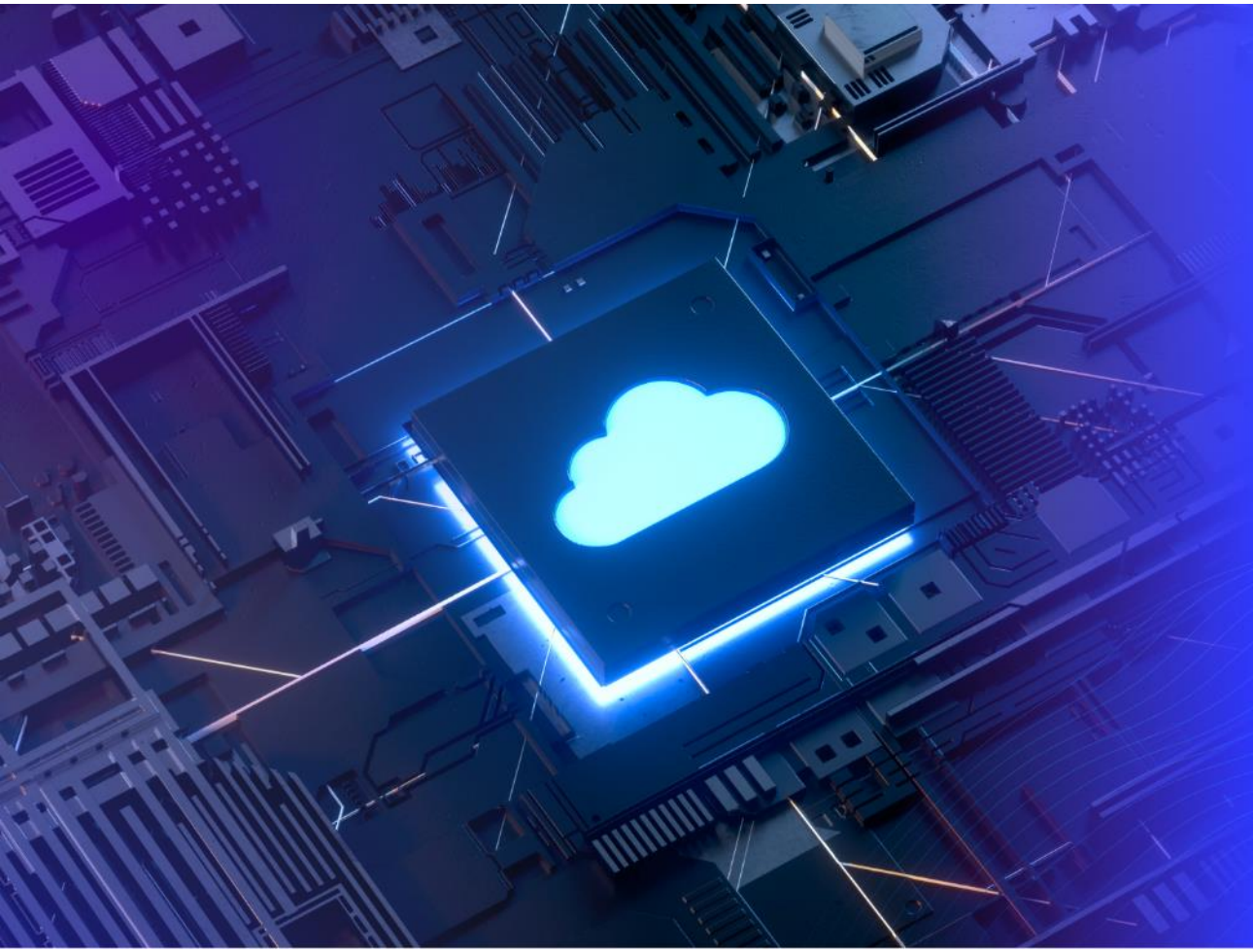
台灣積體電路製造股份有限公司
Taiwan Semiconductor Manufacturing Company, Ltd.



TSMC IT × NCTU CS 課號 5270

CLOUD NATIVE
Development Best Practice

TESTING & DEPLOYMENT

TSID
FOPD | Julian Shen, Kevin Yang
April , 2022

About Us

Julian Shen (沈冠廷)

- 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
  - jkyangc@tsmc.com

AGENDA

- **Testing**
- **Deployment**

Testing

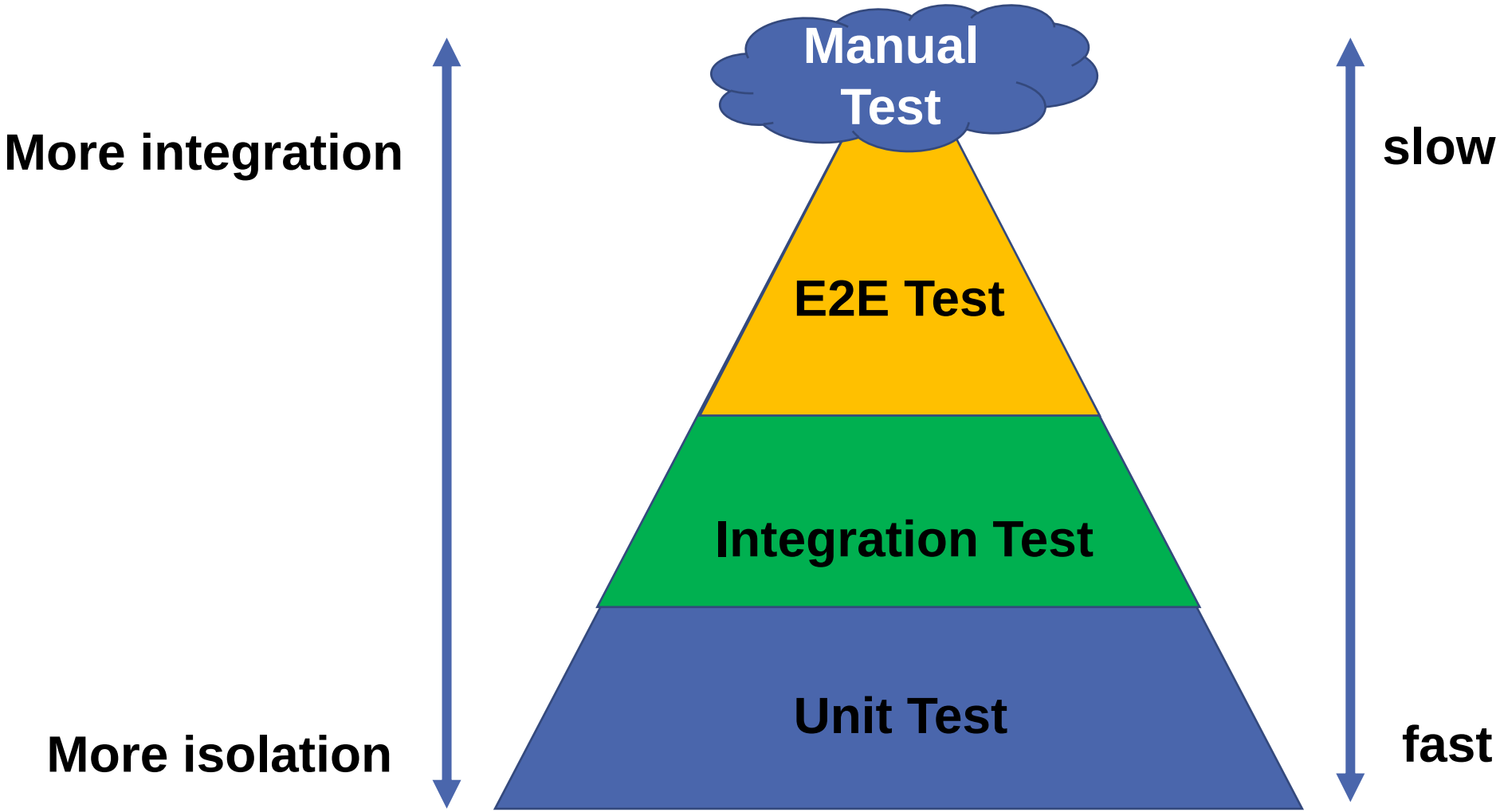
Testing

- Why?
- When?
- What?
- Who?
- How?

Testing

- **The different types of tests**
  - Unit Test (單元測試)
  - Integration Test (整合測試)
  - End to End Testing (E2E測試)
  - Acceptance testing (驗收測試)
  - Performance testing

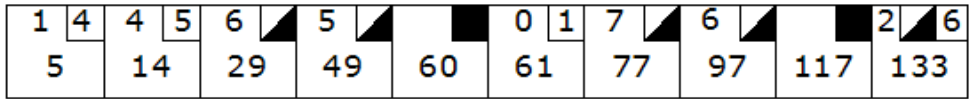
Testing



TDD

- What's TDD (Test-Driven Development)
- Why?
- How?

User story



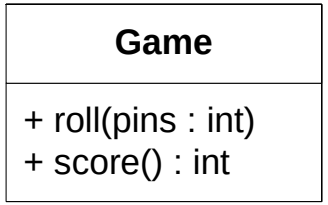
The game consists of 10 frames as shown above. In each frame the player has two opportunities to knock down 10 pins. The score for the frame is the total number of pins knocked down, plus bonuses for strikes and spares.

A spare is when the player knocks down all 10 pins in two tries. The bonus for that frame is the number of pins knocked down by the next roll. So in frame 3 above, the score is 10 (the total number knocked down) plus a bonus of 5 (the number of pins knocked down on the next roll.)

A strike is when the player knocks down all 10 pins on his first try. The bonus for that frame is the value of the next two balls rolled.

In the tenth frame a player who rolls a spare or strike is allowed to roll the extra balls to complete the frame. However no more than three balls can be rolled in tenth frame.

The Requirements



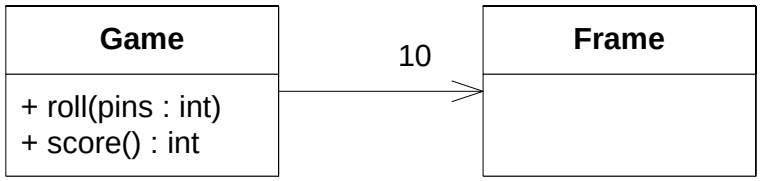
- **Write a class named “Game” that has two methods**
  - **roll(pins : int) is called each time the player rolls a ball. The argument is the number of pins knocked down.**
  - **score() : int is called only at the very end of the game. It returns the total score for that game.**

A quick design session

Game
+ roll(pins : int) + score() : int

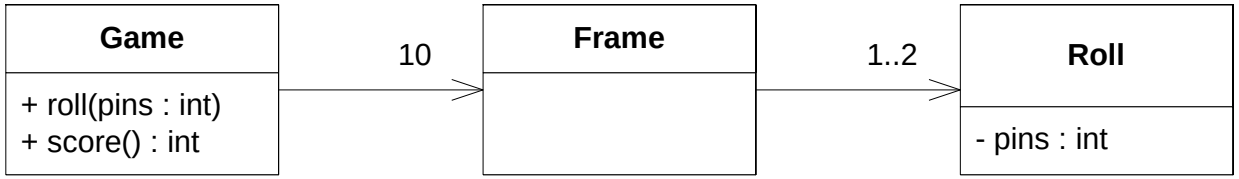
Clearly we need the Game class.

A quick design session



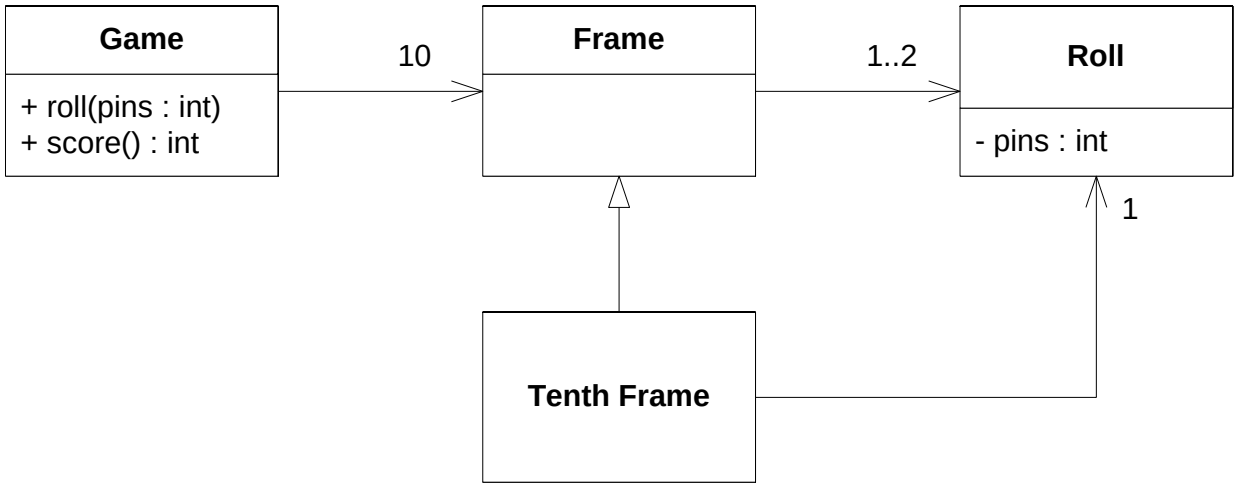
A game has 10 frames.

A quick design session



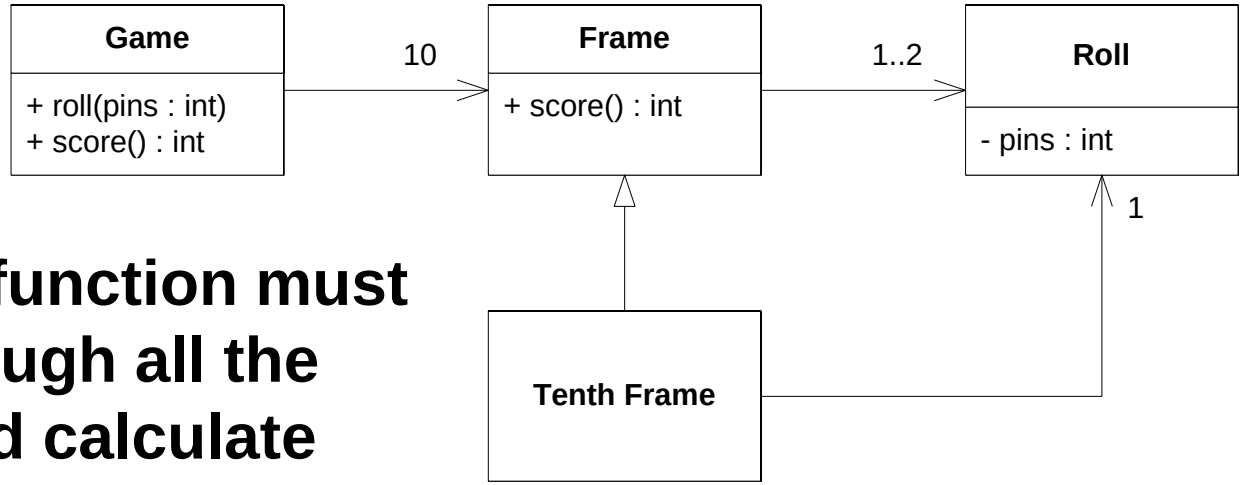
A frame has 1 or two rolls.

A quick design session



The tenth frame has two or three rolls.
It is different from all the other frames.

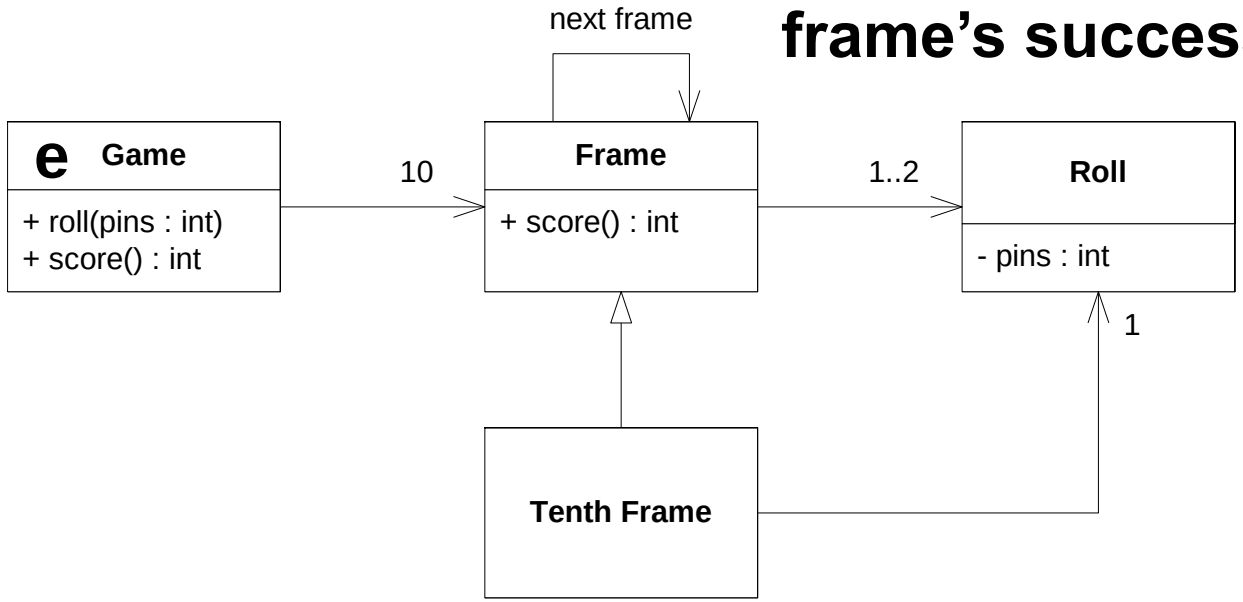
A quick design session



The score function must iterate through all the frames, and calculate all their scores.

A quick design session

The score for a spar or a strike depends on the frame's successor



Live Coding

Deployment

Deployment

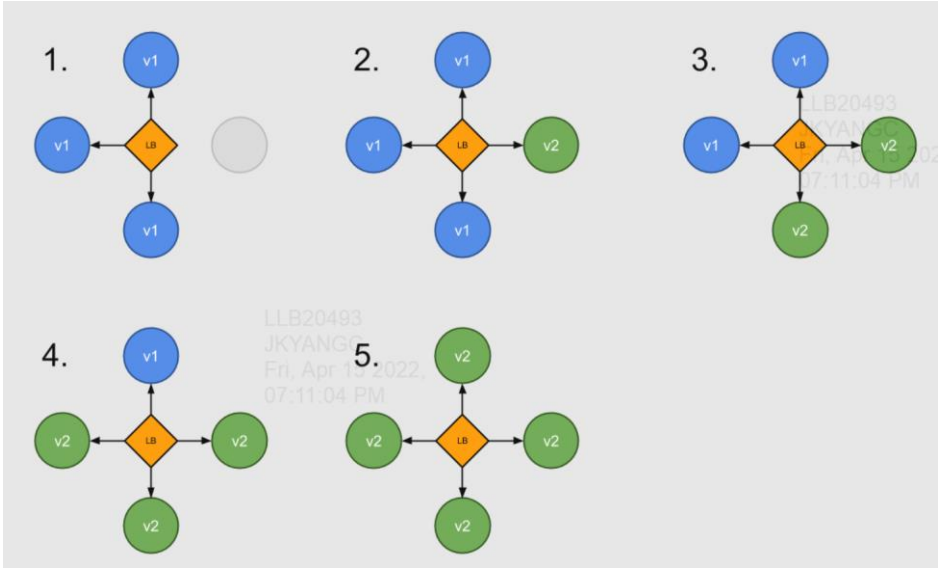
- **deployment strategy**
- **Recreate**: Version A is terminated then version B is rolled out.
- **Ramped** (also known as rolling-update or incremental): Version B is slowly rolled out and replacing version A.
- **Blue/Green**: Version B is released alongside version A, then the traffic is switched to version B.
- **Canary**: Version B is released to a subset of users, then proceed to a full rollout.
- **A/B testing**: Version B is released to a subset of users under specific condition.

Deployment - Recreate

```yaml
spec:
  replicas: 3
  strategy:
    type: Recreate
```

- **Pro:**
- application state entirely renewed
- **Cons:**
- downtime that depends on both shutdown and boot duration of the application

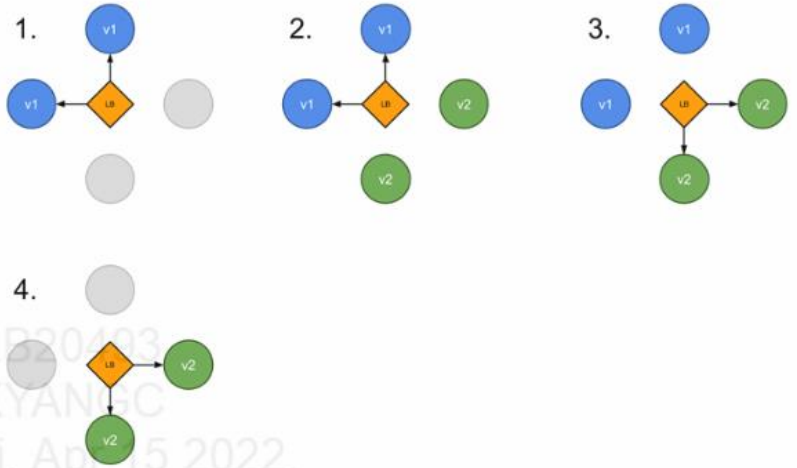
Ramped



Ref: https://blog.container-solutions.com/kubernetes-deployment-strategies

- **Pro:**
  - version is slowly released across instances
  - convenient for stateful applications that can handle rebalancing of the data
- **Cons:**
  - rollout/rollback can take time
  - supporting multiple APIs is hard
  - no control over traffic

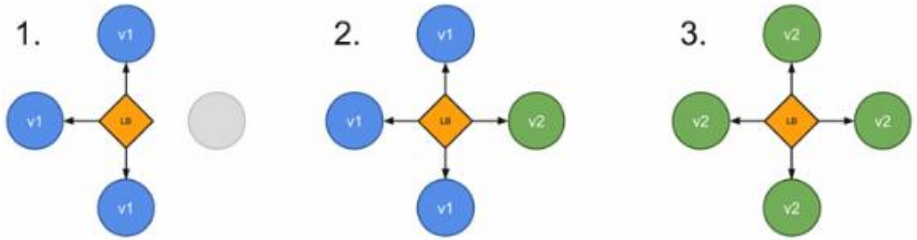
Blue/Green



Ref: https://blog.container-solutions.com/kubernetes-deployment-strategies

- **Pro:**
- instant rollout/rollback
- avoid versioning issue, change the entire cluster state in one go
- **Cons:**
- requires double the resources
- proper test of the entire platform should be done before releasing to production
- handling stateful applications can be hard

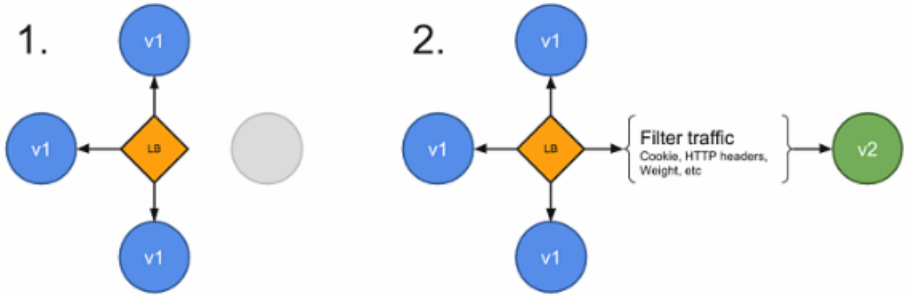
Canary



Ref: https://blog.container-solutions.com/kubernetes-deployment-strategies

- **Pro:**
- version released for a subset of users
- convenient for error rate and performance monitoring
- fast rollback
- **Cons:**
- slow rollout
- fine tuned traffic distribution can be expensive (99% A/ 1%B = 99 pod A, 1 pod B)

A/B testing



- **Pro:**
- requires intelligent load balancer
- several versions run in parallel
- full control over the traffic distribution
- **Cons:**
- hard to troubleshoot errors for a given session, distributed tracing becomes mandatory
- not straightforward, you need to setup additional tools

Deployment – things to consider

- Release management
- Sanity check
- Rollback plan
- Testing
- …etc

Reference

Reference

- The different types of software testing
- Kubernetes deployment strategies
- GitHub Actions
- https://nx.dev/ci/monorepo-ci-github-actions
- Google Kubernetes Engine - GitHub Actions
- 其他補充教材

Assignment

Assignment

- Finish kata Potter with following requirements.
  - Must have commit history of each steps
  - Use GitHub action to execute testing and publish test result
- https://codingdojo.org/kata/Potter/