HandBook - IrontLab

# Scrum

### What is it?

Scrum is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems.

In a nutshell, Scrum requires a Scrum Master to foster an environment where:

1. A Product Owner orders the work for a complex problem into a Product Backlog.
2. The Scrum Team turns a selection of the work into an Increment of value during a Sprint.
3. The Scrum Team and its stakeholders inspect the results and adjust for the next Sprint.
4. *Repeat*

Scrum is founded on empiricism and lean thinking. Empiricism asserts that knowledge comes from experience and making decisions based on what is observed. Lean thinking reduces waste and focuses on the essentials.

Scrum employs an iterative, incremental approach to optimize predictability and to control risk. Scrum engages groups of people who collectively have all the skills and expertise to do the work and share or acquire such skills as needed.

Scrum combines four formal events for inspection and adaptation within a containing event, the Sprint. These events work because they implement the empirical Scrum pillars of transparency, inspection, and adaptation.

### Scrum Values

***Commitment, Focus, Openness, Respect, and Courage***

These values give direction to the Scrum Team with regard to their work, actions, and behavior. The decisions that are made, the steps taken, and the way Scrum is used should reinforce these values, not diminish or undermine them. The Scrum Team members learn and explore the values as they work with the Scrum events and artifacts. When these values are embodied by the Scrum Team and the people they work with, the empirical Scrum pillars of transparency, inspection, and adaptation come to life building trust.

### Scrum Team

The fundamental unit of Scrum is a small team of people, a Scrum Team. The Scrum Team consists of one Scrum Master, one Product Owner, and Developers. Within a Scrum Team, there are no sub-teams or hierarchies. It is a cohesive unit of professionals focused on one objective at a time, the Product Goal.

Scrum Teams are cross-functional, meaning the members have all the skills necessary to create value each Sprint. They are also self-managing, meaning they internally decide who does what, when, and how.

Members:

* Scrum Master:The scrum master helps to keep the team accountable to their commitments to the business and also remove any roadblocks that might impede the team’s productivity. The role of a scrum master is to coach and motivate team members, not enforce rules to them.
* Product Owner: Is the Team member who knows what the customer wants and the relative business value of those wants. He or she can then translate the customer’s wants and values back to the Scrum team.he must have the authority to make all decisions necessary to complete the project, in other words, the Product Owner is responsible for managing the [Product Backlog](https://www.visual-paradigm.com/scrum/what-is-product-backlog-in-scrum/)
* Developers: Development Teams are structured and empowered by the organization to organize and manage their own work. The resulting synergy optimizes the Development Team’s overall efficiency and effectiveness.

### Scrum Events

Sprint

They are fixed length events of one month or less to create consistency. A new Sprint starts immediately after the conclusion of the previous Sprint.The big advantage of the short time frame of a sprint is that developers are forced to focus on pushing out small, incremental changes rather than large, sweeping changes. As a result, far less debugging is required, and clients using the software can get a more seamless experience with the product.

Sprint Planning

Though the sprint itself might not last long, there's a lot of careful planning behind it — what goals should be achieved, how long it should last, and when to start.Sprint planning involves product owners working with [Software Engineers](https://www.codecademy.com/resources/blog/what-does-a-software-engineer-do/?utm_source=ccblog&utm_medium=ccblog&utm_campaign=ccblog&utm_content=cw_what_is_sprint_blog) and various technical teams to ensure that the sprint outcomes are relevant and achievable.

Spring Daily

The Daily Scrum is a 15-minute event for the Developers of the Scrum Team. To reduce complexity, it is held at the same time and place every working day of the Sprint.Daily Scrums improve communications, identify impediments, promote quick decision-making, and consequently eliminate the need for other meetings.

Sprint Review

During the event, the Scrum Team and stakeholders review what was accomplished in the Sprint and what has changed in their environment. Based on this information, attendees collaborate on what to do next. The Product Backlog may also be adjusted to meet new opportunities. The Sprint Review is a working session and the Scrum Team should avoid limiting it to a presentation.is timeboxed to a maximum of four hours for a one-month Sprint. For shorter Sprints, the event is usually shorter.

Sprint Retrosprective

The Scrum Team inspects how the last Sprint went with regards to individuals, interactions, processes, tools, and their Definition of Done. Inspected elements often vary with the domain of work. Assumptions that led them astray are identified and their origins explored. The Scrum Team discusses what went well during the Sprint, what problems it encountered, and how those problems were (or were not) solved.It is timeboxed to a maximum of three hours for a one-month Sprint. For shorter Sprints, the event is usually shorter.

### Scrum Artifacts

Scrum’s artifacts represent work or value. They are designed to maximize transparency of key information. Thus, everyone inspecting them has the same basis for adaptation.

Each artifact contains a commitment to ensure it provides information that enhances transparency and focus against which progress can be measured:

* For the Product Backlog it is the Product Goal.
* For the Sprint Backlog it is the Sprint Goal.
* For the Increment it is the Definition of Done.

Product Backlog

The Product Backlog is an emergent, ordered list of what is needed to improve the product. It is the single source of work undertaken by the Scrum Team.Product Backlog items that can be Done by the Scrum Team within one Sprint are deemed ready for selection in a Sprint Planning event.

Sprint Backlog

The Sprint Backlog is composed of the Sprint Goal (why), the set of Product Backlog items selected for the Sprint (what), as well as an actionable plan for delivering the Increment (how).The Sprint Backlog is a plan by and for the Developers. It is a highly visible, real-time picture of the work that the Developers plan to accomplish during the Sprint in order to achieve the Sprint Goal.

Increment

An Increment is a concrete stepping stone toward the Product Goal. Each Increment is additive to all prior Increments and thoroughly verified, ensuring that all Increments work together. In order to provide value, the Increment must be usable.

**Story points** are a unit of measurement for estimating the effort required to complete a work item on the backlog. They are a number that the Developers on the Scrum Team come up with and agree on during the Backlog Refinement or Sprint Planning event.Most teams choose to follow the Fibonacci sequence (0, 1, 1, 2, 3, 5, 8, 13, 21, etc.).

**Velocity** is a metric for the total amount of work that the Scrum Team has completed in a given sprint. Teams that estimate their work items in story points also measure their velocity in story points

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

# GIT

### What is Git?

Git is a [free and open source](https://git-scm.com/about/free-and-open-source) distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git doesn’t think of or store its data this way. Instead, Git thinks of its data more like a series of snapshots of a miniature filesystem. With Git, every time you commit, or save the state of your project, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot. To be efficient, if files have not changed, Git doesn’t store the file again, just a link to the previous identical file it has already stored

### Most Common Commands

#### Git Config

Before you can start using Git, you need to configure it. This command allows you to specify the username and email address that will be used with your commits.

#### Git Init

Create a new Git repository.The git init command creates all of the necessary files and directories for Git to keep track of everything. All of these files are kept in a directory called .git.

#### Git Clone

The command is git clone, and the path to the Git repository you want to clone is passed as an argument.When you clone a repository, the code is downloaded to your machine automatically. If you have permission, this command will add the original location as a remote location, allowing you to grab changes from it and push changes to it.

#### Git status

The git status command is our key to Git’s mind. It will inform us of Git’s thoughts and the state of our repository as seen by Git.

#### Git add

To move files from the Working Directory to the Staging Index, use the git add command. The git add command saves your changes in a file to the staging area, allowing you to compare your local version to the remote repository’s version.

#### Git commit

This command saves a log message along with the commit id of the modifications made to the git repository. The modifications are saved in your local repository with git commit. You must include a brief description of the changes made every time you commit your code changes. This commit message aids others in comprehending the changes made.

#### Git push

This command pushes the contents of your local repository to the remote repository you’ve added. This pushes your master branch’s commits to the newly added remote repository.

#### Git branch

Add a new branch to an existing branch, list all existing branches, and delete a branch with git branch.

#### Git checkout

We can use the git checkout command to switch to an existing branch or to create and switch to a new one. To accomplish this, the branch you want to switch to must be present in your local system, and any modifications made to your current branch must be committed or stashed before switching.

#### Git merge

The command git merge joins your branch to the parent branch. there are no conflicts, it will make a new commit automatically.

#### Git pull

The contents of the remote repository are fetched and integrated into your local repository using this command. git pull pulls the most recent changes from the remote server into the local repository, ensuring you have the most up-to-date information from your coworkers.

### Tagging

Git has the ability to tag specific points in a repository’s history as being important.

Listing the existing tags in Git is straightforward. Just type git tag (with optional -l or --list)

Git supports two types of tags: **lightweight** and **annotated**.

A lightweight tag is very much like a branch that doesn’t change — it’s just a pointer to a specific commit.

Annotated tags, however, are stored as full objects in the Git database. They’re checksummed; contain the tagger name, email, and date; have a tagging message; and can be signed and verified with GNU Privacy Guard (GPG).

the git push command doesn’t transfer tags to remote servers. You will have to explicitly push tags to a shared server after you have created them. This process is just like sharing remote branches — you can run git push origin <tagname>.

If you have a lot of tags that you want to push up at once, you can also use the --tags option to the git push command. This will transfer all of your tags to the remote server that are not already there.

### Stashing

Stashing takes the dirty state of your working directory — that is, your modified tracked files and staged changes — and saves it on a stack of unfinished changes that you can reapply at any time (even on a different branch).To demonstrate stashing, you’ll go into your project and start working on a couple of files and possibly stage one of the changes. If you run git status, you can see your dirty state.Now you want to switch branches, but you don’t want to commit what you’ve been working on yet, so you’ll stash the changes. To push a new stash onto your stack, run git stash or git stash push.At this point, you can switch branches and do work elsewhere; your changes are stored on your stack. To see which stashes you’ve stored, you can use git stash list.You can reapply the one you just stashed by using the command shown in the help output of the original stash command: git stash apply. If you want to apply one of the older stashes, you can specify it by naming it, like this: git stash apply stash@{2}. If you don’t specify a stash, Git assumes the most recent stash and tries to apply it.

<https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F>

https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/