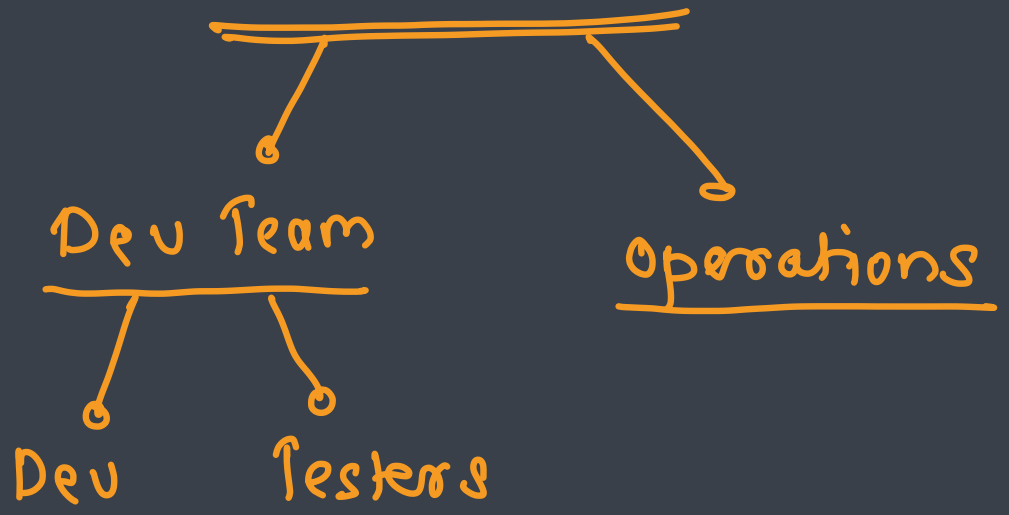
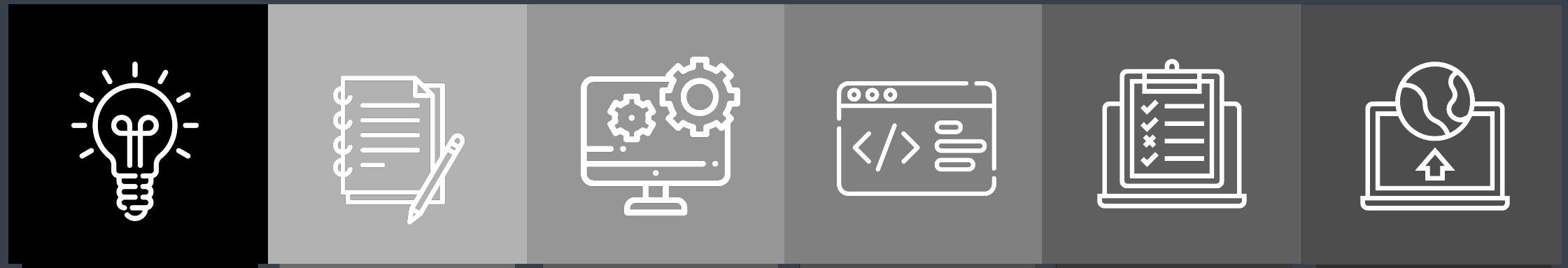
DevOps



Software Development Lifecycle



PLANNING

Talk to customer and understand the requirements

DEFINING

Define the requirements and stick to them

SRS

DESIGNING

Design the solution with right approach

DDS

BUILDING

Development following guidelines

App / product

TESTING

Make sure that your code is working

well tested product

DEPLOYMENT

Make your app available for rest of the world

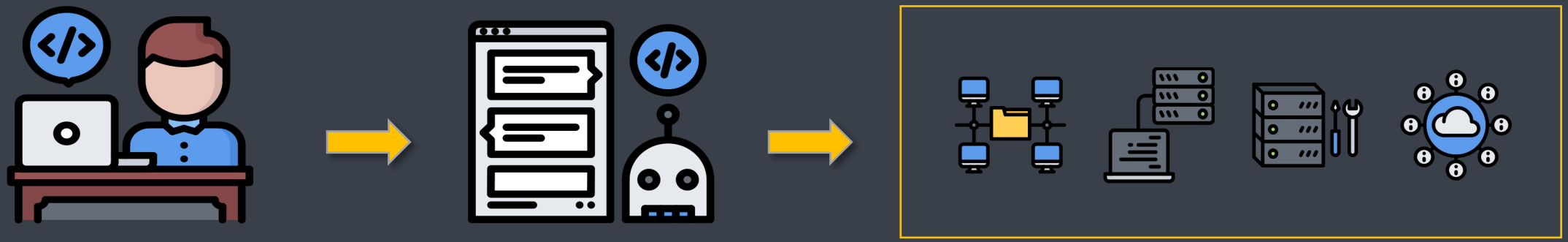
End users access

Waterfall Model

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

Entities involved

Dev Team



Developer

Coding

Testers

testing

Operations Team

Responsibilities

Dev Team

Developers and Testers

SAB

- 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

Operations Team

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

Challenges

Developers and Testers

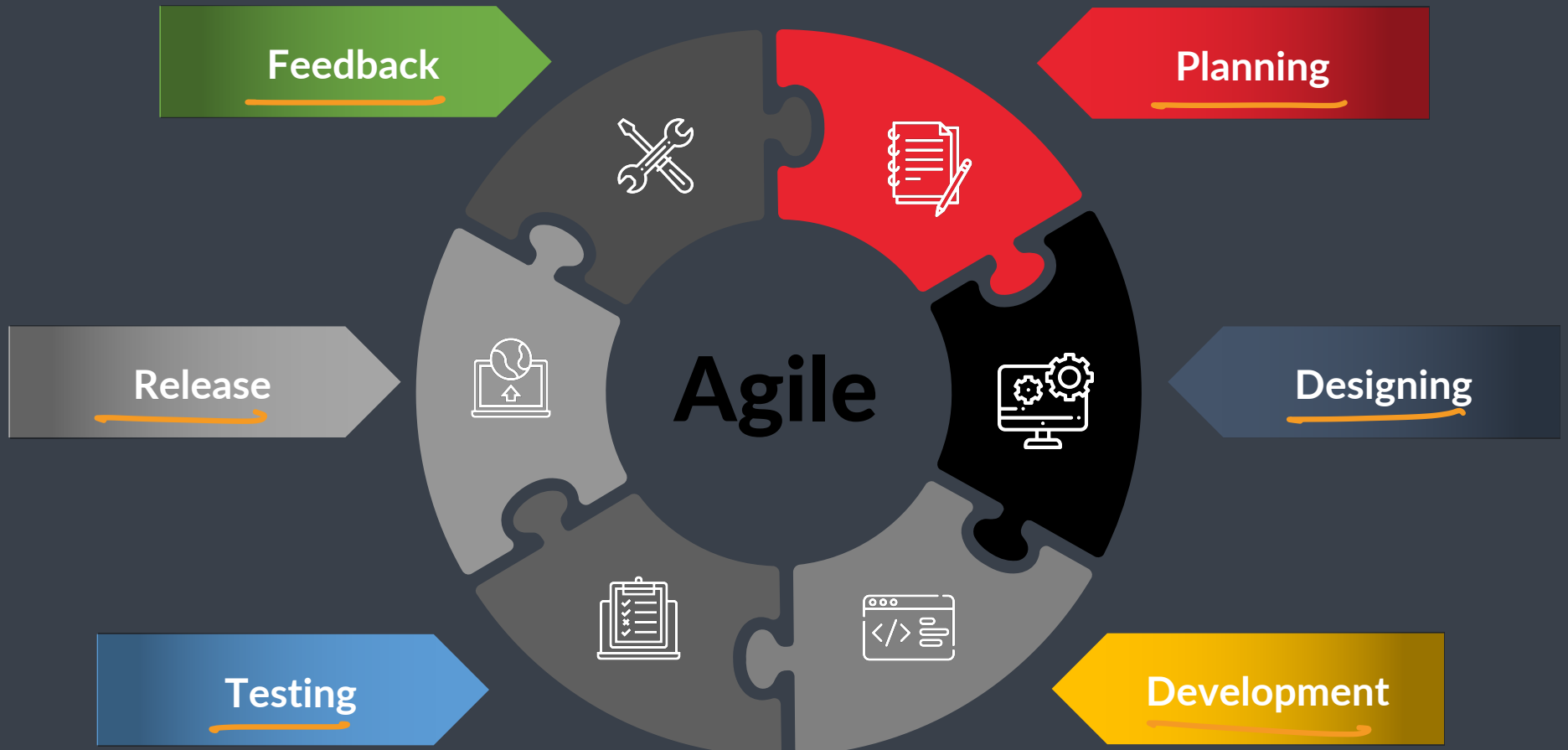
- The process is slow
- The pressure to work on the newer features and fix the older code
- Not flexible

Operations Team

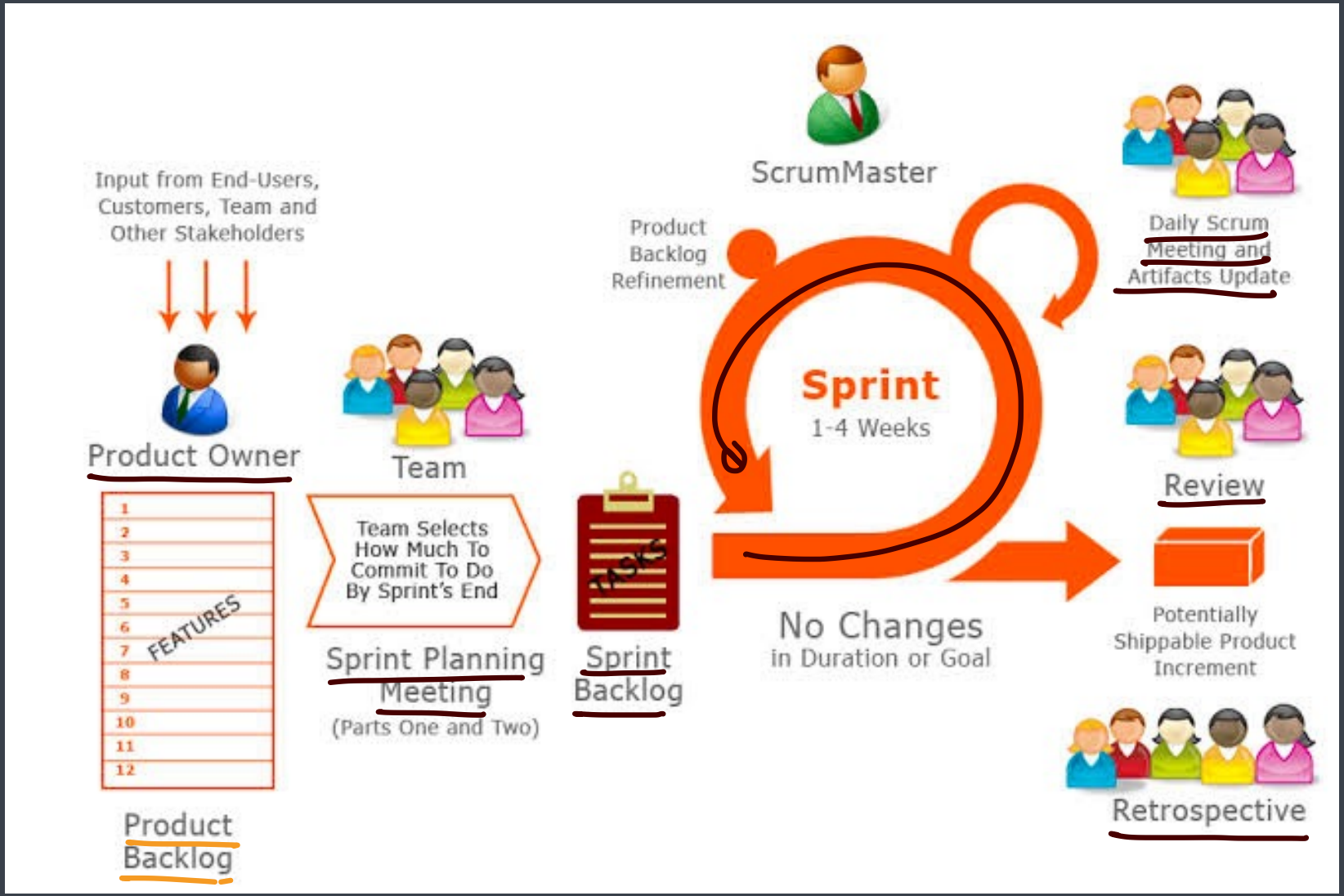
- Uptime - maximize
- Configure the huge infrastructure
- Diagnose and fix the issue

Agile Development

Sprint



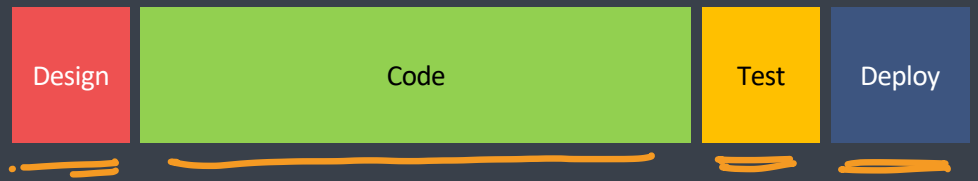
Scrum Process



Waterfall Vs Agile

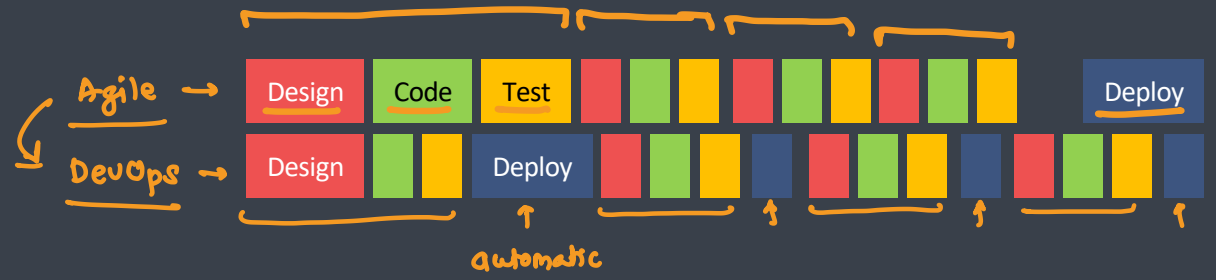
The Waterfall Process

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



Problems

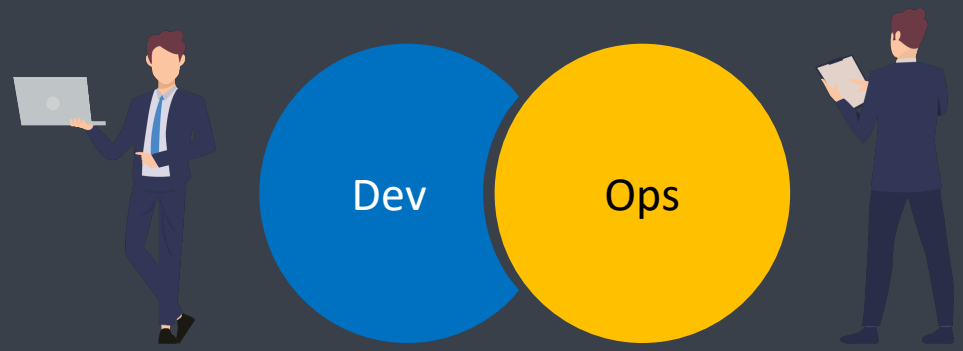
- Managing and tracking changes in the code is difficult
- Incremental builds are difficult to manage, test and deploy
- Manual testing and deployment of various components/modules takes a lot of time
- Ensuring consistency, adaptability and scalability across environments is very difficult task
- Environment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

Solutions to the problem

- Managing and tracking changes in the code is difficult: **SCM tools**
  - CVCS → SVN
  - DVCS → 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** Test Automation
- 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

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



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

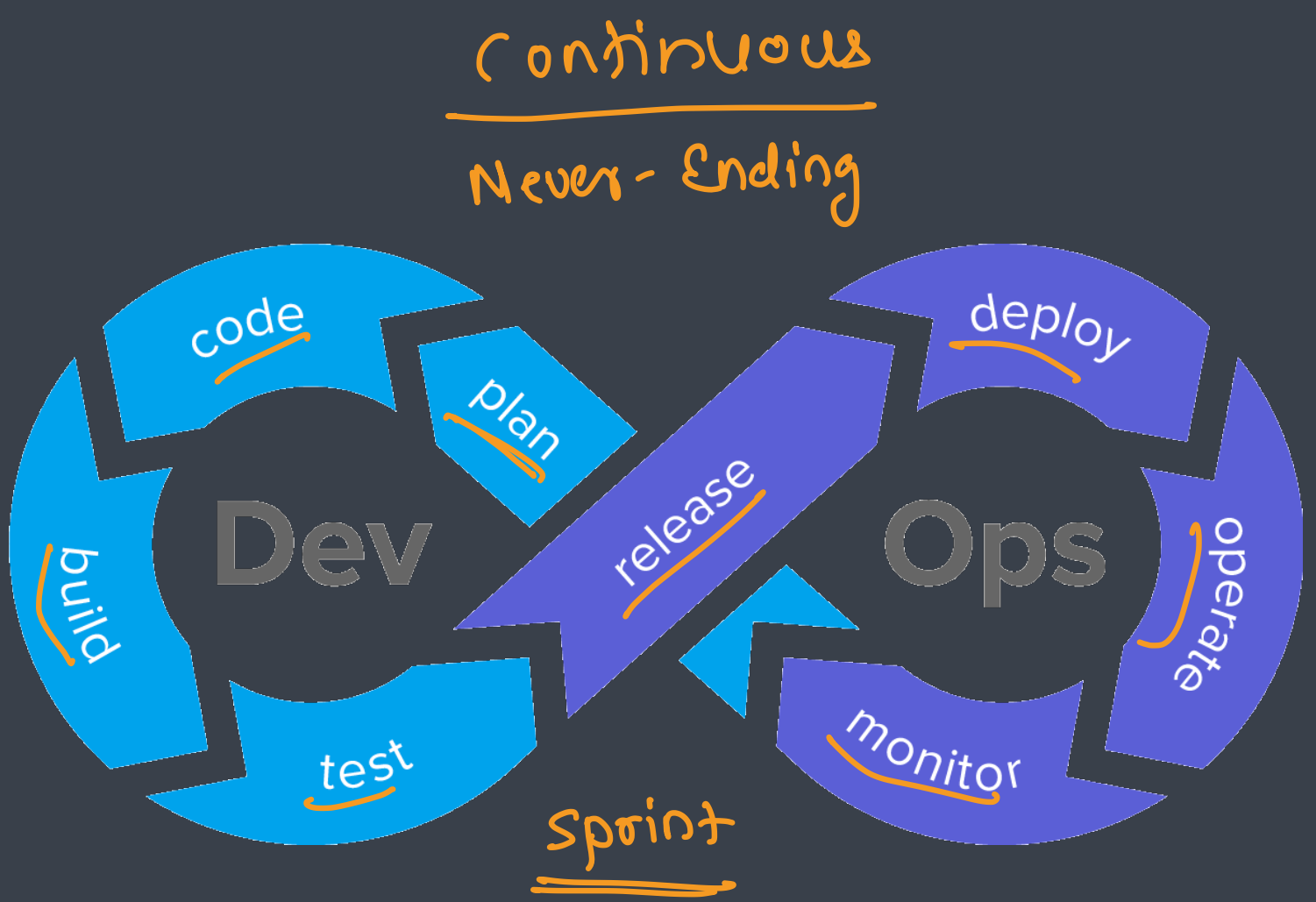
Common misunderstanding

- DevOps is not a role, person or organization → DevOps Engineer
- DevOps is not a separate team
- DevOps is not a product or a tool → collection of many tools
- DevOps is not just writing scripts or implementing tools – mindset / continuous 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



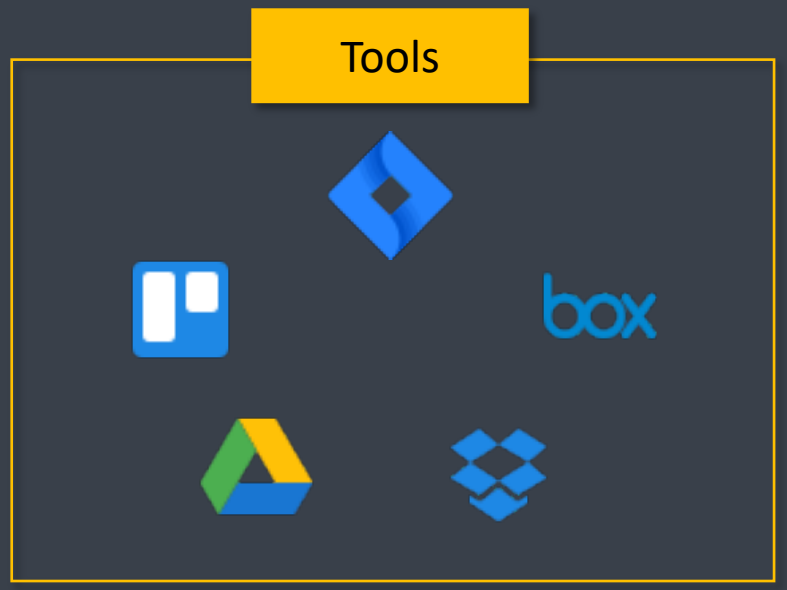
DevOps Lifecycle - Plan → Jira

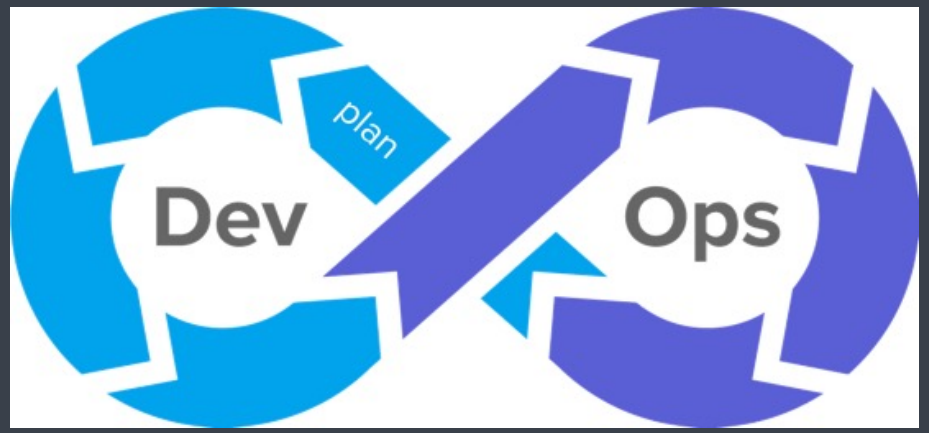
- First stage of DevOps lifecycle where you plan, track, visualize and summarize your project before you start working on it

general tools
- Excel
- text files

Cloud tools
- Cloud Drive - google drive, one drive, drop box, box

planning / Scrum tools
- Jira
- Orange Scrum
- ScrumTools





DevOps Lifecycle - Code

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

Editors
- vim
- VSCode
- Atom
- Sublime

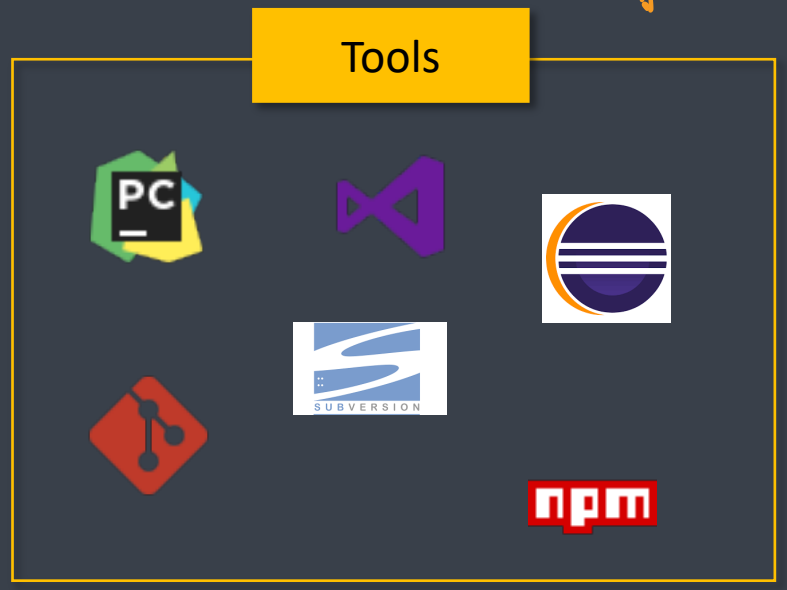
IDEs
- VS
- Android studio
- PyCharm
- IDEA / Eclipse
- WebStorm

SDK
- .Net SDK
- Android SDK
- iOS SDK
- JDK

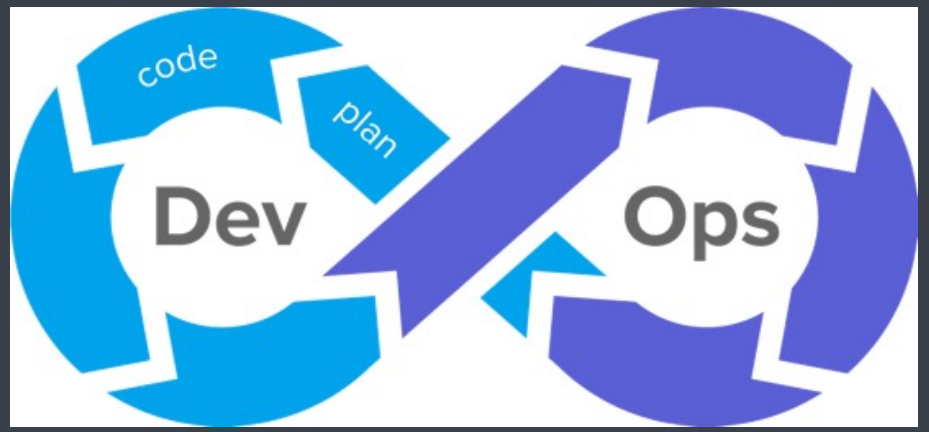
package manager
- Node – npm / pnpm / yarn
- Windows - nugget
- Python - pip (Python Index Packages)
- Ruby - Gems / pod
- iOS - cocoa-pods

languages
- compiled - C, C++, Go, Swift
- interpreted - ruby, Javascript,
- mixed - Java, Python

Tools



SCM tools
- LVCS - sourcesafe
- CVCS - SVN / CVS
- DVCS - git, bazaar, bitkeeper



DevOps Lifecycle -Build = compile the code + add dependencies & create deployable package

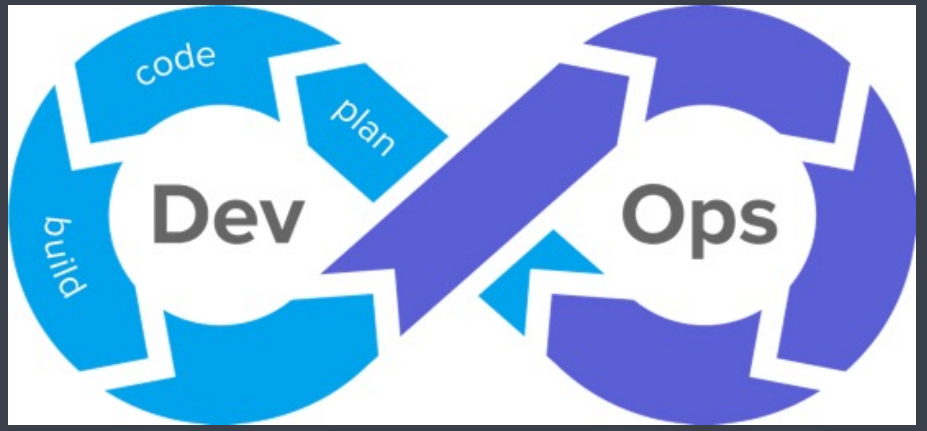
- Integrating the required libraries
- Compiling the source code
- Create deployable packages

tools
- ant [deprecated]
- maven
- gradle [groovy]

deployable package
- web - bundle
- android - .apk, .aab
- ios - .ipa
- windows - .exe, .msi
- linux - .rpm, .deb
- macos - .dmg

Tools

maven



DevOps Lifecycle - Test

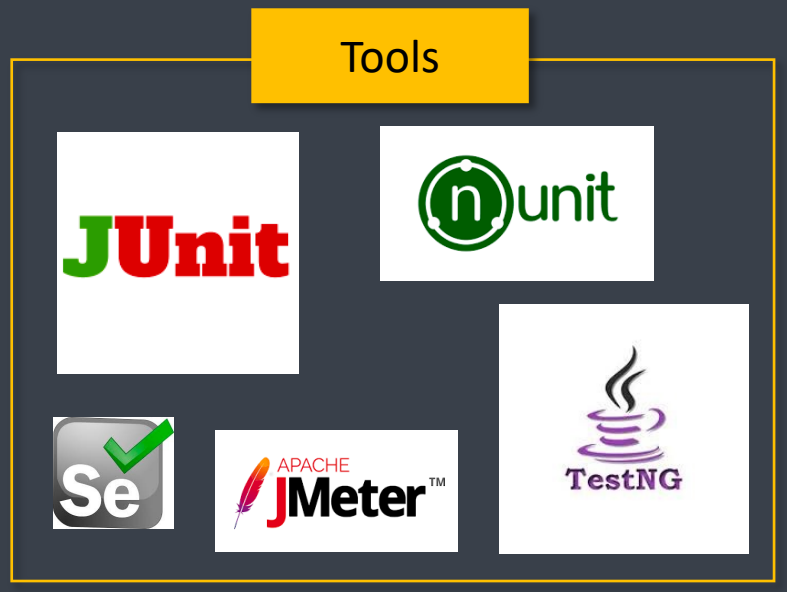
- Process of executing automated tests
- The goal here is to get the feedback about the changes as quickly as possible

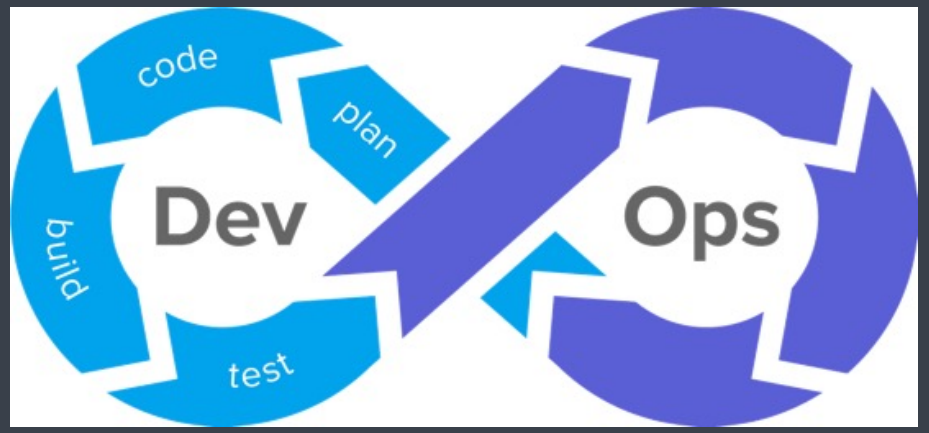
Unit testing
- JavaScript - Jest, Jasmin
- python - PyUnit
- C# - NUnit
- Java - JUnit

Performance testing
- LoadRunner
- JMeter
  - scalability testing

UI testing
- selenium (python, C++, C#, Java, JS)
- cypress
- TestNG

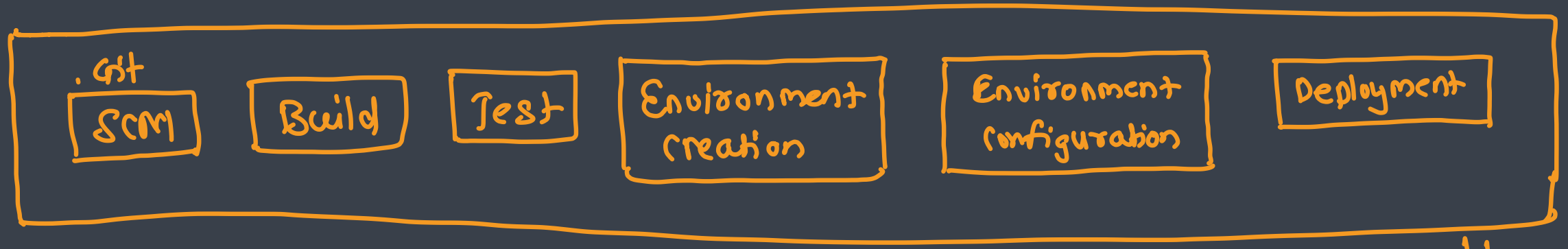
Tools



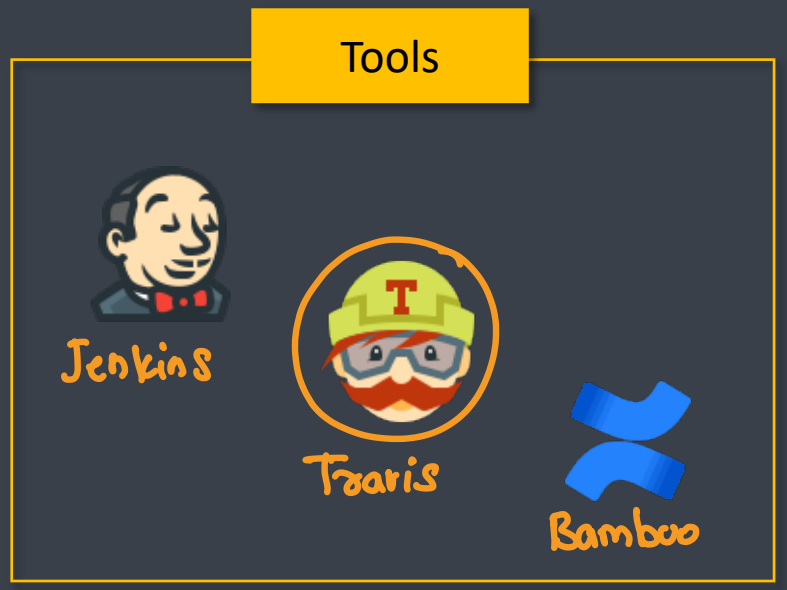


DevOps Lifecycle - Release : CI/CD pipeline tools

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



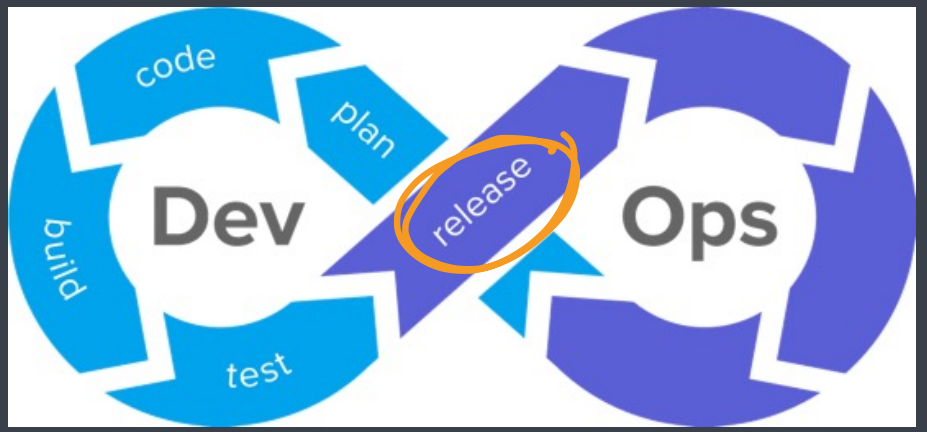
CI/CD pipeline



CI tools
- Travis CI, Circle CI, GitLab CI

CD tools
- ArgoCD

CI/CD tools
- Jenkins, Bamboo



DevOps Lifecycle - Deploy

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

traditional deployment ✕

- using physical machines

virtualized deployment [VM]

- Virtual Box, VMware, parallels, Bosch, Qemu

Tools

cloud

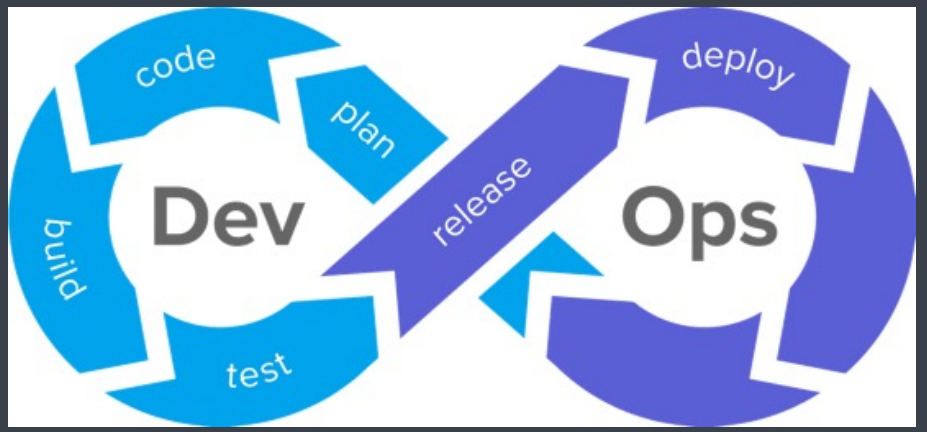
- AWS, Azure, GCP
- Rackspace, Digital Ocean, IBM cloud, Alibaba Cloud

containerized deployment

- docker, podman, LMCTFY, containerd

container orchestration

- docker swarm, kubernetes, Mesos, marathon



DevOps Lifecycle - Operate

- This stage where the updated system gets operated

Environment creation tools

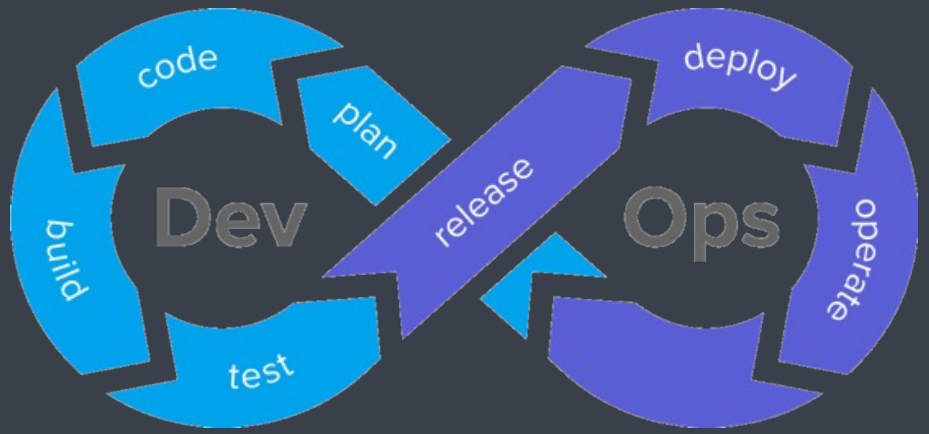
Local environment - vagrant

cloud environment → terraform, AWS - Cloud formation

Environment Configuration tools

- puppet, chef, ansible, saltstack

Tools



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

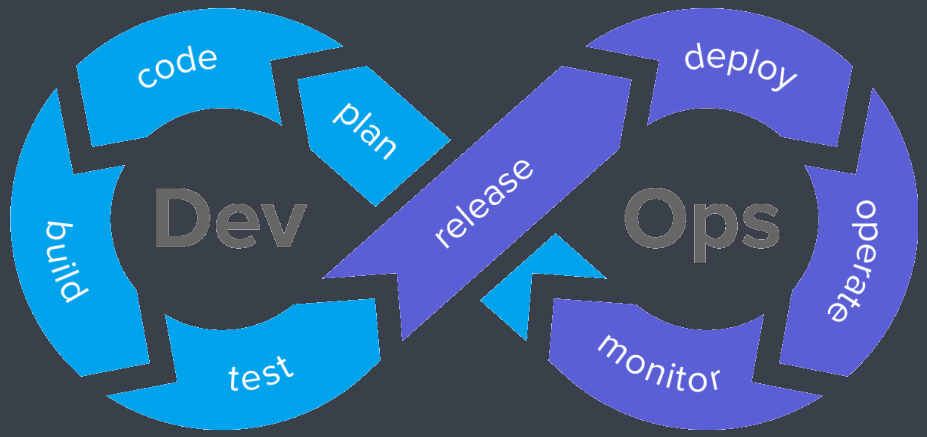
tools - Nagios, Splunk, DataDog, NewRelik

Tools

splunk>

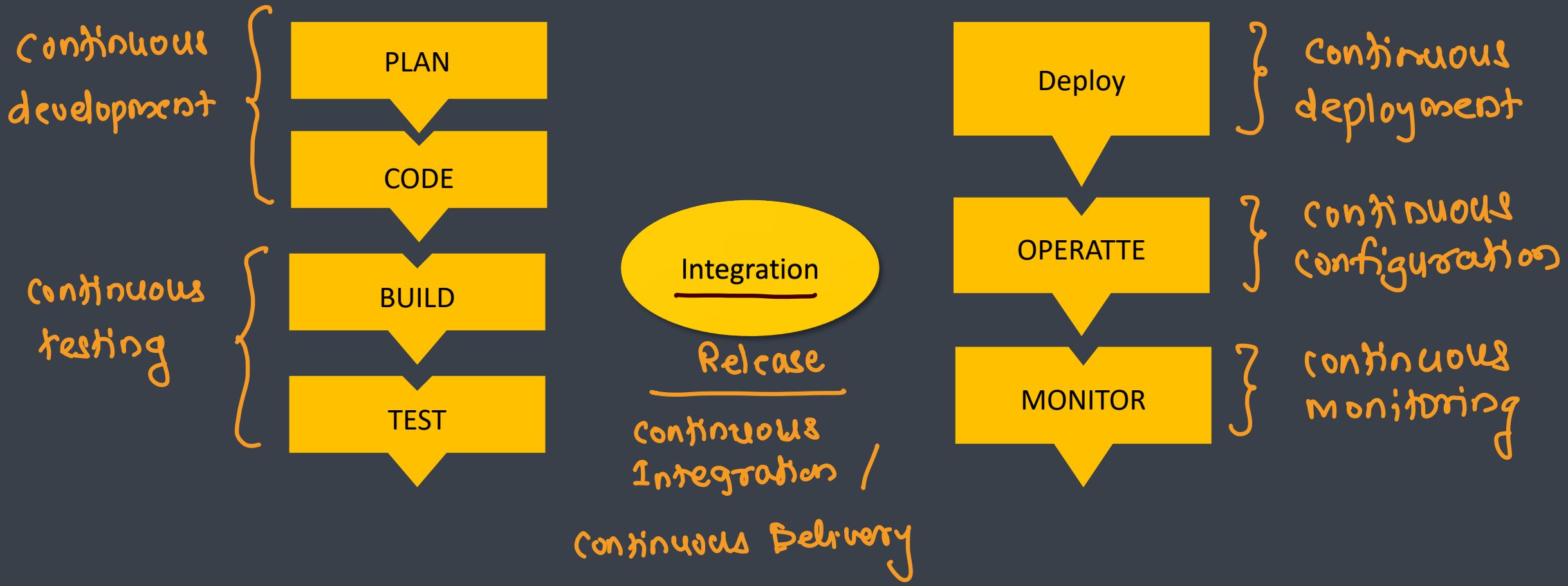
DATADOG

Nagios®



DevOps Terminologies

Continuous Learning



Responsibilities of DevOps Engineer

Linux

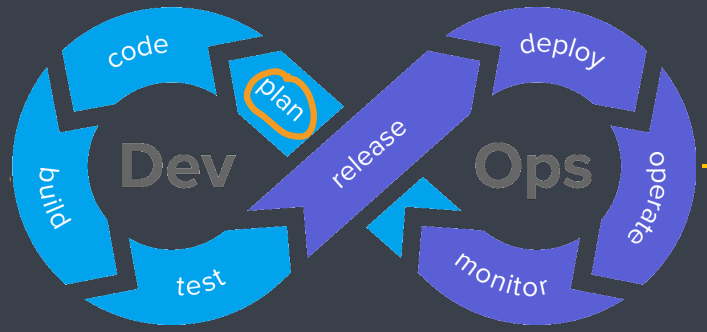
- Be an excellent sysadmin

cloud

- Deploy Virtualization

CCNA

- 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