Who we are

Ann Marie

- DevOps and Security Lead, Commerce Platform
- 15+ years of software development
- DevOps evangelist since 2011
- 5+ years of production ops
- 3 years as a development manager
- 2 years leading DevSecOps program
- 1 year as privacy and security lead

Who we are

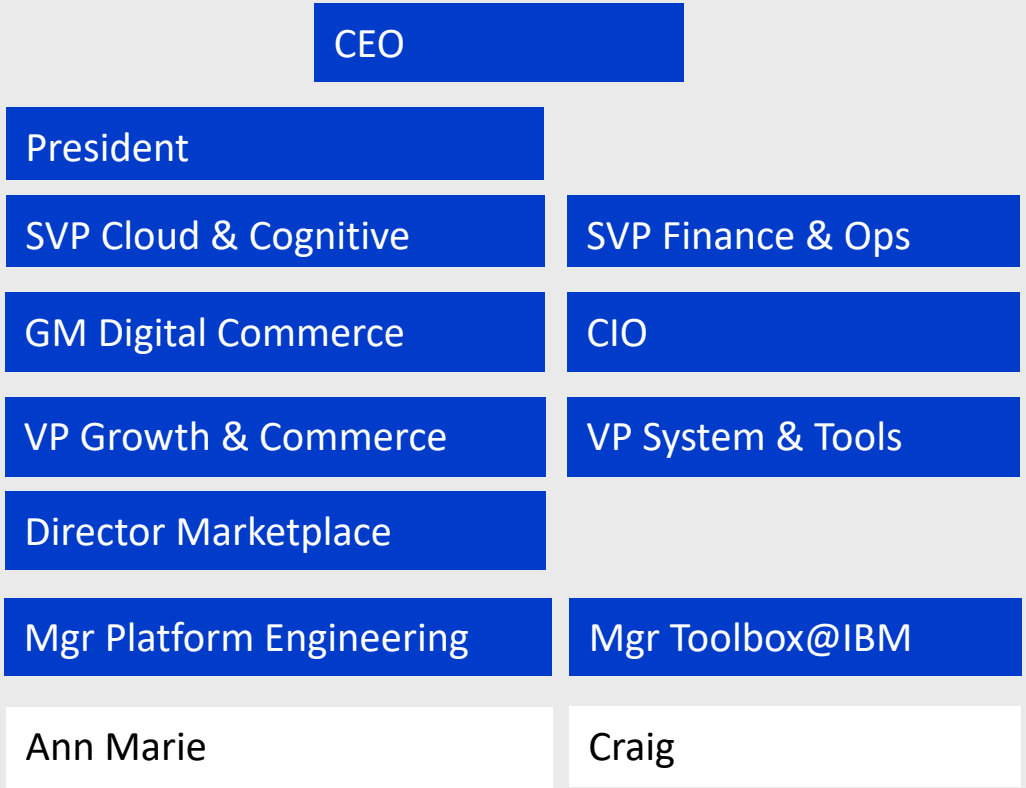
Craig

- Prior role: DevOps Coach
- Influenced 50+ squads without direct authority
- Technical Lead for Toolbox@IBM Operations Tooling
- 20+ years infrastructure experience
- 2+ years management experience
- DevOps Days Raleigh organizer
- Improv performer

IBM legal disclaimer

Where we fit in IBM's organizational structure

- Size – 300,000+
- Leading IBM's Digital Transformation
- Power IBM's Software as a Service platform
- Spotify Squad Model
- Test and Operations squad



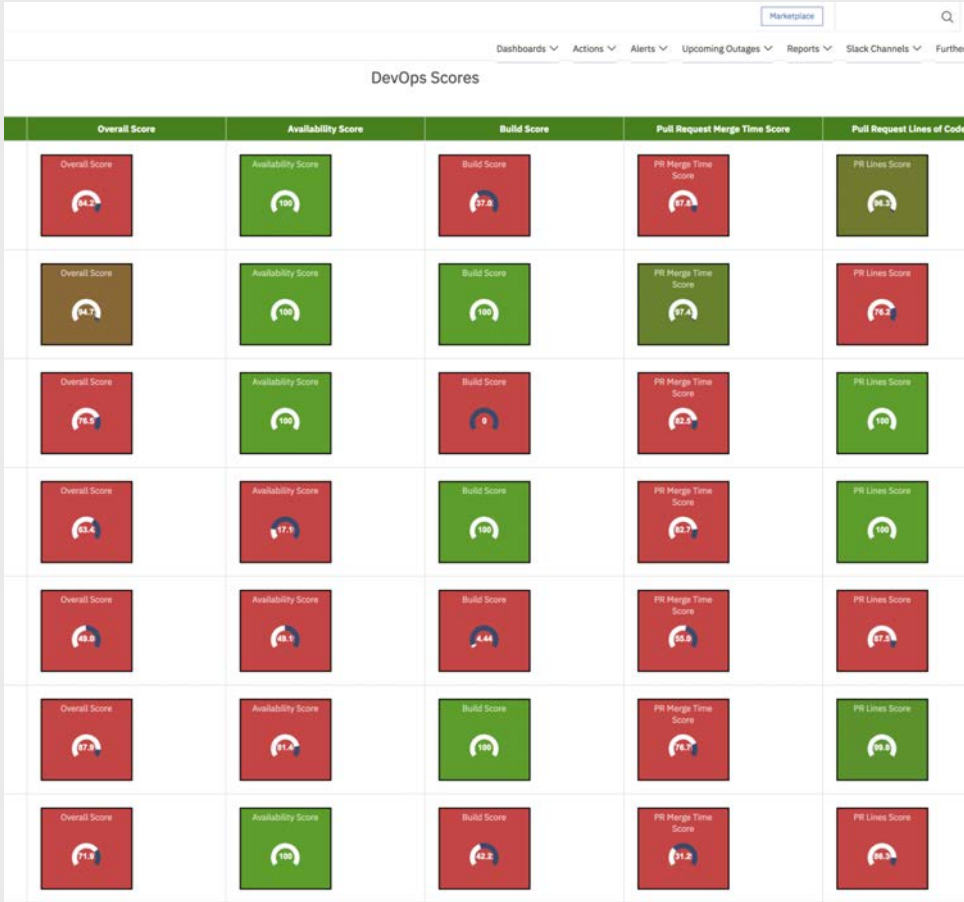
The Ask

Velocity Dashboard

A good idea…

Squad Metrics

A better idea!



Roadmap

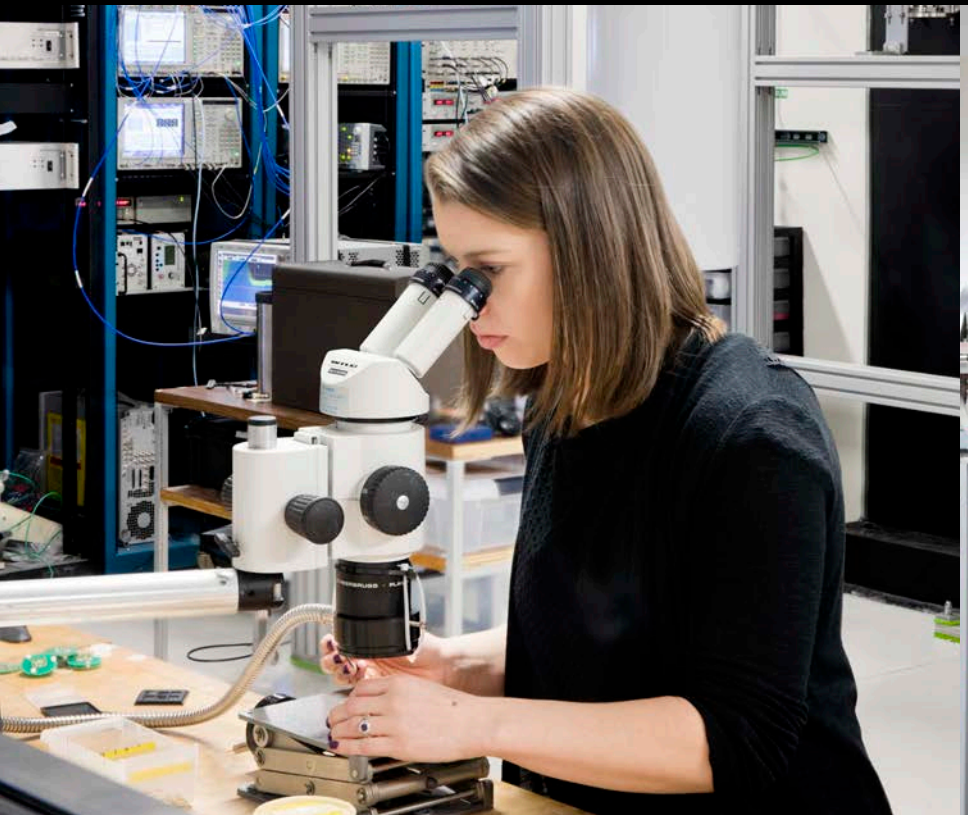
I. Discovery Phase

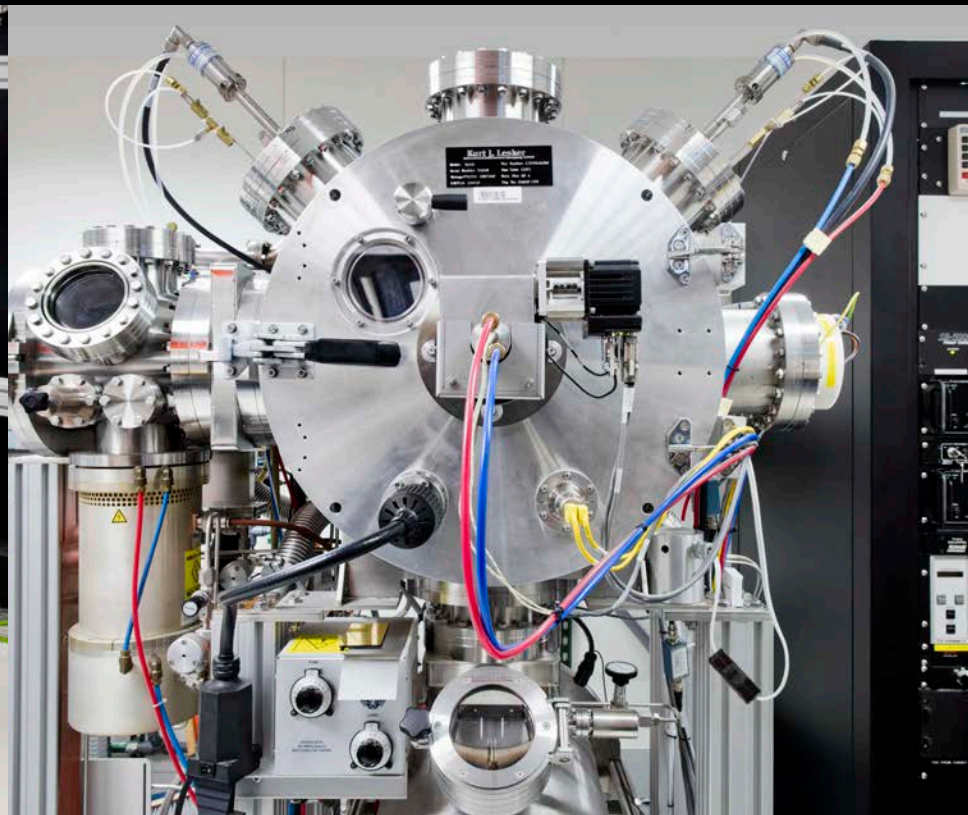
II. Prioritizing Metrics

III. Metrics in Detail

IV. Outcomes

1. Discovery Phase





Discovery Phase

Pre-built vs. Custom Built

Discovery Phase

Researching Metrics

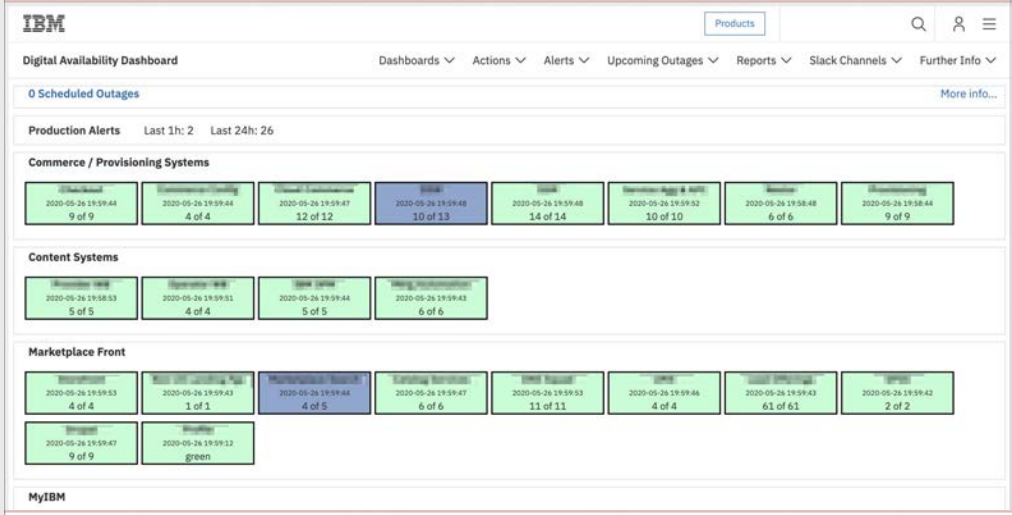
We needed data demonstrating business value

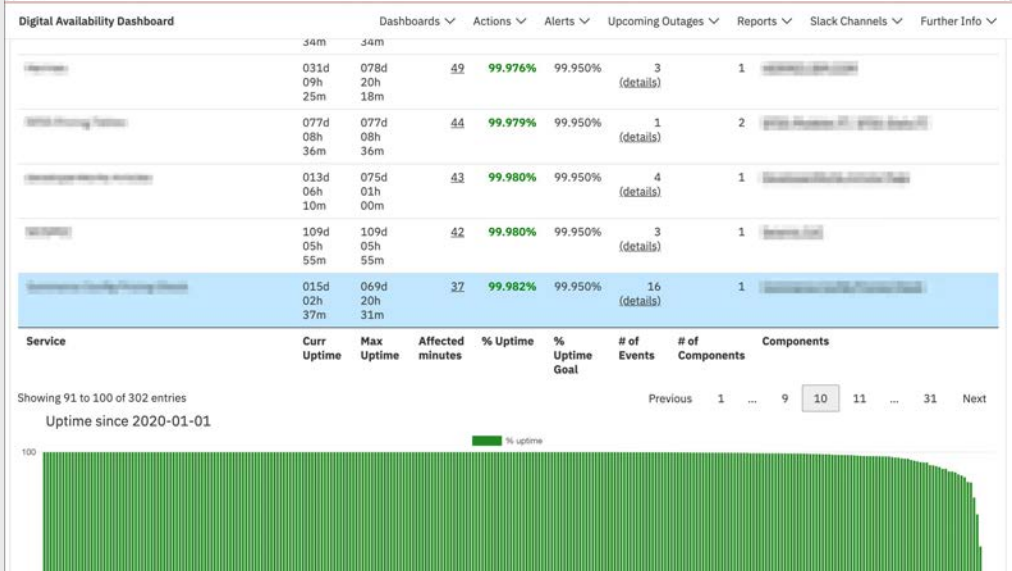
- Flow Metrics from Domenica DeGrandis
- Accelerate book
- Industry standard metrics
- Microsoft blogs
- SonarQube (Security, Code Quality, Test Coverage)
- A few invented based on our pain points (Deploy Stability, Overall, ...)

Discovery Phase

Availability Dashboard

- Monitors availability of over 300 services
- Data from many sources: New Relic, CheckMK, Wiki scrapes, Jenkins jobs, …
- Live and historical views





Discovery Phase

MVP

Cupcake

- What could we have in production in 1 month?
- Simple, useful, fully functional.

Feedback

- What teams loved
- What they feared
- Who are we optimizing for?
- Executive training

II. Prioritizing Metrics

1. Do this first
2. Do this next
3. Do this last
4. Not going to happen

What could we visualize?

Spreadsheet

- Areas
- Categorize
- Priority
- Easy of enabling
- Value

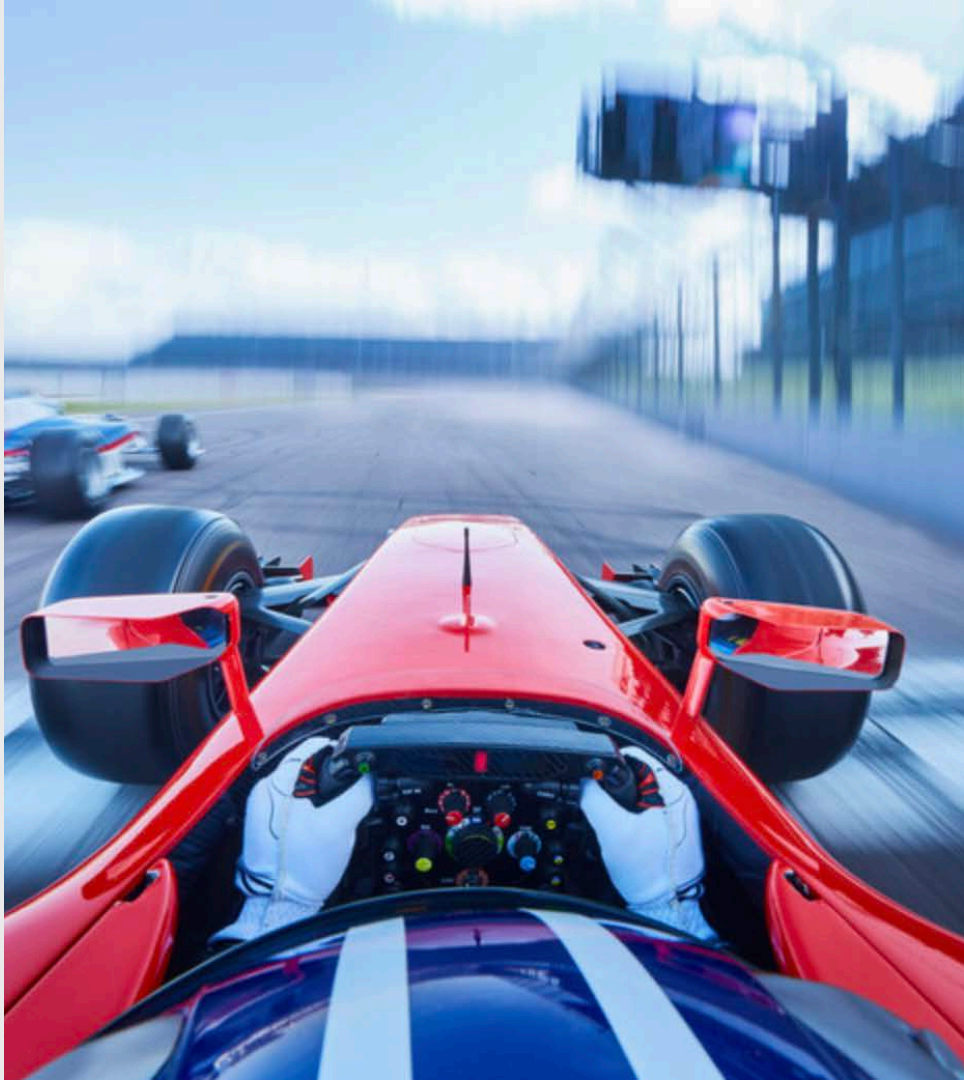
Dashboard Metrics Exec Summary Spreadsheet

Area	Category	Metric	Priority	Ease of enablem	Value Statement	Method of Capture	Enablement	Review Comment	References
Operations	Availability	Uptime / Availability	1st (done)	Easy - already in availability dashboard for most services	High performing services provide high availability of their service, regardless of defined SLO	Availability metrics as captured by the Digital Availability Dashboard	All services availability should be viewable from the Availability Dashboard. Metric will be a simple capture of uptime %.		Industry standard metric
Operations	Availability	SLO Attainment	2nd	Easy - already in availability dashboard for most services	High performing services at least meet their SLO and need to be aware when they are not hitting this target.	Availability metrics as captured by the Digital Availability Dashboard. SLOs can be defined by service within the dashboard as well.	Need all services to define and publicize an SLO. Once done, a sliding score based on attainment of SLO can be calculated. Commerce has started with 100 being 100% availability; 80 being		Industry standard metric
Development + Operations	Value Delivery Speed	Delivery Lead Time: From start of build to live in production	1st	Medium (needed for add'l build services)	Data shows that services with fast delivery times have higher quality, a faster mean time to recovery, and are able to deliver new features more quickly.	Source: Travis, Jenkins, UCD. "Sta	We have this on the DevOps dashboard already for Travis builds. Relabel the "Deployment" score to use this standard term. Need to extend to Jenkins and UCD. Here's a sample query		Industry standard metric

Drive best practices

Fast with safety

- High Availability
- Frequent deployments
- Work in Progress (WIP)
- Test coverage
- Security



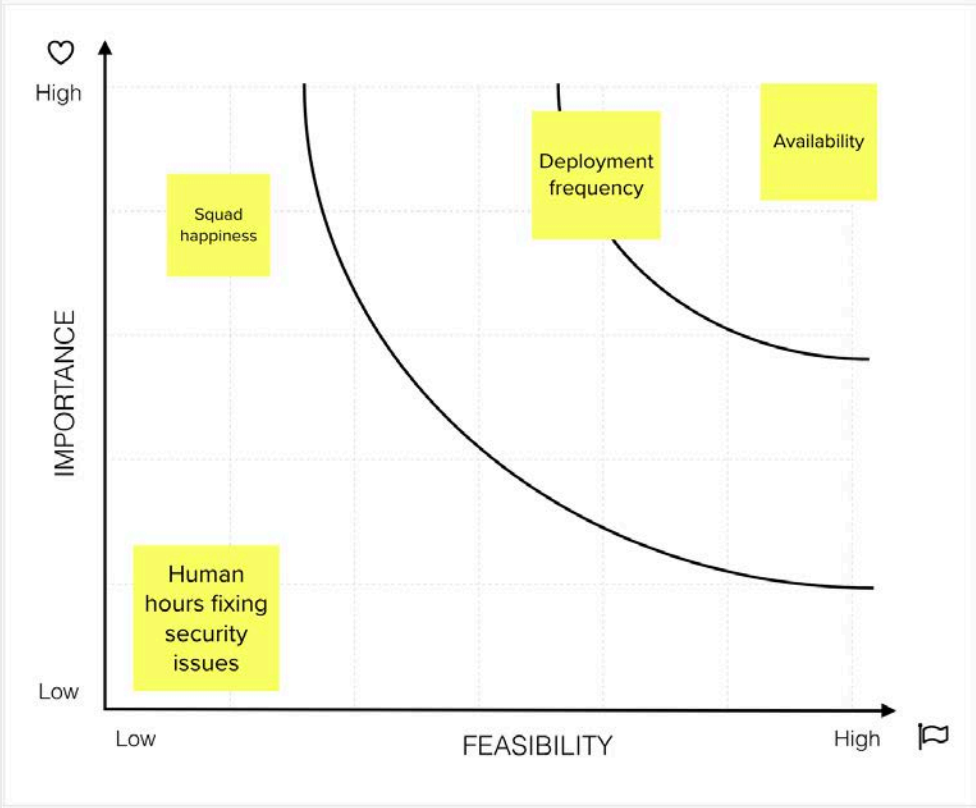
What we avoided and why

Friction as a motivator

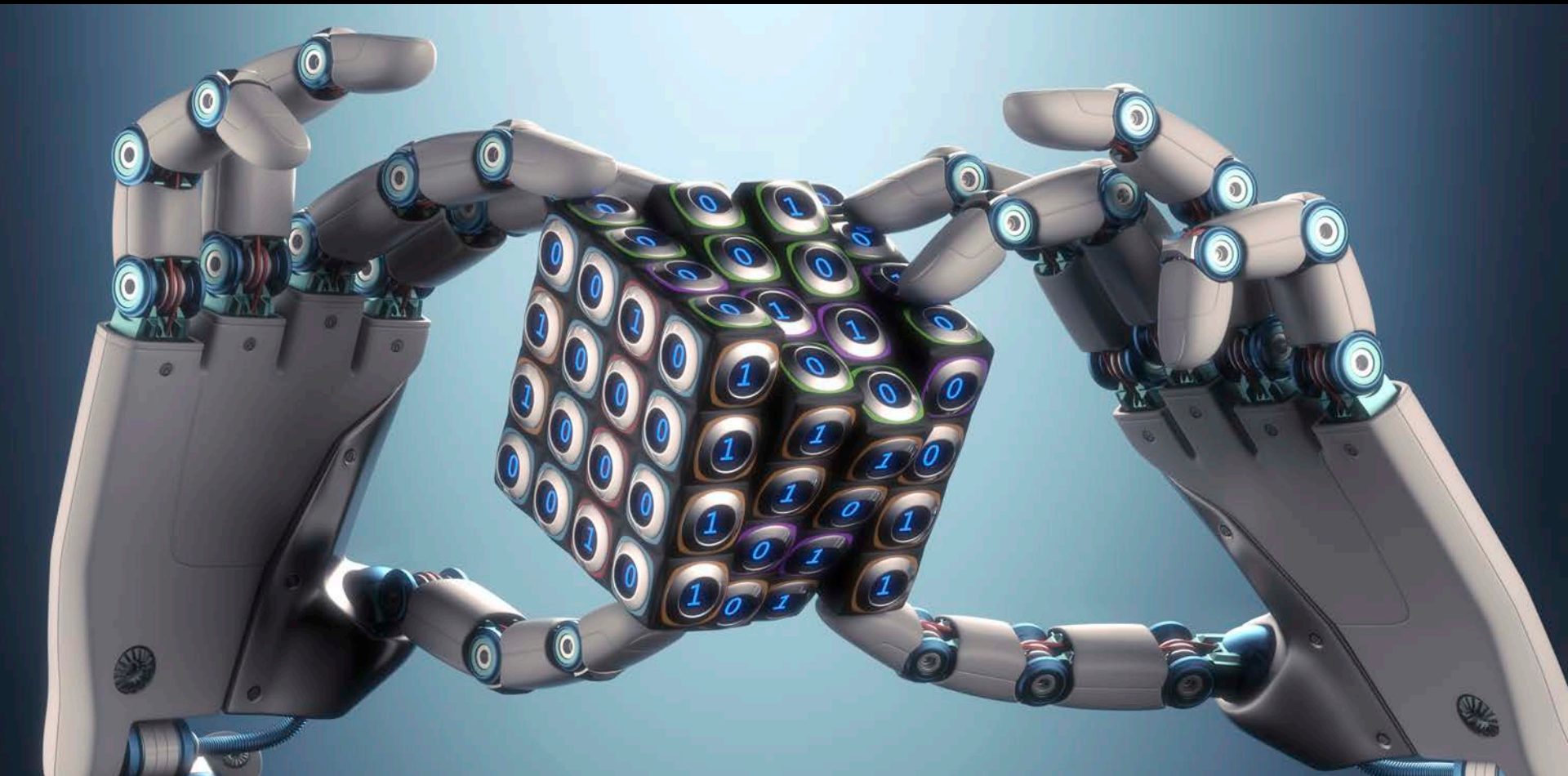
- Story points
- Metrics that are not automated
- Lead Time
- Defects counts
- Defects out of SLA
- Things that would have required all squads to adopt a certain workflow

Ranking system

- Where should we start?



II. Metrics in Detail



Score Details

Overall Score

The overall score is a weighted average of the other scores.
If data is not available for a score, it is omitted from the calculation of the overall score.

Release score
= average(DeployFrequency 0.70, DeployStability 1.00, DeploySpeed 1.00, RepoSpeed 1.00, RepoEfficiency 1.00)
= 0.94
Code score
= average(Security 1.00, CodeQuality 0.75, TestCoverage 0.12)
= 0.62
Overall score
= average(Availability 1.00, Release 0.94, Code 0.62)
= 0.85

Availability Score

Deployment Score

Repository Speed Score

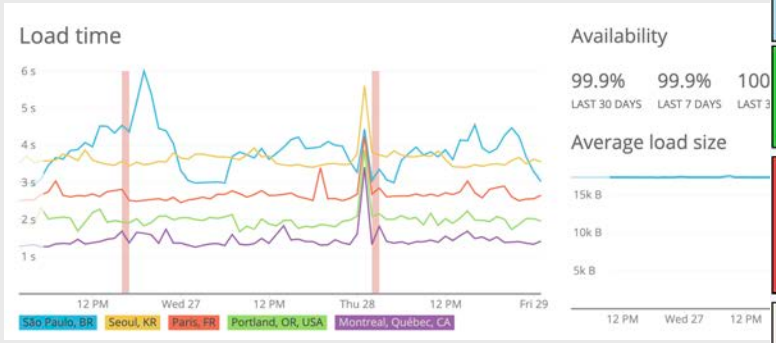
Repository Efficiency Score

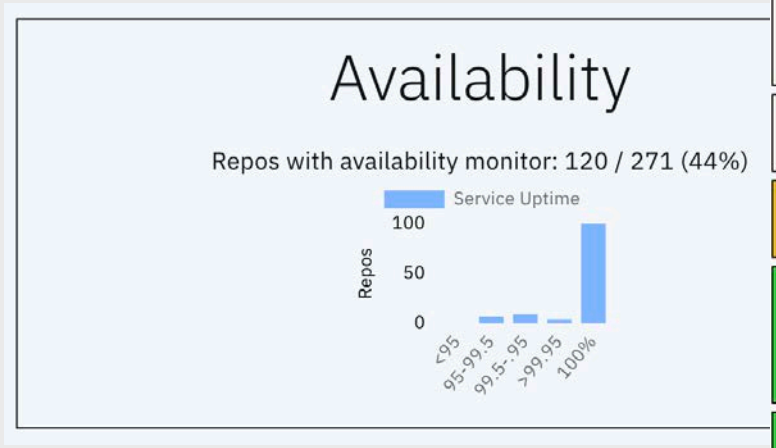
Security, Code Quality, and Code Coverage

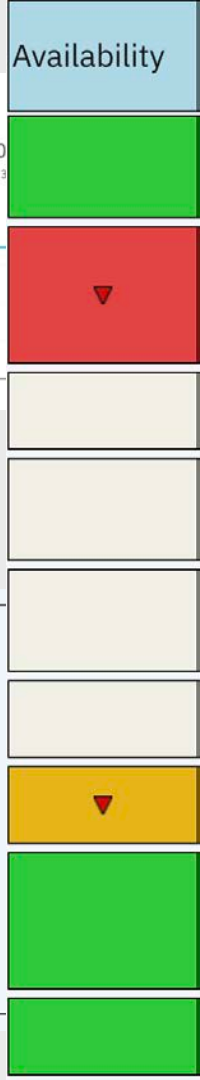
Calculating Scores:
Overall Approach

Availability vs. SLO Attainment

- The availability score is based on the uptime of the deployed application, relative to its Service Level Objective, over the last 30 days. A score of 100 is given for 100% uptime, decreasing to 80 for meeting the SLO, and zero at four times below the SLO.
- Goal: The site can never be down.







Deployment Frequency

- Deployment frequency is based on the number of successful deployments, over the last 30 days. 0-1 deployments: red, 2-3: yellow, 4+: green.
- Goals: Make small, frequent, low-risk changes. Use continuous delivery. Fully automate tests. Never change servers running in production; change code in Github and re-deploy. Patch services frequently.

Deployment Stability

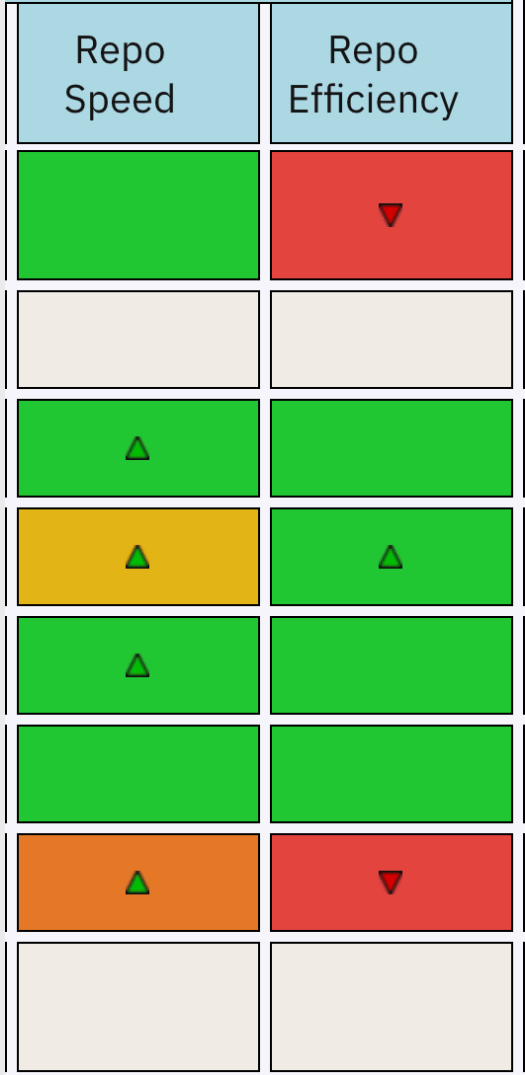
- Deployment stability is the percentage of time when the most recent build was successful. 0-50%: red, 50-90: yellow, 90+: green.
- Goals: Shift left to find errors earlier, on developer workstations. Fix long-running build and deployment issues to make developers more productive.

Deployment Speed

- Deployment speed is based on the time to deploy changes to production. A perfect score is given for build times under 20 minutes, decreasing to zero above 90 minutes.
- Goals: Judicious use of build parallelization to speed up builds. Fast deployments improve the Mean Time to Recovery when re-deployments are needed to fix problems. Deploy more often and get faster feedback.

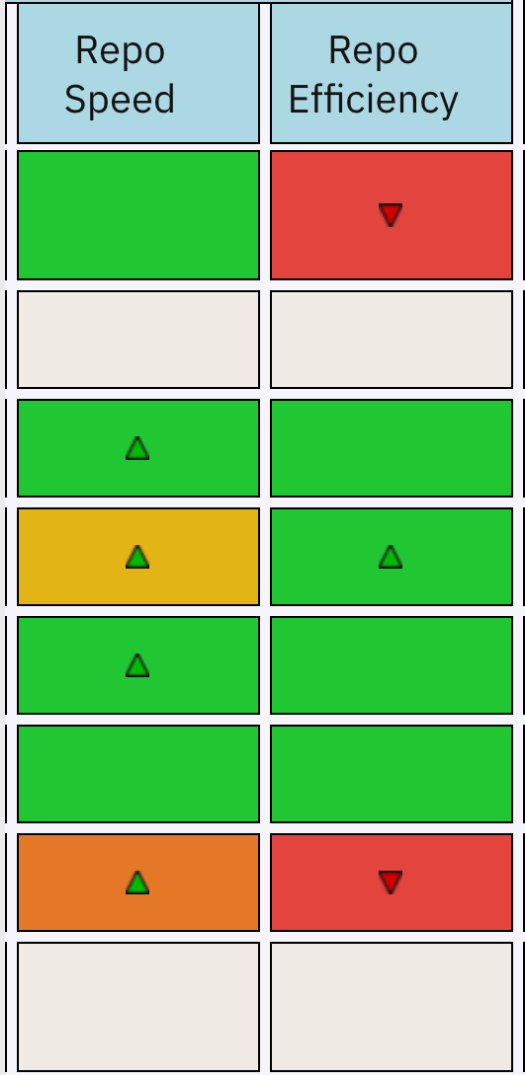
Repository Speed

- The repository speed score is based on the time from submission to merge (i.e. review duration) of GitHub pull requests, over the last 30 days.
  A perfect score is given when the average time per PR is 0-2 weekdays (M-F), decreasing to zero at 5 weekdays.
- Goals: Don’t neglect pull requests. Reduce wasted work. Reduce Work in Progress.



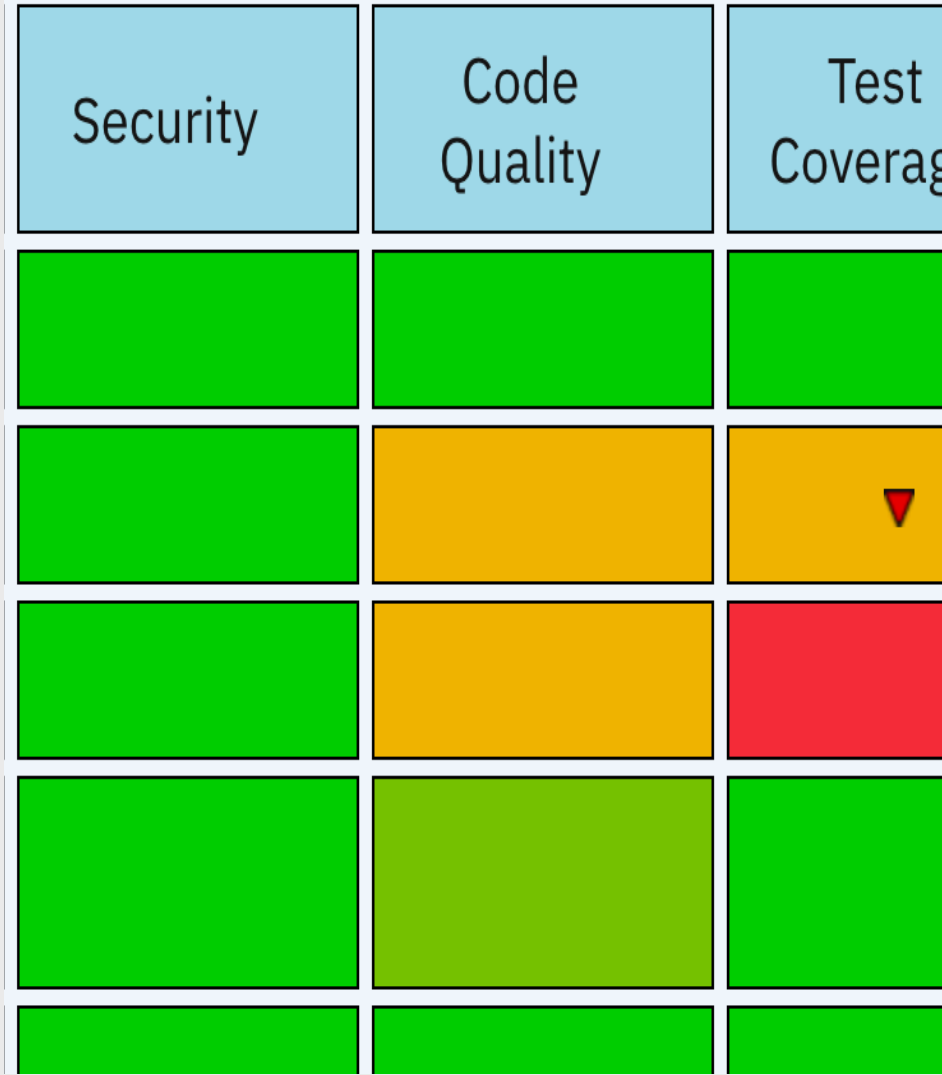
Repository Efficiency

- The repository efficiency score is based on added lines of code in GitHub pull requests, over the last 30 days.
  A perfect score is given when the median lines added per PR is under 150 lines, decreasing to zero at 500 lines.
  Deleted lines and changes to certain machine-generated files (e.g. package-lock.json) do not affect the score.
- Goals: Keep changes small and low-risk. Make code reviews easier and faster.



Metrics from SonarQube

- Security
- Avg(Reliability + Maintainability)
- Test coverage



Squad Level

- Release score = average (DeployFrequency 1.00, DeployStability 0.76, DeploySpeed 1.00, RepoSpeed 1.00, RepoEfficiency 1.00)
- Code score = average(Security 1.00, CodeQuality 1.00, TestCoverage 1.00)
- Overall score = average(Availability 1.00, Release 0.95, Code 1.00)
- Goals: Quick overall view for the squad

Epics and Squad Comments

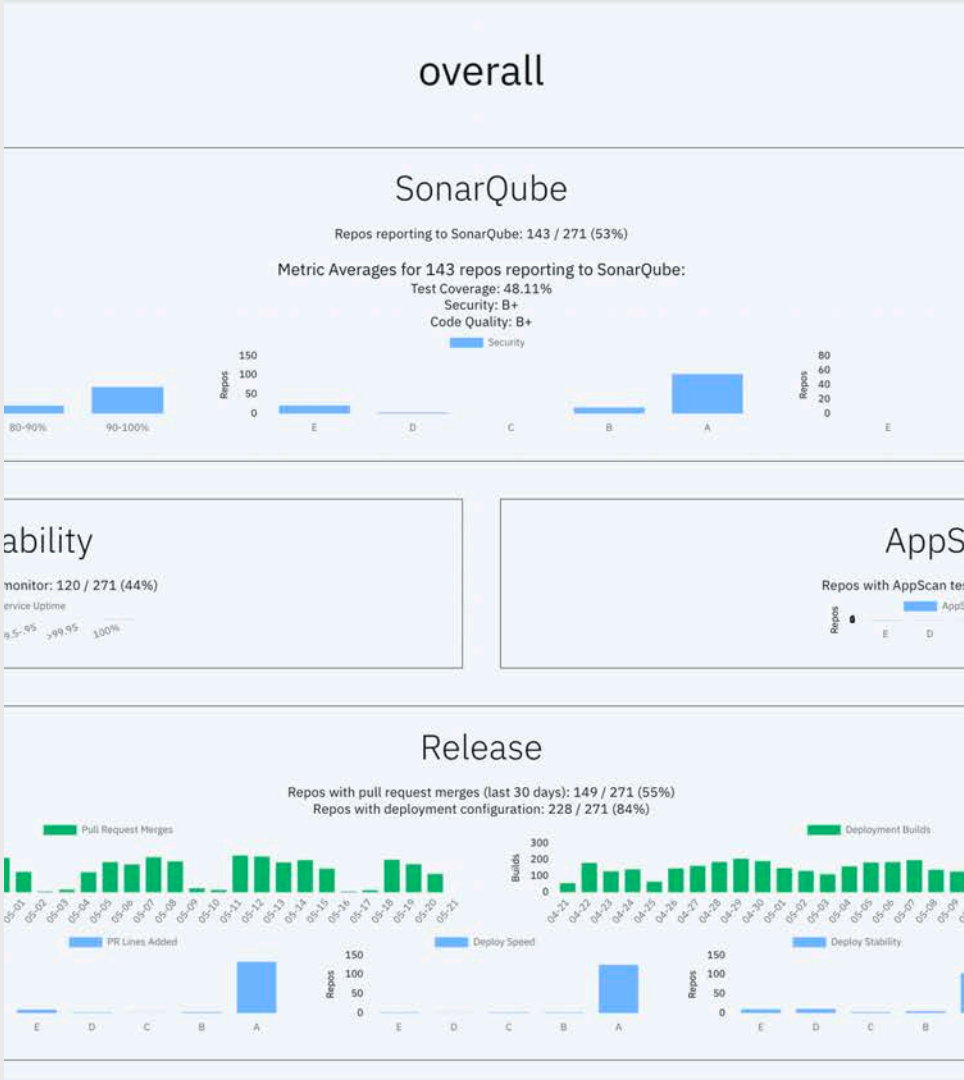
- How we use epics
- How we track them
- Squad comments

Agile metrics

- Work in Progress
- Work Completed
- Aging

Platform-level metrics

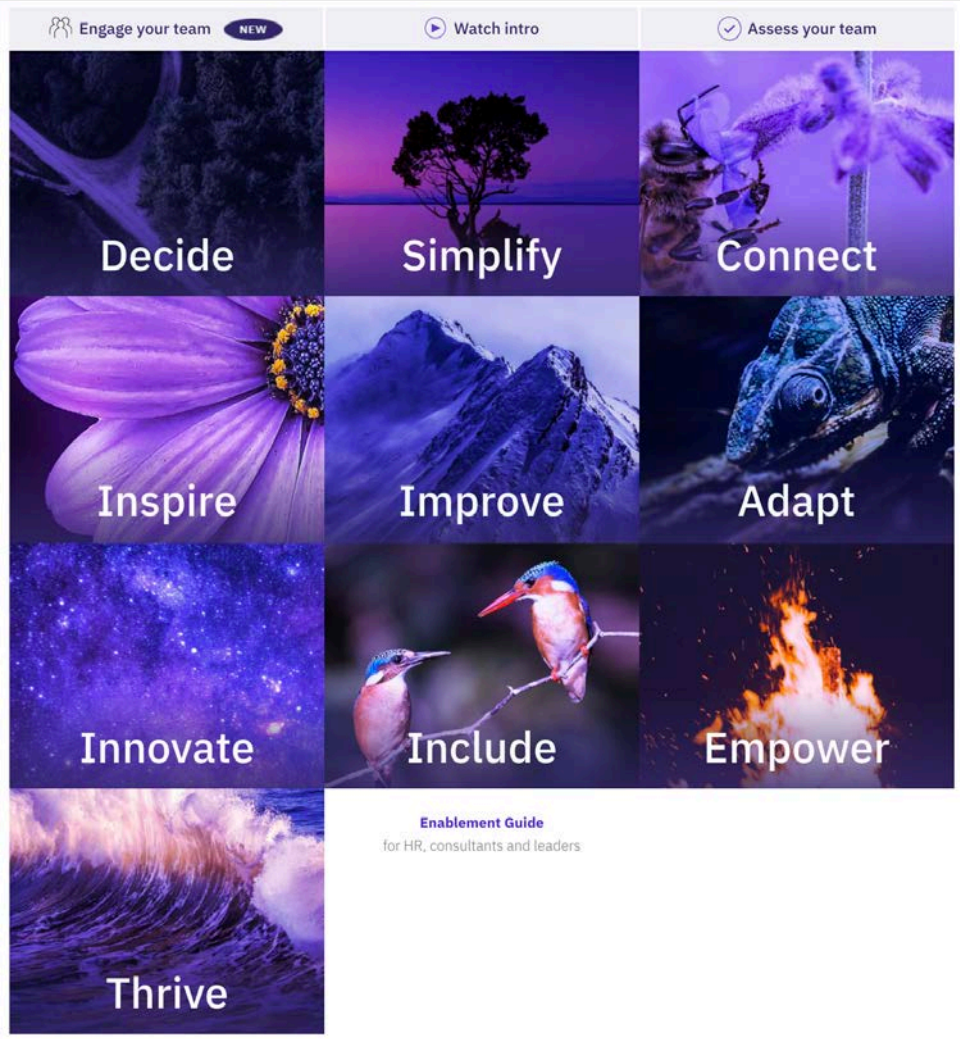
- General Manager and VP views
- Goal: Easy reviews



Metrics we wish we had

Important but harder to automate:

- Development Lead Time
- Flow Efficiency from Domenica DeGrandis: ((work)/(wait+work))*100%
- Squad Health Metrics from Henrik Kniberg at Spotify (source: https://labs.spotify.com/2014/09/16/squad-health-check-model/)
- Employee Engagement Survey



Metrics in Detail

Colors vs. Letter Grades vs. Numbers

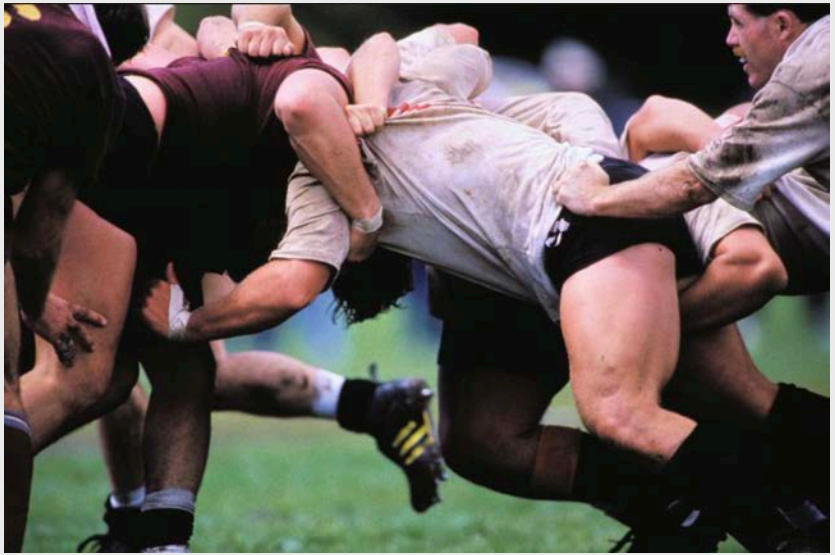
- A story of rejection and acceptance

Availability	Security	Release				
Availability	AppScan	Deploy Frequency	Deploy Stability	Deploy Speed	Repo Speed	Repo Efficiency
A	N/A	A	A	A	A	A
A	N/A	A	D ▼	A	A	A
E ▼	A	A	A	A	A	E ▼
A	N/A	A	A	A	A	A

Availability	Security	Release				
Availability	AppScan	Deploy Frequency	Deploy Stability	Deploy Speed	Repo Speed	Repo Efficiency
1.00		1.00	1.00	1.00	1.00	1.00
1.00		1.00	0.63 ▼	1.00	1.00	1.00
0 ▼	0.94	1.00	1.00	1.00	1.00	0.47 ▼
1.00		1.00	1.00	1.00	1.00	1.00

Don’t compare teams

- Every team is different



Feedback is a gift

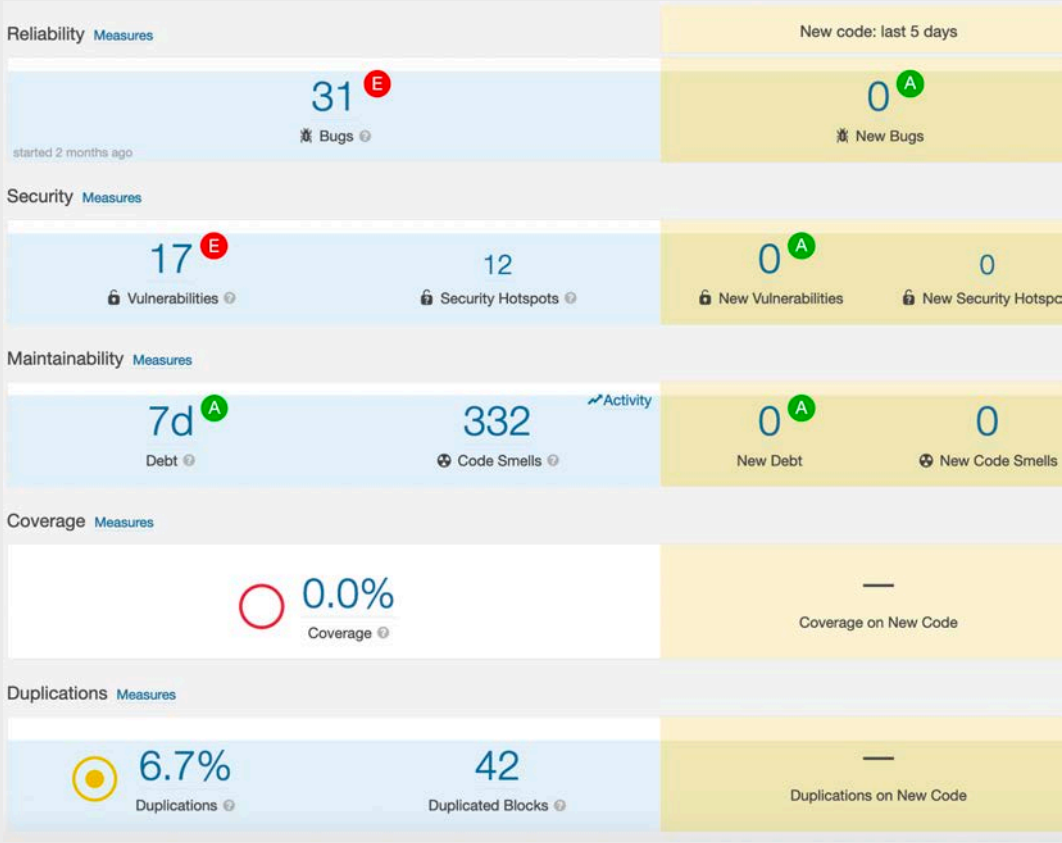
IV. Outcomes

Changes over time

Outcomes

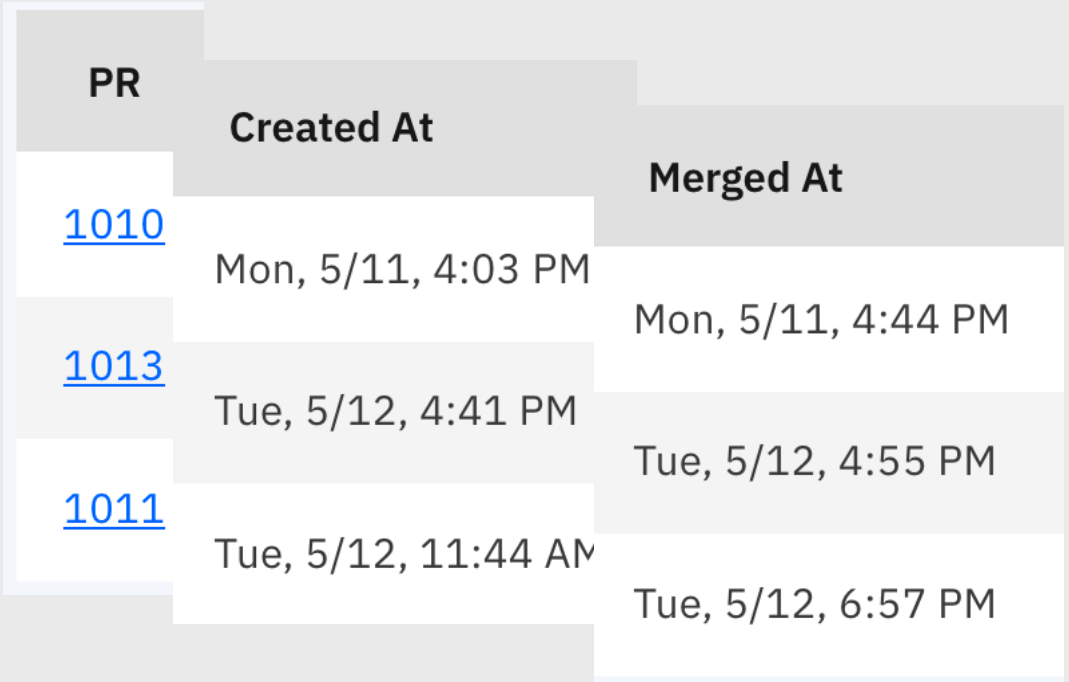
Visible SonarQube Metrics

- Poor results
  - Motivation to improve



Repository Speed

- Time from PR creation to merge
- Mob Programming



PR	Created At	Merged At
1010	Mon, 5/11, 4:03 PM	Mon, 5/11, 4:44 PM
1013	Tue, 5/12, 4:41 PM	Tue, 5/12, 4:55 PM
1011	Tue, 5/12, 11:44 AM	Tue, 5/12, 6:57 PM

Deployment frequency

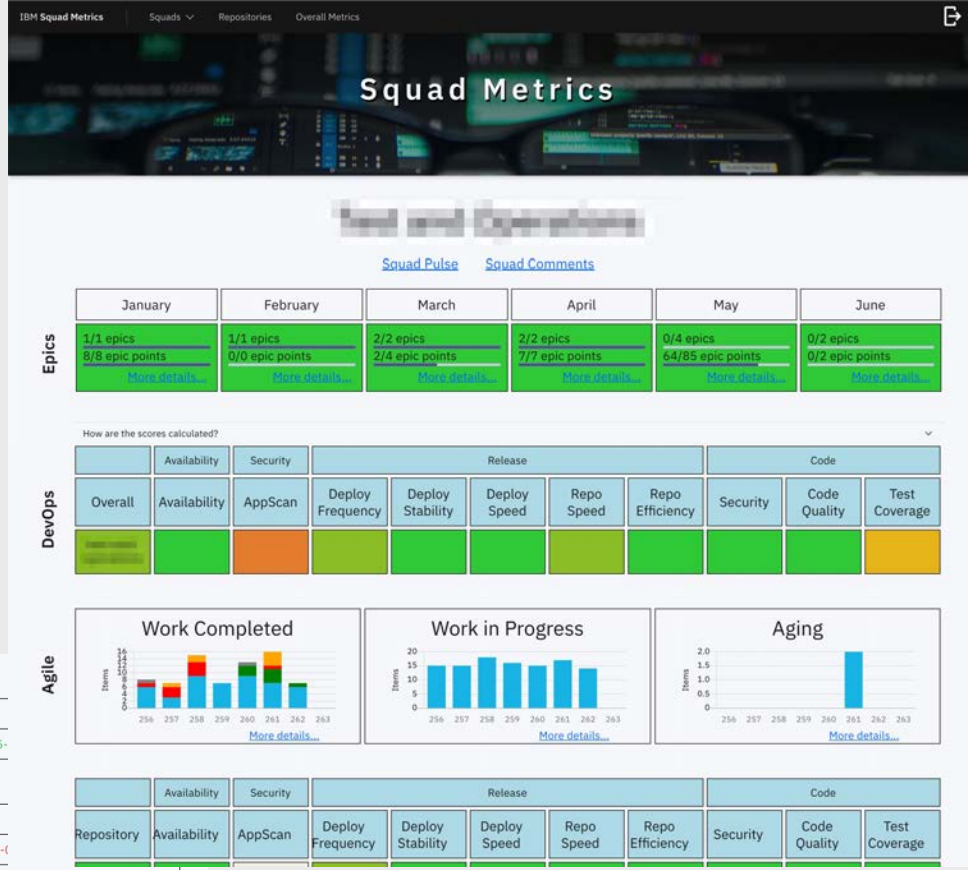
- Daily builds
- npm audit fixer

Build	Start Time
...rics/builds/31597178	Tue, 5/19, 3:09 PM
...rics/builds/31583351	Tue, 5/19, 12:09 PM
...rics/builds/31523739	Mon, 5/18, 4:29 PM
...rics/builds/31520564	Mon, 5/18, 3:38 PM
...rics/builds/31507960	Mon, 5/18, 12:39 PM

Outcomes

Our Squad

- Work in Progress
- Deploy Stability
- Deploy Frequency
- PR List

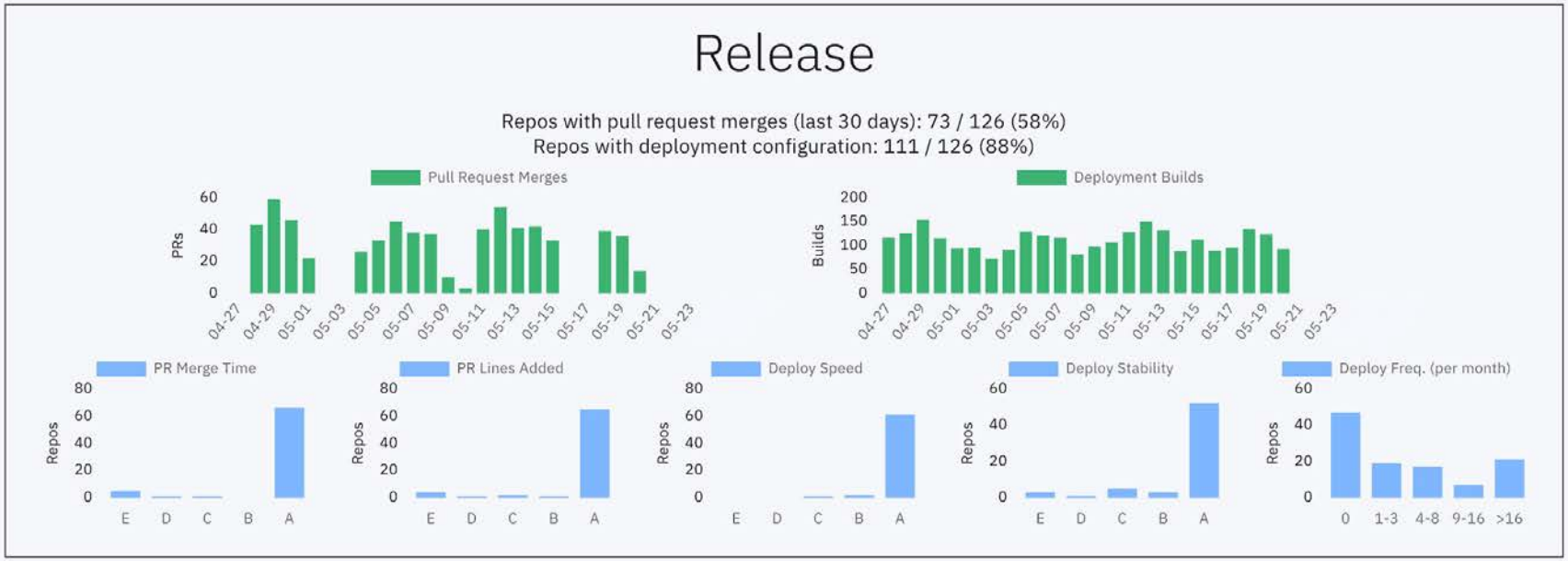


Repo Status

Repository	Last Build	AppScan Issues	Pull Requests
	Passed 05-28		PROD RELEASE Created: 05-28
	Passed 05-28		Feat/notification schema validation Created: 05-
	Passed 05-28	2 Low 05-20	
	Passed 05-29		
	Passed 05-27		Feat/configurable order processor Created: 05-(
	Passed 05-12	No Issues 05-20	
	Passed 05-28		
	Passed 10-09		

In summary:

1. Why should you care about DevOps Metrics?
2. How can you incentivize the right behaviors?

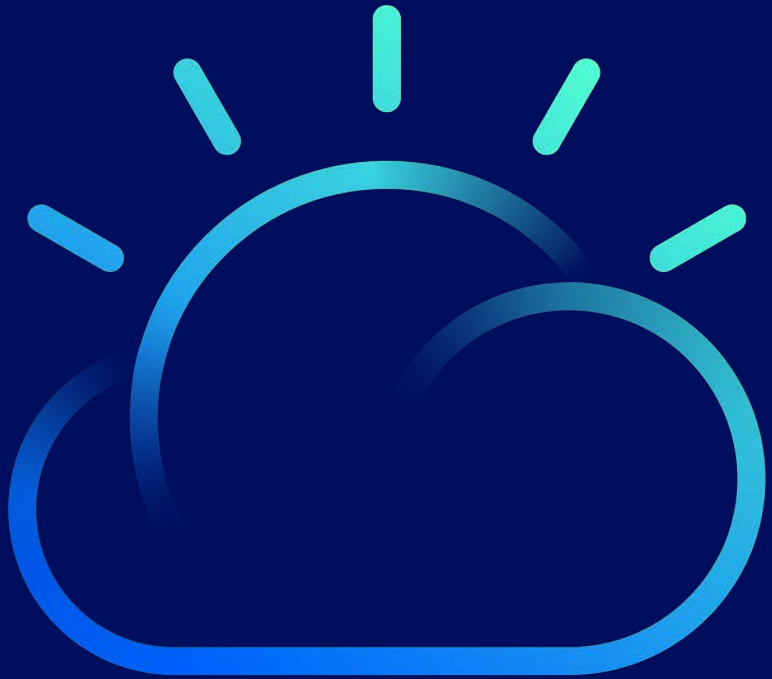


Thank you!

—

Ann Marie Fred
DevOps and Security Lead
Twitter: @DukeAMO

Craig Cook
Technical Lead for Toolbox@IBM
Operational Toolling
Twitter: @CraigCookIT



IBM Cloud