Quick poll…

Welcome to sunny London!

Tammer Saleh

Geek: **Unix, Ruby, Golang**, etc

Cloud Foundry @ Pivotal

http://tammersaleh.com | tsaleh@pivotal.io

Microservice Anti-patterns

How not to go down in flames.

Why microservices?

What is a **microservice**, and **why do I care**?

Monolithic

Entire application in a **single codebase**, deployed and scaled as **a single unit**.

Monolithic

Hard to scale the **application**.
Impossible to scale the **team**.

It’s not about code… **It’s about teams.**

But it can go wrong.

Here are the most **common problems** we see in the wild, and **how to fix them**.

Overzealous Services

The most common mistake is to **start with microservices**

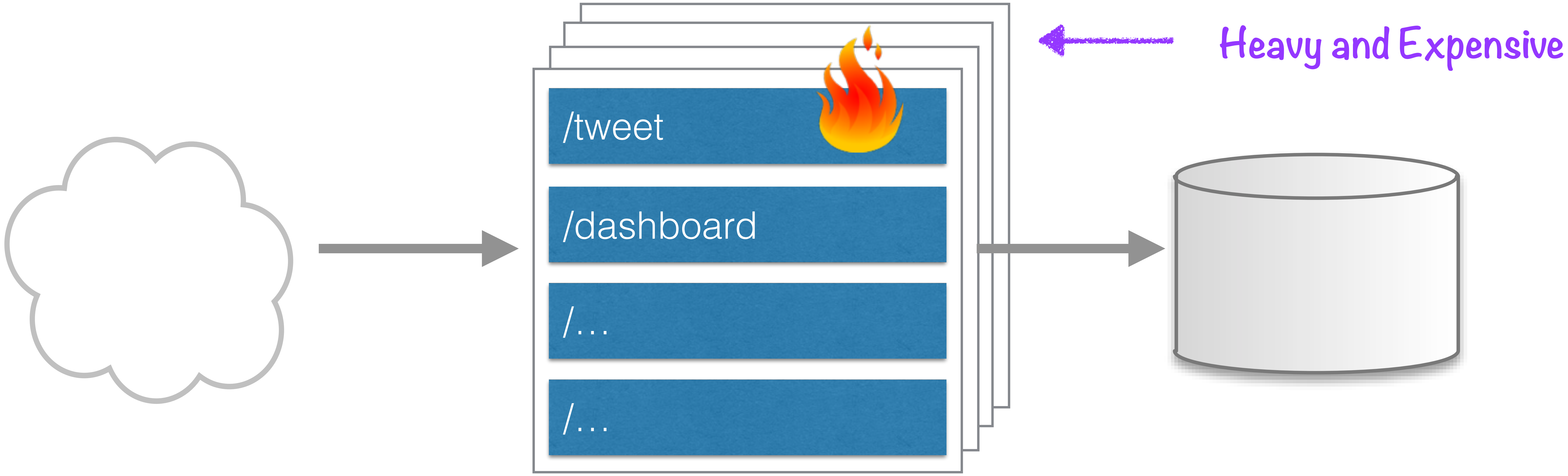
Boring is Beautiful™

Solution: Start monolithic and extract

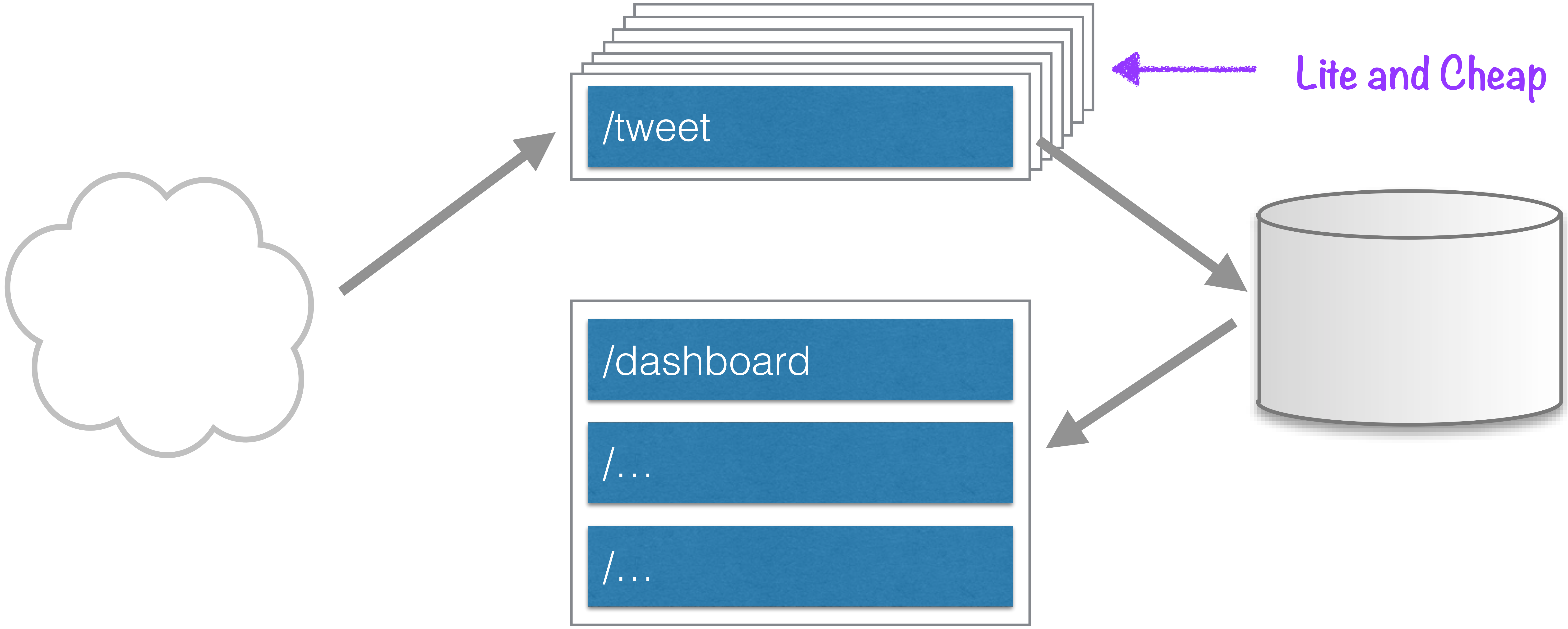
Microservices are complex and add a **constant tax to development**.

Build a **boring application** and extract services as needed.

Twitter



Twitter



/tweet

/dashboard

/…

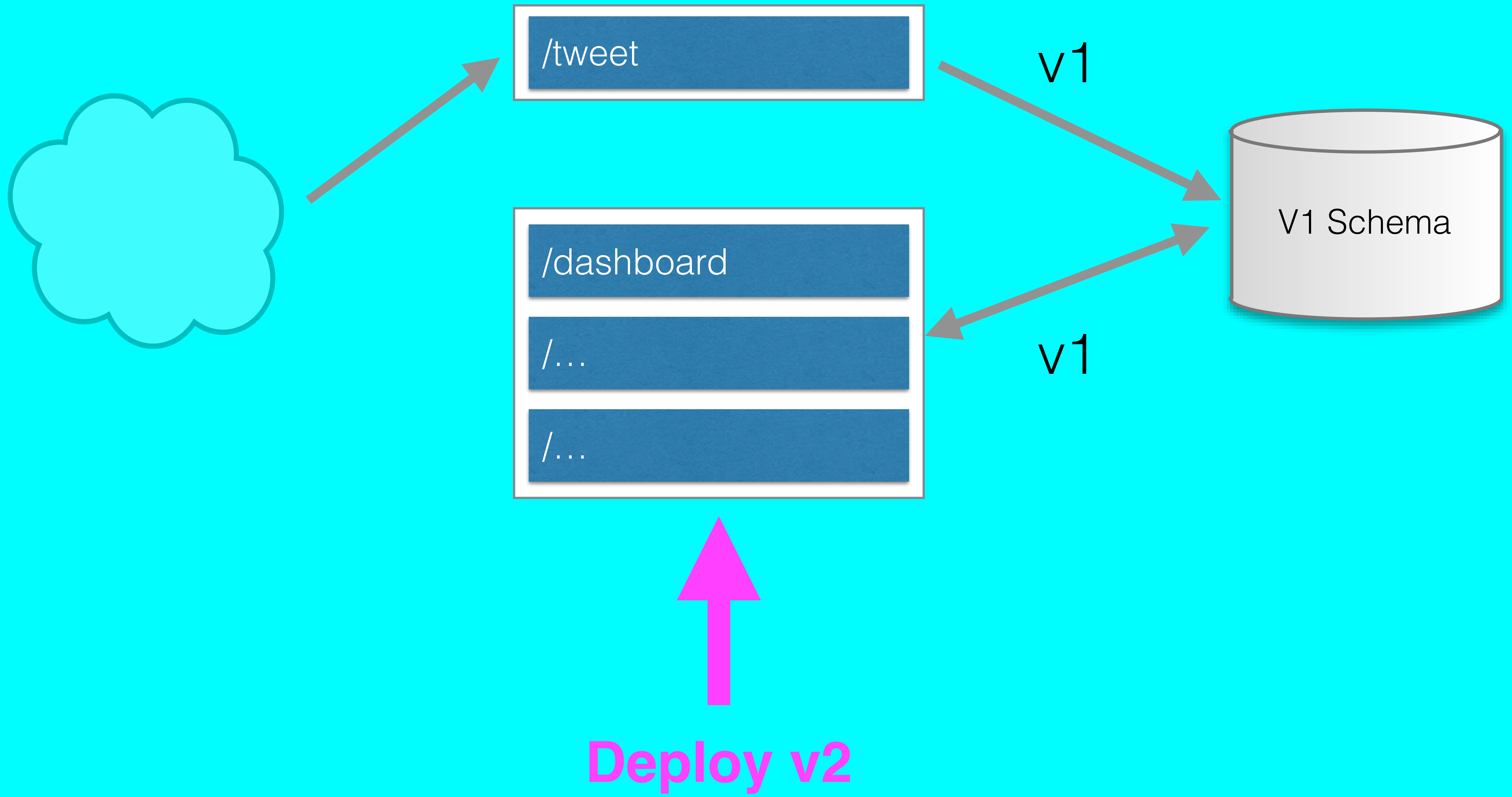
/…

Lite and Cheap

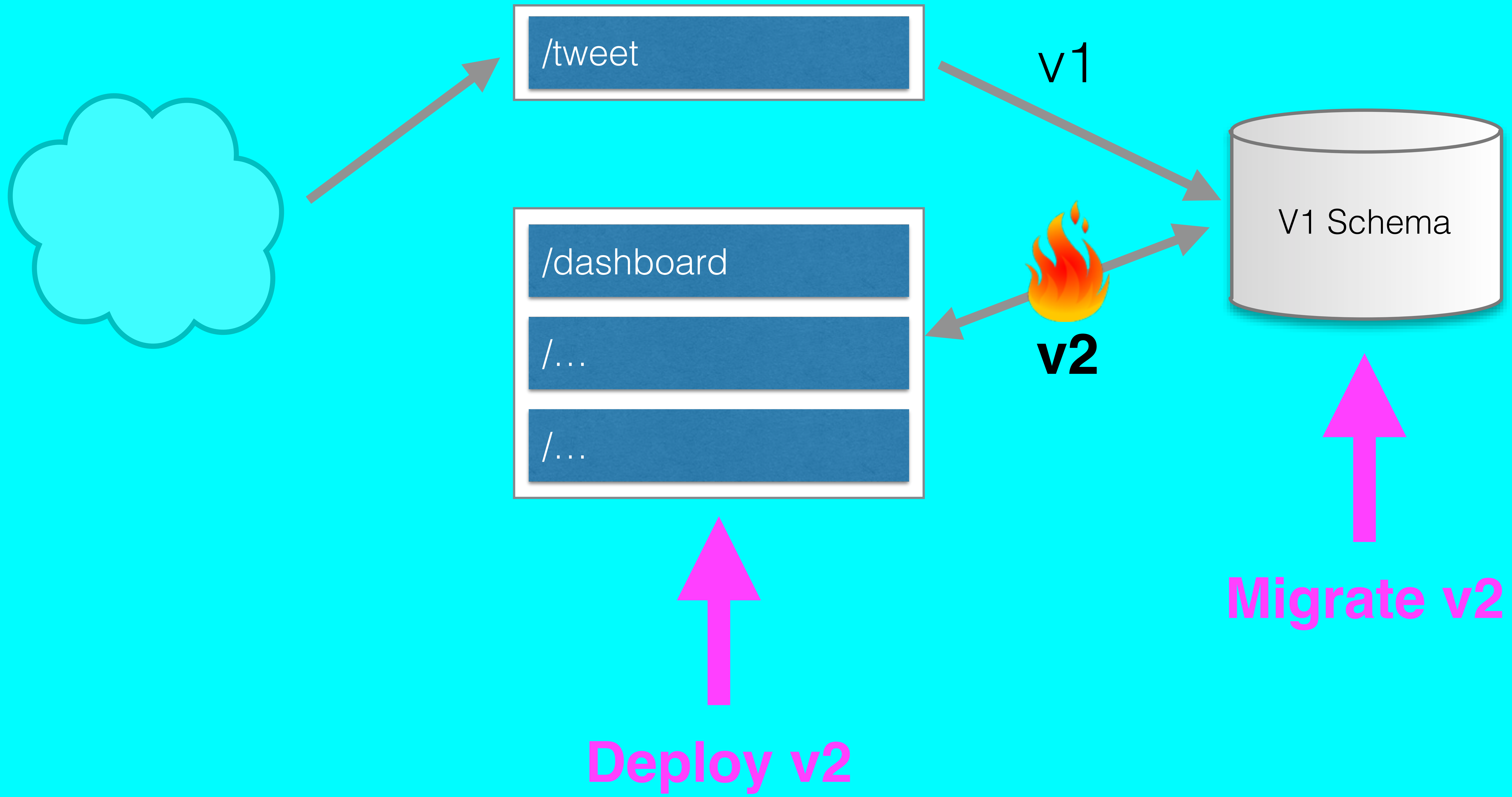
Congratulations

You’re now a microservice architect.

Schemas everywhere



Schemas everywhere



/tweet

v1

V1 Schema

/dashboard

/…

/…

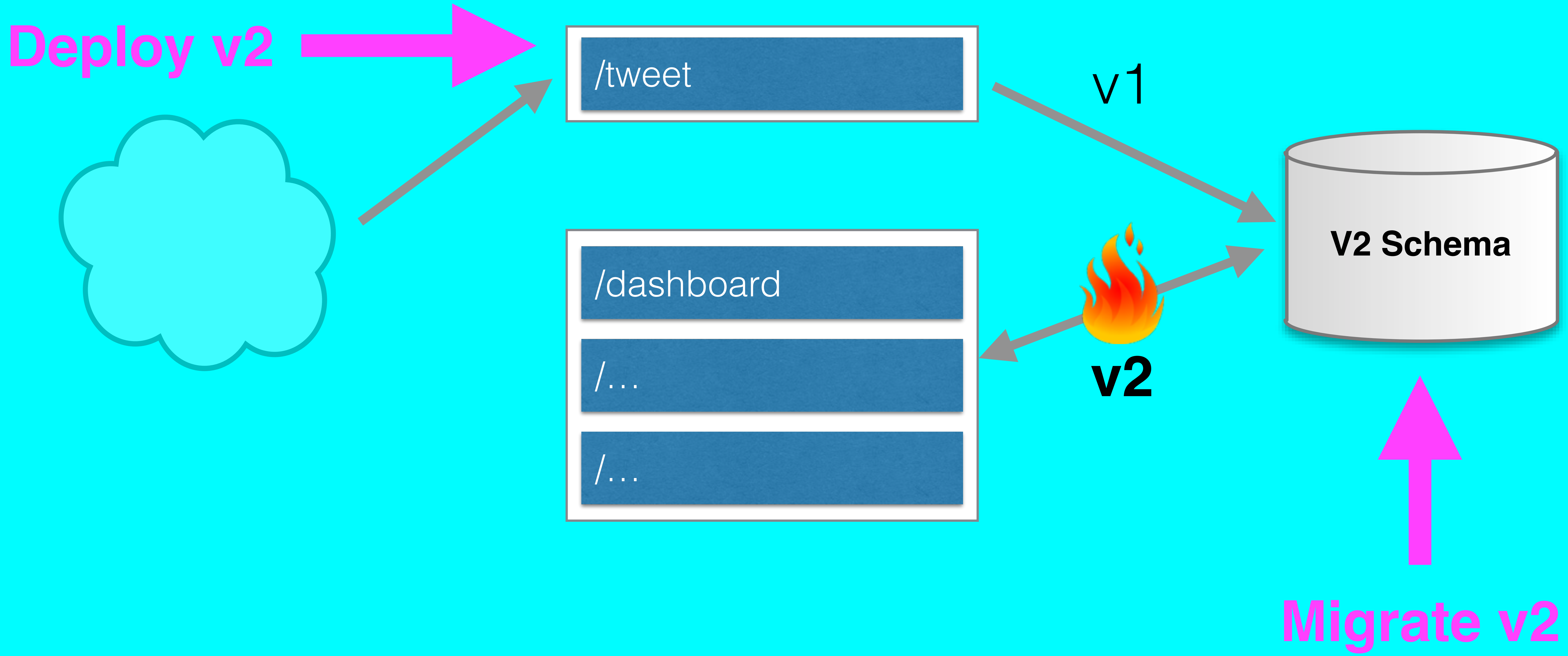
🔥

v2

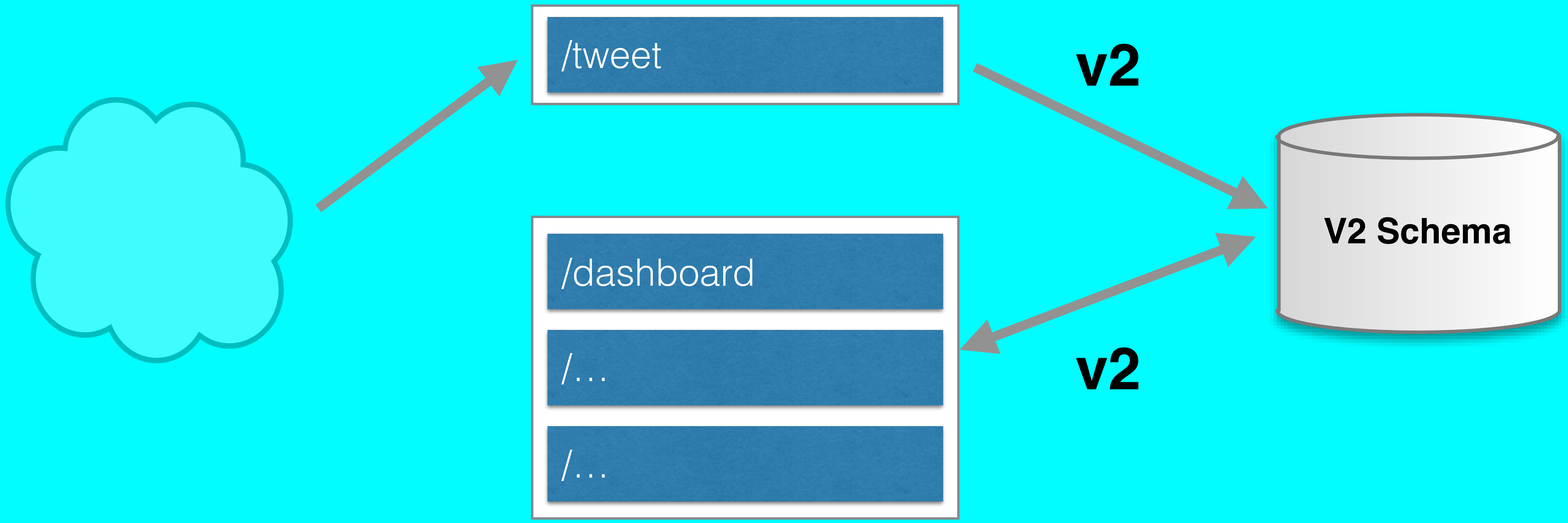
Deploy v2

Migrate v2

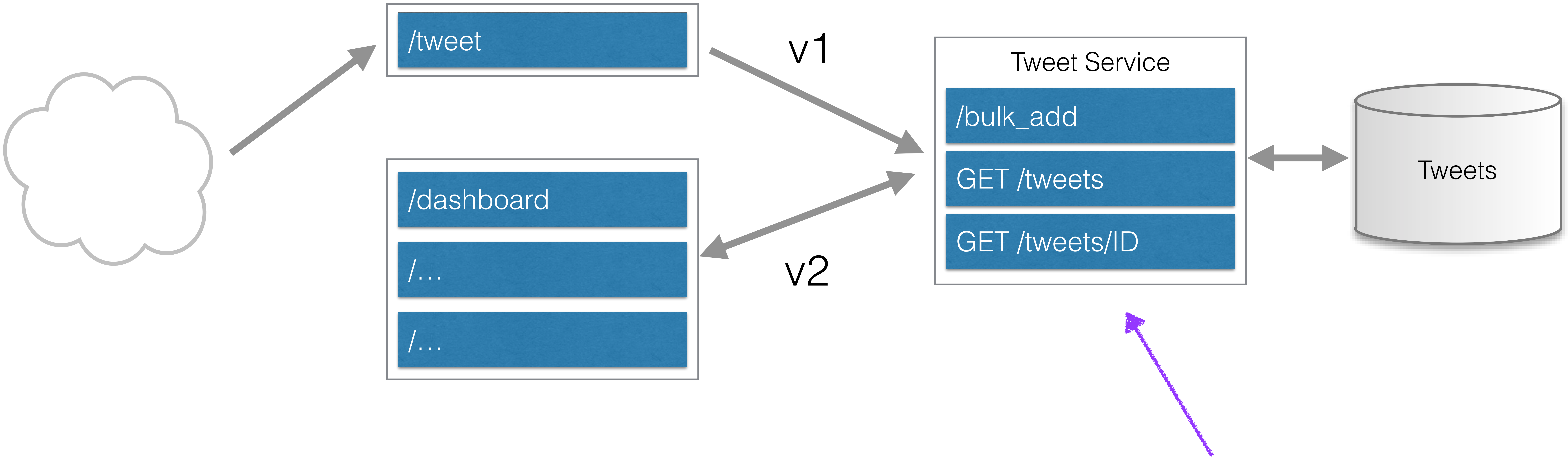
Schemas everywhere



Schemas everywhere



Solution: Gatekeeper



/tweet

/dashboard

/…

/…

v1

v2

Tweet Service

/bulk_add

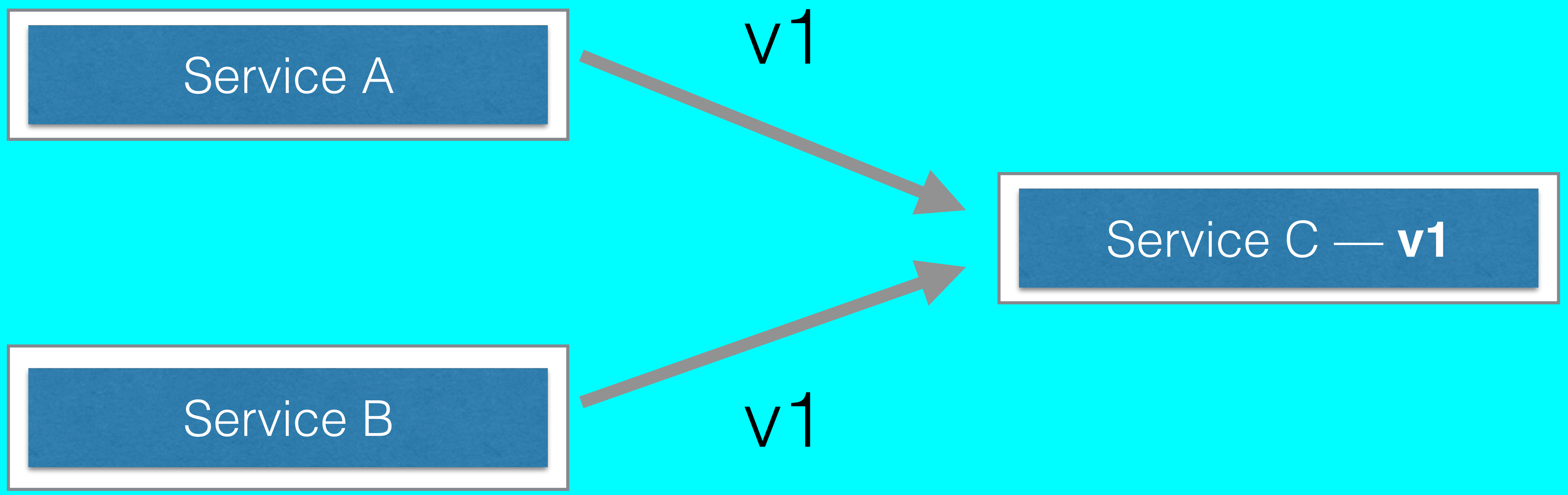
GET /tweets

GET /tweets/ID

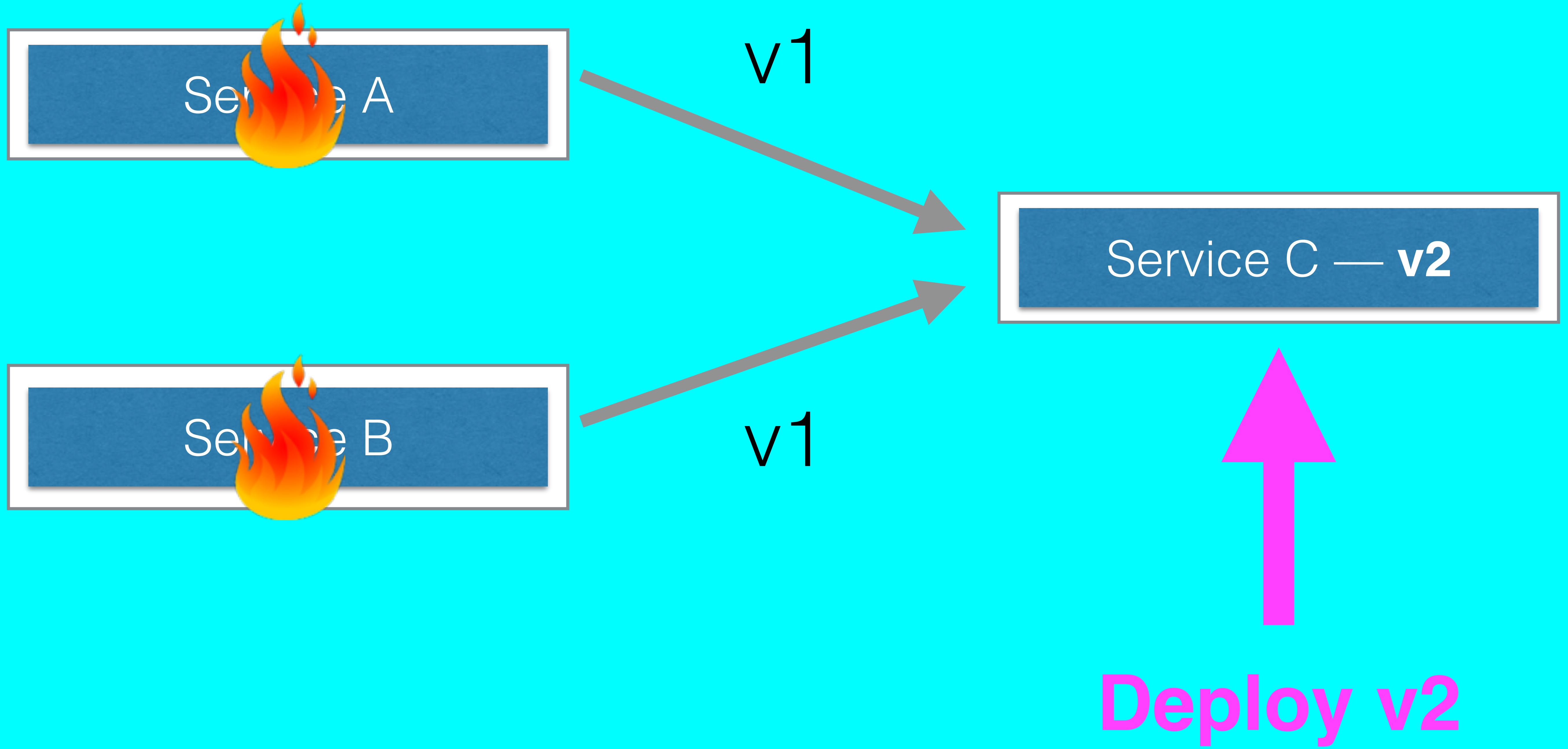
Tweets

Owns database and migrations

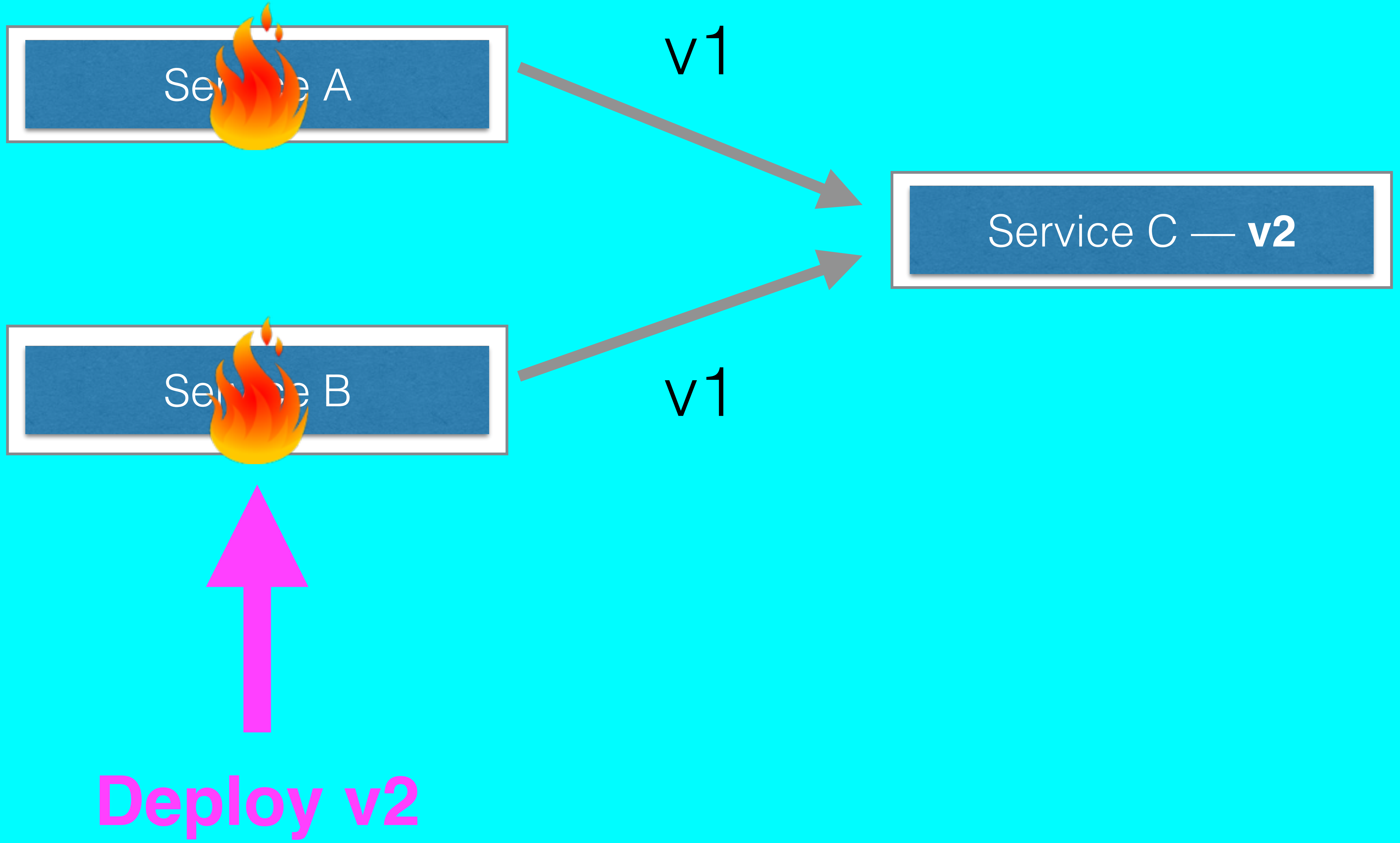
Lock-step deployment



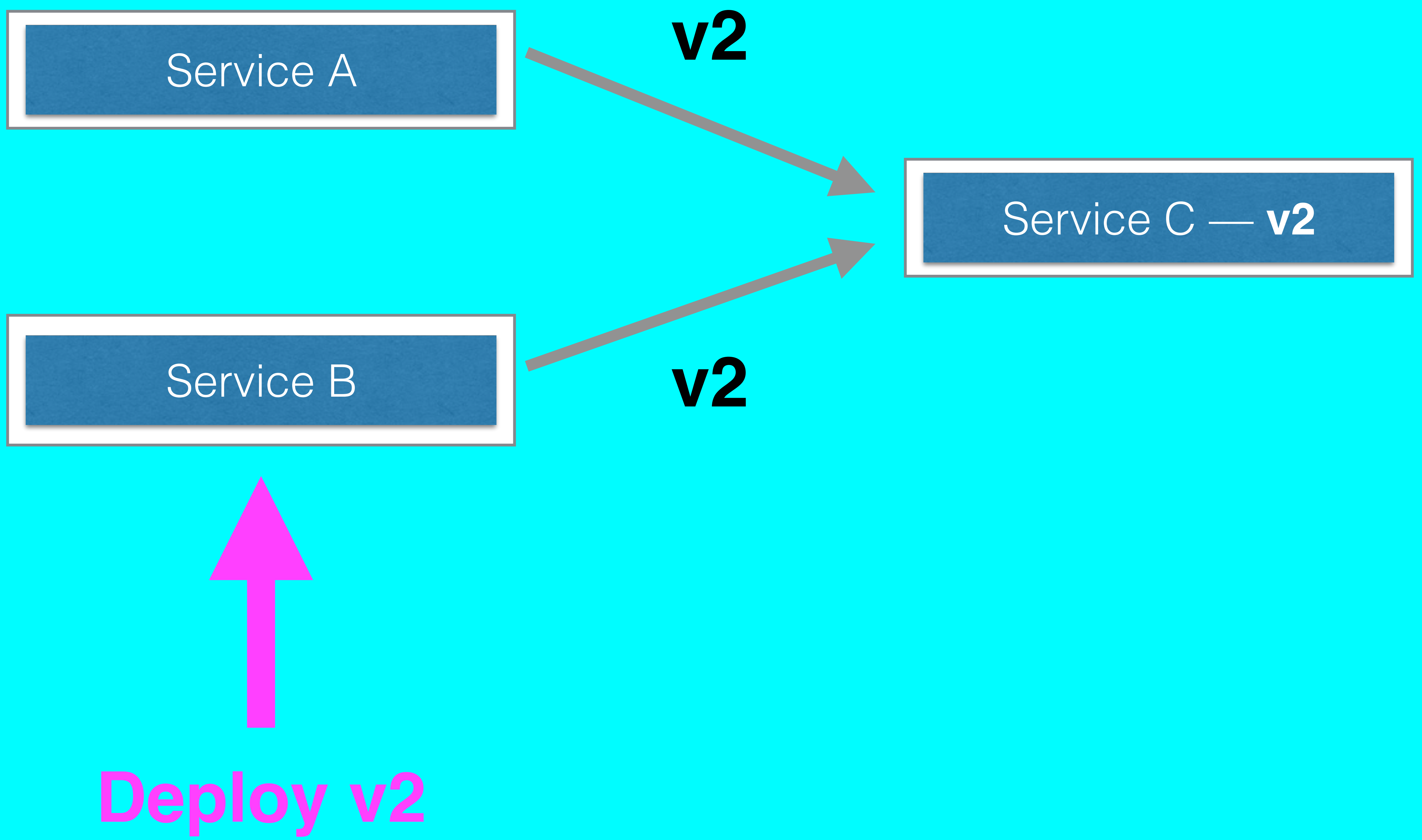
Lock-step deployment



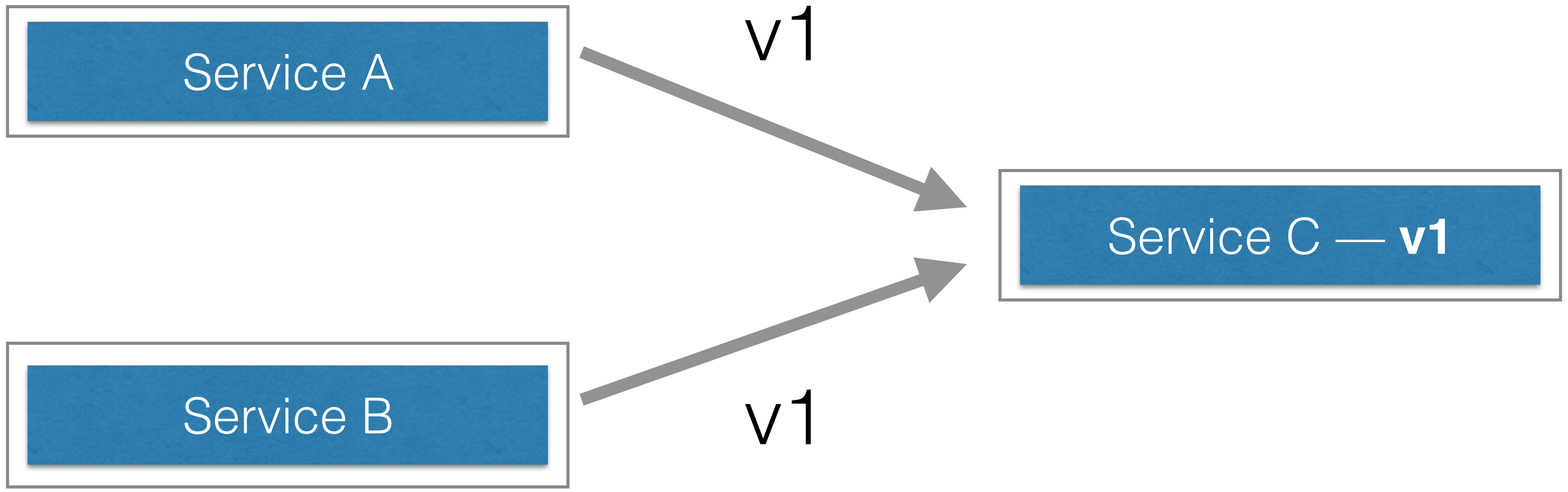
Lock-step deployment



Lock-step deployment



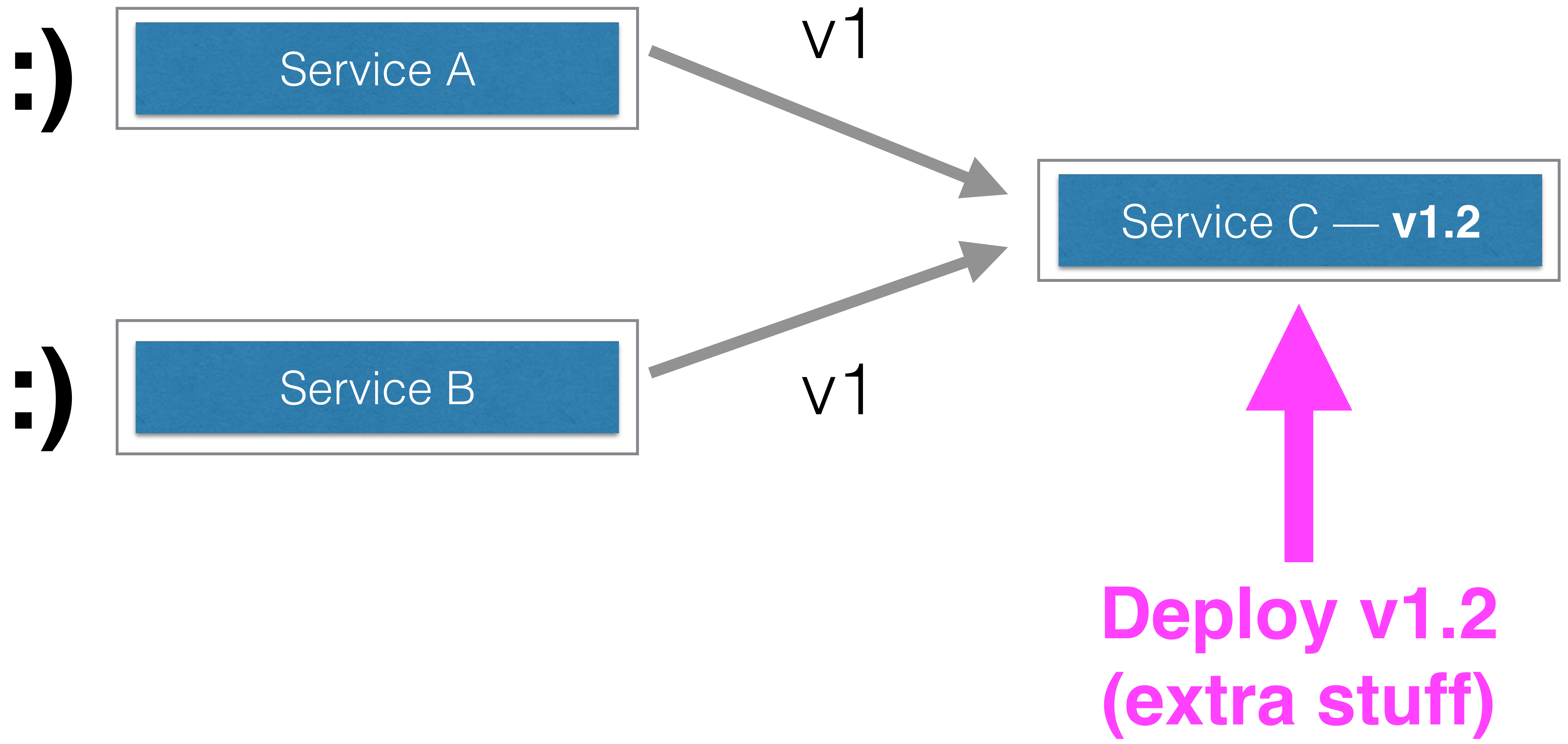
Solution: Semantic Versioning



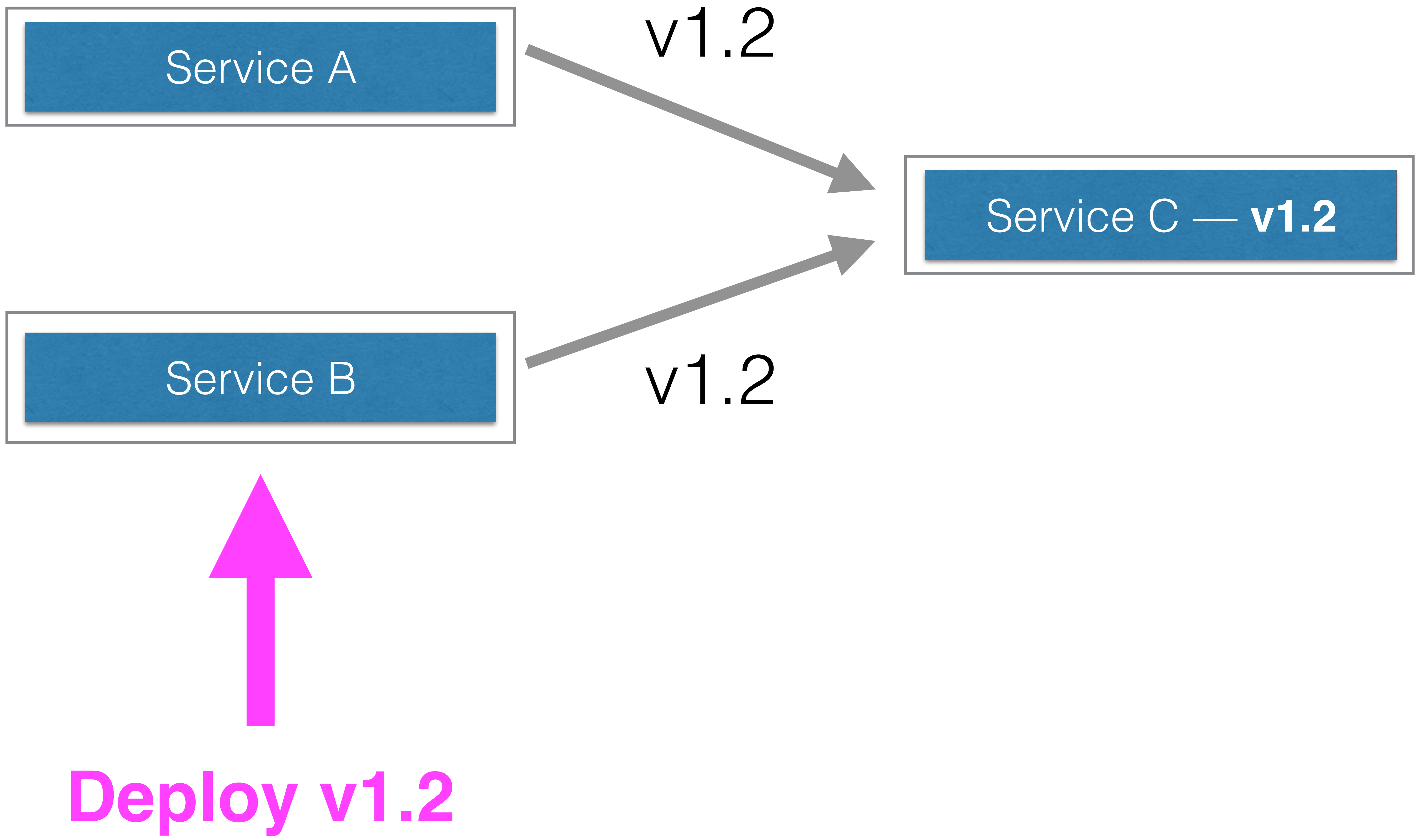
vMajor.Minor.Patch

MYBAD.SHINY.OOPS

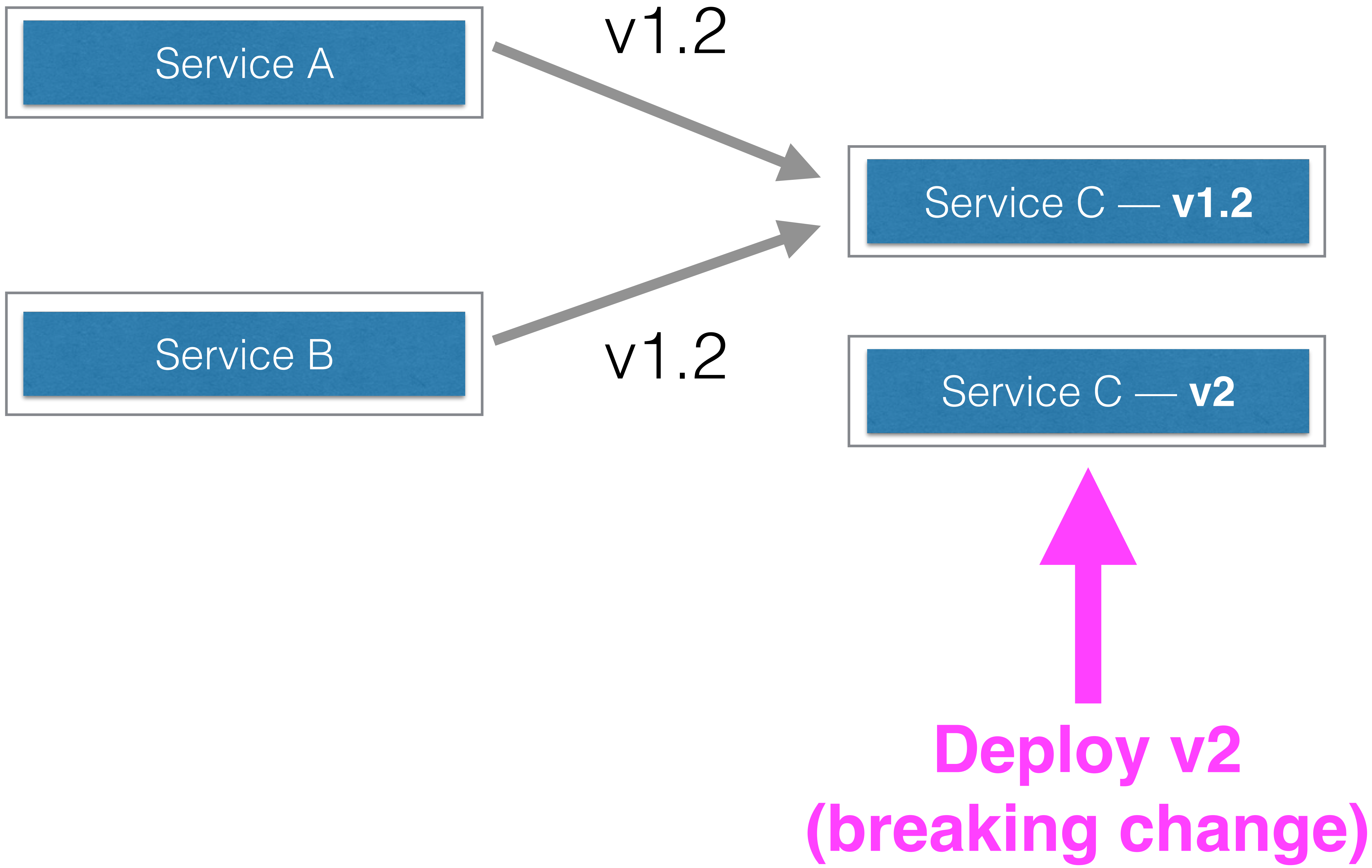
Solution: Semantic Versioning



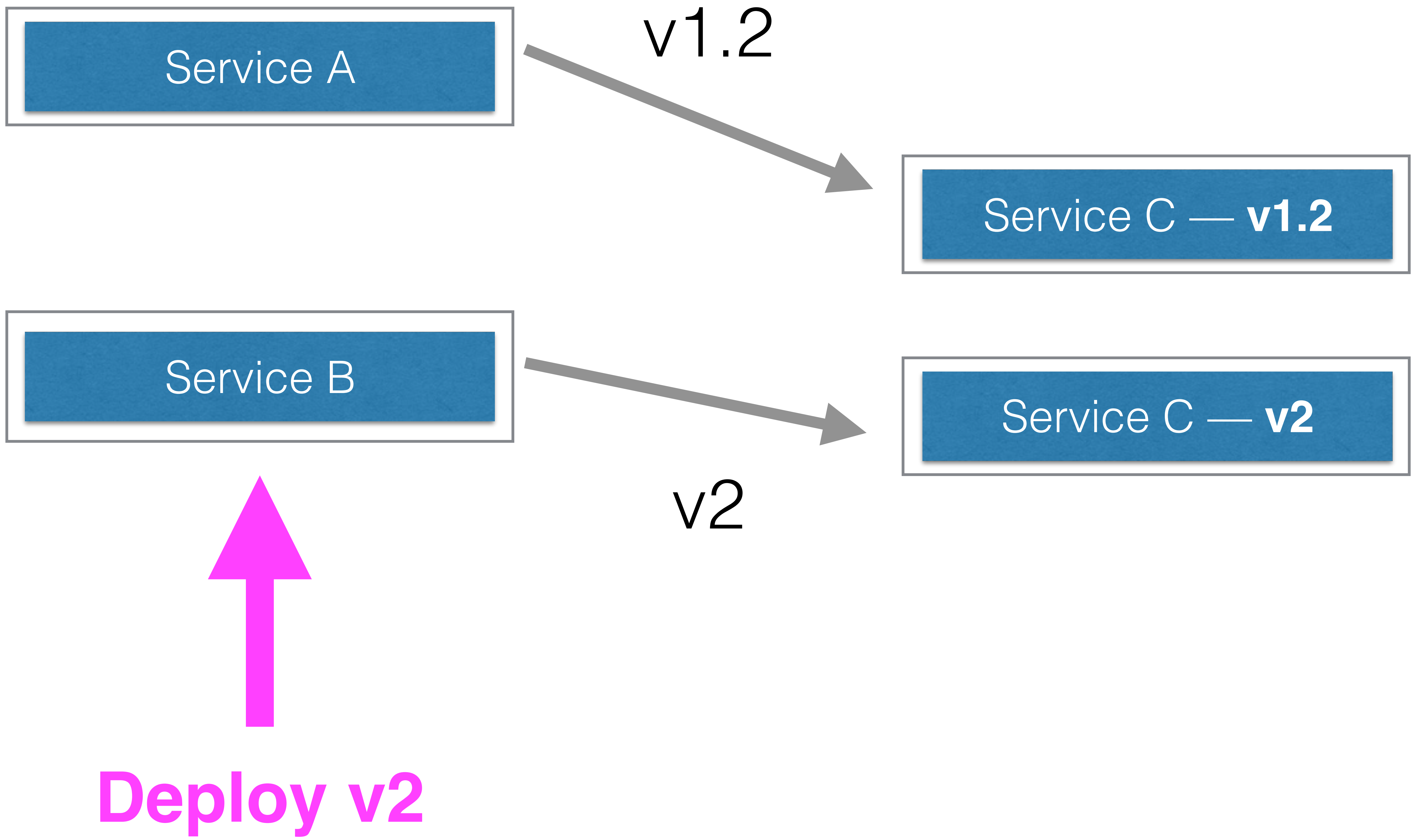
Solution: Semantic Versioning



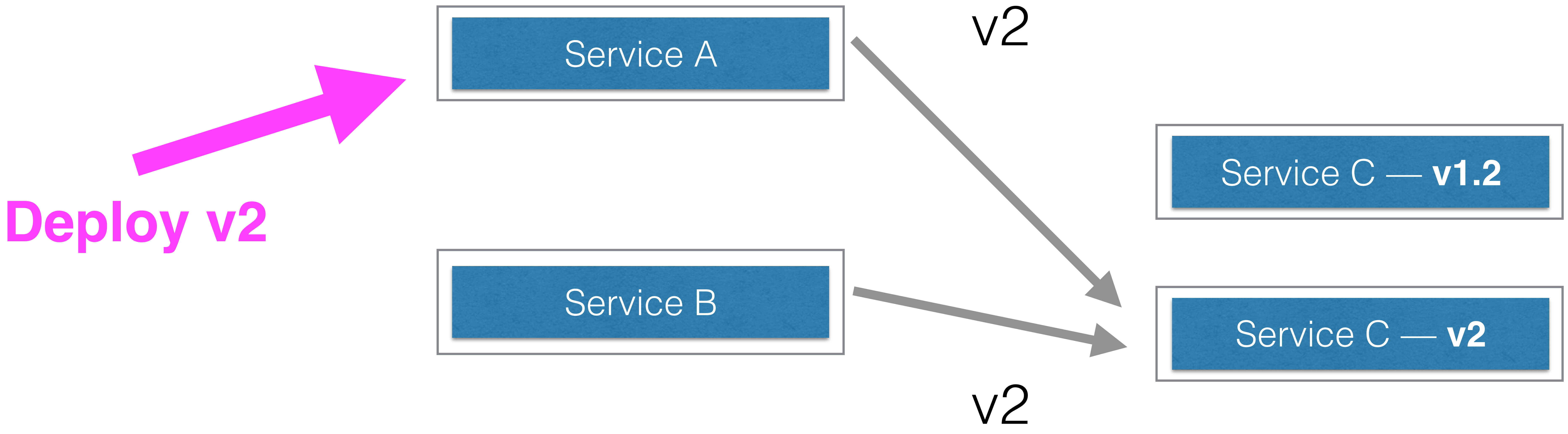
Solution: Semantic Versioning



Solution: Semantic Versioning



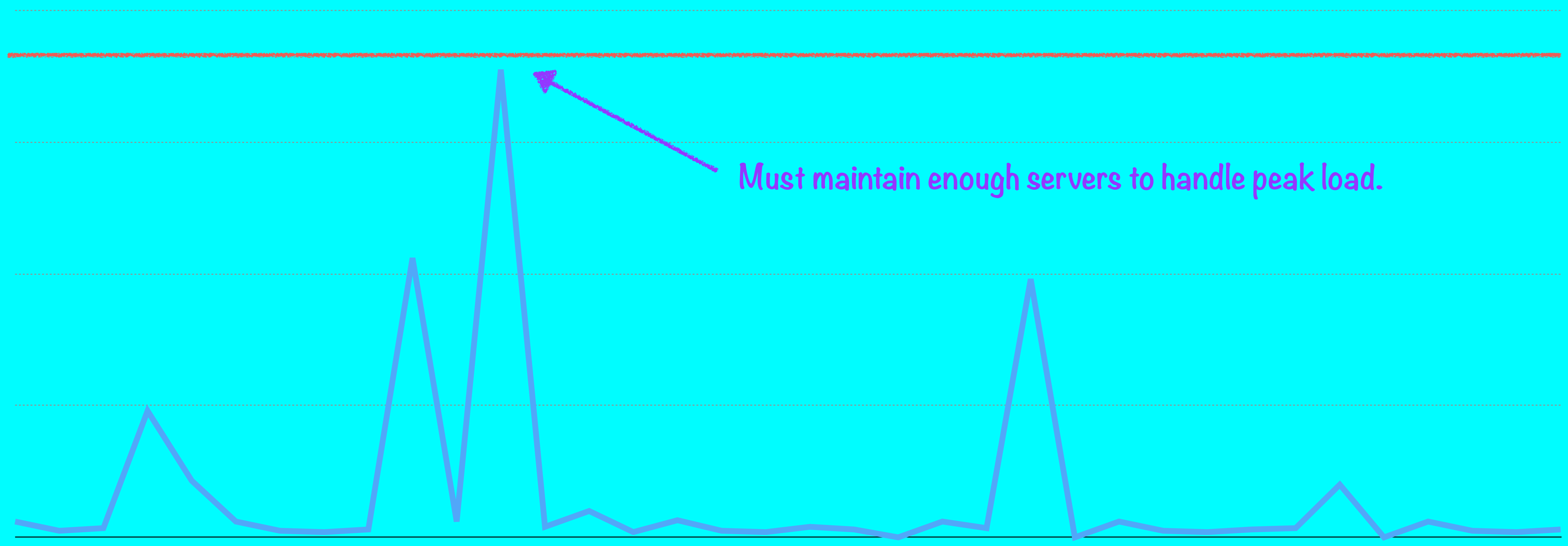
Solution: Semantic Versioning



OMG **ALL THE STEPS!!!**

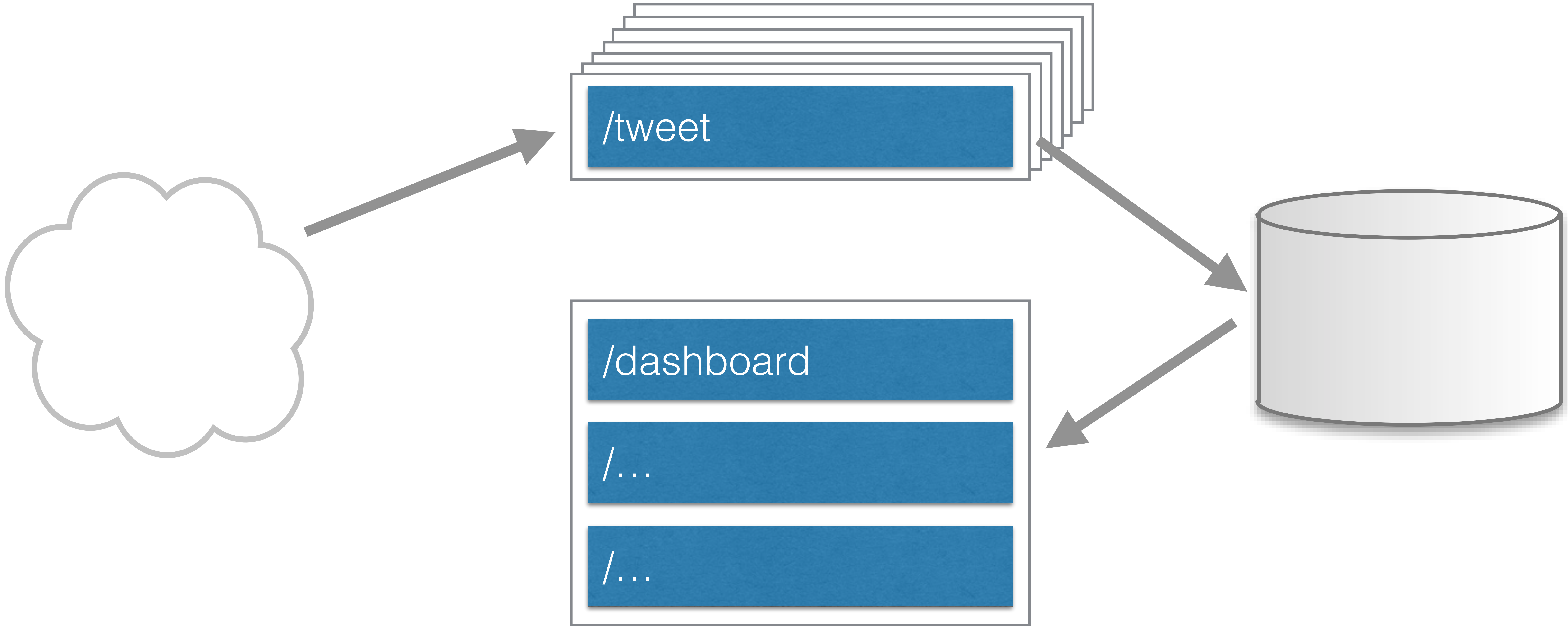
See Rule #1

Spiky load between services

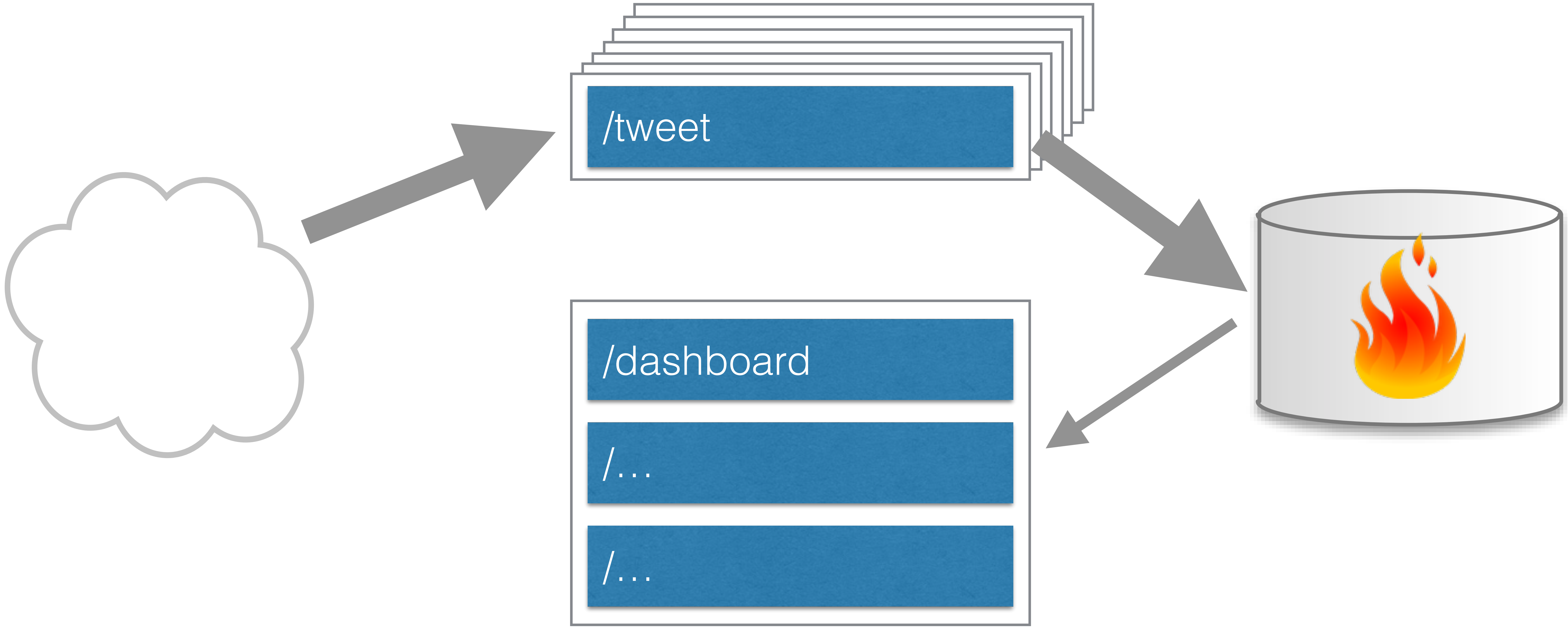


Must maintain enough servers to handle peak load.

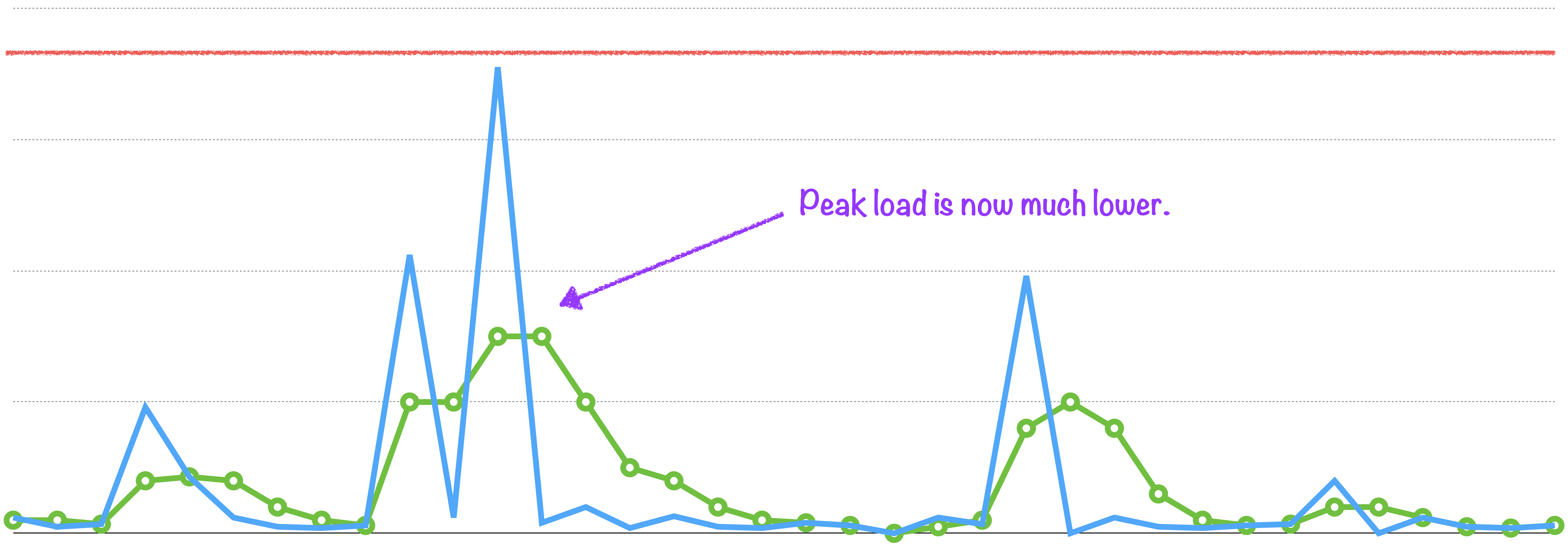
Spiky load between services



Spiky load between services

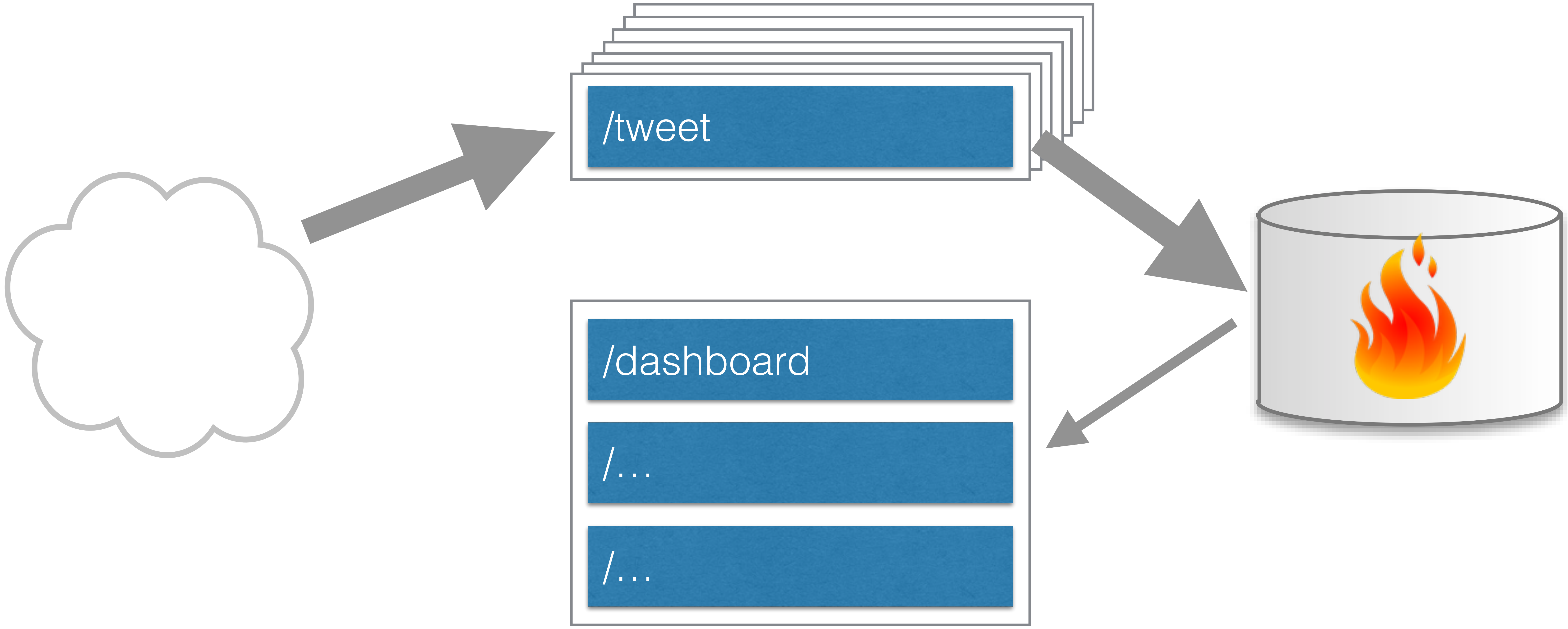


Solution: Amortize via queues



Queues in between services provide buffers that **smooth traffic**.

Solution: Amortize via queues



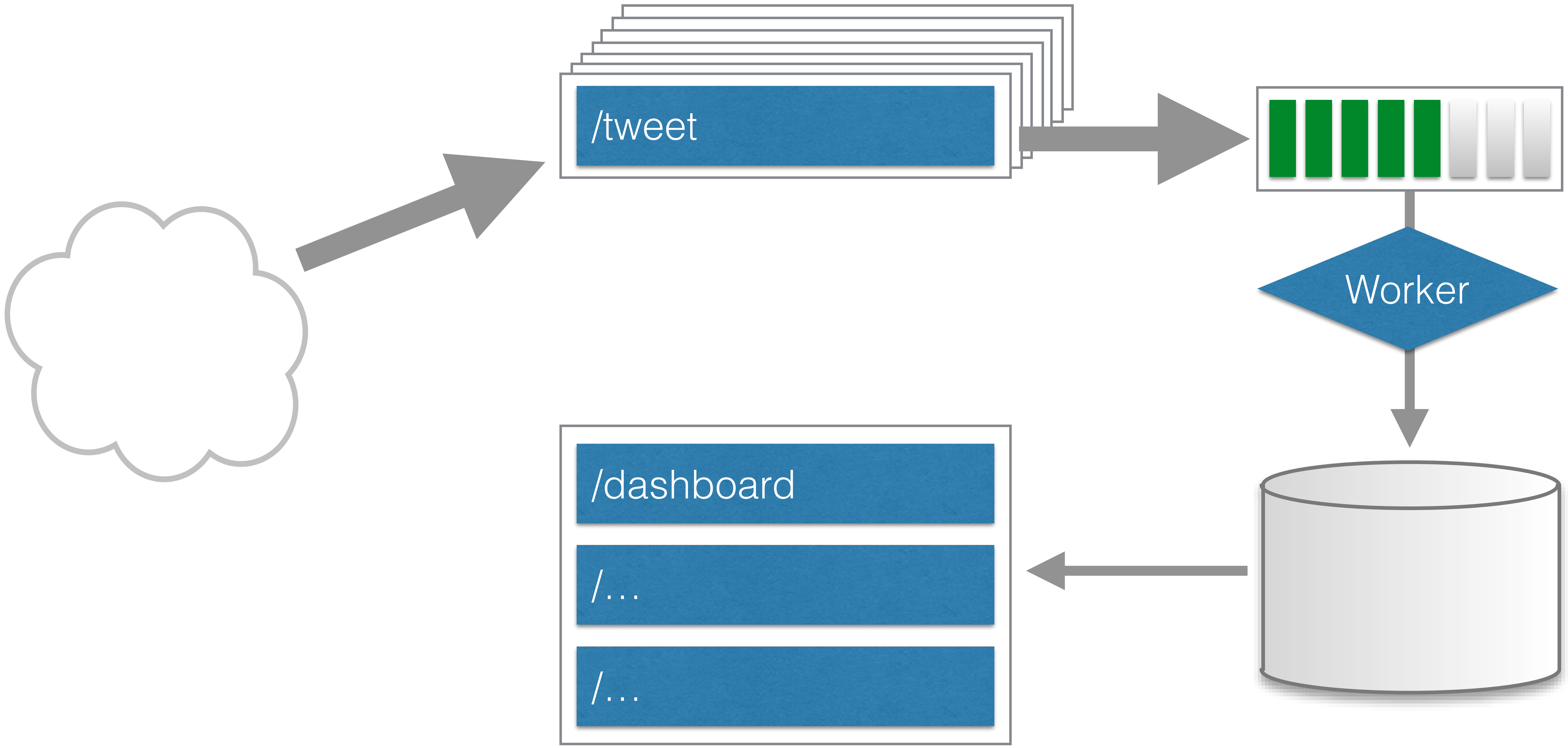
/tweet

/dashboard

/…

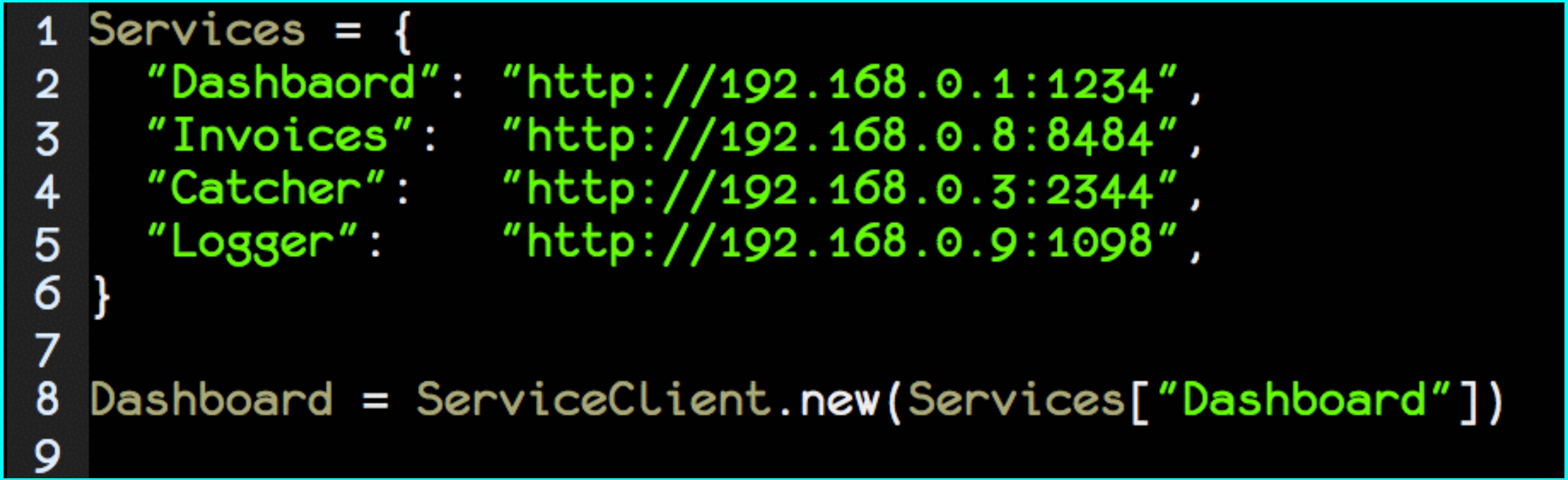
/…

Solution: Amortize via queues



Complexity: Now clients must deal with asynchronous responses.

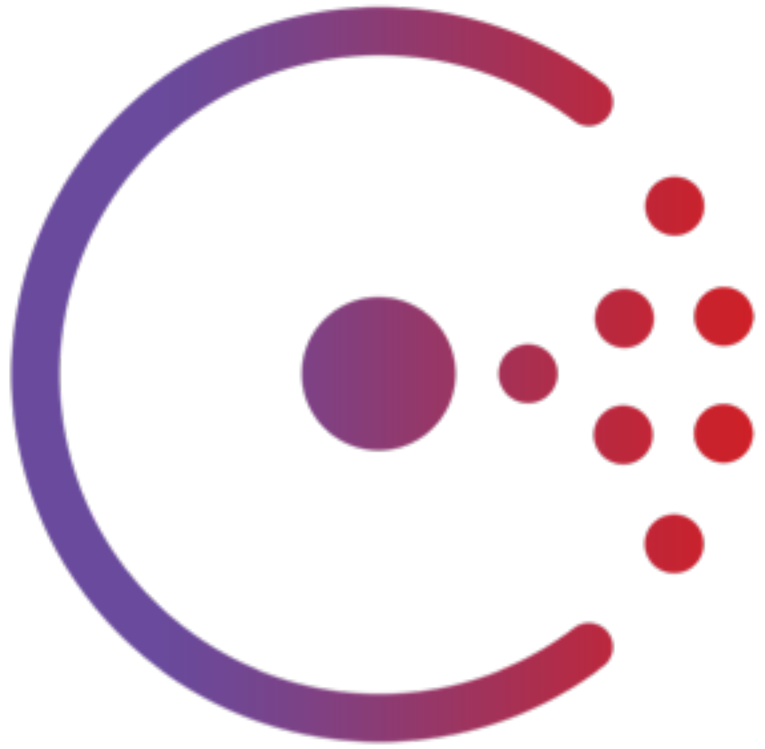
Hardcoded IPs and Ports



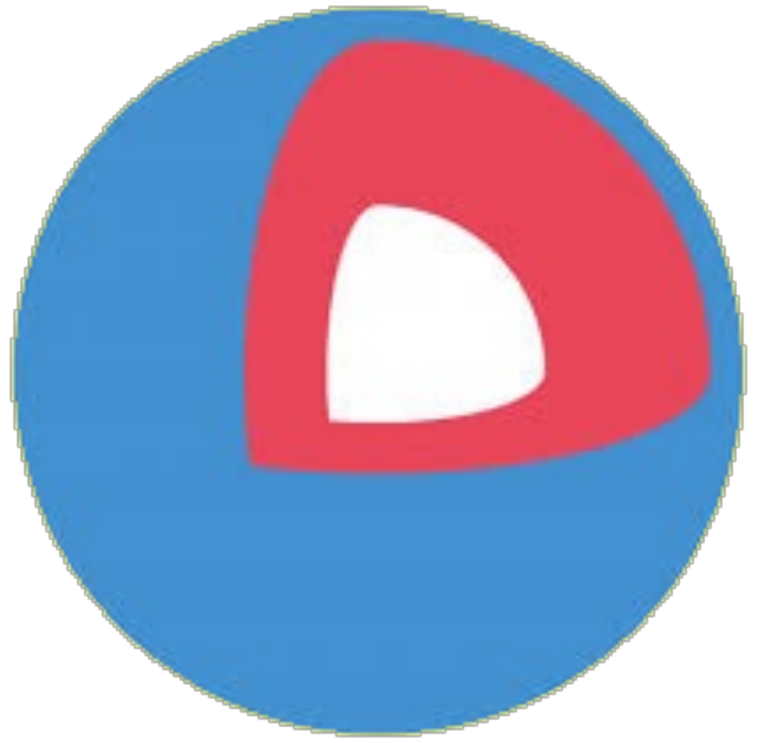
```
Services = {
   "Dashbaord": "http://192.168.0.1:1234",
   "Invoices":  "http://192.168.0.8:8484",
   "Catcher":   "http://192.168.0.3:2344",
   "Logger":    "http://192.168.0.9:1098",
}

Dashboard = ServiceClient.new(Services["Dashboard"])
```

Simple to get started, but immediately leads to deployment issues.

Solution 1: Discovery Service

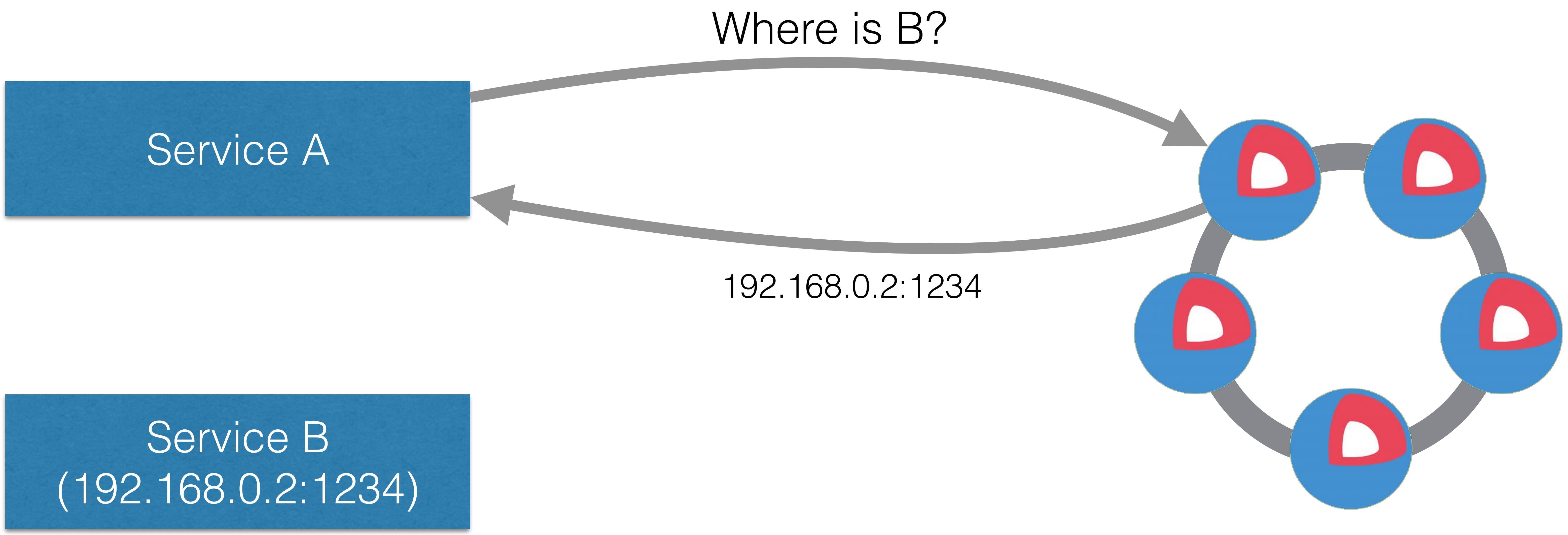


consul



etcd

Solution 1: Discovery Service



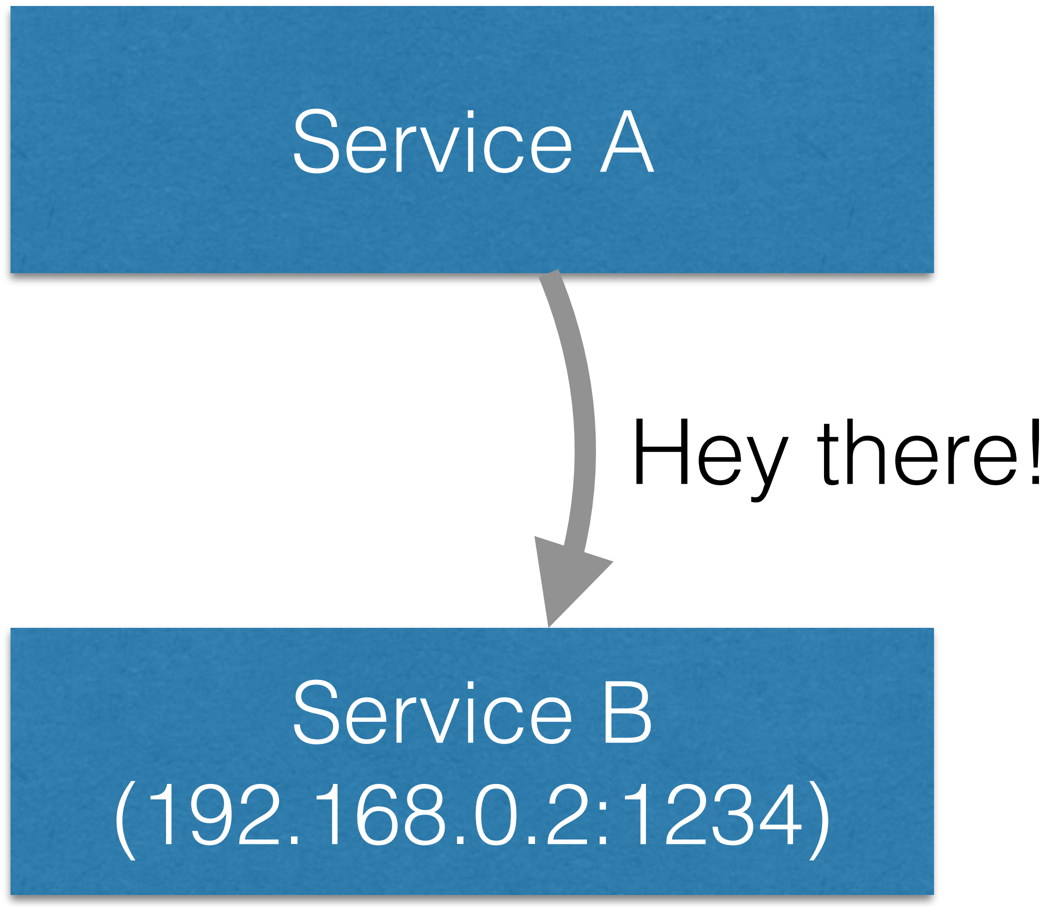
Service A

Service B
(192.168.0.2:1234)

Where is B?

192.168.0.2:1234

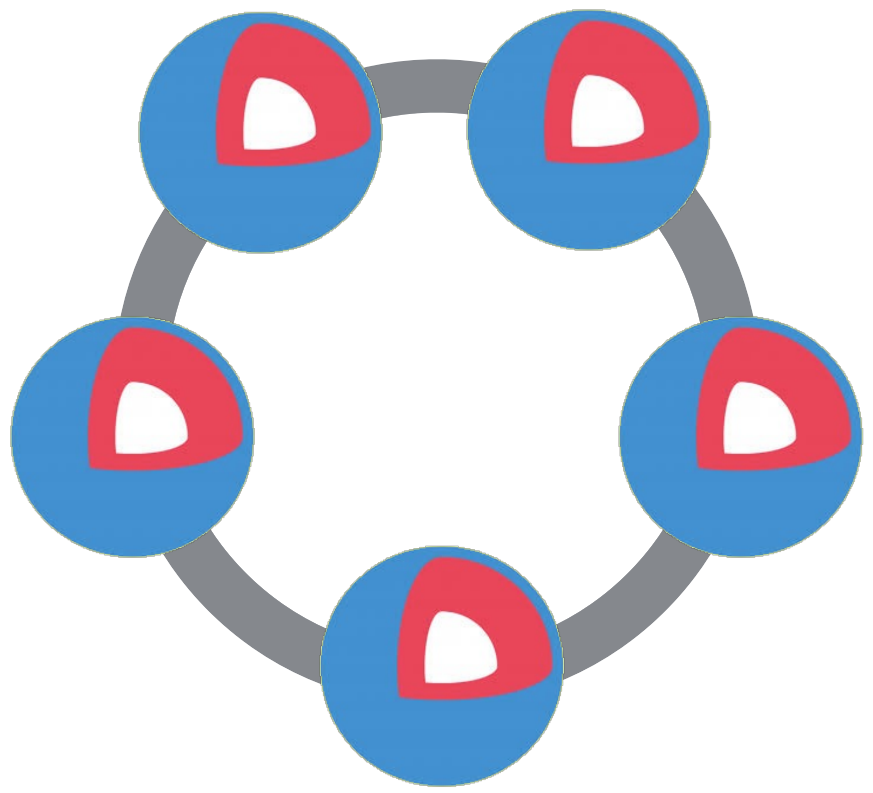
Solution 1: Discovery Service



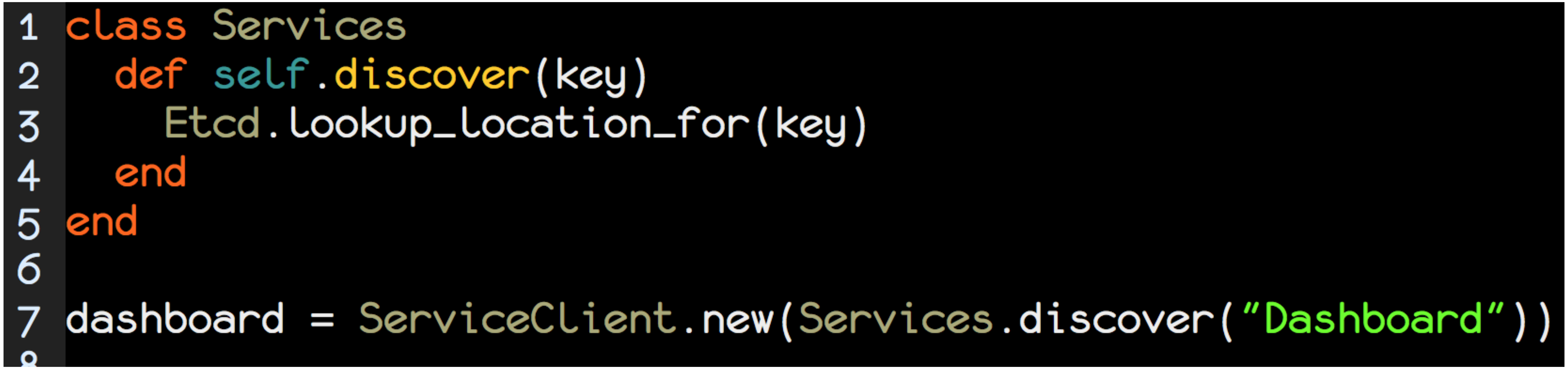
Service A

Hey there!

Service B
(192.168.0.2:1234)



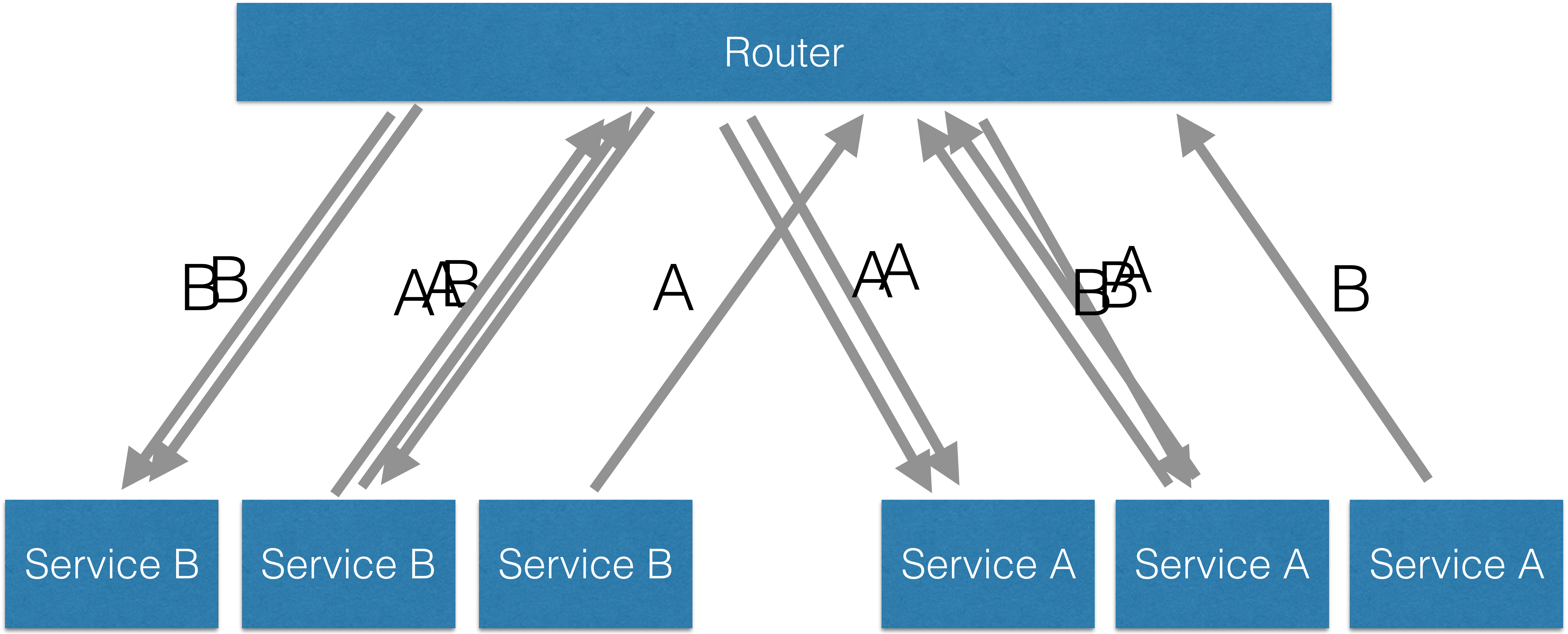
Solution 1: Discovery Service



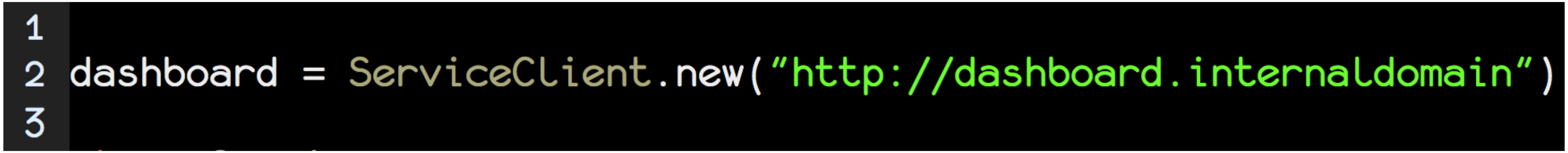
```ruby
class Services
  def self.discover(key)
    Etcd.lookup_location_for(key)
  end
end

dashboard = ServiceClient.new(Services.discover(“Dashboard”))
```

Complexity: Your code must understand the service lookup system.

Solution 2: Centralised router



Solution 2: Centralised router



```

dashboard = ServiceClient.new(“http://dashboard.internaldomain”)

```

Simplicity: “It’s just DNS.”

Router vs Discovery Service

Both require service **registration.**

Both require **HA** and **scalability**.

Router is **transparent**.

Router can be **exposed externally**.

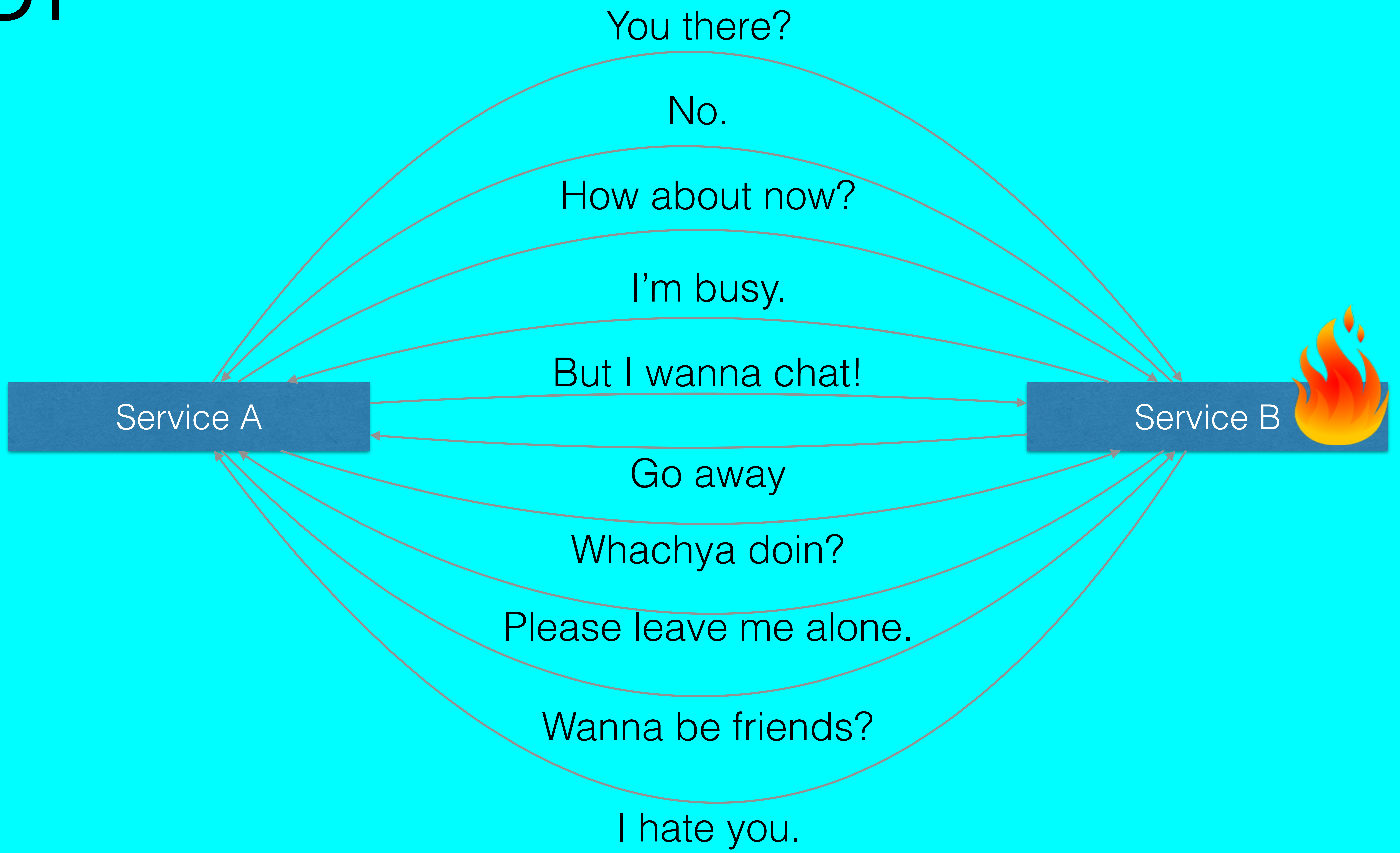
Router can **cache** transparently.

Router can **round-robin**.

Discovery Service is **simpler to build and scale**, since it doesn’t need to route all data.

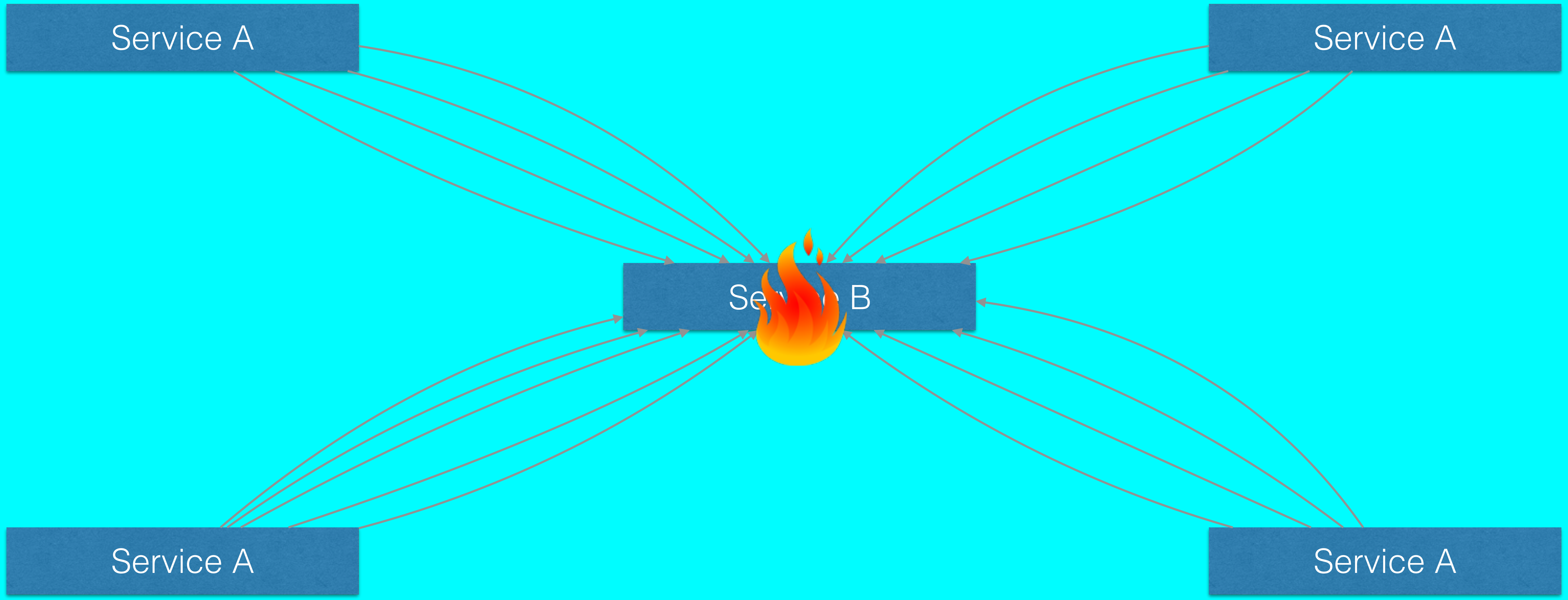
Discovery service does **fewer network hops**.

Dogpiles

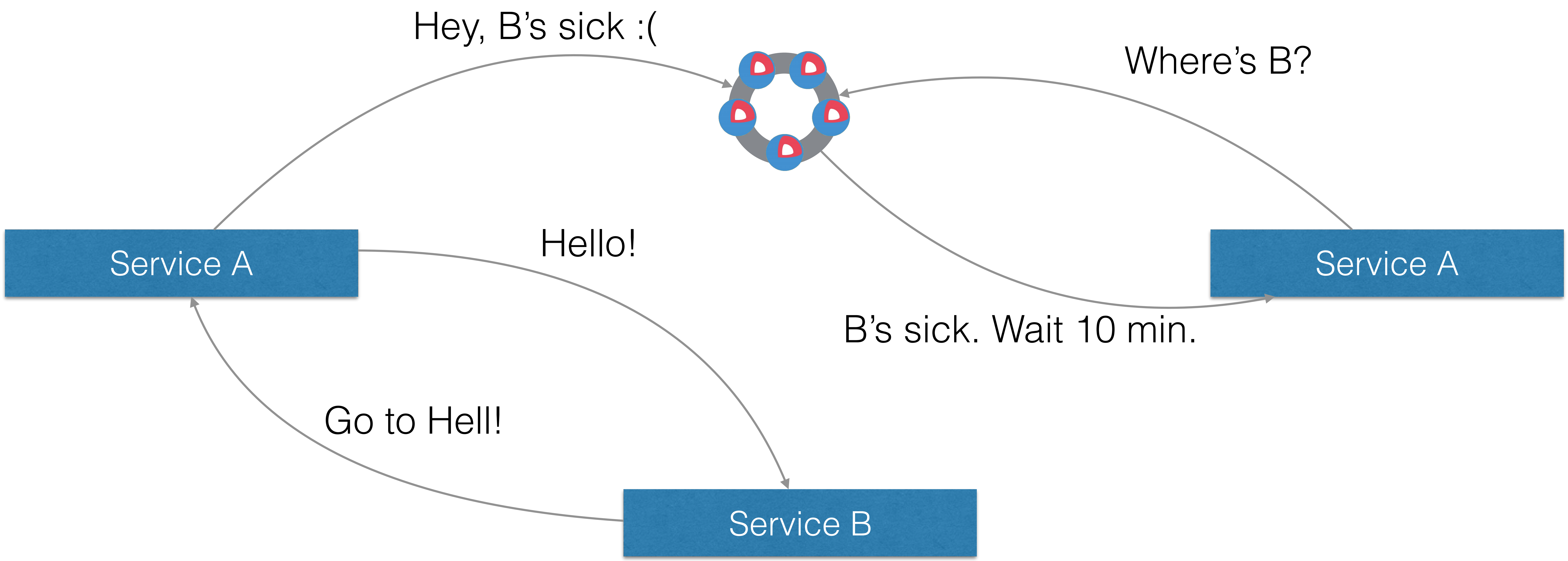


You there?
No.
How about now?
I'm busy.
But I wanna chat!
Service A
Service B
Go away
Whachya doin?
Please leave me alone.
Wanna be friends?
I hate you.

Dogpiles **x 100**



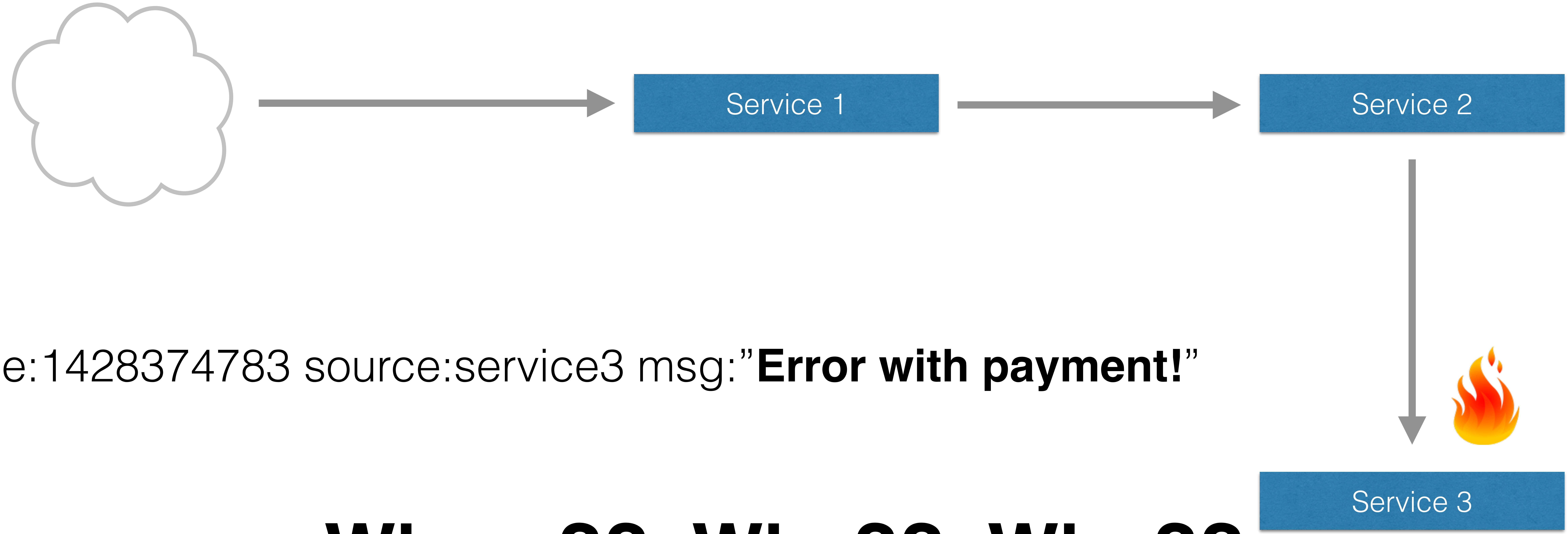
Solution: Circuit Breaker



Debugging hell

Turns out, distributed systems are hard.

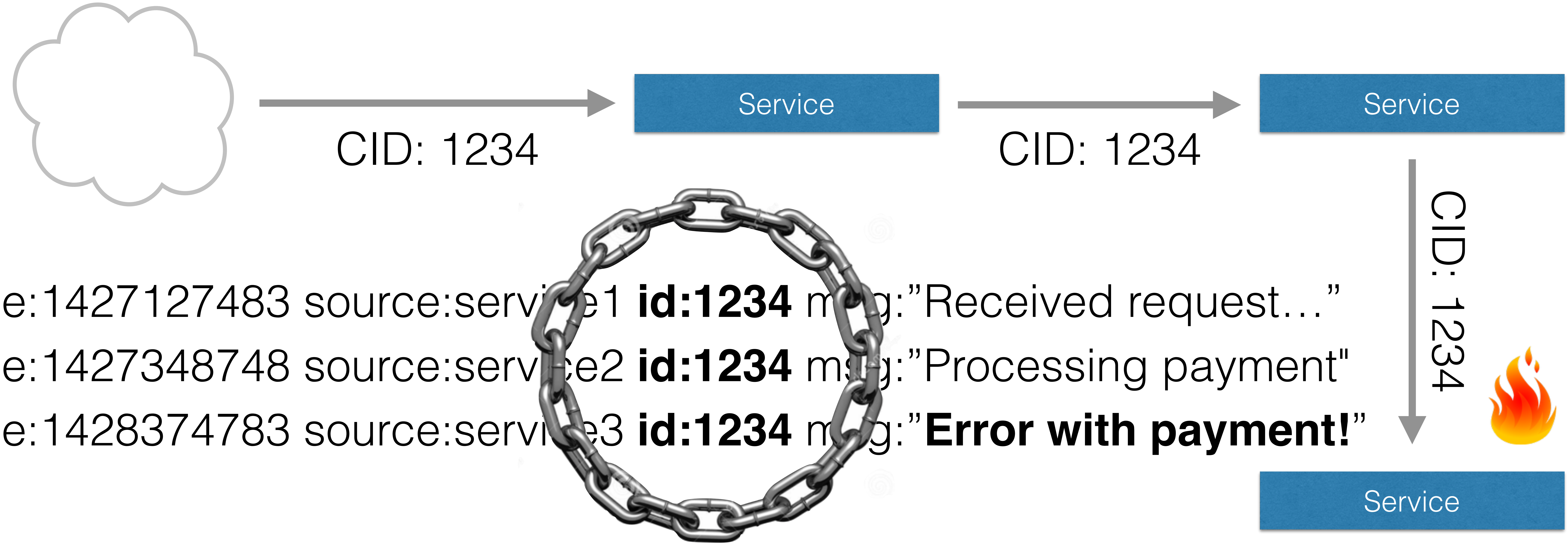
Debugging Hell



time:1428374783 source:service3 msg:”**Error with payment!**”

Where?? Why?? Who??

Solution: Correlation IDs



time:1427127483 source:service1 **id:1234** msg:”Received request…”

time:1427348748 source:service2 **id:1234** msg:”Processing payment"

time:1428374783 source:service3 **id:1234** msg:”**Error with payment!**”

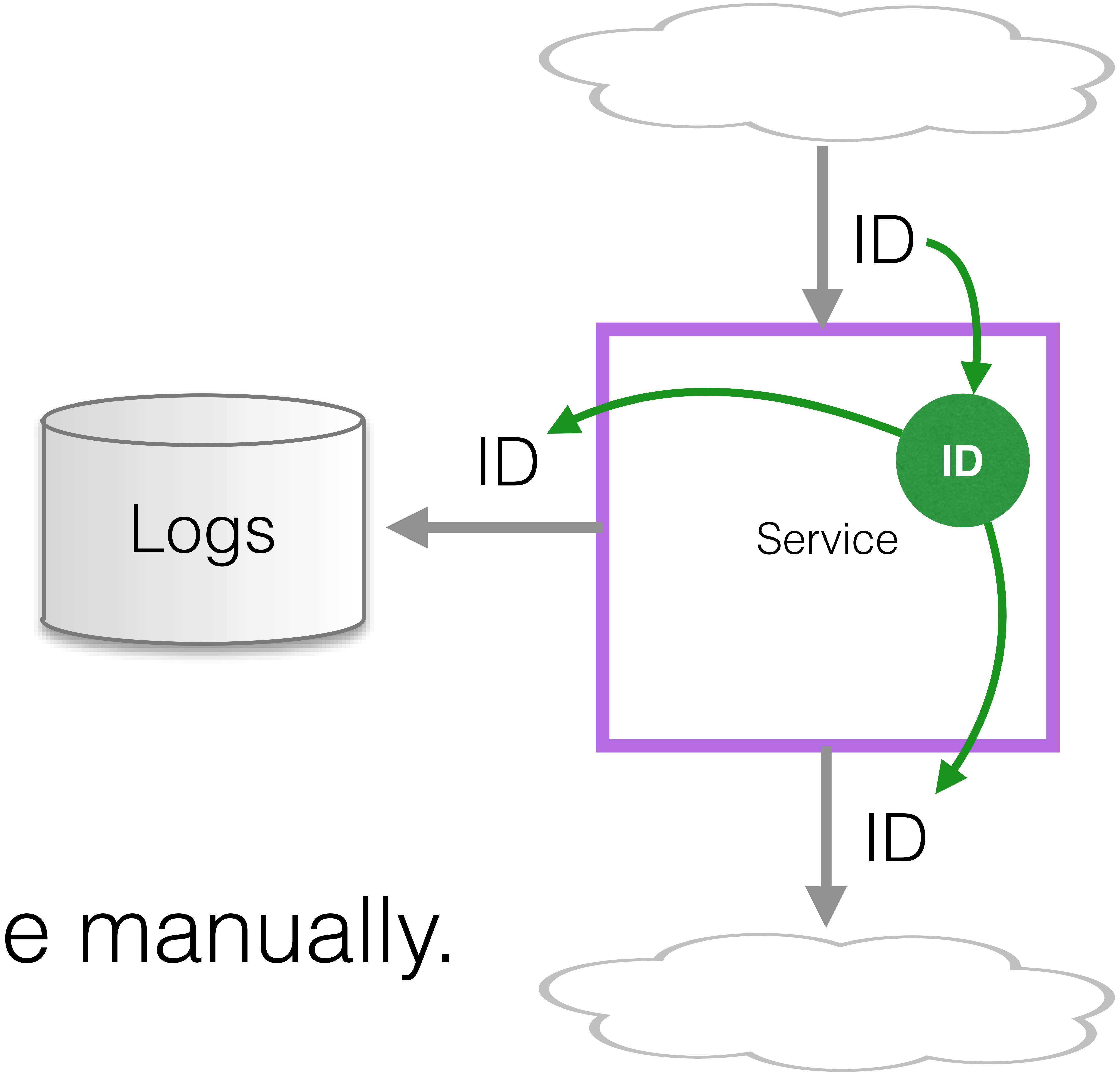
Solution: Correlation IDs

1. Tag all incoming requests with unique ID

2. Service saves ID for all incoming requests

3. Include that ID in all log lines, etc.

4. Tag *new* requests with that ID

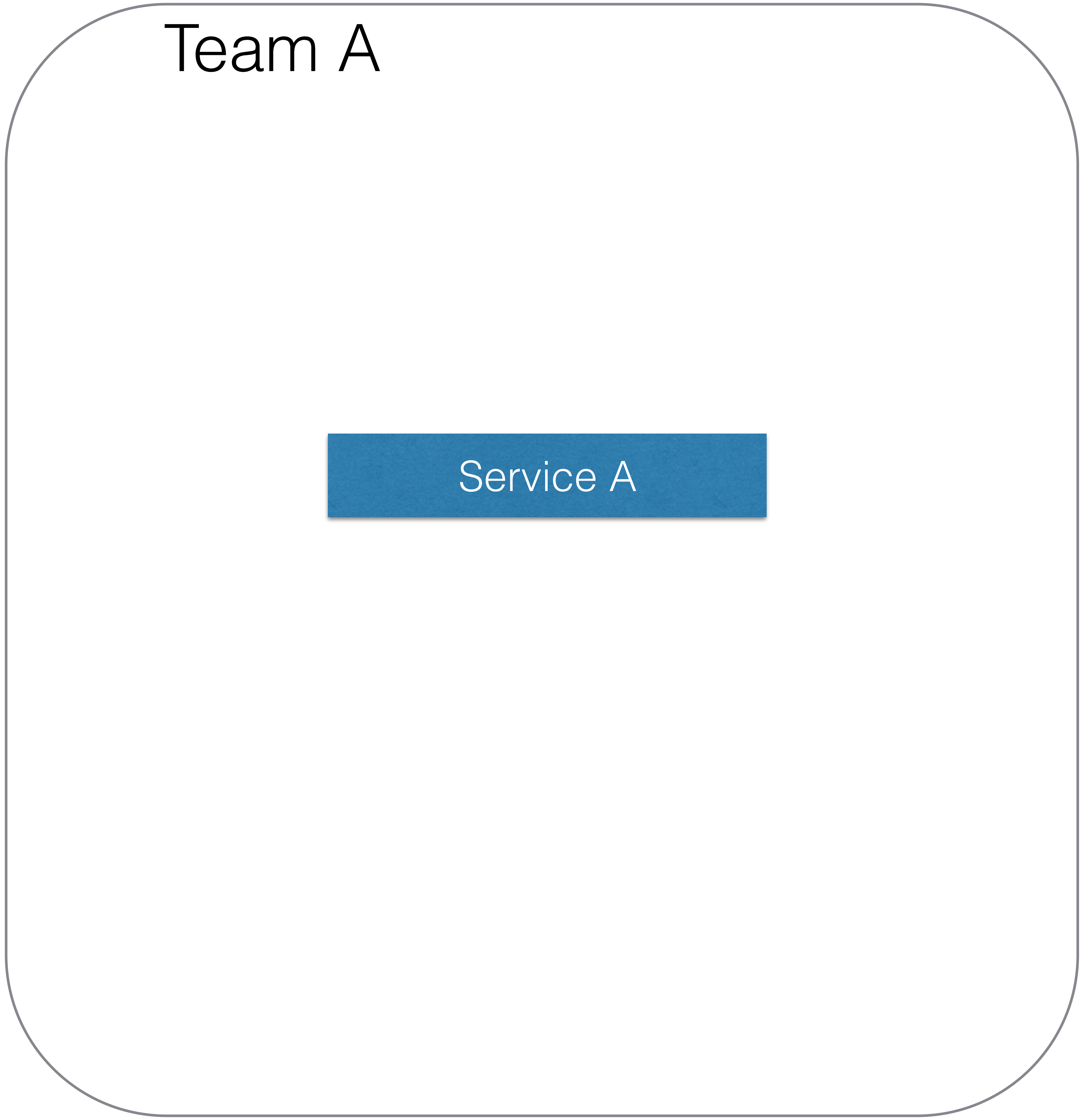


Complexity: Must be done manually.

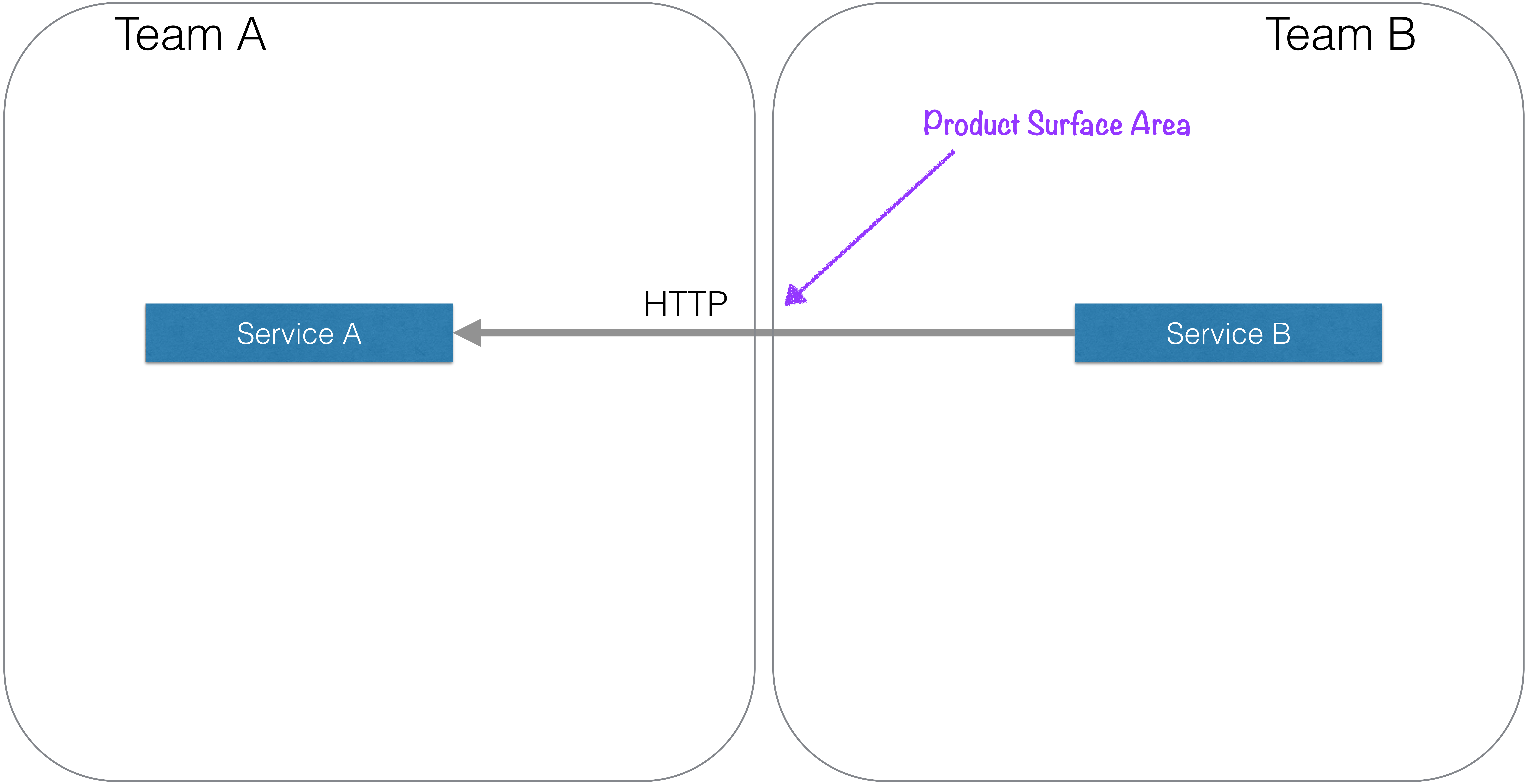
Missing Mock Servers

Each consuming team has to create
their own mocks and stubs.

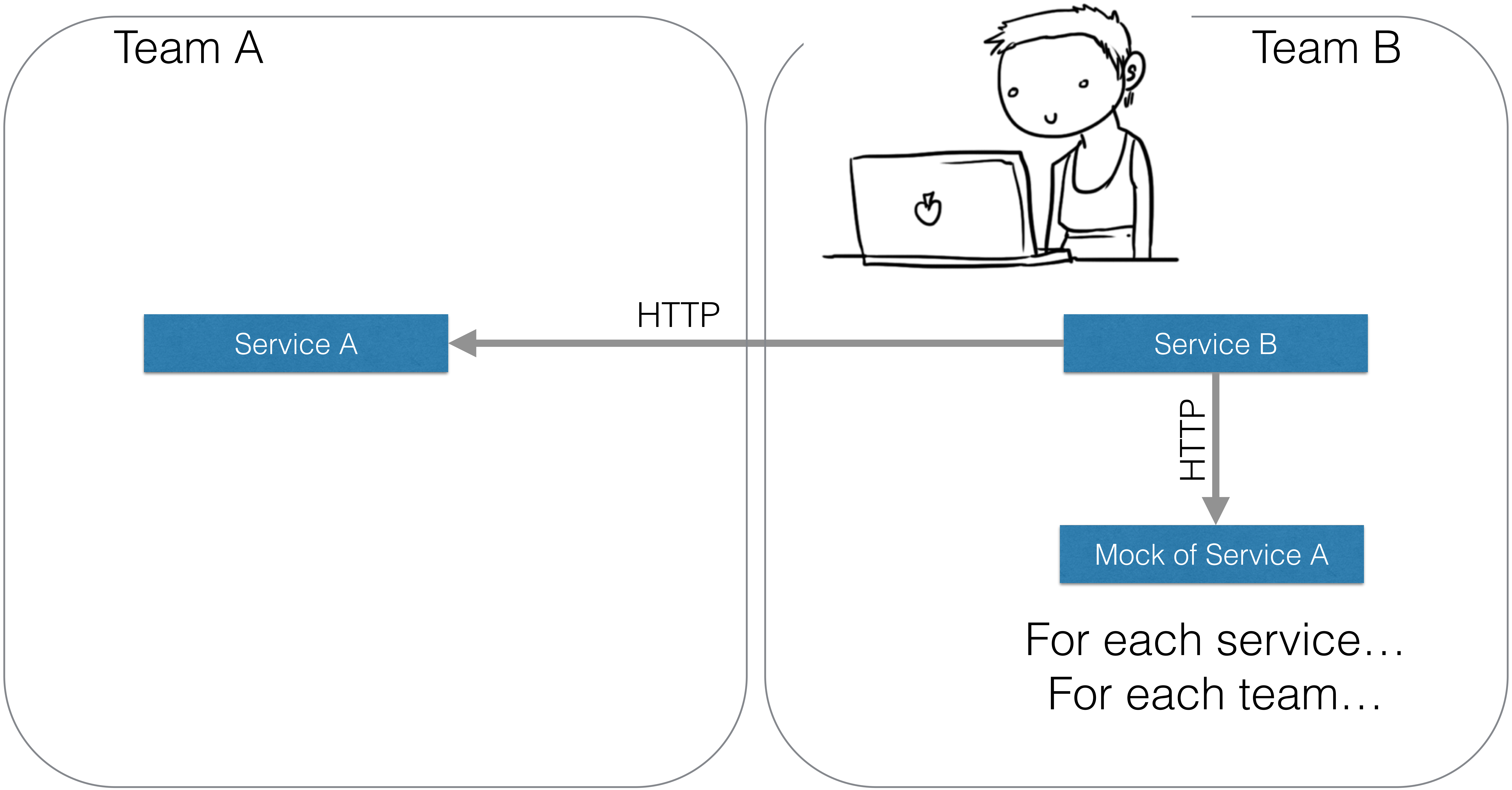
Missing Mock Servers



Missing Mock Servers



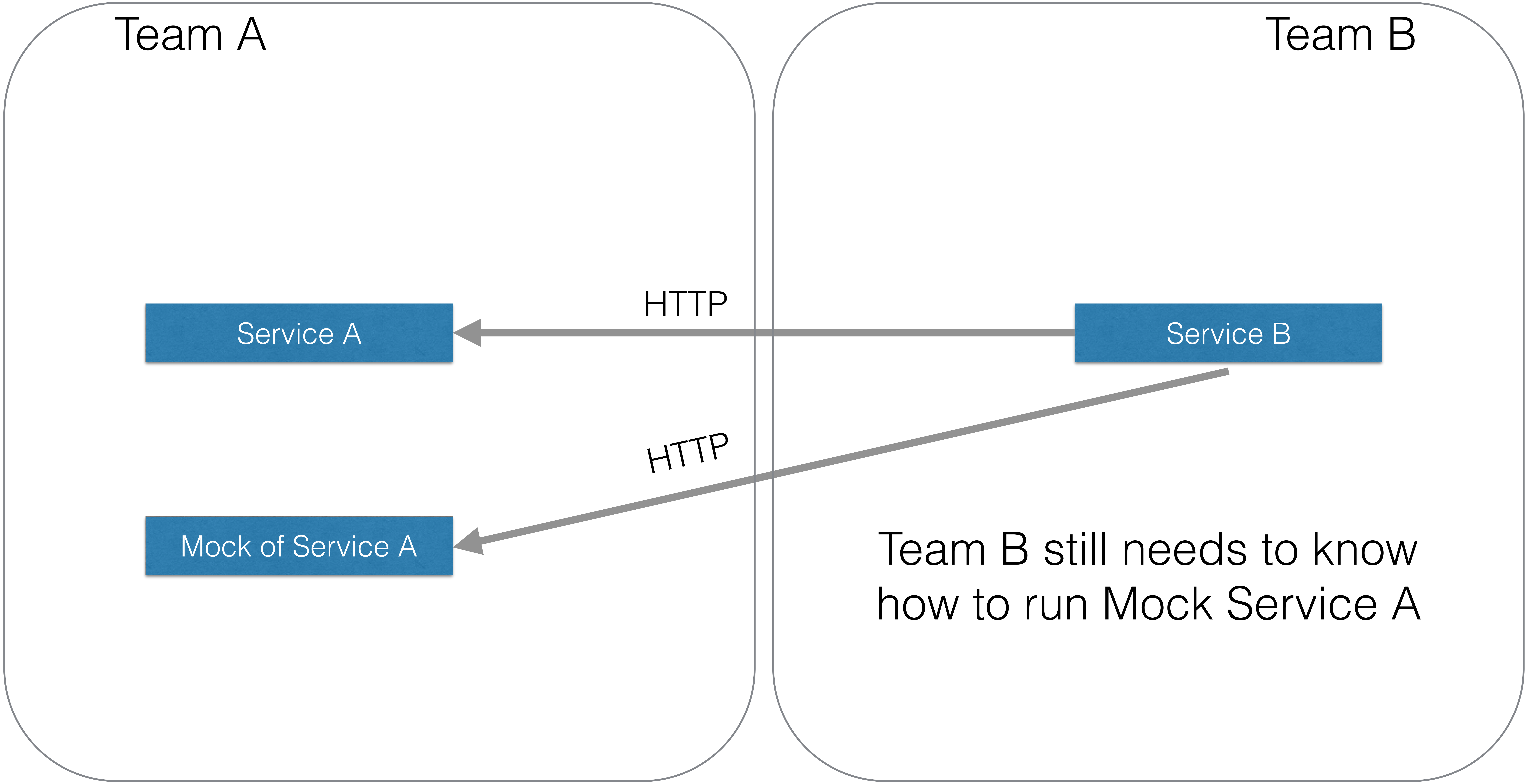
Missing Mock Servers



For each service…
For each team…

Solution: Service Team Provides the Mock

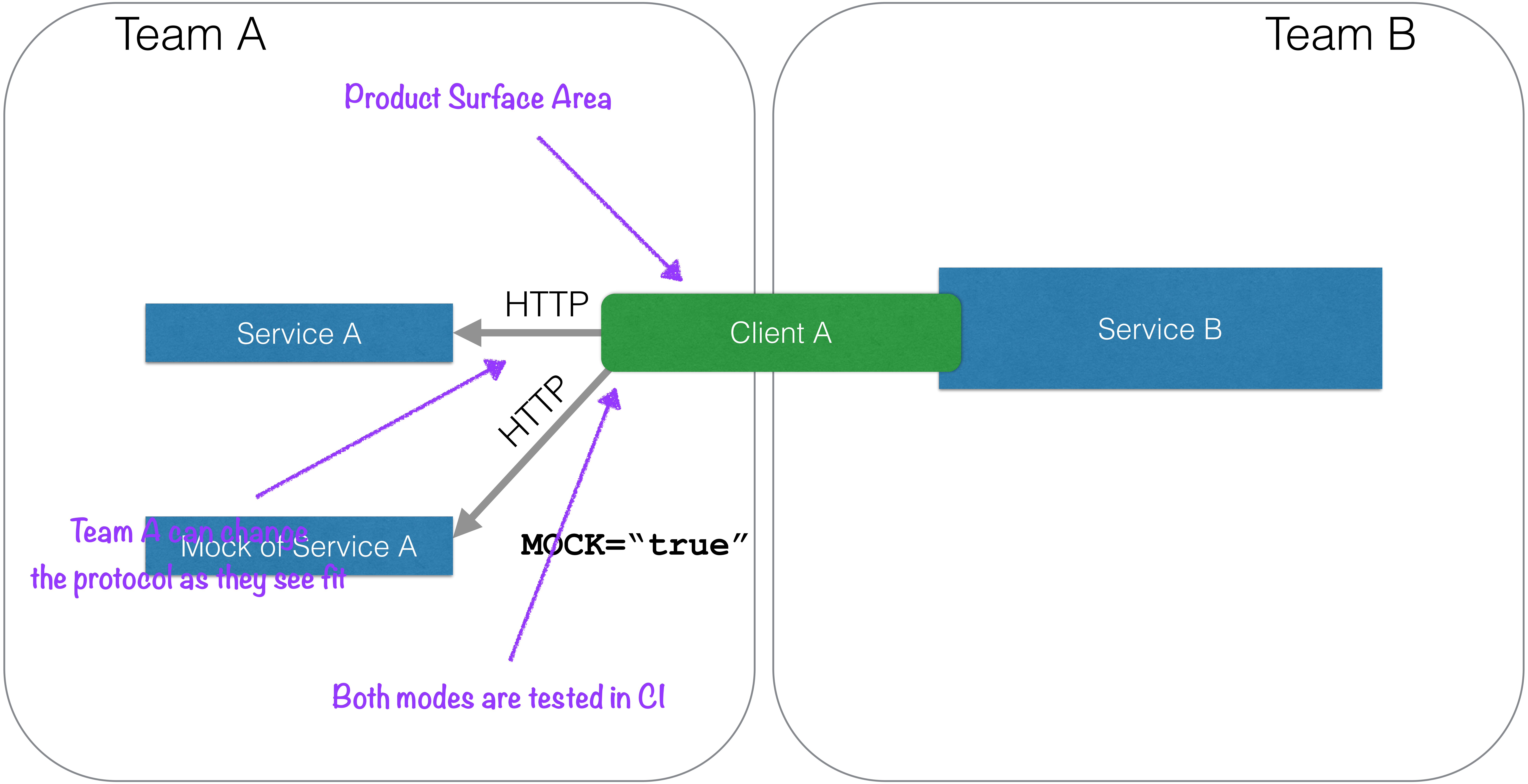
Better…



Team B still needs to know how to run Mock Service A

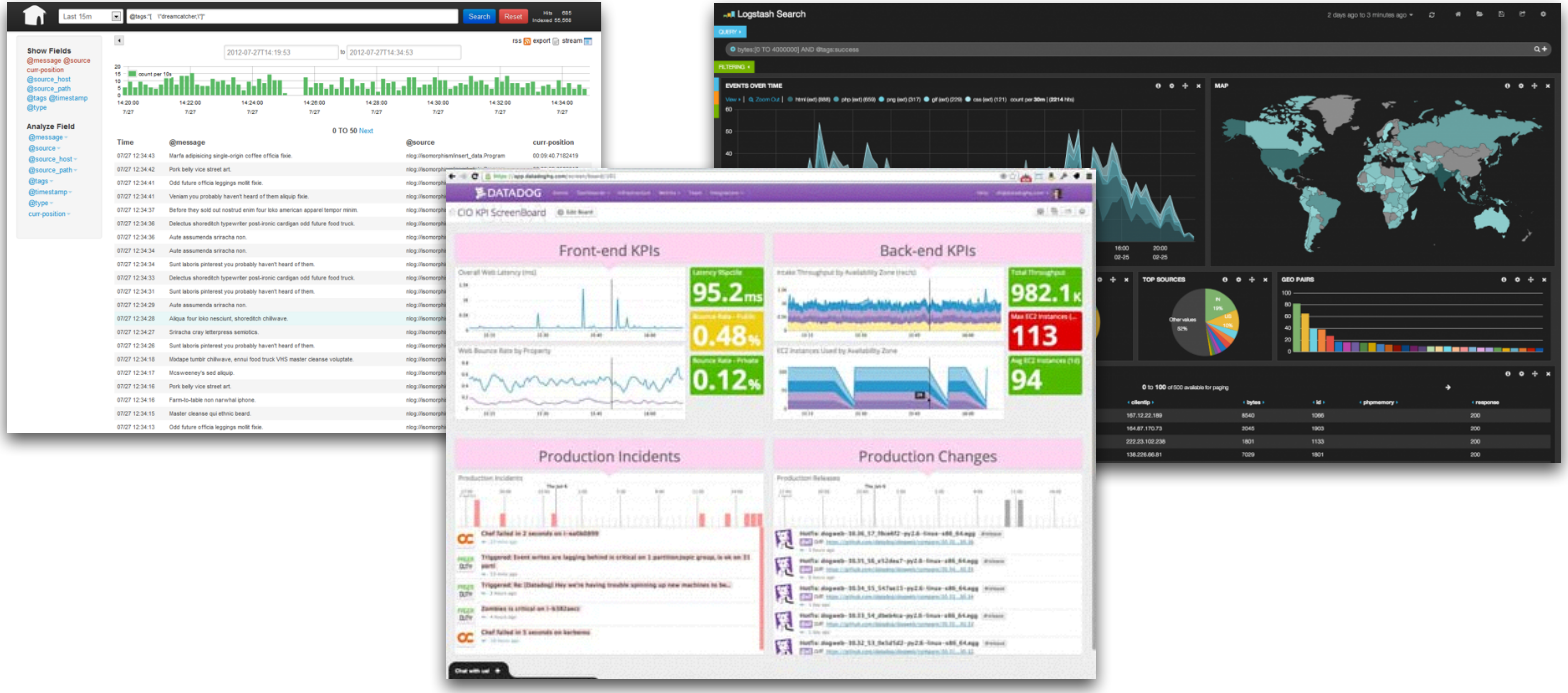
Solution: Service Team Owns the Client

Best...



Flying Blind

Solution: Graphs, alerts, pages.

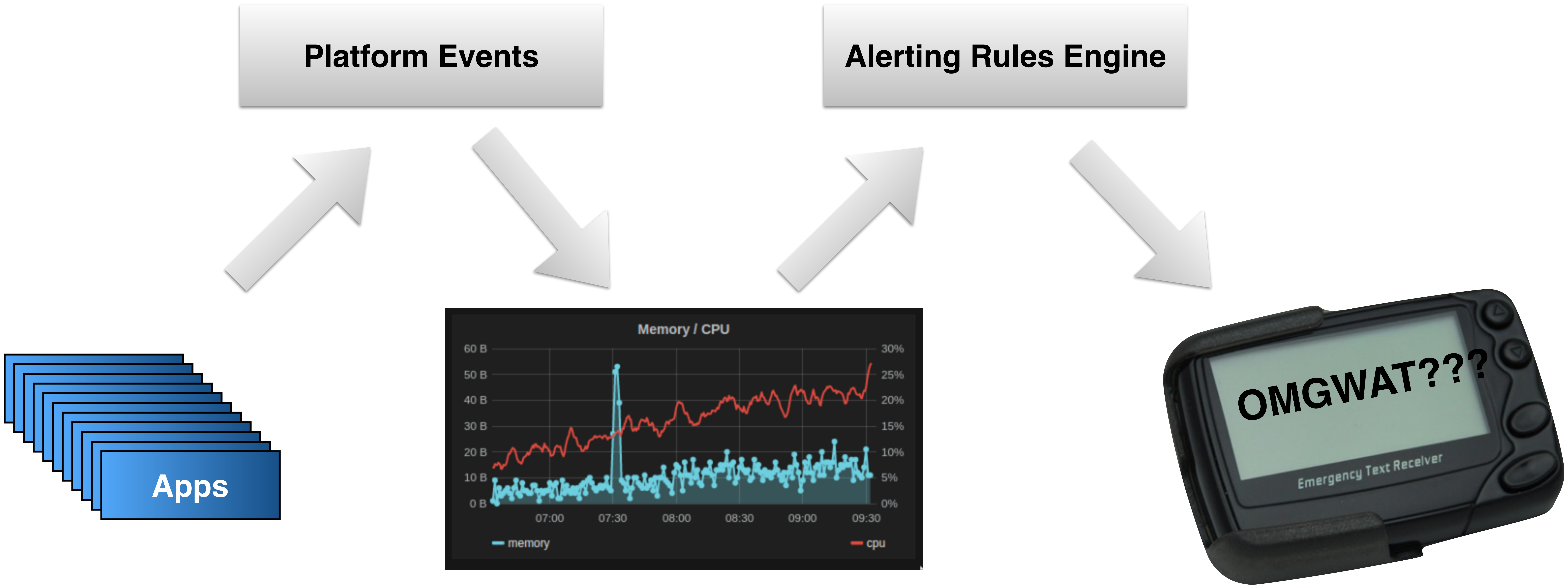


Time	@message	@source	curr-position
07/27 12:34:43	Marfa adipisicing single-origin coffee officia fixie.	nlog://isomorphism/insert_data.Program	00:09:40.7182419
07/27 12:34:42	Pork belly vice street art.	nlog://isomorphi	
07/27 12:34:41	Odd future officia leggings mollit fixie.	nlog://isomorphi	
07/27 12:34:41	Veniam you probably haven't heard of them aliquip fixie.	nlog://isomorphi	
07/27 12:34:37	Before they sold out nostrud enim four loko american apparel tempor minim.	nlog://isomorphi	
07/27 12:34:36	Delectus shoreditch typewriter post-ironic cardigan odd future food truck.	nlog://isomorphi	
07/27 12:34:36	Aute assumenda sriracha non.	nlog://isomorphi	
07/27 12:34:34	Aute assumenda sriracha non.	nlog://isomorphi	
07/27 12:34:34	Sunt laboris pinterest you probably haven't heard of them.	nlog://isomorphi	
07/27 12:34:33	Delectus shoreditch typewriter post-ironic cardigan odd future food truck.	nlog://isomorphi	
07/27 12:34:31	Sunt laboris pinterest you probably haven't heard of them.	nlog://isomorphi	
07/27 12:34:29	Aute assumenda sriracha non.	nlog://isomorphi	
07/27 12:34:28	Aliqua four loko nesciunt, shoreditch chillwave.	nlog://isomorphi	
07/27 12:34:27	Sriracha cray letterpress semiotics.	nlog://isomorphi	
07/27 12:34:26	Sunt laboris pinterest you probably haven't heard of them.	nlog://isomorphi	
07/27 12:34:18	Mixtape tumblr chillwave, ennui food truck VHS master cleanse voluptate.	nlog://isomorphi	
07/27 12:34:17	Mcsweeney's sed aliquip.	nlog://isomorphi	
07/27 12:34:16	Pork belly vice street art.	nlog://isomorphi	
07/27 12:34:16	Farm-to-table non narwhal iphone.	nlog://isomorphi	
07/27 12:34:15	Master cleanse qui ethnic beard.	nlog://isomorphi	
07/27 12:34:13	Odd future officia leggings mollit fixie.	nlog://isomorphi	

0 to 100 of 500 available for paging

clientip	bytes	id	phpmemory	response
167.12.22.189	8540	1066		200
164.87.170.73	2045	1903		200
222.23.102.238	1801	1133		200
138.226.66.81	7029	1801		200

Solution: Graphs, alerts, pages.



Platform Events

Alerting Rules Engine

Apps

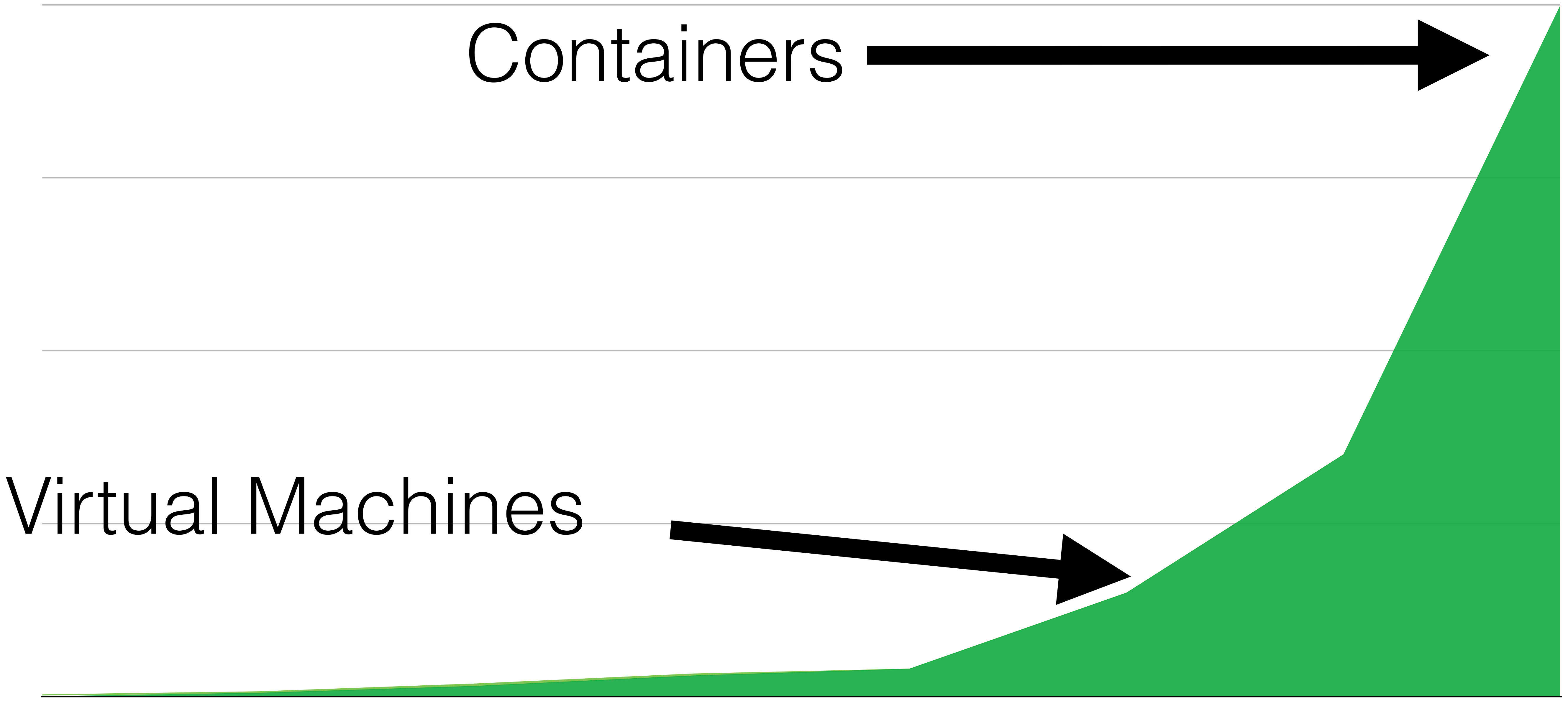
Snowflakes

Snowflakes

BASH remote exploit!

Ruby XSS exploit!

Snowflakes

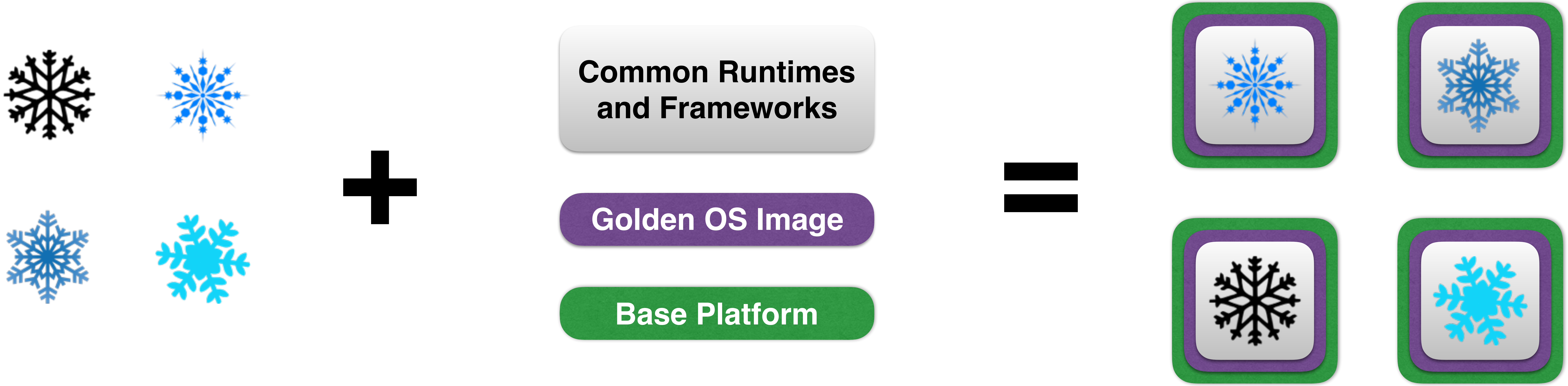


Containers

Virtual Machines

Solution: Golden Image

Solution: Golden Image



Doomsday Deployments

Solution: Predictable Pipelines

Solution: Predictable Pipelines

http://concourse.ci

concourse
publish
bosh-lite
rubbish-bin
concourse
atc
fly
tsa
blackbox
jettison
baggageclaim
go-concourse
concourse
concourse
concourse
concourse
concourse
concourse
concourse
version
garden-linux
version
minor
major
version
version
rc
concourse
version
garden-linux
bin
houdini
garden-linux-src
bin-rc
bosh-rc
concourse
version
bin
bin-rc
version
garden-linux
concourse
deployments
lite-stemcell
candidate-release
bin-testflight
testflight
concourse
version
bin-rc
concourse
version
garden-linux
candidate-release
deployments
aws-stemcell
bosh-lites
lite-deployment
testflight-logs
deploy
concourse
version
garden-linux
bin-rc
candidate-release
deployments
tracker
aws-deployment
shipit
version
final-release
concourse-master
patch
version

Solution: Predictable Pipelines

Need to trust your **tests**, your **platform**, and your **automation**.

Operational Explosion!

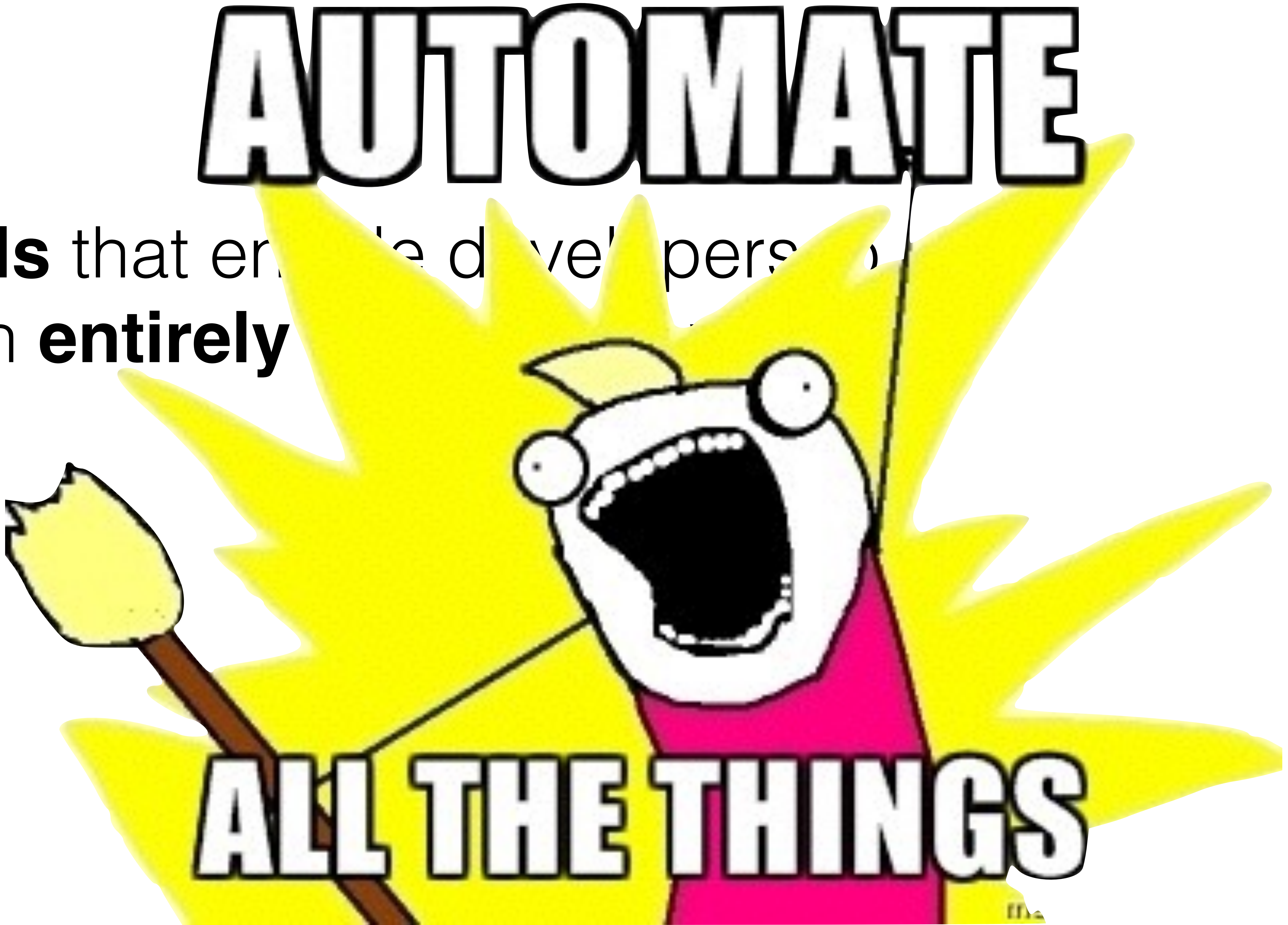
redis
mongoDB
elasticsearch.
python™
MEMCACHED
node JS™
cassandra
Scala
MariaDB
ERLANG
RAILS
Clojure
PIVOTAL HD
Neo4j
riak cs

Operational Explosion!

Operations block **Development**

Solution: AUTOMATE ALL THE THINGS!!!!!

Form a **team** to **build tools** that enable developers to manage the system in an **entirely**



OMG ALLTHETHINGS????

App deployment, infrastructure provisioning, OS installation, configuration management, database provisioning, disaster recovery, application monitoring, HA, blue-green deployments, self-healing, scaling, runtime installation, log rotation, backups, security updates, database upgrades, application logs, system logs, database logs, continuous integration, continuous deployment, service discovery, monitor queue usage, security monitoring, hotspot detection, error monitoring, issue notification and escalation, virtual machine migration, shard rebalancing, circuit breaker monitoring, resiliency testing, database snapshots, flux capacitors, ion overdrive maintenance, change the oil, dog feeding, cat shooting, pig eating…

Solution: AUTOMATE ALL THE THINGS!!!!!

Time and Money

CLOUD **FOUNDRY**

In summary…

Start boring and extract to services.

Understand the **hidden schemas**.

Amortize traffic with **queues**.

Decouple through **discovery tools**.

Contain failures with **circuit breakers**.

Enable other teams through **mockable clients.**

Kill your **snowflakes**.

Automate your **deployments**.

Build in **operations tools** from the beginning.

Make use of a platform like **Cloud Foundry.**