AWS Foundation

AWS Lambda, Elastic Beanstalk & AWS OpsWorks

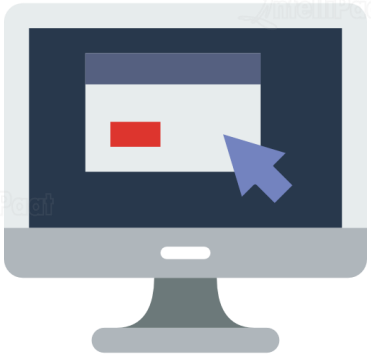
Agenda

1. Distributed Application Architecture
2. What is AWS Lambda?
3. How Lambda is Different?
4. Benefits & Limitations of Lambda
5. How does Lambda work?
6. Use Cases of Lambda
7. Lambda Concepts
8. Using Lambda with S3 & Hands-on
9. What is Elastic Beanstalk?
10. How does Beanstalk work?
11. Elastic Beanstalk Concepts
12. Demo: Launching a sample application
13. Elastic Beanstalk Pricing
14. What is Configuration Management?
15. What is AWS OpsWorks?
16. AWS OpsWorks Benefits
17. CloudFormation vs OpsWorks
18. AWS OpsWorks Services
19. AWS OpsWorks Stacks
20. Demo: Launching a stack
21. AWS OpsWorks Pricing
22. Quiz

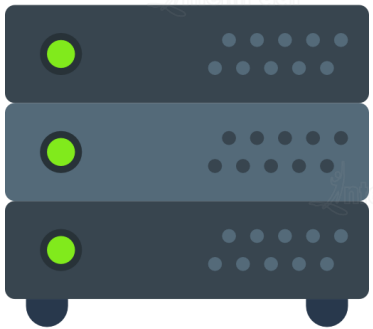
Distributed Application Architecture

Distributed Application Architecture

A typical application has three components – Frontend, Backend, and Database or other storage types



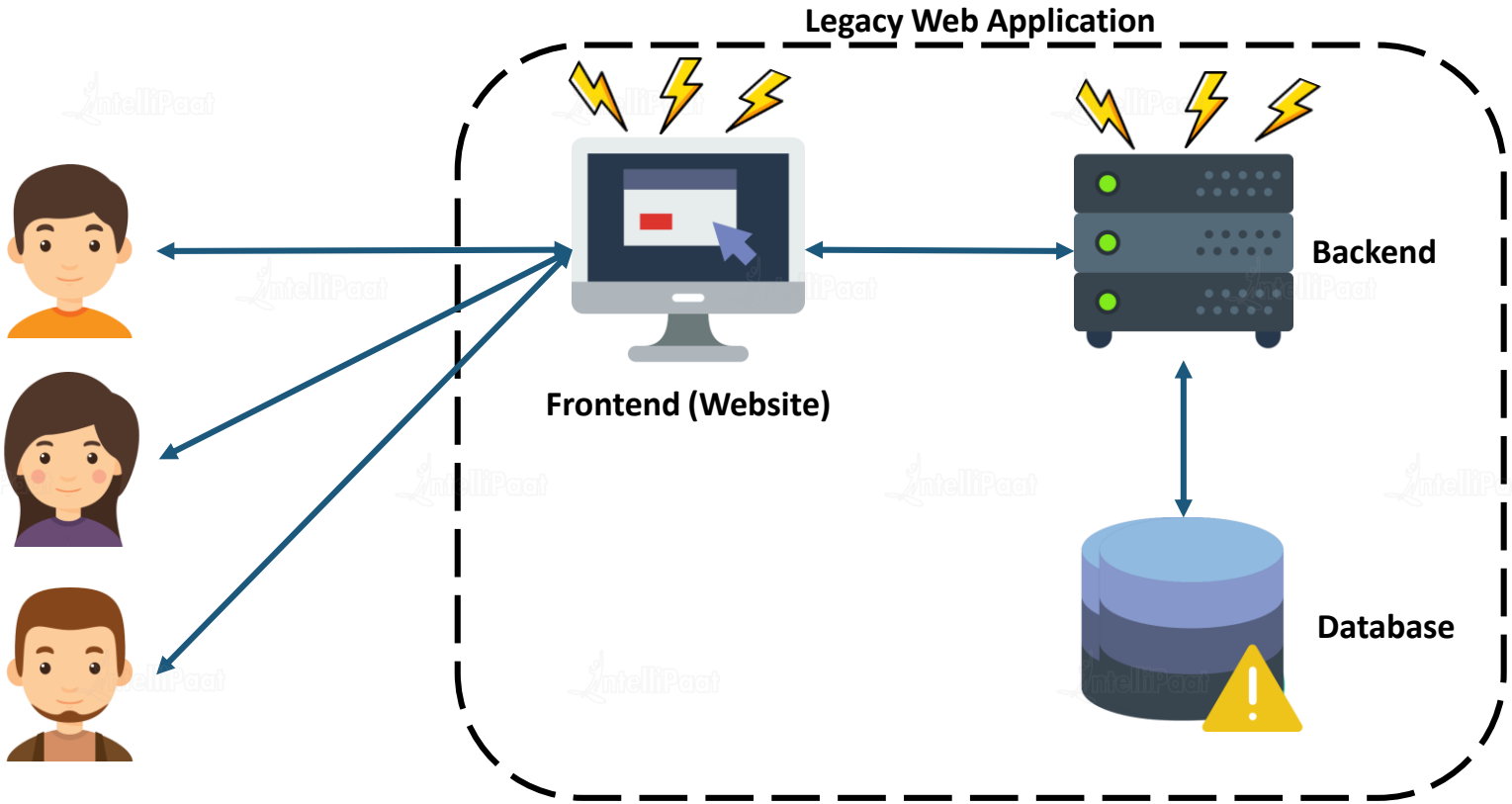
Frontend (Website)



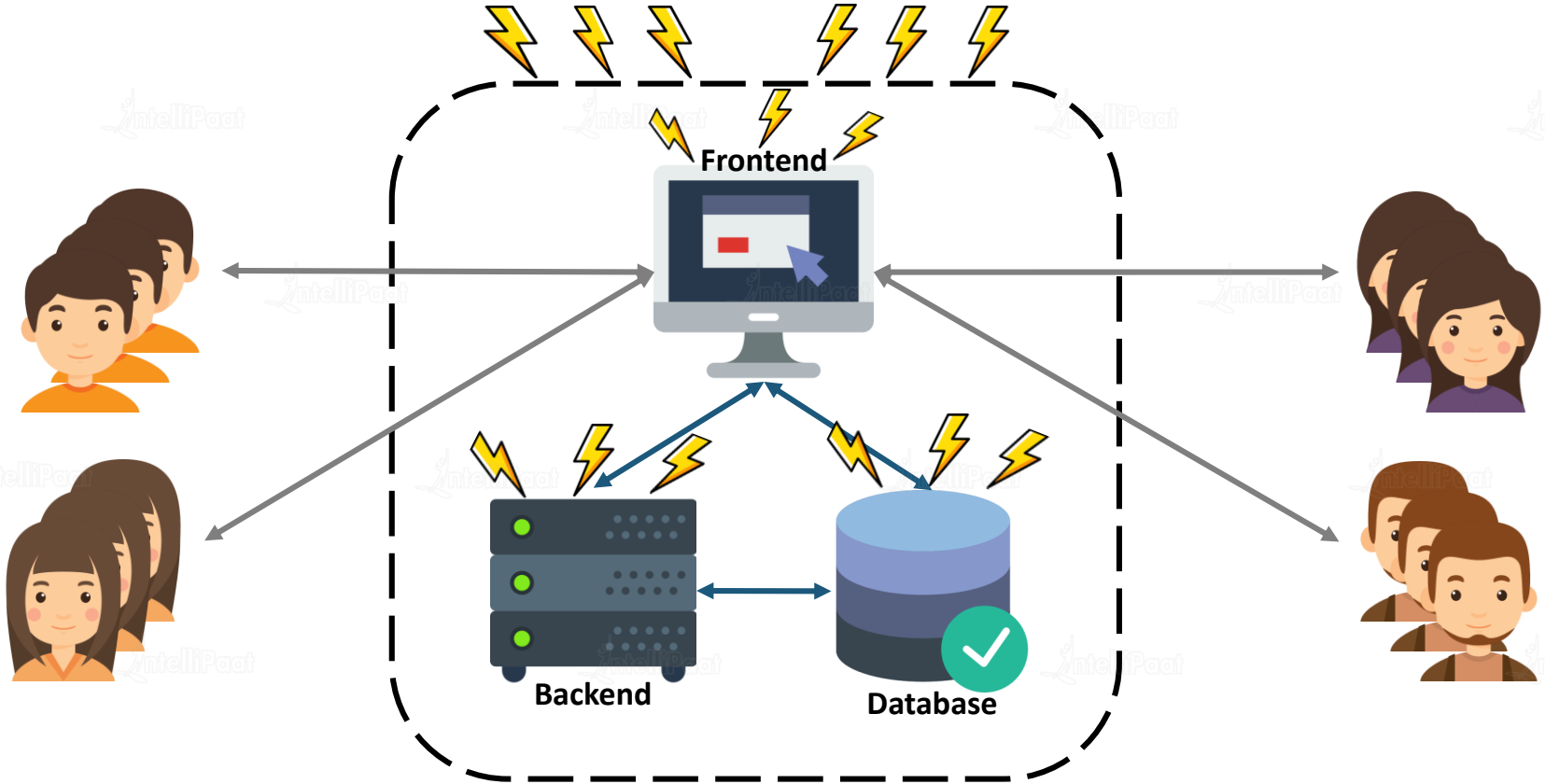
Backend (Logic)

Database Service

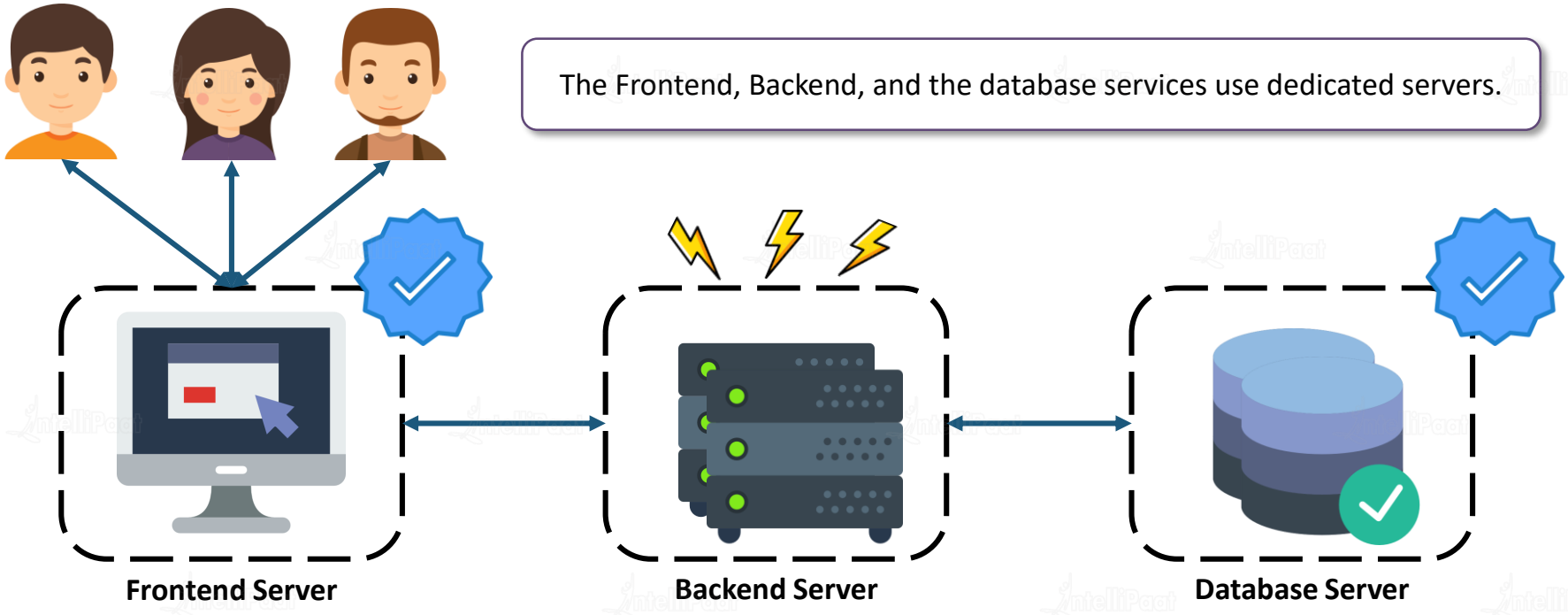
Distributed Application Architecture



Distributed Application Architecture



Distributed Application Architecture



The Frontend, Backend, and the database services use dedicated servers.

Frontend Server

Only contains the design and user interface

Backend Server

All the processing and logical operations are done here

Database Server

Database server is only triggered when it is queried

What is AWS Lambda?

What is AWS Lambda?

AWS Lambda is a ‘serverless’ compute service where your code is triggered in response to a event occurrence and it also manages the underlying compute resources for you!

How Lambda is Different?

How Lambda is Different?

AWS Lambda

- AWS Lambda is a Platform as a Service (PaaS) with a remote platform to run & execute your back-end code.
- Environment restrictions, as it is restricted to few languages only.
- Just choose your environment where you want to run your code and push the code into AWS Lambda.
- No flexibility to log in to compute instances, choose customized operating system or language runtime.

Amazon EC2

- AWS EC2 is an Infrastructure as a Service (IaaS) provided with the virtualized computing resources.
- No environment restrictions are there.
- For the first time in EC2, you have to choose the OS and Install all the software required and then push your code in EC2.
- Offers the flexibility to choose the variety of instances, custom operating systems, network & security patches etc.

How Lambda is Different?

AWS Lambda

- AWS Lambda is used only for running and executing your Back-end code
- You cannot select the AWS resources, like a type of EC2 instance, lambda provides you resources based on your workload.
- It is a stateless system.

Elastic Beanstalk

- Deploy and manage the apps on AWS Cloud without worrying about the infrastructure that runs those applications.
- Freedom to select AWS resources, like, choose an EC2 instance type which is optimal for your application.
- It is a stateful system.

Benefits & Limitations of Lambda

Benefits of AWS Lambda

- Serverless Architecture
- Code freely
- No VM needs to be created
- Pay-as-you-go
- Monitor you performance

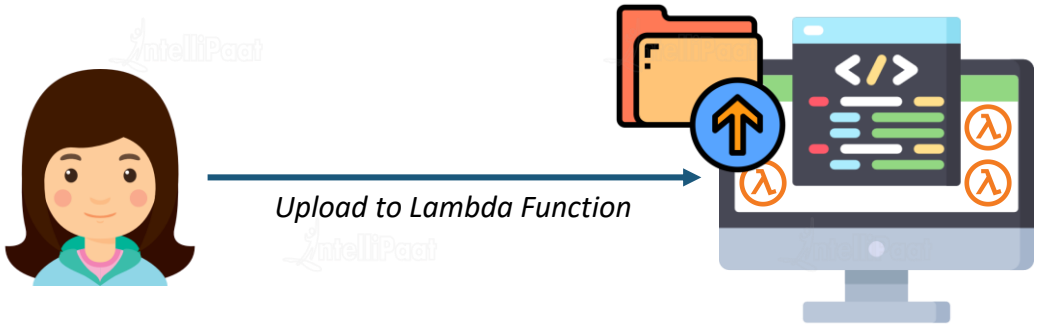
Limitations of AWS Lambda

- The maximum disk space provided is 512 MB for the runtime environment.
- Its memory volume varies between 128 to 3008 MB to the function while execution
- The function timeout is set to only 900 seconds (15 Minutes). Default is 3 seconds.
- Only available languages in the Lambda editor can be used

How does it work?

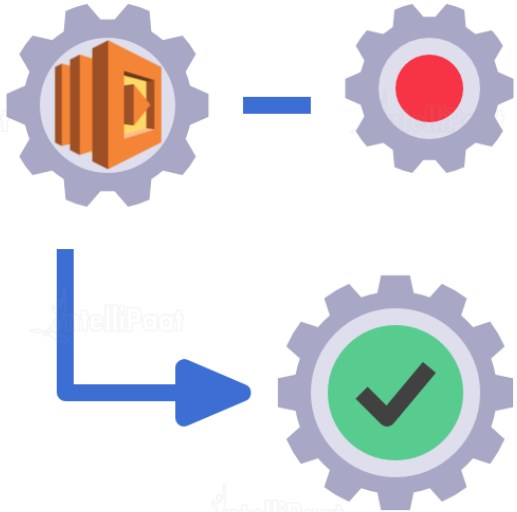
How does it work?

- First you upload your code to Lambda in one or more lambda functions
- AWS Lambda will then execute the code in your behalf
- After the code is invoked, Lambda automatically takes care of provisioning and managing the required servers



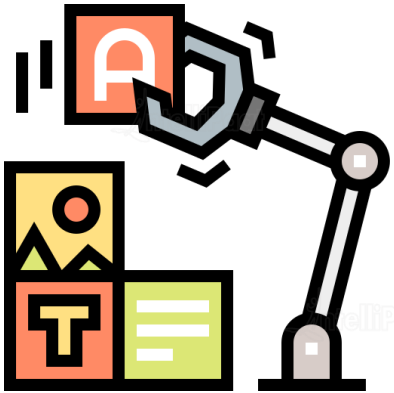
How does it work?

- First you upload your code to Lambda in one or more lambda functions
- AWS Lambda will then execute the code in your behalf
- After the code is invoked, Lambda automatically takes care of provisioning and managing the required servers



How does it work?

- First you upload your code to Lambda in one or more lambda functions
- AWS Lambda will then execute the code in your behalf
- After the code is invoked, Lambda automatically takes care of provisioning and managing the required servers



Provisioning servers

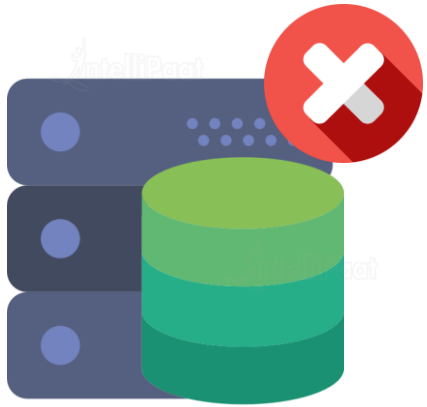
Monitor and manage servers

Use Cases of Lambda

Use Cases of Lambda

- Serverless Websites
- Automated Backups
- Filter and Transform data

- Building a serverless website allows you to focus on your website code.
- You don’t have to manage and operate its infrastructure.
- Hosting your static website over S3 and then making it available using AWS Lambda makes it easier for the users to keep track of their resources being used.



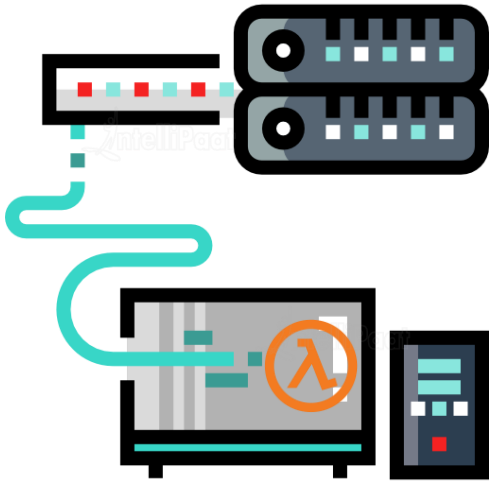
Use Cases of Lambda

Serverless Websites

Automated Backups

Filter and
Transform data

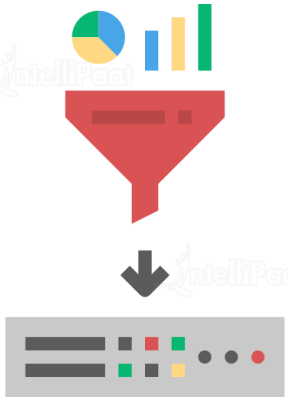
- One can easily schedule the Lambda events and create back-ups in their AWS Accounts.
- Create the back-ups, check if there are any idle resources or not, generate reports using Lambda in no time.



Use Cases of Lambda

- Serverless Websites
- Automated Backups
- Filter and Transform data

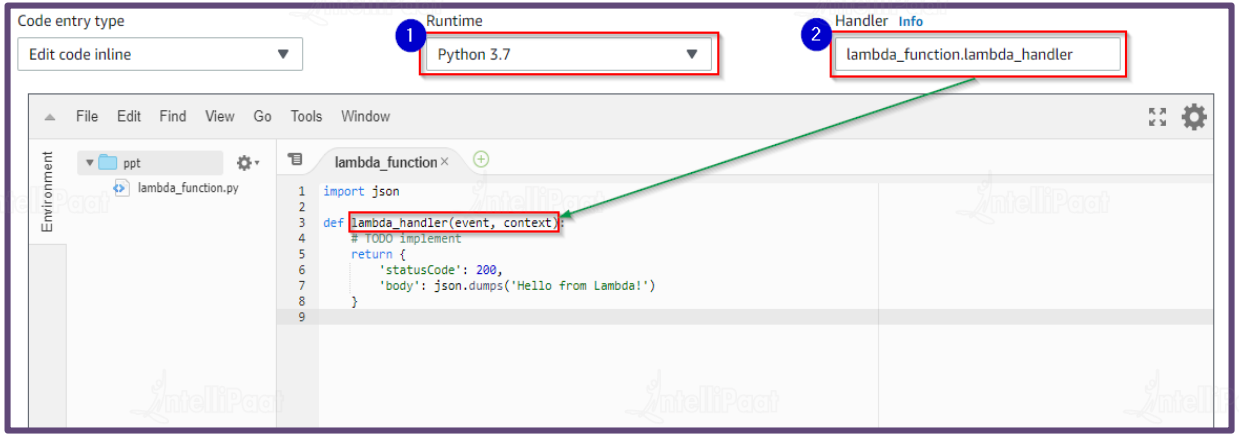
- One can easily use it for transferring the data between Lambda and other Amazon Services like S3, Kinesis, Redshift and database services
- Also, you can filter the data before sending
- One can easily transform and load data between Lambda and these services.



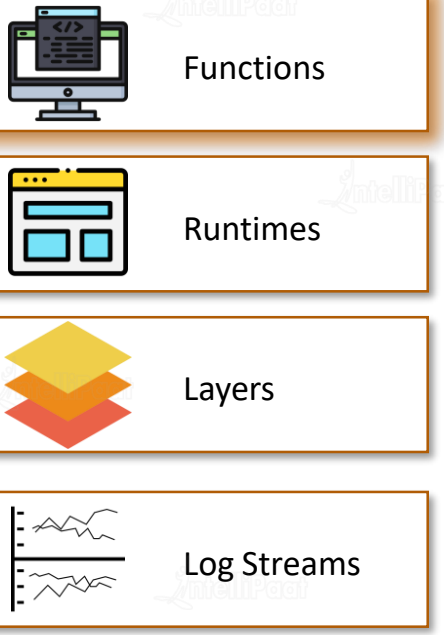
Lambda Concepts

Lambda Concepts

- A script or program that runs in AWS Lambda. Lambda passes invocation events to your function.
- The function processes an event and returns a response.



Lambda Concepts



Function Settings

- **Code** – the logic you use in the lambda function
- **Runtime** – lambda runtime executes your function
- **Handler** – the method your runtime executes when your function is invoked
- **Tags** – Key-value pairs your function is attached with
- **Description** – describing the function
- **Timeout** – maximum time a function is allowed to execute

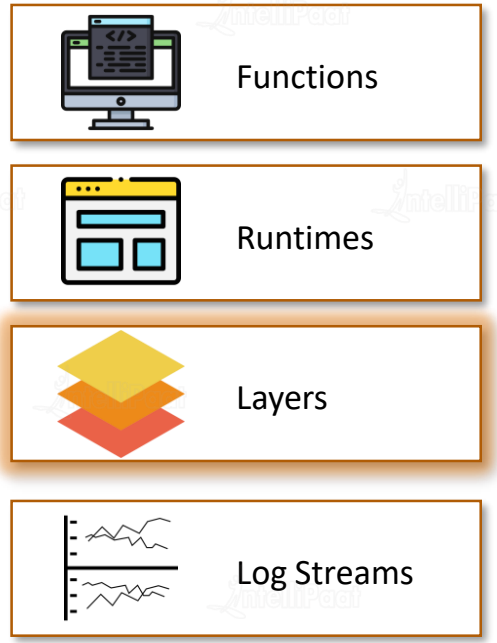
Lambda Concepts

- Lambda runtimes allow functions in different languages to run in the same base execution environment.
- The runtime sits in-between the Lambda service and your function code, relaying invocation events, context information, and responses between the two.
- You can use runtimes provided by Lambda, or build your own.

Latest supported
.NET Core 2.1 (C#/PowerShell)
Go 1.x
Java 8
Node.js 10.x
Python 3.7
Ruby 2.5

Other supported
.NET Core 1.0 (C#)
Node.js 8.10
Python 2.7
Python 3.6

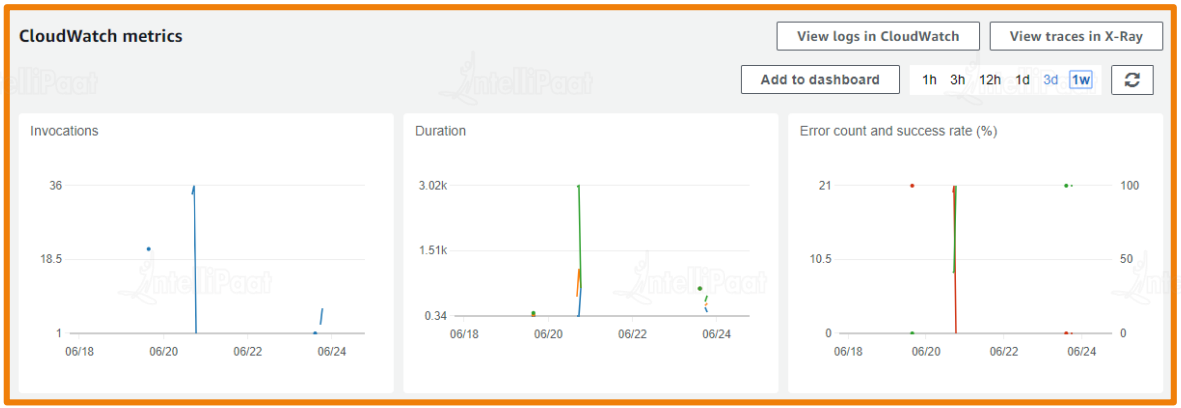
Lambda Concepts



- Lambda layers are a distribution mechanism for libraries, custom runtimes, and other function dependencies.
- Layers let you manage your in-development function code independently from the unchanging code and resources that it uses.
- You can configure your function to use layers that you create, layers provided by AWS, or layers from other AWS customers.

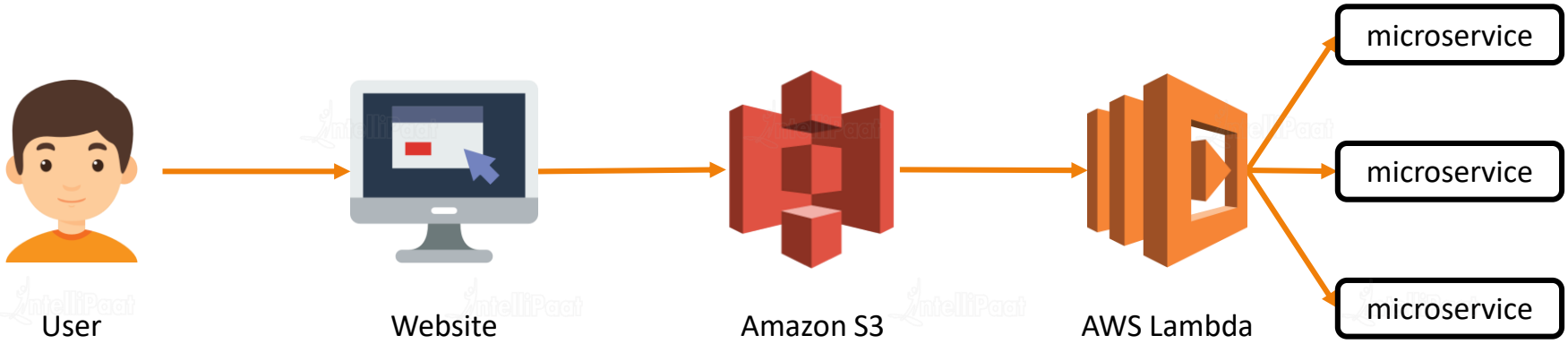
Lambda Concepts

- Lambda automatically monitors your function invocations and reports metrics to CloudWatch.
- You can annotate your function code with custom logging statements that allow you to analyze the execution flow and performance of your Lambda function to ensure it's working properly.



Using AWS Lambda with S3

Using AWS Lambda with S3



Hands-on

Hands-on

- Create a function
- Provide the trigger as S3 and choose the bucket
- Upload an object to that bucket
- This should have triggered the Lambda function. Go to the monitoring tab under the Lambda console and check the log stream to confirm the invocation of the function

Lambda pricing

Lambda Pricing

Free Tier

1M REQUESTS

per month

400,000 GB-SECONDS

of compute time per month.

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

Requests

1M REQUESTS FREE

First 1M requests per month are free.

$0.2 PER 1M REQUESTS THEREAFTER

$0.0000002 per request.

Duration

400,000 GB-SECONDS PER MONTH FREE

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

$0.0000166667 FOR EVERY GB-SECOND USED THEREAFTER

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

What is Elastic Beanstalk?

What is Elastic Beanstalk?

- Beanstalk is a service where you can create, run and manage application without worrying about the underlying infrastructure. It is a PaaS.
- You can directly upload your website code to Beanstalk, and it will automatically host the application for you with a URL. You can concentrate on the code of your application rather than the architecture which it is hosted on.

What is Elastic Beanstalk?

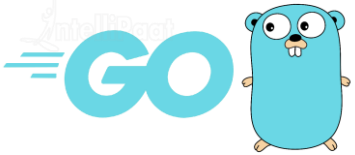
Beanstalk supported programming languages

Beta
- Corretto 11
- Corretto 8

Generic
- Docker
- Multi-container Docker

Preconfigured
- Elastic Beanstalk Packer Builder
- Go
- .NET (Windows/IIS)
- Java
- Node.js
- Ruby
- PHP
- Python
- Tomcat

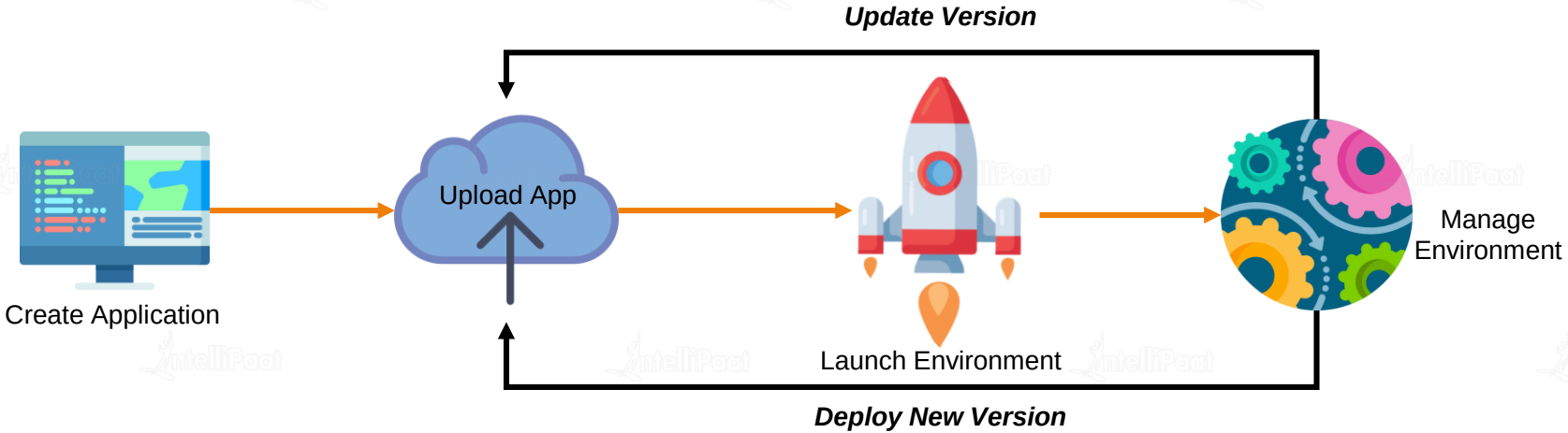
Preconfigured – Docker
- GlassFish
- Go
- Python



How does Beanstalk work?

How does Beanstalk work?

You can simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring. We will look at the details of it.



Elastic Beanstalk Concepts

Elastic Beanstalk Concepts

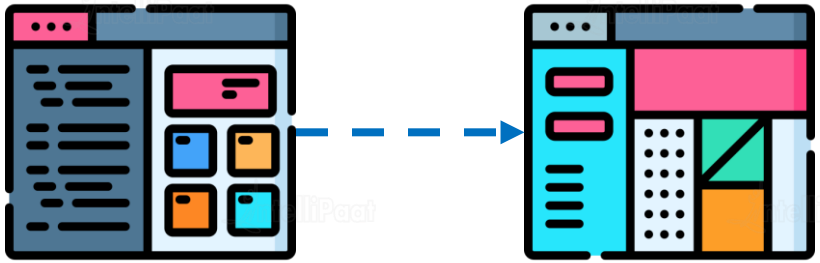
- **Application**
- **Application Version**
- **Environment**
- **Environment Tier**
- **Environment Configuration**
- **Saved Configuration**
- **Platform**

In Beanstalk, an application is the folder you upload which is a logical collection of the app version, environment and configurations

Elastic Beanstalk Concepts

- Application
- Application Version
- Environment
- Environment Tier
- Environment Configuration
- Saved Configuration
- Platform

It is a specific iteration of the code which is uploaded to Beanstalk. For example, if first version is 1 then the next would be 1.1 and so on.

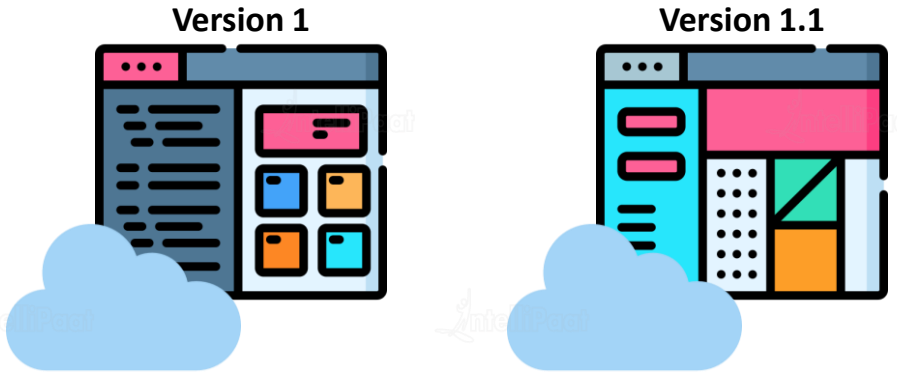


Version 1 Version 1.1

Elastic Beanstalk Concepts

- Application
- Application Version
- Environment
- Environment Tier
- Environment Configuration
- Saved Configuration
- Platform

An environment is a collection of AWS resources on which your application runs on. In an environment, at a time period, only one version of an application can be running. If you want to run multiple application versions at the same time, then you will have to create more environments and run an other application version on it



Elastic Beanstalk Concepts

- Application
- Application Version
- Environment
- Environment Tier
- Environment Configuration
- Saved Configuration
- Platform

When you launch an environment in beanstalk, you will have to choose one of the tiers given below. The tier decides the AWS resources which should be assigned for that environment and the type of application which it can run.

Web Server Environment

- Using this environment you can directly upload your application's code and get it hosted
- You will be provided with a URL like this **myapp.us-west-2.elasticbeanstalk.com**

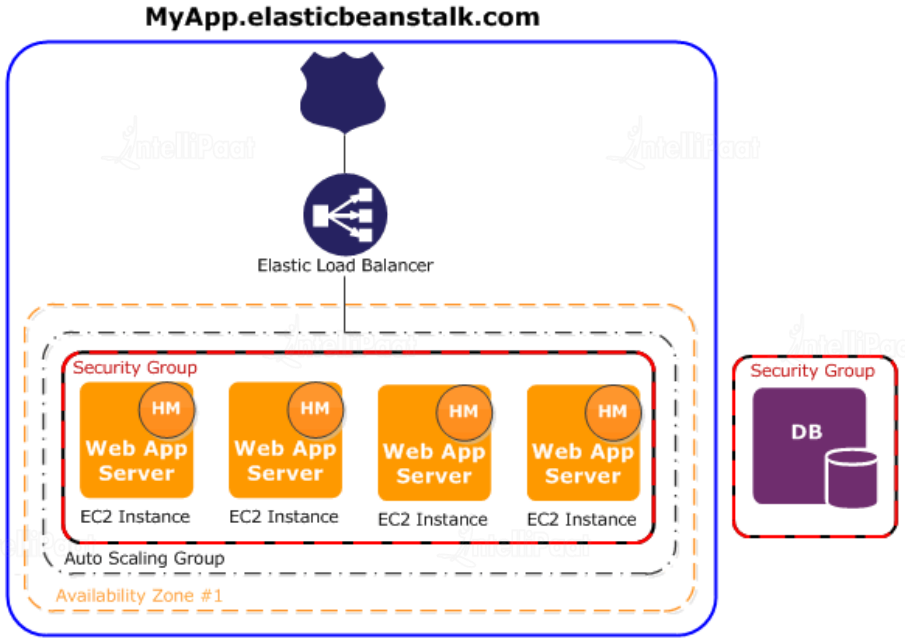
Worker Environment

- If you have operations in your web server which might take a long time to complete, you can offload them to the worker
- For example, a task which processes videos or one which generates a ZIP archive

Elastic Beanstalk Concepts

- Application
- Application Version
- Environment
- Environment Tier
- Environment Configuration
- Saved Configuration
- Platform

Web Server Environment Architecture

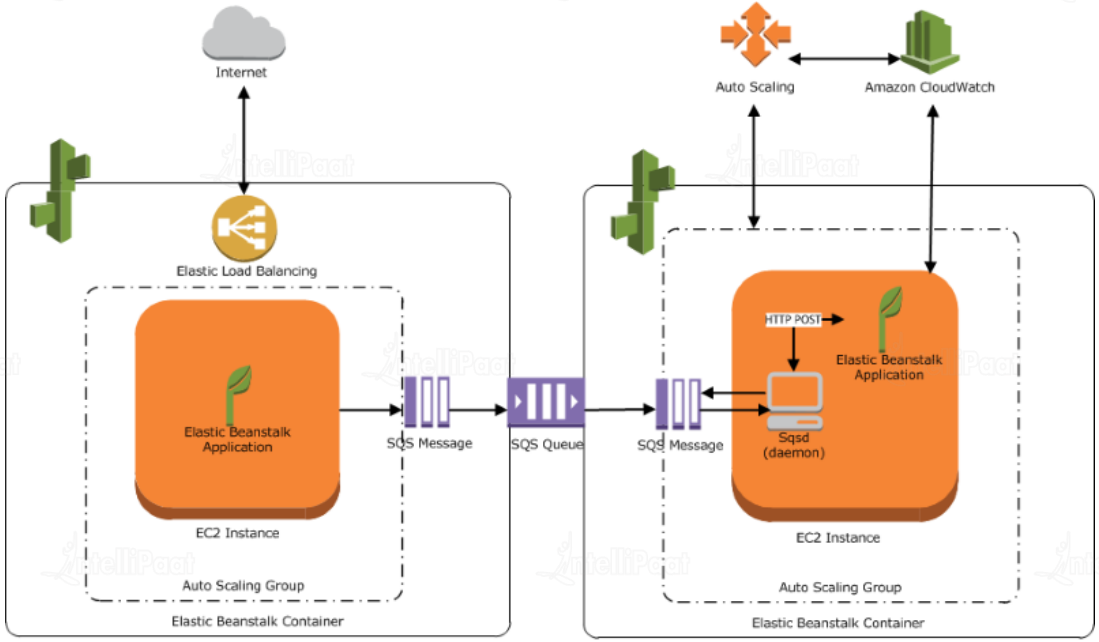


Source: docs.aws.amazon.com

Elastic Beanstalk Concepts

- Application
- Application Version
- Environment
- Environment Tier
- Environment Configuration
- Saved Configuration
- Platform

Worker Environment Architecture



Web Server Environment

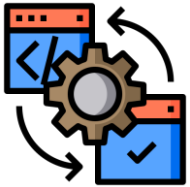
Worker Environment

Source: docs.aws.amazon.com

Elastic Beanstalk Concepts

- Application
- Application Version
- Environment
- Environment Tier
- **Environment Configuration**
- Saved Configuration
- Platform

These are the parameters which you enter while creating an environment. Let's say you have selected Node.js as your runtime, then it will be completely installed on the servers which it spins up.



- You can also update your configuration whenever you need
- Beanstalk will either apply the changes to existing resources or deletes and deploys new resources

Elastic Beanstalk Concepts

- Application
- Application Version
- Environment
- Environment Tier
- Environment Configuration
- **Saved Configuration**
- Platform

It is a template which you can create. These templates contain the configuration details and you can directly use them to create your own unique environment. Even if you delete the environment which is running, you can create the exact same environment with this template.

Elastic Beanstalk Concepts

- Application
- Application Version
- Environment
- Environment Tier
- Environment Configuration
- Saved Configuration
- Platform

Combination of an operating system, programming language runtime, web server, application server, and Elastic Beanstalk components

- Packer Builder
- Single Container Docker
- Multicontainer Docker
- Preconfigured Docker
- Go
- Java SE
- Java with Tomcat
- .NET on Windows Server with IIS
- Node.js
- PHP
- Python
- Ruby

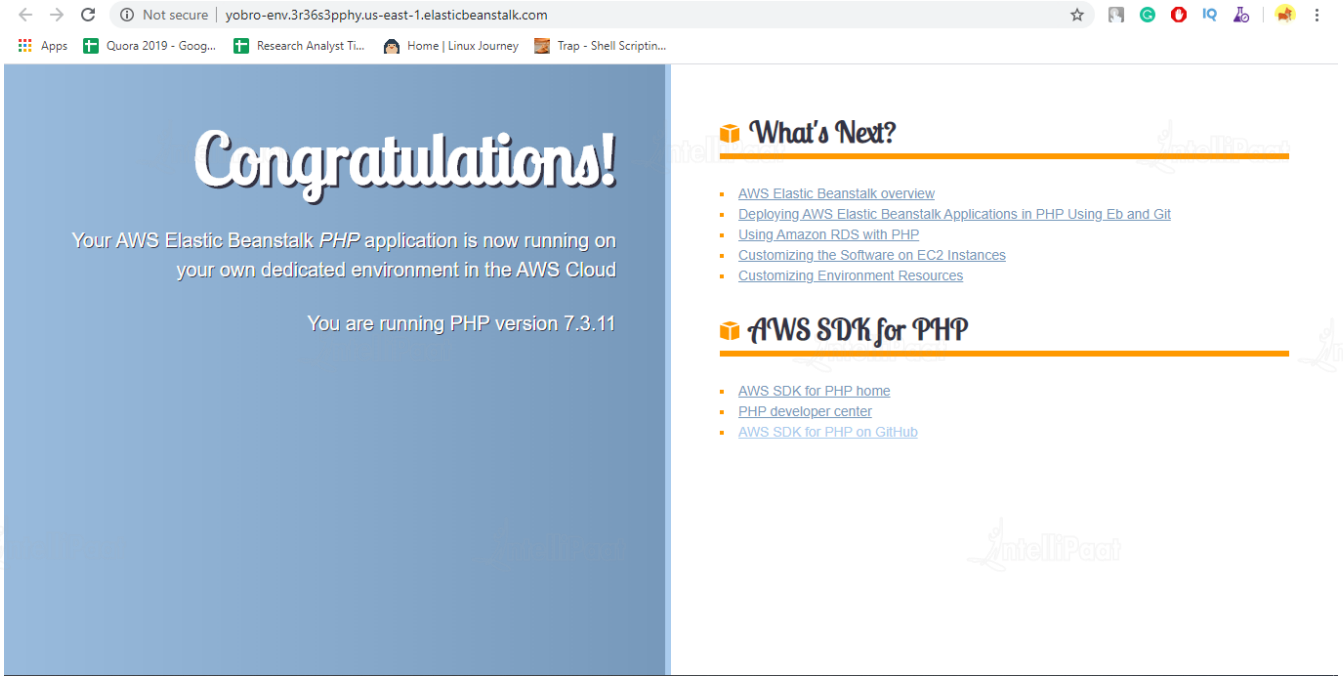
Demo: Launching a sample application

Demo: Launching a sample application

- Create a web app
- Give the application's name and choose the platform you want (such as PHP)
- Click on Configure more options
- Go ahead and explore the options; then proceed with the **Single instance (Free Tier eligible)** because you will not be billed
- Once it completes, click on the DNS name provided and view the sample PHP application which is running

Demo: Launching a sample application

Output of the sample application



Elastic Beanstalk Pricing

- There is no additional charge for using Elastic Beanstalk
- You pay for the underlying resources which the Beanstalk uses like EC2 servers, RDS databases and S3 storage.

What is Configuration Management?

What is Configuration Management?

- The main purpose of Configuration Management is to automate server management
- How do we manage? Let's assume you have 10 similar web servers running and you want to run a configuration update on all of them. To open each server and install that software is a time-consuming process. Instead, you can use create a script and run it on each server automatically to roll updates on all servers at the same time.

What is AWS OpsWorks?

What is AWS OpsWorks?

OpsWorks is a configuration management service which lets you configure and operate applications on the cloud. OpsWorks gives you the benefit of choosing between Chef or Puppet.

AWS OpsWorks Benefits

Support any application

Automation to run at scale

Configuration as Code

Resource Organization

CloudFormation vs OpsWorks

CloudFormation vs OpsWorks

CloudFormation	OpsWorks
Provides business a way of creating a collection of AWS resources in a chronological order	Provides businesses managed instances of Chef & Puppet to automate server management on your EC2 instances or on-premises instances
CloudFormation concentrates mainly on what and how AWS resources are created	OpsWorks mainly works on the orchestration of Servers and managing the software configuration rather than looking into the architecture
Use CloudFormation when you must create multiple architectures of the same design	Use OpsWorks when you must manage the configuration of multiple servers which server a similar purpose
Example: Replication of an available architecture	Example: Installing a web server in 100 instances and running a web page inside them

AWS OpsWorks Services

AWS OpsWorks Services

AWS OpsWorks for Puppet Enterprise

AWS OpsWorks for Chef Automate

AWS OpsWorks Stacks

OpsWorks for Puppet Enterprise lets you create AWS-managed Puppet master servers. A Puppet master server manages nodes in your infrastructure, stores facts about those nodes, and serves as a central repository for your Puppet modules.

AWS OpsWorks Services

AWS OpsWorks for Puppet Enterprise

AWS OpsWorks for Chef Automate

AWS OpsWorks Stacks

AWS OpsWorks for Chef Automate lets you create AWS-managed Chef server. A Chef server manages nodes in your environment, stores information about those nodes, and serves as a central repository for your Chef cookbooks.

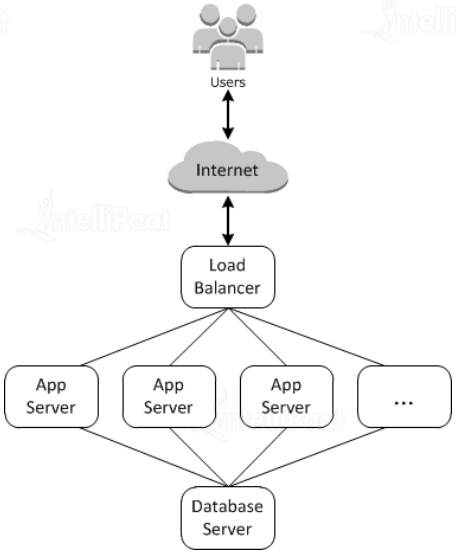
AWS OpsWorks Services

AWS OpsWorks for Puppet Enterprise

AWS OpsWorks for Chef Automate

AWS OpsWorks Stacks

AWS OpsWorks Stacks, the original service, provides a simple and flexible way to create and manage stacks and applications. For example, a web application requires servers, RDS instances, load balancers and storage. A combo of all these resources can be called as a stack.

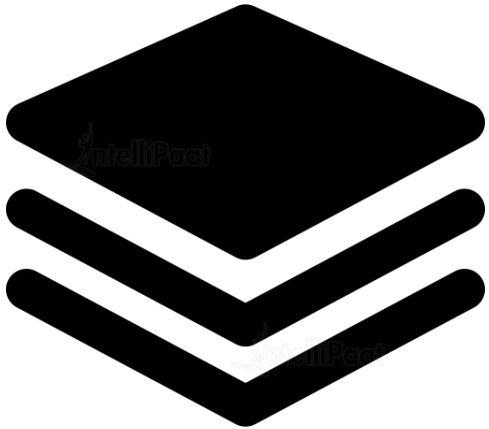


AWS OpsWorks Stacks

AWS OpsWorks Stacks

- **Stacks**
- **Layers**
- **Instances**
- **Apps**
- **Customizing your Stack**

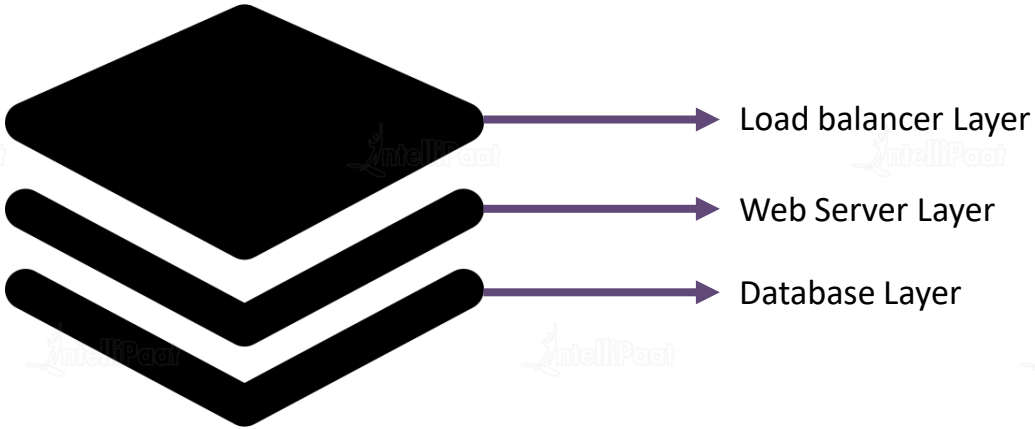
Top-level entity of in AWS OpsWorks stacks. It represents a collection of instances which are used for the same purpose, such as nginx or apache2 on them.



AWS OpsWorks Stacks

- Stacks
- Layers
- Instances
- Apps
- Customizing your Stack

Each stack will have a layer. Each layer describes a component. Like the load balancer layer or database layer.



AWS OpsWorks Stacks

- Stacks
- Layers
- Instances
- Apps
- Customizing your Stack

Instance is a computing resource like an EC2 instance. It helps in hosting applications, serving a database or load balancing. You can use Linux distributions or Windows Server 2012 R2.

- Amazon Linux 2
- Amazon Linux 2018.03
- Amazon Linux 2017.09
- Amazon Linux 2017.03
- Amazon Linux 2016.09
- Amazon Linux 2016.03
- CentOS Linux 7
- Red Hat Enterprise Linux 7
- Ubuntu 18.04 LTS
- Ubuntu 16.04 LTS
- Ubuntu 14.04 LTS
- Use custom Linux AMI

- Microsoft Windows Server 2012 R2 Base
- Microsoft Windows Server 2012 R2 with SQL Server Express
- Microsoft Windows Server 2012 R2 with SQL Server Standard
- Microsoft Windows Server 2012 R2 with SQL Server Web
- Use custom Windows AMI

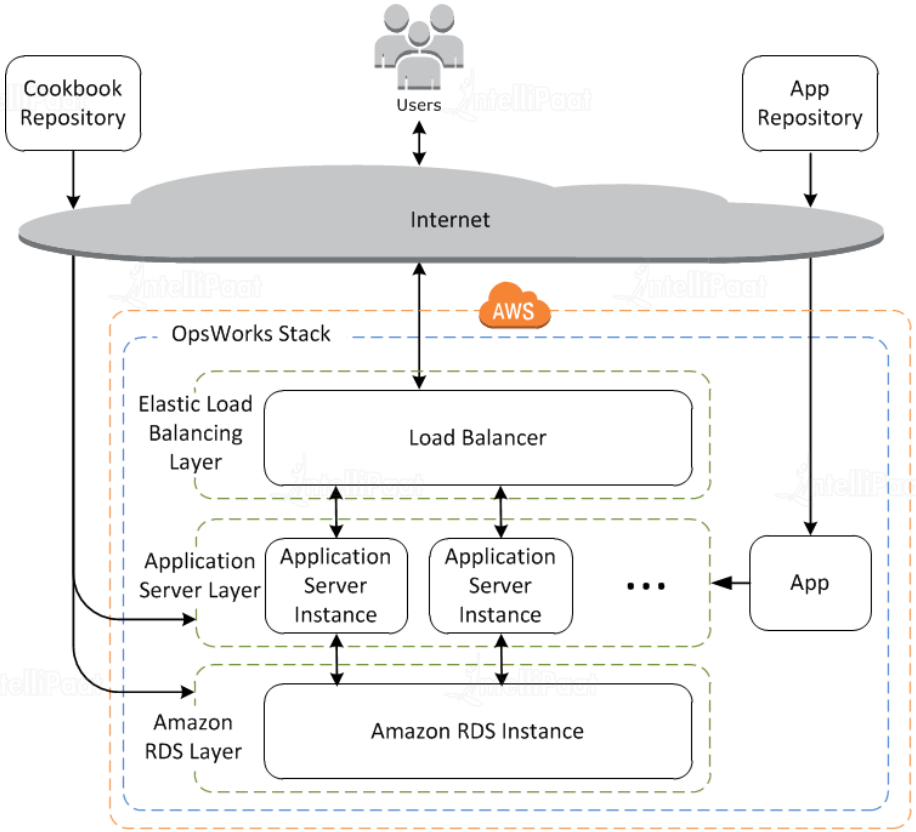
AWS OpsWorks Stacks

- Stacks
- Layers
- Instances
- Apps
- Customizing your Stack

OpsWorks app option represents the code that you want to run on the server. The code can reside in S3 or a repository like GitHub. App has the information required to deploy the code to app servers.

AWS OpsWorks Stacks

- Stacks
- Layers
- Instances
- Apps
- Customizing your Stack



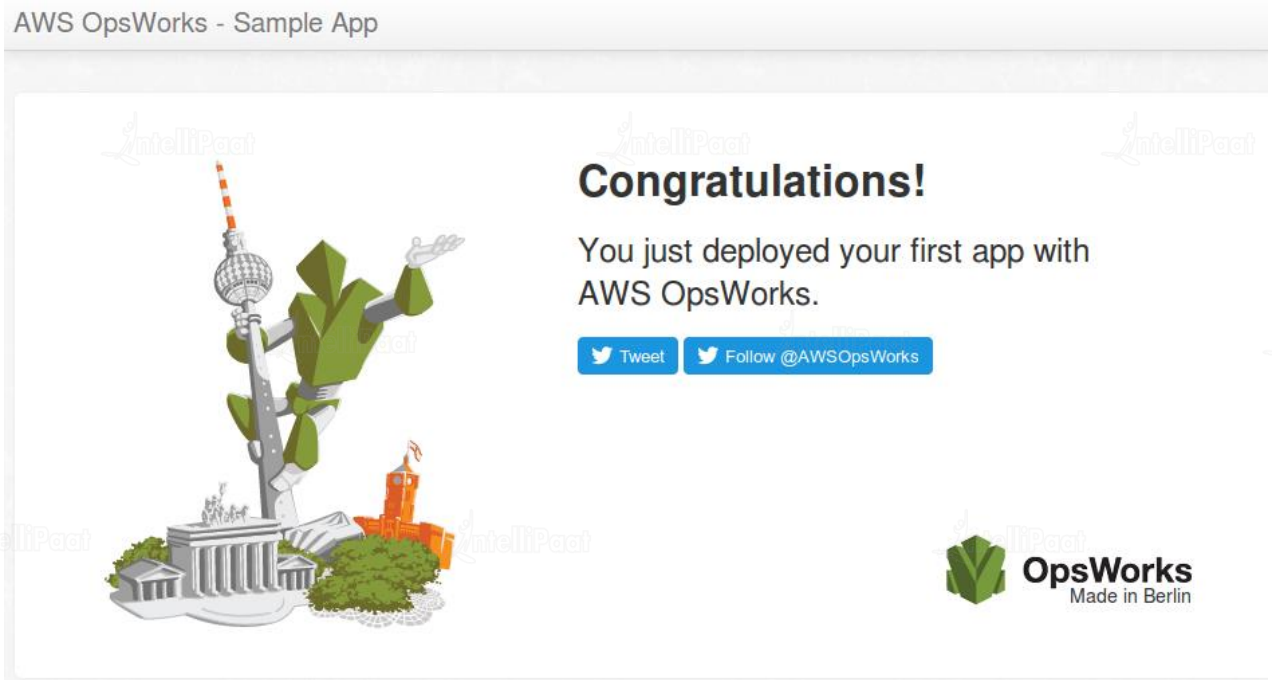
Demo: Launching a sample stack

Demo: Launching a sample stack

- Create a stack by clicking on the sample stack option
- Go to instances and before clicking on start, edit the instance's configuration by changing instance type from t2.medium to t2.micro so that you don't get charged for it.
- Then click Start and explore the other options in the stack. Such as apps.
- Once the instance is running, go to the public IP and find the website successfully hosted
- Delete the stack after deleting the instances

Demo: Launching a sample stack

Output of the sample application



AWS OpsWorks Pricing

- There is no additional charge for using OpsWorks Stacks
- You pay for the underlying resources which the OpsWorks uses like EC2 servers, RDS databases.
- The pricing for each on-premises server on which you install the OpsWorks Stacks agent is $0.02 per hour

Quiz

Quiz

1. What is the maximum time out period of a Lambda function?

A. 10 seconds

B. 10 minutes

C. 15 minutes

D. 200 seconds

Quiz

2. What cloud service model does Elastic Beanstalk come under?

A. IaaS

B. PaaS

C. SaaS

D. FaaS

Quiz

3. Elastic Beanstalk uses EC2 for hosting the application.

A. Yes

B. No

Quiz

4. OpsWorks runs both Chef and Puppet on the same server.

A. Yes

B. No

Quiz

5. What type of a computing platform is Lambda?

A. Serverfull

B. Serverless

C. Code as an Infrastructure

D. Database

India : +91-7847955955

US : 1-800-216-8930 (TOLL FREE)

support@intellipaat.com

24X7 Chat with our Course Advisor