While we're waiting to start...

Before we start:

- Sit with people with different jobs or experience level from yours
- Start writing down your challenges you hope to get help with today (more time for this later)

The Whole Team Approach to Testing in Continuous Delivery

Ashley Hunsberger & Lisa Crispin
With some material from Abby Bangser

@lisacrispin
@aahunsberger

A little about us...

Blackboard

@lisacrispin
@aahunsberger

How about you?

How do you primarily identify yourself? Please go stand by the sign that best describes you - or make your own!

- Tester/QA
- Developer/coder
- Manager
- Business analyst
- Coach
- Other roles?

Learning intentions

- Continuous delivery concepts
- Common terminology
- Questions to engage you & your team with pipelines
- Test suite canvas to help design your pipeline
- Use pipelines for feedback, mitigating risks, answering questions
- Ways to succeed with testing in a continuous delivery world

Schedule

9:00 Start

10:30 Break

12:00 Lunch

1:00 Start afternoon

3:00 Break

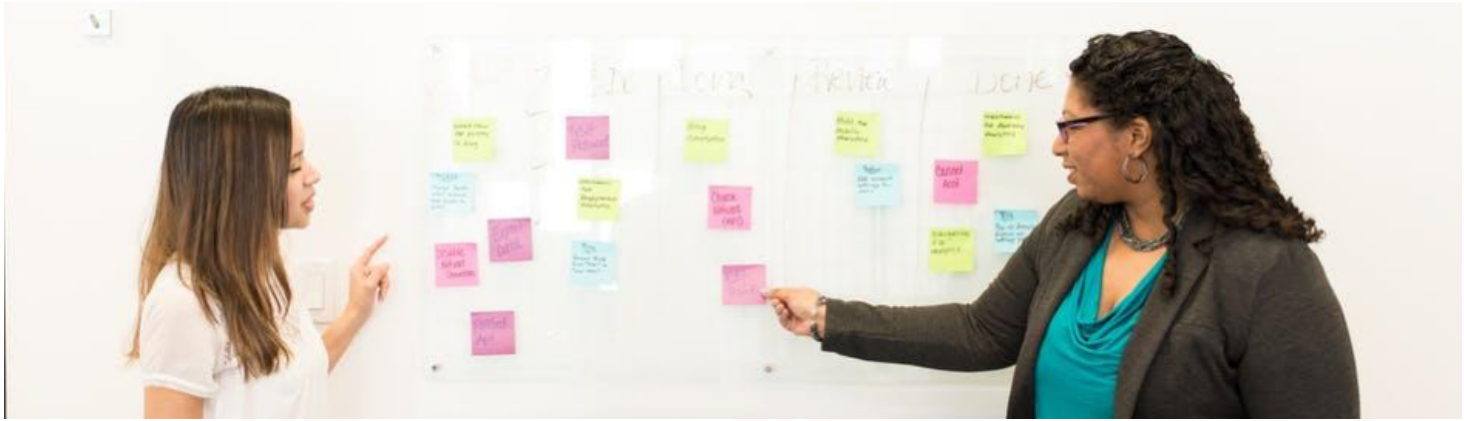
5:00 End

Your stories & questions always welcome!

@lisacrispin
@aahunsberger

Your specific challenges - in your table groups:

- Write down challenges you want help with today, 1 per sticky note
- Go around the table, each person briefly introduce their challenges
- Put the sticky notes on your wall chart. Group similar topics.
- Dot vote to choose your 3 top topic areas - each person gets 3 votes
- Be ready to share your chosen topics with the large group



@lisacrispin
@aahunsberger

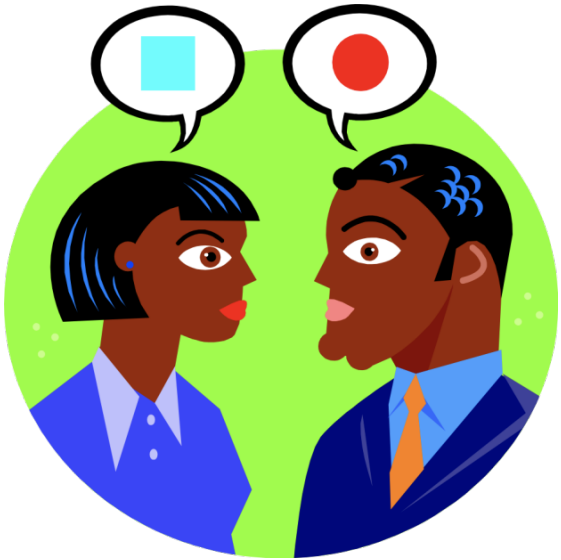
Share your challenges!

- Rep from each group shares the top 3 items

@lisacrispin
@aahunsberger

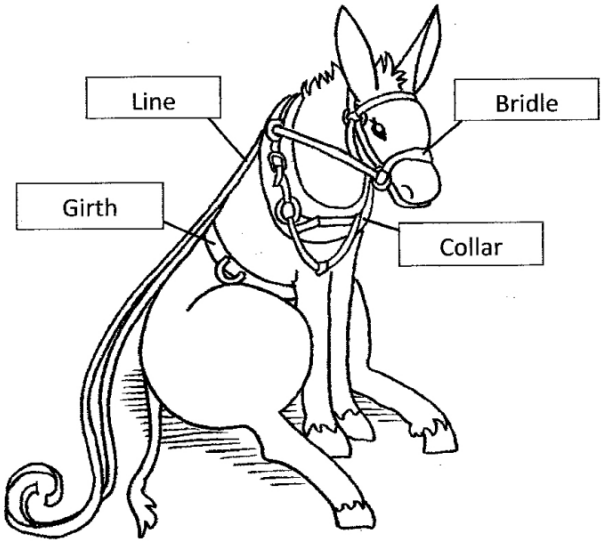
What do you think when you hear “Continuous Integration”?

How about “Continuous Delivery”?



@lisacrispin
@aahunsberger

Let’s agree on some common terminology



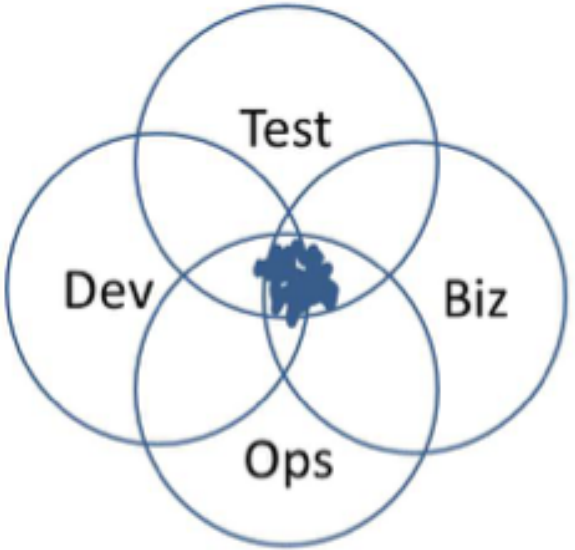
@lisacrispin
@aahunsberger

“DevOps”

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

It’s all about:

- Collaboration
- Continuous improvement
- Continuous learning



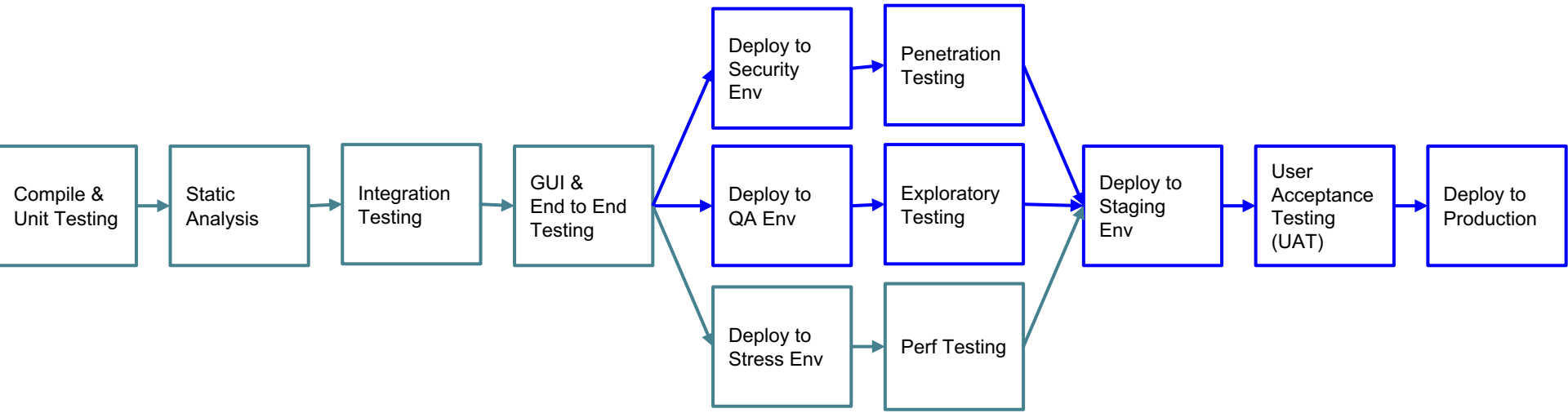
Continuous Integration

- Integrate code into a shared repository multiple times per day
- All code is integrated onto the same branch multiple times per day
- Typically the start of a pipeline
- Each check-in can be verified by an automated build with automated regression tests

Continuous Delivery (CD)

- Ability to get many types of changes into production safely, quickly and sustainably
  - eg. new features, configuration changes, bug fixes experiments
- Heavily benefits from automated testing but is not an explicit dependency
- Each commit is independently verified as a deployable release candidate
- A deployable release candidate is always available

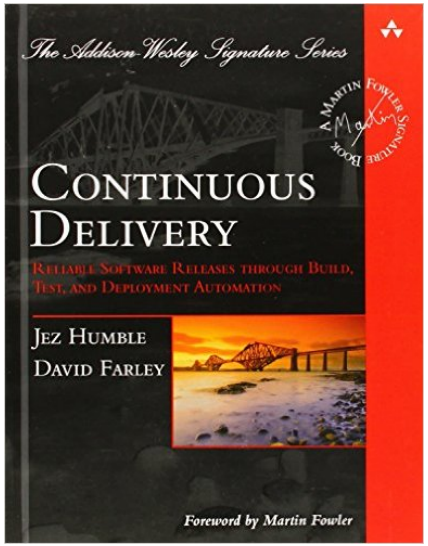
Continuous Delivery Example



@lisacrispin
@aahunsberger

Principles of continuous delivery

From Jez Humble and David Farley, continuousdelivery.com:

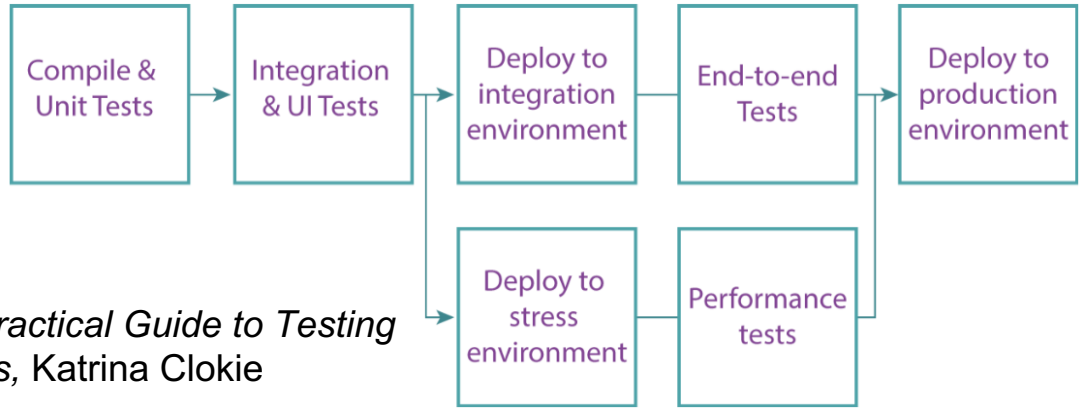


- Build quality in
- Work in small batches
- Computers perform repetitive tasks, people solve problems
- Relentlessly pursue continuous improvement
- Everyone is responsible

@lisacrispin
@aahunsberger

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



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

Remember...

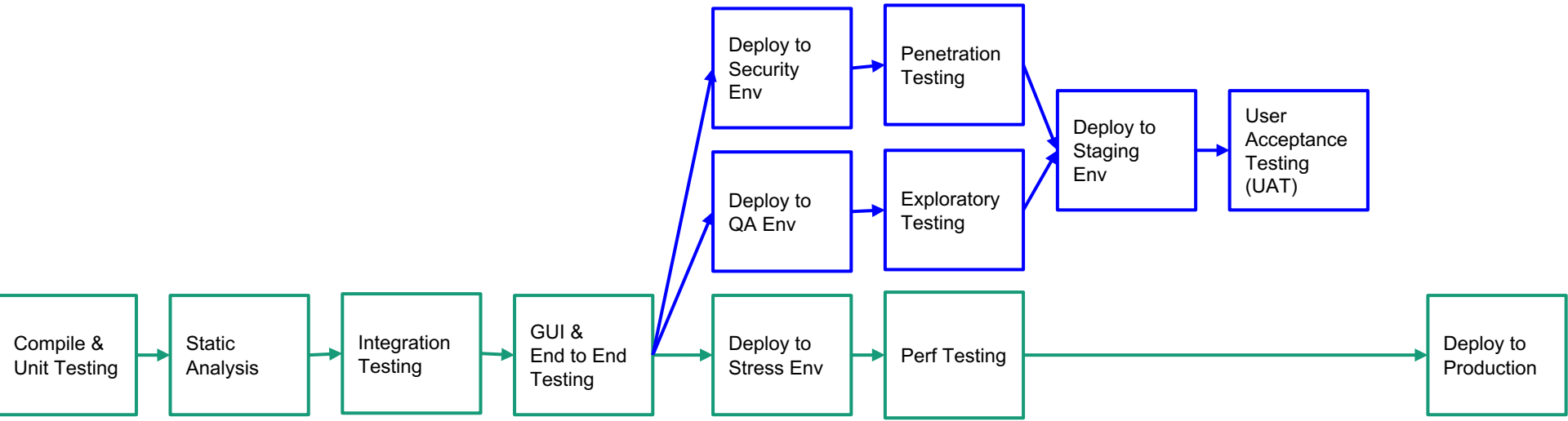
Some, or many, steps in the pipeline may be manual!
And that is ok!

- Deploys
- Exploratory testing
- Visual checking
- … what else can you think of?

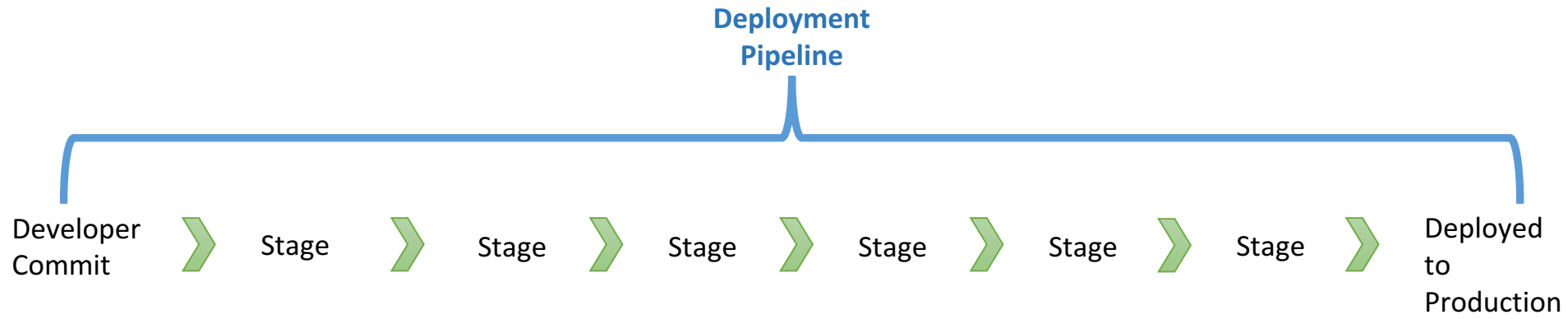
Continuous Deployment (also CD :-/)

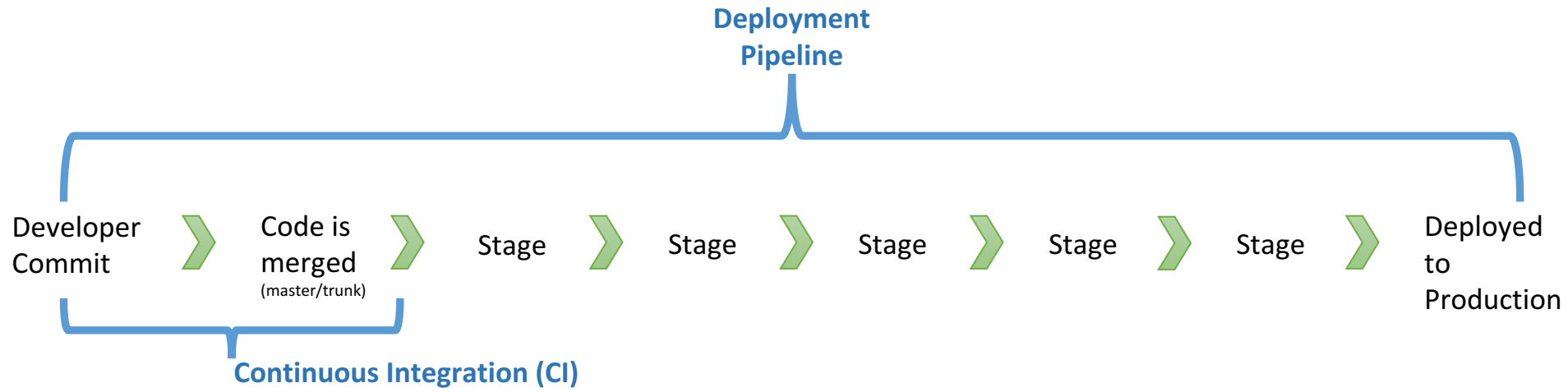
- Each commit is independently verified as a deployable release candidate and given it passes all verifications is automatically deployed to production
- Again, heavily benefits from automated testing and Continuous Delivery environment, but does not actually require either
- Deployments occur on every successfully verified commit. Often many a day.

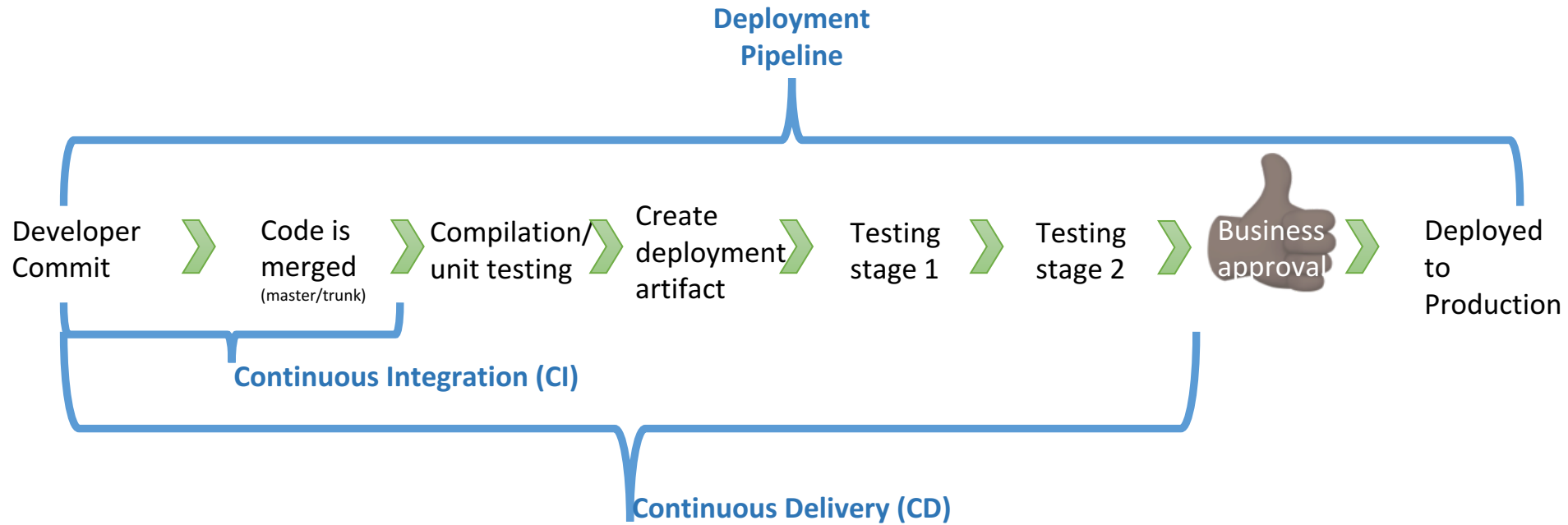
Continuous Deployment Example

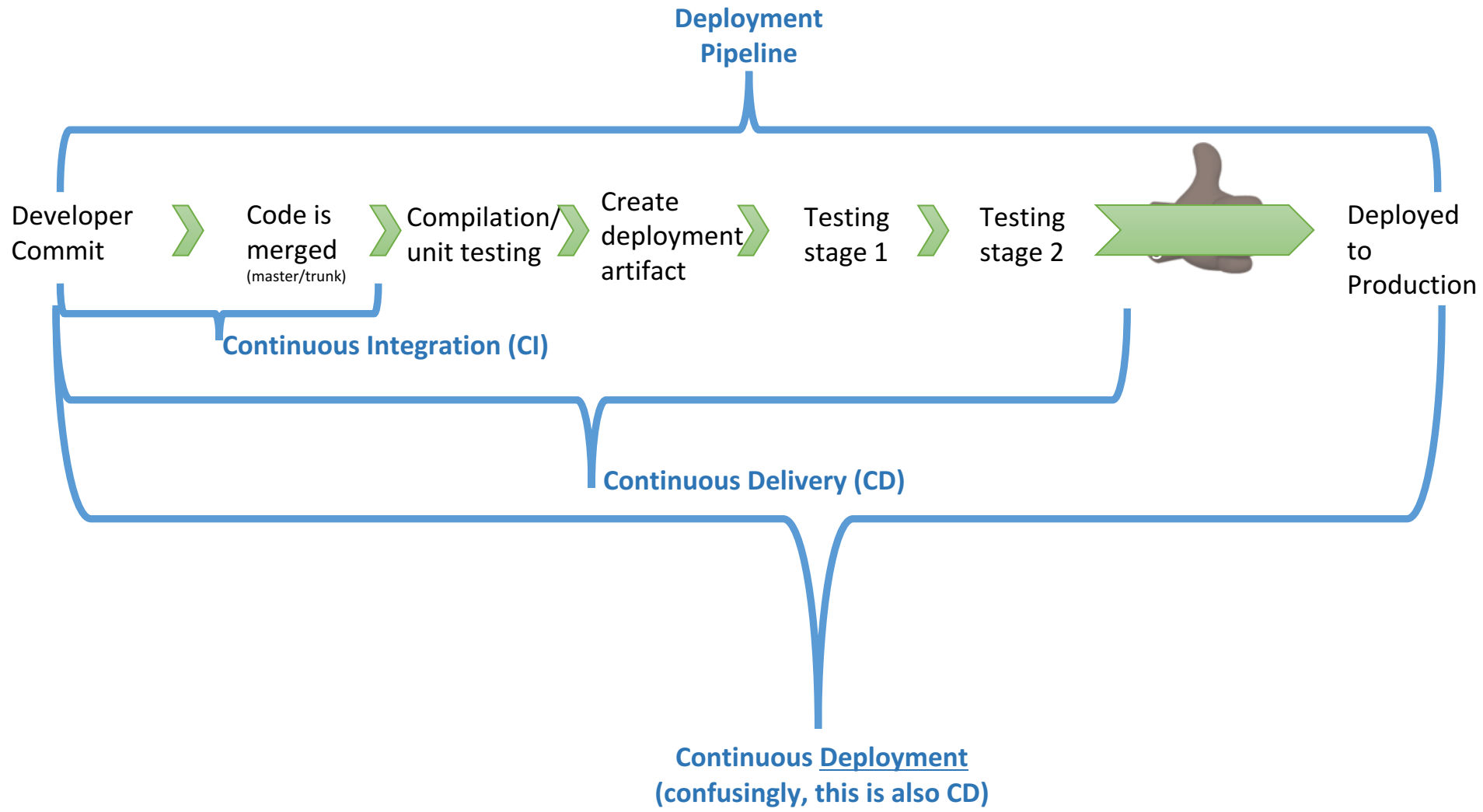


@lisacrispin
@aahunsberger


Deployment Pipeline
Developer Commit
Stage
Stage
Stage
Stage
Stage
Stage
Deployed to Production


Deployment Pipeline
Developer Commit
Code is merged
(master/trunk)
Stage
Stage
Stage
Stage
Stage
Deployed to Production
Continuous Integration (CI)


Deployment Pipeline
Developer Commit
Code is merged
(master/trunk)
Compilation/ unit testing
Create deployment artifact
Testing stage 1
Testing stage 2
Business approval
Deployed to Production
Continuous Integration (CI)
Continuous Delivery (CD)


Deployment Pipeline
Developer Commit
Code is merged
(master/trunk)
Compilation/ unit testing
Create deployment artifact
Testing stage 1
Testing stage 2
Deployed to Production
Continuous Integration (CI)
Continuous Delivery (CD)
Continuous Deployment
(confusingly, this is also CD)

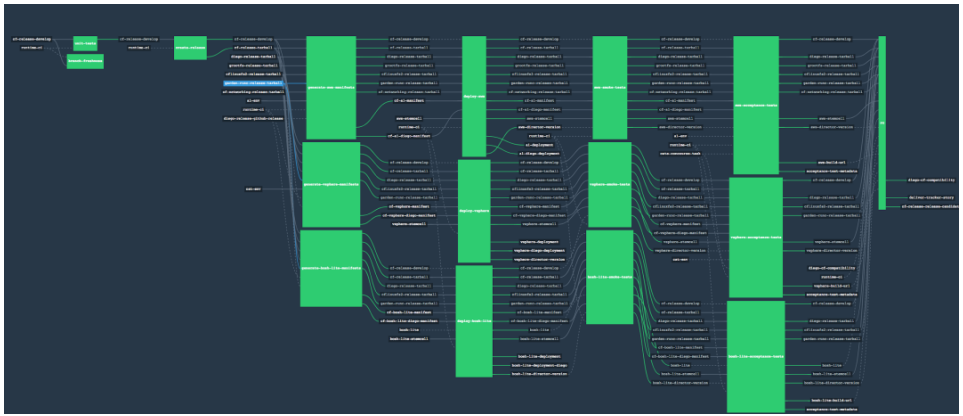
Let’s explore pipelines!

Your table group is going to be a team on one of these products for today:

- Waze/Google Maps (Navigation app)
- Etsy (E-commerce something homemade)
- Kayak (book a trip)
- Paypal (banking)
- Make up your own!

How does your code flow to prod? In your table groups:

- Write up a bug for your chosen product
- Now the bug is fixed and the change has been committed.
- How does that fix get to the customer?
    - Use the step cards provided to visualize the flow.
    - Think about examples from your own team's process.

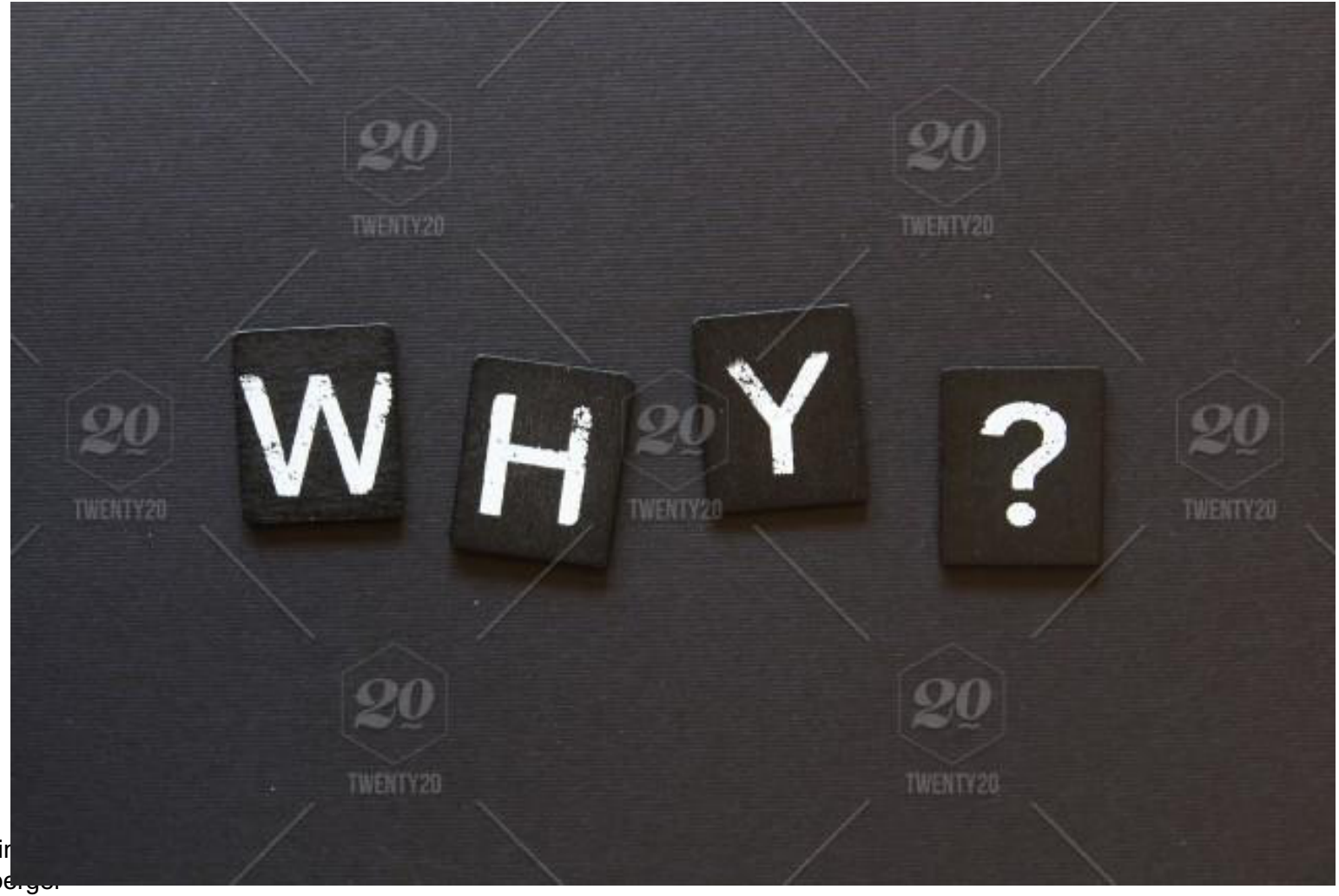


@lisacrispin
@aahunsberger

Debriefing the code flow exercise

- Did anything surprise you?
- Is anything missing from your visual?
- Were the customers happy?

@lisacrispin
@aahunsberger



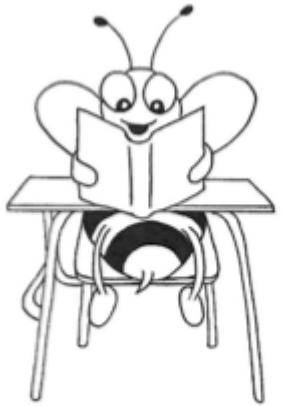
@lisacrispin
@aahunsberger

What do we want to learn from each step?

What business questions can each step in our pipeline answer?

- Automated test suites
- Manual testing

What risks can we mitigate with this step?



A few real world examples…

API Test Suite

- Am I getting proper responses that warrant UI testing?

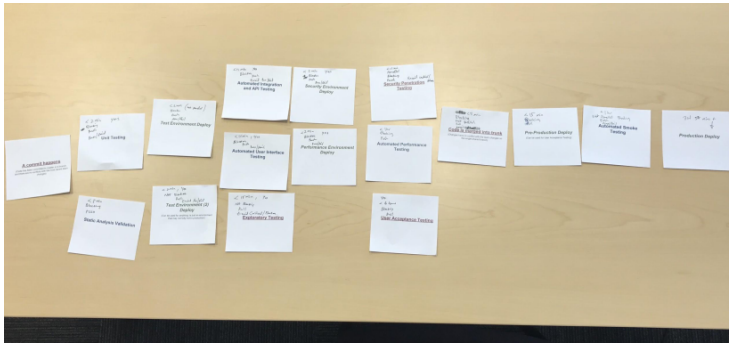
Build Installer Testing

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

In your table groups, ask “why”

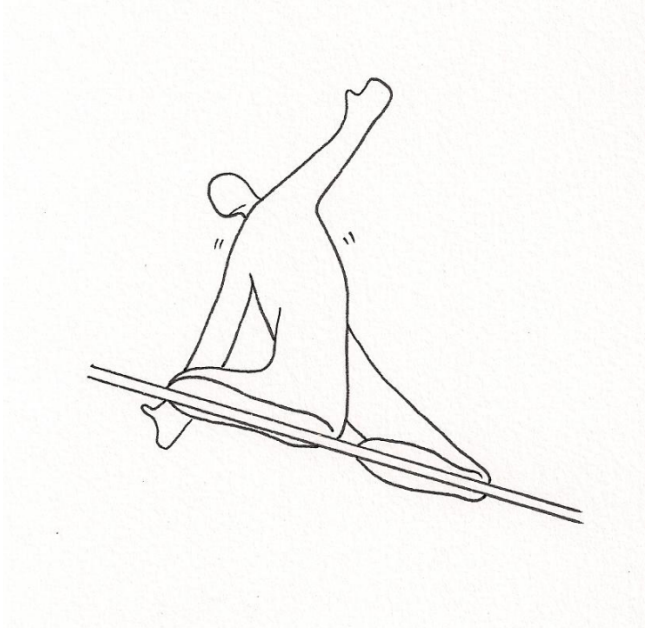
For each step:

- Identify business questions that can be answered, risks that can be mitigated
- Who on the team could benefit from that information?
- Write on sticky notes and put with the step



Debriefing the WHY exercise

- What risks did you talk about?
- Give an example of a step in the pipeline that mitigated that risk



@lisacrispin
@aahunsberger

Dependencies

- What needs to be in place for this step to run successfully?
  - Other systems
  - Tools
  - Data, environments...

Constraints

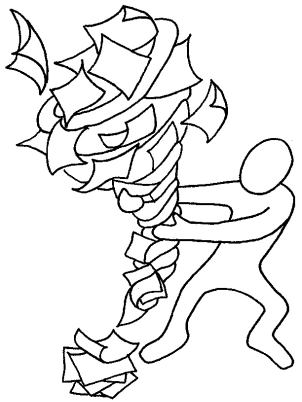
- What's preventing us from optimizing this step?
- For example: is it a manual step we could automate?
- What are our known workarounds?

In your table groups:

- Which steps are dependent on others? Which are potentially constrained by lack of skills, time, other factors?
- See if you can reorganize the steps to help with dependencies and constraints
- What actions might you take to reduce dependencies and constraints?

Dependencies and constraints outcomes

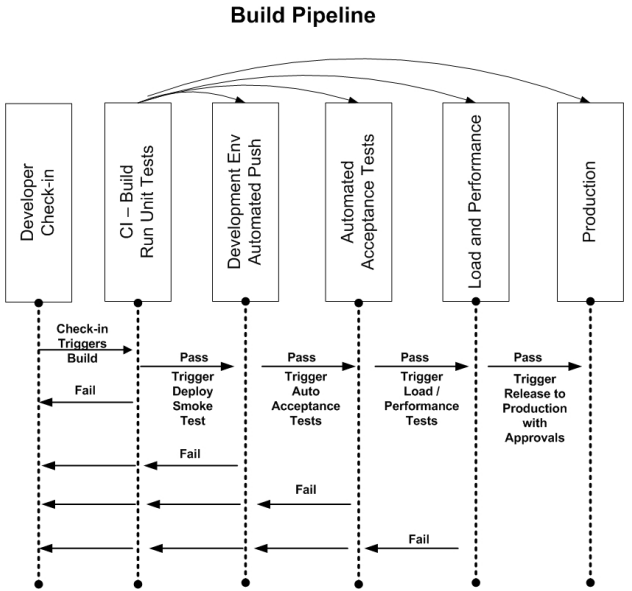
- Did you reorganize anything? Why?
- Are there any dependencies or constraints outside of your delivery team?



What triggers pipeline steps? In your groups:

For each step of your pipeline:

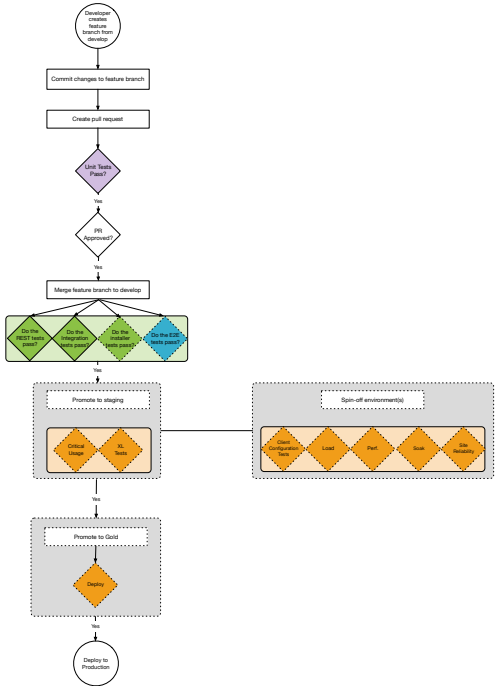
- What triggers this step? Is it triggered automatically? Manually? Based on the state of another step?
- Do you see more opportunities to parallelize steps and shorten feedback loops?



@lisacrispin
@aahunsberger

Pipeline execution discussion

- At what point in your pipeline are you running your test suites?

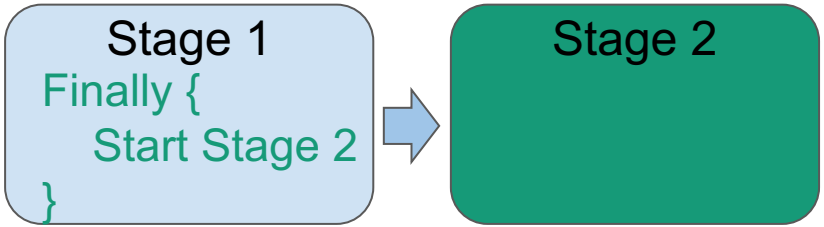


@lisacrispin
@aahunsberger

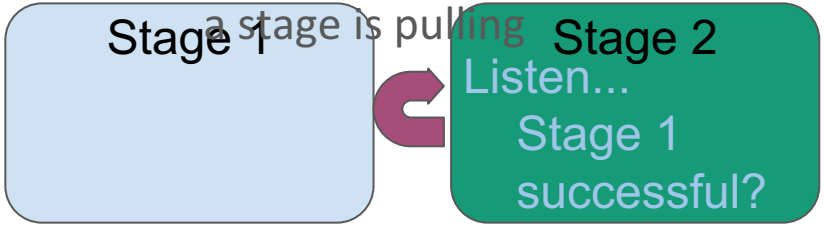
More vocabulary

Push vs Pull -

Asking a successful stage to kick off the next is pushing



Giving a stage prerequisites to poll (check periodically) for or to listen to

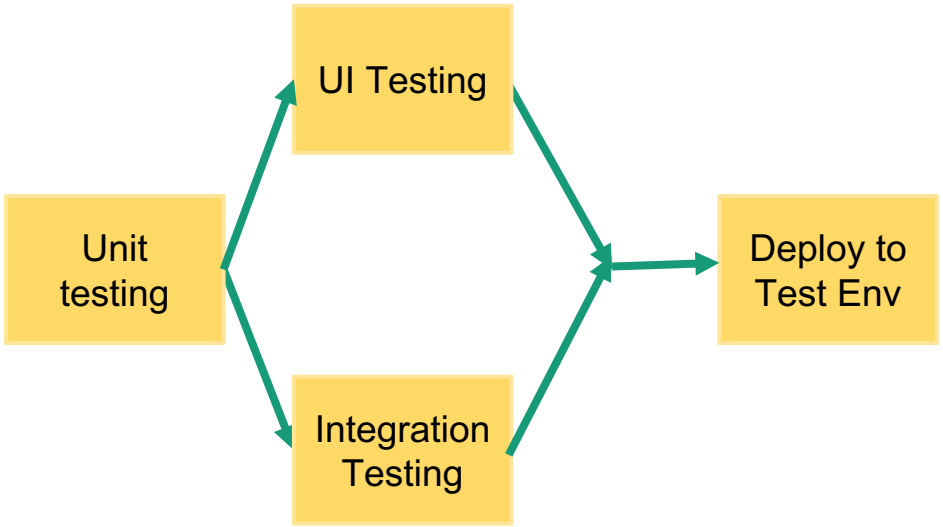


Something to think about: Push can fail a stage for reasons associated with the next stage. Pull can silently not work correctly if checking the wrong thing.

@lisacrispin
@aahunsberger

“Fan in” dependency management

Something to think about: if only one of the Prerequisite stages passes,
do you run the shared step?



Explore feedback loops some more

In your groups, take turns figuring out for each step:

- How long, for each step in your pipeline, until feedback (eg, pass or fail) is provided?
  - Write it on a sticky note and put with the step
- Who should be alerted if the step fails? How might an alert be communicated/displayed? Add this information on stickies also

Feedback Loop Discussion

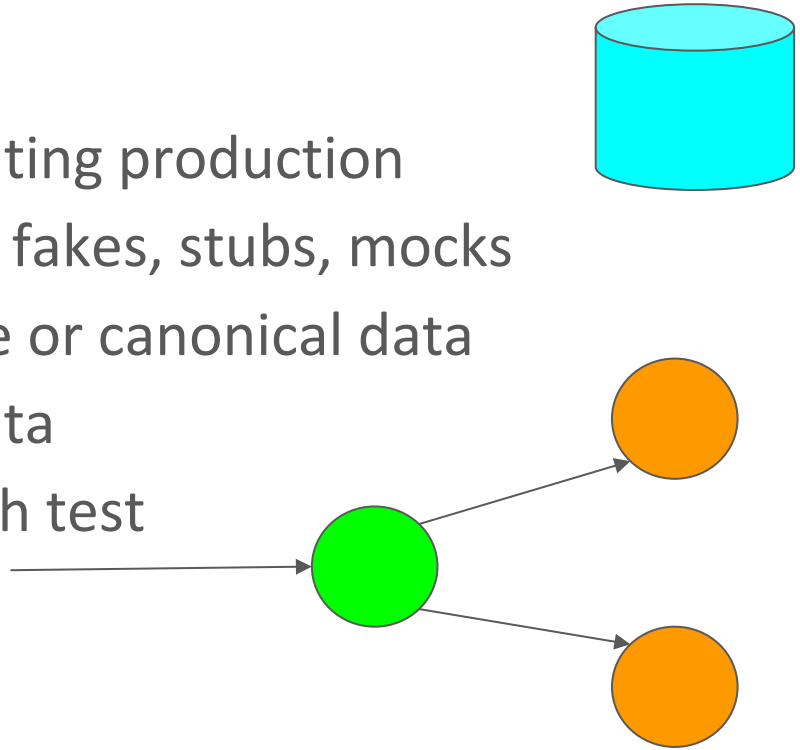
- How are you communicating feedback?
- How is feedback addressed?
- How would you make sure failures are addressed?

@lisacrispin
@aahunsberger

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



@lisacrispin
@aahunsberger

What test data do your current teams use?

- Gather with your table group around a sheet of flip chart paper on the wall.
- Take turns briefly explaining the test data that your team currently uses for different types of automated tests.
- If your team doesn’t have automated tests yet, explain the data you use for manual testing.

Test Data Discussion

- Did you discover new ways to create test data?
- What were some advantages or disadvantages of your approach over another?

@lisacrispin
@aahunsberger

Improving pipeline effectiveness

@lisacrispin
@aahunsberger

“Stop the line” mentality - from Toyota

- Every employee on the assembly line has a responsibility to “stop the line” when they see a defect
- Pushing the “big red button” is an investment that leads to improvements:
  - Knowledge sharing
  - Cost, speed
  - Reliability

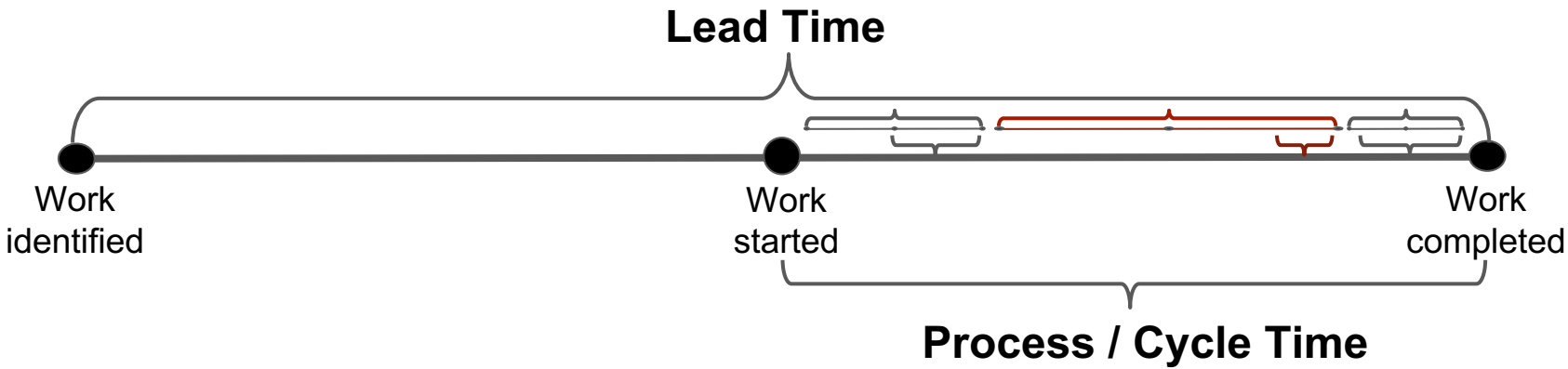
Gates!

- Automatically stop defects from making it any further down stream
- TRUST your tests (reliability)
- Whole team culture

Measuring our flow of work

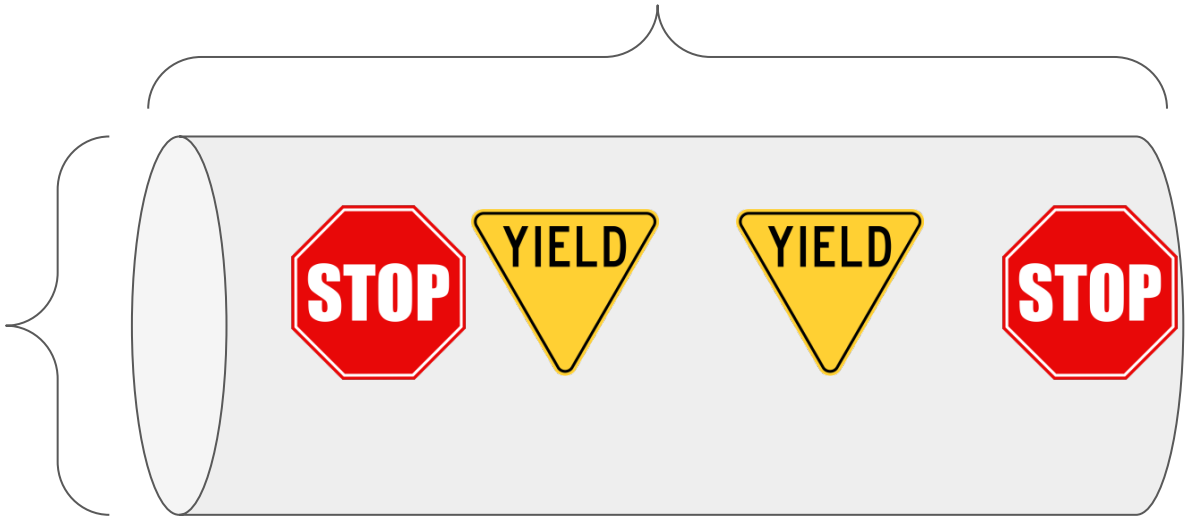
Cycle time: how long from start to delivery?

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



Latency is the amount of time to get through a pipeline line including any introduced delays

Throughput is the number and size of items that can be sent at any one time through a pipeline



@lisacrispin
@aahunsberger

How throughput and latency actually manifest

If a pipeline has **HIGH latency**, people will attempt to overcome these challenges by **increase BATCH SIZE of throughput**.

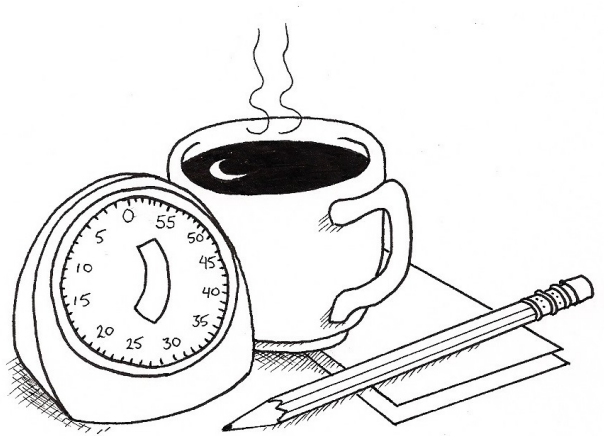
A pipeline with **LOW latency**, will encourage people to **decrease BATCH SIZE** and **increase FREQUENCY of throughput**.

Let’s look at your table group pipelines again

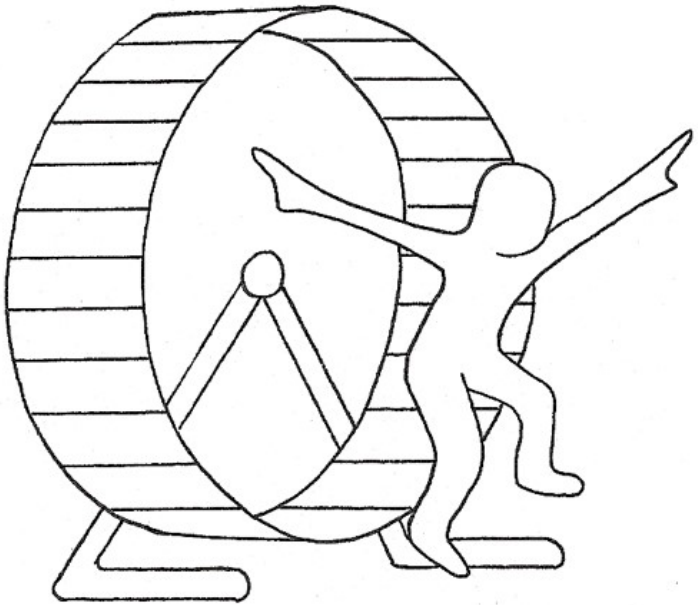
- Would you put any gates in your pipeline?
- Are there ways you can speed up your pipeline?
- What is your total lead time?

Improving the Pipeline Discussion

- Did you add gates?
- What were the pros and cons of doing that?
- How long does it take you to run your pipeline? Are there areas you can make that faster?



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



@lisacrispin
@aahunsberger

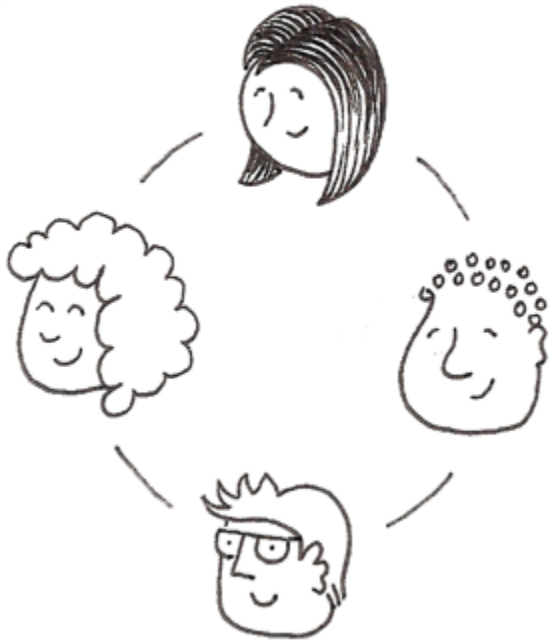
What holds your team back?

Let’s revisit your challenges from earlier - have you thought of any new ones?

@lisacrispin
@aahunsberger

Whole team responsibility

- Commitment to a level of confidence
  - Preventing bugs as well as finding them
  - Learning from production use, errors
- Diverse perspectives, skill sets
- Help overcome unconscious biases



Design an experiment - part 1

In the context of your group's product:

- Take your top 3 challenges from our opening exercise
  - If your group desires, you can substitute new ones
- Dot vote to choose the top challenge - each person gets 3 votes
- Become a cross-functional team: role-play if needed so you have a variety: product owner, tester, coder, designer, operations...

Design an experiment - part 2

For your chosen challenge:

- Create a measurable hypothesis with a prediction
- Design an experiment to test the hypothesis in the chosen work context (be specific)
- Make a poster showing challenge, hypothesis & experiment

Challenge: _____

Hypothesis:

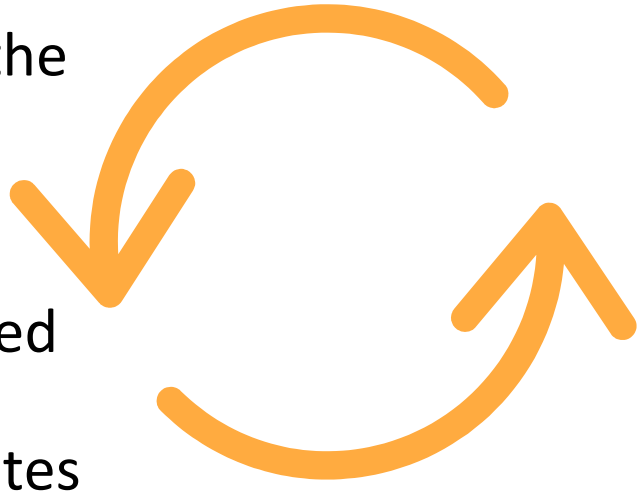
- We believe that _____
- will result in _____
- We'll know we have succeeded when _____

Experiment:

- _____
- _____

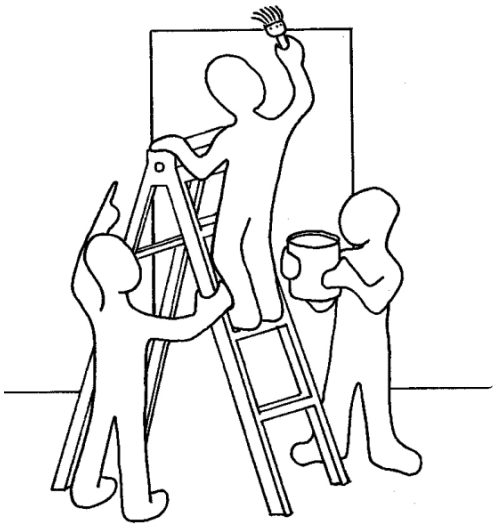
Design an experiment - part 3: Get some feedback!

- Choose a spokesperson to take your poster and travel to the next table on the left
- Everyone else stays at their table
- Spokespeople explain the challenge, hypothesis and experiment to the visited group
- The group gives feedback and contributes their ideas for the presented experiment



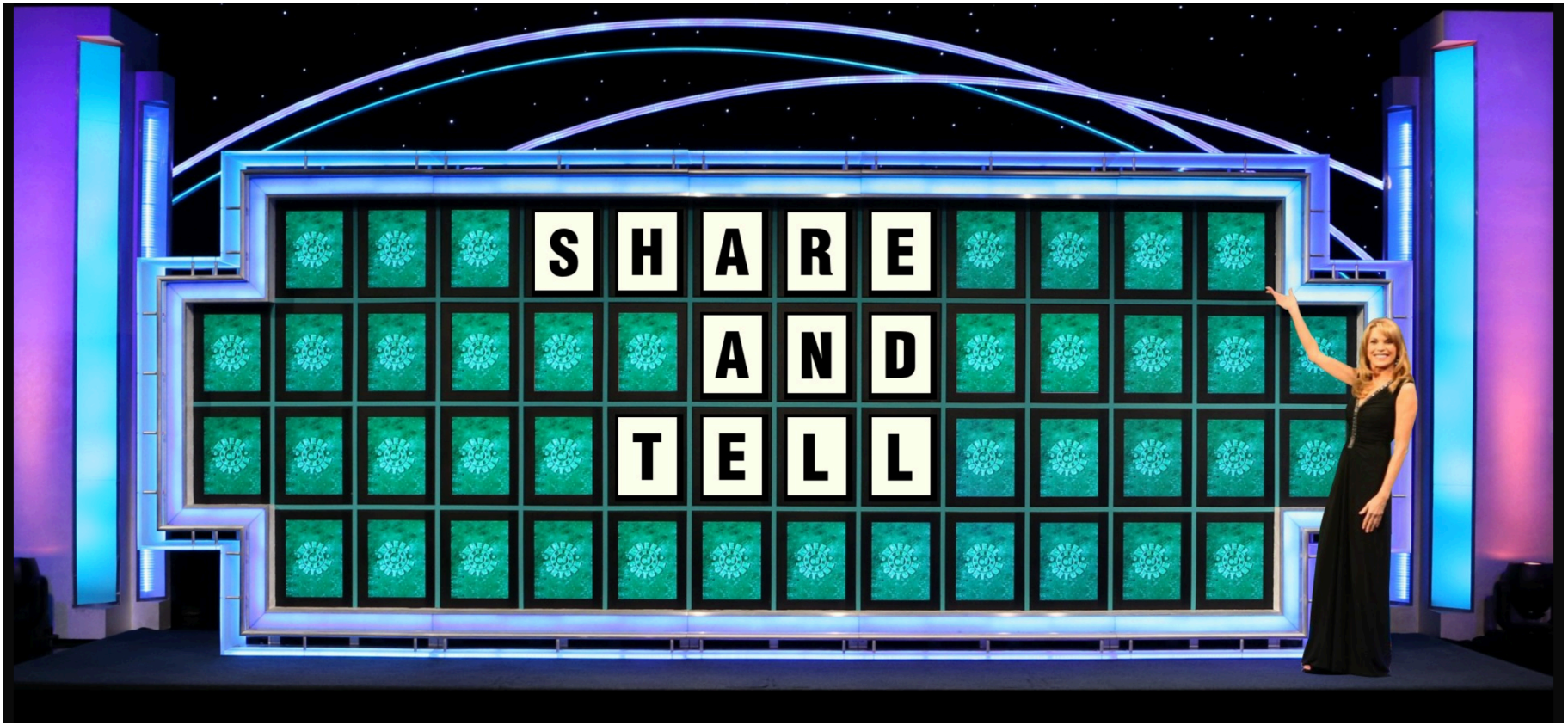
Design an experiment - part 4: Incorporate feedback

- Tweak your hypothesis and experiment based on the feedback you got from others
- Make a new poster if needed
- Get ready to present your challenge, hypothesis and experiment to everyone ☺



@lisacrispin
@aahunsberger

Group presentations



@lisacrispin
@aahunsberger

Each group has 1 minute to present:

- the challenge/problem,
- the measurable hypothesis,
- the designed experiment,
- the most valuable idea you got from the other groups that you decided to incorporate

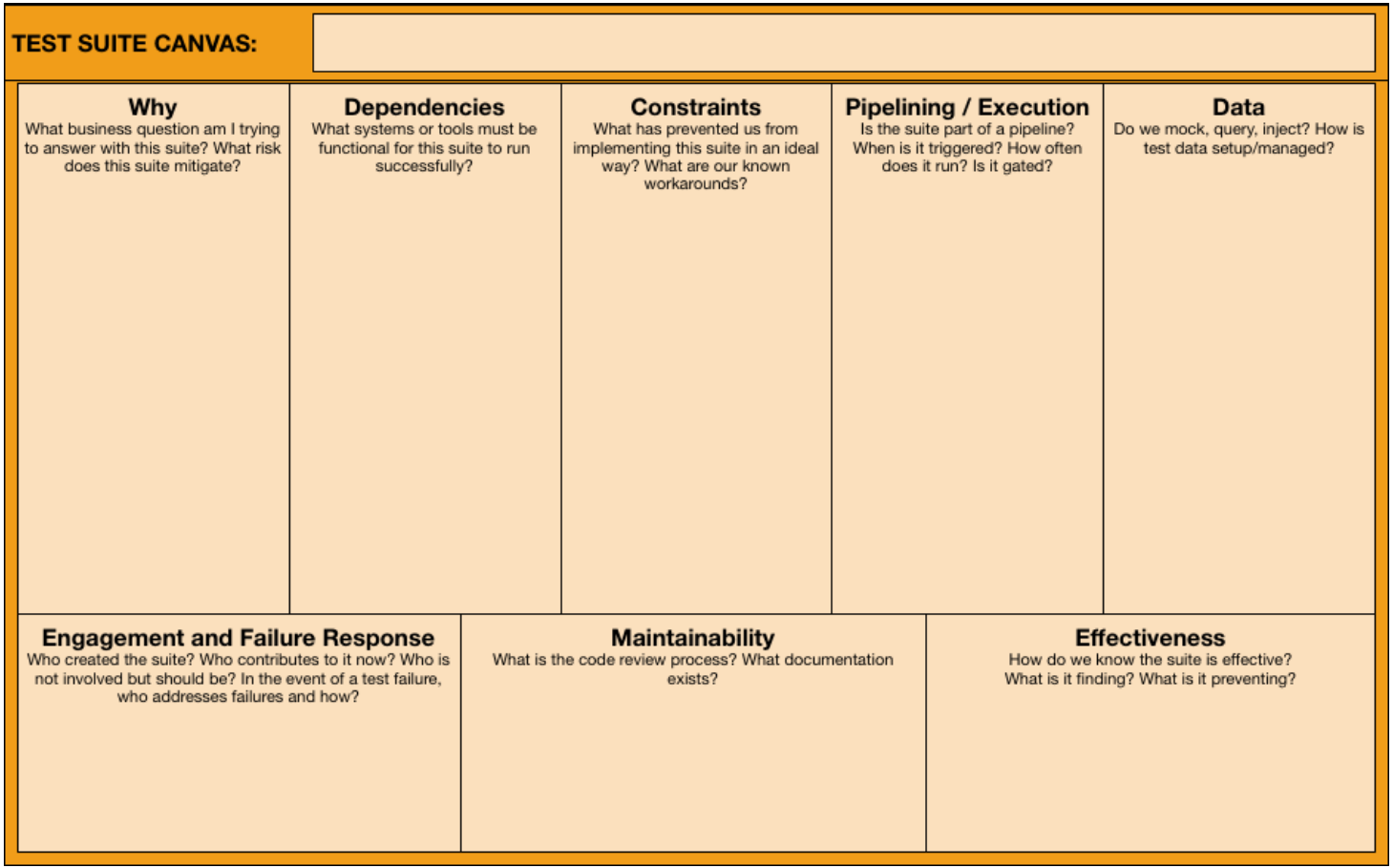
Reflecting on our experiments

- Would you be willing to try this with your real world team?
- What was helpful about getting feedback?

More tools for your toolbox

@lisacrispin
@aahunsberger

Using a test suite canvas (inspired by Katrina Clokie)



@lisacrispin
@aahunsberger

Resources list

- Agile Testing (Lisa and Janet)
- More Agile Testing (Lisa and Janet)
- Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation (Jez Humble)
- A Practical Guide to Testing in DevOps (Katrina Clokie)
- Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations (Nicole Forsgren, Jez Humble, Gene Kim)
- https://github.com/ahunsberger/TestSuiteDesign

Succeeding with the whole team approach

- Collaborating to continuously improve pipelines
- Whole team commitment
- Small batches reduce risk, speeds up the pipeline
- Visualize - draw a simple picture!
- Baby steps - it’s a process

Agile Testing and More Agile Testing

Save 35%* off the books or ebooks

- eBook formats include EPUB, MOBI, & PDF

Agile Testing Essentials video

Save 50%* on *Agile Testing Essentials LiveLessons* Video Training

Use code **AGILETESTING** at informit.com/agile

*Discount taken off list price. Offer only good at informit.com and is subject to change.

informIT the trusted technology learning source

Pearson

@lisacrispin
@aahunsberger

Selenium Conf
CHICAGO

@lisacrispin
@aahunsberger

Feedback for a chocolate :)

Your Feedback Matters

We would like to know whether you found this workshop valuable and how you plan to use what you learned. Please take a few minutes to fill in this feedback form.

1. My overall learning experience was (circle one): Great / Good / Unsatisfactory

2. I learned things that will help me in my job (circle one): Many / Some / None

3. I got the most value from the following part:

4. I got the least value from the following p

@lisacrispin
@aahunsberger

Thank you!

Let’s keep the conversation going...

@aahunsberger

@LisaCrispin