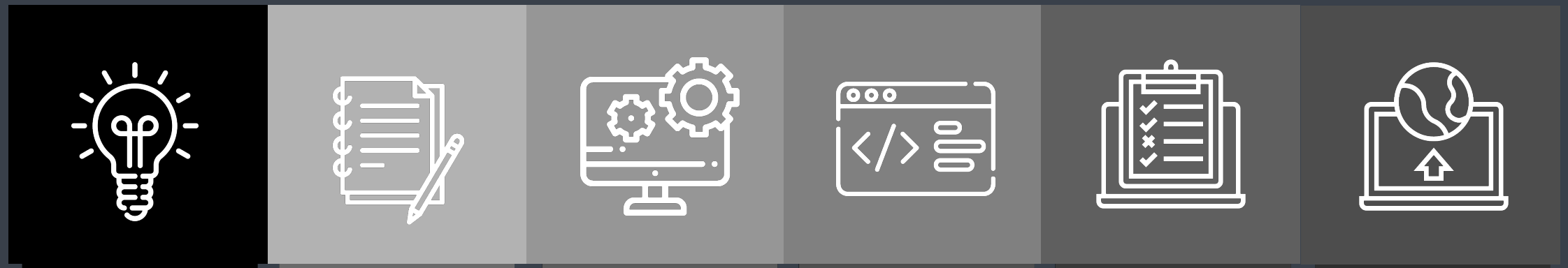
automation

Extension to Agile

DevOps

Software Development Lifecycle



PLANNING

Talk to customer and understand the requirements

Requirements

DEFINING

Define the requirements and stick to them

SRS

DESIGNING

Design the solution with right approach

DDS

BUILDING

Development following guidelines

Software

TESTING

Make sure that your code is working

Well tested software

DEPLOYMENT

Make your app available for rest of the world

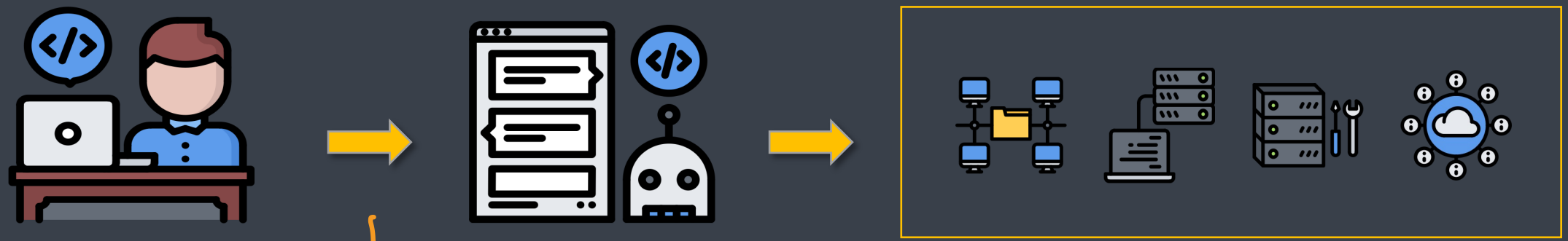
deployed application

Waterfall Model

- Requirement Specification
- System Design
- Design Implementation
- Verification & Testing
- System Deployment
- Software Maintenance

Entities involved

Dev team



Developer

Testers

Operations Team

UI/UX designers

Responsibilities

Developers and Testers

- Developers
  - Develop the application
  - Package the application
  - Fix the bugs
  - Maintain the application
- Testers
  - Thoroughly test the application manually or using test automation
  - Report the bugs to the developer

package
- binary / executable / code
- libraries / packages / frameworks
- resources
- documentation

package
- React → bundle
- android → apk
- ios → ipa
- desktop → .exe, .msi, .deb, .rpm, .dmg

Operations Team

- Make all the necessary resources ready
- Deploy the application
- Maintain multiple environments
- Continuously monitor the application
- Manage the resources

environment
- resources needed to run / develop the app
- OS, platform, db, ws, runtime, networking resources
- development, staging, pre-prod, prod

Challenges

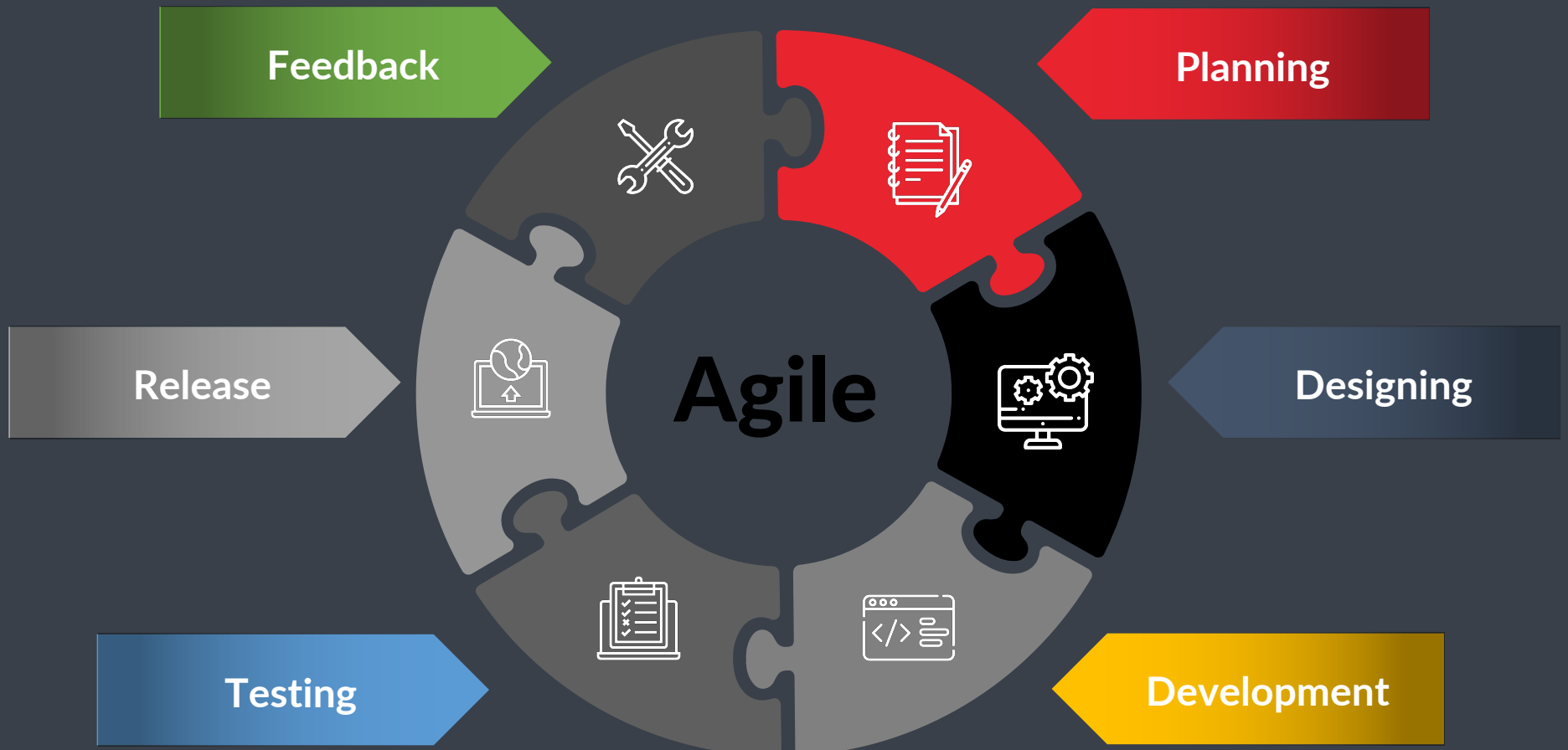
Developers and Testers

- The process is slow
- The pressure to work on the newer features and fix the older code
- Not flexible

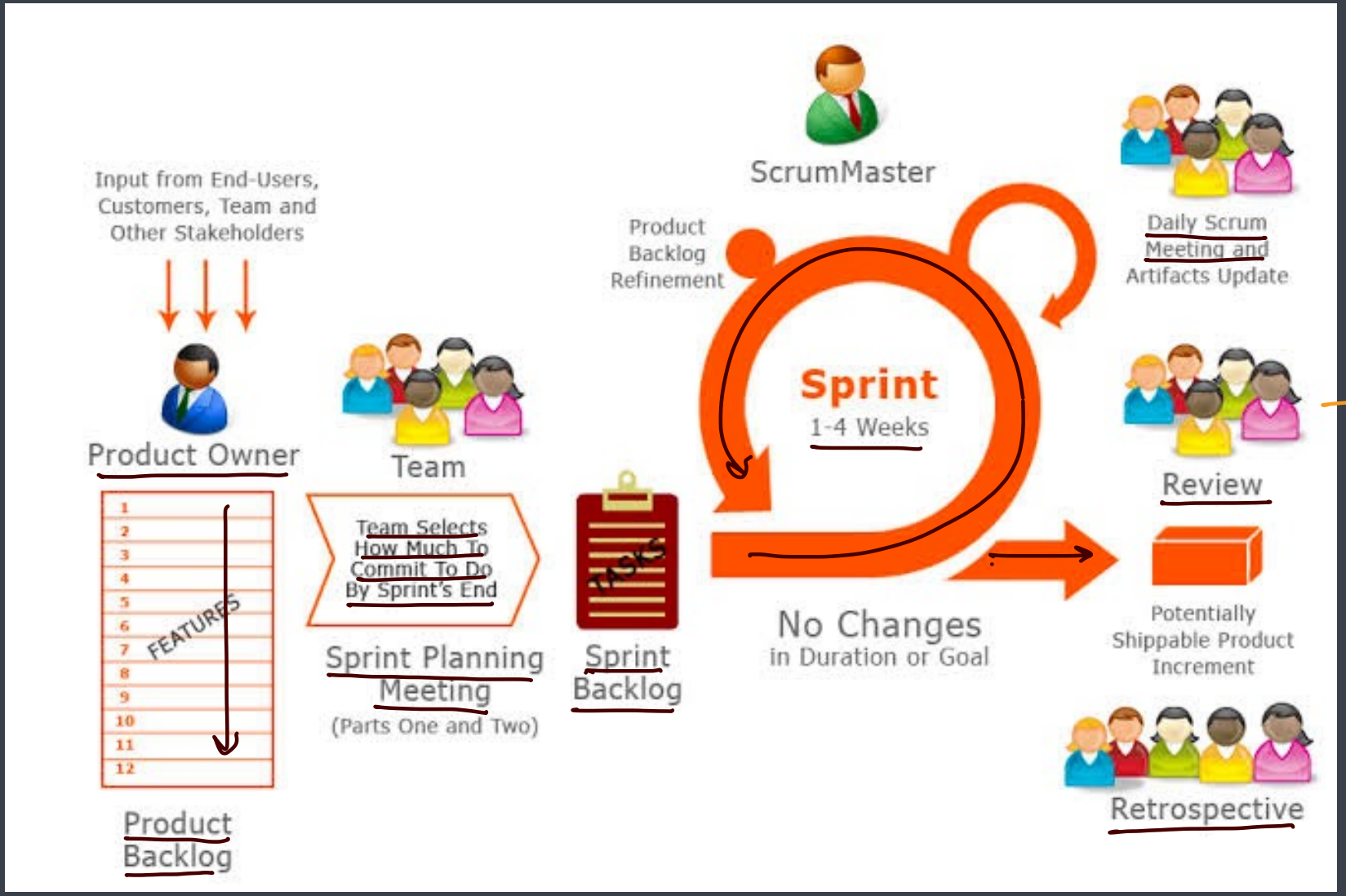
Operations Team

- Uptime
- Configure the huge infrastructure
- Diagnose and fix the issue

Agile Development



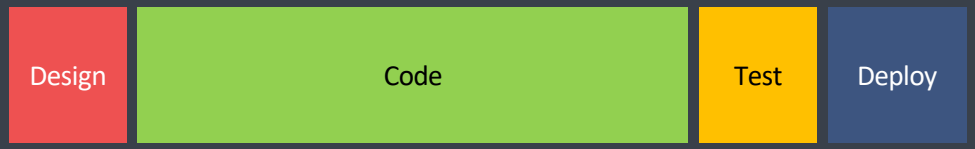
Scrum Process



Waterfall Vs Agile

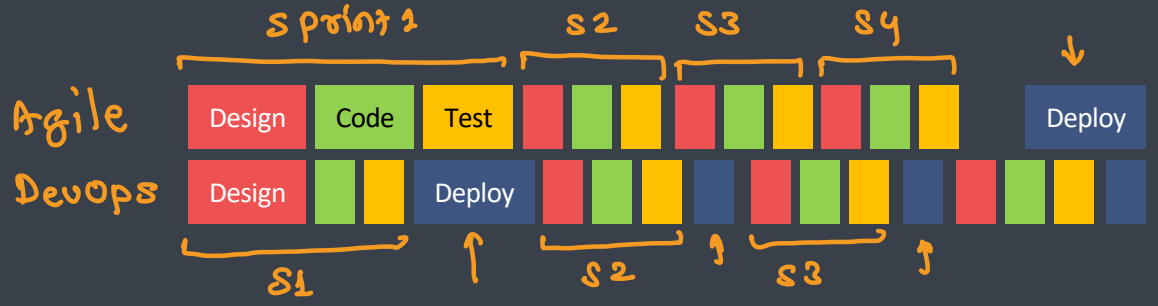
The Waterfall Process

This project has got so big.
I am not sure I will be able to deliver it!



The Agile Process

It is so much better delivering
this project in bite-sized sections



Problems

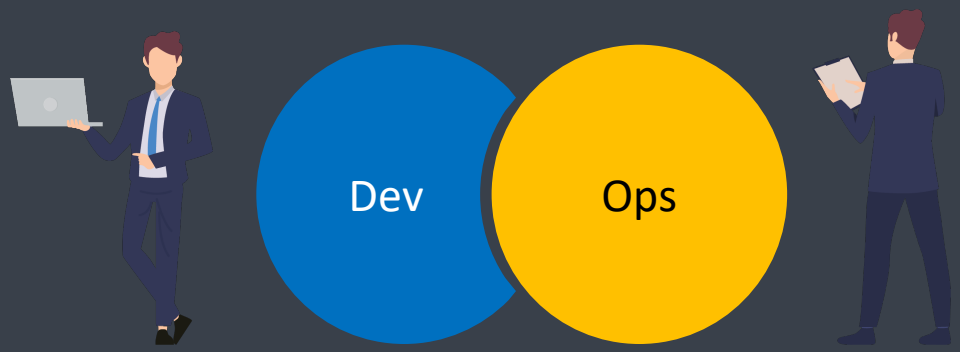
- Managing and tracking changes in the code is difficult
- Incremental builds are difficult to manage, test and deploy
- Manual testing and deployment of various components/modules takes a lot of time
- Ensuring consistency, adaptability and scalability across environments is very difficult task
- Environment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

Solutions to the problem

- Managing and tracking changes in the code is difficult: **SCM tools** → SVN, CVS, git, bazaar
- 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** → automated testing
- Ensuring consistency, adaptability and scalability across environments is very difficult task: **Puppet** → continuous configuration
- Environment dependencies makes the project behave differently in different environments: **Docker** → Containerization & container orchestration

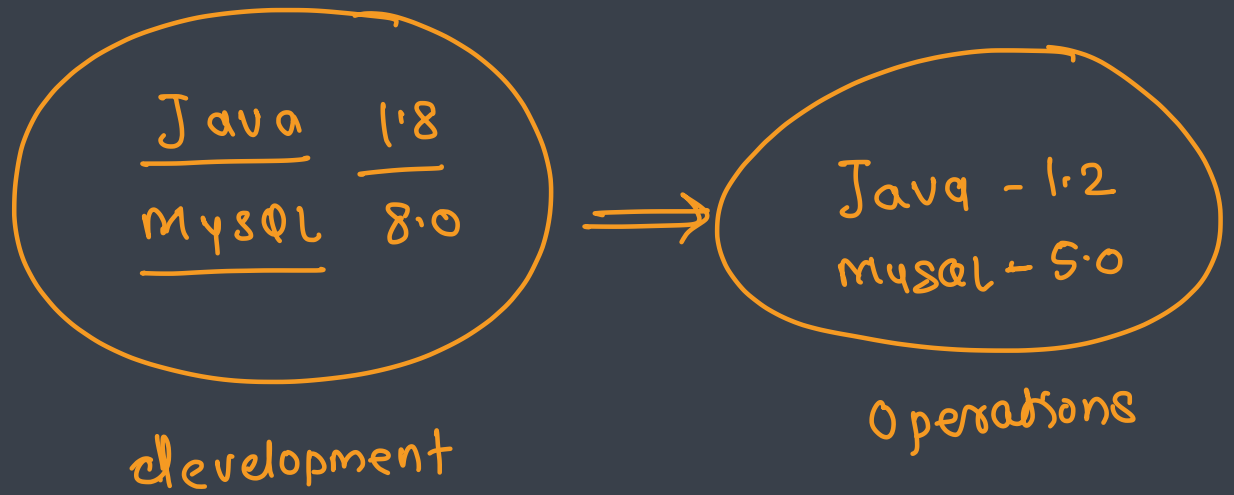
What is DevOps ?

- 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
- 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 - monitoring



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



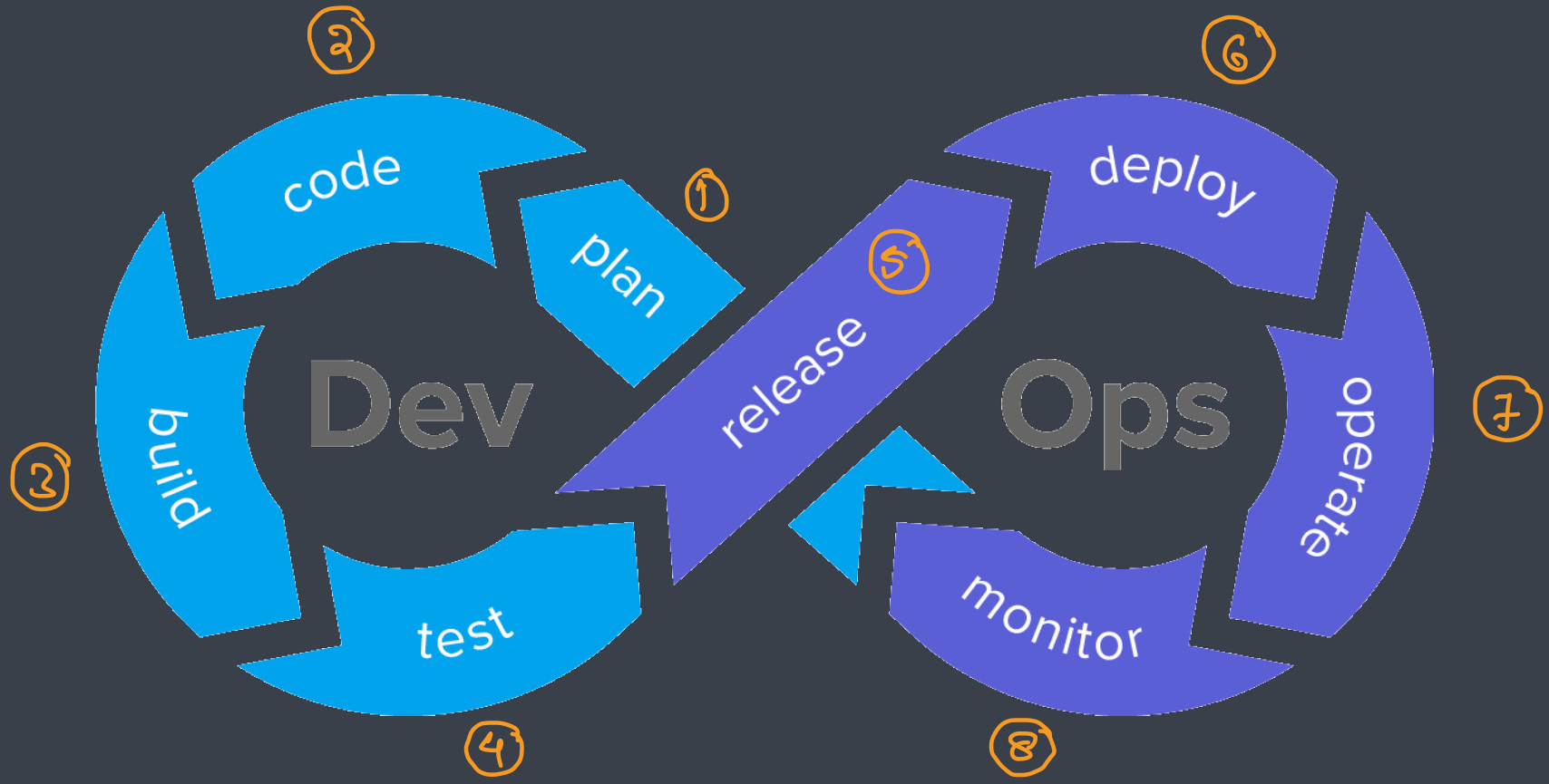
Common misunderstanding

- 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

DevOps Lifecycle



DevOps Lifecycle - Plan

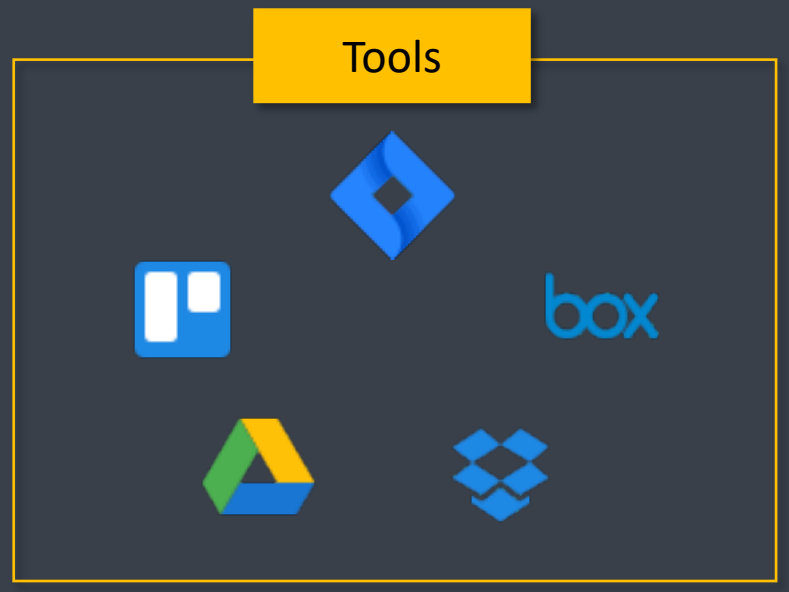
ticket = story = task

Jira

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

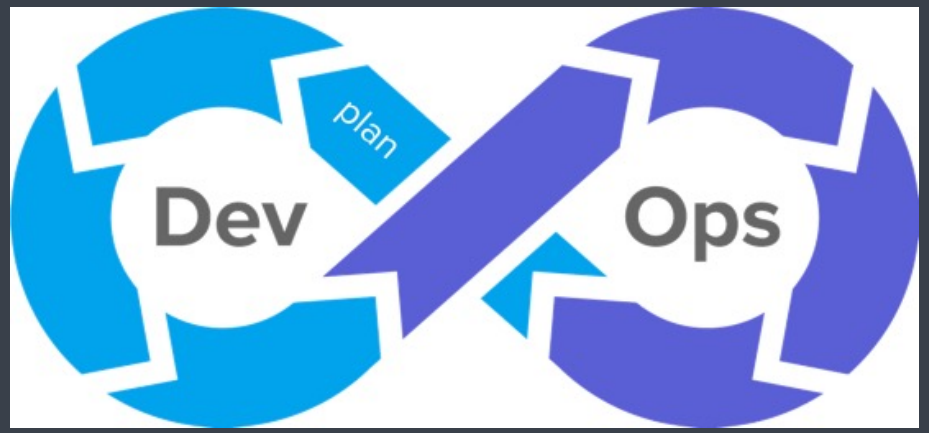
tools : text file , Excel , Google sheet

box, onedrive, googledrive



Specialized tools

- Jira , Confluence
- Trello
- GitScrum
- OrangeScrum



DevOps Lifecycle - Code

Git

- Second stage where developer writes the code using favorite programming language

languages
- C, C++, C#
- JS / TS
- python / pearl / PHP
- Go
- ruby

Editors
- vim
- vscode
- atom
- sublime

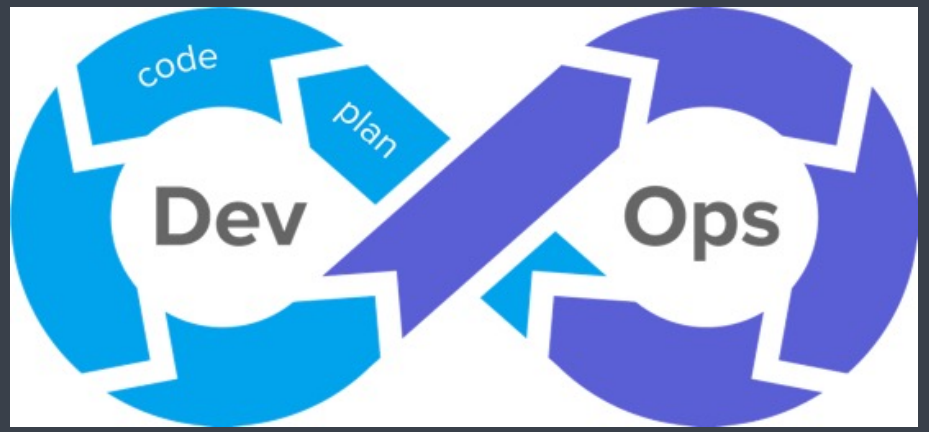
IDEs
- VS
- Webstorm
- IDEA
- PHPStorm
- Android studio
- XCode

Debugger
- Java - jdb
- C, C++ → gdb
- python - pydb
- JS - browser

scm tools
- cvs
- svn
- git **

package manager
- Node - npm, yarn, pnpm
- ios - pod
- C# - chocolate
- python - pip

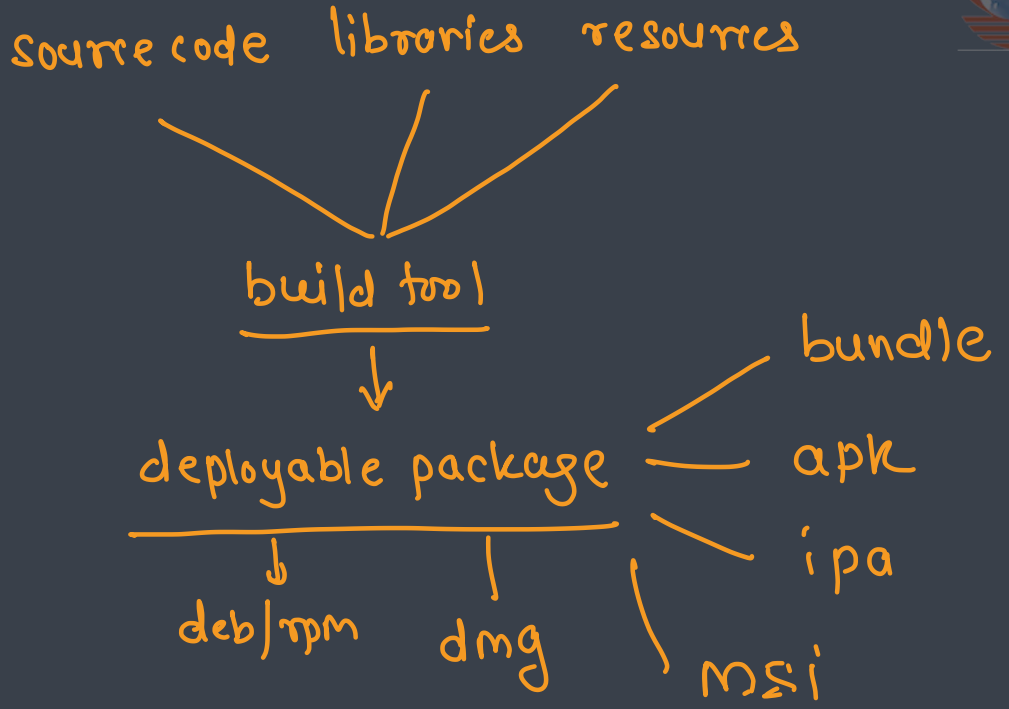
Tools

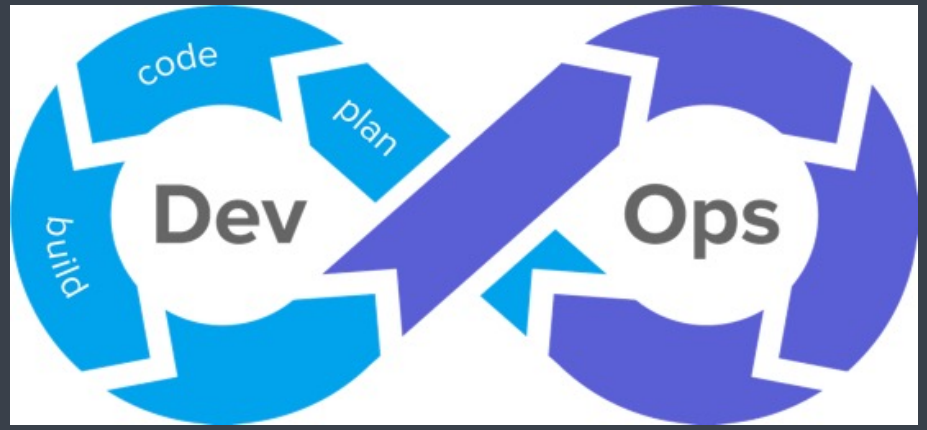


DevOps Lifecycle -Build

- Integrating the required libraries
- Compiling the source code
- Create deployable packages

tools
- ant - deprecated
- maven
- gradle





DevOps Lifecycle - Test

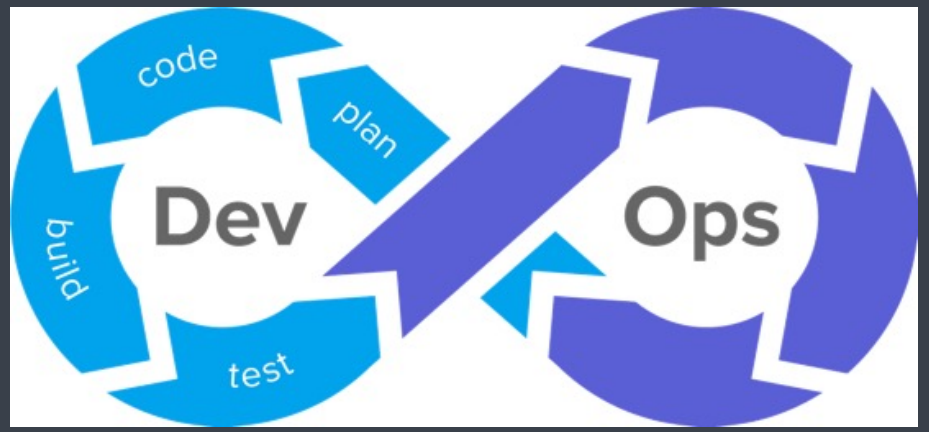
Selenium

- Process of executing automated tests
- The goal here is to get the feedback about the changes as quickly as possible

* Unit testing
  - Js/TS - jasmine/jest
  - python - PyUnit
  - C# - NUnit
  - Java - Junit

* UI testing
  - Selenium
  - Cypress

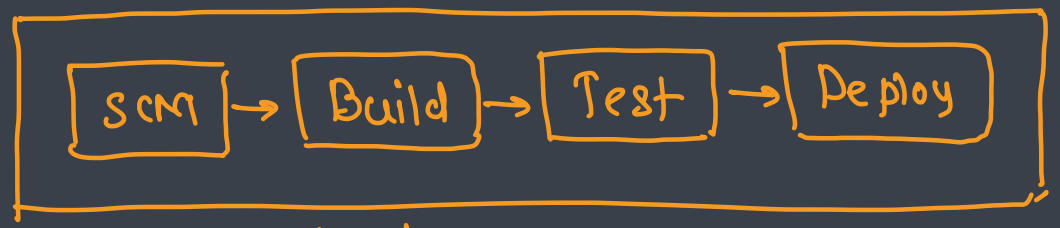
Tools



DevOps Lifecycle - Release → CI/CD pipeline [sequence & stages]

Jenkins

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

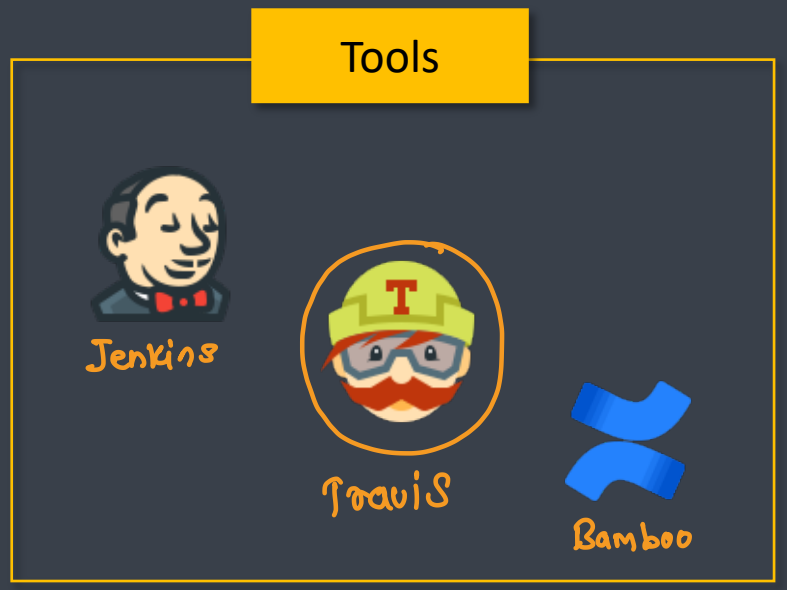


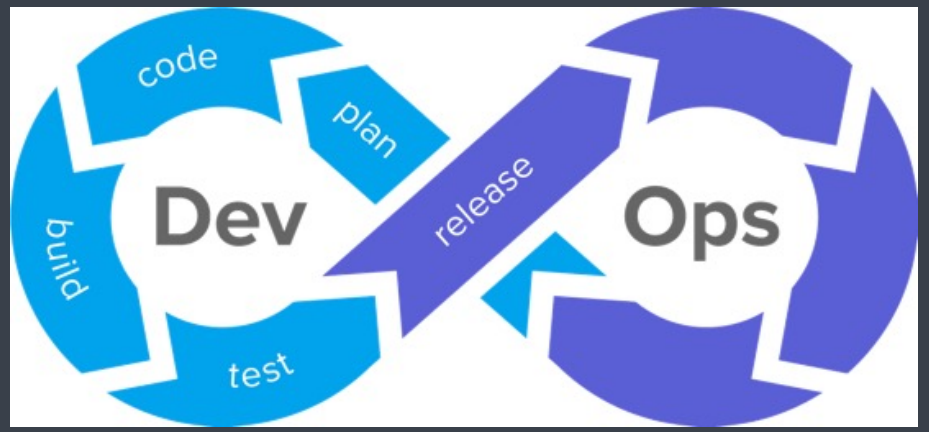
CI/CD pipeline

CI tools – TravisCI, GitlabCI, GitHub Actions

CD tools – ArgoCD

CI/CD tools → Jenkins, Bamboo





DevOps Lifecycle - Deploy

Docker & K8S

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

virtualized deployment

- On Premise → VMWare, Virtual Box, Parallels
- Cloud → AWS, Azure, GCP, Digital Ocean, Rackspace, Alibaba Cloud, IBM Cloud

Tools

docker

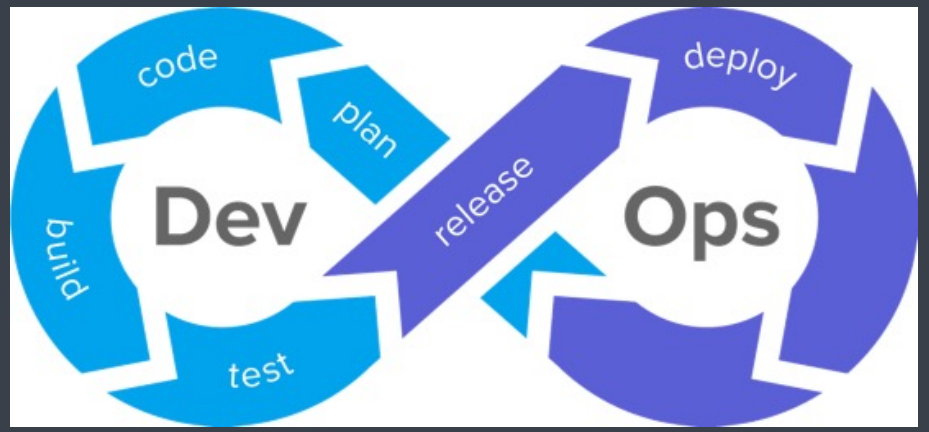
VBox

Kubernetes

vmware

containerized deployment

- container tools
  - docker, podman, rocket, LMCIFY (Google)
- container orchestration
  - docker swarm, kubernetes, mesos, marathon

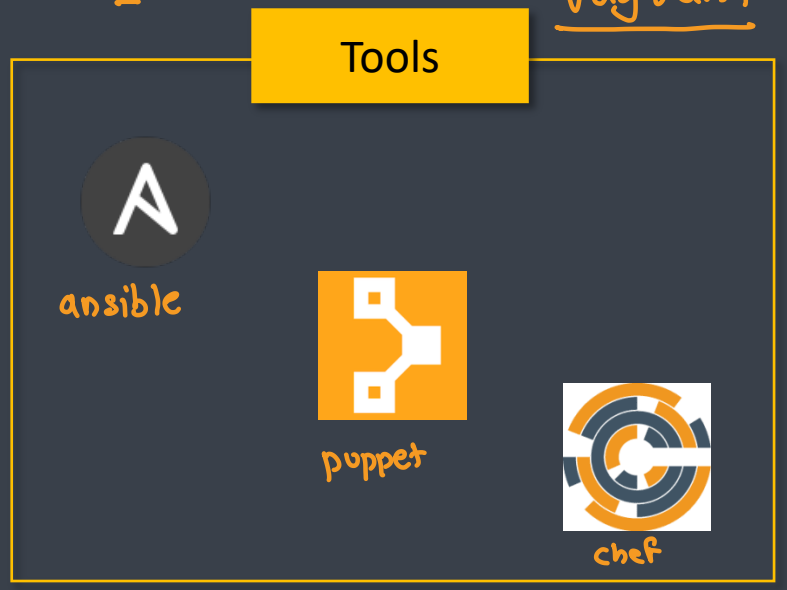


DevOps Lifecycle - Operate

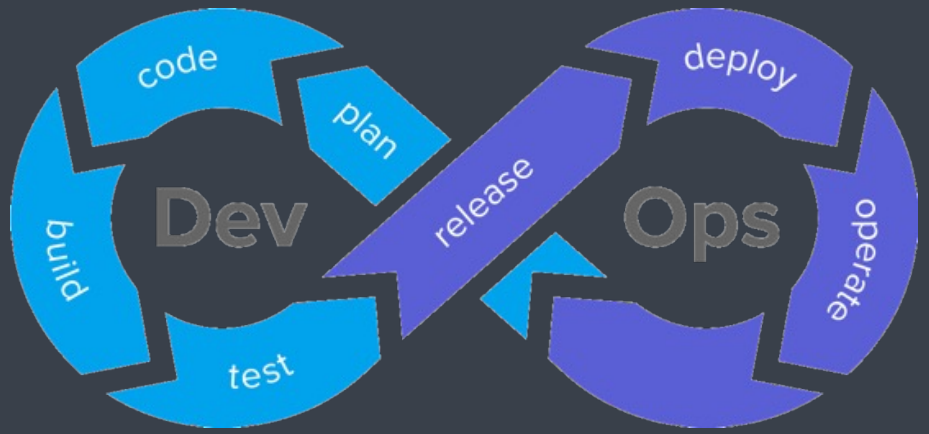
- This stage where the updated system gets operated

tools to configure enviroment

- cloud agnostic tools for enviroment creation
  - terraform, ansible
- local vm management tools
  - vagrant



* environment configuration tools
  - puppet, chef, ansible, saltstack
* cloud specific
  - aws - cloud formation



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

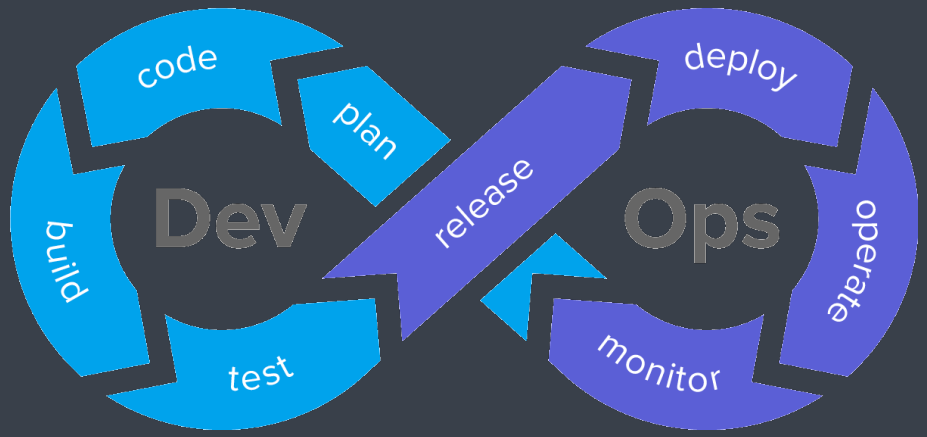
tools - DataDog, Nagios, Splunk

Tools

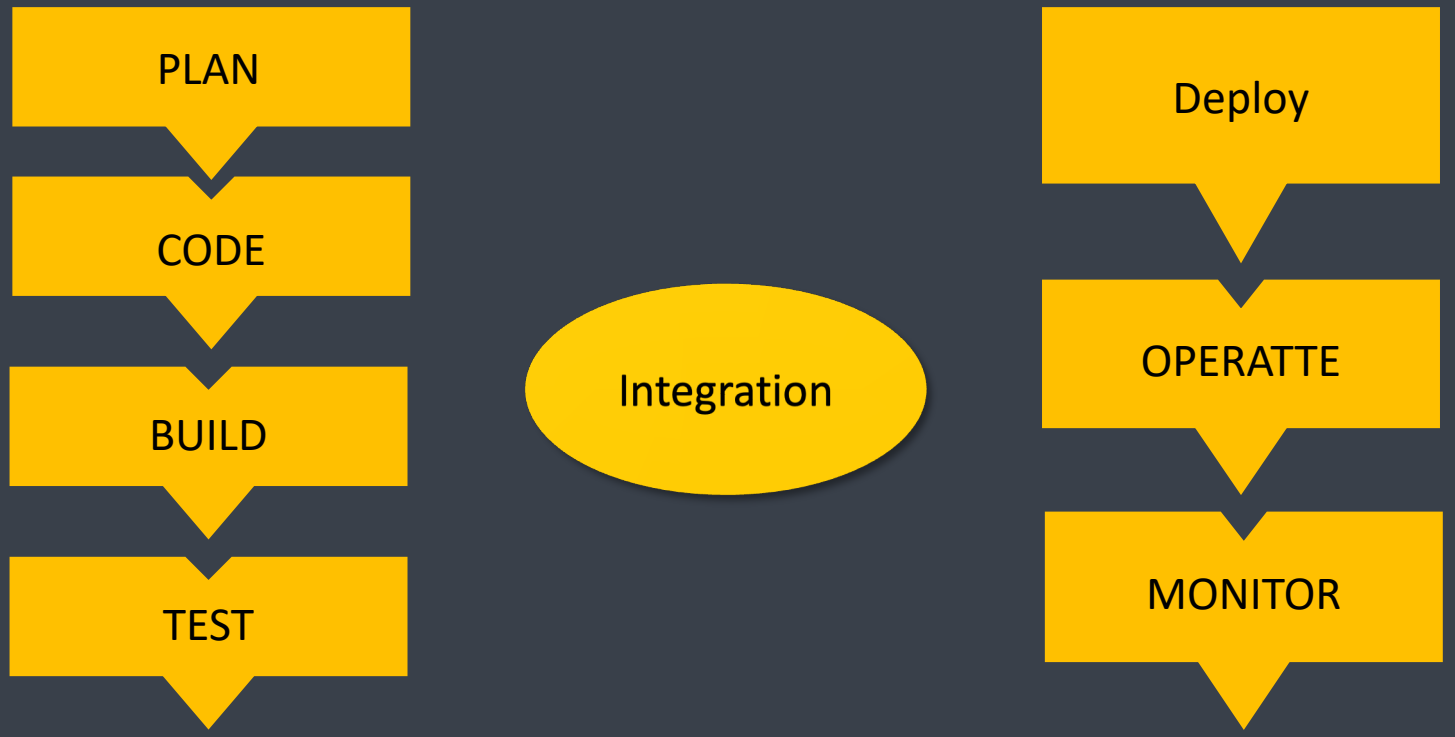
splunk>

DATADOG

Nagios®

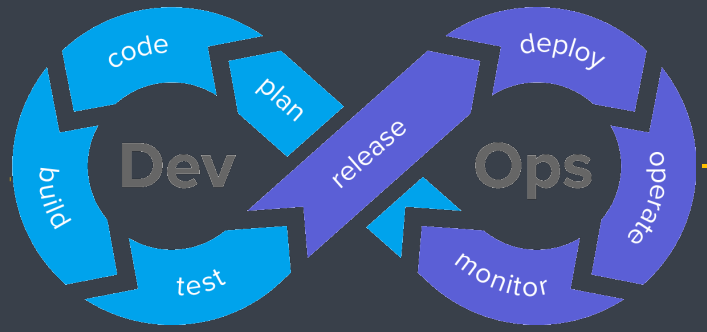


DevOps Terminologies



Responsibilities of DevOps Engineer

- Be an excellent sysadmin
- Deploy Virtualization
- Hands-on experience in network and storage
- Introduction to coding



- Soft skills
- Automation tools
- Software Testing knowledge
- IT security

Skills of a DevOps Engineer

Skills	Description
Tools	• Version Control – Git/SVN • Continuous Integration – Jenkins • Virtualization / Containerization – Docker/Kubernetes • Configuration Management – Puppet/Chef/Ansible • Monitoring – Nagios/Splunk
Network Skills	• General Networking Skills • Maintaining connections/Port Forwarding
Other Skills	• Cloud: AWS/Azure/GCP • Soft Skills • People management skill