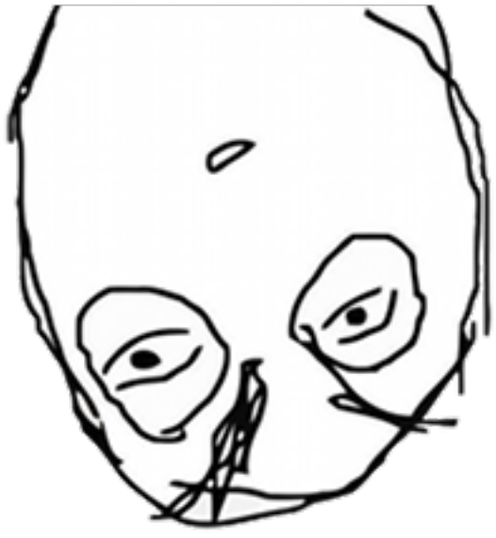
Modern Project Management

Brendan Bartels
B.S. Electrical Engineering
B.S. Biochemistry

Accessibility into the past

The past is the key to your success as an engineer



You will become an expert

“Tell me about a time when you enabled yourself to remember something *after you had forgotten it.*”

“Tell me about a time when you enabled your *team* to learn something that you know *in your absence*.”

¯_(ツ)_/¯

Archaic Project Management

Brendan Bartels
B.S. Electrical Engineering
B.S. Biochemistry

Modern ~~Archaic~~ Project Management

Brendan Bartels
B.S. Electrical Engineering
B.S. Biochemistry

You Need To Manage

- Tasks
- Communication Channels
- Files

You Need To Manage

- Tasks
- Communication Channels
- Files

You Need To Manage

- Tasks
- Communication Channels
- Files

GitLab

What is Git?

- Tool that allows you to **track file changes** over time
- https://git-scm.com/

What is Gitlab?

- A web application that allows you to
  - organize tasks
  - track work
  - document changes
  - integrate with Git

- This is the primary tool that will give you accessibility into your project

- https://about.gitlab.com/

How To Manage

- Tasks
- Communication Channels
- Files

How To Manage

- Tasks
- Communication Channels
- Files

Managing Tasks

- Why do we care?
    - We have to complete the tasks of the project to succeed… obviously…
    - The entire team will not equally understand the tasks that need to be done.
        - Can lead to sudden disagreements
        - Can lead to people being left on the fringes

Managing Tasks

- How do we manage tasks?
    - Determine the tasks
        - This is typically the responsibility of leadership
        - Rely on the expertise throughout your team
    - Write the tasks down *in detail*
        - Make the tasks **accessible** to everyone on the team
        - This is also the responsibility of leadership

Managing Tasks

- What tools should we use?
  - Trello is good, but…
  - Use **Gitlab Issues**
    - Gitlab Issue = Task
    - Has a Trello-like board UI to help visualize tasks

Open 11 Closed 13 All 24

Edit Issues New issue

Search or filter results... Oldest updated

MicroCART: Documentation (Dr. Jones Feedback) 4
#24 · opened 3 months ago by phjones Documentation updated a week ago

Rotate filtered flow velocity 0
#23 · opened 3 months ago by dawehr Quad updated a week ago

Implement angle unwrapping for yaw 0
#22 · opened 3 months ago by dawehr Quad updated a week ago

Remove read_kill and associated functions in favor of ROC functions 0
#21 · opened 3 months ago by bbartels Quad updated a week ago

Upgrade our unit testing framework 1
#20 · opened 3 months ago by bbartels Improvement updated a week ago

Divide quad_main of quad_app into smaller functions (setup, loop, teardown) 0
#19 · opened 4 months ago by bbartels Quad updated a week ago

Blink LED when sending logs 0
#18 · opened 4 months ago by bbartels Improvement Quad updated a week ago

Stop setting the hardware interfaces as global variables 1
#17 · opened 4 months ago by bbartels Quad Tech Debt updated a week ago

Closed Issue #15 opened 4 months ago by dawehr Edit Reopen issue New issue

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.

Related issues 0 +

1 Related Merge Request

!13 quad: add valgrind to CI Merged

0 0 New branch unavailable

dawehr @dawehr commented 4 months ago Developer

Valgrind can be launched by simply calling:

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

bbartels @bbartels mentioned in issue #18 4 months ago

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

bbartels @bbartels commented 4 months ago Master

Resolved in !13 (merged).

Edited about an hour ago by bbartels

Todo Add todo »

Assignee Edit
bbartels
@bbartels

Milestone Edit
None

Time tracking
No estimate or time spent

Due date Edit
No due date

Labels Edit
None

Weight Edit
None

2 participants

Notifications Unsubscribe

Reference: danc/MicroCART#15

Project Repository **Issues** 11 Merge Requests 0 Pipelines Snippets Members Settings

List **Boards** Labels Milestones

Development | Search or filter results... | Add list | Add issues

Backlog 10 +

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

In Progress 1 +

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

Closed 13

Delete Unused Files in quad_app #16
Cleanup 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

Managing Tasks

- Examples:
  - https://git.ece.iastate.edu/danc/MicroCART/boards?=
  - https://git.ece.iastate.edu/danc/MicroCART/issues/1

Managing Tasks
FAQ

- How do I add pictures/code/links to my Gitlab Issues?
  - All Gitlab posts **support Markdown**
    - (https://docs.gitlab.com/ee/user/markdown.html)

bold	**Bold**
italic	*Italic*
# Heading 1	Heading 1
## Heading 2	Heading 2
[link](http://example.com)	Link
(For pictures, just copy and paste into the draft)	
> Blockquote	Blockquote
- List - List	• List • List
1. One 2. Two	1. One 2. Two
`Inline code` with backticks	`Inline code` with backticks

How To Manage

- Tasks
- Communication Channels
- Files

How To Manage

- Tasks
- Communication Channels
- Files

Managing Communication Channels

- Why do we care?
    - Communication between people is **difficult**.
    - People forget things.
    - Not all communication tools are equal.
        - We want the best tools to ensure information is most accessible between team members.

Managing Communication Channels

- What communication channels should we use?
    - When you aren't talking about project tasks, use something that **prioritizes speed**
        - Slack is a good choice
    - When you are talking about project tasks, use something **prioritizes accessibility**
        - Slack is **not a good** choice

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
    - You can notify any team members with their @netid to get them involved in a conversation
        - Sends them an email

Closed Issue #16 opened 4 months ago by dawehr Edit Reopen issue New issue

Delete Unused Files in quad_app

There are several legacy files in the quad_app which are unused. Removing these would help next year's team get up to speed quicker.

Related issues 0 +

0 0 New branch unavailable

bbartels @bbartels added Tech Debt label 4 months ago

bbartels @bbartels commented 4 months ago Master

It also might be worth deleting unused code and unused fields on structs. Everything is recoverable from the git history, and this will also help things stay concise.

bbartels @bbartels commented 4 months ago Master

After an initial audit, I've come up with a list of files I believe are either not used or obsolete, and eligible for deletion. Can you guys confirm? @dawehr @erlom @jpbush

- PID.c
- PID.h
- README.txt
- controllers.c
- gam.h
- new_PID.h
- new_log_data.h
- new_log_data.c
- old_log_data.h
- packet_processing.c
- packet_processing.h
- quadposition.h
- update_gui.c
  - This will require us to remove the update gui function from the main loop
- update_gui.h
  - This will require us to remove the update gui function from the main loop

FYI: @phjones

Edited 4 months ago by bbartels

dawehr @dawehr commented 4 months ago Developer

`PID.h` needs to be kept. It holds all the default PID values.

`gam.h` has the definition for `gam_t`, which is used. That typedef should probably just be moved into `type_def.h` instead, and `gam.h` deleted.

Todo Add todo »

Assignee Edit
No assignee - assign yourself

Milestone Edit
May17-16 Final Version

Time tracking
No estimate or time spent

Due date Edit
No due date

Labels Edit
Cleanup Tech Debt

Weight Edit
None

5 participants

Notifications Unsubscribe

Reference: dang/MicroCART#16

Use team members' username and they will get an email notification

Managing Communication Channels

- Okay… okay…
    - 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

Closed **Issue #9** opened 4 months ago by **bbartels** | Edit | Reopen issue | New issue

Use a normalized PWM value

In the quad app, we are using a legacy unit of 1e-8 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 450 Hz, so our duty cycle would range from 40%-80% in the hardware layer.

Related issues 0 +

0 0

New branch unavailable

bbartels @bbartels added **Tech Debt** label 4 months ago

bbartels @bbartels commented 4 months ago Master

On this issue:

> brendan [5:06 PM] In the process of fixing a unit in the quad code, a piece of me would like to just fix the unit to something we all like. From the model perspective, would you guys prefer to work with duty cycle, or does it really not matter? FYI @david
> andy [6:36 PM] I don't know what is best honestly. As far as I know (and feel free to correct me if I am wrong), ESCs typically operate between the 1ms to 2ms pulse range. This corresponds to a duty cycle of 40% to 80% given the frequency we are running at. To me it makes more sense to have work with the pulse range, as that doesn't typically change even if the ESCs get upgraded. Working with duty cycles means that if sometime in the future they want to run at a different frequency, some values in the model will have to be changed...
> brendan [7:12 PM] That makes sense to me. The issue is that we'd like to keep the quad application decoupled from a specific hardware implementation. We can keep hardware specific stuff in a relatively thin driver layer. Keeping that level of platform agnosticism for our app keeps it maximally compatible with any system it might find itself running on. So I guess I'm trying to shoot for what the most inter-platform constant variable would be. At first, I thought that would be duty cycle (isn't pulse width frequency dependent?)
> andy [7:25 PM] If you want the quad application decoupled from hardware then duty cycle

Todo Add todo »

Assignee Edit
No assignee - assign yourself

Milestone Edit
None

Time tracking
No estimate or time spent

Due date Edit
No due date

Labels Edit
Tech Debt

Weight Edit
None

5 participants

Notifications Unsubscribe

Reference: dane/MicroCART#9

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 work on the same files, 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 should we manage files?
  - For things that require **real-time collaboration**
    - Use Google Docs or Cybox + Office 365
- Otherwise…
  - Use Git + Branch-Review-Merge workflow

Managing Files

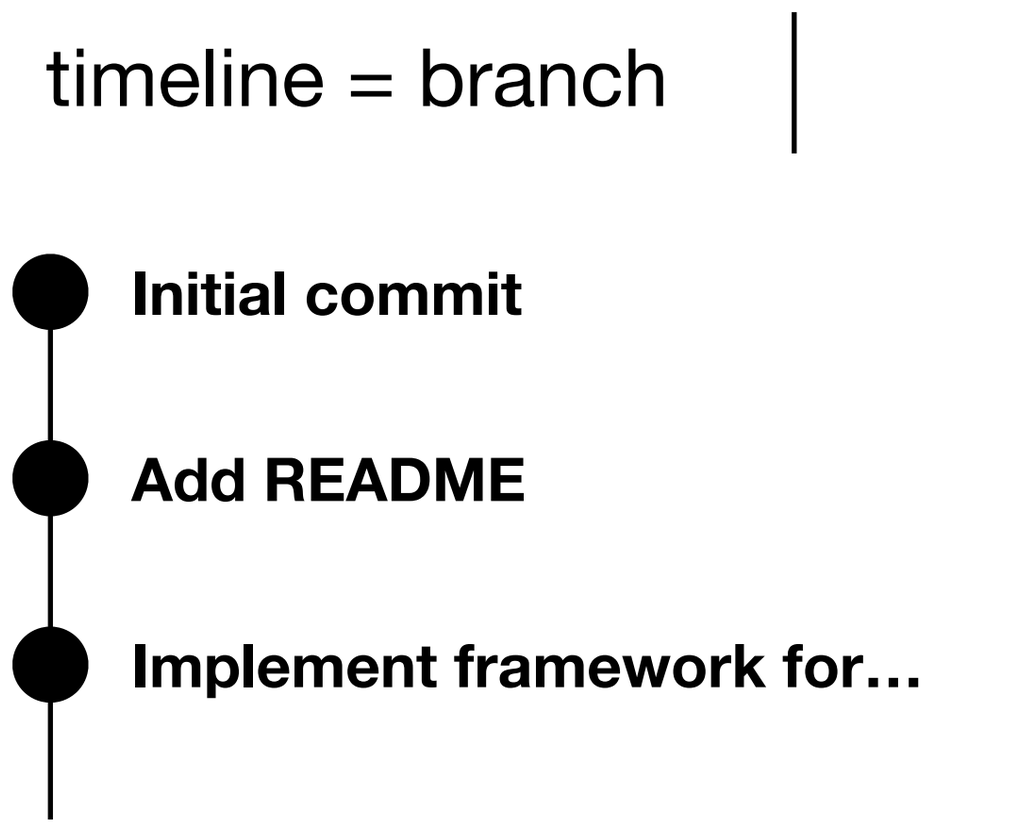
- Using Git
- Using a Branch-Review-Merge workflow

Managing Files

- Using Git
- Using a Branch-Review-Merge workflow

Intro to Git

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



https://git.ece.iastate.edu/danc/MicroCART/blob/master/documentation/how_to_use_git.md#how-do-i-make-changes

```
hello_world $ git init
Initialized empty Git repository in /Users/brendanbartels/workspace/hello_world/.git/
```

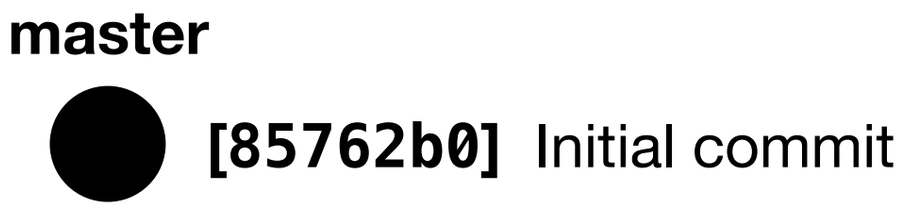
To start tracking a project with git, enter the directory and use the “git init” command

```
hello_world (master)$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        Cargo.lock
        Cargo.toml
        src/

nothing added to commit but untracked files present (use "git add" to track)
```

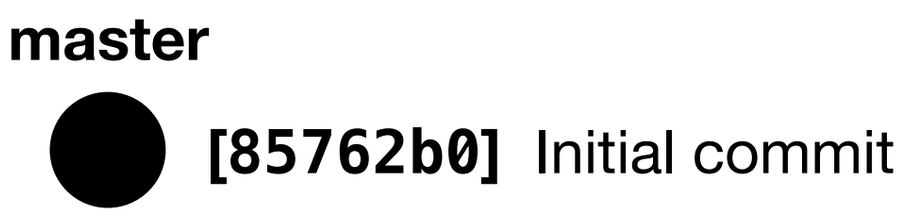
Hint: Use “git status” to get a hint about what you should do

```
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
```

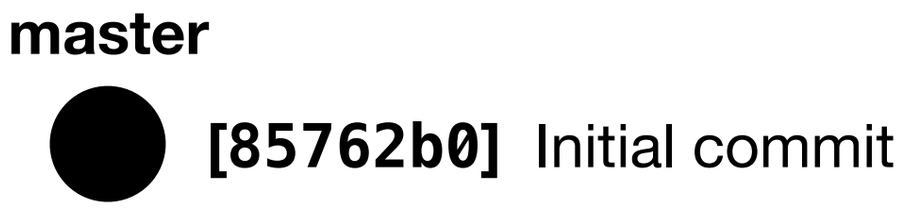
```
hello_world (master)$ git commit -m "Initial commit"
[master (root-commit) 85762b0] Initial commit
 4 files changed, 14 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 Cargo.lock
 create mode 100644 Cargo.toml
 create mode 100644 src/main.rs
```



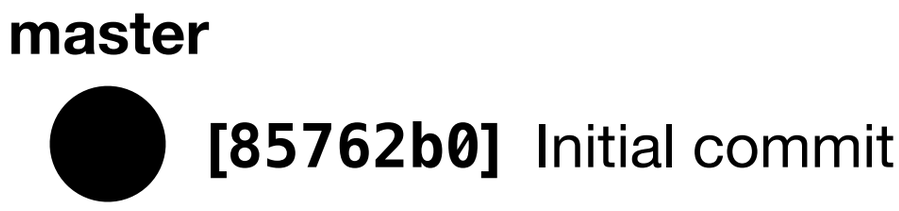
```
hello_world (master)$ git status
On branch master
nothing to commit, working tree clean
```



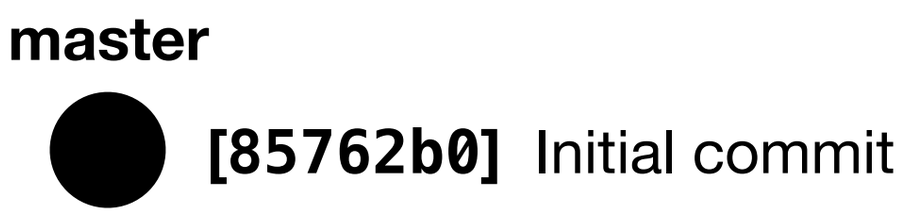
```
hello_world (master)$ emacs src/main.rs
hello_world (master)$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   src/main.rs

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



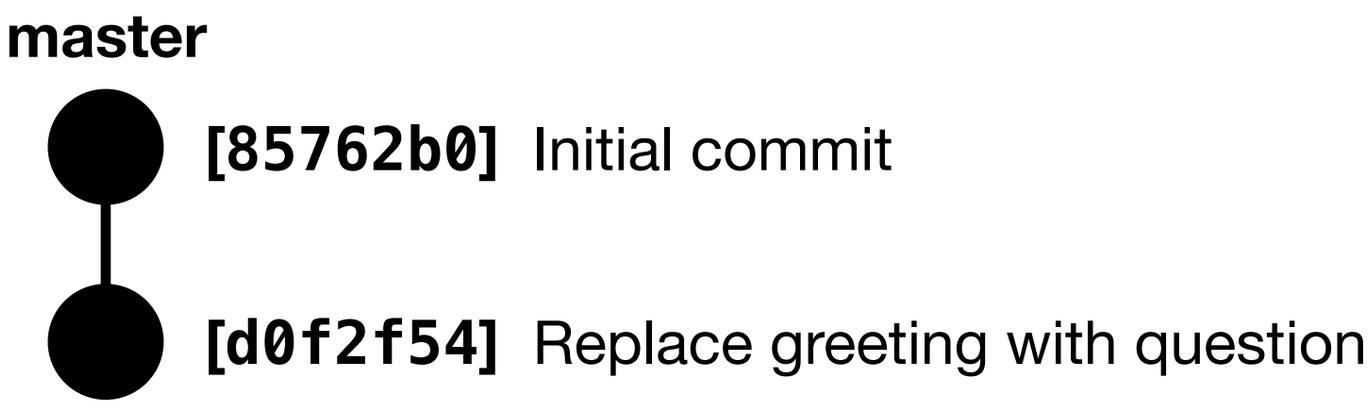
```
hello_world (master)$ git diff
diff --git a/src/main.rs b/src/main.rs
index e7a11a9..394f234 100644
--- a/src/main.rs
+++ b/src/main.rs
@@ -1,3 +1,3 @@
 fn main() {
-    println!("Hello, world!");
+    println!("What's up, world?");
 }
```



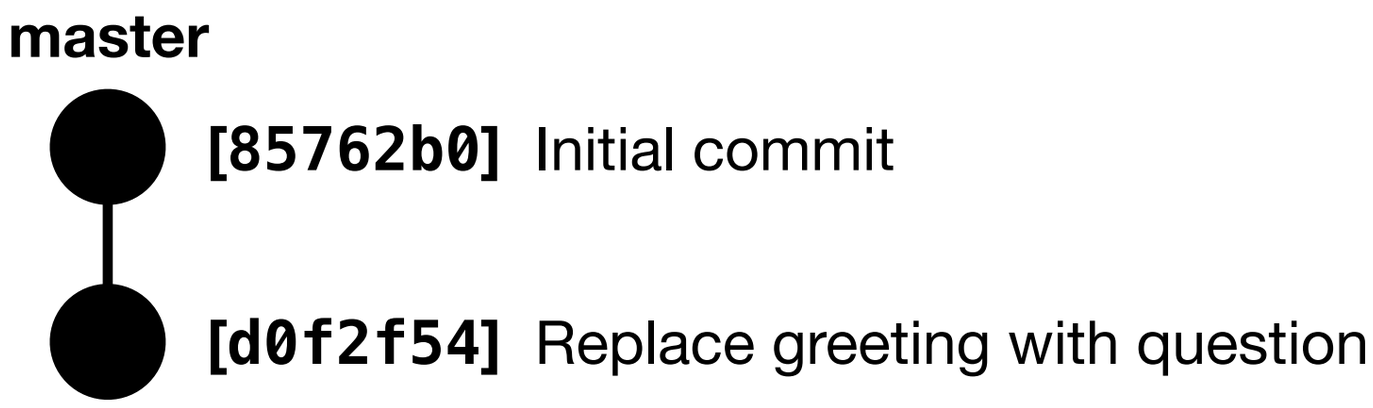
```
hello_world (master)$ git add src/main.rs
hello_world (master)$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   src/main.rs
```



```
hello_world (master)$ git commit -m "Replace greeting with question"
[master d0f2f54] Replace greeting with question
 1 file changed, 1 insertion(+), 1 deletion(-)
```

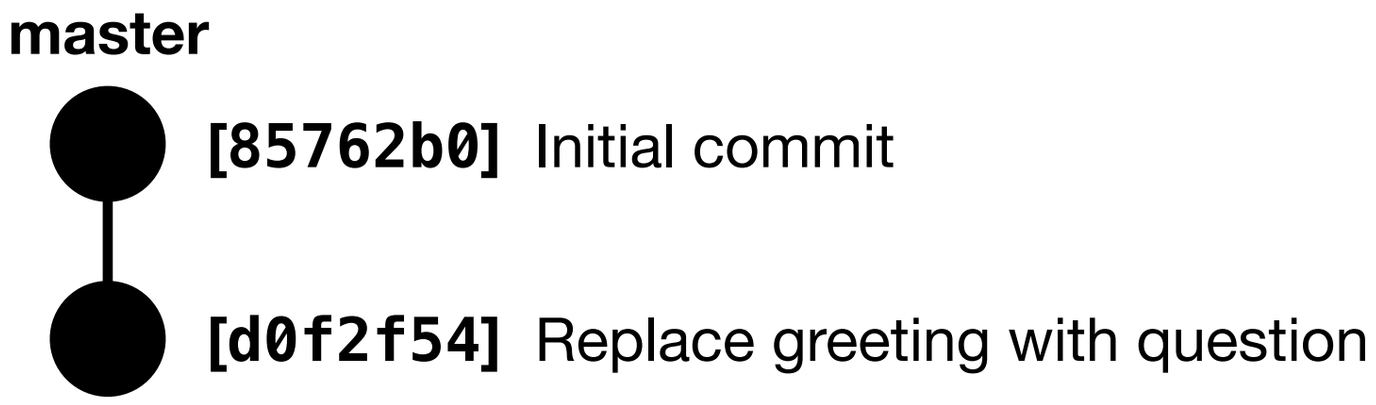


```
hello_world (master)$ git log
commit d0f2f54b0979afe04f0654a69162927c9395217e
Author: Brendan Bartels <bbartels@iastate.edu>
Date:   Mon Aug 28 20:54:09 2017 -0500

    Replace greeting with question

commit 85762b00d392a49670cfd660749955f11c7dad9a
Author: Brendan Bartels <bbartels@iastate.edu>
Date:   Mon Aug 28 20:45:23 2017 -0500

    Initial commit
```



```
hello_world (master)$ git help rm
hello_world (master)$ git help
```

- Use “git help” to learn about commands and get help on those specific commands.



```
hello_world (master)$ git remote add origin git@git.ece.iastate.edu:bbartels/example-hello
-world.git
hello_world (master)$ git remote -v
origin  git@git.ece.iastate.edu:bbartels/example-hello-world.git (fetch)
origin  git@git.ece.iastate.edu:bbartels/example-hello-world.git (push)
```

- Now we need to get your local changes to the central remote host for the team

```
hello_world (master)$ 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

- Using Git
- Using a Branch-Review-Merge workflow

Managing Files

- Using Git
- Using a Branch-Review-Merge workflow

Branch-Review-Merge Workflow

- Why do we need to do this?
    - 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

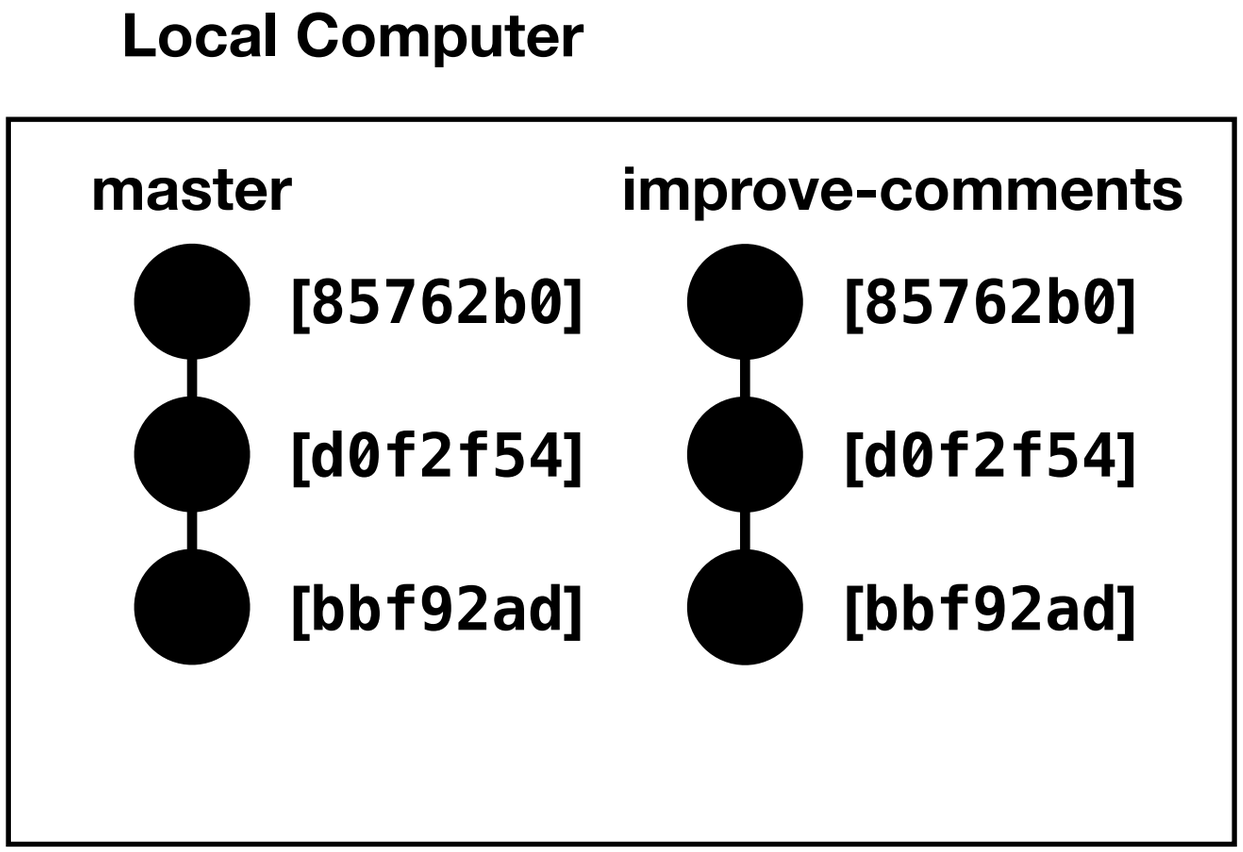
Branch-Review-Merge Workflow

- Big Idea:

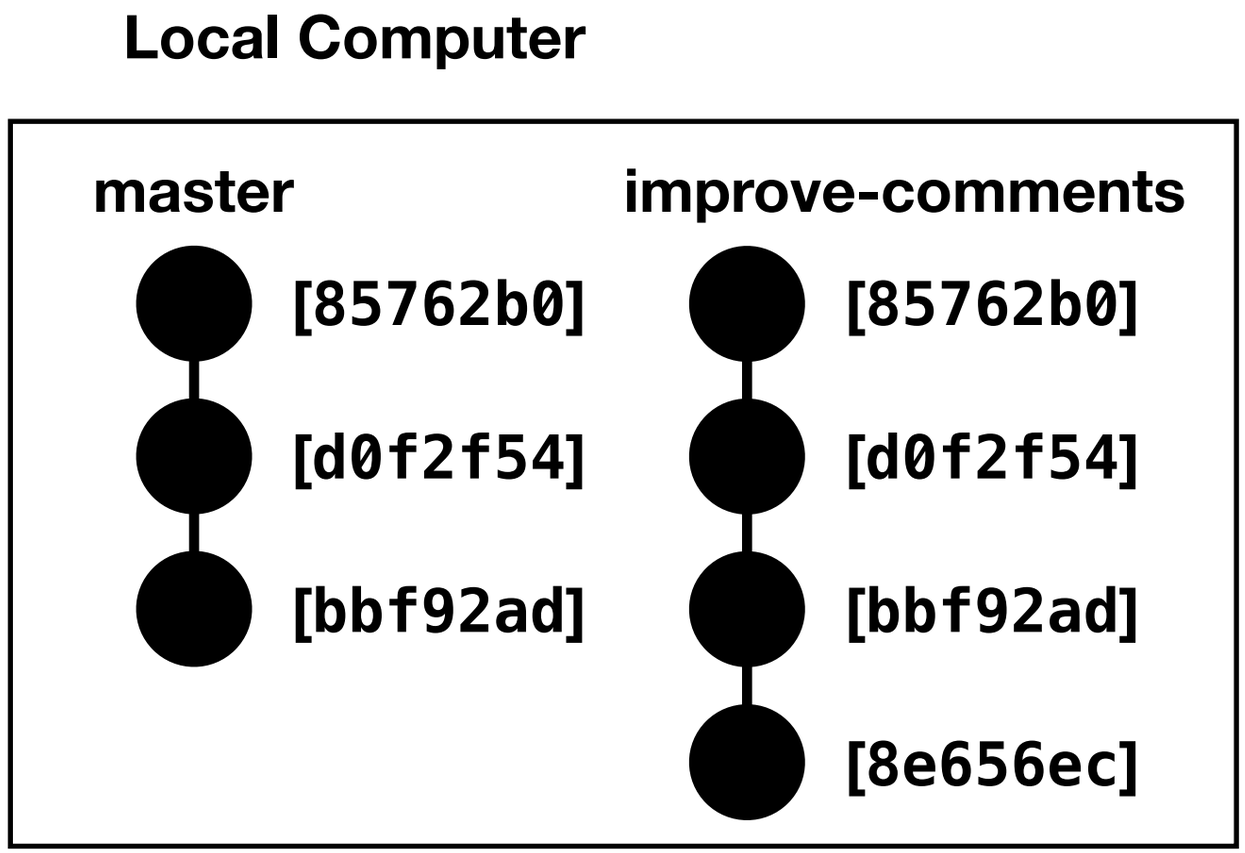
1. Create your own personal **branch** (copy of master)
   - This gives you a safe environment to make your changes
2. When finished, create a merge request for your branch
   - This gives your teammates a chance to **review** your code and offer feedback before changes become final
3. After approval, **merge** your changes into master
   - And then delete your personal branch

```
hello_world (master)$ git checkout -b improve-comments
Switched to a new branch 'improve-comments'
```

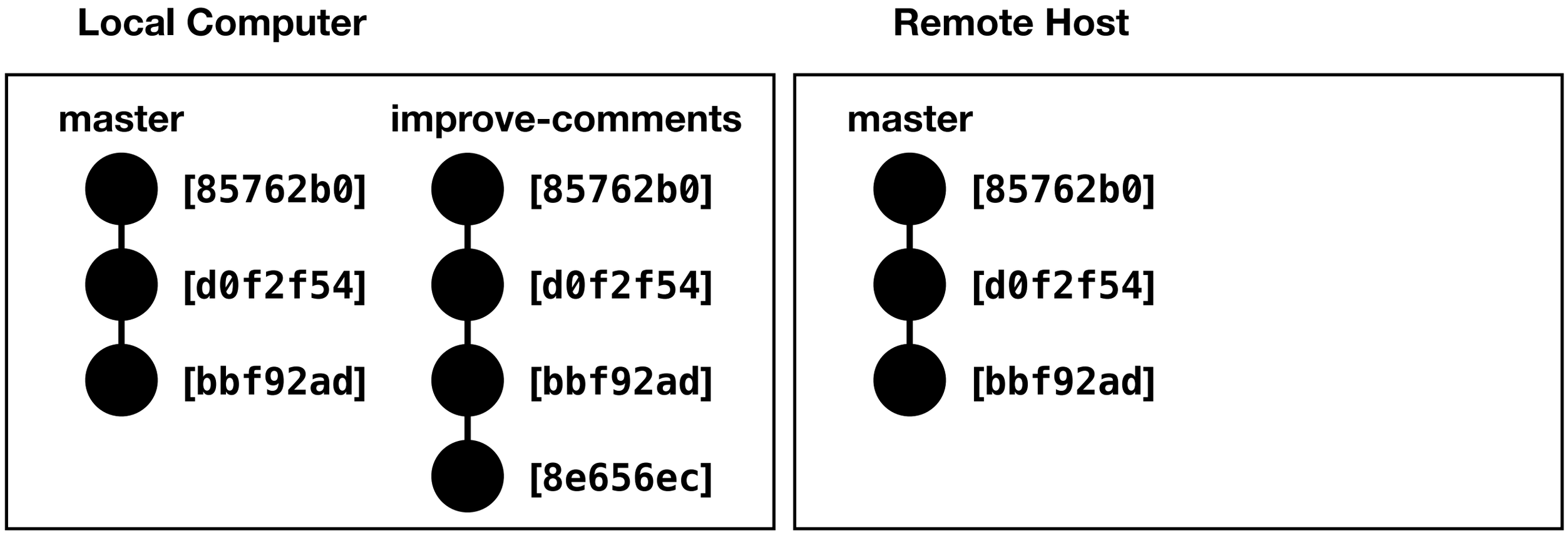
- Create a new branch that is a copy of master



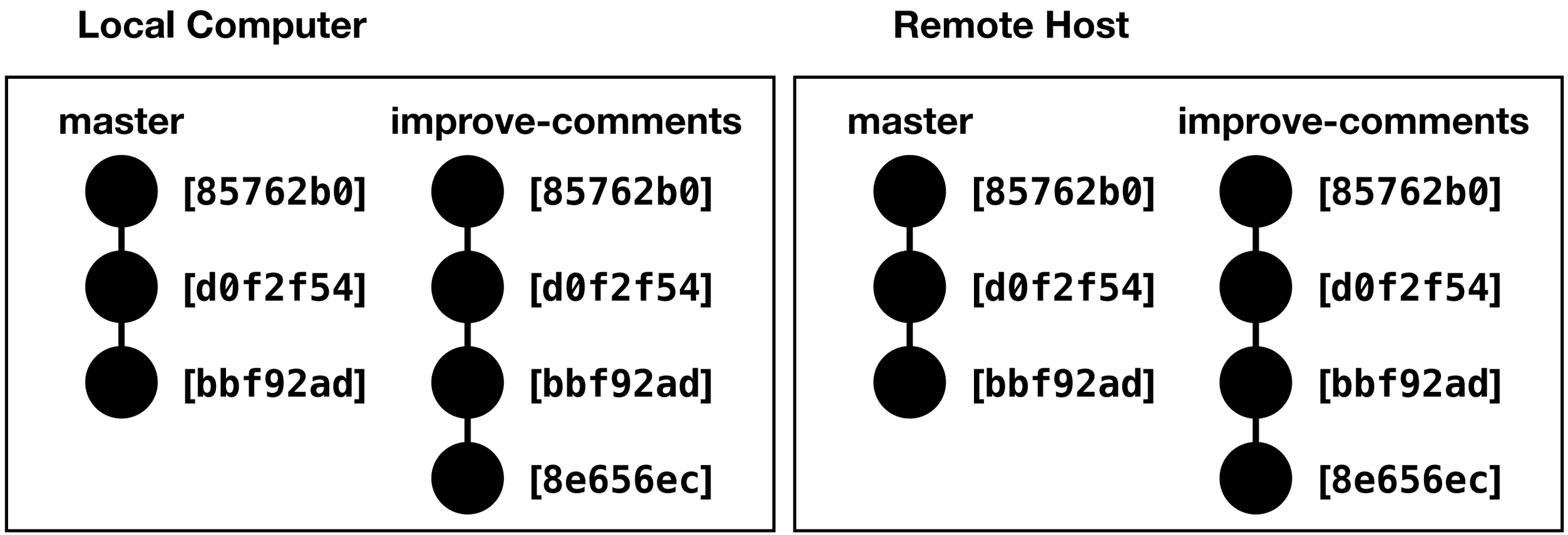
```
(hello_world (improve-comments)$ emacs src/main.rs
(hello_world (improve-comments)$ git add src/main.rs
(hello_world (improve-comments)$ git commit
[improve-comments 8e656ec] Add doc comment to main function
 1 file changed, 1 insertion(+)
```



- 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?merg
e_request%5Bsource_branch%5D=improve-comments
remote:
To git.ece.iastate.edu:bbartels/example-hello-world.git
 * [new branch]      improve-comments -> improve-comments
```



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

New Merge Request

From improve-comments into master

Change branches

Title: Add doc comment to main function

Start the title with WIP: to prevent a **Work In Progress** merge request from being merged before it's ready.

Add description templates to help your contributors communicate effectively!

Description

Write | Preview

```
## Problem
We have no comments on our functions

## Solution
Add some comments.
```

Markdown and quick actions are supported — Attach a file

Assignee: Assignee — Assign to me

Milestone: Milestone

Labels: Labels

Source branch: improve-comments

Target branch: master — Change branches

Remove source branch when merge request is accepted.

Squash commits when merge request is accepted. About this feature

Submit merge request — Cancel

Commits 1 | Changes 1

Showing **1 changed file** with **1 additions** and **0 deletions** — Inline | Side-by-side

src/main.rs — Edit | View file @ 8e656ec5

```
+/// the main function
 fn main() {
     println!("What's up, world?");
 }
```

Notify team members that you want to review your changes in the description of your merge request, so that they get an email

Open **Merge request !1** opened less than a minute ago by **bbartels**

Edit Close merge request

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` Check out branch

Merge Remove source branch Modify commit message

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

0 0

Discussion 0 Commits 1 Changes 1

Write Preview

Write a comment or drag your files here...

Markdown and quick actions are supported Attach a file

Comment Close merge request

Todo Add todo »

Assignee Edit
No assignee - assign yourself

Milestone Edit
None

Time tracking
No estimate or time spent

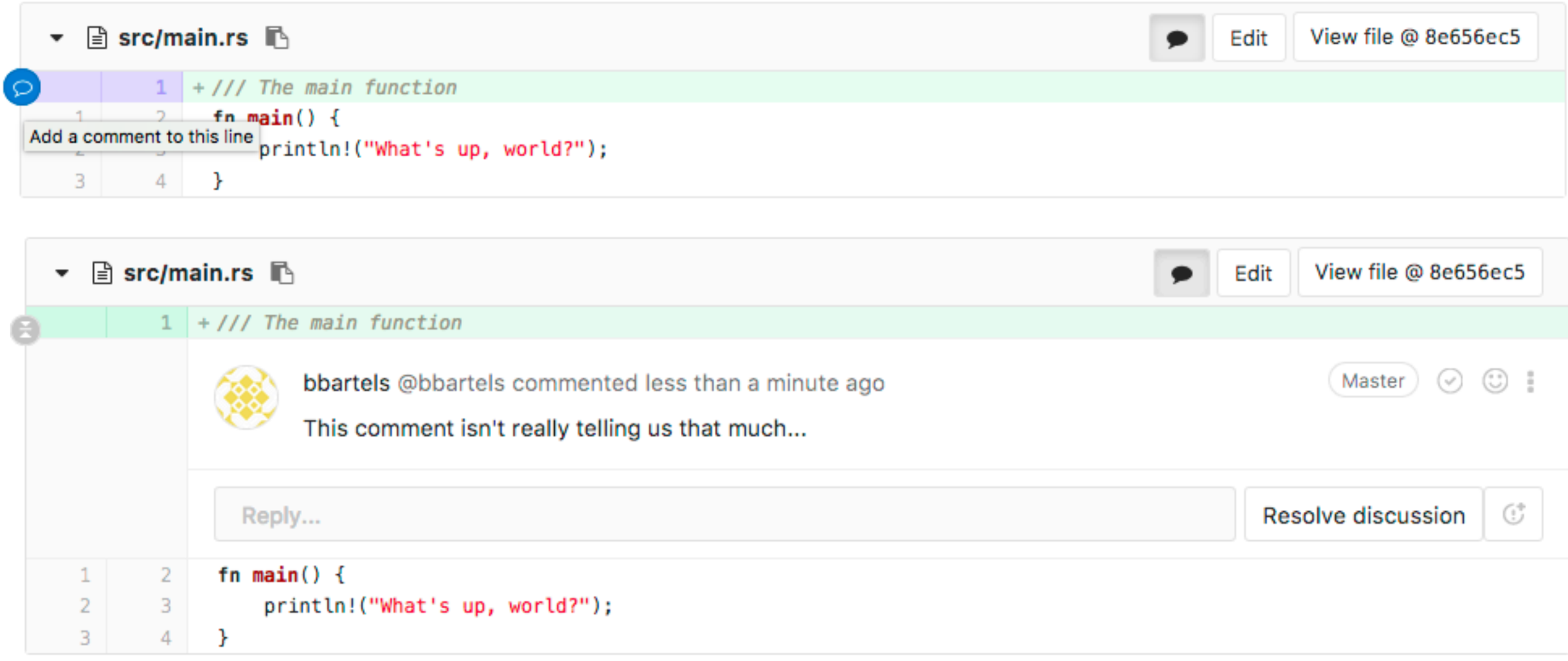
1 participant

Notifications Unsubscribe

Reference: bbartels/example-...

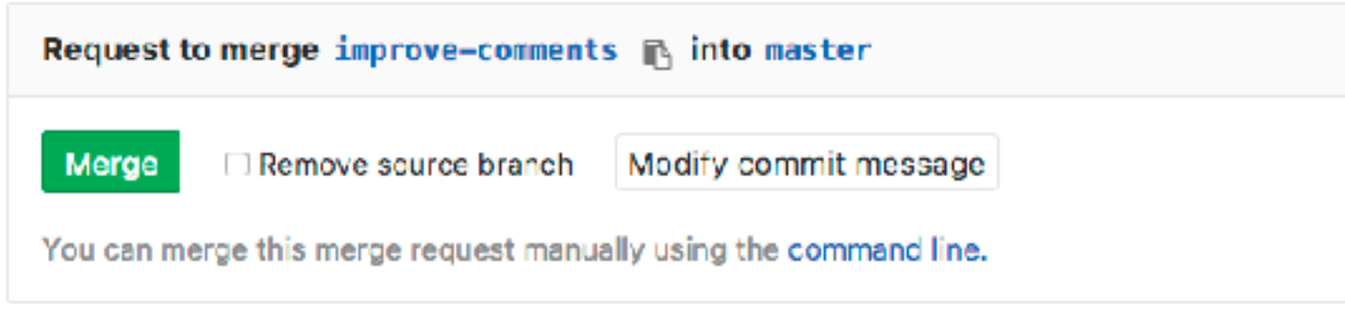
Branch-Review-Merge Workflow

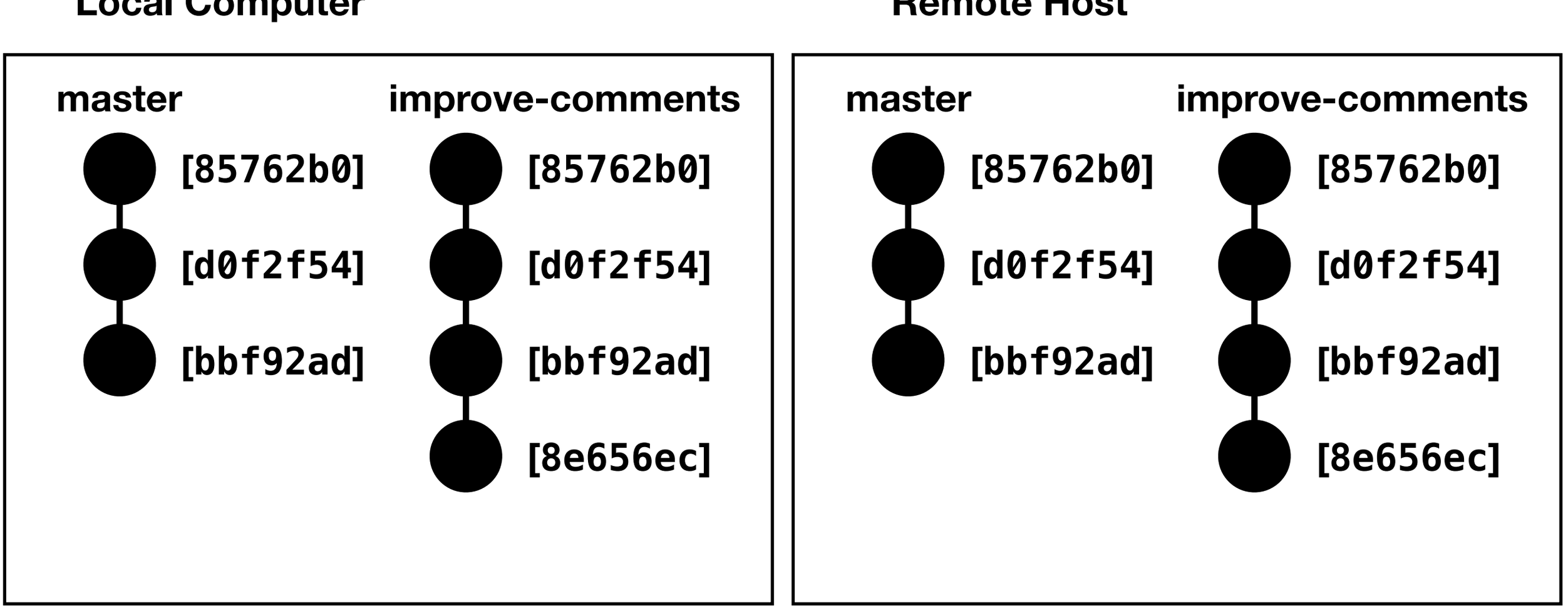
- Use the inline code comments to give specific feedback on changes



Branch-Review-Merge Workflow

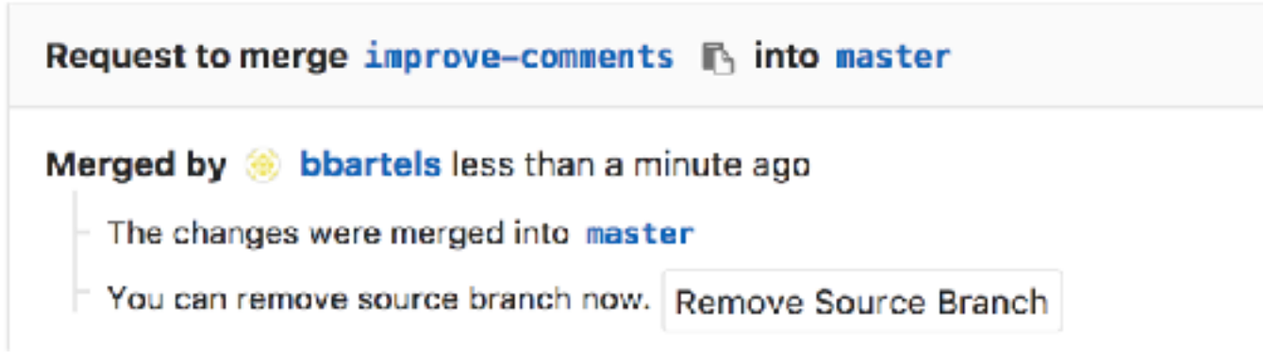
- When your teammates have reviewed and approved your code, merge it!

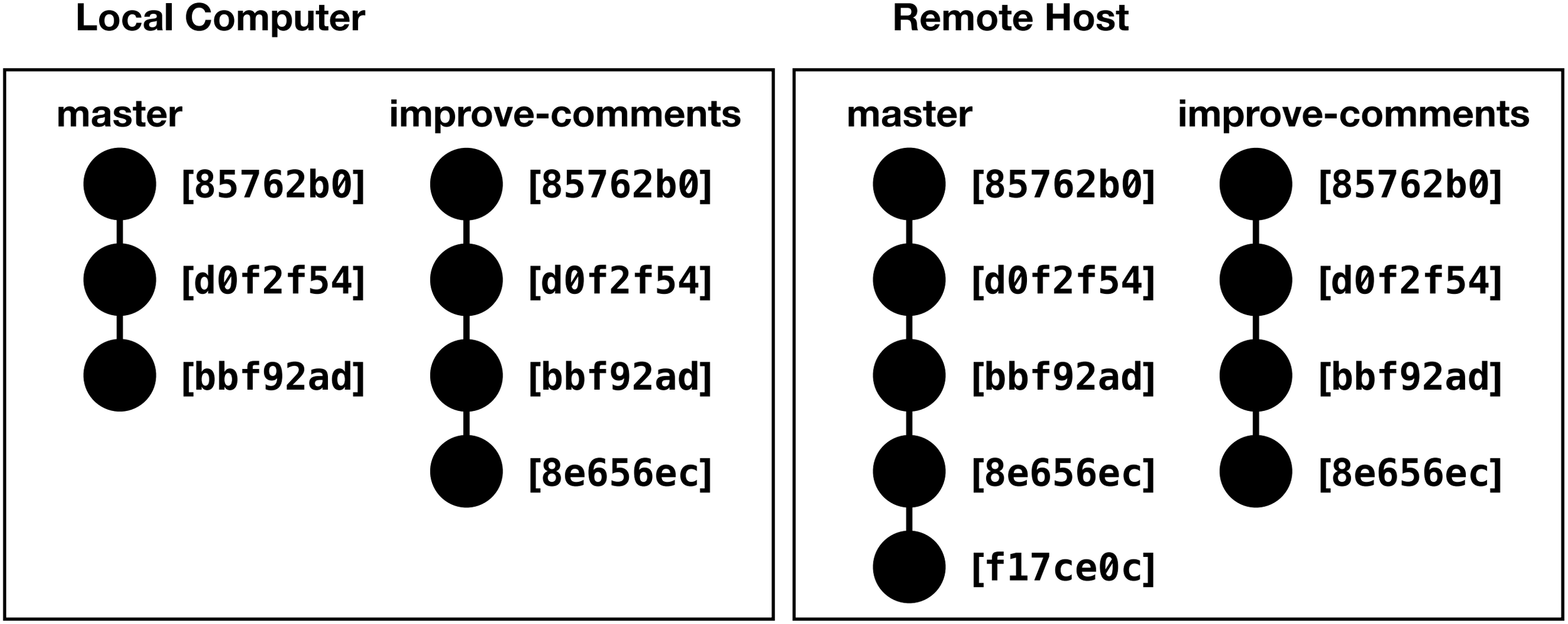




Branch-Review-Merge Workflow

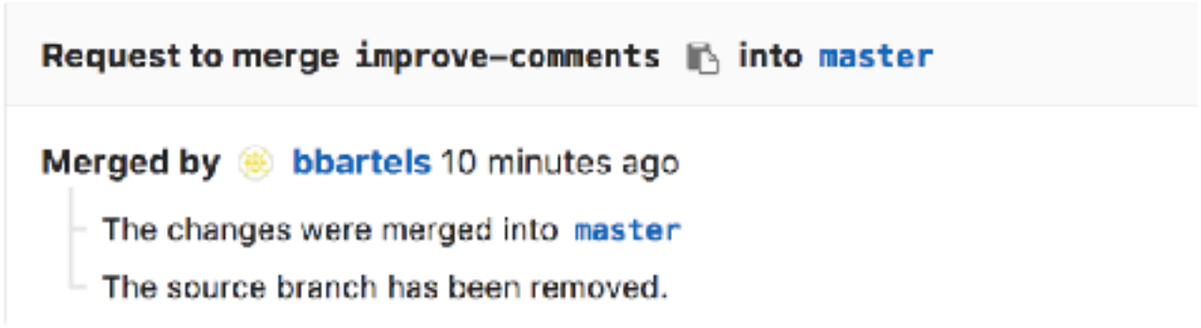
- When your teammates have reviewed and approved your code, merge it!



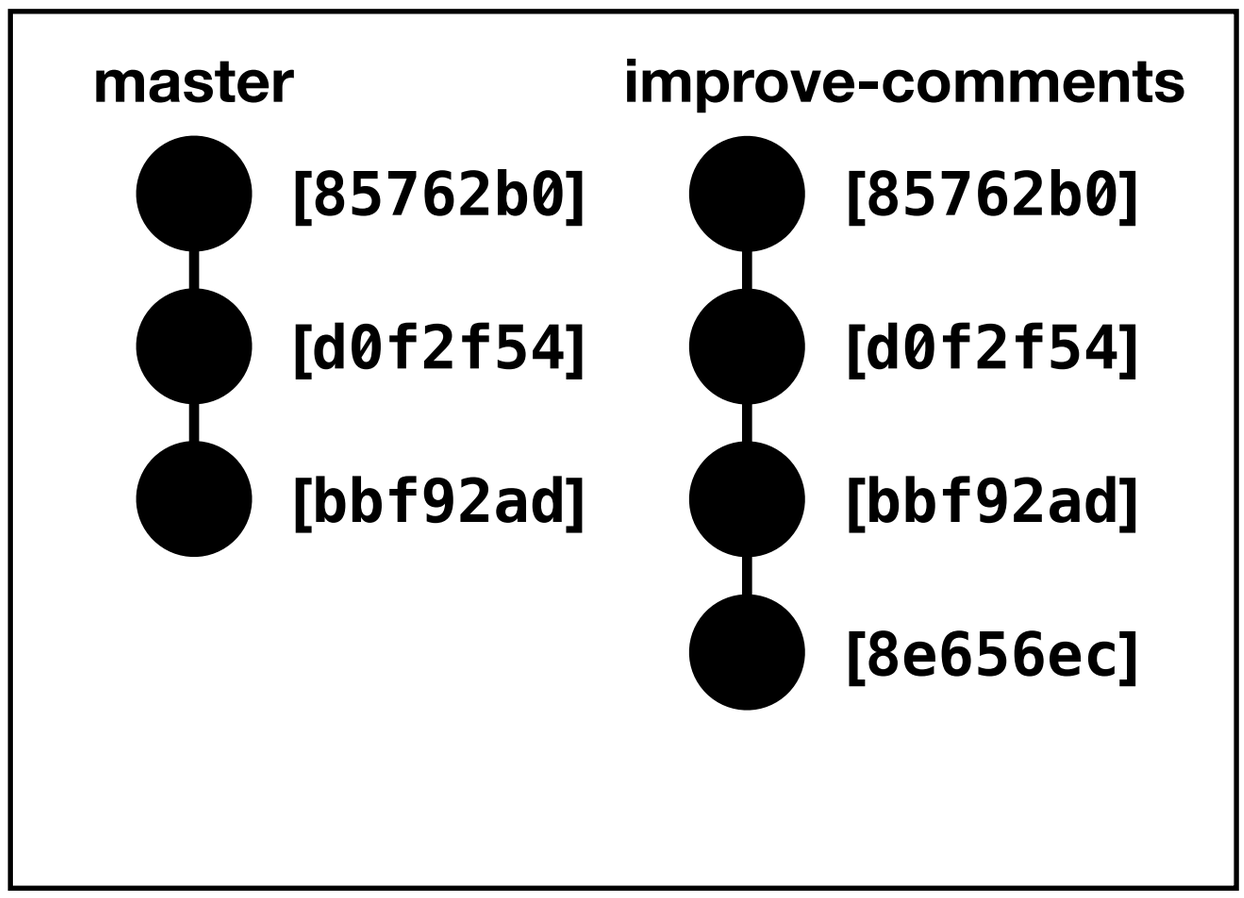


Branch-Review-Merge Workflow

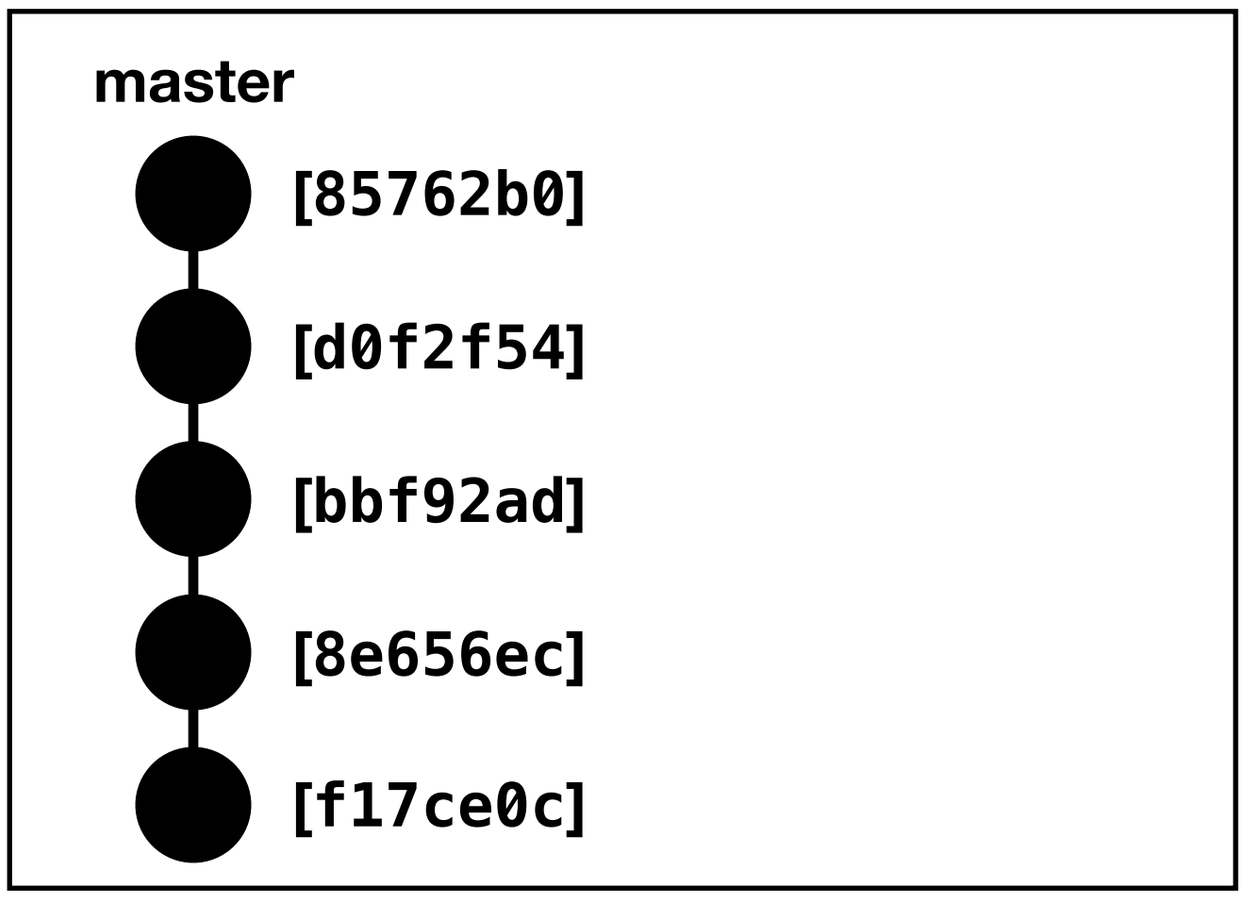
- When your teammates have reviewed and approved your code, merge it!



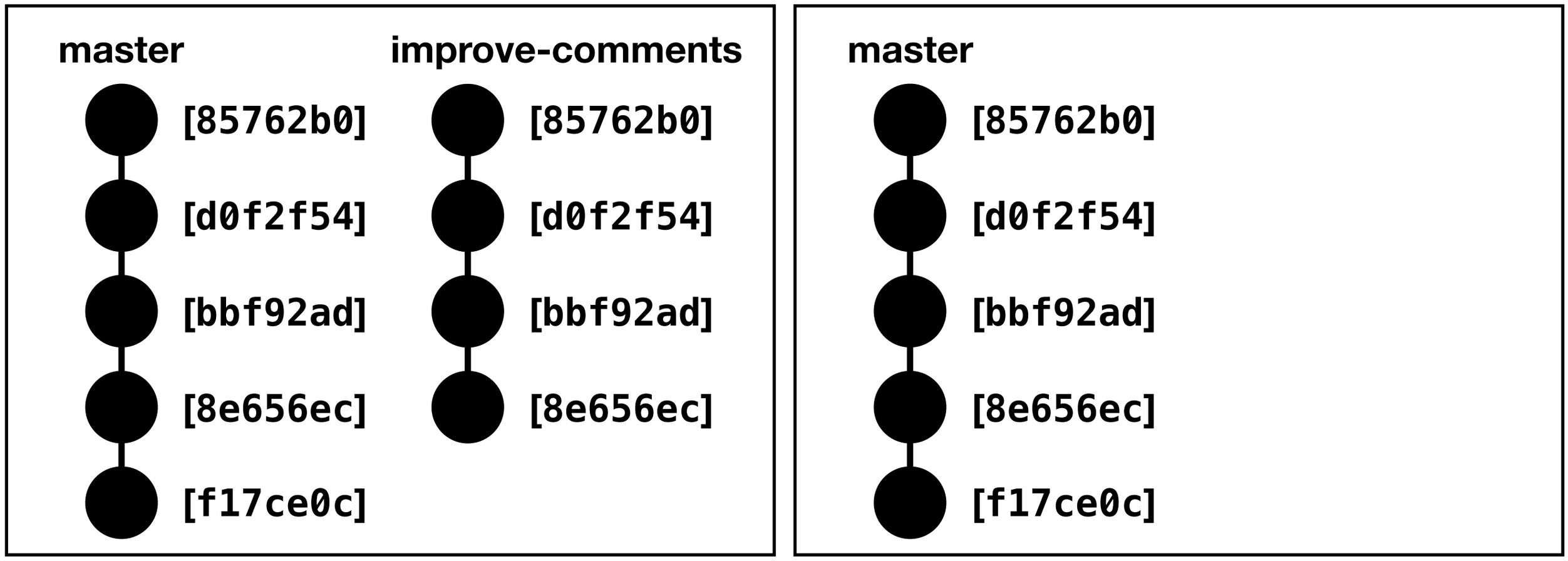
Local Computer



Remote Host



```
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(+)
```



```
hello_world (master)$ git branch -d improve-comments
Deleted branch improve-comments (was f17ce0c).
```

Local Computer

master

[85762b0]

[d0f2f54]

[bbf92ad]

[8e656ec]

[f17ce0c]

Remote Host

master

[85762b0]

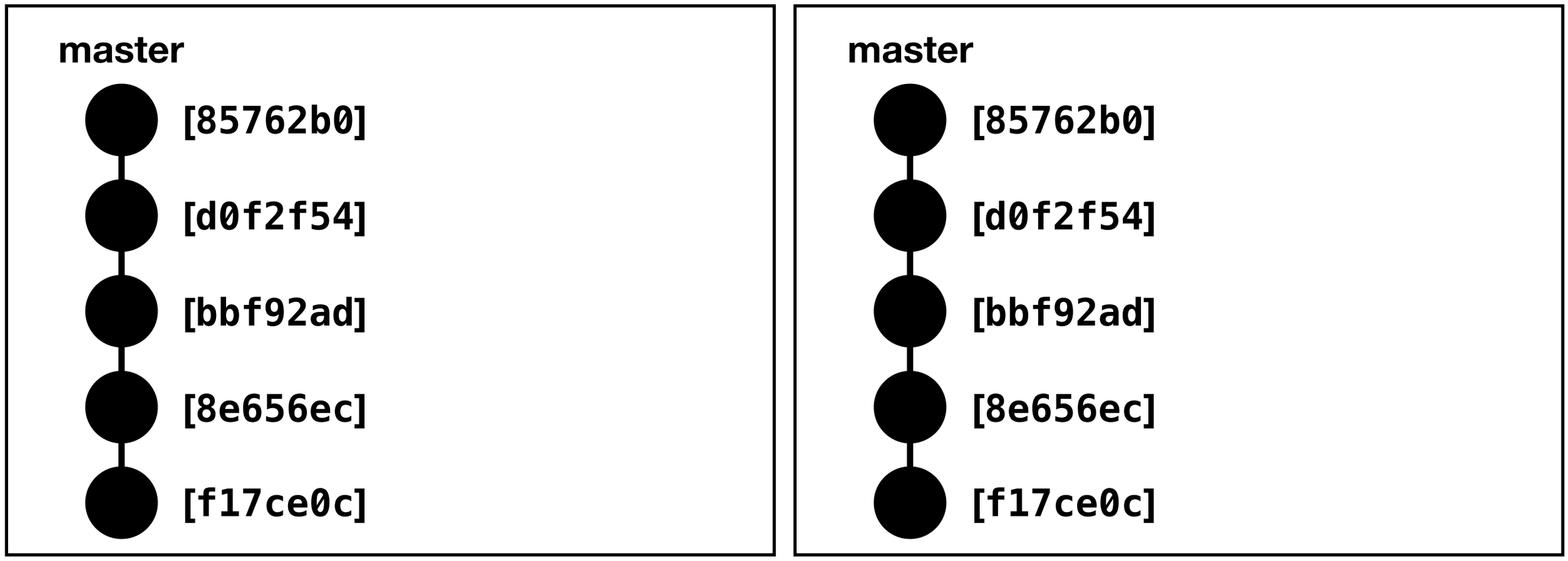
[d0f2f54]

[bbf92ad]

[8e656ec]

[f17ce0c]

```
hello_world (master)$ git log --pretty=oneline
f17ce0cd6b3b6346fbad01e0041309328bc18f29 Merge branch 'improve-comments' into 'master'
8e656ec5e368126486bbdb5f4af2f581bac7e391 Add doc comment to main function
bbf92ad2b0f4b859b6d76466358dfcfbfba475e3 Add README.md
d0f2f54b0979afe04f0654a69162927c9395217e Replace greeting with question
85762b00d392a49670cfd660749955f11c7dad9a Initial commit
```

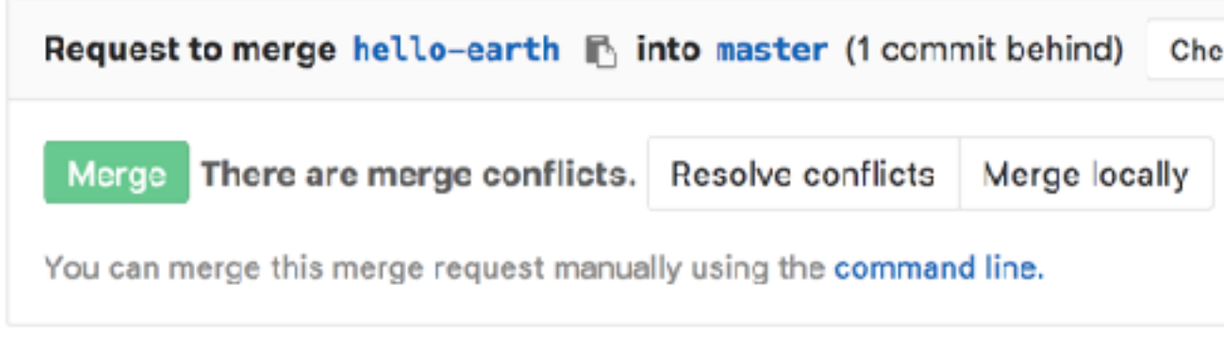


Branch-Review-Merge Workflow

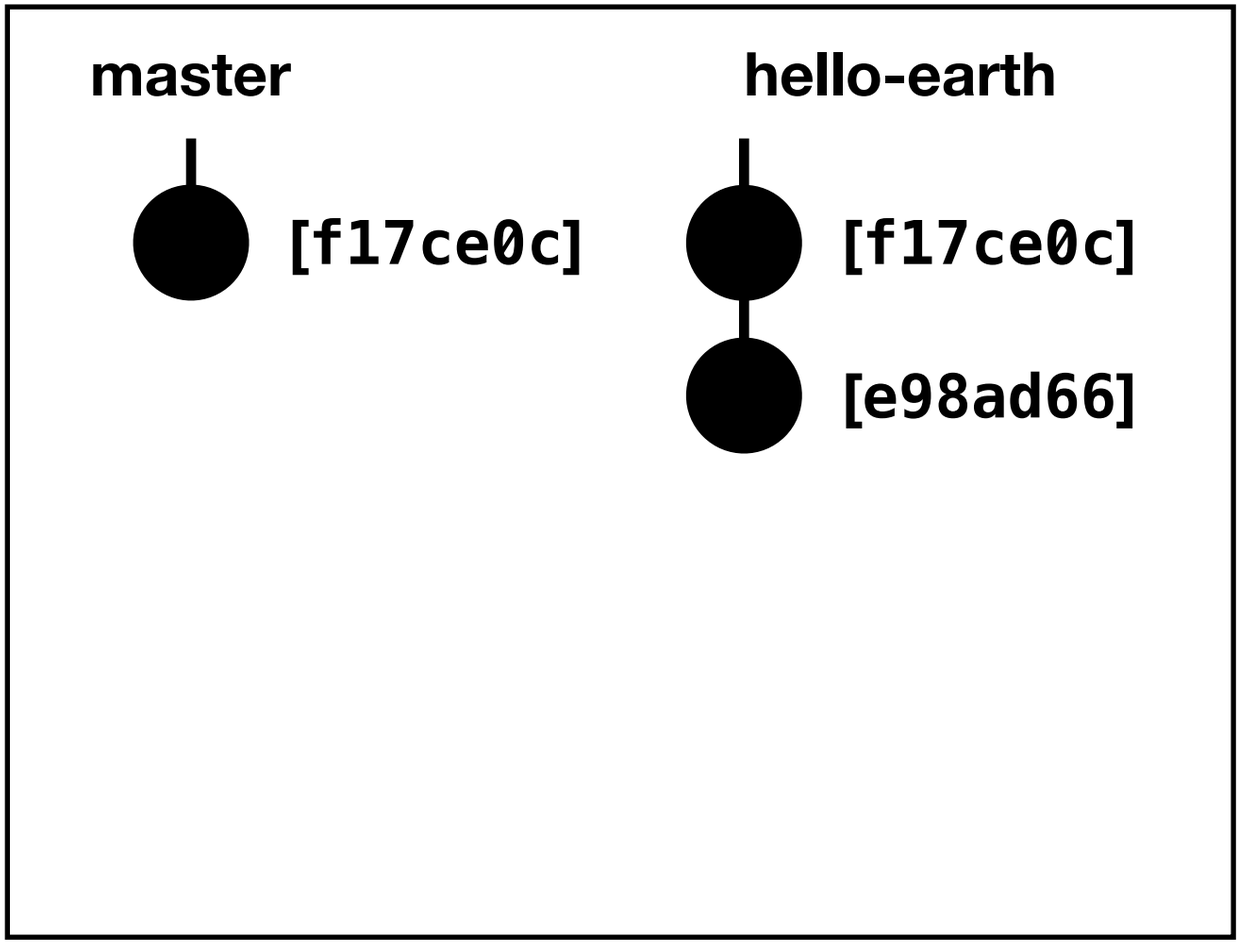
- Other Examples:
  - https://git.ece.iastate.edu/danc/MicroCART/merge_requests/11

Branch-Review-Merge Workflow

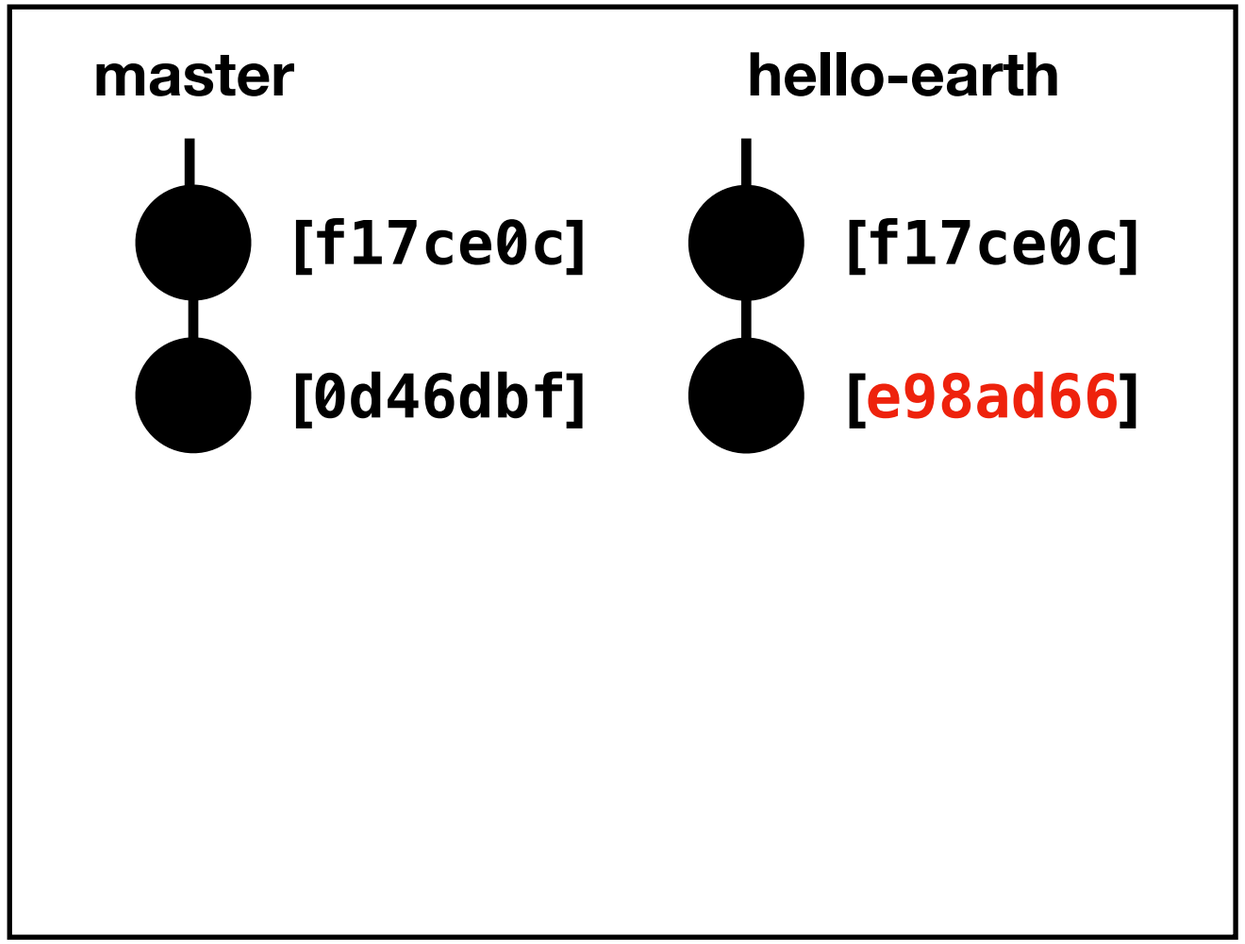
- What about merge conflicts?



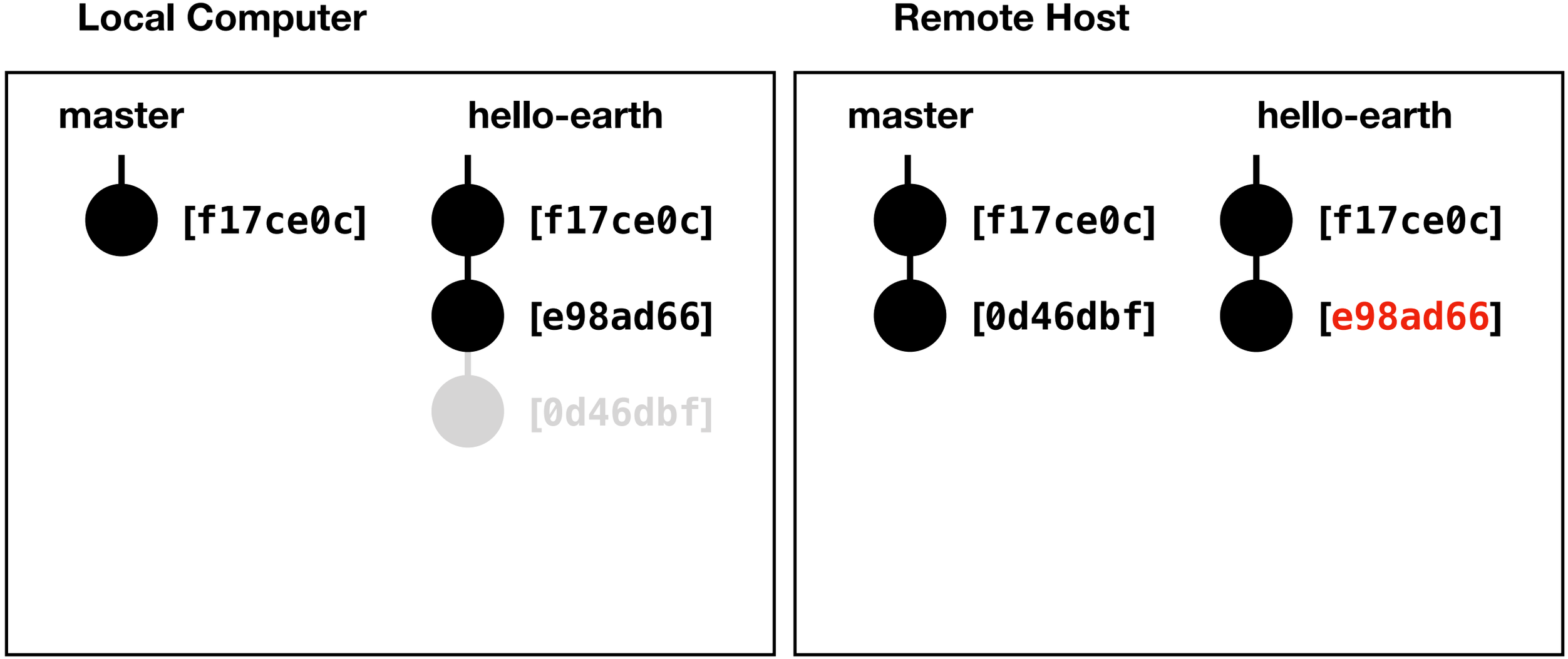
Local Computer



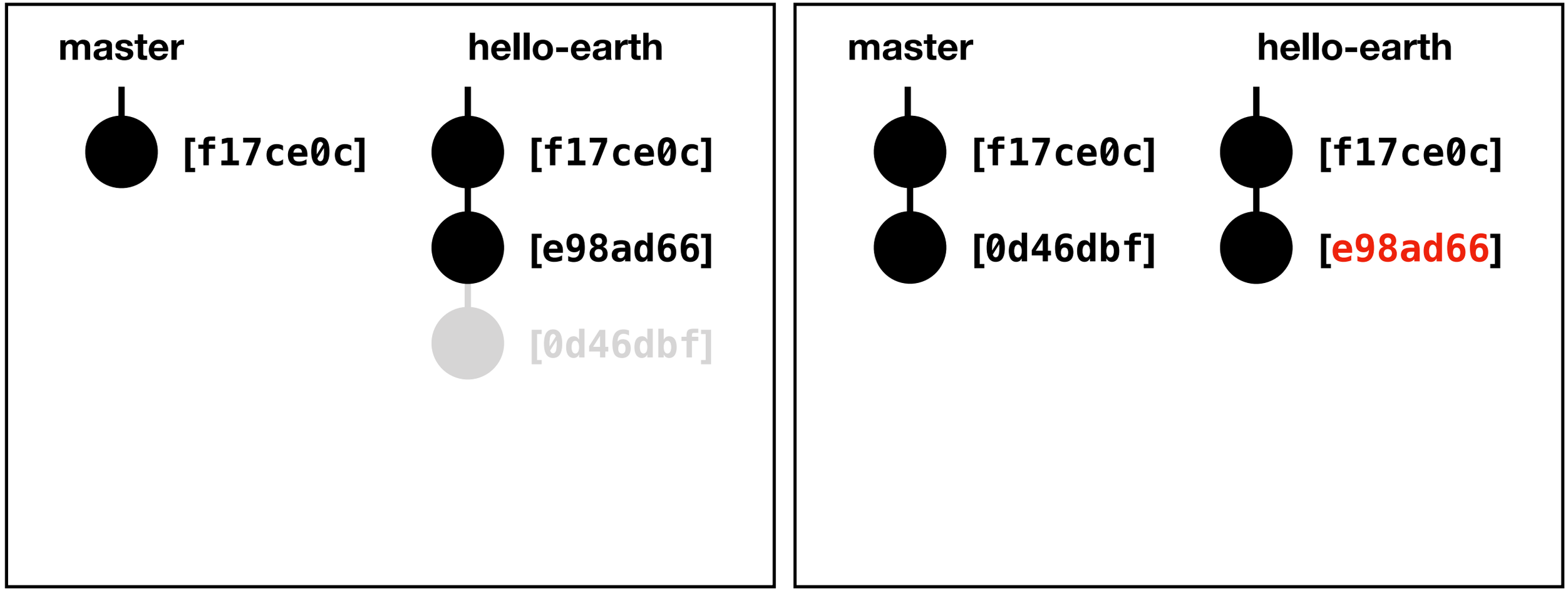
Remote Host



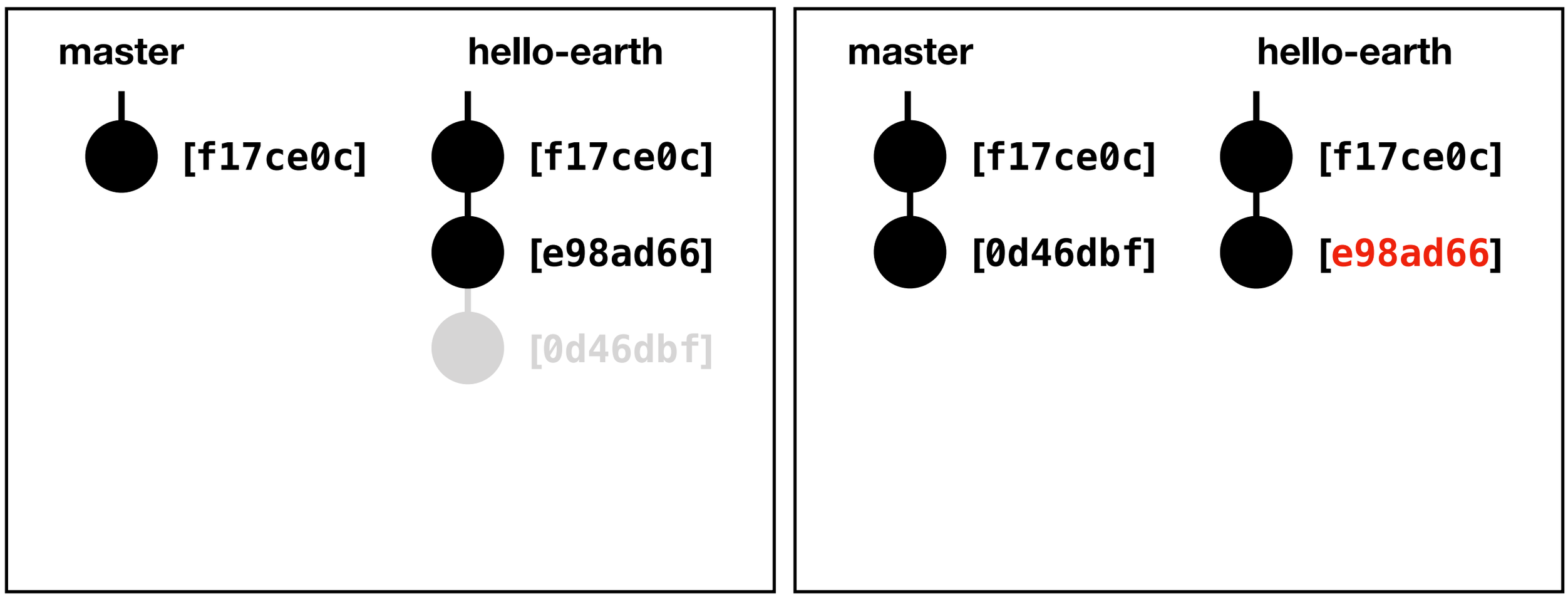
```
hello_world (hello-earth)$ git pull origin master
From git.ece.iastate.edu:bbartels/example-hello-world
 * branch            master     -> FETCH_HEAD
Auto-merging src/main.rs
CONFLICT (content): Merge conflict in src/main.rs
Automatic merge failed; fix conflicts and then commit the result.
hello_world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



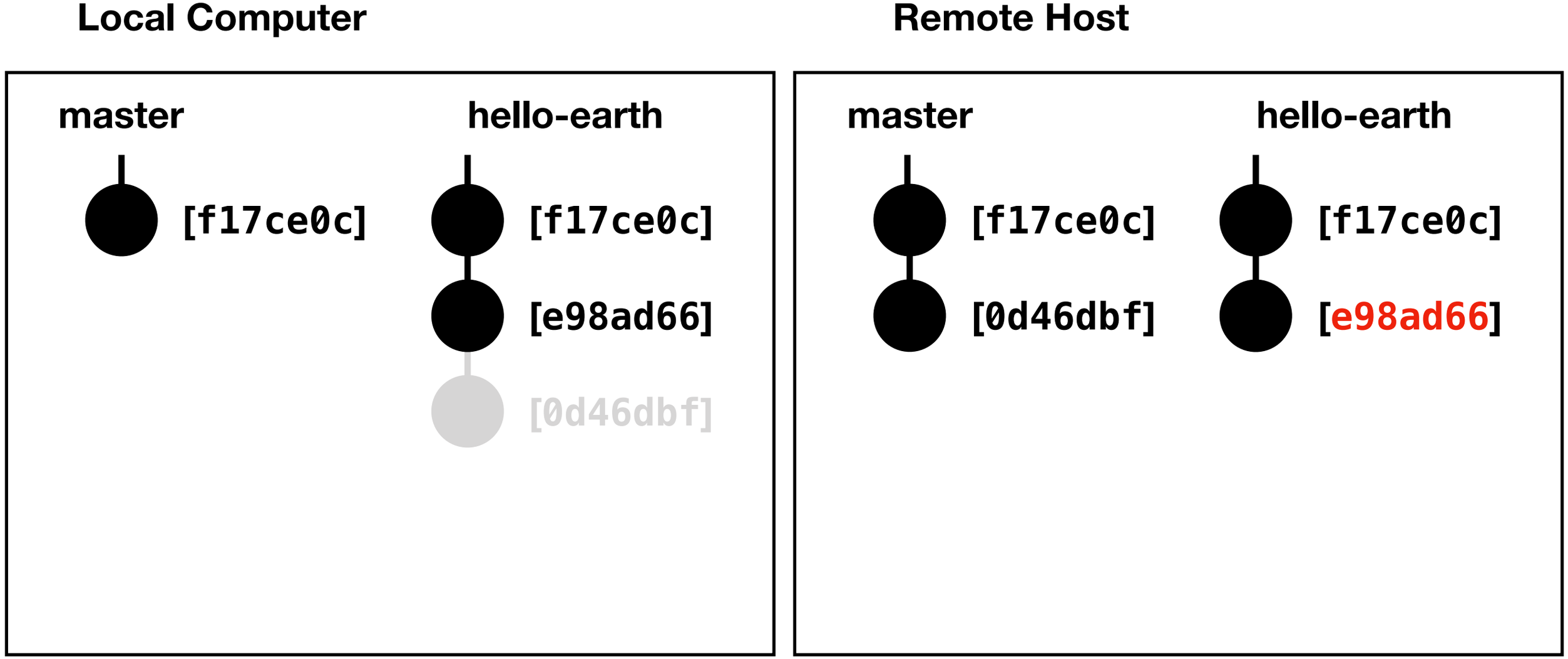
```
hello_world (hello-earth|MERGING)$ git status
On branch hello-earth
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

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")
```



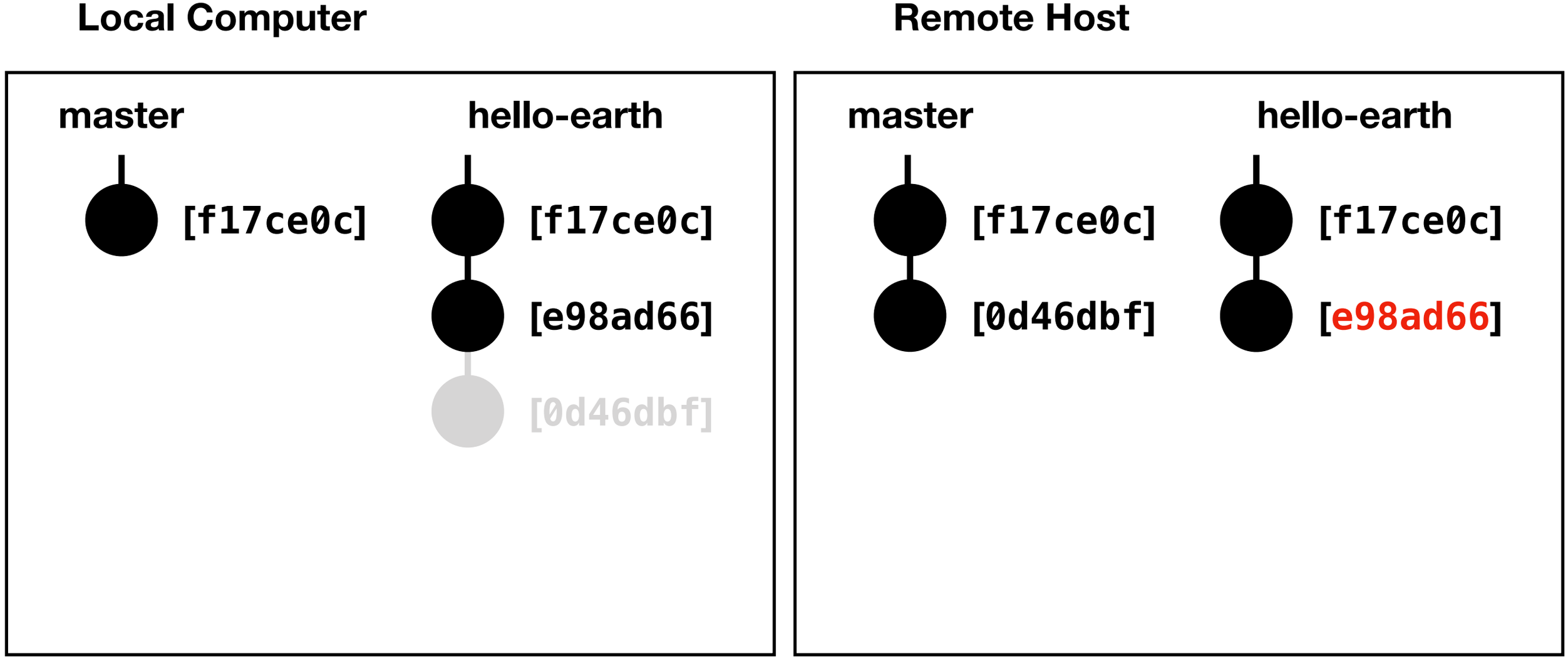
```
File Edit Options Buffers Tools Help
 1// The main function
 2fn main() {
 3<<<<<<< HEAD
 4    println!("What's up, earth?");
 5||||||| merged common ancestors
 6    println!("What's up, world?");
 7=======
 8    println!("Hello, world!");
 9>>>>>>> 0d46dbfb7b7878222e8404354e48dd26a1950270
10}
```



```
File Edit Options Buffers Tools Help
1// The main function
2fn main() {
3    println!("Hello, earth!");
4}
```

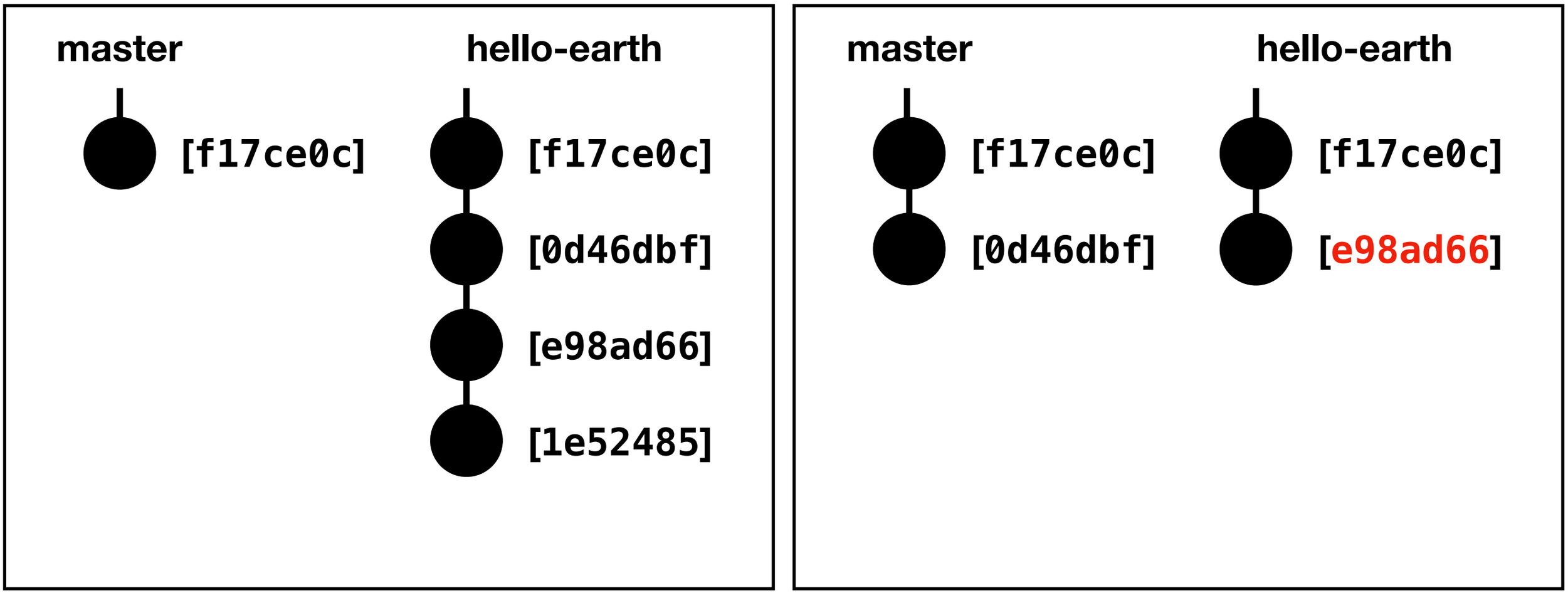


```
hello_world (hello-earth|MERGING)$ git status
On branch hello-earth
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

        modified:   src/main.rs
```

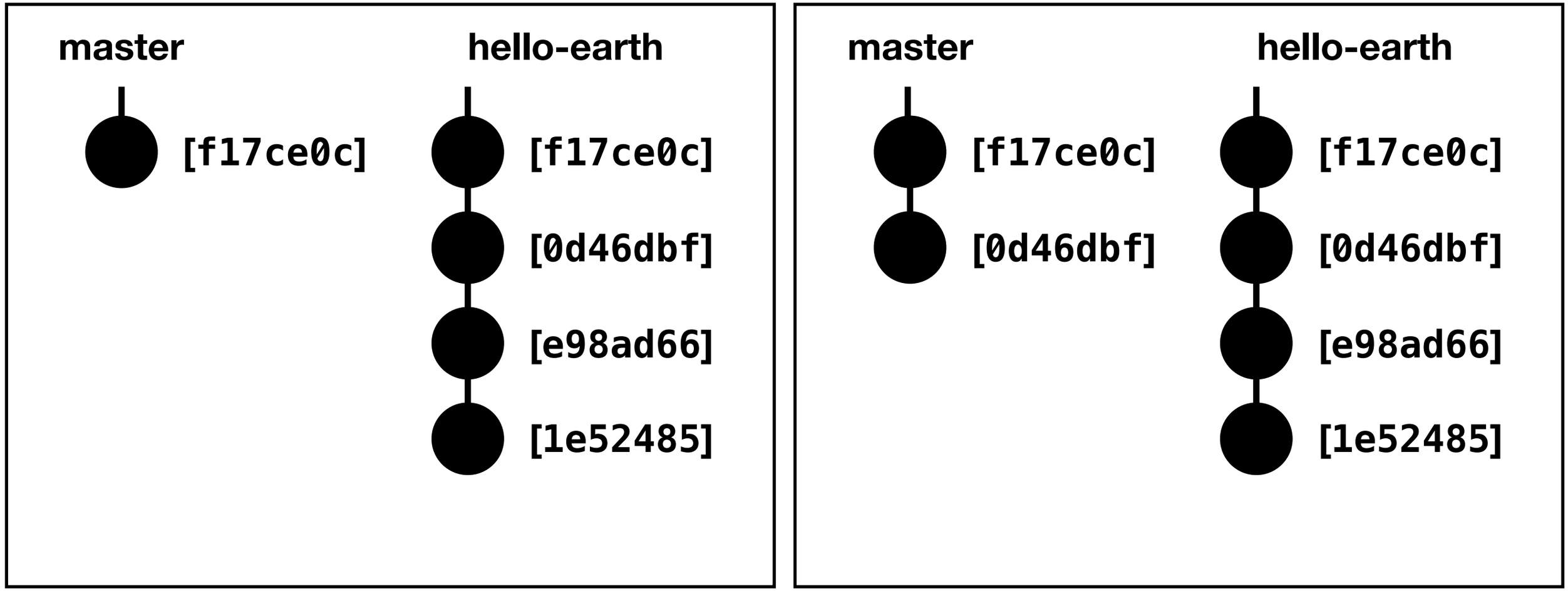


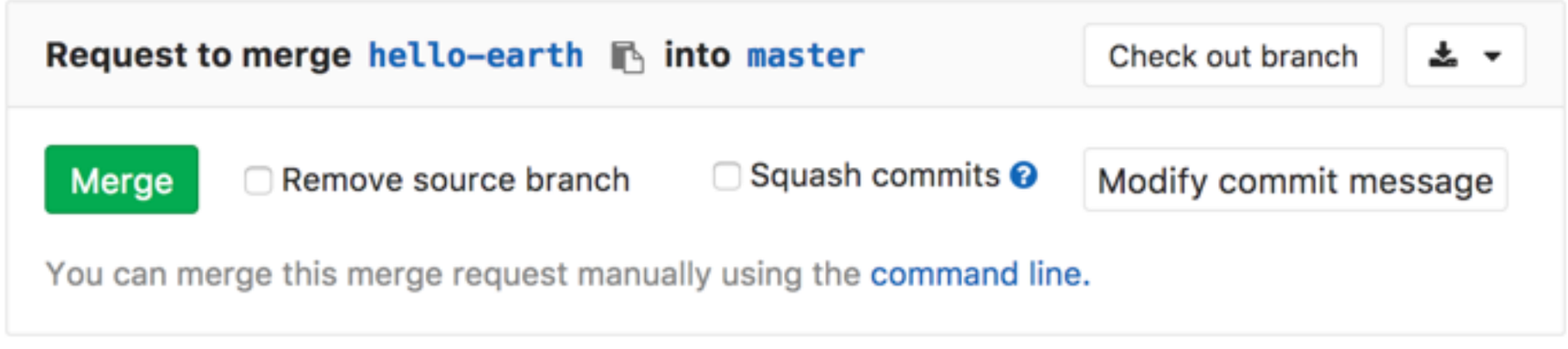
```
hello_world (hello-earth|MERGING)$ git commit
[hello-earth 1e52485] Merge branch 'master' of git.ece.iastate.edu:bbartels/example-hello-
world into hello-earth
```

- Git will likely open a text-editor (vim) for the commit
  - Use “:q” to just quit and use the commit message as is

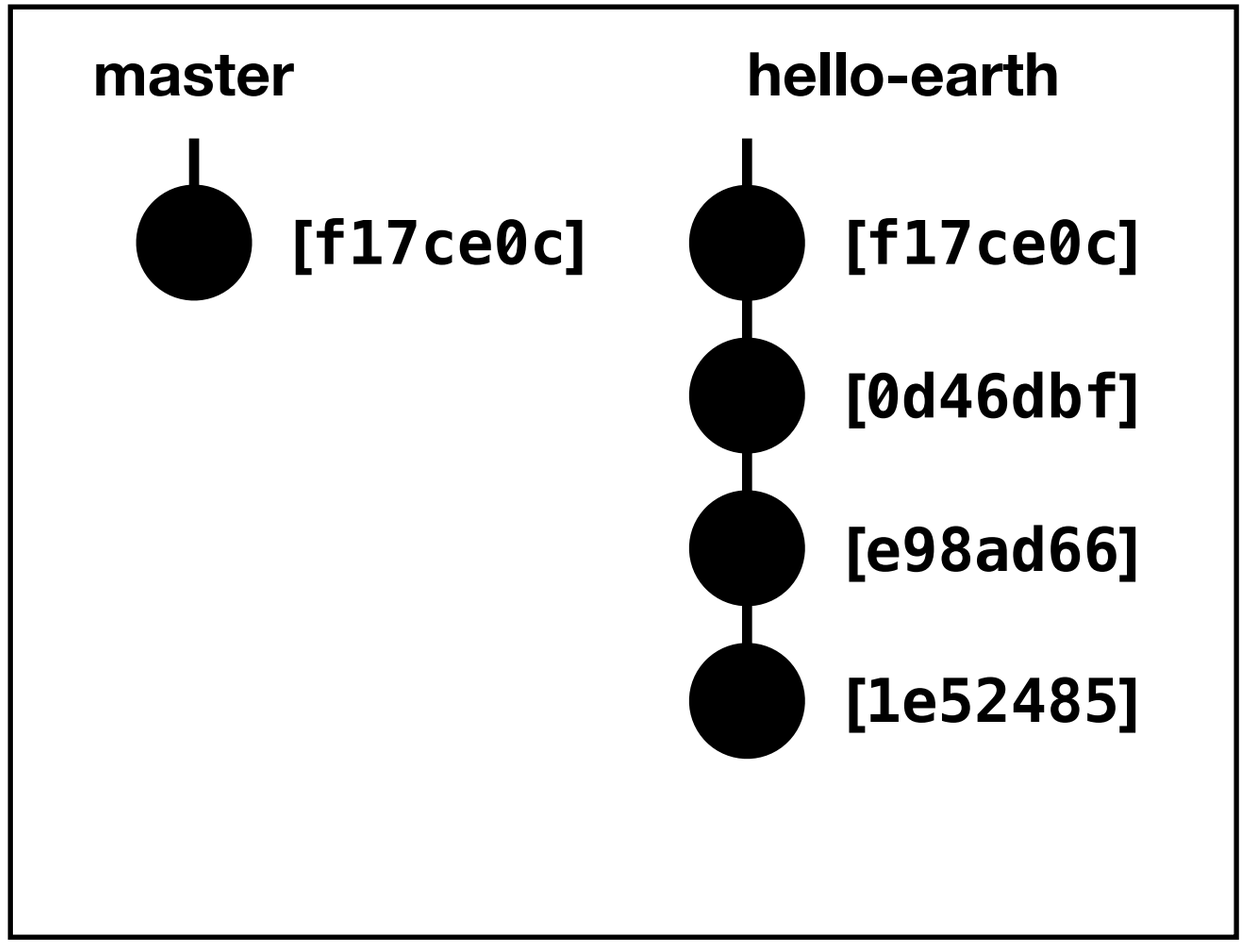


```
hello_world (hello-earth)$ git push origin hello-earth
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 436 bytes | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote:
remote: View merge request for 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
```

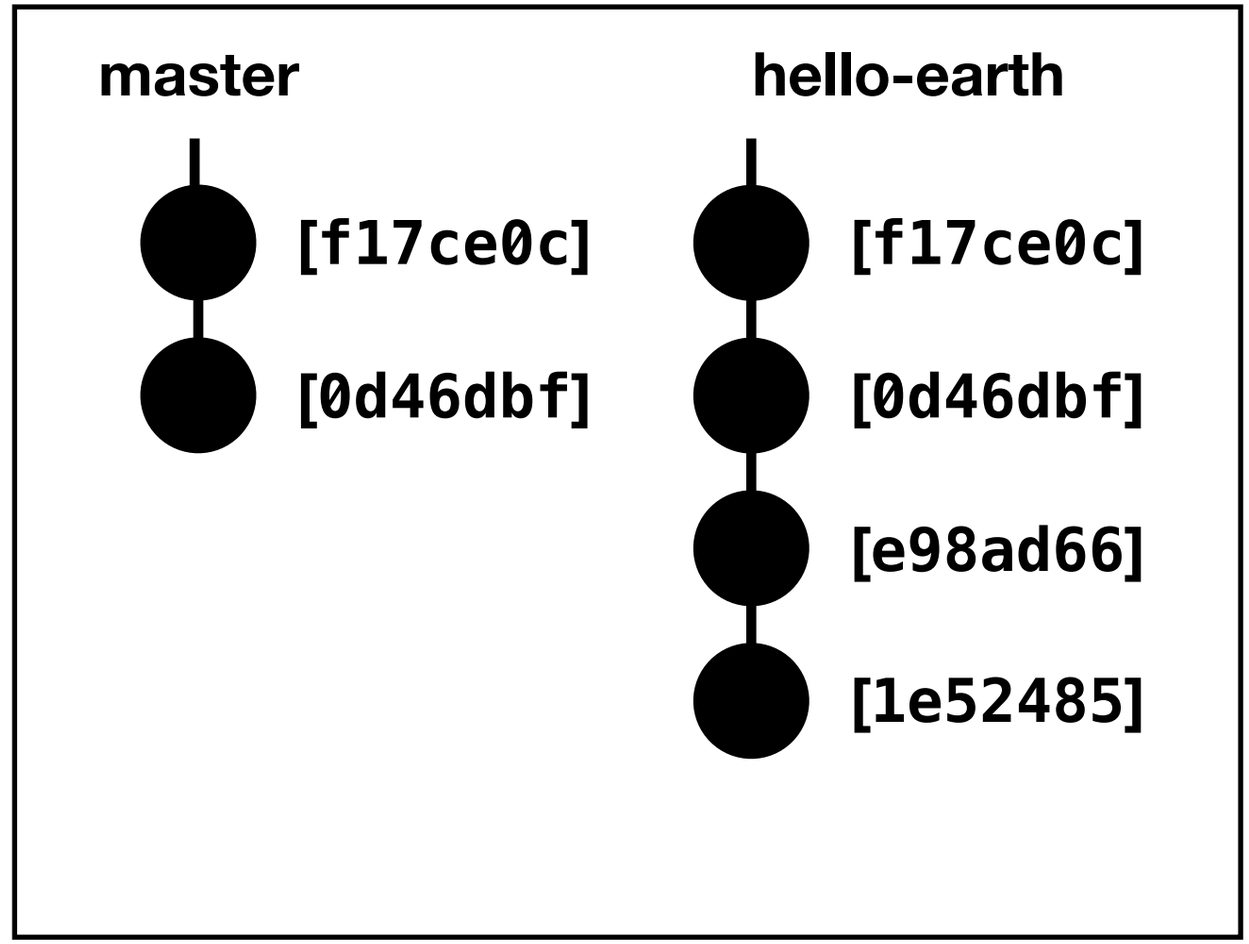


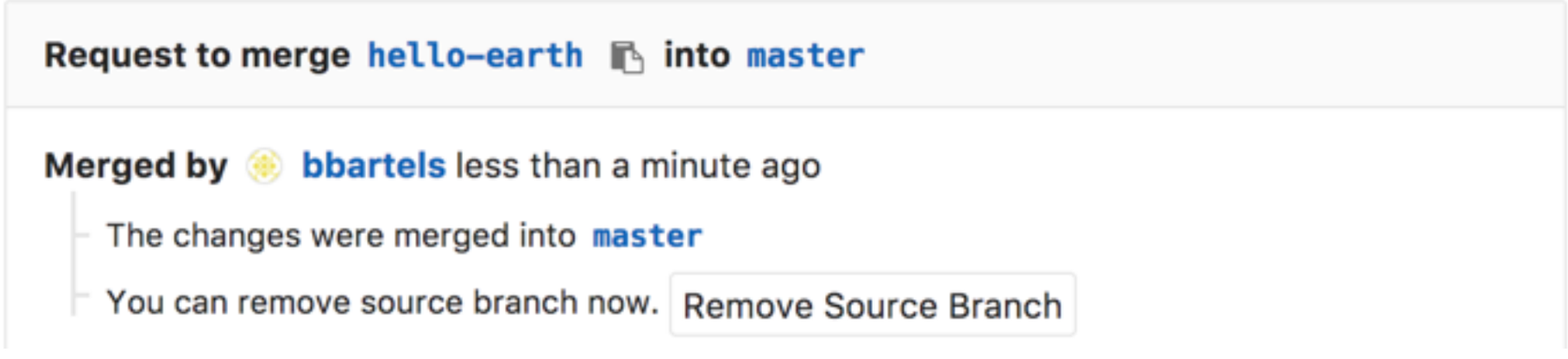


Local Computer

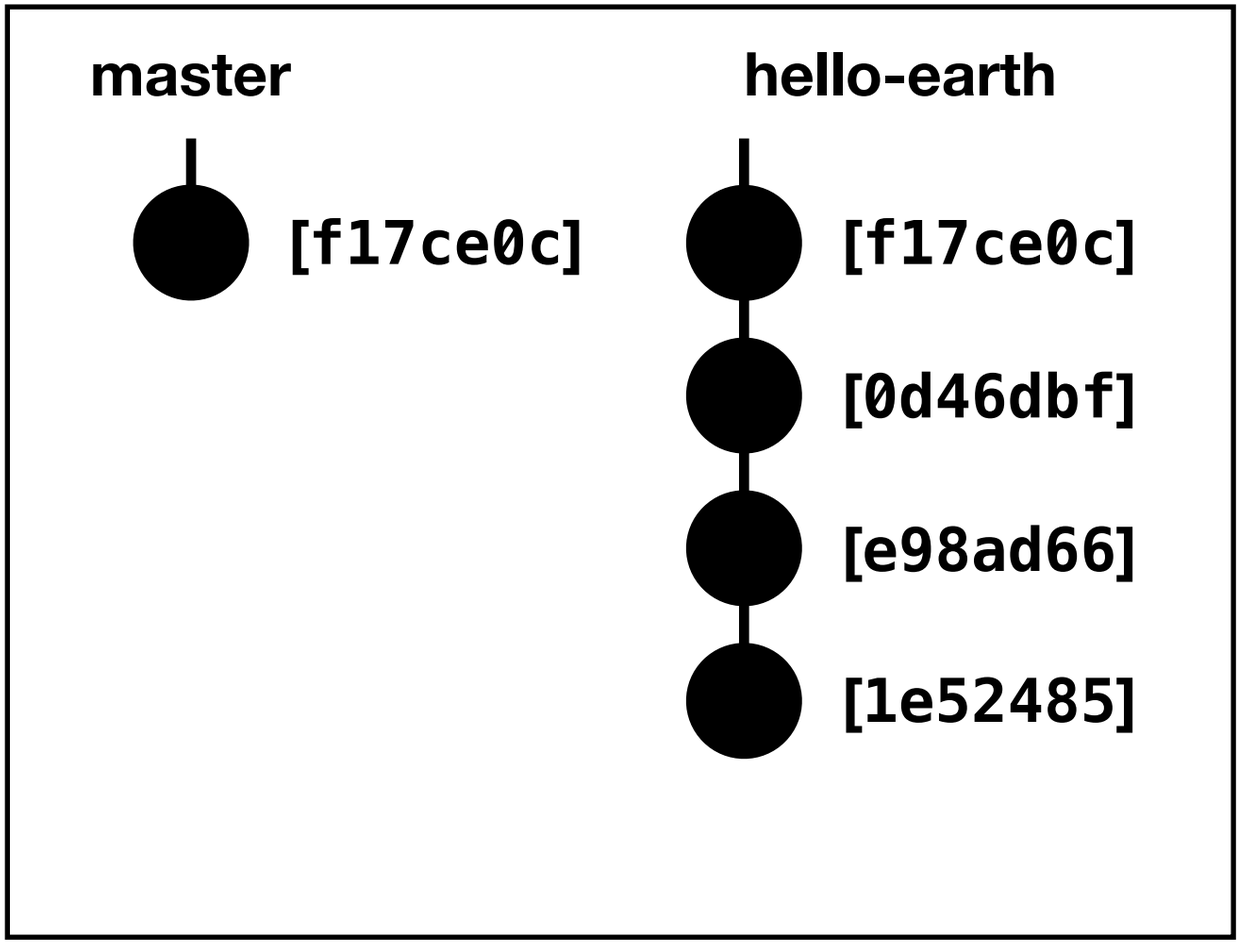


Remote Host

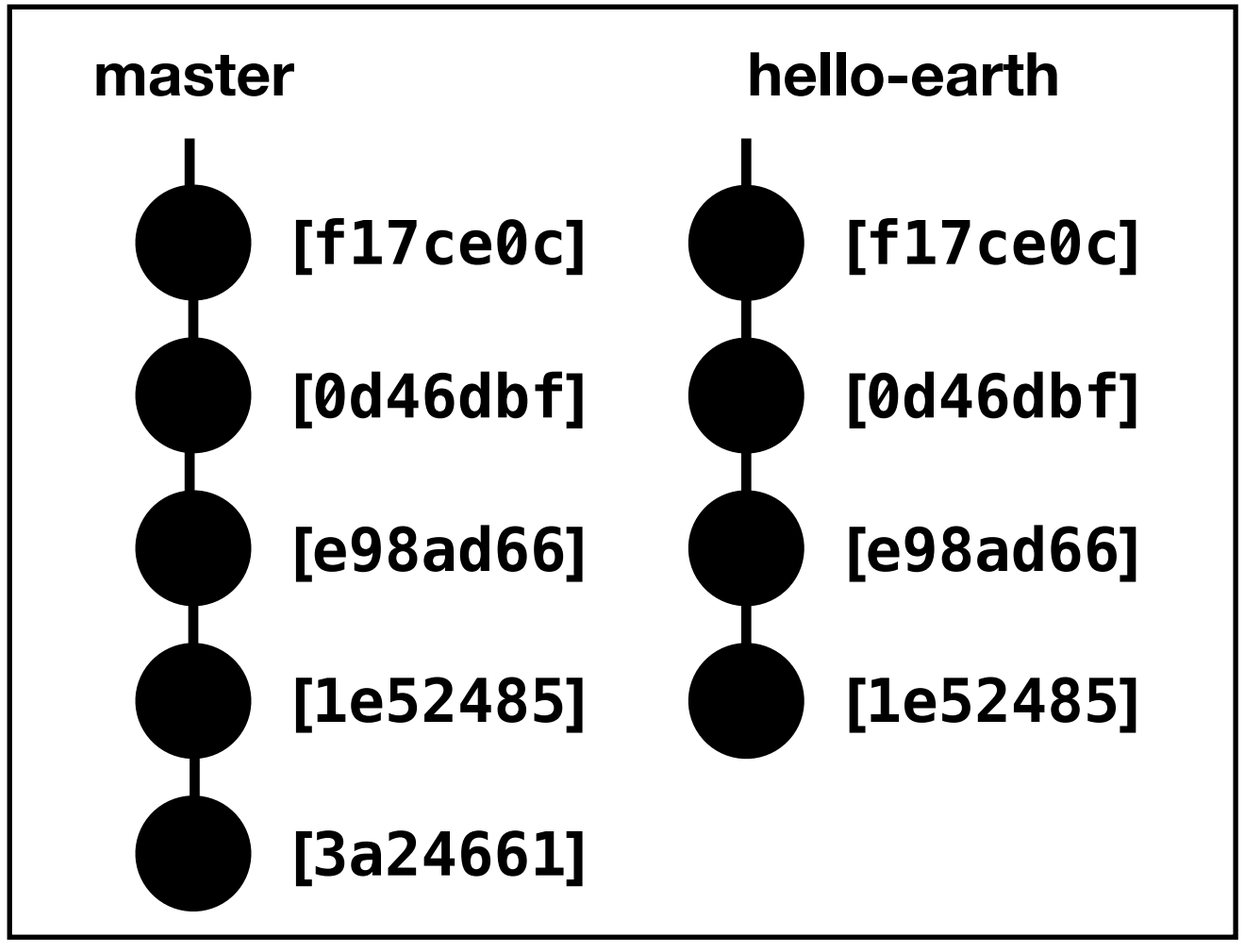




Local Computer



Remote Host



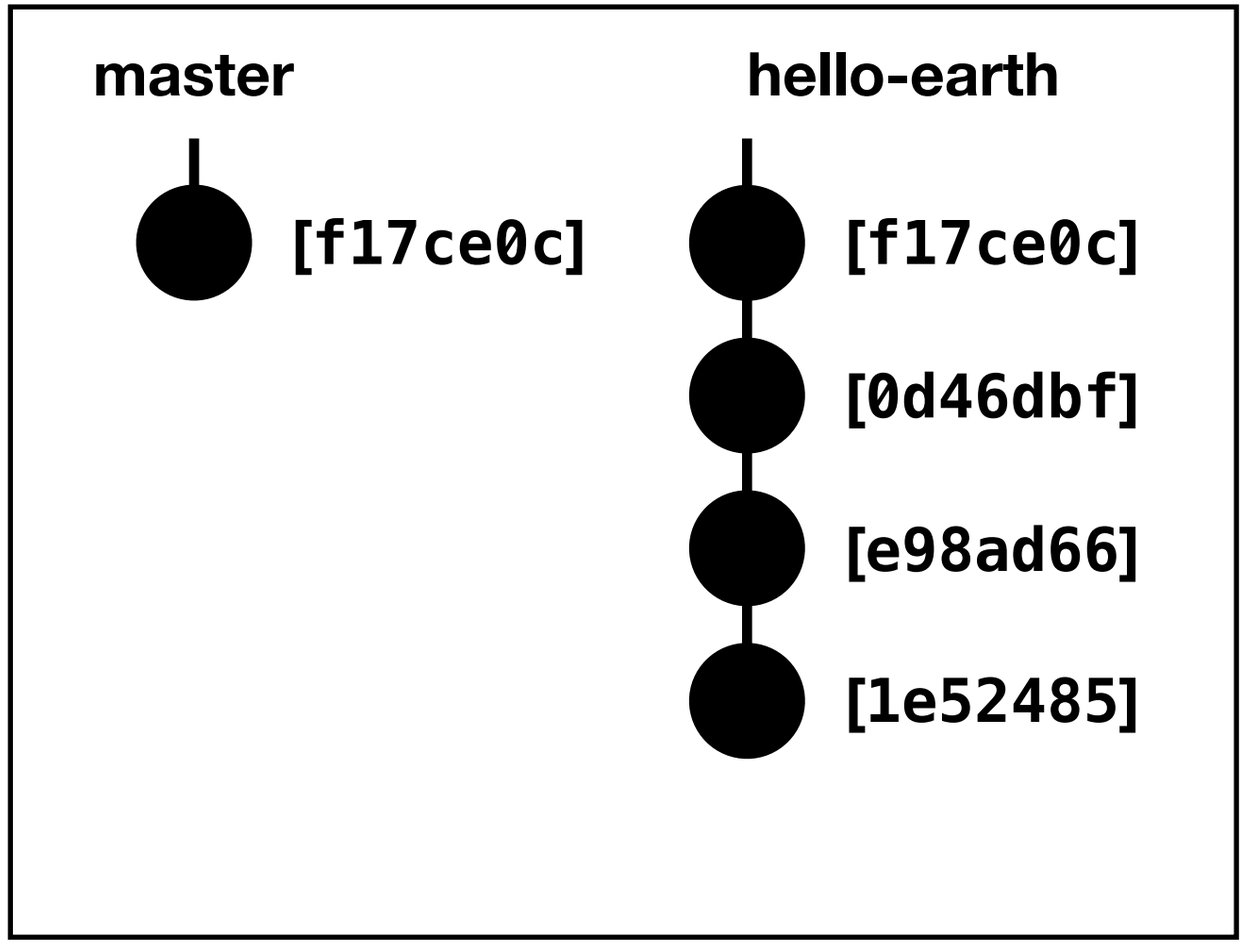
Request to merge `hello-earth` **into** `master`

Merged by bbartels about a minute ago

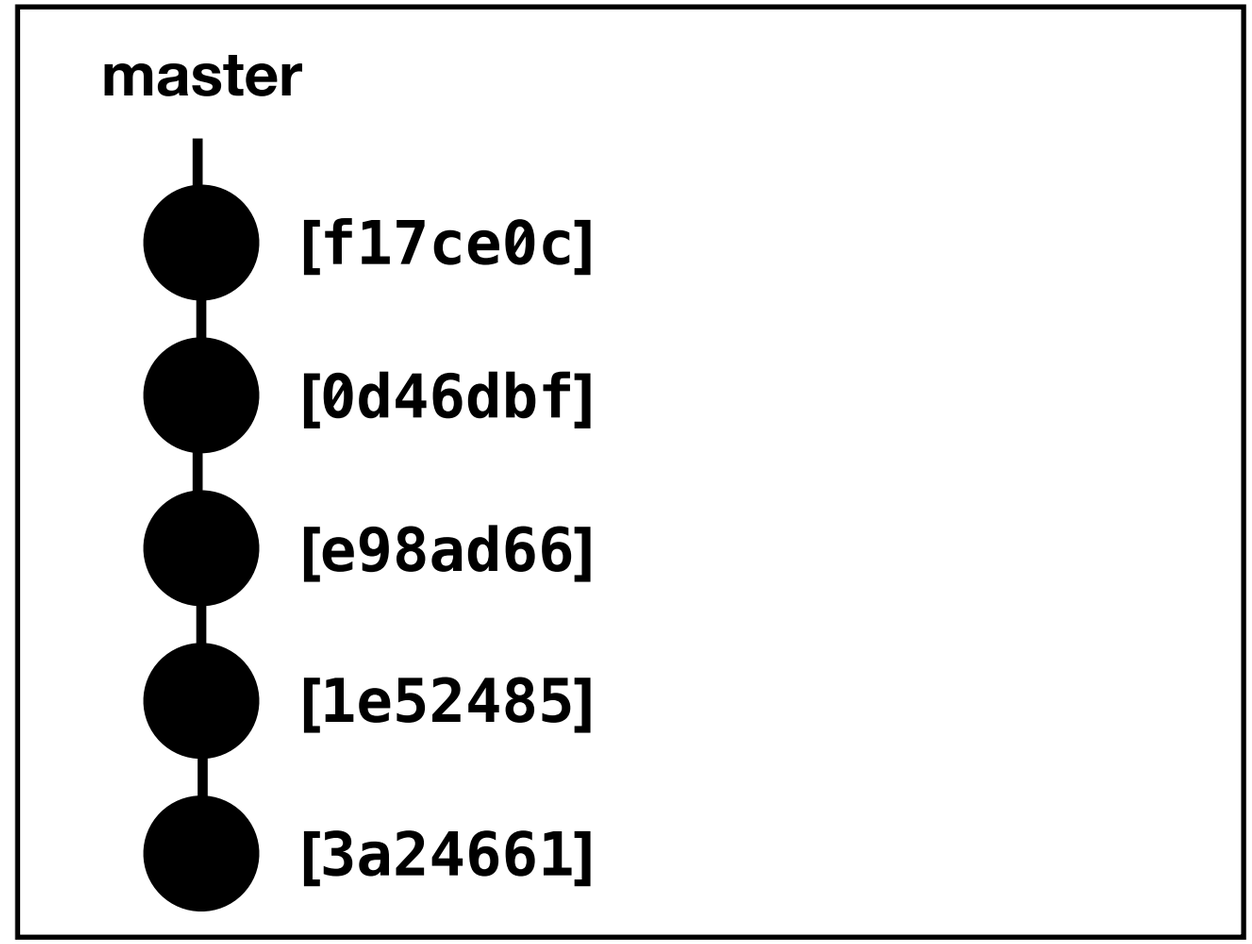
The changes were merged into `master`

The source branch has been removed.

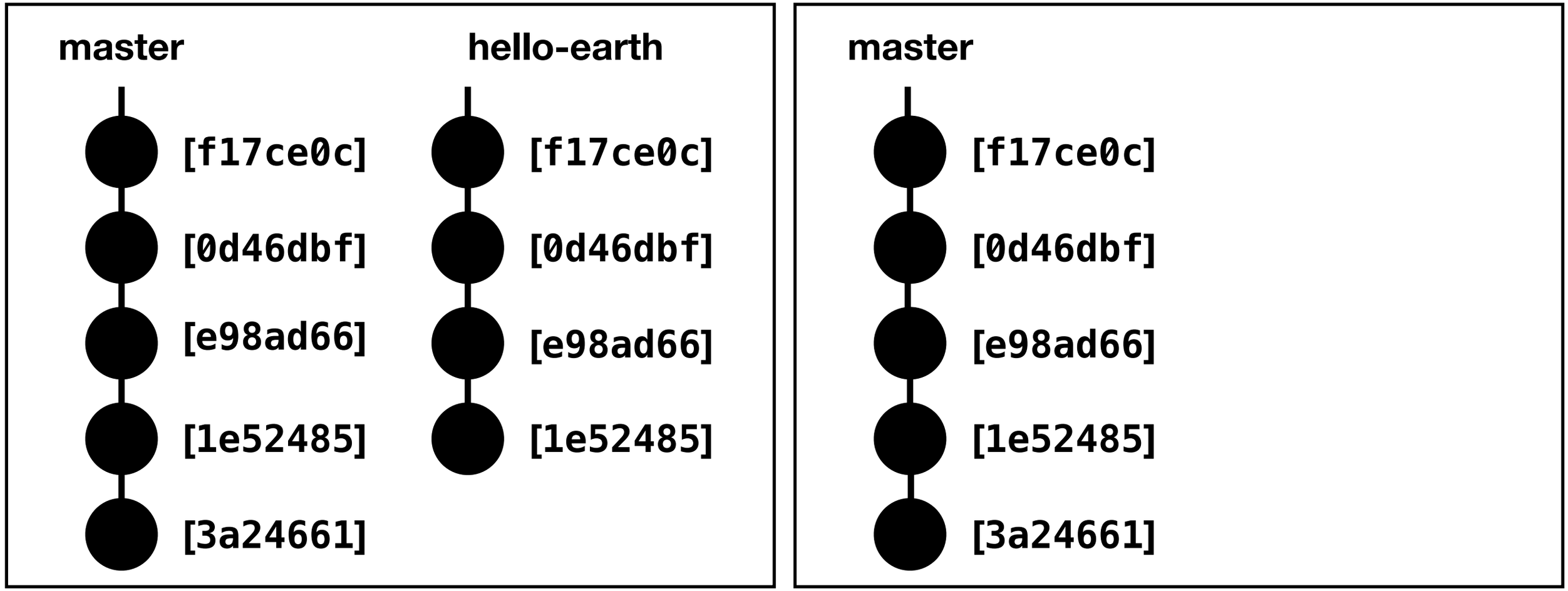
Local Computer



Remote Host

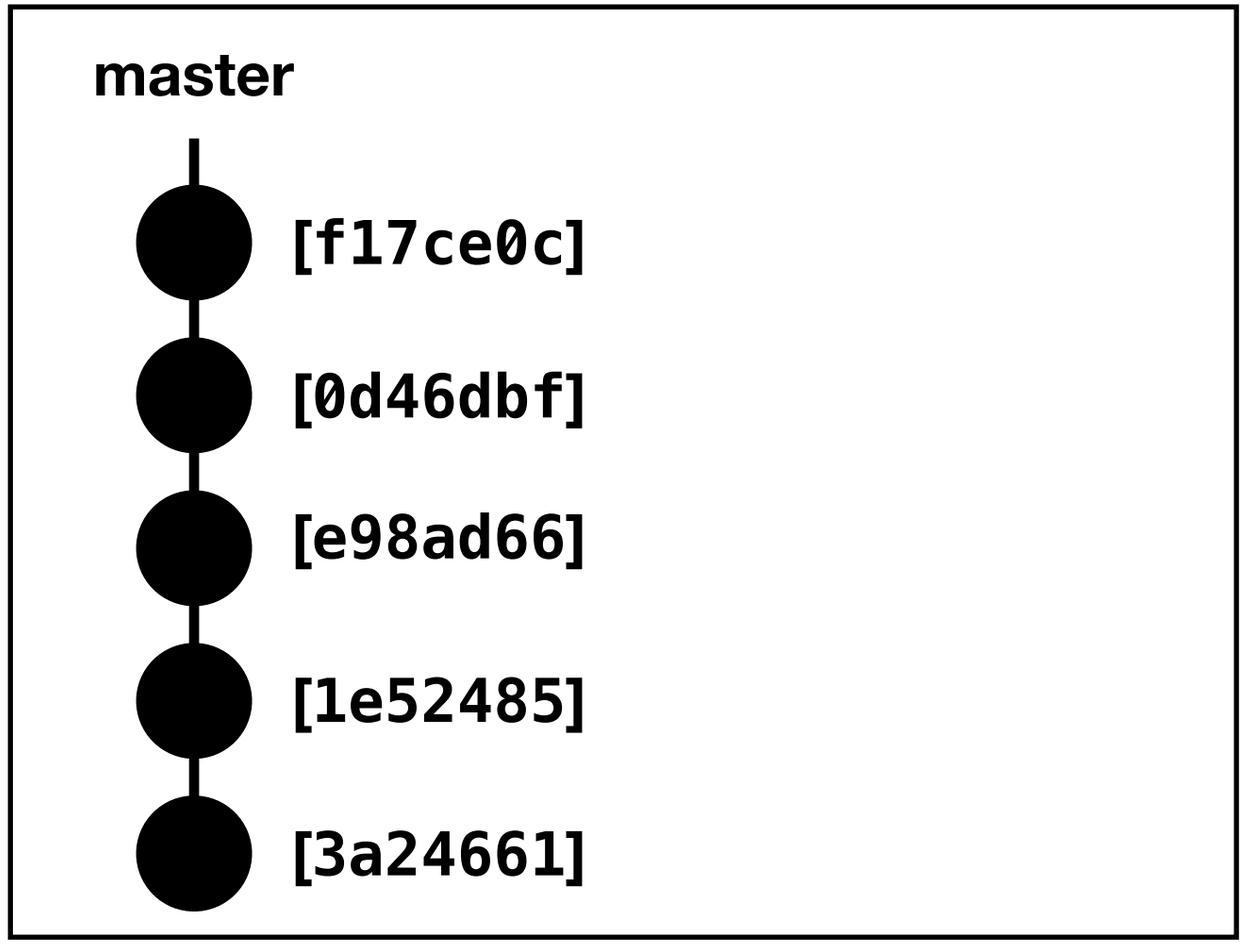


```
hello_world (hello-earth)$ git checkout master
Switched to branch 'master'
hello_world (master)$ git pull origin master
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 1 (delta 0)
Unpacking objects: 100% (1/1), done.
From git.ece.iastate.edu:bbartels/example-hello-world
 * branch            master     -> FETCH_HEAD
   0d46dbf..3a24661  master     -> origin/master
Updating 0d46dbf..3a24661
Fast-forward
 src/main.rs | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

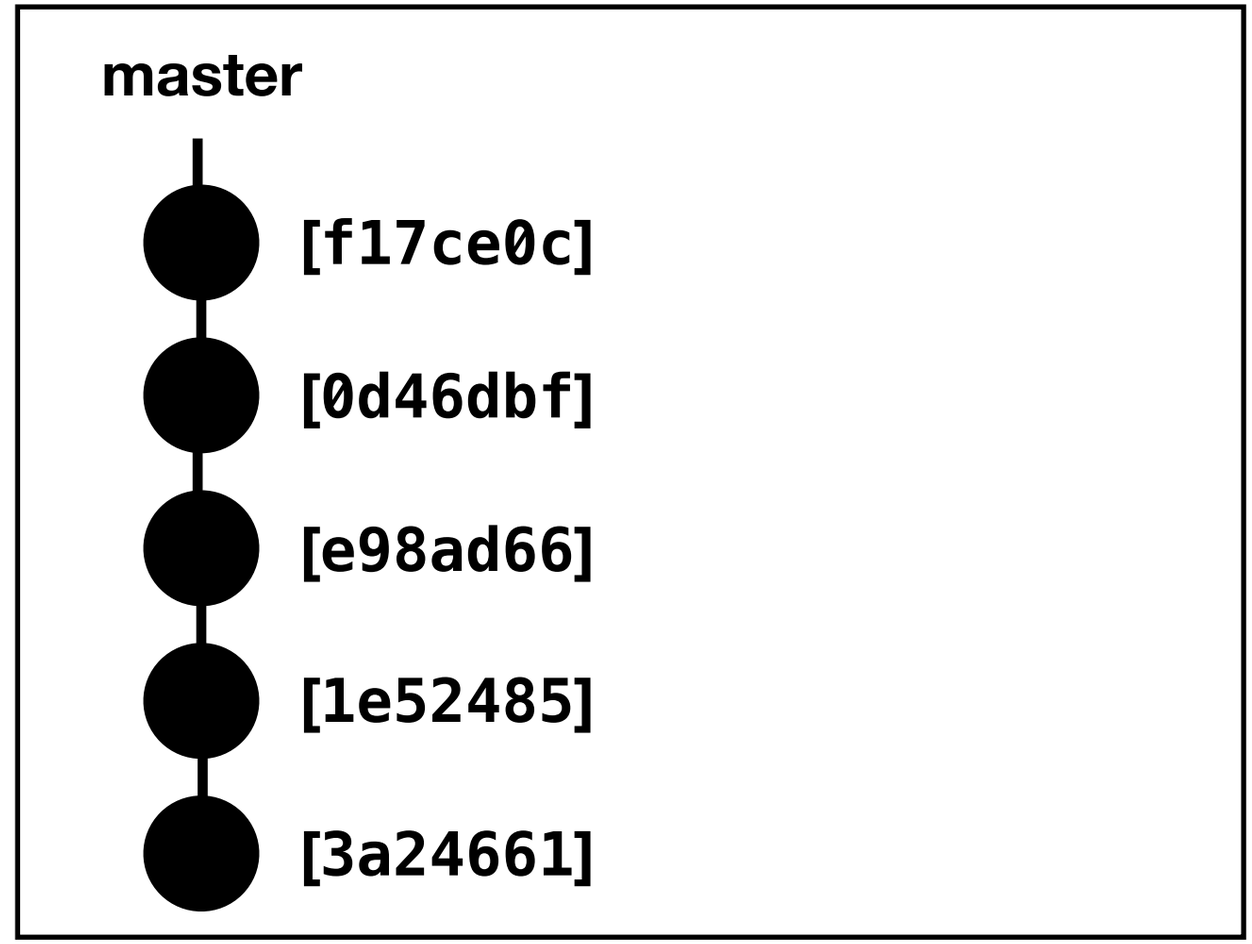


```
hello_world (master)$ git branch -d hello-earth
Deleted branch hello-earth (was 1e52485).
```

Local Computer

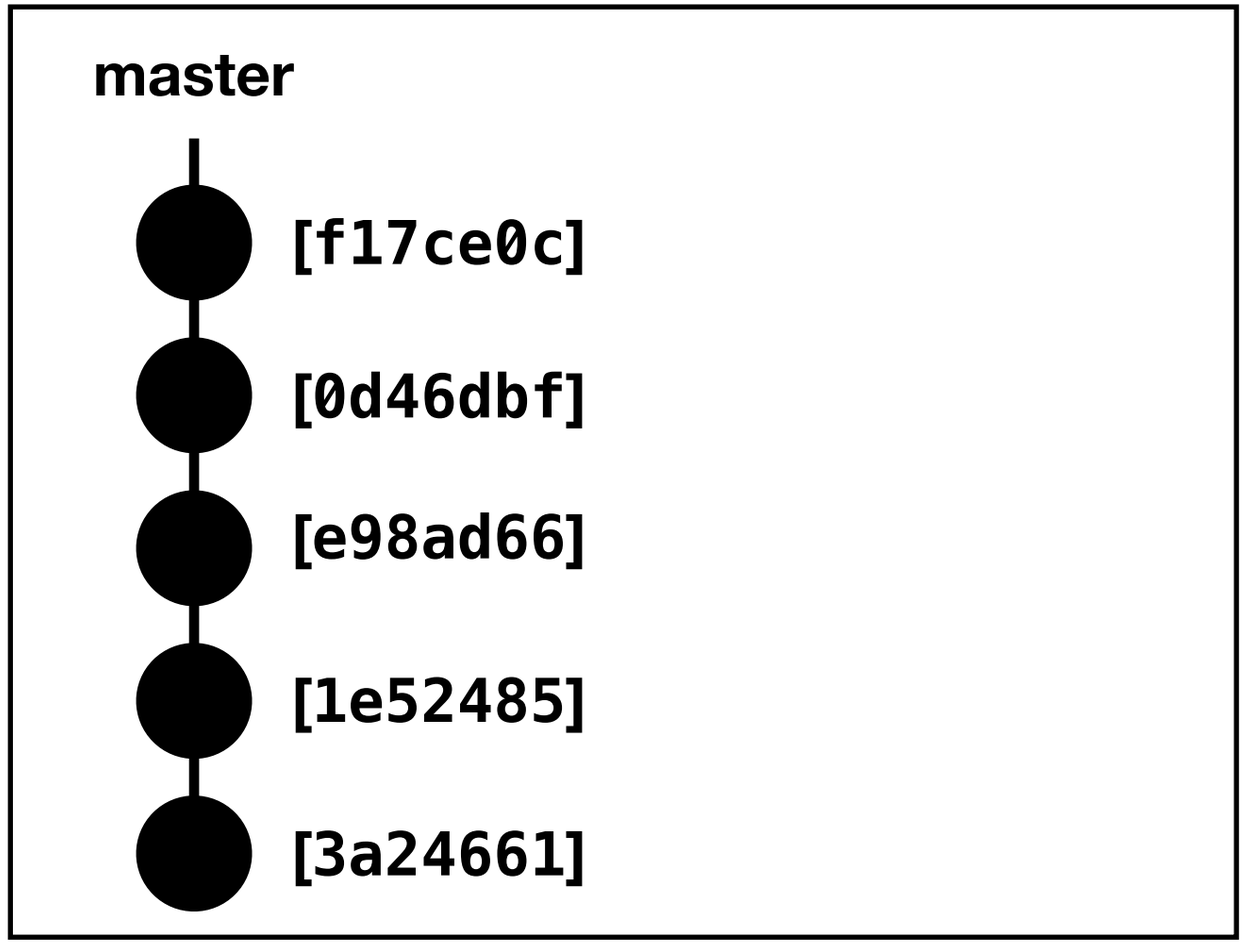


Remote Host

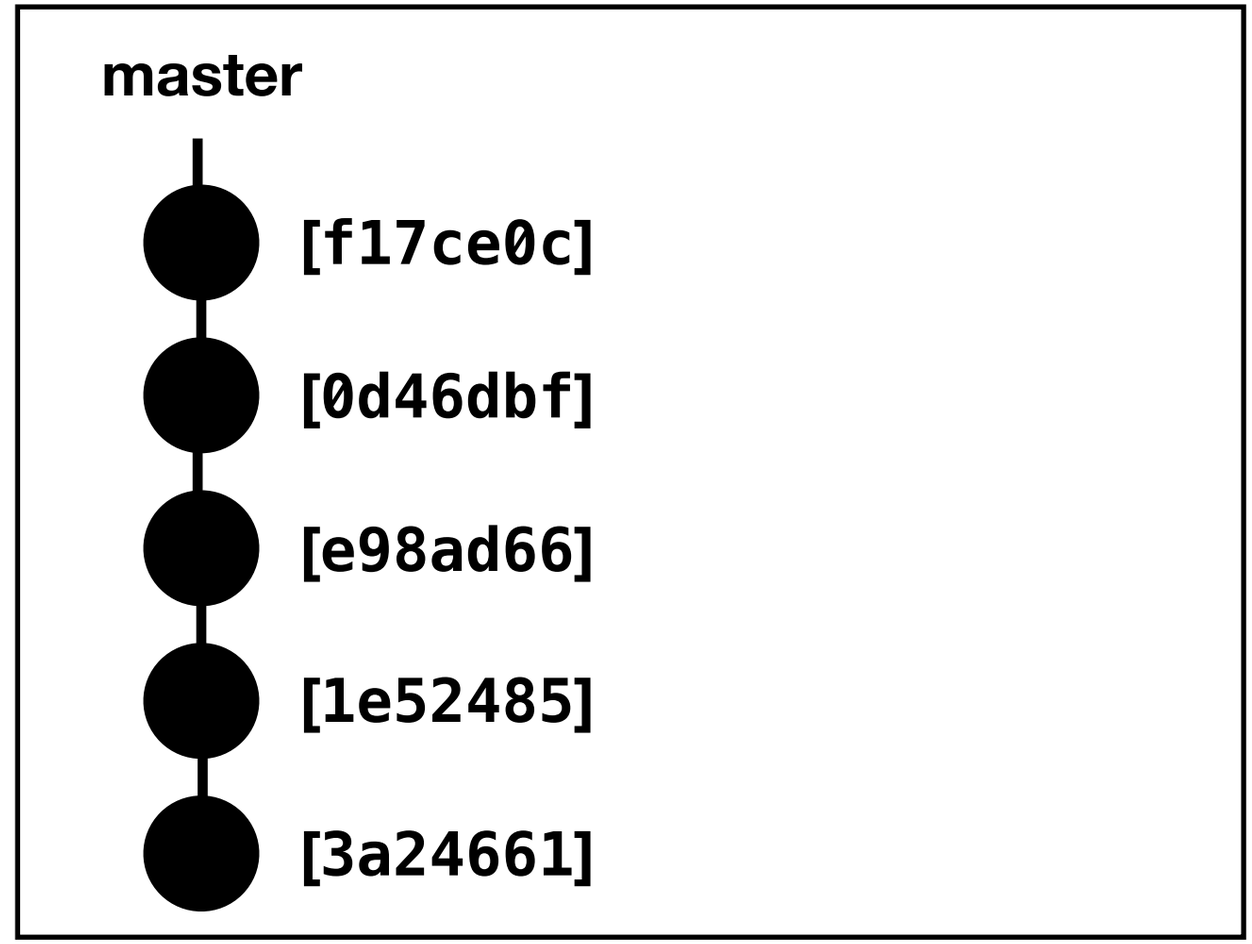


```
hello_world (master)$ git log --pretty=oneline
3a246611411a9ca9de8ad6aa9571586501046b90 Merge branch 'hello-earth' into 'master'
1e52485a53cc699479766e5ed42caab7272c68b7 Merge branch 'master' of git.ece.iastate.edu:bbart
e98ad66e5468ee921fcbfb14b7e688ded9fb92f2 Change subject from world to earth
0d46dbfb7b7878222e8404354e48dd26a1950270 Revert "Replace greeting with question"
f17ce0cd6b3b6346fbad01e0041309328bc18f29 Merge branch 'improve-comments' into 'master'
8e656ec5e368126486bbdb5f4af2f581bac7e391 Add doc comment to main function
bbf92ad2b0f4b859b6d76466358dfcfbfba475e3 Add README.md
d0f2f54b0979afe04f0654a69162927c9395217e Replace greeting with question
85762b00d392a49670cfd660749955f11c7dad9a Initial commit
```

Local Computer



Remote Host



Branch-Review-Merge Workflow FAQ

- What about files generated by software? (not directly written by you)
    - Can 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

- Web Pages
    - 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**
    - Gitlab's Markdown Reference:
        - https://docs.gitlab.com/ee/user/markdown.html

```
[![build status](https://git.ece.iastate.edu/danc/MicroCAR
```

MicroCART

Microprocessor Controlled Aerial Robotics Team

Project Statement: "To create a modular platform for re

Since 1998, MicroCART has been building aerial robots. Cur
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, incor
    Zybo board that provides the processor and a FPGA, an I
    props, a LiPo battery, a receiver for manual remote con
    that holds it all together.
- The **ground station**
  - The ground station is responsible for issuing important
    (like position data from the camera system) and issuing
    for things like configuration and path following direct
- The **controls model**
  - The quadcopter processor is programmed to implement a P
    a Simulink model to streamline the PID tuning process a
    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)
[How to charge the LiPo batteries](documentation/how_to_cha
[Continuous Integration FAQ](documentation/ci_faq.md)
[How to document things on Gitlab](documentation/how_to_doc
[How to update the website](website/README.md)
[How to use Git](documentation/how_to_use_git.md)

Stable Releases
To browse stable releases from previous teams, view the [Ta

README.md

build success

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 ca
autonomously.

MicroCART has 3 areas of development:

- The **quadcopter** itself
  - The quadcopter has been built from the ground up, incorporating a Zybo board that provides t
    processor and a FPGA, an IMU sensor, motors, props, a LiPo battery, a receiver for manual rem
    control, and a frame that holds it all together.
- The **ground station**
  - The ground station is responsible for issuing important data to the quad (like position data fron
    camera system) and issuing commands to the quad for things like configuration and path follow
    directives.
- The **controls model**
  - The quadcopter processor is programmed to implement a PID controller. We use a Simulink mo
    streamline the PID tuning process and to facilitate effective characterization of the quad.

Sections

Quadcopter
Ground Station
Controls

Documentation

How to demo the quadcopter
How to charge the LiPo batteries
Continuous Integration FAQ
How to document things on Gitlab
How to update the website
How to use Git

Stable Releases

To browse stable releases from previous teams, view the Tags.

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?

Activity