pickyourtrail
unwrap the world

Scaling Engineering @ PYT
Vinayak - VP of Engg

Mar 2020

Agenda

- Introduction
- PYT Architecture & Tools
- How Tech Evolved
- Challenges & Solutions
- Next Steps

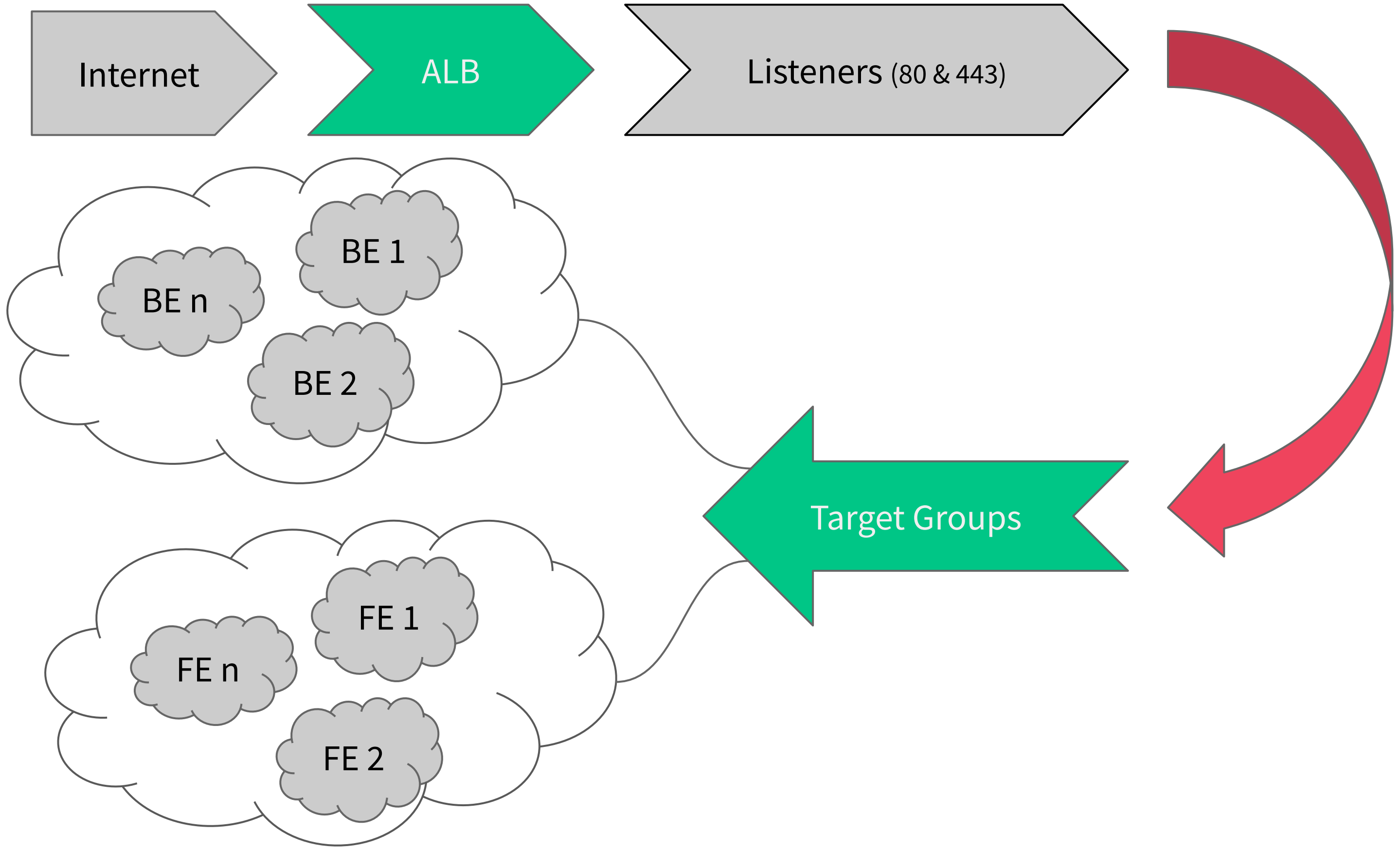
Introduction

- Worked with Fortune 500 companies like MSFT, MS to startups like ZoomIn (Head of Engineering) and now Pickyourtrail
- Enjoy Solving problems
- Scale Engineering with tools & processes
- Certified Heartfulness Meditation Coach - 4+ years

PYT Architecture & Tools

aws
Java
php
jQuery
appium
mongoDB
MySQL
Site24x7

Production Setup



How Tech Evolved - beginning

- 12 devs, 2 QA members
- Sprints got just started
- Requirements were missed
- Code merge/push issues
- Spent more time on fixing bugs (Less time on features)
- Untested code
- Test cases were evolving
- No differentiation between Regression and New Issue
- 3-5 builds per week

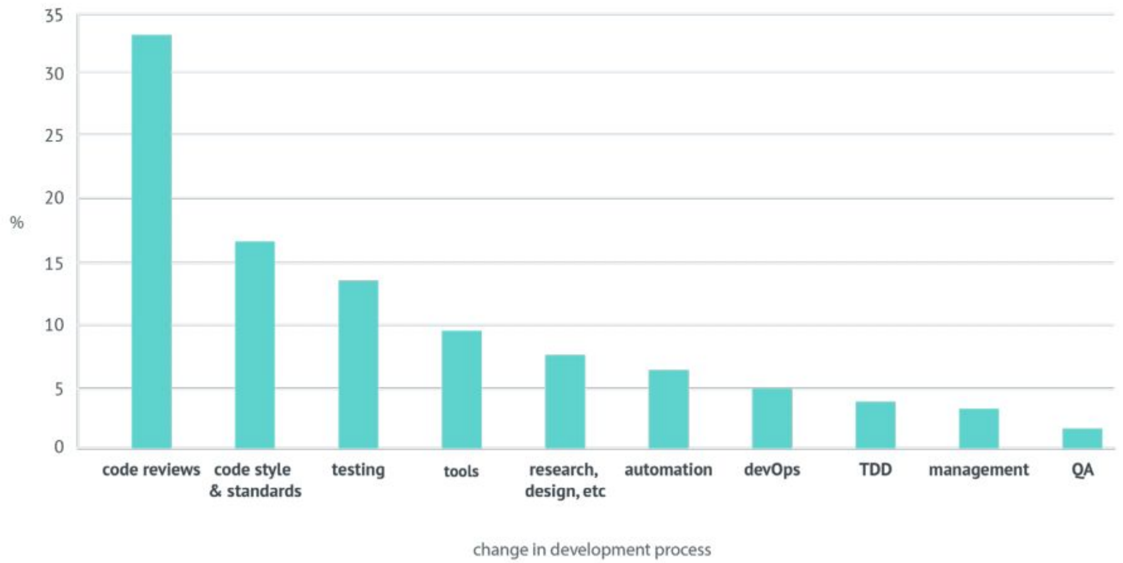
How Tech Evolved - Figure Why?

- On the ground
- Root cause issues
- Understood the challenges
- Started to track bugs more rigorously
- Adhere to scope

What we found?

Code Quality

What change in your development process had the biggest impact on code quality?*



How Tech Evolved - Code Quality

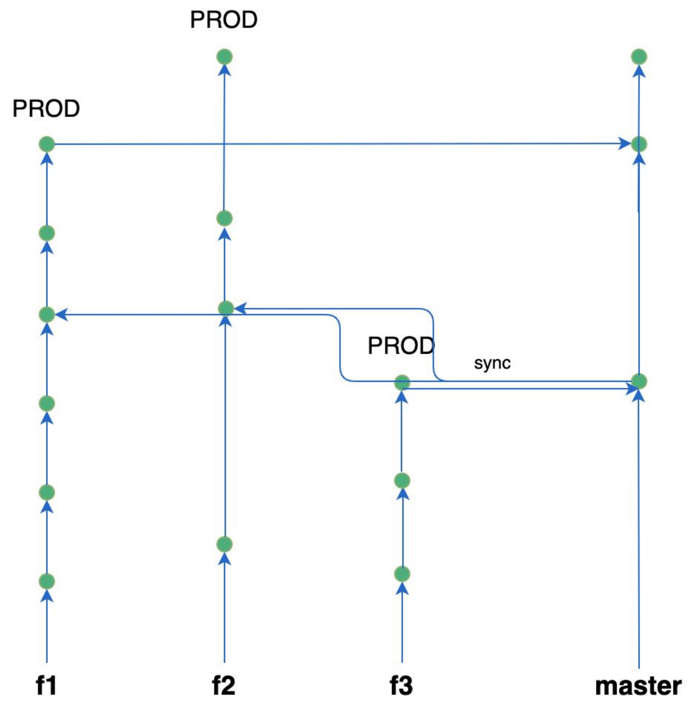
Peer review – an activity in which people other than the author of a software deliverable examine it for defects and improvement opportunities – is one of the most powerful software quality tools available. Peer review methods include inspections, walkthroughs, peer deskchecks, and other similar activities. After experiencing the benefits of peer reviews for nearly fifteen years, I would never work in a team that did not perform them. - **Karl Wiegers**

… software testing alone has limited effectiveness – the average defect detection rate is only 25 percent for unit testing, 35 percent for function testing, and 45 percent for integration testing. In contrast, **the average effectiveness of design and code inspections are 55 and 60 percent**. Case studies of review results have been impressive: - **McConell**

How Tech Evolved - Code Branching

- Open Source PR model
- Developers create branches on their forks
- Create a branch on central and raise a PR
- PR is reviewed and closed
- Code is tested
- Deployed to live

How Tech Evolved - Code Branching

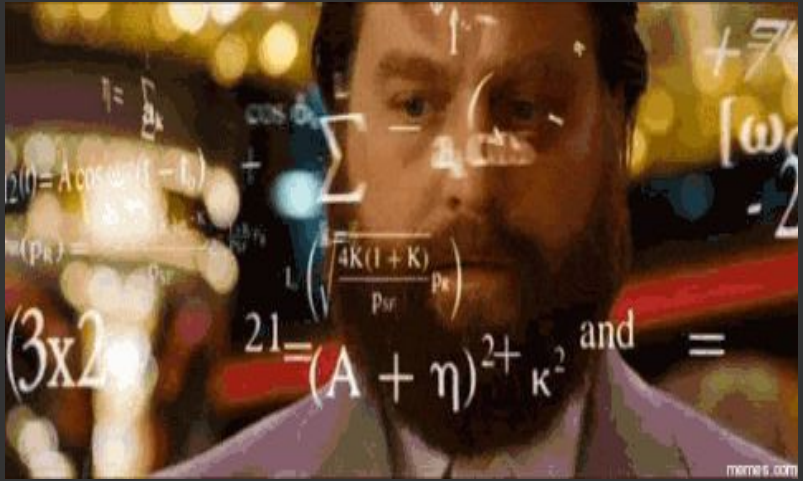


How Tech Evolved - Code Review

- Don’t skip code reviews — your team will be less productive if you do. Also, review code before deployment — not after.
- Make sure all of your developers get to review code from time to time, as this will make them feel empowered and improve their skills.
- Make sure your developers spend half a day to one day per week reviewing code — this is the sweet spot for time spent versus high code quality.
- Make code reviews blocking and don’t deploy before they have been carried out.
- Be strict and thorough while reviewing code — your code quality and velocity will thank you.

It worked

Did it scale?



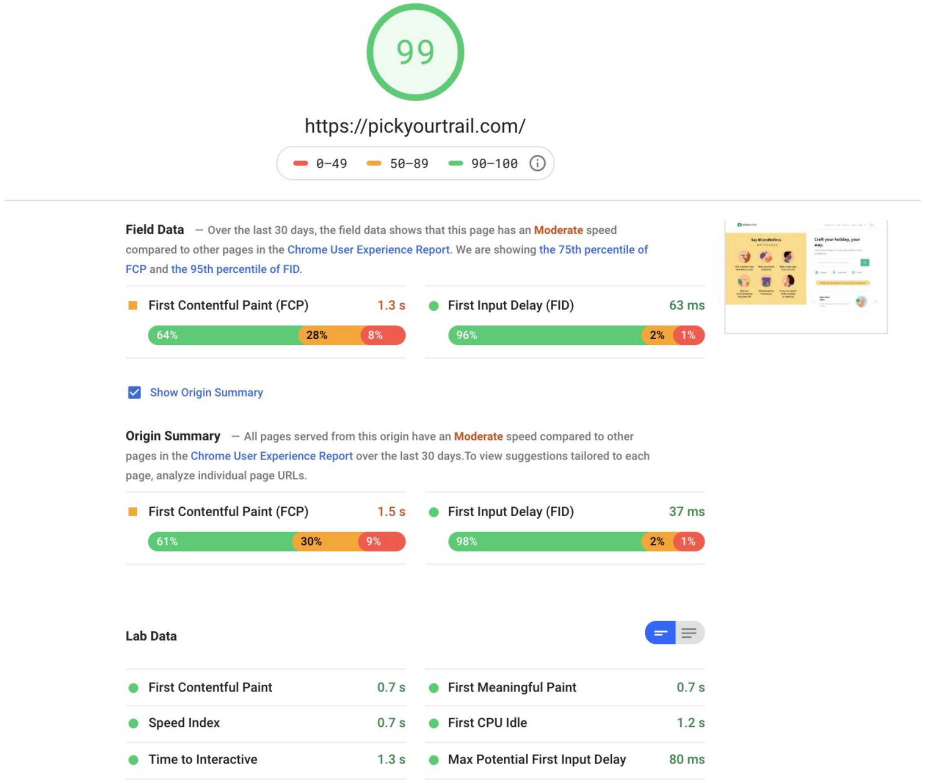
Challenges

- Team grew 3x
- Code Reviews - became slow
- Code was getting missed
- Not enough environments to test
- Regression took 6 hours

Solutions

- Automate deployments - ~20-25 builds per week
- Automation to check code misses
- QA Automation
  - Sanity
  - Regression
  - Url validations
- Automate environments
- Run regression
- Page speed automation

Pagespeed



99

https://pickyourtrail.com/

🟥 0–49 🟧 50–89 🟩 90–100 ⓘ

Field Data — Over the last 30 days, the field data shows that this page has an **Moderate** speed compared to other pages in the Chrome User Experience Report. We are showing the 75th percentile of FCP and the 95th percentile of FID.

First Contentful Paint (FCP)	1.3 s	First Input Delay (FID)	63 ms
64% / 28% / 8%		96% / 2% / 1%	

✓ Show Origin Summary

Origin Summary — All pages served from this origin have an **Moderate** speed compared to other pages in the Chrome User Experience Report over the last 30 days.To view suggestions tailored to each page, analyze individual page URLs.

First Contentful Paint (FCP)	1.5 s	First Input Delay (FID)	37 ms
61% / 30% / 9%		98% / 2% / 1%	

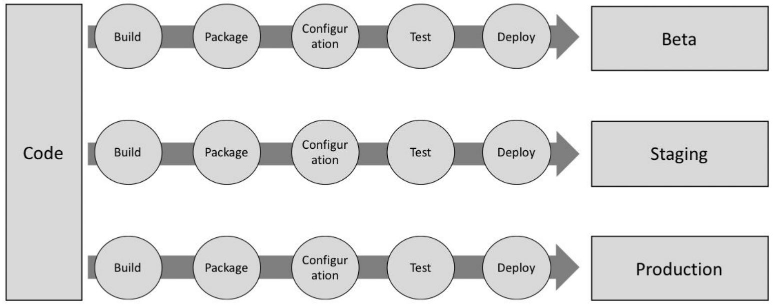
Lab Data

First Contentful Paint	0.7 s	First Meaningful Paint	0.7 s
Speed Index	0.7 s	First CPU Idle	1.2 s
Time to Interactive	1.3 s	Max Potential First Input Delay	80 ms

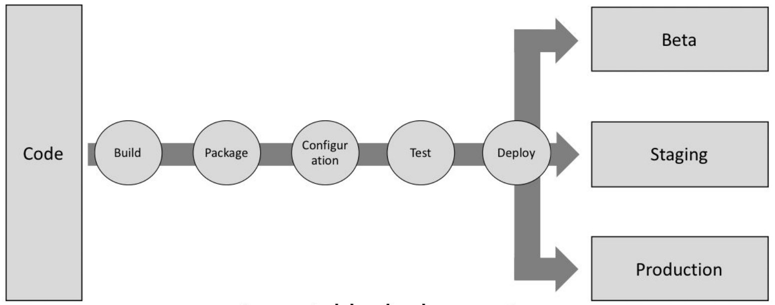
Next Steps

- Immutable Infra
- More automation
- Improve CI and leverage CD
- Better Tech Debt Management
- More unit tests coverage
- Api Automation
- Mobile App Automation
- N+1 model

Next Steps

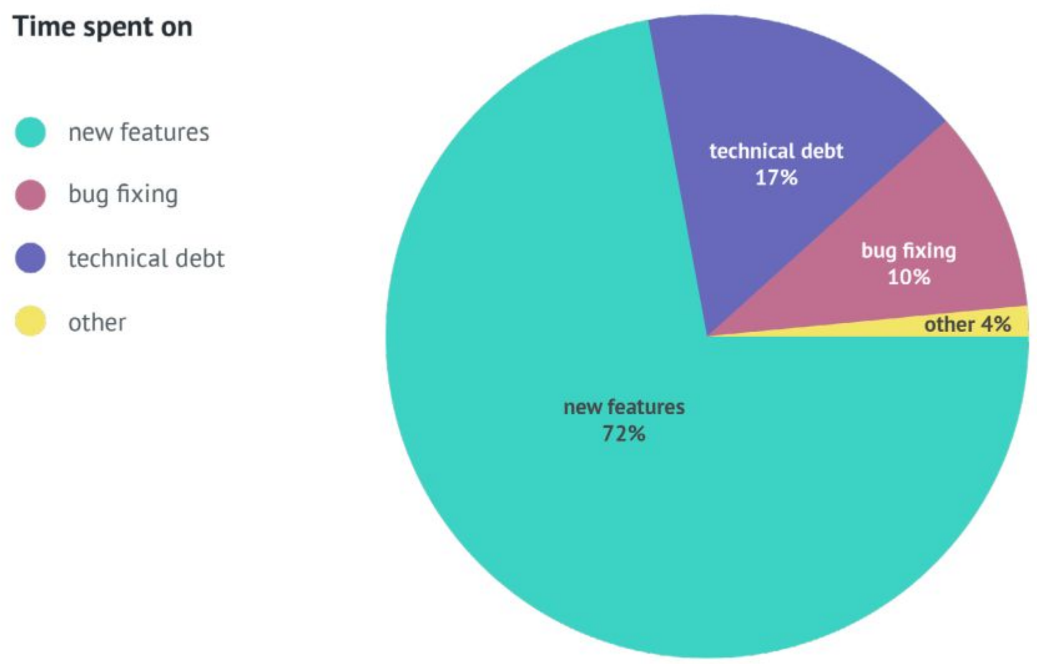


Mutable deployments



Immutable deployments

Next Steps


Time spent on
new features
bug fixing
technical debt
other
technical debt 17%
bug fixing 10%
other 4%
new features 72%

References

- https://medium.com/clarifai-champions/99-pr-oblems-a-beginners-guide-to-open-source-abc1b867385a
- https://paulhammant.com/2013/04/05/what-is-trunk-based-development/
- https://blog.logrocket.com/the-git-workflow-you-need-how-to-deal-with-multiple-teams-in-a-single-repository-faf5bb17a6e4/
- https://martinfowler.com/articles/itsNotJustStandingUp.html
- https://www.codacy.com/ebooks/guide-to-code-reviews-II
- https://medium.com/@adhorn/immutable-infrastructure-21f6613e7a23

Thanks!