Introduction to DevOps

By
Narasimha Rao T
Microsoft.Net FSD Trainer
tnrao.trainer@gmail.com

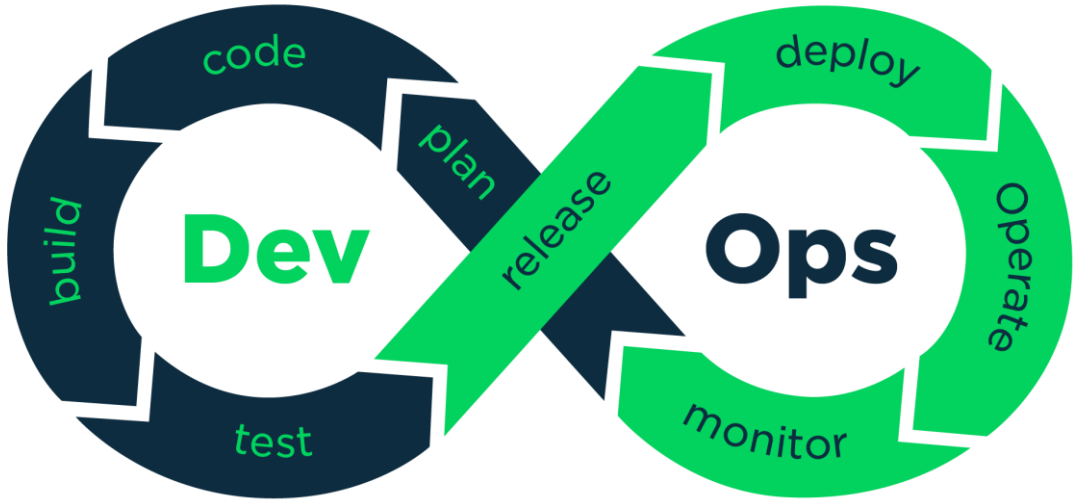
1. Introduction to DevOps

What is DevOps?

- **DevOps** is a set of practices that combines **software development (Dev)** and **IT operations (Ops)**.
- The goal is to shorten the development lifecycle and deliver high-quality software continuously.

Key Objectives

- Improve **collaboration** between development and operations.
- Automate and streamline the **software delivery pipeline**.
- Foster a culture of **shared responsibility**, **transparency**, and **rapid feedback**.



DevOps Lifecycle

- **Plan → Develop → Build → Test → Release → Deploy → Operate → Monitor**
- It’s a **cyclical** and **automated** flow rather than a linear process.

2. Continuous Integration (CI)

What is CI?

- **Continuous Integration** is the practice of automatically building and testing code every time a developer pushes changes to version control.

Benefits:

- Early bug detection.
- Improved code quality and integration.
- Encourages frequent commits.

Typical CI Flow:

1. Developer pushes code to Git repo.
2. CI tool (e.g., Jenkins, GitHub Actions) runs automated build.
3. Unit tests are executed.
4. Feedback is sent to developers.

Tools:

- **Jenkins**
- **CircleCI**
- **GitLab CI/CD**
- **Travis CI**

3. Continuous Deployment (CD)

What is Continuous Deployment?

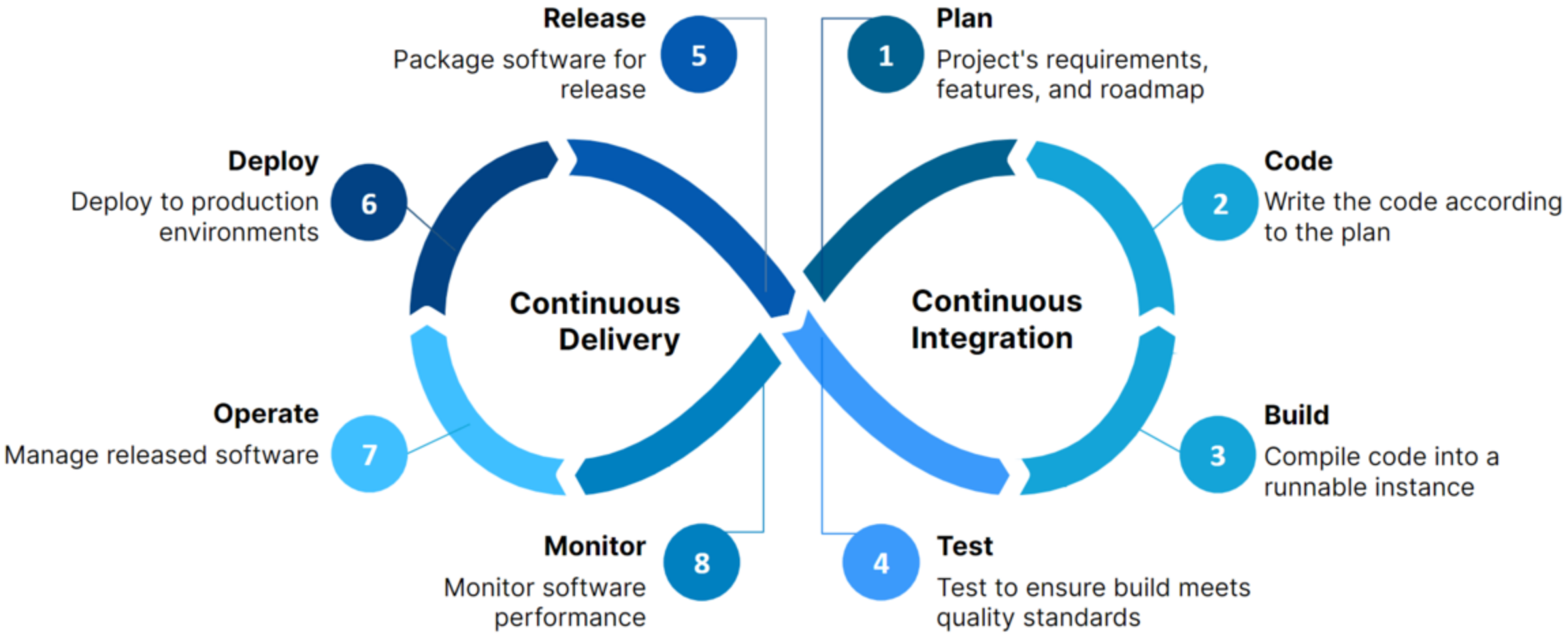
- Automatically deploying every code change that passes all tests into production **without human intervention**.

Difference from Delivery:

- In **Continuous Delivery**, deployment to production is **manual**.
- In **Continuous Deployment**, it is **automatic**.

Benefits:

- Faster time-to-market.
- Reduced human error.


Release
Package software for release
5
Plan
1
Project's requirements, features, and roadmap
Deploy
Deploy to production environments
6
Code
2
Write the code according to the plan
Continuous Delivery
Continuous Integration
Operate
Manage released software
7
Build
3
Compile code into a runnable instance
Monitor
Monitor software performance
8
4
Test
Test to ensure build meets quality standards

Tools:

- Spinnaker
- Argo CD
- Octopus Deploy

4. Continuous Delivery

What is Continuous Delivery?

- Practice of keeping codebase **deployable at all times**.
- Ensures that **tested and validated** code is ready for release on demand.

Practices:

- Automated testing.
- Environment parity (same code/configs from test to prod).
- Approval gates before production.

CD Pipeline Stages:

1. Build
2. Test
3. Integration
4. Staging
5. Release

5. Configuration Management

What is Configuration Management?

- Managing and maintaining consistency of software and infrastructure.
- Tracks and controls changes in software and system configurations.

Key Features:

- **Version-controlled configurations**.
- **Infrastructure as Code (IaC)**.
- Ensures **environment consistency** across dev, test, and prod.

Popular Tools:

- **Ansible** (agentless, YAML-based)
- **Chef** (Ruby-based)
- **Puppet** (declarative language)
- **Terraform** (for IaC, works well with cloud)

6. Microservices Architecture

What are Microservices?

- Architectural style where applications are composed of **small, independent services** that communicate via APIs.

Benefits:

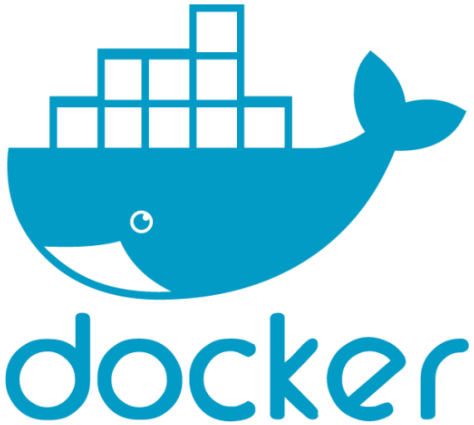
- Each service can be **developed, deployed, and scaled independently**.
- Enables **faster releases** and **fault isolation**.

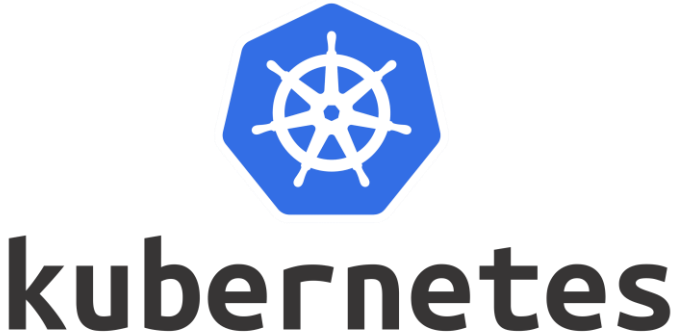
DevOps Alignment:

- Microservices + DevOps = modular, scalable, and fast-moving development.
- Containerization (e.g., Docker) is often used.

Supporting Tools:

- **Docker**
- **Kubernetes**
- **Istio** (service mesh)





7. Tools for Monitoring

Why Monitor?

- Visibility into application performance, availability, and errors.
- Helps in proactive issue resolution and decision-making.

Types of Monitoring:

- **Application Monitoring** (APM)
- **Infrastructure Monitoring**
- **Log Aggregation**
- **Alerting Systems**

Common Tools:

- **Prometheus + Grafana** – metrics and dashboards.
- **ELK Stack (Elasticsearch, Logstash, Kibana)** – log analysis.
- **Datadog**, **New Relic**, **Splunk** – all-in-one platforms.

8. DevOps and Cloud: Relationship

Why Cloud + DevOps?

- Cloud offers **on-demand resources**, enabling **scalability** and **automation**.
- DevOps needs the **agility and flexibility** provided by cloud environments.

DevOps Cloud Benefits:

- **Elastic infrastructure**: auto-scaling, fault tolerance.
- **IaC**: provision infrastructure using code.
- **Global deployments**: easier and faster.

Cloud Platforms Supporting DevOps:

- **AWS** (CodePipeline, CloudFormation, ECS)
- **Azure DevOps**
- **Google Cloud Build**, **GKE**

Synergies:

- DevOps drives **automation** → Cloud provides **platform**.
- DevOps enables **fast deployment** → Cloud enables **fast provisioning**.
- Together they promote **agility**, **efficiency**, and **resilience**.

Optional Visuals to Include:

1. **DevOps Lifecycle Diagram** (circular model).
2. **CI/CD Pipeline Flowchart**.
3. **Microservices vs Monolith** comparison diagram.
4. **DevOps + Cloud Integration Map**.
5. **Monitoring Stack Diagram** (e.g., ELK stack architecture).

Agile vs DevOps

Shared Goals

- Faster delivery of value to customers.
- Improved collaboration and communication.
- Focus on continuous improvement.

Key Differences

Feature	Agile	DevOps
Scope	Software development process	End-to-end software delivery (dev → ops)
Focus	Iterative development & feedback	Automation, integration, deployment
Team Structure	Dev teams (Product Owner, Devs, etc.)	Dev + Ops teams working as one
Cycle	Sprint-based releases	Continuous integration & delivery

How They Work Together

- **Agile** focuses on *how* software is **developed** (requirements → code).
- **DevOps** focuses on *how* software is **delivered and maintained** (build → deploy → operate).

Q & A