



# Getting started with Git and GitHub for the reluctant IT Pro

Thomas Vochten

# Agenda

1. Introduction to git
2. The tools you need to get started
3. Getting basic stuff done with git
4. Where to go next?
5. Contributing to open source



# Thomas Vochten

@thomasvochten



<https://thomasvochten.com>



[mail@thomasvochten.com](mailto:mail@thomasvochten.com)



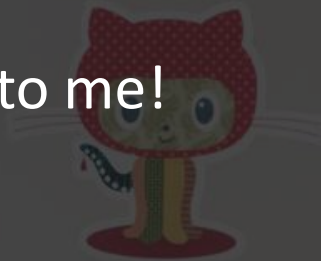
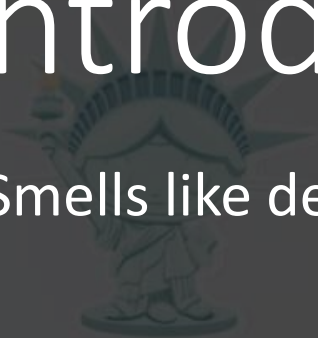
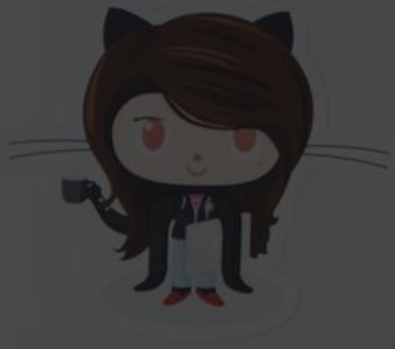
Technology Evangelist   

#Microsoft365 #Azure #CommunityRocks



# 1 | Introduction to git

Smells like developer stuff to me!



# “Developer” good practices

- Simplicity
- Portability
- Scalability
- Reusability
- Predictability
- Maintainability
- Efficiency
- ...

# “Developer” Acronym Soup



**DRY, KISS & YAGNI**

Don't Repeat Yourself

Keep It Stupid Simple

You Ain't Gonna Need It

# DTSTTCPW

Do The Simplest Thing That Could Possibly Work

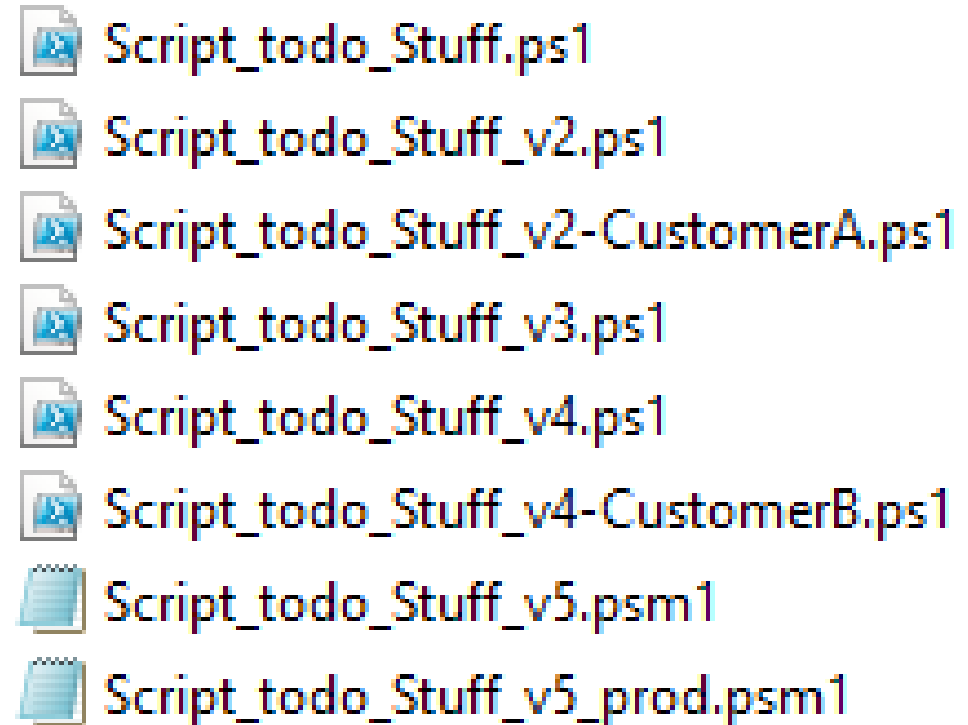
The image features two puppets against a dark red background with vertical wooden slats. On the left, a puppet of Albert Einstein with wild white hair and a mustache is shown from the chest up, looking down with a somber expression. On the right, a puppet of a man with a large, bulbous nose and a wide, toothy grin is shown from the chest up, looking towards the left. The overall lighting is dim, creating a somber and slightly humorous atmosphere.

Why exactly would I care?











# Where and how do you store your stuff?

- Computer
- File share
- OneNote
- Cloud storage
- ...



A list of PowerShell script files, each preceded by a small icon representing the file type. The icons are blue for .ps1 files and a blue notepad for .psm1 files.

-  Script\_todo\_Stuff.ps1
-  Script\_todo\_Stuff\_v2.ps1
-  Script\_todo\_Stuff\_v2-CustomerA.ps1
-  Script\_todo\_Stuff\_v3.ps1
-  Script\_todo\_Stuff\_v4.ps1
-  Script\_todo\_Stuff\_v4-CustomerB.ps1
-  Script\_todo\_Stuff\_v5.psm1
-  Script\_todo\_Stuff\_v5\_prod.psm1



# Source control to the rescue

- Like versioning in SharePoint, but different!
- Keep different versions of your files
- Ability to go back if you messed up
- Work together with other people
- Work on new stuff while keeping the existing files intact
- Make sure your new stuff stays up to date when you change the existing files
- ...



Git is a free and open source distributed  
version control system

<https://thvo.me/git>



# What about GitHub then?

- Hosted git
- Acquired by Microsoft in 2018
- The de facto standard

## Some alternatives



Bitbucket



Gitlab



Azure Repos

THIS IS GIT. IT TRACKS COLLABORATIVE WORK  
ON PROJECTS THROUGH A BEAUTIFUL  
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL  
COMMANDS AND TYPE THEM TO SYNC UP.  
IF YOU GET ERRORS, SAVE YOUR WORK  
ELSEWHERE, DELETE THE PROJECT,  
AND DOWNLOAD A FRESH COPY.





<https://ohshitgit.com/>

# Oh shit, git!

Git is hard: screwing up is easy, and figuring out how to fix your mistakes is fucking impossible. Git documentation has this chicken and egg problem where you can't search for how to get yourself out of a mess, unless you *already know the name of the thing you need to know about* in order to fix your problem.

So here are some bad situations I've gotten myself into, and how I eventually got myself out of them *in plain english*.\*

## Oh shit, I did something terribly wrong, please tell me git has a magic time machine!?!

```
git reflog
# you will see a list of every thing you've done in git, across all br
# each one has an index HEAD@{index}
# find the one before you broke everything
git reset HEAD@{index}
# magic time machine
```

You can use this to get back stuff you accidentally deleted, or just to remove some stuff you tried that broke the repo, or to recover after a bad merge, or just to go back to a time when things actually worked. I use `reflog` A LOT. Mega hat tip to the many many many many many people who suggested adding it!

## Oh shit, I committed and immediately realized I need to make one small change!

# Dangit, git!

Git is hard: screwing up is easy, and figuring out how to fix your mistakes is nigh on impossible. Git documentation has this chicken and egg problem where you can't search for how to get yourself out of a mess, unless you *already know the name of the thing you need to know about* in order to fix your problem.

So here are some bad situations I've gotten myself into, and how I eventually got myself out of them *in plain english*.\*

## Dangit, I did something terribly wrong, please tell me git has a magic time machine!?!

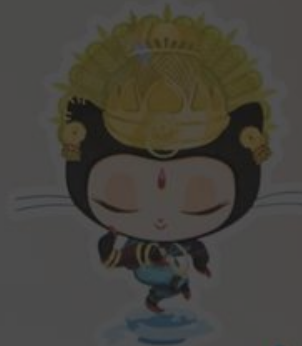
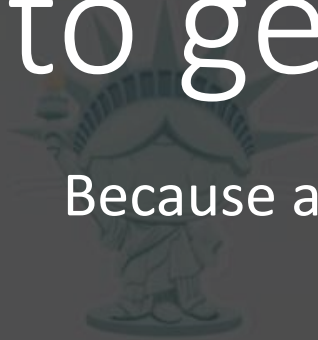
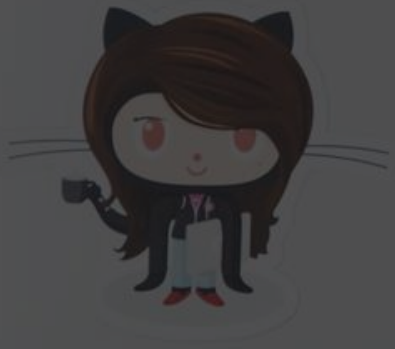
```
git reflog
# you will see a list of every thing you've done in git, across all br
# each one has an index HEAD@{index}
# find the one before you broke everything
git reset HEAD@{index}
# magic time machine
```

You can use this to get back stuff you accidentally deleted, or just to remove some stuff you tried that broke the repo, or to recover after a bad merge, or just to go back to a time when things actually worked. I use `reflog` A LOT. Mega hat tip to the many many many many many people who suggested adding it!

## Dangit, I committed and immediately realized I need to make one small change!

## 2 | The setup you need to get started

Because all geeks love tools

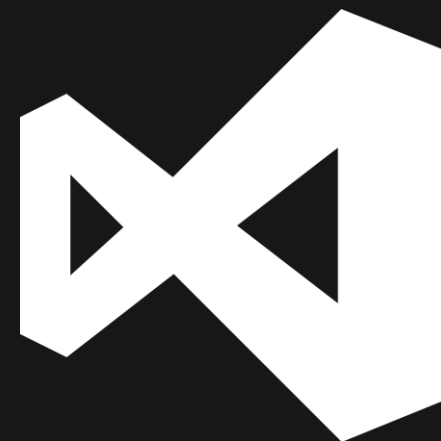




git



Windows Terminal



Visual Studio Code

# Upgrade your command prompt

```
C:\Dev\MyRepository > ps main &
```

```
C:\Dev\MyRepository > ps main & +0 ~1 -0 !
```



# Upgrade your command prompt

- Install Windows Terminal
- Install a Powerline-capable font such as “`Cascadia Code PL`”

<https://thvo.me/cascadia>

- Setup git & Powerline integration in PowerShell

```
Install-module posh-git  
Install-Module oh-my-posh  
Set-Theme Paradox
```

- Modify your Windows Terminal profile settings to use the font

```
"fontFace": "Cascadia Code PL"
```

# Integration with Visual Studio Code



- Supports the most common git commands by default
- Use the command line for more advanced scenarios
- Useful extensions:



GitLens

<https://thvo.me/gitlens>



Git History

<https://thvo.me/githistory>

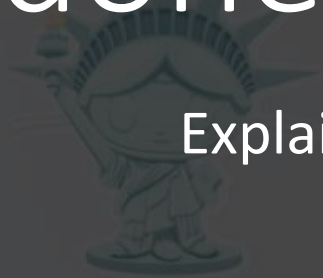
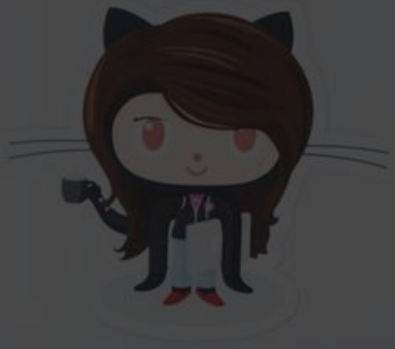


GitHub Pull Requests & Issues

<https://thvo.me/vscodegithub>

# 3 | Getting basic stuff done with git

Explain it like I'm 5



# Essential Lingo

- Just “downloading” code from the internet (**cloning**)
- Creating your own version of existing code (**forking**)
- Creating a separate area to work on stuff (**branching**)
- Checking in your changes (**committing**)
- Move your changes to a remote repository (**pushing**)
- Getting changes from the remote repository (**pulling**)
- Integrating changes (**merging**)

① Many practices are just conventions

# Demo 1 - Common Operations

git clone, git status, git commit, git add, git push, git pull



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

# Demo 2 - Working with a Remote Repository in GitHub

git init, git remote

## Quick setup — if you've done this kind of thing before



Set up in Desktop

or

HTTPS

SSH

`https://github.com/thomasvochten/M365UK.git`



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

## ...or create a new repository on the command line

```
echo "# M365UK" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/thomasvochten/M365UK.git
git push -u origin master
```



## ...or push an existing repository from the command line

```
git remote add origin https://github.com/thomasvochten/M365UK.git
git push -u origin master
```



## ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

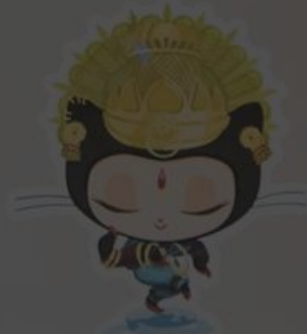
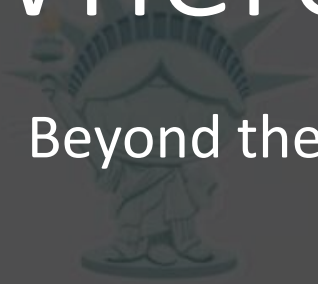
Import code

# Demo 3 - VSCode Integration

native support, extensions

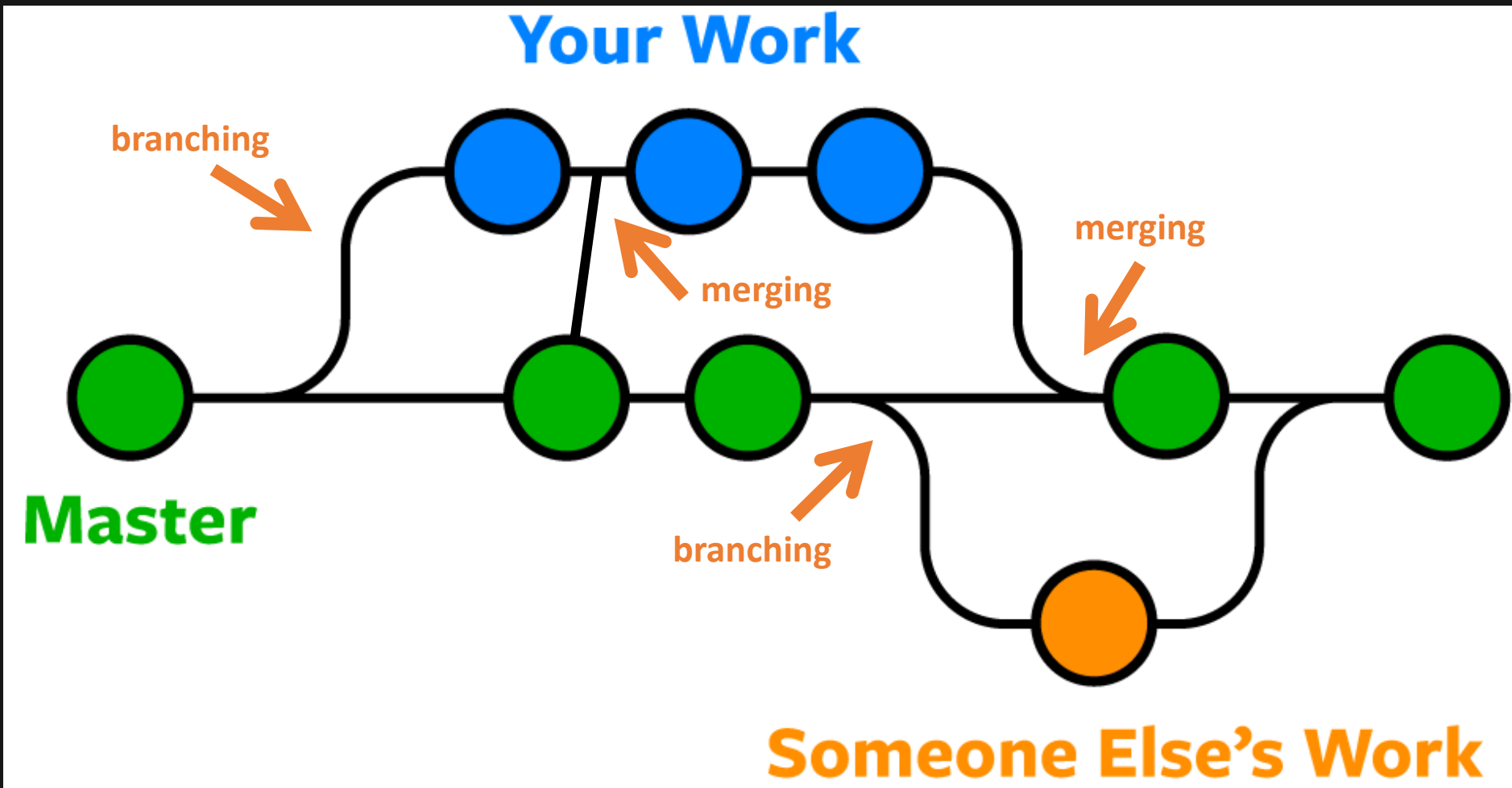
# 4 | Where to go next?

Beyond the next - next - finish





# Branching & Merging

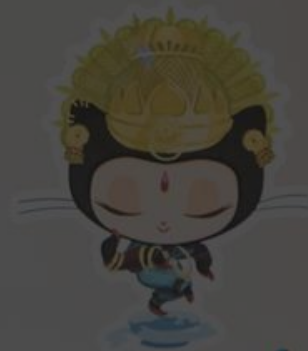
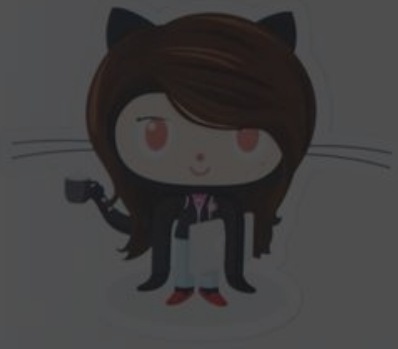


# Demo 4 – Branching & merging

git branch, git checkout, git merge

# 5 | Contributing to open source

Power to the people!



# Sharing is caring!

Everyone can contribute.

Share the things you're proud of:


- PowerShell scripts & modules
- Samples and tutorials
- Documentation
- ...

# Contributing to docs.microsoft.com

Set up development environment

Bookmark Feedback **Edit** Share

## Set up your SharePoint Framework development environment

07/31/2020 • 5 minutes to read •  +19

You can use Visual Studio or your own custom development environment to build SharePoint Framework solutions. You can use a Mac, PC, or Linux environment as well.


### Note

Before following the steps in this article, be sure to [Set up your Microsoft 365 tenant](#).

You can also follow these steps by watching this video on the SharePoint PnP YouTube Channel:



Is this page helpful?

 Yes  No

### In this article

[Install Node.js](#)

[Install a code editor](#)

[Install development toolchain prerequisites](#)

[Install a modern web browser](#)

[Trusting the self-signed developer certificate](#)


[Optional tools](#)

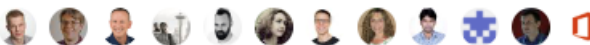
[SPFx & SharePoint Server 2016](#)

[Next steps](#)

[Troubleshooting](#)




master [sp-dev-docs / docs / spfx / set-up-your-development-environment.md](#) [Go to file](#) [...](#)

 heythisispaul Merge branch 'master' into fix/trusting-self-signed ✓ Latest commit bb13e2d 12 days ago [History](#)

[25 contributors](#)  +13

149 lines (94 sloc) | 8.91 KB

[Edit the file in your fork of this project](#)


[Raw](#) [Blame](#)   


	title	description	ms.date	ms.prod	localization_priority	ms.custom
--	-------	-------------	---------	---------	-----------------------	-----------

You're making changes in a project you don't have write access to. Submitting a change will write it to a new branch in your fork thomasvochten/sp-dev-docs, so you can send a pull request.


[sp-dev-docs / docs / spfx /](#) [set-up-your-developmei](#) [Cancel](#)

<> Edit file

 Preview changes

Spaces 


2 

Soft wrap 

```
1 ---
2 title: Set up your SharePoint Framework development environment
3 description: Use Visual Studio or your own custom development environment to build SharePoint Framework solutions. You can use a Mac, PC, or Linux.
```

# Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

 base repository: SharePoint/sp-dev-docs ▾

base: master ▾

←

head repository: thomasvochten/sp-dev-docs ▾

**compare: patch-2 ▾**

✓ **Able to merge.** These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

🔗 1 commit

± 1 file changed

💬 0 comments

👤 1 contributor

Commits on Aug 12, 2020

  Typo correction

Unverified 65ecef1

± Showing 1 changed file with 1 addition and 1 deletion.

Unified Split

2 docs/spfx/set-up-your-development-environment.md

<> 📄 ...

↑ @@ -4,7 +4,7 @@ description: Use Visual Studio or your own custom development environment to bui



# Typical workflow when contributing

- Fork the repo you want to contribute to
- Create a branch for your change
- Make & test your changes
- Create a pull request
- Wait for the pull request to be accepted
- Remove your fork




 Learn about pull requests etiquette...

# Key takeaways

- Getting started with git is relatively simple
- The learning curve is quite dramatic once you go beyond the basics
- Don't be afraid of the command line, but also:
- Take advantage of tools like VSCode to help you
- Commit as often as you can.  
Don't forget to pull regularly too.
- Work with branches
- Learn how about pull requests to contribute back

In case of fire



1. git commit 
2. git push 
3. leave building 

# Additional resources

Video series by Scott Hanselman

<https://thvo.me/scotthagit>

Git documentation

<https://thvo.me/gitdocs>

Set up Powerline in Windows Terminal

<https://thvo.me/wintermpowerline>

@thomasvochten

THANK  
YOU



Download session materials