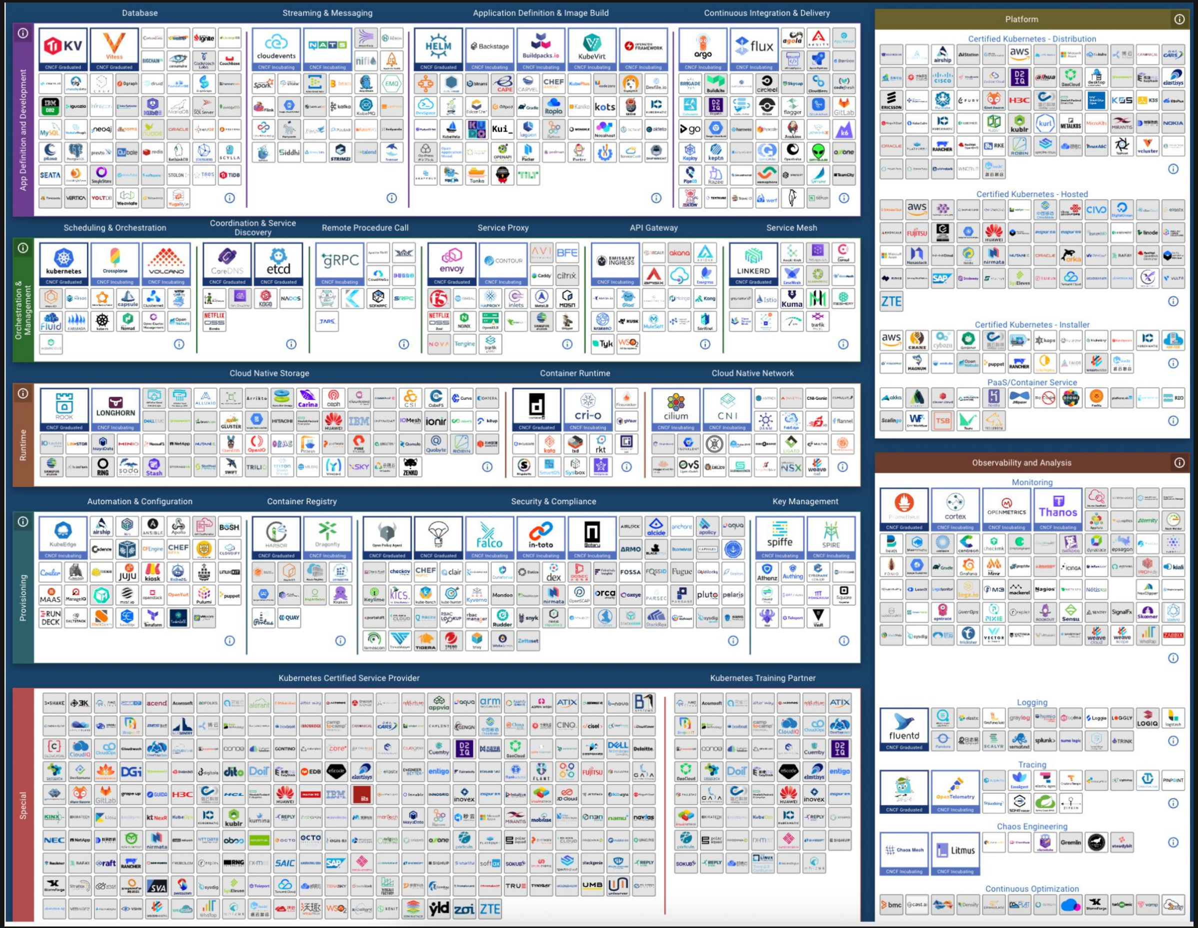
Learnings on Better Software Delivery Principles Through a Panini

DevOpsDays
BIRMINGHAM, AL

@IAmJerdog

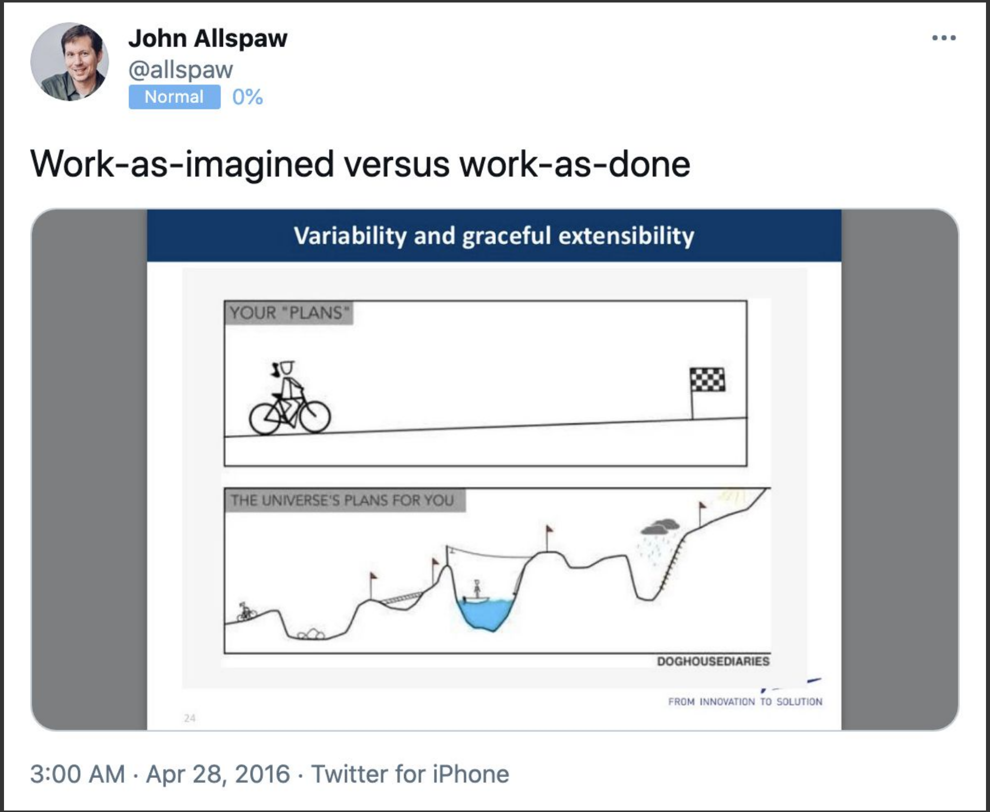
Pixabay via athree23


App Definition and Development
Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
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
Observability and Analysis
Monitoring
Logging
Tracing
Chaos Engineering
Continuous Optimization
Special
Kubernetes Certified Service Provider
Kubernetes Training Partner

Pixabay via athree23

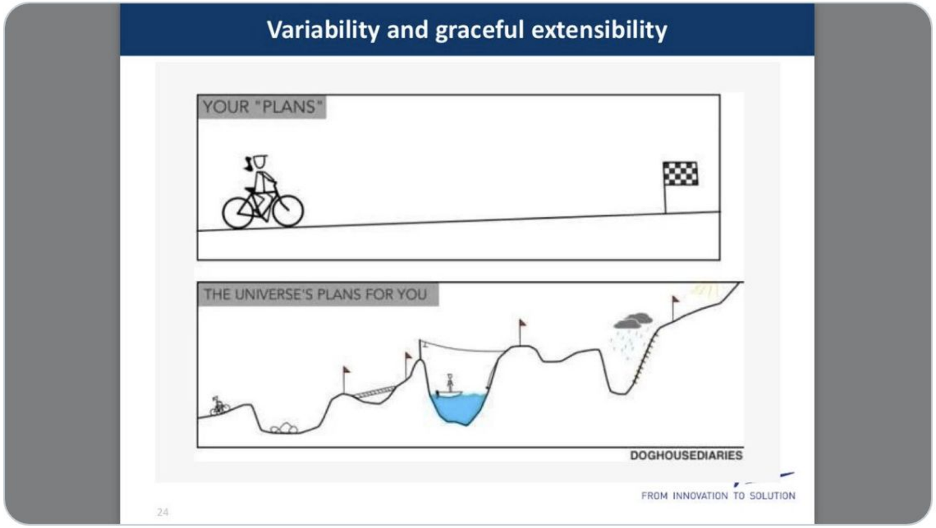
circleci

@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

Jeremy Meiss

Director, DevRel & Community

circleci

DevOpsDays BIRMINGHAM, AL

@IAmJerdog

201
STATE
DevO
REPO
2016 State of DevOps Report
Presented by:
pup
2017 State of DevOps Report
Presented by:
puppet + DORA DEVOPS RESEARCH & ASSESSMENT
Sponsored by:
Atlassian
splunk>
amazon web services
WAVEFRONT by vmware
Hewlett Packard Enterprise
Deloitte.
ElectricCloud
2018 State of DevOps Report
puppet + splunk>
aws
cloudability
CYBERARK
Cognizant
diaxion
2019 State of DevOps Report
puppet
circleci
splunk>
ANITIAN
servicenow

@IAmJerdog

Dataset

257 mil+
workflows

44,000+
orgs

290,000+
projects

1,000x
Larger than surveys

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

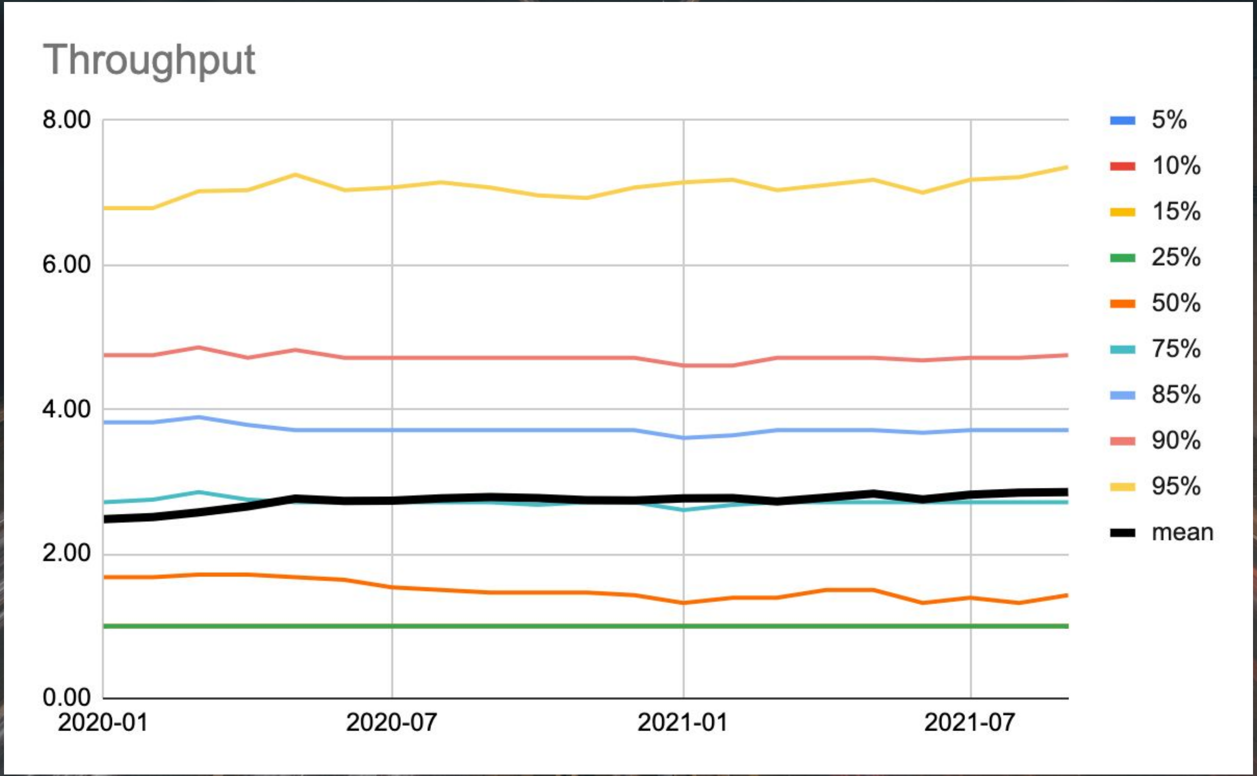
Throughput

the average number of workflow runs per day

@IAmJerdog

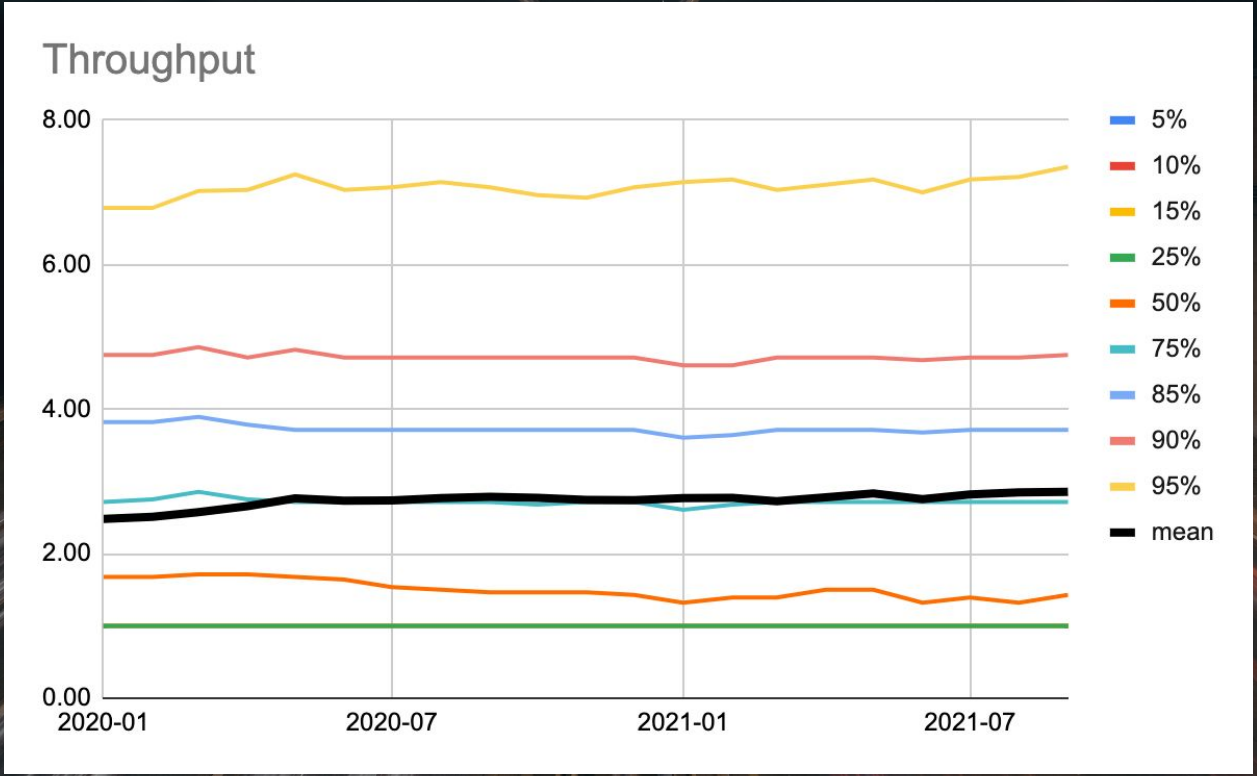
circleci

Throughput

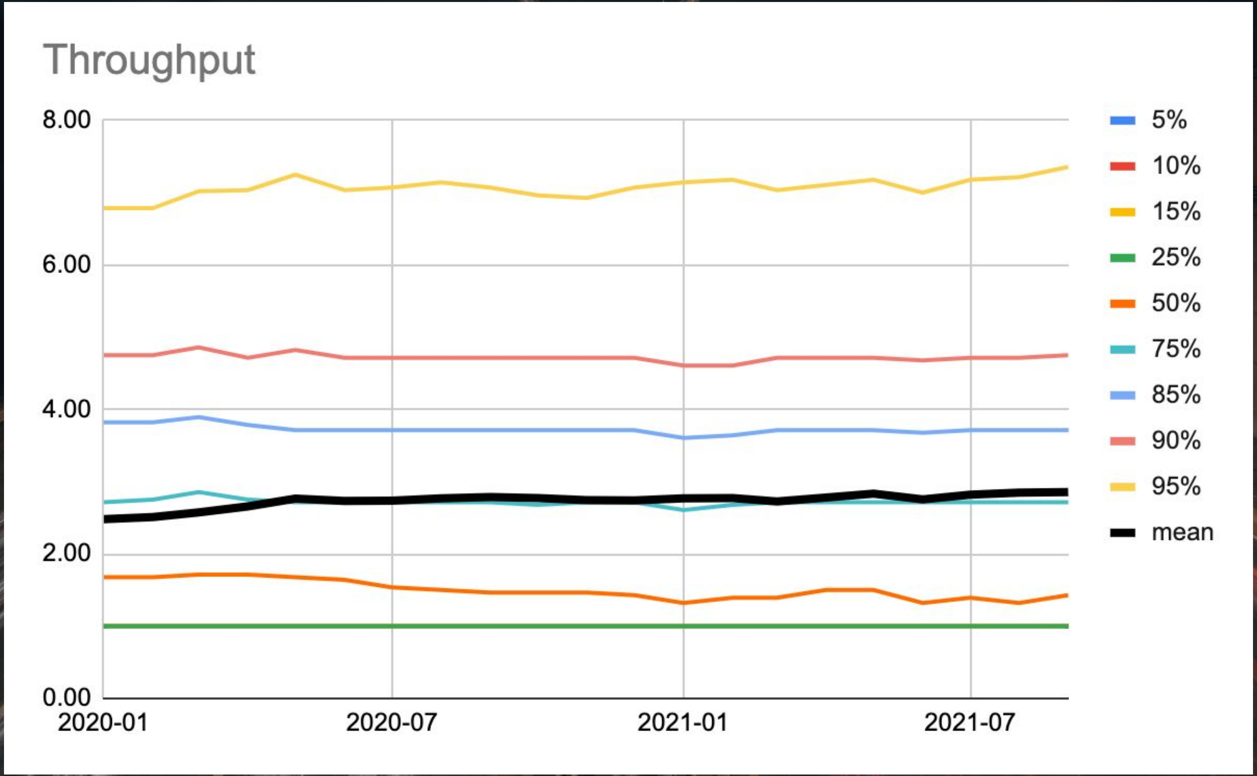


@IAmJerdog

Throughput


Throughput
8.00
6.00
4.00
2.00
0.00
2020-01
2020-07
2021-01
2021-07
5%
10%
15%
25%
50%
75%
85%
90%
95%
mean

Throughput



@IAmJerdog

Most teams are not deploying dozens of times per day

@IAmJerdog

Image by Pawan Kolhe from Pixabay

Image by Pawan Kolhe from Pixabay

Duration

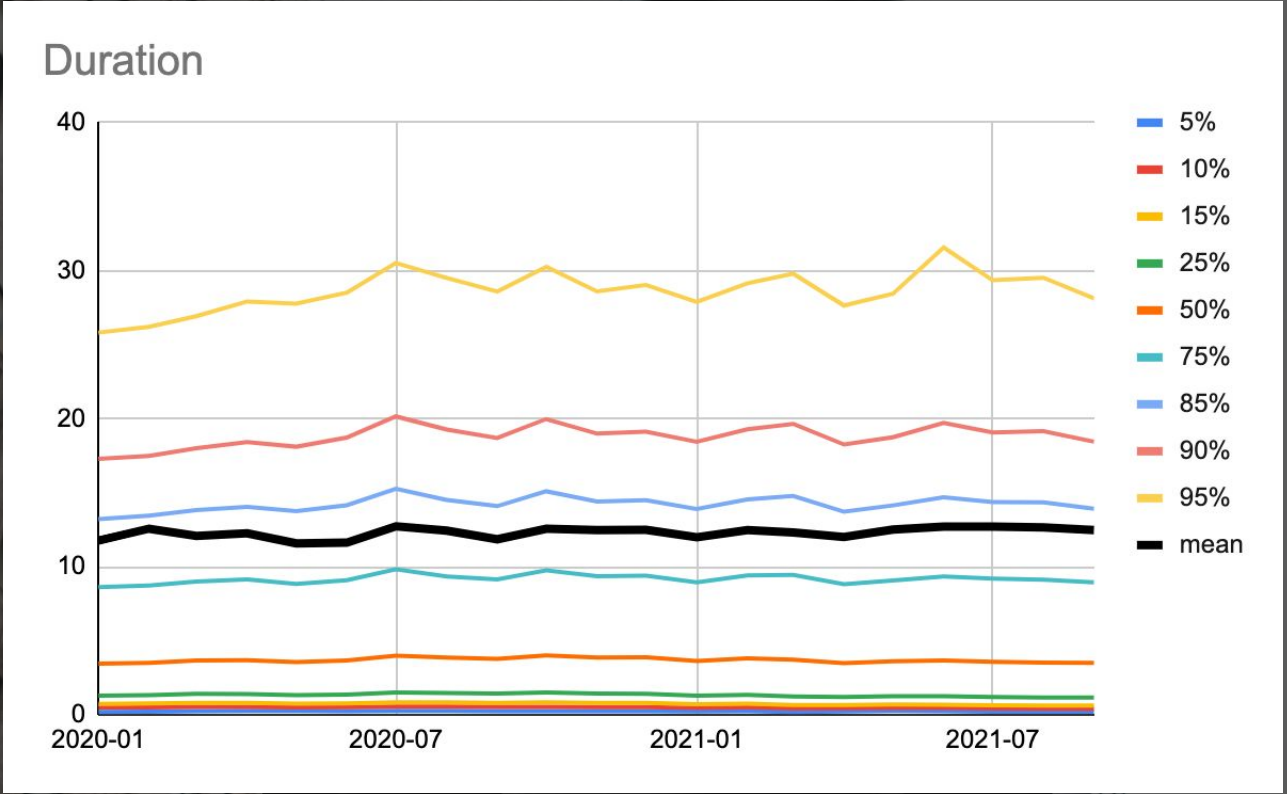
the length of time it takes for a workflow to run

@IAmJerdog

circleci

Image by Pawan Kolhe from Pixabay

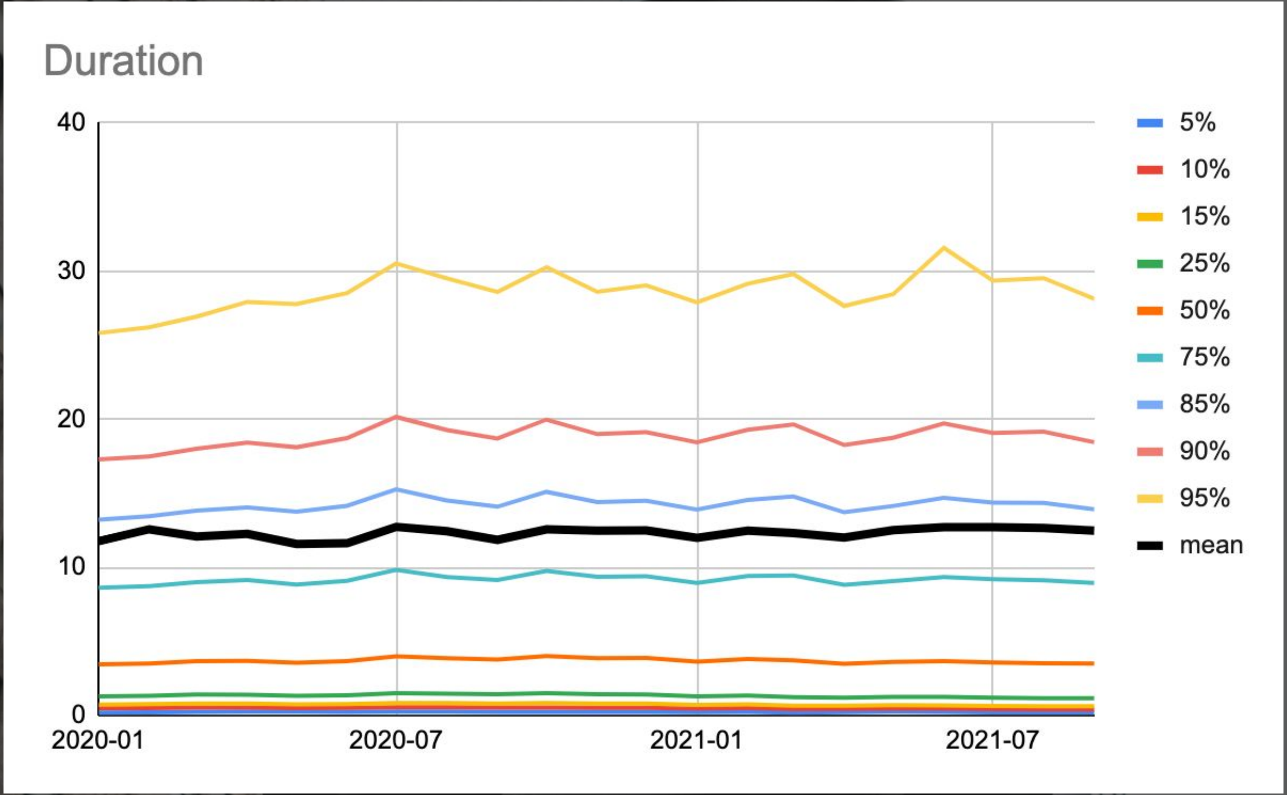
Duration



@IAmJerdog

circleci

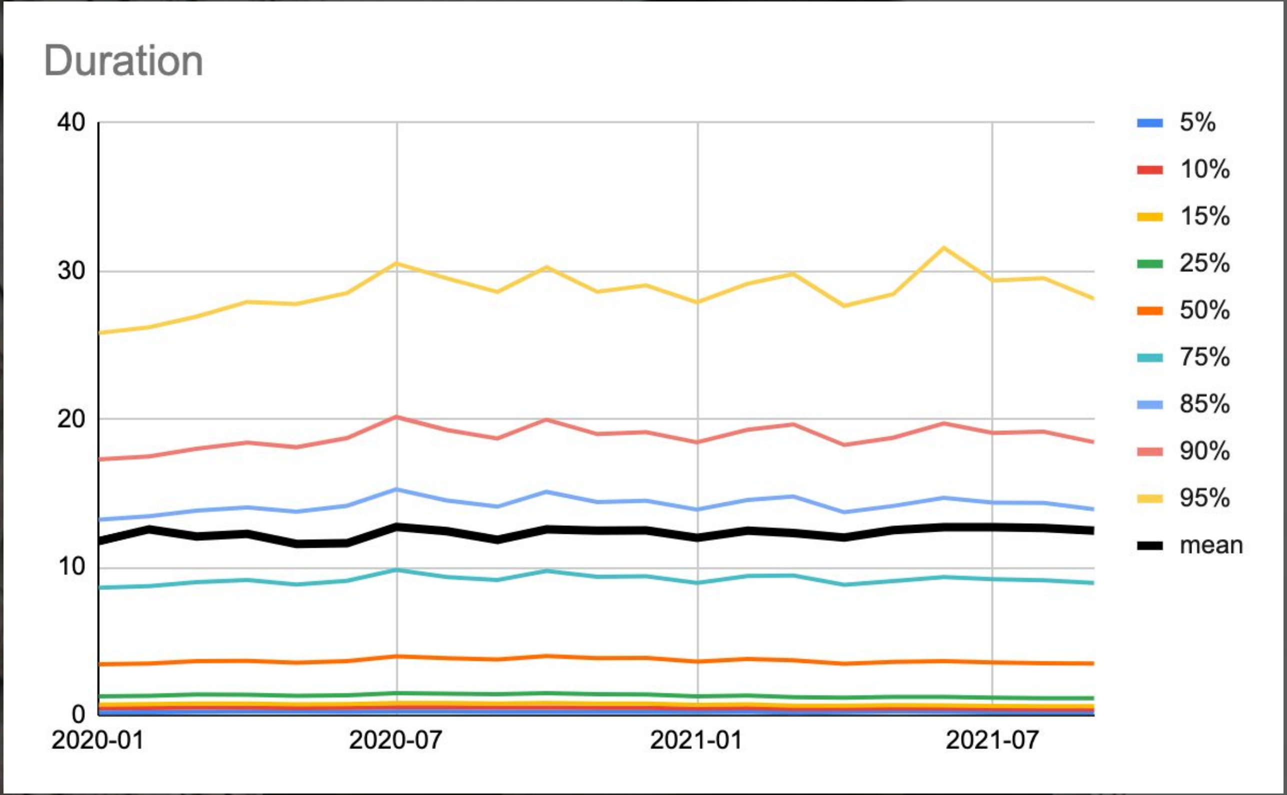
Duration



@IAmJerdog

Image by Pawan Kolhe from Pixabay

Duration



@IAmJerdog

Photo by Brett Sayles from Pexels

Mean time to recovery

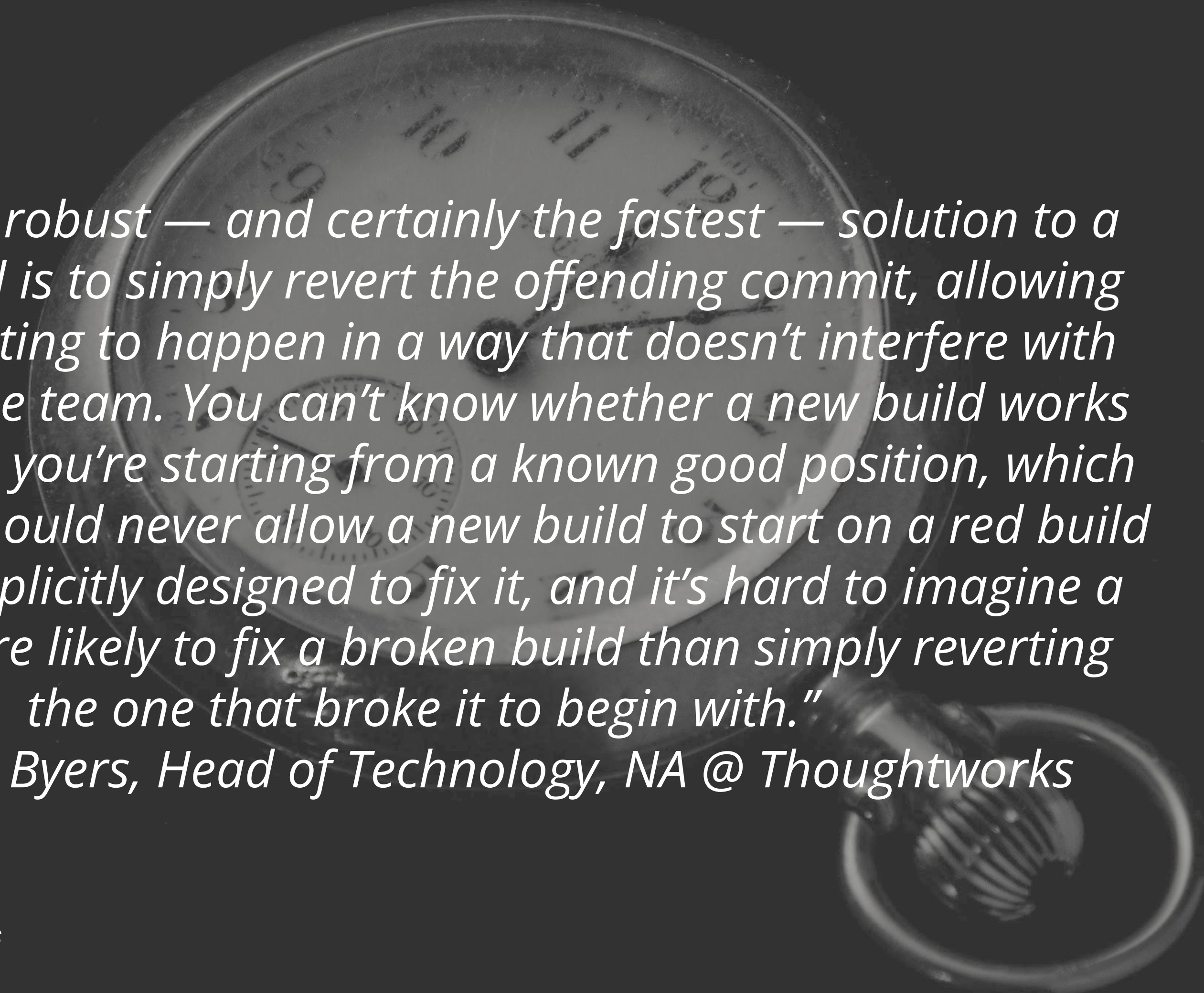
average time between a pipeline's failure and its next success

@IAmJerdog

Mean time to recovery

shortest MTTR $\propto$ Duration

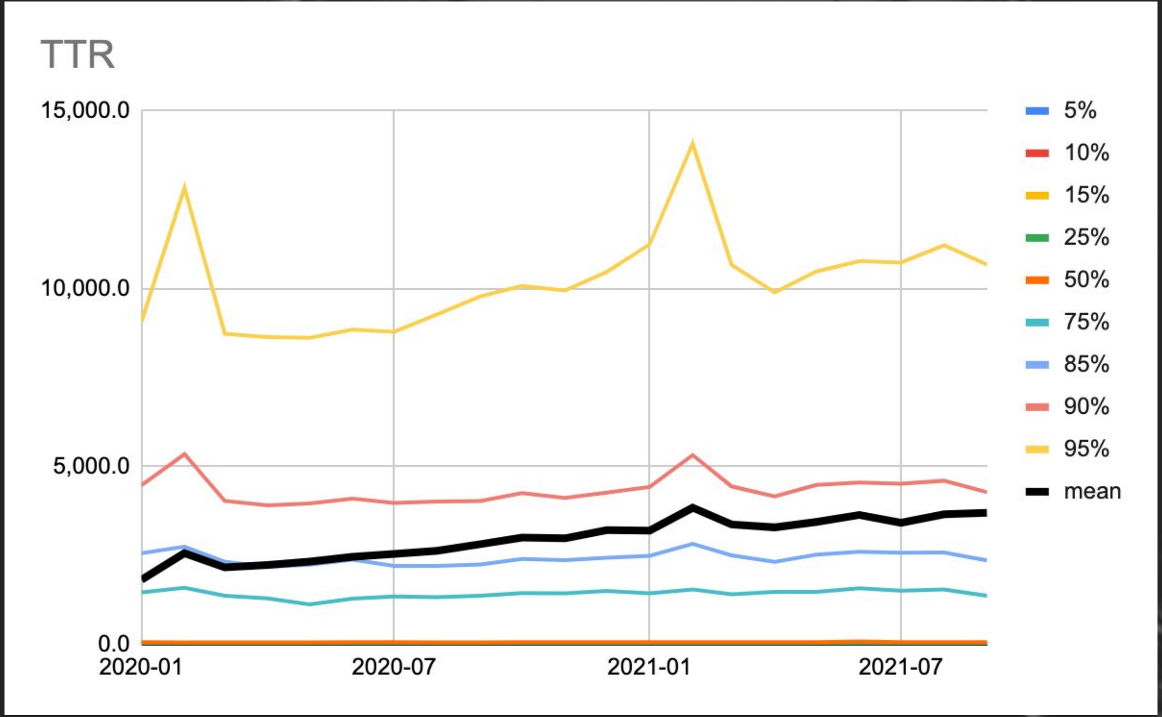
@IAmJerdog



"...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

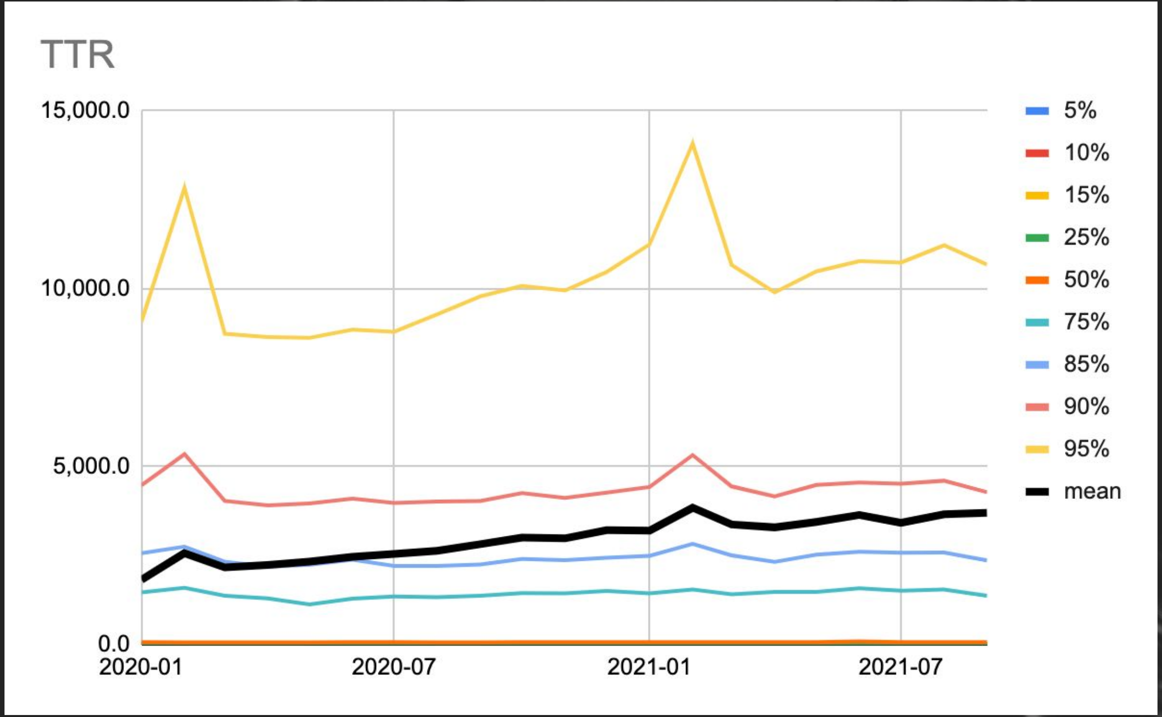
Photo by Brett Sayles from Pexels

Recovery Time



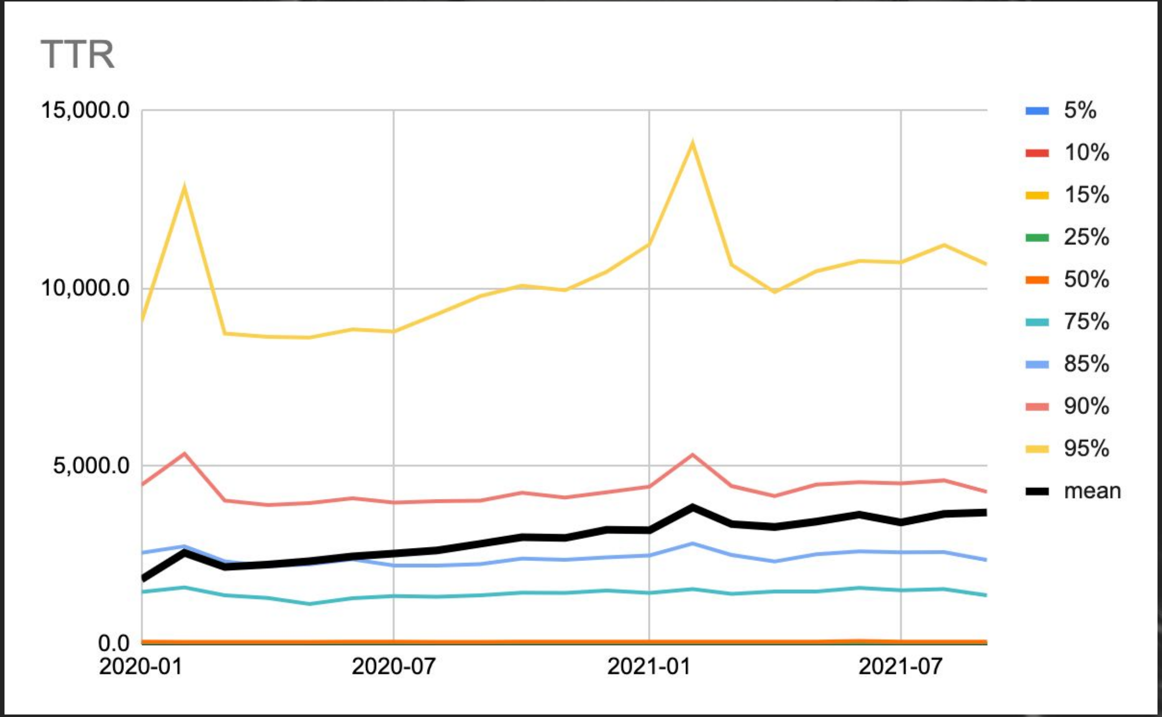
@IAmJerdog

Recovery Time



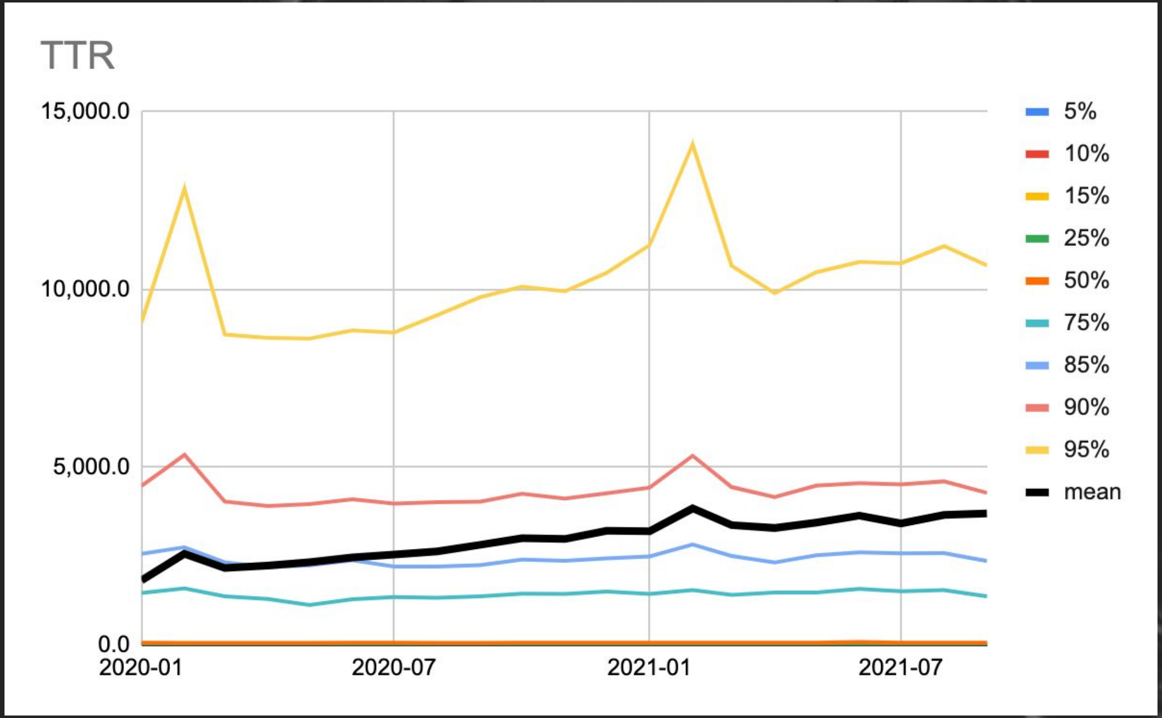
@IAmJerdog

Recovery Time



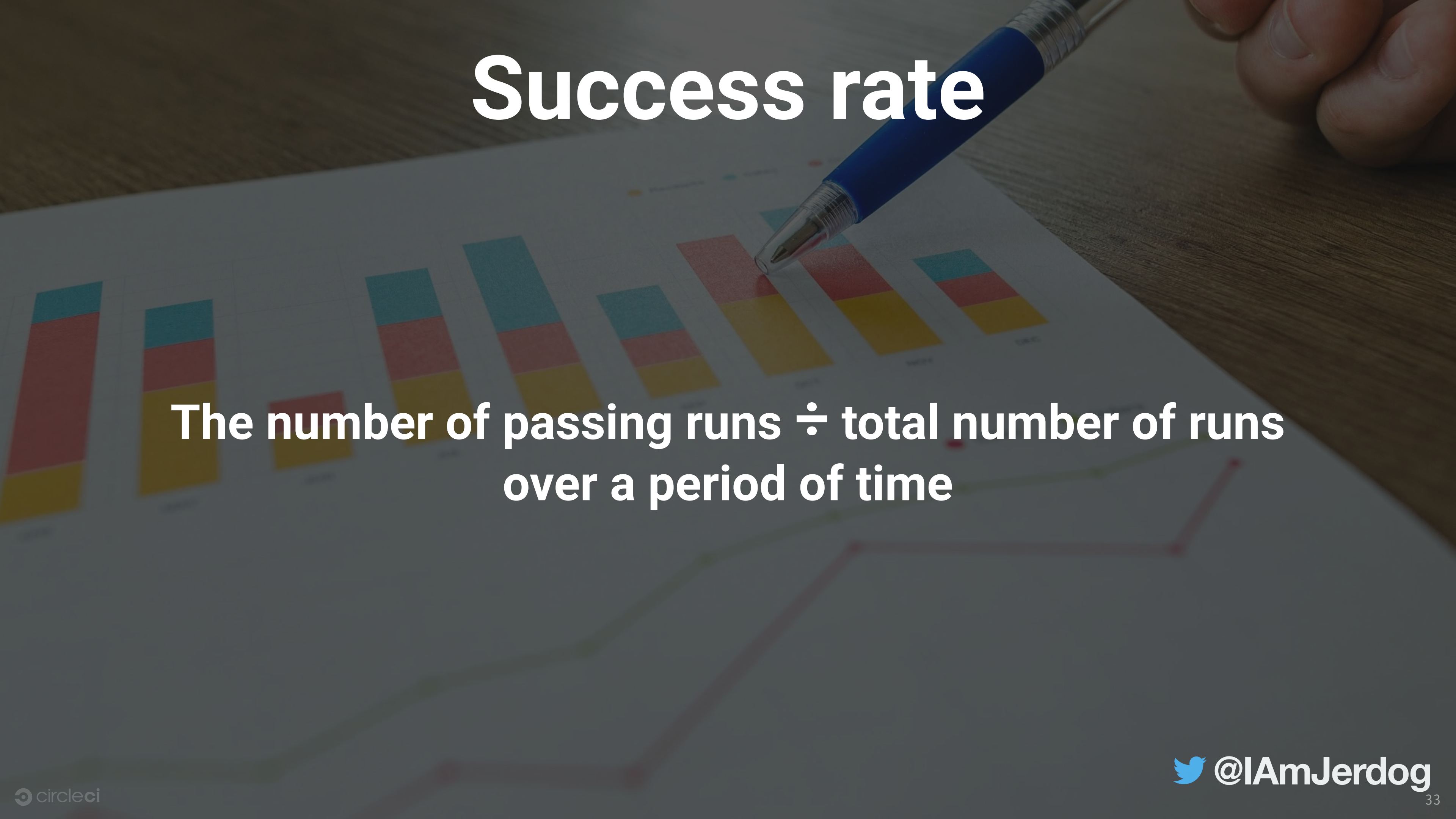
@IAmJerdog

Recovery Time



@IAmJerdog

Photo by Lukas from Pexels

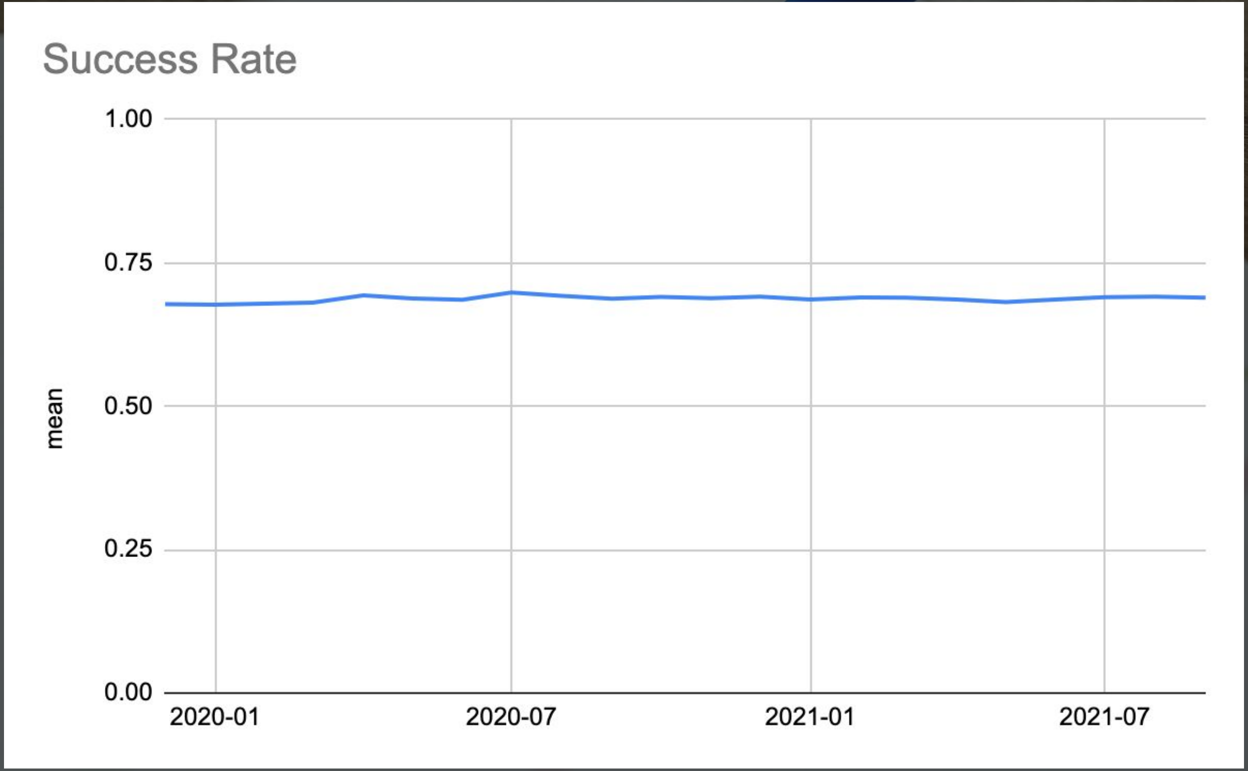


Success rate

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

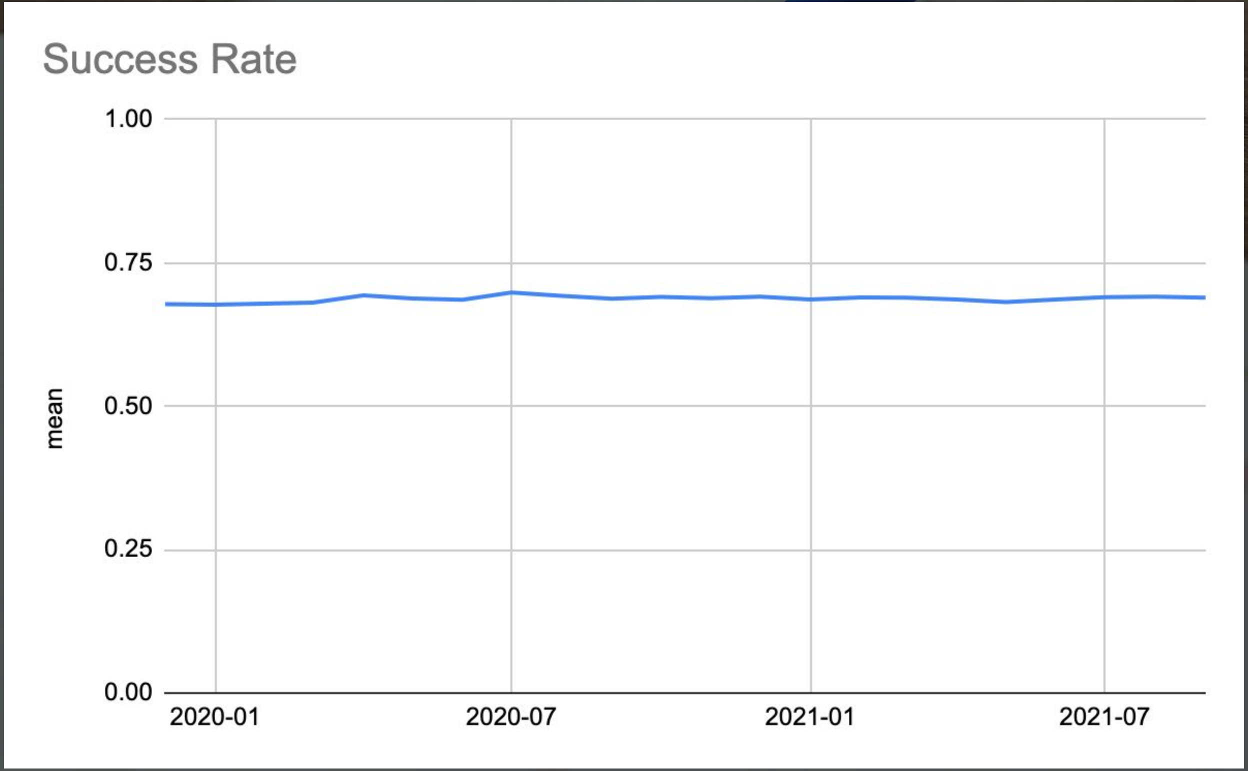
@IAmJerdog

Success rate



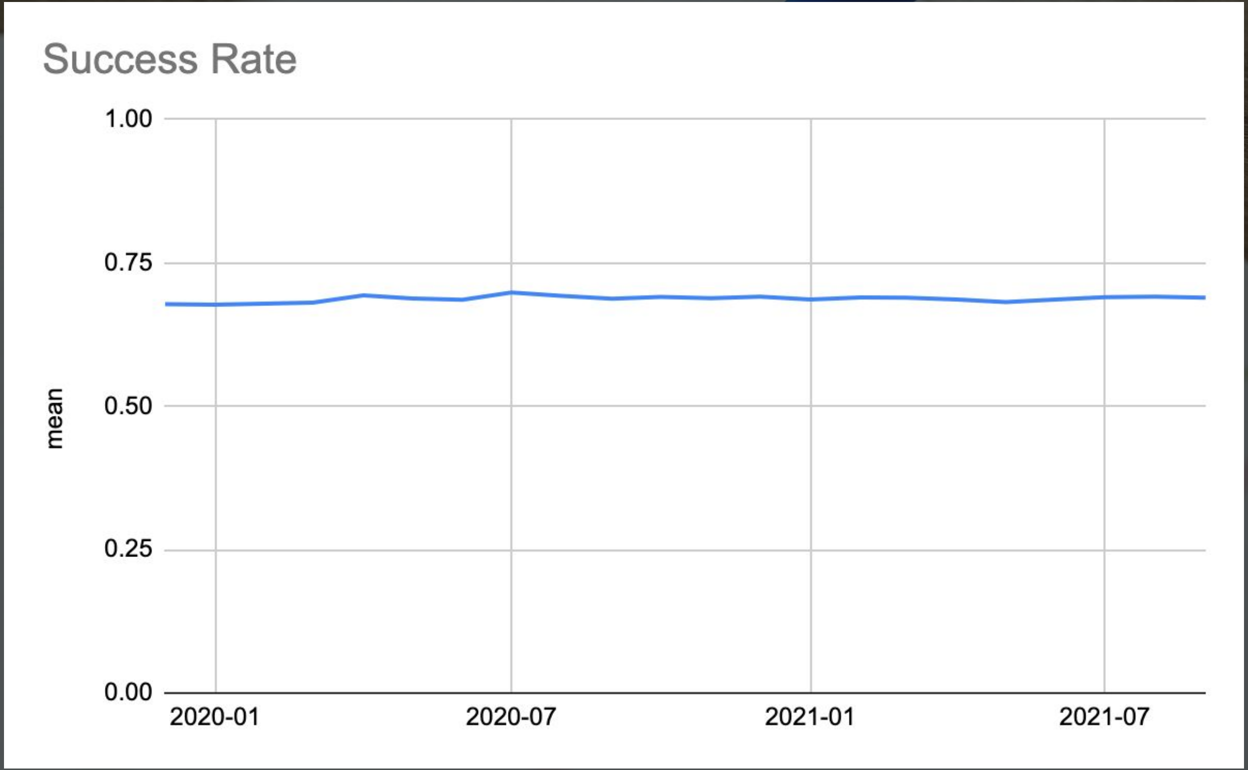
@IAmJerdog

Success rate



@IAmJerdog

Success rate



@IAmJerdog

circleci

	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

circleci

@IAmJerdog

Extra Insights

202x has been a year.

@IAmJerdog

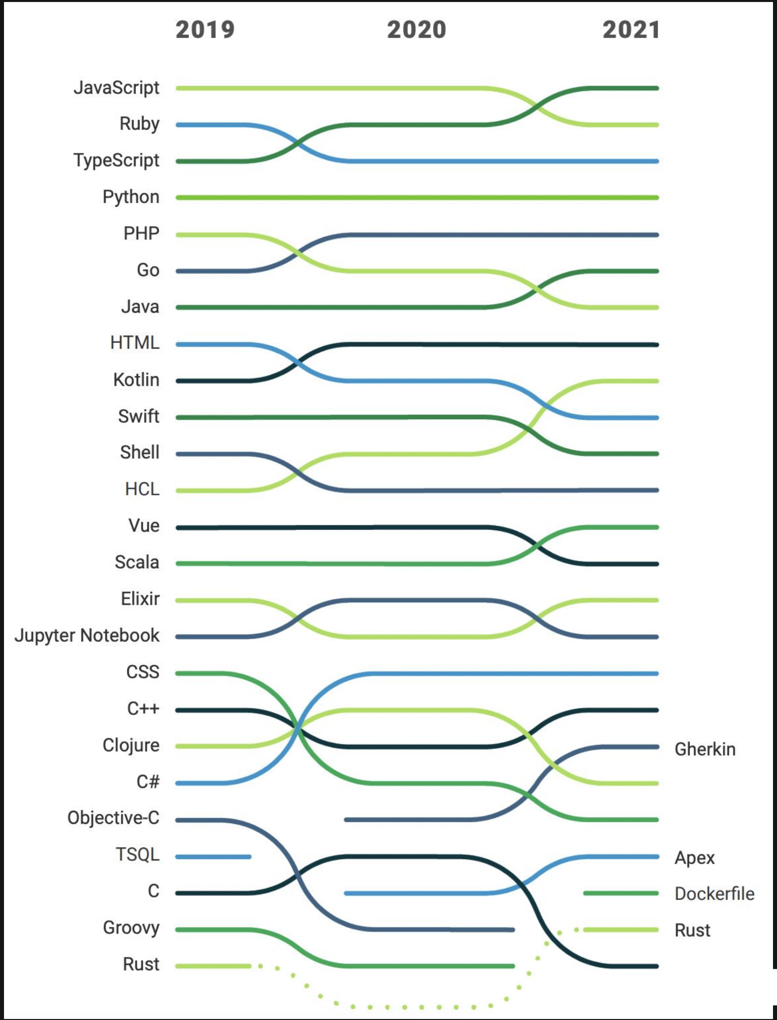
“Don’t deploy on Friday” is not a thing.

@IAmJerdog

“Don’t Deploy on Friday” is not a thing

- 70% less **Throughput** on weekends
- 11% less **Throughput** on Friday (UTC)
- 9% less **Throughput** on Monday (UTC)

Language shifts over the last few years



og

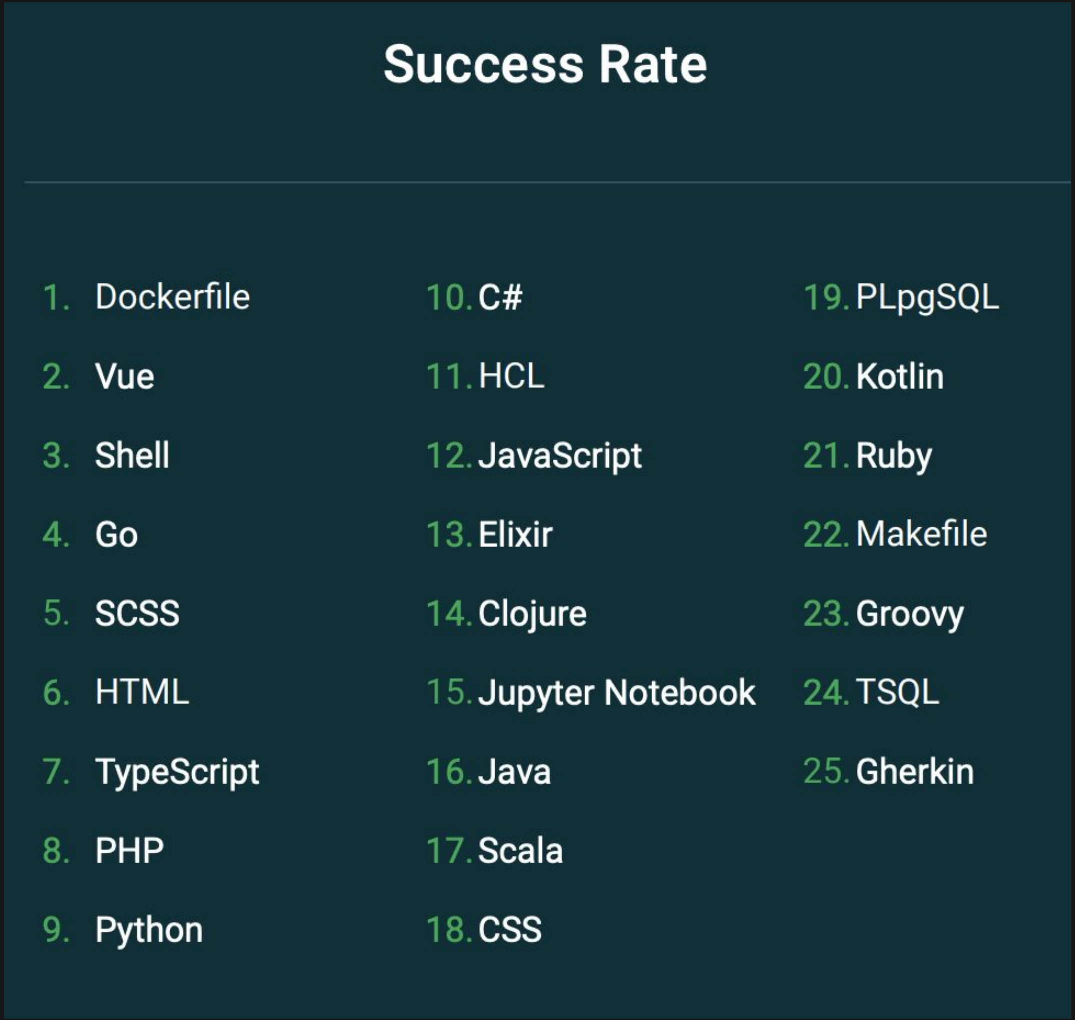
circleci

Duration

1. Batchfile
2. SaltStack
3. Makefile
4. Smarty
5. Jsonnet
6. Shell
7. Mustache
8. HCL
9. FreeMarker
10. Dockerfile
11. PLSQL
12. Jinja
13. Elm
14. Lua
15. Liquid
16. VCL
17. Open Policy Agent
18. Groovy
19. Go
20. Starlark
21. API Blueprint
22. Roff
23. HTML
24. R
25. Python

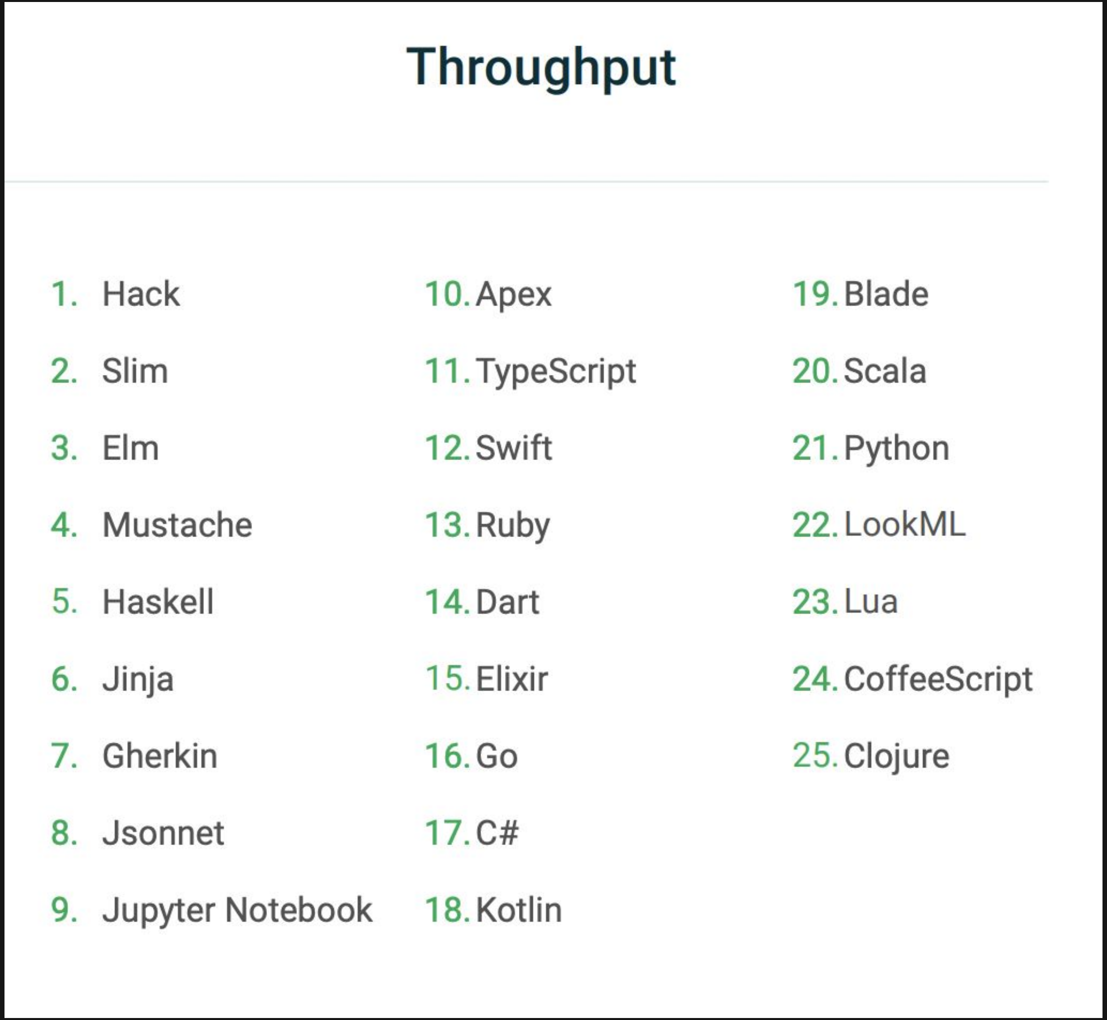
MTTR

1. Gherkin
2. HCL
3. JavaScript
4. Go
5. Clojure
6. C#
7. Vue
8. TypeScript
9. Ruby
10. Python
11. PHP
12. Perl
13. Shell
14. Kotlin
15. Elixir
16. HTML
17. Scala
18. Jupyter Notebook
19. Java
20. Swift
21. Apex
22. CSS
23. C++
24. Rust
25. C



Success Rate

1. Dockerfile
2. Vue
3. Shell
4. Go
5. SCSS
6. HTML
7. TypeScript
8. PHP
9. Python
10. C#
11. HCL
12. JavaScript
13. Elixir
14. Clojure
15. Jupyter Notebook
16. Java
17. Scala
18. CSS
19. PLpgSQL
20. Kotlin
21. Ruby
22. Makefile
23. Groovy
24. TSQL
25. Gherkin

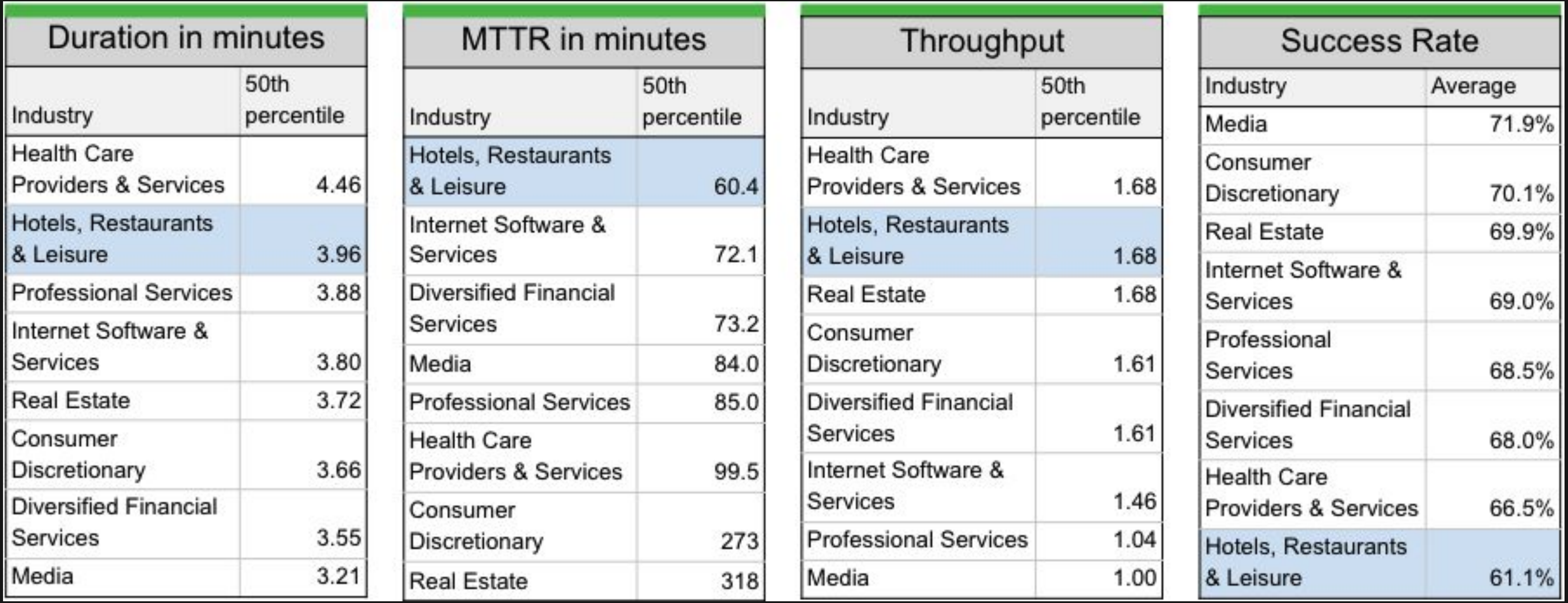


Throughput

1. Hack
2. Slim
3. Elm
4. Mustache
5. Haskell
6. Jinja
7. Gherkin
8. Jsonnet
9. Jupyter Notebook
10. Apex
11. TypeScript
12. Swift
13. Ruby
14. Dart
15. Elixir
16. Go
17. C#
18. Kotlin
19. Blade
20. Scala
21. Python
22. LookML
23. Lua
24. CoffeeScript
25. Clojure

@IAmJerdog

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

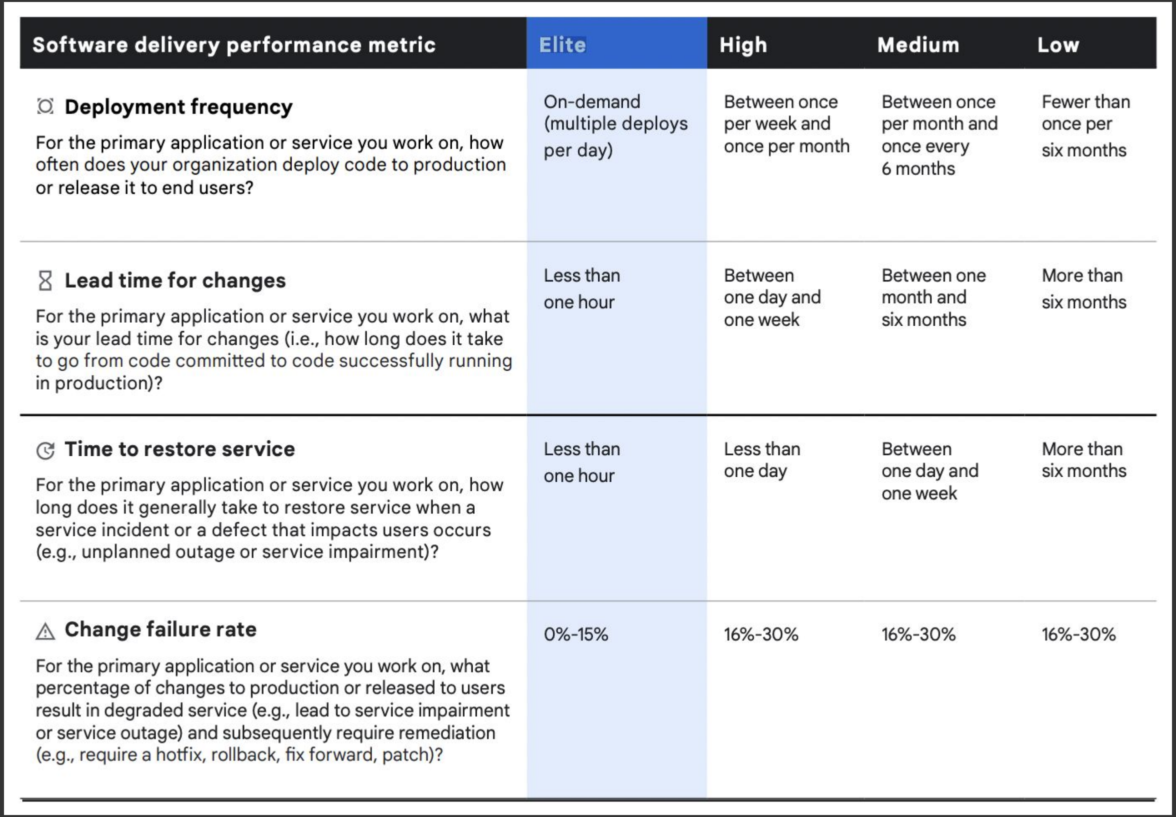
MTTR in minutes	
Industry	50th percentile
Hotels, Restaurants & Leisure	60.4
Internet Software & Services	72.1
Diversified Financial Services	73.2
Media	84.0
Professional Services	85.0
Health Care Providers & Services	99.5
Consumer Discretionary	273
Real Estate	318

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%

@IAmJerdog

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

Full 2022 Report

https://circle.ci/ssd2020

https://circle.ci/ssd2022

@IAmJerdog

Thank you.

For feedback and swag: circle.ci/jeremy

Timeline.jerdog.me

IAmJerdog

jerdog

/in/jeremymeiss

DevOpsDays BIRMINGHAM, AL

@IAmJerdog