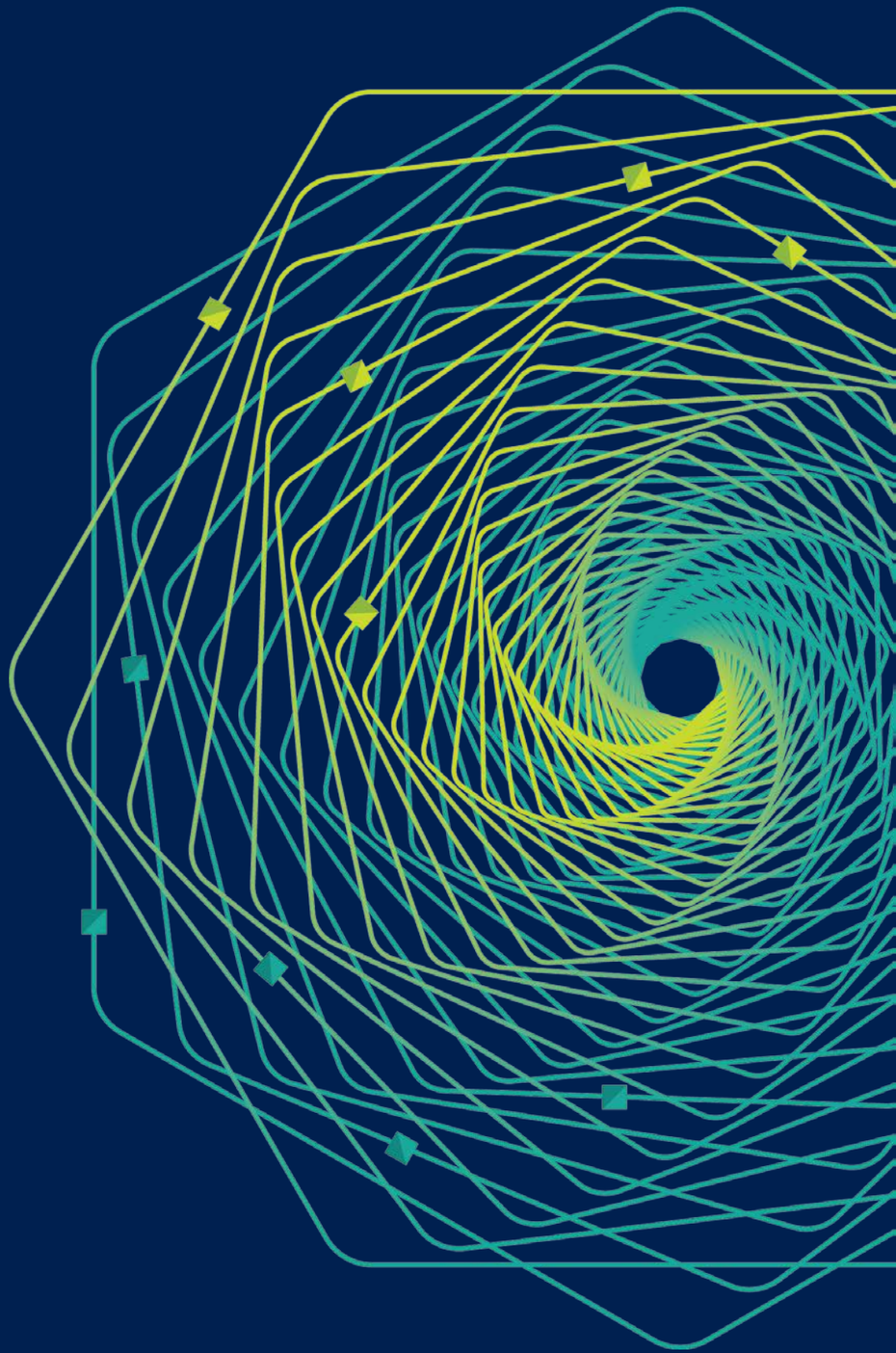
Microsoft

Research Faculty Summit 2018

Systems | Fueling future disruptions

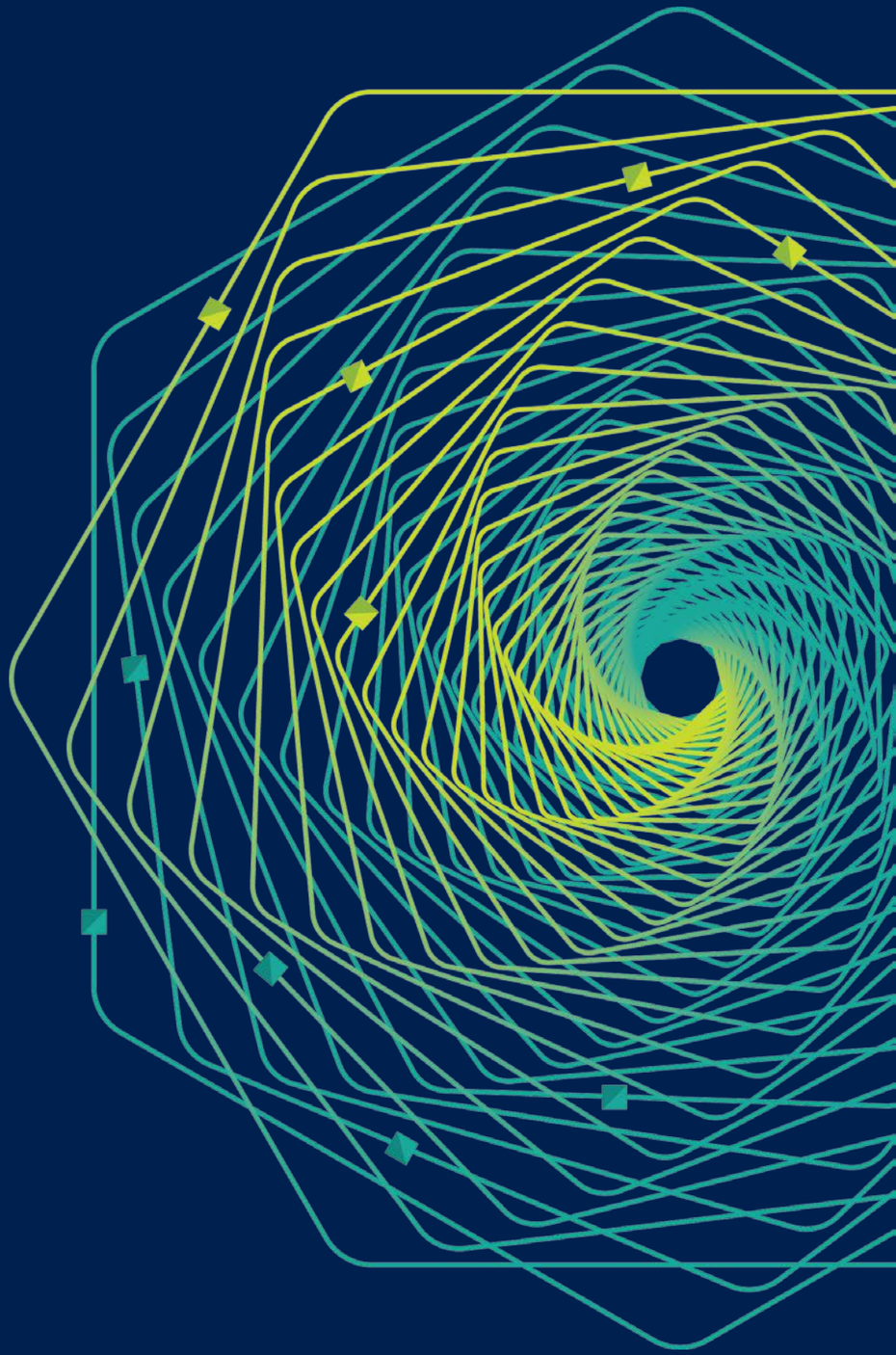


Microsoft

Continuous Delivery for Bing UX

Chap Alex

Engineering Manager, Microsoft



Core Bing-wide Principles

Live-site quality is paramount

- DevOps only, responsibility lies with individuals

Constant innovation

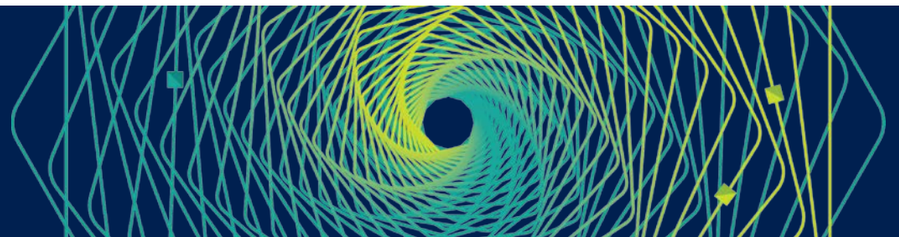
- We're the underdog but expected to win

Data rules

- Experiments with real users dictate what eventually ships

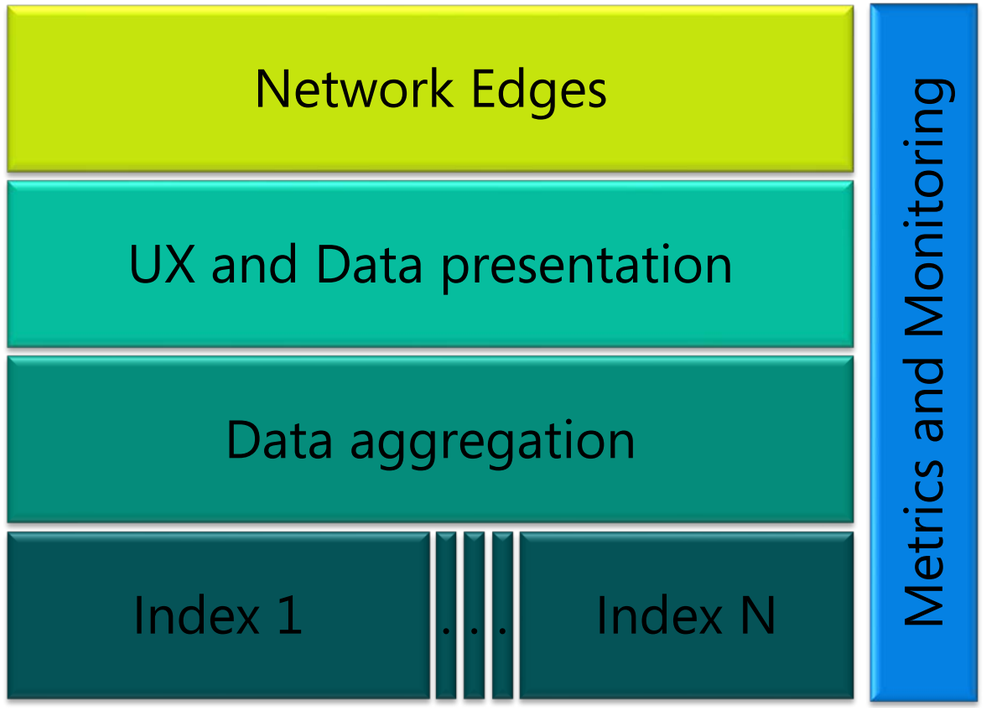
Search is an expensive business

- Do more with less



Bing Serving Stack

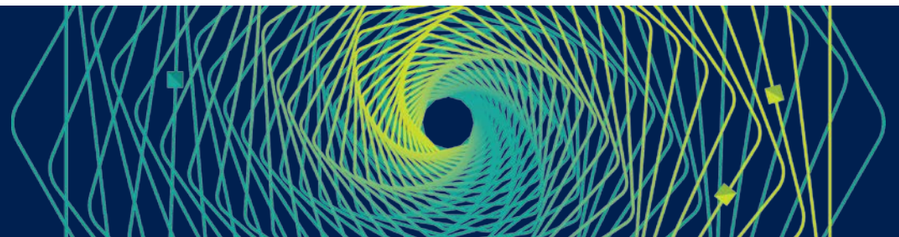
- Billions of queries per day
- Sub-second page load time
- Geo-distributed data centers
- Multi-tiered serving stack
  - Ingress, routing, throttling
  - UX rendering tier
  - Data query and aggregation tier
  - Specific data indexes
  - Offline metrics and monitoring



Bing UX Tier

- Monthly distinct UX developers
  - Many hundreds of developers
  - Exponential growth
- > 100 changes per day
- < 100ms latency @ 95th
- Renders
  - HTML/JS/CSS/JSON/XML/Images
  - Bing/Cortana results and features
  - Bing/Cognitive APIs
  - Partners: Yahoo, Apple, AOL, etc.

UX and Data presentation



Why did Bing invest in Agility?

A question of survival for Bing

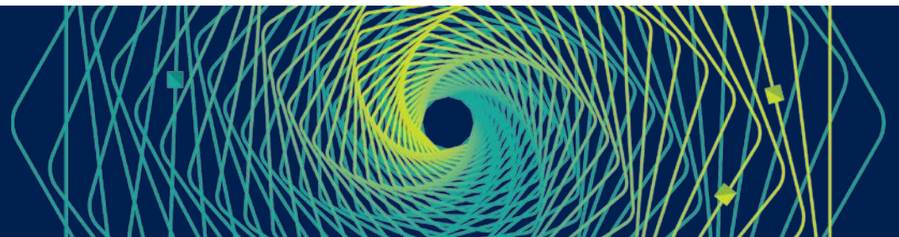
- We are competing with established market leaders
- We need to grow and thrive to survive

How does Bing improve, grow, and capture market share

- We run experiments on the live site. A lot of experiments.

How can we run more experiments?

- Dramatic increase in experimentation
- Dramatic increase in number of developers creating experiments
- Dramatic increase in developer productivity—more output!



Agility's Impact on Bing

Impact on engineering

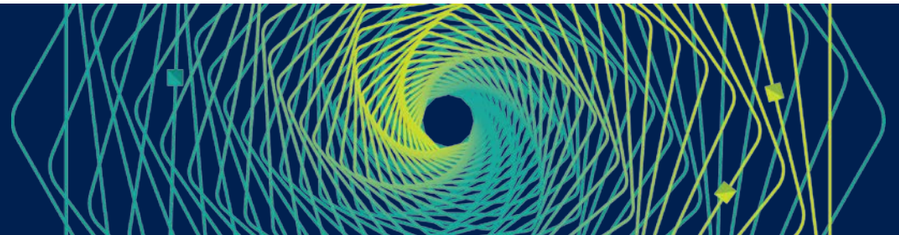
- Developer satisfaction and productivity dramatically improved
- Significantly scaled team size and feature complexity
- Agility pipeline has scaled quickly and consistent

Impact on Live Site

- Number of live site incidents dropped from 6+ monthly to 1 or less
- Financial impact of incidents dropped as well
- Identification of issues is simpler through better granularity

Impact on code base

- Code quality improved through smaller, more frequent changes
- Code reviews shorter and more focused on changes
- Quality-empowered reviewers guarantee attention to testing



Pre-Agility Metrics

Audience
~80 engineers

Development
90 min build
30 min start
60 min test

Source Control
5 dev repos
1 main repo
1 release repo

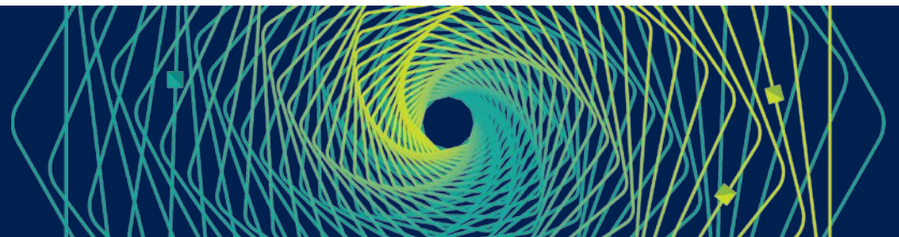
Release Cadence
1X per month

Test collateral
3K tests
80% pass
Ad-hoc execution

Multiplexing
<3 browsers

PROD Scenarios
Bing web results

Experiments
10's per month



Pre-Agility Metrics

Audience
~10X engineers

Development
15 min build
5 min start
20 min test

Source Control
1 GIT repo

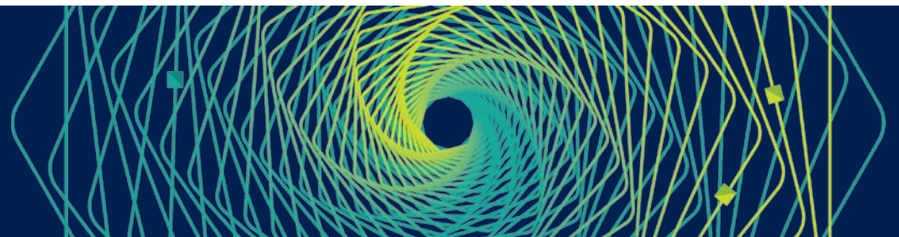
Release Cadence
14X per week

Test collateral
43K tests
99.99% pass
Auto execution

Multiplexing
14 browsers
34 devices
16 OS clients

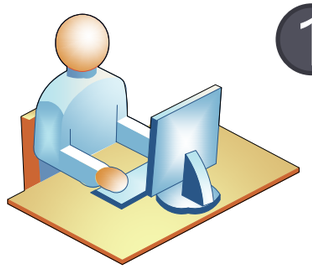
PROD Scenarios
Bing.com (all)
Windows 10
Cortana
APIs
Mobile
XBOX

Experiments
1000's per month



Small and Frequent Changes

Continuous Integration



1 Start Pull Request...

VSTS CI

2 Picks up PR

Build

Test

3 Deploy to TIP environment

4 Tests gate check-in

Continuous Delivery

1 2 3 4 5 6 7 8

5 Deploy to staging environment

VSTS CD

6 Ship gated by 100% pass

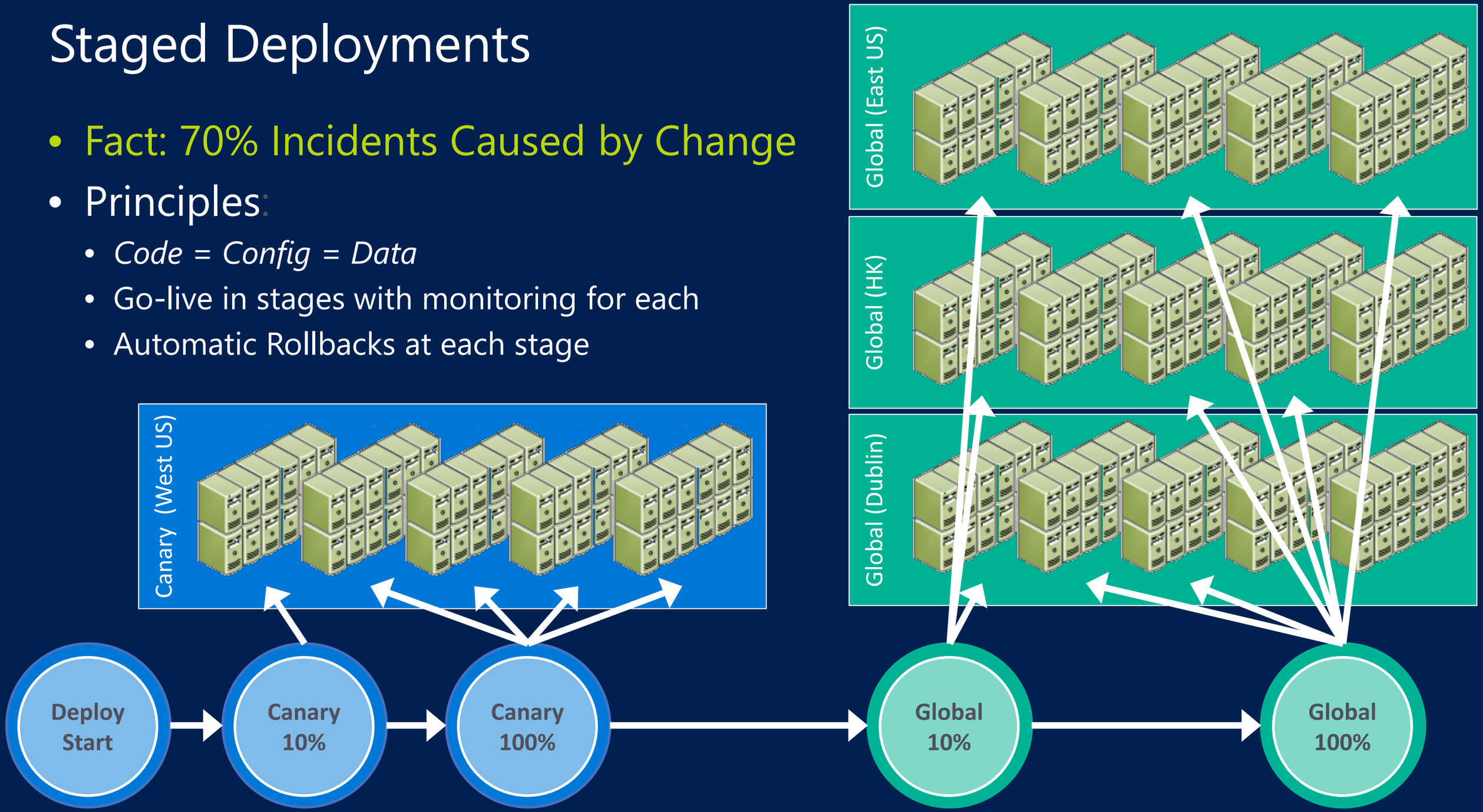
Production Environment

Selenium Grids

Staging Environment

Staged Deployments

- Fact: 70% Incidents Caused by Change
- Principles:
  - *Code = Config = Data*
  - Go-live in stages with monitoring for each
  - Automatic Rollbacks at each stage



The reality of Large-scale Agility

We break things everywhere

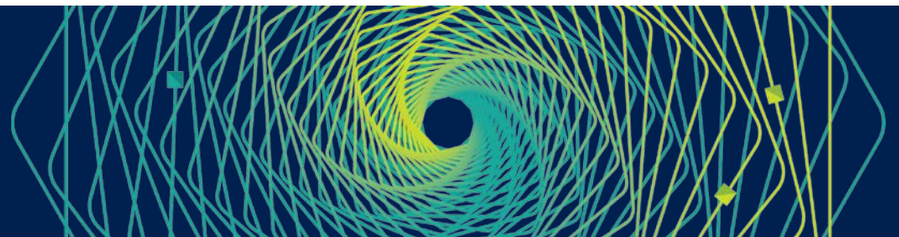
- Our feature test load is often higher than live site volume
- Our codebase is the third largest GIT repo at Microsoft
- Our Azure storage footprint is over 400TB—for test!

We improve everything we can

- Build times improved though target caching and code organization
- Test time improvement through massive parallelization
- Scale magnifies even modest improvements and **regressions**

We play well with others

- Our discoveries flow back to other teams like .NET and VSTS



Key Learning: Fewer moving parts

Single code repo

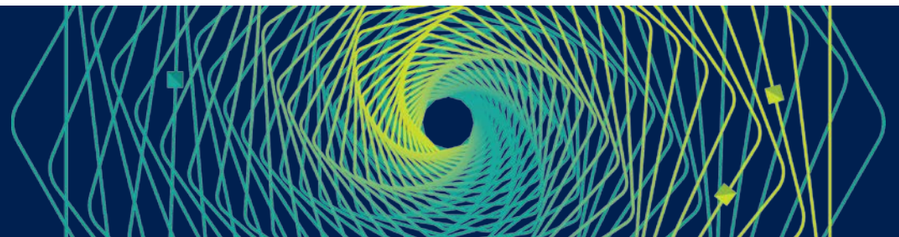
- Continuous modernization of the codebase and its dependencies
- Ship code in development side-by-side with live code (isolate via config)

One environment to monitor (Production)

- No long-standing “test” environments mirroring branches
- Less to configure and maintain—and less misconfiguration

One validation gate

- Depend wholly on automated testing prior to code submission
- All tests must pass or the **pull request is rejected**
- Tests are co-located with code



Key Learning: Optimize CI

Take every optimization possible

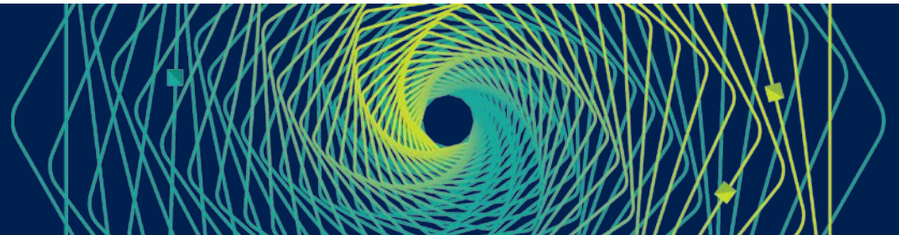
- Drive builds to single digit minutes, validation to low tens
- Start validation of a change while developers are still code reviewing
- Optimize validation by being selective about what tests get run
- Mitigate risk of selective test by running full passes on the Deployment Loop

One environment to monitor (Production)

- No long-standing “test” environments mirroring branches
- Less to configure and maintain—and less misconfiguration

One validation gate

- Depend wholly on automated testing prior to code submission
- All tests must pass or the pull request is rejected
- Tests are co-located with code



Key Learning: Optimize CD

Reuse CI mechanics

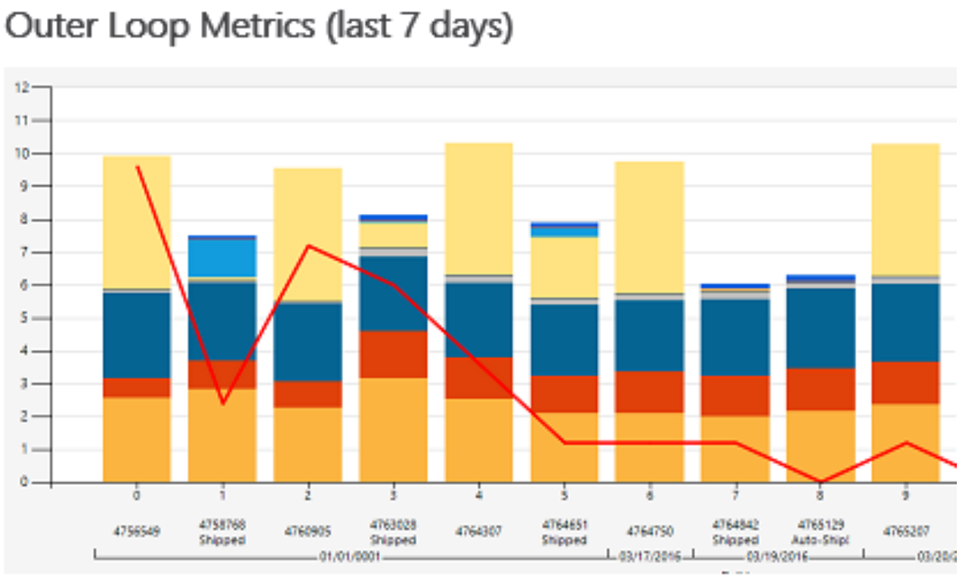
- Build and Validation stages can be re-used
- Discovery of issues halts that build's progress—though not the loop

Measure all aspects of CD

- Build times, validation time, deployment time, etc.
- Alert on regressions to guide optimizations
- Expose for public viewing and review

Remove humans from CD

- Agility DRI chases bug owners only
- Fully automate deployment to PROD
- PROD DRI can focus on true live site issues



Key Learning: Implement. Measure. Iterate.

Plan for constant improvements

- Engineers expect the system to continually improve—even today
- Feature teams always follow the path of least friction—loyalty is a myth
- Every single line of pipeline/platform code is eventually inefficient

Multiple feedback mechanisms

- Engagement varies wildly among developers
- Balance ratings systems with verbatim feedback
- Public forums expose the problems and their size to both sides

Documentation and Training

Voting for Topics

845 votes	Provide a standard solution for Documentation that is reliable and discoverable
332 votes	Provide a Learning & Development platform where engineers can find and take the relevant trainings

Verbatim Feedback

"Engineers should be **encouraged to keep documentation and code comments up-to-date**."
"**Focus on best practices and mandatory training**. Enable fair use policies for resources."
"The **Wiki was out-of-date** and difficult to search for useful information there."
"Better documentation, more tools that work on mobile."

Documentation and Training

Voting for Topics

501 votes	Provide a dev environment in cloud (VM) that enables a new-hire to setup in mins, enables engineers to work from anywhere and anytime
258 votes	Provide support for building, testing and operating containerized microservices (Linux / Windows) (nodejs/C#/python/etc.)

Verbatim Feedback

"Bing Build tools, **data aggregation tier. Painful to submit code and deploy**. PAINFUL!!!"
"If they could **make build really fast**, it would improve experience … with UX tier"
"Recent **switch to GIT and VSTS PRs made experience much better**"
"**Local development should be moved to cloud** as much as possible"

Thank you!

Microsoft