CodeWizard

visitors 42276 nirgeier

NIRG@CODEWIZARD.CO.IL / 054 8122310

Git Advanced topics

	Level	Duration
	Advanced	2 days

On one hand Git is a pretty simple tool, on the other hand to manage git on organization level...
This is another story.

Git Advanced course main objectives

- The main objective of this course is to **learn how to manage Git**,
- In this course the participants will learn what are the essentials and critical aspects tha need attention when you working in teams/ organizations
- The course will teach best practices and recommendation for managing Git with **AzureDevOps**
- At the end fo this course the participants should know how to avoid and resolve critical and import issues like:
  - Avoiding conflicts and how to resolve them effectively
  - Block deleting GIT history / branches and how to fix those issues if someone did it by mistake
  - Advanced topics like working with Pull Requests and verifying the bew content doesn't break the existing branch.
  - What are git hooks and when to use them
  - Managing multiple fixes on multiple branches simultaneously
  - Best practices and policies for different teams who are using Git

Audience and prerequisites

- This course assume the participants has been working with Git and have prior knowledge before attending this course.
- The course is for `Developers` / `DevOps` / `Git Administrators`

Main Goals

- How to configure git for your organization (configuration, hooks, aliases, gitconfig)
- What are branches and how to use them efficiently to improve your team productivity

- How do merges work (`ff` , `no-ff` , `rebase`) and when to choose each merge strategy
- GitFlow and why it is recommended to use it
- Git hooks, what are git hooks, how to create & use them, and why/when to use them to enforce policies and to improve productivity
- Why using Pull Request is important and how to use it correctly.
- CI/CD for Git repositories for improving productivity
- Tips & Tricks and Beyond

Session	Content
Intro	
	◆ Git & Azure DevOps - overview & configuration
	◆ Azure Repos Setting
	◆ Azure Repos Policies
	◆ Azure Repos Security
	◆ Azure Repos Advanced Security
Project Management	
	◆ Git & Project management
	◆ Integration with AzureDevOps/Jira or any other management tool
	◆ How to add visibility for managers
	◆ Integration with AzureDevOps/3rd part tools like Jenkins
	◆ How to link git commits/branches/pr/builds to AzureDevOps
Git Administrations	
	◆ Managing Git (Cross teams / Multiple teams)
	◆ Define the suitable branching model for your team / organization
	◆ Managing branches
	◆ Managing changes (single/multiple branches like hotfix)
	◆ Define git hooks
GitFlow	
	◆ What is GitFlow
	◆ Deep understanding of the GitFlow model
	◆ Why should we use it
	◆ How can this model improve out productivity
	◆ What are the different branches in the model
	◆ How can we use the GitFlow scripts for automating the flow
	◆ Best practice for GitFlow
Advanced Topics	
	◆ Advanced git features focusing on commands / features for administrators
	◆ **assume-unchanged** Ability to change files locally without exposing changes to git
	◆ **auto-completion / autocorrect**

	◆ **bisect** Search git history for code changes (finding bugs, wrong merges ...)
	◆ **cherry-pick** Ability to pick specific commits to different repositories or branches
	◆ **smudge / clean** One of the most **important** features of Git
	◆ **merge** Deep understating of git merge and how to configure merges across teams to avoid conflicts and more
	◆ **git LFS** Manage binaries and big files with GIT
	◆ **hooks** Manage hooks to enforce policies, improve productivity and integration with other tools
	◆ **notes** What are git notes, when to use them and why
	◆ **reflog** One of the most **important** commands of Git which allow you to fix many common problems fast and easily
	◆ **rerere** Letting git resolve conflict you **already resolved** automatically
	◆ **squash** What is `squash`, when and how to use it to keep history organized and clean
	◆ **stash**
	◆ **submodule / subtree** Working with dependencies and multiple git projects
	◆ **tags**
	◆ **interactive rebase** Rewriting git history
	◆ **partial add** Adding partial file content and not the whole file
	◆ **worktree** The most efficient way of **truly** working with multiple branches
	◆ **Detached HEAD** What is it, why we got it, how to "fix" it
	◆ **reset/revert** How to use `reset` & `revert` to fix problems
Pull Request	
	◆ What is pull request

	🔶 Why should we always use it
	🔶 What does the pull request include
	🔶 CI/CD with pull request
	🔶 Code review
	🔶 Approvers
	🔶 Verifying code before merging it
Best practices & Tips	
	🔷 Managing binaries / artifacts (Git LFS)
	🔷 Managing sensitive information with AzureDevOps
	🔷 Managing sensitive information with Git (Secrets, certificates, tokens etc)
	🔷 Searching history (for deleted files, for changes to specific lines, blocks of code)
	🔷 Generating release notes
	🔷 Advanced Smudge/ Clean demos and abilities
	🔷 Advanced branching management / policies
	🔷 Advanced log (flags, grep ...)
	🔷 Remove content from git (in case some one committed unwanted content like sensitive information)
	🔷 Managing hooks globally for teams/ organization
	🔷 Repository management (clean old branches, find in which branches contains certain commits)
	🔷 Using `worktree` to improve productivity
	🔷 Enforcing git message structure (ex: must have link to ticket system)
	🔷 More... `git show`, `git whatchanged`

Back to courses list
