GTAC

KIRKLAND 2014

Move Fast & Don't Break Things

Ankit Mehta, Google

MOVE Fast & DON'T Break THINGS

Ankit Mehta, Google

GTAC 2014

Why am I talking @ GTAC

- Sharing my experience of a decade @ Google in Test Engineering
- Passionate about balance between velocity and quality
- Taking an opportunity to share Google’s take at balancing velocity and quality

Interesting side projects @ Google:

MAD (Millions of Automated Documents)

Scale Google’s Bug database

Surveytool

Microprocessor controlled pick n place manipulator (school)

Test Engineering

"what is our purpose?"

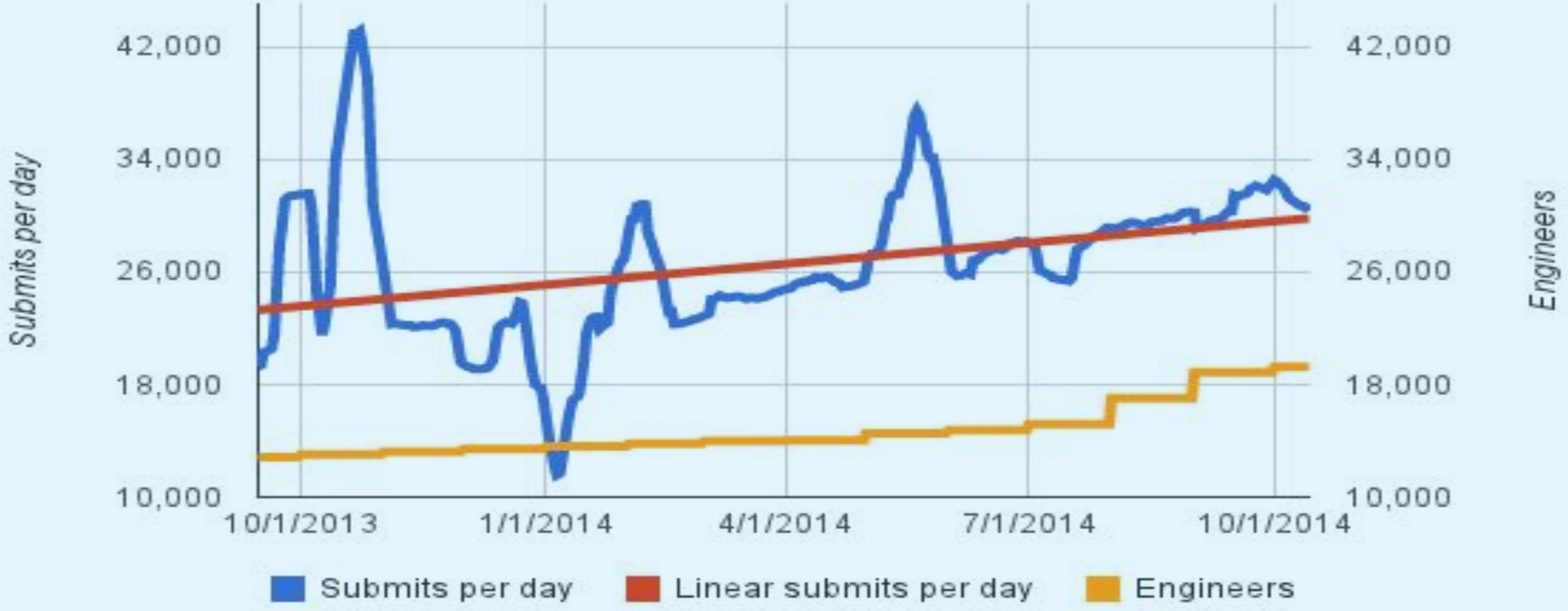
Build world class infrastructure to launch high quality innovative products fast that delight our users

Google from the outside..

Google from the inside…

More code

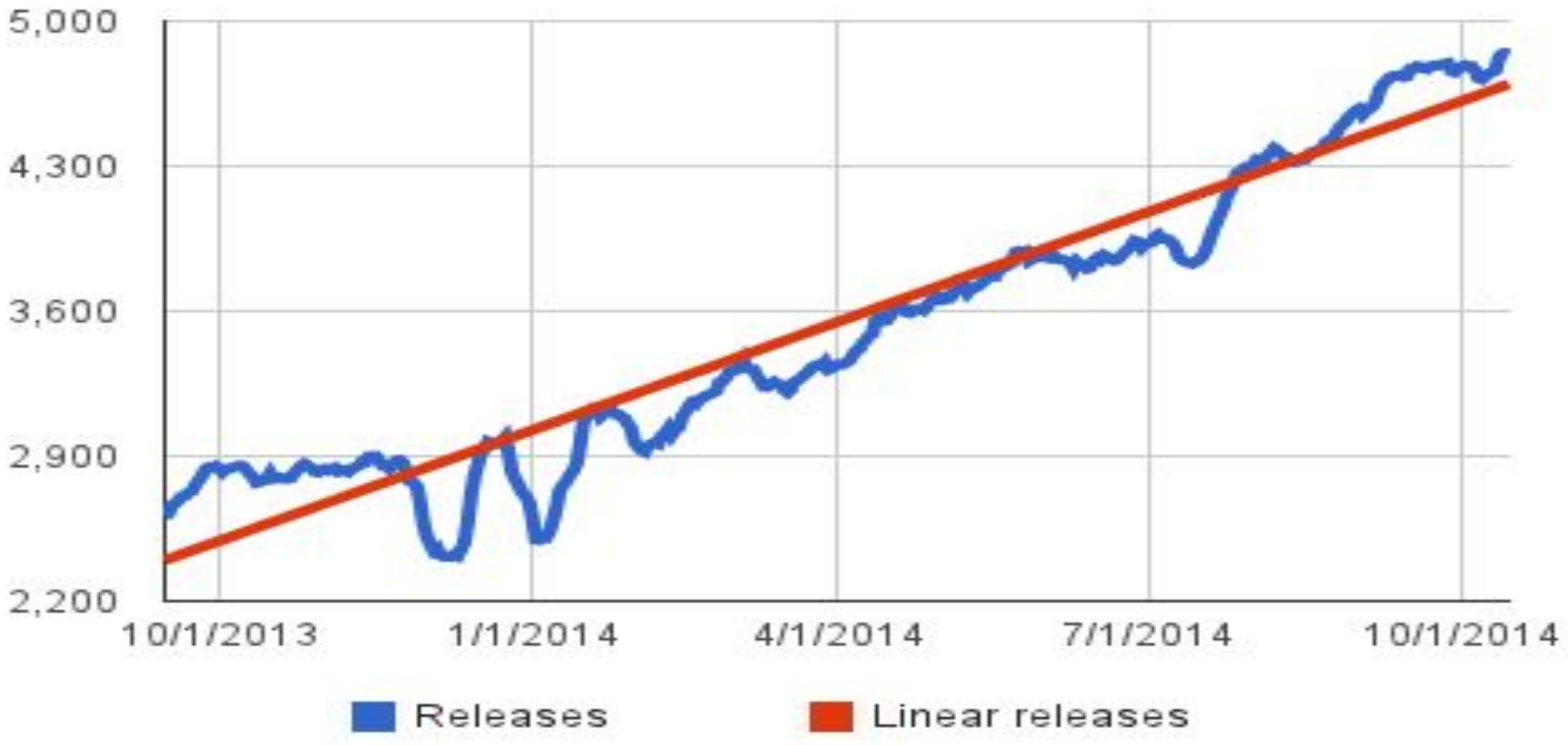
- 30K check-ins per day
- A check-in every 3s!

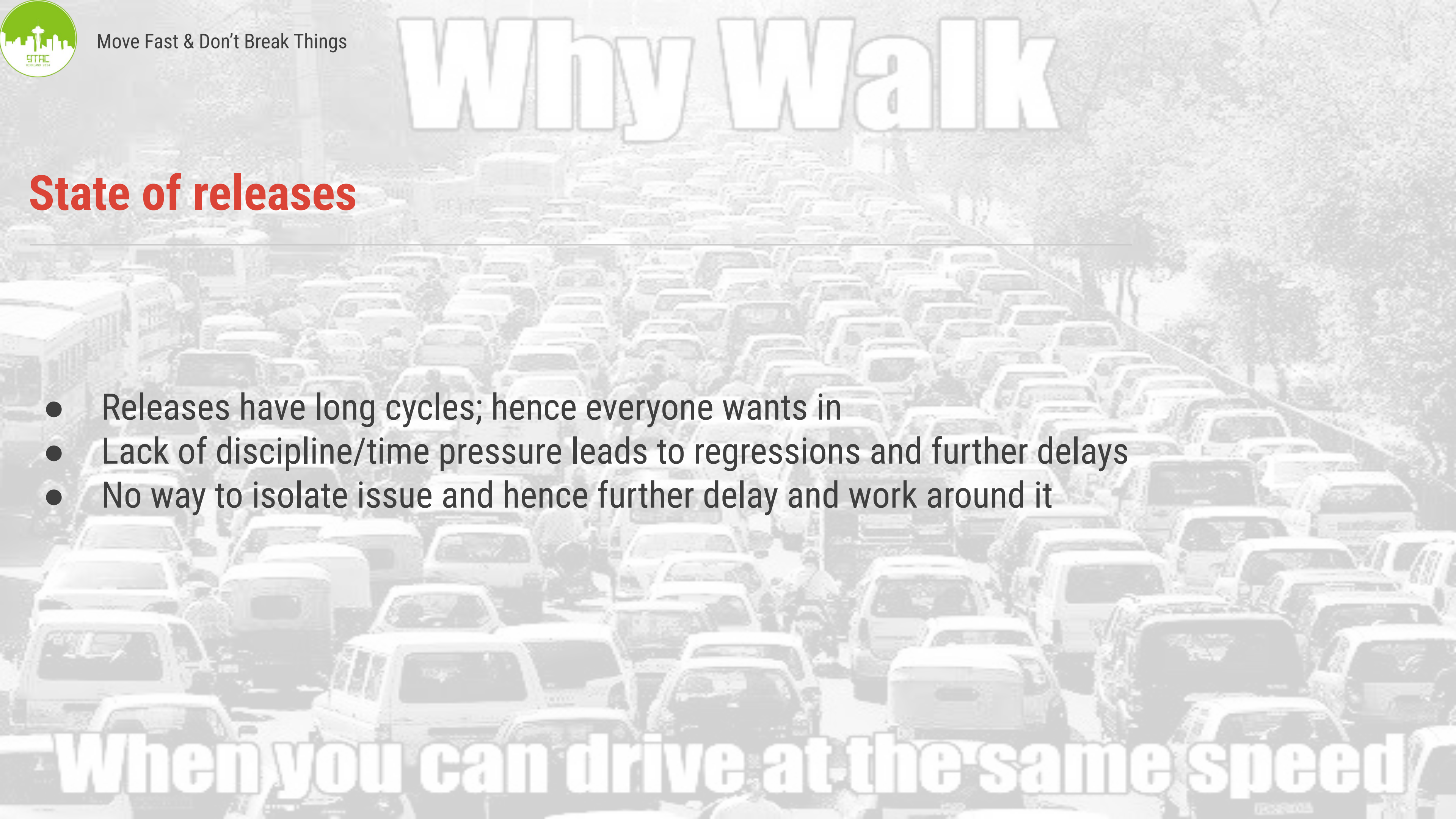


Move Fast & Don't Break Things

More releases

- 2x more releases





State of releases

- Releases have long cycles; hence everyone wants in
- Lack of discipline/time pressure leads to regressions and further delays
- No way to isolate issue and hence further delay and work around it

Moving Fast is Good!

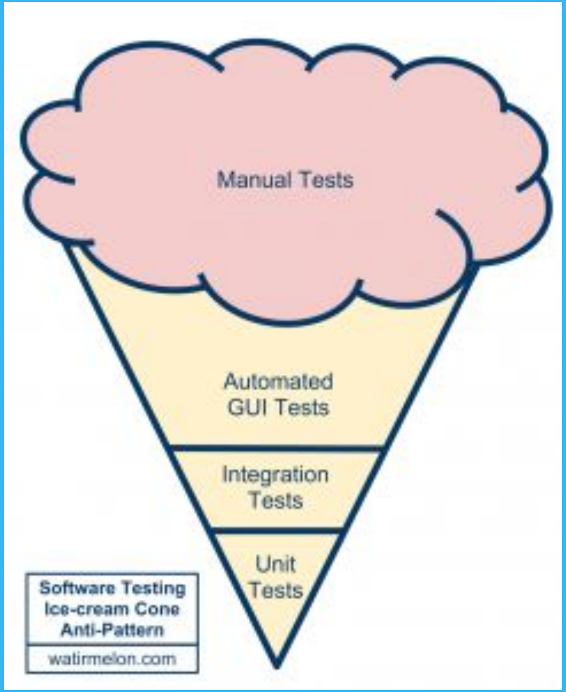
- Innovate
- Address flaws quickly
- Better productivity
- Better Code Health

… But Breaking things isn’t

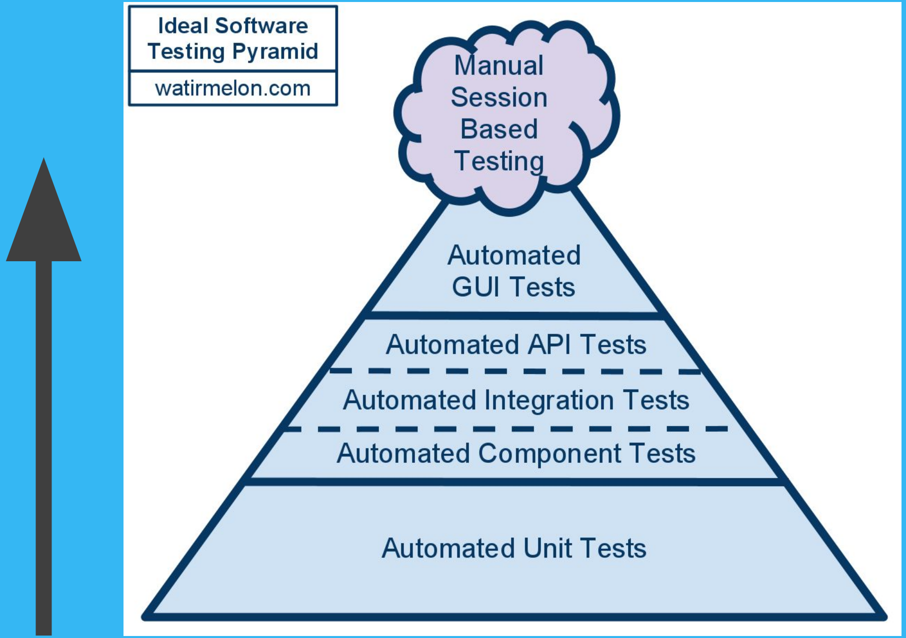
- User trust/satisfaction
- Uphold the brand
- Launch products
- Set a high bar

My Testing Philosophy

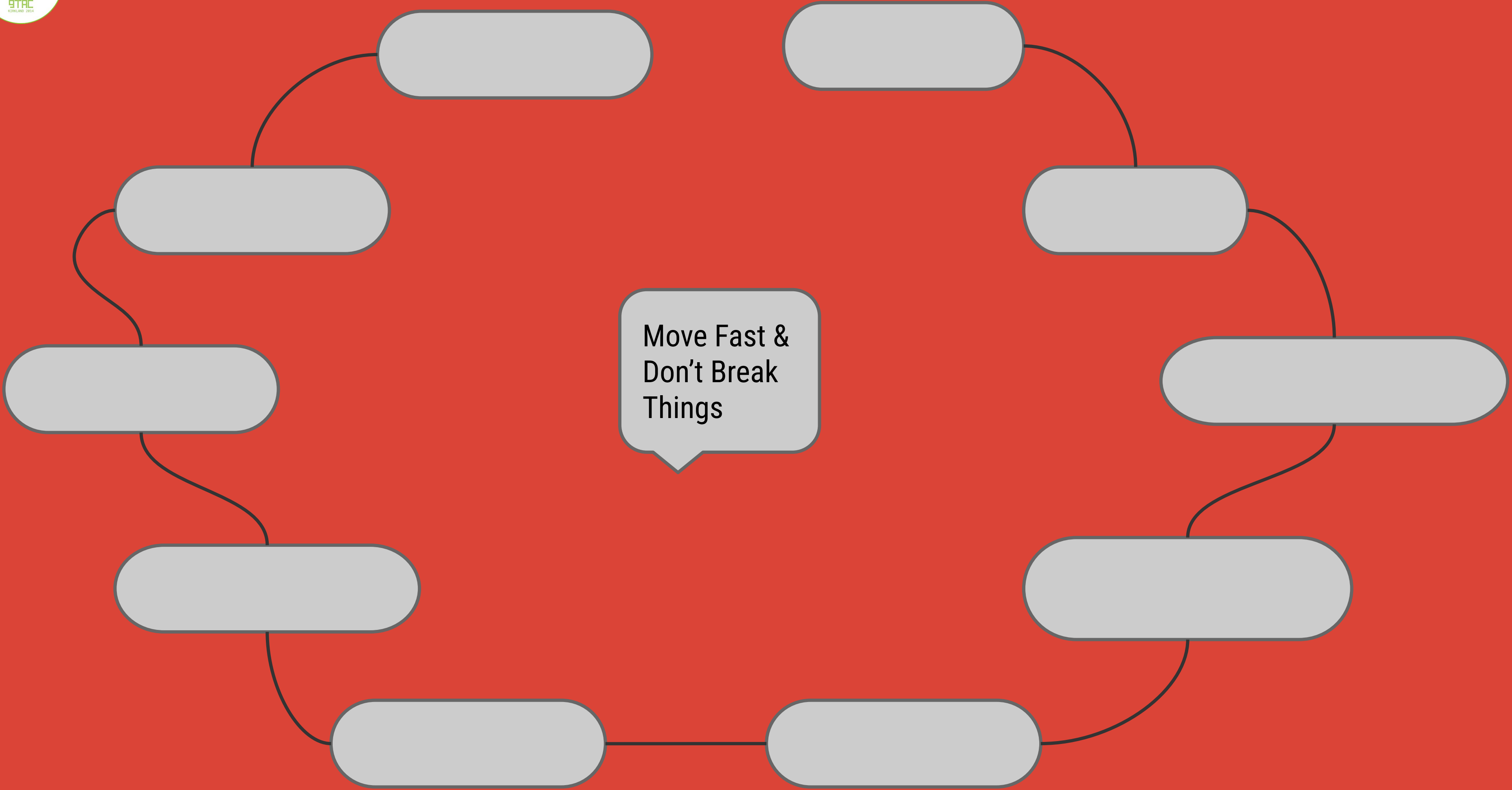
What many teams do

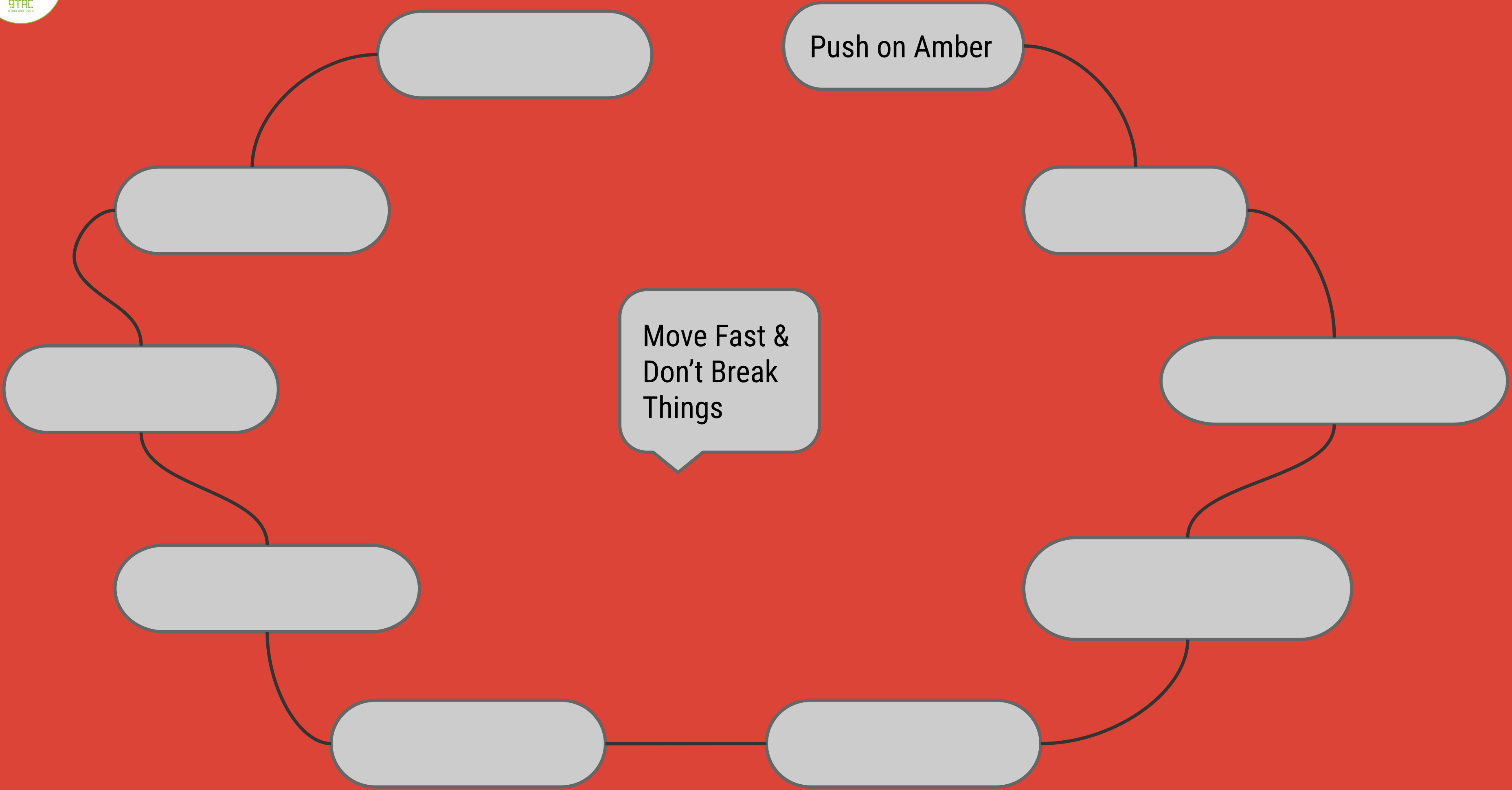


How it should be done


Maintenance
Slower tests
Flakiness
Ideal Software Testing Pyramid
watirmelon.com
Manual Session Based Testing
Automated GUI Tests
Automated API Tests
Automated Integration Tests
Automated Component Tests
Automated Unit Tests

Move Fast & Don't Break Things



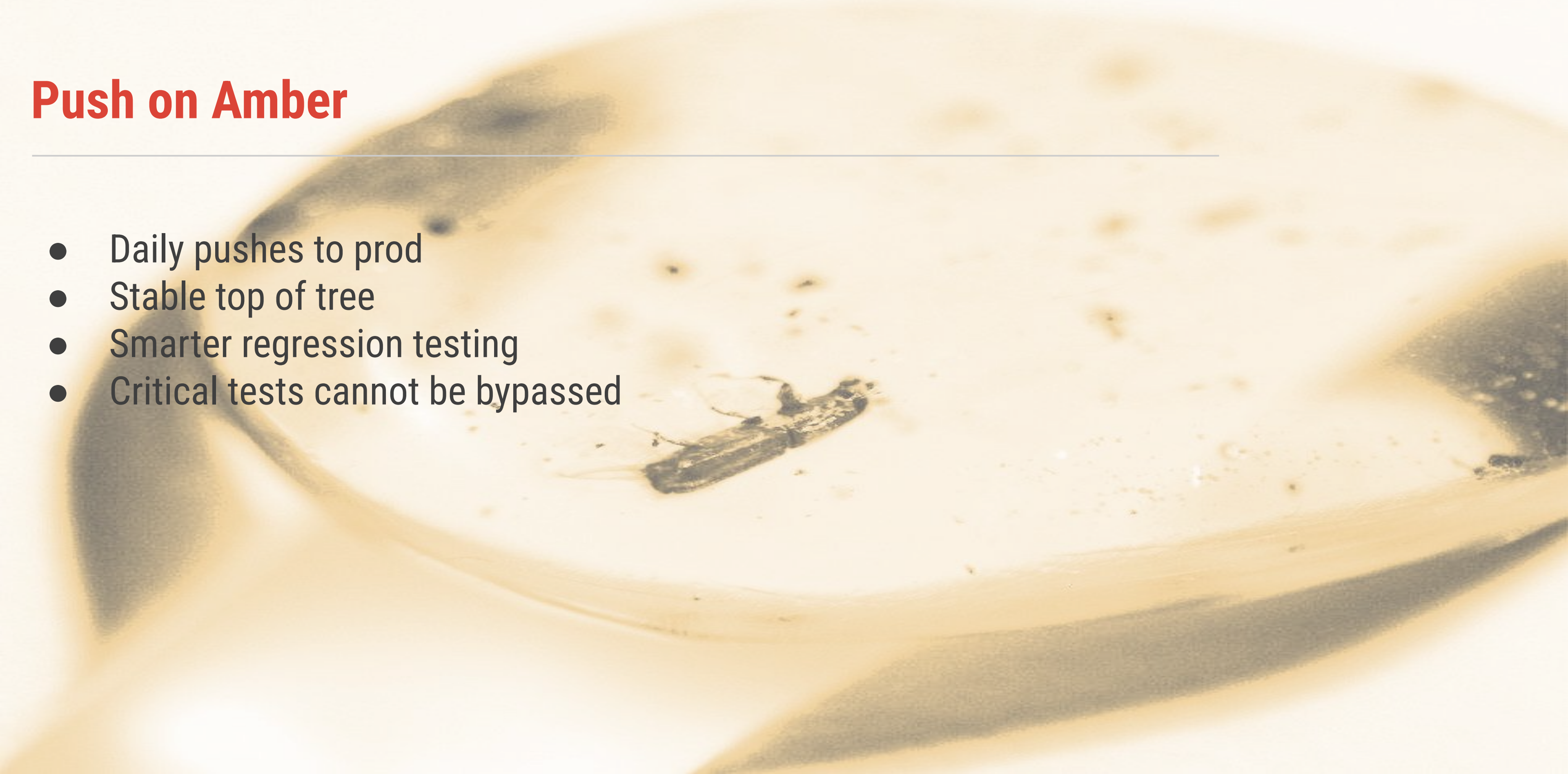

Push on Amber
Move Fast & Don’t Break Things

Push on Green..

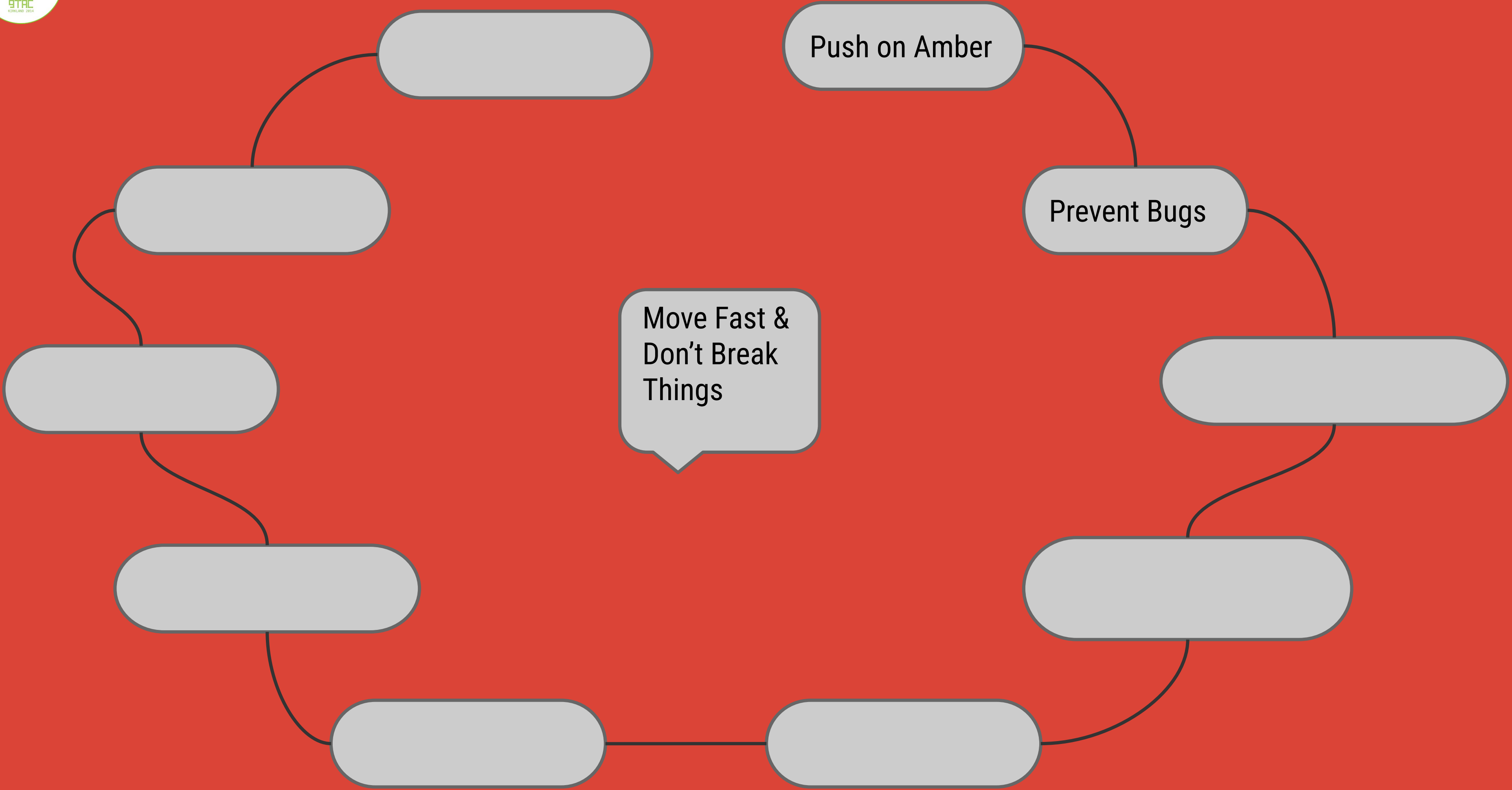
“As soon as test suites go green, deployment to production is automatically started”

This has evolved at Google. We have tens of thousands of tests for some projects, some of which could be failing and/or flaky

Push on Amber



- Daily pushes to prod
- Stable top of tree
- Smarter regression testing
- Critical tests cannot be bypassed


Push on Amber
Prevent Bugs
Move Fast & Don't Break Things

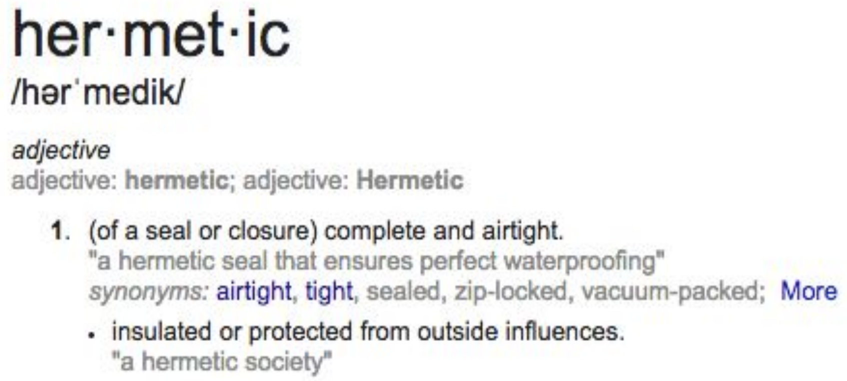
Prevent Bugs

- Prevent bugs and not catch them
- Deterministic hermetic tests
- Prevent bad code from getting in
- High presubmit coverage and usage

What is a Hermetic Test?

The short definition would be a “*test in a box*”.

My version: run a test while on a airplane *without network



her·met·ic

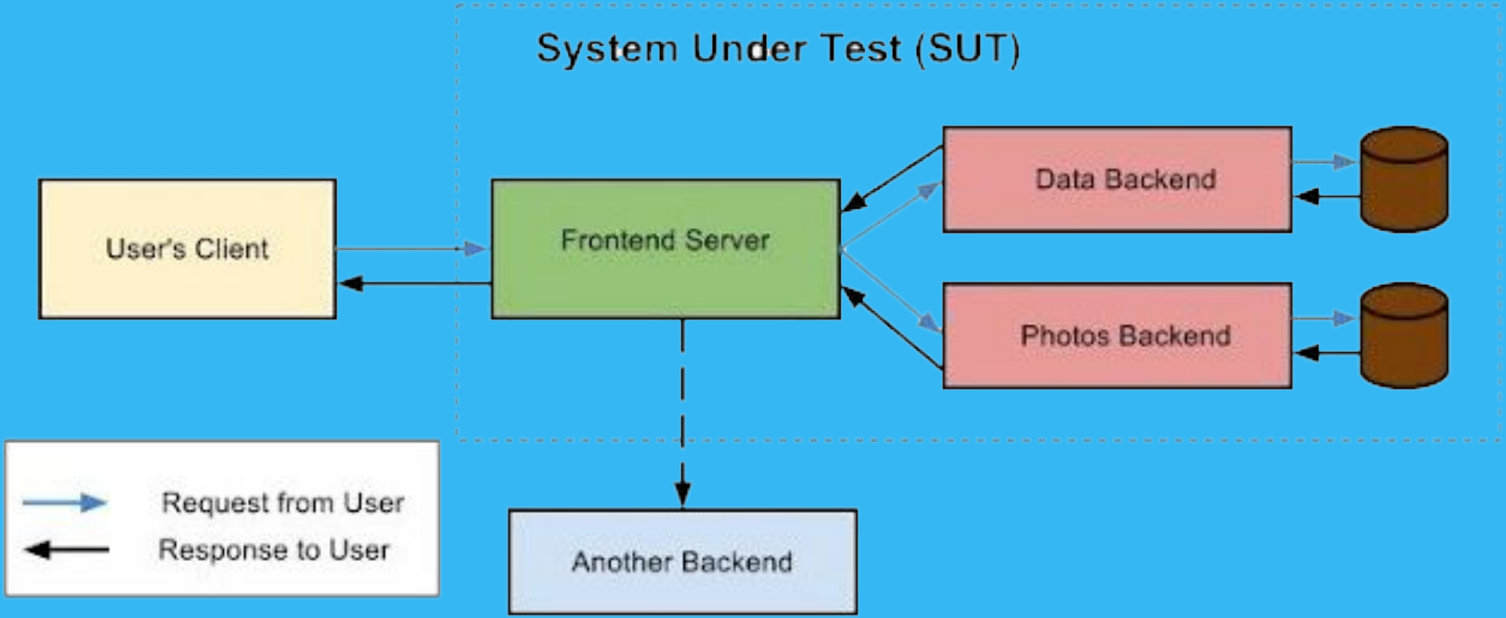
/hərˈmedik/

adjective

adjective: **hermetic**; adjective: **Hermetic**

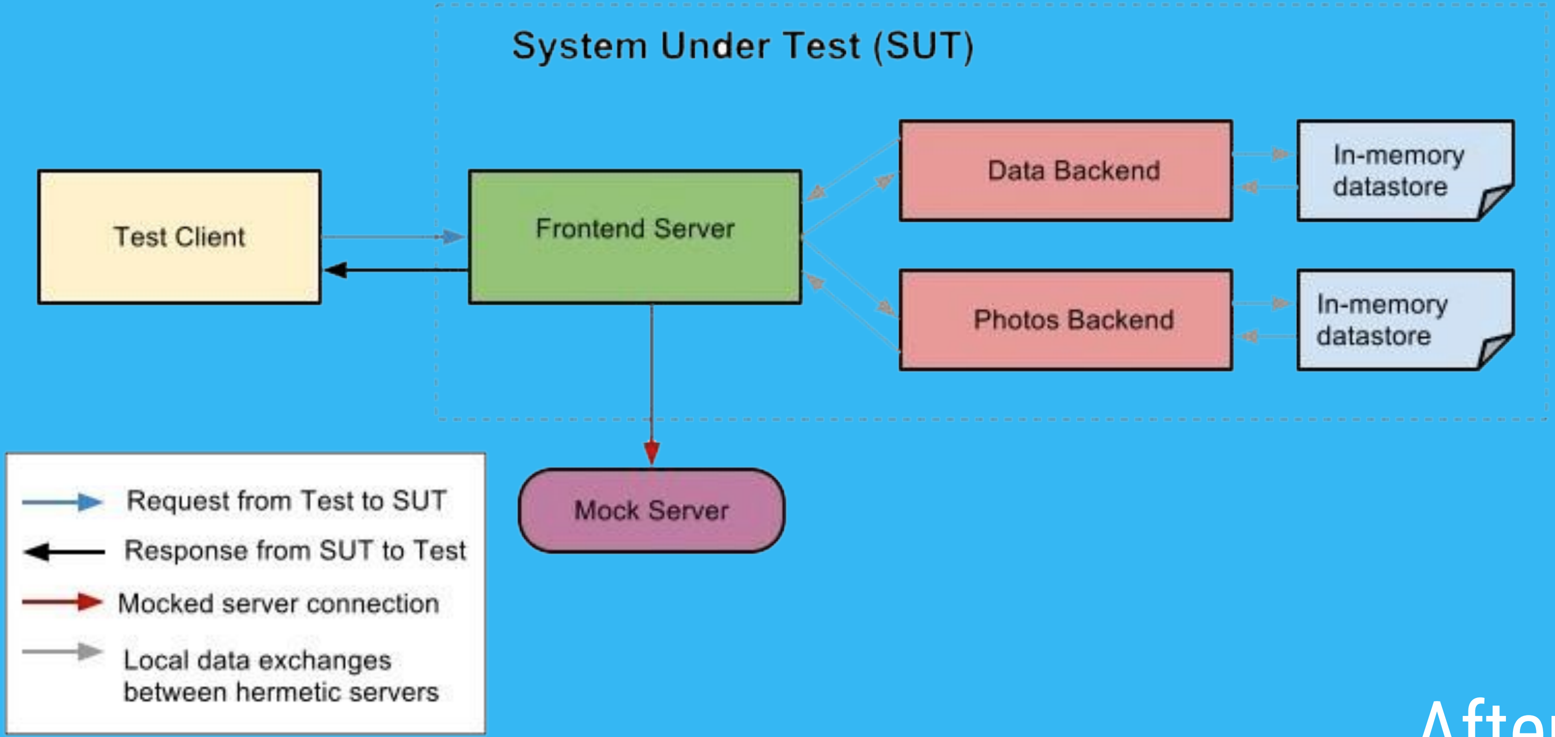
1. (of a seal or closure) complete and airtight.
   "a hermetic seal that ensures perfect waterproofing"
   synonyms: airtight, tight, sealed, zip-locked, vacuum-packed; More
   - insulated or protected from outside influences.
     "a hermetic society"

Non-Hermetic Servers



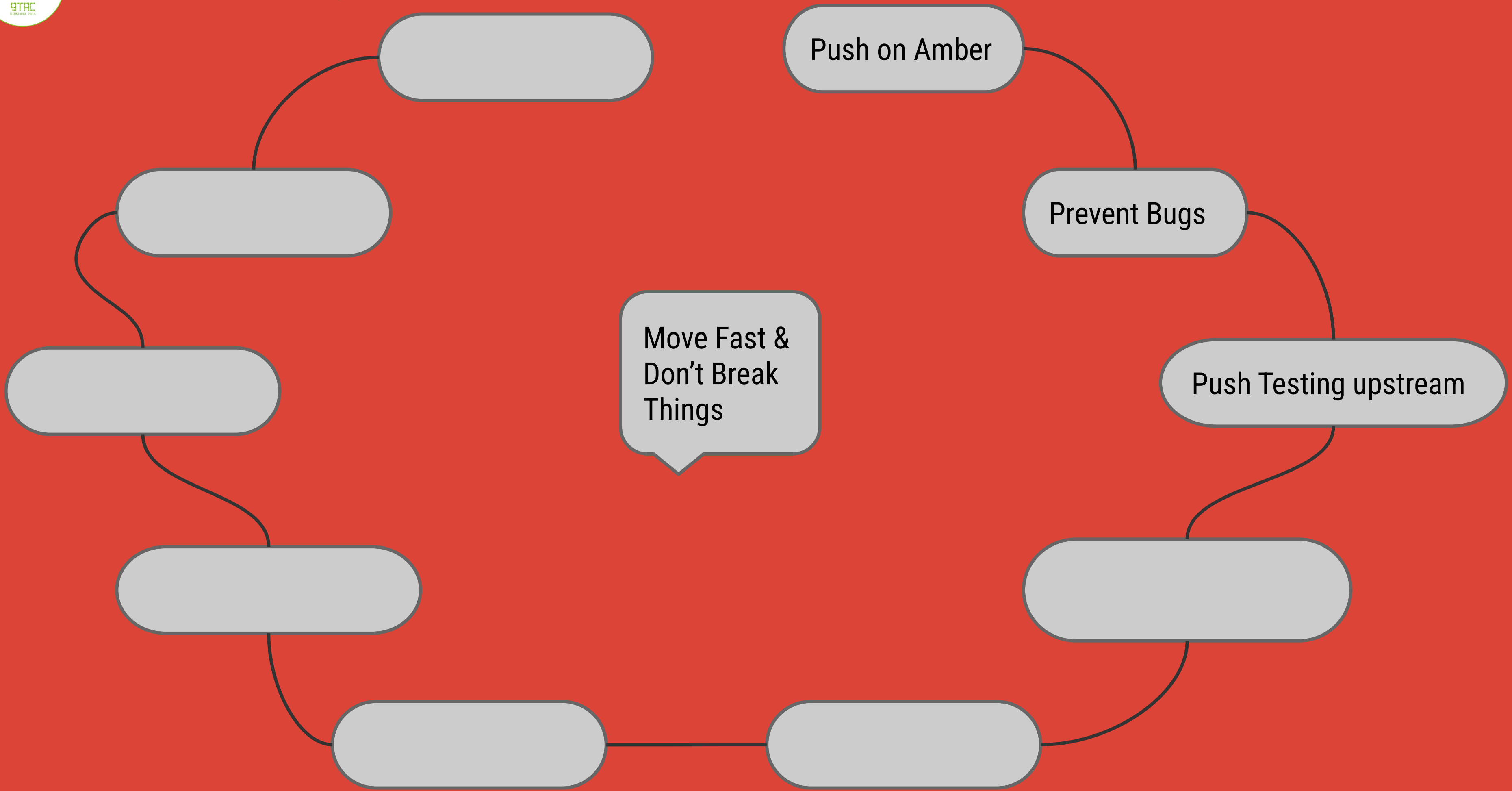
Before

Hermetic Servers

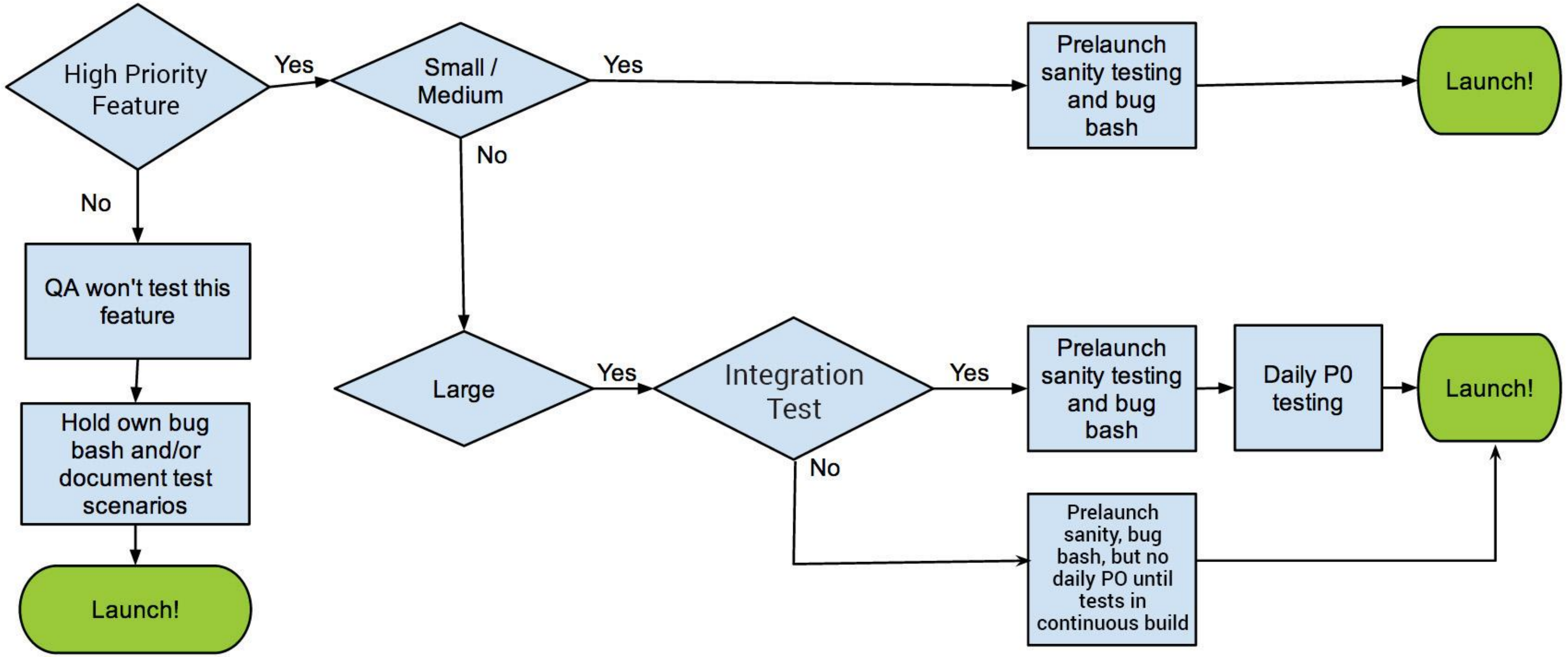


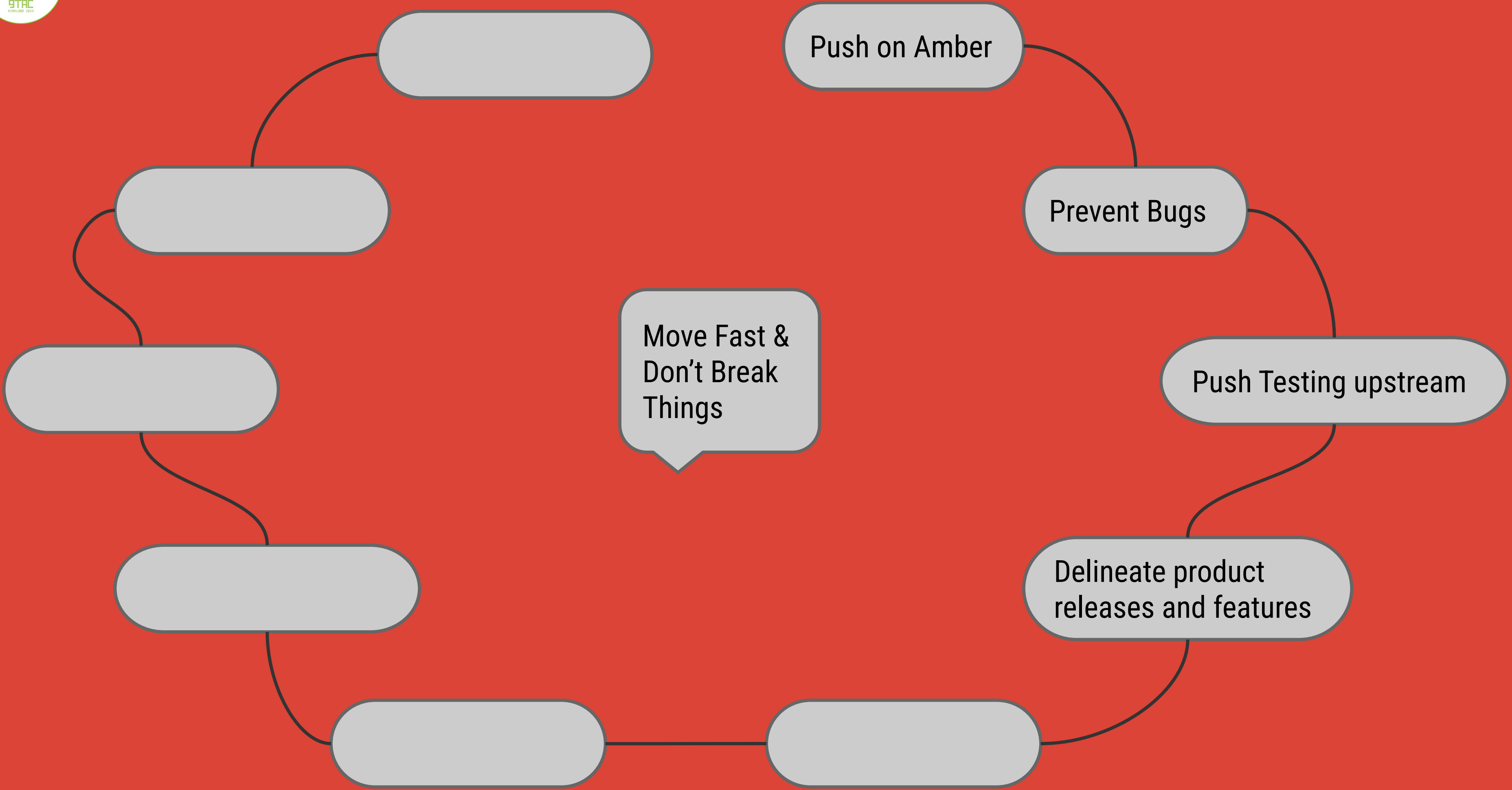
After

Move Fast & Don’t Break Things



Pushing Testing Upstream

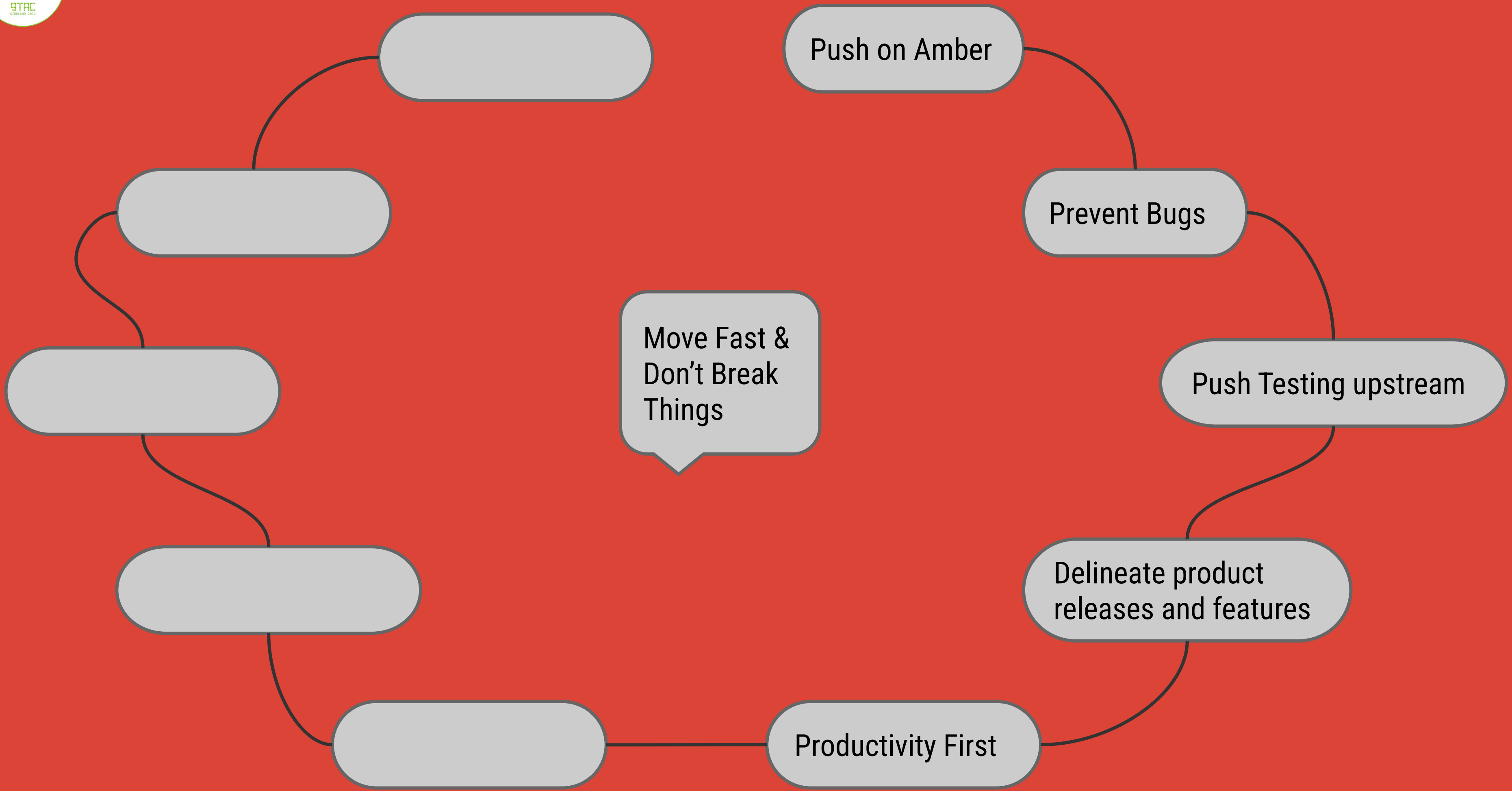

High Priority Feature
Yes
No
Small / Medium
Yes
No
Prelaunch sanity testing and bug bash
Launch!
QA won't test this feature
Hold own bug bash and/or document test scenarios
Launch!
Large
Yes
Integration Test
Yes
No
Prelaunch sanity testing and bug bash
Daily P0 testing
Launch!
Prelaunch sanity, bug bash, but no daily P0 until tests in continuous build


Move Fast & Don't Break Things
Push on Amber
Prevent Bugs
Push Testing upstream
Delineate product releases and features

Delineate product releases and features

- Releases always keep rolling
- Dark launch features
- Revert features and not revert releases
- Launch releases without leaking features

Move Fast & Don't Break Things


Move Fast & Don't Break Things
Push on Amber
Prevent Bugs
Push Testing upstream
Delineate product releases and features
Productivity First

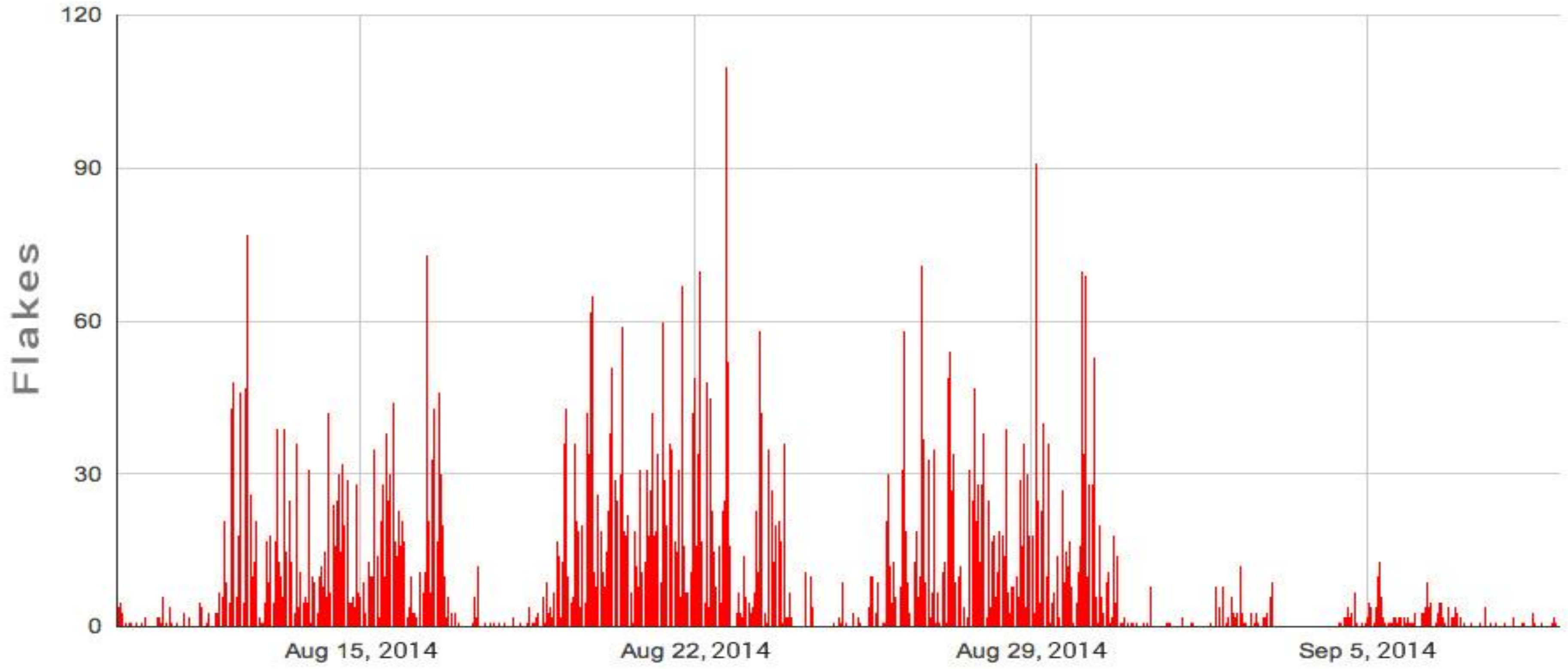
Productivity First..

- Invisible Tests
- Tests an asset and not a liability
- Fast UI Automation
- Zero tolerance on flakiness

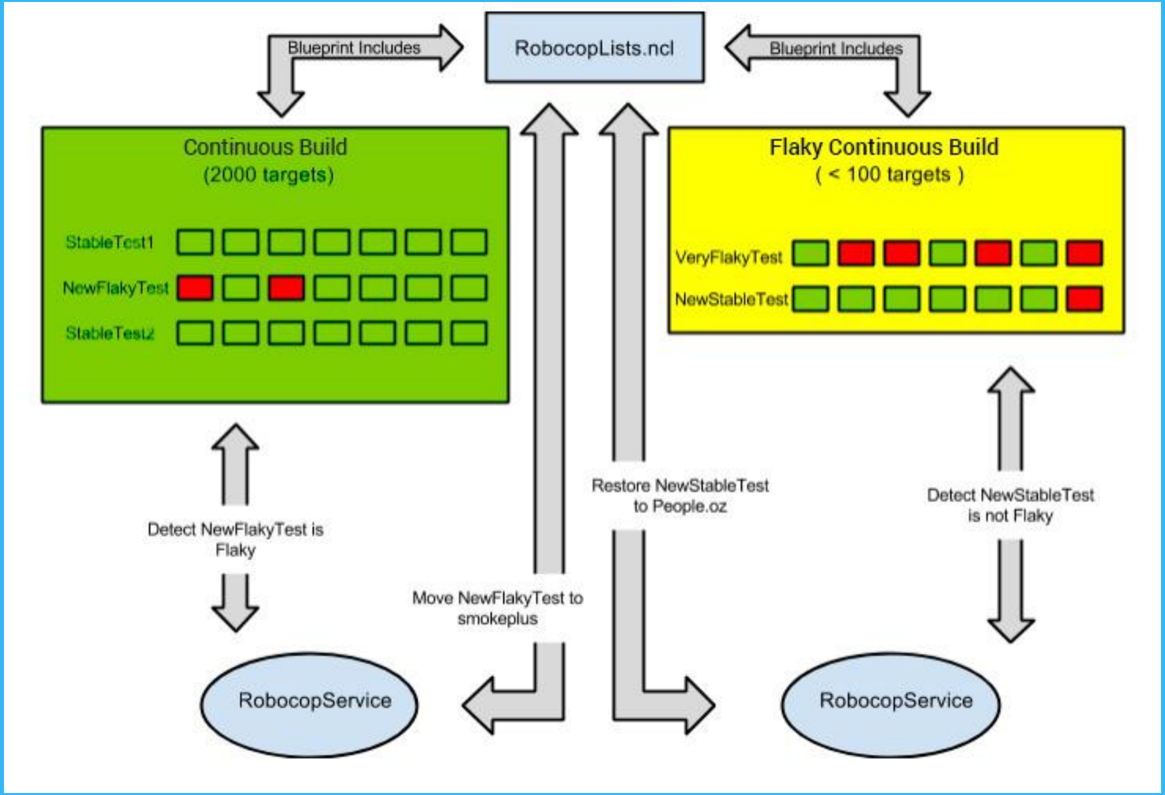
Attack Bad Tests

- Slow Tests
- Flaky Tests

Flaky tests are worse than no tests

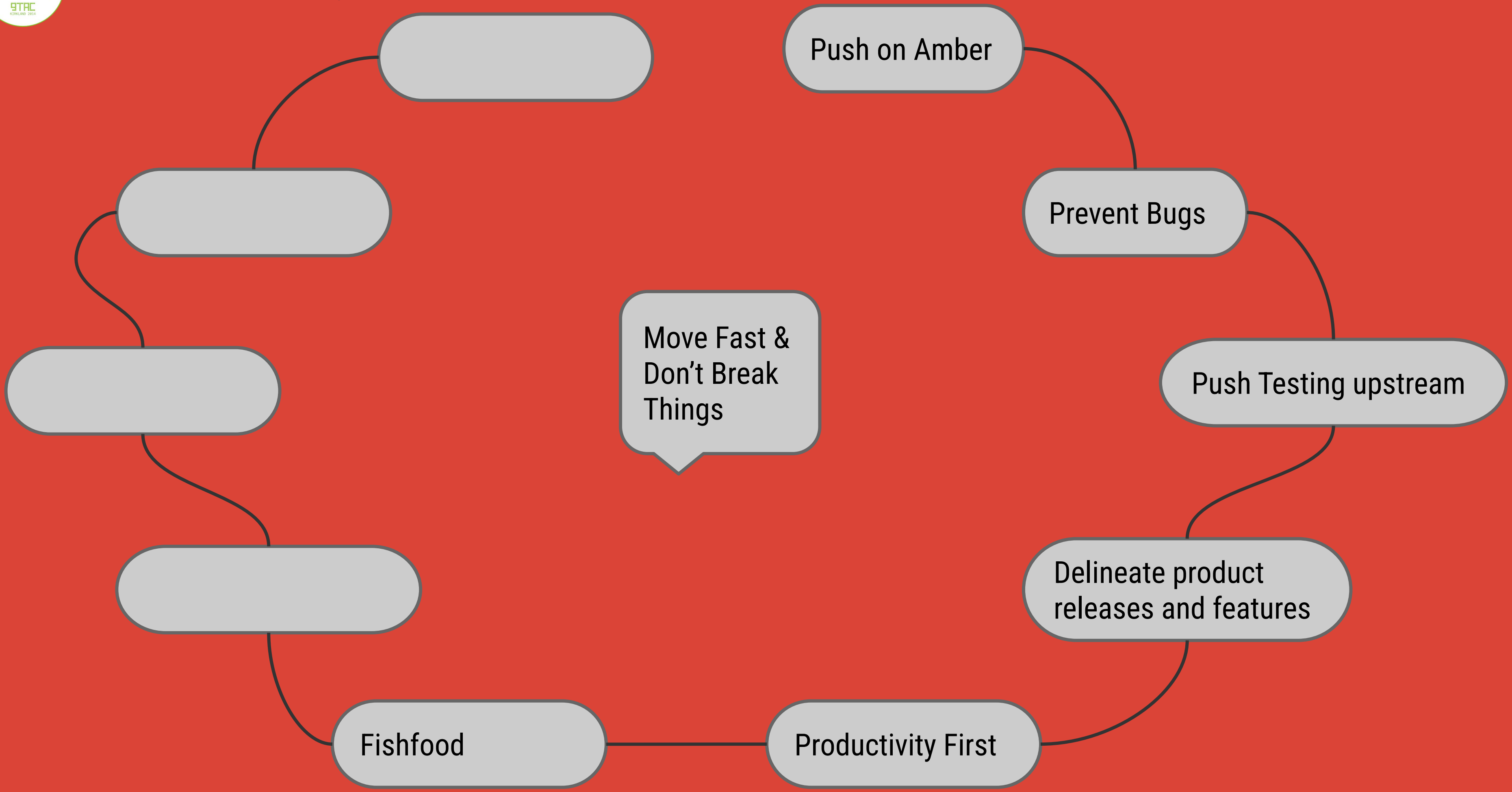


Robosheriff



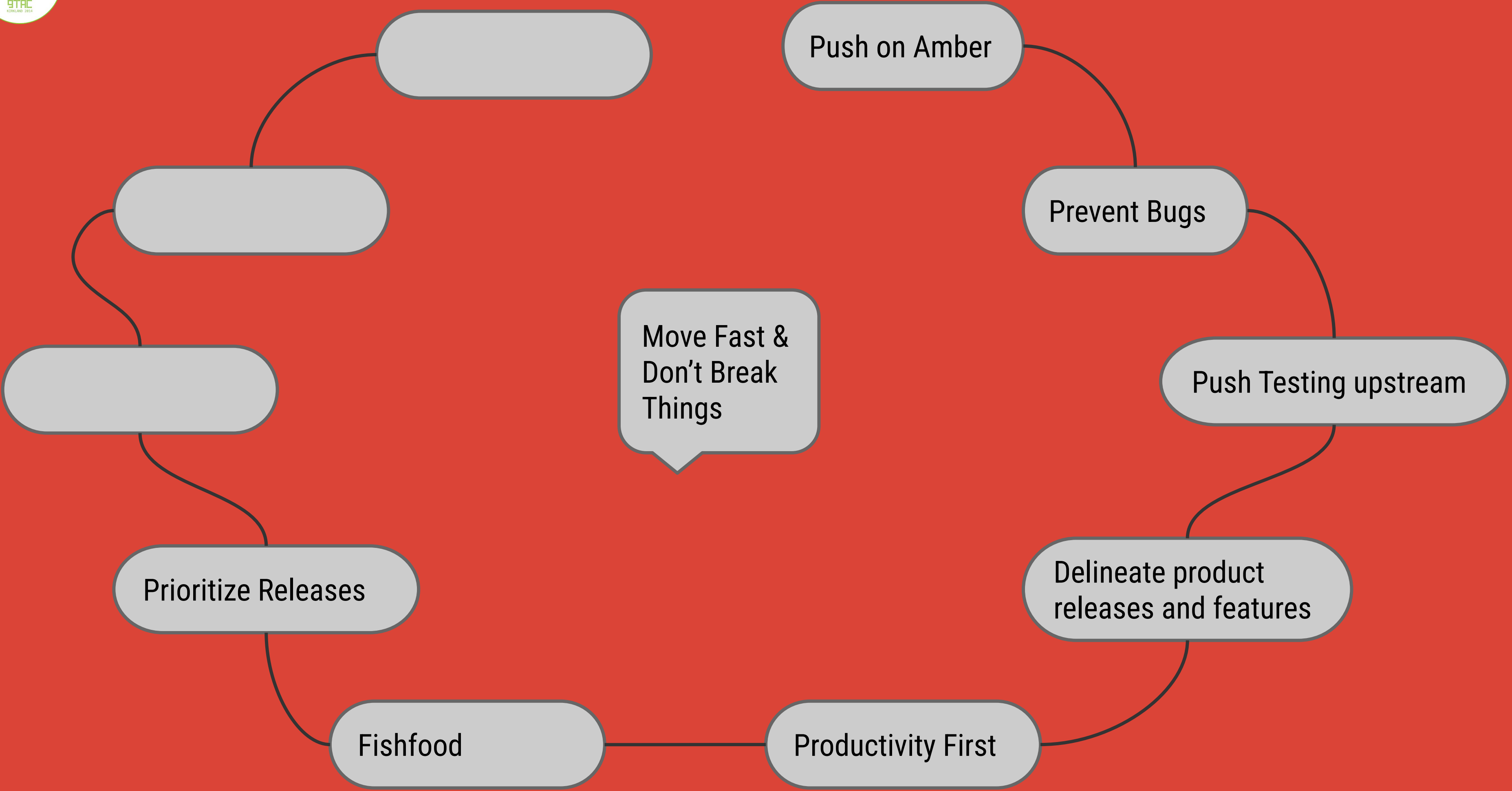
Social @ Google	**2012**	**2013**	**2014**	**Assessment**
Average presubmit time	21 min	28 min	22 min	Productivity
Code Coverage	???	72.80%	75.60%	Automation
% Green Cycles	71.40%	82.40%	89.60%	Test Hygiene
Avg Submit to Prod Time	17 hrs	11 hrs	13 hrs	Velocity
Total presubmit run time	196 d	11392 d	8033 d	Better Tests
Total automation time	22697 d	52785 d	114040 d	Better Tests
P1 bugs avg resolution time	69 d	28 d	13 d	Bug Hygiene

Move Fast & Don't Break Things



Fishfood

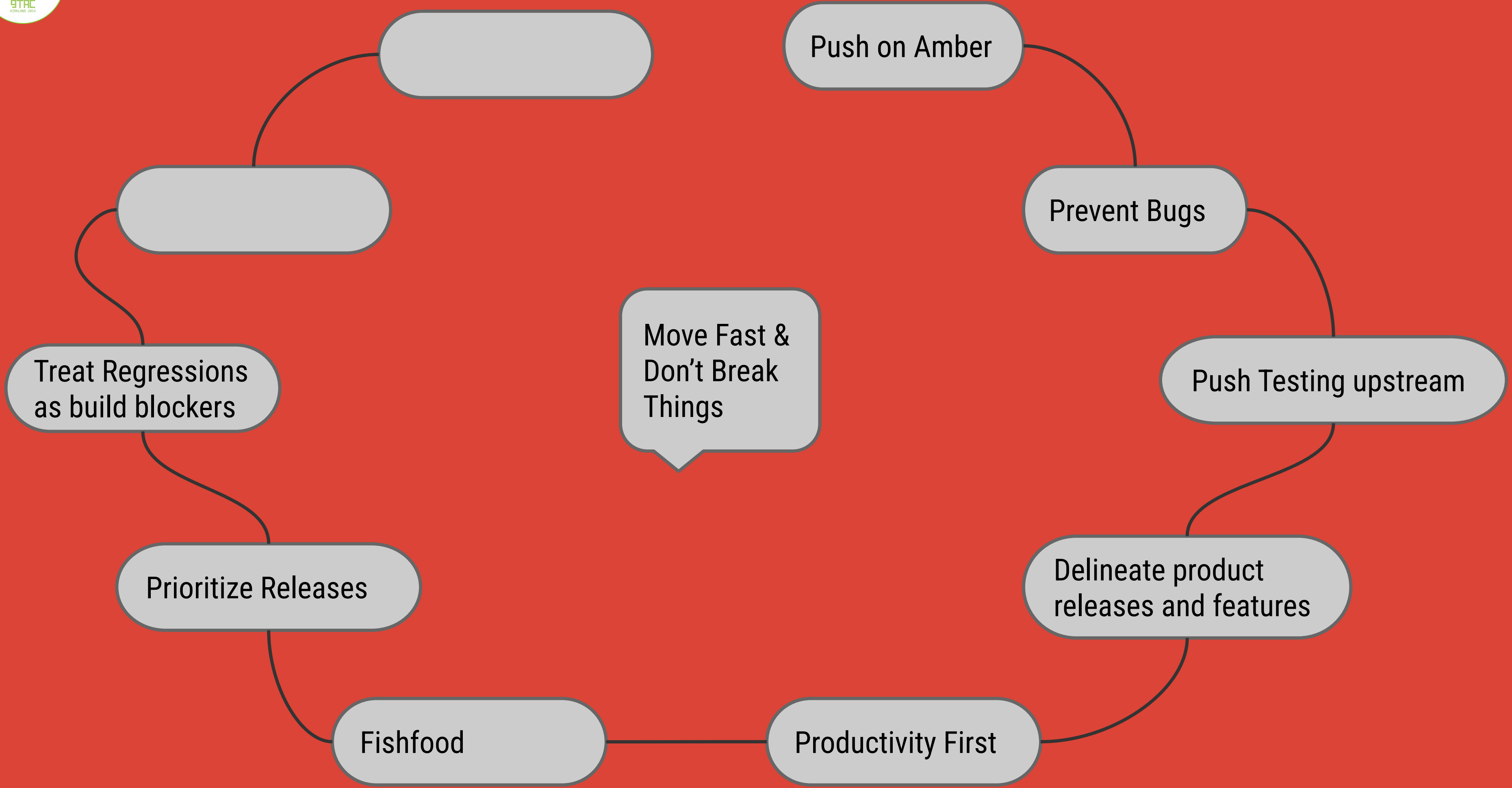
- Live on the bleeding edge
- Bugs get found/fixed
- No SLA for fishfood from test
- Rapid iterations. 4 hours from design to bug bash for a feature.


Move Fast & Don’t Break Things
Push on Amber
Prevent Bugs
Push Testing upstream
Delineate product releases and features
Productivity First
Fishfood
Prioritize Releases

Prioritize Releases

- Must be an ongoing commitment
- All must want to fix root problems (post-mortem!)
- Have a dedicated release team
- Make sure everyone understands it

Move Fast & Don't Break Things


Push on Amber
Prevent Bugs
Push Testing upstream
Delineate product releases and features
Productivity First
Fishfood
Prioritize Releases
Treat Regressions as build blockers
Move Fast & Don't Break Things

Treat regressions as build breaks

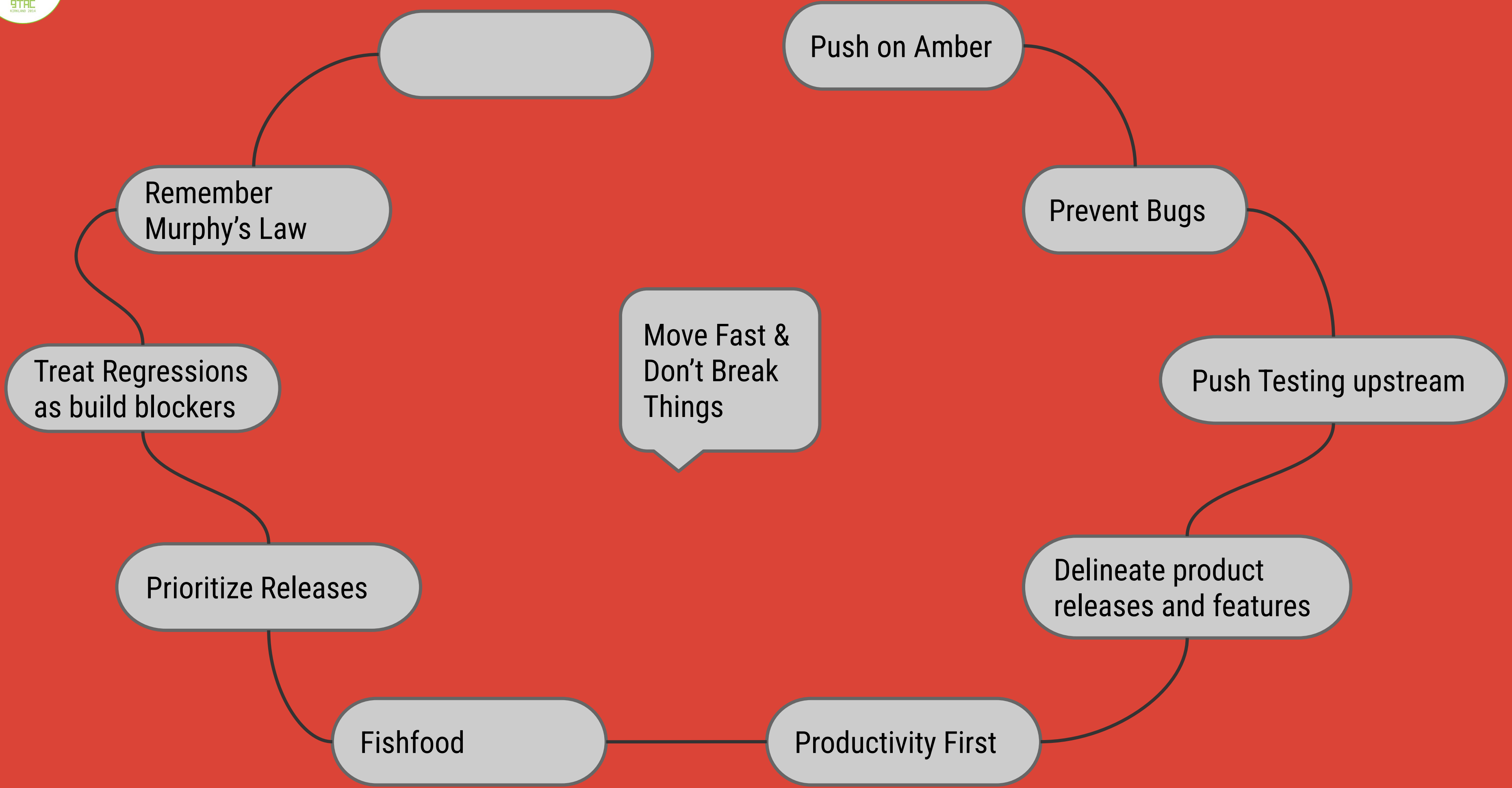
Cultural shift

Rollback == guaranteed fix

Verifications are simpler

Devs not under gun for fix

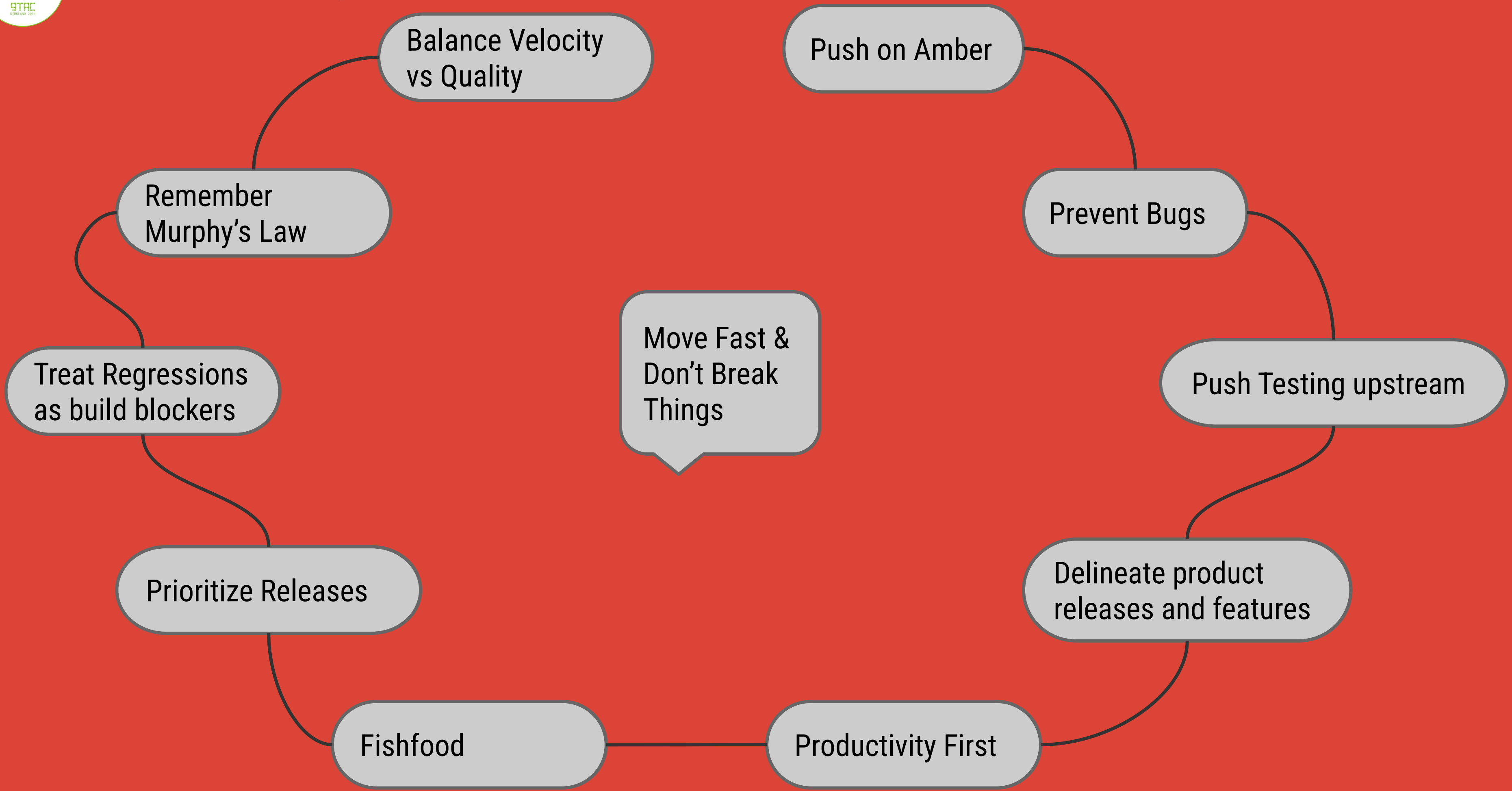
Move Fast & Don’t Break Things



Remember Murphy’s Law

- Kill switches for features
- Big refactorings behind flags
- All user visible changes behind experiments

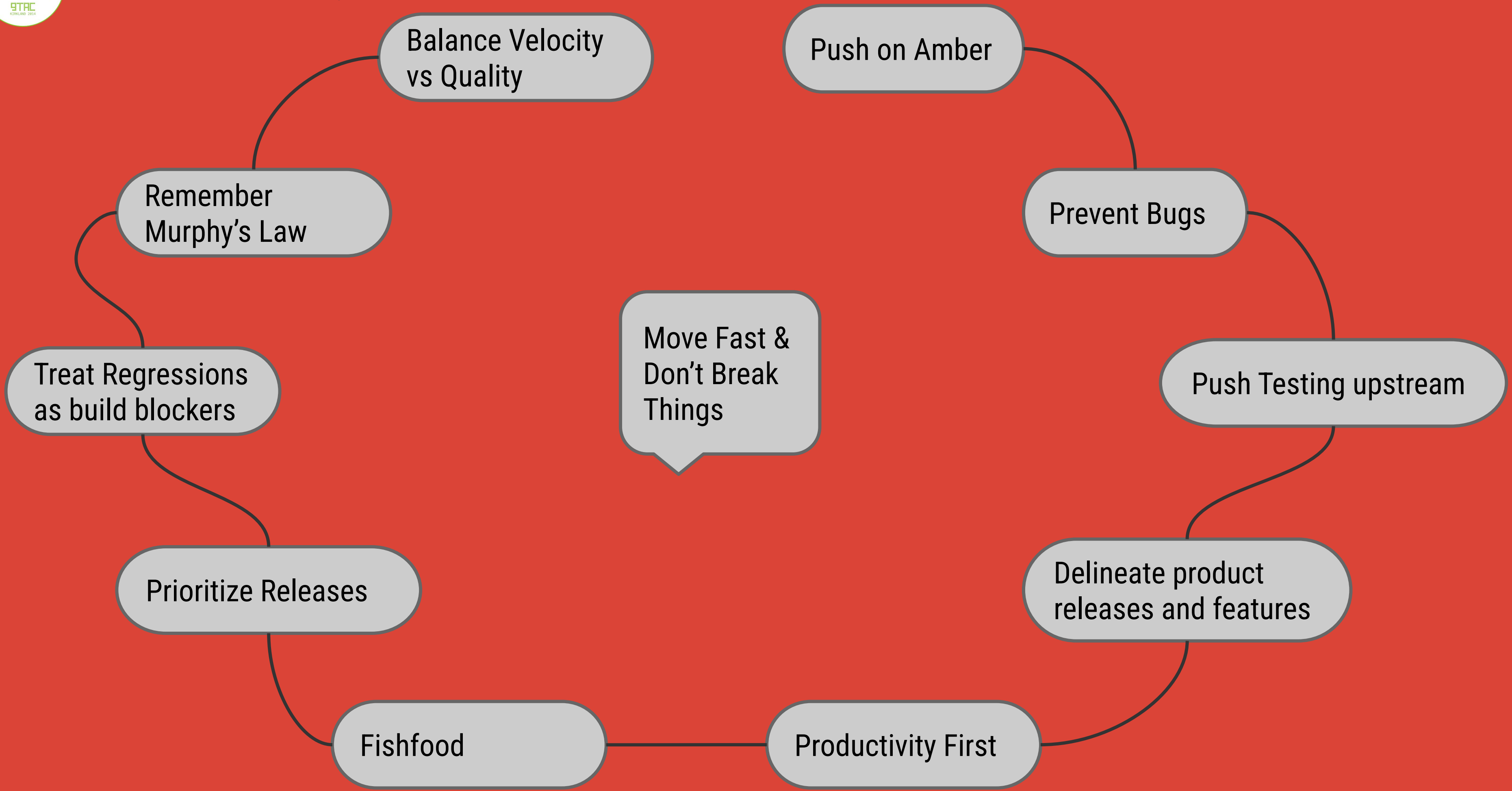
Move Fast & Don’t Break Things


Move Fast & Don’t Break Things
Push on Amber
Prevent Bugs
Push Testing upstream
Delineate product releases and features
Productivity First
Fishfood
Prioritize Releases
Treat Regressions as build blockers
Remember Murphy’s Law
Balance Velocity vs Quality

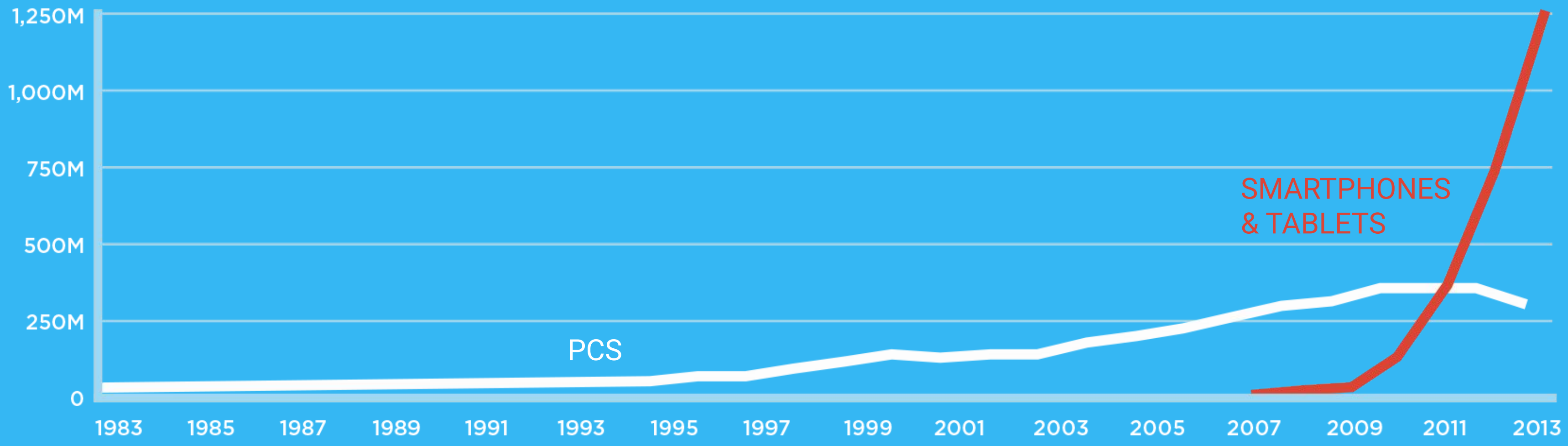
Balance Velocity vs Quality

- ~~Gate keepers~~
- Provide risk assessment
- Trusted Testers
- Things that can't be risked with: ACLs, data, security, upgrades, migrations

Move Fast & Don't Break Things


Move Fast & Don't Break Things
Push on Amber
Prevent Bugs
Push Testing upstream
Delineate product releases and features
Productivity First
Fishfood
Prioritize Releases
Treat Regressions as build blockers
Remember Murphy's Law
Balance Velocity vs Quality

It's no secret mobile is taking over.

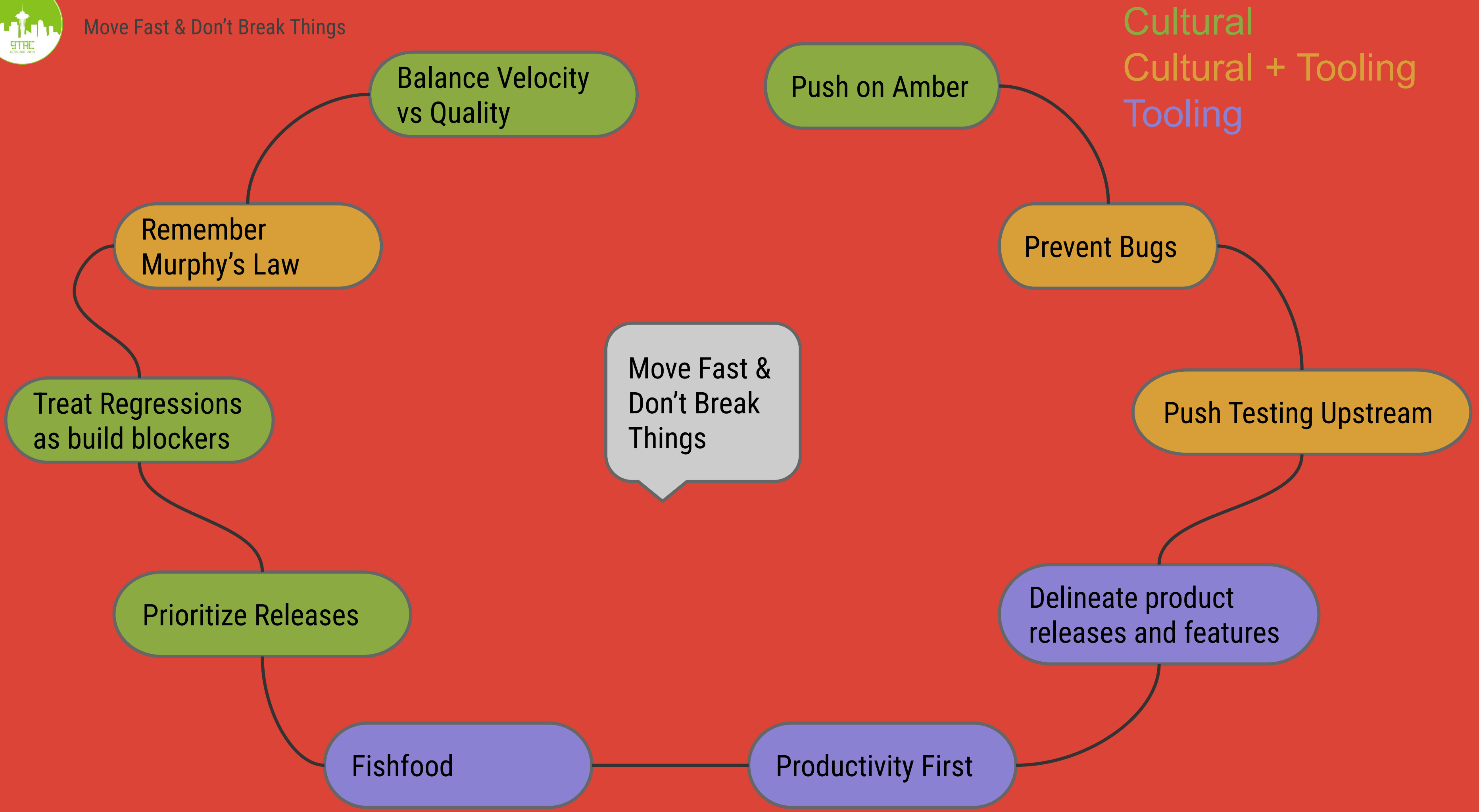


“Mobile First” Challenges

- Balance release velocity - can’t push daily to users.
- Mobile app updates use battery and cellular data.
- Cannot roll back a bad mobile app easily; higher quality bar needed.

Come join the discussion @ “Move Fast & Don’t Break Things” G+ Community

Move Fast & Don’t Break Things


Cultural
Cultural + Tooling
Tooling
Move Fast & Don’t Break Things
Balance Velocity vs Quality
Push on Amber
Prevent Bugs
Push Testing Upstream
Delineate product releases and features
Productivity First
Fishfood
Prioritize Releases
Treat Regressions as build blockers
Remember Murphy’s Law