

# Introduction to Git and GitHub

Managing your code: introducing *Git* - a friend for life

Thanks to all contributors:

Alison Pamment, Sam Pepler, Ag Stephens, Stephen Pascoe, Kevin Marsh, Anabelle Guillory, Graham Parton, Esther Conway, Eduardo Damasio Da Costa, Wendy Garland, Alan Iwi, Matt Pritchard and Tommy Godfrey.

# Contents

- What is version control?
- What are Git & GitHub?
- Nice features of GitHub
- What you need to know - the basic workflow

# Foreword

Git is easy to use but will take a bit of practice to get comfortable with. With that in mind:

- You only need to understand the basics
- Use the Cheatsheet: <https://education.github.com/git-cheat-sheet-education.pdf>
- The exercise following this presentation will get you set up to use Git and GitHub
- (We will encourage you to commit your work at the end of every exercise)
- Give it a try – you'll learn by *doing*

# What is a version control system (VCS)?

- Version control software keeps track of your changes
- Allows you to revert back to a previous points in history
- Uses freeze points which won't change
- Can manage contributions from multiple people
- Stores the full history of the things under version control including who did what, when?

# Why might you need VCS?

- Scientists are typically **required to publish data and code** (by their funders/institutions).
- Collaboration between scientists requires data-sharing; this implicitly relies upon **code-sharing**.
- There are **tools that make it easy** to record our changes, document our workflow and create “fixed” releases of our code at important steps along the way.
- Reduce errors and admin burden ("latest", "new2"...)
- Allows you test ideas with confidence, you can always go back.

# Why use Git and GitHub?

- There are a number of tools and web services that can do some, or all, of the tasks managed by Git/GitHub.
- They are as good, if not better, than alternatives.
- In your daily life as a scientist, we think that you will encounter these two tools most frequently.
- If you learn how to use them, then the principles will be transferrable to other tools (if needed).

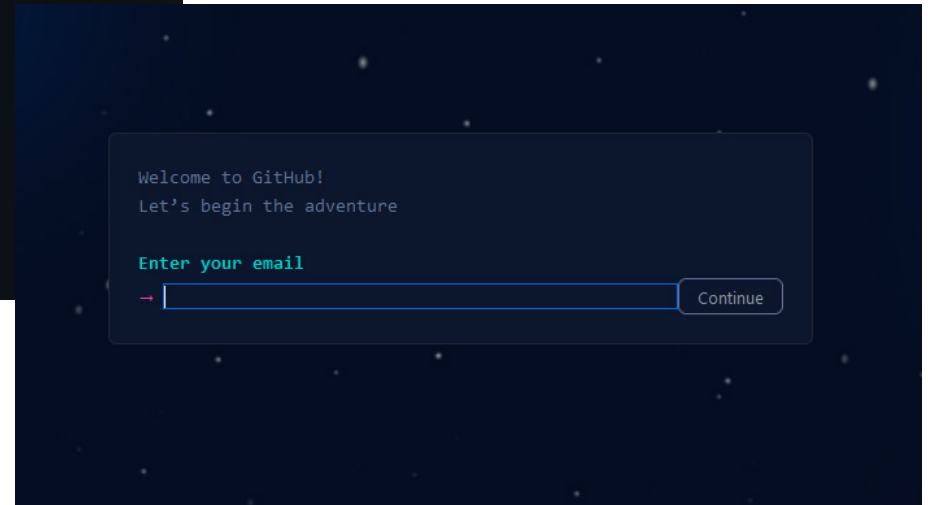
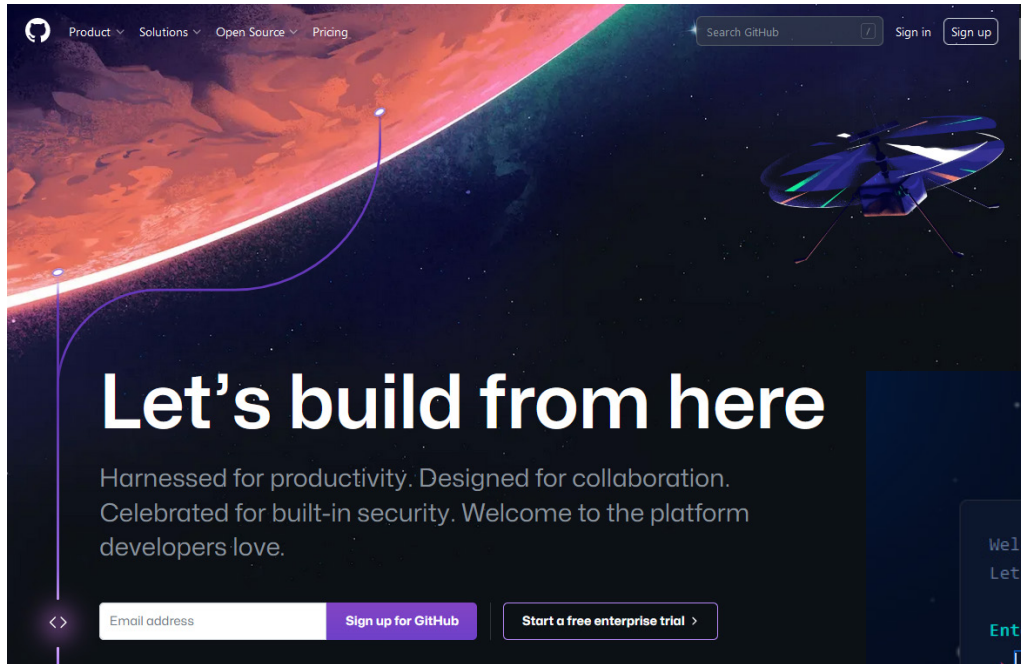
So, working on the premise that we accept that we need to know about, and use, version control...



We will use Git and GitHub



# Introducing GitHub



<https://github.com/>

# What is GitHub?

---

“A **web-based** Git repository **hosting service**”

GitHub allows you to:

- Share your repositories with others
- Access other users' repositories
- Store remote copies as a backup of your local repositories
- Add bug tracking, feature requests, wikis, ...

GitHub is **free** for most use cases

# Git vs GitHub

---

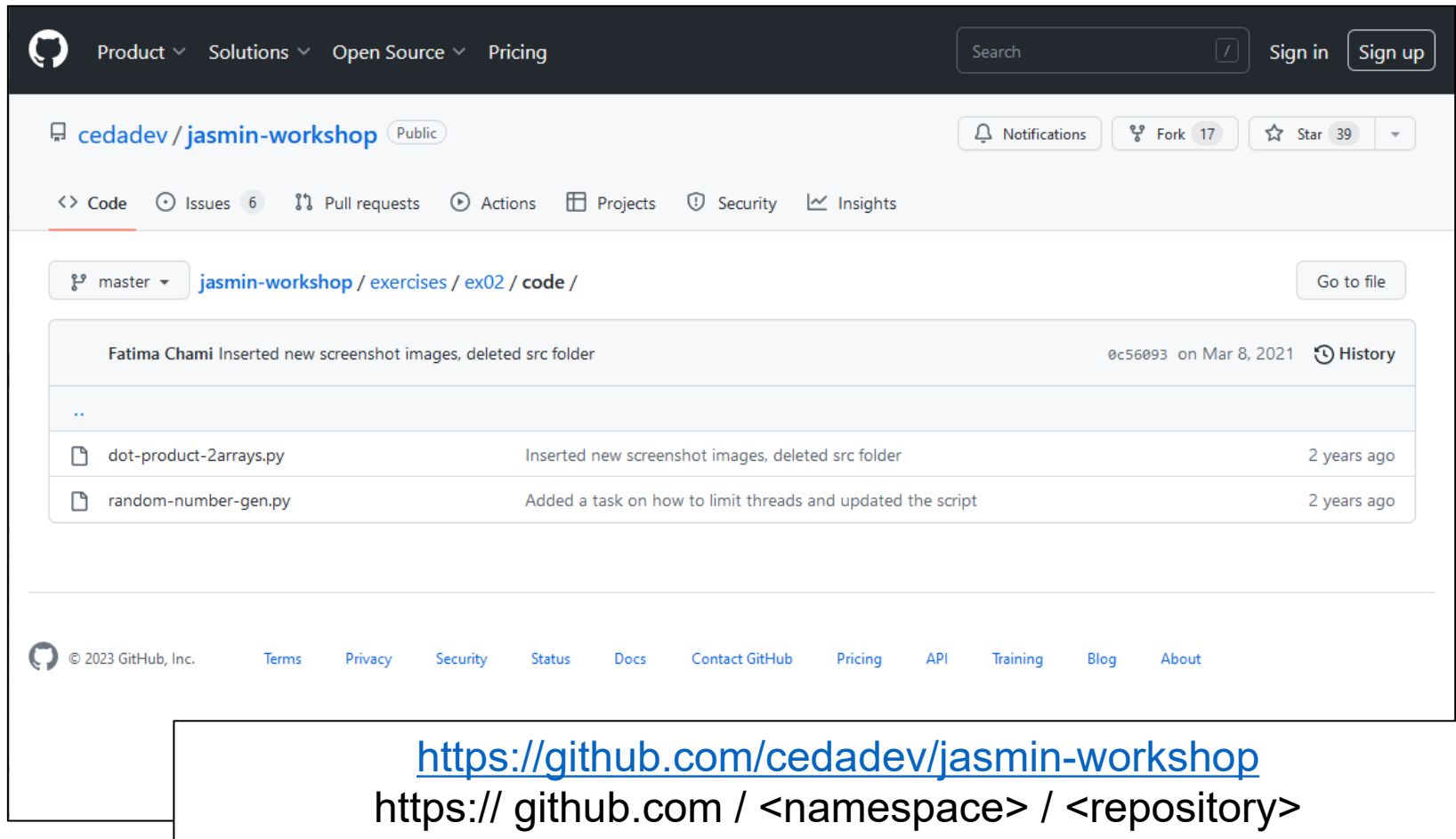
**Git** is a *revision control system*, a tool to manage your source code history.

**GitHub** is a *hosting service for Git repositories*.

**They are not the same thing.** **Git** is the **tool**, **GitHub** is a **web service**.

You **do not** need GitHub to use Git, but GitHub adds useful functionality.

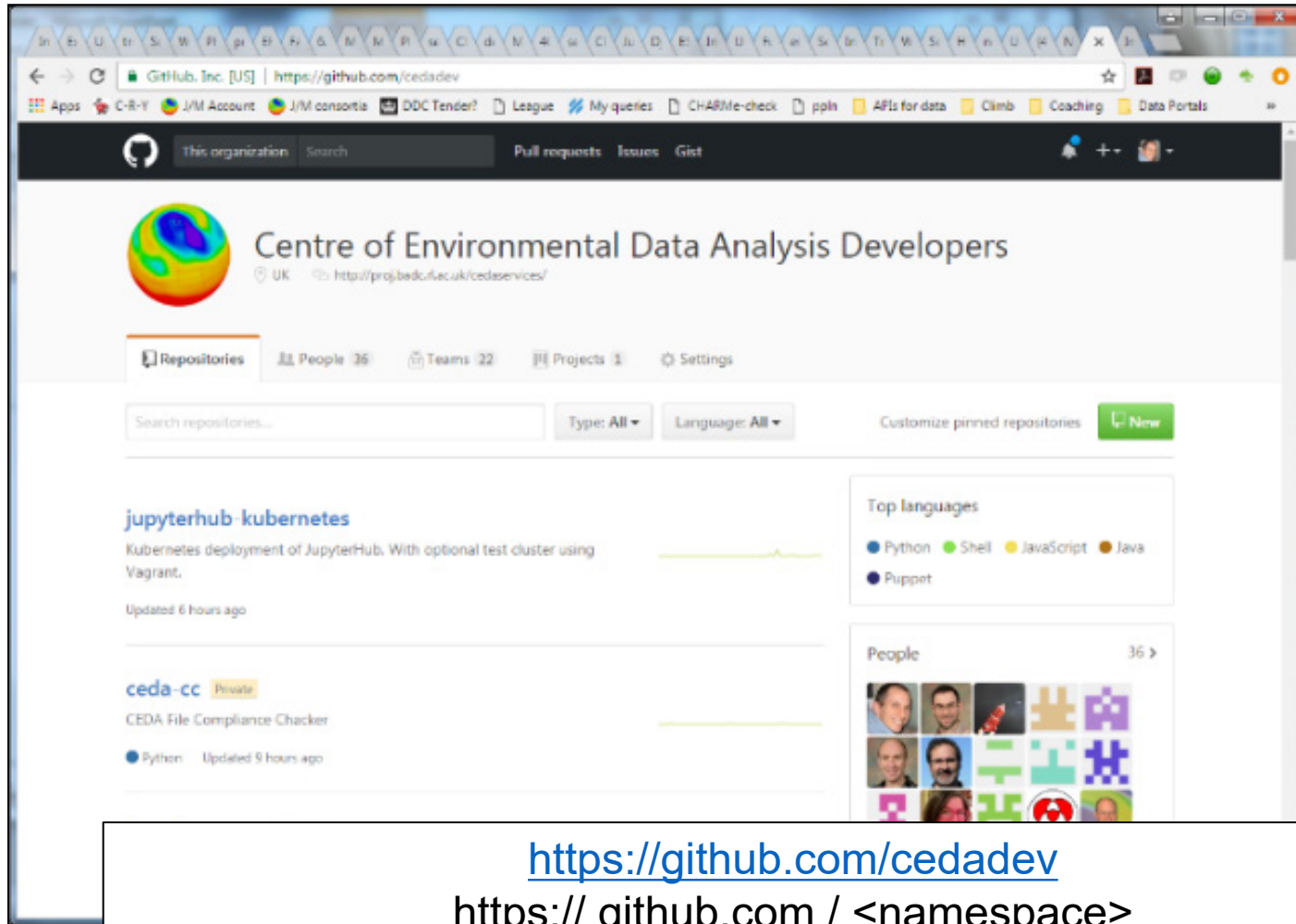
# GitHub: repositories (public or private)



The screenshot shows the GitHub interface for the repository `cedadev / jasmin-workshop`. The repository is public and has 17 forks and 39 stars. The navigation bar includes links for Product, Solutions, Open Source, and Pricing, along with a search bar and Sign in / Sign up buttons. The repository page shows the file structure with tabs for Code, Issues (6), Pull requests, Actions, Projects, Security, and Insights. The current view is the `code` tab, showing the file `random-number-gen.py` with a commit message "Added a task on how to limit threads and updated the script" from 2 years ago. The commit history shows a commit by Fatima Chami on Mar 8, 2021, with the message "Inserted new screenshot images, deleted src folder".

<https://github.com/cedadev/jasmin-workshop>  
`https:// github.com / <namespace> / <repository>`

# GitHub: organisations

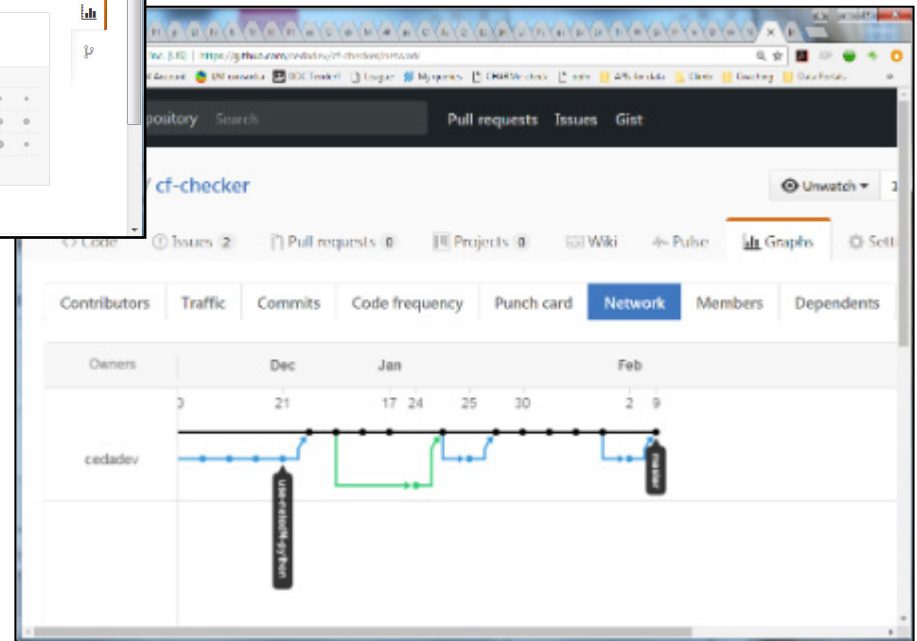
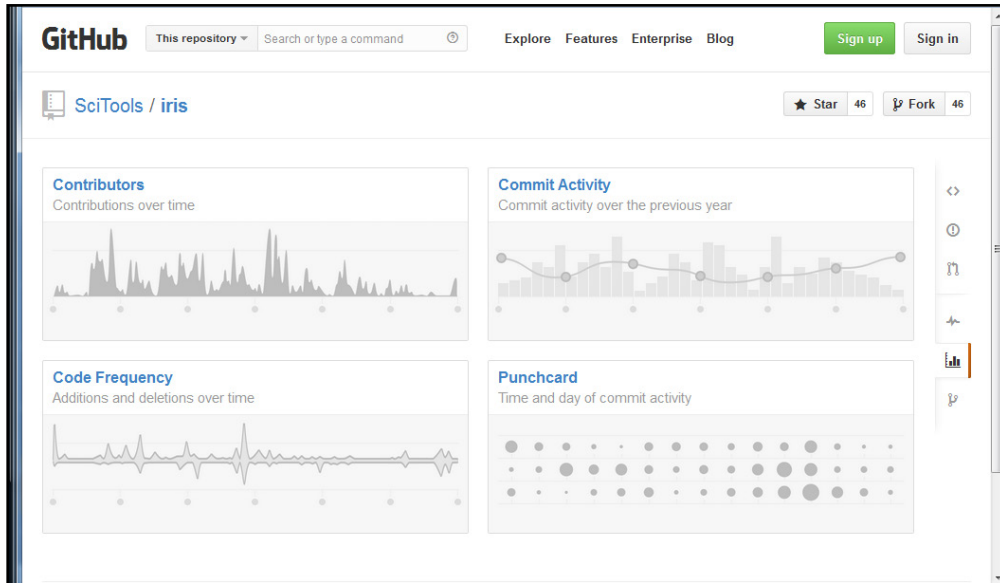


The screenshot shows the GitHub profile page for the organisation 'Centre of Environmental Data Analysis Developers' (cedadev). The page header includes the GitHub logo, a search bar, and navigation links for 'Pull requests', 'Issues', and 'Gist'. The organisation's profile section displays a colorful globe logo, the name 'Centre of Environmental Data Analysis Developers', a location tag for 'UK', and a website link 'http://proj.bedc.rl.ac.uk/cedaservices/'. Below this, tabs for 'Repositories', 'People', 'Teams', 'Projects', and 'Settings' are visible. The 'Repositories' tab is active, showing a search bar, filters for 'Type' and 'Language', and a 'Customize pinned repositories' button. Two repositories are listed: 'jupyterhub-kubernetes' (Kubernetes deployment of JupyterHub) and 'ceda-cc' (CEDA File Compliance Checker). A 'Top languages' section shows Python, Shell, JavaScript, Java, and Puppet. A 'People' section shows a grid of 36 team members.

<https://github.com/cedadev>

`https:// github.com / <namespace>`

# GitHub: collaboration (branch/fork)




# GitHub: Issue tracking

The screenshot displays the GitHub interface for the repository `cedadev / ceda_moles_django`, which is marked as `Private`. The repository has 11 watchers, 0 stars, and 0 forks. The `Issues` tab is selected, showing 104 issues. A search filter `is:issue is:open` is applied. The issues list includes:

- Can we embed schema.org tags into MOLES returned content to improve indexing by search engines** (#147, opened 5 days ago by philipkershaw, 1 comment)
- Fix database connection problem in parallel connections** (#145, opened 27 days ago by agstephens)
- Update MOLES PyDAP link from `dap.ceda.ac.uk` to `data.ceda.ac.uk`** (#143, opened on 14 Nov 2016 by gap736uk, labels: `Quick Item`, `urgent`, `User View`)
- Add cci-tagger to dependencies in MOLES deploy on ingest1** (#142, opened on 8 Nov 2016 by agstephens)
- Create tests: Validation of DRSDataset properties** (#141, opened on 7 Nov 2016 by agstephens)
- Export: templating issues to resolve** (#140, opened on 3 Nov 2016 by gap736uk, labels: `Export`, `high priority`, progress: 0 of 4, DCS checks: reco...)
- Adapt Obs Col and Obs templates to display selected GEMET theme(s) for the record** (label: `high priority`, `User View`)

# GitHub: history and change

```
6  cedamoles_app/admin_tools/integrity/routine_checks.py

@@ -41,10 +41,10 @@ def run_checks(self):

41 41
42 42     class ResultChecks(ChecksBase):
43 43
44 -     def check_internalPath(self):
44 +     def check_dataPath(self):
45 45         found = Counter()
46 46         for result in Result.objects.all():
47 -         path = result.internalPath
47 +         path = result.dataPath
48 48         found.update([path])
49 49
50 50         dupes = [(path, count) for (path, count) in found.items() if count > 1]
@@ -53,7 +53,7 @@ def check_internalPath(self):

53 53         for path, count in dupes:
54 54             print path, count
```





# GitHub: wikis

The screenshot shows a GitHub repository page for 'cedadev / ceda-fbs'. The repository has 13 pulls, 1 star, and 0 forks. The 'Wiki' tab is selected, showing a page titled 'Elasticsearch Filters' edited by 'jrainnie' on 19 Sep 2016. The page content includes a section for 'Boolean filter' and a JSON snippet for a 'bool' filter. A sidebar on the right lists other wiki pages: Home, Elasticsearch Filters, Elasticsearch Mapping, Example queries, and Example queries dumped from Sense interface. At the bottom, there is a section to 'Clone this wiki locally' with the URL 'https://github.com/cedadev/'.

This repository Search Pull requests Issues Gist

cedadev / ceda-fbs Unwatch 13 Star 1 Fork 0

Code Issues 1 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

## Elasticsearch Filters

jrainnie edited this page on 19 Sep 2016 · 1 revision

### Elasticsearch Filters

#### Boolean filter

The "bool" filter is used to combine other filters, and is composed of three sections;

```
{
  "bool" : {
    "must" : [],
    "should" : [],
    "must_not" : [],
  }
}
```

- "must" clause is equivalent of && (Logical AND) operator
- "should" clause is equivalent of || (Logical OR) operator

Pages 5

- Home
- Elasticsearch Filters
- Elasticsearch Mapping
- Example queries
- Example queries dumped from Sense interface

+ Add a custom sidebar

Clone this wiki locally

<https://github.com/cedadev/>

# GitHub does lots of funky things, but...

- To start with, we want to learn how to use it as a remote repository.
- We are going to concentrate on simply using Git.

# Where to start?

There are three different start points when using Git:

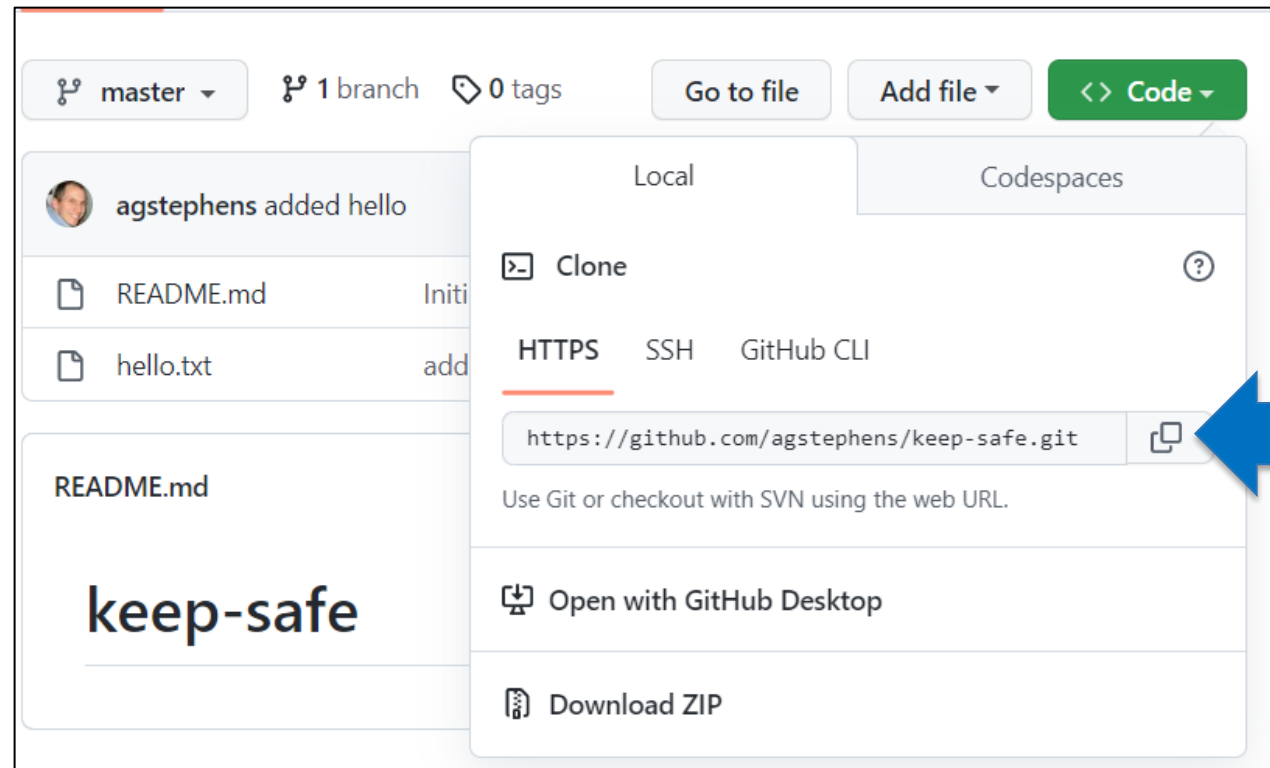
1. Clone an existing repository **from GitHub**
2. Create a new, empty repo and clone it **from GitHub**
3. Turn an existing **local directory** into a Git repo; it can contain files or be empty



Most  
common

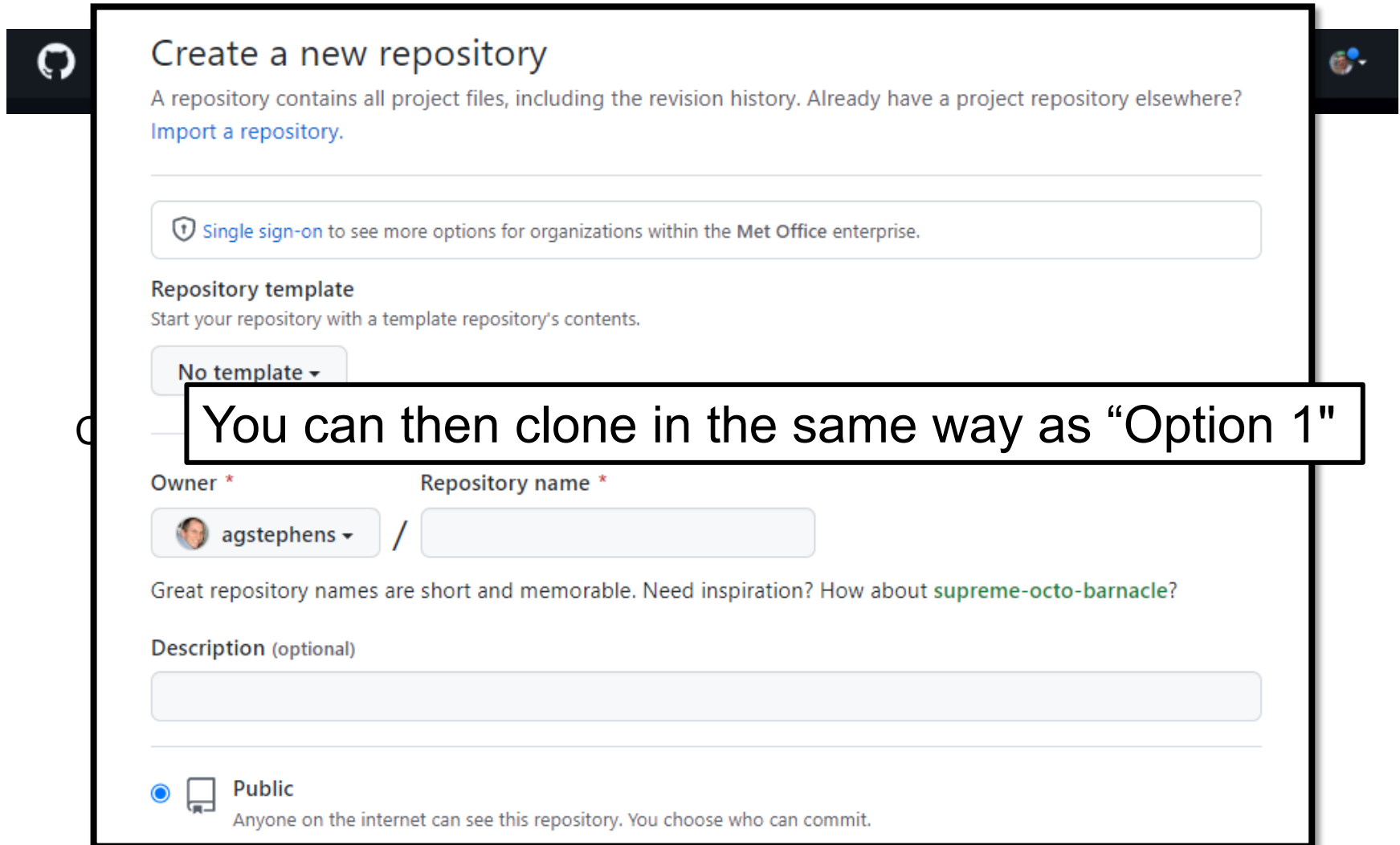
# Option 1: Clone Existing Repo

This makes a copy of a repository locally.




```
$ git clone https://github.com/agstephens/keep-safe
```

# Option 2: Create a repository on GitHub



**Create a new repository**


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

 [Single sign-on](#) to see more options for organizations within the Met Office enterprise.

**Repository template**  
Start your repository with a template repository's contents.


**No template** ▾

**Owner \*** **Repository name \***

 agstephens ▾ /

Great repository names are short and memorable. Need inspiration? How about [supreme-octo-barnacle?](#)

**Description (optional)**

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

You can then clone in the same way as “Option 1”

## Option 3: repository from an existing local directory

```
$ ls
x      y      z

$ git init
Initialized empty Git repository in:
/users/someone/test-package/.git/

$ git add .

$ git commit -m 'Initial commit from existing files'
[master (root-commit) 71ecfcf] Initial commit from existing files
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 x
create mode 100644 y
create mode 100644 z
```

# The basic workflow: Adding a file

1. Enter the repository directory:

```
$ cd keep-safe/
```

2. Create a new file:

```
$ echo "hello world" > hello.txt
```

3. Tell Git about the file:

```
$ git add hello.txt
```

ADD

4. Commit the file to the **local** Git repository:

```
$ git commit -m "added hello"
```

COMMIT

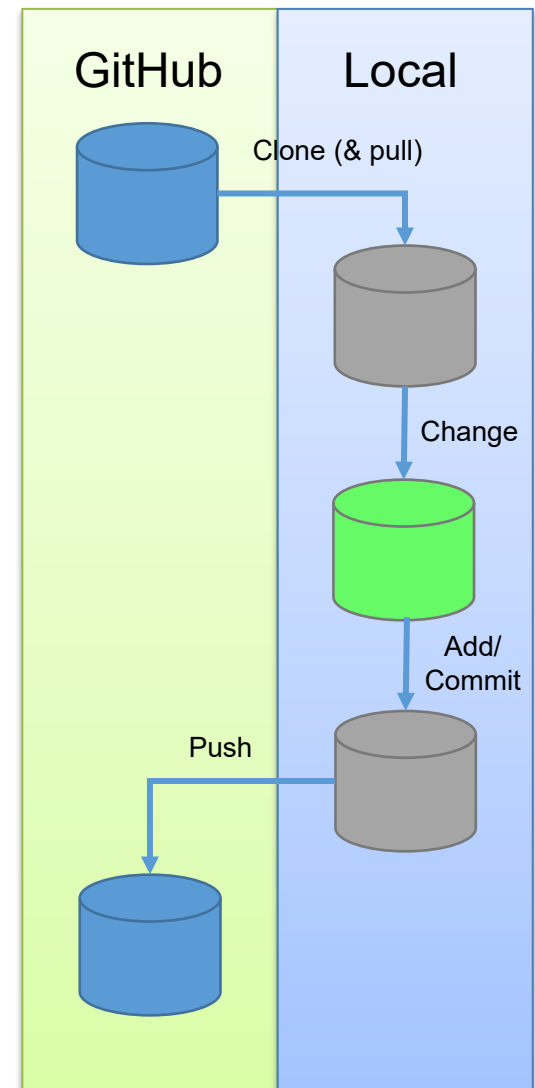
5. Push any updates to the **remote** GitHub repo:

```
$ git push
```

PUSH

# So, what just happened?

- We *cloned* the remote repository to our file system (using Option 1).
  - Now there are two identical copies of one repo.
- We *created* a new text file.
- We *added* and *committed* that new file to the local version of the repo.
- We used *push* to update the remote repo.





# Let's look on GitHub

Before...

main 1 branch 0 tags Go to file Add file <> Code

agstephens Create data.dat bed2d28 1 minute ago 6 commits

README.md	Initial commit	6 years ago
data.dat	Create data.dat	1 minute ago

After...

main 1 branch 0 tags Go to file Add file <> Code

agstephens added hello ba6aa17 1 minute ago 7 commits

README.md	Initial commit	6 years ago
data.dat	Create data.dat	4 minutes ago
hello.txt	added hello	1 minute ago

New!



Data Analysis  
JULIAN AND TALIPOLOU HARRIS  
NATURAL ENVIRONMENT RESEARCH COUNCIL



Atmospheric Science  
NATURAL ENVIRONMENT RESEARCH COUNCIL



Centre for Earth Observation  
NATURAL ENVIRONMENT RESEARCH COUNCIL

# Learn by using

- This stuff is hard to learn - we know that from experience.
- A presentation is quickly forgotten.
- So, we propose that you use Git/GitHub every day.
- We encourage you to create and update your own GitHub repository with files from exercises throughout our courses, and in your daily work!