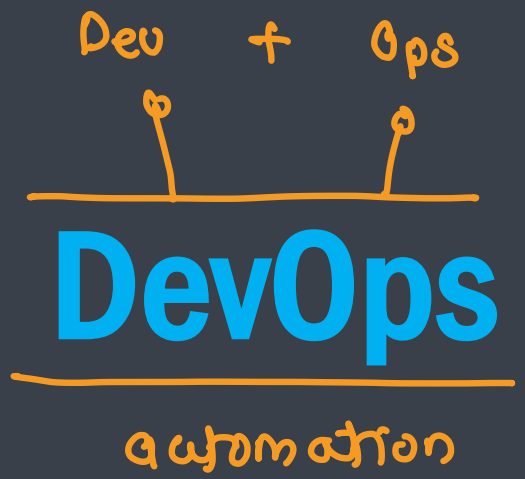
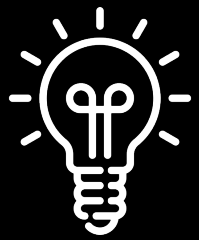

Dev + Ops
DevOps
automation

Software Development Lifecycle

Requirement gathering + Requirement Analysis



PLANNING

Talk to customer and understand the requirements

Requirements

DEFINING

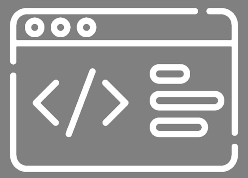
Define the requirements and stick to them

SRS

DESIGNING

Design the solution with right approach

DDS



BUILDING

Development following guidelines

product

TESTING

Make sure that your code is working

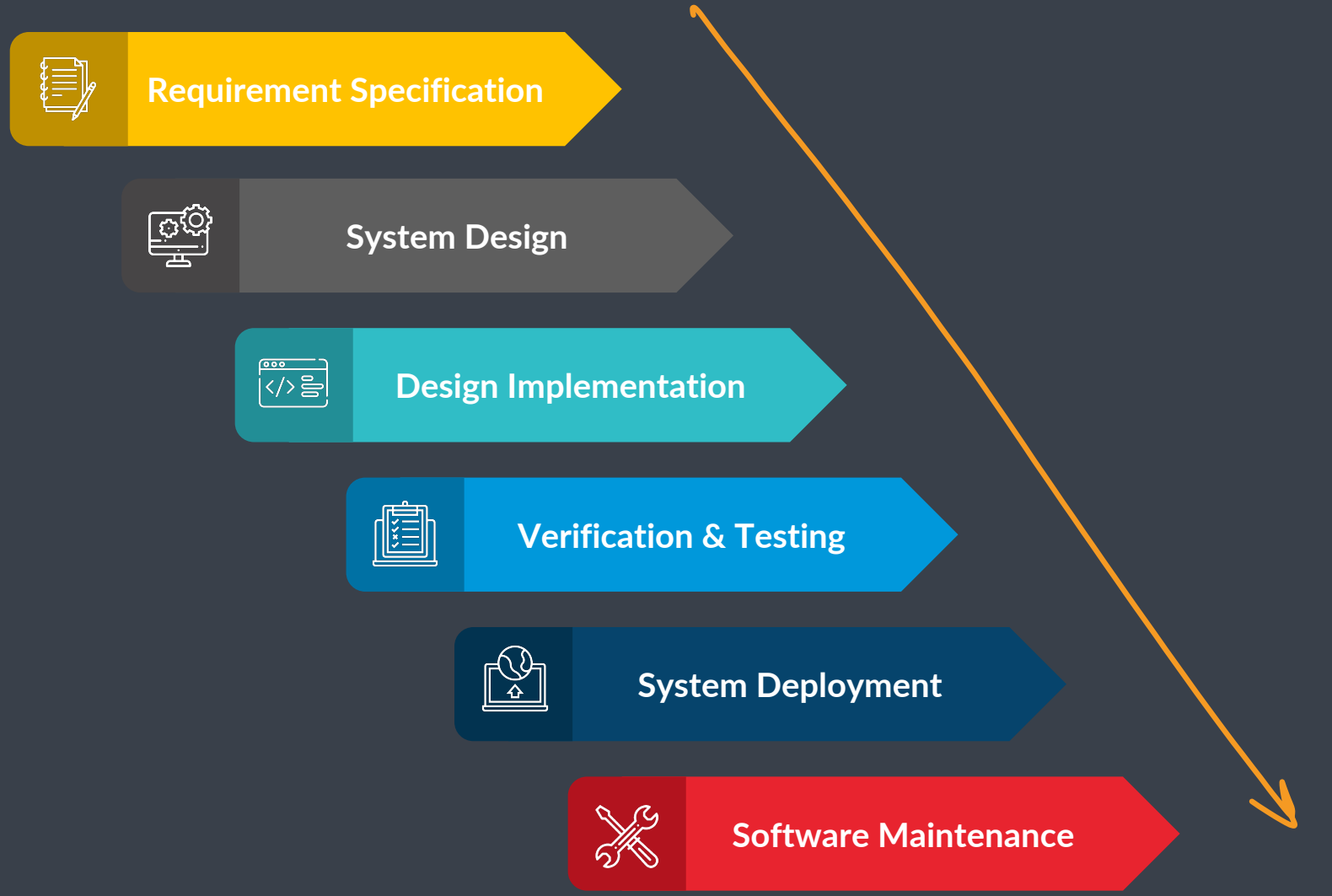
well tested product

DEPLOYMENT

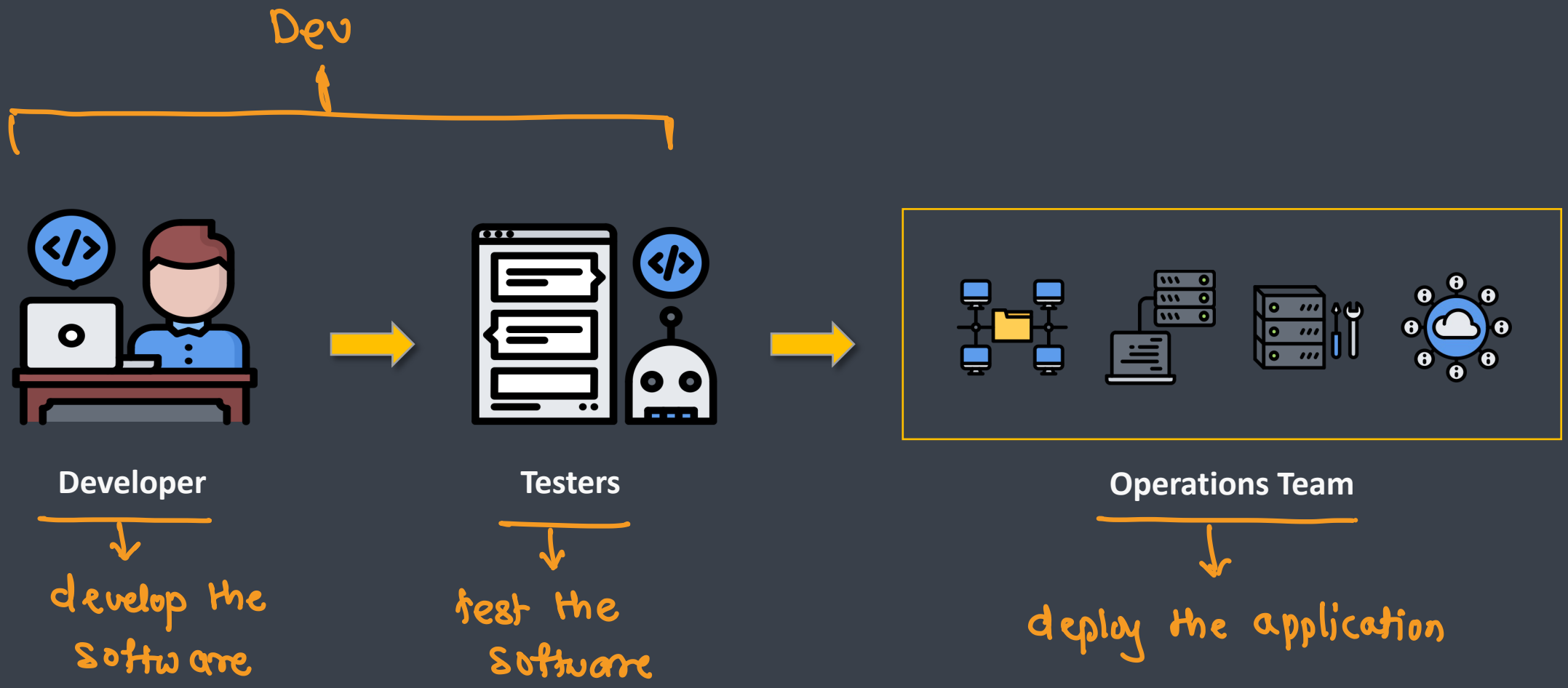
Make your app available for rest of the world

product accessible to end users

Waterfall Model → the whole software at once



Entities involved



Responsibilities

Dev

Developers and Testers

- Developers → add new features
  - Develop the application
  - Package the application → building → package
  - Fix the bugs
  - Maintain the application
- Testers
  - Thoroughly test the application manually or using test automation → Selenium
  - Report the bugs to the developer

building
packaging → process of creating deployable package

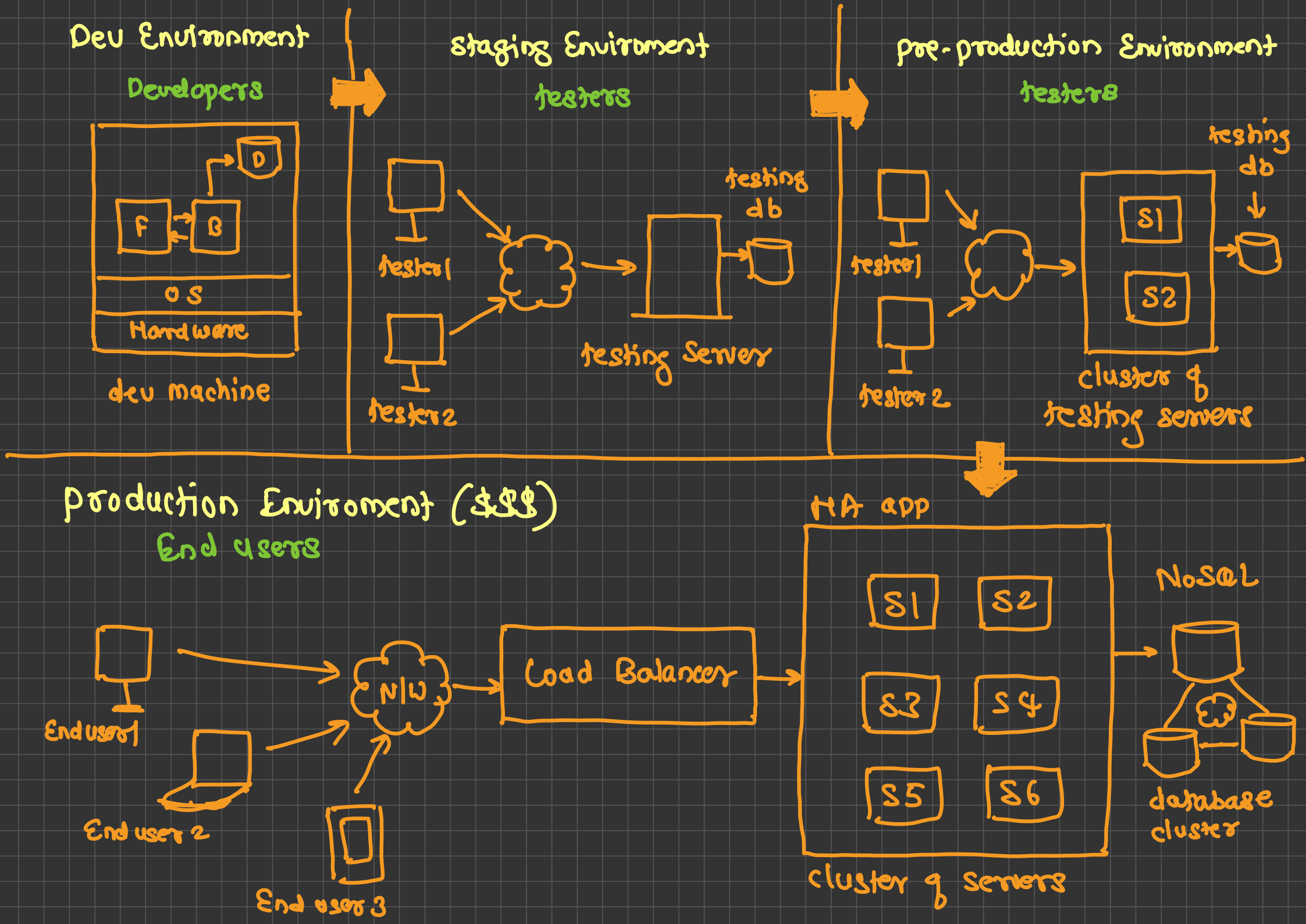
Ops

Operations Team

- Make all the necessary resources ready → machines / infrastructure, licenses of apps, network, devices
- Deploy the application
- Maintain multiple environments
- Continuously monitor the application
- Manage the resources

Android
Java + XML → resources.arsc + .dex → .apk (app package)
building or packaging

ios → .ipa
windows native → .msi
websites → webpack


Dev Environment
Developers
D
F
B
OS
Hardware
dev machine
staging Environment
testers
tester1
tester2
testing db
testing Server
pre-production Environment
testers
tester1
tester2
S1
S2
testing db
cluster of testing servers
Production Environment ($$$)
End users
End user1
End user 2
End user 3
N/W
Load Balancer
HA app
S1
S2
S3
S4
S5
S6
cluster of servers
NoSQL
database cluster

Challenges

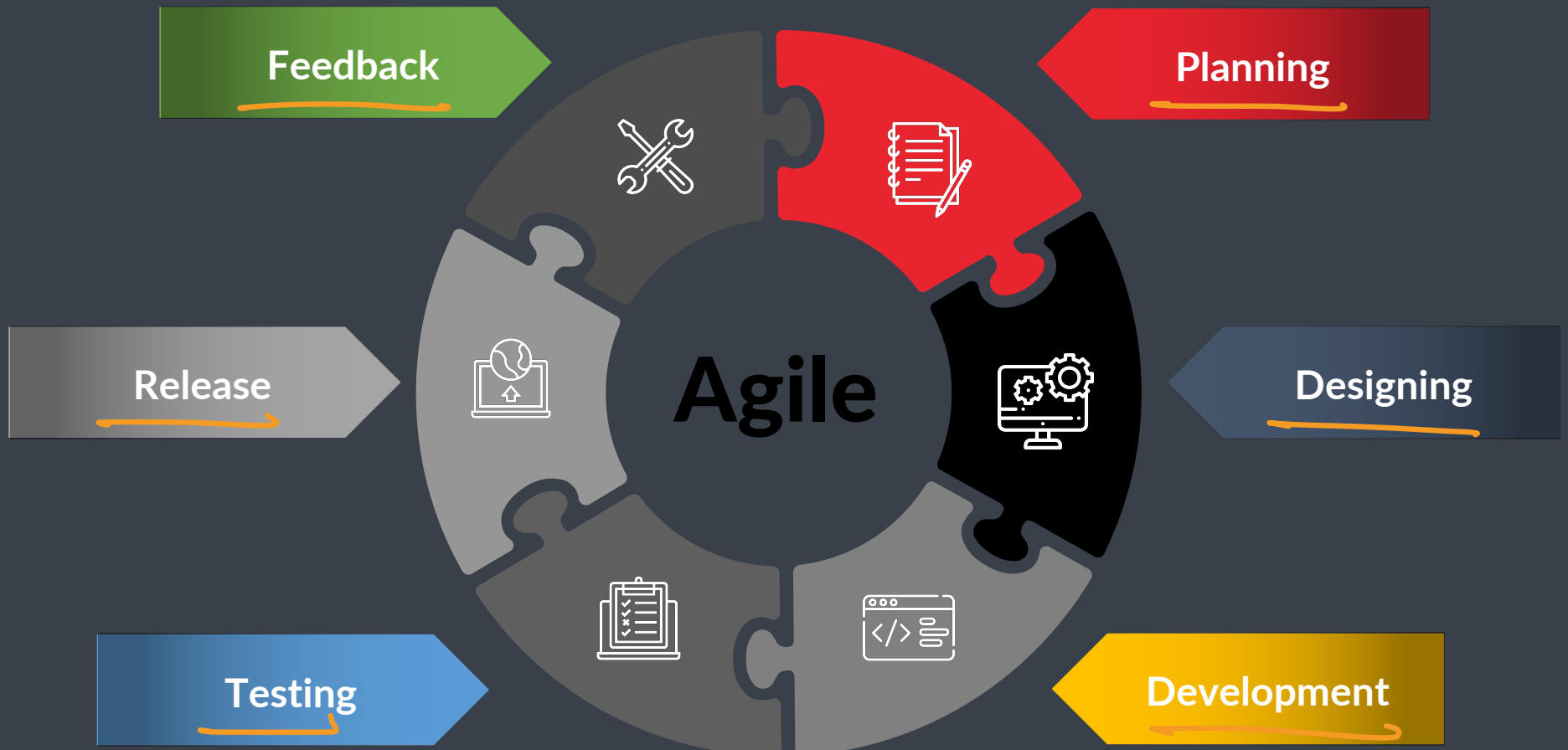
Developers and Testers

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

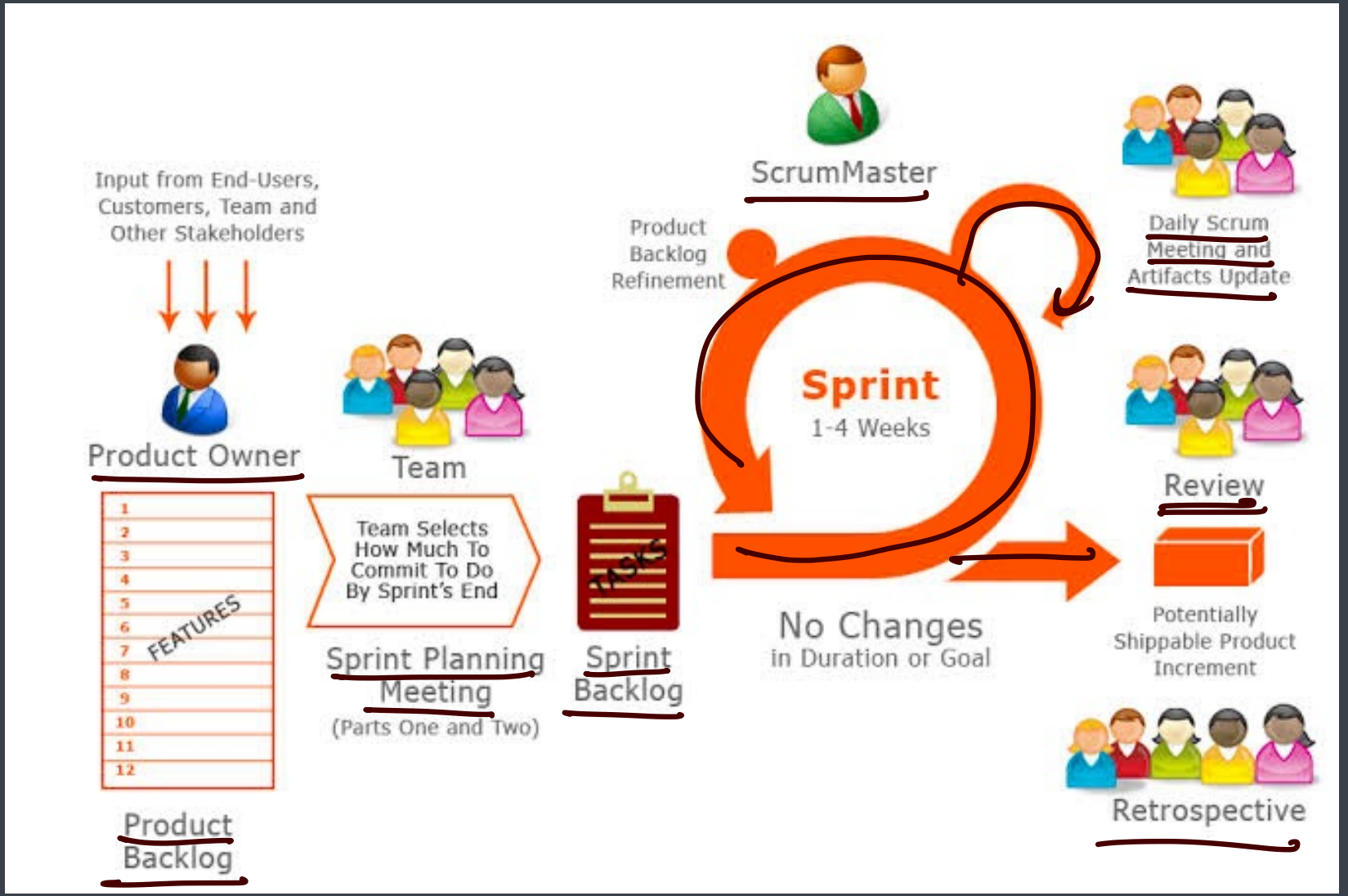
Operations Team

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

Agile Development



Scrum Process

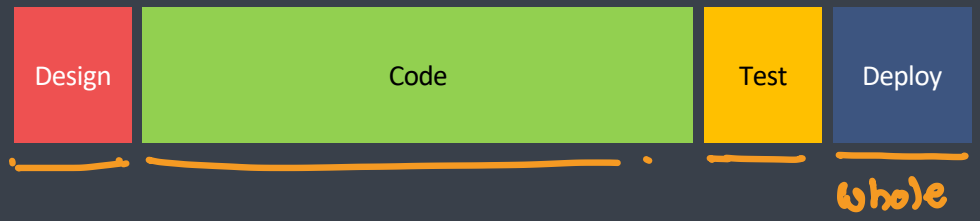


Waterfall Vs Agile

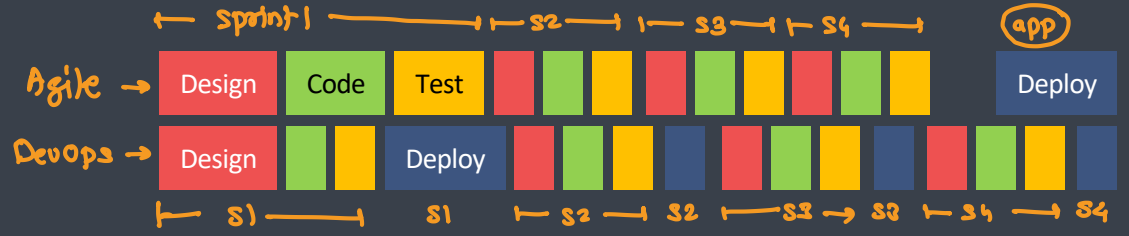
The Waterfall Process

The Agile Process

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



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

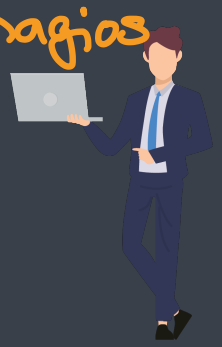
- Managing and tracking changes in the code is difficult: **SCM tools** → git
- Incremental builds are difficult to manage, test and deploy: **Jenkins** → CI/CD pipeline
- Manual testing and deployment of various components/modules takes a lot of time: **Selenium** → automated testing
- Ensuring consistency,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

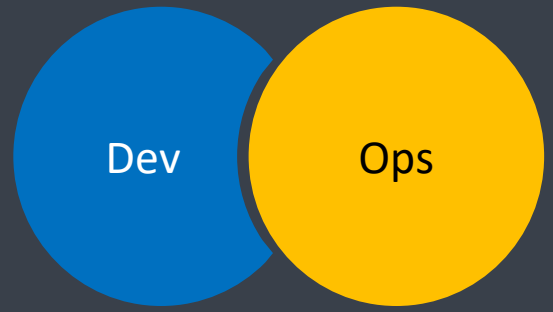
What is DevOps ?

DevOps → better collaboration between Dev + Ops

dev testing

- 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 → automation
- 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 → sprint by sprint dev
- It integrates Development and Operations teams
- It improves collaboration and productivity by
  - Automating infrastructure → creating & configuring env automatically (terraform, puppet)
  - Automating workflow → building + testing + deployment will be automated → CI/CD pipeline
  - Continuously measuring application performance → monitoring → nagios





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

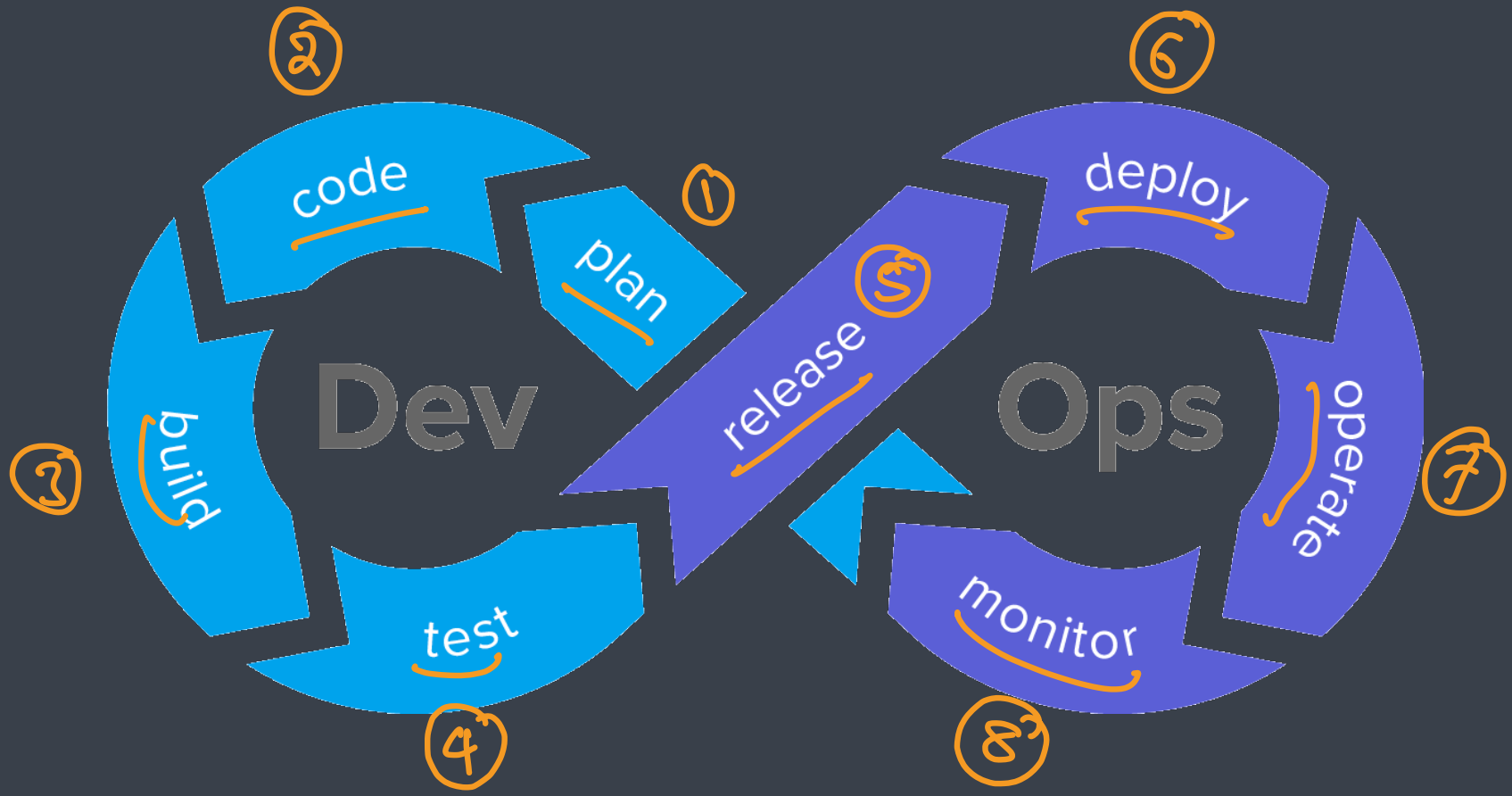
- DevOps is not a role, person or organization *
- DevOps is not a separate team *
- DevOps is not a product or a tool → collection of tools
- DevOps is not just writing scripts or implementing tools → mindset → continuous never ending process

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

with respect to a sprint



DevOps Lifecycle - Plan → planning & sprint

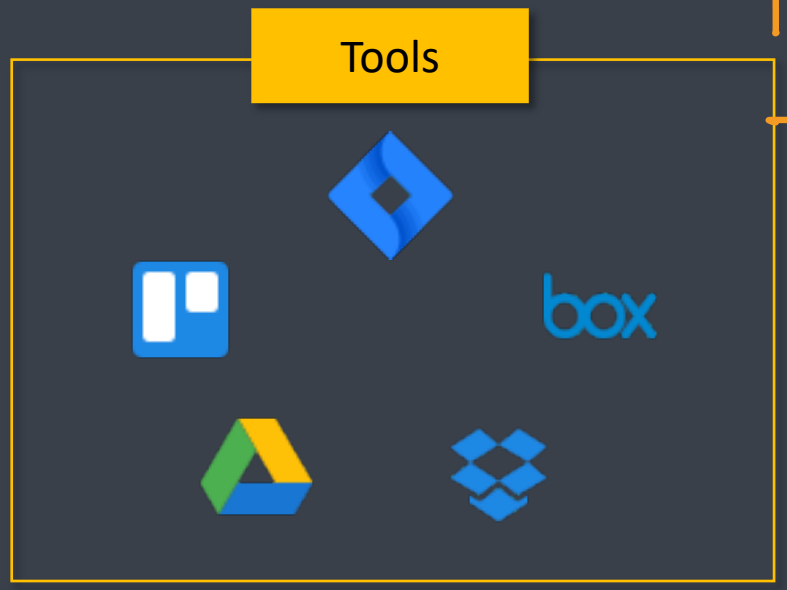
- First stage of DevOps lifecycle where you plan, track, visualize and summarize your project before you start working on it

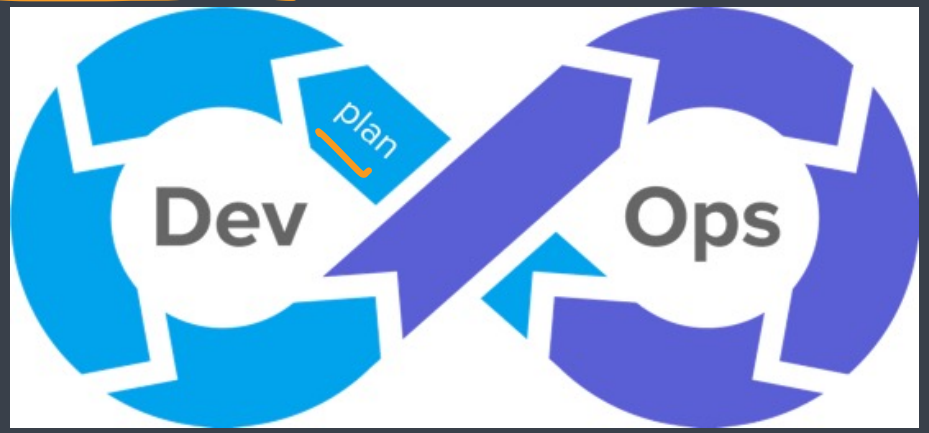
Files
- → text files
- → spreadsheet
- → document files

standard app
- Excel
- word
- Editors

cloud tools
- drop box
- box
- google drive
- one drive
- icloud

Agile tools
- jira ##
- trello
- OrangScrum
- gitScrum
- Github scrum/ Kanban





DevOps Lifecycle - Code → developers will implement features

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

languages

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

Editors

- vim
- VS code
- atom
- Cursor
- sublime

IDEs

- visual studio
- netbeans
- Eclipse
- IntelliJ IDEA
- Android studio
- WS
- PHPStorm
- WebStorm

package managers

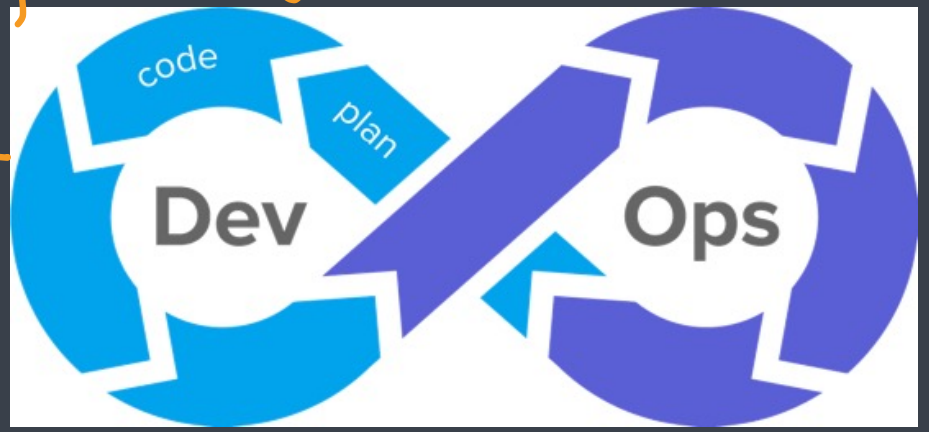
- Node - npm / yarn / pnpm
- .Net - nuget
- → Windows - chocolety
- → Linux - apt / rpm
- → macOS - brew
- → ruby → gem
- -python - pip

Tools




SCM tools

- git
- CVS
- SVN
- Bitkeeper



DevOps Lifecycle -Build

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

build tools
- ant → deprecated
- maven
- gradle +++

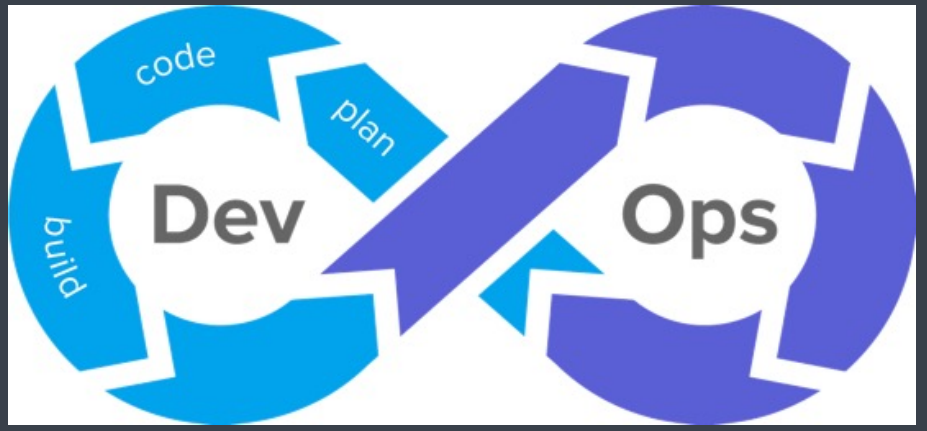
building
- compiling
- integrating dependencies

⇓

deployable package

Tools

maven



DevOps Lifecycle - Test

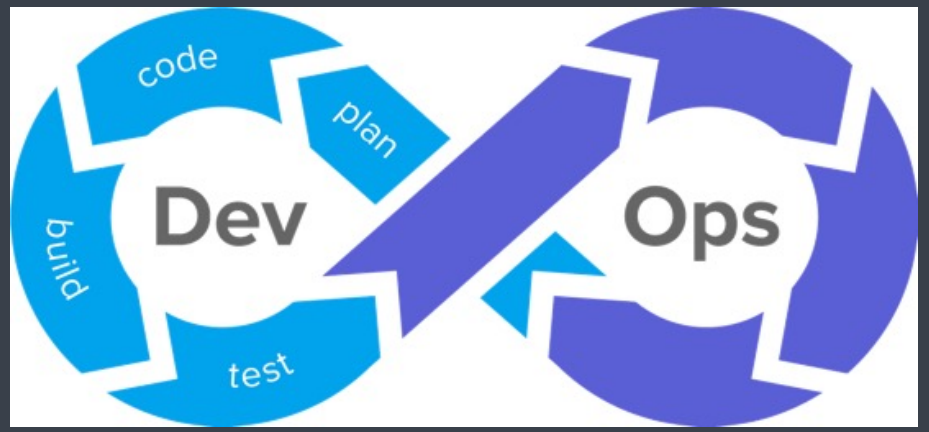
- Process of executing automated tests
- The goal here is to get the feedback about the changes as quickly as possible

Unit testing → developer
- javascript → jasmine, karma, jest
- python - PyUnit
- C# - NUnit
- Java - JUnit

UI testing
- website → selenium, TestNG, Cypress
- mobile app → mockito, apium

Non-functional
- load testing → JMeter, load runner
  - ↳ scaling
- stress testing - stress runner
  - ↳ testing app under abnormal condition

Tools



DevOps Lifecycle - Release → CI/CD pipeline → sequence of stages

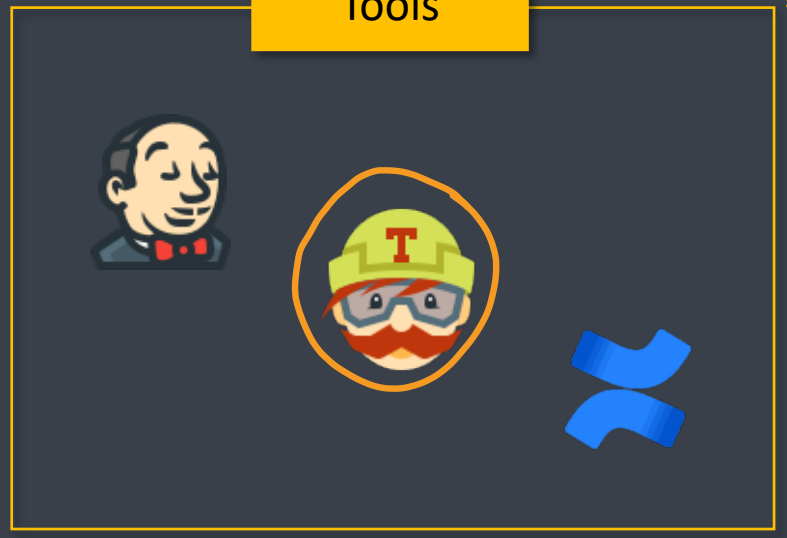
- This phase helps to integrate code into a shared repository using which you can detect and locate errors quickly and easily

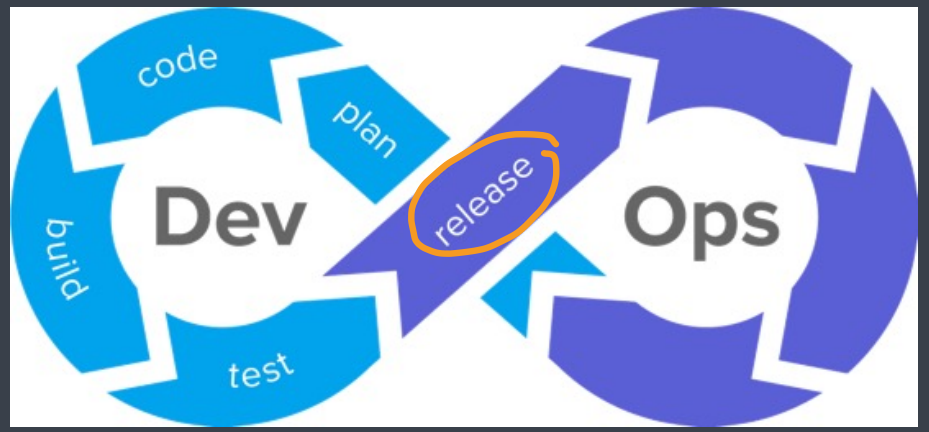
CI tools
- Circle CI
- GitLab CI
- Travis CI
- GitHub Actions

CD tools
- Argo CD

CI/CD tools
- Jenkins
- Bamboo

Tools





DevOps Lifecycle - Deploy

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

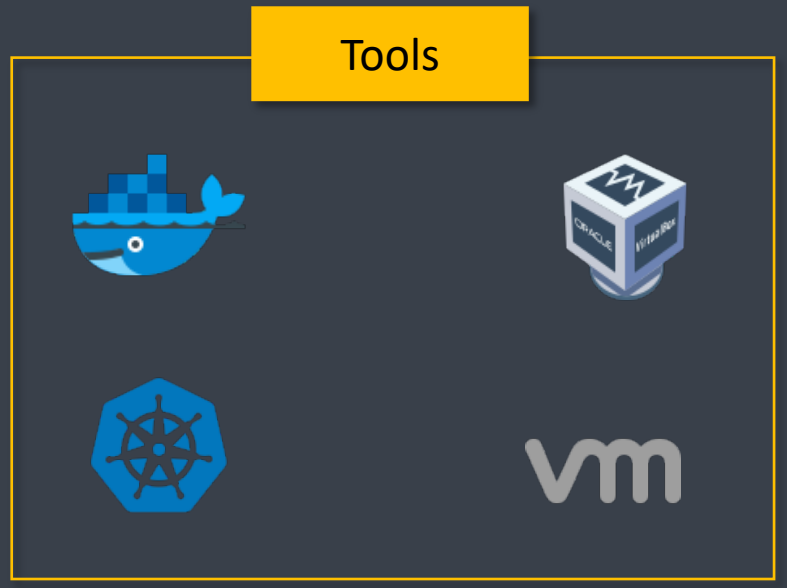
Deployment → traditional deployment → uses physical infra → deprecated

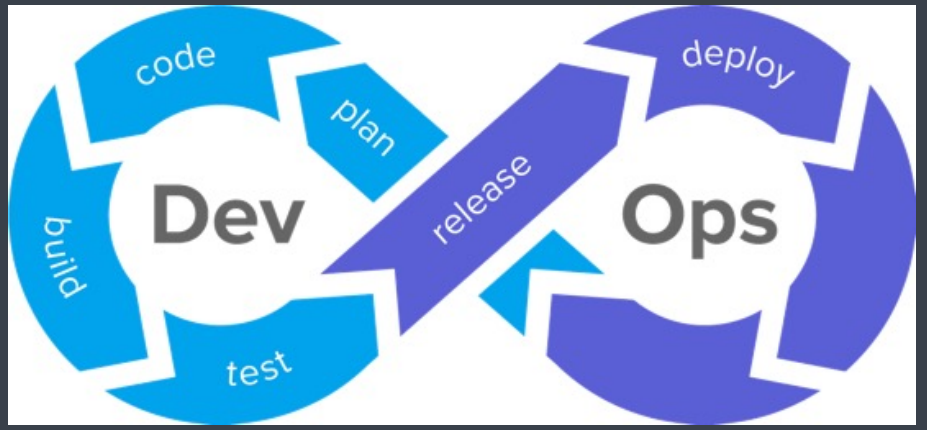
virtualized deployment → uses VM → VMware, VirtualBox, parallels

Containerized deployment → uses containers → docker, podman,

Container orchestration → docker swarm, kubernetes, mesos, marathon

Tools



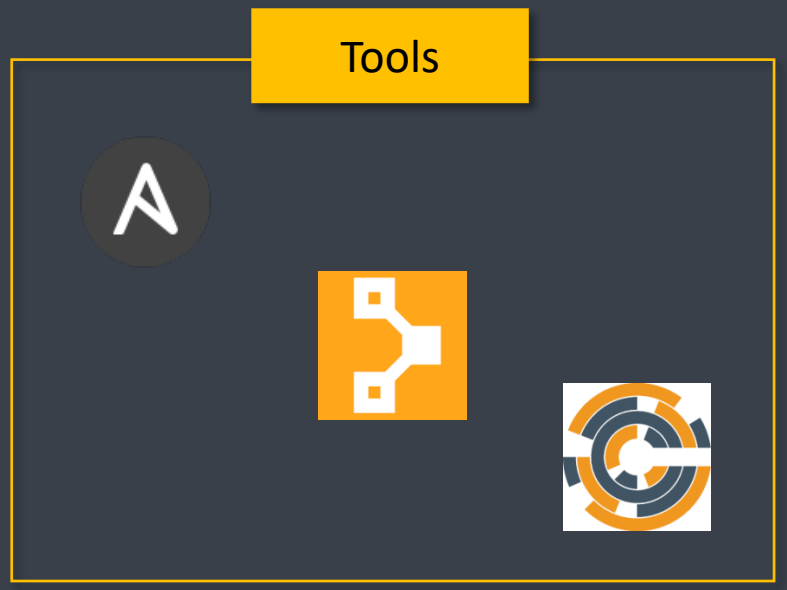


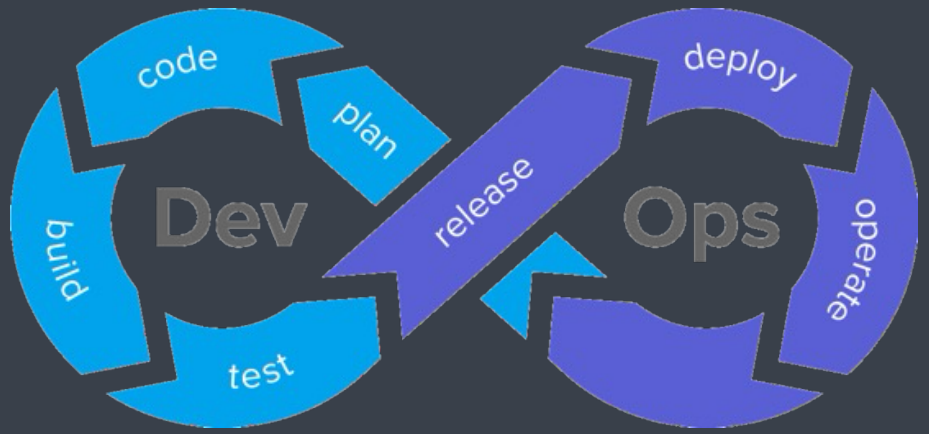
DevOps Lifecycle - Operate → create and configure Environment

- This stage where the updated system gets operated

create environment → terraform, vagrant, cloud formation (aws)

configure environment → puppet, chef, ansible, saltstack





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 → nagios, datadog, splunk,

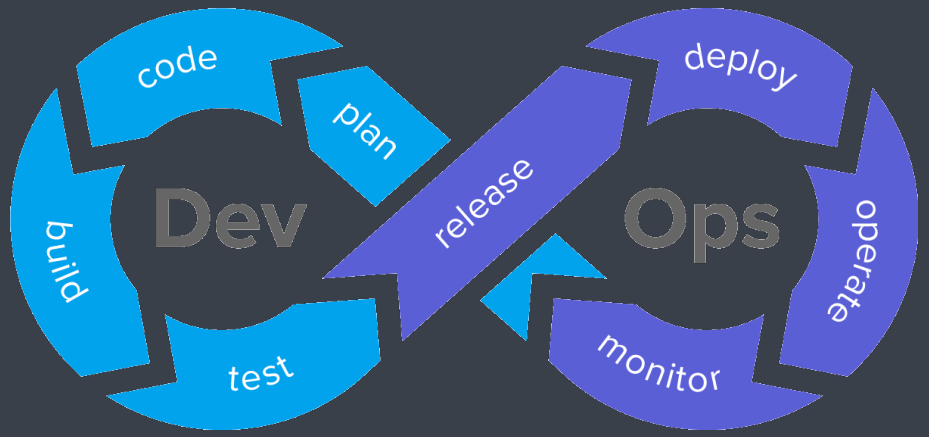
developers libraries → Sentry, New Relic

Tools

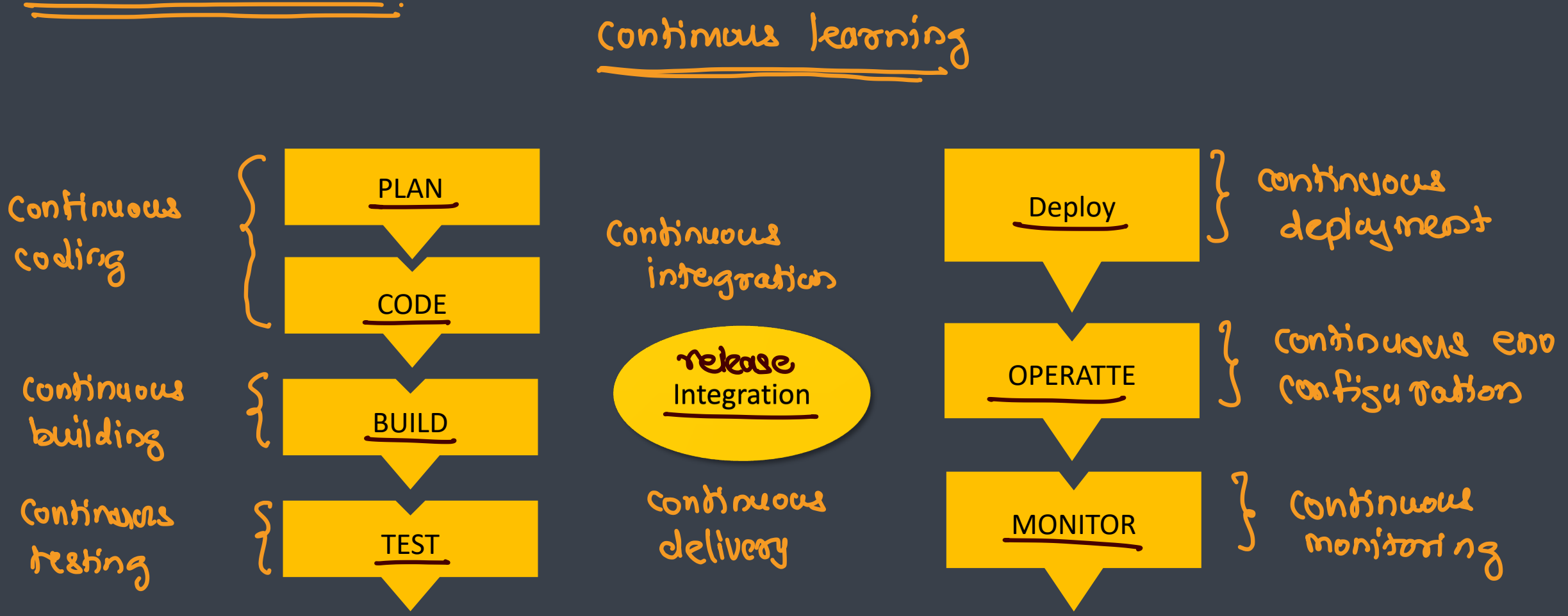
splunk>

DATADOG

Nagios®



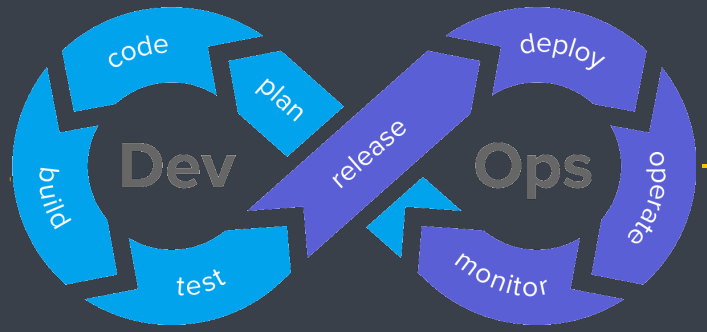
DevOps Terminologies



Responsibilities of DevOps Engineer

Linux → administrator

- 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