Byte-Sized

Computer Science for Data People

Part 3: Collaboration

Topics We'll Cover

1

Clean Code

Writing performant code that others will be excited to reuse

2

System Design

Building systems and products that scale

3

Collaboration

Working productively with other people

Principles of Collaboration

Version Control  Sharing, reviewing, and improving code on a shared platform

Automated Testing Making life easier for your team and future self through rigorous testing

Working Habits  Communication and organizational best practices for working with engineers

Deep-Dive: Version Control

Big Ideas

- **Always use a version control system for your code, data, and configs**
- **Don't keep multiple versions of the same code; use pull requests instead**
- **Keep your repo fresh with descriptive commits and good branch hygiene**

Writing new code in a copied script, then dealing with versioning issues when you want to incorporate it into the master

Handling versioning issues collaboratively and automatically on GitHub

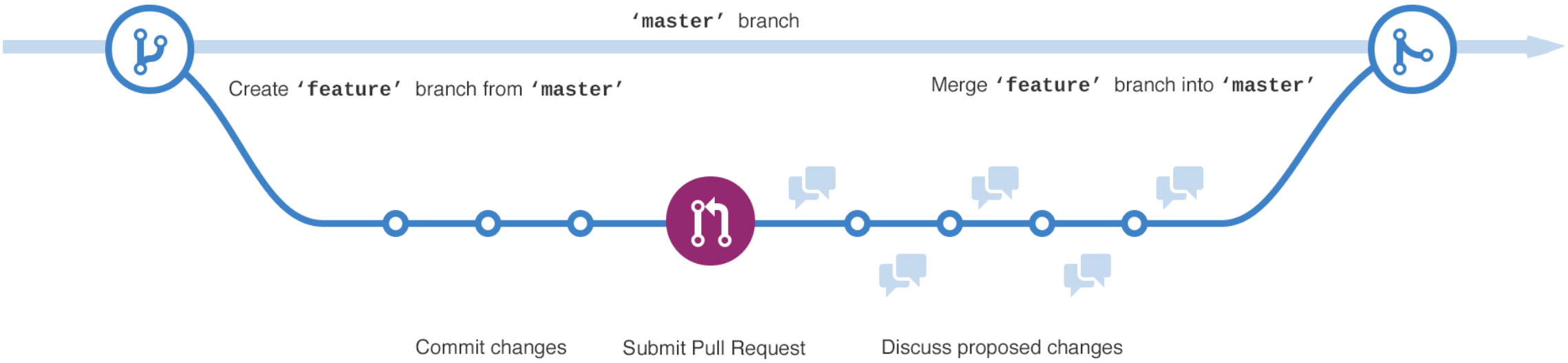
Benefits

- Collaborate on the same codebase without fear of versioning issues
- Easily view historical changes and reverse bad commits
- Provides an effective way to review and discuss' changes as a team

Related CS Concepts

- Git
- Reversibility
- Metadata

Collaborating on GitHub



Source: GitHub Guides

Deep-Dive: Automated Testing

Big Ideas

- **No one is considered finished until tests are written and passed**
- **Set goals around test performance and periodically review results**
- **Automatically run tests when pushing to GitHub**

Fumbling from one broken function to the next the night before a big meeting

Running automated tests as you push changes so you don't have to worry about undiscovered landmines

Benefits

- Save time and frustration by catching issues early
- Less 'what if's' means quicker, cheaper deployments
- Sturdy foundations enables everyone to innovate faster

Related CS Concepts

- Test-Driven Development (TDD)
- Deployment Frequency / Change Fail Rate / Time to Restore
- Continuous Integration / Continuous Delivery

Deep-Dive: Working Habits

Big Ideas

- **Phrase requirements in terms of stories about real people**
- **Be protective of development time**
- **Make code reviews and post mortems a team priority**

Trying to get code to run despite pings, emails, calls, and meetings

Minimizing PMO and aligning on 'ways of working' upfront so that everyone can be more efficient

Benefits

- Developers understand the who, what, and why behind their asks
- Code reviews help the entire team learn and grow
- Post mortems act as forcing functions to share knowledge and preempt future issues

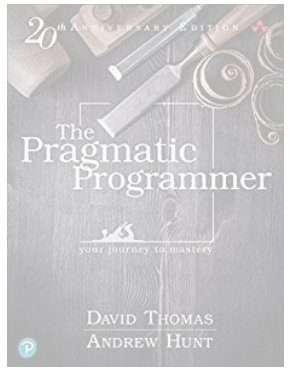
Related CS Concepts

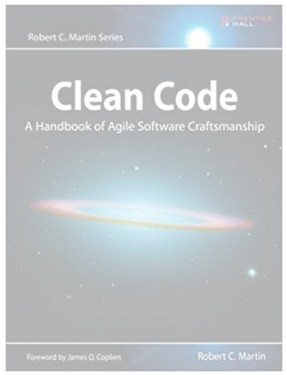
- Agile
- Maker's vs. Manager's Schedule
- DevOps

Book Recommendations

1

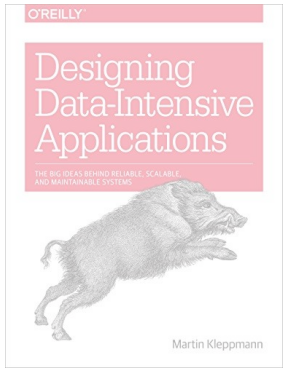
Clean Code

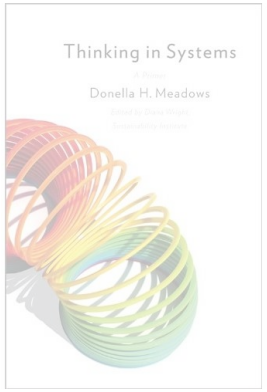




2

System Design





3

Collaboration