

Microservices in your Datacentre

Phil Calçado

@pcalcado

philcalcado.com

Microservices in your Datacentre

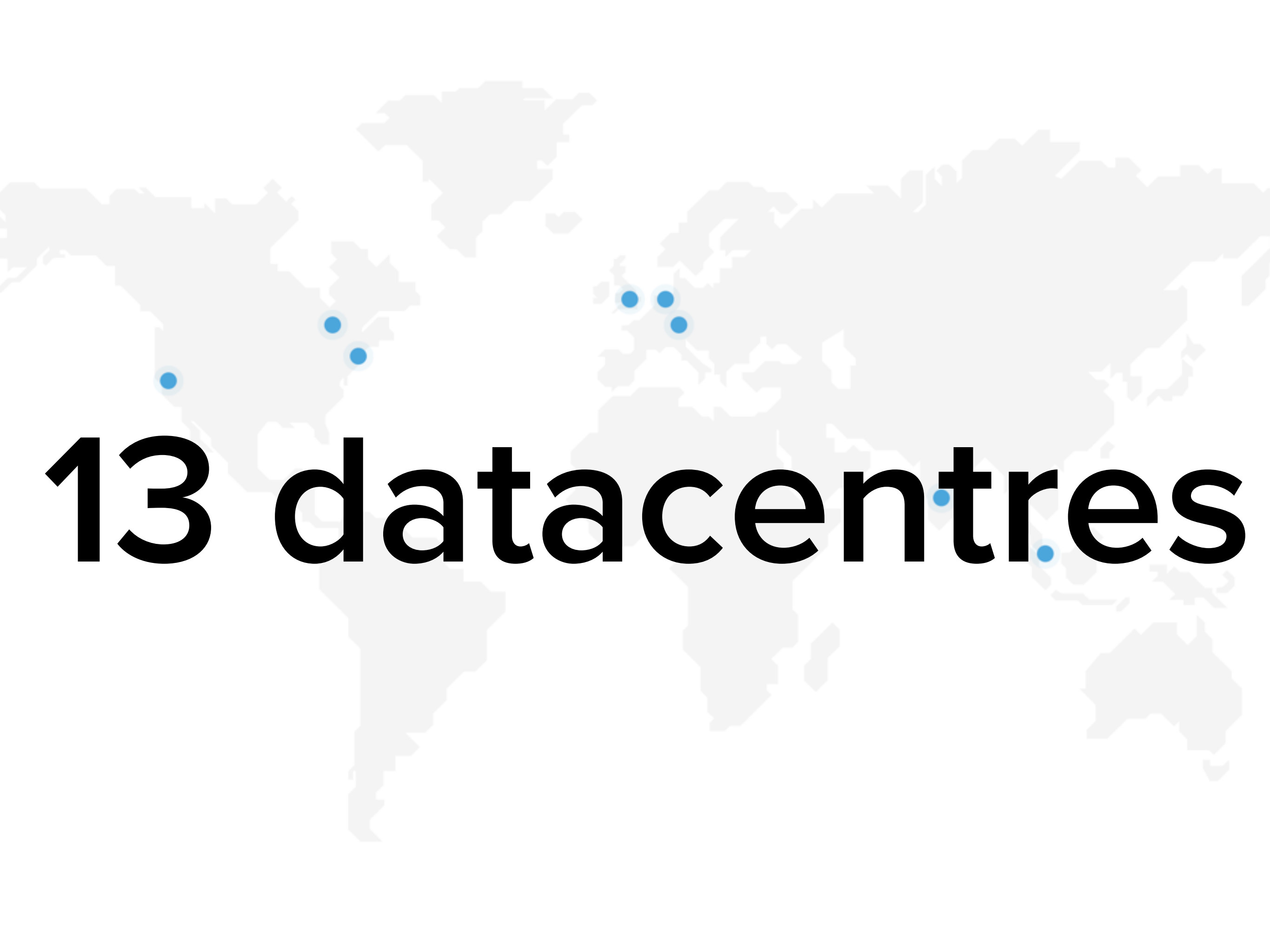
Phil Calçado

@pcalcado


philcalcado.com



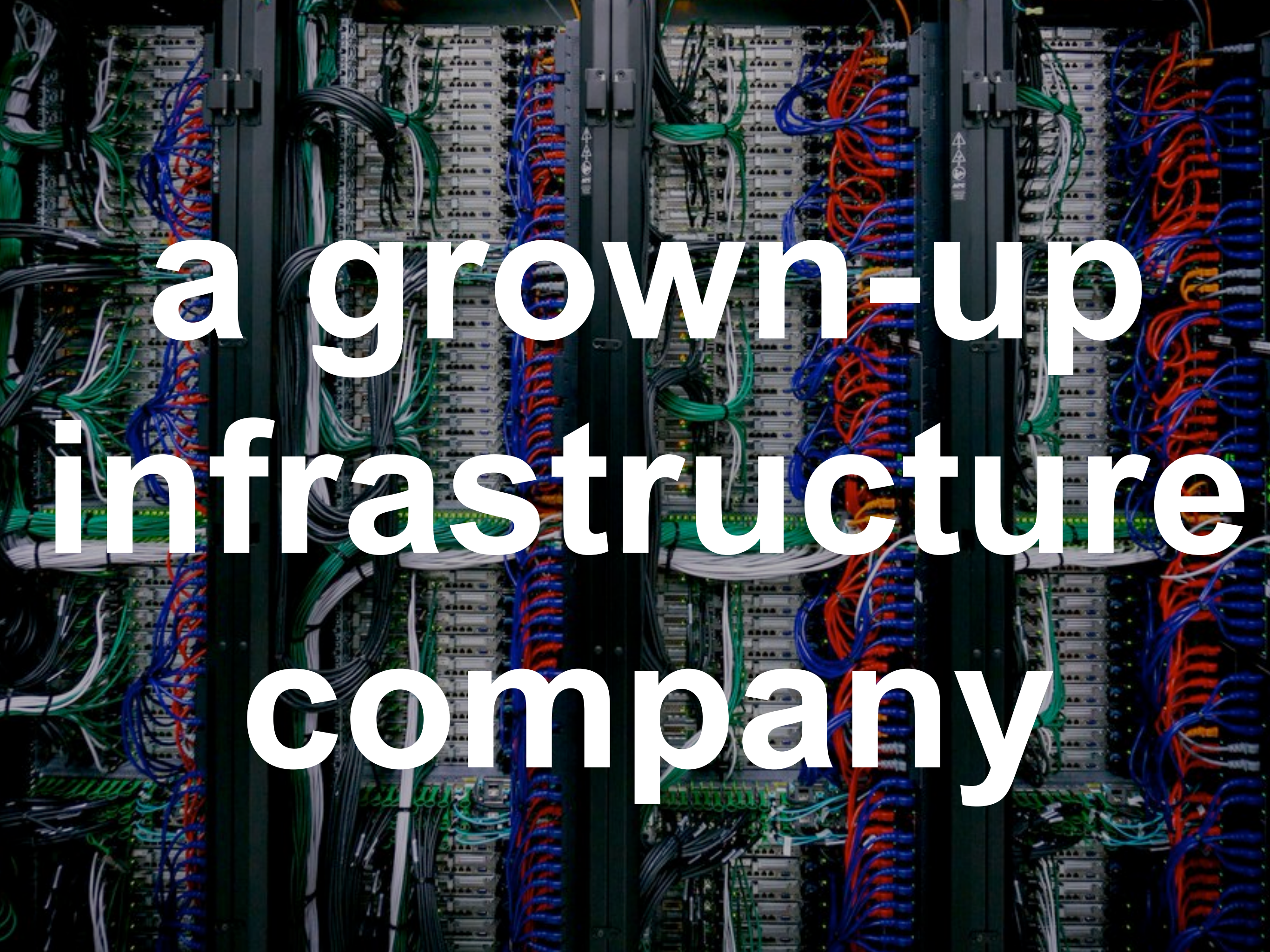
DigitalOcean



13 datacentres



**~18 million
droplets**



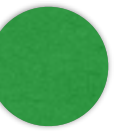
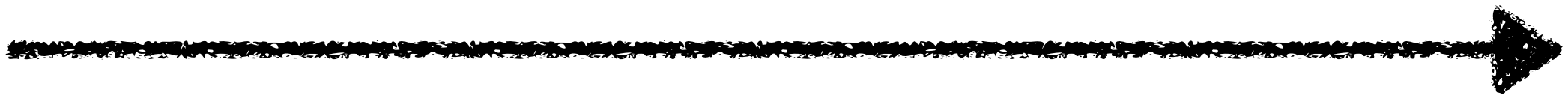
a grown-up
infrastructure
company

Why am I even here?

Projects take too long



idea



\$\$

Upon closer look





**Product
Engineering**



**Cloud
Engineering**

We will be talking about
Product Engineering



I am completely biased

"The general tendency is to over-design the second system, using all the ideas and frills that were cautiously sidetracked on the first one."

— Fred Brooks, "The Second-System Effect"

finagle scalamux service disc
every build pipelines consul ra
bbitmq kubernetes docker zip
what to
prioritise?
kin event sourcing services dir
ectory golang bffs sidecars thrif
tcircuits brokers zookeeper k
afka prometheus jenkins rund
eck cassandra hadoop hystrix

Some stuff is just mandatory.

MicroservicePrerequisites



Martin Fowler
28 August 2014

As I talk to people about using a **microservices architectural style** I hear a lot of optimism. Developers enjoy working with smaller units and have expectations of better modularity than with monoliths. But as with any architectural decision there are trade-offs. In particular with microservices there are serious consequences for operations, who now have to handle an ecosystem of small services rather than a single, well-defined monolith. Consequently if you don't have certain baseline competencies, you shouldn't consider using the microservice style.

You must be
this tall to use
microservices



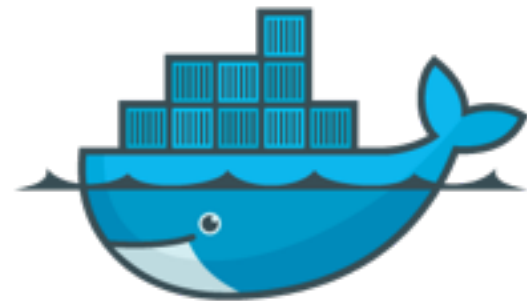
- Rapid provisioning
- Basic Monitoring
- Rapid application deployment

- Rapid provisioning 🕶️
- Basic Monitoring 🦌
- Rapid application deployment 🦌

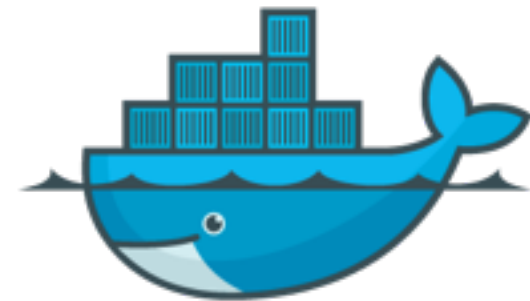


docker





docker



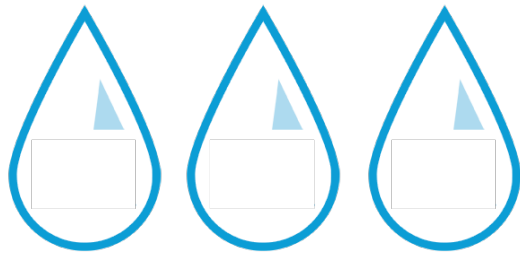
docker



Value-Added Service



docker



Critical Path Service



docker



PCI, ect.

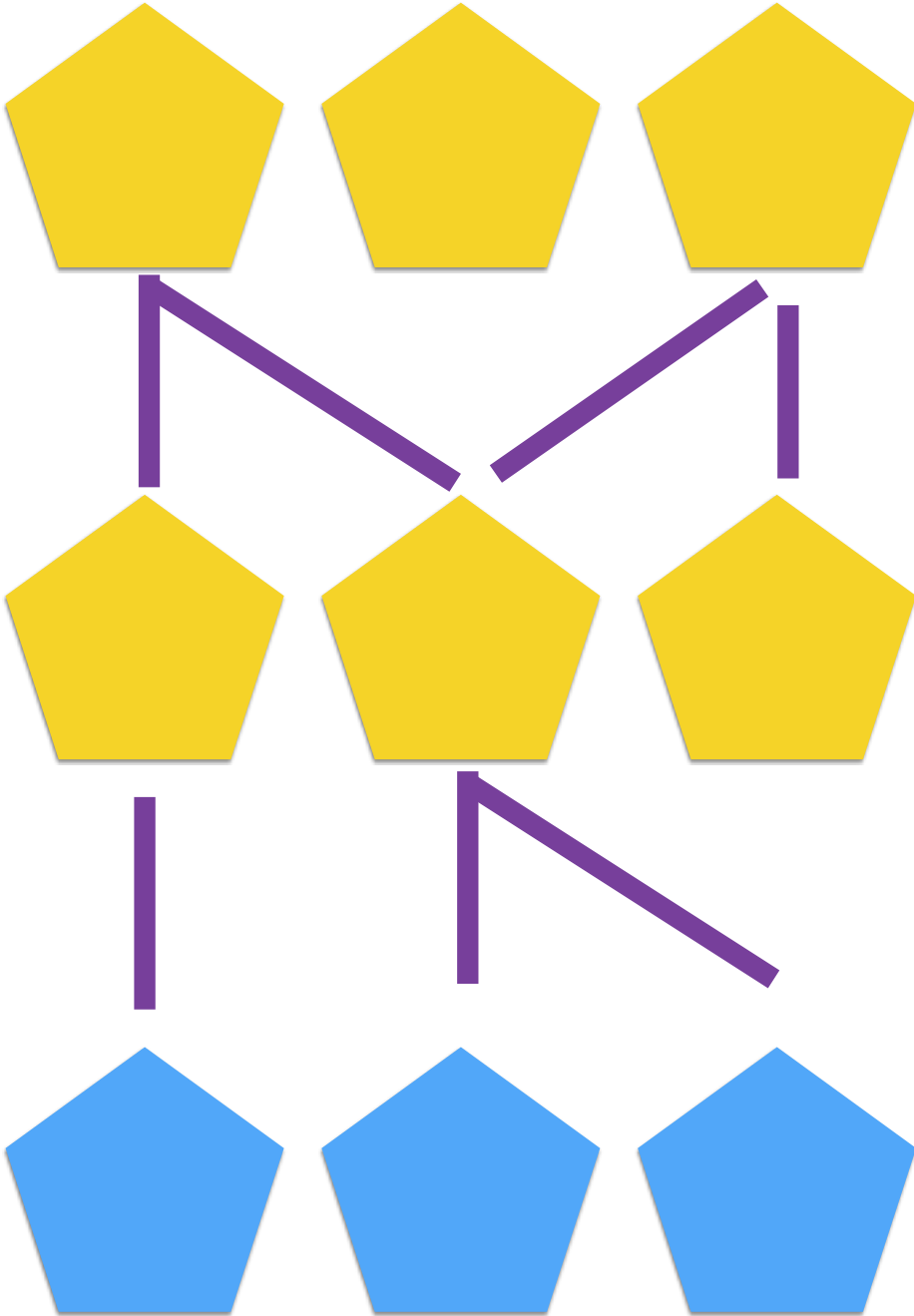


docker



And some other stuff
is just taste.

HTTP+JSON



Performance isn't likely to
be your problem using it.

Tolerant Reader leads to lots
of duplicated code.

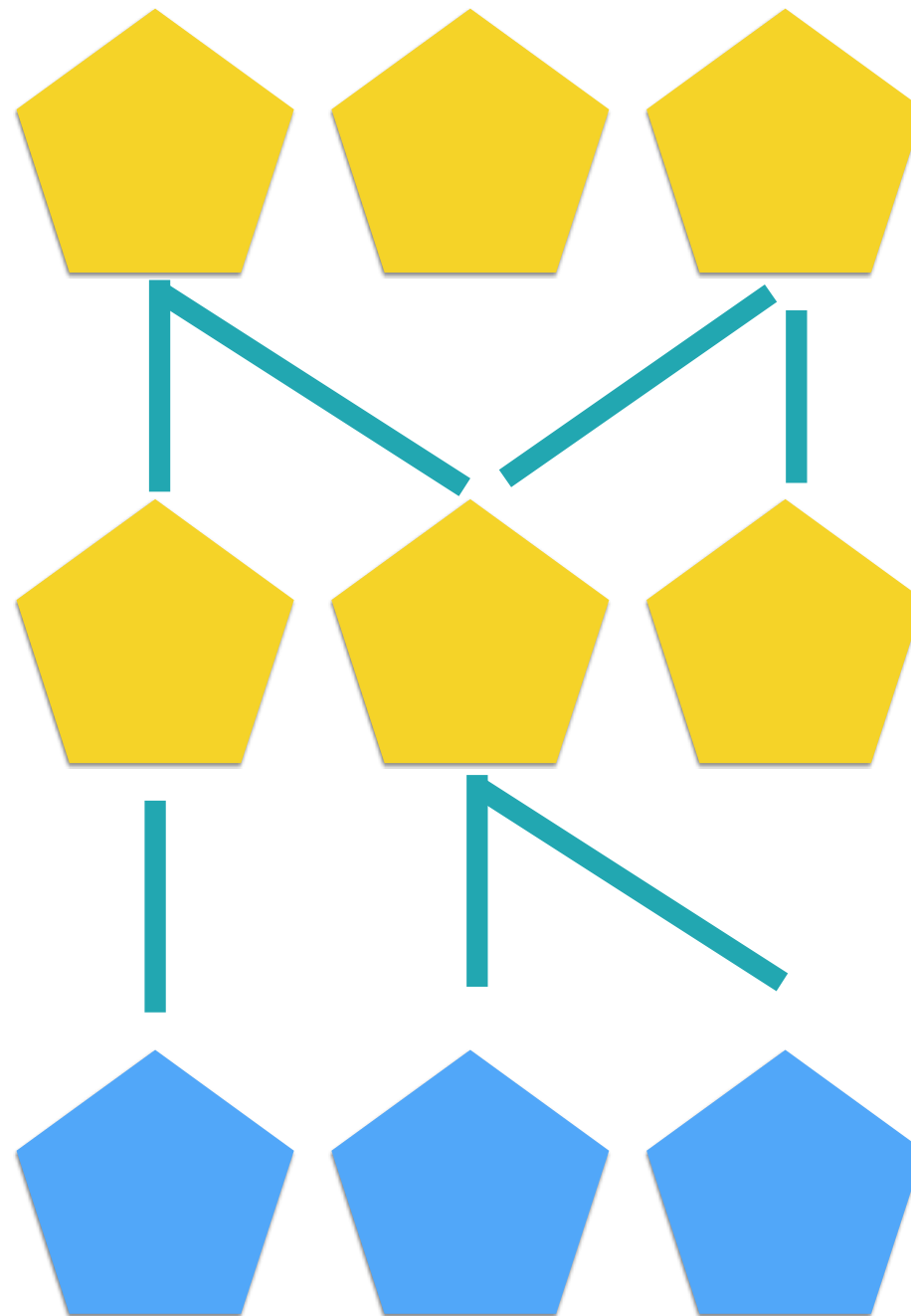
Duplicated code leads to
client libraries.

Client libraries lead to release management overhead.

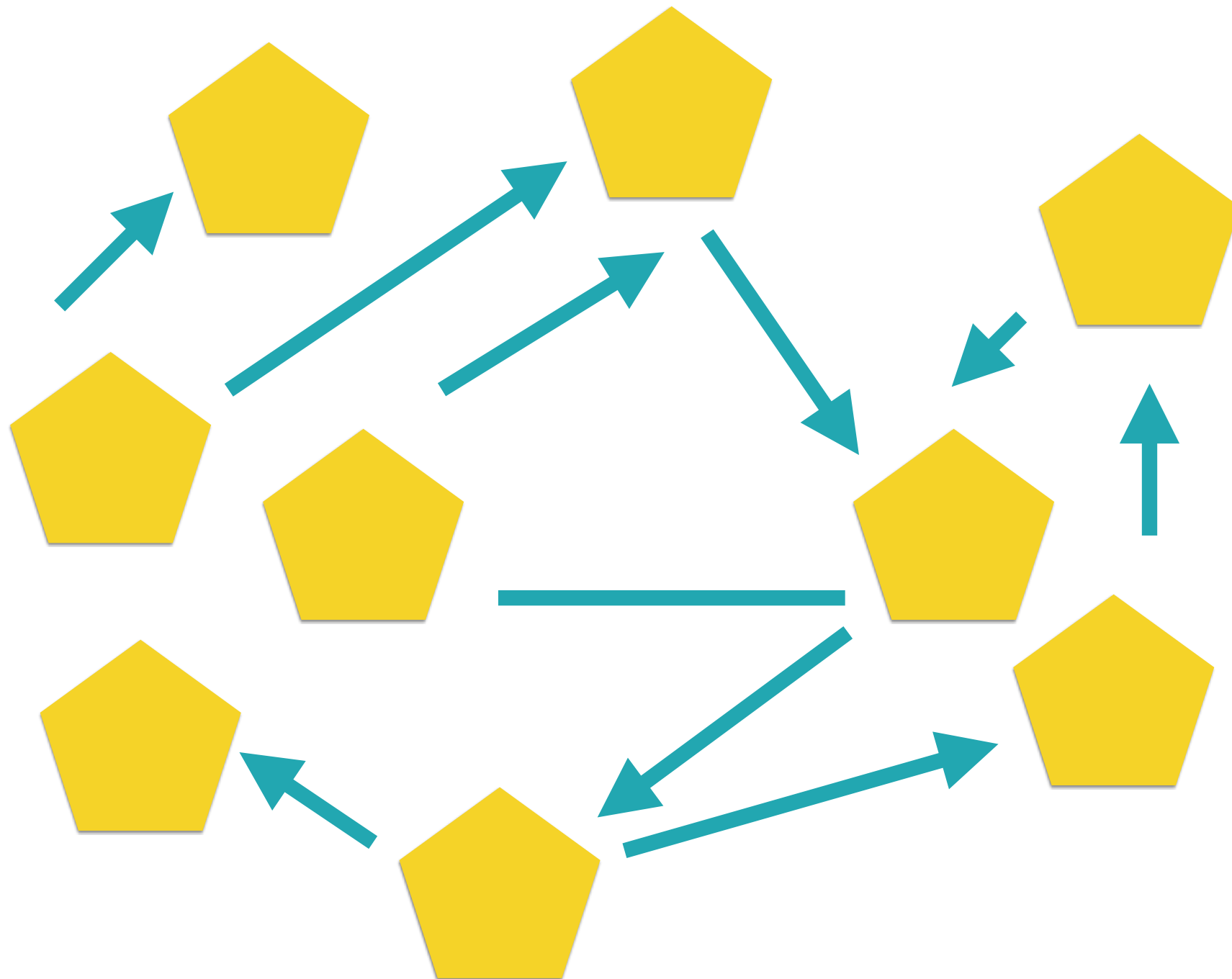
Client libraries lead to
hidden business logic.

Client Libraries need to be
efficient, lean, robust.

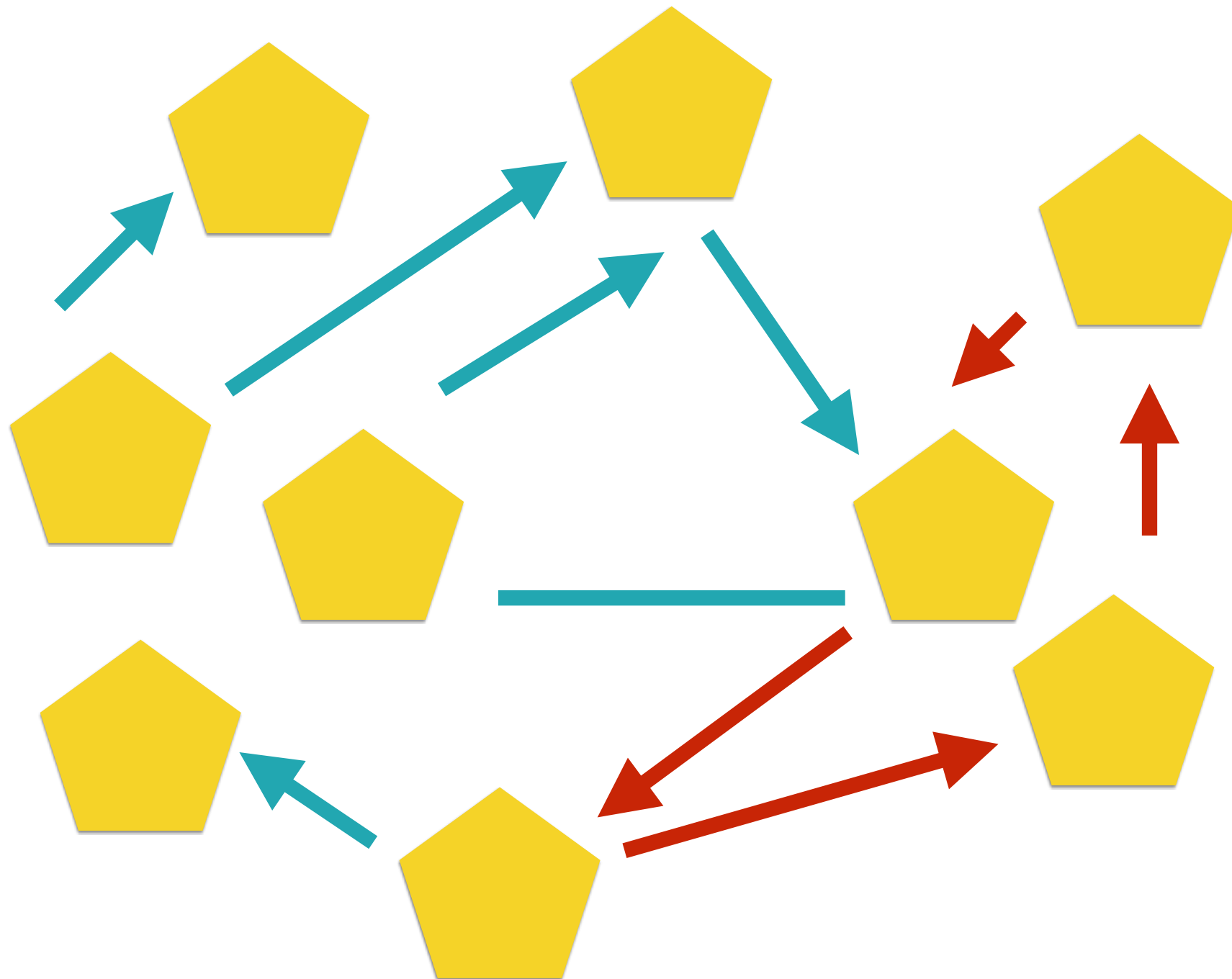
They don't need to
be written manually.



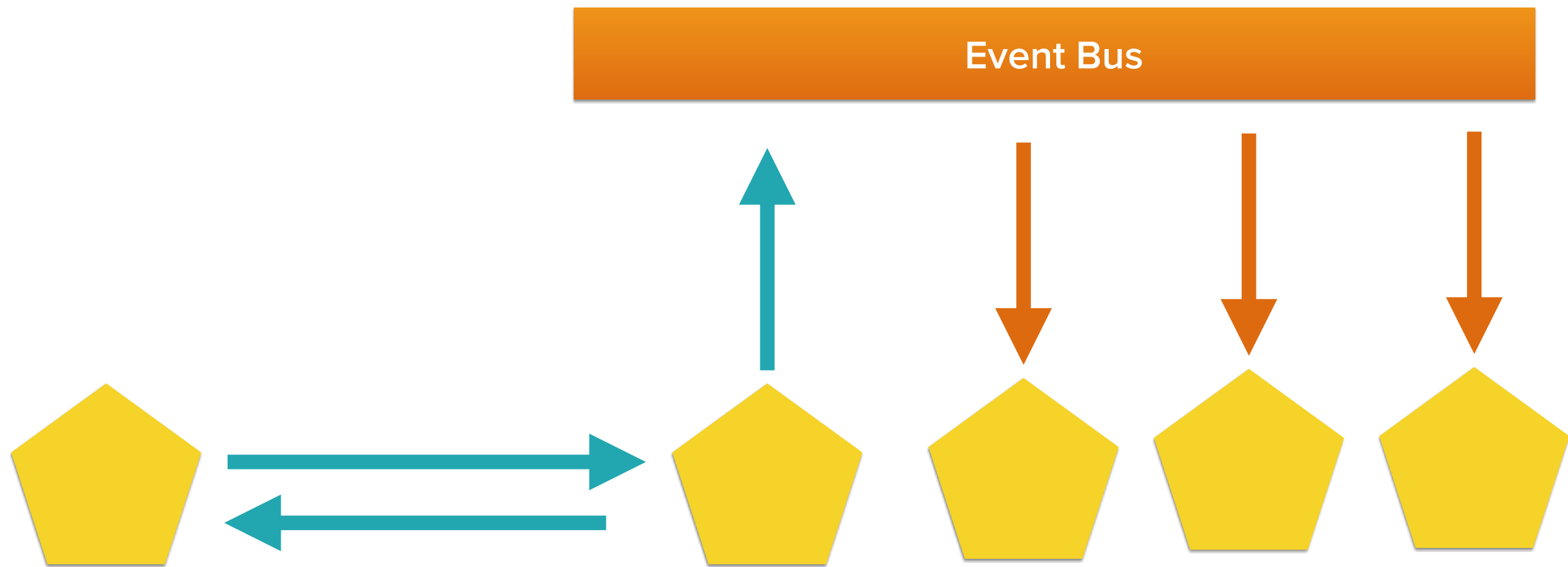
Something I'm sure
you'll have:



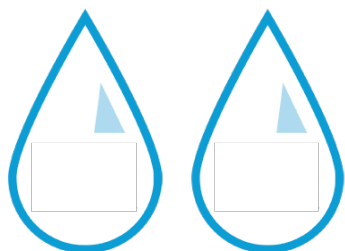
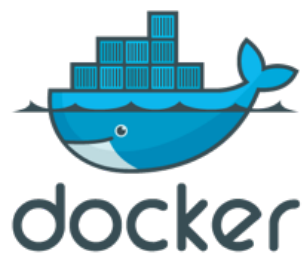
Circular dependencies.



Circles can often be broken by async events



We are event driven by
nature



REQUEST BODY

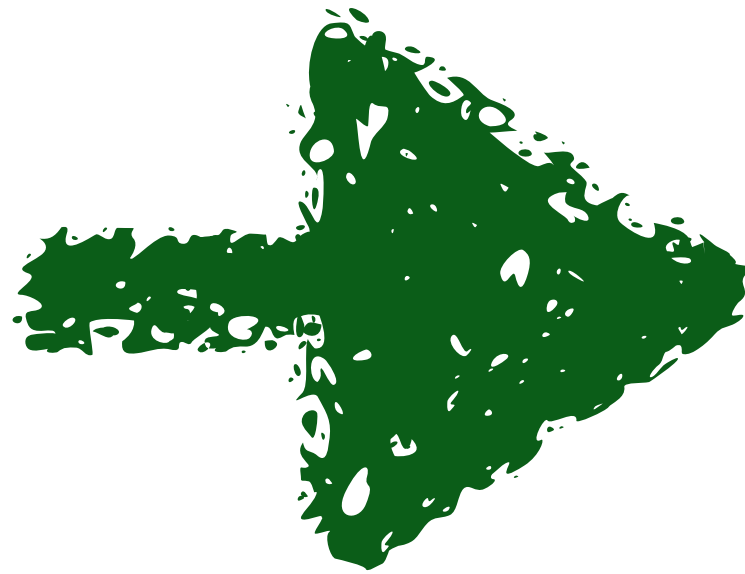
```
{  
  "type": "resize",  
  "disk": true,  
  "size": "lgb"  
}
```

RESPONSE HEADERS

```
content-type: application/json; charset=utf-8  
status: 201 Created  
ratelimit-limit: 1200  
ratelimit-remaining: 1046  
ratelimit-reset: 1415984218
```

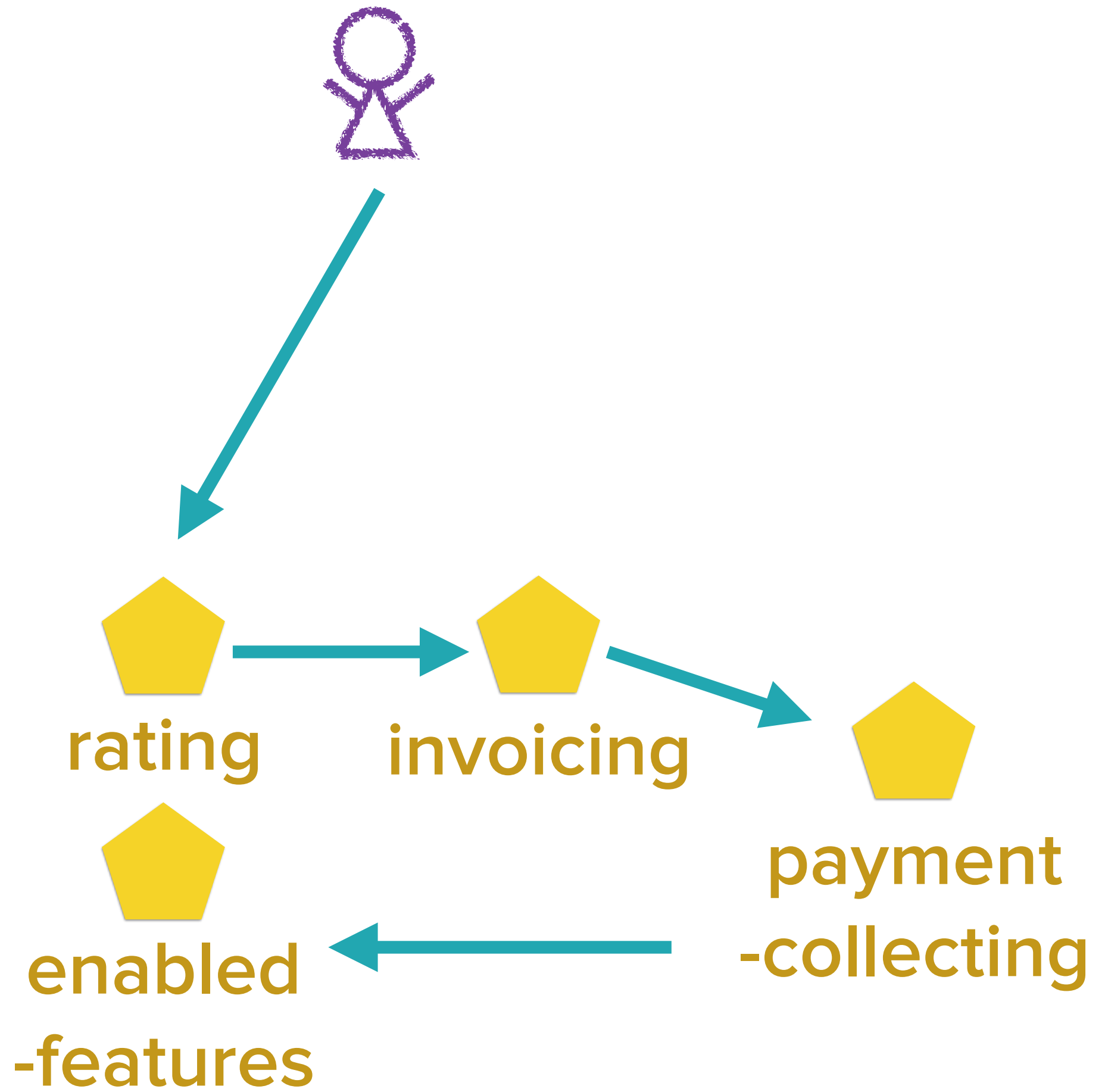
RESPONSE BODY

```
{  
  "action": {  
    "id": 36804888,  
    "status": "in-progress",  
    "type": "resize",  
    "started_at": "2014-11-14T16:33:17Z",  
    "completed_at": null,  
    "resource_id": 3164450,  
    "resource_type": "droplet",  
    "region": "nyc3",  
    "region_slug": "nyc3"  
  }  
}
```

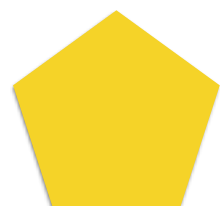


What should be a
service?

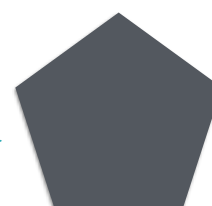




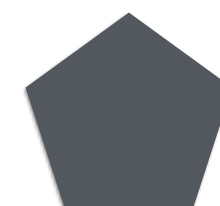
Highly
visible



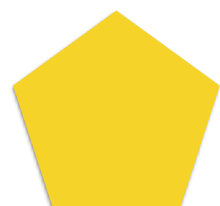
rating



invoicing



payment
-collecting



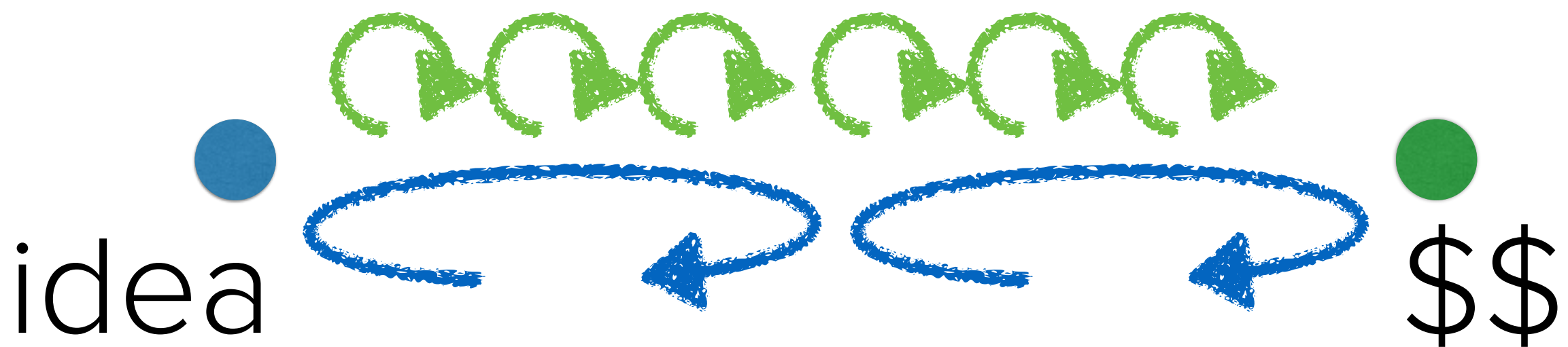
enabled
-features

Not
visible

Novel

Commodity

It's early days, but we can
already see progress.



Q&A