

Red Hat Summit Connect 2022 Warsaw

Easy-to-test, modern, Kubernetes-native apps with **Knative**

Chris Suszyński



@ksuszynski



/in/krzysztof-suszynski



About me

Chris Suszynski



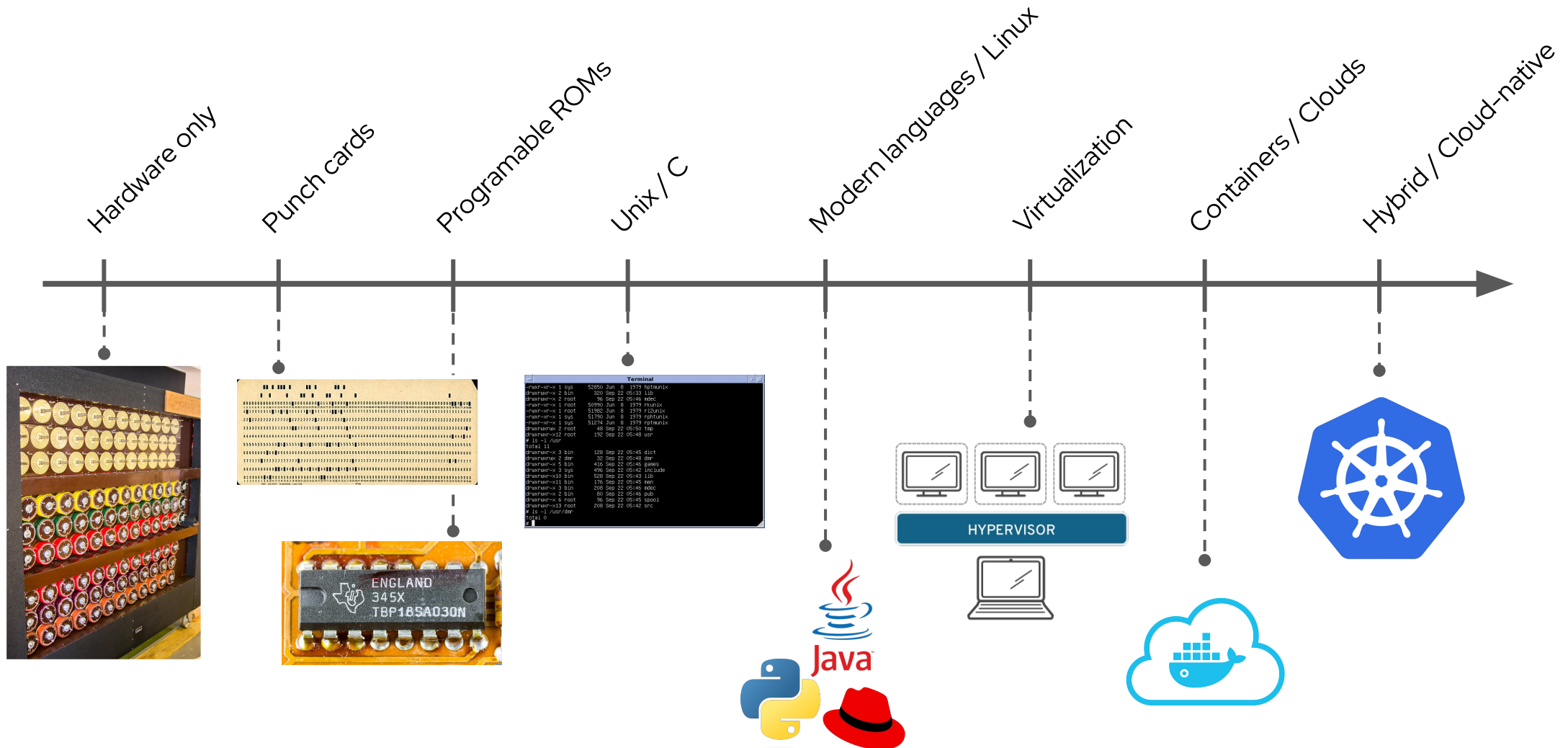
- Senior Software Engineer at Red Hat
- Work on OpenShift Serverless
- Golang lover
- Interested in Kubernetes & WASI
- On a Java & Puppet rehab
- 16y+ of dev experience
- breathe Open Source
- happy father & husband



Agenda

1. A bit of history
2. Apps of modern era
3. Serverless
4. Event Mesh
5. Kubernetes-native apps using Knative
6. Q&A

A bit of history

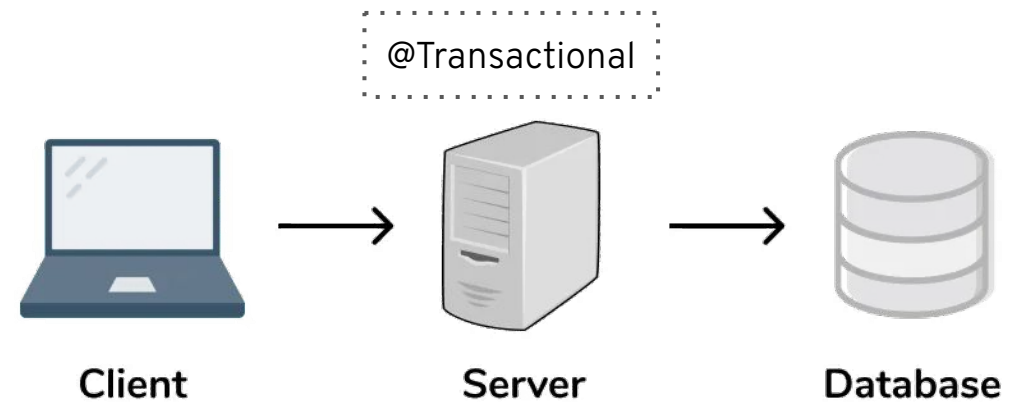


Unfortunately

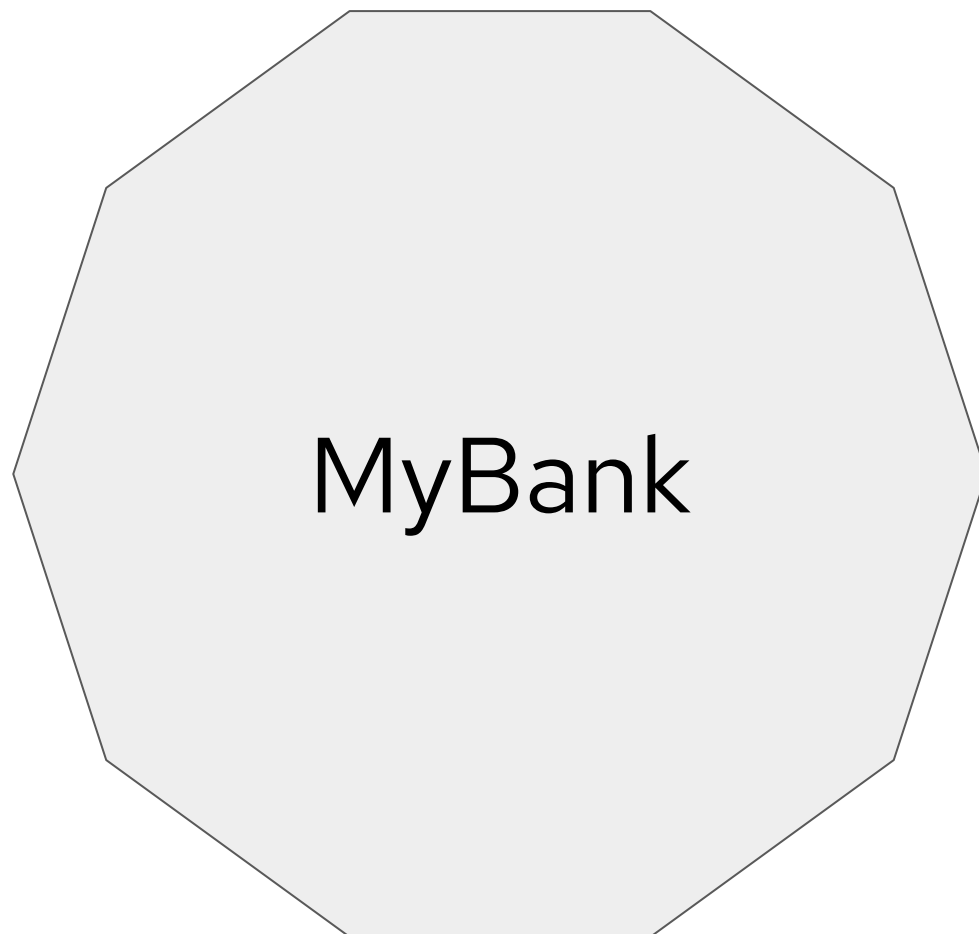
Our apps design is
still in the past.

Examples of legacy design

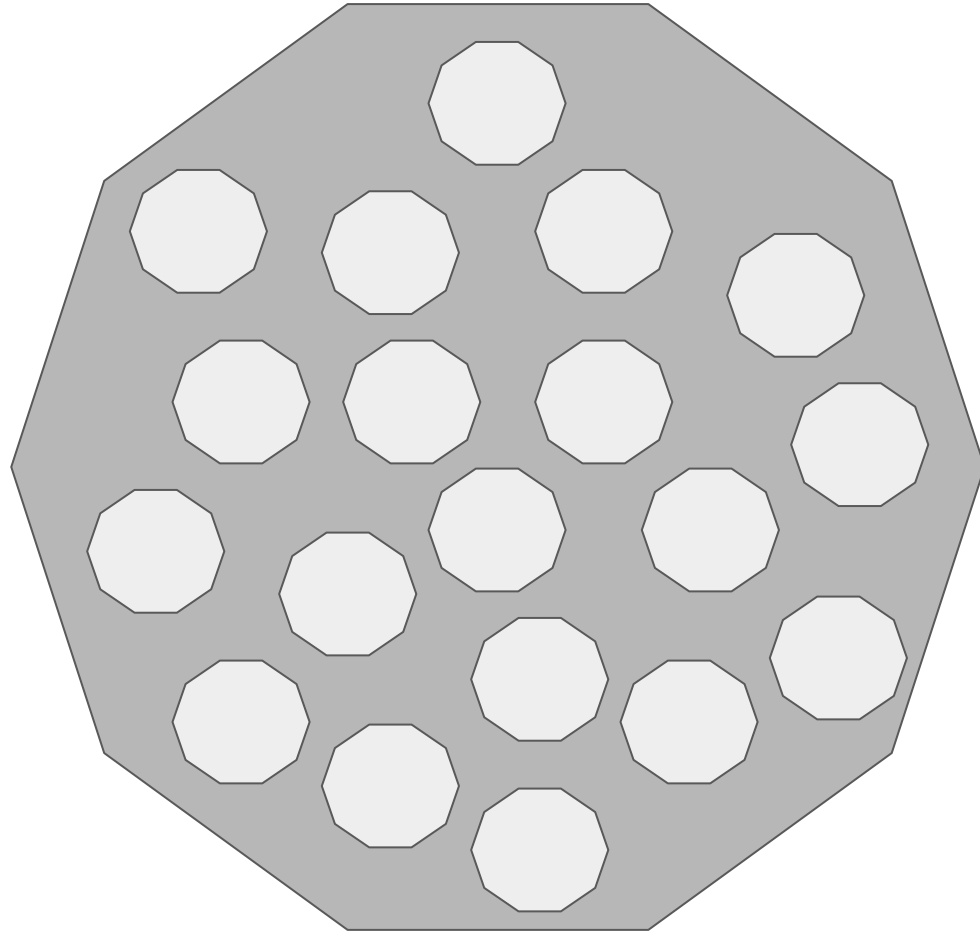
- 3-tier monolithic architecture
- Transactional everything



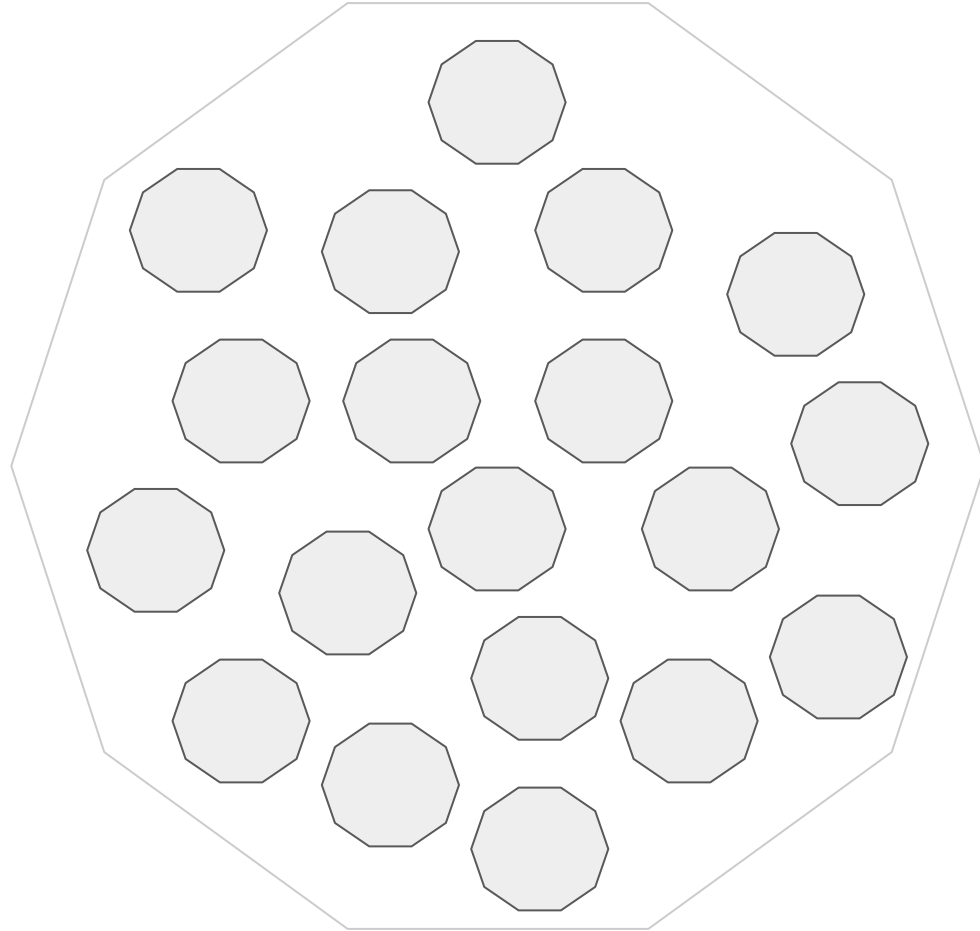
Monolith



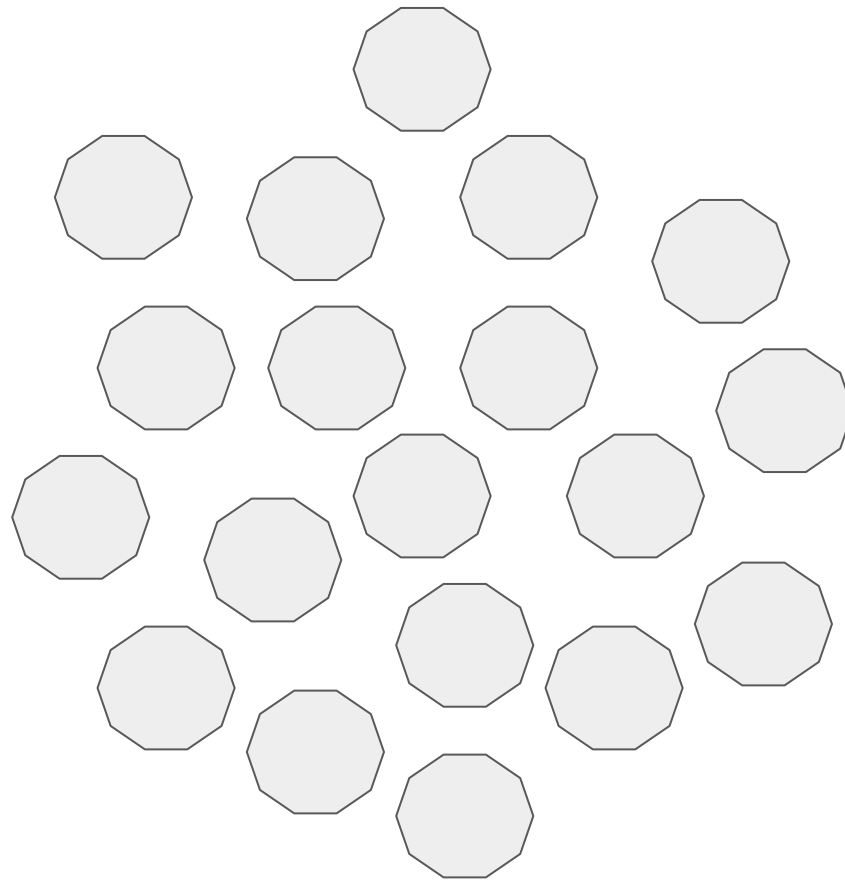
Inside monolith



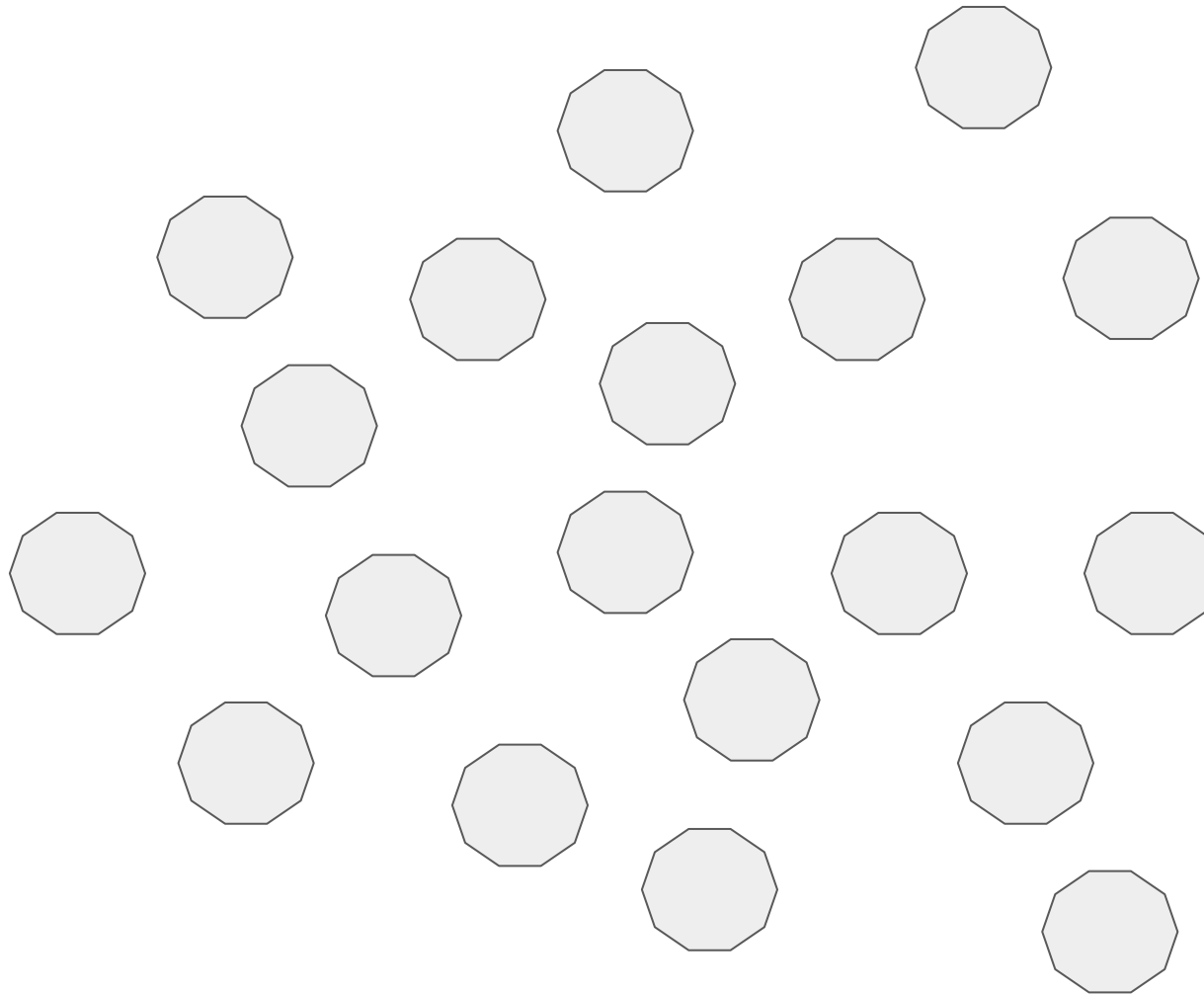
Dividing monolith



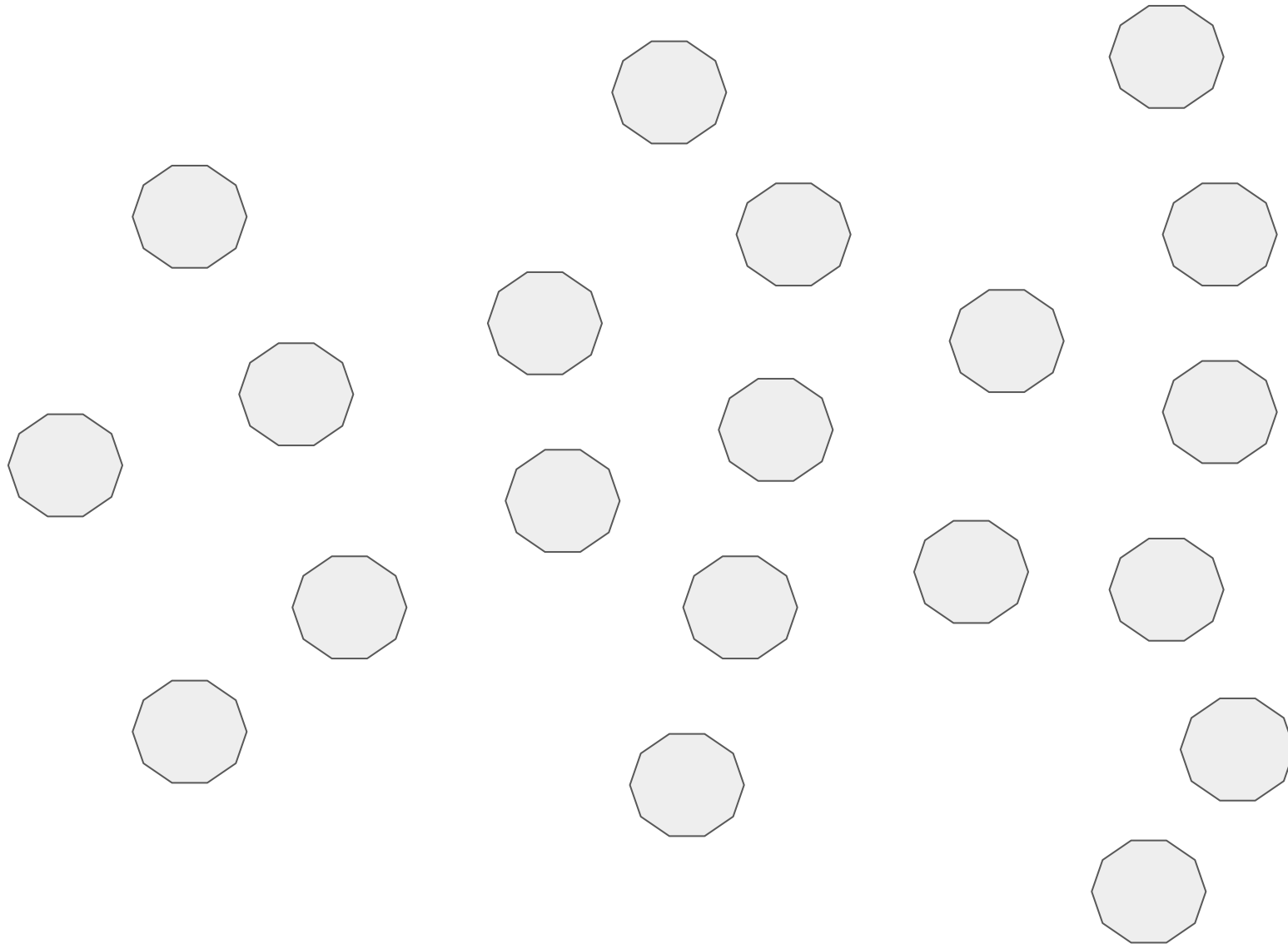
Enter microservices



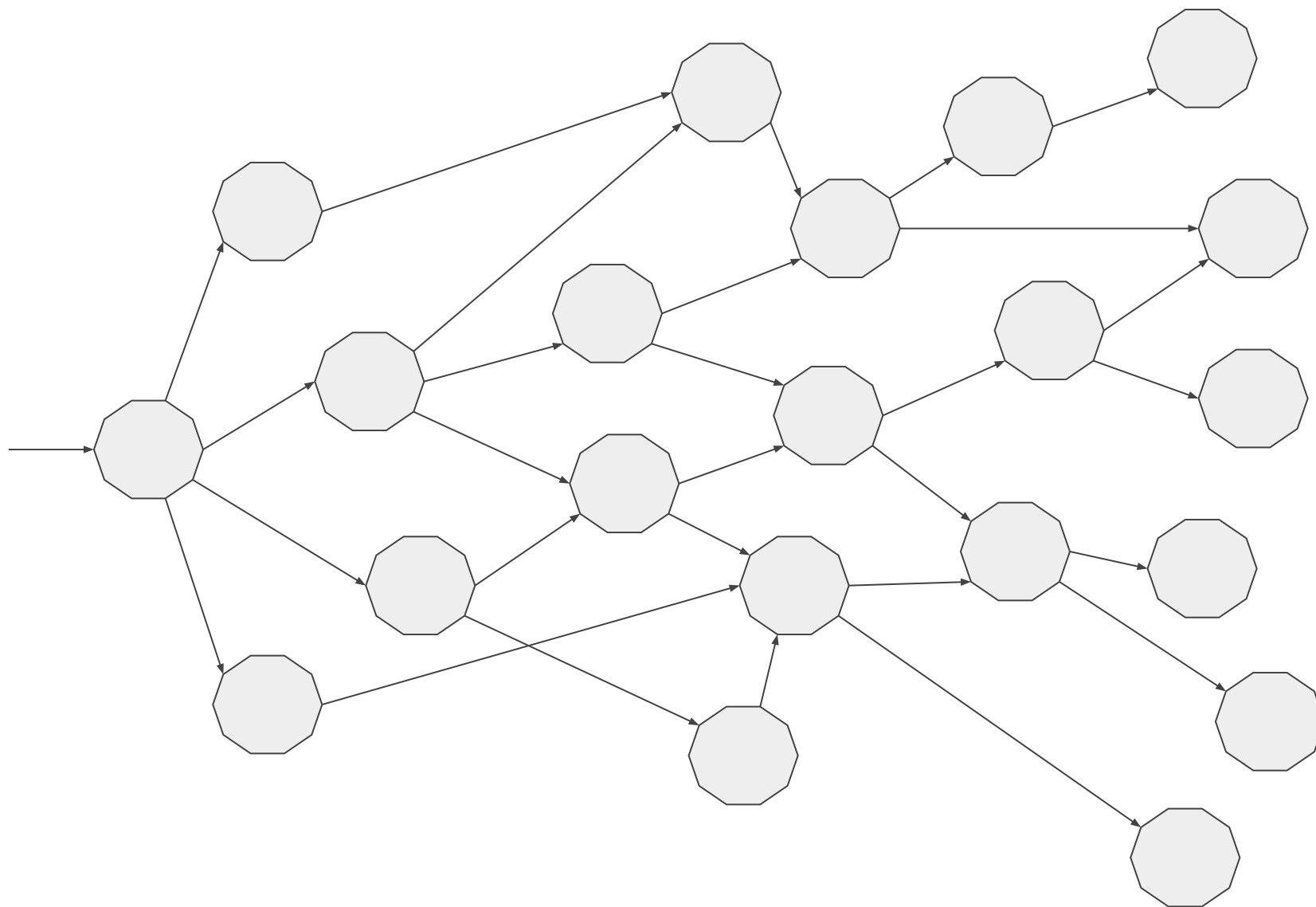
Enter microservices



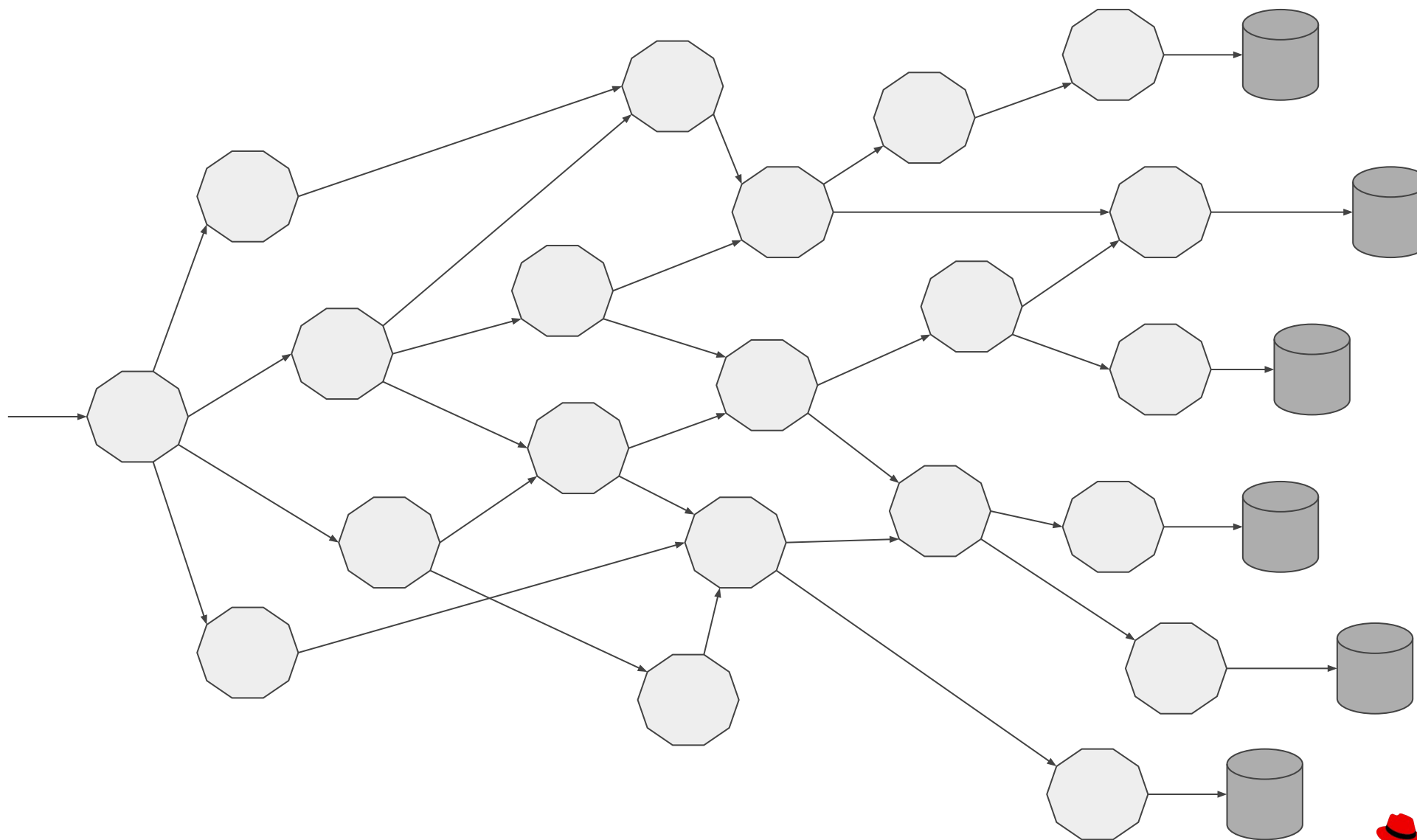
Enter microservices



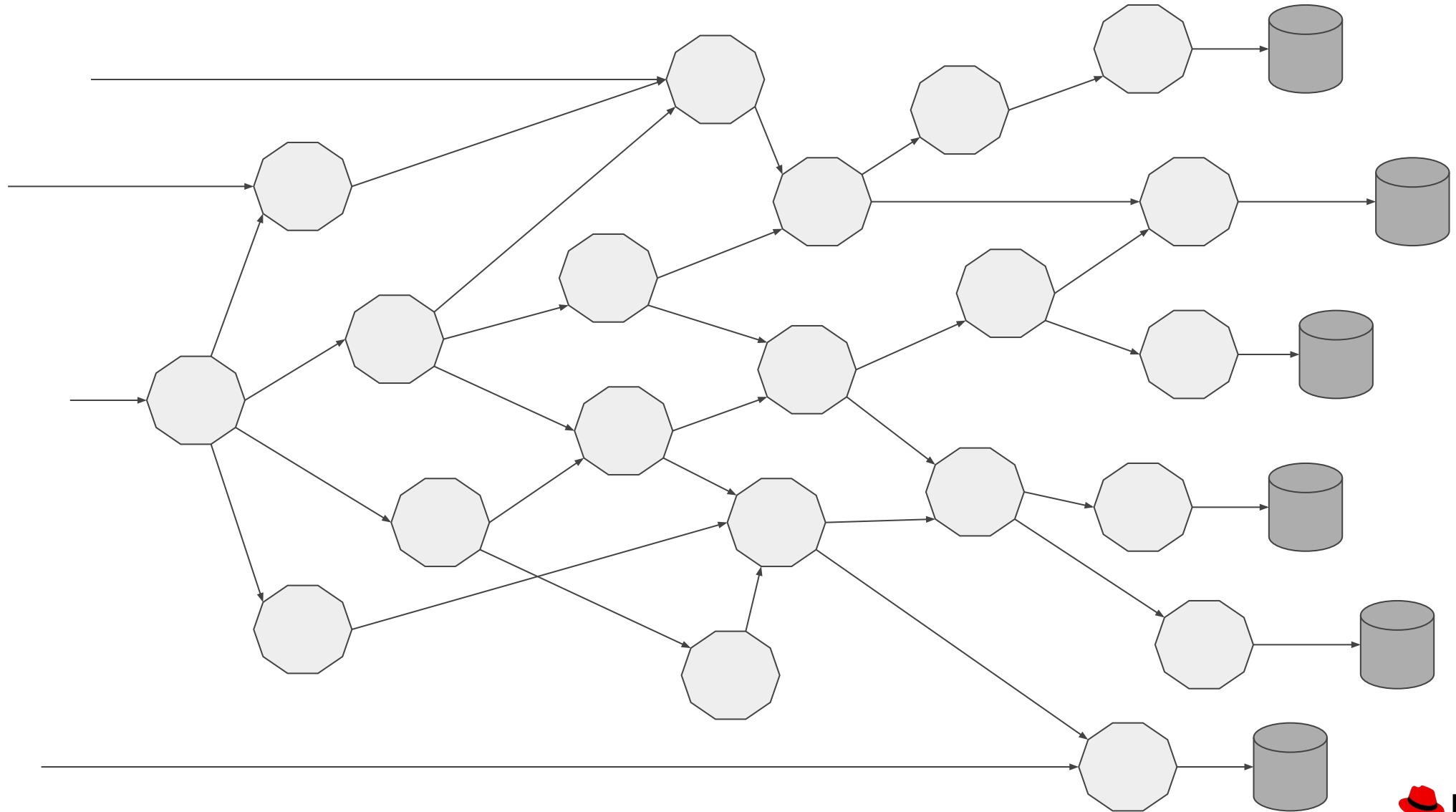
Network of microservices



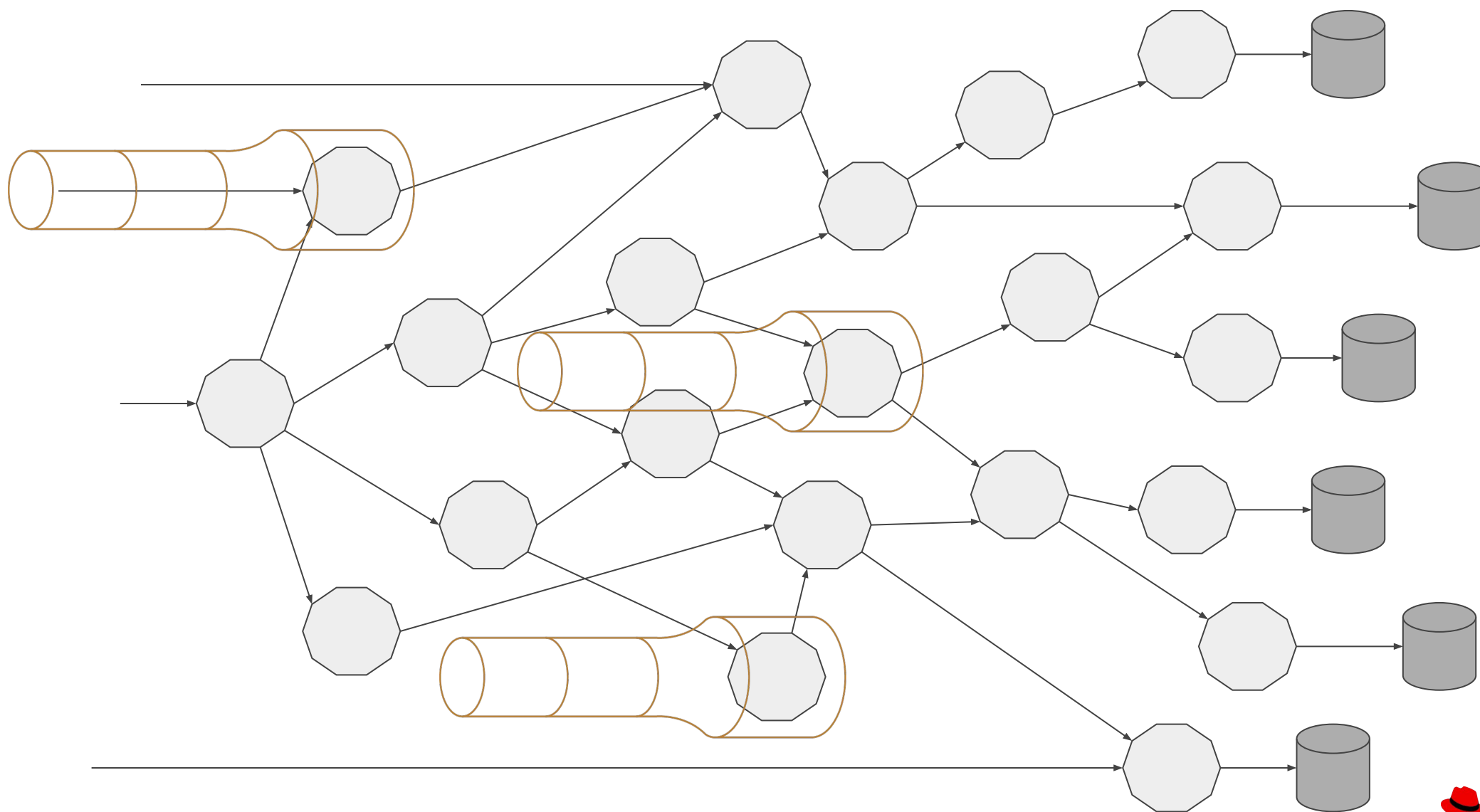
Microservices with their data



Multiple points of entry



Teams & pipelines

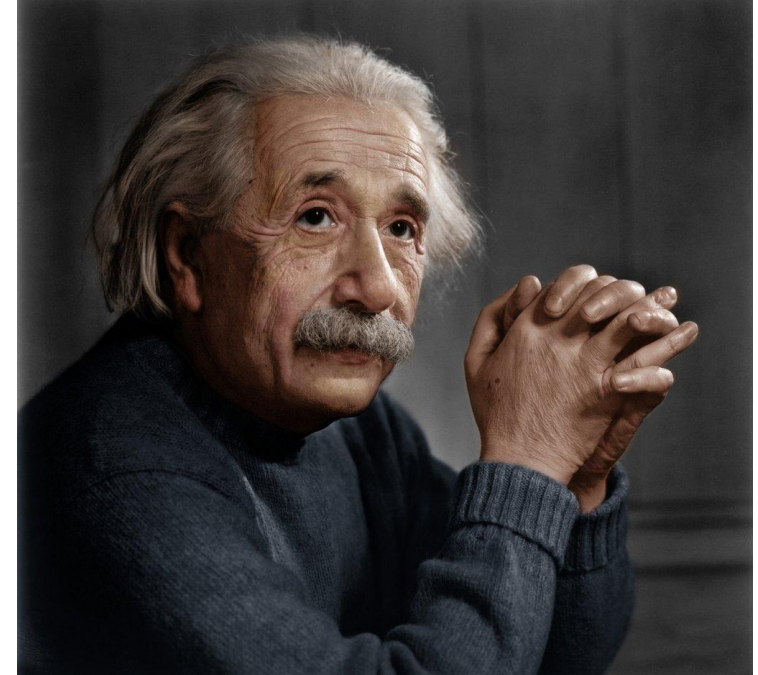


Mindblown



We need new ideas

“We cannot solve our problems with the same thinking we used when we created them”



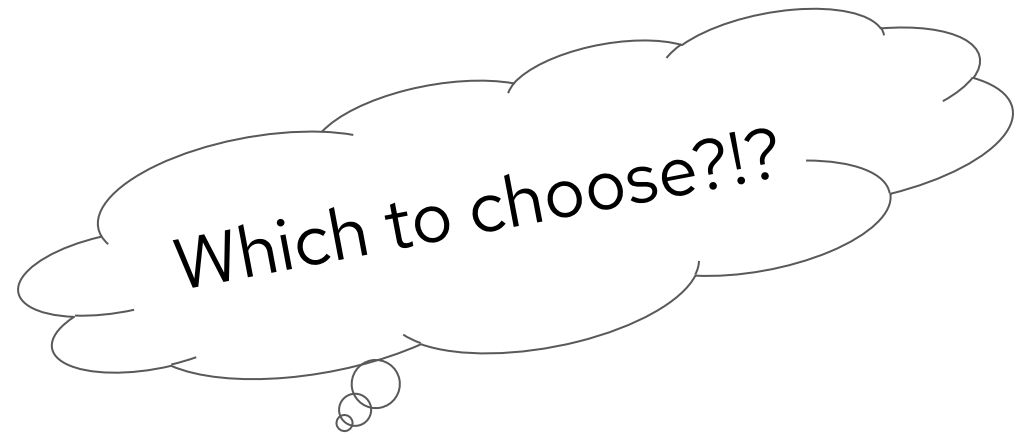
— Albert Einstein
(Theoretical Physicist)



New ideas!

New ideas

- 12-factor app
- Microservices
- Command & Query Separation
- Event Sourcing
- Ports and Adapters / Hexagonal architecture
- Eventual consistency
- Ease of testing



Road to Awesomeness



Re-Org to
DevOps



Self-Service,
On-Demand,
Elastic
Infrastructure



Automation



CI & CD
Deployment
Pipeline



Advanced
Deployment
Techniques



Microservices

Serverless Computing

What is

Serverless computing

"Serverless computing refers to the concept of building and running applications that do not require server management.

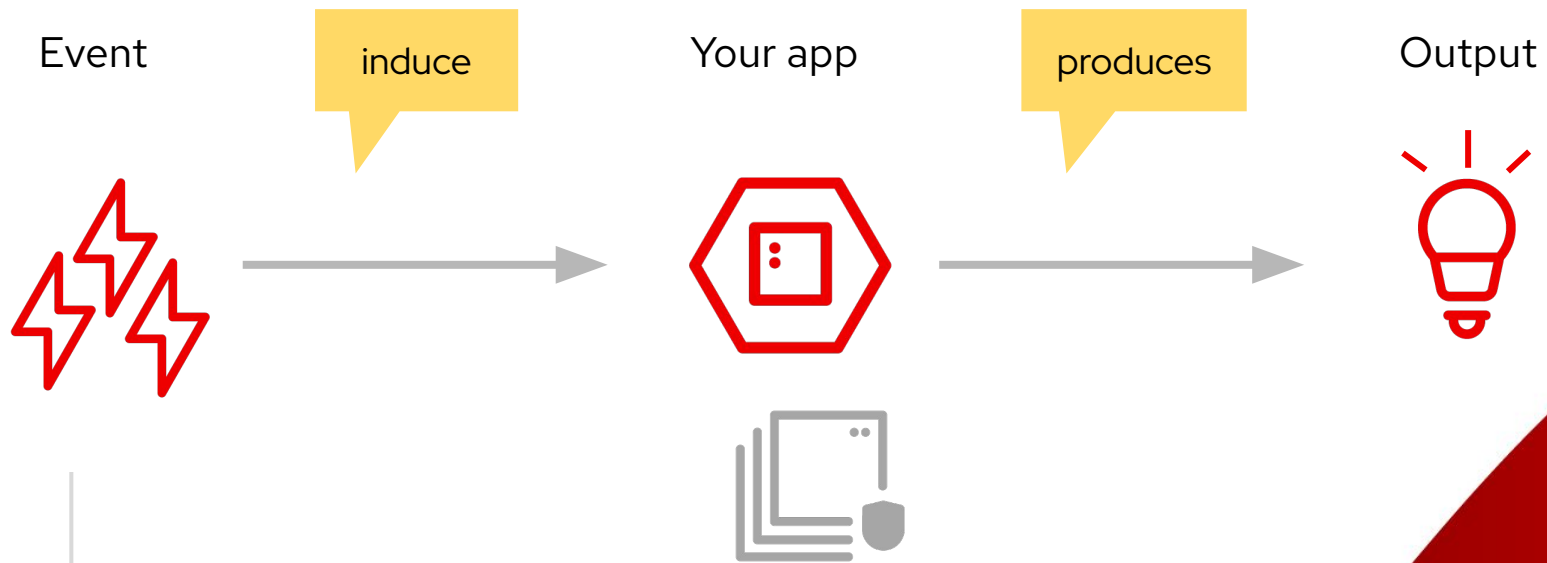
It describes a finer-grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment."

—— Cloud Native Computing Foundation

<https://www.cncf.io/blog/2018/02/14/cncf-takes-first-step-towards-serverless-computing>



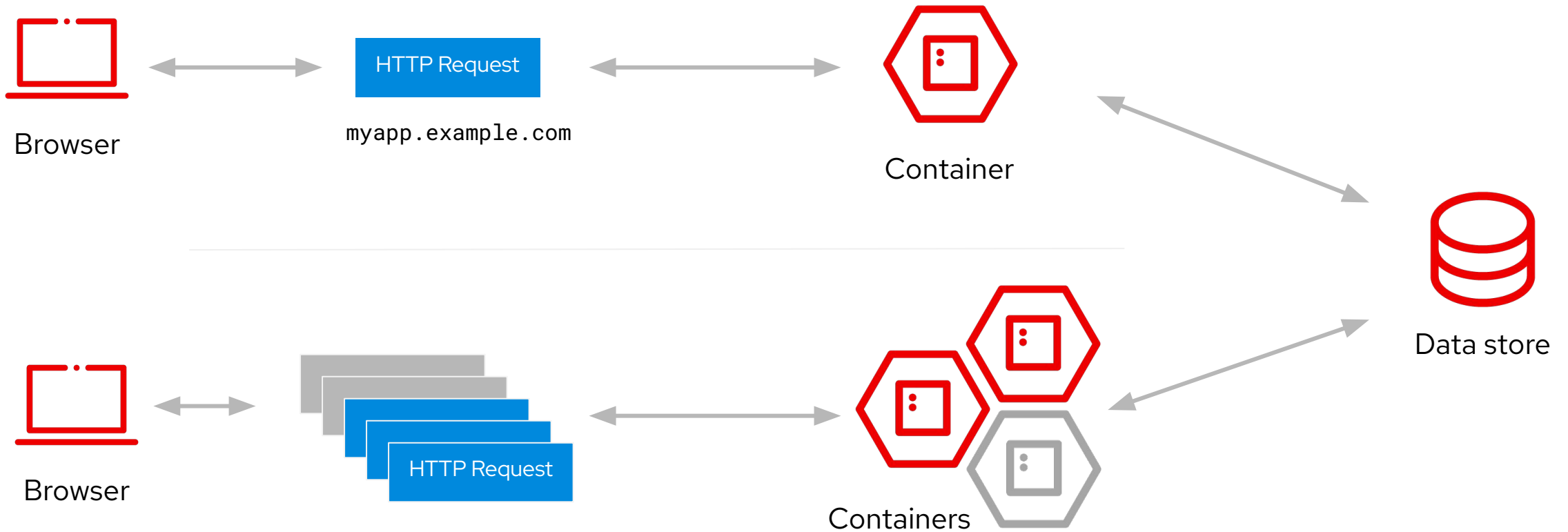
"Serverless" pattern



HTTP calls
Kafka messages
Photo upload
New order
User sign-in

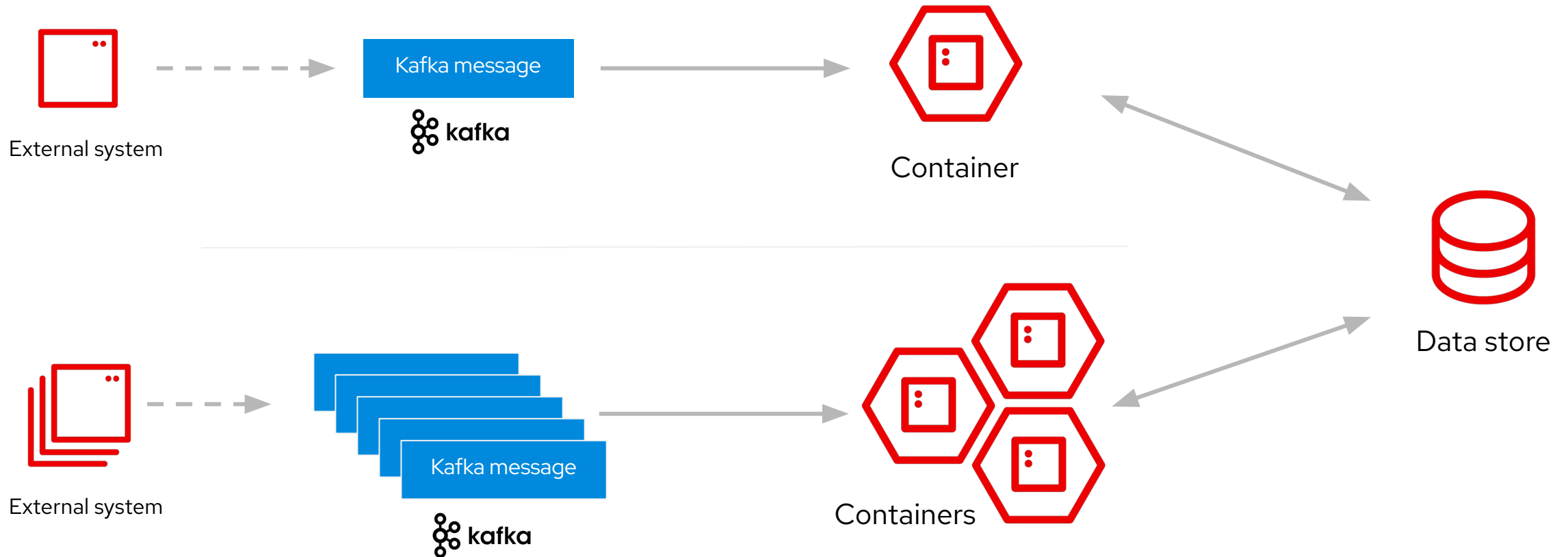
"Serverless" pattern

Serverless web app



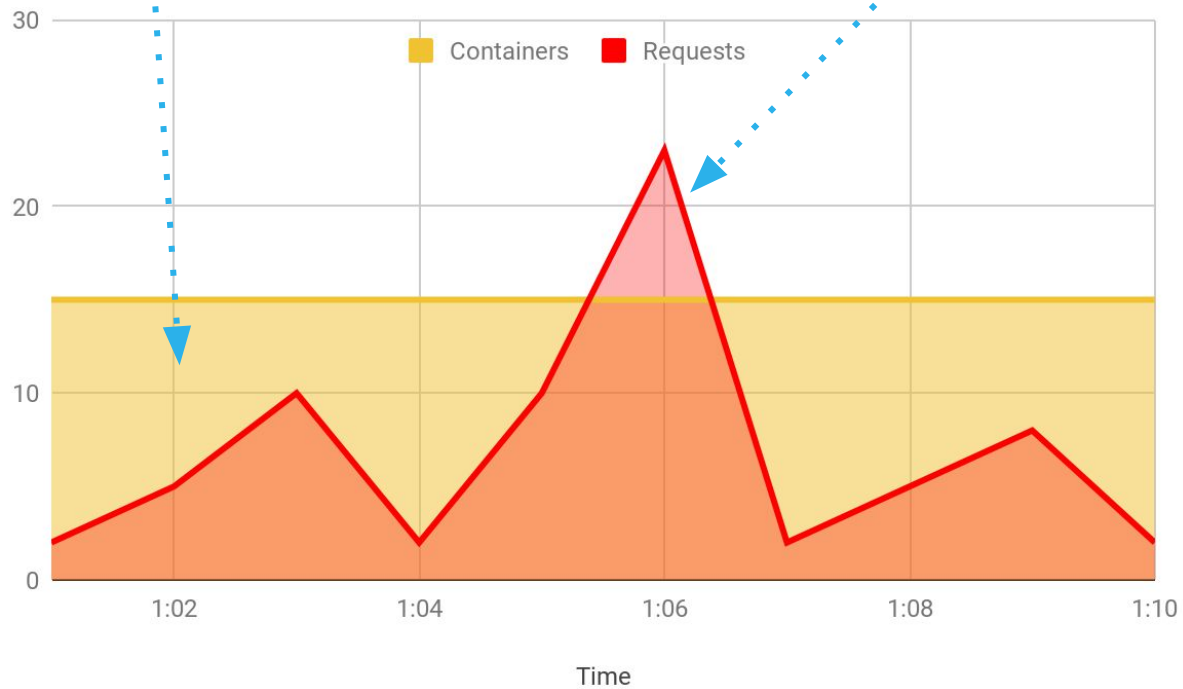
"Serverless" pattern

Kafka message processing



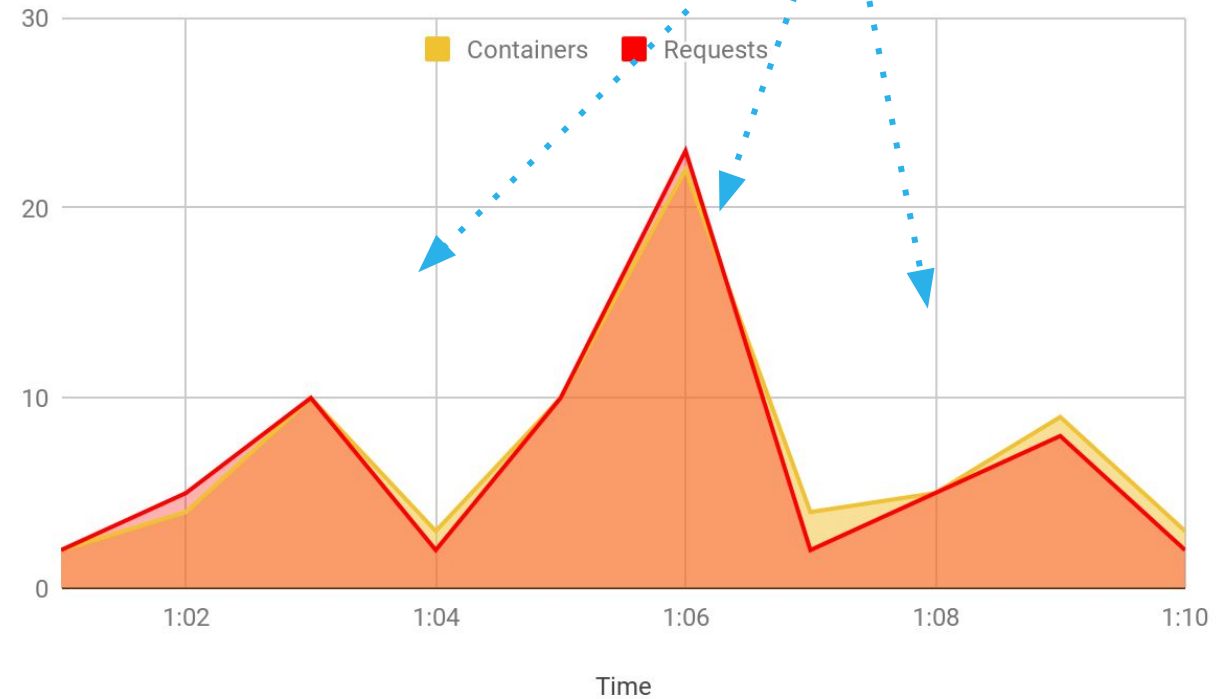
Serverless operational gains

Too many resources
Costs of unused resources



Not enough resources
Loss of revenue
Poor quality of services

Usage adaptation
Direct correlation between
costs and business
revenue



without Serverless

with Serverless

Event Mesh

What is

Event Mesh

*"An event mesh is a **configurable** and **dynamic** infrastructure layer for distributing **events** among **decoupled applications**, cloud services and devices.*

*It enables event communications to be **governed**, flexible, **reliable** and **fast**. An event mesh is created and enabled through a network of interconnected **event brokers**."*

— Solace

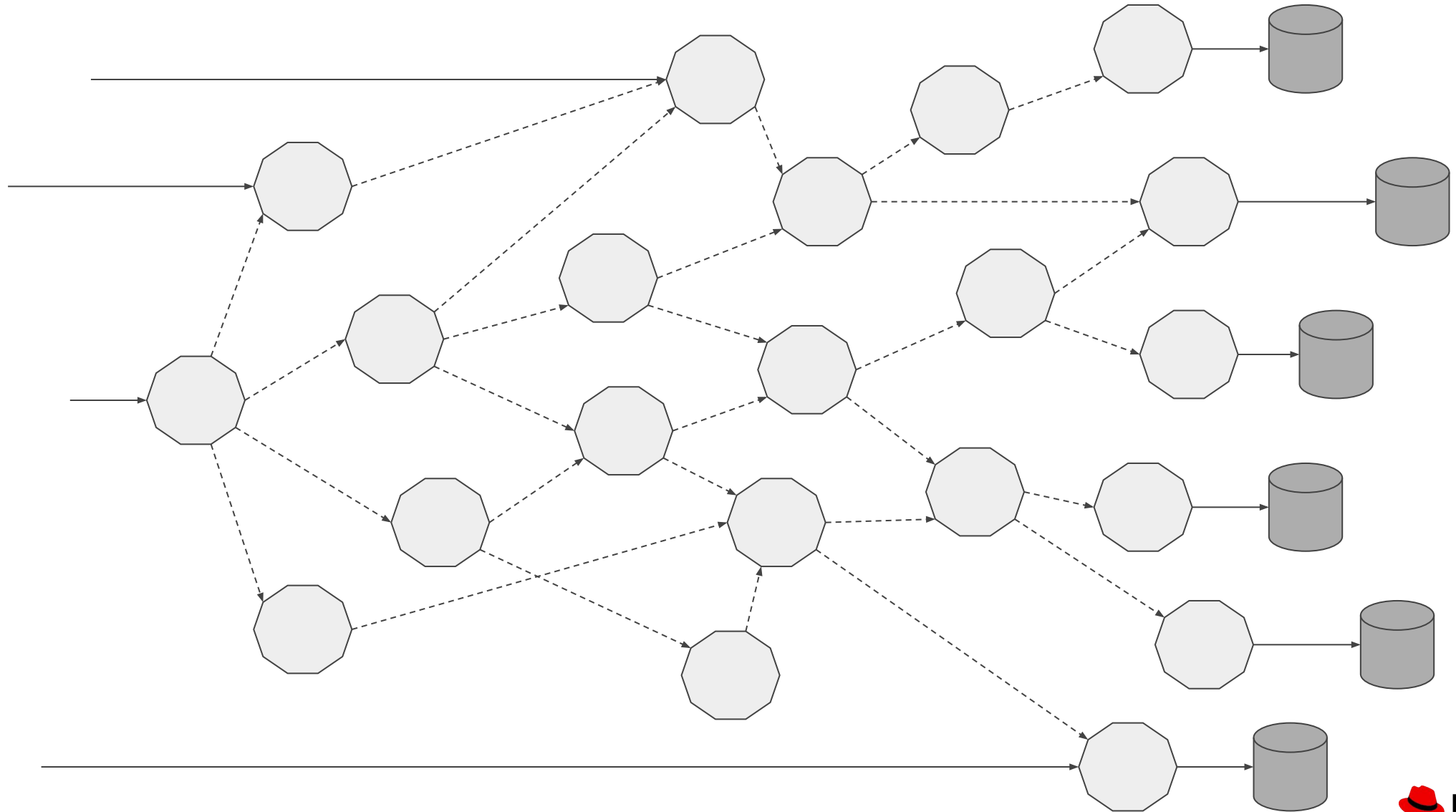
<https://solace.com/what-is-an-event-mesh>



Service Mesh vs Event Mesh

	Service Mesh		Event Mesh	
Similarities	<ul style="list-style-type: none">● Flexibility● Robustness● Decoupling			
Differences	<ul style="list-style-type: none">● Synchronous● Request and response● Better for queries		<ul style="list-style-type: none">● Asynchronous● Event● Better for commands	

Eventual consistency = Event Mesh + CQRS




OpenShift Serverless

based on Knative



Knative 

is a Kubernetes  extension
that allows you to **deploy** and
manage modern **serverless**
apps.

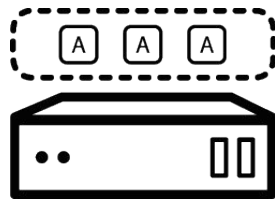
Knative in OpenShift

- Knative is a **CNCF Open Source** project
- A community driven by multiple stakeholders <https://knative.dev>
 - Supported by **Google**, **Red Hat**, **IBM**, **VMware**, **TriggerMesh**, **SAP** and more
- OpenShift **Serverless**: <https://www.openshift.com/learn/topics/serverless>
- Latest production-ready release: **1.25.0** (Knative 1.4)

Knative components

Serving

A request-based model that serves an app container and can "scale to zero."



Eventing

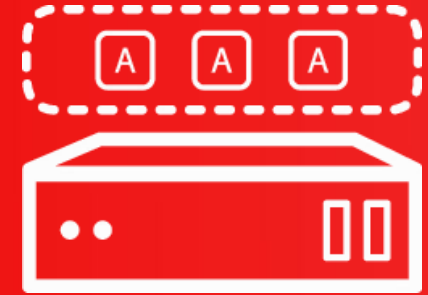
Framework for propagation of events that will stimulate apps.



Demo!

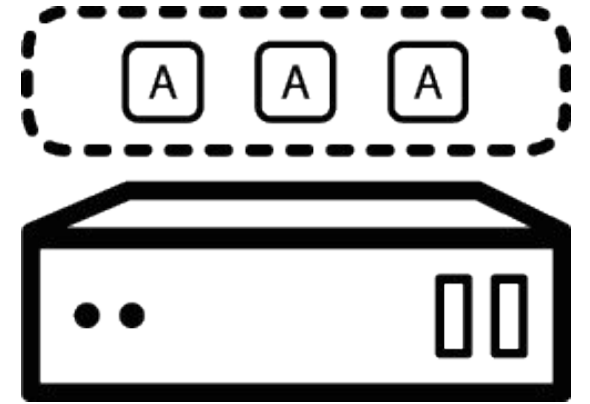
Hello World

Serverless Serving



Easy **routing**, scaling **to zero** and
to **the demand** plus automatic
revision tracking

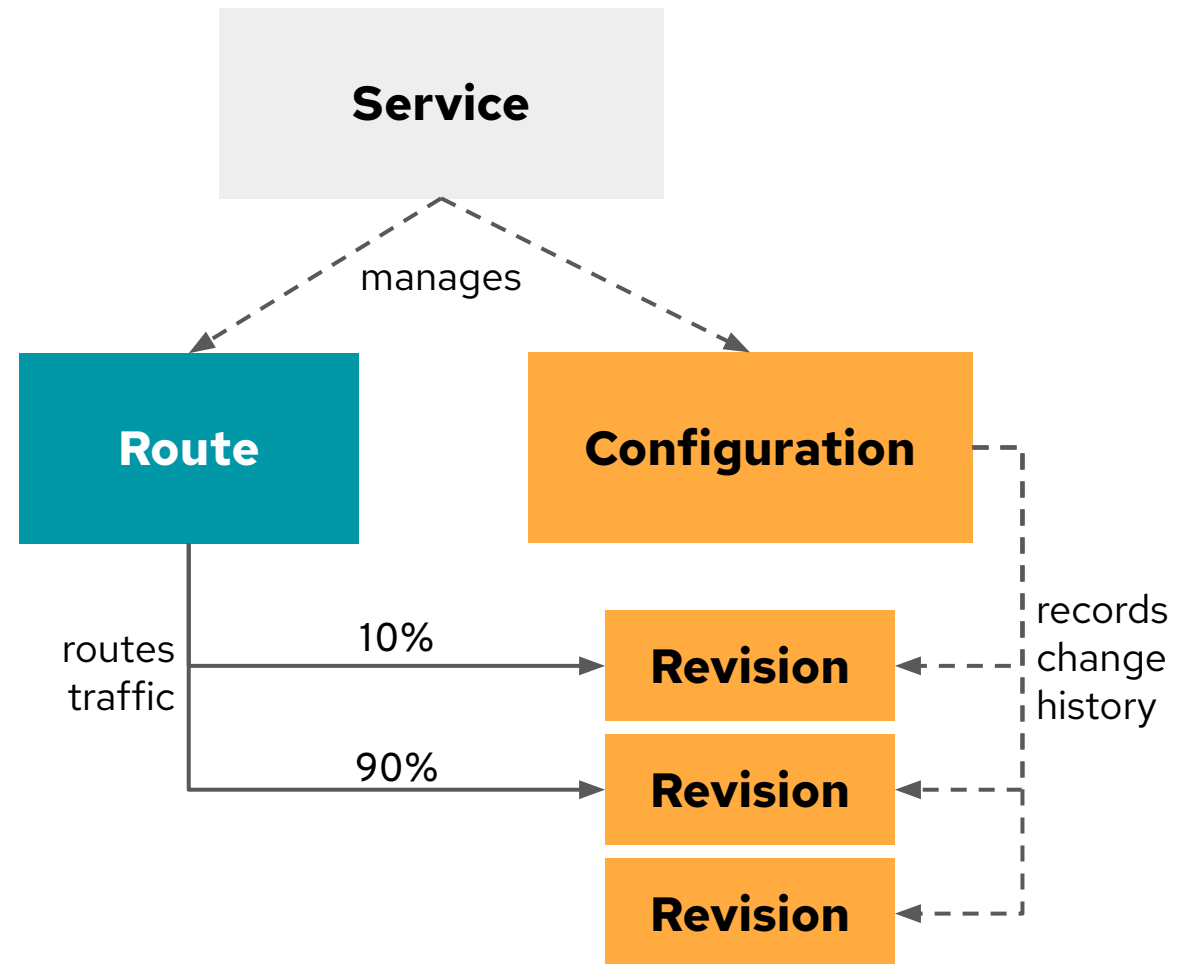
Serving concepts



- **Automatic request-based scaling**, including scaling to zero
- Separation of code from configuration
- An opinionated **deployment model** tailored to stateless applications
- **Traffic routing** capabilities for secure deployment of new versions

Serving resources

- **Configuration** represents the "floating HEAD" of the **Revision** history
- **Revision** represents an immutable snapshot of code and configuration
- **Route** configures ingress using a set of revisions
- **Service** (it's not K8s service!) is the public entity that we will operate, a facade for the user



Migrating to Knative

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: random
spec:
  replicas: 1
  selector:
    matchLabels:
      app: random
  template:
    metadata:
      labels:
        app: random
    spec:
      containers:
      - image: rhsummit2020/random:1.0
        name: random
        ports:
        - containerPort: 8080
```

... and **K8s Service,
Route, Autoscaler**

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: random
spec:
```

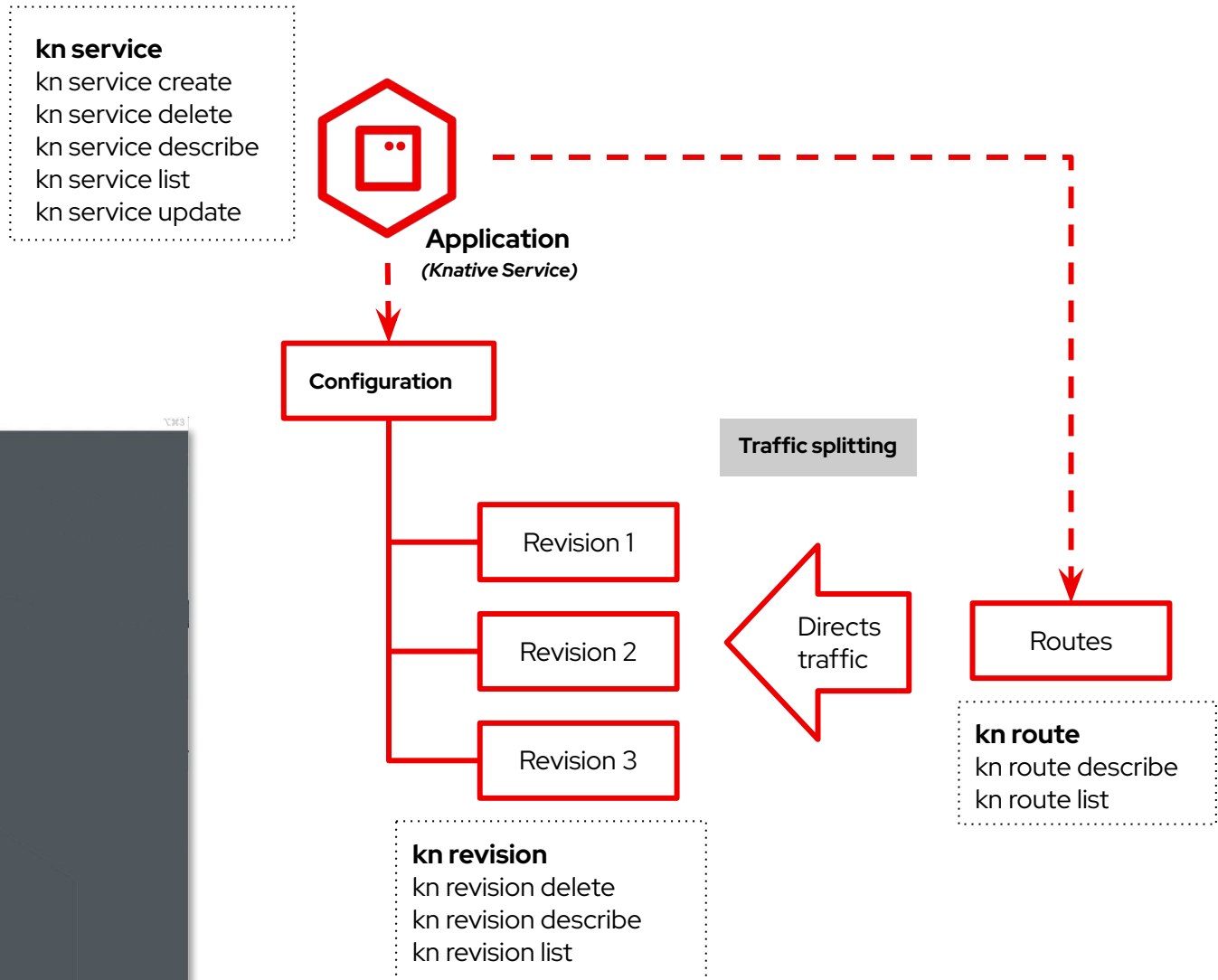
**Routing and autoscaling
out-of-the-box**

```
  replicas: 1
  selector:
    matchLabels:
      app: random
  template:
    metadata:
      labels:
        app: random
    spec:
      containers:
      - image: rhsummit2020/random:1.0
        name: random
        ports:
        - containerPort: 8080
```


Developer UX

- From image address to container in seconds
- Simpler development experience for K8s
- Built-in versioning, traffic splitting and more
- Simplified lightweight installation with Kourier
- Automatic TLS/SSL

```
markito@anakin: ~/projects/redhat
$ kn service create greeter --image quay.io/rhdevelopers/knative-tutorial-greeter:quarkus
```



Demo!

Autoscaling

Serverless Eventing



An universal **Event Mesh** based
sources, **brokers**, channels and sinks
for **CNCF Cloud Events**

Serverless Eventing



cloudevents

- Based on CNCF CloudEvents (regular HTTP)
- Exchangeable transport: **Channels** and **Brokers**
 - In-Memory (dev only)
 - Apache Kafka
 - Google Pub-Sub, ...
- Flexible event routing from **Sources** to **Sinks**
 - **Source**: adapter that integrates external systems and emits CloudEvents
 - **Sink**: addressable (HTTP) endpoint receiving CloudEvents (can be Kn Service or K8s Service)

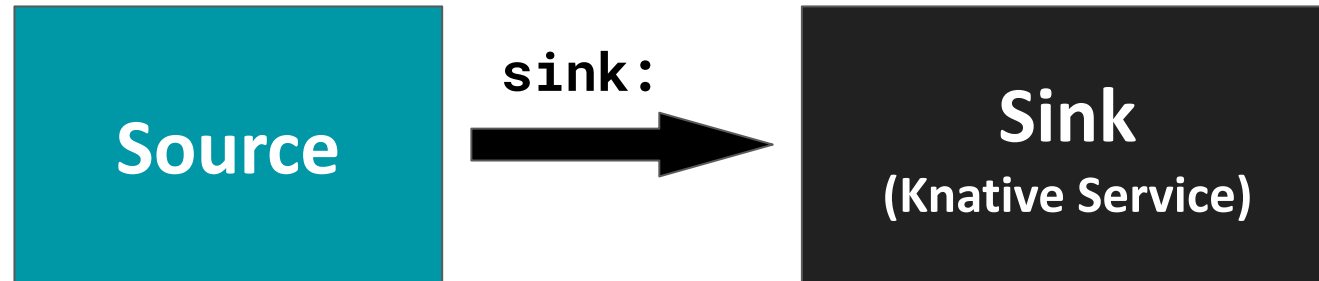
Sources

Built-in sources	
PingSource	Periodically emits a static CloudEvent
ApiServerSource	Reports K8s API Server events as CloudEvent
SinkBinding	Connects a pod to Event Mesh

Other sources	
GitHubSource	Converts GitHub webhooks to CloudEvents
KafkaSource	Kafka messages as CloudEvents
CamelKSource	KApache Camel components as CE sources

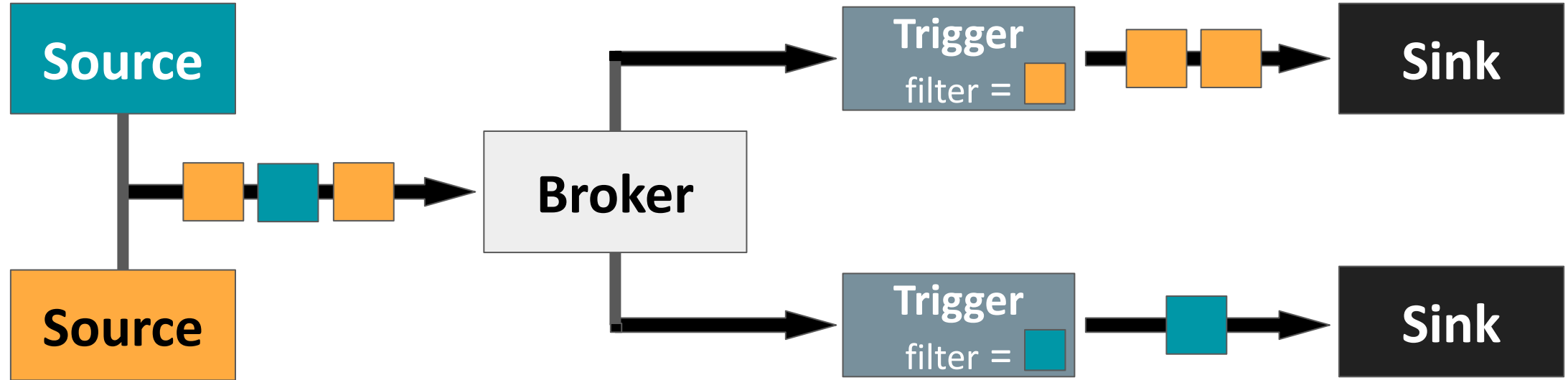
and many more: <https://knative.dev/docs/eventing/sources/>

Source → Sink : Directly



- The easiest way to send CloudEvent to a service
- Disadvantages:
 - No queuing support if the service is unavailable
 - No back-pressure mechanism
 - Only one service can consume events
 - No filtering, Sink will always get all events

Source → Sink : Broker and Trigger



Broker

- Build-in queue
- Back pressure
- Persistence (some implementations)

Trigger

- Filters events by their attributes from CloudEvents (i.e. type)
- Connects Broker to the Sink

Demo!

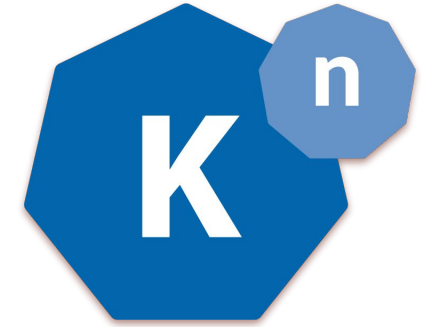
Event Mesh

Kubernetes-native apps

Applications using Knative along with Tekton **are a natural** for Kubernetes.

We could cover all our ideas, while avoiding vendor lock-in!

Kubernetes-native apps



- 12-factor app : Knative, Tekton, OpenTelemetry
- Microservices, Functions : Knative Serving
- Command & Query Separation : Serving for queries,
Eventing for commands
- Event Sourcing : Persistent Event Mesh
- Ports and Adapters / Hexagonal architecture :
code nicely, please
- Eventual consistency : reconcile loop like Kubernetes
operators
- Ease of testing : HTTP calls only

Knative Resources

- bit.ly/knative-tutorial
- developers.redhat.com/coderland/serverless
- github.com/cardil/knative-serving-showcase
- bit.ly/kubernetes-tutorial
- bit.ly/quarkus-tutorial
- developers.redhat.com



Thank you



<http://linkedin.com/company/Red-Hat>



<http://facebook.com/RedHatinc>



<http://youtube.com/user/RedHatVideos>



<https://twitter.com/RedHat>

Chris Suszyński



[@ksuszynski](https://twitter.com/ksuszynski)



[/in/krzysztof-suszynski](http://in/krzysztof-suszynski)