

Software Development Environment & Tools

Day #1

Wholeness

- In this lesson 1a, we will focus on setting up our software development environment, with the essential tools.
- Having a well set-up development environment, with the tools correctly configured and ready for use, is important for effective software development process.
- Science of Consciousness: *The field of all possibilities is the source of all solutions.*

Outline

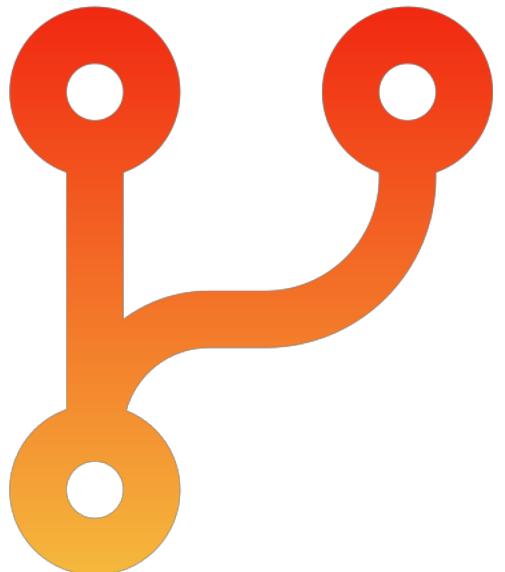
- Java Platform, Standard Edition (Java SE):
- Integrated Development Environment (IDE):
 - Jetbrains IntelliJ IDEA
 - Spring Tools Suite (Eclipse-based IDE for Spring)
- Version Control System: Git and Github
- DevOps and Build Automation Tools:
 - Apache Maven
 - Gradle

Java SE Development Kit (JDK)

- JDK 23 is the latest release of the Java SE Platform.
- JDK 21 is the latest Long-Term Support (LTS) release of the Java SE Platform.
- Obtain, install and configure the Java JDK for your OS. Minimum required version is JDK17
 - Oracle JDK: <https://www.oracle.com/java/technologies/downloads/>
 - If required, consider using SDKMAN (<https://sdkman.io/>) for managing multiple versions of software development kits (SDKs).

Git & GitHub

Unlocking the Power of Git and GitHub



Git

What does Git do?

- Manage projects with Repositories
- Clone a project to work on a local copy
- Control and track changes with Staging and Committing
- Branch and Merge to allow for work on different parts and versions of a project
- Pull the latest version of the project to a local copy
- Push local updates to the main project



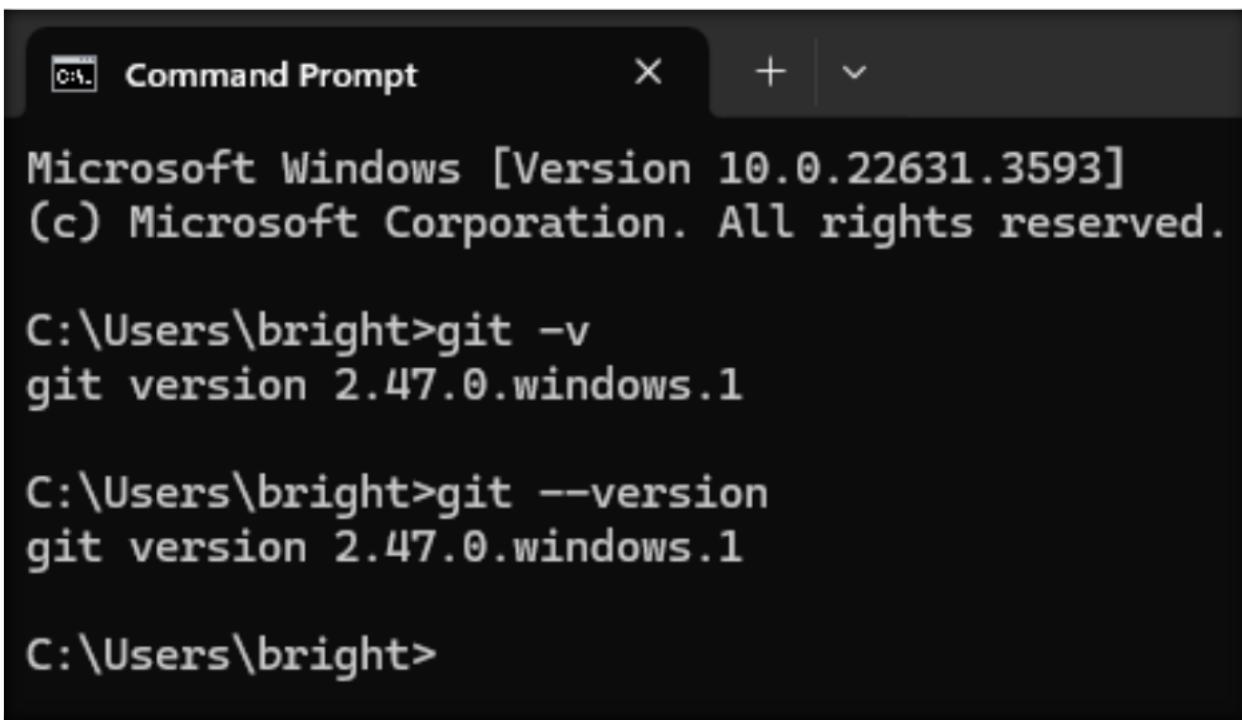
Getting Started With Git

Install Git

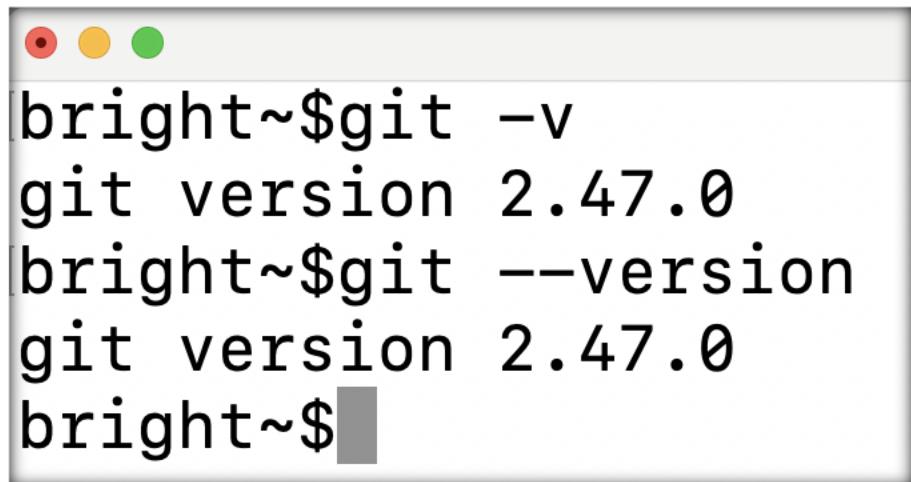
<https://git-scm.com>

Using Git

Command line



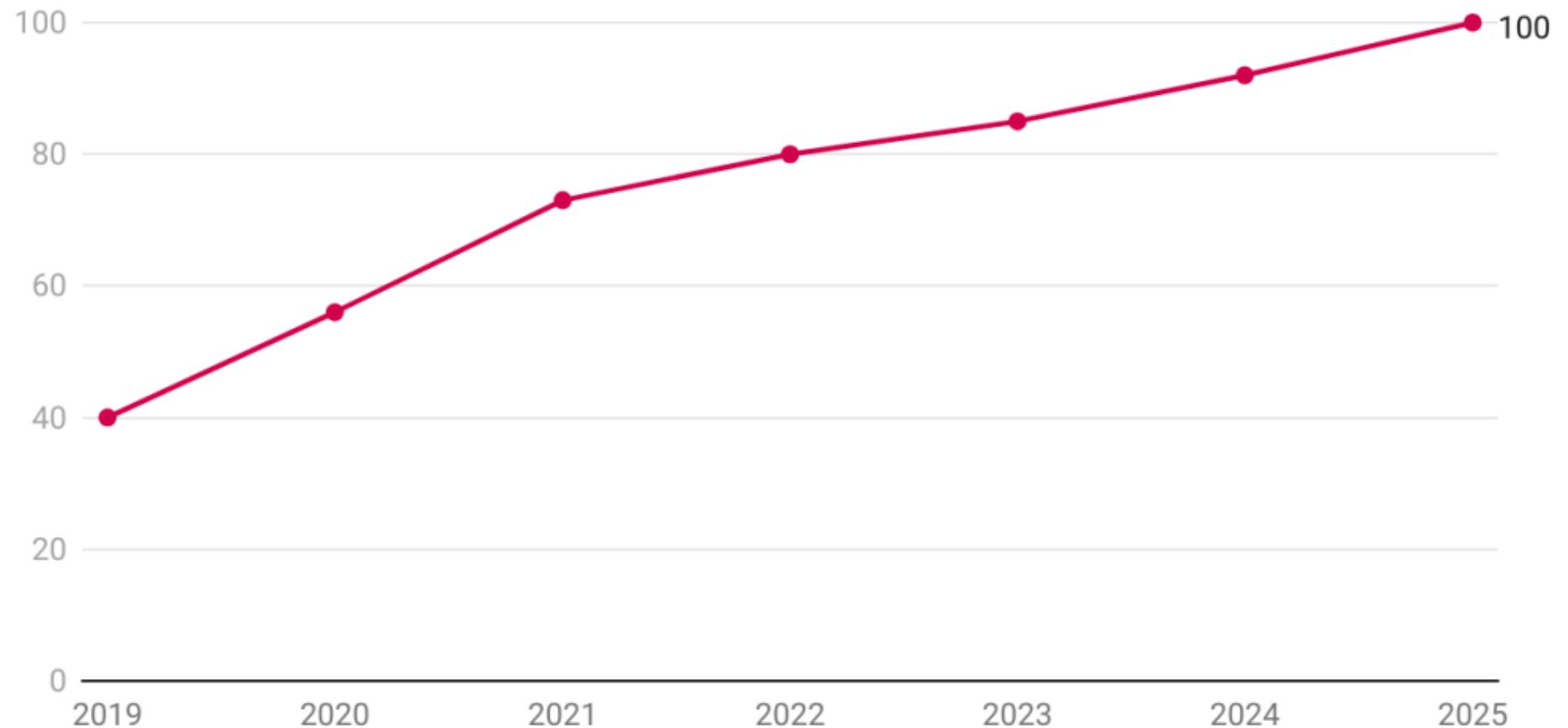
```
Command Prompt × + ▾  
Microsoft Windows [Version 10.0.22631.3593]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\bright>git -v  
git version 2.47.0.windows.1  
  
C:\Users\bright>git --version  
git version 2.47.0.windows.1  
  
C:\Users\bright>
```



```
bright~$git -v  
git version 2.47.0  
bright~$git --version  
git version 2.47.0  
bright~$
```

The GitHub community is set to hit 100 million users by 2025

Total number of users in millions



Source: Enterprise Apps Today

Introduction to Git Configurations

What are Git Configurations?

- Settings that control Git's behavior.
- Can be set at three levels:
 - local
 - global
 - system

Config levels and files

Git

```
bright~$pwd  
/Users/bright/Documents/June 2024/git-demo  
bright~$ls -a  
.  
..  
.git  
welcome.txt  
bright~$cd .git  
bright~$ls -a  
.HEAD  
..config  
COMMIT_EDITMSG  
description  
bright~$cat config  
[core]  
repositoryformatversion = 0  
filemode = true  
bare = false  
logallrefupdates = true  
ignorecase = true  
precomposeunicode = true  
[remote "origin"]  
url = https://github.com/brightgeevarghese/git-demo.git  
fetch = +refs/heads/*:refs/remotes/origin/*  
[branch "main"]  
remote = origin  
merge = refs/heads/main  
bright~$
```

- -local
 - Local configuration values are stored in a file that can be found in the repo's .git directory: .git/config

Config levels and files

```
bright~$ cat ~/.gitconfig
[user]
    name = Bright Gee Varghese
    email = brightlsom@gmail.com
[filter "lfs"]
    clean = git-lfs clean -- %f
    smudge = git-lfs smudge -- %f
    process = git-lfs filter-process
    required = true

[safe]
    directory = /Volumes/HD II

[init]
    defaultBranch = main

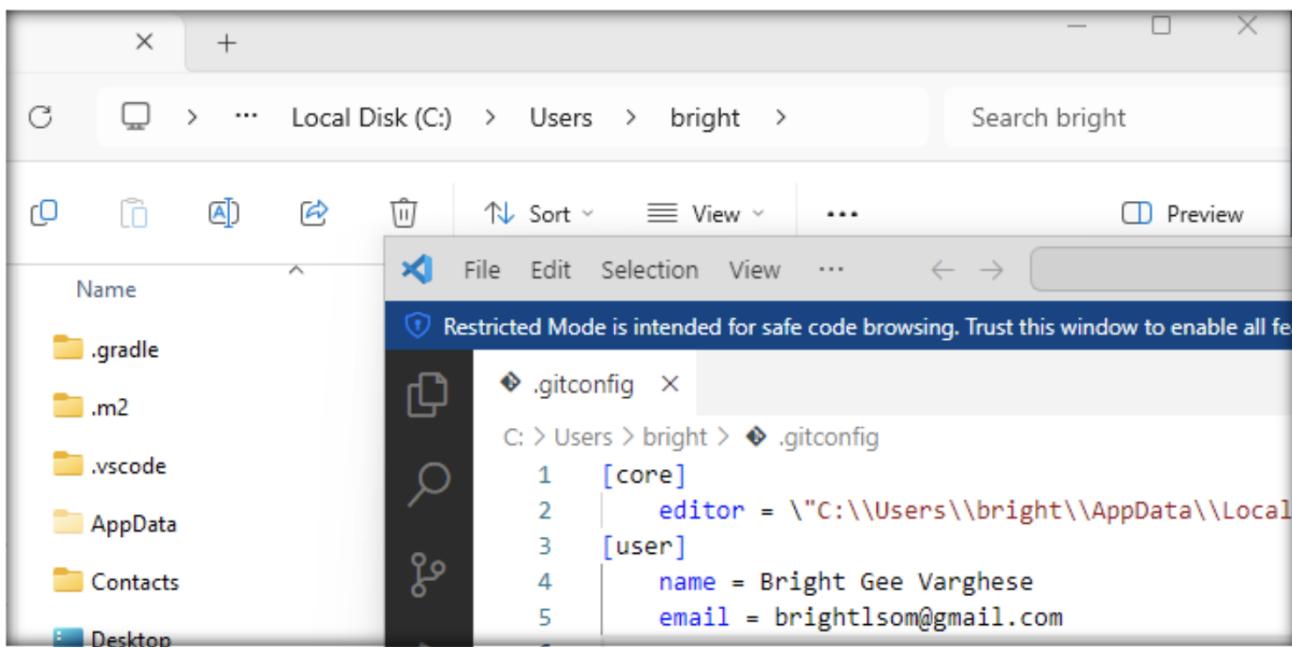
[pull]
    ff = only

[core]
    editor = nano

[credential]
    helper = osxkeychain
bright~$
```

- **--global**

- Global level configuration is user-specific, meaning it is applied to an operating system user.
- Global configuration values are stored in a file that is located in a user's home directory.
 - `~/.gitconfig` on unix systems
 - `C:\Users\[User]\.gitconfig` on windows



Config levels and files

Git

```
bright~$git config --system --list
credential.helper=osxkeychain
bright~$cat /usr/local/etc/gitconfig
[credential]
    helper = osxkeychain
bright~$
```

- - -system
 - System-level configuration is applied across an entire machine.
 - This covers all users on an operating system and all repos.
 - The system level configuration file lives in a gitconfig file off the system root path. \$(prefix)/etc/gitconfig on unix systems. On windows this file can be found at C:\ProgramFiles\Git\etc\gitconfig on Windows 11.

cycle Bin

Microsoft Edge

database11

PDF upgradeList

Viso key autorun Viso key gitconfig Microsoft git ref gitconfig gitcon +

File Edit View

```
[diff "astextplain"]
    textconv = astextplain
[filter "lfs"]
    clean = git-lfs clean -- %f
    smudge = git-lfs smudge -- %f
    process = git-lfs filter-process
    required = true
[http]
    sslBackend = openssl
    sslCAInfo = C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
[core]
    autocrlf = true
    fscache = true
    symlinks = false
    fsmonitor = true
[pull]
    rebase = false
[credential]
    helper = manager
[credential "https://dev.azure.com"]
    useHttpPath = true
[init]
    defaultBranch = main
[user]
    email = brightlsom@gmail.com
    name = Bright Gee Varghese
```

Administrator: Command Prompt

Microsoft Windows [Version 10.0.22631.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>git config --system user.email "brightlsom@gmail.com"
C:\Windows\System32>git config --system user.name "Bright Gee Varghese"
C:\Windows\System32>

etc

< This PC >

Name Date modified Type Size

Name	Date modified	Type	Size
pkcs11	5/7/2024 2:53 PM	File folder	
pki	5/7/2024 2:53 PM	File folder	
profile.d	5/7/2024 2:53 PM	File folder	
ssh	5/7/2024 2:53 PM	File folder	
bash	4/29/2024 8:09 PM	Bash Logout Sour...	1 KB
bash	4/29/2024 8:09 PM	Bash RC Source File	3 KB
DIR_COLORS	4/29/2024 8:09 PM	File	5 KB
docx2txt	4/29/2024 8:09 PM		
fstab	4/29/2024 8:09 PM		
gitattributes	4/29/2024 8:09 PM		
git-bash	5/7/2024 2:53 PM		
gitconfig	5/7/2024 10:02 PM		
hosts	5/7/2022 12:22 AM		
inputrc	4/29/2024 8:09 PM		

26 items | 1 item selected 551 bytes |

Config levels and files

- -system

Git

system

Windows

How to edit git configurations?

```
bright~$git config --system --list
```

```
credential.helper=osxkeychain
```

```
bright~$git config --system user.name "Bright Gee Varghese"
```

```
bright~$git config --system user.email "brightlsom@gmail.com"
```

```
bright~$git config --system --list
```

```
credential.helper=osxkeychain
```

```
user.name=Bright Gee Varghese
```

```
user.email=brightlsom@gmail.com
```

```
bright~$
```

```
git config --global --unset user.name  
git config --global --unset user.email
```

Commands to See all configurations

```
bright~$git config --local --list fatal: --local can only be used inside a git repository
```

```
bright~$git config --global --list
```

```
bright~$git config --system --list
```

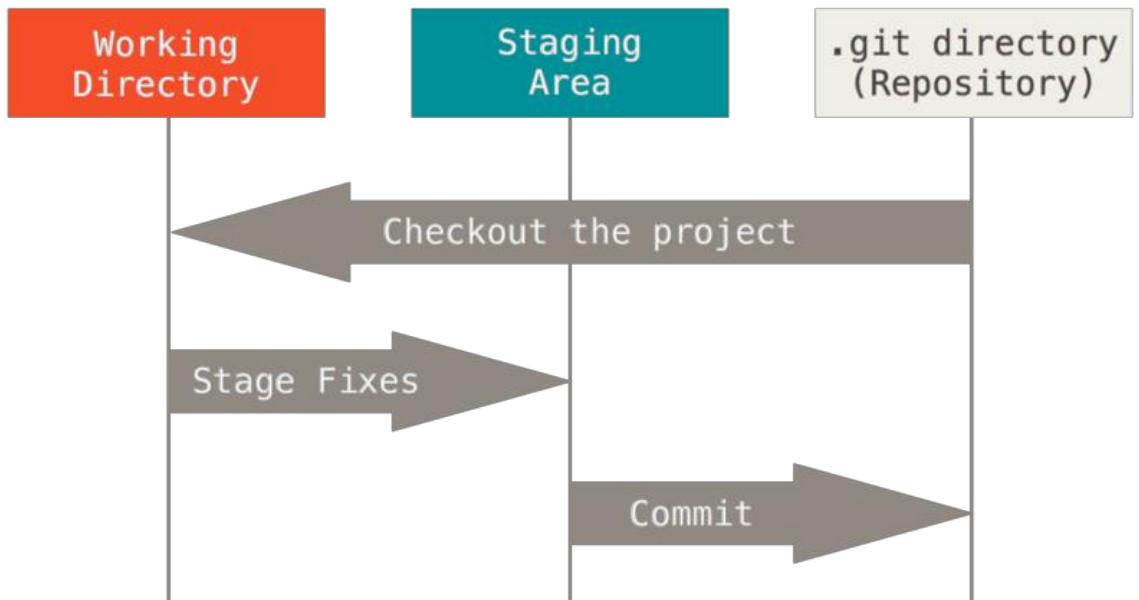
Order of Priority

Git

- The order of priority for configuration levels are:
 - local
 - global
 - system
- This means when looking for a configuration value, Git will start at the local level and bubble up to the system level.

States

Git



1. **Working Directory**: Where you work on your files locally.
2. **Staging Area (Index)**: A temporary holding area where you choose changes to include in the next commit.
3. **Repository**: Where all committed changes and project history are stored.

```
bright~$pwd
/Users/bright/Documents/June 2024
bright~$mkdir mygitdemo
bright~$cd mygitdemo
bright~$ls -a
.
..
bright~$git init
Initialized empty Git repository in /Users/bright/Documents/June 2024/mygitdemo/.git/
bright~$ls -a
[.] [..] .git
bright~$
```

Initialize

Git

Navigate to project folder

git init

.git folder is created

Add New File

Git



The screenshot shows a Java code editor interface. On the left is the Explorer sidebar with icons for File, Search, Git, and Project. The 'OPEN EDITORS' section shows a file named 'HelloWorld.java'. The 'MYGITDEMO' section shows a '.gitignore' file and another 'HelloWorld.java' file. The main editor area displays the following Java code:

```
1 public class HelloWorld{  
2     public static void main(String[] args) {  
3         System.out.println("Hello World!");  
4     }  
5 }
```

The code editor has syntax highlighting and includes a 'Run | Debug' button above the code.

```
.DS_Store  
*.class
```

Add New File & Commit

Git

```
bright~$touch .gitignore
```

```
bright~$nano .gitignore
```

```
bright~$git add .  
bright~$git status  
On branch main
```

```
No commits yet
```

```
Changes to be committed:  
(use "git rm --cached <file>..." to unstage)  
  new file:  .gitignore  
  new file:  HelloWorld.java
```

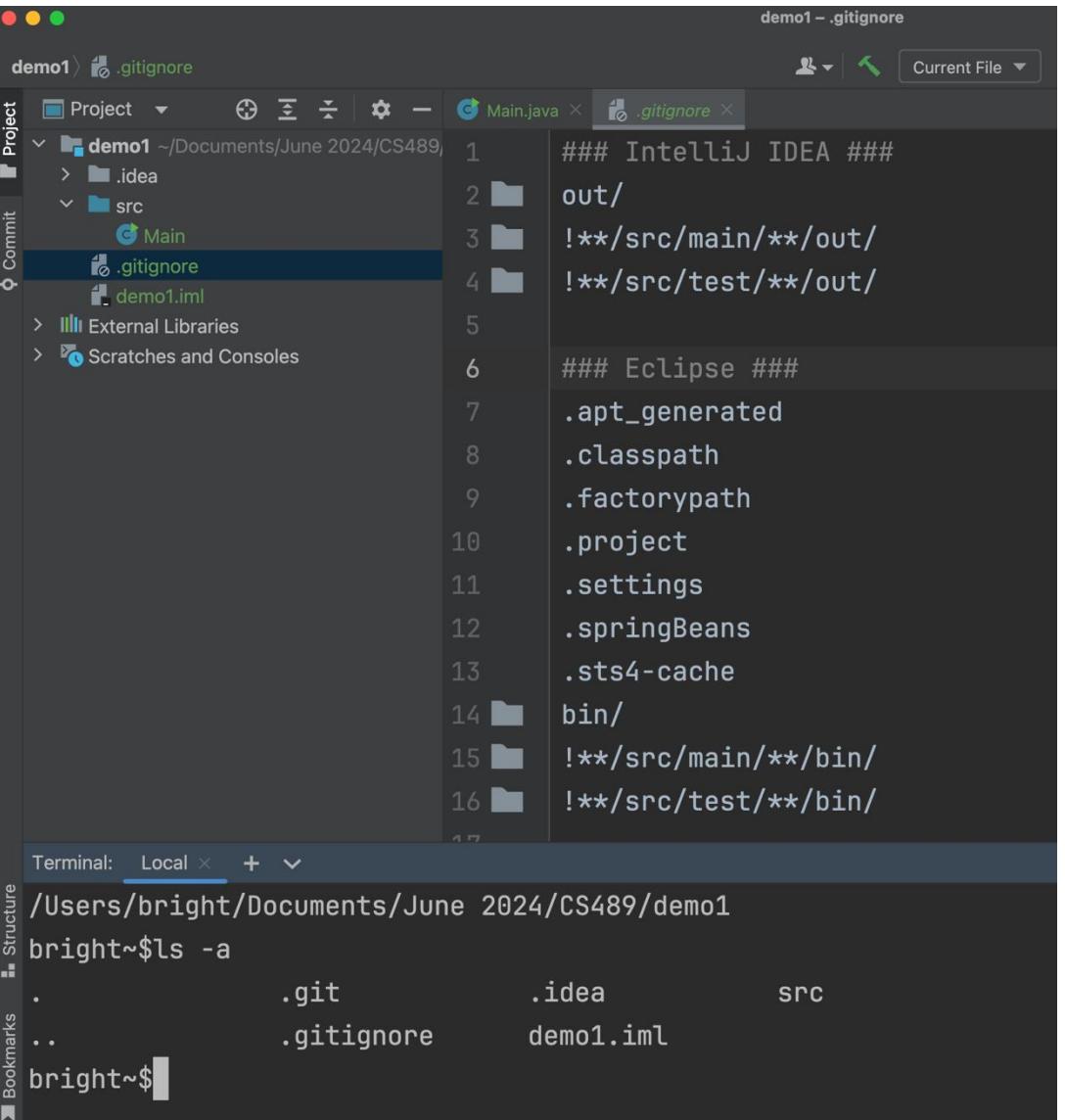
```
bright~$git commit -m "Added HelloWorld.java and .gitignore"  
[main (root-commit) 74d2436] Added HelloWorld.java and .gitignore  
  2 files changed, 7 insertions(+)  
  create mode 100644 .gitignore  
  create mode 100644 HelloWorld.java  
bright~$
```



Git ignore

Ignored files are usually build artifacts and machine generated files that can be derived from your repository source or should otherwise not be committed. Some common examples are:

- dependency caches, such as the contents of /node_modules or /packages
- compiled code, such as .o, .pyc, and .class files
- build output directories, such as /bin, /out, or /target
- files generated at runtime, such as .log, .lock, or .tmp
- hidden system files, such as .DS_Store or Thumbs.db
- personal IDE config files, such as .idea/workspace.xml



The screenshot shows the IntelliJ IDEA interface with the project 'demo1' open. The 'Project' tool window on the left shows the directory structure: demo1, .idea, src, Main, .gitignore, and demo1.iml. The 'Current File' tab at the top right is set to '.gitignore'. The code editor on the right displays the contents of the .gitignore file:

```
## IntelliJ IDEA ##
out/
!**/src/main/**/out/
!**/src/test/**/out/

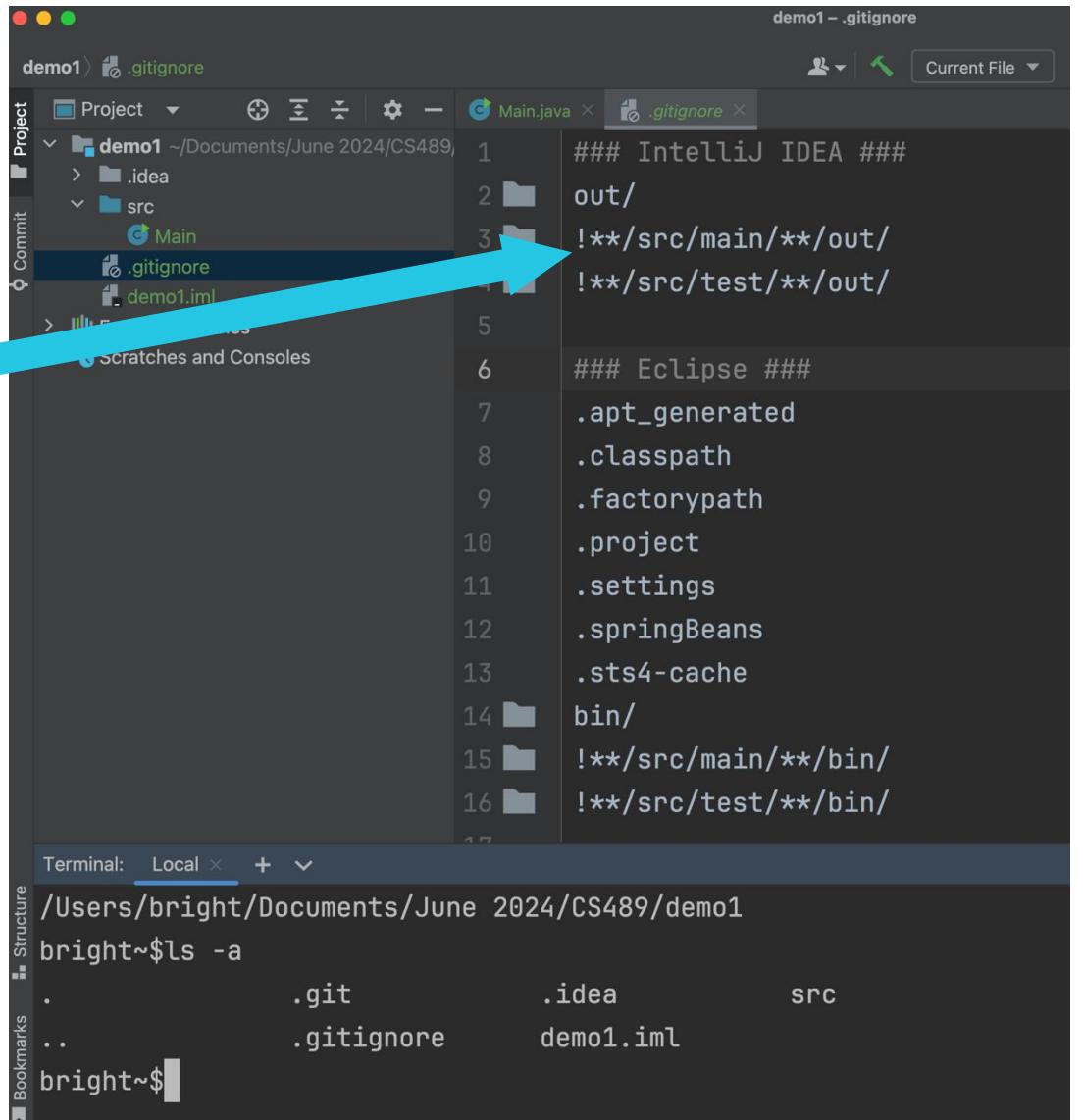
## Eclipse ##
.apt_generated
.classpath
.factorypath
.project
.settings
.springBeans
.sts4-cache
bin/
!**/src/main/**/bin/
!**/src/test/**/bin/
```

Below the code editor is a terminal window showing the command 'ls -a' executed in the project directory:

```
/Users/bright/Documents/June 2024/CS489/demo1
bright~$ls -a
.          .git          .idea          src
..         .gitignore     demo1.iml
```

Git ignore

These lines tell IntelliJ IDEA to ignore all "out" directories within the src/main and src/test folders and their subfolders.



The screenshot shows the IntelliJ IDEA interface with the ".gitignore" file open. A large blue arrow points from the explanatory text above to the file content. The file contains the following ignore patterns:

```
## IntelliJ IDEA ##
out/
!**/src/main/**/out/
!**/src/test/**/out/

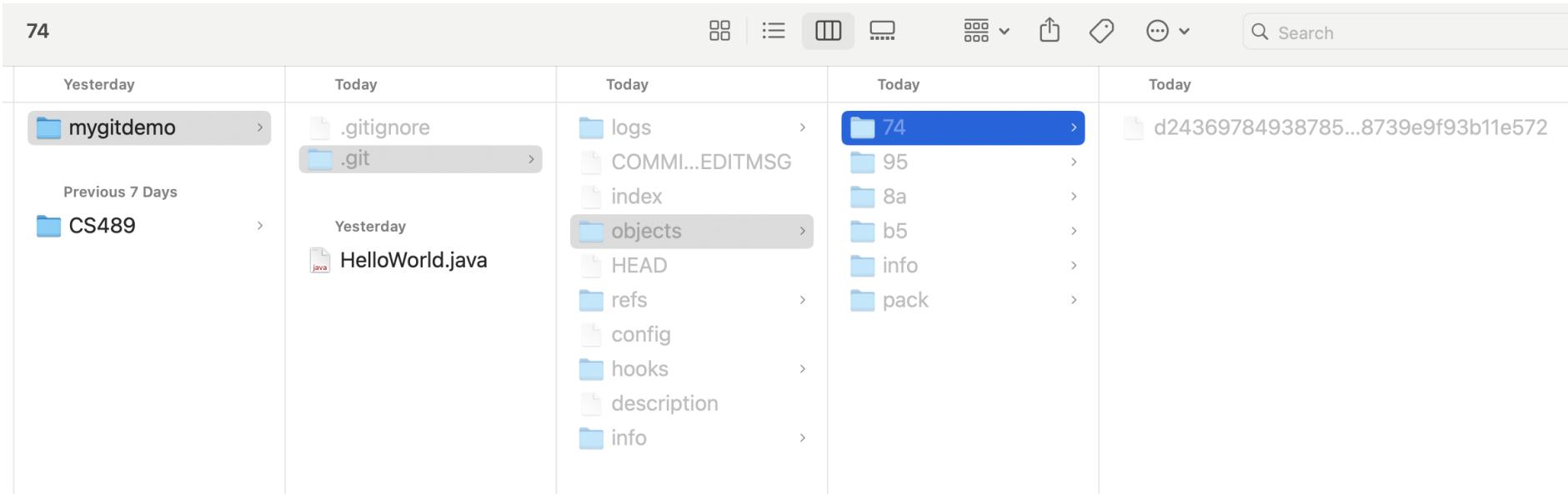
## Eclipse ##
.apt_generated
.classpath
.factorypath
.project
.settings
.springBeans
.sts4-cache
bin/
!**/src/main/**/bin/
!**/src/test/**/bin/
```

In the bottom right corner, a terminal window shows the command `ls -a` being run in the project directory, displaying the files and folders listed in the screenshot.

Commit

Git

```
bright~$git commit -m "Added HelloWorld.java and .gitignore"
[main (root-commit) 74d2436] Added HelloWorld.java and .gitignore
 2 files changed, 7 insertions(+)
  create mode 100644 .gitignore
  create mode 100644 HelloWorld.java
bright~$
```



tree .git

Git

```
bright~$ tree .git
.git
├── COMMIT_EDITMSG
├── HEAD
├── config
├── description
└── hooks
    ├── applypatch-msg.sample
    ├── commit-msg.sample
    ├── fsmonitor-watchman.sample
    ├── post-update.sample
    ├── pre-applypatch.sample
    ├── pre-commit.sample
    ├── pre-merge-commit.sample
    ├── pre-push.sample
    ├── pre-rebase.sample
    ├── pre-receive.sample
    ├── prepare-commit-msg.sample
    ├── push-to-checkout.sample
    └── sendemail-validate.sample
        └── update.sample
    └── index
    └── info
        └── exclude
    └── logs
        ├── HEAD
        └── refs
            └── heads
                └── main
    └── objects
        ├── 74
        │   └── d24369784938785259c28a8739e9f93b11e572
        ├── 8a
        │   └── 98ec140d01doa92df55614a10f827ee91e1dof
        ├── 95
        │   └── f6061257cb18342dfacc6df204a29d731bcc00
        ├── b5
        │   └── 4392da52039f22673a868d09b704f37e377667
        └── refs
            └── heads
                └── main
            └── tags
```

16 directories, 27 files
bright~\$

Install tree

- Mac
 - brew install tree
- Linux
 - sudo apt-get install tree
- Windows
 - <https://gnuwin32.sourceforge.net/packages/tree.htm>

tree .git

Git

bright~\$tree .git

.git

```
└── COMMIT_EDITMSG
    └── HEAD
    └── config
    └── description
    └── hooks
        ├── applypatch-msg.sample
        ├── commit-msg.sample
        ├── fsmonitor-watchman.sample
        ├── post-update.sample
        ├── pre-applypatch.sample
        ├── pre-commit.sample
        ├── pre-merge-commit.sample
        ├── pre-push.sample
        ├── pre-rebase.sample
        ├── pre-receive.sample
        ├── prepare-commit-msg.sample
        ├── push-to-checkout.sample
        └── sendemail-validate.sample
```

- config is a text file that contains your git configuration for the current repo.
- HEAD contains the current head of the repo.
- hooks contain any scripts that can be run before/after git does anything.
- objects contains the git objects, ie the data about the files, commits etc in your repo.
- refs stores references(pointers).
 - refs/heads contains pointers to branches and refs/tags contains pointers to tags.

Create a new repository

GitHub

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner *



Repository name *

gitdemo

gitdemo is available.

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-waddle](#) ?

Description (optional)

 Public

Anyone on the internet can see this repository. You choose who can commit.

 Private

You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

 You are creating a public repository in your personal account.

Create repository

Add a remote repository

Git

```
bright~$git remote add origin https://github.com/brightgeevarghese/gitdemo.git
```

It tells Git that you want to manage remote repositories, which are copies of your local repository hosted on a server.

It specifies that you want to add a new remote repository.

This is the name you're giving to the remote repository.

This is the URL of the remote repository you want to add.

This command adds a remote repository named origin to your local Git repository, and the remote repository's URL points to the GitHub repository specified.

Rename your current branch

Git

```
bright~$git branch -M main
```

It tells Git that you want to manage branches, which are pointers to specific commits in your Git repository.

-M flag will actually rename your current branch to main.

It is used in Git to rename your current branch to main.

Push the local main branch to a remote repository named origin

Git

```
bright~$git push -u origin main
```

It tells Git that you want to push your local commits to a remote repository.

This is the name of the remote repository you want to push your commits to.

This specifies the local branch you want to push. In this case, you're pushing the main branch.

It establishes a connection (upstream) between your local main branch and the remote main branch on origin. This simplifies future push operations, as Git will remember where to push subsequent changes made to your main branch. You can then simply use `git push` without specifying the remote and branch name each time.

brightgeevarghese / **gitdemo**

Type ↕

Issues Pull requests Actions Projects Wiki Security Insights Settings

 **gitdemo** Public

Pin Unwatch 1 ▾

main 1 Branch 0 Tags Go to file Add file Code

 **brightgeevarghese** Added HelloWorld.java and .gitignore 74d2436 · 1 hour ago 1 Commits

 .gitignore Added HelloWorld.java and .gitignore 1 hour ago

 HelloWorld.java Added HelloWorld.java and .gitignore 1 hour ago

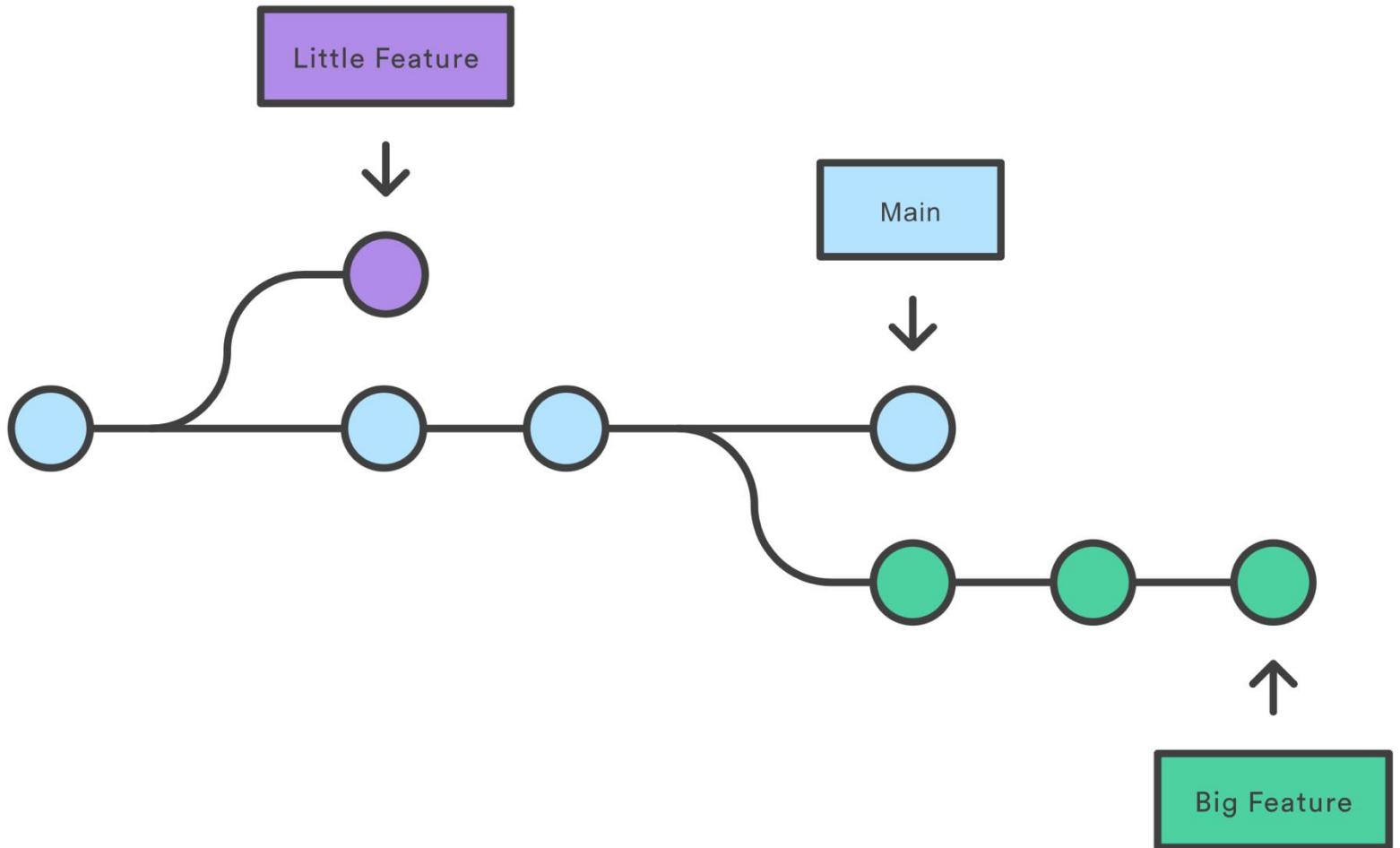
 README

Add a README

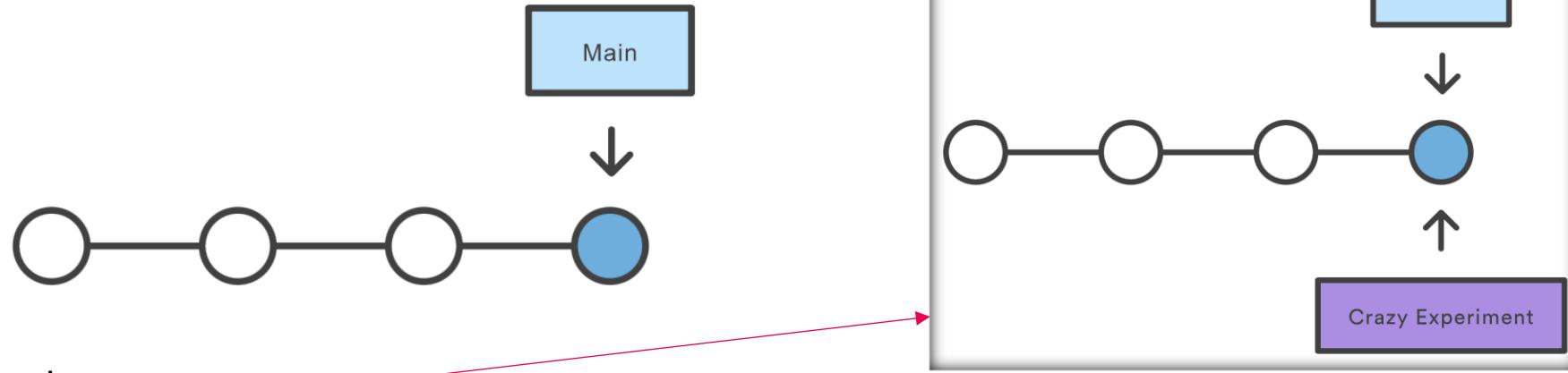
Help people interested in this repository understand your project by adding a README.

Add a README

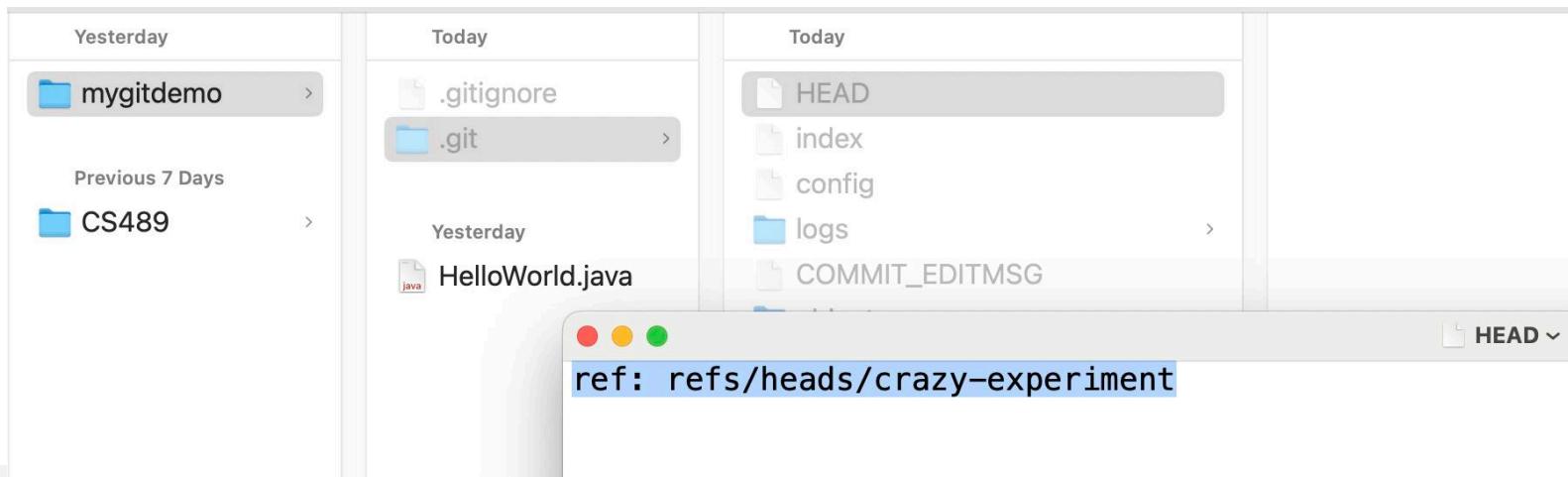
Git Branches



Creating Branches



```
bright~$git branch crazy-experiment
bright~$git branch
  crazy-experiment
* main
bright~$git checkout crazy-experiment
Switched to branch 'crazy-experiment'
bright~$git branch
* crazy-experiment
  main
bright~$
```



Checkout to the new branch

crazy-experiment

```
bright~$git branch  
* crazy-experiment  
 main  
bright~$
```

```
//Add a new line  
public class HelloWorld{  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
        System.out.println("Second line in our file");  
    }  
}
```

Checkout to the new branch

crazy-experiment

```
bright~$git add --all
bright~$git status
On branch crazy-experiment
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
modified: HelloWorld.java
```

```
bright~$git commit -m "Added second line to HelloWorld.java"
[crazy-experiment 7bc2fbc] Added second line to HelloWorld.java
 1 file changed, 1 insertion(+)
```

Checkout to the new branch

crazy-experiment

```
bright~$git branch
* crazy-experiment
  main
bright~$git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
bright~$cat HelloWorld.java
public class HelloWorld{
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
bright~$
```

```
bright~$git checkout crazy-experiment
Switched to branch 'crazy-experiment'
bright~$cat HelloWorld.java
public class HelloWorld{
    public static void main(String[] args) {
        System.out.println("Hello World!");
        System.out.println("Second line in our file");
    }
}
bright~$
```

Push Your Work to the Remote Branch

crazy-experiment

```
bright~$git branch
* crazy-experiment
  main
bright~$git push origin crazy-experiment
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 380 bytes | 380.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'crazy-experiment' on GitHub by visiting:
remote:   https://github.com/brightgeevarghese/gitdemo/pull/new/crazy-experiment
remote:
To https://github.com/brightgeevarghese/gitdemo.git
 * [new branch]  crazy-experiment -> crazy-experiment
bright~$
```

Compare and pull request

crazy-experiment

brightgeevarghese / gitdemo

Issues Pull requests Actions Projects Wiki Security Insights Settings

 gitdemo Public Pin Unwatch 1

 crazy-experiment had recent pushes 1 minute ago Compare & pull request

 main  1 Branch  0 Tags Go to file Add file Code

Author	Commit Message	Date	Commits
 brightgeevarghese	Added HelloWorld.java and .gitignore	74d2436 · 2 hours ago	 1 Commits
	.gitignore	Added HelloWorld.java and .gitignore	2 hours ago
	HelloWorld.java	Added HelloWorld.java and .gitignore	2 hours ago

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#). Learn more about [pull requests](#)

base: main ▾ ← compare: crazy-experiment ▾ ... ✓ **Able to merge.** These branches can be automatically merged.

 Add a title

Added second line to HelloWorld.java

Add a description

Write Preview H B I ℹ️ <> ↗️ | ½ ℹ️ ℹ️ ⌂️ | ⌂️ @ ↵️ ↶ ↷️

Added an another display statement

 Markdown is supported  Paste, drop, or click to add files

Create pull request ▾

 Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Merge

Added second line to HelloWorld.java #1

 Open brightgeevarghe... wants to merge 1 commit into `main` from `crazy-experiment` ↗ now

 Conversation 0  Commits 1  Checks 0  Files changed 1

 brightgeevarghe... commented now

Added an another display statement



 Added second line to HelloWorld.java

7bc2fbc

Add more commits by pushing to the `crazy-experiment` branch on [brightgeevarghe.../gitdemo](#).

 Merge pull request ▾ You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

 Require approval from specific reviewers before merging
Rulesets ensure specific people approve pull requests before they're merged. [Add rule](#)

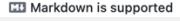
 Continuous integration has not been set up
[GitHub Actions](#) and [several other apps](#) can be used to automatically catch bugs and enforce style.

 This branch has no conflicts with the base branch
Merging can be performed automatically.

 Add a comment

Write [Preview](#) H B I ↵ ↵ ↵ ↵ ↵ ↵ ↵ ↵ ↵ ↵

Add your comment here...

 Markdown is supported  Paste, drop, or click to add files

Added second line to HelloWorld.java #1

 Merged brightgeevarghe... merged 1 commit into `main` from `crazy-experiment` ↗ now

 Conversation 0  Commits 1  Checks 0  Files changed 1



brightgeevarghe... commented 1 minute ago

Added an another display statement



 Added second line to HelloWorld.java

 brightgeevarghe... merged commit `2ec9ad8` into `main` now



Pull request successfully merged and closed

You're all set—the `crazy-experiment` branch can be safely deleted.

Delete branch ↗

Delete Remote Branch

crazy-experiment

```
bright~$ git push origin --delete crazy-experiment
```

To https://github.com/brightgeevarghese/gitdemo.git

- [deleted] crazy-experiment

bright~\$

gitdemo Public

main 2 Branches 0 Tags

brightgeevarghese Merge pull request #1 from brightgeevarghese/crazy-experi... 2ec9ad8 · 2 minutes ago 3 Commits

.gitignore Added HelloWorld.java and .gitignore 2 hours ago

HelloWorld.java Added second line to HelloWorld.java 25 minutes ago



gitdemo Public

main 1 Branch 0 Tags

brightgeevarghese Merge pull request #1 from brightgeevarghese/crazy-experi... 2ec9ad8 · 2 minutes ago 3 Commits

.gitignore Added HelloWorld.java and .gitignore 2 hours ago

HelloWorld.java Added second line to HelloWorld.java 25 minutes ago

Delete Remote Branch

crazy-experiment

```
bright~$git push origin --delete crazy-experiment
```

This tells Git that you want to push something to a remote repository.

This specifies the name of the remote repository you want to push to.

This flag tells Git that you want to delete the branch on the remote repository **after** pushing.

This is the name of the branch you want to delete on the remote repository.

Delete local new branch and update the local main

crazy-experiment

```
bright~$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
bright~$ git branch -d crazy-experiment
error: the branch 'crazy-experiment' is not fully merged
hint: If you are sure you want to delete it, run 'git branch -D crazy-experiment'
hint: Disable this message with "git config advice.forceDeleteBranch false"
bright~$ git branch -D crazy-experiment
Deleted branch crazy-experiment (was 7bc2fbc).
bright~$ git pull origin main
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 2), reused 2 (delta 1), pack-reused 0
Unpacking objects: 100% (4/4), 1.14 KiB | 53.00 KiB/s, done.
From https://github.com/brightgeevarghese/gitdemo
 * branch      main    -> FETCH_HEAD
   74d2436..2ec9ad8 main    -> origin/main
Updating 74d2436..2ec9ad8
Fast-forward
 HelloWorld.java | 1 +
 1 file changed, 1 insertion(+)
bright~$
```

Delete local new branch and update the local main

crazy-experiment

```
bright~$git branch
* main
bright~$cat HelloWorld.java
public class HelloWorld{
    public static void main(String[] args) {
        System.out.println("Hello World!");
        System.out.println("Second line in our file");
    }
}%
bright~$
```

Merge branches locally

emergency-fix

Let us assume, we have two branches

1. main
2. emergency-fix

Change to *main*

- **git checkout main**

Merge the current branch (*main*) with *emergency-fix*

- **git merge emergency-fix**

Delete the branch, *emergency-fix*

git branch -d emergency-fix

Merge branches locally

emergency-fix

```
bright~$git pull origin main

//Adding a for loop
public class HelloWorld{
    public static void main(String[] args) {
        System.out.println("Hello World!");
        System.out.println("Second line in our file");
        for (int i = 0; i < args.length; i++) {
            System.out.println(i);
        }
    }
}

bright~$git add HelloWorld.java

bright~$git commit -m "Added a for loop"
[main a9bde92] Added a for loop
 1 file changed, 3 insertions(+)
```

Merge branches locally

When you've pulled changes from the remote repository and started working on your local code, it's possible that someone else has pushed changes to the remote repository in the meantime. In this scenario, it's crucial to ensure that your local changes are in sync with the latest changes from the remote repository before pushing your code.

```
bright~$git pull origin main
```

```
//Adding a for loop
public class HelloWorld{
    public static void main(String[] args) {
        System.out.println("Hello World!");
        System.out.println("Second line in our file");
        for (int i = 0; i < args.length; i++) {
            System.out.println(i);
        }
    }
}
```

```
bright~$git add HelloWorld.java
```

```
bright~$git commit -m "Added a for loop"
[main a9bde92] Added a for loop
1 file changed, 3 insertions(+)
```

Pull latest changes from GitHub Repository

Diverging means both your local branch and the remote branch have new commits that are not present in each other's history.

```
bright~$git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 1.01 KiB | 257.00 KiB/s, done.
From https://github.com/brightgeevarghese/gitdemo
 * branch      main    -> FETCH_HEAD
 2ec9ad8..179bc0b main    -> origin/main
hint: Diverging branches can't be fast-forwarded, you need to either:
hint:
hint: git merge --no-ff
hint:
hint: or:
hint:
hint: git rebase
hint:
hint:
hint: Disable this message with "git config advice.diverging false"
fatal: Not possible to fast-forward, aborting.
bright~$
```

Pull latest changes from GitHub Repository

git merge origin/main

The screenshot shows a code editor with an open file `HelloWorld.java`. The editor's sidebar lists files in the project structure: `OPEN EDITORS` contains `J HelloWorld.java` and `.gitignore`; `MYGITDEMO` contains `.gitignore` and `J HelloWorld.java`. The main editor area displays the Java code for `HelloWorld`:

```
1 public class HelloWorld{
2     public static void main(String[] args) {
3         System.out.println("Hello World!");
4         System.out.println("Second line in our file");
5 <<<<< HEAD (Current Change)
6         for (int i = 0; i < args.length; i++) {
7             System.out.println(i);
8         }
9 =====
10        System.out.println("Somebody added: Third line for display");
11 >>>>> origin/main (Incoming Change)
12    }
13 }
14 
```

Below the editor, a terminal window titled "mygitdemo -- zsh -- 94x5" shows the output of the command `git merge origin/main`:

```
bright~$git merge origin/main
Auto-merging HelloWorld.java
CONFLICT (content): Merge conflict in HelloWorld.java
Automatic merge failed; fix conflicts and then commit the result.
bright~$
```

Pull latest changes from GitHub Repository

git merge origin/main

J HelloWorld.java > ...

```
1 public class HelloWorld{  
2     public static void main(String[] args) {  
3         System.out.println("Hello World!");  
4         System.out.println("Second line in our file");  
5     }  
6 }  
7
```

Resolve in Merge Editor

The screenshot shows a merge editor interface with two panes. The left pane, labeled 'Incoming' (commit 179bc0b), contains the initial code:1 public class HelloWorld{
2 public static void main(String[] args) {
3 System.out.println("Hello World!");
4 System.out.println("Second line in our file");
5 }
6 }A new line 'System.out.println("Somebody added: Third line")' has been added at line 5, highlighted in green. The right pane, labeled 'Current' (commit a9bde92), contains the code after the merge:1 public class HelloWorld{
2 public static void main(String[] args) {
3 System.out.println("Hello World!");
4 System.out.println("Second line in our file")
5 for (int i = 0; i < args.length; i++) {
6 System.out.println(i);
7 }
8 }
9 }A large blue arrow points from the bottom right of the current pane towards the bottom right of the result pane. The result pane shows the final merged code:1 public class HelloWorld{
2 public static void main(String[] args) {
3 System.out.println("Hello World!");
4 System.out.println("Second line in our file");
5 }
6 }A message 'No Changes Accepted' is displayed at the bottom of the result pane.

Merge Conflict

Visual Studio

1. Accept Incoming | Accept Combination (Incoming First) | Ignore:

1. **Accept Incoming:** This option will accept the changes from the incoming branch (usually the changes from the remote repository).
2. **Accept Combination (Incoming First):** This option will combine the changes from both branches, with the incoming changes being applied first.
3. **Ignore:** This option will ignore the changes and leave the file as it is.

2. Accept Current | Accept Combination (Current First) | Ignore:

1. **Accept Current:** This option will accept the changes from the current branch (usually the changes you made locally).
2. **Accept Combination (Current First):** This option will combine the changes from both branches, with the current changes being applied first.
3. **Ignore:** This option will ignore the changes and leave the file as it is.

Pull latest changes from GitHub Repository

git merge origin/main

The screenshot shows a code merge interface with three tabs at the top: 'J HelloWorld.java ↓M, !' (Incoming), 'J HelloWorld.java > ↗ HelloWorld > ⚙ main(String[])' (Current), and '.gitignore'. The 'Incoming' tab shows code from commit 179bc0b, and the 'Current' tab shows code from commit a9bde92. The 'Result' tab shows the merged code. A large blue arrow points from the 'Result' tab down to the bottom-left editor.

```
1 public class HelloWorld{  
2     public static void main(String[] args) {  
3         System.out.println("Hello World!");  
4         System.out.println("Second line in our file");  
5         System.out.println("Somebody added: Third line");  
6     }  
7 }  
8  
Result HelloWorld.java  
1 public class HelloWorld{  
2     public static void main(String[] args) {  
3         System.out.println("Hello World!");  
4         System.out.println("Second line in our file");  
5         System.out.println("Somebody added: Third line for display");  
6         for (int i = 0; i < args.length; i++) {  
7             System.out.println(i);  
8         }  
9     }  
10 }
```

The bottom-left editor shows the merged Java code from the previous screenshot. A large blue arrow points from the 'Result' tab up to this editor. A 'Complete Merge' button is visible in the top right corner of the editor area.

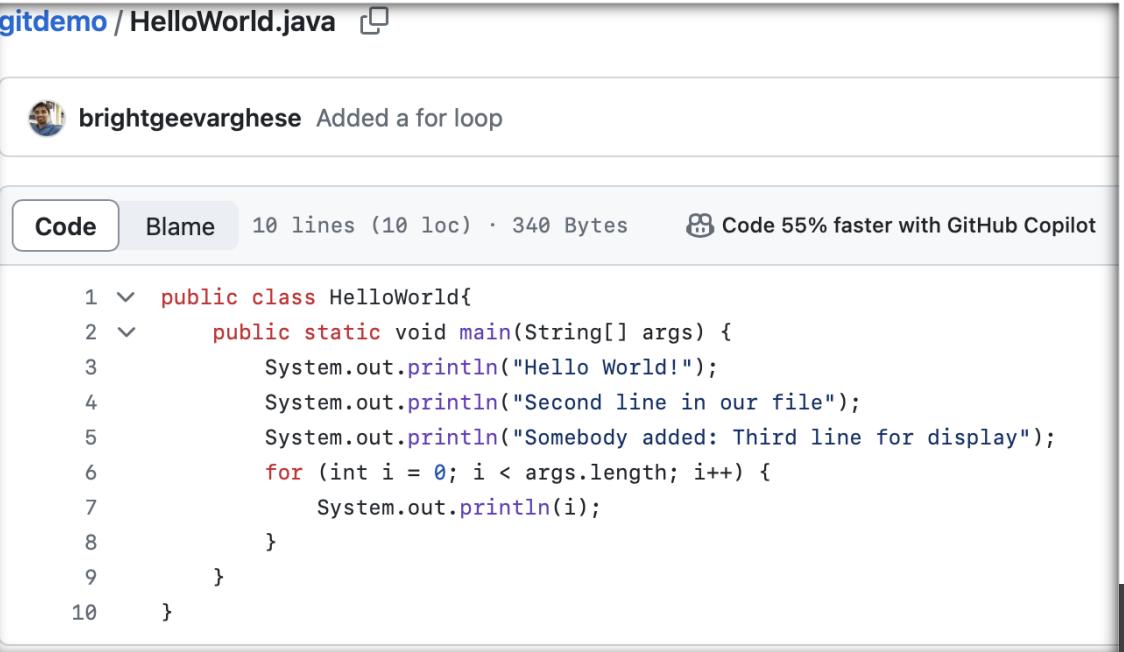
```
1 public class HelloWorld{  
2     public static void main(String[] args) {  
3         System.out.println("Hello World!");  
4         System.out.println("Second line in our file");  
5         System.out.println("Somebody added: Third line for display");  
6         for (int i = 0; i < args.length; i++) {  
7             System.out.println(i);  
8         }  
9     }  
10 }
```

Commit the merge & Push the Merged Changes to Remote

Visual Studio

```
bright~$git commit -m "Added a for loop"  
[main 929291c] Added a for loop  
bright~$git push  
Enumerating objects: 10, done.  
Counting objects: 100% (10/10), done.  
Delta compression using up to 12 threads  
Compressing objects: 100% (6/6), done.  
Writing objects: 100% (6/6), 658 bytes | 658.00 KiB/s, done.
```

```
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
```



remote: Resolving deltas: 100% (6/6) completed.
To https://github.com/brightgeevarghese/gitdemo
 179bc0b..929291c [new branch] HEAD -- Added a for loop
bright~\$

```
1 < public class HelloWorld{  
2 <     public static void main(String[] args) {  
3 <         System.out.println("Hello World!");  
4 <         System.out.println("Second line in our file");  
5 <         System.out.println("Somebody added: Third line for display");  
6 <         for (int i = 0; i < args.length; i++) {  
7 <             System.out.println(i);  
8 <         }  
9 <     }  
10    }
```

Exercise

Experimenting is fun 😊

Contribute to 3rd party github repository:

- You have been tasked to work on this repository:
<https://github.com/brightgeevarghese/novgitdemo.git>
- Fork the repository from my account to your GitHub account.
- Clone your forked copy from your GitHub account to your local machine
- Add your new folder(s)/file(s)
- Commit your change(s)/addition(s)
- Push your commit(s) to the remote repository named origin, which published it to your GitHub account
- Create/send a Pull Request from your GitHub account to the upstream remote repository.