

Devops

ow

Waterfall Model

Sequential Phase-by-Phase Approach

Waterfall Flow

Requirements

|

Design

|

Implementation

|

Testing

|

Deployment

|

Maintenance

Key Characteristics

- ✓ Linear and sequential approach
- ✓ Each phase must be completed before next
- ✓ Comprehensive documentation
- ✓ Clear milestones and deliverables
- ✓ Limited customer involvement after requirements
- ✓ Changes are difficult and costly

Evolution: Waterfall Era

1970s-1990s

Waterfall Methodology

Sequential, phase-by-phase development

✓ Advantages

- Clear documentation
- Well-defined stages
- Easy to manage
- Predictable timelines

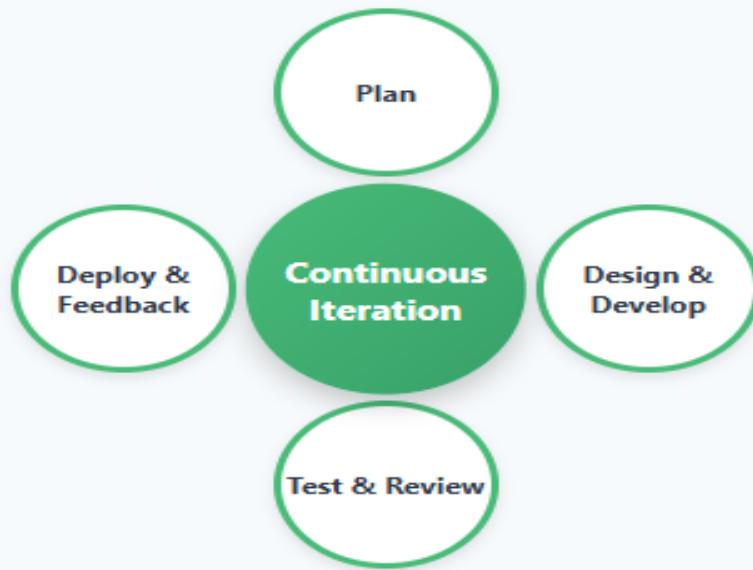
✗ Disadvantages

- Inflexible to changes
- Late testing phase
- Slow delivery
- High risk

Agile Methodology

Iterative and Incremental Approach

Agile Cycle



Core Values

- ✓ Individuals and interactions over processes
- ✓ Working software over documentation
- ✓ Customer collaboration over negotiation
- ✓ Responding to change over following plan
- ✓ Continuous delivery of value
- ✓ Regular feedback and adaptation

Evolution: Agile Revolution

2001-Present

Agile Methodology

Iterative development with continuous feedback

Agile Values

- ✓ Individuals over processes
- ✓ Customer collaboration
- ✓ Working software over docs
- ✓ Responding to change

Faster delivery achieved, but Dev-Ops silos remained

Waterfall vs Agile

Side-by-Side Comparison

Aspect	Waterfall	Agile
Approach	Sequential, linear	Iterative, incremental
Flexibility	Rigid, difficult to change	Highly flexible
Customer Involvement	Beginning and end only	Continuous throughout
Testing	After development phase	Continuous in each iteration
Delivery Time	Single delivery at end	Frequent small releases
Documentation	Comprehensive and detailed	Minimal, just enough
Team Size	Can be large	Usually small teams

Evolution: DevOps Emergence

2009-Present

DevOps Revolution

Unifying Development & Operations

Evolution Timeline

Waterfall

Months to Years

Agile

Weeks to Months

DevOps

Hours to Days

What is DevOps?

Definition

A set of practices, tools, and cultural philosophy that automates and integrates software development and IT operations.

Dev

Development

Plan, Code, Build, Test

Ops

Operations

Deploy, Monitor, Operate

What is DevOps? Principles

Core Foundations

1. Collaboration

Breaking silos between Dev and Ops

2. Automation

Automate repetitive pipeline tasks

3. Continuous Improvement

Iterative enhancement via feedback

4. Customer-Centric

Rapid value delivery to users

What is DevOps? Components

Three Pillars

Culture

Collaboration, Trust, Transparency

Tools

Jenkins, Docker, K8s, Terraform

Practices

CI/CD, IaC, Monitoring

Infinity Loop

Plan → Code → Build → Test → Release → Deploy → Operate → Monitor

Why DevOps?

Business Drivers

Traditional Problems

- ✗ Slow releases
- ✗ Frequent failures
- ✗ Team silos
- ✗ Manual errors

DevOps Solutions

- ✓ Daily deployments
- ✓ Reliable automation
- ✓ Unified teams
- ✓ Quality assurance

Why DevOps? Advantages

Strategic Benefits

⚡ Speed to Market

Beat competitors with faster releases

💡 Innovation

Experiment rapidly, learn quickly

🎯 Quality

Automated testing ensures reliability

😊 Satisfaction

Continuous value delivery

Why DevOps? Statistics

DORA Report

208x

More Deployments

106x

Faster Lead Time

7x

Lower Failure Rate

2604x

Faster Recovery

Benefits: Speed & Efficiency

Accelerated Delivery

Faster Time to Market

Deploy multiple times daily vs months

Automation

60-80% reduction in manual work

Reduced Lead Time

Minutes from commit to production

Benefits: Quality

Enhanced Excellence

✓ Continuous Testing

Catch bugs early

✓ Early Detection

Fix in dev, not prod

✓ Quick Recovery

Fast rollbacks

✓ Consistency

Uniform environments

60% Fewer Production Defects

Benefits: Culture

Organizational Impact

Collaboration

Shared responsibility, no silos

Productivity

Focus on innovation

Innovation

Safe experimentation

Satisfaction

Better work-life balance

DevOps Stages: Development

Left Side

1. PLAN

Define requirements, sprints

Tools: Jira, Azure Boards

2. CODE

Write and version control

Tools: Git, GitHub, GitLab

3. BUILD

Compile and create artifacts

Tools: Maven, Gradle, npm

4. TEST

Automated quality assurance

Tools: JUnit, Selenium

DevOps Stages: Operations

Right Side

5. RELEASE

Package for deployment

Tools: Jenkins, GitLab CI

6. DEPLOY

Automated to production

Tools: Kubernetes, Docker

7. OPERATE

Manage infrastructure

Tools: Terraform, AWS

8. MONITOR

Track and gather feedback

Tools: Prometheus, Grafana

DevOps Stages: Loop

Continuous Cycle

∞ Continuous Feedback

Each stage feeds into next

Development

Plan → Code → Build → Test

Creating Value

Operations

Release → Deploy → Operate → Monitor

Delivering Value

Monitoring drives new planning

DevOps Lifecycle: Overview

End-to-End

Complete Lifecycle

Concept to production and feedback

PLAN

Features, requirements

DEVELOP

Code, build, test

DELIVER

Release and deploy

OPERATE

Manage and optimize

DevOps Lifecycle: Practices

Key Enablers

CI

Frequent merges, automated builds

CD

Automated pipeline

IaC

Infrastructure as code

Monitoring

Real-time feedback

All practices in harmony

DevOps Lifecycle: Feedback

Continuous Learning

Data-Driven Improvement

Feedback drives optimization

Metrics

Deployment frequency, lead time, MTTR, failure rate

Sources

Tests, monitoring, users, retrospectives

Action

Refine, fix, improve continuously

Various Automation

Foundation of DevOps

Why Automation?

Speed, consistency, error reduction

Without

- Manual work
- Human errors
- Slow processes
- Inconsistent

With

- Automated pipelines
- Minimal errors
- Rapid delivery
- Predictable

Automation Types (Part 1)

Core Areas

1. Build Automation

Compile and package

Maven, Gradle, MSBuild

2. Test Automation

Unit, integration, E2E

Selenium, JUnit, Cypress

3. Deployment Automation

Automated releases

Jenkins, GitLab CI

4. Infrastructure Automation

Provision as code

Terraform, Ansible

Automation Types (Part 2)

Advanced Areas

5. Configuration Management

Server and app config

Ansible, Puppet, Chef

6. Monitoring Automation

Alerts and dashboards

Prometheus, Datadog

7. Security Automation

Automated security scanning

SonarQube, Snyk, OWASP

Overview of CI/CD

Continuous Integration & Delivery

CI/CD Pipeline

Automated software delivery from commit to production

CI

Continuous Integration

Merge code frequently with automated testing

CD

Continuous Delivery

Automated deployment to production

CI - Continuous Integration

Integration Process

What is CI?

Automatically integrate code changes frequently

1. Commit Code

Developers push code to repository

2. Automated Build

System automatically compiles code

3. Automated Tests

Run unit and integration tests

4. Feedback

Immediate notification on success/failure

CD - Continuous Delivery/Deployment

Delivery Process

What is CD?

Automated deployment pipeline to production

Continuous Delivery

Code always ready for deployment, manual approval needed

Continuous Deployment

Fully automated to production, no manual approval

Build → Test → Stage → Deploy → Monitor