Development Environment

SCM – Source Control Management

SCM challenges

◎ Allow many developers simultaneously work with same code base

◎ Maintain history of the changes

◎ Implement code review process

◎ Be part of solid CIDD workflow

Client-server vs Distributed

Client-Server
- ◎ User management from one place, SSO integration
- ◎ One DB, simple backup
- ◎ Require direct connection to the server
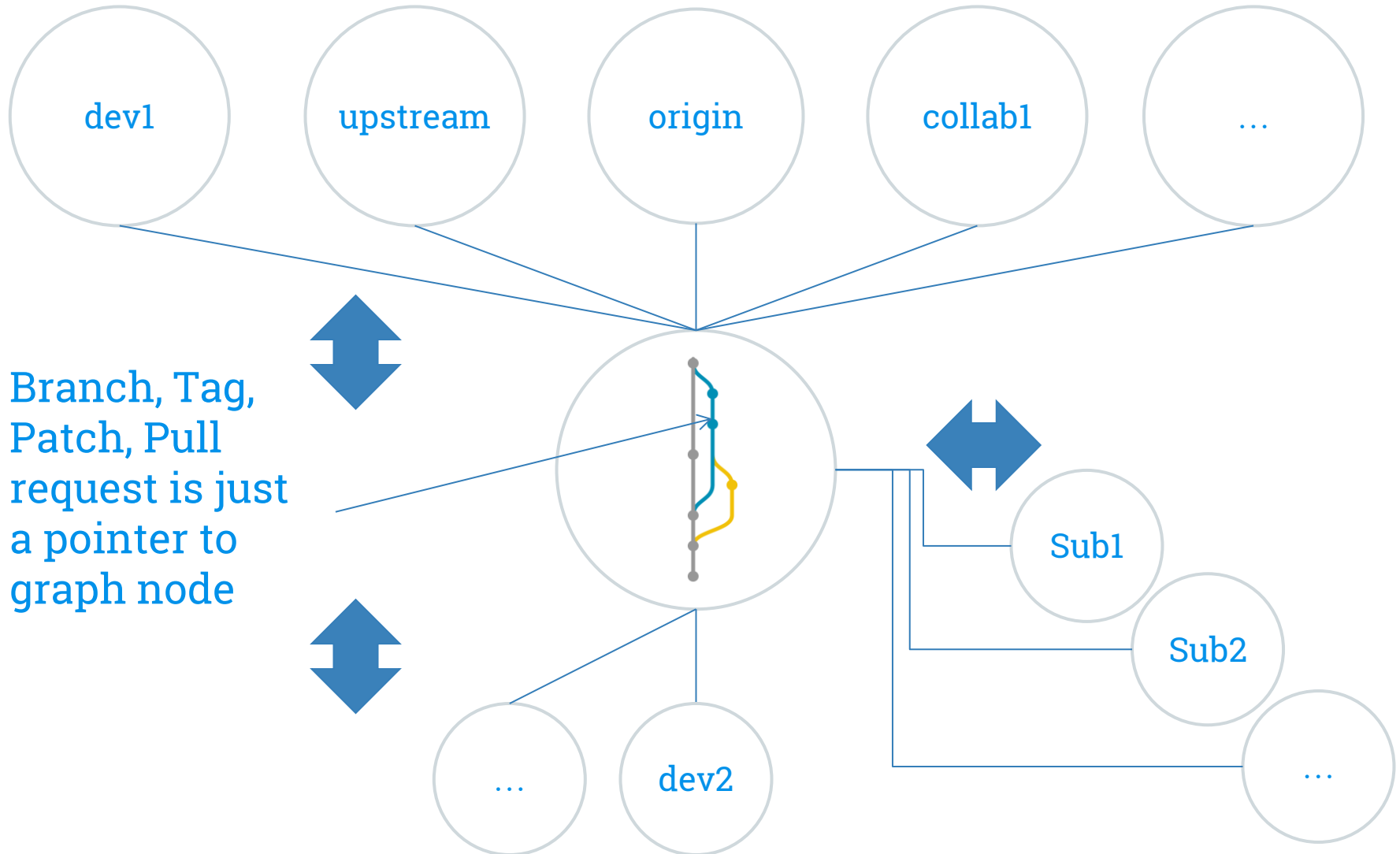- ◎ "Heavy" branching

Distributed
- ◎ SSH/HTTPS access
- ◎ Additional rules for project management
- ◎ Can work without direct access to the any server
- ◎ Simple implementing a rich number of dev workflows

https://en.m.wikipedia.org/wiki/List_of_version_control_software

Git

distributed SCM,
commits, remotes,
submodules, branches,
branching mode,
merging

Commits, Branches , Remotes, Submodules, Subtree

dev1

upstream

origin

collab1

...

Branch, Tag,
Patch, Pull
request is just
a pointer to
graph node

Sub1

Sub2

...

...

dev2

Git bare repository quick setup

https://github.com/swe-course/content/tree/master/git

Branching model, merging
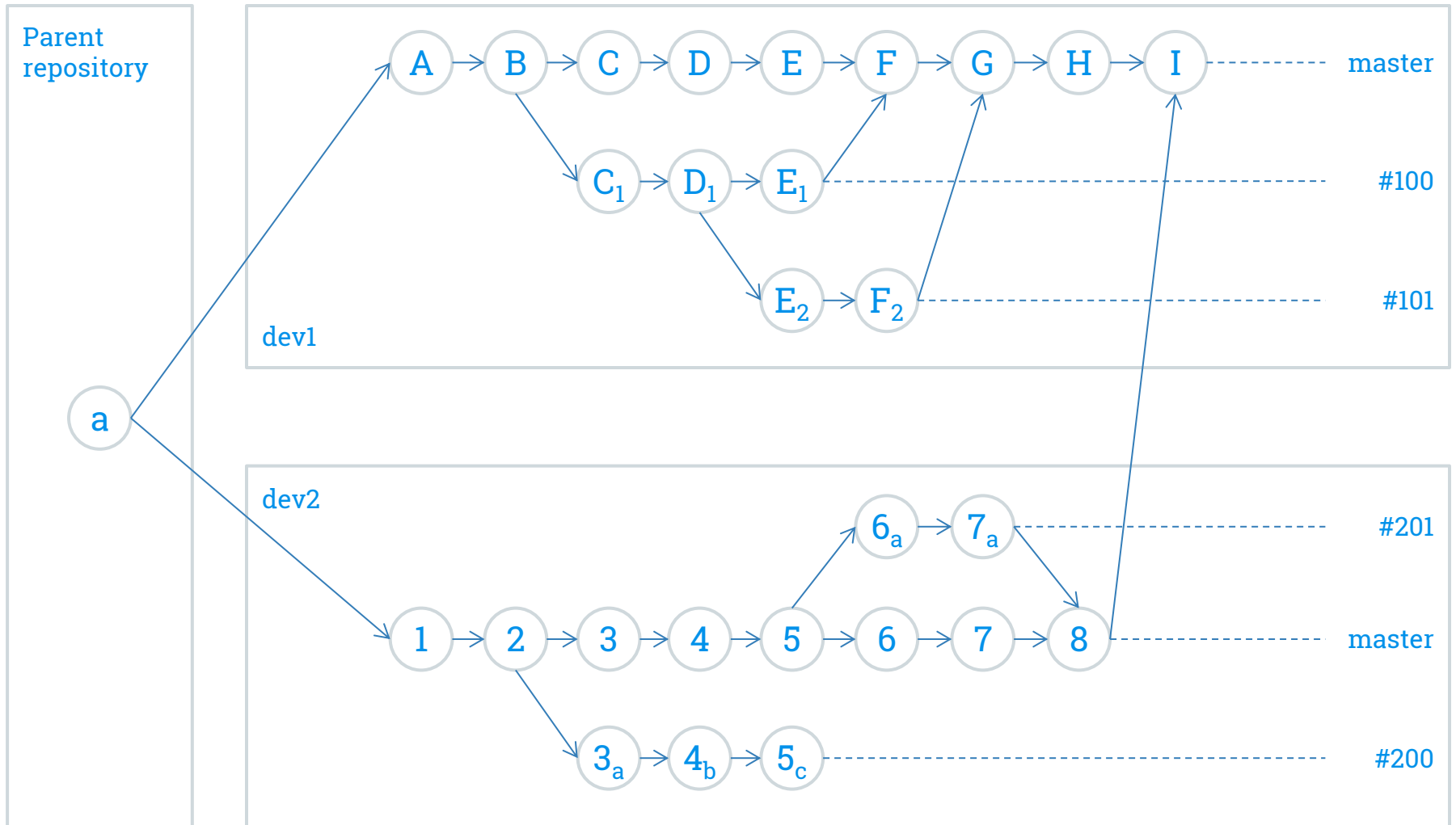
У НАС СВОИ ДАТЫ
РЕЛИЗОВ

Release model

Release model

18.2.0

A → B - master
(18.2.0-SNAPSHOT)

18.2.0 & **18.5.0**

A → B → C → D → E → F → G → H - - - - - - - - - - - - master
(18.5.0-SNAPSHOT)

Code freeze

C_1 → D_1 → E_1 - 18.2.0

18.2.0 & **18.5.0** & **18.2.1**

A → B → C → D → E → F → G → H → I → J → K - - - - master
(18.5.0-SNAPSHOT)

C_1 → D_1 → E_1 - 18.2.0

Release
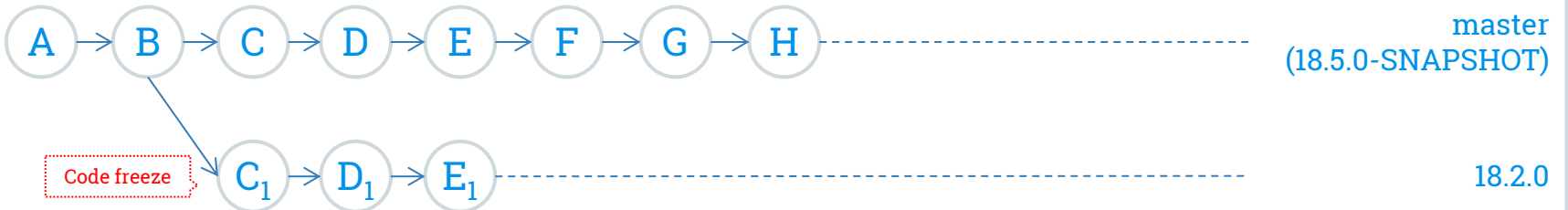
F_2 → G_2 - 18.2.1
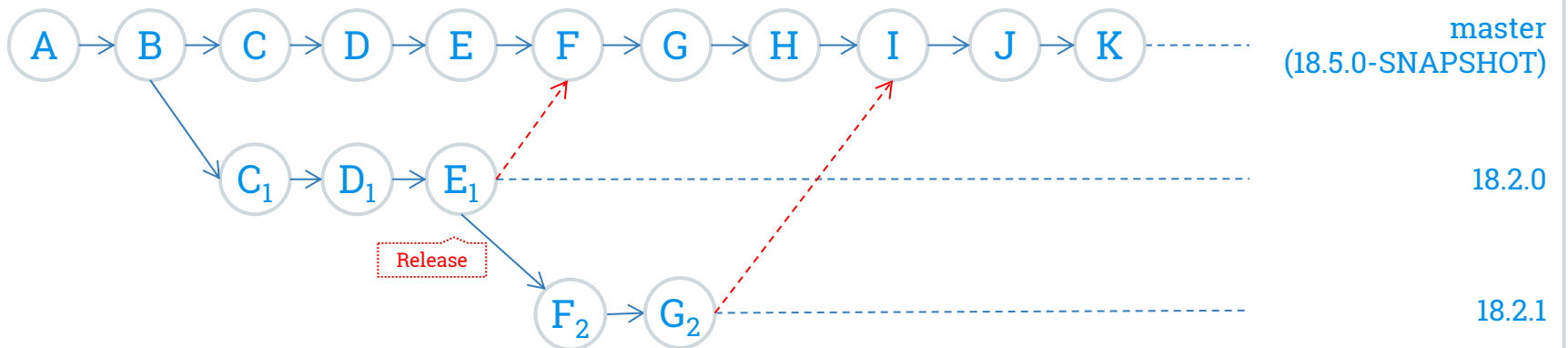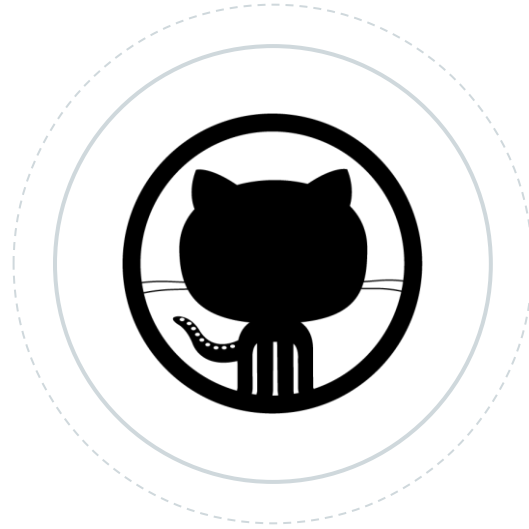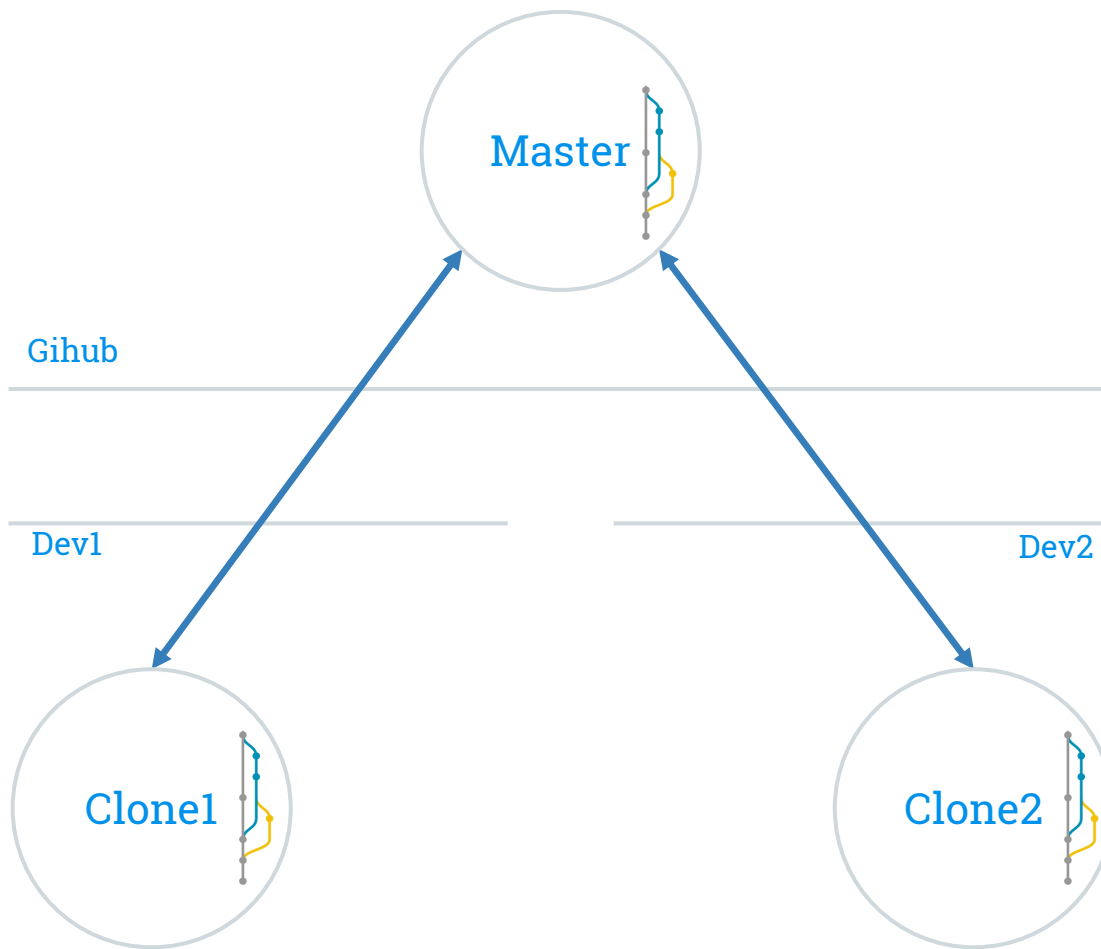
Github

pull-request based
workflow, code review,
integration with
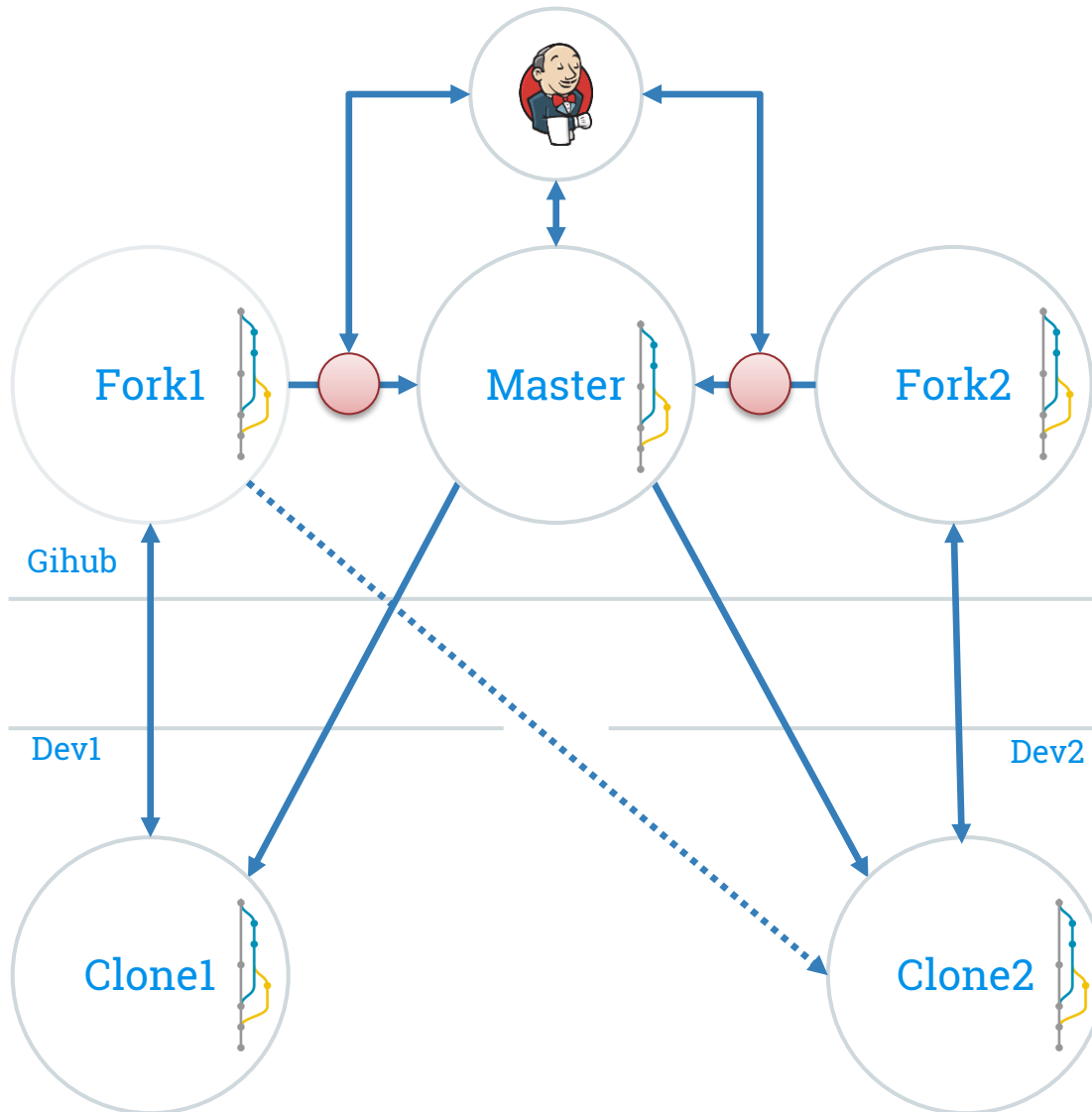Jenkins, start build on
approved pull request,
pull request pre-checks

Master repository + Clones

◎ Multiple branches
◎ All branches are visible for all developers
◎ Pull requests are created between braches in master repository
◎ Requires selected Release Manager role

Master repository + Forks + Clones + Multiple upstreams

- A few **Release** branches in the master repository
- **Any** number of branches inside **forks**
- **Pull requests** are created between braches in **master** and **fork** repositories
- Code **merge** after **code review** only
- Two main remotes: origin & upstream

Use Story isolation & Team collaboration
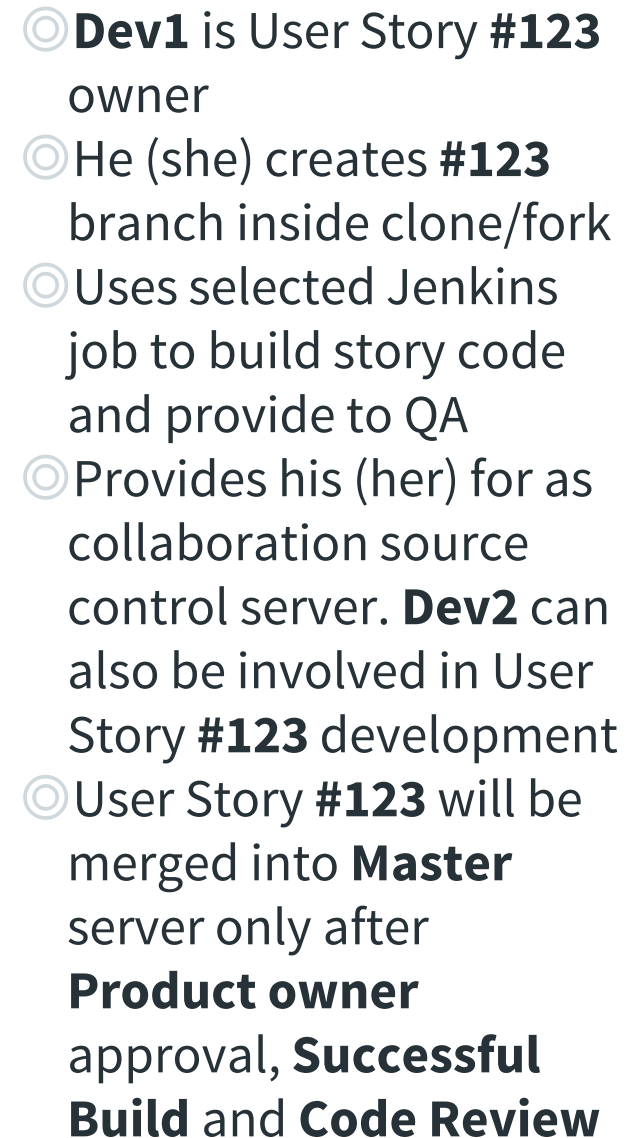
◎ **Dev1** is User Story **#123** owner

◎ He (she) creates **#123** branch inside clone/fork

◎ Uses selected Jenkins job to build story code and provide to QA

◎ Provides his (her) for as collaboration source control server. **Dev2** can also be involved in User Story **#123** development

◎ User Story **#123** will be merged into **Master** server only after **Product owner** approval, **Successful Build** and **Code Review**

High-level picture

Github

Cloud

Dev1

Fork1

Master

Dev2

Fork2

Staging

Dev/ QA

Prod