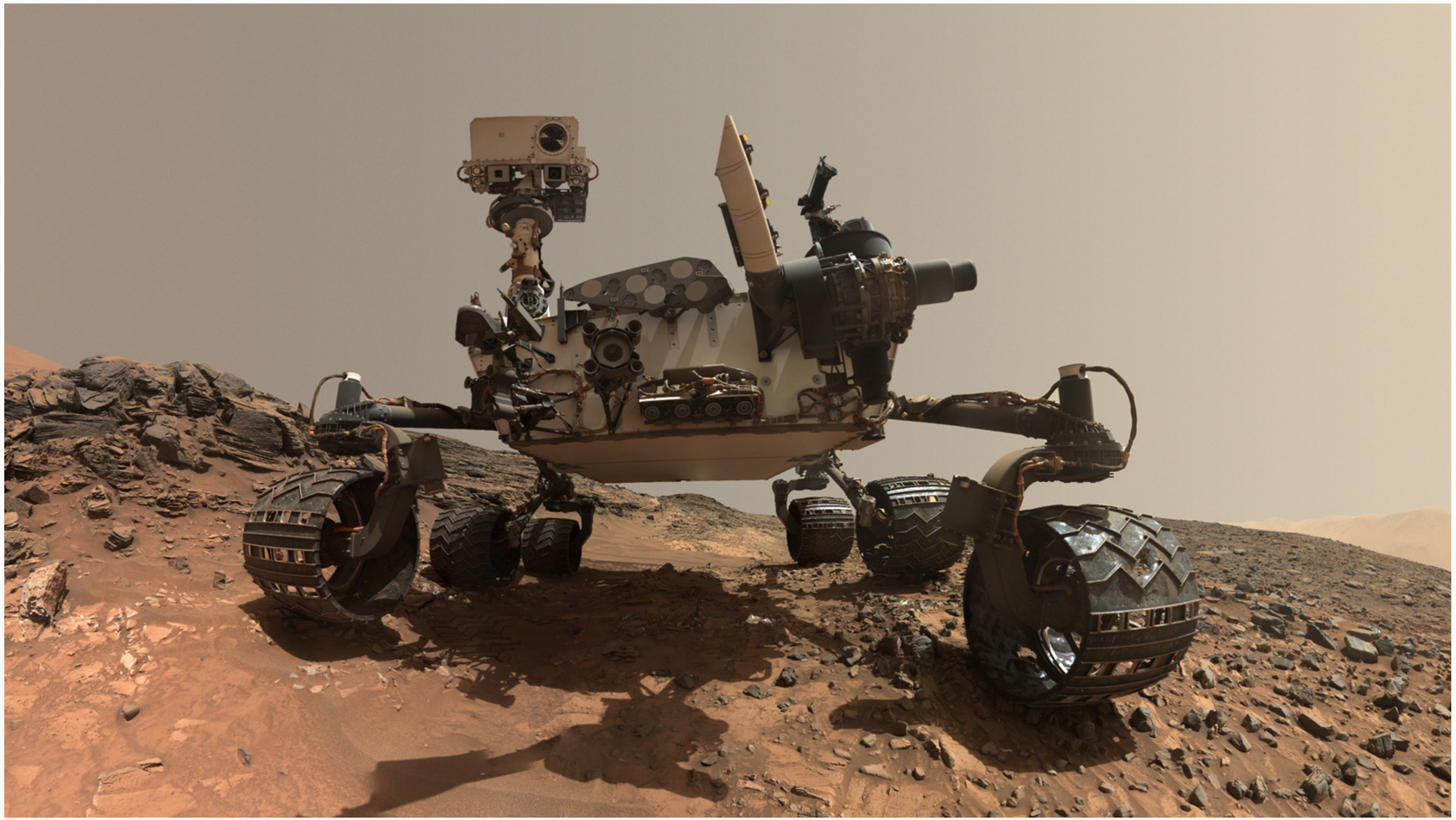
EPFL

CS-311: Collaborative Software Development

Prof. George Candea
School of Computer & Communication Sciences

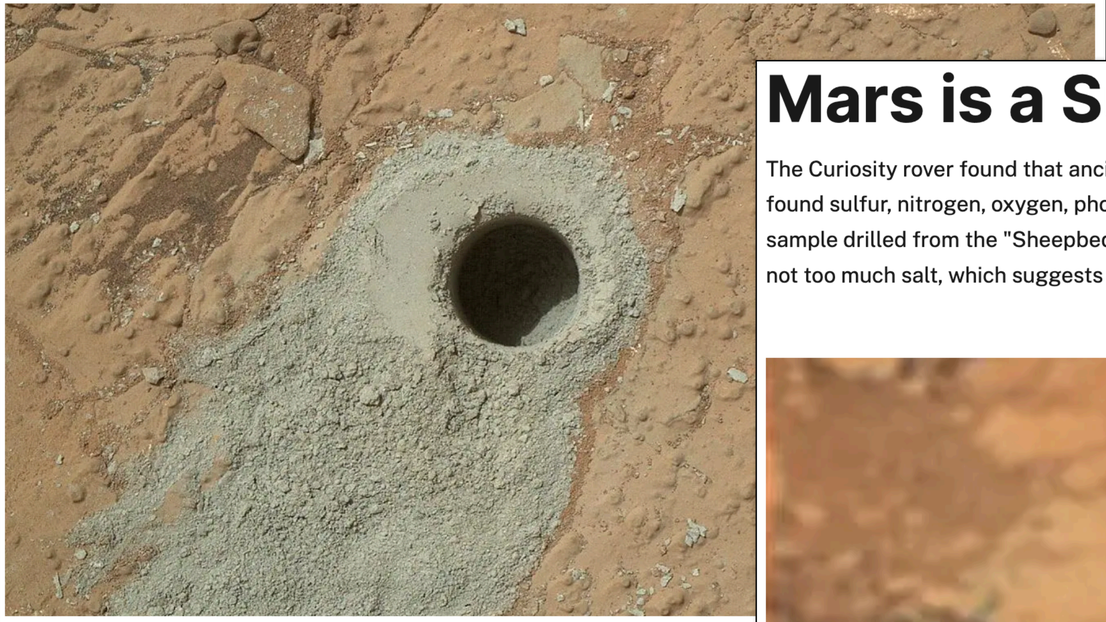


https://www.gstatic.com/culturalinstitute/searchar/assets/curiosity/desktop_light.mp4


NASA
JPL

Organic Carbon Found in Mars Rocks

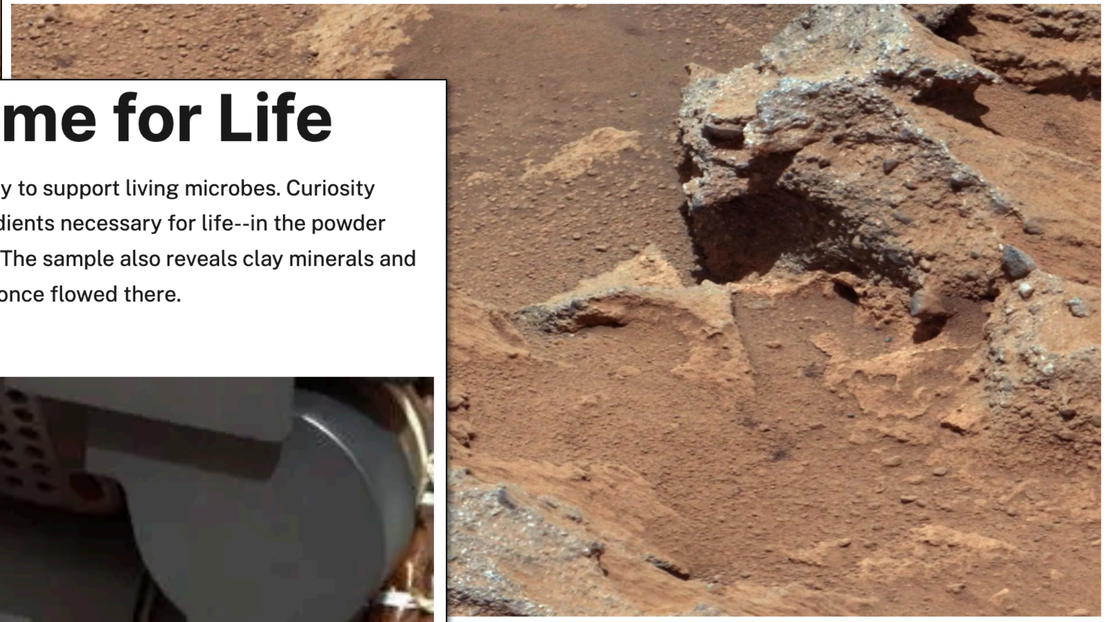
Organic molecules are the building blocks of life, and they were discovered on Mars after a long search by the Sample Analysis at Mars (SAM) instrument in several samples drilled from Mount Sharp and the surrounding plains. The finding doesn't necessarily mean there is past or present life on Mars, but it shows that raw ingredients existed for life to get started there at one time. It also means that ancient organic materials can be preserved for us to recognize and study today.



Cumberland Target Drilled by Curiosity

Curiosity Finds Evidence of Persistent Liquid Water in the Past

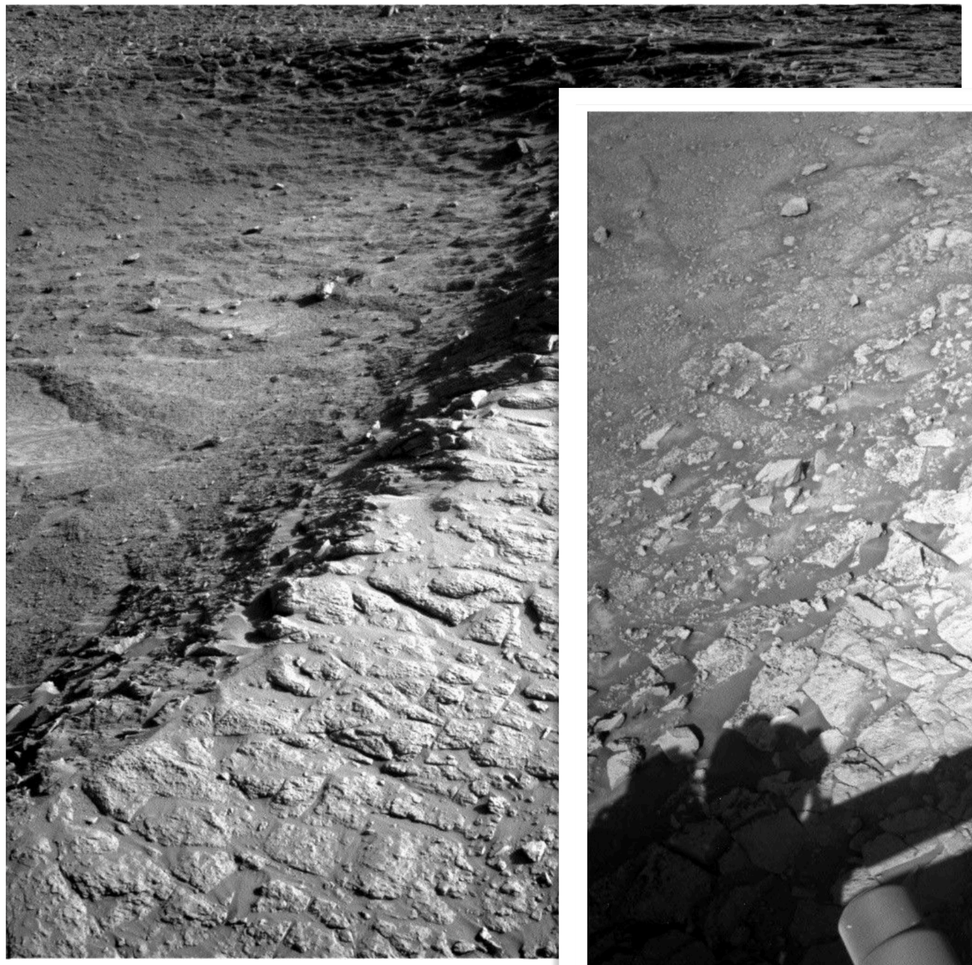
Just after landing, Curiosity found smooth, rounded pebbles that likely rolled downstream for at least a few miles in a river that was ankle- to hip-deep. When Curiosity reached Mount Sharp, the team found that over 1,000 vertical feet of rock formed originally as mud at the bottom of a series of shallow lakes. Rivers and lakes persisted in Gale crater for perhaps a million years or longer.



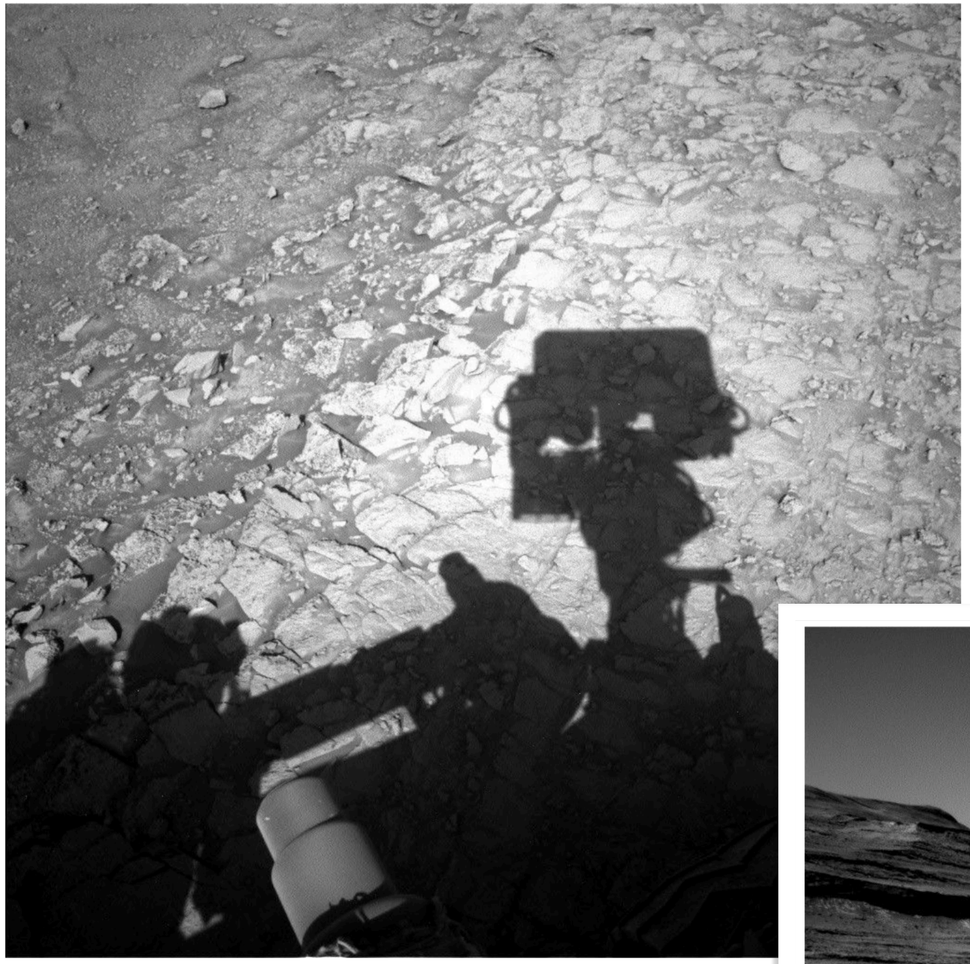
Mars is a Suitable Home for Life

The Curiosity rover found that ancient Mars had the right chemistry to support living microbes. Curiosity found sulfur, nitrogen, oxygen, phosphorus and carbon--key ingredients necessary for life--in the powder sample drilled from the "Sheepbed" mudstone in Yellowknife Bay. The sample also reveals clay minerals and not too much salt, which suggests fresh, possibly drinkable water once flowed there.

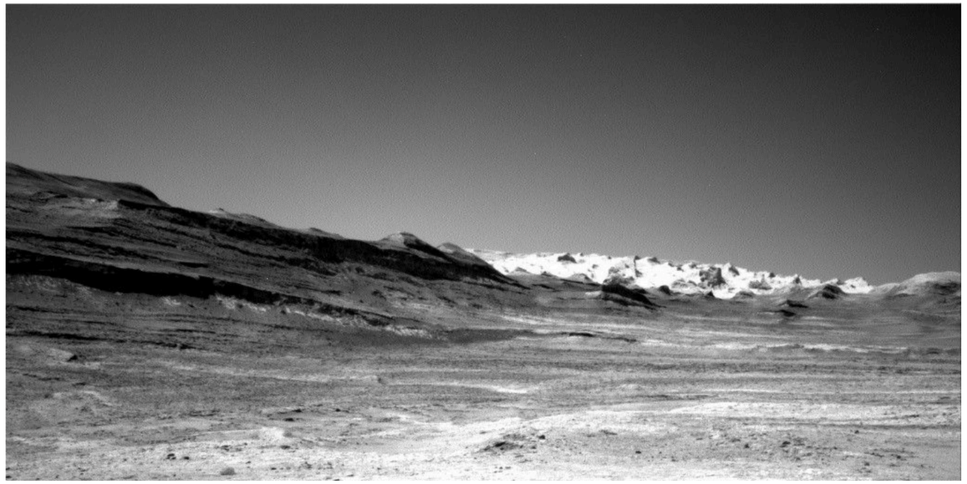
First Curiosity Drilling Sample in the Scoop



NASA's Mars rover Curiosity acquired this image on Aug. 21, 2025, looking
near the edge of, looking down into the "Thumb" region that mission planne
captured this image using its Left Navigation Camera on Sol 4636, or Mart
mission, at 16:09:13 UTC.



NASA's Mars rover Curiosity acquired this image using its Left Navigation Camera, showing the transitio
bedrock (right) to more nodular bedrock (bottom left to top middle) on the edge of a shallow hollow (top
masthead shadow is also visible, captured this image on Sept. 5, 2025 — Sol 4650, or Martian day 4,650
Laboratory mission — at 00:22:34 UTC.



NASA's Mars rover Curiosity acquired this image, showing the boxwork terrain in the foreground and the bright wind-sculpted material in the distance, on Sept. 12, 2025. Curiosity used its Right Navigation Camera on Sol 4657, or Martian day 4,657 of the Mars Science Laboratory mission, at 00:50:58 UTC.

Outline

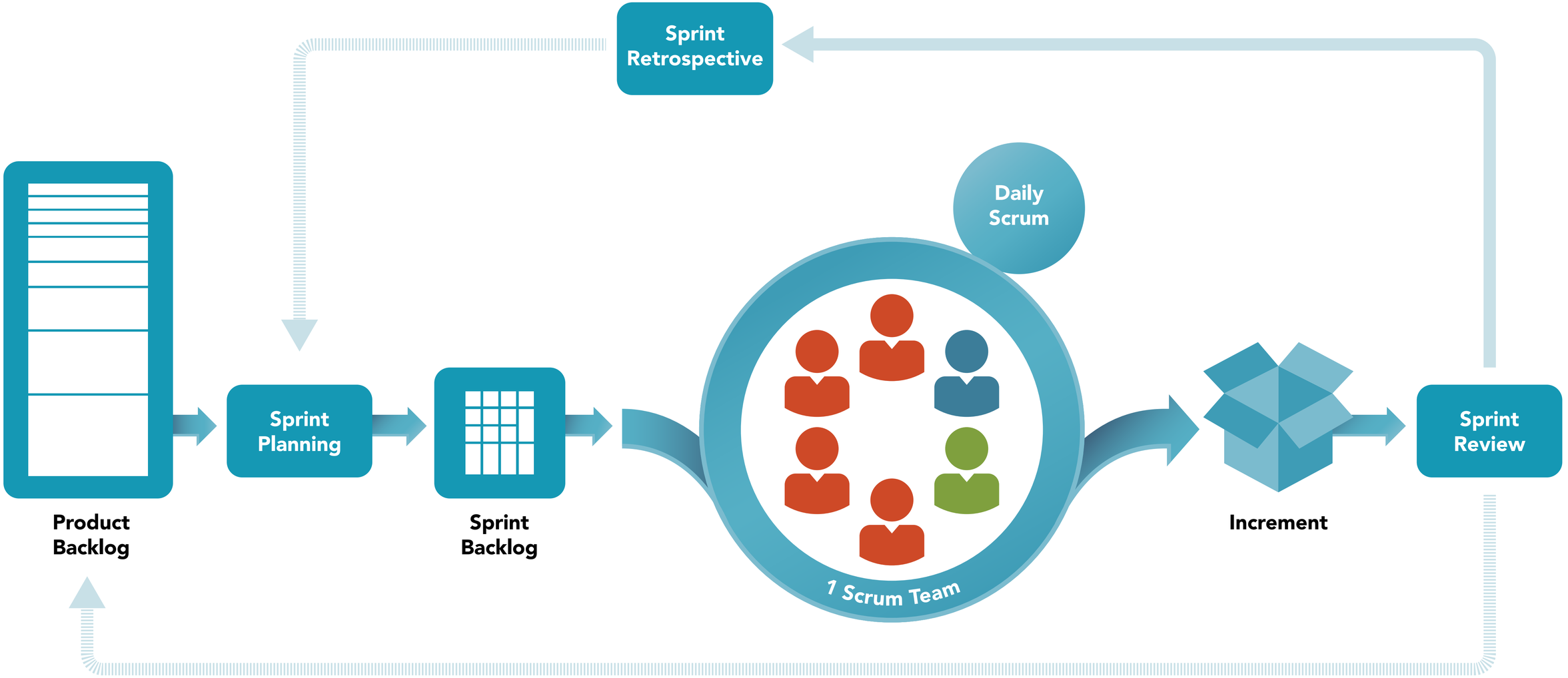
- Development process: Scrum
- Collaboration workflows
- CI / CD
- Writing good commit messages — *online*
- Coding standards — *online*

Development Process: Scrum

Scrum Method

- Basic structure = ***Sprint***
  - *1-4 weeks (rigidly fixed length)*
  - *one Sprint after another*
  - *working product at the end of each sprint*
- Cross-functional development teams of 3-10 people
- Meet daily

Scrum Workflow



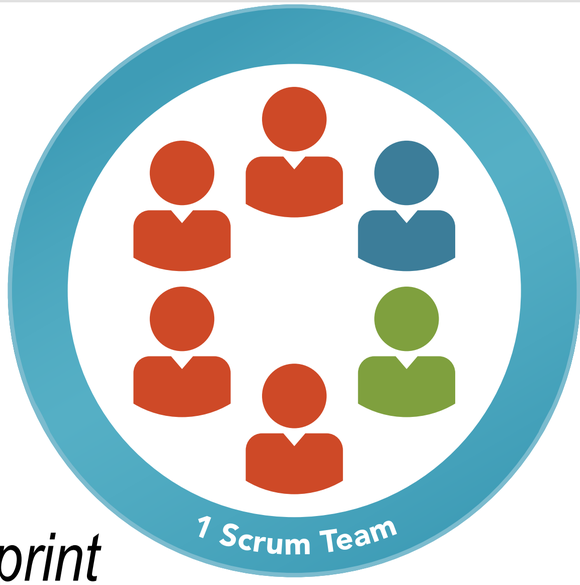
https://www.scrum.org/resources/scrum-framework-poster

- Players
- Workflow

- **Players**
- **Workflow**

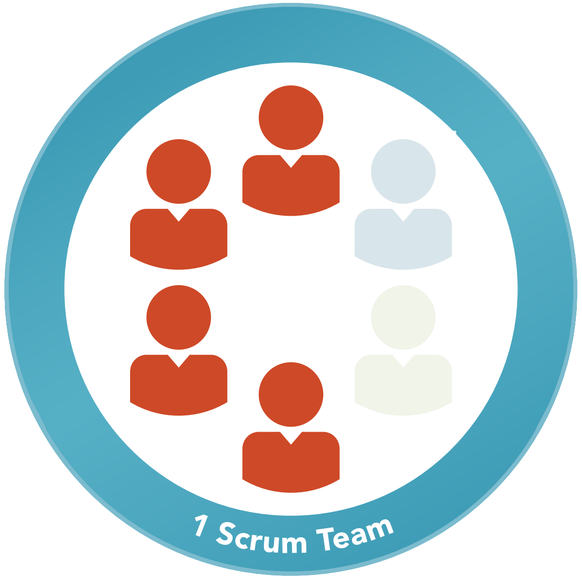
Scrum Team

- N Developers +
  1 Product Owner (PO) +
  1 Scrum Master (SM)
- Cohesive unit of ≤10 professionals
  - *small enough to be nimble, big enough to do significant work in one Sprint*
  - *no hierarchy, no sub-teams, self-managing, autonomous → accountable*
  - *for big products, can have multiple Scrum teams with same Goal, Backlog, PO*
- Scrum team does "everything"
  - *dev code, test, maintain, research, stakeholder collaboration, etc.*
- Accountable for creating a valuable, useful Increment every Sprint

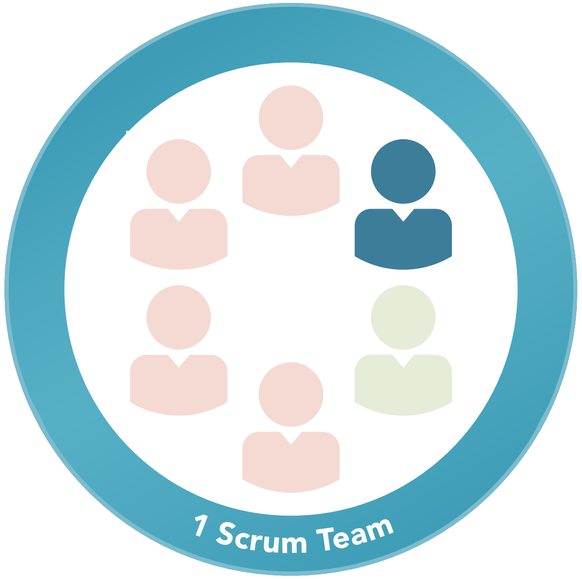


Scrum Team: Developers

- Write the code, build the product, maintain, devops, etc.
- Cross-functional
  - *developers, UX designers, testers, ...*
  - *≤ 8 developers on a Scrum team*
- Accountable for
  - *creating the Sprint Backlog and adapting the plan during the Sprint*
  - *keep focus on Sprint Goal, produce Increment*
  - *hold each other to high professional standards*



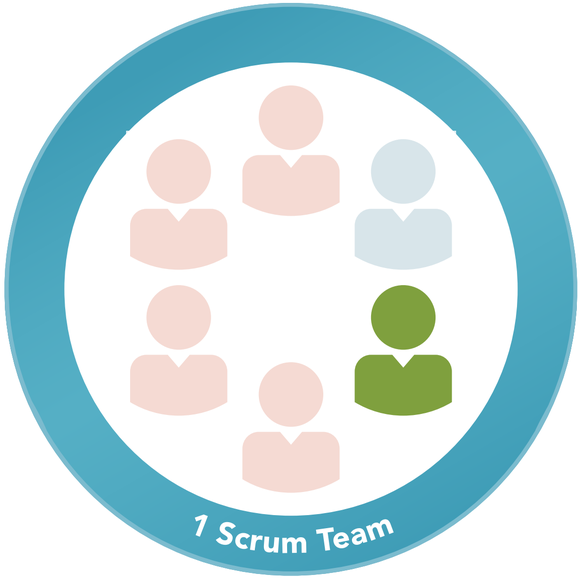
Scrum Team: Product Owner



- Represents stakeholders' interests
  - *sort of like a traditional product manager*
  - *translates stakeholders' needs into a priority list*
  - *thinks about user stories, gathers requirements, etc.*
- PO = one single person
- Accountable for
  - *maximizing business value of Team's product*
  - *managing the Product Backlog:*
    creating PB items, communicating them, prioritizing them, etc.
  - *accountable ⇏ do all the work*

Scrum Team: Scrum Master

- Accountable for team's success and effectiveness
  - *a facilitator and enabler, not a manager*
- Help developers
  - *coach on how to self-manage and cross-function*
  - *focus on Sprint Goal + protect devs from outside interference*
  - *help remove impediments*
  - *make sure Sprint events take place, are productive, and properly timeboxed*
- Help PO
  - *manage PB, collaborate w/ stakeholders, plan product, ...*
- SM could be one of the developers with suitable training



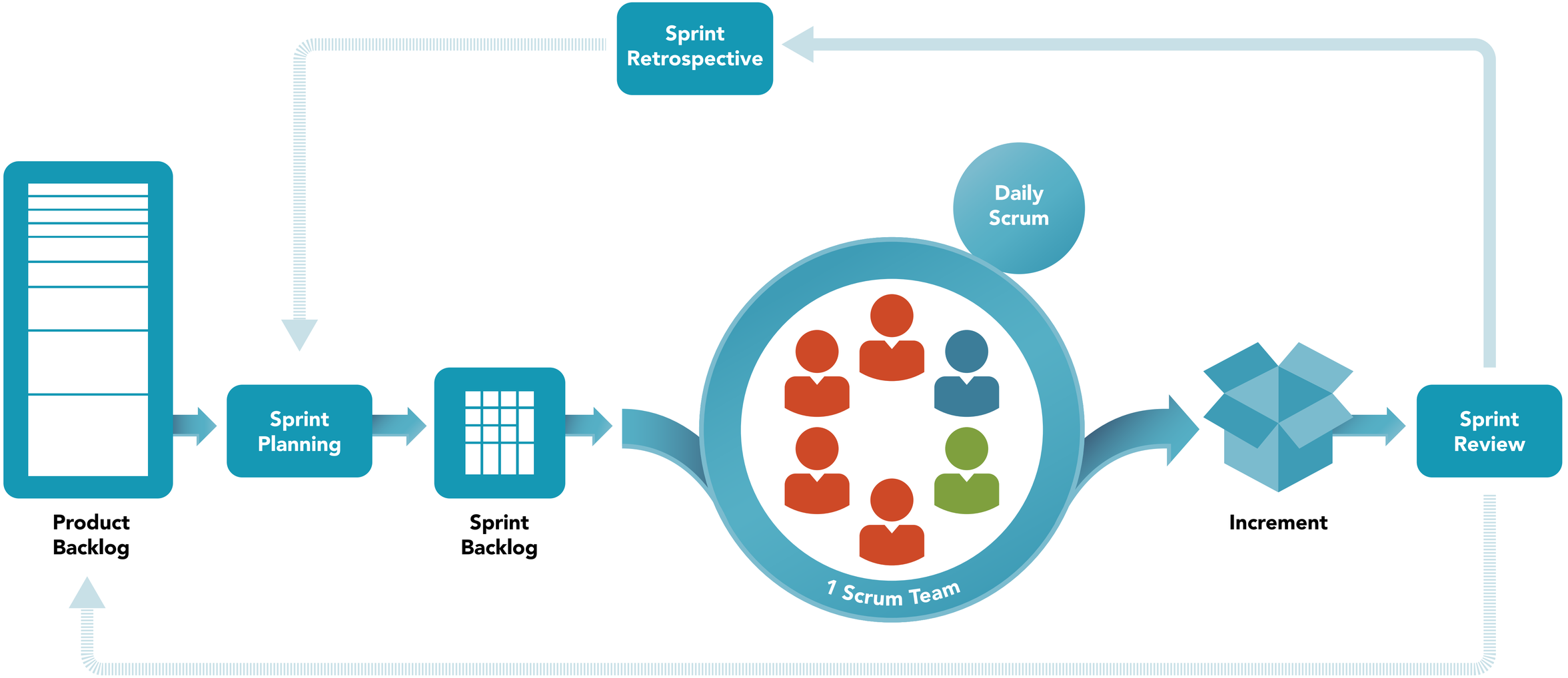
- **Players**
- **Workflow**

- Players
- **Workflow**

Sprint

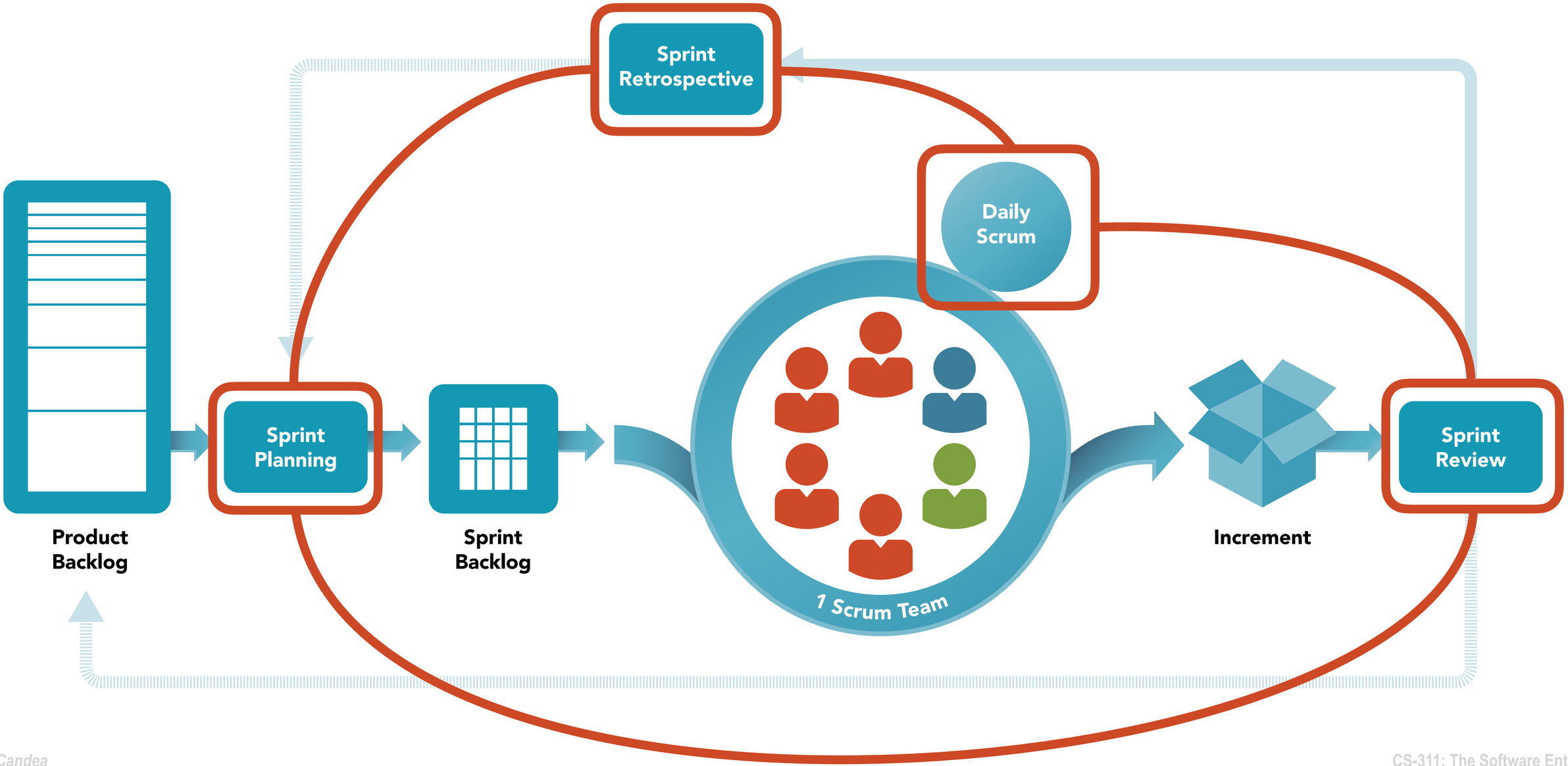
- Pick Sprint duration (1-4 weeks) and stick with it for the entire project
  - *think of each Sprint as a mini-project*
  - *helps team improve their estimates*
- Sprint is never extended
  - *if goals not meet, team must own up to it*
  - *takes some experience to determine how long tasks will take*
    → over time, team gets better at it

Scrum Events



https://www.scrum.org/resources/scrum-framework-poster

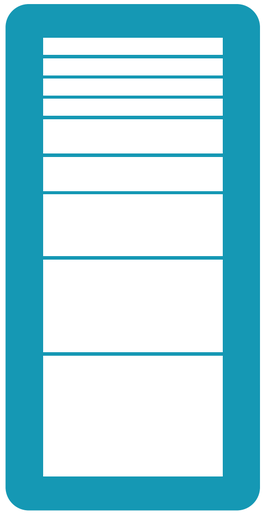
Scrum Events



https://www.scrum.org/resources/scrum-framework-poster

Product Backlog (PB)

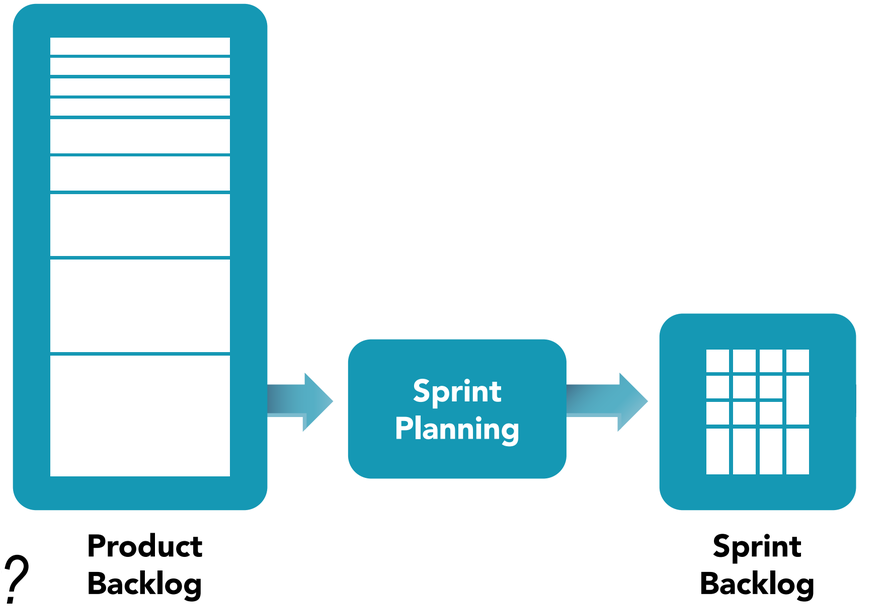
- Product Goal = product vision (by PO)
  - *how will it provide value and to whom*
- Rest of PB:
  - *list of user stories, research tasks, bug fixes, etc.*
  - *prioritized by value to stakeholders*
- Evolves over time (change is a given!)
- Team provides time estimates in person-hours/days
  - *PO uses time estimates as input in prioritization of PB*
- All changes to the PB done exclusively by the PO



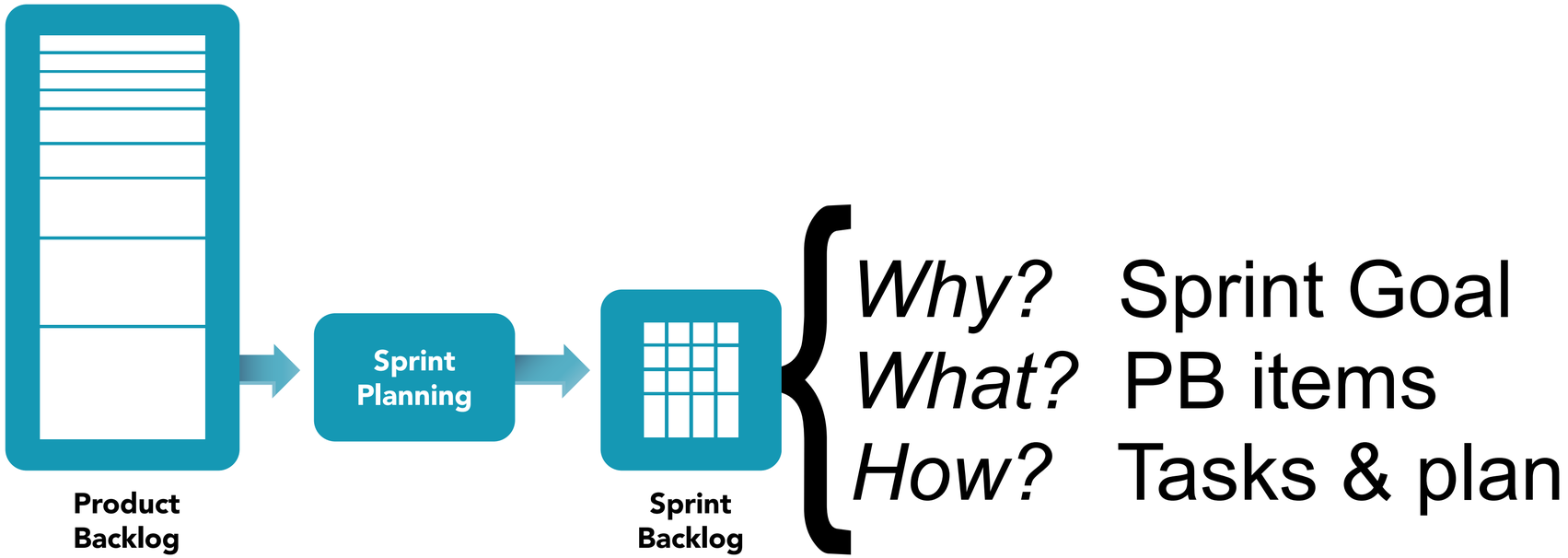
Product Backlog

Scrum Events: Sprint Planning

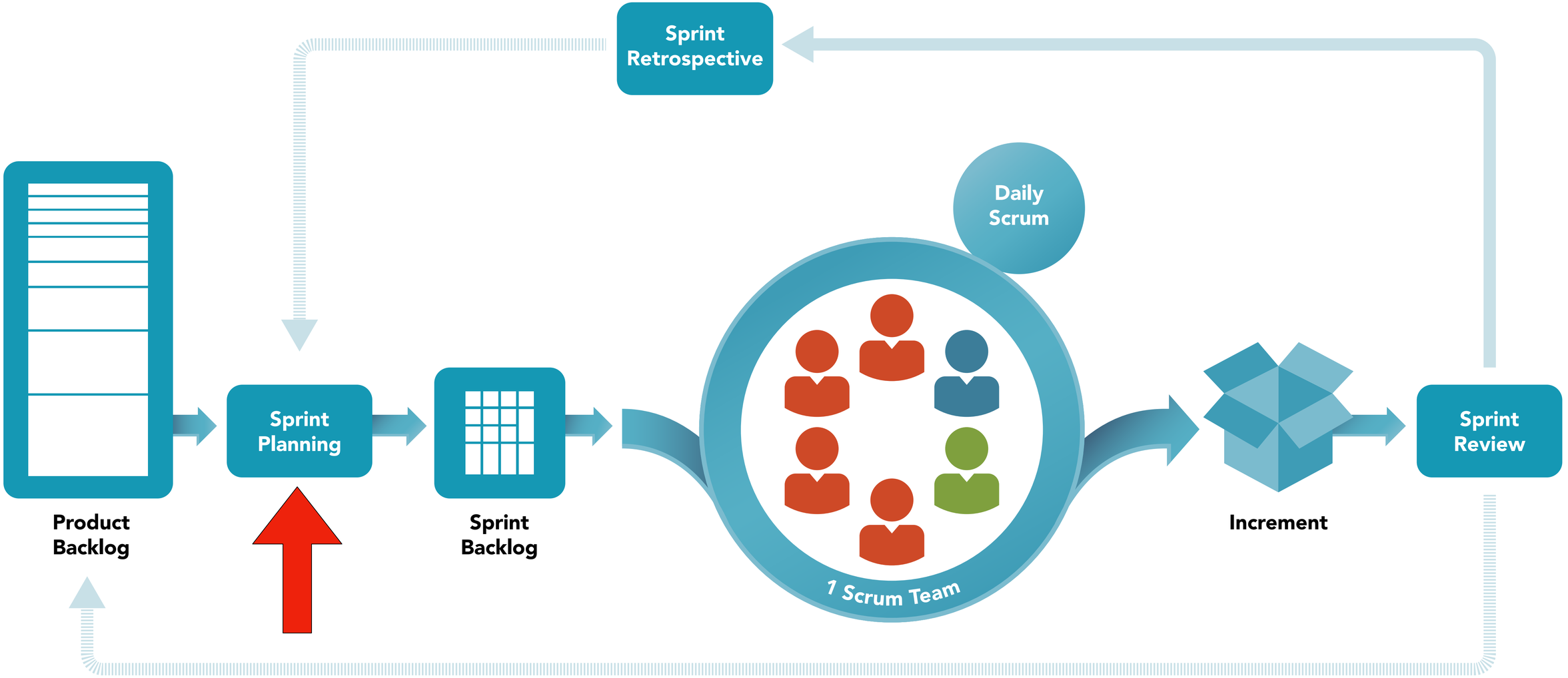
- Attendees: Scrum team (devs + PO + SM)
  - *can invite others to provide advice (e.g., tech experts)*
- Three topics
  - *Why is this Sprint valuable? → leads to Sprint Goal*
  - *Which items at the top of the PB can be done this Sprint?*
  - *How to create an Increment: turn each claimed PB item into tasks*
- Gives team insight into the thinking of the customer



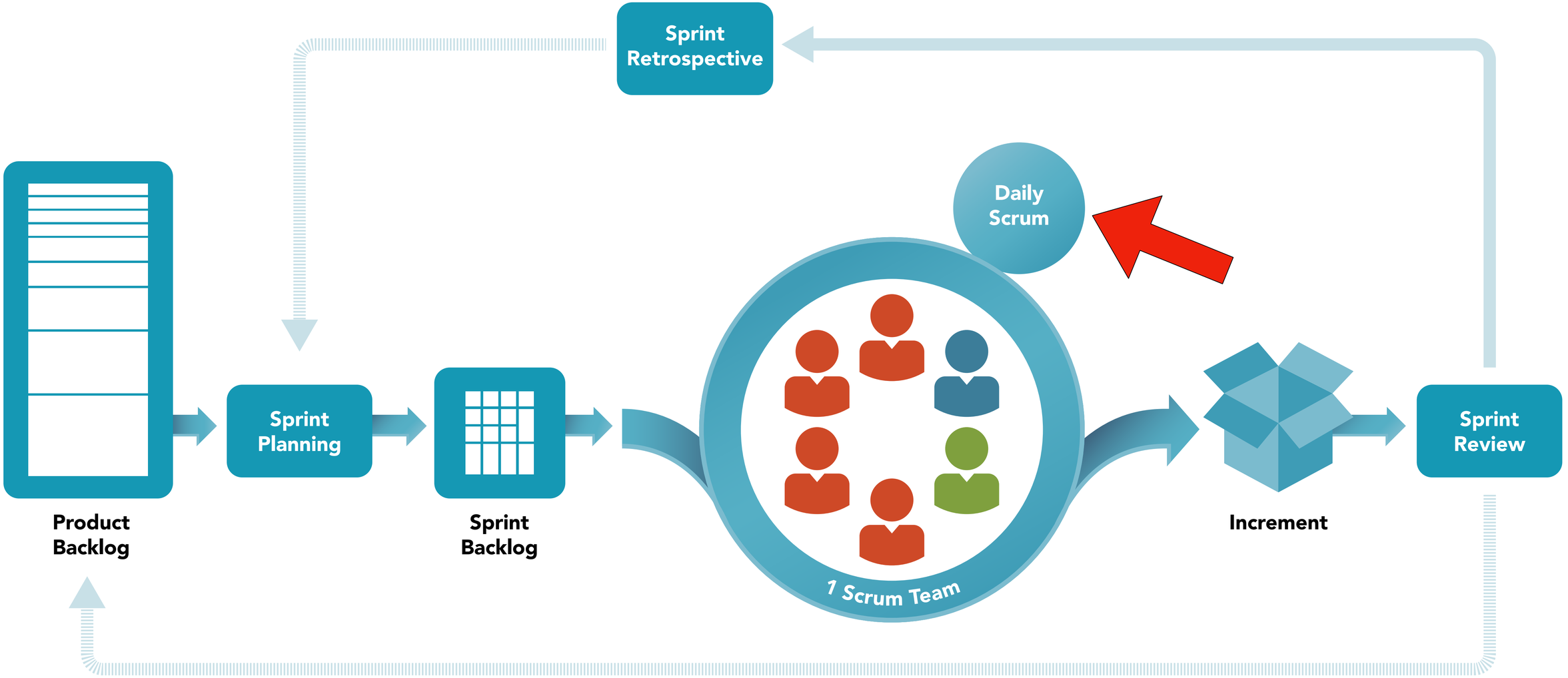
The Sprint Backlog



Scrum Events

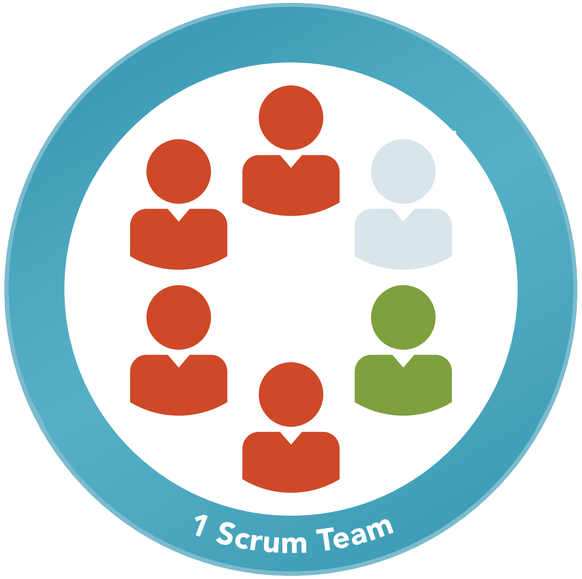


Scrum Events

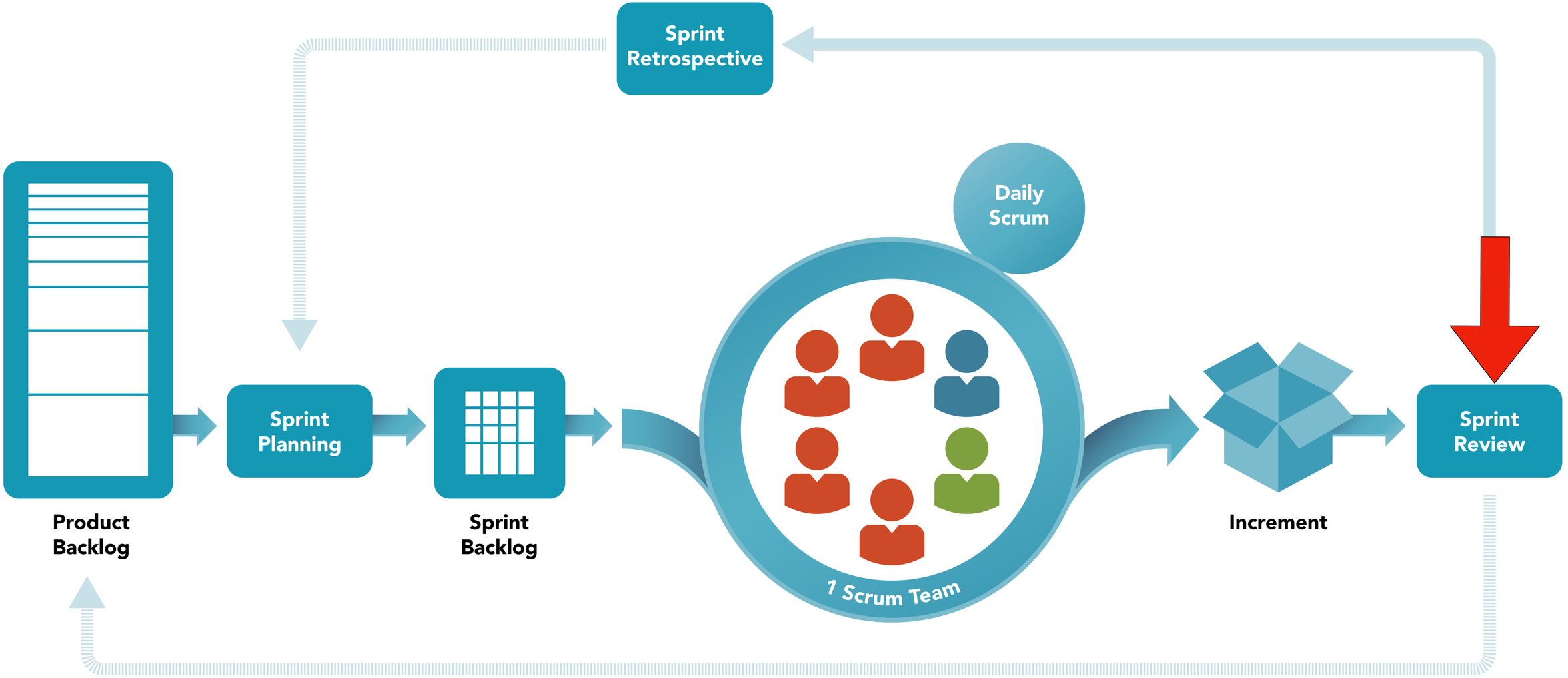


Scrum Events: Daily Scrum

- Attendees: Developers + Scrum Master
  - *at same time, same place every day of the Sprint*
  - *meeting lasts <= 15 minutes, everyone stands*
- Each developer reports 3 items (no discussion!):
  - *What (s)he has done since last meeting*
  - *What (s)he will do until next meeting*
  - *Any blocks or impediments*
- After meeting
  - *Scrum Master resolves impediments and updates progress metrics*



Scrum Events



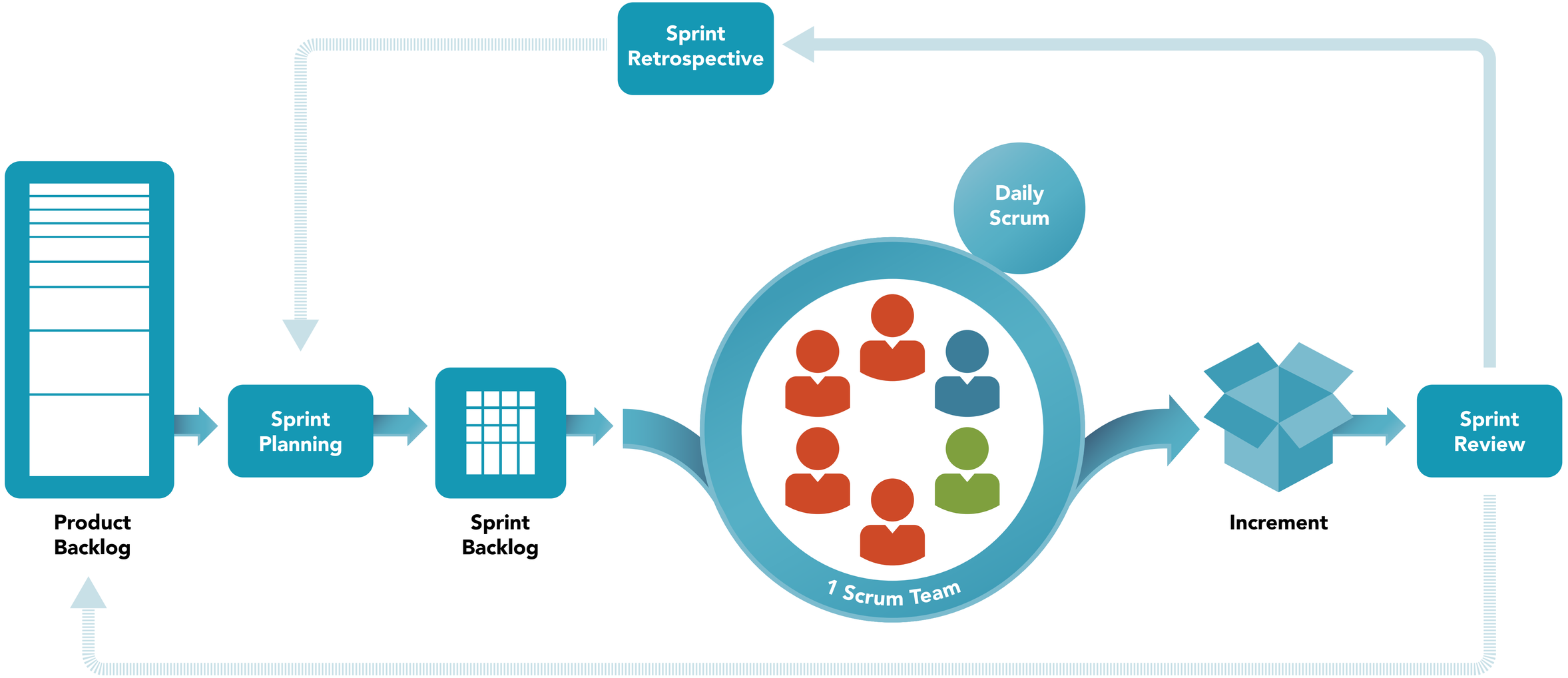
Scrum Events: Sprint Review

- Attendees: Scrum Team + Stakeholders
- Demo the Sprint Increment
  - *stakeholders inspect the outcome of the Sprint*
  - *determine future adaptations*

Scrum Events: Sprint Retrospective

- Attendees: Scrum Team
  - *may be facilitated by a neutral outside (e.g., other SM)*
- Identify and plan ways to increase quality and effectiveness
  - *what went wrong? what worked well?*
  - *how can we improve?*
- PO updates PB based on Review & Retrospective

Scrum Workflow



https://www.scrum.org/resources/scrum-framework-poster

https://scrumguides.org/scrum-guide.html

Resources

- https://agilepainrelief.com/blog/scrum-by-example.html
- Wrong way of doing things...
  - *https://www.youtube.com/watch?v=I1yWusiaLCM (Requirements & Specifications)*
  - *https://www.youtube.com/watch?v=oLmDe8pAc6I (Stand-Ups)*
  - *https://www.youtube.com/watch?v=FJezcyKno5k (Retrospectives)*
- *User stories for SwEnt*
  - *https://www.youtube.com/watch?v=apOvF9NVguA (Creating User Stories)*
- *Tutorial (with Jira, but ignore that) — good preparation for SwEnt*
  - *Part 1: https://www.atlassian.com/agile/tutorials/how-to-do-scrum-with-jira-software*
  - *Part 2: https://www.atlassian.com/agile/tutorials/how-to-do-advanced-scrum-practices-with-jira-software*

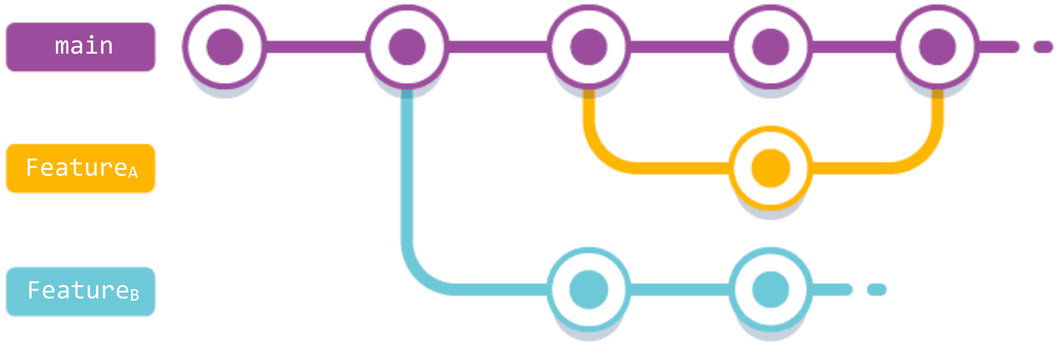
Outline

- Development process: Scrum
- Collaboration workflows
- Writing good commit messages
- Coding standards
- CI / CD

Collaboration Workflows

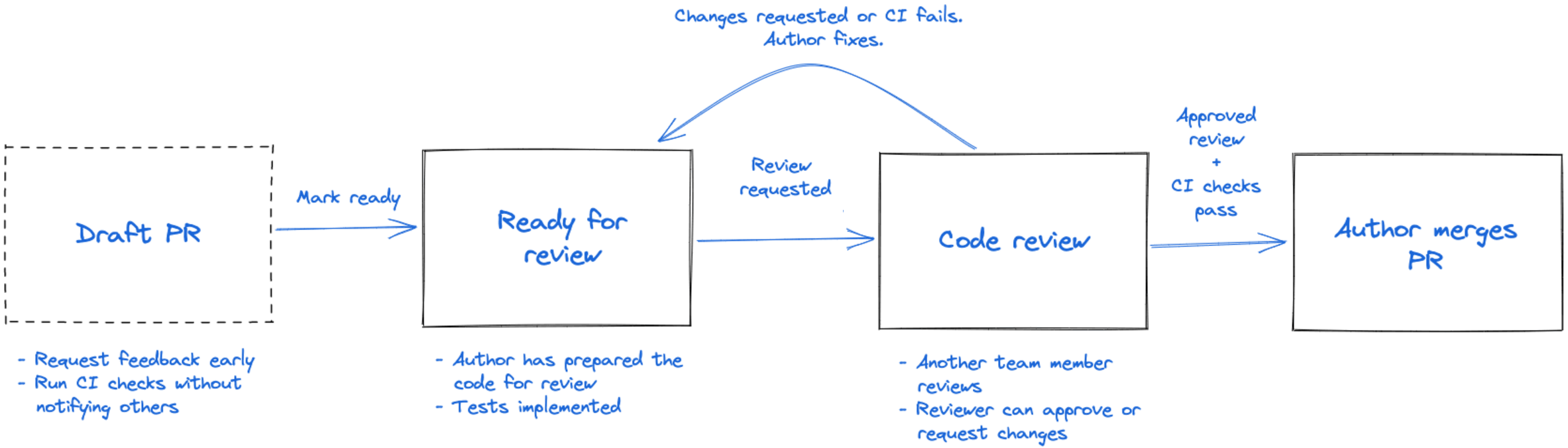
Basic "Feature Branch" Workflow

- Create new branch for each feature/bug
- Commit all related code to this branch
- Regular commits + branch sync-up
- Test → Code review
- Resolve merge conflicts
- Changes approved → merge branch to main
  - *fast-forward merge*
  - *merge commmit*
  - *rebase & merge*
  - *squash merge*
- Delete the feature branch



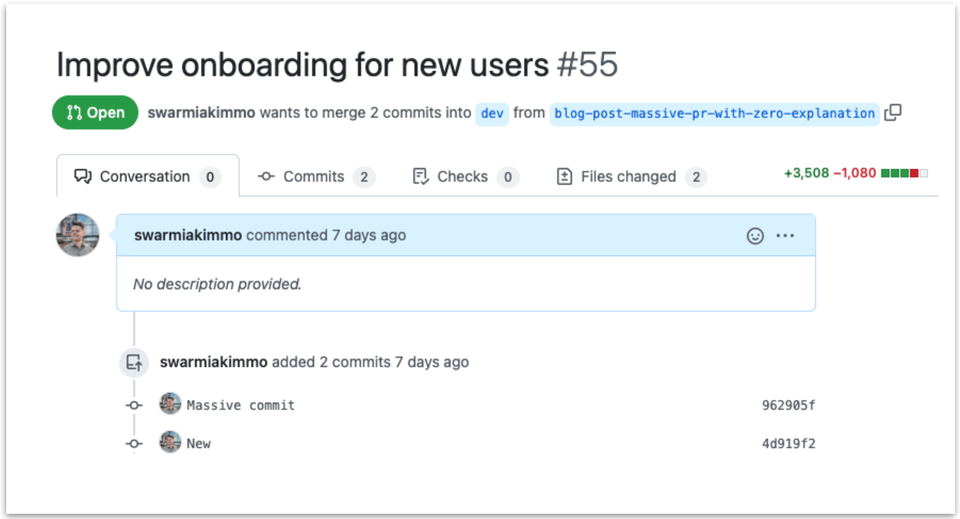
Workflow Overview

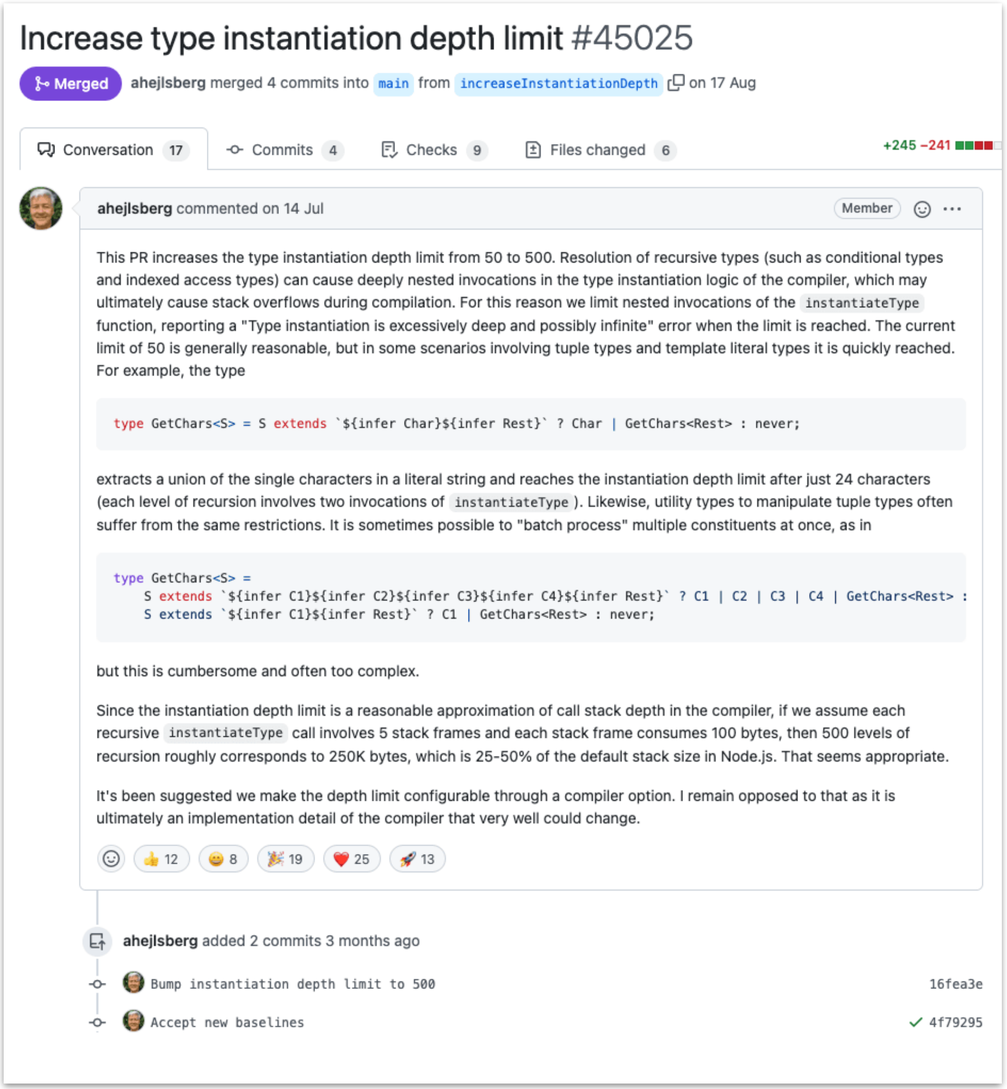
- "Fork & Pull" model vs. "Shared Repo" model



Working with PRs

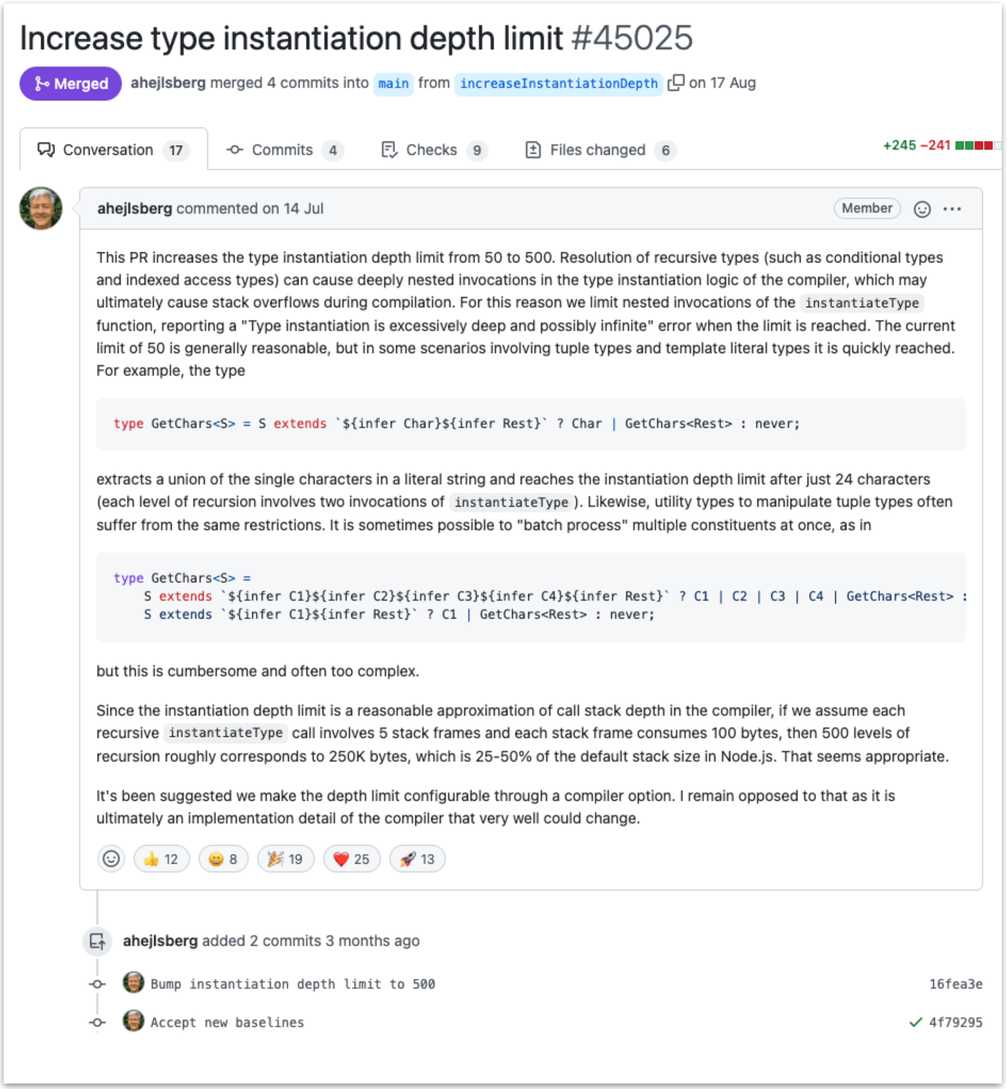
- Provide context for each PR





Working with PRs

- Provide context for each PR
- Link PR to corresponding issue (if any)



Increase type instantiation depth limit #45025

Merged ahejlsberg merged 4 commits into main from increaseInstantiationDepth on 17 Aug

Conversation 17 Commits 4 Checks 9 Files changed 6 +245 −241

ahejlsberg commented on 14 Jul — Member

This PR increases the type instantiation depth limit from 50 to 500. Resolution of recursive types (such as conditional types and indexed access types) can cause deeply nested invocations in the type instantiation logic of the compiler, which may ultimately cause stack overflows during compilation. For this reason we limit nested invocations of the `instantiateType` function, reporting a "Type instantiation is excessively deep and possibly infinite" error when the limit is reached. The current limit of 50 is generally reasonable, but in some scenarios involving tuple types and template literal types it is quickly reached. For example, the type

```
type GetChars<S> = S extends `${infer Char}${infer Rest}` ? Char | GetChars<Rest> : never;
```

extracts a union of the single characters in a literal string and reaches the instantiation depth limit after just 24 characters (each level of recursion involves two invocations of `instantiateType`). Likewise, utility types to manipulate tuple types often suffer from the same restrictions. It is sometimes possible to "batch process" multiple constituents at once, as in

```
type GetChars<S> =
    S extends `${infer C1}${infer C2}${infer C3}${infer C4}${infer Rest}` ? C1 | C2 | C3 | C4 | GetChars<Rest> :
    S extends `${infer C1}${infer Rest}` ? C1 | GetChars<Rest> : never;
```

but this is cumbersome and often too complex.

Since the instantiation depth limit is a reasonable approximation of call stack depth in the compiler, if we assume each recursive `instantiateType` call involves 5 stack frames and each stack frame consumes 100 bytes, then 500 levels of recursion roughly corresponds to 250K bytes, which is 25-50% of the default stack size in Node.js. That seems appropriate.

It's been suggested we make the depth limit configurable through a compiler option. I remain opposed to that as it is ultimately an implementation detail of the compiler that very well could change.

👍 12 😄 8 🎉 19 ❤️ 25 🚀 13

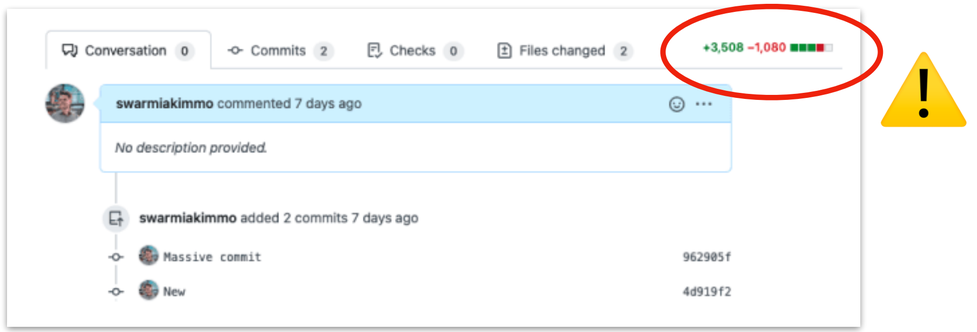
ahejlsberg added 2 commits 3 months ago

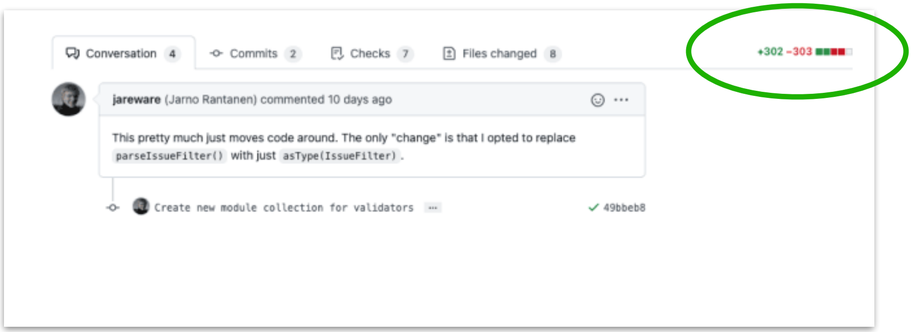
Bump instantiation depth limit to 500 16fea3e

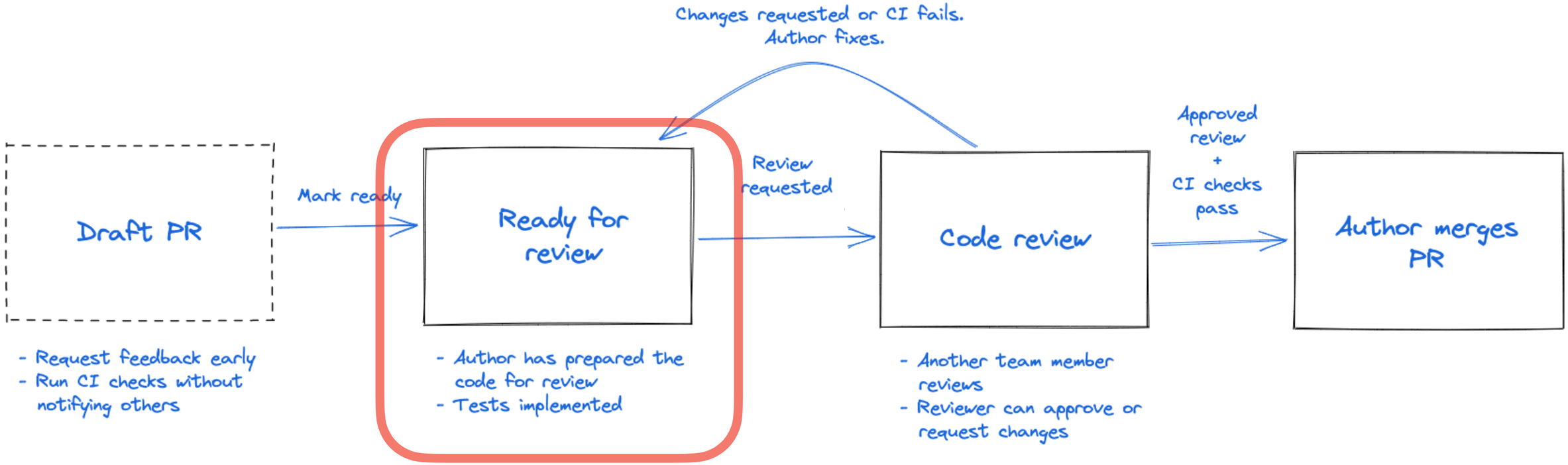
Accept new baselines ✓ 4f79295

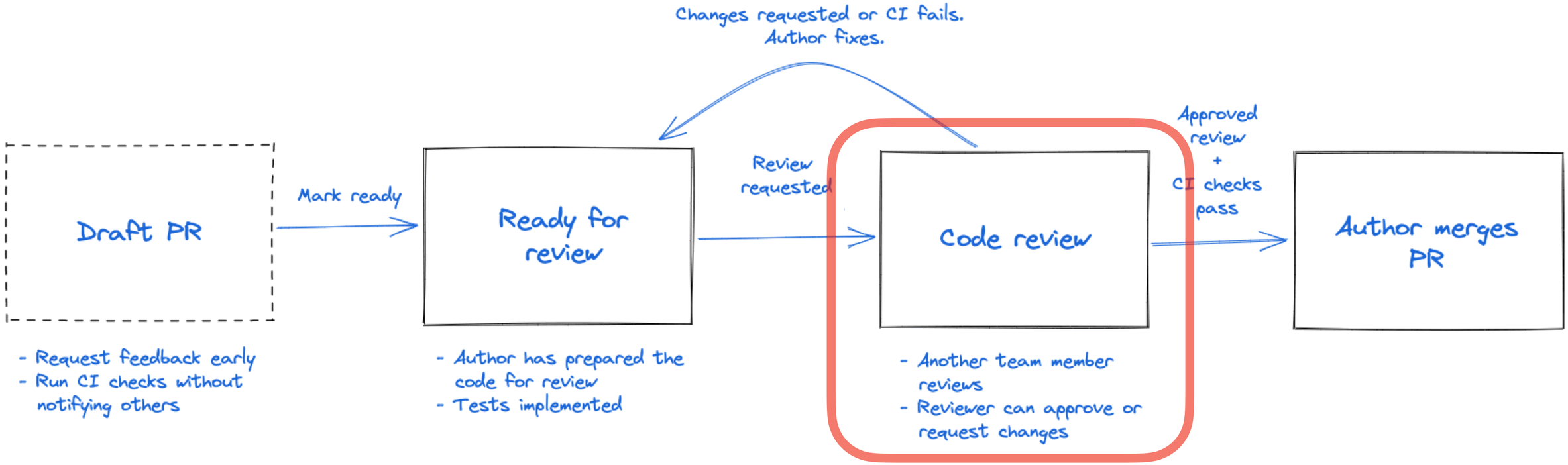
Keep PRs small

- Good for author
  - *higher quality code*
  - *less wasted work if PR is rejected*
  - *easier to merge*
- Good for reviewer
  - *more quickly*
  - *more thoroughly*






Changes requested or CI fails.
Author fixes.
Draft PR
Mark ready
Ready for review
Review requested
Code review
Approved review + CI checks pass
Author merges PR
- Request feedback early
- Run CI checks without notifying others
- Author has prepared the code for review
- Tests implemented
- Another team member reviews
- Reviewer can approve or request changes


Changes requested or CI fails.
Author fixes.
Draft PR
Mark ready
Ready for review
Review requested
Code review
Approved review + CI checks pass
Author merges PR
- Request feedback early
- Run CI checks without notifying others
- Author has prepared the code for review
- Tests implemented
- Another team member reviews
- Reviewer can approve or request changes

Getting your PR Reviewed

- First review it yourself !
- Pick PR reviewers with care
  - *author / owner of the code you modified*
  - *peers on your dev team*
  - *experts on the code or on the kind of changes you made*
  - *other stakeholders who may care (and who might cause trouble later)*
- Communicate clearly with the reviewers
  - *define the goals and objectives of the code review*
  - *e.g., review functionality, performance, reliability, maintainability, security, ... ???*

Reviewing The PR

- Develop a checklist
  - *E.g., functionality, design, complexity, naming, comments, documentation*
- Review rate: 200-500 LOC/hour*
  - *Do not review for more than 1h at a time*
- Finish review speedily and decide (Comment/Approve/Request changes)
- Be rational, clear, and explicit
- Don't point out defects but opportunities for improvement

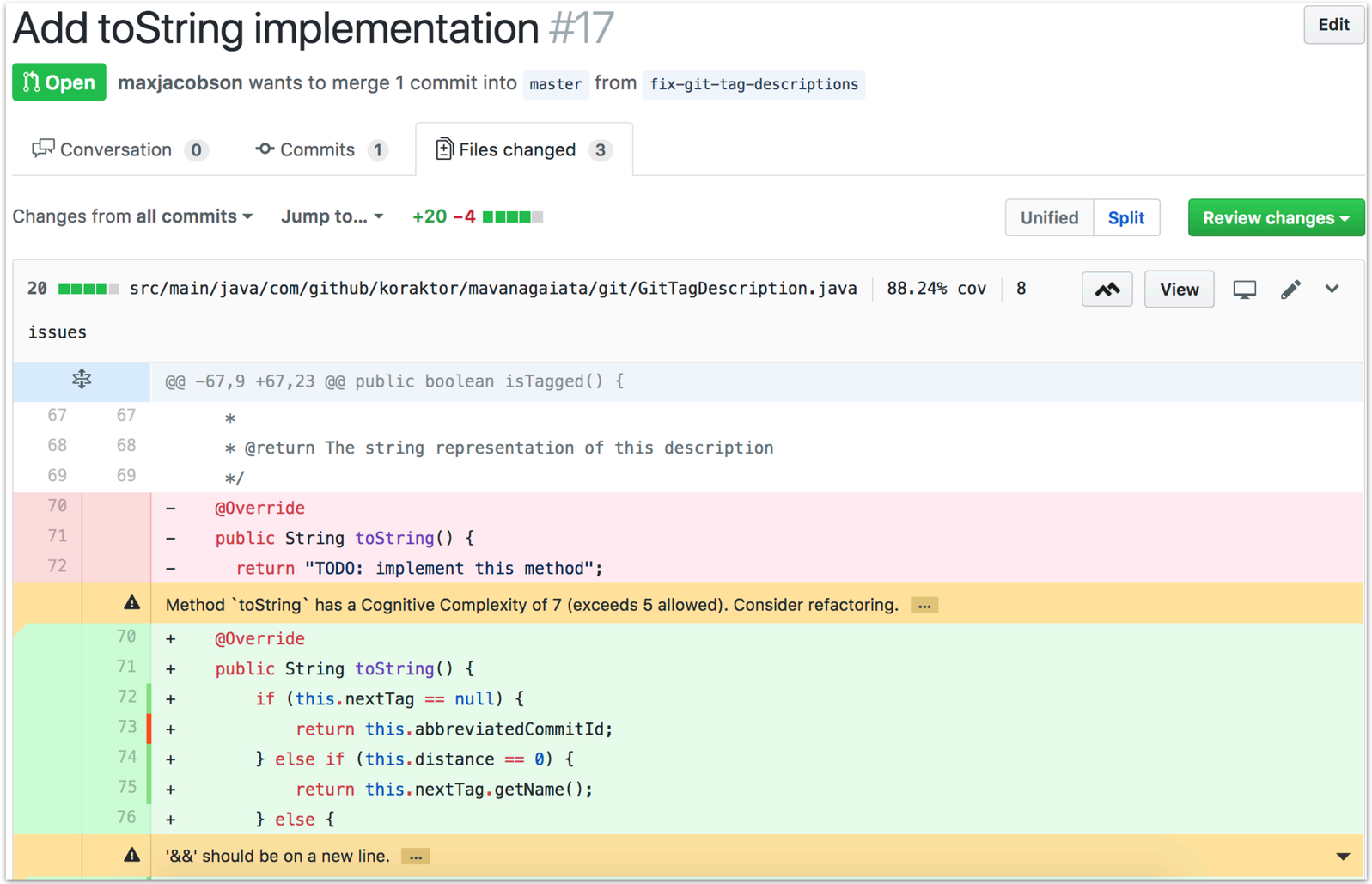
https://www.freepik.com/premium-photo/man-thinking-hand-book-dark-background_9555778.htm

Check out
https://google.github.io/eng-practices/review/reviewer/

* https://static1.smartbear.co/support/media/resources/cc/book/code-review-cisco-case-study.pdf

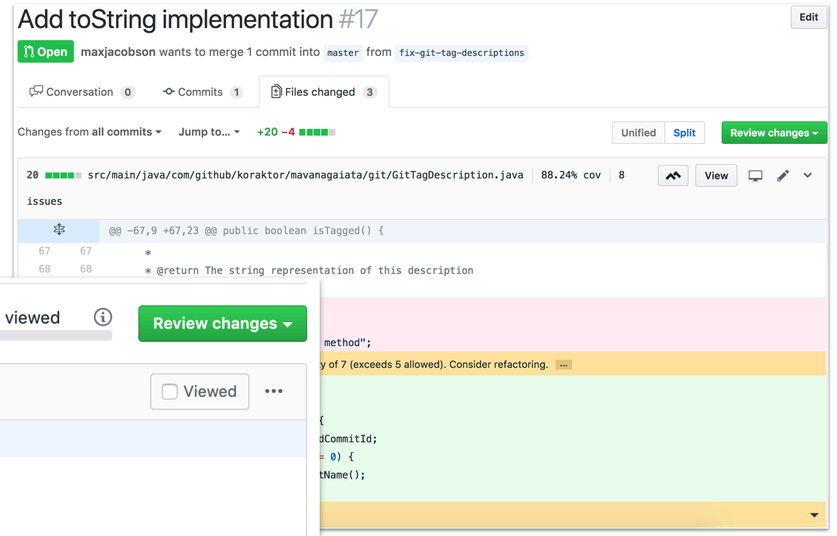
Delegate to Computers Everything that Computers Can Do

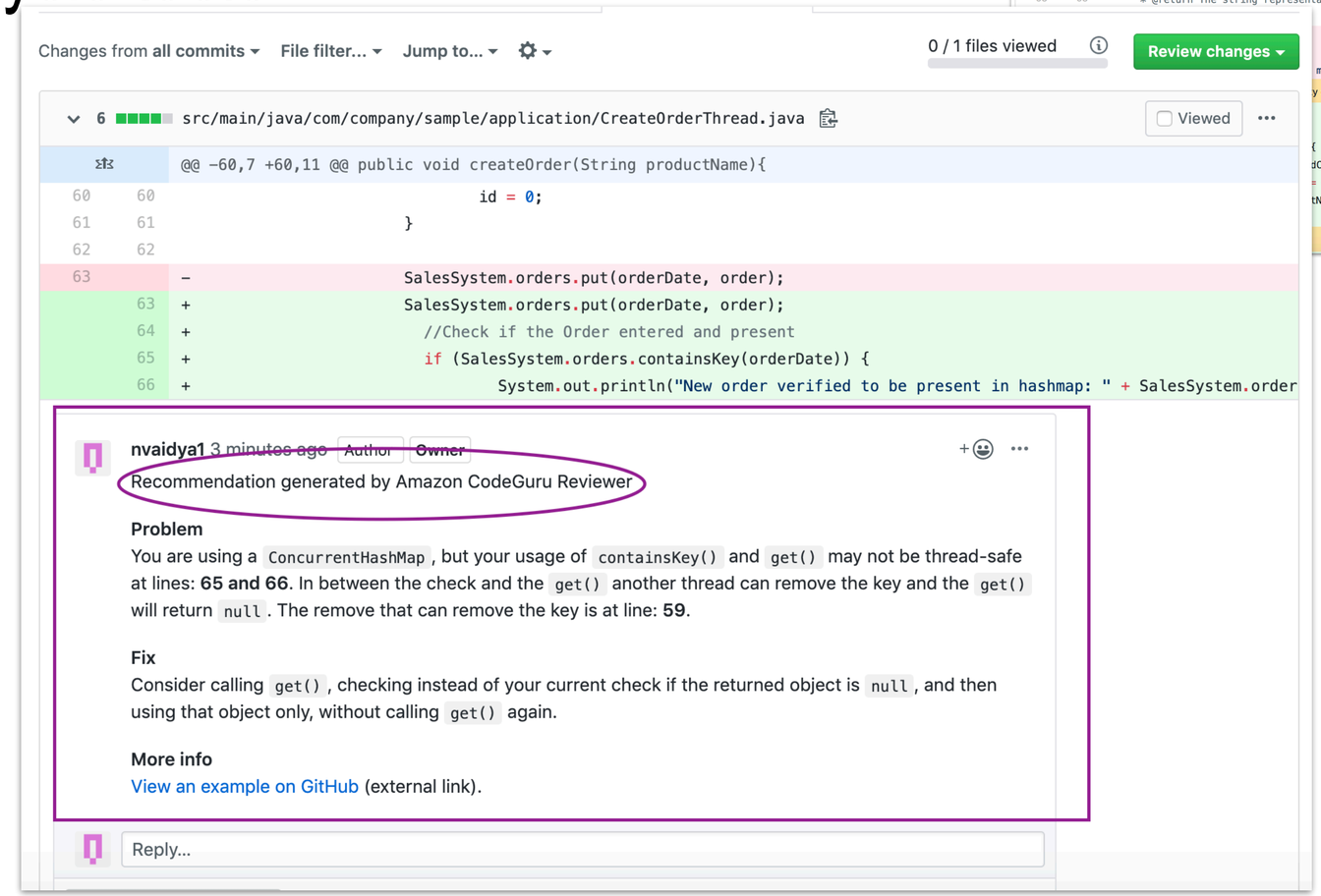
- Use lint tools



Delegate to Computers Everything that Computers Can Do

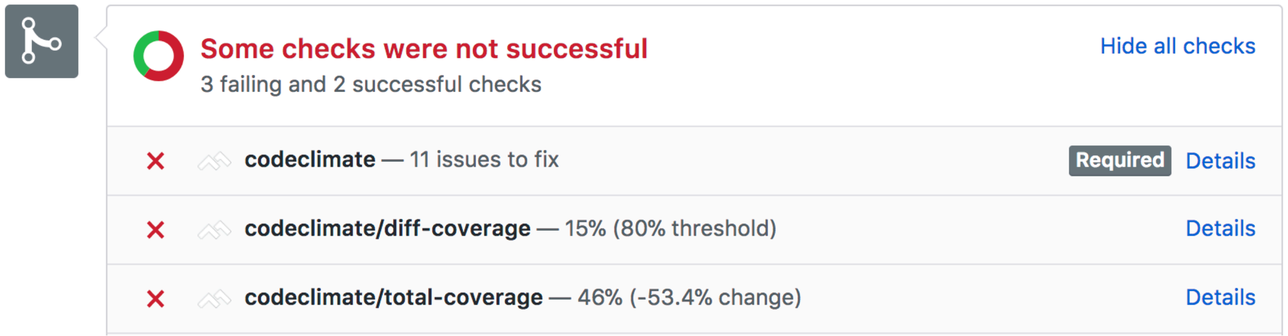
- Use lint tools
- Use static analysis tools

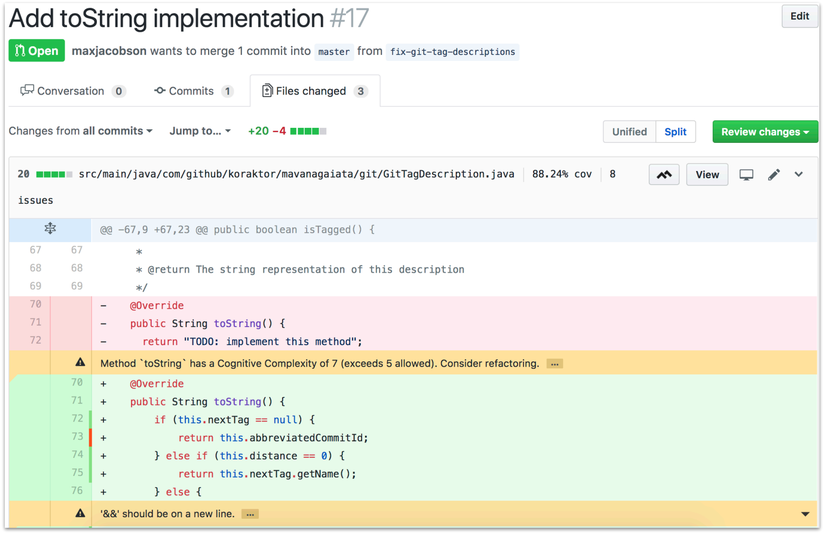


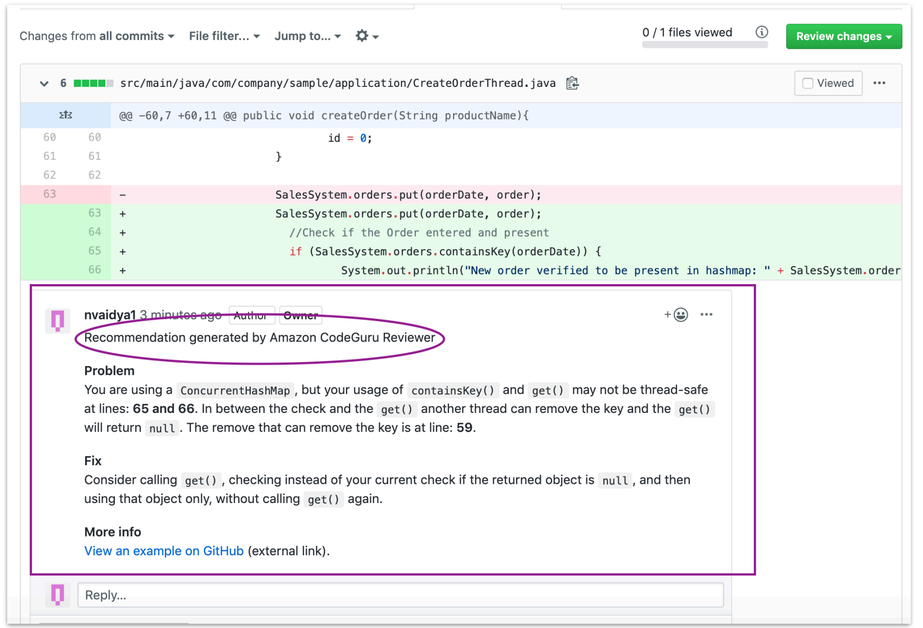


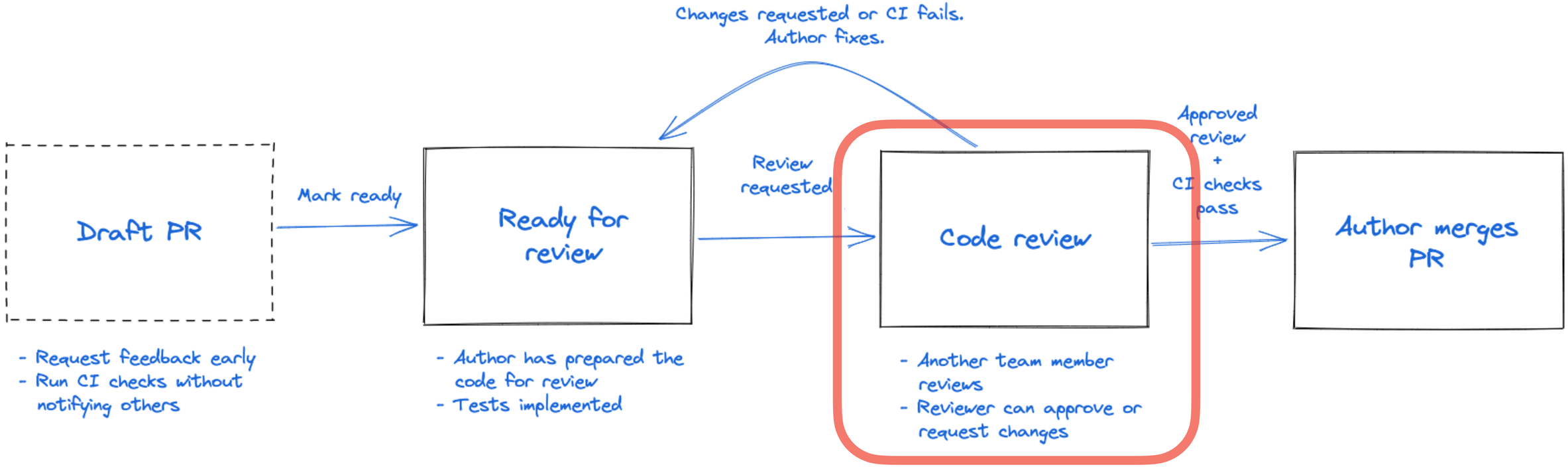
Delegate to Computers Everything that Computers Can Do

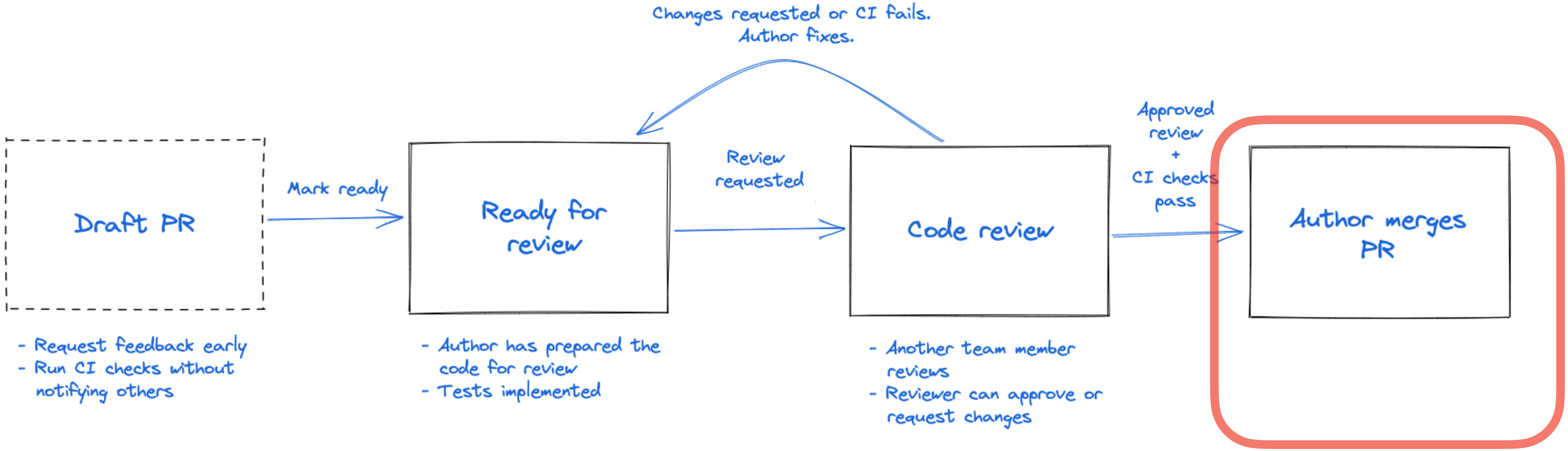
- Use lint tools
- Use static analysis tools
- Integrate with CI

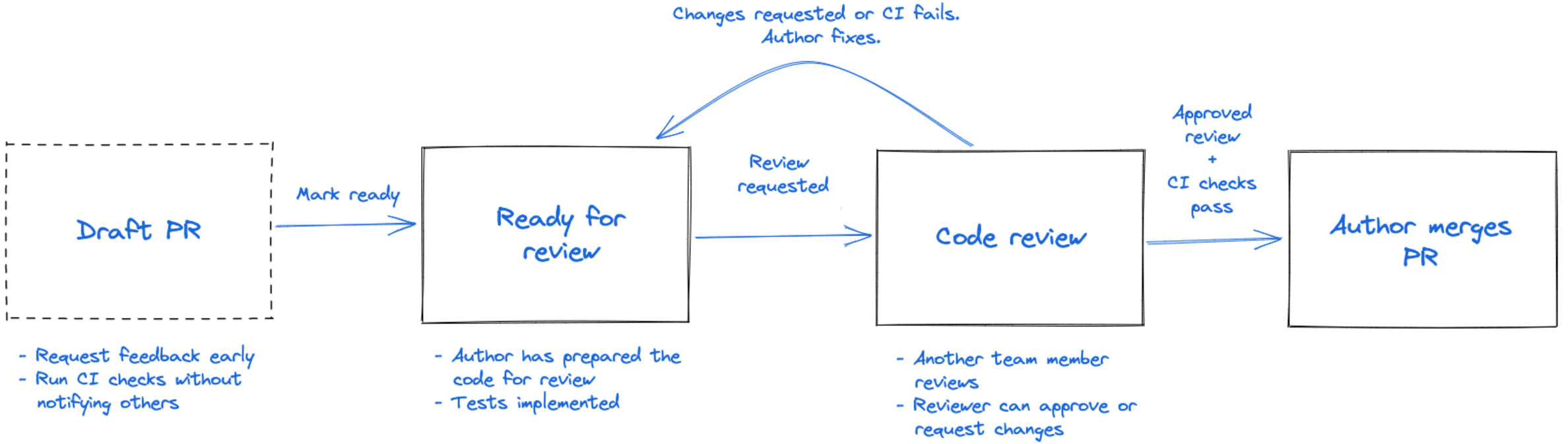







Changes requested or CI fails.
Author fixes.
Draft PR
Mark ready
Ready for review
Review requested
Code review
Approved review + CI checks pass
Author merges PR
- Request feedback early
- Run CI checks without notifying others
- Author has prepared the code for review
- Tests implemented
- Another team member reviews
- Reviewer can approve or request changes


Changes requested or CI fails.
Author fixes.
Draft PR
Mark ready
Ready for review
Review requested
Code review
Approved review + CI checks pass
Author merges PR
- Request feedback early
- Run CI checks without notifying others
- Author has prepared the code for review
- Tests implemented
- Another team member reviews
- Reviewer can approve or request changes


Changes requested or CI fails.
Author fixes.
Draft PR
Mark ready
Ready for review
Review requested
Code review
Approved review + CI checks pass
Author merges PR
- Request feedback early
- Run CI checks without notifying others
- Author has prepared the code for review
- Tests implemented
- Another team member reviews
- Reviewer can approve or request changes


main
feature1
feature2
PR
PR

Outline

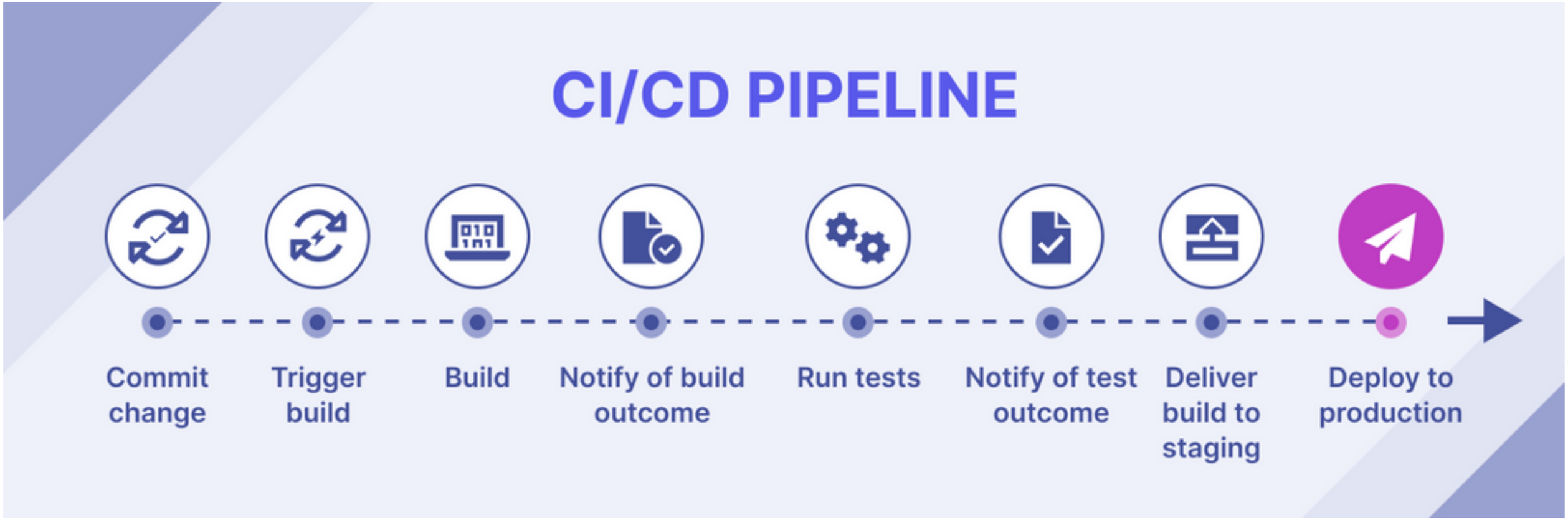
- Development process: Scrum
- Collaboration workflows
- CI / CD
- Writing good commit messages — *online*
- Coding standards — *online*

CI / CD

Continuous Integration / Continuous Delivery & Deployment

- = automated assembly line for your software product
- goals of Continuous Integration
  - *merge code changes from multiple devs and ensure that they work together*
  - *catch bugs and integration problems as early as possible*
- goals of Continuous Delivery
  - *constantly have the software in "deployable state"*
  - *(automatically package software ready for deployment)*
- goals of Continuous Deployment
  - *automatically deploy to testing / staging / production*

Typical CI/CD Workflow

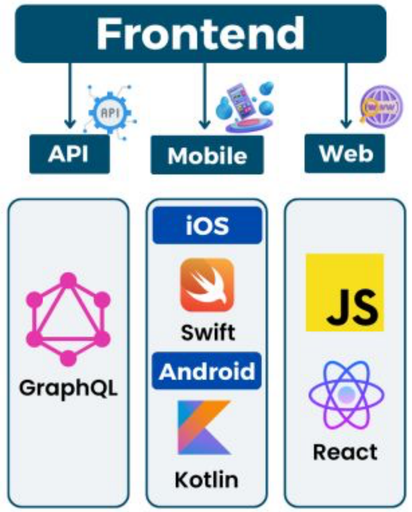


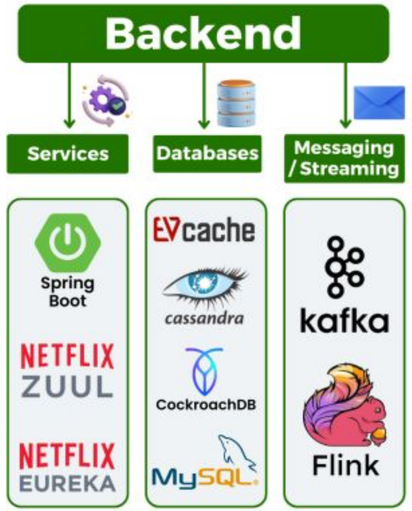
https://katalon.com/resources-center/blog/ci-cd-pipeline

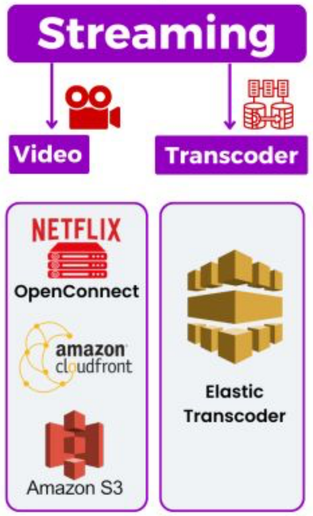
CI/CD Tools

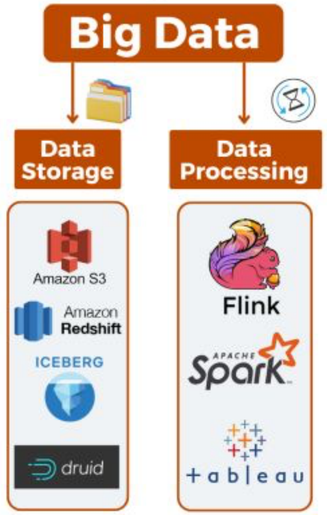
- Version control
- CI/CD server to orchestrate the build—test—deploy processes
- Builder and dependency handler
- Testing frameworks
- Code review tools
- Static code analysis
- VMs and/or containers
- Infrastructure as Code (IaC)
- Monitoring and alerting

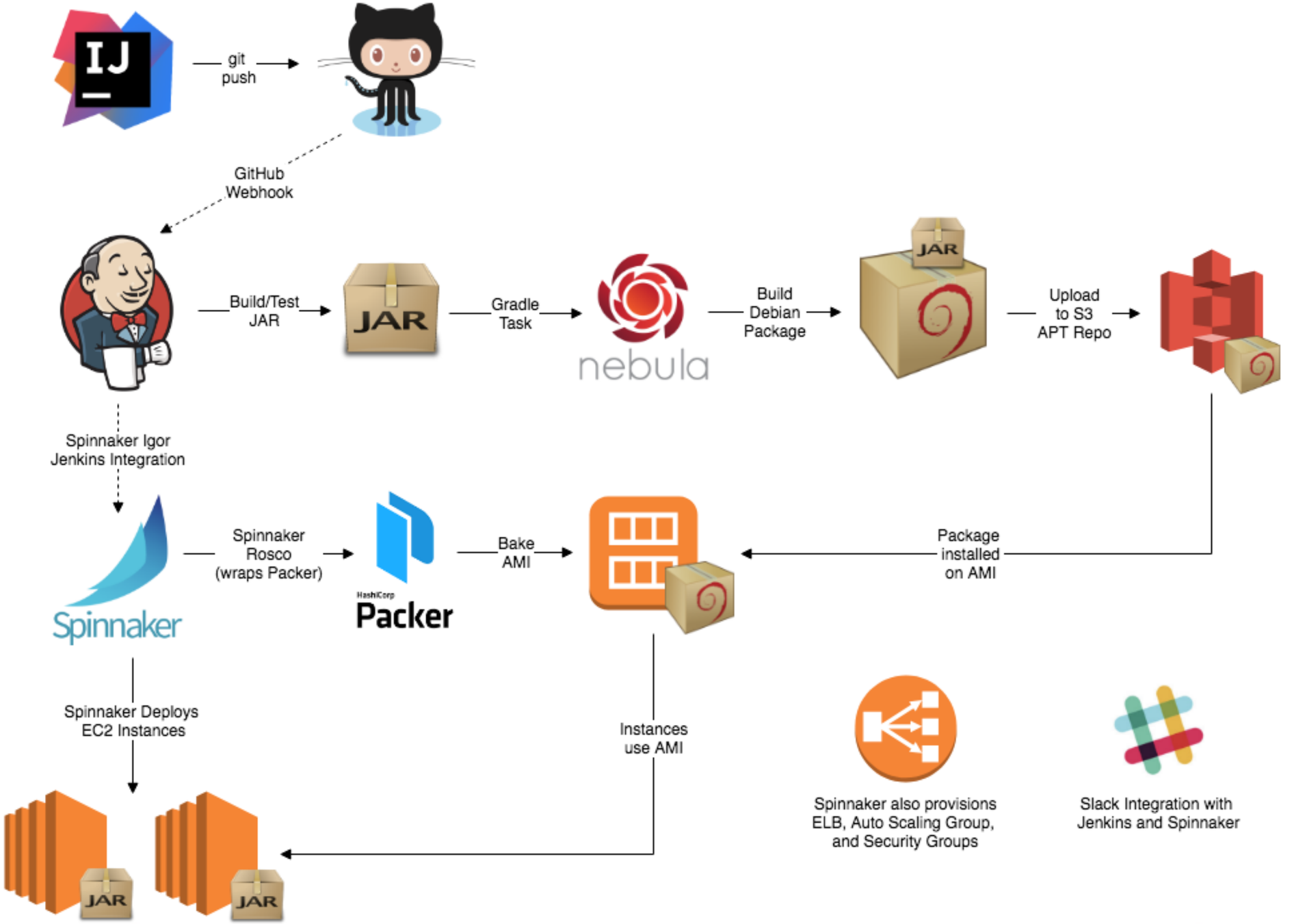
NETFLIX



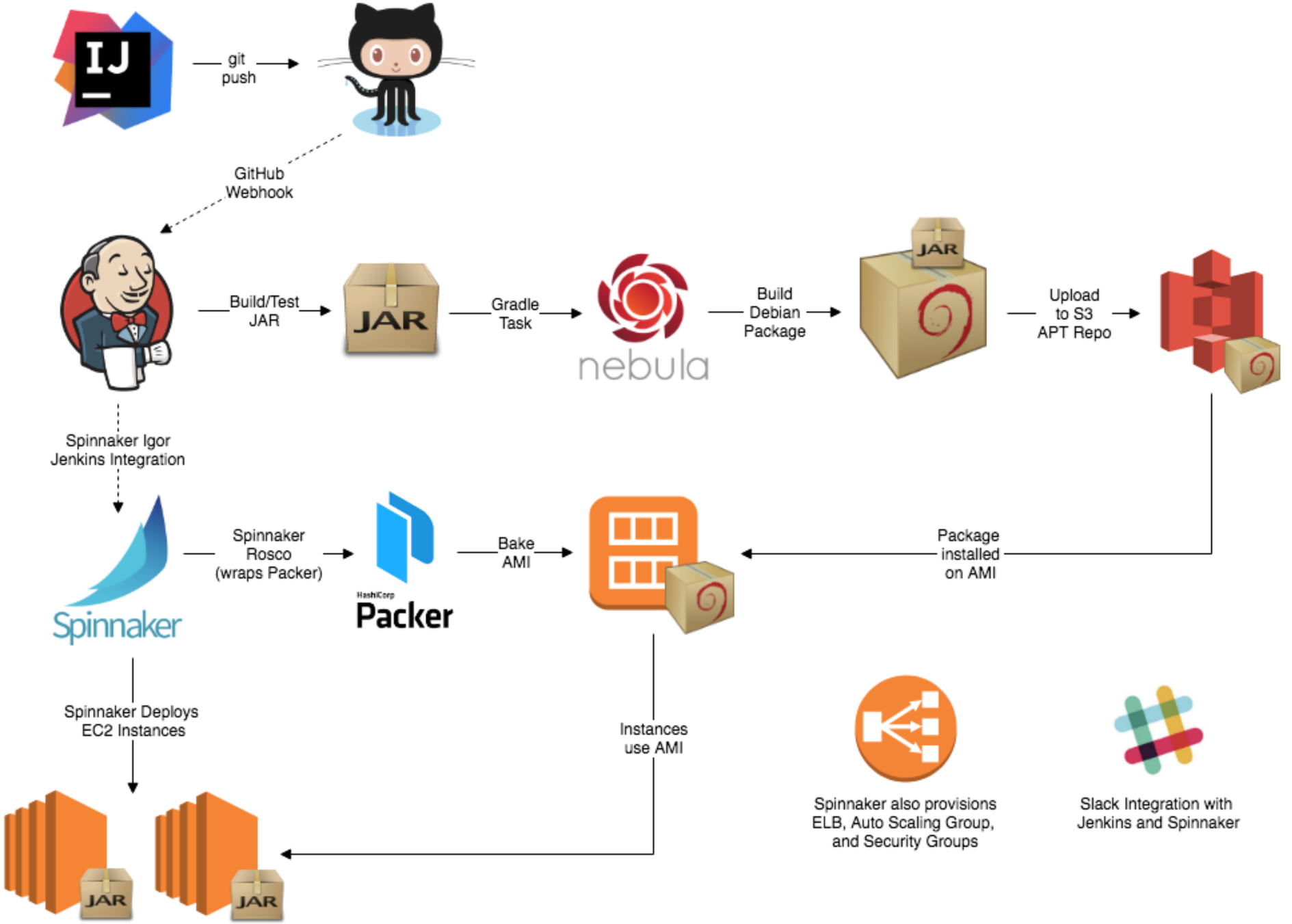




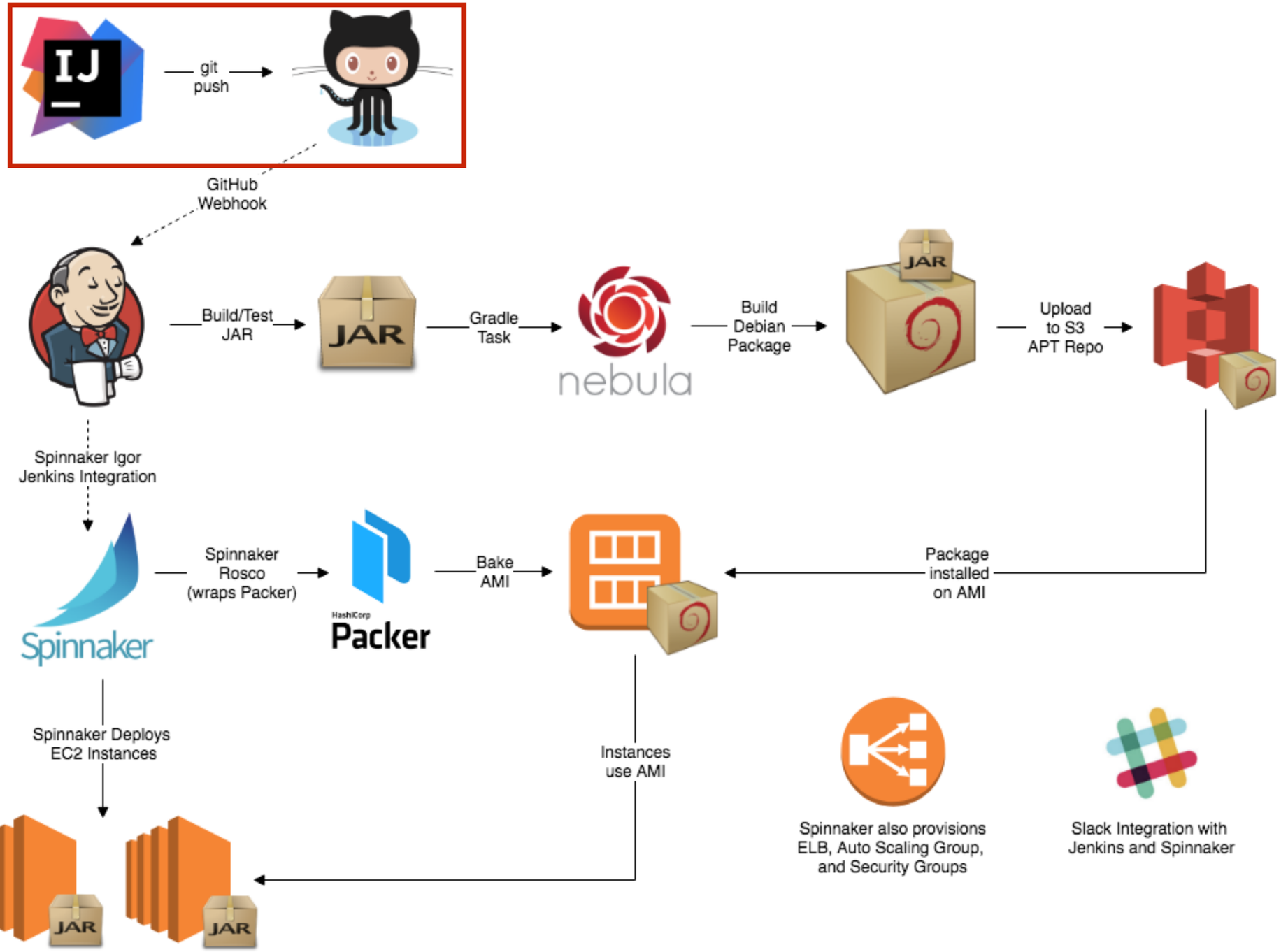




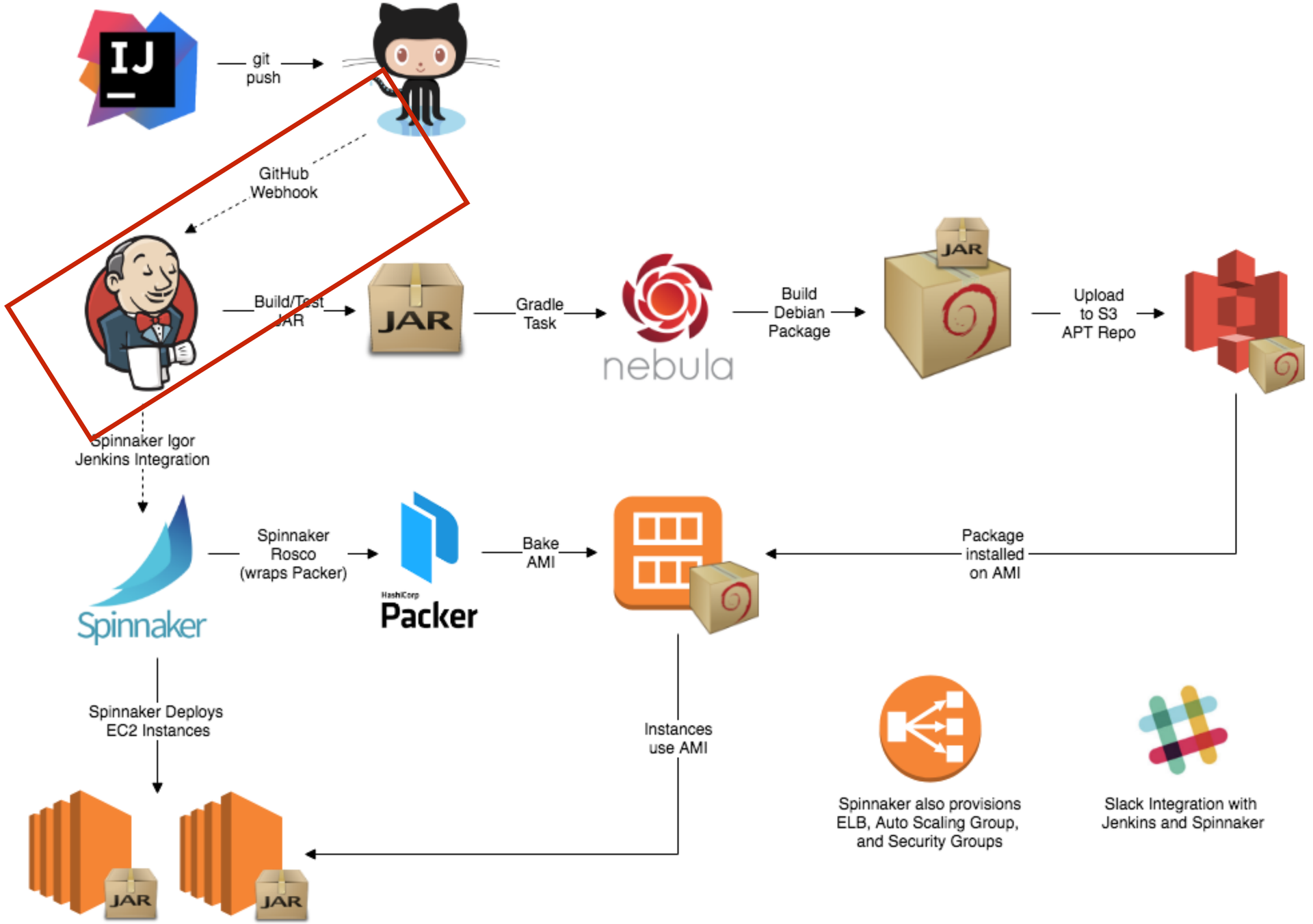
https://programmaticponderings.com/2018/05/12/deploying-spring-boot-apps-to-aws-with-netflix-nebula-and-spinnaker-part-2-of-2/



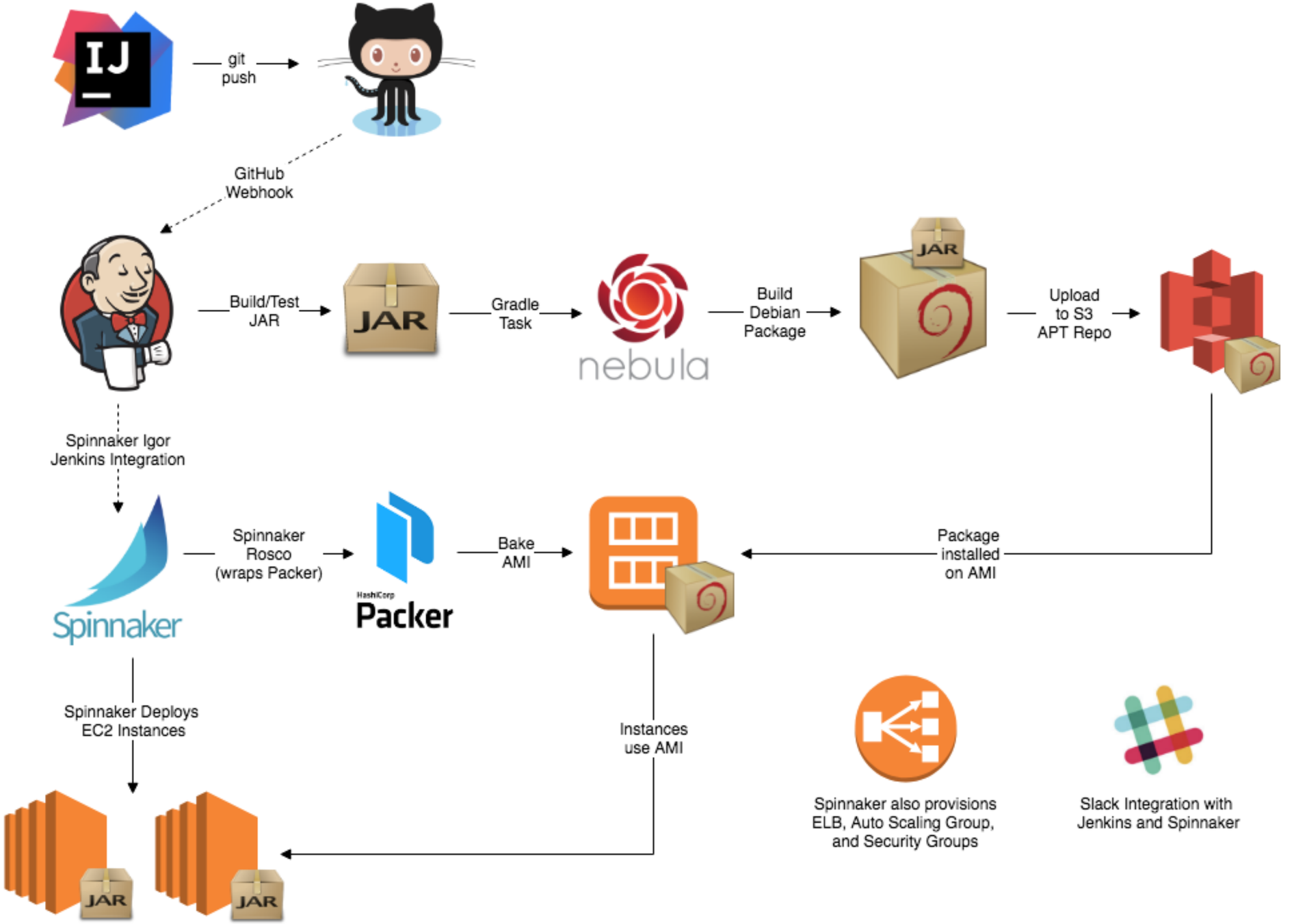
https://programmaticponderings.com/2018/05/12/deploying-spring-boot-apps-to-aws-with-netflix-nebula-and-spinnaker-part-2-of-2/



https://programmaticponderings.com/2018/05/12/deploying-spring-boot-apps-to-aws-with-netflix-nebula-and-spinnaker-part-2-of-2/



https://programmaticponderings.com/2018/05/12/deploying-spring-boot-apps-to-aws-with-netflix-nebula-and-spinnaker-part-2-of-2/



https://programmaticponderings.com/2018/05/12/deploying-spring-boot-apps-to-aws-with-netflix-nebula-and-spinnaker-part-2-of-2/

Chaos Monkey

Randomly disables production instances

Janitor Monkey

Identifies and disposes unused resources

Chaos Kong

Drops a full AWS Region

Conformity Monkey

Shuts down instances not adhering to best-practices

Chaos Gorilla

Outage of entire Amazon Availability Zone

Security Monkey

Finds security violations and vulnerability

NETFLIX SIMIAN ARMY

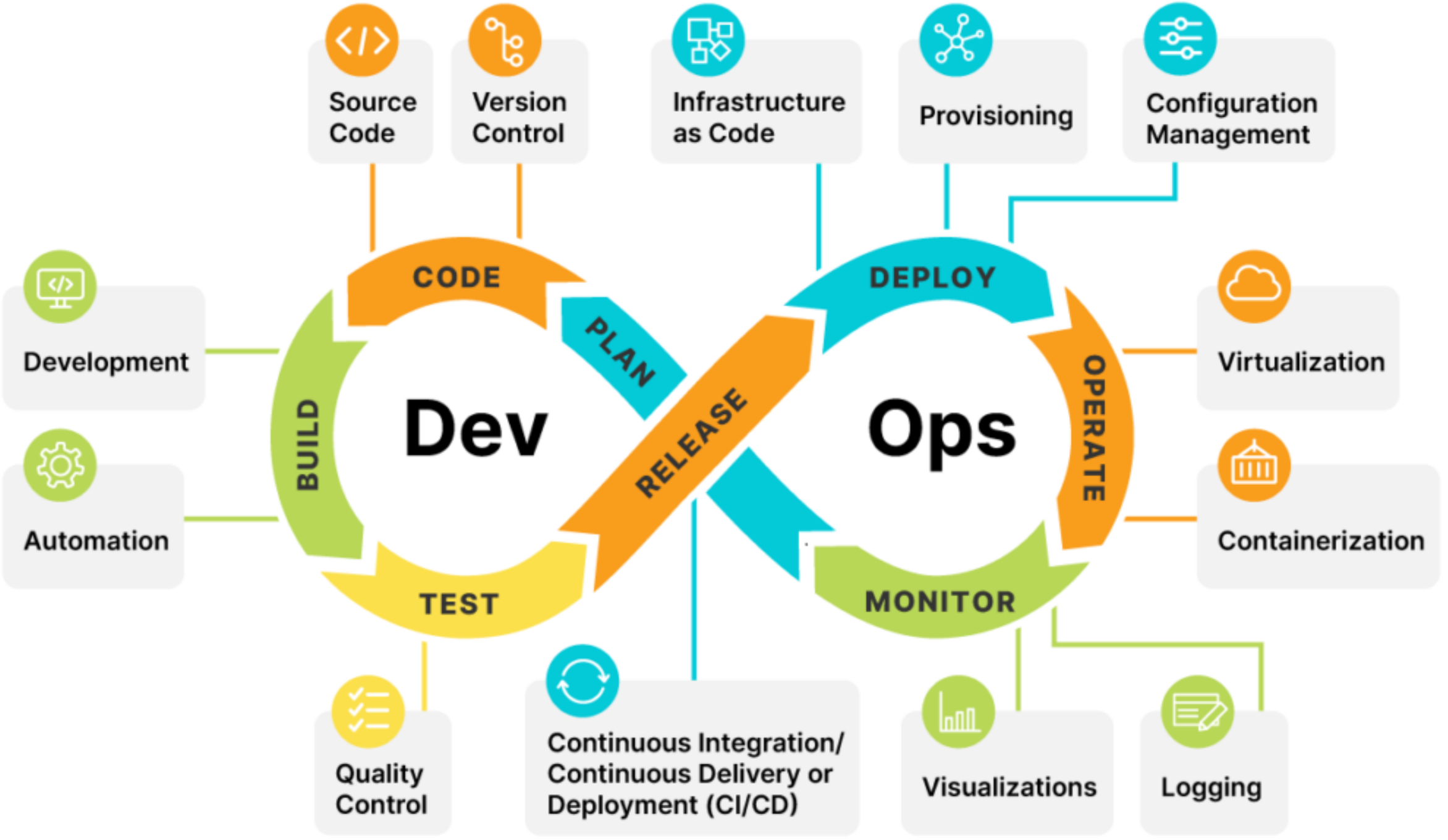
@geosley

Doctor Monkey

Taps into health checks and fixes unhealthy resources

Latency Monkey

Simulate degradation or outages in a network


Source Code
Version Control
Infrastructure as Code
Provisioning
Configuration Management
Development
Automation
CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR
Dev
Ops
Virtualization
Containerization
Quality Control
Continuous Integration/ Continuous Delivery or Deployment (CI/CD)
Visualizations
Logging

Outline

- Development process: Scrum
- Collaboration workflows
- CI / CD
- Writing good commit messages — online
- Coding standards — online