DevOps with AWS - Master Program (13 - DEVOPS)

Pre-Requisites to join DevOps

-> No Pre-Requisites to learn **DevOps with AWS Cloud**

- Your time (4 months and daily 5 hours)
- Your Effort & Dedication is required

Note: We will teach from zero to hero level

Who can join this course?

- IT Employees (Developers / Testers / Support Engineers)
- Non-IT People
- People who have some career gap also can join

Course Content

Module-1: Software Industry Details

Module-2 : DevOps world (What, Why, Lifecycle, Roles & Responsibilities)

Module-3 : Linux & Shell Scripting

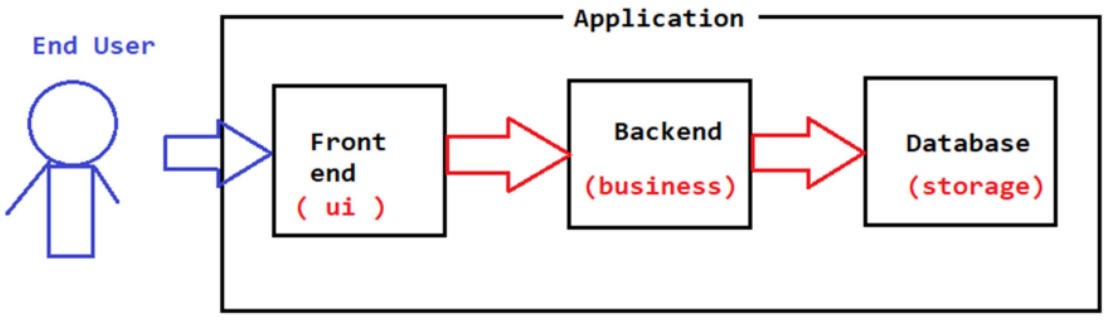
Module-4 : AWS Cloud (10+ Services)

Module-5 : DevOps Tools (12+ Tools)

Module-6 : Real-World Projects

Module-7 : Interview Guide (Resume Prep, FAQs, Mock Interview, Tips & Tricks)

Software Application Architecture



What is Database?

- ⇨ Database is used to store the data
- ⇨ For every software application will use one or more Databases for storing application data

Ex: Oracle, MySQL, PostgreSQL, SQL Server, MongoDB etc.

What is Backend?

- ⇨ Backend contains business logics of the application

Ex:

1) Verify User login credentials
2) User Registration
3) Sending Emails
4) Sending OTPs
5) Calculations etc..

- ⇨ **To develop Backend of the application we have below technologies**
  1) Java
  2) Dot Net
  3) Python
  4) PHP
  5) Node JS etc..

What is Frontend?

- ⇨ Frontend contains User interface (presentation logic)
- ⇨ End users will communicate with application using Frontend only

- ⇨ **To develop Frontend of the application we have below technologies**
  1) HTML
  2) CSS
  3) Java Script
  4) Boot Strap
  5) Angular
  6) React JS

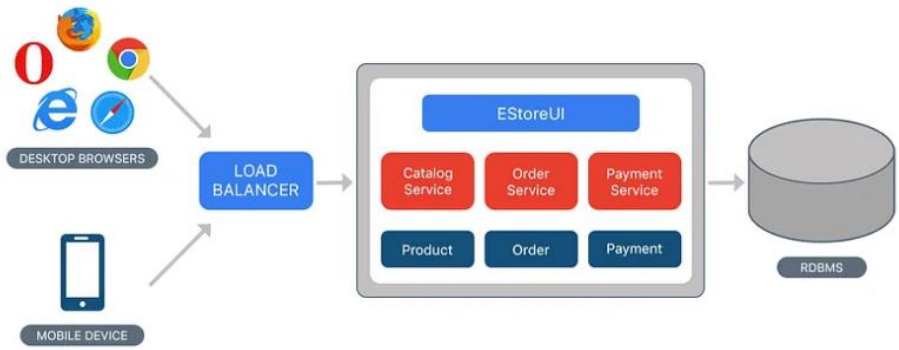
Software Application Architecture Styles

- ⇨ Now days we can see 2 types of Architectures

1) Monolith Architecture
2) Microservices Architecture

What is Monolith Architecture?

- Monolith means composed all in one piece.
- The Monolithic application describes a single-tiered software application in which different components combined into a single program from a single platform.
- Despite having different components/modules/services, the application is built and deployed as one Application for all platforms (i.e. desktop, mobile and tablet).



Monolithic Architecture (for E-Commerce Application)

Monolith Benefits:

- **Simple to develop** — At the beginning of a project it is much easier to go with Monolithic Architecture.
- **Simple to test**. For example, you can implement end-to-end testing by simply launching the application and testing the UI with Selenium.
- **Simple to deploy.** You have to copy the packaged application to a server.
- Simple to scale horizontally by running multiple copies behind a load balancer.

Monolith Drawbacks

- Maintenance — If Application is too large and complex to understand entirely, it is challenging to make changes fast and correctly.
- The size of the application can slow down the start-up time.
- You must redeploy the entire application on each update.
- Monolithic applications can also be challenging to scale when different modules have conflicting resource requirements.
- Reliability — Bug in any module (e.g., memory leak) can potentially bring down the entire process. Moreover, since all instances of the application are identical, that bug impact the availability of the entire application

What is Microservices Architecture?

- Microservices are an approach to application development in which a large application is built as a suite of modular services (i.e. loosely coupled modules/components).
- Each module supports a specific business goal and uses a simple, well-defined interface to communicate with other sets of services.
- Instead of sharing a single database as in Monolithic application, each microservice has its own database.

- Having a database per service is essential if you want to benefit from microservices, because it ensures loose coupling.

Microservices Architecture (for E-Commerce Application)

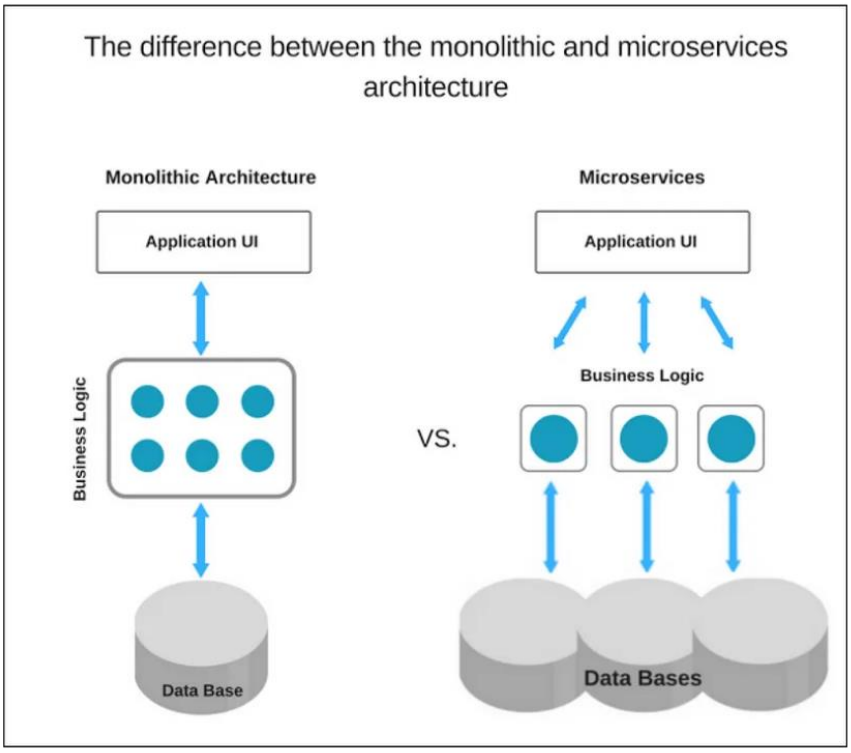
Microservices Architecture Benefits

- Microservices Enables the continuous delivery and deployment of large, complex applications.
- Better testability — services are smaller and faster to test.
- Better deployability — services can be deployed independently.
- It enables you to organize the development effort around multiple teams. Each team is responsible for one or more single service. Each team can develop, deploy and scale their services independently of all of the other teams.
- Each microservice is relatively small
- Comfortable for a developer to understand
- The IDE is faster making developers more productive
- The application starts faster, which makes developers more productive, and speeds up deployments
- Improved fault isolation. For example, if there is a memory leak in one service then only that service is affected. The other services continue to handle requests. In comparison, one misbehaving component of a monolithic architecture can bring down the entire system.
- Microservices Eliminates any long-term commitment to a technology stack. When developing a new service, you can pick a new technology stack. Similarly, when making major changes to an existing service you can rewrite it using a new technology stack.

Challenges with Microservices Architecture

- Developers must deal with the additional complexity of creating a distributed system.
- Developer tools/IDEs are oriented on building monolithic applications and don't provide explicit support for developing distributed applications.

- Testing is more difficult as compared to Monolith applications.
- Developers must implement the inter-service communication mechanism.
- Implementing use cases that span multiple services without using distributed transactions is difficult.
- Implementing use cases that span multiple services requires careful coordination between the teams.
- Deployment complexity. In production, there is also the operational complexity of deploying and managing a system comprised of many different service types.



Difference b/w Monolith and MicroService Architectures

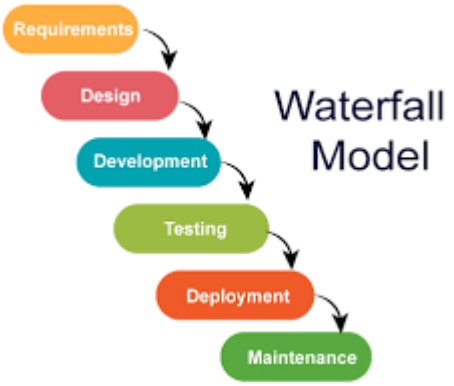
Software Development Life Cycle (SDLC)

=> SDLC Represents end to end application development process

=> We will have several phases in SDLC like below

1) **Requirements Gathering ->** (Functional Team / SME) -> FDD / SRS
2) **Analysis ->** Understand requirements & ask questions
3) **Planning ->** Prepare plan & prototype
4) **Development ->** Coding
5) **Testing ->** Verification and validation
6) **Deployment ->** Running application in the server
7) **Delivery ->** Handover to client (DevOps)
8) **Maintenance ->** Based on SLA

Waterfall Methodology



-> Earlier people used to follow Waterfall Methodology to develop projects

-> Waterfall is a linear methodology to develop and deliver projects

-> Everything will happen step by step

-> If one step completed then only we will go to next step

-> We will move only in forward direction (No backward direction)

-> Requirements are fixed

-> Budget is fixed

-> Client involvement is very less

-> Client will see the project at the end

Note: Waterfall Methodology is not suitable for big projects

Agile Methodology

-> Agile is an iterative approach to develop and deliver the projects

-> Development and testing will happen parallelly

-> Client involvement will be very high

-> We will deliver project in multiple releases (Sprints)

-> For every release we will take client feedback

-> Requirements are not fixed

-> Budget is not fixed

-> Project Development, Testing & Delivery is very frequent is Agile

What is DevOps?

DevOps = Development + Operations

=> DevOps is a culture

=> DevOps is a process

=> DevOps is set of practises

=> DevOps culture is used to collaborate Development and Operations in Software Project

=> Using DevOps culture, we can simplify software project delivery process to clients

=> DevOps is used throughout software development life cycle process

DevOps Advantages

- Speed
- Rapid Development
- Quick Releases
- Reliability
- Security
- Client Satisfaction
- Teams Collaboration

DevOps Tools Overview

1) Terraform: To provision / creating infrastructure in Cloud

Ex: Machines, Network, Database, Servers, Storage etc...

2) Ansible: To perform Configuration Management.

Ex: install java s/w, copy files from one machine to another machine etc...

3) Git Hub / Bitbucket: Source Code Repository Servers

4) Maven / Gradle / NPM: Code Build Tools (convert code to executable format)

5) SonarQube: Code Review Software (To identify developer mistakes)

6) Nexus / JFrog: Artifactory Repository Servers (To maintain shared libraries)

7) Tomcat / IIS: It is a web server to run web applications.

8) Jenkins: It is used for CI & CD (To automate build & deployment)

9) Docker: Used for Containerization (Containers creation)

10) Kubernetes: Used for Orchestration (To manage container)

11) Grafana & Prometheus: For Monitoring Purpose

12) EFK (Elastic Search + File Beat + Kibana): For application logs monitoring

Development Team (Programmers) Roles

- Collect Requirements
- Analyse Requirements
- Planning (Designing)
- Implementation (Coding)
- Unit Testing
- Code Integration
- Integration Testing
- Bug Fixing

DevOps Engineer Roles & Responsibilities

- Setup machines
- Setup Network
- Setup Servers to run application
- Setup Database
- Prepare Infrastructure for Project execution
- Take project code from Repository (GitHub)
- Perform Project Build (Compile)
- Perform Code Review Report
- Package Project (executable artifact)
- Deploy Project to Server
- Deliver Project to Client
- Monitor Infrastructure & health checks
- Monitor application & health checks

What is Infrastructure?

- ⇨ The resource which are required to run our business is called as Infrastructure
- ⇨ To run a school, we need below infrastructure

-> Board
-> Tables
-> Chairs
-> Markers
-> Duster etc..

- ⇨ To run hotel, we need below infrastructure

1) Building 2) Power 3) Air Conditioner 4) Tables 5) Chairs 6) Plates 7) Spoons

- ⇨ To run a software company, we need below infrastructure

-> Machines
-> Web Servers
-> Application Servers
-> Database Servers
-> Load Balancer
-> Network
-> Power
-> Storage
-> Security
-> Monitoring

-> Some companies will purchase and setup infrastructure that is called on On-Premises Infrastructure.

-> If we setup on-prem infrastructure then we have to purchase, setup & manage everything required for project.

1) We have to purchase computers
2) We have to purchase Web servers
3) We have to purchase Database Servers
4) We have to purchase Network
5) We have to purchase Storage
6) We have to purchase Power & Power Backup
7) We need to setup Server room
8) We need to setup Air Conditioner (AC)
9) We need to hire Network Admin to setup network
10) We need to hire Server admin to setup servers
11) We need to hire DB admin to setup Database
12) We need to take a room for rent
13) We need a security guard to monitor our server room
14) We need to keep high security for our server room

-> Below are the challenges if we maintain our own infrastructure

- Security
- Privacy
- Scalability
- Availability
- Network Issues
- Disaster Recovery

-> To overcome above issues, today companies are using Cloud Infrastructure

What is Cloud Computing?

-> Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing.

-> Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider.

Note: IT Resource means, computing, network, power, security, storage, servers etc...

Advantages with Cloud Computing?

- Low Cost
- Pay As You Go
- Scalability
- Availability
- Reliability
- Security
- Unlimited Storage
- Backup

Cloud Providers

=> The company which is providing cloud services is called as Cloud Provider.

1) Amazon - AWS cloud
2) Microsoft - Azure cloud
3) Google - GCP cloud
4) Salesforce cloud
5) Alibaba
6) Oracle Cloud
7) IBM cloud etc.....

Cloud Service Models

1) IAAS : Infrastructure as a service

=> Cloud Provider will give only Infrastructure. We need to manage runtime + application

2) PAAS : Platform as a service

=> Cloud Provider will give infrastructure + runtime. We need to deploy our application in their runtime

3) SAAS : Software as a service

=> Cloud Provider will give infrastructure + runtime + application. We need to use their application to run our business.