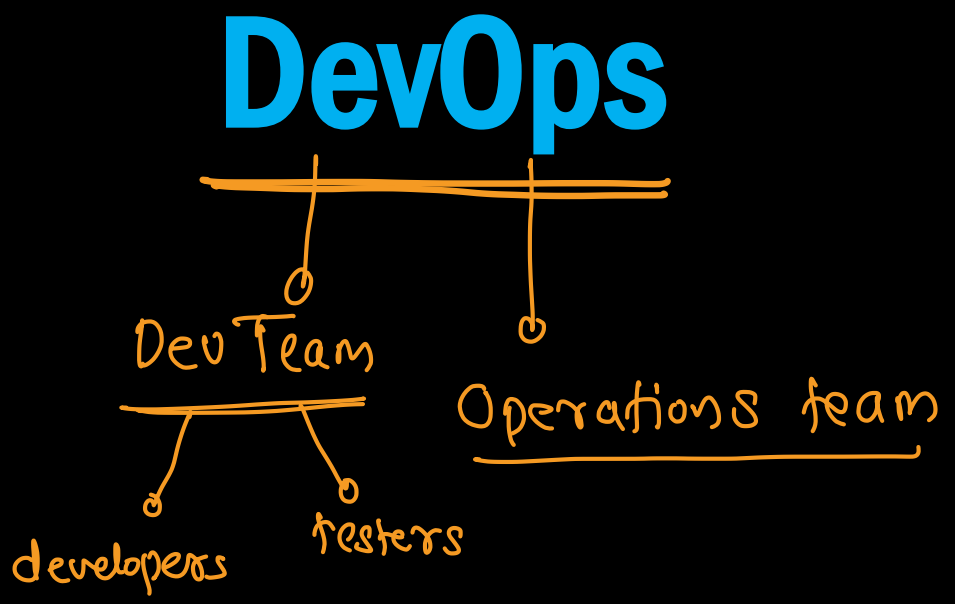

DevOps
Dev Team
Operations team
developers
testers

Responsibilities

Dev Team

- **Developers**
  - Requirement understanding
  - Coding → languages, technologies
  - Version control → SCM → git
  - Unit testing → TDD
  - Documentation → comments
  - Collaboration → with other team members, testers & ops team
- **Testers**
  - Requirement analysis → for testing
  - Test planning, designing and execution
  - Defect tracking → Jira / bugzilla
  - Reporting
  - collaboration → with team members and developers

Operations Team

- Infrastructure Management → VM / containers / network
- Security & Compliance → auditing, licenses
- Deployment & Release Management → deployment → moving app from one environment to another
- Monitoring, Logging & Alerting → maintain uptime
- Incident Management & Troubleshooting → downtime, hacking events, data theft
- Cloud & Cost Optimization → AWS / Azure / GCP
- Backup & Disaster Recovery → databases, machines
- Collaboration & Support
- Performance & Capacity Planning
  - → no of resources required to maintain the app uptime

Challenges

Dev Team

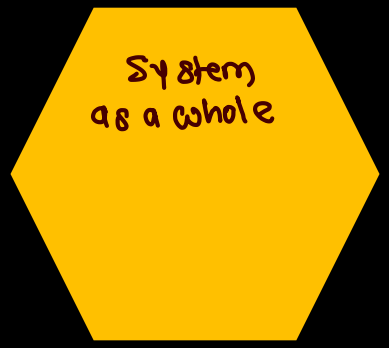
- Environment inconsistency
- Delayed feedback loop
- Integration issues
- Manual build and deployment
- Poor visibility after deployment
- Pressure for faster delivery
- Communication gaps

Operations Team

- Frequent and sometimes unstable releases → deployments
- Lack of environment standardization
- Manual configuration management
- Limited collaboration with dev team
- Poor monitoring and alert fatigue
- Downtime and ~~indent~~ pressure
- Security and compliance

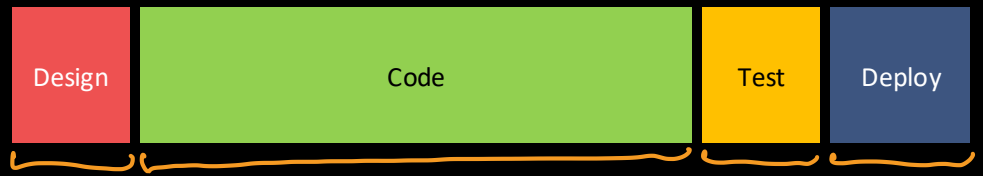
Waterfall Vs Agile

The Waterfall Process

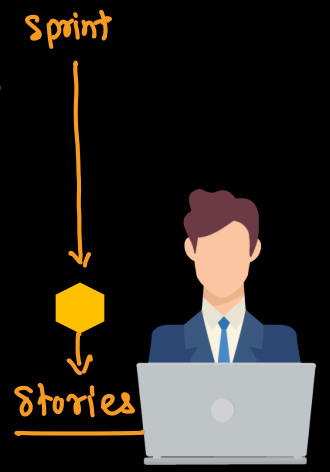


takes lot of time

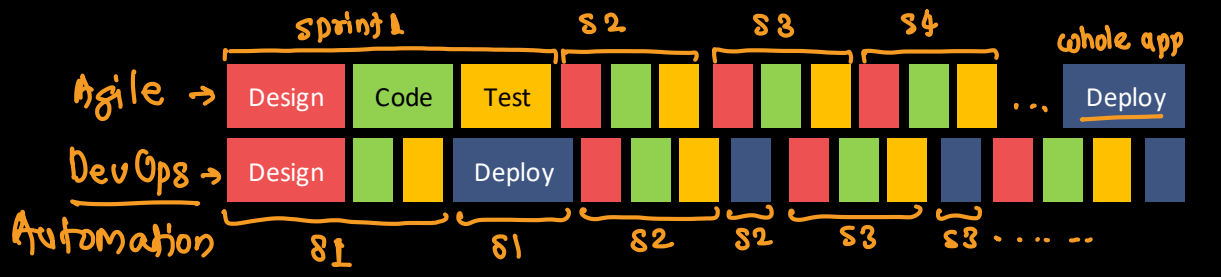
This project has got so big.
I am not sure I will be able to deliver it!



The Agile Process

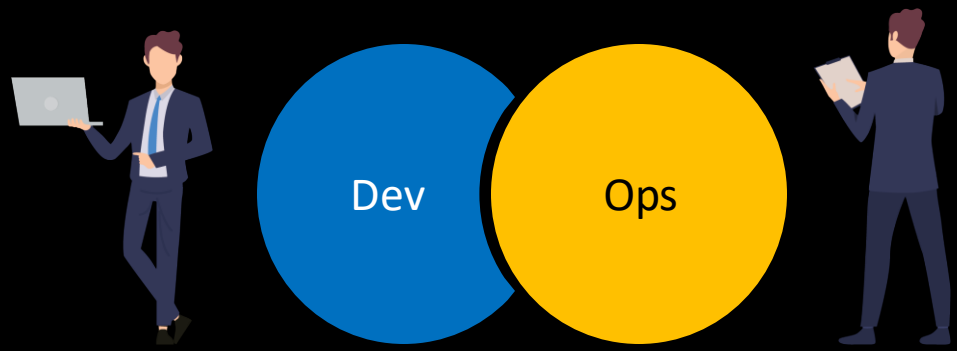


It is so much better delivering
this project in bite-sized sections (sprints)



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
- It’s **not just a tool or role**, but a **culture, methodology, and set of practices** that bring together: **Developers (who build software) and Operations (who deploy and manage it)** to **work collaboratively**, with the goal of: Delivering software **faster, more reliably**, and **continuously** to end-users



How DevOps Solves Challenges

DevOps Solution	What It Fixes	Tools / Practice
Continuous Integration (CI)	Merge & test code frequently	Jenkins, GitHub Actions, GitLab CI
Continuous Delivery (CD)	Automate deployment pipelines	ArgoCD, Spinnaker, Bamboo
Infrastructure as Code (IaC)	Standardize environments	Terraform, Ansible, Puppet
Containerization	Eliminate “works on my machine”	Docker, Kubernetes
Monitoring & Logging	Give both teams visibility	Prometheus, Grafana, ELK Stack
Collaboration Tools	Improve communication	Slack, Microsoft Teams, Jira
Automation Testing	Faster feedback for devs	Selenium, Cypress, JMeter

Goals of DevOps

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

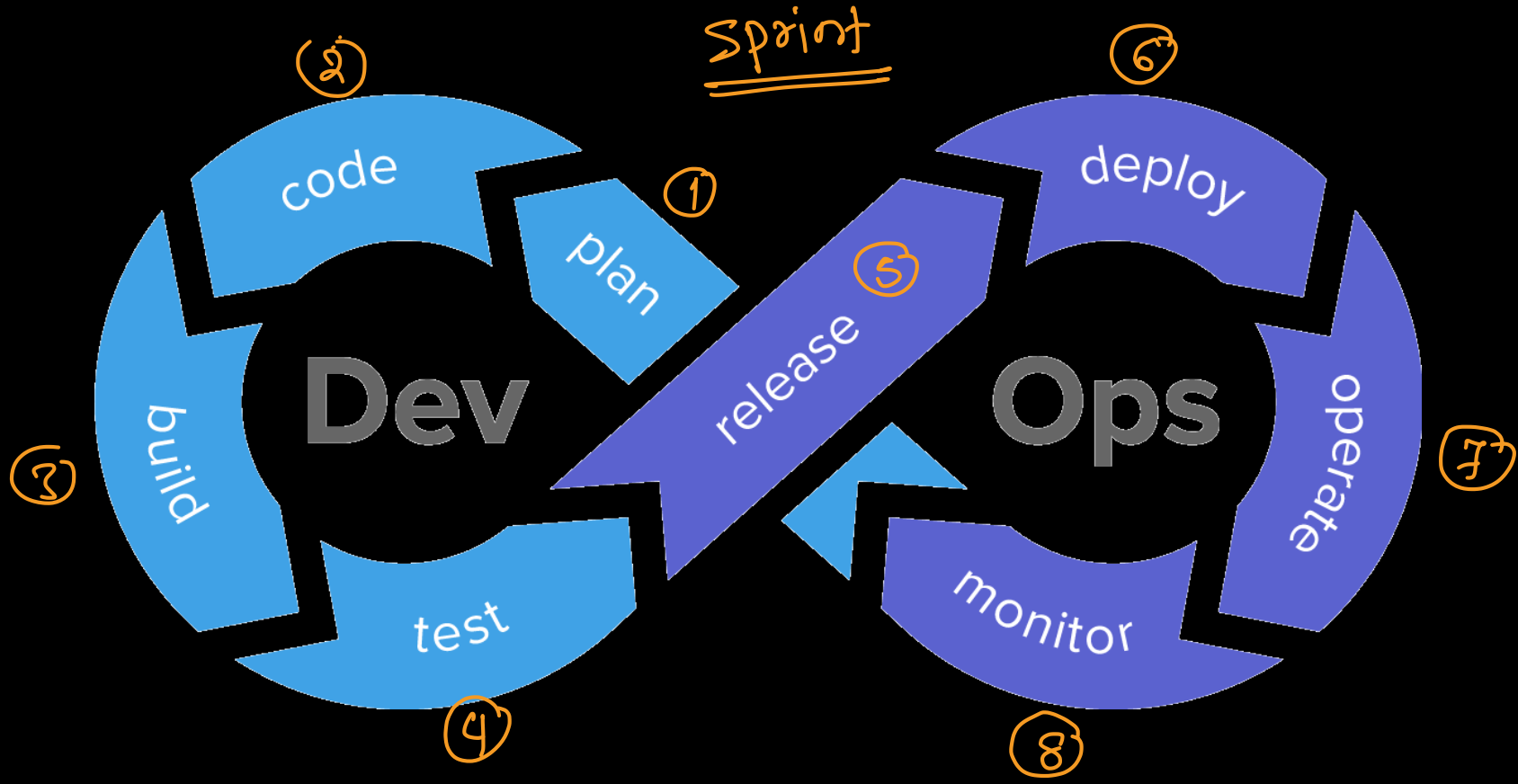
Key Principles of DevOps

Principle	Description
Collaboration & Communication	Break silos between Dev, QA, and Ops. Everyone shares responsibility for success.
Automation	Automate everything: builds, testing, deployment, monitoring - to reduce manual errors.
Continuous Integration (CI)	Developers merge code frequently and automatically test it.
Continuous Delivery (CD)	Automatically release tested code to staging/production environments.
Continuous Monitoring	Track system health, user feedback, and performance in real-time.
Infrastructure as Code (IaC)	Manage infrastructure (servers, networks) through code, not manual setup.
Continuous Feedback	Learn from real-time data to improve the next iteration.

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



PLAN – Planning and Tracking

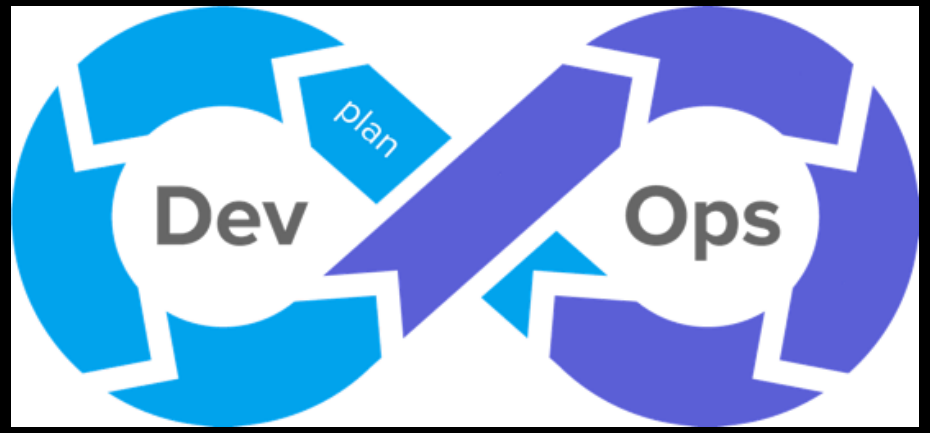
- **Goal:** Define project goals, features, and timelines → Requirement analysis
- **Activities:**
  - Gather and prioritize requirements → product owner
  - Plan sprints, track progress, and manage tasks → visibility
  - Collaborate between development, QA, and operations

transparency

Sprint backlog

Story 1 → 8 hrs
Story 2 → 20 hrs
Story 3 → 20 hrs

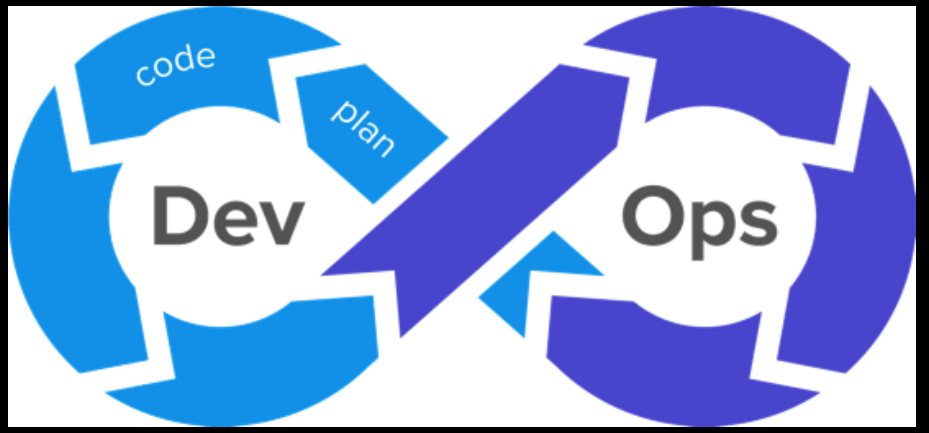
- **Common Tools:**
  - Agile Project Management: Jira, Trello, Asana, ClickUp, Azure Boards
  - Documentation: Confluence, Notion, Google Docs
    - → project
    - → knowledge base
    - → sprint event logs



CODE – Development → developers

- **Goal:** Write, review, and manage code efficiently
  - → white box testing
- **Activities:**
  - Code development using IDEs / Editors
  - Version control via Git → SCM tool
  - Code review and collaboration
- **Common Tools:**
  - **Version Control:** Git, GitHub, GitLab, Bitbucket, SVN, CVS, perforce
  - **Code Review:** GitHub PRs, Gerrit, Crucible, Gitlab MR, PR - Pull Request, MR - Merge Request
  - **IDEs:** VS Code, IntelliJ IDEA, PyCharm, Eclipse
  - **Code Quality:** SonarQube, Codacy, ESLint, Prettier

- Editors → VS code, vim, atom, sublime, cursor, windsurf, bolt
- package managers →
  - → JS/TS → npm, yarn, pnpm
  - → python → pip, conda, poetry
  - → ruby → gem
  - → swift → cocoapod



BUILD – Integration and Compilation

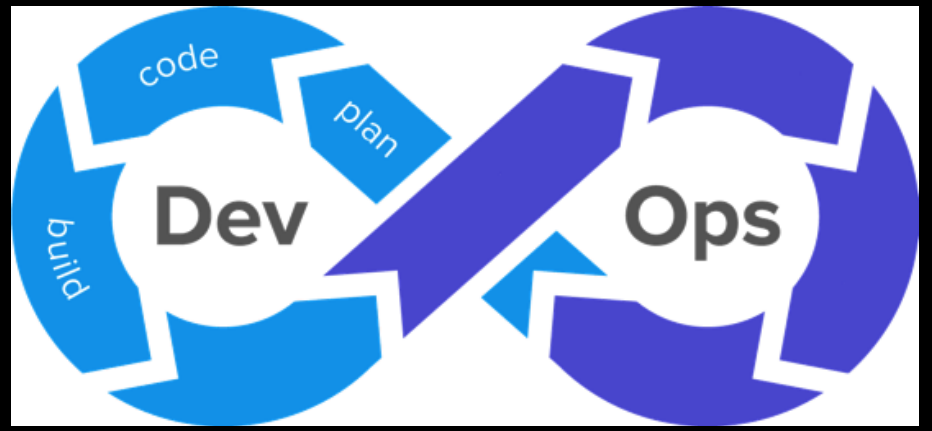
- **Goal:** Compile source code into executable artifacts
- **Activities:** and create package
  - Continuous Integration (CI)
  - Build automation and packaging
  - Dependency management
- **Common Tools:**
  - **Build Automation:** Maven, Gradle, Ant (→ deprecated), npm, Make
  - **CI Servers:** Jenkins, GitLab CI, CircleCI, GitHub Actions, TeamCity
  - **Artifact Repository:** Nexus, JFrog Artifactory, AWS CodeArtifact

deployable packages

windows → .msi
linux → debian (.deb)
RedHat (.rpm)
macos → .dmg
ios → .ipa
android → .apk, .aab
websites → webpack
java → .jar or .war

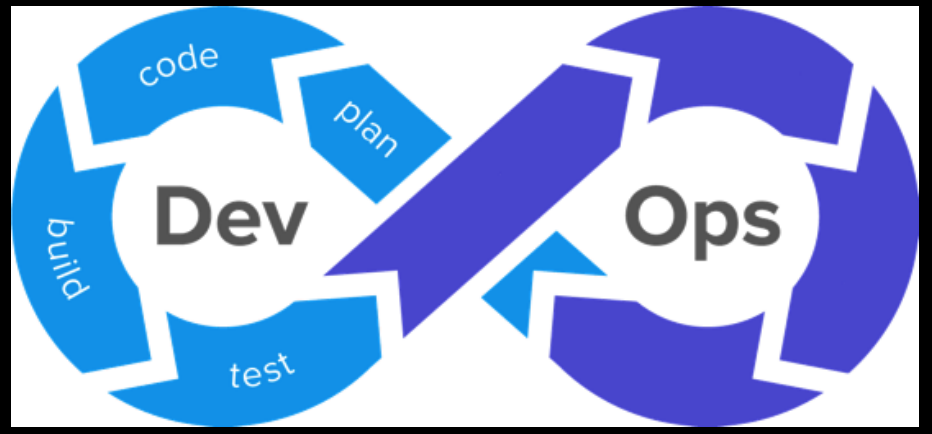
build

→ compile the code
→ integrate all dependencies
  → libraries
  → frameworks
  → resources
  → config files
  → documentation
→ build deployable package



TEST – Automated Testing

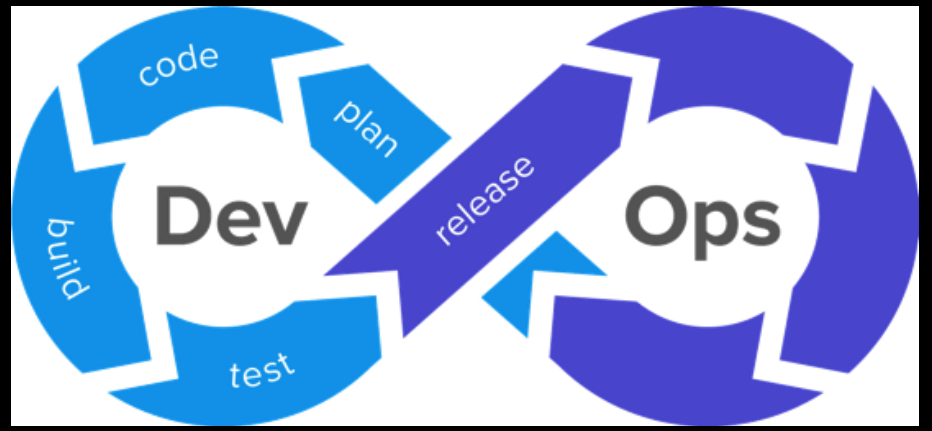
- **Goal:** Ensure software quality and reliability
- **Activities:**
  - Run automated tests (unit, integration, functional)
  - Identify bugs early in the CI pipeline
- **Common Tools:**
  - **Unit Testing:** JUnit, PyTest, NUnit, Mocha, Jest
  - **Functional/UI Testing:** Selenium, Cypress, Playwright
  - **API Testing:** Postman, RestAssured, SoapUI
  - **Performance Testing:** JMeter, Gatling, k6
  - **Security Testing:** OWASP ZAP, Burp Suite, SonarQube Security



RELEASE – Versioning and Approval

- **Goal:** Prepare the build for deployment into production-like environments automatically
- **Activities:**
  - Version tagging, change approval, and documentation
  - Store tested builds for release → artifact → artifact Repository
    - ↳ deployable package
- **Common Tools:**
  - **Release Automation:** Jenkins, GitLab CI/CD, Bamboo, Azure DevOps, TravisCI, CircleCI, ArgoCD, GitHub Actions
  - **Version Control:** Git Tags, Semantic Versioning, GitHub Releases
  - **Change Management:** ServiceNow, Jira Service Management

CI/CD pipeline → sequence of stages

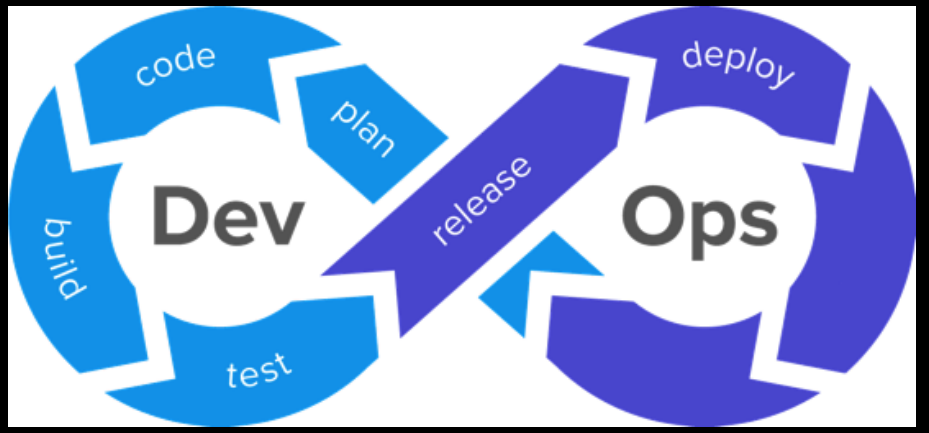


DEPLOY – Continuous Deployment / Delivery

- **Goal:** Deliver the software to target environments (staging, production)
- **Activities:**
  - Continuous Delivery (manual approval)
  - Continuous Deployment (fully automated)
  - Environment configuration via Infrastructure as Code (IaC)
- **Common Tools:**
  - **Containerization:** Docker, Podman
  - **Orchestration:** Kubernetes, OpenShift, Docker Swarm
  - **IaC (Infrastructure as Code):** Terraform, Ansible, Puppet, Chef
  - **Cloud Platforms:** AWS, Azure, GCP, DigitalOcean

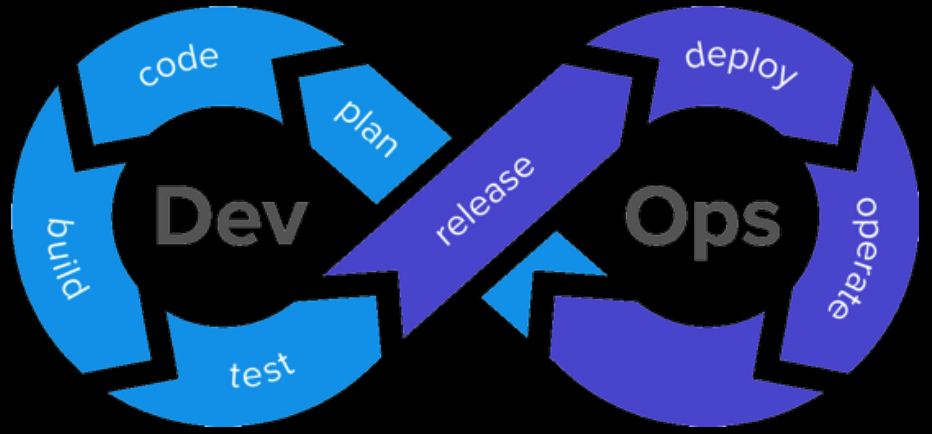
deployment

→ traditional → physical machines
  ↳ deprecated
→ virtualized deployment → VMs
  ↳ vmware, VBox, parallels
→ containerized deployment → containers
  ↳ container runtime – docker, LXC, LXD, podman, rkt.
  ↳ orchestration tools → Docker swarm, K8S, marathon, mesos



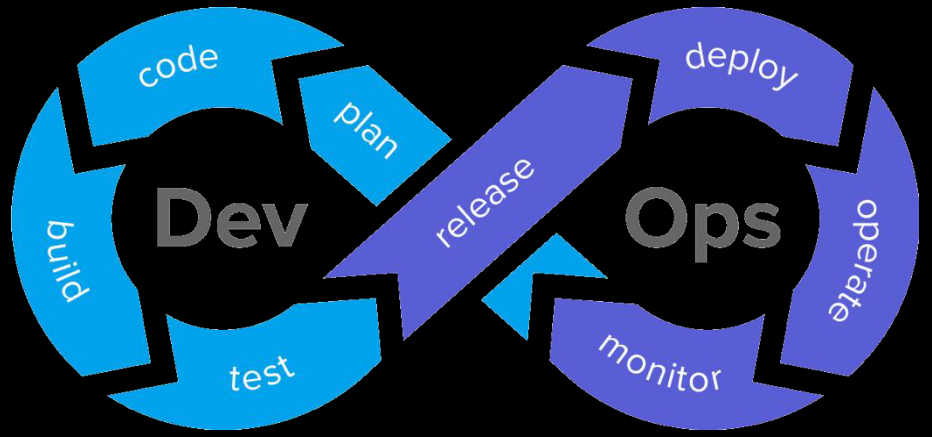
OPERATE – Configuration and Infrastructure Management

- **Goal:** Manage and maintain systems after deployment
- **Activities:**
  - Monitor uptime, performance, and incidents
  - Automate infrastructure provisioning and scaling
- **Common Tools:**
  - **Configuration Management:** Ansible, Puppet, Chef, SaltStack
  - **Server Monitoring:** Nagios, Datadog, New Relic
  - **Log Management:** ELK Stack (Elasticsearch, Logstash, Kibana), Fluentd, Splunk



MONITOR – Continuous Feedback

- **Goal:** Continuously observe system health, user experience, and performance
- **Activities:**
  - Collect and analyze metrics
  - Detect failures and trigger alerts
  - Feed insights back into planning phase
- **Common Tools:**
  - **Monitoring:** Prometheus, Grafana, Zabbix, Datadog
  - **Logging:** ELK Stack, Graylog, Splunk, Fluentd
  - **Alerting:** PagerDuty, Opsgenie, Slack Alerts
  - **Feedback:** Sentry, New Relic, AppDynamics



DevOps Toolchain Example

Stage	Tools
Plan	Jira, Confluence, Notion
Code	Git, GitHub, Bitbucket
Build	Jenkins, Maven, Gradle
Test	Selenium, Cypress, JMeter
Release	GitLab CI/CD, Bamboo, Jenkins
Deploy	Docker, Kubernetes, ArgoCD
Operate	Ansible, Puppet, Terraform
Monitor	Prometheus, Grafana, ELK Stack

Responsibilities of DevOps Engineer

linux administrator

- Be an excellent sysadmin

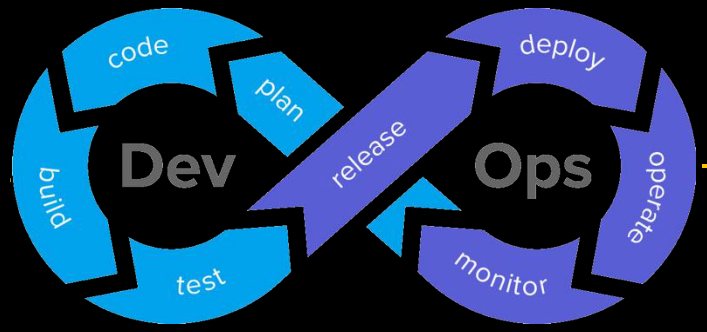
Virtual machine → VMware

- Deploy Virtualization

network engineering

- Hands-on experience in network and storage

- Introduction to coding



- Soft skills

- Automation tools

- Software Testing knowledge

- IT security