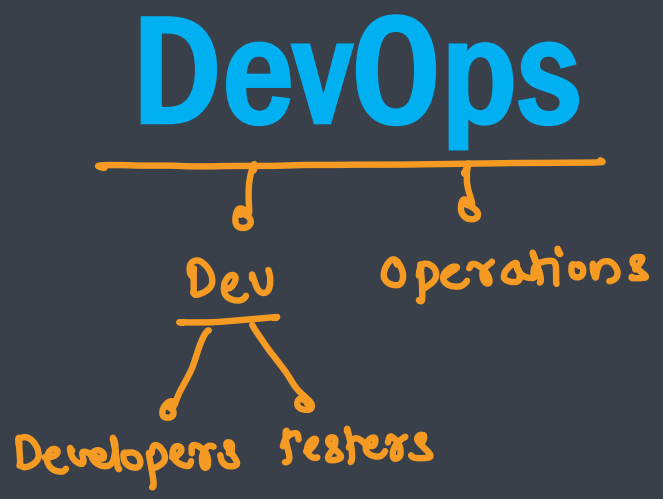
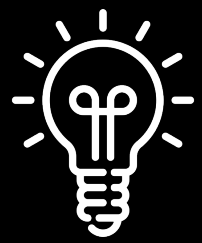

DevOps
Dev
Operations
Developers
Testers

Software Development Lifecycle

Requirements → SRS → DDS → product → well-tested product → product running in production Enviroment



PLANNING

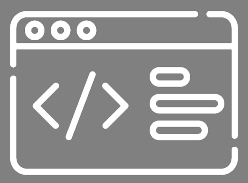
Talk to customer and understand the requirements

DEFINING

Define the requirements and stick to them

DESIGNING

Design the solution with right approach



BUILDING

Development following guidelines

TESTING

Make sure that your code is working

DEPLOYMENT

Make your app available for rest of the world

+ Maintenance

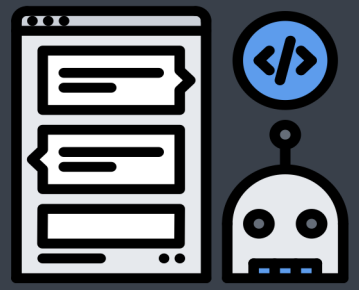
Waterfall Model

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

Entities involved

Developer

development/ programming/ coding



Testers

test the product

Operations Team

Responsibilities

android → .apk
ios → .ipa
website → webpack vite

windows native → .msi
Linux native → .deb / .rpm
macos native → .dmg

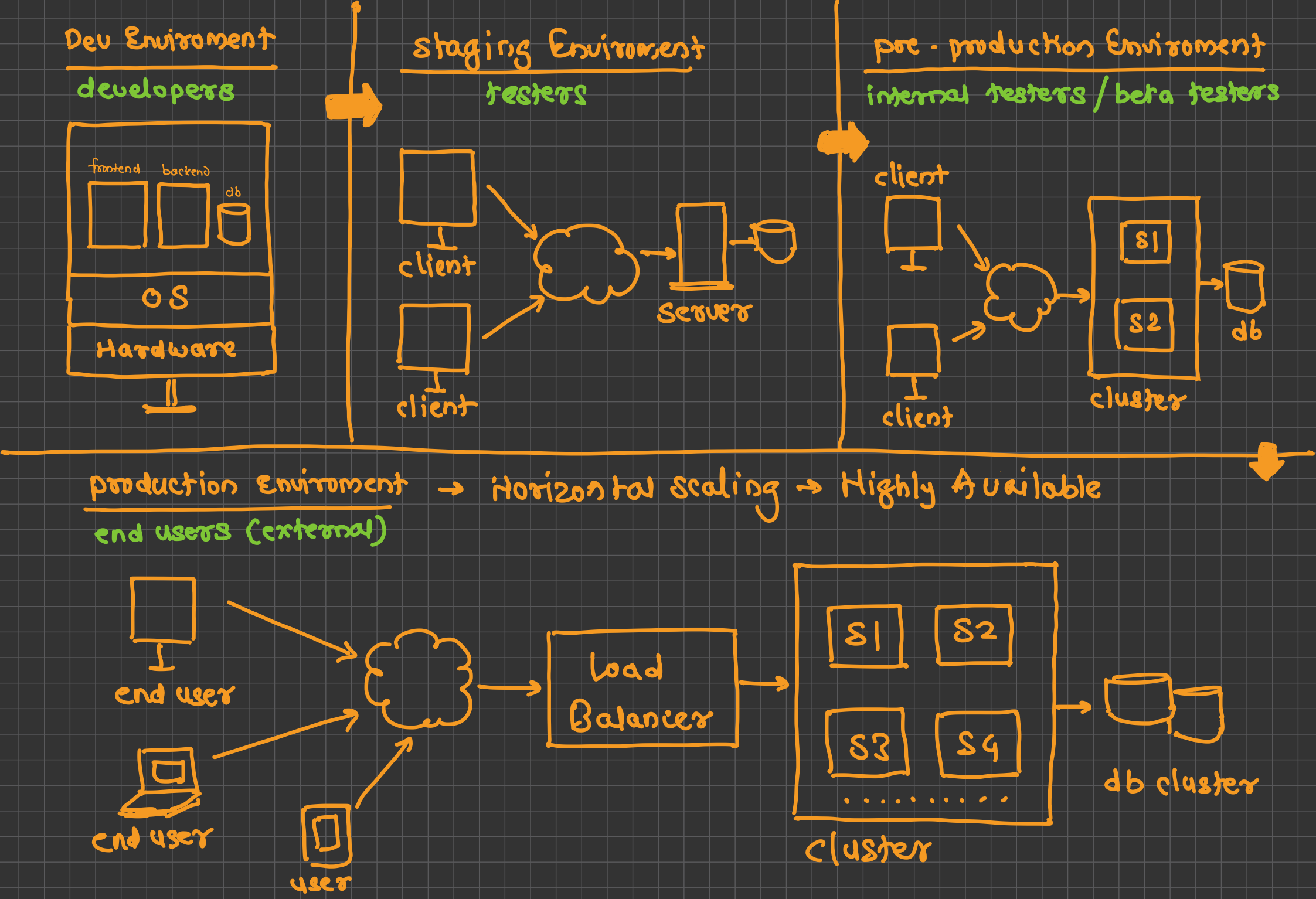
Dev Team

Developers and Testers

- Developers → coding
  - 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

Operations Team

- Make all the necessary resources ready → moving app from one to another environment
  - → computers
  - → software
  - → Licenses
  - → Networking
  - → Hardware
- Deploy the application
- Maintain multiple environments
- Continuously monitor the application
- Manage the resources


Dev Environment
developers
frontend
backend
db
OS
Hardware
staging Environment
testers
client
client
server
pre - production Environment
internal testers / beta testers
client
client
S1
S2
db
cluster
production Environment → Horizontal scaling → Highly Available
end users (external)
end user
end user
user
Load Balancer
S1
S2
S3
S4
cluster
db cluster

Challenges

Developers and Testers

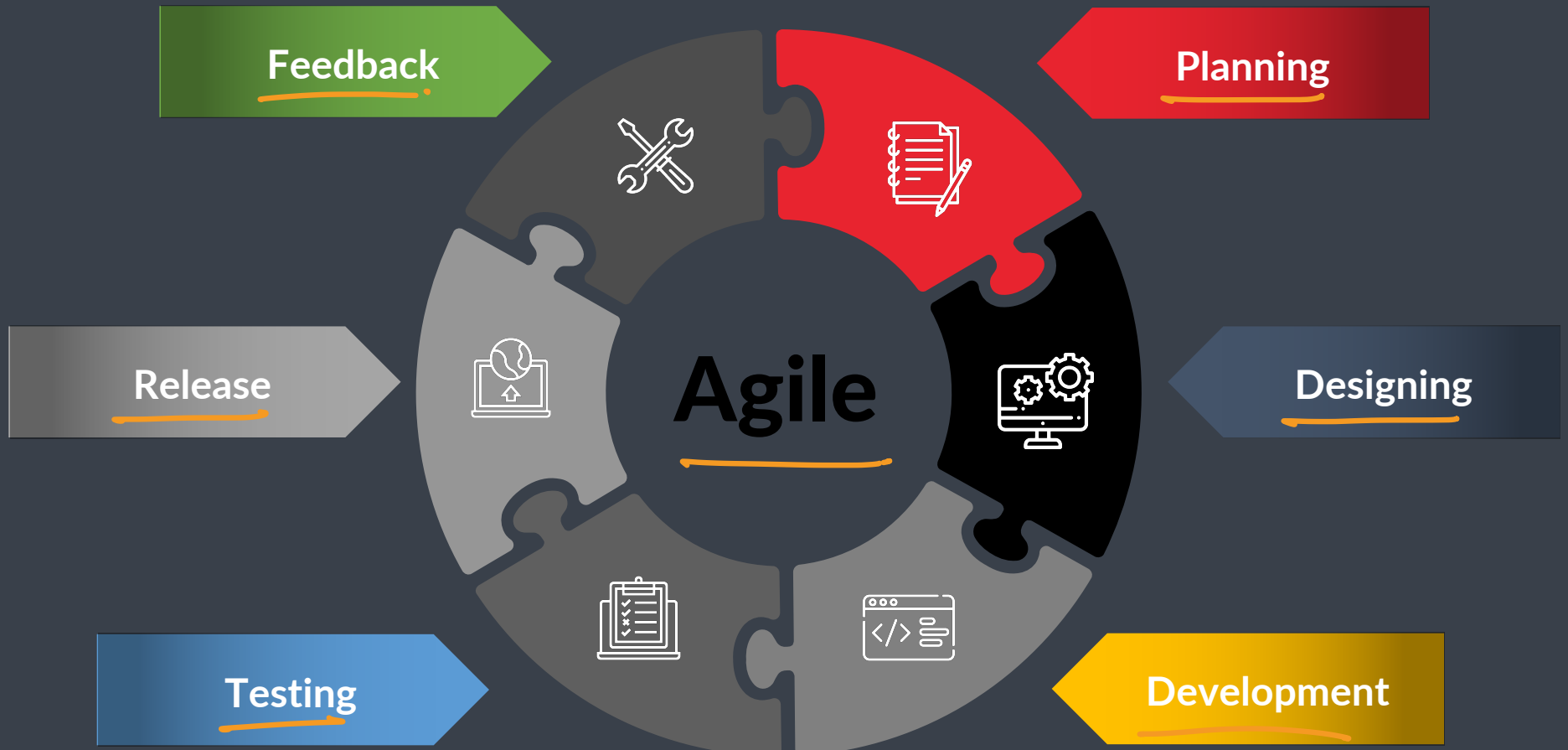
- The process is slow
- The pressure to work on the newer features and fix the older code
- Not flexible

Operations Team

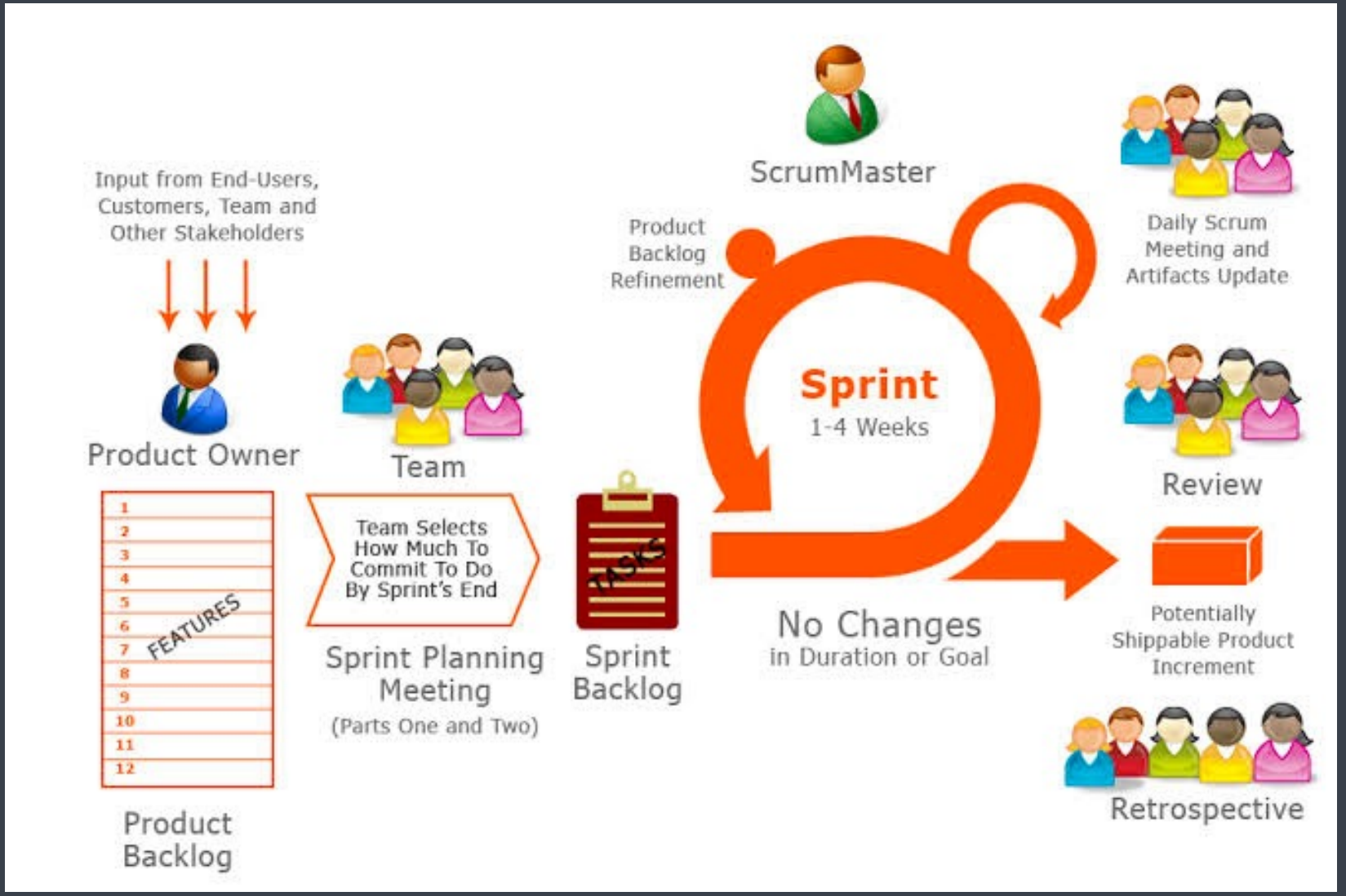
- Uptime x downtime
- Configure the huge infrastructure
- Diagnose and fix the issue

Agile Development

sprint



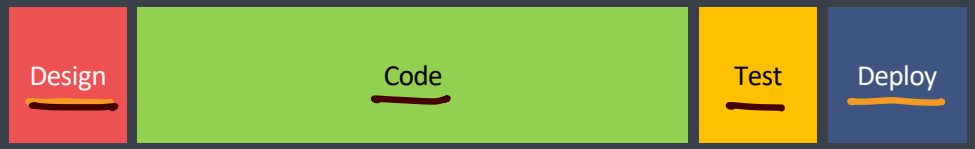
Scrum Process



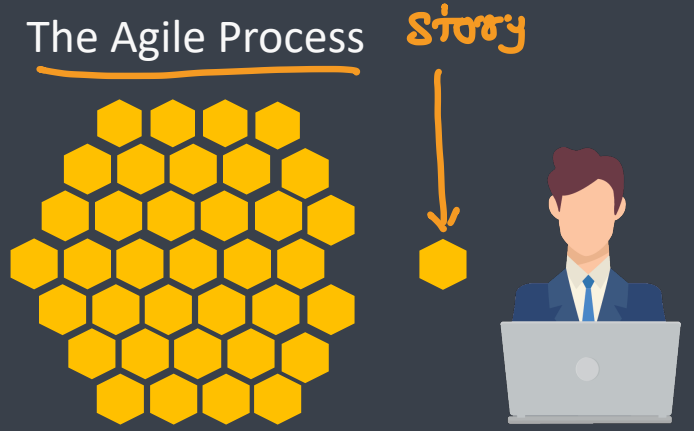
Waterfall Vs Agile

The Waterfall Process

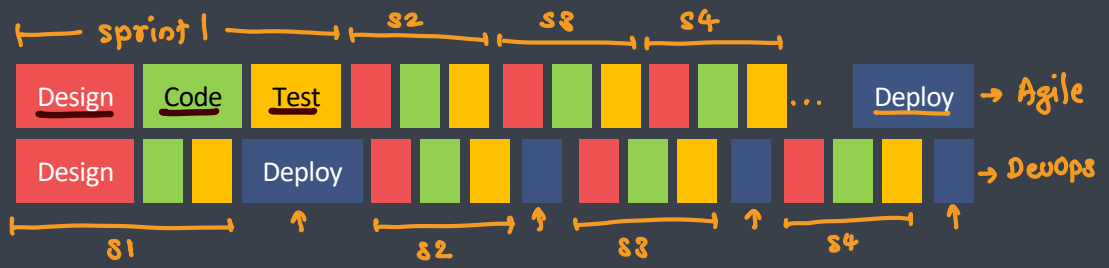
This project has got so big.
I am not sure I will be able to deliver it!



The Agile Process

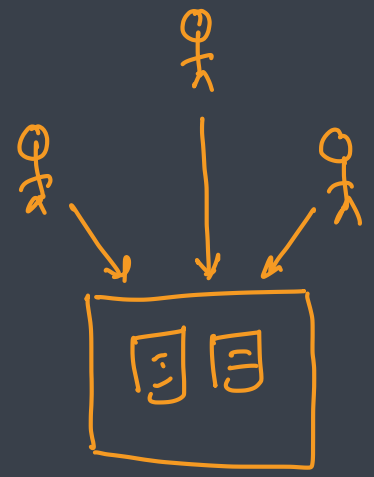


It is so much better delivering
this project in bite-sized sections → stories



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



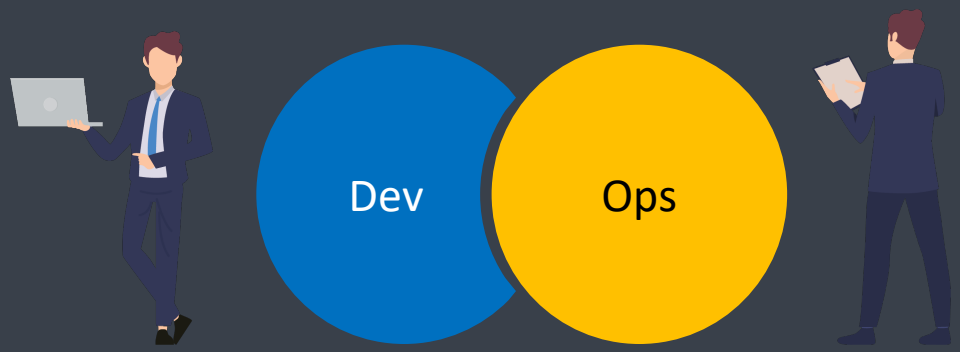
Solutions to the problem

- Managing and tracking changes in the code is difficult: **SCM tools** → git
- 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 config
- Environment dependencies makes the project behave differently in different environments: **Docker** → containerization

What is DevOps ?

dev testing ops

- 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 → environments / resources
  - Automating workflow → building + testing + deployment
  - Continuously measuring application performance → monitoring uptime



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

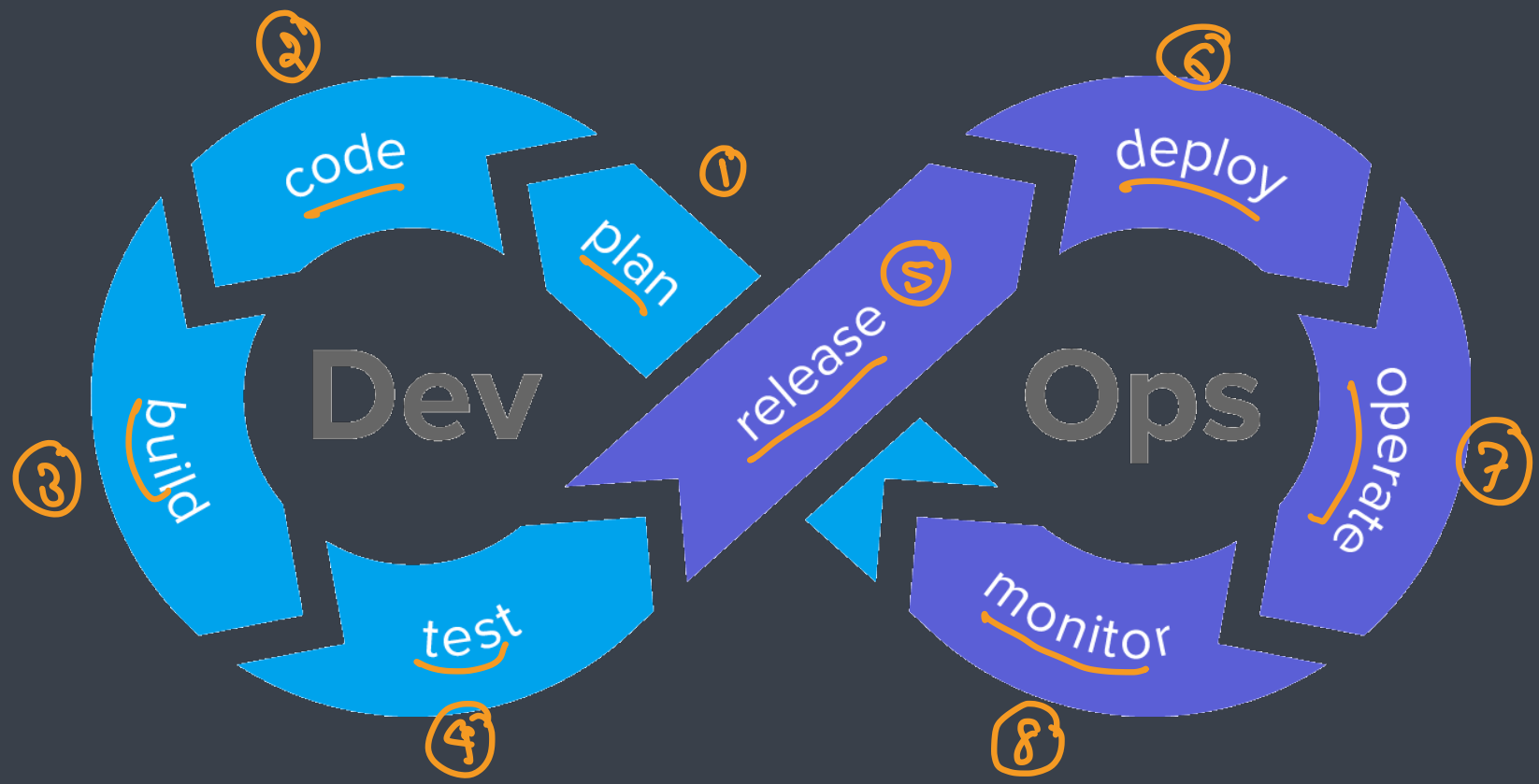
- DevOps is not a role, person or organization
- DevOps is not a separate team
- DevOps is not a product or a tool
- DevOps is not just writing scripts or implementing tools

mindset & continuous improvement

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 a sprint →

- sprint backlog formation
- assign the stories to developers
- track the progress

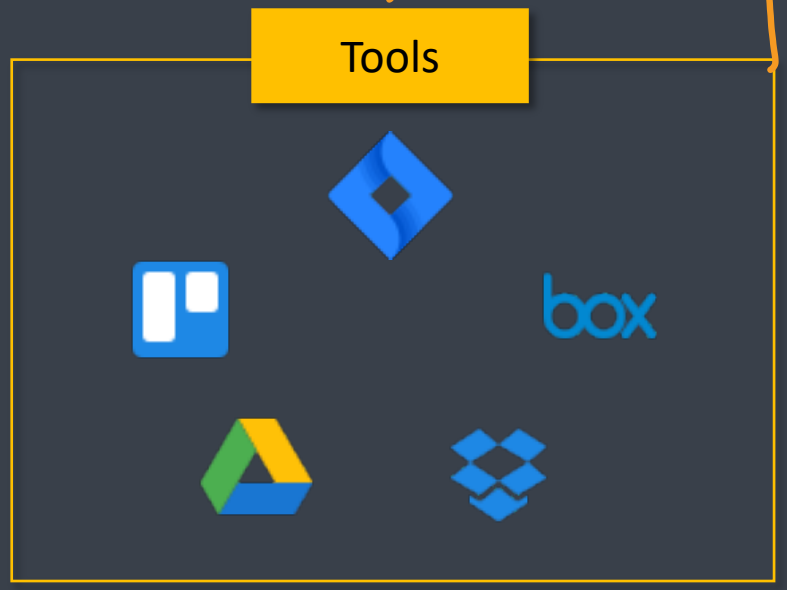
- First stage of DevOps lifecycle where you plan, track, visualize and summarize your project before you start working on it

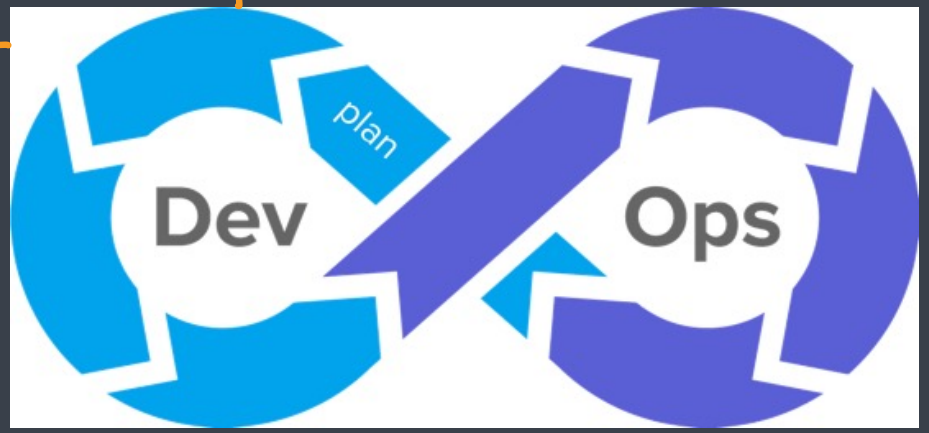
text Editors
- notepad
- textEdit
- MS word

files
- → text files
- → json file
- → Excel file

tools
- MS Excel
- Jira
- OrangeScrum
- Notion

Cloud
- DropBox
- Box
- One Drive
- Google Drive
- icloud





DevOps Lifecycle - Code → development & product

SATA → Serial ATA
ATA → AT Attached
AT → Advanced Technology

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

languages

- JavaScript
- TypeScript
- Java
- C#
- C, C++
- Golang
- Rust

Editors

- vim
- vs code
- atom
- Sublime
- Emacs

IDE

- STS
- Eclipse
- IntelliJ IDEA
- VS
- Android studio
- Webstorm
- PHPStorm

SCM / VCS

- git
- subversion (SUN)
- bazaar
- bitkeeper

scm on cloud

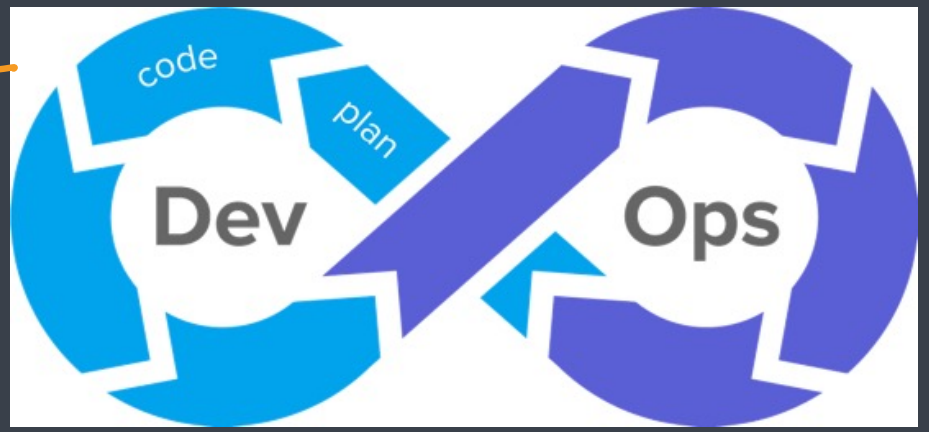
- GitHub
- Gitlab
- Bitbucket

Tools



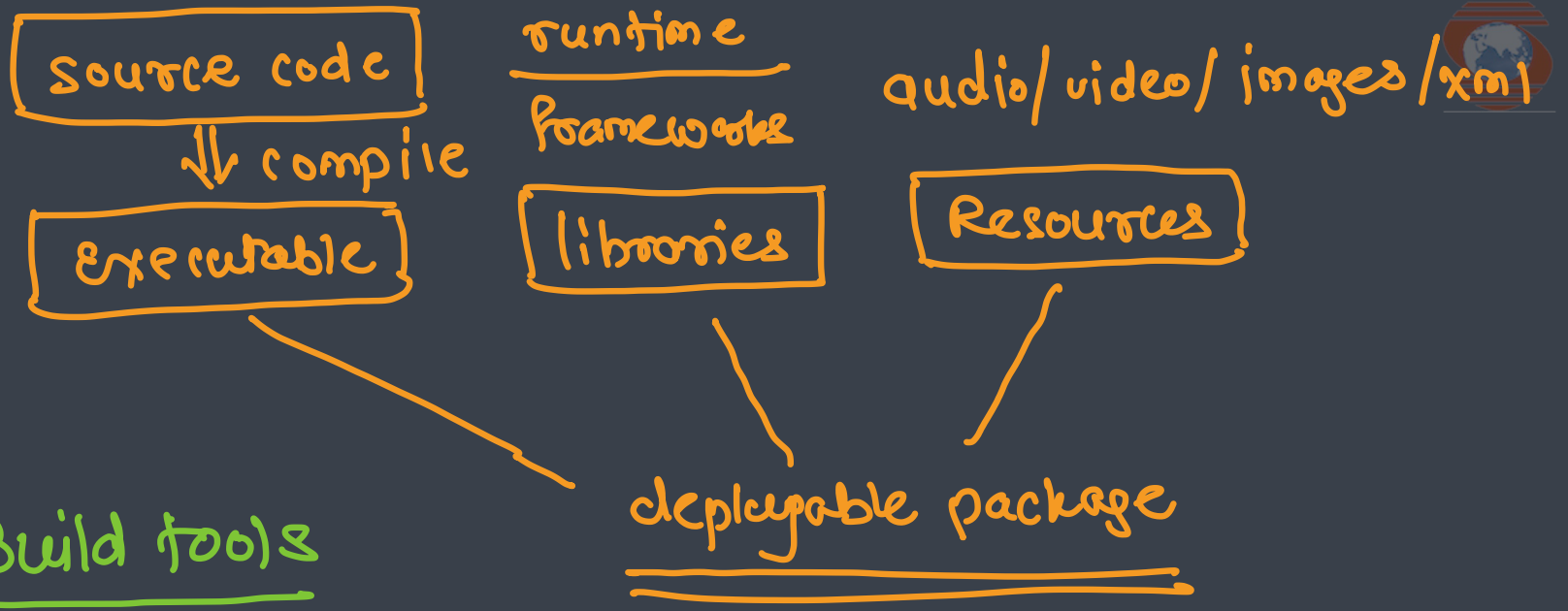

package managers

- node - npm / yarn / pnpm
- python - pip / conda
- C# - nugget / chocolate



DevOps Lifecycle -Build

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

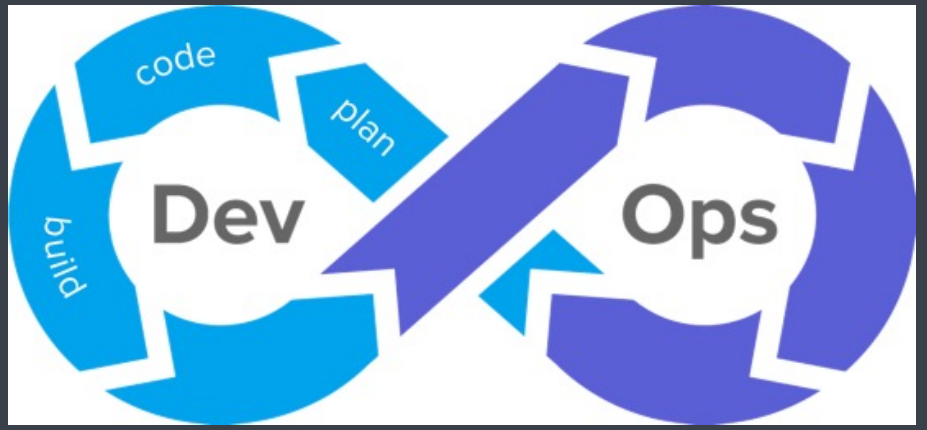


Build tools

- ant ← deprecated
- maven
- gradle

Tools

maven



DevOps Lifecycle - Test → automated testing

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

unit testing

- python - pyunit
- java - Junit

C# - Nunit

JS → Karma / Jasmin

Non-functional testing

- load runner
- stress runner
- Jmeter

mobile testing

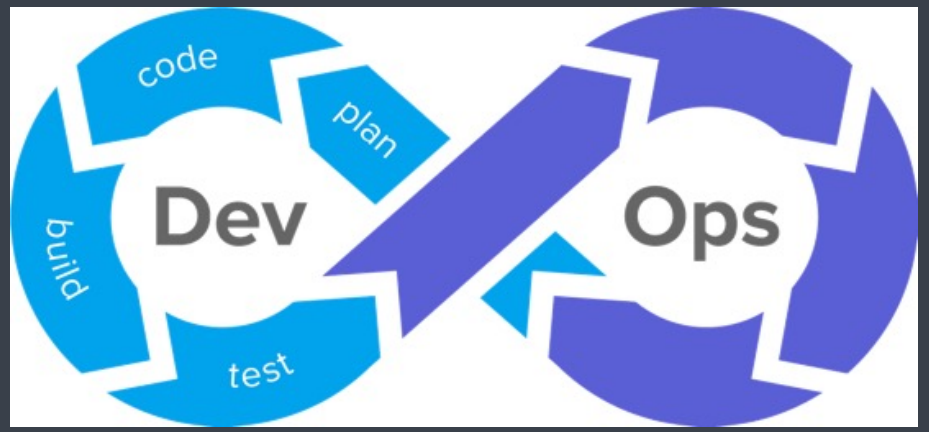
- apium
- browserstack

Tools

JUnit, nunit, Selenium, APACHE JMeter™, TestNG

Website GUI testing

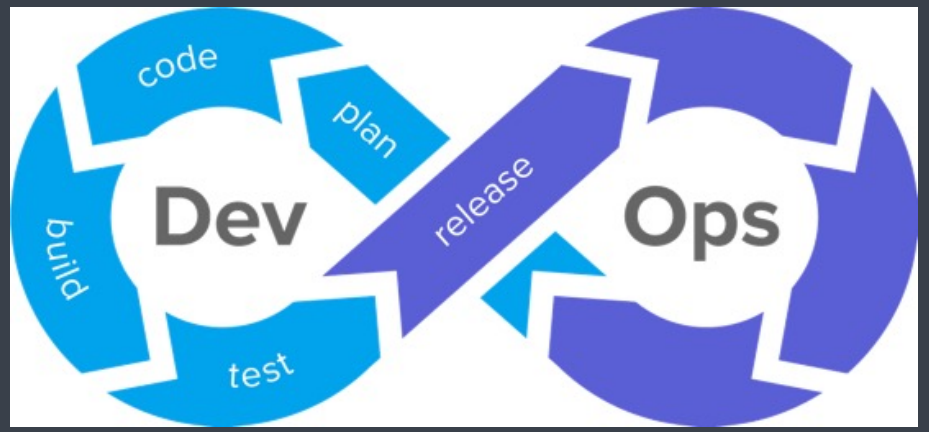
- Selenium
- TestNG
- Cypress



DevOps Lifecycle - Release → CI/CD pipeline

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

CI tools	CI - CD tools
- Circle CI	- GitHub Actions
- GitLab CI	- Jenkins
- Travis CI	- Bamboo



DevOps Lifecycle - Deploy

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

deployment

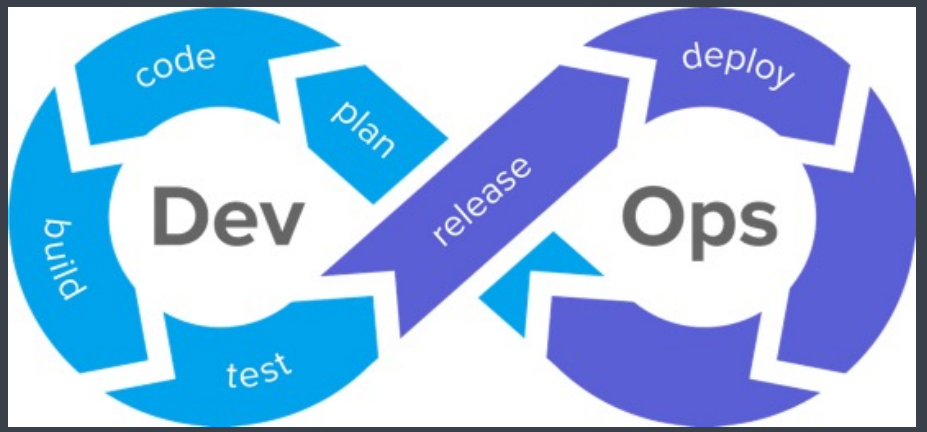
- traditional → using physical machines ✗
- virtualized → using VM → vmware / VirtualBox / Hyper-V / Parallels
- containerized → using docker, podman, LMCTFY, LXC, LXD, containerd

Tools




→ container orchestration

- docker swarm
- K8S
- apache marathon
- apache mesos



DevOps Lifecycle - Operate → Environment creation and configuration

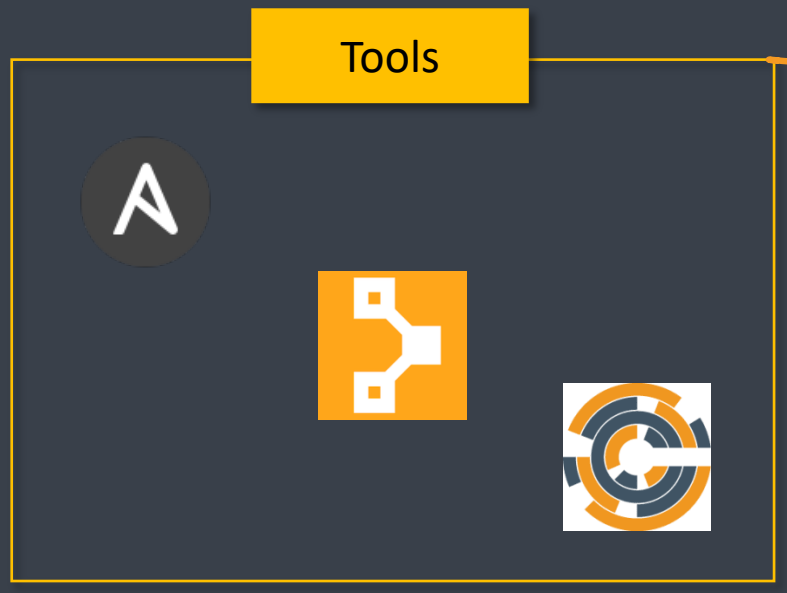
- This stage where the updated system gets operated

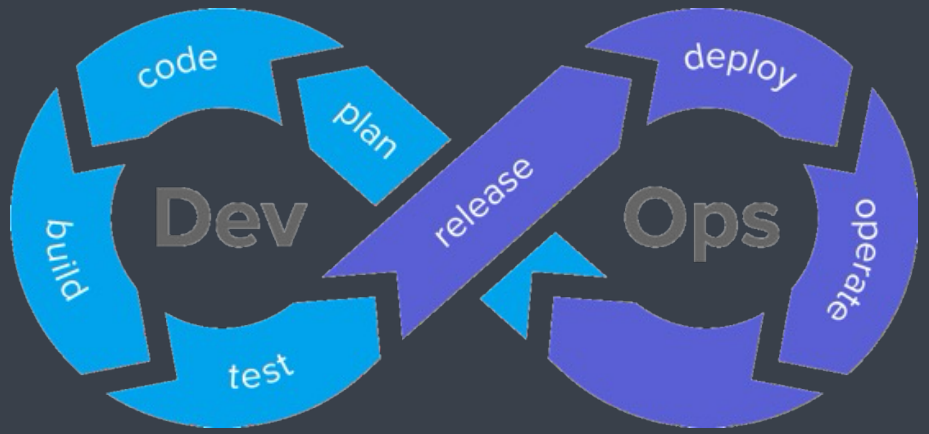
Environment creation tools

- terraform
- vagrant
- AWS Cloud Formation

Environment configuration tools

- Ansible
- Puppet
- Chef
- Saltstack





DevOps Lifecycle - Monitor → monitor application

- 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

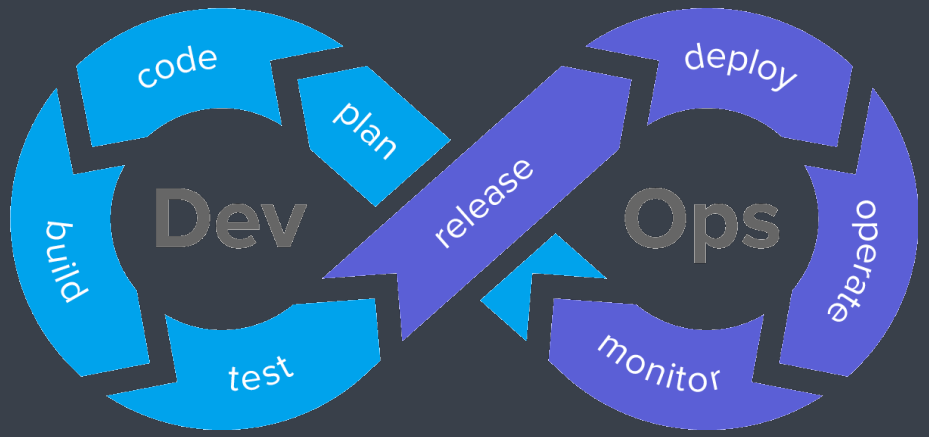
tools
- data dog
- nagios
- splunk

Tools

splunk>

DATADOG

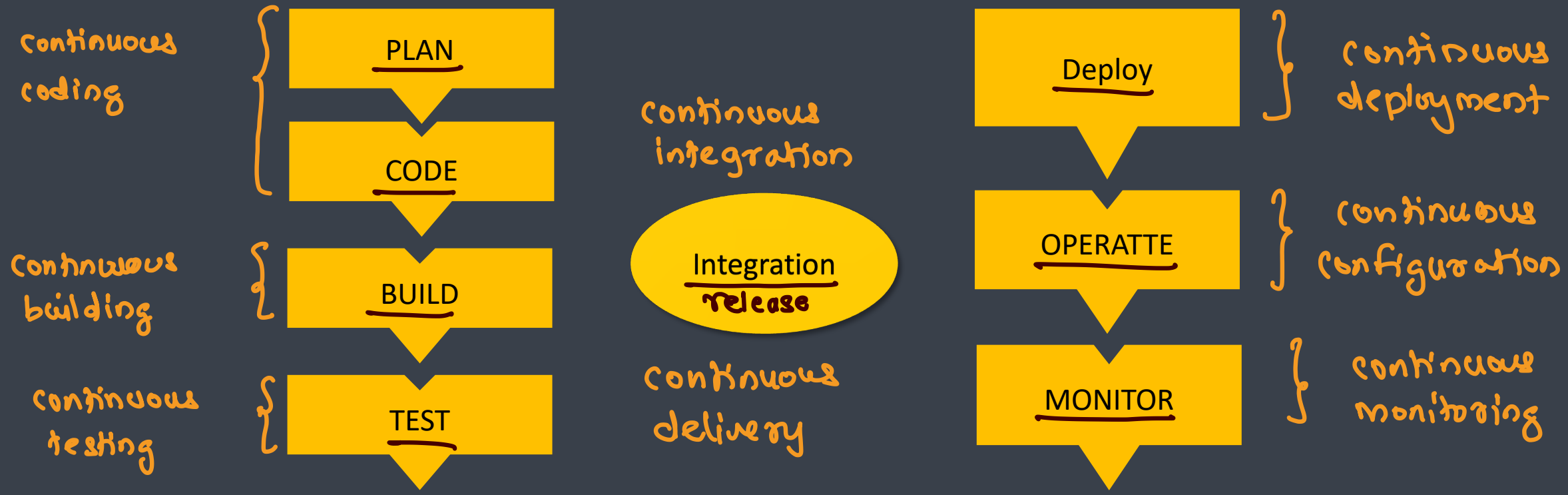
Nagios®



- plan → Jira
- code → vs code
- build → gradle
- test → selenium
- release → Jenkins
- deploy → docker + k8s
- operate → ansible
- monitor → nagios

DevOps Terminologies

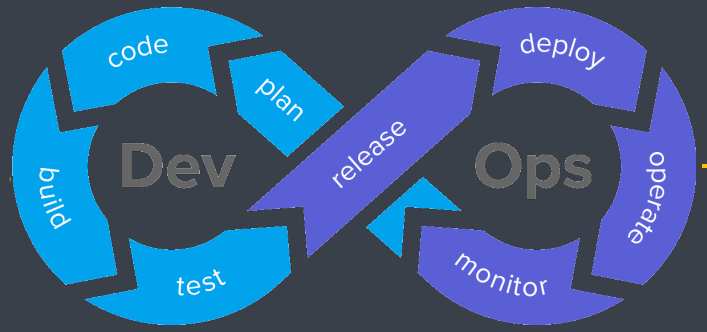
Continuous learning



Responsibilities of DevOps Engineer

Linux

- 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