gepardec

GIT

in Software Development

Date : 03.06.2020
Author: Ing. Thomas Herzog M.Sc

gepardec

Agenda

// Git Basics

// Branching Models

// Fork & Pull Model

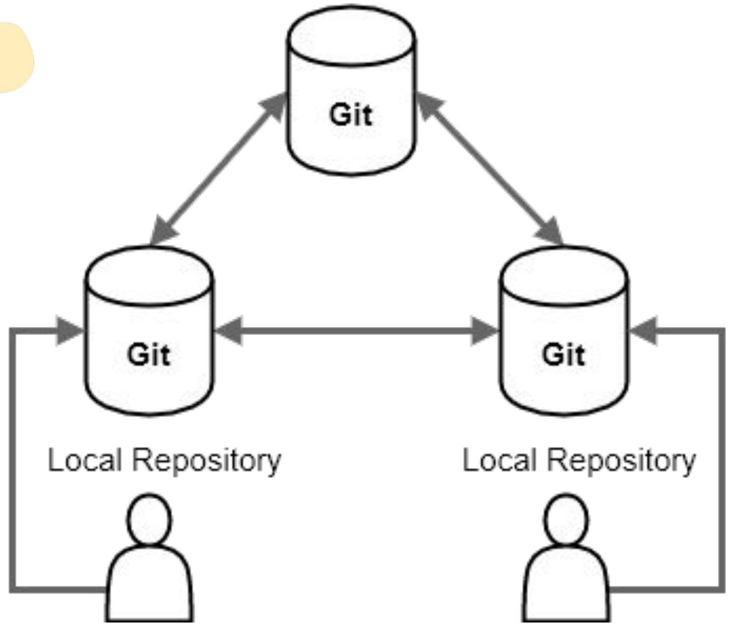
// Branches/Tags and Versions

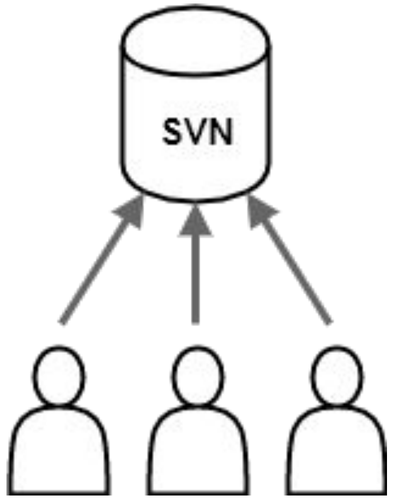
// Git and CI/CD

// Hands on with Git and CI/CD

gepardec

Git Basics (Git vs. SVN)





// Distributed repository

// Local copy of whole repository

// Offline work possible

// Git-Workflows (e.g Github-Flow)

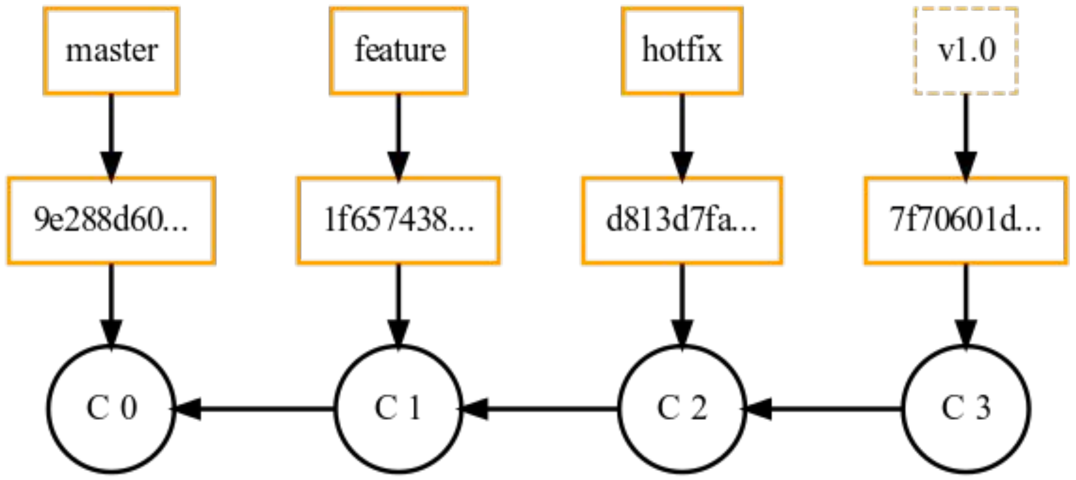
// Centralized repository

// Local working copy

// Offline work not possible

// No workflows have established

Git Basics (Git Commit/Branch/Tag)



// Everything is a Commit

// Commit is represented by a hash

// Branch/Tag are Labels for Commits

// Branch creates a new path

// Tag is a immutable reference to a Commit

Git Basics (Git Three Trees)

// Git tracks changes in three ways:

// **1. Working Copy**

- In sync with the local file system
- Is aware of new/modified/deleted files

// **2. Staging Index**

- Tracks working copy changes
- Only knows changes added via `git add`
- Actually a caching mechanism

// **3. Commit History**

- Adds changes to a snapshot
- Contains Staging Index State at the time of the Commit

gepardec

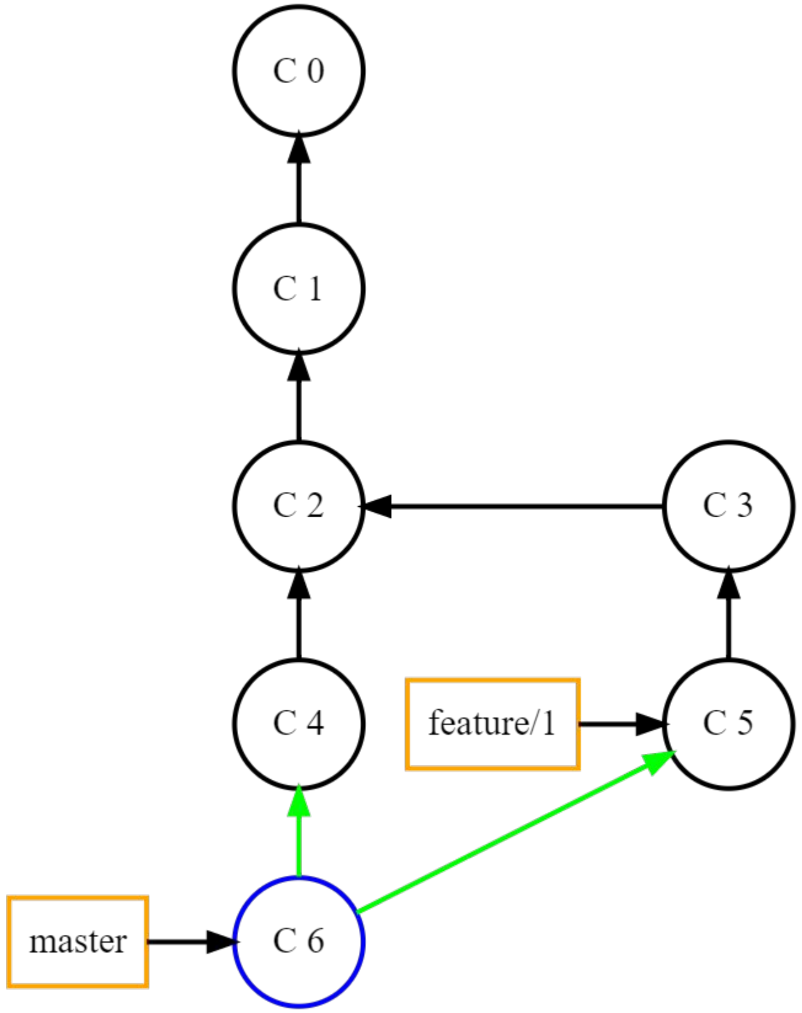
Branching Models

- // Branching is essential in Git
- // Branches isolate work of team members
- // Branches start from a Commit *(it matters from which Commit!)*
- // Branches get integrated into mainstream Branch
- // Branches can be deleted *and all Commits of it!*
- // Branching Models define how to handle Branches
  - Github Flow (https://guides.github.com/introduction/flow/)
  - A successful Git Branching Model (https://nvie.com/posts/a-successful-git-branching-model/)
  - A custom Git Workflow

Branching Model (Github Flow)

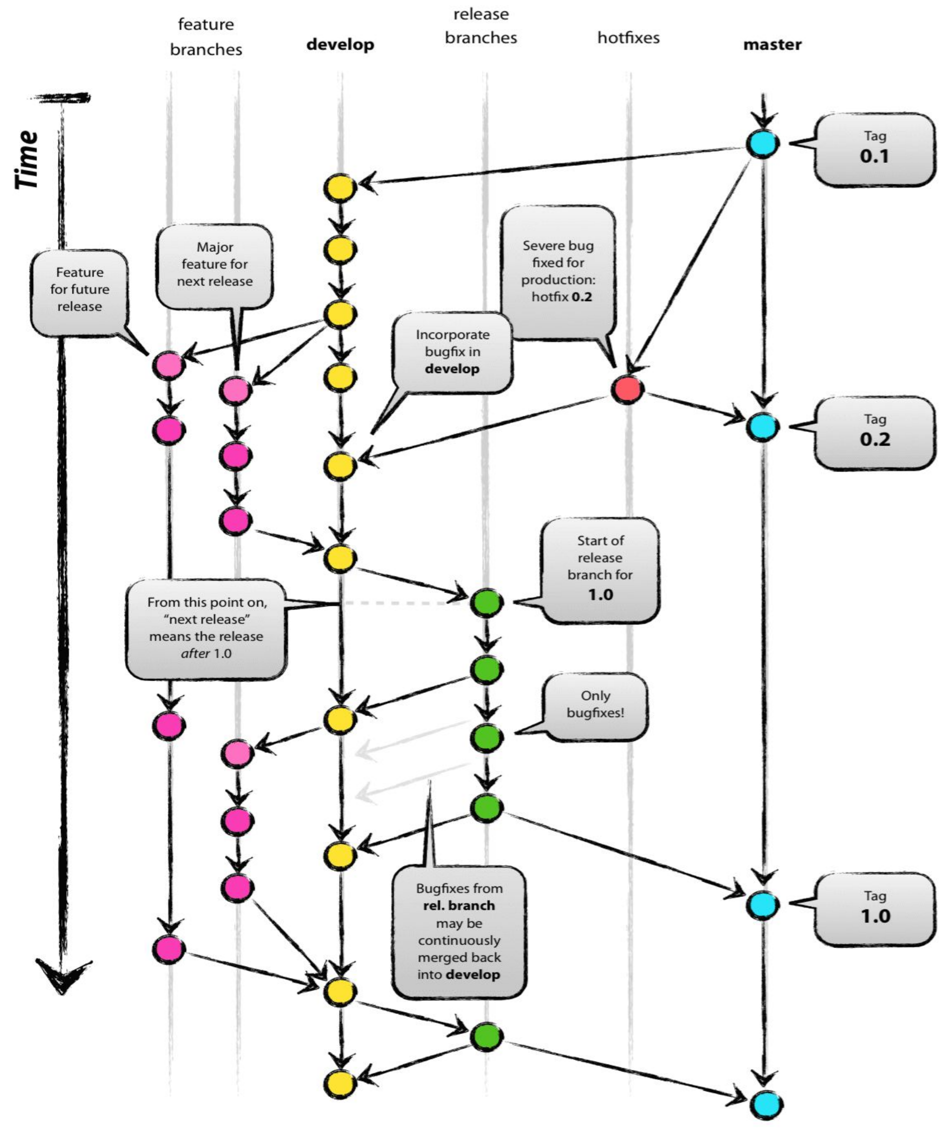
gepardec

- // Defined by Github
  https://guides.github.com/introduction/flow/
- // Very simple!
- // Only Feature Branches and Master Branch
- // Merged via Merge Request
- // Merge Requests are reviewed
- // Merged directly to Master Branch
- **// Master Branch is always deployable!!!**



Branching Model (A successful Git Branching Model)

- // Defined by Vincent Driessen *(10 years ago)*
  https://nvie.com/posts/a-successful-git-branching-model/
- // Complex Workflow
- // Supports handling of multiple versions
- // Multiple Branch Types
- // **Master** Branch replaced by **Develop** Branch
- // Master Branch is always on latest Release
- // Still Merge Requests and reviews


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Tag 0.2
Tag 1.0
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
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

Branching Model (A custom Git Workflow)

// There is no hard spec for Git Flows

// Anyone can define one

// Branch Convention can be defined freely

// Start with an easy one and add complexity as you need

https://www.stickpng.com/img/icons-logos-emojis/question-marks/plain-black-question-mark

Fork & Pull Model (Workflow)

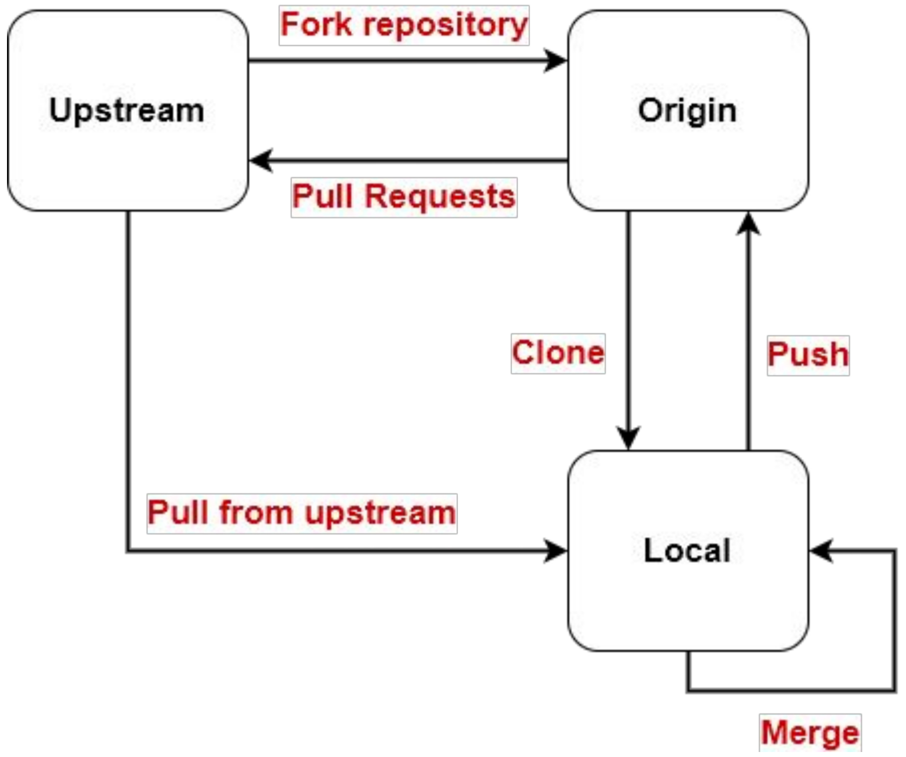
// Developers **Fork** a repository

// Developers **Clone** the fork repository

// Developers **Push** changes to fork repository

// Developers create **Pull Requests** on Upstream

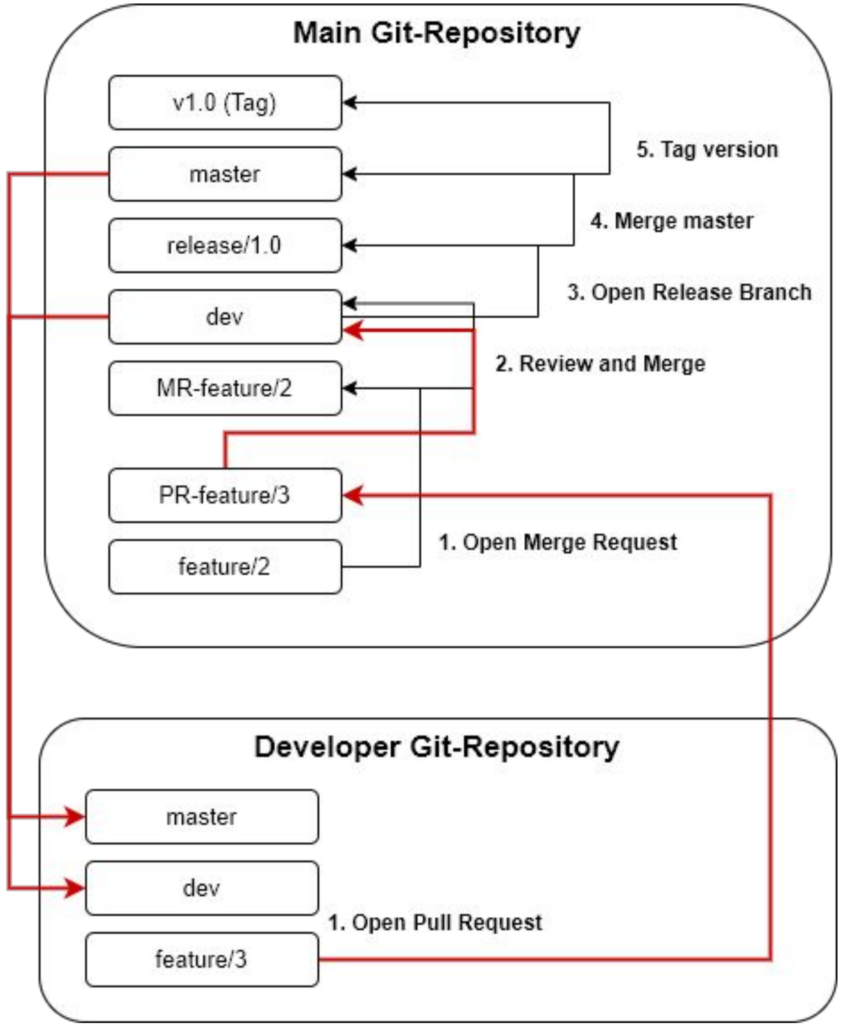
// Developers **Pull from Upstream** to local repository



gepardec

Fork & Pull Model (Merge/Pull Request)

- // Forks provide isolation by repository borders
- // Leads maintain main repository
- // Developers work on forks
- // Developers create Pull Requests
- // Pull Requests are reviewed
- // Pull Requests are merged to main repository
- // Work in main repository still possible
- // Within main repository we use Merge Requests



Fork & Pull Model (Origin and Upstream)

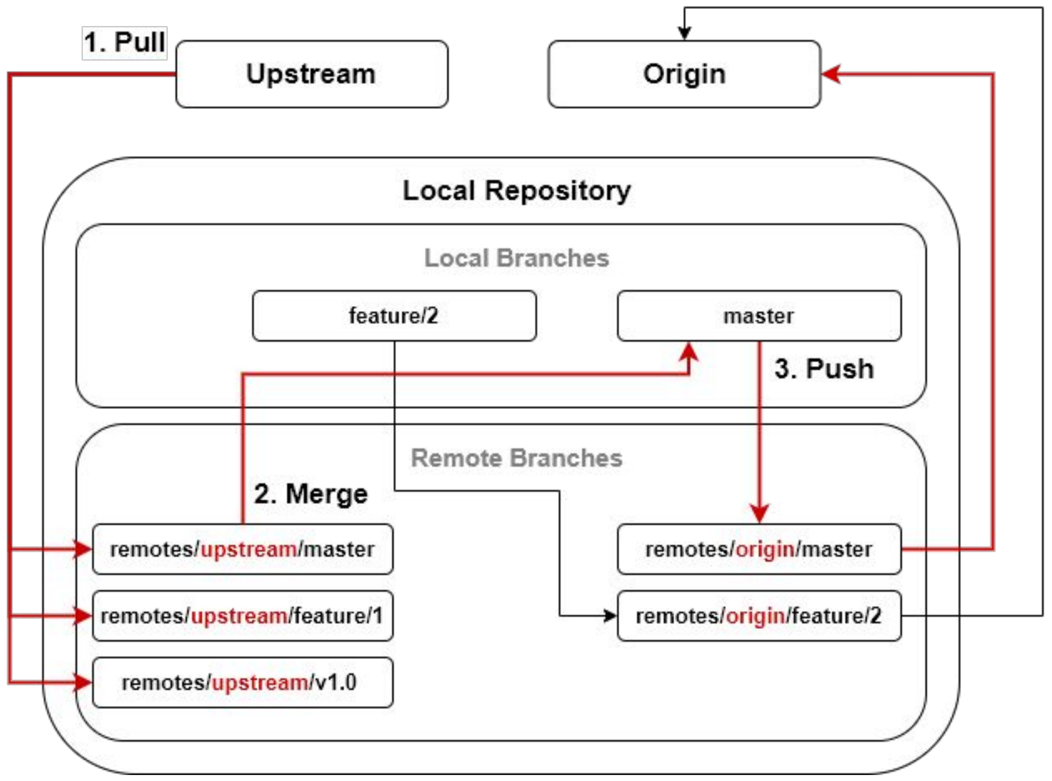
// A Git Repository has remote references

// Remotes hold references to remote repositories

// With forks we have two remotes *(origin, upstream)*

// **Upstream** holds upstream repository references

// **Origin** holds fork repository references



Branches/Tags and Versions

// dev	=	3.0.0-SNAPSHOT
// feature/1	=	FEATURE-1-SNAPSHOT
// release/2.0.0	=	2.0.0.RC[1..n]
// hotfix/1.0.1	=	1.0.1.RC[1..n]
// bugfix/1.1.0	=	1.1.0.RC[1..n]

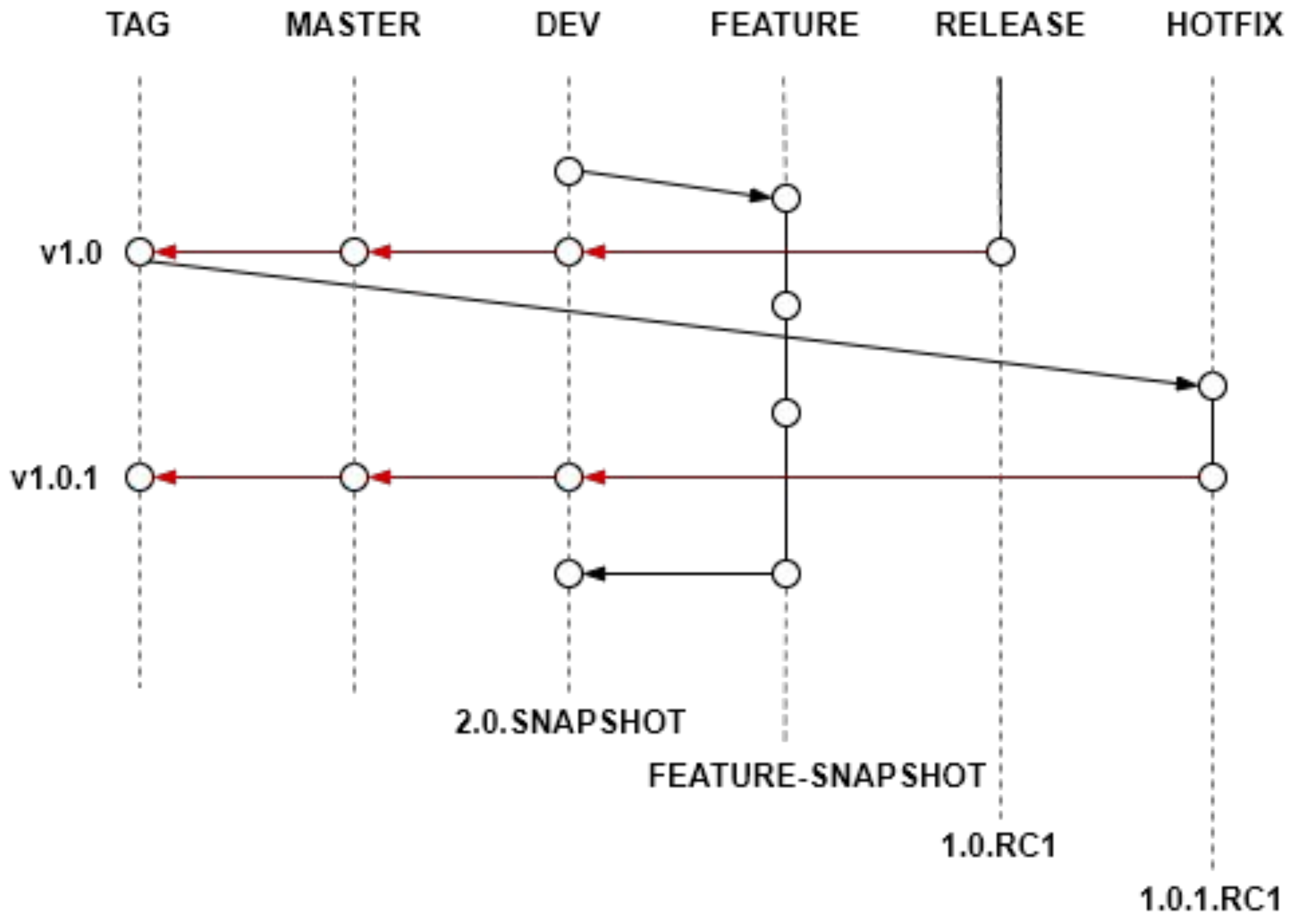
// Branches are strongly related to Versions

// Each Branch translates to an unique Version

// Each Branch produces a deployable artifact

// Each branch **technically** can be deployed to production

Branches/Tags and Versions

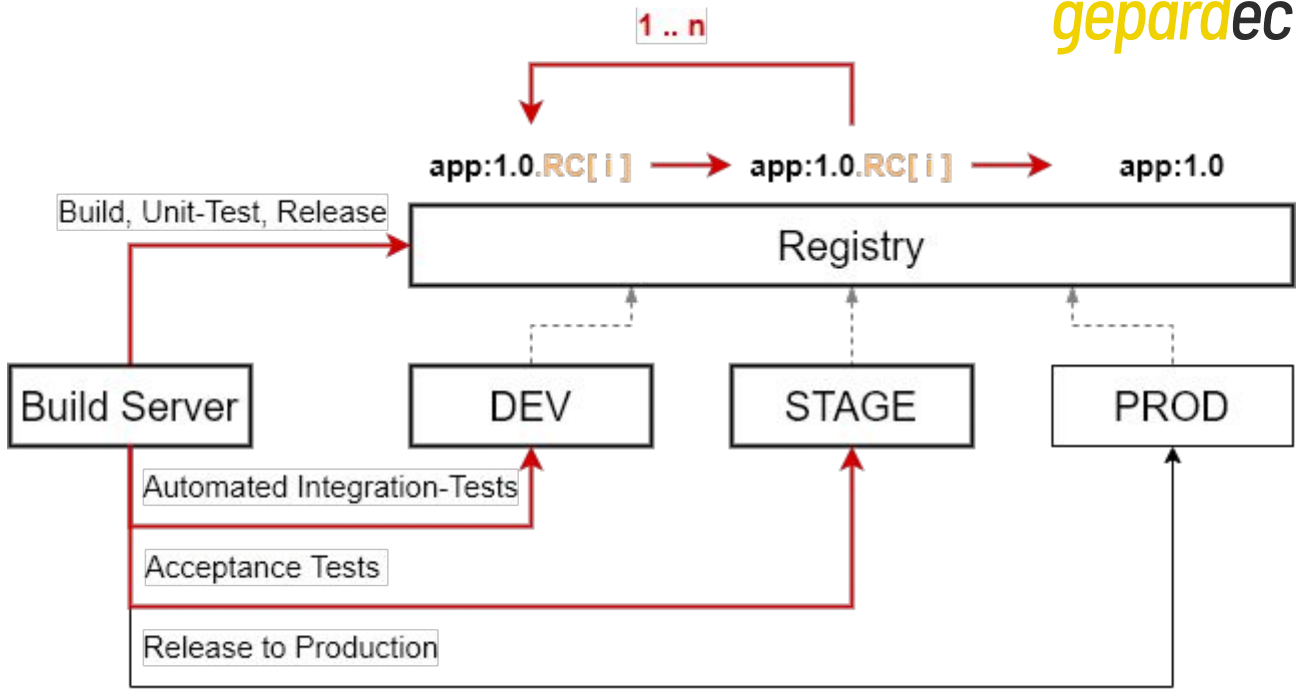


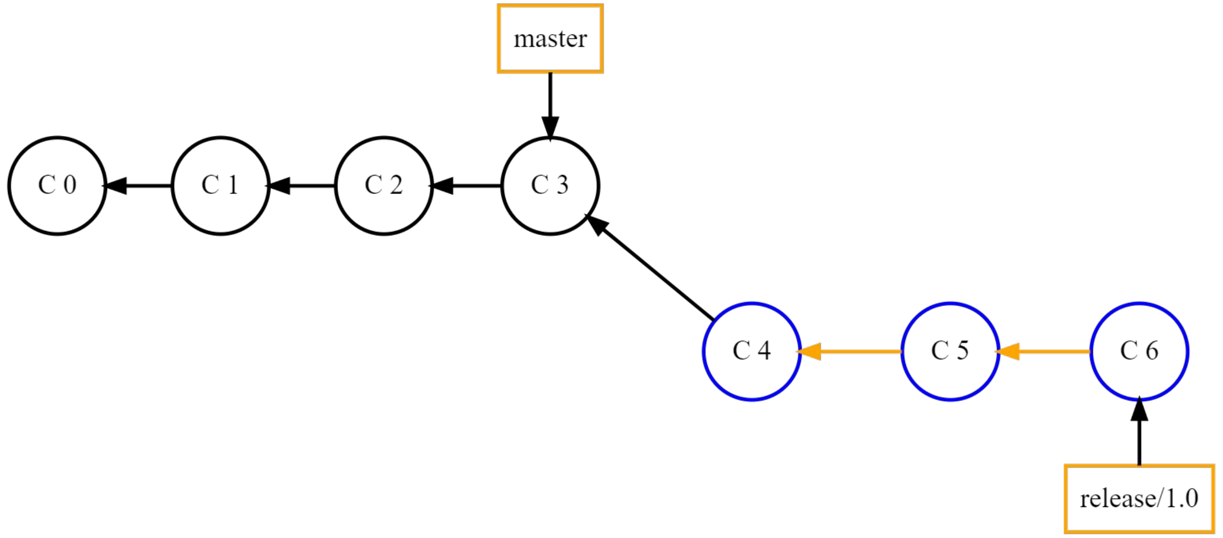
Git and CI/CD (Releases)

- // release/1.0 = 1.0.RC[1 .. n]
- // Each RCs moved through

  DEV -> STAGE -> PROD
- // One Commit = one round trip of

  DEV -> STAGE
- // Last Commit is released to *PROD*
- // No rebuilds for stages, rebuilds for Commits





Git and CI/CD (Releases)

// *C4, C5* and *C6* are **-RC1, *-RC2* and **-RC3*

// **-RC[1..2]* moved through

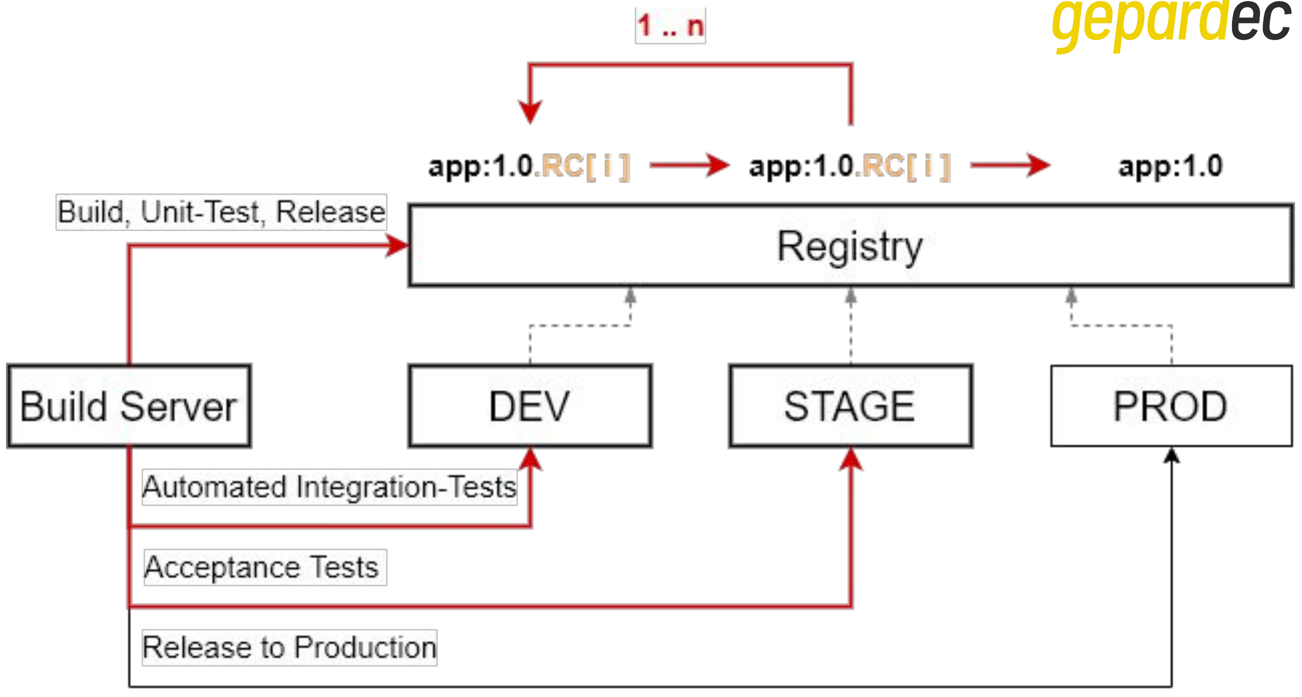
DEV -> STAGE

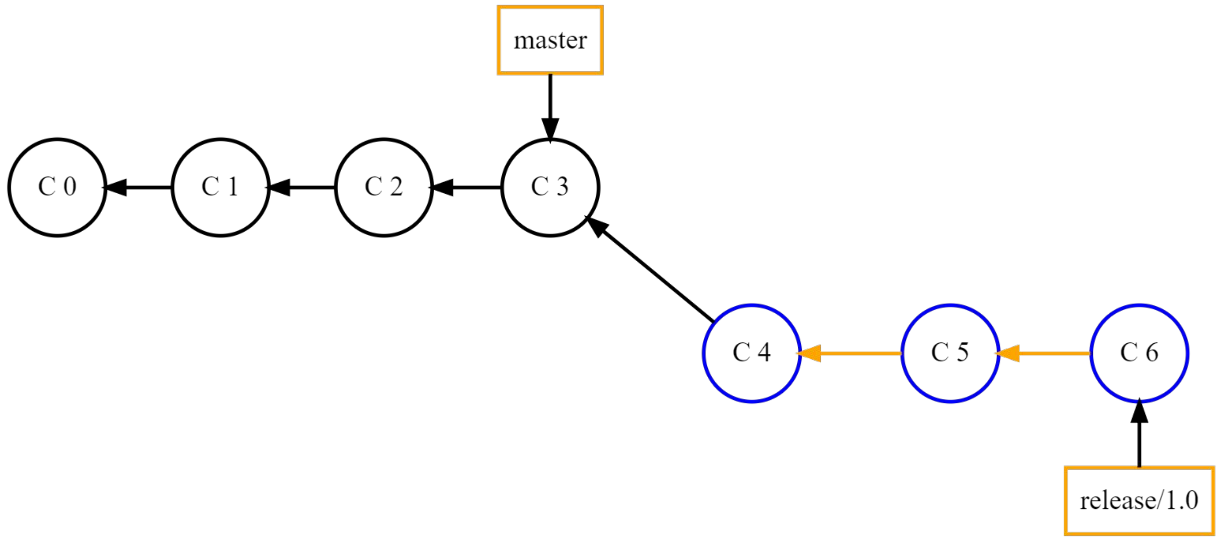
// **-RC3* moved through

DEV -> STAGE -> PROD

// *C6 = 1.0-RC3 = 1.0* = final commit to release

// **-RC3* suffix is removed when released





Hands on with Git and CI/CD (What we want to do)

// Build multiple Branches

// Create an unique Version depending on the Branch type

// Build an runnable artifact and Docker Image

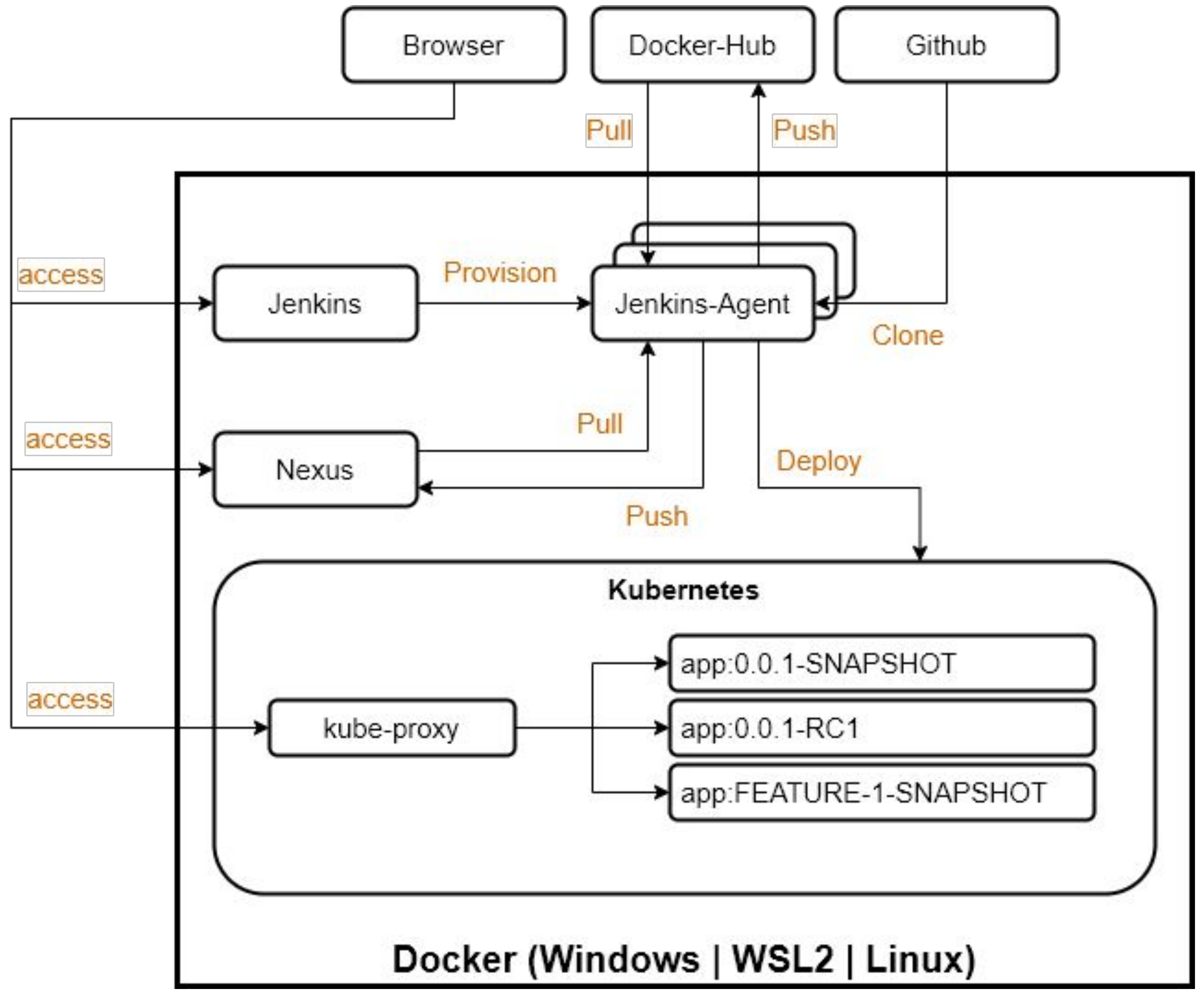
// Release the build artifact to Nexus and the Docker Image to Docker Hub

// Deploy each built Docker Image from Docker Hub to Kubernetes

// https://github.com/Gepardec/GranitTreff-030620

gepardec

Hands on with Git and CI/CD (How does the infrastructure look like)



gepardec

Hands on with Git and CI/CD (How does the infrastructure run)

// The infrastructure runs in

- Docker Desktop for Windows
- WSL 2 (Ubuntu 20.04.LTS + Debian 10.x)
- native Linux OS

and is defined by Docker Compose.

// Jenkins

- is configured via Configuration-as-Code (CASC)
- and the builds run in a Jenkins Docker JNLP Agent.

https://github.com/jenkinsci/configuration-as-code-plugin

// Secrets are managed by Docker

// Fairly little effort to setup

// Definitely not a production setup!!