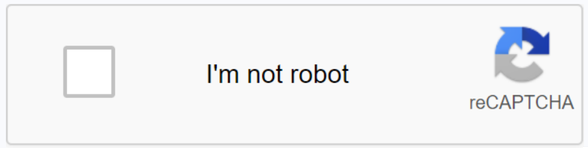

Continue

Bitbucket tutorial español pdf

Looking for bitbucket keywords? Try Ask4Keywords This section provides an overview of what bitbucket is and why a developer might want to use it. You should also mention any big topic within the bitbucket and link to related topics. Because the documentation for bitbucket is new, you may need to create initial versions of these related topics. Create a go to click Start in the top right corner. Enter your email address and click Continue Enter your full name, password and verification code. Then click Continue email to verify that you created your account. After that, you now have a Bitbucket account and you can connect to it. PDF - Download bitbucket for free Learn the basics of Git with this space-centered tutorial. Mission Summary Our mission is to learn how Git works by performing this tutorial, as well as tracking all of your team's space stations. The commands discussed in this tutorial are as follows: git clone, git config, git add, git status, git commit, git push, git pull, git branch, git checkout, and git merge Create a Git repository As our new Bitbucket space station administrator, organization must be very important to you. When you create files for your space station, you'll want to keep them in one place that can be shared with team members, regardless of where they are in the universe. With Bitbucket, this involves adding everything to a repository. Let's create it. Some interesting data about repositories You have access to all files in your local repository, both working with a file and several of them. You can view public repositories without a bitbucket account if you have the URL of these repositories. Each repository belongs to a user account or computer. For user accounts, this user owns the repository. In the case of your computer, your computer is the owner. The repository owner is the only person who can delete a repository. A project code can consist of multiple repositories across multiple accounts, but it can also be a single repository for a single account. Each repository has a size limit of 2 GB, but we recommend that the repository does not exceed 1 GB in size. Step 1. Create the repository Initially, the repository you create in Bitbucket will be empty without any code inside. Okay: You'll start adding files immediately. This bitbucket repository will be the central repository of your files, which means that others can access this repository if you grant them permission. You will also copy a version of this repository to your local system: this way you can update it from a and then transfer these changes to the other. Follow these steps to create a repository: From Bitbucket, click the + icon in the global sidebar and select Repository. Bitbucket displays the Create New Repository page. Take the time to take a look at the contents of the dialog box. With the exception of the repository type, everything you enter on this page can be modified later. Enter BitbucketStationLocations in the Name field. Bitbucket uses this name in the repository URL. For example, if the the_best has a repository called awesome_repo, the URL of this repository . In Access Level, leave the check box checked This is a private repository. Only you and the people you grant access to can see your private repository. If this box is not checked, everyone can see your repository. Choose Git as a repository type. Note that you cannot change the repository type after clicking Create Repository. Click Create repository. Bitbucket creates your repository and displays it on the Presentation page. Step 2. Browse your new repository Take the time to browse the repository you have created. You must be on the Repository Presentation page: Click + from the global sidebar for common actions in a repository. Click the items in the navigation sidebar to see what each contains, including Settings to update repository information and other settings. To see the keyboard shortcuts available to explore these items, press the ? on the keyboard. When you click Sidebar Confirmations, you'll see no confirmations because you haven't created content for the repository. Your repository is private and you have not invited anyone to participate, so the only person who can create or edit repository content right now is you, the owner of the repository. Now that you have a place to add and share your space station files, you need a way to access them from your local system. To do this, you must copy the Bitbucket repository to your system. In Git, we refer to this as cloning a repository. When you clone a repository, create a connection between Bitbucket Server (specified as a source in Git) and its local system. Step 1. Clone the repository on your local system Open a browser and terminal window from your desktop. Once you have opened the terminal window, follow these steps: Navigate to the home directory. $cd : When you start using Bitbucket more often, you'll work with multiple repositories. For this reason, it is a good idea to create a directory that contains all these repositories. Create a directory where all your repositories are stored. $ mkdir repos From the terminal, update the directory in which work in your new repository directory. $cd/repos from Bitbucket, access the BitbucketStationLocations repository. Click the + icon in the global sidebar and select Clone this repository. Bitbucket displays a pop-up cloning dialog box. By default, the cloning dialog box defines the protocol in HTTPS or SSH, depending on the configuration. In this tutorial, do not modify the default protocol. Copies the highlighted cloning command. In the terminal window, copy the command you copied from Bitbucket and press Enter. Enter your Bitbucket password when prompted. If you've created a Google-linked account, use your Google-linked account password. If a Windows password error occurs, follow these steps: Some versions of the Microsoft Windows and Git operating system may fail with an error similar to the following example. Windows Cloning Password Error Example $git clone @bitbucket.org/emmap1/bitbucketstationlocations.git cloning in 'bitbucketspacestation'... Fatal: Could not read Password for ' @bitbucket.org': No such file or directory If this error occurs, enter this information in the command line: $ git config --global core.askpass Next, go back to step 4 and repeat the cloning process. The bash agent must request the password. You only need to perform this action once. At this point, the terminal window should look similar to this: $cd/repos $ git clone @bitbucket.org/emmap1/bitbucketstationlocations.git Cloning in 'bitbucketstationlocations'... Password warning: You appear to have cloned an empty repository. You already knew your repository was empty, didn't you? Note that you have not yet added source files. Lists the contents of the repository directory to see the bitbucketstation station directory in the list. $ls congratulations! You have cloned your repository on your local system. Step 2. Add a file to your local repository and add it to Bitbucket When you have the repository on your local system, it's time to get going. You'll need to start tracking all the locations of your space stations. To do this, create a file with all locations. Access the terminal window and go to the top of the local repository. $ cd /repos/bitbucketstationlocations/ Enter this line in the terminal window to create a new content file. $ eco Earth Moon >> locations.txt If the command line does not return any results, you will have created the file correctly. Get the status of your local repository. The git status command tells you the progress of the project compared to the bitbucket repository. At this point, Git already knows you've created a new file and you'll see something like this: $git status On branch master Initial commit Untracked (use git add... to include in which you will commit) locations.txt nothing added to publish but uns tracked files present (use git add to track) The file has no trace, which means that Git sees a file that does not belong to a previous commission. The status result also shows you the following step to follow: add the file. Instruct Git to track the new locations.txt file using the git add command. Just like when you created the file, the git add command does not return anything when you enter it correctly. $ git add locations.txt The git add command sends changes from the current directory to the Git staging environment. The scenic setting is the area where a snapshot of a set of changes is prepared before committing them to official history. Check file status. $ git status In the initial branch master commit Changes to commit: (use git rm --cached ... to unearth) new file: locations.txt Now you can see that the new file (ready) has been added. The git status command shows the status of the current directory and the snapshot to which it was prepared. Enter the git commit command with a confirmation message, as shown in the next line. The –m indicates that it includes a confirmation message. $git commit -m 'Initial commit' [master (root-commit) fedc3d3] Initial commit 1 file changed, 1 insertion (+) create mode 100644 locations.txt The git commit command confirms the ready snapshot in the project history. In combination with git add, this process defines the basic workflow for all Git users. So far, all the steps you've followed have been taken on your local system so that they can't be viewed in the bitbucket repository until you submit your changes. Learn more about Git and remote repositories Git's ability to communicate with remote repositories (if any, Bitbucket is the remote repository) is the basis of all git collaboration workflows. The Git collaboration model provides each developer with their own copy of the repository, which includes their own branch structure and local history. Users often need to share a series of posts instead of a single set of changes. Instead of committing a single set of changes to a local copy of the central repository, Git allows you to share entire branches between repositories. Manage connections to other repositories and publish your local history by sending branches to other repositories. You can see what others have contributed by adding changes from other branches to your local repository. Go back to the local terminal window and send confirmed changes to Bitbucket using the push git source master command. This command specifies that you are sending changes to the master branch (the bitbucket branch) to the source (the You should see something like the following answer: $git push origin master Counting objects: 3, done. Writing objects: 100% (3/3), 253 bytes 0 bytes/s, done. Total 3 (delta 0), reused 0 (delta 0) A @bitbucket.org/emmap1/bitbucketstationlocations.git * [new branch] master -> branch master created to track the remote branch master from the source. Publications are now in the remote repository (source). Go to your BitbucketStationLocations repository in Bitbucket. If you click Confirmations in the sidebar, you'll see a single confirmation in the repository. Bitbucket combines everything you've done on this confirmation and shows you. You can see the Author column shows the value you used when you set up the global Git file (.gitconfig). If you click Font in the sidebar, you'll see that you have a single source file in your repository, the locations.txt file you just added. Remember what the repository was like when you created it? Now it may look different. Incorporate changes from your Git repository to Bitbucket Cloud Next to the space station administrator's list of activities, you need a file with more information about locations. Because you don't have as many locations right now, add them from Bitbucket. Step 1. Create a file in Bitbucket To add a new location file, please do as follows: From your BitbucketStationLocations repository, click Font to open the source directory. Note that you only have one file, locations.txt, in the directory. A. Home page: Click the link to open this page. B. Selection of offices: choose the office you want to see. C. More Options button: Click to open the menu with more options, such as Add File. D. Source file area: View the file directory in Bitbucket. On the Source page, click the More Options button in the top right corner and select Add File from the menu. The More Options button is only displayed when you add at least one file to the repository. A page opens to create a new file, as shown in the following image. A. Branch with new file: Changed if you want to add a new file to a different branch. B. New File Zone: Add content here for the new file. Enter stations in the file name field. Select HTML from the Syntax Mode list. Add the following HTML code to the text box: Bitbucket has the following space stations: Earth Moon Headquarters Click Confirm. The Confirmation Message field appears with this message: stations created online with Bitbucket. Click Confirm in the message field. You already have a new file in Bitbucket! Then you will go to a page with confirmation details, where you can see the change you have just made: . If you want to see a list of confirmations that you have So far, click Confirmations in the sidebar. Step 2. Extract changes from a remote repository Now we need to bring the new file to your local repository. The process is quite simple, basically is to do the procedure contrary to the presentation you did to bring the locations.txt file to Bitbucket. To bring the file to your local repository, follow these steps: open the terminal window and go to the top of your local repository.$ cd /repos/bitbucketstationlocations/ Enter the git pull --all command to incorporate all bitbucket changes (in more complex branch workflows, incorporating and merging all changes may be appropriate). Enter your Bitbucket password when prompted. Your terminal should be similar to this:$git pull --all Fetching origin remote: Counting objects: 3, done. Remote: Compress objects: 100% (3/3), done. remote: Total 3 (delta 0), reused 0 (delta 0) Unpacking objects: 100% (3/3), made. Since fe5a280.. fcbeeb0 Master ->/master Update fe5a280.. fcbeeb0 Quick Stations 5 ++++++++++++++++++++1 file changed, 5 inserts(+) create mode 100644locations The git pull command merges the file from your remote repository (Bitbucket) with your local repository using a single command. Browse the folder in your local system repository and you'll see the file you just added. Fantastic! By adding two files about the space station location, you followed the basic Git workflow (clone, add, confirm, send, and incorporate changes) between Bitbucket and your local system. Using a Git branch to merge a Manage Space Station file involves some responsibilities. Sometimes, you need to store the information with a security lock, especially when you project new solar system locations. Learning offices let you update your files and share information only when you're ready. Branches are more powerful when you work as a team. You can work with the part of a project from your own branch, incorporate update changes from Bitbucket, and then combine all your work in the main branch when you're ready. Our documentation includes a more detailed explanation of why it is a good idea to use the offices. A branch represents a line of development independent of its repository. Consider it a completely new working directory, staging environment and project history. Before you create a new office, you'll start working automatically on the main branch (called master). As a visual example, this diagram shows the master branch and another branch with an error resolution update. Step 1. Create a branch and make a change Create an office where you can add future plans for the space station that are not ready for confirmation. When you want everyone to know these plans, you can combine the changes in the Bitbucket repository and delete the branch you no longer need. It is important to understand that branches are only pointers to commit. When you create a branch, all git has to do is create a new pointer; does not create new sets of files or folders. Before you start, your repository looks like this: To create a branch, follow these steps: Go to the terminal window and go to the top of your local repository using the following command: $cd/repos/bitbucketstationlocations/ Create a branch for the terminal window. $git future plans of the branch This command creates a branch, but does not change to this branch, so your repository resembles this: the repository history does not change, all that happens is that you get a pointer in the current branch. To start working with the new branch, you need to extract the branch you want to use. . Extract the new branch you created to start using it. $git checkout future plans changed to branch 'future plans' The git checkout command works in conjunction with the git branch. Because you're creating a branch to work on something new, every time you create a new branch (with git branch), you need to make sure you extract it (with git purchase) if you're going to use it. Now that you've extracted the new branch, your Git workflow will look something like this: find the bitbucketstation site folder on your local system and open it. You will see that due to the new branch, there are no additional files or folders in the directory. Open the delocations file of the stations using a text editor. Apply a change to the file by adding another station location: Bitbucket has the following space stations: The Headquarters of the Earth Moon, the Mars Recreation Department, saves and closes the file. Enter the git status command in the terminal window. You'll see something like this: $git status On branch future-plans Unscensored changes for publication: (use git add ... to update what will be committed) (use git checkout -- ... to discard modified work directory changes: Are stations no changes added to the post (use git add and/or git commit -a) Have you seen the future line of plans in the branch? If you entered the git status command before, the line has been defined as branch master because you only had one master branch. Before preparing or confirming a change, check this line to make sure the branch to which you want to apply the change is extracted. Prepare the file. $ git add stations Enter the git commit command in the terminal window, as shown below: $ git commit stationlocations -m 'making a change in a branch' [future plans e3b7732] making a change to a changed branch 1 file, 4 inserts(+) With this recent post, the repository will look similar to this: Ha Ha Ha when combining the change you just applied to the master branch. Step 2. Merge your office: Quick fusion Your space station is growing, and it's time to celebrate the opening of your location on Mars. Because you have only created one branch and applied a change, use the branch quick mixing method. You can quickly mix because you have a linear path that goes from the end of the current branch to the target branch. Instead of merging the branches, all Git has to do to integrate the stories is to move (i.e. quickly use forward) the end of the current branch to the end of the target branch. This effectively combines stories, since all counts accessible from the destination branch are already available in the current branch. This branch workflow is common in transitional thematic branches with smaller changes that are not as common as long-term functions. To complete a fast-forward fusion, Follow these steps: Go to your terminal window and navigate to the top level of your local repository.$ cd /repos/bitbucketstationlocations/ Enter the git status command to make sure you commit all your changes and find out which branch you've extracted.$ git status On branch future-plans nothing to commit, working clean directory Switch to the master branch.$ git checkout master Switched to branch 'master' branch is up-to-date with 'origin/master'. Merge future plan office changes with the master office. It should look like something like this:$ git merge future upgrade plans fcbeeb0.. e3b7732 Quick Stations 4++++ 1 file changed, 4 inserts (+)Basically, you moved the master branch pointer to the current header and your repository looks a lot like the previous quick forward combination. Since you no longer plan to use the future plans branch, you can delete it.$ git branch -d future-plans Deleted branch future-plans (era e3b7732). When you delete your future plans office, you can continue accessing it from your master office using a confirmation input document. For example, if you want to undo changes you've added from future plans, use the confirmation ID you just received to return to that office. Enter the git status command to see the results of the combination, which show that your local repository is one step ahead of the remote repository. $git been used In branch master Your office is ahead of origin /master by 1 commitment. (use git push to publish your local counts) nothing to publish, clear the working directory This is what you've done so far: Create a branch and extract it Apply a change to the new Confirm Change branch in the new Integrate Change branch in the main branch the office you no longer use. Then we need to send all this work to Bitbucket, the remote repository. Step 3. Send your change to Bitbucket What you want is for everyone to see the location of your new space station, so you need to send the current status of your local repository to Bitbucket. This diagram shows what happens when your local repository has changes with which the central repository does not count and which you send to Bitbucket. This is how changes are sent to the remote repository: From the repository directory of the terminal window, enter the git drive source master command to send the changes. You will receive a similar result to this: $git push origin master Counting objects: 3, done. Delta compression using up to 8 threads. Compress objects: 100% (3/3), made. Writing objects: 100% (3/3), 401 bytes 0 bytes/s, done. Total 3 (delta 0), reused 0 (delta 0) Https://emmap1@bitbucket.org/emmap1/bitbucketstationlocations.git fcbeeb0.. Master e3b7732 -> Click the Summary page of your bitbucket repository and note that you can see what you sent to the recent activity flow. Click Confirmations and you'll see confirmation that you've applied to your local system. Note that the change still maintains the same confirmation ID it had on your local system. You can also see that the line to the left of the confirmation list has a direct path that does not show offices. This is because the future plans branch never interacted with the remote repository, only the change we have created and committed has made. Click Offices and note that the page also does not have an office record. Click Fonts, and then click the station archive. You can see that the last change that appears in the file has the confirmation ID you just sent. Click file history to see confirmed changes to this file, which will resemble this image. That's it! Do you think you won't remember all the git commands you've just learned? There was no air conditioning in the room. Check out our basic git commands page so you can check it whenever you need it. Need.