**Source Code Management**

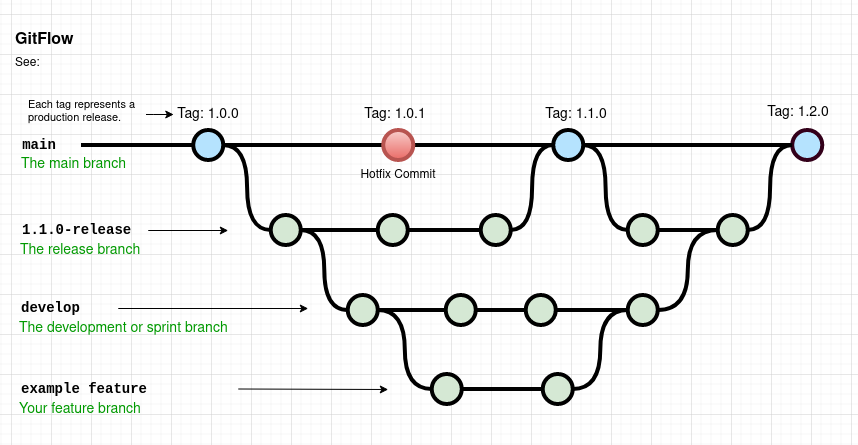
**GitFlow**

Permite a los equipos de desarrollo trabajar de forma separada a la rama principal en las nuevas funcionalidades dónde cada nueva funcionalidad será creada a partir de esta rama principal.

Una vez que los cambios están completos, los desarrolladores fusionan estos cambios de vuelta a la rama principal para que se dé un despliegue de una nueva versión.

Las ramas que se usan en esta estrategia son:

* Main
* Develop
* Feature: las nuevas **features** salen de la rama **develop**
* Release: usualmente esta rama sale de **main** o de **develop**
* Hotfix: es una rama que se crea cuando se quiere resolver un **bug**. Permite a los desarrolladores seguir trabajando en sus propios cambios mientras el **bug** se está solucionando

****

**Convenciones para Release branches y Hotfix branches.**

Para los branches usaremos la estructura

**<token>/<short-descriptive-name>**

* **chore**: mejoras en temas de administración/mantenimiento del proyecto (i.e. actualización de dependencias).
* **docs**: creación/actualización de documentación (i.e.: guía de configuración del proyecto).
* **feature**: nuevas funcionalidades que serán incluidas en el proyecto. (i.e. visualización de cursos).
* **fix/hotfix/patch:** corrección de un bug esperado o inesperado (i.e. links rotos).
* **refactor**: mejoras/reescritura de features existentes, no agrega un cambio grande a lo que actualmente tiene. (i.e. cambiar estados locales usando stateless components conectados a Redux).
* **test**: agrega tests a un feature existente que no cuenta con los mismos (i.e. unit testing del componente de login).

**Ejemplos**:

* docs/setup
* feature/hide-navbar
* refactor/settings-components

**Versionado semántico**

**Fase inicial de desarrollo:** Comenzar publicando tu desarrollo inicial en 0.1.0 y luego incrementar la versión menor por cada siguiente lanzamiento.

**Versión 1.0:** Cuando la página web está lista para ser enviada a producción debería ya estar en la versión 1.0.