Headline Supporter

myob

Passionate Supporter

Tabar
tabar.com.au

Supporters

SWINBURNE

460degrees

REA Group

ZXM

Backers

Sprint Agile

elabor8

Networking event

DIUS

SPARK MELBOURNE

LAST Conference Anywhere

Introduction to CI/CD

Steve Mactaggart
State Manager - Victoria

cevo

Agenda

- What is CI and CD
- What makes good CI / CD
- How we get there

- Business benefits of it all….

What is CI and CD?

- Continuous Integration
  - An approach to be continually validating the state of a codebase through automated testing.
  - Best achieved through integration with version control
- Continuous Delivery / Deployment
  - An approach to regularly deploying artifacts that successfully pass the CI phase to ensure confidence around the deployment

Delivery vs Deployment

Continuous integration, **continuous deployment**, and **continuous delivery** are like vectors that have the same direction, but different magnitude.

Their goal is the same: make our software development and release process faster and more robust.

The key difference between the three is in the scope of automation applied.

Delivery vs Deployment

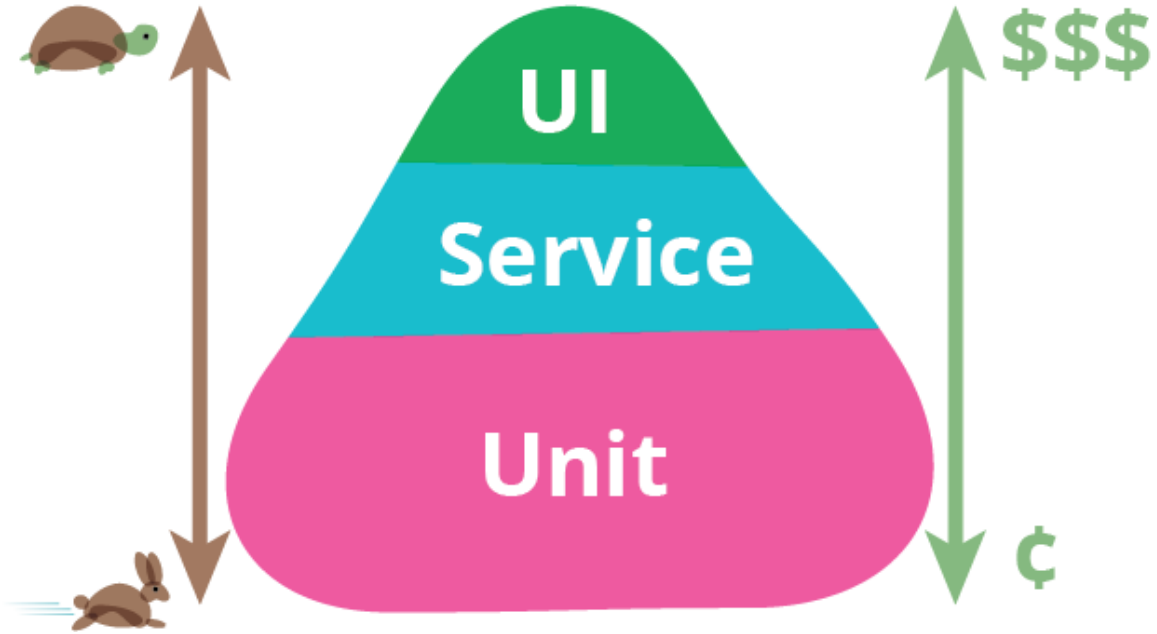
- Continuous Delivery
  - Automatically prepare and track a release to production
  - The desired outcome is that anyone with sufficient privileges to deploy a new release can do so at any time in one or a few clicks. By eliminating nearly all manual tasks, developers become more productive.
- Continuous Deployment
  - Every change in the source code is deployed to production automatically, without explicit approval from a developer.
  - As long as it passes the quality controls

What does good look like?

What makes for good CI?

1. Decoupled stages
    - Each step in CI should do a single focused task
2. Repeatable
    - Automated in a way that is consistently repeatable
    - Tooling should work for local developers too – Local/Remote parity
3. Fail fast
    - Fail at the first sign of trouble

The test pyramid

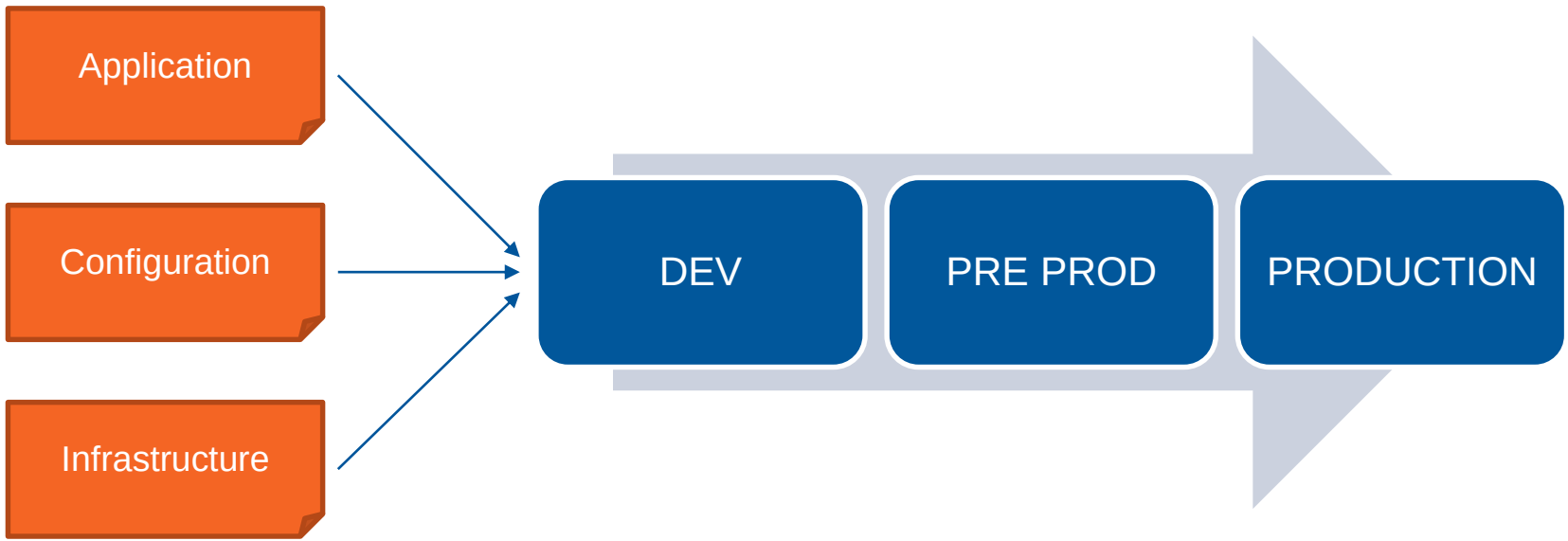


https://martinfowler.com/bliki/TestPyramid.html

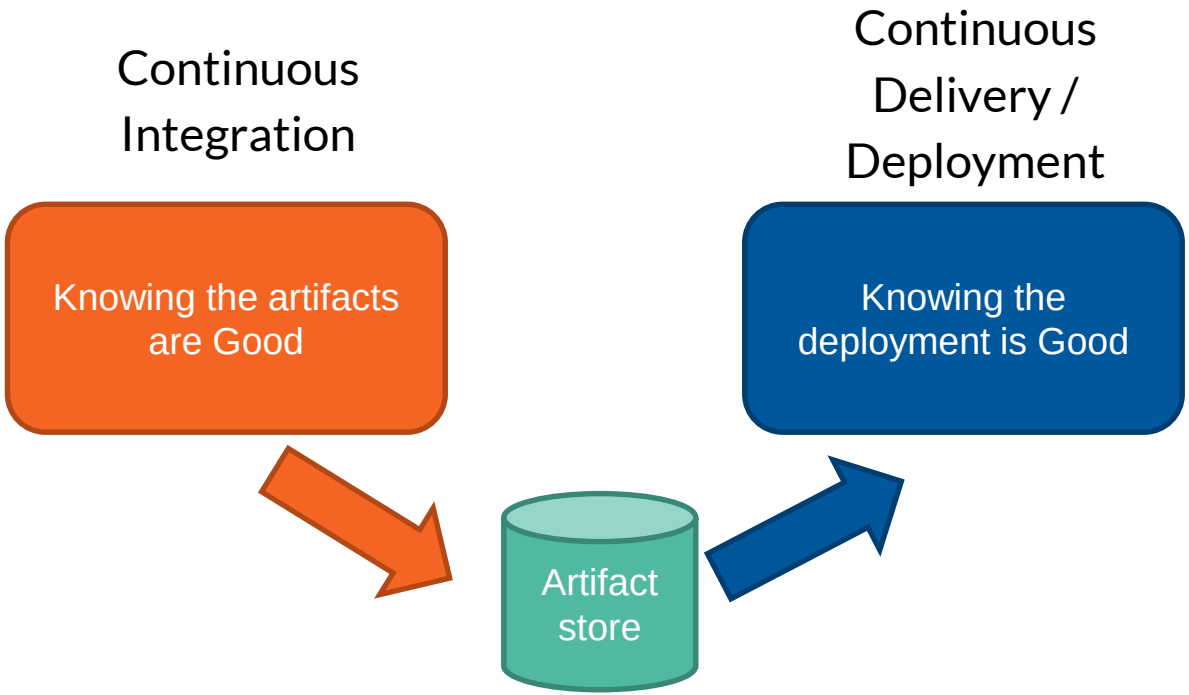
What makes for good CD?

1. Design with the system in mind
    - Cover as many parts of a deployment as possible
    - Application | Infrastructure | Configuration | Data
2. Pipelines
    - Continually increase confidence as you move towards production
3. Globally unique versions
    - Know the state of the system at any time
    - Be able to demonstrate difference between current and future state

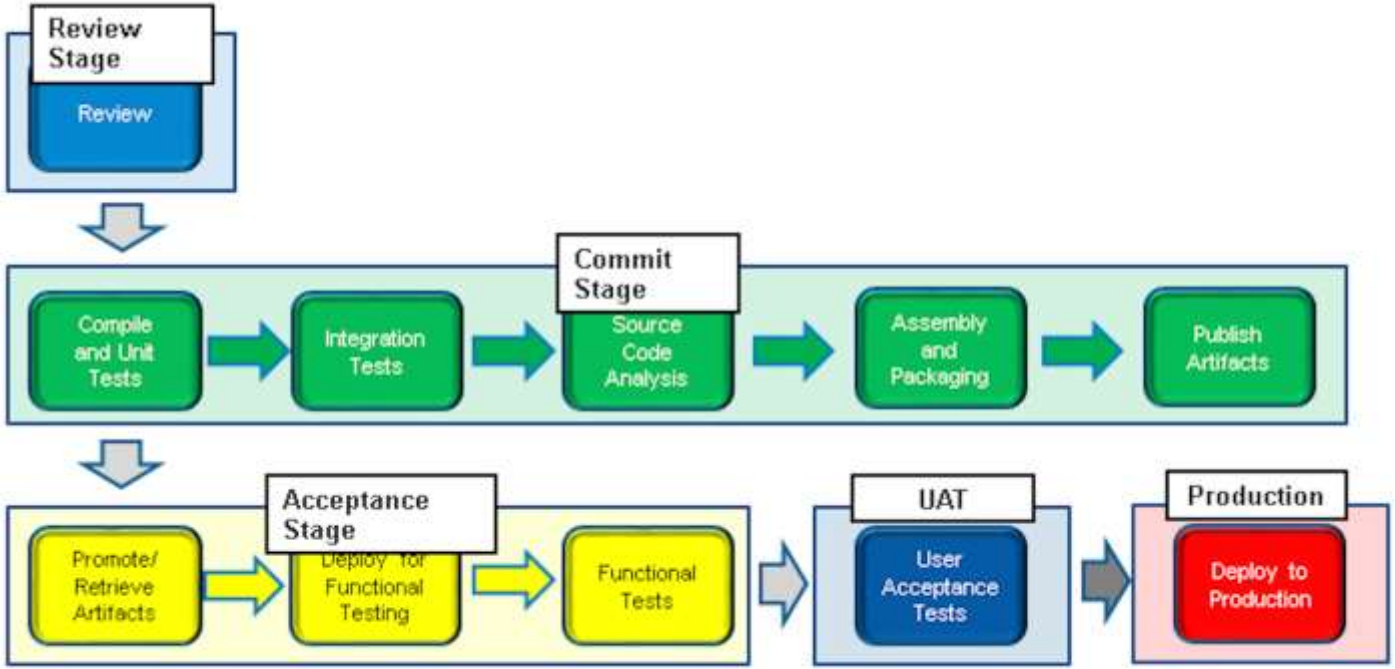
CD Pipeline flow



Bringing CI & CD together



Full deployment pipeline



https://www.oreilly.com/learning/configuring-a-continuous-delivery-pipeline-in-jenkins

Tracking Confidence

Where do we start?
How do we know we are progressing?

CI/CD Pipeline Maturity Metrics

Confidence Map DD/MM/YYYY

Application	Owner or custodian	Development					Validation			Deployment				
		Pipeline as code	PR/ branch builder	Fail Fast	Unit tests	Artifacts into central repository	Automated functional tests	Fully resolved artifacts	Automated PVT	Infra' as code	Automated deployment	Environment configuration management	Automated data migration	Automated fallback
		🕒 💰 ⚠ 🚀	🕒 🗨 🚀	🗨 🕒 💰	🕒 📨 ⚠	🕒 ⚠ 🚀	🕒 📨 ⚠	⚠	🕒 📨 ⚠	🕒 📨 ⚠	💰 📨 ⚠	🕒 📨 ⚠	🚀 🕒 📨 ⚠	⚠ 💰 🕒
Application A	Team A	✔	✔	✔	✔	✔	🚀	✔	✘	✔	✔	✔	✔	✘
Application B	Team A	✘	✘	✘	✘	✔	?	?	✘	✔	✘	✘	🚀	✘
Service X	Team A	✘	✘	?	?	✔	✘	?	✘	✘	✘	✘	✘	✘
Service Y	Team B	✘	✘	?	?	?	✘	?	✘	✘	✘	✘	✘	✘
Service Z	Team B	✘	✘	?	?	?	✘	?	✘	✘	✘	✘	✘	✘
Application C	Team C	✘	✘	?	?	?	✘	?	✘	✘	✘	✘	✘	✘
Application D	Team C	✘	✘	?	?	?	✘	?	✘	✘	✘	✘	✘	✘

🕒 Reduces cycle time
💰 Controls costs
📨 Validates business requirements
⚠ Controls risk
🗨 Enables innovation
🚀 Enables rapid starts

CI/CD Pipeline Maturity Metrics

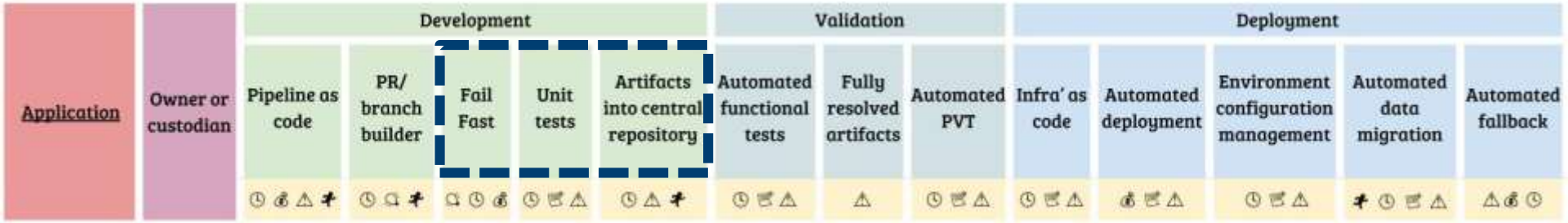
Confidence Map DD/MM/YYYY

Application	Owner or custodian	Development					Validation			Deployment				
		Pipeline as code	PR/ branch builder	Fail Fast	Unit tests	Artifacts into central repository	Automated functional tests	Fully resolved artifacts	Automated PVT	Infra' as code	Automated deployment	Environment configuration management	Automated data migration	Automated fallback

1. Assess the current state
2. WSJF
3. Think globally – act locally
    - Proof of Concept and Spikes are your friends

CI/CD Pipeline Maturity Metrics

Confidence Map DD/MM/YYYY



1. Assess the current state
2. WSJF
3. Think globally – act locally
    - Proof of Concept and Spikes are your friends

Unit testing and shifting left

- Make the tests easy to run
  - Run them locally
    - Invest in good quality IDE support
  - Consistent entrypoint - Abstract the complexity
    - Makefiles / Gradle
  - Pull request unit test execution

Failing fast

- Code inspection
  - Code linting
  - Security inspection
  - Code format
- Code coverage
  - Track coverage changes

Centralised artifacts

- Managing output from a CI process
  - Only the strong survive
  - Central common location
  - Clearly understood name and version approach
  - Build them only once
  - Make them timeless – externalise configuration

Business benefits of CI/CD

Why do this?

- Reduction of delivery risk
  - No longer do we need to rely on humans with specific knowledge as the gate-keepers of quality
  - Reduced chance of humans not following the process
  - Reduced chance of mis-communication on executing the change

Why do this?

- To encode the process, we need to know the process
  - If we know all the tests pass,
  - If we know all the steps in deployment,
  - What is stopping us from releasing?

Why do this?

- Better visibility on change
  - As our systems and tools are version controlled
  - And we know what the current state of production is
  - And we can describe the process by which it will be changed
  - We can diff the system states with confidence
- Opens up more avenues for review and increased audit compliance

Why do this?

- Increased efficiency and delivery options
  - Enables us to deliver things with reduced effort
  - This leads us to deploy change more frequently
  - Which leads to getting feedback faster
  - That enables us to experiment easier
  - This leads to smaller batch sizes
  - Which leads to and increased flow of the entire system

Why do this?

- Enhanced learning from failure
  - When we have an issue or failure, we write a test to cover it
  - This test gets added to our suite and executed every time
  - Decreases our risk of this issue occurring again

That’s all folks…

Steve Mactaggart
State Manager – Victoria

steve.mactaggart@cevo.com.au