Coding as a Team, Part 2

17-356/17-766
Software Engineering for Startups

https://cmu-17-356.github.io

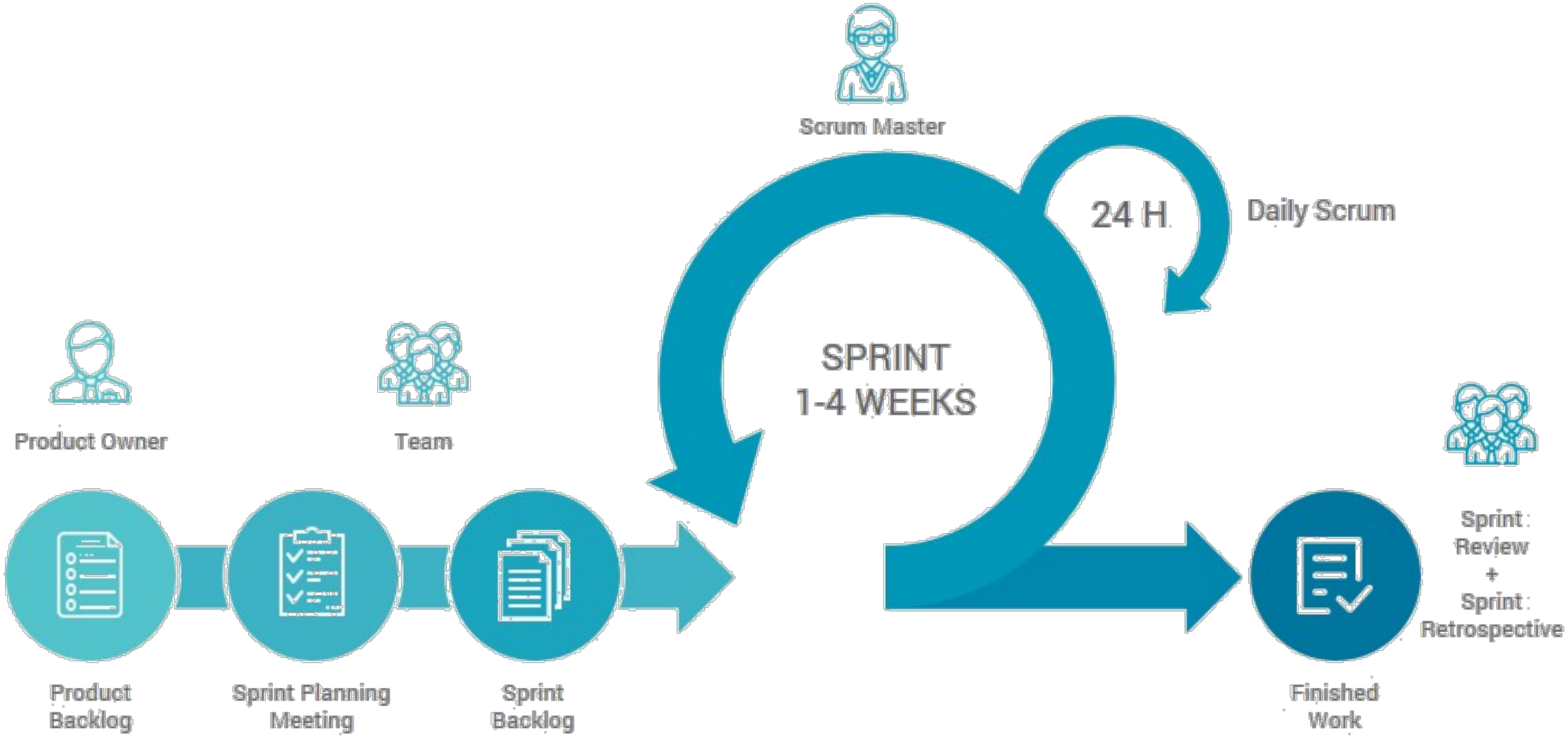
Andrew Begel and Fraser Brown

P1 released! You will have ~three weeks

Please start early

Why bother with process?

Agile Process Overview



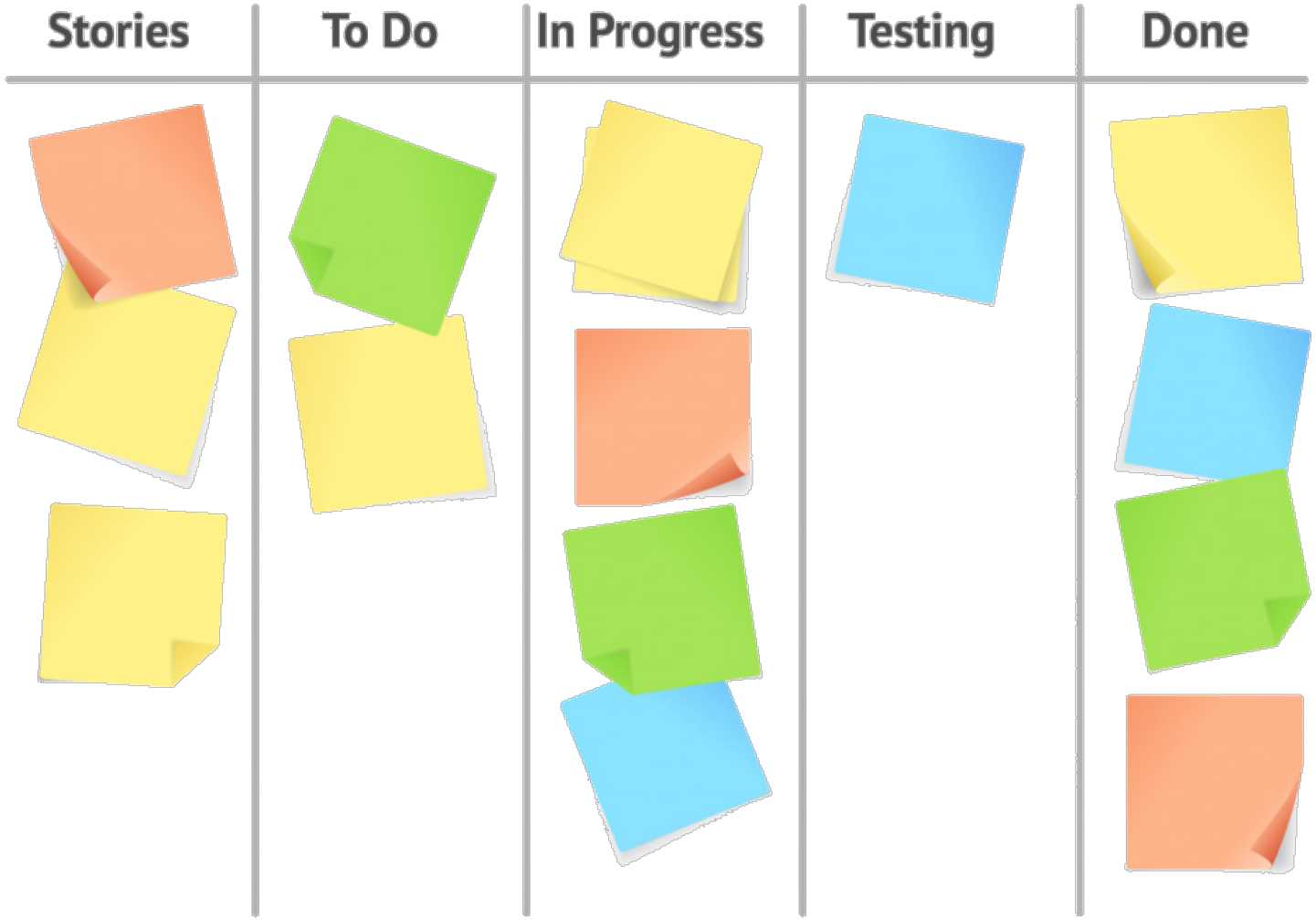
Sprint Planning: Product and Sprint Backlogs

All tasks are maintained in an ordered list called a backlog, sorted by priority.

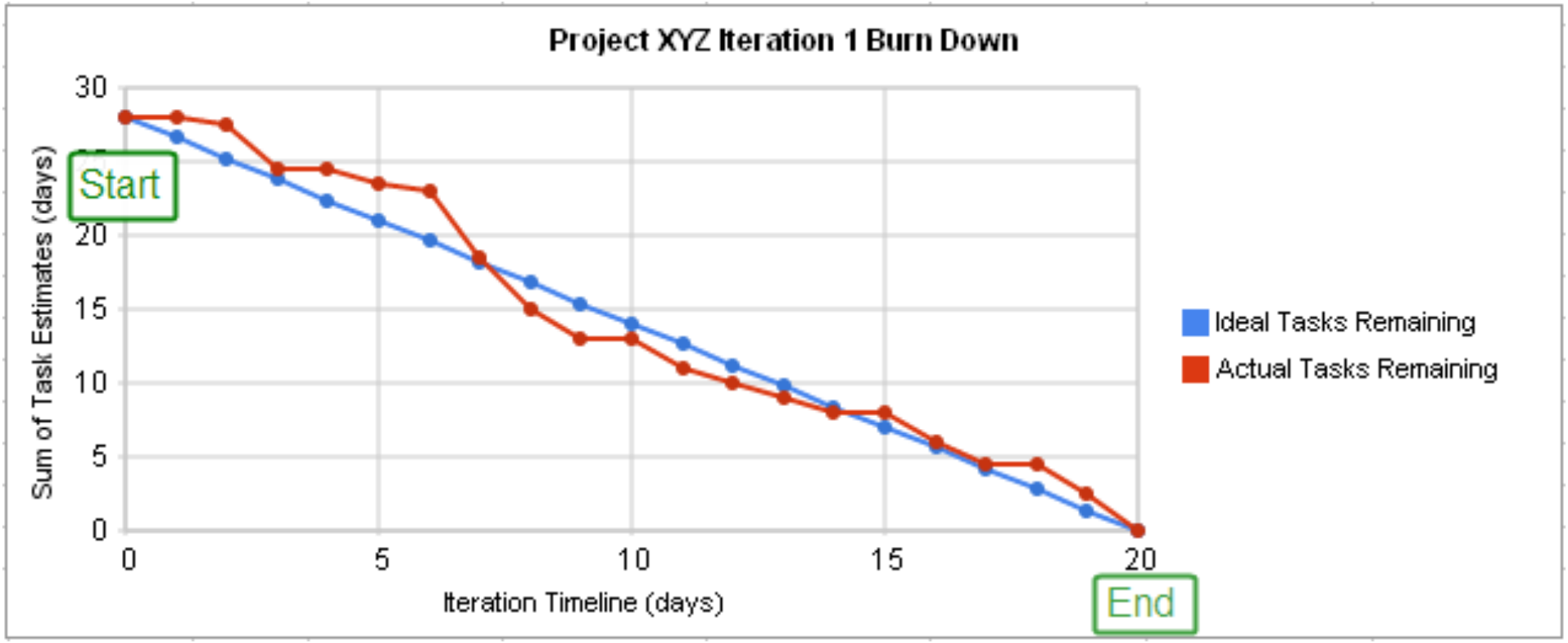
After every sprint, the team visits the backlog and reprioritizes tasks into a *sprint* backlog for the next release.

For your teams: You must use GitHub Issues (or Asana, Trello, Backcamp, or Jira) to keep track of your team's development tasks.

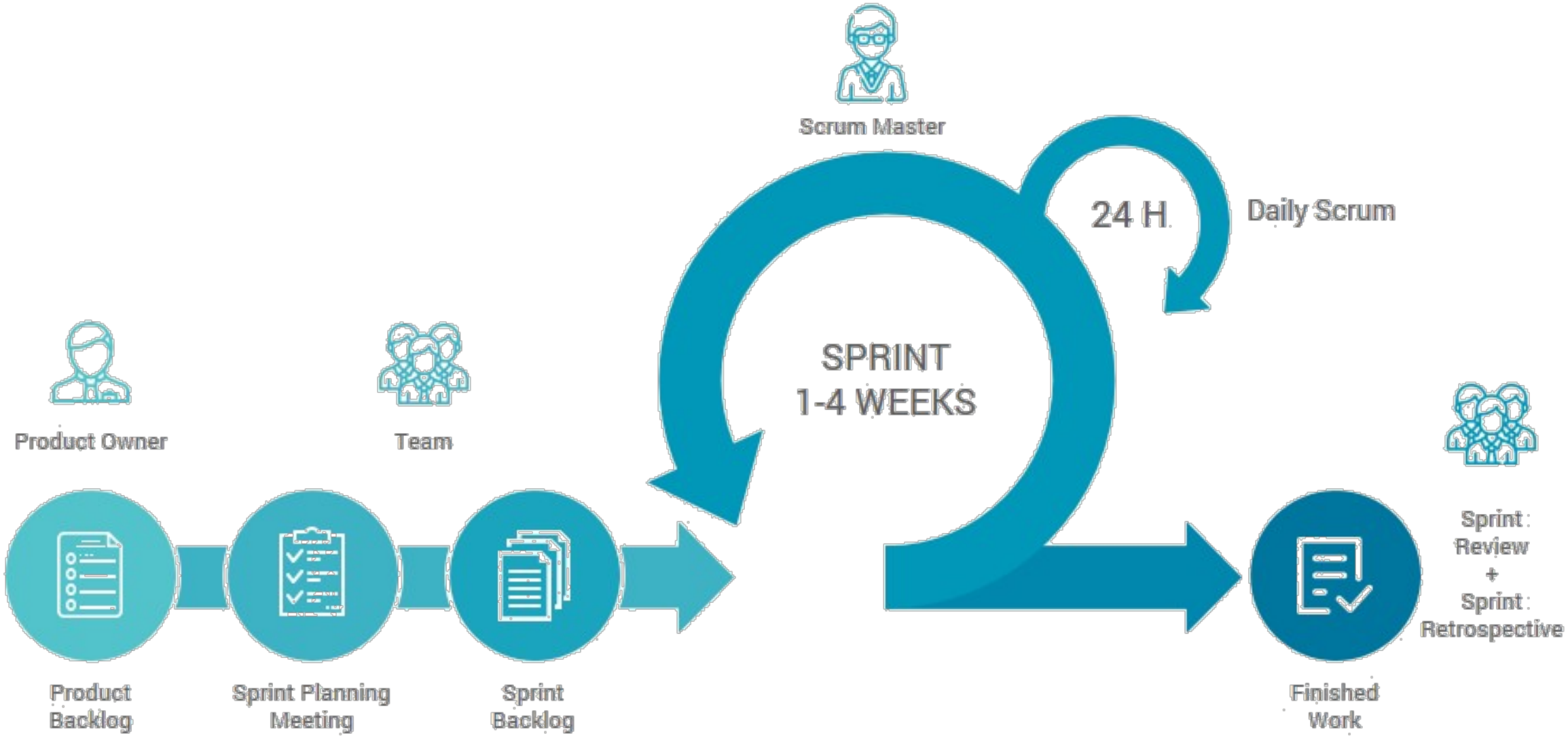
Kanban Board



Track Progress with Burndown charts



Agile Process Overview



Sprint Retrospective

In between sprints, the team inspects itself and creates a plan for self-improvement for the next sprint.

- What went well in the sprint?
- What could be improved?
- What will we commit to improve in the next sprint?

- **For your teams:** You will conduct sprint retrospectives after Sprint #1 (Feb 18), #3 (March 11), #5 (March 25), and at the end of the semester (April 24).

Exercise: Mini P0 retro

In between sprints, the team inspects itself and creates a plan for self-improvement for the next sprint.

- What went well in the sprint?
- What could be improved?
- What will we commit to improve in the next sprint? **One key thing**

Development best practices

Development best practices

You'll practice some of these in HW3, HW4, and HW5. You'll also be using these practices when developing your product.

Development best practices (GH mostly)

- CI
- Small Releases
- Feature Branches
- Pull Requests
- Branch Protection Rules
- Code Review
- GitHub Issues
- GitHub Project
  - Kanban Board
- GitHub Actions
  - Continuous Integration
  - Continuous Delivery

Continuous Integration (CI)

New code is *integrated* with the current system quickly after checkin.

When integrating, the system is built from scratch and all tests must pass, or the changes are discarded.

For your teams: Your team must run tests on every checkin.

Small Releases

The system is put into production in a few months, before solving the whole problem. New releases are made often—anywhere from daily to monthly.

Small releases imply small bugs. Most bugs are caused by yesterday's checkins.

For your teams: You will release daily.

Jobs

- build
- deploy-preview
- deploy-prod

Run details

- Usage
- Workflow file

[redacted] main | Success | 3h 11m 57s | 1

deploy.yml
on: push

build	2m 31s	deploy-preview	46s	deploy-prod	44s

Deployment protection rules
Reviewers, timers, and other rules protecting deployments in this run

Event	Environments	Comment
[redacted] approved yesterday	prod	

Feature Branches

- Whenever you start working on a user story, create a new named branch with Git.

```
git branch my-new-feature
git checkout my-new-feature
```

- All code work should occur on the new branch.

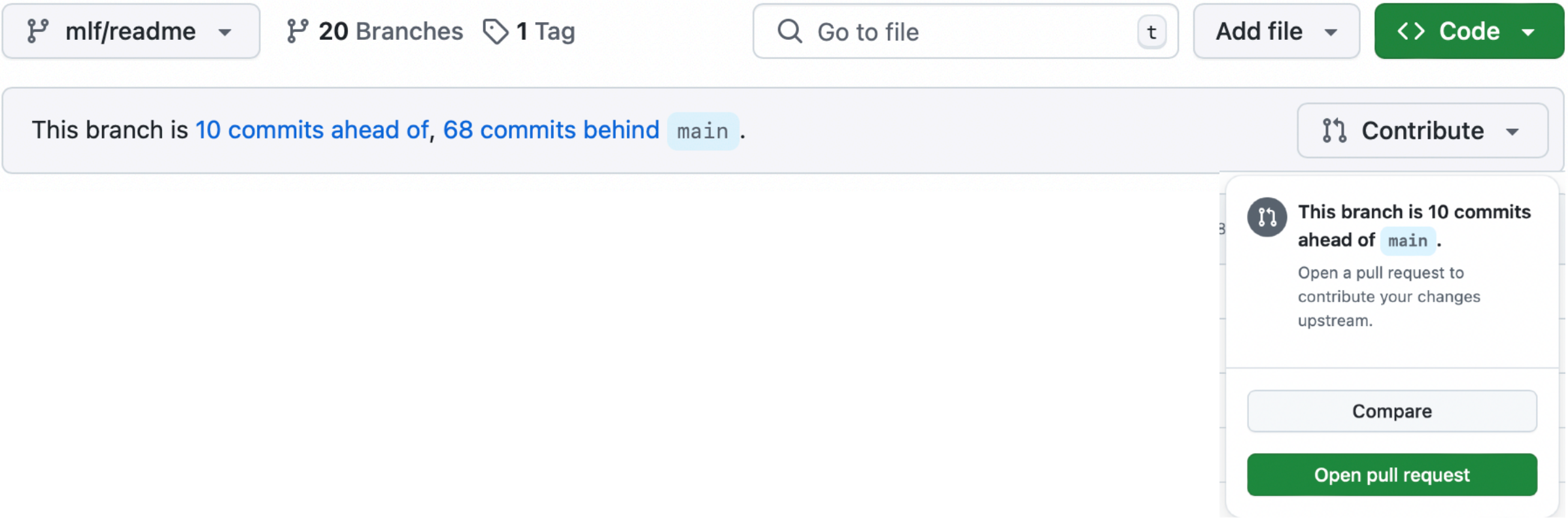
```
<write some code>
git add newfile.js
git commit –m “Made a new file.”
git push –u origin my-new-feature
```

- When you are done with the work, submit a *pull request* to merge the branch back into main.

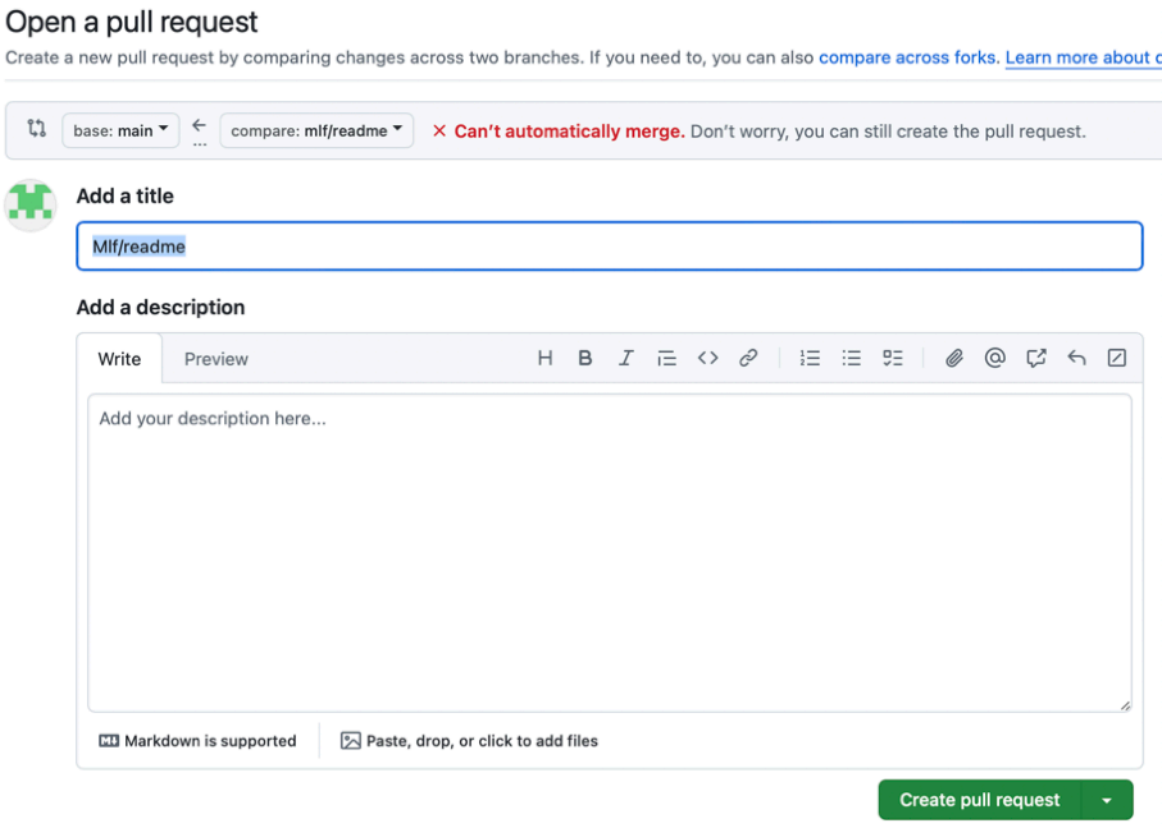
Pull Requests

- A proposal to merge a set of changes from one branch into another.
- Do this with the GitHub UI.
- Once a pull request has been submitted, at least one team member, preferably the user story's secondary owner, should review the code.
- If it passes review, approve the pull request to have the code merged into main.
- If it doesn't pass review, continue the code review conversation and potentially add additional checkins to the pull request to resolve the issues.

Pull Requests



Pull Requests



Code Review (we will discuss more)

- All checked in code must be reviewed by at least one member of your team.
- Reviewer looks over the code and makes comments and asks questions of the committer on every change that looks suspicious.
- Reviewer and committer have conversations that may include additional commits until they come to agreement.
- Reviewer signs off on the checkin.

Branch Protection Rules

- All branches should be protected with enforced processes.
- Turn on these options in your repository settings in GitHub.
  - Require pull requests
  - Require approvals (at least 1)
  - Require review from code owners
    - In case a non-owner checks in code.
  - Require status checks to pass before merging.
  - Require branches to be up to date before merging.
  - Require deployments to succeed before merging.
  - Turn off Allow force pushes.

Branch Protection Rules

- All branches should be protected with enforced processes.
- Turn on these options in your repository settings in GitHub.
  - Require pull requests
  - Require approvals (at least 1)
  - Require review from code owners
    - In case a non-owner checks in code.
  - Require status checks to pass before merging.
  - Require branches to be up to date before merging.
  - Require deployments to succeed before merging.
  - Turn off Allow force pushes.

Why…

GitHub Issues

- Put all your user stories and bug reports here.
- Use a template to fill in these fields:
  - Priority (0, 1, 2)
  - Assigned owners (Primary and Secondary)
  - Date due (date you intend to check it in)
  - Sprint assignment (sprint number it will be done in)
  - Label – Bug Report/User Story
  - Acceptance criteria

GitHub Project

- Create a GitHub project in your GitHub organization.
- Attach it to your repo to pull in the user stories from your GitHub Issues.
- Projects give you
  - Kanban Boards
  - Gantt Charts
  - Burnup Charts

Kanban board

Upstream issues to track 4

https://github.com/git-lfs/git-lfs/issues/2627

Add card | Cancel

Git LFS 2.3.1 seems to break Windows
#2627 opened by larsxschneider

docker build limit io disk
#35012 opened by sztwiorok
area/builder | kind/feature

Reference to moby/moby

repl: allow `await` in REPL
#13209 opened by benjamingr
cli | feature request | promises | repl

Reference to nodejs/node

New things to check out 4

Implement split diffs
1 of 6
#866 opened by BinaryMuse
work-in-progress

Reference to atom/github

Change license and remove references to PATENTS
#10804 opened by sophiebits
CLA Signed

Reference to facebook/react

"Clone in Desktop" flow now recognizes gists
#2939 opened by shiftkey
ready-for-review

Reference to desktop/desktop

Fixes to upgrade for 4

#3311 opened by kdzwinel
audit

Reference to GoogleChrome/lighthouse

Error: Undefined variable: "$h1-size-mobile"
#229 opened by kaelig

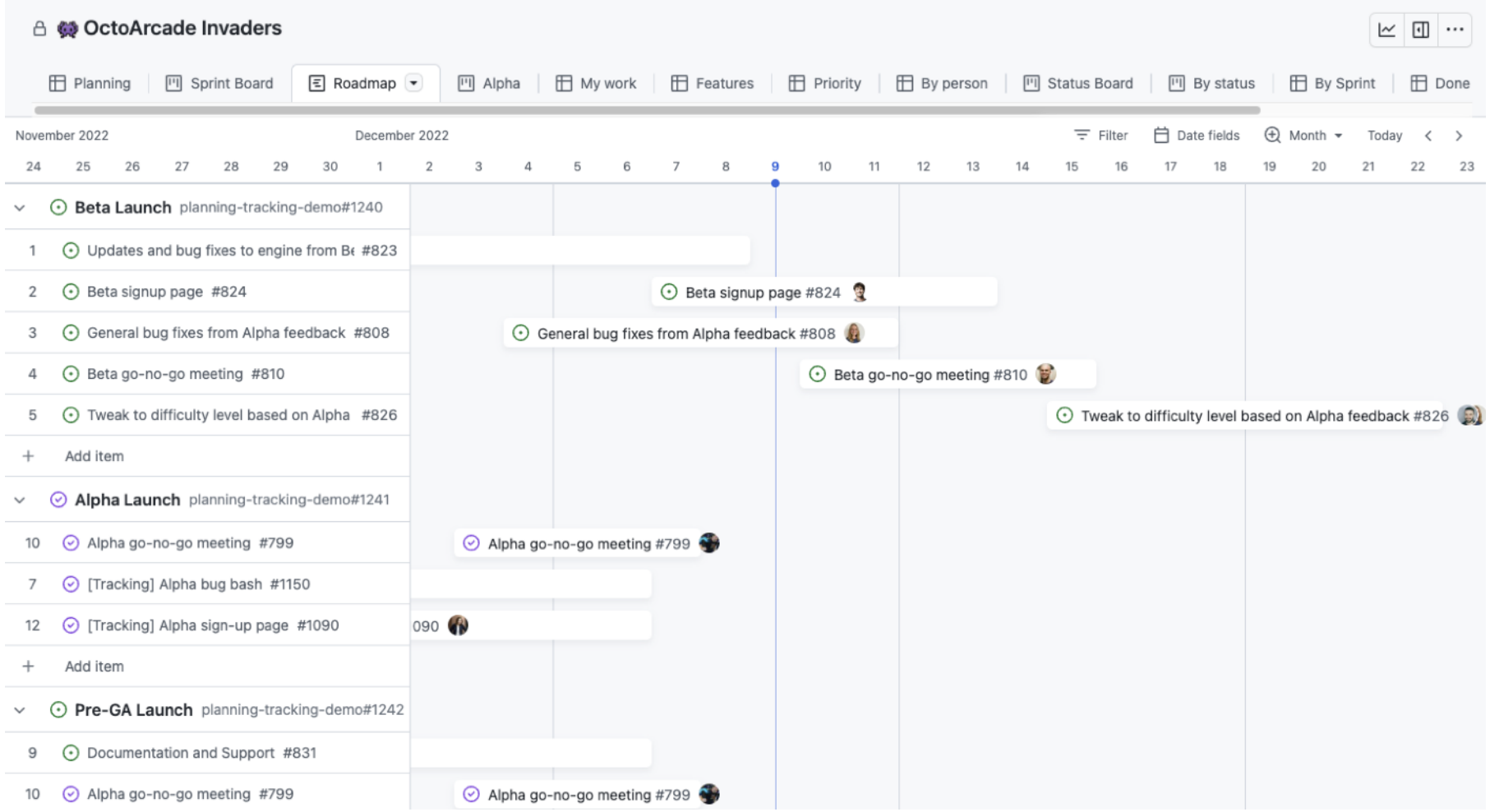
Reference to primer/primer-css

util: use faster -0 check
3 of 3
#15726 opened by mscdex
performance | util

Reference to nodejs/node

Git LFS 2.3.1 seems to break Windows
#2627 opened by larsxschneider

Roadmap (i.e. Gantt Chart)



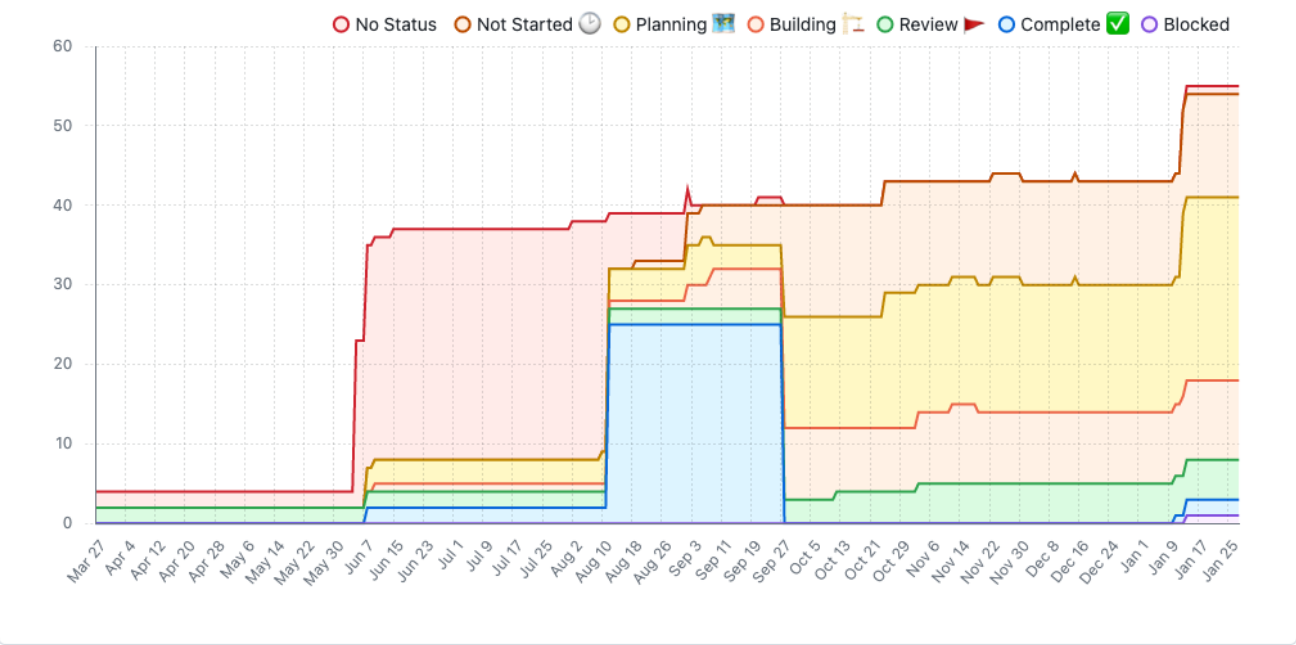
Burndown (Burnup Chart)

Burn up

Burn up shows the progress being made towards completion of the project, development flow, and scope changes over time. Use this chart to identify bottlenecks and issues blocking the team from making progress.

Filter by keyword or by field

2W 1M 3M max Custom



GitHub Actions

- Set up actions for continuous integration and continuous deployment
- Create YAML files that live in .github/workflows folder.

Code | Pull requests 11 | Actions

Actions

New workflow

All workflows

Dependabot Updates

Deploy to Netlify

PR checks

Semgrep

Management

Caches

Deployments

Attestations

Runners

Usage metrics

Performance metrics

Process for this class

- Each team must have a GitHub repository
- The project backlog will be in GitHub Issues.
- Each repository must have a project board.
  - The project board will have cards for (at least) all issues in the sprint backlog.
  - The project board will keep track of issues in progress, done, etc.
- Each card will have a number of attributes.
  - Tags labeling features, bugs, t-shirt size, etc.
  - All in-progress cards must have at least two assigned team members.
- All checkins must be done by pull request, not pushed directly to main branch.
- All pull requests must be linked to a card to support traceability.