This site uses cookies to provide you with a better browsing experience. To learn more about the different cookies we're using, please see our privacy policy. Accept Decline Cloud Platform Coverage Why Varonis? Company Partners Resources Contact us Blog / PowerShell Git Branching and Merging: A Step-By-Step Guide Jeff Brown 6 min read Published May 17, 2021 Last updated June 27, 2022 Contents What is Git Branching? How to Create a Branch in Git Merging Branches Git Branching: Frequently Asked Questions Closing In previous articles, you learned â€œHow to Revert a Commit in Gitâ€ (a PowerShell Git tutorial) and â€œHow to Merge in Git: Remote and Local Git Repositories Tutorial.â€ You can also use Git to create branches in your project. Git branching allows multiple developers to work on a project by modifying the working codebase. In this article, you will learn more about Git branching, multiple ways to create branches, and how to merge these branches to a local or remote repository. Table of Contents What is Git Branching Branch Naming Strategies How to Create a Branch in Git Merging Branches FAQs Get Started: To follow along with this Git tutorial and learn how to create branches, you will need: The PowerShell Git client installed on your system (download and installation guide) A free GitHub account Connect to GitHub with SSH What is Git Branching? Git branching allows developers to diverge from the production version of code to fix a bug or add a feature. Developers create branches to work with a copy of the code without modifying the existing version. You create branches to isolate your code changes, which you test before merging to the main branch (more on this later). There is nothing special about the main branch. It is the first branch made when you initialize a Git repository using the git initcommand. When you create a commit, Git identifies that snapshot of files with a unique SHA-1 hash. When you initially create a branch, Git creates a new pointer to the same commit the main branch is currently on. The diagram below shows both branches have the same snapshot of code at this point. As you create commits in the new branch, Git creates new pointers to track the changes. The latest commits are now ahead of the main branch commits. As you continue to make commits, each branch keeps track of its version of files. Git knows which branch you have checked out by using a special pointer called HEAD. When you create a new branch, Git doesnâ€™t immediately change the HEAD pointer to the new branch. Youâ€™ll see HEAD in the tutorial when you create branches and view the commit log. This branching function is what makes Git really powerful. Multiple people create separate branches to work on their code and merge their changes into the main branch. Branches are meant to be temporary and should be deleted when work is completed. git push security main Can you 100% tell me that everyone with access to your repos has the correct rights? If not, ask Varonis for help now with tracking usage across source control, SaaS apps, and on-prem. First Name* Last Name* Email* Please note that we will use your contact details to provide you with marketing communications from Varonis that we think may be of interest to you. You can unsubscribe from these communications at any time by clicking a link at the bottom of each email. For more information on our privacy practices, and how we're committed to protecting your information, please see our privacy policy. Branch Naming Strategies Branch names can be anything youâ€™d like. However, your organization or project may have standards outlined for branch naming. For example, naming the branch based on the person responsible for working on the branch and a description or work item: username/description username/workitem You can name a branch to indicate the branchâ€™s function, like a feature, bug fix, or hotfix: bugfix/description feature/feature-name hotfix/description Another branching strategy is having branches dedicated to the different development cycles, like feature or hotfix. As work items come up, you create a branch for that item from its respective branch. Yes, you can create branches from branches! Check out Option 4 below for an example. How to Create a Branch in Git Enough theory, letâ€™s create some branches! These examples will be using PowerShell 7 on a Windows 10 system; however, you can use any terminal that supports Git commands. Option 1: Creating a Branch To create a branch, use the git branchcommand followed by the name of the branch. After making the branch, usegit branchagain to view available branches. Notice that creating a branch this way does not automatically switch to the new branch. Git uses an asterisk and a different colored font to identify which branch is active. This designation represents the HEAD pointer showing which branch is active. git branch git branch Option 2: Creating a Branch using Checkout If you want to create a branch and checkout the branch simultaneously, use the git checkoutcommand. The switch -b specifies the name of the branch. Note that after command completion, Git has moved HEAD to the new branch. git checkout -b git branch Option 3: Creating a Branch from a Commit You can create a branch from a previous commit on an existing branch. Remember, a commit is just a snapshot in time of the files in a repository. You create a branch from a commit if you want to work on a specific snapshot of the files. Before creating the branch, you need the SHA-1 identifier of the commit. To find the identifier, use the git logcommand to view previous commits. Each commit will have a complete SHA-1 hash as the identifier. However, you only need the first few characters to identify the commit. Next, use the same git branchcommand from Option 1 but append the commit identifier at the end. This example is using 40b4d7 from the second commit as the identifier. Note the HEAD designator is on the main branch, which is the active branch. The other branches jeff/feature1 and jeff/feature2 point to the same commit when you created them earlier. Both point to the same snapshot as each branch has not had additional commits made to each one since creation. git log git branch Option 4: Creating a Branch from Another Branch If you use branches dedicated to hotfixes or features, you create branches from these other branches to work on the item. Creating a branch from another branch is no different from creating from the main branch. You just need to specify the name of the other branch as the starting point. This example shows creating the feature4 branch from the develop branch. git checkout -b feature4 develop Option 5: Download Branch from Remote Repository While you have a local copy of a repository to work with, so do other developers. These developers will have branches they are working on, and they can push their branches to a remote repository. Along the way, you may need to work on another branch that isnâ€™t local on your system. You can pull or download specific branches from a remote repository to use on your system. In a central repository hosted in GitHub, notice the branches available are the same ones on the local system (main, feature1, feature2, and hotfix1). However, another developer named Maggie has a branch for hotfix2 that is not on the local system. Maggie requests your assistance working on a hotfix, so you need to download this branch to your system. To retrieve the branch from the remote repository, use git pullagainst the origin and specify the name of the branch. If you check available local branches, the new branch doesnâ€™t appear automatically. However, you can check out the branch and begin working on this new branch. git pull origin git branch git checkout git branch Merging Branches Once youâ€™ve completed work on your branch, it is time to merge it into the main branch. Merging takes your branch changes and implements them into the main branch. Depending on the commit history, Git performs merges two ways: fast-forward and three-way merge. Letâ€™s examine both of these based on the branches and commit history in the following diagram. When you merge the hotfix branch into the main branch, Git will move the main branch pointer forward to commit nr7jk. Git does this because the hotfix branch shares a direct ancestor commit with the main branch and is directly ahead of its commit. This commit is a fast-forward merge. Once you merge the hotfix branch, continue working on the feature1 branch. As you continue making commits on the feature1 branch, the commit history diverges. Git is unable to move the pointer to the latest commit like in a fast-forward commit. To bring the feature1 branch into the main branch, Git performs a three-way merge. Git takes a snapshot of three different commits to create a new one: The common commit both branches share (a90hb) The latest commit of the branch (az84f) The commit of the branch to merge into (nr7jk) Merging Branches in a Local Repository To merge branches locally, use git checkoutto switch to the branch you want to merge into.

This branch is typically the main branch. Next, use git mergeand specify the name of the other branch to bring into this branch. This example merges the jeff/feature1 branch into the main branch. Note that this is a fast-forward merge. git checkout main git merge jeff/feature1 Â Work continues on the main and other branches, so they no longer share a common commit history. Now a developer wants to merge the jeff/feature2 branch into the main branch. Instead, Git performs a three-way (or recursive) merge commit. git checkout main git merge jeff/feature2 Merging Branches to Remote Repository If you create a branch in your local repository, the remote repository is not aware of the branchâ€™s existence. Before you can push the branch code in the remote repository, you set the remote repository as the upstream branch using the git pushcommand. This command simultaneously sets the upstream branch and pushes the branch contents to the remote repository. git push --set-upstream origin Merging Main into a Branch While you are working on your branch, other developers may update the main branch with their branch. This action means your branch is now out of date of the main branch and missing content. You can merge the main branch into your branch by checking out your branch and using the same git mergeÂ command. git checkout git merge main Git Branching: Frequently Asked Questions What is creating a branch in Git? Creating a branch takes a snapshot of the existing code so you can work on it independently of the main branch. How do I create a new branch in Git? Use the git branch command and specify the branch name, e.g., git branch feature1. How can I view which branch is active? Use git branch to view available branches and which one is active. Git typically designates the active branch with an asterisk and a different colored font. How can I switch to working on another branch? Use git checkout and the name of the branch to make it active. Can I create a branch inside another branch? Yes, specify the name of the branch to base the new branch, e.g., git branch feature-bug feature1. My branch only exists locally. How can I add my branch to my remote Git repository? Yes, use the git push command to set the upstream branch, e.g., git push â€“set-upstream origin . How can I update the main branch with my branchâ€™s changes? Switch to the main branch using the git checkout command, then merge the branch using the git merge command along with the branch name. How can I delete a branch? When you are done with a branch, delete it using the git branch command and the -d switch, e.g., git branch -d feature1. Closing Git branching is a powerful feature that allows teams to work on the code independently of each other. Knowing how to create, name, and merge branches are essential skills for any developer, systems administrator, or DevOps engineer. What you should do now Below are three ways we can help you begin your journey to reducing data risk at your company: Schedule a demo session with us, where we can show you around, answer your questions, and help you see if Varonis is right for you. Download our free report and learn the risks associated with SaaS data exposure. Share this blog post with someone you know who'd enjoy reading it. Share it with them via email, LinkedIn, Twitter, Reddit, or Facebook. Jeff Brown Jeff Brown is a cloud engineer specializing in Microsoft technologies such as Office 365, Teams, Azure and PowerShell. You can find more of his content at https://jeffbrown.tech. Try Varonis free. Get a detailed data risk report based on your companyâ€™s data. Deploys in minutes. Get started View sample Keep reading How to Install and Import Active Directory PowerShell Module Jeff Brown November 3, 2021 The Active Directory PowerShell module is a powerful tool for managing Active Directory. Learn how to install and import the module in this detailed tutorial! Threat Update 49 â€“ SeriousSAM & Black Hat 2021 Kilian Englert July 30, 2021 Cybersecurity folks find themselves in a â€œZero-Dazeâ€ as they get hit with another new 0-day attack, called SeriousSAM, that allows attackers to get access to the Windows Security Account Manager (SAM) file containing hashed account passwords from a system. Threat Update 45 â€“ Ransomware Early Warning: AD Attacks Kilian Englert July 2, 2021 Attackers leverage a number of techniques, but two of the most common are password spray attacks and kerberoasting. Join Kilian and Kyle Roth from the Varonis Incident Response team as they discuss how and why attackers leverage each technique and look at a real-life example of each type of attack from one of our attack lab scenarios. Platform Protection packages Microsoft 365 & Entra ID SaaS & IaaS Windows & NAS Products Overview DatAdvantage Automation Engine Data Classification Engine Data Classification Labels Policy Pack DatAnswers DatAlert Edge Data Transport Engine DataPrivilege DatAdvantage Cloud Data Classification Cloud Solutions By use case Cloud data protection Data discovery & classification Compliance management Data loss prevention Data activity auditing DSPM Least privilege automation Insider risk management Proactive incident response Ransomware prevention SSPM Zero Trust By industry Finance Healthcare Federal government Education Manufacturing State & local government Integrations Microsoft 365 On-prem data & apps Cloud data, SaaS, & IaaS Directory services NAS Network devices Third-party apps Why Varonis? Case studies Operational plan Industry recognition Customer success IR & forensics team Measurable ROI Why Varonis SaaS Company About Varonis Careers Investor relations Press Corporate responsibility Trust & security Brand Partners Partner program Partner locator Partner portal Service providers Technology partners Buy on AWS marketplace Buy on Azure marketplace Resources Resource library Blog Free security courses Product training SecurityFWD Webinars Events Support Community Contact Us Get a demo Get support +1 (877) 292-8767 English Legal | Trust | Privacy Â© 2023 Varonis