DevOps Fundamentals - Version Control System (VCS)

[**🔍 1. Purpose** 1](#_Toc204938844)

[**📘 2. Theory** 1](#_Toc204938845)

[**🧰 3. Prerequisites** 2](#_Toc204938846)

[**🔧 4. Step-by-Step: Basic Git Workflow** 2](#_Toc204938847)

[**🔁 5. Common Git Commands** 2](#_Toc204938848)

[**🔗 6. DevOps Integration with Git** 2](#_Toc204938849)

[**📸 7. Snapshot: Git in DevOps Pipeline** 3](#_Toc204938850)

[**✅ 8. Summary** 3](#_Toc204938851)

**🔍 1. Purpose**

To understand how **Version Control Systems (VCS)**—especially **Git**—are essential in DevOps for **tracking changes**, **collaborating** on code, and **automating deployments** in CI/CD pipelines.

**📘 2. Theory**

**🔹 What is a Version Control System (VCS)?**

A **VCS** tracks changes to files over time so you can recall specific versions later, collaborate with others, and manage multiple branches of code development.

**Git** is the most widely used modern distributed version control system.

**🔹 Centralized vs Distributed VCS**

| **Type** | **Examples** | **Key Difference** |
| --- | --- | --- |
| Centralized | SVN, TFS | Single central server |
| Distributed | Git, Mercurial | Every user has full repo locally |

**🔹 Why Git in DevOps?**

| **✅ Benefit** | **💡 Description** |
| --- | --- |
| Branching & Merging | Develop features independently and merge cleanly |
| Collaboration | Multiple developers work on the same repo |
| History & Rollback | Audit changes, revert if needed |
| Automation Friendly | Trigger CI/CD workflows via commit or PR |

**🧰 3. Prerequisites**

* Git installed on your system (git --version)
* GitHub / Azure Repos / GitLab account
* Basic CLI knowledge or Git UI tool (e.g., GitHub Desktop, VS Code)

**🔧 4. Step-by-Step: Basic Git Workflow**

**✅ Step 1: Initialize a Git Repository**

git init

**✅ Step 2: Add Remote Repository**

git remote add origin https://github.com/yourname/repo.git

**✅ Step 3: Stage, Commit, and Push Changes**

git add .

git commit -m "Initial commit"

git push -u origin main

**🔁 5. Common Git Commands**

| **Action** | **Git Command** |
| --- | --- |
| Clone a repo | git clone <url> |
| Check status | git status |
| Create a new branch | git checkout -b feature-x |
| Merge a branch | git merge feature-x |
| Pull latest changes | git pull |
| View commit history | git log |
| Undo changes | git checkout -- <filename> |

**🔗 6. DevOps Integration with Git**

* Code commits can **trigger builds** in Azure Pipelines, Jenkins, GitHub Actions, etc.
* Git repositories support **branching strategies** (e.g., Git Flow, trunk-based).
* Pull requests enable **code review workflows**.

**📸 7. Snapshot: Git in DevOps Pipeline**

Developer → Commit Code to Git →

CI/CD Tool (Azure DevOps / GitHub Actions) →

Build / Test / Deploy →

Feedback to Developer

**✅ 8. Summary**

| **Key Point** |
| --- |
| **Git** is a distributed version control system that plays a critical role in DevOps by enabling **collaborative development**, **history tracking**, and **seamless integration with CI/CD** pipelines. |