DevOps

Developers

- dev
- tester

- write code ←
- Building code ←
  └→ deployable package
- testing → manual testing
  → env → dev
→ latest updates →
  → node - 13.x ←
  → mysql - 8.0
  - angular - 8.0

apache2

Operations env → prod.

EC2 inst
→ node
  └ 12.x
→ mysql
  - 5.7
→ Angular
  - 7.0

- infrastructure
  → machine
    └→ configure
  └→ network

- manage info

Problems

- Managing and tracking changes in the code is difficult
- Incremental builds are difficult to manage, test and deploy
- Manual testing and deployment of various components/modules takes a lot of time
- Ensuring consistency, adaptability and scalability across environments is very difficult task
- Environment dependencies makes the project behave differently in different environments

Solutions to the problem

- Managing and tracking changes in the code is difficult: **SCM tools**
- Incremental builds are difficult to manage, test and deploy: **Jenkins**
- Manual testing and deployment of various components/modules takes a lot of time: **Selenium**
- Ensuring consistency, adaptability and scalability across environments is very difficult task: **Puppet**
- Environment dependencies makes the project behave differently in different environments: **Docker**

Overview

- DevOps is a combination of two words development and operations

  cloud [AWS]

- Promotes collaboration between Development and Operations Team to deploy code to production faster in an automated & repeatable way

- DevOps helps to increases an organization's speed to deliver applications and services

- It allows organizations to serve their customers better and compete more strongly in the market

- Can be defined as an alignment of development and IT operations with better communication and collaboration

Write code → build → test → Configure prod → deploy
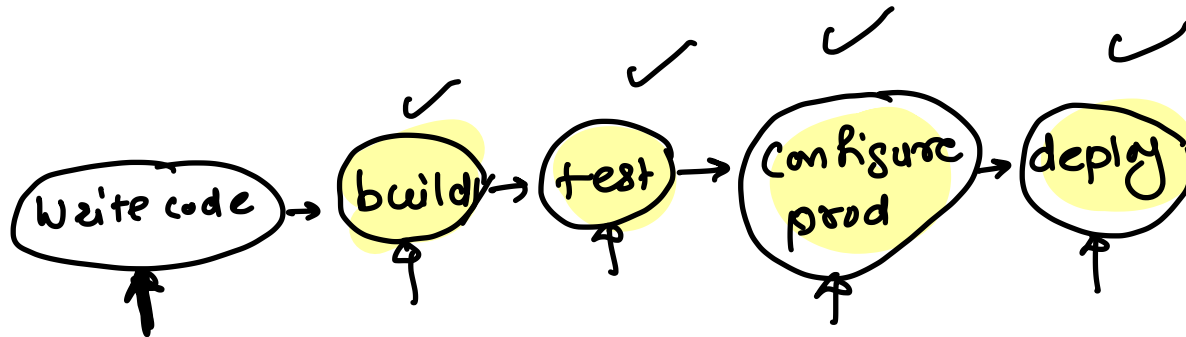
Why DevOps is Needed?

- Before DevOps, the development and operation team worked in complete isolation
- Testing and Deployment were isolated activities done after design-build. Hence they consumed more time than actual build cycles.
- Without using DevOps, team members are spending a large amount of their time in testing, deploying, and designing instead of building the project.
- Manual code deployment leads to human errors in production
- Coding & operation teams have their separate timelines and are not in synch causing further delays

What is DevOps ?

- DevOps is not a goal but a never-ending process of continuous improvement

- It integrates Development and Operations teams

- It improves collaboration and productivity by
  - Automating infrastructure
  - Automating workflow
  - Continuously measuring application performance

Common misunderstanding

- DevOps is not a role, person or organization
- DevOps is not a separate team
- DevOps is not a product or a tool
- DevOps is not just writing scripts or implementing tools

Reasons to use DevOps

- **Predictability:** DevOps offers significantly lower failure rate of new releases

- **Reproducibility:** Version everything so that earlier version can be restored anytime

- **Maintainability:** Effortless process of recovery in the event of a new release crashing or disabling the current system

- **Time to market:** DevOps reduces the time to market up to 50% through streamlined software delivery. This is particularly the case for digital and mobile applications

- **Greater Quality:** DevOps helps the team to provide improved quality of application development as it incorporates infrastructure issues

- **Reduced Risk:** DevOps incorporates security aspects in the software delivery lifecycle. It helps in reduction of defects across the lifecycle

- **Resiliency:** The Operational state of the software system is more stable, secure, and changes are auditable
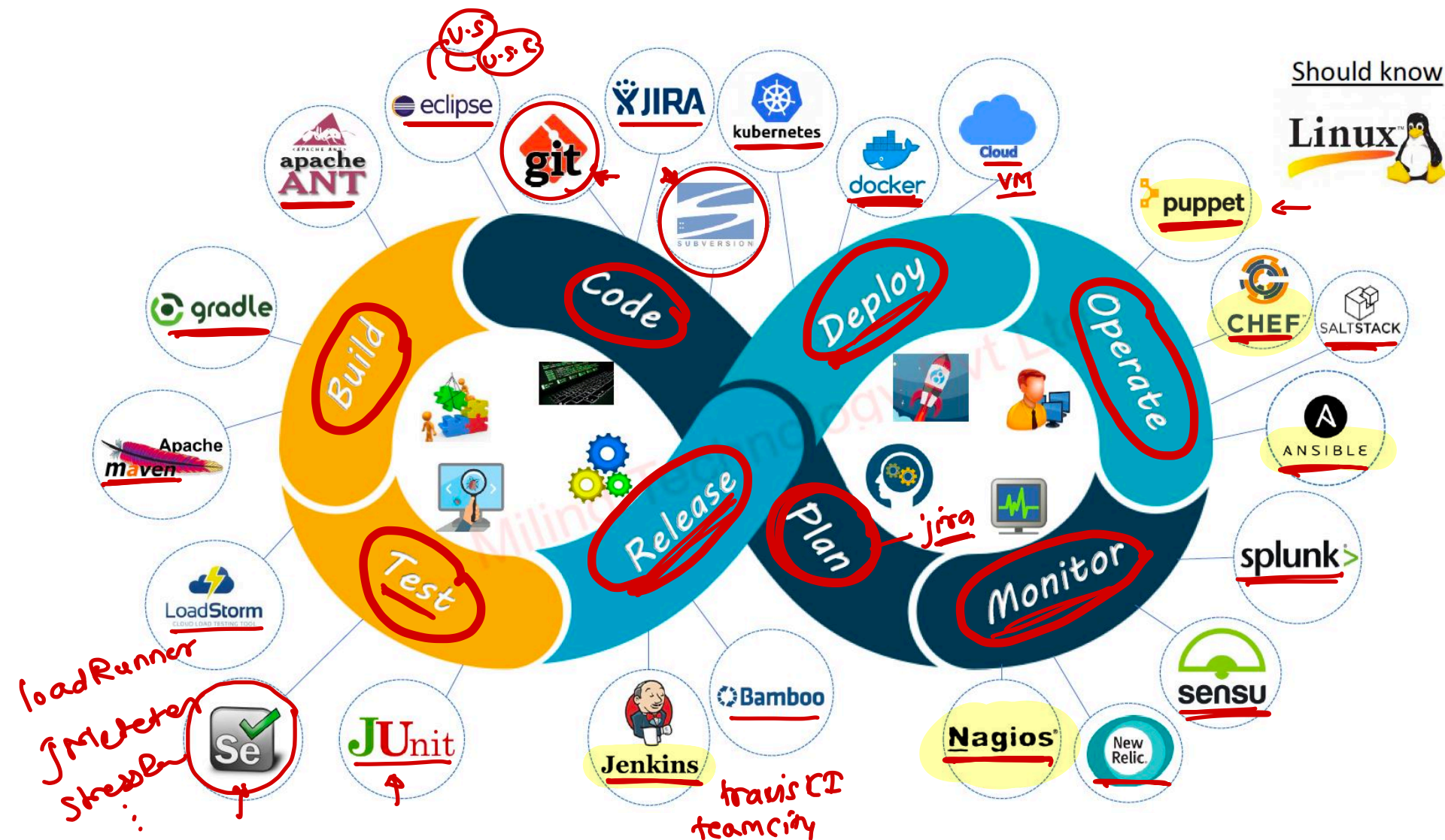
Reasons to use DevOps

- **Cost Efficiency:** DevOps offers cost efficiency in the software development process which is always an aspiration of IT companies' management

- **Breaks larger code base into small pieces:** DevOps is based on the agile programming method. Therefore, it allows breaking larger code bases into smaller and manageable chunks
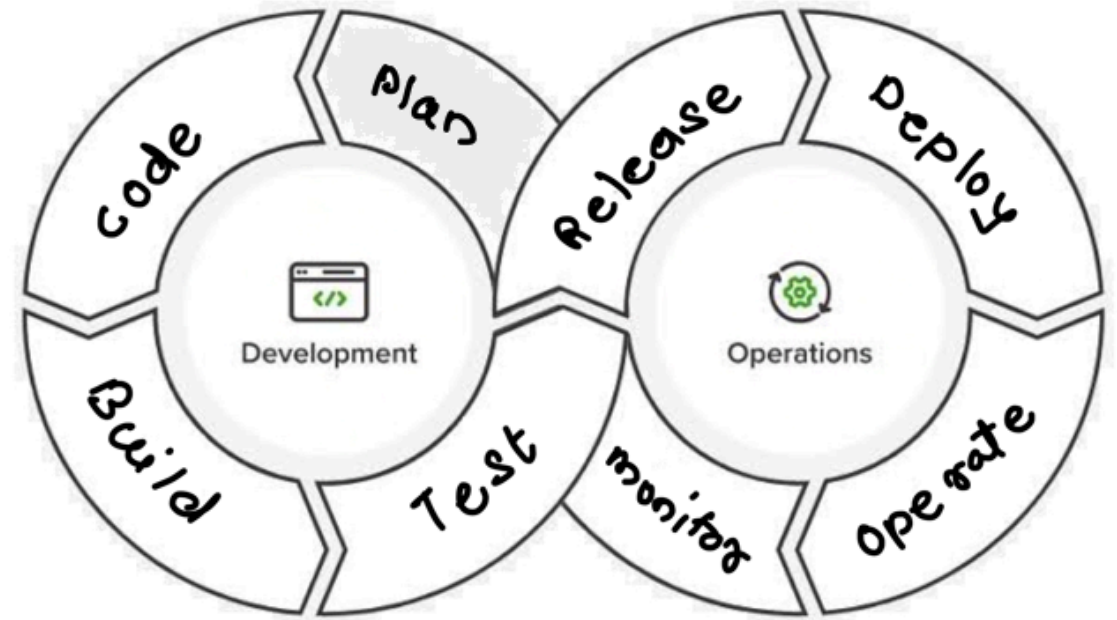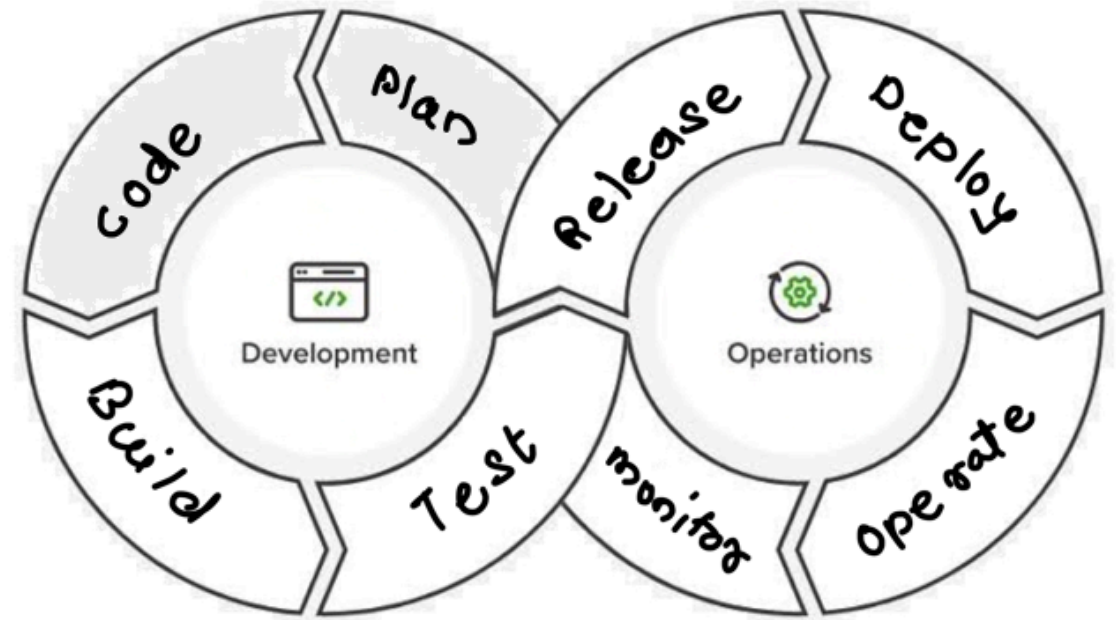
DevOps Lifecycle

DevOps Lifecycle - Plan

- First stage of DevOps lifecycle where you plan, track, visualize and summarize your project before you start working on it

- Planning tools
    - Google sheet
    - Box
   - Dropbox
    - Trello
    - Jira
    - Planio

DevOps Lifecycle - Code

- Second stage where developer writes the code using favorite programming language
- Coding Tools
  - IDEs: Eclipse, Visual Studio etc.
  - SCM: Git, Subversion, CVS etc.
  - Package management: npm etc.

DevOps Lifecycle -Build

- Integrating the required libraries

- Compiling the source code

- Create deployable packages

- Build tools
    - Maven
    - Gradle
    - Ant

① Java
   ↳ maven > .jar, .war

② Angular
   ↳ ng build > .html, .js

③ Android
   ↳ java → gradle > .apk

④ ios
   ↳ swift → xcodebaild > .ipa

DevOps Lifecycle - Test

- Process of executing automated tests

- The goal here is to get the feedback about the changes as quickly as possible

- Testing tools
  - JMeter → load
  - Selenium → java, c++, js, python
  - JUnit → java
  - QUnit → ...
  - NUnit → .net
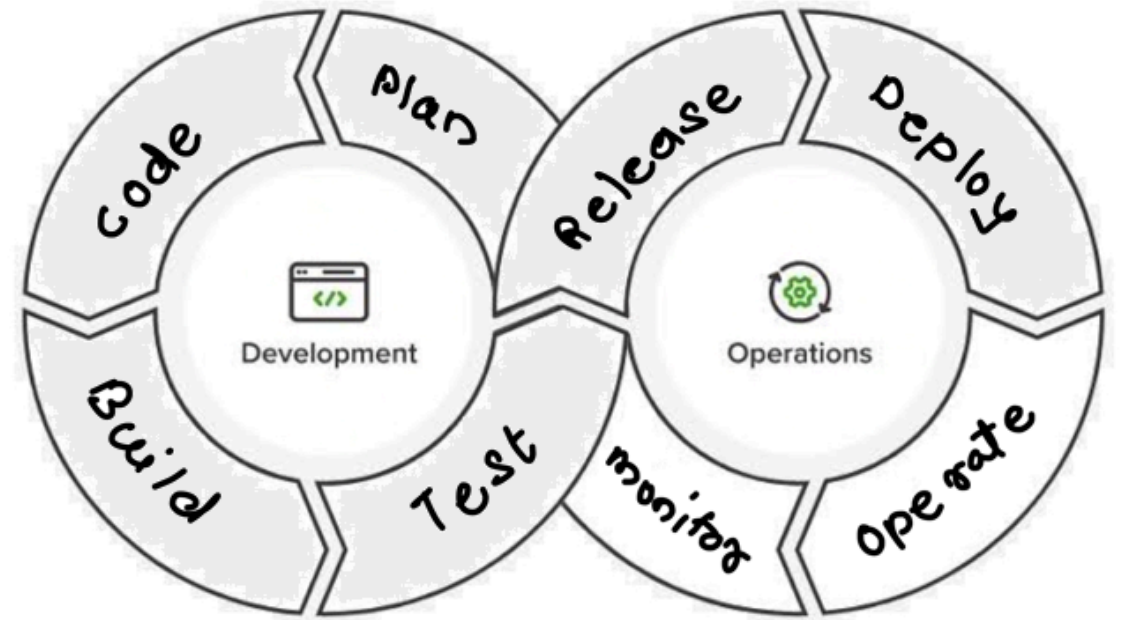 - Appium → android
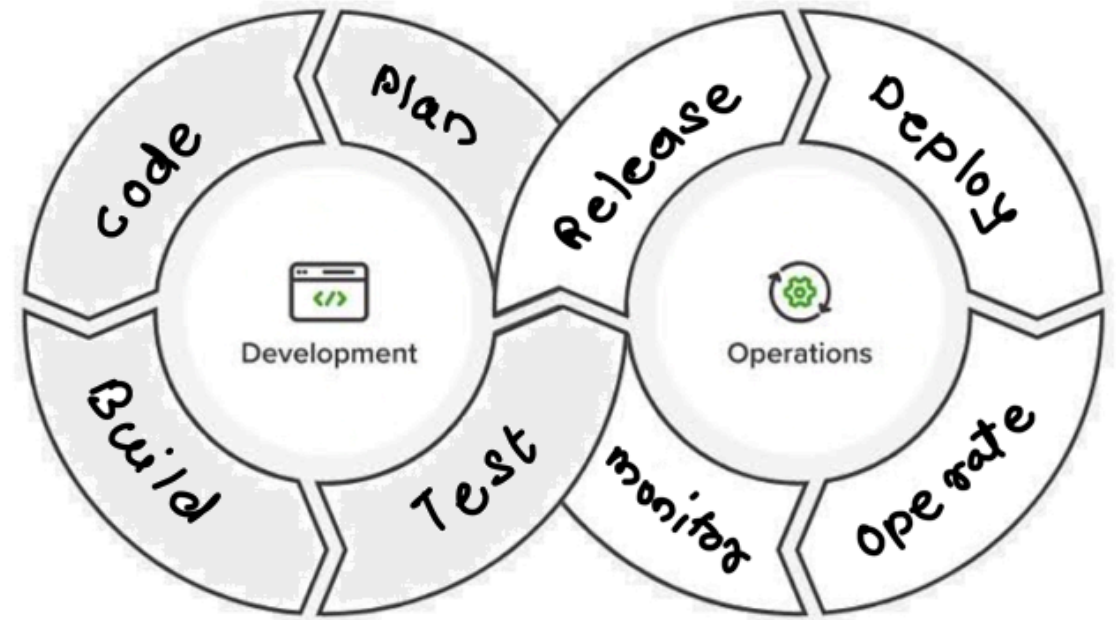
DevOps Lifecycle - Release

- This phase helps to integrate code into a shared repository using which you can detect and locate errors quickly and easily

- Release tools
  - Jenkins
  - Travis CI
 - Bamboo
  - GitLab CI
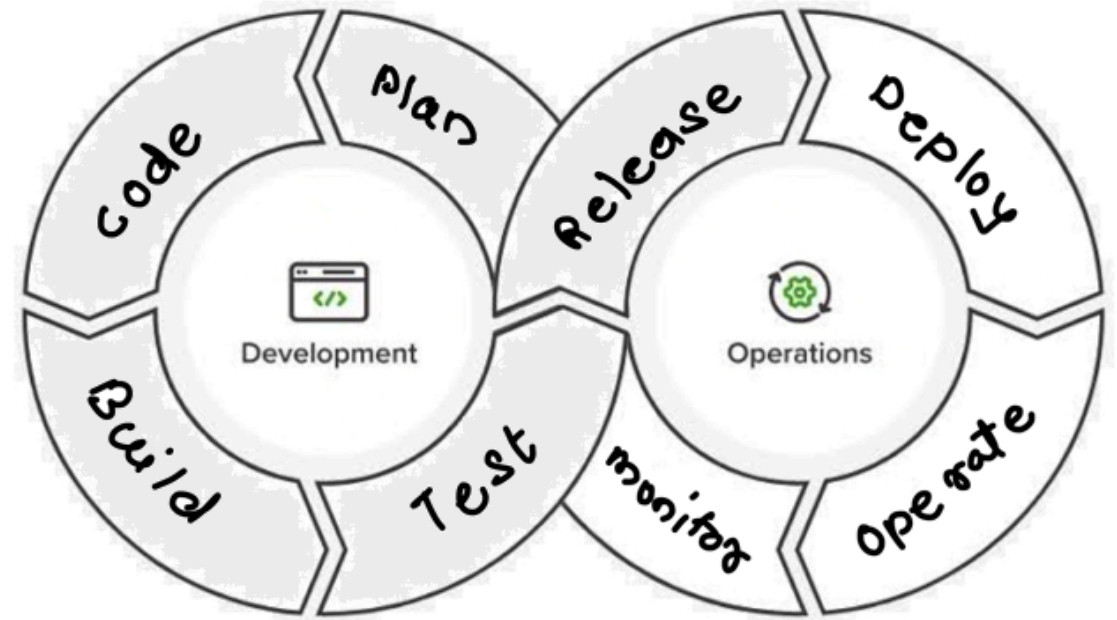
DevOps Lifecycle - Deploy

- Manage and maintain development and deployment of software systems and server in any computational environment

- Deployment tools
  - ✓ Docker / *VM* — *EC2 instance*
  - ✓ Kubernetes / *docker swarm*
    - Virtual Machines

- Configuration management tools
  - Puppet
  - Chef   *many machines*
 - Ansible

DevOps Lifecycle - Operate

- This stage where the updated system gets operated

- Operating Tools
  - Puppet
  - Chef
  - Ansible
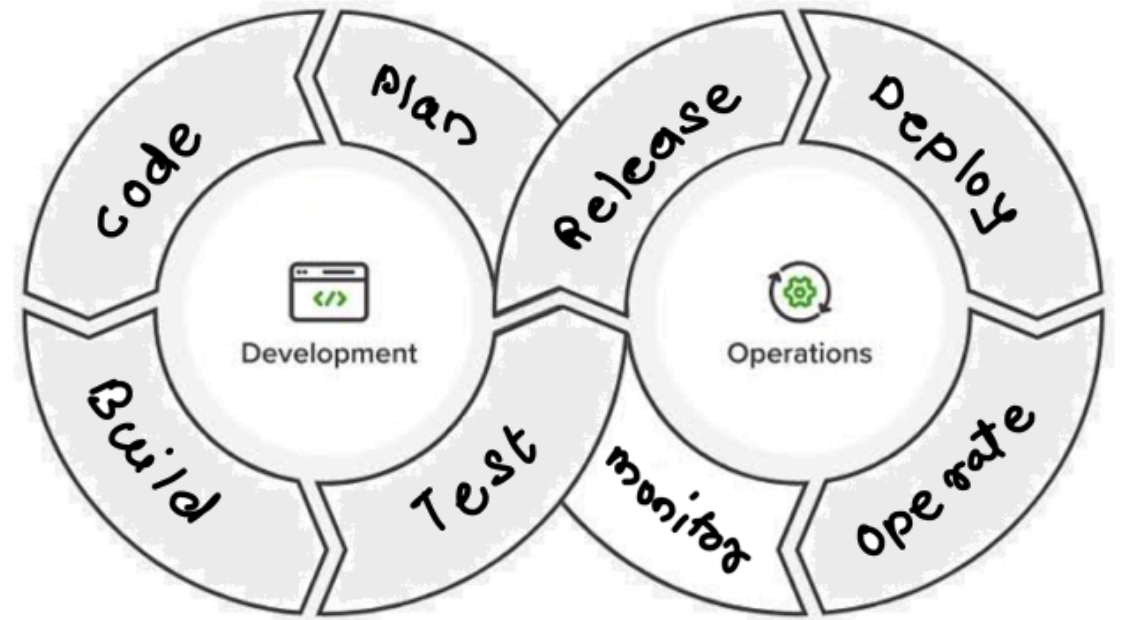
DevOps Lifecycle - Monitor

- It ensures that the application is performing as expected and the environment is stable

- It quickly determines when a service is unavailable and understand the underlying causes

- Monitoring tools
    - Nagios
    - Sensu
    - Splunk
    - DataDog
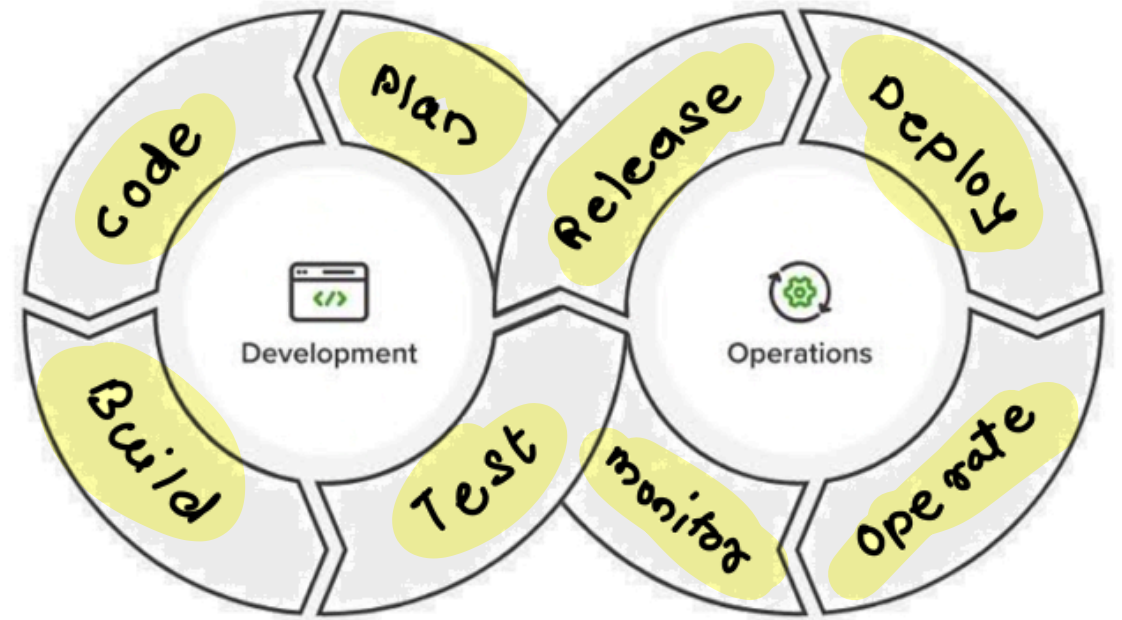
DevOps Terminologies

- Continuous Development
- Continuous Testing
- Continuous Integration
- Continuous Delivery
- Continuous Deployment
- Continuous Monitoring
- Continuous Learning