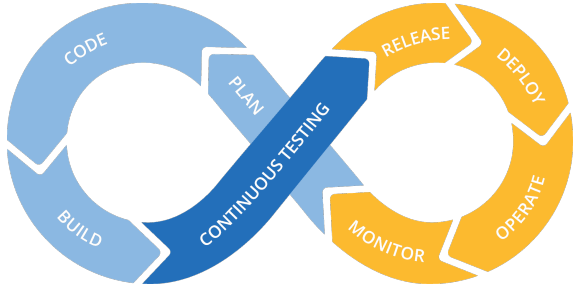

CODE
PLAN
BUILD
CONTINUOUS TESTING
RELEASE
DEPLOY
MONITOR
OPERATE

DevOps

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** (git, cvs, SUN, bazar)
- Incremental builds are difficult to manage, test and deploy: **Jenkins** (CI/CD pipeline)
- 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**

containerization

Kubernetes

Overview

Developer Tester

- DevOps is a combination of two words development and operations
- 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

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

Development

① develop / code

② testing

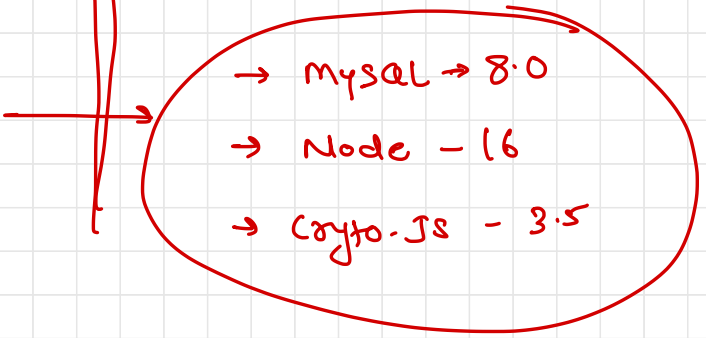
③ creating build

Express
- MySQL 8.1
- Crypto-JS - 4.0
- Node 17
- :

Operations

① managing resources
- machines
- HW
- network

② deployment of app



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 → dev - deployment
  - Continuously measuring application performance

* mindset
* culture

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

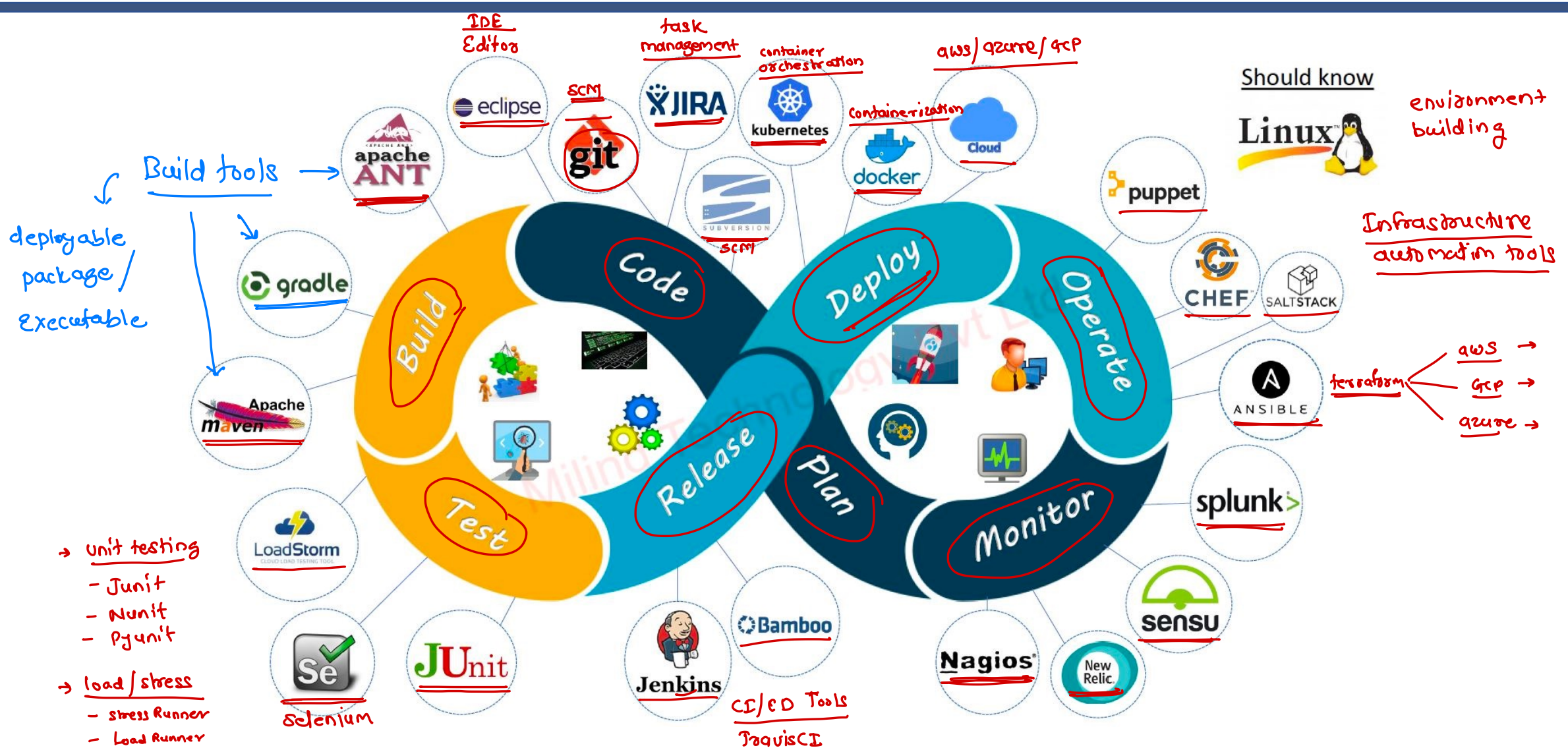
Reasons to use DevOps

- **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

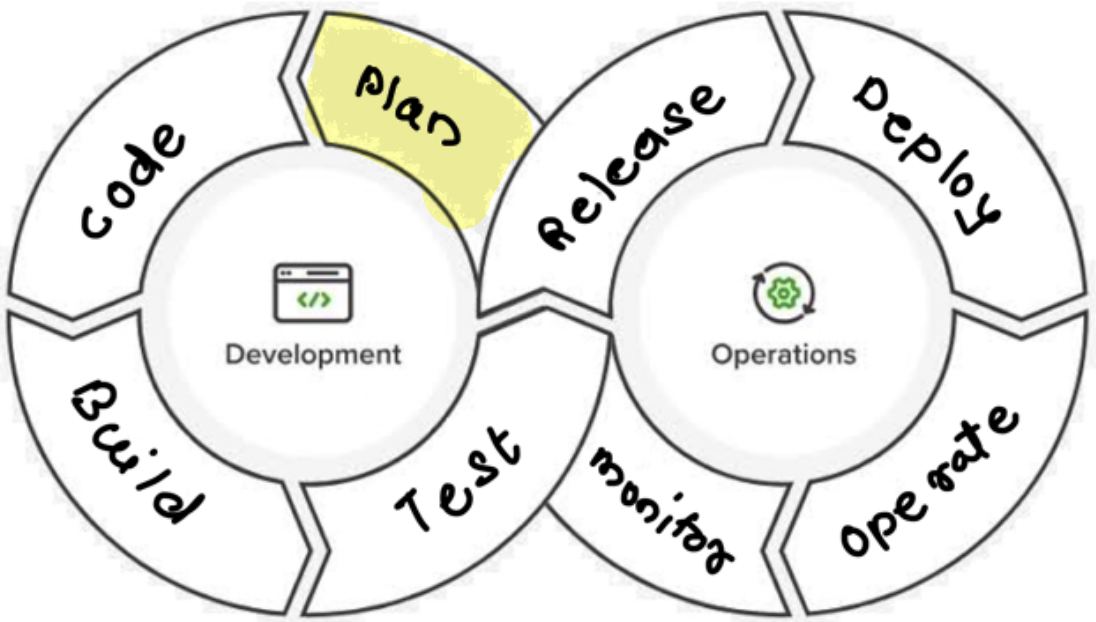
- **Cost Efficiency:** DevOps offers cost efficiency in the software development process which is always an aspiration of IT companies' management
- **Breaks larger code base into small pieces:** DevOps is based on the agile programming method. Therefore, it allows breaking larger code bases into smaller and manageable chunks

DevOps Lifecycle



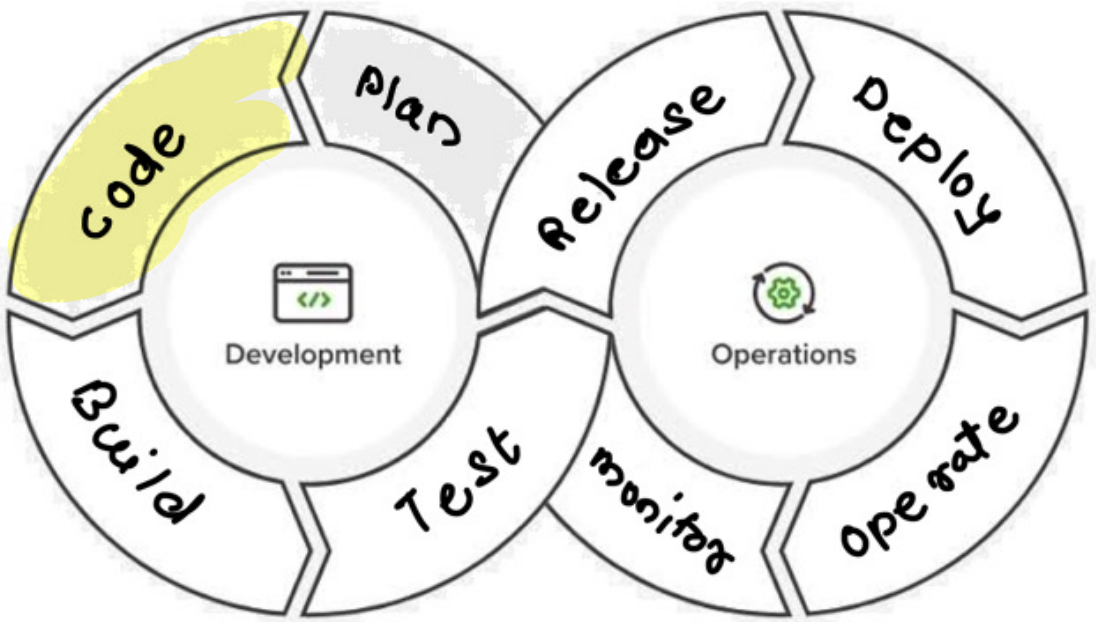
DevOps Lifecycle - Plan

- 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

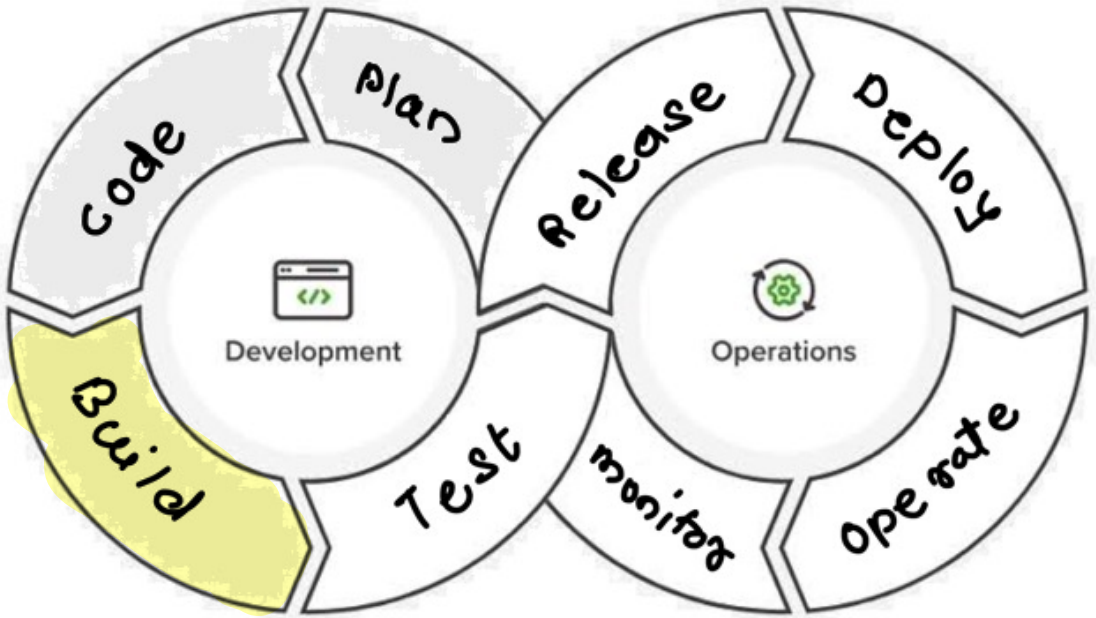
- Second stage where developer writes the code using favorite programming language
- Coding Tools
  - IDEs: Eclipse, Visual Studio etc.
  - SCM: Git, Subversion, CVS etc.
  - Package management: npm etc.



DevOps Lifecycle -Build

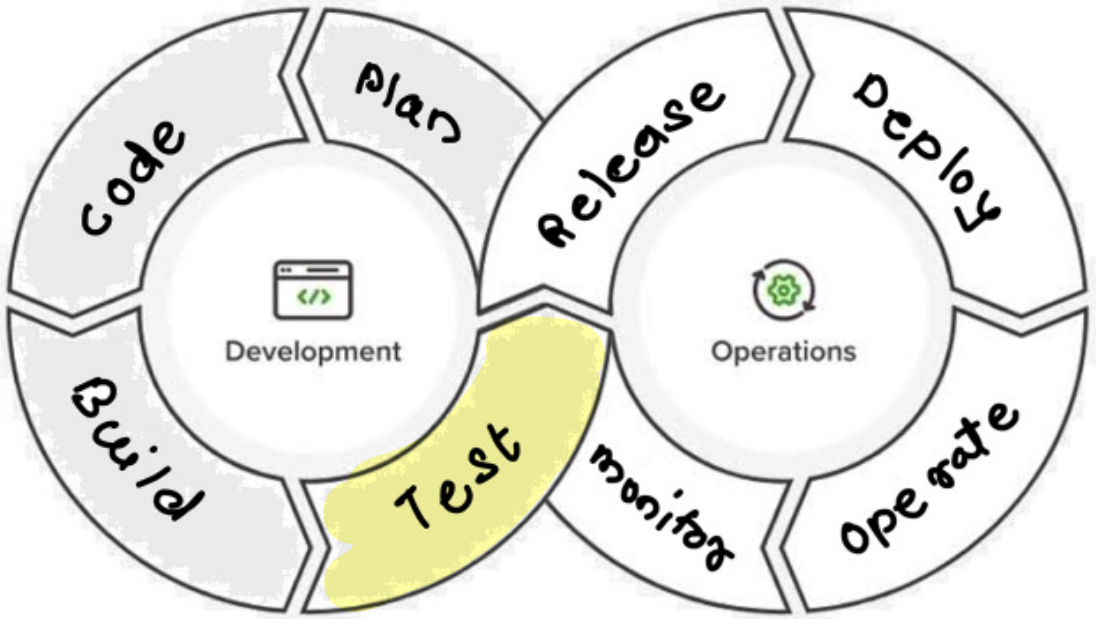
- Integrating the required libraries
- Compiling the source code
- Create deployable packages
- Build tools
  - Maven
  - Gradle
  - Ant

. libraries + compiled code ⇒ package



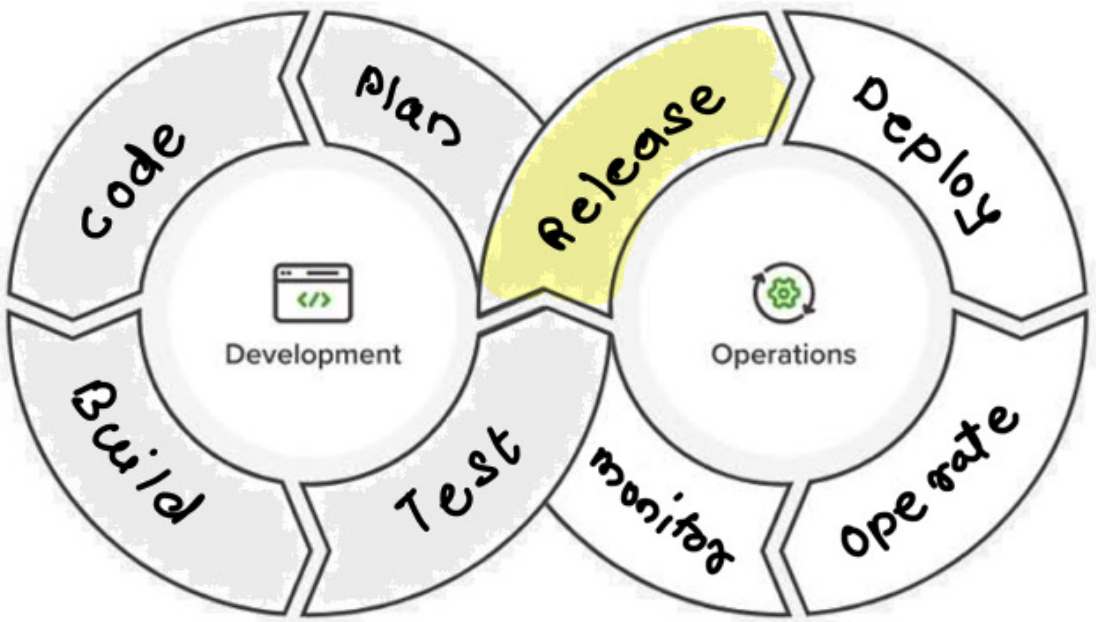
DevOps Lifecycle - Test

- Process of executing automated tests
- The goal here is to get the feedback about the changes as quickly as possible
- Testing tools
  - JMeter
  - Selenium
  - JUnit
  - QUnit
  - NUnit
  - Appium



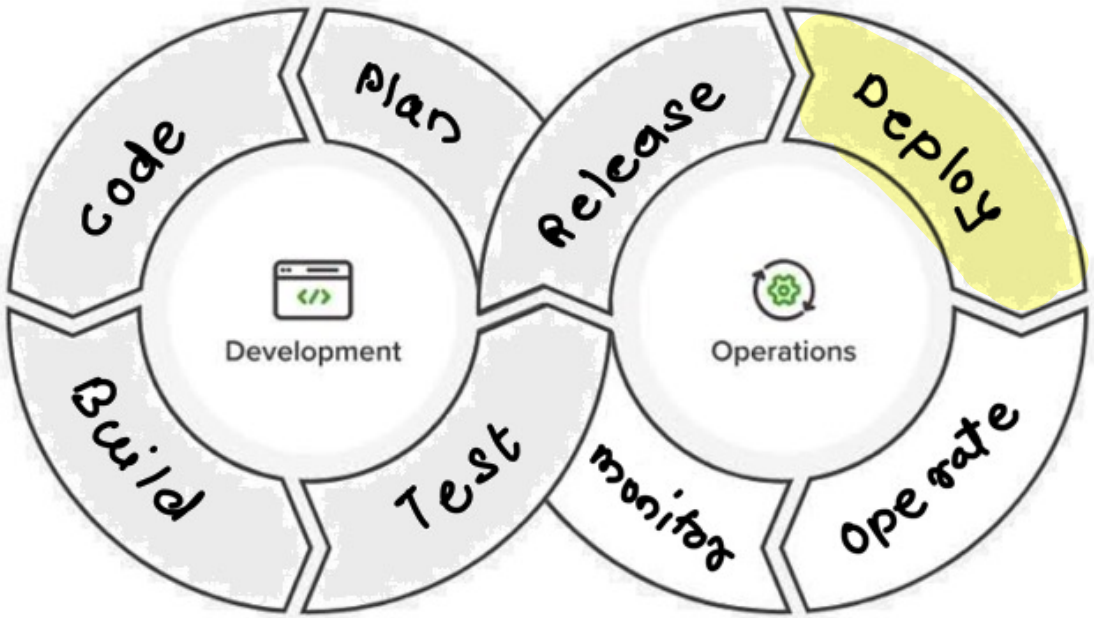
DevOps Lifecycle - Release

- 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

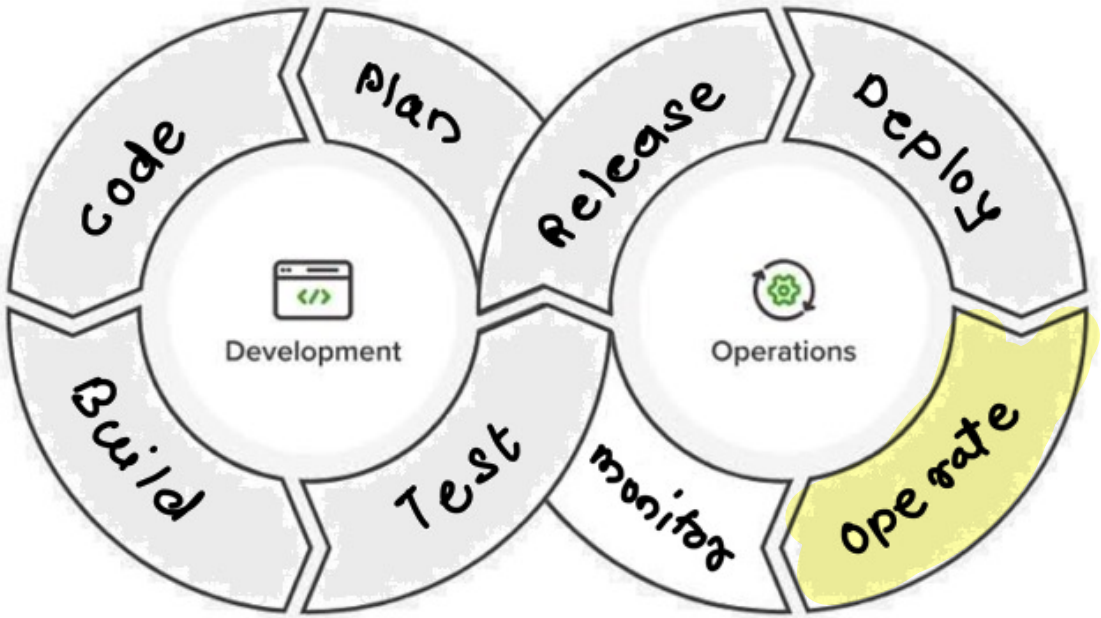
- Manage and maintain development and deployment of software systems and server in any computational environment
- Deployment tools
  - Docker
  - Kubernetes
  - Virtual Machines
- Configuration management tools
  - Puppet
  - Chef
  - Ansible



DevOps Lifecycle - Operate

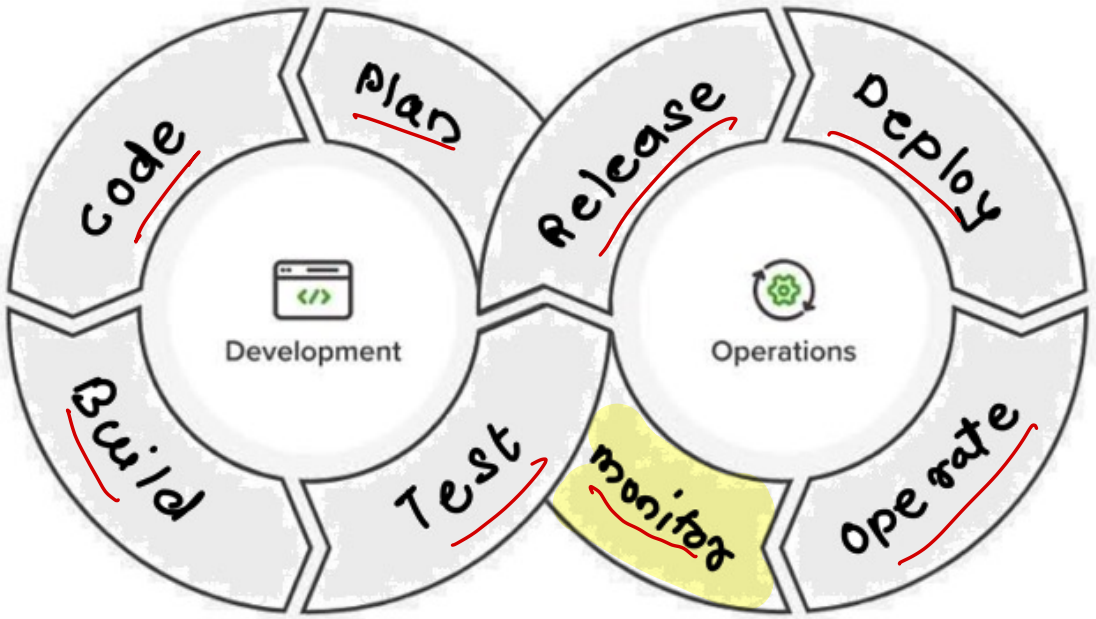
- This stage where the updated system gets operated
- Operating Tools
  - Puppet
  - Chef
  - Ansible

Configuration management



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