Devops

DevOps Overview

DevOps is a process of improving the application delivery through these key practices:

Core Pillars:

1. **Automation** - Automating repetitive tasks like builds, deployments, infrastructure provisioning, and configuration management to reduce manual errors and speed up processes
2. **Quality** - Implementing automated testing, code reviews, and quality gates to ensure high standards throughout the development lifecycle
3. **Continuous Monitoring** - Real-time tracking of application performance, infrastructure health, and user experience to quickly identify and resolve issues
4. **Continuous Testing** - Running automated tests continuously throughout the development pipeline to catch bugs early and ensure code reliability

---

This diagram illustrates the **evolution from traditional to DevOps practices**:

Traditional Approach (10 Years Ago - Left Side)

The organization had **separate, siloed teams**:

- **Sys Admin** - Managed servers and infrastructure
- **BRE (Build/Release Engineer)** - Handled builds and releases
- **Server Admin** - Maintained servers

Each team worked independently with limited collaboration, leading to:

- Slow handoffs between teams
- Communication gaps
- Longer deployment cycles
- Potential conflicts and delays

Modern DevOps Approach (Right Side)

Now everything converges into a **unified workflow**:

Key Components:

1. **App Dev** - Developers write application code

2. **Pro (Production) Customer** - Production environment serving customers
3. **App Server** - Where the application runs
4. **Deploy** - Automated deployment process

The Integration:

All three traditional roles (Sys Admin, BRE, Server Admin) now work together, feeding into a **continuous pipeline** that:

- Runs CVS (version control) code
- Uses **vmware, openstack, xen** (virtualization/cloud platforms)
- Enables rapid deployment to production

The Result: Instead of isolated teams with weeks-long handoffs, you have an integrated process where code flows smoothly from development through deployment to customers, dramatically reducing delivery time and increasing collaboration.

What is SDLC?

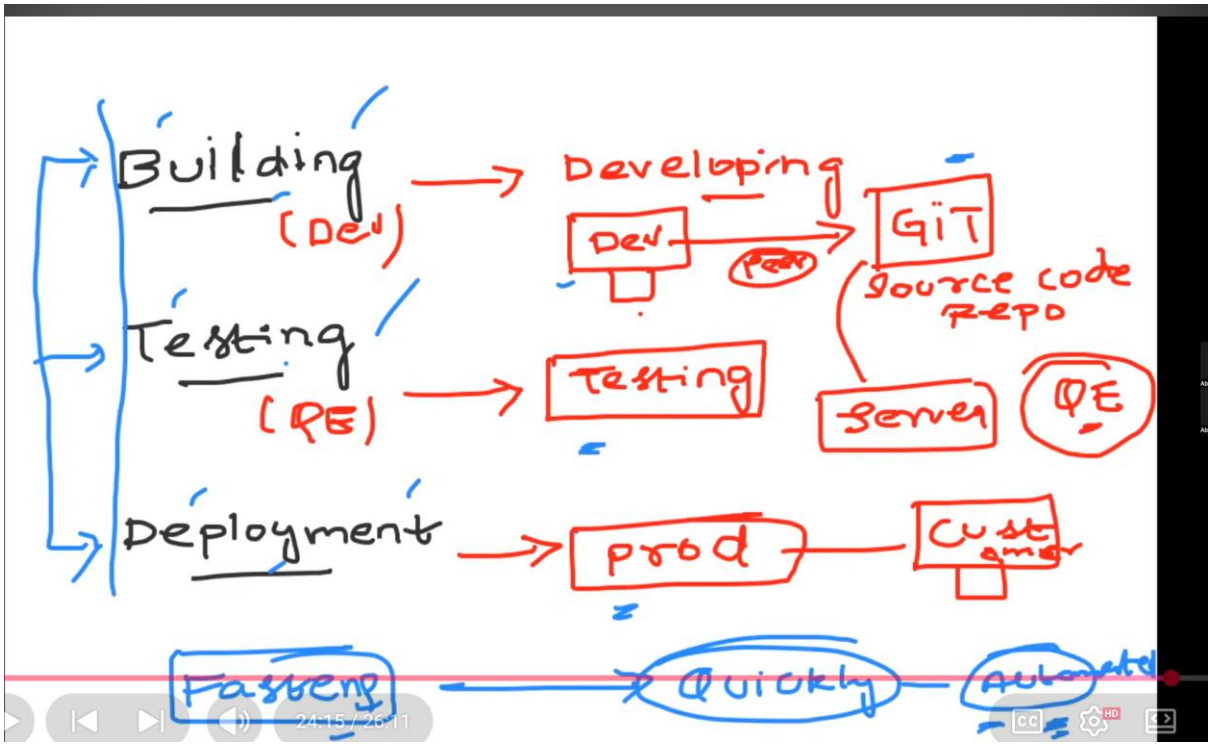
SDLC is the structured process of planning, creating, testing, and deploying software applications. It provides a framework for developing quality software efficiently.

The SDLC Cycle (Center Circle):

1. **Planning** - Define requirements (e.g., 6-12 versions, 1-4 men/women team size, KPIs, documentation)
2. **Defining** - Specify technical requirements (HLD - High Level Design, LCD - Low Level Design)
3. **Designing** - Create architecture and system design
4. **Building** - Write the actual code
5. **Testing** - Verify the application works correctly
6. **Deploy** - Release to production

DevOps Enhancements (Green annotations):

- **Requirements+** (at Planning) - Enhanced requirement gathering
- **Documentation** (at Defining) - Proper documentation practices
- **Automation** (at Deploy/Testing) - Automated deployments and tests



Left Side - SDLC Phases:

1. **Building (Dev)** - Development phase
2. **Testing (PE)** - Performance/Quality Engineering phase
3. **Deployment** - Release to production

Right Side - Team Responsibilities & Tools:

1. Developing (Building Phase)

- **Dev Team** writes code
- **Git** - Source code repository for version control
- Code flows from developers into Git

2. Testing Phase

- **Testing Team** validates the application
- **Server (QE)** - Quality Engineering/Testing environment
- Tests run on dedicated testing servers

3. Deployment Phase

- **Prod (Production)** - Live environment serving customers

- **Customer** - End users accessing the production application

Bottom Section - Key Concepts:

Three circled terms represent **automation approaches**:

- **Framework** - Testing frameworks for automated tests
- **Quickly** - Emphasis on speed and efficiency
- **Automated** - Automation of repetitive tasks

The DevOps Flow:

The diagram illustrates how code moves through environments: **Dev → Git → Testing/Server (QE) → Prod → Customer**

Each phase has dedicated teams and infrastructure, but they're all connected in a **continuous pipeline** that enables rapid, automated delivery from development to customers. The emphasis on "quickly" and "automated" at the bottom reinforces that DevOps accelerates this entire process through automation frameworks