UNIT-2 DevOps

1. DevOps on Achieving Business Agility
2. Continuous testing in DevOps
3. Monolithic Architecture, Micro Services Architecture, Architecture Rules of thumb in DevOps
4. Separation of Concerns in Software Architecture
5. Handling Database Migrations, Resilience in DevOps
6. Impact of DevOps on Software Quality and reliability
7. Relationship between DevOps and Software Architecture.
8. How well design Architecture Supports DevOps principles.

DevOps on Achieving Business Agility

ability to move Quickly and easily.

Business Agility :-

Business agility means a Company can easily adjust to market changes by being flexible and quick to adapt.

How DevOps helps to achieve business agility :-

DevOps is about making team work between developers and operations Smoother. Automating tasks and deploying Software frequently this helps Companies reach faster to changes in market and what Customers want.

Example :-

Imagine a Software Company that ~~want~~ Used to have long waits between when developers finished their work and when it was deployed by the operations team. With DevOps they implemented automated Systems that allows developers to Send their Code Stright to deployment. this means new features and updates Can reach Customers much faster and able to adopt Quickly to Customer feedback.

Examples of Companies that have adopted DevOps to achieve faster Software delivery and increased Customer responsiveness :-

Amazon :-

Amazons DevOps practices enable them to rapidly deploy updates and new features to their e-Commerce platform amazon.in, For example, during peak Shopping Seasons like prime Day, they Smoothly handle increased traffic and transactions, ensuring a Smooth Customer experience.

Netflix :-

Netflix relies on DevOps to Continuously enhance their Streaming Service.

Google :-

Google focuses on working together, being clear about what's happening, and always trying to get better to make Sure their products make Customer happy. they Use tools that automatically check the Code for any issues and quickly deploy updates.

microsoft :-

with tools like Azure DevOps, they accelerate product development Cycles, allowing them to release updates to products like microsoft office and windows more frequently by ~~using~~ focusing on User feedback and Security.

Adobe :-

Adobe utilizes DevOps to enhance its Creative Software Suite, including photoshop and illustrator.

Continuous Testing In DevOps

In DevOps, Continuous testing means testing your Software non-Stop, from the moment you start writing it until it's ready to place in a Server. its about automating tests so they run automatically whenever there's a change, helping to catch bugs early and ensure the Software works well.

Definition and example for each type of Continuous testing:

(1) Unit Testing:-

testing each piece of Code Separately to make Sure it functions Correctly.

(2) Integration testing.

Integration testing makes Sure that all the different pieces of the Software fit together properly and work well together.

(3) Functional testing:

Functional testing ensures that the Software's features and functions behave as expected.

(4) Regression Testing:

Regression testing Confirms that new changes haven't broken existing features.

(5) Performance Testing:

Performance testing ensures that how well the Software performs under different Conditions, Such as high User loads

⑥ Security Testing:-

Security testing checks the Software for any weaknesses and make Sure that Sensitive data is protected.

challenges:-

① Automation Complexity: Automating tests across various environments and platforms can be complex and time-consuming.

② Tool integration:
Integrating testing tools with other DevOps tools can be challenging.

③ Test Data management:

Ensuring the availability and manageability of test data for various Scenarios and environments can be difficult

④ Benefits:

① Early Bug Detection
② Faster feedback Loop
③ improved Release Confidence
④ Cost and time Saving
⑤ Collaboration between development, testing and Operations teams.

Tools Commonly Used for Continuous testing in DevOps includes

1. JUnit & TestNG – for Unit testing
2. Selenium & Cypress – for functional testing
3. Jenkins & Travis CI – for Continuous testing automation & integration
4. JMeter & Gatling – for performance testing
5. OWASP ZAP & Nessus – for Security testing

Monolithic Architecture in DevOps

Monolithic architecture in DevOps refers to a traditional Software design where all Components of an application like front end, Backend and Database all these are tightly integrated into a Single Code base, which means everything is kept together with in a Single folder.

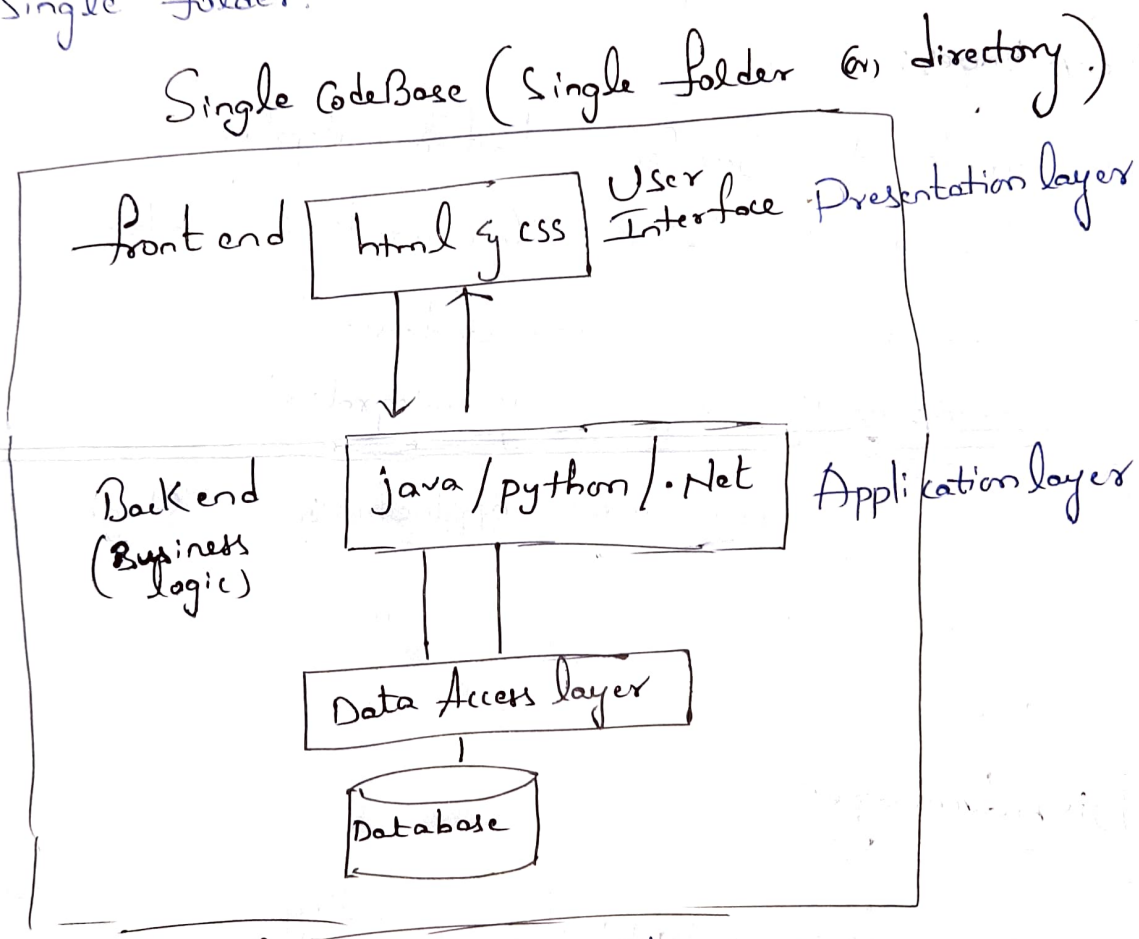

fig: Monolithic Architecture

⇒ In monolithic Architecture, User interface, Business logic and Data Access layer works as a Single unit. So, its easier to manage all operations because all Components of the Software located in one place.

⇒ In a monolithic Architecture, changes made in one layer can affect other layers because they all are tightly integrated.

For example, I created a College website. in that College website, I want to add a new feature like adding new courses. So, whenever I change the presentation layer, I need to change the business logic layer and data access layer too.

Advantages:-

1. Simple to Develop, test and deploy Since everything is in one place
2. Communication between Components is faster Since " " " "
3. Developers have a clear understanding of entire application as all Components are tightly integrated
4. Deploying monolithic applications is Stright-forward Compared to microServices Since there's only one Code base to manage.

Dis Advantages:

1. its challenging to update existing ones as the entire application needs to be modified.
2. A fault in one part of the application can bring down the entire System Since everything is tightly Coupled.
3. as everything is tightly Coupled it is difficult for teams ~~members~~ to work independently, leading to Coordination challenges.

Micro Services Architecture in DevOps

If you Consider monolithic Architecture, in monolithic Architecture entire Application is placed in one Single folder. So changes made to one Component will effect other Components. but where as in micro Services Architecture your Application is placed in bunch of folders, each folder is responsible for a Specific function. So changes made in one folder will not effect other folders.

⇒ In a DevOps environment, microServices architecture shines because it allows for better Scalability, flexibility and faster deployment.

- Scalability: you can modify Specific part of our application independently without effecting other parts.
- flexibility: Teams can choose the best tools and technologies for each microService, making it easier to adapt changing requirements
- faster deployment: Since each microService is independent, you can deploy updates without affecting whole Syste

Resilience ⟶ If one micro Service fails, it doesn't bring down the entire application. the rest can Still function normally.

these are Benefits of micro Service Architecture.

challenges :-

Complexity - managing multiple microServices can be more Complex then monolithic Architecture.

Communication: Services need to Communicate effectively, which requires good design and Coordination

monitoring and Debugging :- with more microServices, monitoring and debugging can be challenging.

Example:-

In a College website, each function like Student attendance, faculty attendance, Student results, College Courses can be considered as Separate Services in a micro Services architecture. changes made to one Service, Such as updating the Student attendance feature. won't affect the other Services like faculty attendance or Student results.

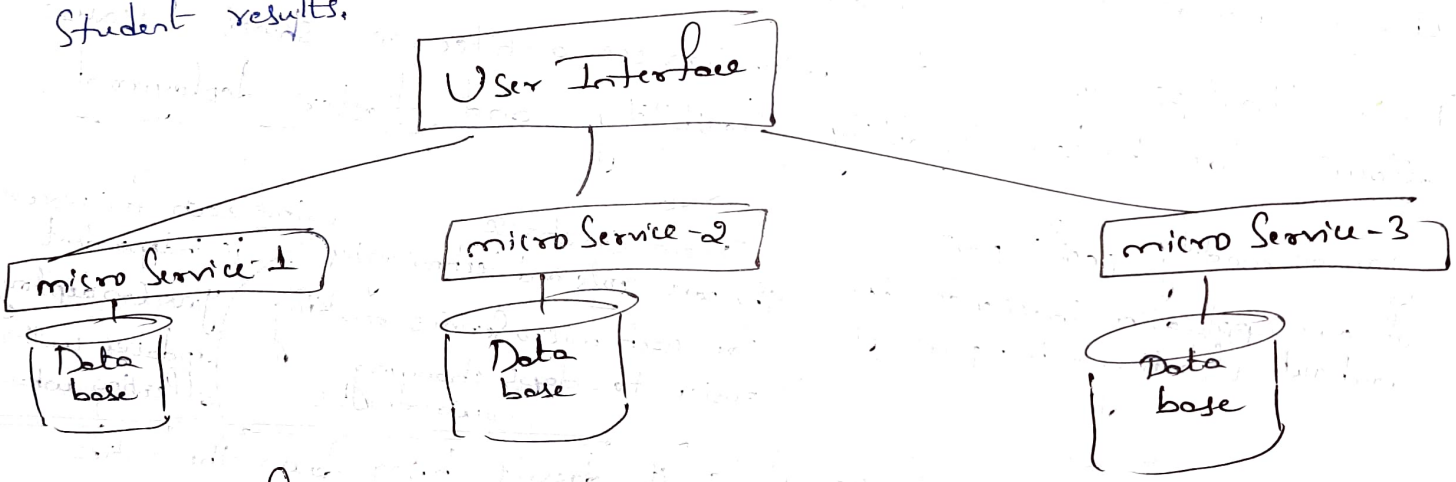

fig: micro Service Architecture

Architectural Rules of thumb in DevOps

⇒ "Rules of thumb" are general principles or guidelines widely used in a specific area. They are not strict rules, more like practical suggestions to achieve positive results in system design and implementation.

⇒ In DevOps architecture, they represent general best practices that teams can follow to achieve effective and efficient system design.

① **modularity**:- Design systems in a modular fashion (i.e., break software into smaller modules that work independently) enabling easier maintenance, updates.

② **Scalability**:- Ensure that architectures can handle increased load without sacrificing performance.

③ **Automation**:-
Automate processes wherever possible, including deployment, testing and monitoring, to increase efficiency and reduce human error.

④ **Resilience**:-
Resilience means building systems that can handle failures by including backup plans, automatic switches.

⑤ **infrastructure as Code (Iac)**:-
manage infrastructure like settings to track changes, recreate setups, and setup automatically.

(6) Continuous integration / Continuous Deployment (CI/CD) :-

automate CI/CD process by creating pipelines

(7) monitoring and Logging :-

Creating monitoring and logging Systems to ~~provide~~ check System performance, and detect issues early.

(8) Security :-

integrate Security practices into every Stage of the development and deployment process, including encryption & access Control.

(9) Containerization :-

Utilize Containerization technologies like Docker tool to package and manage applications efficiently.

(10) Feedback loop :-

Establish a feedback loop between development, Operations and Customers to Continuously improve processes and Systems based on real world Usage and feedback.

Separation of Concerns in Software Architecture

Separation of Concerns (SoC) means dividing entire program into various sections based on their functionalities, so that each section runs independently without affecting other services. changes made in one section will not affect other sections.

⇒ Separation of Concerns (SoC) will improve,

maintainability:- Easier to update and fix parts of the code without affecting the whole system

Scalability:- Simple to expand the system to handle more load and add simple to add new features

Readability:- Code is easier to understand, making it simpler for developers to work with that code.

Debugging:- Simple to identify and fix bugs

For example, I am creating a college application, my college application contains four pages, so I will divide it into four modules. the first module contains code related to the student login page, the second module contains code related to the student results page, the third module contains code related to the student attendance page, and fourth module contains code related to faculty details page. changes made in one module will not affect other modules because they all are seperated.

College Application

- 1st module: Contains code related to Student login page
- 2nd module: Contains code related to Student results page
- 3rd module: Contains code related to Student attendance page
- 4th module: Contains code related to faculty details page

all are Separated. So changes made in one module will not affect other modules.

Real world Examples of SoC implementation:-

① model-View-Controller (MVC) Architecture:-

MVC is a classic example of SoC because it divides an application into three Components →
- model :- manages data and business logic
- view :- handles the presentation layer
- Controller :- ~~intermediate~~ interface between model & view. Process User inputs

Ex:- web applications

② Micro Services Architecture:-

In a microService Architecture, an application is built as a collection of loosely Coupled Services. eachere each Service will perform Certain task.

Ex:- e-Commerce platforms.

③ Client-Server Architecture:-

this architecture Splits an application into two main Components:

client & Server → Backend that processes User requests & manages data.

client → front end that interacts with Users

Ex:- Social media applications

④ Layered Architecture: It divides an application into layers, each layer has Specific role.

Presentation layer	Application layer	Business logic layer	Data Access layer
User interface	business logic	business rules & logic	Used to Communicate

Ex:- Online banking System

Handling Database Migrations in DevOps

⇒ Database migration means transferring data from one database to another. This process can not only include just moving the data itself; but also transforming it to fit the new database structure (schema) updating database structure, and ensuring data integrity and consistency throughout the migration.

⇒ In DevOps, handling database migrations means managing the process of moving data from one database to another, while ensuring it's done smoothly and without disturbances. This is important because changes to the database structure or data need to be matched with application updates to avoid errors.

Techniques Used for database migration in DevOps:-

① Version Control:- (Git)

One of the essential techniques for database migration in DevOps is to use Version Control for both the application code and the database schema (structure). tools like Liquibase and Flyway can be integrated into CI/CD pipelines to automatically apply database schema changes when corresponding changes are made to the code.

(2) Schema migration tools :-

These tools automate making and running SQL Scripts that change your database Structure (schema). they prevent mistakes, and reduce downtime. Examples are SQL Server Data tools and mySQL workbench. (just write SQL script to Specific changes instead of writing Complete Code)

(3) Data migration tools :-

Sometimes, you need to move data between databases. Data migration tools make this easy and Safe. they Can handle big data, different formats, and keep everything in Sync. Examples are Aws Database migration Services (DMS) and Azure Data factory.

(4) Testing Strategies :-

Lastly, you need to test the migrated database to make Sure it works well. this includes Various tests, like checking performance and Security. testing ensures the databases meets all requirements and fixes any issues. Examples are SQL Test and JMeter.

Resilience in DevOps

Resilience in DevOps means the ability of Systems to recover quickly from problems and Continue working Smoothly. It ensures that Services remain available even when there are issues like failures or high traffic.

How Organizations Achieve Resilience Using DevOps :-

① **Automation:-**
Automate tasks like deployment and testing. Reduces human errors and Speeds up recovery

② **Continuous integration / Continuous Deployment (CI/CD):-**
Regularly integrate and deploy Small changes. makes it easier to find and fix issues quickly.

③ **monitoring and Logging:** → mean recording detailed information about the operations of Software
Constantly check Systems for problems. Use logs to understand issues and respond faster.

④ **Infra Structure as Code (Iac):-** → mean hardware, Software, networks, Servers, database and other Components
manage infrastructure with code. Easily rebuild or update Systems using Scripts.

⑤ **micro Services Architecture:-**
Break applications into Smaller, independent Services. If one Service fails, others Continue to work.

⑥ Regular Backups and Disaster Recovery plans:

Keep backups of data and Systems. Have plans to recover from major failures

⑦ Scalability:-

Design Systems to handle more load automatically.

Organizations Using DevOps for Building Resilient Architecture:

① Netflix:-

Uses automation CI/CD, and micro Services. "chaos Monkey" is a tool Created by Netflix to test resilience by intentionally causing failures.

② amazon:-

Employs automation, Iac and Scalability. Ensures Services are available even during high traffic periods like "Amazon Great indian festival"

③ google:-

Utilizes monitoring, logging and automated recovery. Ensures Services like Search and Gmail are highly available