Focus Your app on Business Code, let the Mesh take care of the rest

Roel Hodzelmans, Red Hat
Service Mesh
Agenda for today

- Setting the scene
- 99 problems...
- Avenues we tried
- The better solution
Setting the stage
Monolith vs Microservices

App Server

HTML
Javascript
Web
Inventory
Catalog
Cart
Promo
Ratings
Orders
Data Access

Web
Node.js

Inventory
Catalog
Cart

Quarkus
Spring Boot
Vert.x

Orders
Service
Runtime

Kogito
Service
Runtime
Actually, Microservices are more like this...

https://www.youtube.com/watch?v=CZ3wluvmHeM
Adopting microservices means accepting architectural complexity in order to gain agility
Distributed Computing is old, but still hard

The fallacies of distributed computing

- The network is reliable.
- Latency is zero.
- Bandwidth is infinite.
- The network is secure.
- Topology doesn't change.
- There is one administrator.
- Transport cost is zero.
- The network is homogeneous.
Red Hat OpenShift Service Mesh

Fool’s gold?

The Death of Microservice Madness in 2018

En Español | Reddit Thread | Hacker News Thread

Microservices became a very popular topic over the last couple of years. ‘Microservice madness’ goes something like this:

Netflix are great at droops.

Netflix do microservices.

Therefore: If I do microservices, I am great at droops.

There are many cases where great efforts have been made to adopt microservice patterns without necessarily understanding how the costs and benefits will apply to the specifics of the problem at hand.

I’m going to describe in detail what microservices are, why the pattern is so appealing, and also some of the key challenges that they present.

I’ll finish with a set of simple questions might be valuable to ask yourself when you are considering whether microservices are the right pattern for you. The questions are at the end of the article.

Good Rebels

To go or not to go micro: the pros and cons of microservices

Generally Available
Wait, I have 99 Problems...?
Partial outages & Cascading failures

- Functionality can require a chain of services
- Any of these services can fail individually
  - permanently or temporarily
- Calling code should handle failures
  - Failures aren’t instantaneous
- Failures can cascade through services
Traffic spikes

- Traffic spikes are unpredictable increased load
- User demand or malicious attacks
- These spikes can be hard on specific services
- API gateways regulate at ingress only
Complex deployments & failures

- Deploying changes are often a failure point
- Untracked deployments is a problem
- “Our process docs” shouldn’t be the solution
- Service versioning adds complexity
- $SvcX:v1$ isn’t compatible with $SvcY:v2$
- Problems can show up only after release
And the challenges keep on coming....

1. Overloaded services
2. Anemic domain models
3. Usage limiting for limited resources
4. Versioning and version specific traffic
5. Lack of service discovery
6. Integration testing
7. Malicious requests
8. Dealing with stateful services
9. Finding data from 100s of logs files
10. Services coupling leading to a Death Star
11. Inability to measure/monitor performance
12. New client libs force rebuild/test of services
13. Network Latency
14. Migration from a monolith
15. Identifying unhealthy services
16. Heterogeneous communication protocols
17. Managing retry logic
18. Testing failure conditions
84. Sneaky bugs that only rear their head in prod
85. Microservice libs unavailable for my language
86. Increased complexity for operations team
87. Code pipelines for 100s of services
88. Finding root cause of a failure
89. Controlling access to service APIs
90. Increased complexity for developers
91. Deployment failures
92. Distributed transactions
93. Insecure communication of data in-transit
94. Legacy data sources
95. Service level security auditing
96. Multi-region deployments
97. Tracing service call chains
98. Lack of role based access control
99. Data consistency
So how do we fix this?
Let’s embed the logic in our service!
Red Hat OpenShift Service Mesh

Really?

Microservice

<table>
<thead>
<tr>
<th>Business Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Discovery</td>
</tr>
<tr>
<td>Circuit Breaker</td>
</tr>
<tr>
<td>Monitoring</td>
</tr>
</tbody>
</table>

Netflix OSS

<table>
<thead>
<tr>
<th>Security Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config Server</td>
</tr>
<tr>
<td>Service Registry</td>
</tr>
<tr>
<td>Monitoring</td>
</tr>
<tr>
<td>API Management</td>
</tr>
</tbody>
</table>

Platform

My Polyglot Service

- API
- Discovery
- Invocation
- Elasticity
- Resilience
- Pipeline
- Authentication
- Logging
- Monitoring
- Tracing
- Resilience
- Pipeline
- Authentication
- Logging
- Monitoring
- Tracing
Red Hat OpenShift Service Mesh

Generally Available
Ok, then what?
USE SIDECARS
Sidecars?

HTTP/1.1, HTTP/2, gRPC or TCP
With or without mTLS

Out: Telemetry
In: Policy check
And my services?

Data Plane

App Pod
App Pod
App Pod
App Pod
And where do I leave my policies, discovery et al?
Together they combine as a Service Mesh

Data Plane
- Envoy
- App
- Pod

Control Plane
- Pilot
- Mixer
- Citadel

Pluggable Tools
- Jaeger
- Kiali

It’s a sidecar
Why RHOSM?

smart is open.

ibm.com/redhat

open is smart.

ibm.com/redhat

IBM

Red Hat

WICKED
Key Features

- A dedicated network for service to service communications
- Observability and distributed tracing
- Policy-driven security
- Routing rules & chaos engineering
- Powerful visualization & monitoring
- Is available via OperatorHub

Red Hat Service Mesh

Generally Available
Enhanced Visualization of Cluster Traffic
With Kiali

Visualization of what Matters most:

- Application Topology
- Traffic throughput
- Error Rates
- Service Latency
- Service Versioning

Generally Available
Protocol Specific Analysis and
Drill down
Convenient Overviews of Individual Services

Labels
- app
- reviews
- service
- reviews

Selectors
- app
- reviews

Type
- ClusterIP

IP
- 172.30.17.223

Created at
- 9/12/2019, 12:06:32 PM

Resource Version
- 5094640

Ports
- TCP http (9080)

Endpoints
- 10.128.2.27:reviews-v1-989d5ffdf-f892gm
- 10.128.2.28:reviews-v3-755c4f7849-rs5sw
- 10.131.0.35:reviews-v2-62f8648c69-tqhnn

Health
- Healthy
- Error Rate over last 1m: 0.00%

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Labels</th>
<th>Created at</th>
<th>Resource version</th>
</tr>
</thead>
<tbody>
<tr>
<td>reviews-v1</td>
<td>Deployment</td>
<td>app reviews version v1</td>
<td>9/12/2019, 12:06:32 PM</td>
<td>5095051</td>
</tr>
<tr>
<td>reviews-v2</td>
<td>Deployment</td>
<td>app reviews version v2</td>
<td>9/12/2019, 12:06:32 PM</td>
<td>5094998</td>
</tr>
<tr>
<td>reviews-v3</td>
<td>Deployment</td>
<td>app reviews version v3</td>
<td>9/12/2019, 12:06:32 PM</td>
<td>5095043</td>
</tr>
</tbody>
</table>

Generally Available
Guided Configuration of Traffic Policies

<table>
<thead>
<tr>
<th>WORKLOAD</th>
<th>TRAFFIC WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>reviews-v1</td>
<td>- 5 + %</td>
</tr>
<tr>
<td>reviews-v2</td>
<td>- 80 + %</td>
</tr>
<tr>
<td>reviews-v3</td>
<td>- 15 + %</td>
</tr>
</tbody>
</table>

- **Evenly distribute traffic**

- **VirtualService Hosts**: reviews
- **TLS**: DISABLE
- **Add LoadBalancer**: OFF
- **Add Gateway**: OFF

Generally Available
Management of URI Matching for Virtual Services

VirtualService: bookinfo

- **Created at:** 9/12/2019, 12:06:46 PM
- **Resource Version:** 5094784

**Hosts**

* 

**Gateways**

bookinfo-gateway

HTTP Route

<table>
<thead>
<tr>
<th>Match</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>productpage</td>
</tr>
<tr>
<td>URL</td>
<td>login</td>
</tr>
<tr>
<td>URL</td>
<td>logout</td>
</tr>
<tr>
<td>URL</td>
<td>/products</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status</th>
<th>Host</th>
<th>Subset</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>productpage</td>
<td>-</td>
<td>9080</td>
</tr>
</tbody>
</table>
Service Mesh Roadmap

**Q4 CY19**
- Beginning of multi channel releases:
  - Tech Preview
  - Stable
  - LTS (To come later)
- Establish baseline for performance improvements
- Expanding automated testing matrix
- Upgrade stabilization
- Migrate Kiali to Patternfly 4

**1H CY20**
- Release of OSM v1.1 (approx Q1 2020)
  - Update to Istio 1.3 (Possibly 1.4)
  - FIPS-140-2 Certification for core components
  - Telepresence.io Support
  - Initial eBPF support
  - Additional configuration via Kiali UI
  - Multi-cluster mesh support
  - Mesh Expansion (Federation of resources into the mesh from outside OpenShift)

**2H CY20**
- Initial integration with cloud.redhat.com/openshift for cluster management
- Additional protocol support
  - MySQL
  - MongoDB
  - Kafka
HOLD UP, WAIT A MINUTE.
What about....?
Questions?

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat
Please don’t forget to rate the keynote sessions in the **Red Hat Events App**:

1. Open the Agenda
2. Select the session you have followed
3. Rate the session