|  |  |
| --- | --- |
| Overview Concepts | Contents |
|  | What is Continuous Integration? |
|  | What is Continuous Delivery? |
|  | What is Continuous Deployment? |
|  | What is GIT ? |
|  | How GIT is differenet from other source code repositories like SVN/TFS/Accurev/.. Etc.. |
|  | How to create GIT repository |

What is Continuous Integration?

**Continuous integration** is a **DevOps** software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.

What is Continuous Delivery?

**Continuous Delivery** is a Process of delivering the software/Updates to production in smaller increments, ensuring that the software can be released at any time.

|  |  |
| --- | --- |
|  | What is Continuous Deployment? |

**Continuous** Delivery. **Continuous** Delivery (CD) is a software development practice in which **continuous**integration, **automated** testing, and **automated deployment** capabilities allow high-quality software to be developed and **deployed** rapidly, reliably and repeatedly with minimal manual overhead.

|  |  |
| --- | --- |
|  | What is GIT ? |

**Git** ([/ɡɪt/](https://en.wikipedia.org/wiki/Help:IPA/English))[[7]](https://en.wikipedia.org/wiki/Git#cite_note-:0-7) is a [distributed version-control](https://en.wikipedia.org/wiki/Distributed_version_control) system for tracking changes in [source code](https://en.wikipedia.org/wiki/Source_code) during [software development](https://en.wikipedia.org/wiki/Software_development).[[8]](https://en.wikipedia.org/wiki/Git#cite_note-effcomp-8) It is designed for coordinating work among [programmers](https://en.wikipedia.org/wiki/Programmer), but it can be used to track changes in any set of [files](https://en.wikipedia.org/wiki/Computer_file). Its goals include speed,[[9]](https://en.wikipedia.org/wiki/Git" \l "cite_note-kernel_SCM_saga-9) [data integrity](https://en.wikipedia.org/wiki/Data_integrity),[[10]](https://en.wikipedia.org/wiki/Git#cite_note-integrity_goals-10) and support for distributed, non-linear workflows.[[11]](https://en.wikipedia.org/wiki/Git#cite_note-linusGoogleTalk-11)

How GIT is differenet from other source code repositories like SVN/TFS/Accurev/.. Etc..

**Git** has a centralized server and repository; **SVN** does not have a centralized server or repository. The content in **Git** is stored as metadata; **SVN** stores files of content.**Git** branches are easier to work with than **SVN** branches. **Git** does not have the global revision number feature like **SVN** has.

**Git** is distributed as everyone has a full copy of the whole repo and its history. **TFS**has its own language: Check-in/Check-out is a **different** concept. **Git** users do commits based on distributed full versions with difference checking. ... Shelve-sets in**TFS** are stored in the centeral server.

Accurev offers both the feature where we can have Client Server version control system and limit the number of files available in Developer's workspace but on the same time we can also let developer have the entire copy of the Source Repository

|  |  |
| --- | --- |
|  | How to create GIT repository |

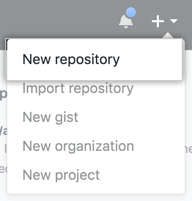
Create a repo

To put your project up on GitHub, you'll need to create a repository for it to live in.

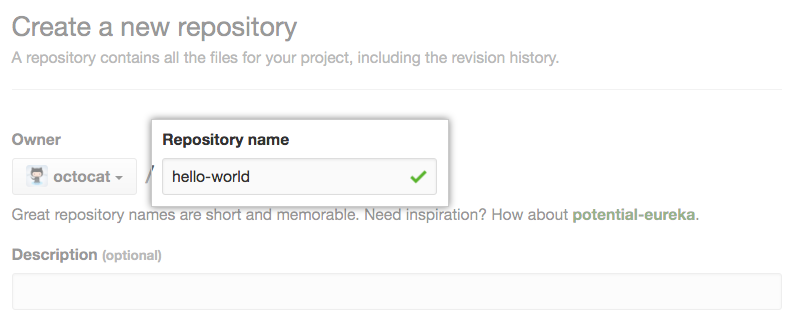
You can store a variety of projects in GitHub repositories, including open source projects. With [open source projects](http://opensource.org/about), you can share code to make better, more reliable software.

**Note:** You can create public repositories for an open source project. When creating your public repository, make sure to include a [license file](http://choosealicense.com/) that determines how you want your project to be shared with others. For more information on open source, specifically how to create and grow an open source project, we've created [Open Source Guides](https://opensource.guide/) that will help you foster a healthy open source community by recommending best practices for creating and maintaining repositories for your open source project. You can also take a free [GitHub Learning Lab](https://lab.github.com/) course on maintaining open source communities.

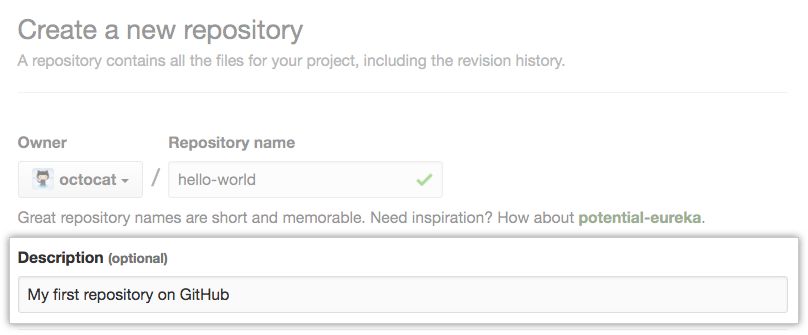
1. In the upper-right corner of any page, click , and then click **New repository**.



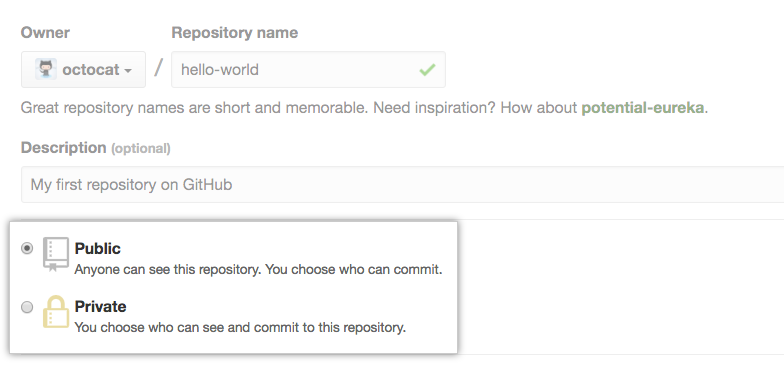
1. Type a short, memorable name for your repository. For example, "hello-world".



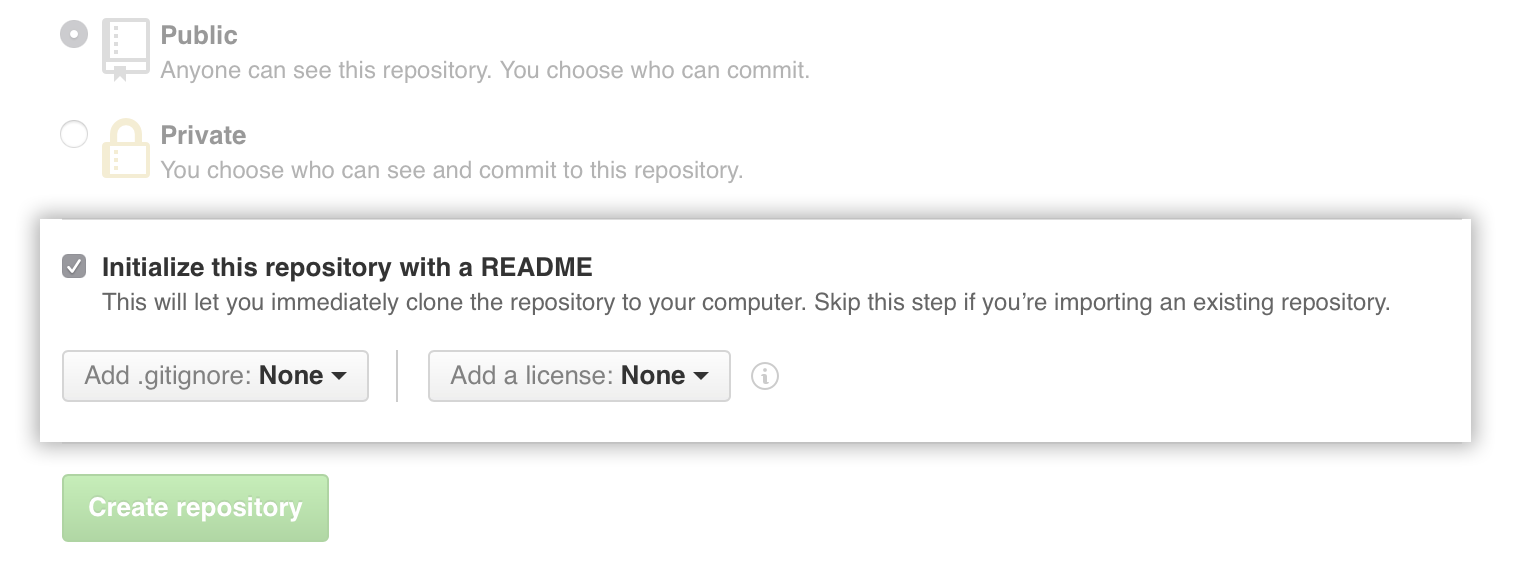
1. Optionally, add a description of your repository. For example, "My first repository on GitHub."



1. Choose to make the repository either public or private. Public repositories are visible to the public, while private repositories are only accessible to you, and people you share them with. For more information, see "[Setting repository visibility](https://help.github.com/en/articles/setting-repository-visibility)."



1. Select **Initialize this repository with a README**.



1. Click **Create repository**.

Congratulations! You've successfully created your first repository, and initialized it with a *README*file.

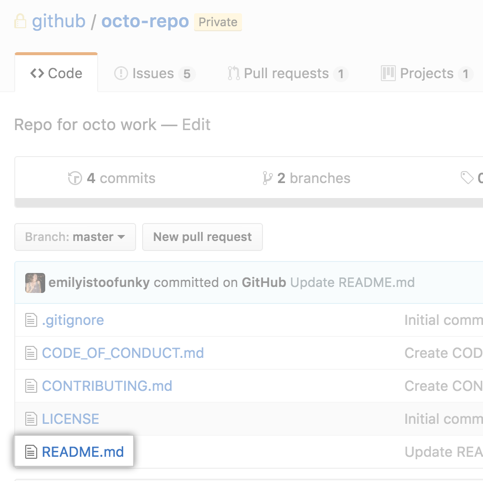
[Commit your first change](https://help.github.com/en/articles/create-a-repo#commit-your-first-change)

A [*commit*](https://help.github.com/en/articles/github-glossary#commit) is like a snapshot of all the files in your project at a particular point in time.

When you created your new repository, you initialized it with a *README* file. *README* files are a great place to describe your project in more detail, or add some documentation such as how to install or use your project. The contents of your *README* file are automatically shown on the front page of your repository.

Let's commit a change to the *README* file.

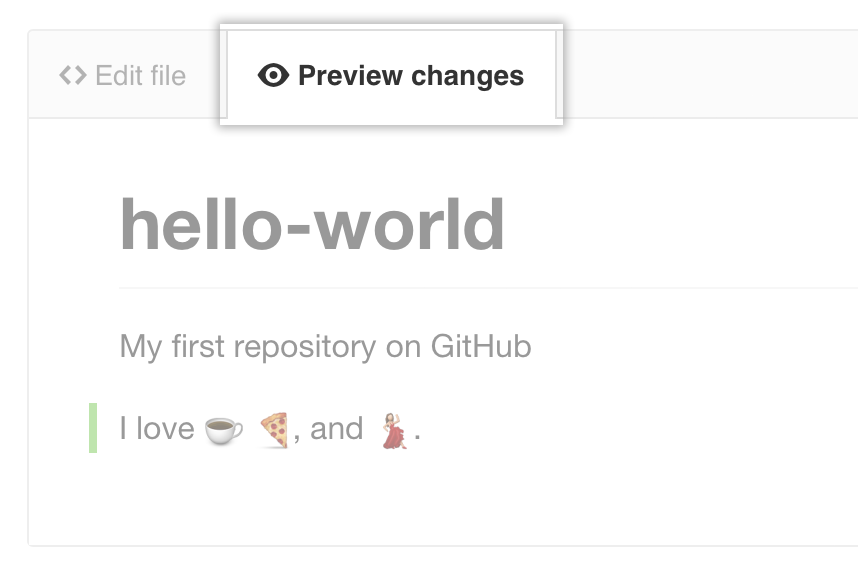
1. In your repository's list of files, click ***README.md***.



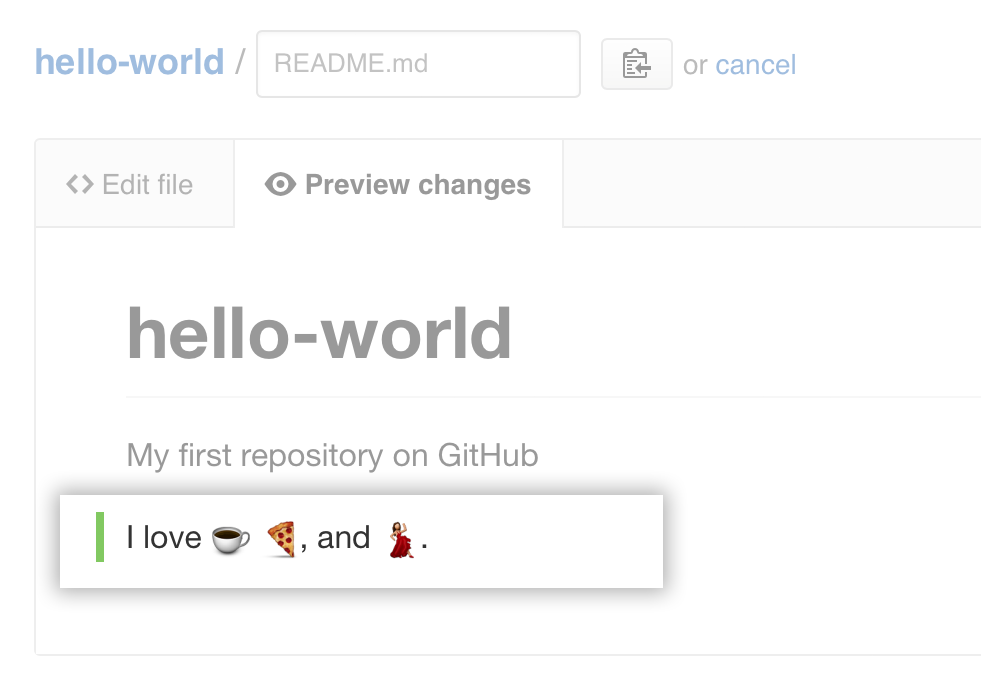
1. Above the file's content, click .
2. On the **Edit file** tab, type some information about yourself.



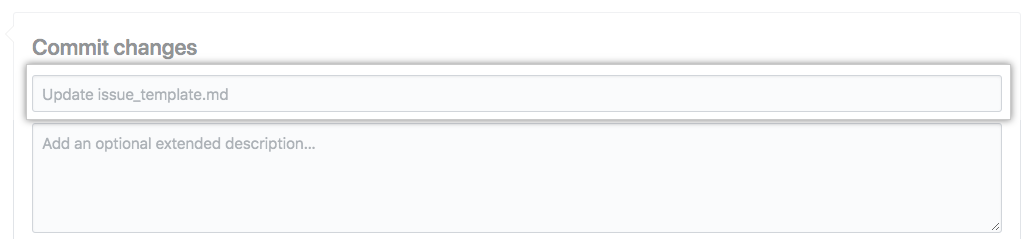
1. Above the new content, click **Preview changes**.



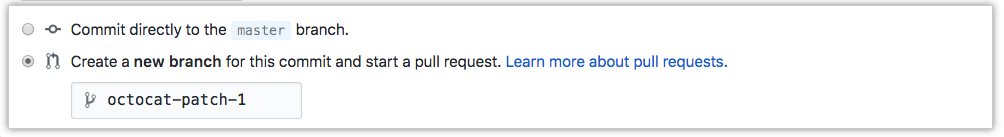
1. Review the changes you made to the file. You'll see the new content in green.



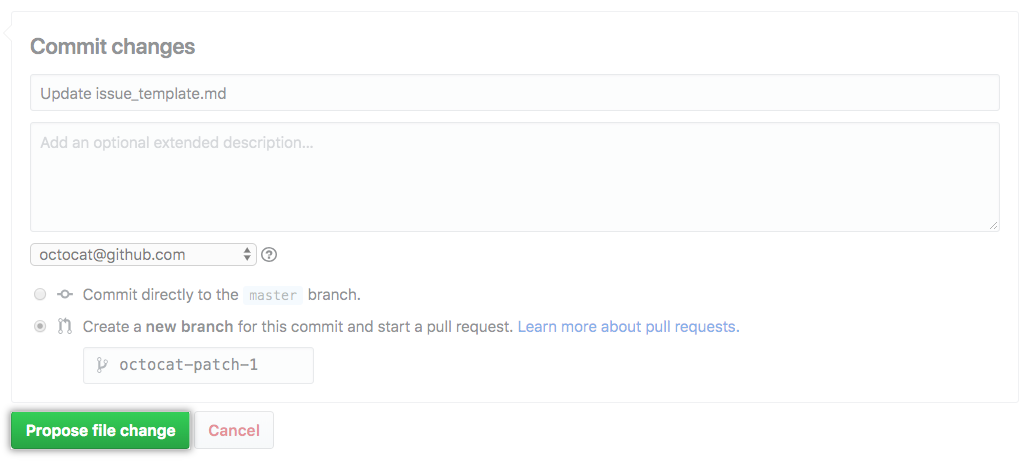
1. At the bottom of the page, type a short, meaningful commit message that describes the change you made to the file. You can attribute the commit to more than one author in the commit message. For more information, see "[Creating a commit with multiple co-authors](https://help.github.com/en/articles/creating-a-commit-with-multiple-authors)."



1. Below the commit message fields, decide whether to add your commit to the current branch or to a new branch. If your current branch is master, you should choose to create a new branch for your commit and then [create a pull request](https://help.github.com/en/articles/creating-a-pull-request).



1. Click **Propose file change**.



GIT BASH:-

# 3. Creating a Project

### Goals

* To learn how to create a git repository from scratch.

## 01Create a “Hello, World!” page

Get started in an empty working directory (for example, work, if you downloaded the file from the previous step) and create an empty directory named “hello”, then create a hello.html file in it with the following contents.

#### RUN:

mkdir hello

cd hello

touch hello.html

#### FILE: HELLO.HTML

Hello, World!

## 02Create a repository

So you have a directory that contains one file. Run git init in order to create a git repo from that directory.

#### RUN:

git init

#### RESULT:

$ git init

Initialized empty Git repository in /Users/alex/Documents/Presentations/githowto/auto/hello/.git/

## 03Add the page to the repository

Now let’s add the “Hello, World” page to the repository.

#### RUN:

git add hello.html

git commit -m "First Commit"

You will see …

#### RESULT:

$ git add hello.html

$ git commit -m "First Commit"

[master (root-commit) 911e8c9] First Commit

1 files changed, 1 insertions(+), 0 deletions(-)

create mode 100644 hello.html

shadharm@DIN80007100 MINGW64 ~

$ mkdir shailaja\_ds

shadharm@DIN80007100 MINGW64 ~

$ cd shailaja\_ds

shadharm@DIN80007100 MINGW64 ~/shailaja\_ds

$ touch hello.html

shadharm@DIN80007100 MINGW64 ~/shailaja\_ds

$ cd hello.html

bash: cd: hello.html: Not a directory

shadharm@DIN80007100 MINGW64 ~/shailaja\_ds

$ cd shailaja\_ds

bash: cd: shailaja\_ds: No such file or directory

shadharm@DIN80007100 MINGW64 ~/shailaja\_ds

$ git init

Initialized empty Git repository in C:/Users/shadharm/shailaja\_ds/.git/

shadharm@DIN80007100 MINGW64 ~/shailaja\_ds (master)

$ git add hello.html

shadharm@DIN80007100 MINGW64 ~/shailaja\_ds (master)

$ git commit -m "First Commit"

[master (root-commit) 05764a8] First Commit

Committer: Dharmapura <shailaja.dharmapura@capgemini.com>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly. Run the

following command and follow the instructions in your editor to edit

your configuration file:

git config --global --edit

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author