Large Scale Development with Git and Gerrit

Matthias Männich

Matthias Sohn

EclipseCon Europe 2012, Ludwigsburg

Git

… a distributed revision control system built by the Linux project to facilitate code review

JGit & EGit

… Git in Java and Eclipse

Gerrit

… a JGit based code review system built by the Android project

How this journey started…

2006 first experiments:

- Git wasn't ready for prime time

2009 SAP joining the Git party

- join JGit, EGit, Gerrit
- piloting Git & Gerrit at Eclipse
- start Lean Development Infrastructure
- first internal pilots

large SAP projects moving to Git&Gerrit

- Feb 2010 "Netweaver Cloud" on-demand platform
- Jul 2011 large C++ Engine Project

Soon: generally available to all kinds of projects

Lean Development Infrastructure

Agile project organization

- **Teams of 10**
- **Large projects: Product team + n dev teams**

Learn from Open Source

- **Tools:**
  Git, EGit, Gerrit, Skalli
  Maven, Nexus, Tycho, Sonar …
- **Communication: mailing lists, forums**
- **Contributions between projects/products**

Git / Gerrit for Lean DI

All repositories hosted on central Gerrit server

- Used around the globe
- 2k repositories, >2k users
- Scales pretty well if repository < 1G

Central admins operate server and ensure standards

Self Services for Projects (Skalli)

- create Git repository
- self-administer their repositories

Introducing Code Review

Learning from successful open source projects

- transparency through code review and mailing lists

Code Review

... is new for most SAP developers

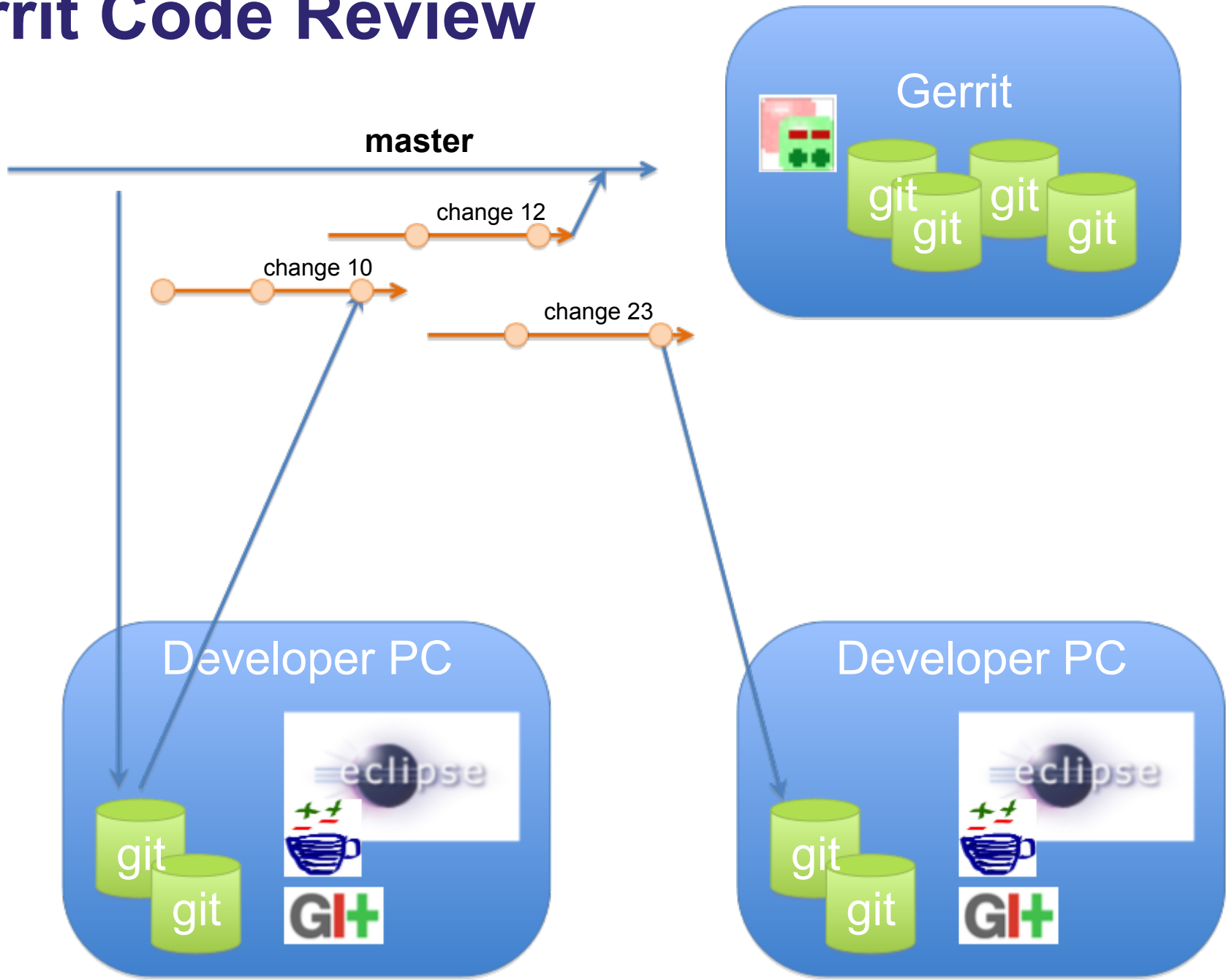
... is optional

... used by many projects today

... some teams do reviews across teams

Takes time to learn

Gerrit Code Review



Code Review

push change

Gerrit notifies reviewers

other teams

verify change

review code

team

Jenkins CI

comment & vote

improve change

abandon change

merge change into master

Examples

Large C++ Engine Project

- **> 250 developers, many locations**
- **large code base: > 1M lines of code**
- **1 huge Git repository, many branches**

“Netweaver Cloud” on-demand platform

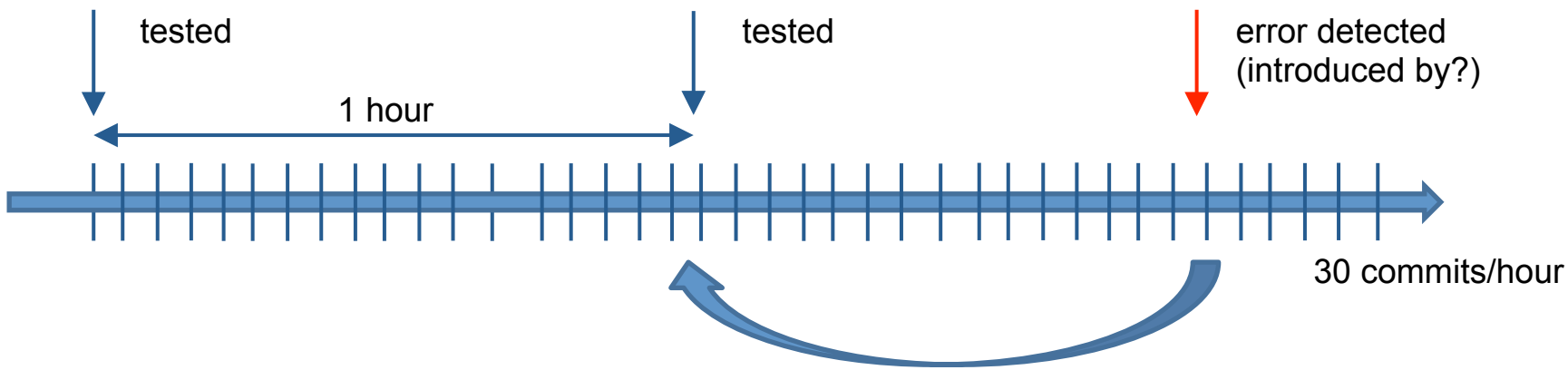
- **~150 developers, 3 locations**
- **modular Java stack**
- **>100 Git repositories**

Migrate large projects - Motivation

in central VCS branching might get complicated

mostly single development branch models

continuous integration does not scale well



not every commit gets full test coverage

high delay between erroneous submit and detection

Migrate large projects - Considerations

good training of the team is essential

- convincing	80%	rather easy
	95%	takes some time
	100%	…

infrastructure needs will increase

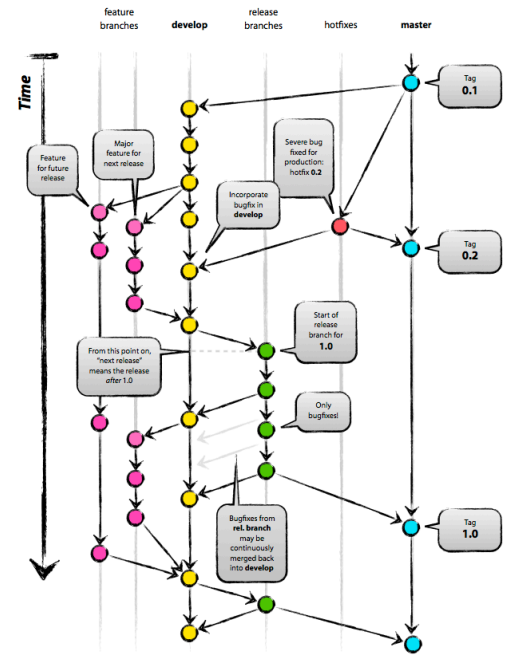
responsibilities might shift from quality team to development

standard compliance (if applicable)

Migrate large projects - Branching

appealing to implement workflows inspired by

- Linux Kernel Development
- Gerrit Development
- Git Flow
- …



preconditions might not be feasible

- strong hierarchies
- modular code structure
- “it’s done when it’s done” <> commercial interests

Migrate large projects - Branching

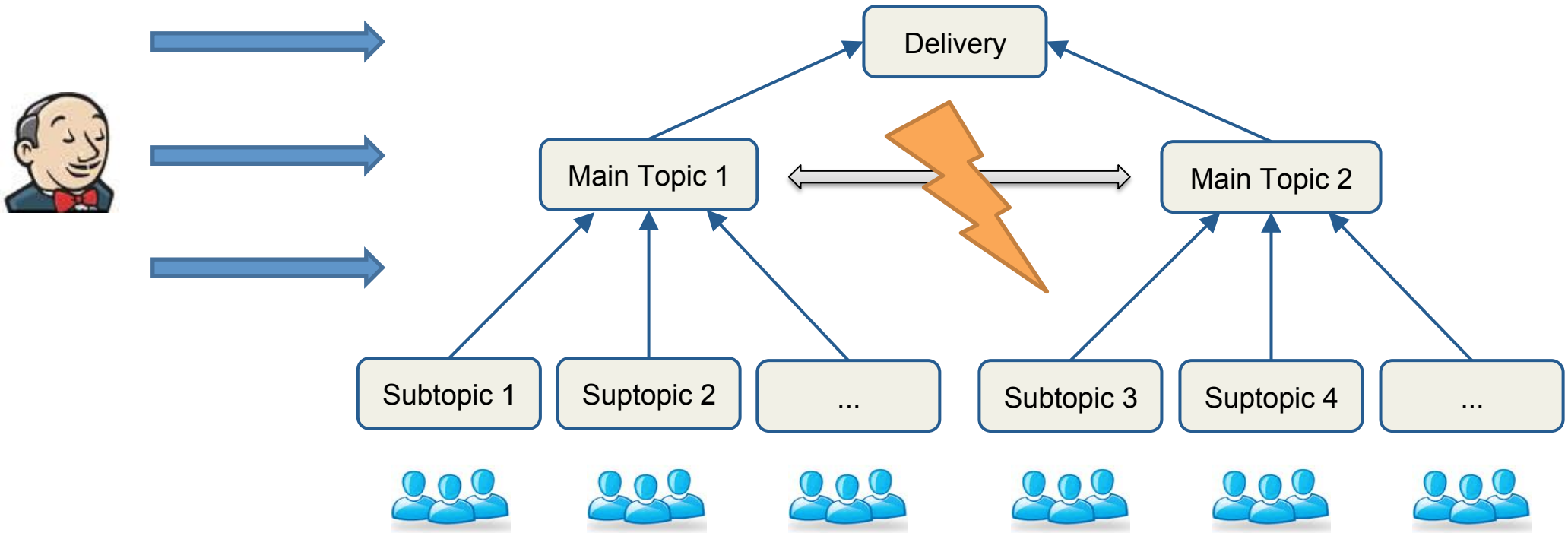
currently implemented and constantly evolving

- lots of topic branches (> 100 active)
    - topic owner concept (responsibility role)
    - small teams
    - different branch lifetimes
    - team defined quality barriers

- flat hierarchy

- high level branches maintained by quality team

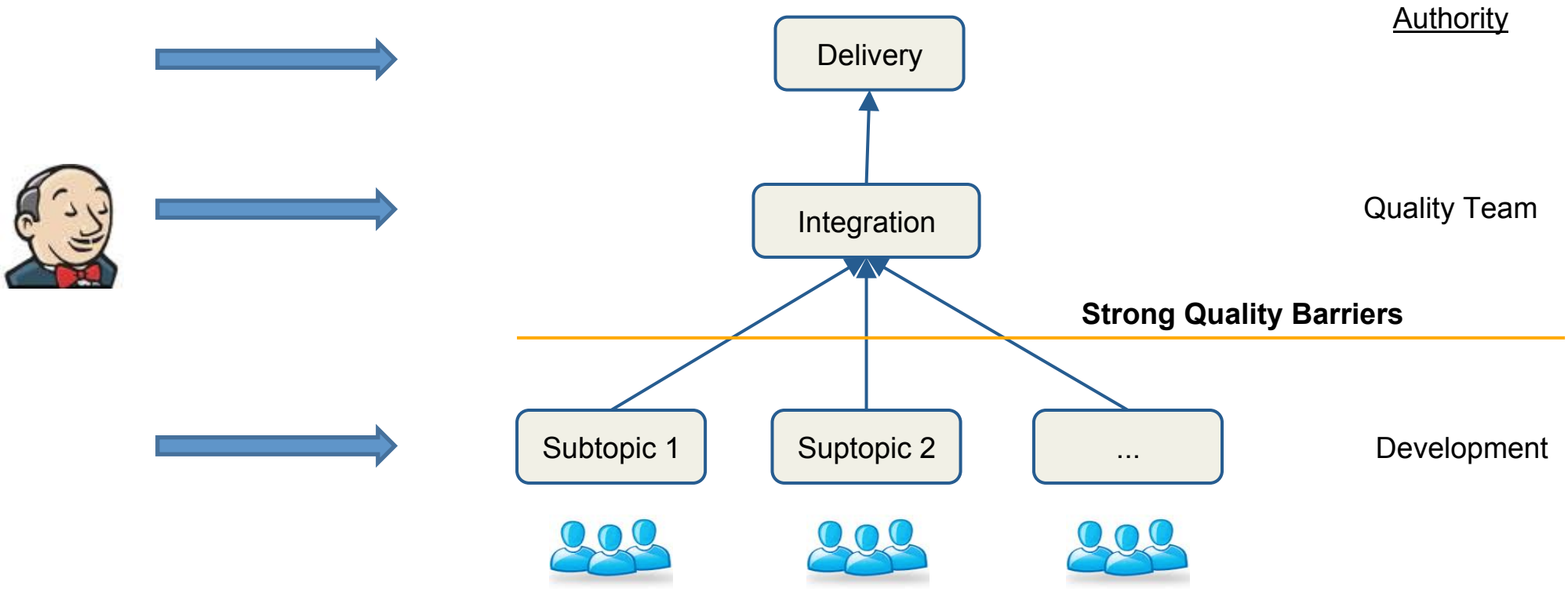
Migrate large projects - Branching



consider

- main topic branches should be distinct development areas (modularization?)
- possible delays caused by topic integration (always acceptable?)
- integration of main topics is done by quality team

Migrate large projects - Branching



integration is done by developers
approval is given by quality team

quality barriers implemented with Gerrit Code Review

pretested code states => always stable integration branch

Migrate large projects – Quality barriers

Gerrit usage allows “pretested commits”

commits have to pass (before integration!)

- Code review (human)
- Builds and tests on several platforms

test coverage dependent on branch hierarchy level

ideal: every quality metric on every level

Migrate large projects - Recommendations

there is no golden rule (sorry)

get inspired by similar projects

discuss and implement a solution

heavily review your design and stay able to adjust

shift responsibilities to expertise location

- code integration: development
- quality judgment: quality team

implement extensive tool support

Migrate large projects – Some rough stats

- 250 developers
- 120 topic branches (lowest level)
- 50k Gerrit changes per year
- 100 “big” build and test servers

Netweaver Cloud Platform

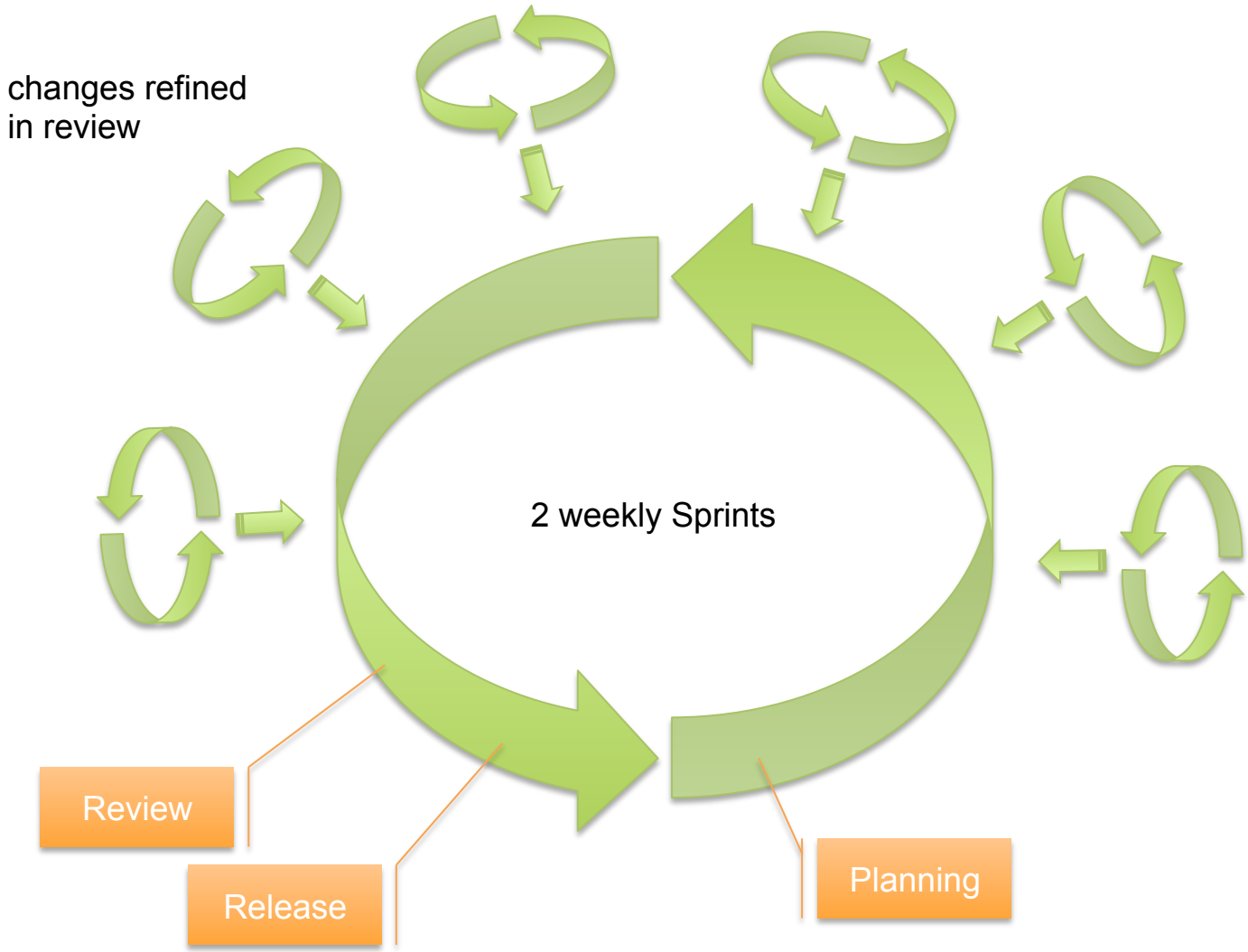
~150 developers

Cloud infrastructure
Cloud services
OSGi (Equinox) based Java server
Tools

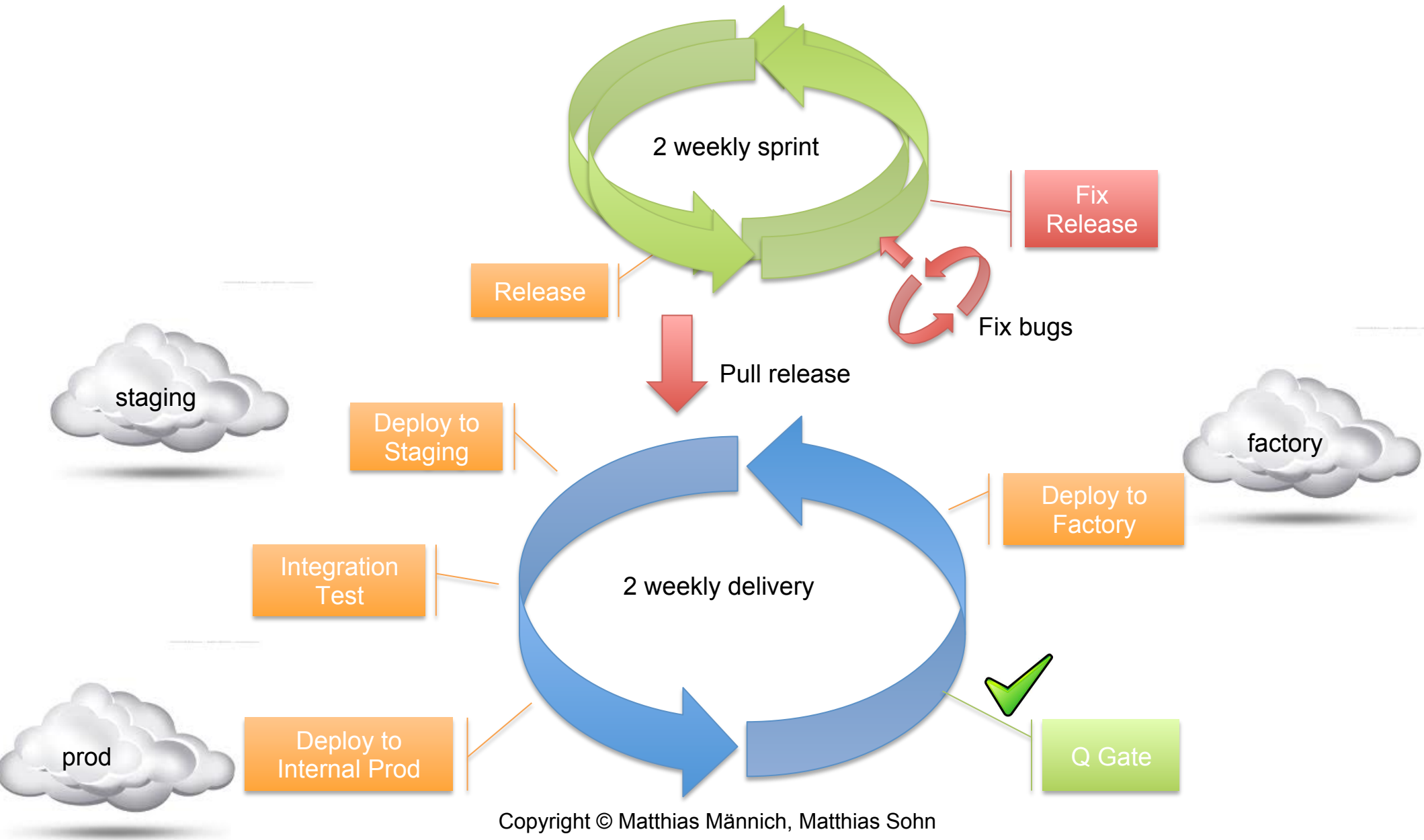
many mid-size repositories
modular Maven build
versioned API
p2 deployment

delivery every 2 weeks

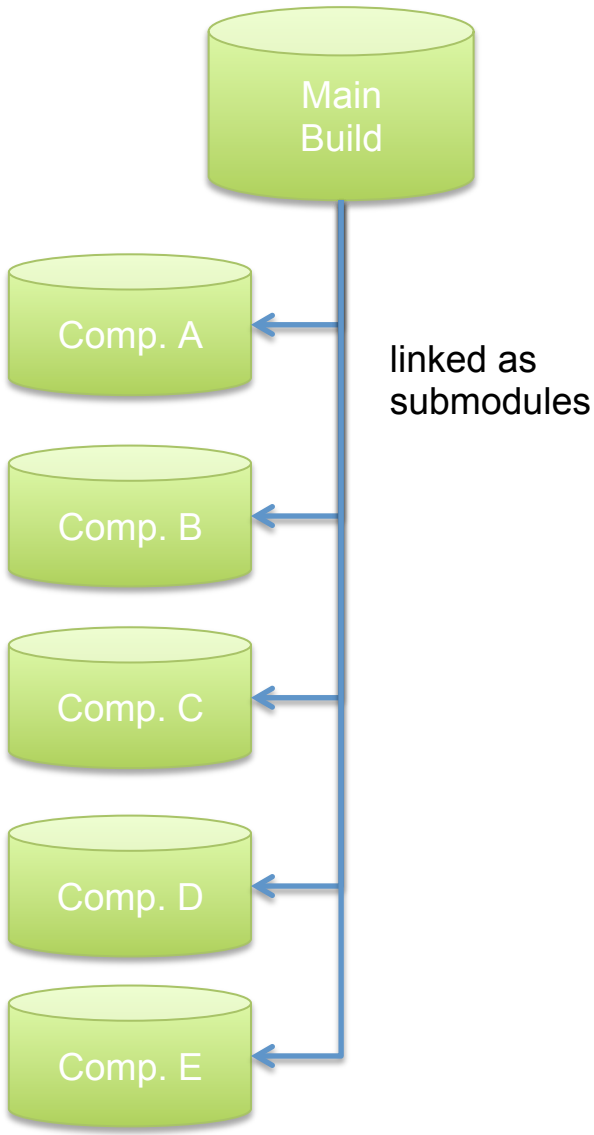
Code Review & Scrum



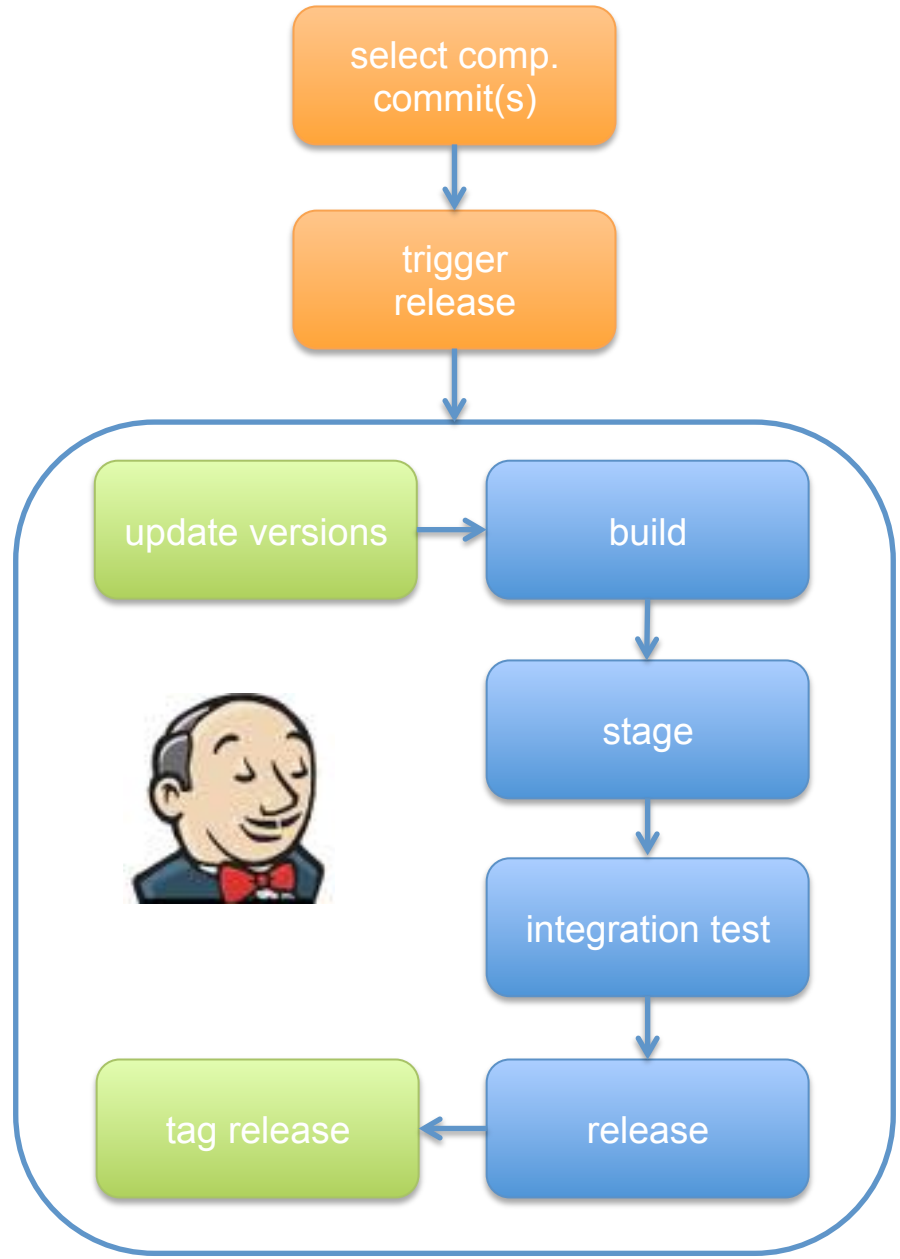
Continuous Delivery



Integrating Core Cloud Infrastructure



linked as submodules



Summary

Benefits

- (many) developers like it
- review and pre-tested commits improve quality
- always able to ship

Future

- scale
- improve tools based on feedback

- make use of submodule support in Gerrit
- better support correction process
- manage contributions between projects (inner source)

Q & A