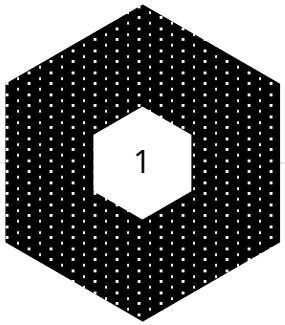
Microservices from the trenches

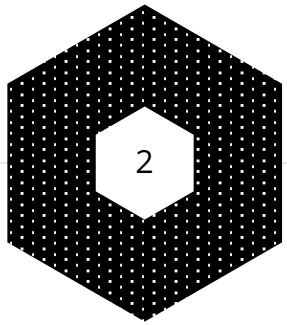
CHRISTIAN BARRA @ PyCon.DE 2018

THE AGENDA

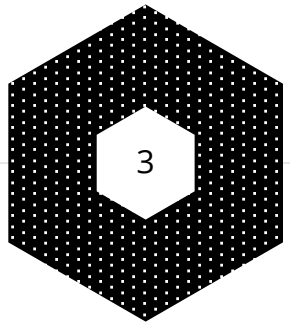
START



WHAT ARE MICROSERVICES

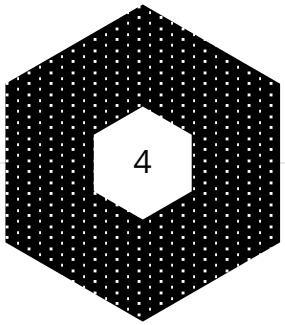


WHY YOU NEED MICROSERVICES

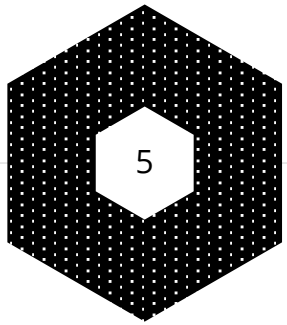


WHAT YOU NEED TO MAKE THEM WORK

THE AGENDA



UNSOLICITED ADVICE



QUESTIONS

END

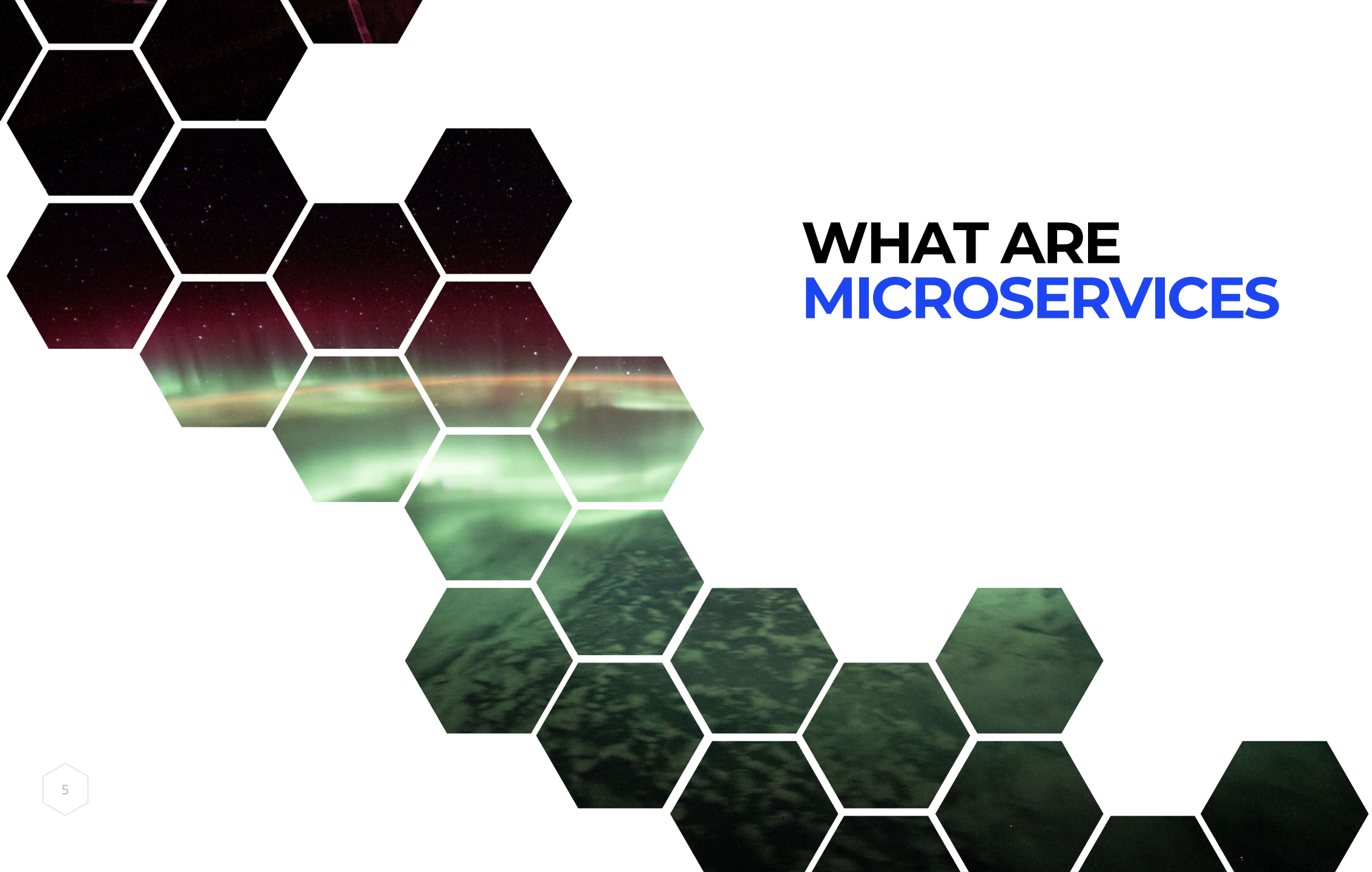
MADE.COM

WE HAVE +50 SERVICES

+102 AWS EC2 INSTANCES

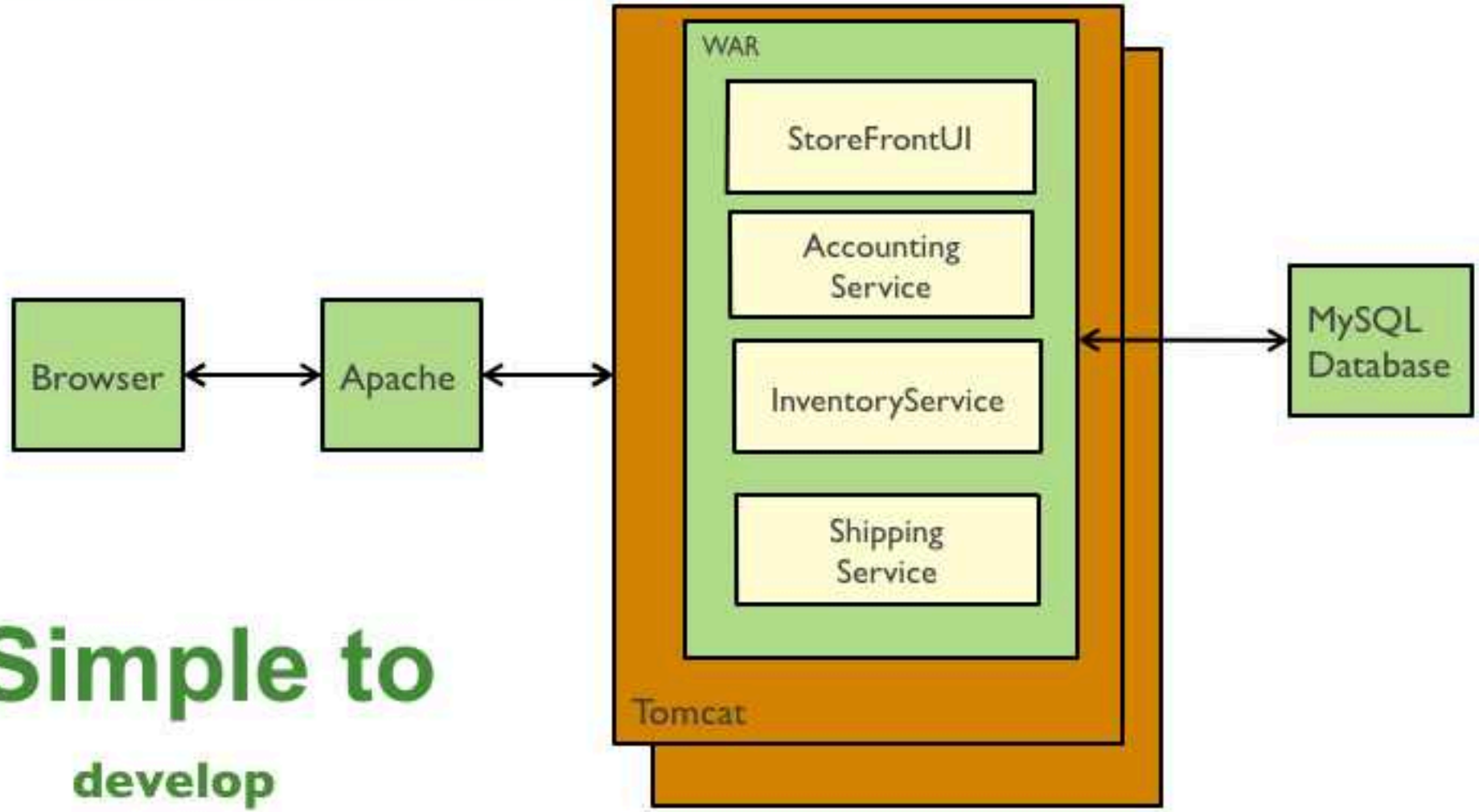
3 ENVS (TEST, UAT AND PROD)

AROUND 15 DAILY DEPLOYMENTS TO PROD



WHAT ARE MICROSERVICES

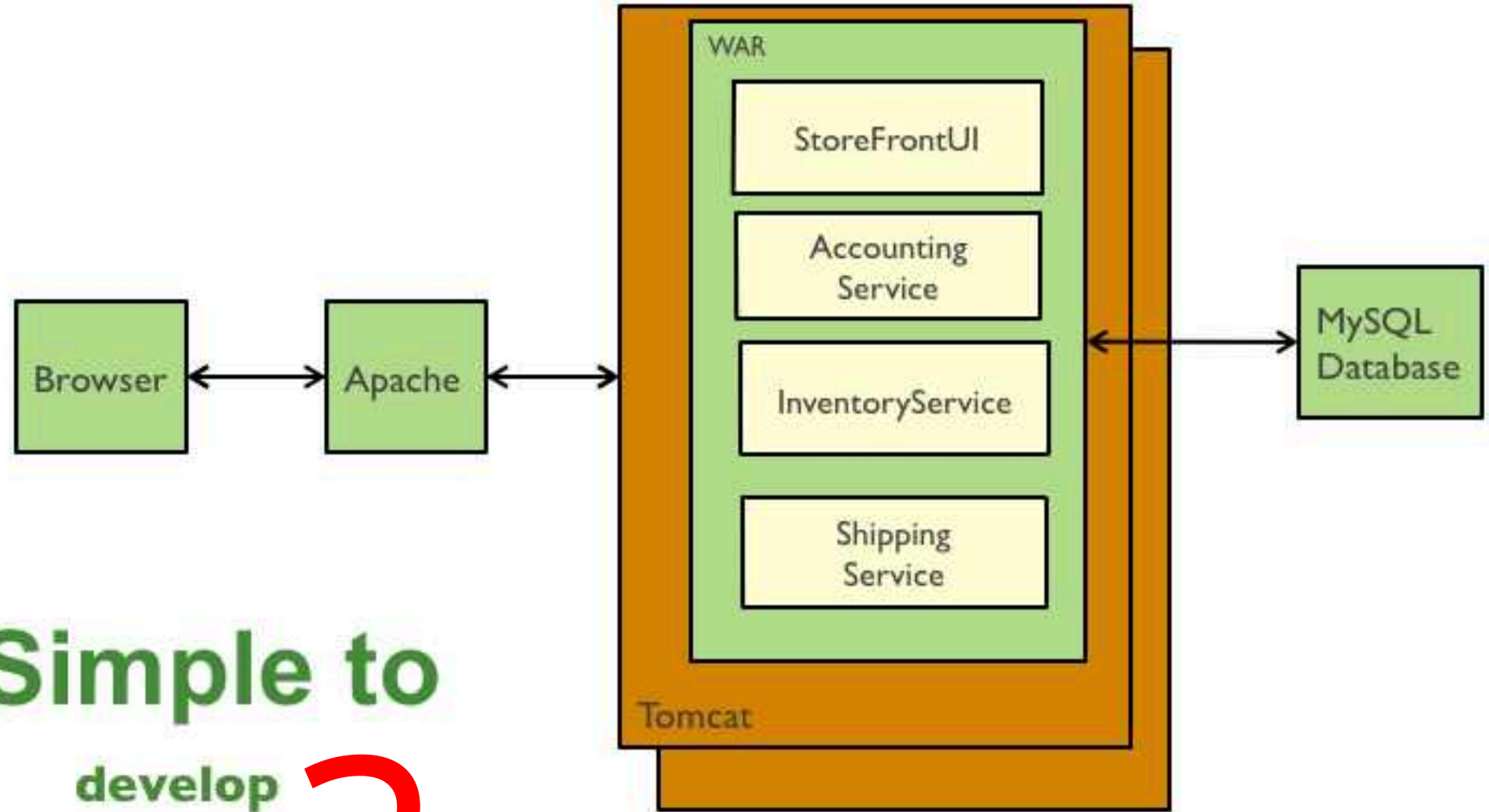
Traditional web application architecture



Simple to

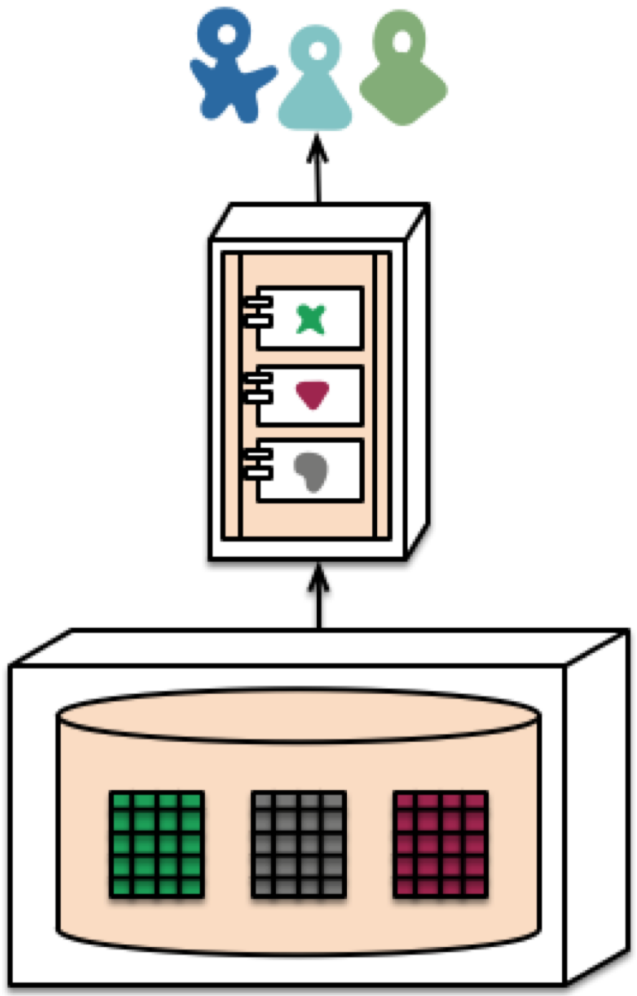
develop
test
deploy
scale

Traditional web application architecture

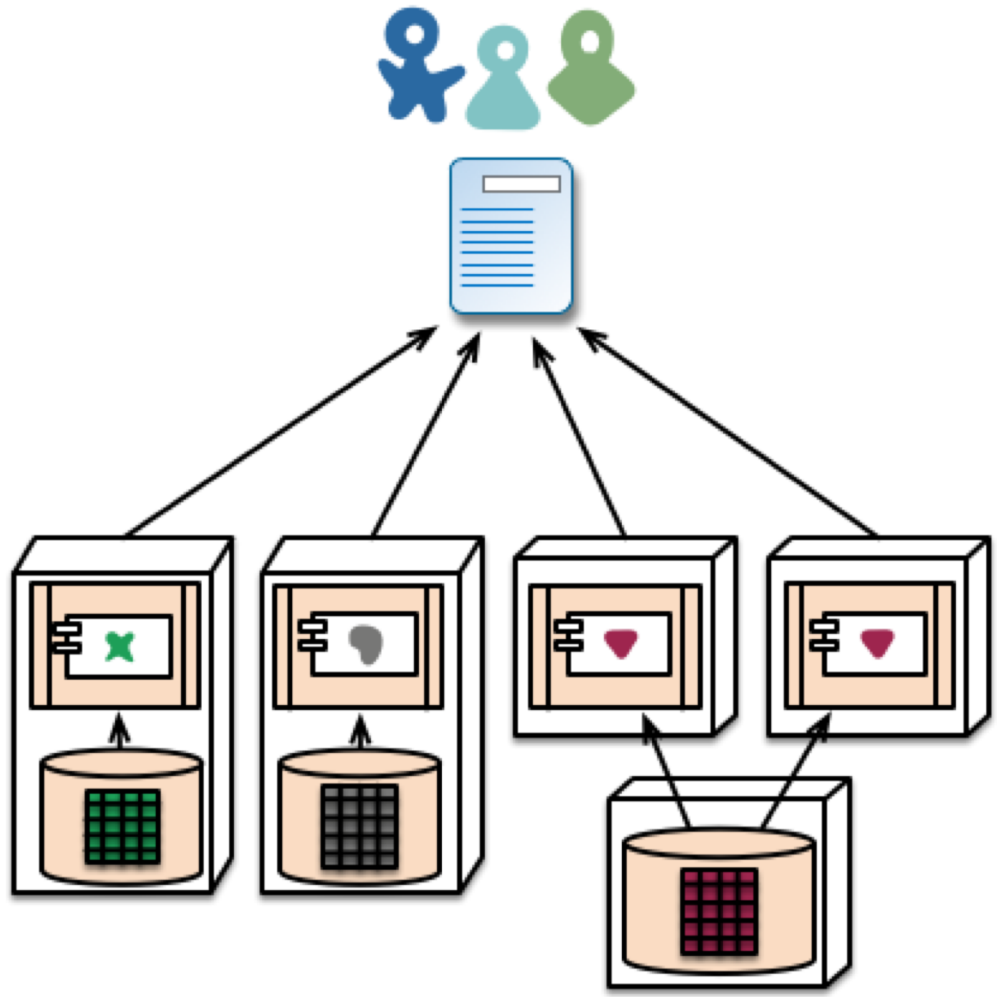


Simple to

develop
test
deploy
scale
?



monolith - single database



microservices - application databases

Monolithic vs Microservices

Monolithic

Microservices

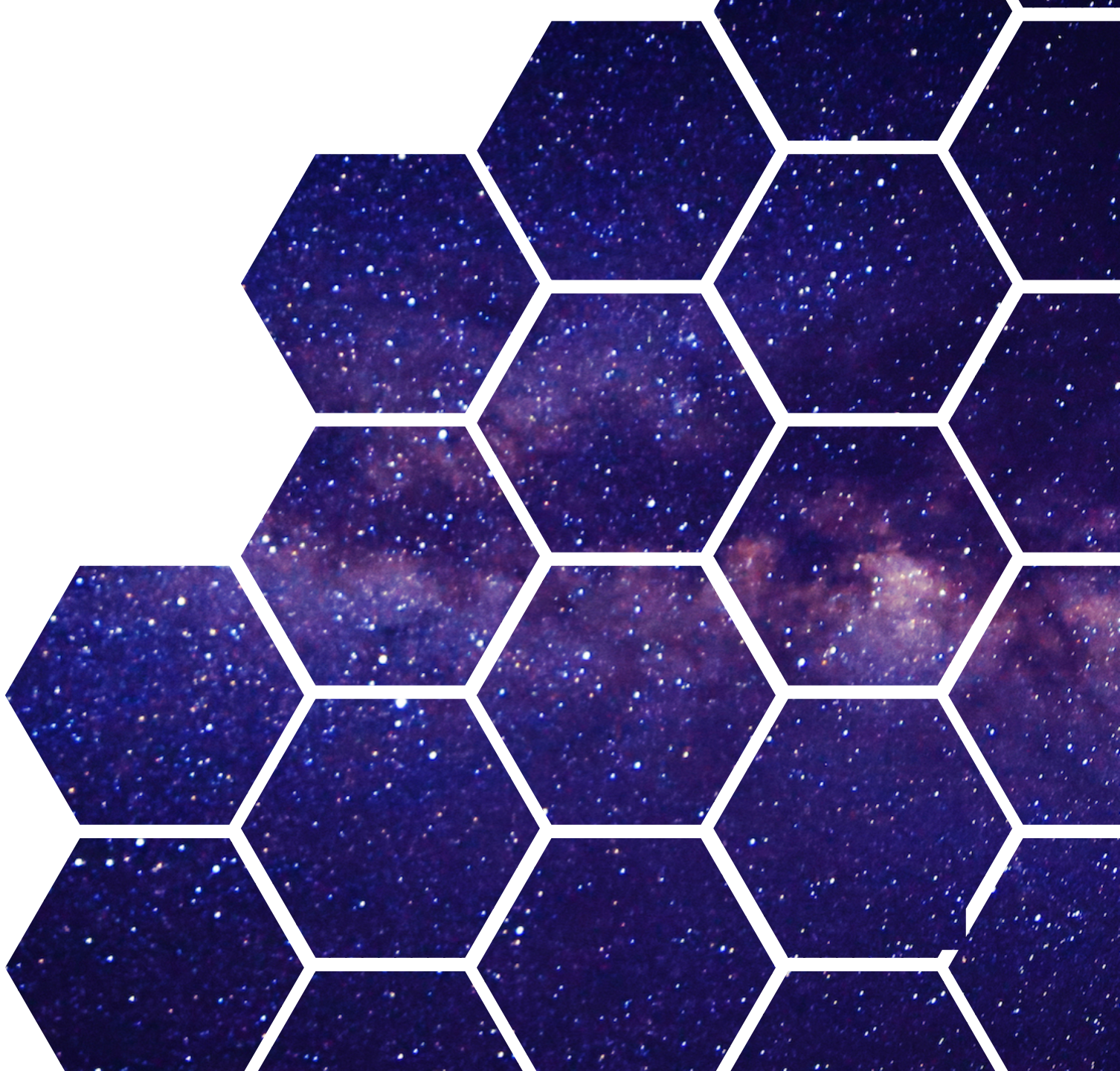
Microservices are **small**, **autonomous (💩)** services that work together.

Thin, **decoupled**, something you can rewrite in...2 weeks

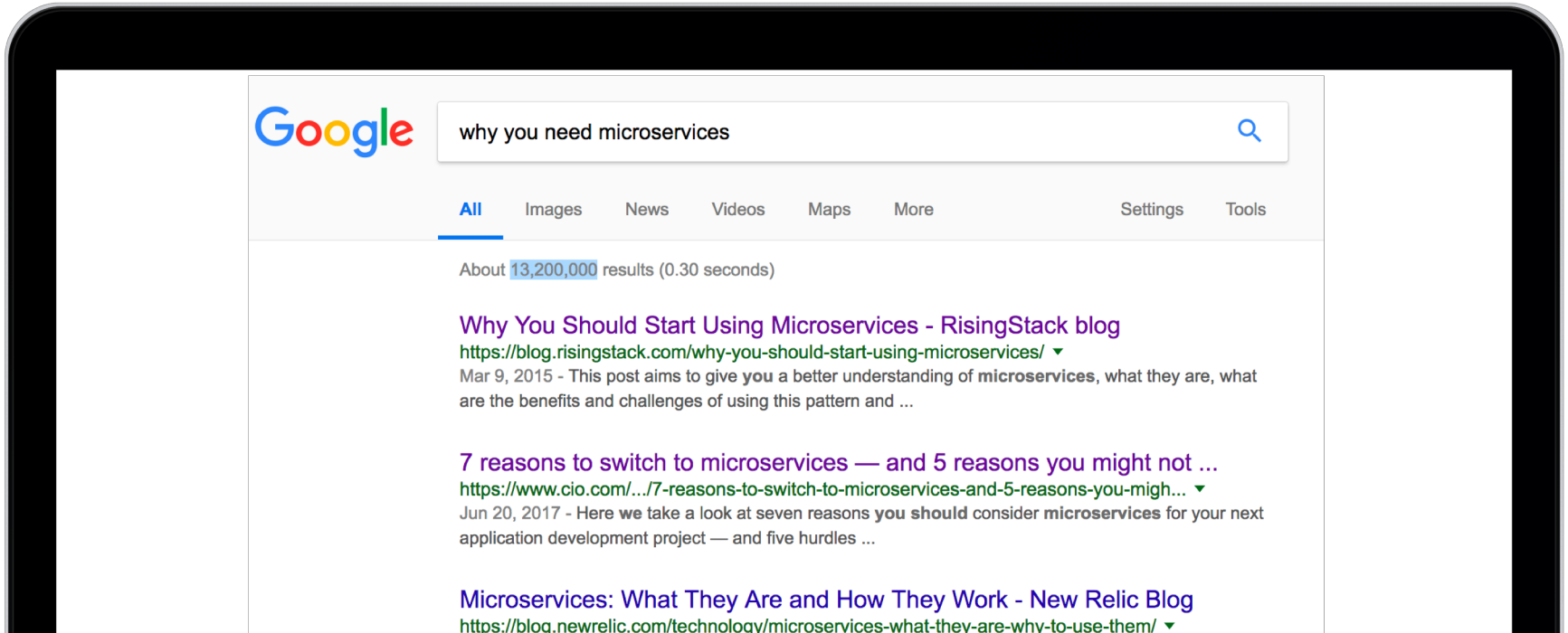
Separation between the services is enforced to avoid the perils of tight coupling and **increase resilience.**

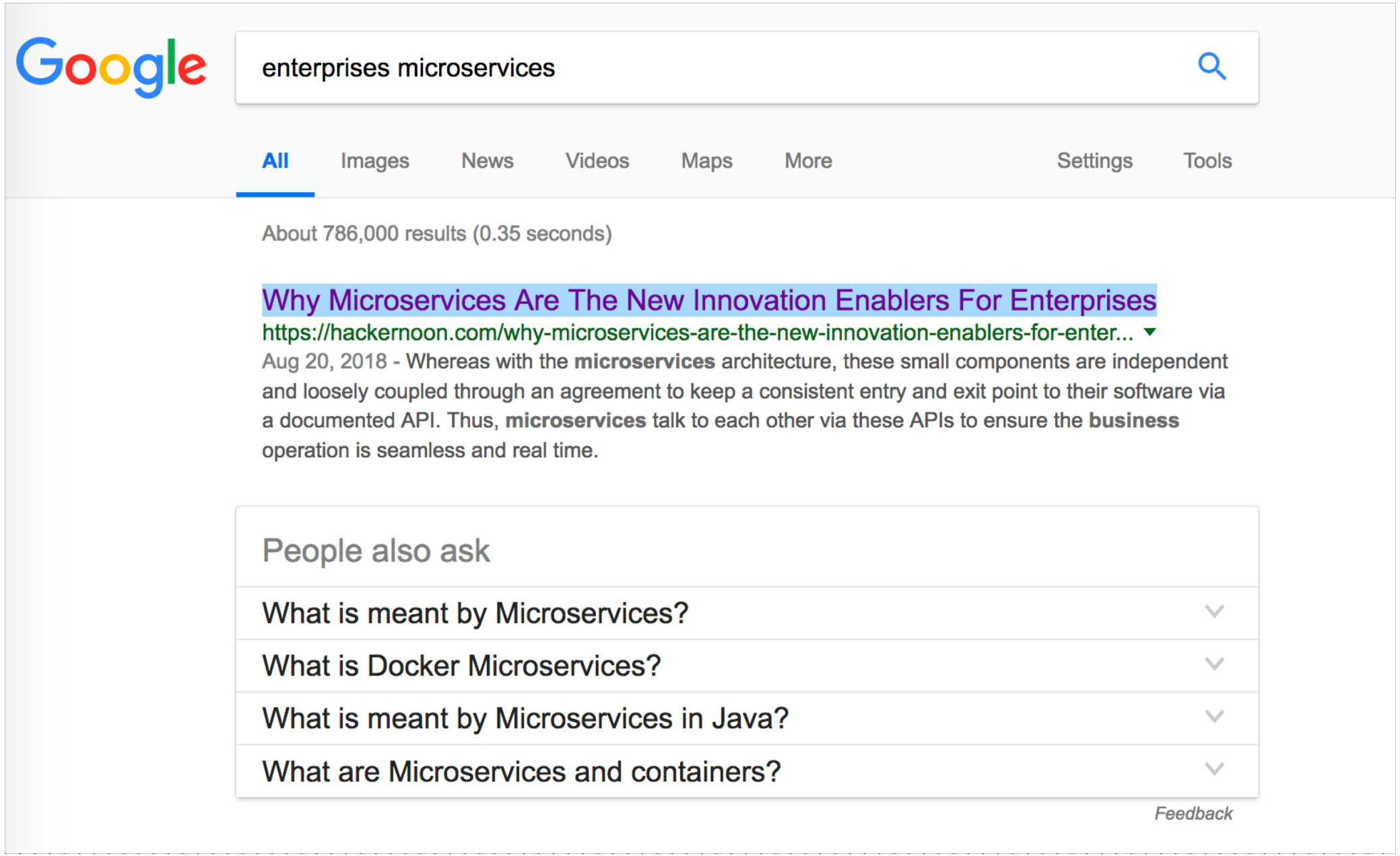
Can you make a change to a service and deploy it by itself without changing anything else?

WHY YOU NEED MICROSERVICES



I didn't know, so I google it.




Google
enterprises microservices
All
Images
News
Videos
Maps
More
Settings
Tools
About 786,000 results (0.35 seconds)
Why Microservices Are The New Innovation Enablers For Enterprises
https://hackernoon.com/why-microservices-are-the-new-innovation-enablers-for-enter...
Aug 20, 2018 - Whereas with the microservices architecture, these small components are independent and loosely coupled through an agreement to keep a consistent entry and exit point to their software via a documented API. Thus, microservices talk to each other via these APIs to ensure the business operation is seamless and real time.
People also ask
What is meant by Microservices?
What is Docker Microservices?
What is meant by Microservices in Java?
What are Microservices and containers?
Feedback


IT'S A TRAP
IT'S UNDOUBTEDLY A TRAP

1. Simplicity

applications become easier to build and maintain

2. Freedom

using different a different stack

3. Speed

applications become easier to build and maintain

4. Testing/QA

each microservice can be tested individually

5. Teams

Microservices are a blessing for distributed teams.

Organizations which design systems ...
are constrained to produce designs which are **copies** of the communication structures of these organizations.

CONWAY'S LAW

6. Isolation/Resilient/Flexible

if one microservice fails, the others will continue to work.

7. Security

simplify security monitoring because the various parts of an app are isolated.

8. Code boundaries

The code is organized around business capabilities.

9. High scalability

Microservices are a blessing for distributed teams.

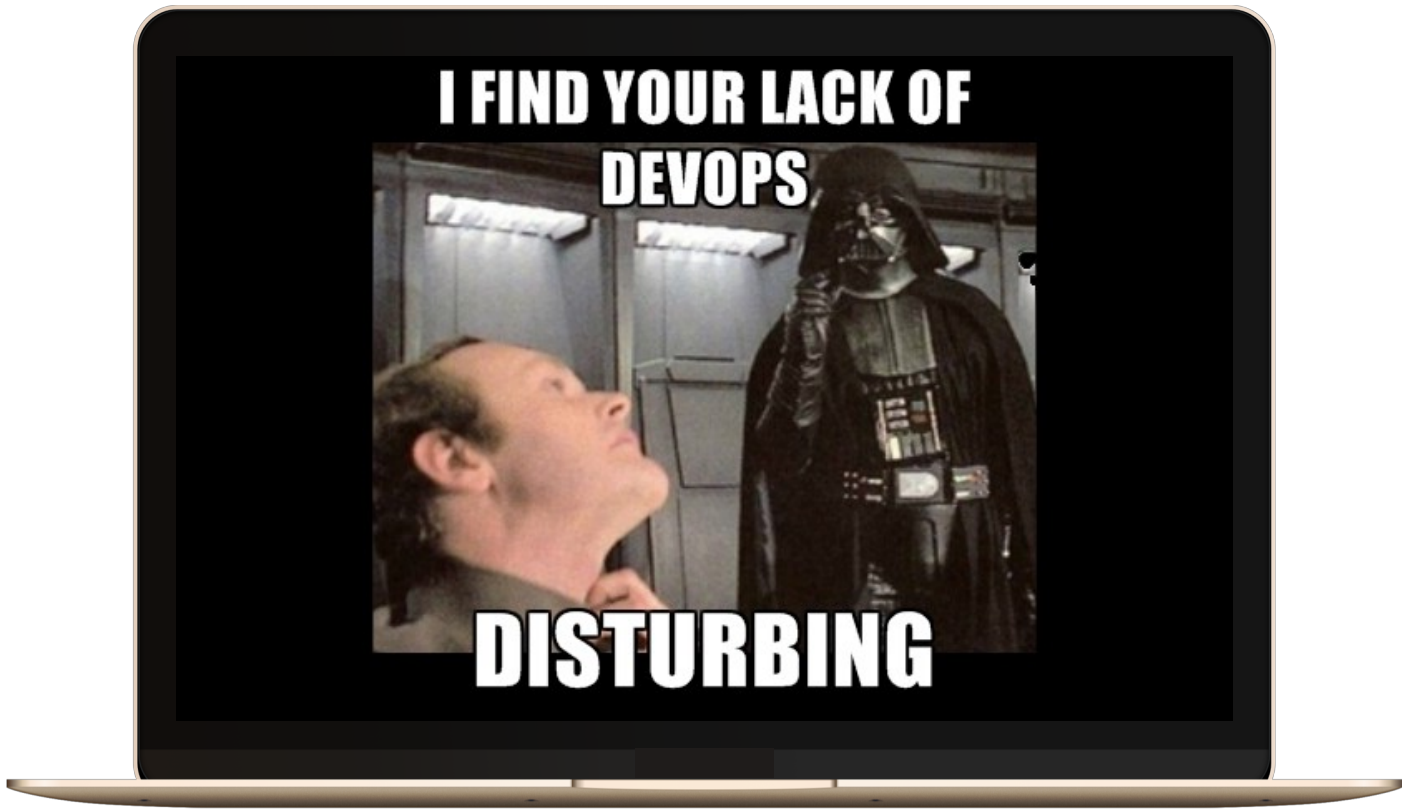
If you need them or not depends
on your **specific context** and **business.**

WHAT YOU NEED TO MAKE THEM WORK

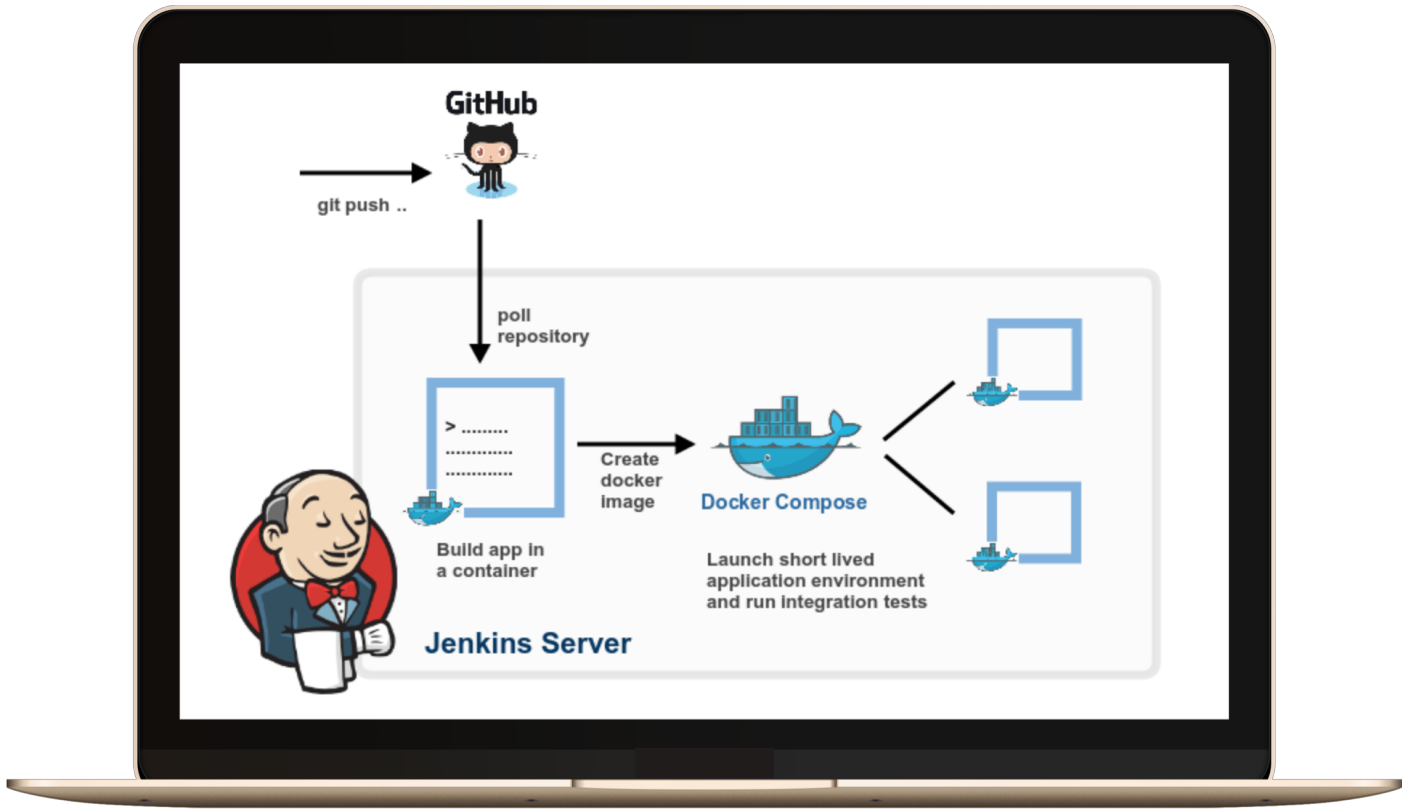
YOU NEED
TO MASTER CI

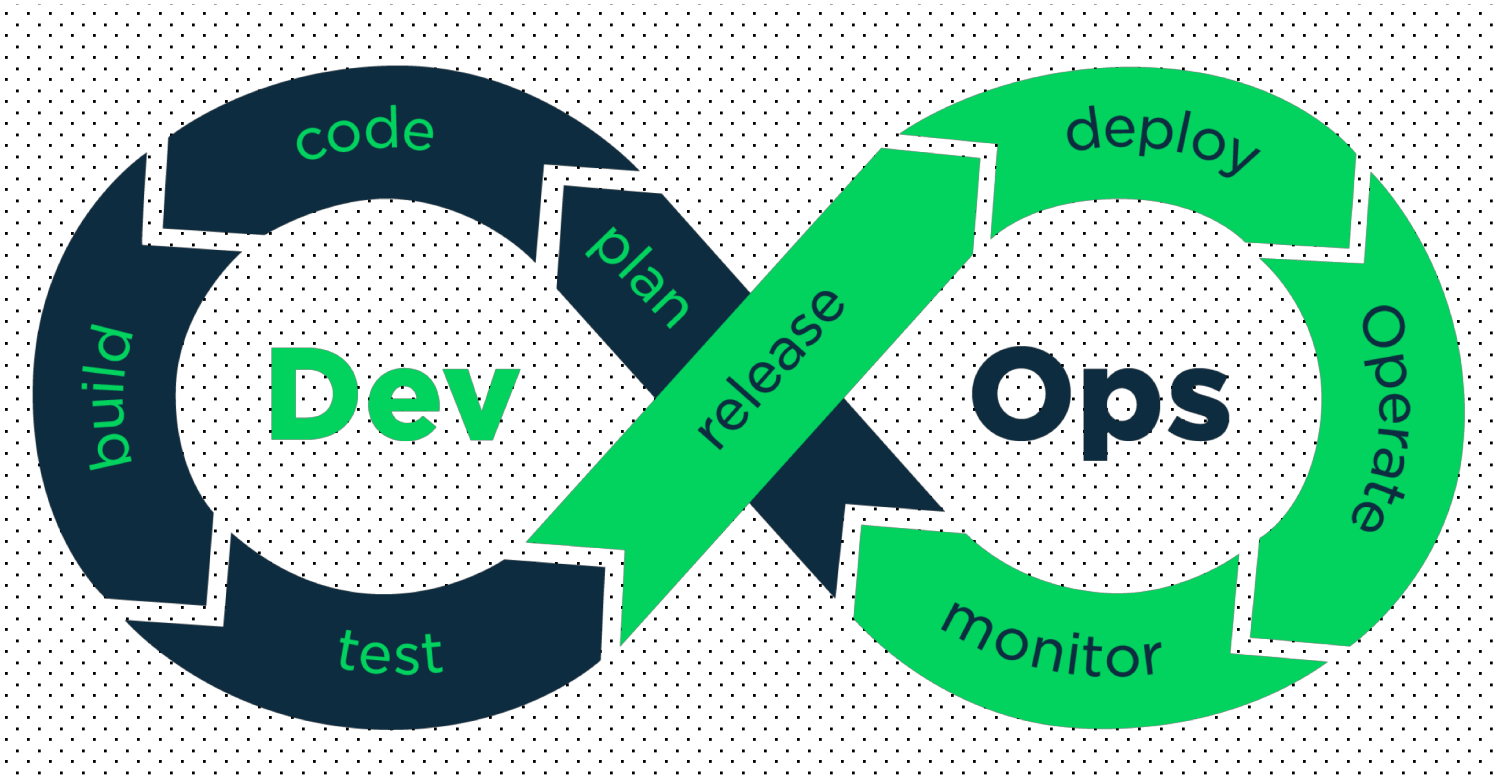
YOU NEED TO HAVE
THE RIGHT
INFRASTRUCTURE

YOU NEED TO HAVE A STRONG DEVOPS CULTURE



GENERALLY
1 CLICK DEPLOYMENT




code
plan
build
Dev
test
release
deploy
Ops
Operate
monitor

UNSOLICITED ADVICE

1. Now you need to **maintain** X services.

2. HTTP is an anti-pattern for microservices communication (if you don’t use a service mesh).

3. It’s a complete different way of developing services compared to Django (in example).

4. It’s **very easy** to do it wrong.

5. If you use a message queue (and you probably should), **don’t forget schemas**.

6. Asyncio makes writing microservices extremely easy.

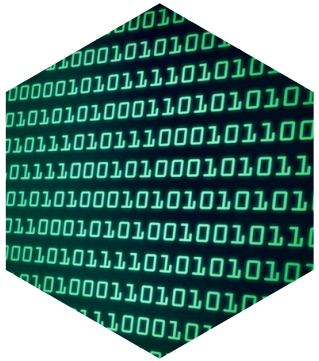
NOTE: cpu-bound tasks still block (the loop) 😊

7. Testing is still **very hard**.

8. Embrace data duplication.

COOL THINGS

TESTING IN PRODUCTION



SERVICE MESH

REDIS STREAMS

CANARY RELEASES

Thanks