**Azure Repos**

Azure Repos is a set of version control tools that you can use to manage your code. Whether your software project is large or small, using version control as soon as possible is a good idea.

Version control systems are software that help you track changes you make in your code over time. As you edit your code, you tell the version control system to take a snapshot of your files. The version control system saves that snapshot permanently so you can recall it later if you need it. Use version control to save your work and coordinate code changes across your team.

Even if you're just a single developer, version control helps you stay organized as you fix bugs and develop new features. Version control keeps a history of your development so that you can review and even roll back to any version of your code with ease.

Azure Repos provides two types of version control:

* + Git : Distributed Version Control
  + Team Foundation Version Control (TFVC) : Centralized Version Control

**Create a new Git Repo in your project:**

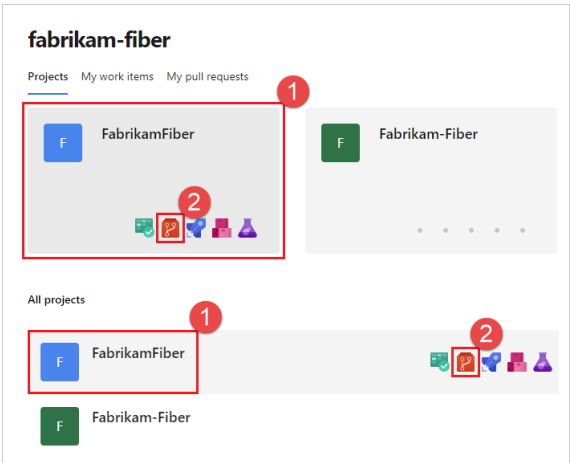
You can use Git repos in your projects to manage your source code as your project grows. This guide shows you how to create a Git repo using the web portal Azure DevOps Services (hosted on Azure).

**Prerequisites:**

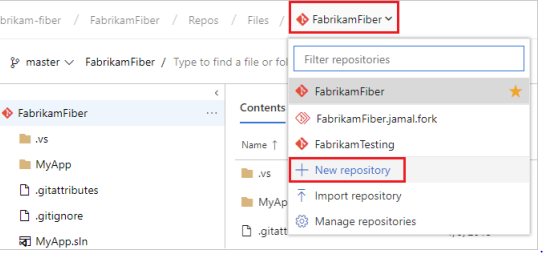
* An organization in Azure DevOps. If you don't have one, you can sign up for one for free. Each organization includes free, unlimited private Git repositories.
* Git command-line tools:
* Install Git for Windows, which includes Git Credential Manager – Windows.

**Create a repo using web portal:**

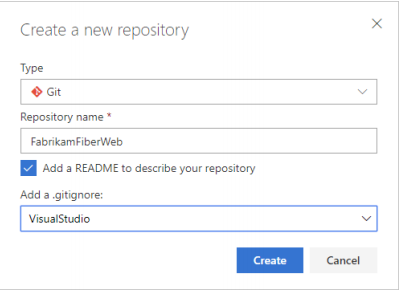
1. Navigate to the Repos page in your project , hovering your mouse over the name of your project, and selecting the Repos icon. You can select it from the All projects list, or from the Recent projects list if you have accessed it recently.



1. From the repo drop-down, select New repository



1. In the Create a new repository dialog, verify that Git is the repo type and enter a name for your new repo. You can also choose to add a README and create a **.gitignore** for the type of code you plan to manage in the repo. A README contains information about the code in your repo, and a **.gitignore** file tells Git which types of files to ignore, such as temporary build files from your development environment.

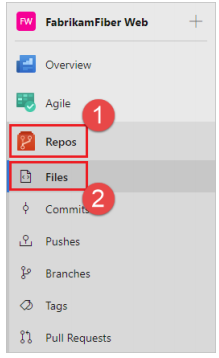


1. When you're done with the repo name and choices, select Create.

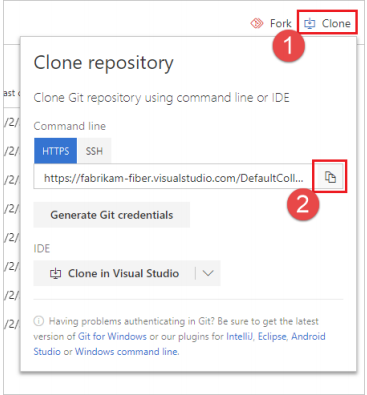
**Cloning the Repo:**

To work with a Git repo, you clone it to your computer. Cloning a repo creates a complete local copy of the repo for you to work with, and downloads all commits and branches in the repo and sets up a named relationship with the repo on the server. Use this relationship to interact with the existing repo, pushing and pulling changes to share code with your team.

1. From your web browser, open the team project for your organization in Azure DevOps and select **Repos > Files**.



1. Select Clone in the upper-right corner of the Files window and copy the clone URL.



1. Open the Git command window (Git Bash on Git for Windows) and browse to the folder where you want the code from the repo stored on your computer. Run **git clone** followed by the path copied from the Clone URL in the previous section, as shown in the following example.

**git clone** [**https://dev.azure.com/fabrikam-fiber/MyFirstProject/\_git/**](https://dev.azure.com/fabrikam-fiber/MyFirstProject/_git/)

Git downloads a copy of the code, including all commits and branches from the repo, into a new folder for Work with the code **cd fabrikam-fiber** you to work with.

1. Switch your directory to the repository that you cloned.

**cd fabrikam-fiber**

Keep this command window open, because you'll use it in the following steps.

**Work with the code:**

In this step, we'll make a change to the files on your computer, commit the changes locally, push the commit up to the repo that is stored on the server, and view the changes there.

1. Browse to the folder on your computer where you cloned the repo and open the README.md file in your editor of choice.

2. Make some changes, for example add **This is my first edit**. to the file, and save and close the file.

3. In the Git command window, navigate to the fabrikam-fiber directory by entering the following command:

**cd fabrikam-fiber**

4.Commit your changes by entering the following command in the Git command window:

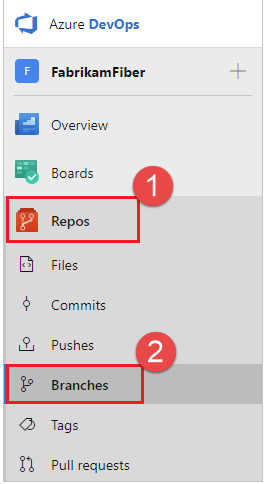
When using **git commit , -a** means to commit all changed files, and **-m** specifies a commit message.

5.Push your changes up to the Git repo on the server by entering the following command into the Git command window:

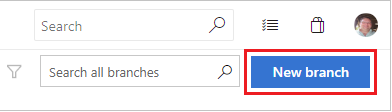
**git push**

**Create a new Git branch:**

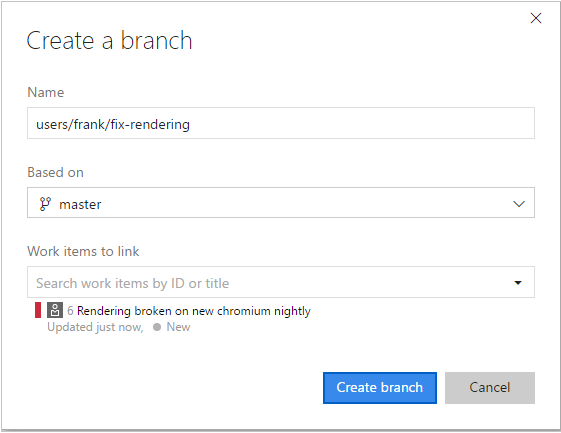
1. View your repo's branches by selecting **Repos** > **Branches** while viewing your repo on the web.



1. Select the **New branch** button in the upper right corner of the page.



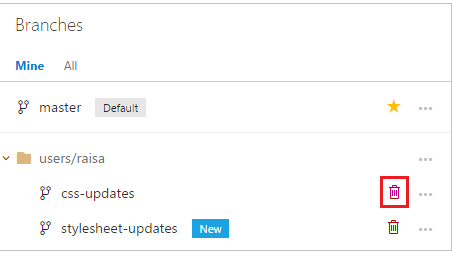
1. In the **Create a branch** dialog, enter a name for your new branch, select a branch to base the work off of, and associate any work items.



1. Select **Create branch**.

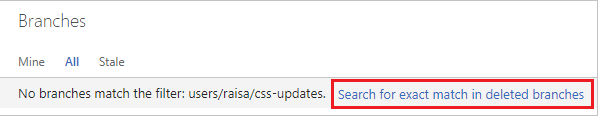
**Delete a branch:**

1. Open your repo on the web and select the branches view.
2. Locate your branch on the branches page. If you don't see it, select **All** to view all branches and filter the branches using the **Search all branches** box in the upper right.
3. Select the trashcan icon next to the branch you want to delete.

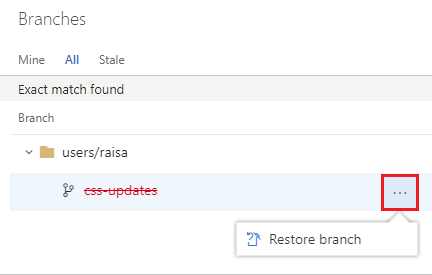


**Restore a deleted branch:**

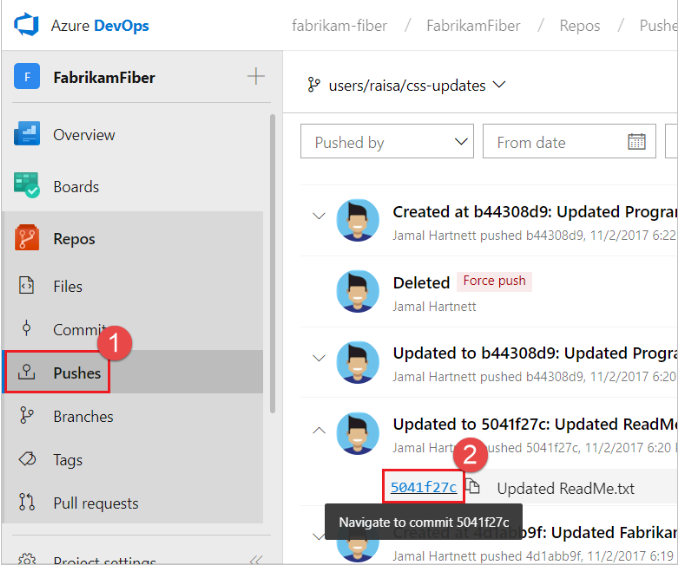
1. Open your repo on the web and select the branches view.
2. Search for the exact branch name using the **Search all branches** box in the upper right.
3. Click the link to **Search for exact match in deleted branches**. If there is a deleted branch that matches your search, you will be able to see which commit it pointed to when it was deleted, who deleted it, and when it was deleted.



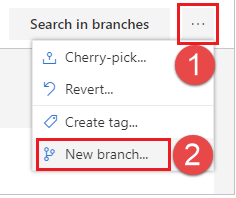
1. To restore the branch, select the **...** icon next to the branch name and then select **Restore branch** from the menu. The branch will be recreated at the last commit to which it pointed. Note that branch policies and permissions will **not** be restored.



If you've re-used the same branch name for different commits, you may not see all the commits you're expecting when you restore the deleted branch. In that case, navigate to the **Pushes** page of the restored branch to see the entire history of the branch.



You can navigate to a specific commit, then select **New branch** from the **...** icon. From there, you can use a pull request, cherry-pick, or merge to get the commits back into the desired branch.

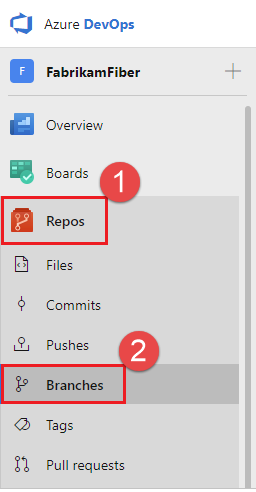


**Branch Policies:**

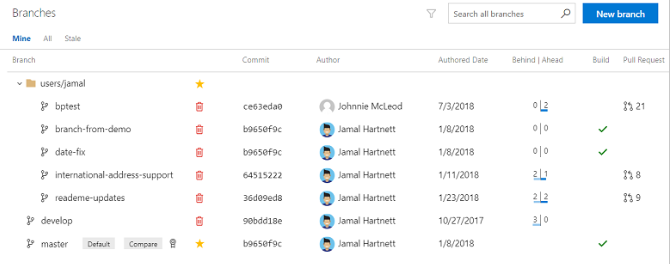
Branch policies help teams protect their important branches of development. Policies enforce your team's code quality and change management standards.

**Configure branch policies:**

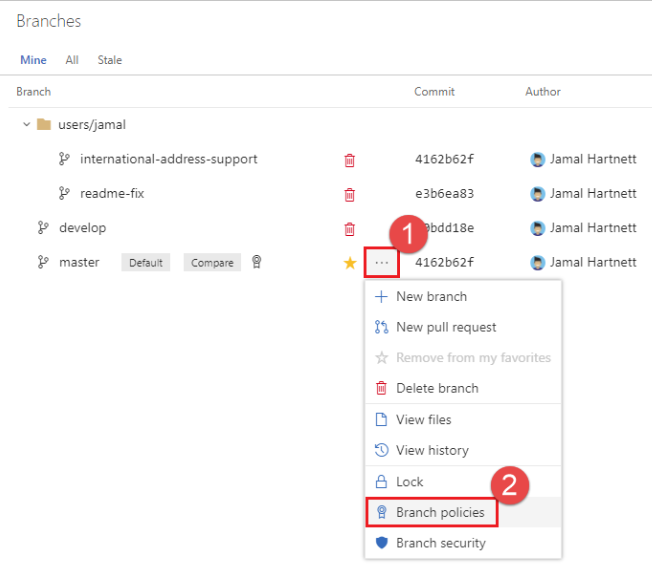
1. Open the **Branches** page by navigating to your project in the web portal and selecting **Repos** > **Branches**.



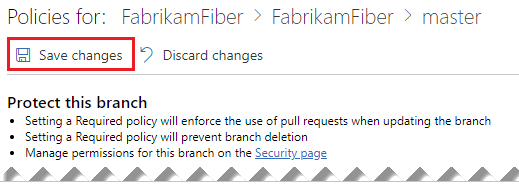
1. Locate your branch in the page. You can browse the list or you can search for your branch using the **Search all branches** box in the upper right



1. Open the context menu for the branch by selecting the **...** icon. Select **Branch policies** from the context menu.



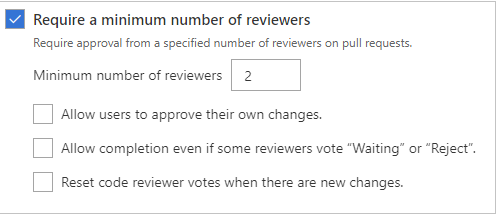
1. Configure your desired policies in the **Policies** page. See the following sections in this article for descriptions for each policy type. Once your policies are configured, select **Save changes** to apply your new policy configuration.



**Require minimum number of reviewers:**

Code reviews are a best practice for most software development projects. To require teams to review their changes before completing a pull request, check **Require a minimum number of reviewers**.

The basic policy requires that a certain number of reviewers approve the code with no rejections.

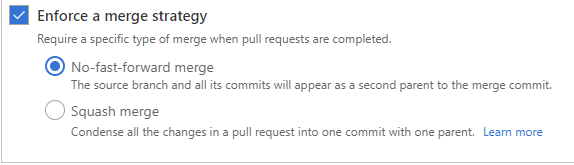


* If **Allow users to approve their own changes** is not selected, the creator of the pull request can still vote **Approve** on their pull request, but their vote won't count toward the **Require a minimum number of reviewers** policy.
* If any reviewer rejects the changes, the pull request cannot be completed unless the **Allow completion even if some reviewers vote "Waiting" or "Reject"** is selected.
* To reset code reviewer votes when new changes are pushed to the source branch, check **Reset code reviewer votes when there are new changes**

When the required number of reviewers approve the pull request, it can be completed.

**Enforce a merge strategy:**

Maintain a consistent branch history by enforcing a merge strategy when a pull request is completed. Select **Enforce a merge strategy** and pick an option to require that pull requests merge using that strategy.



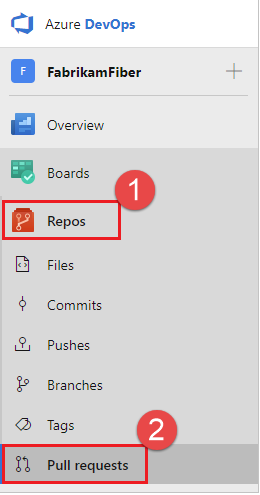
* **No fast-forward merge** - This merges the commit history of the source branch when the pull request closes and creates a merge commit in the target branch.
* **Squash merge** - Complete all pull requests with a squash merge, creating a single commit in the target branch with the changes from the source branch.

**Review the code with pull requests:**

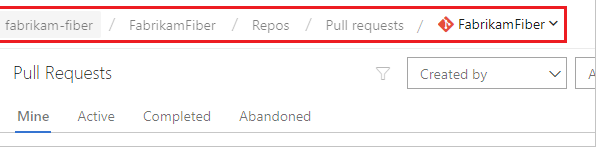
Create pull requests to review and merge code in a Git project. Pull requests let your team review code and give feedback on changes before merging it into the master branch. Pull requests can come from either topic branches within the same repository or from a branch in a fork of the original repository. Reviewers can step through the proposed changes, leave comments, and vote to approve or reject the code.

**View and manage your pull requests:**

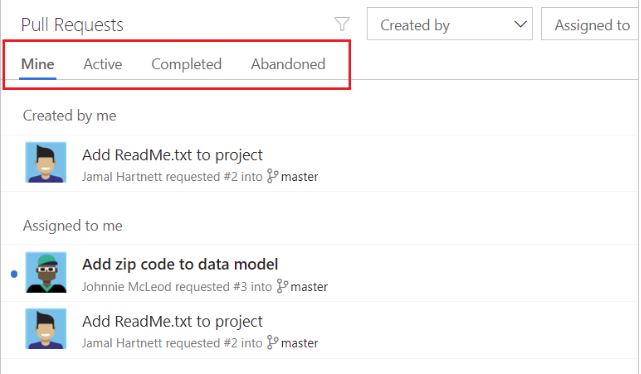
1. To view pull requests in a specific repository in a project, navigate to that project in the web portal and select **Repos** > **Pull requests**.



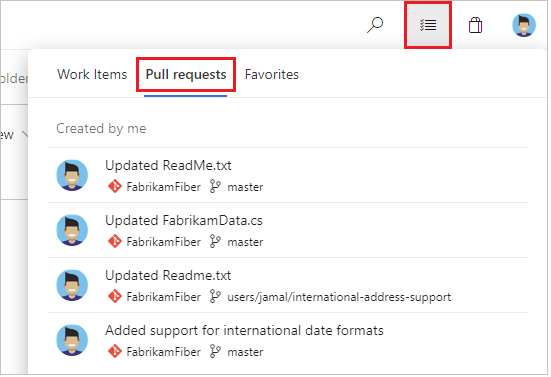
1. Verify that the correct repository is selected.



1. Select **Active** to show all active pull requests for the current repository. Select **Completed** or **Abandoned** to bring up a history of closed pull requests



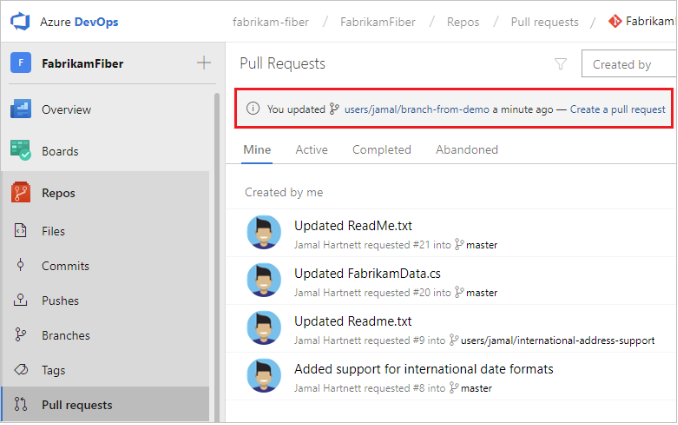
You can view all of your pull requests across different projects in your organization by choosing **Pull requests** in the **My Work** view.

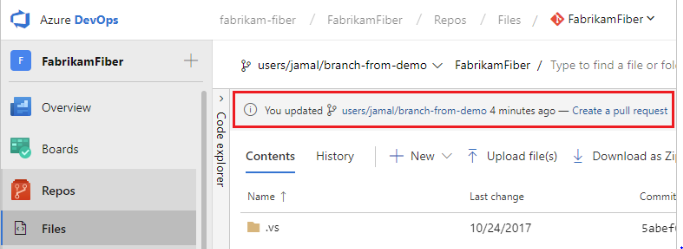


**Create a new pull request:**

**After pushing a branch:**

When you publish or update a feature branch in Azure Repos, you get a prompt asking if you would like to create a pull request. This prompt is displayed on the **Pull Requests** page and the **Files** page.

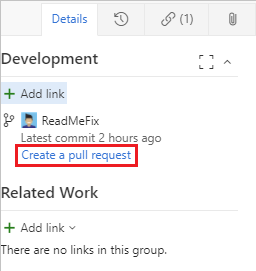


Select the **Create a pull request** link to go to a page where you can enter your pull request details and create the pull request.

### **From a linked work item:**

Create a pull request directly from a work item linked to the branch.

1. From the **Backlogs** or **Queries** tab in the **Work** view , open the work item with the linked branch.
2. In the **Development** area of the work item, there's a link to create a pull request under the branch name.



Select the link to go to a page where you can enter your pull request details and create the pull request.

### **From the Pull requests page on the web:**

Create pull requests from any branch from the **Pull Request** page on the web.



Select **New pull request** in the upper right to go to a page where you can enter your pull request details and create the pull request. Pick the branch you wish to have reviewed and the branch you want to merge the changes into, such as the master branch.

