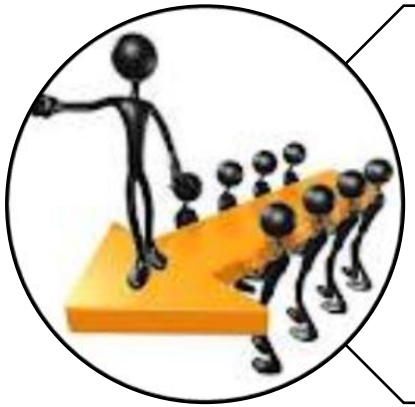
Universidad Continental

Construcción de software

Ingeniería de Sistemas e Informática

Propósito y contenido de la sesión



Propósito de la sesión

- Implementa un proyecto para gestionar las tareas utilizando el flujo de trabajo Gitflow, la metodología de desarrollo guiado por pruebas y el lenguaje de programación Python..

Contenido de la sesión

- Flujo de trabajo Gitflow y desarrollo guiado por pruebas.

Cinco Git Workflows para mejorar nuestros proyectos

Elabore un resumen de 50 a 100 palabras de cada uno de los cinco Git workflows.

Enlace: https://oscaraguadoweb.com/git/cinco-git-workflows-para-mejorar-nuestros-proyectos/.

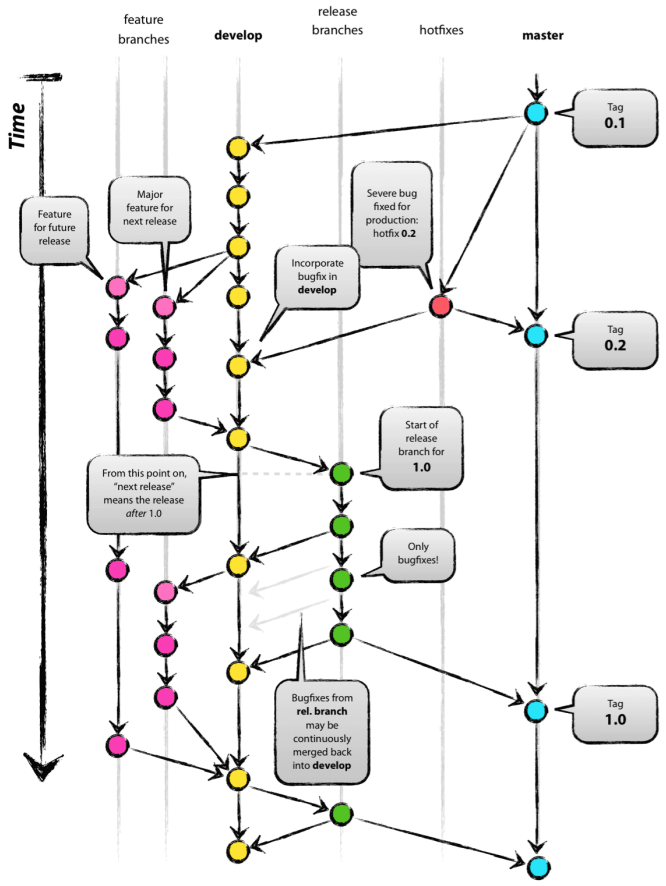
Universidad Continental

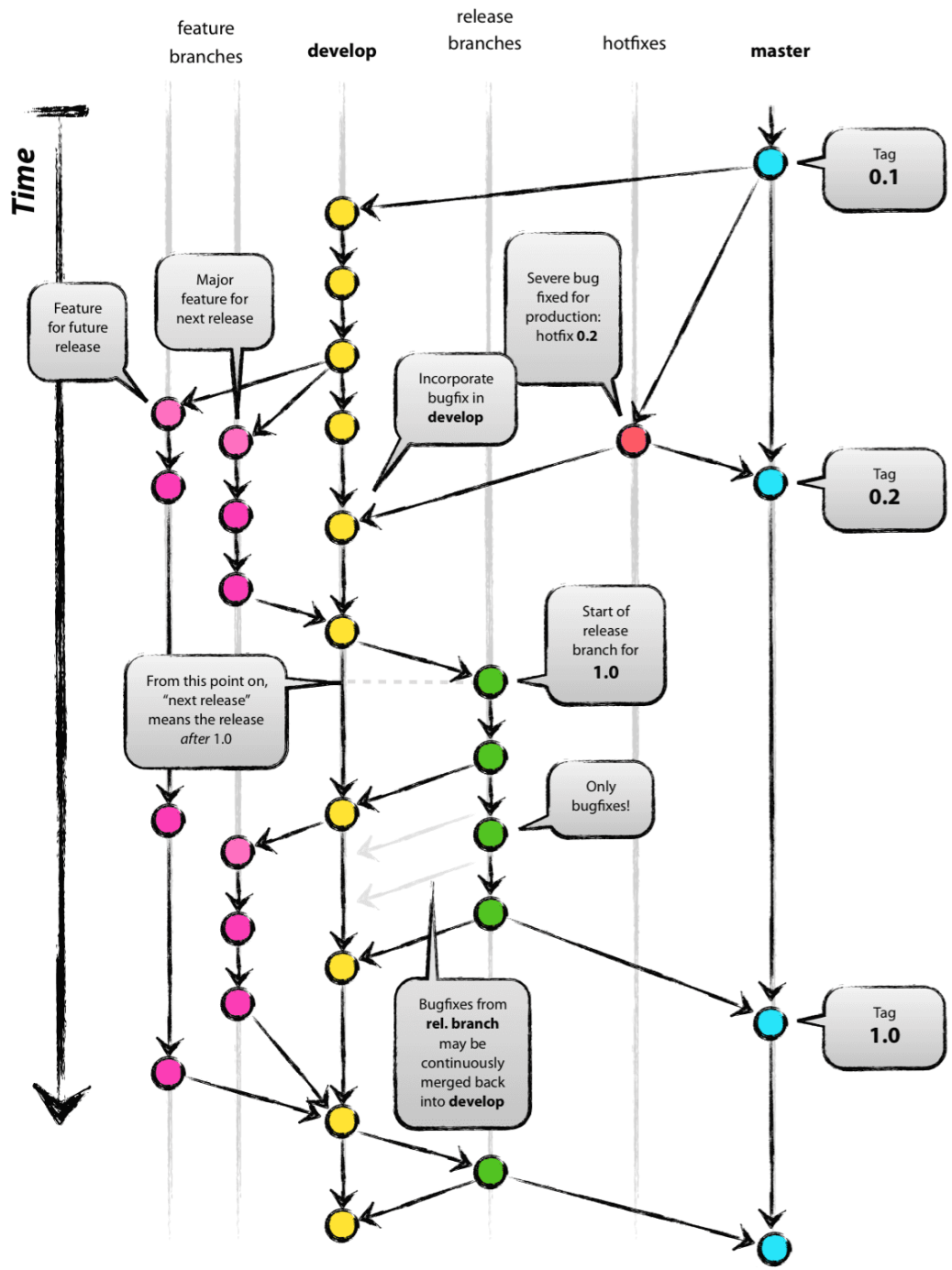
Git Flow

Flujo de trabajo

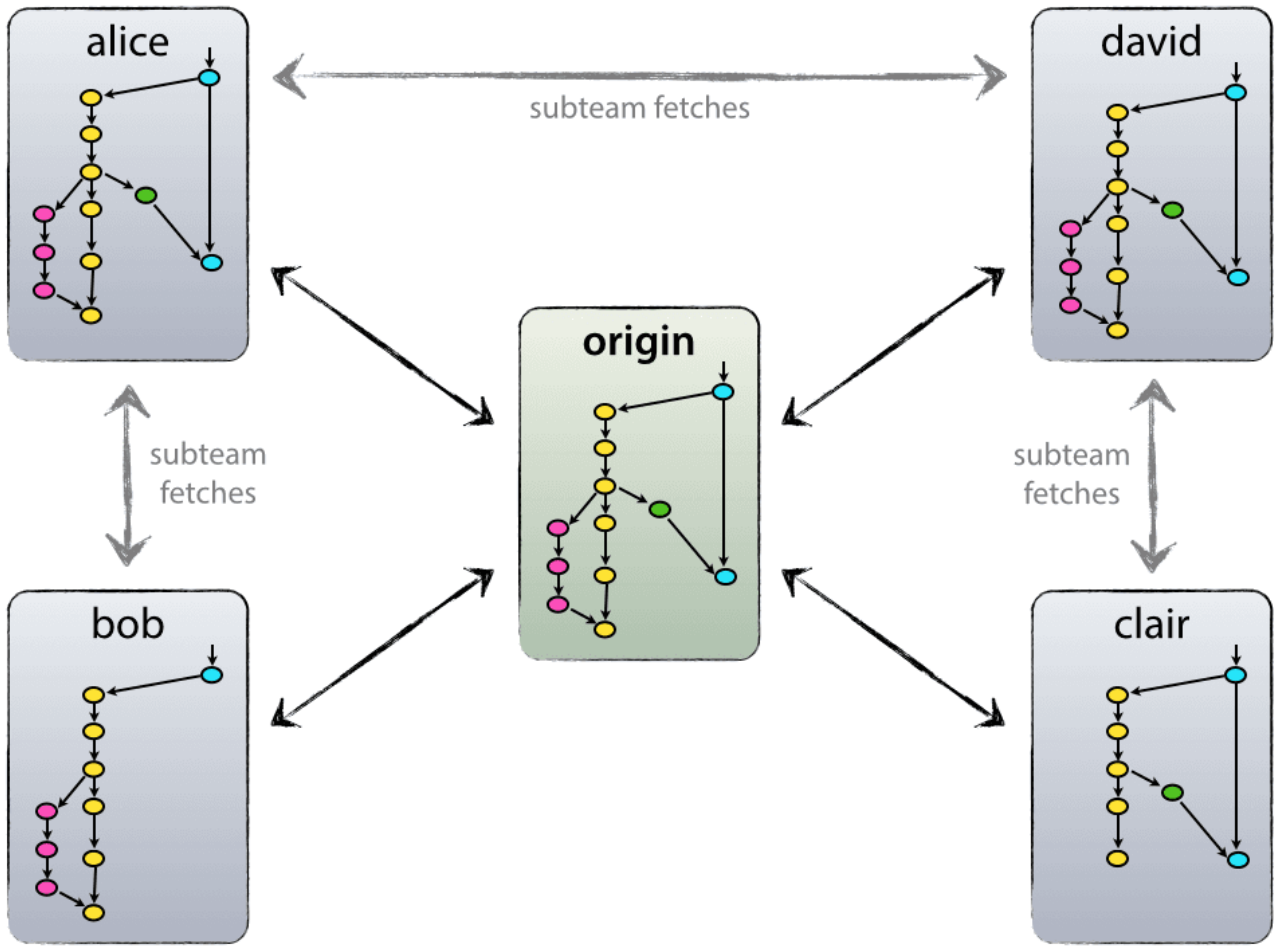
Gitflow

- GitFlow – Vincent Drissen (2010)
  - https://nvie.com/posts/a-successful-git-branching-model/




feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Repositorios



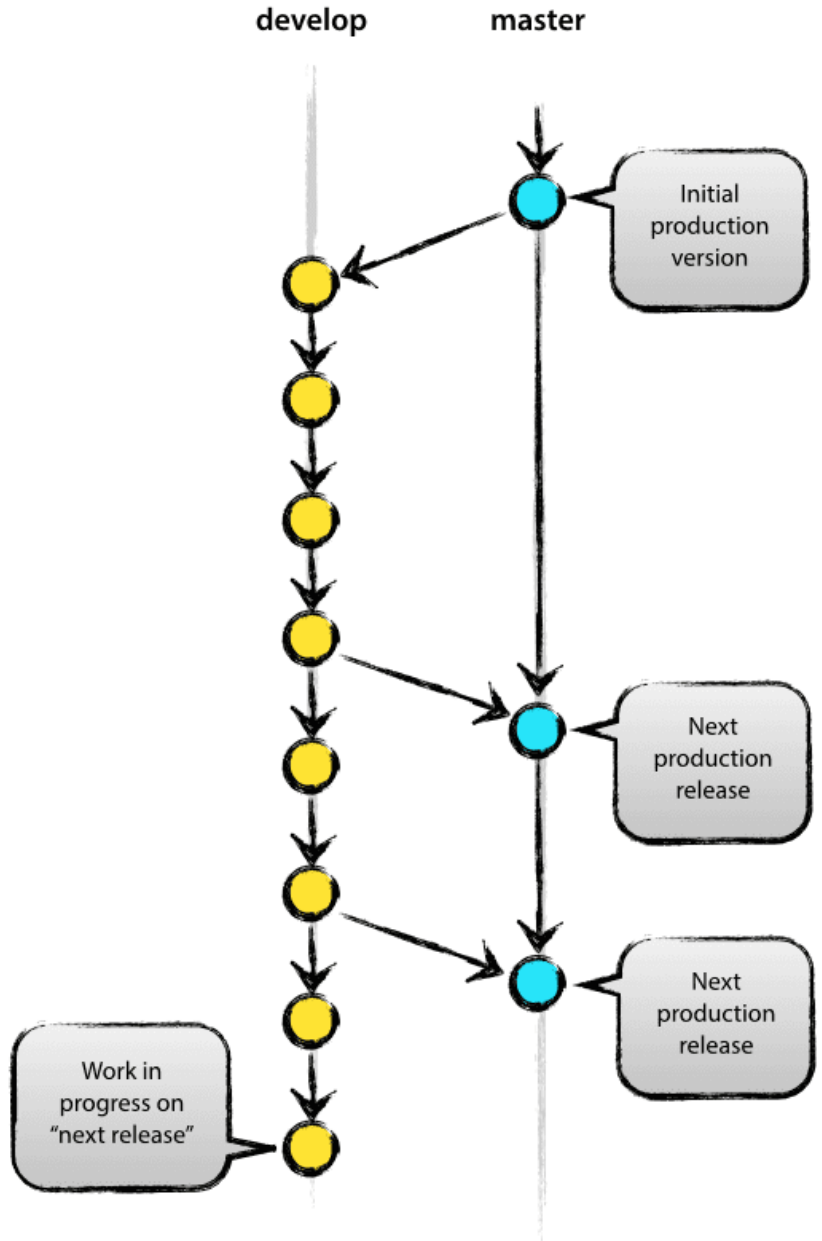
Ramas permanentes

Master (main)

- Versiones liberadas del producto

Development

- Evolución del producto: Para las ramas feature.


develop
master
Initial production version
Next production release
Next production release
Work in progress on "next release"

Ramas temporales

Feature

- Para cada nueva característica o funcionalidad del producto, historia detallada del producto.
- Se derivan de la rama development el momento que se inicia el desarrollo de un feauture
- Se integran en development cuando se termina su implementación
- Los desarrolladores trabajan en las ramas locales, y hacen: git add, git commit y git push; de las mismas al repositorio remoto del producto.

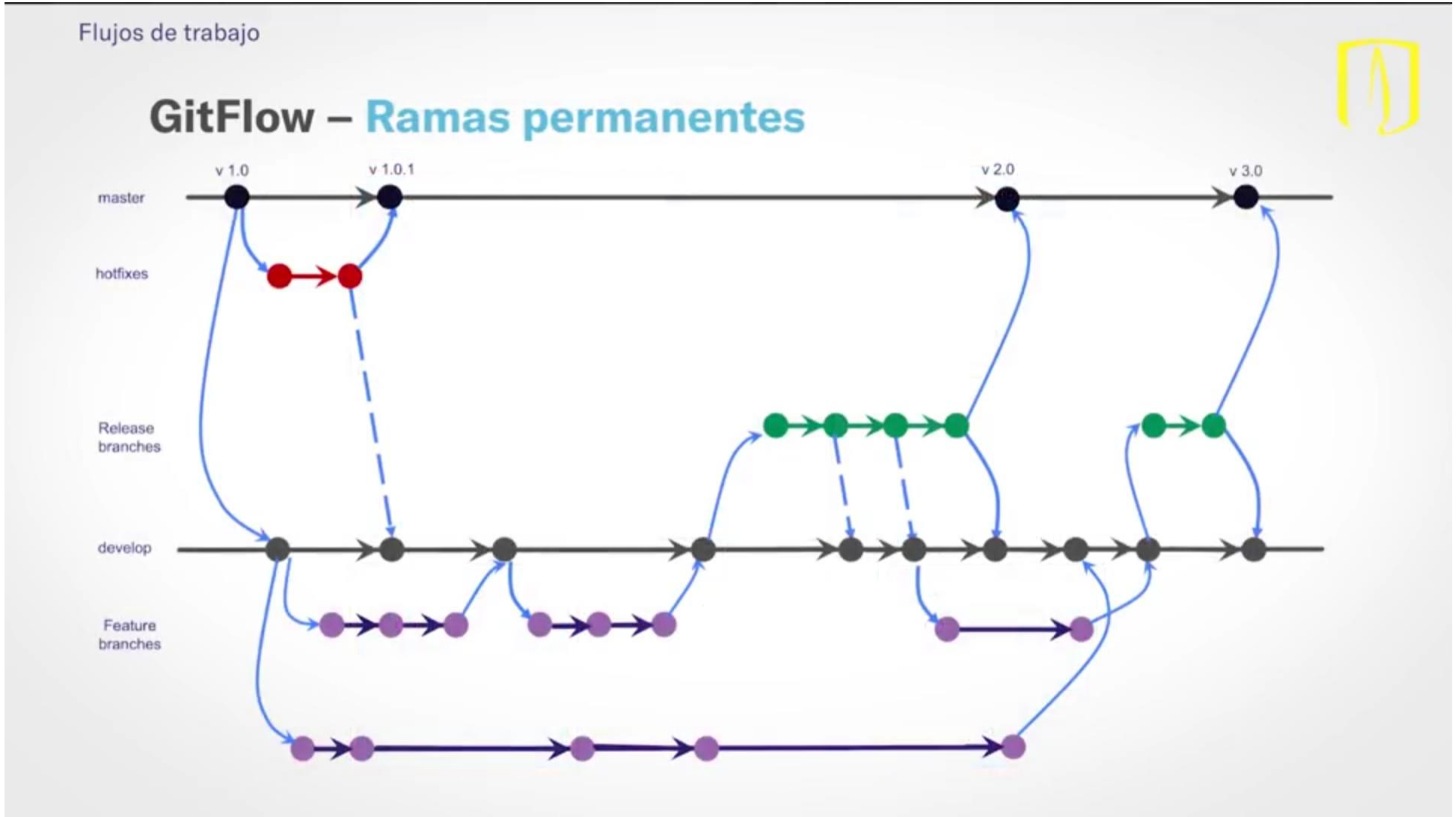
Release

- Cuando un feature está completamente implementada se integra con la rama development y master, pero antes se corren las pruebas y se corrigen los conflictos.

Hotfix

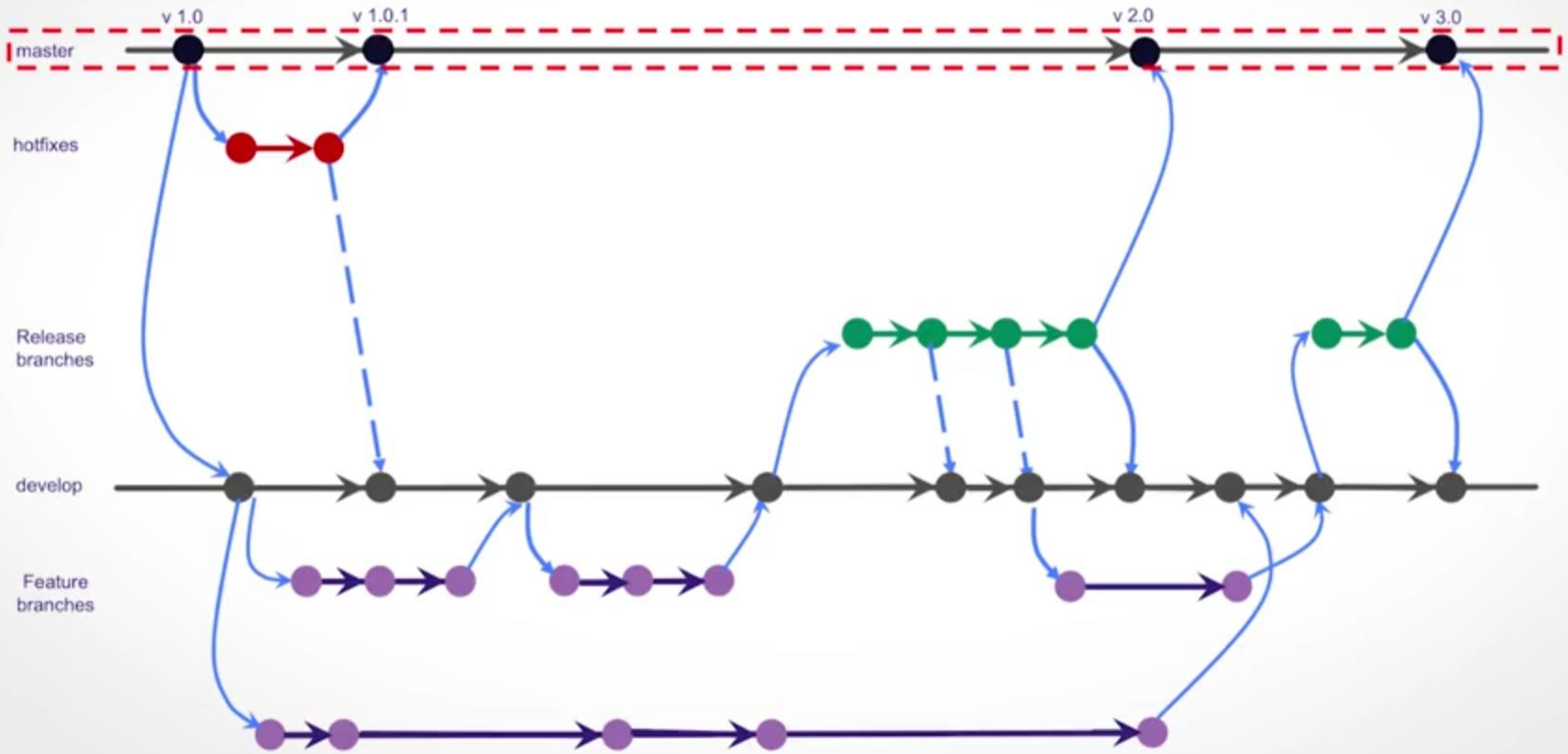
- Cuando se detecta un defecto se crea una rama hotfix para corregirla.

Flujos de trabajo



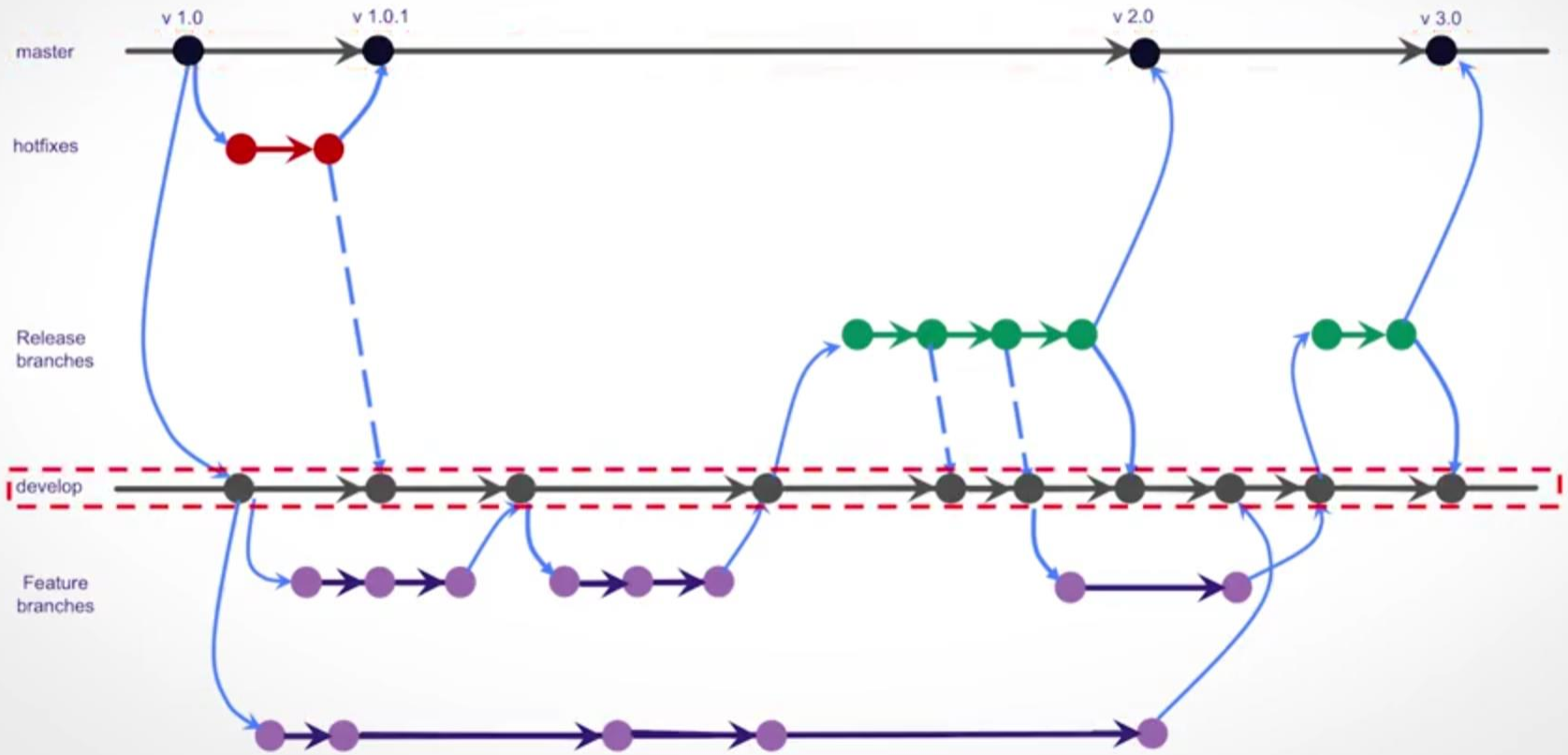
Flujos de trabajo

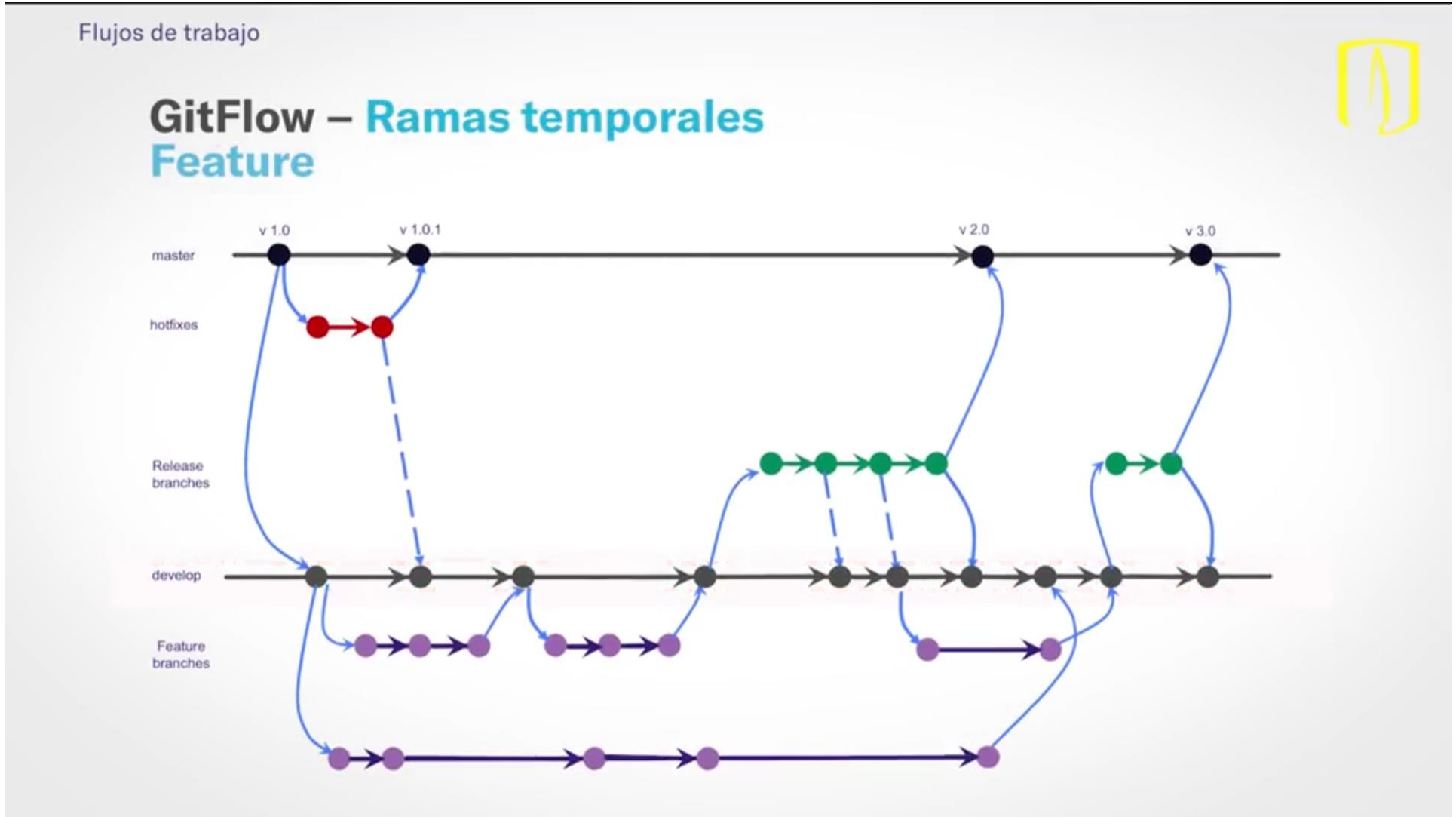
GitFlow – Ramas permanentes



Flujos de trabajo

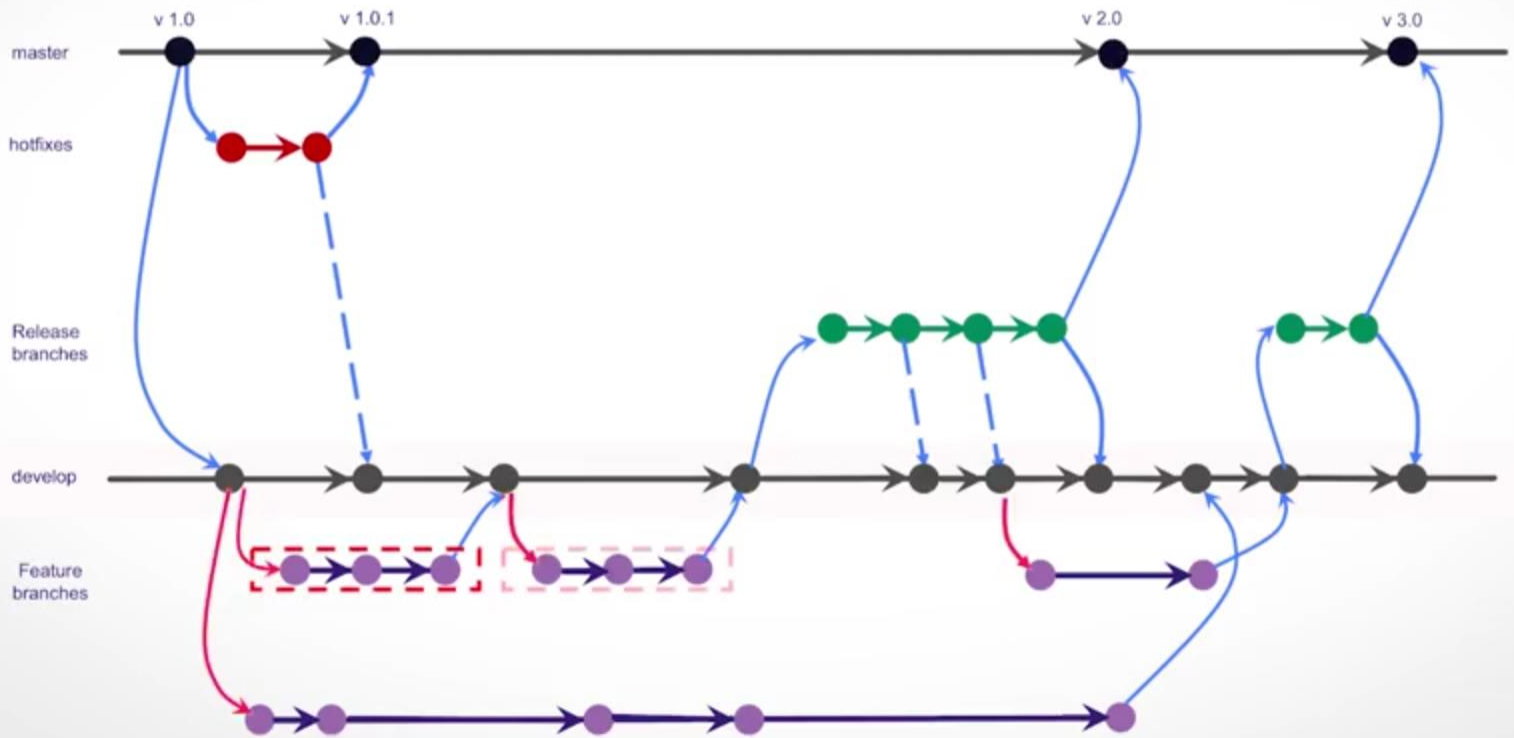
GitFlow – Ramas permanentes



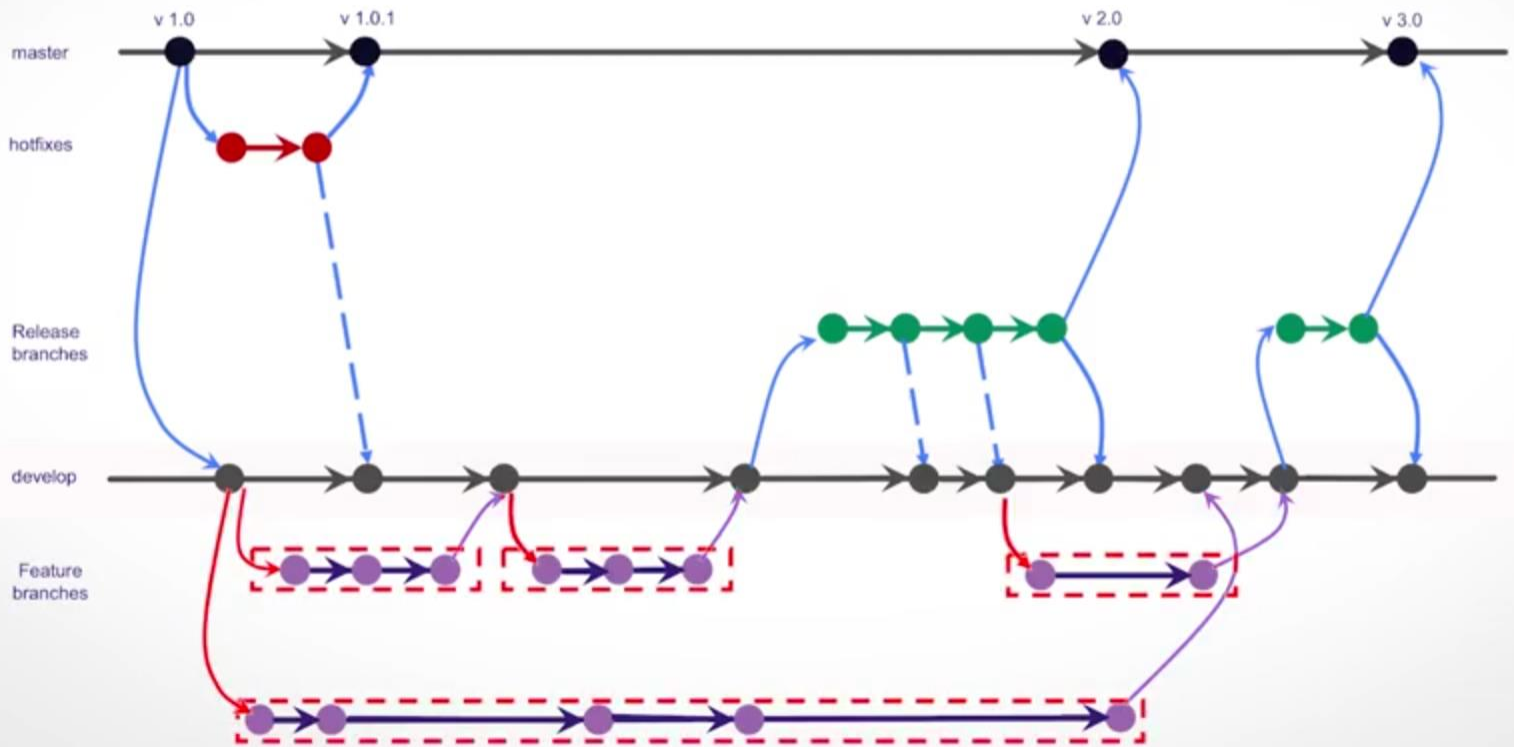


GitFlow – Ramas temporales
Feature

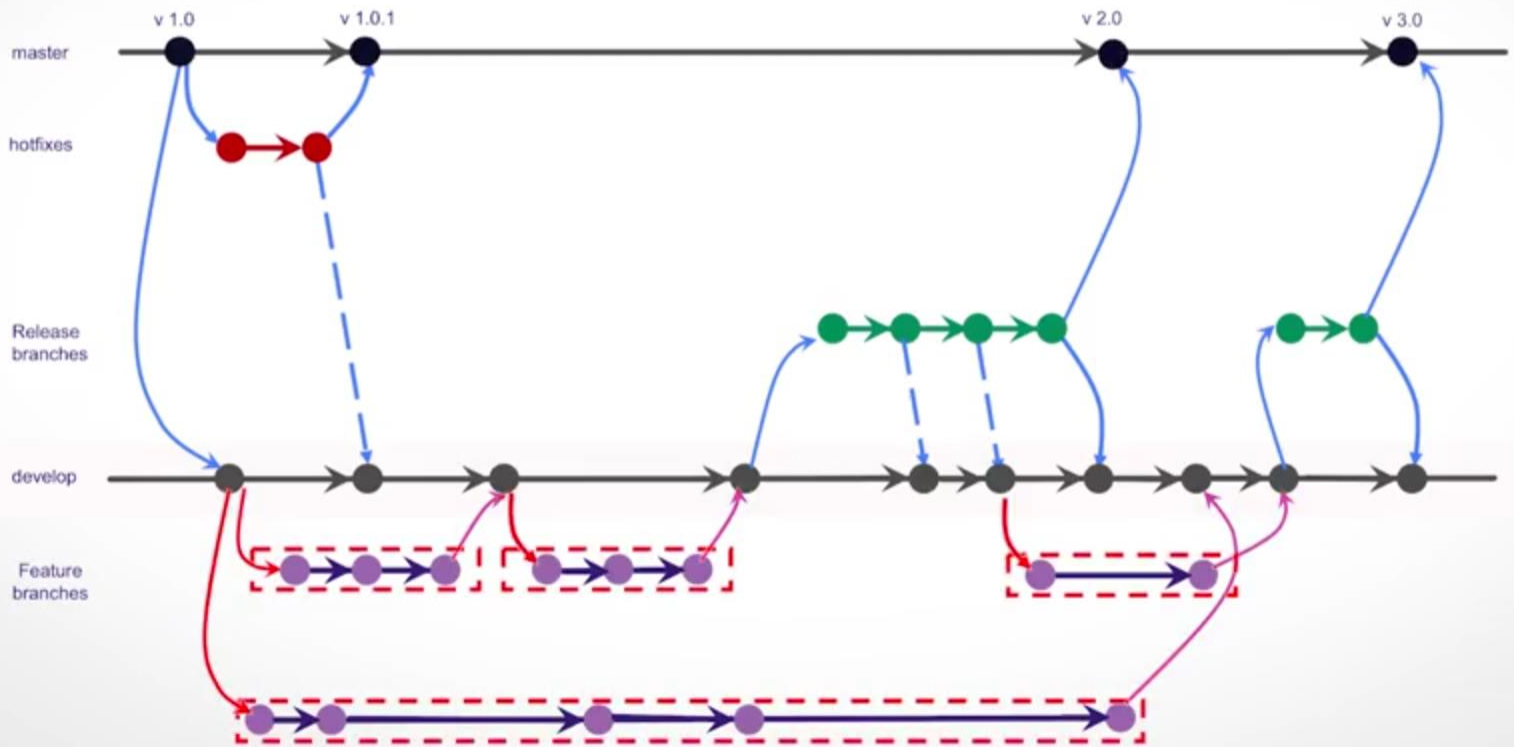
GitFlow – Ramas temporales
Feature

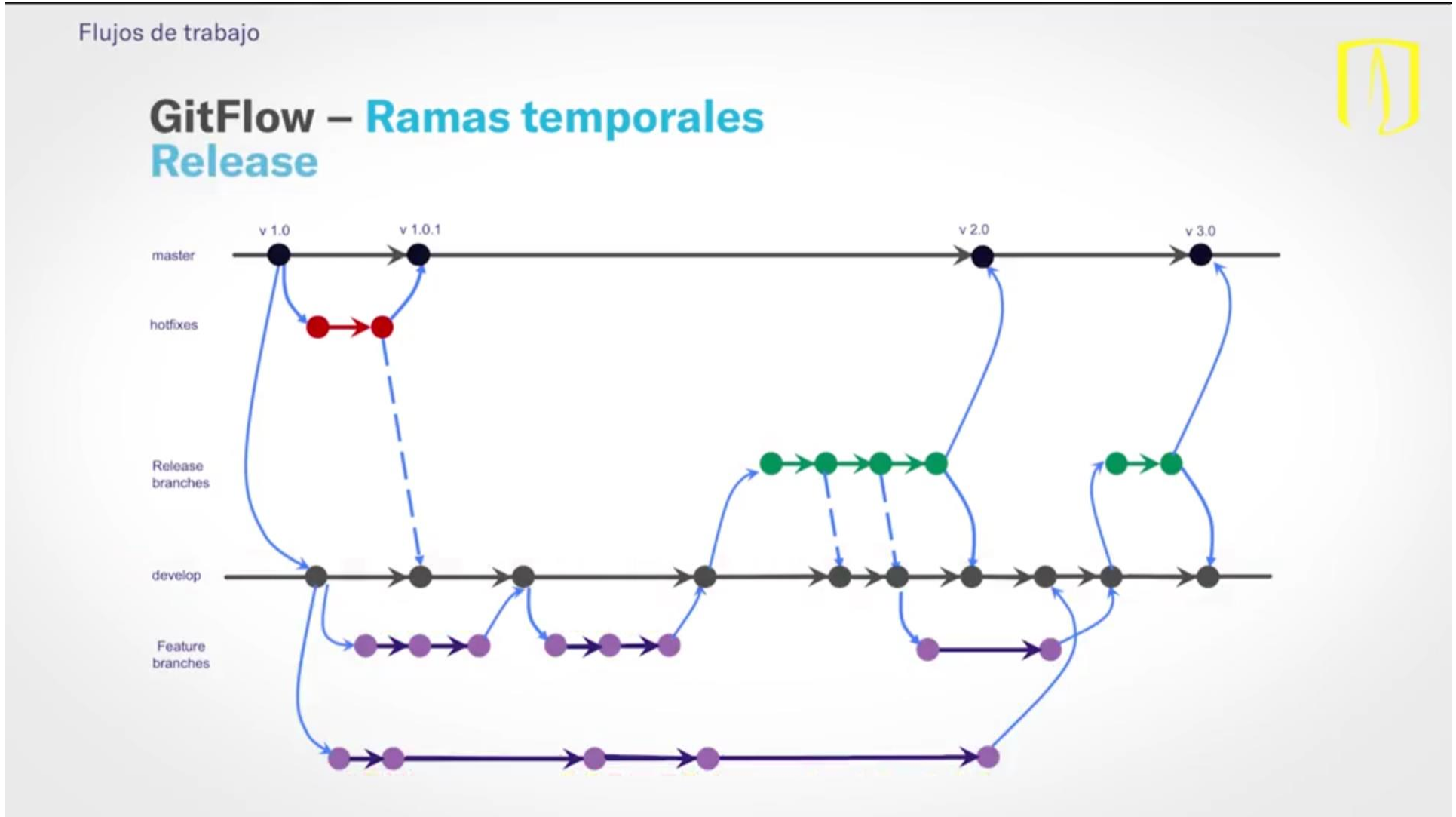


GitFlow – Ramas temporales
Feature



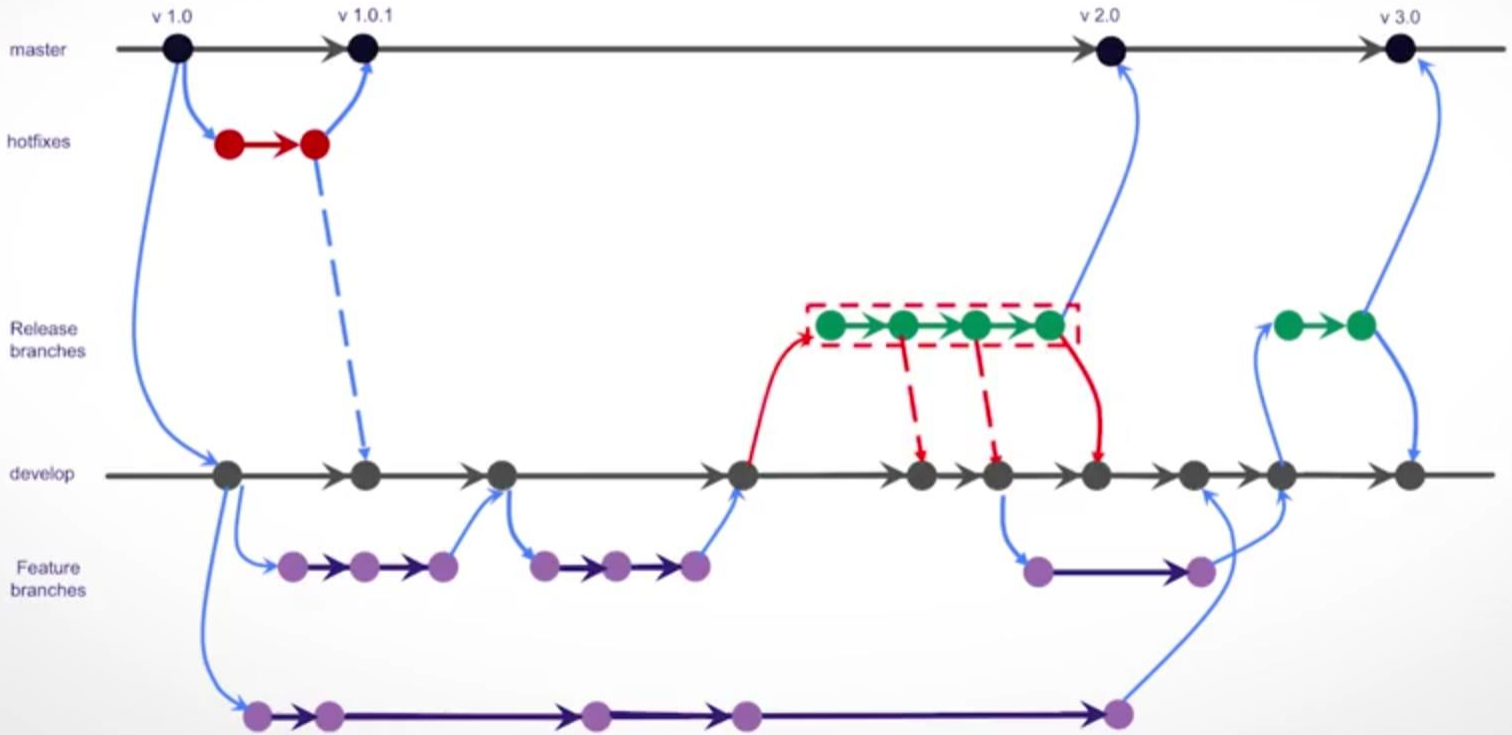
GitFlow – Ramas temporales
Feature



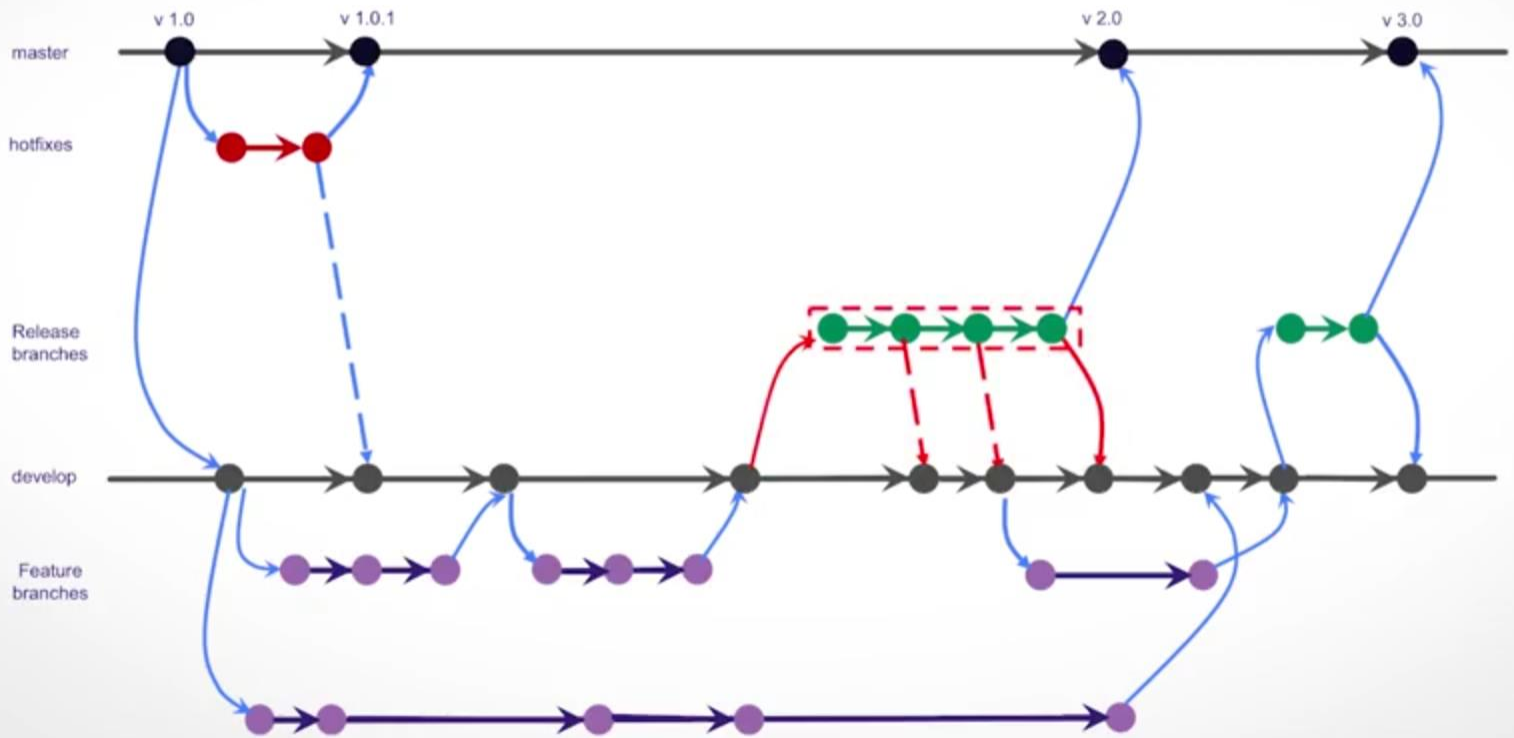


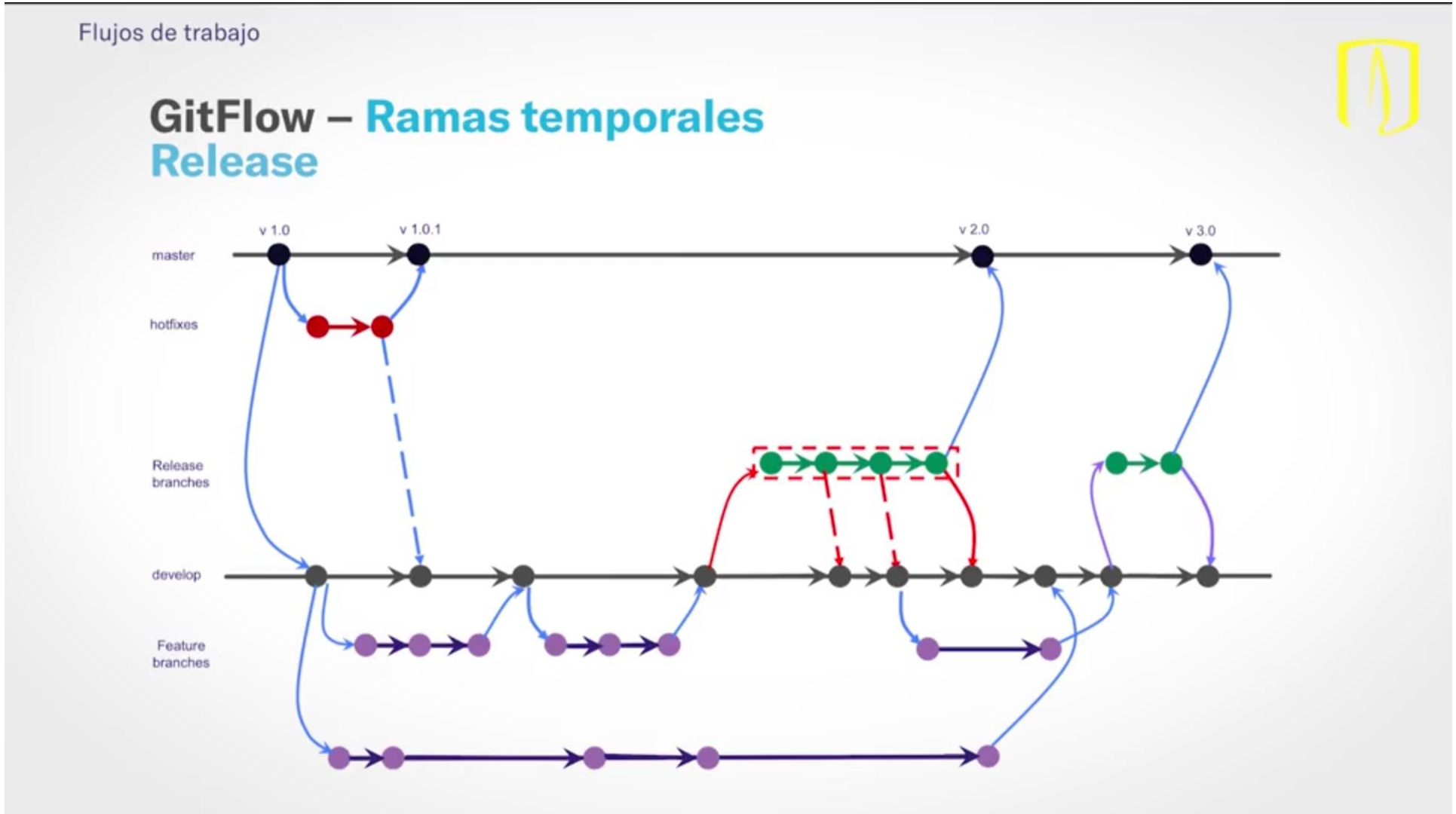
GitFlow – Ramas temporales
Release

GitFlow – Ramas temporales
Release

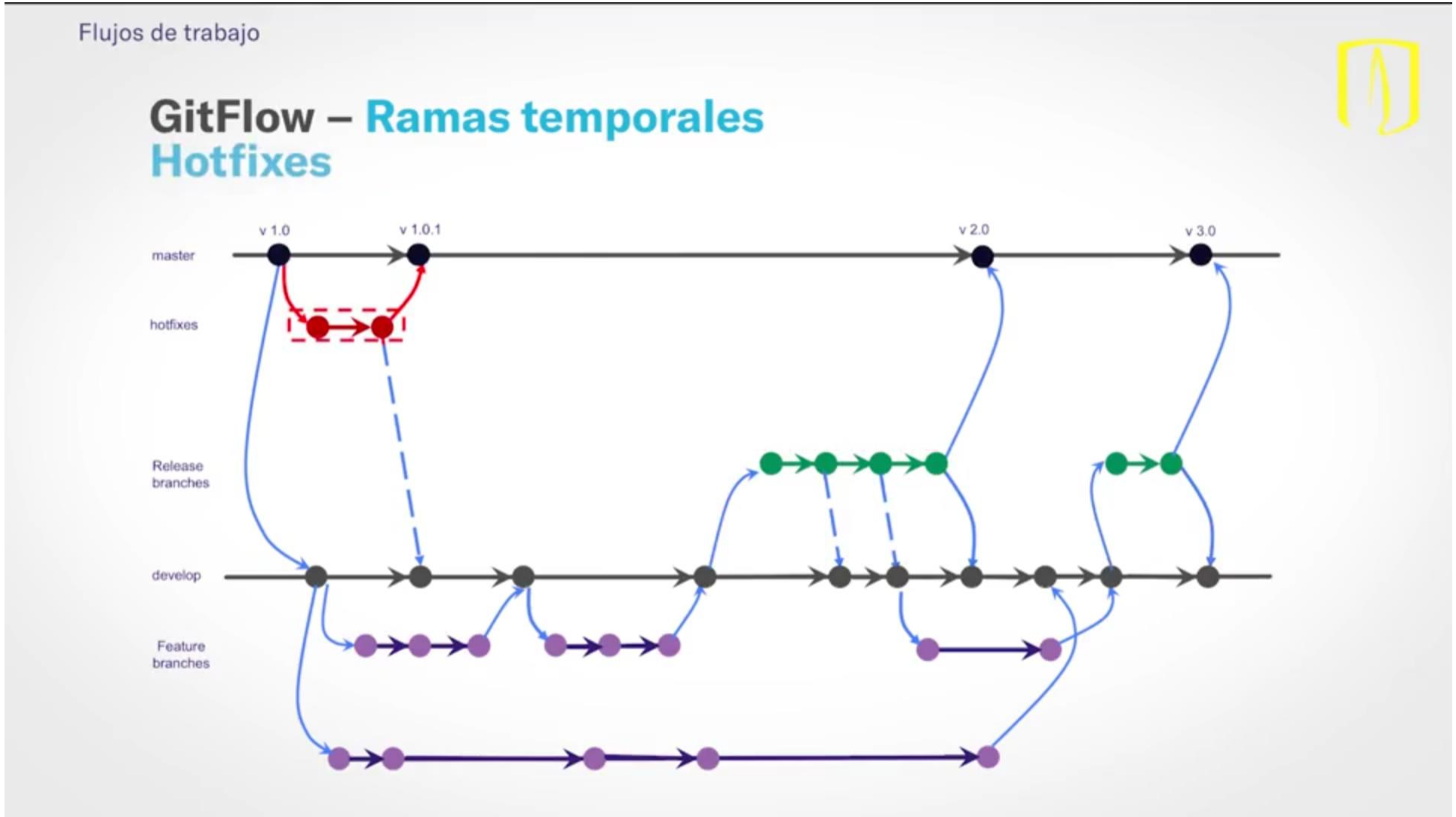


GitFlow – Ramas temporales
Release

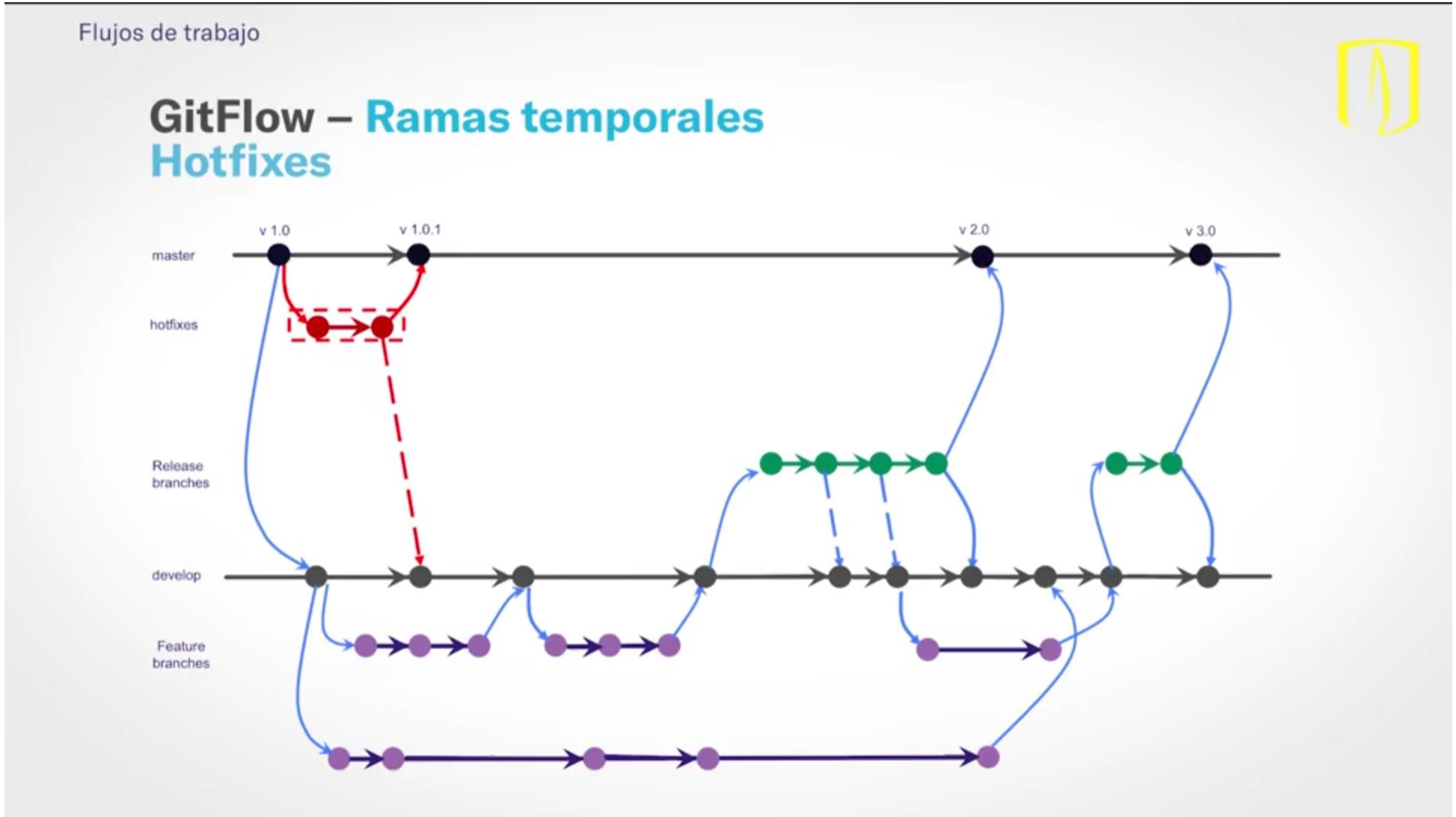




GitFlow – Ramas temporales
Release



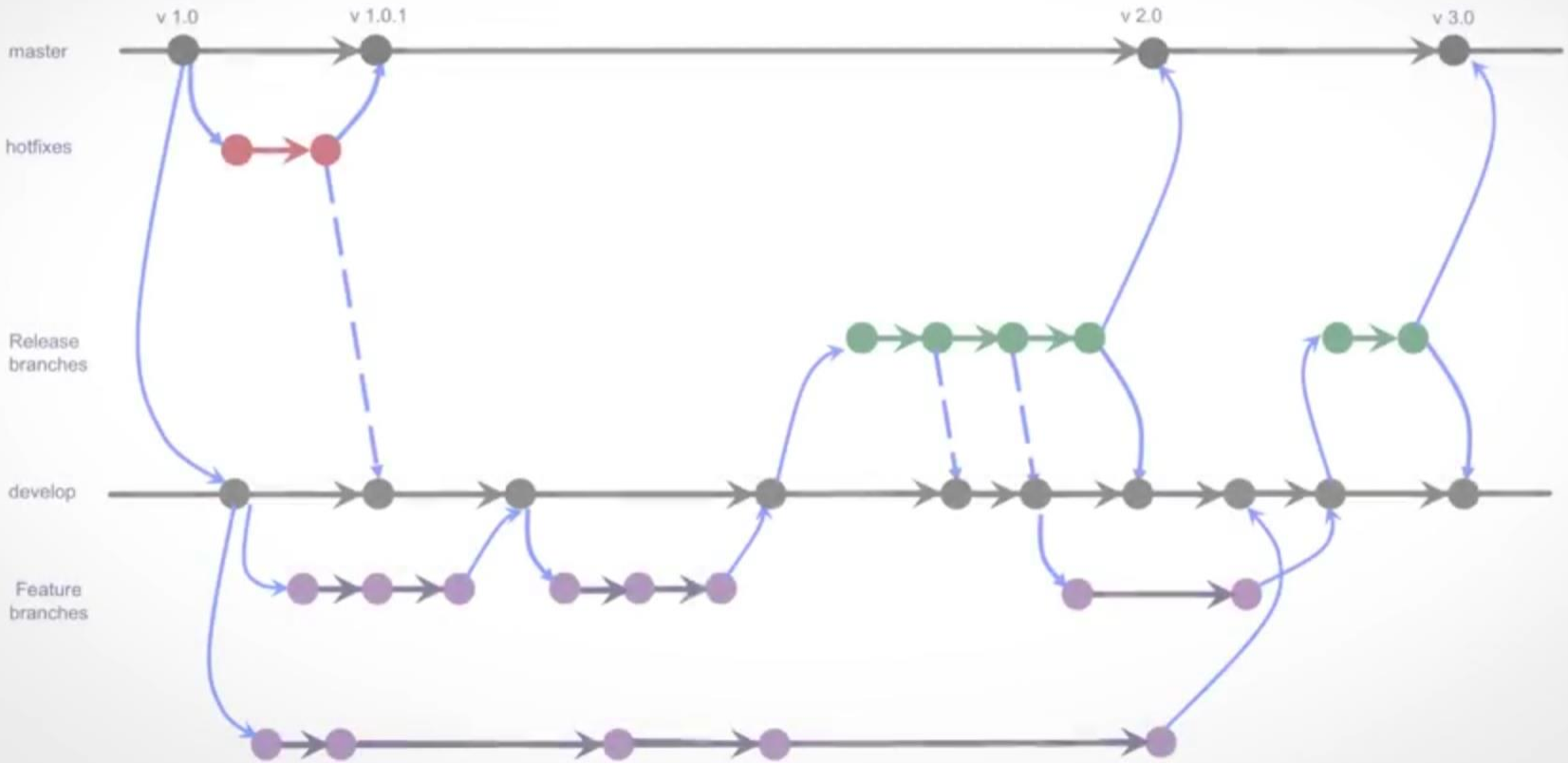
GitFlow – Ramas temporales
Hotfixes



GitFlow – Ramas temporales
Hotfixes

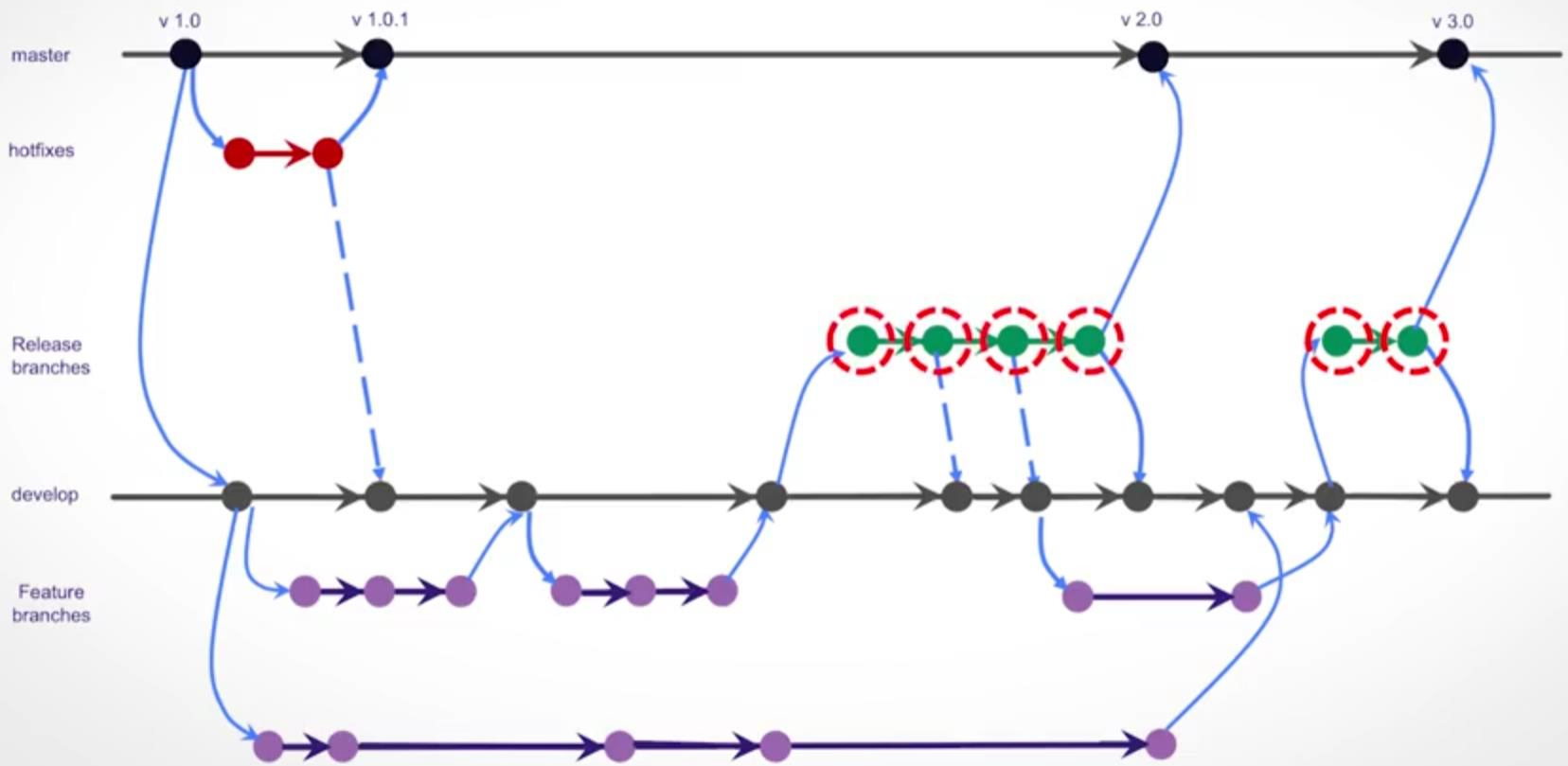
Flujos de trabajo

GitFlow – Ambientes



Flujos de trabajo

GitFlow – Ambientes



Enlaces de interés

Aprende Git de manera sencilla: Git Flow

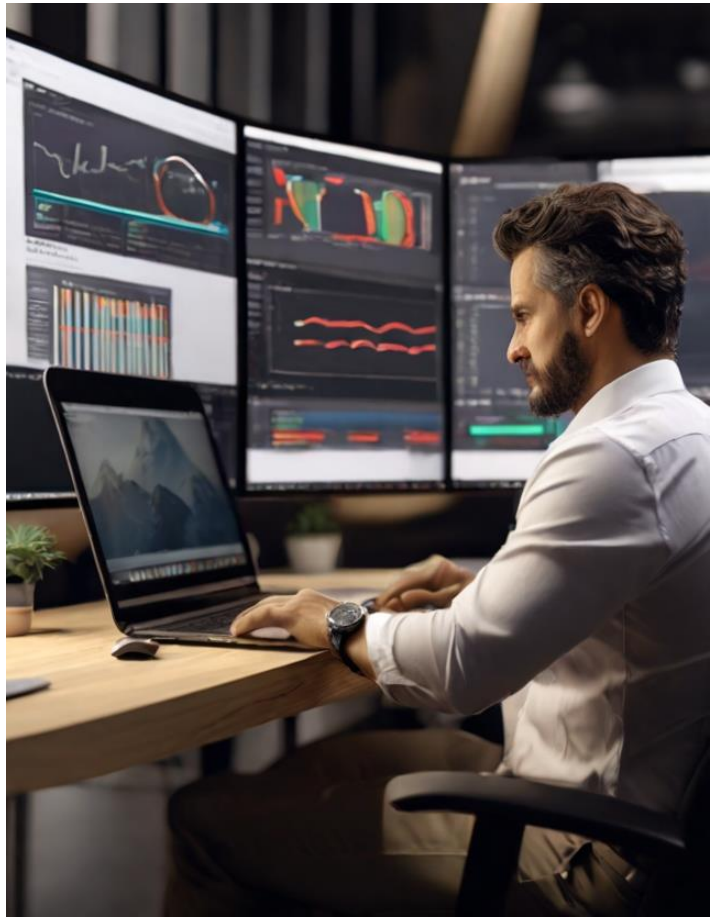
- https://desarrollowp.com/blog/tutoriales/aprende-git-de-manera-sencilla-git-flow/

Git Flow explicado en 10 minutos | 4SoftwareDevelopers

- https://www.youtube.com/watch?v=abtqhoMqCWY

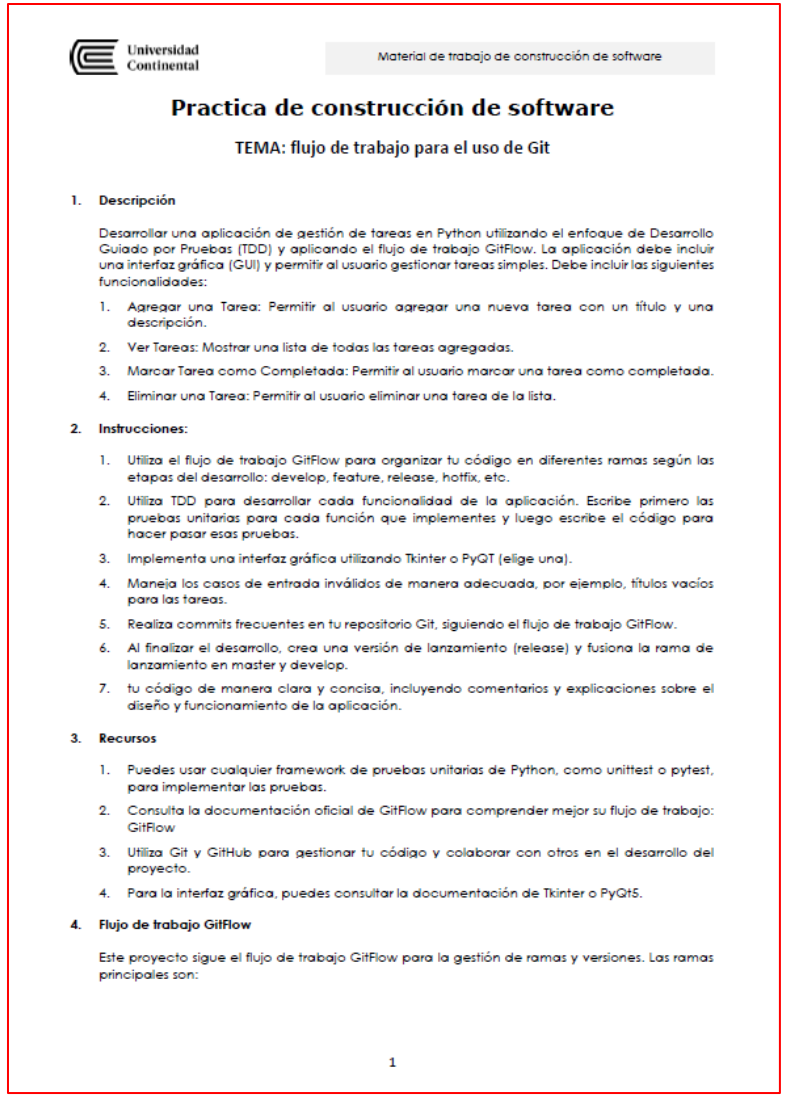
GitFlow en Github

- https://www.youtube.com/watch?v=LkYWop93S70&t=20s



Desarrollo de una Aplicación de Gestión de Tareas en Python

Gestión de tareas



Universidad Continental

Material de trabajo de construcción de software

Practica de construcción de software

TEMA: flujo de trabajo para el uso de Git

1. **Descripción**

   Desarrollar una aplicación de gestión de tareas en Python utilizando el enfoque de Desarrollo Guiado por Pruebas (TDD) y aplicando el flujo de trabajo GitFlow. La aplicación debe incluir una interfaz gráfica (GUI) y permitir al usuario gestionar tareas simples. Debe incluir las siguientes funcionalidades:

   1. Agregar una Tarea: Permitir al usuario agregar una nueva tarea con un título y una descripción.
   2. Ver Tareas: Mostrar una lista de todas las tareas agregadas.
   3. Marcar Tarea como Completada: Permitir al usuario marcar una tarea como completada.
   4. Eliminar una Tarea: Permitir al usuario eliminar una tarea de la lista.

2. **Instrucciones:**

   1. Utiliza el flujo de trabajo GitFlow para organizar tu código en diferentes ramas según las etapas del desarrollo: develop, feature, release, hotfix, etc.
   2. Utiliza TDD para desarrollar cada funcionalidad de la aplicación. Escribe primero las pruebas unitarias para cada función que implementes y luego escribe el código para hacer pasar esas pruebas.
   3. Implementa una interfaz gráfica utilizando Tkinter o PyQT (elige una).
   4. Maneja los casos de entrada inválidos de manera adecuada, por ejemplo, títulos vacíos para las tareas.
   5. Realiza commits frecuentes en tu repositorio Git, siguiendo el flujo de trabajo GitFlow.
   6. Al finalizar el desarrollo, crea una versión de lanzamiento (release) y fusiona la rama de lanzamiento en master y develop.
   7. tu código de manera clara y concisa, incluyendo comentarios y explicaciones sobre el diseño y funcionamiento de la aplicación.

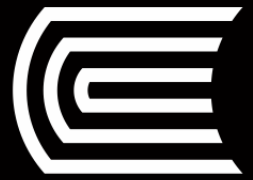
3. **Recursos**

   1. Puedes usar cualquier framework de pruebas unitarias de Python, como unittest o pytest, para implementar las pruebas.
   2. Consulta la documentación oficial de GitFlow para comprender mejor su flujo de trabajo: GitFlow
   3. Utiliza Git y GitHub para gestionar tu código y colaborar con otros en el desarrollo del proyecto.
   4. Para la interfaz gráfica, puedes consultar la documentación de Tkinter o PyQt5.

4. **Flujo de trabajo GitFlow**

   Este proyecto sigue el flujo de trabajo GitFlow para la gestión de ramas y versiones. Las ramas principales son:

1



ucontinental.edu.pe