Better (Small) Scientific Software Teams

Better Scientific Software Tutorial

David E. Bernholdt

Oak Ridge National Laboratory

ISC High Performance Conference

June 16, 2019

License, citation, and acknowledgments

License and Citation

- This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
- Requested citation: **David E. Bernholdt and Michael A. Heroux, Better (Small) Scientific Software Teams, in Better Scientific Software Tutorial, ISC High Performance Conference, Frankfurt, Germany, 2019. DOI: https://doi.org/10.6084/m9.figshare.8242859**

Acknowledgements

- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- This work was performed in part at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.
- Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Outline

- Small Team Models, Challenges.
- Agile workflow management for small teams
  - Intro to terminology and approaches
  - Overview of Kanban
  - Free tools: Trello, GitHub.
- Hands-on example of project management using GitHub

Small Teams

Ideas for managing transitions and steady work.

Small team interaction model

- Team composition:
  - Senior staff, faculty:
    - Stable presence, in charge of science questions, experiments.
    - Know the conceptual models well.
    - Spend less time writing code, fuzzy on details.
  - Junior staff, students:
    - Transient, dual focus (science results, next position).
    - Staged experience: New, experienced, departing.
    - Learning conceptual models.
    - Write most code, know details.

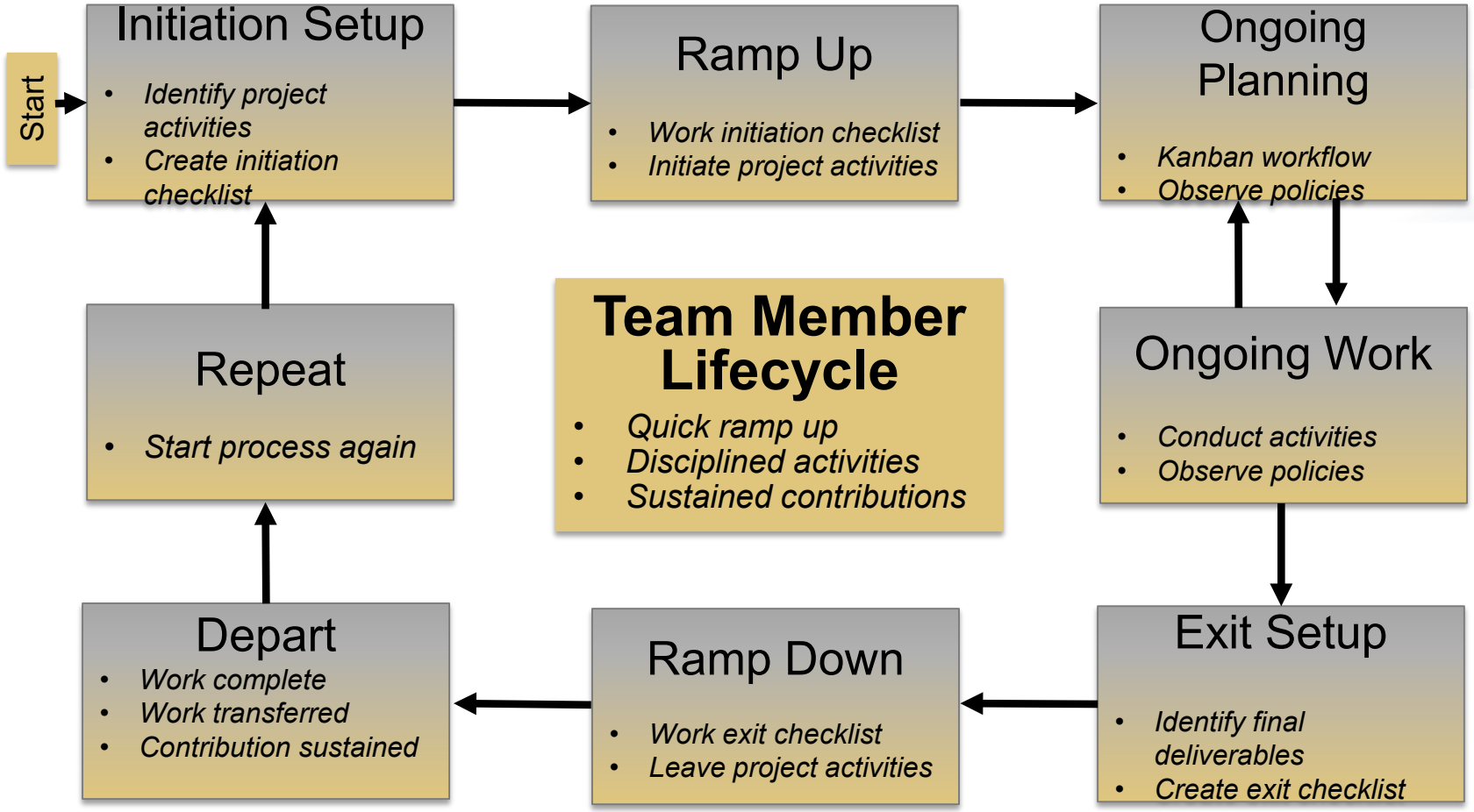
Large team challenges

- Composed of small teams (and all the challenges).
- Additional interaction challenges.
- Policies, regularly cultural exchanges important.

Small team challenges

- Ramping up new junior members:
  - Background.
  - Conceptual models.
  - Software practices, processes, tools.
- Preparing for departure of experienced juniors.
  - Doing today those things needed for retaining work value.
  - Managing dual focus.

Research Team Member Lifecycle



Checklists & Policies

Team Member Phase		
New Team Member	Steady Contributor	Departing Member
Checklist	Policies	Checklist

- New, departing team member checklists:
  - Example: Trilinos New Developer Checklist.
  - https://software.sandia.gov/trilinos/developer/sqp/checklists/index.html
- Steady state: Policy-driven.
  - Example: xSDK Community policies.
  - https://xsdk.info/policies/

Your checklists & policies?

- Checklist: New team member?
- Policies: Ongoing work?
- Checklist: Before someone departs?

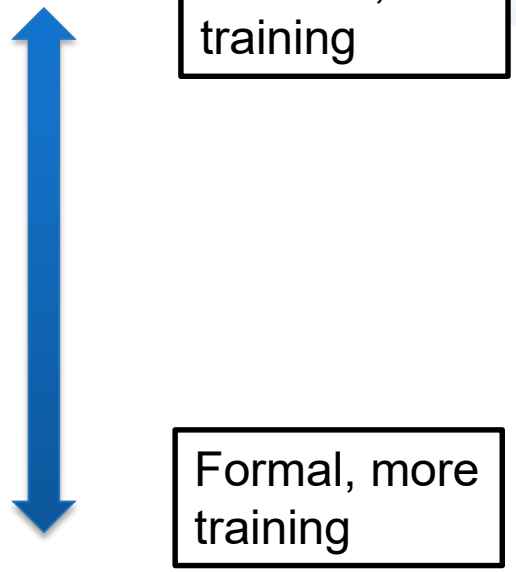
Collaborative Work Management

Managing with Kanban

Managing issues: Fundamental software process

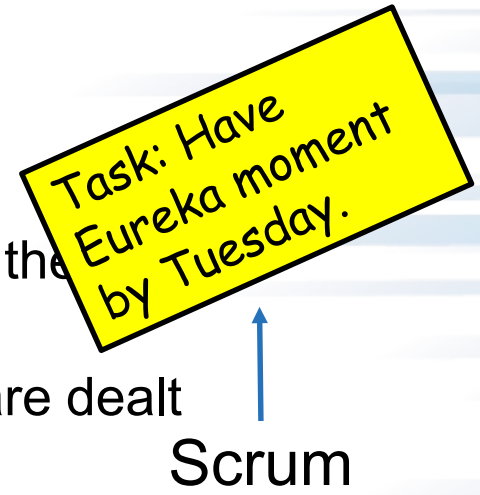
Continual improvement

- Issue: Bug report, feature request
- Approaches:
  - Short-term memory, office notepad
  - ToDo.txt on computer desktop (1 person)
  - Issues.txt in repository root (small co-located team)
  - …
  - Web-based tool + Kanban (distributed, larger team)
  - Web-based tool + Scrum (full-time dev team)



Kanban principles

- Limit number of “In Progress” tasks
- Productivity improvement:
  - Optimize “flexibility vs swap overhead” balance. No overcommitting.
  - Productivity weakness exposed as bottleneck. Team must identify and fix the bottleneck.
  - Effective in R&D setting. Avoids a deadline-based approach. Deadlines are dealt with in a different way.
- Provides a board for viewing and managing issues
- *Can be applied to any existing software project immediately!*



Basic Kanban

Backlog	Ready	In Progress	Done
• Any task idea • Trim occasionally • Source for other columns	• Task + description of how to do it. • Could be pulled when slot opens. • Typically comes from backlog.	• Task you are working on *right now.* • **The only kanban rule: Can have only so many "In Progress" tasks.** • Limit is based on experience, calibration. • **Key: Work is *pulled*. You are in charge!**	• Completed tasks. • Record of your life activities. • Rate of completion is your "velocity".

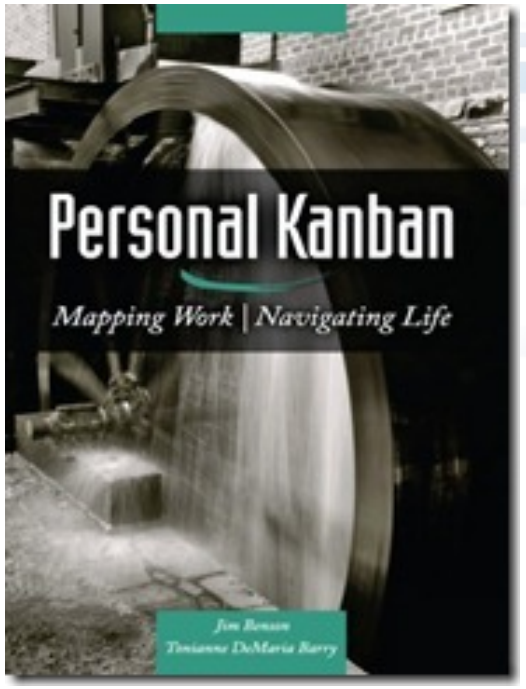
Notes:

- Ready column is not strictly required, sometimes called "Selected for development".
- Other common column: In Review
- Can be creative with columns:
  - Waiting on Advisor Confirmation.
  - Tasks I won't do.

Personal Kanban

- Personal Kanban: Kanban applied to one person.
  - Apply Kanban principles to your life.
  - Fully adaptable.

- Personal Kanban: Commercial book/website.
  - Useful, but not necessary.



http://www.personalkanban.com

Kanban tools

- Wall, whiteboard, blackboard: Basic approach.
- Software, cloud-based:
  - Trello, JIRA, GitHub Issues.
  - Many more.
- I use Trello (browser, iPhone, iPad).
  - Can add, view, update, anytime, anywhere.

Big question: How many tasks?

- Personal question.
- Approach: Start with 2 or 3. See how it goes.
- Use a freeway traffic analogy:
  - Does traffic flow best when fully packed? No.
  - Same thing with your effectiveness.
- Spend time consulting board regularly.
  - Brings focus.
  - Enables reflection, retrospection.
  - Use slack time effectively.
  - When you get out of the habit, start up again.

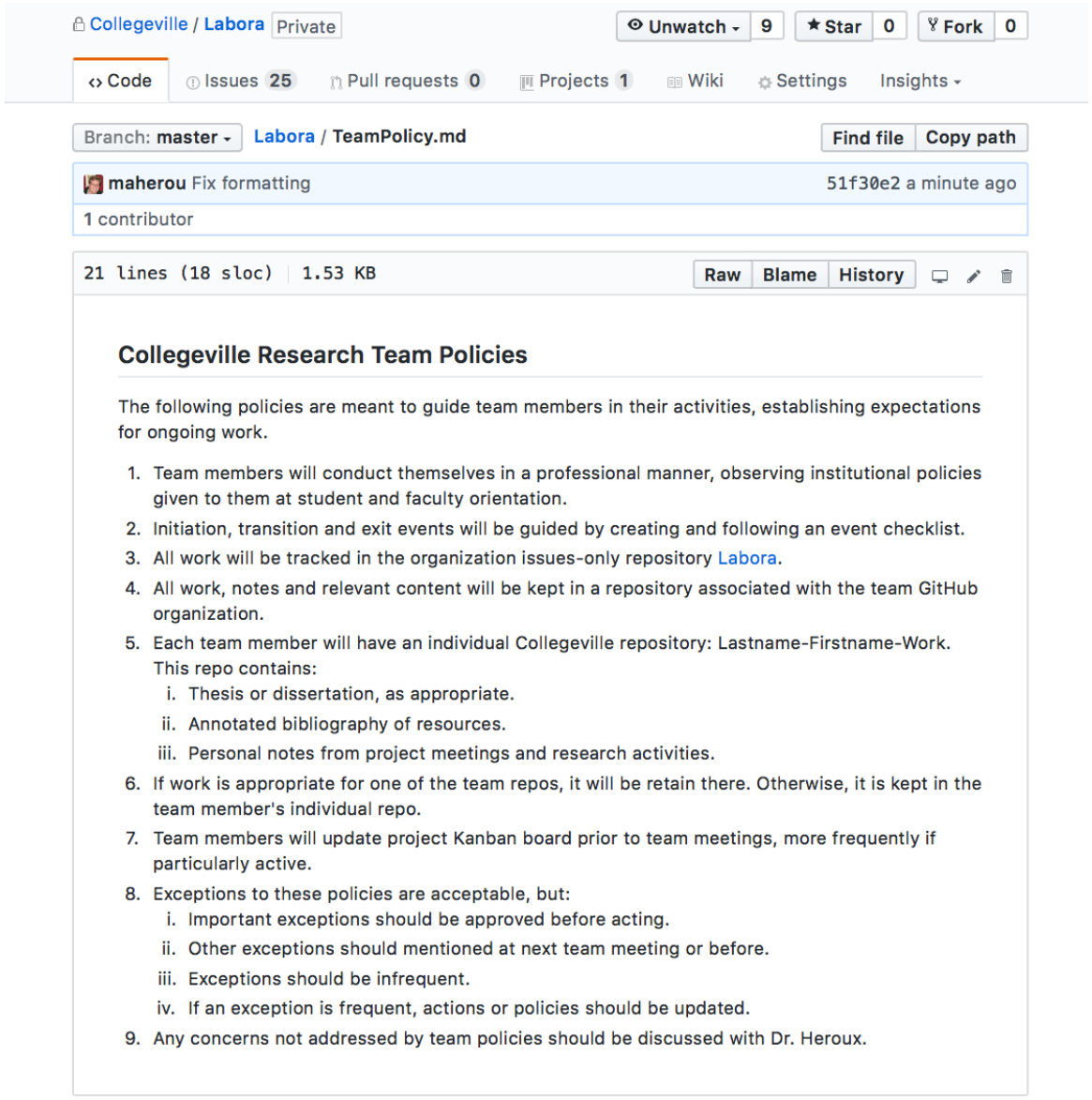
Importance of “In Progress” concept for you

- Junior community members:
  - Less control over task.
  - Given by supervisor.
- In Progress column: Protects you.
  - If asked to take on another task, respond:
    - Is this important enough to become less efficient?
    - Sometimes it is.

Key Team Management Elements

- **Checklists:**
  - Initiation, Transition, Exit
- **Policies:**
  - How team conducts its work
- **Issue tracking system:**
  - All work tracked, visible to team
  - Milestones: Aggregate related issues.
  - Kanban board
  - Regular meetings, updates

Samples from Collegeville Org: Policies, Initiation Checklist



Collegeville / Labora Private

Unwatch 9 | Star 0 | Fork 0

Code | Issues 25 | Pull requests 0 | Projects 1 | Wiki | Settings | Insights

Branch: master ▾ | Labora / TeamPolicy.md | Find file | Copy path

maherou Fix formatting — 51f30e2 a minute ago

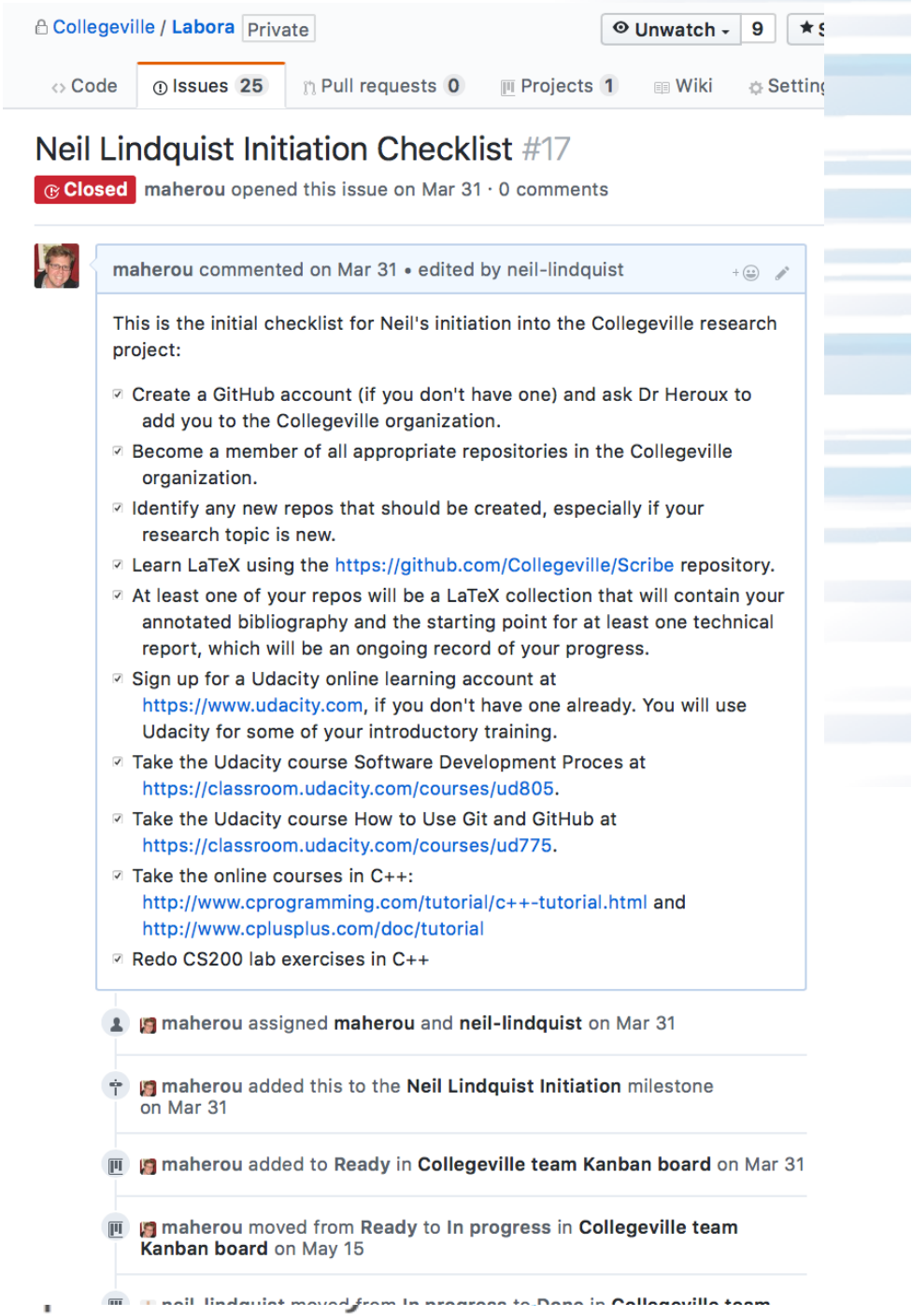
1 contributor

21 lines (18 sloc) | 1.53 KB | Raw | Blame | History

Collegeville Research Team Policies

The following policies are meant to guide team members in their activities, establishing expectations for ongoing work.

1. Team members will conduct themselves in a professional manner, observing institutional policies given to them at student and faculty orientation.
2. Initiation, transition and exit events will be guided by creating and following an event checklist.
3. All work will be tracked in the organization issues-only repository Labora.
4. All work, notes and relevant content will be kept in a repository associated with the team GitHub organization.
5. Each team member will have an individual Collegeville repository: Lastname-Firstname-Work. This repo contains:
   i. Thesis or dissertation, as appropriate.
   ii. Annotated bibliography of resources.
   iii. Personal notes from project meetings and research activities.
6. If work is appropriate for one of the team repos, it will be retain there. Otherwise, it is kept in the team member's individual repo.
7. Team members will update project Kanban board prior to team meetings, more frequently if particularly active.
8. Exceptions to these policies are acceptable, but:
   i. Important exceptions should be approved before acting.
   ii. Other exceptions should mentioned at next team meeting or before.
   iii. Exceptions should be infrequent.
   iv. If an exception is frequent, actions or policies should be updated.
9. Any concerns not addressed by team policies should be discussed with Dr. Heroux.



Collegeville / Labora Private

Unwatch 9

Code | Issues 25 | Pull requests 0 | Projects 1 | Wiki | Setting

Neil Lindquist Initiation Checklist #17

Closed maherou opened this issue on Mar 31 · 0 comments

maherou commented on Mar 31 • edited by neil-lindquist

This is the initial checklist for Neil's initiation into the Collegeville research project:

- [x] Create a GitHub account (if you don't have one) and ask Dr Heroux to add you to the Collegeville organization.
- [x] Become a member of all appropriate repositories in the Collegeville organization.
- [x] Identify any new repos that should be created, especially if your research topic is new.
- [x] Learn LaTeX using the https://github.com/Collegeville/Scribe repository.
- [x] At least one of your repos will be a LaTeX collection that will contain your annotated bibliography and the starting point for at least one technical report, which will be an ongoing record of your progress.
- [x] Sign up for a Udacity online learning account at https://www.udacity.com, if you don't have one already. You will use Udacity for some of your introductory training.
- [x] Take the Udacity course Software Development Proces at https://classroom.udacity.com/courses/ud805.
- [x] Take the Udacity course How to Use Git and GitHub at https://classroom.udacity.com/courses/ud775.
- [x] Take the online courses in C++: http://www.cprogramming.com/tutorial/c++-tutorial.html and http://www.cplusplus.com/doc/tutorial
- [x] Redo CS200 lab exercises in C++

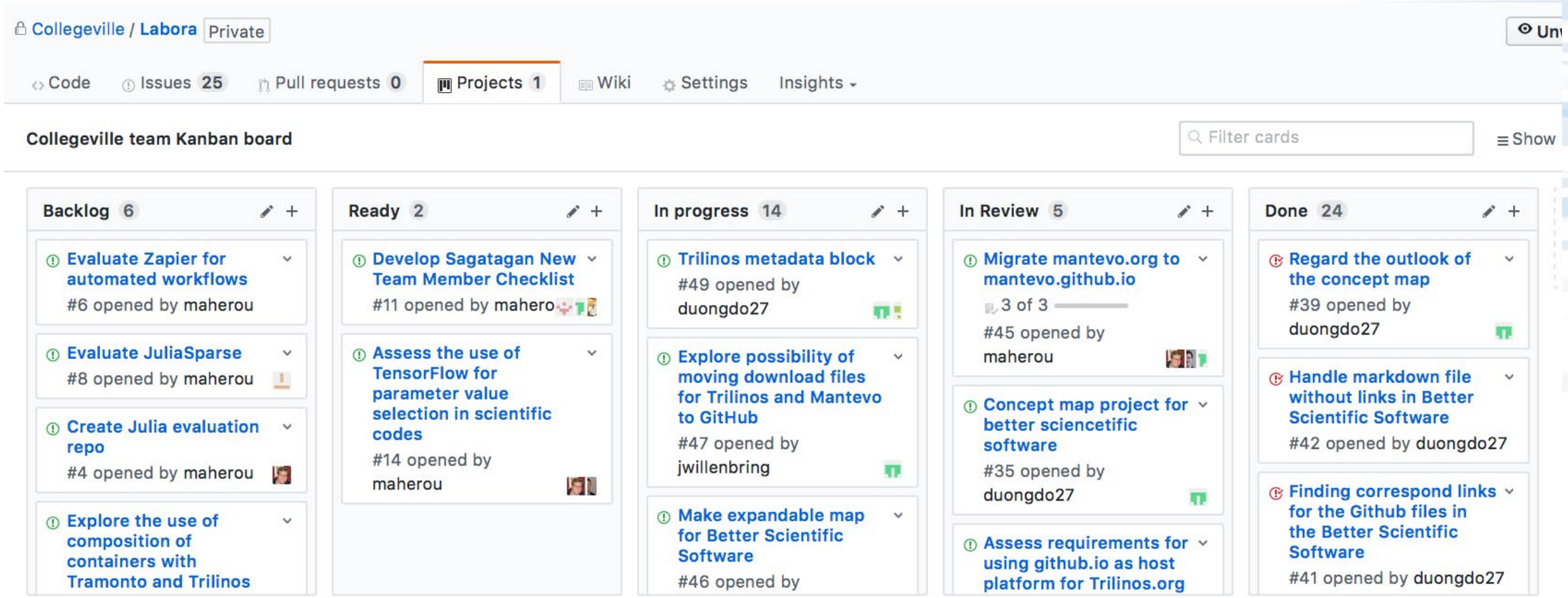
maherou assigned maherou and neil-lindquist on Mar 31

maherou added this to the Neil Lindquist Initiation milestone on Mar 31

maherou added to Ready in Collegeville team Kanban board on Mar 31

maherou moved from Ready to In progress in Collegeville team Kanban board on May 15

Samples from Collegeville Org: Kanban Board



Team Management Example

Team Policy

Checklists

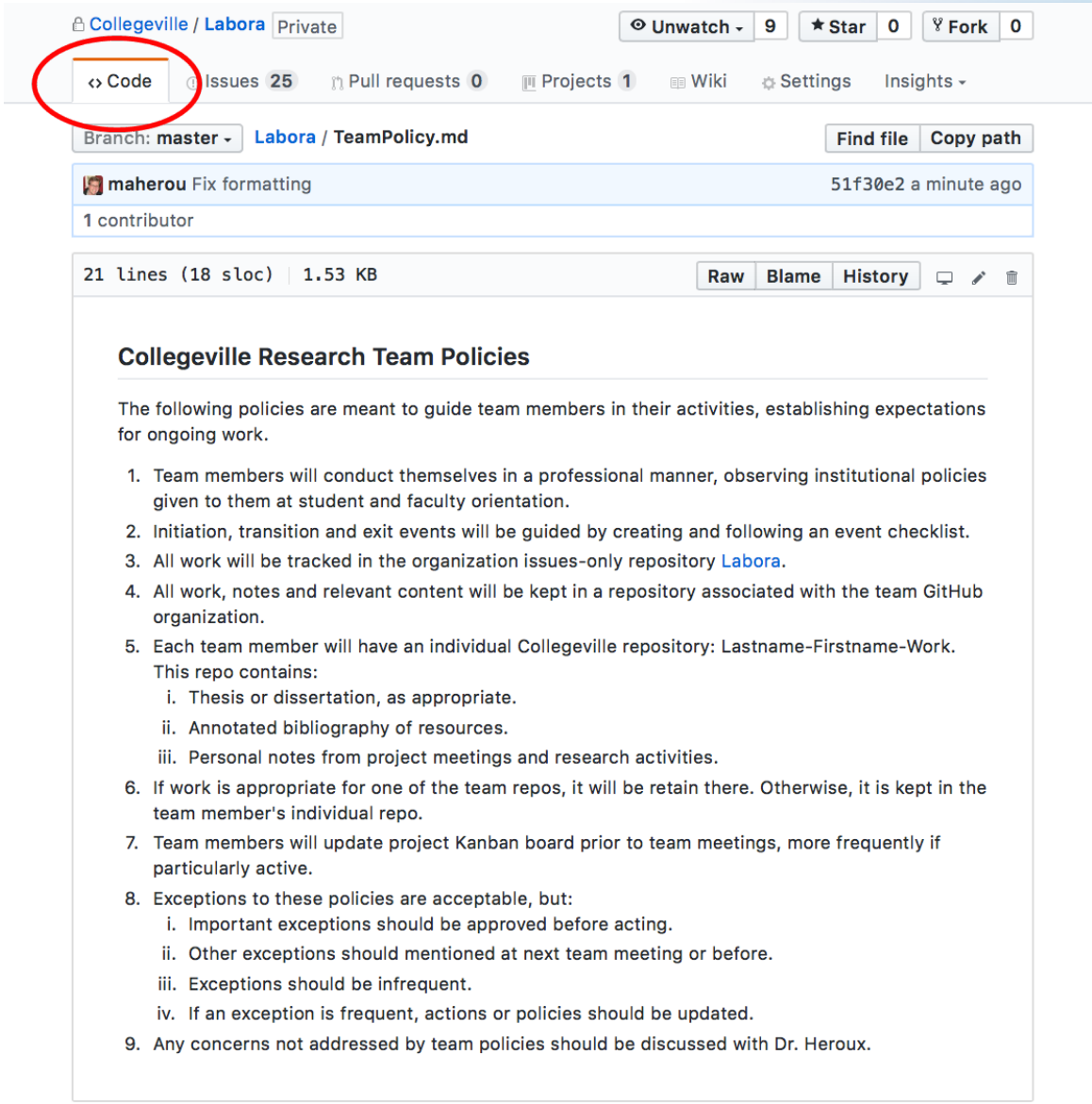
Kanban Board

Step 1: Create Issues-only GitHub repo

- Go to https://github.com/username
  - Example: https://github.com/maherou
- Create new repo:
  - Click on “+” (upper right).
  - Select New repository…
  - Give repo a name, e.g., **Issues**
  - Select Public. In real life, this repo is often private (requires $ or special status)
  - Init with README.
  - Don’t add .gitignore or license.
  - Click Create Repository.

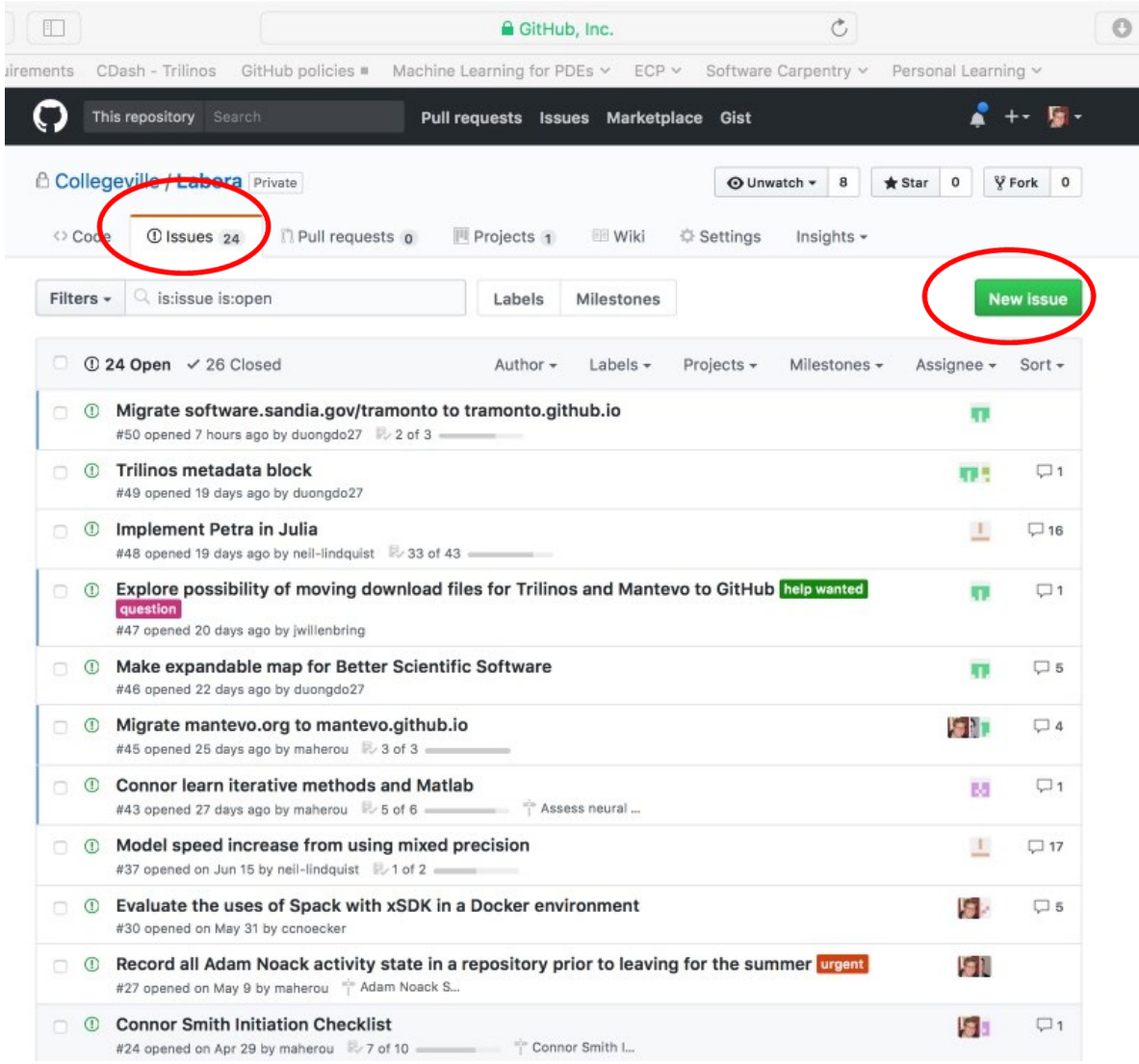
Step 2: Define Team Policy

- Create file:
  - Go to new repo: Issues.
  - Select <> Code tab.
  - Select Create new file TeamPolicy.md
- Questions to address:
  - How members support team?
  - How team supports members?
- Community version:
  - http://contributor-covenant.org
- Policy is living document:
  - Informal good practices added.
  - Avoidable bad situations addressed.



Step 3a: Create Issues

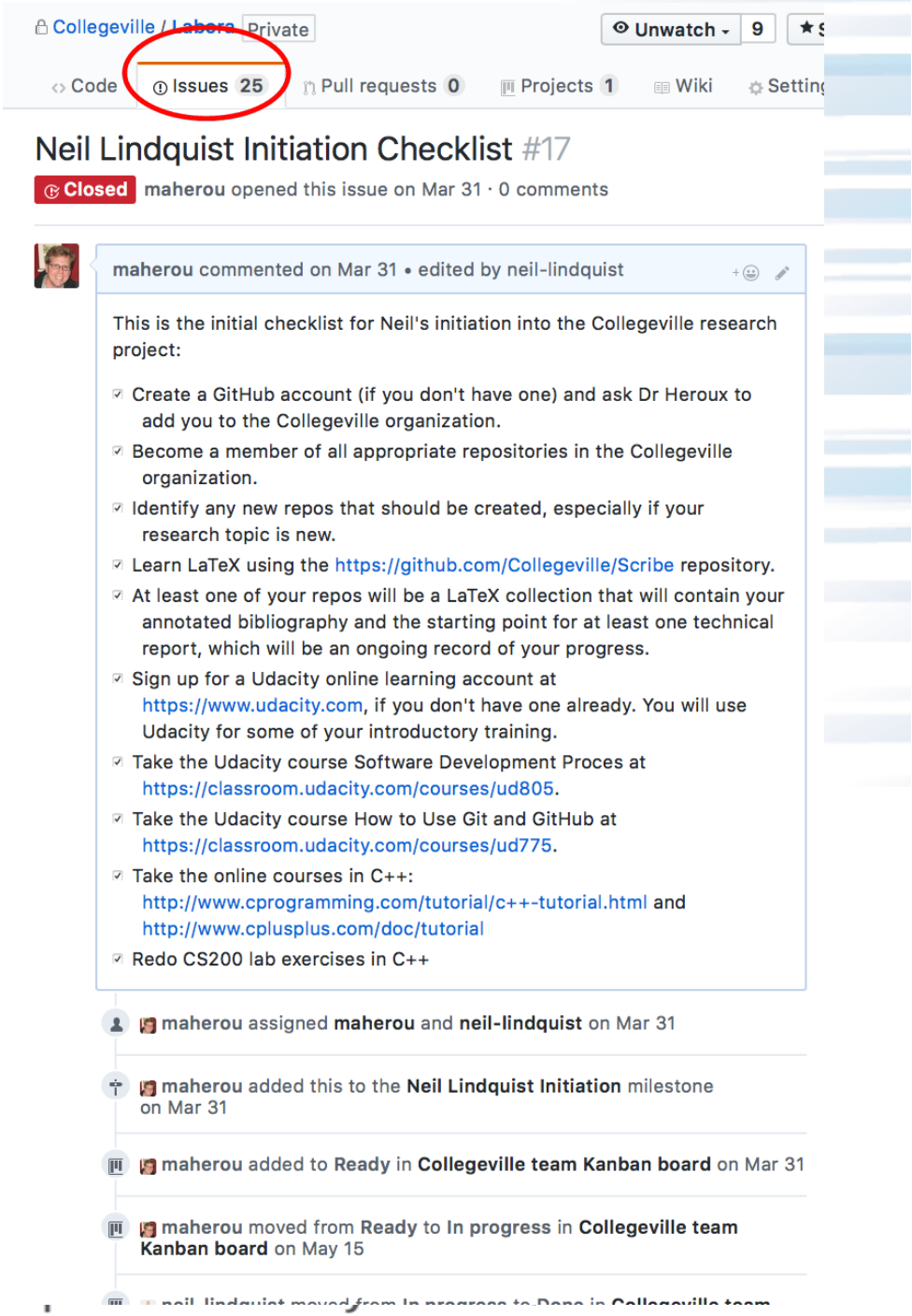
- Select the Issues tab.
- Click on New Issue.
- Type in task statement 1 (from list).
  - Type in title only.
- Click Submit new issue
- Repeat.



Step 3b: Create Initiation Checklist

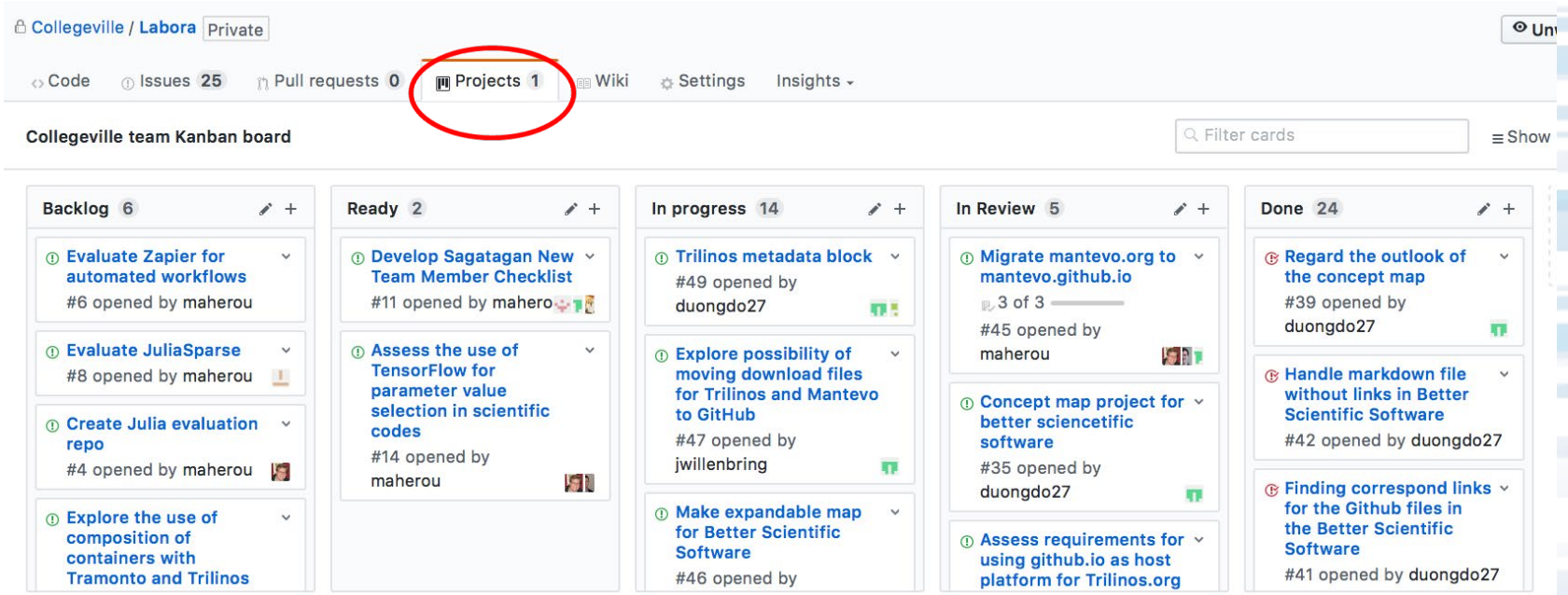
- Select the Issues tab.
- Click on New Issue.
- Select a classmate.
- Type in title: Pat Evans Initiation Checklist
- Add checklist items:
  - Use syntax:
    - [ ] Description

Spaces required



Step 4: Create Kanban Board

- Select Projects tab
- Click New Project
- Use title
  - Team Kanban board
- Add these columns:
  - Backlog, Ready, In progress,
- Click on +Add cards (upper right).
  - Move each issue to the proper Kanban column

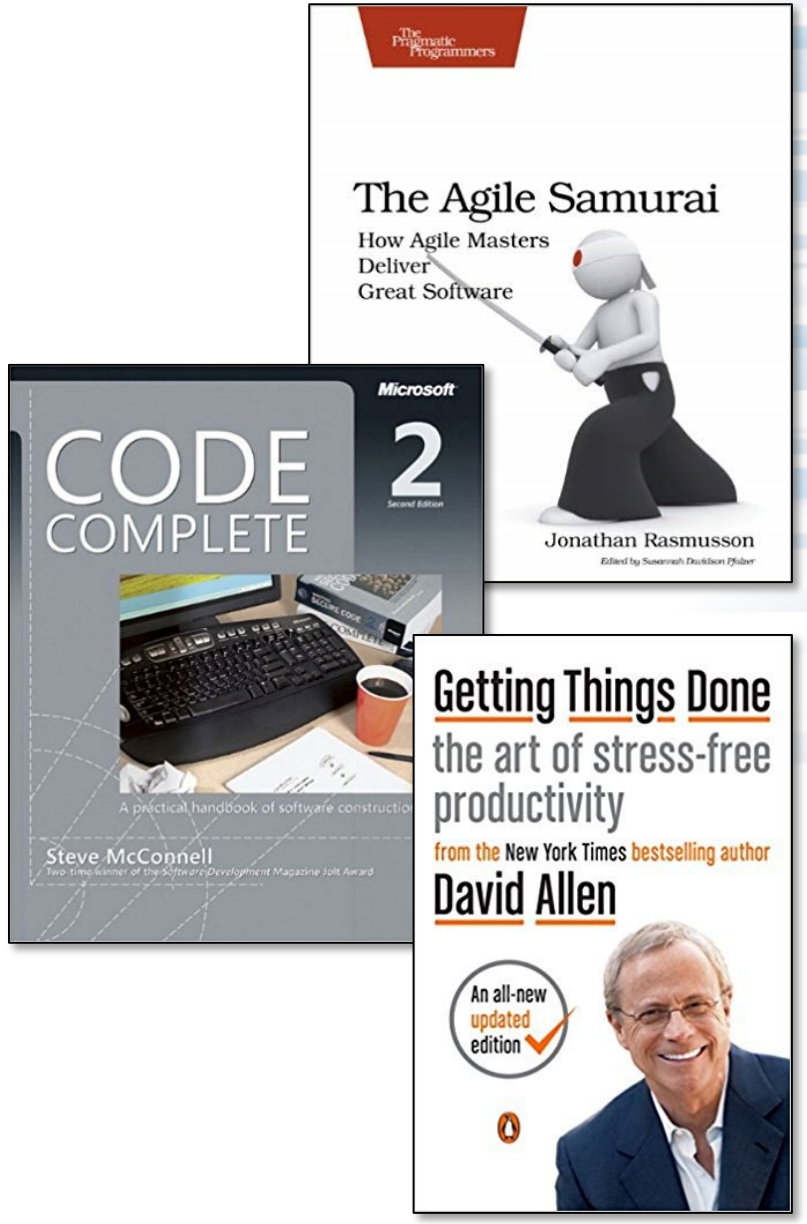


Next Steps: Real Life

- Create a GitHub Org and set of repos for your team:
  - Each team member has an individual repo.
  - Each project has a repo.
  - One special repo for issues.
- Track all work:
  - Use checklists for initiation, exit, any big new effort.
  - Create Kanban board. Keep it current.
  - Aggregate related issues using milestones.
- Drive meetings using Kanban board.
- Adapt this approach to meet your needs.
- When you start to get sloppy, get back on track.

Other Resources

- **The Agile Samurai: How Agile Masters Deliver Great Software (Pragmatic Programmers),** Jonathan Rasmusson.
  - http://a.co/eUGIe95
  - Excellent, readable book on Agile methodologies.
  - *Also available on Audible.*
- **Code Complete: A Practical Handbook of Software Construction,** Steve McConnell.
  - http://a.co/eEgWvKj
  - Great text on software.
  - *Construx website has large collection of content.*
- **Getting Things Done: The Art of Stress-Free Productivity,** David Allen
  - http://a.co/22EPvt6
  - A classic in the personal productivity literature



IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Agenda

Time	Module	Topic	Speaker
2:00pm-2:40pm	01	Overview of Best Practices in HPC Software Development	Anshu Dubey, ANL
2:40pm-3:20pm	02	Better (Small) Scientific Software Teams	David E. Bernholdt, ORNL
3:20pm-4:00pm	03	Improving Reproducibility through Better Software Practices	David E. Bernholdt, ORNL
4:00pm-4:30pm		*Break*	
4:30pm-5:15pm	04	Verification & Refactoring	Anshu Dubey, ANL
5:15pm-6:00pm	05	Git Workflow & Continuous Integration	Jared O'Neal, ANL

https://r.isc-hpc.com/tut130

(Please note the R before our domain)

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT