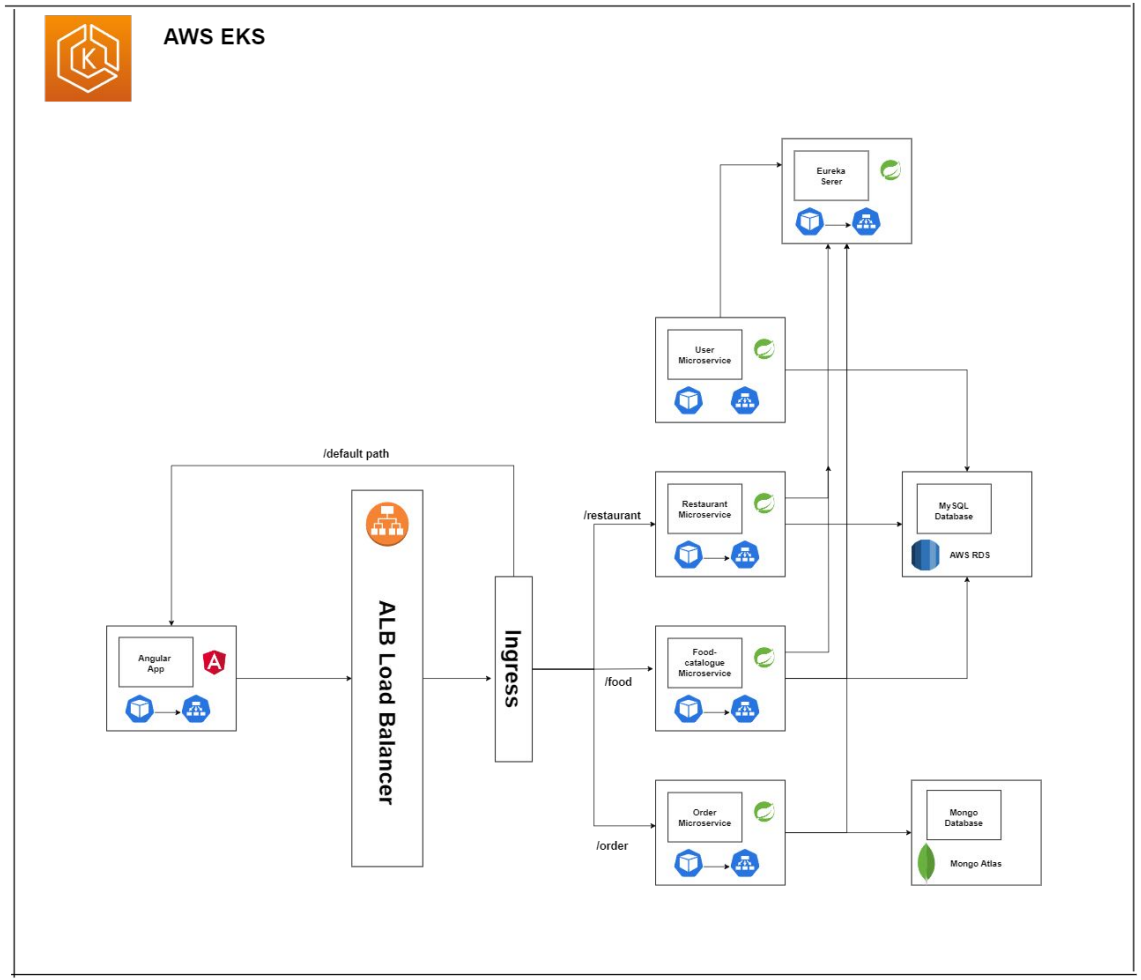
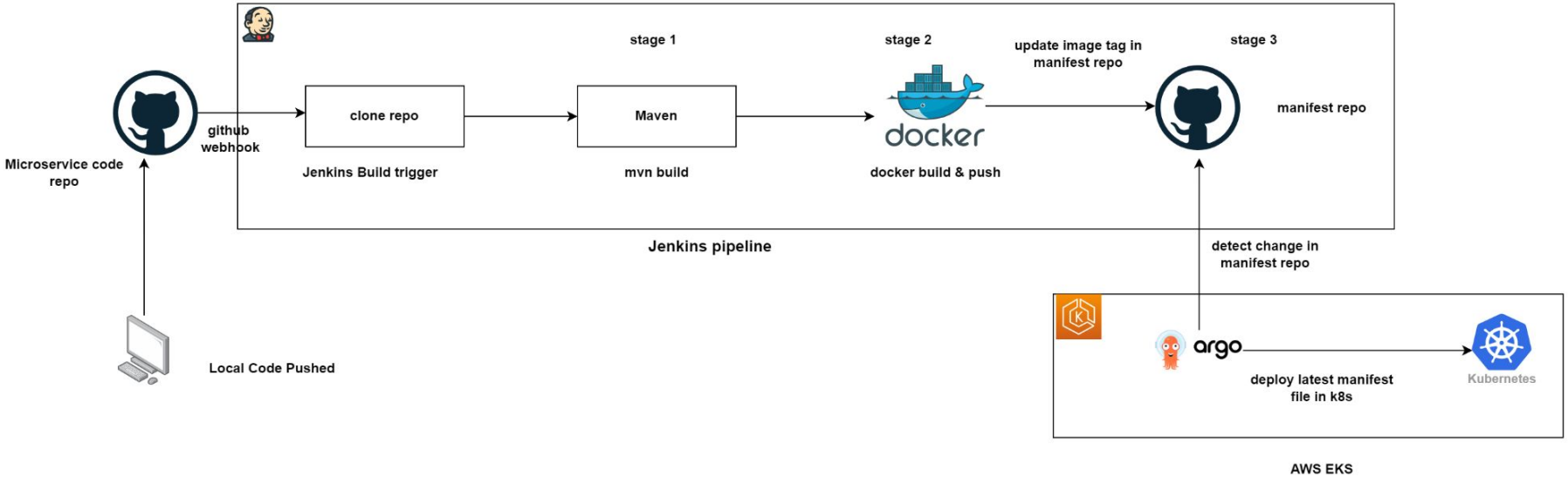
Introduction



Application Architecture Diagram


stage 1
stage 2
update image tag in manifest repo
stage 3
clone repo
Maven
docker
manifest repo
github webhook
Microservice code repo
Jenkins Build trigger
mvn build
docker build & push
Jenkins pipeline
detect change in manifest repo
argo
Kubernetes
deploy latest manifest file in k8s
Local Code Pushed
AWS EKS

Technologies used

- Rest/ Restful
- Spring boot
- Microservices
- Angular
- Github
- Junit
- Sonar
- Docker
- Kubernetes
- AWS EKS
- AWS ALB
- AWS EC2
- AWS RDS (SQL)
- Mongo Atlas (No-SQL)
- Jenkins
- ArgoCD

Spring boot basics

What is Spring boot

- Spring boot is a Framework for RAD build using Spring framework with extra support of auto-configuration and embedded application server (like tomcat, jetty).
- It provides RAD – Rapid application development.
- It helps us in creating efficient fast stand-alone applications which you can just run it basically removes a lot of configurations and dependencies

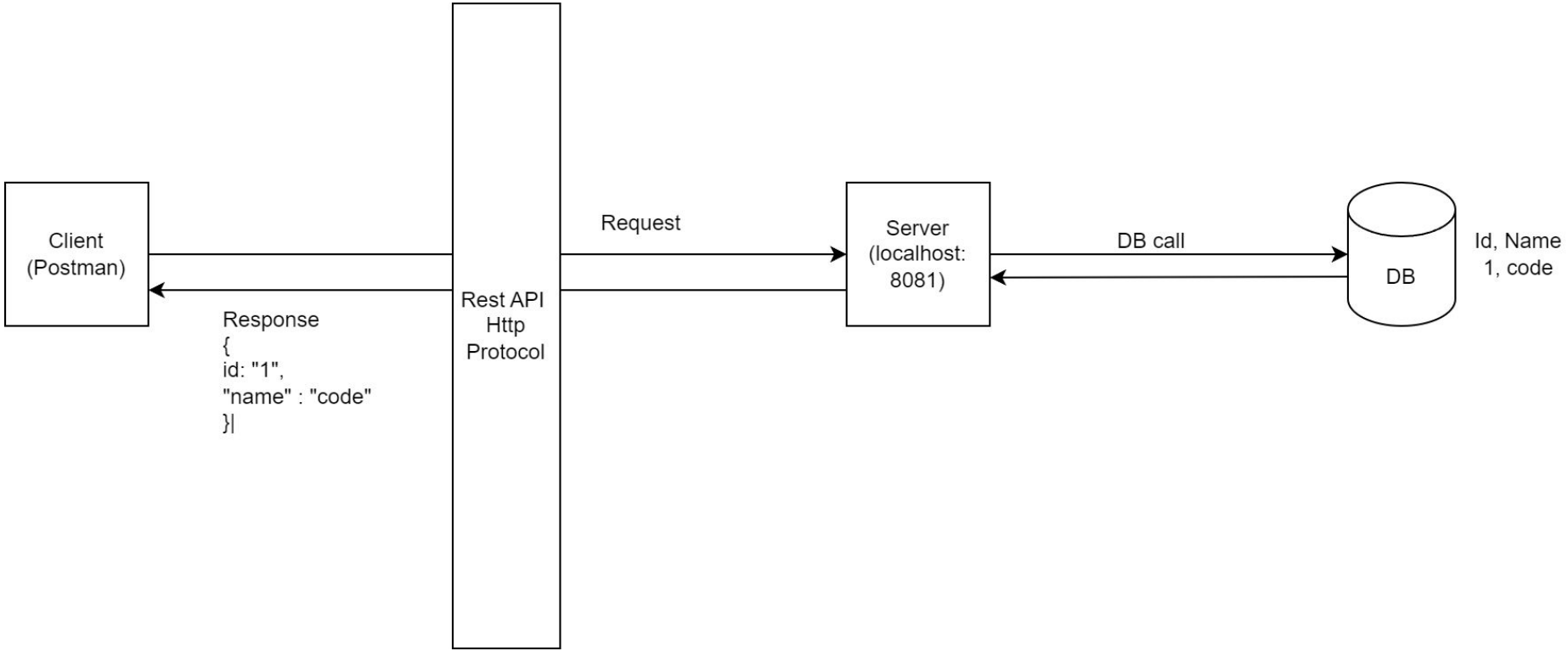
What is RAD

- Business Modeling
    - Business model is designed for the product to be developed
- Data Modeling
    - Data model is designed, The relation between these data objects are established using info gathered in first phase
- Process Modeling
    - Process model is designed. Process descriptions for adding, deleting, retrieving or modifying a data object are given
- Application Generation
    - The actual Product is built using coding. Convert process and data models into actual prototypes.
- Testing and Turnover
    - Product is tested and if changes are required then whole process starts again.

What is Rest and Restful

What is REST and RESTFUL

- REST represents REpresentational State Transfer
- RESTFUL web services are web services that follows REST architectural concept (stateless client-server architecture).
- Its is an architectural style for developing applications that can be accessed over the network


Client
(Postman)
Response
{
id: "1",
"name" : "code"
}|
Rest API
Http
Protocol
Request
Server
(localhost:
8081)
DB call
DB
Id, Name
1, code

What is a REST Resource?

- Every content in the REST architecture is considered a resource.

- The resource == object in the object-oriented programming world.
- Can be represented as text files, HTML pages, images, or any other dynamic data.

- The REST Server provides access to these resources whereas the REST client consumes (accesses and modifies) these resources. Every resource is identified globally by means of a URI

What is URI?

- Uniform Resource Identifier is the full form of URI which is used for identifying each resource of the REST architecture. URI is of the format:

- <protocol>://<service-name>/<ResourceType>/<ResourceID>

- https://en.wikipedia.org/wiki/Representational_state_transfer

- They are of 2 types:
    - URN
    - URL

What are the HTTP Methods?

- HTTP Methods are also known as HTTP Verbs. They form a major portion of uniform interface restriction followed by the REST that specifies what action has to be followed to get the requested resource. Below are some examples of HTTP Methods:

- GET: This is used for fetching details from the server and is basically a read-only operation.

- POST: This method is used for the creation of new resources on the server.

- PUT: This method is used to update the old/existing resource on the server or to replace the resource.

- DELETE: This method is used to delete the resource on the server.

What are the HTTP Methods?

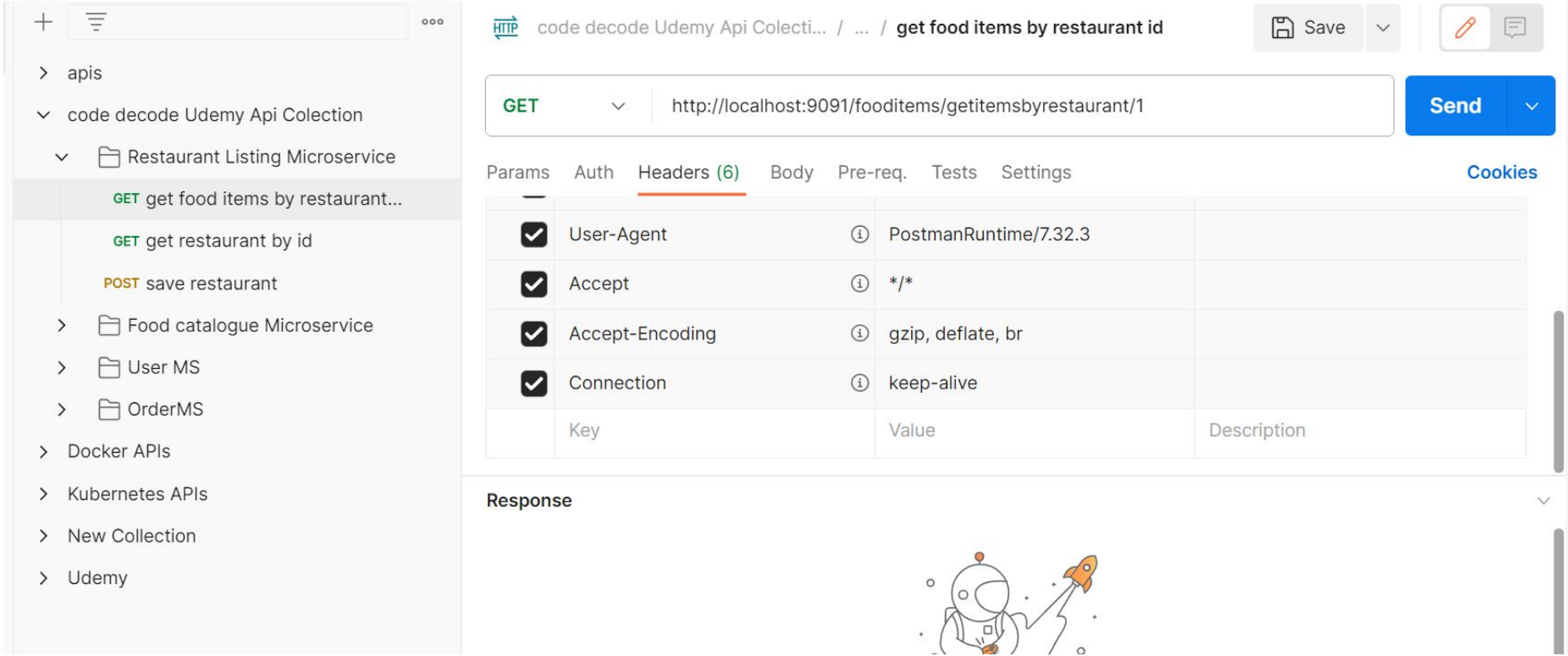
C ——— Post
R ——— Get
U ——— Put
D ——— Delete

What are the features of RESTful Web Services?

- The service is based on the Client-Server model.
- The service uses HTTP Protocol for fetching data/resources, query execution, or any other functions.
- The medium of communication between the client and server is called “Messaging”.
- Resources are accessible to the service by means of URIs.
- It follows the statelessness concept where the client request and response are not dependent on others and thereby provides total assurance of getting the required data.
- These services also use the concept of caching to minimize the server calls for the same type of repeated requests.

What is the concept of statelessness in REST?

As per REST architecture, a RESTful web service should not keep a client state on server. This restriction is called statelessness. It is responsibility of the client to pass its context to server and then server can store this context to process client's further request. For example, session maintained by server is identified by session identifier passed by the client.

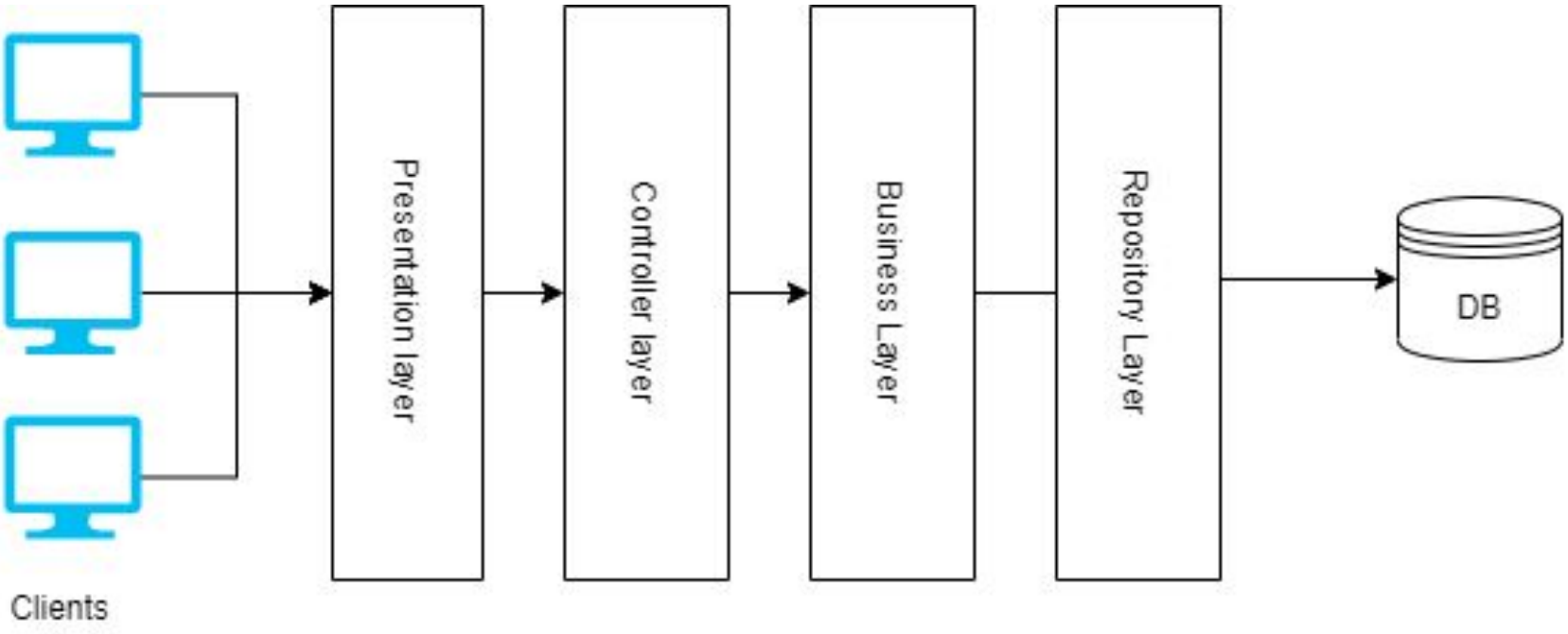

apis
code decode Udemy Api Colection
Restaurant Listing Microservice
GET get food items by restaurant...
GET get restaurant by id
POST save restaurant
Food catalogue Microservice
User MS
OrderMS
Docker APIs
Kubernetes APIs
New Collection
Udemy
code decode Udemy Api Colecti... / ... / get food items by restaurant id
Save
GET
http://localhost:9091/fooditems/getitemsbyrestaurant/1
Send
Params
Auth
Headers (6)
Body
Pre-req.
Tests
Settings
Cookies
User-Agent
PostmanRuntime/7.32.3
Accept
/
Accept-Encoding
gzip, deflate, br
Connection
keep-alive
Key
Value
Description
Response

What are Monolithic Application

What are Monolithic Applications ?

- In Monolithic Architecture the application is build as a single unit.
- Such Applications comprises of client side interface, server-side application and a database.
- Normally a monolithic application have one large code base and it lack modularity

Monolithic Application Architecture



Disadvantages of Monolithic Application

- The code base get larger in size with time and hence it’s very difficult to manage.
- It is very difficult to introduce new technology as it affects the whole application.
- A single bug in any module can bring down the whole application
- It is very difficult to scale a single module. One has to scale the whole application
- Continuous deployment is extremely difficult. Large monolithic applications are actually an obstacle to frequent deployments. In order to update one component, we have to redeploy the entire application.

What are Microservices

What are Microservices ?

- While Monolith Application works a single component, a Microservice Architecture breaks it down to independent standalone small applications, each serving one particular requirement.

- Example, we can divide the whole monolithic application of food delivery app in to following microservices.
    - Restaurant Listing service
    - Food catalogue service
    - Order service
    - User service

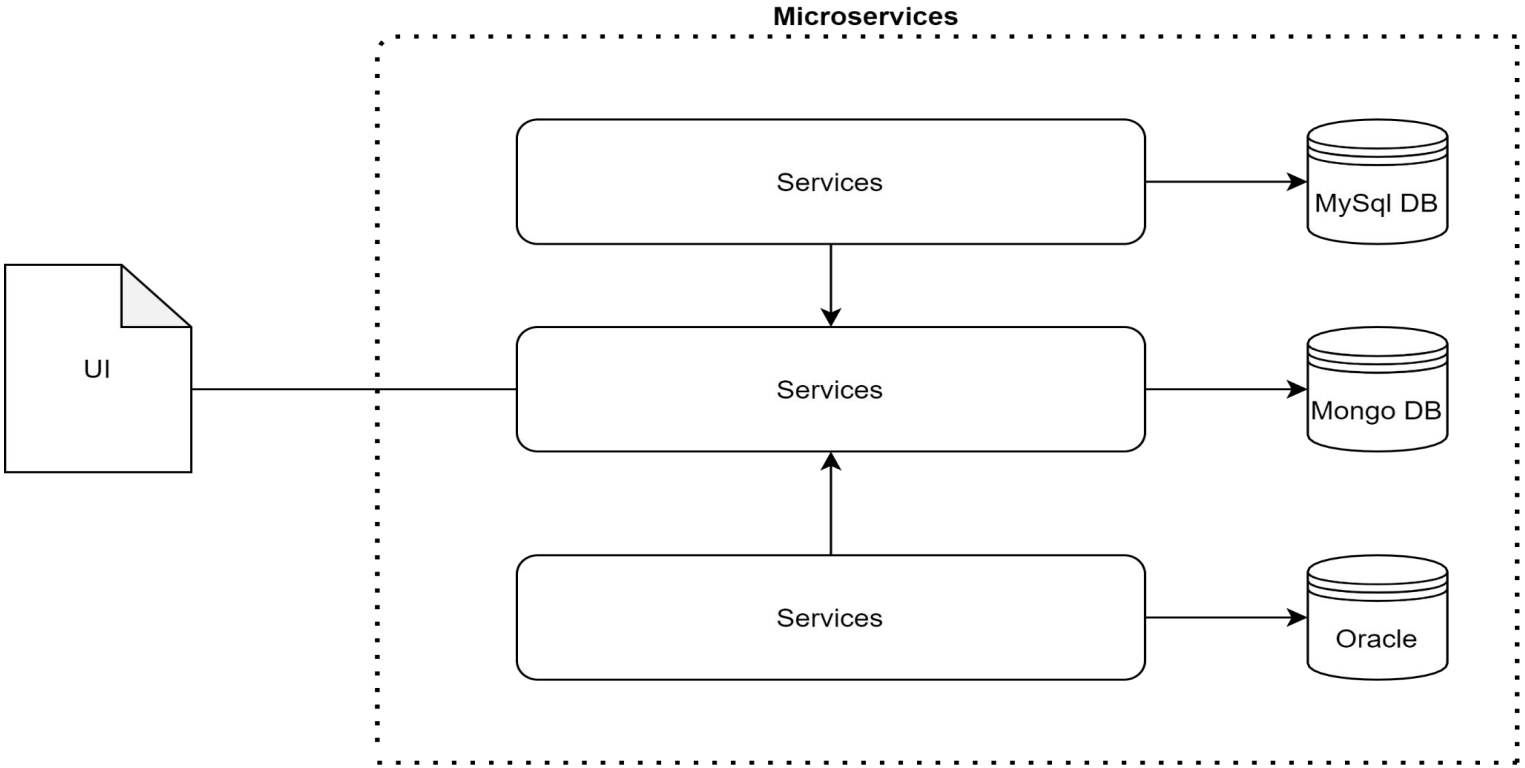
- Within this microservice architecture, the entire functionality is split in independent deployable module which communicate with each other through API’s(RESTful web services)

Advantages of Microservices

Advantages of Microservices

- All the services are independent of each other. Therefore testing and deployment is easy as compare to Monolith application
- If there is bug in one microservice it has an impact only on a particular service and does not affect the entire application. Even if ur Order service is down, you can still see the list of restaurant using restaurant listing service
- With microservice architecture, it's easy to build complex applications.
- It will give flexibility to choose technologies and framework for each microservices independently

Microservice Architecture



Why Eureka Server

For our Microservices

Why Eureka?

Registry

Microservices can register themselves with Eureka Server, providing information like (**IP address, port, health status** etc. This allows Eureka Server to maintain an up-to-date **list of available services** in the system.

Discovery

Clients (other microservices) can **query** Eureka Server to dynamically discover and **locate the services** they need. Eureka enables seamless **communication** between microservices.

Fault tolerance/resiliency

Eureka Server continuously **monitors the health** of registered services. Eureka Server automatically **removes unhealthy services** from the registry

Client side Or server side Discovery will we use?

In microservices architecture, there are two approaches to service discovery: client-side discovery and server-side discovery.

- **Client-side discovery**: In client-side discovery, the **responsibility of service discovery lies with the client**. Each client is aware of the service registry and is responsible for locating and communicating with the desired services

- **Server-side discovery**: In server-side discovery, the responsibility of service discovery is shifted to a dedicated component called a service registry or discovery server. The service registry acts as a centralized repository where services register themselves when they start up. Client applications make requests to the service registry to obtain the network locations of the desired services. The registry performs load balancing and routing based on its internal algorithms and returns the appropriate service instance to the client.

Setting up the Eureka Server

A server side registry / discovery for our Microservices

Steps

- Add eureka server Dependency in pom.xml
- Annotate with @EnableEurekaServer
- Configure in property file -
    - Server port
    - Fetch registry and register with eureka if we have multiple eureka servers on network

Setting up the Restaurant Listing

Microservices

Tech Stack for Restaurant Listing Microservice

- Restaurant listing Microservice is responsible to List all Restaurants on the front end Page / UI

- Tech stack used

    - Microservice architecture
    - Rest APIs
    - Java 11
    - MySql Relational DB as Datasource
    - Spring Boot
    - Lombok
    - Eureka Client
    - mapstruct

Steps to Configure

- Create Spring Boot application from https://start.spring.io/
- Add starter dependencies for web, eureka, lombok, mysql connector, Data jpa
- Configure Application.yml properties
- Create Web MVC Architecture -
  - Controller layer,
  - Service Layer
  - Repo Layer
  - Entity
  - DTO layer

Setting up the Food Catalogue Listing

Microservice

Tech Stack for Food Catalogue Microservice

- On Front end page where all restaurants are listed , when we select 1 restaurant, Ll its food menu is shown. Also restaurant details , address etc too is shown.

- Food Catalogue Microservice is responsible to-
  - List **all food items list of that particular Restaurant** and
  - **Complete Restaurant Details** too on the front end Page / UI

- Tech stack used

  - Microservice architecture
  - Rest APIs
  - Java 11
  - MySql Relational DB as Datasource
  - Spring Boot
  - Lombok
  - Eureka Client
  - mapstruct

Steps to Configure

- Create SB application
- Add starter dependencies for web, eureka, lombok, mysql connector, Data jpa
- Configure Application.yml properties
- Create Web MVC Architecture -
  - Controller layer, - An API to **add Food list** for a restaurant and another API to **fetch Food Item list for a particular restaurant and restaurant details from Restaurant Listing Microservice** we created earlier.
  - Service Layer
  - Repo Layer
  - Entity
  - DTO layer

Steps to Configure

- Create load balanced Resttemplate Bean in main spring boot app file.

```
@Bean
	@LoadBalanced
	public RestTemplate getRestTemplate() {
		return new RestTemplate();
	}
```

To fetch restaurant use this ->

```
 Restaurant restaurant =
restTemplate.getForObject("http://RESTAURANT-SERVICE/restaurant/id/" + id,
Restaurant.class);
```

Setting up the User Details

Microservice

Tech Stack for User Service

- This Microservice handles user information like its id, name, password, address where to deliver the food etc.

- Tech stack used
    - Microservice architecture
    - Rest APIs
    - Java 11
    - MySql Relational DB as Datasource
    - Spring Boot
    - Lombok
    - Eureka Client
    - mapstruct

Setting up the Order Service

Microservice

Tech Stack for Order Service

- When we Hit **OrderNow** button, the order details and user id reached backed, This MS saves all order details like
  - Food items list
  - Restaurant details
  - User Information

- Tech stack used

  - Microservice architecture
  - Rest APIs
  - Java 11
  - Mongo DB as Datasource
  - Spring Boot
  - Lombok
  - Eureka Client

Tech Stack for Order Service

- findAndModify: This method is used to **find a document and modify it in a single atomic operation.**

- **query(where("_id").is("sequence")):** This specifies the query criteria. It uses the where method to define the condition and is to specify the value of the _id field. In this case, it searches for a document with the _id equal to "sequence".

- **new Update().inc("sequence", 1):** This part defines the update operation. The inc method increments the value of the "sequence" field by 1. It creates a new Update object and specifies the field to update and the increment value.

- **options().returnNew(true).upsert(true):** Here, the options for the findAndModify operation are defined. The returnNew(true) option indicates that the modified document should be returned as the result. The upsert(true) option specifies that if the document matching the query is not found, a new document should be created.

Tech Stack for Order Service

- Sequence.class: The last parameter specifies the class type of the document. In this case, it is the Sequence class.

- counter.getSeq(): Finally, the code retrieves the value of the "seq" field from the modified document and returns it.

- Overall, this code performs an atomic find-and-modify operation on a MongoDB collection. It searches for a document with the specified _id, increments the value of the "seq" field by 1, and returns the updated value. If the document doesn't exist, a new document is created with the specified _id and an initial "seq" value of 1.

Steps to Configure

- Create SB application

- Add starter dependencies for web, eureka, lombok, -starter-data-mongodb,
- Configure Application.yml properties

- Create Web MVC Architecture -
  - Controller layer, - An API to add order Details
  - Service Layer
  - Repo Layer
  - Entity
  - DTO layer

Setting up the Front End of

Food Delivery Application

Wireframe pages - Restaurant Listing page

Code Decode Food Delivery

Hey JohnDoe

The Hungry Hut

4.5 (500+)

Order Now

Spice Fusion

4.5 (500+)

Order Now

Seafood Paradise

4.5 (500+)

Order Now

Tandoori Delight

4.5 (500+)

Order Now

The Green Garden

4.5 (500+)

Order Now

Curry House

4.5 (500+)

Order Now

Saffron Sizzlers

4.5 (500+)

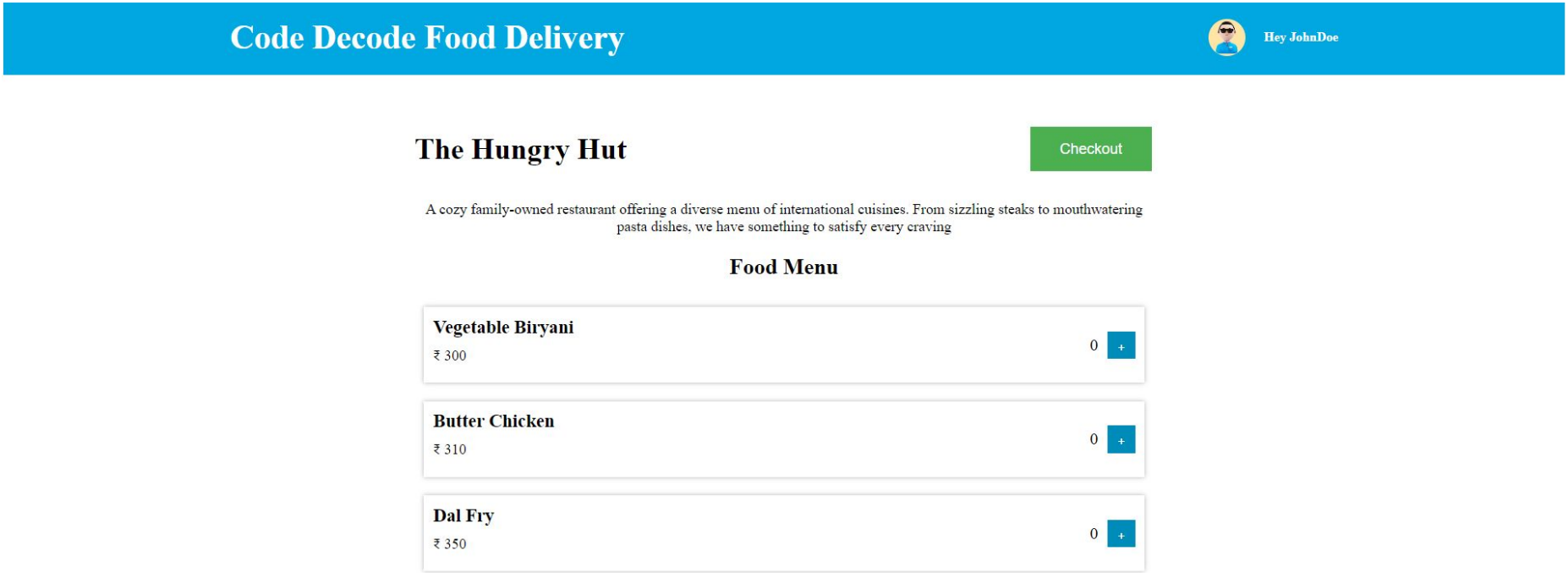
Order Now

Fireside Grill & Bar

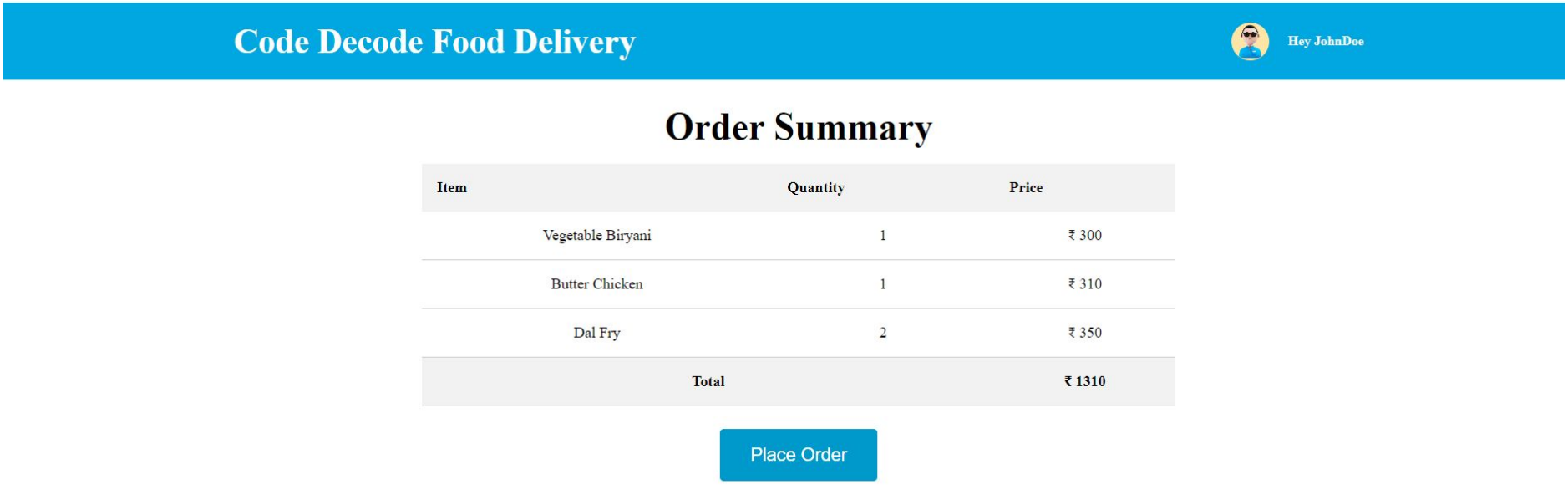
4.5 (500+)

Order Now

Wireframe pages - Food Catalogue page



Wireframe pages - Order Summary page



Tech Stack to be used and Why

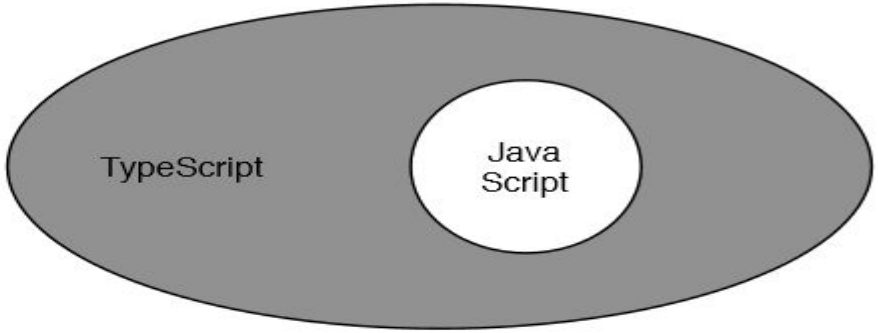
- We will be using Angular as a front end web application development framework (developed and maintained by Google).
- Angular creates dynamic, scalable, SPA - loads all necessary resources on initial page load and dynamically updates the content as the user interacts with the application, without requiring a full page refresh.
- Why Angular -
- Angular follows the **component-based architecture,** where the application is divided into reusable components that encapsulate the presentation logic and data. These components are composed together to create complex web applications.
- Very similar to backend Architecture of having Controller, service and repo layers segregated for better modularization
- Angular is written in TypeScript, a statically-typed superset of JavaScript

What is Angular and TypeScript

Why Typescript?

TypeScript is JavaScript!!!!!!
with few additional features.

1) Object Orientation
2) Strong type checking checking
3) compile-time error checks
4) maintainable and reusable code



TypeScript is
compiled to JavaScript in the end
!!!!!!

TypeScript starts with JavaScript
and ends with JavaScript

ECMAScript

The ECMAScript specification is a standardized specification of a scripting language

TypeScript is aligned with the ECMAScript6 specification.

All TypeScript code is converted into its JavaScript equivalent for the purpose of execution

Installation for Angular

To create and run you first angular app. You need to install following things

- Visual Studio Code
  https://code.visualstudio.com/download

- Node Js
  https://nodejs.org/en/download

Why and what is Visual studio Code

- Visual Studio Code is a lightweight but powerful source code editor developed by Microsoft which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, Typescript and Node.js.

- So you can code, compile and execute your typescript code in this editor.

- You can debug your code too. I will show that to you in later videos

Why and what is Node Js

- First I will tell you what Node Js is not :
  - Node JS is not a framework.
  - And it’s not a programming language.
- Most of the people are confused and understand it’s a framework or a programming language, but its not
- Rather Node.js is an open source runtime environment for executing JavaScript code.
- Its built on Google chrome’s JavaScript engine.
- And as discussed “TypeScript compiles to clean, simple JavaScript code which runs on any browser or in Node.js(JavaScript engine). ”
- So you can happily run your typescripts code in nodejs runtime environment, which interprets your code and show your results in browsers.

Some Important Node js commands

- **npm install -g @angular/cli**
    - Used to install Angular CLI. In code it will create node modules.

- The Angular CLI is a command-line interface tool that you use to initialize, develop, run and maintain Angular applications.

- But Before you can use Angular CLI, you must have
    - Node.js 6.9.0 and
    - npm 3.0.0 or higher installed on your system.

- **ng new my-first-project**
    - The Angular CLI makes it easy to create an application that already works, right out of the box. It already follows all best practices!

Some Important Node js commands

- **ng generate**
  - Generate components, routes, services and pipes with a simple command. The CLI will also create simple test shells(*spec.ts) for all of these.

- **ng serve**
  - Easily test your app locally while developing.

Services and Dependency Injection

Services and DI in Angular

Services and DI in Angular

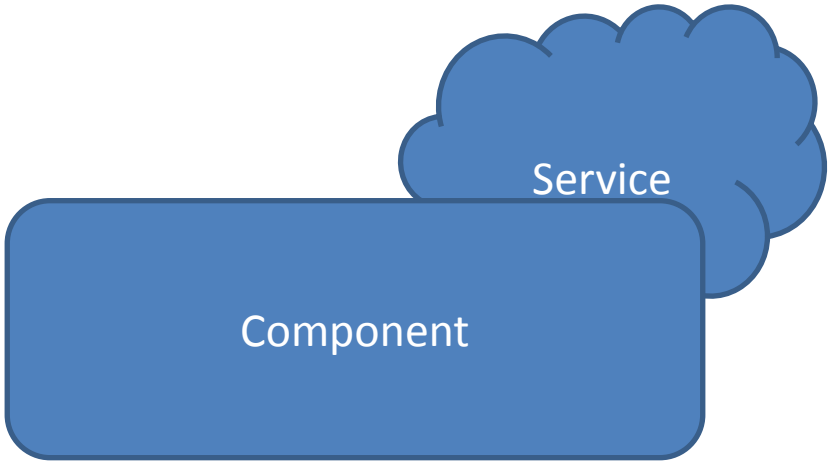
- A service is typically a class with a narrow, well-defined purpose. It should do something specific and do it well.

- In angular we have component and services as different concepts so as to increase modularity and reusability. By using services you can make your services lean and efficient.

- A component can delegate some task to services such as fetching data from server, logging etc

- Keep in mind angular does not force you to create services , you can write your logic in component itself but it’s a good practice to create separate services for application logic and .Use that service in component using dependency injection.

DI in Angular



We use DI in angular framework to provide new components these services. Components consumes services through DI mechanism.

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
  // providers:  [ HeroService ]
})
export class AppComponent {
  title = 'my-first';


  constructor(private service: CustomService) {

  }
```

Steps to create a service

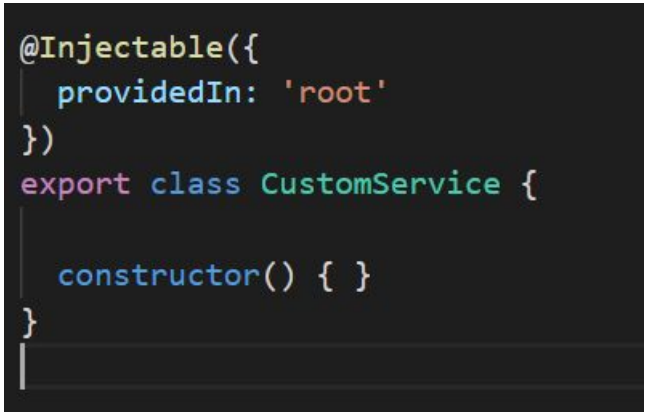
- Change your directory to the folder whose child will be this service.
- Angular CLI can generate service by following command :
  - ng generate service serviceName
- This command will generate a skeleton of your service. The most important part of an angular service is @Injectable() metadata.
- The injector is responsible for creating service instances and injecting them into classes that needs them and use them using dependency injection

Steps to create a service

- Services are singletons. When Angular creates new component instance, it checks constructor for all required dependencies. When it finds that a component requires a service then it first if the injector has any existing instances of that service. If a requested service instance doesn't yet exist, the injector makes one using the registered provider, and adds it to the injector before returning the service to Angular.

- A provider tells an injector how to create the service.

- Root means angular will use application level injector as a parent injector.



Levels for defining providers

Set provider metadata at these 3 levels

In @Injectable

In @Component

In @NgModule

```
@Injectable({
  providedIn: 'root'
})
```

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  providers:  [ CustomService ]
})
export class AppComponent {
```

```
@NgModule({
  declarations: [
    AppComponent,
    CustomComponent,
    ChildComponent,
    FormExampleComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [CustomService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

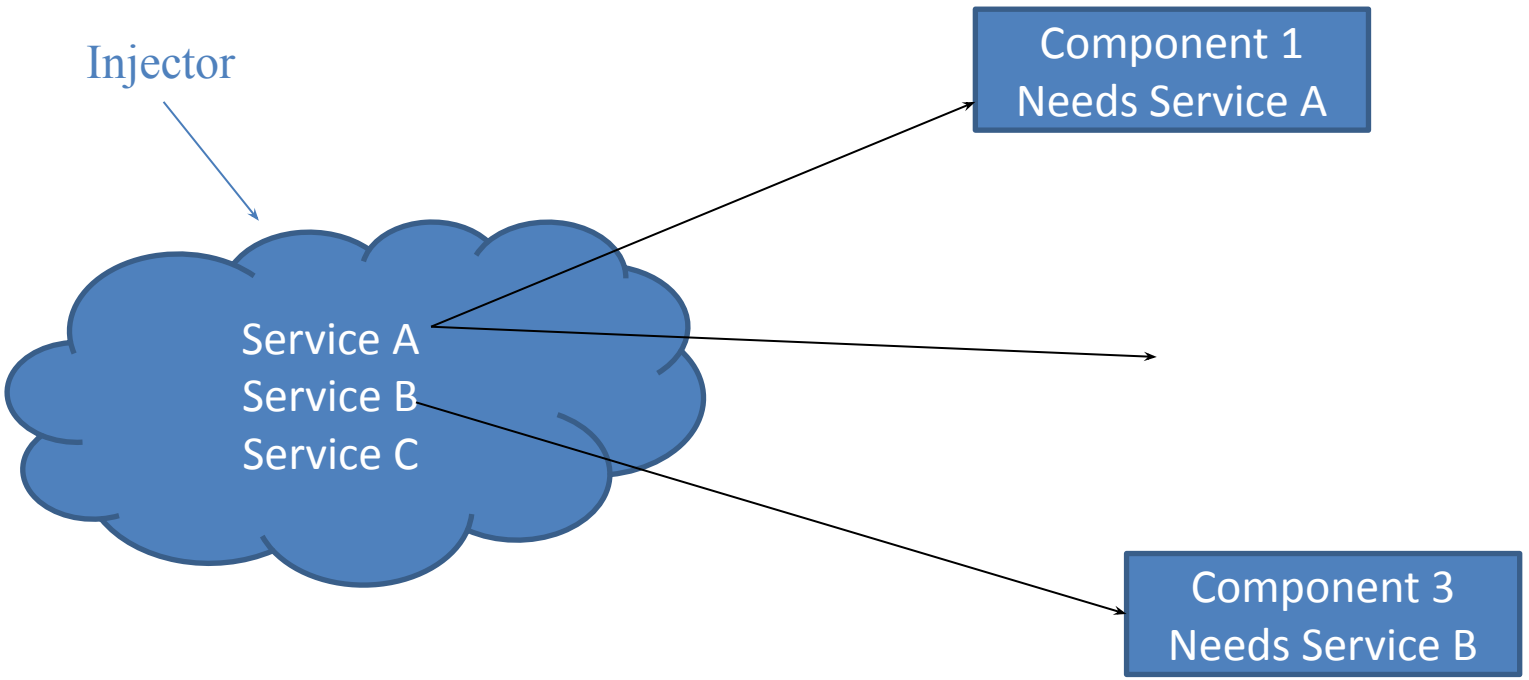
Levels for defining providers

- Provider in service's own metadata: This will make service available everywhere as injector is the root injector. Angular creates one shared instance of service for whole application

- Register provider in @NgModule: Same instance is available to all components in that module.

- Register provider in @Component: you get a new instance of the service with each new instance of that component

Dependency Injection



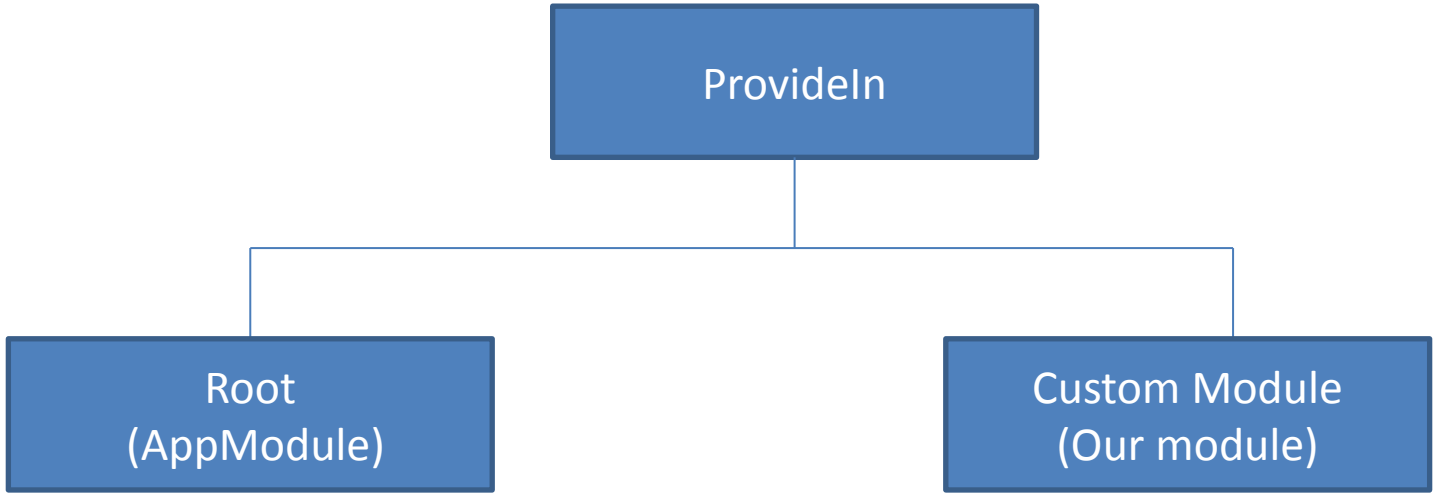
Tree Shakeable providers

Earlier modules declare what services it needs

To implement this we use provideIn property in @injectable decorator

Now service itself declares where it should be provided.

ProvideIn Ways



ProvideIn Root

Advantage: Service will bundled only if its used.
Imp: Using service in say components , components must also be used somewhere

We don’t need to import service in modules and thereby importing modules wherever services are required

Steps to Create First Angular application in local

- **Install Node.js:** Angular requires **Node.js and npm** (Node Package Manager) to be installed on your machine. Visit the official Node.js website (**https://nodejs.org**) and download it. Node.js is an **open source runtime environment like JVM** for executing JavaScript code. npm -v and ng version will be used to verify downloads

- **Install Angular CLI:** Angular CLI (Command Line Interface) is a powerful tool that simplifies the creation and management of Angular projects. Open a command prompt or terminal and run the following command to install Angular CLI globally on your machine:

- **npm install -g @angular/cli**
    - This will install Angular CLI globally and make it available as a command-line tool

- **Create a new Angular project:** Once Angular CLI is installed, you can create a new Angular project by running the following command:

Steps to Create First Angular application in local

- **ng new my-app**

- Replace "my-app" with the desired name for your project. Angular CLI will create a new directory with the project structure and necessary files.

- **Navigate to the project directory:** Change your working directory to the newly created project directory:

- **cd my-app**

- **Serve the application**: To start a local development server and serve your Angular application, run the following command:

- **ng serve**
  - This will compile your application and make it available at http://localhost:4200. Open your web browser and visit that URL to see your application running.

Understanding Folder structure

- **app/**
  - Contains the component files in which your application logic and data are defined. This is the folder containing business logics and screens code.

- **Assets/**
  - Contains image and other asset files to be copied as-is when you build your application.

- **environments/**
  - Contains build configuration options for particular target environments. By default there is an unnamed standard development environment and a production ("prod") environment. You can define additional target environment configurations.

- **Favicon.ico**
  - An icon to use for this application in the bookmark bar. I will show you how.

Understanding Folder structure

- **Index.html**
  - The **main HTML page** that is served when someone visits your site. This is the very first page from where your application's life starts. As soon as you hit localhost:4200/ you will land at a main page. Right click and select inspect. You will be able to see your index.html content and with that some extra code like scripts, app code data. The CLI automatically adds all JavaScript and CSS files when building your app, so you typically don't need to add any <script> or<link> tags here manually. So cli modifies this file for us.

- **Main.ts**
  - The main entry point for your application. It bootstraps the application's root module (AppModule) to run in the browser.

- **Polyfills.ts**
  - Provides polyfill scripts for browser support.

Understanding Folder structure

- **styles.sass/ styles.css**
  - Lists CSS files that supply styles for a project. The extension reflects the style preprocessor you have configured for the project.

@CrossOrigin

- To connect your FE to Microservices at backend use this at controller level else u will face following error:
    - Access to XMLHttpRequest at 'http://localhost:9091/restaurant/fetchAllRestaurants' from origin 'http://localhost:4200' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

Commands used

- cd .\src\app\
- ng generate module restaurant-listing --routing
- ng generate component restaurant-listing --module=restaurant-listing

What is Router outlet

- By using RouterOutlet in your application, you can create a dynamic and responsive user interface that seamlessly updates the displayed content as users navigate through different routes or URLs.

Understanding File structure

- **app/app.module.ts**
  - Defines the root module, named AppModule, that tells Angular how to assemble the application. Initially declares only the AppComponent. As you add more components to the app, they must be declared here.
  - The purpose of a NgModule is to declare each thing you create in Angular, and group them together (like Java packages)
  - Your Angular application has at least one module which is called the root module

- **Bootstrapping :**
  - Reading and compiling is the bootstrapping process, ie creating an instance in v simple terms. Bootstrapping is where the application is loaded when Angular comes to life.

Understanding File structure

- The most important properties are as follows.
    - **declarations** - Declare views to make them privately available in this module.
    - **exports** - Makes the declared view public so they can be used by other modules.
    - **imports** - This is where you import other modules.
    - **providers** - Defines services that can be injected into this module's views.
    - **bootstrap** - The component used to launch the app, the AppComponent by default. All apps must have at least one.
- run the following command to create a custom module named admin:
    - $ **ng g module admin** or **ng generate module <module_name>**

Understanding File structure

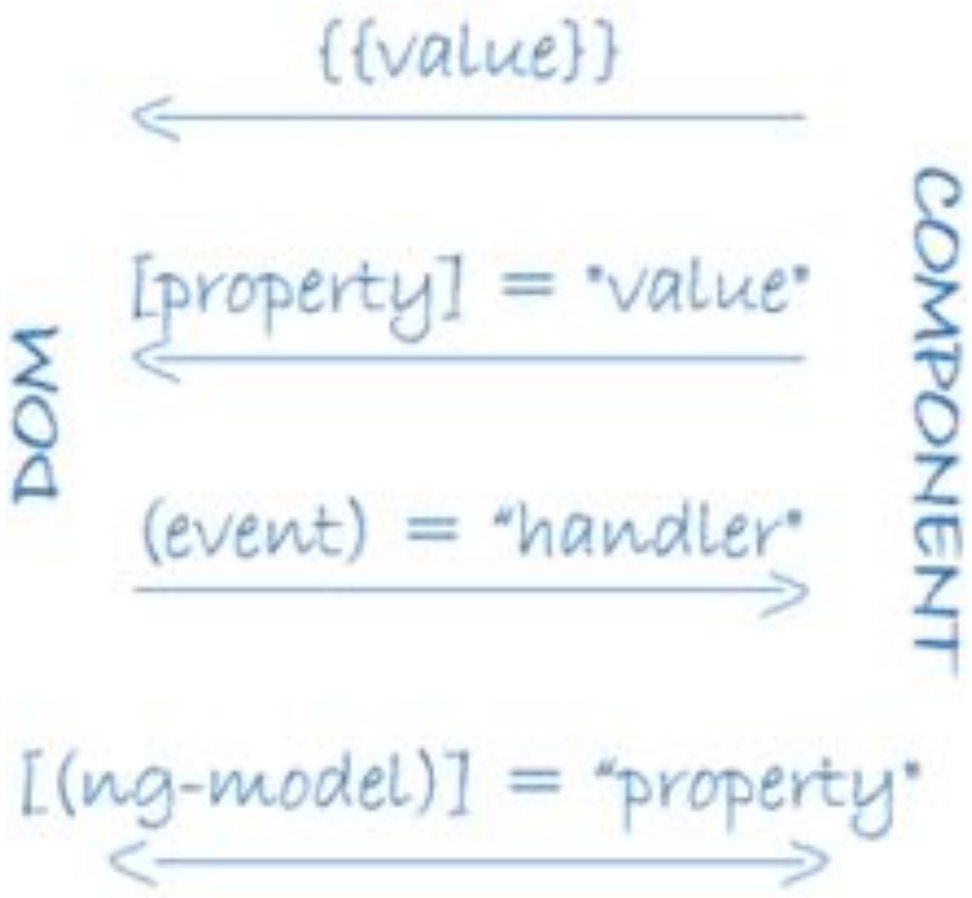
- **app/app.component.ts**
  - Defines the logic for the app's root component, named AppComponent. The code here is responsible for the part of the screen you see in front of you.
  - A **component controls a patch of screen called a view**.
  - @Component decorator identifies the class immediately below it as a component class. It's not a component until you mark it as one with the @Component decorator.
  - What you see as arguments in component decorator is the metadata for a component.
  - The metadata for a component tells Angular where to get the major building blocks that it needs to create and present the component and its view.

- You can create a custom component by command :
  - **Ng generate component “compname” or ng g c “compname”.**

Understanding File structure

- **Template**:
- Template defines a view of your screen. So what you see in front of you is the result of html and angular logics written in your template file
- A template looks like regular HTML, except that it also contains Angular syntax.
- Angular syntaxes alters the HTML view based on your app's DOM data.
- Data binding to coordinate the app and DOM/ view data.
- There are 4 types of data binding:
    - Interpolation(one-way data binding)
    - Property binding(one-way data binding)
    - Event binding (one-way data binding)
    - 2 way binding

Understanding File structure



Understanding File structure

- **Selector**: how the component can be referenced in HTML . A CSS selector that tells Angular to create and insert an instance of this component wherever it finds the corresponding tag in template HTML, For example, if an app's HTML contains <custom-comp></ custom-comp >, then Angular inserts an instance of the HeroListComponent view between those tags.

- **styleUrls**: contains array of style sheets to be applied on html / template for showing the view.

- **providers**: An array of providers for services that the component requires.

Using Built-in structural directives

- Structural directives are responsible for HTML layout. They shape or reshape the DOM's structure, typically by adding, removing, and manipulating the host elements to which they are attached.

- This section is an introduction to the common built-in structural directives:

    - NgIf—conditionally creates or destroys subviews from the template.
    - NgFor—repeat a node for each item in a list.
    - NgSwitch—a set of directives that switch among alternative views.

Why we create Models as Interfaces and not classes

- Interfaces are useful for describing the shape and structure of data objects.
- Interfaces are lightweight and do not generate any code in the compiled JavaScript output.
- They are mainly used for type-checking and providing a contract for objects to adhere to.
- While Classes can provide more functionality and encapsulation compared to interfaces.
- They are often used when you need to instantiate objects, maintain internal state, or perform operations on the data.

Why we need to export a component from module if needed in another module

- Import the HeaderComponent and add it to the declarations and exports arrays:
- Interfaces are lightweight and do not generate any code in the compiled JavaScript output.
- They are mainly used for type-checking and providing a contract for objects to adhere to.
- While Classes can provide more functionality and encapsulation compared to interfaces.
- They are often used when you need to instantiate objects, maintain internal state, or perform operations on the data.

Kubernetes

What is Kubernetes?

- Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications.

- Orchestration involves managing the lifecycle of containers, including their deployment, scaling, networking, storage, and monitoring. Kubernetes acts as the central control plane that automates and coordinates these tasks, ensuring that the desired state of the application or system is maintained.

- Containerization allows applications to be packaged along with their dependencies, such as libraries and configuration files, into lightweight, portable units called containers. Kubernetes provides a framework for managing and running these containers across a cluster of machines.

Why Kubernetes?

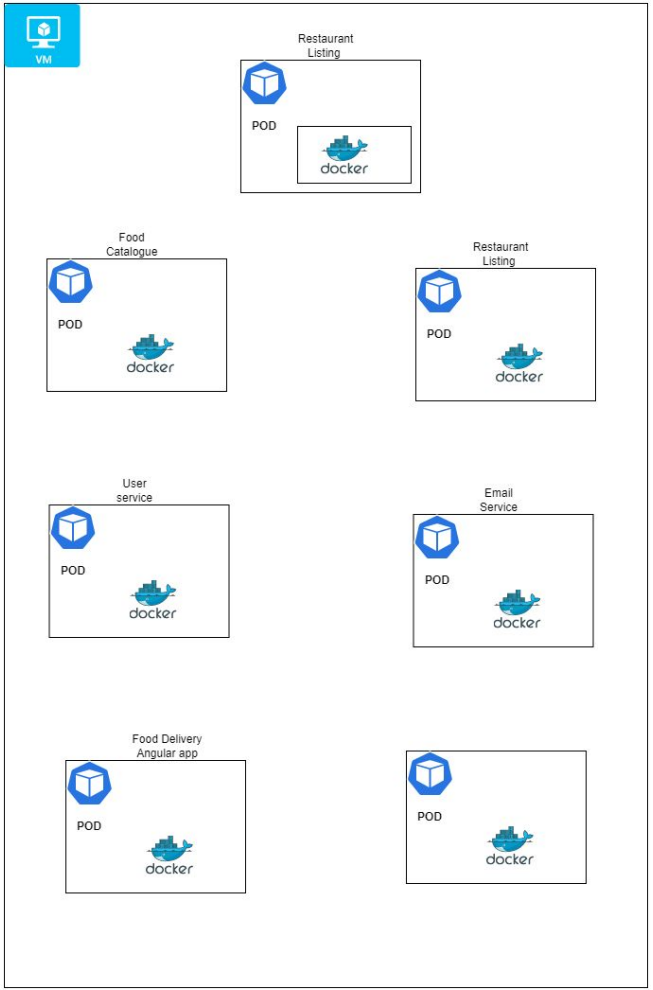
- Simplified Deployment
- Automated Scaling and Load Balancing
- Self-Healing and Fault Tolerance
- Rollouts and Rollbacks
- High availability with technically no downtime
- Kubernetes makes it easier to develop, deploy, and manage applications, allowing developers and operations teams to focus more on delivering value and less on infrastructure management.

Components of Kubernetes?

- Node
- Pods
- Services
- Ingress
- Config maps
- Secrets
- Volumes
- Deployments
- Stateful set

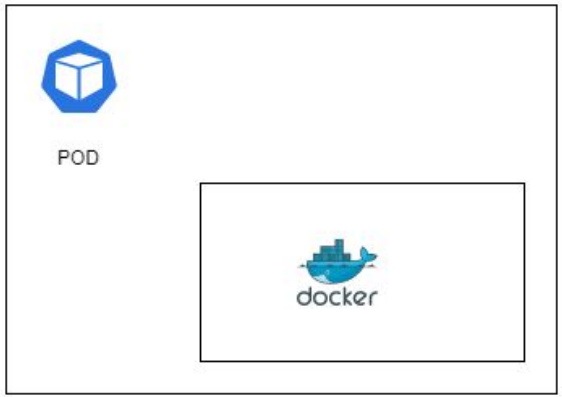
What is a node?

- You need to deploy your app on a physical server or a virtual machine (VM) and that machine is called a worker node or simply a "node. Or a slave



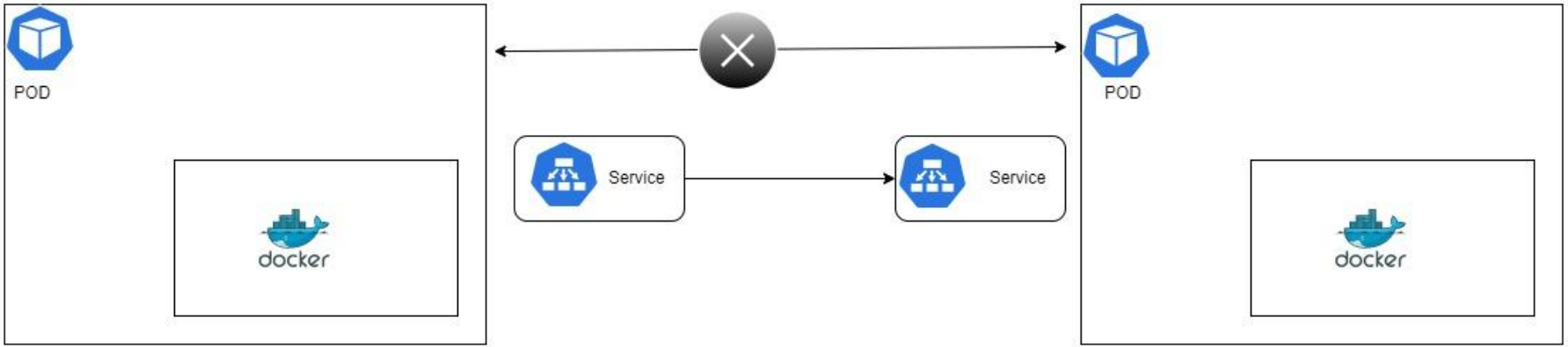
What is a POD?

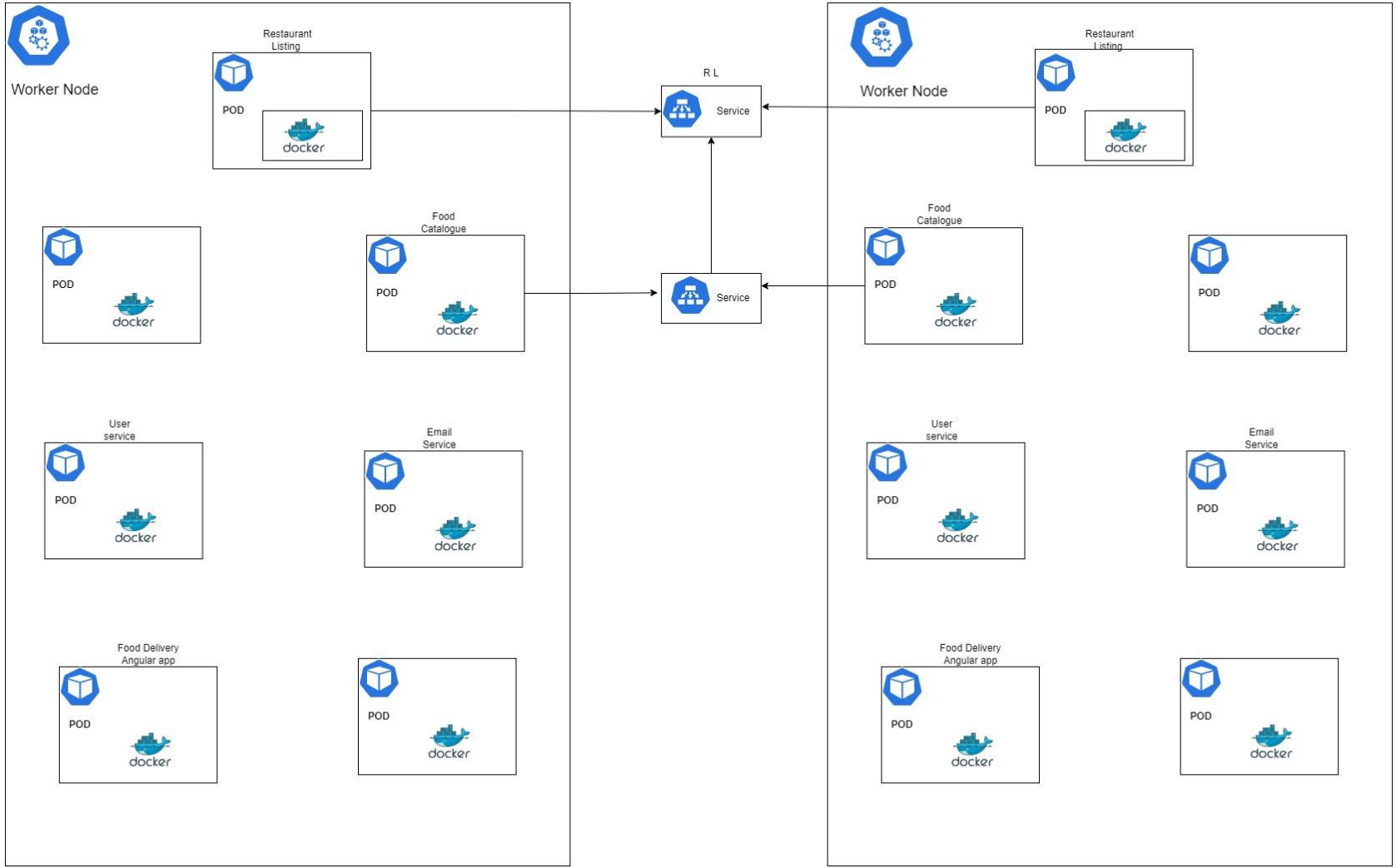
- In Kubernetes (K8s), a pod is the smallest deployable unit and the basic building block of the platform.
- It represents a group of one or more tightly coupled containers that are scheduled and run together on the same node within a cluster.
- But in real time scenarios u will see maximum 1 Container running inside each POD
- Pods are designed to be crashable and disposable
- Each Pod has its own IP address
- Each POD can communicate using These IP addresses
- But if POD crashes then? Will u change mapping Everytime the POD crashes?



What is a Service?

- a service provides a consistent and reliable way to access and communicate with a group of pods
    - Provides permanent IP address to PODs
    - Load balancing




Worker Node
Restaurant Listing
POD
docker
R L
Service
Worker Node
Restaurant Listing
POD
docker
POD
docker
Food Catalogue
POD
docker
Service
Food Catalogue
POD
docker
POD
docker
User service
POD
docker
Email Service
POD
docker
User service
POD
docker
Email Service
POD
docker
Food Delivery Angular app
POD
docker
POD
docker
Food Delivery Angular app
POD
docker
POD
docker

Types of Service?

- **ClusterIP**: This is the default service type. It exposes the service on an internal IP within the cluster, and it is only accessible from within the cluster.

- **NodePort**: This type exposes the service on a specific port of each cluster node's IP address. It makes the service accessible from outside the cluster by using the node's IP address and the assigned NodePort.

- **Load Balancer**: This type provisions an external load balancer (e.g., from a cloud provider) that directs traffic to the service. It automatically assigns an external IP address, making the service accessible from outside the cluster.

How to expose our services to resources hosted outside cluster?

- To expose a service within a Kubernetes cluster to outside client browsers, you have a few options depending on your cluster setup and requirements

- **NodePort**: Define your service with a type: NodePort in its Service specification.

    - Note: This approach is typically used for development or testing purposes and may not be suitable for production deployments.

- **LoadBalancer**: If your cluster is running in a cloud provider environment that supports load balancers, you can expose the service using a LoadBalancer type service

    - Define your service with a type: LoadBalancer in its Service specification.
    - The cloud provider provisions a load balancer that distributes incoming traffic to the service.
    - The load balancer gets an external IP address, allowing clients to access the service using that IP.
    - Note: This approach relies on the cloud provider's load balancer and may incur additional costs.

How to expose our services to resources hosted outside cluster?

- **Port Forwarding:** For testing purposes or accessing services from your local machine, you can use port forwarding.
    - Use the kubectl port-forward command to forward traffic from a local port to the desired port of the service within the cluster.
    - Clients can access the service using localhost or the specified local IP and port.
- Note: Port forwarding is typically used for development or debugging and is not recommended for production scenarios.
- Now if u have 100 of Pods/services you can’t keep on port forwarding so better to use Ingress.
- Because of Its powerful routing mechanism, It gives u one single IP / URL to interact with All Backend as well as Front end applications also

How to expose our services to resources hosted outside cluster?

- **Ingress**: Ingress is a powerful option for exposing services within a cluster using HTTP/HTTPS. so instead of service the request goes first to Ingress and it does the forwarding then to the service
    - Deploy an Ingress controller in your cluster (e.g., Nginx Ingress Controller, Traefik, etc.).
    - Its same as API Gateway (Entry point + Routing rules are defined here). Define an Ingress resource with the desired routing rules, including the hostname, URL paths, and the backend service to forward the traffic to.
    - The Ingress controller configures the underlying load balancer or reverse proxy to handle the traffic based on the Ingress rules.
    - Clients can access the service using the Ingress IP or hostname, which is configured to point to the Ingress controller.
    - Note: Ingress requires an Ingress controller to be installed and properly configured in your cluster.

How to expose our services to resources hosted outside cluster?

- If you are using AWS - then steps to implement Ingress -

- **Set up your Kubernetes cluster on AWS**: Create a Kubernetes cluster using a managed service like Amazon Elastic Kubernetes Service (EKS) or using a self-managed cluster on EC2 instances.

- **Deploy an Ingress controller**: Choose an Ingress controller that is compatible with AWS, such as the AWS ALB Ingress Controller or Nginx Ingress Controller. Deploy the selected Ingress controller into your Kubernetes cluster. Each controller has its own installation instructions, so refer to the specific documentation for the chosen controller.

- **Create and configure Ingress resources**: Define Ingress resources that specify the routing rules for your services. The Ingress resources will include details such as hostnames, URL paths, and backend services. Apply these Ingress resources to your cluster using the kubectl apply command or other deployment methods.

How to expose our services to resources hosted outside cluster?

- **Provision an Application Load Balancer (ALB**): If you are using the AWS ALB Ingress Controller, it requires an ALB to be provisioned in AWS. Follow the documentation provided by the AWS ALB Ingress Controller project for detailed instructions on how to set up the ALB. The ALB will be responsible for routing traffic based on the Ingress rules defined in your cluster.

- **Configure security groups and firewall rules**: Set up appropriate security groups and configure firewall rules in AWS to allow incoming traffic to the ALB on the specified ports.

- **Verify and test:** Once the Ingress resources and the ALB are set up, you can test the connectivity by accessing your services using the ALB's DNS name or IP address.

What is a ConfigMap?

- ConfigMap is used to store configuration data that can be consumed by containers running in pods.
- It provides a way to decouple the configuration from the application code, allowing for easy configuration changes without modifying the container image.
- Else every time you have to rebuild the application then push it to the repository and then pull that new image in your pod and restart the whole thing
- Thus you can separate the configuration concerns from the application logic, making it easier to manage and update the configuration independently of the container images. It promotes flexibility, maintainability, and portability in a Kubernetes environment.

What is Secrets?

- secret is just like config map but the difference is that it's used to store secret data credentials for example and it's stored not in a plain text format of course but in base64 encoded format.

- Secret is used to store and manage sensitive information, such as passwords, API keys, tokens, or certificates. Secrets provide a way to securely store and distribute sensitive data to containers running in pods.

Difference between Secrets and config maps?

- Secrets are specifically designed to store and manage sensitive data, while ConfigMaps are used for non-sensitive configuration data.

- Secrets provide encryption, additional security features, and are immutable, whereas ConfigMaps are more flexible for configuration updates.

- The choice between Secrets and ConfigMaps depends on the type of data you need to store and the level of sensitivity associated with that data.

What are volumes on K8?

- Volumes are a way to provide persistent storage to containers running in pods . It plays a crucial role in managing and persisting data in Kubernetes.
- They enable containers to work with shared and persistent storage, allowing for data integrity, data sharing between containers, and reliable data storage beyond the lifecycle of individual containers.
- Means even if your DB Pod crashes, your data is safe in volumes .
- If you don’t use them, then your data will reside in PODs. Pods are designed to be crashable units and when they crash, all your data is gone !!
- But now if u replicate these DB Pods, they must use same Volumes for data consistency so it’s always advisable to move these DB to some external DB services like AWS RDS services, Azure DBs

How to create Replicas sets in K8

- For High available , fault tolerance and load balancing we need replicate sets for same pod
- So if I need 3 Replica set of each application will u create 3 pods manually ?.
- Answer is NO !! we never work at POD level. We work at Deployment level.

What are deployments?

- Deployment in Kubernetes is like a blueprint or set of instructions for running and managing your application.

- Think of a Deployment as a manager for your application. It ensures that the desired number of copies of your application are always running, even if a pod or node fails. If you want to scale your application, you can simply tell the Deployment to add more replicas, and it will automatically create and manage them for you.

- Deleting Deployment will delete Pods, replica sets but will not delete services, ConfigMaps, Secrets, and any associated persistent volumes.

What are stateful sets?

- StatefulSet in Kubernetes is a way to manage applications that need to maintain their identity and persist data, such as databases.

- StatefulSets also ensure that each pod has its own persistent storage, so data can be stored and retrieved even if the pod is terminated or moved to a different node. This is important for applications that need to keep their data intact, like databases.

- StatefulSets provide a way to deploy, scale, and update these stateful applications

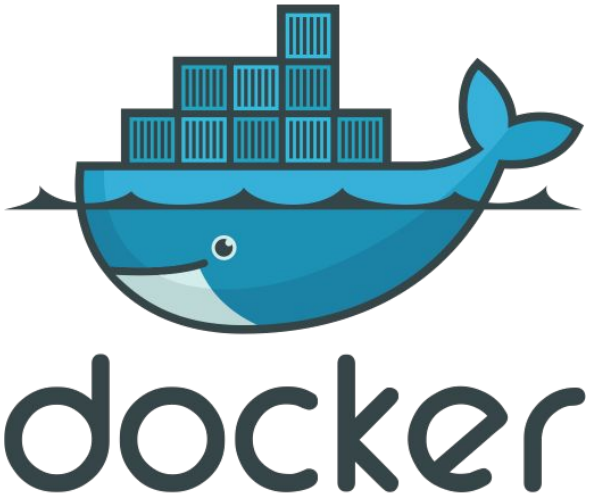
Architecture of K8

- It consists of a Master Node that manages the cluster and worker nodes that execute containers.

What is Docker

What is Docker

- We can Imagine Docker as a **Cargo Ship** that has the ability to hold big boxes (containers) having their distinct objectives and id.
- These containers contain **items** (application + Dependencies) that are required to make that container useful and runnable anywhere.
- These **items** are manufactured using **templates** (Docker File).



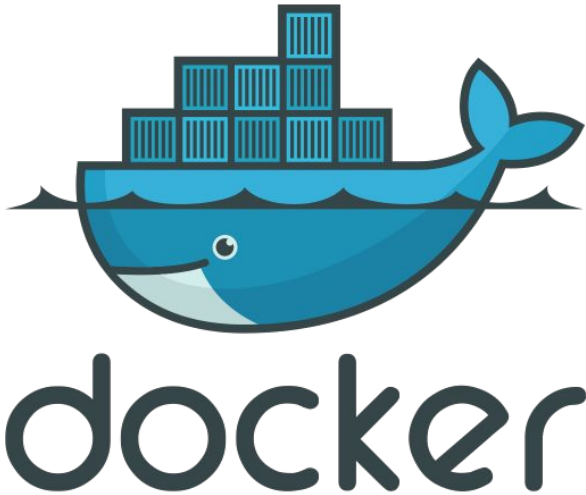
What is Docker

- Docker - Docker is a **containerization platform** which packages your application and all its dependencies together in the **form of containers** so as to ensure that your application works seamlessly in any environment, be it development, test or production.

- Hence A container **contains complete ecosystem of your application**

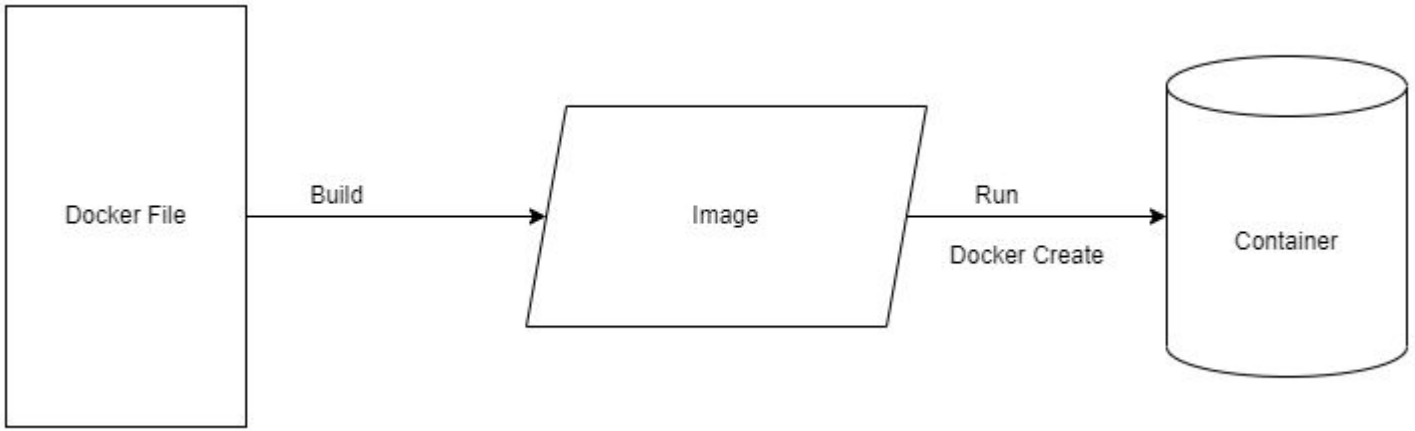
- including the application and all of its dependencies, and everything that is needed **for an application to run on the server**

- This guarantees that the software will always run the same, regardless of its environment.



How Dockerfile, Image & Container are created

How Docker works



What is Dockerfile

- This text file provides a set of instructions to build a Docker image, including the operating system, languages, environmental variables, file locations, network ports, and any other components it needs to run.

- Each time a command is run in a Dockerfile, a new layer is created on top of the previous layers. This allows Docker images to be built incrementally, with each layer representing a separate instruction in the Dockerfile. It also allows Docker to reuse layers between images, which can help reduce the size of images and improve build times.

- For example, if you specify a command to install a package in your Dockerfile, a new layer will be created that includes the package and its dependencies. If you make another change, such as adding a file to the image, another layer will be created on top of the previous one

What is Docker image ?

- Docker-images are a **read-only binary template** (Like snapshots) used to build containers. Images also contain metadata that describe the container's capabilities and needs.

- Create a docker image using the **docker build** command whenever you pass a Dockerfile to the docker build command then the **docker daemon** will create a docker image according to the Dockerfile instruction.

- Docker images can't be executed by themselves and cannot run or start. It is just a blueprint for creating Docker containers

- Run the docker images using the **docker run** command. whenever we pass the command to docker client then the docker client passes this command to the docker daemon then docker daemon will **create the container for that image.**

- Push the docker image to the public registry like DockerHub using the **docker push command** after pushed you can access these images from anywhere using **docker pull command.**

What is Docker container ?

- A container is a **runnable instance of an image**. You can create, start, stop, move, or delete a container using the Docker API or CLI.

- Containers provide you with a **lightweight and platform-independent way of running your applications**. Every container is isolated but access to resources on another host or container can be allowed with the help of docker networking.

- A container is **volatile** it means whenever you remove or kill the container then all of its data will be lost from it. If you want to persist the container data use the **docker storage** concept.

- **Containers only have access to resources that are defined in the image**, unless additional access is defined when building the image into a container

- All the docker images become docker containers when they run the on **Docker Engine**

What is Docker hub ?

- **Docker images are stored** in a registry, which is a **centralized location** for storing and distributing Docker images.

- When you run a Docker container, Docker pulls the image from the registry and runs it on your local machine.

- Several public registries are available, such as Docker Hub, the default registry for Docker. You can also set up your private registry if you want to store and manage images internally.

- Docker hub is a **cloud-based registry service** that allows you to link to code repositories, build your images and test them, store manually pushed images, and link to the Docker cloud so you can deploy images to your hosts. It provides a centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline.

  https://hub.docker.com/

Did you use Docker in your Project? If yes why ?

- Yeah we did. We had a common problem of Developers saying “BUT IT WORKS ON MY MACHINE”
- There are some scenarios which motivated us to move to dockerized application
- We sometimes faced issue that our same code base when deployed to test, dev , worked fine but breaks on prod or QA servers . We found that Even though code base is same but dependencies / their version were problematic hence if we are able to package everything in one big container and run application in that container with same configuration, dependence then things wont break abruptly. Hence we moved to Docker
- Apart from that we achieved few more advantages like -
- Docker is an open platform for developing, shipping, and running applications.
- Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.

Did you use Docker in your Project? If yes why ?

- With Docker, you can manage your infrastructure in the same ways you manage your applications.
- By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.
- Use cases involves -
- Environment standardization
- Faster configuration with consistency
- Better disaster recovery - Disaster is unpredictable. However, you can back up a Docker image (also called "snapshot") for the state of the container at that back-up moment, and retrieve it later when serious issues happen. For example, a hardware failure just happened and you need to switch your work to a new hardware. With Docker, you can easily replicate the file to the new hardware.

Did you use Docker in your Project? If yes why ?

- Sometimes we found a bug when deploying a new version of one particular software. We can revert to the last version with the previous Docker image easily. Without Docker, we have to set up the rollback step from runtime to runtime

- Improvement in adoption of DevOps

Disadvantages of using Docker

- Security - Containers are lightweight, but you pay for this with security. Since containers in most cases use a common operating system, there is a general risk that several containers will be compromised at once if a host system is attacked. This is less likely with VMs as each VM uses its own operating system.

- Containers don't run at bare-metal speeds - Docker containers use the Host Kernal only hence it will use as much system resources as the host's kernel scheduler will allow. Docker might even make it slower. if the kernel detects that the host machine's memory is running too low to perform important system functions, it could start killing important processes (may be your docker process too)

- Persistent data storage is complicated. By design, all of the data inside a container disappears forever when the container shuts down, unless you save it somewhere else first. There are ways to save data persistently in Docker, such as Docker Data Volumes, but this is arguably a challenge that still has yet to be addressed in a seamless way.

- Not For Simpler Deployable applications - So, if you have a complicated and tedious deployment process, Docker will help you out a lot. If you have a simple app, it just adds unnecessary complexity.

Disadvantages of using Docker

- Security - Containers are lightweight, but you pay for this with security. Since containers in most cases use a common operating system, there is a general risk that several containers will be compromised at once if a host system is attacked. This is less likely with VMs as each VM uses its own operating system.

- Containers don't run at bare-metal speeds - Docker containers use the Host Kernal only hence it will use as much system resources as the host's kernel scheduler will allow. Docker might even make it slower. if the kernel detects that the host machine's memory is running too low to perform important system functions, it could start killing important processes (may be your docker process too)

- Persistent data storage is complicated. By design, all of the data inside a container disappears forever when the container shuts down, unless you save it somewhere else first. There are ways to save data persistently in Docker, such as Docker Data Volumes, but this is arguably a challenge that still has yet to be addressed in a seamless way.

- Not For Simpler Deployable applications - So, if you have a complicated and tedious deployment process, Docker will help you out a lot. If you have a simple app, it just adds unnecessary complexity.

Adding Dockerfile

To all Backend and Frontend Applications

Creating Dockerfile in all Backend Microservices

- We are not using Expose and env instructions because
    - **EXPOSE instruction is used to inform Docker that a container process listens on a specific network port at runtime**. It does not actually expose the port to the host or make it accessible from outside the container. The purpose of the EXPOSE instruction is to **provide metadata about the intended network ports that the container process will use.**
    - When using Kubernetes for deployment, you do not need to rely on the EXPOSE instruction in the Dockerfile. Instead, **you define the desired ports in the Kubernetes Service configuration to expose your application within the cluster or to external clients.**
    - ENV PORT instruction in the Dockerfile is not mandatory when using Kubernetes (k8s) for deployment. **The ENV PORT instruction is typically used in a Dockerfile to set an environment variable called "PORT" with a specific value.** Instead, you can specify the desired port(s) in the Kubernetes Service or Deployment configuration

Creating Dockerfile in all Front end Angular Application

- Dockerfile For FE consists of two stages, each with its own purpose. Let's go through each stage:

    - **Stage 1: Build the Angular app**
      FROM node:16 as build
      WORKDIR /app
      COPY package*.json ./
      RUN npm install
      COPY . .
      RUN npm run build

    - In this stage, the base image is set to node:16, which includes Node.js version 16. The as build tag is used to give this stage a name, which can be referenced later.

    - The WORKDIR instruction sets the working directory inside the container to /app. This is where the subsequent commands will be executed.

Creating Dockerfile in all Front end Angular Application

- Next, the COPY command copies the package.json and package-lock.json files from the local directory to the /app directory inside the container.
- The RUN npm install command installs the dependencies specified in the package.json file.
- After that, the COPY . . command copies all the remaining files and directories from the local directory (the context where the Docker build command is run) into the /app directory inside the container.
- Finally, the RUN npm run build command executes the build script defined in the package.json file. This builds the Angular app within the container and generates the build artifacts

Creating Dockerfile in all Front end Angular Application

- Stage 2: Serve the Angular app using Nginx

```
FROM nginx:alpine
COPY --from=build /app/dist/food-ordering-app /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- In this stage, the base image is set to nginx:alpine, which includes Nginx, a lightweight web server.

- The COPY command copies the build artifacts from the previous stage (named build) into the /usr/share/nginx/html directory inside the container. This is the default location where Nginx looks for static files to serve.

- The EXPOSE instruction indicates that the container will listen on port 80, which is the default HTTP port used by Nginx.

Creating Dockerfile in all Front end Angular Application

- Finally, the CMD instruction specifies the command to run when the container starts. It starts the Nginx server with the "daemon off;" option, which keeps Nginx running in the foreground and prevents it from detaching from the container.

Build Image for all Applications

- Docker build -t codedecode25/app-name:0.0.1 .
- . represents current director
- Apps can be - restaurant-listing-service, order-service etc. All lower case segregated by a hyphen for better readability

Push Image to DockerHub

- Log in to Docker Hub using
  - docker login

- Docker push codedecode25/app-name:0.0.1

Introduction to AWS

For Deployment

What is Aws

- It is a platform that offers cloud computing solutions
- The platform is developed with a combination of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings.

Services of AWS to be used

- **AWS RDS** - This Database AWS service is easy to set up, operate, and scale a relational database in the cloud
- **AWS EKS** - The tool allows you to use Kubernetes on Amazon cloud environment without installation.
- **AWS ALB** - Load balancer
- **AWS EC2 Instance** - EC2 is a virtual machine in the cloud on which you have OS level control. You can run this cloud server whenever you want.

How to setup AWS account

- https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all
- https://docs.aws.amazon.com/accounts/latest/reference/manage-acct-creating.html

Install Chocolatey

To Install multiple softwares

How to Install Chocolatey

- Follow Instructions on https://chocolatey.org/install

Getting started with Amazon EKS

To install eksctl, kubectl and AWS CLI

How to install different CLI’s using chocolatey

- choco install awscli
- choco install eksctl
- choco install kubernetes-cli
- While creating EKS clusters we need these 3 command line tools to interact with AWS, n k8 clusters created using eksctl.

AWS EKS

To create eks

What is EKS

- AWS EKS (Amazon Elastic Kubernetes Service) is a **managed container orchestration service** provided by Amazon Web Services (AWS).
- It simplifies the deployment, management, and scaling of containerized applications using Kubernetes.
- AWS EKS is suitable for organizations and developers looking to leverage Kubernetes for container orchestration **without the complexity of managing the underlying infrastructure**.
- It offers a reliable, scalable, and secure environment for deploying and managing containerized applications in the AWS cloud.

What is Amazon EKS control plane architecture

- The control plane in a Kubernetes cluster is like the brain or central command center of the cluster. It is responsible for managing and controlling all the nodes (machines) and applications running within the cluster.
- It runs on managed master nodes that are fully managed and maintained by AWS.
- The control plane consists of several components that work together:
- **API Server:** This is like the interface or entry point for users and other components to interact with the cluster. It receives requests and instructions and communicates with other components to carry out the necessary actions.
- **etcd**: This is like the memory or database of the cluster. It stores the current state and configuration information of the cluster, including things like which nodes are part of the cluster, which applications are running, and other important details.

What is Amazon EKS control plane architecture

- **Controllers**: These are like the managers or supervisors of the cluster. They constantly monitor the cluster's state, compare it with the desired state (as defined by users or configurations), and take actions to bring the cluster to the desired state. For example, if a node fails, the Node Controller will detect it and take action to replace or repair the node.

- **Scheduler**: This component is responsible for deciding where and how to run new applications or workloads within the cluster. It considers factors like available resources, constraints, and requirements to make smart decisions about workload placement.

- **Networking**: The control plane also handles networking aspects, ensuring that nodes can communicate with each other and that applications running on different nodes can reach each other.

How does Amazon EKS work?

- **Create an Amazon EKS cluster** in the AWS Management Console or with the AWS CLI or one of the AWS SDKs.

- **Launch managed or self-managed Amazon EC2 nodes,** or deploy your workloads to AWS Fargate.

- When your cluster is ready, you can **configure** your favorite Kubernetes tools, such as **kubectl**, t**o communicate with your cluster.**

- **Deploy and manage workloads on your Amazon EKS cluster the same way that you would with any other Kubernetes environment (** like Rancher, **)**. You can also view information about your workloads using the AWS Management Console.

Prerequisites for creating EKS cluster

Prerequisite

- Before starting with EKS cluster creation, you must install and configure the following tools and resources that you need to create and manage an Amazon EKS cluster.
  - Kubectl
  - eksctl
  - Required IAM permissions with AWS CLI

Connect to AWS using CLI

Why do we need AWS CLI

- The AWS Command Line Interface (CLI) is a powerful tool that allows users to interact with various Amazon Web Services (AWS) resources and services through a command-line interface.

Generate Access key in AWS IAM for local connect through CMD

- Sign in to the AWS Management Console using your root user credentials.
- Open the AWS Management Console and navigate to the AWS Identity and Access Management (IAM) service.
- Inside IAM click manage Access Keys
- Click create access key and download .csv file

Connect to AWS through CMD using CLI

- Open cmd and run command “aws configure”
- Enter AWS Access Key ID and AWS secret access key

What is eksctl

What is EKSCTL

- eksctl is a simple command line tool for creating and managing Kubernetes clusters on Amazon EKS.

- https://eksctl.io/.

- We can create a basic cluster in minutes with just one command
  - eksctl create cluster

- A cluster will be created with default parameters :

- exciting auto-generated name, e.g., fabulous-mushroom-1527688624
- two m5.large worker nodes
- use the official AWS EKS AMI
- us-west-2 region
- a dedicated VPC (check your quotas)

What is EKSCTL

- But we might need some customization so either we can
  - Create a yaml
  - Give params in command line

- eksctl provides the fastest and easiest way to create a new cluster with nodes for Amazon EKS

How to deploy EKS cluster

- Deploy eks cluster using below command
- eksctl create cluster --name cluster name --region region name --nodegroup-name node name --node-type t3.medium --nodes 1
- **--name cluster-name:** Specifies the name of the EKS cluster you want to create. Replace "cluster-name" with your desired name for the cluster.
- **--region region-name:** Specifies the AWS region where you want to create the EKS cluster. Replace "region-name" with the desired AWS region, such as "us-west-2" for US West (Oregon).
- **--nodegroup-name node-name:** Specifies the name of the node group within the EKS cluster. Replace "node-name" with your desired name for the node group.

How to deploy EKS cluster

- **--node-type t3.medium:** Specifies the EC2 instance type for the worker nodes in the node group. In this case, it is set to "t3.medium". You can choose a different instance type according to your requirements.

- **--nodes 1:** Specifies the desired number of worker nodes in the node group. In this case, it is set to "1". You can adjust this number as needed.

- By running this command, eksctl will create an EKS cluster with the specified name, in the specified region, with a node group using t3.medium instances and a desired capacity of 1 node.

- When the eksctl create cluster command completes successfully, it generates the **kubeconfig** file with the appropriate configurations, including the cluster's endpoint, authentication details, and other necessary information. This kubeconfig file is then stored on your local machine in the default location.

Tasks done by this simple command

- Cluster Configuration:.
- **Node Group Creation**: eksctl provisions the specified node group(s) within the EKS cluster. This involves launching EC2 instances or Fargate pods as **worker nodes** that join the cluster. The command sets up the necessary configuration, such as instance types, instance profiles, and scaling options.
- **kubeconfig Update**: Once the cluster, control plane, and node groups are created, eksctl updates the kubeconfig file on your local machine. The kubeconfig file is configured with the necessary authentication details, cluster endpoint, and other configurations required to connect to the EKS cluster using tools like kubectl.
- **IAM Role Creation**: eksctl creates an IAM role for the EKS cluster's control plane. This role grants necessary permissions to manage the cluster and its resources.

Tasks done by this simple command

- **VPC Creation**: If a VPC is not already available, eksctl creates a new Amazon Virtual Private Cloud (VPC) with the required subnets, routing, and security groups. This VPC will be used for the EKS cluster's networking.

- **Control Plane Provisioning**: Control plane is a master node. The command provisions the EKS control plane, which manages the cluster's resources, networking, and scaling. eksctl interacts with the Amazon EKS service to create and configure the control plane components.

- **Cluster Verification**: After the cluster creation process, eksctl performs verification checks to ensure the cluster is successfully provisioned and accessible. It confirms that the nodes are running and communicating with the control plane..

What is Kubeconfig file

Kubeconfig

- The kubeconfig file is necessary for authenticating and accessing the EKS cluster.
- By default, the kubeconfig file is updated with the necessary information to connect to the newly created EKS cluster, allowing you to use tools like kubectl to interact with the cluster from your local machine..

Database Selection

How to choose between a Relational Database and a NoSQL Database?

- **Scenario 1: READ Vs WRITE Intensive Application**
- When your application is read intensive the use NOSQL db .
- If the application is Write Intensive then go for Relational DB.
- Why ? If we have to read a lot and update less, then having multiple relational DBs and having join queries to read a lot of data base will decrease the performance. So better go with Non relational DBs. Now everything will be in one document and no joins, views complex queries wool be required which decreases the readability, performance , speed and other non functional parameters of application

How to choose between a Relational Database and a NoSQL Database?

- **Scenario 2: Unstructured VS Structured Data**

- When your application is going to persist the data which is not structured, there is no guarantee on how many fields we will get and we need to store then its unstructured data. Then go for No sql DB choice as there is no restriction on columns or fields in which data has to be inserted .

- But If we are very sure the data structure wont change ever in whole life cycle of application then even relational DB is also a good choice

How to choose between a Relational Database and a NoSQL Database?

- **Scenario 3: Concurrency and atomicity is top priority**
- In applications like hotel / movie ticket booking - Several users simultaneously read and write the data.
- To prevent data inconsistency, such as the double booking of hotel rooms, we must coordinate such transactions.
- Transactional access to the data in a relational database manages concurrency.
- A transaction is regarded as an atomic operation, making it possible to roll back or commit a transaction after a successful execution in error handling.
- To facilitate this feature we need Relational Dbs .

How to choose between a Relational Database and a NoSQL Database?

- **Scenario 4: Cost associated**
- If we are allotted a very less budget for the project, go for open source NO Sql db as many NoSQL databases are open source and freely available..
- While with Relational DBs, these are not open source usually n we need to purchase the License for it.

How to choose between a Relational Database and a NoSQL Database?

- **Scenario 5: Redundancy tolerance**
- Eliminating data redundancy is one of the relational database's main benefits.
- One table contains information about a particular entity, whereas other tables connected by foreign keys include information pertinent to that entity.

What is AWS RDS

How to configure DB in AWS

Why AWS RDS

- We can Deploy Our DB also as a Deployment / POD
- But what if POD crashes?
- Your data is lost and that's the task of DB - to manage data
- So now u have following options
    - Deployments with Persistent volumes - but not recommended bcz its for stateless apps
    - Stateful set with PV - difficult to manage n create
    - Best way is to segregate it completely outside the cluster
    - So use AWS RDS

What is AWS RDS

- AWS RDS (Amazon Relational Database Service) is a fully managed database service provided by Amazon Web Services (AWS). It enables you to set up, operate, and scale relational databases in the cloud easily.

- By leveraging AWS RDS, you can **quickly provision database instances**, scale resources up or down as needed, and easily manage your databases using the AWS Management Console, command-line interface (CLI), or APIs. This allows you to focus on building applications without worrying about the underlying infrastructure and maintenance of your databases.

- With AWS RDS, we have the option to choose from various popular relational database engines including
  - **MySQL**
  - PostgreSQL
  - MariaDB
  - Oracle
  - Microsoft SQL Server

How to setup AWS RDS

- **Sign in to the AWS Management Console**

- **Navigate to the RDS service**: Once you're logged in, search for "RDS" in the AWS Management Console search bar or locate the "Database" category and click on "RDS."

- **Choose a database engine**: On the RDS dashboard, click on the "Create database" button. You'll be prompted to choose a database engine. AWS RDS supports various engines like MySQL, PostgreSQL, Oracle, SQL Server, etc. Select the engine that suits your requirements and click on it.

- **Select the appropriate database edition**: Next, choose the edition and version of the database engine you want to use. The available options will vary depending on the database engine you selected.

- **Specify database details**: Provide the necessary information for your RDS instance, such as the DB instance identifier, username, password, and database name. You can also choose the instance size, allocated storage, and other configuration options according to your needs.

How to setup AWS RDS

- **Configure additional settings**: Set up the remaining configuration options, such as the Virtual Private Cloud (VPC) and security groups for network access, database port number, backup settings, maintenance preferences, etc. Adjust these settings based on your requirements.

- **Review and launch:** Once you've configured all the necessary settings, review your selections and make sure everything is accurate. Double-check the cost implications and ensure you understand the pricing associated with your RDS instance. When you're ready, click on the "Create database" or "Launch database" button.

- **Wait for the creation process:** The RDS instance creation process may take a few minutes. You can monitor the progress on the RDS dashboard or check the status notifications.

- **Access and manage your RDS instance:** Once the RDS instance is created successfully, you can access it using various methods like connecting through an application, using a database management tool, or connecting via the AWS Command Line Interface (CLI). The specific steps for accessing and managing your RDS instance will depend on the database engine you chose.

How to setup AWS RDS

- The need for an EC2 connection would typically arise if you have specific networking requirements, such as establishing a private, secure connection between your EC2 instances and the RDS database instance without exposing it to the public internet. In such cases, you would configure the necessary networking settings, security groups, and subnets to establish the EC2-to-RDS connection.

What is Mongodb Atlas

How to configure NOSQL /DB in AWS

What is Mongo DB Atlas

- MongoDB Atlas is a fully managed cloud database service provided by MongoDB
- It allows you to deploy, manage, and scale MongoDB databases on the cloud infrastructure of your choice, such as Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure.
- MongoDB Atlas simplifies the process of deploying and managing MongoDB databases, allowing you to focus on building your applications

What is handled By Mongo DB team and what is handled by AWS/Azure Cloud team

- Tasks such as database setup, configuration, patching, backups, and scaling are handled by the MongoDB team

- While AWS and Azure provide the cloud infrastructure where MongoDB Atlas runs

- AWS and Azure handle the underlying **infrastructure, networking, and virtual machine management,** while MongoDB Atlas focuses on **managing the MongoDB database software and related services**. The responsibility for managing and maintaining the database service lies with the MongoDB team

- **MongoDB Atlas leverages Amazon Elastic Compute Cloud (EC2) instances as part of its infrastructure on AWS**. When you create a MongoDB Atlas cluster on AWS, MongoDB Atlas provisions EC2 instances to **host the MongoDB nodes**. These instances are responsible for running the MongoDB database software and storing the data.it abstracts the complexities of managing EC2 instances away from the users.

How to setup MongoDB Atlas

- Sign up and log in: Go to the MongoDB Atlas website https://www.mongodb.com/cloud/atlas and click on the "Try Free" or "Get Started Free" button. Create a new MongoDB Atlas account or log in using your existing MongoDB account.

- Create a new project: Once you're logged in, click on the "Create a New Project" button and provide a name for your project. The project acts as a container for your MongoDB clusters and related resources.

- Build a cluster: Inside your project, click on the "Build a Cluster" button to create a new MongoDB cluster. A cluster is a set of interconnected MongoDB instances that store your data.

- Whitelist IP addresses: By default, your cluster only allows connections from IP addresses on the Atlas IP whitelist. Go to the "Network Access" tab and click on the "Add IP Address" button to add your IP address or IP range to the whitelist. Alternatively, you can allow access from anywhere (0.0.0.0/0), but this is not recommended for production environments.

- Configure cluster settings , Wait for cluster creation n Connect to your cluster

Profiling in Spring Boot

What is Spring Boot profiling

- Spring Boot profiling is a feature that allows developers to configure and customize their application's behavior based on different runtime environments

- By using Spring Boot profiling, you can define different sets of properties, configurations, and dependencies for each profile. This enables you to have separate configurations for different environments, such as using an in-memory database for development and a production database for the production environment.

How profiling works in Spring Boot

- **Configuration Files**: Spring Boot allows you to define multiple configuration files, each associated with a specific profile. These files are typically named **application-{profile}.properties** or **application-{profile}.yml**, where {profile} is the name of the profile.

- **Application Properties:** Inside **each profile-specific configuration file, you can specify different properties based on your requirements**. These properties can include database connection details, logging settings, cache configurations, and any other application-specific configurations.

- **Activating Profiles:** You can activate a specific profile by setting the **spring.profiles.active** property. This can be done in various ways, such as by **setting an environment variable, using the -D flag when running the application,** or programmatically in code.
  For example, to activate the development profile, you can set spring.profiles.active=development.

Why are we using Spring Boot Profiling ?

- We are using profiling for Eureka configurations.
- Now the service URL for eureka in cloud cluster will be **http://eureka-0.eureka-service.default.svc.cluster.local:8761/eureka**
- Rather than http://localhost:8761/eureka/

Why are we using Spring Boot Profiling ?

- We are using profiling for Eureka configurations.
- Now the service URL for eureka in cloud cluster will be **http://eureka-0.eureka-service.default.svc.cluster.local:8761/eureka**
- Rather than http://localhost:8761/eureka/
- Also we will change eureka dashboard URl to http://localhost:8761 to http://localhost:8761/eurekaDashboard

What does new URL mean ?

- Now the service URL for eureka in cloud cluster will be http://eureka-0.eureka-service.default.svc.cluster.local:8761/eureka

- **http://** This specifies the protocol used for communication, in this case, HTTP.

- **eureka-0**: This is the hostname of the Eureka server instance within the Kubernetes cluster. The "0" in the hostname suggests that it might be the first instance of the Eureka server. The specific naming convention may vary depending on the deployment configuration.

- **eureka-service**: This is the name of the Kubernetes service associated with the Eureka server. Kubernetes services provide a stable endpoint for accessing pods within the cluster.

- **default**: This is the namespace in which the Eureka service and pods are deployed. Kubernetes namespaces provide a way to isolate resources within a cluster.

- **svc.cluster.local**: This is the domain suffix for Kubernetes services within the cluster.

What does new URL mean ?

- **8761**: This is the port number on which the Eureka server is listening for incoming requests. By default, Eureka uses port 8761 for its HTTP-based communication.

- **/eureka**: This is the path or endpoint within the Eureka server. In this case, it refers to the Eureka server's own API endpoint. Typically, this endpoint is used by Eureka clients (services) to register themselves and discover other services.

- In summary, it **represents the Eureka server's endpoint within a Kubernetes cluster**, where services can register and discover other services using the Eureka service registry.

Kubernetes manifest files

What is kubernetes manifest file

- A Kubernetes manifest file is like a blueprint that tells Kubernetes how to set up and manage different parts of your application or system. It's written in a format called YAML or JSON,

- A Kubernetes manifest file typically includes definitions for various Kubernetes objects, such as deployments, services, pods, replica sets, config maps, secret and more. These objects represent the building blocks of your application or system and define how they should be created, configured, and orchestrated within the Kubernetes cluster.

Why kubernetes files is needed ?

- **Declarative Configuration:** Manifest files describe the desired state of your application and infrastructure in a Kubernetes cluster.
- **Infrastructure as Code:** Manifest files allow you to treat infrastructure configuration as code, enabling version control, review, and collaboration.
- **Automation and Orchestration:** Kubernetes uses manifest files to automate and orchestrate deployment, scaling, updating, and deletion of resources.
- **Reproducibility and Portability:** Manifest files capture all necessary configurations, making it easier to reproduce environments across different clusters and providers.
- **Collaboration and Sharing:** Manifest files serve as a common language for teams, facilitating sharing, collaboration, and best practice adoption.
- **Resource Management:** Manifest files define and manage various Kubernetes resources, specifying resource requirements, networking, storage, and other parameters.

Manifest file creation

Deployment manifest file

- **apiVersion: apps/v1**: Specifies the API version of the Kubernetes object being defined (Deployment).
- **kind: Deployment**: Indicates that the object being defined is a Deployment.
- **metadata**: Contains metadata about the Deployment, such as name and labels.
  - **name: restaurantapp**: Specifies the name of the Deployment.
  - **labels**: Defines labels for the Deployment.
- **spec**: Describes the desired state and configuration for the Deployment.
  - **replicas: 2**: Specifies 2 replicas (instances) for the Deployment.
  - **selector**: Specifies how replicas are selected.
    - **matchLabels**: Selects replicas based on the label "app: restaurantapp".

Deployment manifest file

- **template**: Defines the Pod template for creating replicas.
  - **metadata**: Contains metadata for the Pod template.
    - **labels**: Specifies labels for the Pods created by the Deployment.
  - **spec**: Defines the specification for the Pods.
    - **containers**: Specifies the containers within the Pods.
      - **name: restaurantapp**: Sets the name of the container.
      - **image: codedecode25/restaurant:latest**: Specifies the container image to be used.
      - **imagePullPolicy: Always**: Specifies that the container image should always be pulled.
      - **ports**: Specifies the ports to be opened for the container.
        - **containerPort: 9090**: Exposes port 9090 within the container.

Deployment manifest file

- **env**: Sets environment variables for the container.
  - **name: SPRING_DATASOURCE_USERNAME**: Specifies the name of the environment variable.
  - **valueFrom**: Defines how the value of the environment variable is derived.
    - **secretKeyRef**: Indicates that the value is retrieved from a Secret.
      - **name: mysql-secret**: Specifies the name of the Secret.
      - **key: mysql-username**: Specifies the key within the Secret to retrieve the value from.

Service manifest file

- `apiVersion: v1`: Specifies the API version of the Kubernetes object being defined, in this case, a Service.
- `kind: Service`: Indicates that the object being defined is a Service.
- `metadata`: Contains metadata about the Service, such as its name.
  - `name: restaurant-service`: Specifies the name of the Service.
- `spec`: Describes the desired state and configuration for the Service.
  - `ports`: Specifies the ports that the Service should listen on.
    - `protocol: TCP`: Specifies the protocol for the port.
    - `port: 9090`: Exposes port 9090 on the Service.
    - `targetPort: 9090`: Routes incoming traffic on the specified port to the Pods' port 9090.

Service manifest file

- `selector`: Specifies the Pods that the Service should route traffic to, based on their labels.
  - `app: restaurantapp`: Selects Pods with the label `app: restaurantapp`. This means that any Pods with the label `app: restaurantapp` will be associated with this Service and traffic will be directed to them.

Configmap manifest file

- `apiVersion: v1`: Specifies the API version of the Kubernetes object being defined, in this case, a ConfigMap.
- `kind: ConfigMap`: Indicates that the object being defined is a ConfigMap.
- `metadata`: Contains metadata about the ConfigMap, such as its name.
  - `name: mysql-configmap`: Specifies the name of the ConfigMap.
- `data`: Defines the key-value pairs of data that make up the ConfigMap.
  - `restaurantdb_url: jdbc:mysql://database-1.cnuctsf5lv7o.ap-northeast-1.rds.amazonaws.com:3306/restaurantdb`: Defines a key-value pair where the key is `restaurantdb_url` and the value is `jdbc:mysql://database-1.cnuctsf5lv7o.ap-northeast-1.rds.amazonaws.com:3306/restaurantdb`. This represents the URL of the restaurant database.

Configmap manifest file

- `restaurantdbname: restaurantdb`: Defines a key-value pair where the key is `restaurantdbname` and the value is `restaurantdb`. This specifies the name of the restaurant database.
- `foodcataloguedb_url: jdbc:mysql://database-1.cnuctsf5lv7o.ap-northeast-1.rds.amazonaws.com:3306/foodcataloguedb`: Defines a key-value pair where the key is `foodcataloguedb_url` and the value is `jdbc:mysql://database-1.cnuctsf5lv7o.ap-northeast-1.rds.amazonaws.com:3306/foodcataloguedb`. This represents the URL of the food catalog database.
- `foodcataloguedbname: foodcataloguedb`: Defines a key-value pair where the key is `foodcataloguedbname` and the value is `foodcataloguedb`. This specifies the name of the food catalog database.

Secret manifest file

- `apiVersion: v1`: Specifies the API version of the Kubernetes object being defined, in this case, a Secret.
- `kind: Secret`: Indicates that the object being defined is a Secret.
- `metadata`: Contains metadata about the Secret, such as its name.
  - `name: mysql-secret`: Specifies the name of the Secret.
- `type: Opaque`: Specifies the type of the Secret. In this case, it is set to "Opaque" which means the Secret's data is base64-encoded and does not have a specific format.

Secret manifest file

- `data`: Defines the key-value pairs of data that make up the Secret. The values are encoded using base64.
  - `mysql-username: YWRtaW4=`: Defines a key-value pair where the key is `mysql-username` and the value is `YWRtaW4=`. The value is base64-encoded and represents the MySQL username. To use the actual value, you would need to decode it from base64.
  - `mysql-password: Y29kZWRlY29kZQ==`: Defines a key-value pair where the key is `mysql-password` and the value is `Y29kZWRlY29kZQ==`. The value is base64-encoded and represents the MySQL password. Similar to the username, you would need to decode it from base64 to use the actual value.

Kubectl commands after manifest file creation

Important Kubectl commands

- `kubectl apply -f .`: Applies Kubernetes resource configurations (from YAML or JSON files) in the current directory to create or update resources in the cluster.
- `kubectl get pods`: Lists all the running pods in the cluster along with their current status.
- `kubectl get svc`: Lists all the services in the cluster along with their details.
- `kubectl log <pod-name>`: Displays the logs of a specific pod identified by its name.
- `kubectl get configmap`: Lists all the config maps in the cluster along with their details.
- `kubectl get secret`: Lists all the secrets in the cluster along with their details.
- `kubectl get deployment`: Lists all the deployments in the cluster along with their details.

Important Kubectl commands

- `kubectl exec <pod-name> -- env`: Executes the command 'env' inside a specific pod to display its environment variables.
- `kubectl delete deployment <deployment-name>`: Deletes a specific deployment identified by its name.
- `kubectl delete svc <svc-name>`: Deletes a specific service identified by its name.

Load Balancer and AWS Load Balancer

What is Load Balancer ?

A load balancer is a networking device or software that distributes incoming network traffic across multiple servers, resources, or applications. It acts as a central point of contact for client requests and intelligently directs the traffic to the appropriate server or resource based on predefined rules or algorithms.

The primary purpose of a load balancer is to evenly distribute the workload across multiple servers, optimizing resource utilization and improving the performance and availability of applications or services. By spreading the traffic across multiple servers, a load balancer can handle a large number of concurrent requests, prevent any single server from being overwhelmed, and ensure that the system can scale horizontally to accommodate increasing demand.

Types of Load Balancer

- **Classic Load Balancer (CLB):** A layer 4 load balancer that distributes traffic based on IP addresses and port numbers. It is suitable for non-HTTP(S) , TCP and UDP traffic and provides both internal and external load balancing.

- **Network Load Balancer (NLB):** A high-throughput, low-latency layer 4 n layer 7 load balancer designed for demanding workloads. It supports millions of requests per second and offers both internal and external load balancing.

- **Application Load Balancer (ALB):** A layer 7 load balancer that operates at the HTTP/HTTPS level. It can route traffic based on HTTP-specific attributes like URL paths and headers. ALB provides advanced features such as content-based routing and SSL/TLS termination.

- There are 7 layers in osi model : Physical Layer, Data Link Layer, Network Layer, Transport Layer, Session Layer, Presentation Layer, Application Layer.

AWS Load Balancer

In Amazon Elastic Kubernetes Service (EKS), an AWS Load Balancer is a service that helps distribute incoming network traffic to the pods running in your EKS cluster. It acts as a gateway for external traffic to reach your Kubernetes services.

Key points about AWS Load Balancer in EKS:

- Acts as an entry point for external traffic to access your applications running in EKS.
- Automatically detects and manages the availability of your pods, routing traffic to healthy instances.
- Provides load balancing capabilities, distributing traffic across multiple pods for improved performance and scalability.

AWS Load Balancer

- Offers different types of load balancers, such as Classic Load Balancer (CLB), Network Load Balancer (NLB), and Application Load Balancer (ALB), depending on your requirements.
- Can be configured to route traffic based on various criteria, such as URL paths, hostnames, or TCP/UDP protocols.
- Supports SSL/TLS termination, health checks, and integration with other AWS services.
- Simplifies the management of traffic routing and load balancing for your Kubernetes services in the AWS cloud environment.

ALB Controller and Ingress

What is ALB Controller and Ingress

- In Kubernetes, "Ingress" and "ALB controller" (can be used as an Ingress Controller in Kubernetes.) are two related concepts that work together to provide external access to services running within a cluster. Here's the difference between them

- **Ingress** is an API object in Kubernetes that **defines rules for routing external traffic** to internal services. It acts as an entry point for external requests.

- An **ALB controller** is a component or a software application responsible for **implementing the rules defined in the Ingress resource.**

- Ingress allows you to define routing rules based on hostnames, paths, or other criteria to direct incoming traffic to the appropriate services within the cluster.

- ALB is a load balancer that can be used by an Ingress controller to achieve load balancing and routing functionality in an AWS environment.

ALB Controller

What is ALB Controller

- The AWS Load Balancer Controller is a tool that helps manage Elastic Load Balancers (ELBs) in an Amazon Web Services (AWS) environment for a Kubernetes cluster.

- It makes it easier to handle incoming traffic by setting up Application Load Balancers (ALBs) for external traffic (from the internet) and Network Load Balancers (NLBs) for internal traffic (within the cluster).

- By using the AWS Load Balancer Controller, you can define rules for how traffic should be directed to your applications, and the controller takes care of creating and managing the load balancers to handle the traffic effectively.

Steps to deploy ALB Controller

Steps to install ALB Controller

- First step is to create and IAM policy for this we need to download the policy using below link
- AWS GovCloud (US-East) or AWS GovCloud (US-West) AWS Regions
  https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.4.7/docs/install/iam_policy_us-gov.json
- All other AWS Regions
  https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.4.7/docs/install/iam_policy.json
- This IAM policy allows it to make calls to AWS APIs on your behalf.

- To apply this policy on your AWS account run below command
  aws iam create-policy --policy-name AWSLoadBalancerControllerIAMPolicy --policy-document file://iam_policy.json

Steps to install ALB Controller

- Below command is used to establish the required connection between the EKS cluster and IAM, enabling the ALB Controller to authenticate and perform necessary actions within the AWS environment.
  eksctl utils associate-iam-oidc-provider --region=ap-southeast-2 --cluster=codedecode-eks --approve

- When deploying the AWS Load Balancer Controller (ALB Controller) on an Amazon EKS cluster, it requires an IAM OIDC provider to authenticate and authorize actions with AWS services. The ALB Controller uses IAM roles and policies to interact with other AWS services, such as creating and managing Application Load Balancers (ALBs). By associating the IAM OIDC provider with the EKS cluster, it allows the ALB Controller to assume IAM roles and access the necessary resources.

Steps to install ALB Controller

- Create an IAM role. Create a Kubernetes service account named aws-load-balancer-controller in the kube-system namespace for the AWS Load Balancer Controller and annotate the Kubernetes service account with the name of the IAM role.
- You can use eksctl or the AWS CLI and kubectl to create the IAM role and Kubernetes service account.
  eksctl create iamserviceaccount --cluster=my-cluster --namespace=kube-system --name=aws-load-balancer-controller --role-name AmazonEKSLoadBalancerControllerRole --attach-policy-arn=arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy --approve
- Replace my-cluster with the name of your cluster, 111122223333 with your account ID, and then run the command. If your cluster is in the AWS GovCloud (US-East) or AWS GovCloud (US-West) AWS Regions, then replace arn:aws: with arn:aws-us-gov:.

Steps to install ALB Controller

- Install cert-manager using one of the following methods to inject certificate configuration into the webhooks.
  kubectl apply --validate=false -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/cert-manager.yaml

- By applying this manifest file, cert-manager resources, such as Custom Resource Definitions (CRDs) and controllers, will be created in the cluster. These resources enable the management and automation of SSL/TLS certificates within the Kubernetes environment.

Steps to install ALB Controller

- Download the controller specification using below url https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.4.7/v2_4_7_full.yaml

- If you downloaded the v2_4_7_full.yaml file, run the following command to remove the ServiceAccount section in the manifest.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
---
```

Steps to install ALB Controller

- Now Edit Deployment file using below configurations

```
spec:
      containers:
        - args:
            - --cluster-name=your-cluster-name
            - --ingress-class=alb
            - --aws-vpc-id=vpc-xxxxxxxx
            - --aws-region=region-code
```

- Change image name

```
codedecode25/aws-load-balancer:v2.4.7
```

Steps to install ALB Controller

- Now, apply the YAML file located at the specified file location.

  kubectl apply -f v2_4_7_full.yaml

Ingress manifest file

Ingress manifest file components

- `apiVersion: networking.k8s.io/v1`: Specifies the API version of the Kubernetes networking resource being defined, which is Ingress in this case.
- `kind: Ingress`: Indicates that the object being defined is an Ingress resource.
- `metadata`: Contains metadata about the Ingress, such as its name and annotations.
  - `name: test3`: Specifies the name of the Ingress.
  - `annotations`: Provides additional configuration options for the Ingress.
    - `kubernetes.io/ingress.class: alb`: Specifies that the Ingress should be associated with the ALB Ingress Controller.
    - `alb.ingress.kubernetes.io/scheme: internet-facing`: Specifies that the ALB should be internet-facing.
    - `alb.ingress.kubernetes.io/target-type: ip`: Specifies that the target type for routing should be IP-based.

Ingress manifest file components

- `spec`: Describes the desired state and configuration for the Ingress.
  - `rules`: Specifies the routing rules for incoming traffic.
    - Each rule defines an HTTP path and the corresponding backend service to forward the traffic to.
    - For example, the path "/" will be routed to the service named "angular-service" on port 80.
    - Similarly, paths "/restaurant", "/fooditems", and "/order" are defined with their respective backend services and ports.

Junits for Spring boot App

What is Junit

- JUnit is a testing framework
- It helps developers test their code to make sure it works correctly.
- Unit Testing: JUnit allows developers to test individual parts of their code to ensure they work as intended.
- Code Quality: JUnit helps improve code quality by catching bugs early and ensuring reliability.
- Refactoring: JUnit provides a safety net when making changes to existing code, making sure nothing breaks.

Best Practices while writing Junits

- **Test Naming Convention**
- **Test Structure**: Follow the **Arrange-Act-Assert (**AAA) pattern in your tests. First, set up the necessary test data and objects (Arrange). Then, perform the action being tested (Act). Finally, verify the expected results (Assert).
- **Test Independence:** Each test should be independent and not rely on the state or outcome of other tests. This ensures that tests can be executed individually without interference.
- **Test Coverage**
- **Test Data:** Use appropriate and representative test data to cover a wide range of scenarios.
- **Test Assertions:** Use relevant assertions to verify expected results. JUnit provides various assertion methods (e.g.,assertEquals(), assertTrue()).

What is Mockito in JUnit?

- Mockito is a Java-based mocking framework used for unit testing of Java application. Its used in conjunction with other testing frameworks such as JUnit and TestNG. It internally uses Java Reflection API in order to create mock objects for a given interface. Mock objects are nothing but proxy for actual implementations.

- To add it to your project add :

```
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-junit-jupiter</artifactId>
    <version>2.19.0</version>
   <scope>test</scope>
</dependency>
```

What is Mocking?

- Mocking is a process of developing the objects that act as the mock or clone of the real objects. In other words, mocking is a testing technique where mock objects are used instead of real objects for testing purposes. Mock objects provide a specific (dummy) output for a particular (dummy) input passed to it.

- Mocking for unit testing is when you create an object that implements the behavior of a real subsystem in controlled ways. In short, mocks are used as a replacement for a dependency.

- To mock objects, you need to understand these key concepts of mocking, i.e., stub, fake, spy, dummy and mock. Some of the unit tests involve only stubs, whereas some involve fake, and mocks.

- A method invoked using mocked reference does not execute the actual method body defined in the class file, rather the method behavior is configured using when(...).thenReturn(...) methods.

What is @Mock and @InjectMocks?

- In mockito-based junit tests, @Mock annotation creates mocks and @InjectMocks creates actual objects and injects mocked dependencies into it.
- We must define the when(...).thenReturn(...) methods for mock objects whose class methods will be invoked during actual test execution.
- So In a junit test, we create objects for the class which need to be tested and its methods to be invoked using inject mock
- We create mocks for the dependencies which will not be present in the test environment and objects are dependent on it to complete the method call.

Why do you need Mocking at first place?

Common targets for mocking are:

- Database connections - Sometimes the database queries can take 10, 20, or more seconds to execute.
- Classes that are slow,
- Classes with side effects like who sends email to clients.
- If we want to test a component that depends on the other component, but it is under development. Without mocking, we need to wait for the completion of the required elements for testing.
- Web services / External resource. Connecting to an external resource can take too much time. And can be down which impacts our testing. Our CI/CD pipeline may break bcz of another web service

Continuous Integration
Continuous Deployment

What is Continuous Integration (CI) ?

- It is a software development practice where developers frequently and automatically integrate their code changes into a shared repository. The main purpose of CI is to catch and resolve integration issues early in the development process, promoting collaboration, faster feedback, and code readiness for deployment.

- Scenario: A team of developers is working on a web application project with multiple features and modules. Each developer is assigned to work on a different module. They need to collaborate and ensure that their code integrates smoothly and doesn't break the application:

- With CI:

- Each developer commits their changes to the shared repository frequently.

- The CI server automatically triggers a build process whenever a new commit is detected.

What is Continuous Integration (CI) ?

- The build process compiles the code, runs automated tests, and performs static code analysis.
- If any issues are found, such as failing tests or code quality violations, the CI server immediately notifies the developers.
- Developers can quickly identify the cause of the issue and fix it since they committed their changes recently.
- Once the issues are resolved, the CI server rebuilds and retests the application, ensuring that the integration is successful.
- The team can confidently release the application, knowing that it has gone through a rigorous integration process.

What is Continuous Deployment (CD) ?

Continuous Deployment (CD) is a software development practice that involves automatically deploying code changes to production environments after passing through the CI process. Here are the key points explaining the concept and benefits of continuous deployment:

- CD helps developers in the following ways:

- **Faster Time to Market**: CD automates the deployment process, allowing developers to release their code changes more frequently. This reduces the time between feature development and deployment, enabling faster delivery of new features and bug fixes to end-users.

- **Reduced Deployment Errors:** CD ensures that deployments are performed consistently and in a standardized manner. By automating the process, it minimizes the risk of human errors that can occur during manual deployments. This leads to more reliable and error-free deployments.

What is Continuous Deployment (CD) ?

- **Rapid Feedback Loop:** CD facilitates a shorter feedback loop by automatically deploying changes to production environments. This allows developers to gather feedback from users and stakeholders more quickly, enabling them to iterate on the software and make necessary improvements faster.

- **Continuous Integration with Deployment**: CD works hand in hand with Continuous Integration (CI) by automating the deployment of code changes that have passed the CI process. This ensures that tested and validated code is promptly deployed, maintaining a seamless workflow from development to production.

- **Increased Efficiency:** CD streamlines the deployment process by automating repetitive and manual tasks. Developers can focus more on writing code and implementing new features instead of spending time on complex deployment procedures. This leads to increased productivity and efficiency within the development team.

Sonarqube

What is Sonarqube

SonarQube is a popular static code analysis tool used in software development projects, including Spring Boot Java projects. It helps developers improve the quality of their code by identifying potential bugs, security vulnerabilities, and code smells (code that is poorly written or structured). SonarQube provides detailed feedback and metrics to help developers understand the state of their codebase and make necessary improvements.

- Analyzes code quality: It checks your code against predefined rules to identify bugs, security vulnerabilities, and poor code structure.
- Improves code maintainability: It helps find code smells that make code harder to maintain, allowing you to write cleaner and more understandable code.
- Enhances security: It identifies common security risks like SQL injection and XSS attacks, making your application more secure.
- Supports CI/CD: It seamlessly integrates with your development pipeline to analyze code on each build, catching issues early on.
- Facilitates team collaboration: It provides a centralized platform for code analysis, enabling team members to share insights and make informed decisions.

Install Sonar as docker container

- Make sure docker is up and running
- Run the below command
  docker run -d -p 9000:9000 --name sonarqube sonarqube
- Wait for SonarQube to start: It may take a few minutes for SonarQube to start up fully. You can check the container logs with the following command to verify:
  docker logs -f sonarqube
- Access SonarQube web interface: Open your web browser and navigate to http://localhost:9000 to access the SonarQube web interface. You should see the SonarQube dashboard.

Sonar dependencies in Microservices

Add below dependencies in Microservices

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-compiler-plugin</artifactId>
   <configuration>
      <source>1.8</source>
      <target>1.8</target>
      <annotationProcessorPaths>
         <path>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.26</version>
         </path>
         <path>
            <groupId>org.mapstruct</groupId>
            <artifactId>mapstruct-processor</artifactId>
            <version>1.4.2.Final</version>
         </path>
      </annotationProcessorPaths>
   </configuration>
</plugin>
<plugin>
   <groupId>org.sonarsource.scanner.maven</groupId>
   <artifactId>sonar-maven-plugin</artifactId>
   <version>3.8.0.2131</version>
</plugin>
```

Add below dependencies in Microservices

```
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.8.8</version>
    <executions>
        <execution>
            <id>prepare-agent</id>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
        </execution>
        <execution>
            <id>report</id>
            <goals>
                <goal>report</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

Also add below exclusion in properties

```
<sonar.exclusions>**/com/codeddecode/restaurantlisting/RestaurantListingApplication.java,
**/com/codeddecode/restaurantlisting/repo/**,
**/com/codeddecode/restaurantlisting/dto/** ,
**/*/com/codeddecode/restaurantlisting/entity/**/*
</sonar.exclusions>
```

Sonar Coverage maven command

- mvn clean org.jacoco:jacoco-maven-plugin:prepare-agent install sonar:sonar -Dsonar.host.url=http://localhost:9000/ -Dsonar.login=squ_a44ef243148d9f75cb3248851df2d555d5342ee2

- clean: Cleans the project by deleting any build artifacts from previous builds.

- org.jacoco:jacoco-maven-plugin:prepare-agent: Configures the JaCoCo Maven plugin to prepare code coverage analysis.

- install: Compiles the source code, runs tests, and packages the application.

- sonar:sonar: Analyzes the project and sends the analysis report to a SonarQube server.

Sonar Coverage maven command

- The command also includes the following parameters:

- -Dsonar.host.url=http://localhost:9000/: Specifies the URL of the SonarQube server where the analysis report will be sent. In this case, it's set to http://localhost:9000/, assuming the SonarQube server is running locally on port 9000.

- -Dsonar.login=squ_a44ef243148d9f75cb3248851df2d555d5342ee2: Provides the SonarQube user token or authentication token to authorize the analysis request.

CI using Jenkins

What is Jenkins ?

Jenkins is a popular open-source automation tool that helps automate various tasks in the software development process, particularly in the context of Continuous Integration (CI). Here's a simple and concise explanation of Jenkins and its usefulness in CI:

- Automation: Jenkins automates repetitive tasks involved in building, testing, and deploying software applications.
- Continuous Integration (CI): Jenkins is widely used for CI, where it automatically builds and tests code changes as they are committed to a version control system like Git.
- Triggering Builds: Jenkins monitors the code repository and triggers builds whenever new code changes are detected, ensuring that the latest changes are integrated and tested.
- Build and Test Automation: Jenkins compiles the code, runs automated tests, and generates reports, helping identify any issues or errors in the codebase early in the development process.

What is Jenkins ?

- Feedback and Notifications: Jenkins provides immediate feedback on the build and test results, notifying developers about the status of their code changes and any errors that need attention.
- Integration with Tools: Jenkins offers a vast ecosystem of plugins that integrate with various tools, such as source code management systems, build tools, testing frameworks, and deployment platforms.
- Scalability and Flexibility: Jenkins can be scaled to handle projects of different sizes and customized to fit specific development workflows and requirements.
- Time and Efficiency: By automating repetitive tasks, Jenkins saves time for developers, allowing them to focus on writing code and addressing critical issues.
- Code Quality: Jenkins promotes code quality by running tests and providing immediate feedback, enabling developers to catch bugs early and ensure the stability of the codebase.

Jenkins File

Understanding Jenkins file

- **pipeline**: This keyword indicates the beginning of a Jenkins pipeline.
- **agent** any: Specifies that the pipeline can run on any available agent in the Jenkins environment.
- **environment**: Defines environment variables that can be used throughout the pipeline. In this case, DOCKERHUB_CREDENTIALS and VERSION are defined.
- **tools**: Allows the use of specific tools within the pipeline. Here, Maven is defined as the tool with the name "Maven".
- **stages**: Contains a collection of stages that represent the different steps of the pipeline.

Understanding Jenkins file

- **stage**('Maven Build'): Represents a stage named "Maven Build". This stage runs the Maven build process by executing the command mvn clean package -DskipTests.

- stage('Run Tests'): Executes the command mvn test to run the tests of the project.

- stage('SonarQube Analysis'): Performs SonarQube analysis on the project by executing the command mvn clean org.jacoco:jacoco-maven-plugin:prepare-agent install sonar:sonar with the appropriate SonarQube parameters.

- stage('Check code coverage'): Uses a script block to perform a code coverage check. It retrieves the code coverage percentage from SonarQube using cURL and verifies if it meets a certain coverage threshold. If the coverage is below the threshold, it throws an error.

Understanding Jenkins file

- stage('Cleanup Workspace'): Deletes the workspace directory to clean up the workspace before the next stage.

- stage('Update Image Tag in GitOps'): Checks out a Git repository, updates the image tag in a YAML file, commits the changes, and pushes them to the repository.

- Overall, this pipeline script demonstrates a typical CI/CD process that includes building, testing, code analysis, and deployment tasks.

Understanding Commands for jenkins

- **sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo:** This command uses the wget utility to download the Jenkins repository configuration file. The -O option specifies the output file path (/etc/yum.repos.d/jenkins.repo), and the URL specifies the location of the repository file to be downloaded. By saving it in the /etc/yum.repos.d/ directory, it makes the Jenkins repository accessible to the package manager (yum).

- **sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key:** **T**his command imports the GPG key used to verify the authenticity of the Jenkins packages. The rpm command is used to manage RPM packages, and the --import option specifies that the provided URL is a key file to import.

Understanding Commands for jenkins

- **sudo yum install jenkins -y:** This command uses yum to install Jenkins from the newly added repository. The jenkins package is specified as the package to be installed. The -y option automatically answers "yes" to any prompts during the installation process.

- **sudo systemctl enable jenkins:** This command enables the Jenkins service to start automatically on system boot. The systemctl command is used to manage system services, and the enable option sets the Jenkins service to be started at boot time.

- **sudo systemctl start jenkins**: This command starts the Jenkins service immediately. The start option is used with systemctl to initiate the service.

Understanding Commands for jenkins

- Overall, these commands download the Jenkins repository configuration, import the GPG key, install Jenkins using yum, enable Jenkins to start on system boot, and then start the Jenkins service. After executing these commands, Jenkins should be installed and running on your system, ready for configuration and use as an automation server.

Argo CD

What is Argo CD

- Argo CD is an open-source continuous delivery (CD) tool specifically designed for Kubernetes-based applications. It helps developers automate the deployment and management of applications in a Kubernetes cluster.

- If you have a bunch of applications, like websites or web services, that you want to run on a Kubernetes system. Instead of manually setting up and managing everything, which can be time-consuming and prone to errors, you can use Argo CD.

- With Argo CD, you can tell it how you want your applications to be set up and configured using simple files like manifest files.

What is Argo CD

- Argo CD continuously checks if everything is set up correctly based on the Manifest files. If there are any changes or updates to be made, Argo CD automatically applies them to the Kubernetes system. It ensures that your applications are always running as you want them to be, without you having to worry about it.

- Argo CD also helps you roll back to a previous version if something goes wrong.

- In simple terms, Argo CD is like having a reliable assistant that understands your application's needs and takes care of deploying and managing them in a Kubernetes system. It saves you time, reduces errors, and makes sure your applications are always running smoothly.