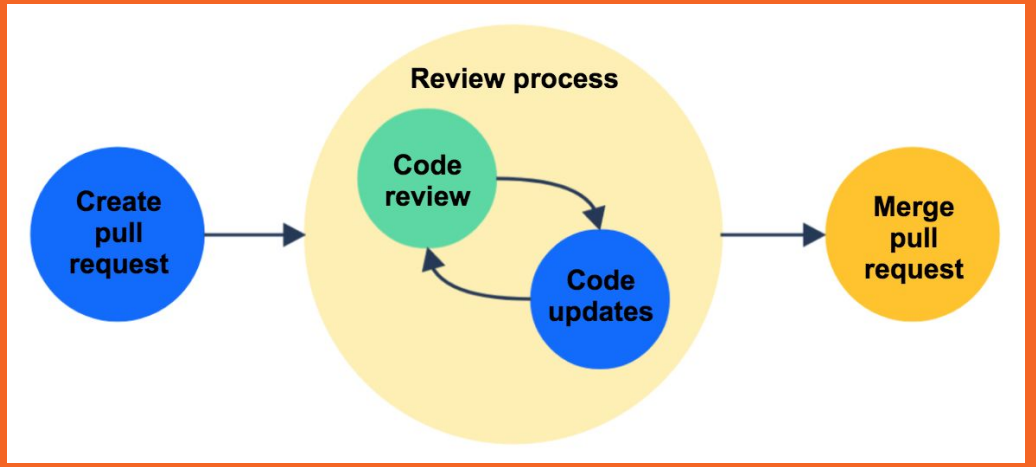
How to Conduct PR Reviews



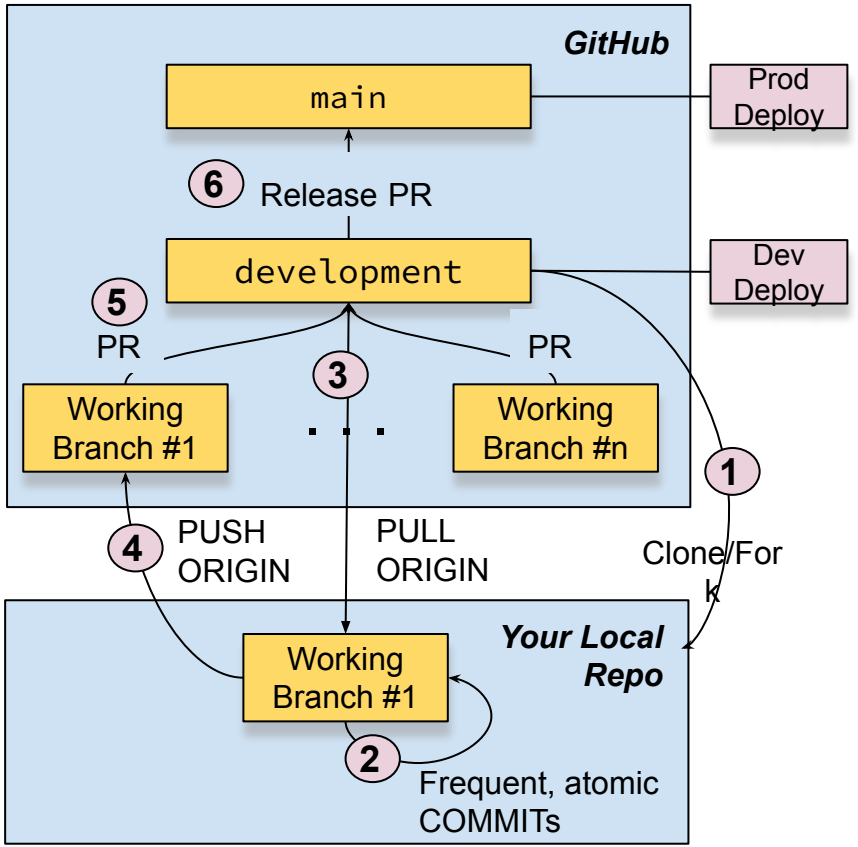
Jim Medlock - September 2023

Our Agenda

- What is a PR?
- Why does a PR need to be reviewed?
- What does the Review Process look like?
- Who should review a PR?
- What should a reviewer look for?
- What are the Best practices to follow?

What is a PR?

- Shorthand for “pull request”
- When a contributor/developer completes a development task and wants to merge the new code into the project branch production is deployed from
- Moves the change through a set of hierarchical branches before landing in production



What is a PR?

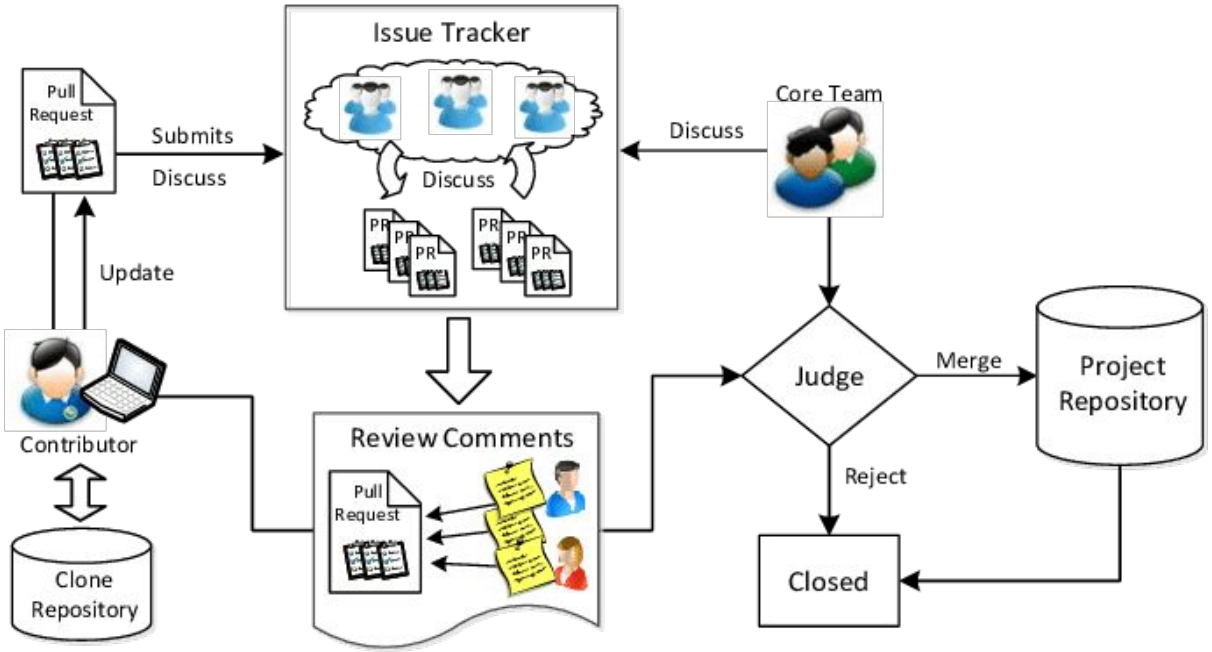
1. At the start of the project
   - Clone the GitHub repo to your local PC
   - Create a new working branch & pull `development` branch to it
2. Develop & unit test your feature
3. Pull `development` branch to get changes made since you started,
4. Push working branch to GitHub
5. Create **Pull Request** to merge into `development`
6. Create **Pull Request** to merge into `main` `after approval`

Why Does a PR Need to be Reviewed?

- Make sure standards are followed
- Test the code to ensure it works
  - Changed code + unaltered code
- Ensure the code is performant
- Make sure the code is readable & maintainable
- Stop technical debt before it starts
- Opportunity to mentor & share knowledge
- Opportunity to discuss the issue to find better solutions to the problem

What does the Review Process look like?

- Collaborators describe proposed changes
- Reviewers comment & question
- Modifications may be requested before the pull request is merged
- Only approved changes are merged into Production
- Unsolicited changes may be rejected



Who Should Review a PR?

- Voyage
  - Any teammate
  - All teammates
  - Rotate responsibility in each Sprint
- Open Source Project
  - Core team member (ie. a Maintainer)
- Commercial Enterprise
  - Defined by management/tech leads

Regardless of your strategy reviews must be approved by someone other than the author

What Should a Reviewer Look For?

- Design
  - Do the changes integrate with existing code in a way that makes sense?
  - Do the changes interact with other parts of the app in a logical way?
  - Do the changes make it easy to extend/add future functionality?
  - Is this the right time to add these changes?
- Functionality
  - Does this code do what it's intended to do?
  - Is the change good for both the users & developers?
- Tests
  - Has the code been unit tested?
  - Are tests correct, sensible and useful?

What Should a Reviewer Look For?

- Naming
  - Have good names been used for variables, functions, classes, etc.?
  - Do they communicate what the item is or does, without being to long?
- Comments
  - Are comments clear and useful?
  - Do they explain "why" or "what" instead of "how"?
  - Could comments be eliminated by clearer code?
- Consistency
  - Does the code follow existing agreements about style, naming, files organization, etc.?

What Should a Reviewer Look For?

- Documentation
  - If the code changes how developers build, test, or release code has the readme.md been updated?
  - If the code changes how users interact with the app has other documentation been updated?
  - Are messages issued by the app clear and are they actionable, or will they result in another call to the Help Desk?
  - Have debugging console messages been removed

What are the Best Practices to follow?

- Contributors & Reviewers:
  - Treat the review as a learning experience
  - Treat each other with respect
  - Don’t judge, blame, or be arrogant or sarcastic
  - Do ask questions, explain your point of view, suggest, and assume everyone is well-intended
  - Work together to make the PR something better than either could do individually
  - Discuss in real time if text conversations become lengthy

What are the Best Practices to follow?

- Contributors:
  - Question your own code before requesting a review
  - Add explanatory comments for complex changes
  - Favor small PR’s over larger ones
  - Link backlog items in review comments
  - Ask yourself if this changes leaves the app better than you found it
  - Answer all comments
  - Summarize the review & actions taken when it’s approved

What are the Best Practices to follow?

- Reviewers:
  - Don’t rush a review. Give it the time it deserves
  - Remember the code is owned by the entire Dev team. There is no personal ownership of any one part
  - Separate your personal preferences from best practices. Don’t assume your way is the only way
  - Treat the review as a way to mentor + learn
  - Identify techniques the entire team should follow
  - Make your feedback/comments actionable
  - Keep in mind that “Perfection is the enemy of progress”

A checklist is your friend

- For example - Chingu Solo Project Evaluation checklist