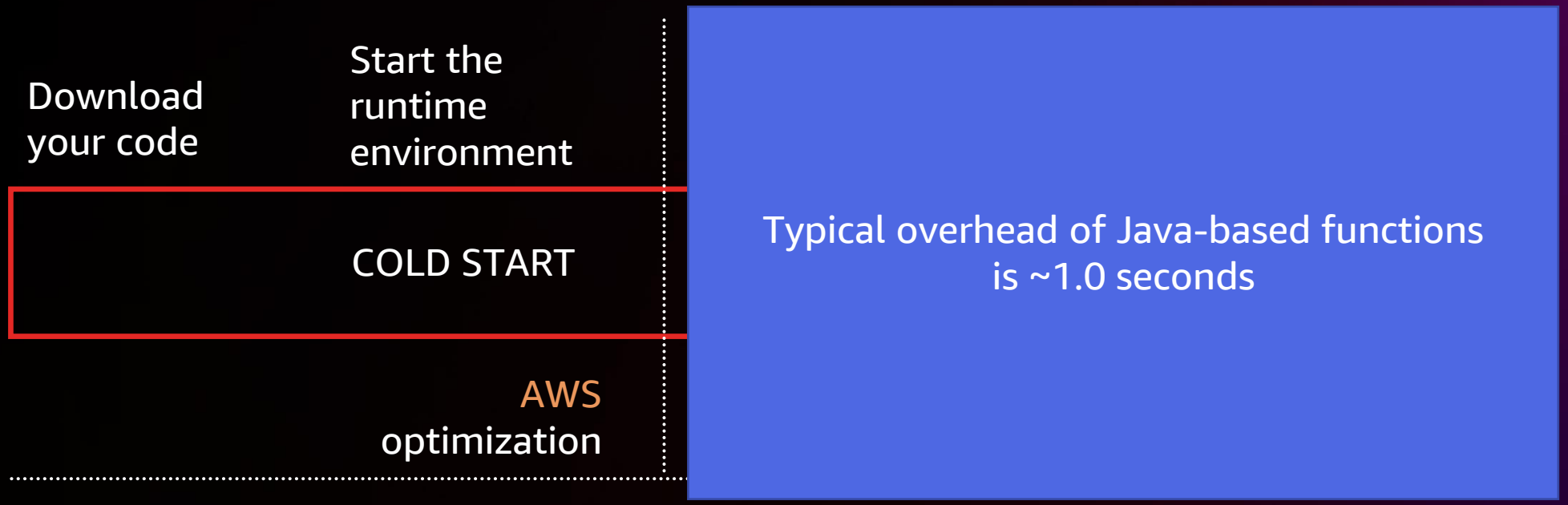
First, why SnapStart?

Problem: I have Java-based applications that I want to run on Lambda, but initialization is slow



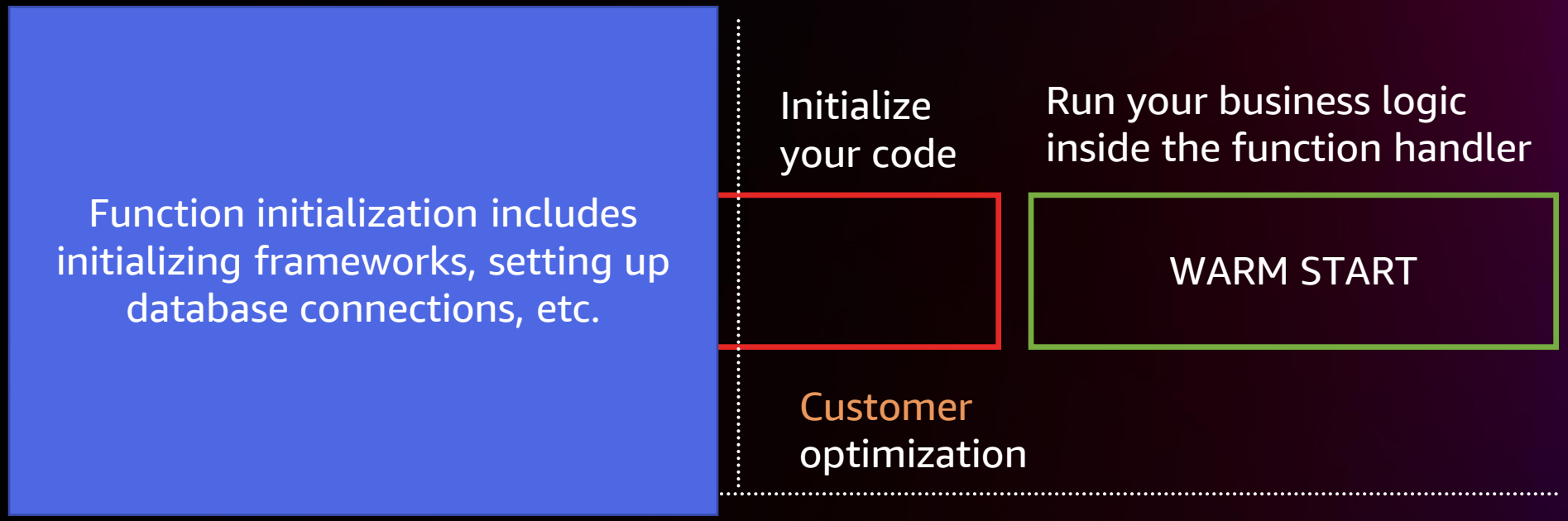
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

First, why SnapStart?



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

First, why SnapStart?



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Cold start optimization strategies

Provisioned concurrency

- Creates a configured number of execution environments ahead of time and always running, i.e., hyper-ready to respond to invocations in milliseconds
- Greatly improved latency consistency across the full long tail of performance
- Integrated with AWS Auto Scaling

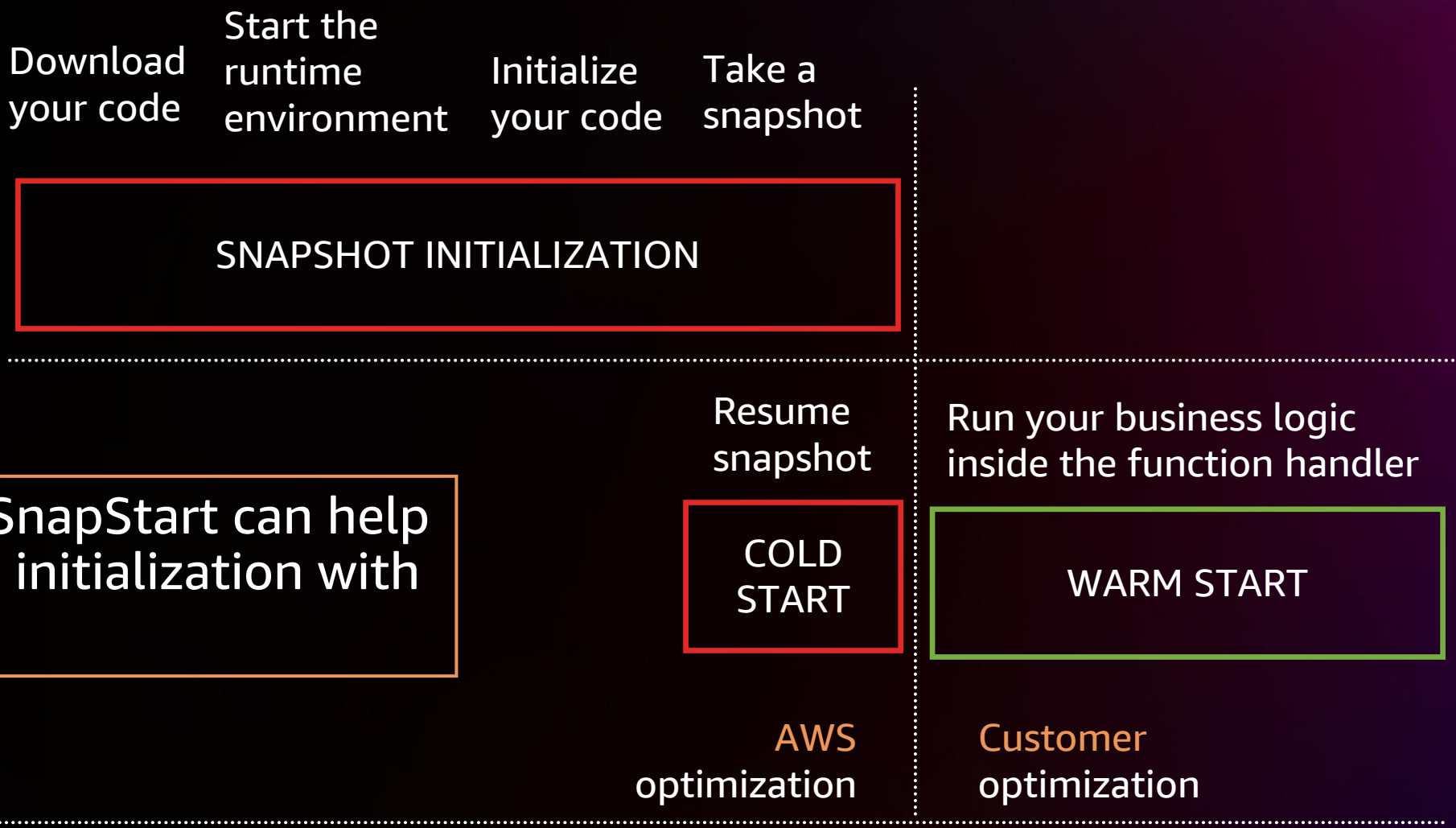
Ahead of time compilation

- Frameworks such as Quarkus and Micronaut integrate GraalVM (which creates a native image of Java applications at build-time) to achieve better cold start performance

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What is SnapStart?



Solution: SnapStart can help accelerate initialization with snapshots

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Hands-on testing with SnapStart

Variety of use cases

- Spring Boot API with Spring Web
- Spring Boot stream processing with Spring Cloud Stream
- Java API with Dagger and netty

Highlights

- Amazon Coretto 11
- No code changes required (beware PRNGs)
- 1,630–15,000 ms reduction in cold-start latency (~88% on average)
- New opportunities for serverless migrations of established applications
- Shifting application initialization left using SnapStart can reduce GB/sec Lambda costs

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Hands-on testing with SnapStart

Application	Avg Cold Start (ms)			
	Pre-SnapStart	Post-SnapStart	Difference	% Improvement
Kotlin API application with Netty, Dagger	2199	565	1634	74.31%
Spring Boot Web API application with Postgres JPA	13000	660	12340	94.92%
Java API application with Netty, Dagger	6467	5489	978	15.12%
Spring Boot Web API application	9532	819	8713	91.41%
Spring Cloud Stream application	16000	933	15067	94.17%

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why is my credential provider failing to obtain credentials with SnapStart?

The credential provider chain uses a chain of providers. For Lambda, there are three options

Will not work!

```
.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
.credentialsProvider(DefaultCredentialsProvider.create())
.credentialsProvider(ContainerCredentialsProvider.create())
```

Preferred

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Is there any environment variable to indicate whether the current invocation is resumed from a snapshot?

`AWS_LAMBDA_INITIALIZATION_TYPE` will be set to "snap-start" if resumed from a snapshot

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why am I not seeing improvement with my function with SnapStart enabled?

SnapStart is enabled by version. Be sure that the function version that is being invoked has SnapStart enabled.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How does the snapshot work with database connections?

If you have network connections that were established during initialization, those connections will need to be refreshed

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How does the snapshot work with random number generation?

If you have applications that use random number generation, e.g., UUIDs, cryptography, you need to evaluate if the application is "snap safe"

Specifically, using `java.security.SecureRandom` and OpenSSL's `RAND_bytes()` or reading from `/dev/{u}random` are snap safe

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Build

Local developer experience

Deploy

Secrets management
Traffic shifting with AWS CodeDeploy

Run

API resiliency patterns
Resolving errors with provisioned concurrency
Speeding up init with burst CPU
SnapStart overview and testing results

Wrap-up

What to avoid/do with AWS Lambda
Continue your AWS serverless learning

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Don’t do these. We have experience ☺

1. Maxed out Lambda configurations (15-minute timeout, 10GB memory)
2. Reserved concurrency == provisioned concurrency
3. Over-allocate provisioned concurrency to peak traffic
4. Setting reserved concurrency on every function
5. Direct synchronous Lambda to Lambda calls
6. Amazon SQS or Amazon Kinesis → Lambda batch sizes = 1
7. ~~Java: Using Spring Boot for real-time Lambdas~~
8. Lambda code that is primarily orchestrating calls and high idle time

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

These are best practices. Follow them!

1. Set Amazon CloudWatch Log retention policy
2. Optimize your memory configuration
3. Optimize your CPU architecture (x86 vs arm64)
4. Optimize code for the cold/warm cycle
5. Estimate your serverless costs
6. Java: Apply JVM tuning options
7. Minimize function deployment size/dependencies
8. Review function performance metrics and logs

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“With great power, comes great responsibility.”

Uncle Ben

Spiderman

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

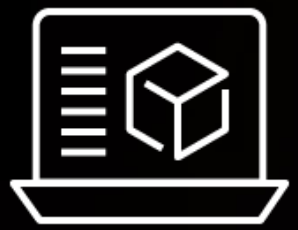
Build	Local developer experience
Deploy	Secrets management Traffic shifting with AWS CodeDeploy
Run	API resiliency patterns Resolving errors with provisioned concurrency Speeding up init with burst CPU SnapStart overview and testing results
Wrap-up	What to avoid/do with AWS Lambda Continue your AWS serverless learning

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Continue your AWS Serverless learning

Learn at your own pace



Expand your serverless skills with our Learning Plan on **AWS Skill Builder**

Increase your knowledge



Use our **Ramp-Up Guides** to build your serverless knowledge

Earn AWS Serverless badge



Demonstrate your Knowledge by achieving **digital badges**

https://s12d.com/serverless-learning

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Check out these other sessions!

SVS303 – Building real-world serverless applications with AWS SAM

SVS308 – AWS serverless developer experience: A day in the life of a developer

SVS314 – AWS Lambda performance tuning: best practices and guidance

SVS318 – Building modern API architectures: Which front door should I use?

SVS401 – Best practices for advanced serverless developers

SVS404 – A closer look at AWS Lambda

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

George Mao
george.mao@capitalone.com

Heeki Park
heeki@amazon.com



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.