



Software Development Methods, Spring 2020

Jeffrey Liu
jeffffrey@gmail.com



Agenda

- Building an Agile Team
- Software Development Process
- DevOps Practices
- Engineering Culture



My journey in Software Development

- 2005~ 2007 : Enterprise Software, Developer
- 2007 ~ 2009 : IBM Emerging Technology Institute, Tech Lead
- 2009 ~ 2015 : IBM DataPower Appliance, incubation projects. Tech Lead
- 2015 ~ 2017 : Robot solution architect
- 2017 ~ 2020: Team Lead, Engineering Director at Appier



What is Agile ?

- Agile is the ability to create and respond to change. It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment.



Agile Manifesto

1 **Customer
Collaboration**
Over Contract Negotiation

3 **Responding
To Change**
Over Following a Plan



2 **Individuals
Interactions**
Over Processes and tools

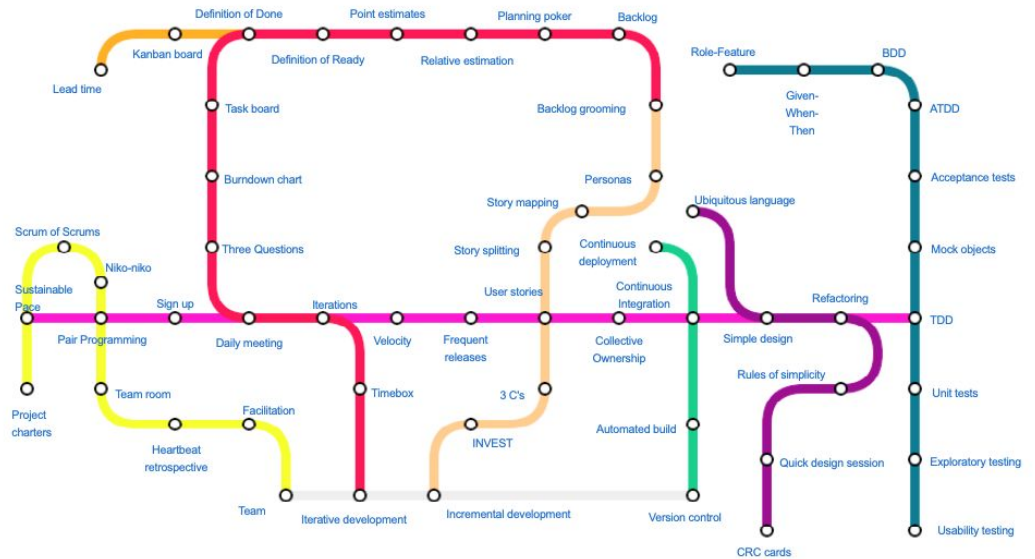
4 **Working
Software**
Over Full Documentation



Agile Software Development

- Agile software development is an umbrella term for a set of frameworks and **practices** based on the values and principles expressed in the Manifesto for Agile Software Development
- One thing that separates Agile from other approaches to software development is the **focus on the people doing the work and how they work together**. Solutions evolve through collaboration between self-organizing cross-functional teams utilizing the appropriate practices for their context.

Agile Subway Map

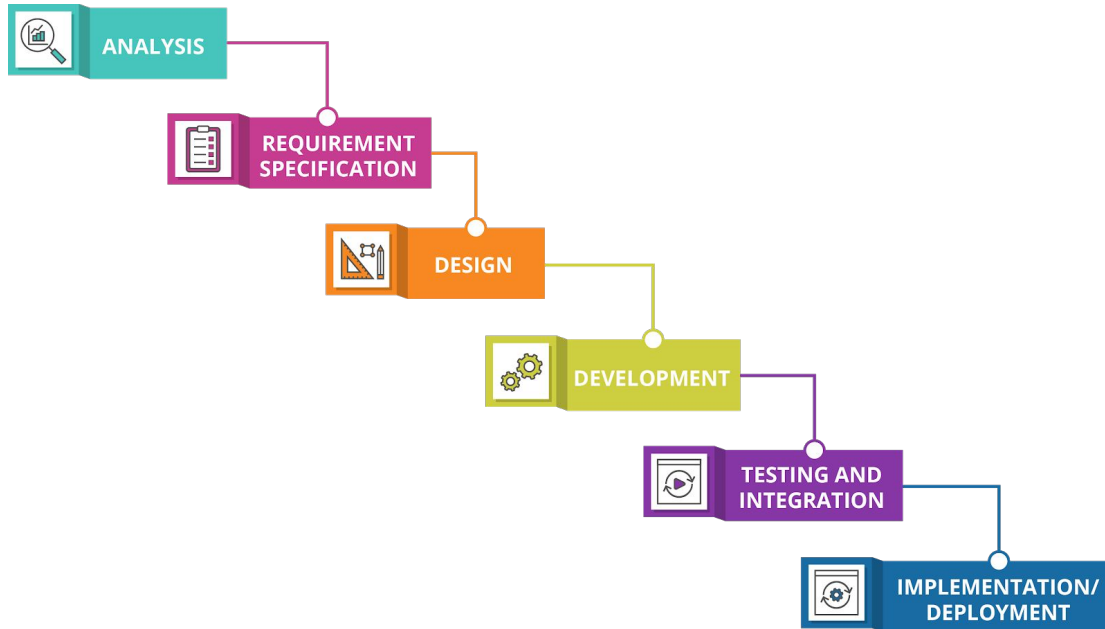


Lines represent practices from the various Agile "tribes" or areas of concern:



<https://www.agilealliance.org/agile101/subway-map-to-agile-practices/>

What is NOT Agile ?

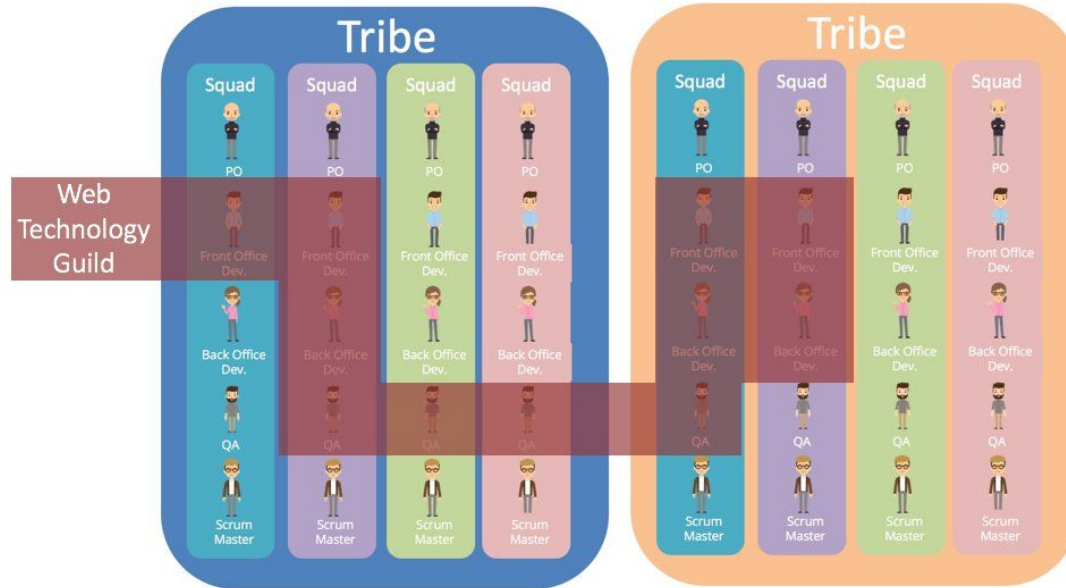




Agile Development Process

- The process used during software development life cycle. Each process contains a set of agile development practices (see the agile subway chart)
- Commonly used development process:
 - Scrum
 - Kanban
 - Scrumban

Agile Team Structure





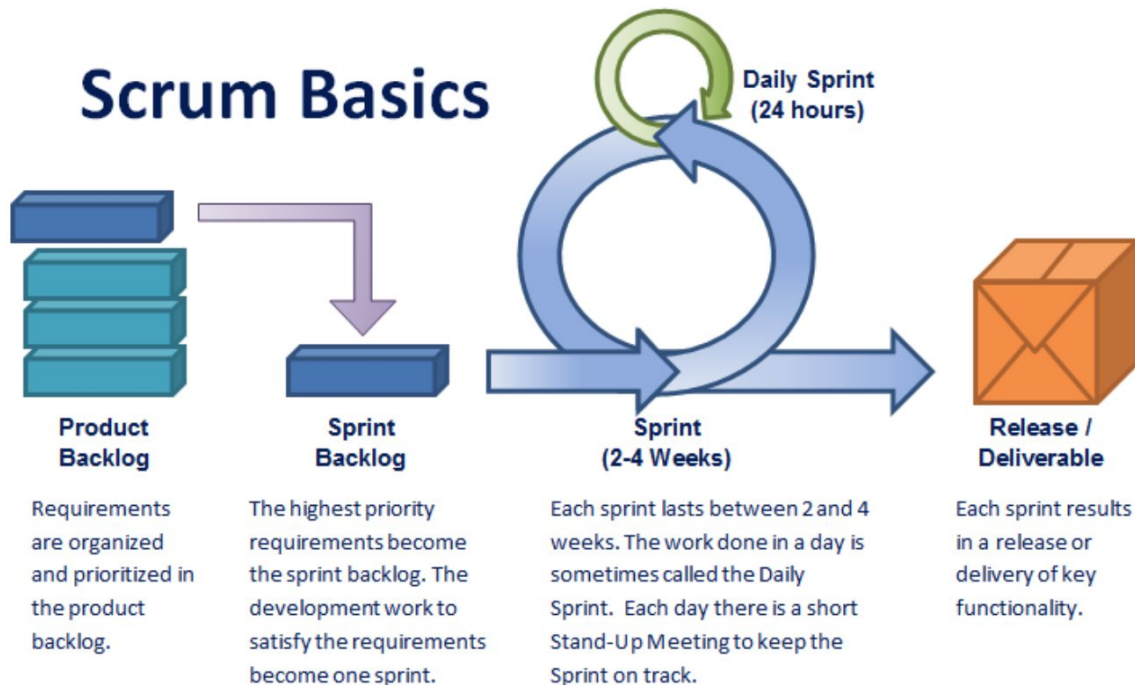
R&R in Agile Team

- Developer
- Product Owner
- Team Lead
- QA
- Agile Coach



Scrum in Action

Scrum Basics

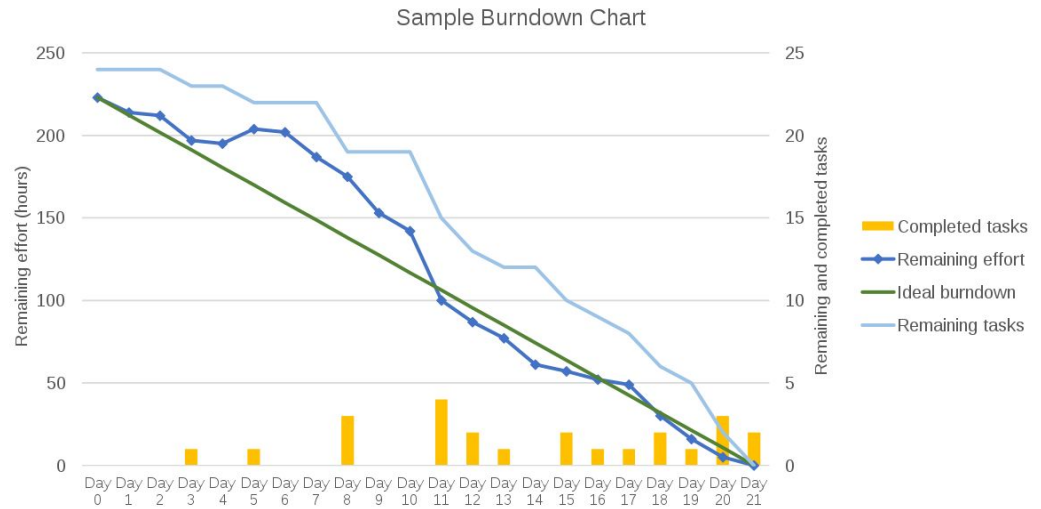




How we run Scrum

- Backlog Grooming
- Sprint Planning
- Daily Standby Meeting
- Sprint Review Meeting
- Sprint Retrospective Meeting
- Scrum of Scrum

Measure Scrum team performance with burndown chart





Kanban in Action



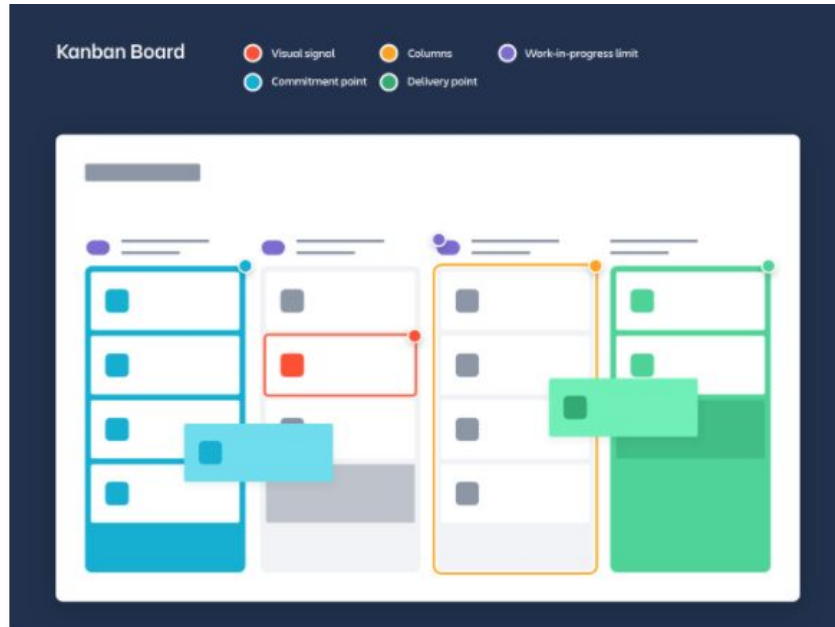
What is Kanban ?

- A kanban board is an agile project management tool designed to help visualize work, limit work-in-progress, and maximize efficiency (or flow).
- Kanban boards use cards, columns, and continuous improvement to help technology and service teams commit to the right amount of work, and get it done!

Original Toyota Kanban

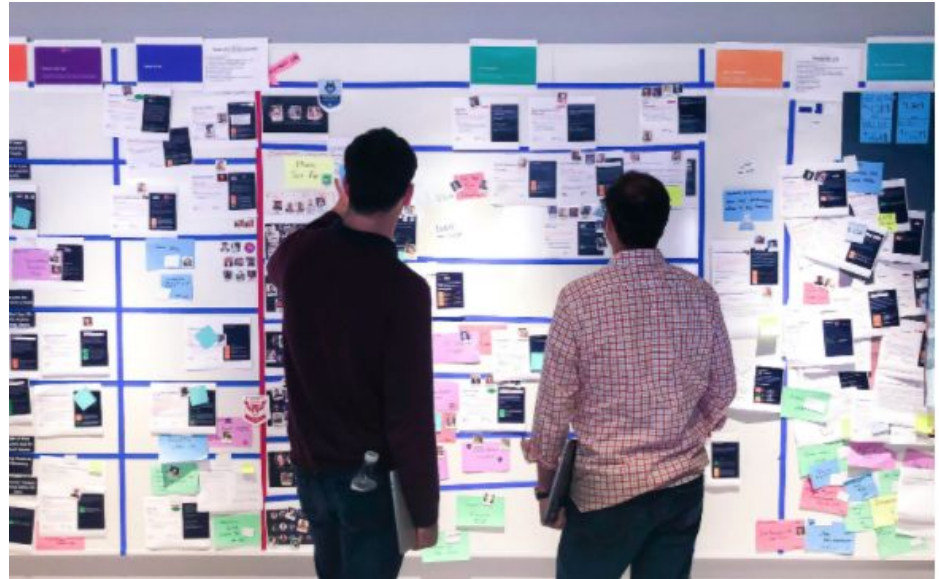


What is a digital Kanban ?



Key Elements in Kanban

1. Visual Signal
2. Columns
3. Work-In-Progress Limits





When to use Kanban

- Lightweight development projects
- Tasks can be broken down into small and concrete pieces
 - E.g. bugs, infra tasks
- Focus on flow and throughput



Scrumban in Action

Introducing Scrumban






6 Key Steps for running Scrumban



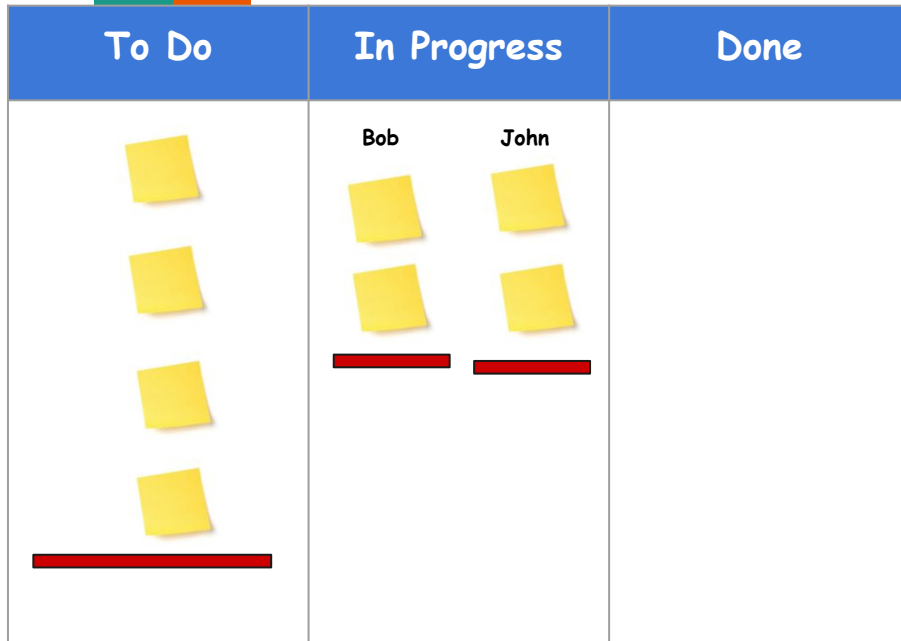
1. Visualize the task pipeline
2. Impose WIP Limit
3. Add more column as buffer
4. Global priority and late binding
5. Consistent task breakdown and estimate with cycletime
6. Team grooming with fixed interval

1. Visualize The Task Pipeline

Every team should keep an up-to-date kanban, so it is clear to everyone who is interested in the overall progress

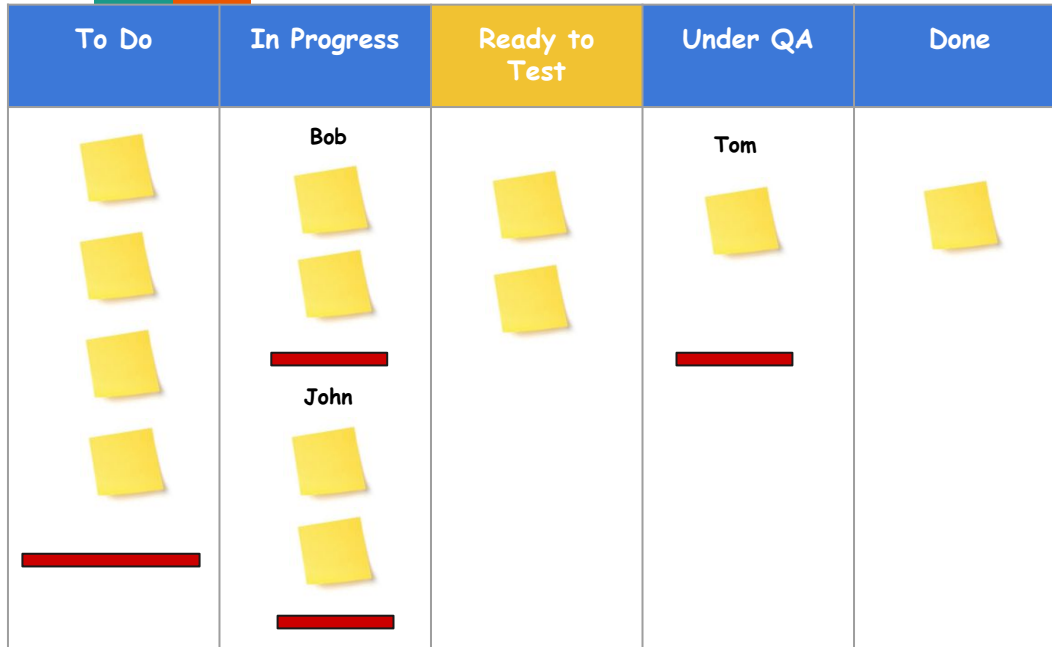
To Do	In Progress	Done
		

2. Impose Work In Progress Limit



- Individual Level Limit (2)
- Team Level Limit

3. Add more columns as buffer


















Use Pull mode for both development and test team

- Dev pull from “todo”
- Test pull from “ready-to-test”
- DevOps pull from “ready-to-release”



4. Global priority and late binding

To Do	In Progress	Ready to Test	Under QA	Done
    	<p>Bob</p>    <p>John</p>   	 	<p>Tom</p>  	

Last Resource Binding

- Task could be implemented by other developer of the same component group

Show Global Priority Ordering

- Tasks are sorted by priority, so everyone can pick task from the list when extra bandwidth available

5. Use task breakdown and estimate with cycletime

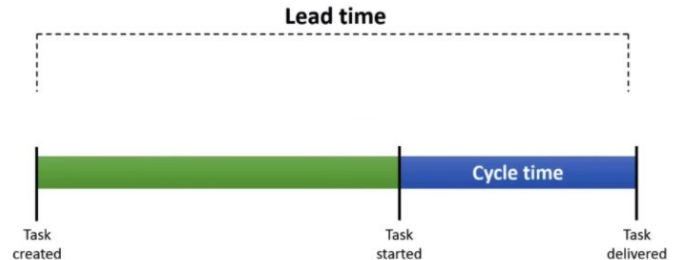
Instead of using story points, we estimate ETA with the number of tasks and the average cycletime

$$\text{Effort} = \text{number_of_sub_task} * \text{average_cycletime}$$

Our assumptions:

1. Each team will breakdown task with similar granularity
2. Cycletime will become stable over time

$$\text{Cycletime} = \text{time}(\text{"In-Progress"}) + \text{time}(\text{"Code Complete"})$$



6. Team grooming with fixed interval



To Do	In Progress	Ready to Test	Under QA	Done
 	Bob 		Tom 	

- Run bi-weekly global grooming
- Each team will run grooming meeting within 2 working days after global grooming.
 - AC change status from backlog to “todo” when task are selected for dev
 - AC provide ETA estimation to PM

Key ceremonies for Scrumban



- Grooming (regular & on demand)
- Daily Standup
- Bi-Weekly Retrospective

Comparison of Agile Development Process



1. Use scrum for the feature development of more stable product, where predictability and commitment is the key
2. Use kanban for small and concrete tasks. Throughput is more important than schedule
3. Use scrumban when task diversity is high and development resource are relatively unstable. More flexibility is needed.

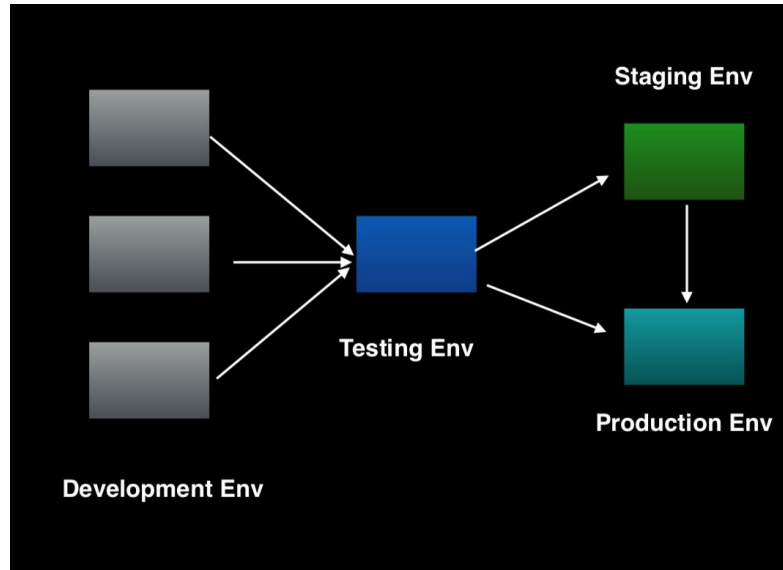


DevOps



Modern Software Development Environments

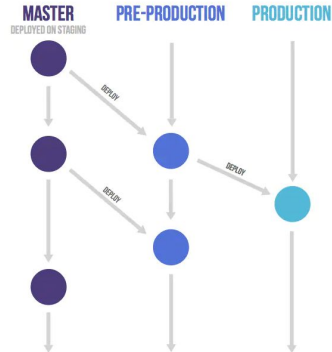
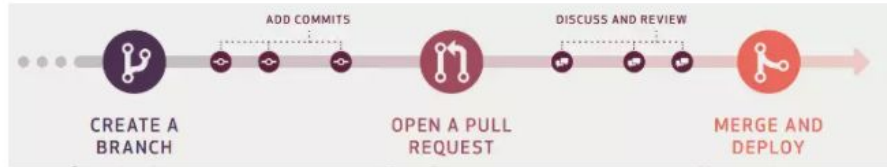
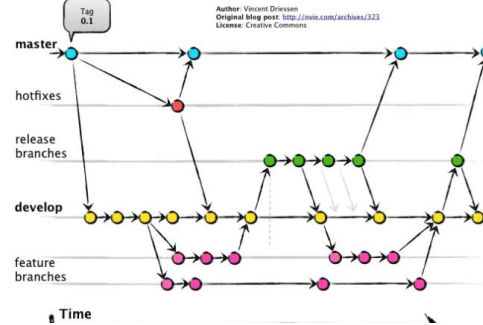
- Production
- Staging
- Testing
- Development



GIT Strategy

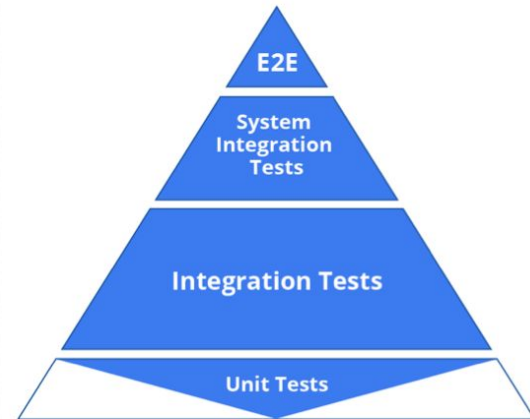
- Git Flow
- Github Flow
- Gitlab Flow
- Trunk-Based

A successful Git branching model



Testing Pyramid

- Unit tests are narrow in scope and typically verify the behaviour of individual methods or functions.
- Integration tests make sure that multiple components behave correctly together. This can involve several classes as well as testing the integration with other services.
- Acceptance tests are similar to the integration tests but they focus on the business context rather than the components themselves.
- UI tests will make sure that the application functions correctly from a user perspective
- E2E test ensure key production features remain un-changed.

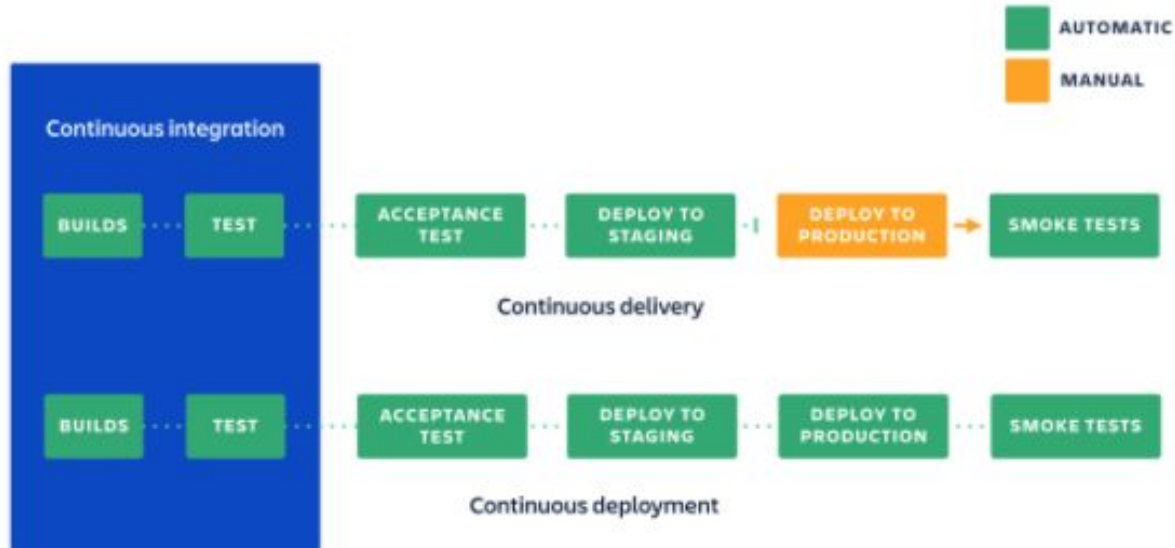




CI / CD

- Continuous integration (CI) is a practice where a team of developers integrate their code early and often to the main branch or code repository. The goal is to reduce the risk of seeing “integration hell” by waiting for the end of a project or a sprint to merge the work of all developers.
- **Continuous delivery** is an extension of continuous integration to make sure that you can release new changes to your customers quickly in a sustainable way. This means that on top of having automated your testing, you also have automated your release process and you can deploy your application at any point of time by clicking on a button.

Continuous Delivery v.s. Continuous Deployment





CI In Action

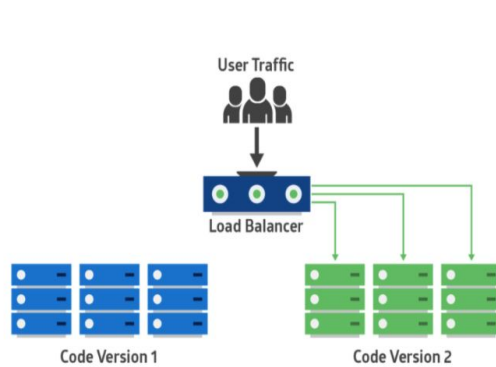
1. Start writing tests for the critical parts of your codebase.
2. Get a CI service to run those tests automatically on every push to the main repository.
3. Make sure that your team integrates their changes everyday.
4. Fix the build as soon as it's broken.
5. Write tests for every new story that you implement.

<https://www.atlassian.com/continuous-delivery/continuous-integration/how-to-get-to-continuous-integration>

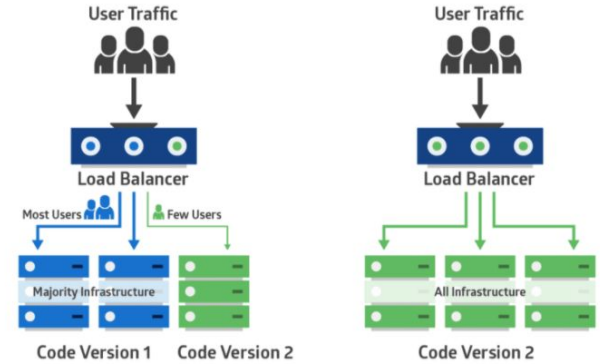
CD In Action - Deployment Strategy



Rolling



Blue / Green



Canary Release

<https://dev.to/mostlyjason/intro-to-deployment-strategies-blue-green-canary-and-more-3a3>

Monitoring

- Monitors are equally important as testcases
- It allow us to respond system issues efficiently, thus improve customer satisfacation
- MUST HAVE for critical system features





Site Reliability Engineering

- Improve system observability with logs and alerts
- Measure system reliability with SLI and monitor
- Define incident handling SOP
- Various practices to ensure system reliability
 - e.g. Error Budgets



Engineering Culture

Mutual Trust

- Mutual Trust with your team
- Trust Between Peers
- Trust with team leads and manager



Find a Pacer

- Stabilize, and then accelerate
- Flying-geese model
-



Open-Minded Debate

We want to avoid group-think

And We agree to disagree



Feedback

- Peer Feedback
- 360 Feedback
- 1x1 Performance Feedback
- Quarterly Performance Feedback





Principles

- We document the best practices and consensus
- These principles are PR reviewed and stored in GIT



Postmortem & Retrospective

- Learn from mistakes. Don't make it twice
- Keep up the good works