Intro to Continuous Delivery and DevOps

From a testing perspective

Lisa Crispin

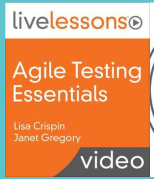
With material from Abby Bangser, Ashley Hunsberger, Lisi Hocke & more

It takes a village…

@lisacrispin

A little about me...

Agile Testing
A Practical Guide for Testers and Agile Teams
Lisa Crispin
Janet Gregory
More Agile Testing
Learning Journeys for the Whole Team
Janet Gregory
Lisa Crispin



AGILE TESTING FELLOWSHIP

Testing advocate with

mabl

@lisacrispin

Learning intentions

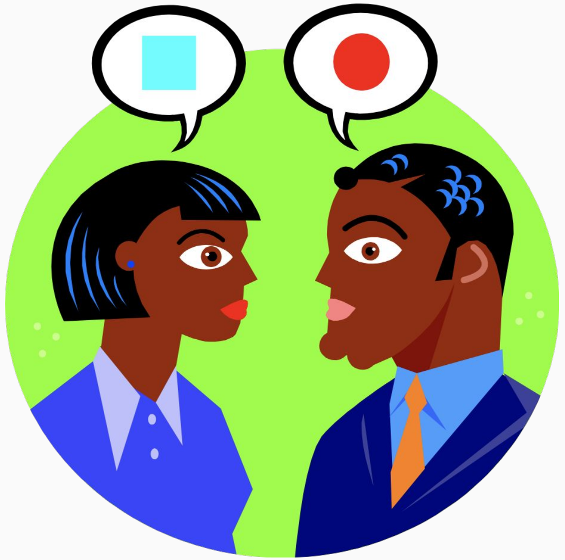
- Ways to engage the whole team in a DevOps (DevTestOps) culture
- Some tools to help shorten feedback loops & mitigate risks
- How to fit all necessary testing activities into the continuous world

Expectations

- There are no “best practices”, there are “leading practices”.
- Read more, and experiment with these ideas to really learn them.
- DevOps is a big area. Today I’ll focus on terminology and deploy pipelines.

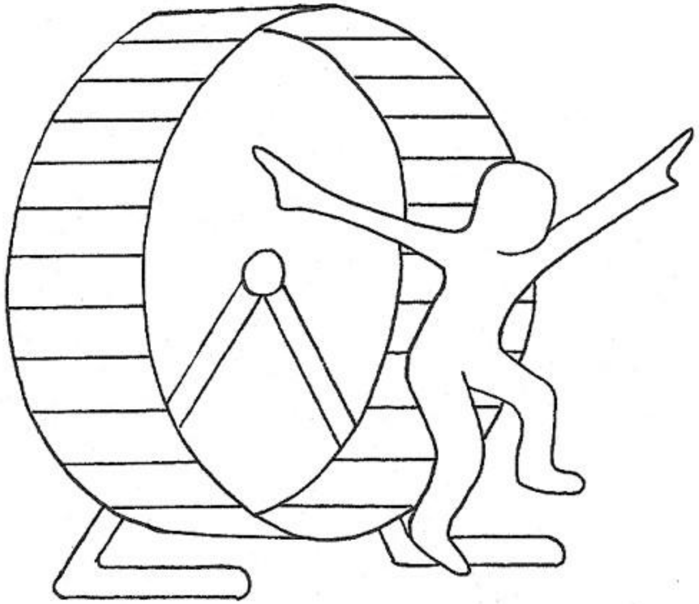
What do you think when you hear “DevOps”?

How about “Continuous Delivery”?



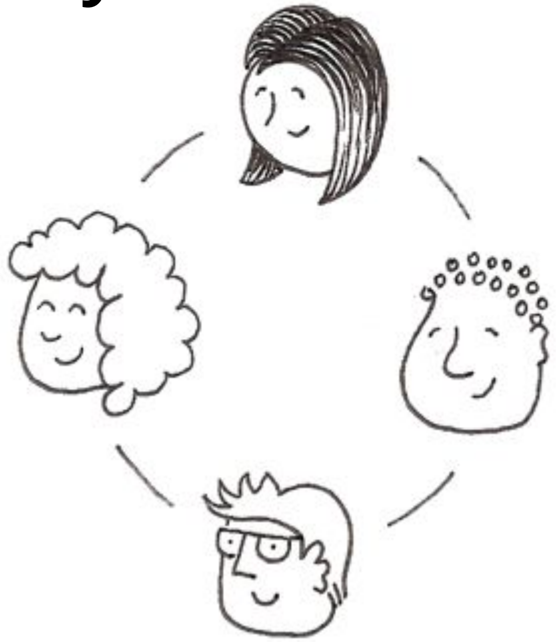
@lisacrispin

What do you think when you hear “Continuous testing”?

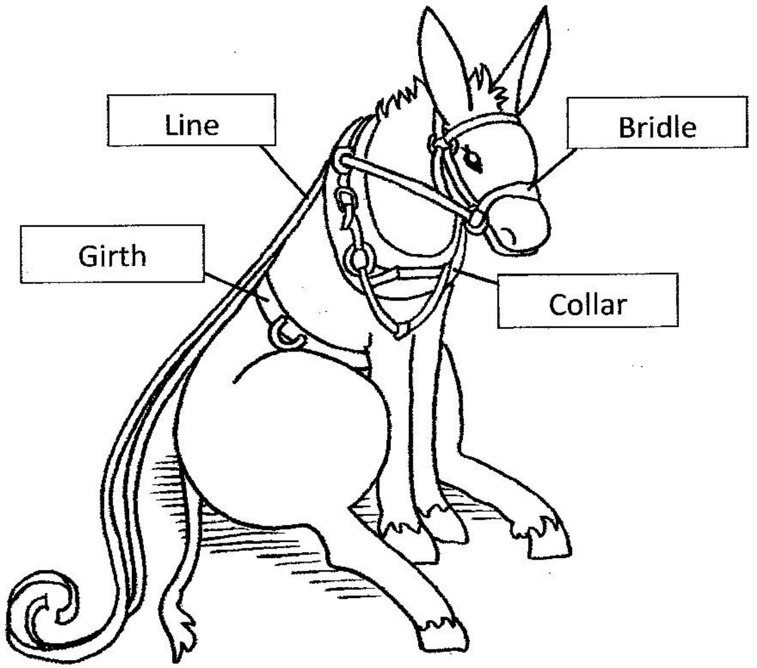


Whole team responsibility for quality

- Commitment to a level of confidence
  - Bug prevention over bug detection
  - Learning from production use, errors
  - ...and responding fast
  - Focus on what's valuable to customers
- Diverse perspectives, skill sets, biases



Let’s agree on some common terminology



@lisacrispin

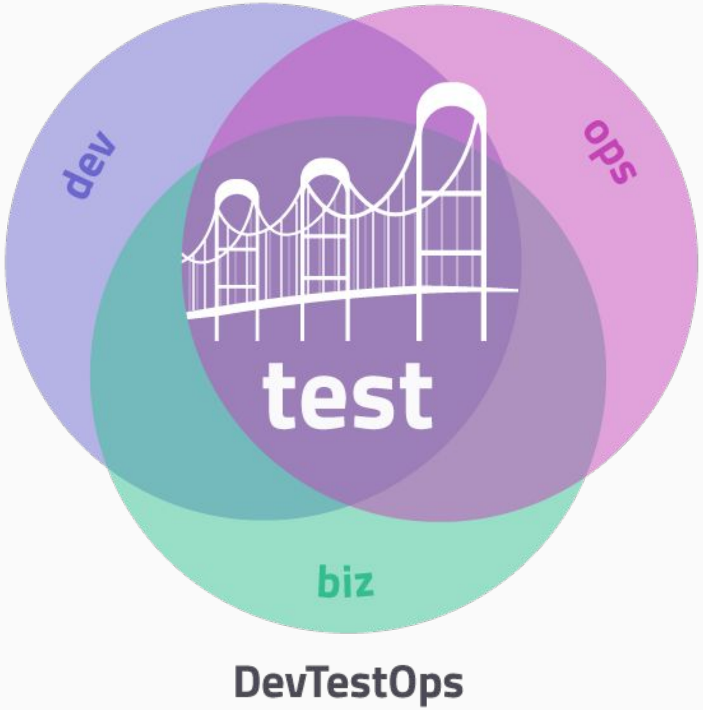
“DevOps”

- Term coined in 2009, but the concept goes back to early days
- Devs, testers, ops, others collaborate
  - Create, test, maintain infrastructure for CI, deployments, test & prod environments
  - Support continuous delivery & testing
  - Make our customers’ day a bit better

It’s all about:

- Collaboration
- Continuous improvement
- Continuous learning

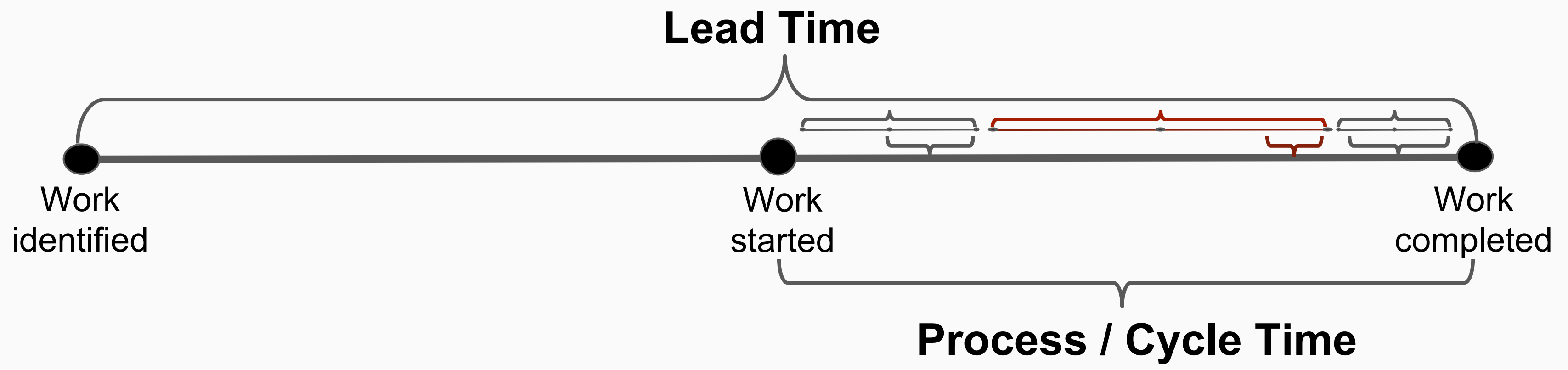
Testing is the bridge between development, operations and the business stakeholders - the heart of DevOps



Measuring our flow of work

Cycle time: how long from start to delivery?

- Re-work slows us down
- Shared understanding speeds us up

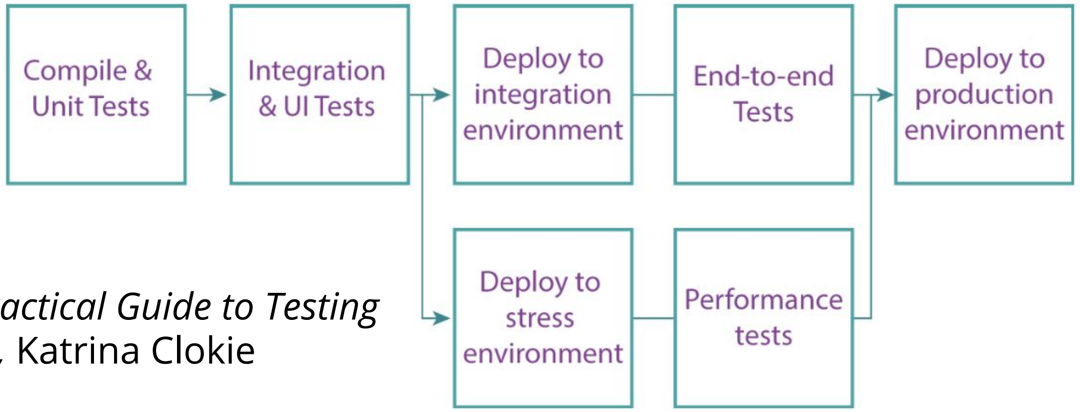


Continuous Integration

- Integrate code into a shared repository multiple times per day
- Preferably on trunk/master, but on branches too
- Typically the start of a pipeline
- Each check-in can be verified by an automated build with automated regression tests

Deployment pipeline

- Break the build into stages to speed up feedback
- Each stage takes extra time & provides more confidence
- Early stages can find most problems -> faster feedback
- Later stages probe more thoroughly
- Automated deployment pipelines are central to continuous delivery


Compile & Unit Tests
Integration & UI Tests
Deploy to integration environment
End-to-end Tests
Deploy to production environment
Deploy to stress environment
Performance tests

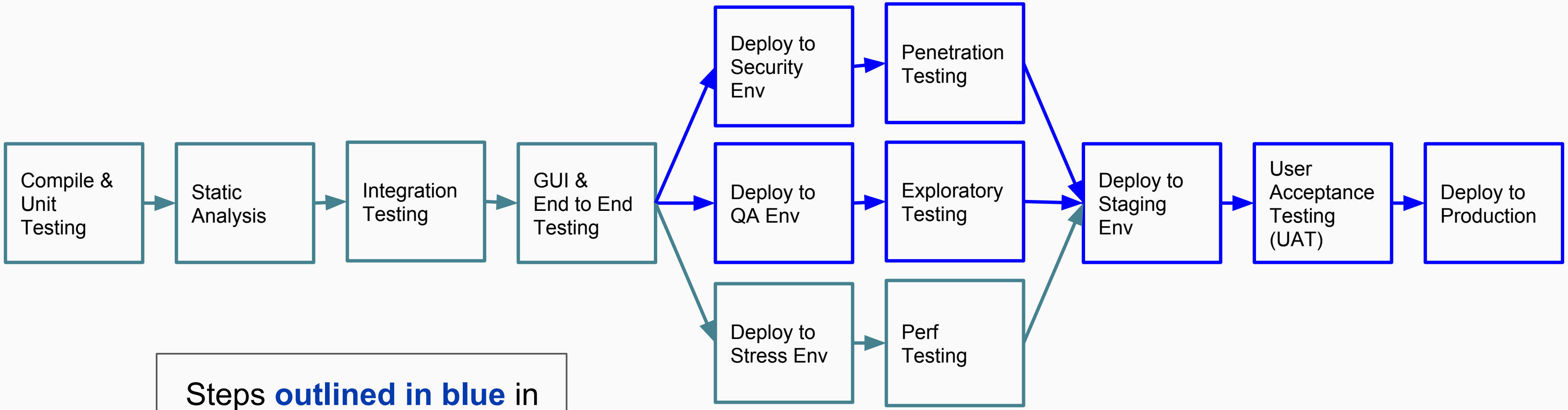
From *A Practical Guide to Testing in DevOps*, Katrina Clokie

@lisacrispin

Continuous Delivery (CD)

- Ability to get many types of changes into production safely, quickly and sustainably (Jez Humble)
    - eg. new features, configuration changes, bug fixes, experiments
- Heavily benefits from, but not dependent on, automated regression tests
- Each commit is independently verified as a deployable release candidate
- A deployable release candidate is always available

Continuous Delivery Example



Steps **outlined in blue** in this example are manual

@lisacrispin

Manual steps in the pipeline might be...

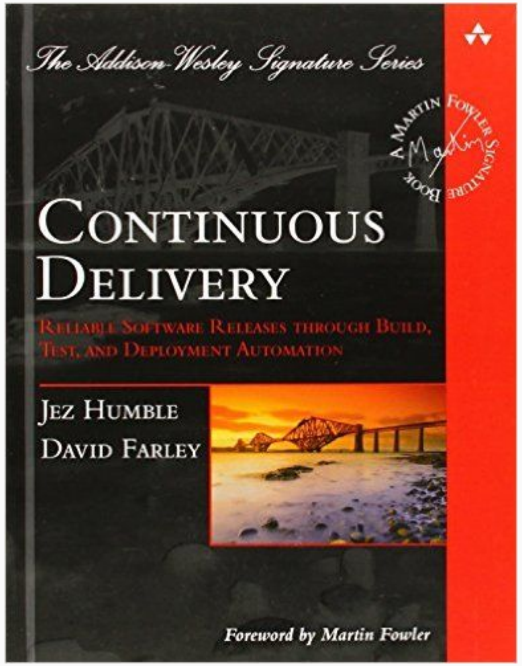
- Deploys
- Exploratory testing
- Visual checking
- ... what else can you think of?
- More on this later

@lisacrispin

Principles of continuous delivery

From Jez Humble and David Farley, continuousdelivery.com:

- Build quality in
- Work in small batches
- Computers perform repetitive tasks, people solve problems
- Relentlessly pursue continuous improvement
- Everyone is responsible

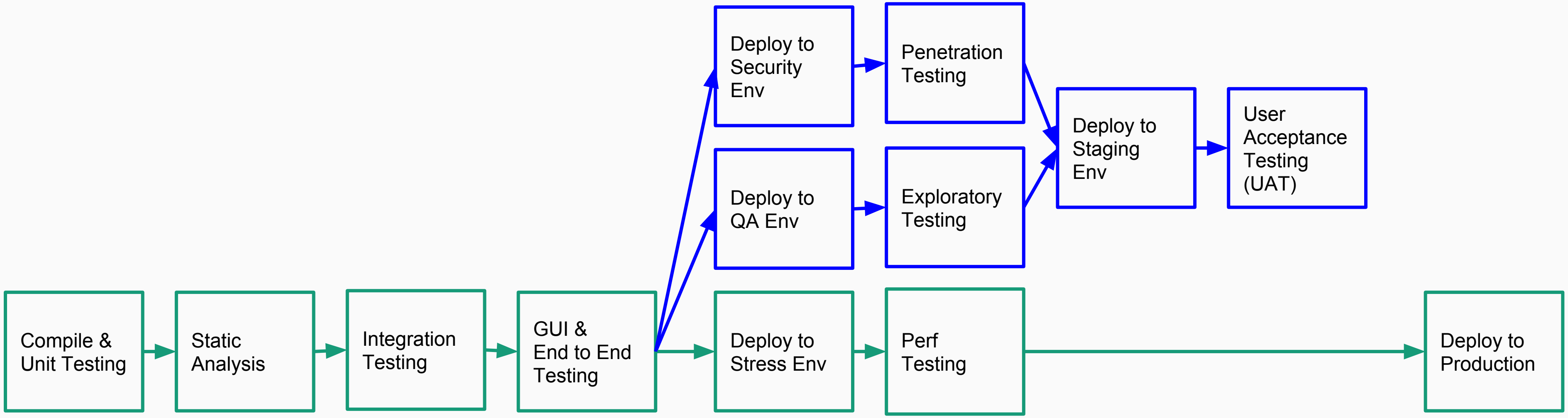

The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler

Continuous Deployment (also CD :-/)

- Deployments occur on every successfully verified commit. Often many a day.
- Heavily from automated testing and Continuous Delivery environment, but does not actually require either

Image: www.squirrelpicnic.com

Continuous Deployment Example



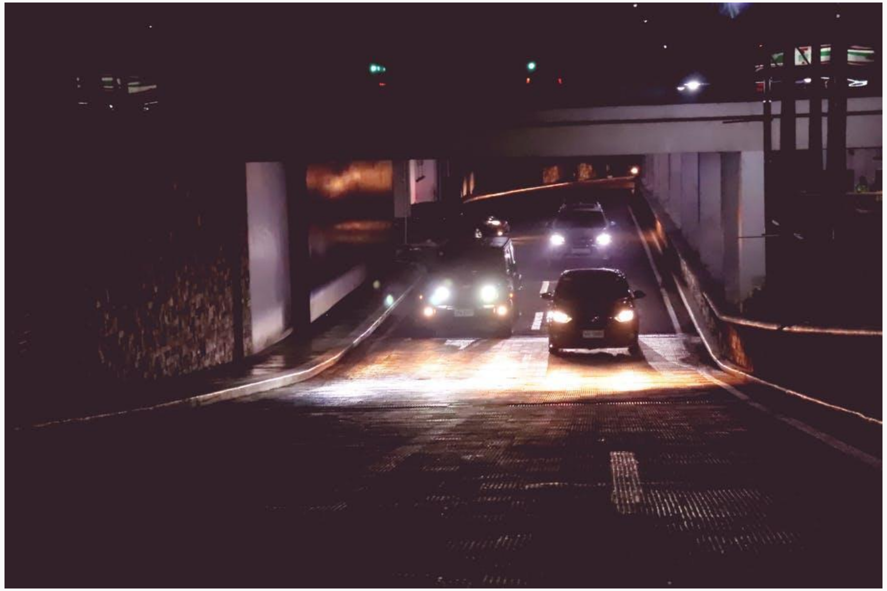
Steps **outlined in blue** in this example are manual

@lisacrispin

CD (either one) without automation...

Like driving at night without your headlights. It's possible... but headlights greatly reduce the risk!

Testing is one headlight, Monitoring is the other



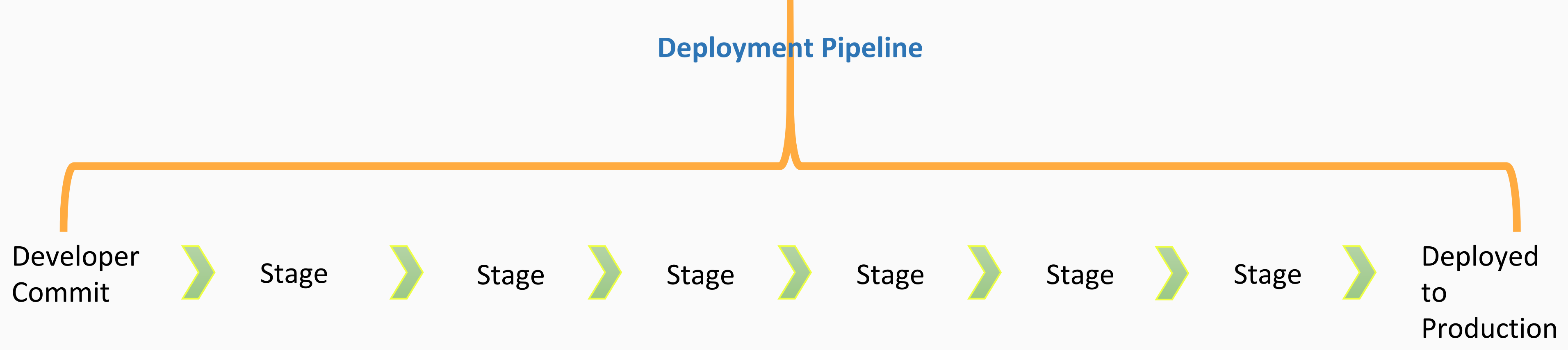
Props to Ashley Hunsberger for the analogy

@lisacrispin

To sum up the terms again…

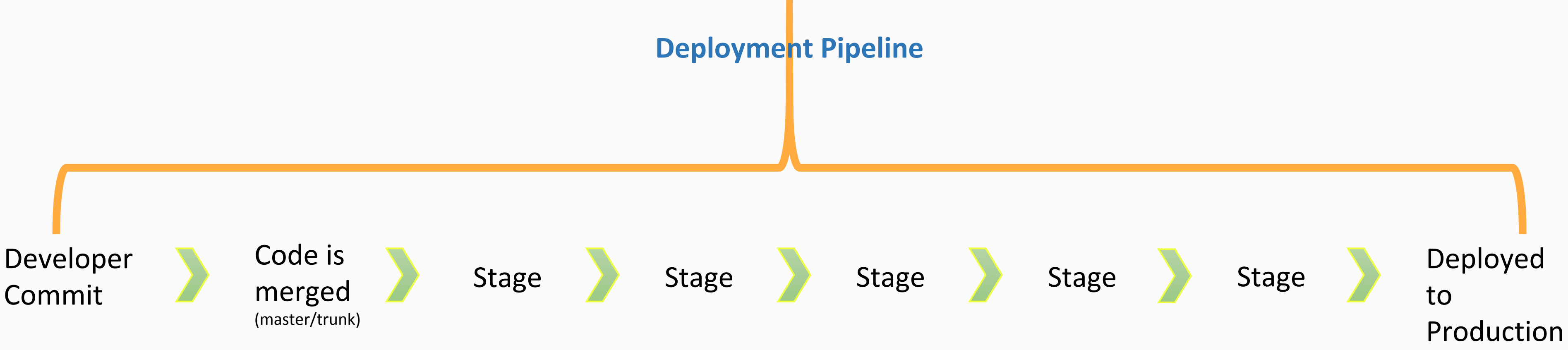
(Thanks to Abby Bangser for the following visual)

Deployment Pipeline

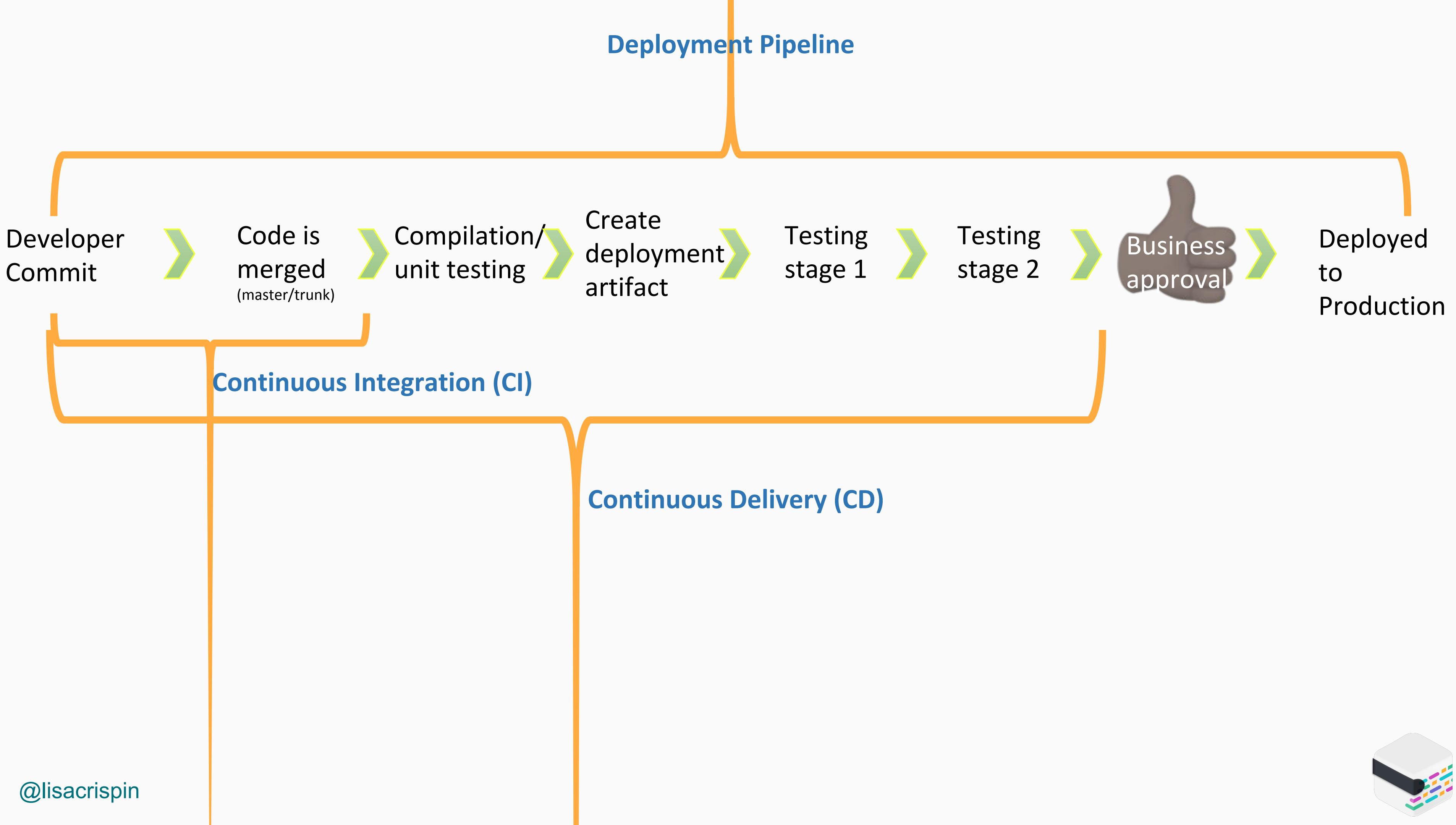


@lisacrispin

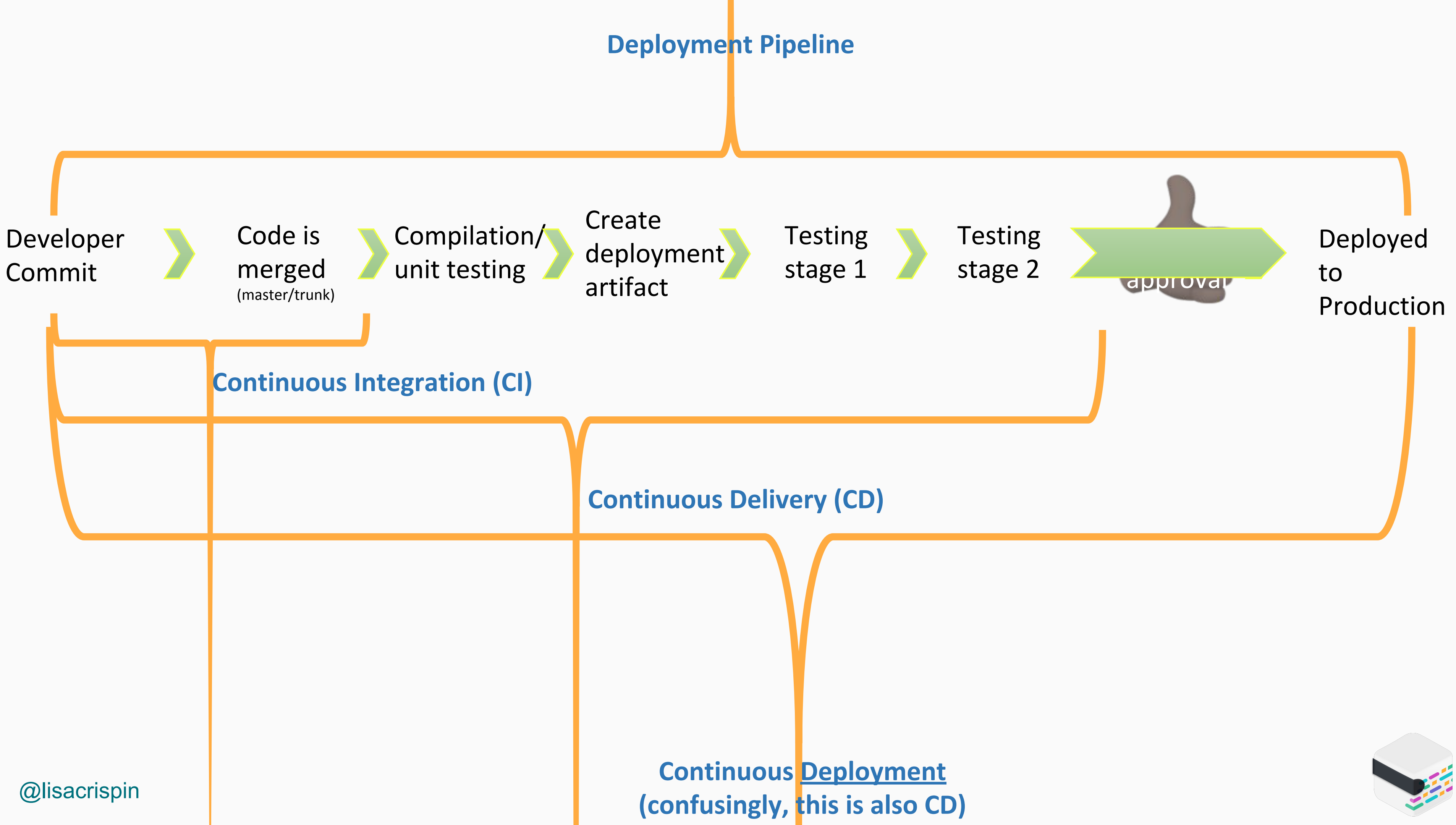
Deployment Pipeline



@lisacrispin



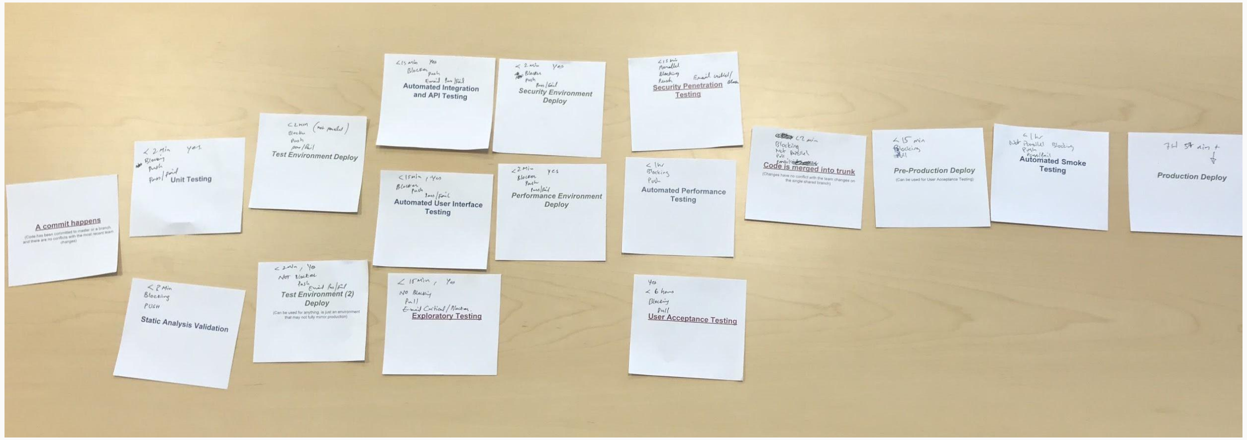
@lisacrispin



@lisacrispin

Visualize your pipeline

- Try getting a cross-functional group of team members together
  - Devs, testers, product folks, ops
- Write your pipeline steps on big stickies (real or virtual)
- Arrange them on a table, wall, virtual whiteboard
- Talk about it!



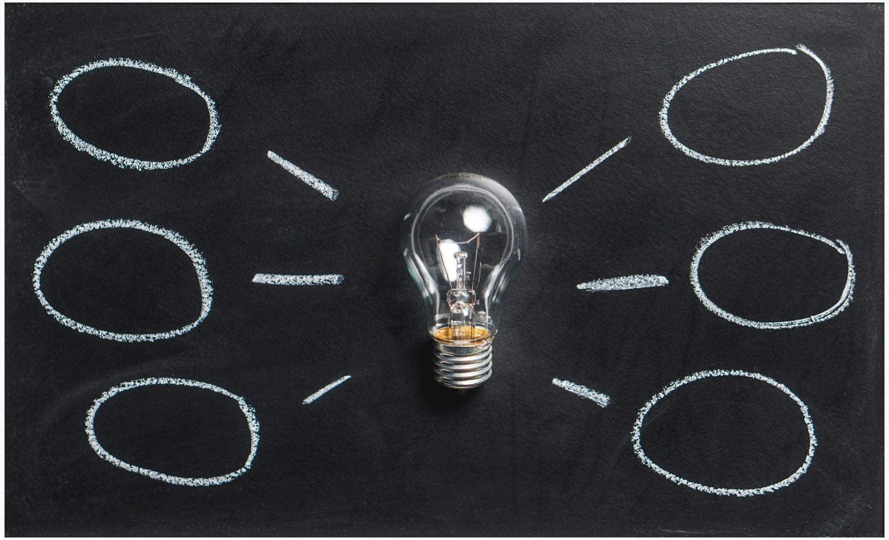
A commit happens
(Code has been committed to master or a branch and there are no conflicts with the most recent team changes)
Unit Testing
Static Analysis Validation
Test Environment Deploy
Test Environment (2) Deploy
(Can be used for anything, is just an environment that may not fully mirror production)
Automated Integration and API Testing
Automated User Interface Testing
Exploratory Testing
Security Environment Deploy
Performance Environment Deploy
Security Penetration Testing
Automated Performance Testing
User Acceptance Testing
Code is merged into trunk
(Changes have no conflict with the team changes on the single shared branch)
Pre-Production Deploy
(Can be used for User Acceptance Testing)
Automated Smoke Testing
Production Deploy

@lisacrispin

See https://www.mabl.com/blog/path-to-production-what-we-can-learn-from-our-deployment-pipelines

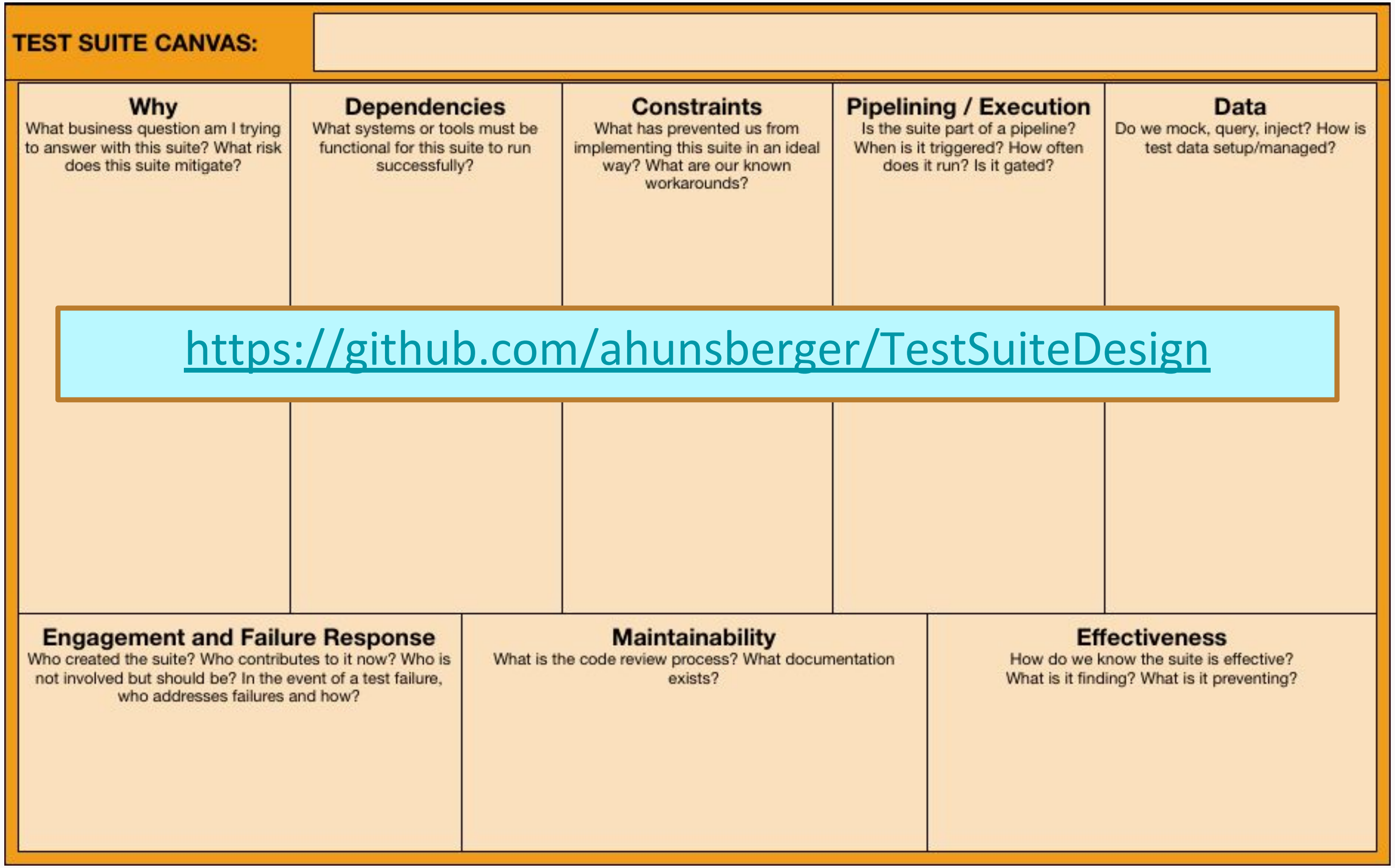
How could you deliver faster?

- Parallelizing steps?
- New tools? More automation? Moving to the cloud?
- What regression tests do you need to run for confidence?



@lisacrispin

The Test Suite Canvas (from Ashley Hunsberger, inspired by Katrina Clokie)



TEST SUITE CANVAS:

Why	Dependencies	Constraints	Pipelining / Execution	Data
What business question am I trying to answer with this suite? What risk does this suite mitigate?	What systems or tools must be functional for this suite to run successfully?	What has prevented us from implementing this suite in an ideal way? What are our known workarounds?	Is the suite part of a pipeline? When is it triggered? How often does it run? Is it gated?	Do we mock, query, inject? How is test data setup/managed?

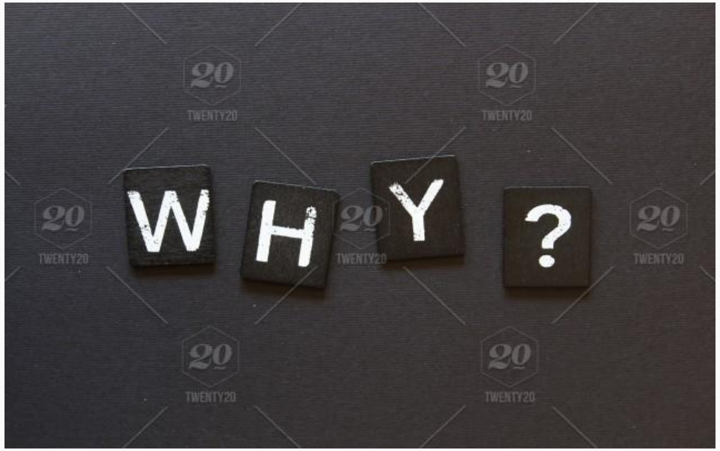
Engagement and Failure Response	Maintainability	Effectiveness
Who created the suite? Who contributes to it now? Who is not involved but should be? In the event of a test failure, who addresses failures and how?	What is the code review process? What documentation exists?	How do we know the suite is effective? What is it finding? What is it preventing?

https://github.com/ahunsberger/TestSuiteDesign

@lisacrispin

Let’s explore a few canvas discussion points

A good place to start is…



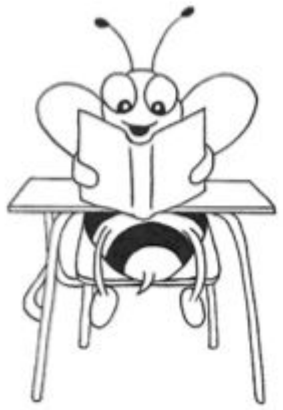
do we do each step in our current pipeline?

What do we want to learn from each step?

What business questions can each step in our pipeline answer?

- Integration and build
- Static code analysis
- Automated test suites
- Manual testing

Who can benefit from the information?
How should they be informed?
What risks can we mitigate with each step?



A few real world examples...

API Test Suite

- Am I getting proper responses that warrant UI testing?

Static Code Analysis

- Are we meeting accessibility standards?

Build Installer Testing

- Does the build install without error so that it is worth further testing?

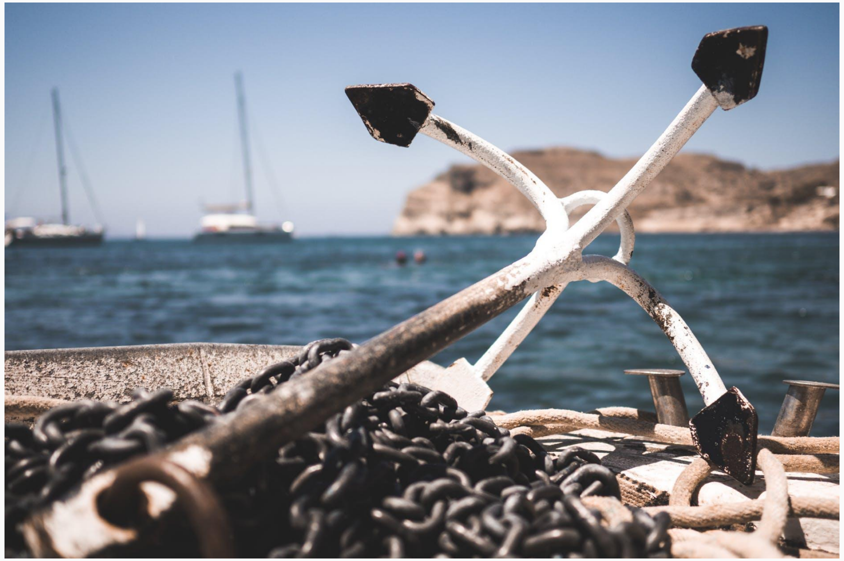
Dependencies

What needs to be in place for a given step to run successfully?

- Other systems
- Tools
- Data, environments...

Constraints

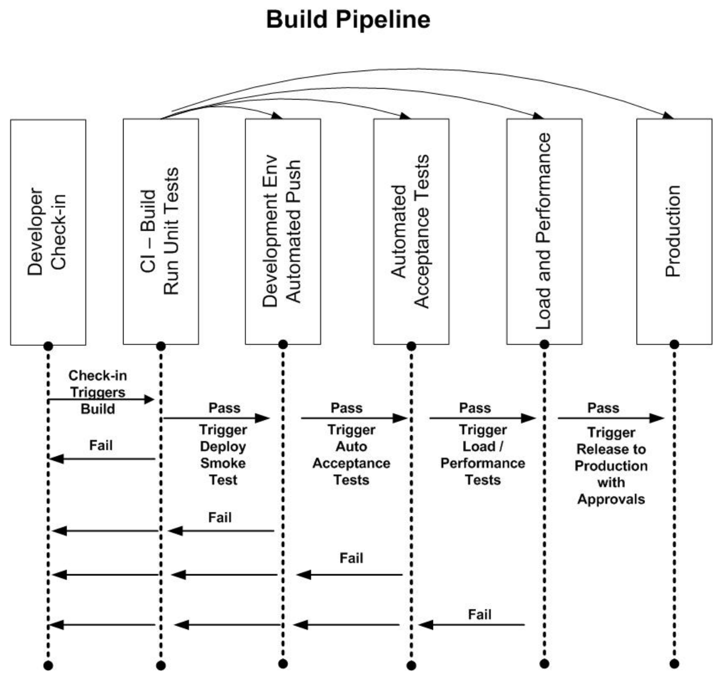
- What’s preventing us from optimizing a given step?
- For example: is it a manual step we could automate?
- Do we have automation, but it’s slow and flaky?
- What are our known workarounds?



@lisacrispin

Triggering each step in your pipeline

- What kicks off each step in your pipeline?
- Can you parallelize to shorten your feedback loop?
- If one prerequisite step fails but another passes – do you run the shared next step? Or stop?


Build Pipeline
Developer Check-in
CI – Build Run Unit Tests
Development Env Automated Push
Automated Acceptance Tests
Load and Performance
Production
Check-in Triggers Build
Fail
Pass
Trigger Deploy Smoke Test
Fail
Pass
Trigger Auto Acceptance Tests
Fail
Pass
Trigger Load / Performance Tests
Fail
Pass
Trigger Release to Production with Approvals

@lisacrispin

Gates

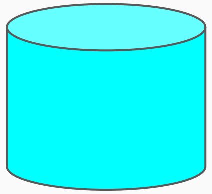
- Automatically stop defects from making it any further downstream – fast feedback
- Trust your tests – flaky tests are useless
- Use new technology to help make tests trustworthy

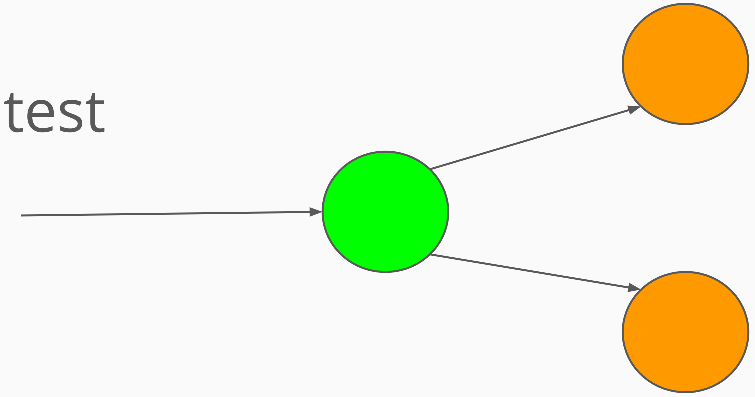
@lisacrispin

Test data

How do we manage test data?

- Tradeoff of speed vs. simulating production
- Unit tests use test doubles - fakes, stubs, mocks
- Higher level tests use fixture or canonical data
  - which simulates prod data
- Setup and teardown for each test





There’s more to the canvas

- Use it to generate conversations
- Make sure you address everything important to give you confidence for continuous delivery

Building a DevOps Culture

DevOps isn't a role or a team

It's collaboration between the
software delivery team (including testers)
and the system administration and operations team

Building relationships

This whole team approach sounds nice, but...

- How can we engage others to collaborate?

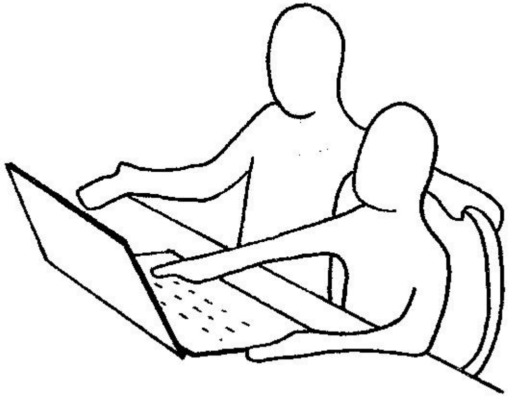
What are your ideas?

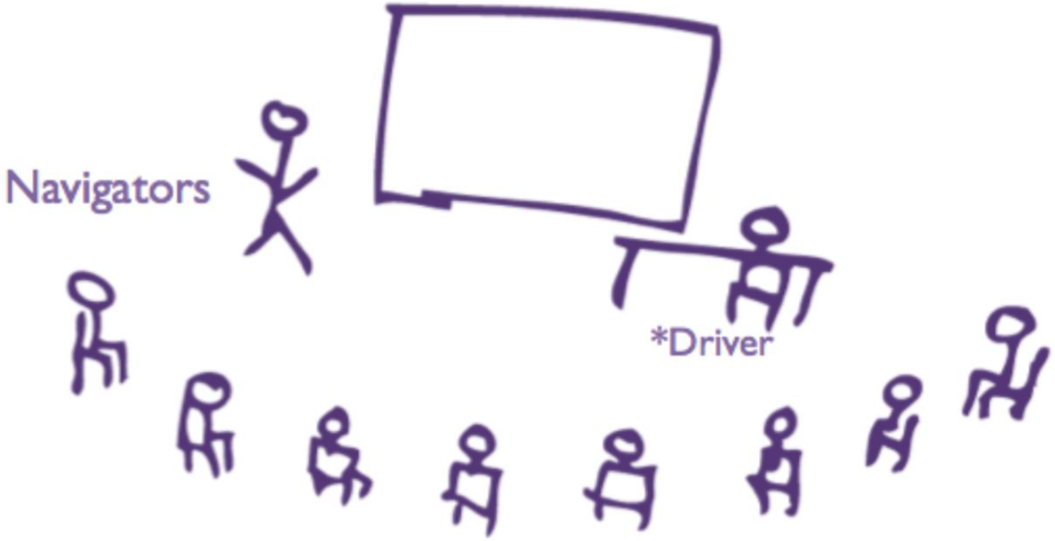
We're humans! (or possibly dragons, donkeys, unicorns...)

- Start with casual, friendly conversations
- Do food
- Share something useful
- Ask people in other roles/teams to participate, share their knowledge

Katrina Clokie has excellent tips in
A Practical Guide to Testing in DevOps

Cross-discipline pairing, mobbing





Picture from Mob Programming Guidebook, Maaret Pyhäjärvi and Llewellyn Falco

“Stop the line” mentality - from Toyota

Every employee on the assembly line has a responsibility to “stop the line” when they see a defect

- Benefit of whole team approach

Pushing the “big red button” is an investment that leads to improvements:

- Knowledge sharing
- Cost, speed
- Reliability

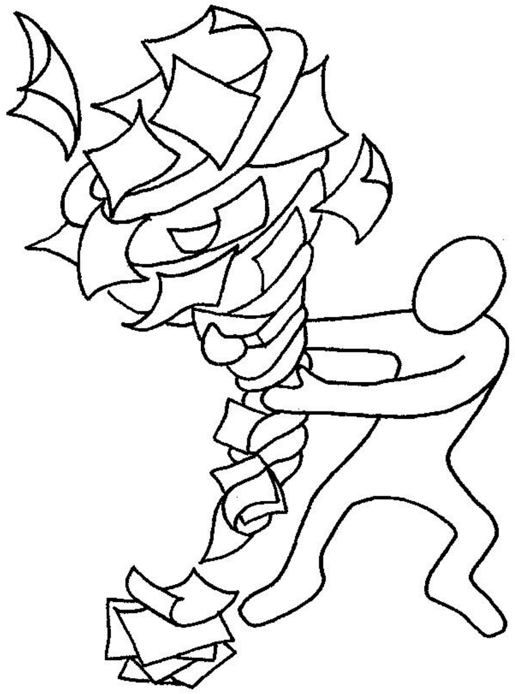
Learning from production use

Monitoring, observing

- Testing in production is a necessity
- Big data and the tools to instrument & monitor it are here
- AI, ML allow us to process the data
- Need ability to respond quickly to pain points
- Team discipline to respond to alerts
- Usage trends can inspire new features
- “Learning releases” aka “MVP”
- A/B, beta testing

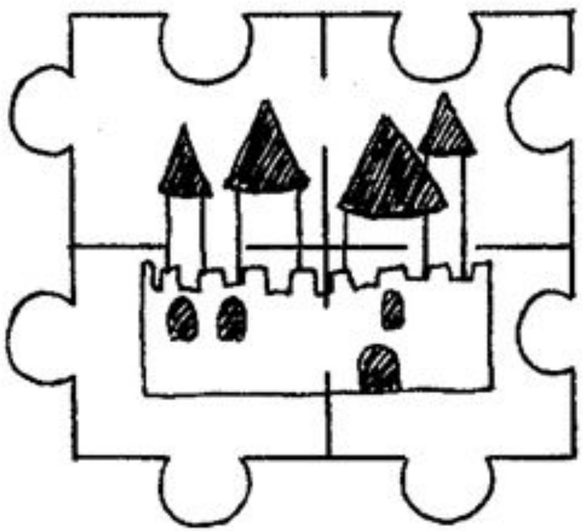
Exposure control

- Your team needs to master feature toggles!
- Dogfooding, canary release, staged rollout, dark launch



Fitting all the different types of testing into CD

- Exposure control, which lets you...
- ...release small changes frequently - manage risk
- Developers exploratory test & more at story level
- Testers pair & mob with devs, designers, product people...

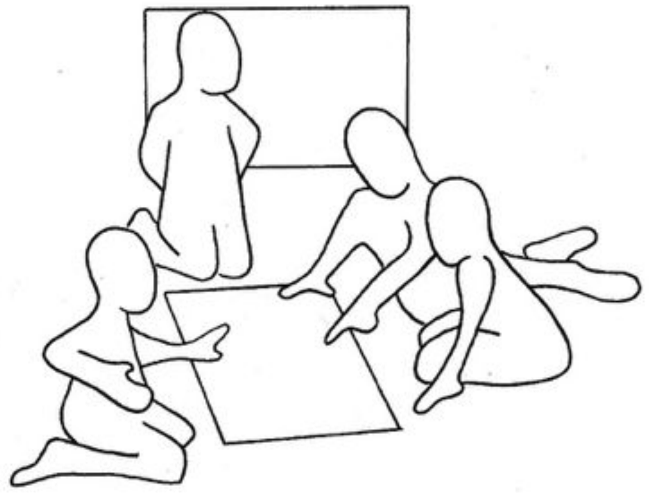


Keep testing visible to keep it continuous

- Stories for all types of testing at feature/epic level go into the backlog with feature stories
  - Exploratory test charters AND
  - A11y, I18n, security, reliability, performance … stories go into the backlog with feature stories
- Anyone can pick up a testing task

Take advantage of new technology

- Use your retrospectives, identify pain points, roadblocks
- Small experiments with new approaches & tools
- Production monitoring, observability
- There’s no silver bullet, but we can continually improve



@lisacrispin

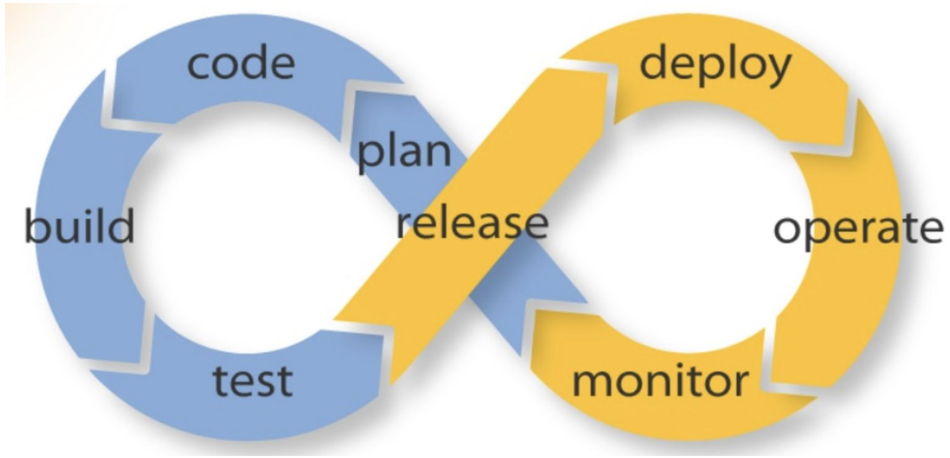
Important DevOps topics I’m not covering today

- Infrastructure as code
- Configuration management
- Containers
- Cloud
- Environment management
- Infrastructure testing

… See *Continuous Delivery* and *A Practical Guide to Testing in DevOps*

Succeeding with the whole team approach

- Collaborating to continuously improve pipelines, feedback
- Whole team commitment, engagement
- Visualize together, experiment
- Baby steps - it's a process
- Not "shifting left or right" – it's infinite!



@lisacrispin

Any questions?

What will you try with your own team?

@lisacrispin

A few resources

- *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation* (Jez Humble and David Farley)
- *A Practical Guide to Testing in DevOps* (Katrina Clokie)
- "The Era of Intelligent Testing" (Dan Belcher) https://www.mabl.com/blog/the-era-of-intelligent-testing
- "What is CI/CD" (Izzy Azeri) https://www.mabl.com/blog/what-is-cicd
- *Accelerate: The Science of Lean Software and DevOps (*Nicole Forsgren, Jez Humble, Gene Kim)
- Test Suite Canvas (Ashley Hunsberger) https://github.com/ahunsberger/TestSuiteDesign
- Charity Majors' blog, monitoring & observability: https://charity.wtf/

Try mabl!