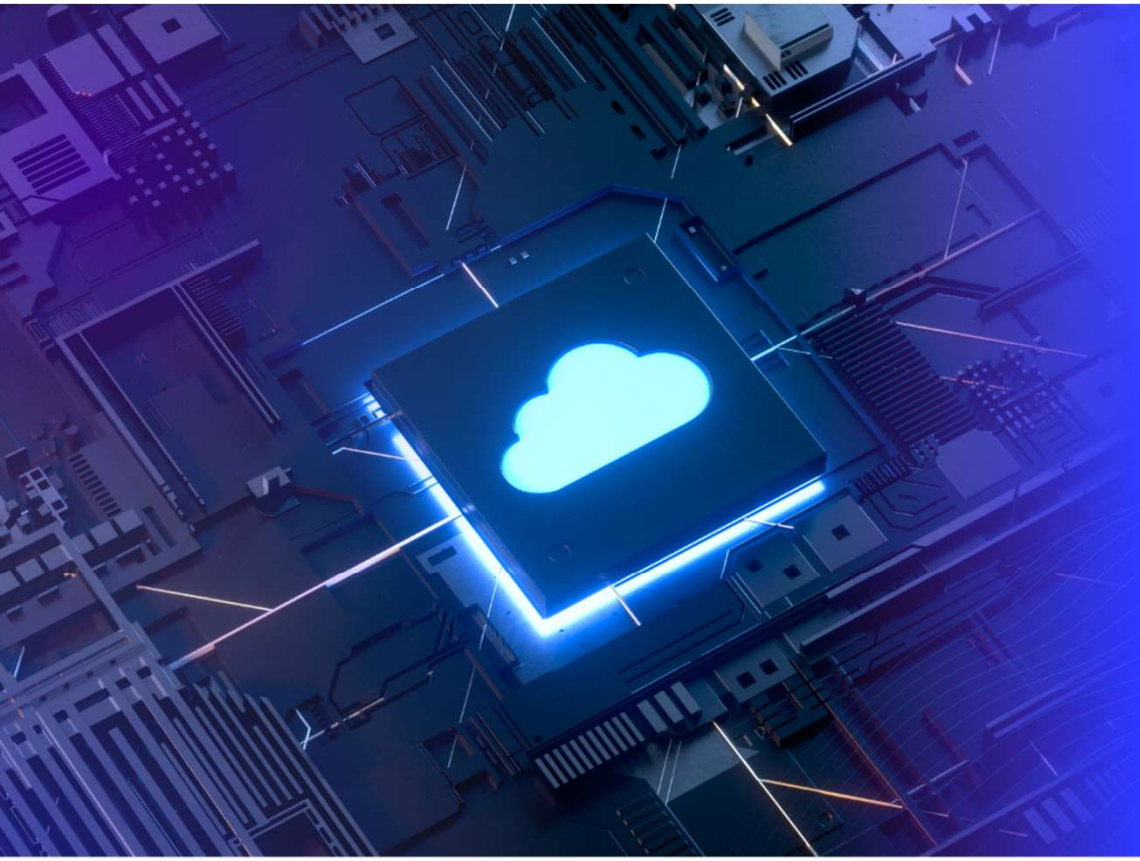
台灣積體電路製造股份有限公司
Taiwan Semiconductor Manufacturing Company, Ltd.



TSMC IT X NCTU CS 課號 5270

CLOUD NATIVE
Development Best Practice

DEVELOPMENT

TSID
FOPD | Julian Shen, Kevin Yang
April , 2022

About Us

Julian Shen (沈冠廷)

- Graduated at NCTU CIS
- Work Experience
  - HTC
  - ASUS
  - LINE
  - TSMC
- Contact
  - gtshen@tsmc.com

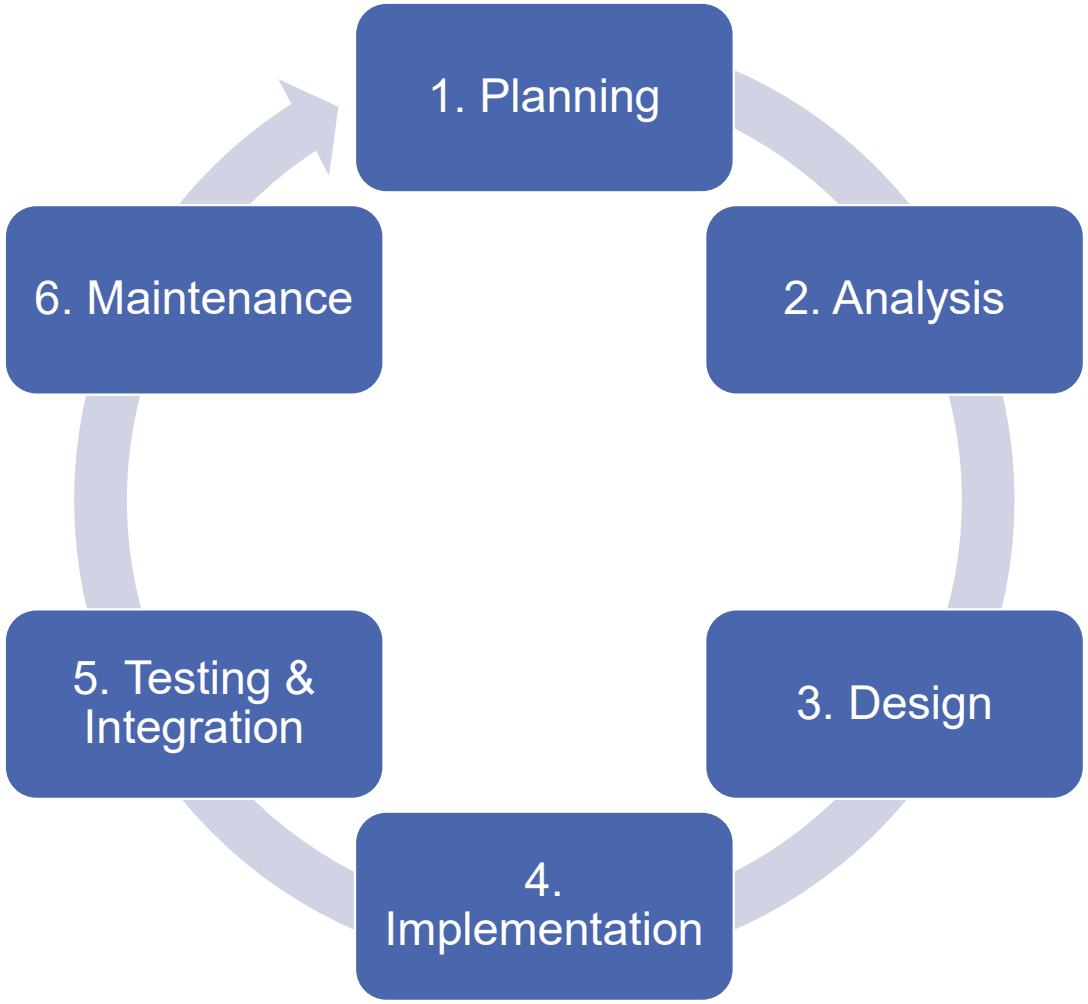
Kevin Yang (楊捷凱)

- 社群小小
  - 自己有間小小的公司
  - 每年接一兩個開發顧問案打發時間
  - TSMC
- Contact
  - chgc.tw@gmail.com

AGENDA

- **What you will learn today?**
  - **How a software product is developed**
  - **How to manage your artifacts**
  - **Deliver quality software**

Software development process



Software Development Life Cycle (SDLC)

Planning

Outline requirements:

A dashboard with user registration and login

Analysis

Define requirement details:

1. User fields in registration and save to database …
2. A dashboard user should be redirected to after login

Design

Design for the software:

User interface, Architecture, tech stacks, specs, and more

Implementation

Start coding

Tester should also design tests

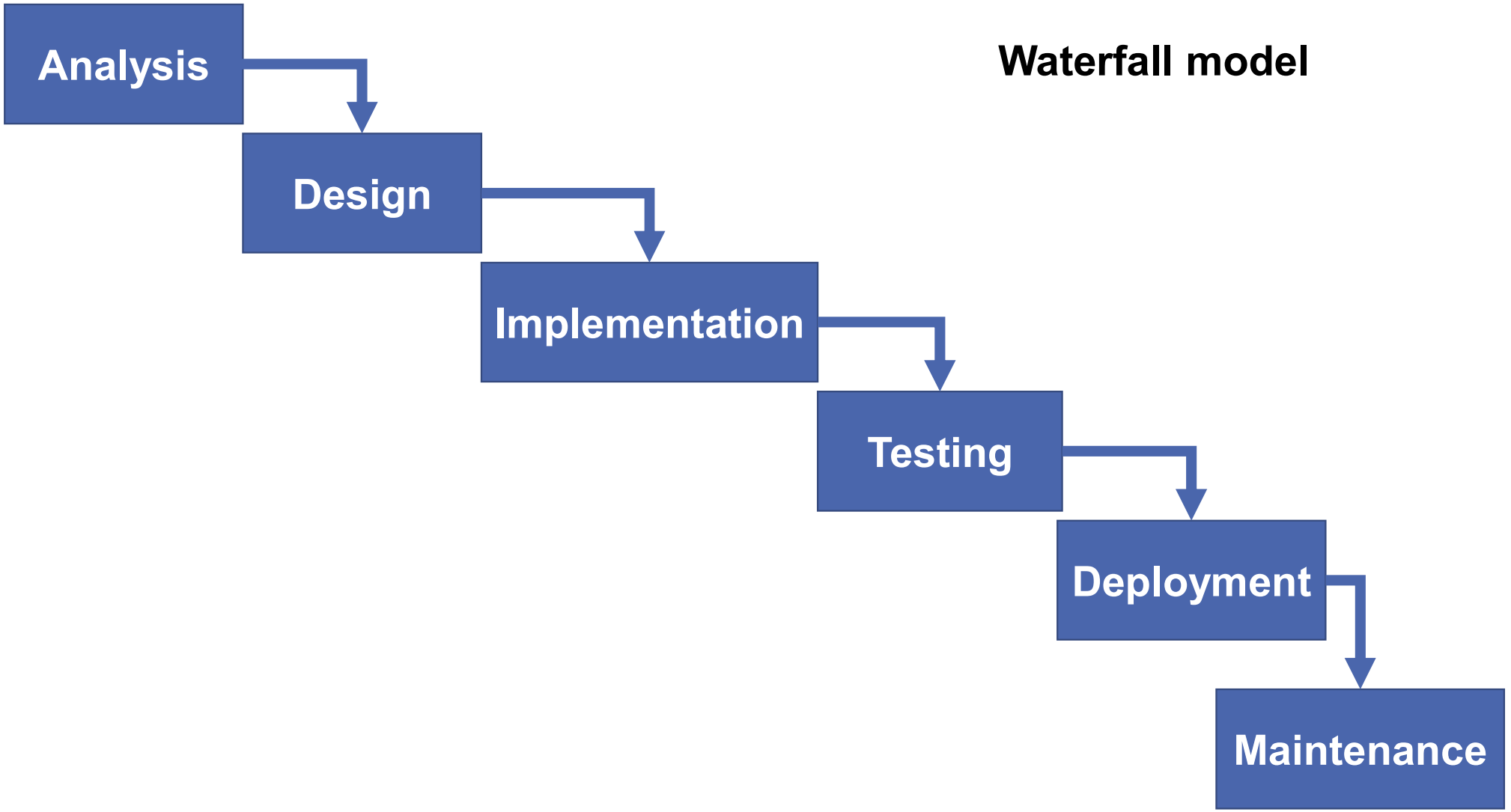
Testing & Integration

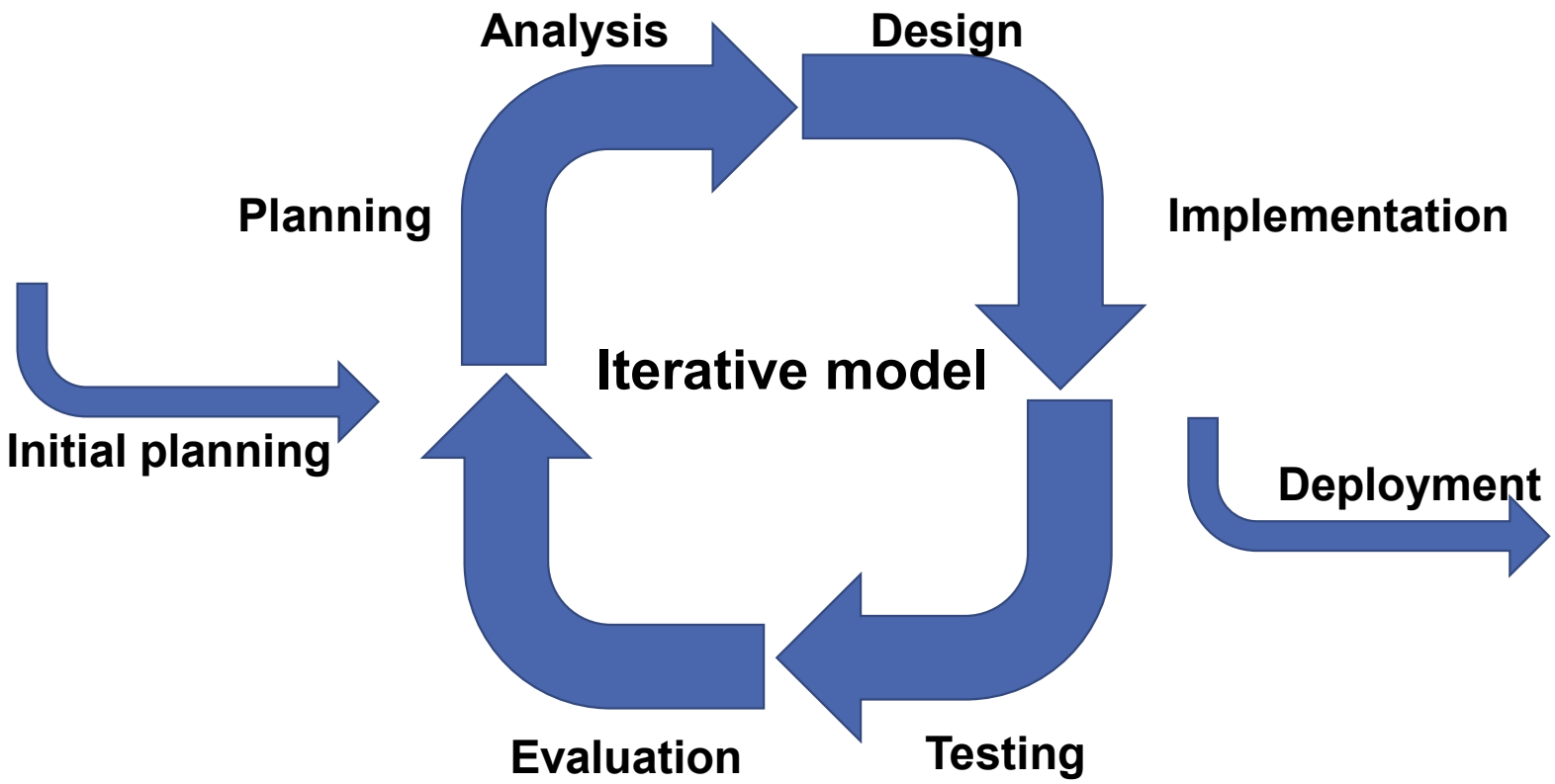
Test the results produced by implementation

Deliver software

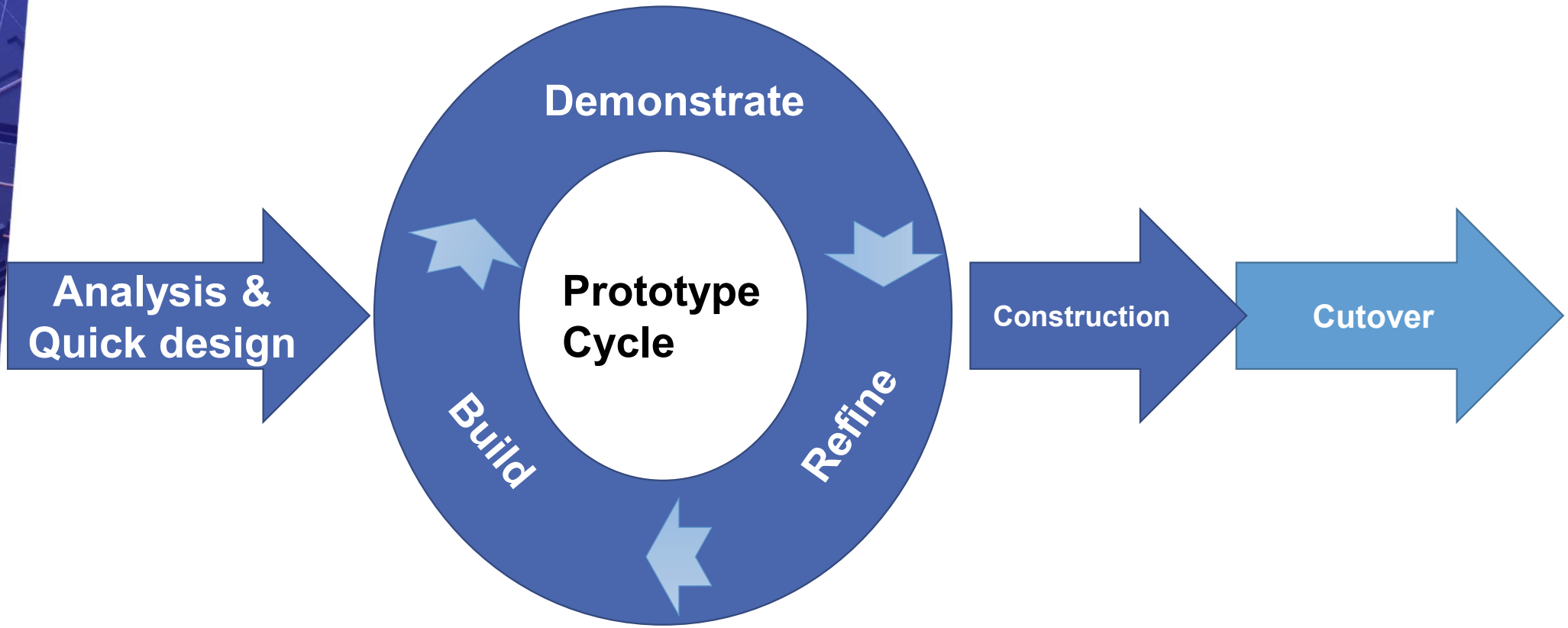
Automated acceptance tests, capacity tests, manual tests

Waterfall model



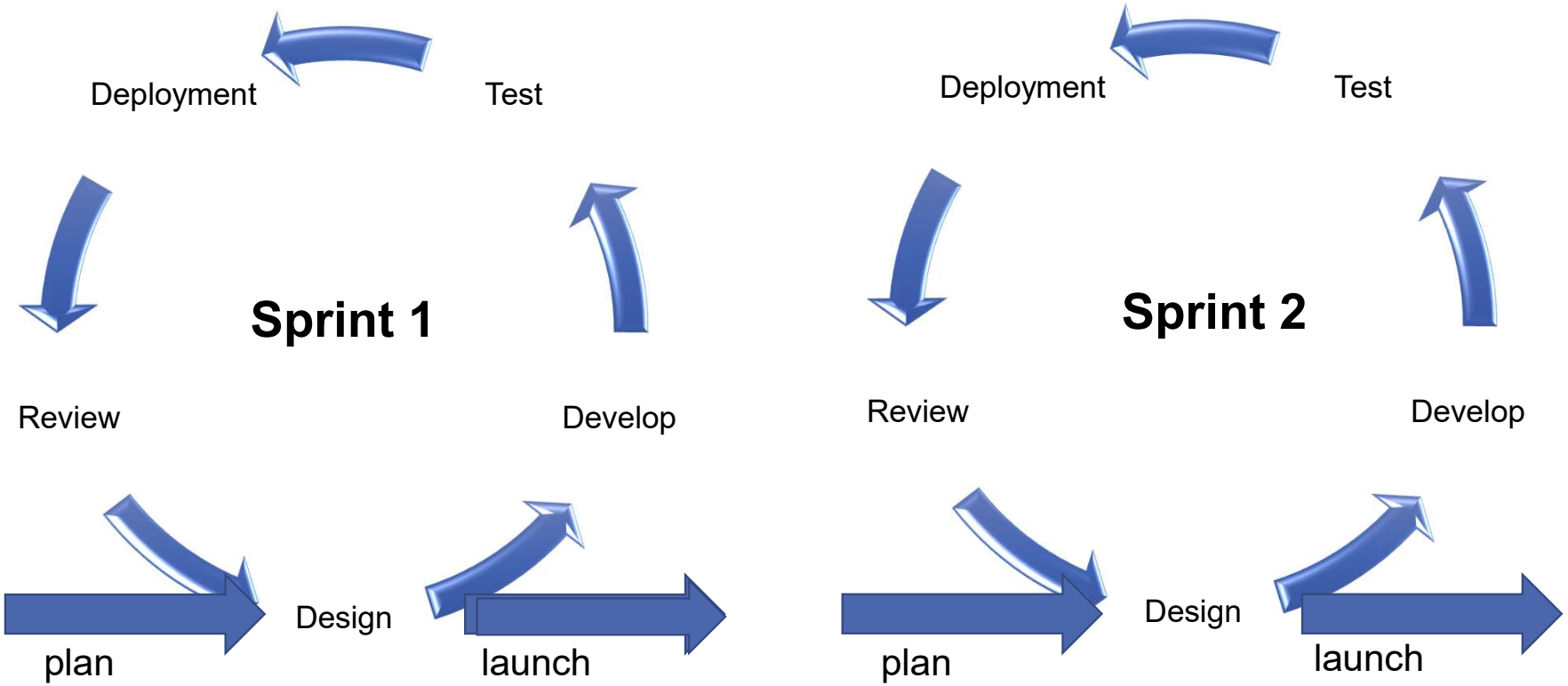


RAD



Agile development

Kanban, Scrum, Lean software



Which is better?

- **Waterfall**
  - Suitable for a software project with clear objective and well defined requirements
  - Lacks of flexibilities
- **Iterative**
  - Major requirements are clear but functionalities may evolve
  - Progress easy to measure
  - Difficult to pin down an end date
- **RAD**
  - For projects want to be done fast
  - Require highly skilled members
  - Unstructured
- **Agile**
  - Building software incrementally, final product is not released first
  - Go to market fast
  - Hard to predict

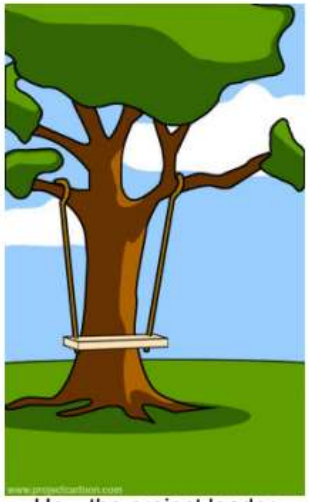
Requirements and user stories

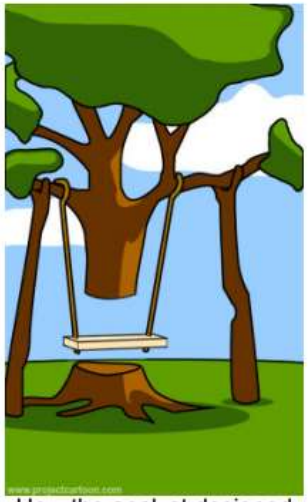
How hard to understand user requirements?

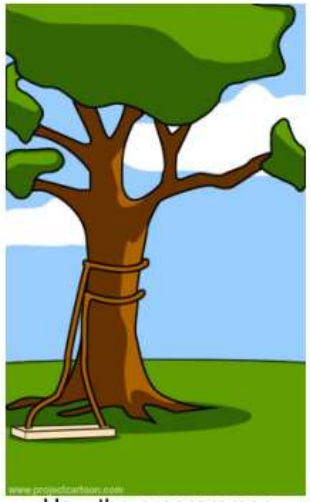
How Projects Really Work (version 1.0)

Create your own cartoon at www.projectcartoon.com

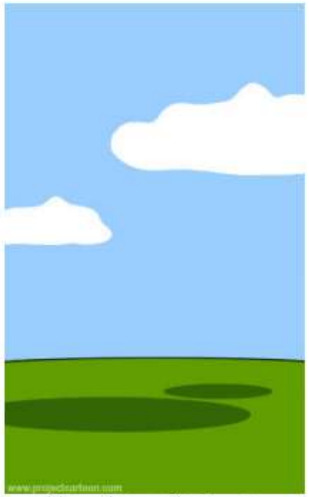

How the customer explained it

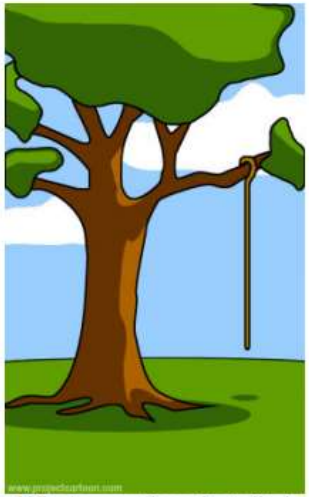

How the project leader understood it

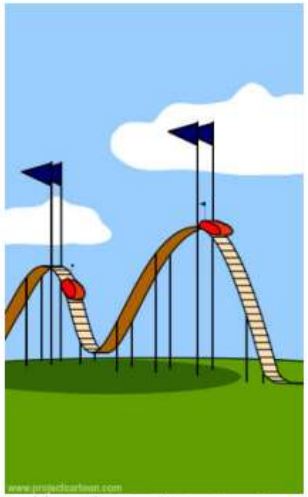

How the analyst designed it

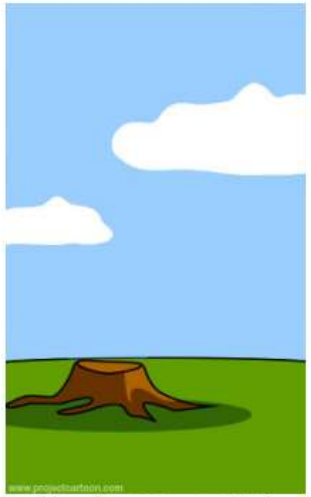

How the programmer wrote it


How the business consultant described it


How the project was documented


What operations installed


How the customer was billed

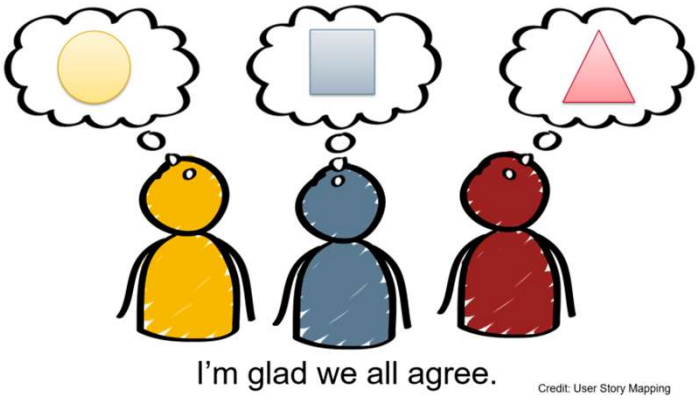

How it was supported

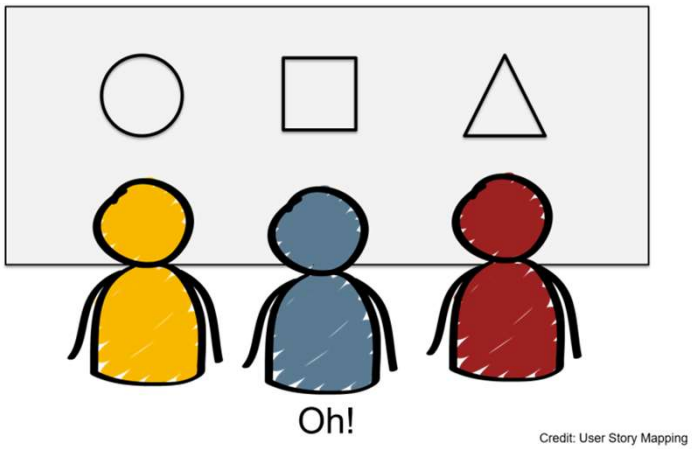

What the customer really needed

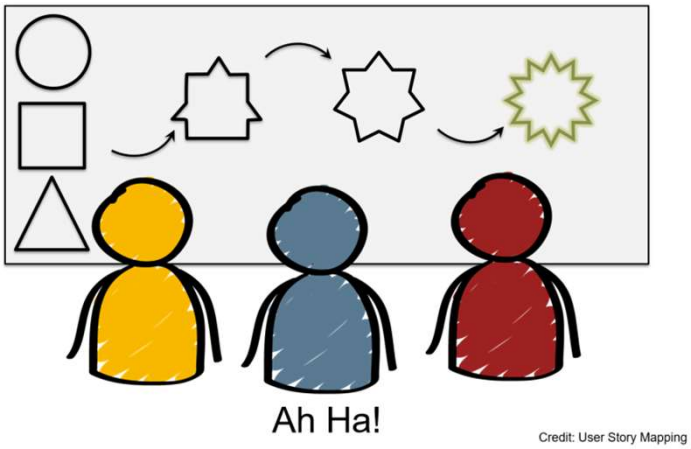
Let’s play a game

Drawing game

It’s about conversation







What is a user story

- A user story is an informal, nature language description of features of a software system
- A user story is not equal to a requirement but helps to better understand the requirement

User story template

As a <persona> ← Who

I want <do something> ← What (Intent)

So that <get benefit> ← Why

As a user I want to book a room

As a business traveler I want to find a room with wifi and desk

As a budgeted traveler I want to find cheapest and discounted room so that I can save my $$

As a host I want to a listing so that I can making $$

AirBnB

A good user story should be

- “I” ndependent (of all others)
- “N” egotiable (not a specific contract for features)
- “V” aluable (to the end user)
- “E” stimable (to a good approximation)
- “S” mall (so as to fit within an iteration)
- “T” estable (can be verified with acceptance criteria)

Acceptance Criteria

- A set of predefined requirements that must be satisfied to make a user story complete
- It is used for
  - To define boundaries
  - To reach consensus
  - To serve as a basis for tests
  - For accurate planning and estimation

Given <pre-condition>
When <condition>
Than <outcome>

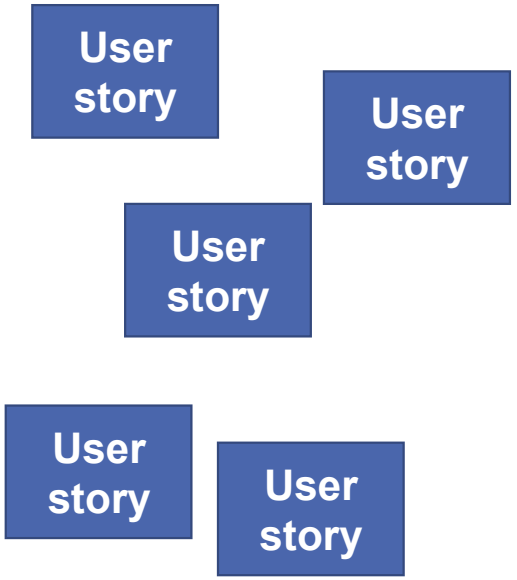
User story: *As a user, I want to be able to recover my password, so that I could access my account in case I forget my password*

GIVEN	*User navigate to login page*
When	*User click “forget password” and enter his email*
THEN	*Send a mail with recovery link to user*
GIVEN	*User receive the email*
When	*User navigate to the link attached in email*
THEN	*Enable user to set a new password*

Practice (10min)

“Discover” not just analyze

Shopping cart/Uber



Deploy

Test

Implement

Design the architecture

Find out solutions

?

Software

- **Architecture is important**
  - Understand your requirements
  - Thinking about components
- **Something you should care**
  - Performance
  - Scalability
  - Portability
  - Extensibility
  - Compliance
- **Choose your tech stack well**
- **Try not to over engineered**

“Applications are social constructions”

- Martin Fowler

Source control

Why version control

- Version control, aka source control, is the practice of tracking and managing changes to software codes
- Software is not built by yourself most of the time
- Source codes act as a single source of truth of a product's knowledge
- Version control enable teams to collaborate development and create a centralized location for code

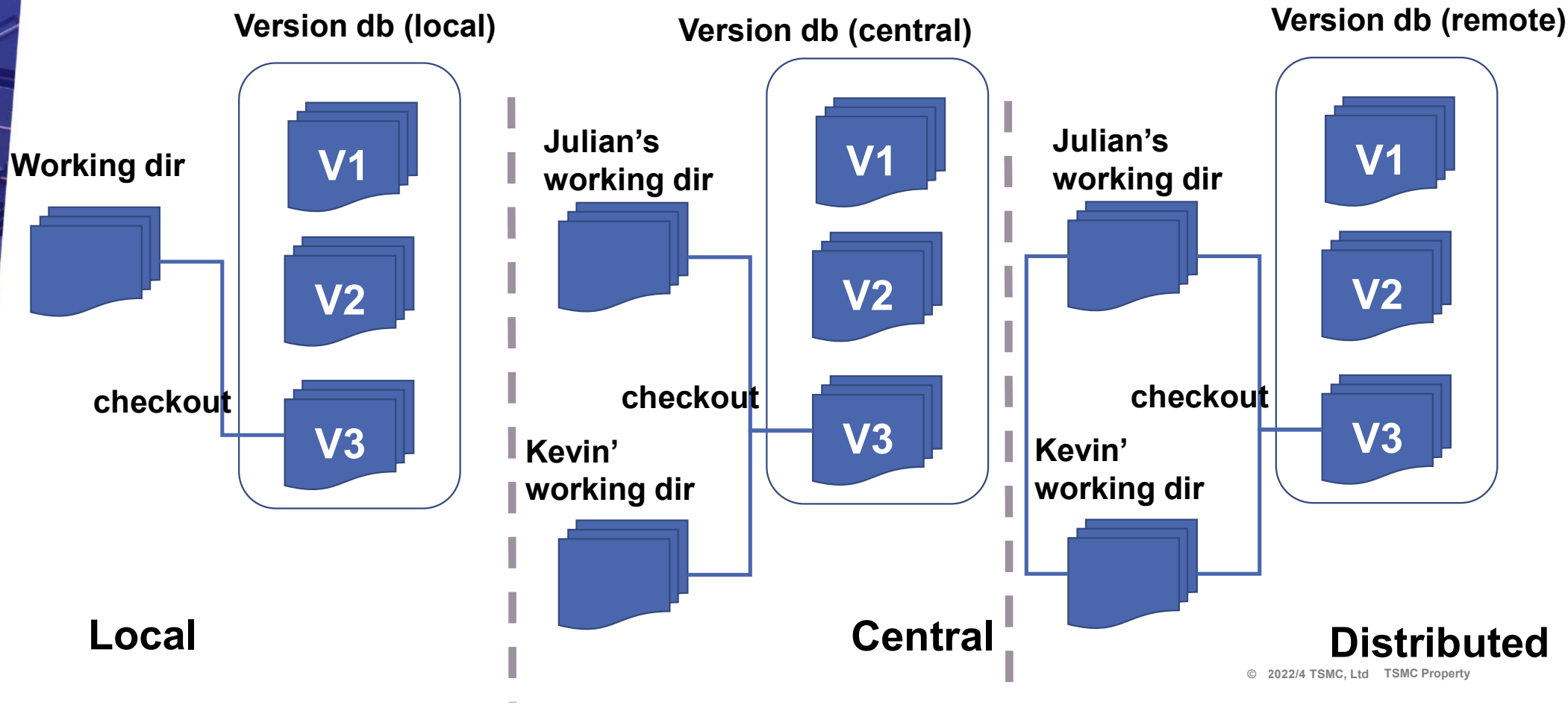
Benefits of version control

- **Quality**
  - Review, comment, and improve each other's codes and assets
- **Acceleration**
  - Branch codes, make changes, and merge faster
- **Visibility**
  - Understand the results of team collaboration

Terminology

- Repository/Repo – A directory contains your project works
- Commit – Save the state of your project
- Working Directory – Project files you see on your local computer
- Checkout – Copy the content in repository to working directory
- Branch – The alternative line of development can be continued without altering the main line
- Revision – Contents of a repository at a single moment in time
- Pull – To receive data from repository
- Pull request(PR) – A process of a developer to notify team members that they have complete a feature
- Push – Upload committed state to a remote repository

Type of VCS



Typical solutions

- Git
- Subversion (SVC)
- Mercurial
- CVS
- SourceSafe
- ClearCase
- Perforce

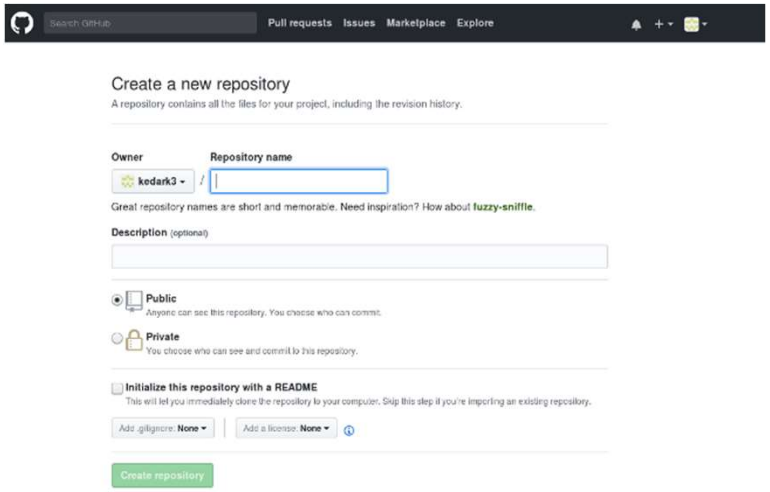
Git is the most popular

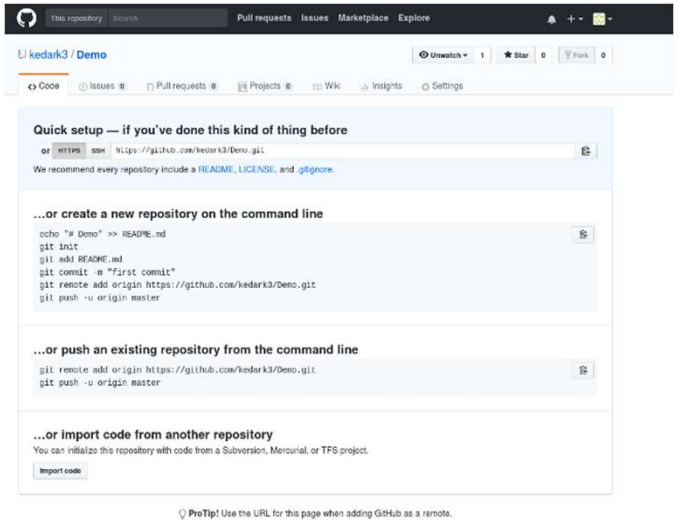
Using git

- Go to your working directory
- Create repository with “git init”
- Add files to your working directory
- Add these files to git with “git add .”
- Commit files with “git commit –am “your comment””
- Add remote repository with “git remote add remote_name remote_url”
- Push commits to remote repository with “git push remote_name branch name”

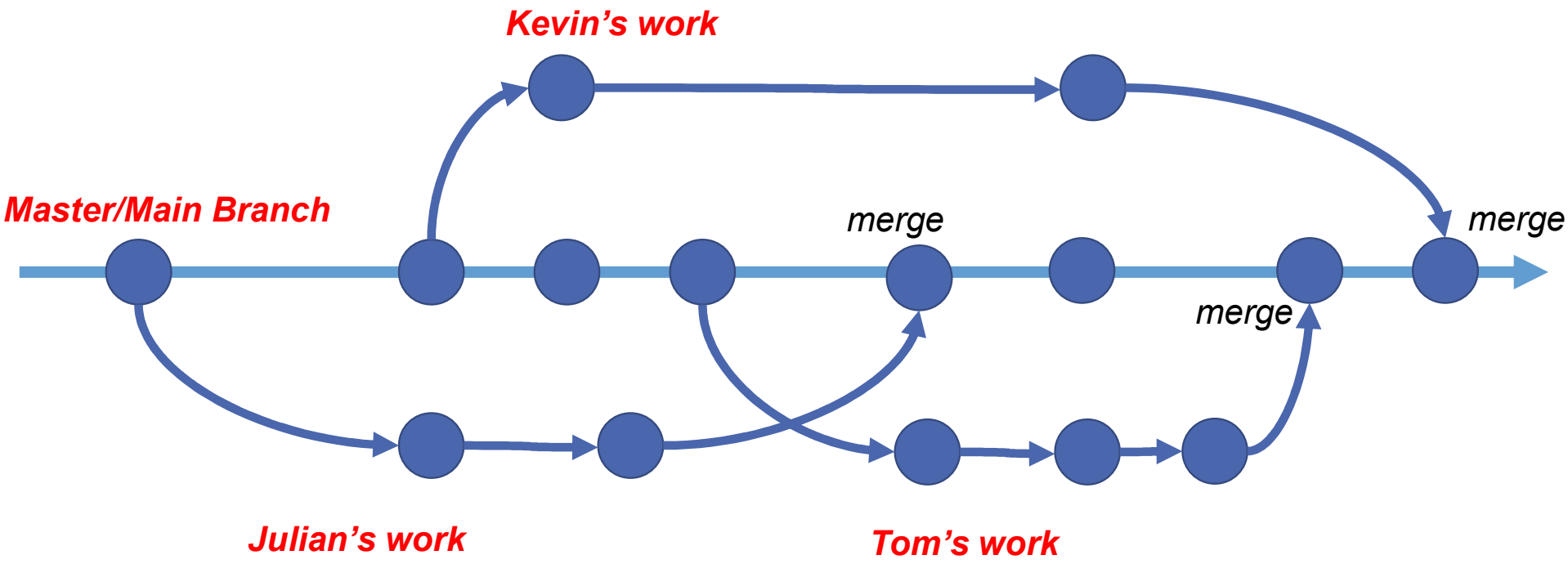
Practice

- Create a github (https://github.com/) account if you don't have it
- Create a new repository
- You will have a remote repository url after created
- Try to commit and push your codes





Branches



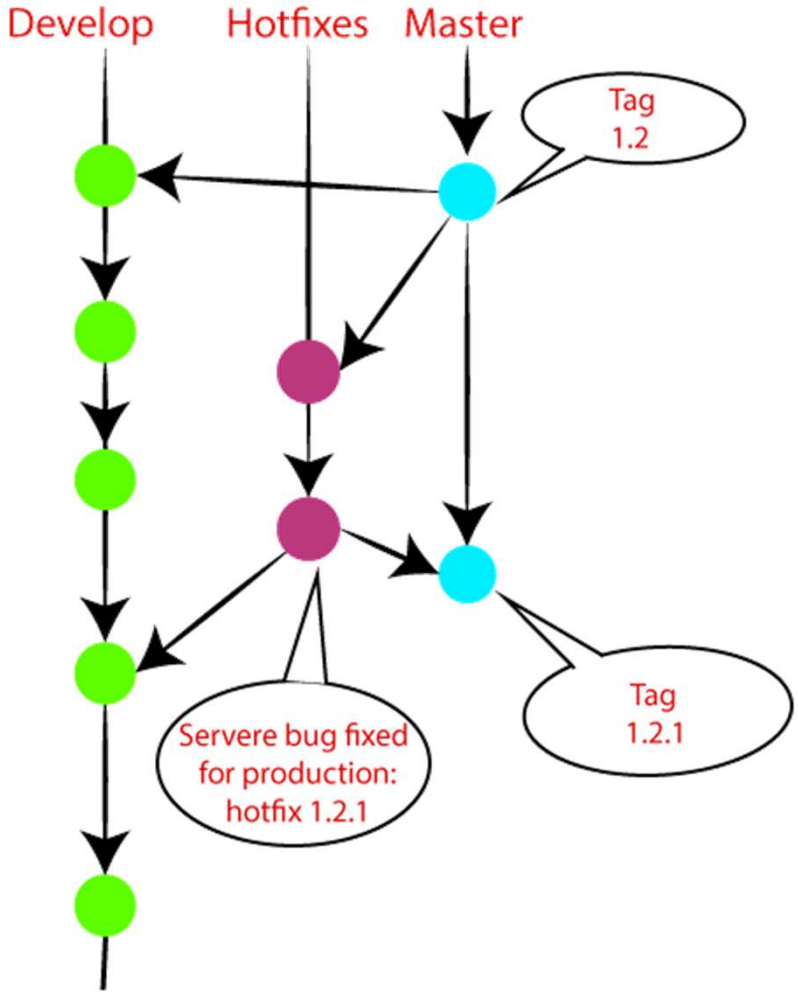
Why Branches

- Sharing one single branch among all team members leads to conflicts every time
- Developing on separate branch won't impact master line
- Developer can focus on his tasks without caring others
- Preventing long live branches (branch live cycle should be short)

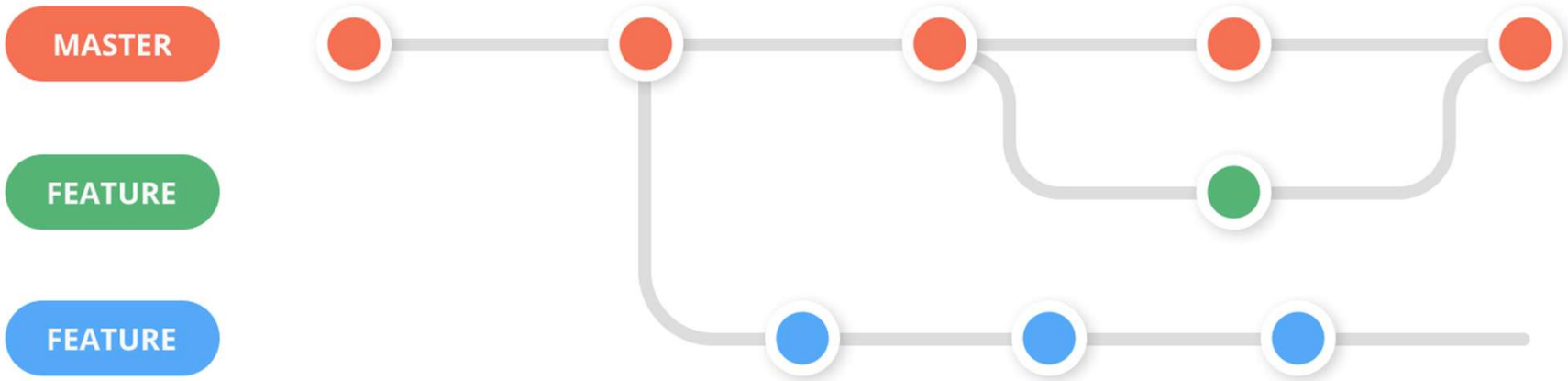
Git workflows

- Develop branch
- Feature branches
- Gitflow
- Github flow
- Gitlab flow
- Trunk based

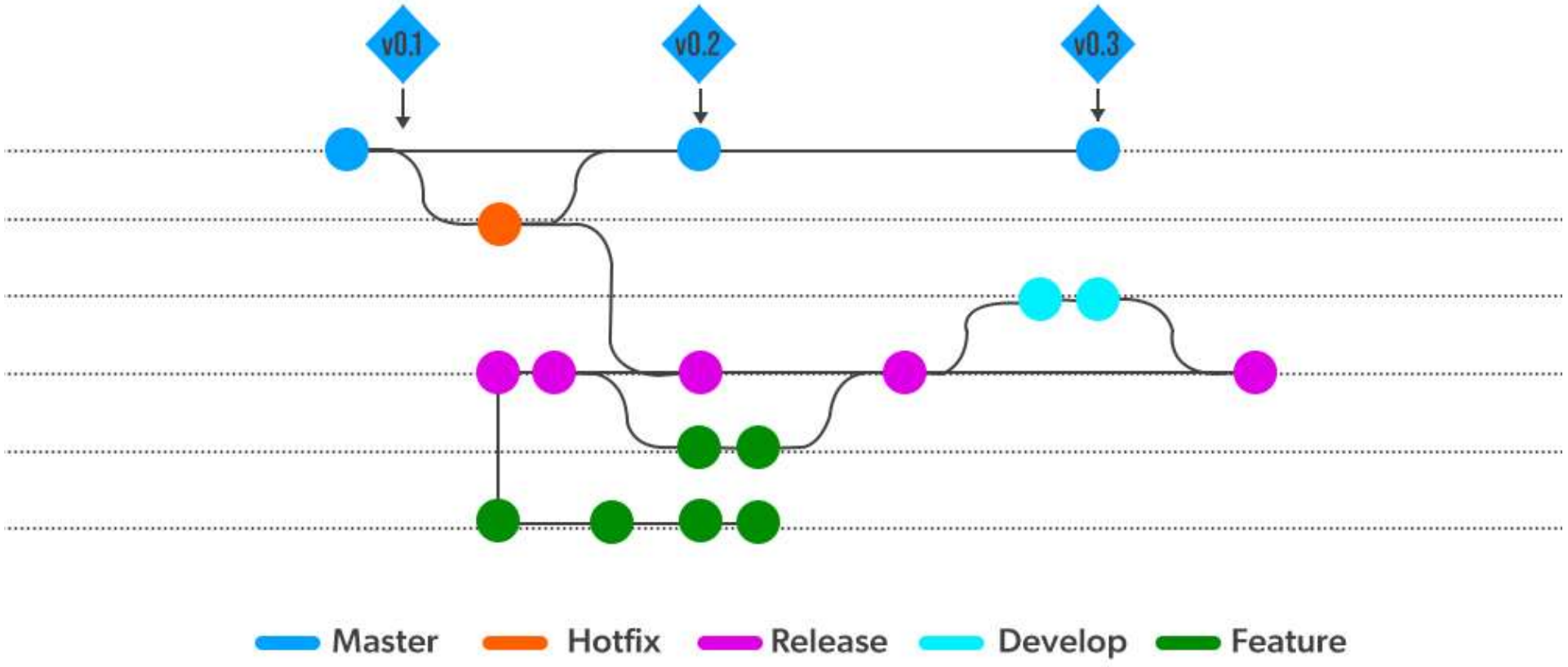
Develop branches



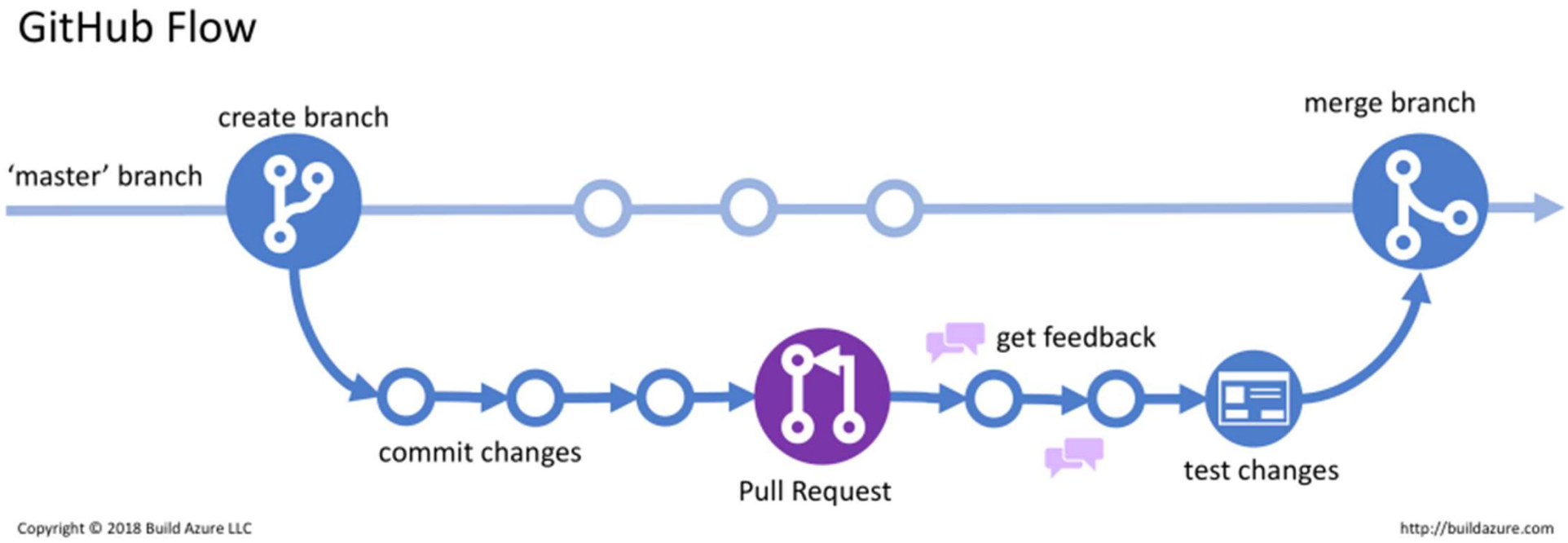
Feature branches



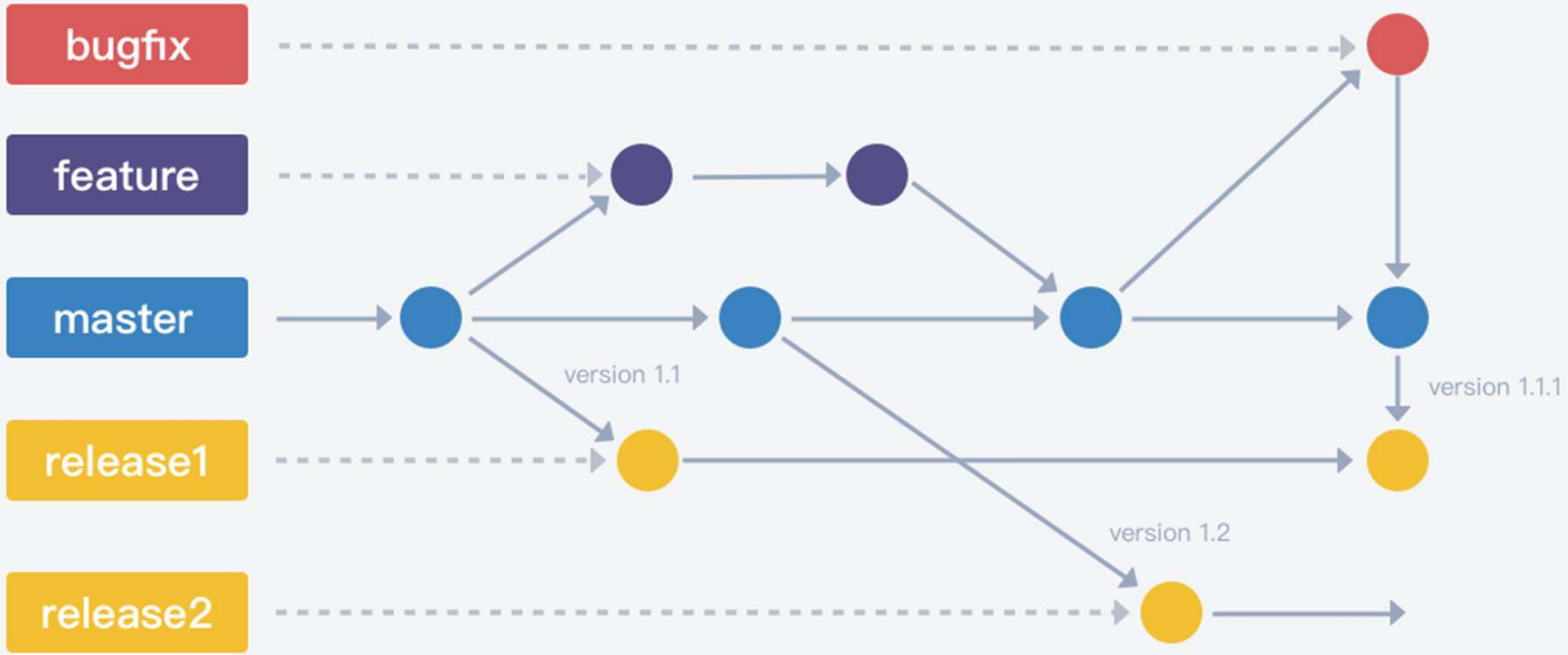
Gitflow



Github flow

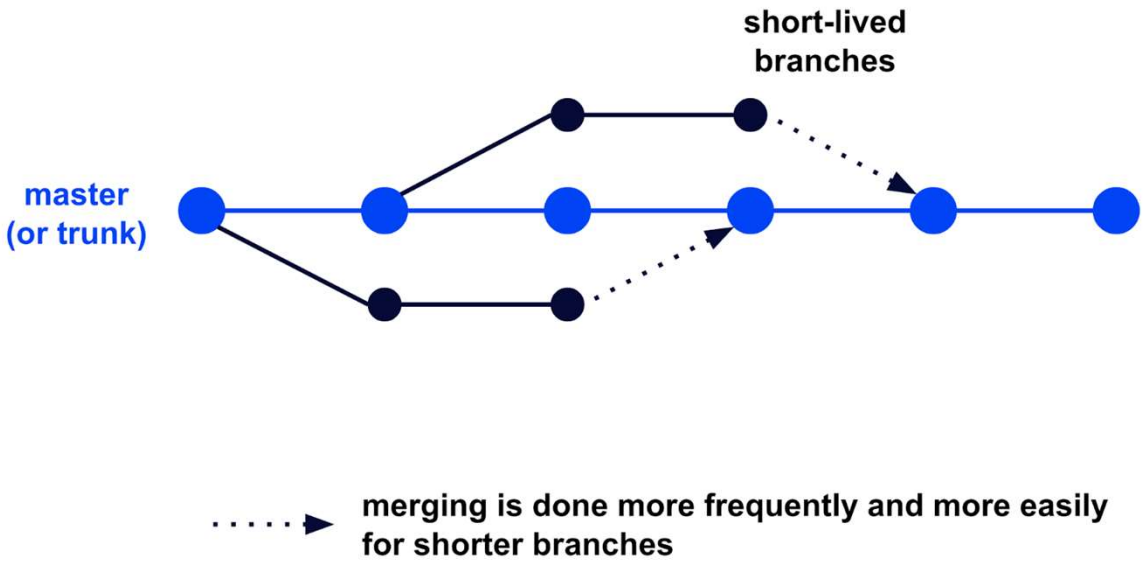


Gitlab flow



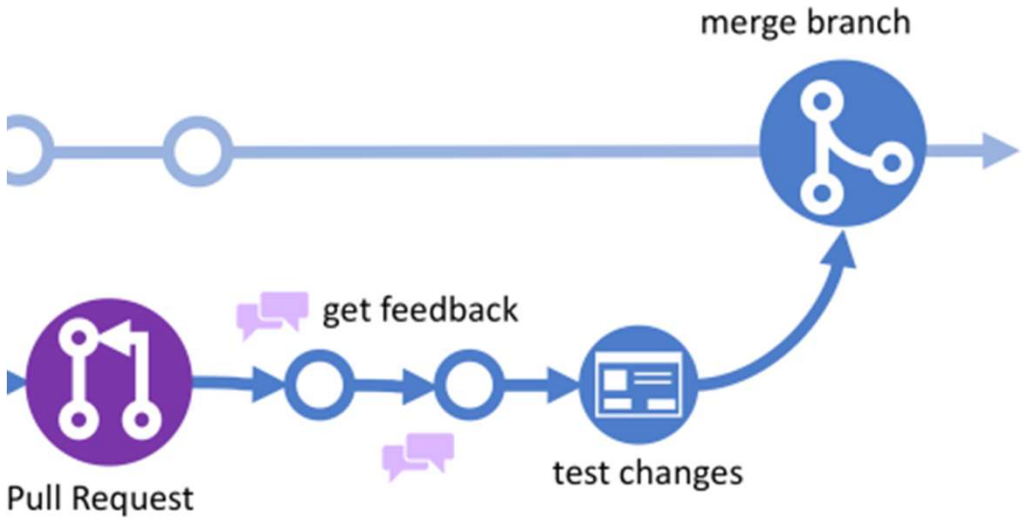
Trunk based development

- Trunk-based development



Ensuring code quality

- **Writing tests (next course)**
  - Unit tests
  - Integration tests
  - End to end tests
  - And etc…
- **Doing quality PR reviews**
  - Peer review
  - Prevent large code commits
  - Check not only happy path



Which is better?

Takeaways

- **Imagine you're trying to develop a "Uber"**
  - Writing user stories
  - Writing acceptance criteria
- **Create a new repository on github**
  - Check in your document of user stories produced by previous requirement
  - Try to branch and modify the document
  - Create a pull request
  - Ask one of your classmate to write review comment and merge