Fitter, Happier, More Productive.

Removing Friction in the Developer Experience

Q-Con New York, June 27th 2017

Ade Trenaman, SVP Engineering, HBC Digital
t: @adrian_trenaman

http://tech.gilt.com

t: @hbcdigital fa: @hbcdigital in: hbc_digital

What are you doing while in the US?

I’m speaking a software conference.

Sir.

What’s your official title?

I’m SVP Engineering at Hudson’s Bay Co.

What’s your talk about?

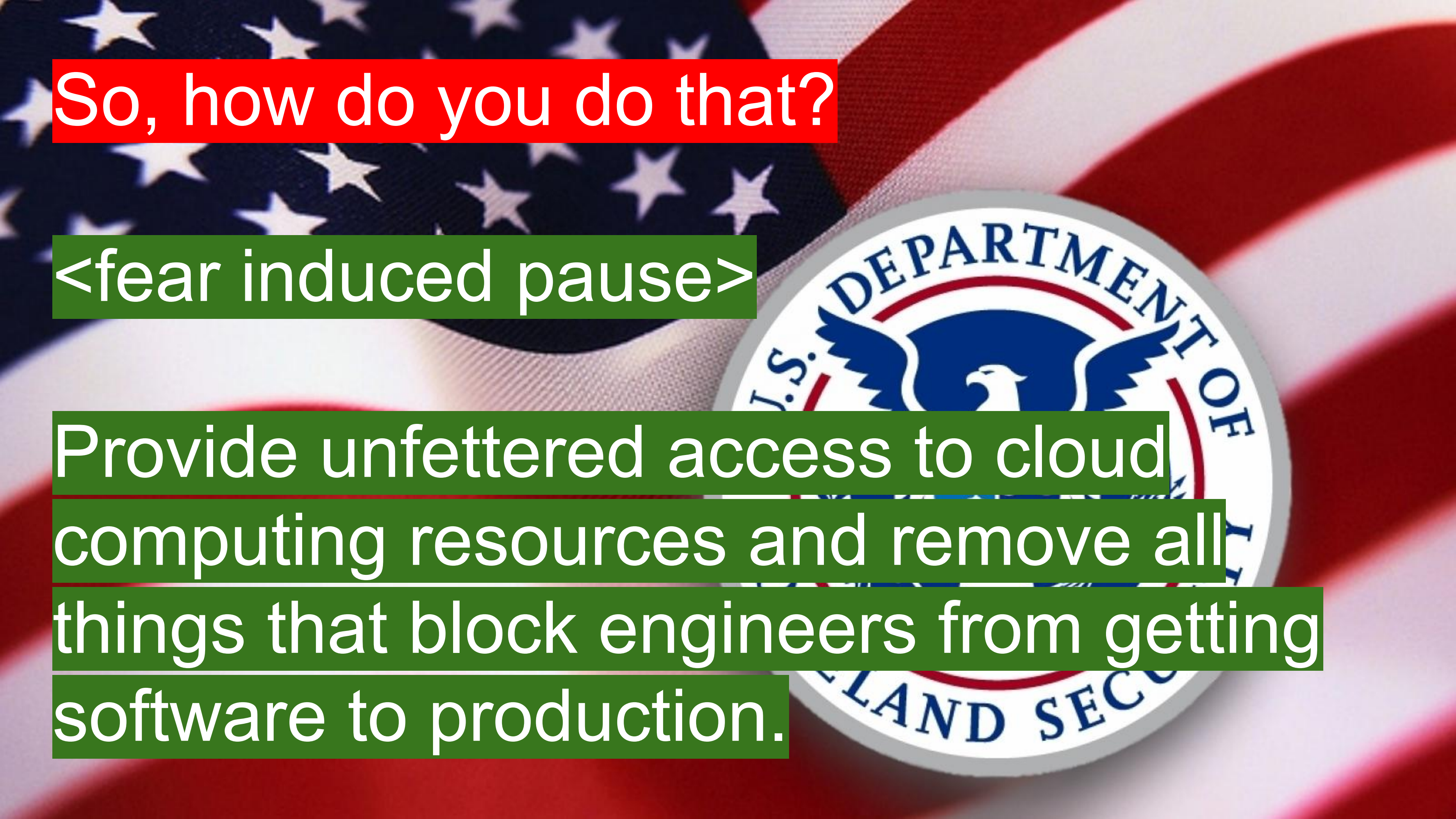
Ummm.

Improving the way we do software engineering.

So, how do you do that?

<fear induced pause>

Provide unfettered access to cloud computing resources and remove all things that block engineers from getting software to production.



It’s hard, huh, all that red tape and bureaucracy?

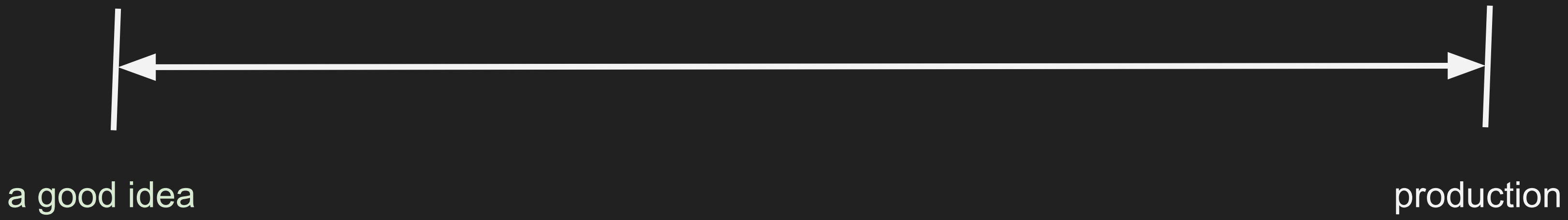
Yup.

Well, hope you solve it! :)
Welcome to America.

```
scala> println ("hello, world.")
hello, world.
```

production

minimise the distance between “hello, world” and production.



minimise the distance between a good idea and production.

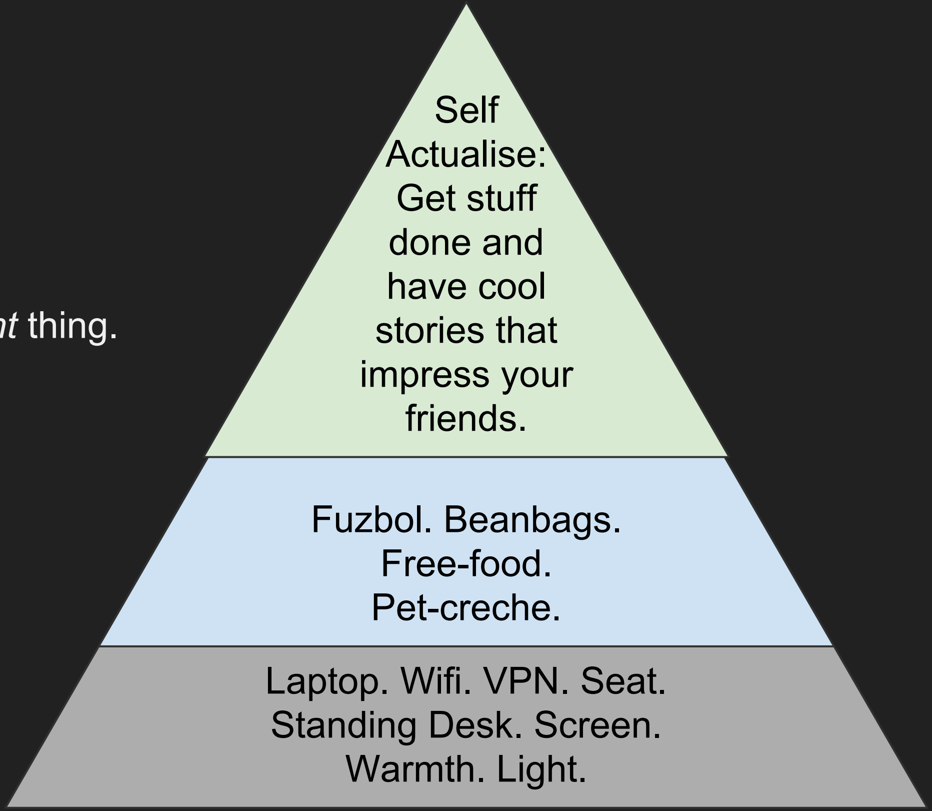
For great dev-ex:

"... build an organisation and architecture that allows you to deploy change **frequently**, **swiftly** and **safely** to production, and **own** the impact of that change"

This is *the most important* thing.

Perks.

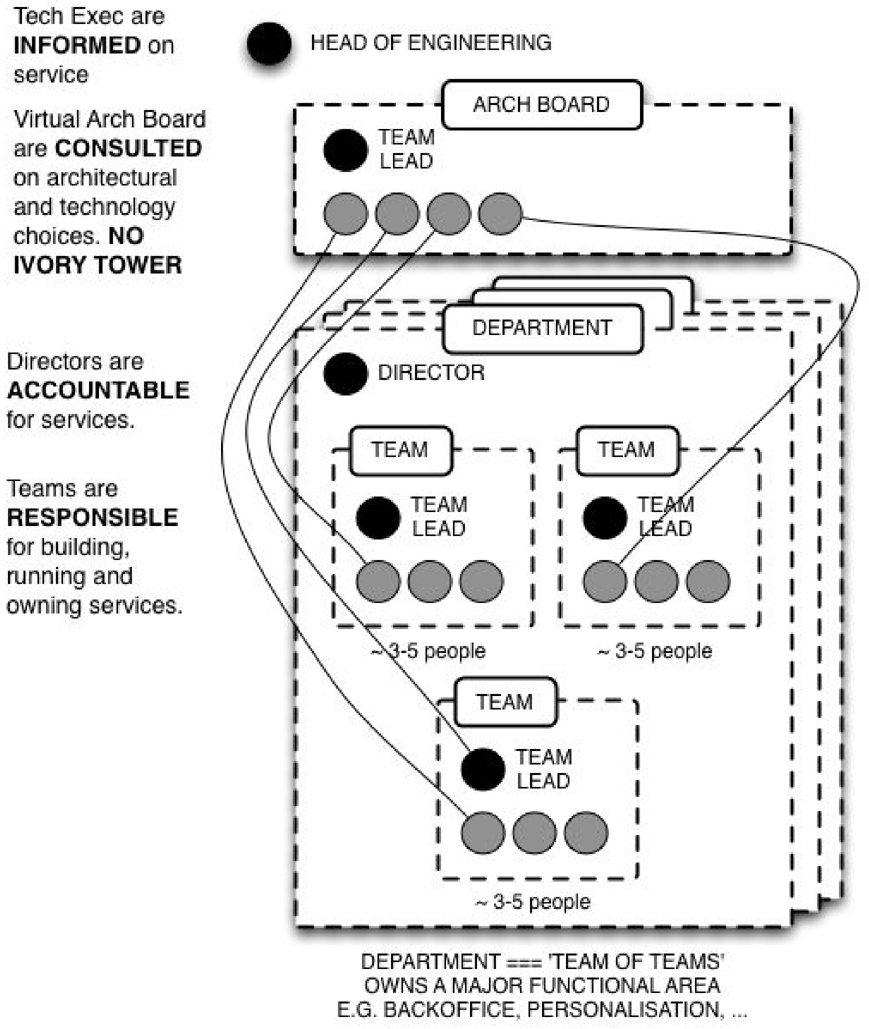
Basics. You must have these.


Self Actualise: Get stuff done and have cool stories that impress your friends.
Fuzbol. Beanbags. Free-food. Pet-creche.
Laptop. Wifi. VPN. Seat. Standing Desk. Screen. Warmth. Light.

DEVELOPER HIERARCHY OF NEEDS

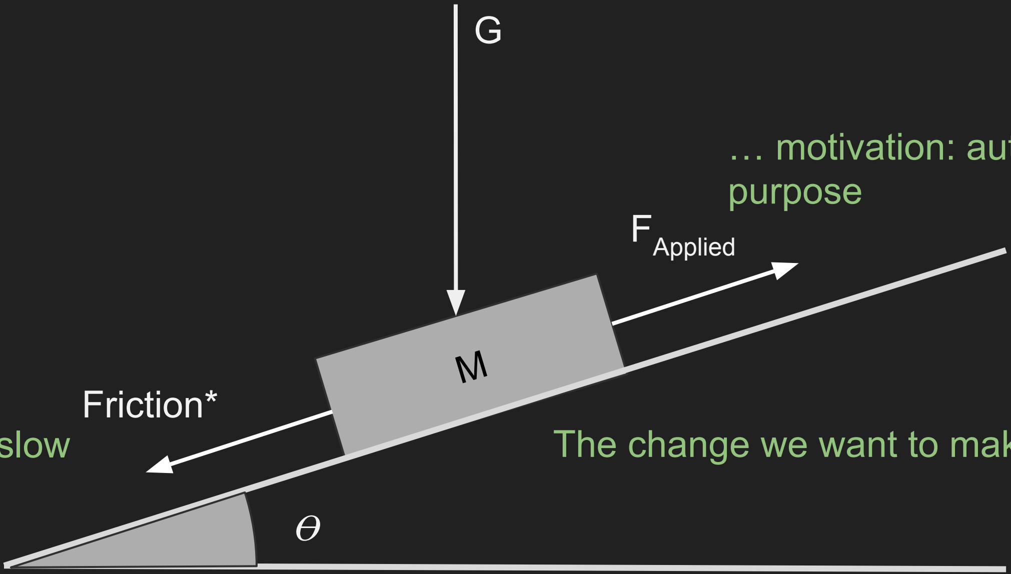
code *first*

Teams: 5±2 in size
Departments: 20±4
#leadersnotmanagers
#leaderswhocode: 85%, 60%, 15%
IC & Lead tracks
#devops
#ownership
#opensource


Tech Exec are **INFORMED** on service
Virtual Arch Board are **CONSULTED** on architectural and technology choices. **NO IVORY TOWER**
Directors are **ACCOUNTABLE** for services.
Teams are **RESPONSIBLE** for building, running and owning services.
HEAD OF ENGINEERING
ARCH BOARD
TEAM LEAD
DEPARTMENT
DIRECTOR
TEAM
TEAM LEAD
~ 3-5 people
TEAM
TEAM LEAD
~ 3-5 people
TEAM
TEAM LEAD
~ 3-5 people
DEPARTMENT === 'TEAM OF TEAMS'
OWNS A MAJOR FUNCTIONAL AREA
E.G. BACKOFFICE, PERSONALISATION, ...

SAY “AUTONOMY, MASTERY, PURPOSE”
ONE MORE TIME

… work is hard.



… motivation: autonomy, mastery, purpose

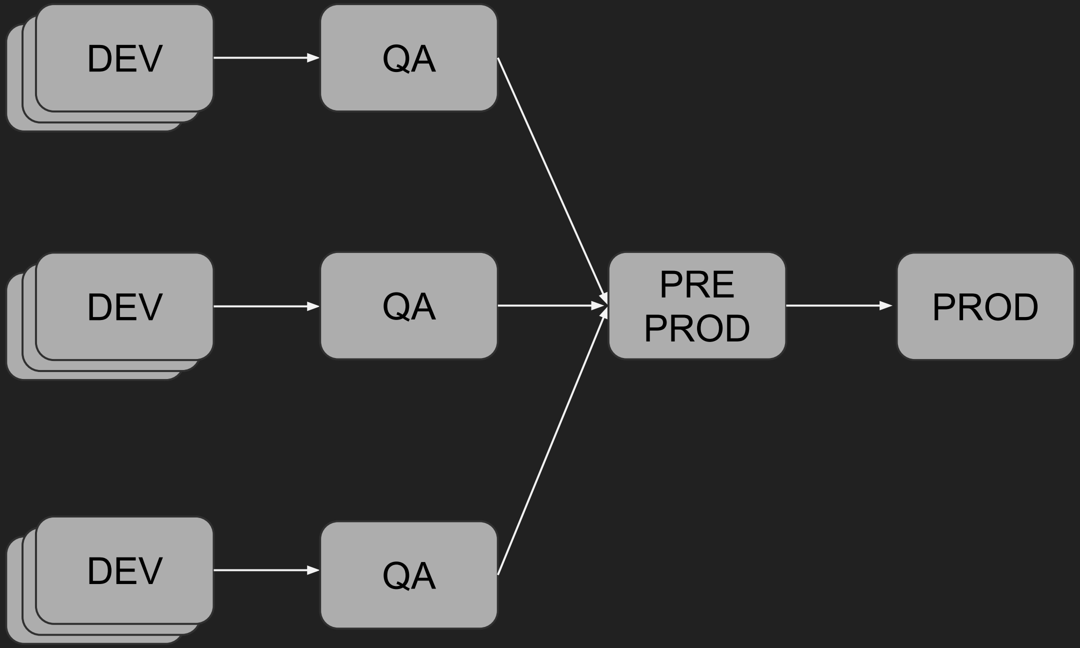
… all the things that slow us down or block us.

The change we want to make...

* reactive force resisting motion

f: Staging/Testing Environments

Prefer to test in production. #srsly



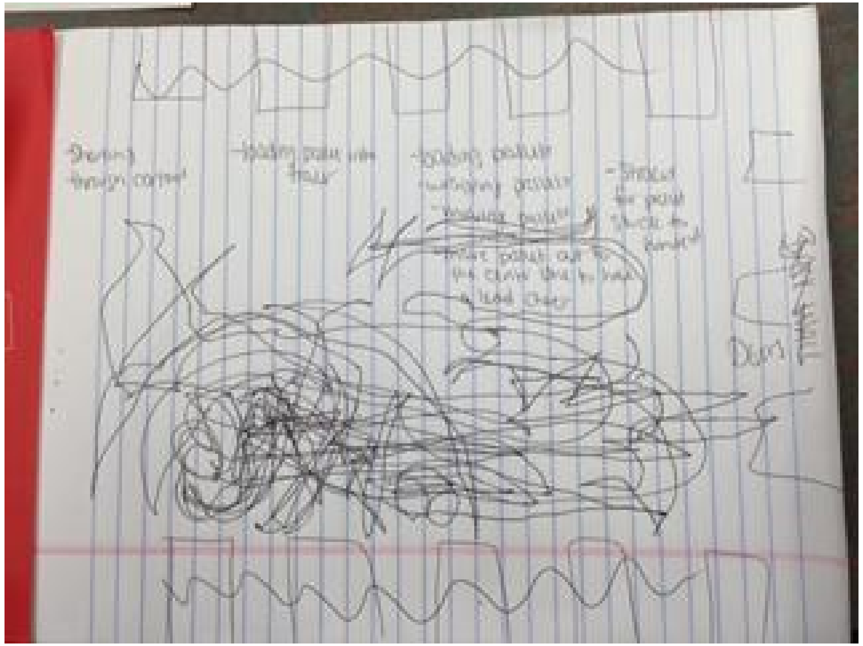
Dev, QA & Test environments are high-friction places to write code.

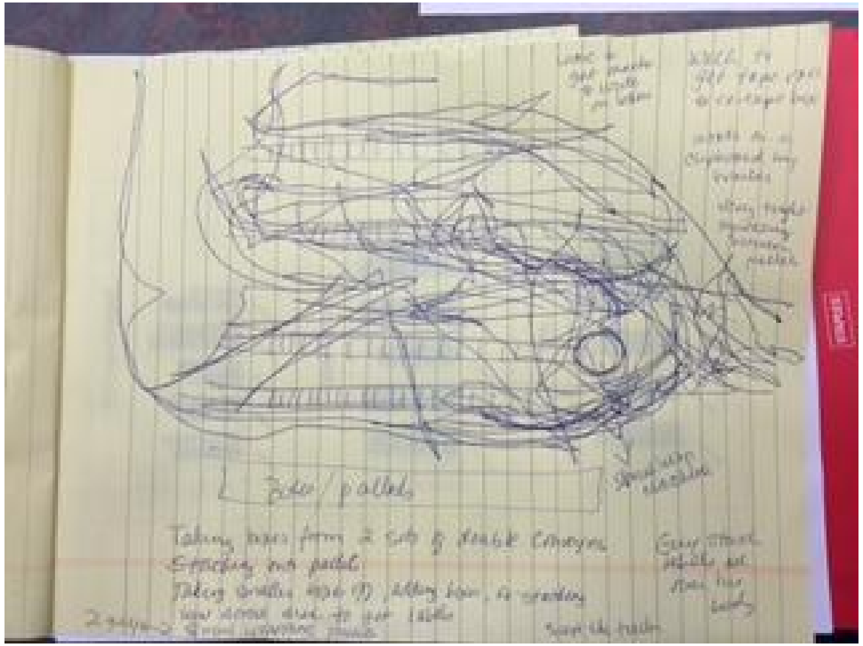
Borrowing from *lean / six-sigma*

Lack of Flow, Excessive Bending, Kneeling, Reaching

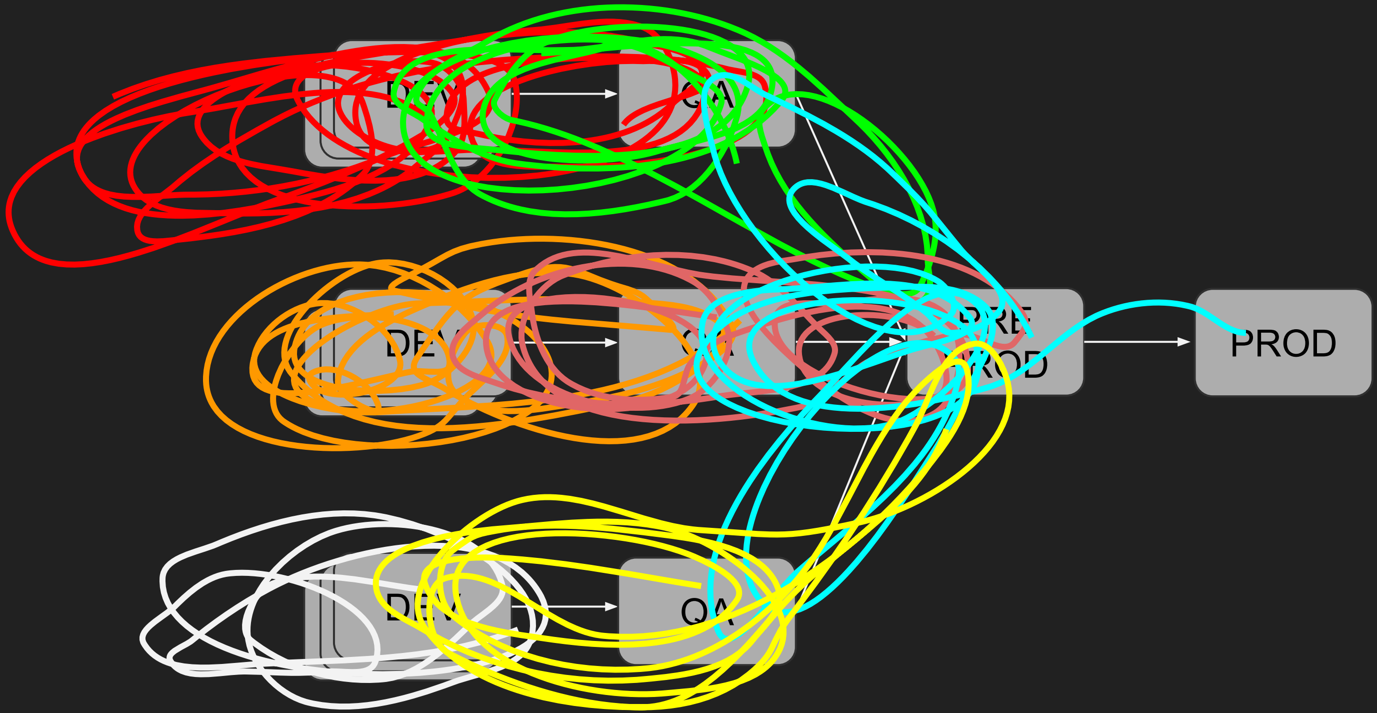
Increased Waste = Lower Productivity, Safety Opportunities

MOTION STUDY – "SPAGHETTI DIAGRAMS"




Boxes / pallets
Taking boxes from 2 sets of double conveyors
Stacking onto pallet
Taking smaller boxes (?) putting them on conveyor
Walk across aisle to get table

Spaghetti Diagrams make poor layouts and wasted motion obvious



Spaghetti diagram of movement and handover within the software delivery process.

Muda - “Waste” in the software software delivery process

Intellect
- Spending time building and debugging environments instead of adding value

Overproduction
- Encourages fewer ‘big bang’ releases

Waiting
- Can’t get my stuff deployed

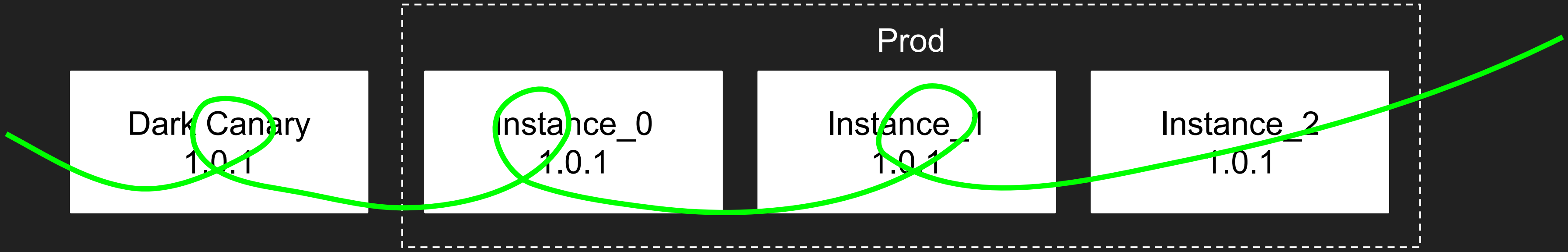
Overprocessing
- Tickets tested and rested in different environments.

Motion
- Commit deploy test commit deploy test commit deploy test...

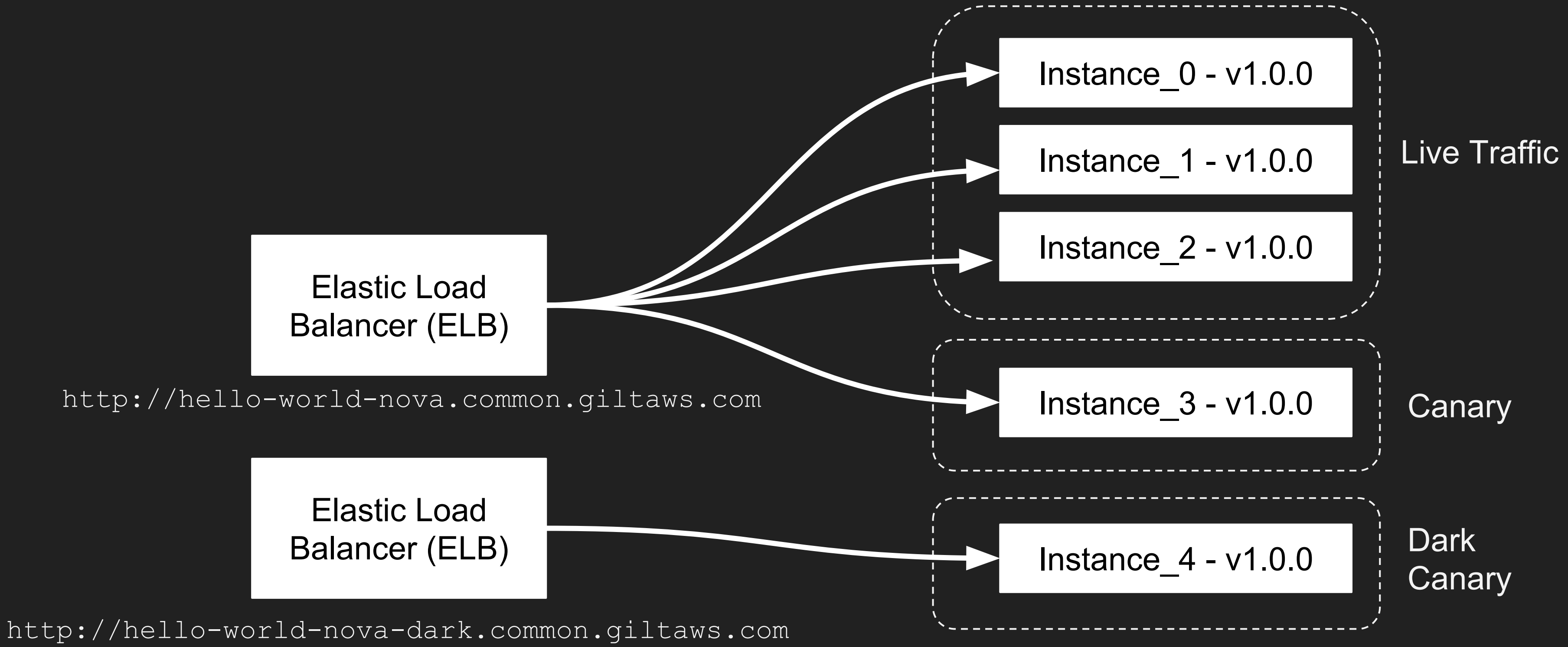
Rework
- Works in one environment, not in another

Inventory
- Lots of commits held up in the pipeline.

Transportation
- Multiple handoffs between Engineers, QA & Ops

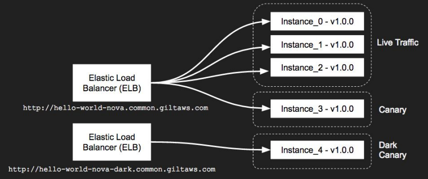


Core idea #1: test in prod with dark canaries, canaries, release, roll-back.


Elastic Load Balancer (ELB)
http://hello-world-nova.common.giltaws.com
Elastic Load Balancer (ELB)
http://hello-world-nova-dark.common.giltaws.com
Instance_0 - v1.0.0
Instance_1 - v1.0.0
Instance_2 - v1.0.0
Live Traffic
Instance_3 - v1.0.0
Canary
Instance_4 - v1.0.0
Dark Canary

github.com/gilt/nova- deployment patterns

Amazon CloudWatch


Elastic Load Balancer (ELB)
http://hello-world-nova.common.giltaws.com
Elastic Load Balancer (ELB)
http://hello-world-nova-dark.common.giltaws.com
Instance_0 - v1.0.0
Instance_1 - v1.0.0
Instance_2 - v1.0.0
Live Traffic
Instance_3 - v1.0.0
Canary
Instance_4 - v1.0.0
Dark Canary

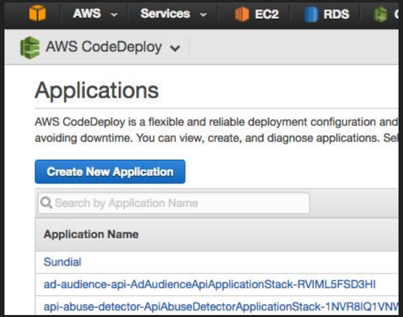
nova.yml

templates

```
$> nova stack create production
```

CloudFormation

CodeDeploy

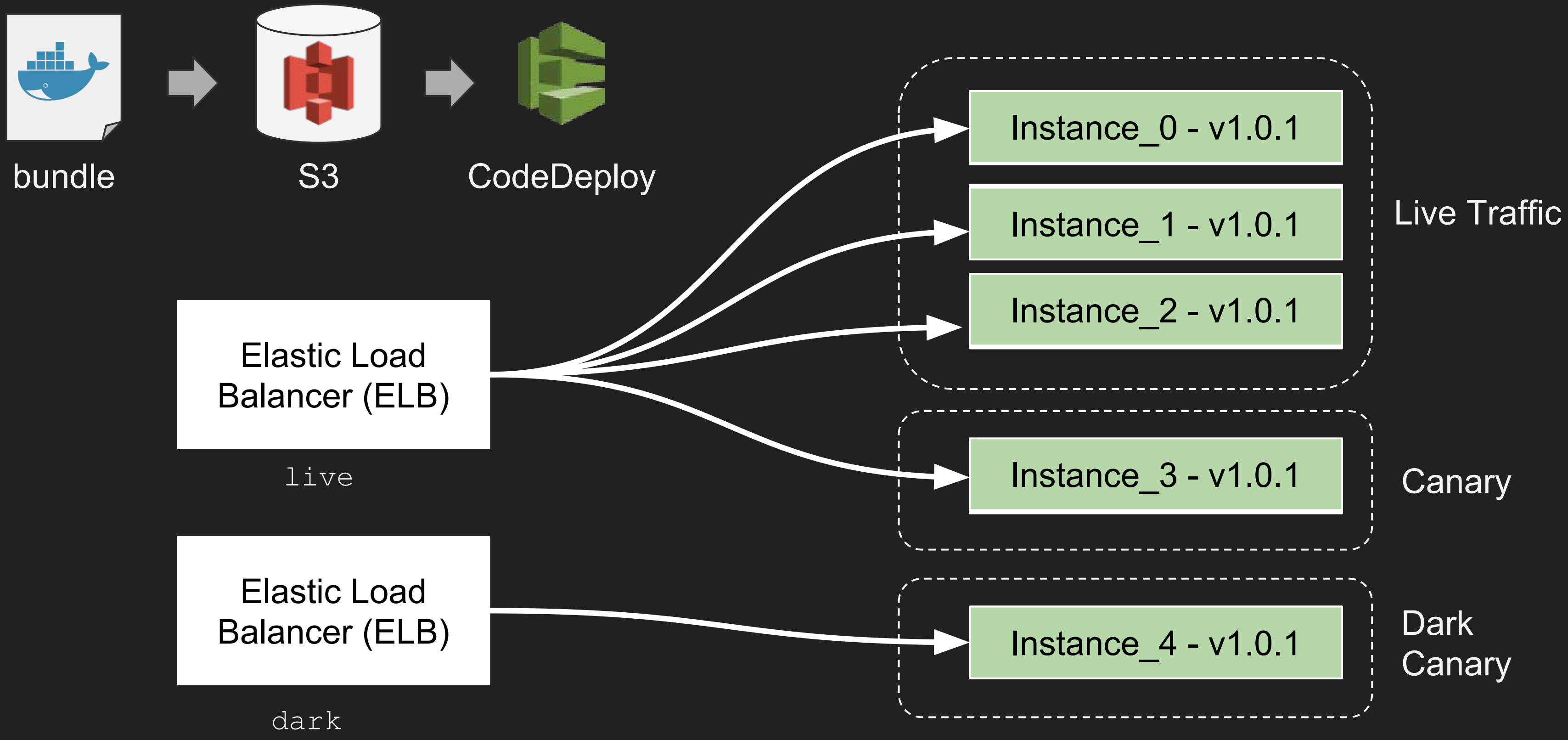


Applications

AWS CodeDeploy is a flexible and reliable deployment configuration and
avoiding downtime. You can view, create, and diagnose applications. Se

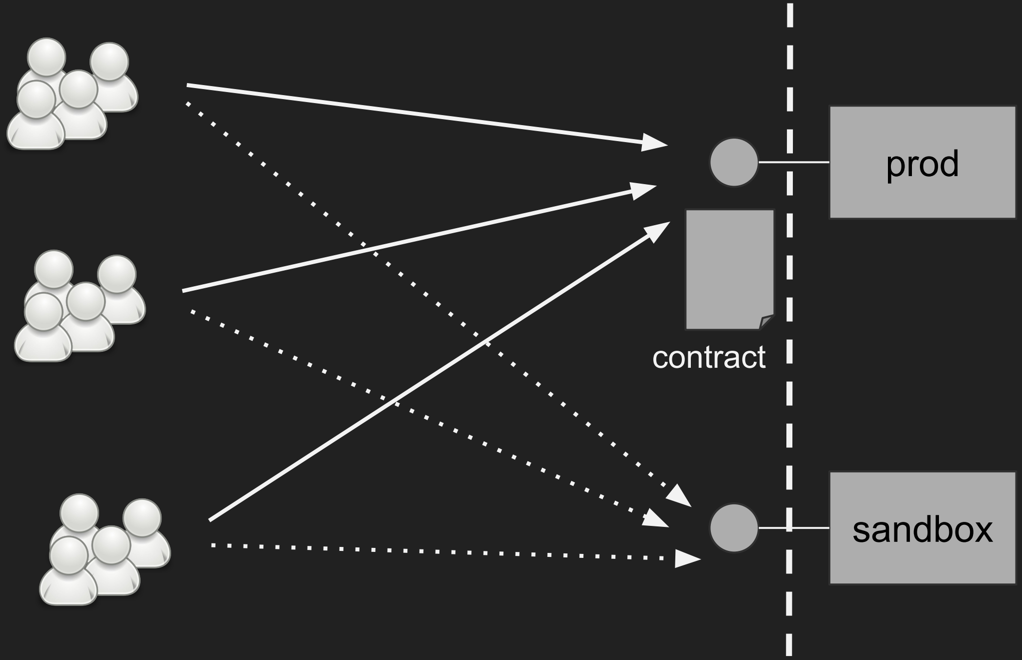
Application Name
Sundial
ad-audience-api-AdAudienceApiApplicationStack-RVIML5FSD3HI
api-abuse-detector-ApiAbuseDetectorApplicationStack-1NVR8IQ1VNV

github.com/gilt/nova - creating environments

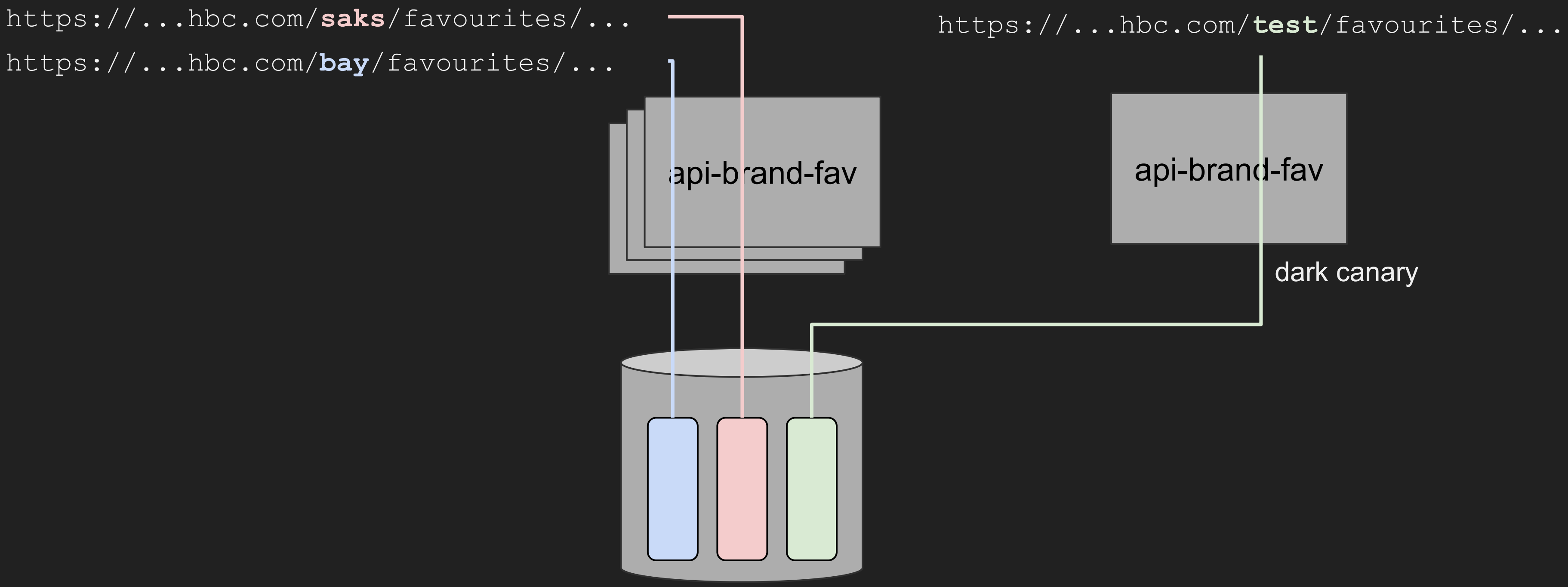


```
$> nova deploy common Production
1.0.1
```

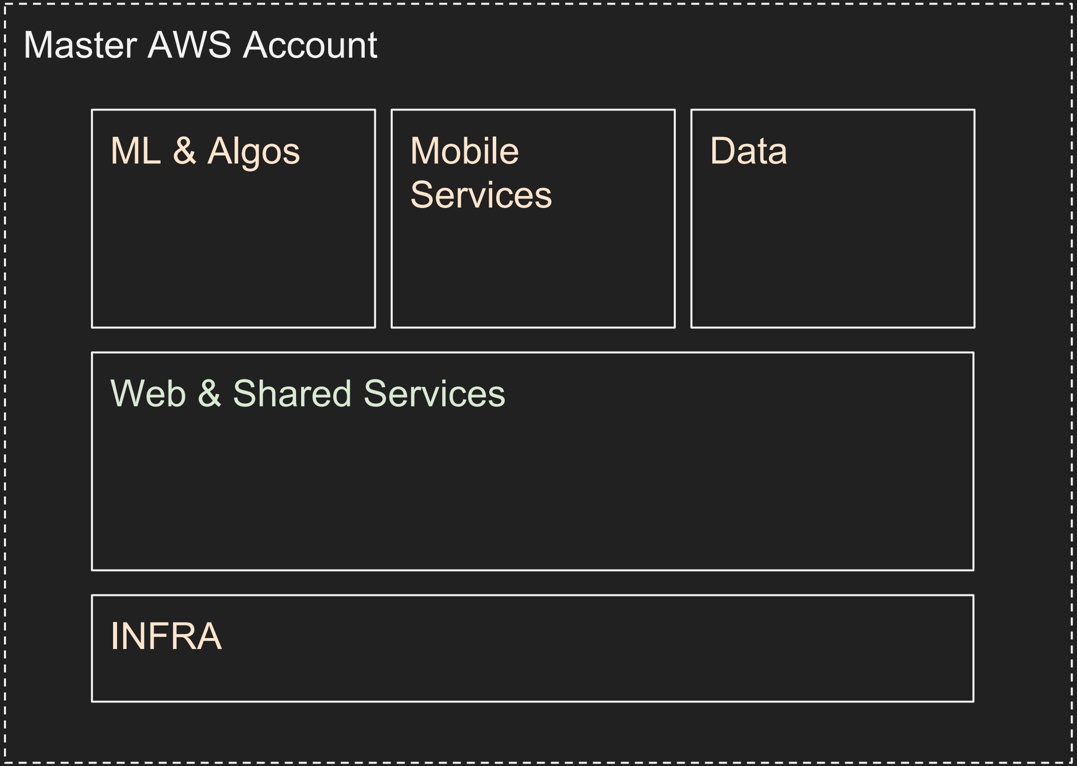
github.com/gilt/nova- deployment



Core idea #2: your teams are startups providing services to other development teams


https://...hbc.com/saks/favourites/...
https://...hbc.com/bay/favourites/...
https://...hbc.com/test/favourites/...
api-brand-fav
api-brand-fav
dark canary

Core idea #3: exploit multi-tenant design for confident testing in production

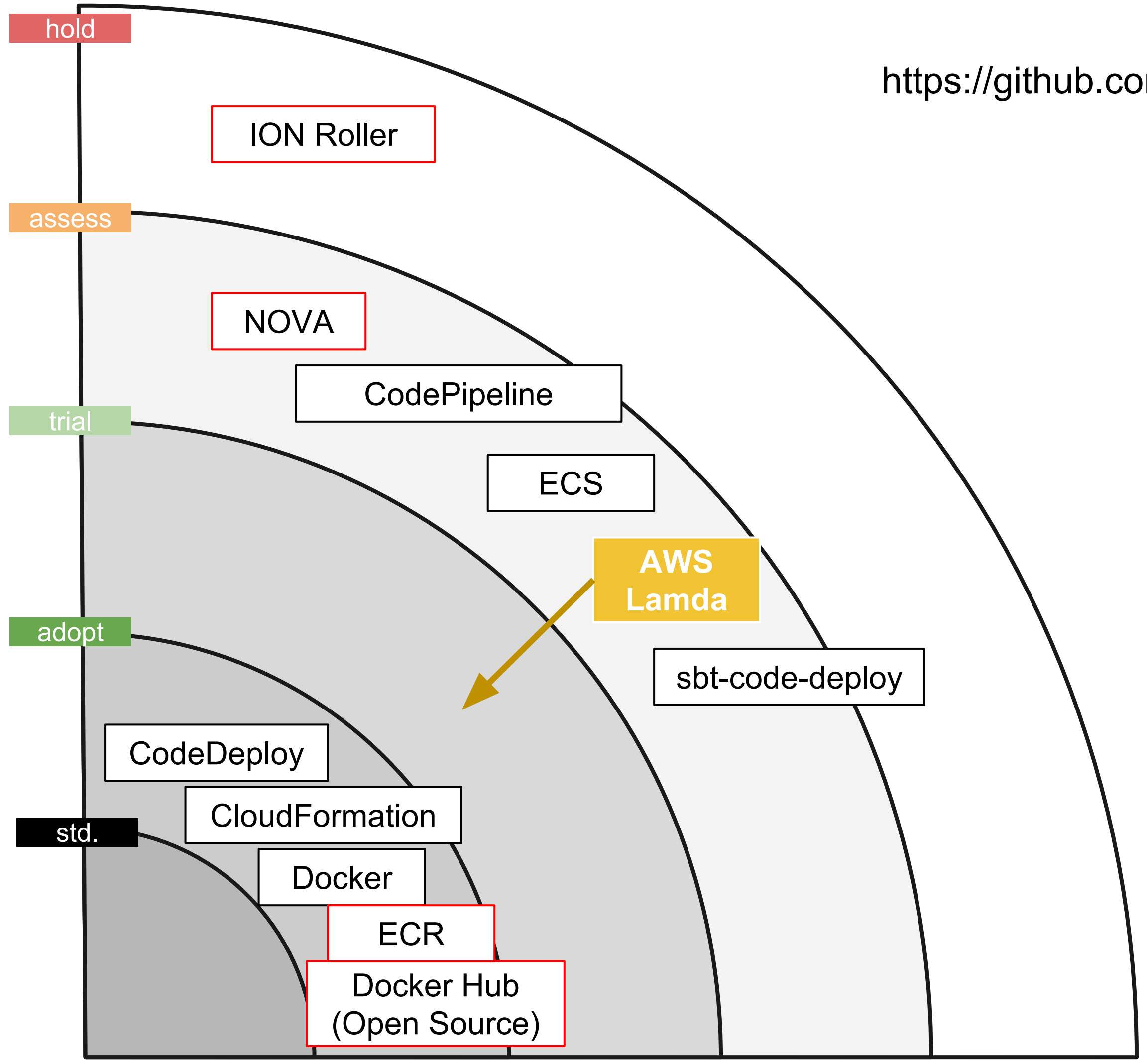


Core idea #4: give your teams secure, unfettered control over their own infrastructure. Segregate and apply command-and-control where you need it most.

f: Forced technology choices.

Prefer voluntary adoption.

https://github.com/gilt/standards



adoption by rule
centralised
uniform
efficient

Steer towards classroom size
consensus

voluntary adoption
decentralised
diverse
effective

HAILO

go

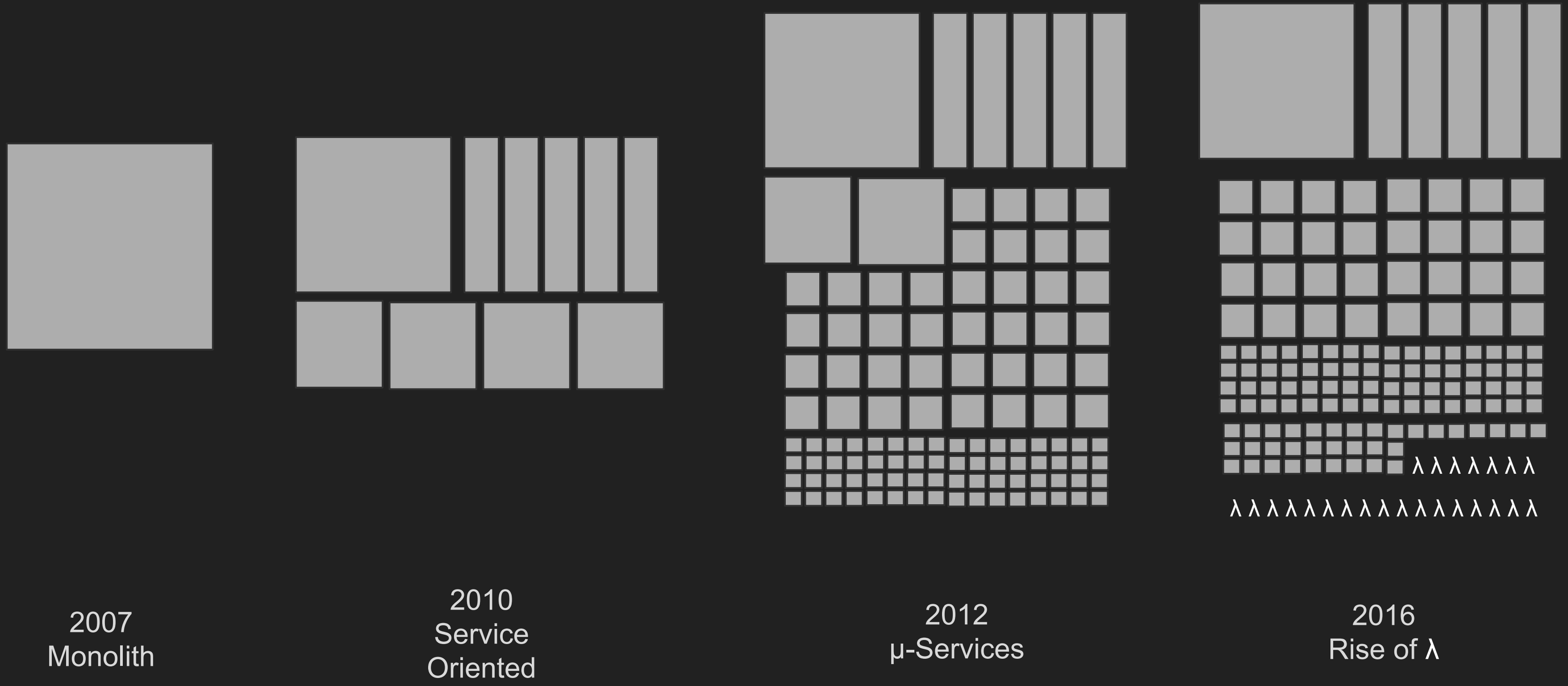
GILT

Scala, Java,
Ruby, Swift, JS,
Node, ...

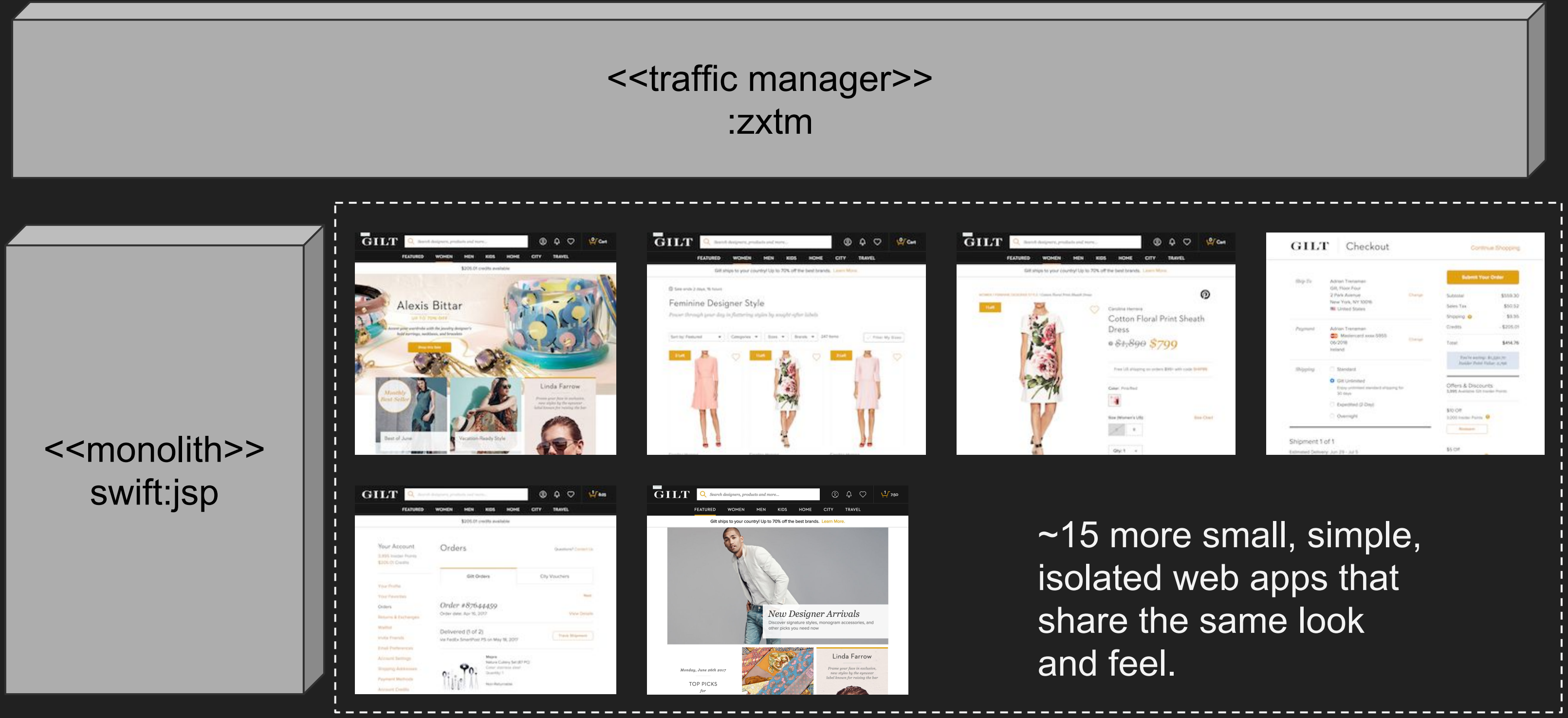
Philosophical note: choose your abstractions &
frameworks carefully.

f: Fear of Breaking All The Things

Adopt μ-services. Adopt λ.
Maximize code-to-cruft-ratio.

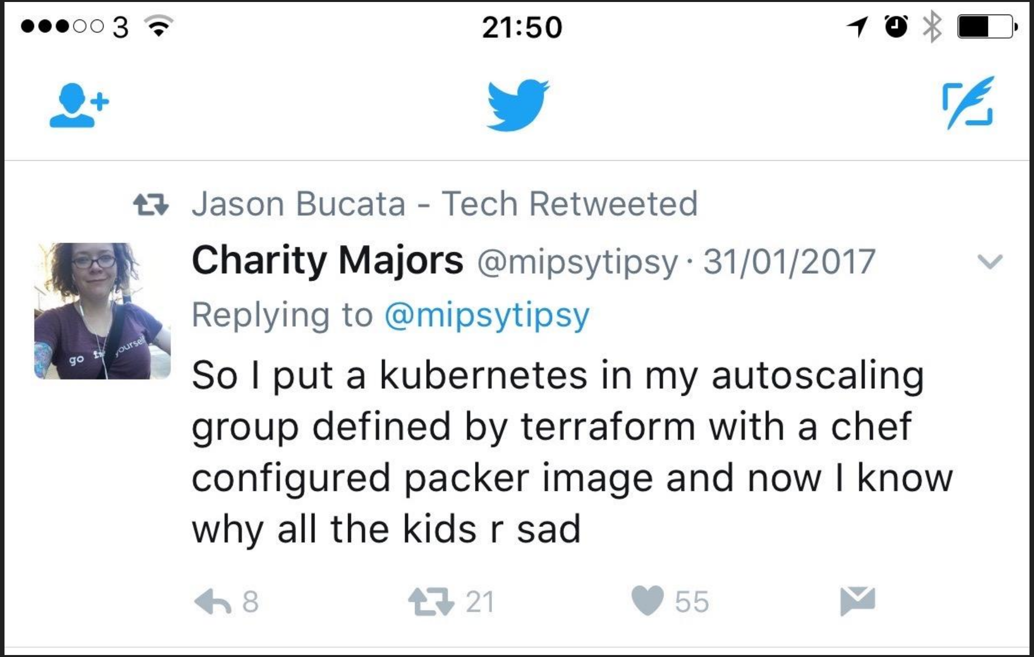


A minimalist abstraction of our architectural evolution



~15 more small, simple, isolated web apps that share the same look and feel.

Lots of Small Apps (LOSA) - AKA “micro-frontends”.



21:50

Jason Bucata - Tech Retweeted

Charity Majors @mipsytipsy · 31/01/2017

Replying to @mipsytipsy

So I put a kubernetes in my autoscaling group defined by terraform with a chef configured packer image and now I know why all the kids r sad

8 21 55

I hear ya.

A small-but-important problem: marketing redirects

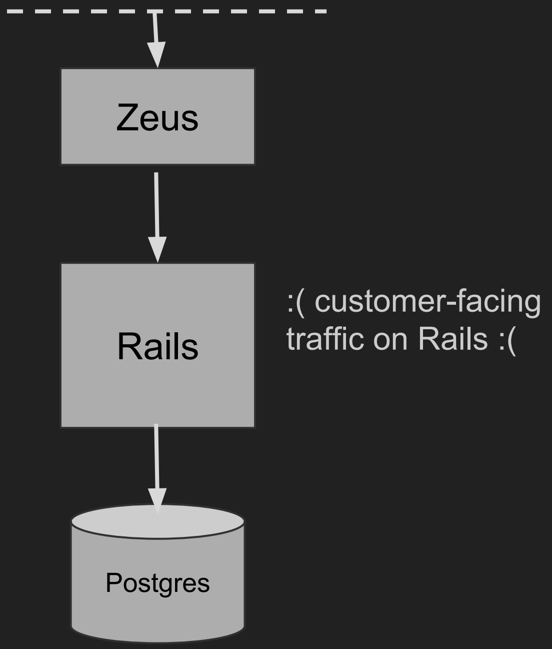
We have marketing URLs like:
https://gilt.com/loveaws

We need to look up the slug ‘loveaws’ and change that to a ‘pkey’ for our login, so we can redirect with 302 to:

https://gilt.com/login?pkey=loveaws&...

Existing solution routed to legacy Ruby on Rails app:

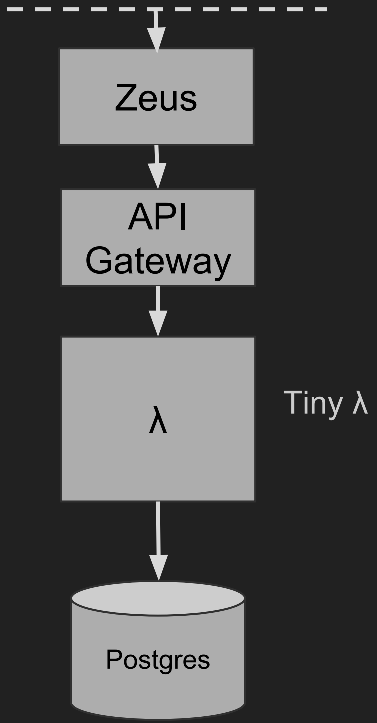
- not scalable.
- not ‘symmetric’

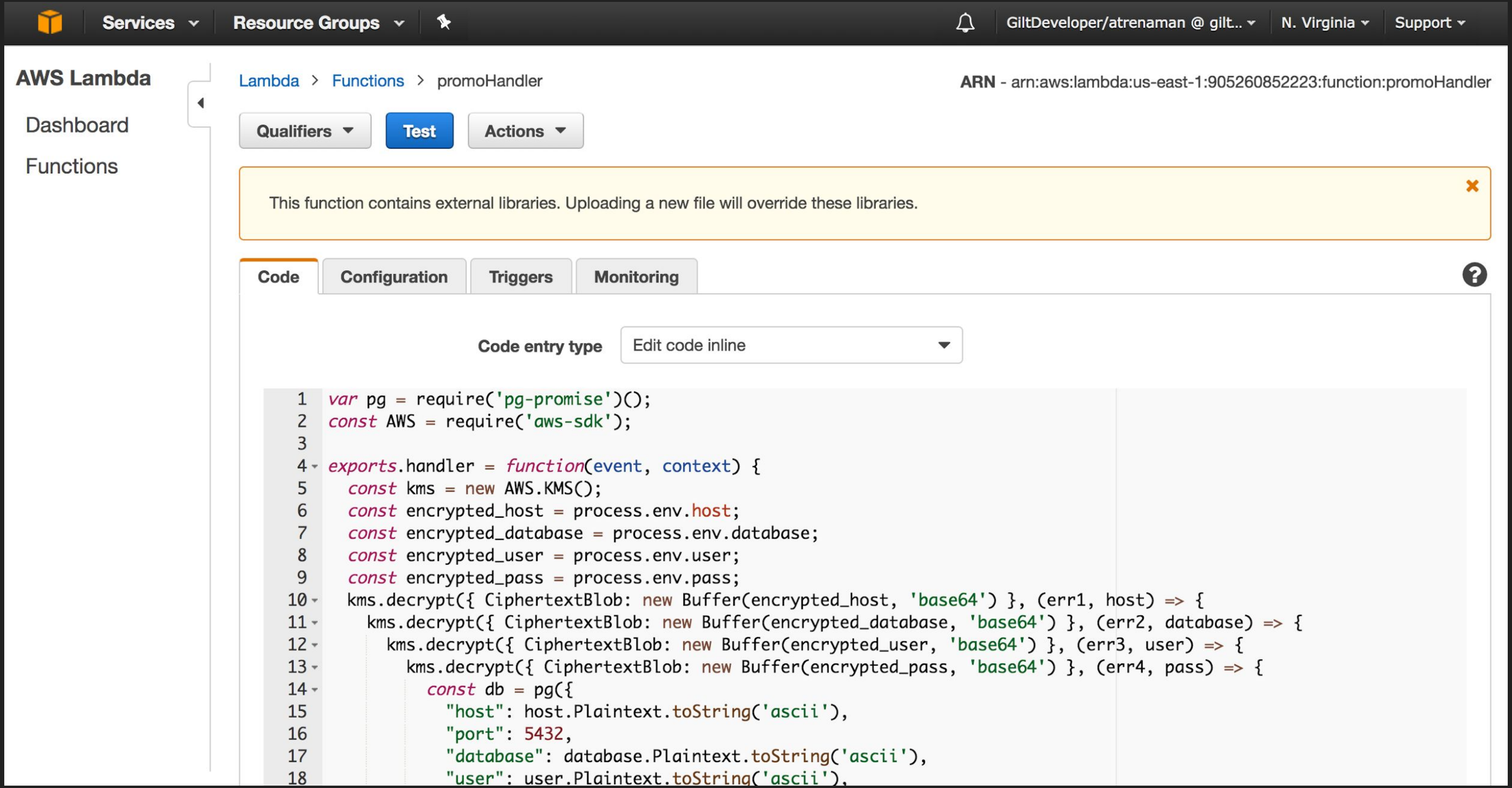


λ-based solution

Replace with solution using API-Gateway + Lambda + KMS

- 3/80 LOC (cruft/code) .js
- KMS used for encrypted DB credentials
- Response cached
- No longer hits Rails!





ARN - arn:aws:lambda:us-east-1:905260852223:function:promoHandler

This function contains external libraries. Uploading a new file will override these libraries.

Code entry type: Edit code inline

```
var pg = require('pg-promise')();
const AWS = require('aws-sdk');

exports.handler = function(event, context) {
  const kms = new AWS.KMS();
  const encrypted_host = process.env.host;
  const encrypted_database = process.env.database;
  const encrypted_user = process.env.user;
  const encrypted_pass = process.env.pass;
  kms.decrypt({ CiphertextBlob: new Buffer(encrypted_host, 'base64') }, (err1, host) => {
    kms.decrypt({ CiphertextBlob: new Buffer(encrypted_database, 'base64') }, (err2, database) => {
      kms.decrypt({ CiphertextBlob: new Buffer(encrypted_user, 'base64') }, (err3, user) => {
        kms.decrypt({ CiphertextBlob: new Buffer(encrypted_pass, 'base64') }, (err4, pass) => {
          const db = pg({
            "host": host.Plaintext.toString('ascii'),
            "port": 5432,
            "database": database.Plaintext.toString('ascii'),
            "user": user.Plaintext.toString('ascii'),
```

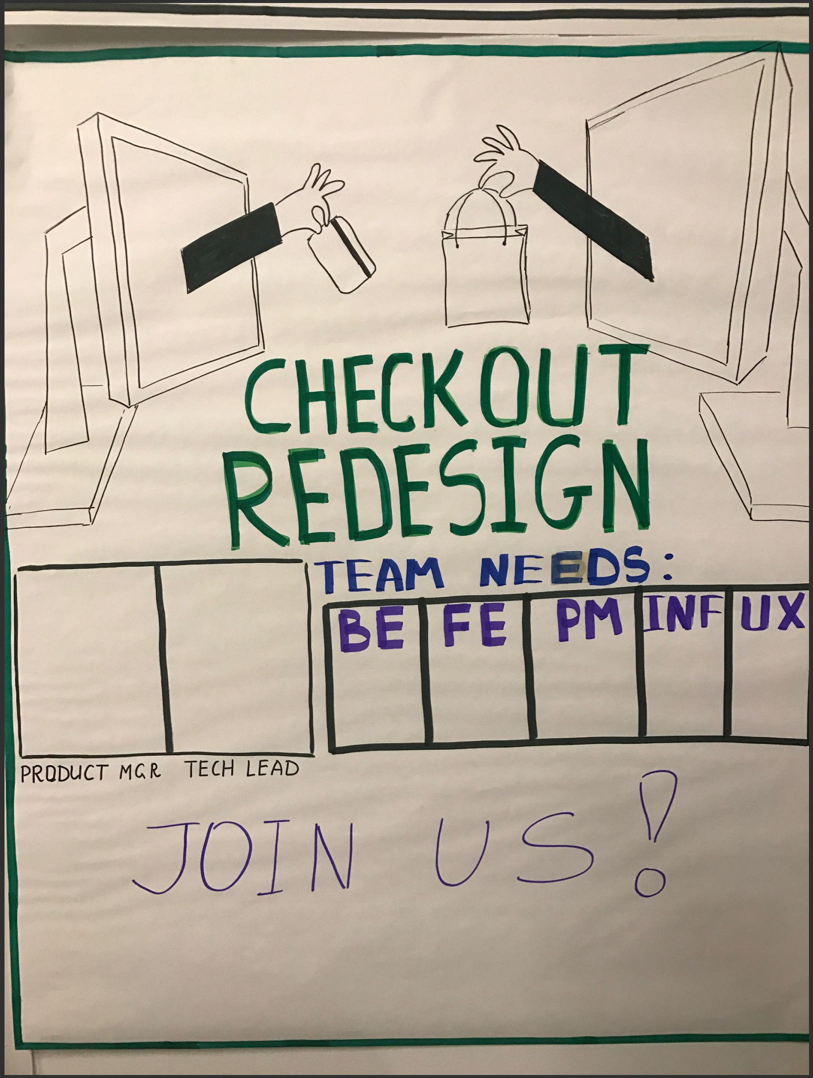
It’s just code.

f: Forced team choices.

Prefer self-selection.

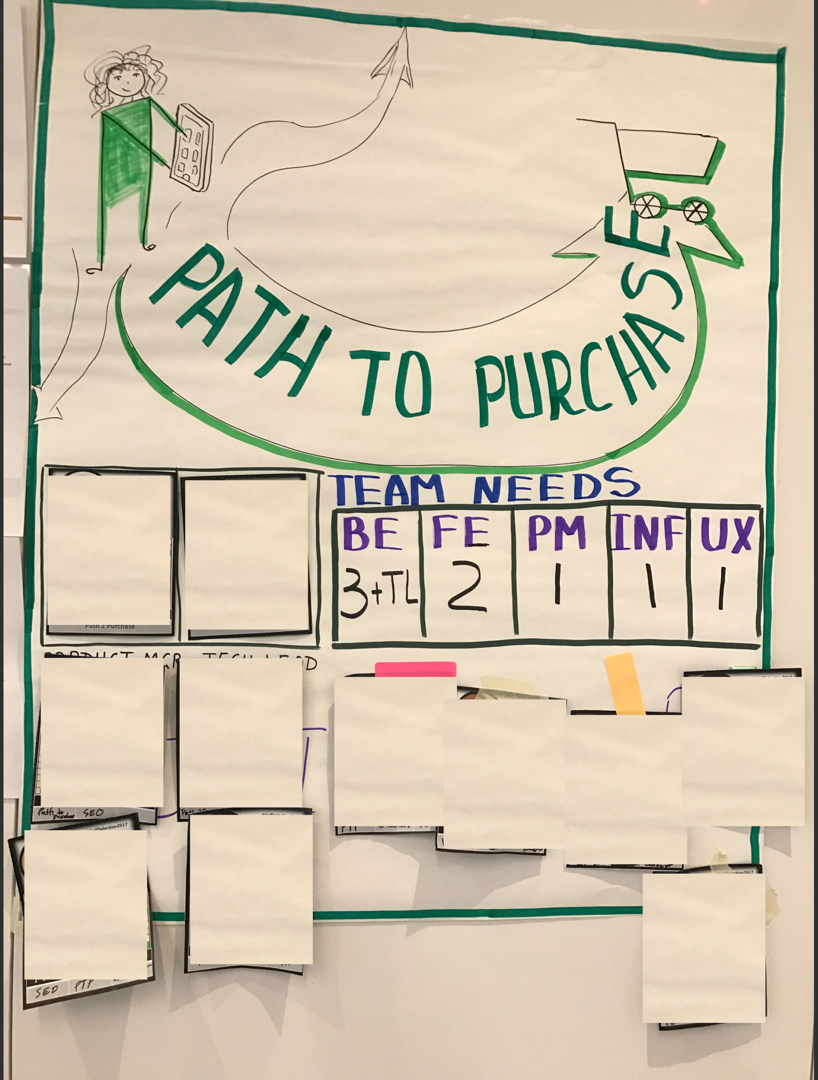
Self Selection

Product Mgr, Tech Lead &
Project Mgr 'pitch' to engineers.



“I love the team I’m on right now!”

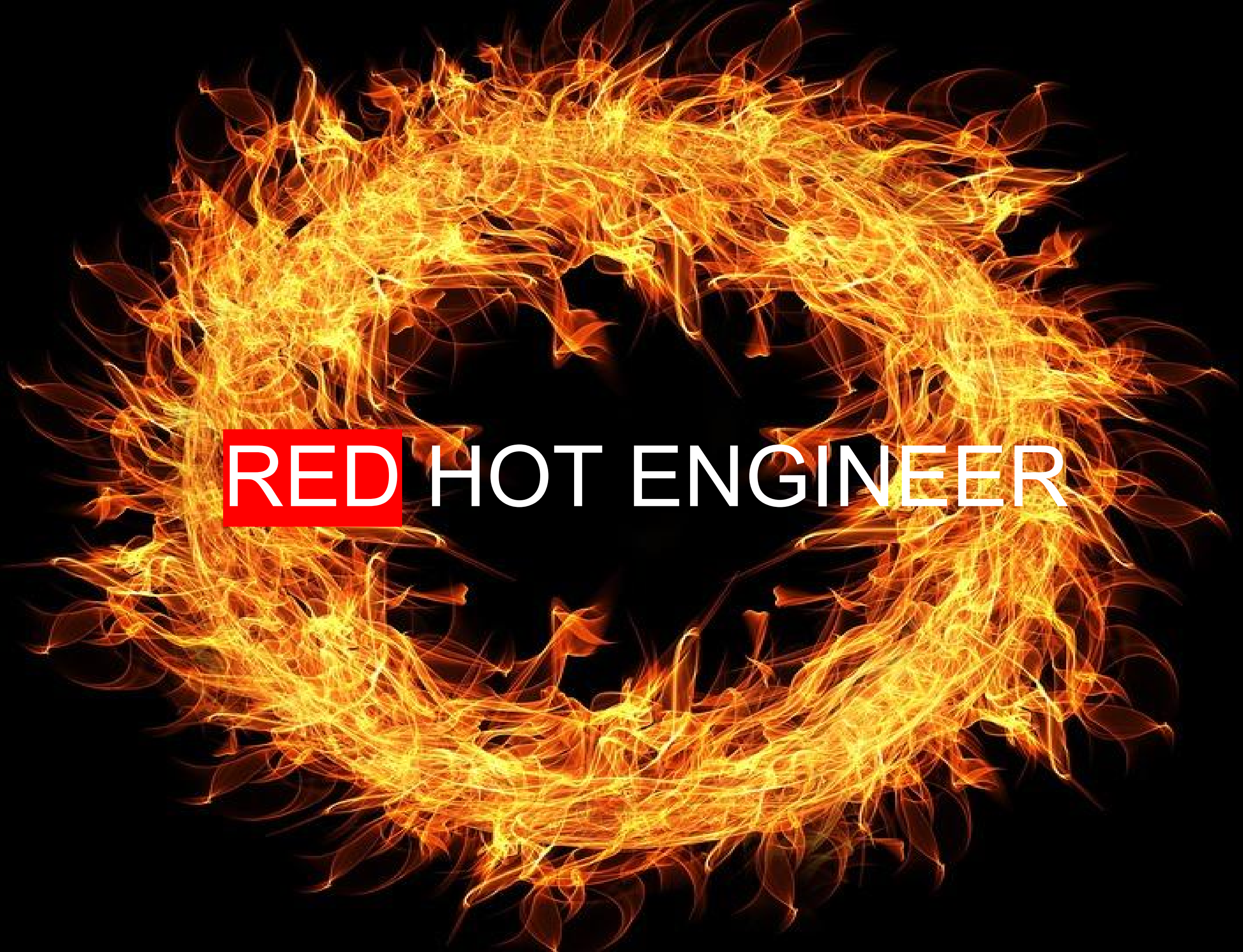
Imagine the power of a fully-aligned team who want to work together.



f: Distractions.

Reinforce the notion that *coding is the primary activity*.

RED HOT ENGINEER



Work your meetings

5@4 (~3w, by location)
Tech Huddle (weekly, by location)
All Hands (monthly, global)
Team KPI meetings: 2-4 weeks
Quality Review
Team meetings? Up to them.

Ask: “was this meeting valuable? should we meet again?”

~ 2.75 - 5 hrs a week

Measure It.

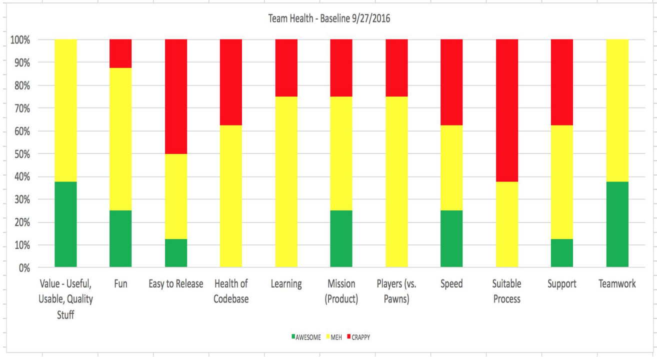
POps Mission

To build and maintain the best product development teams in the world through establishing the models around how we staff and organize our teams, how we plan and execute our work, and how we develop our people and our culture.

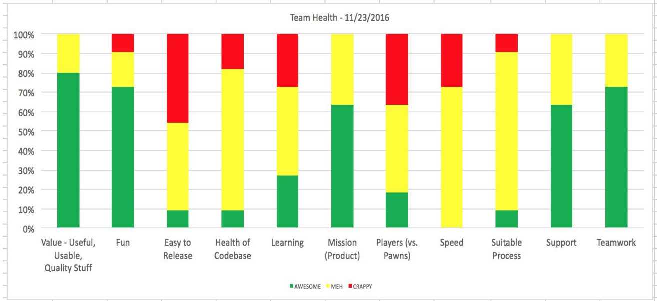
Reduce the Friction in the Employee Experience!

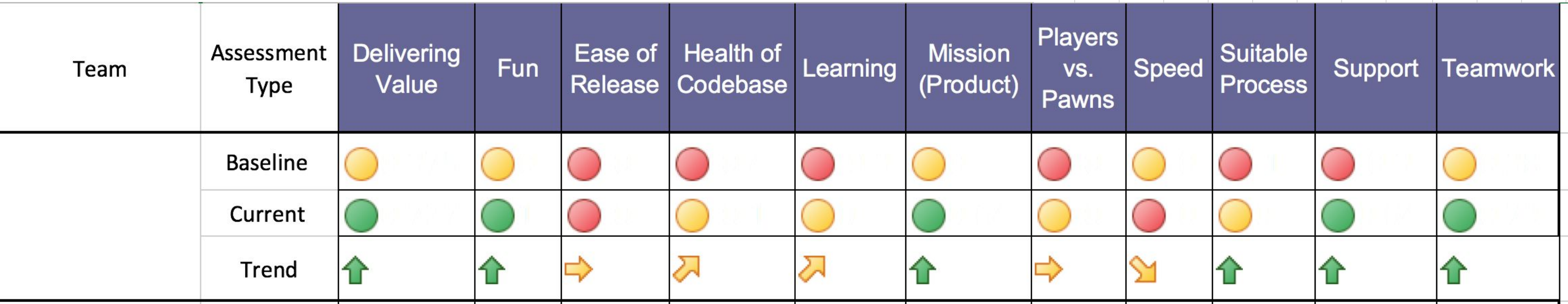
Team Health Check - Trends

Baseline
9/27/2016



Current
11/23/2016





Team	Assessment Type	Delivering Value	Fun	Ease of Release	Health of Codebase	Learning	Mission (Product)	Players vs. Pawns	Speed	Suitable Process	Support	Teamwork
	Baseline	🟡	🟡	🔴	🔴	🔴	🟡	🔴	🟡	🔴	🔴	🟡
	Current	🟢	🟢	🔴	🟡	🟡	🟢	🟡	🔴	🟡	🟢	🟢
	Trend	⬆	⬆	➡	↗	↗	⬆	➡	↘	⬆	⬆	⬆

Seek out and *remove* friction in your engineering process.

Give *freedom-of-choice* & *freedom-of-movement* to your engineers.

Code is the *primary* artifact.

Minimize the distance between “hello, world” and prod.

#thanks @adrian_trenaman @gilttech @hbcdigital

Muda - “Waste” in manufacturing process

Overproduction

- Routinely exceed customer needs ("gold-plating")
- Exceeding scope of SLAs

Intellect

- Mismatched work functions with skill sets
- Lack of best practice sharing across groups

Waiting

- Idle time during automated program runs
- Waiting between assignments

Overprocessing

- Unnecessary system replacement, patching
- Backup/defrag runs earlier than needed
- Excessive documentation

Motion

- Interruptions leading to context switching, mental motion
- Lack of or sub-optimal Standard Operating Procedures (SOP)

Rework

- Misrouted tickets
- Inadequate testing before production
- Poor change-window planning

Inventory

- Large number of servers due to a low server utilization
- System-generated alerts clogging ticket queues

Transportation

- Multiple handoffs of incidents, changes
- Sub-optimal dispatch and routing
- Insufficient use of remote diagnosis