**Git & GitHub Learning Path: Beginner to Expert**

**This syllabus is designed to take you from a complete beginner to an expert in Git (version control) and GitHub (collaboration platform). Each level builds on the previous one, ensuring a structured learning experience.**

**Level 1: Beginner (Fundamentals)**

**Week 1: Introduction to Version Control**

* **What is Git & why use it?**
* **Difference between Git & GitHub**
* **Installing Git (Windows, macOS, Linux)**
* **Basic Git configuration (git config)**

**Week 2: Basic Git Commands**

* **Creating a repository (git init, git clone)**
* **Tracking changes (git status, git add, git commit)**
* **Viewing commit history (git log)**
* **Undoing changes (git restore, git reset, git checkout)**

**Week 3: Introduction to GitHub**

* **Creating a GitHub account**
* **Creating & cloning repositories**
* **Pushing & pulling changes (git push, git pull)**
* **Understanding .gitignore**

**Week 4: Basic Collaboration**

* **Forking a repository**
* **Creating & merging Pull Requests (PRs)**
* **Resolving simple merge conflicts**

**Level 2: Intermediate (Workflows & Branching)**

**Week 5: Branching in Git**

* **Why use branches?**
* **Creating & switching branches (git branch, git checkout, git switch)**
* **Merging branches (git merge)**
* **Comparing branches (git diff)**

**Week 6: Advanced GitHub Features**

* **GitHub Issues & Projects**
* **GitHub Actions (Basic CI/CD)**
* **GitHub Wikis & Pages**
* **Using GitHub Desktop (Optional GUI tool)**

**Week 7: Git Workflows**

* **Centralized Workflow**
* **Feature Branch Workflow**
* **GitFlow Workflow**
* **Forking Workflow (Open Source Contribution)**

**Week 8: Rebasing & Stashing**

* **Rebasing vs. Merging (git rebase)**
* **Interactive rebase (git rebase -i)**
* **Stashing changes (git stash)**

**Level 3: Advanced (Efficiency & Troubleshooting)**

**Week 9: Advanced Git Commands**

* **Cherry-picking commits (git cherry-pick)**
* **Reflog & recovering lost commits (git reflog)**
* **Bisecting to find bugs (git bisect)**
* **Submodules & subtrees (git submodule)**

**Week 10: GitHub Advanced Features**

* **GitHub CLI (gh command-line tool)**
* **GitHub Codespaces (Cloud-based development)**
* **GitHub Copilot (AI-assisted coding)**
* **Managing GitHub Organizations**

**Week 11: Git Hooks & Automation**

* **Pre-commit & post-commit hooks**
* **Automating workflows with GitHub Actions**
* **Custom Git aliases for efficiency**

**Week 12: Open Source Contribution & Best Practices**

* **Finding & contributing to open-source projects**
* **Maintaining a professional GitHub profile**
* **Writing good commit messages & READMEs**
* **Security best practices (SSH keys, PATs, 2FA)**

**Level 4: Expert (Mastery & Real-World Scenarios)**

**Week 13: Git Internals (How Git Works)**

* **The .git directory structure**
* **Git objects (blobs, trees, commits, tags)**
* **How Git stores & compresses data**

**Week 14: Large Repositories & Performance**

* **Handling large files with Git LFS**
* **Shallow cloning (git clone --depth=1)**
* **Speeding up Git operations**

**Week 15: Troubleshooting & Debugging**

* **Fixing corrupted repositories**
* **Recovering from bad merges**
* **Handling complex conflicts**

**Week 16: Real-World Projects & Case Studies**

* **Simulating team workflows**
* **Reviewing famous Git disasters & fixes**
* **Final project: Managing a full-scale repo with CI/CD**

**Final Assessment**

* **Beginner: Complete a small project with basic Git commands & push to GitHub.**
* **Intermediate: Contribute to an open-source project via PRs.**
* **Advanced: Set up a CI/CD pipeline using GitHub Actions.**
* **Expert: Debug & optimize a complex Git repository.**

**Additional Resources**

* **Books: *Pro Git* (free online)**
* **Courses: GitHub Skills Lab, Udemy Git courses**
* **Practice: GitHub’s "First Contributions" tutorial**

**This structured approach ensures you gradually build expertise while applying concepts in real-world scenarios. 🚀**

**Would you like recommendations for hands-on projects at each level?**

**Grok-3**

**Git and GitHub Syllabus: Beginner to Expert**

**This syllabus is designed to take a beginner with no prior experience in version control to an expert level in Git and GitHub over approximately 12 weeks, assuming 5-10 hours of study and practice per week. The course combines theoretical understanding, hands-on practice, and real-world application to master Git and GitHub.**

**Week 1-2: Introduction to Version Control and Git Basics**

**Objective: Understand version control concepts and learn fundamental Git commands.**

**Topics**

* **What is version control? Why use Git?**
* **Installing Git (Windows, macOS, Linux)**
* **Configuring Git (git config)**
* **Initializing a repository (git init)**
* **Basic Git workflow: add, commit, status, log**
* **Understanding the staging area and commits**
* **Introduction to GitHub: Creating an account and exploring repositories**

**Activities**

* **Install Git and configure user settings**
* **Create a local Git repository and make initial commits**
* **Practice basic commands (add, commit, status, log)**
* **Create a GitHub account and explore a public repository**
* **Write a short summary of version control benefits**

**Resources**

* **Pro Git Book (Chapters 1-2)**
* **GitHub Learning Lab**
* **Video: Git Tutorial for Beginners**

**Week 3-4: Branching and Merging**

**Objective: Master branching, merging, and resolving conflicts.**

**Topics**

* **What are branches? (git branch)**
* **Creating and switching branches (git checkout, git switch)**
* **Merging branches (git merge)**
* **Handling merge conflicts**
* **GitHub: Creating a repository, pushing commits (git push), pulling changes (git pull)**
* **Introduction to .gitignore files**

**Activities**

* **Create and switch between branches in a local repository**
* **Merge branches and resolve a simple merge conflict**
* **Create a GitHub repository and push a local repo to it**
* **Practice .gitignore by excluding files (e.g., .DS\_Store, node\_modules)**
* **Collaborate with a partner on a small GitHub repo (push/pull)**

**Resources**

* **Pro Git Book (Chapter 3)**
* **GitHub Docs: About Pull Requests**
* **Interactive: Learn Git Branching**

**Week 5-6: Collaboration with GitHub**

**Objective: Learn collaborative workflows using GitHub features.**

**Topics**

* **Forking repositories**
* **Pull requests (PRs): Creating, reviewing, and merging**
* **Cloning vs. forking**
* **GitHub Issues: Creating and managing issues**
* **Basic GitHub Actions for CI/CD**
* **Rebasing vs. merging (git rebase)**

**Activities**

* **Fork a public repository and submit a pull request**
* **Review and merge a pull request in a team setting**
* **Create and resolve an issue in a GitHub repository**
* **Set up a simple GitHub Action (e.g., linting a codebase)**
* **Practice rebasing to clean up commit history**

**Resources**

* **GitHub Docs: Collaborating with Pull Requests**
* **Pro Git Book (Chapter 5)**
* **GitHub Actions Quickstart**

**Week 7-8: Advanced Git Techniques**

**Objective: Gain proficiency in advanced Git commands and workflows.**

**Topics**

* **Stashing changes (git stash)**
* **Cherry-picking commits (git cherry-pick)**
* **Interactive rebasing (git rebase -i)**
* **Bisecting to find bugs (git bisect)**
* **Working with tags (git tag)**
* **Undoing changes: reset, revert, reflog**
* **Git hooks for automation**

**Activities**

* **Use git stash to temporarily save changes**
* **Perform an interactive rebase to squash commits**
* **Use git bisect to find a bug in a sample project**
* **Create and push a tagged release to GitHub**
* **Recover a deleted commit using git reflog**
* **Write a simple Git hook (e.g., pre-commit linting)**

**Resources**

* **Pro Git Book (Chapters 6-7)**
* **Atlassian Git Tutorials**
* **Video: Advanced Git Tutorial**

**Week 9-10: Git Workflows and Large-Scale Collaboration**

**Objective: Understand professional Git workflows and contribute to open-source projects.**

**Topics**

* **Common Git workflows: Gitflow, GitHub Flow, Trunk-Based Development**
* **Managing large repositories**
* **Code reviews and best practices**
* **Contributing to open-source projects**
* **Git submodules and subtrees**
* **GitHub Pages for hosting static sites**

**Activities**

* **Implement Gitflow in a sample project**
* **Contribute a small change to an open-source project (e.g., documentation fix)**
* **Set up a GitHub Pages site from a repository**
* **Add a submodule to a project and update it**
* **Participate in a mock code review with peers**

**Resources**

* **Gitflow Workflow**
* **GitHub Docs: Contributing to Open Source**
* **GitHub Pages Guide**

**Week 11-12: Expert-Level Git and GitHub**

**Objective: Master advanced GitHub features and optimize workflows for teams.**

**Topics**

* **Advanced GitHub Actions: Custom workflows, matrix builds**
* **Git LFS (Large File Storage)**
* **GitHub CLI (gh tool)**
* **Automating releases with semantic versioning**
* **Managing team permissions and branch protection rules**
* **Debugging complex Git issues**
* **Migrating repositories (e.g., from Bitbucket to GitHub)**

**Activities**

* **Create a custom GitHub Action workflow for automated testing**
* **Use Git LFS to manage large files in a repository**
* **Automate a release process using GitHub CLI**
* **Set up branch protection rules for a team repository**
* **Migrate a sample repository to GitHub**
* **Debug a complex merge conflict in a team project**

**Resources**

* **GitHub Actions Documentation**
* **Git LFS Tutorial**
* **GitHub CLI Manual**
* **Pro Git Book (Chapter 10)**

**Capstone Project**

**Objective: Apply all learned skills in a comprehensive project.**

**Project Description**

* **Create a collaborative open-source project on GitHub (e.g., a simple web app, CLI tool, or documentation site).**
* **Implement a professional Git workflow (e.g., GitHub Flow).**
* **Use GitHub Issues, pull requests, and code reviews.**
* **Set up GitHub Actions for automated testing and deployment.**
* **Publish the project on GitHub Pages or release it with tags.**
* **Write a detailed README and contribute to another team’s project.**

**Deliverables**

* **A fully functional GitHub repository with all features implemented**
* **Documentation of the Git workflow used**
* **A blog post or video explaining the project and lessons learned**

**Assessment and Progress Tracking**

* **Weekly Quizzes: Test understanding of Git commands and GitHub features**
* **Hands-On Labs: Complete exercises for each topic (e.g., merging, rebasing)**
* **Project Milestones: Submit progress on the capstone project**
* **Peer Reviews: Participate in code reviews for collaborative tasks**
* **Final Presentation: Showcase the capstone project and explain the Git workflow**

**Recommended Tools**

* **Git: Command-line interface**
* **GitHub Desktop: For GUI-based Git operations**
* **VS Code: For editing code and Git integration**
* **GitKraken: For visualizing Git history**
* **GitHub CLI: For advanced GitHub automation**

**Learning Tips**

* **Practice daily with small Git repositories to build muscle memory.**
* **Join open-source communities on GitHub to observe real-world workflows.**
* **Use interactive tools like Learn Git Branching to reinforce concepts.**
* **Document your learning journey in a personal GitHub repository.**

**By the end of this syllabus, learners will be able to confidently use Git and GitHub in professional settings, contribute to open-source projects, and manage complex team workflows.**

**Git & GitHub Mastery Syllabus**

**👶 Beginner Level (Week 1-2)**

* **Introduction to Version Control and Git**
* **Installing Git and Initial Setup**
* **Understanding Repositories (Local vs. Remote)**
* **Basic Git Commands (git init, git clone, git status, git add, git commit, git log)**
* **Working with Branches (git branch, git checkout, git merge)**
* **Introduction to GitHub**
* **Creating a GitHub account and repositories**
* **Making commits and pushing changes (git push)**

**🚀 Intermediate Level (Week 3-6)**

* **Cloning repositories and pulling changes (git pull, git fetch)**
* **Understanding merge conflicts and resolving them**
* **Working with remote repositories (git remote, git origin, git upstream)**
* **GitHub workflows: Issues, Forking, Pull Requests**
* **Collaborative workflows (GitHub Teams & Organizations)**
* **Managing tags and releases (git tag)**
* **Using .gitignore for project configurations**
* **Interactive rebasing and committing best practices (git rebase, git commit --amend)**

**🧑‍🎓 Advanced Level (Week 7-10)**

* **Git Hooks and automation**
* **Understanding Git Internals (SHA-1 hashes, Git objects, DAG structure)**
* **Rewriting commit history (git reflog, git reset, git revert)**
* **Using advanced Git log commands (git log --oneline, git log --graph)**
* **Using git bisect for debugging**
* **Advanced branching strategies (GitFlow, Trunk-based development)**
* **Stashing and patching workflows (git stash, git cherry-pick)**
* **Submodules and managing large projects (git submodule)**
* **GitHub Actions for automation and CI/CD**
* **Security and best practices (SSH keys, Token authentication)**

**🎓 Expert Level (Week 11-15)**

* **Understanding Git Worktrees (git worktree)**
* **Writing custom Git aliases**
* **Handling large-scale repositories efficiently**
* **GitHub API & automation scripting**
* **Optimizing Git configurations for productivity (git config)**
* **Deep dive into Git LFS (Large File Storage)**
* **Scaling GitHub workflows in enterprises**
* **Contributing to open-source projects professionally**
* **Advanced troubleshooting techniques**

**Phase 1: Foundations (Weeks 1-4)**

**Week 1: Introduction to DevOps & CI/CD**

* What is DevOps? (Culture, Practices, Benefits)
* Understanding CI/CD (Continuous Integration vs. Continuous Deployment)
* Traditional vs. DevOps Workflow
* Popular CI/CD Tools (Jenkins, GitHub Actions, GitLab CI, CircleCI)
* Setting up a Python Development Environment

**Week 2:** **Version Control with Git & GitHub**

* Git Basics (Repositories, Commits, Branches, Merging)
* Git Workflows (Feature Branch, Git Flow, GitHub Flow)
* Collaborating on GitHub (Pull Requests, Code Reviews)
* Hands-on: Managing a Python project with Git

**Week 3: Python for Automation**

* Python Basics (Syntax, Data Structures, Functions)
* File Handling & OS Operations in Python
* Automating Tasks with Python Scripts
* Working with APIs (REST, GitHub API)
* Hands-on: Writing Python scripts for file operations & API calls

**Week 4: Linux & Shell Scripting Basics**

* Basic Linux Commands (File System, Permissions, Processes)
* Writing Shell Scripts for Automation
* Environment Variables & Configuration
* Hands-on: Automating server tasks with Bash & Python

**Phase 2: CI/CD Pipelines (Weeks 5-8)**

**Week 5: Introduction to CI/CD Pipelines**

* What is a Pipeline? (Stages, Jobs, Workflows)
* YAML Syntax for Pipeline Definitions
* Setting up a Simple CI Pipeline (GitHub Actions / GitLab CI)
* Hands-on: Running a Python test suite in CI

**Week 6: Automated Testing in Python**

* Unit Testing (Pytest, Unittest)
* Integration Testing & Mocking
* Code Coverage (Coverage.py)
* Hands-on: Writing & Running Tests in CI

**Week 7: Building & Packaging Python Applications**

* Python Packaging (setuptools, Poetry, pip)
* Building Docker Containers for Python Apps
* Hands-on:
  + Creating a Python package
  + Dockerizing a Python app

**Week 8: Deployment Strategies**

* Blue-Green Deployment
* Canary Releases
* Rolling Deployments
* Hands-on: Deploying a Python Flask app using CI/CD

**Phase 3: Advanced CI/CD & DevOps (Weeks 9-12)**

**Week 9: Infrastructure as Code (IaC)**

* Introduction to Terraform & Ansible
* Automating Cloud Deployments (AWS/GCP/Azure)
* Hands-on: Deploying a Python app using Terraform

**Week 10: Monitoring & Logging**

* Logging in Python (Logging Module, ELK Stack)
* Monitoring with Prometheus & Grafana
* Hands-on: Setting up alerts for a Python app

**Week 11: Security in CI/CD (DevSecOps)**

* Static Code Analysis (Bandit, SonarQube)
* Dependency Scanning (Safety, Dependabot)
* Secrets Management (Vault, AWS Secrets Manager)
* Hands-on: Securing a Python CI/CD pipeline

**Week 12: Advanced Topics & Real-World Projects**

* Multi-stage Pipelines (Dev, Staging, Production)
* Feature Flags & A/B Testing
* Serverless CI/CD (AWS Lambda, GitHub Actions)
* Final Project:
  + Build a full CI/CD pipeline for a Python web app
  + Include testing, Docker, deployment, and monitoring

**Phase 4: Expert Level (Optional Advanced Topics)**

* Kubernetes & Helm for CI/CD
* Scaling CI/CD Pipelines (Parallel Jobs, Caching)
* GitOps (ArgoCD, Flux)
* Chaos Engineering in CI/CD (Gremlin, Chaos Monkey)

**Learning Resources**

* **Books:**
  + ***"The Phoenix Project"* (Gene Kim) – DevOps Culture**
  + ***"Continuous Delivery"* (Jez Humble) – CI/CD Best Practices**
* **Online Courses:**
  + **GitHub Actions/ GitLab CI Documentation**
  + **Udemy: "DevOps with Python"**
* **Tools to Master:**
  + **Git, GitHub Actions, Docker, Terraform, Kubernetes, Pytest**

**Final Goal**

**By the end of this syllabus, you will:  
✅ Build and manage CI/CD pipelines for Python applications  
✅ Automate testing, building, and deployment  
✅ Implement Infrastructure as Code (IaC)  
✅ Apply DevSecOps best practices  
✅ Deploy scalable and resilient applications**

**Would you like any modifications or additional topics? 🚀**

**Syllabus: CI/CD & DevOps with Python (Hands-on Approach)**

**📖 Module 1: Introduction to DevOps & CI/CD**

✅ Overview of DevOps & CI/CD ✅ Key Principles of Continuous Integration, Delivery, and Deployment ✅ Understanding Pipelines in DevOps ✅ Setting Up Python Development Environment

**🚀 Module 2: Version Control with Git & GitHub**

✅ **Hands-on:** Installing Git & Setting Up GitHub ✅ Git Workflow: Cloning, Branching, Merging, and Pull Requests ✅ **Python Example:** Managing a Flask-based Web App with Git ✅ Automating GitHub Actions for Python Projects

**⚙️ Module 3: Containerization with Docker**

✅ Introduction to Containers & Docker ✅ **Hands-on:** Docker Installation & Basic Commands ✅ Writing Dockerfiles for Python Applications ✅ Building & Running Python Apps in Containers ✅ Using Docker Compose for Multi-Service Applications

**☁️ Module 4: CI/CD with GitHub Actions & Jenkins**

✅ **Hands-on:** Setting Up GitHub Actions for Python CI ✅ Writing Custom Workflows for Python Unit Tests ✅ **Jenkins Integration:** Automating Python Builds & Tests ✅ Deploying Python Apps with CI/CD Pipelines

**📈 Module 5: Infrastructure as Code (IaC) with Terraform & Ansible**

✅ Introduction to Infrastructure as Code (IaC) ✅ **Hands-on:** Deploying Python Apps on AWS using Terraform ✅ Using Ansible to Automate Python App Configurations

**🔐 Module 6: Security & Monitoring in DevOps**

✅ DevSecOps Best Practices for Python Apps ✅ **Hands-on:** Static Code Analysis & Security Scanning ✅ Logging & Monitoring Python Apps with Prometheus & Grafana

**🚀 Module 7: Cloud Deployment & Scaling**

✅ Deploying Python Applications on AWS/GCP/Azure ✅ **Hands-on:** Using Kubernetes for Python App Deployment ✅ Scaling Python Apps Using Load Balancing Technique