

# **GENERATING WITTY COMMENTS...**

LET'S GO!



# GETTING TO KNOW THE **AMAZING AMAZON WEB SERVICES**

---

Luigi Libero Lucio Starace

[luigi.starace@gmail.com](mailto:luigi.starace@gmail.com)

June 4, 2018

University of Naples, Federico II

# OUTLINE

## 1 A little bit of context

# OUTLINE

1 A little bit of context

2 An AWS bestiarium

# OUTLINE

- 1 A little bit of context
- 2 An AWS bestiarium
- 3 Serverless architectures

# OUTLINE

- 1 A little bit of context
- 2 An AWS bestiarium
- 3 Serverless architectures
- 4 Demo: a serverless web application

# OUTLINE

- 1 A little bit of context
- 2 An AWS bestiarium
- 3 Serverless architectures
- 4 Demo: a serverless web application
- 5 Take Home Messages

## A LITTLE BIT OF CONTEXT

---

Cloud computing is the on-demand delivery of computing resources through a cloud services platform via the internet with pay-as-you-go pricing.

Cloud computing is the **on-demand delivery** of computing resources through a cloud services platform via the internet with pay-as-you-go pricing.

Cloud computing is the **on-demand delivery** of computing resources through a cloud services platform via the internet with **pay-as-you-go** pricing.

- Software as a Service (SaaS)

## ■ Software as a Service (SaaS)

The service vendor provides the user with a completed product that is run and managed by the service provider.

- **Software as a Service (SaaS)**

The service vendor provides the user with a completed product that is run and managed by the service provider.

- **Platform as a Service (PaaS)**

- **Software as a Service (SaaS)**

The service vendor provides the user with a completed product that is run and managed by the service provider.

- **Platform as a Service (PaaS)**

The service vendor provides the user with a set of API which can be used to build, test and deploy applications.

## ■ Software as a Service (SaaS)

The service vendor provides the user with a completed product that is run and managed by the service provider.

## ■ Platform as a Service (PaaS)

The service vendor provides the user with a set of API which can be used to build, test and deploy applications.

## ■ Infrastructure as a Service (IaaS)

## ■ Software as a Service (SaaS)

The service vendor provides the user with a completed product that is run and managed by the service provider.

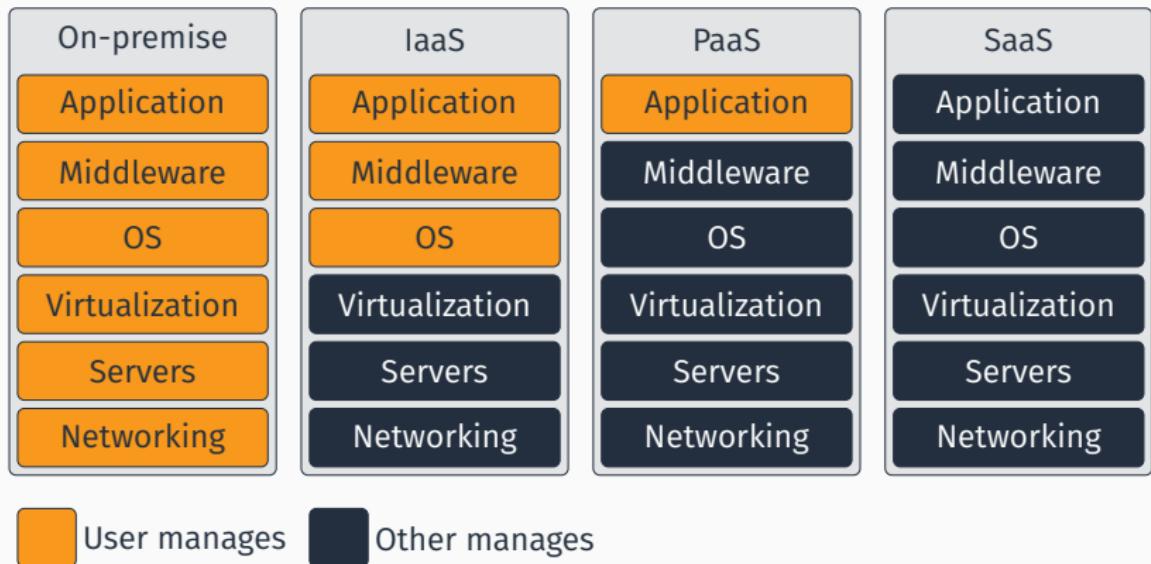
## ■ Platform as a Service (PaaS)

The service vendor provides the user with a set of API which can be used to build, test and deploy applications.

## ■ Infrastructure as a Service (IaaS)

The service vendor provides users access to computing resources such as servers, storage and networking.

# SERVICE MODELS: A VISUAL COMPARISON

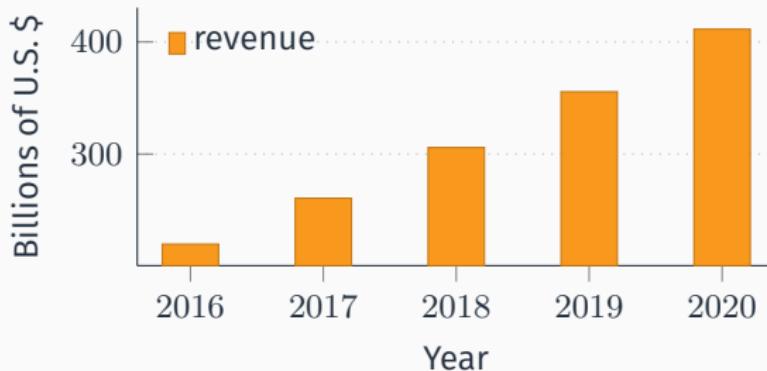


**Figure 1:** A service models comparison

## SOME STATS

Worldwide Public Cloud Services Revenue Forecast (Billions of U.S. Dollars) [Gar17]

2016	2017	2018	2019	2020
219,6	260,6	305,8	355,6	411,4



# THE BIGWIGS

- Google



Google Cloud

## THE BIGWIGS

- Google
- IBM



## THE BIGWIGS

- Google
- IBM
- Microsoft



## THE BIGWIGS

- Google
- IBM
- Microsoft
- Alibaba



## THE BIGWIGS

- Google
- IBM
- Microsoft
- Alibaba
- Oracle

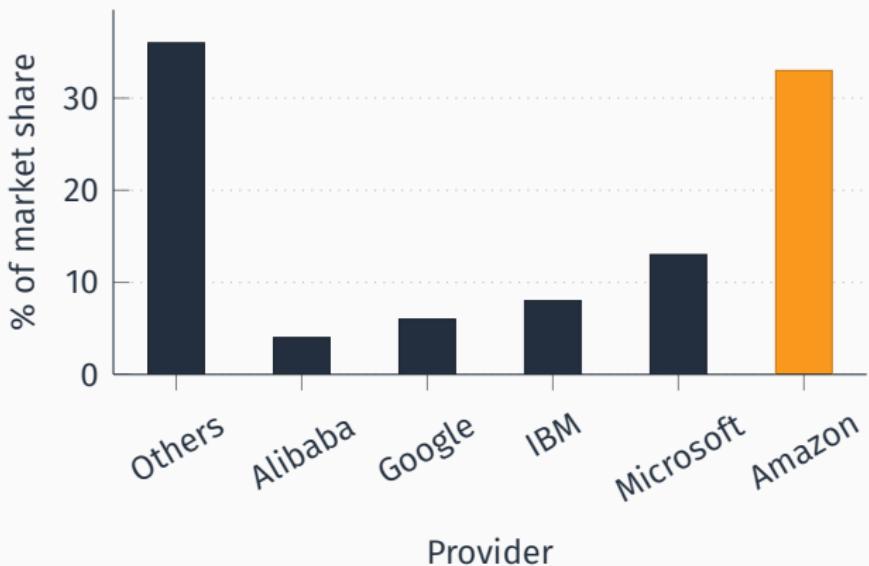


## THE BIGWIGS

- Google
- IBM
- Microsoft
- Alibaba
- Oracle
- Amazon



# MARKET SHARE



**Figure 2:** Market share in Q4 2017 (IaaS, PaaS, Hosted Private Cloud)  
[Syn18]

# AN AWS BESTIARIUM

---



Amazon Web Services is a collection of cloud-based services.



Amazon Web Services is a collection of cloud-based services.  
**A very big one.**



Amazon Web Services is a collection of cloud-based services.  
**A VERY big one.**

# **AN AWS BESTIARIUM**

---

## **DATABASE SERVICES**

# RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.



# RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.



# RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.
- Supports:



# RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.
- Supports:
  - MySQL, PostgreSQL, MariaDB



# RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.
- Supports:
  - MySQL, PostgreSQL, MariaDB
  - Oracle, MS SQL Server



# RELATIONAL DATABASE SERVICE (RDS)

- Set up, operate a relational database in the cloud.
- Takes care of backups, patching.
- Supports:
  - MySQL, PostgreSQL, MariaDB
  - Oracle, MS SQL Server
  - Amazon Aurora



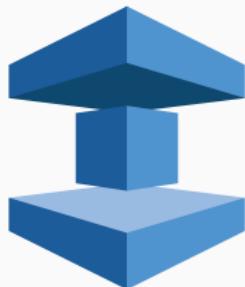
# NON RELATIONAL DATABASE SERVICES

- DynamoDB
  - *Fast and flexible NoSQL database service for any scale.*



# NON RELATIONAL DATABASE SERVICES

- DynamoDB
  - *Fast and flexible NoSQL database service for any scale.*
- ElastiCache
  - In memory data store.
  - Supports memcached, Redis



# NON RELATIONAL DATABASE SERVICES

- DynamoDB
  - *Fast and flexible NoSQL database service for any scale.*
- ElastiCache
  - In memory data store.
  - Supports memcached, Redis
- Neptune
  - Graph database service
  - Supports RDF, SPARQL, ...



## **AN AWS BESTIARIUM**



### **CLOUD STORAGE**

## CLOUD STORAGE PRODUCTS

- Elastic Block Storage (EBS)
  - Persistent local storage for EC2 instances.



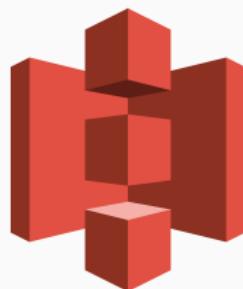
# CLOUD STORAGE PRODUCTS

- Elastic Block Storage (EBS)
  - Persistent local storage for EC2 instances.
- Elastic File System (EFS)
  - File system interface to share data between EC2 instances.



# CLOUD STORAGE PRODUCTS

- Elastic Block Storage (EBS)
  - Persistent local storage for EC2 instances.
- Elastic File System (EFS)
  - File system interface to share data between EC2 instances.
- Simple Storage Service (S3)



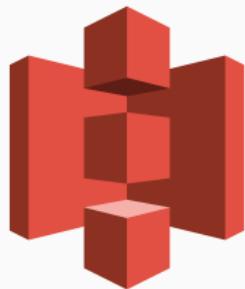
# CLOUD STORAGE PRODUCTS

- Elastic Block Storage (EBS)
  - Persistent local storage for EC2 instances.
- Elastic File System (EFS)
  - File system interface to share data between EC2 instances.
- Simple Storage Service (S3)
- Glacier
  - Durable and cheap long-term storage.



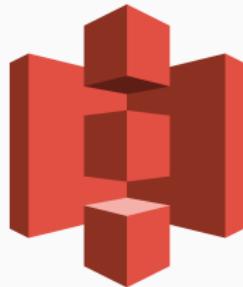
# AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*



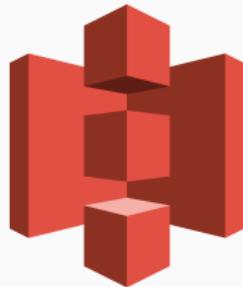
# AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)



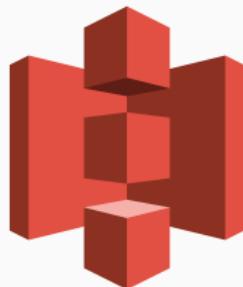
## AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones



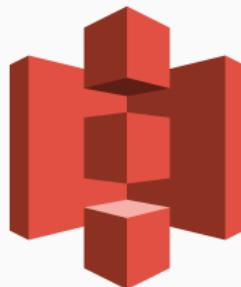
# AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*



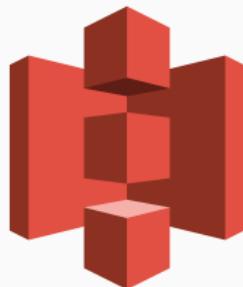
# AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes



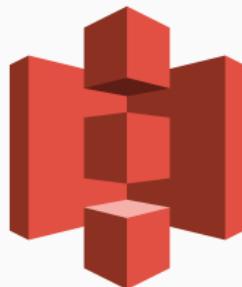
# AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes
  - Standard



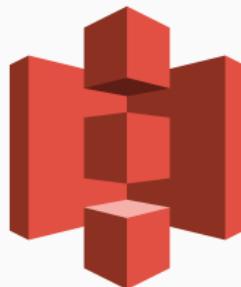
# AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes
  - Standard
  - Infrequent Access



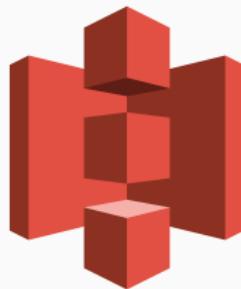
# AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes
  - Standard
  - Infrequent Access
  - One zone-Infrequent Access



# AMAZON SIMPLE STORAGE SERVICE (S3)

- *store and retrieve any amount of data from anywhere*
- 99.99999999% durability (nine nines!)
- Data is distributed across a *minimum* of three availability zones
- A logical unit of storage is a *bucket*
- Multiple storage classes
  - Standard
  - Infrequent Access
  - One zone-Infrequent Access
  - Amazon Glacier



## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

Multiple storage classes

Storage class	Storage	Retrieval (per 1K req.)
Standard	\$0.022 per GB	\$0.0004
Infrequent access	\$0.0125 per GB	\$0.001
IA single zone	\$0.01 per GB	\$0.001

Table 1: S3 pricing (Ireland)

## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services



## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services
  - Machine Learning



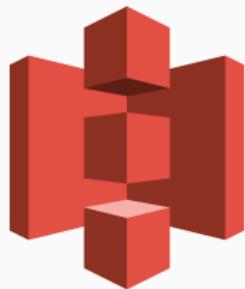
## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services
  - Machine Learning
  - Big Data Analysis



# AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services
  - Machine Learning
  - Big Data Analysis
- REST API



## AMAZON SIMPLE STORAGE SERVICE (S3) - MORE

- Well-integrated with other services
  - Machine Learning
  - Big Data Analysis
- REST API
- Can be used to host static websites



## AN AWS BESTIARIUM

---

### DEVELOPER TOOLS

- CodeCommit



- CodeCommit
  - Managed, scalable, private git server



## ■ CodeCommit

- Managed, scalable, private git server
- Pricing based on active users (5 free,  
1\$ for each additional user)



## ■ CodeCommit

- Managed, scalable, private git server
- Pricing based on active users (5 free,  
1\$ for each additional user)

## ■ CodeBuild



## ■ CodeCommit

- Managed, scalable, private git server
- Pricing based on active users (5 free, 1\$ for each additional user)

## ■ CodeBuild

- Managed, scalable build server



## ■ CodeCommit

- Managed, scalable, private git server
- Pricing based on active users (5 free, 1\$ for each additional user)

## ■ CodeBuild

- Managed, scalable build server
- Pay-per-minute spent building your code



## DEVELOPER TOOLS - MORE

- CodeDeploy



- CodeDeploy

- Automates deployment to computing services (also to instances running on-premise)



### ■ CodeDeploy

- Automates deployment to computing services (also to instances running on-premise)
- Tries to avoid downtime



### ■ CodeDeploy

- Automates deployment to computing services (also to instances running on-premise)
- Tries to avoid downtime
- 0.02\$ per-on-premise deployment

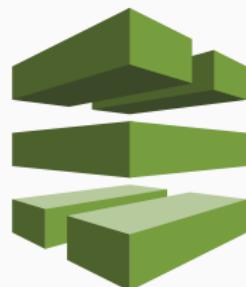


## DEVELOPER TOOLS - MORE

- CodeDeploy

- Automates deployment to computing services (also to instances running on-premise)
- Tries to avoid downtime
- 0.02\$ per-on-premise deployment

- CodePipeline

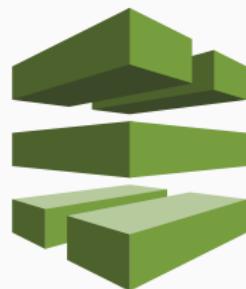


### ■ CodeDeploy

- Automates deployment to computing services (also to instances running on-premise)
- Tries to avoid downtime
- 0.02\$ per-on-premise deployment

### ■ CodePipeline

- Continuous integration e continuous delivery



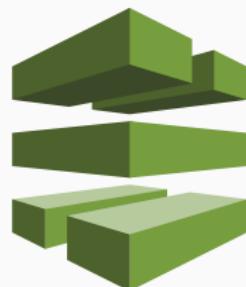
## DEVELOPER TOOLS - MORE

### ■ CodeDeploy

- Automates deployment to computing services (also to instances running on-premise)
- Tries to avoid downtime
- 0.02\$ per-on-premise deployment

### ■ CodePipeline

- Continuous integration e continuous delivery
- Define your own workflow and stages



### ■ CodeDeploy

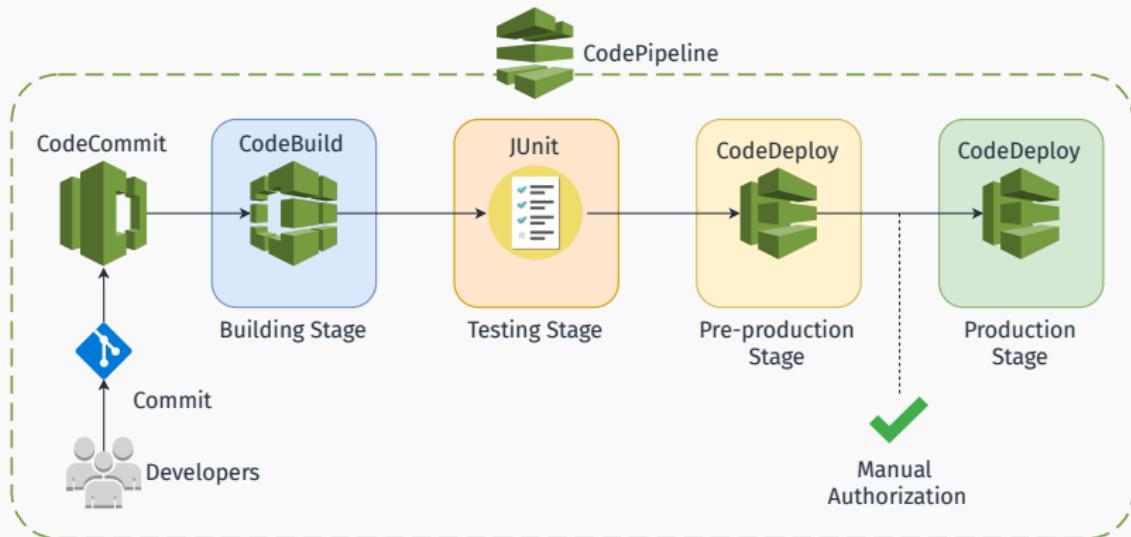
- Automates deployment to computing services (also to instances running on-premise)
- Tries to avoid downtime
- 0.02\$ per-on-premise deployment

### ■ CodePipeline

- Continuous integration e continuous delivery
- Define your own workflow and stages
- 1\$ per-month per active pipeline



# CODEPIPELINE



Interested in CI/CD on AWS? Check these out:

- *Practicing Continuous Integration and Continuous Delivery on AWS* (whitepaper) [[Ama17](#)]
- *Set up a Continuous Deployment Pipeline using AWS CodePipeline* [[Amab](#)]
- *Tutorial: Create a Four-Stage Pipeline* [[Amac](#)]

## DEVELOPER TOOLS - MORE

### ■ CodeStar



## DEVELOPER TOOLS - MORE

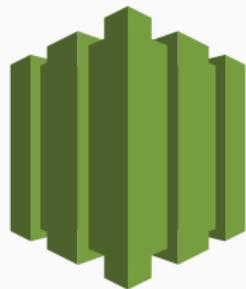
- CodeStar
  - Wrapper around developer tools to simplify setup



## DEVELOPER TOOLS - MORE

- CodeStar

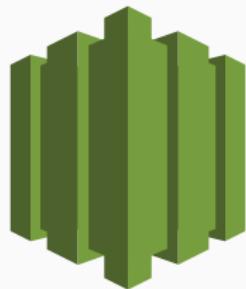
- Wrapper around developer tools to simplify setup
- Templates



## DEVELOPER TOOLS - MORE

- CodeStar

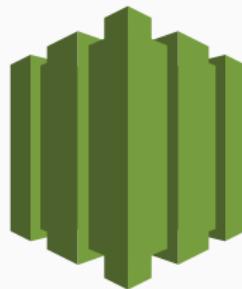
- Wrapper around developer tools to simplify setup
- Templates
- Team Management



## DEVELOPER TOOLS - MORE

### ■ CodeStar

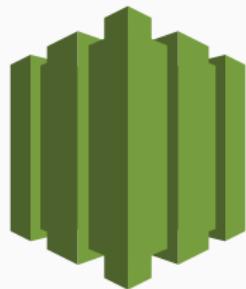
- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard



## DEVELOPER TOOLS - MORE

### ■ CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge



## DEVELOPER TOOLS - MORE

- CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge

- Cloud9



## DEVELOPER TOOLS - MORE

- CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge

- Cloud9

- Cloud-based full-fledged IDE



## DEVELOPER TOOLS - MORE

- CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge

- Cloud9

- Cloud-based full-fledged IDE
- Runs in a web browser



## DEVELOPER TOOLS - MORE

### ■ CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge

### ■ Cloud9

- Cloud-based full-fledged IDE
- Runs in a web browser
- Collaborative editing and chat



## DEVELOPER TOOLS - MORE

### ■ CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge

### ■ Cloud9

- Cloud-based full-fledged IDE
- Runs in a web browser
- Collaborative editing and chat
- Greatly-integrated with AWS



## DEVELOPER TOOLS - MORE

### ■ CodeStar

- Wrapper around developer tools to simplify setup
- Templates
- Team Management
- Central Project Dashboard
- Free of charge

### ■ Cloud9

- Cloud-based full-fledged IDE
- Runs in a web browser
- Collaborative editing and chat
- Greatly-integrated with AWS
- Free of charge



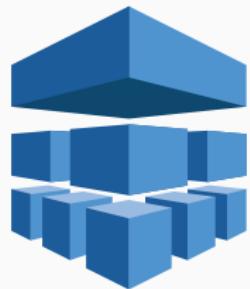
## **AN AWS BESTIARIUM**

---

### **MACHINE LEARNING**

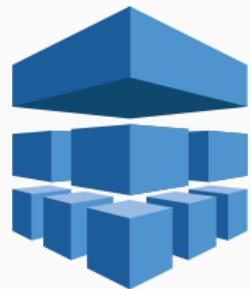
# MACHINE LEARNING: PLATFORM SERVICES

- Amazon SageMaker



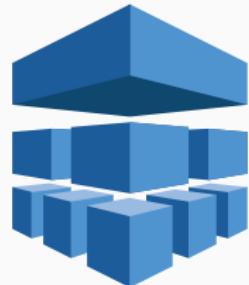
# MACHINE LEARNING: PLATFORM SERVICES

- Amazon SageMaker
  - Preconfigured for Tensorflow, MXNet...



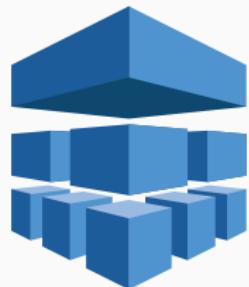
# MACHINE LEARNING: PLATFORM SERVICES

- Amazon SageMaker
  - Preconfigured for Tensorflow, MXNet...
  - Build, Train and Deploy phases



- Amazon SageMaker

- Preconfigured for Tensorflow, MXNet...
- Build, Train and Deploy phases
- Pay based on build time, train time and hosting time



# MACHINE LEARNING: APPLICATION SERVICES

- Comprehend (for NLP) [!\[\]\(9a70d4a0fb50e2d0d3ecaa2339b6399b\_img.jpg\) Dashboard](#)



# MACHINE LEARNING: APPLICATION SERVICES

- Comprehend (for NLP) [!\[\]\(995676b7cf637a7b2955f43d080e8365\_img.jpg\) Dashboard](#)
- Rekognition (Visual Analysis) [!\[\]\(2b87b3c7383f25f95a15008dfe80421a\_img.jpg\) Dashboard](#)



# MACHINE LEARNING: APPLICATION SERVICES

- Comprehend (for NLP) [!\[\]\(0d114b8369b12192285ab4f5912492ef\_img.jpg\) Dashboard](#)
- Rekognition (Visual Analysis) [!\[\]\(bd854b137ffc951f612f614e59402ee2\_img.jpg\) Dashboard](#)
- Translate

# MACHINE LEARNING: APPLICATION SERVICES

- Comprehend (for NLP) [!\[\]\(9f0f8193deabcd24f8fafca184947e2f\_img.jpg\) Dashboard](#)
- Rekognition (Visual Analysis) [!\[\]\(c94cf346764706e1a46749cafdce8aa8\_img.jpg\) Dashboard](#)
- Translate
- Polly (text-to-speech)

# MACHINE LEARNING: APPLICATION SERVICES

- Comprehend (for NLP) [!\[\]\(e2057f2cdcf4928a5180f709a4b10c93\_img.jpg\) Dashboard](#)
- Rekognition (Visual Analysis) [!\[\]\(e92ee5caa04755377be9c4c4b01d564c\_img.jpg\) Dashboard](#)
- Translate
- Polly (text-to-speech)
- Transcribe (speech-to-text)

# **AN AWS BESTIARIUM**

---

**MISCELLANEA**

- Cognito



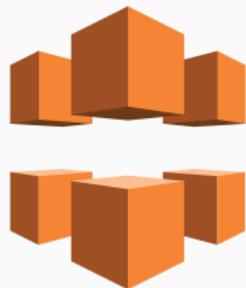
- Cognito
  - Sign-up and authentication



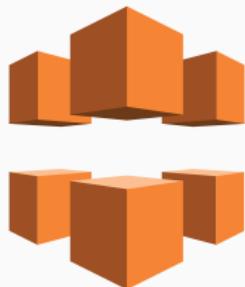
- Cognito
  - Sign-up and authentication
  - Federated identities



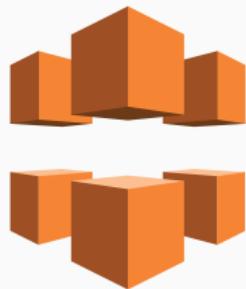
- Cognito
  - Sign-up and authentication
  - Federated identities
- CloudFront



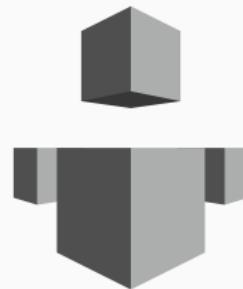
- Cognito
  - Sign-up and authentication
  - Federated identities
- CloudFront
  - Content Delivery Network



- Cognito
  - Sign-up and authentication
  - Federated identities
- CloudFront
  - Content Delivery Network
  - 116 Points of Presence in 56 cities across 24 countries

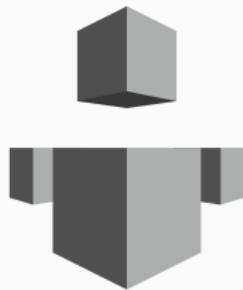


- Cognito
  - Sign-up and authentication
  - Federated identities
- CloudFront
  - Content Delivery Network
  - 116 Points of Presence in 56 cities across 24 countries
- Mechanical Turk



# AMAZON MECHANICAL TURK

- ???

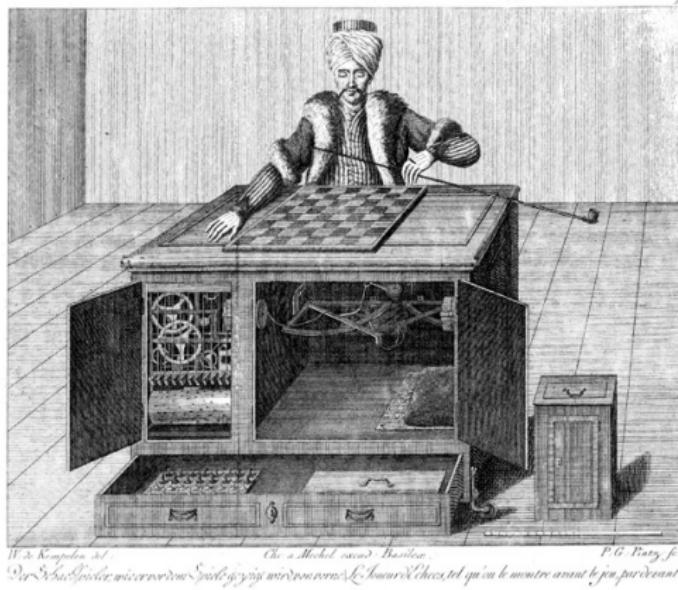


## THE TURK

*The Turk* was a chess-playing automaton built in 1770.

# THE TURK

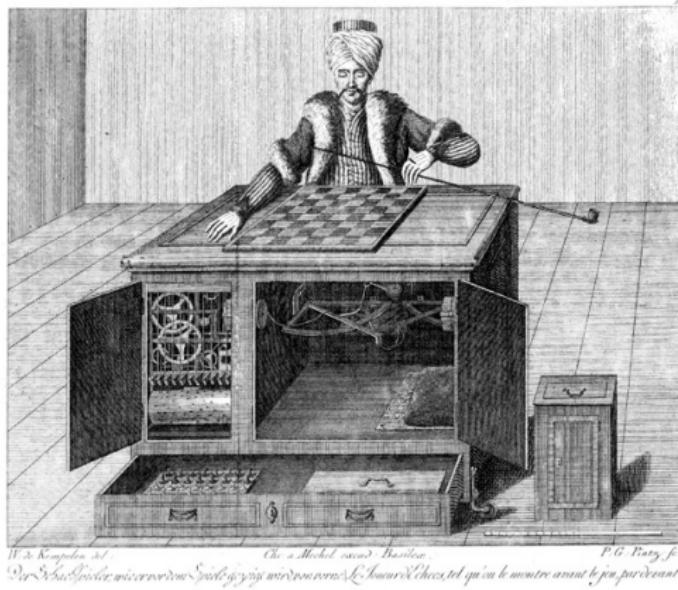
*The Turk* was a chess-playing automaton built in 1770.



W. de Kempelen ac.  
Ob. a Michel mecad. Basileus.  
D'après la Peinture qui servit à la construction de l'Automate d'Echecs, tel qu'il le montra avant le jeu précédent.  
P.G. Remy sc:

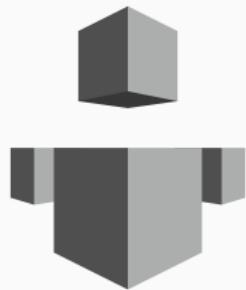
# THE TURK

*The Turk* was a chess-playing automaton built in 1770.  
Obviously it was a fraud.

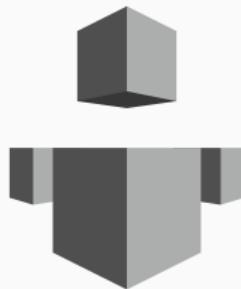


# AMAZON MECHANICAL TURK

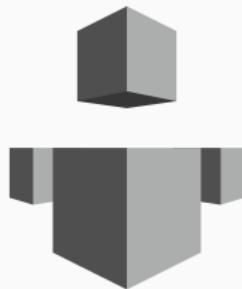
- ???



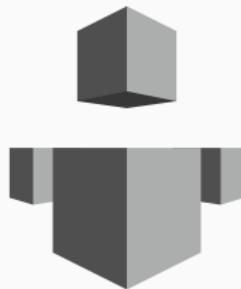
- Human Intelligence through an API



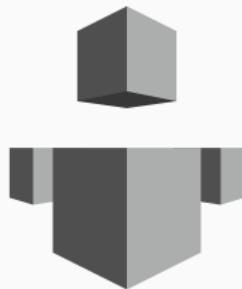
- Human Intelligence through an API
- Create HIT (Human Intelligence Task)



- Human Intelligence through an API
- Create HIT (Human Intelligence Task)
- Elastic, on-demand workforce



- Human Intelligence through an API
- Create HIT (Human Intelligence Task)
- Elastic, on-demand workforce
- Available 24/7



# **AN AWS BESTIARIUM**

---

**COMPUTING**

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand



---

Azure: Virtual Machines [web](#)

Google Cloud: Compute Engine [web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs



---

Azure: Virtual Machines [web](#)

Google Cloud: Compute Engine [web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing



---

Azure: Virtual Machines [web](#)

Google Cloud: Compute Engine [web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing
- Data transfer **not** included!



---

Azure: Virtual Machines [web](#)

Google Cloud: Compute Engine [web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing
- Data transfer **not** included!
- Persistent storage **not** included!



---

Azure: Virtual Machines [web](#)

Google Cloud: Compute Engine [web](#)

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing
- Data transfer **not** included!
- Persistent storage **not** included!

-  EBS/EFS



---

Azure: Virtual Machines  web

Google Cloud: Compute Engine  web

# AMAZON ELASTIC COMPUTE CLOUD (EC2)

- (Virtual) Servers on demand
- Different types of instances to suit computing needs
- Per-second (or per-hour) billing
- Data transfer **not** included!
- Persistent storage **not** included!
  -  EBS/EFS
- Scaling **not** included!



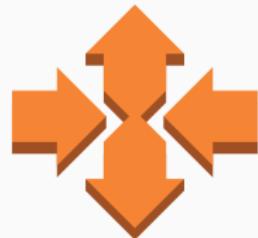
---

Azure: Virtual Machines  web

Google Cloud: Compute Engine  web

# AMAZON EC2 AUTO SCALING

- *Scaling is the ability to increase or decrease the compute capacity of your application*

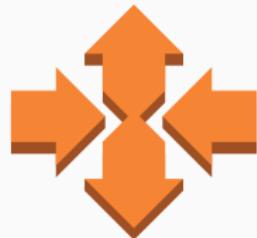


---

Azure: Virtual Machine Scale Sets [web](#)  
Google Cloud: Load Balancing [web](#)

# AMAZON EC2 AUTO SCALING

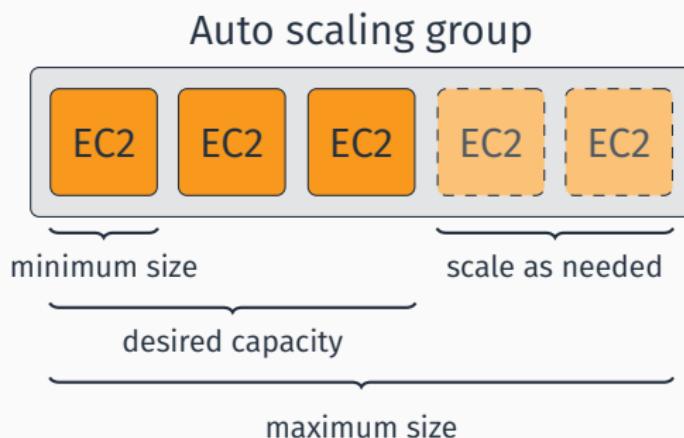
- *Scaling is the ability to increase or decrease the compute capacity of your application*
- Scale your application manually, on a scheduled basis or on demand



---

Azure: Virtual Machine Scale Sets [web](#)  
Google Cloud: Load Balancing [web](#)

# AMAZON EC2 AUTO SCALING: DETAILS



# AMAZON ELASTIC LOAD BALANCING (ELB)

- Distributes incoming traffic across multiple EC2 instances



---

Azure: Load Balancer  web

# AMAZON ELASTIC LOAD BALANCING (ELB)

- Distributes incoming traffic across multiple EC2 instances
- Pay-per-use billing



---

Azure: Load Balancer  web

# AMAZON ELASTIC LOAD BALANCING (ELB)

- Distributes incoming traffic across multiple EC2 instances
- Pay-per-use billing
  - Execution time



---

Azure: Load Balancer web

# AMAZON ELASTIC LOAD BALANCING (ELB)

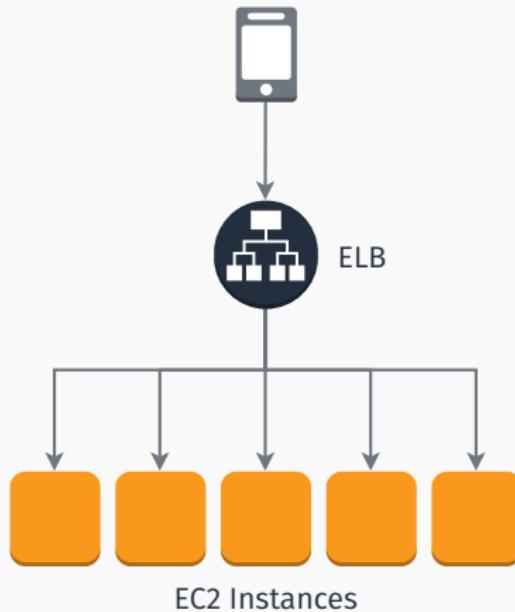
- Distributes incoming traffic across multiple EC2 instances
- Pay-per-use billing
  - Execution time
  - Number of requests / traffic



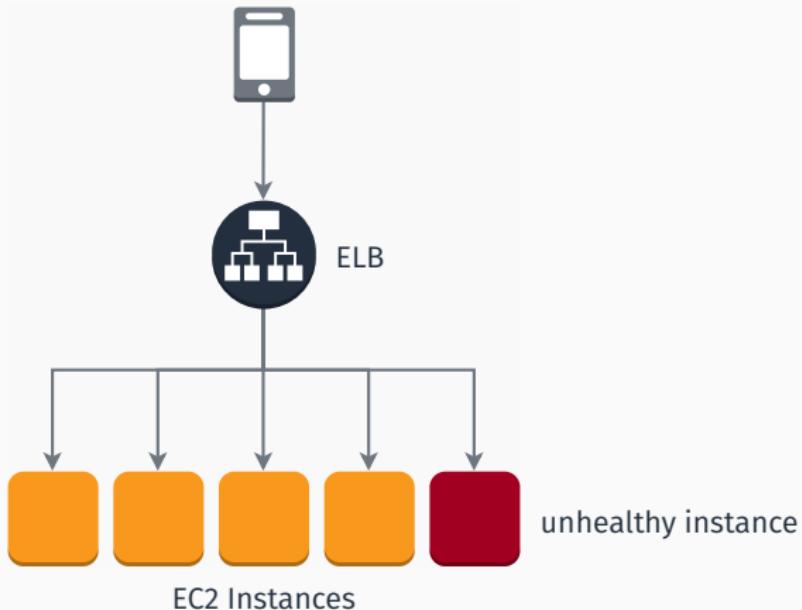
---

Azure: Load Balancer  web

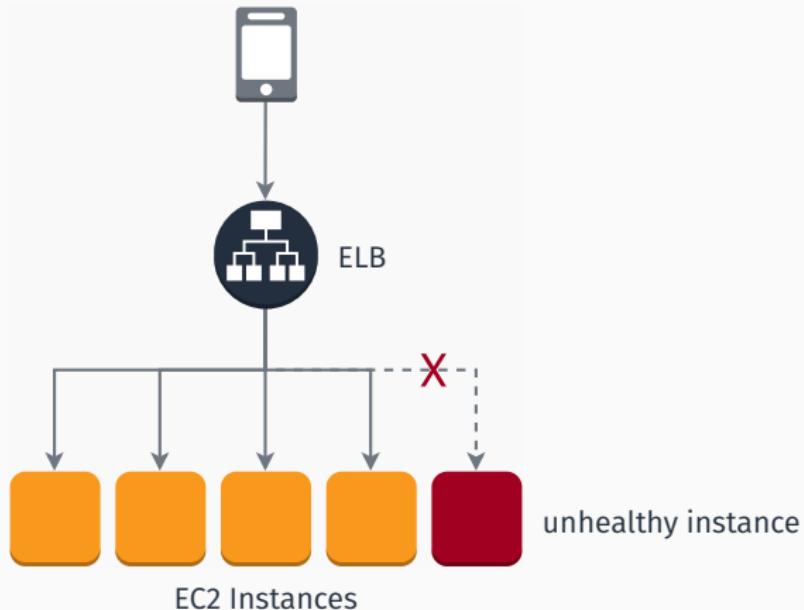
# AMAZON ELASTIC LOAD BALANCING (ELB)



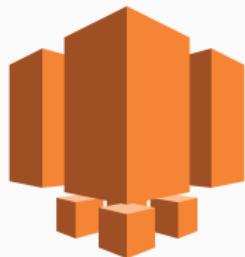
# AMAZON ELASTIC LOAD BALANCING (ELB)



# AMAZON ELASTIC LOAD BALANCING (ELB)



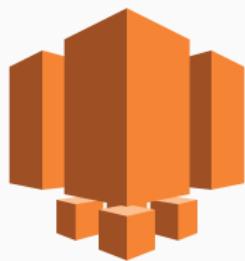
- A lightweight, simplified offer



---

Websites: [EC2](#) [Lightsail](#)

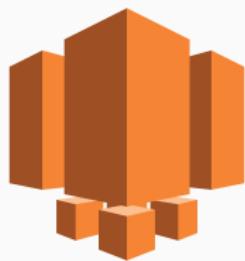
- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity



---

Websites: [EC2](#) [Lightsail](#)

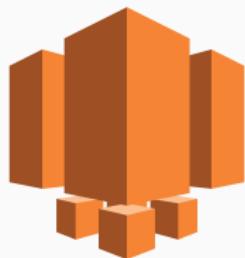
- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for



---

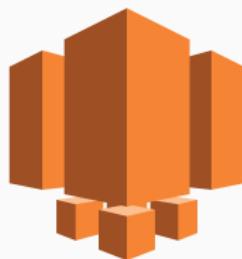
Websites: [EC2](#) [Lightsail](#)

- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for
  - Debian, Windows Server, ...



Websites: [EC2](#) [Lightsail](#)

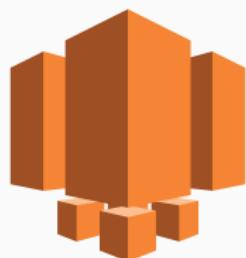
- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for
  - Debian, Windows Server, ...
  - Wordpress, Magento, Redmine, ...



---

Websites: [EC2](#) [Lightsail](#)

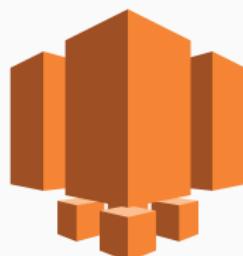
- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for
  - Debian, Windows Server, ...
  - Wordpress, Magento, Redmine, ...
  - LAMP stack, Nginx, ...



---

Websites: [EC2](#) [Lightsail](#)

- A lightweight, simplified offer
- Bundles computing, storage, and networking capacity
- Preconfigured instances for
  - Debian, Windows Server, ...
  - Wordpress, Magento, Redmine, ...
  - LAMP stack, Nginx, ...
- Low and **predictable** monthly costs



---

Websites: [EC2](#) [Lightsail](#)

# AMAZON ELASTIC BEANSTALK

- “Easy to begin, impossible to outgrow”



# AMAZON ELASTIC BEANSTALK

- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps



# AMAZON ELASTIC BEANSTALK

- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more



# AMAZON ELASTIC BEANSTALK

- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more
- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker



- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more
- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker
- Manages auto-scaling, load balancing, health monitoring



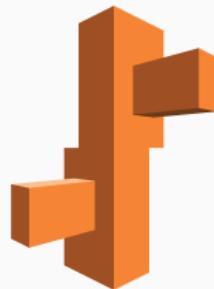
# AMAZON ELASTIC BEANSTALK

- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more
- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker
- Manages auto-scaling, load balancing, health monitoring
- Customizable



# AMAZON ELASTIC BEANSTALK

- “Easy to begin, impossible to outgrow”
- Easy-to-use service to deploy web apps
- Supports Apache, Nginx, IIS and more
- Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker
- Manages auto-scaling, load balancing, health monitoring
- Customizable
- Free of charge. Pay only for the AWS resources you use.



## A LITTLE RECAP

So far we've seen:

## A LITTLE RECAP

So far we've seen:

- Elastic Compute Cloud (EC2)

## A LITTLE RECAP

So far we've seen:

- Elastic Compute Cloud (EC2)
  - Auto-scaling, Elastic Load Balancing

## A LITTLE RECAP

So far we've seen:

- Elastic Compute Cloud (EC2)
  - Auto-scaling, Elastic Load Balancing
- Lightsail

## A LITTLE RECAP

So far we've seen:

- Elastic Compute Cloud (EC2)
  - Auto-scaling, Elastic Load Balancing
- Lightsail
- Elastic Beanstalk

## A LITTLE RECAP

So far we've seen:

- Elastic Compute Cloud (EC2)
  - Auto-scaling, Elastic Load Balancing
- Lightsail
- Elastic Beanstalk

We have to (somewhat) care about the infrastructure!



It's demo time!

## WHAT WE'RE GOING TO DO IN THIS DEMO

- Checkout a very simple web application written in PHP

## WHAT WE'RE GOING TO DO IN THIS DEMO

- Checkout a very simple web application written in PHP
- Run it locally (optional)

## WHAT WE'RE GOING TO DO IN THIS DEMO

- Checkout a very simple web application written in PHP
- Run it locally (optional)
- Deploy it to the cloud using Amazon Elastic Beanstalk

## WHAT WE'RE GOING TO DO IN THIS DEMO

- Checkout a very simple web application written in PHP
- Run it locally (optional)
- Deploy it to the cloud using Amazon Elastic Beanstalk
- Doable in 30 minutes at home.

## ABOUT THE WEB APP

We'll deploy a very simple website for this very talk. The web app has two pages:

## ABOUT THE WEB APP

We'll deploy a very simple website for this very talk. The web app has two pages:

- a substantially static **homepage**

## ABOUT THE WEB APP

We'll deploy a very simple website for this very talk. The web app has two pages:

- a substantially static **homepage**
- a **comment** page allowing users to leave feedbacks.

## ABOUT THE WEB APP

We'll deploy a very simple website for this very talk. The web app has two pages:

- a substantially static **homepage**
- a **comment** page allowing users to leave feedbacks.

## ABOUT THE WEB APP

We'll deploy a very simple website for this very talk. The web app has two pages:

- a substantially static **homepage**
- a **comment** page allowing users to leave feedbacks.

Technologies involved:

- Symfony framework

## ABOUT THE WEB APP

We'll deploy a very simple website for this very talk. The web app has two pages:

- a substantially static **homepage**
- a **comment** page allowing users to leave feedbacks.

Technologies involved:

- Symfony framework
- Doctrine ORM

## ABOUT THE WEB APP

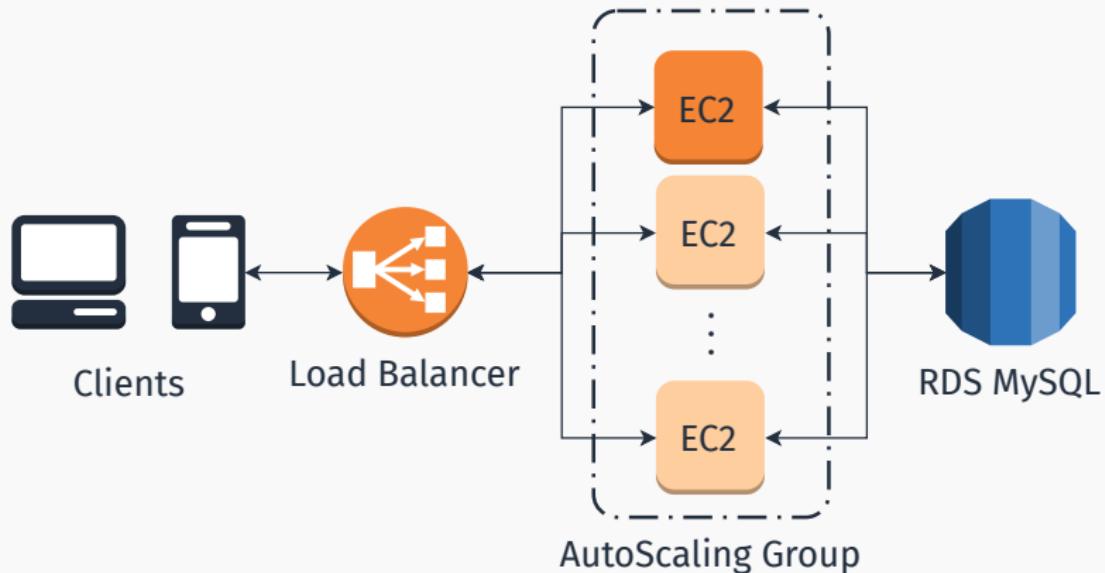
We'll deploy a very simple website for this very talk. The web app has two pages:

- a substantially static **homepage**
- a **comment** page allowing users to leave feedbacks.

Technologies involved:

- Symfony framework
- Doctrine ORM
- Webpack, Sass

# ARCHITECTURE



▶ Skip tutorial

## WHAT YOU'LL NEED

- An AWS account (a free one will suffice)

## WHAT YOU'LL NEED

- An AWS account (a free one will suffice)
- git version control (recommended)

## WHAT YOU'LL NEED

- An AWS account (a free one will suffice)
- git version control (recommended)
- If you want to build and run the app locally:

## WHAT YOU'LL NEED

- An AWS account (a free one will suffice)
- git version control (recommended)
- If you want to build and run the app locally:
  - An AMP (Apache, PHP  $\geq$  7.1.3, MySQL  $\geq$  5.7) stack

## WHAT YOU'LL NEED

- An AWS account (a free one will suffice)
- git version control (recommended)
- If you want to build and run the app locally:
  - An AMP (Apache, PHP  $\geq$  7.1.3, MySQL  $\geq$  5.7) stack
  - Composer package manager

## WHAT YOU'LL NEED

- An AWS account (a free one will suffice)
- git version control (recommended)
- If you want to build and run the app locally:
  - An AMP (Apache, PHP  $\geq$  7.1.3, MySQL  $\geq$  5.7) stack
  - Composer package manager
  - Node.js

## STEP 1: GET THE APP

Clone the git repository



```
D:\Desktop> git clone https://github.com/luistar/  
serverful-webapp.git serverful-webapp
```

## STEP 2: INSTALL DEPENDENCIES

Install dependencies with composer

```
D:\Desktop> cd serverful-webapp
```

```
D:\Desktop\serverful-webapp> composer install
```

Then install Node.js dependecies

```
D:\Desktop\serverful-webapp> npm install
```

## STEP 3: CONFIGURATION PARAMETERS

Start your database instance and create an user for the webapp. Once you are done, update the configuration file config/packages/database-config.php accordingly.

```
2 //get parameter from environment or fallback to defaults
3 $db_host = (
4     (isset($_SERVER['RDS_HOST'])) ?
5         ($_SERVER['RDS_HOST']) : ('localhost')
6 );
7 /* And following lines */
```

In config/services.yaml replace the dummy text with your Google Maps API Key.

```
1 parameters:
2     locale: 'en'
3     app.gmaps_api_key: '<YOUR GMAPS API KEY HERE>'
```

## STEP 4: BUILD ASSETS AND CREATE DATABASE SCHEMA

Build assets with

```
D:\Desktop\serverful-webapp> npm run webpack-dev
```

Then create the database and the data schema by running

```
D:\Desktop\serverful-webapp> npm run drop-database  
D:\Desktop\serverful-webapp> npm run create-database  
D:\Desktop\serverful-webapp> npm run create-schema
```

## STEP 5: RUN THE APP

Now you can start the dev server anche check out the app.

```
D:\Desktop\serverful-webapp> npm run serve
```

Once the server started, visit the webapp at localhost:8000

## STEP 6: CREATE A SOURCE BUNDLE

Elastic Beanstalk requires a single WAR or ZIP archive containing your app. To create a source bundle for our app, run

```
D:\Desktop\serverful-webapp> npm run create-source-bundle
```

A `serverful-app.zip` (our source bundle) archive will be created in the app root.

## STEP 7: CREATE A DATABASE INSTANCE

Go to the RDS Console and select “instances” .

The screenshot shows the AWS RDS (Amazon Relational Database Service) console. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a star icon, a bell icon, and user names 'Luigi Starace' and 'Frank'. Below the navigation bar, the main title 'Amazon RDS' is displayed with a close button. On the left, a sidebar menu lists several options: 'Dashboard' (highlighted in orange), 'Instances' (with a cursor icon pointing to it), 'Clusters', 'Performance Insights' (with a 'PREVIEW' badge), 'Snapshots', 'Reserved instances', 'Subnet groups', 'Parameter groups', 'Option groups', 'Events', 'Event subscriptions', and 'Notifications'. The main content area features a large blue circular icon at the top, followed by the text 'Amazon Relational Database Service' in bold. Below this, a descriptive paragraph reads: 'Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale relational databases in the cloud. It provides cost-efficient and resizable'. At the bottom of the page, there are links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'.

# STEP 7: CREATE A DATABASE INSTANCE

Select “Launch DB instance” .

The screenshot shows the AWS RDS Instances page. The left sidebar menu includes options like Dashboard, Instances (which is selected and highlighted in orange), Clusters, Performance Insights, Snapshots, Reserved instances, Subnet groups, Parameter groups, Option groups, Events, Event subscriptions, and Notifications. The main content area displays the 'Instances (0)' section with a 'Launch DB instance' button, which is highlighted with a mouse cursor. Below it is a search bar labeled 'Filter instances' and a table header 'DB instance'. At the bottom of the page, there are links for Feedback, English (US), Privacy Policy, and Terms of Use.

# STEP 7: CREATE A DATABASE INSTANCE

Select MySQL DBMS.

Screenshot of the AWS RDS "Launch DB instance" wizard, Step 1 of 4: Select engine.

The "Engine options" section shows four choices:

- Amazon Aurora (selected)
- MySQL (highlighted with a blue border and cursor icon)
- MariaDB (with a lizard icon)
- PostgreSQL (with a blue P icon)

Navigation bar at the top: Services > Instances > Launch DB instance.

Footer navigation: Feedback, English (US), Privacy Policy, Terms of Use.

## STEP 7: CREATE A DATABASE INSTANCE

Enable only free-tier options and continue.

The screenshot shows the AWS RDS 'Create Database Instance' wizard at Step 7: Database Engine Selection. The MySQL engine is selected. Below it, a detailed description of MySQL is provided, followed by a bulleted list of its features. At the bottom, there is a checkbox for enabling free-tier options and a prominent orange 'Next' button.

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 16 TB.
- Instances offer up to 32 vCPUs and 244 GiB Memory.
- Supports automated backup and point-in-time recovery.
- Supports cross-region read replicas.

Only enable options eligible for RDS Free Usage Tier [Info](#)

[Cancel](#) [Next](#)

[Feedback](#) [English \(US\)](#) [Privacy Policy](#) [Terms of Use](#)

# STEP 7: CREATE A DATABASE INSTANCE

Select MySQL version 5.7.21

The screenshot shows the AWS RDS 'Launch DB instance' wizard, Step 2 of 3. The title is 'Specify DB details'. The 'Instance specifications' section is highlighted. It shows the following settings:

- DB engine: MySQL Community Edition
- License model: general-public-license
- DB engine version: mysql 5.7.21

At the bottom, there are links for 'Feedback', 'Known Issues/Limitations', 'Privacy Policy', and 'Terms of Use'.

# STEP 7: CREATE A DATABASE INSTANCE

Select db.t2.micro instance.

The screenshot shows the AWS RDS 'Create Database Instance' wizard. At the top, there's a banner for the 'Free tier' which states: 'The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions [here](#)'. Below this, a checkbox is checked with the label 'Only enable options eligible for RDS Free Usage Tier' and a 'Info' link.

**DB instance class** [Info](#)  
db.t2.micro — 1 vCPU, 1 GiB RAM

**Multi-AZ deployment** [Info](#)  
 Create replica in different zone  
Creates a replica in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.  
 No

**Storage type** [Info](#)  
General Purpose (SSD)

**Allocated storage**  
20  GB  
(Minimum: 20 GB, Maximum: 20 GB) Higher allocated storage [may improve](#) IOPS performance.

[Feedback](#) [English \(US\)](#) [Privacy Policy](#) [Terms of Use](#)

## STEP 7: CREATE A DATABASE INSTANCE

Enter your desired settings (remember the password! **A**).

The screenshot shows the AWS Management Console interface for creating a database instance. The top navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, and user names 'Luigi Starace' and 'Frank'. The main content area has a left sidebar with a 'Settings' tab selected. The 'DB instance identifier' field contains 'serverful-webapp-db'. Below it, a note specifies the identifier must be unique and contain 1 to 63 alphanumeric characters or hyphens. The 'Master username' field contains 'serverfulwebapp', with a note that it must start with a letter and contain 1 to 16 alphanumeric characters. The 'Master password' and 'Confirm password' fields both show masked input. A note below these fields states the password must be at least eight characters long and can include any printable ASCII character except '/', ' ', or '@'. At the bottom, there are 'Cancel', 'Previous', and 'Next' buttons, with 'Next' being highlighted in orange. The footer includes 'Feedback', language selection ('English (US)'), and links to 'Privacy Policy' and 'Terms of Use'.

DB instance identifier [Info](#)  
Specify a name that is unique for all DB instances owned by your AWS account in the current region.

DB Instance identifier is case insensitive, but stored as all lower-case, as in "mydbinstance". Must contain from 1 to 63 alphanumeric characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Cannot end with a hyphen or contain two consecutive hyphens.

Master username [Info](#)  
Specify an alphanumeric string that defines the login ID for the master user.

Master Username must start with a letter. Must contain 1 to 16 alphanumeric characters.

Master password [Info](#)

Confirm password [Info](#)

Master Password must be at least eight characters long, as in "mypassword". Can be any printable ASCII character except "/", " ", or "@".

[Cancel](#) [Previous](#) [Next](#)

[Feedback](#) [English \(US\)](#) [Privacy Policy](#) [Terms of Use](#)

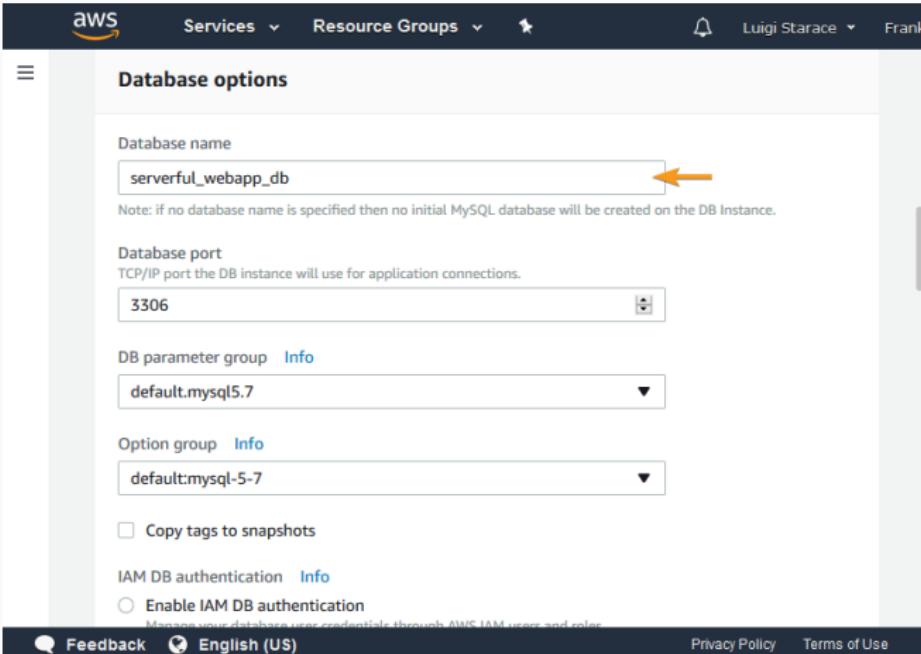
## STEP 7: CREATE A DATABASE INSTANCE

Be sure to select “create a new security group”.

The screenshot shows the AWS Management Console interface for creating a database instance. The top navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, and user names 'Luigi Starace' and 'Frank'. The main content area is titled 'Subnet group' with an 'Info' link. It describes a DB subnet group defining which subnets and IP ranges the DB instance can use in the VPC. A dropdown menu is open, showing 'default' as the current selection. Below this, the 'Public accessibility' section is shown, with the 'Yes' radio button selected. The description states that EC2 instances and devices outside the VPC will connect to the DB instances, and users must also select one or more VPC security groups. The 'No' option would prevent public connectivity. The 'Availability zone' section follows, with an 'Info' link and a dropdown menu showing 'No preference'. At the bottom, the 'VPC security groups' section is present, with a description of security groups authorizing connections from EC2 instances. Two options are available: 'Create new VPC security group' (selected, indicated by a blue circular icon with a hand cursor) and 'Choose existing VPC security groups'.

## STEP 7: CREATE A DATABASE INSTANCE

Enter a database name for the instance (**important!**  ) and leave the rest as is.



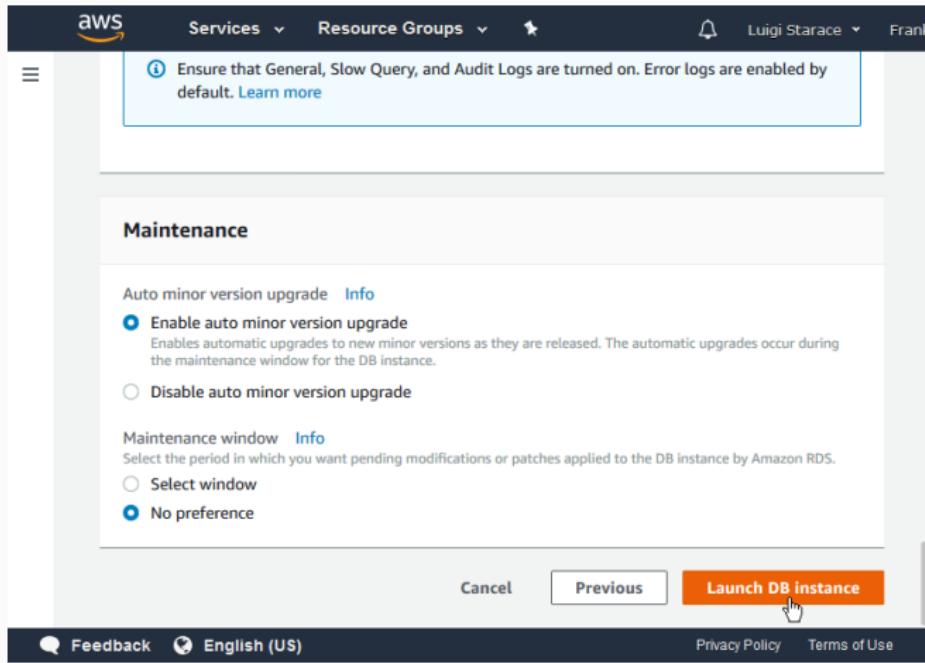
The screenshot shows the AWS Management Console interface for creating a database instance. The top navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a search bar, and user profile icons for Luigi Starace and Frank. The main content area is titled 'Database options'. It contains the following fields:

- Database name:** A text input field containing 'serverful\_webapp\_db', which has an orange arrow pointing to its right side.
- Note:** A small note below the database name field states: "Note: if no database name is specified then no initial MySQL database will be created on the DB Instance."
- Database port:** A text input field containing '3306'.
- DB parameter group:** A dropdown menu set to 'default.mysql5.7'.
- Option group:** A dropdown menu set to 'default:mysql-5.7'.
- Copy tags to snapshots:** An unchecked checkbox.
- IAM DB authentication:** A section with a radio button labeled 'Enable IAM DB authentication'. Below it, a small note says: "Manage your databases user credentials through AWS IAM users and roles".

At the bottom of the page, there are links for 'Feedback', language selection ('English (US)'), and legal links ('Privacy Policy', 'Terms of Use').

# STEP 7: CREATE A DATABASE INSTANCE

Click on “Launch DB Instance”.



## STEP 7: CREATE A DATABASE INSTANCE

The creation process takes around 15 minutes. Click on “View DB Instance Details” to visit the detail page for the instance you just created.

A screenshot of the AWS RDS Instances Launch DB instance page. The top navigation bar shows the AWS logo, Services dropdown, Resource Groups dropdown, a bell icon, Luigi Starace, and Frank. The breadcrumb navigation shows RDS > Instances > Launch DB instance. A green callout box contains the text "Your DB instance is being created." with a checkmark icon, and a note below it says "Note: Your instance may take a few minutes to launch." Below this, a section titled "Connecting to your DB instance" provides instructions: "Once Amazon RDS finishes provisioning your DB instance, you can use a SQL client application or utility to connect to the instance." A blue link "Learn about connecting to your DB instance" is provided. At the bottom, there are two buttons: "All DB instances" and a prominent orange "View DB instance details" button, which has a hand cursor icon pointing to it. The footer includes links for Feedback, English (US), Privacy Policy, and Terms of Use.

Your DB instance is being created.  
Note: Your instance may take a few minutes to launch.

Connecting to your DB instance

Once Amazon RDS finishes provisioning your DB instance, you can use a SQL client application or utility to connect to the instance.

Learn about connecting to your DB instance

All DB instances

View DB instance details

Feedback English (US) Privacy Policy Terms of Use

56/187

## STEP 7: CREATE A DATABASE INSTANCE

When done, the status in your instance detail page will change to “available” .

The screenshot shows the AWS RDS Instances detail page for a database named "serverful-webapp-db". The top navigation bar includes "Resource Groups", a user icon, "Luigi Starace", "Frankfurt", and "Support". Below the navigation, the breadcrumb path is "RDS > Instances > serverful-webapp-db". The main content area has a title "serverful-webapp-db" and an "Instance actions" button. A summary card displays the following information:

Engine MySQL 5.7.21	DB instance class db.t2.micro	Info	DB instance status <b>available</b>	Pending maintenance none
------------------------	----------------------------------	------	--	-----------------------------

An orange arrow points to the "available" status in the DB instance status column. Below the summary card is a "CloudWatch (54)" section with a search bar and a legend entry for "serverful-webapp-db". The footer contains copyright information: "© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved." and links to "Privacy Policy" and "Terms of Use".

# STEP 7: CREATE A DATABASE INSTANCE

Notice a few important elements in the details section. We're going to need these later.

Details		Modify	
Configurations	Security and network	Instance and IOPS	Maintenance details
ARN arn:aws:rds:eu-central-1:788880174327:db:serverful-webapp-db	Availability zone eu-central-1c	Instance Class db.t2.micro	Auto minor version upgrade <b>Yes</b>
Engine MySQL 5.7.21	VPC vpc-12e77979	Storage Type General Purpose (SSD)	Maintenance window mon:01:28-mon:01:58 UTC (GMT)
License Model General Public License	Subnet group default	Storage 20 GB	Backup window <b>20:55-21:25 UTC (GMT)</b>
Created Time Sun Apr 15 08:40:55 GMT+200 2018	Subnets subnet-e4184a8f subnet-97274ada subnet-f53fb788	<b>Availability and durability</b>	
DB Name serverful_webapp_db	Security groups rds-launch-wizard-1 (sg-03a2d775170d52c34) ( active )	DB instance status available	Pending Modifications <b>None</b>
Username serverfulwebapp	Publicly accessible Yes	Multi AZ <b>No</b>	Pending maintenance none
Option Group default:mysql-5.7	Endpoint serverful-webapp-db.civya0ewont.eu-central-1.rds.amazonaws.com	Automated backups <b>Enabled (7 Days)</b>	<b>Encryption details</b>
Parameter group default:mysql5.7 (in-sync)		Latest restore time April 15, 2018 at 8:45:00 AM UTC+2	Encryption enabled <b>No</b>

## STEP 7: CREATE A DATABASE INSTANCE

We'll need this instance to be accessible by our web application. To do so we're going to add a new rule to allow all instances in the same security group to access the database instance.

## STEP 7: CREATE A DATABASE INSTANCE

Click on the security group in the section *Security Group Rules*.

Security group rules (2)			
<input type="text"/> Filter security group rules			
Security group	Type	Rule	
<a href="#">rds-launch-wizard-1 (sg-03a2d775170d52c34)</a>	CIDR/IP - Inbound	79.51.216.139/32	
<a href="#">rds-launch-wizard-1 (sg-03a2d775170d52c34)</a>	CIDR/IP - Outbound	0.0.0.0/0	

## STEP 7: CREATE A DATABASE INSTANCE

Select the *Inbound* tab then click on the Edit button.

The screenshot shows the AWS Security Groups console. On the left, there's a navigation sidebar with links for EC2 Dashboard, Events, Tags, Reports, Limits, Instances (selected), Launch Templates, Spot Requests, Reserved Instances, Dedicated Hosts, Images (AMIs), and Elastic Block Store (Volumes). The main area has a title bar with 'Create Security Group' and 'Actions'. Below it is a search bar with the query 'rds-launch-wizard-1'. A table lists one security group: 'sg-03a2d775170d52c34' with the name 'rds-launch-wizard-1'. The 'Inbound' tab is selected, and an 'Edit' button is highlighted with a cursor. The table below shows a single rule: Type: Custom TCP, Protocol: TCP, Port Range: 3306, Source: 79.51.216.139/. At the bottom, there are links for Feedback, English (US), Privacy Policy, and Terms of Use.

Type	Protocol	Port Range	Source	Description
Custom TCP	TCP	3306	79.51.216.139/	

## STEP 7: CREATE A DATABASE INSTANCE

Add a new rule as shown in the picture. Be sure to select the same security group of the database instance. Then save and return to the RDS instance detail page.

The screenshot shows the AWS Management Console with the AWS logo at the top left. The top navigation bar includes 'Services', 'Resource Groups', a user icon for 'Luigi Starace', and 'Frank'. Below the navigation is a search bar with the placeholder 'search ... rds-launch-wizard-1' and a pagination area showing '1 to 1 of 1'. The main content area has a title 'Edit inbound rules'. It contains a table with four columns: 'Type', 'Protocol', 'Port Range', and 'Source'. There are two rows of data:

Type	Protocol	Port Range	Source
Custom TCP	TCP	3306	Custom 79.51.216.139/32
MySQL/Aurora	TCP	3306	Custom sg-03a2d775170d52c34

Below the table is a button labeled 'Add Rule' with a cursor hovering over it. A note below the table states: 'NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new data that rule to be dropped for a very brief period of time until the new rule can be created.' At the bottom of the interface are tabs for 'STORE' and 'Volumes', and a footer with links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'. The footer also includes the copyright notice '© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.'

## STEP 7: CREATE A DATABASE INSTANCE

The rule you just added should be displayed among the other two.

Security group rules (3)		
<input type="text"/> Filter security group rules		
Security group	Type	Rule
rds-launch-wizard-1 (sg-03a2d775170d52c34)	CIDR/IP - Inbound	79.51.216.139/32
rds-launch-wizard-1 (sg-03a2d775170d52c34)	Security Group - Inbound	sg-03a2d775170d52c34
rds-launch-wizard-1 (sg-03a2d775170d52c34)	CIDR/IP - Outbound	0.0.0.0/0

## STEP 7: CREATE A BEANSTALK APPLICATION

Go to the Beanstalk console and select *Create New Application*.



The screenshot shows the AWS Elastic Beanstalk console. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, and user names 'Luigi Starace' and 'Frank'. Below the navigation bar is a search bar with the text 'Elastic Beanstalk' and a 'Create New Application' button, which has a white cursor arrow pointing to it. The main content area features a large 'Welcome to AWS Elastic Beanstalk' heading. Below it, a paragraph explains that with Elastic Beanstalk, you can deploy, monitor, and scale an application quickly and easily. It encourages users to let the service handle the heavy lifting so they can focus on their business. To the left of the welcome text is a screenshot of the Elastic Beanstalk monitoring dashboard for an application named 'MovieBackend'. The dashboard displays various metrics: Average Latency (53.6 ms), Sum Requests (148K), CPU Utilization (65%), Max Network In (354KB), and Maximum DiskReadbytes (12KB). It also includes two line graphs: 'Average Latency in seconds' and 'Sum Requests in count', both showing data over time from 8/18/2018 to 8/19/2018.

Welcome to AWS Elastic Beanstalk

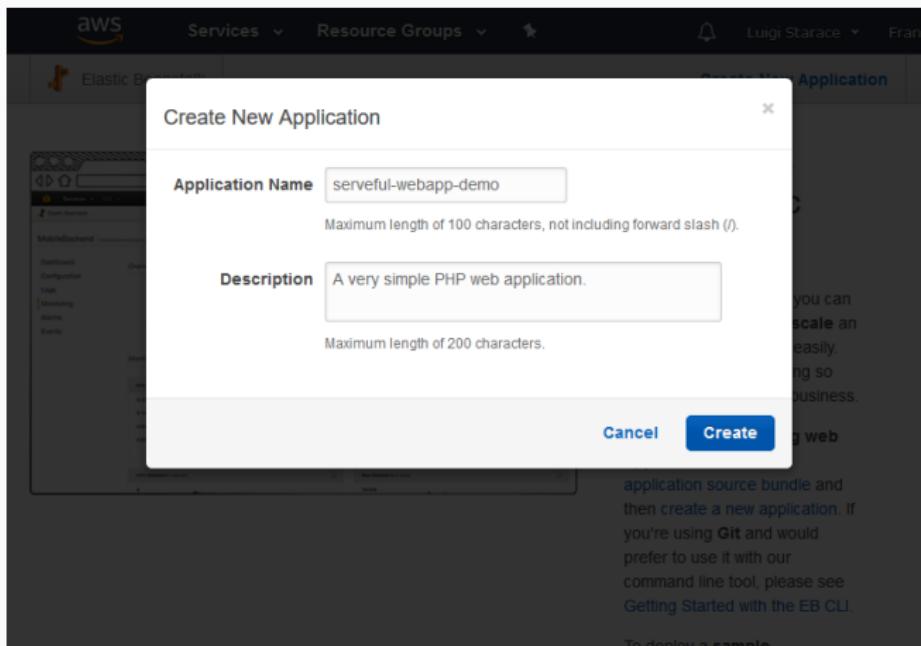
With Elastic Beanstalk, you can **deploy, monitor, and scale** an application quickly and easily. Let us do the heavy lifting so you can focus on your business.

To deploy your **existing web application**, create an application source bundle and then **create a new application**. If you're using **Git** and would prefer to use it with our command line tool, please see **Getting Started with the EB CLI**.

To deploy a **sample**

## STEP 7: CREATE A BEANSTALK APPLICATION

Fill the form with your application information and continue.



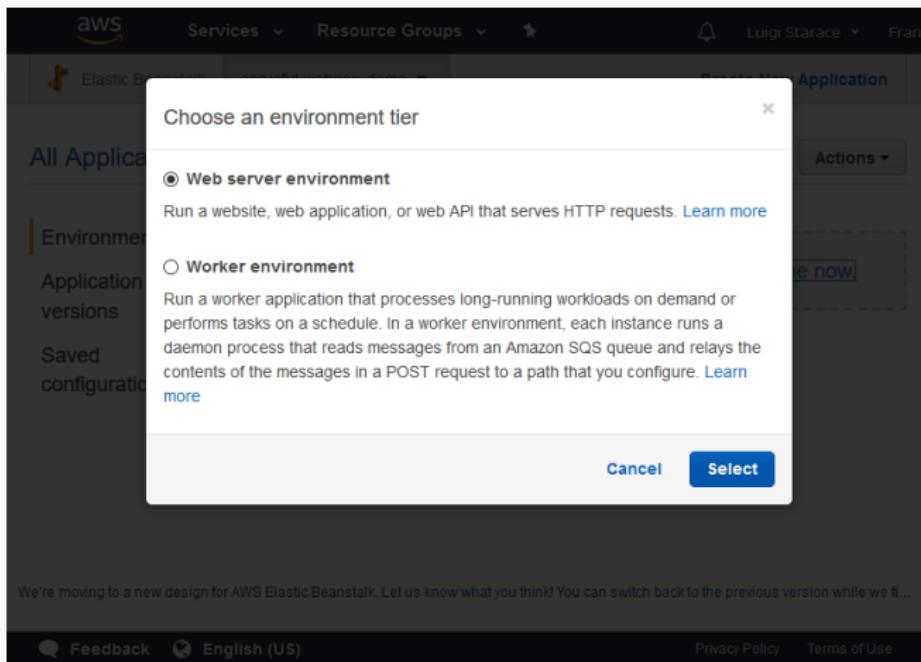
## STEP 7: CREATE A BEANSTALK APPLICATION

Then select *Create one now* to create a new environment for your application.

The screenshot shows the AWS Elastic Beanstalk console interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, and user profile 'Luigi Starace'. Below the navigation bar, there's a search bar with the text 'Elastic Beanstalk' and a dropdown menu showing 'serveful-webapp-demo'. To the right of the search bar is a blue button labeled 'Create New Application'. The main content area has a breadcrumb navigation 'All Applications > serveful-webapp-demo' and an 'Actions' dropdown. On the left, there are two sections: 'Environments' (which is selected and highlighted in orange) and 'Application versions'. The 'Environments' section contains a message: 'No environments currently exist for this application. [Create one now.](#)' with a cursor icon pointing to the link. Below this, there's another section for 'Saved configurations'. At the bottom of the page, there's a footer note: 'We're moving to a new design for AWS Elastic Beanstalk. Let us know what you think! You can [switch back to the previous version](#) while we f...' and a feedback link. The footer also includes links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'.

## STEP 7: CREATE A BEANSTALK APPLICATION

Select *Web Server Environment*, as we are going to deploy a web application.



## STEP 7: CREATE A BEANSTALK APPLICATION

Fill the form with information about your environment.



### Create a new environment

Launch an environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. [Learn more](#)

#### Environment information

Choose the name, subdomain, and description for your environment. These cannot be changed later.

**Application name** serveful-webapp-demo

**Environment name** ServefulWebappDemo-production

**Domain**

serverful-webapp

.eu-central-1.elasticbeanstalk.com

**Check availability**

serverful-webapp.eu-central-1.elasticbeanstalk.com **is available.**

**Description**

The production environment.

## STEP 7: CREATE A BEANSTALK APPLICATION

Select PHP as preconfigured platform and upload the source bundle you previously prepared.

Tier Web Server ([Choose tier](#))

Platform  Preconfigured platform  
Platforms published and maintained by AWS Elastic Beanstalk.  
 PHP

Custom platform [NEW](#)  
Platforms created and owned by you. [Learn more](#)  
-- Choose a custom platform --

Application code  Sample application  
Get started right away with sample code.  
 Existing version  
Application versions that you have uploaded for serveful-webapp-demo.  
-- Choose a version --

Upload your code  
Upload a source bundle from your computer or copy one from Amazon S3.  
 **Upload** serveful-webapp-demo-source 

# STEP 7: CREATE A BEANSTALK APPLICATION

Select *Configure More Options* and continue.

-- Choose a custom platform --

**Application code**

- Sample application  
Get started right away with sample code.
- Existing version  
Application versions that you have uploaded for serveful-webapp-demo.
- Upload your code  
Upload a source bundle from your computer or copy one from Amazon S3.

-- Choose a version --

**Upload** serveful-webapp-demo-source ↗

---

[Cancel](#) [Configure more options](#) **Create environment**

We're moving to a new design for AWS Elastic Beanstalk. Let us know what you think! You can [switch back to the previous version](#) while we fi...

## STEP 7: CREATE A BEANSTALK APPLICATION

In the configuration page, select *modify* con the *Instances* card.

 Configure ServefulWebappDemo-production

Start from a preset that matches your use case or choose *Custom configuration* to unset recommended values and use the service's default values.

**Configuration presets**  Low cost (*Free Tier eligible*)  
 High availability  
 Custom configuration

Platform 64bit Amazon Linux 2017.09 v2.6.6 running PHP 7.1 [Change platform configuration](#)

<b>Software</b> Rotate logs: disabled (default) Log streaming: disabled (default) Environment properties: 0	<b>Instances</b> EC2 instance type: t2.micro EC2 image ID: ami-babee351 Root volume type: container default Root volume size (GB): container default Root volume IOPS: container default Security groups: <i>none</i>	<b>Capacity</b> Environment type: single instance
<a href="#">Modify</a>	<a href="#">Modify</a>	<a href="#">Modify</a>

## STEP 7: CREATE A BEANSTALK APPLICATION

In the instances configuration page, add the t2 instance to the same security group as the DB instance. Then save and continue.

Input/output operations per second for a provisioned IOPS (SSD) volume.

EC2 security groups

	Group name	Group ID	Name
<input type="checkbox"/>	default	sg-c6660dab	
<input type="checkbox"/>	rds-launch-wizard	sg-09c29f2dc 61f55e1a	
<input checked="" type="checkbox"/> 	rds-launch-wizard-1	sg-03a2d775 170d52c34	

[Cancel](#) [Save](#)

## STEP 7: CREATE A BEANSTALK APPLICATION

In the configuration page, select *modify* con the *Software* card.

 Configure ServefulWebappDemo-production

Start from a preset that matches your use case or choose *Custom configuration* to unset recommended values and use the service's default values.

**Configuration presets**

- Low cost (*Free Tier eligible*)
- High availability
- Custom configuration

Platform 64bit Amazon Linux 2017.09 v2.6.6 running PHP 7.1 [Change platform configuration](#)

<b>Software</b> Rotate logs: disabled (default) Log streaming: disabled (default) Environment properties: 0	<b>Instances</b> EC2 instance type: t2.micro EC2 image ID: ami-babee351 Root volume type: container default Root volume size (GB): container default Root volume IOPS: container default Security groups: sg-03a2d775170d52c34	<b>Capacity</b> Environment type: single instance
<a href="#">Modify</a>	<a href="#">Modify</a>	<a href="#">Modify</a>

## STEP 7: CREATE A BEANSTALK APPLICATION

Enter “/public” as the document root and scroll down.

 Modify software

**Container Options**

The following settings control container behavior and let you pass key-value pairs in as OS environment variables. [Learn more](#)

**Document root**  

The child directory of your project that acts as the public facing web root. If your root document is stored in your project directory, leave this set to /. If your root document is in a child directory (e.g., /public), set this value to match the child directory. Values should begin with a / character, and may NOT begin with a .. (period).

**Memory limit**

The amount of memory allocated to the PHP environment. This value is written to the php.ini file.

**Zlib output compression**

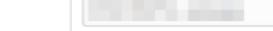
Whether PHP should use compression for output. This value is written to the php.ini file.

## STEP 7: CREATE A BEANSTALK APPLICATION

Enter the required environment parameters shown in the picture. Be careful, deployment might fail if you mess up! 

### Environment properties

The following properties are passed in the application as environment properties. [Learn more](#)

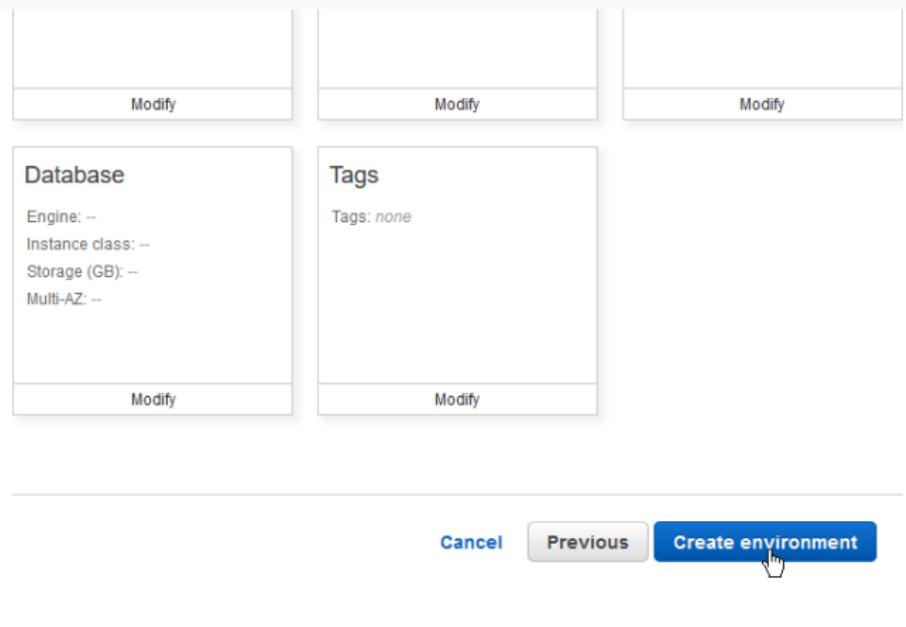
Name	Value
APP_ENV	prod 
RDS_HOST	-central-1.rds.amazonaws.com 
RDS_NAME	serverful_webapp_db 
RDS_USER	serverfulwebapp 
RDS_PASSWORD	 
RDS_PORT	3306 

[Cancel](#)

[Save](#)

## STEP 7: CREATE A BEANSTALK APPLICATION

Click on *Create Environment* and continue.



## STEP 7: CREATE A BEANSTALK APPLICATION

Wait for the environment to be created. This takes about 10 minutes.

The screenshot shows the AWS Elastic Beanstalk console. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a user icon for Luigi Starace, and another for Frank. Below the navigation bar, there's a search bar with the text 'Elastic Beanstalk' and a dropdown menu showing 'serveful-webapp-demo'. To the right of the search bar is a blue button labeled 'Create New Application'. The main content area has a breadcrumb trail: 'All Applications > serveful-webapp-demo > ServefulWebappDemo-prod'. Below the breadcrumb is a note: '( Environment ID: e-isdwk7la4q, URL: serverful-webapp.eu-central-1.elasticbeanstalk.com )'. On the left, a modal window titled 'Creating ServefulWebappDemo-prod' contains the message 'This will take a few minutes.' and a log entry: '9:55am Using elasticbeanstalk-eu-central-1-788880174327 as Amazon S3 storage bucket for environment data.'. On the right, there are two columns of links: 'Learn More' (Get started using Elastic Beanstalk, Modify the code, Create and connect to a database, Add a custom domain) and 'Featured' (Create your own custom platform). At the bottom, there are links for 'Command Line Interface (v3)' and 'Installing the AWS EB CLI'.

All Applications > serveful-webapp-demo > ServefulWebappDemo-prod ( Environment ID: e-isdwk7la4q, URL: serverful-webapp.eu-central-1.elasticbeanstalk.com )

**i Creating ServefulWebappDemo-prod**

This will take a few minutes.

9:55am Using elasticbeanstalk-eu-central-1-788880174327 as Amazon S3 storage bucket for environment data.

9:55am createEnvironment is starting.

**Learn More**

- Get started using Elastic Beanstalk
- Modify the code
- Create and connect to a database
- Add a custom domain

**Featured**

- Create your own custom platform

**Command Line Interface (v3)**

Installing the AWS EB CLI

## STEP 8: LOAD BALANCING

Right now we have our application running on a single (virtual) web server. That's not scaling at all. Let's take advantage of the cloud and make the web application load balanced.

## STEP 8: LOAD BALANCING

Select the environment's configuration view, then select the Capacity card.

Configuration overview

<b>Software</b> Rotate logs: disabled (default) Log streaming: disabled (default) Environment properties: 6  <a href="#">Modify</a>	<b>Instances</b> EC2 instance type: t2.micro EC2 image ID: ami-babee351 Monitoring interval: 5 minute Root volume type: container default Root volume size (GB): container default Root volume IOPS: container default Security groups: sg-03a2d775170d52c34, sg-03c173ae822a52c04  <a href="#">Modify</a>	<b>Capacity</b> Environment type: single instance  <a href="#">Modify</a>
<b>Load balancer</b> <i>This configuration does not contain a load balancer.</i>  <a href="#">Modify</a>	<b>Rolling updates and deployments</b> Deployment policy: All at once Rolling updates: disabled Health check: enabled  <a href="#">Modify</a>	<b>Security</b> Service role: aws-elasticbeanstalk-service-role Virtual machine key pair: -- Virtual machine instance profile: aws-elasticbeanstalk-ec2-role  <a href="#">Modify</a>

## STEP 8: LOAD BALANCING

Select “Load balanced” as the environment type and customize the Auto Scaling Group.

### Modify capacity

#### Auto Scaling Group

Configure the compute capacity of your environment and Auto Scaling settings to optimize the number of instances used.

Environment type

Instances

Availability Zones

Number of Availability Zones (AZs) to use.

Placement

Specify Availability Zones (AZs) to use.

Scaling cooldown

## STEP 8: LOAD BALANCING

Select some triggers (you can even setup time based ones), then save your changes.

Scaling triggers

**Metric** CPUUtilization ▾  
Change the metric that is monitored to determine if the environment's capacity is too low or too high.

**Statistic** Average ▾  
Choose how the metric is interpreted.

**Unit** Percent ▾

**Period** 5 ▾ Min  
The period between metric evaluations.

**Breach duration** 5 ▾ Min  
The amount of time a metric can exceed a threshold before triggering a scaling operation.

**Upper threshold** 100 ▾ Percent

**Scale up increment** 1 EC2 instances

**Lower threshold** 90 ▾ Percent

**Scale down increment** -1 EC2 instances

## STEP 9: ENJOY YOUR WEB APP

When it's done you should see something like this. Click on the URL to visit the load-balanced web application you just deployed on Beanstalk!

The screenshot shows the AWS Elastic Beanstalk console interface. At the top, there's a navigation bar with 'Services', 'Resource Groups', and other account details. Below it, a breadcrumb trail shows 'All Applications > serveful-webapp-demo > ServefulWebappDemo-prod'. A 'Create New Application' button is also visible.

The main area is titled 'Overview' and contains several key pieces of information:

- Health:** Shows a green circle with a checkmark and the status 'Ok'.
- Running Version:** 'serveful-webapp-demo-source'.
- Configuration:** Shows 'PHP 7.1 running on 64bit Amazon Linux/2.6.6'.
- Actions:** Buttons for 'Upload and Deploy' and 'Change'.

On the left, a sidebar lists navigation options: Dashboard, Configuration, Logs, Health, Monitoring, Alarms, Managed Updates, Recent Events, Events, and Tags. The 'Recent Events' section displays a table of log entries:

Time	Type	Details
2018-04-15 10:08:37 UTC+0200	INFO	createConfigurationTemplate completed successfully.
2018-04-15 10:08:36 UTC+0200	INFO	createConfigurationTemplate is starting.
2018-04-15 10:00:21 UTC+0200	INFO	Successfully launched environment: ServefulWebappDemo-prod
2018-04-15 09:59:55 UTC+0200	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 29 seconds ago and took 3 minutes.
2018-04-15 09:57:55 UTC+0200	INFO	Added instance [i-0a2daceaf5fe9f58e6] to your environment.

## STEP 9: ENJOY YOUR WEB APP

Sweet!

The screenshot shows a web page with a dark header bar. In the top-left corner of the header, the text "GettingToKnowAWS" is displayed, with "ToKnowAWS" in orange. To the right of this are three white links: "Home", "Comments", and "Pricing". The main content area has a light blue background with a white cloud pattern. At the top of this area, there is large, bold, black text that reads "Eew! What do you mean you don't know AWS?". Below this, in a smaller black font, is the text "Don't worry, we've got you covered!". At the bottom left of the main content area is a dark rectangular button with the white text "Learn more". At the very bottom of the slide, there is a white footer bar containing the text "Get to know Amazon Web Services with our talk" in black.

**LET'S GET BACK TO COMPUTING SERVICES**

---

# AMAZON LAMBDA

- You provide the code and say when to run it.



## AMAZON LAMBDA

- You provide the code and say when to run it.
- Execution is triggered by events



## AMAZON LAMBDA

- You provide the code and say when to run it.
- Execution is triggered by events
  - S3, Cognito, DynamoDB



## AMAZON LAMBDA

- You provide the code and say when to run it.
- Execution is triggered by events
  - S3, Cognito, DynamoDB
  - CodeCommit, Scheduled Event



## AMAZON LAMBDA

- You provide the code and say when to run it.
- Execution is triggered by events
  - S3, Cognito, DynamoDB
  - CodeCommit, Scheduled Event
- Support for Java, Node.js, C# e Python  
(more to come).



## AMAZON LAMBDA

- You provide the code and say when to run it.
- Execution is triggered by events
  - S3, Cognito, DynamoDB
  - CodeCommit, Scheduled Event
- Support for Java, Node.js, C# e Python (more to come).
- Pay only for **actual** execution time.



## AMAZON LAMBDA

- You provide the code and say when to run it.
- Execution is triggered by events
  - S3, Cognito, DynamoDB
  - CodeCommit, Scheduled Event
- Support for Java, Node.js, C# e Python (more to come).
- Pay only for **actual** execution time.
- Run your code without thinking about infrastructure



## AMAZON LAMBDA

- You provide the code and say when to run it.
- Execution is triggered by events
  - S3, Cognito, DynamoDB
  - CodeCommit, Scheduled Event
- Support for Java, Node.js, C# e Python (more to come).
- Pay only for **actual** execution time.
- Run your code without thinking about infrastructure
  - No need to worry about provisioning, load balancing, scaling...



## AMAZON LAMBDA: LIMITS

AWS Lambda imposes some limits

- Max 300 seconds execution time.



## AMAZON LAMBDA: LIMITS

AWS Lambda imposes some limits

- Max 300 seconds execution time.
- Max 3008 MB memory allocation.



## AMAZON LAMBDA: LIMITS

AWS Lambda imposes some limits

- Max 300 seconds execution time.
- Max 3008 MB memory allocation.
- Deployment package must be smaller than 50 MB (negotiable).



## AMAZON LAMBDA: LIMITS

AWS Lambda imposes some limits

- Max 300 seconds execution time.
- Max 3008 MB memory allocation.
- Deployment package must be smaller than 50 MB (negotiable).
- No more than 10000 concurrent invocation of a Lambda function in a given region (negotiable).



## AMAZON LAMBDA: LIMITS

AWS Lambda imposes some limits

- Max 300 seconds execution time.
- Max 3008 MB memory allocation.
- Deployment package must be smaller than 50 MB (negotiable).
- No more than 10000 concurrent invocation of a Lambda function in a given region (negotiable).
- For a complete list: [Lambda docs](#)



## FaaS (Functions as a Service)

- Functions are the unit of deployment

## FaaS (Functions as a Service)

- Functions are the unit of deployment
- Executed in ephemeral, stateless containers

## FaaS (Functions as a Service)

- Functions are the unit of deployment
- Executed in ephemeral, stateless containers
- Event driven

## FaaS (Functions as a Service)

- Functions are the unit of deployment
- Executed in ephemeral, stateless containers
- Event driven
- No provisioning, scales automatically

## FaaS (Functions as a Service)

- Functions are the unit of deployment
- Executed in ephemeral, stateless containers
- Event driven
- No provisioning, scales automatically
- Azure: Functions [▶ web](#)

## FaaS (Functions as a Service)

- Functions are the unit of deployment
- Executed in ephemeral, stateless containers
- Event driven
- No provisioning, scales automatically
- Azure: Functions [▶ web](#)
- Google Cloud: Functions [▶ web](#)

## FaaS (Functions as a Service)

- Functions are the unit of deployment
- Executed in ephemeral, stateless containers
- Event driven
- No provisioning, scales automatically
- Azure: Functions [▶ web](#)
- Google Cloud: Functions [▶ web](#)
- IBM: Cloud Functions [▶ web](#)

## FaaS (Functions as a Service)

- Functions are the unit of deployment
- Executed in ephemeral, stateless containers
- Event driven
- No provisioning, scales automatically
- Azure: Functions [▶ web](#)
- Google Cloud: Functions [▶ web](#)
- IBM: Cloud Functions [▶ web](#)
  - Based on Apache OpenWhisk [▶ web](#)

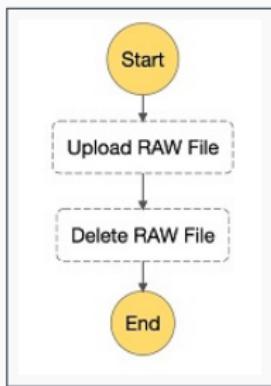
- Orchestrating Lambda functions



- Orchestrating Lambda functions
- Define a state machine



# AMAZON STEP FUNCTIONS: SAMPLE I



**Figure 3:** Sequential steps, from [AWS]

## AMAZON STEP FUNCTIONS: SAMPLE II

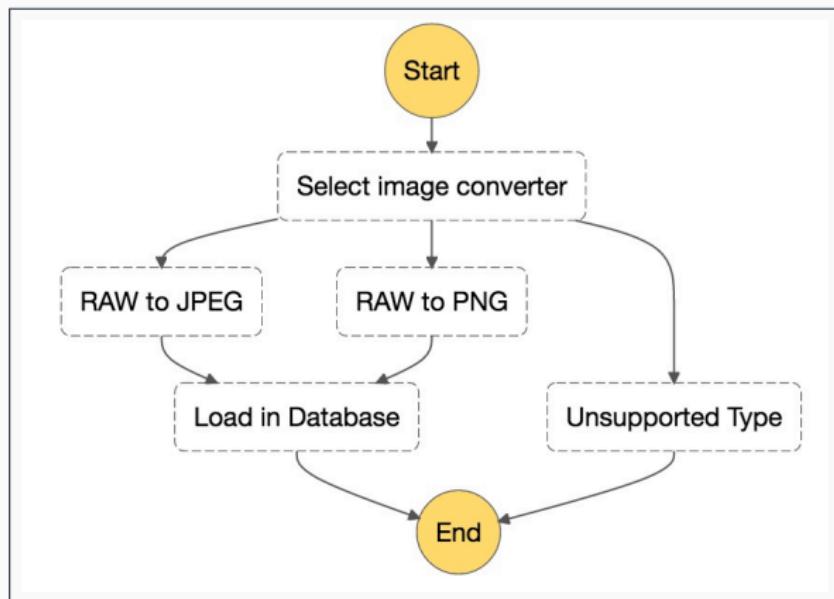
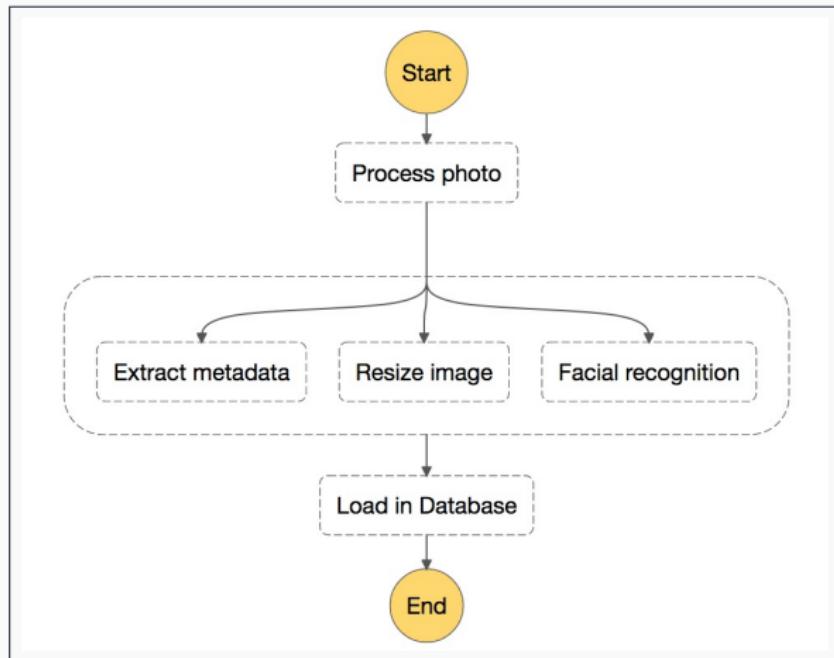


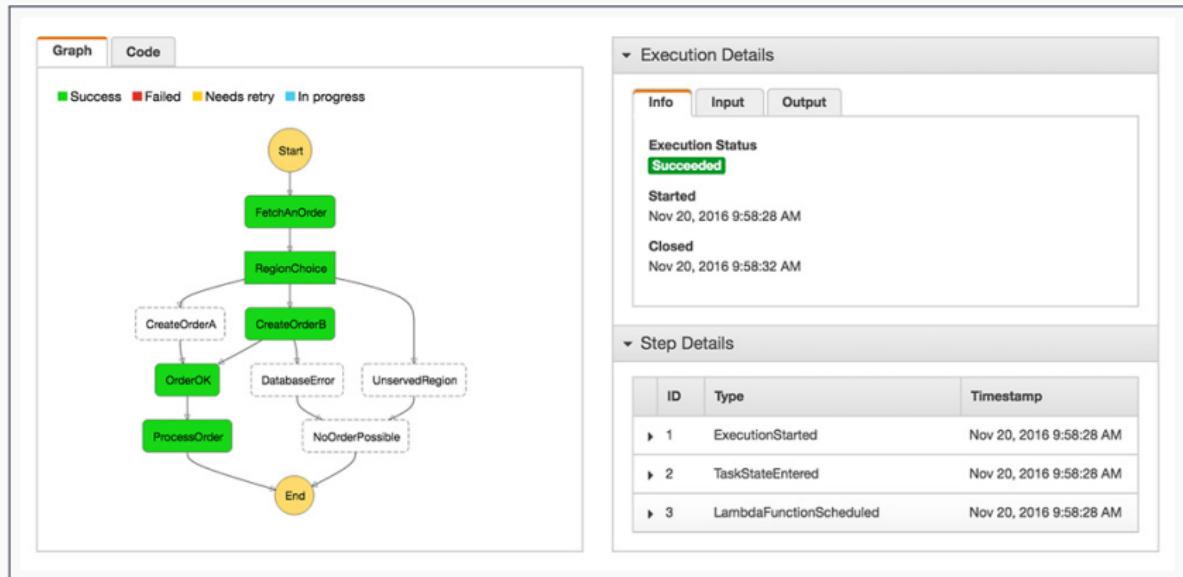
Figure 4: Branching, from [AWS]

# AMAZON STEP FUNCTIONS: SAMPLE III



**Figure 5:** Parallel execution, from [AWS]

# AMAZON STEP FUNCTIONS: SAMPLE IV



- Create, publish, and secure APIs at any scale



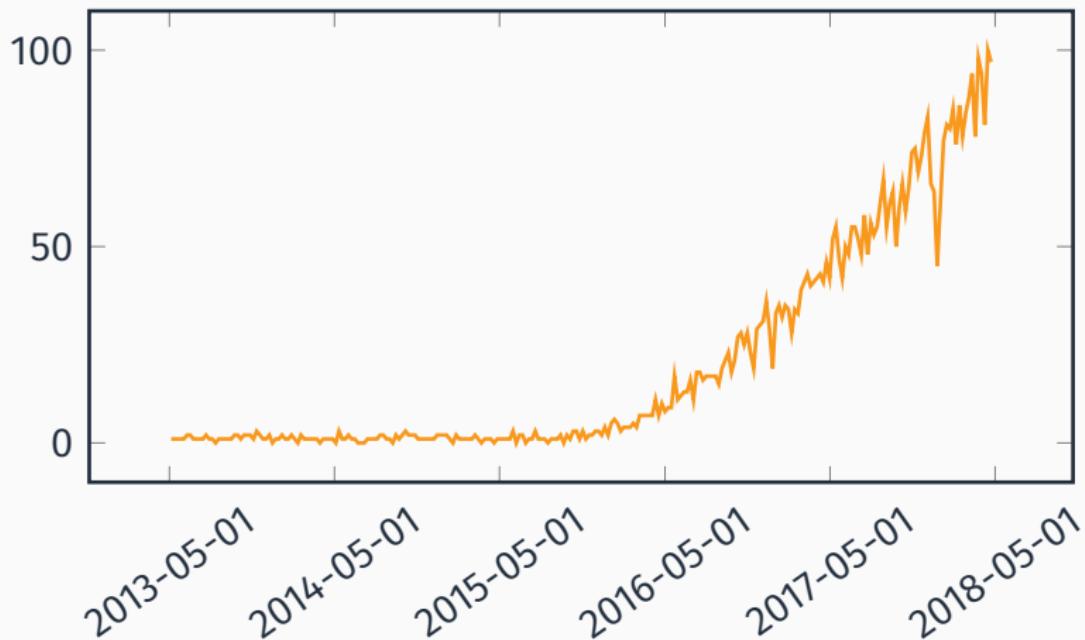
- Create, publish, and secure APIs at any scale
- Authorizers (Cognito)



## Serverless?

What's all the FaaS about?

# SERVERLESS TREND



**Figure 7:** Last five years trend on Google for “serverless”

# SERVERLESS ARCHITECTURES

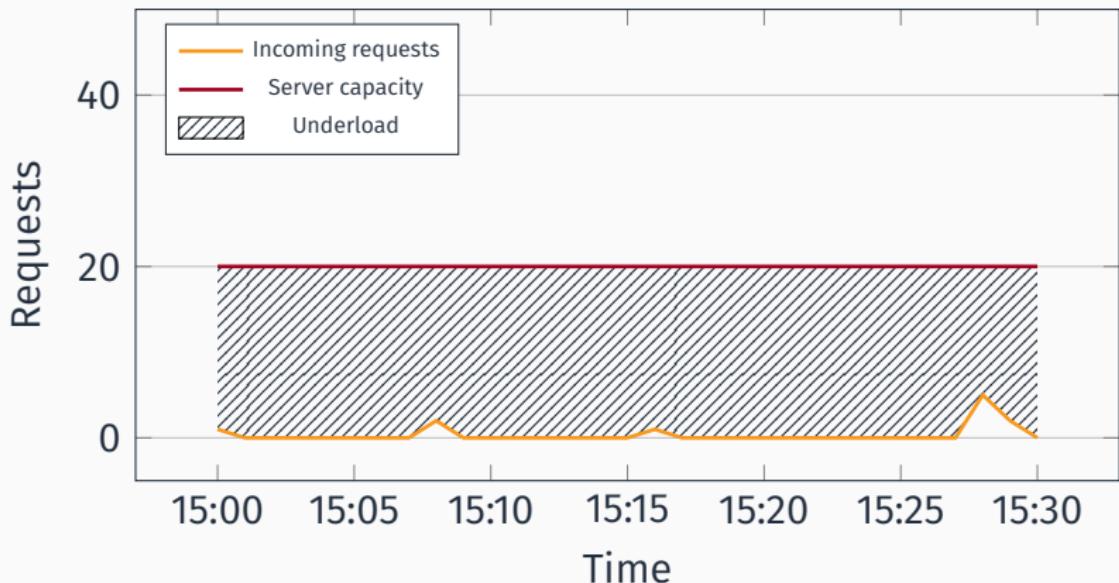
- No, they're not *actually* serverless...

- No, they're not *actually* serverless...
- Rely on FaaS and third-party services so that traditional always-on servers are no longer needed

- No, they're not *actually* serverless...
- Rely on FaaS and third-party services so that traditional always-on servers are no longer needed
- No worries about provisioning and scaling

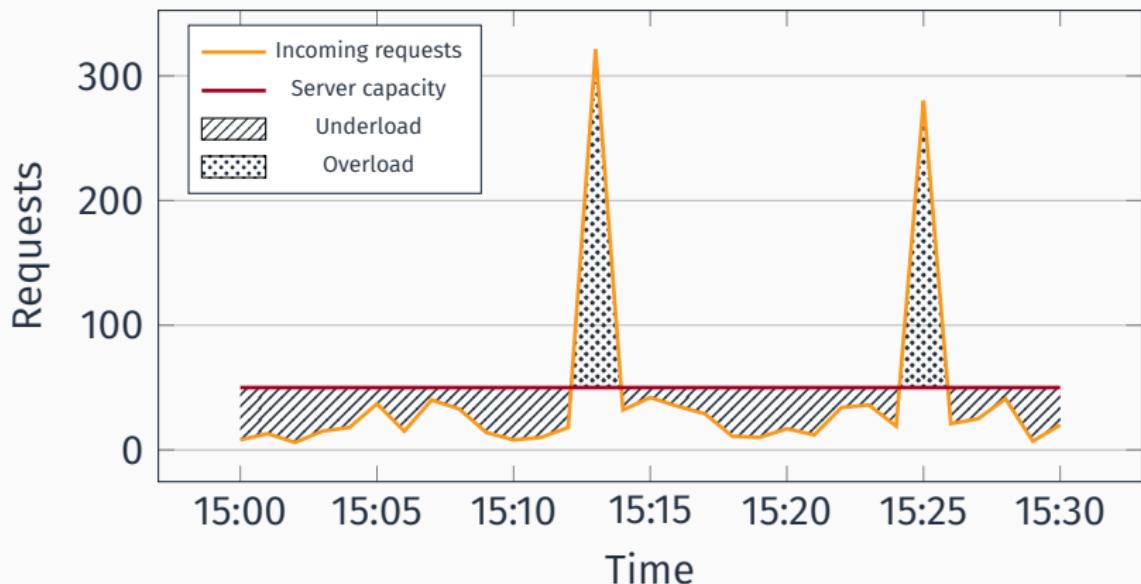
- No, they're not *actually* serverless...
- Rely on FaaS and third-party services so that traditional always-on servers are no longer needed
- No worries about provisioning and scaling
- “Smarter” clients

## SERVERLESS USE CASES: SPORADIC REQUESTS



**Figure 8:** Sporadic requests example

## SERVERLESS USE CASES: INCONSISTENT REQUESTS



**Figure 9:** Inconsistent requests example

# SERVERLESS ARCHITECTURES: TRADE-OFFS

## Pros

- Reduce costs  servers.LOL

# SERVERLESS ARCHITECTURES: TRADE-OFFS

## Pros

- Reduce costs 
- No worries about provisioning, scaling

# SERVERLESS ARCHITECTURES: TRADE-OFFS

## Pros

- Reduce costs  servers.LOL
- No worries about provisioning, scaling
- Less time to market

# SERVERLESS ARCHITECTURES: TRADE-OFFS

## Pros

- Reduce costs 
- No worries about provisioning, scaling
- Less time to market

## Cons

# SERVERLESS ARCHITECTURES: TRADE-OFFS

## Pros

- Reduce costs 
- No worries about provisioning, scaling
- Less time to market

## Cons

- Limits

# SERVERLESS ARCHITECTURES: TRADE-OFFS

## Pros

- Reduce costs  servers.LOL
- No worries about provisioning, scaling
- Less time to market

## Cons

- Limits
- Vendor lock-in

# SERVERLESS ARCHITECTURES: TRADE-OFFS

## Pros

- Reduce costs 
- No worries about provisioning, scaling
- Less time to market

## Cons

- Limits
- Vendor lock-in
- Testing

# SERVERLESS ARCHITECTURES: TOOLS

## ■ Serverless Framework



web



# SERVERLESS ARCHITECTURES: TOOLS

- Serverless Framework  web
  - “toolkit to deploy and operate serverless architecture” .



# SERVERLESS ARCHITECTURES: TOOLS

## ■ Serverless Framework

- “toolkit to deploy and operate serverless architecture” .
- Works with AWS, Google, Microsoft, IBM.



# SERVERLESS ARCHITECTURES: TOOLS

## ■ Serverless Framework web

- “toolkit to deploy and operate serverless architecture” .
- Works with AWS, Google, Microsoft, IBM.

## ■ APEX web



# SERVERLESS ARCHITECTURES: TOOLS

- Serverless Framework ↗ web
  - “toolkit to deploy and operate serverless architecture” .
  - Works with AWS, Google, Microsoft, IBM.
- APEX ↗ web
- AWS SAM ↗ web



# SERVERLESS ARCHITECTURES: TOOLS

- Serverless Framework  

- “toolkit to deploy and operate serverless architecture” .
- Works with AWS, Google, Microsoft, IBM.

- APEX  

- AWS SAM  

- Serverless Application Model



# SERVERLESS ARCHITECTURES: TOOLS

- Serverless Framework ↗ web
  - “toolkit to deploy and operate serverless architecture” .
  - Works with AWS, Google, Microsoft, IBM.
- APEX ↗ web
- AWS SAM ↗ web
  - Serverless Application Model
  - “Define serverless applications with a simple and clean syntax”



# SERVERLESS ARCHITECTURES: TOOLS

## ■ Serverless Framework

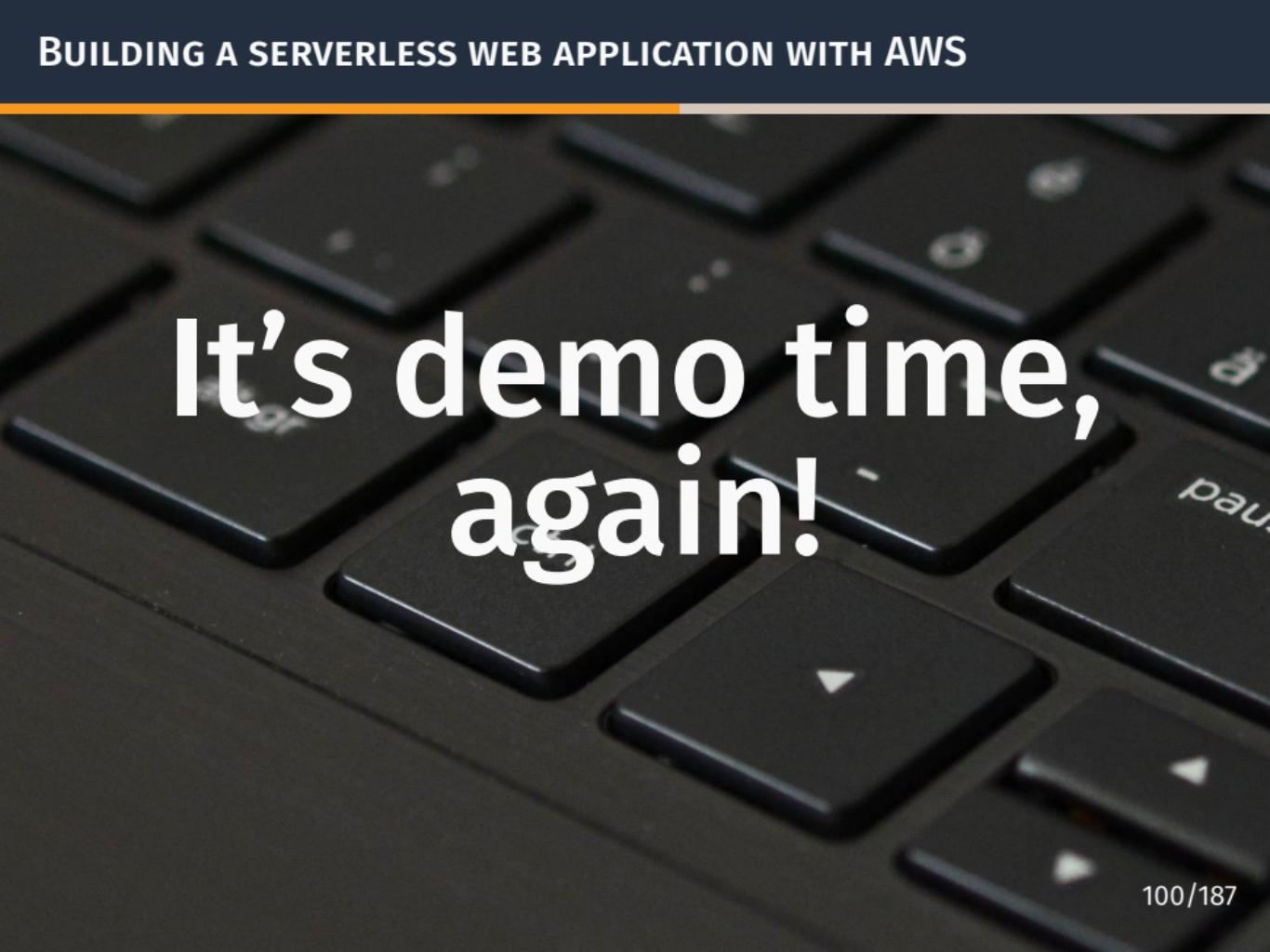
- “toolkit to deploy and operate serverless architecture” .
- Works with AWS, Google, Microsoft, IBM.

## ■ APEX

## ■ AWS SAM

- Serverless Application Model
- “Define serverless applications with a simple and clean syntax”
- SAM Local: CLI tool for local development and testing





It's demo time,  
again!

## WHAT WE'RE GOING TO DO IN THIS DEMO

- Remember the web application for this very talk we deployed earlier?

## WHAT WE'RE GOING TO DO IN THIS DEMO

- Remember the web application for this very talk we deployed earlier?
- Now we'll make it **serverless**, and add more features:

## WHAT WE'RE GOING TO DO IN THIS DEMO

- Remember the web application for this very talk we deployed earlier?
- Now we'll make it **serverless**, and add more features:
  - Sign-up and Authentication (Amazon Cognito)

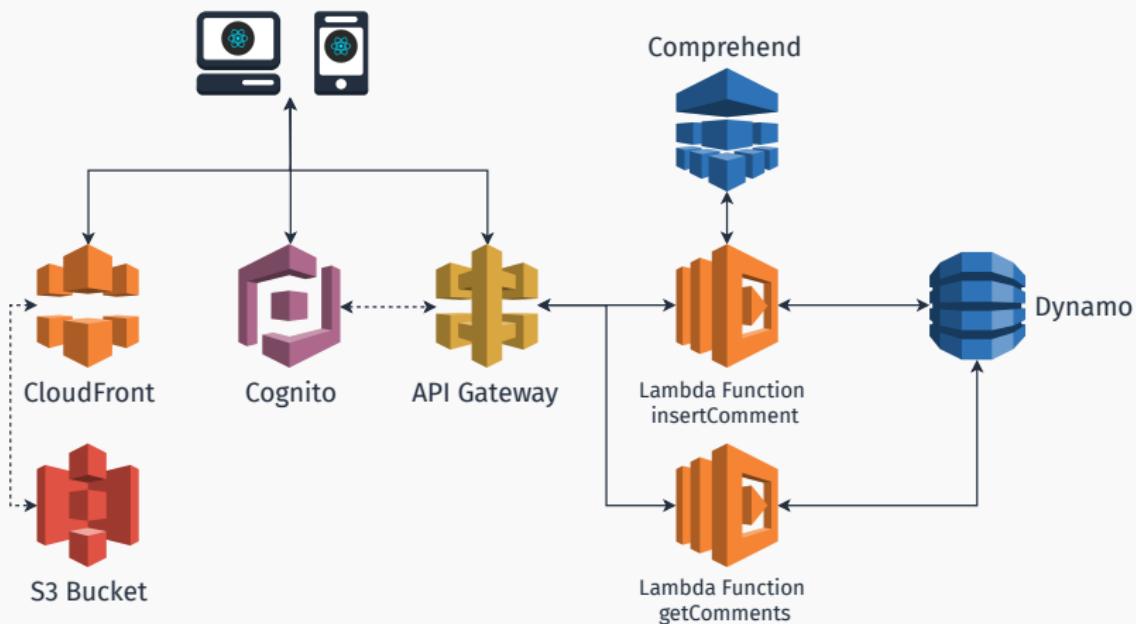
## WHAT WE'RE GOING TO DO IN THIS DEMO

- Remember the web application for this very talk we deployed earlier?
- Now we'll make it **serverless**, and add more features:
  - Sign-up and Authentication (Amazon Cognito)
  - Language detection and **sentiment analysis** on comments (Amazon Comprehend)

## WHAT WE'RE GOING TO DO IN THIS DEMO

- Remember the web application for this very talk we deployed earlier?
- Now we'll make it **serverless**, and add more features:
  - Sign-up and Authentication (Amazon Cognito)
  - Language detection and **sentiment analysis** on comments (Amazon Comprehend)
  - Deploy it on a global CDN to minimize latency (Amazon CloudFront)

# ARCHITECTURE



▶ Skip tutorial

## WHAT YOU'LL NEED

- An AWS account (a free one will suffice)

## WHAT YOU'LL NEED

- An AWS account (a free one will suffice)
- git version control (recommended)

## WHAT YOU'LL NEED

- An AWS account (a free one will suffice)
- git version control (recommended)
- Node.js

## WHAT YOU'LL NEED

- An AWS account (a free one will suffice)
- git version control (recommended)
- Node.js
- Python (recommended)

## STEP 1: GET THE APP

Clone the git repository



Github

```
D:> git clone https://github.com/luistar/serverless-  
webapp.git serverless-webapp
```

## STEP 2: INSTALL DEPENDENCIES

```
D:> cd serverless-webapp  
D:\serverless-webapp> npm install
```

## STEP 3: INSTALL AWS-MOBILE CLI TOOL

```
D:\serverless-webapp> npm -g install awsmobile-cli
```

## STEP 4: CREATE A NEW USER ON IAM

In order to use awsmobile-cli you're gonna need an access key id and a secret access key. If you don't already have one, go the IAM console and create a new user.

## STEP 4: CREATE A NEW USER ON IAM

The screenshot shows the AWS IAM service interface. The top navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon for notifications, and a user profile for 'Luigi Starace'. On the left, a sidebar menu lists 'Dashboard', 'Groups', 'Users' (which is selected and highlighted with an orange border), 'Roles', 'Policies', 'Identity providers', 'Account settings', 'Credential report', and 'Encryption keys'. The main content area has a search bar labeled 'Search IAM' and two buttons: 'Add user' (highlighted with a mouse cursor) and 'Delete user'. Below these buttons is a search bar with the placeholder 'Find users by username or access key'. A table lists existing users, with one row visible: 'serverless...', 'None', '2 days', and 'Not used'.

User name	Groups	Access key age	Last used
serverless...	None	2 days	Not used

## STEP 4: CREATE A NEW USER ON IAM

The screenshot shows the AWS IAM 'Add user' wizard, Step 1: Set user details. The top navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, and a user profile for 'Luigi Starace'. Below the header, the title 'Add user' is displayed, followed by a progress indicator showing step 1 of 2.

**Set user details**

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name\*

[+ Add another user](#)

**Select AWS access type**

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

**Access type\***

**Programmatic access**  
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

**AWS Management Console access**  
Enables a **password** that allows users to sign-in to the AWS Management Console.

## STEP 4: CREATE A NEW USER ON IAM

The screenshot shows the AWS IAM 'Add user' interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, and a user profile 'Luigi Starace'. Below the navigation is a title 'Add user' and a progress indicator '1 / 2'. The main section is titled 'Set permissions for serverless-webapp-administrator'. It contains three buttons: 'Add user to group', 'Copy permissions from existing user', and 'Attach existing policies directly' (which is highlighted with a blue border and a cursor icon). Below these buttons is a note: 'Attach one or more existing policies directly to the users or create a new policy. [Learn more](#)'. There are 'Create policy' and 'Refresh' buttons. A table follows, filtered by 'Policy type'. The columns are 'Policy name', 'Type', 'Attachments', and 'Description'. The first row has a checked checkbox and is selected, showing 'AdministratorAccess' as a 'Job function' with 1 attachment, described as 'Provides full access to AWS services'. Other rows show 'AlexaForBusinessDeviceManager' (AWS managed), 'AlexaForBusinessFullAccess' (AWS managed), and 'AlexaForBusinessGatewayExecutionAccess' (AWS managed).

	Policy name	Type	Attachments	Description
<input checked="" type="checkbox"/>	AdministratorAccess	Job function	1	Provides full access to AWS services
<input type="checkbox"/>	AlexaForBusinessDeviceManager	AWS managed	0	Provide device setup access to Alexa
<input type="checkbox"/>	AlexaForBusinessFullAccess	AWS managed	0	Grants full access to AlexaForBusiness
<input type="checkbox"/>	AlexaForBusinessGatewayExecutionAccess	AWS managed	0	Provide gateway execution access to Alexa

# STEP 4: CREATE A NEW USER ON IAM

The screenshot shows the AWS IAM 'Add user' interface in the 'Review' step. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, 'Luigi Starace' profile, 'Global' dropdown, and 'Support'. Below the navigation, the title 'Add user' is followed by a progress bar with four steps: 1, 2, 3 (which is highlighted in blue), and 4.

**Review**

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

**User details**

User name	serverless-webapp-administrator
AWS access type	Programmatic access - with an access key

**Permissions summary**

The following policies will be attached to the user shown above.

Type	Name
Managed policy	AdministratorAccess

At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Create user' (which has a mouse cursor hovering over it).

## STEP 4: CREATE A NEW USER ON IAM

Be sure to write up your keys and to keep them safe!

[Read more about security](#)

The screenshot shows the AWS Management Console interface for creating a new user. At the top, there's a navigation bar with the AWS logo, 'Services', 'Resource Groups', a bell icon, 'Luigi Starace', 'Global', and 'Support'. Below the navigation, a breadcrumb trail shows '1 Add user'. The main content area has a green header 'Success' with a checkmark icon. The message reads: 'You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.' Below this message, it says 'Users with AWS Management Console access can sign-in at: https://[REDACTED] signin.aws.amazon.com/console'. There's a 'Download .csv' button with a CSV icon. A table lists the newly created user 'serverless-webapp-administrator' with columns for 'User', 'Access key ID', and 'Secret access key'. The 'Access key ID' and 'Secret access key' are redacted. A 'Hide' link is visible next to the redacted secret key. At the bottom right of the modal is a 'Close' button.

User	Access key ID	Secret access key
serverless-webapp-administrator	[REDACTED]	[REDACTED] [REDACTED] Hide

## STEP 5: CONFIGURE AWSMOBILE-CLI

Configure AWS Mobile CLI.

```
D:\serverless-webapp> awsmobile configure  
  
configure aws  
? accessKeyId: <YOUR_ACCESS_KEY_ID>  
? secretAccessKey: <YOUR_SECRET_ACCESS_KEY>  
? region: eu-central-1
```

## STEP 6: INITIALIZE A NEW AWS MOBILE PROJECT

```
D:\serverless-webapp> awsmobile init
```

Please tell us about your project:

- ? Where is your project's source directory: src
- ? Where is your project's distribution directory that stores build artifacts: build
- ? What is your project's build command: npm.cmd run-script build
- ? What is your project's start command for local test run: npm.cmd run-script start
- ? What awsmobile project name would you like to use: serverless-webapp

## STEP 6: INITIALIZE A NEW AWS MOBILE PROJECT

Visit the AWS Mobile Console. Your newly created project should be waiting for you there.

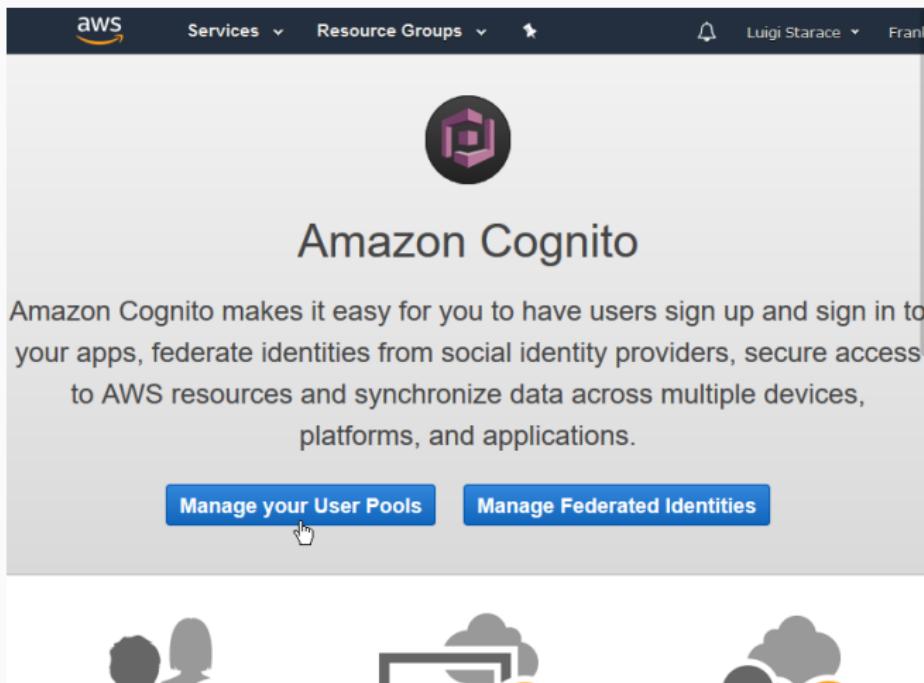
The screenshot shows the AWS Mobile Hub console interface. At the top, there is a navigation bar with the AWS logo, "Mobile Hub", a user profile for "Luigi Starace", and a "Support" dropdown. Below the header, the main content area is titled "Your Projects". It features a callout message: "Have your own app? Create a project to cloud enable your app with AWS services." with "Create" and "Import" buttons. A list of projects is displayed, with one item named "serverless-webapp" highlighted in blue. This project entry includes a "REGION" field set to "EU (Frankfurt)" and a "CREATED" field set to "April 21, 2018". At the bottom, there is a section titled "Starter Kits and Tutorials" with the subtext: "No app? Kick the tires with one of our cloud enabled starter kits. Or follow a step-by-step tutorial to cloud enable a sample app yourself."

## STEP 7: CREATE A COGNITO USER POOL

First we're gonna need a Cognito User Pool to authenticate our users. Let's create one.

## STEP 7: CREATE A COGNITO USER POOL

Visit the Cognito Console.



## STEP 7: CREATE A COGNITO USER POOL



# STEP 7: CREATE A COGNITO USER POOL

Insert a name for your user pool.

## What do you want to name your user pool?

Give your user pool a descriptive name so you can easily identify it in the future.

Pool name

serverless-webapp-user-pool

## How do you want to create your user pool?

### Review defaults

Start by reviewing the defaults and then customize as desired

### Step through settings

Step through each setting to make your choices

## STEP 7: CREATE A COGNITO USER POOL

Make sure only an email is required.

### Which standard attributes do you want to require?

All of the standard attributes can be used for user profiles, but the attributes you select will be required for sign up. You will not be able to change these requirements after the pool is created. If you select an attribute to be an alias, users will be able to sign-in using that value or their username. [Learn more about attributes.](#)

Required	Attribute	Required	Attribute
<input type="checkbox"/>	address	<input type="checkbox"/>	nickname
<input type="checkbox"/>	birthdate	<input type="checkbox"/>	phone number
<input checked="" type="checkbox"/>	email	<input type="checkbox"/>	picture
<input type="checkbox"/>	family name	<input type="checkbox"/>	preferred username
<input type="checkbox"/>	gender	<input type="checkbox"/>	profile
<input type="checkbox"/>	given name	<input type="checkbox"/>	zoneinfo
<input type="checkbox"/>	locale	<input type="checkbox"/>	updated at
<input type="checkbox"/>	middle name	<input type="checkbox"/>	website
<input type="checkbox"/>	name		

# STEP 7: CREATE A COGNITO USER POOL

Review your configuration and create the pool.

aws User Pools | Federated Identities Luigi Starace EU (Frankfurt) Support

### Create a user pool

[Cancel](#)

Name

Attributes

Policies

MFA and verifications

Message customizations

Tags

Devices

App clients

Triggers

**Review**

Required attributes  [Edit](#)

Alias attributes [Choose alias attributes...](#)

Username attributes [Choose username attributes...](#)

Custom attributes [Choose custom attributes...](#)

Minimum password length  [Edit](#)

Password policy  [Edit](#)

User sign ups allowed?  [Edit](#)

MFA [Enable MFA...](#) [Edit](#)

Verifications  [Edit](#)

Tags [Choose tags for your user pool...](#) [Edit](#)

App clients [Add app client...](#) [Edit](#)

Triggers [Add triggers...](#) [Edit](#)

[Create pool](#)

## STEP 8: CONFIGURE USER SIGN-IN IN THE MOBILE APP CONSOLE

Return to the AWS Mobile Console and open your project.

The screenshot shows the AWS Mobile Hub interface. At the top, there's a navigation bar with the AWS logo, "Mobile Hub", a user dropdown for "Luigi Starace", and a "Support" link. Below the header, the main title "Your Projects" is displayed in bold. A sub-instruction below it reads: "Have your own app? Create a project to cloud enable your app with AWS services." There are two buttons: a blue "Create" button and a white "Import" button. The main content area lists a single project: "serverless-webapp". To the right of the project name is a three-dot ellipsis menu icon. Below the project name, there are two columns of information: "REGION" (EU (Frankfurt)) and "CREATED" (April 21, 2018). At the bottom of the page, under the heading "Starter Kits and Tutorials", there is a descriptive text: "No app? Kick the tires with one of our cloud enabled starter kits. Or follow a step-by-step tutorial to cloud enable a sample app yourself."

# STEP 8: CONFIGURE USER SIGN-IN IN THE MOBILE APP CONSOLE

Add user sign in to the project.

## Add More Backend Features



### User Sign-in

Let your users sign in with public identity providers or your own identity system.

Powered by Amazon Cognito



### NoSQL Database

Store data in a fully managed cloud database.

Powered by Amazon DynamoDB



### User File Storage

Store files in the cloud.

Powered by Amazon S3



### Conversational Bots

Add voice and chat bots to your mobile app.

Powered by Amazon Lex



## STEP 8: CONFIGURE USER SIGN-IN IN THE MOBILE APP CONSOLE

Import your newly created user pool.

**Add sign-in Providers**

  
Email and Password

  
Facebook Login

  
Google Sign-In

  
SAML Federation

---

**Create new or import**

Create a new user pool  
Create a basic user pool powered by Cognito

Import an existing user pool  
Use one of your existing Cognito user pools

---

**Select user pool**

X

< 1 >

Name	Id
 serverless-webapp-user-pool	eu-central-1_kznXIIlDc

## STEP 8: CONFIGURE USER SIGN-IN IN THE MOBILE APP CONSOLE

Pull your new project configuration with

```
D:\serverless-webapp> awsmobile pull
```

If you were to start the application locally with

```
D:\serverless-webapp> npm start
```

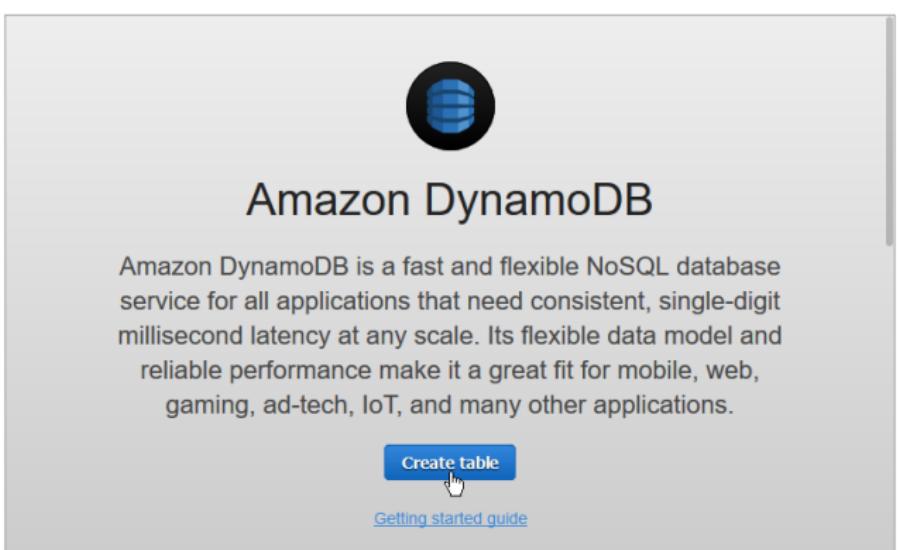
The authentication will now work.

## STEP 9: CONFIGURE AMAZON DYNAMO DATABASE

Next thing we're gonna need is a database to store the comments. In this tutorial we'll use the NoSQL database Amazon Dynamo.

## STEP 9: CONFIGURE AMAZON DYNAMO DATABASE

Visit the Dynamo Dashboard.



The screenshot shows the Amazon DynamoDB dashboard. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a star icon, a bell icon, and user names 'Luigi Starace' and 'Frank'. The main area features a large blue circular icon with horizontal stripes, followed by the text 'Amazon DynamoDB'. Below this, a descriptive paragraph explains that DynamoDB is a fast and flexible NoSQL database service suitable for various applications like mobile, web, gaming, ad-tech, IoT, etc. At the bottom of the main content area are two buttons: a blue 'Create table' button with a white hand cursor icon pointing to it, and a smaller blue 'Getting started guide' button.

Feedback English (US)

Privacy Policy Terms of Use

## STEP 9: CONFIGURE AMAZON DYNAMO DATABASE

Create a new Comments table as shown. Leave other fields with their default values.

The screenshot shows the 'Create DynamoDB table' interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, and user names 'Luigi Starace' and 'Frank'. Below the title, a descriptive text explains that DynamoDB is schema-less and requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

**Table name\***:  ⓘ

**Primary key\***: Partition key

User  ⓘ

Add sort key

Timestamp  ⓘ

**Table settings**

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

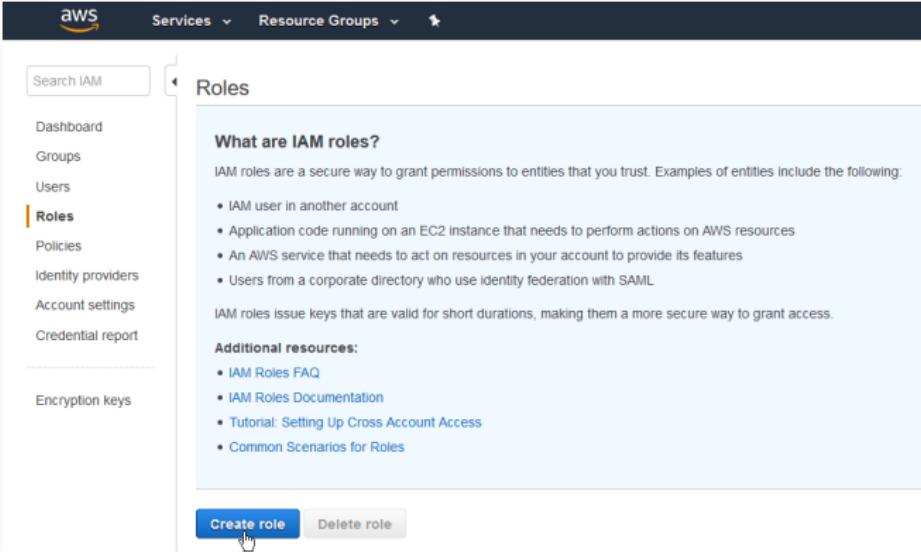
Feedback English (US) Privacy Policy Terms of Use

## STEP 10: CREATE A ROLE FOR THE LAMBDA FUNCTIONS

Before we create our Lambda functions, let's create a new role defining the authorizations we want them to have.

## STEP 10: CREATE A ROLE FOR THE LAMBDA FUNCTIONS

Return to the IAM console and select the “role” tab, then the “create role” button.



The screenshot shows the AWS IAM Roles page. The left sidebar has a navigation menu with options: Dashboard, Groups, Users, Roles (which is selected and highlighted with an orange border), Policies, Identity providers, Account settings, Credential report, and Encryption keys. The main content area is titled "Roles" and contains a section titled "What are IAM roles?". It describes IAM roles as a secure way to grant permissions to entities you trust, listing examples like IAM users in other accounts, application code running on EC2 instances, AWS services, and corporate directory users. It also notes that IAM roles issue short-lived keys for secure access. Below this, there's a section for "Additional resources:" with links to IAM Roles FAQ, Documentation, a tutorial on cross-account access, and common scenarios. At the bottom of the page are two buttons: a blue "Create role" button and a grey "Delete role" button. A cursor arrow is pointing at the "Create role" button.

## STEP 10: CREATE A ROLE FOR THE LAMBDA FUNCTIONS

Select “AWS Service” and “Lambda” in the wizard, as shown in the picture.

The screenshot shows the 'Create role' wizard in the AWS Lambda console. The top navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, and the user name 'Luigi Starace'. The main section is titled 'Create role' and 'Select type of trusted entity'. There are four options: 'AWS service' (selected, highlighted in blue), 'Another AWS account', 'Web identity', and 'SAML'. Below each option is a brief description and a 'Learn more' link. The next section is 'Choose the service that will use this role', with 'EC2' and 'Lambda' listed. 'Lambda' is selected and highlighted in blue, with a note: 'Allows Lambda functions to call AWS services on your behalf.' Below this is a table of services that can use the role:

API Gateway	Config	Elastic Container Service	Lex
AppSync	DMS	Elastic Transcoder	Machine Learning
Application Auto Scaling	Data Pipeline	Elastic Load Balancing	MediaConvert

At the bottom right are 'Cancel' and 'Next: Per' buttons.

## STEP 10: CREATE A ROLE FOR THE LAMBDA FUNCTIONS

Add the AWSLambdaBasicExecutionRole, as shown.

The screenshot shows the AWS Lambda 'Create role' interface. At the top, there are tabs for 'Services', 'Resource Groups', and a user icon. Below the tabs, the title 'Create role' is displayed. A sub-header 'Attach permissions policies' is followed by a note: 'Choose one or more policies to attach to your new role.' Two buttons are present: 'Create policy' and 'Refresh'. A search bar at the top right contains the text 'AWSLambdaBasicExecutionRole'. Below the search bar, a table header includes columns for 'Policy name', 'Attachments', and 'Description'. A single row is listed: 'AWSLambdaBasicExecutionRole' with '0' attachments and a description 'Provides write permissions to CloudWatch Logs.' The row is highlighted with a blue background. On the far right of the table, it says 'Showing 1 result'. Navigation icons 1, 2, and 3 are located on the right side of the table.

## STEP 10: CREATE A ROLE FOR THE LAMBDA FUNCTIONS

Insert a name and a description and create the role.

The screenshot shows the 'Create role' review step in the AWS Lambda console. The top navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, and the user 'Luigi Starace'. The main title is 'Create role' and the sub-section is 'Review'. A note below says 'Provide the required information below and review this role before you create it.' The 'Role name\*' field contains 'CommentsLambdaRole' with a descriptive placeholder. The 'Role description' field contains 'Allows Lambda functions to call AWS services on your behalf.' with a character limit note. Under 'Trusted entities', it lists 'AWS service: lambda.amazonaws.com'. At the bottom, there's a 'Policies' section with a blue cube icon and the policy 'AWSLambdaBasicExecutionRole' with a blue edit icon. Navigation buttons at the bottom right include 'Cancel', 'Previous', and a blue 'Create' button.

Create role

Review

Provide the required information below and review this role before you create it.

**Role name\*** CommentsLambdaRole  
Use alphanumeric and '+=\_,@\_-.' characters. Maximum 64 characters.

**Role description** Allows Lambda functions to call AWS services on your behalf.  
Maximum 1000 characters. Use alphanumeric and '+=\_,@\_-.' characters.

**Trusted entities** AWS service: lambda.amazonaws.com

**Policies** AWSLambdaBasicExecutionRole

**Cancel** **Previous** **Create**

## STEP 10: CREATE A ROLE FOR THE LAMBDA FUNCTIONS

Go back to the roles tab in the IAM Dashboard and select your newly created role.

The screenshot shows the AWS IAM Roles page. The left sidebar has a 'Roles' section selected, with other options like 'Dashboard', 'Groups', 'Users', 'Policies', 'Identity providers', 'Account settings', and 'Credential report'. The main area has a search bar with 'CommentsLambdaRole' typed in. Below it is a table with columns 'Role name', 'Description', and 'Trusted entities'. A single row is shown for 'CommentsLambdaRole', which is described as 'Allows Lambda functions to call A...' and has 'AWS service: lambda' under Trusted entities. A cursor is hovering over the 'CommentsLambdaRole' link in the table.

Role name	Description	Trusted entities
CommentsLambdaRole	Allows Lambda functions to call A...	AWS service: lambda

## STEP 10: CREATE A ROLE FOR THE LAMBDA FUNCTIONS

Add inline policies to allow the role to access Dynamo DB and Comprehend.

The screenshot shows the AWS IAM 'CommentsLambdaRole' summary page. The role ARN is arn:aws:iam::788880174327:role/CommentsLambdaRole. The role description is 'Allows Lambda functions to call AWS services on your behalf.' The path is '/'. The creation time is 2018-04-21 12:19 UTC+0200, and the maximum CLI/API session duration is 1 hour. The 'Permissions' tab is selected, showing one attached policy: 'AWSLambdaBasicExecutionRole' (an AWS managed policy). A button labeled '+ Add inline policy' is visible at the bottom right of the policy list.

Policy name	Policy type
AWSLambdaBasicExecutionRole	AWS managed policy

+ Add inline policy

## STEP 10: CREATE A ROLE FOR THE LAMBDA FUNCTIONS

Add an inline policy to allow this role to access Dynamo DB tables.

The screenshot shows the AWS IAM inline policy configuration page. On the left, there's a vertical sidebar with the word "Documentation". The main area has a dark header with the AWS logo, "Services", "Resource Groups", and user information ("Luigi Starace", "Global", "Support"). Below the header, there's a "DynamoDB (3 actions)" section. This section includes a "Service" dropdown set to "DynamoDB", a "Clone" and "Remove" button, and tabs for "Actions" (set to "Read") and "Resources". Under "Actions", it lists "GetItem", "Scan", and "PutItem". Under "Resources", it lists the ARN of a DynamoDB table: "arn:aws:dynamodb:eu-central-1:788880174327:table/Comments". At the bottom, there's a "Request conditions" section with a link to "Specify request conditions (optional)" and a blue "Add additional permissions" button.

# STEP 10: CREATE A ROLE FOR THE LAMBDA FUNCTIONS

Save the inline policy.

The screenshot shows the AWS IAM 'Create policy' interface. At the top, there's a navigation bar with the AWS logo, 'Services', 'Resource Groups', a bell icon, 'Luigi Starace', 'Global', and 'Support'. Below the navigation, the title 'Create policy' is displayed, followed by a progress indicator showing '1' in a white circle and '2' in a blue circle. The main section is titled 'Review policy' with the sub-instruction: 'Before you create this policy, provide the required information and review this policy.' A 'Name\*' field contains 'DynamoCommentsPolicy'. A note below it says: 'Maximum 128 characters. Use alphanumeric and '+,-,@,\_' characters.' On the left, a 'Summary' section includes a search bar labeled 'Filter' and a table with three columns: 'Service', 'Access level', and 'Resource'. Under 'Service', 'DynamoDB' is listed. Under 'Access level', 'Limited: Read, Write' is shown. Under 'Resource', 'TableName | string like | Com' is listed. At the bottom of the summary table, there are buttons for 'Allow (1 of 136 services)' and 'Show remaining 135'.

## STEP 10: CREATE A ROLE FOR THE LAMBDA FUNCTIONS

Now add another inline policy to allow this role to access Comprehend's detect language and detect sentiment features.

The screenshot shows the AWS IAM 'Create policy' interface. At the top, there are tabs for 'Visual editor' (which is selected) and 'JSON'. Below the tabs, there is a note about what a policy is and how to create it. There are also 'Import managed policy' and 'Actions' buttons. The main area displays a policy structure for the 'Comprehend' service, which includes actions for 'Read' (specifically 'DetectDominantLanguage' and 'DetectSentiment') and a note that all resources are selected. At the bottom, there is a section for 'Request conditions' and a button to 'Add additional permissions'.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "comprehend:DetectDominantLanguage",
            "Effect": "Allow",
            "Service": "comprehend"
        },
        {
            "Action": "comprehend:DetectSentiment",
            "Effect": "Allow",
            "Service": "comprehend"
        }
    ]
}
```

# STEP 10: CREATE A ROLE FOR THE LAMBDA FUNCTIONS

Your role should look like this.

The screenshot shows the AWS Lambda Role configuration page for the role 'CommentsLambdaRole'. The 'Summary' tab is selected. Key details shown include:

- Role ARN:** arn:aws:iam::788880174327:role/CommentsLambdaRole
- Role description:** Allows Lambda functions to call AWS services on your behalf. | [Edit](#)
- Instance Profile ARNs:** None
- Path:** /
- Creation time:** 2018-04-21 12:19 UTC+0200
- Maximum CLI/API session duration:** 1 hour [Edit](#)

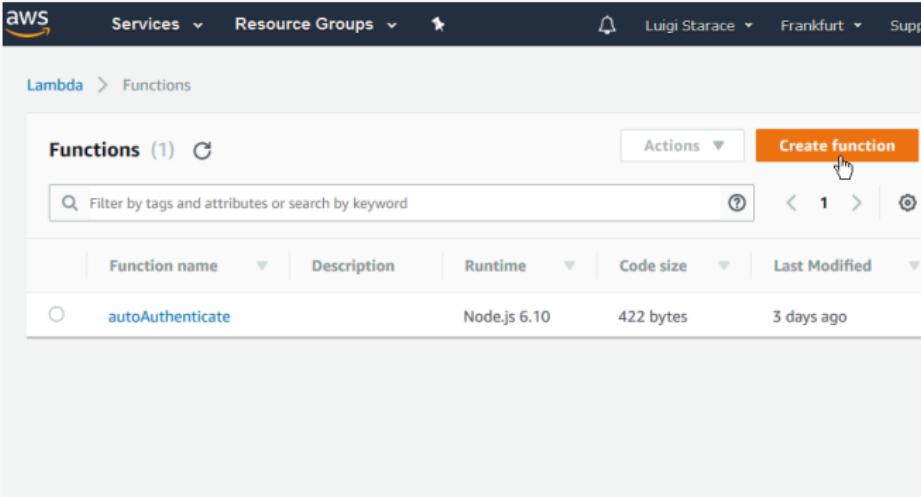
Below the summary, the 'Permissions' tab is active, showing the attached policies:

Policy name	Policy type	Actions
AWSLambdaBasicExecutionRole	AWS managed policy	X
ComprehendCommentsPolicy	Inline policy	X
DynamoCommentsPolicy	Inline policy	X

At the bottom right of the permissions section, there is a link: [Add inline policy](#).

## STEP 11: CREATE THE LAMBDA FUNCTIONS

Go to the Lambda Dashboard and click on the “Create function” button.



The screenshot shows the AWS Lambda service dashboard. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, user name 'Luigi Starace', location 'Frankfurt', and a 'Support' link. Below the navigation is a breadcrumb trail: 'Lambda > Functions'. The main area is titled 'Functions (1)'. It features a search bar with placeholder text 'Filter by tags and attributes or search by keyword'. To the right of the search bar are buttons for 'Actions' and 'Create function', with a hand cursor icon over the 'Create function' button. Below these are filter and sorting options: 'Function name', 'Description', 'Runtime', 'Code size', and 'Last Modified'. A single function entry is listed: 'autoAuthenticate' (Node.js 6.10, 422 bytes, last modified 3 days ago). The overall interface is clean with a light gray background and white header bars.

# STEP 11: CREATE THE LAMBDA FUNCTIONS

Select “Author from scratch”

The screenshot shows the AWS Lambda 'Create function' interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a star icon, a bell icon, and user names 'Luigi Starace' and 'Frank'. Below the navigation, the path 'Lambda > Functions > Create function' is shown. The main title is 'Create function'. There are three options: 'Author from scratch' (selected, indicated by a blue border and a blue dot), 'Blueprints' (indicated by a grey dot), and 'Serverless Application Repository' (indicated by a grey dot). The 'Author from scratch' section contains the sub-section 'Author from scratch' with the sub-sub-section 'Info'. Under 'Info', there are fields for 'Name' (containing 'insertComment') and 'Runtime' (containing 'Node.js').

## STEP 11: CREATE THE LAMBDA FUNCTIONS

Name the function `insertComment` and select Node.js 6.10 as the runtime and the role we created earlier as the role.

**Author from scratch** Info

Name

Runtime

Role  
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Existing role  
You may use an existing role with this function. Note that the role must be assumable by Lambda and must have Cloudwatch Logs permissions.

Cancel **Create function**

## STEP 11: CREATE THE LAMBDA FUNCTIONS

Insert the code provided in the lambda/insertComment.js file in the next screen, then save the lambda function.

## STEP 11: CREATE THE LAMBDA FUNCTIONS

Proceed similarly and create the getComments Lambda function.

## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Once we have our Lambda functions, let's hook 'em up with an API our web app can rely upon.

## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Visit the API Gateway Dashboard

The screenshot shows the Amazon API Gateway dashboard. At the top, there is a navigation bar with 'Services' and 'Resource Groups' dropdowns, a user icon for 'Luigi Starace', and a location dropdown for 'Frankfurt'. Below the navigation is the Amazon API Gateway logo, which consists of a yellow hexagonal icon with internal lines forming a cube-like structure. The main title 'Amazon API Gateway' is displayed in large, bold, black font. A descriptive paragraph explains the service's purpose: 'Amazon API Gateway helps developers to create and manage APIs to back-end systems running on Amazon EC2, AWS Lambda, or any publicly addressable web service. With Amazon API Gateway, you can generate custom client SDKs for your APIs, to connect your back-end systems to mobile, web, and server applications or services.' Below this text is a blue 'Get Started' button with a white outline, which has a small white cursor icon pointing to it. Underneath the button is the text 'Getting Started Guide'.

## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Create a new API and select a name and a description.

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a star icon, a bell icon, and a user profile 'Luigi Starace'. Below the navigation bar, the path 'Amazon API Gateway > APIs > Create' is visible. The main title is 'Create new API'. A note below it states: 'In Amazon API Gateway, an API refers to a collection of resources and methods that can be invoked through HTTPS endpoint:'. There are three radio button options: 'New API' (selected), 'Import from Swagger', and 'Example API'. The 'Settings' section is open, prompting the user to 'Choose a friendly name and description for your API.' It contains three input fields: 'API name\*' with the value 'serverless-webapp-api', 'Description' with the value 'REST API to store and retrieve comments.', and 'Endpoint Type' with the value 'Edge optimized'. At the bottom left, there's a note '\* Required'. At the very bottom, there are links for 'Feedback', 'English (US)', 'Privacy Policy', and 'Terms of Use'.

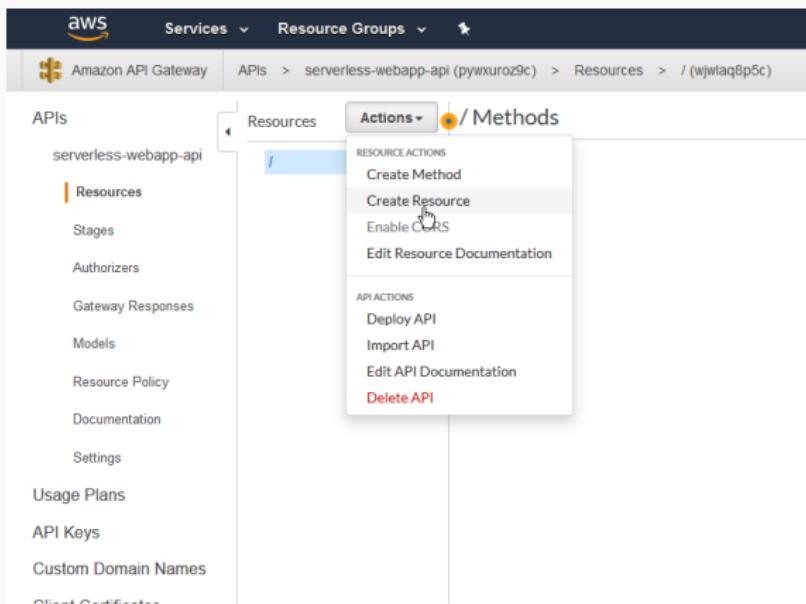
## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Select the Authorizers tab and create a new Authorizer for your API. Give it a name, select the user pool we created earlier, and enter “Authorization” in the “Token Source” field.

The screenshot shows the AWS Amazon API Gateway interface. The left sidebar has a tree structure with 'APIs' selected, followed by 'serverless-webapp-api'. Under 'serverless-webapp-api', 'Authorizers' is highlighted. The main content area is titled 'Authorizers' and contains the sub-instruction: 'Authorizers enable you to control access to your APIs using Amazon Cognito User Pools or a Lambda function.' A prominent blue button labeled '+ Create New Authorizer' is centered. Below it, a modal window titled 'Create Authorizer' is open. The 'Name \*' field contains 'serverless-webapp-api-authorizer'. The 'Type \*' section has two options: 'Lambda' (radio button unselected) and 'Cognito' (radio button selected). The 'Cognito User Pool \*' dropdown is set to 'eu-central-1 / serverless-webapp-user-pool'. The 'Token Source \*' field contains 'Authorization'. At the bottom of the modal are 'Create' and 'Cancel' buttons, with the 'Create' button being highlighted with a cursor icon.

## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Select the resources tab and create a new Resource



## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Name the resource comments, enable CORS and continue.

 New Child Resource

Use this page to create a new child resource for your resource. 

Configure as  proxy resource  

**Resource Name\***  

**Resource Path\***  

You can add path parameters using brackets. For example, the resource path **{username}** represents a path parameter called 'username'. Configuring **/[proxy+]** as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to **/foo**. To handle requests to **/**, add a new ANY method on the **/** resource.

Enable API Gateway CORS  

---

\* Required  

## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Select the comments resource and create a new POST method.

The screenshot shows the AWS API Gateway console interface. The top navigation bar includes the AWS logo, Services dropdown, Resource Groups dropdown, and a star icon. Below the navigation is a breadcrumb trail: APIs > serverless-webapp-api (pywxuroz9c) > Resources > /comments (hhn412). On the left, a sidebar menu lists various API management components: APIs, Resources (which is selected and highlighted in orange), Stages, Authorizers, Gateway Responses, Models, Resource Policy, Documentation, Settings, Usage Plans, API Keys, Custom Domain Names, and Client Certificates. The main content area displays the resources for the 'serverless-webapp-api'. Under the 'Resources' section, there is a tree view showing a single node: '/comments'. A context menu is open over this node, with the 'Actions' tab selected. The menu items under 'OPTIONS' are: Create Method (highlighted with a blue circle), Create Resource (with a cursor arrow pointing at it), Enable CORS, Edit Resource Documentation, and Delete Resource. Under 'API ACTIONS', the options are: Deploy API, Import API, Edit API Documentation, and Delete API. To the right of the tree view, there is a large, mostly empty, light-gray rectangular area.

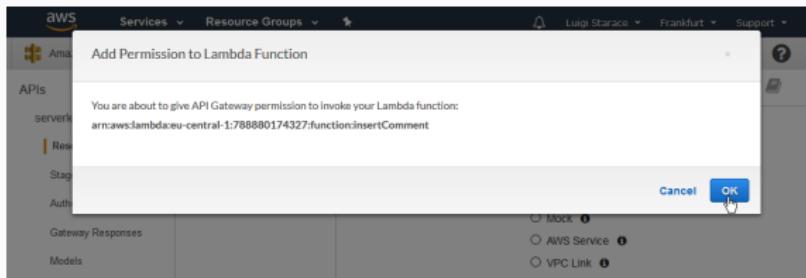
## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

As shown, select the insertComment Lambda function you created earlier as the integration point.

The screenshot shows the AWS API Gateway configuration interface for a POST method on the '/comments' resource. The left sidebar lists 'OPTIONS' and 'POST' methods. The 'POST' method is selected, highlighted with a blue background. The main panel title is '/comments - POST - Setup'. A sub-instruction says 'Choose the integration point for your new method.' Below it, the 'Integration type' section is set to 'Lambda Function' (radio button selected). Other options include 'HTTP', 'Mock', 'AWS Service', and 'VPC Link'. The 'Use Lambda Proxy integration' checkbox is checked. The 'Lambda Region' dropdown is set to 'eu-central-1'. Under 'Lambda Function', the name 'insertComment' is typed into the input field. The 'Use Default Timeout' checkbox is checked. At the bottom right is a blue 'Save' button.

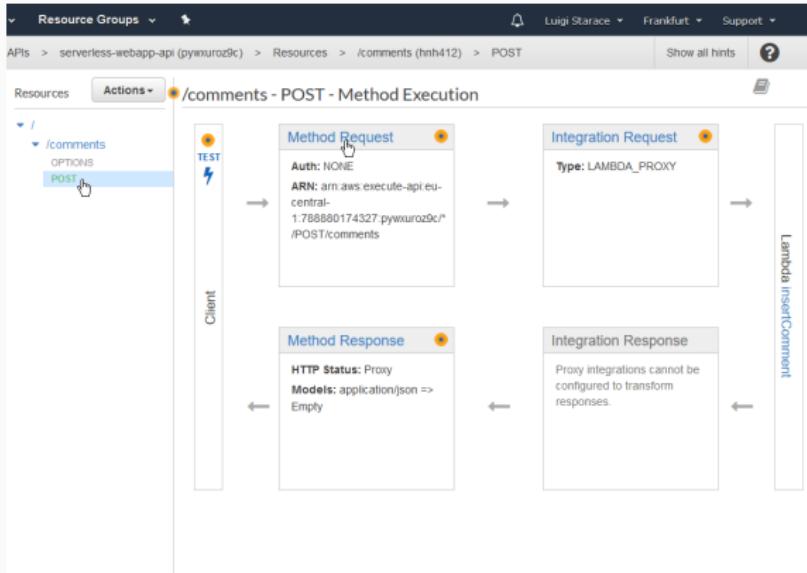
## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Give API Gateway the permission to invoke the Lambda function



## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Select the POST method on the comments resource, then select the Method Request card.



## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Select the authorizer you created earlier for the Authorization field in the Settings section.

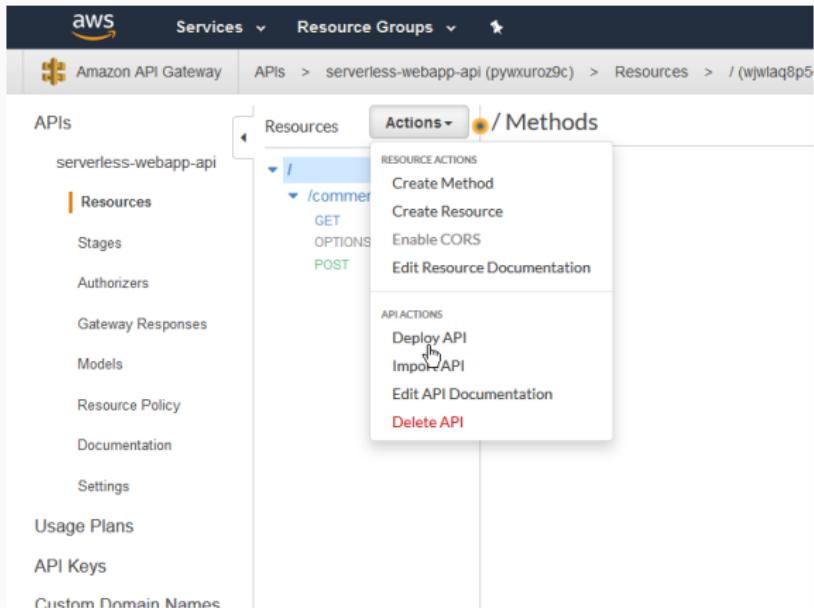
The screenshot shows the AWS Lambda API settings page. The URL in the browser is: APIs > serverless-webapp-api (pywxuroz9c) > Resources > /comments (hhn412) > POST. On the left, under 'Resources', there's a tree view with a node expanded to show '/comments' with 'OPTIONS' and 'POST' methods. The 'POST' method is selected. In the main panel, the title is '/comments - POST - Method Request'. Below it, a note says 'Provide information about this method's authorization settings and the parameters it can receive.' Under 'Settings', there are three fields: 'Authorization' (set to 'serverless-webapp-api-authorizer'), 'Request Validator' (set to 'NONE'), and 'API Key Required' (set to 'false'). An orange arrow points to the 'Authorization' dropdown menu. The 'Authorization' dropdown menu is open, showing the selected value 'serverless-webapp-api-authorizer' and three other options: 'AWS Lambda', 'AWS IAM User', and 'AWS IAM Role'.

## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Proceed similarly to hook up the GET method with the getComments Lambda function. This time authorization is not needed. We want non-authenticated users to be able to fetch the comments.

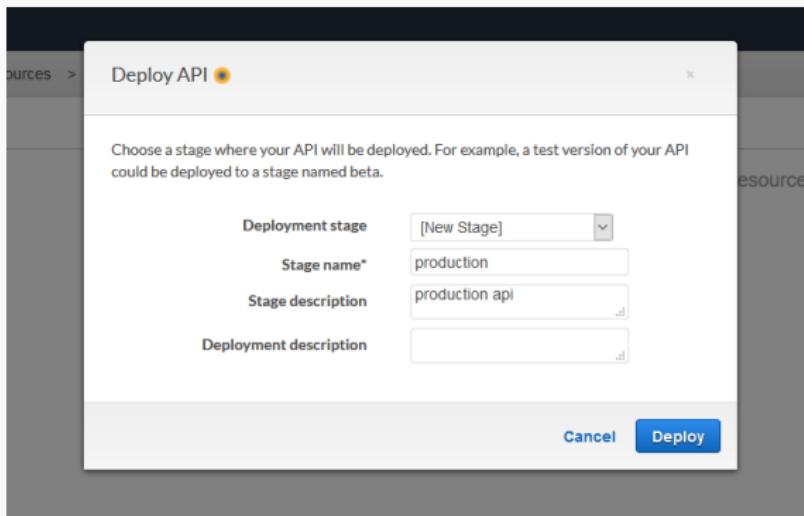
## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Once you are done setting up the GET method, select the root resource, then Deploy API.



## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Insert deployment stage informations and deploy.



## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Select the stages tab and note the invoke url.

The screenshot shows the AWS API Gateway Stage Editor for the 'production' stage of the 'serverless-webapp-api'. The left sidebar lists various API resources like 'APIs', 'Resources', 'Stages' (which is selected), 'Authorizers', etc. The main panel is titled 'production Stage Editor' and contains tabs for 'Settings', 'Logs', 'Stage Variables', 'SDK Generation', 'Export', 'Deployment History', 'Documentation History', and 'Canary'. A prominent orange box highlights the 'Invoke URL' field, which contains the value `https://pywuroz9c.execute-api.eu-central-1.amazonaws.com/production`. An orange arrow points from the top right towards this field. Below the URL, there's a section for 'Cache Settings' with a checkbox for 'Enable API cache' (unchecked) and a 'Default Method Throttling' section with a note about rate and burst settings.

## STEP 12: CREATE THE APIs TO EXPOSE THE LAMBDA FUNCTIONS

Change the CommentsAPI class accordingly in  
src/API/CommentsAPI.js.

```
class CommentsAPI {  
  
    constructor(){  
        this.endpoint = '<YOUR_INVOKE_URL_HERE>';  
    }  
  
    // ...  
}
```

# STEP 13: HOST THE STATIC FILES WITH S3

Visit the S3 Dashboard and create a new bucket.

The screenshot shows the AWS S3 dashboard. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a user icon for 'Luigi Starace', and a 'Global' dropdown. Below the navigation is a dark blue header bar with the text 'Documentation'. The main content area has a title 'Amazon S3' with a puzzle piece icon, a 'Discover the new console' link, and a 'Quick tips' link. A search bar labeled 'Search for buckets' is present. Below the search bar is a button labeled '+ Create bucket' with a hand cursor icon pointing at it. To its right are buttons for 'Delete bucket', 'Empty bucket', '1 Buckets', '0 Public', '1 Regions', and a refresh icon. A table below lists one bucket: 'Bucket name' is 'elasticbeanstalk-eu-central-1-788880174327'; 'Access' is 'Not public'; 'Region' is 'EU (Frankfurt)'; and 'Date created' is 'Apr 13, 2018 7:11:22 PM GMT+0200'. A small note at the bottom states: '\* Objects might still be publicly accessible due to object ACLs. [Learn more](#)'.

## STEP 13: HOST THE STATIC FILES WITH S3

Select your newly created S3 bucket.

The screenshot shows the Amazon S3 console interface. At the top, there is a navigation bar with the Amazon S3 logo, a search bar labeled "Search for buckets", and links for "Discover the new console" and "Quick tips". Below the navigation bar, there are buttons for "+ Create bucket", "Delete bucket", and "Empty bucket". It also displays "1 Regions" and "3 Buckets" with "0 Public" buckets. A "refresh" icon is also present. The main table lists the buckets:

Bucket name	Access	Region	Date created
elasticbeanstalk-eu-central-1-78...	Not public *	EU (Frankfurt)	Apr 13, 2018 7:11:22 PM GMT+0200
serverless-webapp-bucket	Not public *	EU (Frankfurt)	Apr 22, 2018 8:38:17 AM GMT+0200

## STEP 13: HOST THE STATIC FILES WITH S3

Under the properties tab, select the Static website hosting card.

The screenshot shows the AWS S3 Properties tab for a bucket named "serverless-webapp-bucket". The "Properties" tab is selected, indicated by a hand cursor icon. The page displays four configuration cards: Versioning, Server access logging, Static website hosting, and Object-level logging. The "Static website hosting" card is highlighted with a blue border and a hand cursor icon, indicating it is the active or selected card. The other cards have a grey border. Each card contains a brief description, a "Learn more" link, and a status indicator (radio button) labeled "Disabled".

Card	Description	Status
Versioning	Keep multiple versions of an object in the same bucket.	Disabled
Server access logging	Set up access log records that provide details about access requests.	Disabled
Static website hosting	Host a static website, which does not require server-side technologies.	Disabled
Object-level logging	Record object-level API activity using the CloudTrail data events feature (additional cost).	Disabled

## STEP 13: HOST THE STATIC FILES WITH S3

Fill the form as shown in the picture. Note the Endpoint, as it will be the URL of the website!

Static website hosting X

Endpoint : <http://serverless-webapp-bucket.s3-website.eu-central-1.amazonaws.com>

Use this bucket to host a website [Learn more](#)

Index document [i](#)

Error document [i](#)

Redirection rules (optional) [i](#)

Redirect requests [Learn more](#)

Disable website hosting

[Cancel](#) [Save](#)

## STEP 13: HOST THE STATIC FILES WITH S3

Now we'll show how to upload the static website via AWS CLI. This operation can also be performed via the web interface of the bucket.

## STEP 13: HOST THE STATIC FILES WITH S3

Install AWS CLI

```
D:\serverless-webapp> pip install awscli --upgrade
```

Then configure it

```
D:\serverless-webapp> aws configure
AWS Access Key ID [None]: <YOUR_ACCESS_KEY_ID>
AWS Secret Access Key [None]: <YOUR_SECRET_ACCESS_KEY>
Default region name [None]: eu-central-1
Default output format [None]: json
```

## STEP 13: HOST THE STATIC FILES WITH S3

Build the website

```
D:\serverless-webapp> npm run build-css  
D:\serverless-webapp> npm run build
```

Then upload the files with

```
D:\serverless-webapp> aws s3 sync ./build s3://  
serverless-webapp-bucket --acl public-read
```

## STEP 13: HOST THE STATIC FILES WITH S3

After the upload is done, your bucket should look like this.

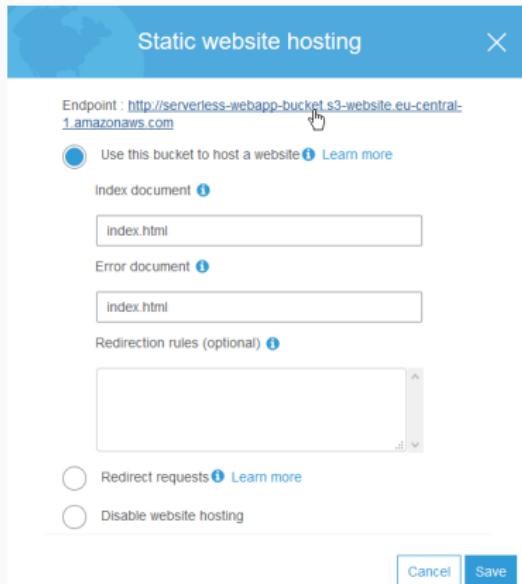
The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, 'Luigi Starace' (user profile), 'Global' dropdown, and 'Support' dropdown. Below the navigation bar, the path 'Amazon S3 > serverless-webapp-bucket' is displayed. The main area has four tabs: 'Overview', 'Properties', 'Permissions' (which is selected and highlighted in orange), and 'Management'. A search bar with placeholder text 'Type a prefix and press Enter to search. Press ESC to clear.' is present. Below the search bar are three buttons: 'Upload', '+ Create folder', and 'More'. To the right of these buttons is the location 'EU (Frankfurt)' with a refresh icon. The main content area displays a table of files in the 'static' folder. The columns are 'Name', 'Last modified', 'Size', and 'Storage class'. The table shows the following data:

Name	Last modified	Size	Storage class
asset-manifest.json	Apr 22, 2018 8:50:08 AM GMT+0200	263.0 B	Standard
favicon.ico	Apr 22, 2018 8:50:08 AM GMT+0200	3.8 KB	Standard
index.html	Apr 22, 2018 8:50:08 AM GMT+0200	742.0 B	Standard
manifest.json	Apr 22, 2018 8:50:08 AM GMT+0200	332.0 B	Standard
service-worker.js	Apr 22, 2018 8:50:08 AM GMT+0200	3.2 KB	Standard

At the bottom of the table area, there's a footer bar with the text 'Viewing 1 to 6'.

## STEP 13: HOST THE STATIC FILES WITH S3

Visit the Static website hosting card again under the properties tab, then click on the endpoint URL.



## STEP 13: HOST THE STATIC FILES WITH S3

You should see a very nice single-page serverless web application!



# But wait, there's more!

Such a nice web application wouldn't be complete without a global CDN to speed up load times. So we'll now set up Amazon CloudFront to distribute the static files all over the globe.

## STEP 14: OPTIMIZE LATENCY WITH CLOUDFRONT

Visit the CloudFront Dashboard and create a new distribution.

The screenshot shows the AWS CloudFront dashboard. The left sidebar has a dark background with white text and includes links for Distributions, What's New (marked with a star), Reports & Analytics, Cache Statistics, Monitoring and Alarms, Popular Objects, Top Referrers, Usage, Viewers, and Security. The main content area has a light gray background and displays the title "Amazon CloudFront Getting Started" in large bold letters. Below the title is a message: "Either your search returned no results, or you do not have any distributions. Click the button below to create a new CloudFront distribution. A distribution allows you to distribute content using a worldwide network of edge locations that provide low latency and high data transfer speeds ([learn more](#))". At the bottom of this message is a blue rectangular button with the white text "Create Distribution". A hand cursor icon is positioned over this button, indicating it is clickable.

# STEP 14: OPTIMIZE LATENCY WITH CLOUDFRONT

Select web delivery method.

The screenshot shows the AWS CloudFront 'Select delivery method' wizard. The top navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, 'Luigi Starace' name, 'Global' dropdown, and a 'Support' link. On the left, a sidebar has 'Step 1: Select delivery method' highlighted in orange, and 'Step 2: Create distribution' below it. The main content area has a title 'Select a delivery method for your content.' with a help icon. It features two sections: 'Web' and 'RTMP'. The 'Web' section is active, showing a sub-section titled 'Create a web distribution if you want to:' with a bulleted list: 'Speed up distribution of static and dynamic content, for example, .html, .css, .php, and graphics files.', 'Distribute media files using HTTP or HTTPS.', 'Add, update, or delete objects, and submit data from web forms.', and 'Use live streaming to stream an event in real time.' Below this, a note says 'You store your files in an origin - either an Amazon S3 bucket or a web server. After you create the distribution, you can add more origins to the distribution.' A large blue 'Get Started' button is at the bottom of the 'Web' section. The 'RTMP' section is partially visible below it.

Step 1: Select delivery method

Step 2: Create distribution

## Select a delivery method for your content. ?

### Web

Create a web distribution if you want to:

- Speed up distribution of static and dynamic content, for example, .html, .css, .php, and graphics files.
- Distribute media files using HTTP or HTTPS.
- Add, update, or delete objects, and submit data from web forms.
- Use live streaming to stream an event in real time.

You store your files in an origin - either an Amazon S3 bucket or a web server. After you create the distribution, you can add more origins to the distribution.

**Get Started**

### RTMP

Create an RTMP distribution to speed up distribution of your streaming media files using Adobe Flash Media Server's RTMP protocol. An RTMP distribution allows an end user to begin playing a media file before the file has finished downloading from a CloudFront edge location. Note the following:

- To create an RTMP distribution, you must store the media files in an Amazon S3 bucket.
- To use CloudFront live streaming, create a web distribution.

## STEP 14: OPTIMIZE LATENCY WITH CLOUDFRONT

Select your website bucket as the origin.

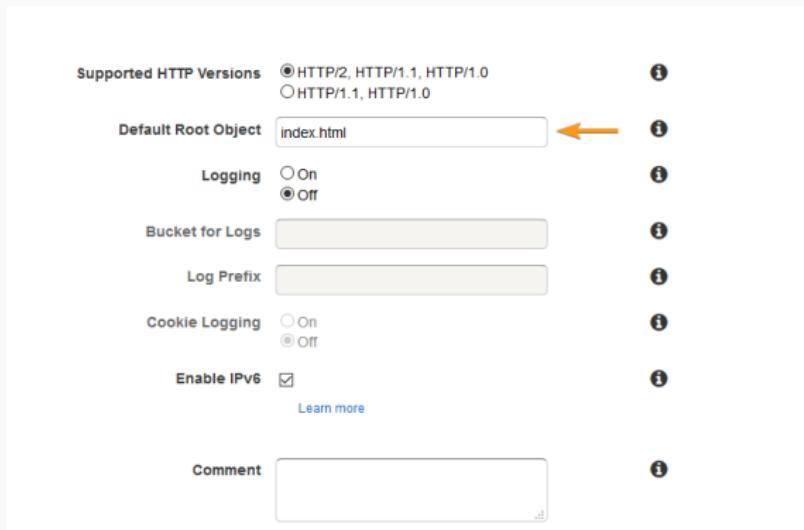
Create Distribution ?

### Origin Settings

Origin Domain Name	serverless-webapp-bucket.s3.amazonaws
Origin Path	— Amazon S3 Buckets — <b>serverless-webapp-bucket.s3.amazonaws</b>
Origin ID	S3-serverless-webapp-bucket
Restrict Bucket Access	<input type="radio"/> Yes <input checked="" type="radio"/> No
Origin Custom Headers	Header Name <input type="text"/>

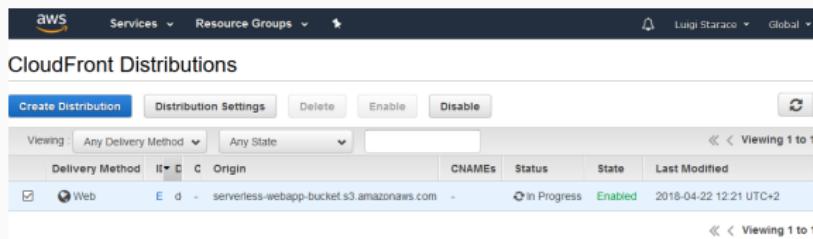
## STEP 14: OPTIMIZE LATENCY WITH CLOUDFRONT

Select index.html as the default root object.



## STEP 14: OPTIMIZE LATENCY WITH CLOUDFRONT

It takes a few minutes to setup the distribution. When it's done the status will change to Deployed.



The screenshot shows the AWS CloudFront Distributions page. At the top, there are buttons for 'Create Distribution' (highlighted in blue), 'Distribution Settings', 'Delete', 'Enable', and 'Disable'. Below this is a search bar with dropdowns for 'Any Delivery Method' and 'Any State', and a 'Viewing 1 to 1' indicator. A table lists the distribution details:

Delivery Method	If	C	Origin	CNAMEs	Status	State	Last Modified
<input checked="" type="checkbox"/> Web	E	d	- serverless-webapp-bucket.s3.amazonaws.com	-	<input checked="" type="radio"/> In Progress	<input checked="" type="radio"/> Enabled	2018-04-22 12:21 UTC+2

At the bottom, there are navigation links for 'Viewing 1 to 1'.

## STEP 14: OPTIMIZE LATENCY WITH CLOUDFRONT

In the distribution detail page add a custom error response as shown in the picture below to make sure 404 errors are handled by the application.

The screenshot shows the 'Create Custom Error Response' dialog box from the AWS CloudFront console. The top navigation bar includes the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, and a star icon. The main title is 'Create Custom Error Response'. Below it is a section titled 'Custom Error Response Settings'. The configuration fields are as follows:

- HTTP Error Code:** A dropdown menu set to '404: Not Found'.
- Error Caching Minimum TTL (seconds):** An input field containing '300'.
- Customize Error Response:** A radio button group where 'Yes' is selected.
- Response Page Path:** An input field containing 'index.html'.
- HTTP Response Code:** A dropdown menu set to '200: OK'.

At the bottom right are 'Cancel' and 'Create' buttons.

## STEP 14: OPTIMIZE LATENCY WITH CLOUDFRONT

Now you can visit the application from the cloudfront URL

The screenshot shows the AWS CloudFront Distributions page. At the top, there are navigation links for 'AWS' and 'Resource Groups', and a user profile for 'Luigi Starace'. Below the header, the title 'CloudFront Distributions' is displayed. A horizontal toolbar contains buttons for 'Create Distribution', 'Distribution Settings', 'Delete', 'Enable', and 'Disable'. Underneath the toolbar, there are filters for 'Viewing : Any Delivery Method' and 'Any State'. The main table lists a single distribution entry:

	Delivery	Domain Name	Origin
<input type="checkbox"/>	Web	dqhlmixviyrw.cloudfront.net	serverless-webapp-bucket.s3.amazonaws.com

A cursor is hovering over the 'Web' link in the Domain Name column of the table.

## TAKE HOME MESSAGES

---

## TAKE HOME MESSAGES

- Cloud computing and service models

## TAKE HOME MESSAGES

- Cloud computing and service models
- AWS

## TAKE HOME MESSAGES

- Cloud computing and service models
- AWS
- Deploy a “classic” web application on AWS

## TAKE HOME MESSAGES

- Cloud computing and service models
- AWS
- Deploy a “classic” web application on AWS
- FaaS and serverless computing

## TAKE HOME MESSAGES

- Cloud computing and service models
- AWS
- Deploy a “classic” web application on AWS
- FaaS and serverless computing
- Build and deploy a serverless one-page web application on AWS

## SECURITY RECOMMENDATIONS

---

- Be **very** careful not to expose your IAM credentials;

## SECURITY RECOMMENDATIONS

- Be **very** careful not to expose your IAM credentials;
- Enforce the least privilege principle: each user should only be able to access the minimum resources necessary to fulfill its purpose.

# SECURITY RECOMMENDATIONS

- The very second you expose your credentials to the public, some bot may use them to spin up large numbers of EC2 instances. If that happens, the billing might be a scary surprise!

 Back to the tutorial

---

IAM best practices  web

Git Secrets - Github repository  web

# SECURITY RECOMMENDATIONS

- The very second you expose your credentials to the public, some bot may use them to spin up large numbers of EC2 instances. If that happens, the billing might be a scary surprise!
- Tools like the AWS-developed git-secrets help avoiding the exposure of IAM credentials

 Back to the tutorial

---

IAM best practices  web

Git Secrets - Github repository  web

## NICE READINGS I

- [Jan16] Badri Janakiraman. *Serverless*. June 20, 2016. URL: <https://martinfowler.com/bliki/Serverless.html> (visited on 05/21/2018).
- [LF14] James Lewis and Martin Fowler. *Microservices: a definition of this new architectural term*. Mar. 25, 2014. URL: <https://martinfowler.com/articles/microservices.html> (visited on 05/21/2018).
- [Rob16] Mike Roberts. *Serverless Architectures*. Apr. 6, 2016. URL: <https://martinfowler.com/articles/serverless.html> (visited on 05/21/2018).

## NICE READINGS II

- [Rus16] Mark Russinovich. *Microservices: An application revolution powered by the cloud.* Mar. 17, 2016. URL: <https://azure.microsoft.com/it-it/blog/microservices-an-application-revolution-powered-by-the-cloud/> (visited on 05/21/2018).
- [Ser] Serverless inc. *Serverless guide.* URL: <https://github.com/serverless/guide> (visited on 05/21/2018).

## REFERENCES I

- [Amaa] Amazon Web Services. *What is cloud computing?*. URL: <https://aws.amazon.com/what-is-cloud-computing/> (visited on 03/30/2018).
- [Amab] Inc. Amazon Web Services. *Set up a Continuous Deployment Pipeline using AWS CodePipeline*. URL: <https://aws.amazon.com/it/getting-started/tutorials/continuous-deployment-pipeline/> (visited on 06/10/2018).

## REFERENCES II

- [Amac] Inc. Amazon Web Services. *Tutorial: Create a Four-Stage Pipeline.* URL: <https://docs.aws.amazon.com/codepipeline/latest/userguide/tutorials-four-stage-pipeline.html> (visited on 06/10/2018).
- [Ama17] Inc. Amazon Web Services. *Practicing Continuous Integration and Continuous Delivery on AWS.* Tech. rep. June 2017. URL: <https://d1.awsstatic.com/whitepapers/DevOps/practicing-continuous-integration-continuous-delivery-on-AWS.pdf> (visited on 06/01/2018).

## REFERENCES III

---

- [AWS] AWS. *AWS Step Functions*. URL:  
[https://aws.amazon.com/step-functions/?nc1=f\\_ls](https://aws.amazon.com/step-functions/?nc1=f_ls) (visited on 05/01/2018).
- [Gar17] Gartner. *Gartner Forecasts Worldwide Public Cloud Services Revenue to Reach \$260 Billion in 2017*. Oct. 12, 2017. URL: <https://www.gartner.com/newsroom/id/3815165> (visited on 03/30/2018).

## REFERENCES IV

---

- [Syn18] Synergy Research Group. *Cloud Growth Rate Increases; Amazon, Microsoft & Google all Gain Market Share*. Feb. 2, 2018. URL: <https://www.srgresearch.com/articles/cloud-growth-rate-increases-amazon-microsoft-google-all-gain-market-share> (visited on 03/30/2018).