Continuous Integration Testing

Fully test your microservices application, early and often

davidstanke@

Hi, I’m Dave.

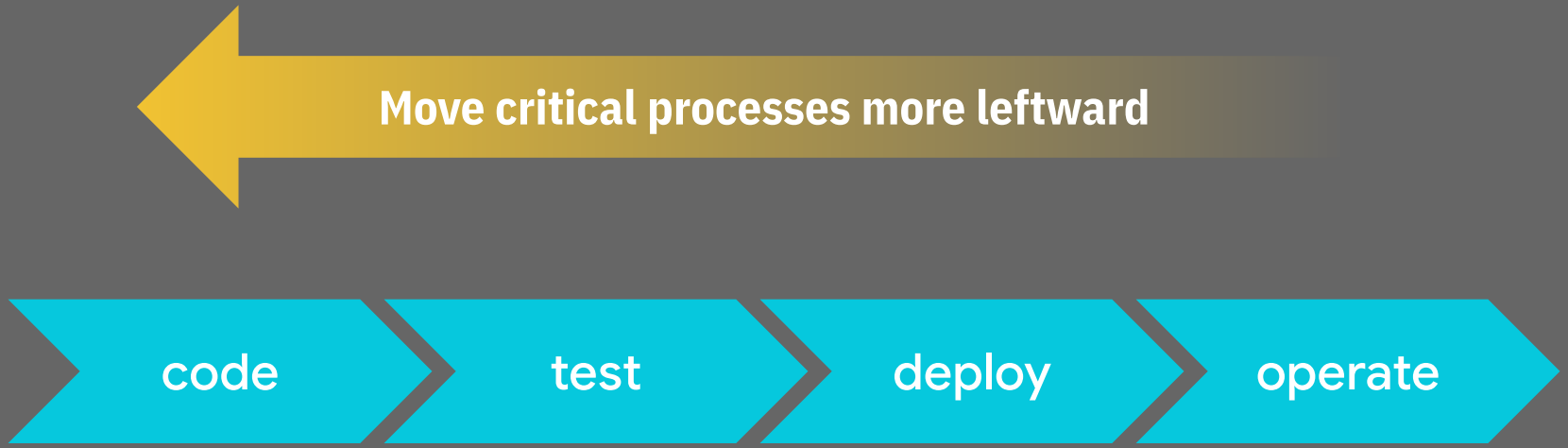
DevOps Advocate @ Google

davidstanke

Continuous Integration Testing

1. Integration testing is hard
2. We can do it anyway
3. We *should* do it anyway
4. Some ways to do it
    a. Kubernetes / microservices example

Shift Left


Move critical processes more leftward
code
test
deploy
operate

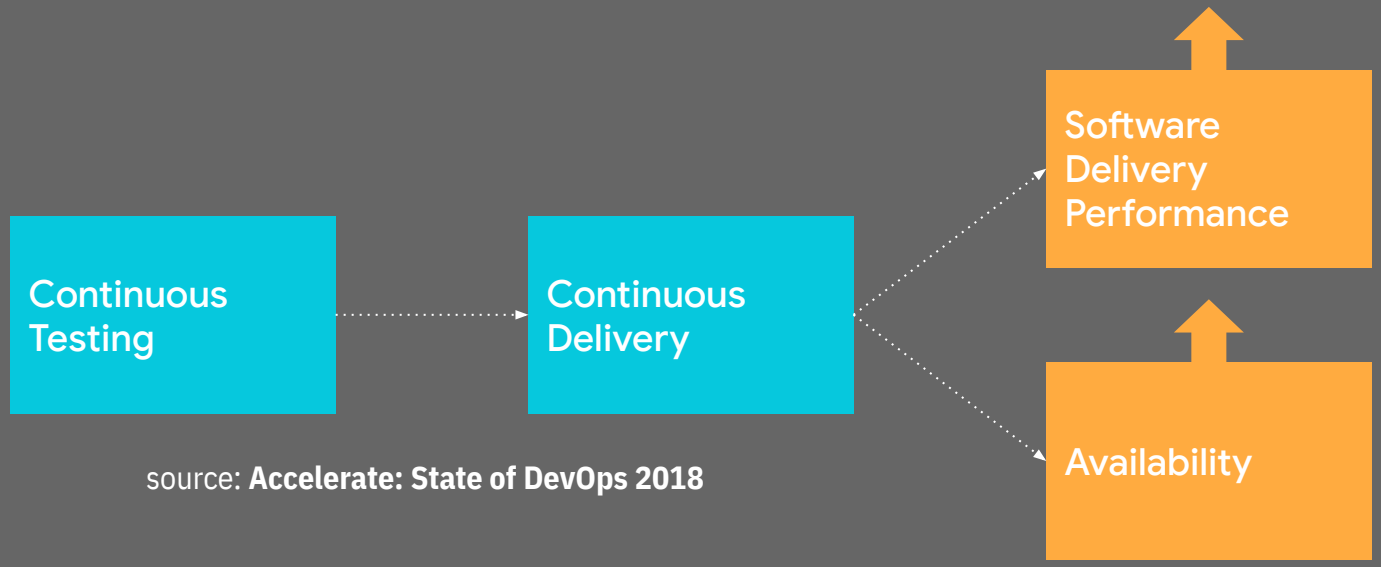
Shift all the things. Shift them left.

Merge

Security

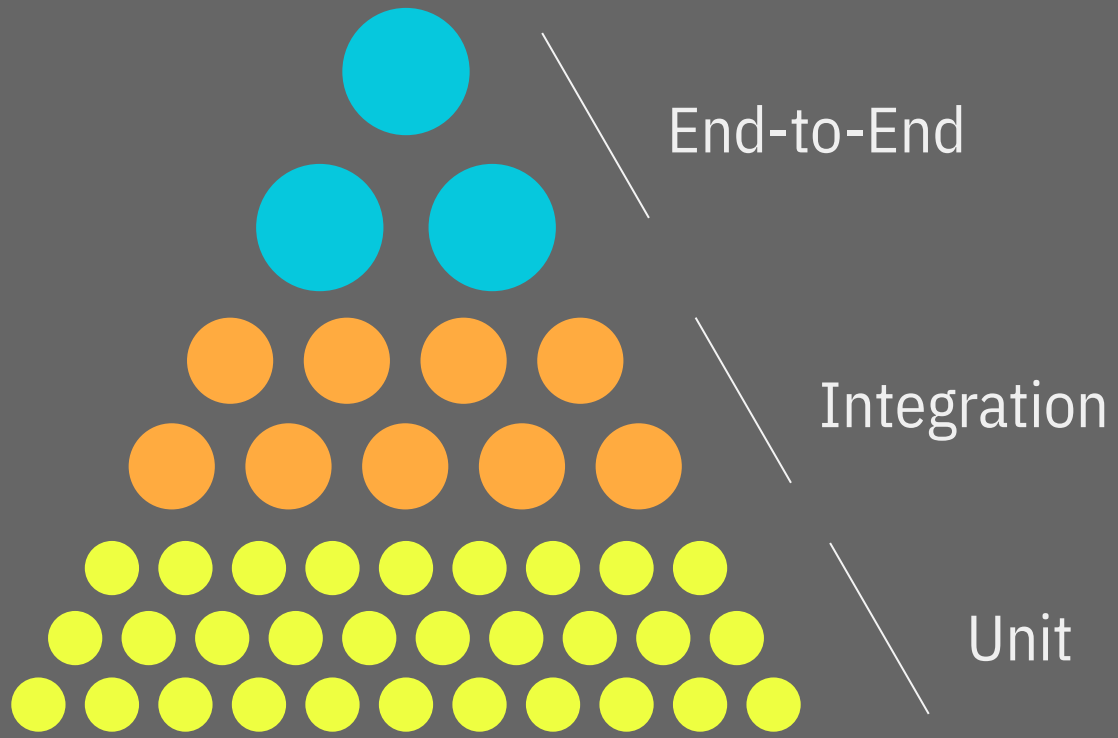
Testing

Shifting left is good for your business

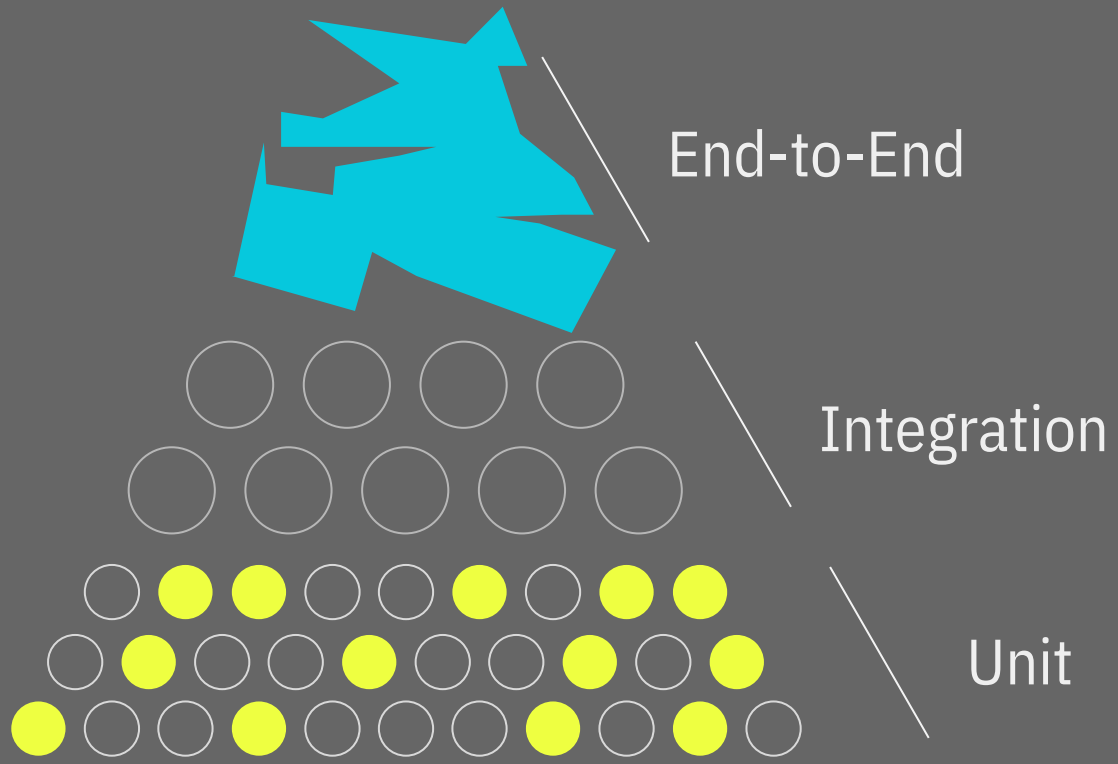


source: **Accelerate: State of DevOps 2018**

Ideal test pyramid



Actual test pyramid



Integration testing is hard

Production runtimes != developer machines

Cost

Speed

Side effects

Why bother?

API contracts / semver aren’t enough

Emergent properties of composed systems

Users experience an *application*

Don’t bother?

Test in Prod?!?

Don’t bother?

Test in Prod!

And before prod.

How bad could it be?

No, seriously...
how bad *could* it be?

We can still do integration tests

They’re hard, but (potentially) worth it

Make informed risk analysis

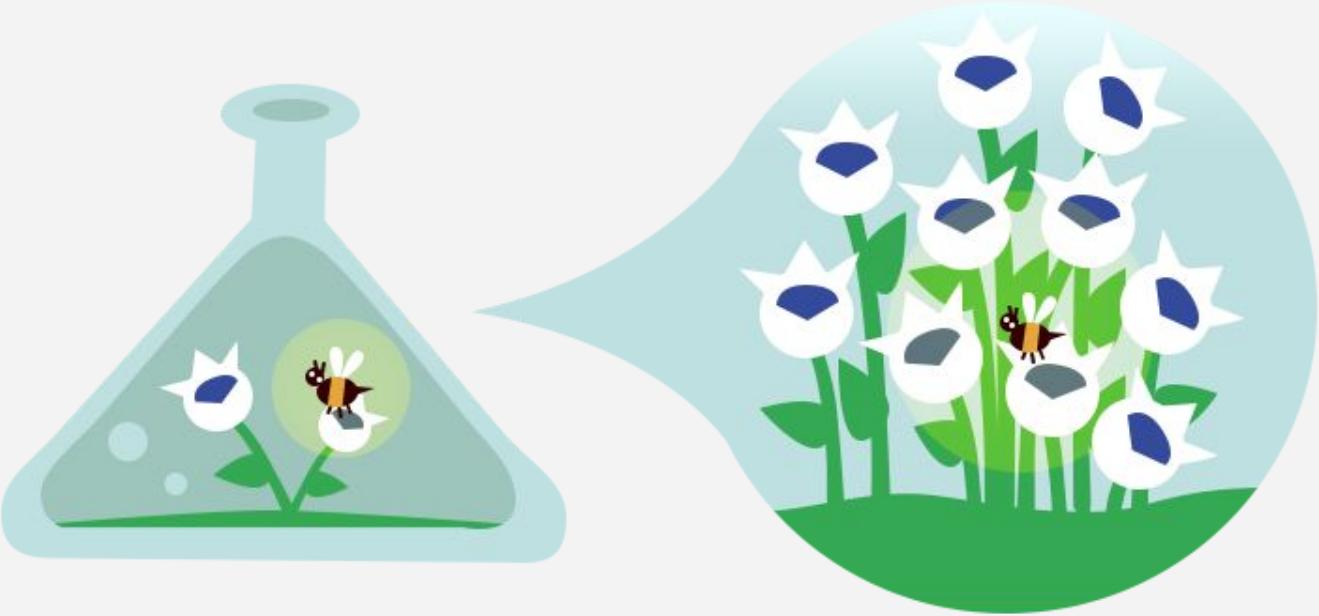
Continuously optimize

What are we optimizing for?

- Fidelity
- Isolation
- Scalability
- Speed
- Ephemerality
- Cost

Integration Testing Goal:

Fidelity



Illustrations by TRACEY LAGUERRE

Integration Testing Goal:

Isolation

Illustrations by TRACEY LAGUERRE

Integration Testing Goal:

Scalability

Illustrations by TRACEY LAGUERRE

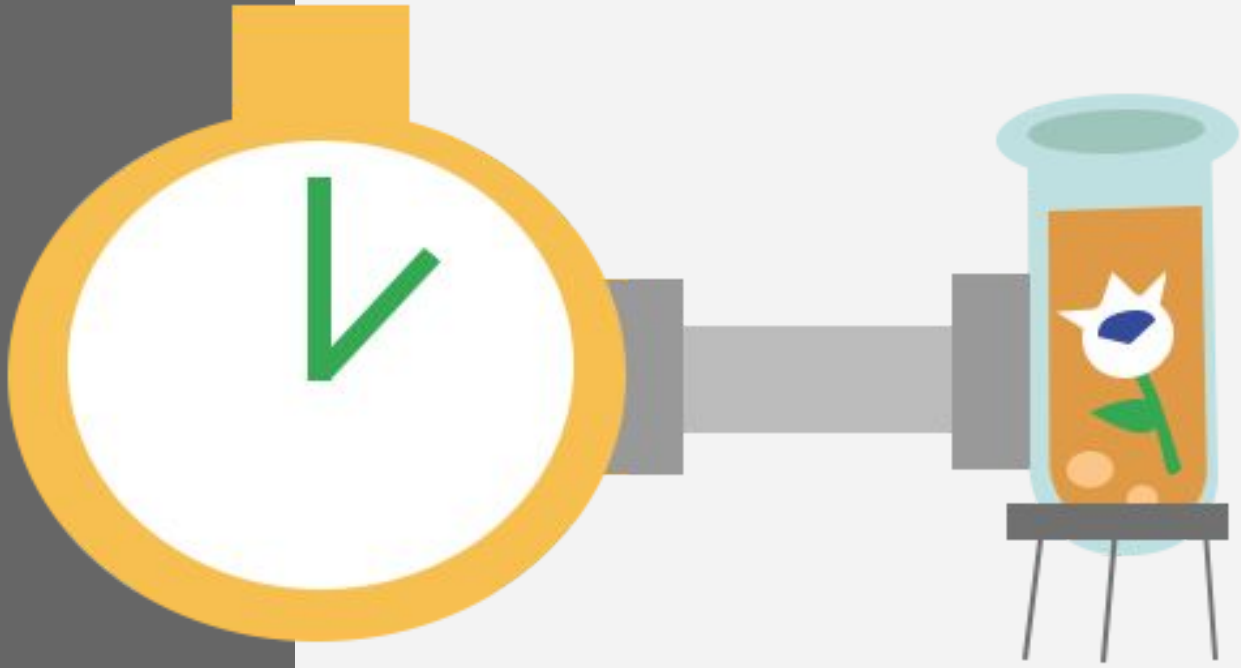
Integration Testing Goal:

Speed

Illustrations by TRACEY LAGUERRE

Integration Testing Goal:

Ephemerality



Illustrations by TRACEY LAGUERRE

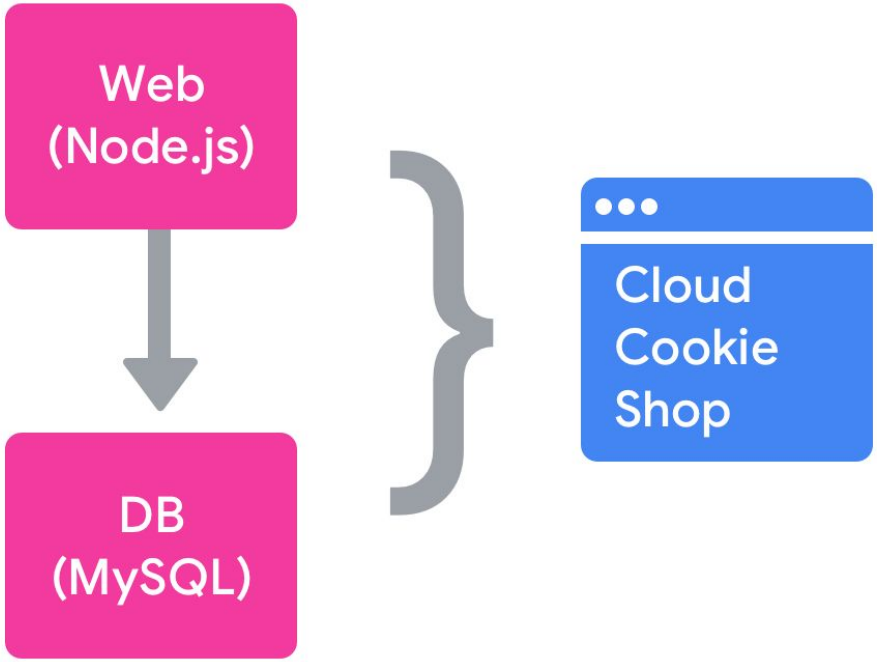
Integration Testing Goal:

Cost

Illustrations by TRACEY LAGUERRE

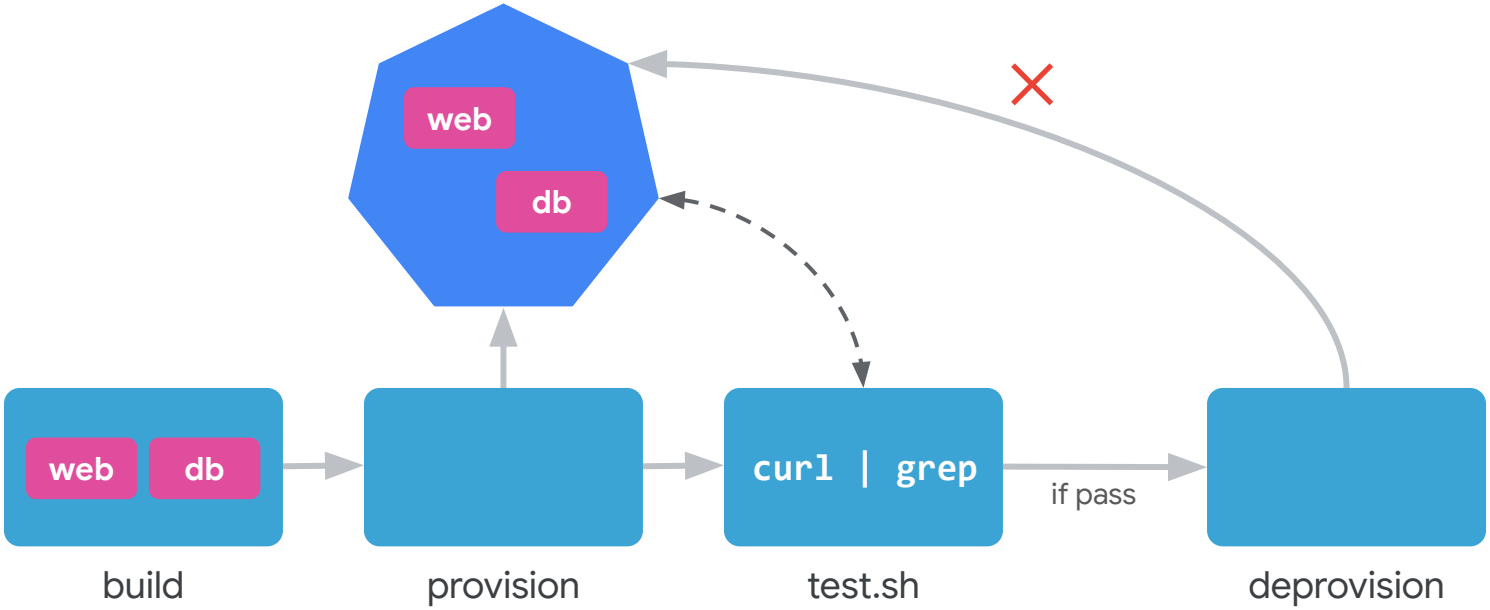
An example

CI pipeline with integration tests

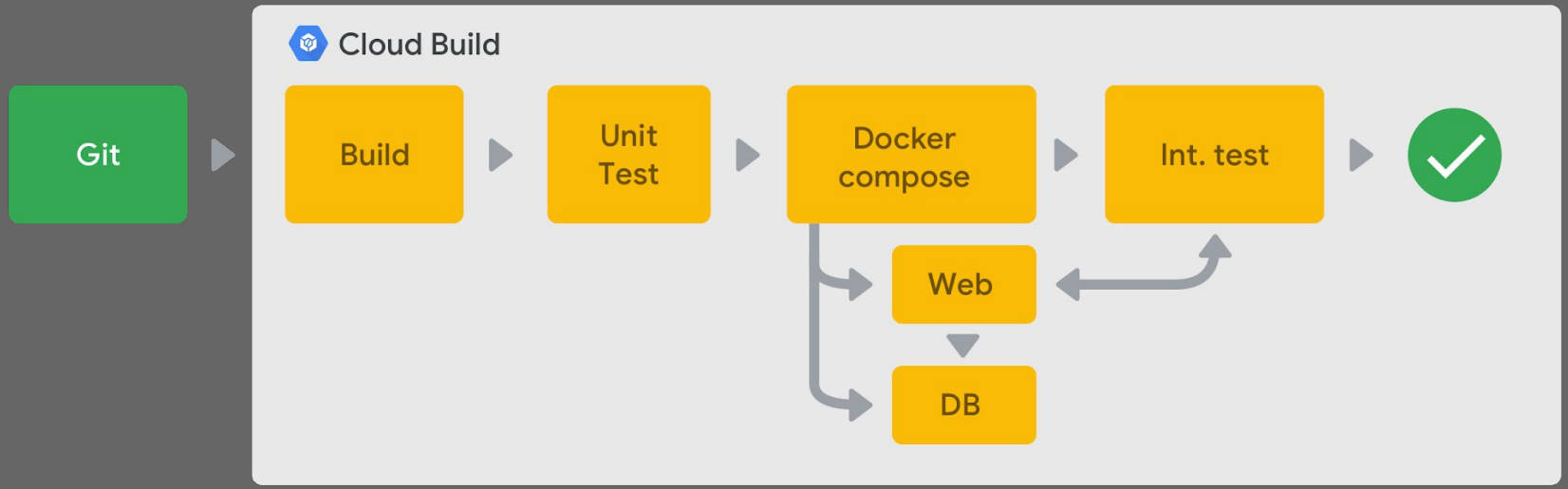


/GoogleCloudPlatform
/cloudbuild-integration-testing

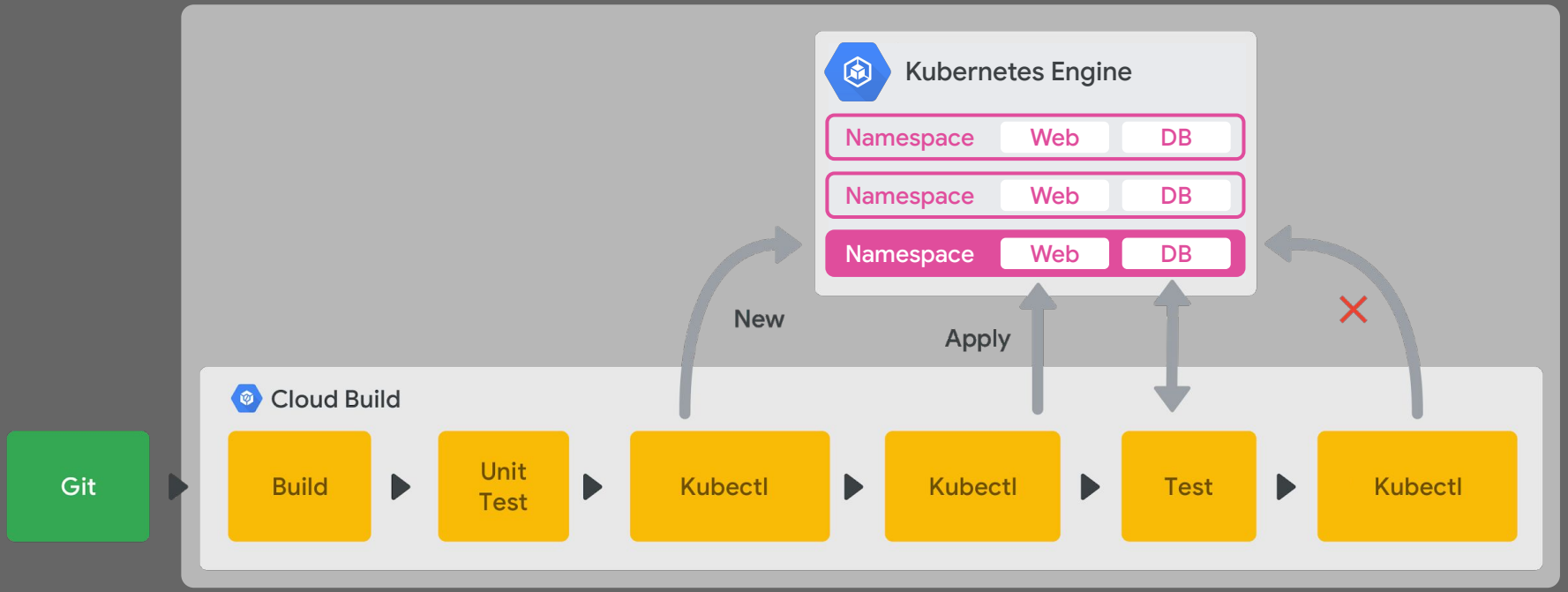
Integration test flow



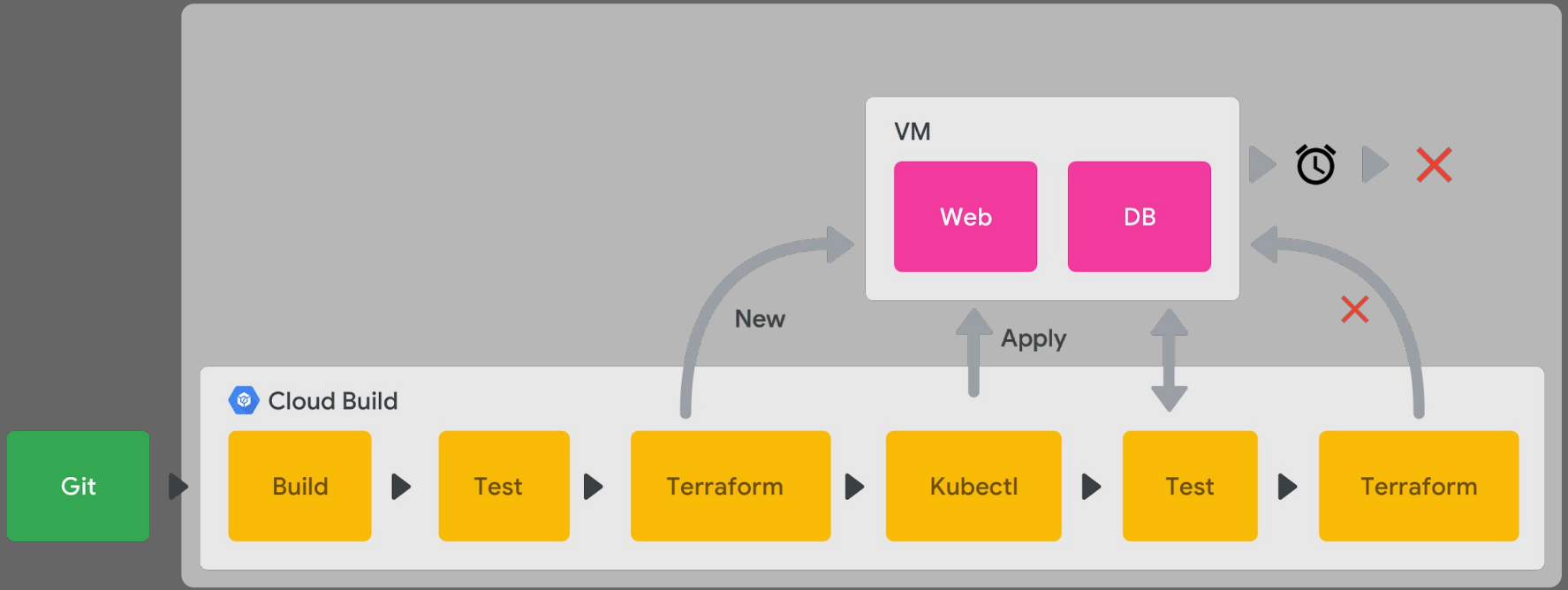
Technique 1: Docker Compose



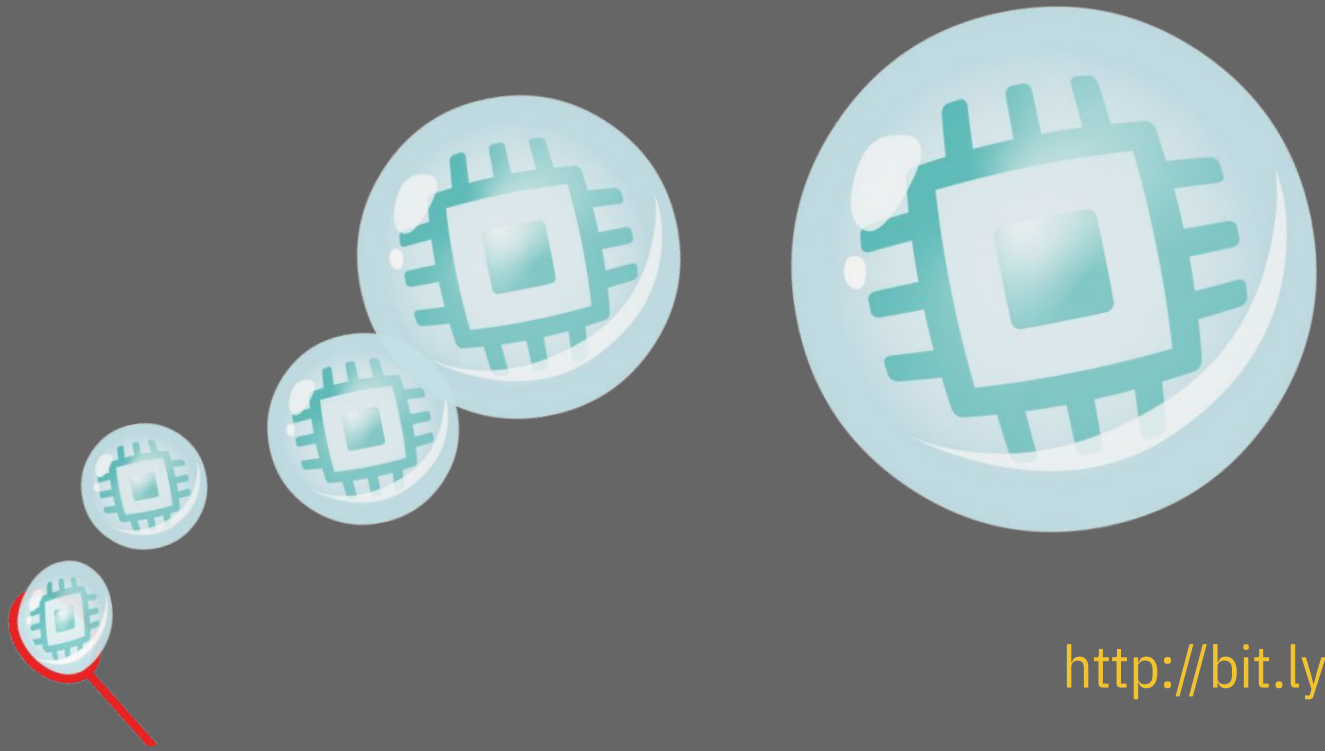
Technique 2: K8s Staging Cluster



Technique 3: Self-destructing VM

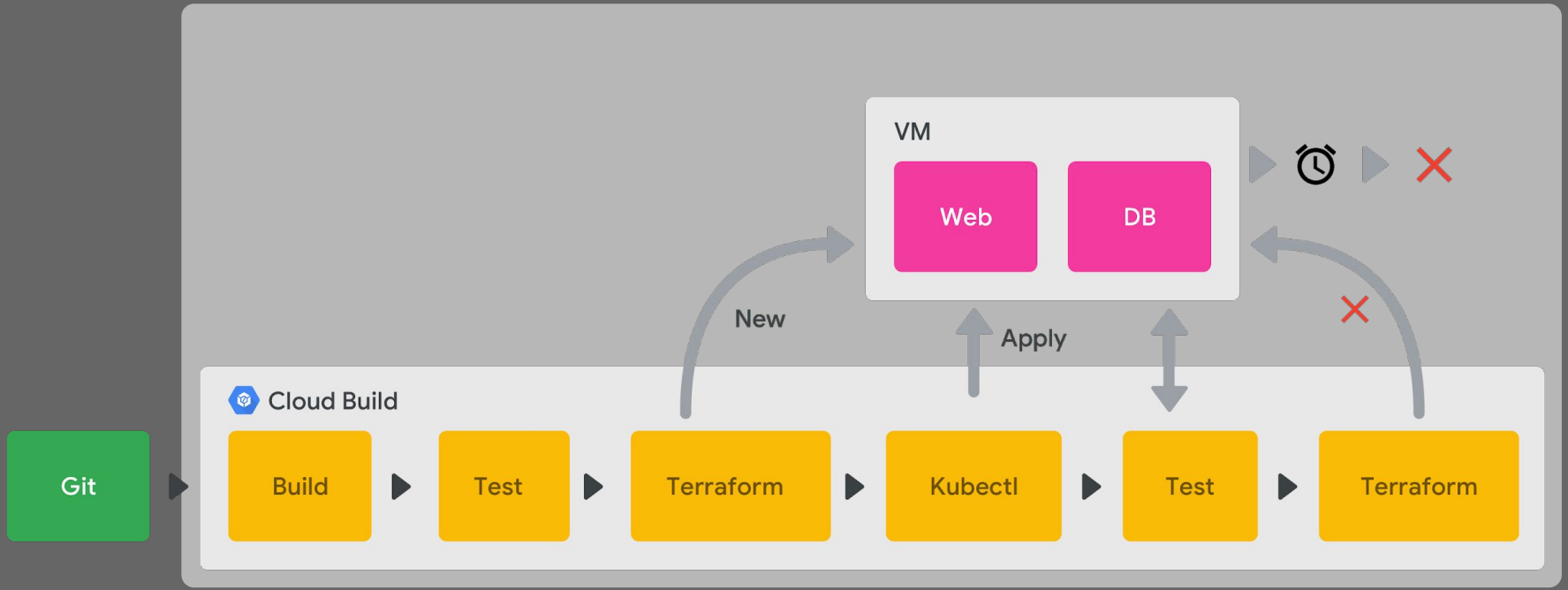


Self-Destructing VM?

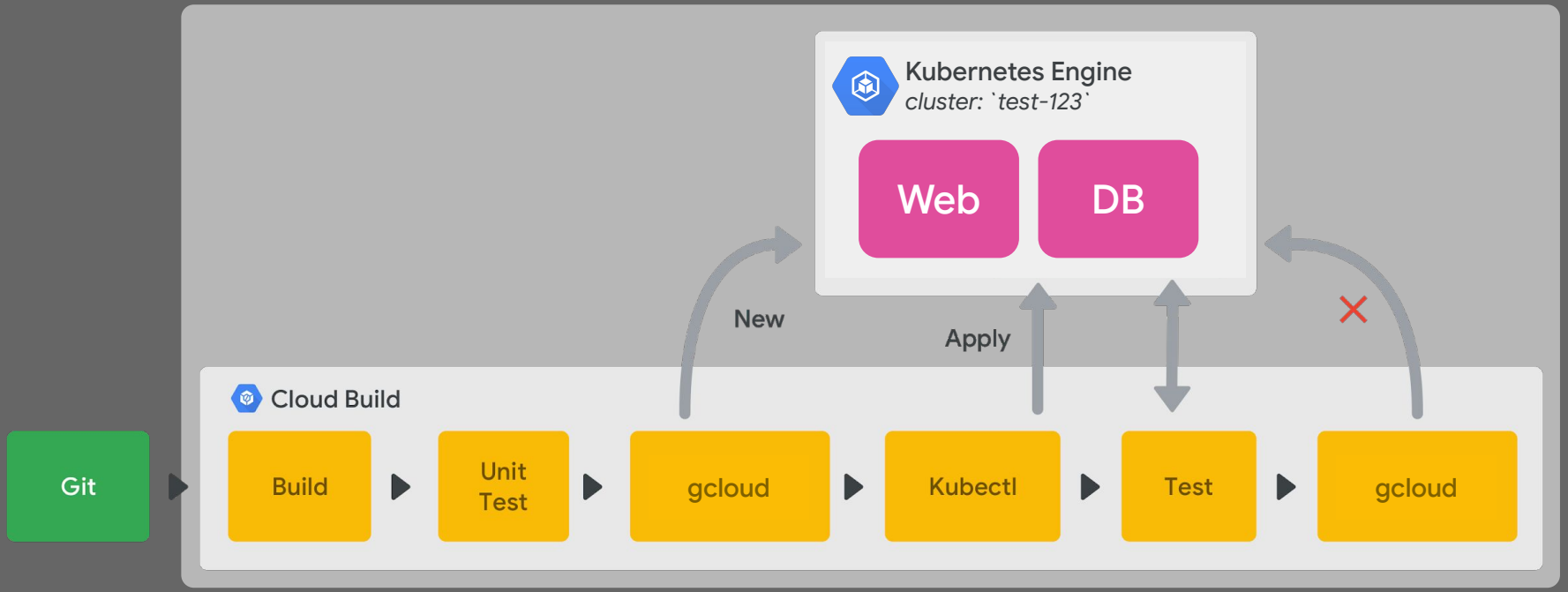


http://bit.ly/self-destructing-vm

Technique 3: Self-destructing VM!



Technique 4: K8s cluster per test



Integration testing techniques

Method	Fidelity	Isolation	Scalability	Ephemerality	Speed	Cost
docker-compose	🔴 low	🟢 highest	🟢 highest	🟡 absolute	🟢 fastest	🟢 low
Shared “staging” K8s	🟢 high	🟡 medium	🟢 high	🟢 variable	🟢 fast	🟢 low
Single-node “k8s”	🟡 medium-high	🟢 high	🟢 high	🟢 variable	🟡 medium	🟡 medium
K8s per test	🟢 highest	🟢 high	?	🟢 variable	🔴 slow	🔴 high

Integration testing techniques

Method	Fidelity	Isolation	Scalability	Ephemerality	Speed	Cost
docker-compose	low	highest	highest	absolute	fastest	low
Shared “staging” K8s	high	medium	high	variable	fast	low
Single-node “k8s”	medium-high	high	high	variable	medium	medium
K8s per test	highest	high	?	variable	slow	high

But Dave, what about...

Lots of services?

Databases?

Test data?

Yes we can (still) do integration testing

There are **trade offs** between different ways of doing it

It's all about **risk tolerance** vs. reward of speed/cost/etc.

You have to **find your own sweet spot**. (When you do, tell me about it!)

davidstanke