DevOps

Software Development Lifecycle

Waterfall

Agile → Scrum → sprint by sprint

developer tester ops team

PLANNING

Talk to customer and understand the requirements

requirements

DEFINING

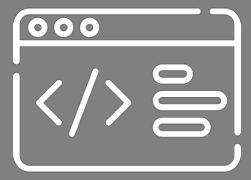
Define the requirements and stick to them

SRS

DESIGNING

Design the solution with right approach

DDS



BUILDING

Development following guidelines

untested 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

Responsibilities

web → webpack
android → .apk
ios → .ipa
windows → .msi
Ubuntu → .deb
Red Hat → .rpm
macos → .dmg

dev team

Developers and Testers

- Developers ↷ add features
  - Develop the application → SRS
  - 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

DevOps engineers ↷ VAPT
security professional, Security Auditor
adminstrator → local, network

Operations Team

↷ hardware / software → Network / machines / resources

- Infrastructure Management → environments
- Security & Compliance
- Deployment & Release Management
- Monitoring, Logging & Alerting
- Incident Management & Troubleshooting
- Cloud & Cost Optimization
- Backup & Disaster Recovery
- Collaboration & Support
- Performance & Capacity Planning

Challenges

Developers and Testers

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

node : 24 , mysql : 9.0

Operations Team

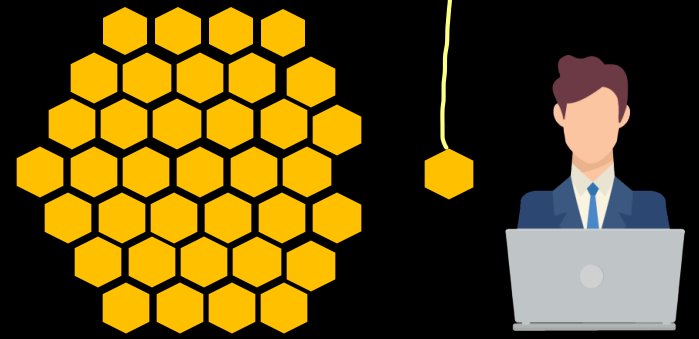
- Uptime → time the app is up & running
- Configure the huge infrastructure
- Diagnose and fix the issue
- stable versions

Node : 21 , MySQL : 8.0

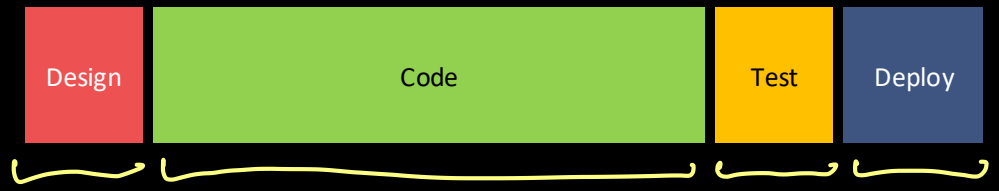
Waterfall Vs Agile

The Waterfall Process

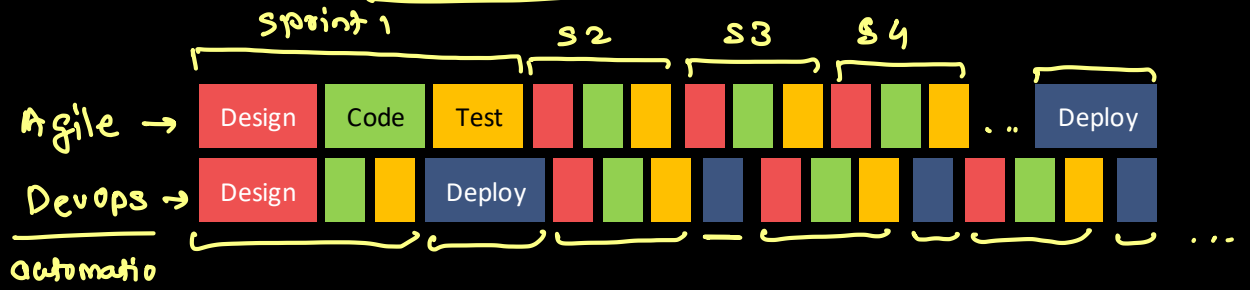
The Agile Process



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

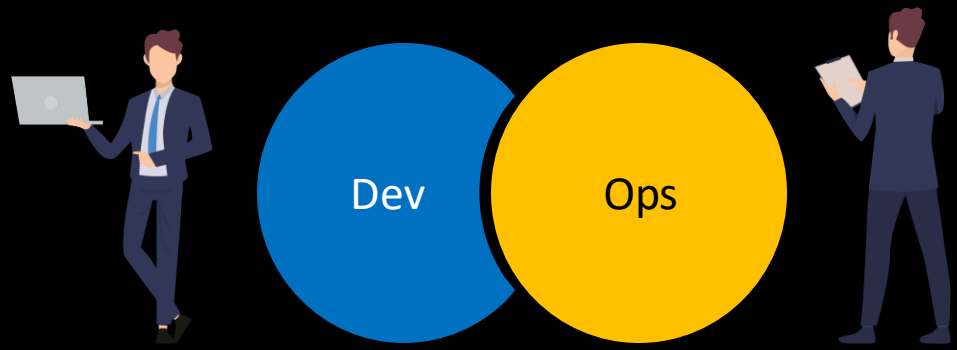


Problems

- Managing and tracking changes in the code is difficult : scm tools : git
- Incremental builds are difficult to manage, test and deploy : CI/CD pipeline → Jenkins
- Manual testing and deployment of various components/modules takes a lot of time : test automation → Selenium
- Ensuring consistency, adaptability and scalability across environments is very difficult task : configuration tools
  ↳ puppet, ansible
- Environment dependencies makes the project behave differently in different environments
  ↳ containerization → docker, podman

What is DevOps ?

- **DevOps** is a combination of **“Development”** and **“Operations.”**
- It’s both a **culture** and a **set of practices/tools** aimed at improving collaboration between **software developers (Dev)** and **IT operations (Ops)** teams
- The main goal is to **deliver software faster, more reliably, and with higher quality** — by **automating** and **integrating** the processes of software development and IT operations
- Before DevOps
  - Developers wrote code: threw it over to Ops
  - Ops deployed and managed it : often led to conflicts like “It works on my machine!”
- With DevOps
  - Dev and Ops **work together from start to finish**
  - They use **automation**, **CI/CD pipelines**, **containers**, **monitoring**, and **collaboration tools** to ensure smoother and faster delivery



Goals of Devops

↳ CI/CD pipeline

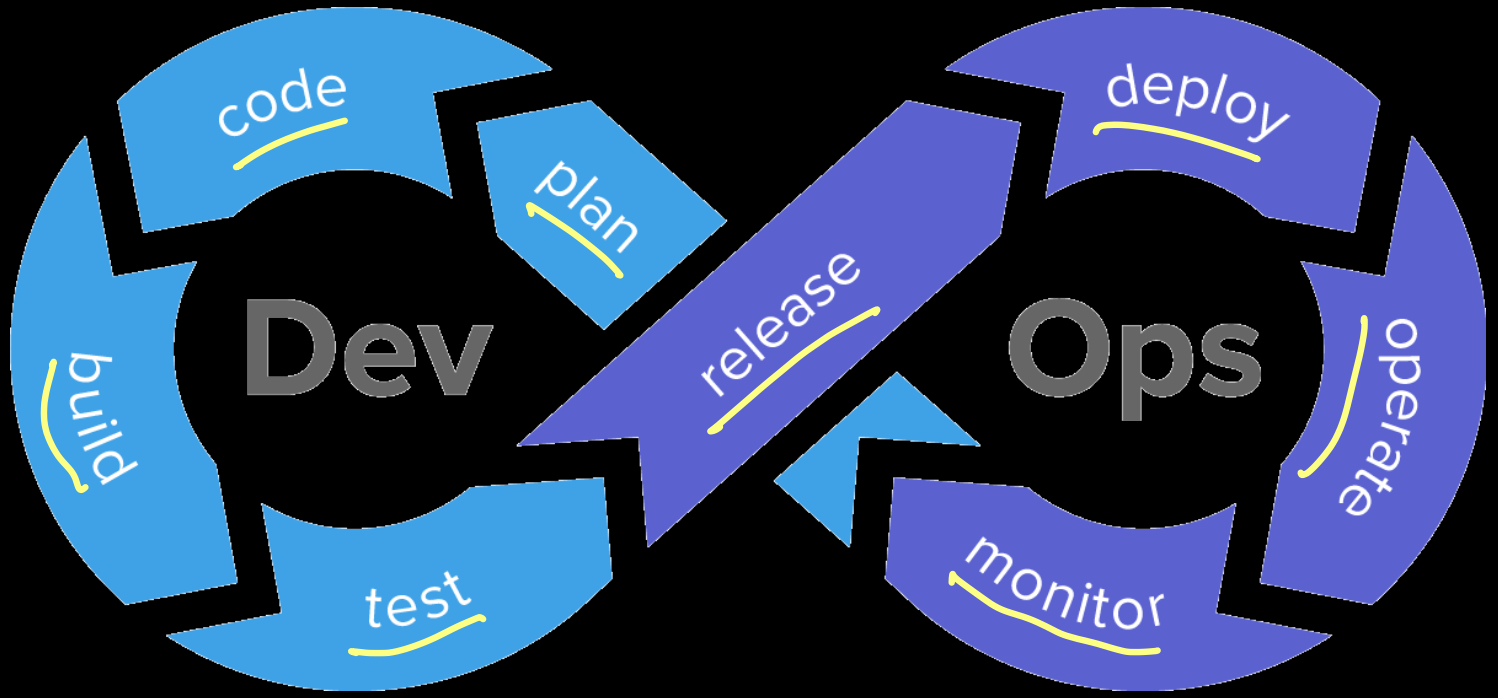
- **Faster delivery:** Shorten time from idea - deployment → Automation
- **Better collaboration:** Break silos between Dev and Ops
- **Automation:** Reduce manual errors and repetitive work → tools
- **Reliability:** Ensure systems are stable and scalable
- **Continuous improvement:** Use feedback to improve processes

Reasons to use DevOps

- **Predictability** → stable env & more uptime
  - DevOps offers significantly lower failure rate of new releases
- **Reproducibility** → rollback
  - Version everything so that earlier version can be restored anytime
- **Maintainability** → rollback
  - 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 → automation
  - 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 → logging

DevOps Lifecycle

sprint-by-sprint



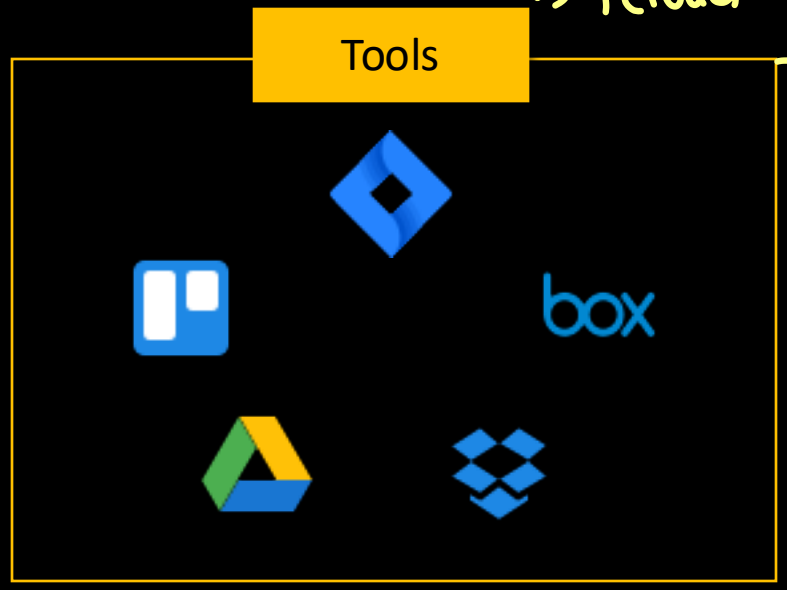
DevOps Lifecycle - Plan → Sprint planning Event

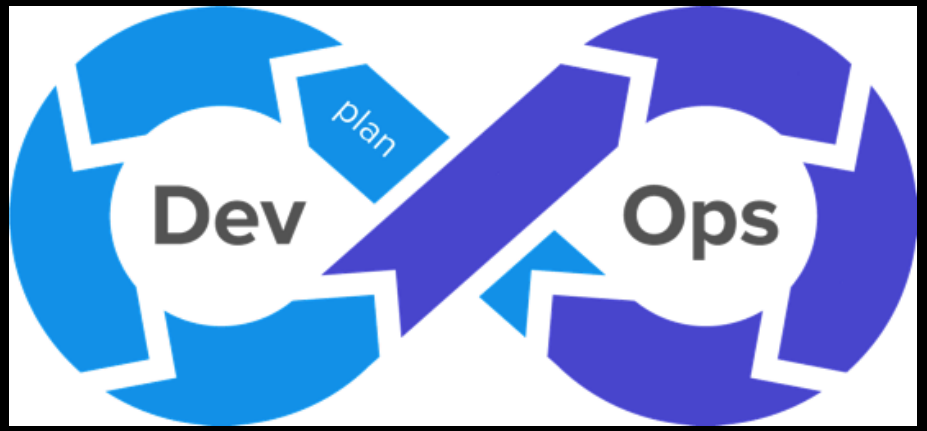
- First stage of DevOps lifecycle where you plan, track, visualize and summarize your project before you start working on it

simple tools
- → text files
- → Excel sheets

cloud tools
- → google drive
- → ms one drive
- → box
- → icloud

special tools
- → Jira
- → Trello
- → GitScrum
- → GitHub





DevOps Lifecycle - Code → developers → programming | coding

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

languages
- → C, C++
- → html | css | JS
- → Typescript
- → Go lang
- → Java
- → Rust
- → Iron
- → swift
- → Kotlin

Editors
- → vim, nano
- → VS code
- → atom
- → Sublime
- → windsuof
- → Cursor

IDEs
- → MS visual Studio
- → Webstorm
- → Jetbrains IDEA
- → Pychoom
- → Android studio
- → XCode

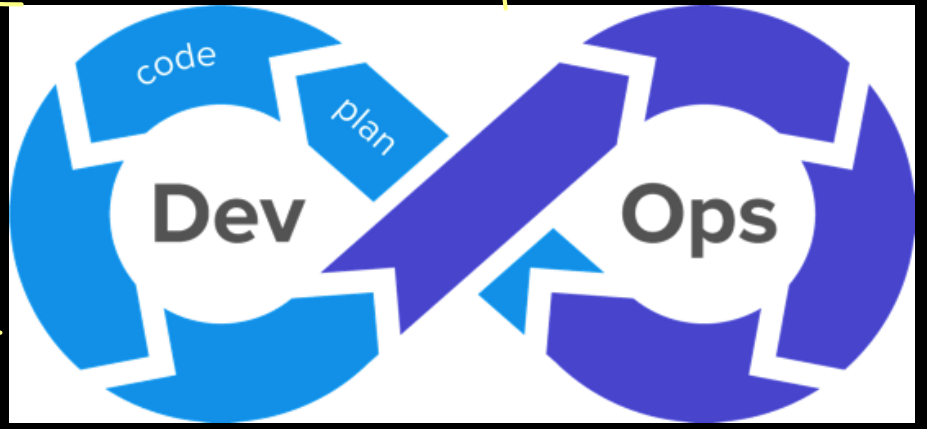
package manager
- → node → npm, yaon, pnpm
- → windows → choco
- → ubuntu → apt, snap
- → RedHat → yum, dnf, rpm
- → macos → brew
- → swift → cocoa pod
- → Java → maven, gradle

Tools






scm tools
- → local VCS → sourresafe, rcs
- → centralized VCS → cvs, svn
- → distributed VCS → git, bitkeeper, bazaar,



DevOps Lifecycle -Build

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

.java → .class → .jar

java + xml → gradle → .apk

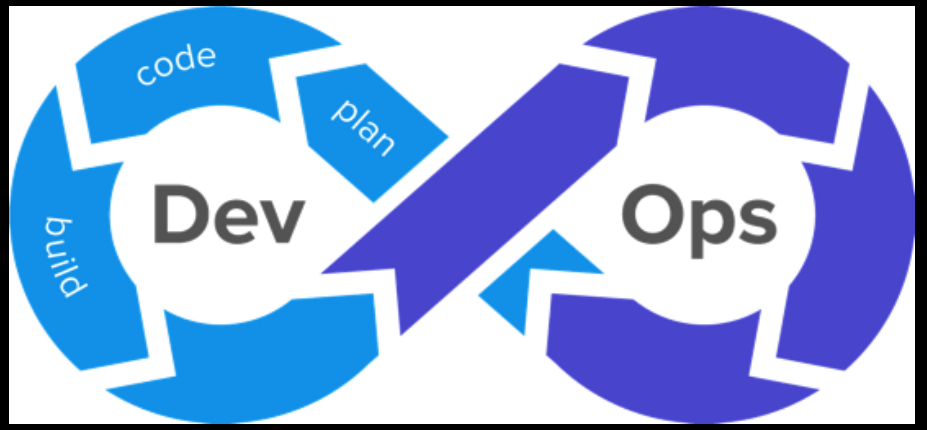
swift + xib → xcodebuild → .ipa

building tools → ant, maven, gradle, xcodebuild,

source code → building tool → package

Tools

maven

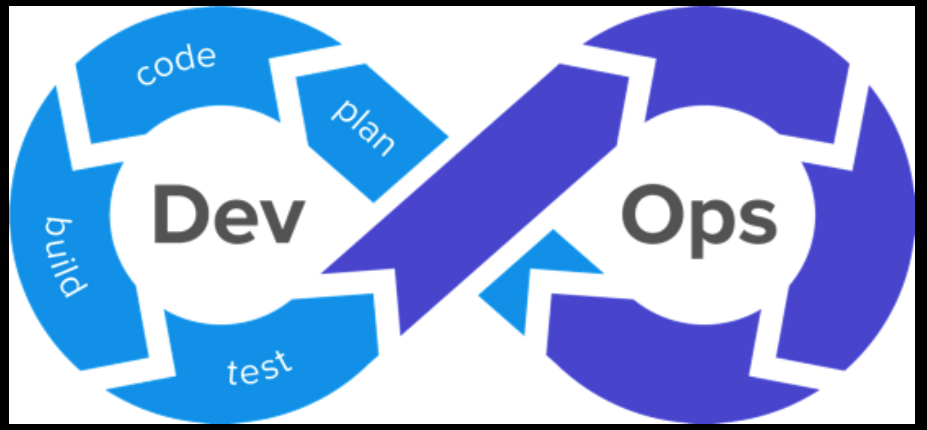


DevOps Lifecycle - Test

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

unit testing tools	Web UI testing	Performance Testing
→ java : Junit → JS/TS : Jasmin / Jest → C# : Nunit. → python: PyUnit	→ Selenium → Cypress	→ JMeter → WinRunner → Load Runner

Tools



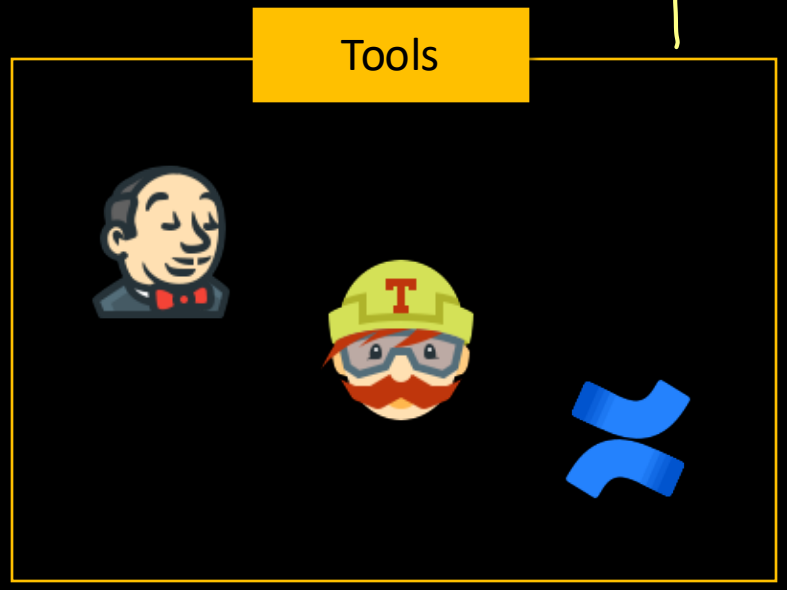
DevOps Lifecycle - Release : CI/CD pipeline : sequence of stages

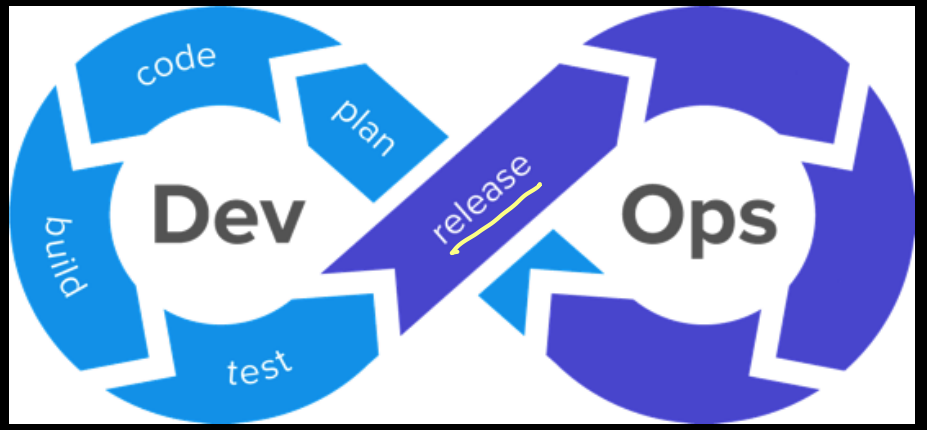
- This phase helps to integrate code into a shared repository using which you can detect and locate errors quickly and easily

CI Tools
- Travis CI
- GitLab CI
- Circle CI

CD tools
- Argo CD

CI/CD Tools
- Jenkins
- GitHub Actions
- Bamboo





DevOps Lifecycle - Deploy → moving an app from one to another env

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

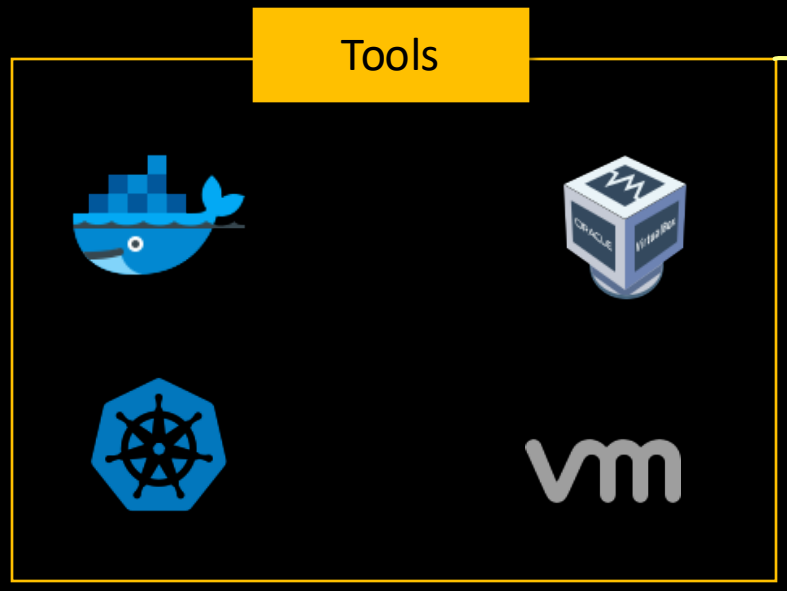
deployment strategies

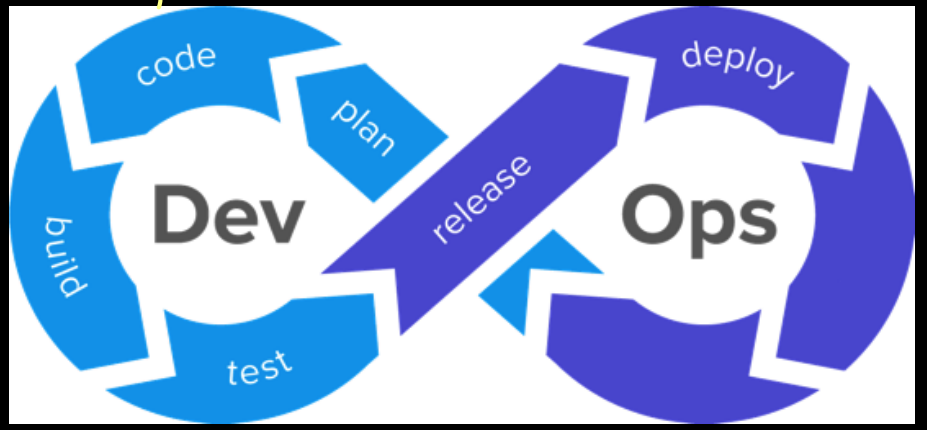
① traditional deployment
↳ using physical machines
↳ deprecated

② virtualized deployment
↳ using virtual machines
→ on premise (private)
→ virtual box
→ VMware
→ Qemu
→ Bluestack
→ on cloud
→ aws → EC2 instance
→ azure → virtual machine
→ GCP → virtual machine

③ containerized deployment
↳ using containers
→ container runtime
→ docker, containerd, podman, LXC, LXD
→ container orchestration
→ K8S, marathon, mesos, Keda, docker swarm

Tools





DevOps Lifecycle - Operate → infrastructure creation & configuration

- This stage where the updated system gets operated

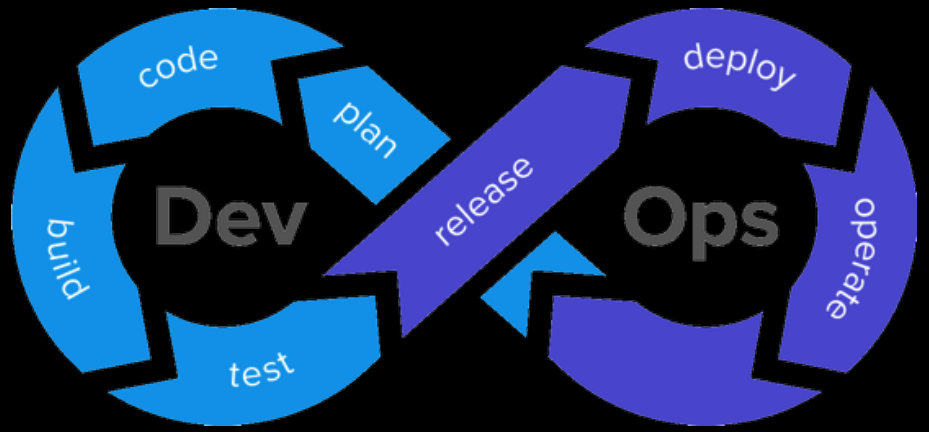
infrastructure creation tools

→ terraform → HashiCorp
→ aws cloud formation
→ vagrant → local vm management

infrastructure configuration tools

→ puppet
→ chef
→ ansible

Tools

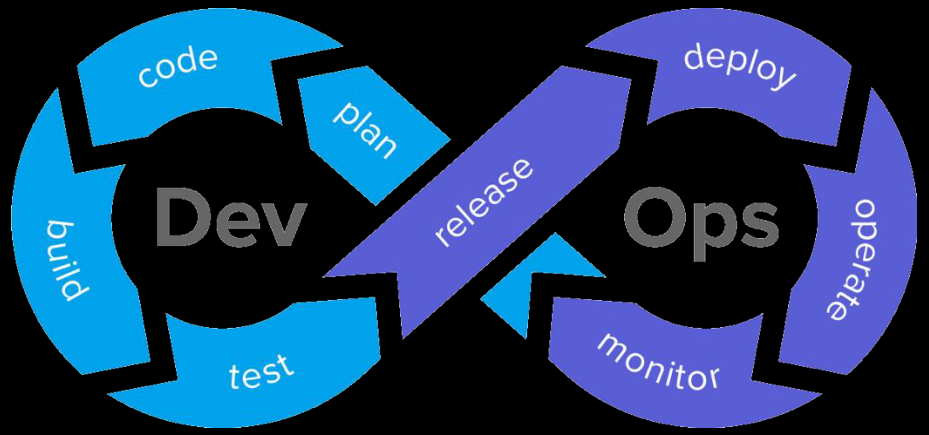


DevOps Lifecycle - Monitor

- 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

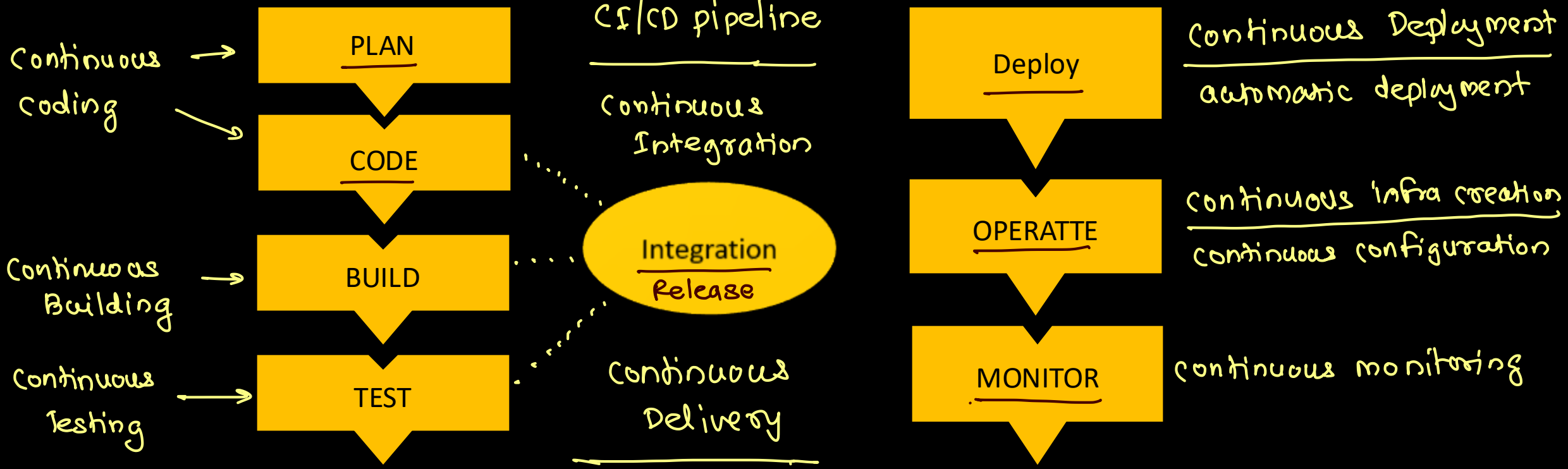
monitoring tools

→ Nagios
→ Splunk
→ ELK Stack
→ Datadog
→ New Relic
→ Prometheus
→ Grafana
→ Kibana



DevOps Terminologies

Continuous learning



Responsibilities of DevOps Engineer

linux, windows, mac os

- Be an excellent sysadmin

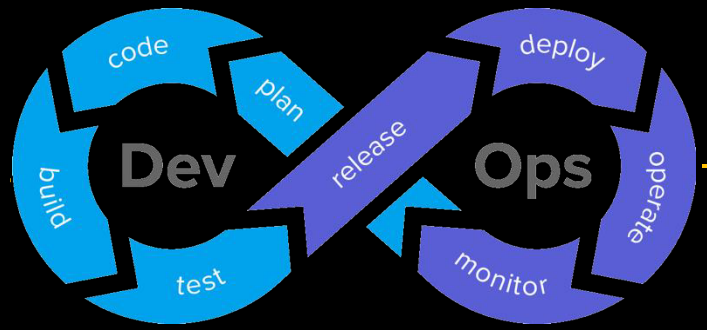
hardware & os virtualization

- Deploy Virtualization

network engineer

- 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