DevOps Introduction

. What is DevOps?

Definition:
DevOps is a **set of practices, cultural philosophies, and tools** that combine **software development (Dev)** and **IT operations (Ops)** to deliver applications and services **faster more reliably**.

Key Idea:

- Break silos between development and operations teams.
- Automate repetitive tasks.
- Ensure continuous delivery of high-quality software.

Characteristics of DevOps:

- Collaboration between developers and operations.
- Continuous integration, testing, and deployment.
- Focus on **automation, monitoring, and feedback loops**.

Example:
Without DevOps: Developers finish code → throw it to operations → deployment fails → long bug-fix cycles.
With DevOps: Developers p ush code → automated CI/CD pipeline tests and deploys → faster, reliable releases.

---

2. Why Do You Need DevOps?

Organizations adopt DevOps to:

1. **Accelerate Delivery:**
    - Reduce release cycles from months to days or hours.
    - Example: Deploying new features weekly instead of quarterly.
2. **Improve Collaboration:**
    - Developers and operations share responsibilities.
    - Example: Both teams maintain infrastructure as code.
3. **Increase Reliability:**

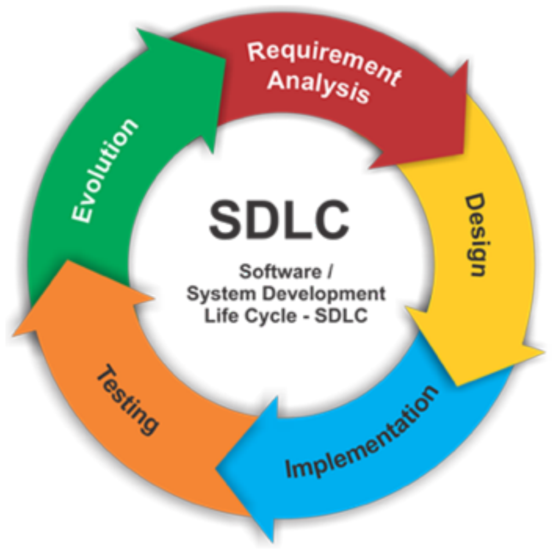
- Automated testing and monitoring reduce production errors.
- Example: CI pipelines catch bugs before deployment.

4. **Automate Repetitive Tasks:**
   - Builds, deployments, and environment setup are automated.
   - Example: Using Jenkins or GitHub Actions to deploy code automatically.
5. **Enhance Scalability & Flexibility:**
   - Cloud-native practices allow applications to scale dynamically.
   - Example: Kubernetes manages containerized app scaling automatically.
6. **Foster Continuous Improvement:**
   - Monitoring feedback leads to better performance, security, and user experience.

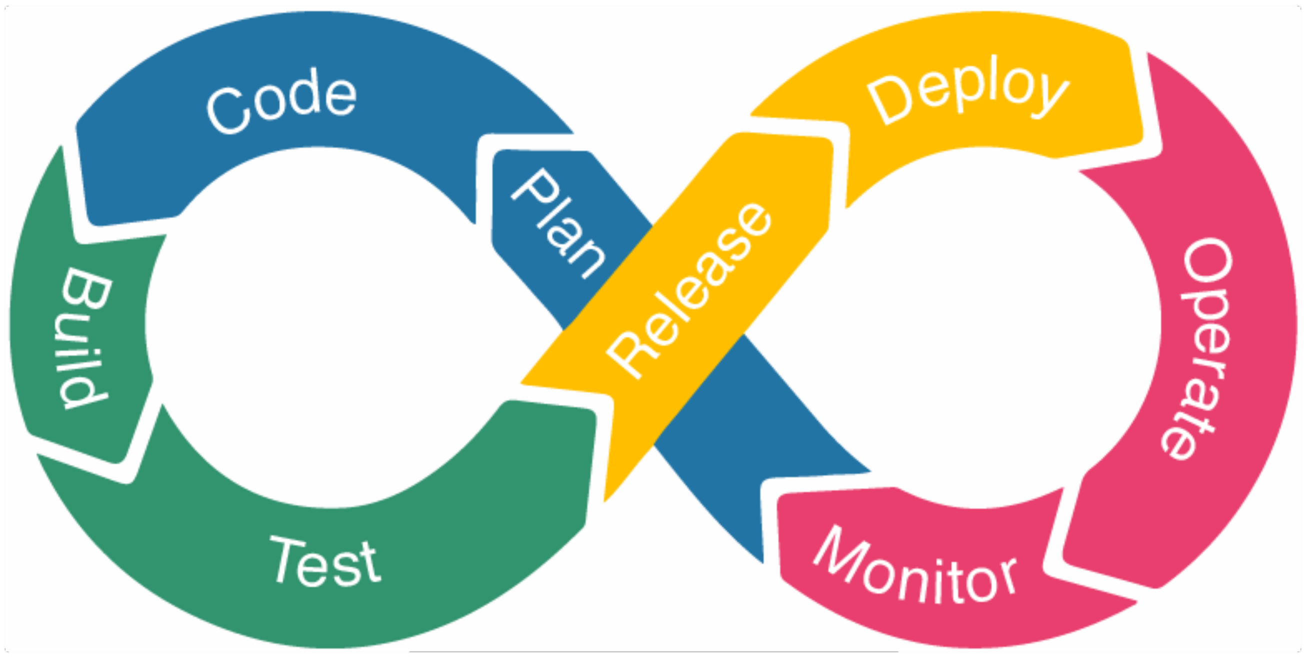
---

3. DevOps Lifecycle

SDLC Lifecycle



DevOps Lifecycle



DevOps lifecycle represents the **continuous process** from development to production. The main stages are:

Stage	Description	Tools / Examples
Plan	Requirement gathering, task planning	Jira, Trello, Confluence
Code	Writing application code, unit test etc.	Git, GitHub, GitLab
Build	Compile code and create deployable artifacts	Maven, Gradle, npm, Dockerfile
Test	Automated testing for quality assurance	Selenium, JUnit, pytest
Release	Package and release code	Jenkins, GitLab CI/CD, CircleCI
Deploy	Deploy to staging/production	Kubernetes, Docker, Ansible, Helm, Kustomize
Operate	Manage infrastructure and monitor apps	AWS CloudWatch, Prometheus, ELK
Monitor	Track performance and collect feedback	Grafana, Nagios, Datadog

Key Point: DevOps is **not linear** — it's **continuous and iterative**. Feedback from monitoring loops back into planning and coding.

---

4. DevOps Principles

DevOps is guided by several principles:

1. **Culture of Collaboration:**
    - Dev and Ops work together throughout the lifecycle.

2. **Automation:**
    - Automate builds, tests, deployments, and infrastructure provisioning.
3. **Continuous Integration & Continuous Delivery (CI/CD):**
    - Integrate code frequently and deploy continuously.
4. **Measurement:**
    - Track performance metrics, deployment frequency, error rates, and system health.
5. **Sharing Knowledge:**
    - Encourage team learning and transparency.
6. **Infrastructure as Code (IaC):**
    - Manage infrastructure declaratively using code (Terraform, Ansible).
7. **Monitoring and Feedback:**
    - Continuous feedback helps improve processes and application reliability.

---

5. DevOps Practices

DevOps is implemented using a set of best practices:

Practice	Description	Benefits
Continuous Integration (CI)	Merge code frequently; automated builds and tests	Early bug detection
Continuous Delivery (CD)	Automate release process to staging	Faster releases
Continuous Deployment	Automate deployment to production	Immediate user access
Version Control	Track changes in code and config	Easy rollback, collaboration
Automated Testing	Unit, integration, and UI testing	Higher software quality
Infrastructure as Code (IaC)	Manage servers and configs as code	Repeatable, auditable environments
Configuration Management	Standardize environment setup	Reduce manual errors
Monitoring & Logging	Track application and infrastructure health	Detect and fix issues quickly
Collaboration & Communication	Shared responsibilities, chatOps	Improved team efficiency

---

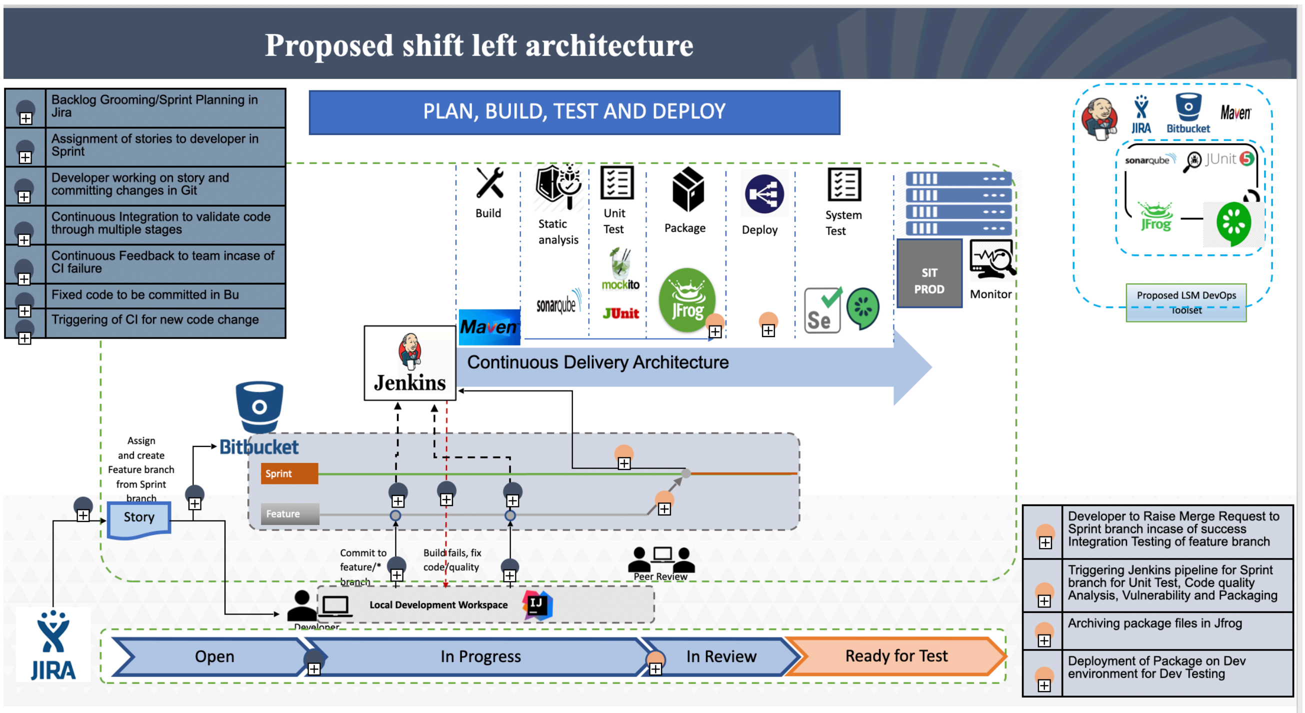
6. Tools in DevOps

DevOps relies on a **toolchain** across different lifecycle stages:

Stage	Tools / Examples	Purpose
Version Control	Git, GitHub, GitLab, Bitbucket	Track source code
CI/CD	Jenkins, GitLab CI, CircleCI, GitHub Actions	Automate build, test, deploy
Configuration Mgmt	Ansible, Chef, Puppet	Automate server setup
Containerization	Docker	Package apps into portable containers
Orchestration	Kubernetes, OpenShift	Deploy and manage containers at scale
Infrastructure as Code (IaC)	Terraform, CloudFormation	Provision and manage cloud resources
Monitoring & Logging	Prometheus, Grafana, ELK Stack, Datadog	Observe system health
Collaboration & ChatOps	Slack, Microsoft Teams, Mattermost	Team communication, alerting
Security	SonarQube, Trivy, Snyk	Integrate security in DevOps pipelines

Key Insight: DevOps is **not a single tool** — it's a combination of **culture, practices, and tools**.

CI


Proposed shift left architecture
Backlog Grooming/Sprint Planning in Jira
Assignment of stories to developer in Sprint
Developer working on story and committing changes in Git
Continuous Integration to validate code through multiple stages
Continuous Feedback to team incase of CI failure
Fixed code to be committed in Bu
Triggering of CI for new code change
PLAN, BUILD, TEST AND DEPLOY
Build
Static analysis
Unit Test
Package
Deploy
System Test
SIT
PROD
Monitor
Maven
sonarqube
mockito
JUnit
JFrog
Jenkins
Continuous Delivery Architecture
Assign and create Feature branch from Sprint branch
Story
Bitbucket
Sprint
Feature
Commit to feature/* branch
Build fails, fix code/quality
Peer Review
Developer
Local Development Workspace
JIRA
Open
In Progress
In Review
Ready for Test
JIRA
Bitbucket
Maven
sonarqube
JUnit 5
JFrog
Proposed LSM DevOps Toolset
Developer to Raise Merge Request to Sprint branch incase of success Integration Testing of feature branch
Triggering Jenkins pipeline for Sprint branch for Unit Test, Code quality Analysis, Vulnerability and Packaging
Archiving package files in Jfrog
Deployment of Package on Dev environment for Dev Testing

Conclusion

- **DevOps** bridges development and operations for faster, reliable software delivery.
- **Why DevOps:** Faster delivery, collaboration, automation, monitoring, continuous improvement.
- **Lifecycle:** Plan → Code → Build → Test → Release → Deploy → Operate → Monitor (iterative).
- **Principles:** Collaboration, automation, CI/CD, measurement, sharing, IaC, monitoring.
- **Practices:** CI/CD, automated testing, IaC, monitoring, configuration management.
- **Tools:** Git, Jenkins, Docker, Kubernetes, Terraform, Prometheus, Grafana, Ansible.

> DevOps is a **mindset, methodology, and tool ecosystem** — mastering it improves **software quality, delivery speed, and team collaboration**.