
AWS
re:Invent

ARC208

Is your serverless application well-architected?

Hernan Garcia

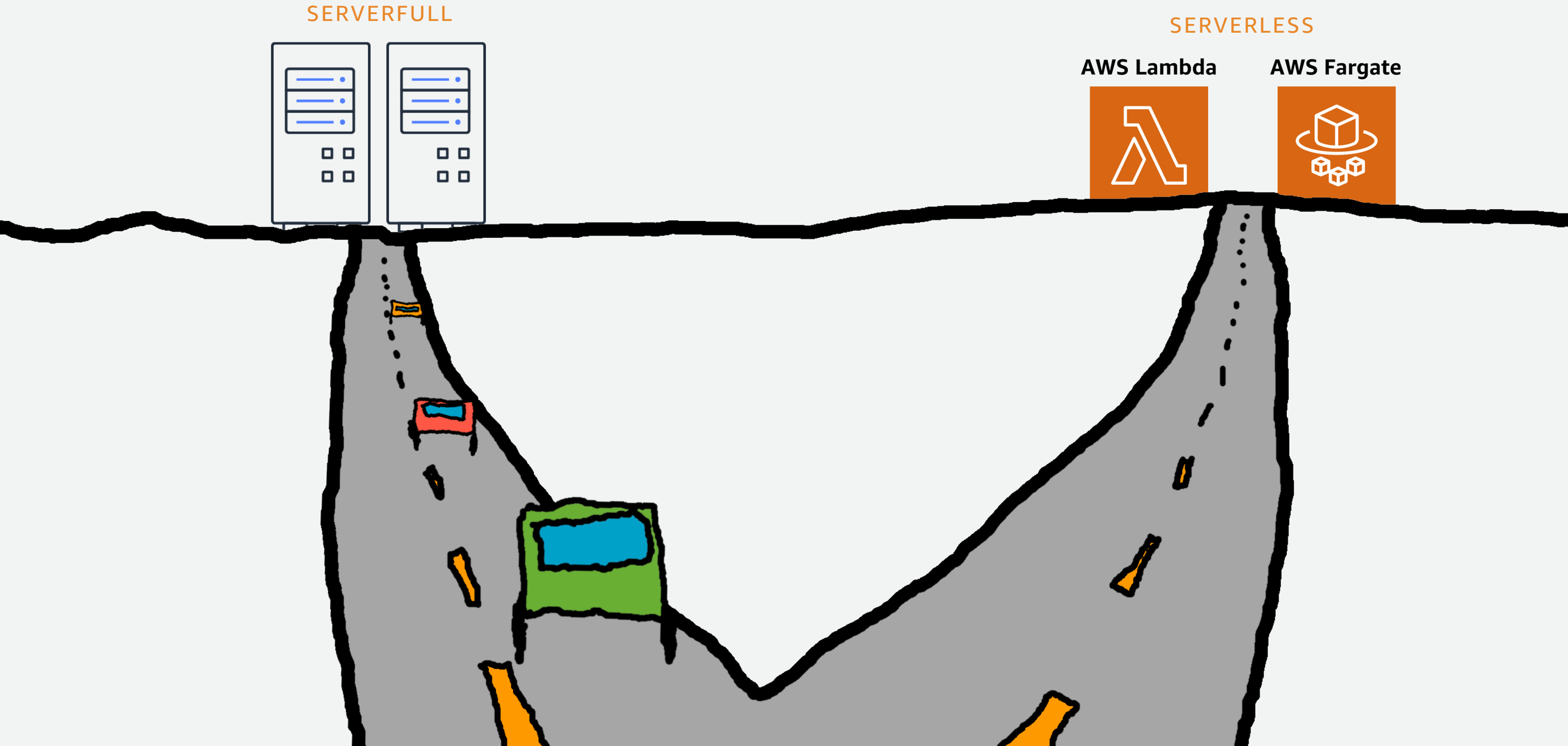
Solutions Architect

AWS

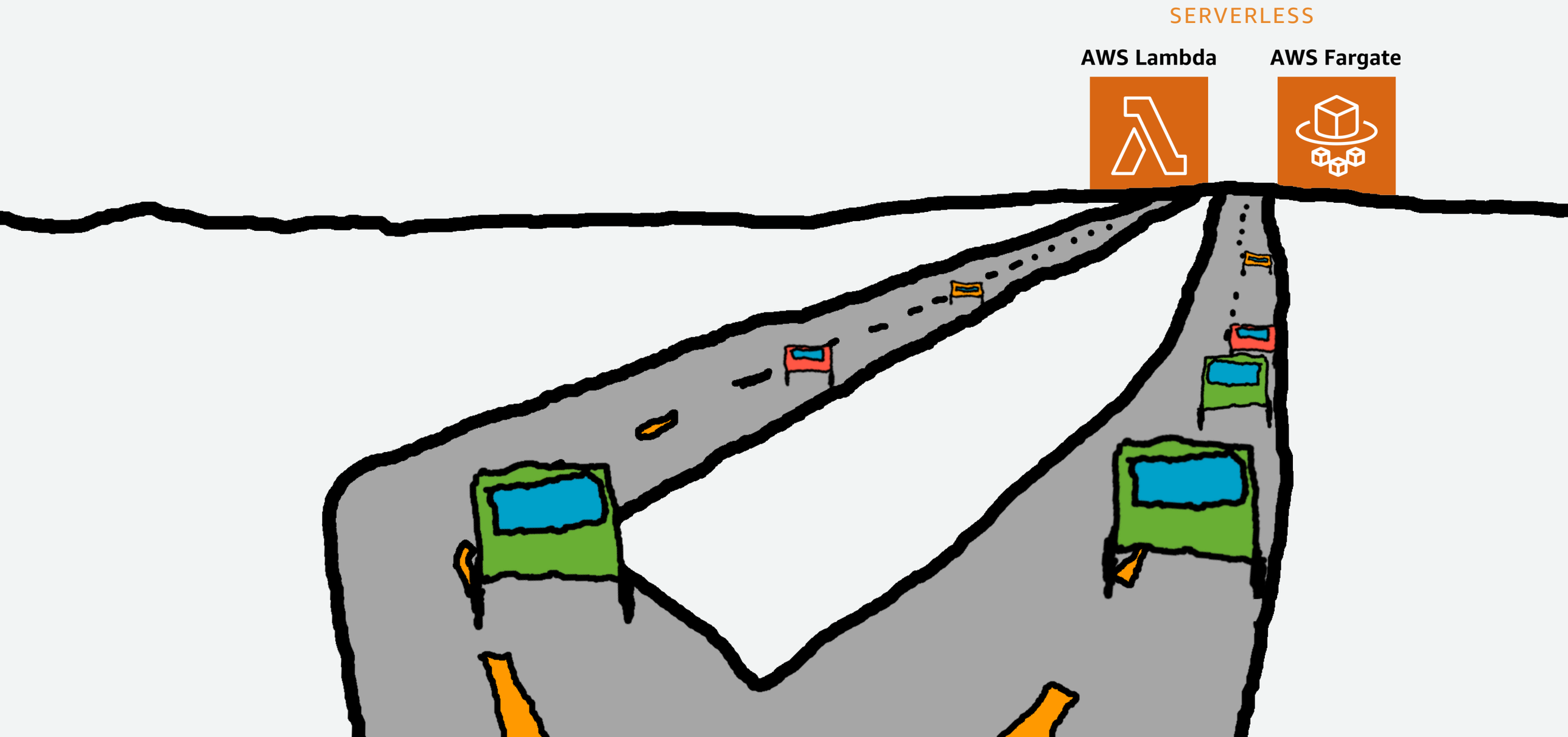
AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“The road less traveled”



All roads lead to serverless!



Why are you watching this?

You need to create value for your organization and customers faster

You want to use serverless services for the task

You want to understand which serverless services can be used

You want to understand the operational best practices for serverless services



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

There’s an improved state of mind for building in the cloud



Rick Houlihan

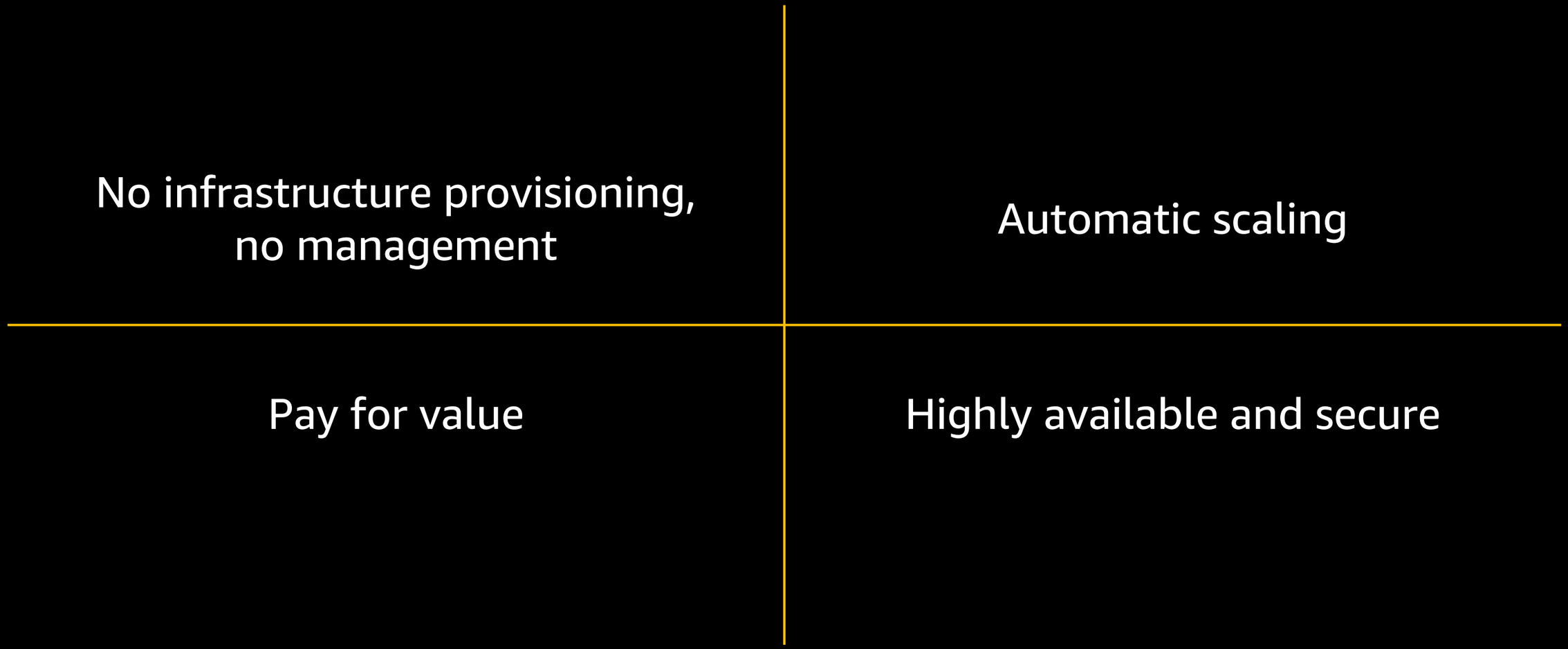
Serverless means:

1. Never having to patch another OS.
2. Never being paged when a rack goes down.
3. Never needing to rollout software on a hundred nodes.
4. Never paying for idle infrastructure.
5. Never losing business because you can't scale.



Serverless = Organizational nirvana

Serverless removes the undifferentiated heavy lifting



Let’s go back to cloud fundamentals

TO THRIVE IN THIS NEW WORLD

Are you well-architected?

Meet and embrace the Well-Architected pillars

aws Well-Architected

OPERATIONAL EXCELLENCE



SECURITY



RELIABILITY



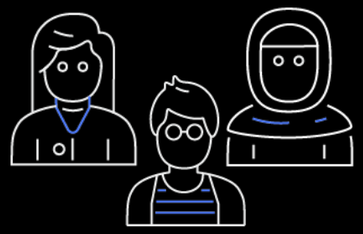
PERFORMANCE EFFICIENCY



COST OPTIMIZATION



We can walk you through this path



Your team

+

(aws Solutions Architect

or

aws AWS Partner)

=

1



Review

2



Report findings

3



Action plan

4



Masterpiece



Beware of the old guard

The truth is …

- It is an objective, constructive conversation
- You will learn AWS best practices
- You can leverage it to make informed architectural decisions
- You will lower or mitigate risks
- You will build and deploy faster





When to use serverless?



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverless is there for you, at every layer

Compute & access	AWS Lambda	AWS Lambda@Edge	AWS Fargate	Amazon API Gateway	AWS Step Functions
Storage	Amazon DynamoDB	Amazon Aurora Serverless	Amazon RDS Proxy	Amazon S3	Amazon ES
Messaging & streaming	Amazon SNS	Amazon SQS	Amazon Kinesis Data Streams	Amazon Kinesis Data Firehose	Amazon AppSync	Amazon EventBridge

Serverless is there for you, at every layer

 User & identity management	 Amazon Cognito	 AWS Identity and Access Management (IAM)
 Delivery at the edge	 Amazon CloudFront	 Amazon API Gateway Edge-optimized
 Analytics	 Amazon Kinesis Data Analytics	 Amazon Athena

Serverless is there for you, at every layer



Provisioning & monitoring



AWS SAM



AWS Amplify



Amazon CloudWatch



AWS X-Ray



AWS CDK



Development tools



AWS CodeCommit



AWS CodeBuild



AWS CodeDeploy



AWS CodePipeline



AWS CodeArtifact

Meet the well-architected serverless applications lens

PRESCRIPTIVE GUIDANCE FOR SERVERLESS APPLICATIONS

aws Well-Architected Serverless Lens



Design and architect + Deploy + Operate = Better and faster



aws Well-Architected

OPERATIONAL EXCELLENCE



SECURITY



RELIABILITY



PERFORMANCE EFFICIENCY



COST OPTIMIZATION

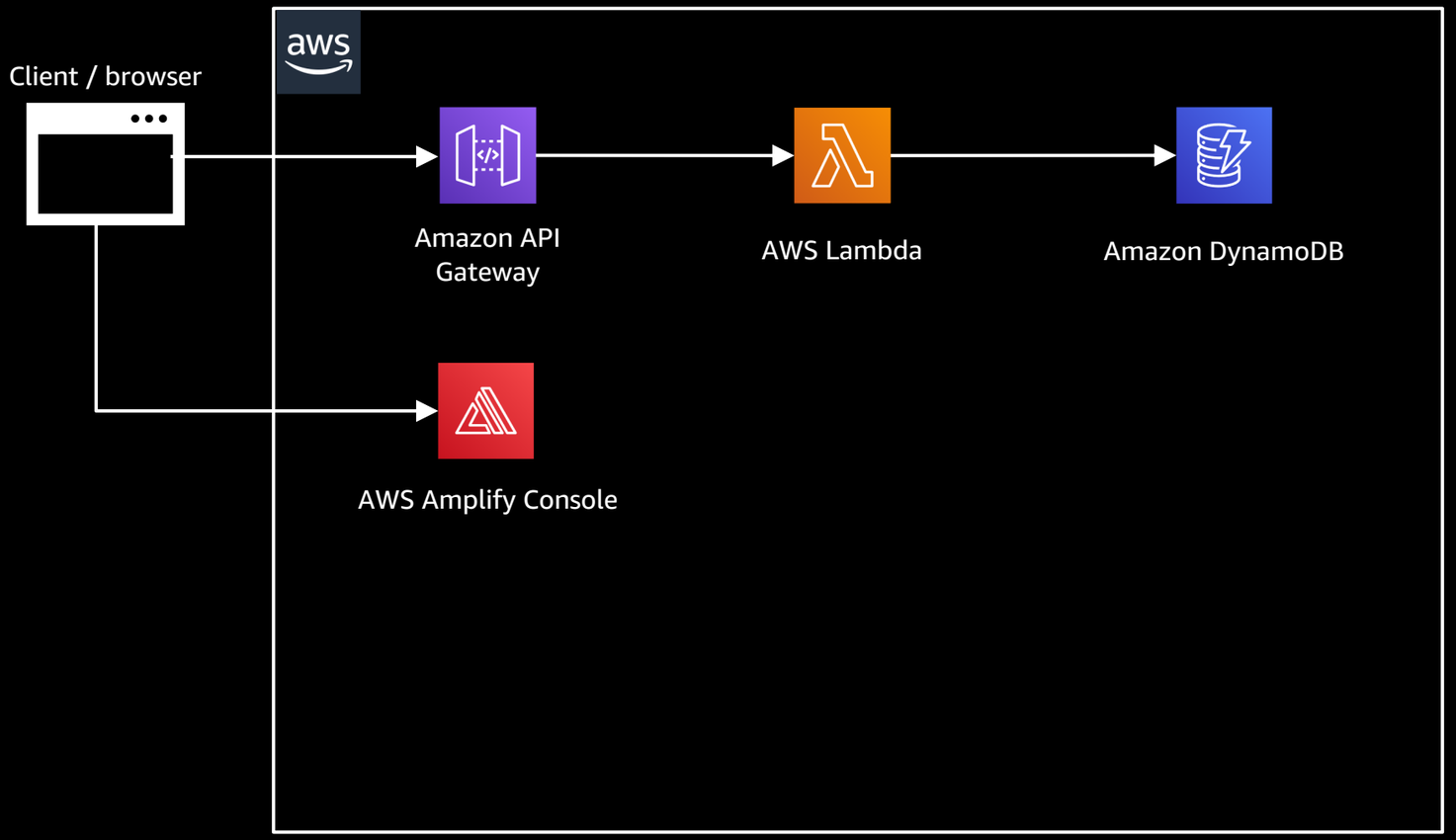


Common customer scenarios



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverless web applications



OPERATIONAL EXCELLENCE

SECURITY

RELIABILITY

PERFORMANCE EFFICIENCY

COST OPTIMIZATION

Serverless web applications

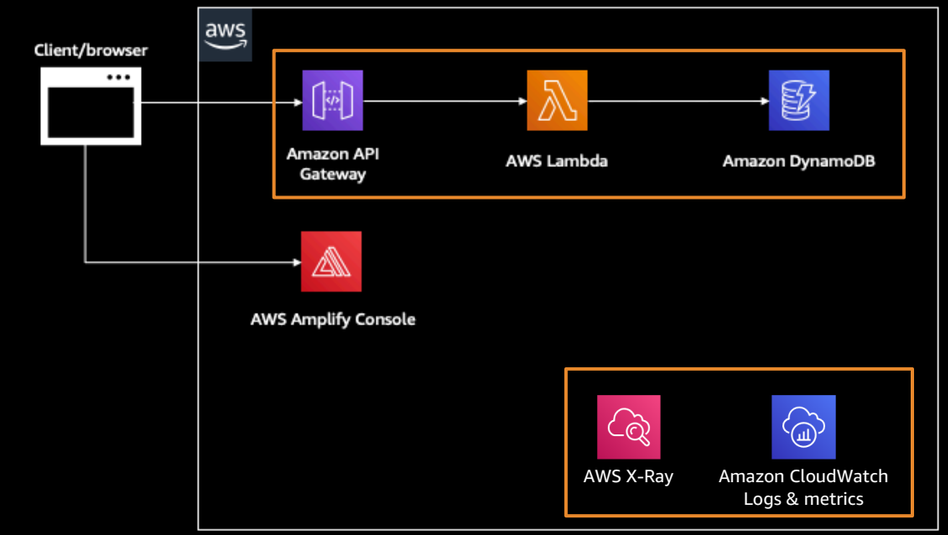
Operational excellence best practices

- Adopt a modern way to build applications
  - Use infrastructure as code
  - Define and deploy infrastructure in stages isolated in separate environments / accounts



AWS SAM

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetProductsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.getProducts
      Runtime: nodejs10.x
      CodeUri: src/
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref ProductTable
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /products/{productId}
            Method: get
  ProductTable:
    Type: AWS::Serverless::SimpleTable
```

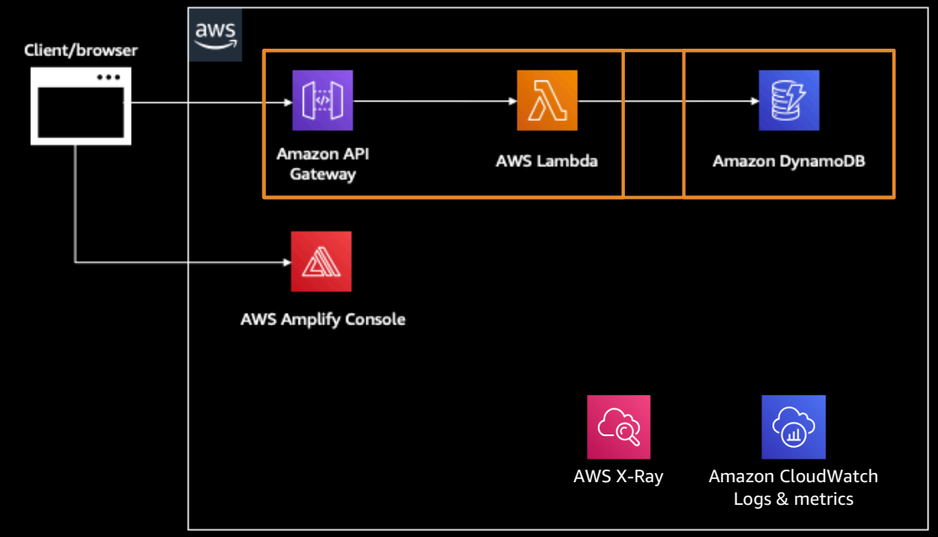


Serverless web applications



Operational excellence best practices

- Adopt a modern way to build applications
  - Use infrastructure as code
  - Define and deploy infrastructure in stages isolated in separate environments / accounts





AWS CDK

CDK construct

```
const apiGatewayToLambda = new ApiGatewayToLambda(
  this,
  "ApiGatewayToLambdaStack",
  {
    lambdaFunctionProps: {
      runtime: lambda.Runtime.NODEJS_12_X,
      handler: "index.handler",
      code: lambda.Code.fromAsset("lambda"),
    },
    apiGatewayProps: {
      defaultMethodOptions: {
        authorizationType: apigw.AuthorizationType.NONE,
      },
    },
  }
);
```

CDK resource

```
const dynamoTable = new dynamodb.Table(this, "items", {
  partitionKey: {
    name: "itemId",
    type: dynamodb.AttributeType.STRING,
  },
});
```

Joining construct and resource

```
const apiFunction = apiGatewayToLambda.lambdaFunction;
apiFunction.addEnvironment("TABLE_NAME", dynamoTable.tableName);
dynamoTable.grantReadWriteData(apiFunction);
```

AWS SAM or AWS CDK?

If you prefer defining your serverless infrastructure in concise declarative templates, use AWS SAM

If you want to define your AWS infrastructure in a familiar programming language, try out AWS CDK



Kim Carpenter – CC BY 2.0
https://flic.kr/p/bXDDh5

Do not start from scratch

Use the available vetted architecture patterns

- AWS SAM sample templates
- AWS Solutions Constructs Library



AWS SAM



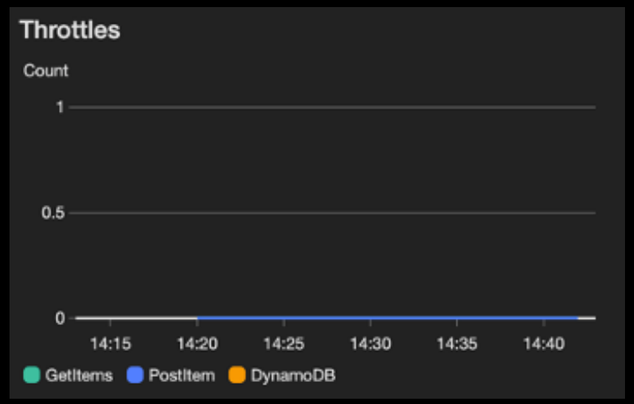
AWS CDK

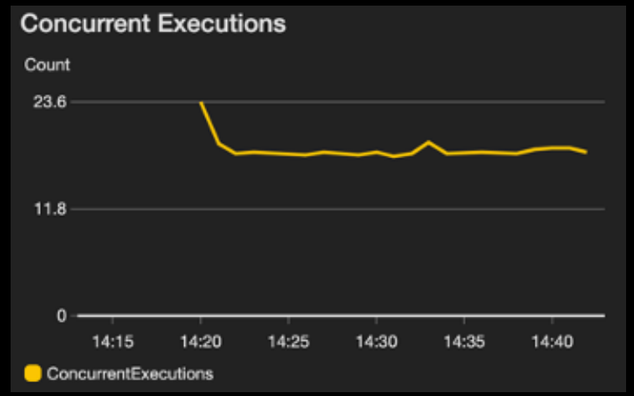
Serverless web applications

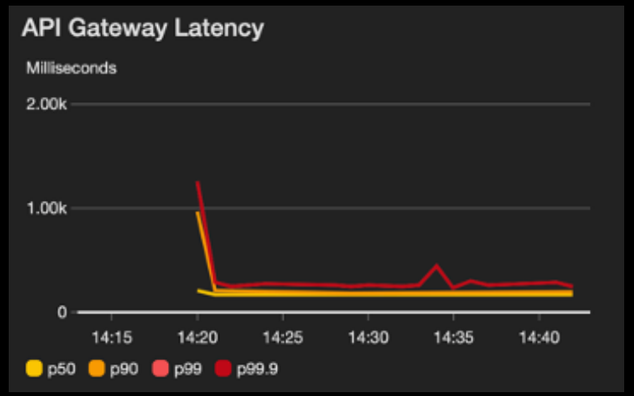
Operational excellence best practices

- Observability is the way to govern the serverless application's health
  - Monitor with Amazon CloudWatch Logs and Metrics – understand, analyze, and alert on metrics provided out of the box

Amazon CloudWatch Logs and Metrics







Serverless web applications

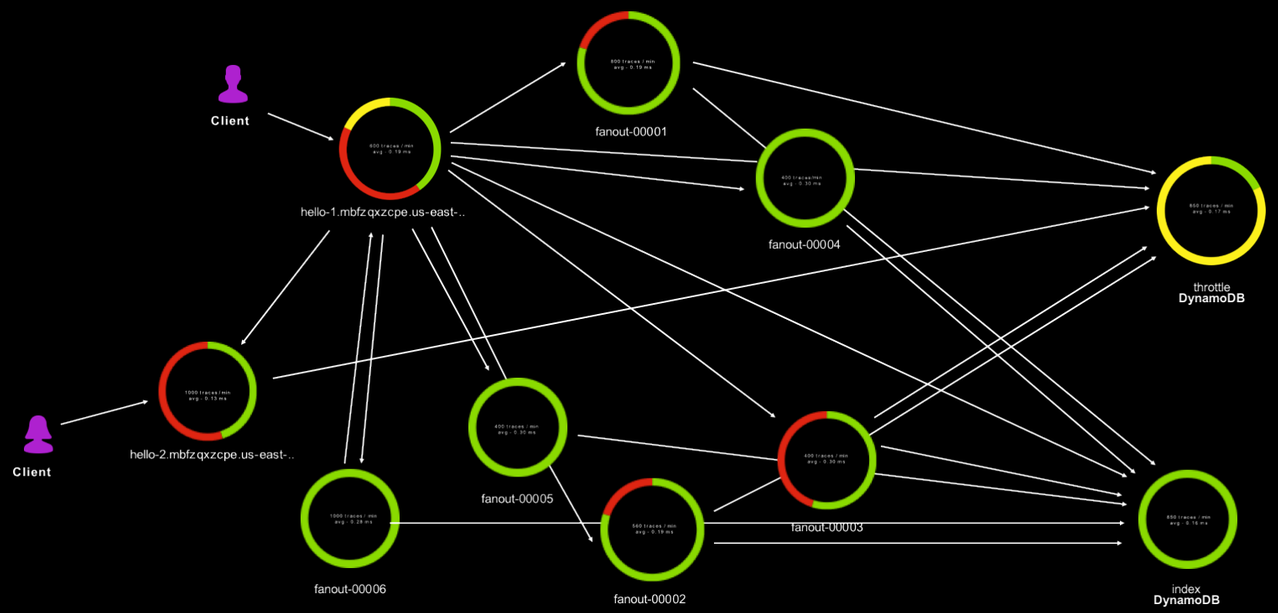
Operational excellence best practices

- Observability is the way to govern the serverless application's health
  - Monitor with Amazon CloudWatch Logs and Metrics – understand, analyze, and alert on metrics provided out of the box
  - Instrument your code with AWS X-Ray

AWS X-Ray

Amazon CloudWatch Logs and Metrics

```
var AWSXRay = require('aws-xray-sdk-core');
var AWS = AWSXRay.captureAWS(require('aws-sdk'));
S3Client = AWS.S3();
```



Serverless web applications



Operational excellence best practices

- Observability is the way to govern the serverless application's health
  - Monitor with Amazon CloudWatch Logs and Metrics
  - Instrument your code with AWS X-Ray
  - Leverage AWS Lambda PowerTools to reduce custom code for logging, tracing, and creating metrics

```python
from aws_lambda_powertools import Logger, Tracer, Metrics
from aws_lambda_powertools.metrics import MetricUnit

tracer = Tracer()
logger = Logger()
metrics = Metrics()


@metrics.log_metrics(capture_cold_start_metric=True)
@tracer.capture_lambda_handler
@logger.inject_lambda_context
def handler(event, context):
    metrics.add_metric(name="CartCheckedOut", unit=MetricUnit.Count, value=1)
    metrics.add_metadata(key="cart_details", value=cart_state)
```

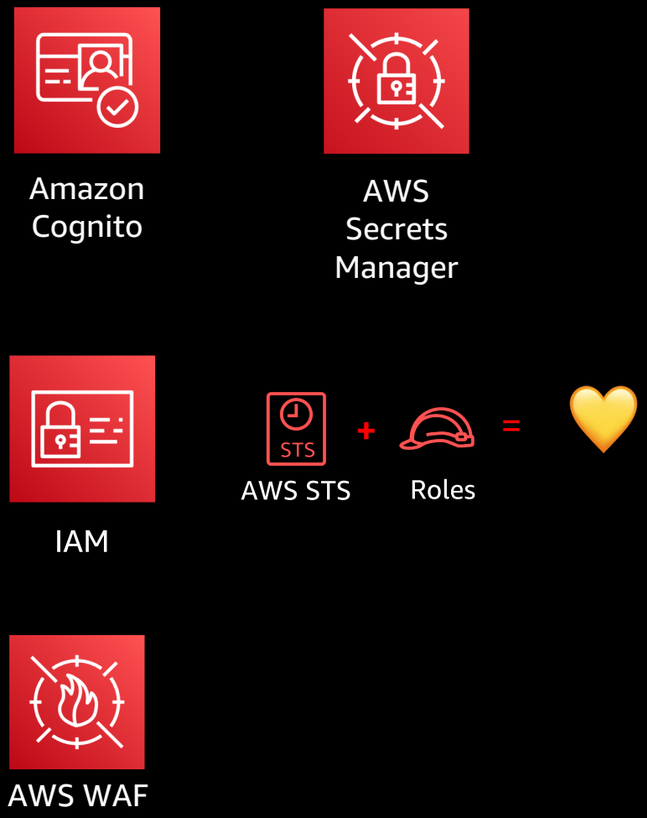
Lambda PowerTools for Java

Lambda PowerTools for Python

Serverless web applications

Security best practices

- Control access to your APIs and implement AppSec
  - Least privilege is a must
  - Never trust user input
  - Use AuthN and AuthZ mechanisms for your APIs; use Amazon Cognito or a SAML / JWT identity provider
  - Enable fine-grain control access to resources and actions; leverage SAM predefined IAM policies
  - Use temporary credentials
  - No secrets in code – use Parameter Store or AWS Secrets Manager
  - Choose between public or private API Gateway endpoints based on customer origin



Serverless web applications



Reliability best practices

- Protect your resources and build resiliency into your serverless application
  - Use throttling
  - Implement back-off and retries
  - Protect non-scalable resources; RDS Proxy can save you hours of sleep
  - If you have long-running transactions, use AWS Step Functions
  - Consider moving to async messaging – seriously, it is possible
  - Decouple your services interactions
    - Amazon SQS
    - Amazon SNS
  - Use Amazon EventBridge to route events reliably; remove that burden from your application logic



Amazon API Gateway



AWS Lambda



Amazon RDS Proxy



AWS Step Functions



Amazon SQS



Amazon SNS



Amazon EventBridge

Serverless web applications



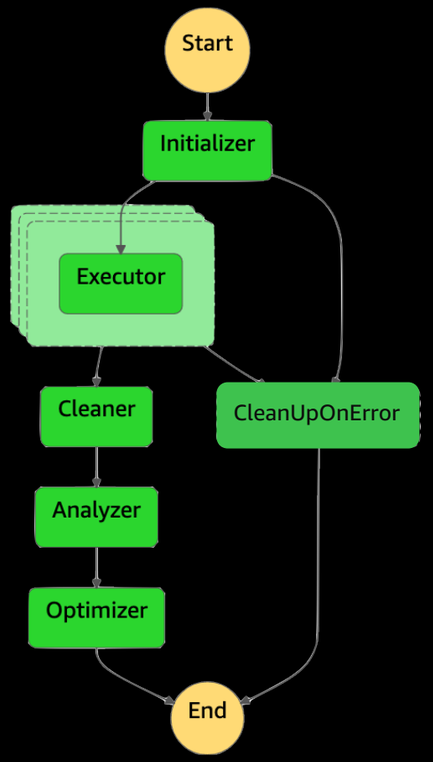
Performance best practices

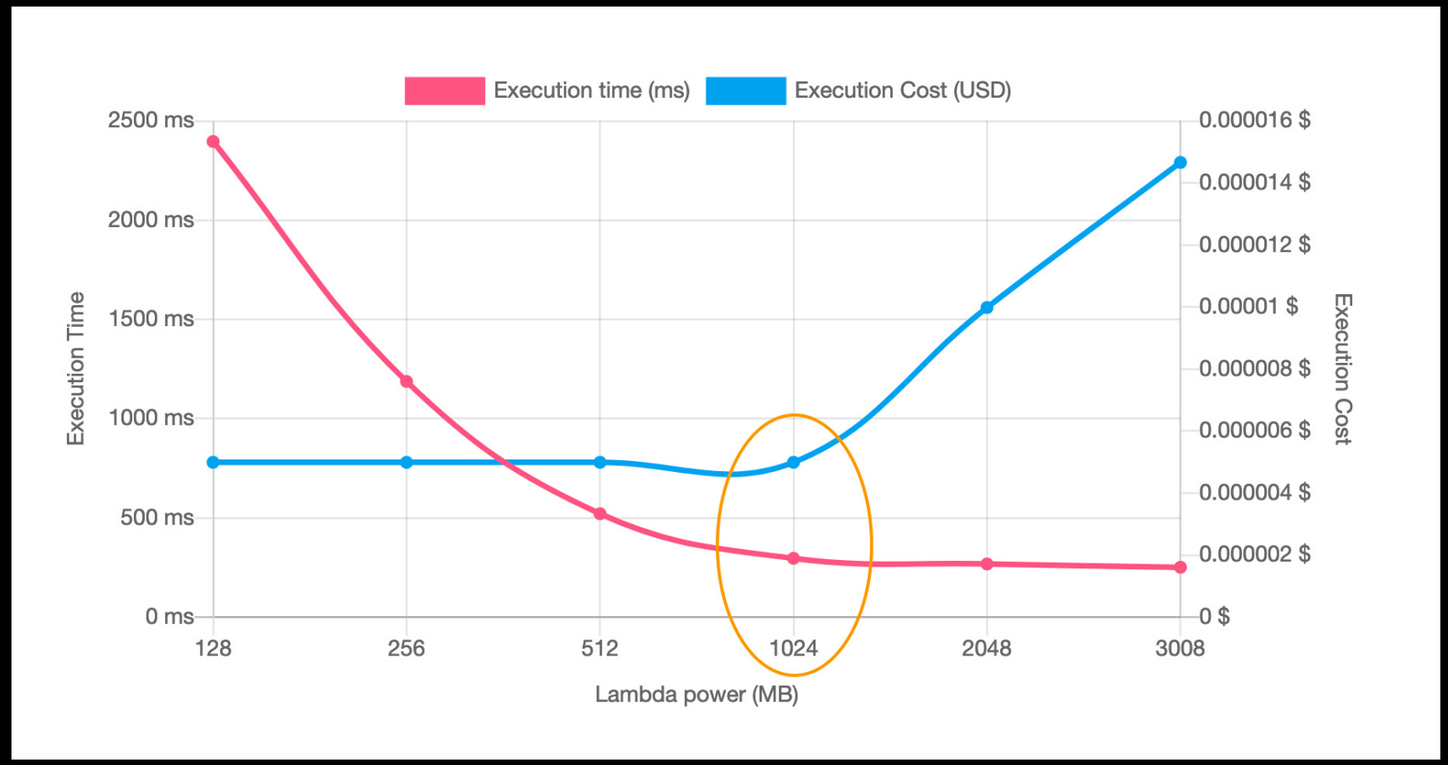
- Optimize your serverless application performance
  - Understand services scaling behavior
  - Optimize application code. Yes! Do it again!
  - DynamoDB on-demand capacity first, then provisioned, if needed
  - Apply caching when possible
  - Use services integrations
  - If needed, evaluate Lambda provisioned concurrency with auto-scaling
  - Use Lambda Power Tuning to tune power

Wait, what's Lambda Power Tuning?

VISUALIZE AND FINE-TUNE THE MEMORY / POWER CONFIGURATION OF LAMBDA FUNCTIONS

Lambda Power Tuning





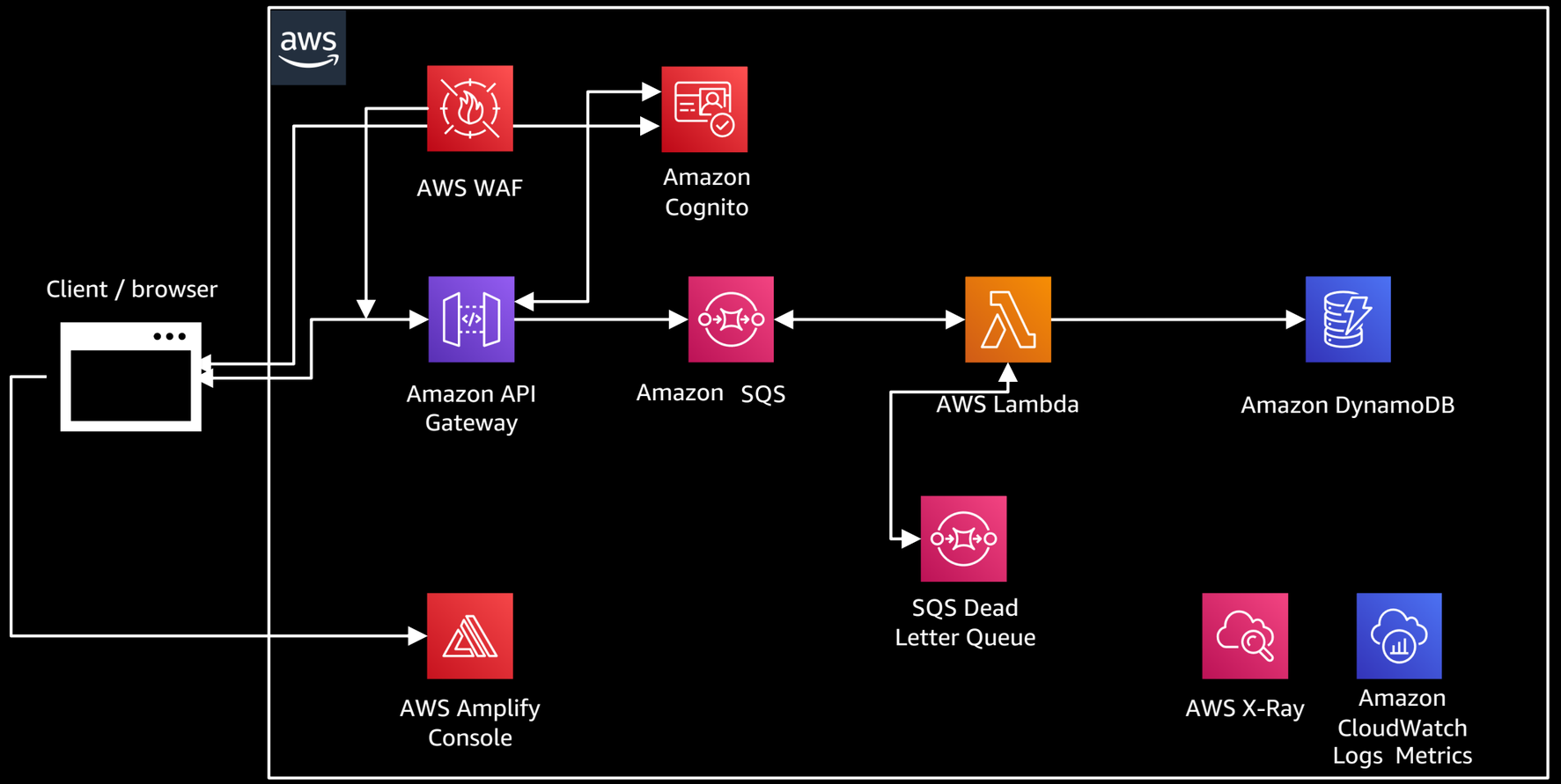
Serverless web applications



Cost-optimization best practices

- Design your application to maximize value
  - Reduce your Lambda functions code size
  - Use asynchronous design patterns
  - Use Embedded Metric Format for CloudWatch
  - Review your log-retention policies
  - Eliminate job polling or task coordination from your application logic
  - Use Lambda Power Tuning to find that sweet spot between performance and cost

Well-architected serverless web applications



OPERATIONAL EXCELLENCE

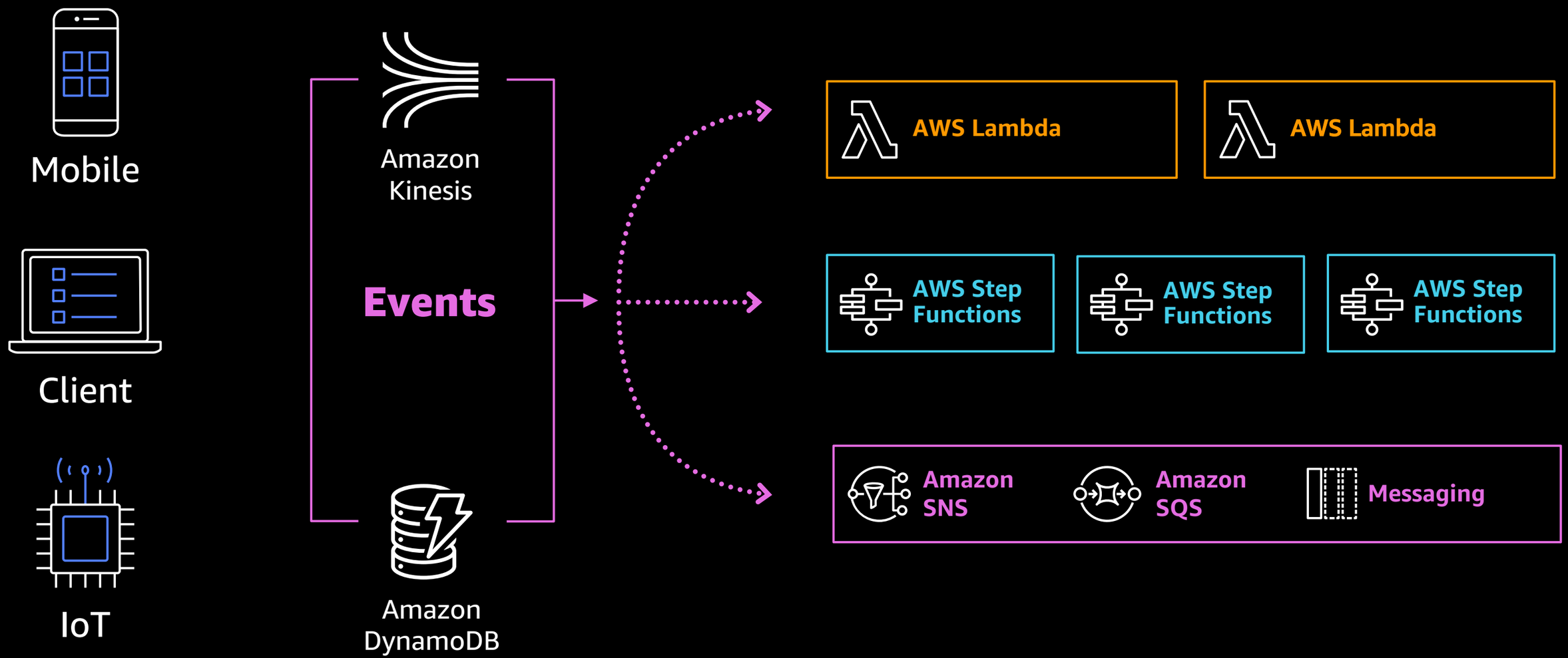
SECURITY

RELIABILITY

PERFORMANCE EFFICIENCY

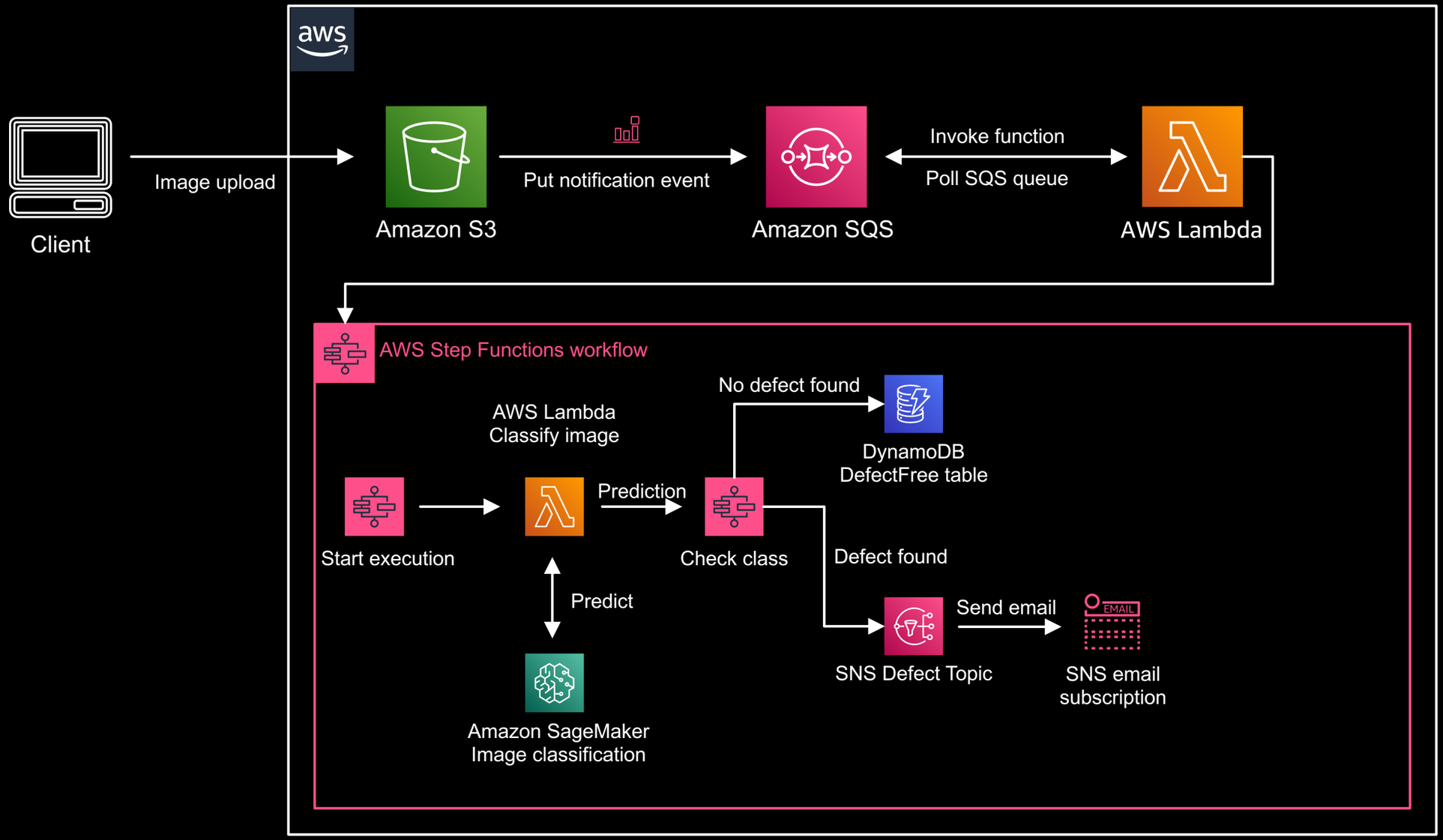
COST OPTIMIZATION

Do you love events?



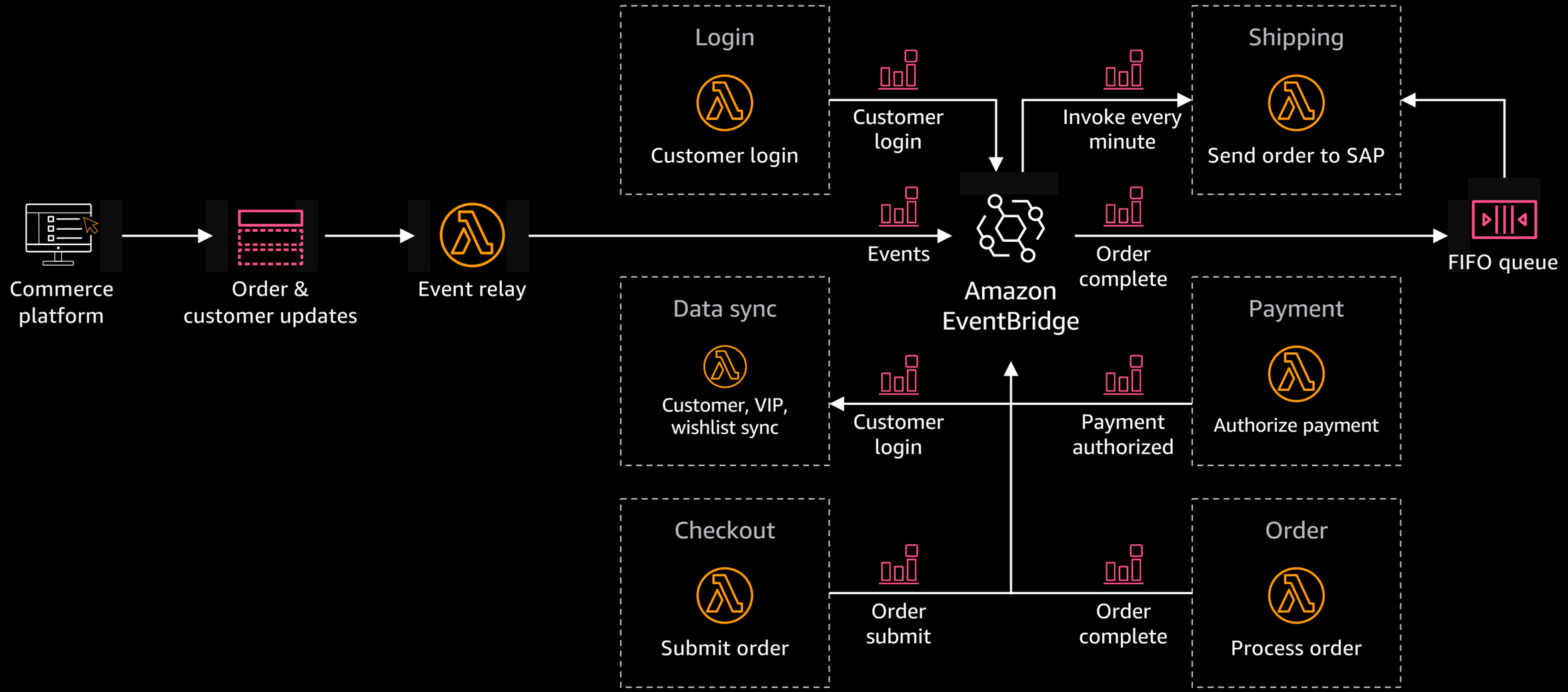
Do you love events?

DETECT AND REPORT DEFECTS IN PRODUCTS IMAGES AT ANY SCALE



Do you love events?

LEGO EVENT-DRIVEN ARCHITECTURE



Remember

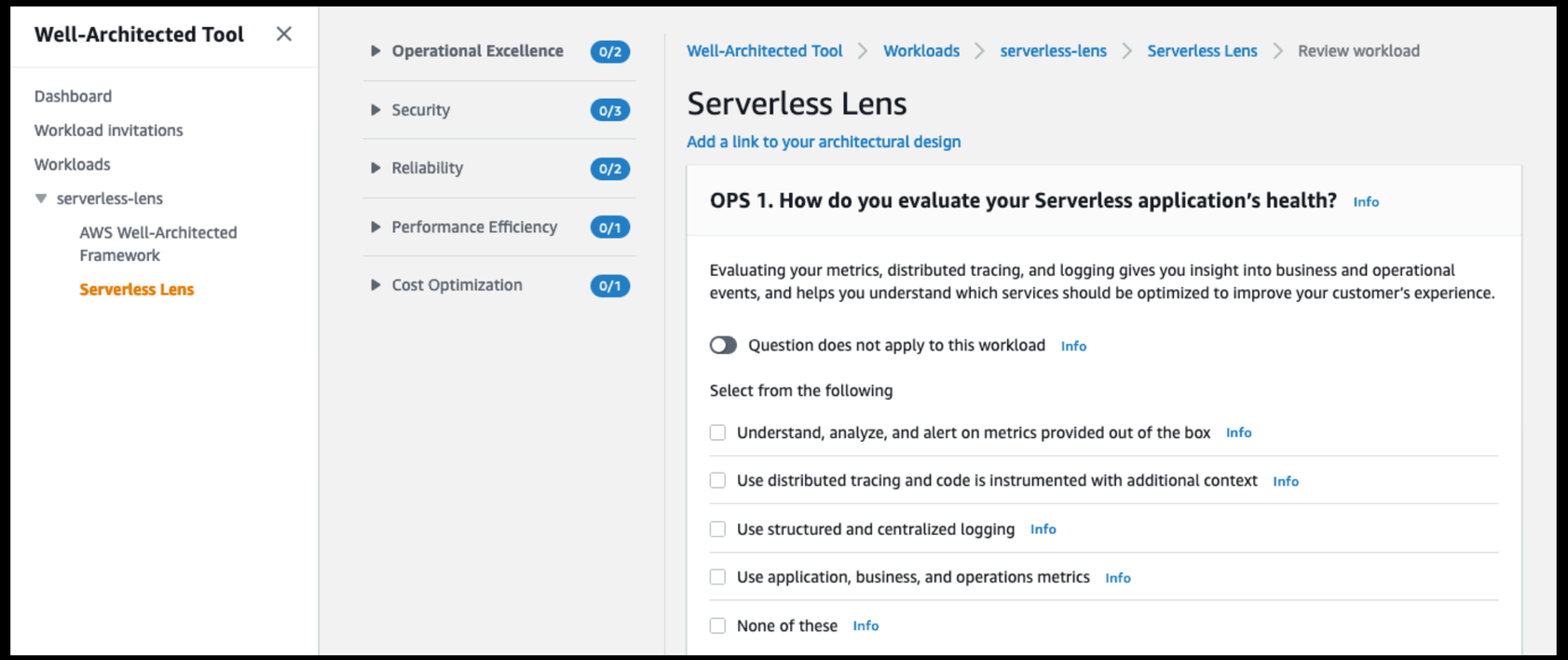
- Make your functions single purpose, concise, short
- Code is debt; write fewer functions
- Understand and take advantage of the serverless concurrency model
- Share nothing; assume no hardware affinity
- Orchestrate with state machines, not application logic
- Use events to trigger transactions
- Leverage managed services when designing for failures and duplicates



aws
Well-Architected
Serverless Lens

Your next steps

- Start a Well-Architected Review right now
  - Call your AWS Solutions Architect
  - Engage your preferred AWS Partner
  - Or do it yourself with the Well-Architected Tool in the AWS Web Console



“Serverless is the *fastest way* to build modern applications”

Adrian Cockcroft
AWS VP Cloud Architecture Strategy



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Hernan Garcia

Solutions Architect
AWS
@hernangarcia

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.