KEMENTERIAN SUMBER MANUSIA

MALAYSIA MADANI

KELAS BIMBINGAN PELAKSANAAN RANGKA KERJA DEVOPS KESUMA

26 – 28 NOV 2024 (SELASA – KHAMIS)
8:30 PAGI HINGGA 4:30 PETANG
SAMA SAMA HOTEL

Objectives

1) Understand basic knowledge of DevOps principles and practices.

2) To understand the best infrastructure setup.

3) To provide overview of Monitoring and Logging in a DevOps environment .

4) To provide knowledge and guide on best practice of software architecture.

AGENDA BENGKEL HARI PERTAMA

TARIKH	WAKTU	AKTIVITI
Introduction to DevOps & Version Control with Git		
26/11/2024 Selasa	08:30 pg – 09:00 pg	Pendaftaran peserta dan sarapan pagi
	09:00 pg – 09:10 pg	Bacaan doa
	09:10 pg – 09:30 pg	Ucapan kata-kata aluan
	09:30 pg – 12:00 tgh	***Introduction to DevOps*** ***Version Control with Git*** ***Hands On***
	12:00 tgh – 02:00 ptg	Rehat dan makan tengah hari
	02:00 ptg – 04:00 ptg	***Git Branching and Collaboration*** ***Hands On***
	04:00 ptg – 04:30 ptg	Rehat dan minum petang

AGENDA BENGKEL HARI KEDUA

TARIKH	WAKTU	AKTIVITI
CI/CD Implementation & Infrastructure Setup		
27/11/2024 Rabu	08:30 pg – 09:00 pg	Pendaftaran peserta dan sarapan pagi
	09:00 pg – 12:00 tgh	***CI/CD Pipeline*** ***Automated Testing in CI/CD*** ***Introduction to Docker (Containerization)*** ***Hands On – Single Branch CI/CD***
	12:00 tgh – 02:00 ptg	Rehat dan makan tengah hari
	02:00 ptg – 04:00 ptg	***Infrastructure Setup for DevOps*** ***Introduction to Load Balancing*** ***Hands On – Load Test using Jenkins***
	04:00 ptg – 04:30 ptg	Rehat dan minum petang

AGENDA BENGKEL HARI KETIGA

TARIKH	WAKTU	AKTIVITI
Advanced DevOps Practices & Architecture		
28/11/2024 Khamis	08:30 pg – 09:00 pg	Pendaftaran peserta dan sarapan pagi
	09:00 pg – 12:00 tgh	***Sambungan Hands On*** ***Monolithic vs Modular Architecture***
	12:00 tgh – 02:00 ptg	Rehat dan makan tengah hari
	02:00 ptg – 04:00 ptg	***Introduction to Kubernetes (Container Orchestration)*** ***Monitoring & Logging in DevOps*** ***Wrap-Up & Q&A Session***
	04:00 ptg – 04:30 ptg	Minum petang dan bersurai

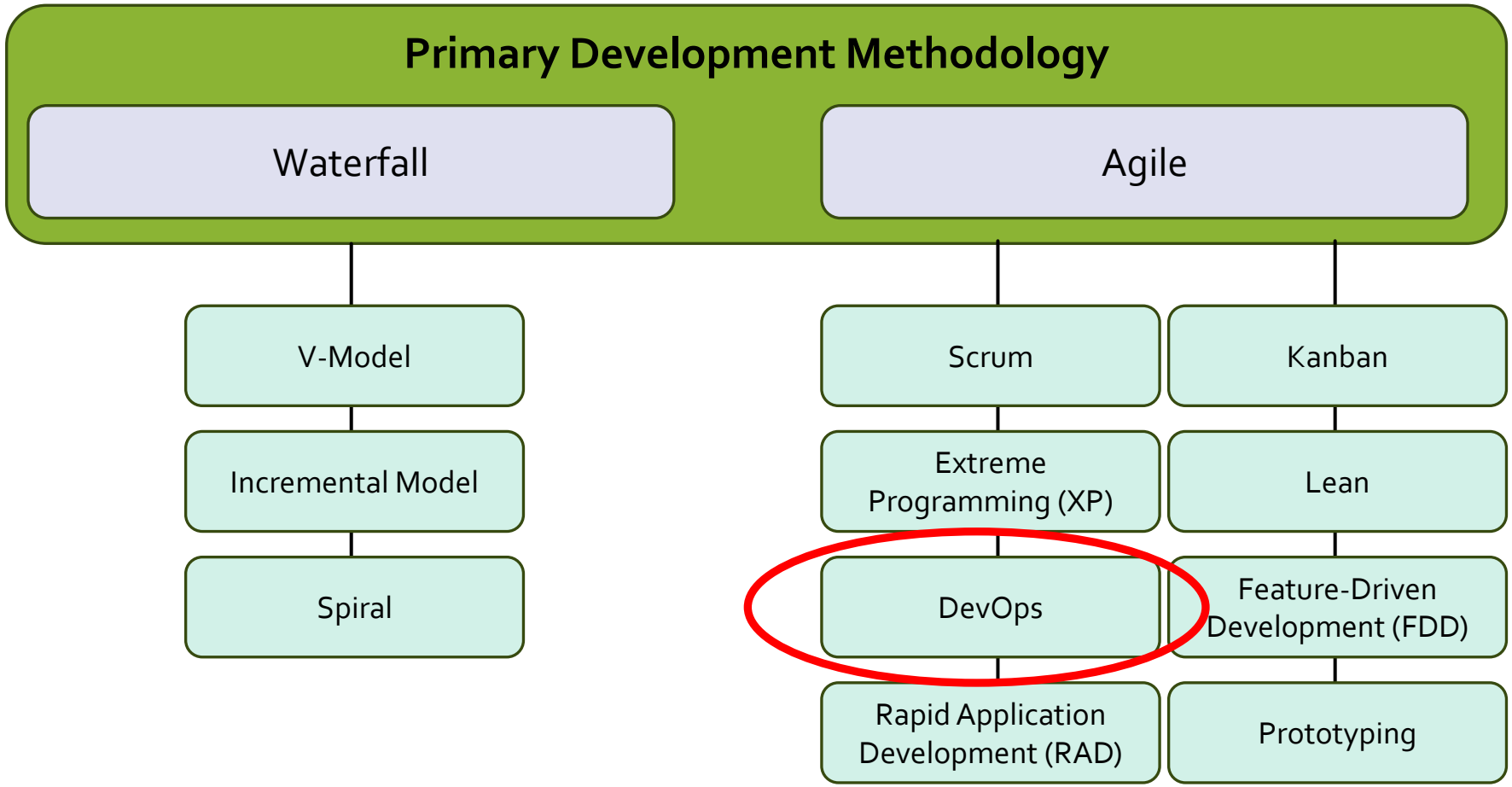
KAEDAH LATIHAN

1. Penerangan
    a. Trainer akan memberi penerangan secara ringkas atau mendalam tentang sesuatu topik.
    b. Sepanjang penerangan, peserta latihan boleh mengajukan soalan yang berkaitan dengan topik yang dibentangkan.
2. Hands-on / Latihan
    a. Peserta bengkel akan diberi penerangan dan langkah-Langkah untuk melakukan latihann berkaitan dengan penerangan yang telah diberi.

Day #1

Introduction to DevOps

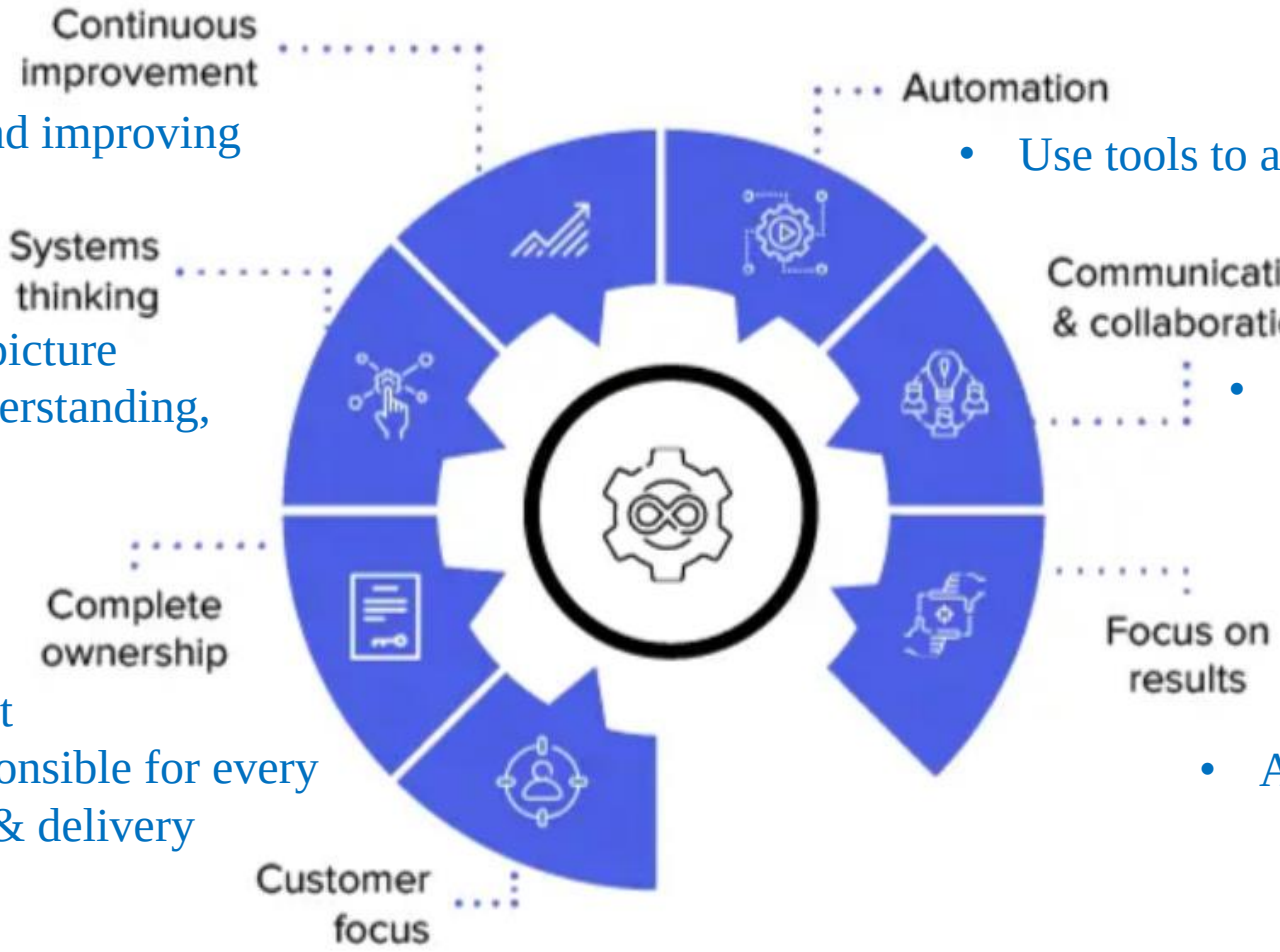
Types of Development Methodology



What is DevOps

- Modern approach to software development and IT operations that promotes **collaboration**, **automation**, and **efficiency**.
- It bridges the gap between two traditionally separate teams:
  - ✓ **Development team(DEV)**, responsible for writing code and building applications
  - ✓ **Operations team (OPS)**, responsible for deploying and managing those applications in production environments.
- The goal of DevOps is to streamline the process of **building**, **testing**, **deploying**, and **maintaining** software, ensuring faster and more reliable delivery.

Principles of DevOps

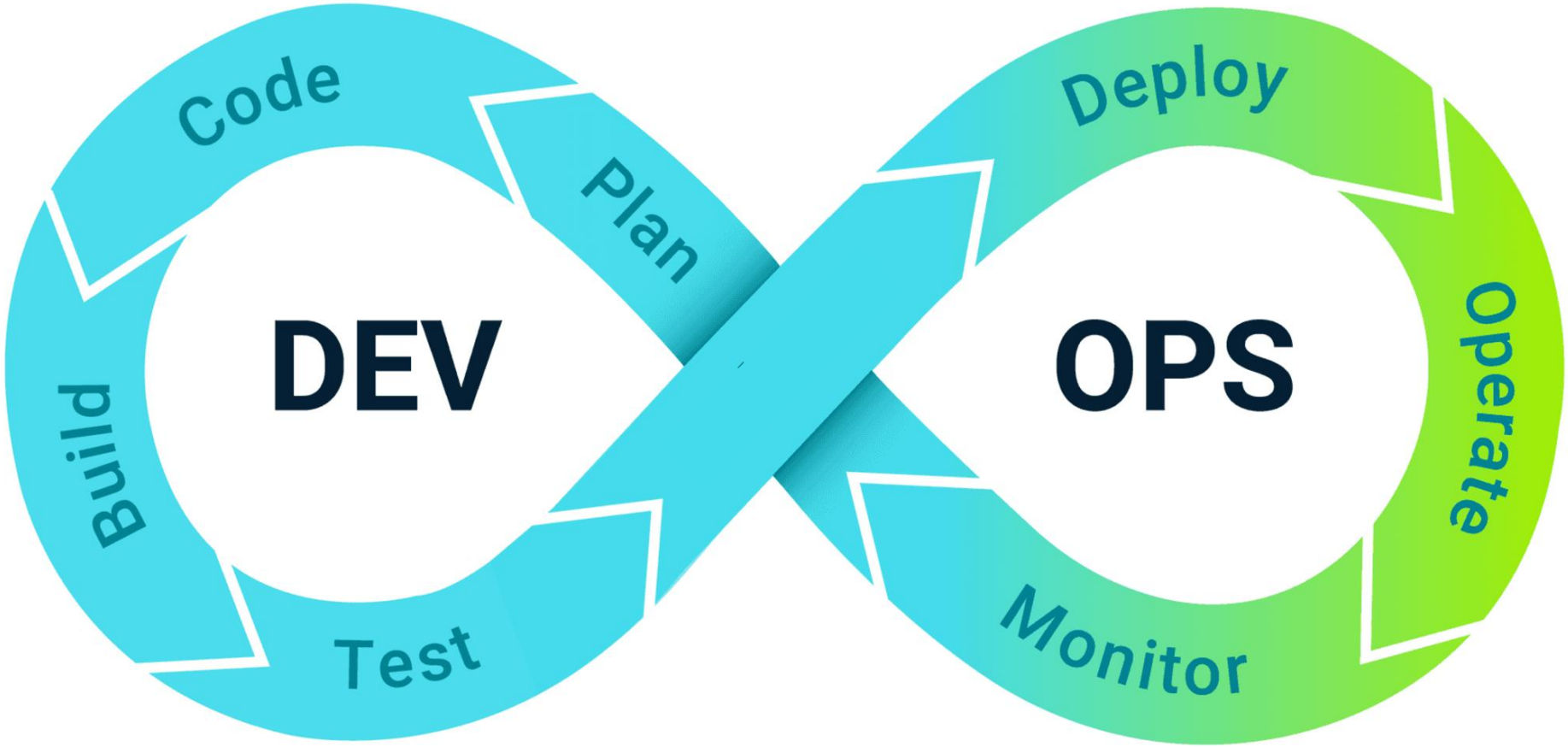


- Never stop learning and improving
- Use tools to automate repetitive tasks
- Help team see bigger picture
- Help productivity, understanding, better product
- Alway improve communication & collaboration
- The One Team mindset
- Dev & Ops Team responsible for every stage of development & delivery
- Always keep in min the end goal
- Prioritize the user or customer.
- Always gather their feedback

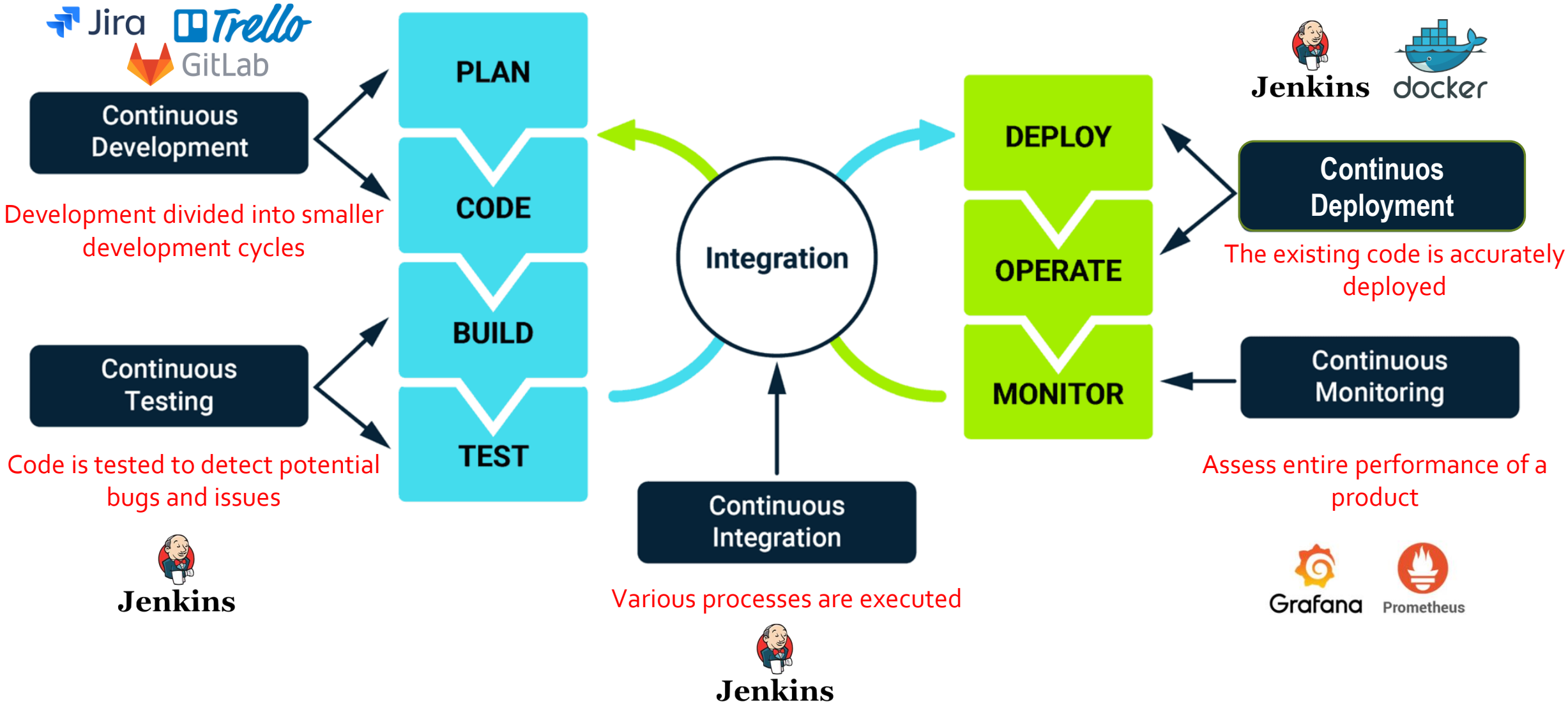
Key Components of DEVOPS

Cultural Shift DevOps is as much about mindset as it is about tools	**Automation** Using tools and technologies to automate repetitive tasks	**Continuous Integration (CI)** Code changes & Automated Testing
Continuous Delivery (CD) Code is automatically prepared for deployment to production	**Monitoring and Feedback** Code is automatically prepared for deployment to production	**Scalability and Flexibility** Easier to scale systems up or down based on demand

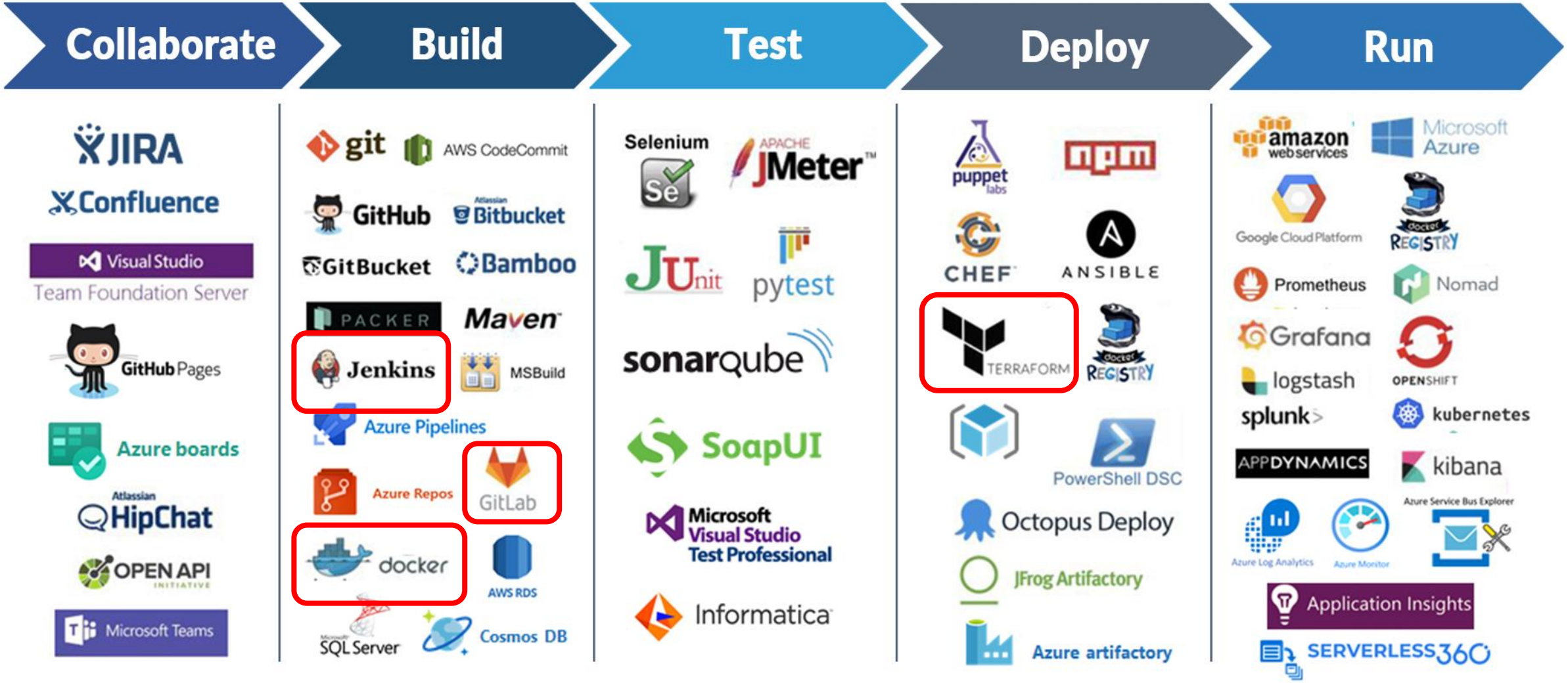
DevOps Life Cycle



DevOps Life Cycle



DevOps Tools and Technologies Choices



Version Control with Git

What are we focusing in this topic

- Version Control
- GIT
- Sourcetree
- Hands-on

What is Version Control

- A system to **manage changes to source code** or files over time.
- Benefits:
  - ✓ Tracks changes
  - ✓ Facilitates collaboration
  - ✓ Provides backups
- Types of Version Control Systems:
  - ✓ Local: Backups on the same machine.
  - ✓ Centralized: Single server for codebase (e.g., SVN).
  - ✓ **Distributed: Every developer has a full copy (e.g., Git).**

Git Basics

- **Git** is a distributed version control system (DVCS) that allows developers to track changes in their codebase, collaborate effectively, and manage project versions over time. It is widely used in software development and other fields that involve collaborative document editing.

Key Features of Git

Distributed System:

- Every developer has a full copy of the project history (not just the latest snapshot).
- This means developer can work offline and still have access to the entire project history.

Version Control:

- Git keeps track of every change made to files in project.
- Developer can revert to a previous version or view the history of changes.

Branching and Merging:

- **Branches** allow developer to work on different features, bug fixes, or experiments simultaneously.
- Can merge changes from one branch into another after reviewing and resolving conflicts.

Collaboration:

- Git enables multiple developers to work on the same project simultaneously.
- Developers can synchronize their work using **push**, **pull**, and **fetch** commands.

Efficient and Secure:

Git uses algorithms that minimize storage and network usage.

Data integrity is ensured through checksums for all objects in the repository.

How Git Works

- Local Repository:
  - ✓ Git stores your changes on your local machine first. You can commit changes locally without affecting the shared repository
- Remote Repository:
  - ✓ Teams typically use services like GitHub, GitLab, or Bitbucket as remote repositories to share and synchronize their code

Git Terminology

- Repository: Storage for your code
- Commit: A snapshot of changes.
- Branch: Parallel versions of code
- Merge: Combining branches
- Push/Pull: Sync changes between local and remote repositories

GitLab Basics

- What is GitLab?
  - ✓ A web-based DevOps lifecycle tool
  - ✓ Provides Git repository management, CI/CD, and issue tracking
- Core GitLab Features
  - ✓ Repositories: Storing code
  - ✓ Issues: Bug tracking
  - ✓ Merge Requests: Reviewing and merging branches
  - ✓ CI/CD Pipelines: Automating builds and deployments

Sourcetree Basics

- What is Sourcetree?
  - ✓ A graphical user interface (GUI) for Git
  - ✓ Simplifies Git commands for developers

Hands on – Sourcetree

- Installation Steps?
  - ✓ Download and install Sourcetree
    - https://www.sourcetreeapp.com/
  - ✓ Connect your GitLab account (SSH or HTTPS)
  - ✓ Clone a repository using Sourcetree
    - https://github.com/laravel/laravel.git

Git Branching and Collaboration

Branching and Collaboration Strategies

- What is Git Branching?
  - ✓ Feature that allows developers to create independent lines of development within a repository. This enables parallel workstreams without affecting the main codebase until the changes are ready to be integrated.
- Types of Branching Strategies
  - ✓ Feature Branching
  - ✓ Git Flow
  - ✓ Trunk-Based Development
  - ✓ Forking Workflow

Best Practise

- Commit often with meaningful messages
- Pull updates frequently to avoid conflicts
- Use descriptive branch names (e.g. feature/login, bugfix/typo)
- Review and test before merging
- Automate testing using CI pipelines in GitLab/GitHub

Hands on Branching using Sourcetree

- Branching
  - ✓ Create a branch with name
  - ✓ Make changes to code
  - ✓ Commit and Push Changes
  - ✓ Pull and Merge branch
  - ✓ Resolve Merge Conflicts

Day #2

CI/CD Pipeline

What is Continuous Integration/Continuos Delivery

- Continuous Integration and Continuous Delivery (CI/CD)
  - ✓ A **software development** approach that **streamlines the process** of **integrating, building, testing, and deploying code changes**.
  - ✓ It focuses on **automating workflows** to enable faster, more reliable delivery of software

Continuous Integration (CI)

- **Definition**:
  - ✓ The practice of frequently merging code changes from multiple developers into a shared repository
- **Key Features**:
  - ✓ **Frequent Code Merges**: Developers integrate code several times a day
  - ✓ **Automated Builds**: Code is automatically built to ensure it compiles without errors.
  - ✓ **Automated Testing**: Unit tests and other automated tests validate code functionality.
  - ✓ **Immediate Feedback**: Developers are notified quickly if there's a failure, enabling rapid fixes.
- **Objective:** To detect and address integration issues early in the development process

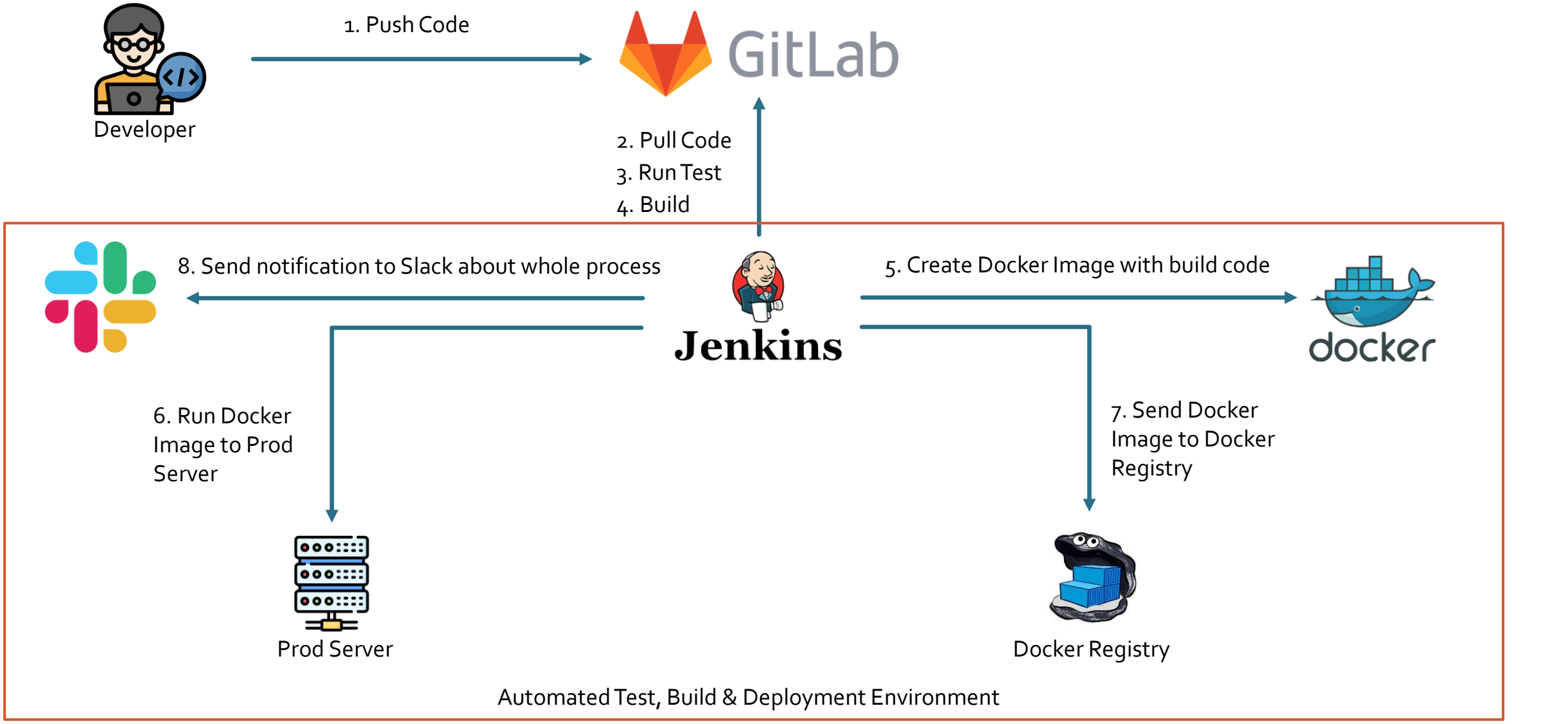
Continuous Delivery (CD)

- **Definition**:
  - ✓ An extension of CI that automates the delivery of validated code to a staging or production environment
- **Key Features**:
  - ✓ **Automated Deployment Pipelines**: After successful CI, code changes are automatically deployed to staging environments
  - ✓ **Manual Approvals (Optional)**: Teams may include manual checks before deploying to production.
  - ✓ **Environment-Specific Configurations**: Ensures that the software functions correctly in different environments (staging, production, etc.).
- **Objective:** To keep code in a deployable state and deliver updates to users as quickly and safely as possible

Key CI/CD Tools

- **Version Control**:
  - ✓ Git, GitHub, GitLab
- **CI/CD Platforms**:
  - ✓ Jenkins, GitLab CI/CD, GitHub Actions, Travis CI, CircleCI, Bamboo
- **Testing Frameworks**:
  - ✓ Jenkins, JUnit, Selenium, Cypress
- **Containerization**:
  - ✓ Docker
- **Orchestration**:
  - ✓ Kubernetes, Docker Swarm

CI/CD Pipeline - Example



Infrastructure Setup for DevOps

Infrastructure Setup Suggestion

1. Code Repository Server:

- ✓ **Purpose**: Host the source code repository and manage version control.
- ✓ **Tools**: Install GitLab.

2. Build and Deployment Server (Jenkins):

- ✓ **Purpose**: Automate the CI/CD processes, including testing, building Docker images, and deployment.
- ✓ **Tools**: Install Jenkins and plugins for Docker, GitLab integration, and Slack notifications.

3. Docker Host Server:

- ✓ **Purpose**: Build Docker Images for the application
- ✓ **Tools**: Install Docker and Docker CLI tools.

4. Docker Registry Server:

- ✓ **Purpose**: Store and manage built Docker image
- ✓ **Tools**: Install Docker Hub or setup local repository.

5. Production Server:

- ✓ **Purpose**: Deploy and run the Dockerized application
- ✓ **Tools**: Install Nginx.

Hands on