Kroger.com

Herding Cats at Mach 1

Stephen Shary

github.com/STeveShary
Twitter: @StephenShary
linkedin.com/in/stephenshary

Myself

- Technical Lead
- Full-stack Developer
- Kroger Customer Facing Website
  - JVM (Java, Clojure)
  - JS (ES6).
    - AngularJS 1.x -> React
  - A bit of Golang
  - Databases: Cassandra, Oracle, Redis, Neo4j, MongoDB

What We Will Cover

- Show our growth and Delivery
- Discuss team structure
- Our architecture
- Culture patterns and anti-patterns

Our Start: 2013

- Major push to overhaul site.
- Add E-com (ClickList)
- Add Pharmacy
- Add Shopping List

Our Start: 2013

- Started with 3 teams: Site, List, Pharmacy.
- ~20 people
- Sharepoint -> AngularJS ~1.0.6
- Embedded Tomcat Server
- Java 1.7
- Completed 2 production releases

Now: 2017

- 30+ teams
- 185+ Developers
- 60+ applications
- AngularJS -> React
- New PAAS.
- Prod: ~150 release per month. (1,800 per year)
- Non-Prod: ~1,200 releases per month. (14,000 per year)

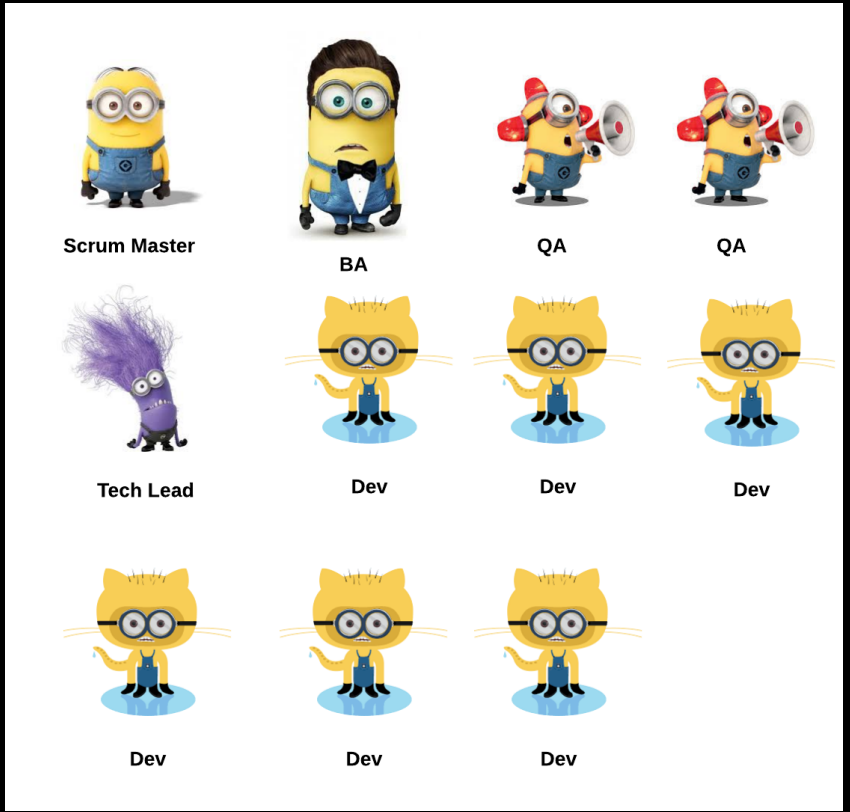
Ohh yeah, don’t forget

- 24/7 uptime.
- Billion Dollar E-com site.
- Over 1 Billion Coupons per year.
- Tens of millions of registered users.

Teams

Teams: Structure

- “two pizza team”
  - Scrum Master
  - Business Analyst
  - 1 to 2 Testers
  - Tech Lead
  - 4 to 8 Devs



Team Traits

- Own their apps.
- Own their support.
- Choose their release schedule.
- Choose their tech stack.
- Choose their name.

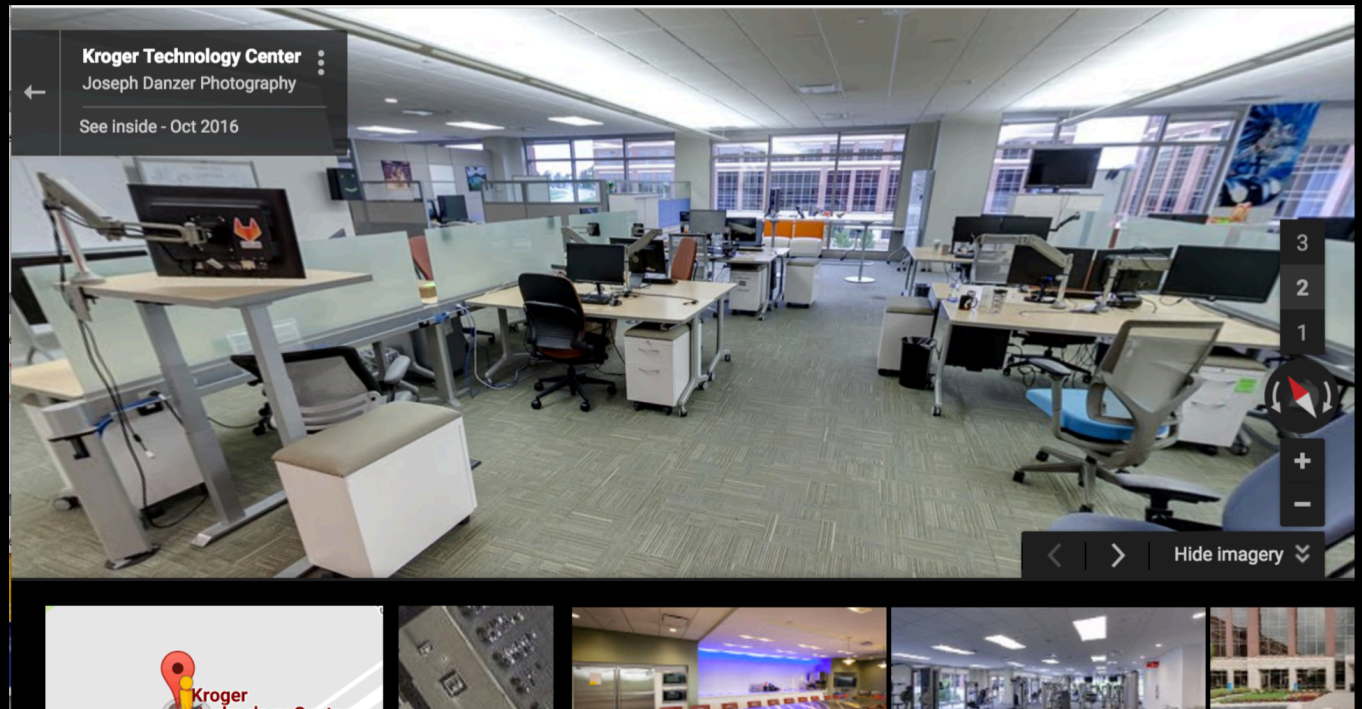
(Some)Team Names

Star Lord
Fury Road
Pharmasaurus Rex
Groot
Ironman
Batman
Punisher
Ronin
Thor
Odin
Night's Watch

Pinky
Pied Piper
Voltron
Good Eats
Old Monks
Brute Squad
Cat Rodeo
Swat Kats
Misfit Toys
Night Fury
Sons of Anarchy
Damaged Sentinels

Our Office

- Google Maps: “Kroger at the Landings”



Team Types

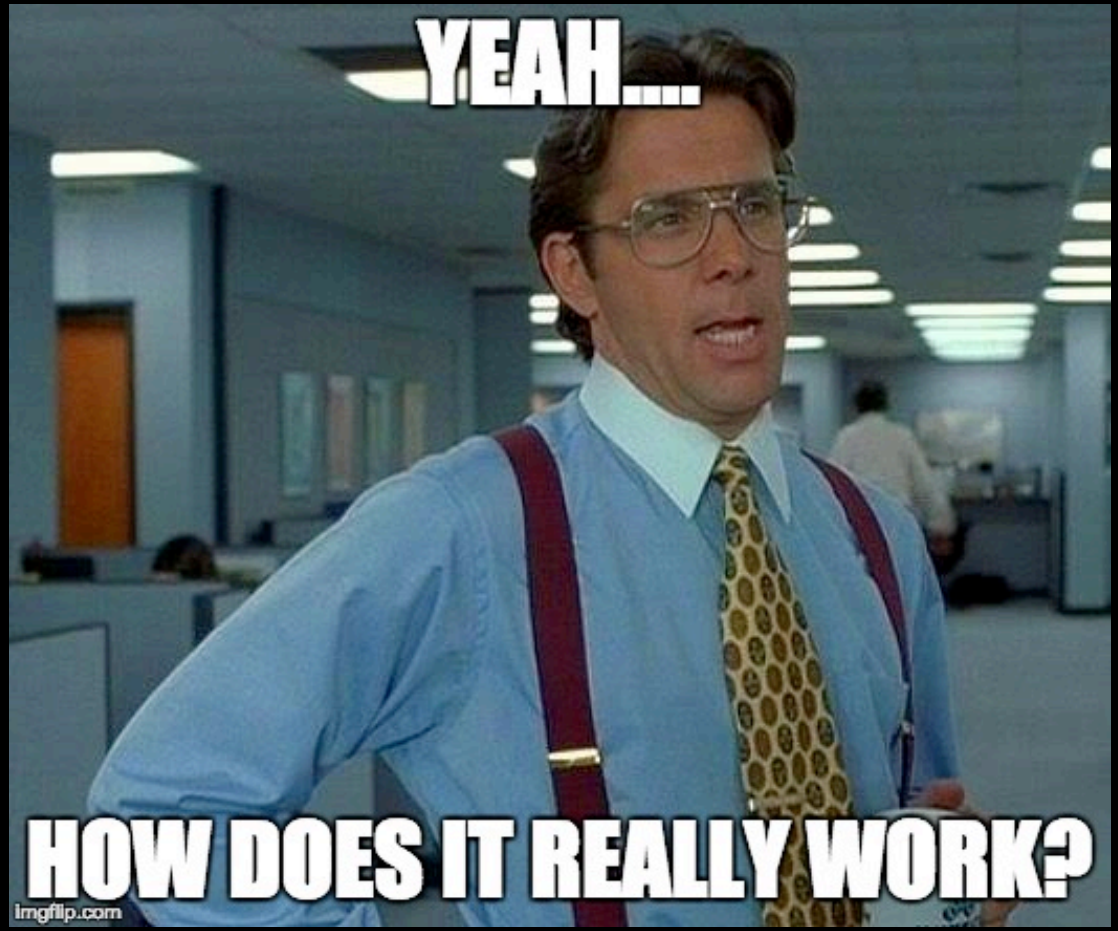
- Two types of teams:
  - Vertical (Owns a domain)
  - Horizontal.
- Inspired by Spotify Model

Vertical Team

- Owns the whole stack.
  - Front End: HTML, CSS, JS
  - Backend: BFF, Web Apps
  - Database
- Owns whole SDLC
  - Dev, Test, Deploy, Support

Horizontal Team

- Provides services to vertical teams.
- Owns common apps
- 4 Teams
  - Ronin - Echo
  - Punisher - Core Engineering
  - Shazam - Core Infrastructure
  - Damaged Sentinels - Core Architecture

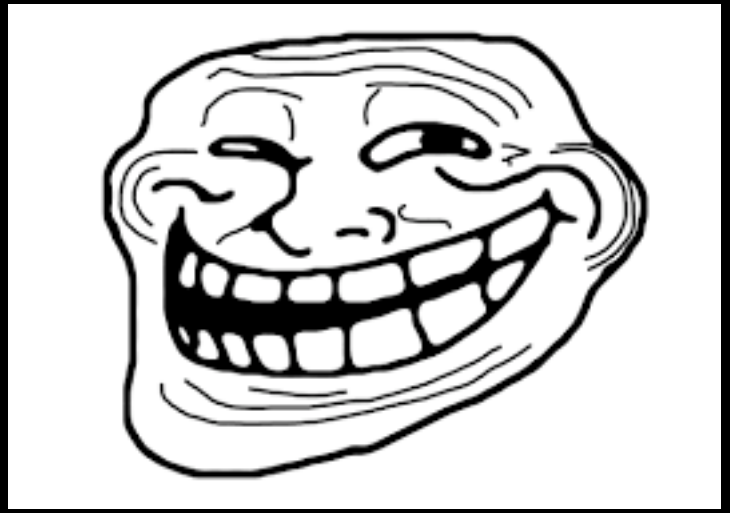

YEAH....
HOW DOES IT REALLY WORK?
imgflip.com

ONE MONOLITH

ALL

JK

LOL



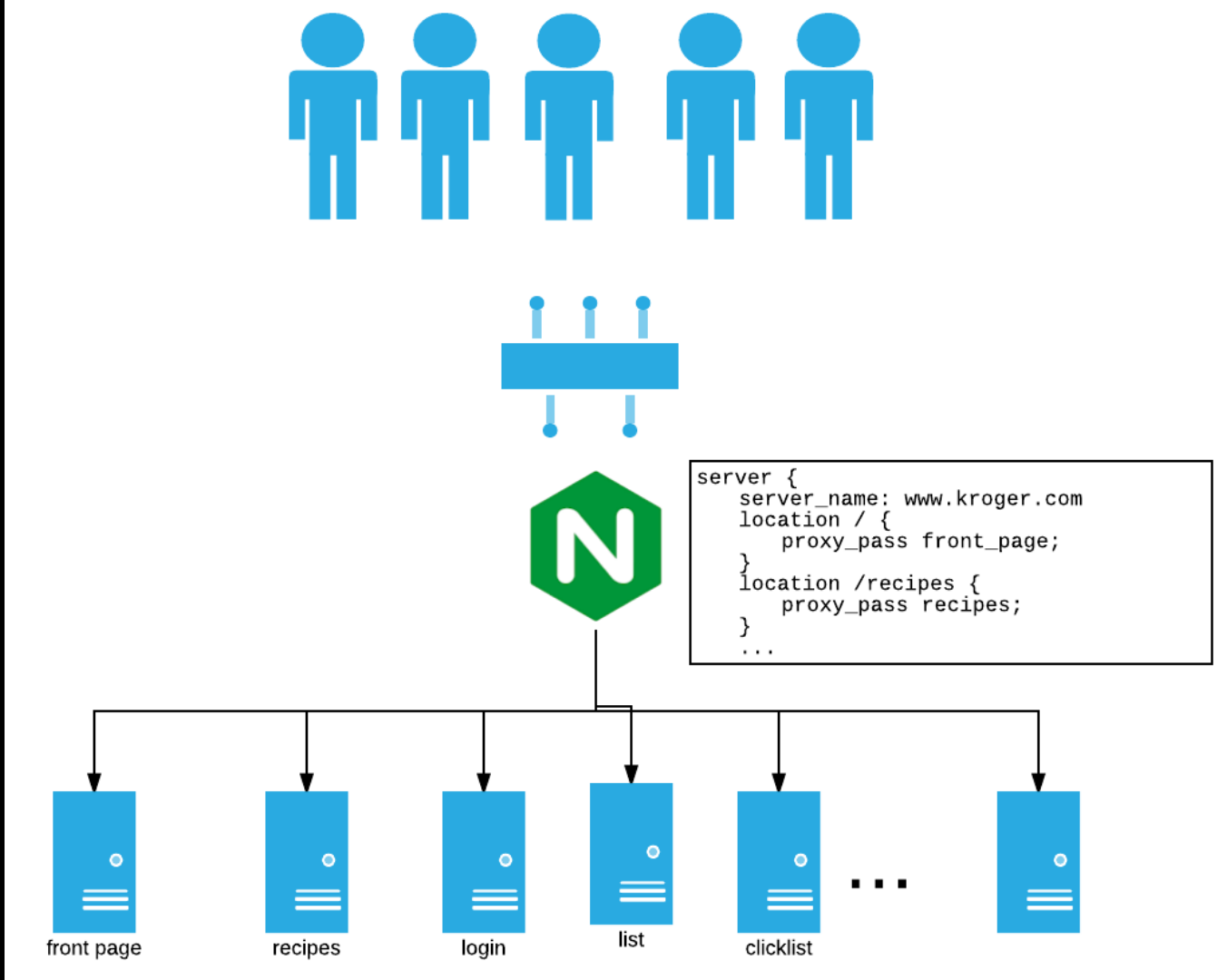
Our Architecture

- Applied Conway's Law

Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations.

But It’s Single Site!

- We have architecture that helps.
  - Reverse Proxy
  - We use Nginx
  - Full CI/CD pipeline with 4,000+ e2e tests.


server {
server_name: www.kroger.com
location / {
proxy_pass front_page;
}
location /recipes {
proxy_pass recipes;
}
...
front page
recipes
login
list
clicklist

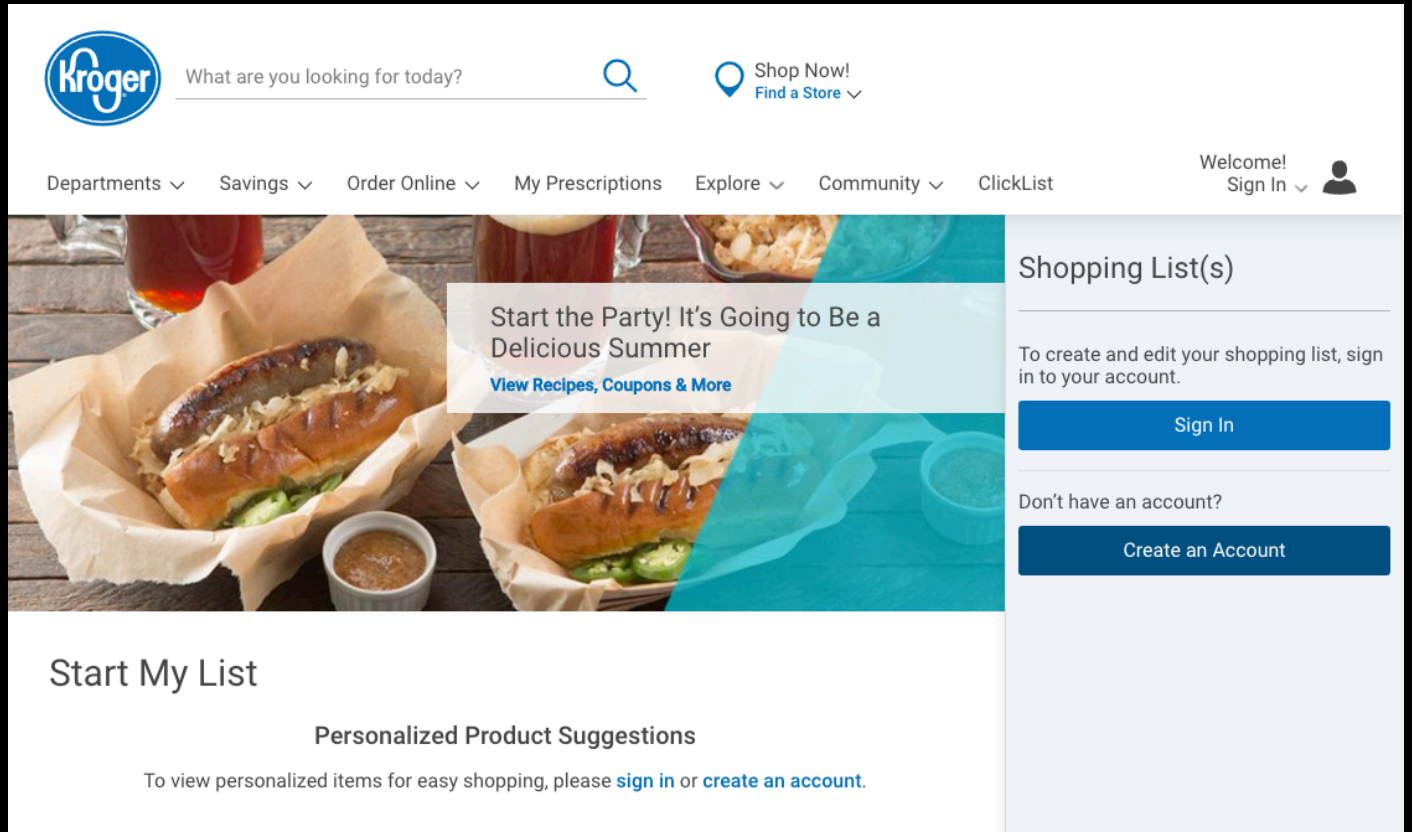
Orchestration

- Software Define Networking
- HA clustering with auto-restart
- Auto-scaling
- Allows
  - Blue/Green
  - Dark

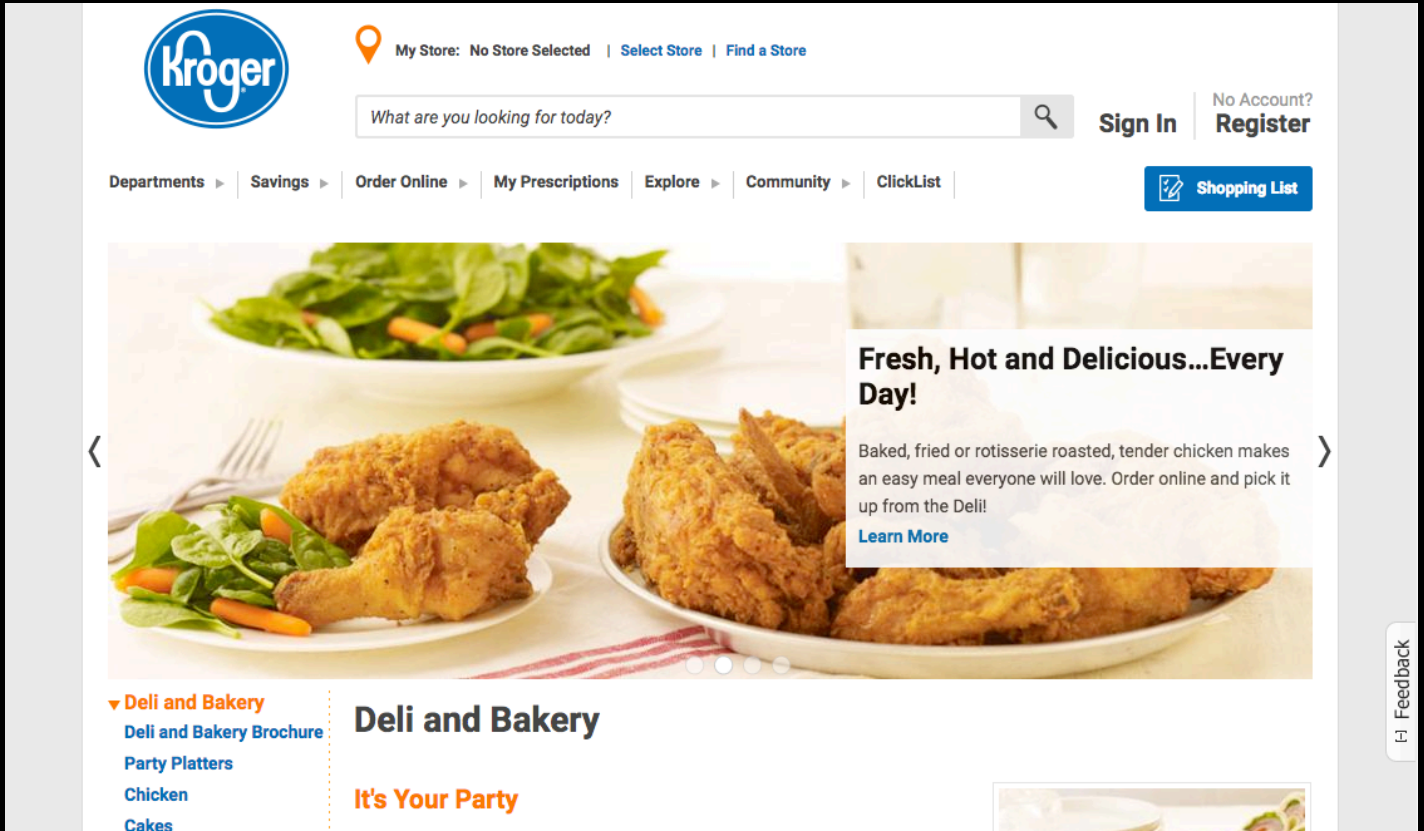
Yeah… But what about

- Compatibility?
- Consistency?

Home Page


What are you looking for today?
Shop Now!
Find a Store
Departments
Savings
Order Online
My Prescriptions
Explore
Community
ClickList
Welcome!
Sign In
Start the Party! It's Going to Be a Delicious Summer
View Recipes, Coupons & More
Shopping List(s)
To create and edit your shopping list, sign in to your account.
Sign In
Don't have an account?
Create an Account
Start My List
Personalized Product Suggestions
To view personalized items for easy shopping, please sign in or create an account.

Hey….. Is that different?



Culture points

- Bring the pain forward.
- Monolith first.
- Reduce risk.
- Automate Everything.
- Make the best path the easiest.

Bring the Pain Forward

- Fix the problem.
- Don’t mask the symptoms.
- What is pain?
  - Paperwork
  - Downtime
  - Manual Testing
  - Process ceremonies

Monolith First

- Don’t make micro services a buzzword
- Split when the team is too big.
- Split when the app crosses two domains.
- Split when the pipeline is too slow.
- Split when you don’t know all the code in one app.

Reduce Risk

- Small changes are better.
- Practice deployments.
- Spend time on backwards compatibility.
- Expect to roll back.
- Toggles.
- QA in Prod (Dark)

Automate Everything

- Bring the pain forward
  - Deployments
  - Tests
  - Workflows
  - Logging -> Metrics -> Feedback
  - Provisioning
  - Timesheets?... Expense Reports?

Make the Best Option the Easiest

- Developers are lazy.
- We choose the wrong tool to get done.
- Recommend, don’t mandate.
- Look at adoption rates.

How to fail

- Make it manual (automation doesn’t pay off)
- Deploy together
- Make deployments = downtime
  - Over night deployments
- The process requires meetings
- The beatings will continue until stability improves.

Questions?

Thank You

Stephen Shary

github.com/STeveShary
Twitter: @StephenShary
linkedin.com/in/stephenshary