ATAL FDP ON CLOUD DEVOPS
DOCKER, CONTAINER, ORCHESTRATION

Dr. J. JESU VEDHA NAYAHI
Senior Assistant Professor
Department of Computer Science and Engineering
Anna University Regional Campus - Tirunelveli

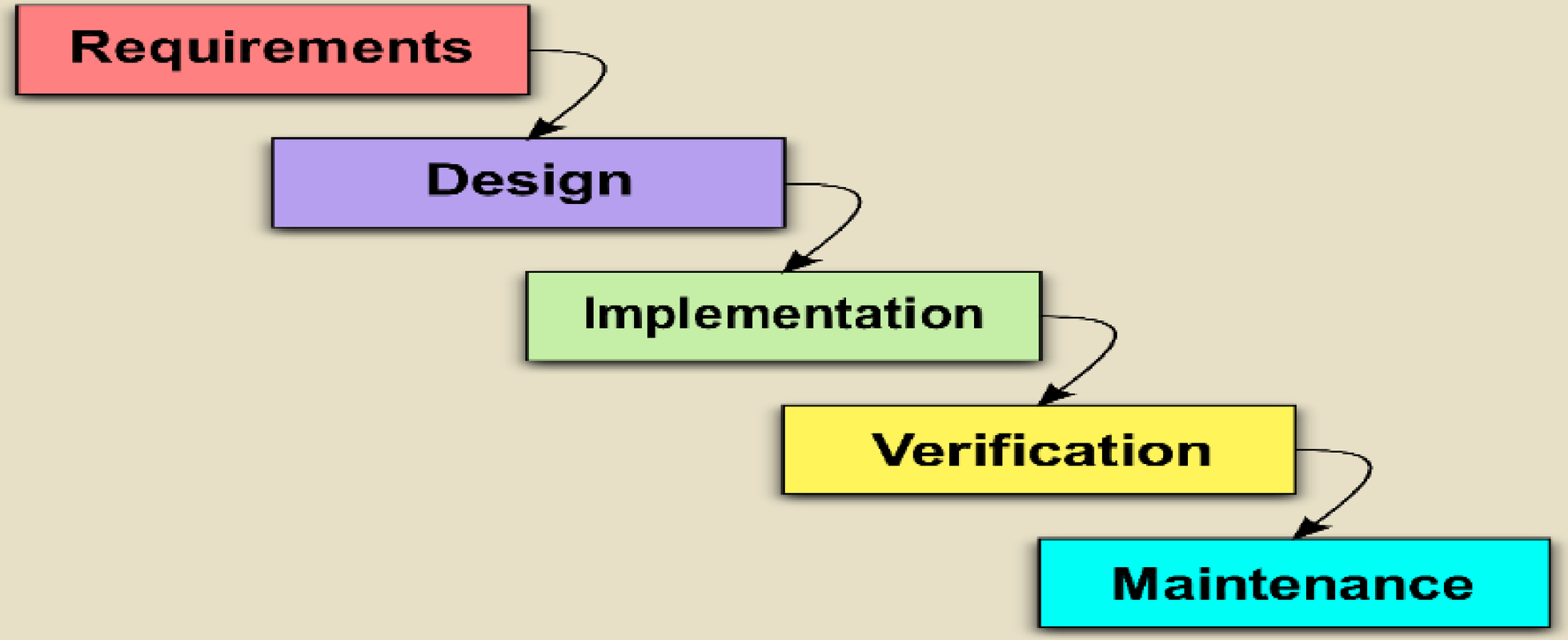
Overview

- Introduction
- DevOps
- Docker
- Container
- Orchestration
- Summary

Software Development Life Cycle

- **Planning:** Define the **requirements** and feasibility.
- **Design: Model** the solution.
- **Implementation: Code** the application.
- **Testing: Validate** the software works as intended.
- **Deployment: Release** the software to users.
- **Maintenance: Update** and fix issues.

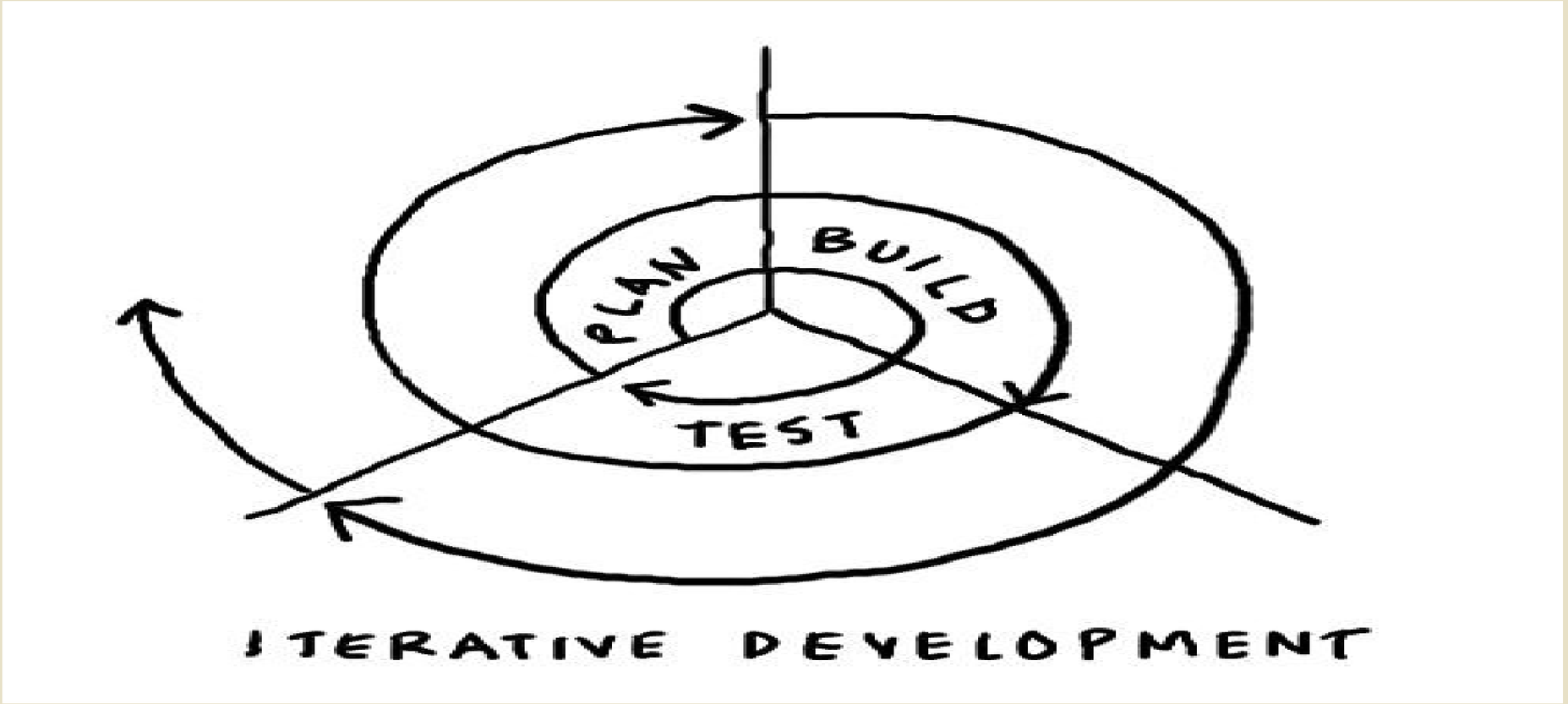
Waterfall Model



Limitations of Waterfall Model

- Longer **Delivery Time**
- Less **Flexible**
- Client **Feedback** is not collected
- **New requirements** cannot be added
- **Design flaws** will require the project to be restarted

Agile Software Development Cycle



Traditional to Modern Software Development

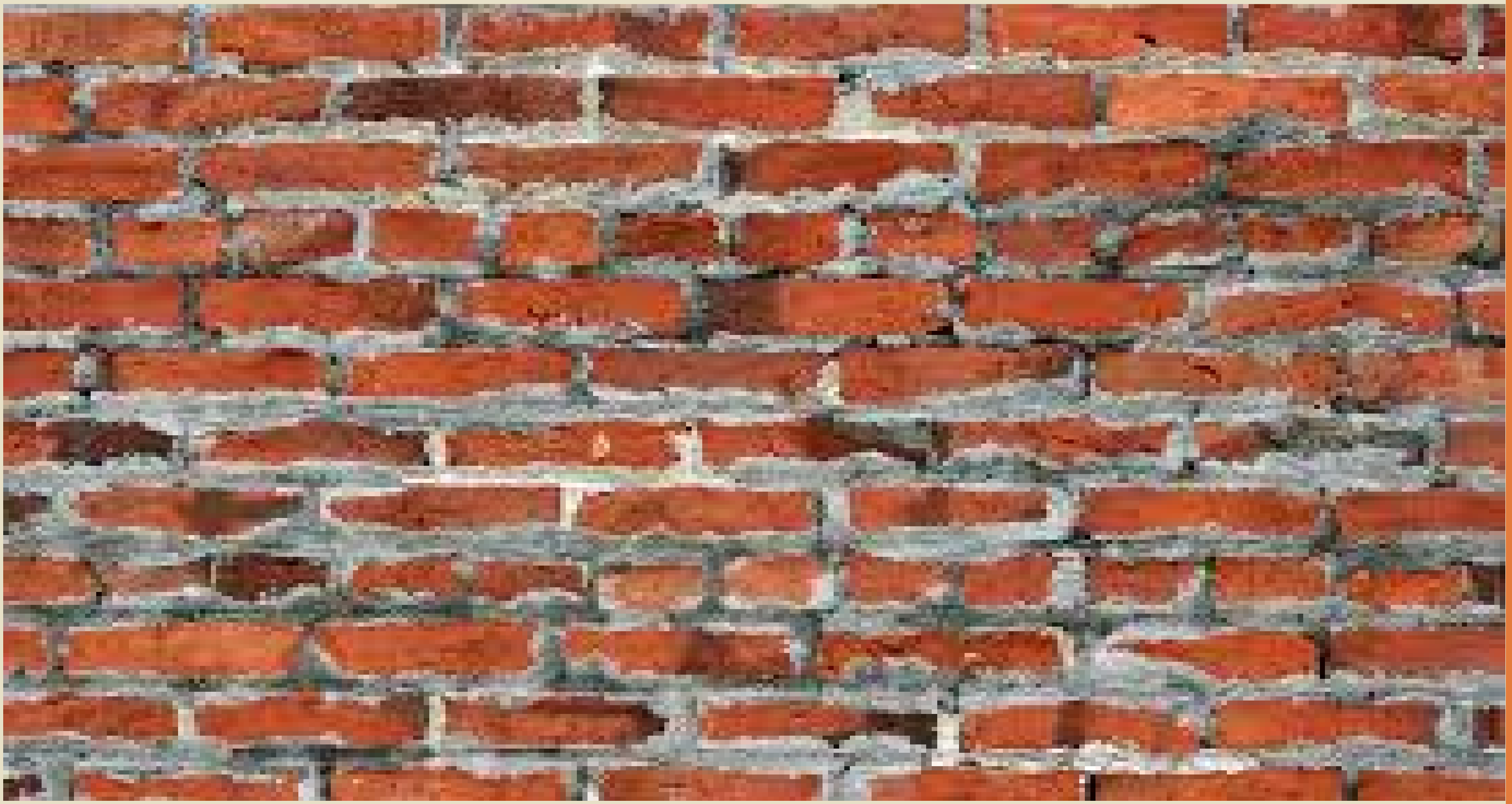
- Software Development Team Consists of two teams
  - **Developer Team** – Develops the plan, design and builds the system
  - **Operations Team** – Testing, Maintenance and deployment, provides feedback to Developers Team
- Extended **Deadlines**
- **Delayed** Software Development Life Cycle
- **Developers team moves to the other project**
- Both teams should work collaboratively
- DevOps

Developer Team

- **Writing, testing, and maintaining** the codebase.
- **Implementing features and fixing bugs.**
- **Collaborating with other developers** to integrate components and ensure the software functions as intended.
- Proficiency in programming languages, problem-solving, and debugging.

Operations Team

- Infrastructure Management
- Deployment of Software
- Feedback
- Maintenance and Support of Software Systems



DEVOPS

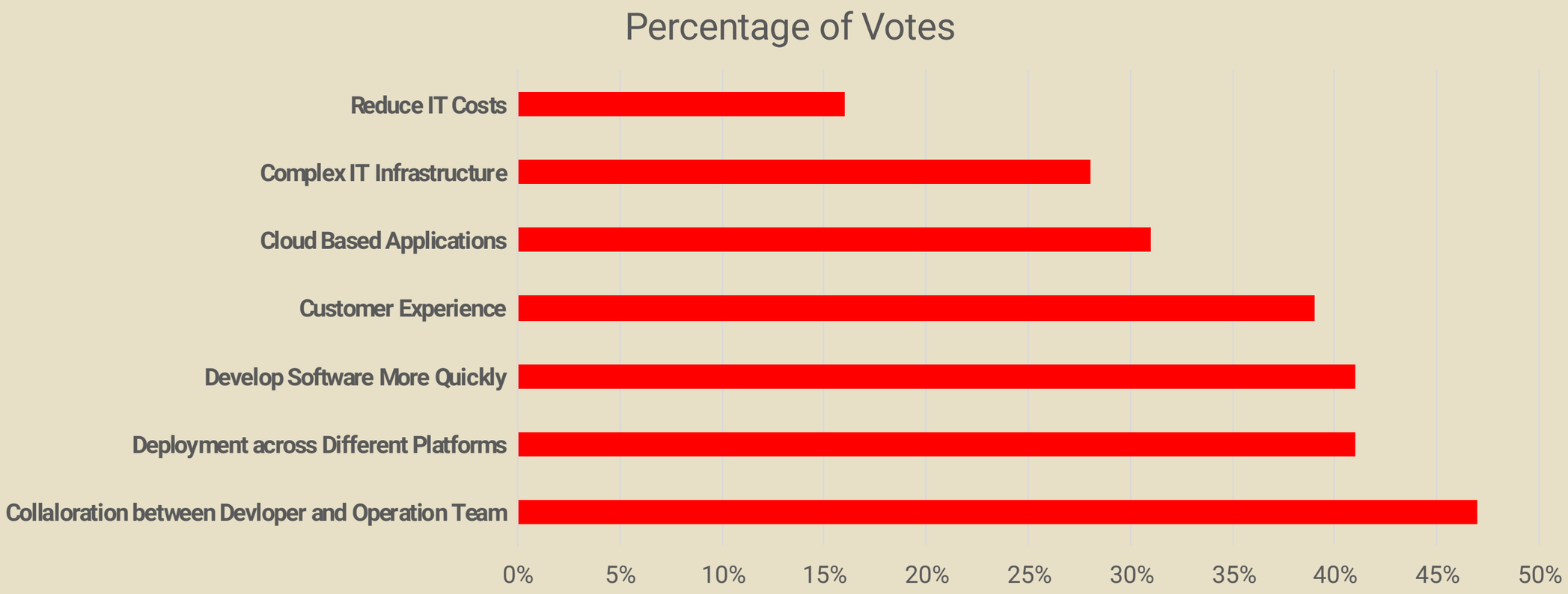
DevOps

- **Patrick Debois**, “ the father of DevOps” , coined the word “ **DevOps**” in 2009.
- Combining two words: “ **Development**” and “ **Operations**” .
- DevOps is a **collaborative way** of developing and deploying software.
- DevOps is a **software development method** that stresses **communication, collaboration and integration** between **software developers and operation professionals.**

What is DevOps?

- DevOps is a **set of practices, tools, and a cultural philosophy** that aims to **automate and integrate** the processes between software development and IT operations.
- Shorten the **software development life cycle** and provide **continuous delivery with high software quality.**
- DevOps **break down the walls between development and operations team.**
- DevOps is an approach based on **agile and lean principles** in which **business owners, development, operations, and quality assurance team collaborate** to deliver software in a continuous stable manner
- DevOps is a set of practices that provides **rapid, reliable software delivery**

Need for DevOps



DevOps - Concepts

- **Collaboration and Communication:** DevOps fosters **a culture where development (Dev) and operations (Ops) teams work closely together**, breaking down **traditional silos.**
- **Automation:** Automation is central to DevOps. Common automation tools include **Jenkins, GitLab CI/CD, and Docker.**
- **Continuous Integration (CI) and Continuous Delivery (CD):** Developers frequently **merge code changes into a central repository,** where **automated builds and tests are run**. This practice helps catch issues early in the development process. Continuous Delivery automates the delivery of code to production or other environments. This allows for quick, reliable, and consistent software deployment.
- **Infrastructure as Code (IaC):** IaC involves **managing and provisioning computing infrastructure** through machine-readable definition files rather than manual processes. Tools like **Terraform and Ansible** allow teams to automate and version-control their infrastructure, making it more consistent and reliable.

Contd...

- **Monitoring and Logging:** Provide **real-time feedback on the performance and health of applications** and infrastructure, allowing teams to **quickly detect and resolve issues**. Tools like **Prometheus, Grafana** are commonly used.
- **Agile Methodology Integration:** DevOps often works hand-in-hand with **Agile practices**, which focus on **iterative development, flexibility, and customer collaboration**.
- **Cultural Shift:** DevOps is as much about **culture** as it is about tools and processes. It requires a shift in **mindset** where teams adopt a shared responsibility for the software product, from development through to production. This culture emphasizes **collaboration, learning, and continuous improvement**.

DevOps Engineer

- Managing the **software's infrastructure and deployment** processes.
- **Automating** build, test, and deployment pipelines.
- Ensuring that the software is **scalable, reliable, and efficiently deployed.**
- Knowledge of **cloud platforms, automation tools (e.g., Jenkins, Docker), and system administration.**

Why DevOps

- **Faster Time to Market :** DevOps practices **automate much of the development and deployment process,** enabling teams to deliver software updates more **frequently and reliably**. This speed allows companies to respond quickly to market demands and customer needs.
- **Improved Collaboration and Communication :** Traditionally, development and operations teams worked in **silos**, often leading to **miscommunication, delays, and conflicts**. DevOps fosters a **culture of collaboration, where developers, operations, and other stakeholders work together** throughout the software development lifecycle.
- **Enhanced Stability and Reliability :** DevOps practices include **managing infrastructure through code**, which ensures consistency and reduces errors caused by manual configuration. **Continuous testing and monitoring** ensure that any issues are detected early, reducing the risk of major failures and ensuring the software’ s stability

Contd…

- .**Scalability and Flexibility**
  - **Efficient Resource Management:** DevOps practices enable more efficient use of resources, making it easier to scale applications up or down based on demand.
  - **Cloud Integration:** DevOps is closely tied to cloud computing, allowing organizations to leverage cloud infrastructure for scalable, flexible, and cost-effective deployments.
- **Continuous Improvement**
  - **Feedback Loops:** DevOps practices create continuous feedback loops from development to operations, allowing teams to learn from each release and make improvements rapidly.
  - **Agile Integration:** DevOps aligns well with Agile methodologies, supporting iterative development and continuous improvement.
- **Cost Efficiency**
  - **Reduced Downtime:** Automated deployments and continuous monitoring help identify and resolve issues before they impact users, reducing downtime and associated costs.
  - **Resource Optimization:** By automating tasks and optimizing workflows, DevOps reduces the need for manual interventions, lowering operational costs.

Contd…

- **Better Quality and Innovation :** With the ability to deploy updates quickly, teams can experiment with **new features, gather user feedback, and iterate on ideas rapidly.** Continuous testing and integration practices ensure that code is consistently tested, leading to higher quality software and fewer defects in production.
- **Competitive Advantage :** Companies that adopt DevOps can respond to **market changes faster, delivering new features and updates more rapidly than competitors.** The efficiency and collaboration enabled by DevOps free up resources and time for innovation, allowing teams to focus on **creating value** rather than managing infrastructure.

Benefits of DevOps

- **Faster Time to Market:** Streamlined processes and automation allow for **quicker development cycles and faster delivery of new features.**
- **Improved Quality and Reliability:** Continuous testing, integration, and monitoring reduce the chances of **errors reaching production.**
- **Increased Efficiency:** Automation of repetitive tasks **frees up time for innovation and complex problem-solving.**
- **Enhanced Collaboration:** Breaking down silos leads to **better communication, shared goals, and a more cohesive team environment.**
- **Scalability:** DevOps practices enable organizations to easily **scale their infrastructure and applications according to demand.**

Common DevOps Tools

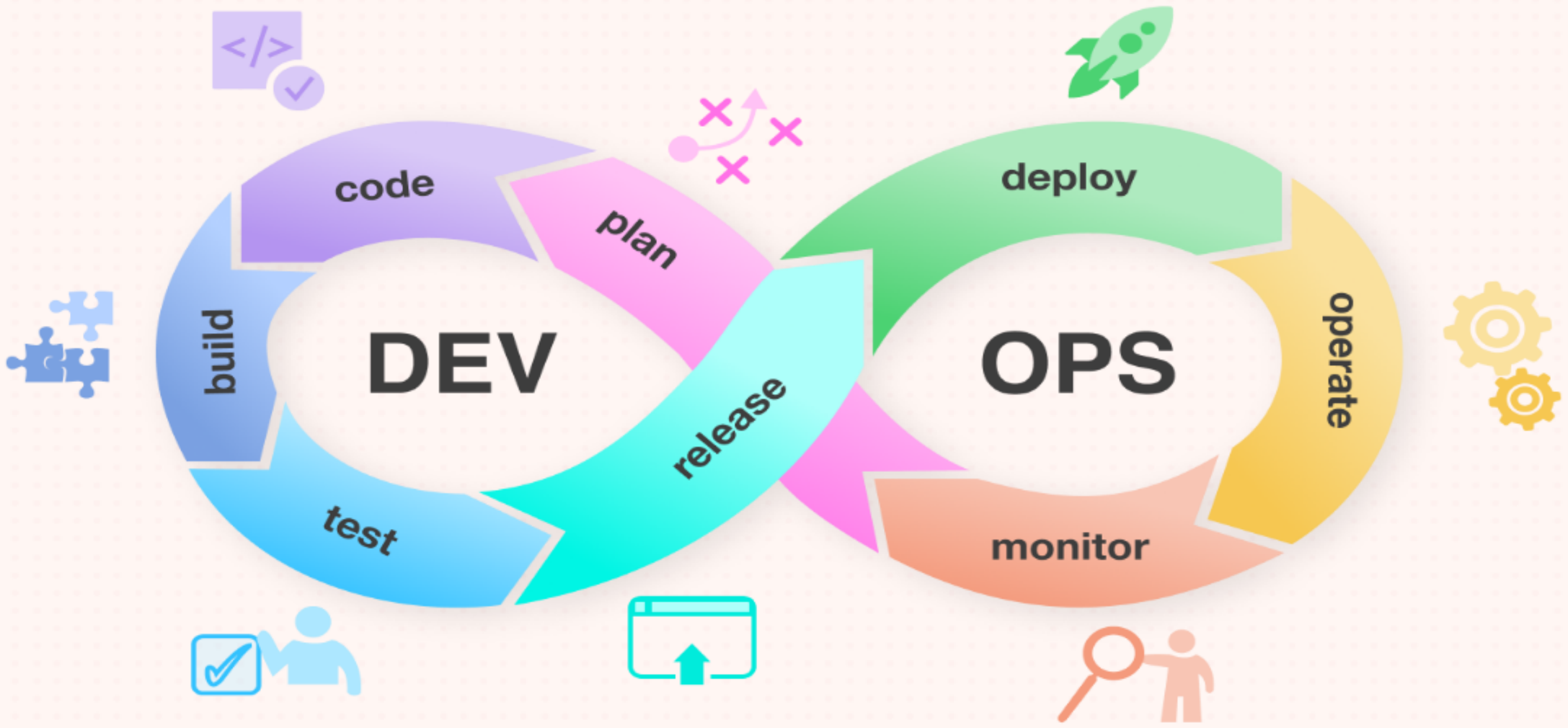
- **Version Control:** Git, GitHub, GitLab
- **CI/CD Pipelines:** Jenkins, CircleCI, GitLab CI/CD
- **Configuration Management:** Ansible, Puppet, Chef
- **Containerization:** Docker
- **Orchestration:** Kubernetes
- **Monitoring:** Prometheus, Grafana, Nagios
- **Cloud Platforms:** AWS, Azure, Google Cloud Platform (GCP)

DevOps Life Cycle

- Plan
- Code
- Build
- Test

- Release

- Deploy
- Operate
- Monitor


code
plan
build
test
release
deploy
operate
monitor
DEV
OPS
SIMFORM

DevOps Practices

- Continuous Integration (CI)
- Continuous Delivery (CD)
- Infrastructure as Code (IaC)
- Automated Testing
- Monitoring and Logging

DevOps Culture

- Collaboration and Communication
- Shared Responsibility
- Continuous Improvement
- Learning and Experimentation

DOCKER

What is Docker?

- Docker is an **open-source platform** that **automates the deployment, scaling, and management** of applications in **lightweight, portable containers.**
- Docker is an **open-source engine** that **automates the deployment of applications into containers.**
- **Containers:** Isolated environments that **package applications with all their dependencies, ensuring consistency across different environments.**
- **Docker Image:** A **read-only template** used to **create containers**, containing the **application code, runtime, libraries, and settings.**
- **Docker Engine:** The core of Docker, enabling **container creation and management.**

Contd…

- Developers – **Applications running inside containers**
- Operations – **Managing the containers**
- Docker aims to reduce the cycle time between code being written and code being tested, deployed, and used.
- It aims to make your **applications portable, easy to build, and easy to collaborate on.**
- Docker is a **container management service.**
- **Develop, ship and run anywhere.**
- The whole idea of Docker is for developers to easily **develop applications, ship them into containers** which can then be deployed anywhere.
- Docker containers are deployed anywhere, on **any physical and virtual machines and even on the cloud.**

Why Docker?

- **Portability:** Containers run consistently across different environments, from a **developer's laptop to production servers.**
- **Efficiency: Containers are lightweight**, sharing the host OS kernel, which makes them faster to start and use fewer resources than traditional virtual machines (VMs).
- **Isolation:** Each container runs in its **isolated environment,** improving security and minimizing conflicts between applications.
- **Scalability:** Docker simplifies **scaling applications up or down by quickly starting or stopping containers.**
- **Version Control: Docker images can be versioned**, allowing for easy rollbacks and consistent deployments.

Docker UseCases

- Microservices
- DevOps
- Cloud Computing
- Continuous Integration/Continuous Deployment (CI/CD)

Docker Vs Virtual Machines

	Docker	Virtual Machine
Architecture	**Containers: Lightweight, portable units** that package an application and its dependencies together. Containers share the host system's OS kernel. **Docker Engine:** The Docker Engine manages and runs containers on the host OS.	**Full Virtualization:** VMs run a **full operating system (OS) on top of a hypervisor.** More resource-intensive. **Hypervisor:** VMs are managed by a hypervisor, which can be hosted or bare-metal.
Resource Efficiency	**Lightweight:** Containers share the host OS kernel, making them more lightweight. **Faster startup times** and more efficient resource usage. **Less Overhead:** Since containers don't require a full OS, they have **minimal overhead** compared to VMs.	**Resource-Intensive:** Each VM includes a full OS, leading to higher resource consumption. This can result in **slower startup times** and increased overhead. **More Isolation:** VMs are fully isolated from each other, which can provide a **higher level of security** but at the cost of efficiency.

Contd…

	Docker	Virtual Machine
Portability	**Highly Portable:** Containers can run consistently across **different environments, from a developer's laptop to cloud servers**, as long as Docker is installed.	**Less Portable: VMs are less portable due to their larger size and the dependence on the underlying hypervisor.**
Isolation and Security	**Process-Level Isolation:** Containers provide **process-level isolation**, which is generally sufficient but not as strong as VMs. **Security Best Practices:** Docker security depends on **proper configuration and best practices.**	**Full Isolation:** VMs provide **full OS-level isolation**, meaning each VM runs independently with its own OS, making them more secure and isolated from one another. **Security:** Since VMs do not share the host OS kernel, they are **less vulnerable to certain types of attacks,** providing a more robust security model.

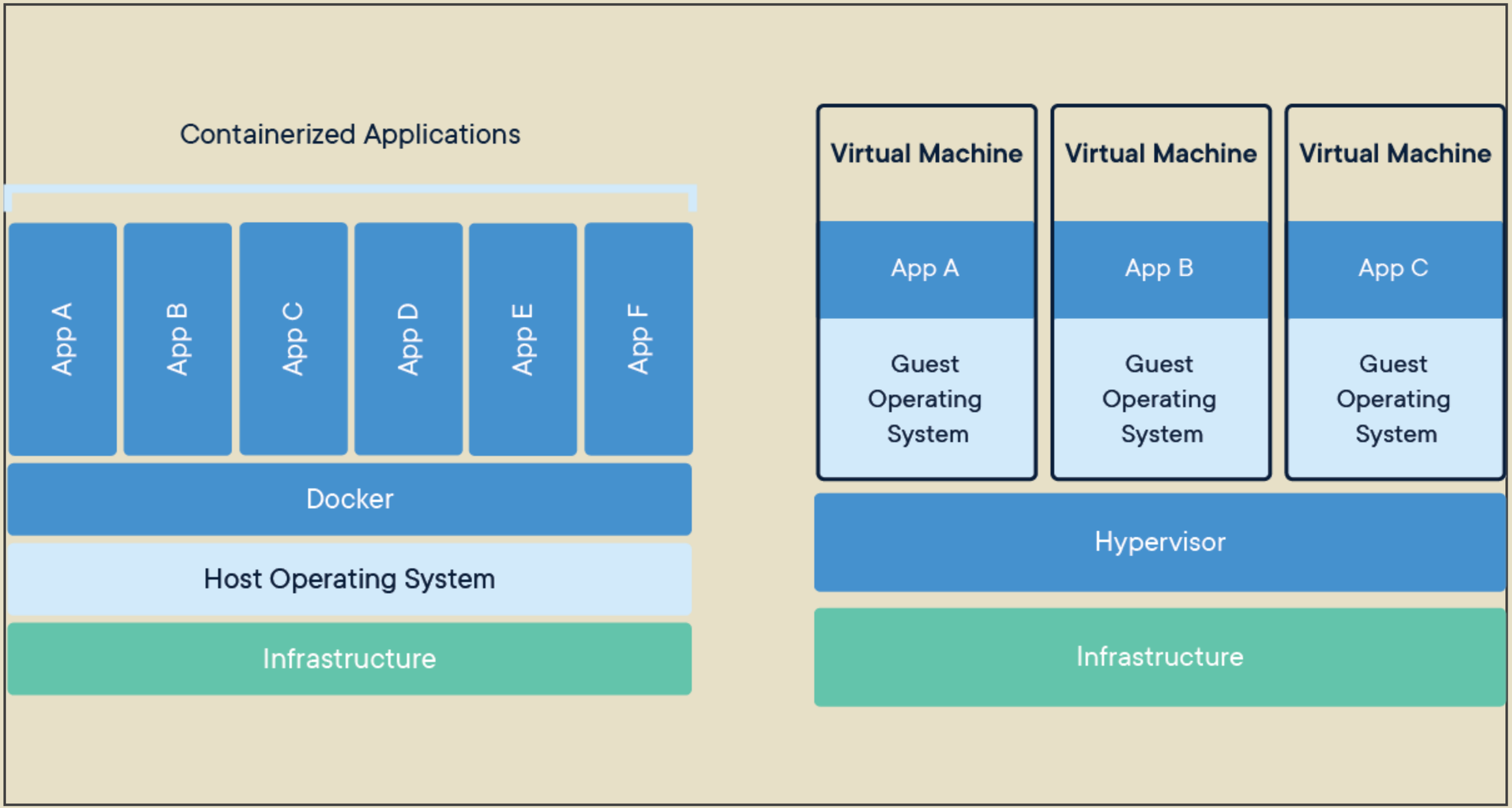
Contd...

	Docker	Virtual Machine
Use Cases	**Microservices and CI/CD:** Ideal for deploying **microservices and integrating into CI/CD pipelines** due to their lightweight nature and fast startup times. **Development and Testing:** Perfect for development and testing environments where consistency and speed are crucial.	**Legacy Applications:** Suitable for running legacy applications that require a specific OS or full OS-level isolation. **Multi-OS Environments:** Useful in scenarios where different OS environments are needed on the same hardware.
Performance	**Faster Startup: Containers start almost instantly** because they don't require booting a full OS. **Better Performance:** Due to **lower overhead,** containers typically have better performance compared to VMs when running the same workloads.	**Slower Startup:** VMs take longer to start as they need to boot a full OS. **More Overhead:** VMs have more overhead due to the need to virtualize hardware and run a full guest OS.

CONTAINERS

Introduction to Containers

- Docker containers are **lightweight, standalone, and executable packages** that include everything needed to run a piece of software, including the **code, runtime, libraries, and system tools.**
- **Isolation:** Containers run in isolated environments, ensuring that applications do not interfere with each other.
- **Portability:** Containers can run consistently across different environments, from development to production.
- **Efficiency:** Containers share the host OS kernel, making them more efficient and faster to start compared to traditional virtual machines.


Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure
Virtual Machine
App A
Guest Operating System
Virtual Machine
App B
Guest Operating System
Virtual Machine
App C
Guest Operating System
Hypervisor
Infrastructure

Working of Docker Container

- **Container Structure:**
  - **Docker Image:** A read-only template used to create containers. It includes the application code, dependencies, and system tools.
  - **Docker Container:** A runtime instance of a Docker image, representing a live application.
- **Container Lifecycle:**
  - **Build:** Create a Docker image using a Dockerfile.
  - **Run:** Instantiate a container from an image.
  - **Stop:** Stop a running container.
  - **Remove:** Delete a container when it's no longer needed.

Benefits of Docker Containers

- **Consistency:** Containers ensure that an **application runs the same way across different environments** (development, testing, production).
- **Scalability:** Containers can be easily **scaled up or down**, making them ideal for microservices and cloud-based applications.
- **Resource Efficiency:** Containers are **lightweight**, using fewer resources than virtual machines, leading to better performance and cost savings.
- **Rapid Deployment:** Containers can be **started, stopped, and deployed quickly,** facilitating rapid development cycles.
- **Isolation and Security:** Containers provide **process-level isolation**, reducing the risk of conflicts and enhancing security.

Docker Vs Container

Docker	Virtual Machine
Docker is a platform that enables **developers to create, deploy, and manage applications using containers.** It includes a set of tools, services, and commands that **facilitate containerization**, making it easier to **package and run applications consistently across different environments.**	A container is a **lightweight, standalone, and executable software package** that includes everything needed to run an application— **code, runtime, libraries, and system tools.** Containers ensure that an application runs consistently regardless of the environment.
Docker provides the tools and environment to **build, run, and manage containers.** It includes components like Docker Engine, Docker Hub, Docker Compose, and Docker Swarm, which help in **container creation, storage, orchestration, and deployment.**	A container is the **actual runtime instance** where an application executes. It is the **end product created using Docker**. Containers encapsulate an application and its dependencies, isolating it from the host system.

Contd…

Docker	Container
Docker **automates the deployment of applications** inside containers, handling the **creation, management, and orchestration of containers.** It also provides features for container networking, storage, and security.	A container is the **environment where an application runs**, ensuring that the application has everything it needs to operate, isolated from other processes. Containers can be **started, stopped, and managed by Docker** or similar containerization platforms.
Docker includes a broader ecosystem of tools and services: **Docker Engine:** The core service for creating and running containers. **Docker Hub:** A cloud-based repository where Docker images can be stored, shared, and accessed. **Docker Compose:** A tool for defining and running multi-container Docker applications. **Docker Swarm/Kubernetes:** Tools for orchestrating and managing clusters of containers.	Containers are a **fundamental unit within the Docker ecosystem.** They are the instances that run the packaged application, isolated from other containers and the host system. **Containers can be created from images** stored on Docker Hub or other registries.

ORCHESTRATION

Orchestration

- **Orchestration** refers to the **automated arrangement, coordination, and management of complex computing systems, services, and workflows.**
- In the context of containerization, **orchestration is the process of managing and automating the deployment, scaling, and operation of containers.**
- As organizations move towards **microservices and cloud-native architectures**, managing the lifecycle of containers becomes more complex. Orchestration simplifies and automates these tasks.

Key Functions of Orchestration

- **Automated Deployment:** Orchestration tools **automate the deployment of containers** across multiple servers, ensuring that the right services are available where and when needed.
- **Scaling:** Dynamically scale applications up or down based on demand, **automatically adjusting the number of containers running at any given time.**
- **Load Balancing: Distribute incoming network traffic across multiple containers** to ensure no single container is overwhelmed.
- **Self-Healing: Automatically restart failed containers or replace them with healthy ones**, ensuring the system remains operational with minimal downtime.

Benefits of Orchestration

- **Efficiency:** Streamlines operations by **automating repetitive tasks**, reducing **manual intervention, and freeing up resources** for other tasks.
- **Reliability:** Enhances system reliability through **automated failover and recovery processes**, reducing downtime and improving system resilience.
- **Consistency:** Ensures **consistent deployment and operation** across different environments, from **development to production**.
- **Scalability:** Easily **manage large-scale deployments**, automatically adjusting resources based on workload demands.

Orchestration Tools

- **Kubernetes:** The most widely used container orchestration platform, known for its **scalability and flexibility in managing large, distributed applications.**
- **Docker Swarm:** Docker’ s native orchestration tool, **simpler to set up and use**, ideal for **smaller or less complex deployments**.
- **Apache Mesos:** A **cluster manager** that can also orchestrate containers, supporting both Docker and other workloads, suitable for large-scale environments.

Summary

- DevOps is a work culture where the developers and operations team work together to deliver incremental products by incorporating customer feedback.
- Docker is a platform used to containerise application and ship anywhere.
- Container is a portable, light weight component to build and run applications.
- Orchestration is the arrangement of container workflow.
- These automated tools helps to deliver complex task in an time effective manner.

THANK YOU