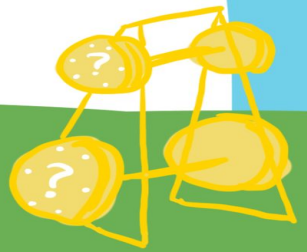
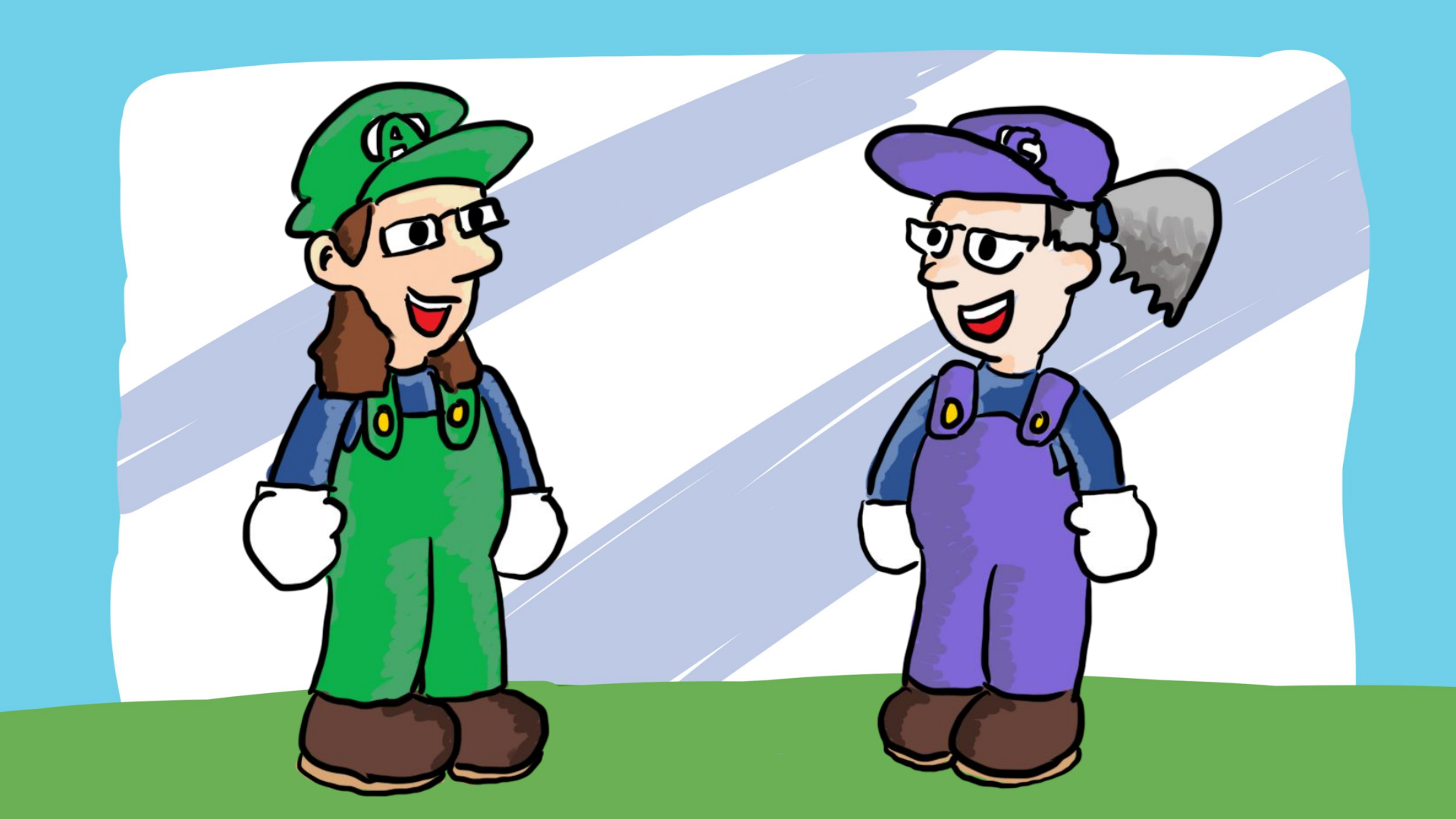
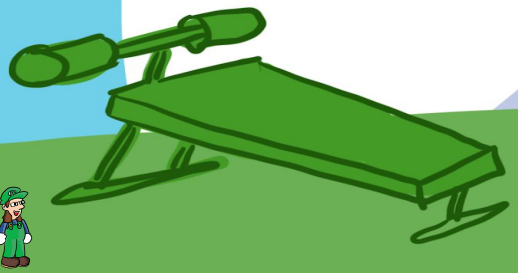
Measuring the Robocat

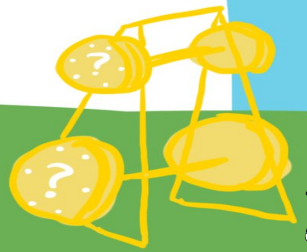
How Tekton Continuously Delivers Tekton



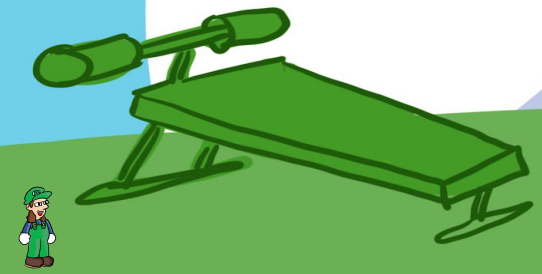


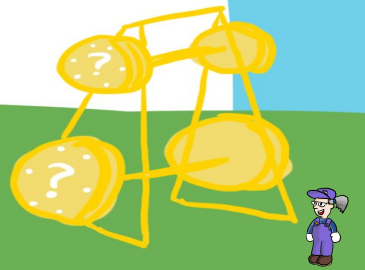
Wouldn’t it be nice to have a fitness tracker for Continuous Delivery?



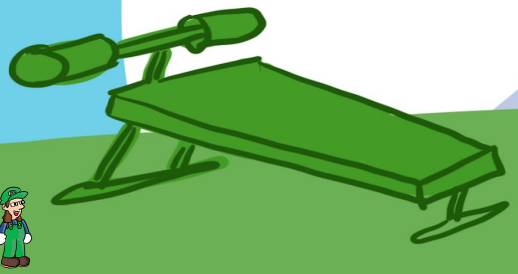


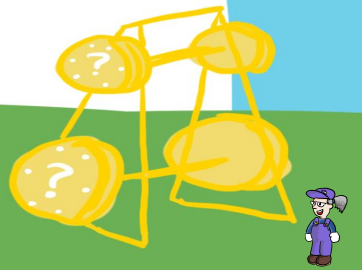
DORA metrics and best practices





Measurement will help you focus your efforts where they are most beneficial





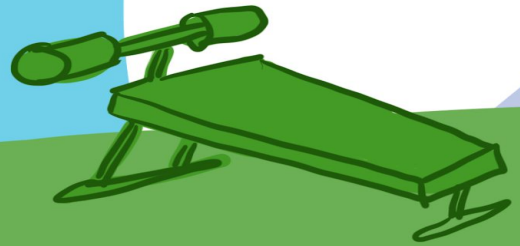
Christie Wilson

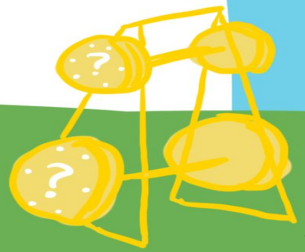
Software Engineer at Google
Tekton Lead

Andrea Frittoli

Software Engineer at IBM
Tekton Lead

What is Tekton?





Tekton is cloud native CD building blocks

Built on Kubernetes

- Built using Kubernetes Custom Resources (CRDs)
- Designed around the Kubernetes Resource Model (KRM)

Tekton is a founding project of the CDF

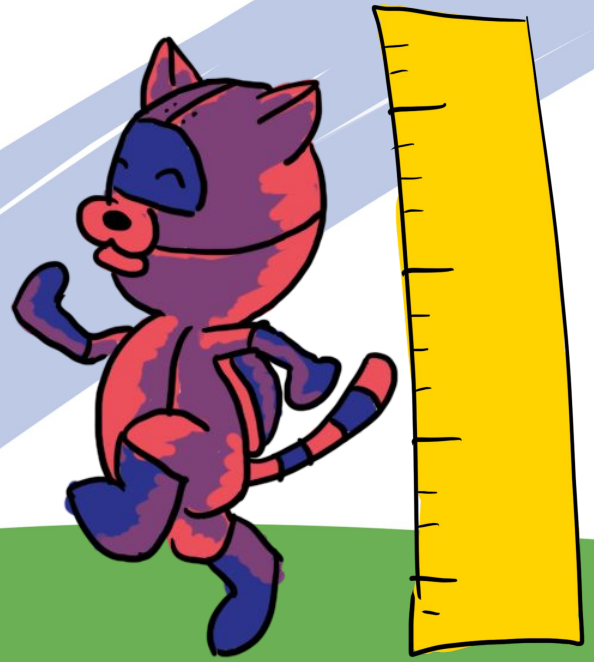
CD.FOUNDATION

- Donated to the CDF
- Group effort, with contributors from more than 150 companies!

Continuous Delivery Inception

Who measures the CD of the CD system?

- Tekton is a system for CD
- But we also continuously deliver Tekton itself!

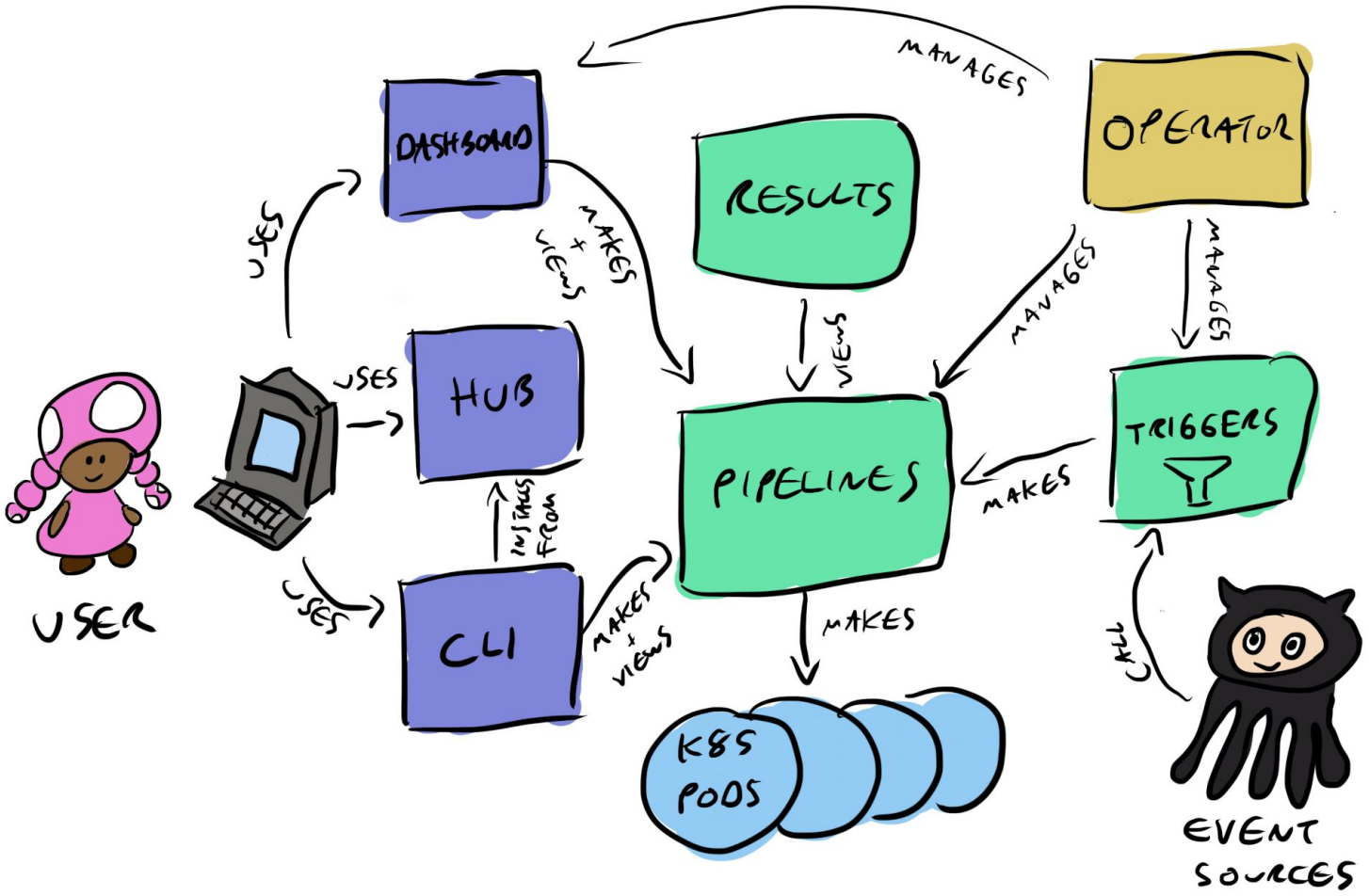


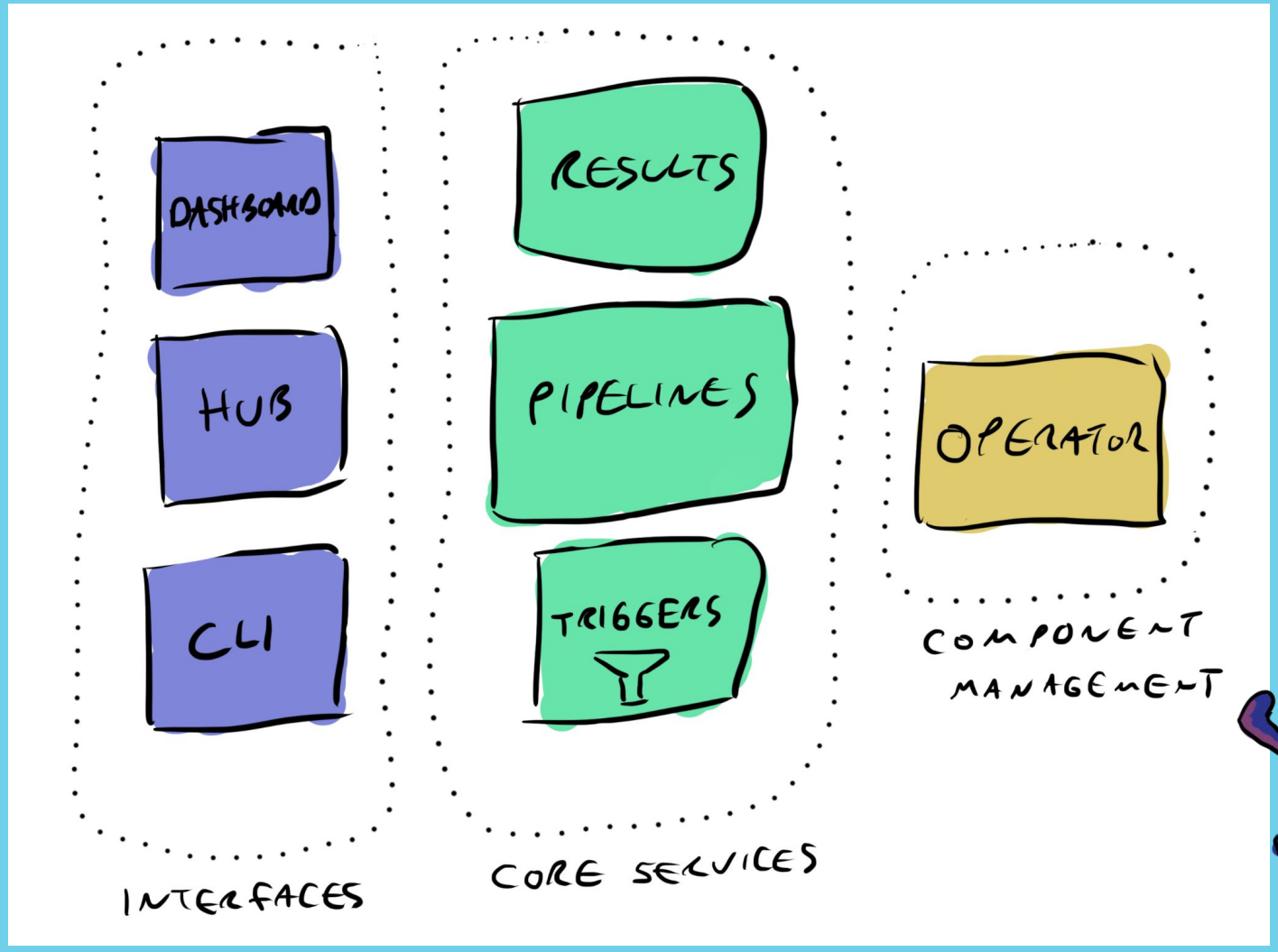
When you say “Tekton” what do you mean....

When we say “Tekton” here, we mean...

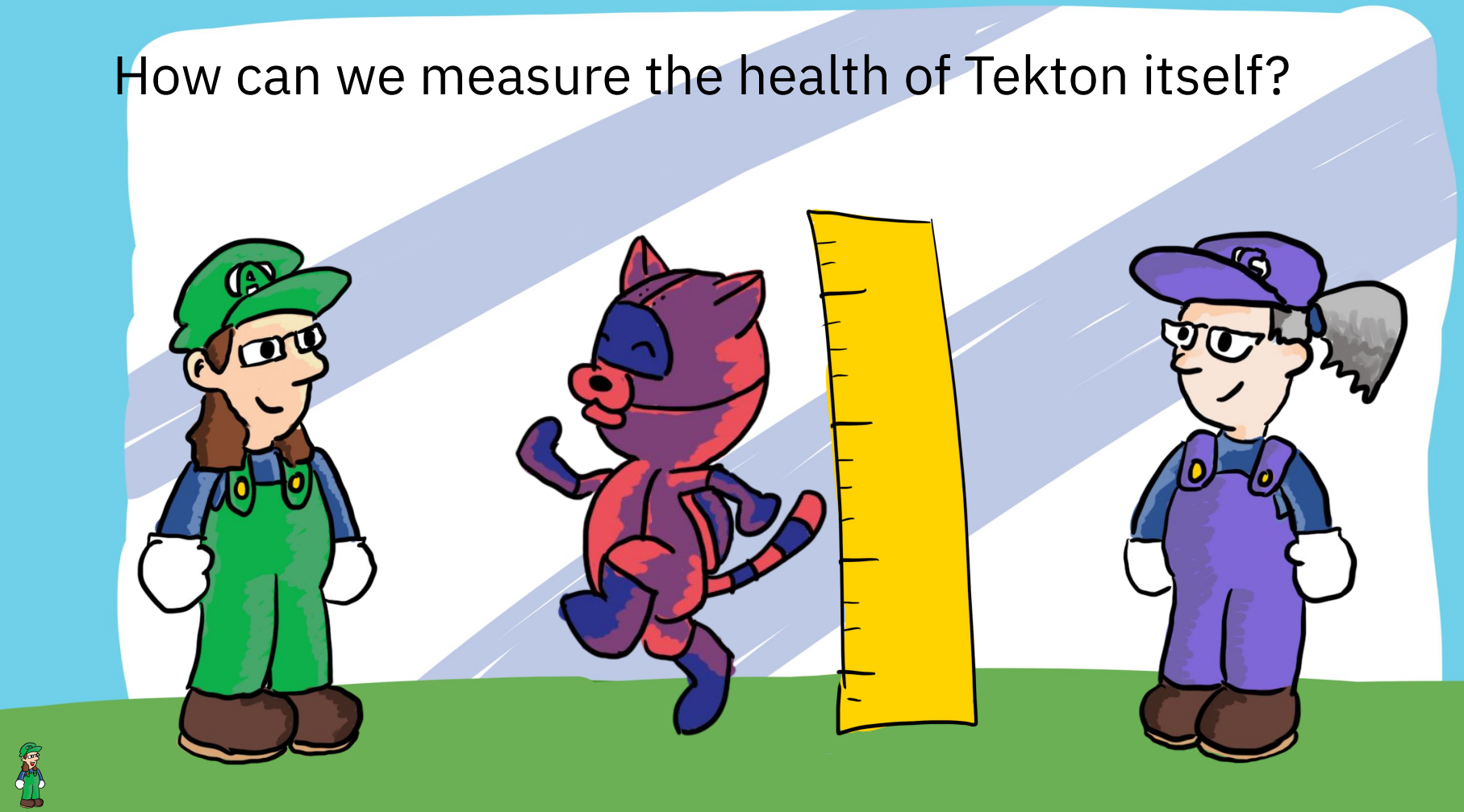
(but there’s even more to Tekton!!)

1. Pipelines
2. Triggers
3. Results
4. CLI
5. Dashboard
6. Hub
7. Operator


MANAGES
DASHBOARD
RESULTS
OPERATOR
USES
MAKES + VIEWS
VIEWS
MANAGES
MANAGES
USES
HUB
TRIGGERS
PIPELINES
INSTALLS FROM
MAKES
USES
CLI
MAKES + VIEWS
MAKES
K8S PODS
CALL
USER
EVENT SOURCES

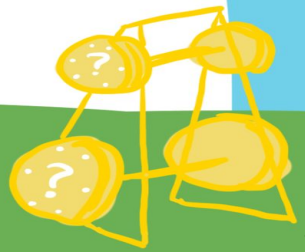

DASHBOARD
HUB
CLI
INTERFACES
RESULTS
PIPELINES
TRIGGERS
CORE SERVICES
OPERATOR
COMPONENT MANAGEMENT

How can we measure the health of Tekton itself?



DORA

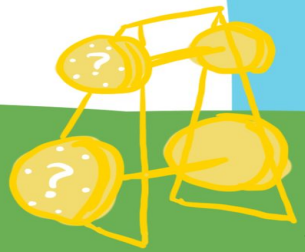
DEVOPS RESEARCH & ASSESSMENT



Measure the performance of your software development team with these 4 cool metrics!

What are the DORA metrics?

1. Deployment Frequency
2. Lead Time for Changes
3. Change Failure Rate
4. Time to Restore Service



Tekton project types

Our users run:

- Pipelines
- Triggers
- Results
- Dashboard
- Operator

We run:

- Hub

Tools:

- CLI

DORA metrics when you aren’t running a service

We don’t actually run:

- Pipelines
- Triggers
- Results
- Dashboard
- Operator
- CLI

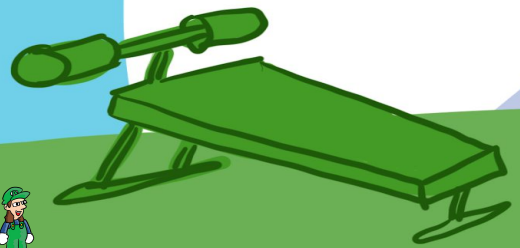
But we DO dogfood:

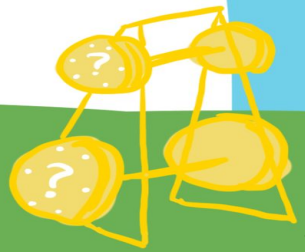
- Pipelines
- Triggers
- Dashboard
- Results

Focusing on releases

1. Deployment Frequency
2. Lead Time for Changes
3. Change Failure Rate
4. Time to Restore Service

1. Release Frequency
2. Lead Time for Changes
3. Change Failure Rate
4. Time to Release Fixes

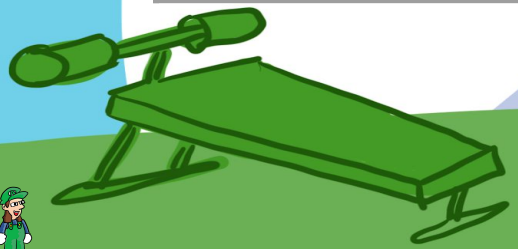


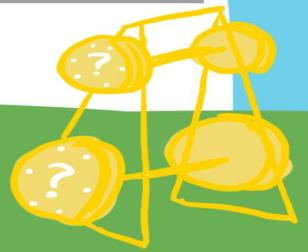


Deployment/Release Frequency

How often an organization successfully deploys to production or releases to end users

Elite	High	Medium	Low
Multiple per day	Once a day - Once per week	Once per week - Once per month	Once per month - Once every 6 months



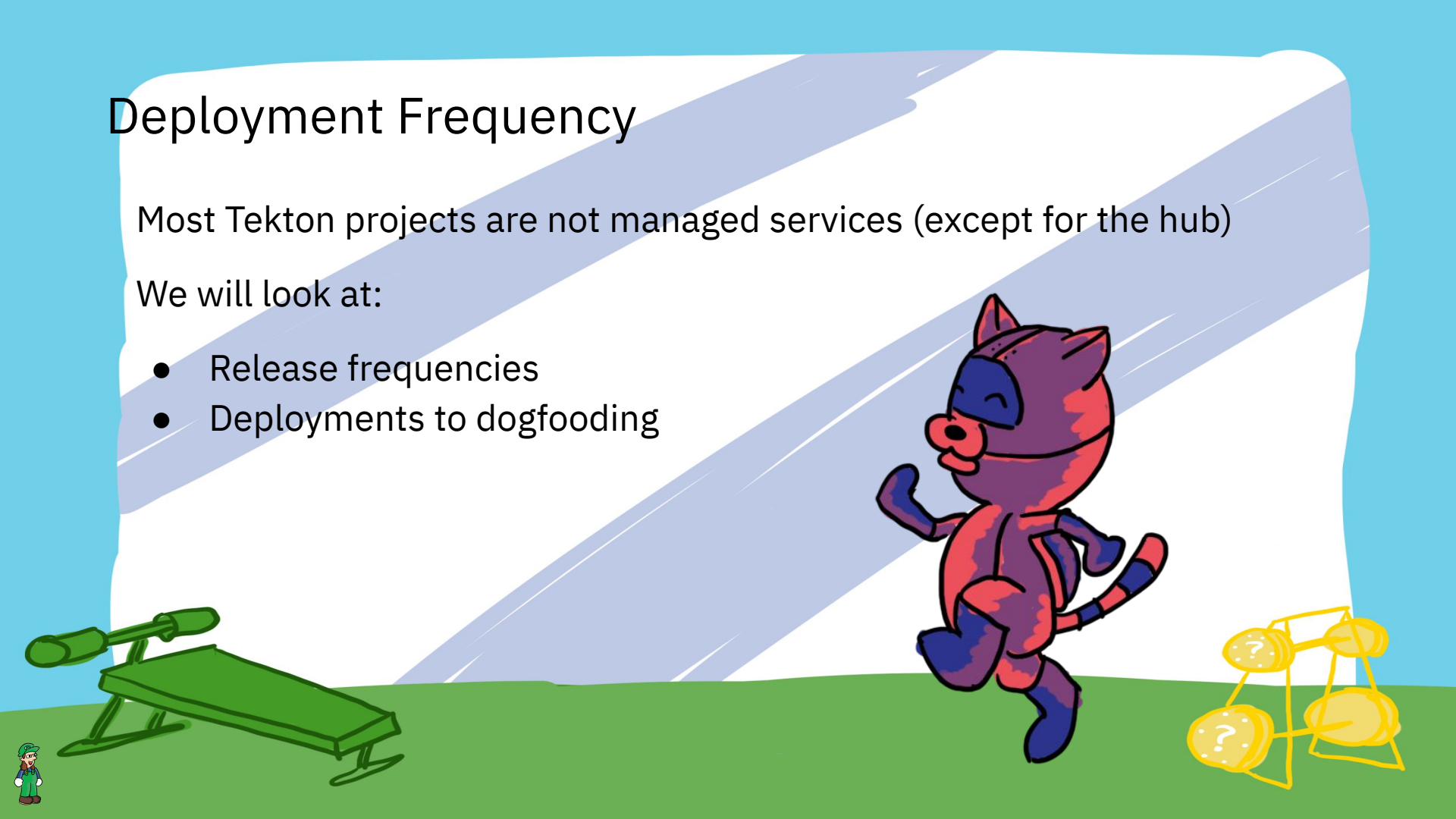


Deployment Frequency

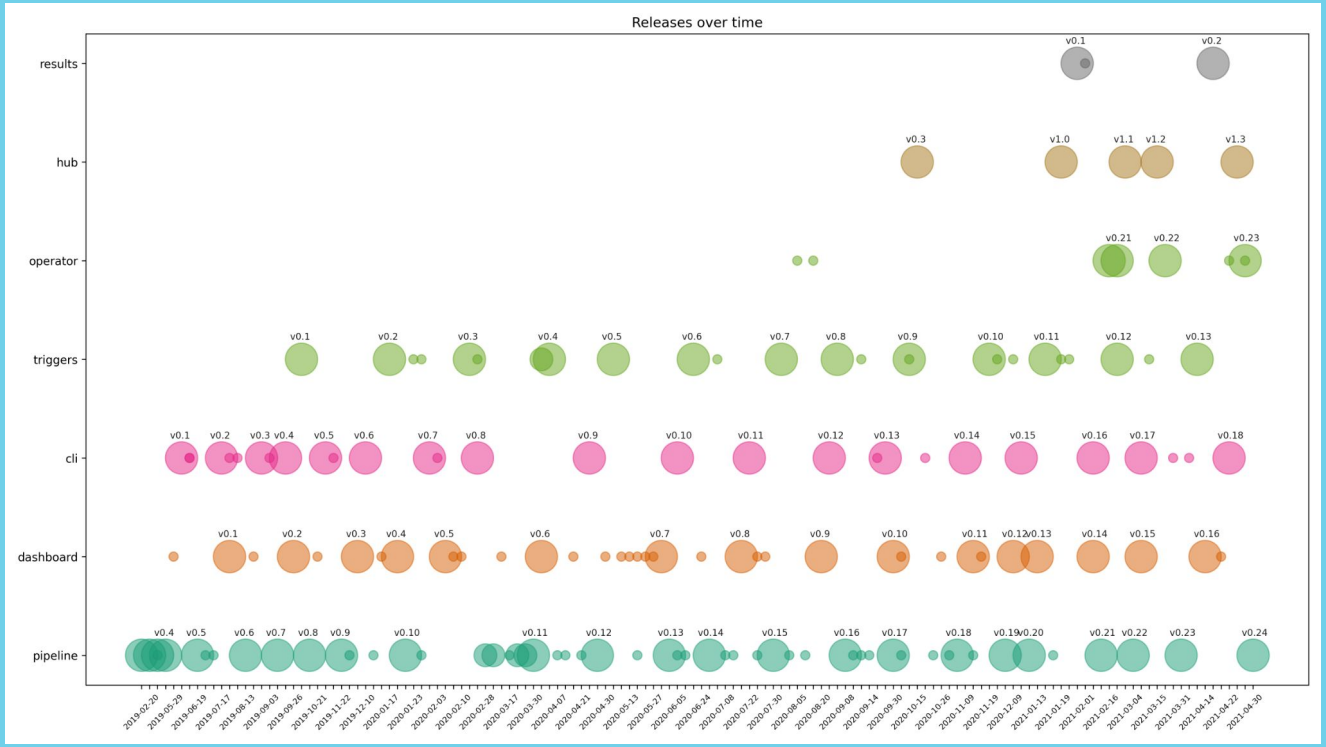
Most Tekton projects are not managed services (except for the hub)

We will look at:

- Release frequencies
- Deployments to dogfooding



Release Frequency



Release Frequency

	Average days b/w releases	Performance
Pipelines	16	Medium
Triggers	24	Medium
Results	39	Low
CLI	24	Medium
Dashboard	19	Medium
Hub	48	Low
Operator	34	Low

Nightly releases

	Average days b/w releases	Performance
Pipelines	1	High
Triggers	1	High
Results	39	Low
CLI	24	Medium
Dashboard	1	High
Hub	48	Low
Operator	1	High

Dogfood Deployments

	Average days b/w deploy	Performance
Pipelines	23	Medium
Triggers	26	Medium
Results	(deployed once)	Low
CLI		
Dashboard	28	Medium
Hub		
Operator		

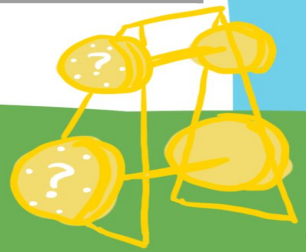
Deployment/Release Frequency

	Release	Including nightly	Dogfood deployments	Overall
Pipelines	Medium	High	Medium	Medium
Triggers	Medium	High	Medium	Medium
Results	Low	Low	Low	Low
CLI	Medium	Medium		Medium
Dashboard	Medium	High	Medium	Medium
Hub	Low	Low		Low
Operator	Low	High		Medium

Lead Time for Changes

The amount of time it takes a commit to get into production

Elite	High	Medium	Low
Less than one day	One day - One week	One week - One month	One month - 6 months



Days between releases

	Average days b/w releases	Performance
Pipelines	35	Low
Triggers	45	Low
Results	78	Low
CLI	39	Low
Dashboard	42	Low
Hub	48	Low
Operator	53	Low

Time between merge and release

	Average between merge and release	Performance
Pipelines	27	Medium
Triggers	25	Medium
Results	30	Medium
CLI	20	Medium
Dashboard	16	Medium
Hub	21	Medium
Operator	82	Low

Time between PR and release

	Average between PR opening and release	Performance
Pipelines	32	Low
Triggers	29	Medium
Results	35	Low
CLI	23	Medium
Dashboard	19	Medium
Hub	34	Low
Operator	87	Low

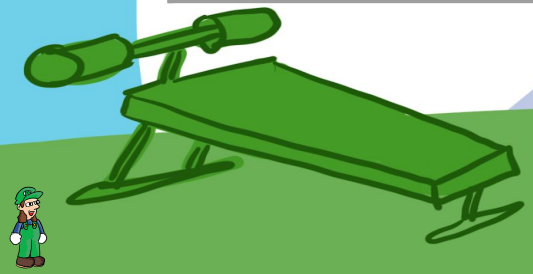
Lead Time for Changes

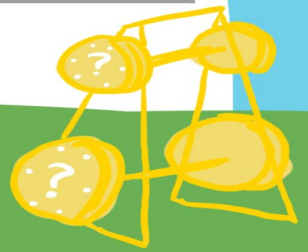
	Releases	Merge and release	PR opening and release	Overall
Pipelines	Low	Medium	Low	Low
Triggers	Low	Medium	Medium	Medium
Results	Low	Medium	Low	Low
CLI	Low	Medium	Medium	Medium
Dashboard	Low	Medium	Medium	Medium
Hub	Low	Medium	Low	Low
Operator	Low	Low	Low	Low

Change Failure Rate

The percentage of deployments causing a failure in production (degraded service requiring remediation)

Elite	High	Medium	Low
0-15%	0-15%	0-15%	46-60%

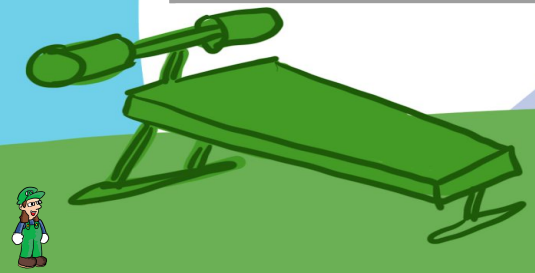


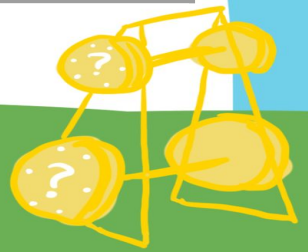


Change Failure Rate

The percentage of deployments causing a failure in production (degraded service requiring remediation)

Elite	High	Medium	Low
0-15%	0-15%	0-15%	46-60%





What’s up with the overlapping intervals?

The State of DevOps Report 2019, https://cloud.google.com/devops/

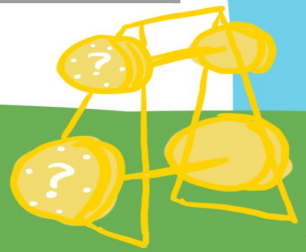
% of releases that need patches

	% of releases that have patch releases	Performance
Pipelines	54.17%	Low
Triggers	61.54%	Low
Results	50.00%	Low
CLI	44.44%	Medium
Dashboard	58.82%	Low
Hub	0	Elite
Operator	33.33%	Medium

Time to Restore Service

How long it takes an organization to recover from a failure in production

Elite	High	Medium	Low
Less than one hour	Less than one day	Less than one day	One week - One month



Days between patch releases

	Days between patch releases	Performance
Pipelines	7	Low
Triggers	7	Low
Results	1	High
CLI	9	Low
Dashboard	12	Low
Hub		
Operator	16	Low

Tekton DORA performance

	Release frequency	Lead time for changes	Change failure rate	Time to restore	**Overall**
Pipelines	Medium	Low	Low	Low	**Low**
Triggers	Medium	Medium	Low	Low	**Medium**
Results	Low	Low	Low	High	**Low**
CLI	Medium	Medium	Medium	Low	**Medium**
Dashboard	Medium	Medium	Low	Low	**Medium**
Hub	Low	Low	Elite		**Medium**
Operator	Medium	Low	Medium	Low	**Medium**

Tekton’s performance overall

	Overall
Pipelines	Low
Triggers	Medium
Results	Low
CLI	Medium
Dashboard	Medium
Hub	Medium
Operator	Medium

How well do the elite/high/medium/low categorizations correlate here?

- Should we use the same values when we’re not talking about a running service?
- What about the open source aspect? A bit different from a team within a company focusing on one project
- Time to restore: harder to fix a release than rollback a deployment

Tekton’s performance overall

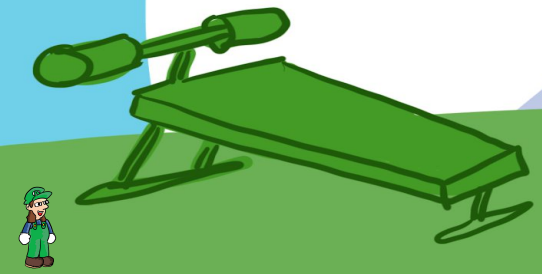
	Overall
Tekton	Medium

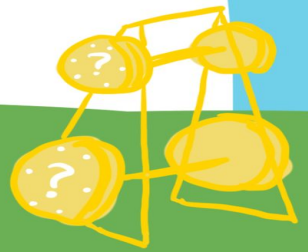
<u>What could push us into high/elite?</u>

More releases would improve:

- Release frequency (more releases)
- Lead time (less time b/w releases)
- Change failure rate (% with errors would be lower)

CD Best Practices



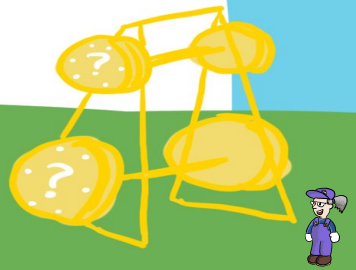


CD Best Practices

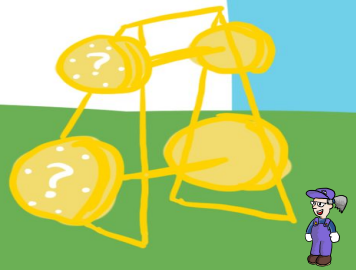
	Branch. strategy	Tests & Coverage	Periodic / Matrix	Results / Flakes	Nightly Releases	Full Releases	Continuous Deployment
Pipelines	✓	✓	●	●	✓	✓	✓
Triggers	✓	✓	●	●	✓	✓	✓
Results	✓	●	✗	●	✗	✓	✗
CLI	✓	✓	●	●	✓	✓	●
Dashboard	✓	✓	●	●	✓	✓	✓
Hub	✓	●	✗	●	✗	●	✗
Operator	✓	✓	●	●	✓	✓	●

Conclusion: Tekton is doing pretty well!

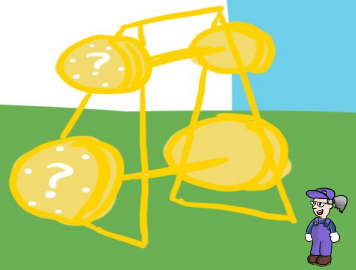
DORA + Best Practices are a really useful glimpse into how well you’re doing CD



Deployment frequency
is a great place to start



How does your project measure up?
Are you an elite performer?



Thank you!

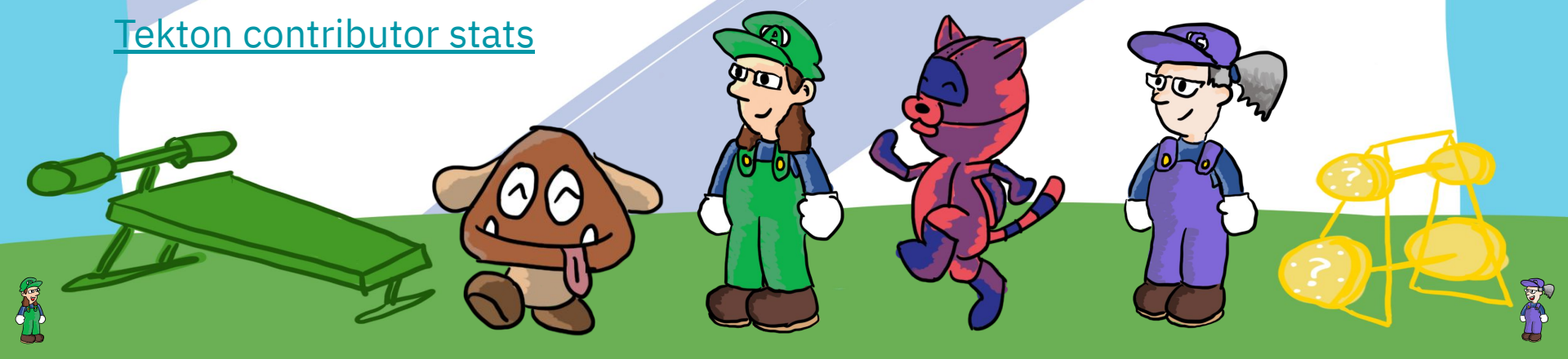
tekton.dev

Using the four keys to measure your devops performance

Kubernetes resource management (KRM)

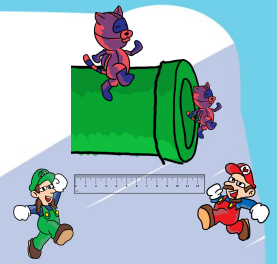
Kubernetes custom resources (CRD)s

Tekton contributor stats



Continuous Delivery Inception

Who measures the CD of the CD system?



- Tekton is a system for Continuous Delivery
- But we also continuously deliver Tekton itself!

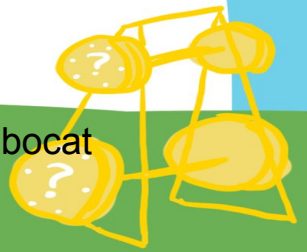


Image: something about eating our own dogfood, maybe robocat measuring the robocat itself...

Diagram showing how these projects relate to each other

Pipelines

Triggers

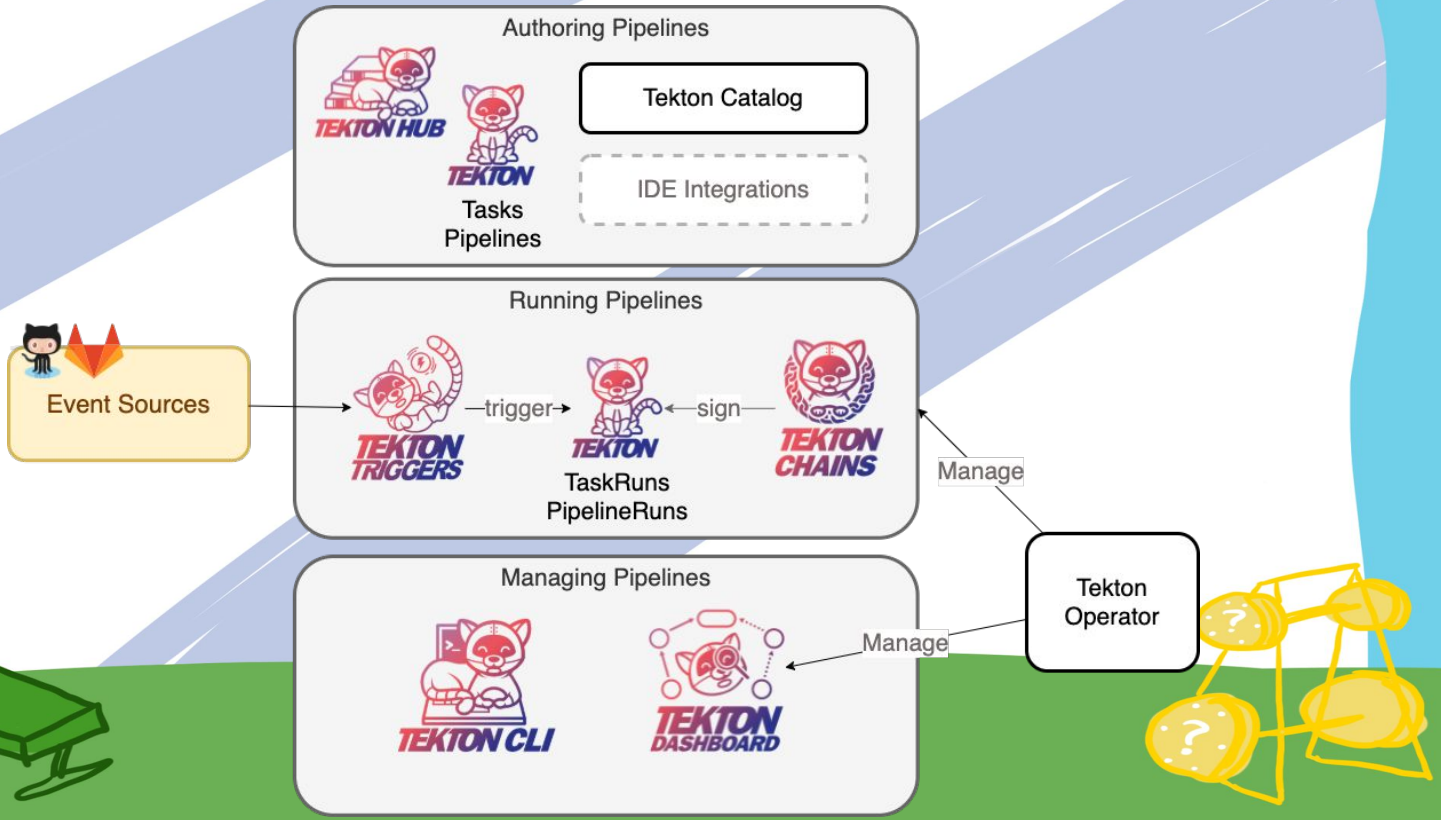
Results

CLI

Dashboard

Hub

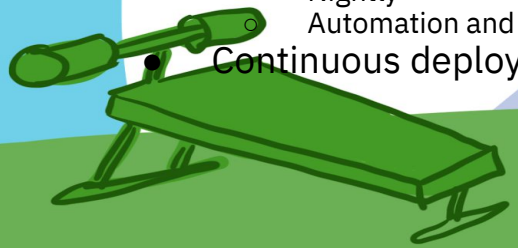
Operator

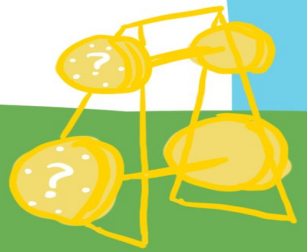


CD Best Practices

This section is less about us saying “these are absolutely the best practices” and more of “here are some lenses to look at the project through” (I think?)

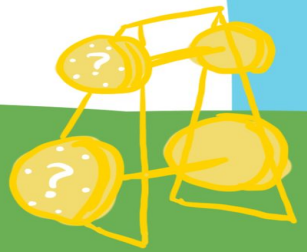
- Version control
  - Trunk based
  - Branching strategies
- Linting
- Continuous Testing
  - Fastest tests first
  - Test coverage
  - Periodic Tests
  - Flakey tests
  - Test matrices
- Releases
  - Nightly
  - Automation and triggering
- Continuous deployment





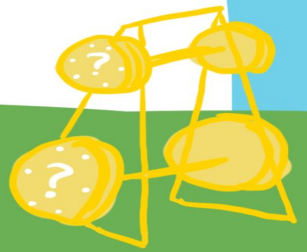
Version control

- Trunk based
- Branching strategies



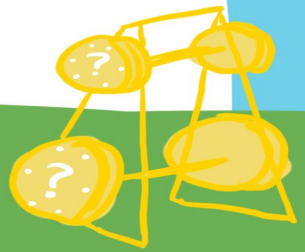
Version control

TODO: how projects stack up against best practices

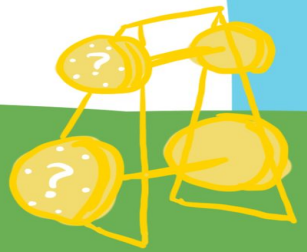


Continuous Testing

- Fastest tests first
- Test coverage
- Periodic Tests
- Flakey tests
- Test matrices

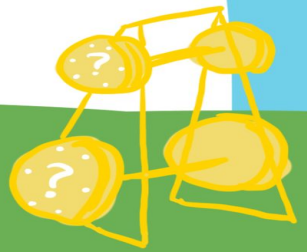


Continuous Testing

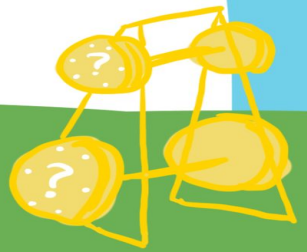


Releases

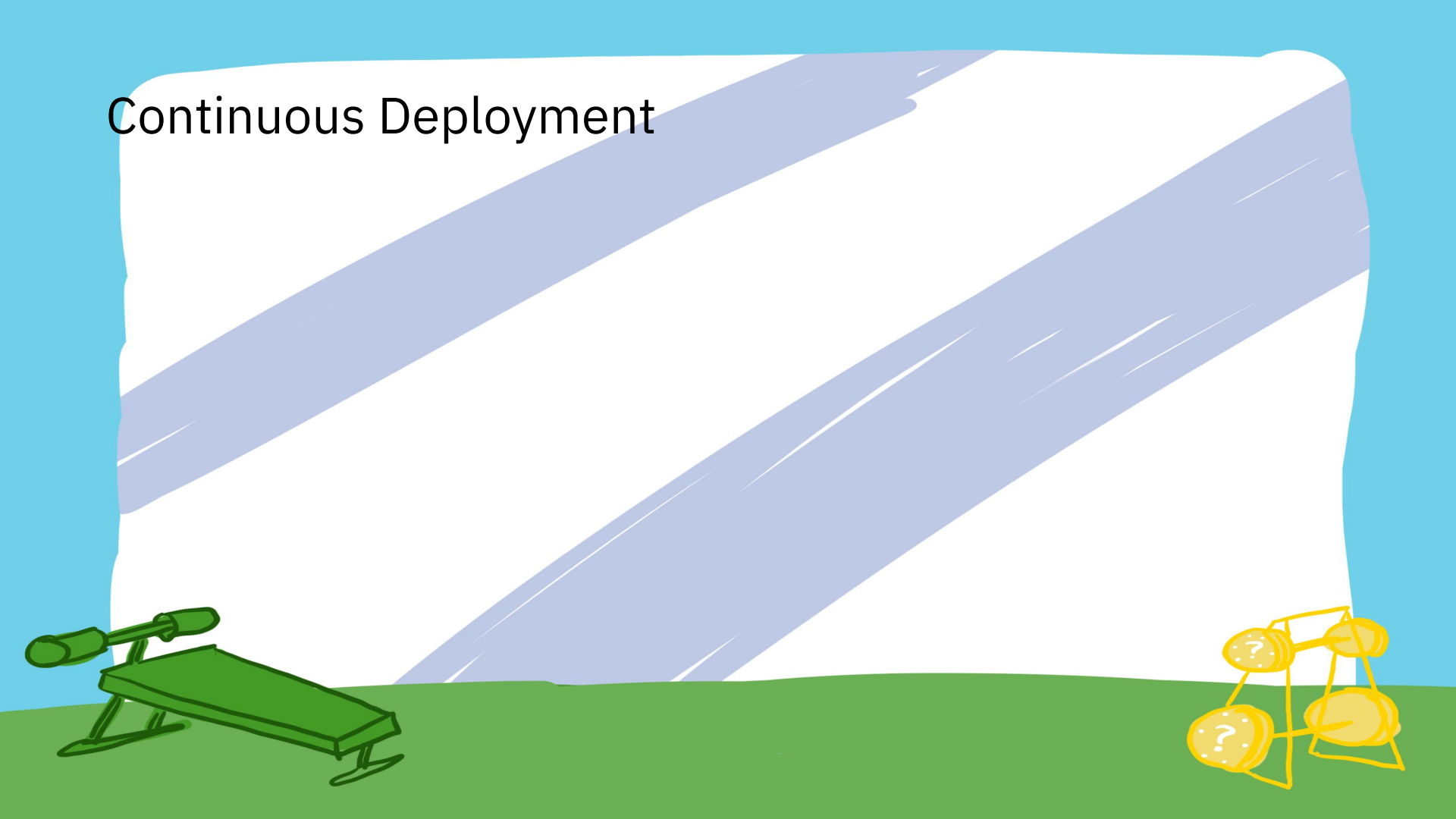
- Nightly
- Automation and triggering



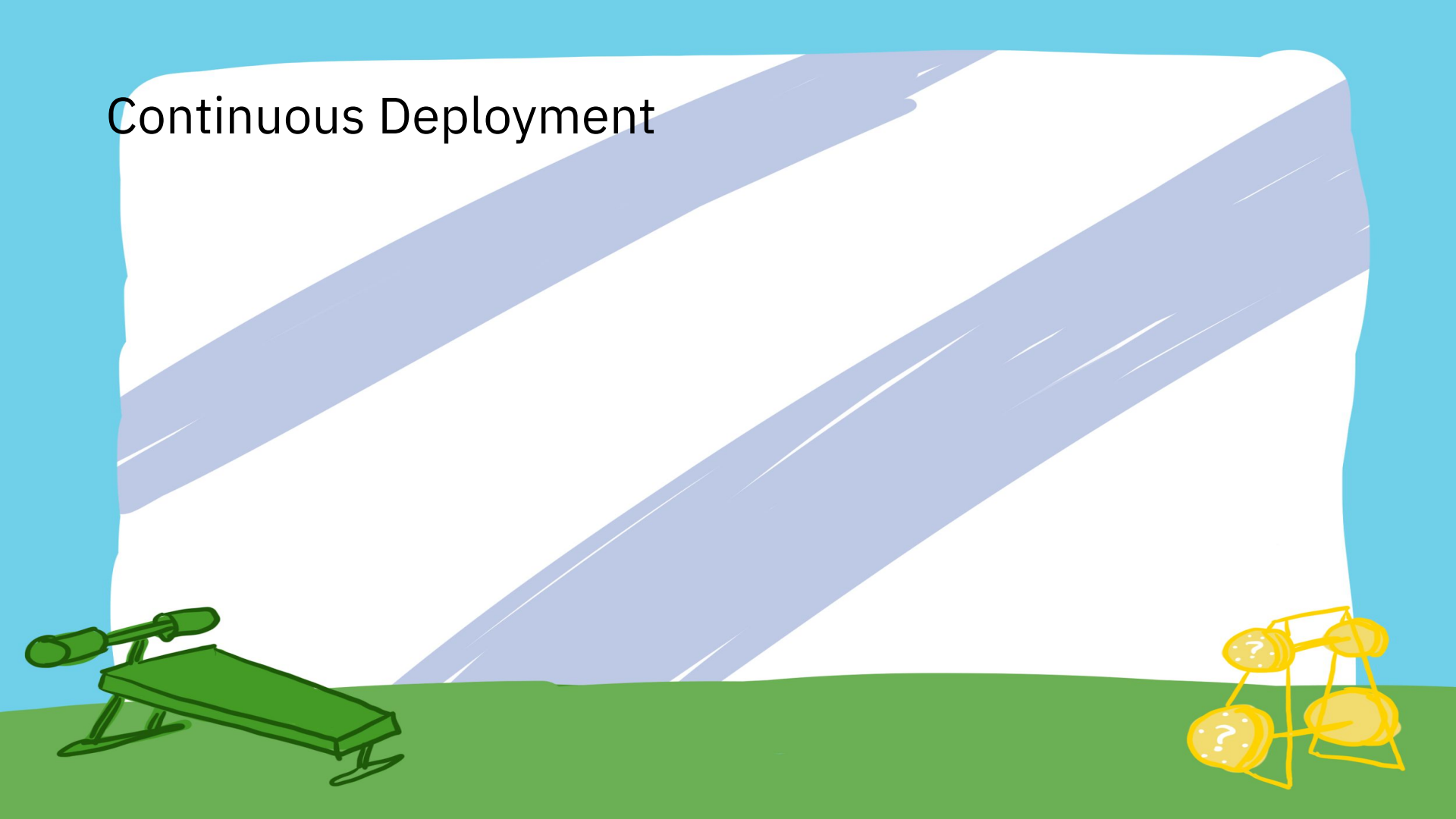
Releases



Continuous Deployment



Continuous Deployment



Elite	High	Medium	Low
Less than one hour	Less than one day	Less than one day	One week - One month

Example Table?

	Days between minor releases	Performance
Pipelines	7	Low
Triggers	7	Low
Results	1	High
CLI	9	Low
Dashboard	12	Low
Hub	0	Elite
Operator	16	Low