SESSION NUMBER ARIAL

# Data-driven DevOps The Key to Improving Speed & Scale


**Kohsuke Kawaguchi**

*Co-CEO Launchable, Inc.*
*Creator of Jenkins*
*kkawaguchi@launchableinc.com*


DEVOPS WORLD by CloudBees

# Who is Kohsuke?

# Jenkins!

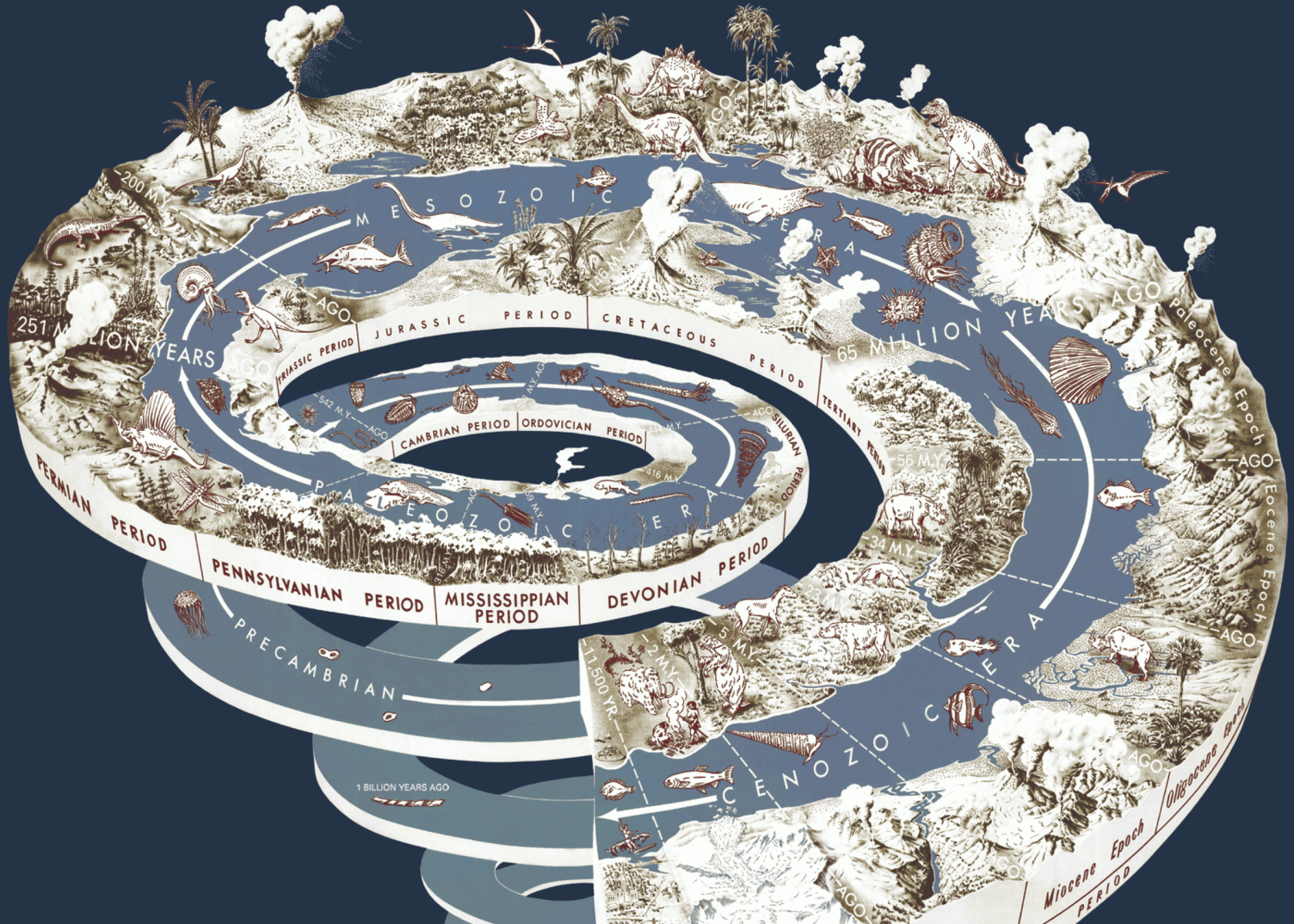
- Open-source project
- 200,000+ installations
- Helping teams everywhere

# CloudBees!

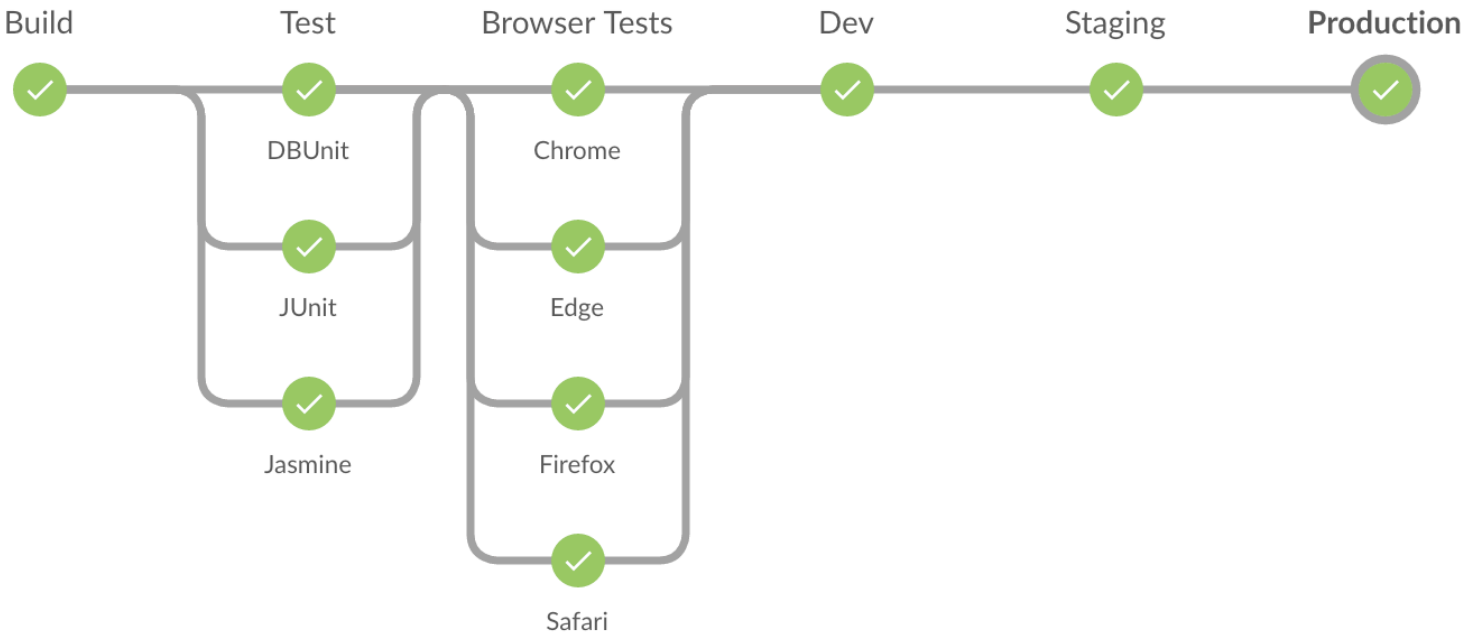
- Helping enterprises everywhere doing DevOps & Digital Transformation

# Launchable!

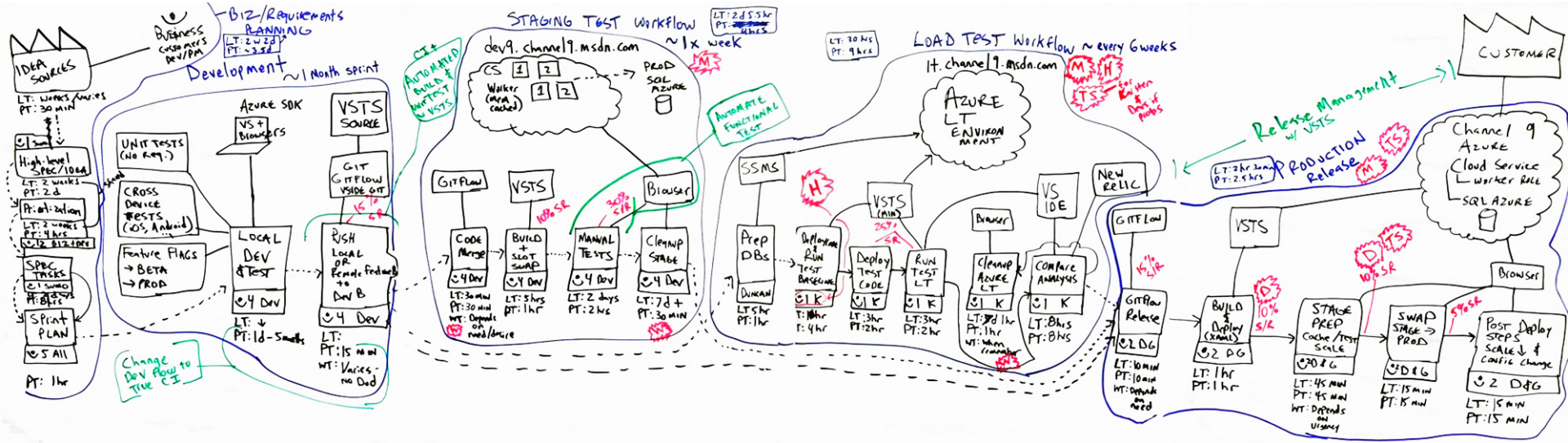
- Smarter Testing, Faster DevOps
- More about this later


MESOZOIC ERA
PALEOZOIC ERA
CENOZOIC ERA
PRECAMBRIAN
TRIASSIC PERIOD
JURASSIC PERIOD
CRETACEOUS PERIOD
TERTIARY PERIOD
CAMBRIAN PERIOD
ORDOVICIAN PERIOD
SILURIAN PERIOD
DEVONIAN PERIOD
MISSISSIPPIAN PERIOD
PENNSYLVANIAN PERIOD
PERMIAN PERIOD
Paleocene Epoch
Eocene Epoch
Oligocene Epoch
Miocene Epoch
PERIOD
65 MILLION YEARS AGO
56 M.Y.
34 M.Y.
5 M.Y.
2 M.Y.
11,500 YR.
200 M.Y.
251 M
542 M.Y.
416 M.Y.
1 BILLION YEARS AGO

# Automation Today



# Actual Automation Today




http://bit.ly/2DLhFHg

# Cost/time trade-off

- Situation
  - 100s of engineers
  - 10s of projects
  - 1 shared Jenkins infrastructure
  - $100,000s of AWS cost
- Question
  - CFO: why do we spend so much on AWS?

# What should have happened

- Visibility into cost at project level
- Make developers aware of the trade-off they are making

| | Small | Medium | Large |
|---|---|---|---|
| Build Time | 15 mins | 10 mins | 8 mins |
| Annual Cost | $1000 | $2000 | $3000 |

# Whose problem is it?

- Situation
  - 1000s of engineers
  - 1 DevOps team that runs the infrastructure
- Question
  - A build failed. Who should be notified first?

# What they have done

- Regular expression pattern matching
- Bayesian filter

## Perhaps more importantly, at Organizational Level

I want to improve our software delivery process

but it doesn’t get prioritized

## As a leader, this is your job!

**Data** (& story) helps your boss see the problem you see

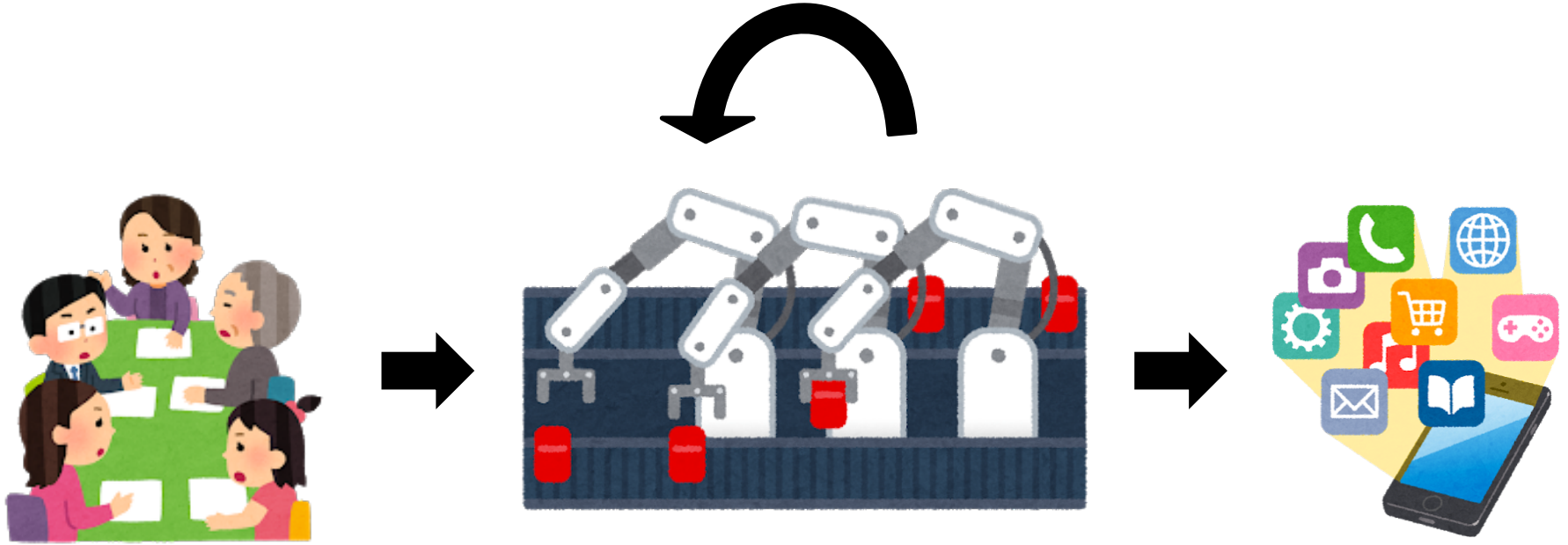
## As a leader, this is your job!

**Data** helps you apply effort to the right place

## As a leader, this is your job!

**Data** helps you show the impact of your work

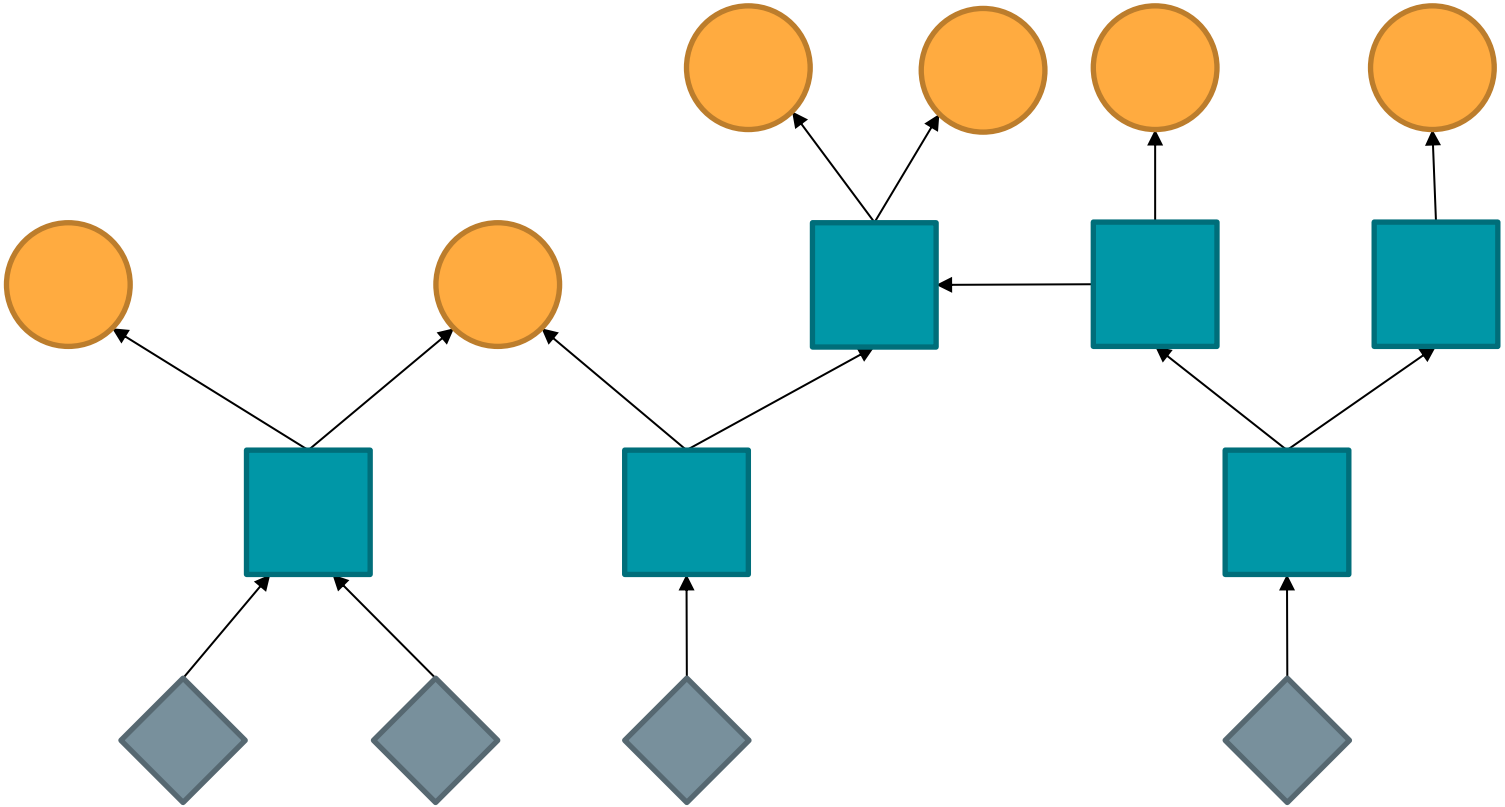
# Continuous Learning & Improvement



# Smarter testing

- Situation
  - You are the DevOps team of a BigCo
  - Massive modularized codebase with web of dependencies
  - Big, time consuming tests around them
- Questions
  - I want to cut cost & time of the software delivery process

# Step 1: Dependency Analysis



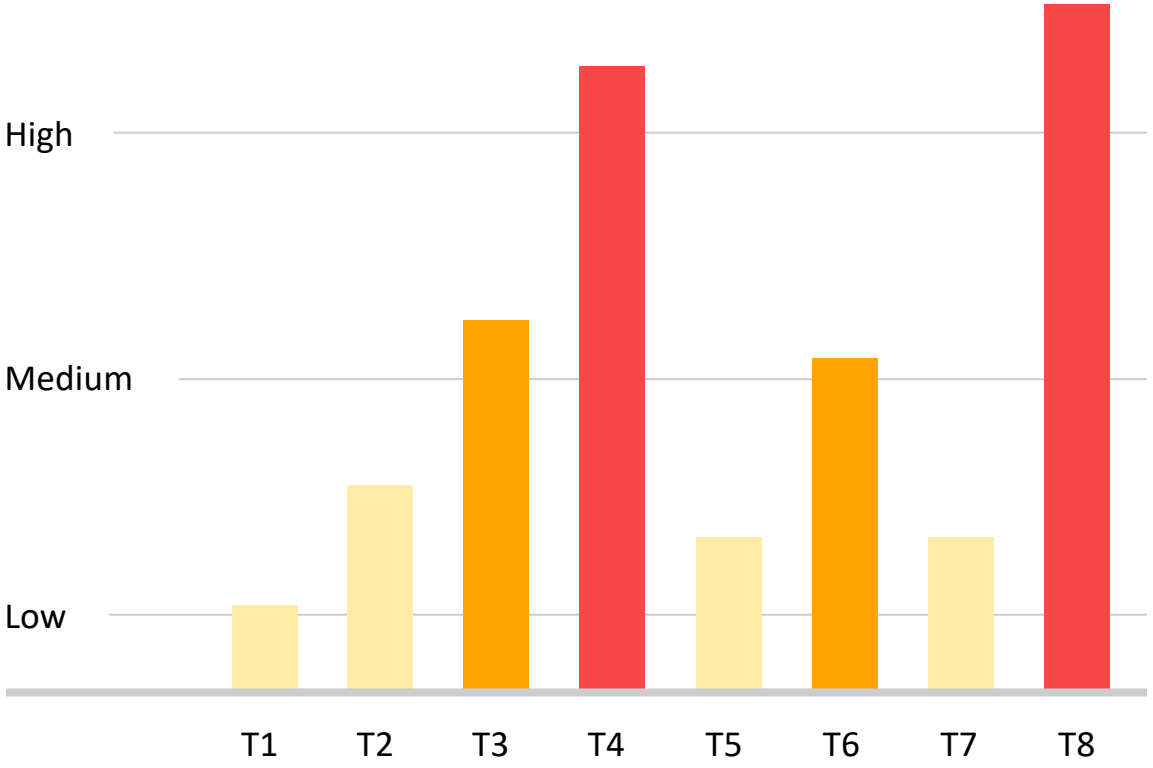
# Step 2: Predictive Test Selection

- ML model predicts useful subset to run
  - Based on information about changes
  - Of $10^5$ changes/mo, 1% is used to train the model
- Impact
  - Only a third of tests are selected
  - Misses just 0.1% of broken changes
  - AWS cost is cut by half

# This is Useful Beyond BigCo

- Predicting the probability of a test failure have many uses
- Situation
  - You wait for 1hr for CI to clear your pull request
  - Your integration tests only run nightly

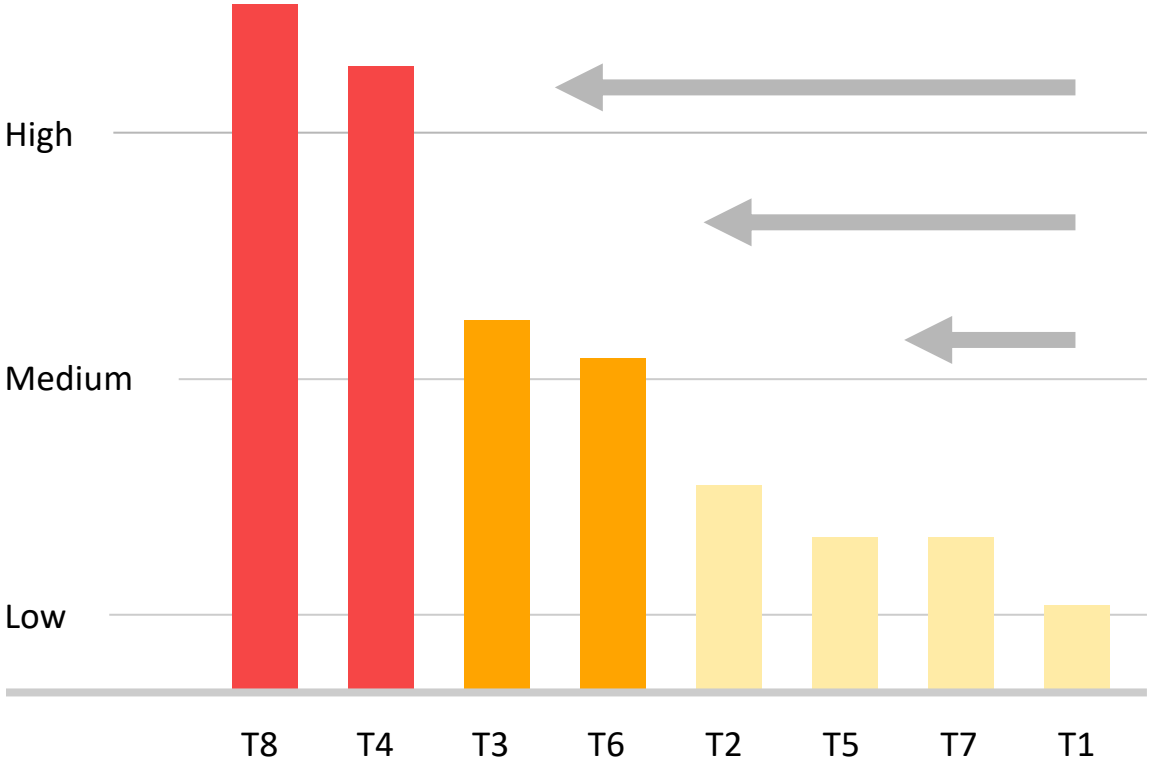
# Likelihood of failure




## What if you could run failing tests first?

Test that are in a random order are not optimized for a low Time to First Failure.

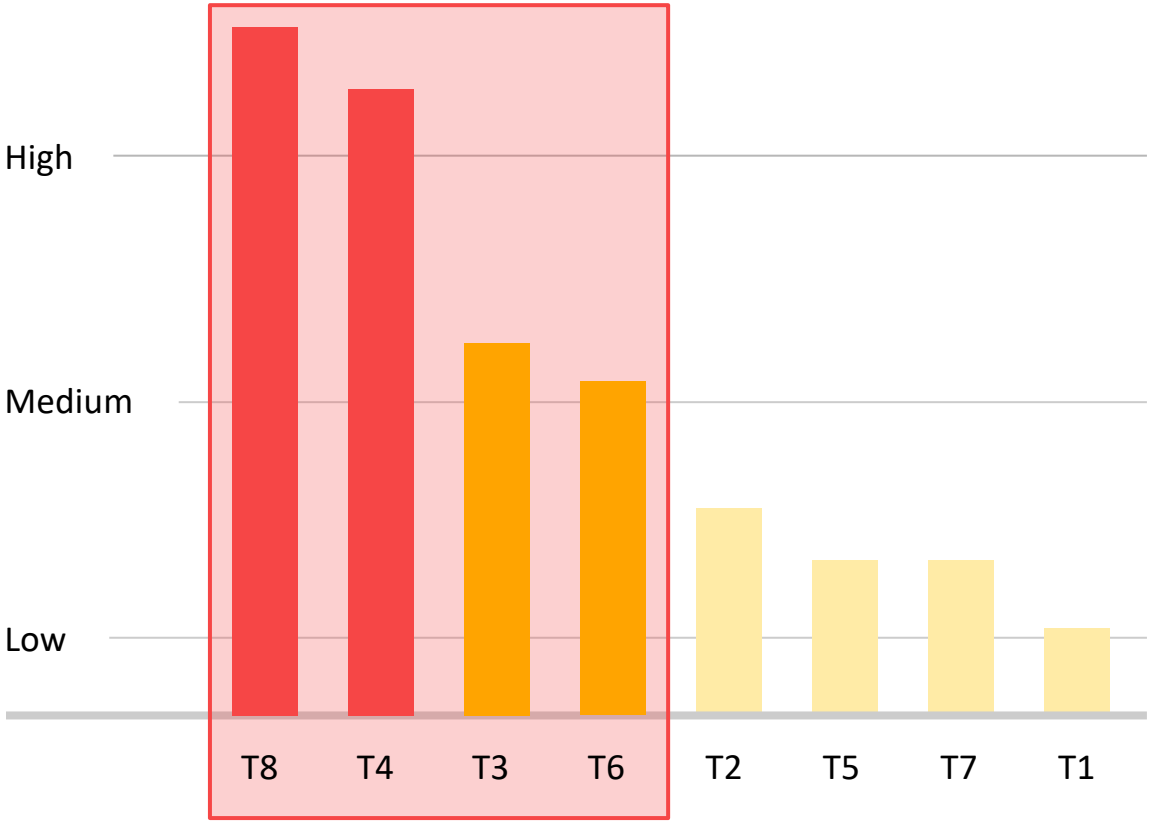
# Reordering tests




## Reducing Time to First Failure (TTFF)

By running tests that are likely to fail first we can reduce the time to first failure by a significant margin.

# Creating an Adaptive Run




## What if you could run a subset of your tests?

Reduce the time to get feedback by running a subset on a more frequent interval.

**At Launchable I am looking more into this.**
**If this interests you:**

**https://launchableinc.com/**

# Deployment Risk Prediction

- Situation
  - You are the SRE team in a BigCo
  - You oversee 100s of apps
  - ~1 deployment/app/day
- Questions
  - Can we flag risky deployments beforehand?

# What they have done

- Train model
  - With 40,000 deployments of which 100 are failures
  - Attributes: app names, commit messages, …
- Impact
  - Predict 99% of failures
  - 5% false alarm rate

# What they have done

- Learning
  - Most outages are estimated as “low risk” by developers
  - Most outages had short time span till approval
  - Long-maintained code is more risky
- Imagine what you can do with this!
  - Require somebody be on call
  - Restrict window of deployment

# Donkeys

- Different teams are doing things differently
- DevOps team ends up being glorified IT, or internal PS
- App teams feel like bullshit is imposed on them

# Unicorns

- Everyone does things one way
- DevOps team is autonomous
- App teams feel like bullshit is taken care of
- Positive feedback loop makes the one way the best way

# Discussion

- Well-funded DevOps team helps. How does that happen?
- Getting culture right earlier & succeeding helps
- Forget brown field and focus on green field?
- Latch on to cloud migration or microservice transition

# Conclusions

- Automation is table stake
- Using data from automation to drive progress isn't
  - Lots of low hanging fruits there
- Unicorns are using "big data" effectively
  - How can the rest of us get there?