GIT CON EL FLUJO GITFLOW

Gitflow es una metodología de flujo de trabajo para el uso de Git, diseñada para facilitar la gestión de proyectos de software. Fue introducida por Vincent Driessen en 2010 y proporciona un marco estructurado para la organización de las ramas y el proceso de desarrollo.

Conceptos Clave de Gitflow

1. **Ramas Principales**

   - **master (main)**: Contiene el código de producción. Cada commit en esta rama debe representar una versión estable del producto.
   - **develop**: Contiene el código de desarrollo integrado. Todas las nuevas características y correcciones de bugs se fusionan aquí para la preparación de la próxima versión.

2. **Ramas de Soporte**

   - **feature**: Usadas para desarrollar nuevas características. Se crean a partir de `develop` y se fusionan de vuelta a `develop` cuando están completas.
   - **release**: Usadas para preparar una nueva versión. Se crean a partir de `develop` y se fusionan tanto en `develop` como en `master` cuando están listas. También se puede usar para aplicar correcciones menores antes del lanzamiento.
   - **hotfix**: Usadas para corregir errores críticos en producción. Se crean a partir de `master` y se fusionan tanto en `develop` como en `master`.

Flujo de Trabajo

1. **Crear una Rama de Característica (feature)**

   - Crear una rama a partir de `develop`

```
git checkout develop
git checkout -b feature/nueva-caracteristica
```

   - Trabajar en la nueva característica y, cuando esté completa, fusionarla de vuelta en `develop`

```
git checkout develop
git merge feature/nueva-caracteristica
git branch -d feature/nueva-caracteristica
```

2. **Crear una Rama de Lanzamiento (release)**

   - Crear una rama a partir de `develop` cuando estés listo para preparar una nueva versión

```
git checkout develop
git checkout -b release/1.0.0
```

   - Realizar las pruebas y ajustes necesarios. Una vez que esté listo para el lanzamiento, fusionarla tanto en `master` como en `develop`

```
git checkout master
git merge release/1.0.0
git tag -a 1.0.0
git checkout develop
git merge release/1.0.0
git branch -d release/1.0.0
```

3. **Crear una Rama de Corrección Urgente (hotfix)**

- Crear una rama a partir de `master` cuando necesites arreglar un problema crítico en producción

```
git checkout master
git checkout -b hotfix/1.0.1
```

- Realizar la corrección, luego fusionarla tanto en `master` como en `develop`

```
git checkout master
git merge hotfix/1.0.1
git tag -a 1.0.1
git checkout develop
git merge hotfix/1.0.1
git branch -d hotfix/1.0.1
```

Ventajas de Gitflow

- **Organización**: Facilita la estructuración y organización del código, diferenciando claramente entre el desarrollo de nuevas características y la estabilización de lanzamientos.
- **Control de Calidad**: Al tener ramas dedicadas para el desarrollo y las correcciones urgentes, se mejora el control de calidad y se minimizan los errores en producción.
- **Facilita la Colaboración**: Los equipos pueden trabajar en paralelo en diferentes características y correcciones sin interferir en el trabajo de otros.

Desventajas de Gitflow

- **Complejidad**: Puede ser excesivamente complejo para proyectos pequeños o equipos que no requieren tanta estructura.
- **Requiere Disciplina**: Todos los miembros del equipo deben seguir rigurosamente el flujo de trabajo para evitar problemas de integración.

GITFLOW.

Inicialización de Git Flow

Antes de usar Git Flow, debes inicializarlo en tu repositorio. Esto configura las ramas principales y define los prefijos para las ramas de soporte.

```
git flow init
```

Durante la inicialización, se te pedirá que confirmes los nombres de las ramas y prefijos. Los valores predeterminados suelen ser suficientes

- Rama principal de producción: `master`
- Rama de desarrollo: `develop`
- Prefijo para ramas de características: `feature/`
- Prefijo para ramas de lanzamientos: `release/`
- Prefijo para ramas de correcciones urgentes: `hotfix/`
- Prefijo para ramas de soporte: `support/`

Comandos de Git Flow

Una vez que Git Flow está inicializado, puedes usar varios comandos específicos para gestionar el flujo de trabajo. Aquí tienes una descripción de los comandos principales

Características (features)

- **Iniciar una nueva característica**

```
git flow feature start nombre-caracteristica
```

- **Finalizar una característica**

```
git flow feature finish nombre-caracteristica
```

- **Publicar una característica para que otros puedan trabajar en ella**

```
git flow feature publish nombre-caracteristica
```

Lanzamientos (releases)

- **Iniciar un nuevo lanzamiento**

```
git flow release start 1.0.0
```

- **Finalizar un lanzamiento**

```
git flow release finish 1.0.0
```

Correcciones urgentes (hotfixes)

- **Iniciar una corrección urgente**

```
git flow hotfix start 1.0.1
```

- **Finalizar una corrección urgente**

```
git flow hotfix finish 1.0.1
```

Flujo de Trabajo con Git Flow

A continuación, un ejemplo de cómo podría verse el flujo de trabajo utilizando Git Flow

1. **Crear una nueva característica**

```
git flow feature start login-feature
```

Trabaja en la característica y, una vez completada

```
git flow feature finish login-feature
```

2. **Preparar un lanzamiento**

```
git flow release start 1.1.0
```

Realiza los ajustes y pruebas necesarios. Luego, finaliza el lanzamiento

```
git flow release finish 1.1.0
```

3. **Corregir un error urgente en producción**

```
git flow hotfix start 1.1.1
```

Aplica la corrección y finaliza el hotfix

```
git flow hotfix finish 1.1.1
```

Ventajas de Usar Git Flow

- **Simplificación del Proceso**: Los comandos específicos de Git Flow simplifican el manejo de ramas y aseguran que se sigan las mejores prácticas del flujo de trabajo.
- **Consistencia**: Mantiene un proceso consistente y estructurado para todos los miembros del equipo.
- **Automatización**: Muchas tareas, como el etiquetado de lanzamientos y la fusión de ramas, se automatizan, reduciendo el margen de error.

EQUIVALENCIAS DE LOS COMANDOS "GIT FLOW" CON EL "GIT" CONVENCIONAL.

Inicialización

Git Flow

```
git flow init
```

Git

```
# Crear ramas 'develop' y establecer prefijos de ramas manualmente.
git checkout -b develop
# No hay un comando Git directo para establecer prefijos, esto es más una convención de nombres.
```

Características (features)

Iniciar una nueva característica

Git Flow

```
git flow feature start nombre-caracteristica
```

Git

```
git checkout develop
git checkout -b feature/nombre-caracteristica
```

Finalizar una característica

Git Flow

```
git flow feature finish nombre-caracteristica
```

Git

```
git checkout develop
git merge --no-ff feature/nombre-caracteristica
git branch -d feature/nombre-caracteristica
```

Lanzamientos (releases)

Iniciar un nuevo lanzamiento

Git Flow

```
git flow release start 1.0.0
```

Git

```
git checkout develop
git checkout -b release/1.0.0
```

Finalizar un lanzamiento

Git Flow

```
git flow release finish 1.0.0
```

Git

```
git checkout master
git merge --no-ff release/1.0.0
git tag -a 1.0.0
git checkout develop
git merge --no-ff release/1.0.0
git branch -d release/1.0.0
```

Correcciones urgentes (hotfixes)

Iniciar una corrección urgente

Git Flow

```
git flow hotfix start 1.0.1
```

Git

```
git checkout master
git checkout -b hotfix/1.0.1
```

Finalizar una corrección urgente

Git Flow

```
git flow hotfix finish 1.0.1
```

Git

```
git checkout master
git merge --no-ff hotfix/1.0.1
git tag -a 1.0.1
git checkout develop
git merge --no-ff hotfix/1.0.1
git branch -d hotfix/1.0.1
```

Publicar y obtener cambios

Publicar una característica

Git Flow

```
git flow feature publish nombre-caracteristica
```

Git

```
git checkout feature/nombre-caracteristica
git push origin feature/nombre-caracteristica
```

Obtener una característica publicada por otro

Git Flow

```
git flow feature publish nombre-caracteristica
```

Git

```
git fetch origin
git checkout -b feature/nombre-caracteristica origin/feature/nombre-caracteristica
```