

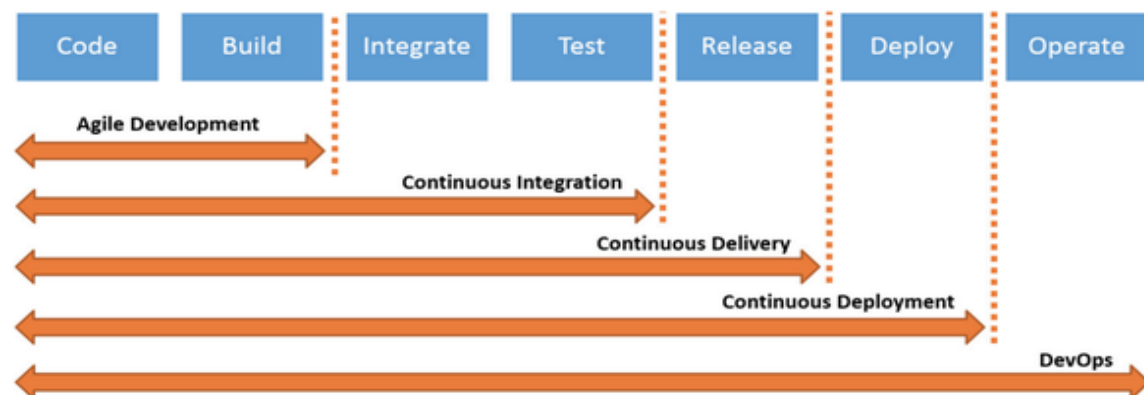
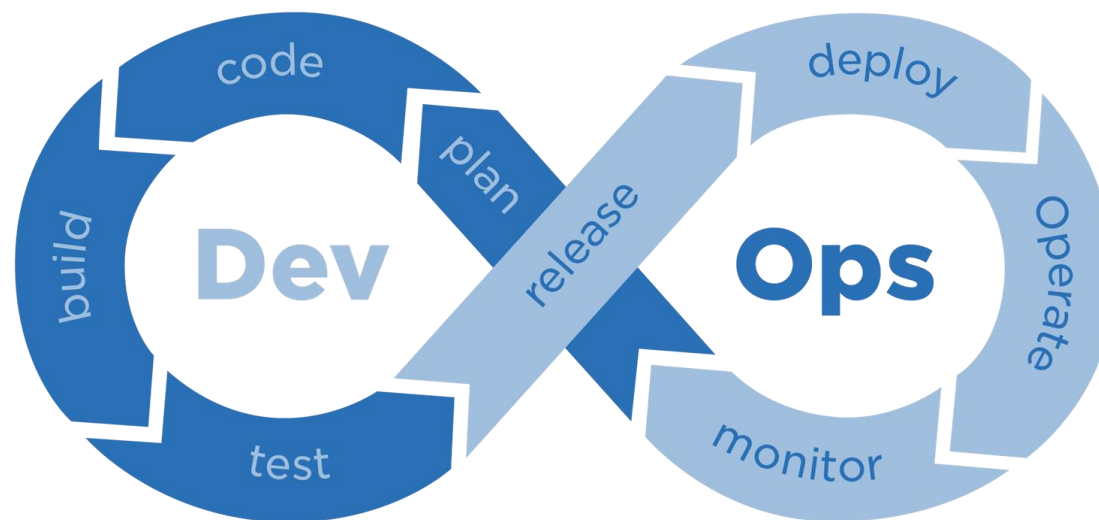
雲原生開發 DevOps CI/CD 持 整合與交付部署自動化

WHO AM I - 高傳詔

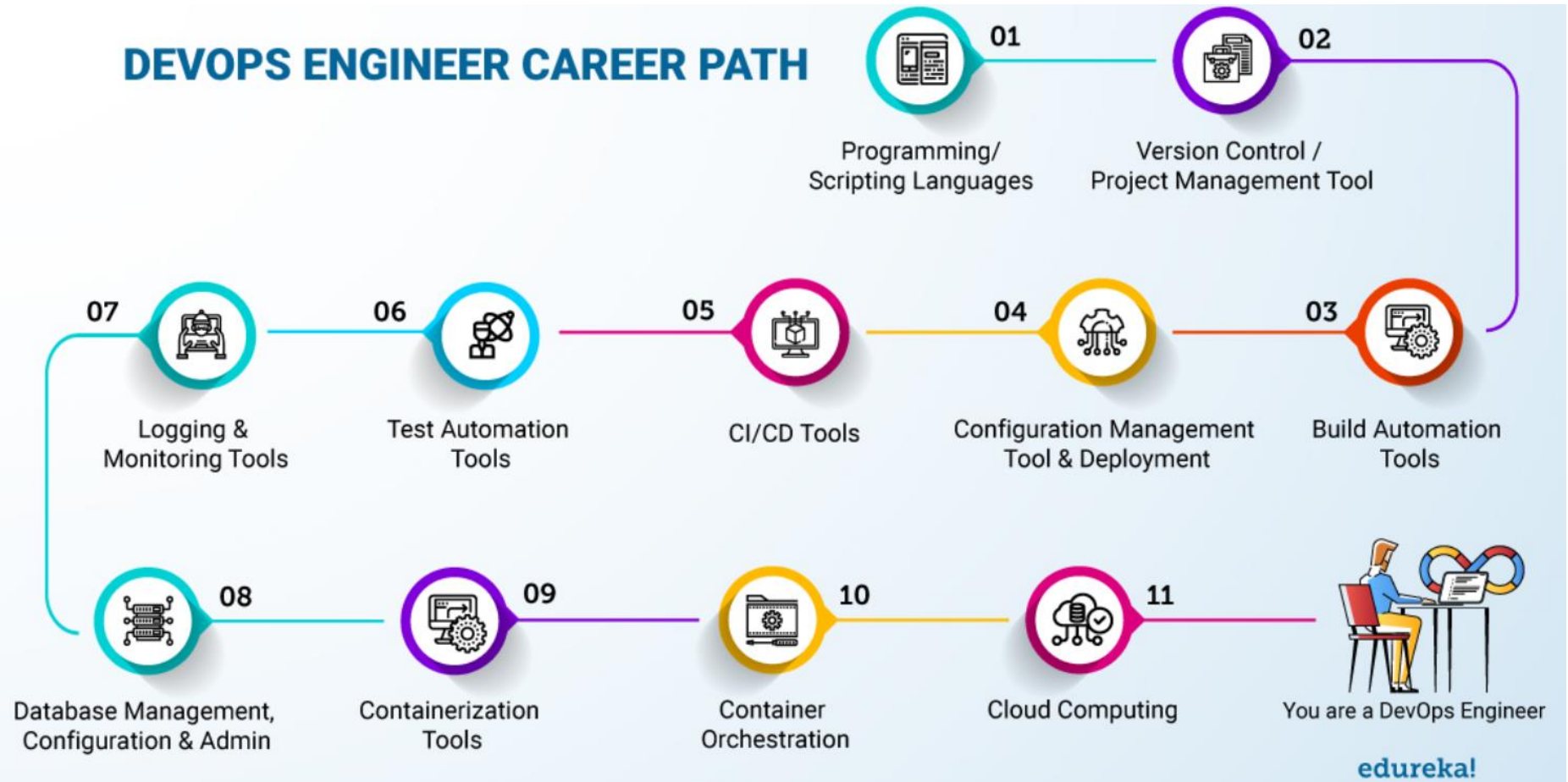
- ❑ WAT, Scanner products development & site operation
- ❑ Development Tool Chain
- ❑ Real time fab collaboration system
- ❑ DevOps transformation, especially for Ops

- ❑ Experienced programming languages
 - Pascal, Assembly, NJ/SML, Python
 - Perl, MFC, C/C++, C#, Java, Scala, ActionScript, JavaScript, dart, go lang (now)

- ❑ Contact: cjkao@tsmc.com



DEVOPS ENGINEER CAREER PATH



A pit stop video 1950, vs now



https://youtu.be/RRy_73ivcms

Agenda

- ❑ Challenge and promise of DevOps
- ❑ DevOps history
- ❑ 3 Pillars of DevOps
 - Flow
 - Feedback
 - Continue learning
- ❑ DevOps Engineer – Version Control
- ❑ DevOps Engineer – Continuous Delivery

Problem

- ❑ We are not able to deploy production changes in minutes or hours, instead requiring weeks or months preparation.
- ❑ We are not able to deploy hundreds or thousands of changes into production per day; instead, we struggle to deploy monthly or even quarterly.
- ❑ In production deployments routine, involving war room and chronic firefighting and heroics.

The promise of DevOps

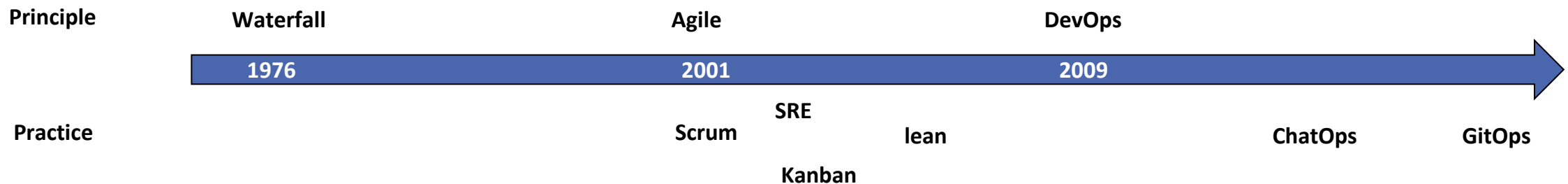
- ❑ Faster delivery of core functionalities online, better teamwork, speedier resolution of issues.
- ❑ The only caveat: DevOps must be applied systematically, so its delivery is not slowed down by legacy applications that were not designed for — nor support — agile and DevOps methodologies.

Software delivery performance metric	Elite	High	Medium	Low
🔄 Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
🕒 Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
🕒 Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
⚠️ Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

DevOps History

- The "official beginning" of the DevOps movement can be traced back to 2009, when Patrick Debois organised the first devopsdays conference.
- In 2016, Patrick Debois, Gene Kim and Jez Humble published [The DevOps Handbook](#): How to Create World-Class Agility, Reliability, and Security in Technology Organizations

Software Development Life Cycle



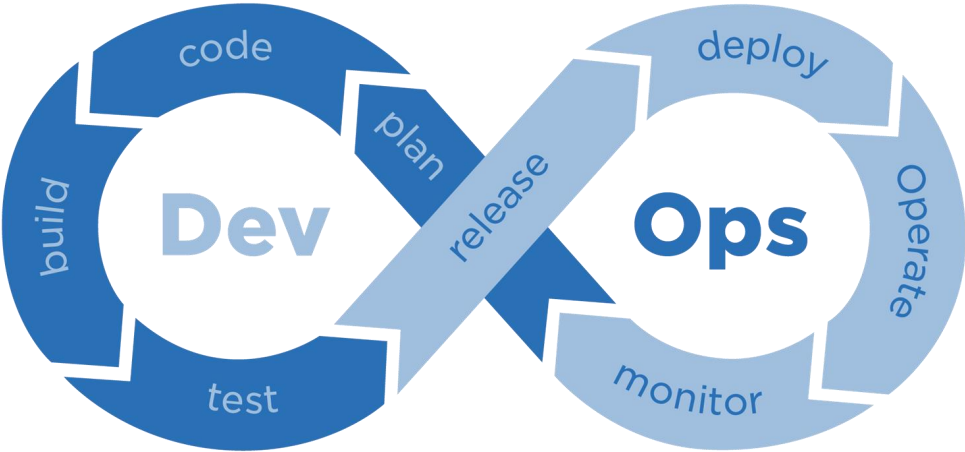
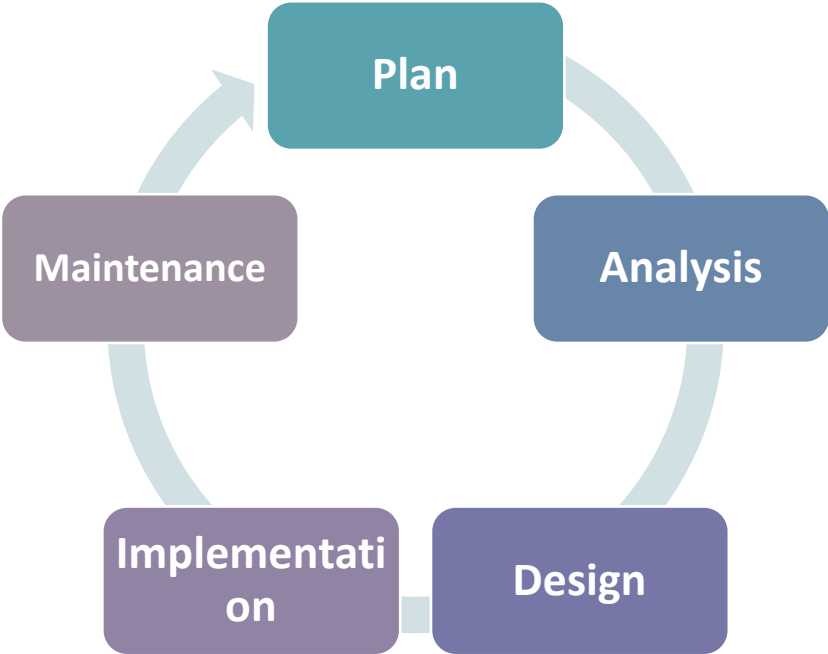
Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

- ❑ **Individuals and interactions** over processes and tools
- ❑ **Working software** over comprehensive documentation
- ❑ **Customer collaboration** over contract negotiation
- ❑ **Responding to change** over following a plan

<https://agilemanifesto.org/>

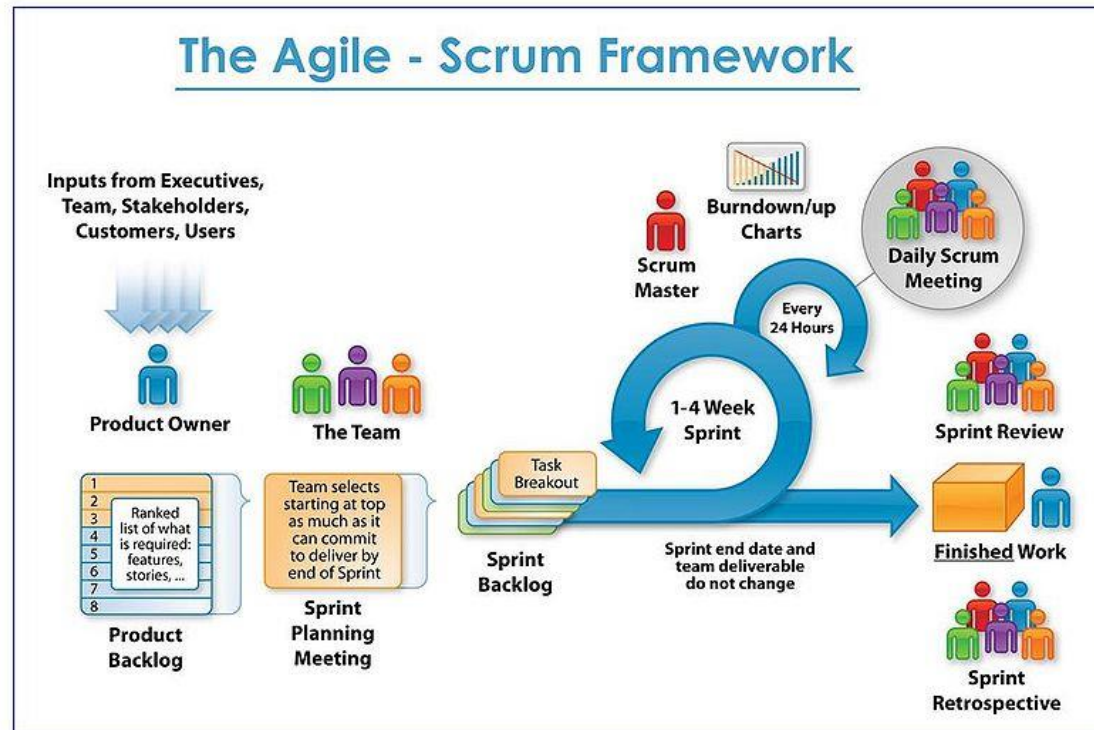
Waterfall versus DevOps



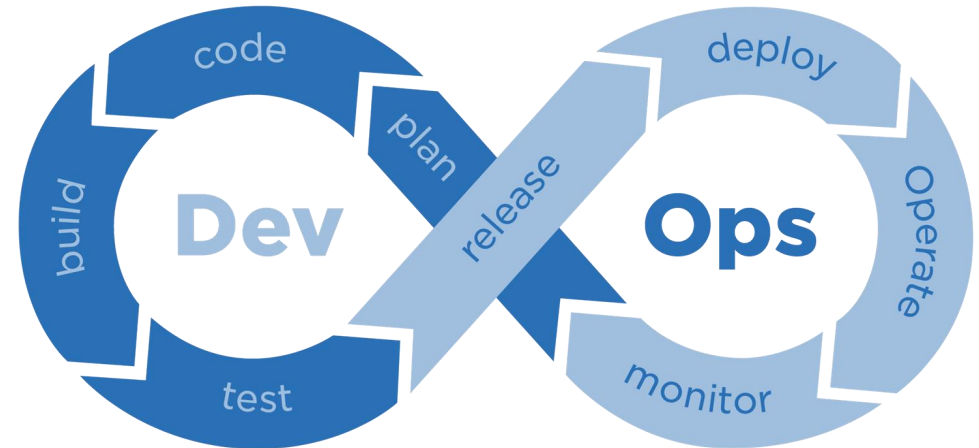
Agile vs DevOps

Tailor process to your team, not just follow

Agile - Scrum



DevOps



Agenda

- ❑ Challenge and promise of DevOps
- ❑ DevOps history
- ❑ 3 Pillars of DevOps
 - Flow
 - Feedback
 - Continue learning
- ❑ DevOps Engineer – Version Control
- ❑ DevOps Engineer – Continuous Delivery

DEVOPS - FLOW

Flow

- ❑ You need a smooth flow of work through your value stream
- ❑ Enabling flow means enabling work in the system to move quickly; it's the part of software development often compared to our factory floor.
- ❑ You could consider different "work stations" through which you want to quickly move "work items".
- ❑ Your stations could be "business story being written", "code being written", "code being tested", "**code merge to main branch and deployed to production**".

Example Backlog

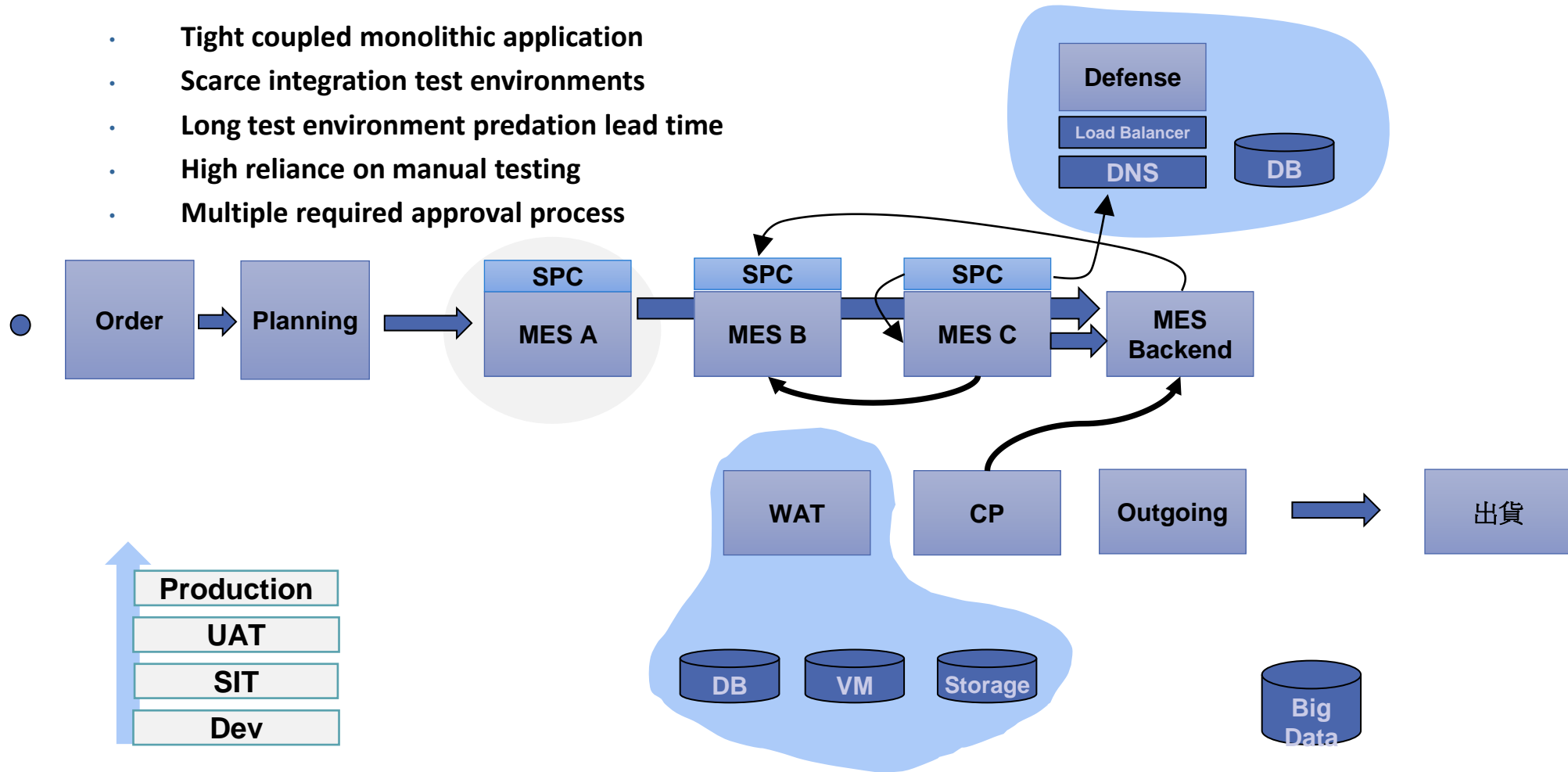
❑ Make work visible, Limit Work In Process

Some day	This sprint 6 / 10	This week 3 / 10	Tomorrow 5 / 5	Today 5 / 5	In progress 4 / 3	>
+ add task	+ add task	+ add task	+ add task	+ add task	+ add task	Done (5)
As a user I can edit all of my details in Profile View	1/25 As a user I can edit all of my details in Profile	Prep. documents for the tax office	Test the Cycle Time script changes	Test new image compressing solutions	Update disavow file	
As a user I want to have access to all recent changes	Call Morty 9252458978	Hire a cleaning service for the Manchester office	Deactivate promo codes	As client I can give feedback through my account, no questions asked	New users should be able to find their recent files	
As an admin I can edit any of the users details	As PM I can review all code before updating the website	Fix issues with cms page 4758	Deliver promo codes to resellers	Each KL's page has to be made searchable	More bugs with SF-78 feature!	
As a user I can sign out	As page admin I can change the layout any time		New bugs in the activity report	Review all code connected to my present work	Call Vijay C. 358738573	
	As an admin I can see all my team activity		German text bug fix on page 4022	Order a gift for Jake		
	As a user I can cancel subscriptions any time		Enter new task here...			

Sample Project

Common Issue:

- Tight coupled monolithic application
- Scarce integration test environments
- Long test environment predation lead time
- High reliance on manual testing
- Multiple required approval process

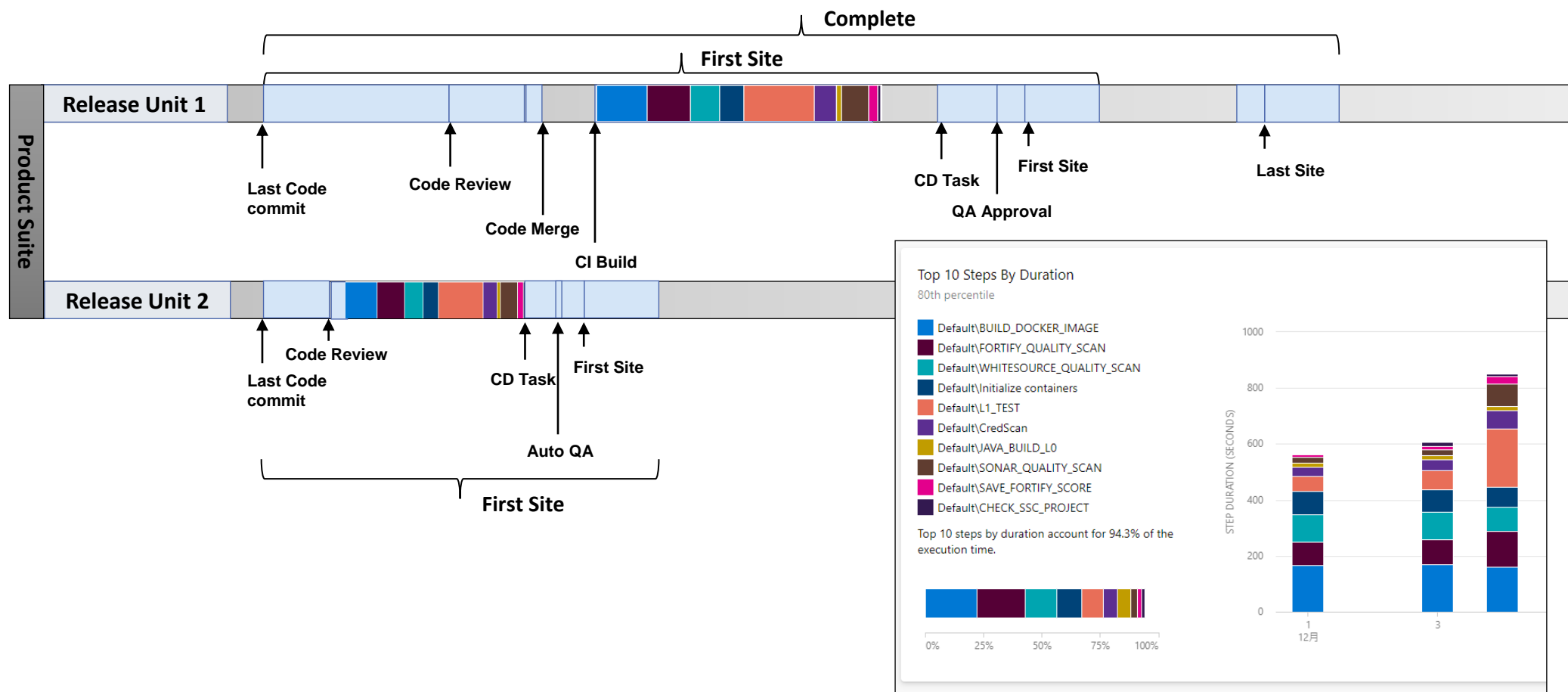


Flow Check List

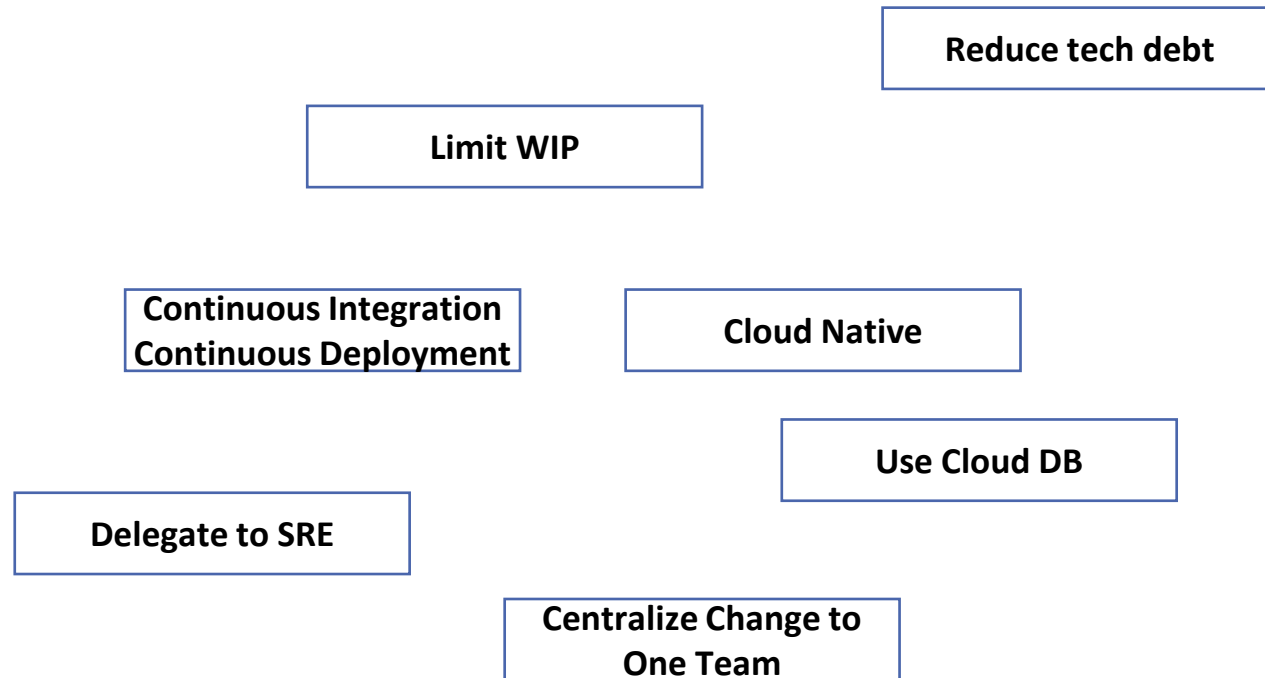
- ❑ How to ensure new change won't break other systems ?
- ❑ How to avoid manual deployment application more than once ?
 - Do we align all version? If not, when to fix it?
- ❑ How to move compiled and packed artifact to runtime environment ?
- ❑ Where to see log if application deploy to multiple instances?
- ❑ How long to provision VM, storage, DB, network, DNS, LB, firewall ?
- ❑ How to validate data against project goal in Dev, SIT, UAT ?
- ❑ How many people involved to approve change ?
- ❑ How many effort to setup monitor ? Does monitor effective ?
- ❑ How to recover system if some service unhealth? Does data corrupted ?
- ❑ How to upgrade/decommission old instance without impact user?
- ❑ Does my sanity check after system change 100% manual ?

Software Deliver Performance Analysis

Assist product manager analysis product delivery bottleneck



How to speed up flow?



How to speed up flow?

Continuous Integration

Cloud Native

Continuous Deployment

CONTINUOUS INTEGRATION

Continuous Integration

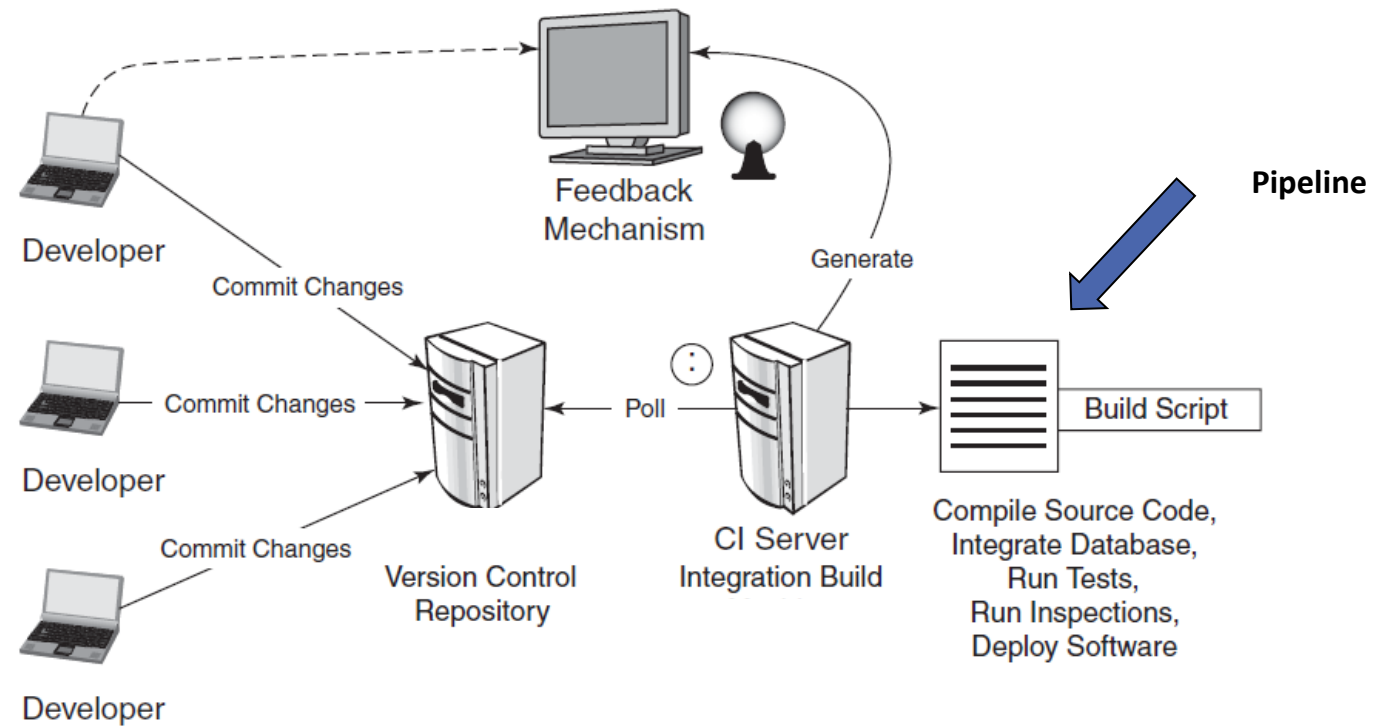
Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly. This article is a quick overview of Continuous Integration summarizing the technique and its current usage.

By Martin Fowler

Scope Of Continuous Integration

- ❑ All developers run private builds on their own machine before committing their code to the version control repository to ensure that their changes don't break the integration build.
- ❑ Developers commit their code to a **version control** repository at least once a day.
- ❑ Integration builds occur several times a day on a separate build machine.
- ❑ 100% of tests must pass for every build.
- ❑ An artifact is generated (e.g., container image etc.) that can be functionally tested.
- ❑ Fixing broken builds is of the highest priority.
- ❑ Review reports generated by the build, such as coding standards and dependency analysis, vulnerability reports, to seek areas for improvement.

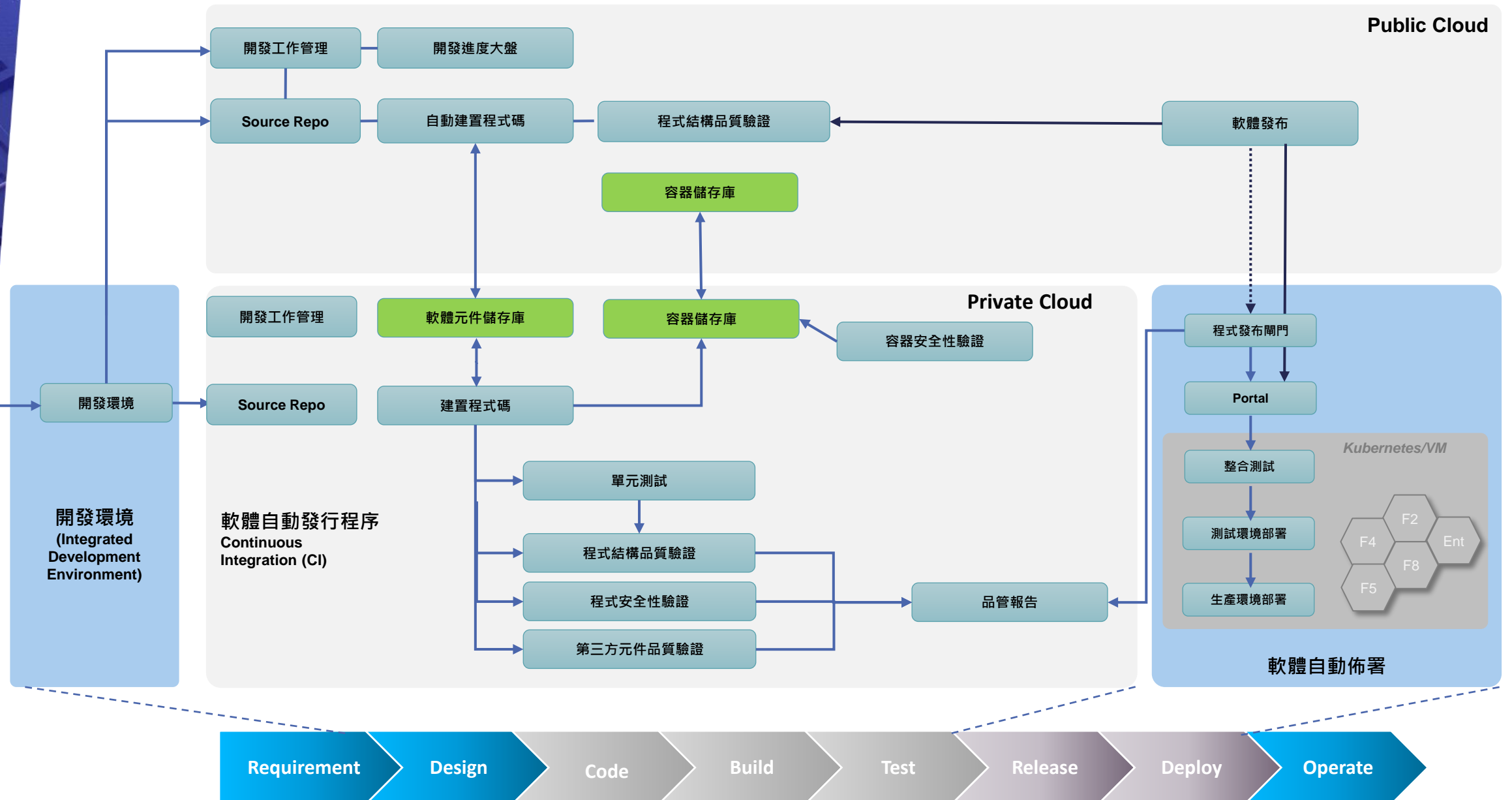
CI structure



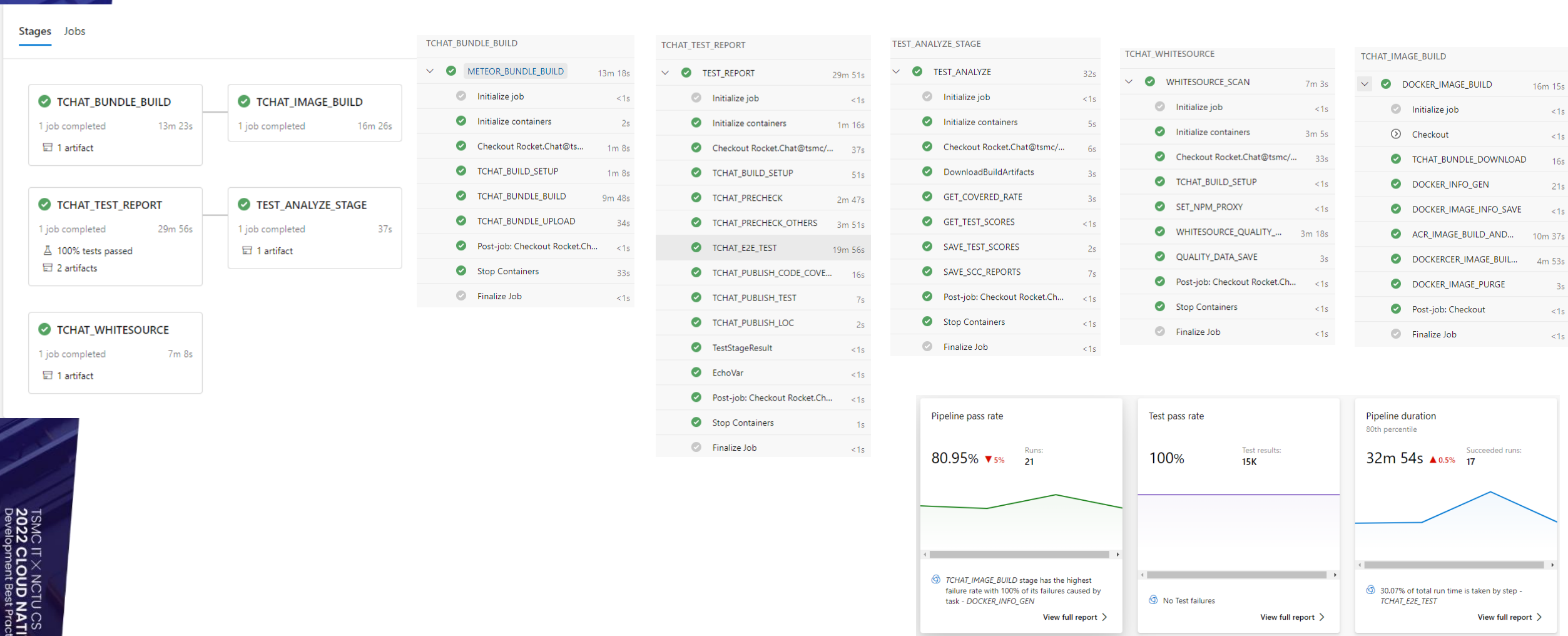
CI check list

- ❑ Do all the software components work together?
- ❑ What is my code complexity?
- ❑ Is the team adhering to the established coding standards?
- ❑ How much code is covered by **automated tests**?
- ❑ Were all the tests successful after the latest change?
 - Does test case only cover happy path?
- ❑ Does my application still meet the performance requirements?
- ❑ Were there any problems with the last deployment?
- ❑ Does my code compliance to security standard?

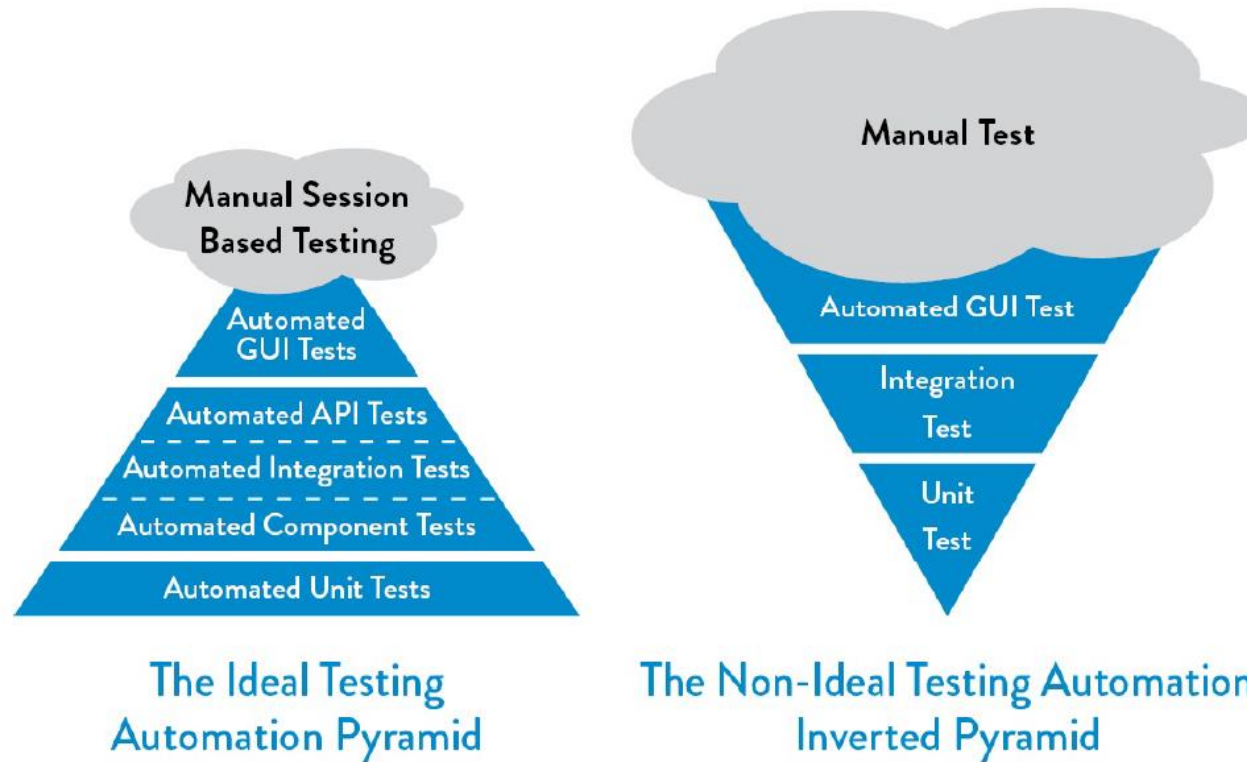
CI Tool Chain Example



Continuous Integration Sample



Why does CI spend a lot of time to run test?



DevOps Handbook

How to speed up flow?

Continuous Integration
Continuous Deployment

Cloud Native

Cloud Native Definition

- ❑ Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. **Containers, service meshes, microservices, immutable infrastructure, and declarative APIs** exemplify this approach.
- ❑ These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined **with robust automation**, they allow engineers to make high-impact changes frequently and predictably **with minimal toil**.

Cloud Native and DevOps



Architecture

Monoliths

SOA

Micro Service

Infrastructure

Physical Server

VM

Cloud

Delivery

Waterfall

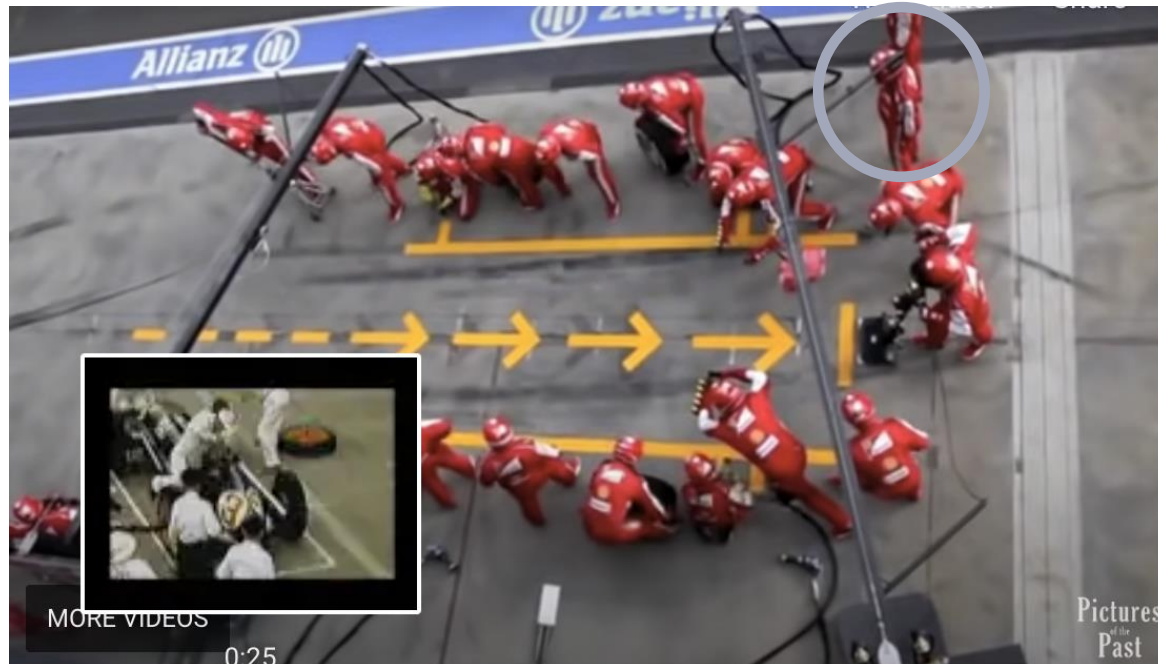
Agile

DevOps

Agenda

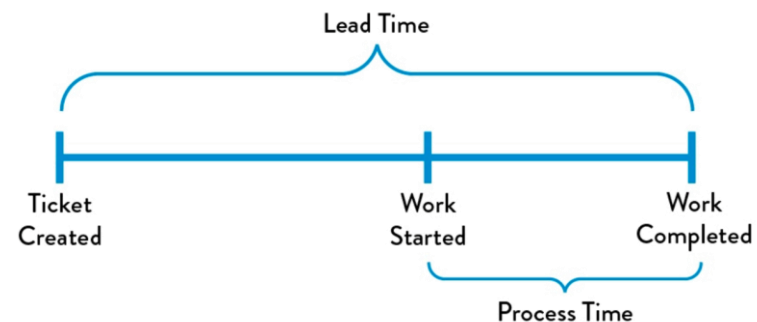
- ❑ Challenge and promise of DevOps
- ❑ DevOps history
- ❑ 3 Pillars of DevOps
 - Flow
 - Feedback
 - Continue learning
- ❑ DevOps Engineer – Version Control
- ❑ DevOps Engineer – Continuous Delivery

Feedback



Measurement

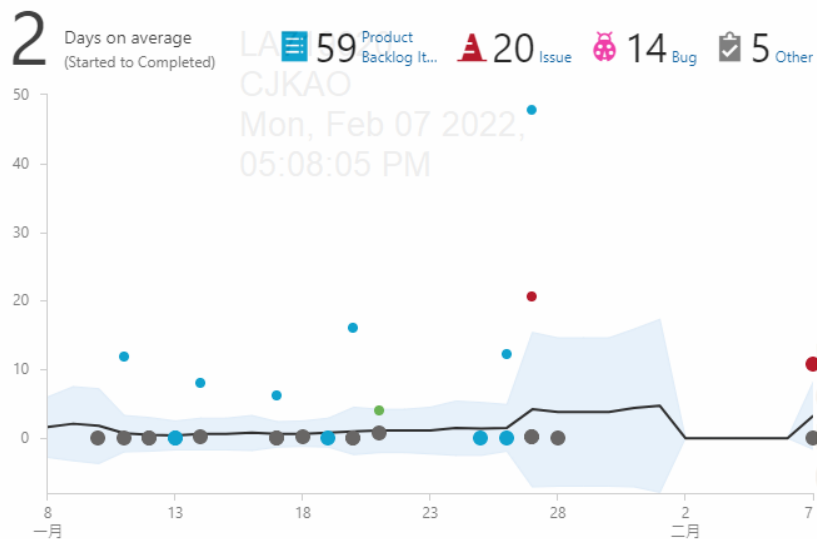
海森堡不確定性原理：粒子的位置與動量不可同時被確定，位置的不確定性越小，則動量的不確定性越大，反之亦然



Backlog Cycle Time

DAPD-04 Cycle Time

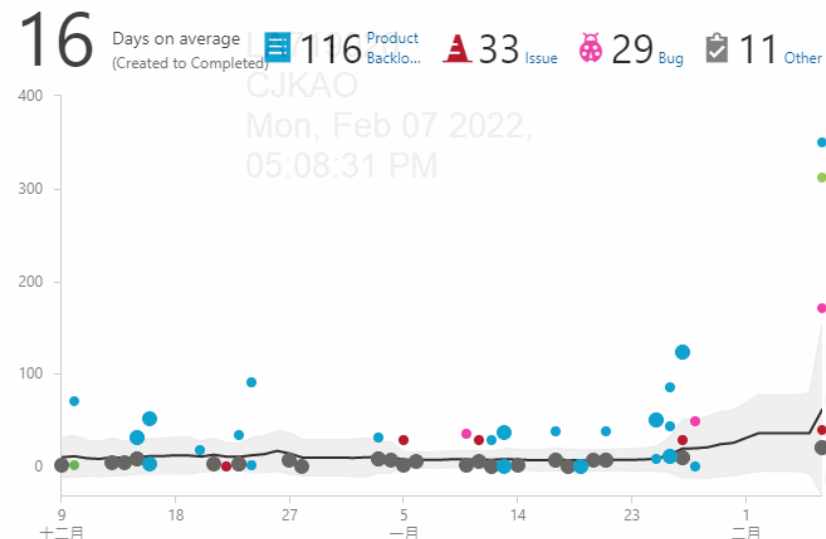
Last 30 days



Backlog Lead Time

DAPD-04 Lead Time

Last 60 days



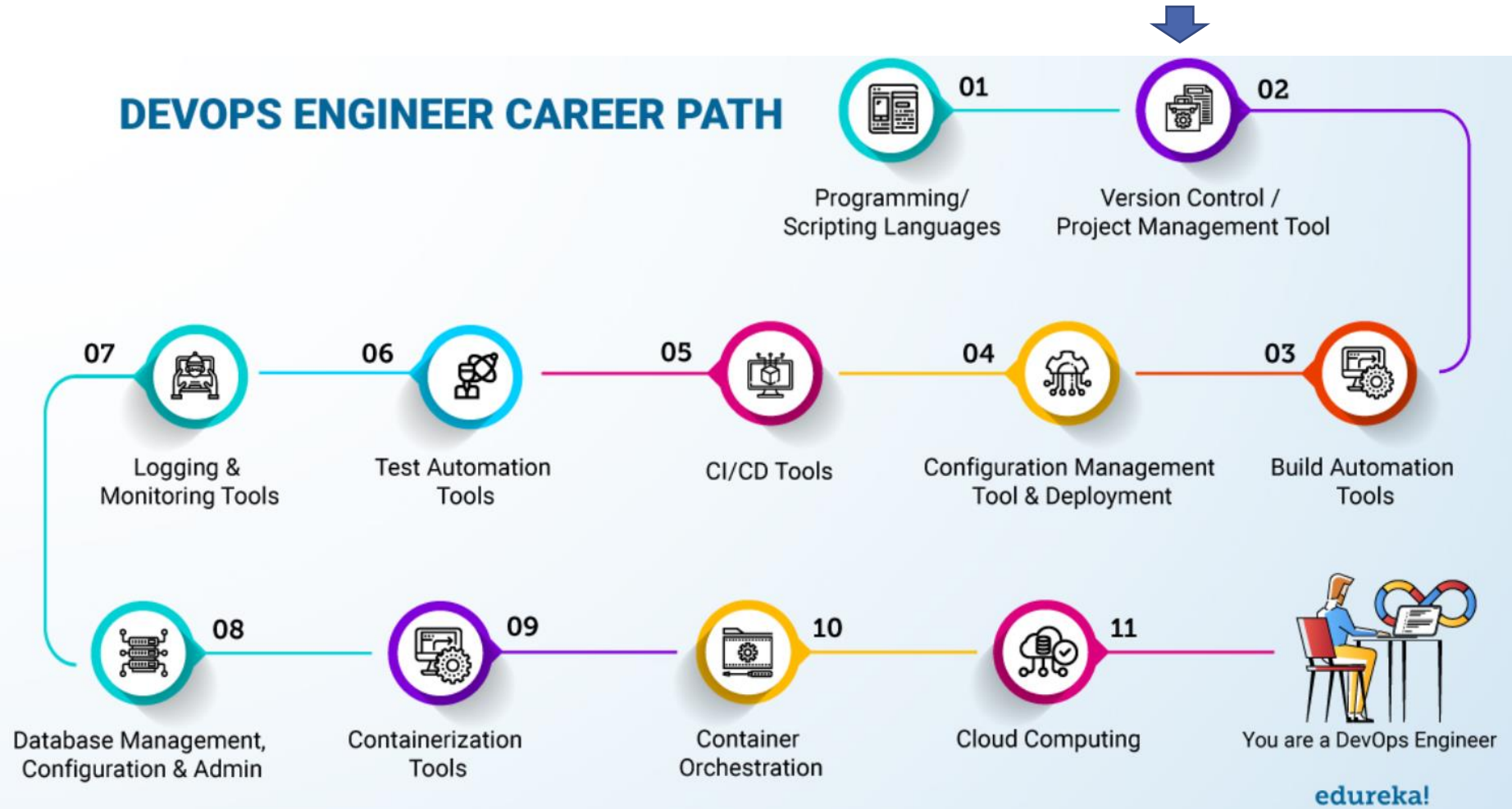
The Feedback Practice

- ❑ **Create Telemetry to Enable Seeing and Solving Problems**
- ❑ **Analyze Telemetry to Better Anticipate Problems and Achieve Goals**
- ❑ **Enable Feedback So Development and Operations Can Safely Deploy Code**
- ❑ **Integrate Hypothesis-Driven Development and A/B Testing into Our Daily Work**
- ❑ **Create Review and Coordination Processes to Increase Quality of Our Current Work**

Agenda

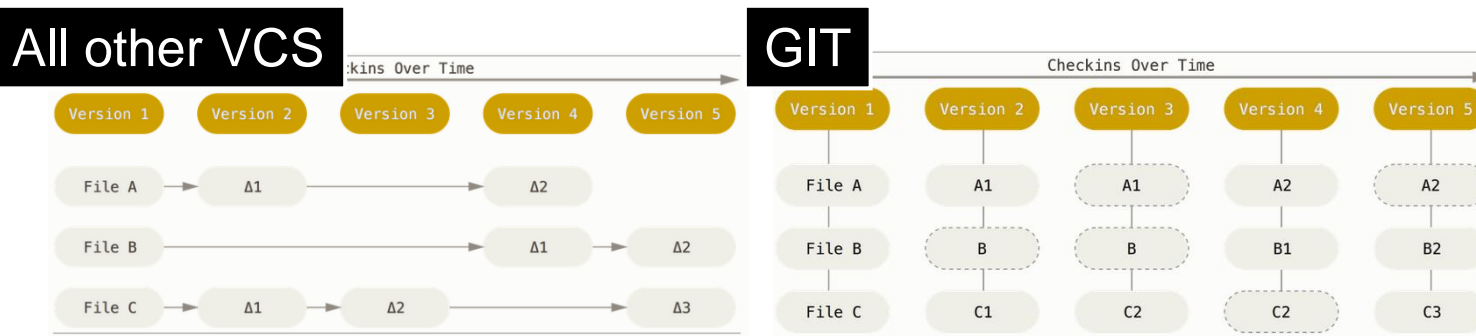
- ❑ Challenge and promise of DevOps
- ❑ DevOps history
- ❑ 3 Pillars of DevOps
 - Flow
 - Feedback
 - Continue learning
- ❑ DevOps Engineer – Version Control
- ❑ DevOps Engineer – Continuous Delivery

DEVOPS ENGINEER CAREER PATH




What is GIT

- **Invent by Linus Torvalds in 2005**, a [distributed revision control](#) system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows
- The Birth of GIT
 - Development of Git began on 3 April 2005
 - The project was announced on 6 April
 - Became [self-hosting](#) as of 7 April
 - The first merge of multiple branches was done on 18 April.
 - Torvalds achieved his performance goals; on 29 April (6.7 patch per second)
 - On 16 June Git managed the kernel 2.6.12 release
 - Torvalds turned over [maintenance](#) on 26 July 2005 to [Junio Hamano](#)



Job Market Required Skills



 [Upload your resume](#) - Let employers find you

Build Engineer

Apple ★★★★★ 1,611 reviews - Santa Clara Valley, CA

Familiarity with version control systems such as subversion or git. Build engineers are at the core of the Apple Software Development Process in delivering the...

9 days ago - [save job](#) - [email](#) - [more...](#)

Senior Software Engineer: Gift Card Systems

Apple ★★★★★ 1,611 reviews - Santa Clara Valley, CA

Experience with subversion, GIT - branching, and merging. Join Apple's Internet Applications Team, within the Information Systems and Technology group, as a...

5 days ago - [save job](#) - [email](#) - [more...](#)

Smart Sign Developer

Apple ★★★★★ 1,611 reviews - Santa Clara Valley, CA

Comfortable with source version control software (SVN, Git). Apple is seeking a Front End Developer to craft high-quality customer experiences for our i....

4 days ago - [save job](#) - [email](#) - [more...](#)

Front-end developer , Global Retail Training

Apple ★★★★★ 1,611 reviews - Santa Clara Valley, CA

Intermediate to Expert with G↓ and github (terminal or GUI). Join Global Retail Training as a Front End Web Developer (UI/UX) to build exciting products....

2 days ago - [save job](#) - [email](#) - [more...](#)

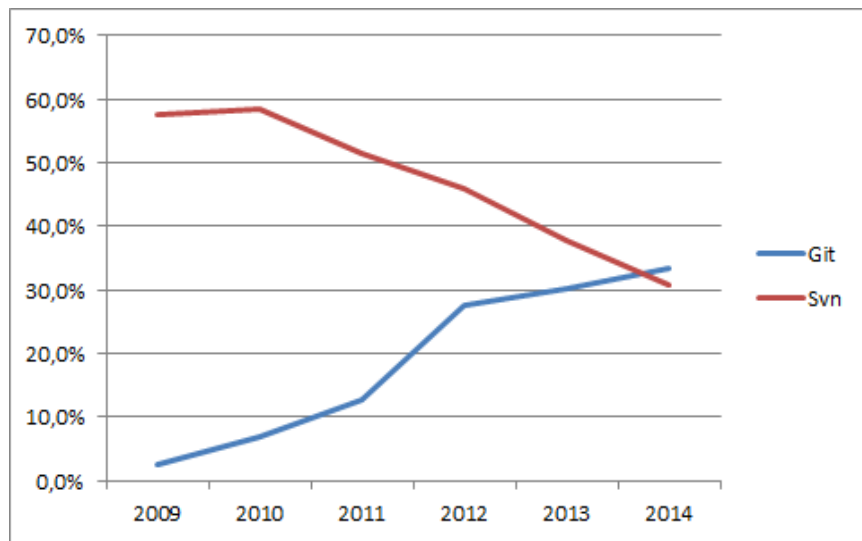
Sr. Application Engineer

Apple ★★★★★ 1,611 reviews - Santa Clara Valley, CA

Familiarity with version control systems such as Git and Subversion. Imagine what you could do here....

11 days ago - [save job](#) - [email](#) - [more...](#)

Cooperate IT Job Demand



- **Subversion**: 2,844 jobs (down from 3,377 on 18 June 2012)
- **Git**: 2,107 jobs (up from 1,208)
- **Team Foundation Server**: 1,772 jobs (up from 1,593)
- **Visual SourceSafe**: 298 jobs (down from 459)
- **ClearCase**: 197 jobs (down from 389)
- **Mercurial**: 187 jobs (up from 172)
- **Perforce**: 142 jobs (down from 204)
- **Borland StarTeam**: 29 jobs (up from 22)
- **AccuRev**: 5 jobs (down from 27)
- **Bazaar**: 5 jobs (no stats for 2012)

**What is the primary source code management system you typically use?
(Choose one.)**

Answer Options	Response Percent	Response Count
BitBucket	3.9%	34
CVS	3.7%	32
Git	33.3%	292
GitHub	9.6%	84
IBM Rational ClearCase	1.5%	13
IBM Rational Team Concert	1.4%	12
Mercurial	2.1%	18
Microsoft Team Foundation Server	1.4%	12
Perforce	0.8%	7
Subversion	30.7%	269
None - I don't use a source code management system	6.4%	56
Don't know	2.9%	25
Other (specify)	2.5%	22
answered question		876

Three Generations of Version Control

VCS	Generation	Networking	Operations	Concurrency	Invent Year
RCS, SCCS, DSEE/ClearCase	First	None	One file at a time	Locks	1972
CVS, SourceSafe, Subversion, Team Foundation Server, RTC	Second	Centralized	Multi-file	Merge before commit	1985
Bazaar, BitKeeper, Git, Mercurial	Third	Distributed	Change sets	Commit before merge	1997~2005

- In first generation tools, concurrent development was **handled solely with locks** (Clear Case). Only one person could be working on a file at a time.
- The second generation tools are a fair bit more permissive about simultaneous modifications, with one notable restriction. **Users must merge the current revisions into their work before they are allowed to commit.**
- The third generation tools **allow merge and commit to be separated.**

GIT's design

- **Strong support for non-linear development**

Git supports [rapid branching and merging](#). A core assumption in Git is that a change will be merged more often than it is written, as it is passed around various reviewers.

- **Distributed development**

Git gives each developer a local copy of the entire development history, and changes are copied from one such repository to another

- **Multiple protocol transport**

Repositories can be published via [HTTP](#), [FTP](#), [rsync](#), sharefolder or a Git protocol over either a plain socket, or [ssh](#)

- **Cryptographic authentication of history**

- **Efficient handling of large projects**

Git as being very fast and scalable,
and performance tests done by Mozilla

Operation	bzr (0.12.0c1)	hg (0.9)	git (1.4.2.4)
diff (top level)	16.957	5.600	1.572
diff in dom/src/base/	16.881	5.504	0.112

- **Toolkit-based design**

- set of programs written in [C](#), and it is easy to chain the components together
- git-MonBo-Jumbo or git funtion name
- Pluggable merge strategies : Resolve, Recursive and octpus

- **Garbage accumulates unless collected**

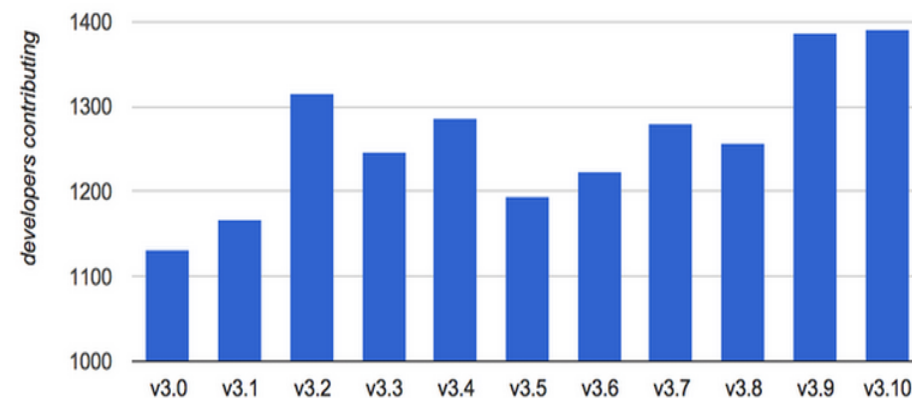
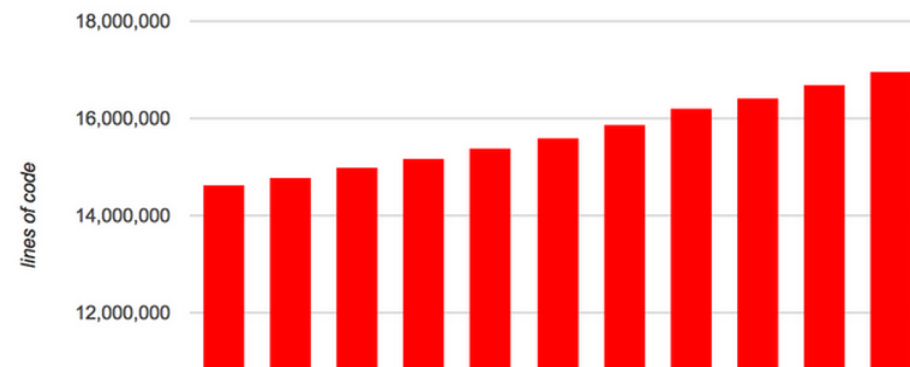
Periodic explicit object packing

Manage very large repo?

When repo grows too large

- GIT – No problem, we can manage Linux kernel, .Net framework, windows
- Other – Out of sync

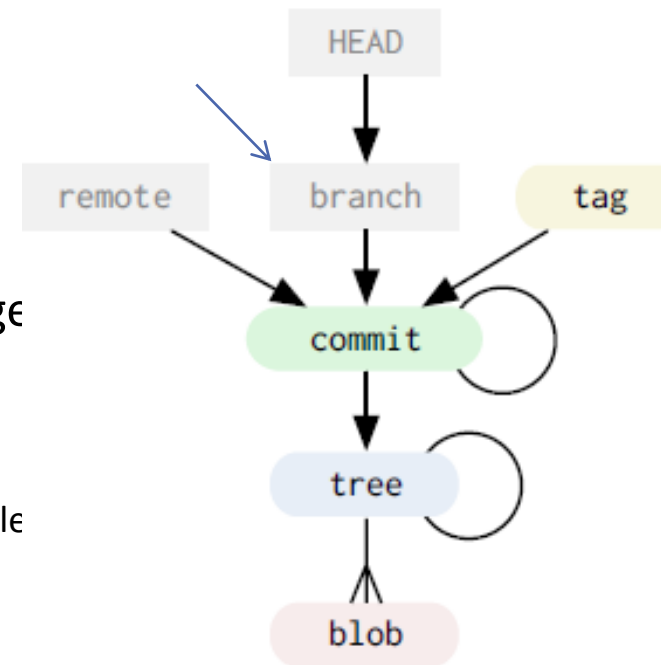
Lines of code in Linux kernel



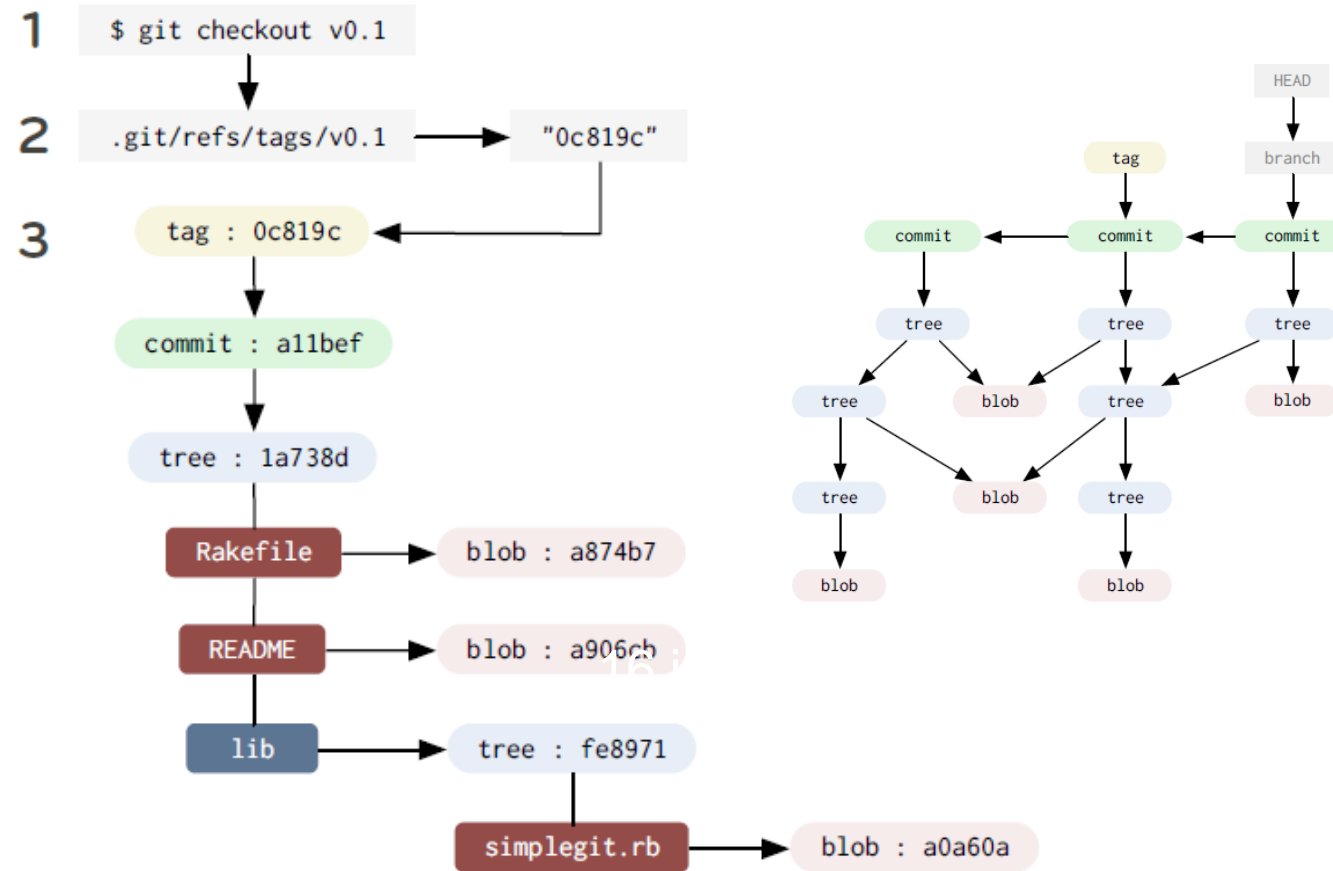
GIT Data Model

Directed Acyclic Graph (DAG)

- Git efficient creates new objects
- Object are generally added, not destroyed
- Unreferenced objects can be garbage collected
- Pack
 - Objects start “loose” but can be packed for file system efficiency
 - “Pack” represent objects as deltas for space efficiency
 - “Pack” also created for transfer between repositories



Tree Traversal



SHA-1 for Data Integrity

2 4 b 9 d a 6 5 5 2 2 5 2 9 8 7 a a 4 9 3 b 5 2 f 8 6 9 6 c d 6 d 3 b 0 0 3 7 3

- Every “Object” has a SHA1 to uniquely identify it
- Everything is check-summed before it is stored and is then referred to by that checksum.

This means it’s impossible to change the contents of any file or directory without Git knowing about it.

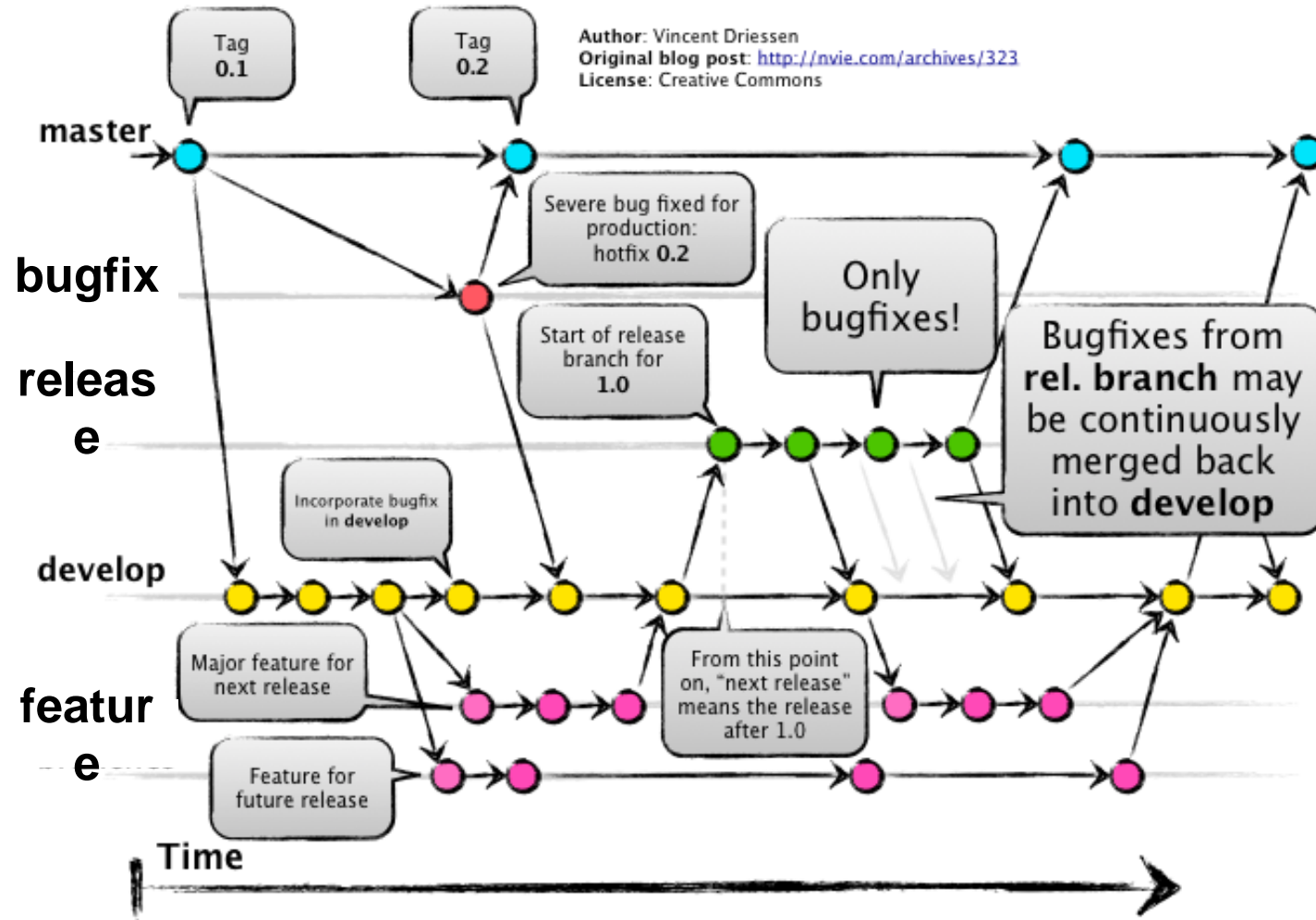
- This functionality is built into Git at the lowest levels and is integral to its philosophy.

You can’t lose information in transit or get file corruption without Git being able to detect it.

Git Workflow

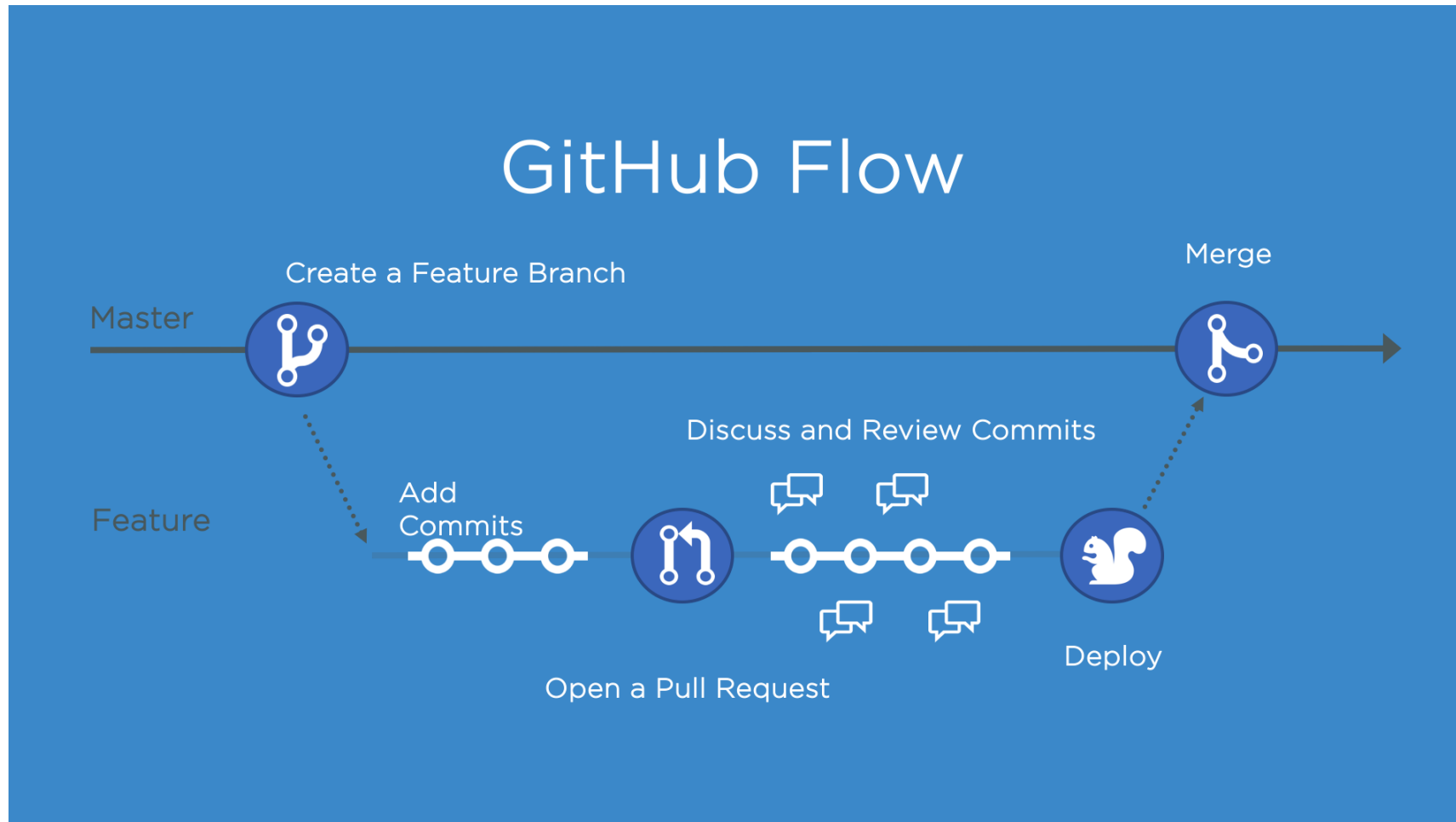
- ▣ **Git flow**
- ▣ **Github flow (trunk base)**

GIT Branch Model



GITHUB FLOW

- The GitHub Flow is a lightweight, branch-based for projects with regular deployments. Aka: trunk base development



Git Cheat Sheet

Remember!
`git <COMMAND> --help`

Global configuration is stored in `~/.gitconfig`.
`git config --help`

master is the default development branch.
origin is the default upstream repository.

🌟 Create

From existing data

```
cd ~/my_project_directory
git init
git add .
```

From existing repository

```
git clone ~/existing_repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://user@host.org/project.git
```

🌟 Show

Files changed in working directory

```
git status
```

Changes made to tracked files

```
git diff
```

What changed between ID1 and ID2

```
git diff <ID1> <ID2>
```

History of changes

```
git log
```

History of changes for file with diffs

```
git log -p <FILE> <DIRECTORY>
```

Who changed what and when in a file

```
git blame <FILE>
```

A commit identified by ID

```
git show <ID>
```

A specific file from a specific ID

```
git show <ID>:<FILE>
```

All local branches

```
git branch
star (*) marks the current branch
```

🌟 Revert

Return to the last committed state

```
git reset --hard
This cannot be undone!
```

Revert the last commit

```
git revert HEAD
Creates a new commit
```

Revert specific commit

```
git revert <ID>
Creates a new commit
```

Fix the last commit

```
git commit -a --amend
(after editing the broken files)
```

Checkout the ID version of a file

```
git checkout <ID> <FILE>
```

🌟 Update

Fetch latest changes from origin

```
git fetch
(this does not merge them)
```

Pull latest changes from origin

```
git pull
(does a fetch followed by a merge)
```

Apply a patch that someone sent you

```
git am -3 patch.mbox
In case of conflict, resolve the conflict and
git am --resolved
```

🌟 Publish

Commit all your local changes

```
git commit -a
```

Prepare a patch for other developers

```
git format-patch origin
```

Push changes to origin

```
git push
```

Make a version or milestone

```
git tag v1.0
```

🌟 Branch

Switch to a branch

```
git checkout <BRANCH>
```

Merge BRANCH1 into BRANCH2

```
git checkout <BRANCH2>
git merge <BRANCH1>
```

Create branch BRANCH based on HEAD

```
git branch <BRANCH>
```

Create branch BRANCH based on OTHER and switch to it

```
git checkout -b <BRANCH> <OTHER>
```

Delete branch BRANCH

```
git branch -d <BRANCH>
```

🌟 Resolve merge conflicts

View merge conflicts

```
git diff
```

View merge conflicts against base file

```
git diff --base <FILE>
```

View merge conflicts against your changes

```
git diff --ours <FILE>
```

View merge conflicts against other changes

```
git diff --theirs <FILE>
```

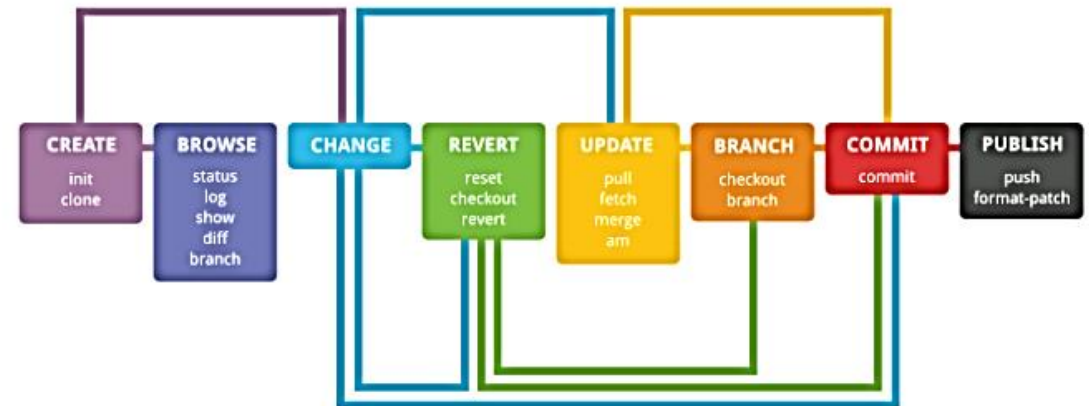
Discard a conflicting patch

```
git reset --hard
git rebase --skip
```

After resolving conflicts, merge with

```
git add <CONFLICTING_FILE>
git rebase --continue
```

🌟 Workflow



Common questions

- Windows & Unix have different new line chars (EOL), thus setup git not to do automatic conversion should help you to get trouble on CI build

```
`git config --global core.autocrlf false`
```

- How to ignore those temporal file? <https://github.com/github/gitignore>
- If I accidentally check-in a binary file, how to remove it?

Homework

- ▣ Complete challenge in <https://learngitbranching.js.org/>
- ▣ Run GitHub Action

Agenda

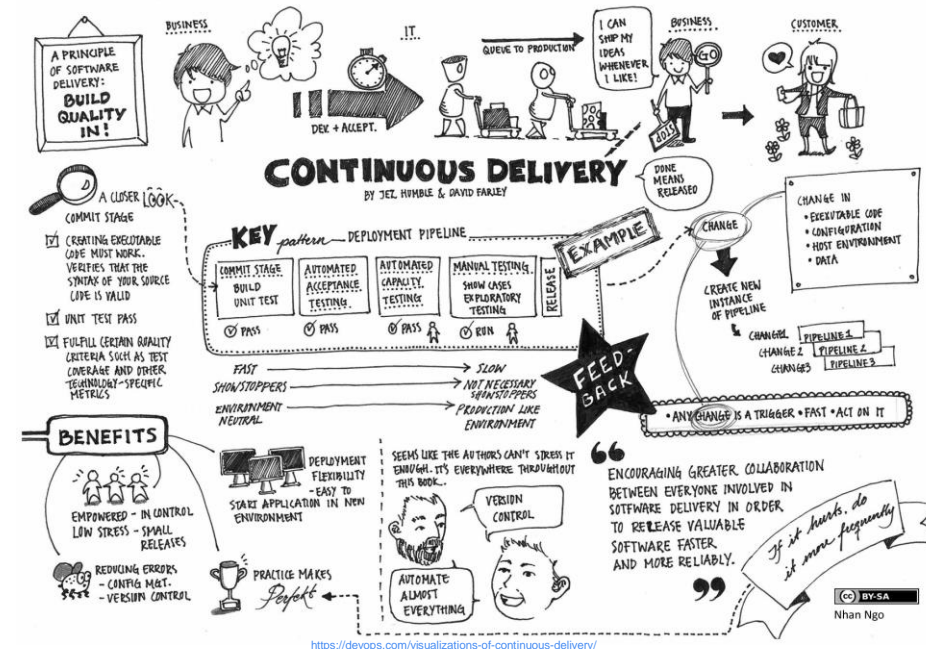
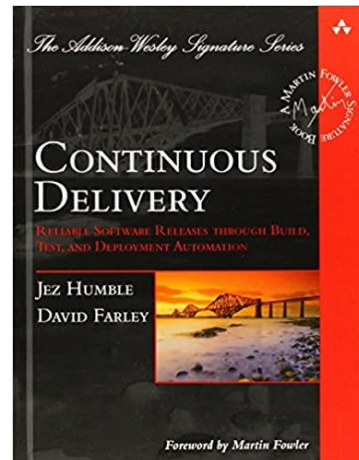
- ❑ Challenge and promise of DevOps
- ❑ DevOps history
- ❑ 3 Pillars of DevOps
 - Flow
 - Feedback
 - Continue learning
- ❑ DevOps Engineer – Version Control
- ❑ DevOps Engineer – Continuous Delivery

WHO AM I – 林秉毅

- ▣ Study and introduce new technology
- ▣ Provide platform services
 - BigData (Hadoop HDFS, HBase, MapReduce)
 - Continuous Delivery (Puppet)
 - Container (Cloud Foundry, Kubernetes)
- ▣ Contact: pylins@tsmc.com

Continuous Delivery

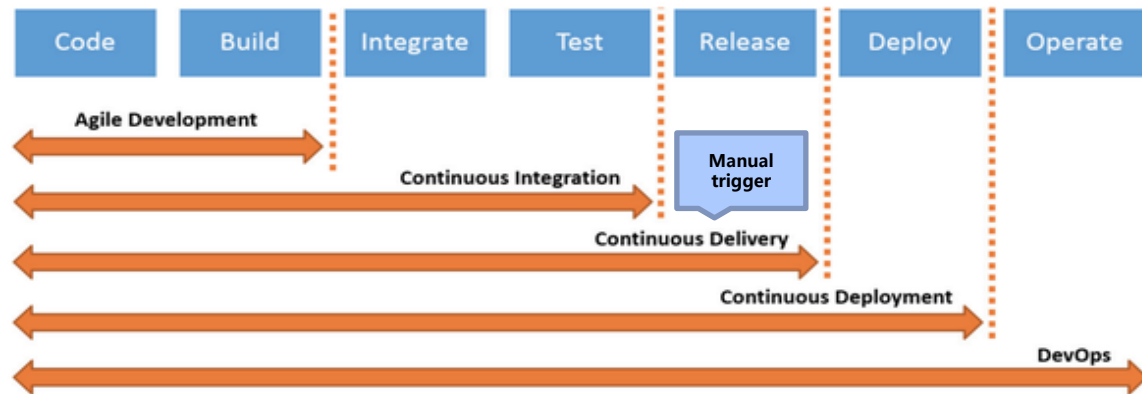
- 2010 – Book 《Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation》
- <https://continuousdelivery.com>



<https://devops.com/visualizations-of-continuous-delivery/>

What is Continuous Delivery

- Continuous Delivery is the ability to get changes of all types into production, or into the hands of users, *safely* and *quickly* in a *sustainable* way.
- Achieve all this by ensuring code is always in a *deployable* state.



NCTU+ 全校課程 根據排課 二手書 活動吧

2022, 網路程式設計 - 吳毅成
最後可步時間 2018-06-11 09:05

▼考試作業▼

這學期的分數比重:

- Homework0 - 5%
- Project1 - 21%
- Project2 - 19%
- Project3 - 23%
- Project4 - 13%
- Final Exam - 20%

很明顯就是這門課是作業loading很重的課
所以會有大量的時間在寫程式上
除了Homework0以外都是寫程式
更重要的是每個Project環環相扣
如果Project1沒寫出來後面可能就掛了
老師說要我們練習維護code的能力...

所以你很可能在寫Project3時還在小改前面的程式

Why Continuous Delivery

■ Benefits adopting Continuous Delivery

- Faster time to market
- Higher quality
- Better products
- Happier teams



We only pushed 1 line of code

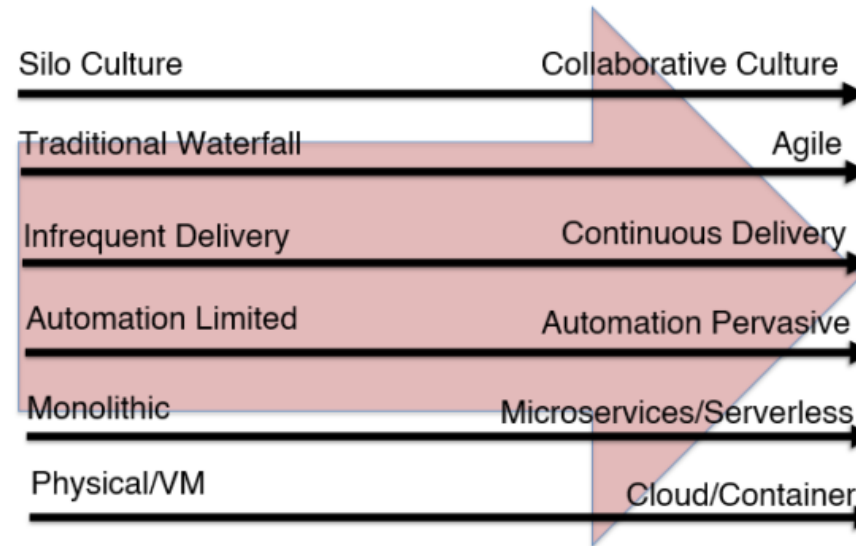
Top 10 programmer replies when their programs don't work

1. It works on my machine.
2. Why do you want to do it that way?
3. Somebody must've changed my code.
4. THIS can't be the source of THAT.
5. You must have the wrong version.
6. It must be a hardware problem.
7. What did you type in wrong to make it crash
8. It worked yesterday.
9. That's weird.
10. Even though it doesn't work, how does it feel?

Extend the development into production
And extend operations into development

Principles

- ❑ Automate almost everything
- ❑ Keep everything in version control
- ❑ If it hurts, do it more frequently, and bring the pain forward
- ❑ Build quality in
- ❑ Everyone is responsible
- ❑ Continuous Improvement



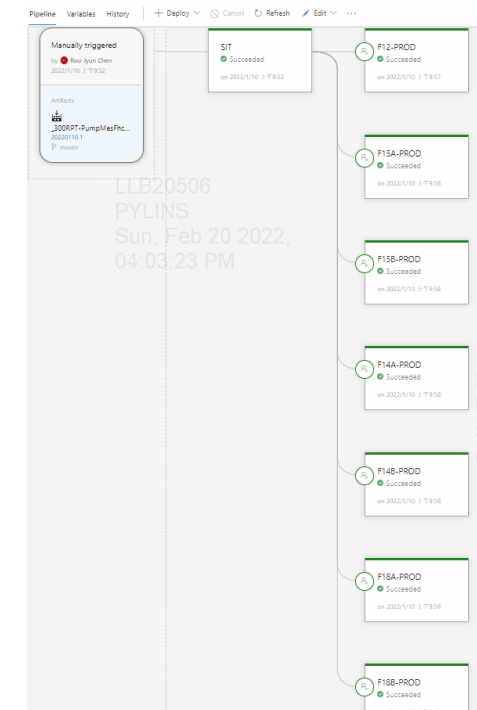
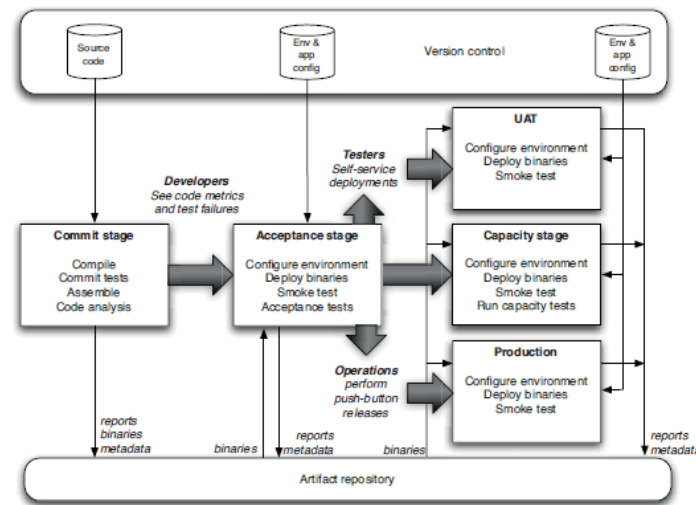
Foundations

■ Configuration Management

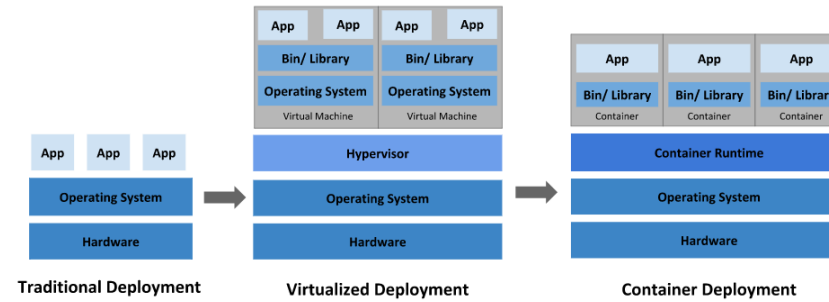
- Operating systems, patch levels, OS configuration, your application stack, its configuration, infrastructure configuration, and so forth should all be managed.
- Deploy the same way to every environment.
- Virtualization helps!

■ Continuous Integration

■ Continuous



Traditional vs Docker vs Kubernetes



Traditional



Build a house on your own

Docker



3D Print a house

Kubernetes



Apartment complex

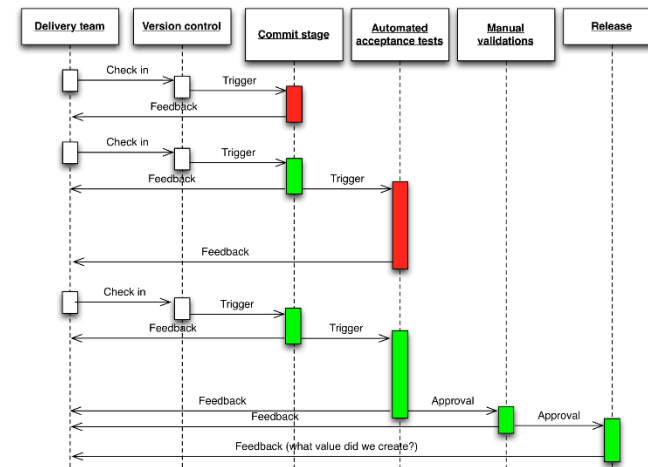
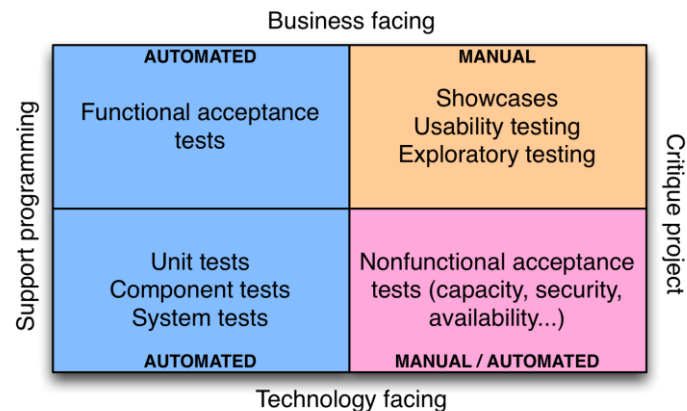
Foundations

□ Configuration Management

□ Continuous Integration

□ Continuous Testing

- Any plan that defers testing to the end of the project is broken because it removes the feedback loop that generates higher quality, higher productivity, and, most importantly of all, any measure of how complete the project is.



Tooling

□ Configuration Management

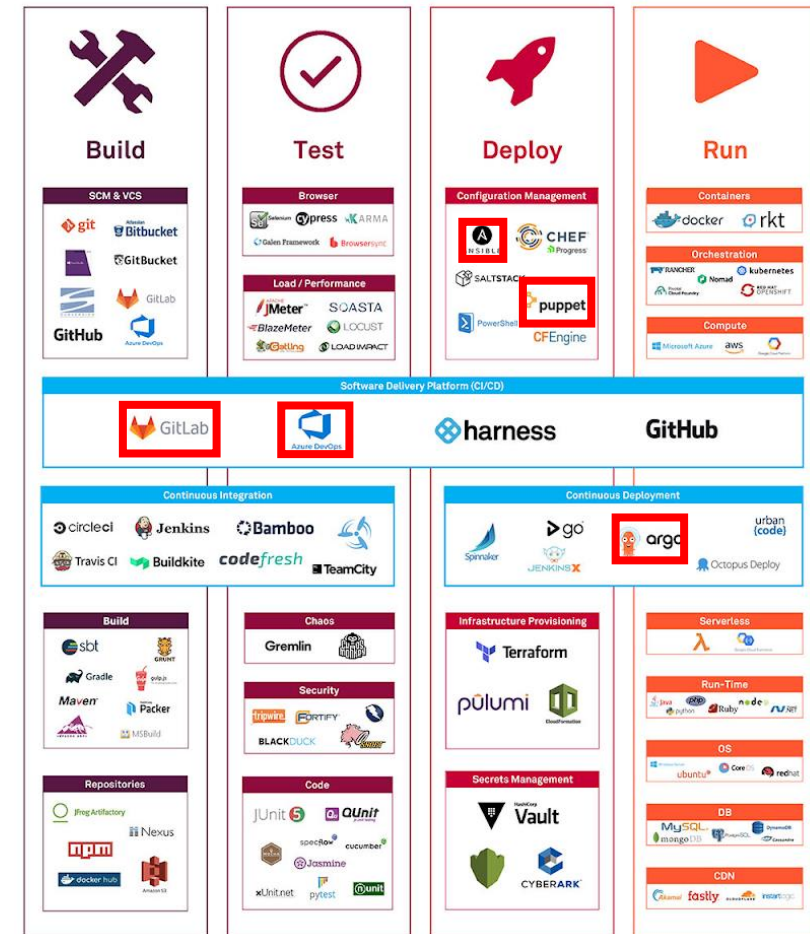
- Puppet
- Ansible

□ Continuous Delivery Platform

- Azure DevOps
- GitLab

□ Continuous Deployment

- ArgoCD



R.I.P. Configuration Management



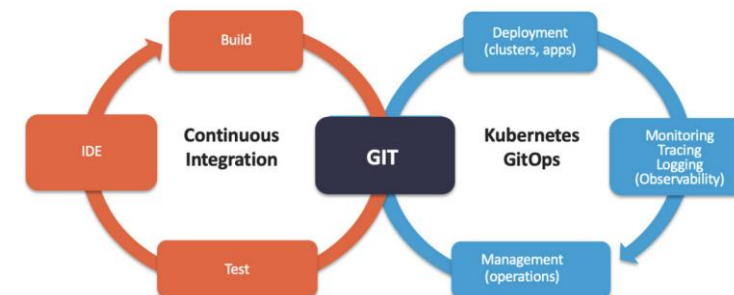
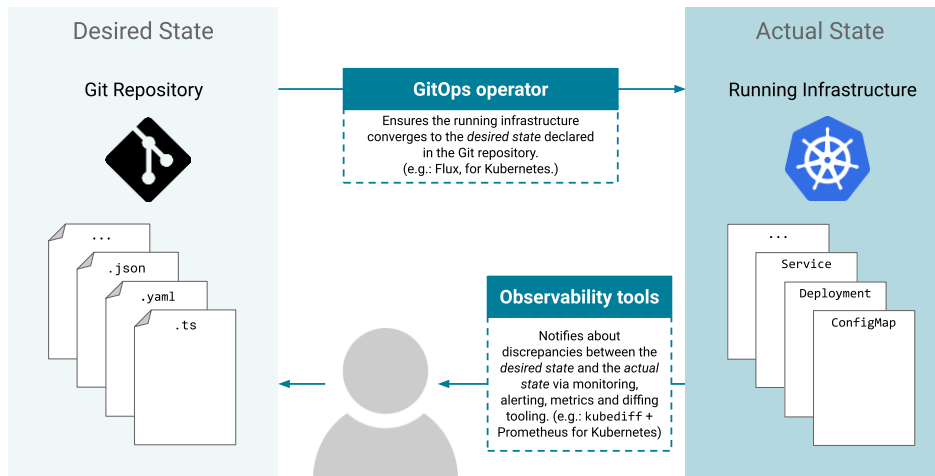
~~Welcome Configuration Management~~ ← 這樣不潮

Welcome GitOps ← 這樣比較潮比較好行銷



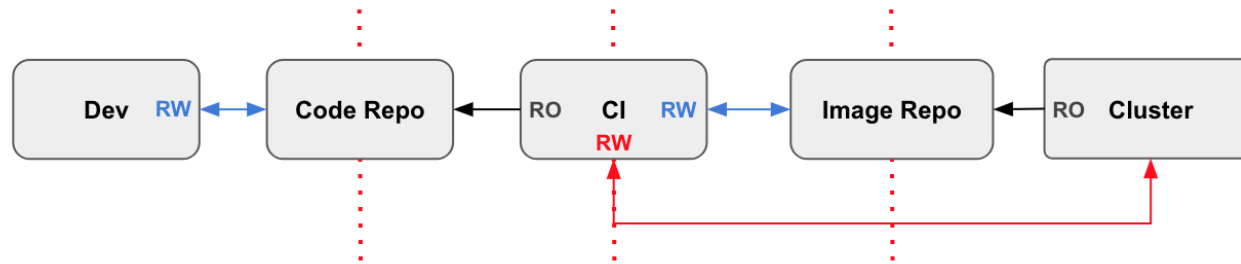
GitOps Continuous Delivery

- GitOps is a way to do Continuous Delivery introduced by Weaveworks in 2017. A **declarative** continuous delivery & operation concept that rely on Git as a source control system.
- Git is the **SSOT**(single source of truth) for the desired state of a system. (in yaml)
- Deployment changes become **traceable** with Git version control.
- Continuously **Diff & Sync** Git to live system.

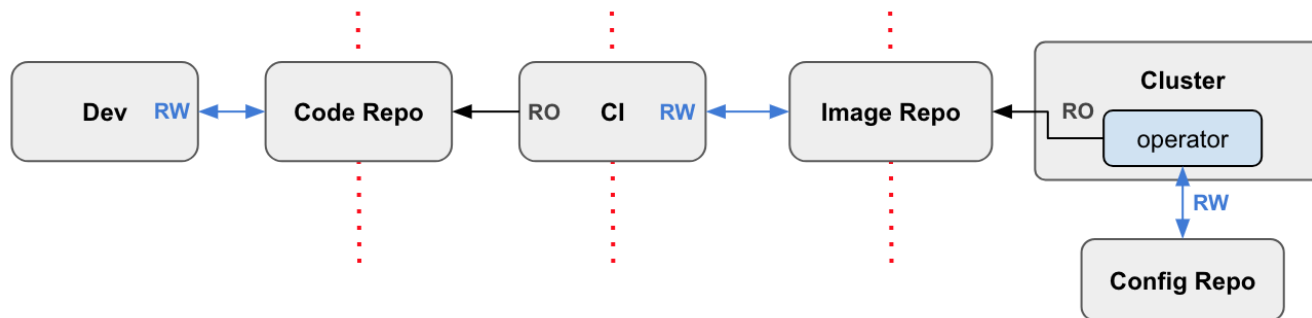


GitOps pipeline

□ Typical CI/CD pipeline

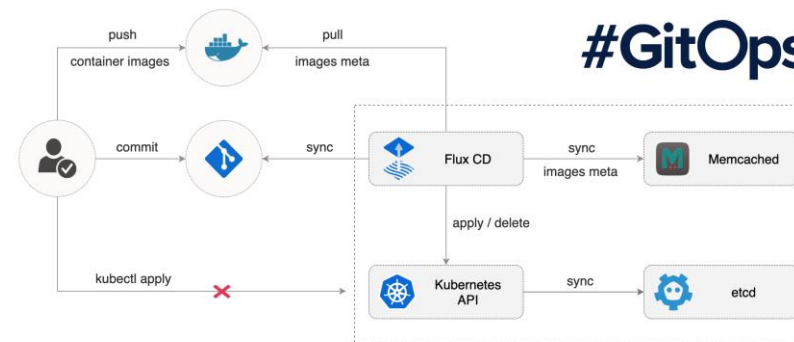
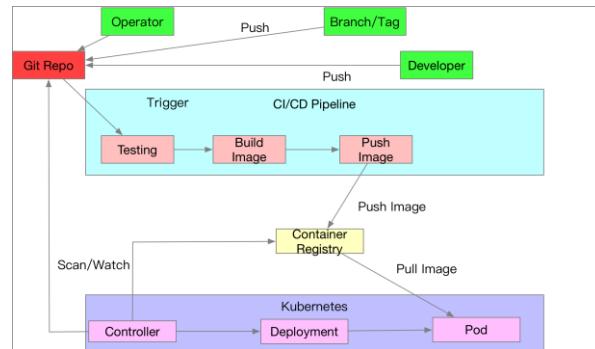


□ CI/CD pipeline



GitOps Details

- In the GitOps model, configuration changes go through a specific workflow.
 1. All configuration is stored in source control.
 2. A configuration change is made via pull request.
 3. The pull request is approved and merged into the production branch.
 4. Automated systems (e.g., a continuous integration pipeline) ensure the configuration of the production branch is in full sync with actual production systems.
- No human should ever directly apply configuration changes to a live cluster.



#GitOps