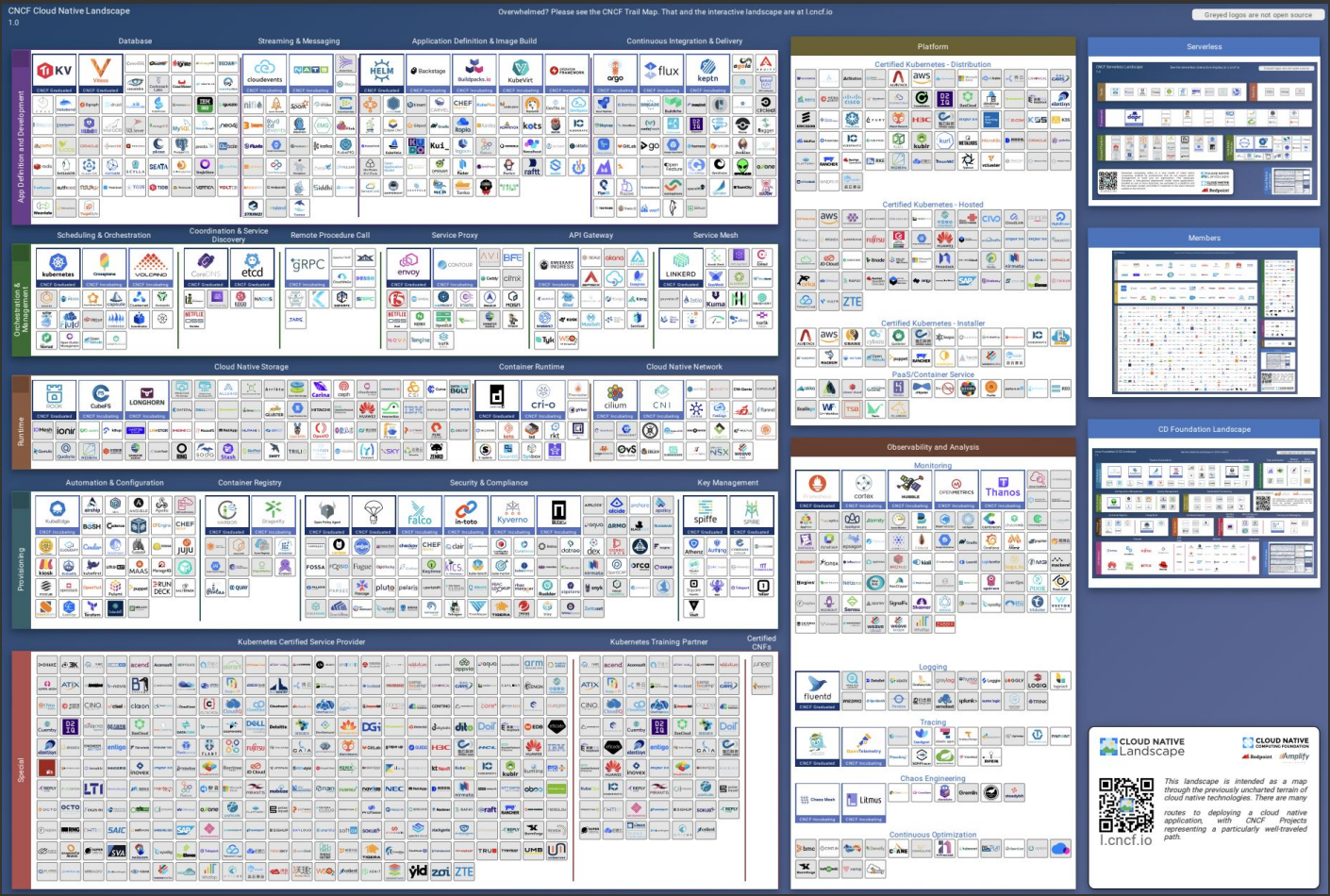
Better DevOps practices for for high performing software teams

@IAmJerdog

@IAmJerdog


CNCF Cloud Native Landscape
1.0
Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io
Greyed logos are not open source
App Definition and Development
Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
Orchestration & Management
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Runtime
Cloud Native Storage
Container Runtime
Cloud Native Network
Provisioning
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Special
Kubernetes Certified Service Provider
Kubernetes Training Partner
Certified CNFs
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Logging
Tracing
Chaos Engineering
Continuous Optimization
Serverless
Members
CD Foundation Landscape
CLOUD NATIVE Landscape
CLOUD NATIVE COMPUTING FOUNDATION
Redpoint
Amplify
This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.
l.cncf.io

@IAmJerdog

Jeremy Meiss

Director, DevRel & Community

circleci

@IAmJerdog


2019
State of DevOps Report
puppet
circleci
splunk>
2017
State of DevOps Report
Presented by:
puppet + DORA
DEVOPS RESEARCH & ASSESSMENT
Sponsored by:
Hewlett Packard Enterprise
splunk>
amazon web services
Atlassian
WAVEFRONT by vmware
Electric Cloud
Deloitte.
2018
State of DevOps Report
puppet · splunk>
aws
CYBERARK
Cognizant
2016
State of DevOps Report
2015
STATE
REPORT

@IAmJerdog


John Allspaw
@allspaw
Normal 0%
Work-as-imagined versus work-as-done
Variability and graceful extensibility
YOUR "PLANS"
THE UNIVERSE'S PLANS FOR YOU
DOGHOUSEDIARIES
FROM INNOVATION TO SOLUTION
24
3:00 AM · Apr 28, 2016 · Twitter for iPhone

@IAmJerdog

performance described

vs

performance derived

@IAmJerdog

Dataset

257 mil+
workflows

44,000+
orgs

290,000+
projects

1,000x
Larger than surveys

Image: *Risk Culture*

Four classic metrics

Deployment frequency

Lead time to change

Change failure rate

Recovery from failure time

CI/CD Benchmarks for high performance teams

	Suggested Benchmarks
 **Throughput** The average number of workflow runs per day	Merge on any pull request
 **Duration** The average length of time for a workflow to run	10 minutes
 **Mean time to recovery** The average time between failures & their next success	Under 1 hour
 **Success rate** The number of successful runs / the total number of runs over a period of time	90% or better on default branch

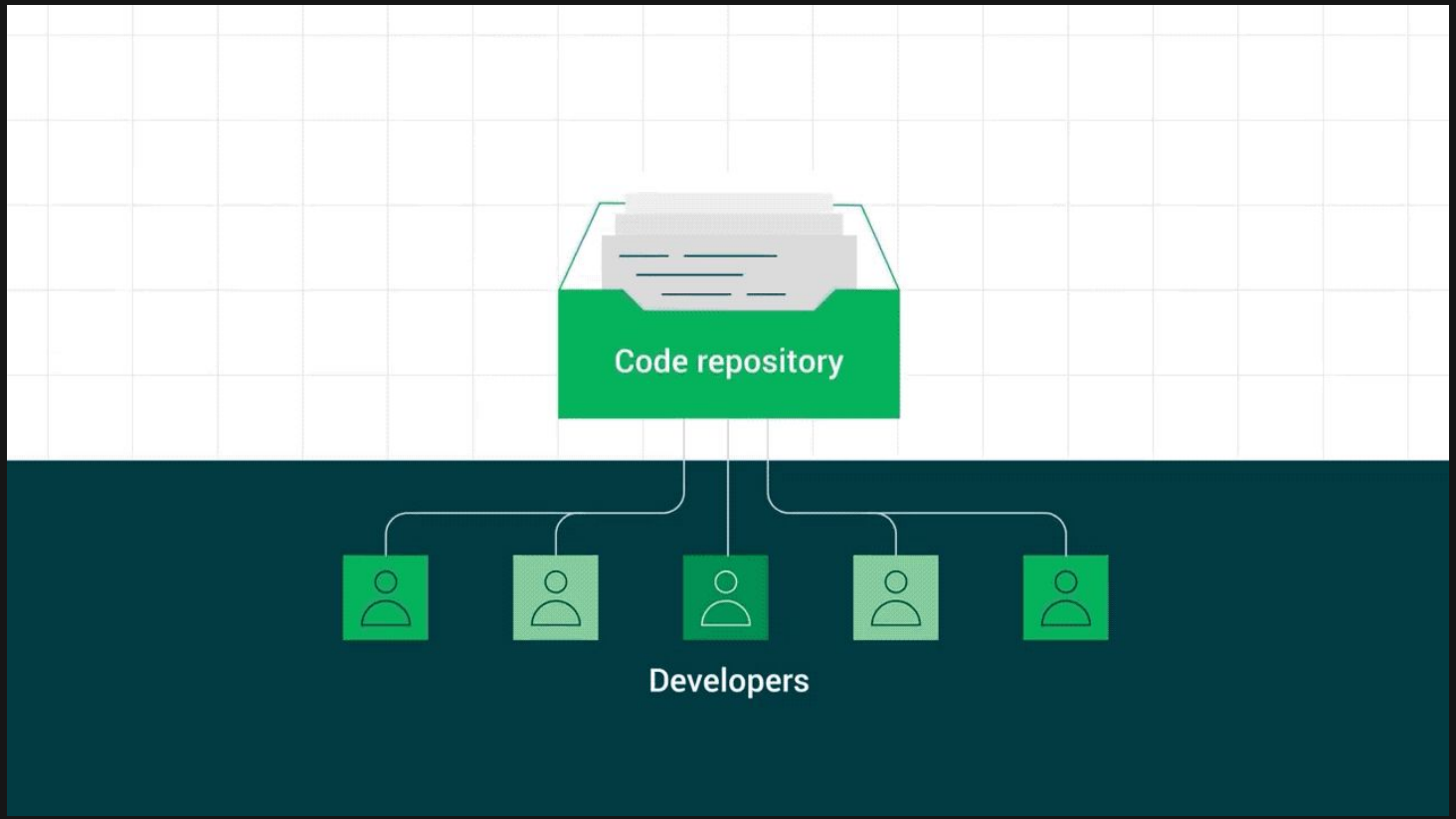
@IAmJerdog

The Data

Photo by: *Matthew Henry*

Throughput

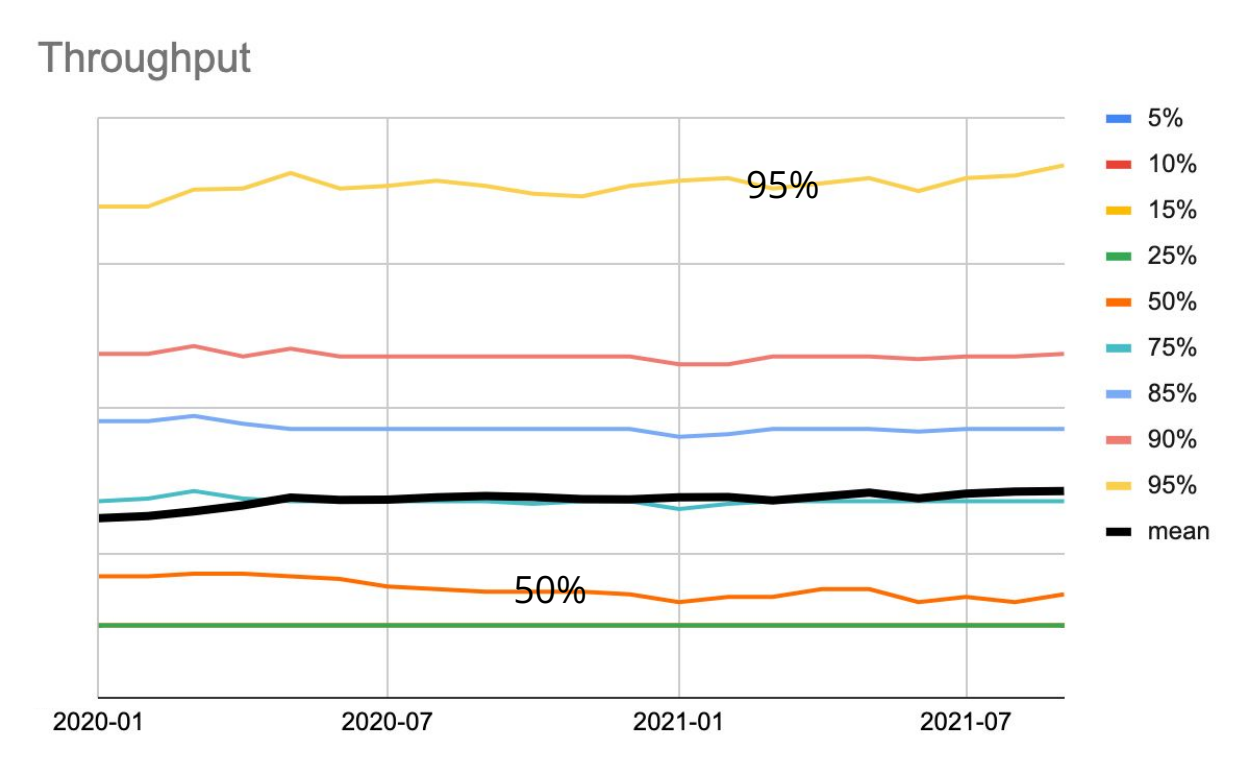
the average number of workflow runs per day


Code repository
Developers

Throughput

TIP: make smaller commits more often

Throughput



@IAmJerdog

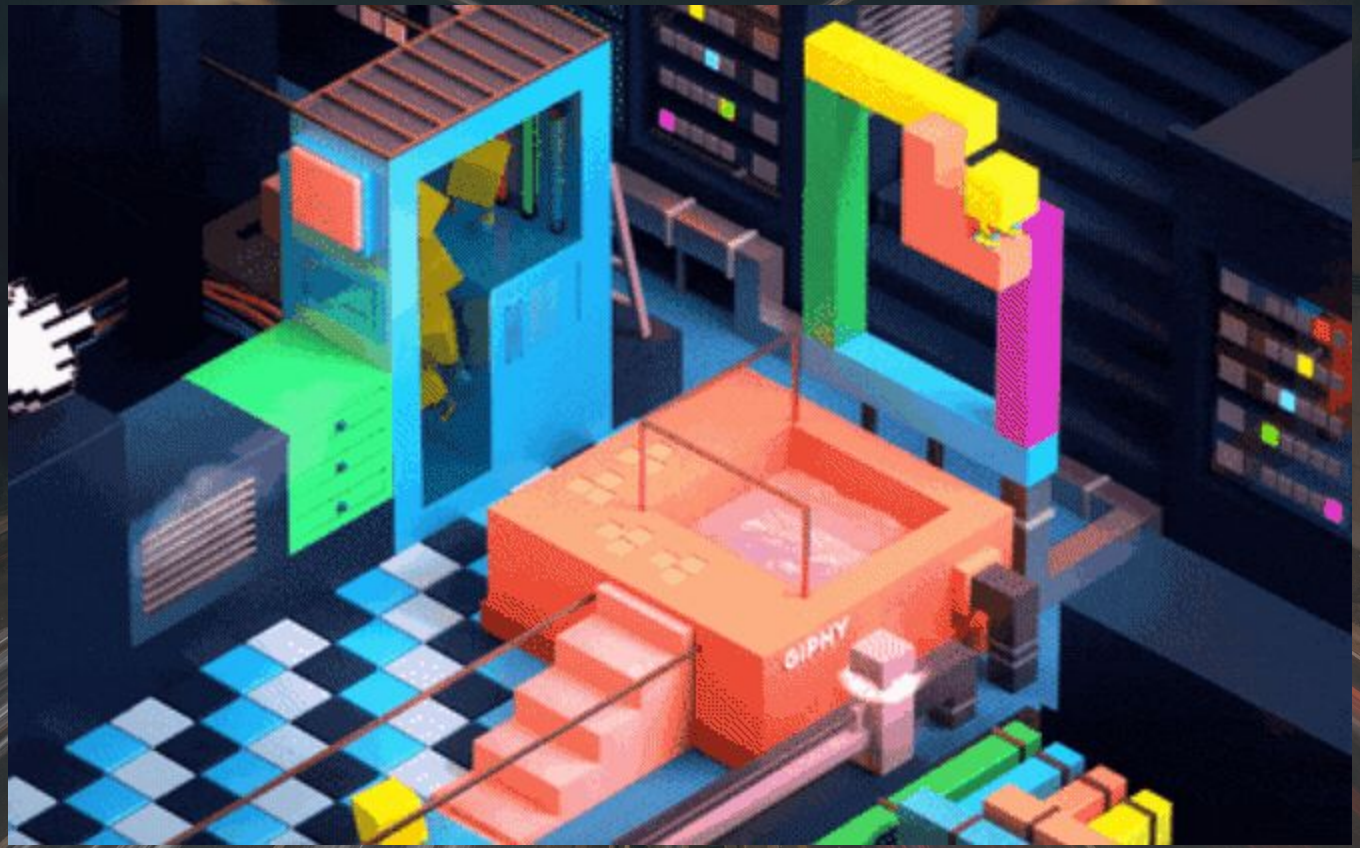
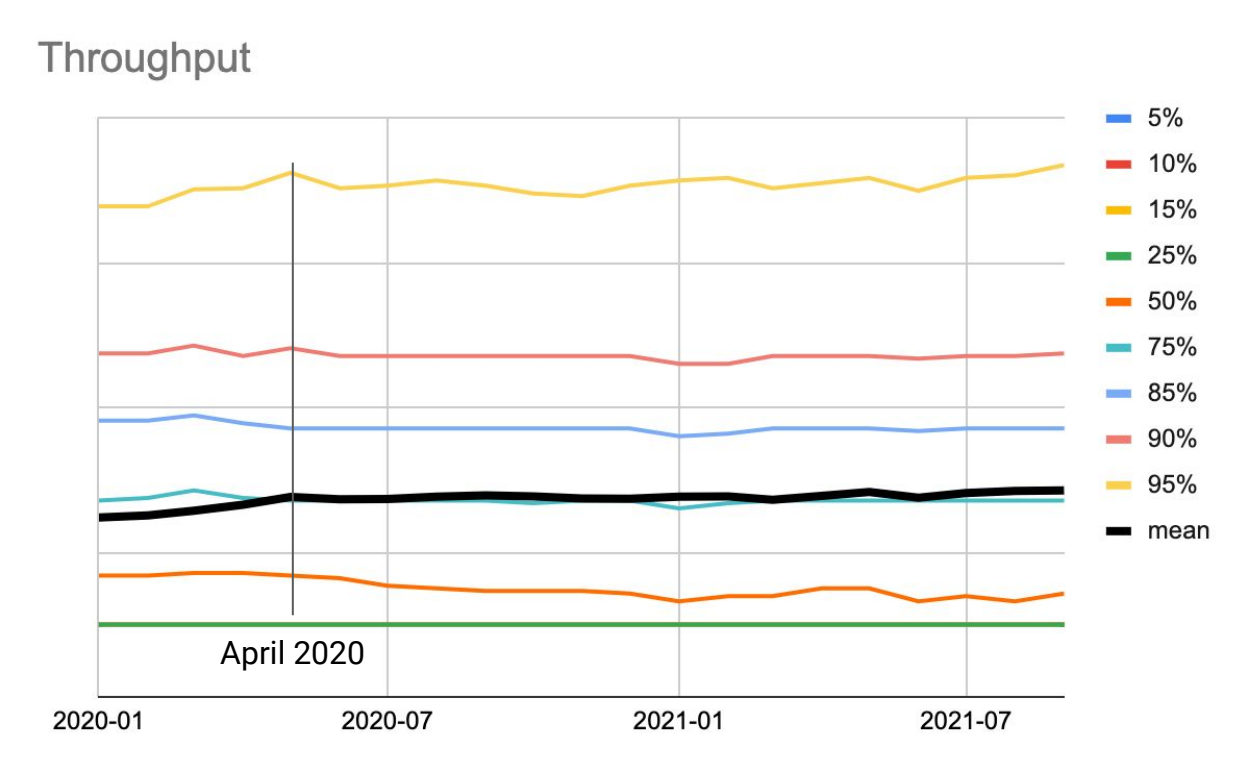


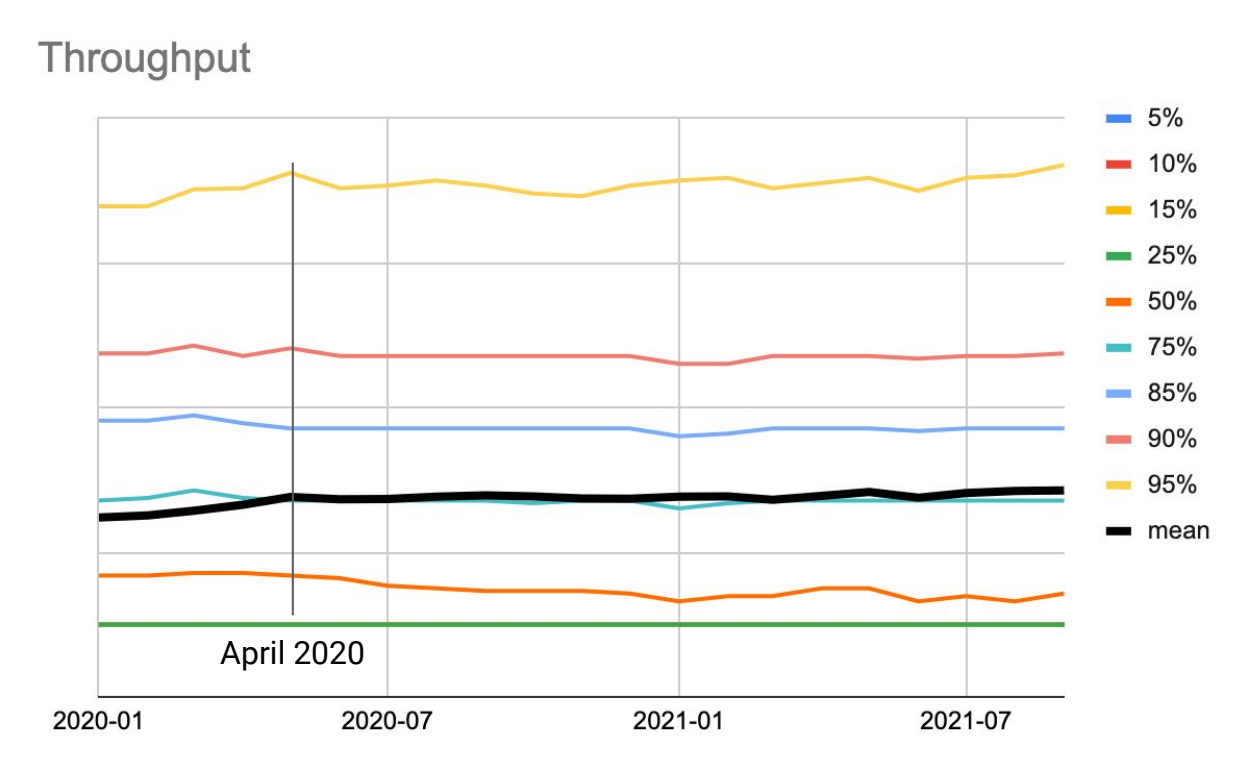
Image credit: Giphy

@IAmJerdog

Throughput


Throughput
5%
10%
15%
25%
50%
75%
85%
90%
95%
mean
April 2020
2020-01
2020-07
2021-01
2021-07

Throughput


Throughput
5%
10%
15%
25%
50%
75%
85%
90%
95%
mean
April 2020
2020-01
2020-07
2021-01
2021-07

Most teams are not deploying dozens of times per day

High-performing Teams & Throughput

- More valuable for orgs to see changes / progress week-over-week
- Prioritize lean, Agile software development patterns with small, incremental changes

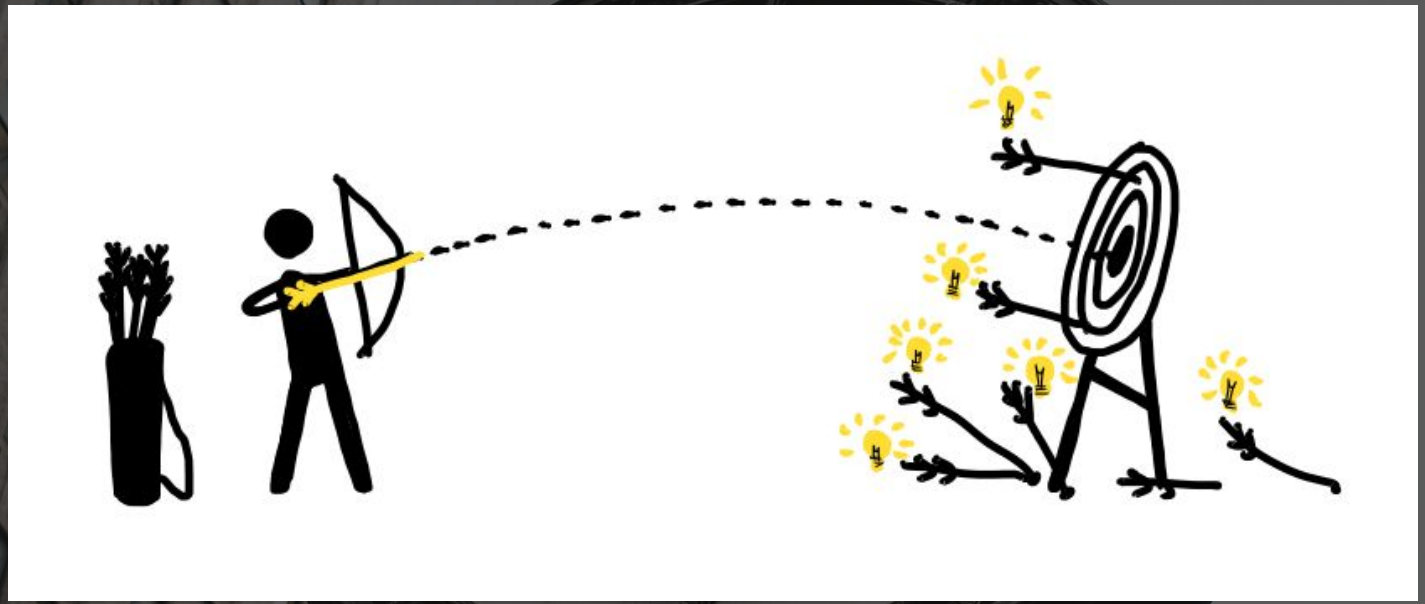
Image by Pawan Kolhe from Pixabay

Image by Pawan Kolhe from Pixabay

Duration

the length of time it takes for a workflow to run

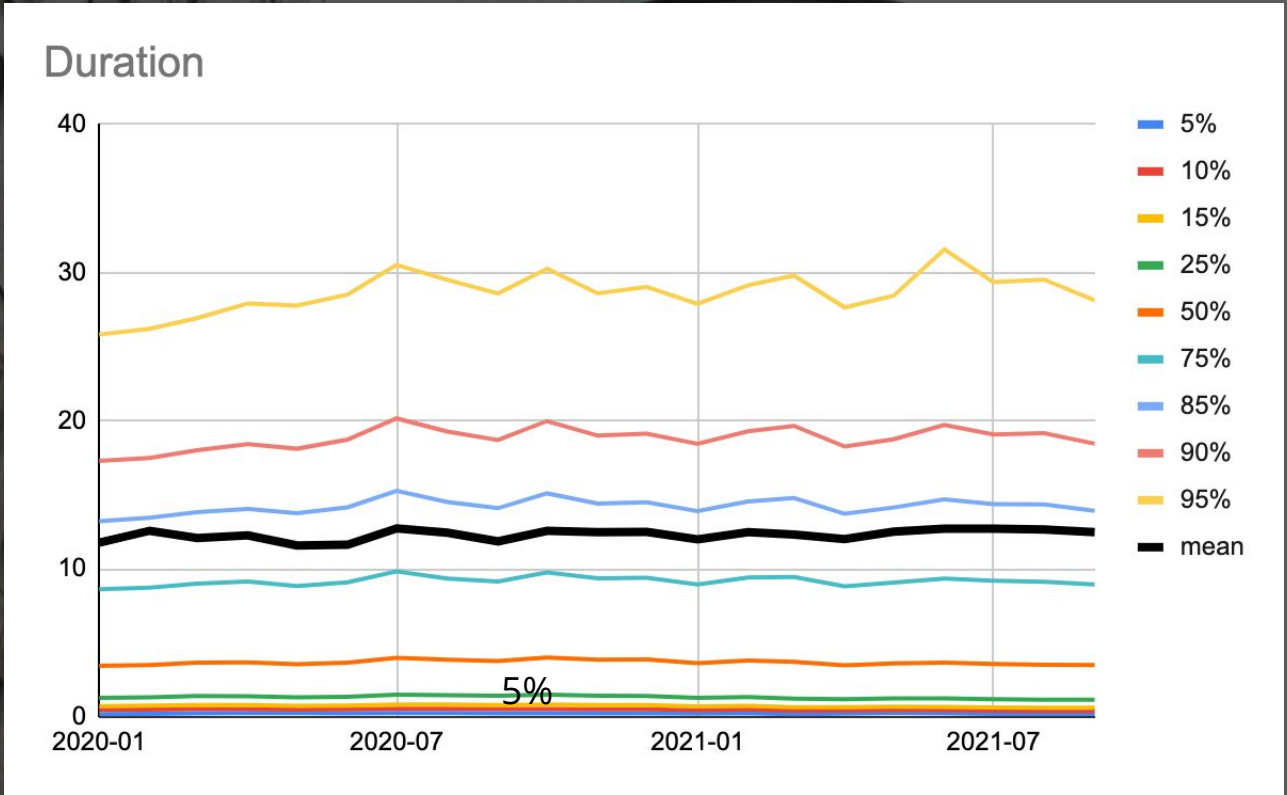
Image by Pawan Kolhe from Pixabay



@IAmJerdog

Image by Pawan Kolhe from Pixabay

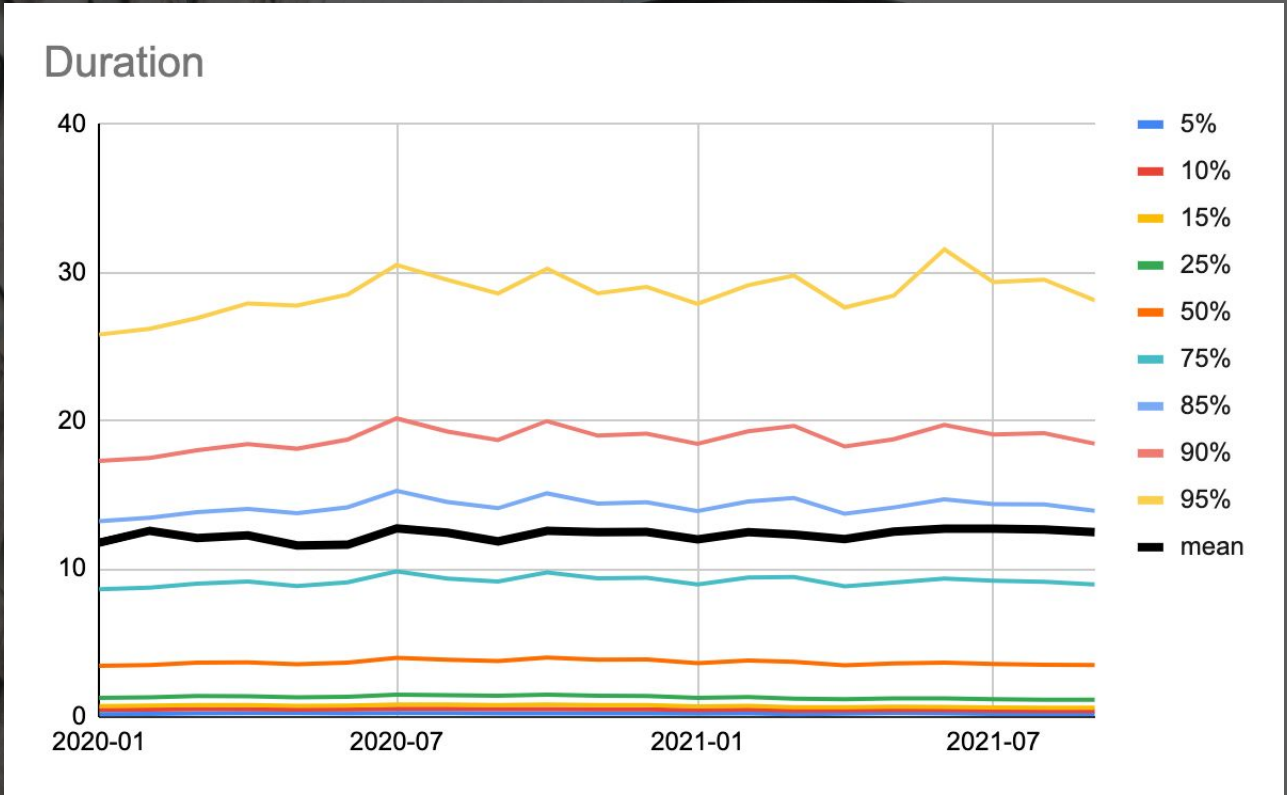
Duration



@IAmJerdog

Image by Pawan Kolhe from Pixabay

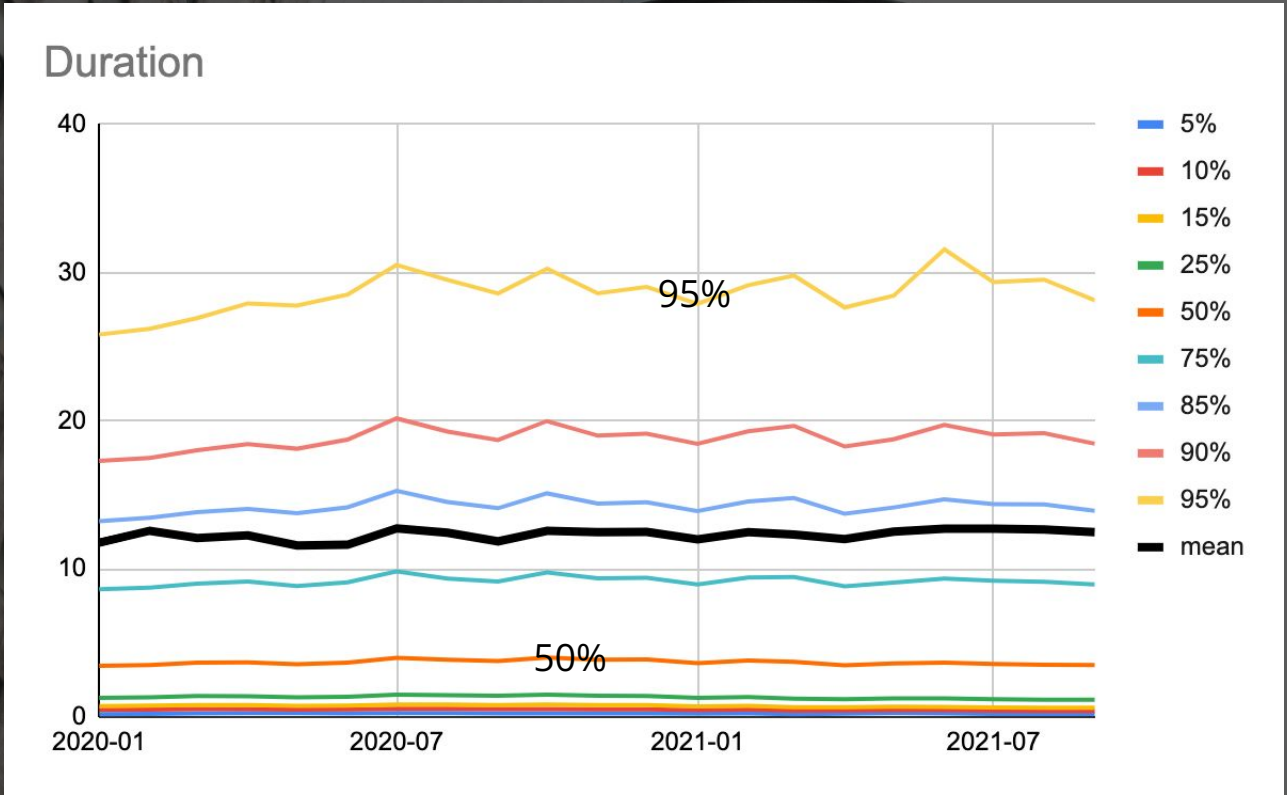
Duration



@IAmJerdog

Image by Pawan Kolhe from Pixabay

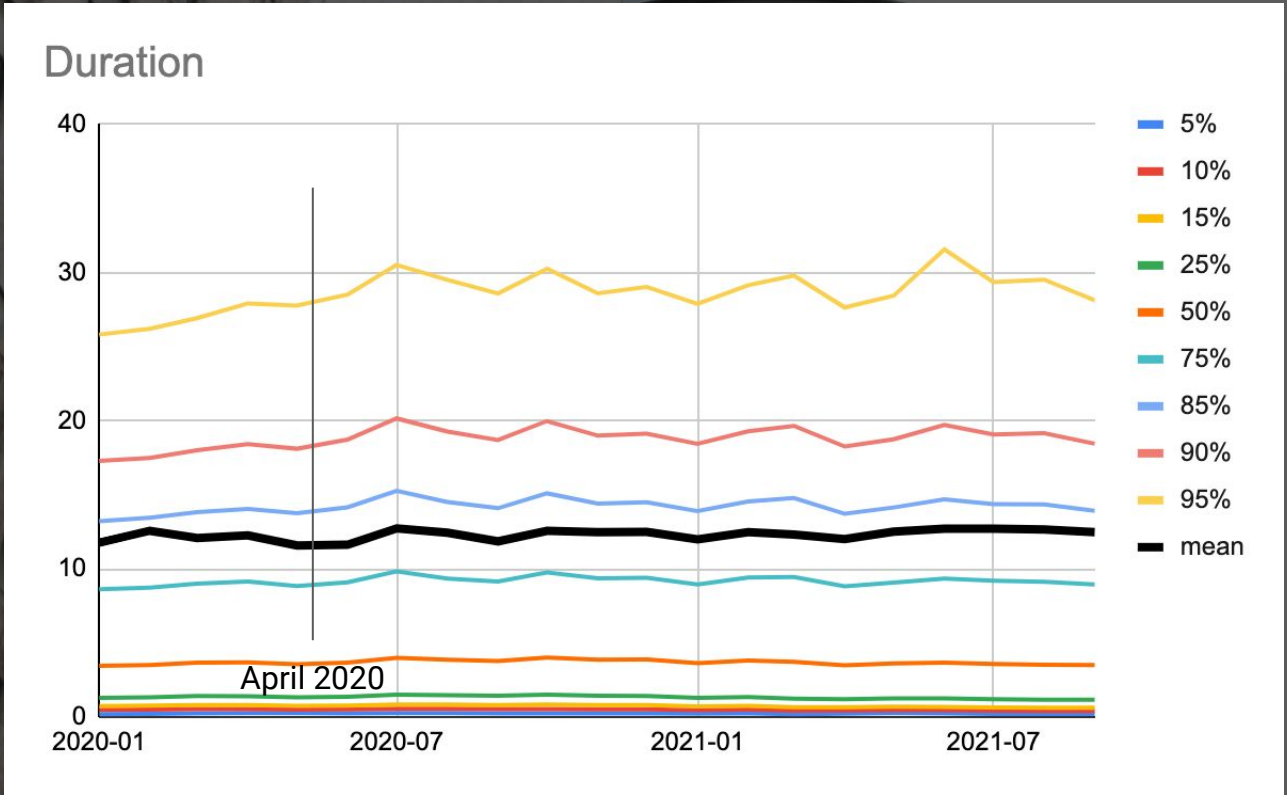
Duration



@IAmJerdog

Image by Pawan Kolhe from Pixabay

Duration



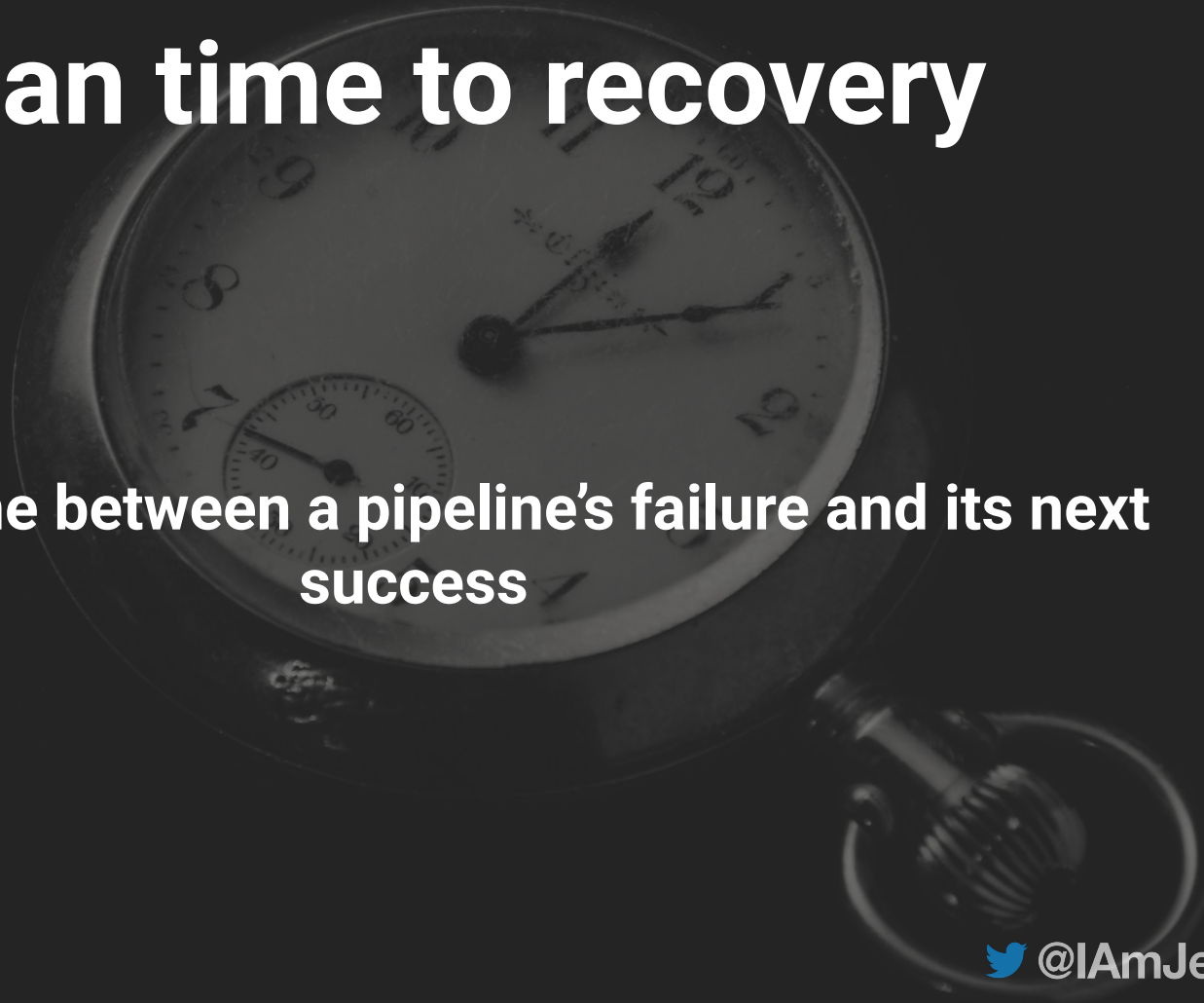
@IAmJerdog

High-performing Teams & Duration

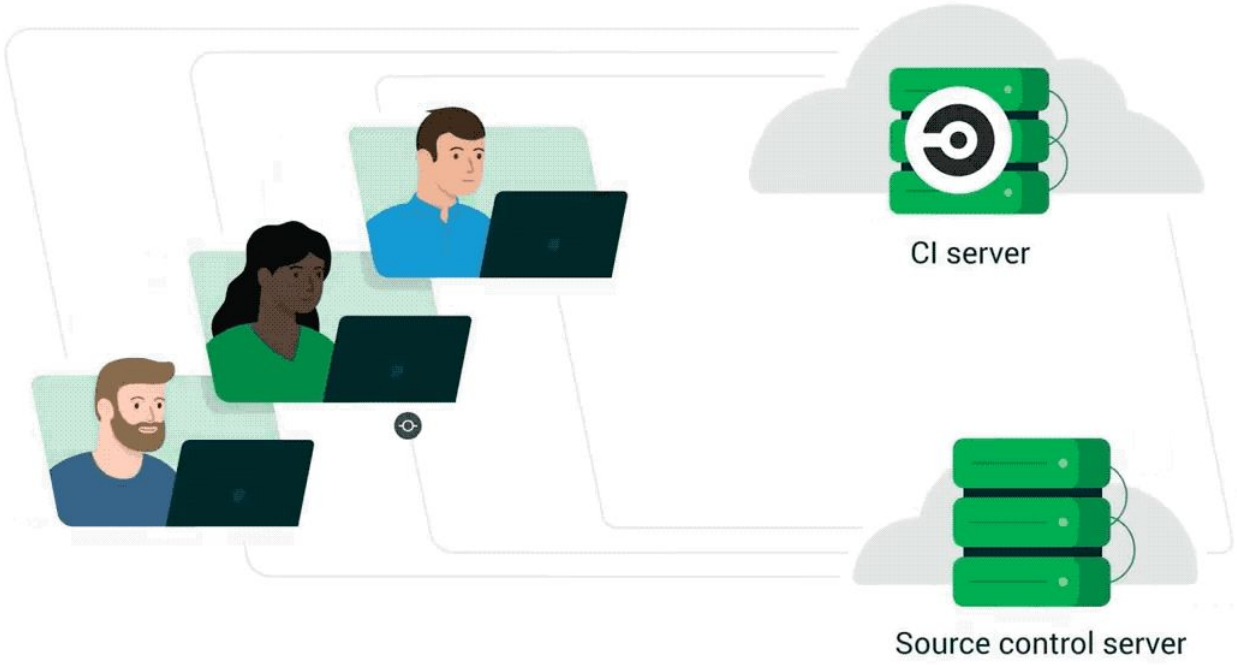
- Use test splitting to split tests
- Use Docker images specific for CI
- Use caching strategies to allow for reuse
- Use optimal size machine to run workflow

Photo by Brett Sayles from Pexels

Mean time to recovery



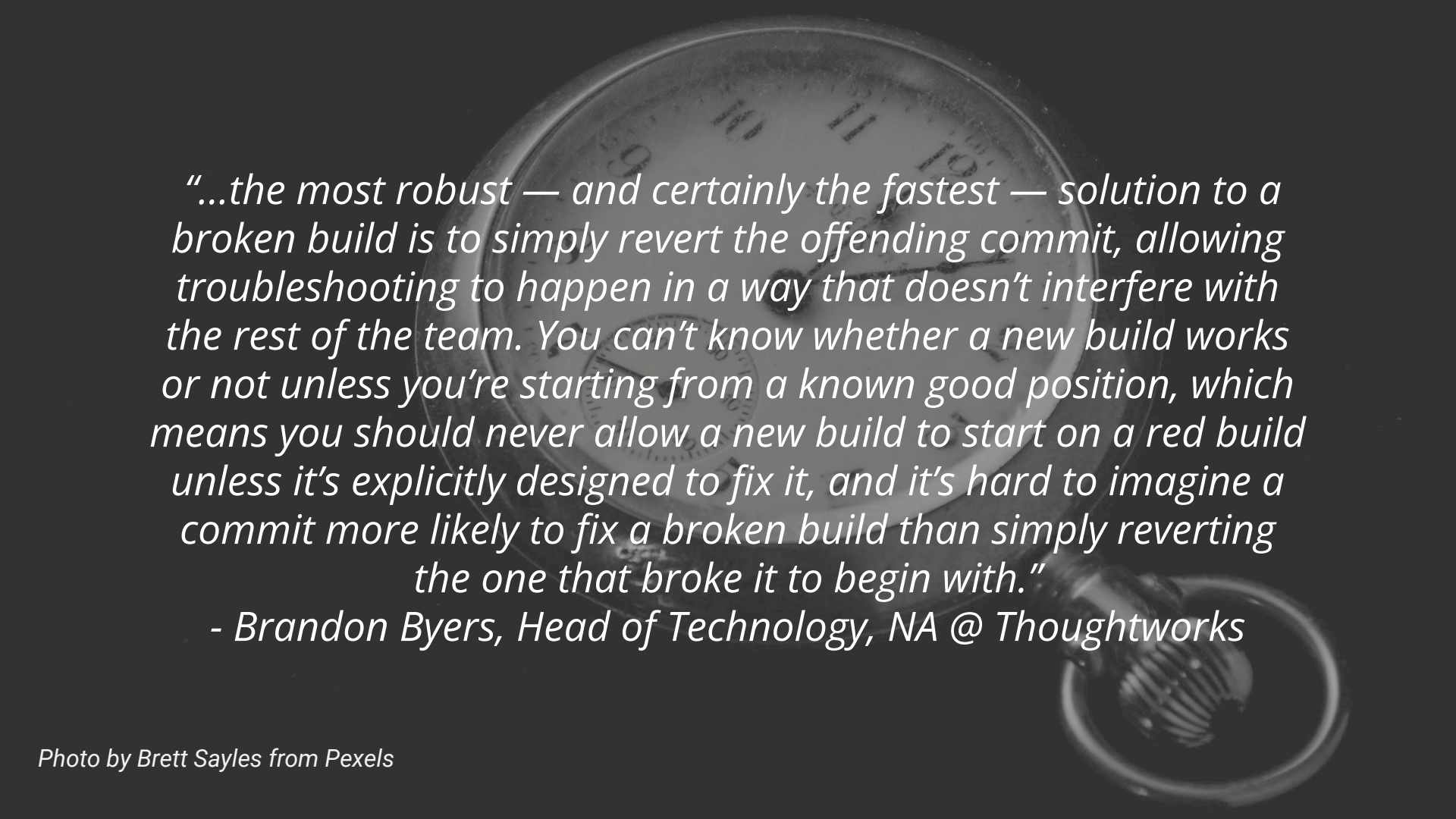
average time between a pipeline's failure and its next success


CI server
Source control server

Mean time to recovery

shortest MTTR $\propto$ Duration

@IAmJerdog

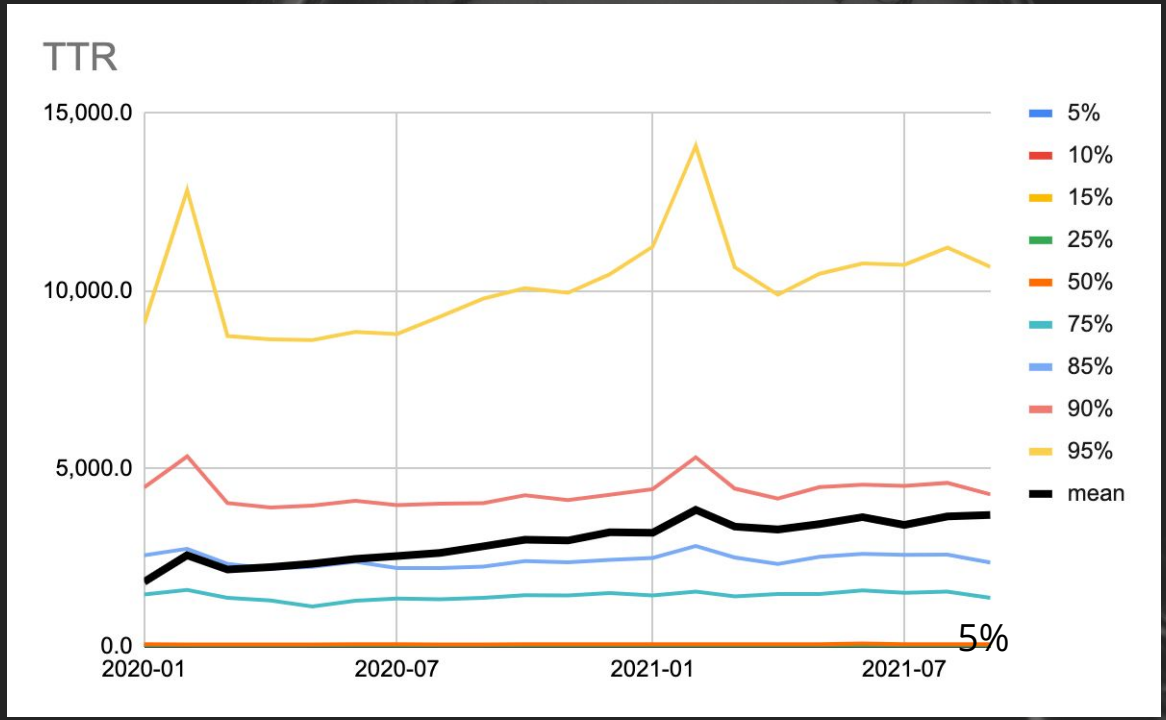


“...the most robust — and certainly the fastest — solution to a broken build is to simply revert the offending commit, allowing troubleshooting to happen in a way that doesn’t interfere with the rest of the team. You can’t know whether a new build works or not unless you’re starting from a known good position, which means you should never allow a new build to start on a red build unless it’s explicitly designed to fix it, and it’s hard to imagine a commit more likely to fix a broken build than simply reverting the one that broke it to begin with.”

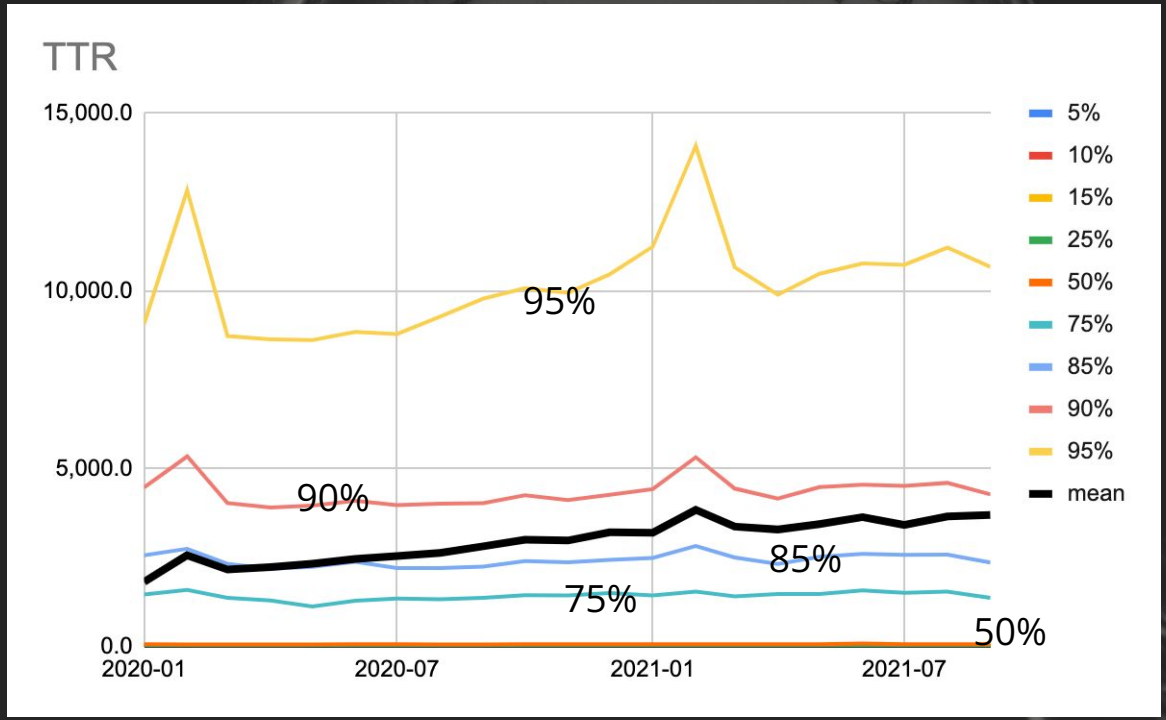
- Brandon Byers, Head of Technology, NA @ Thoughtworks

Photo by Brett Sayles from Pexels

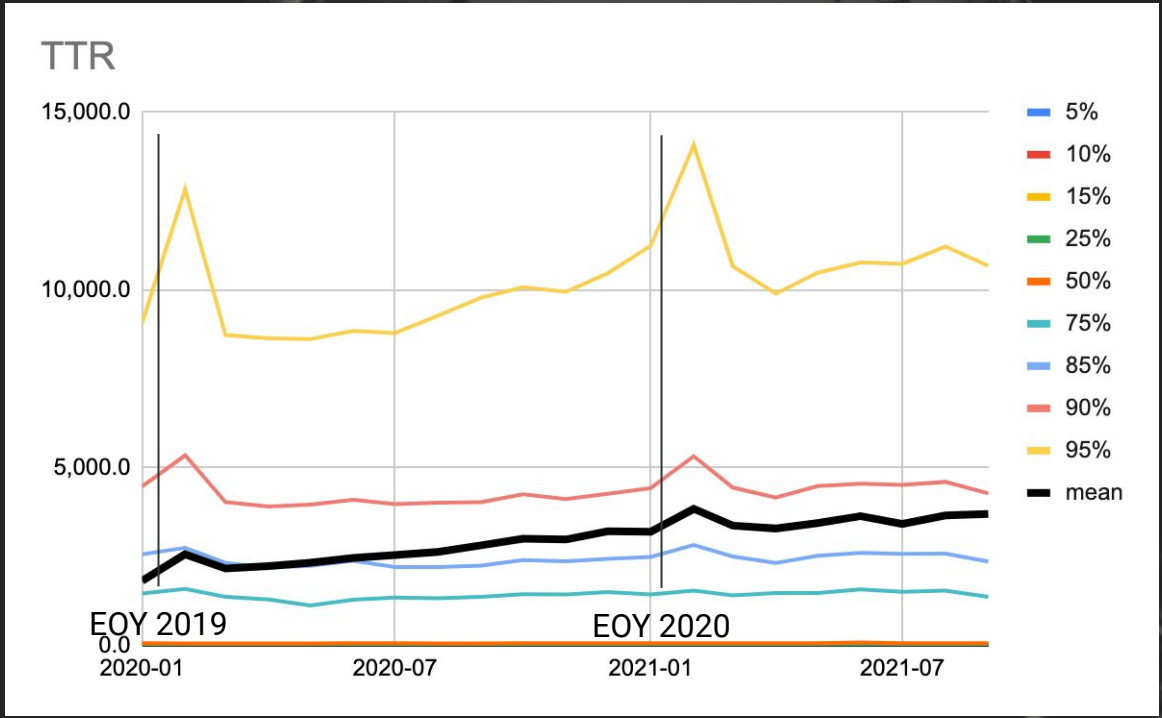
Recovery Time



Recovery Time



Recovery Time

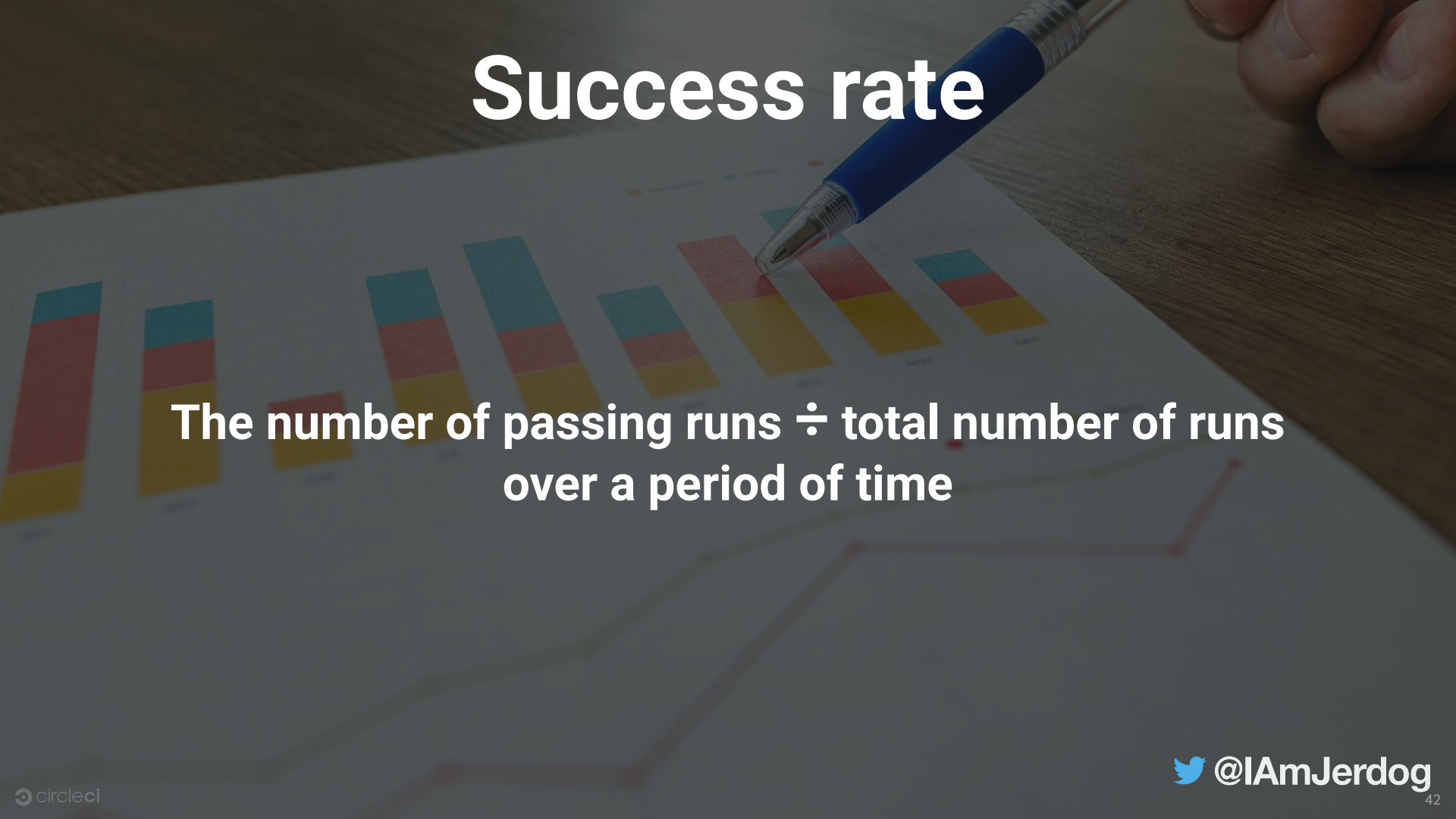


High-performing Teams & MTTR

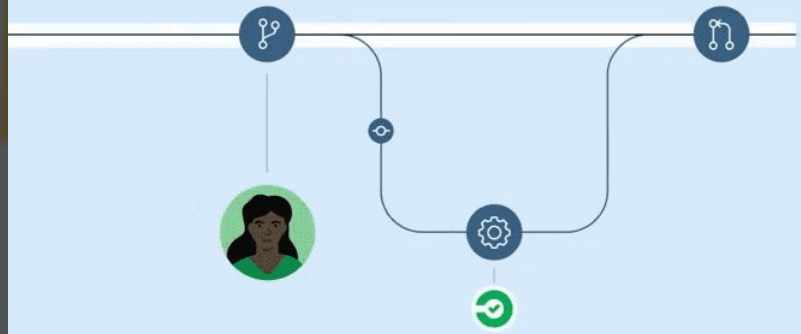
- Duration is the most important factor to optimizing TTR - optimize it first
- Implement tooling for rapid identification and notification of failure
- Write tests to include expert error reporting to quickly identify the problem
- Debug on the remote machine where failure occurs, or at least rich, robust, verbose log output

Photo by Lukas from Pexels

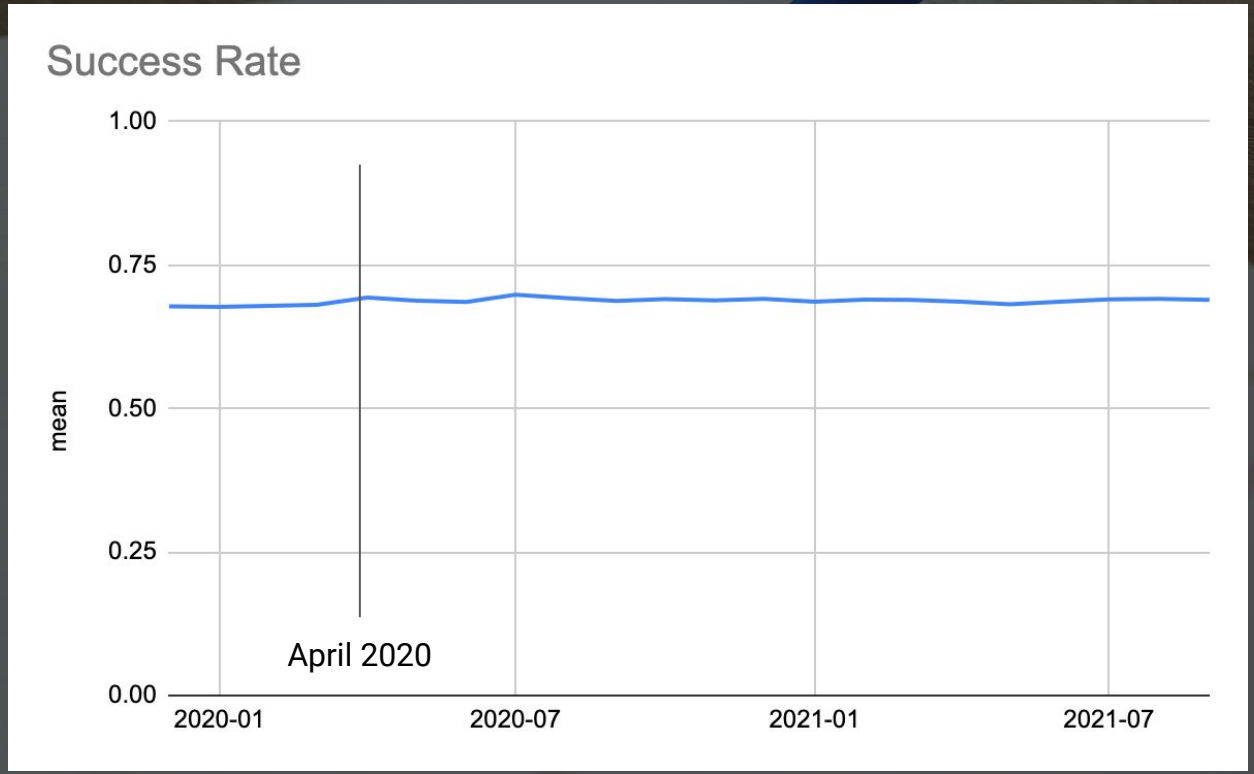
Success rate



The number of passing runs ÷ total number of runs over a period of time



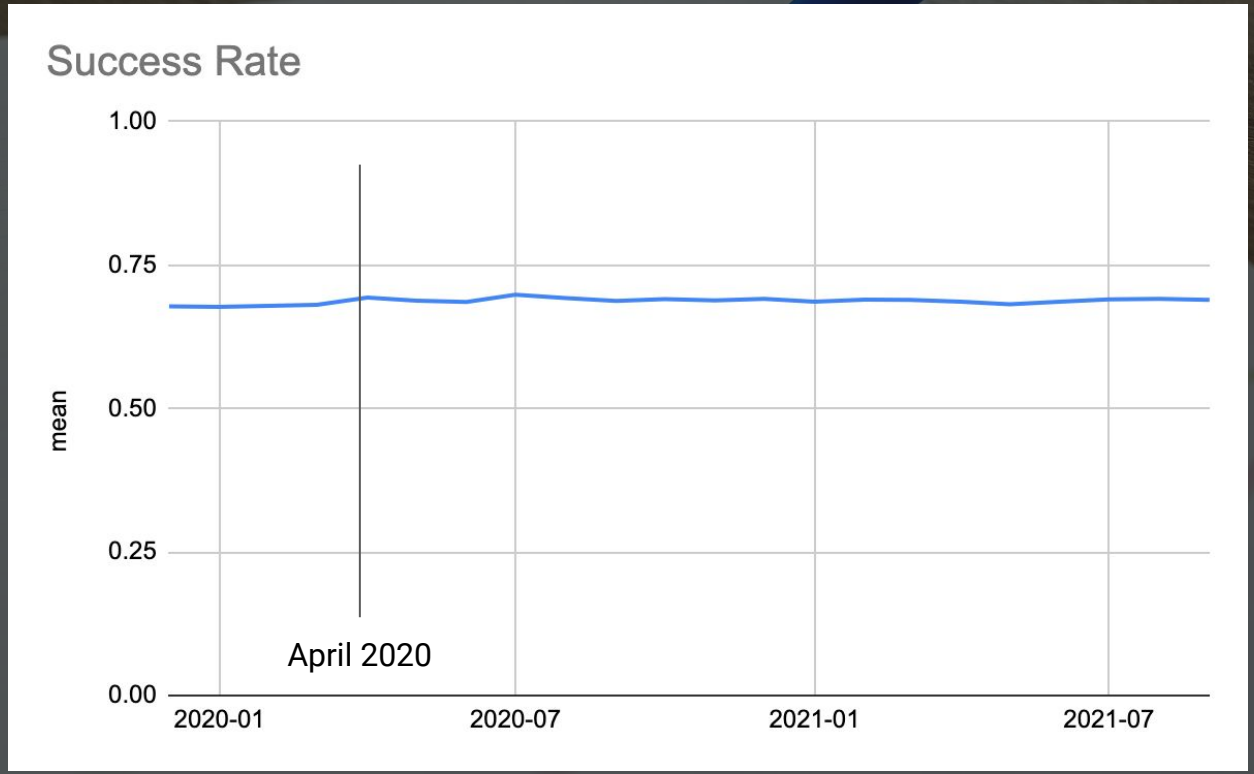
Success rate



@IAmJerdog

Success rate

Success rate



Default branches should have high success rates; feature and dev branches should have low success rates

High-performing Teams & Success rate

- Success rate should always be high on the primary branch, feature branches shouldn’t be
- Feature branches should have lower Success rates without negatively affecting the product, but monitor MTTR for signs of insufficient test output

So what should a high-performing team look like?

	2019 (median)	2020 (median)	This Year (median)	Benchmark
Duration The average length of time for a workflow to run	3.38 min	3.96 min	**3.7 min**	5-10 minutes
TTR The average time between failures & their next success	52.5	55.11	**73.6 min**	< 60 minutes
Success rate The number of successful runs / the total number of runs over a period of time	60%	61%	**77%**	Average should be +90% on default branch
Throughput The average number of workflow runs per day	0.80/day	0.70/day	**1.43/day**	As often as your business requires - not a function of your tooling

Vertical splits

Duration in minutes	
Industry	50th percentile
Health Care Providers & Services	4.46
Hotels, Restaurants & Leisure	3.96
Professional Services	3.88
Internet Software & Services	3.80
Real Estate	3.72
Consumer Discretionary	3.66
Diversified Financial Services	3.55
Media	3.21

MTTR in minutes	
Industry	50th percentile
Hotels, Restaurants & Leisure	60.4
Internet Software & Services	72.1
Diversified Financial Services	73.2
Media	84.0
Professional Services	85.0
Health Care Providers & Services	99.5
Consumer Discretionary	273
Real Estate	318

Throughput	
Industry	50th percentile
Health Care Providers & Services	1.68
Hotels, Restaurants & Leisure	1.68
Real Estate	1.68
Consumer Discretionary	1.61
Diversified Financial Services	1.61
Internet Software & Services	1.46
Professional Services	1.04
Media	1.00

Success Rate	
Industry	Average
Media	71.9%
Consumer Discretionary	70.1%
Real Estate	69.9%
Internet Software & Services	69.0%
Professional Services	68.5%
Diversified Financial Services	68.0%
Health Care Providers & Services	66.5%
Hotels, Restaurants & Leisure	61.1%

Elite Performer validation

Software delivery performance metric	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

50th percentile on CircleCI fit into the “Elite performer” category on the 2021 State of DevOps report

@IAmJerdog

2020 Report

https://circle.ci/ssd2020

Full 2022 Report

https://circle.ci/ssd2022

circleci

Enter for a chance to win

Sony WF-1000XM4 Wireless
Noise Canceling Earphones

&

Yamazaki 12 Years
Single Malt Whiskey

The winner will be notified by email on Sept 8, 2022

Thank you.

For feedback and swag: **circle.ci/jeremy**


Timeline.jerdog.me


IAmJerdog

jerdog

/in/jeremymeiss