**Pablo Gonzalez Bootcamp- first part**

**SCRUM**

What is scrum?

Agile project management framework that helps teams work together, generate value through adaptive solutions for complex problems.

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

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

# Scrum Theory

Scrum is founded on empiricism and lean thinking.

Scrum employs an iterative, incremental approach to optimize predictability and to control risk.

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.**

### Transparency: The emergent process and work must be visible to those performing the work as well as those receiving the work Transparency enables inspection. Inspection without transparency is misleading and wasteful.

### Inspection: The Scrum artifacts and the progress toward agreed goals must be inspected frequently and diligently to detect potentially undesirable variances or problems.

Inspection enables adaptation. Inspection without adaptation is considered pointless. Scrum events are designed to provoke change.

### Adaptation: If any aspects of a process deviate outside acceptable limits or if the resulting product is unacceptable, the process being applied or the materials being produced must be adjusted. The adjustment must be made as soon as possible to minimize further deviation. Scrum Team is expected to adapt the moment it learns anything new through inspection.

# Scrum Values

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

***The Scrum Team:***

***—- commits to achieving its goals and to supporting each other.***

***—- primary focus is on the work of the Sprint***

***—- are open about the work and the challenges.***

***—- members respect each other to be capable, independent people.***

***—- have the courage to do the right thing, to work on tough problems***

These values give direction to the Scrum Team.

The steps taken, and the way Scrum is used should reinforce these values, not diminish or undermine them

# 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.

are cross-functional meaning the members have all the skills necessary to create value each Sprint.

self-managing meaning they internally decide who does what, when, and how.

small enough to remain nimble

large enough to complete significant work within a Sprint

* **there are no sub-teams**
* **is responsible** for all product-related activities from stakeholder collaboration, verification, maintenance, operation, experimentation, research and development

## Developers

are the people in the Scrum Team that are committed to creating any aspect of a usable Increment each Sprint.

Developers are always accountable for:

* Creating a plan for the Sprint, the Sprint Backlog;
* Instilling quality by adhering to a Definition of Done;
* Adapting their plan each day toward the Sprint Goal; and,
* Holding each other accountable as professionals.

## Product Owner

is accountable for maximizing the value of the product resulting from the work of the Scrum

The Product Owner is also accountable for effective Product Backlog management, which includes:

* Developing and explicitly communicating the Product Goal;
* Creating and clearly communicating Product Backlog items;
* Ordering Product Backlog items; and,
* Ensuring that the Product Backlog is transparent, visible and understood.

The Product Owner is one person, not a committee.

## Scrum Master

is accountable for

* establishing Scrum as defined in the Scrum Guide. They do this by helping everyone understand Scrum theory and practice, both
* for the Scrum Team’s effectiveness
* are true leaders
* serves the Scrum Team in several ways, including:
* Coaching the team members
* Helping the Scrum Team focus
* Ensuring that all Scrum events take place and are positive, productive, and kept within the timebox.
* serves the Product Owner in several ways, including:
* Helping find techniques for effective Product Goal definition and Product Backlog management;
* Helping establish empirical product planning for a complex environment
* Facilitating stakeholder collaboration as requested or needed.
* The Scrum Master serves the organization in several ways, including:
* Leading, training, and coaching the organization in its Scrum adoption;
* Planning and advising Scrum implementations within the organization;
* Helping employees and stakeholders understand and enact an empirical approach for complex work
* Removing barriers between stakeholders and Scrum Teams.

# Scrum Events

**The Sprint is a container for all other events**

These events are specifically designed to enable the transparency required

Events are used in Scrum to create regularity and to minimize the need for meetings not defined in Scrum.

Optimally, all events are held at the same time and place

# The Sprint

**Sprints are the heartbeat of Scrum, where ideas are turned into value.**

They are fixed length events of one month or less

create consistency

A new Sprint starts immediately after the conclusion of the previous Sprint.

All the work necessary to achieve the Product Goal, including Sprint Planning, Daily Scrums, Sprint Review, and Sprint Retrospective, happen within Sprints.

During the Sprint:

* No changes are made that would endanger the Sprint Goal;
* Quality does not decrease;
* The Product Backlog is refined as needed; and,
* Scope may be clarified and renegotiated with the Product Owner as more is learned.

When a Sprint’s horizon is too long the Sprint Goal may become invalid

Shorter Sprints can be employed to generate more learning cycles and limit risk of cost and effort to a smaller time frame. **Each Sprint may be considered a short project**.

A Sprint could be canceled if the Sprint Goal becomes obsolete. **Only the Product Owner has the authority to cancel the Sprint.**

### Sprint Planning

### Sprint Planning initiates the Sprint by laying out the work to be performed for the Sprint.

### The Product Owner ensures that attendees are prepared to discuss the most important

### Product Backlog items and how they map to the Product Goal.

### The Scrum Team may also invite other people to attend Sprint Planning to provide advice.

### Sprint Planning addresses the following topics:

#### Topic One: Why is this Sprint valuable?

### The Product Owner proposes how the product could increase its value and utility in the current Sprint. The whole Scrum Team then collaborates to define a Sprint Goal The Sprint Goal must be finalized prior to the end of Sprint Planning.

#### Topic Two: What can be Done this Sprint?

### Through discussion with the Product Owner, the Developers select items from the Product Backlog to include in the current Sprint.

### Selecting how much can be completed within a Sprint may be challenging.

### However, the more the Developers know about their past performance, their upcoming capacity, and their Definition of Done, the more confident they will be in their Sprint forecasts.

#### Topic Three: How will the chosen work get done?

### For each selected Product Backlog item, the Developers plan the work necessary to create an Increment that meets the Definition of Done.

### This is often done by decomposing Product Backlog items into smaller work items of one day or less.

### How this is done is at the sole discretion of the Developers. No one else tells them how to turn Product Backlog items into Increments of value.

### **The Sprint Goal, the Product Backlog items selected for the Sprint,**

### Sprint Planning is timeboxed to a maximum of eight hours for a one-month Sprint.

### Daily Scrum

The purpose of the Daily Scrum is to inspect progress toward the Sprint Goal and adapt the Sprint Backlog as necessary, adjusting the upcoming planned work.The Daily Scrum is a 15-minute event for the Developers of the Scrum Team.

Held at the same time and place every working day

If the Product Owner or Scrum Master are actively working on items in the Sprint Backlog, they participate as Developers.

Daily Scrums improve communications, identify impediments, promote quick decision-making, and consequently **eliminate the need for other meetings.**

### Sprint Review

The purpose of the Sprint Review is to inspect the outcome of the Sprint and determine future adaptations. The Scrum Team presents the results of their work to key stakeholders and progress toward the Product Goal is discussed.

The Sprint Review is the second to last event of the Sprint and is timeboxed to a maximum of four hours for a one-month Sprint

### Sprint Retrospective The purpose of the Sprint Retrospective is to plan ways to increase quality and effectiveness.

## Scrum Artifacts Scrum’s artifacts represent work or value.

### 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.

#### Commitment: Product Goal

*A product is a vehicle to deliver value. It has a clear boundary, known stakeholders, well-defined users or customers. A product could be a service, a physical product, or something more abstract.*

<https://www.atlassian.com/agile/scrum>

<https://scrumguides.org/>

## what is Git? sistema de control de versiones

# Ramificaciones en Git - ¿Qué es una rama?

A diferencia de otros sistemas de control de versiones, Git promueve un ciclo de desarrollo donde las ramas se crean y se unen ramas entre sí, incluso varias veces en el mismo día.

Una rama Git es simplemente un apuntador móvil apuntando a una de esas confirmaciones. La rama por defecto de Git es la rama master. Con la primera confirmación de cambios que realicemos, se creará esta rama principal master apuntando a dicha confirmación. En cada confirmación de cambios que realicemos, la rama irá avanzando automáticamente.

## Comandos mas usados:

### **git branch:**

El comando git branch es en realidad una especie de herramienta de gestión de ramas. Puede listar las ramas que tienes, crear una nueva rama, eliminar ramas y cambiar el nombre de las ramas.

### **git checkout**

### El comando git checkout se usa para cambiar de rama y revisar el contenido de tu directorio de trabajo.

### **git log**

El comando git log se utiliza para mostrar la historia registrada alcanzable de un proyecto desde la más reciente instantánea confirmada hacia atrás. Por defecto sólo se mostrará la historia de la rama en la que te encuentres, pero pueden ser dadas diferentes e incluso múltiples cabezas o ramas desde la que hacer el recorrido. También se utiliza a menudo para mostrar las diferencias entre dos o más ramas a nivel de commit.

### **git init**

Para tomar un directorio y convertirlo en un nuevo repositorio Git en el que puedas empezar a controlar sus versiones, simplemente puedes ejecutar git init.

**git clone**

El comando 'git clone` es en realidad una especie de envoltura alrededor de varios otros comandos. Éste crea un nuevo directorio, entra en él y ejecuta git init para que sea un repositorio vacío de Git, añade uno remoto (git remote add) hacia la dirección URL que se le pasa (por defecto llamado origin), ejecuta un git fetch de ese repositorio remoto y después activa el último commit en el directorio de trabajo con git checkout.

### **git add**

El comando git add añade contenido del directorio de trabajo al área de ensayo (staging area o **'index**') para la próxima confirmación. Cuando se ejecuta el comando git commit, éste, de forma predeterminada, sólo mira en esta área de ensayo, por lo que git add se utiliza para fabricar exactamente lo que te gustaría fuese tu próxima instantánea a confirmar.

### 

### **git status**

El comando git status te mostrará los diferentes estados de los archivos en tu directorio de trabajo y área de ensayo. Qué archivos están modificados y sin seguimiento y cuáles con seguimiento pero no confirmados aún. En su forma normal, también te mostrará algunos consejos básicos sobre cómo mover archivos entre estas etapas.

### **git diff**

El comando git diff se utiliza cuando deseas ver las diferencias entre dos árboles. Esto prodría ser la diferencia entre tu entorno de trabajo y tu área de ensayo (git diff por sí mismo), entre tu área de ensayo y tu última confirmación o commit (git diff --staged), o entre dos confirmaciones (git diff master branchB).

### **git commit**

El comando git commit toma todos los contenidos de los archivos a los que se les realiza el seguimiento con git add y registra una nueva instantánea permanente en la base de datos y luego avanza el puntero de la rama en la rama actual.

### **git reset**

El comando git reset se utiliza sobre todo para deshacer las cosas, como posiblemente puedes deducir por el verbo. Se mueve alrededor del puntero HEAD y opcionalmente cambia el index o área de ensayo y también puede cambiar opcionalmente el directorio de trabajo si se utiliza --hard. Esta última opción hace posible que este comando pueda perder tu trabajo si se usa incorrectamente, por lo que asegúrese de entenderlo antes de usarlo.

Utilizamos git reset --hard para abortar una fusión en [Abortar una Fusión](https://git-scm.com/book/es/v2/ch00/r_abort_merge), donde también usamos git merge --abort, el cual es una especie de envoltorio para el comando git reset.

### **git rm**

El comando git rm se utiliza para eliminar archivos del área de ensayo y el directorio de trabajo para Git. Es similar a git add en que pone en escena una eliminación de un archivo para la próxima confirmación.

### **git clean**

El comando git clean se utiliza para eliminar archivos no deseados de tu directorio de trabajo. Esto podría incluir la eliminación de artefactos de construcción temporal o la fusión de archivos en conflicto.

### **git merge**

La herramienta git merge se utiliza para fusionar uno o más ramas dentro de la rama que tienes activa. A continuación avanzará la rama actual al resultado de la fusión.

### **git stash**

El comando git stash se utiliza para almacenar temporalmente el trabajo no confirmado con el fin de limpiar el directorio de trabajo sin tener que confirmar el trabajo no acabado en una rama.

Básicamente esto es enteramente cubierto en [Guardado rápido y Limpieza](https://git-scm.com/book/es/v2/ch00/r_git_stashing)

### **git tag**

El comando git tag se utiliza para dar un marcador permanente a un punto específico en el historial del código fuente. Generalmente esto se utiliza para cosas como las liberaciones (releases).

Este comando se introduce y se trata en detalle en [Etiquetado](https://git-scm.com/book/es/v2/ch00/r_git_tagging) y lo usamos en la práctica en [Etiquetando tus versiones](https://git-scm.com/book/es/v2/ch00/r_tagging_releases).

### **git stash**

Use git stash when you want to record the current state of the working directory and the index, but want to go back to a clean working directory. The command saves your local modifications away and reverts the working directory to match the HEAD commit.

### **git hooks**

Hooks are programs you can place in a hooks directory to trigger actions at certain points in git’s execution. Hooks that don’t have the executable bit set are ignored.

**https://git-scm.com/book/en/v2**

JavaScript

JavaScript (JS) es un lenguaje de programación ligero, interpretado, o compilado [justo-a-tiempo](https://es.wikipedia.org/wiki/Compilaci%C3%B3n_en_tiempo_de_ejecuci%C3%B3n) (just-in-time) con [funciones de primera clase](https://developer.mozilla.org/es/docs/Glossary/First-class_Function). Si bien es más conocido como un lenguaje de scripting (secuencias de comandos)

multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo programación funcional)

**https://developer.mozilla.org/es/docs/Web/JavaScript**