Principles of Software Construction: Objects, Design, and Concurrency

Software engineering in practice

Teams, branch-based development, and workflows

Josh Bloch **Charlie Garrod**

Carnegie Mellon University
School of Computer Science

institute for SOFTWARE RESEARCH

Administrivia

- Homework 4c due tonight
  - Can regain up to 75% of lost Homework 4a credit
    - Directly address TA comments
    - Turn in revised design documents + description of what you changed to Gradescope before Monday night
- No lecture next Tuesday: please vote
- Homework 5 team sign-up deadline Wednesday 5 p.m.
- Midterm exam next Wednesday/Thursday
  - Practice exam released tomorrow
  - Exam review session Monday 6:30-8:30 p.m.
  - Exam released Wednesday night, due Thursday 11:59 p.m.
  - No lecture next Thursday

Key concepts from the past week

Key design principle: Information hiding

- "When in doubt, leave it out."

Minimize Mutability

- Parameter types should be immutable
  - Eliminates need for defensive copying
- Classes should be immutable unless there's a good reason to do otherwise
  - Advantages: simple, thread-safe, reusable
  - Disadvantage: separate object for each value
- If mutable, keep state-space small, well-defined
  - Make clear when it's legal to call which method

Bad: `Date`

Good: `java.time.Instant`

"Fail Fast" – prevent failure, or fail quickly, predictably, and informatively

- Ideally, API should make misuse impossible
  - Fail at compile time or sooner
- Misuse that's statically detectable is second best
  - Fail at build time, with proper tooling
- Misuse leading to prompt runtime failure is third best
  - Fail when first erroneous call is made
  - Method should succeed or have no effect *(failure-atomicity)*
- Misuse that can lie undetected is what nightmares are made of
  - Fail at an undetermined place and time in the future

Don’t let your output become your de facto API

- Document the fact that output formats may evolve in the future
- Provide programmatic access to all data available in string form

```
org.omg.CORBA.MARSHAL: com.ibm.ws.pmi.server.DataDescriptor; IllegalAccessException  minor code: 4942F23E  comp
        at com.ibm.rmi.io.ValueHandlerImpl.readValue(ValueHandlerImpl.java:199)
        at com.ibm.rmi.iiop.CDRInputStream.read_value(CDRInputStream.java:1429)
        at com.ibm.rmi.io.ValueHandlerImpl.read_Array(ValueHandlerImpl.java:625)
        at com.ibm.rmi.io.ValueHandlerImpl.readValueInternal(ValueHandlerImpl.java:273)
        at com.ibm.rmi.io.ValueHandlerImpl.readValue(ValueHandlerImpl.java:189)
        at com.ibm.rmi.iiop.CDRInputStream.read_value(CDRInputStream.java:1429)
        at com.ibm.ejs.sm.beans._EJSRemoteStatelessPmiService_Tie._invoke(_EJSRemoteStatelessPmiService_Tie.ja
        at com.ibm.CORBA.iiop.ExtendedServerDelegate.dispatch(ExtendedServerDelegate.java:515)
        at com.ibm.CORBA.iiop.ORB.process(ORB.java:2377)
        at com.ibm.CORBA.iiop.OrbWorker.run(OrbWorker.java:186)
        at com.ibm.ejs.oa.pool.ThreadPool$PooledWorker.run(ThreadPool.java:104)
        at com.ibm.ws.util.CachedThread.run(ThreadPool.java:137)
```

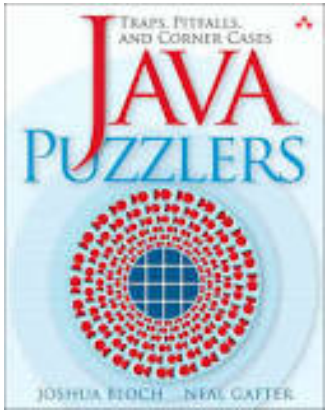
Today: Toward software engineering in practice

- Two puzzlers
- Software engineering for teams
  - Challenges of working as a team
  - Tools and processes for teams
    - Branch-based development, et al.

1. “Time for a Change” (2002)

If you pay $2.00 for a gasket that costs $1.10, how much change do you get?

```
public class Change {
    public static void main(String args[]) {
        System.out.println(2.00 - 1.10);
    }
}
```



What does it print?

(a) `0.9`
(b) `0.90`
(c) It varies
(d) None of the above

```
public class Change {
    public static void main(String args[]) {
        System.out.println(2.00 - 1.10);
    }
}
```

What does it print?

(a) `0.9`

(b) `0.90`

(c) It varies

(d) None of the above: `0.8999999999999999`

Decimal values can't be represented exactly by `float` or `double`

Another look

```
public class Change {
    public static void main(String args[]) {
        System.out.println(2.00 - 1.10);
    }
}
```

How do you fix it?

```
// You could fix it this way...
import java.math.BigDecimal;
public class Change {
    public static void main(String args[]) {
        System.out.println(
            new BigDecimal("2.00").subtract(
                new BigDecimal("1.10")));
    }
}
```

Prints 0.90

```
// ...or you could fix it this way
public class Change {
    public static void main(String args[]) {
        System.out.println(200 - 110);
    }
}
```

Prints 90

The moral

- Avoid `float` and `double` where exact answers are required
  - For example, when dealing with money
- Use `BigDecimal`, `int`, or `long` instead

2. “A Change is Gonna Come”

If you pay $2.00 for a gasket that costs $1.10, how much change do you get?

```
import java.math.BigDecimal;

public class Change {
    public static void main(String args[]) {
        BigDecimal payment = new BigDecimal(2.00);
        BigDecimal cost = new BigDecimal(1.10);
        System.out.println(payment.subtract(cost));
    }
}
```

What does it print?

(a) `0.9`
(b) `0.90`
(c) `0.8999999999999999`
(d) None of the above

```
import java.math.BigDecimal;

public class Change {
    public static void main(String args[]) {
        BigDecimal payment = new BigDecimal(2.00);
        BigDecimal cost = new BigDecimal(1.10);
        System.out.println(payment.subtract(cost));
    }
}
```

What does it print?

(a) `0.9`

(b) `0.90`

(c) `0.8999999999999999`

(d) None of the above:

`0.899999999999999911821580299874767661094665527343 75`

Another look

We used the wrong `BigDecimal` constructor.

The spec says:

```
public BigDecimal(double val)
```

Translates a double into a BigDecimal which is the exact decimal representation of the double's binary floating-point value.

```
import java.math.BigDecimal;

public class Change {
    public static void main(String args[]) {
        BigDecimal payment = new BigDecimal(2.00);
        BigDecimal cost = new BigDecimal(1.10);
        System.out.println(payment.subtract(cost));
    }
}
```

How do you fix it?

```
import java.math.BigDecimal;

public class Change {
    public static void main(String args[]) {
        BigDecimal payment = new BigDecimal("2.00");
        BigDecimal cost = new BigDecimal("1.10");
        System.out.println(payment.subtract(cost));
    }
}
```

Prints `0.90`

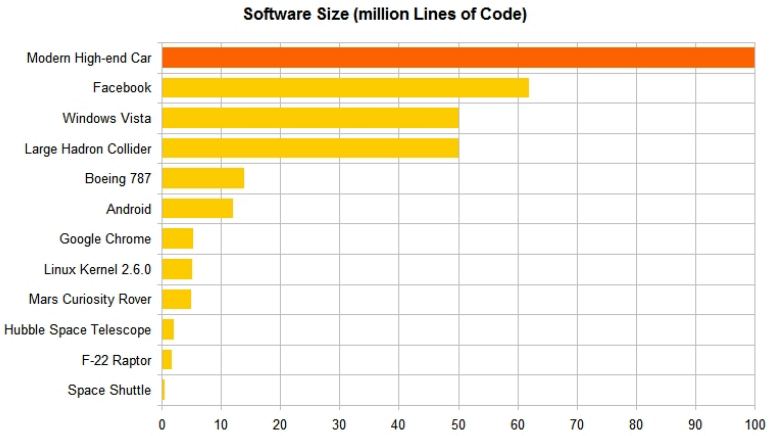
The moral

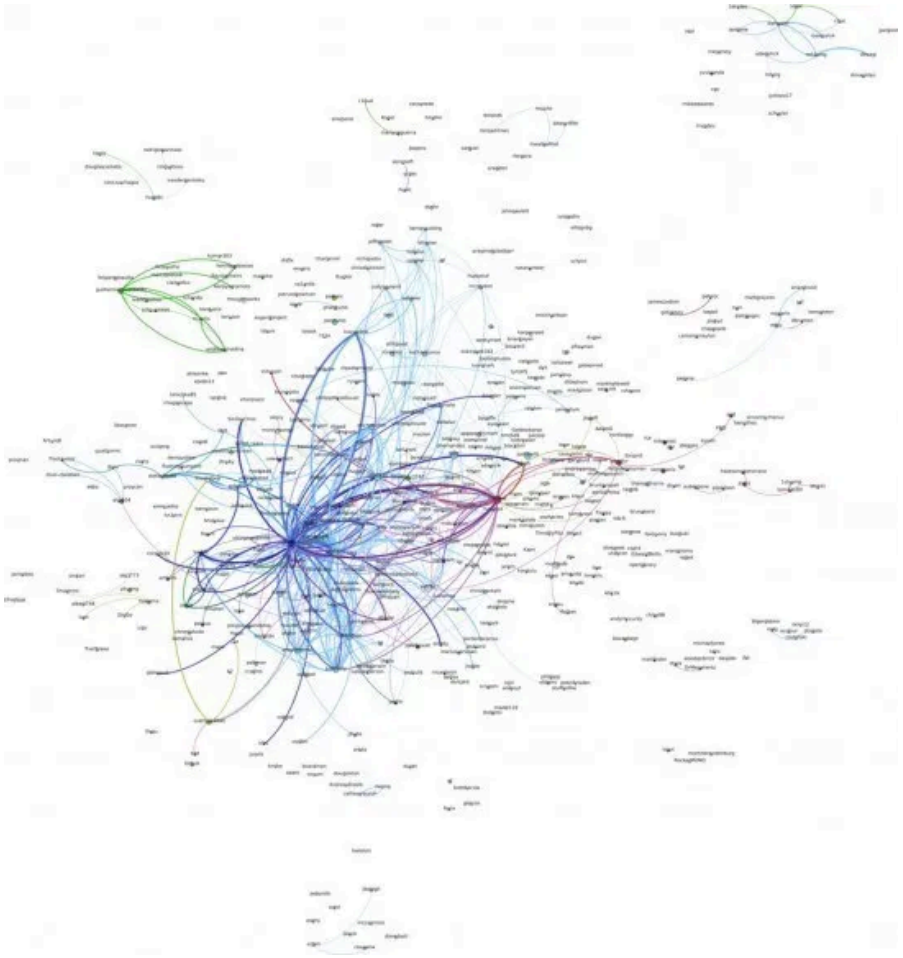
- Use `new BigDecimal(String)`, not `new BigDecimal(double)`
- `BigDecimal.valueOf(double)` is better, but not perfect
  - Use it for non-constant values.
- For API designers
  - Make it easy to do the commonly correct thing
  - Make it hard to misuse
  - Make it possible to do exotic things

Today: Toward software engineering in practice

- Two puzzlers
- Software engineering for teams
  - Challenges of working as a team
  - Tools and processes for teams
    - Branch-based development, et al.

Software engineering is inherently collaborative





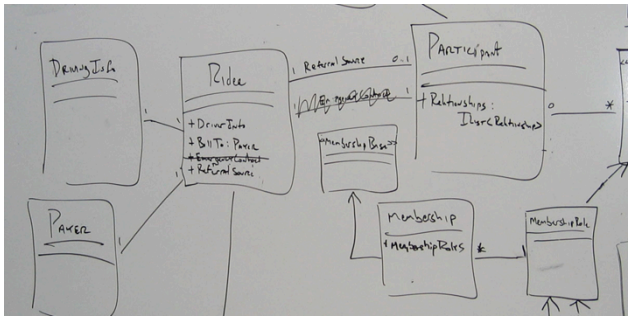
institute for SOFTWARE RESEARCH

Challenges of working as a team:

Challenges of working as a team: Aligning expectations

- How does the team make decisions?
- How do you divide the work?
- Does the team share the same goals and incentives?
- What happens when work isn't completed as expected?
- When do team members like to work?
- What other commitments do your team members have?
- Where will you get the work done?
- …

Decide what to build, then design the API




```
// A collection of elements (root of the collection hierarchy)
public interface Collection<E> {

    // Ensures that collection contains o
    boolean add(E o);

    // Removes an instance of o from collection, if present
    boolean remove(Object o);

    // Returns true iff collection contains o
    boolean contains(Object o) ;

    // Returns number of elements in collection
    int size() ;

    // Returns true if collection is empty
    boolean isEmpty();

    ...  // Remainder omitted
}
```

Basic Process:

(1) **Determine minimal feature set**

(2) **Draw UML on a whiteboard.**

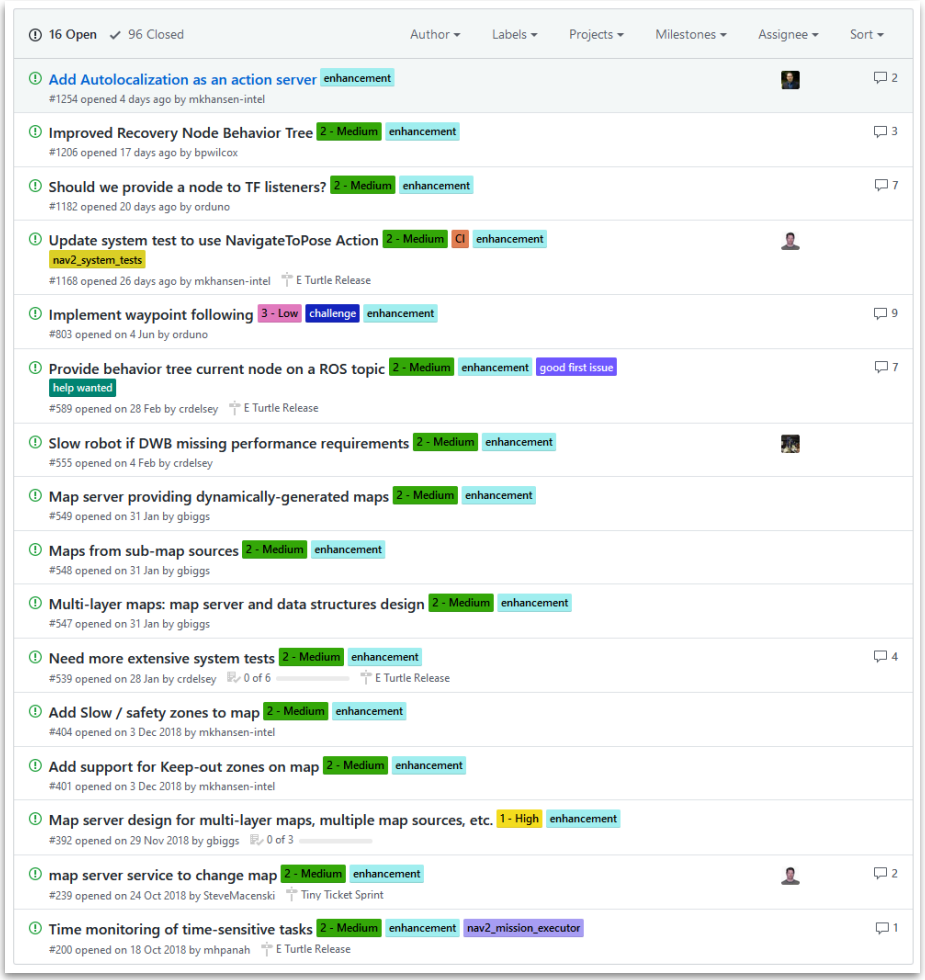
(3) **Sketch out your API on paper**

(4) **Write example code**

(5) **Review**

(6) **Repeat**

Break up tasks into GitHub Issues

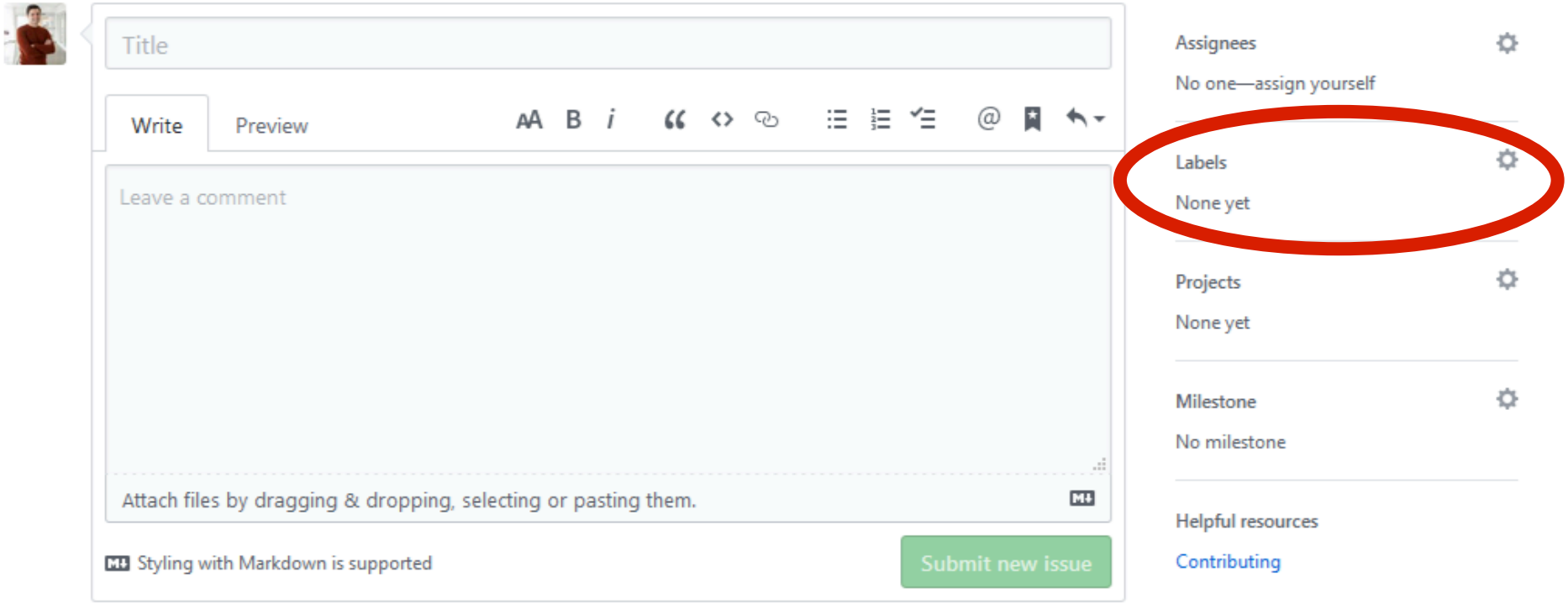


Issues can represent both tasks and bugs that need to be fixed.

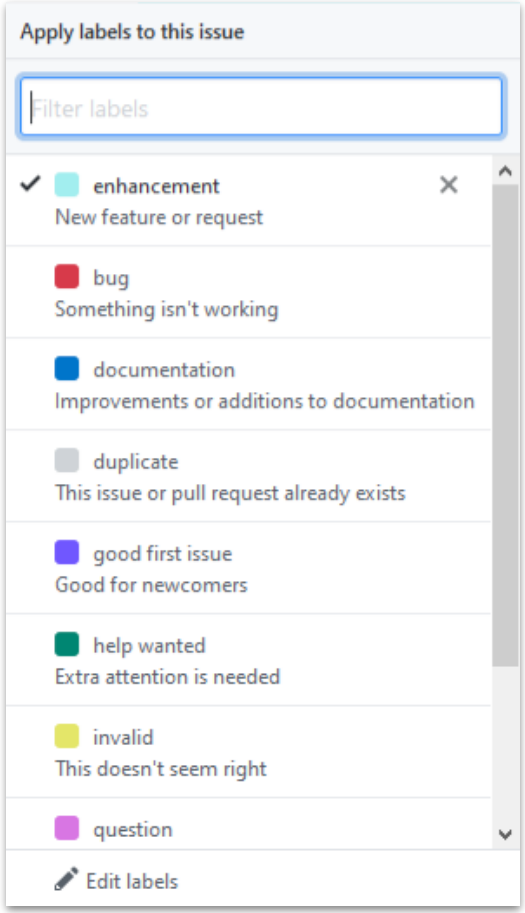
Issues should be:

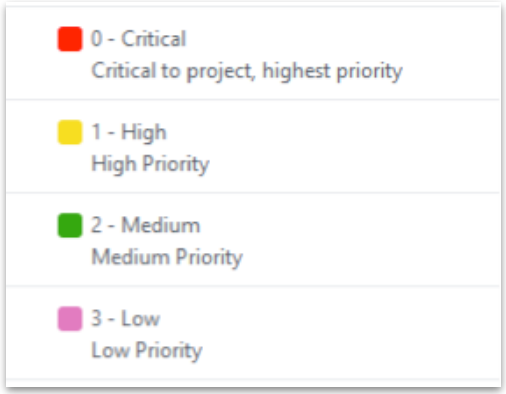
- a *reasonable* chunk of work
- focused and cohesive

Break up tasks into GitHub Issues

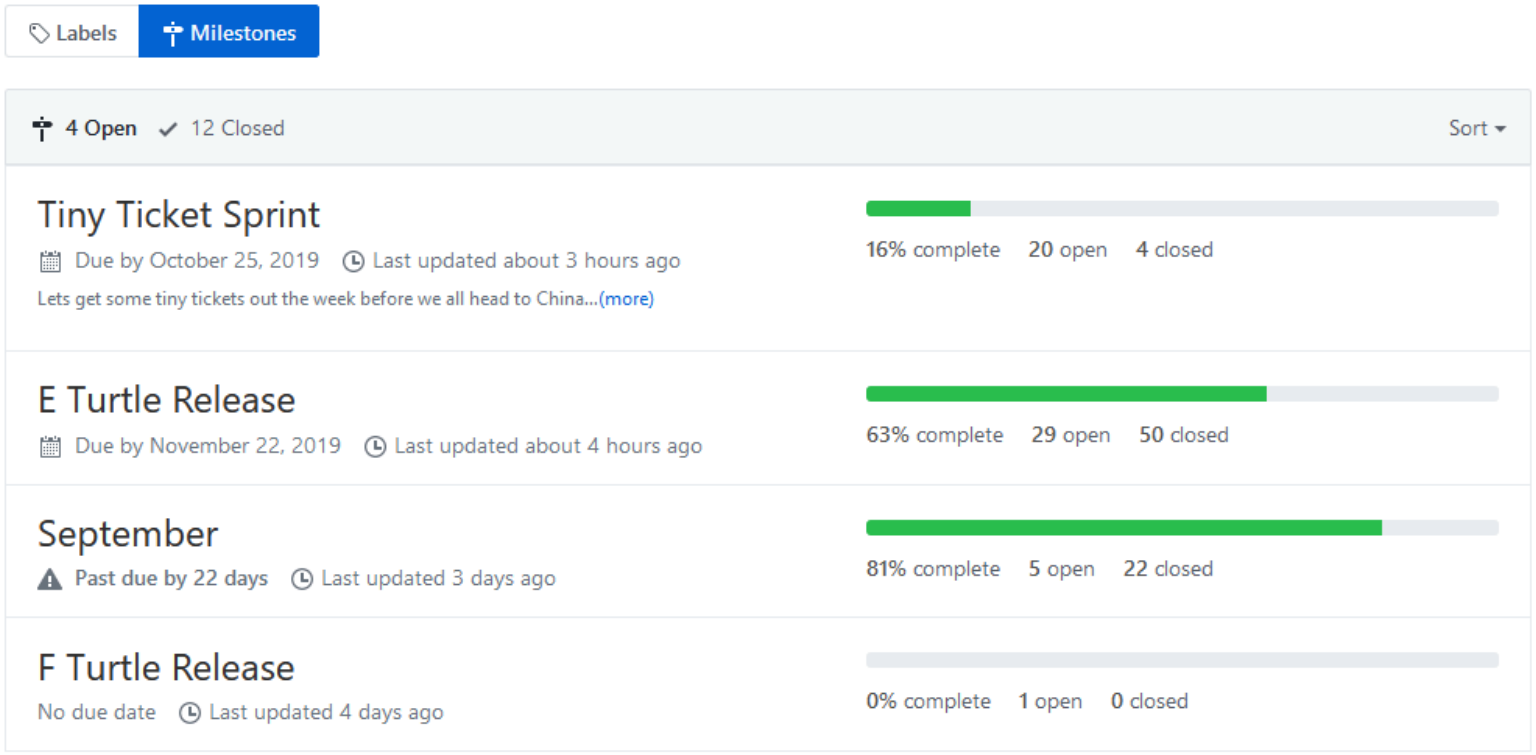


Use labels to indicate priority and differentiate bugs from features





Consider using milestones (e.g., HW5a, HW5b)

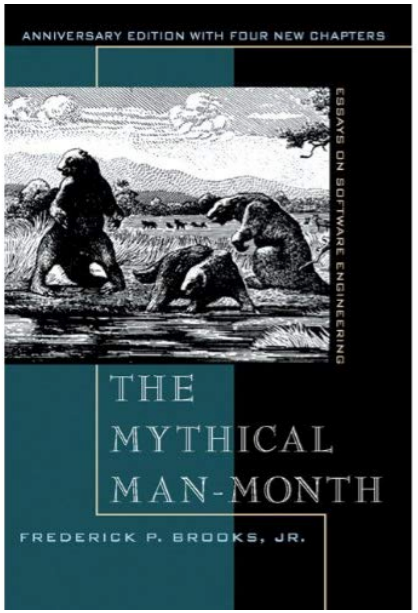


How does a large software project get to be one year late?

How does a large software project get to be one year late?

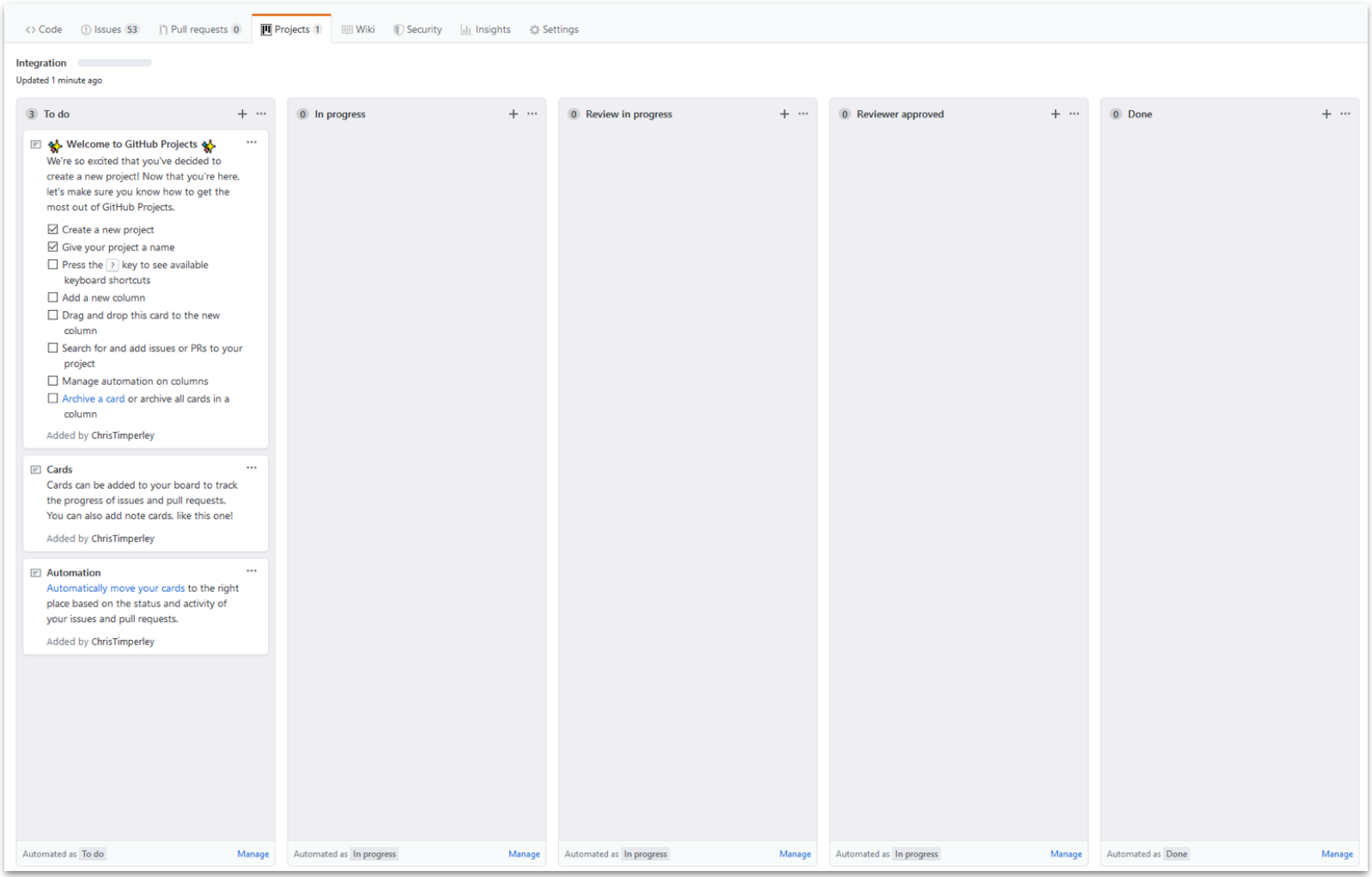
One day at a time.

— Fred Brooks, *The Mythical Man-Month*

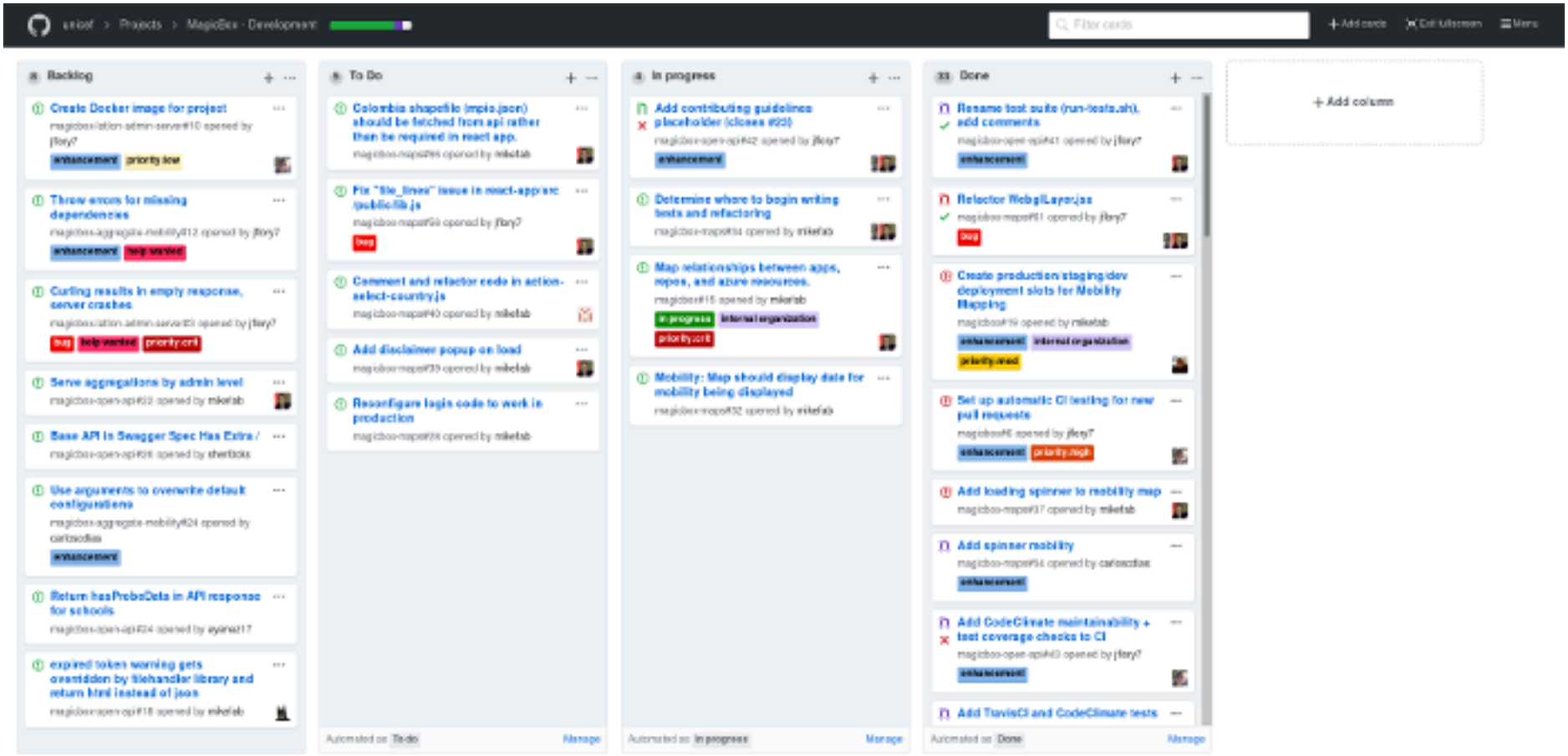


https://en.wikipedia.org/wiki/The_Mythical_Man-Month

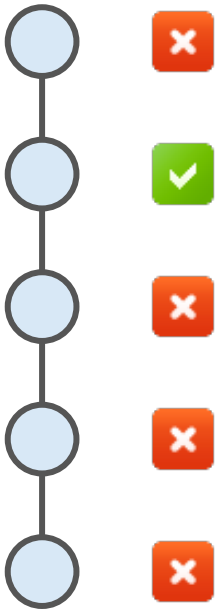
Use a simple Kanban board to measure progress



Use a simple Kanban board to measure progress



Single-branch development doesn't scale to teams



Master

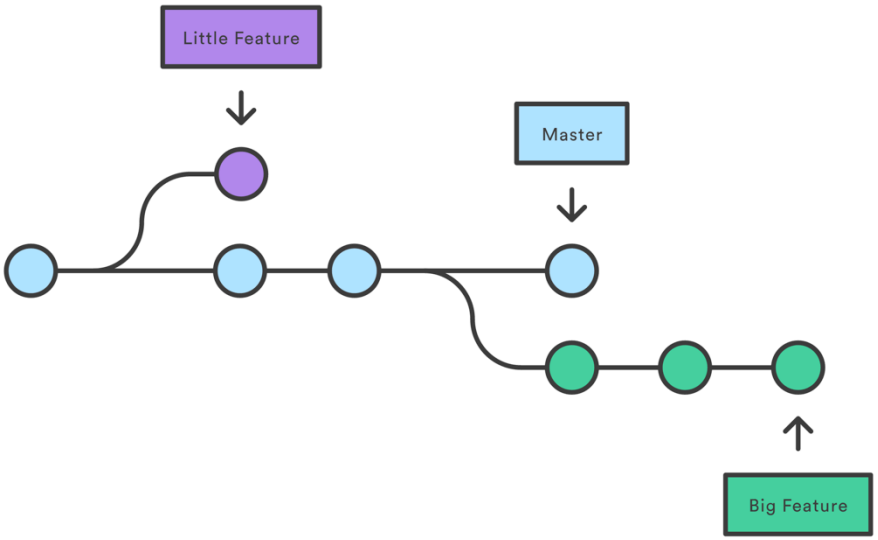
merge_conflict — bash — 80×24

bash | python | bash | bash

```
Your branch is up-to-date with 'origin/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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
dhcp-10-101-250-155:merge_conflict SUYEONSON$ git add index.html
dhcp-10-101-250-155:merge_conflict SUYEONSON$ git commit -m 'tryna make a merge
conflict'
[master ee28024] tryna make a merge conflict
 1 file changed, 1 deletion(-)
dhcp-10-101-250-155:merge_conflict SUYEONSON$ git push
To git@github.com:suyeonson/merge_conflict.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@github.com:suyeonson/merge_conflict.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
dhcp-10-101-250-155:merge_conflict SUYEONSON$
```

build failure

Use simple **branch-based development**



Commits on Oct 20, 2019

Added file checking methods to FileSystem (#28) ChrisTimperley committed yesterday — Verified — da32e4a

Commits on Oct 19, 2019

Implemented basic filesystem API (#27) ChrisTimperley committed 2 days ago — Verified — 73d331e

Added workaround for shell calls without both stdout and stderr (#26) ChrisTimperley committed 2 days ago — Verified — 06aa050

Added Container class for holding Docker container details (#24) ChrisTimperley committed 3 days ago — Verified — 05c61e8

Commits on Oct 13, 2019

Added DockerDaemon for maintaining connnections to daemon (fixes #21) (... ChrisTimperley committed 8 days ago — Verified — 79ad8e7

Added environ method to Shell (#20) ChrisTimperley committed 9 days ago — Verified — 4494af4

Added basic popen to shell (fixes #6) (#19) ChrisTimperley committed 9 days ago — Verified — cf79374

Add encoding and text parameters to Shell commands (fixes #9) (#17) ChrisTimperley committed 9 days ago — Verified — cef114c

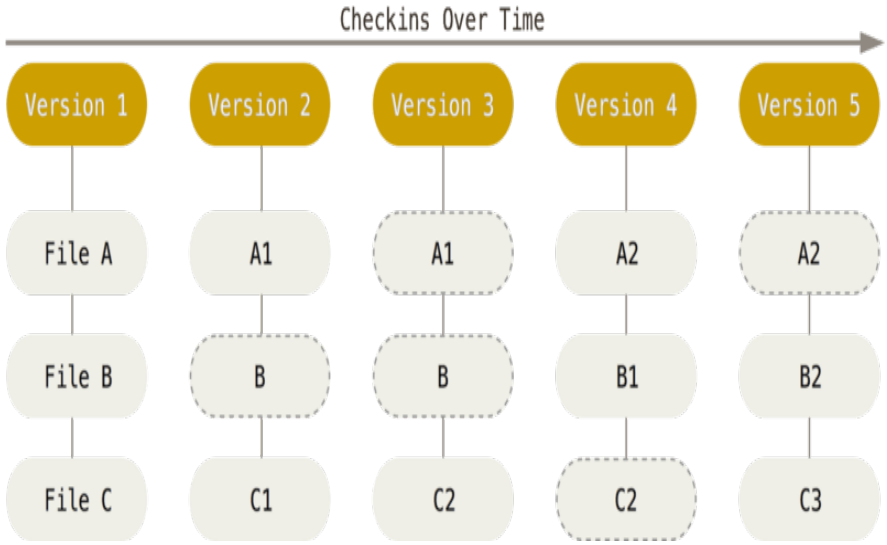
Create a new branch for each feature.

- allows parallel development
- no dealing with half-finished code
- no merge conflicts!

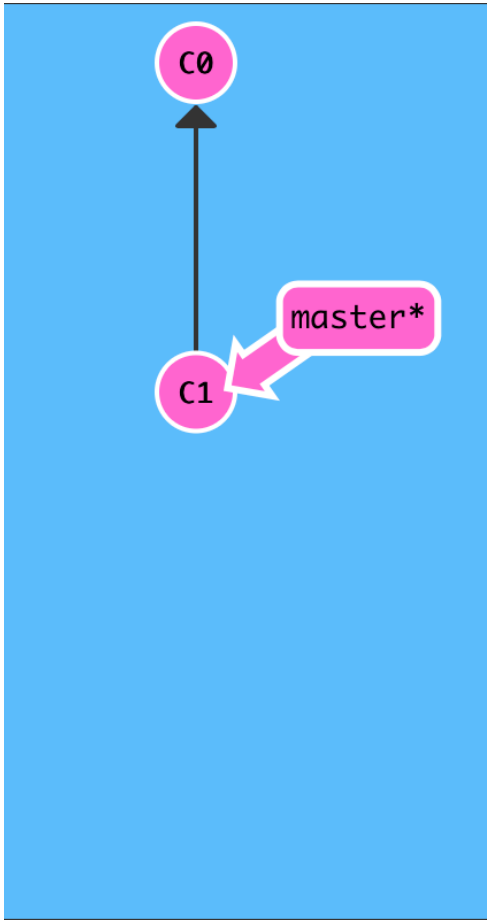
Every commit to “master” should pass your CI checks.

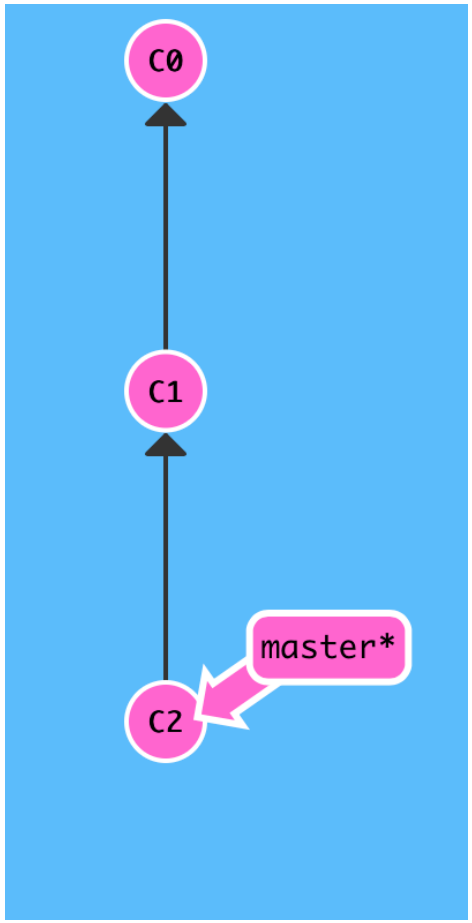
Git, practically

- Git stores each version as a snapshot
- If files have not changed, only a link to the previous file is stored
- Each version is referred by the SHA-1 hash of the contents



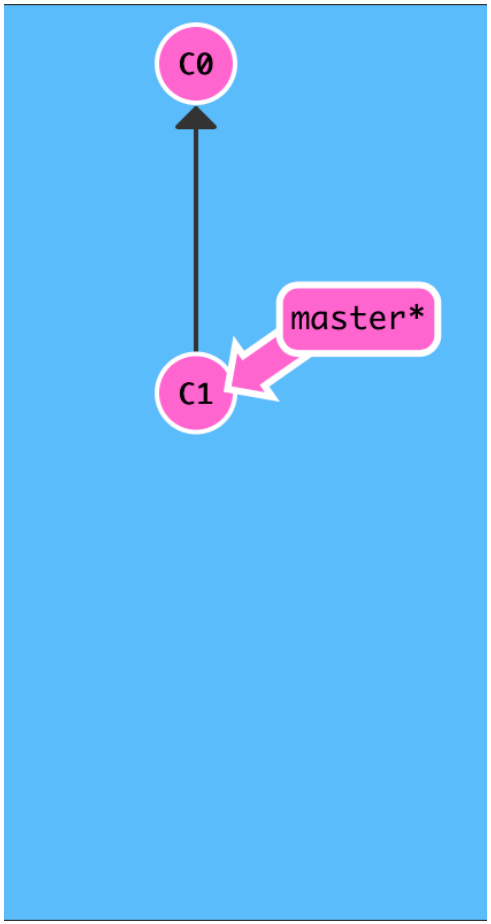
git commit

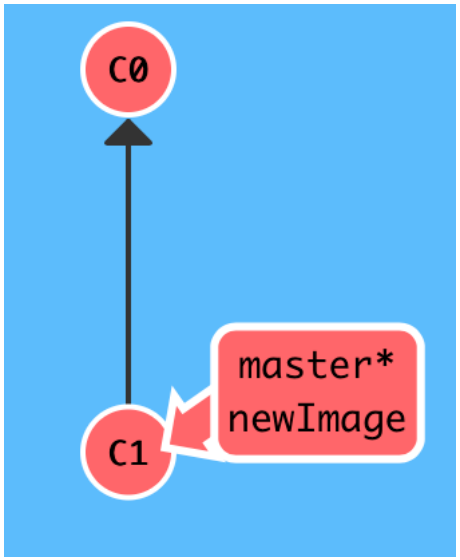




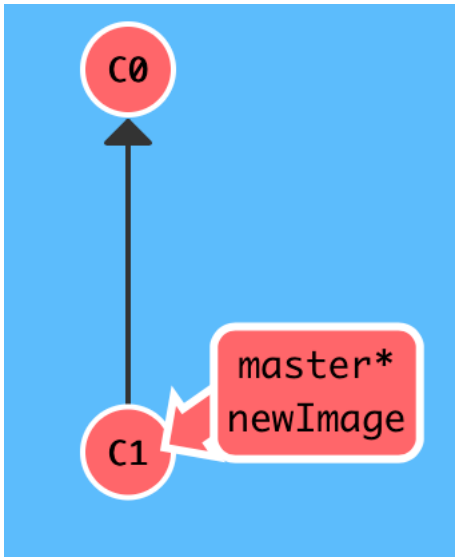
Graphics by https://learngitbranching.js.org

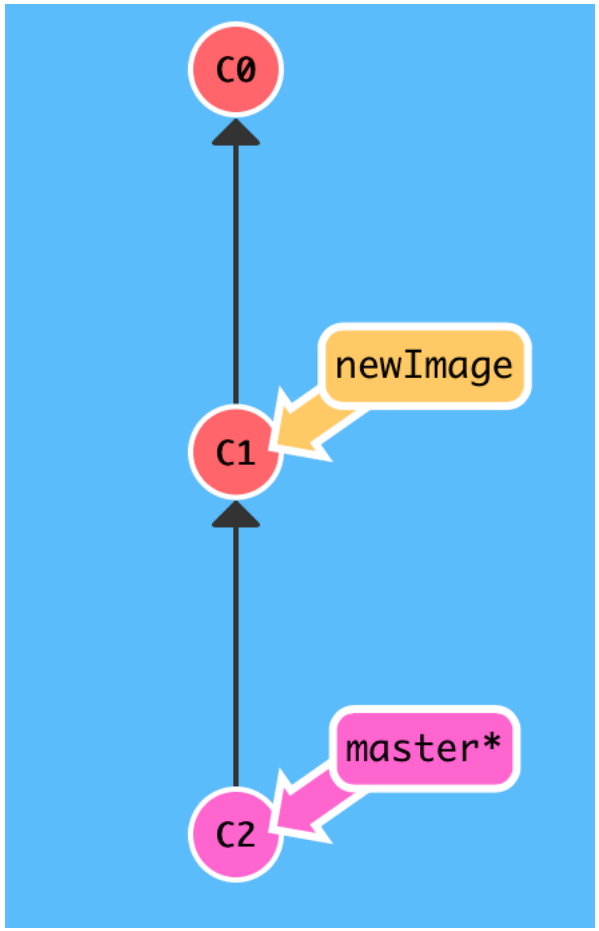
git branch newImage



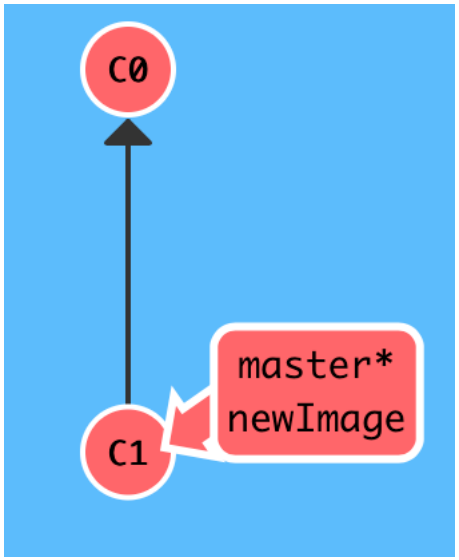


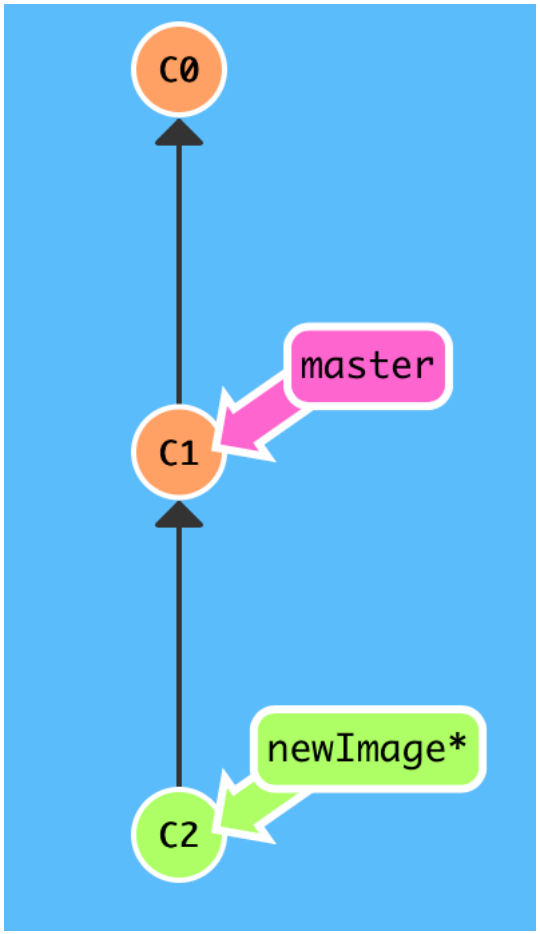
git commit



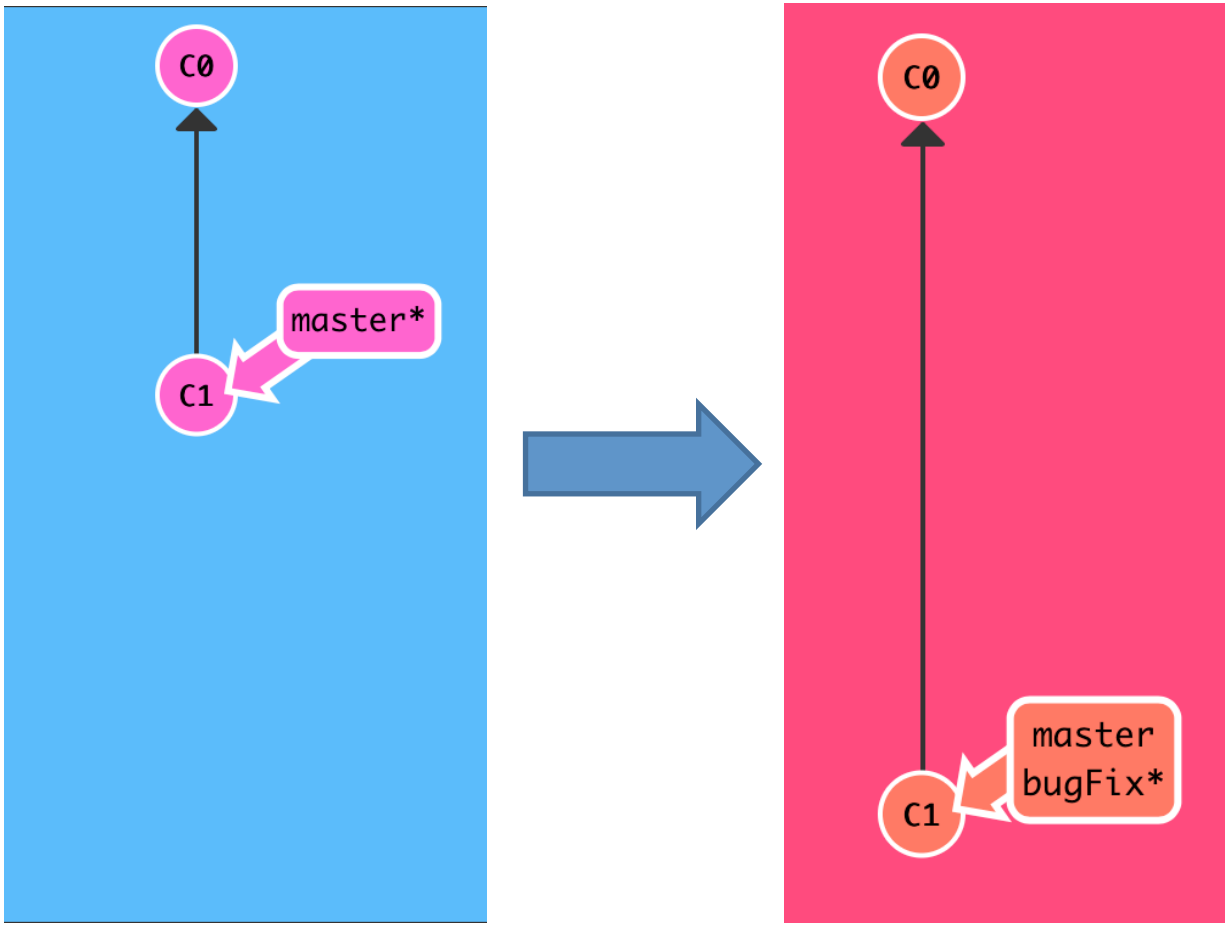


git checkout newImage; git commit



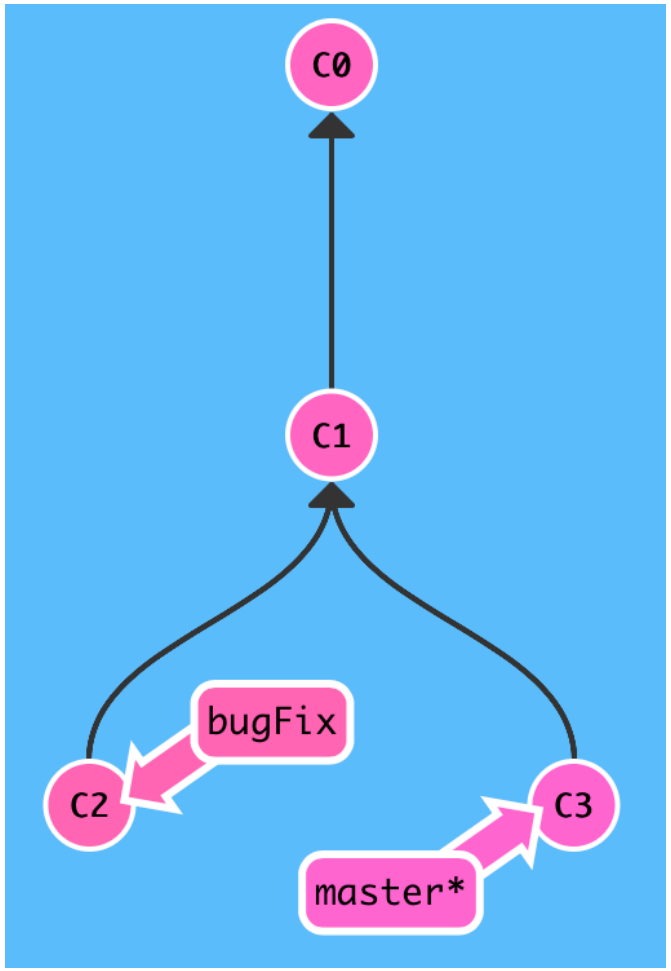


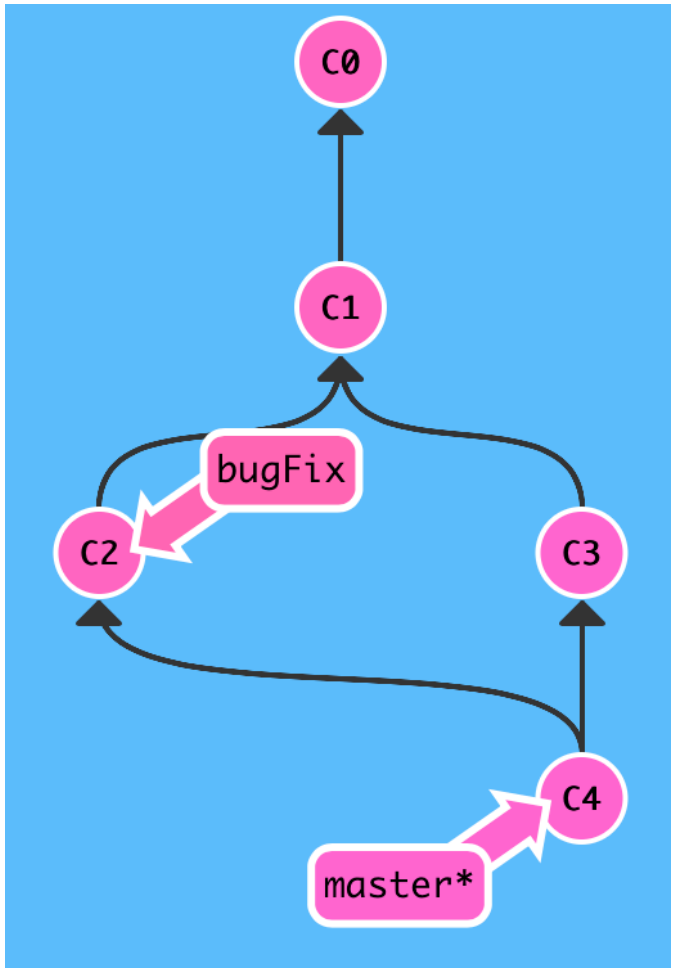
Activity: Make a new branch named bugFix and switch to that branch



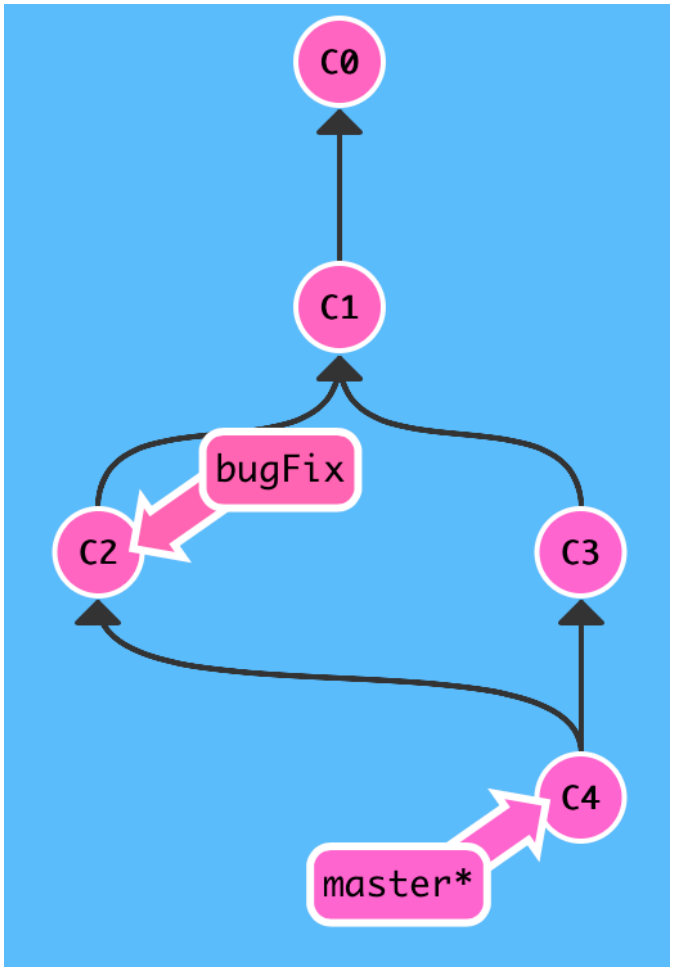
Three ways to move work around between branches

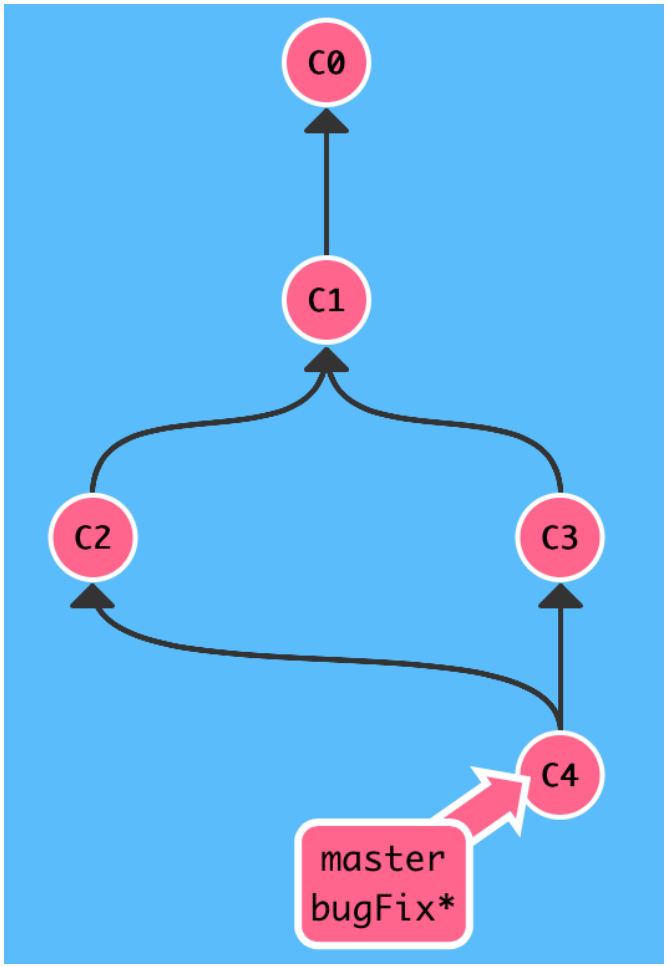
1) `git merge bugFix` (into `master`)





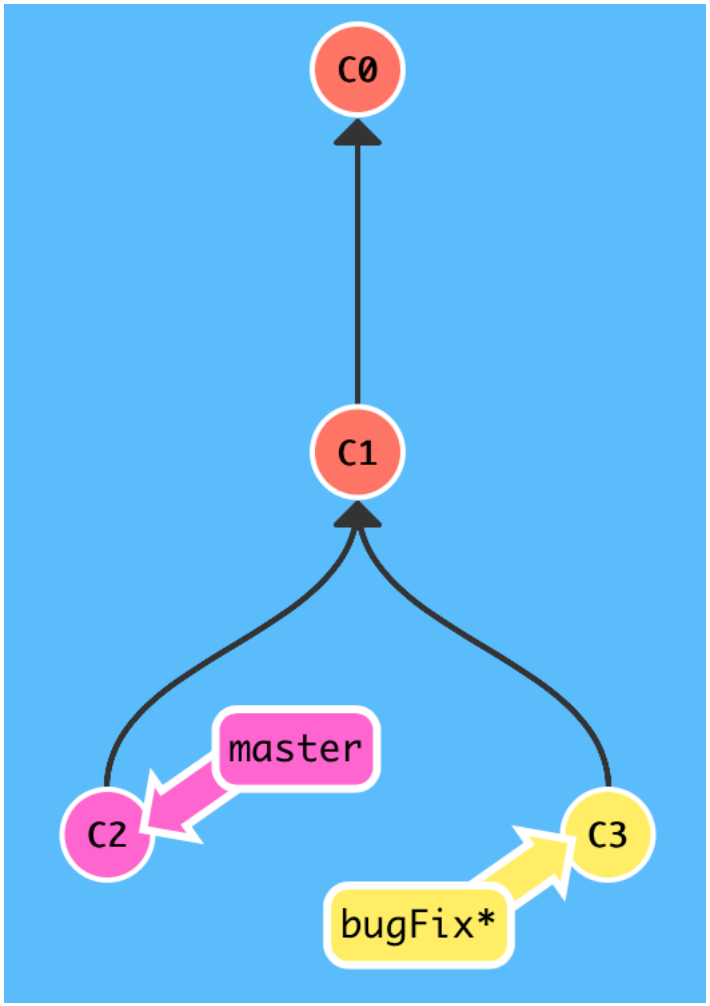
`git checkout bugfix; git merge master` (into bugFix)

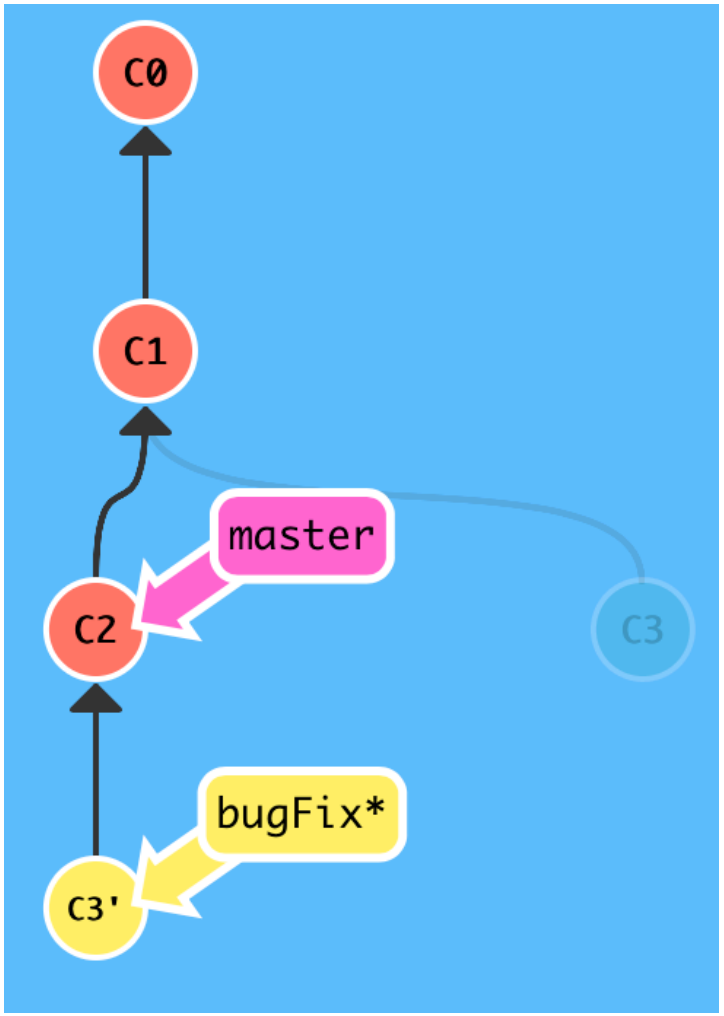




Move work from bugFix directly onto master

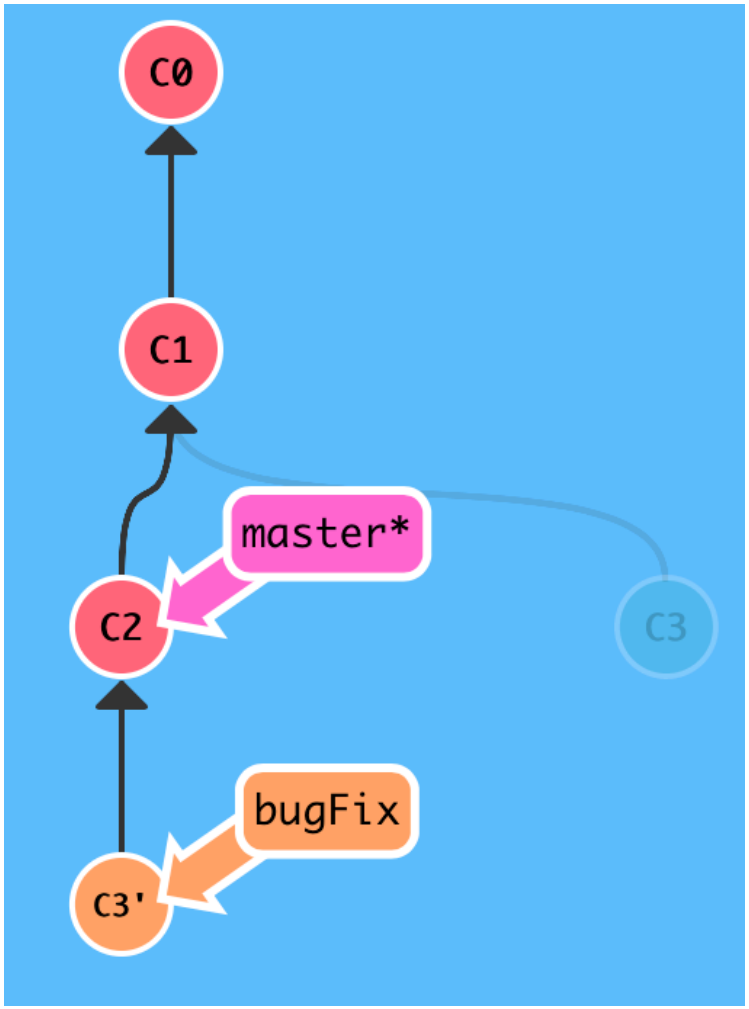
```
2) git rebase master
```

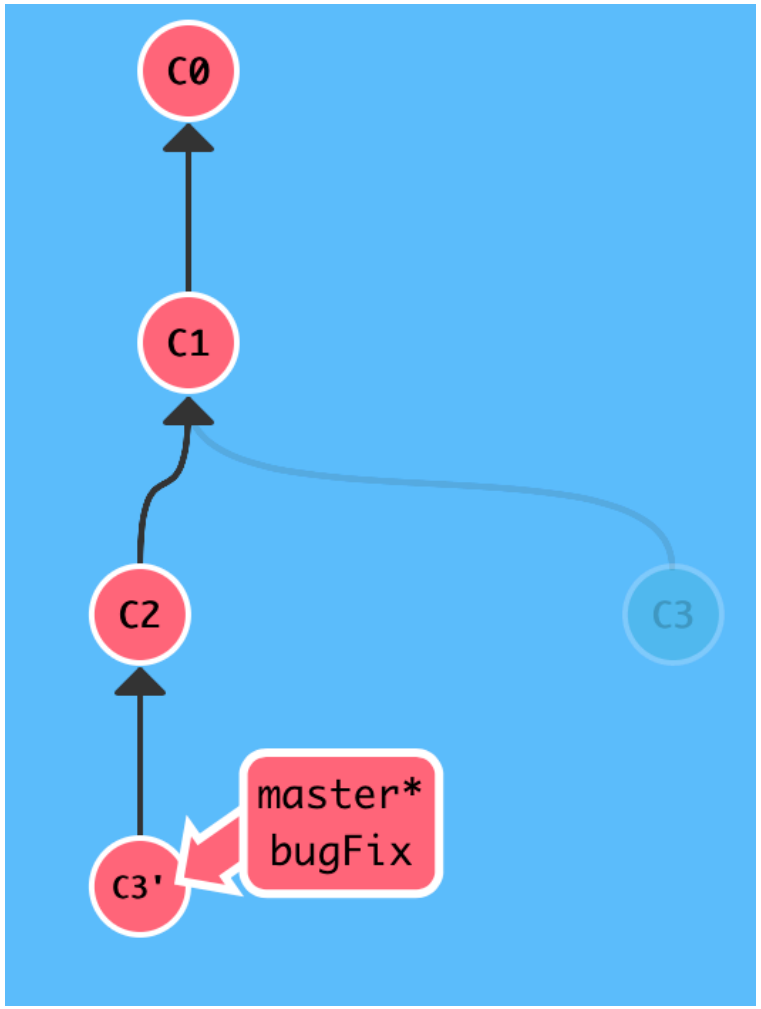




But master hasn't been updated, so:

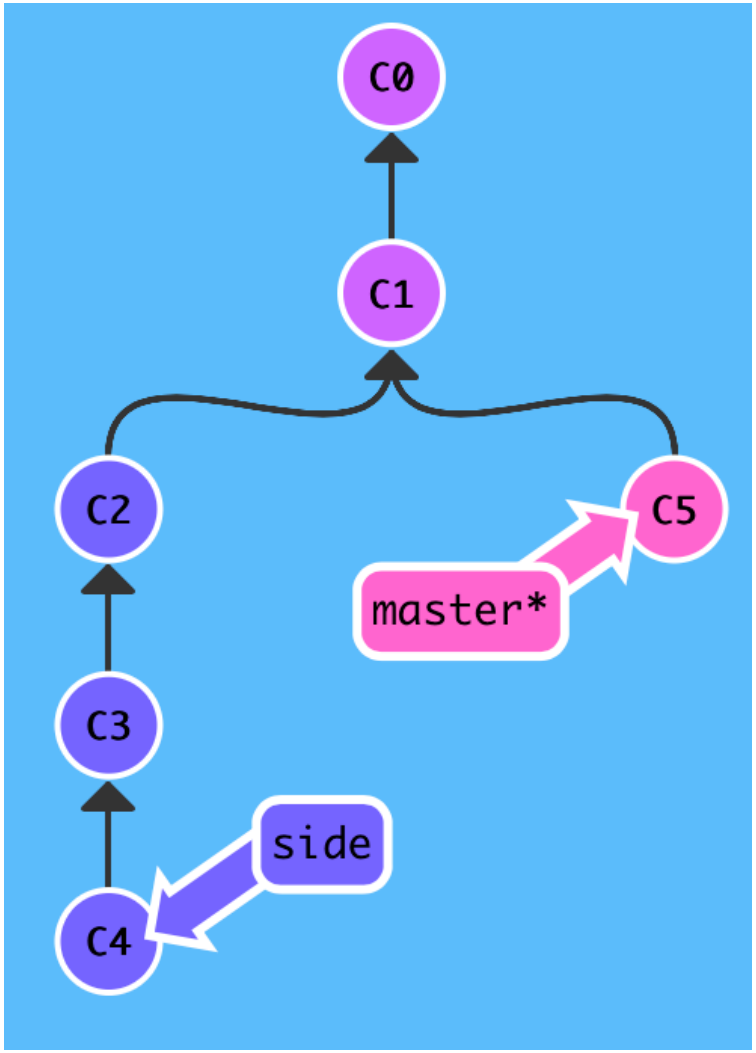
```
git checkout master; git rebase bugFix
```

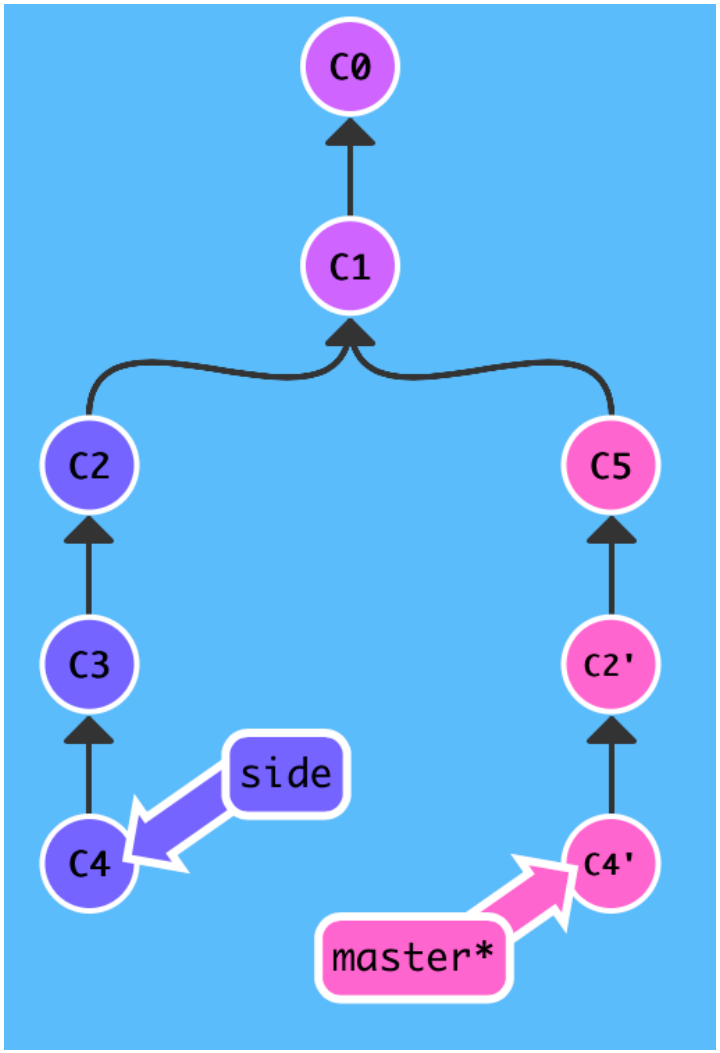




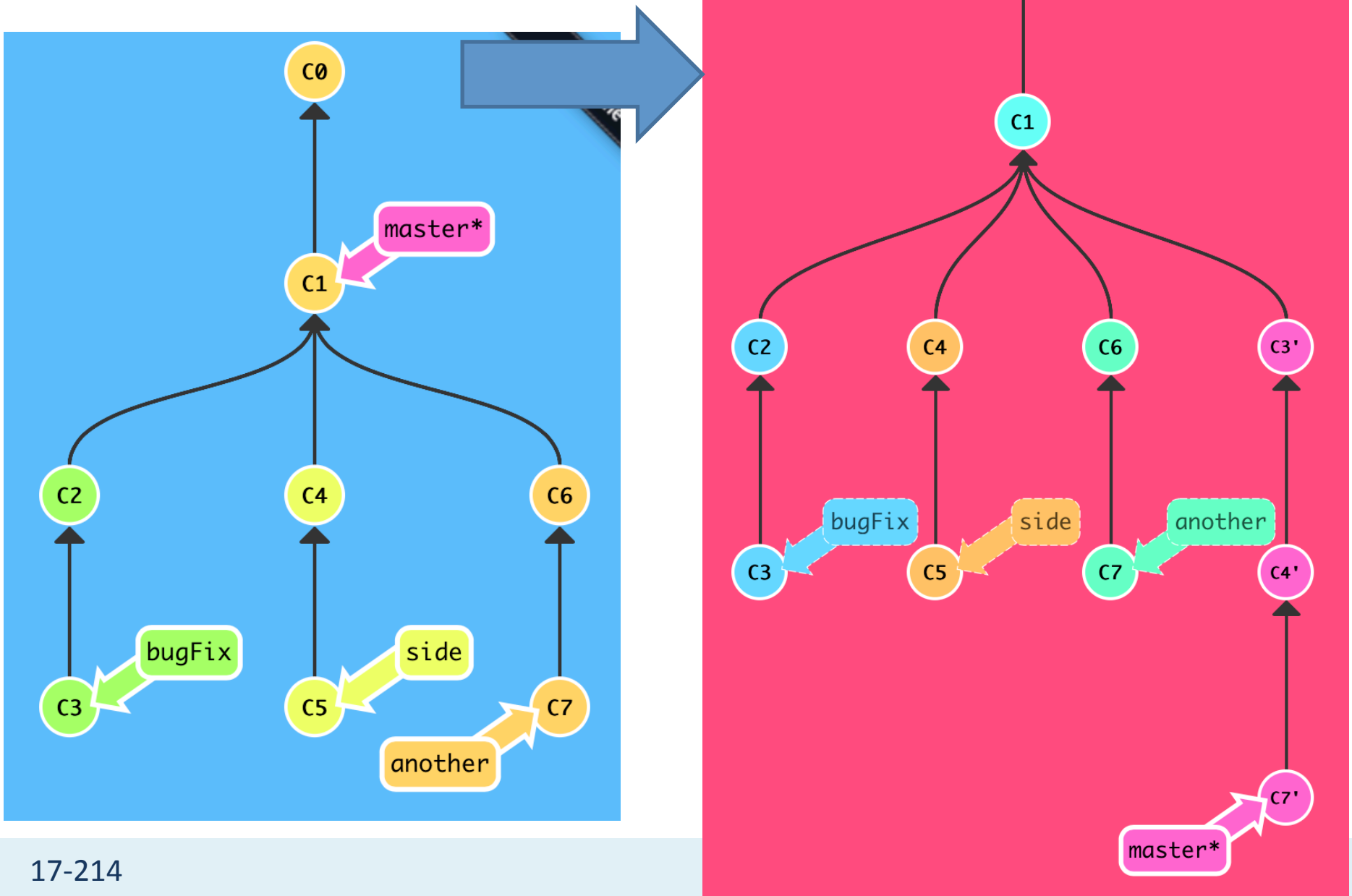
Copy a series of commits below current location

```
3) git cherry-pick C2 C4
```

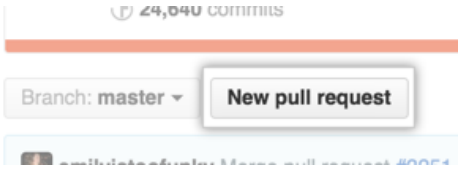


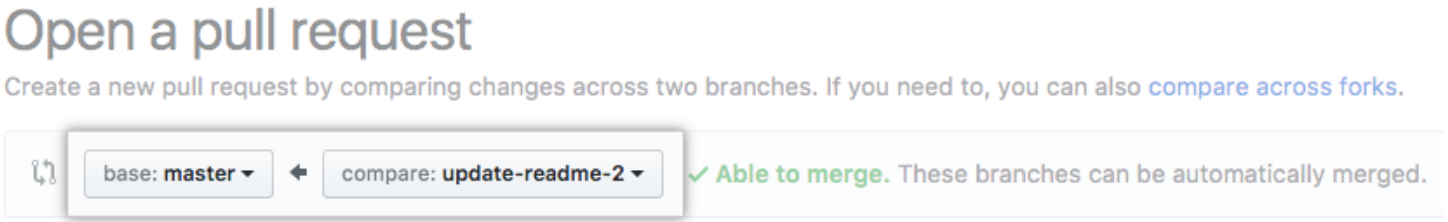


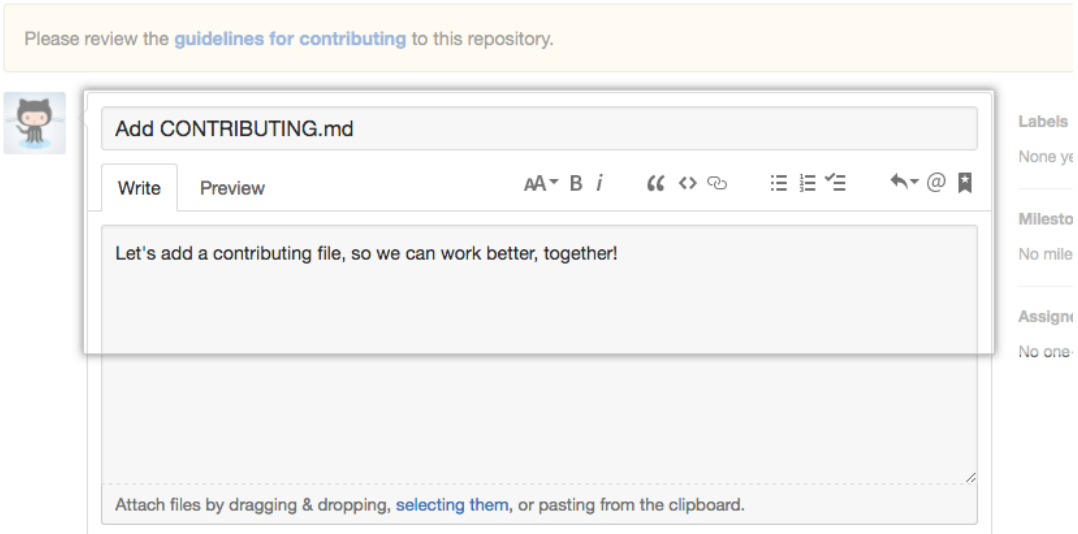
Activity:


C0
master*
C1
C2
C4
C6
bugFix
side
C3
C5
C7
another
C0
C1
C2
C4
C6
C3'
bugFix
side
another
C3
C5
C7
C4'
C7'
master*

Use GitHub **pull requests** to review and merge changes

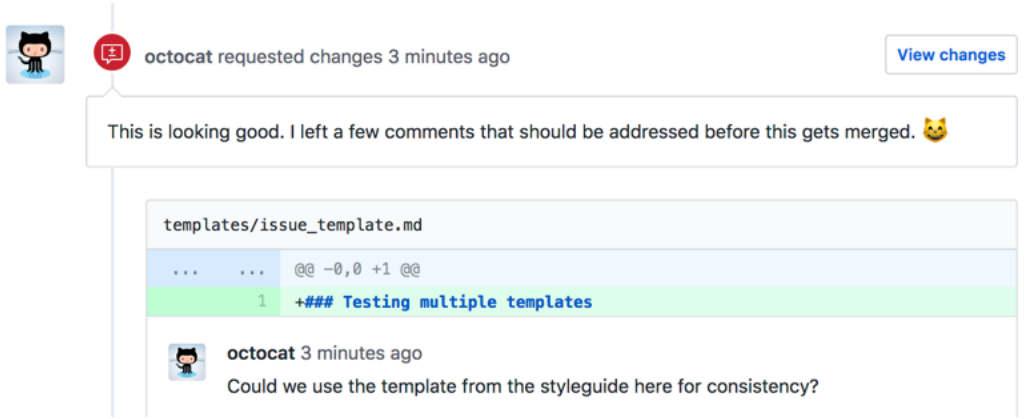


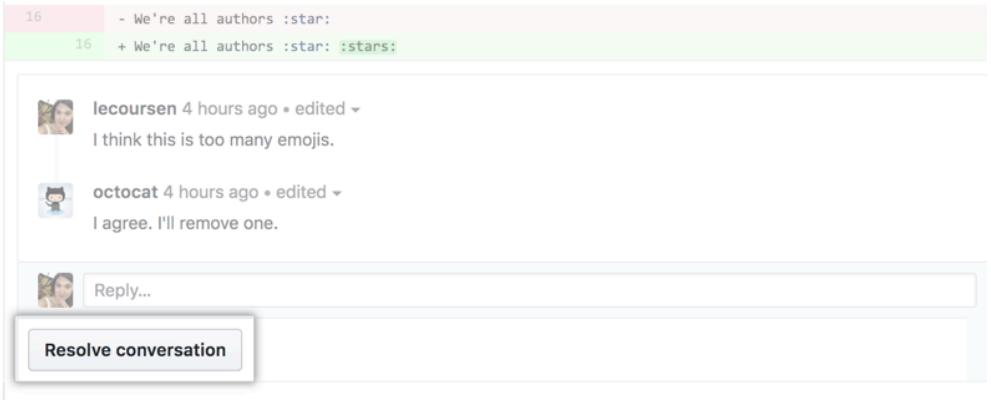




https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/creating-a-pull-request

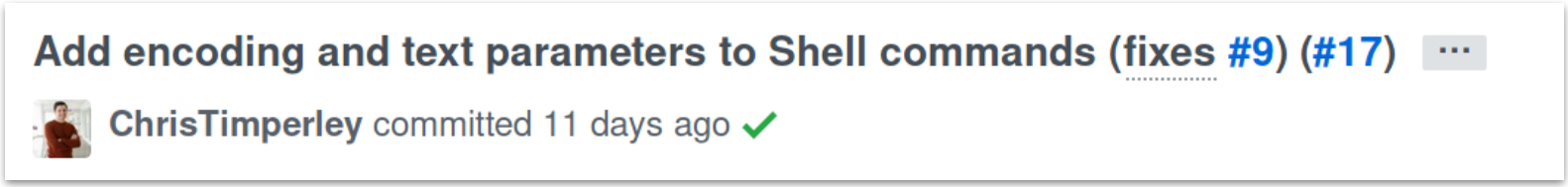
Ask your teammates to review your pull request





https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-request-reviews

Bonus tip: Automatically close issues in commits/PRs



Use any of the following words:

- close #N, closes #N, closed #N
- fix #N, fixes #N, fixed #N
- resolve #N, resolves #N, resolved #N

Summary

- Identify and discuss risks within your team
  - Get to know your teammates, and agree on your process
- Use standard tools to improve your process
- Please vote!