Teams

Agile Team Organization

- Hero engineer is no longer viable
- Plan-and-document depends on an experienced manager

Scrum

- '2 Pizza' team size $\triangleq$ 4-9 people
- Name inspired by frequent short meetings
  - Generally daily stand-ups
- Meetings answer 3 questions:
  1. What have you done since yesterday?
  2. What are you planning to do today?
  3. Are there any impediments or stumbling blocks?
- Work in 'sprints' of 2-4 weeks

Roles

- Good to rotate through roles (especially product owner) each iteration
- Scrum Master
  - Acts as buffer between the team and external distractions
  - Keeps team focused on task at hand
  - Enforces team rules (coding standard)
  - Removes impediments that prevent team from making progress
- Product Owner
  - Not the scrum master
  - Represent the voice of the customer and prioritizes user stories

Conflicts

- List all items on which the sides agree
  - Don't start with list of disagreements
- Each side articulates the other's arguments, even if they don't agree with some
  - Avoids confusion about terms or assumptions
- Constructive confrontation $\triangleq$ if you have a strong opinion that a person is proposing the wrong thing technically, you are obligated to bring it up even to your bosses
- Disagree and commit $\triangleq$ once decision made, need to embrace it and move ahead

Branches

- Creating branch is cheap
- Switch among branches with `checkout`
- Separate commit histories per branch
- Merge branch back into master

Branch Per Feature

- To work on a new feature, create a new branch just for that feature
- Use branch only for changes needed for this feature
- Able to remove feature with revert
- Ideally, branch doesn't touch too many parts of app

Rebase and Pull Request

- Rebase branch against $x \triangleq$ try to pretend it was branched from x
  - Must resolve merge conflicts manually
  - Can be used to 'incrementally' merge
- Make a pull request (or merge) against main
  - Avoids conflicts
- `git cherry-pick` can be used to merge only specific commits

Pull Request (PRs)

- Requesting someone else (or yourself) to pull your changes into another branch
- Happen within or between repos
  - Don't need to be main, can be any branch
- Typically comes with a description
- Provide 'hooks' for automation and tools
- Opening PR means expect other team members to review and comment on it
- Depending on review outcome, PR may be closed (withdrawn) or revised before merging

Deployment

- Automatic deployment strategies
- Deploy from master is the most common
- Branch per release is alternate strategy
  - Generate a new branch for each release

Branch vs Fork

- Branch if you have commit access to repo
  - Can merge with team approval
  - Creates a branch in the actual repo
- Fork if you do not have commit access to repo
  - Courtesy: check with maintainer about their process
  - Fork: clone entire repo on GitHub to one that you can branch, push, etc.
  - Finalize your work on branch
  - Courtesy: rebase your branch w/ commit squash
  - Open PR, so the maintainer van pull the commit

Gitfalls

- Stomping on changes after doing a merge or switching branches
- Making 'simple' changes directly on master branch
  - Never edit main / master

Git Undo Strategies

- `git reset --hard ORIG_HEAD` $\triangleq$ reset to head of current origin branch
- `git reset --hard HEAD` $\triangleq$ reset to last local commit
- `git checkout` $\triangleq$ reset to a commit (can specify files with `-- <file names>`)
- `git diff` $\triangleq$ see difference (can specify files with `-- <file names>`)
  - Can specify hashes, times, and days like: `git diff "main@{2 days ago}"`
- `git show <branch>:<file>` to show the status of just that file
- `git blame <files | commit>` to see who last touched a file
- `git log` gives a log of git actions

Reviews

- Design review $\triangleq$ Meeting where authors present design
  - Benefit from experience of attendees
- Code review $\triangleq$ held after design implemented
  - Looks at code
- Prepare with list of questions / issues to be discussed
- Start with high-level description of customer desires
- Give SW architecture, showing APIs and highlighting design patterns at each level of abstraction
- Go through code and documentation: project, plan, schedule, testing plan, ...
- Approach reviews $\triangleq$ a few senior developers assist team in coming up with an approach to solve the problem
  - Group brainstorm about different approaches

SAMOSAS

- General framework for good meetings
- Start and stop meeting promptly
- Agenda created in advance; no agenda, no meeting
- Minutes recorded, so everyone can recall results
- One speaker at a time; no interrupting talker
- Send material in advance, since reading is faster
- Action items at end of meeting, so know what each should do as a result of the meeting
- Set the date and time of the next meeting

Quantitative Metrics

- Study many projects to record averages, set baseline for new projects, compare this one
- Code size (KLOC)
- Effort (months)
- Milestones fulfilled
- Test cases done
- Defect discovery / repair rate per month
- Cannot replace reviews