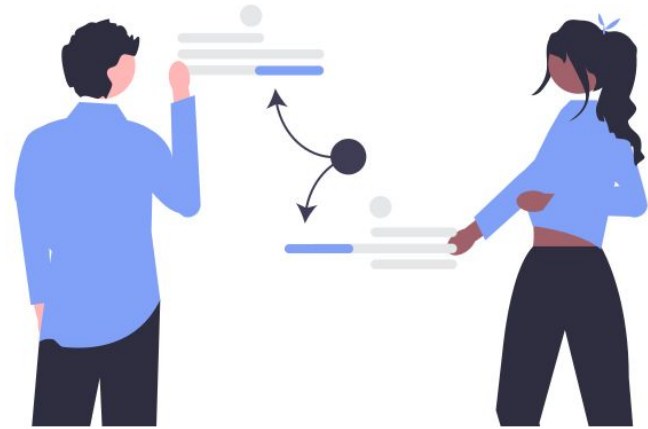


# Best Practises für Git-Vorgehensmodelle in größeren Teams

Vortrag für das Modul “Software Engineering”

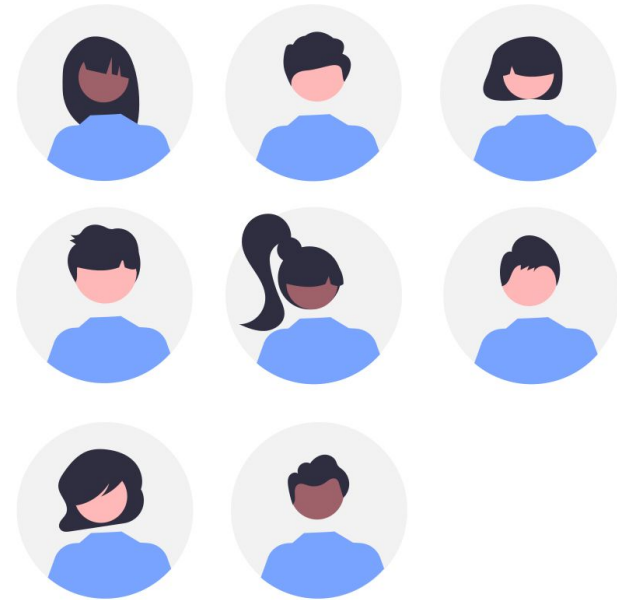
# Probleme im Umgang mit Git

- Fehler auf `main`
- komplizierter Merge-Prozess, Merge-Konflikte
- lange Release-Zyklen
- Beitrag zu interessanten Projekt

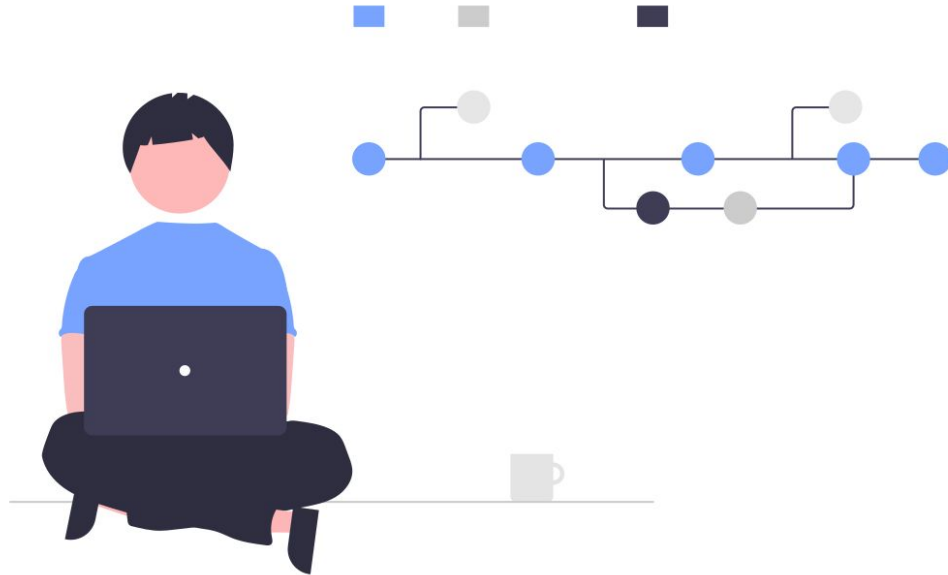


# Forderungen an Vorgehensmodelle

- effiziente Zusammenarbeit durch einheitlichen Änderungsfluss
- passend zum Entwicklungsprozess, Projekt und Team
- Vermeidung von Konflikten
- stabile Codebasis, CI/CD
- passende Delivery-Zeiten
- Kommunikation im Team



# Vorgehensmodelle



# Szenario

- viele Features gleichzeitig in der Entwicklung
- eine Vorgangstracking-Software genutzt wird
- parallele Verwaltung mehrerer Versionen

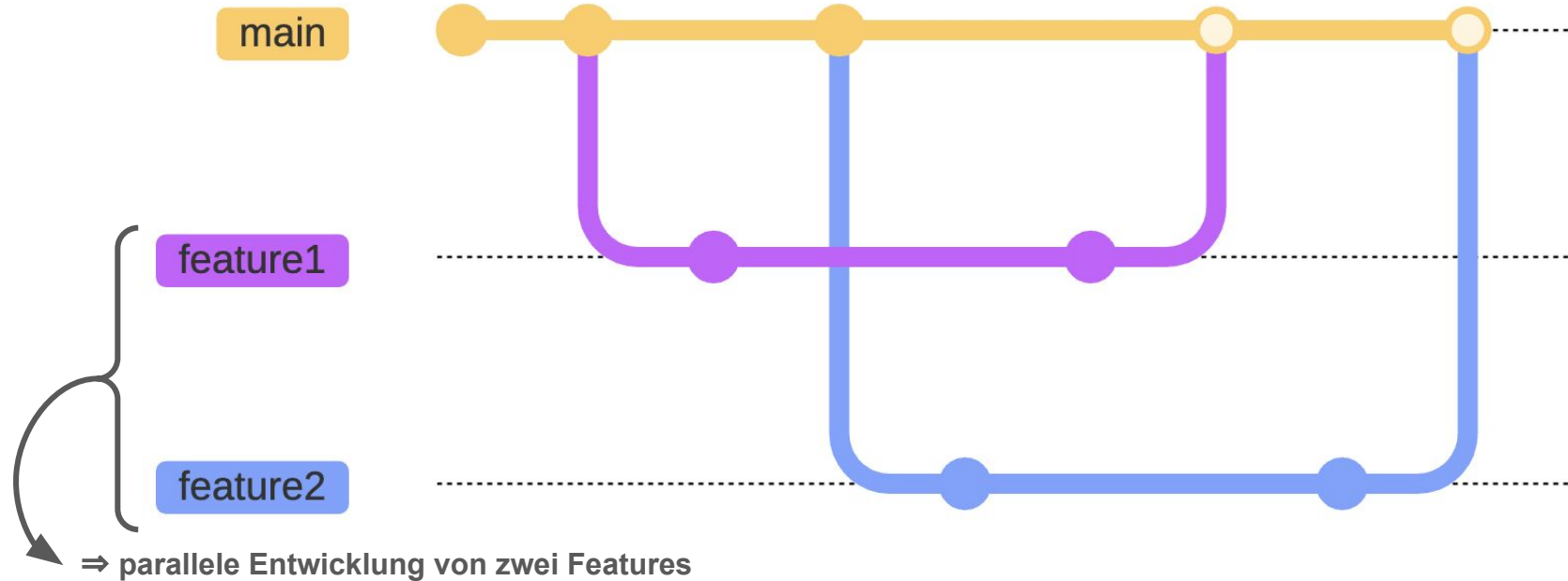
[[Atlassian 2023](#), [Dumitrescu 2021](#)]



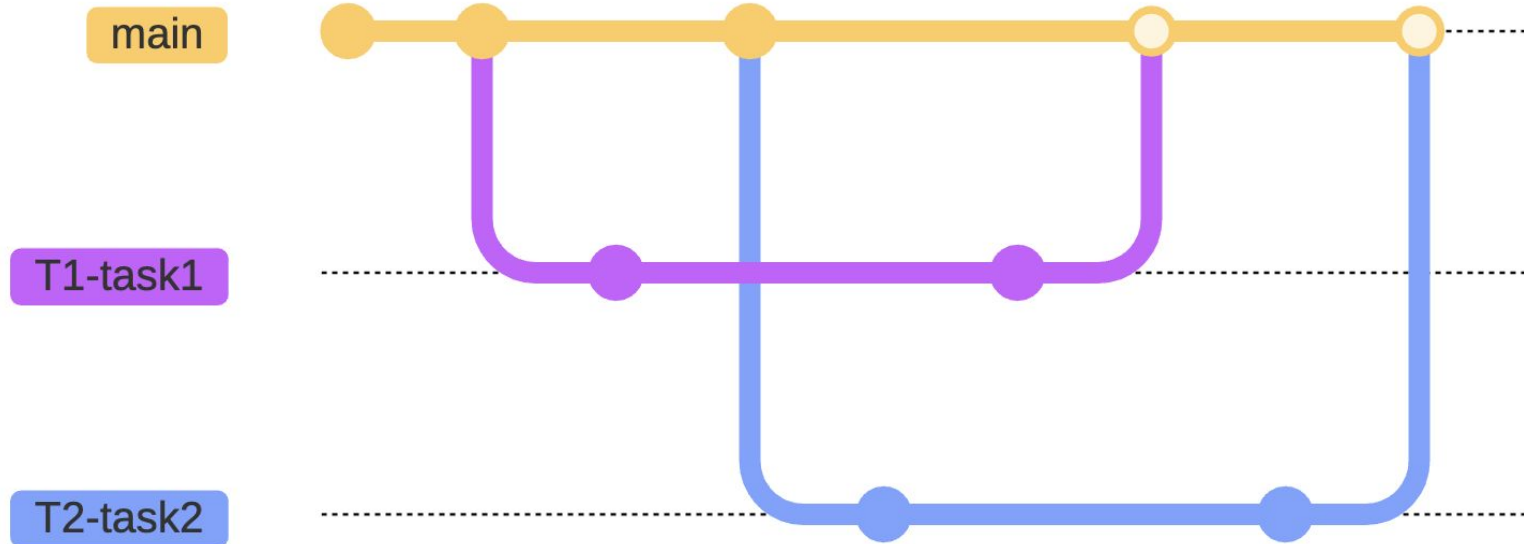
**Branches**

⇒ gekapselte Umgebung

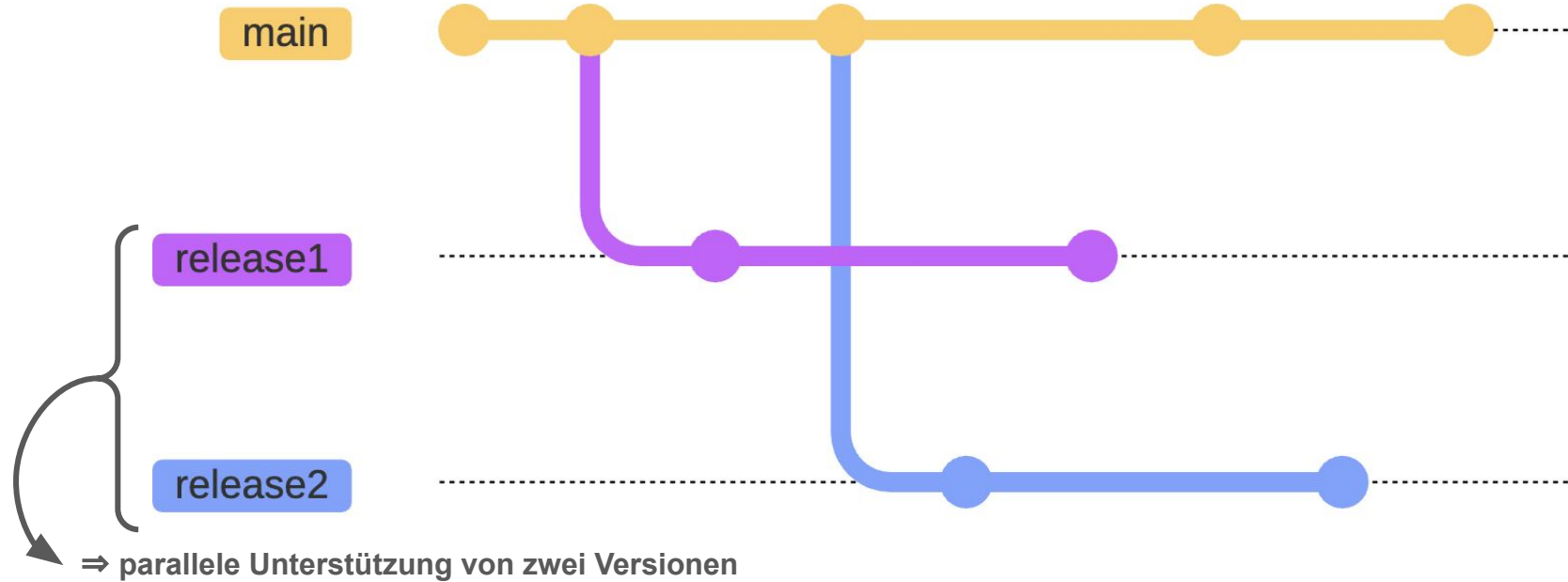
# Branching-Strategien > Feature-Branching



# Branching-Strategien > Task-Branching



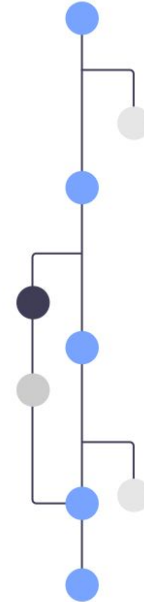
# Branching-Strategien > Release-Branching





# Probleme & Szenario

- keine Trennung von Prod und Dev  
⇒ Fehler fallen erst in Prod auf
- stabile Releases
- Unterstützung einer Version
- großes Team, >100 Entwickler:innen
- Release-Vorbereitungen abgetrennt vom Entwicklungsprozess

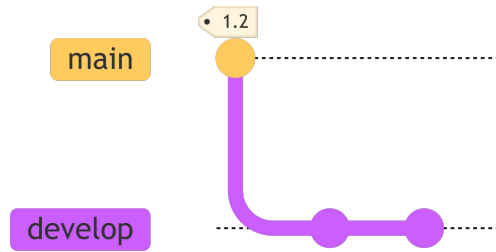


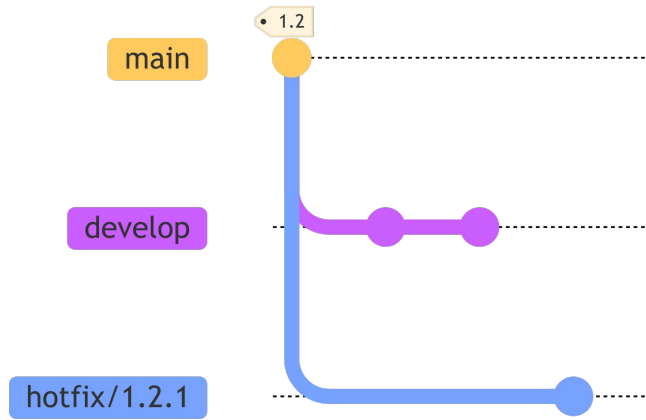
**Gitflow**

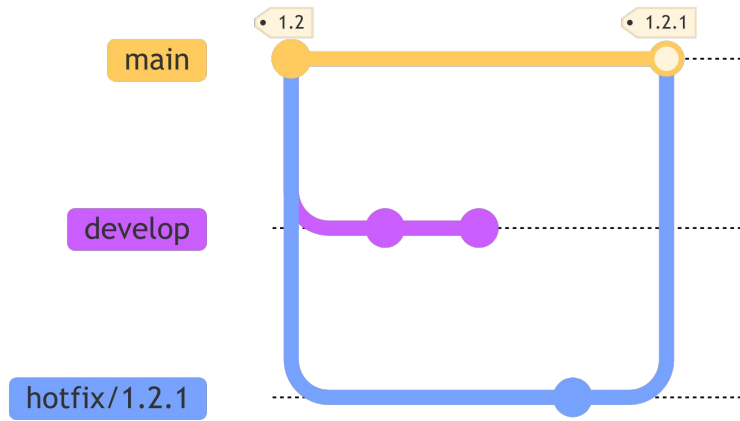
# Gitflow

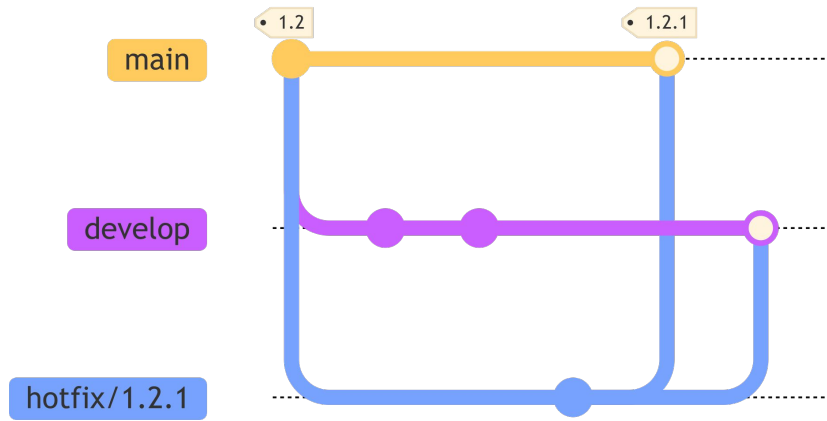
- Release-zentrierter Ansatz
- dedizierte, spezifische Rolle für jeden Branch

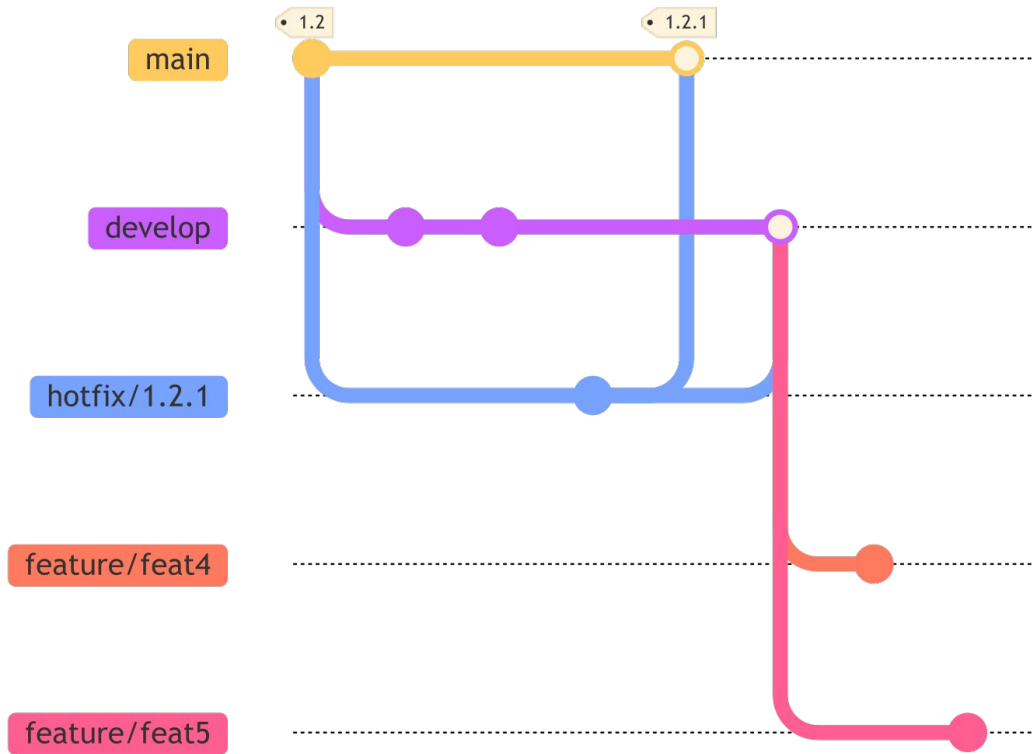
Branch	Eigenschaften
main	langlebig, stets produktionsreif, Commit := Release, Release-Historie

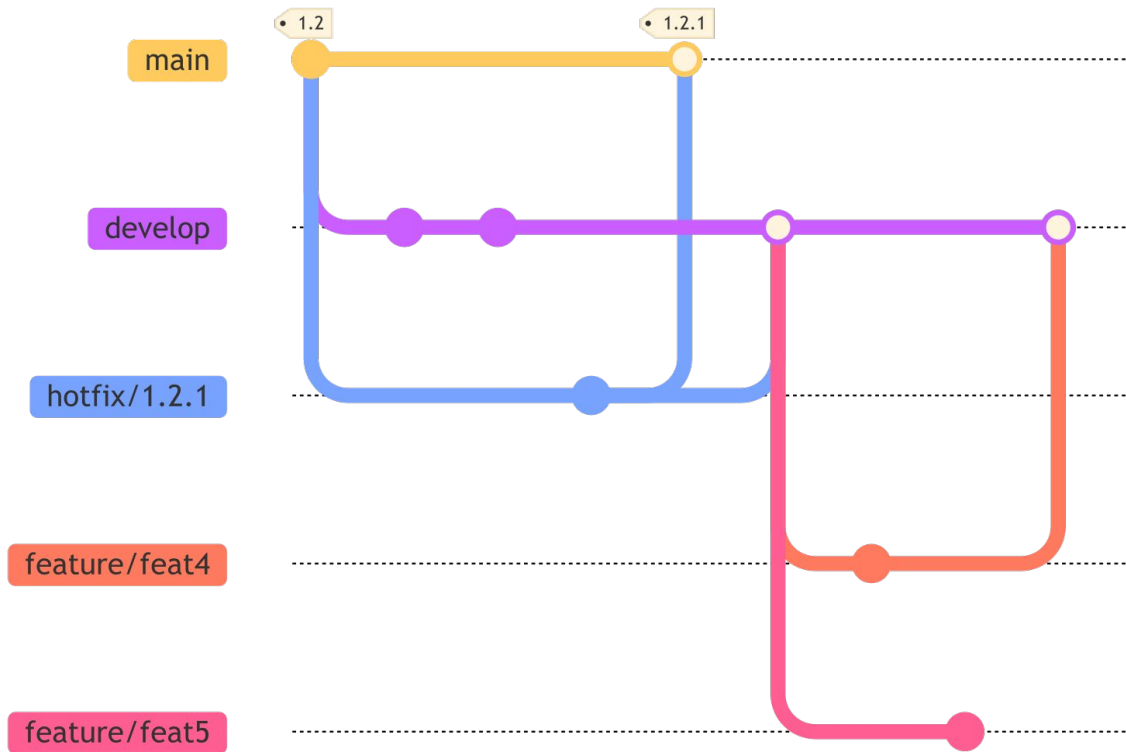




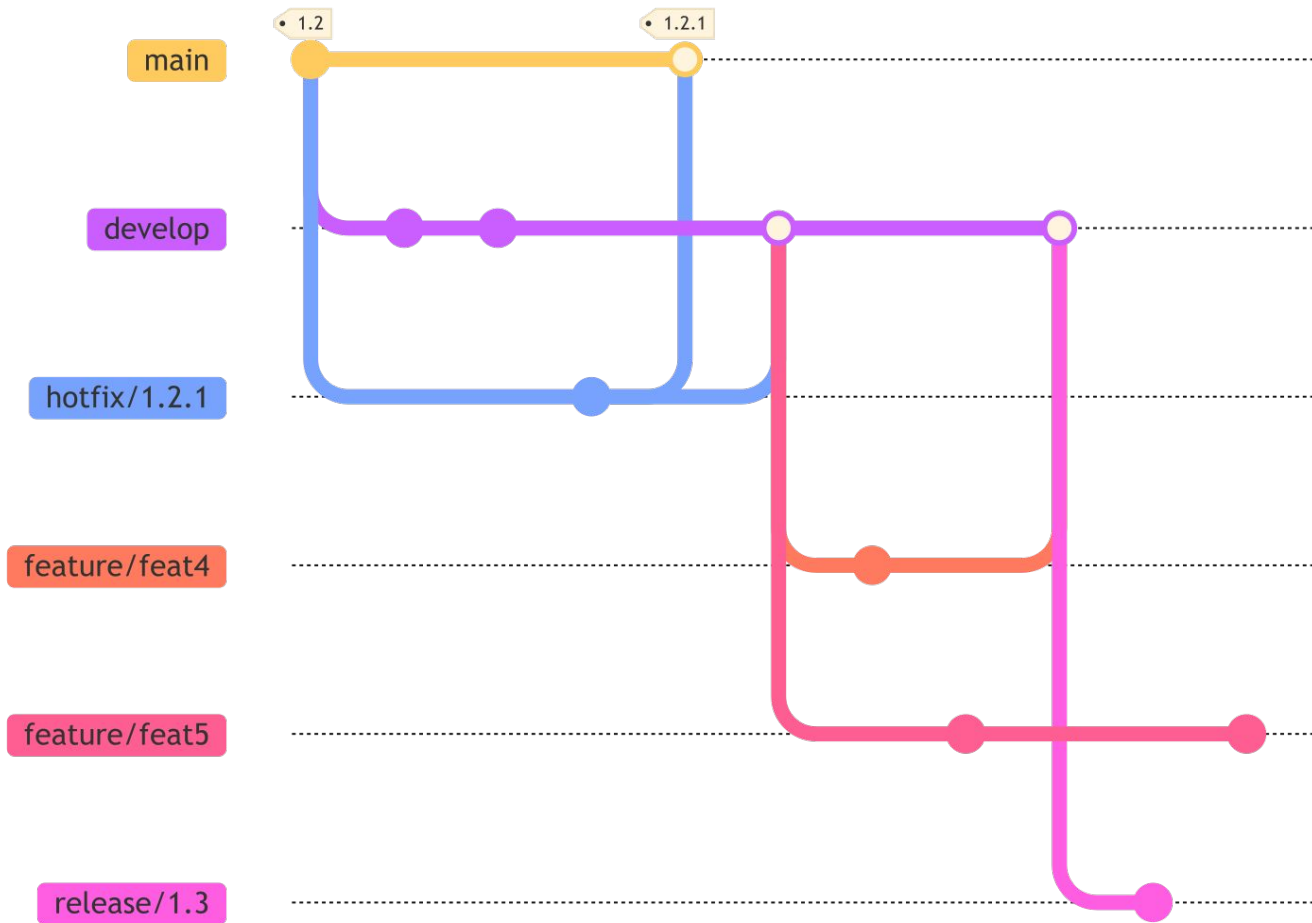


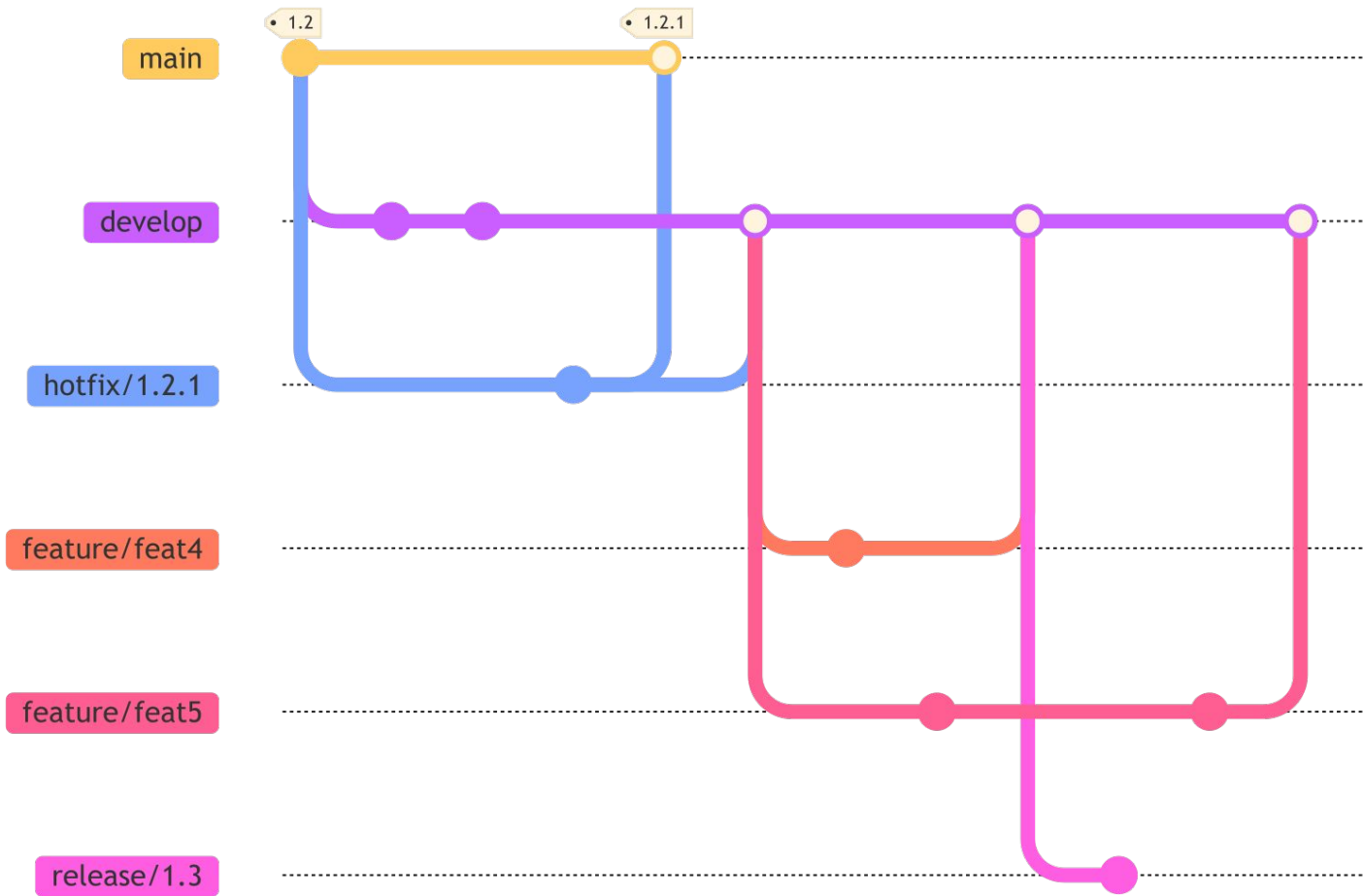


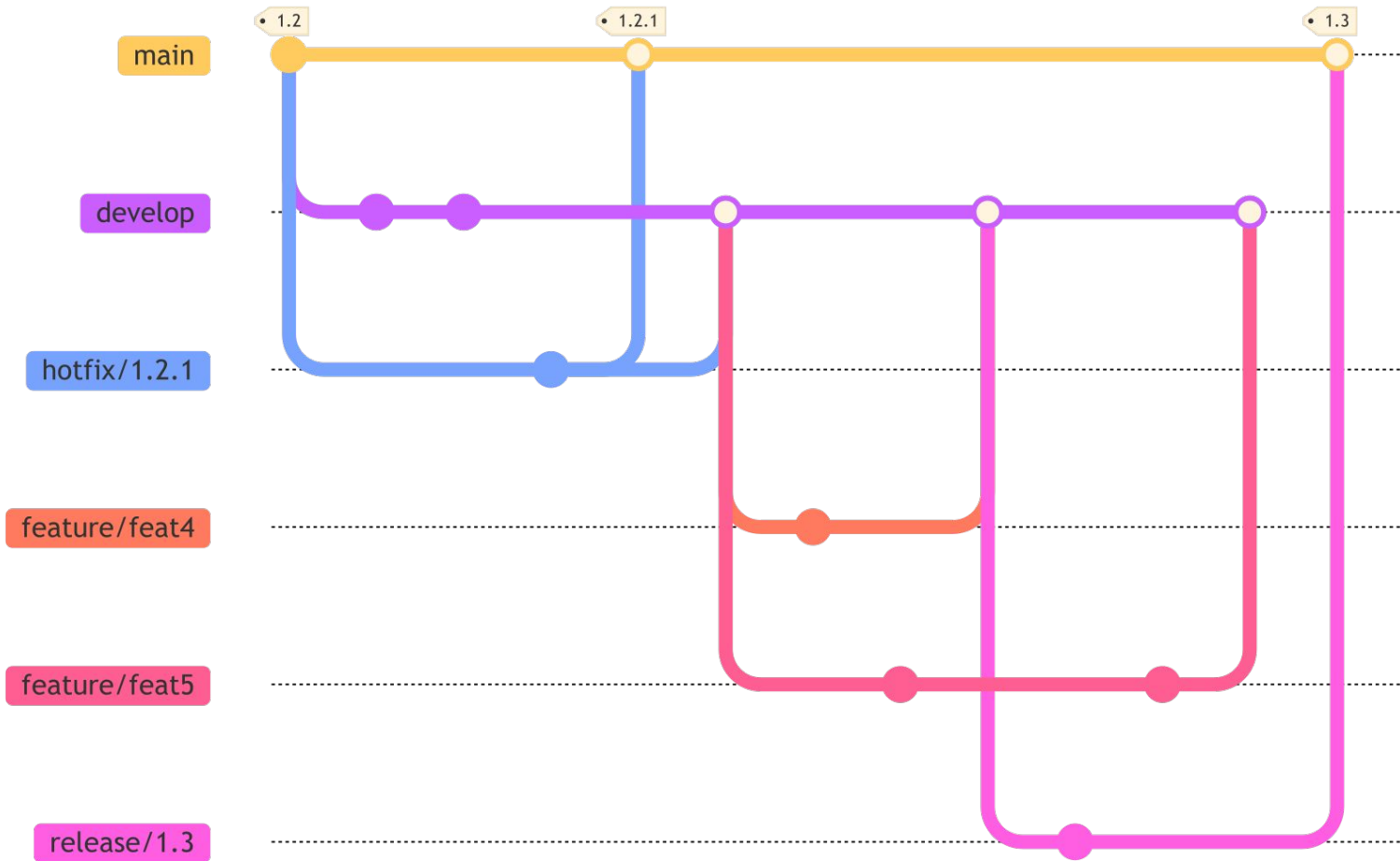


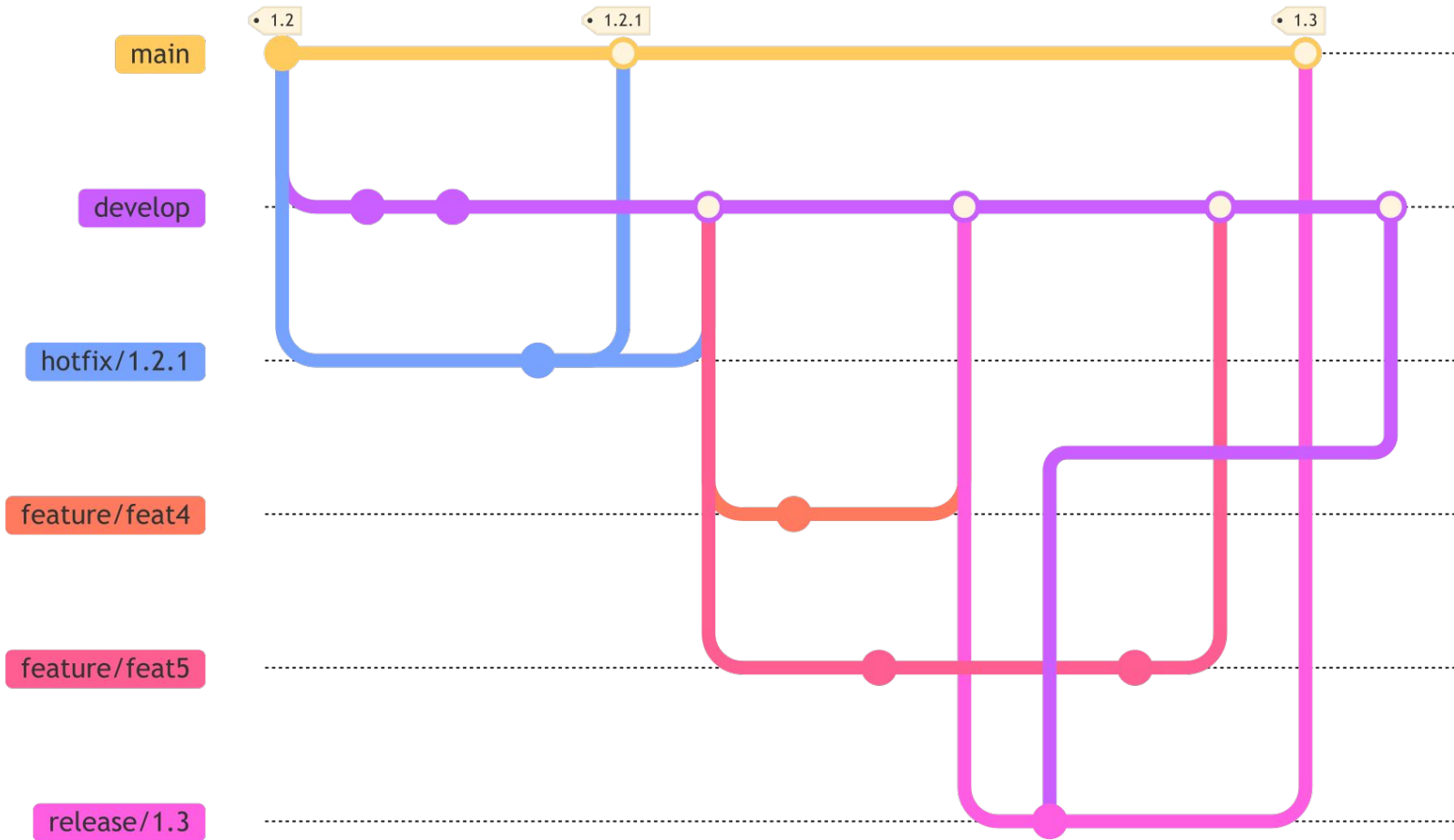












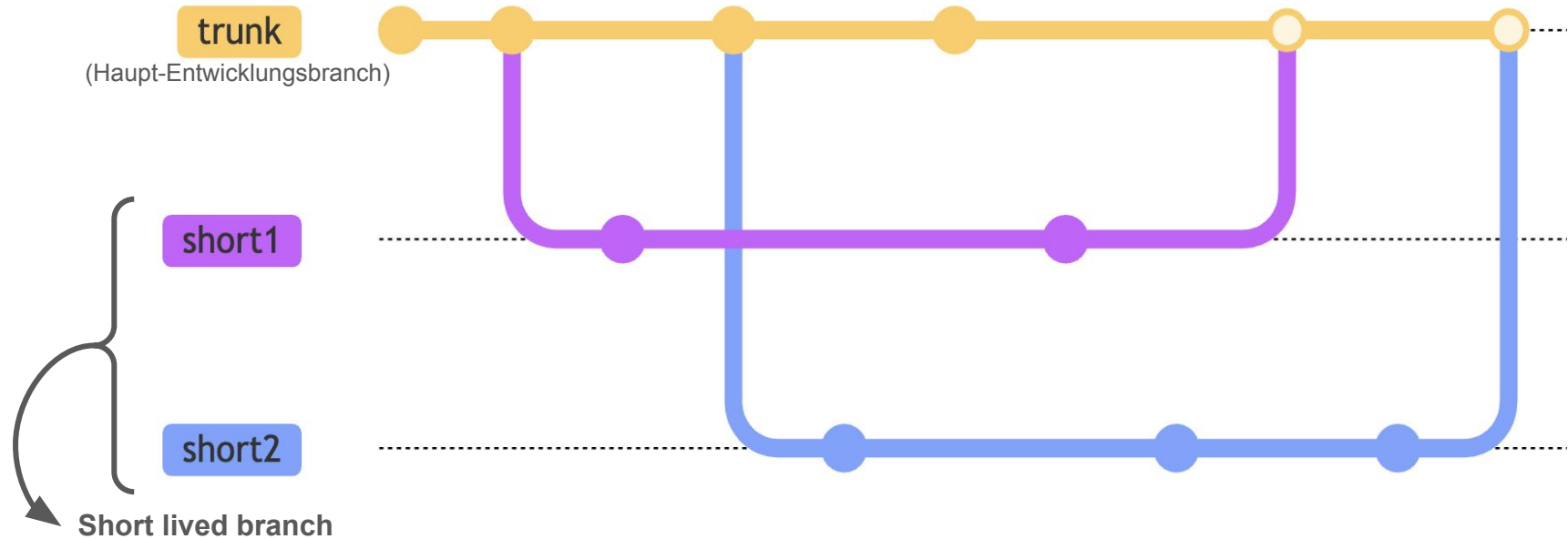
# Szenario

- neues Projekt
- schnelles Kunden-Feedback erwünscht  
→ kontinuierliche Delivery
- kleines, hoch erfahrenes Team
- Vermeidung der Merge-Hell



## Trunk-based Development

# Trunk-based Development



# Trunk-based Development

## CI/CD

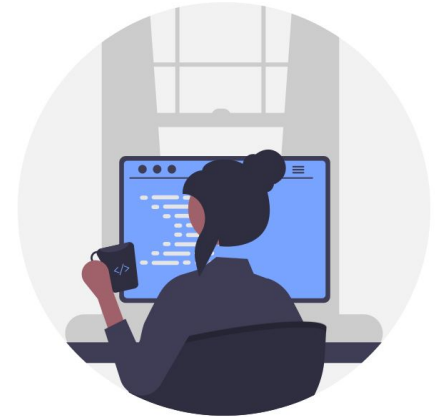
automatische Tests,  
Optimierung von  
Deployment-Zeiten

## Best Practices

1x pro Tag in Trunk integrieren,  
max. 3 kurzlebige Branches

## Feature Flags

Ein- und Ausschalten  
von Features



# Szenario

- Verwalter:in eines Open-Source-Projekts
- auf uns unbekannte Entwickler:innen angewiesen
- müssen Codequalität sicherstellen
- *kreatives Chaos*

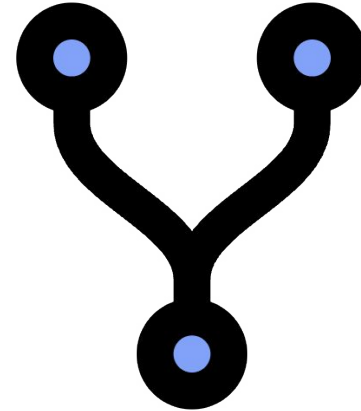


Forking-Workflow



# Forking-Workflow

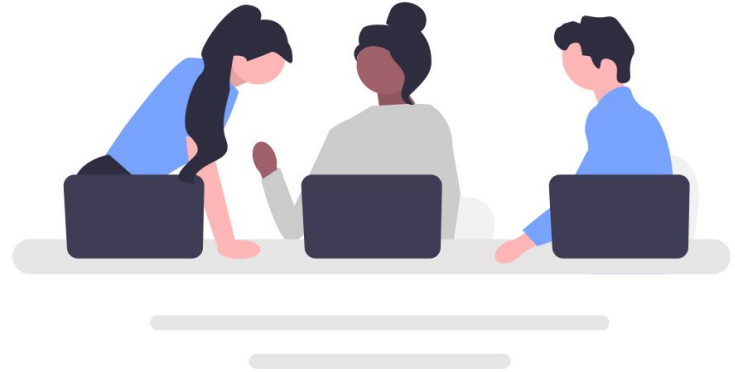
- Erstellen eines *Forks*
- Entwicklung gekapselt im Fork
- Vorschlagen der Änderungen als Pull Request in das Haupt-Repository
- häufig in Open-Source-Projekten genutzt



# Bewertung

- kognitiver Overhead
- Kultur und Skalierbarkeit des Teams
- Release-Zyklus und Delivery
- Merge-Konflikte und -Prozess
- Übersichtlichkeit
- Stabilität und Erfahrung
- Release-Vorbereitungen

⇒ siehe [Paper](#)



# Bewertung

- **kleines**, hoch erfahrenes Team
- kontinuierliche Delivery
- einfacher Merge-Prozess

Trunk-based  
Development



Gitflow



Forking-  
Workflow



# Bewertung

- **großes**, hoch erfahrenes, **koordiniertes Team**
- kontinuierliche Delivery
- **klare Trennung von Prod und Dev**

Gitflow



Forking-  
Workflow



Trunk-based  
Development



# Bewertung

- großes Team mit **unterschiedlichen** Erfahrungen
- kontinuierliche Delivery
- klare Trennung von Prod und Dev

Gitflow



Forking-  
Workflow



Trunk-based  
Development



# Bewertung

- parallele Unterstützung von **mehreren** Versionen
- **Stabilität**
- klare Trennung von Prod und Dev

**Gitflow**  
(mit Abwandlung)



**Forking-  
Workflow**

**Trunk-based  
Development**



# Bewertung

- großes Team mit unterschiedlichen Erfahrungen
- wenig Koordination / *Kreatives Chaos*
- keine Verpflichtungen
- keine Bekanntheit

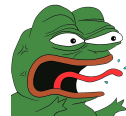
Forking-  
Workflow



Gitflow



Trunk-based  
Development



# Fazit



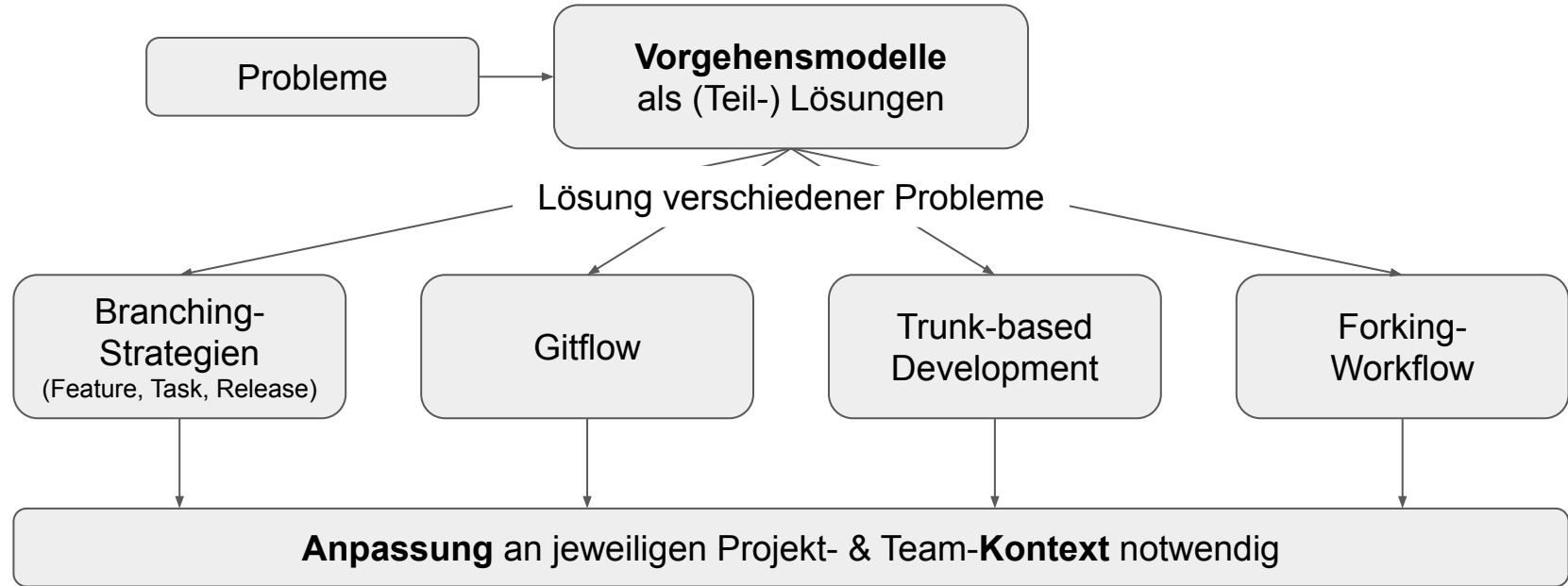
Vincent Driessen  
(Gitflow-Entwickler)

“Denken Sie abschließend immer daran, dass es **keine Allheilmittel** gibt. Berücksichtigen Sie Ihren **eigenen Kontext.**”

[[Driessen 2012](#)]



# Zusammenfassung & Diskussion



# Literatur

Atlassian. 2023. Comparing Git workflows: What you should know. <https://www.atlassian.com/git/tutorials/comparing-workflows>

Atlassian. 2023. Hintergrundwissen zur Branching-Strategie. <https://www.atlassian.com/de/agile/software-development/branching>

Sorin Dumitrescu. 2021. What Is a Feature Branch + How they Improve the Dev Process. <https://www.bunnyshell.com/blog/what-is-a-feature-branch/>

GitHub Inc. 2023b. Contributing to projects. <https://ghdocs-prod.azurewebsites.net/en/get-started/quickstart/contributing-to-projects>

GitHub Inc. 2023c. Informationen zu Forks. <https://docs.github.com/de/pull-requests/collaborating-with-pull-requests/working-with-forks/about-forks>

GitLab B.V. 2023. What is a Git workflow? <https://about.gitlab.com/topics/version-control/what-is-git-workflow/>

Vincent Driessen. [n. d.]. Nvie2x. <https://nvie.com/img/nvie@2x.jpg>

Vincent Driessen. 2010. A successful Git branching model. <http://nvie.com/posts/a-successful-git-branching-model>

Vincent Driessen. 2012. Git-Flow. <https://github.com/nvie/gitflow>

JetBrains s.r.o. 2023. Was ist trunkbasierte Entwicklung? | TeamCity CI/CD-Leitfaden. <https://www.jetbrains.com/de-de/teamcity/ci-cd-guide/concepts/trunk-based-development/>

Split Software Inc. 2023. A COMPLETE GUIDE TO TRUNK-BASED DEVELOPMENT. <https://www.split.io/blog/a-complete-guide-to-trunk-based-development/>