# Git Presentation
# TR Unconference 2022

Dan Jakob Ofer
@ofer987
https://github.com/ofer987
dan.ofer@thomsonreuters.com

First Slide!

# Yorkville

## Pilot Coffee

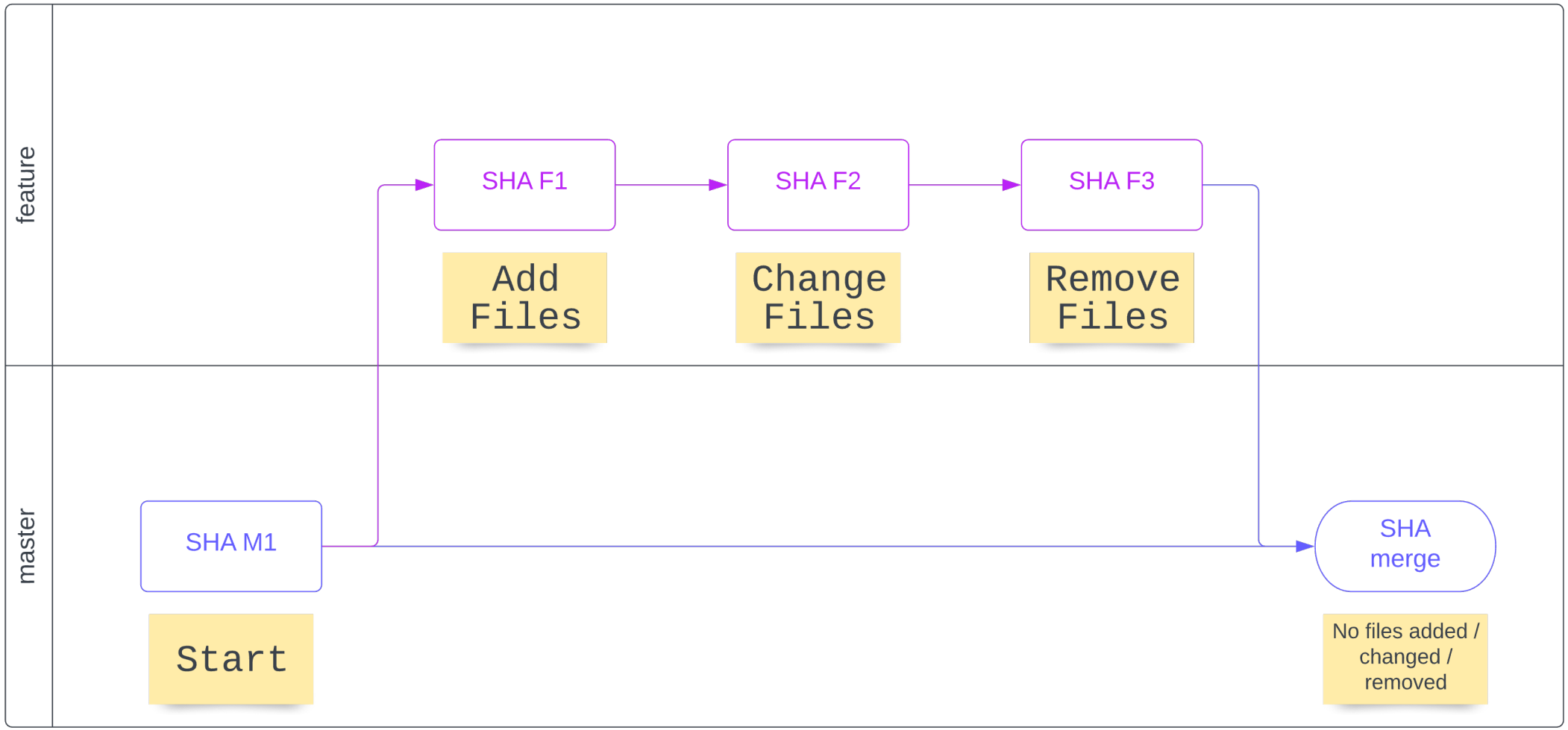
## Jacked Up Coffee


## Eataly

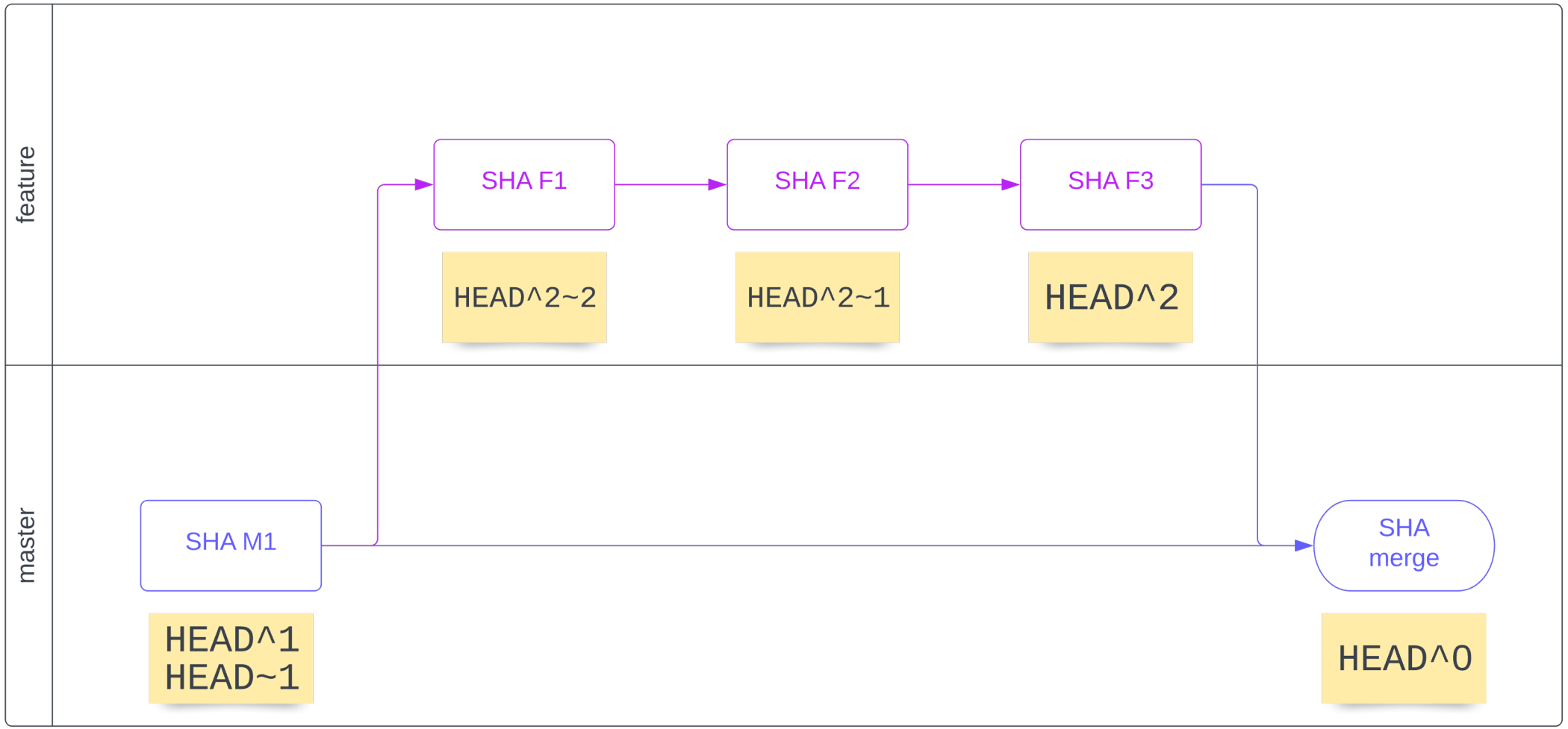
# Presentation

- Understanding git source control management
- Case study for facilitating deploying TR’s websites

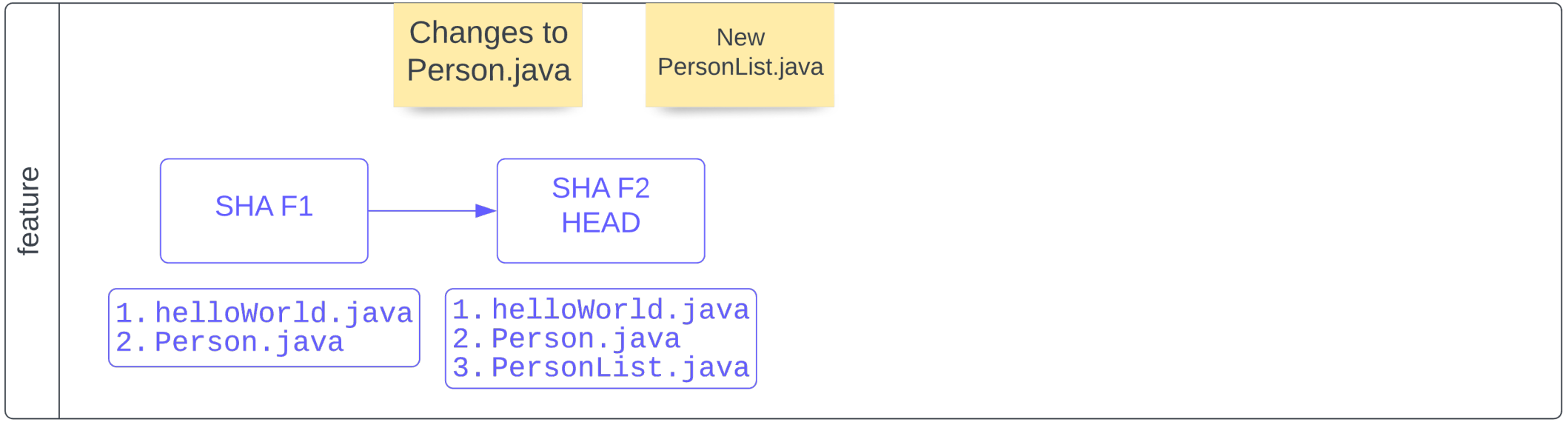
# Overview


feature
master
SHA F1
SHA F2
SHA F3
Add Files
Change Files
Remove Files
SHA M1
SHA merge
Start
No files added / changed / removed

# Commit references (absolute and relative)

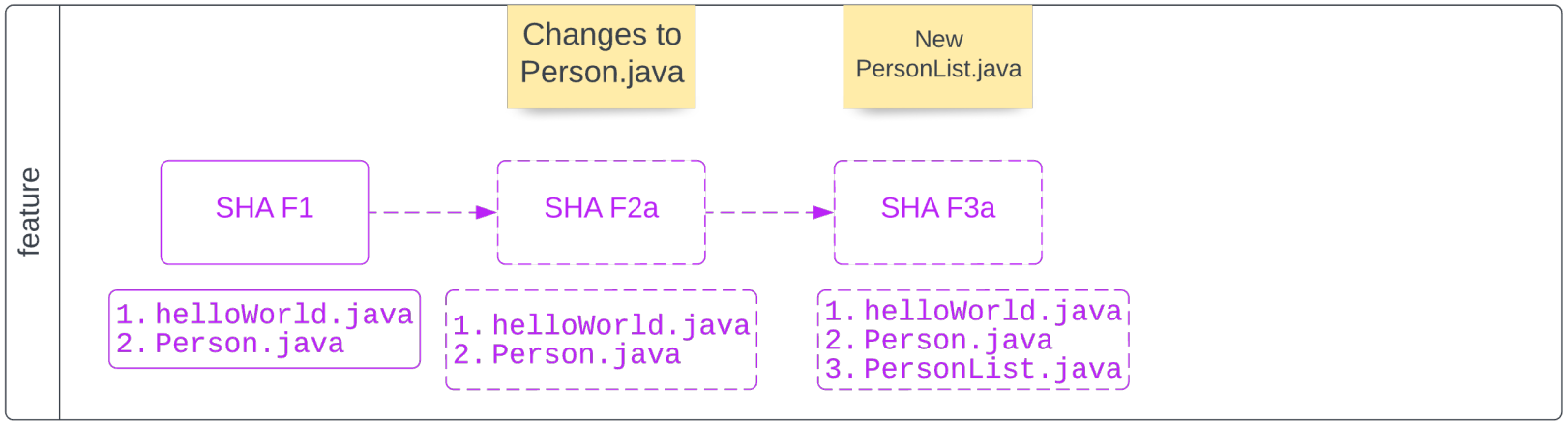


# Fix commit history (part 1)

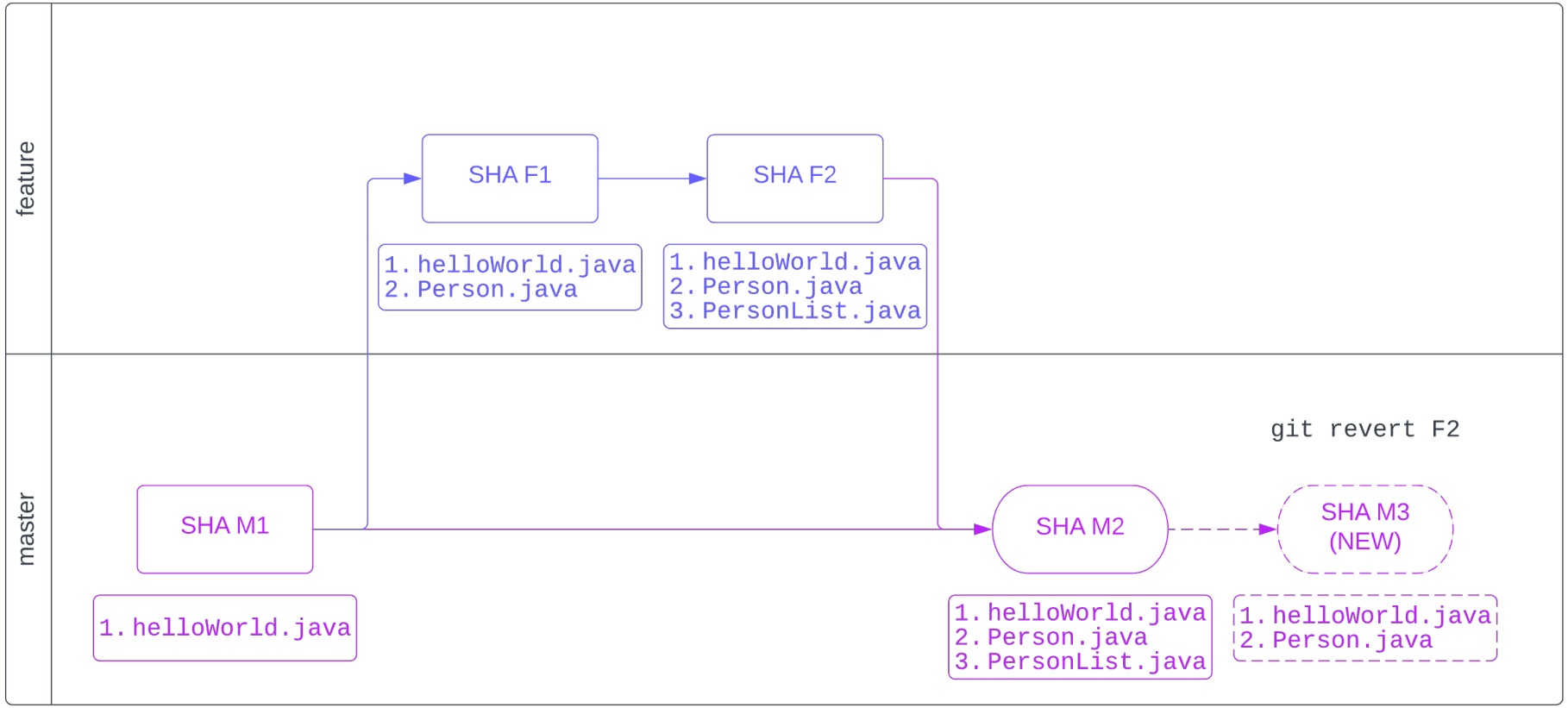


# Fix commit history (part 2)


1. Change the HEAD to F1, `git reset F1`
2. Add changes to Person.java, `git add Person.java && git commit -m "changes to Person.java"`
3. Add new PersonList.java, `git add PersonList.java && git commit -m "add PersonList.java"`



# Revert a commit

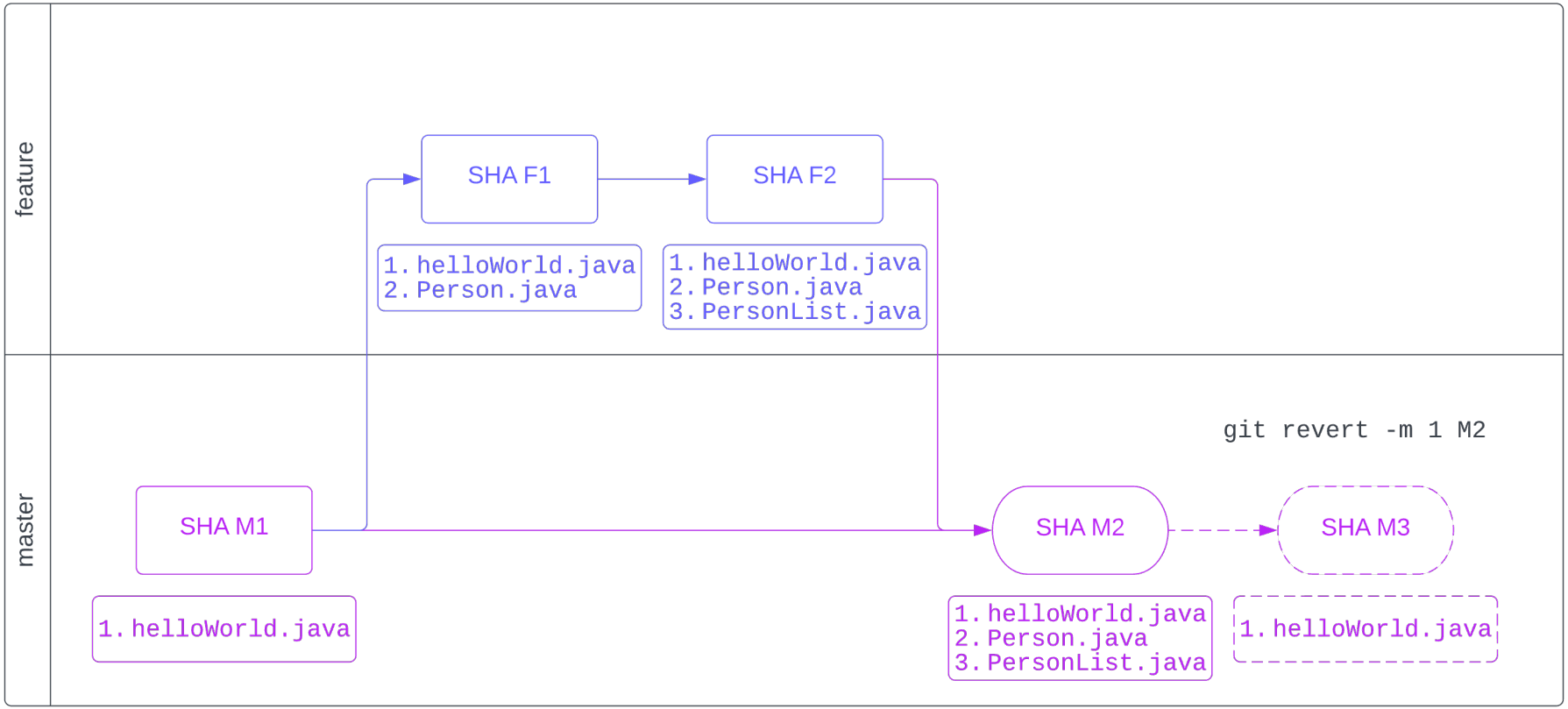



## Instructions

To revert the the F2 commit out of the `master` branch,

```
1. git revert F2
```

# Revert a merge commit (e.g., a Pull Request)

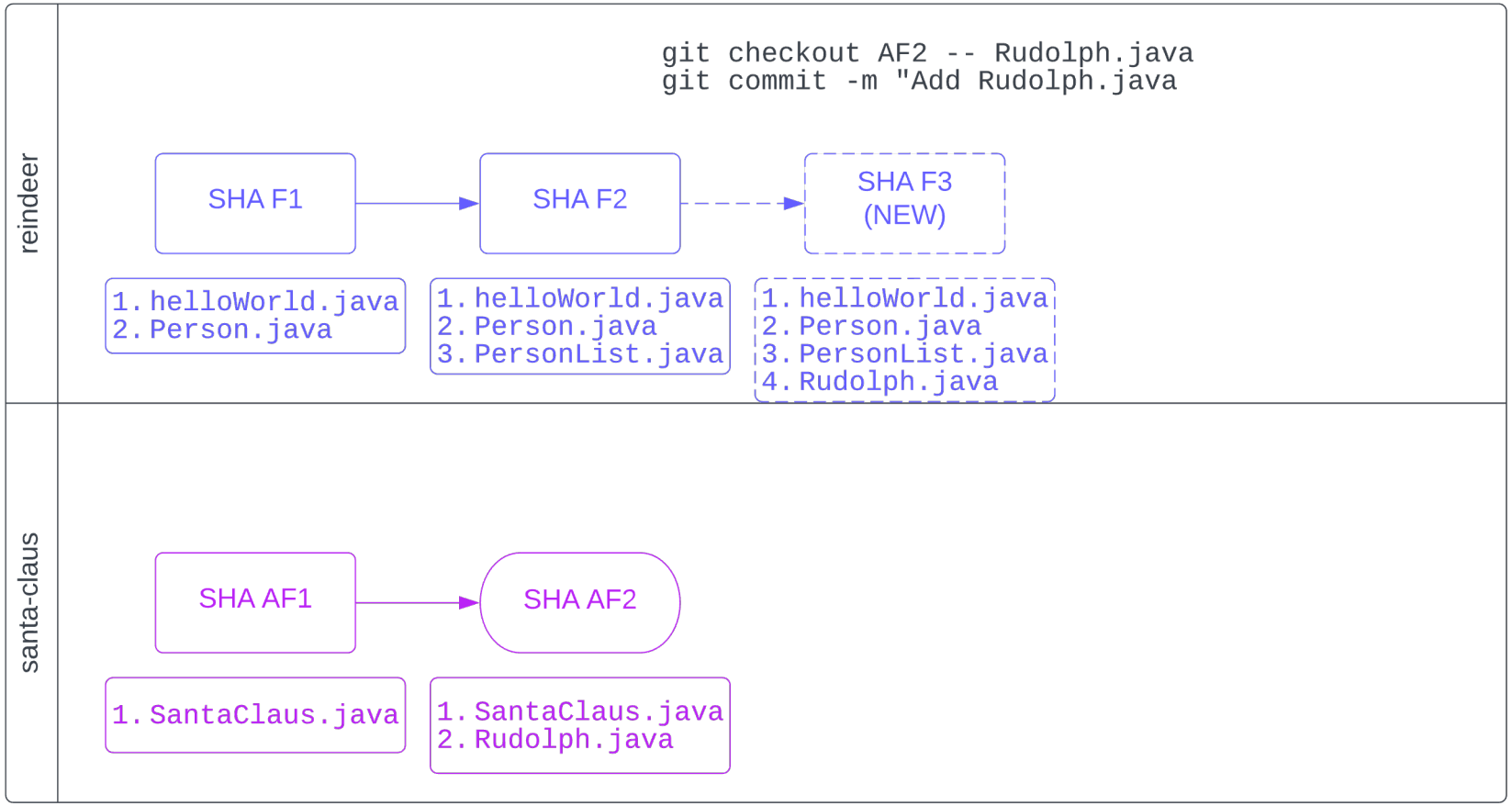



## Instructions

To revert the `feature` branch out of the `master` branch,

1. `git revert -m 1 M2`

# Checkout files from commit

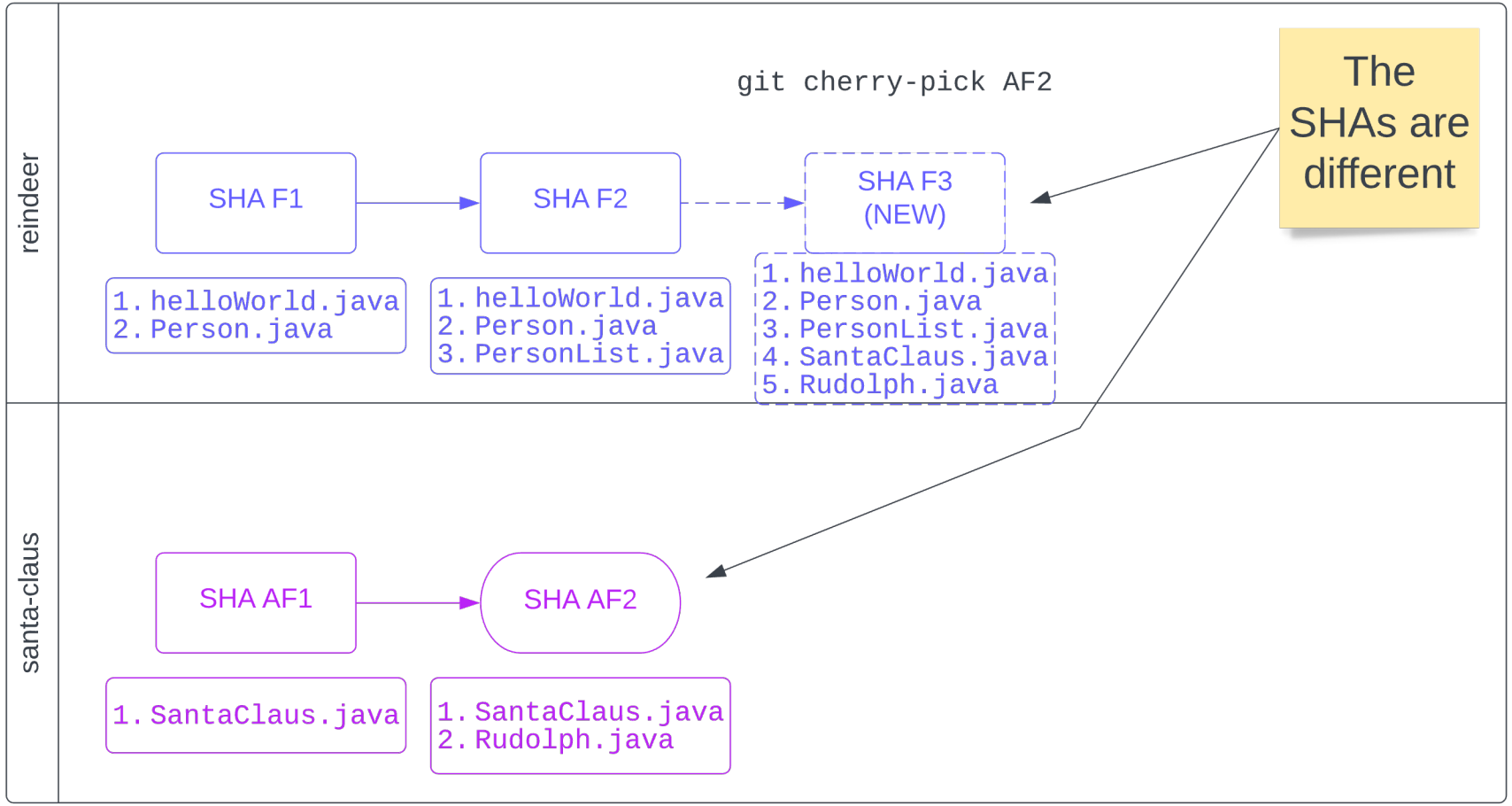



## Instructions

To add When working on the `feature` branch, add Rudolph.java

```
1. git checkout AF2 -- Rudolph.java
2. git commit -m "Add Rudolph.java"
```

# Cherry-pick commit from another branch




## Instructions

To add cherry-pick AF2, which will add both SantaClaus.java and Rudolph.java

```
1. git cherry-pick AF2
```

# Mission: Improve release process!

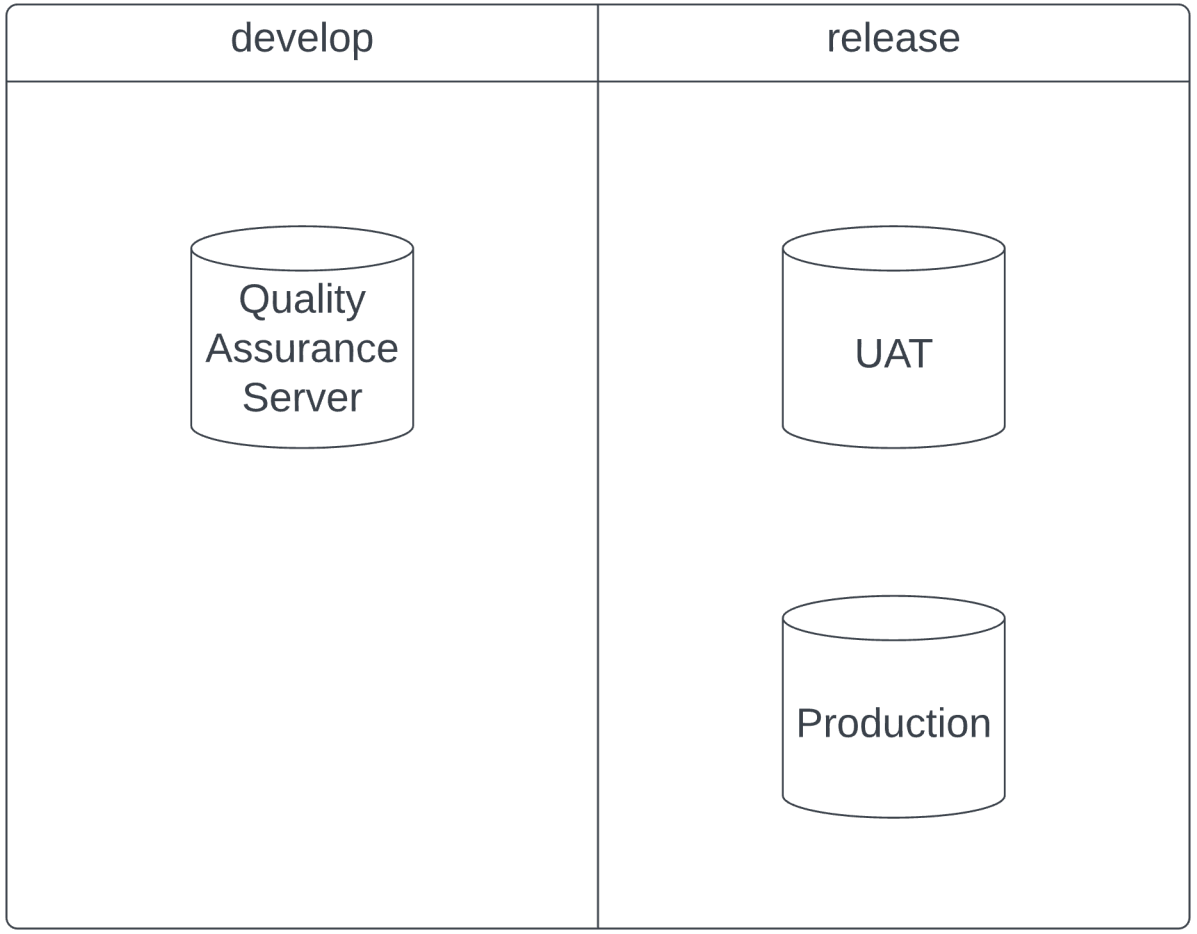
## Who am I?

- I work on the Digital Team
- Responsible for improving our release processes

## Problems we are facing

- Several server environments
- Hundreds of developers
- Two-week Sprints
- Four weeks between **End-of-sprint** to **Production Release**

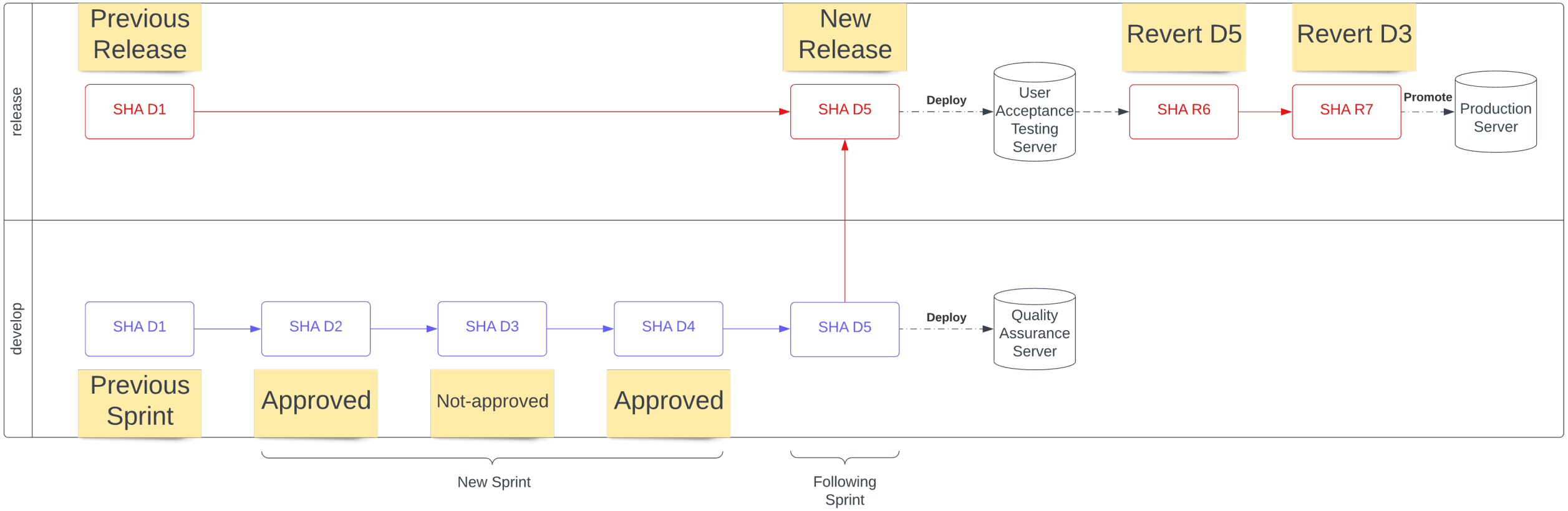
# Server Environments



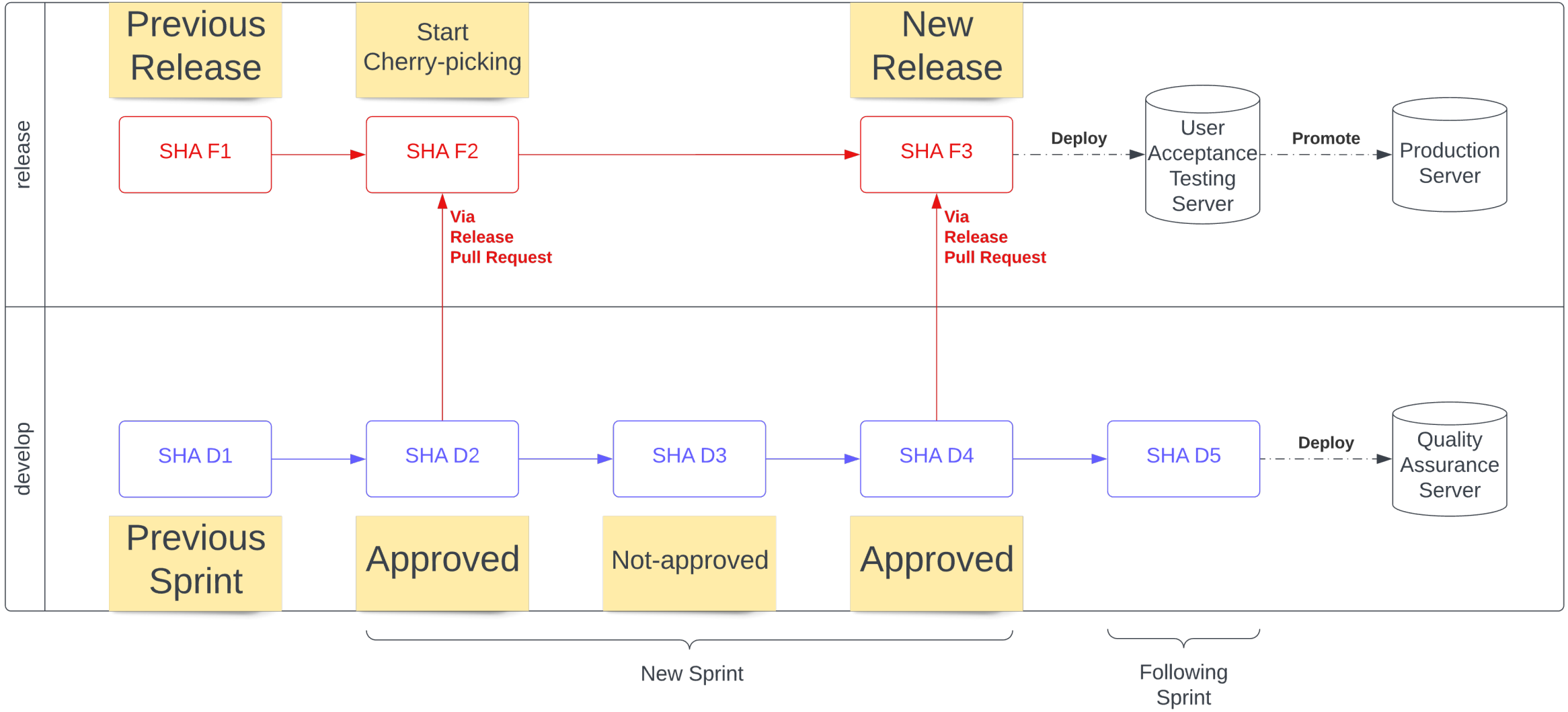
# Development and Release History

| Year | Developers | Location | Release Process | Notes |
|---|---|---|---|---|
| 2018 | 3 | All in London, UK | **Legacy Process** | Manual Deployments ☹ |
| 2019 | 10 | All in Toronto, Canada | **Release Process 1.0** | Automated Deployments ☺<br>*But it is confusing!* |
| 2020 | > 10, but < 100 | Worldwide, e.g., Toronto, United States, Eastern Europe, India | | |
| 2021 | > 100 | Worldwide, e.g., Toronto, United States, Eastern Europe, India | **Release Process 2.0** | *Still confusing.* |
| 2022 (>= Sept) | > 200 | Worldwide, e.g., Toronto, United States, Eastern Europe, India | **Release Process 3.0** | 1. Simpler<br>2. Facilitates deployment<br>3. Easier to communicate |

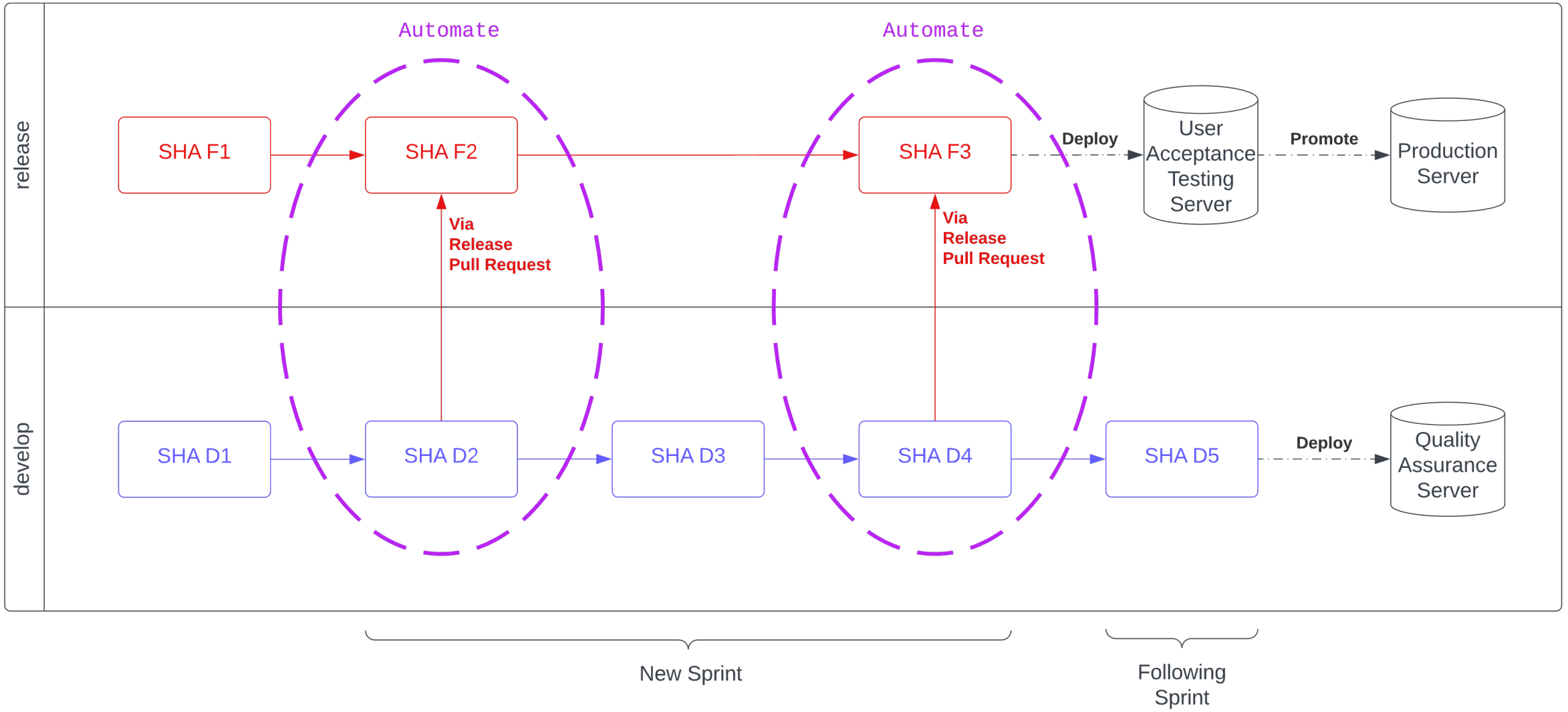
# Release Process 2.0



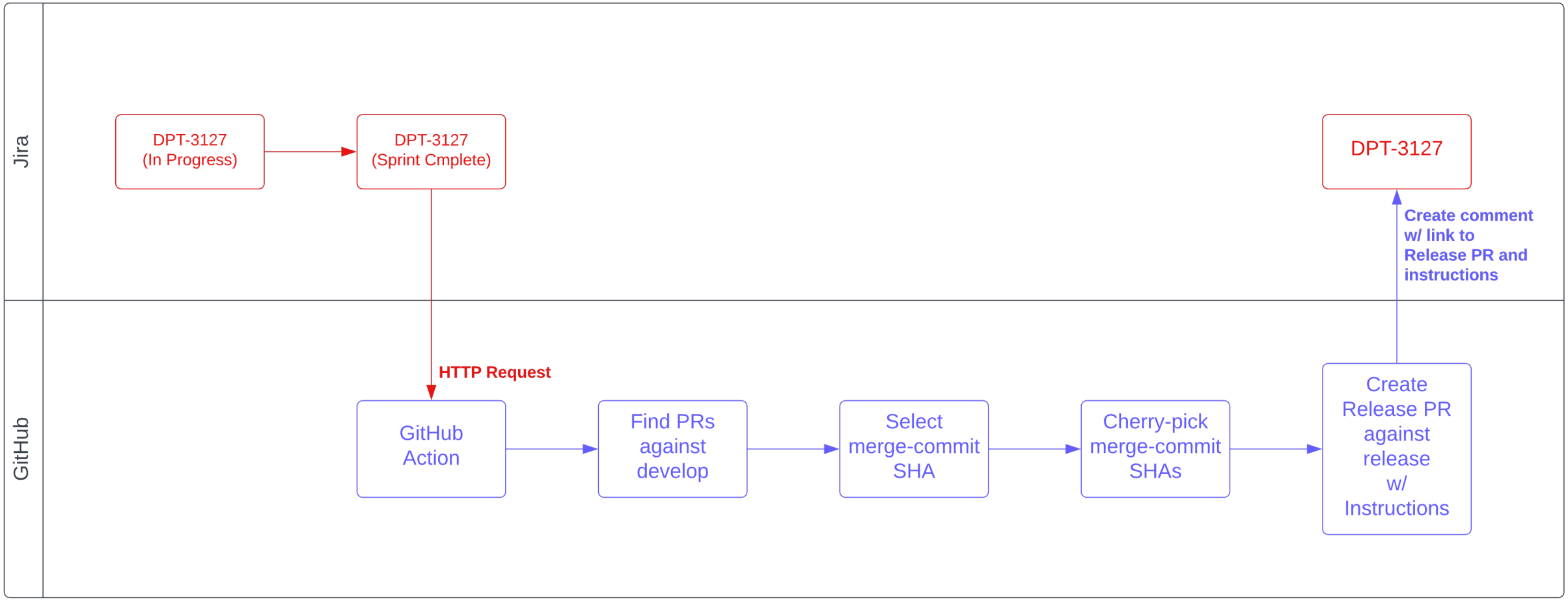
# Release Process 3.0



# Automate the process!



# Flow between Jira and GitHub



# Current release process

- Deploy everything in the develop branch
- Revert unapproved Jira Issues
- Sometimes we deploy work-in-progress code our code from next Sprint, and we find out in Production

# What is a Release Pull Request?

- A Release Pull Request is an Approved Jira Issue’s commits amalgamated into a new Pull Request against Production’s branch, release.
- Release Pull Requests ensure that only approved code is delivered to the Production server.

# Moving code from QA to Production

1. Jira Issue’s state is moved from Validated to Sprint Complete
2. Jira Server sends HTTP Request to GitHub Action
3. GitHub Action intelligently finds associated Pull Requests
4. Creates Release Pull Request to deliver to Production
5. All actions, including commands, printed to both the Jira Issue and Release Pull Request
6. In case of merge conflicts, the developer can repair the Pull Request, using the printed commands!

# Automate a Cherry-picking Pull Request

1. Create local Cherry-pick branch, `releases/<jira-key>`
2. Checkout Cherry-pick branch, `releases/<jira-key>`
3. Add empty commit (to ensure that Pull Request can be created)
4. Cherry-pick merge commit SHAs of associated Pull Requests
5. Push branch to GitHub
6. Open Pull Request in GitHub
7. All actions (including git CLI commands) printed in,
    1. Pull Request's body
    2. Jira Issue comment