Modern Project Management

Brendan Bartels
Why do we care?

• Makes your team more efficient
• Makes your journey less stressful
• Makes your successes more accessible
You Need To Manage

- Tasks
- Communication Channels
- Files
You Need To Manage

- Tasks
- Communication Channels
- Files
You Need To Manage

- Tasks
- Communication Channels
- Files
What is Git?

- Tool that allows you to **track file changes** over time
  - Like a diary
- [https://git-scm.com/](https://git-scm.com/)
What is Gitlab?

- A web application that allows you to
  - integrate with Git
  - organize tasks
  - track work
  - document changes
  - in the process you get a rich record of all your work
    - (for free)

- https://about.gitlab.com/
How To Manage

- Tasks
- Communication Channels
- Files
How To Manage

• Tasks
• Communication Channels
• Files
Managing Tasks

• Why do you have to care?
  • You will **not agree** on the tasks that need to be done
  • People who don’t know the tasks **won’t work well**
  • If you don’t complete the tasks, **you don’t succeed**
Managing Tasks

• How do we manage tasks?

  • Agree on the tasks

    • This is the responsibility of the team leader

  • Write the tasks down

    • Make the tasks **easily accessible** to everyone on the team
Managing Tasks

• What tools should we use?
  • Trello is good, but…
  • Use Gitlab Issues
    • Tasks correlate nicely with Gitlab Issues
    • Gitlab also has a Trello-like board to render Issues as cards
<table>
<thead>
<tr>
<th>Issue Number</th>
<th>Title</th>
<th>Assigned To</th>
<th>Date of Last Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>#24</td>
<td>MicroCART: Documentation (Dr. Jones Feedback)</td>
<td>phjones</td>
<td>updated a week ago</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#23</td>
<td>Rotate filtered flow velocity</td>
<td>dawehr</td>
<td>updated a week ago</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#22</td>
<td>Implement angle unwrapping for yaw</td>
<td>dawehr</td>
<td>updated a week ago</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#21</td>
<td>Remove read_kill and associated functions in favor of ROC functions</td>
<td>bbartels</td>
<td>updated a week ago</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#20</td>
<td>Upgrade our unit testing framework</td>
<td>bbartels</td>
<td>updated a week ago</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#19</td>
<td>Divide quad_main of quad_app into smaller functions (setup, loop,</td>
<td>bbartels</td>
<td>updated a week ago</td>
</tr>
<tr>
<td></td>
<td>teardown)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#18</td>
<td>Blink LED when sending logs</td>
<td>bbartels</td>
<td>updated a week ago</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#17</td>
<td>Stop setting the hardware interfaces as global variables</td>
<td>bbartels</td>
<td>updated a week ago</td>
</tr>
</tbody>
</table>
Add Valgrind to Continuous Integration Tests

Valgrind is a tool that can detect:

- Invalid memory reads and writes
- Use of uninitialized memory
- Memory leaks
- And more!

Adding it to our automated tests will help verify that new code is memory safe, a common source of problems in C.

Valgrind can be launched by simply calling:

```bash
valgrind --leak-check=full --log-file=./valgrind_out.bin/virt-quad
```

The leak-check flag enables memory leak checking, and the log-file flag feeds all output to a log file that can be analyzed.

@bbartels mentioned in issue #11 (closed) 4 months ago

@bbartels mentioned in issue #19 4 months ago

@bbartels mentioned in merge request #13 (merged) 4 months ago

@bbartels commented 4 months ago

Resolved in #13 (merged).

Edited about an hour ago by @bbartels
- MicroCART: Documentation (Dr. Jones Feedback) #24
  - Documentation

- Rotate filtered flow velocity #23
  - Quad

- Implement angle unwrapping for yaw #22
  - Quad

- Remove read, kill and associated functions in favor of ROC functions #21
  - Quad

- Upgrade our unit testing framework #20

- Stop setting the hardware interfaces as global variables #17
  - Quad
  - Tech Debt

- Add Valgrind to Continuous Integration Tests #15

- quad "make deep-clean" should just be "make clean" #12
  - Cleanup

- Quad functional tests should fail explicitly if virt-quad segfaults #11
  - Tech Debt

- Add -Wall flag to quad builds #10

- Delete Unused Files in quad_app #10
  - Cleanup
  - Tech Debt
Managing Tasks

• Examples:

  • https://git.ece.iastate.edu/danc/MicroCART/boards?=
Managing Tasks
FAQ

• How do I add pictures/code/links to my Gitlab Issues?

  • All Gitlab posts **support Markdown**, which is a simple language you can use to format your posts, make links, and insert pictures (and even videos).

  • [https://docs.gitlab.com/ee/user/markdown.html](https://docs.gitlab.com/ee/user/markdown.html)
How To Manage

• Tasks

• Communication Channels

• Files
How To Manage

• Tasks
• Communication Channels
• Files
Managing Communication Channels

• Why do we care?

• You will experience setbacks in your project if you lose technical details of your project.

• You will lose technical details if you use the wrong communication channel.
Managing Communication Channels

- What communication channels should we use?
  - When you aren’t talking about project tasks, use something that prioritizes speed.
    - You’ll probably use Slack.
  - When you are talking about project tasks, use something that makes it easy to find those details later.
Managing Communication Channels

“The event horizon of the Slack black hole”

Your team has more than 10,000 messages in its archive, so although there are older messages than are shown below, you can't see them. Find out more about upgrading your team.
Managing Communication Channels

- Use Gitlab issues when talking about project tasks:
  - Post questions on relevant Issue threads
  - Post updates on relevant Issue threads
  - “Mention” any team members that you want to contribute the conversation
    - @netid
“mention” team members that you want to join the discussion in any post, and they will get an email notification.
Managing Communication Channels

- Other Examples:
  - https://git.ece.iastate.edu/danc/MicroCART/issues/1
  - https://github.com/jOOQ/jOOQ/issues/2536
Managing Communication Channels

- A concession:
  - Slack is easier to use than discussion boards for technical conversations that need to happen quickly
  - Just copy and paste the discussion into a relevant Issue thread when the conversation is finished
Use a normalized PWM value

In the quad app, we are using a legacy unit of 1e-3 seconds to represent the PWM pulse width. This is somewhat platform dependent and just not very helpful at the application layer. We should replace this unit with something like Duty cycle, a float from 0 to 1.

The following places will likely need to be updated:

- the hardware/application interface
- the control algorithm uses this unit, so all control parameters will need to be updated to use the new unit

Edit

Instead of duty cycle, use a normalized value from (0,1) to represent the active region of the ESCs (1ms-2ms pulse width). In our specific case, we are operating at a frequency of 480 Hz, so our duty cycle would range from 40%-80% in the hardware layer.

Related issues 0 0 +

This text was copied from a Slack conversation
Managing Communication Channels

- A word of caution
  - Don’t overuse private communication channels
How To Manage

• Tasks

• Communication Channels

• Files
How To Manage

• Tasks

• Communication Channels

• Files
Managing Files

Why do we care?

- People need to be able to make changes locally (important for software changes), so we need a strategy to coordinate distributed changes

- People need to be able to find files, so we need as few locations to store files as possible
Managing Files

- How to best manage digital assets?
  - For files that require real-time collaboration
    - Use Google Docs OR Cybox + Office 365 integration
  - For everything else
    - Use Git + Branch-Review-Merge workflow
Managing Files

• Managing “everything else”
  • Use Git
  • Use a Branch-Review-Merge workflow
Managing Files

- Managing “everything else”
  - Use Git
  - Use a Branch-Review-Merge workflow
Intro to Git

- Git is a timeline of snapshots
  - snapshots = commits
  - timeline = branch

- Initial commit
- Add README
- Implement framework for...

https://git.ece.iastate.edu/danc/MicroCART/blob/master/documentation/how_to_use_git.md#how-do-i-make-changes
To start tracking a project with git, enter the directory and use the “git init” command.
Hint: Use “git status” to get a hint about what you should do.
hello_world (master)$ git add .
hello_world (master)$ git status
On branch master

Initial commit

Changes to be committed:
(use "git rm --cached <file>..." to unstage)

new file:   .gitignore
new file:   Cargo.lock
new file:   Cargo.toml
new file:   src/main.rs
hello_world (master)$ git commit -m "Initial commit"
[master (root-commit) 85762b0] Initial commit
  4 files changed, 14 insertions(+)
create mode 100644 .gitignore
create mode 100644 Cargo.lock
create mode 100644 Cargo.toml
create mode 100644 src/main.rs
master

[85762b0] Initial commit
hello_world (master)$ emacs src/main.rs
hello_world (master)$ git status

On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   src/main.rs

no changes added to commit (use "git add" and/or "git commit -a")
```diff
- fn main() {
-     println!("Hello, world!");
+     println!("What's up, world?");
}
```
master

[85762b0] Initial commit
Replace greeting with question
Initial commit

Replace greeting with question

Initial commit
• Use “git help” to learn about commands and get help on those specific commands.
• Now we need to get your local changes to the central remote host for the team
$ git push origin master
Counting objects: 11, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (11/11), 892 bytes | 0 bytes/s, done.
Total 11 (delta 1), reused 0 (delta 0)
To git.ece.iastate.edu:bbartels/example-hello-world.git
  * [new branch] master -> master
Let’s say someone else added a commit to the remote host, such that your local copy is obsolete.
hello_world (master)$ git pull origin master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From git.ece.iastate.edu:bbartels/example-hello-world
  * branch         master  -> FETCH_HEAD
  d0f2f54..bbf92ad  master  -> origin/master
Updating d0f2f54..bbf92ad
Fast-forward
README.md | 3 +++
1 file changed, 3 insertions(+)
create mode 100644 README.md
Intro to Git
FAQ

• What should I put in my commit message?

  • A commit message is typically composed of a header and (sometimes) a body. The header is typically 50 characters or less, and the body is a couple lines down having variable length.

  • It’s typically advised to use an imperative tone in the header, such as “change this” or “add this” as opposed to “changed this” or “added this.”

• How often should I commit?

  • Commit the smallest stable change

  • Or prefix an unstable commit with “wip: ” (work in progress)
Managing Files

• Suggestion

  • Use Git

  • Use a Branch-Review-Merge workflow
Managing Files

• Suggestion

  • Use Git

• Use a Branch-Review-Merge workflow
Branch-Review-Merge Workflow

• Motivation

• Traditional workflows involve everyone making changes to a central branch/trunk

• Not Great

  • Changes are delicate because everyone is working on the stable branch

  • Changes are often made without notice, which can lead to team member confusion
Branch-Review-Merge Workflow

• Big Idea:

1. Create your own personal **branch** (copy of master)
   • This gives you a safe environment to make your changes

2. When finished, create a merge request for your branch
   • This gives your teammates a chance to **review** your code and offer feedback before changes become final

3. After approval, **merge** your changes into master
   • And then delete your personal branch
• Create a new branch that is a copy of master
Local Computer

```
hello_world (improve-comments)$ emacs src/main.rs
hello_world (improve-comments)$ git add src/main.rs
hello_world (improve-comments)$ git commit
[improve-comments 8e656ec] Add doc comment to main function
  1 file changed, 1 insertion(+)
```
• Recall the state of our remote host
hello_world (improve-comments)$ git push origin improve-comments
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 378 bytes | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote:
remote: To create a merge request for improve-comments, visit:
remote: https://git.ece.iastate.edu/bbartels/example-hello-world/merge_requests/new?merge_request%5Bsource_branch%5D=improve-comments
remote:
To git.ece.iastate.edu:bbartels/example-hello-world.git
* [new branch] improve-comments -> improve-comments

Local Computer

Remote Host

master

improve-comments

master

improve-comments

[85762b0]

[d0f2f54]

[bbf92ad]

[8e656ec]
Branch-Review-Merge Workflow

• Now that we have your branch on the remote host, open a Merge Request for your branch to be merged into master
“mention” team members that you want to review your changes in the description of your merge request, so that they get an email.
Add doc comment to main function

Problem
We have no comments on our functions

Solution
Add some comments.

Request to merge improve-comments into master

You can merge this merge request manually using the command line.

Discussion 0 Commits 1 Changes 1

Write a comment or drag your files here...

Markdown and quick actions are supported
Branch-Review-Merge Workflow

- Use the inline code comments to give specific feedback on changes
Branch-Review-Merge Workflow

- When your teammates have reviewed and approved your code, merge it!

Local Computer

- master: [85762b0]
- improve-comments: [85762b0]
- [d0f2f54]
- [bbf92ad]
- [8e656ec]

Remote Host

- master: [85762b0]
- improve-comments: [85762b0]
- [d0f2f54]
- [bbf92ad]
- [8e656ec]
Branch-Review-Merge Workflow

- When your teammates have reviewed and approved your code, merge it!
Branch-Review-Merge Workflow

• When your teammates have reviewed and approved your code, merge it!

Local Computer

```
master
[85762b0]
[d0f2f54]
[bbf92ad]
```

```
improve-comments
[85762b0]
[d0f2f54]
[bbf92ad]
[8e656ec]
```

Remote Host

```
master
[85762b0]
[d0f2f54]
[bbf92ad]
```

```
[8e656ec]
[f17ce0c]
```
hello_world (improve-comments)$ git checkout master
Switched to branch 'master'
hello_world (master)$ git pull origin master
From git.ece.iastate.edu:bbartels/example-hello-world
  * branch            master       -> FETCH_HEAD
Updating bbf92ad..f17ce0c
Fast-forward
  src/main.rs | 1 +
  1 file changed, 1 insertion(+)

**Diagram:**

**Local Computer**
- master
  - [85762b0]
  - [d0f2f54]
  - [bbf92ad]
  - [8e656ec]
  - [f17ce0c]

**Remote Host**
- master
  - [85762b0]
  - [d0f2f54]
  - [bbf92ad]
  - [8e656ec]
  - [f17ce0c]
hello_world (master)$ git branch --delete improve-comments
Deleted branch improve-comments (was f17ce0c).
`git log --pretty=oneline`

```
85762b0 Merge branch 'improve-comments' into 'master'
8e656ec Add doc comment to main function
bbf92ad Add README.md
b0f2f54 Replace greeting with question
85762b0 Initial commit
```
Branch-Review-Merge Workflow

• Other Examples:

  • https://git.ece.iastate.edu/danc/MicroCART/merge_requests/11
What about merge conflicts?
hello_world (hello-earth)$ git pull origin master
From git.ece.iastate.edu:bbartels/example-hello-world
 * branch            master    -> FETCH_HEAD
Auto-merging src/main.rs
CONFLICT (content): Merge conflict in src/main.rs
Automatic merge failed; fix conflicts and then commit the result.
hello_world (hello-earth|MERGING)$

Local Computer

```
master  hello-earth
[f17ce0c]  [f17ce0c]  [e98ad66]
```

Remote Host

```
master  hello-earth
[f17ce0c]  [f17ce0c]  [e98ad66]
[0d46dbf]  [0d46dbf]
```
hello_world (hello-earth|MERGING)$ git status
On branch hello-earth
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge —abrupt" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

  both modified:  src/main.rs

no changes added to commit (use "git add" and/or "git commit -a")

Local Computer

master

  [f17ce0c]
  ├── hello-earth
  │    └── [f17ce0c]
  │        └── [e98ad66]
  └── [0d46dbf]

Remote Host

master

  [f17ce0c]
  ├── hello-earth
  │    └── [f17ce0c]
  │        └── [e98ad66]
  └── [0d46dbf]
```java
// The main function
fn main() {
    println!("What's up, earth?");
    println!("What's up, world?");
    println!("Hello, world!");
}
```

Local Computer

- master: [f17ce0c]
- hello-earth: [f17ce0c] [e98ad66] [0d46dbf]

Remote Host

- master: [f17ce0c] [0d46dbf] [e98ad66]
- hello-earth: [f17ce0c] [e98ad66]
```rust
// The main function
fn main() {
    println!("Hello, earth!");
}
```
```bash
hello_world (hello-earth|MERGING)$ git status
On branch hello-earth
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

    modified:  src/main.rs
```
• Git will likely open a text-editor (vim) for the commit
  • Use “:q” to just quit and use the commit message as is
hello_world (hello-earth)$ git push origin hello-earth
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 436 bytes | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote:
remote: View merge request for hello-earth:
remote: https://git.ece.iastate.edu/bbartels/example-hello-world/merge_requests/2
remote:
To git.ece.iastate.edu:bbartels/example-hello-world.git
e98ad66..1e52485  hello-earth -> hello-earth
Request to merge hello-earth into master

Merge  □ Remove source branch  □ Squash commits  Modify commit message

You can merge this merge request manually using the command line.

Local Computer

<table>
<thead>
<tr>
<th>master</th>
<th>hello-earth</th>
</tr>
</thead>
<tbody>
<tr>
<td>[f17ce0c]</td>
<td>[f17ce0c]</td>
</tr>
<tr>
<td>[0d46dbf]</td>
<td>[0d46dbf]</td>
</tr>
<tr>
<td>[e98ad66]</td>
<td>[e98ad66]</td>
</tr>
<tr>
<td>[1e52485]</td>
<td>[1e52485]</td>
</tr>
</tbody>
</table>

Remote Host

<table>
<thead>
<tr>
<th>master</th>
<th>hello-earth</th>
</tr>
</thead>
<tbody>
<tr>
<td>[f17ce0c]</td>
<td>[f17ce0c]</td>
</tr>
<tr>
<td>[0d46dbf]</td>
<td>[0d46dbf]</td>
</tr>
<tr>
<td>[e98ad66]</td>
<td>[e98ad66]</td>
</tr>
<tr>
<td>[1e52485]</td>
<td>[1e52485]</td>
</tr>
</tbody>
</table>
Request to merge **hello-earth** into **master**

Merged by **bbartels** less than a minute ago

The changes were merged into **master**

You can remove source branch now. Remove Source Branch
Request to merge hello-earth into master

Merged by bbartels about a minute ago
The changes were merged into master
The source branch has been removed.
hello_world (hello-earth)$ git checkout master
Switched to branch 'master'
hello_world (master)$ git pull origin master
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 1 (delta 0)
Unpacking objects: 100% (1/1), done.
From git.ece.iastate.edu:bbartels/example-hello-world
 * branch master -> FETCH_HEAD
   0d46dbf..3a24661 master -> origin/master
Updating 0d46dbf..3a24661
Fast-forward
src/main.rs | 2 +
1 file changed, 1 insertion(+), 1 deletion(-)

Local Computer

<table>
<thead>
<tr>
<th>master</th>
<th>hello-earth</th>
</tr>
</thead>
<tbody>
<tr>
<td>[f17ce0c]</td>
<td>[f17ce0c]</td>
</tr>
<tr>
<td>[0d46dbf]</td>
<td>[0d46dbf]</td>
</tr>
<tr>
<td>[e98ad66]</td>
<td>[e98ad66]</td>
</tr>
<tr>
<td>[1e52485]</td>
<td>[1e52485]</td>
</tr>
<tr>
<td>[3a24661]</td>
<td></td>
</tr>
</tbody>
</table>

Remote Host

<table>
<thead>
<tr>
<th>master</th>
</tr>
</thead>
<tbody>
<tr>
<td>[f17ce0c]</td>
</tr>
<tr>
<td>[0d46dbf]</td>
</tr>
<tr>
<td>[e98ad66]</td>
</tr>
<tr>
<td>[1e52485]</td>
</tr>
<tr>
<td>[3a24661]</td>
</tr>
</tbody>
</table>
hello_world (master)$ git branch -d hello-earth
Deleted branch hello-earth (was 1e52485).
hello_world (master)$ git log --pretty=oneline
3a246611411a9ca9de8ad6aa9571586501046b0  Merge branch 'hello-earth' into 'master'
1e52485a53cc699479766e5ed42caab7272c68b7  Merge branch 'master' of git.ece.iastate.edu:bbart
20e98ad66e5468ee921fcbfb14b7e688ded9fb92f2  Change subject from world to earth
0d46dbf7b7878222e8404354e48dd26a1950270  Revert "Replace greeting with question"
8f17ce0cd6b3b6346fbadb01e0041309328bc18f29  Merge branch 'improve-comments' into 'master'
e8656ec5e368126486bbdb5f4af2f581bac7e391  Add doc comment to main function
bbf92ad2b0f4b859b6d76466358dfcfbfa475e3  Add README.md
d0f2f54b0979afe04f0654a69162927c9395217e  Replace greeting with question
85762b00d392a49670cf660749955f11c7dad9a  Initial commit

Local Computer

- master
  - [f17ce0c]
  - [0d46dbf]
  - [e98ad66]
  - [1e52485]
  - [3a24661]

Remote Host

- master
  - [f17ce0c]
  - [0d46dbf]
  - [e98ad66]
  - [1e52485]
  - [3a24661]
Branch-Review-Merge Workflow FAQ

• What about files generated by software? (not directly written by you)
  
  • Still use Git, but keep in mind the diffs will be unmanageable, don’t bother examining them

  • Resolve merge conflicts with “git checkout —ours” or “git checkout —theirs”
    • (This essentially ignores all of one team member’s changes, so it is best to avoid this scenario. Coordinate with your teammates so that only one person is ever working on binary files at a time.)
Managing Files

• BONUS

• For tutorials, how-to’s, or orientational documents, write these documents as Markdown files, and then gitlab will render these as web pages for you

• Markdown is a very simple and natural syntax

• It’s quite easy to add links, pictures, code snippets

• Gitlab’s Markdown Reference:
  • https://docs.gitlab.com/ee/user/markdown.html
# MicroCART

_Microprocessor Controlled Aerial Robotics Team_

**Project Statement**: “To create a modular platform for research in controls and embedded systems.”

Since 1998, MicroCART has been building aerial robots. Currently, we are building a quadcopter that can fly autonomously.

MicroCART has 3 areas of development:
- The **quadcopter** itself
  - The quadcopter has been built from the ground up, incorporating a Zyro board that provides the processor and a FPGA, an IMU, props, a LiPo battery, a receiver for manual remote control, and a frame that holds it all together.
- The **ground station**
  - The ground station is responsible for issuing important data to the quad including position data from the camera system and issuing commands to the quad for things like configuration and path following directives.
- The **controls model**
  - The quadcopter processor is programmed to implement a PID controller. We use a Simulink model to streamline the PID tuning process and to facilitate effective characterization of the quad.

## Sections

- [Quadcopter](quad/README.md)
- [Ground Station](groundStation/README.md)
- [Controls](controls/README.md)

## Documentation

- How to demo the quadcopter ([documentation/how_to_demo.md](documentation/how_to_demo.md))
- How to charge the LiPo batteries ([documentation/how_to_charge_lipo.md](documentation/how_to_charge_lipo.md))
- Continuous Integration FAQ ([documentation/ci_faq.md](documentation/ci_faq.md))
- How to document things on GitLab ([documentation/how_to_document.md](documentation/how_to_document.md))
- How to update the website ([website/README.md](website/README.md))
- How to use Git ([documentation/how_to_use_git.md](documentation/how_to_use_git.md))

## Stable Releases

To browse stable releases from previous teams, view the [Tags].
Demo
Review How to Manage

• Tasks
  • Define and track tasks with Gitlab Issues

• Communication Channels
  • Use Gitlab issue threads whenever you are talking about tasks

• Files
  • Use Git and Gitlab Merge Requests
  • Use Markdown to turn files into web pages on Gitlab
Questions?