Welcome to Advance Workshop!

App Modernization with Serverless

Rudi & Yudho
Solutions Architect, Indonesia

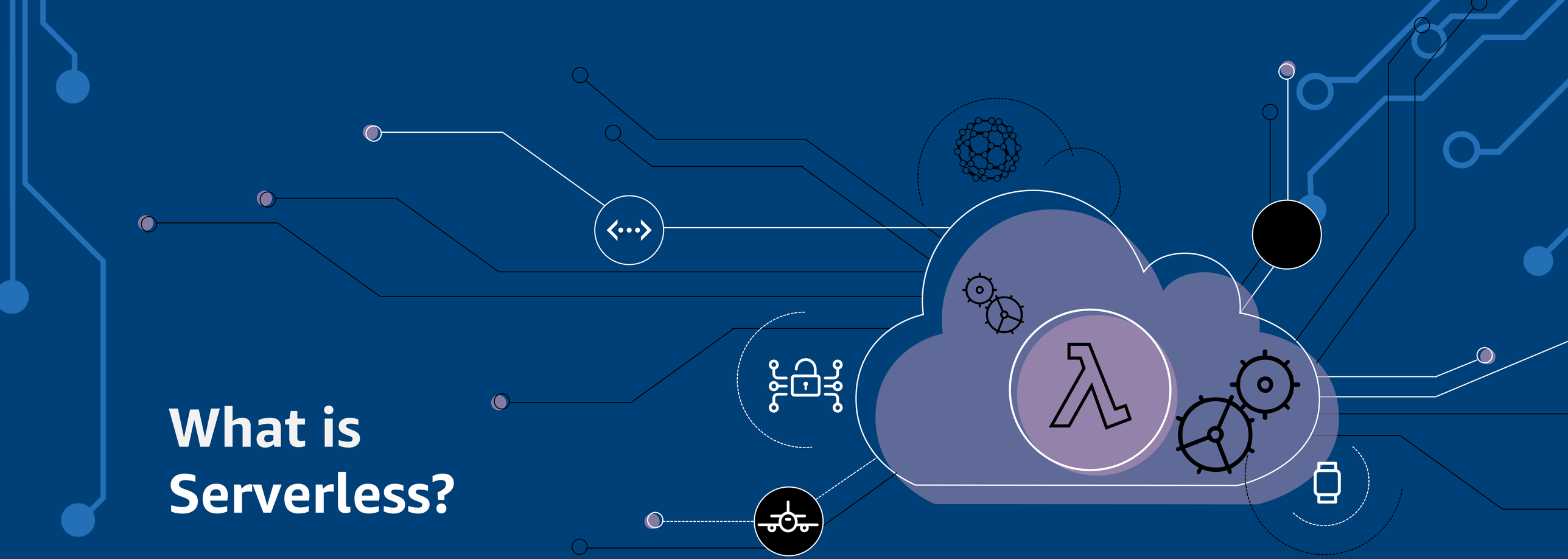
What to expect from this workshop

- Serverless architectures
- Overview of AWS Lambda
- Overview of Amazon API Gateway
- Get Your Hands Dirty– Time to build!
- Quiz/Wrap-up/Feedback

Theory : 20%,
Lab: 80%

Logistic Requirements

- Bring your own laptop with software installed : SSH terminal, latest Chrome browser, text editor
- Recommended to each participant have one AWS account, to avoid naming conflicts for certain resources. You can create your personal AWS account, all of the resources you will launch as part of this workshop are eligible for the AWS free tier if your account is less than 12 months old.
- Willingness to Hands-On!



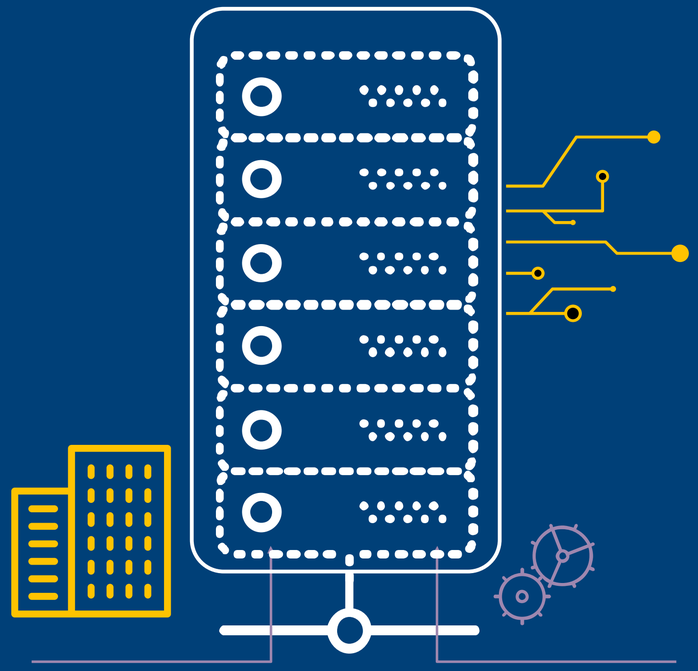
What is Serverless?

Build and run applications
without thinking about servers

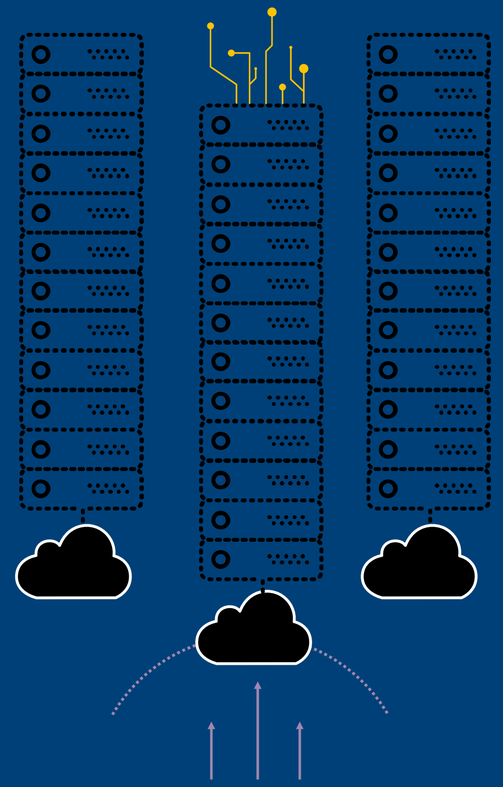
Let’s take a look at the evolution of computing

Physical Servers in Datacenters

Virtual Servers in Datacenters



Virtual Servers in the Cloud

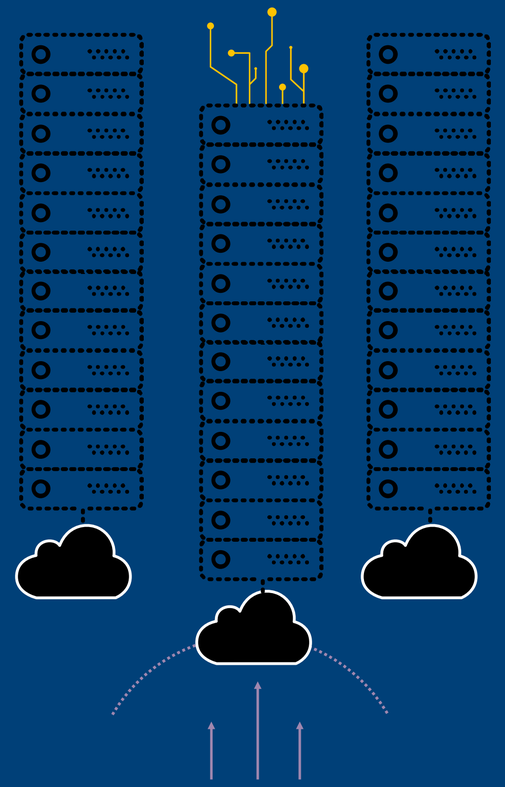


Each progressive step was better

- Higher utilization
- Faster provisioning speed
- Improved uptime
- Disaster recovery
- Hardware independence

- Trade CAPEX for OPEX
- More scale
- Elastic resources
- Faster speed and agility
- Reduced maintenance
- Better availability and fault tolerance

Virtual Servers in the Cloud

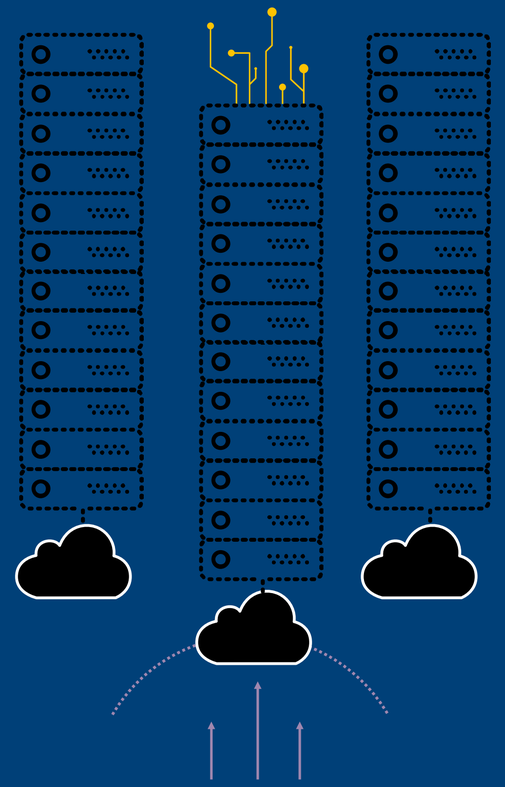


But there are still limitations

- Still need to administer virtual servers
- Still need to manage capacity and utilization
- Still need to size workloads
- Still need to manage availability, fault tolerance
- Still expensive to run intermittent jobs

- Trade CAPEX for OPEX
- More scale
- Elastic resources
- Faster speed and agility
- Reduced maintenance
- Better availability and fault tolerance

Virtual Servers in the Cloud

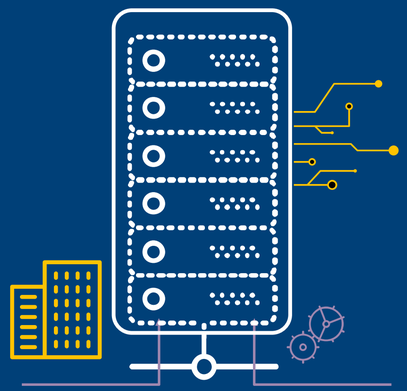


aws

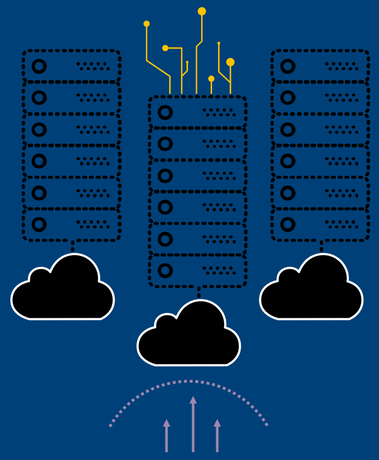
Evolving to Serverless

Physical Servers in Datacenters

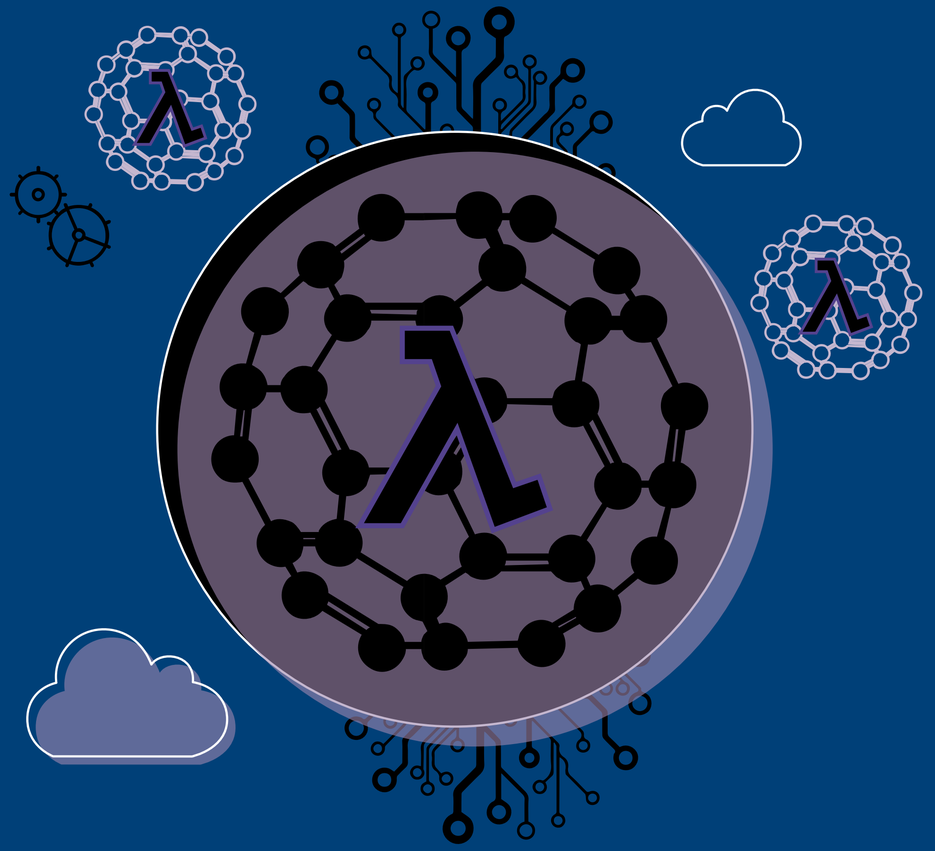
Virtual Servers in Datacenters



Virtual Servers in the Cloud



SERVERLESS



aws

No server is easier to manage than no server

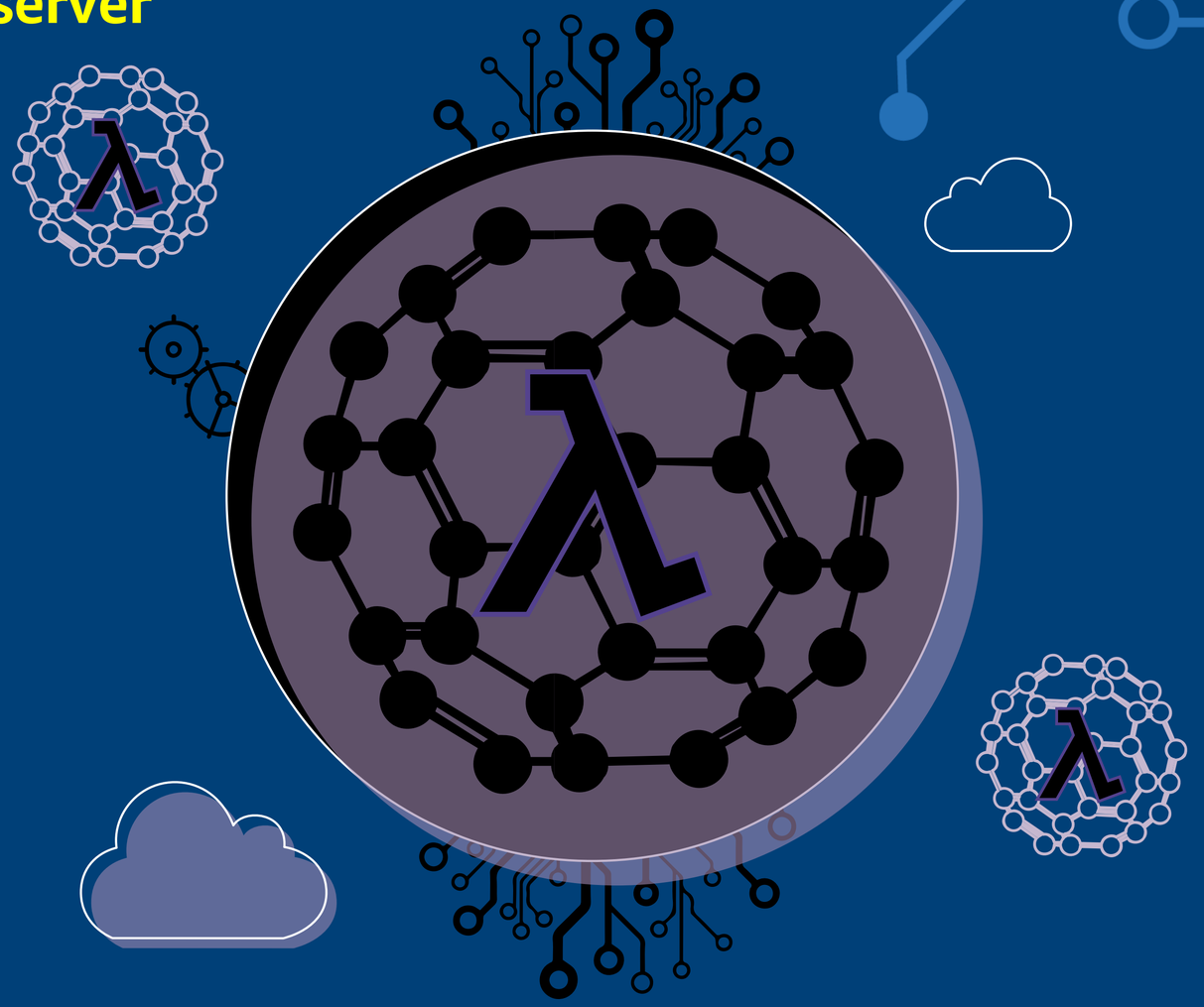
All of this goes away

~~Provisioning and utilization~~

~~Availability and fault tolerance~~

~~Scaling~~

~~Operations and management~~

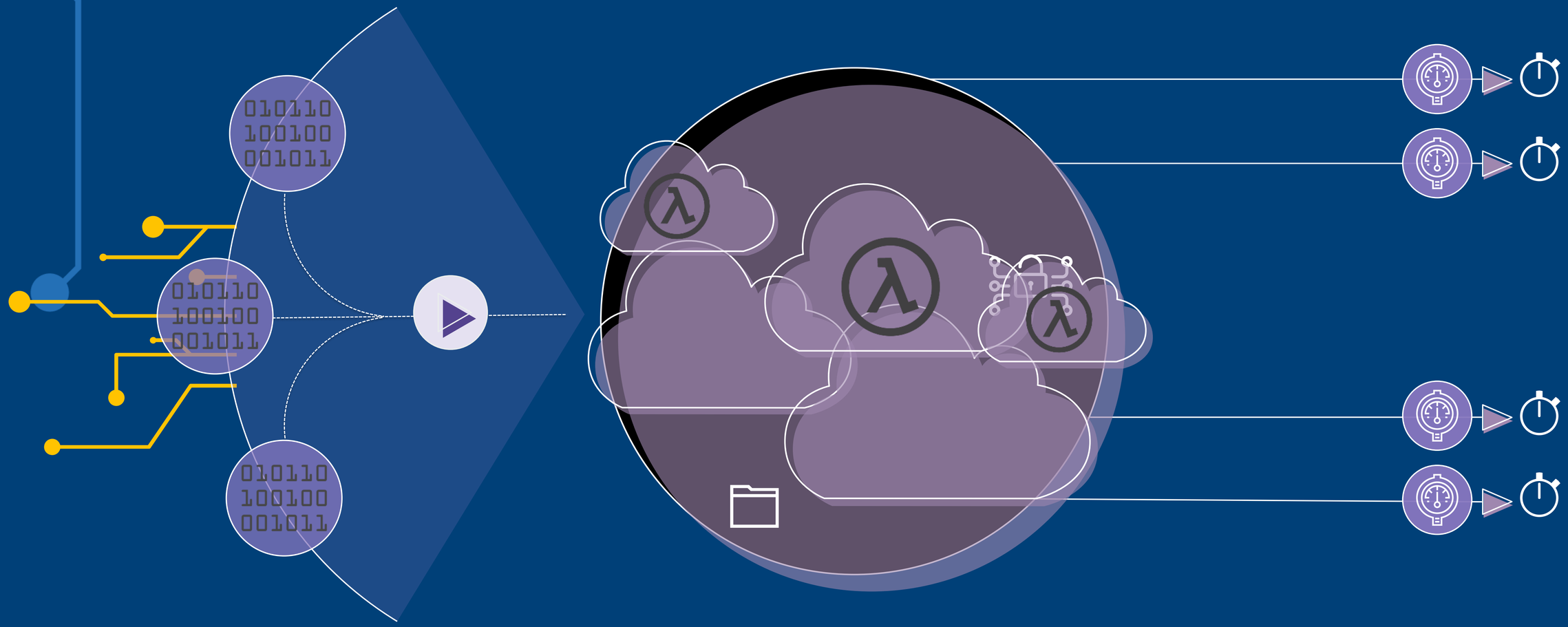


Deliver on demand, never pay for idle

EVENT DRIVEN

CONTINUOUS SCALING

PAY BY USAGE



What is Serverless?

Serverless means…

No servers to provision or manage

Scales with usage

Never pay for idle

Availability and fault tolerance built in

aws

AWS: A Mature Serverless Portfolio

Compute

AWS Lambda

Storage

Amazon S3

Database

Amazon DynamoDB

Amazon Aurora Serverless

API Proxy

Amazon API Gateway

Messaging

Amazon SQS

Amazon SNS

Analytics

Amazon Kinesis

AWS Glue

Amazon Athena

Orchestration

AWS Step Functions

Monitoring and Debugging

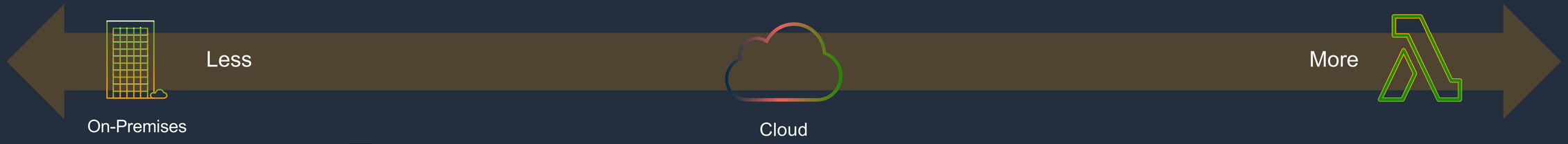
AWS X-Ray

Edge Compute

AWS Greengrass

Lambda@Edge

AWS operational responsibility models



Compute	Virtual Machine	EC2	Elastic Beanstalk		Fargate	AWS Lambda
Databases	MySQL	MySQL on EC2	RDS MySQL	RDS Aurora	Aurora Serverless	DynamoDB
Storage	Storage					S3
Messaging	ESBs		Amazon MQ	Kinesis		SQS / SNS
Analytics	Hadoop	Hadoop on EC2	EMR	Elasticsearch Service		Athena

AWS Lambda

Your feedback helped create Lambda!

- No direct responsibility for infrastructure resources
- Quick and simple deployments
- Highly available and scalable apps with zero administration
- Costs that are closely aligned to application usage

aws

AWS compute offerings

Amazon EC2

Resizable virtual servers in the cloud

Amazon ECS

Container management service for running Docker on EC2

AWS Lambda

Serverless compute, run code in response to events

Benefits of using Lambda

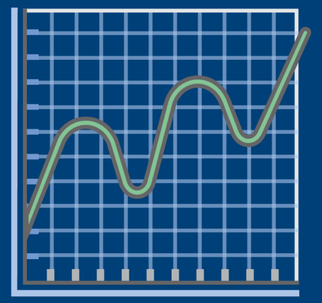
1

No Servers to Manage

Lambda automatically runs your code without requiring you to provision or manage servers. Just write the code and upload it to Lambda.

2

Continuous Scaling



Lambda automatically scales your application by running code in response to each trigger. Your code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload.

3

Subsecond Metering

With Lambda, you are charged for every 100 ms your code executes and the number of times your code is triggered. You don’t pay anything when your code isn’t running.

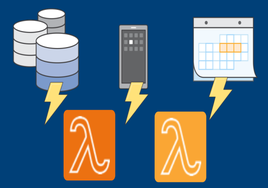
AWS Lambda – How it works

Bring your own code

- Node.js, Java, Python, Go, C#, Ruby
- Java = Any JVM based language such as Scala, Clojure, etc.
- Bring your own libraries

Simple resource model

- Select memory from 128MB to 3GB in 64MB steps
- CPU & Network allocated proportionately to RAM
- Reports actual usage



Flexible invocation paths

- Event or RequestResponse invoke options
- Existing integrations with various AWS services

Fine grained permissions

- Uses IAM role for Lambda execution permissions
- Uses Resource policy for AWS event sources

Serverless applications

EVENT SOURCE

Changes in data state

Requests to endpoints

Changes in resource state

FUNCTION

Node.js
Python
Java
C#
Go
Ruby

SERVICES (ANYTHING)

aws

Event Sources which can Trigger Lambda

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- Amazon CloudWatch Events
- AWS Application Load Balancer
- AWS CodeCommit
- Scheduled Events (powered by Amazon CloudWatch Events)
- AWS Config
- Amazon Alexa
- Amazon Lex
- Amazon API Gateway
- AWS IoT Button
- Amazon CloudFront
- Amazon Kinesis Data Firehose
- Other Event Sources: Invoking a Lambda Function On Demand

aws

Sample of Event – S3 Put

```
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "s3": {
        "configurationId": "testConfigRule",
        "object": {
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901",
          "key": "HappyFace.jpg",
          "size": 1024
        },
        "bucket": {
          "arn": bucketarn,
          "name": "sourcebucket",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          }
        },
        "s3SchemaVersion": "1.0"
      },
      "responseElements": {
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH",
        "x-amz-request-id": "EXAMPLE123456789"
      },
      "awsRegion": "us-east-1",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "eventSource": "aws:s3"
    }
  ]
}
```


aws

Sample of Event – IoT Button

```
{
  "serialNumber": "ABCDEFG12345",
  "clickType": "SINGLE",
  "batteryVoltage": "2000 mV"
}
```

Sample of Event – AWS SQS

```
{
     "Records": [
        {
             "messageId": "c80e8021-a70a-42c7-a470-796e1186f753",
             "receiptHandle": "AQEBJQ+/u6NsnT5t8Q/VbVxgdUl4TMKZ5FqhksRdIQvLBhwNvADoBxYSOVeCBXdnS9P+erlTtwEAI
             "body": "{\"foo\":\"bar\"}",
             "attributes": {
                 "ApproximateReceiveCount": "3",
                 "SentTimestamp": "1529104986221",
                 "SenderId": "594035263019",
                 "ApproximateFirstReceiveTimestamp": "1529104986230"
             },
             "messageAttributes": {},
             "md5OfBody": "9bb58f26192e4ba00f01e2e7b136bbd8",
             "eventSource": "aws:sqs",
             "eventSourceARN": "arn:aws:sqs:us-west-2:594035263019:NOTFIFOQUEUE",
             "awsRegion": "us-west-2"
        }
    ]
}
```

Anatomy of a Lambda function

Handler() function

Function to be executed upon invocation

Event object

Data sent during Lambda Function Invocation

Context object

Methods available to interact with runtime information (request ID, log group, etc.)

```
public String handleRequest(Book book, Context context) {
    saveBook(book);

    return book.getName() + " saved!";
}
```

Lambda Programming Model – Node.js

```
exports.myHandler = function(event, context, callback) {
    ... function code
    callback(null, "some success message");
   // or
   // callback("some error type");
}
```

myHandler – This is the name of the function AWS Lambda invokes. Suppose you save this code as helloworld.js

context – AWS Lambda uses this parameter to provide details of your Lambda function's execution. For more information, see The Context Object (Node.js).

callback (optional) - The Node.js runtimes v6.10 and v8.10 support the optional callback parameter. You can use it to explicitly return information back to the caller.
The general syntax is: callback(Error error, Object result);

Lambda Programming Model – Python

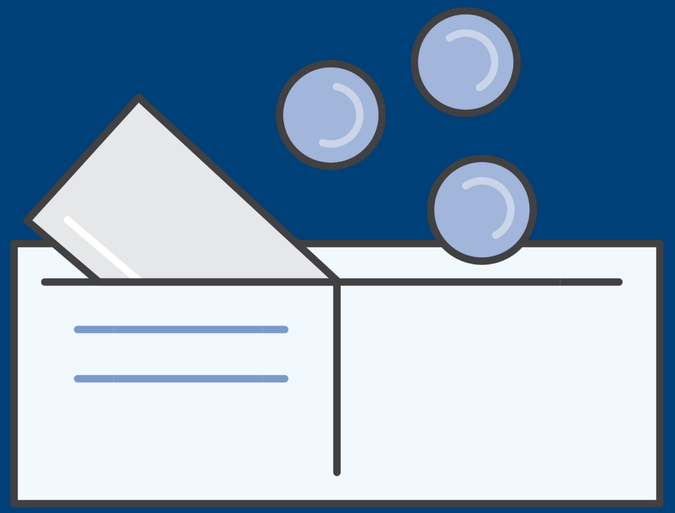
```
def handler_name(event, context):
    ...
    return some_value
```

event – AWS Lambda uses this parameter to pass in event data to the handler. This parameter is usually of the Python dict type. It can also be list, str, int, float, or NoneType type.

context – AWS Lambda uses this parameter to provide runtime information to your handler. This parameter is of the LambdaContext type.

Optionally, the handler can return a value

Fine-Grained Pricing



Buy compute time in 100ms increments

Low request charge

No hourly, daily, or monthly minimums

No per-device fees

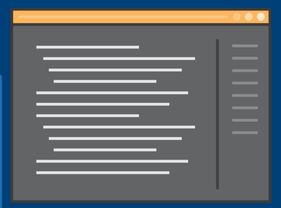
Never pay for idle

Free Tier

1M requests and 400,000 GB-s of compute.

Every month, every customer.

Using AWS Lambda



Authoring functions

- Cloud9
- WYSIWYG editor or upload packaged .zip
- Third-party plugins (Eclipse, Visual Studio)

Monitoring and logging

- Metrics for requests, errors, and throttles
- Built-in logs to Amazon CloudWatch Logs
- X-Ray integration

Programming model

- Use processes, threads, /tmp, sockets normally
- AWS SDK built in (Python and Node.js)

Stateless

- Persist data using external storage
- No affinity or access to underlying infrastructure

Lambda execution models

Synchronous (push)

Amazon
API Gateway

/order

AWS Lambda
function

Asynchronous (event)

Amazon
SNS

Amazon
S3

reqs

AWS Lambda
function

Poll-based

Amazon
DynamoDB

Amazon
SQS

Amazon
Kinesis

changes

AWS Lambda
service

function

aws

Lab 1 :

Go to Qwiklabs.com

Search Lab : **Introduction to AWS Lambda**

Amazon API Gateway

Your feedback

Managing multiple versions and stages of an API is difficult

Monitoring third-party developers’ access is time consuming

Access authorization is a challenge

Traffic spikes create an operational burden

What if I don’t want servers at all?

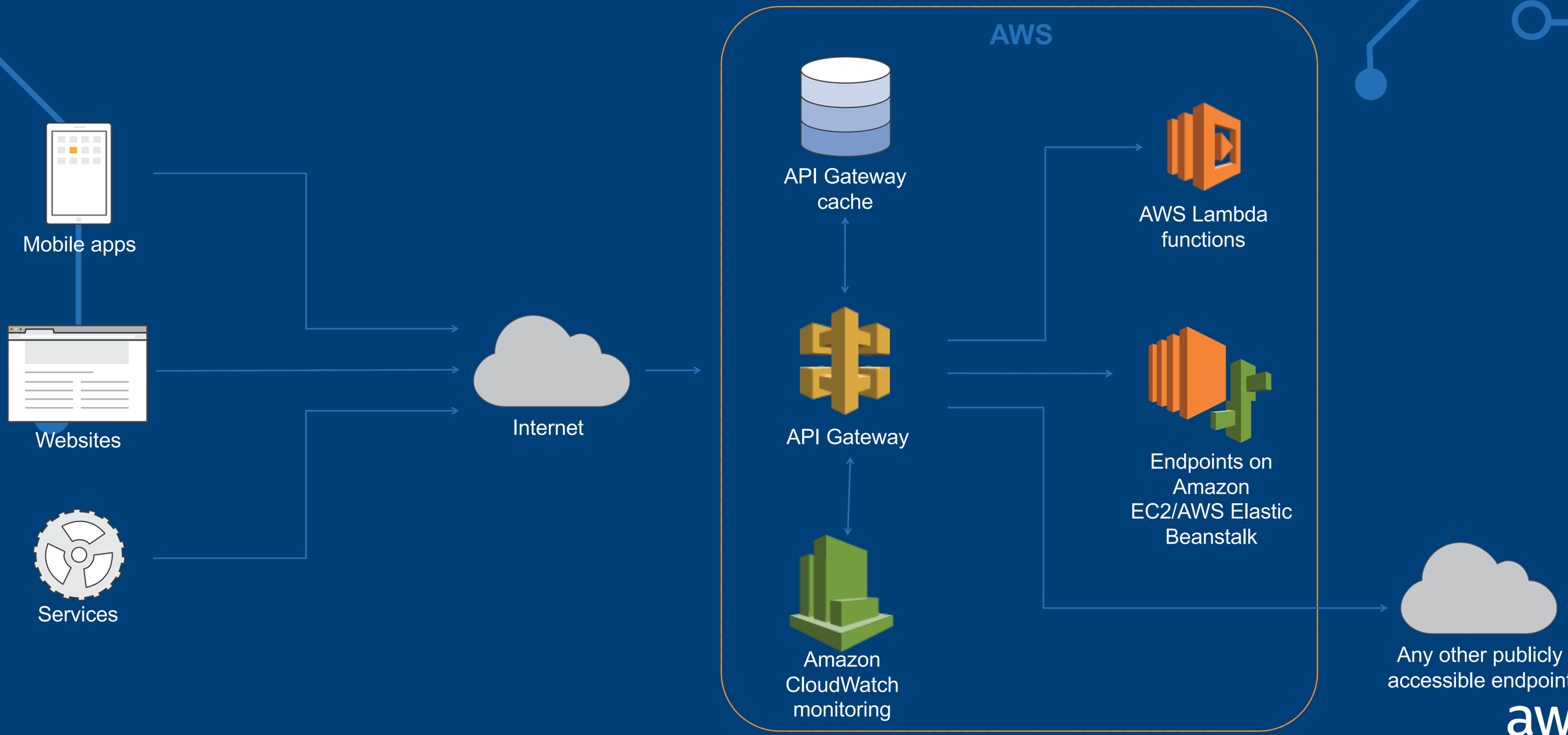
API Gateway - Capabilities

- Host multiple versions and stages of your APIs
- Create and distribute API keys to developers
- Leverage signature version 4 to authorize access to APIs
- Throttle and monitor requests to protect your backend
- Utilize Lambda as a backend

Benefits of API Gateway

- Managed cache to store API responses
- Reduced latency and distributed denial of service (DDoS) protection through Amazon CloudFront
- SDK generation for iOS, Android, and JavaScript
- Swagger support
- Request and response data transformation

An API call flow



Lab 2 : (Optional if time permit)

Go to Qwiklabs.com

Search Lab : **Introduction to Amazon API Gateway**

Amazon Cognito

Amazon Cognito Identity

Cognito User Pools

You can easily and securely add sign-up and sign-in functionality to your mobile and web apps with a fully-managed service that scales to support 100s of millions of users.

Federated User Identities

Your users can sign-in through social identity providers such as Facebook, Twitter and SAML providers and you can control access to AWS resources from your app.

Amazon Cognito User Pools

1

Serverless Authentication and User Management

Add user sign-up and sign-in easily to your mobile and web apps without worrying about server infrastructure

2

Managed User Directory

A simple, secure, low-cost, and fully managed service to create and maintain a user directory that scales to 100s of millions of users

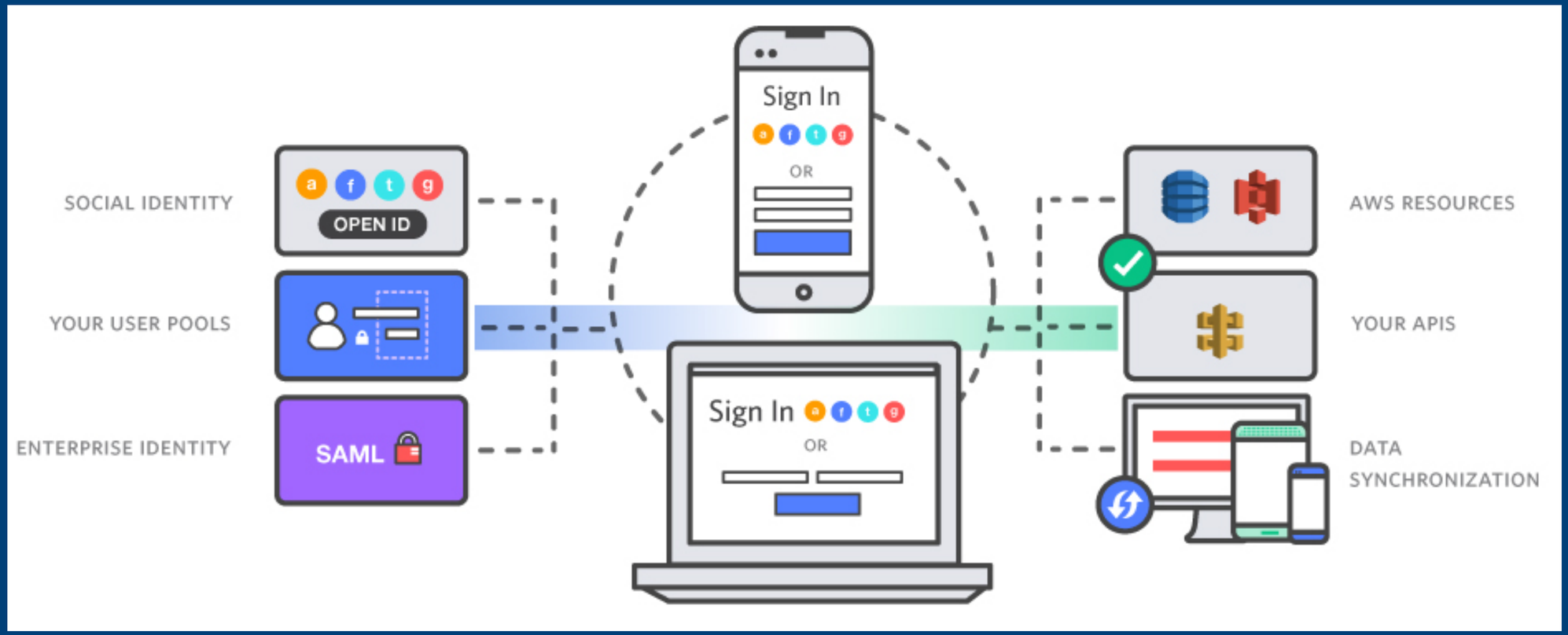
3

Enhanced Security Features

Verify phone numbers and email addresses and offer multi-factor authentication

aws

Comprehensive Support for Identity Use Cases

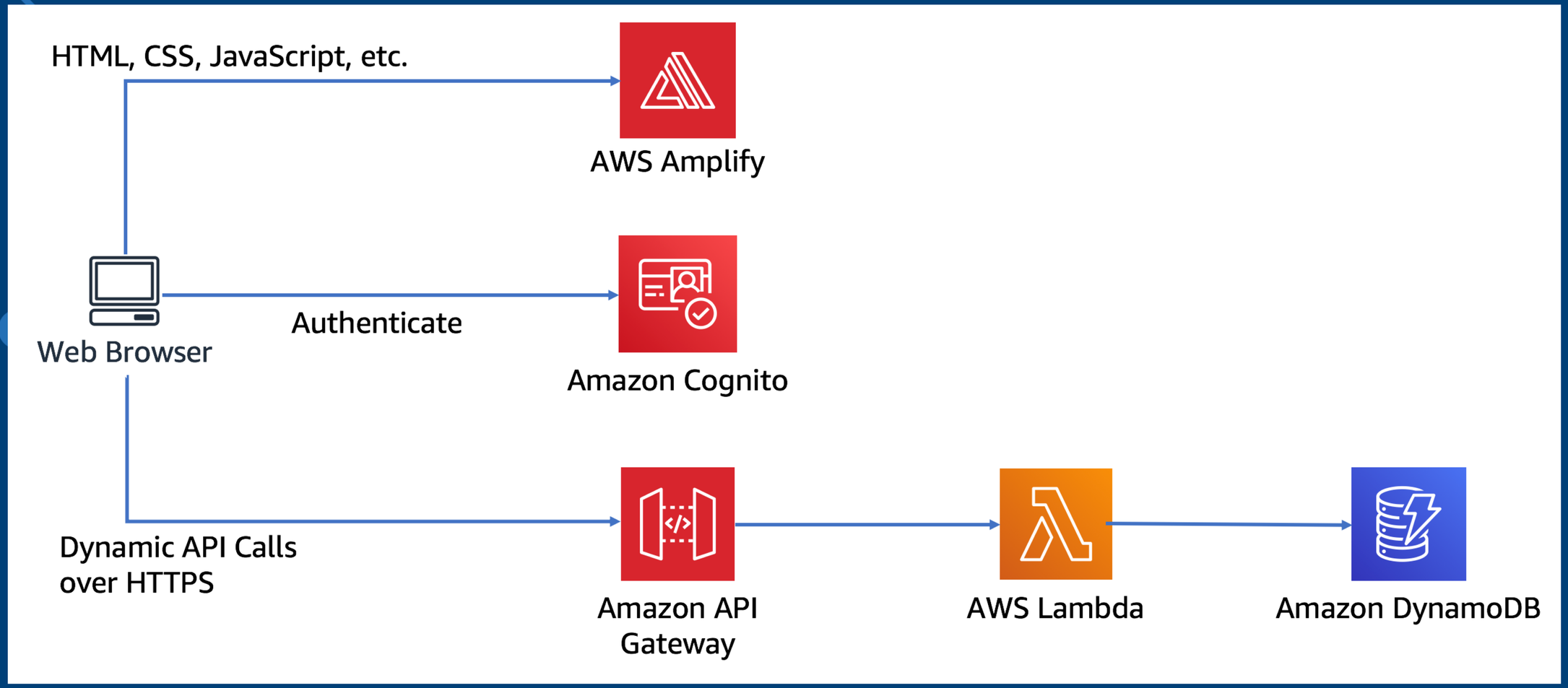


Lab Architectures

What you'll build today!

Lab 3 :

https://bit.ly/2Iwev9m



Serverless is an operational model that spans many different categories of services

COMPUTE


AWS Lambda


AWS Fargate

DATA STORES


Amazon S3


Amazon Aurora Serverless


Amazon DynamoDB

INTEGRATION


Amazon API Gateway


Amazon SQS

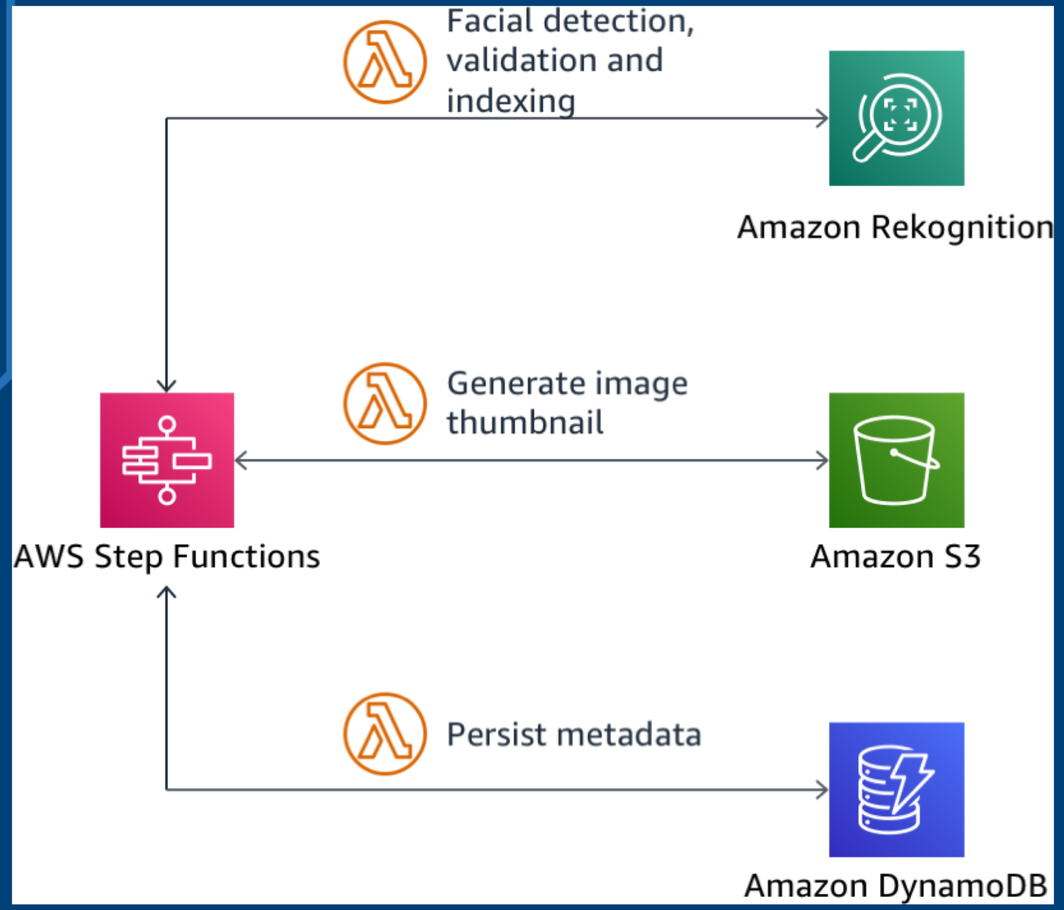

Amazon SNS

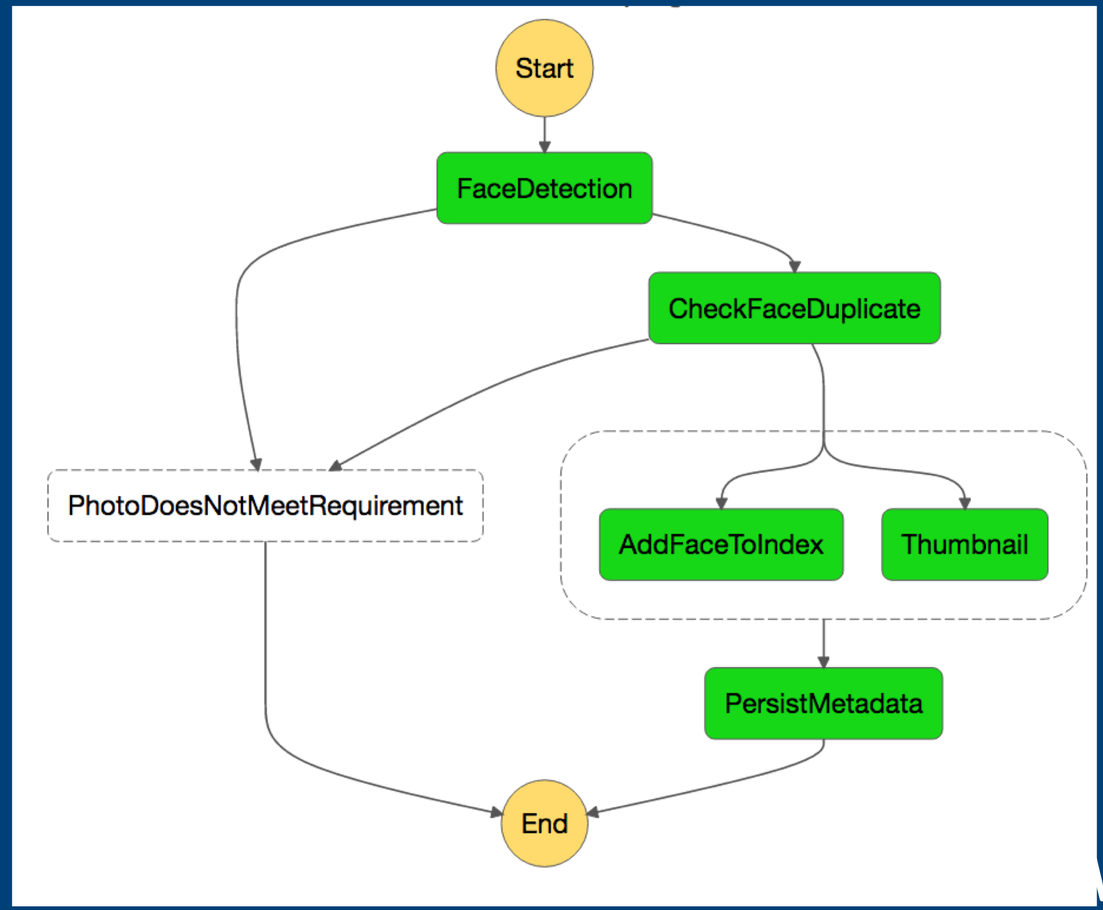

AWS Step Functions


AWS AppSync

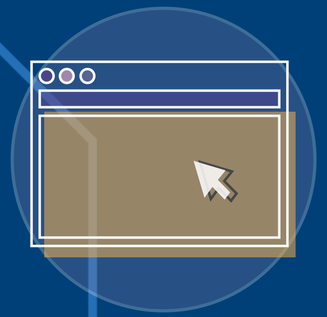
Lab 4 :

https://bit.ly/2Jm8pK3



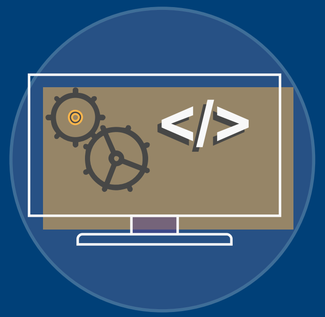


Summary - Lambda use cases



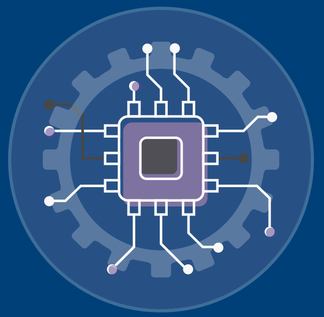
Web Applications

- Static websites
- Complex web apps
- Packages for Flask and Express



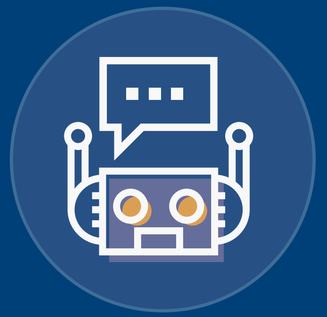
Backends

- Apps & services
- Mobile
- IoT



Data Processing

- Real time
- MapReduce
- Batch

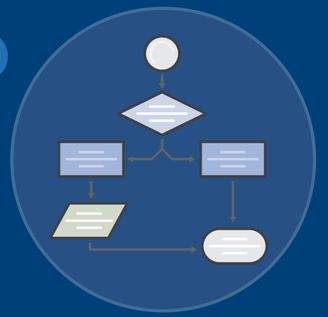


Chatbots

- Powering chatbot logic

Amazon Alexa

- Powering voice-enabled apps
- Alexa Skills Kit



IT Automation

- Policy engines
- Extending AWS services
- Infrastructure management

Lambda Pricing

Free Tier

1M REQUESTS

per month

400,000 GB-SECONDS

of compute time per month.

The Lambda free tier does not automatically expire at the end of your 12 month AWS Free Tier term, but is available to both existing and new AWS customers indefinitely.

Requests

1M REQUESTS FREE

First 1M requests per month are free.

$0.20 PER 1M REQUESTS THEREAFTER

$0.0000002 per request.

Duration

400,000 GB-SECONDS PER MONTH FREE

First 400,000 GB-seconds per month, up to 3.2M seconds of compute time, are free.

$0.00001667 FOR EVERY GB-SECOND USED THEREAFTER

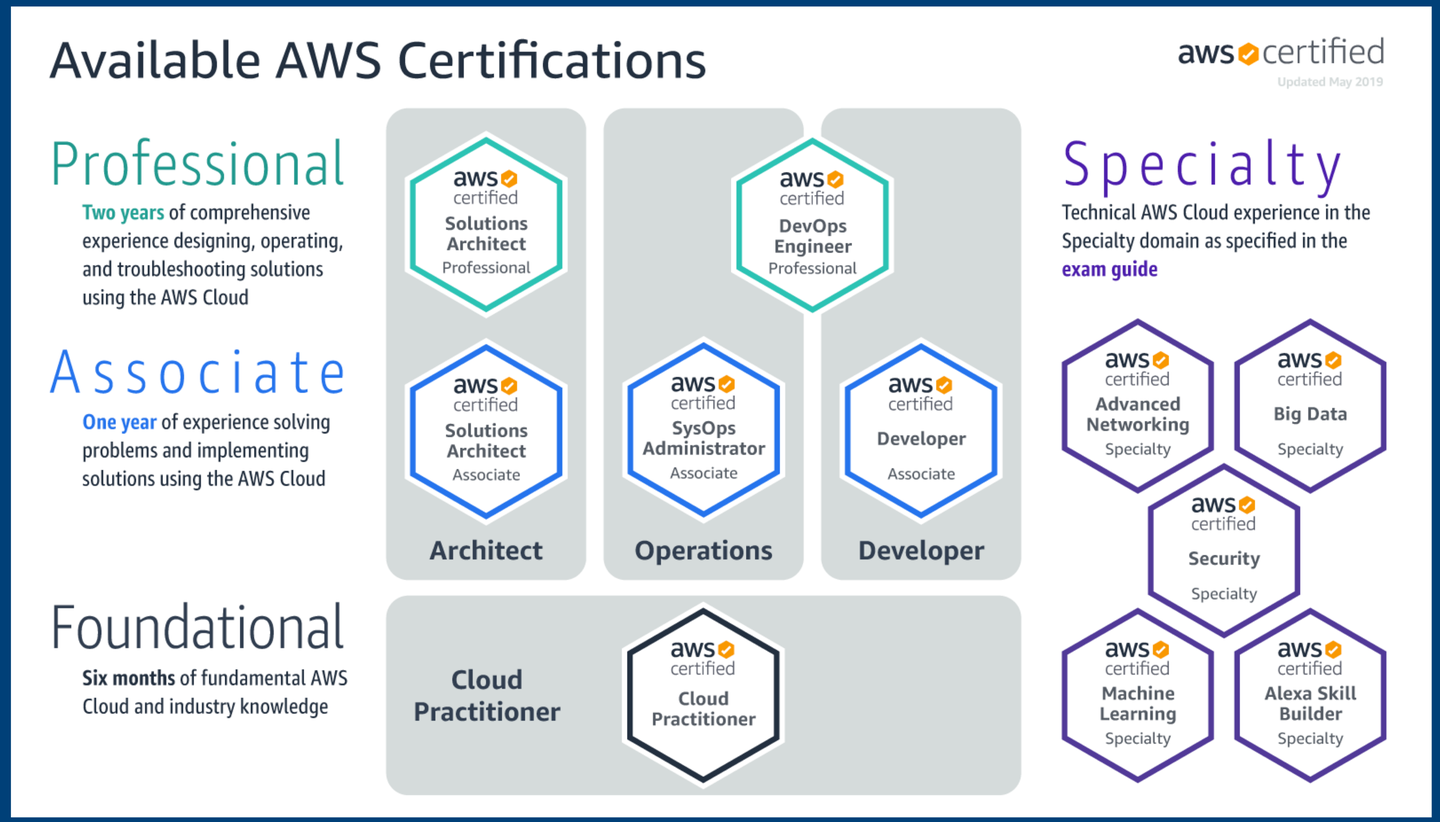
The price depends on the amount of memory you allocate to your function.

What's Next ?

1. **Try Other Labs :**
    A. **DevOps :** https://github.com/aws-samples/aws-serverless-workshops/blob/master/DevOps
    B. **Other Serverless workshop:** https://github.com/aws-samples/aws-serverless-workshops
2. **Read The Developer Guide :**
   https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
3. **Explore Frameworks**
    A. General AWS Serverless Application Model (AWS SAM):
       https://docs.aws.amazon.com/lambda/latest/dg/serverless_app.html
    B. Python:
        - https://www.zappa.io/
        - https://github.com/aws/chalice
    C. Node.js
        - https://github.com/serverless/serverless
        - https://github.com/awslabs/aws-serverless-express
        - https://github.com/claudiajs/claudia

What's Next ?

4. Enroll to AWS Free Digital Training at https://aws.training
5. Get Train in Authorized Training Partner
6. Demonstrate you skill, proof it by getting AWS Certification



What's Next ?

Tell Your Friends to Attend Regular Workshop with AWS

A. Beginner/Intermediate Level:
   1) Launch Your First Workload on AWS

B. Advance Level
   1) App Modernization - Serverless
   2) App Modernization - Container
   3) Big Data & AI/ML on AWS