

Chapter One: Getting Started

Getting started with a good plan is the most challenging part of building software. This chapter covers how to do that.

Effective Async Technical Discussions

What makes a useful technical discussion? Several techniques significantly enhance a professional conversation around technical details.

Here is a screencast on how to create a useful technical discussion.

Video Link: <https://www.youtube.com/watch?v=gcbjlq3B4cw>³

Reproducible Code

If a discussion involves code, the ability to reproduce the system significantly enhances the conversation. The source code that is shared or discussed must run smoothly. If not, then it could add zero or even negative value to sharing it. Hosted `git` and hosted Jupyter Notebooks⁴ are two common ways to solve this problem.

Hosted `git`

Three main versions of hosted `git` are: bitbucket⁵, github⁶ and GitLab⁷. They all provide ways to share and reproduce code. This code can share within the context of a software development project, and it can also share in an async based discussion like chat.

Let's focus on Github, the most commonly encountered of these options. There are two main ways to share code with others. One method is to create a public repo.⁸ and share code and/or markdown⁹ files. One nice side effect of markdown files is that they can also serve out via webpages through GitHub Pages¹⁰ or through a blog engine like Hugo¹¹, which can build pages <1 ms per page.

Another powerful feature of Github is a gist¹². What is particularly useful about a gist is that it can be shared with syntax highlighting and formatting. Here are the steps:

³<https://www.youtube.com/watch?v=gcbjlq3B4cw>

⁴<https://jupyter.org/>

⁵<https://bitbucket.org/product>

⁶<https://github.com/>

⁷<https://about.gitlab.com/>

⁸<https://help.github.com/en/github/administering-a-repository/setting-repository-visibility>

⁹<https://guides.github.com/features/mastering-markdown/>

¹⁰<https://pages.github.com/>

¹¹<https://gohugo.io/>




¹²<https://gist.github.com/>


1. Create gist


GitHub Gist


Search...


All gists Back to GitHub

 [cloudfordata.py](#)
This is a code snippet l...

 [paiml-labs-news...](#)
No description.

 [preface.asciidoc](#)
No description.

 [pg431.py](#)
No description.


[See all of your gists](#)

This is a code snippet!


cloudfordata.py

Spaces 2 No wrap

```
1 def hello():
2     print("Hello Cloud")
```

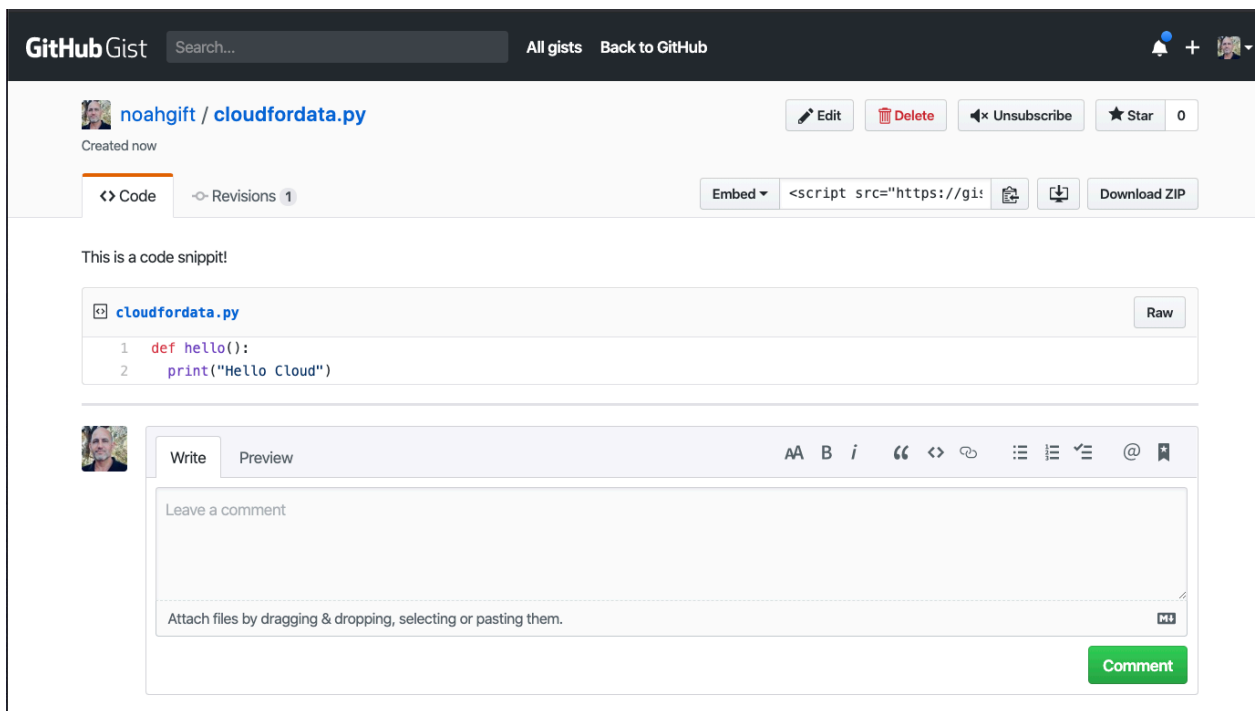


Add file

 Create secret gist Create public gist

creategist

2. Share gist



sharegist

3. Here is the URL to share:

Gist Example¹³

Many chat programs will automatically render out the code snippet.

Hosted Jupyter Notebooks

In theory, Jupyter Notebooks solve a massive problem in creating reproducible code, but it needs some help in practice. A fundamental limitation of Jupyter is the Python packaging environment. It is a helpless victim to the untamed complexity of the underlying operating system.

Fortunately, there is an easy solution. Jupyter notebooks that have a portable runtime are the reproducible ones. Portable runtimes include docker¹⁴ and colab¹⁵. Docker format files can specify what the runtime should be like, including the packages that need for installation.

One example of a hosted runtime can be found in this project: Container Microservices project¹⁶.

For a user to recreate the code and run it locally, they can do the following:

¹³<https://gist.github.com/noahgift/b6eec243c70ba4f71033954c4da75dd3>

¹⁴<https://www.docker.com/>

¹⁵<https://colab.research.google.com/>

¹⁶<https://github.com/noahgift/container-revolution-devops-microservices>

```
1  #!/usr/bin/env bash
2
3  # Build image
4  docker build --tag=flasksklearn .
5
6  # List docker images
7  docker image ls
8
9  # Run flask app
10 docker run -p 8000:80 flasksklearn
```

This approach is optimized for deployment and has some advantages for communication focused on deploying software. A second approach is the `colab` approach. In this `colab` example¹⁷ the notebook note only has the complete code, but with a click of the “Open in Colab” button, a user can completely reproduce what was shared.

¹⁷https://github.com/noahgift/functional_intro_to_python/blob/master/Public_Master_SafariOnline_Day1_Part1.ipynb

sharecolab

Audio, Video and Images

Adding audio, video, and images can significantly enhance a technical discussion.

Sharing images

One simple “hack” for sharing images is to use Github issues. Here is an example of this in action¹⁸.

¹⁸<https://github.com/noahgift/cloud-data-analysis-at-scale/issues/1>

sharehack

Screencasts

Doing a quick screencast can boost a discussion value. Here is a screencast of creating an AWS Lambda function; this is an excellent example of a short demo video.

Video Link: <https://www.youtube.com/watch?v=AlRUeNFuObk>¹⁹

Here is another screencast on what to consider when creating a technical video.

Video Link: <https://www.youtube.com/watch?v=upQEE9jwI3M>²⁰

You can create screencasts quickly using the software you probably already have on your machine. Options include: Zoom²¹, QuickTime Player²² and Camtasia²³.

Produce Once, Reuse Many

One thing to keep in mind with technical discussion is *produce once, reuse many*. There are many outlets for a professional review, including classroom discussions, work discussions, books you are

¹⁹<https://www.youtube.com/watch?v=AlRUeNFuObk>

²⁰<https://www.youtube.com/watch?v=upQEE9jwI3M>

²¹<https://zoom.us/>

²²<https://support.apple.com/guide/quicktime-player/record-your-screen-qt97b08e666/mac>

²³<https://www.techsmith.com/video-editor.html>

writing, or software projects you are contributing.

You can use these notes and code samples for years or even the rest of your life if you produce high-quality technical notes. Why not create high-quality comments so you can “reuse” these assets in many ways.

Technical discussions as a form of active learning

One substantial advantage of technical discussions is they serve as a form of active learning. Writing software in a professional setting with modern software development practices often involves many team interactions (i.e. pull requests²⁴). This is a form of “super-charged” learning that enables software engineers to learn at an extraordinary pace.

Effective Async Technical Discussions Conclusions

Building software or doing data science is not about setting aside a session and building something to stop. It is an iterative form of group communication. In turning in homework assignments or finishing a commercial project ticket, the conversation is where the most value occurs versus just the raw software code.

²⁴<https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests>

²⁵<https://slack.com/>

²⁶<https://piazza.com/>

²⁷<https://canvas.instructure.com/>