GovTool
Git(Hub) workflow proposal

by example

User story

As a team leader

I want to coordinate delivery of features to staging,

so they can be reviewed by product owners.

Case study

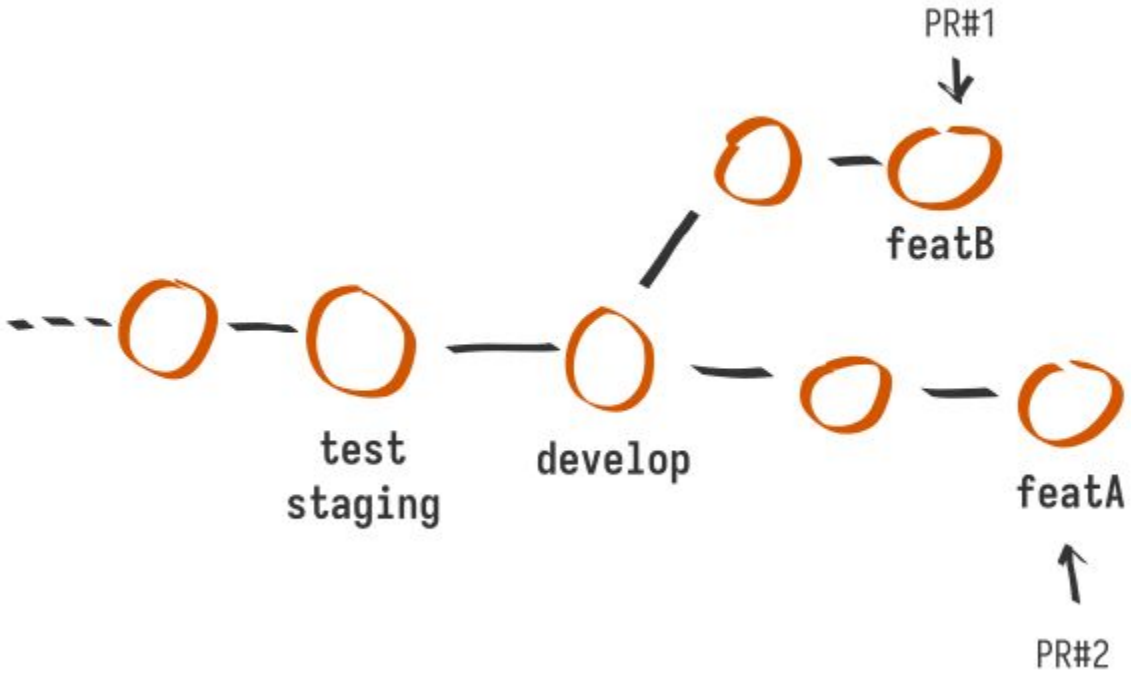
- Two features are ready.
- Only one is picked to be delivered.
- The picked feature is reviewed and QA tested.
- The picked feature end up on staging/preprod to be further received by PO.

Scenario #1

current workflow

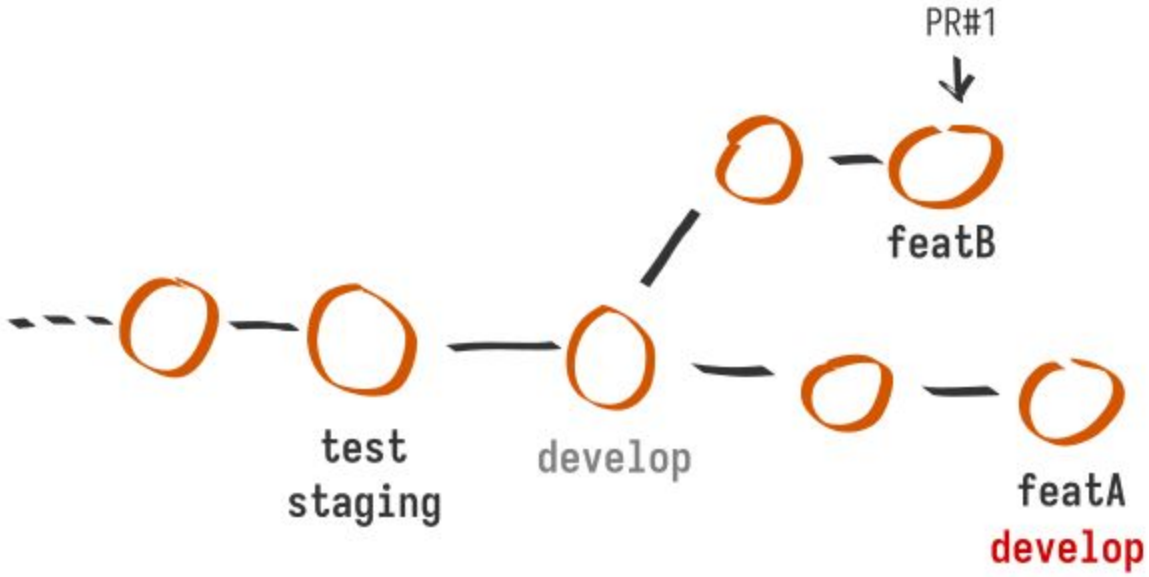
Scenario #1: current workflow

- 2 developers are working on 2 different tasks.



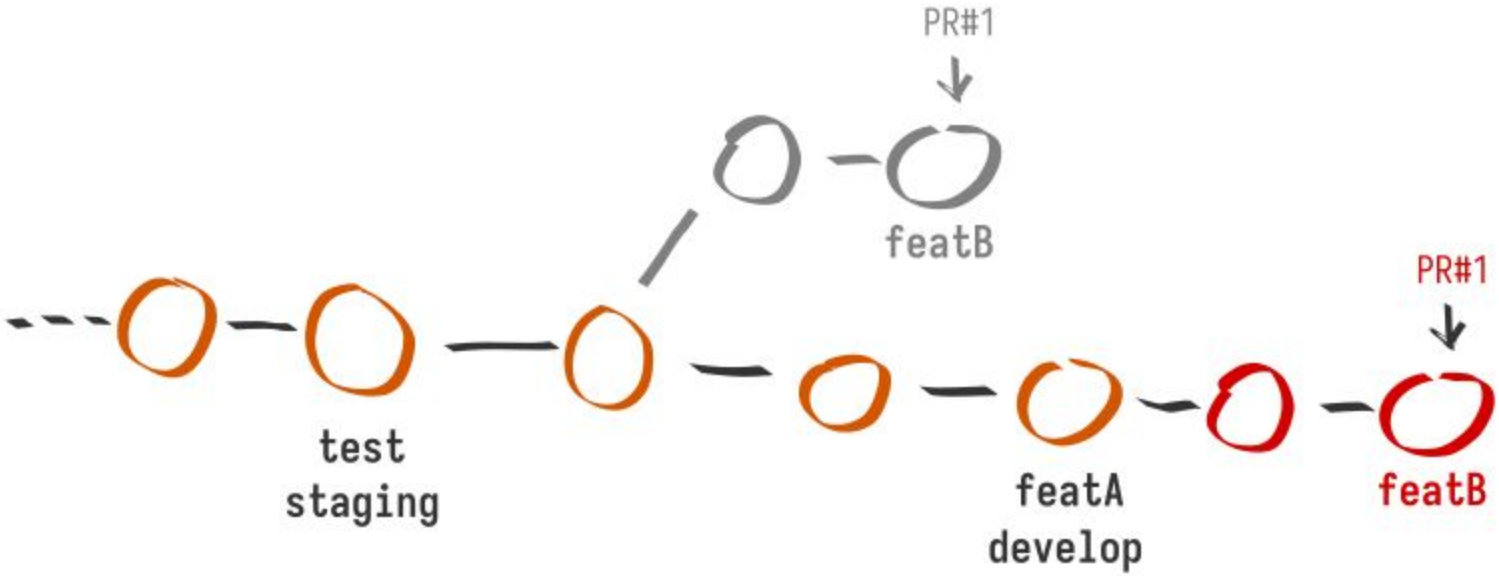
Scenario #1: current workflow

- 2 developers are working on 2 different tasks.
- **featA** is merged after a review.



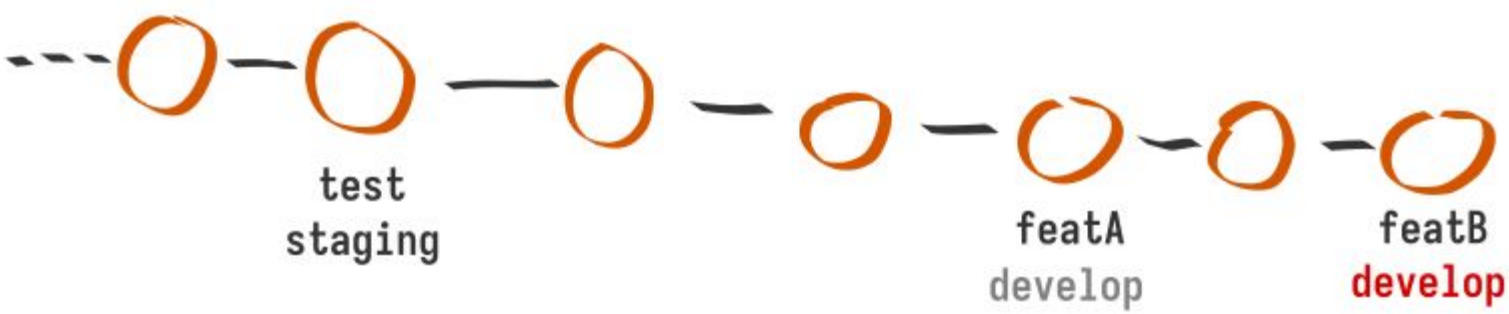
Scenario #1: current workflow

- 2 developers are working on 2 different tasks.
- **featA** is merged after a review.
- **featB** has to be rebased to actual **develop**.



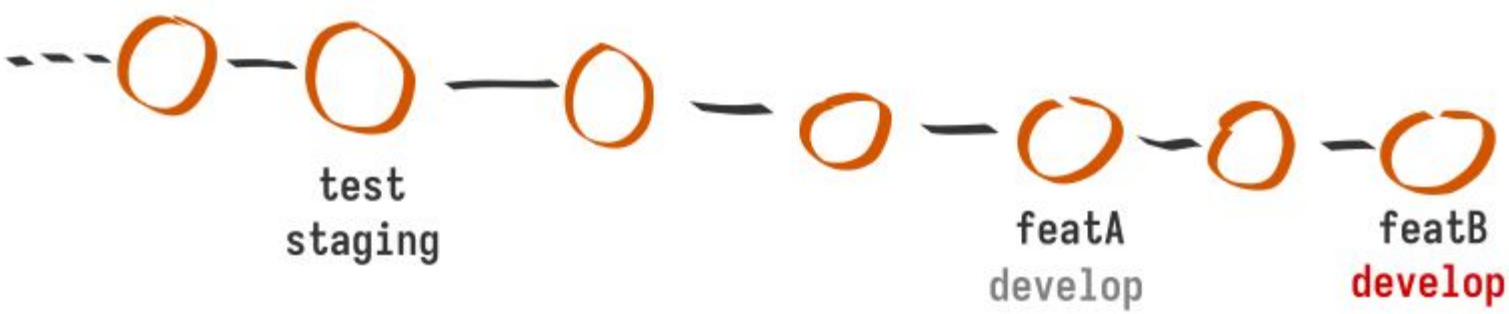
Scenario #1: current workflow

- 2 developers are working on 2 different tasks.
- **featA** is merged after a review.
- **featB** has to be rebased to actual **develop** and **featB** is merged after review to **develop**.



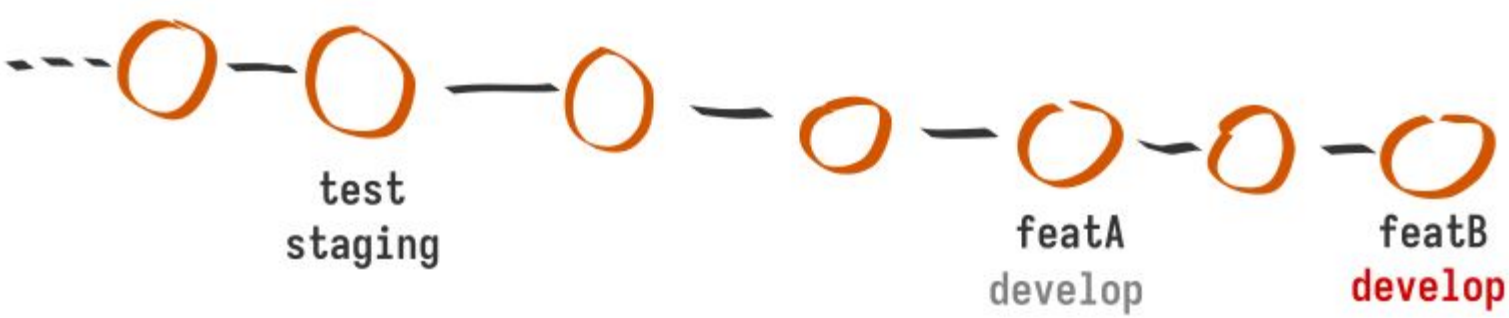
Scenario #1: current workflow

- 2 developers are working on 2 different tasks.
- **featA** is merged after a review.
- **featB** has to be rebased to actual **develop** and **featB** is merged after review to **develop**.
- A decision is made to deliver only **featA**.



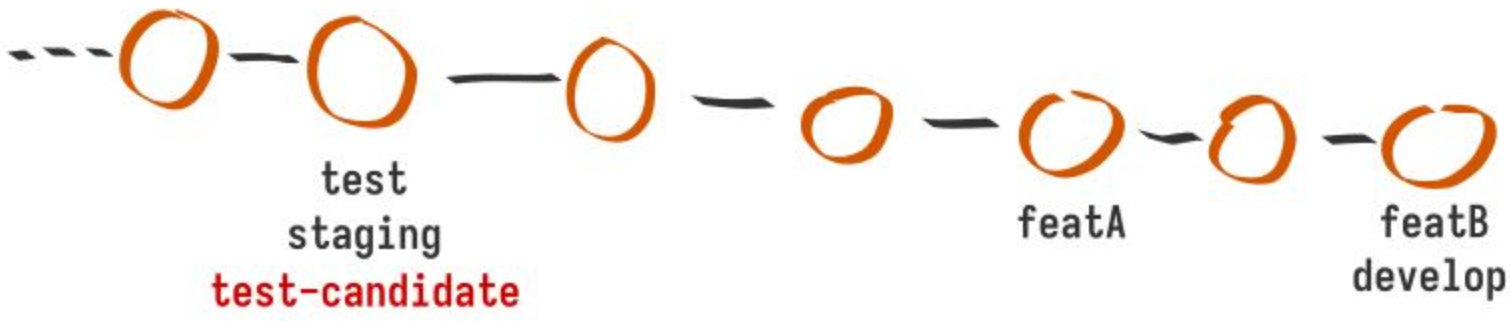
Scenario #1: current workflow

- 2 developers are working on 2 different tasks.
- **featA** is merged after a review.
- **featB** has to be rebased to actual **develop** and **featB** is merged after review to **develop**.
- A decision is made to deliver only **featA**.
- To deliver to **test** we need to:
  - create a branch from **test**,
  - cherry-pick **featA** commits,
  - merge to **test**.



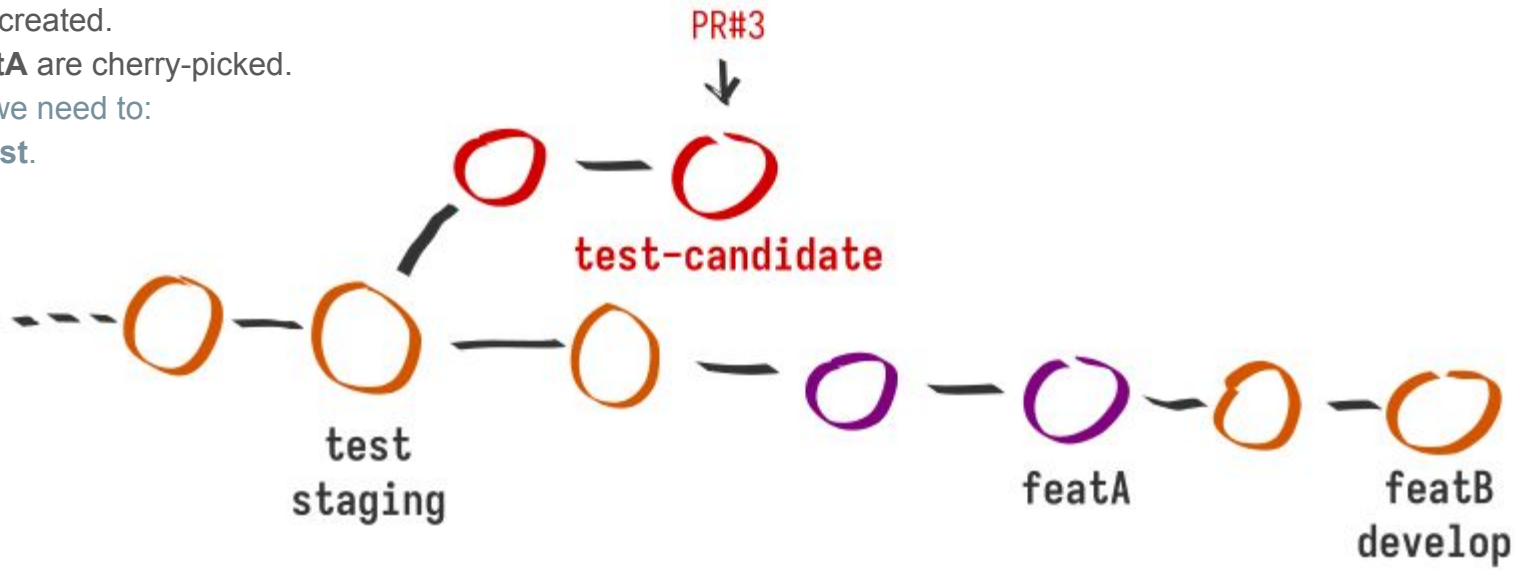
Scenario #1: current workflow

- 2 developers are working on 2 different tasks.
- **featA** is merged after a review.
- **featB** has to be rebased to actual **develop** and **featB** is merged after review to **develop**.
- A decision is made to deliver only **featA**.
- **test-candidate** is created.
- To deliver to **test** we need to:
  - cherry-pick **featA** commits,
  - merge to **test**.



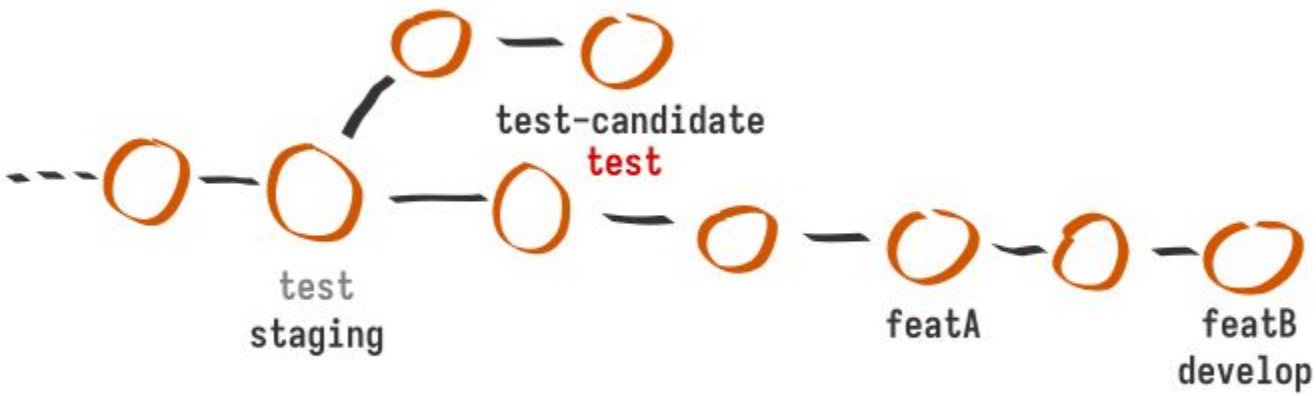
Scenario #1: current workflow

- 2 developers are working on 2 different tasks.
- **featA** is merged after a review.
- **featB** has to be rebased to actual **develop** and **featB** is merged after review to **develop**.
- A decision is made to deliver only **featA**.
- **test-candidate** is created.
- Commits from **featA** are cherry-picked.
- To deliver to **test** we need to:
  - merge to **test**.



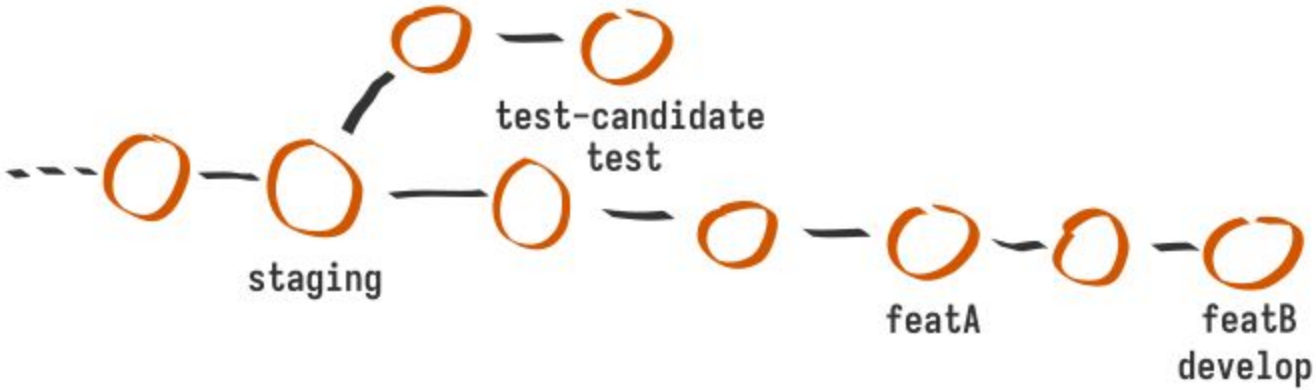
Scenario #1: current workflow

- 2 developers are working on 2 different tasks.
- **featA** is merged after a review.
- **featB** has to be rebased to actual **develop** and **featB** is merged after review to **develop**.
- A decision is made to deliver only **featA**.
- **test-candidate** is created.
- Commits from **featA** are cherry-picked.
- **test-candidate** is merged after a review to **test**.



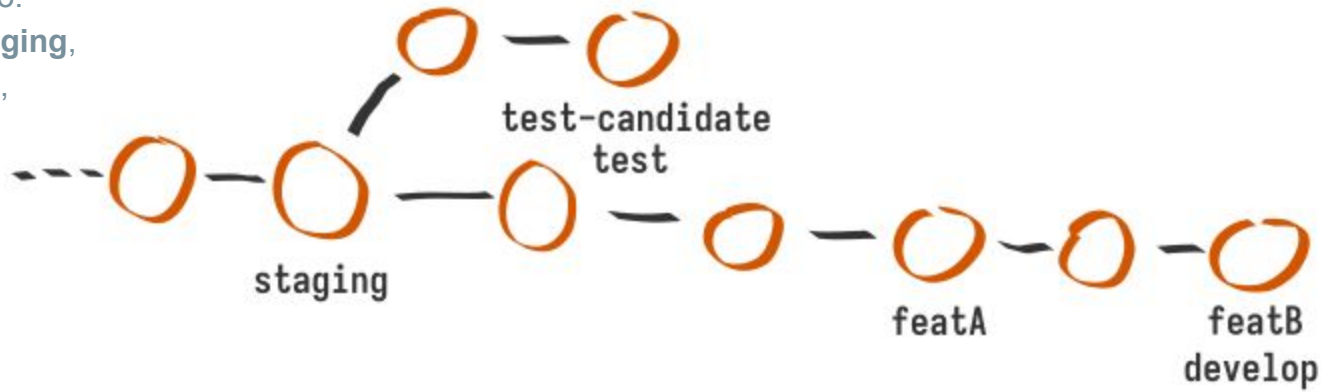
Scenario #1: current workflow

- 2 developers are working on 2 different tasks.
- **featA** is merged after a review.
- **featB** has to be rebased to actual **develop** and **featB** is merged after review to **develop**.
- A decision is made to deliver only **featA**.
- **test-candidate** is created.
- Commits from **featA** are cherry-picked.
- **test-candidate** is merged after a review to **test**.
- QA tests are performed on **test**.



Scenario #1: current workflow

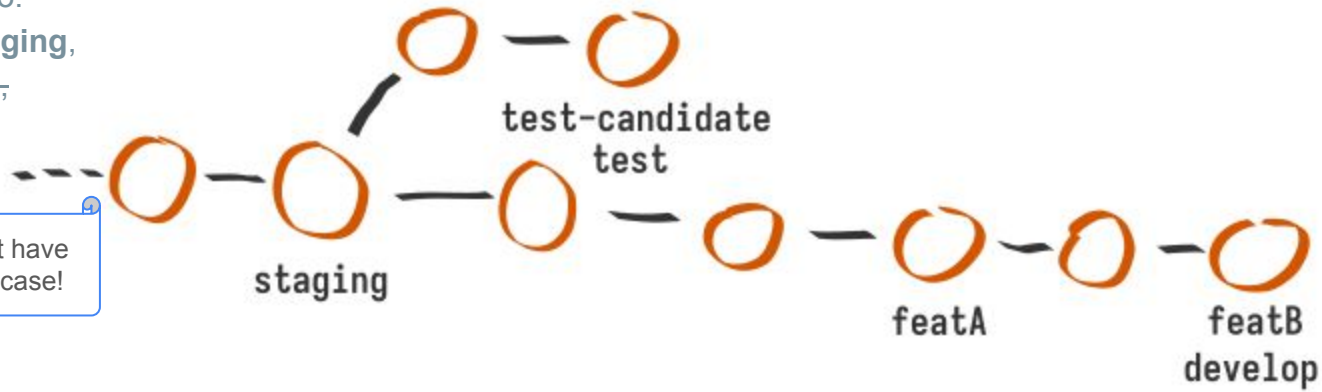
- 2 developers are working on 2 different tasks.
- **featA** is merged after a review.
- **featB** has to be rebased to actual **develop** and **featB** is merged after review to **develop**.
- A decision is made to deliver only **featA**.
- **test-candidate** is created.
- Commits from **featA** are cherry-picked.
- **test-candidate** is merged after a review to **test**.
- QA tests are performed on **test**.
- To deliver to **staging** we need to:
  - create a branch from **staging**,
  - cherry-pick **test** commits,
  - merge to **staging**.



Scenario #1: current workflow

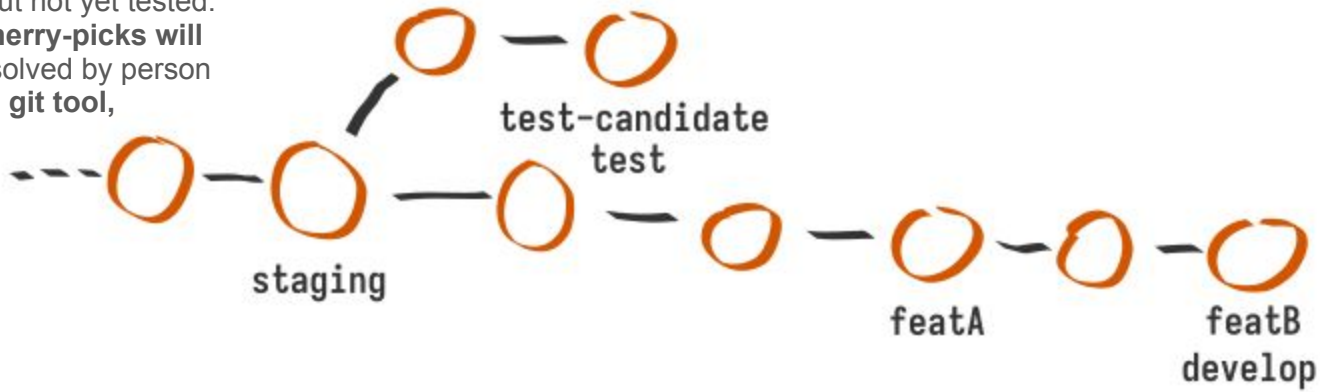
- 2 developers are working on 2 different tasks.
- **featA** is merged after a review.
- **featB** has to be rebased to actual **develop** and **featB** is merged after review to **develop**.
- A decision is made to deliver only **featA**.
- **test-candidate** is created.
- Commits from **featA** are cherry-picked.
- **test-candidate** is merged after a review to **test**.
- QA tests are performed on **test**.
- To deliver to **staging** we need to:
  - create a branch from **staging**,
  - ~~cherry-pick **test** commits,~~
  - merge **test** into **staging**,
  - merge to **staging**.

CAVEAT: In this simple scenario we don't have to cherry pick commits, but it is an edge case!



Scenario #1: current workflow - risks

- Every **decision to pick** only some selected **features** from the pool of all available features **leads to cherry-picking** of either regular commits or merge commits.
- **Cherry-picking creates a potential of human error** (missing parts of a feature, dependencies, etc.)
- **A pull request** crafted by cherry-picking features **has to be reviewed** because of the above.
- **The source branch** (like branch “develop” in example) **is never a continuation of the target branch** (like “test”), it is a fork. Therefore the **cherry-picking is necessary**.
- **This** burdensome process **is repeated on three different stages** of the current workflow: develop > test, test > staging and staging > beta.
- **Testing on reviewed features is on hold until the test-candidate is merged**, thus tasks are in limbo state where they are completed, but not yet tested.
- There is a possibility that **some cherry-picks will cause conflicts** that has to be resolved by person with a **knowledge not only in the git tool, but also in a domain, language, frameworks, libraries, etc.**

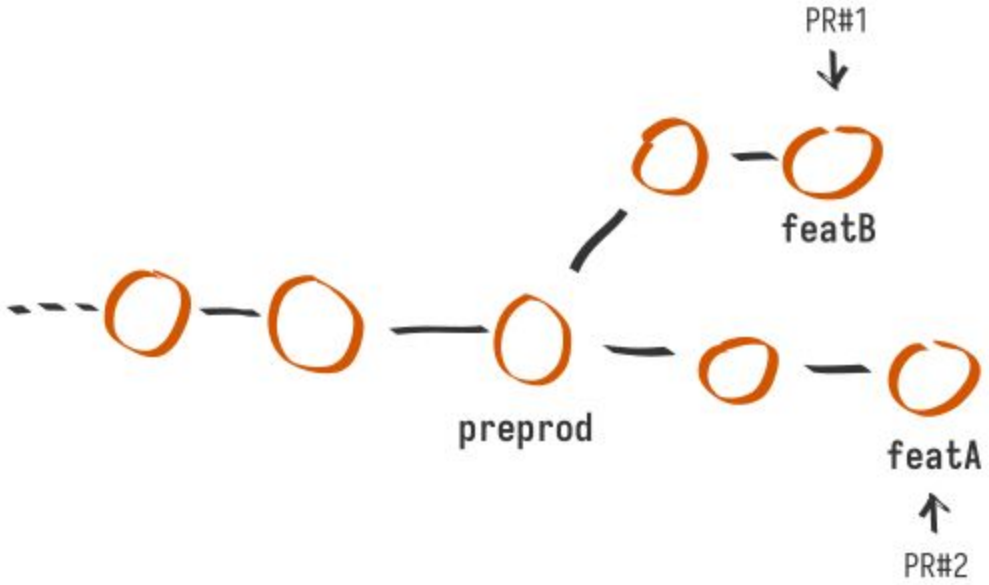


Scenario #2

proposal

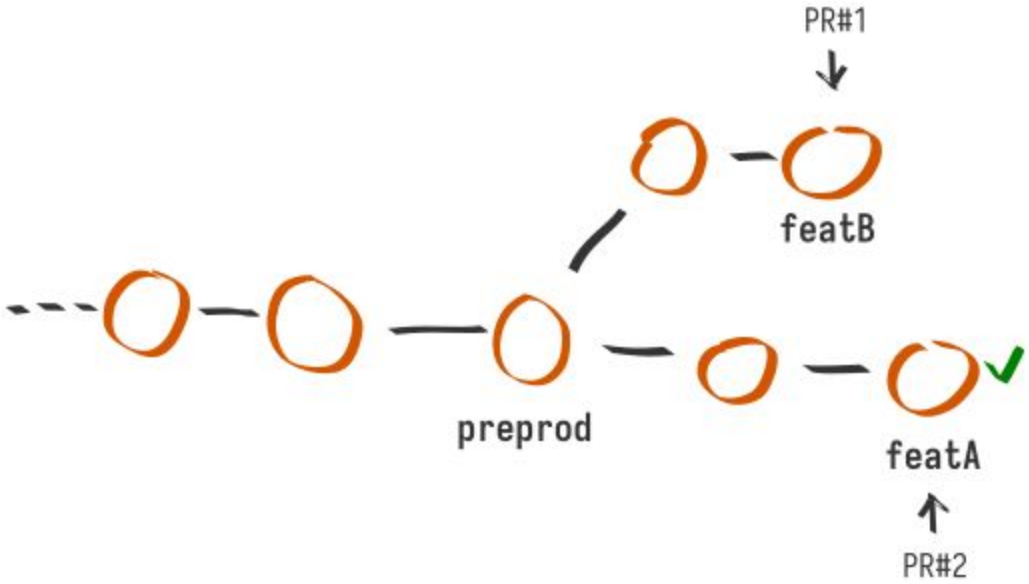
Scenario #2: proposal

- 2 developers are working on 2 different tasks.



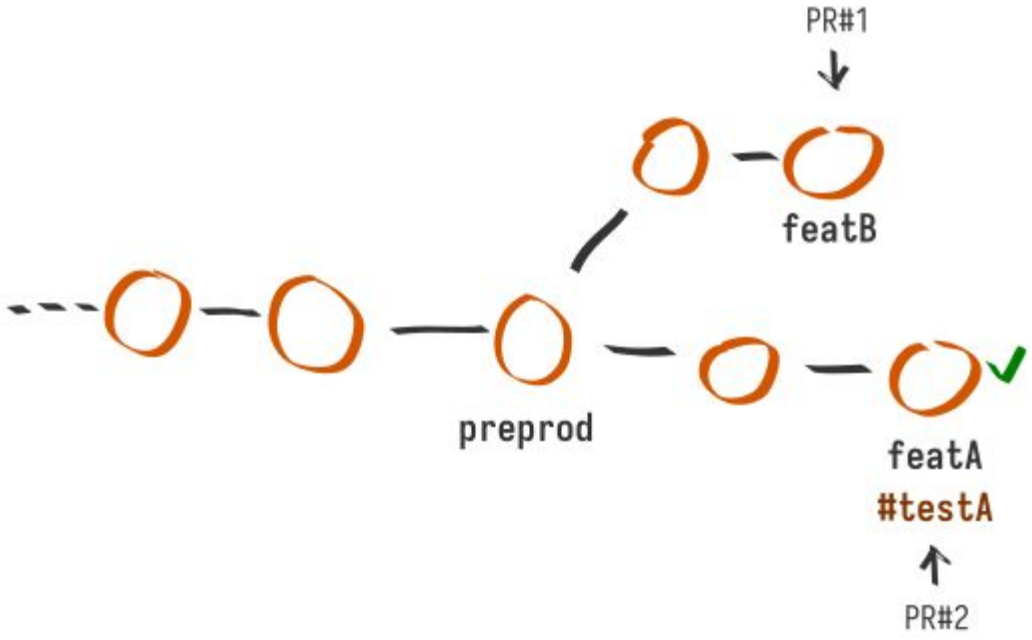
Scenario #2: proposal

- 2 developers are working on 2 different tasks.
- **featA** is reviewed.



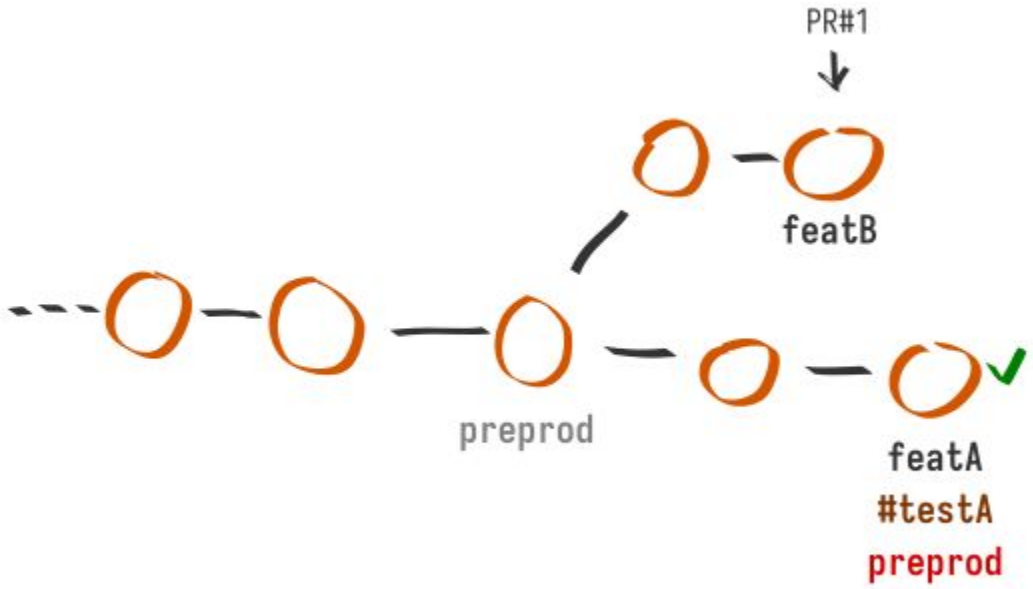
Scenario #2: proposal

- 2 developers are working on 2 different tasks.
- **featA** is reviewed.
- **featA** is deployed to be QA tested.
- At this point the **featA** is ready to be deployed on preprod (aka staging).



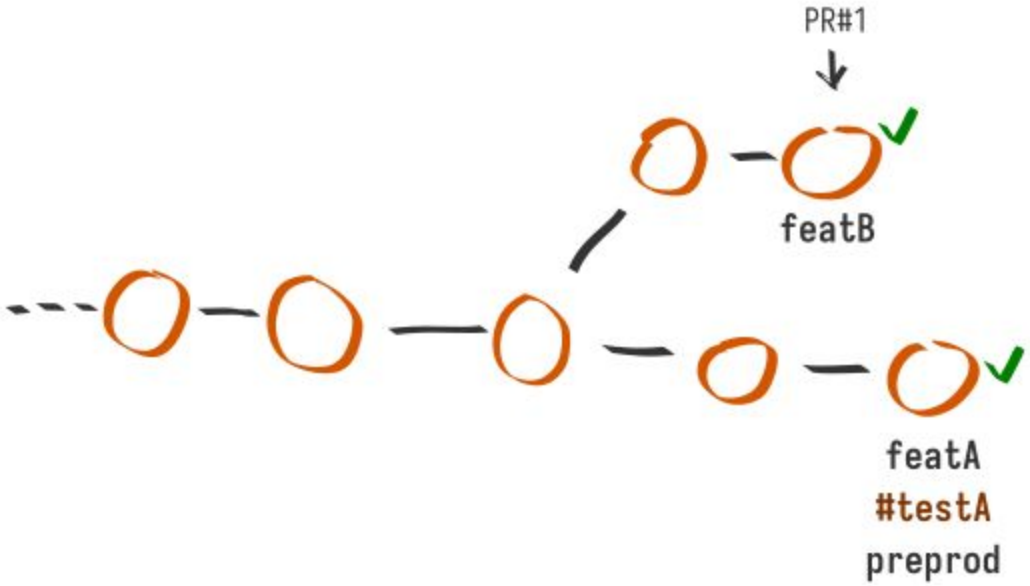
Scenario #2: proposal

- 2 developers are working on 2 different tasks.
- **featA** is reviewed.
- **featA** is deployed to be QA tested.
- **featA** is merged to **preprod**.



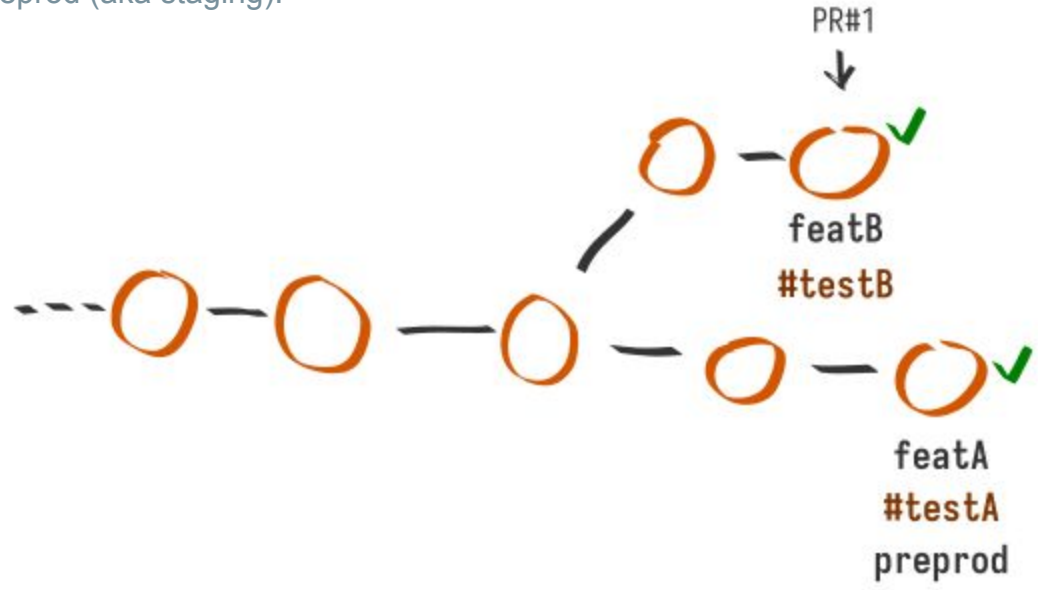
Scenario #2: proposal

- 2 developers are working on 2 different tasks.
- **featA** is reviewed.
- **featA** is deployed to be QA tested.
- **featA** is merged to **preprod**.
- **featB** is reviewed.



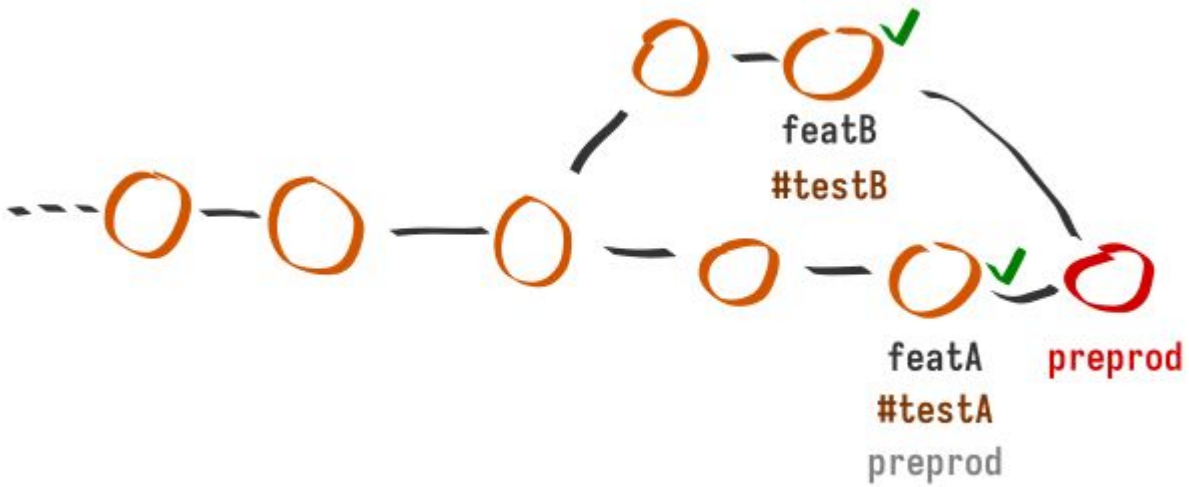
Scenario #2: proposal

- 2 developers are working on 2 different tasks.
- **featA** is reviewed.
- **featA** is deployed to be QA tested.
- **featA** is merged to **preprod**.
- **featB** is reviewed.
- **featB** is deployed to be QA tested.
- At this point the **featB** is ready to be deployed on preprod (aka staging).



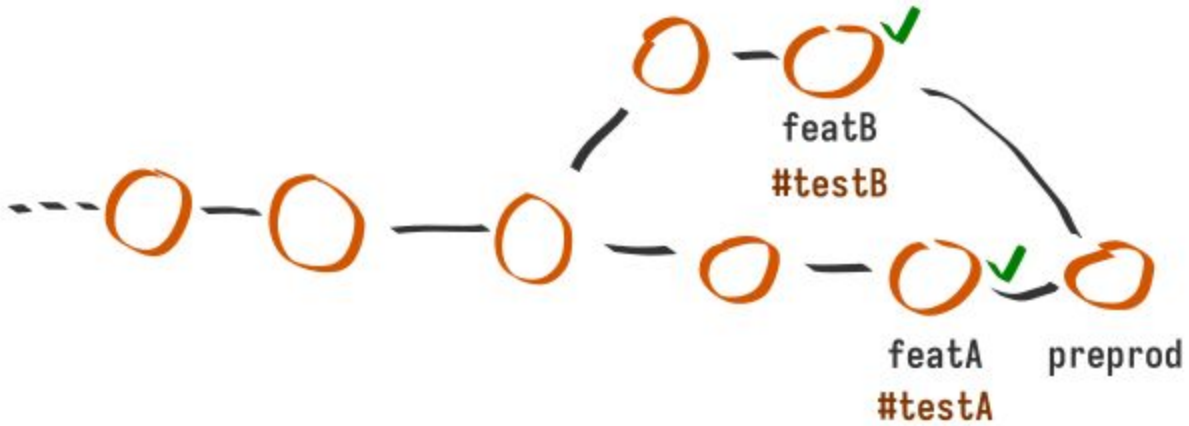
Scenario #2: proposal

- 2 developers are working on 2 different tasks.
- **featA** is reviewed.
- **featA** is deployed to be QA tested.
- **featA** is merged to **preprod**.
- **featB** is reviewed.
- **featB** is deployed to be QA tested.
- **featB** is merged to **preprod**.



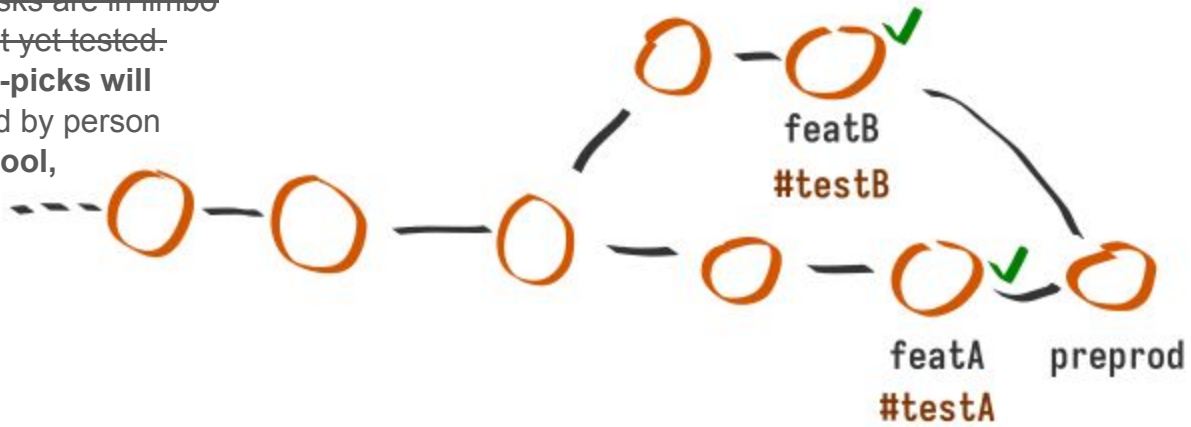
Scenario #2: proposal

- 2 developers are working on 2 different tasks.
- **featA** is reviewed.
- **featA** is deployed to be QA tested.
- **featA** is merged to **preprod**.
- **featB** is reviewed.
- **featB** is deployed to be QA tested.
- **featB** is merged to **preprod**.
- …



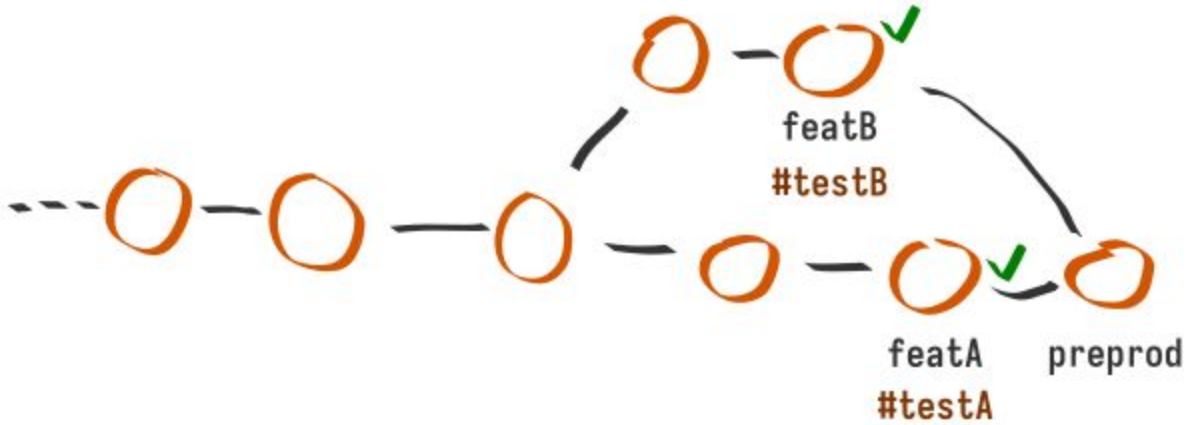
Scenario #2: proposal - risks

- Every **decision to pick** only some selected **features** from the pool of all available features **leads to cherry-picking** of either regular commits or merge commits.
- **Cherry-picking creates a potential of human error** (missing parts of a feature, dependencies, etc.)
- **A pull request** crafted by cherry-picking features **has to be reviewed** because of the above.
- **The source branch** (like branch “develop” in example) **is never a continuation of the target branch** (like “test”), it is a fork. Therefore the **cherry-picking is necessary**.
- **This** process **is repeated on ONE stage** of the workflow: staging > beta.
- ~~**Testing on reviewed features is on hold until the test-candidate is merged**, thus tasks are in limbo state where they are completed, but not yet tested.~~
- There is a possibility that **some cherry-picks will cause conflicts** that has to be resolved by person with a **knowledge not only in the git tool, but also in a domain, language, frameworks, libraries, etc.**



Scenario #2: proposal - benefits

- Pull requests are merged only when fully reviewed and tested.
- The preprod branch is a place to start development from.
- No need of cherry picking anything to deliver work to QA nor PO.
- By utilising tags QA can trace their work by referring to a certain points in preprod history.
- Less PRs to review.
- Less points where human error can occur.
- Less engagement needed of the competent developer to formulate push-candidates.



Q&A