Domain Driven Design of Large Software Systems

Master Digital Science / Software Architecture, WS 2025 / 26

Prof. Dr.-Ing. Stefan Bente

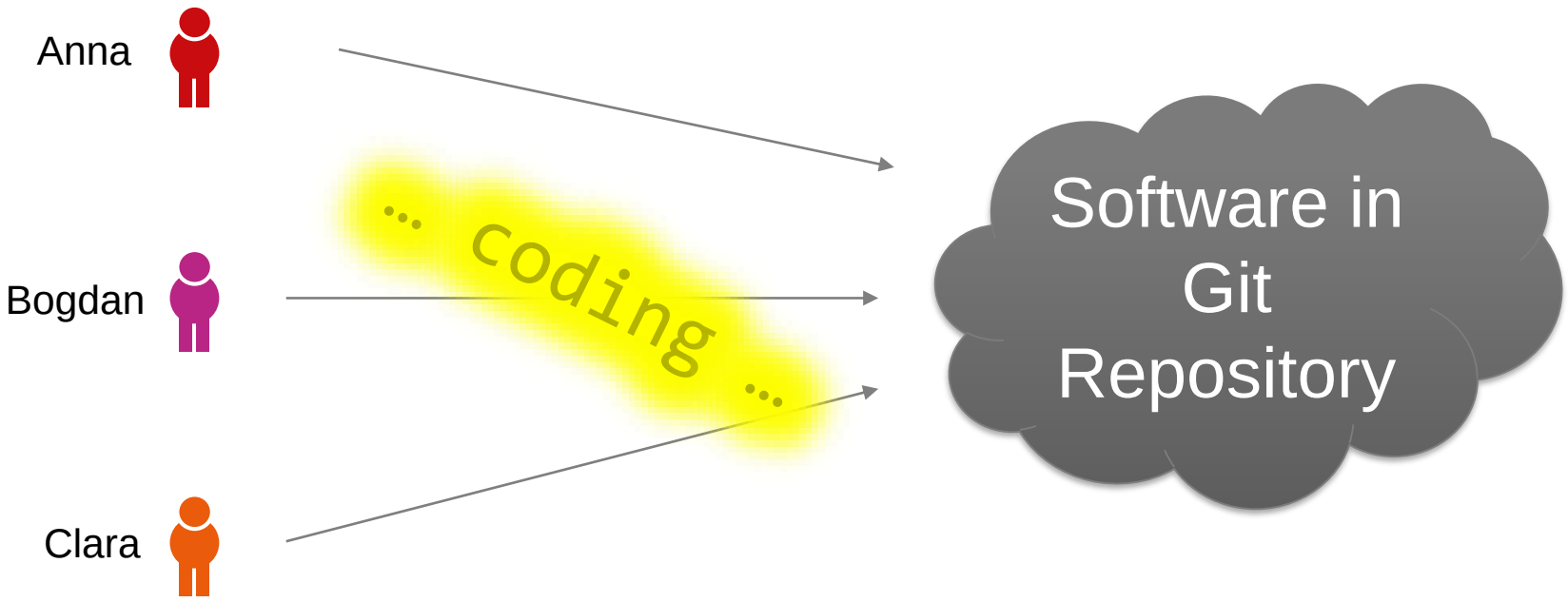
Team Development Basics:
Branching Strategies

21.11.2025

Technology
Arts Sciences
TH Köln

Motivation

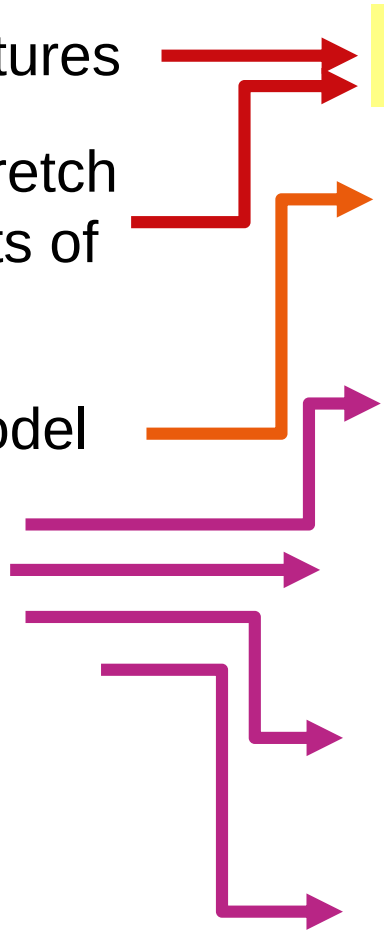
Development as a Team



- You'll be working in (small) teams on the same Git repo
- in Microservice context: usually 1 repo per service
- in Modulith context: 1 mono-repo for the whole system

Challenges of Collaborative Development (and what helps)

1. Independent features
2. Features may stretch over several parts of the code
3. Open Source Model
4. Governance: Ensure uniform quality standards



1. Branching Strategies
2. Merge Request Governance
3. Central documentation and knowledge mgmt
4. API Documentation
5. Architectural principles as guidelines for development teams
6. Architectural Decision Records (ADR)

Branching Strategies

Source: Martin Fowler, Patterns for Managing Source Code Branches
https://martinfowler.com/articles/branching-patterns.html

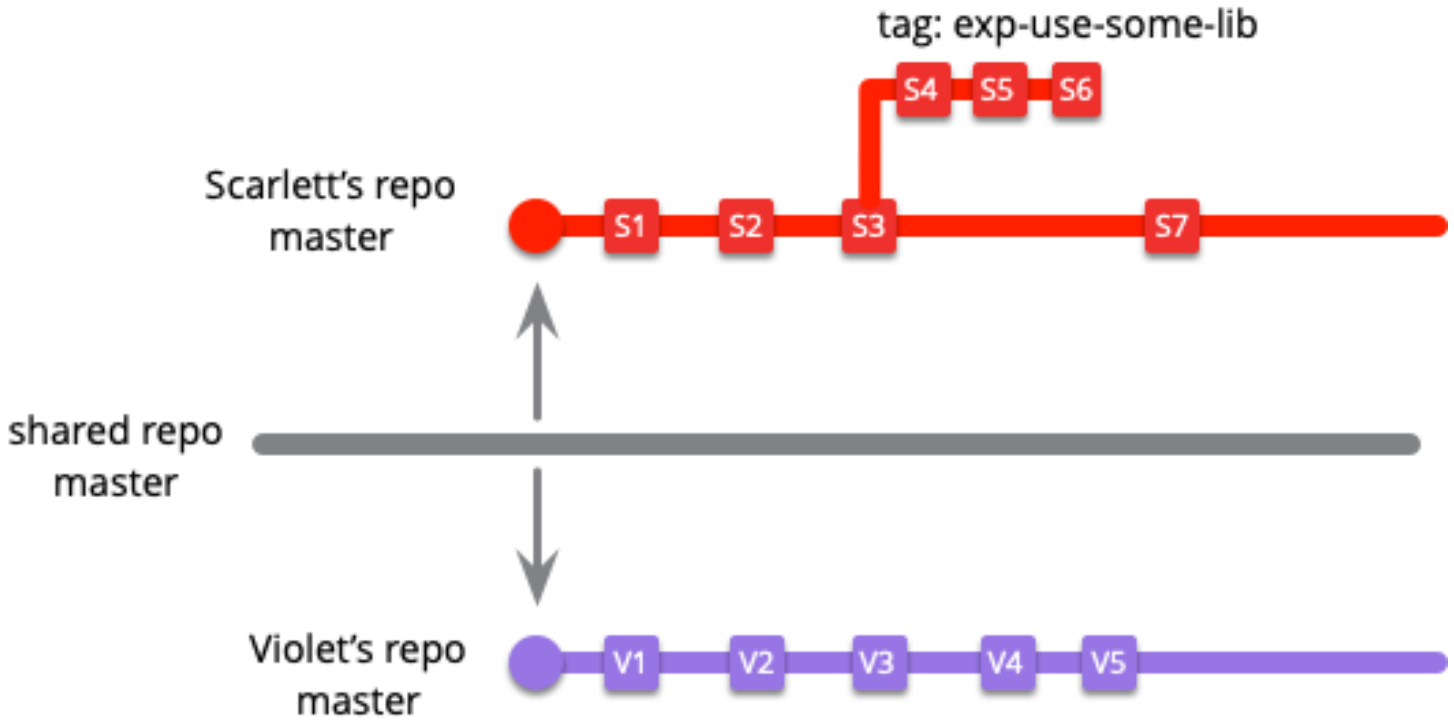
Worth reading in full!

All subsequent illustrations from this source, or using the same style

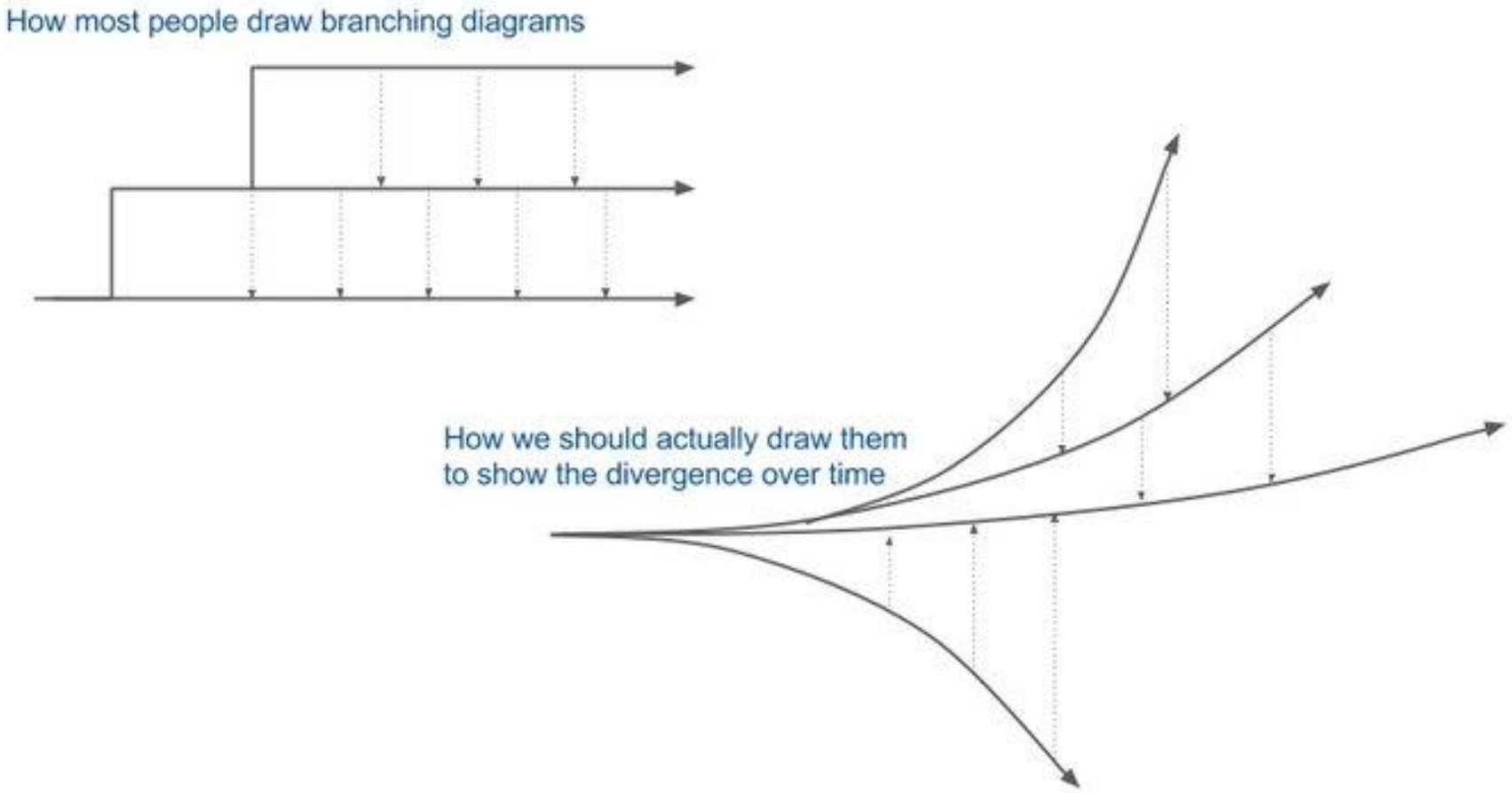
The 4 Main Options according to Fowler

- Big Bang Integration at Project End
  - Each developer creates own branches and does his/her work there; integration at the end of the project
- Mainline Integration (working with Main Branch)
  - Developers integrate their work by pulling from mainline, merging, and - if healthy - pushing back into mainline
- Feature Branching
  - Put all work for a feature on its own branch, integrate into mainline when the feature is complete.
  - Low vs. high Frequency
- Continuous Integration
  - Developers do mainline integration as soon as they have a healthy commit they can share, usually less than a day's work

1) Big Bang Integration at Project End

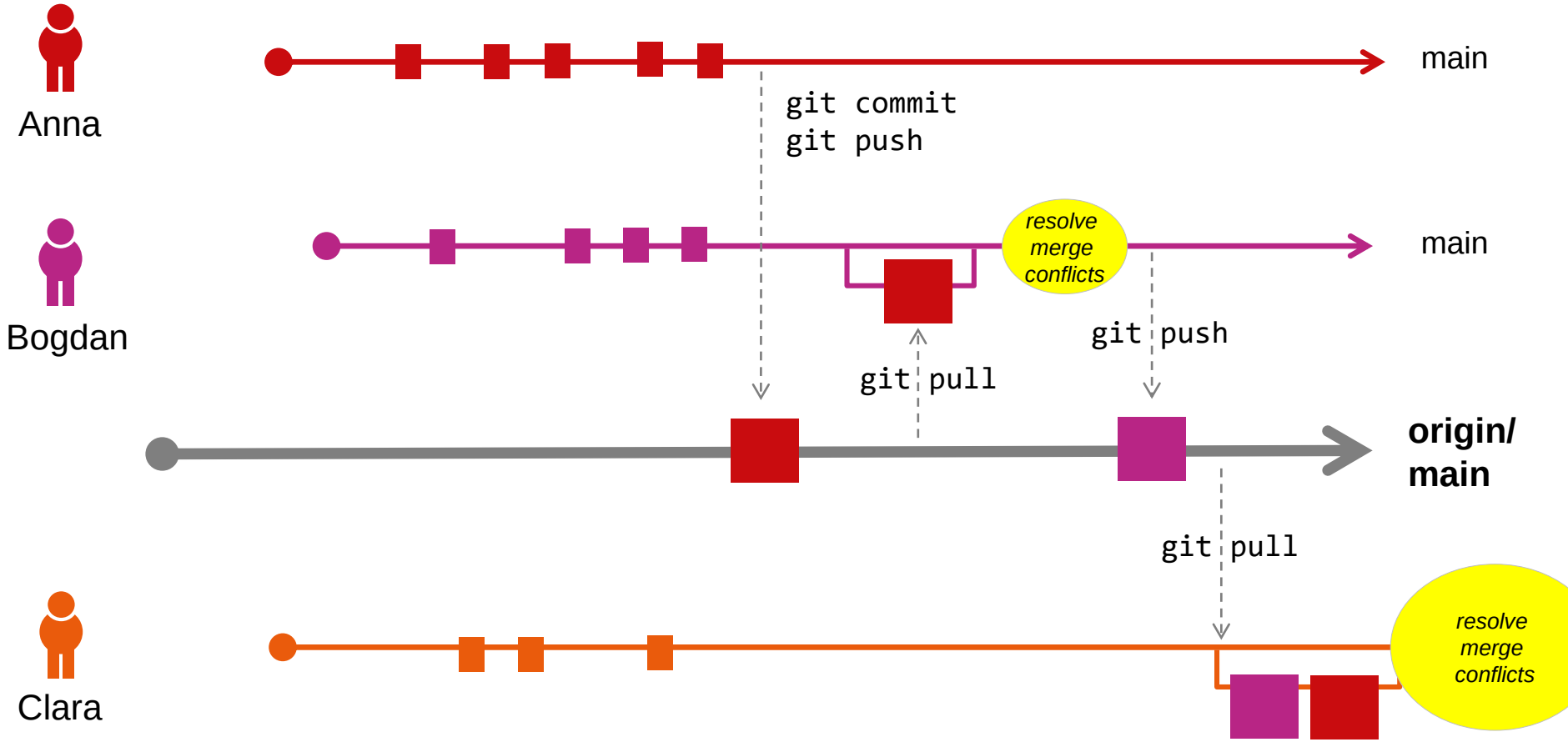


… and the Effect



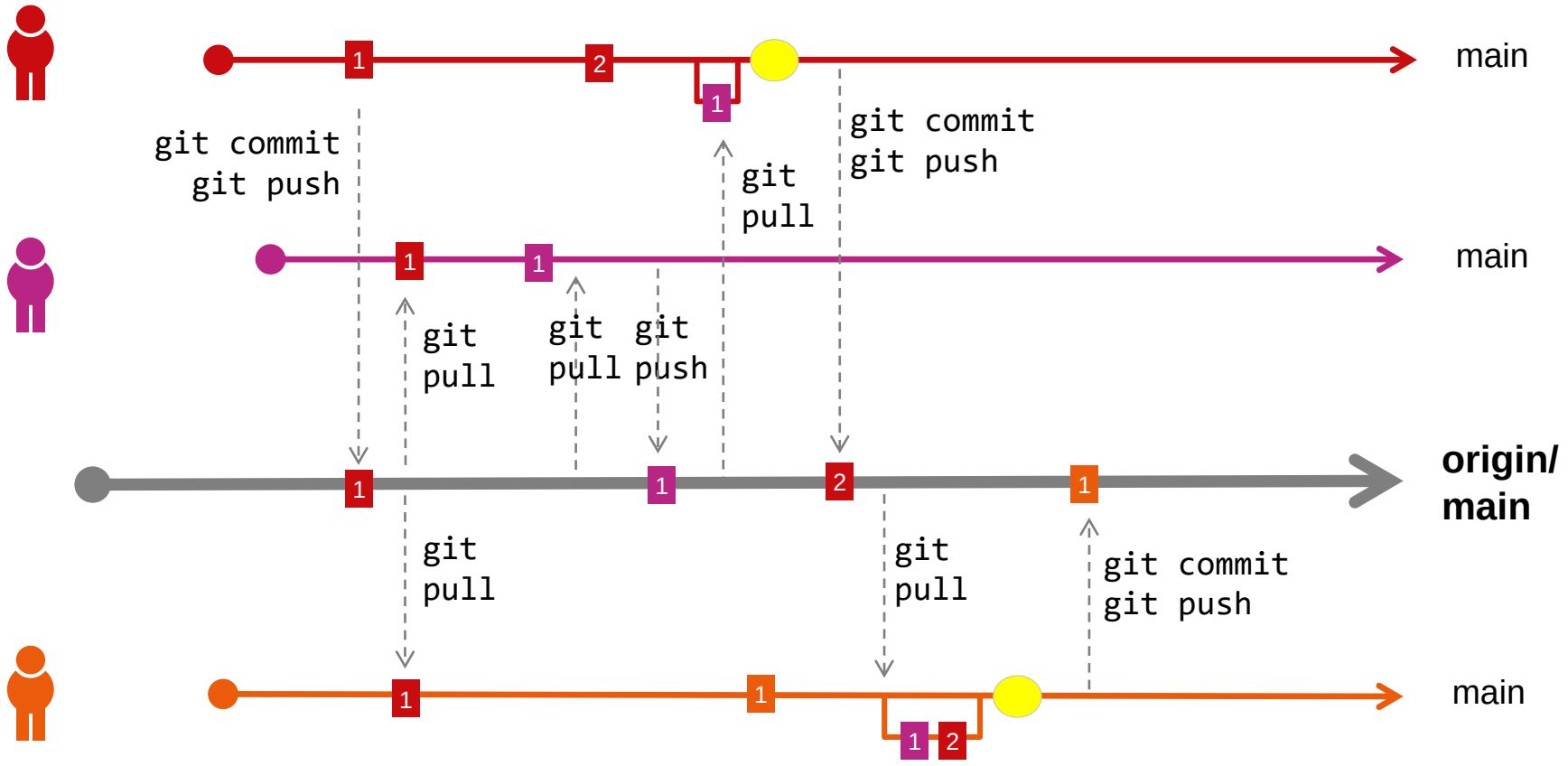
2) Mainline (working with a Master Branch)

2) Mainline (working with a Main Branch)



(Darstellungsweise angelehnt an:
Martin Fowler, Patterns for Managing Source Code Branches, https://martinfowler.com/articles/branching-patterns.html

2) Common Main Branch: How to avoid Merge Hell?



2) Common Main Branch: **How to avoid Merge Hell?**

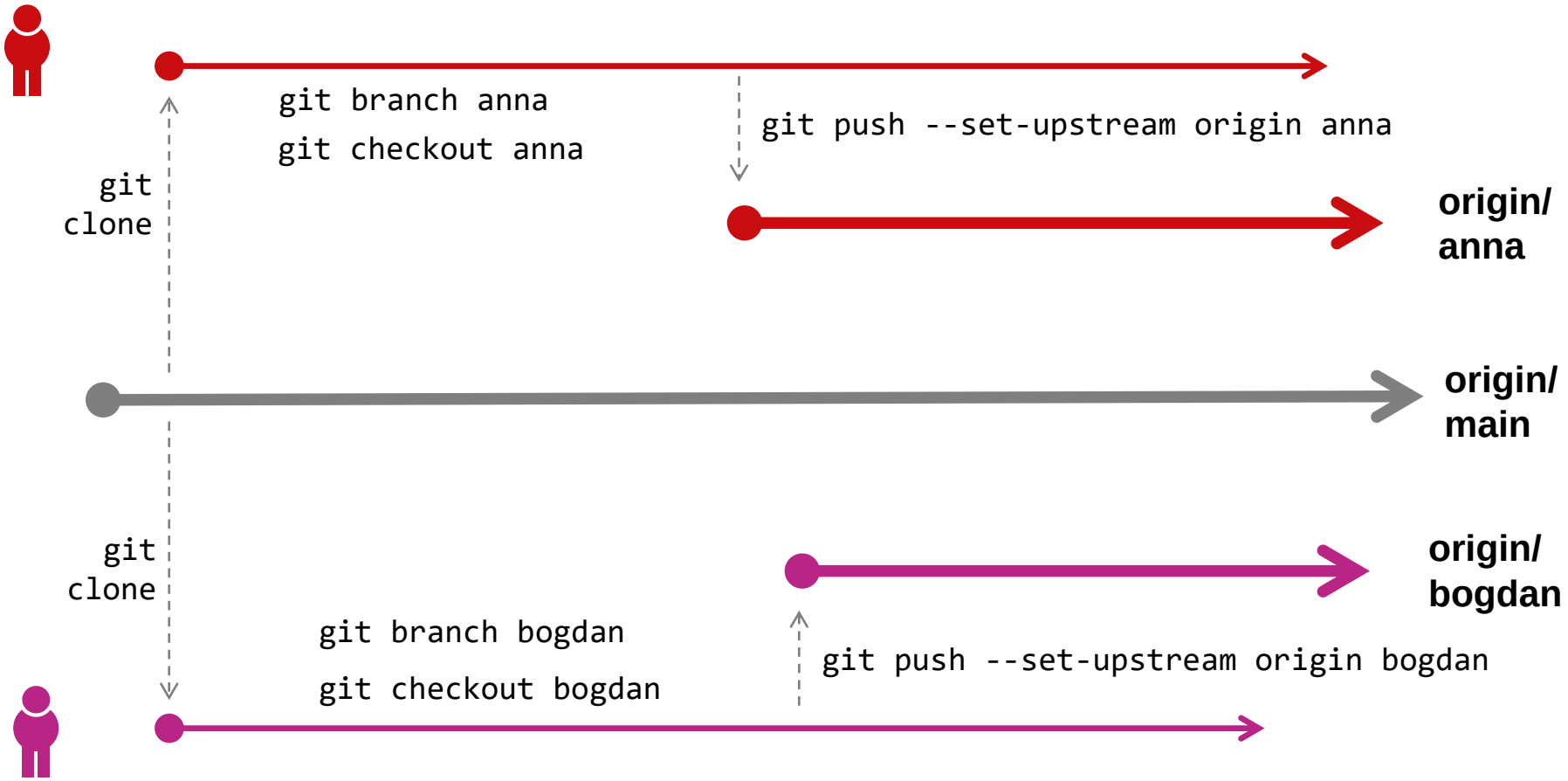
- IMPORTANT: Each day at least 1x …
- `git commit`
- `git pull`
- `(resolve merge conflicts)`
- `git push`

- Otherwise, the project will be completely ruined after just 1-2 days. ☹

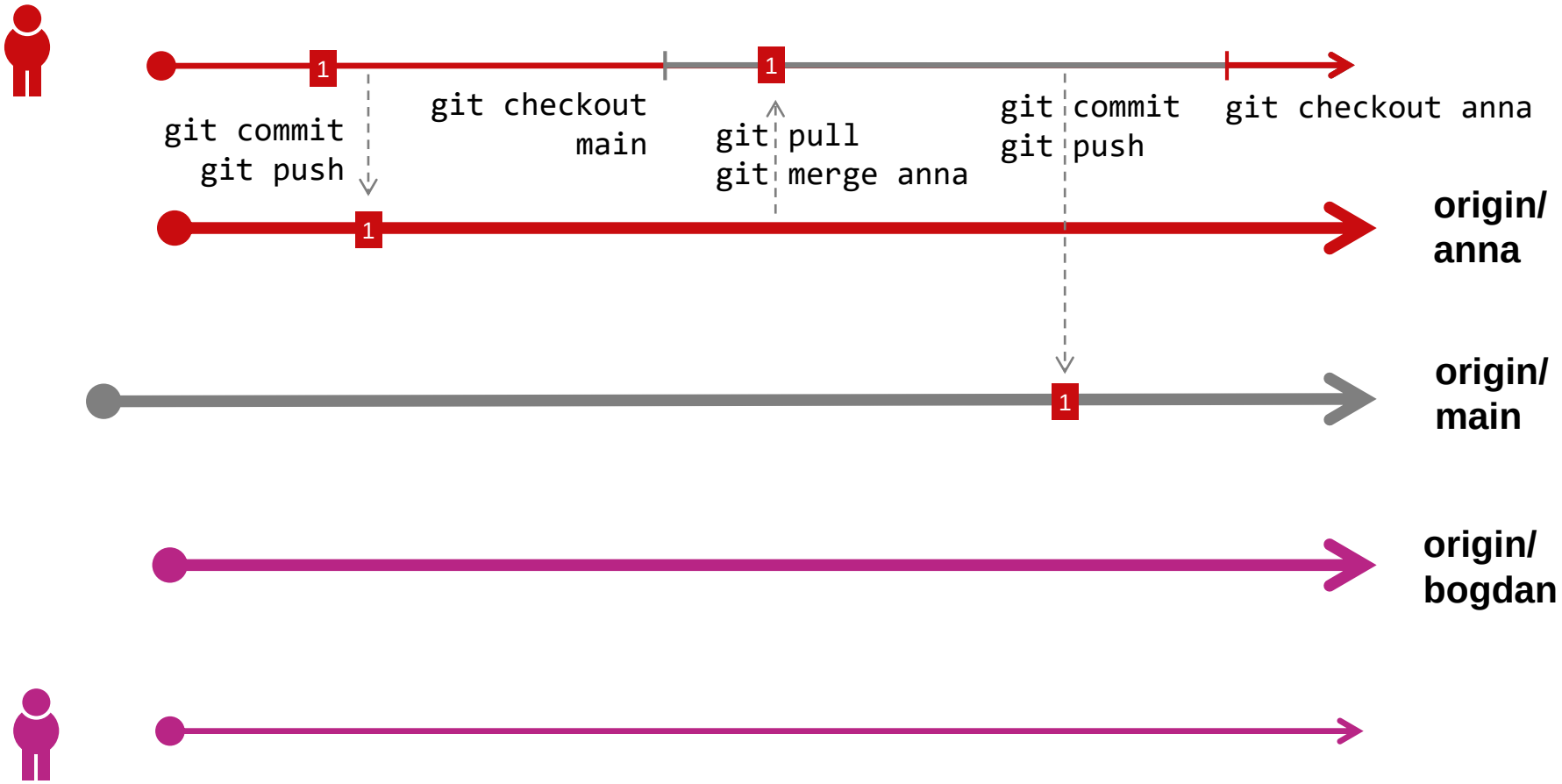
- With >= 3 team members: **use feature branches**

3) Working with Feature Branches (1)

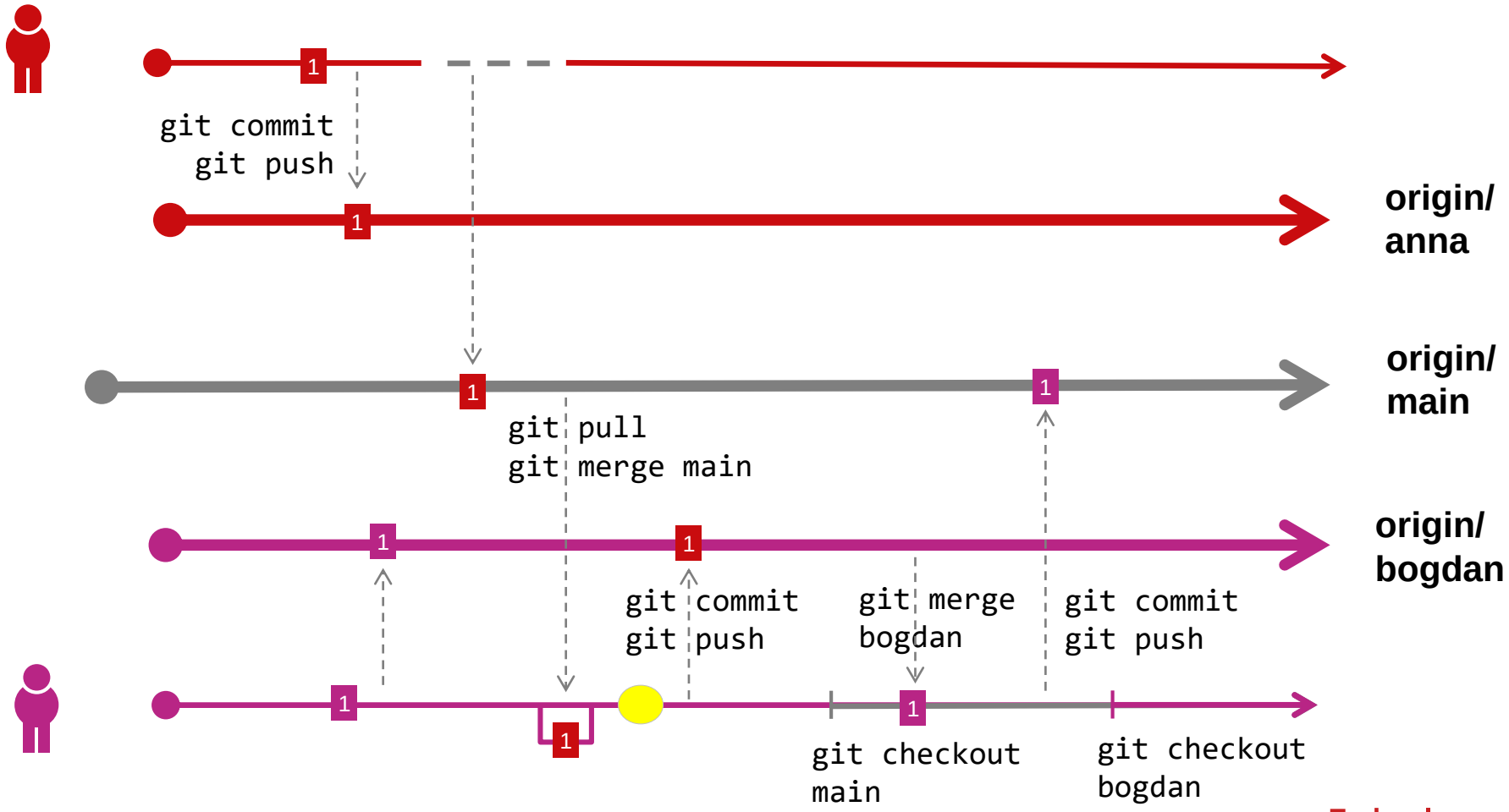
(For simplicity's sake, only shown with 2 team members, but of course also works with >2)



3) Working with Feature Branches (2)



3) Working with Feature Branches (3)



3) Working with Feature Branches (4)

- For simplicity reasons, the previous slides showed a continuous feature branch per developer.
  - This, however, is **not how it works in practice**.

- You should create a **dedicated feature branch** for each major step of the development.
  - Make sure to use a high frequency approach
  - commit / pull / resolve conflicts 1x per day

- You better agree on a **naming convention** for feature branches
- Also, you should agree on a simple governance
  - Make sure to have at least one person **review your merge request**

4) Continuous Integration

Continuous Integration =

- „Healthy Branches“
  - Thoroughly tested
  - Will definitely be incorporated into the main branch

  +

- High Frequency Integration

Feature Branches vs. Continuous Integration

Feature Branches

- Works for a loosely collaborating team
- Allows quality gate before transfer to main
- Fewer merges overall, but more difficult to plan
- Merges have the potential to fail!
- Less frequent integration => less productive overall due to higher friction losses

Continuous Integration

- More frequent merges, but less merge effort
- Higher productivity (overall)
- Makes refactoring easier (lower threshold)
- Only works with a team that knows each other well, and works closely together
- Requires healthy branches with good test coverage