RED HAT®
DEVELOPER
PROGRAM

Serverless and Servicefull

Where Microservices compliments Serverless

 @rafabene

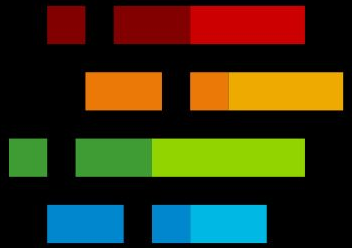
 benevides@redhat.com

Link → http://bit.ly/serverlessfull

Rafael Benevides
Director of Developer Experience at Red Hat
Apache DeltaSpike P.M.C

benevides@redhat.com
@rafabene

Java Certifications:
SCJA / SCJP / SCWCD / SCBCD / SCEA
JBoss Certifications:
JBCD / JBCAA
Red Hat Certifications:
OpenShift / Containers / Ansible
Other Certifications:
SAP Netweaver / ITIL / IBM Software Quality

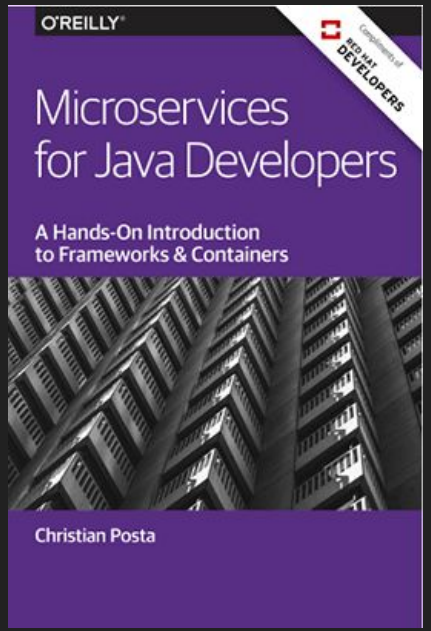


RED HAT® DEVELOPER

Get software and know-how.
Get started with Red Hat technologies.

Join at **developers.redhat.com**.

bit.ly/javamicroservicesbook

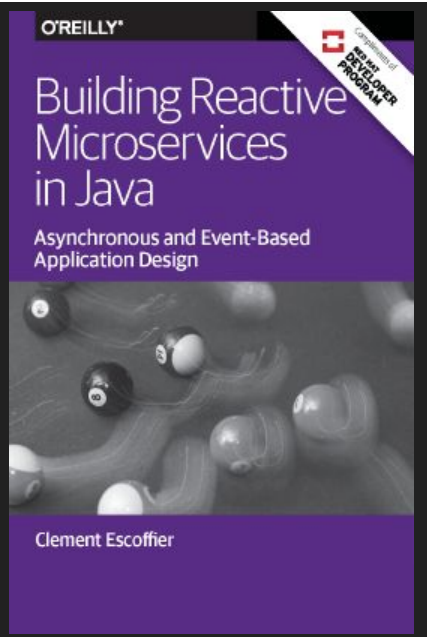


Free eBooks from developers.redhat.com

Microservices Introductory Materials

Demo: bit.ly/msa-instructions
Slides: bit.ly/microservicesdeepdive
Video Training: bit.ly/microservicesvideo
Kubernetes for Java Developers

bit.ly/reactivemicroservicesbook



Advanced Materials

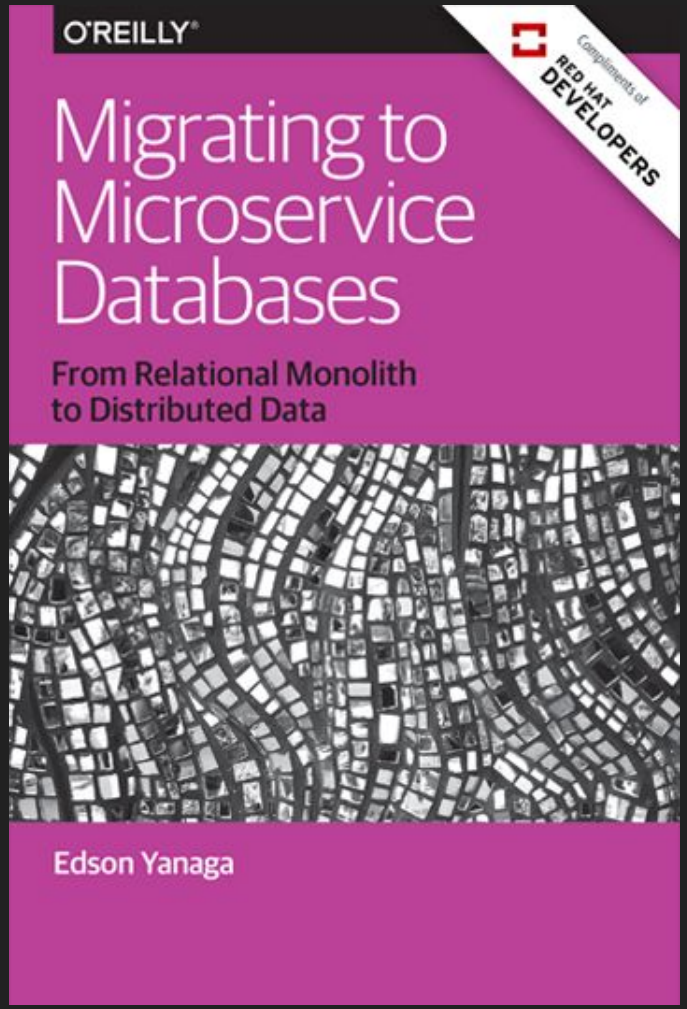
bit.ly/istio-tutorial
learn.openshift.com/servicemesh

bit.ly/faas-tutorial
learn.openshift.com/serverless

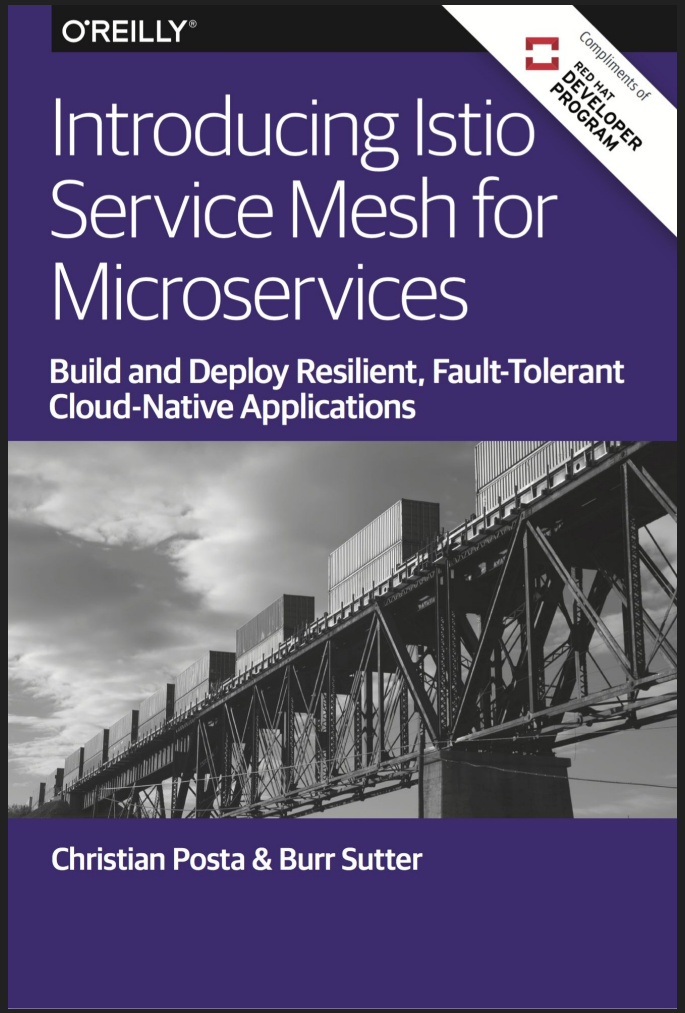
http://bit.ly/serverlessfull

@rafabene

RED HAT DEVELOPER PROGRAM



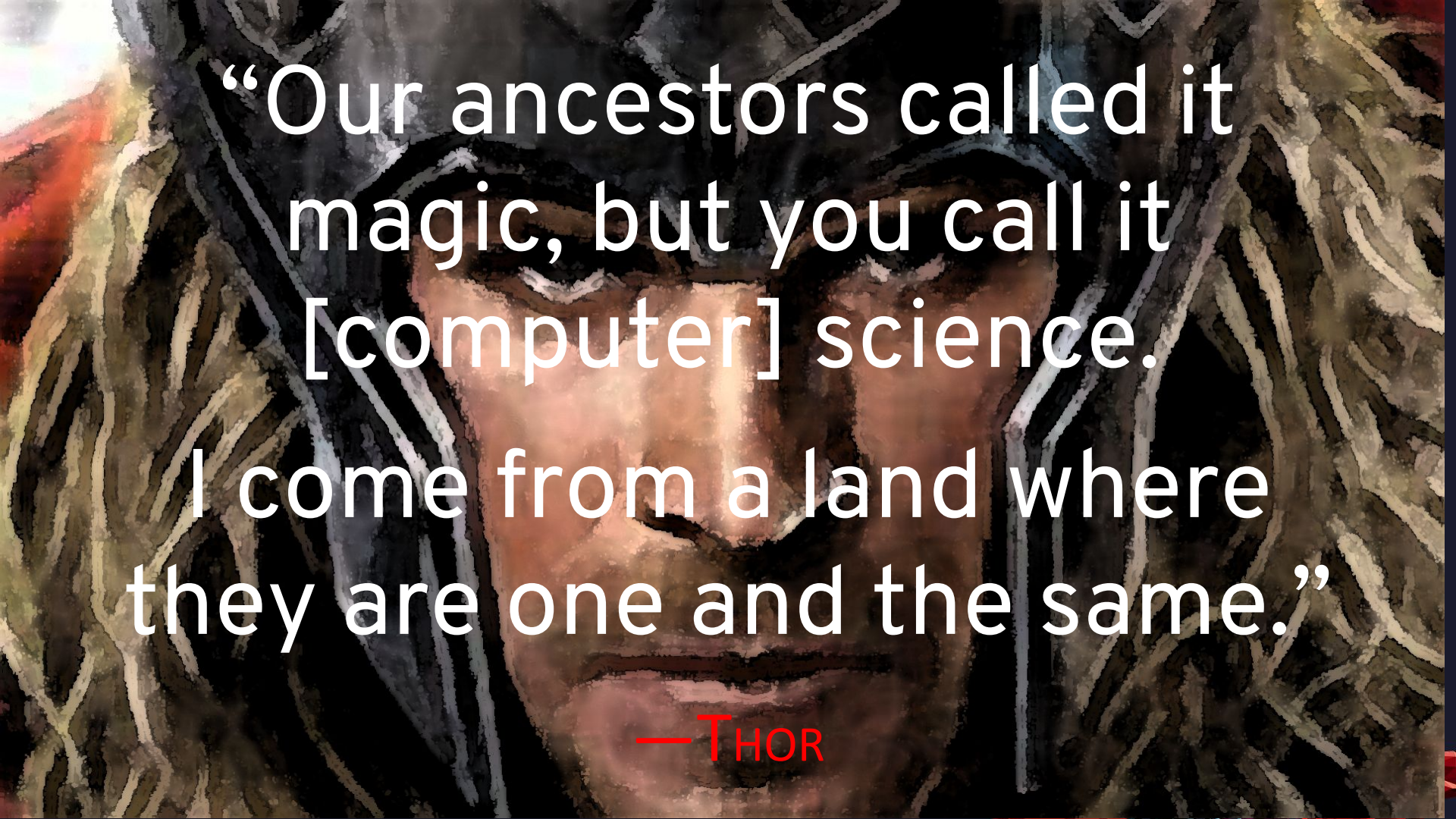
bit.ly/mono2microdb



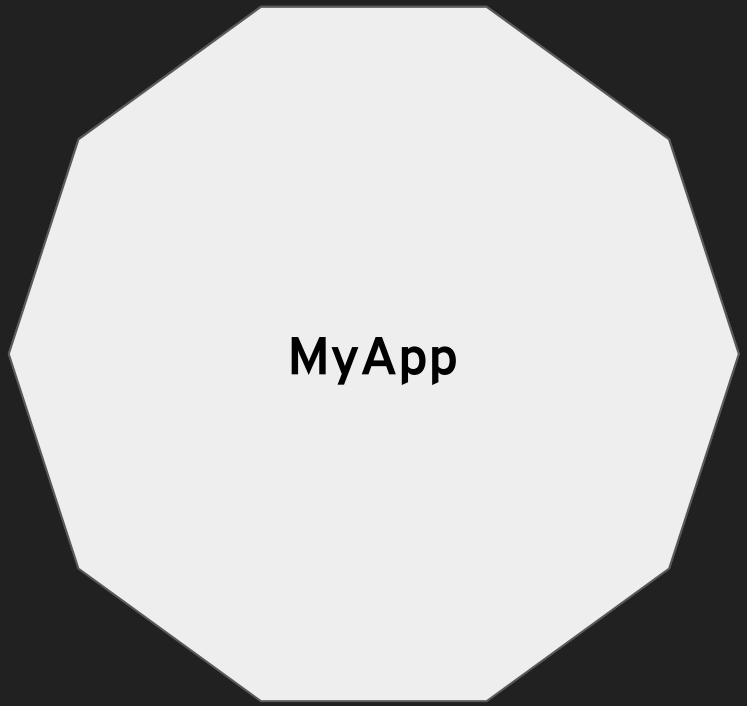
bit.ly/istio-book

Keynote Demo 2018

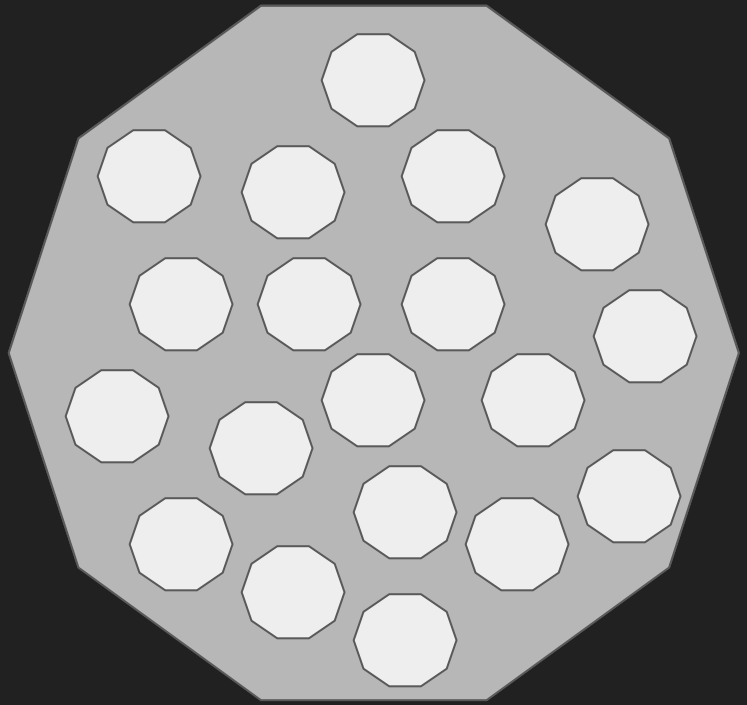
https://youtu.be/hu2BmE1Wk_Q?t=477


“Our ancestors called it magic, but you call it [computer] science.
I come from a land where they are one and the same.”
—THOR

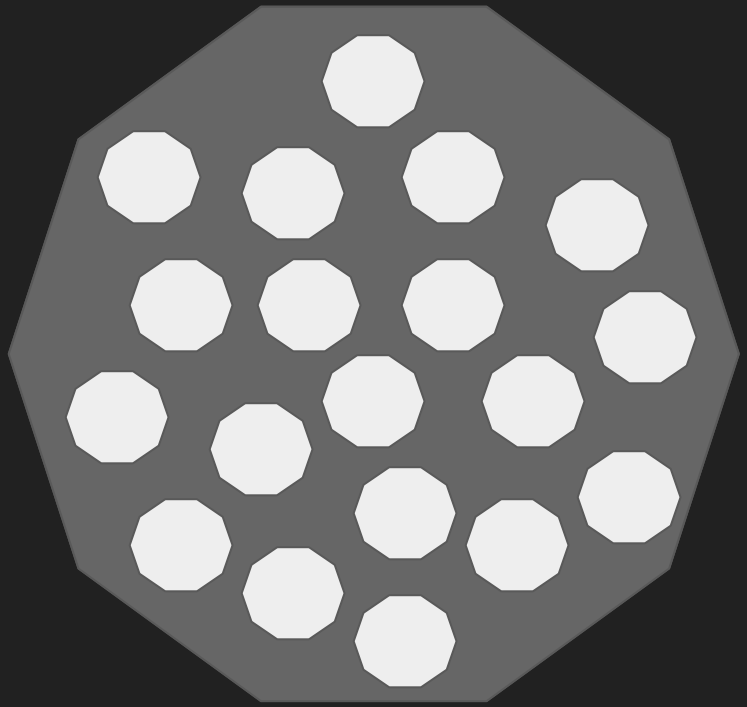
Monolith



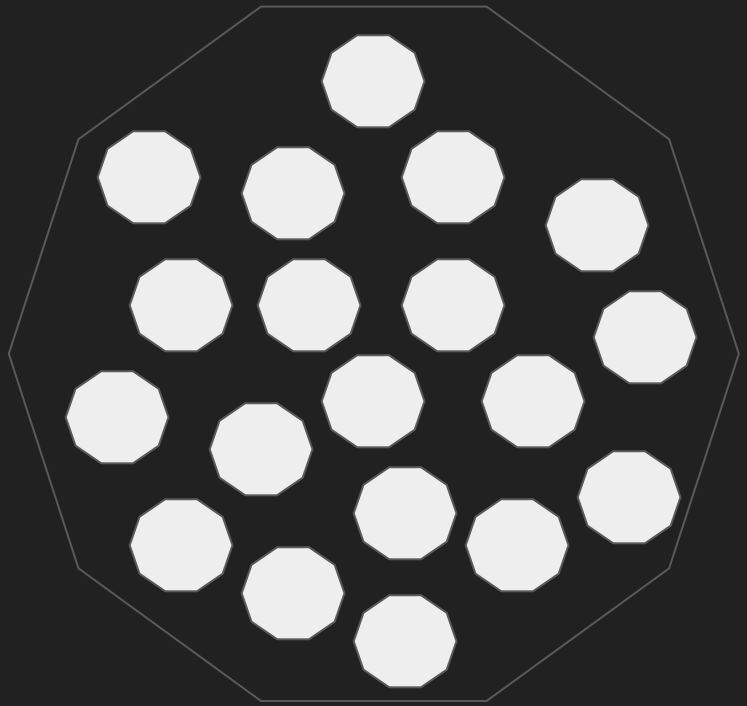
Modules



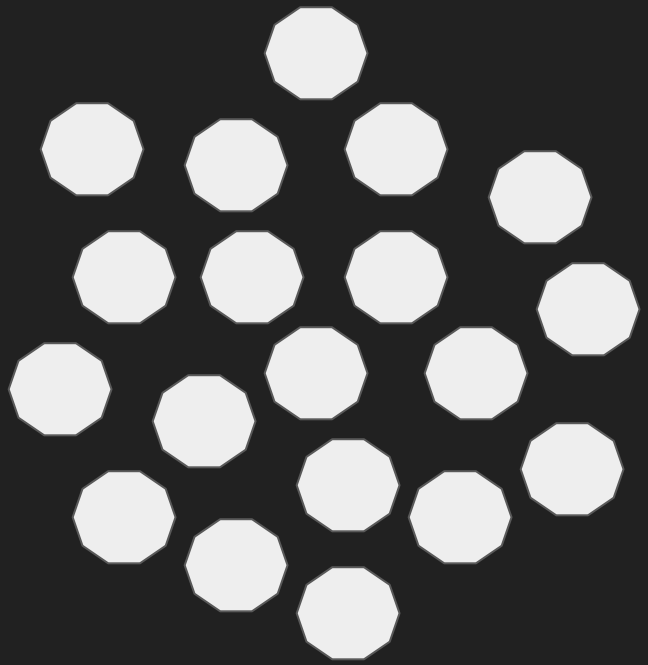
Microservices



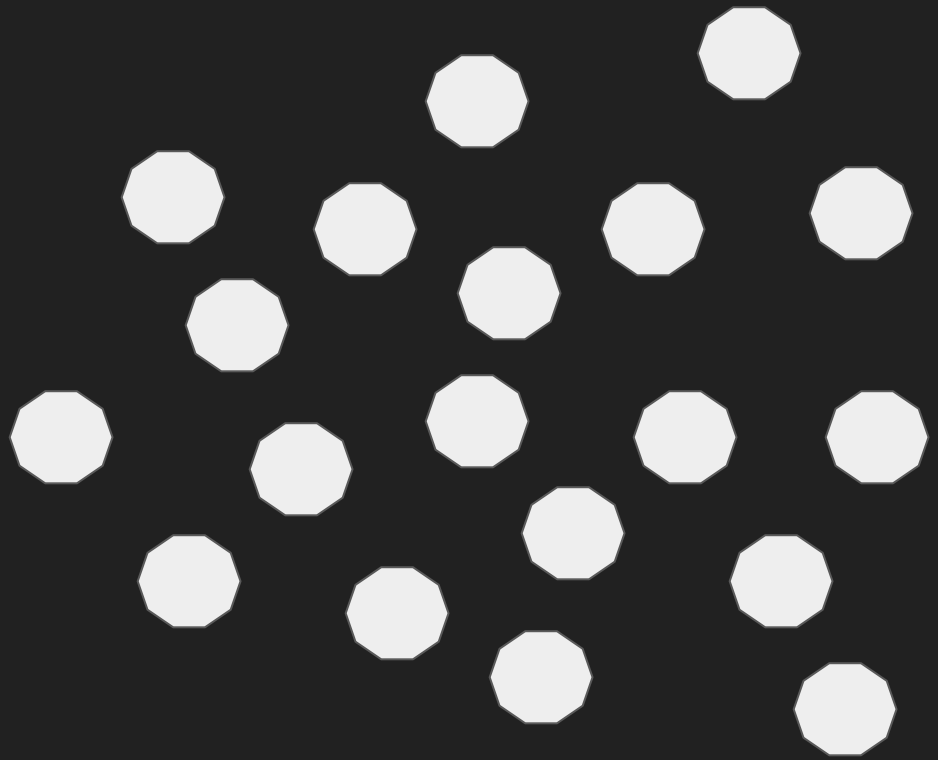
Microservices



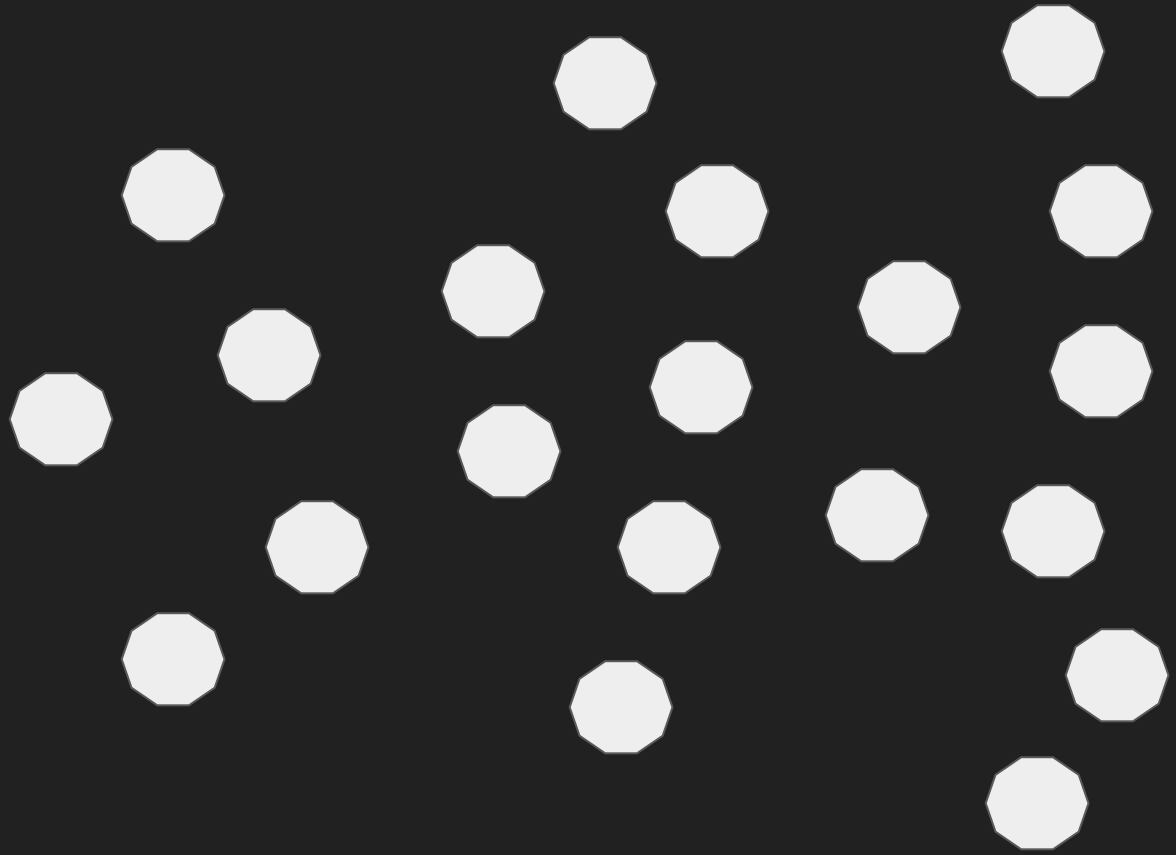
Microservices



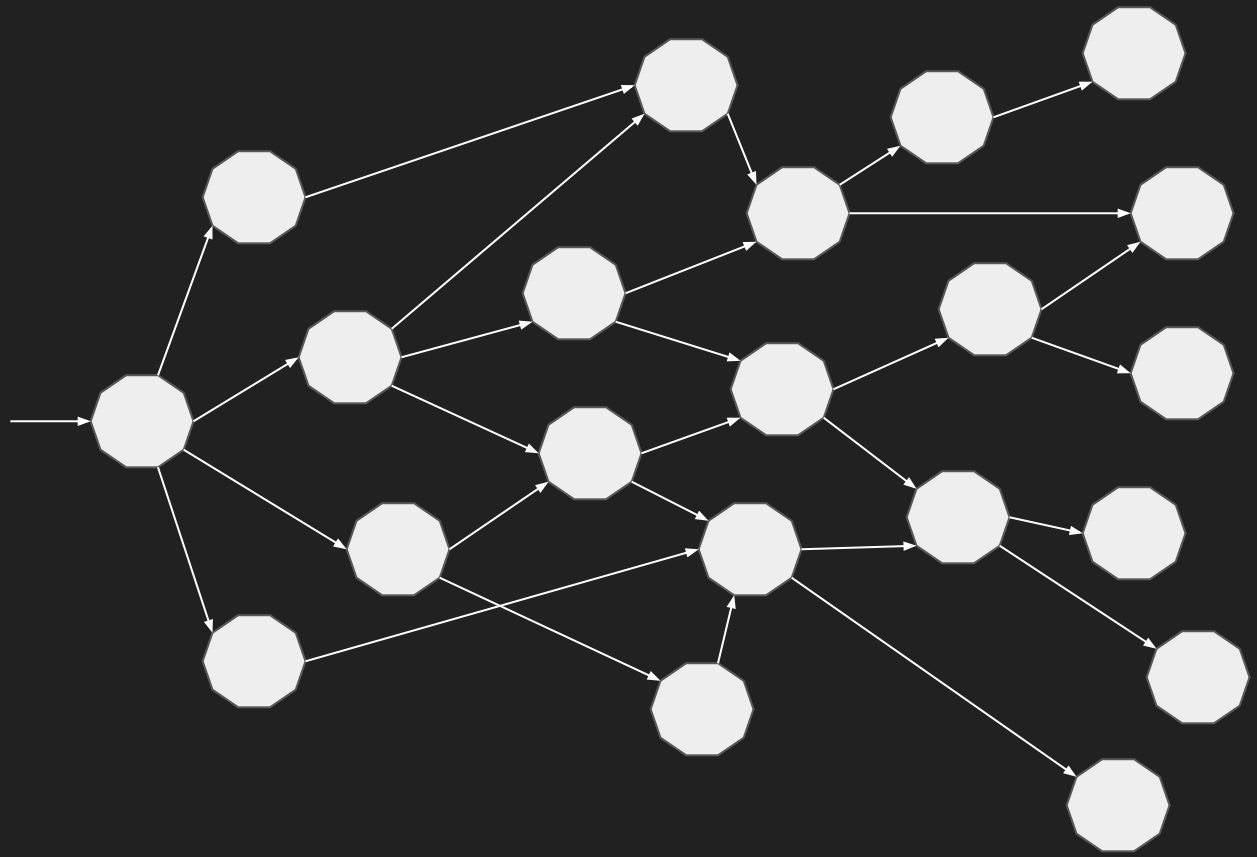
Microservices



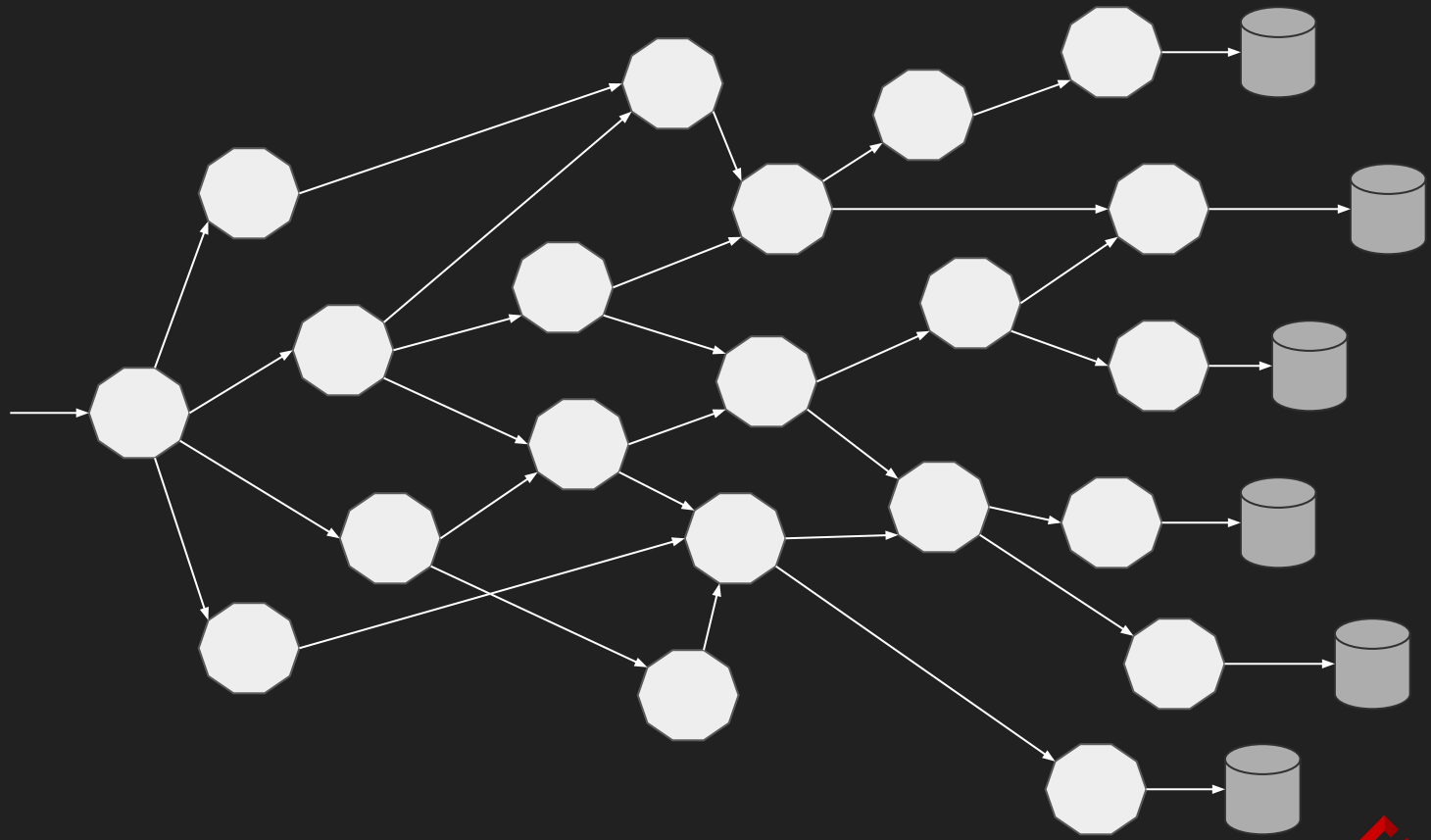
Microservices



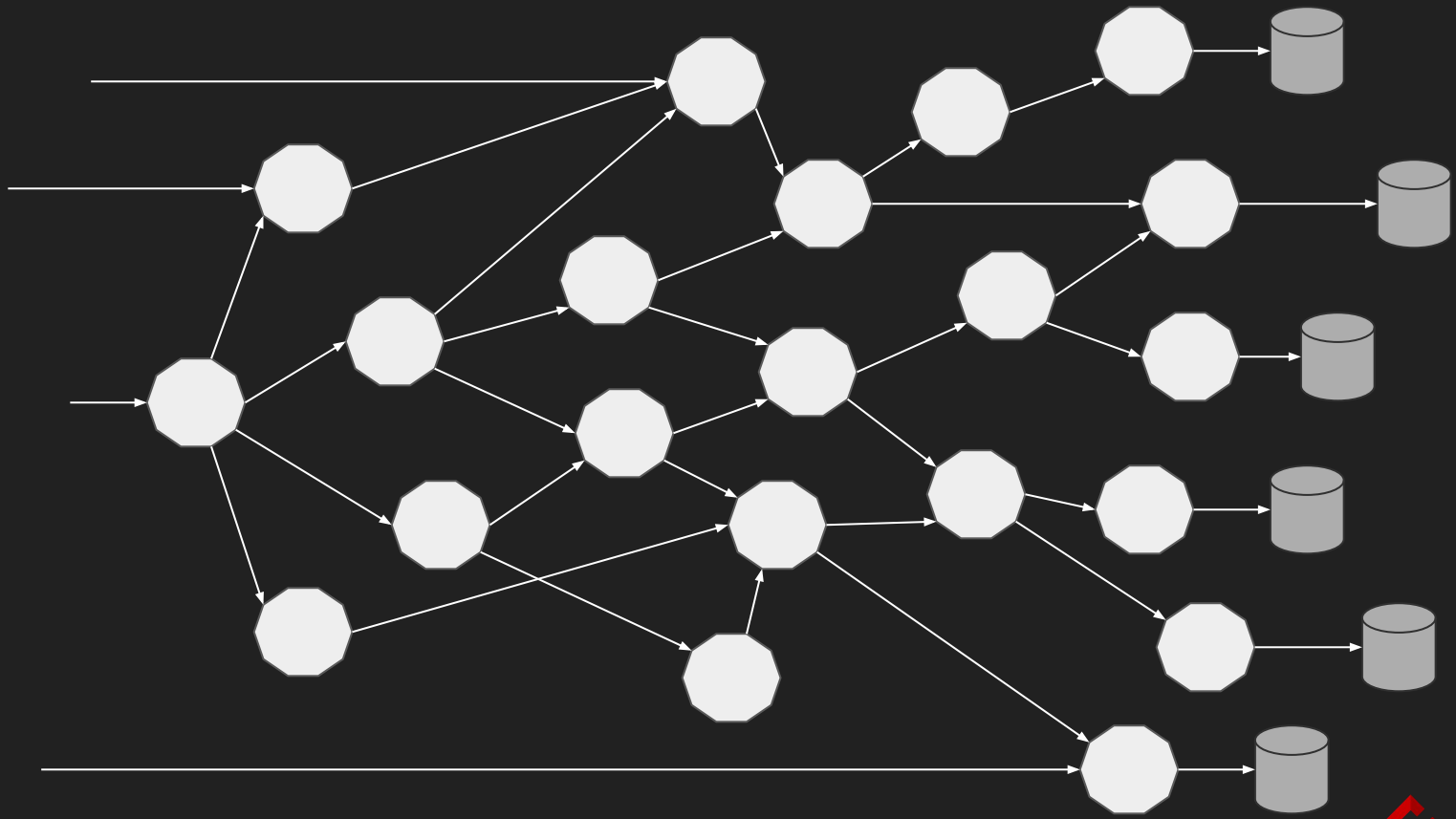
Network of Services - Mesh



Microservices own their Data



Multiple Points of Entry

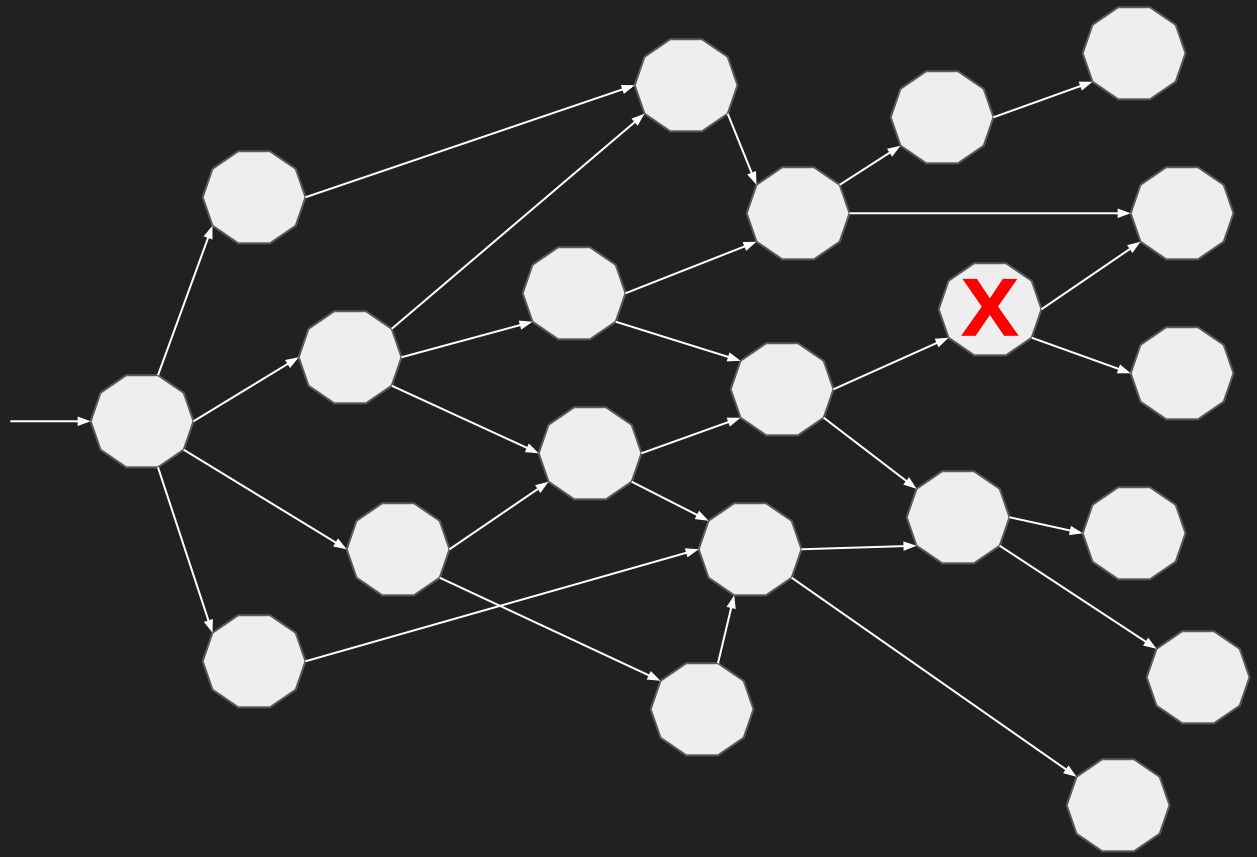


Fallacies of Distributed Computing

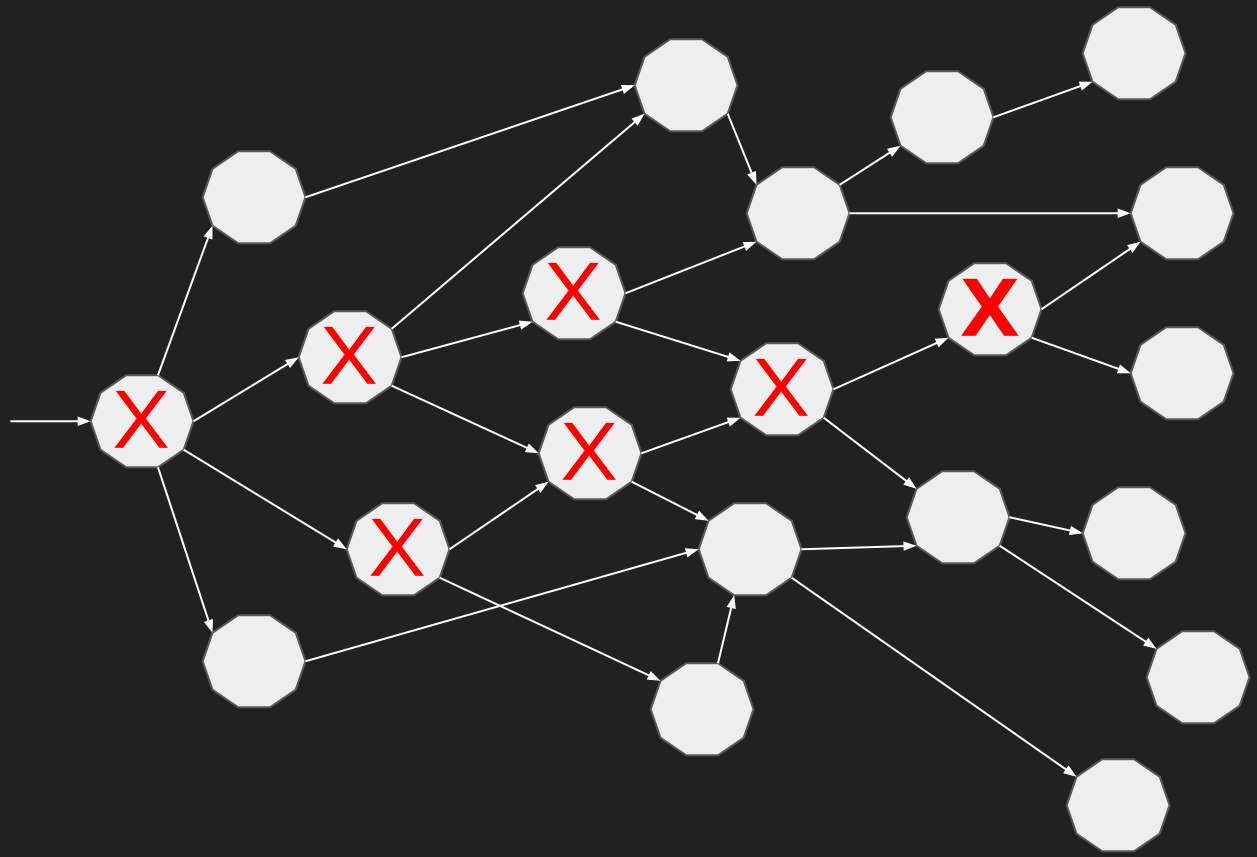
- The Network is Reliable
- Latency is zero
- Bandwidth is infinite
- Topology does not change
- There is one administrator
- Transport cost is zero
- The network is homogeneous

https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

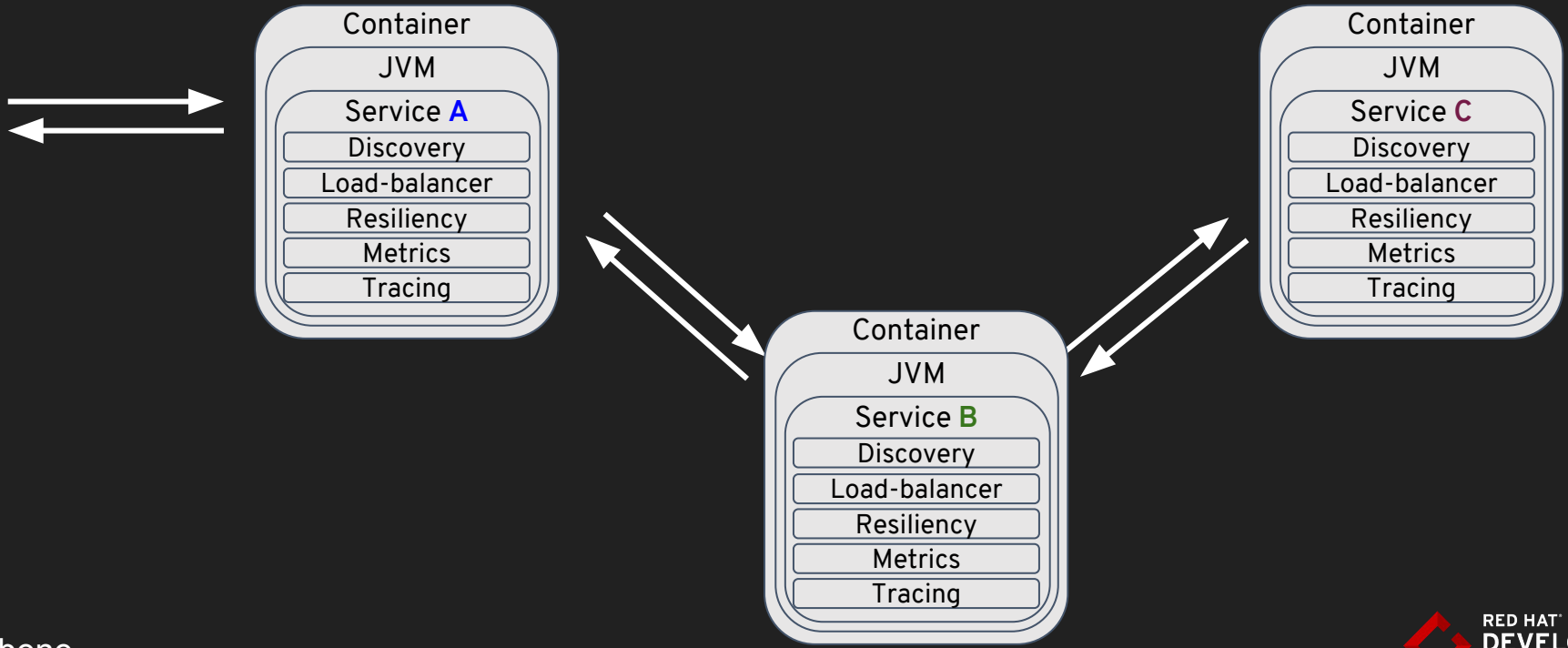
Failure of a Service



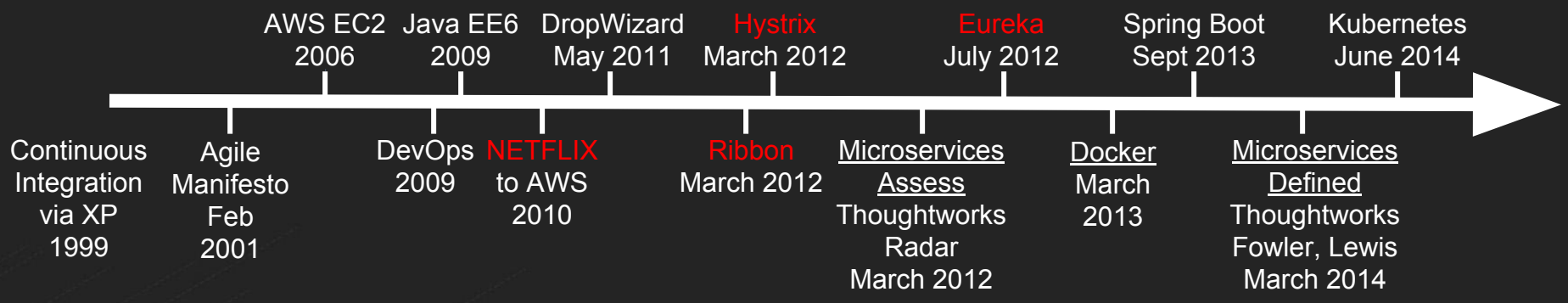
Cascading Failure



Microservices embedding Capabilities



Short History of Microservices



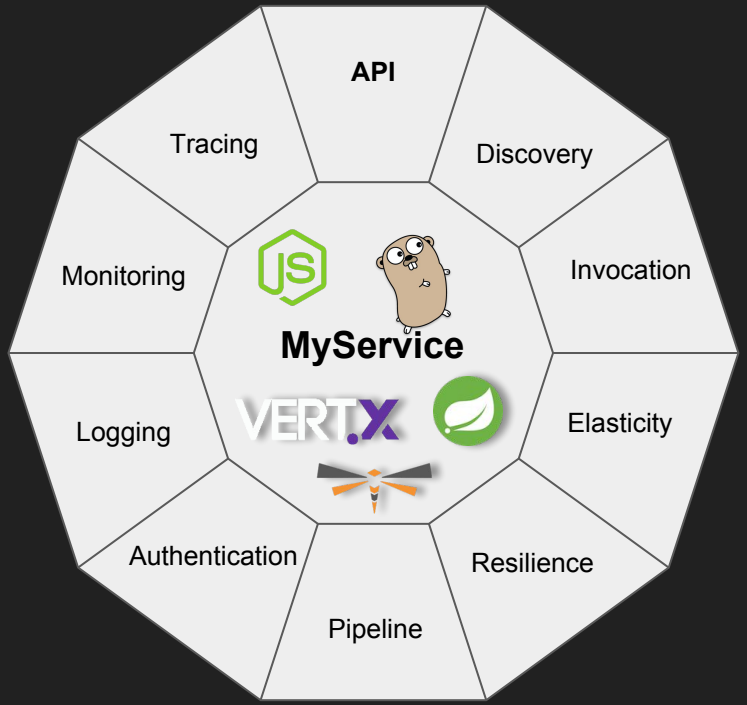
NETFLIX | OSS

What's Wrong with Netflix OSS?

Java Only

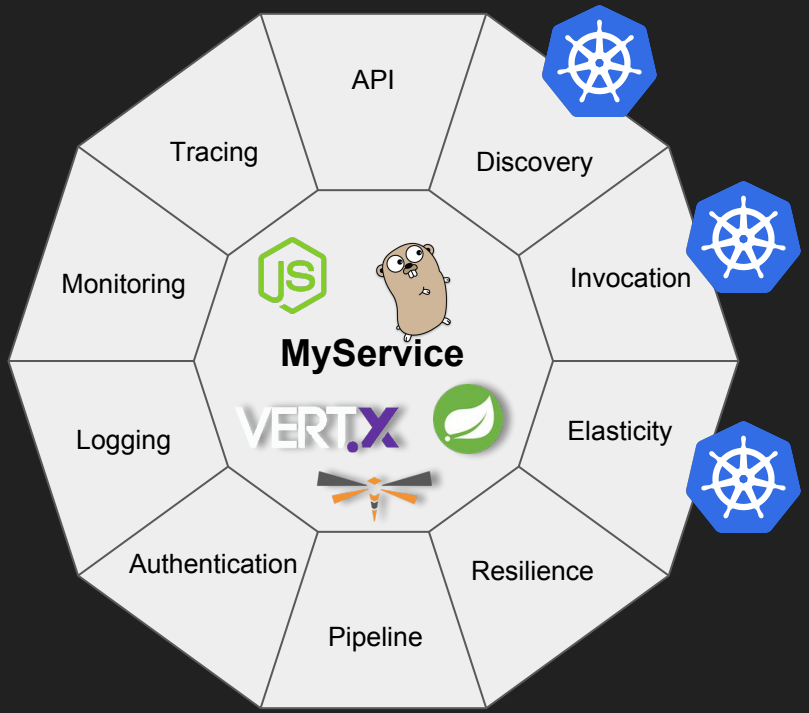
Adds a lot of libraries to YOUR code

Microservices'ilities

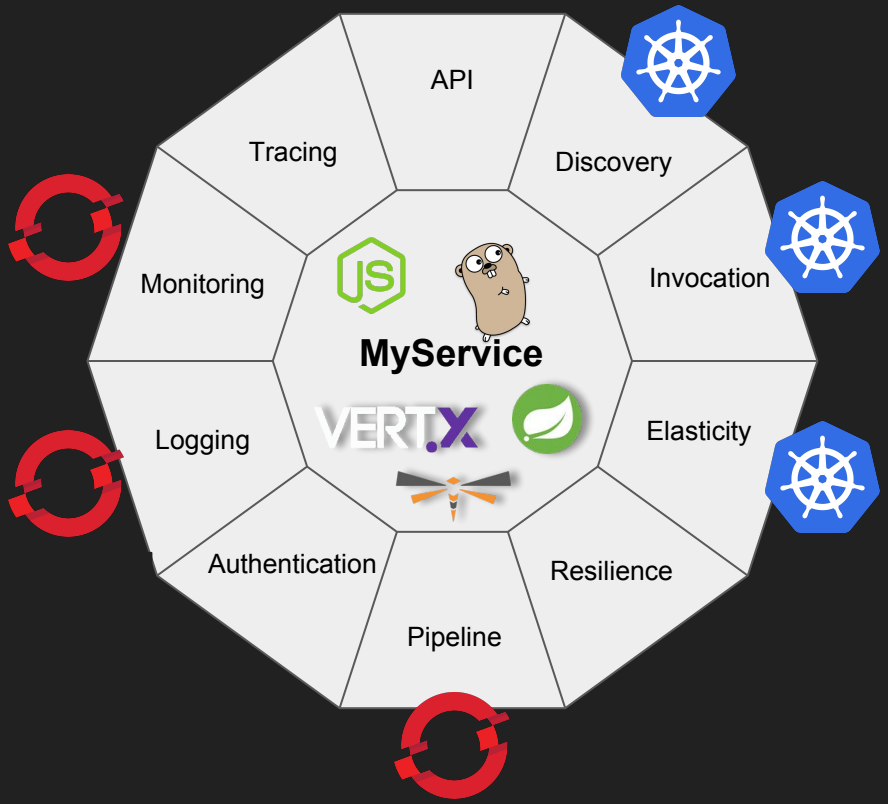


OPENSHIFT

Microservices'ilities + Kubernetes



Microservices'ilities + OpenShift



Istio - Sail

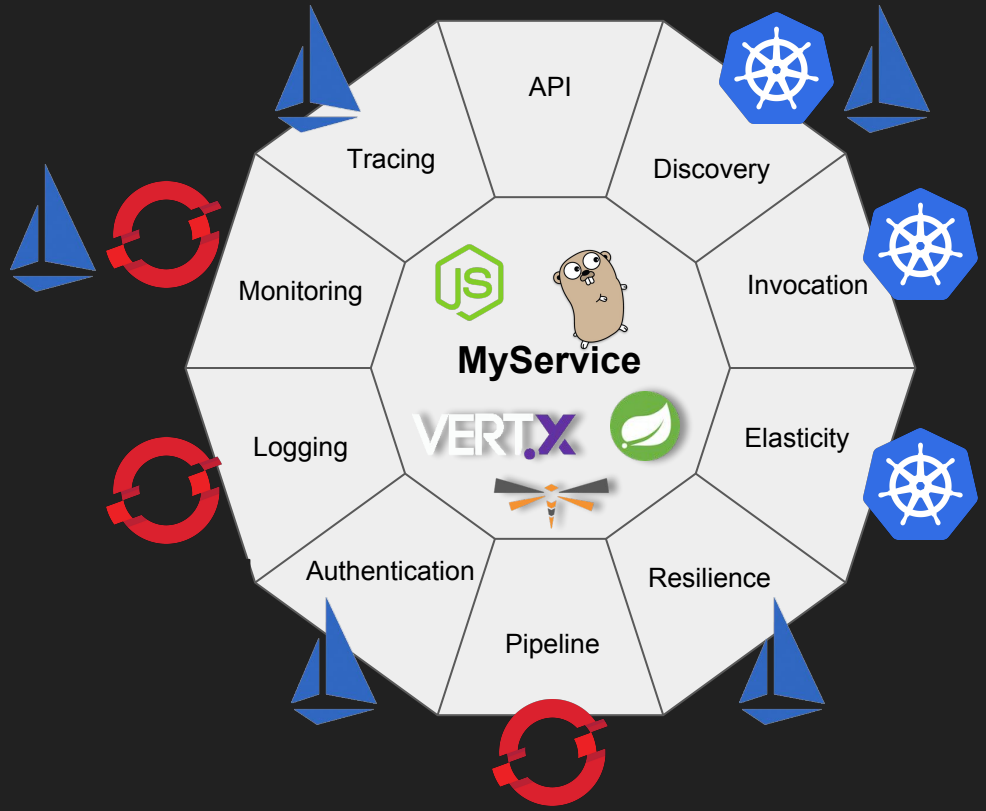
(Kubernetes - Helmsman or ship's pilot)

Service Mesh Defined

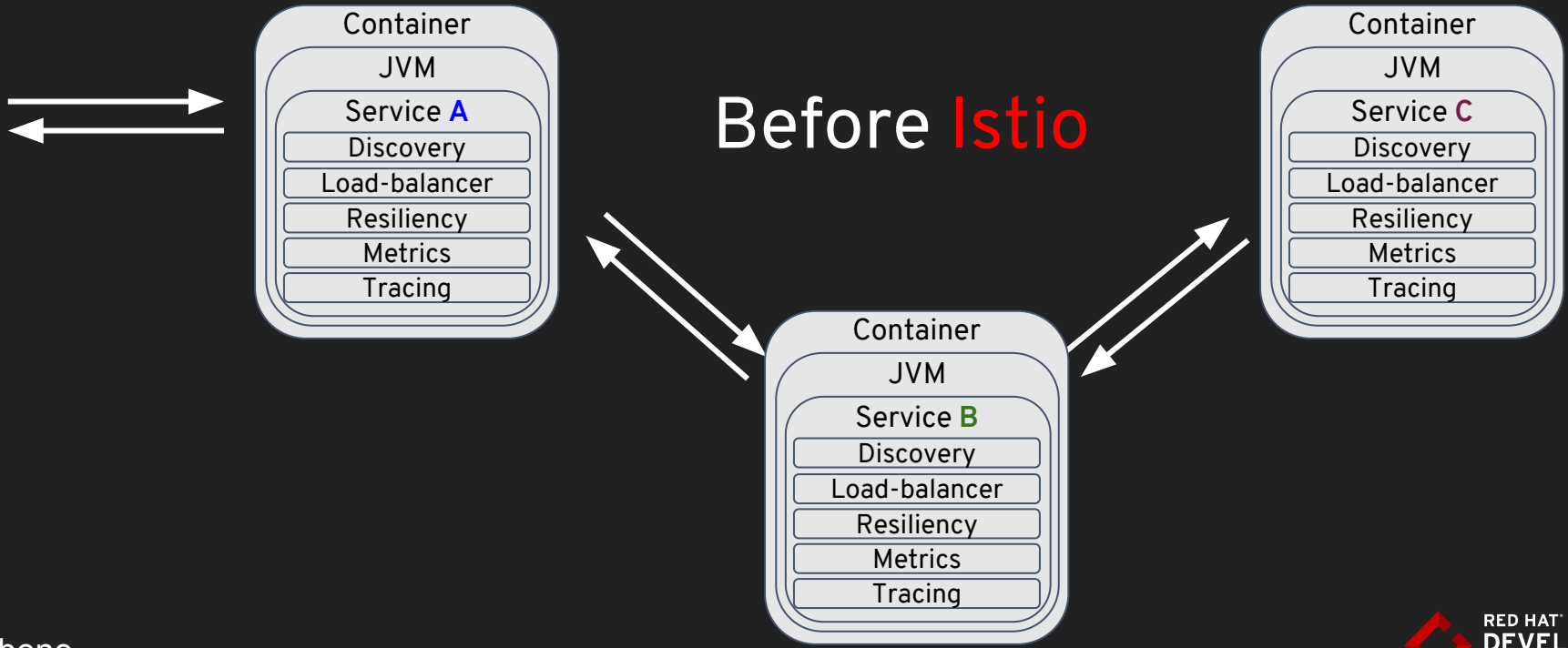
A service mesh is a dedicated infrastructure layer for handling service-to-service communication. It's responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud native application. In practice, the service mesh is typically implemented as an array of lightweight network proxies that are deployed alongside application code, without the application needing to be aware

https://buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one/

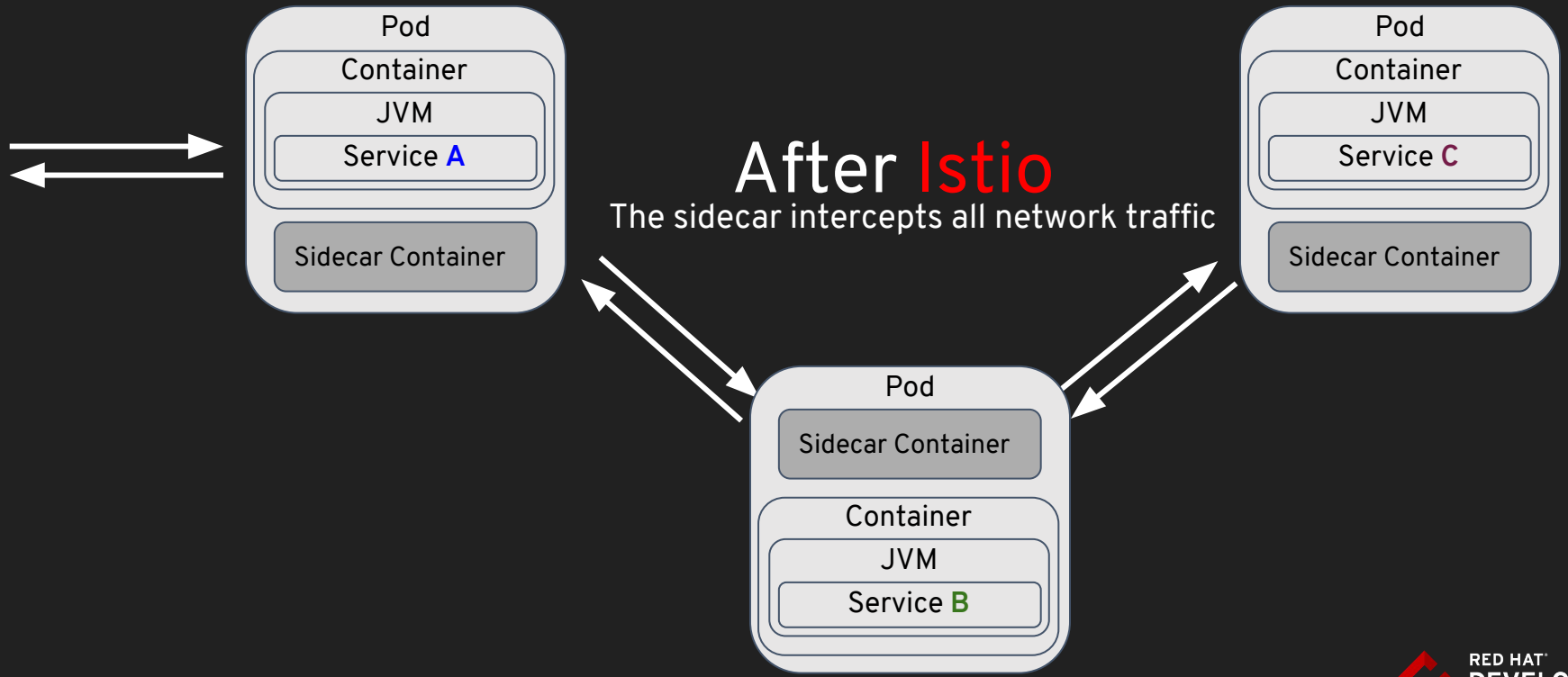
Microservices'ilities + Istio



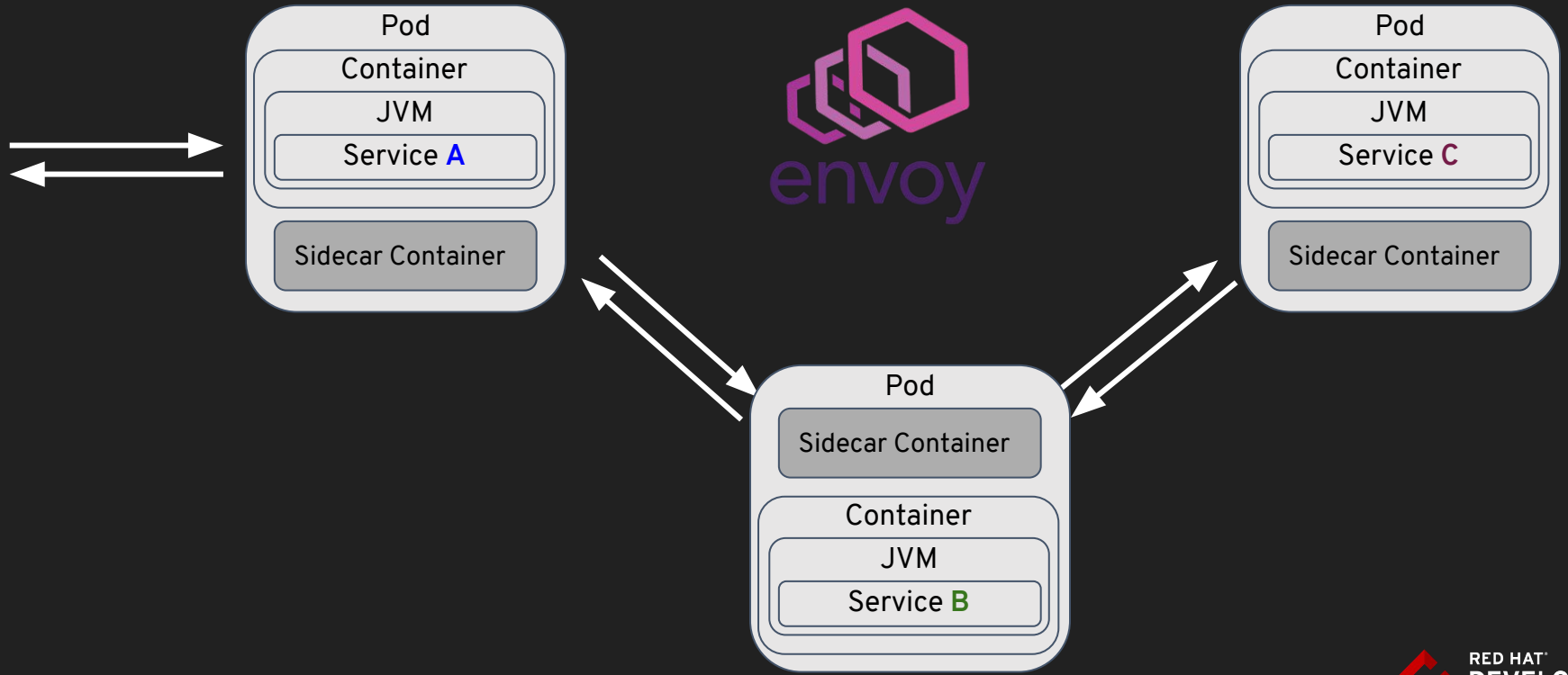
Microservices embedding Capabilities



Microservices externalizing Capabilities



Envoy is the current sidecar

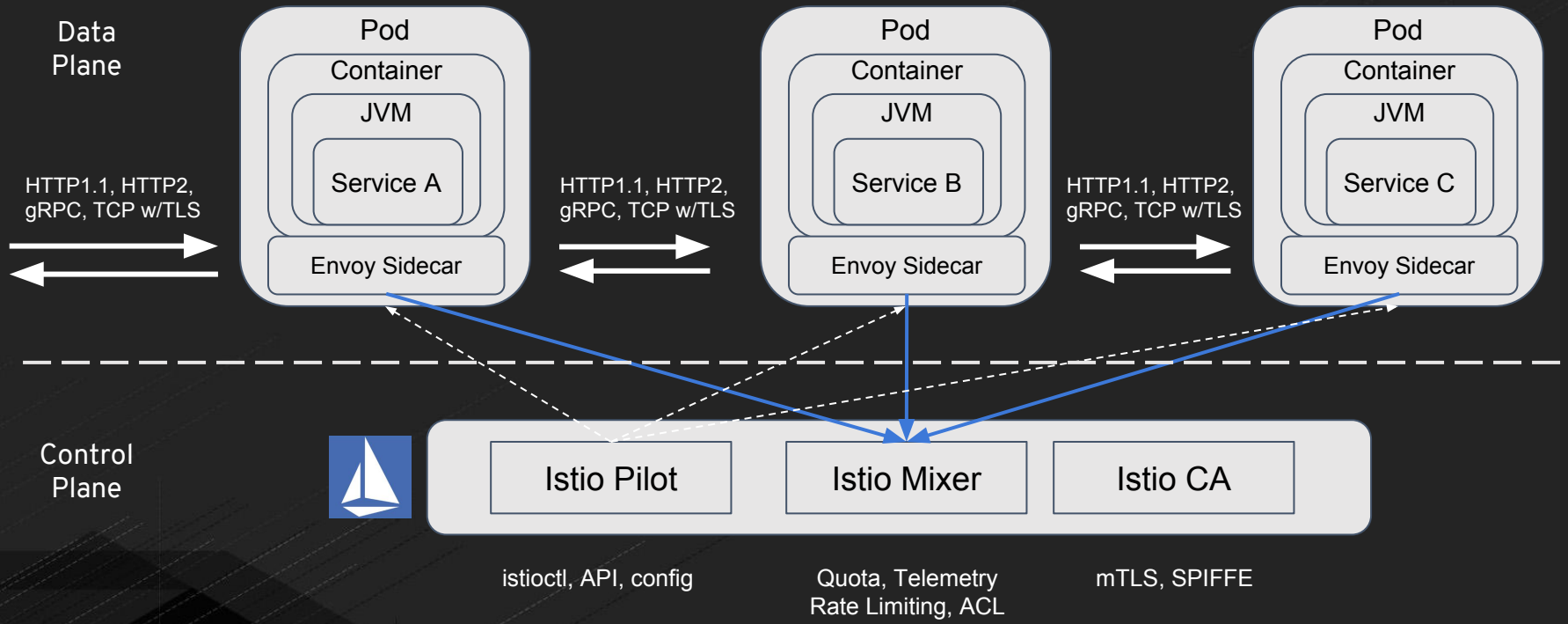


Next Generation Microservices - Service Mesh

Code Independent (Polyglot)

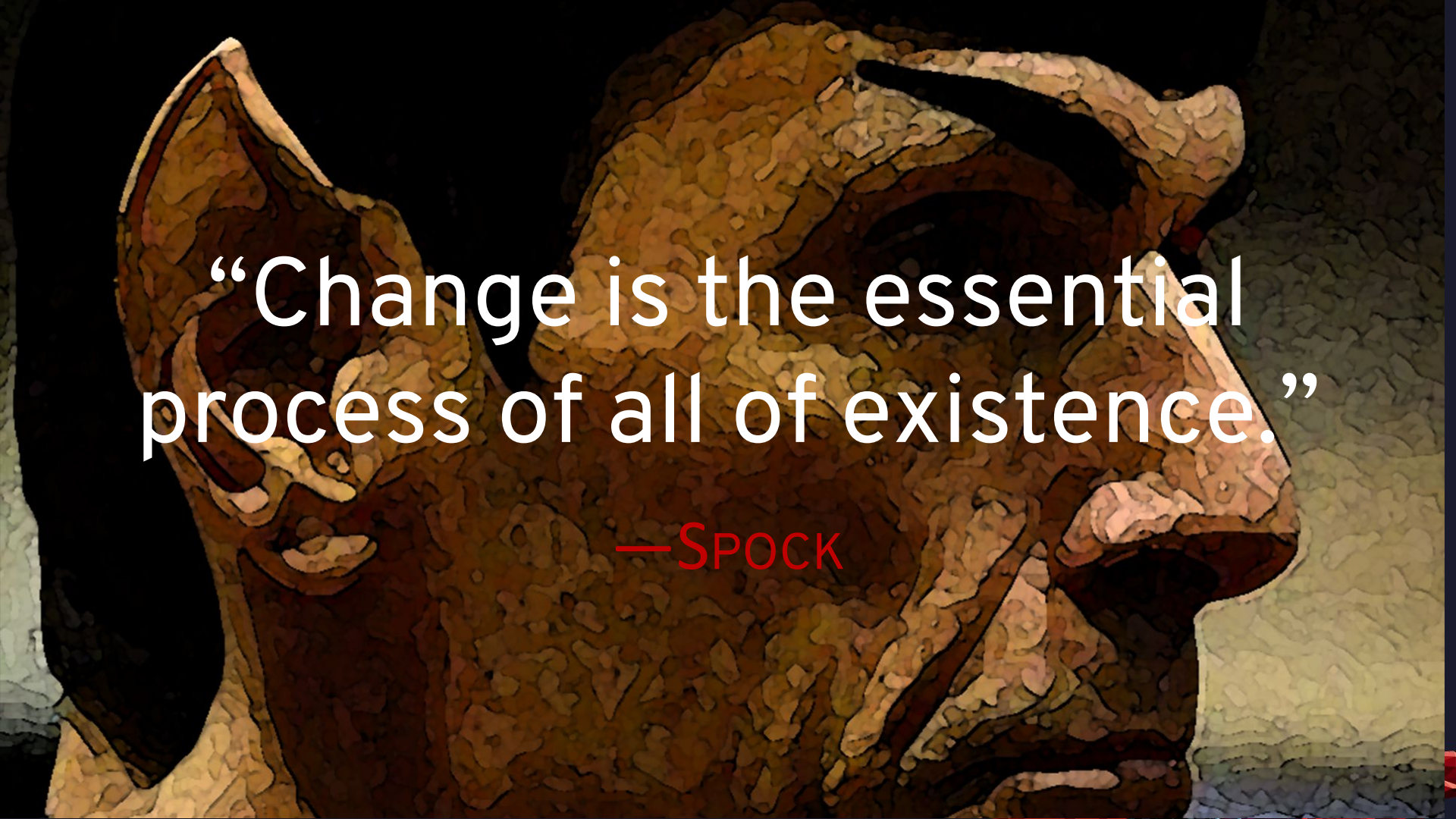
- Intelligent Routing and Load-Balancing
  - A/B Tests
  - Smarter Canary Releases
- Chaos: Fault Injection
- Resilience: Circuit Breakers
- Observability: Metrics and Tracing
- Fleet wide policy enforcement

Istio Data Plane vs Control Plane

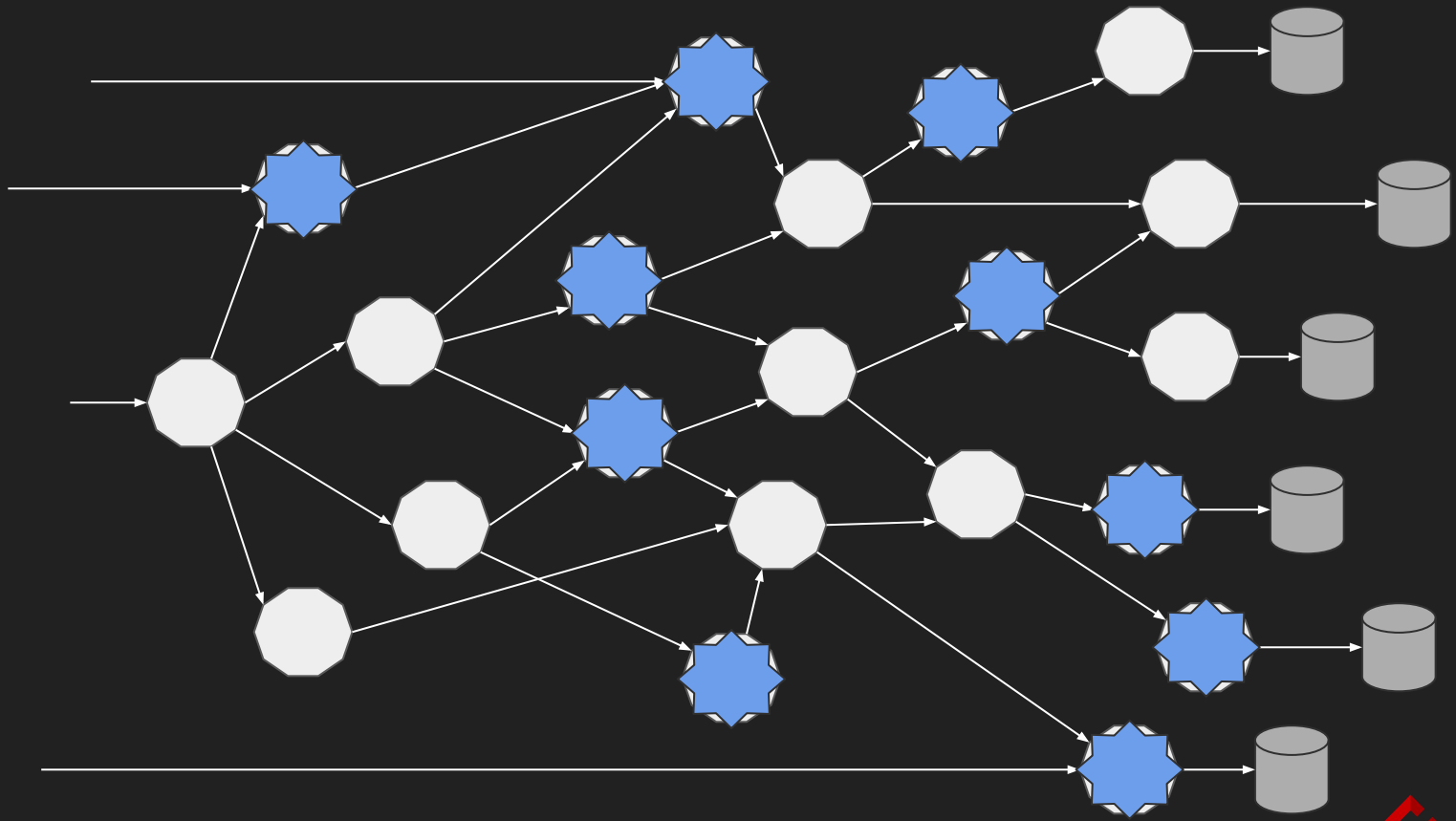


Demo

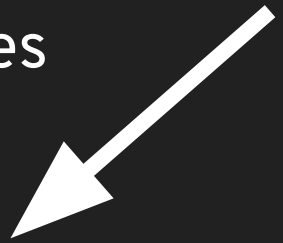
bit.ly/istio-tutorial


“Change is the essential process of all of existence.”
—Spock

Let there be Functions



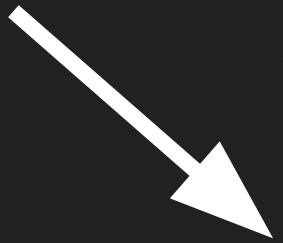
Microservices



Your Control
Long-Lived Processes
Known Programming Model
Often Sync Request-Response

Mature:
IDE Integration
Debuggers
Tracers
Monitoring
CI/CD

Serverless Functions



Cloud Control
Short-Lived Processes
New Programming Model
Event-Driven Async

Immature:
?

@rafabene

Serverless Defined

Serverless architectures refer to applications that significantly depend on third-party services (knows as **Backend as a Service or "BaaS"**) or on custom code that's run in **ephemeral containers** (**Function as a Service or "FaaS"**), the best known vendor host of which currently is AWS Lambda. By using these ideas, and by moving much behavior to the front end, such architectures remove the need for the traditional '**always on**' server system sitting behind an application. Depending on the circumstances, such systems can significantly **reduce operational cost** and **complexity** at a cost of vendor dependencies and (at the moment) immaturity of supporting services.

https://martinfowler.com/articles/serverless.html

It is Serverless, because of SaaS (managed by another party services).

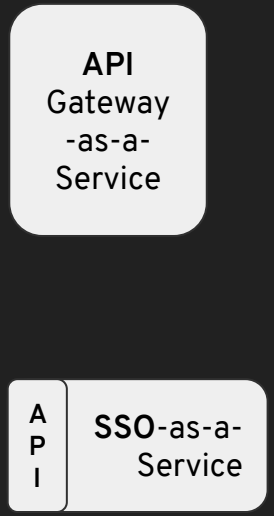
The first question is “is there a suitable service I can consume?” before “is there something I can buy and set up using a cloud provider?”.

Thoughtworks Nov 29 2017

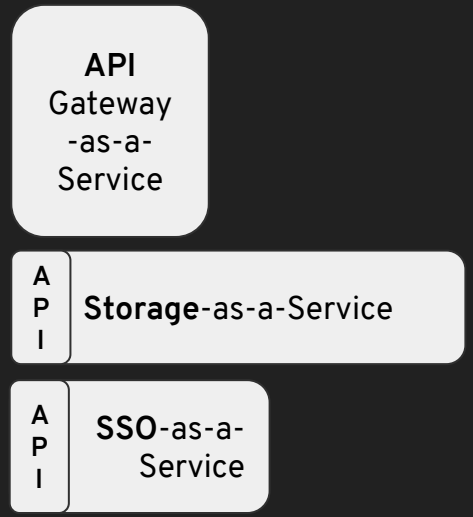
It is all about the Services

HTTP Input/Output Service

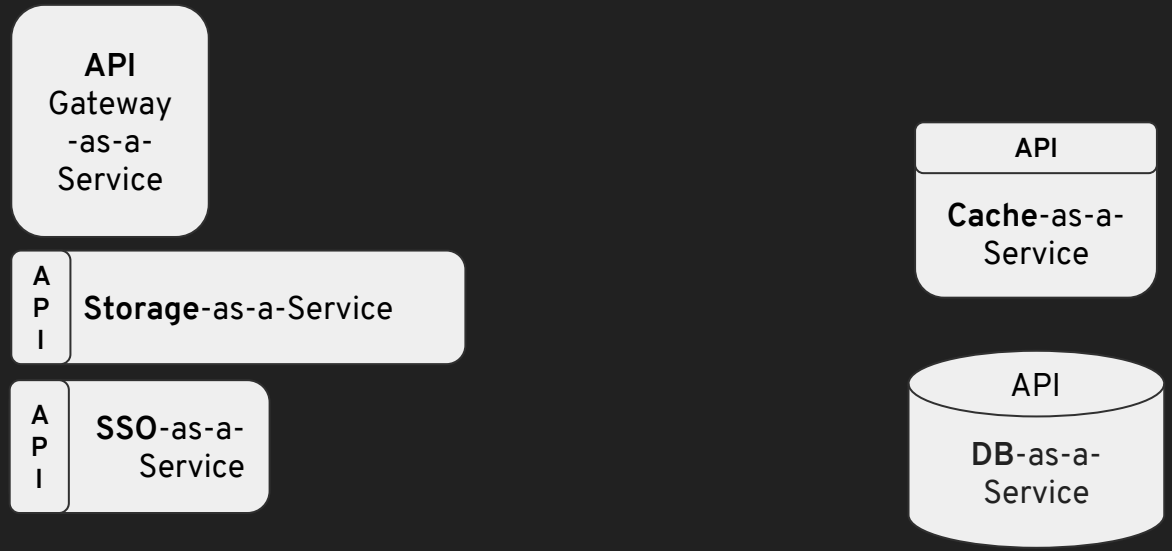
Authentication Service



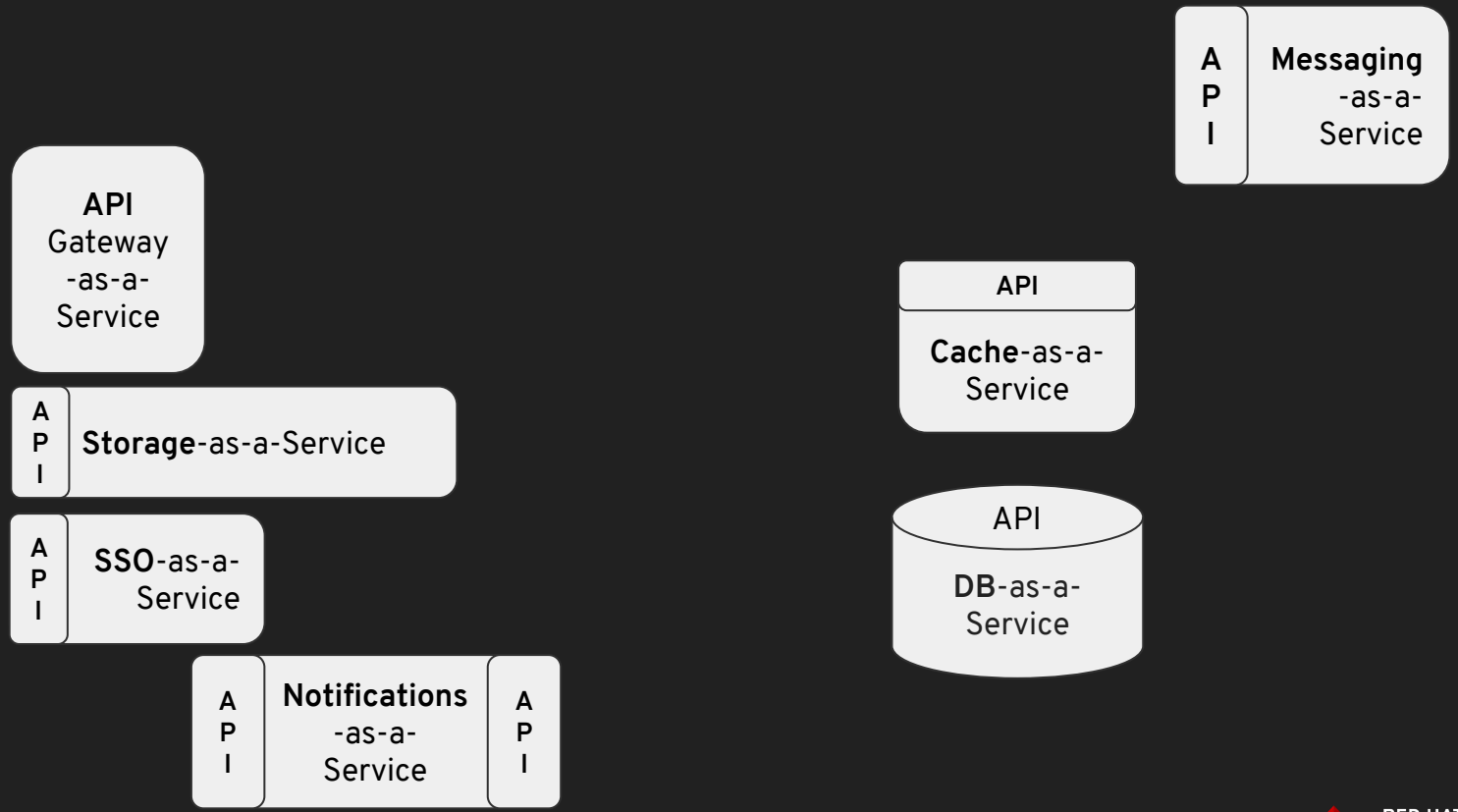
File Storage Service



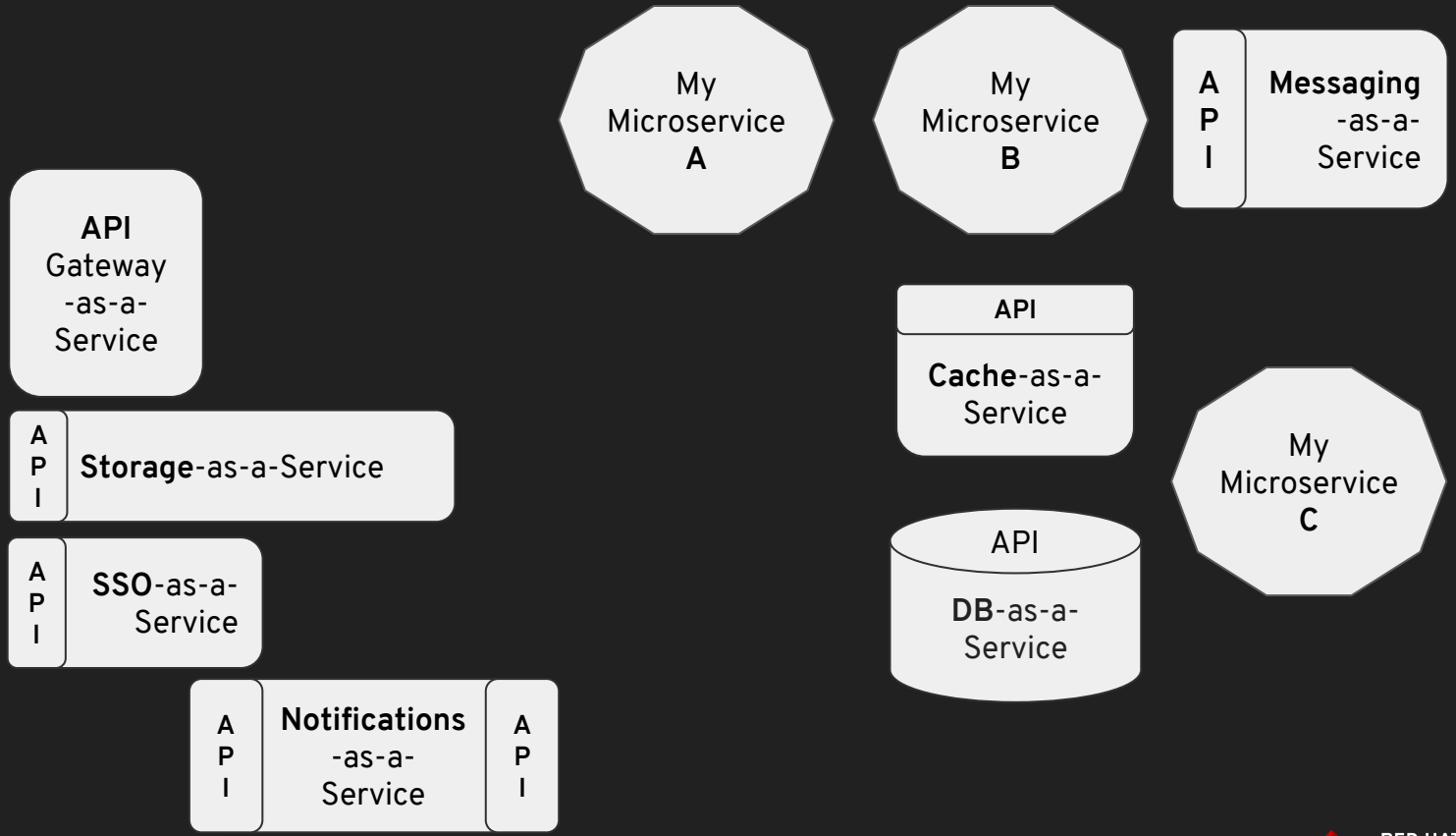
Data Services



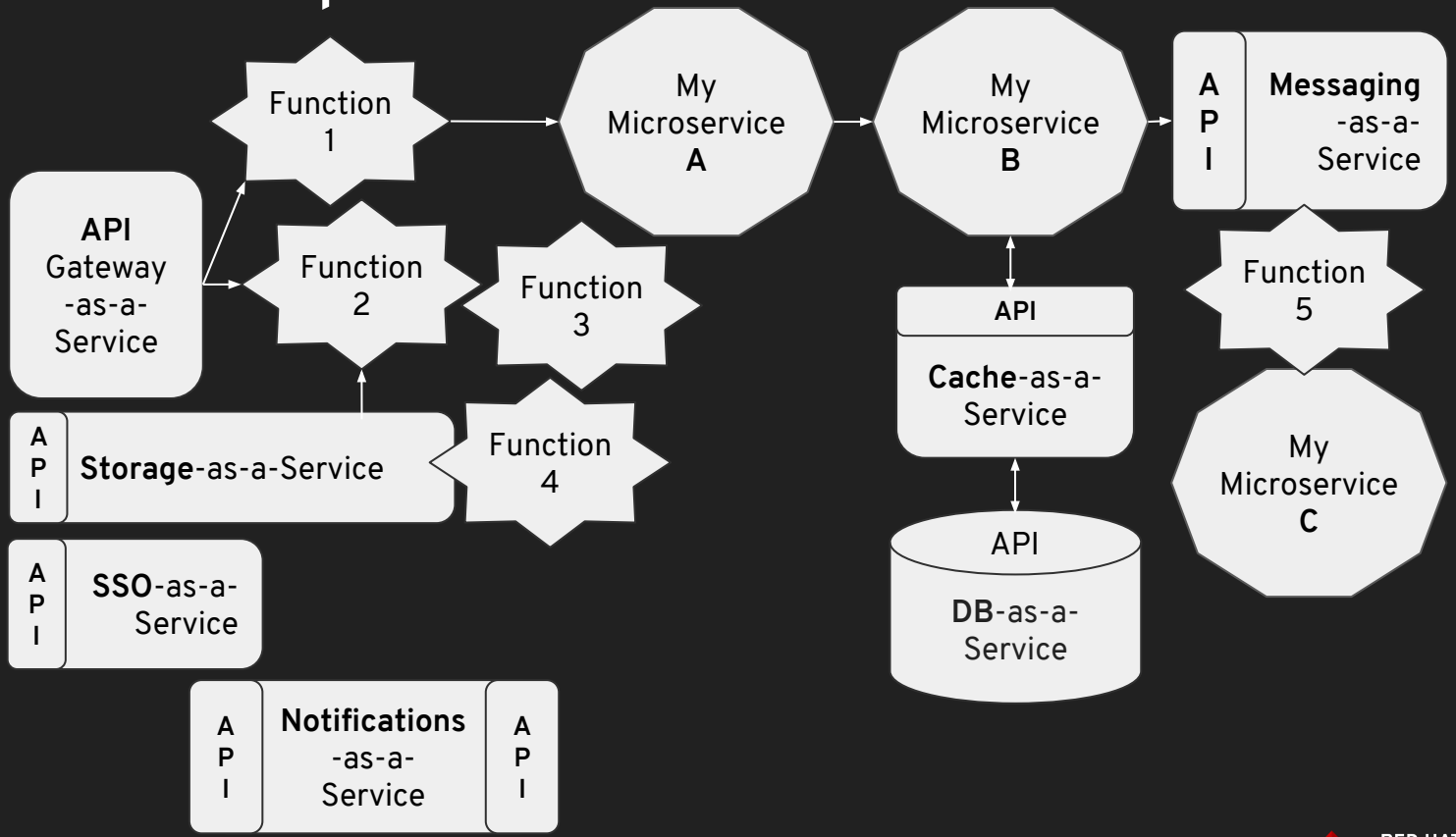
Connectivity Services



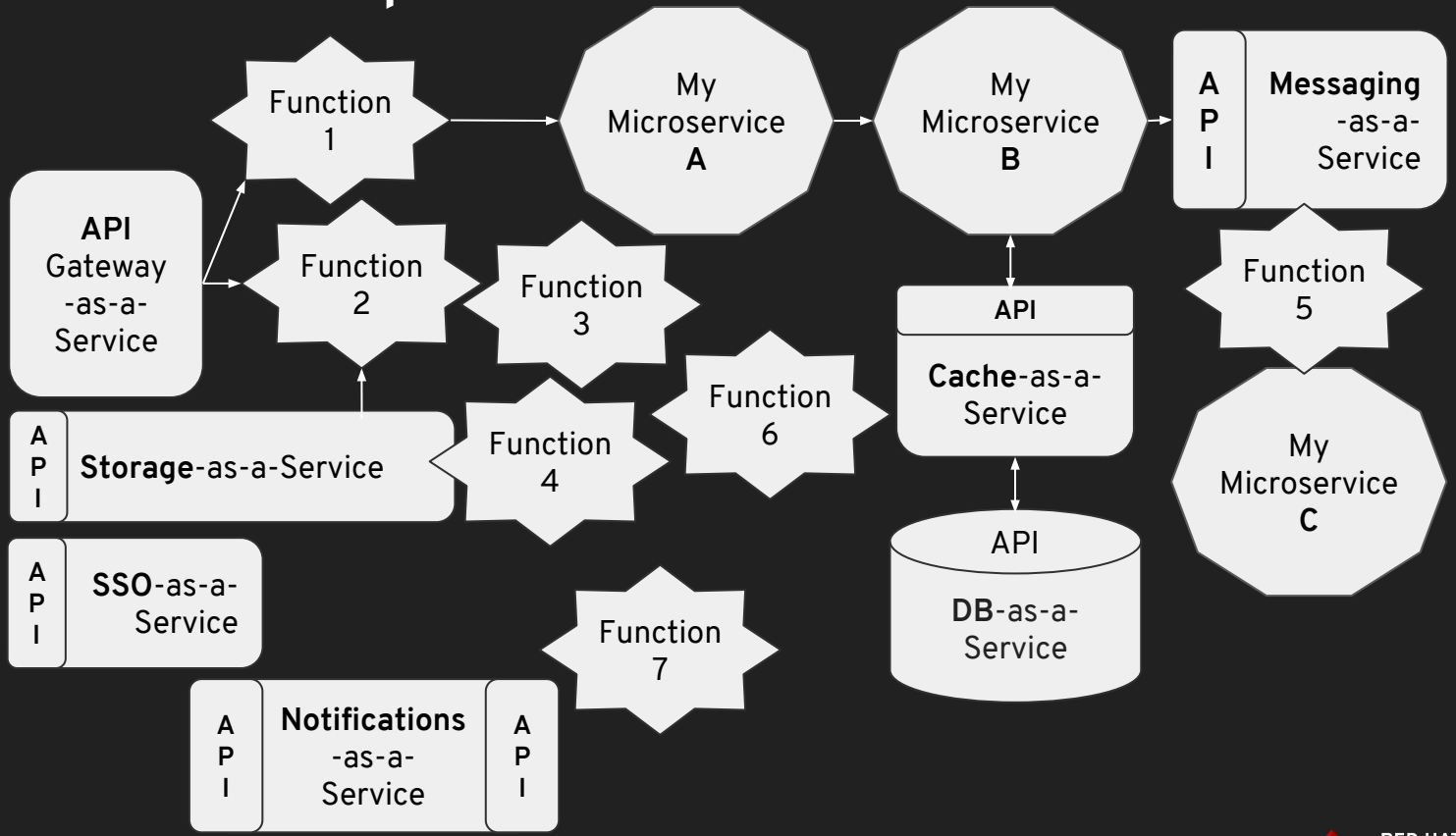
Your Containerized Services



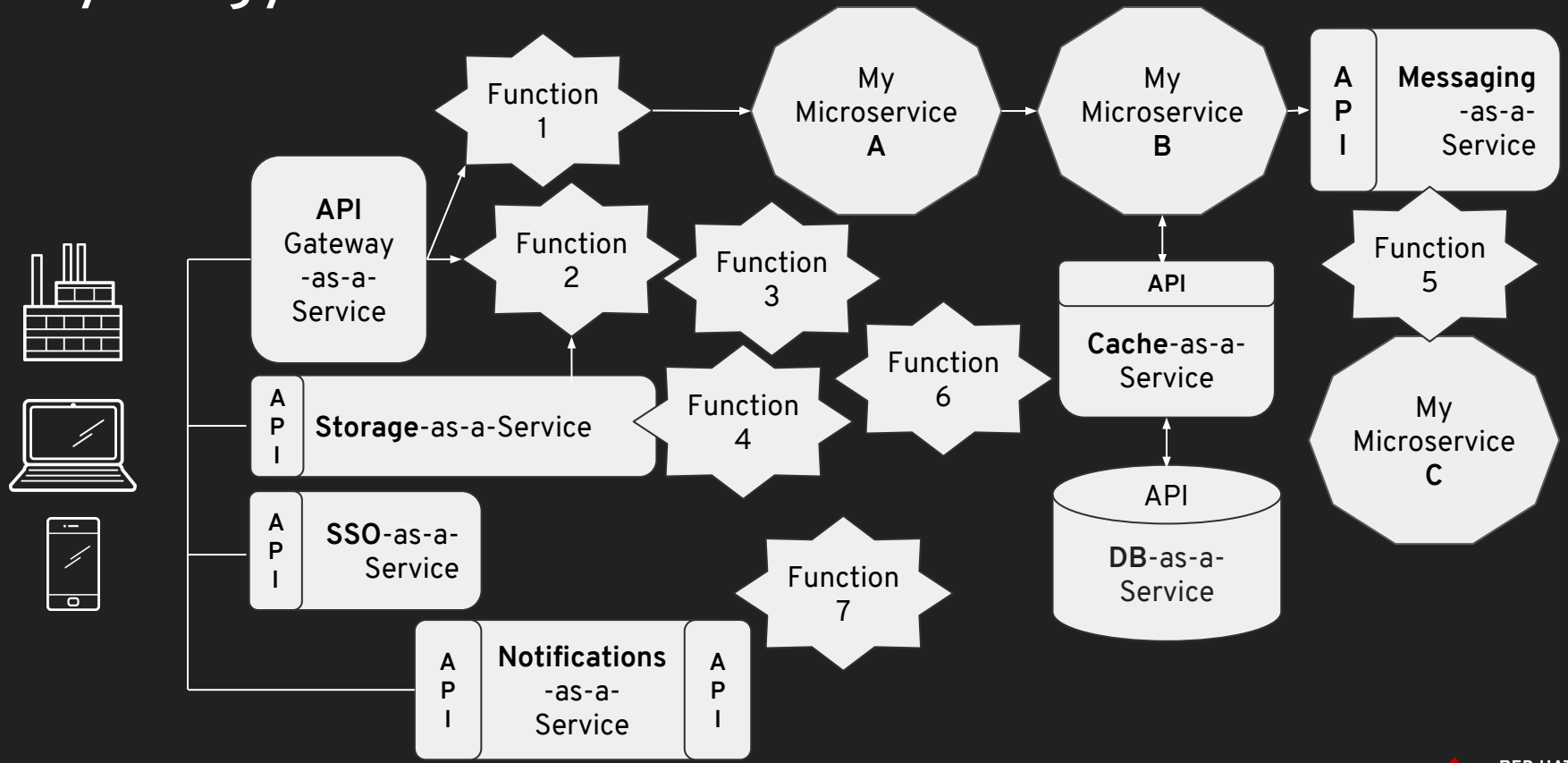
Event-Driven Input



Event-Driven Output

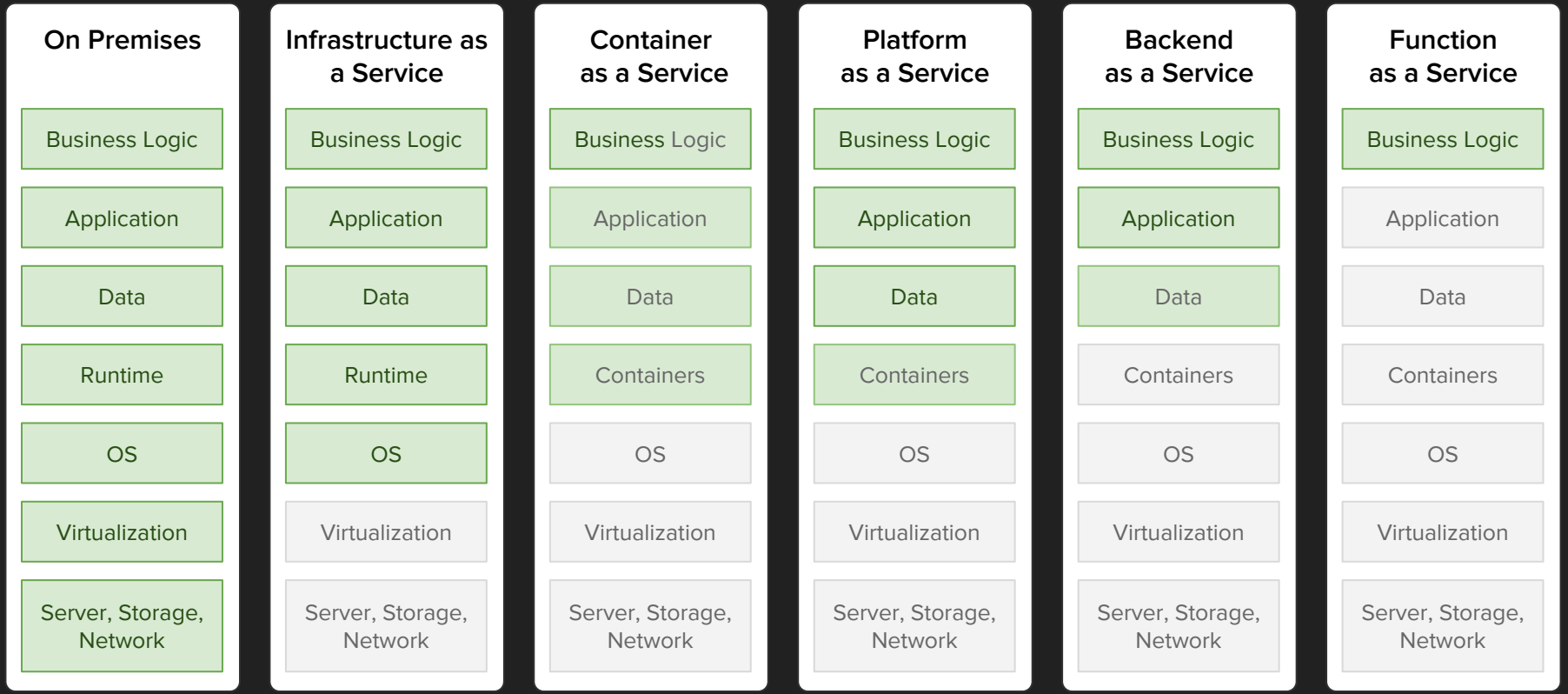


Synergy



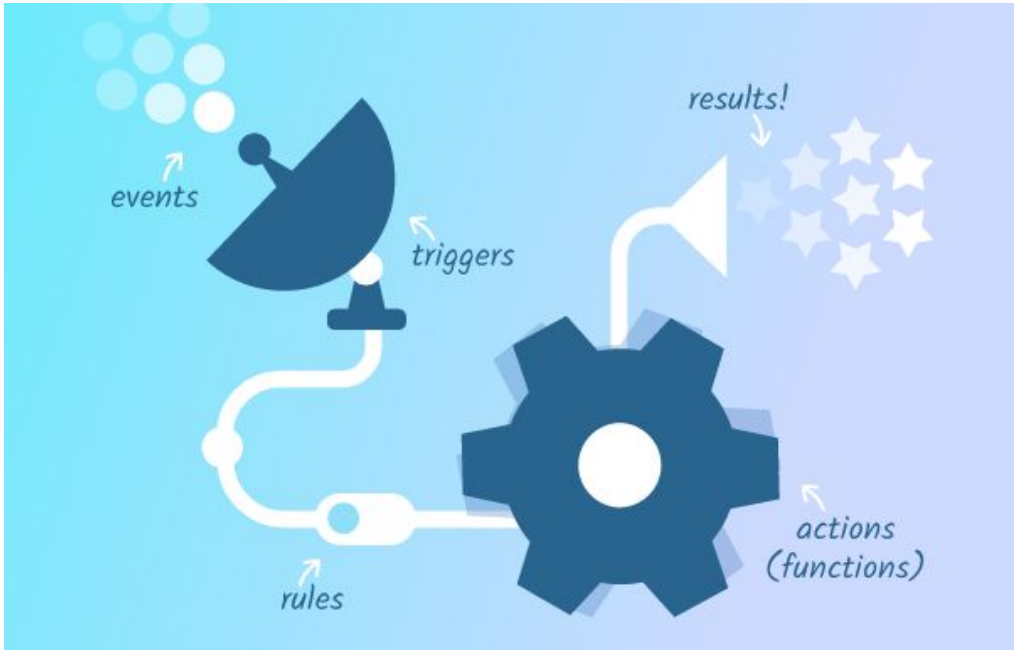
FaaS Kubernetes Players

Platform Responsibilities



Apache OpenWhisk

- Open Source incubating under Apache
- A FaaS platform to execute functions written in:
  - JavaScript
  - Swift
  - Java
  - Python
  - PHP
  - Docker
  - Go
- Deployable on
  - Any platform where docker can be run
  - Kubernetes/OpenShift



Action (aka Function) JavaScript

```
function main(params) {
    console.log("JS msgaction args: %j", params);
    Object.keys(params["body"]).forEach(function(key) {
        console.log("%s: %j", key, params["body"][key])
    });
    return { payload: 'JavaScript on OpenWhisk on OpenShift ' + params.body.text};
}
```

```
wsk -i action update msgaction msgaction.js

wsk -i action invoke msgaction --param body '{"text":"my stuff"}' --result

{ “payload”: “JavaScript OpenWhisk on OpenShift my stuff” }

wsk -i activation get -l  # see the details, logs
```

Action (aka Function) Python

```
def main(params):
    print "Py msgaction params: " + str(params)
    for key in params["body"]:
         print key, params["body"][key]
    return { "payload" :
      "Python on OpenWhisk on OpenShift " + str(params["body"]["text"])}
```

```
wsk -i action update msgaction msgaction.py

wsk -i action invoke msgaction --param body '{"text":"my stuff"}' --result

{ “payload”: “Python on OpenWhisk on OpenShift my stuff” }

wsk -i activation get -l  # see the details, logs
```

maven archetype

```
git clone https://github.com/apache/incubator-openwhisk-devtools

cd incubator-openwhisk-devtools/java-action-archetype

mvn -DskipTests clean install
```

bit.ly/faas-tutorial

learn.openshift.com/serverless

maven archetype

```
mvn archetype:generate \
  -DarchetypeGroupId=org.apache.openwhisk.java \
  -DarchetypeArtifactId=java-action-archetype \
  -DarchetypeVersion=1.0-SNAPSHOT \
  -DgroupId=com.example \
  -DartifactId=actionjava \
  -Dversion=1.0-SNAPSHOT \
  -DinteractiveMode=false
```

bit.ly/faas-tutorial

learn.openshift.com/serverless

Action (aka Function) Java

```
public static JsonObject main(JsonObject params) {
   System.out.println("msgaction params: " + params);
   JsonObject body = params.get("body").getAsJsonObject();
   for (Map.Entry<String, JsonElement> entry : body.entrySet()) {
     System.out.println(entry.getKey() + " " + entry.getValue().getAsJsonPrimitive().getAsString());
   }
   JsonObject response = new JsonObject();
   response.addProperty("payload", "Java on OpenWhisk on OpenShift " +
        params.getAsJsonObject("body").get("text").getAsString());
   return response;
 }
```

```
mvn clean package

wsk -i action update msgaction target/actionjava.jar --main com.example.FunctionApp

wsk -i action invoke msgaction --param body '{"text":"my stuff"}' --result

{ “payload”: “Java on OpenWhisk on OpenShift my stuff” }

wsk -i activation get -l  # see the details, logs
```

Web action

wsk -i action update msgaction target/actionjava.jar --main com.example.FunctionApp --web=true

wsk -i action get msgaction --url

Sequence Action

```
wsk -i action invoke redhat-developers-demo/splitter --param text
"zebra,cat,antelope" --result | tee ~/split.json
```

```
wsk -i action invoke redhat-developers-demo/sorter --param-file  ~/split.json
--result | tee ~/sorted.json
```

```
wsk -i action invoke redhat-developers-demo/uppercase --param-file ~/sorted.json
--result
```

Sequence Action

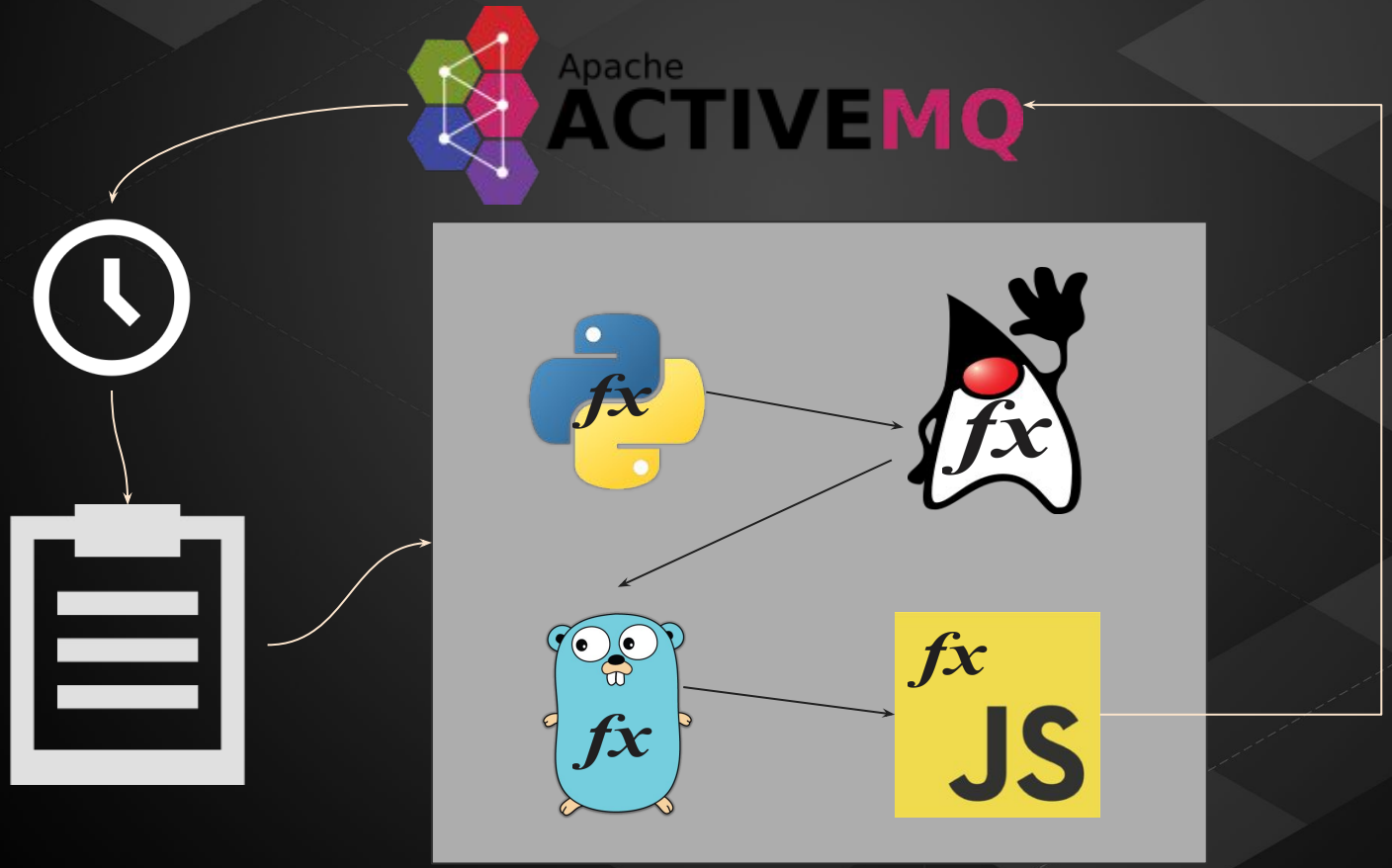
wsk -i action create redhat-developers-demo/splitUpperAndSort --sequence redhat-developers-demo/splitter,redhat-developers-demo/uppercase,redhat-developers-demo/sorter

wsk -i action invoke redhat-developers-demo/splitUpperAndSort --param text "zebra,cat,antelope" --result

Terms

- **Actions**
  - A **stateless** code snippet that gets executed (aka Functions)
- **Package**
  - A **bundle** of actions and feeds synonymous to Java Packages
- **Feeds**
  - Acts as **Interface** between external **Event Sources** and OpenWhisk **Actions**
- **Triggers**
  - Stimulus of external Events via Feeds
- **Rules**
  - Ties one or more Triggers to an Action or vice-versa
- **Activation**
  - Records the output/response of Running/Ran actions

Demo - Feeds and Events



Cloud Events

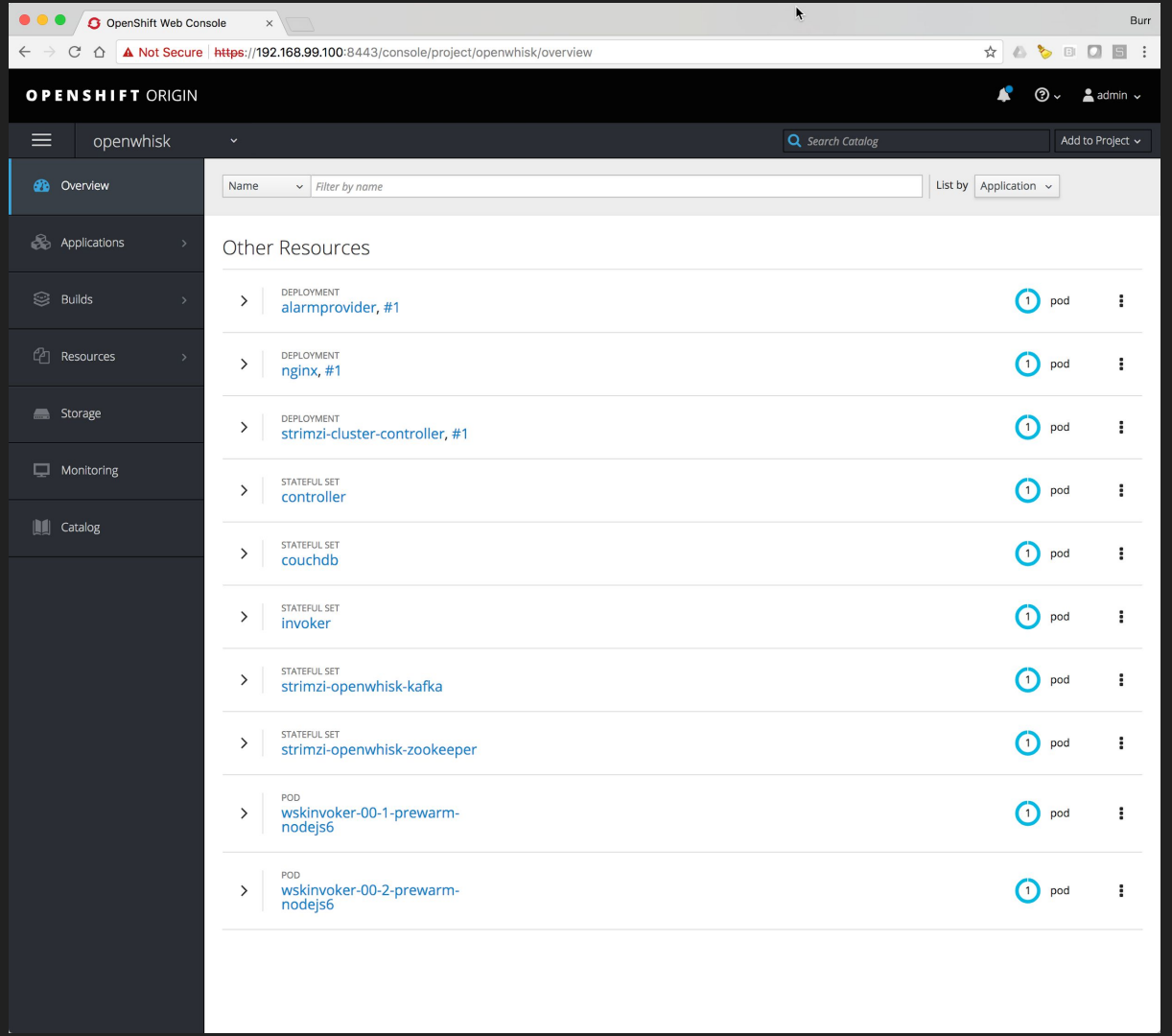
- Define specification for Cloud Events
  - Effort via CNCF's Serverless Working Group
- Major Cloud Providers, SaaS providers are part of working committee
- https://cloudevents.io/

OpenWhisk

On

OpenShift

(bit.ly/faas-tutorial)



@rafabene

RED HAT®
DEVELOPER
PROGRAM

Serverless and Servicefull

Where Microservices compliments Serverless

 @rafabene

 benevides@redhat.com

Link → http://bit.ly/serverlessfull