Networked Git

POLITECNICO OPEN
unix LABS
Come hack with us.

Such Repository

Perchè utilizzare i repository remoti?

- Vogliamo condividere e/o mantenere i backup di un progetto creato in locale su una delle tante piattaforme (e.g. GitLab),
- Vogliamo scaricare e /o contribuire al sorgente di un progetto

Such Repository

Per poter collaborare ad altri repo si necessita di strumenti che permettono di:

- Pubblicare i nostri repository
- Accedere alle modifiche dei collaboratori
- Gestire i branch remoti di sviluppo

Ma prima...

Bisogna introdurre almeno due concetti usati quando si parla di Git!

HEAD

Con il termine **head** si indica, generalmente, un puntatore ad un commit.
Quando si parla del puntatore all'ultimo commit si indica con **HEAD**, in maiuscolo.

Remote

I remotes sono i repository con cui possiamo sincronizzare il nostro repo locale.
Possiamo avere più remotes, anche da servizi differenti, in cui possiamo fare solo operazioni di push, solo di pull o entrambe.
Anche i remotes hanno l'HEAD, che punta all'ultimo commit presente sul branch.

Git remote

Per gestire i repository remote si utilizza il comando `git remote`:

- **Aggiungere** un repository remoto
  `git remote add <nome> <url>`
- **Eliminare** repository remoto
  `git remote rm <nome>`
- **Rinominare** remote
  `git remote rename <oldname> <newname>`

Git Push

Il comando `git push` permette di trasferire i commit di uno specifico branch ad un repository remote:
`git push <remote> <branch>`
Se il branch non esiste sul server remoto viene creato.
Per inviare tutti i branch al server remoto si utilizza:
`git push <remote> --all`.
Per visualizzare quali branch esistono nei remote configurati si utilizza il comando `git branch -r`.

Git Fetch/Pull

Per recuperare i commit dai branch remoti su utilizza il comando **`git fetch`**.

```
git fetch <remote> <branch>
```

Questa operazione scarica solamente i commit ma non esegue il merge. Per farlo bisogna eseguire:

```
git merge <remote> <branch>
```

Il comando **`git pull`** esegue entrambe le operazioni, sia il fetch che il merge, con la medesima sintassi:

```
git pull <remote> <branch>
```

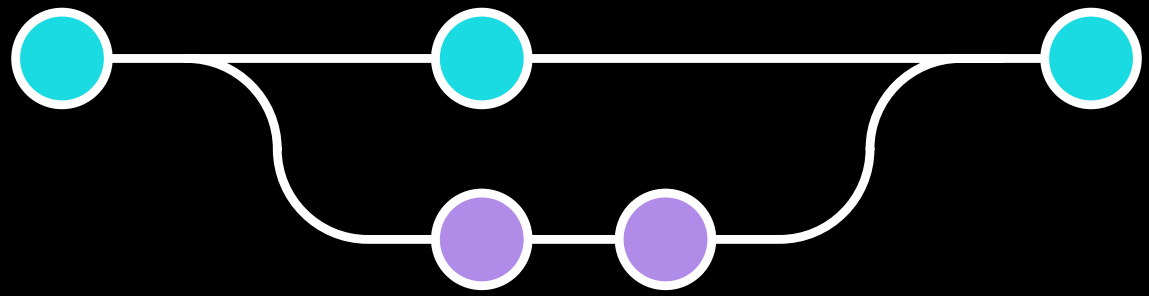
Try it yourself!

Many Workflows

Git permette di avere diversi tipi di workflows:

1. **Centralized Workflows**:
   - Featured Branch Workflow
   - Gitflow Workflow
2. **Distribuited Workflows:**
   - Integration Manager Workflow
   - Dictator and Lieutenants Workflow

Featured Branch Workflow

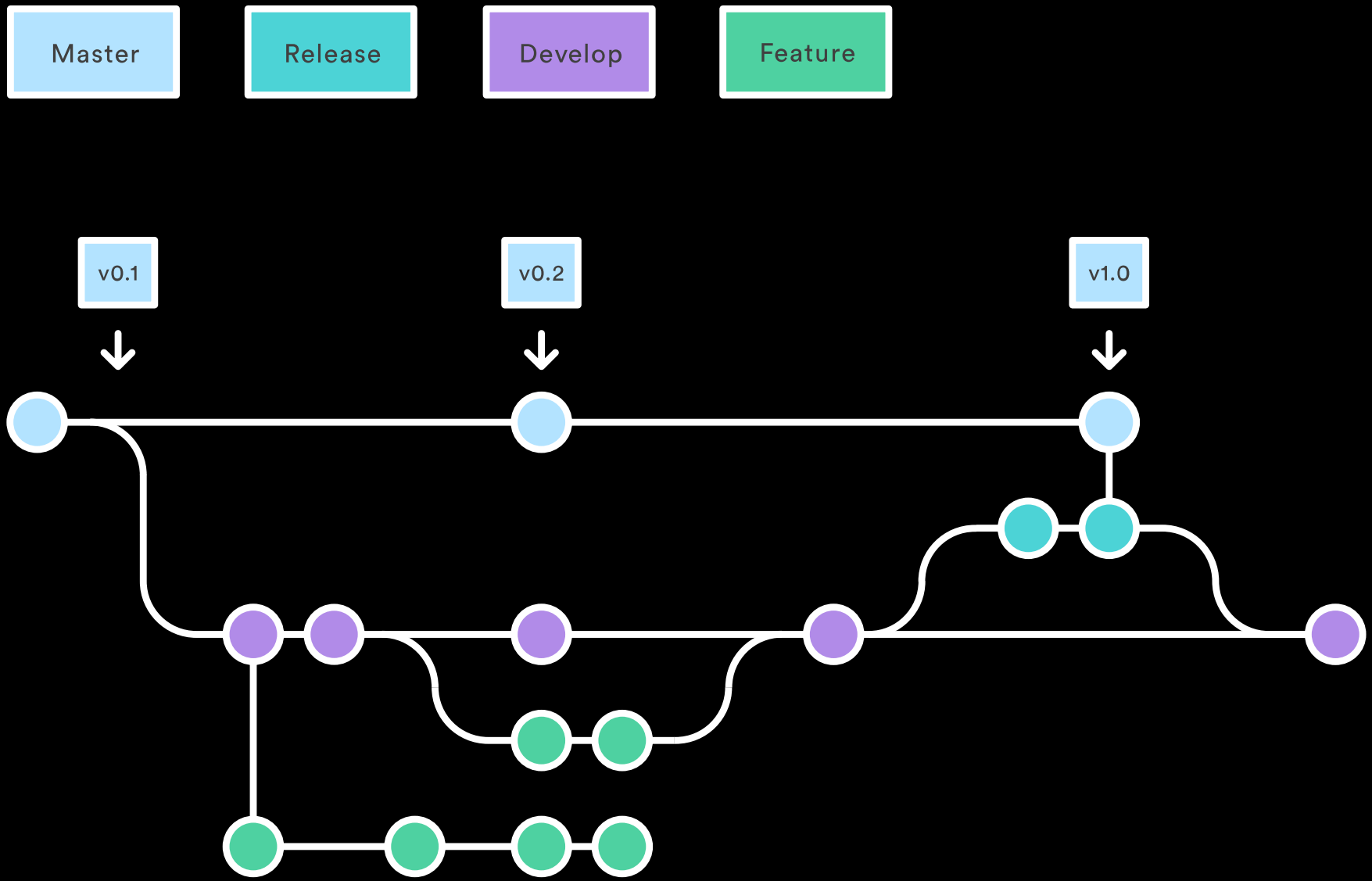


Featured Branch Workflow

Caratteristiche:

- Esiste un unico **repository ufficiale** con un **branch ufficiale** (solitamente il branch **master**)
- Ogni sviluppatore ha un proprio **repository privato** ma manda i commit a quello ufficiale
- Gli sviluppatori **non committano** direttamente nel **branch ufficiale**, ma **lavorano sui branch** creati appositamente per le modifiche introdotte
- Questi diversi branch **devono** essere presenti anche sul repository ufficiale, in modo che le modifiche siano **visibili a tutti i membri del team**

Gitflow Workflow

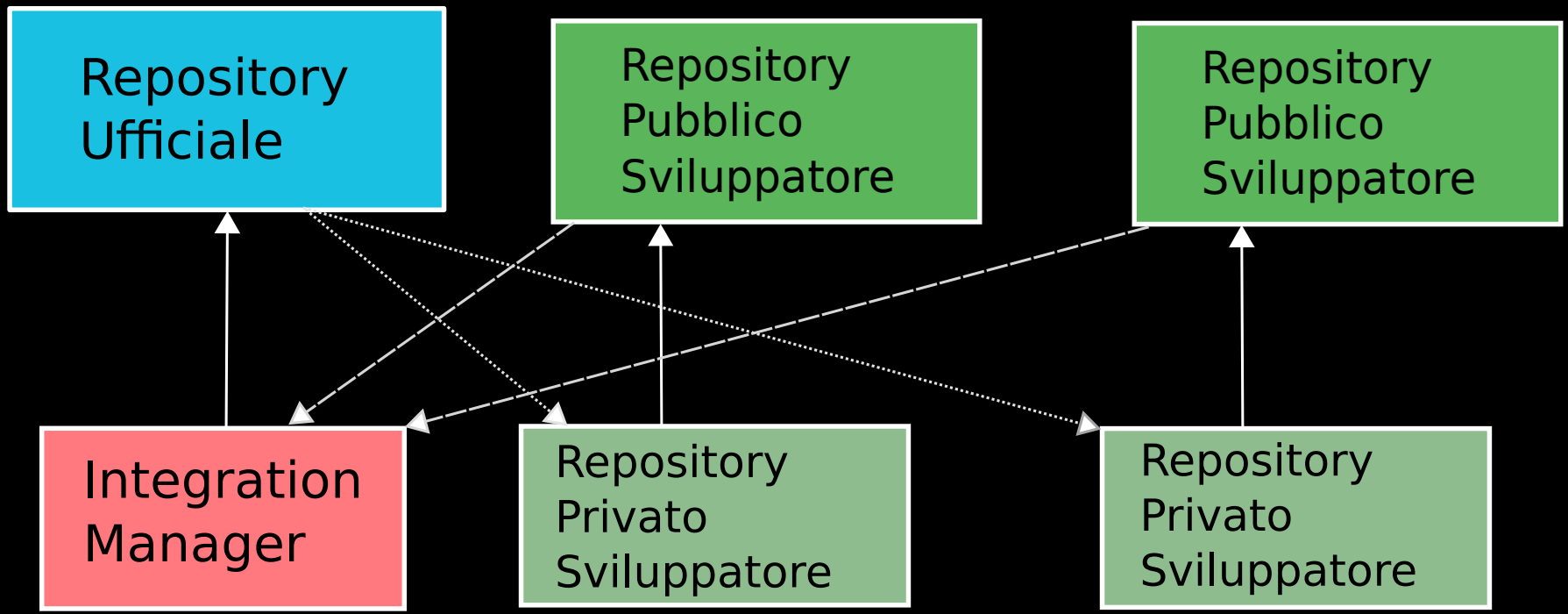


Gitflow Workflow

Caratteristiche:

- È una variante del “Featured Branch”
- L’unica vera differenza è la struttura dei branch.
- In questo workflow i branch vengo usati anche per il rilascio delle diverse versioni, delle fature o dei fix introdotti.

Integration Manager Workflow

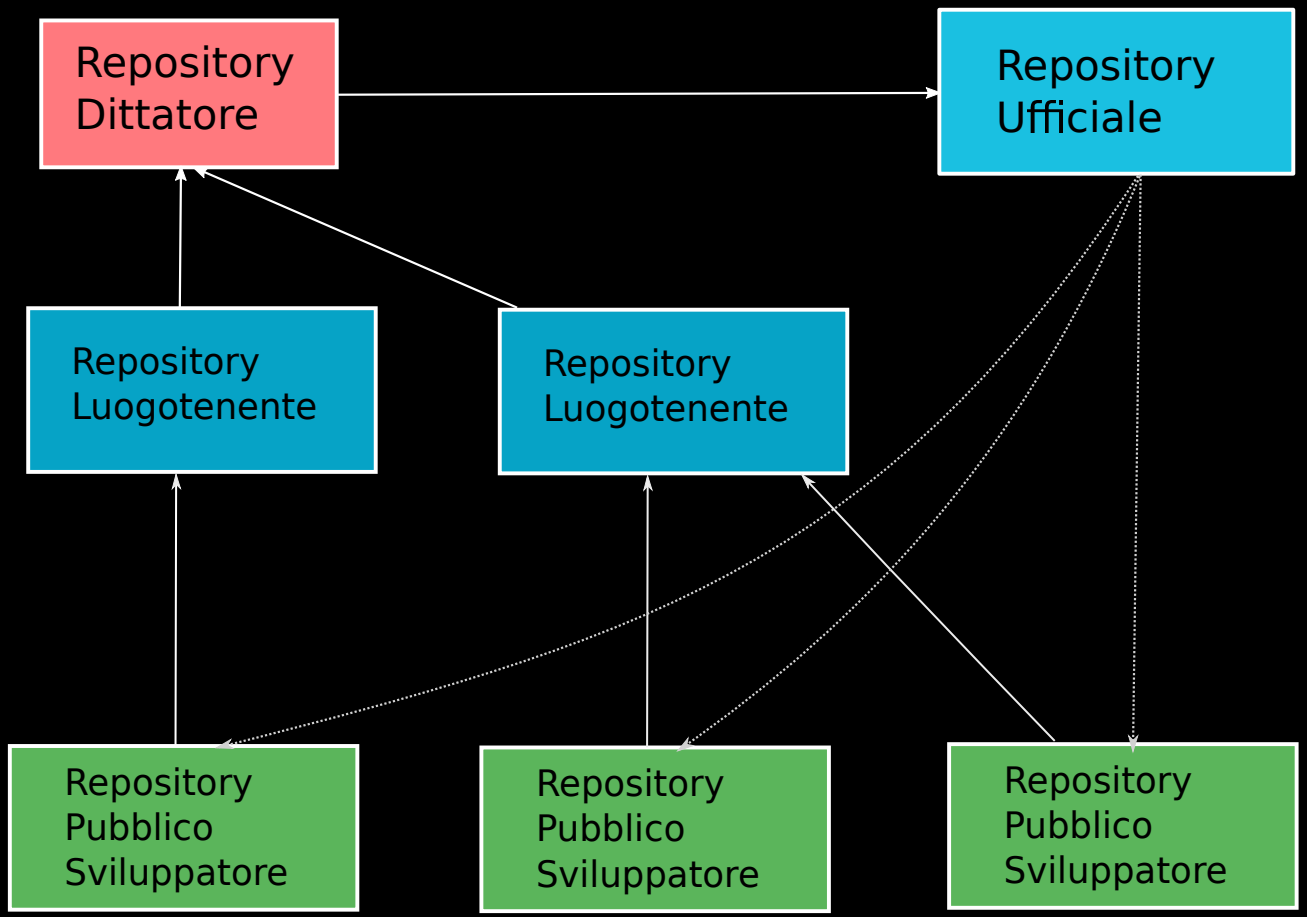


Integration Manager Workflow

Caratteristiche:

- Esiste un unico **repository ufficiale** in "**sola lettura**"
- L'unico con i **permessi di scrittura** verso questo repository è il **capo progetto**
- Ogni sviluppatore ha un proprio **repository privato** ed uno **pubblico**
- Ogni svilupatore fa delle modifiche al progetto e chiede al capo progetto di integrarle con una **pull-request**

Dictator and Lieutenants Workflow



Dictator and Lieutenants Workflow

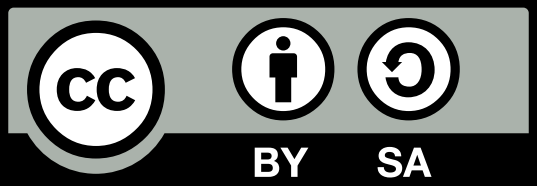
Caratteristiche:

- Esiste un unico **repository ufficiale** in "**sola lettura**"
- L'unico con i **permessi di scrittura** verso questo repository è il **capo progetto**
- Ogni sviluppatore ha un proprio **repository privato** ed uno **pubblico**
- Ci sono una o più figure intermedie dette **luogotenenti**
- Ogni svilupatore fa delle modifiche al progetto e chiede al **luogotenente** di integrarle con una **pull-request** nei loro repository
- Il dittatore integra, a sua discrezione, le modifiche dei branch dei luogotenenti nel repository ufficiale.

Fonti

- `https://git-scm.com`
- `https://www.atlassian.com/git/tutorials`
- Vecchi corsi Git: `https://poul.org`
- Google

Grazie per l'attenzione!



https://www.poul.org