

OpenShift Developer

Distance Learning

Optional Learnings and homework assignment



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[facebook.com/redhatinc](https://www.facebook.com/redhatinc)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



twitter.com/RedHat

Self introduction

Name: Wanja Pernath

Email: wpernath@redhat.com

Base: Germany (very close to the Alps)

Role: EMEA Technical Partner Development

Manager - OpenShift and MW

Experience: Years of Consulting, Training,

PreSales at Red Hat and before



Agenda

Agenda

- What you've done so far
- Optional Learnings - Introduction
 - Serverless / knative
 - ServiceMesh / Istio
- Homework
 - Description
 - Where to upload?
- Summary & Thank you

What you've done so far...

Week #1

- You've started to become familiar with development on OpenShift
 - Basic understanding
 - Using the CLI and the UI
 - First apps created
- More advanced basic development

Week #2

- You've learned what and how to package your app to do
 - Release management
 - CI/CD
 - You've learned to differentiate between the need of tools for both
- You might come to the conclusion that you will use
 - Kustomize for CI/CD and releases
 - Helm Charts for simple app deployment
 - Operators for more complex apps which require special handling of Day 2 operations

Week #3

- You've learned how to do CI/CD with OpenShift
- You've learned and have an understanding of GitOps and how to use it
- You've a good understanding of Tekton / OpenShift Pipelines
- You understand that you still need BOTH tools for your projects
 - ArgoCD (or similar things) for CD
 - And something like Tekton to do CI

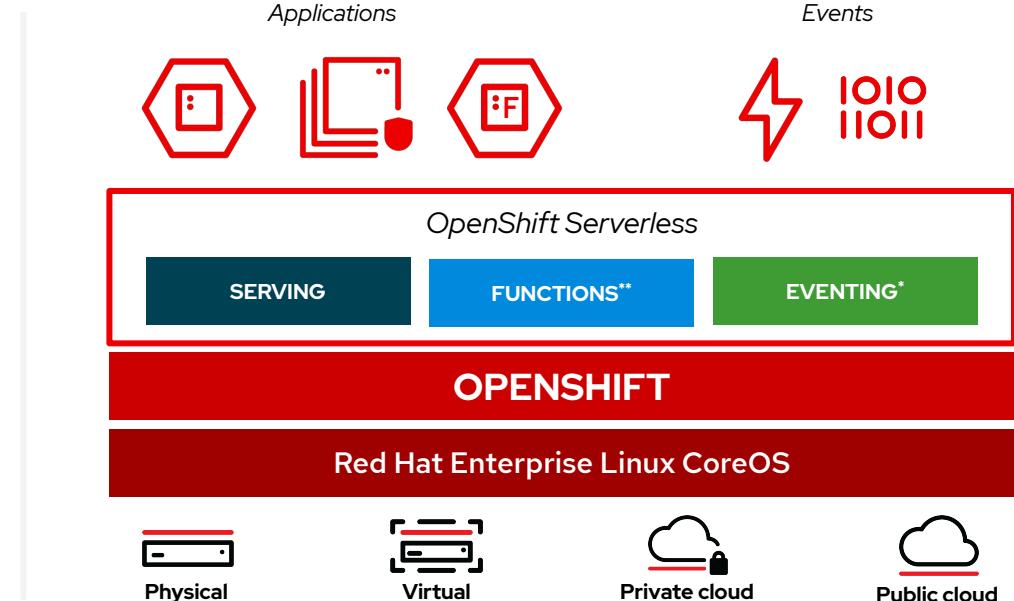
Optional - Serverless / knative



OpenShift Serverless

Event-driven serverless containers and functions

- Deploy and run **serverless containers**
- Use any programming language or runtime
- Modernize existing applications to run serverless
- Powered by a rich ecosystem of event sources
- Manage serverless apps natively in Kubernetes
- Based on open source project **Knative**
- Run anywhere OpenShift runs



* Eventing is currently in Technology Preview

** Functions are currently a work in progress initiative

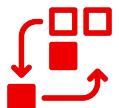
OpenShift Serverless

Key Features



Containers made easy

Simplified developer experience to deploy applications/code on serverless containers abstracting infrastructure & focusing on what matters.



Immutable revisions

Deploy new features: performing canary, A/B or blue-green testing with gradual traffic rollout with no sweat and following best practices.



Automatic scaling

No need to configure number of replicas, or idling. Scale to zero when not in use, auto scale to thousands during peak, with built-in reliability and fault-tolerance.



Ready for the Hybrid

Portable serverless running anywhere OpenShift runs, that is on-premises or on any public cloud. Leverage data locality and SaaS when needed.



Any programming language

Use any programming language or runtime of choice. From Java, Python, Go and JavaScript to Quarkus, SpringBoot or Node.js.



Event Driven Architectures

Build loosely coupled & distributed apps connecting with a variety of built-in or third-party event sources or connectors powered by Operators.

Installation experience

"Easy day 1 and even better for day 2"

- Click Install experience
- Developer & admin experience in Console
- Built-in event sources
- No external dependencies.

 OpenShift Serverless Operator

1.7.0 provided by Red Hat, Inc.

[Install](#)

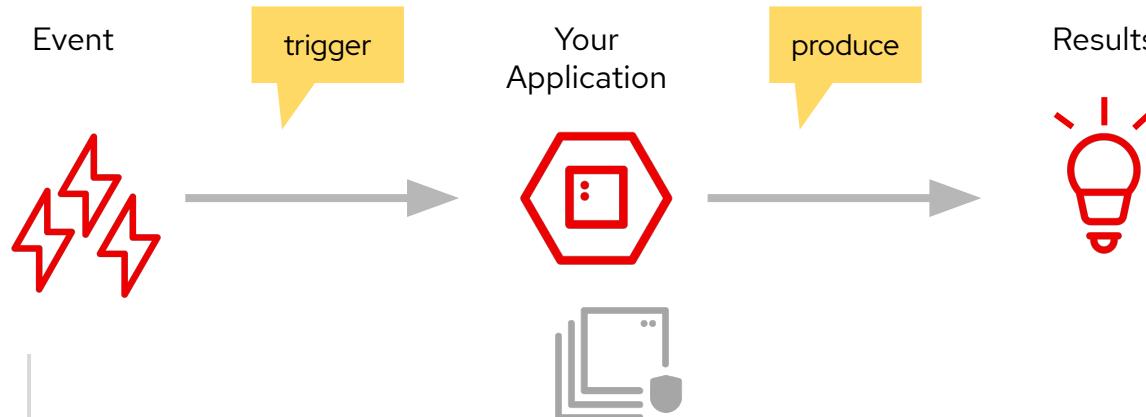
OPERATOR VERSION
1.7.0

PROVIDER TYPE
Red Hat

PROVIDER
Red Hat, Inc.

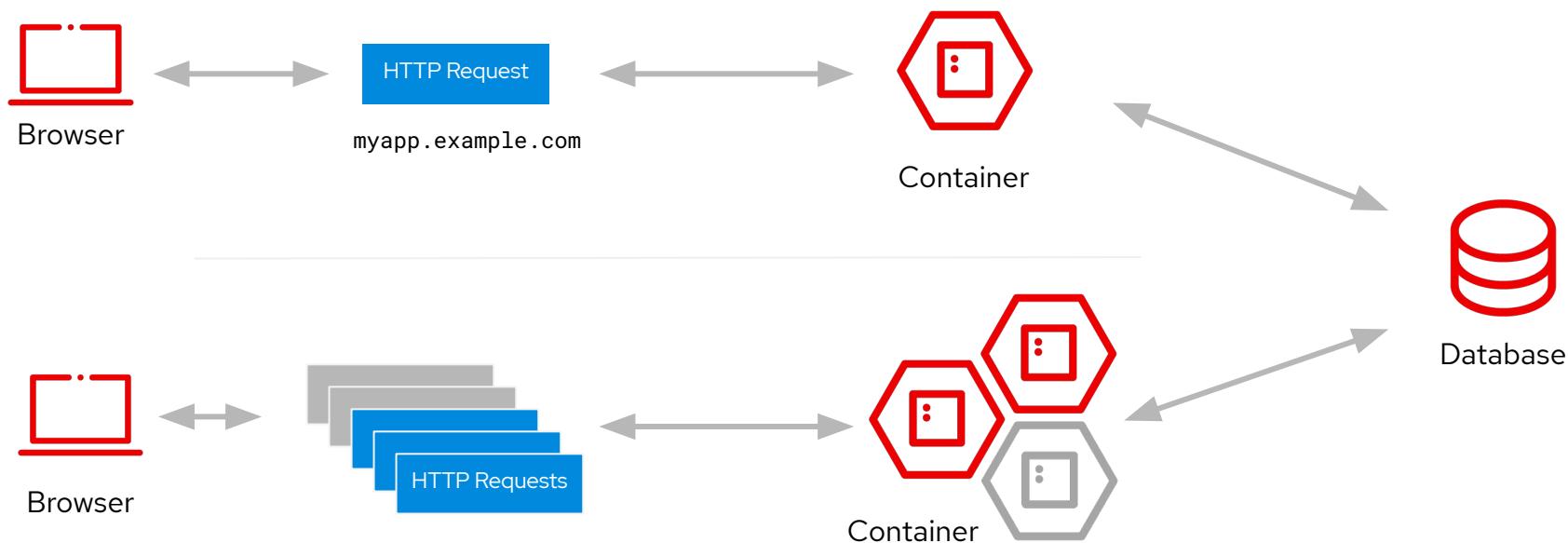
The Red Hat OpenShift Serverless operator provides a collection of APIs that enables containers, microservices and functions to run "serverless". Serverless applications can scale up and down (to zero) on demand and be triggered by a number of event sources. OpenShift Serverless integrates with a number of platform services, such as Metering and Monitoring and it is based on the open source project Knative.

The "Serverless Pattern"



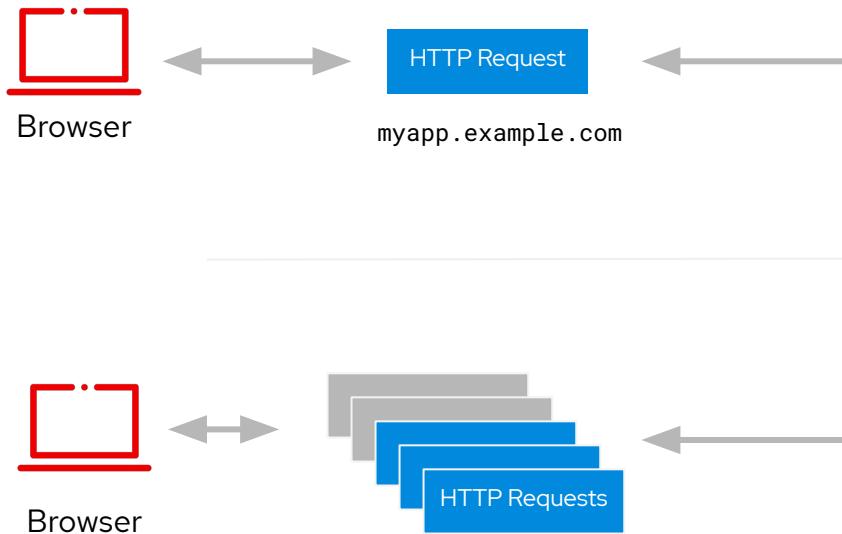
The "Serverless Pattern"

A serverless web application



The "Serverless Pattern"

A serverless web application

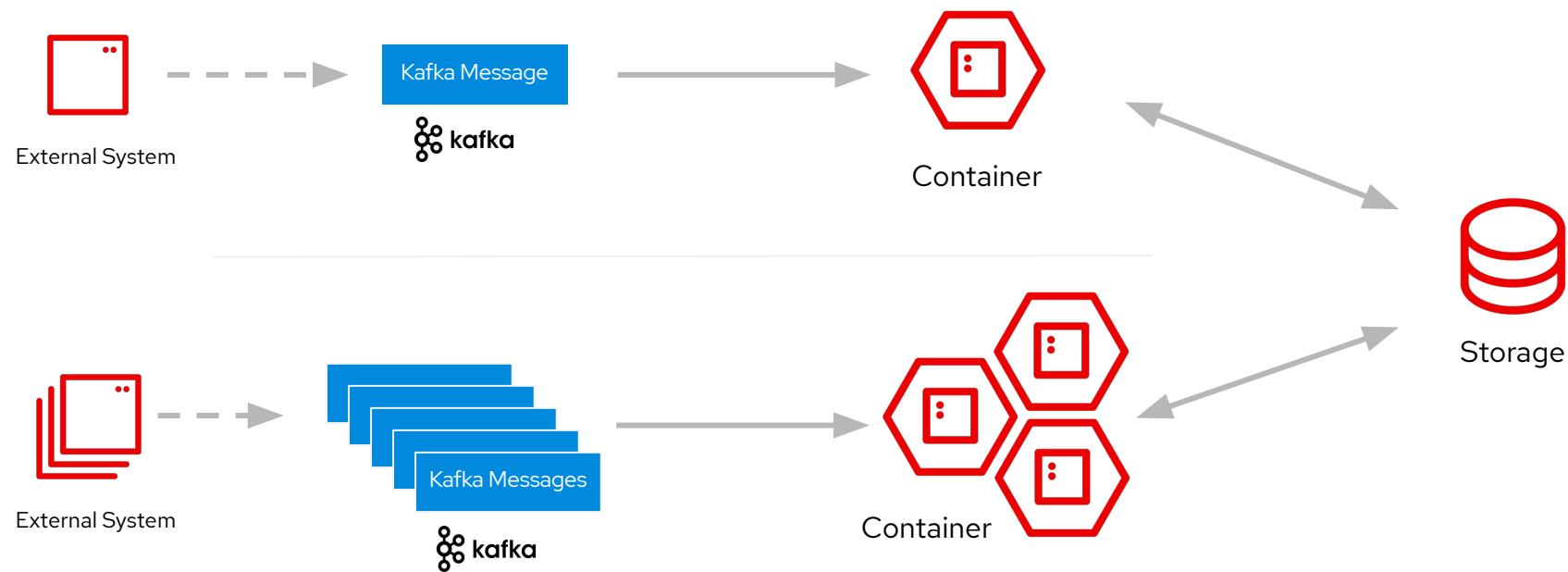


Benefits of this model:

- No need to setup auto-scaling and load balancers
 - Scale down and save resources when needed.
 - Scale up to meet the demand.
- No tickets to configure SSL for applications
- Enable Event Driven Architectures (EDA) patterns
- Enable teams to associate cost with IT
- Modernize existing applications to run as serverless containers

The "Serverless Pattern"

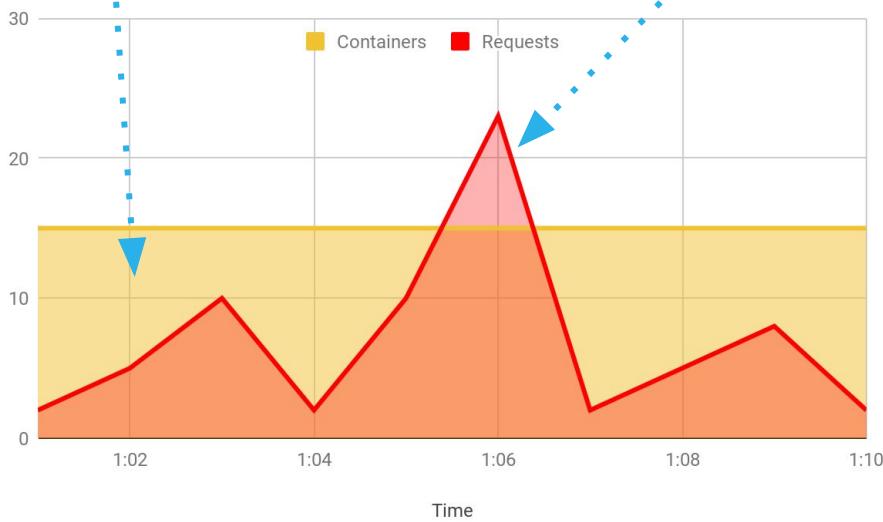
Processing a Kafka message



Serverless Operational Benefits

Over provisioning

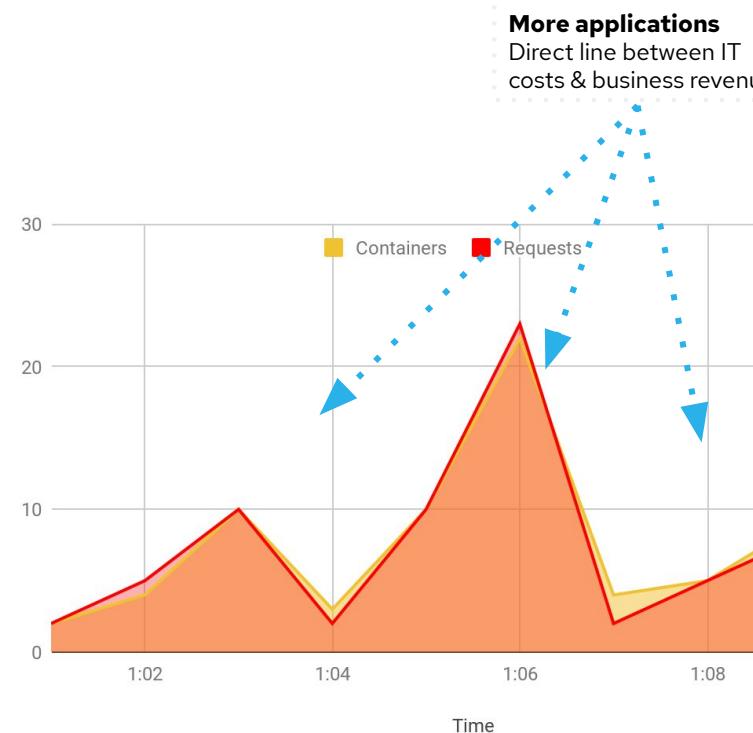
Time in capacity planning
IT cost of idle resources



NOT Serverless

Under provisioning

Lost business revenue
Poor quality of service



with Serverless

Choosing the Right Tool

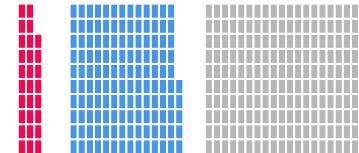
Your Team



The Ecosystem



Performance





Developer experience

</> Java
</> Node
</> Python
</> Go
</> Ruby
</> Ruby on Rails
</> PHP
</> Perl
</> .NET Core

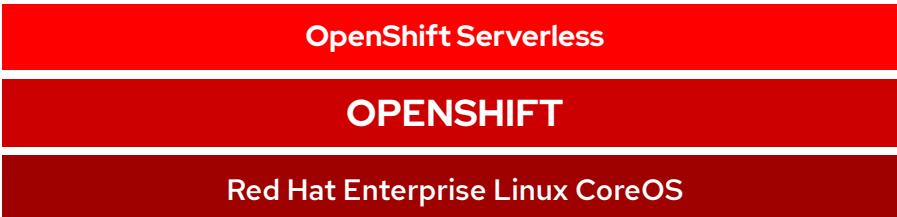
```
$ kn service create --image=
```



CLI



UI



Physical



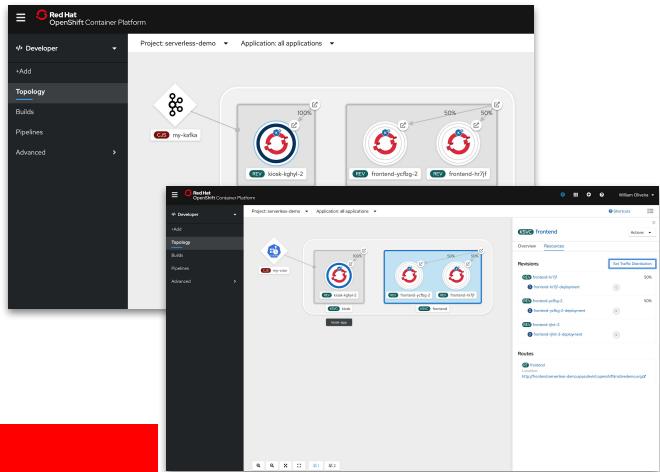
Virtual



Private cloud



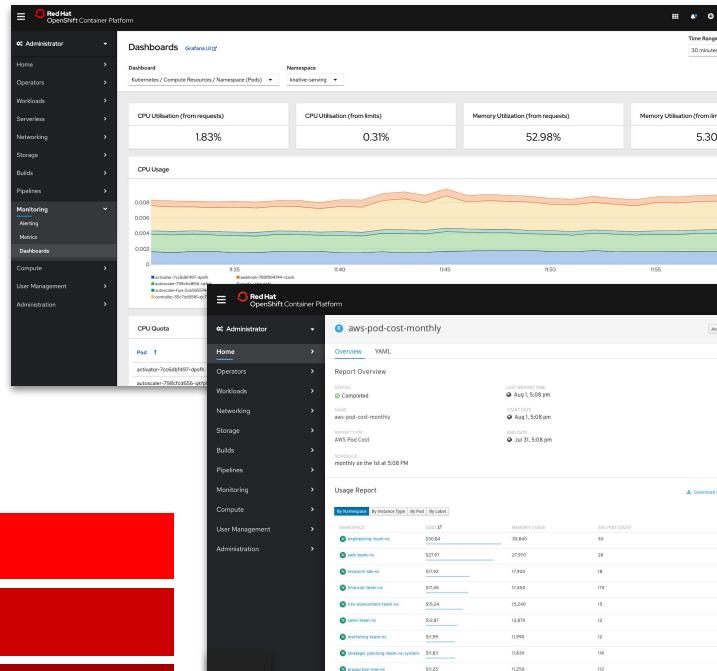
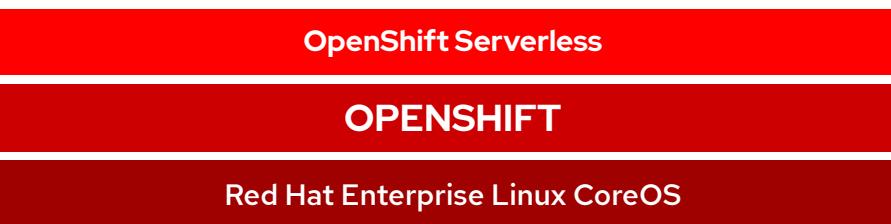
Public cloud





Admin experience

- Monitoring, Metering and Logging
- Disconnected install support (air-gapped)
- Egress proxy with TLS support
- Over the air updates and patches



Physical



Virtual



Private cloud



Public cloud



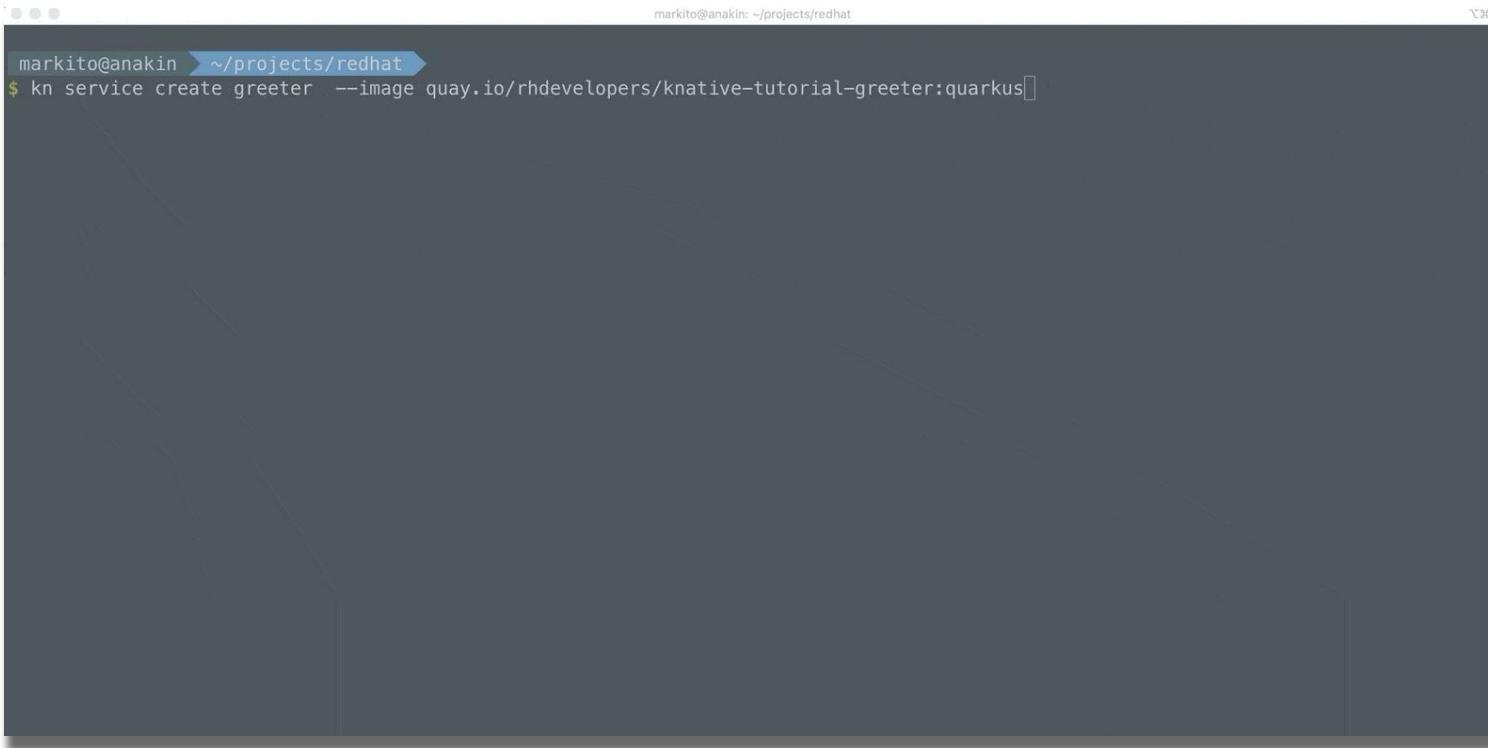
Command Line Experience

```
$ kn service create myService --image=[registry/mycontainer:v1] --min-scale=1 --max-scale=100
```

```
$ kn service update myService --traffic myService-rev1=50,myService-rev2=50
```

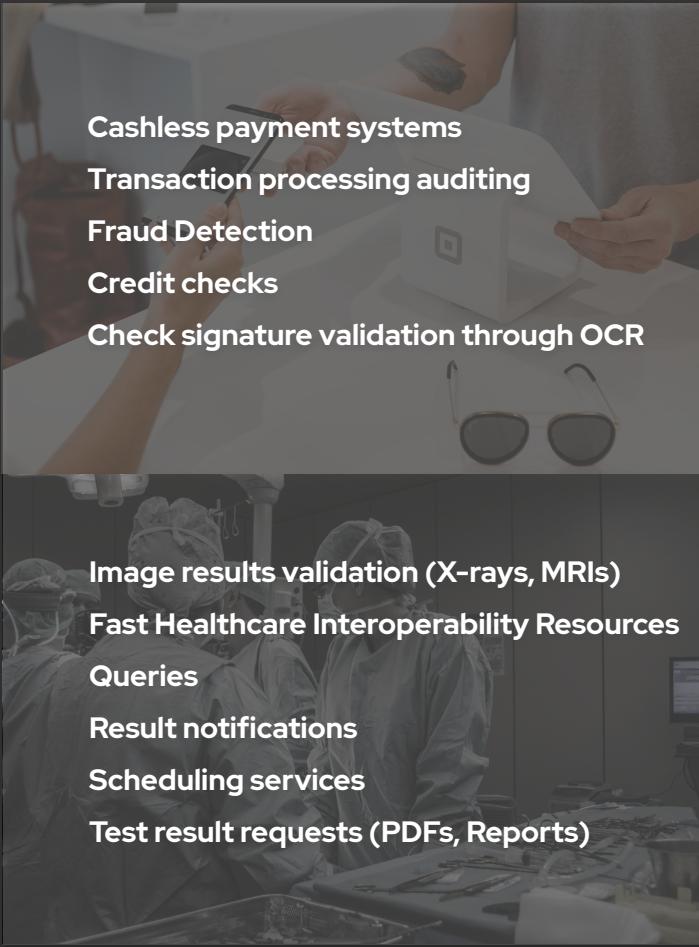
```
$ kn source cronjob create my-cron --schedule "* * * * */1" --data "ping" --sink svc:myService
```

Hello World with Quarkus!



A dark terminal window with a light blue header bar. The header bar shows the user's name 'markito@anakin' and the path '~/.projects/redhat'. The main area of the terminal is dark gray and shows a command being typed:

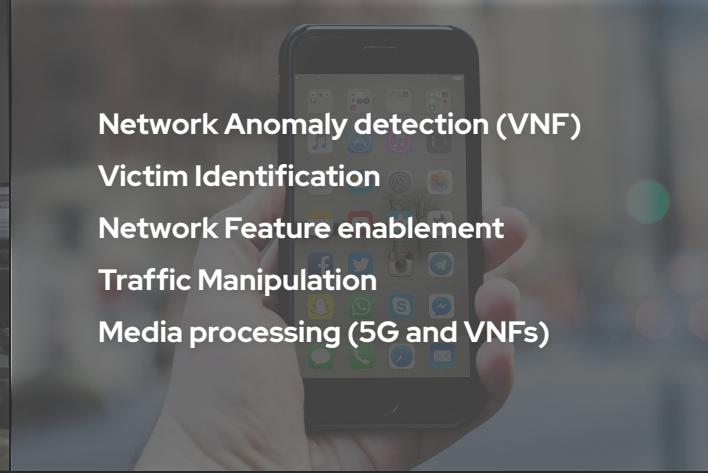
```
markito@anakin ~/.projects/redhat
$ kn service create greeter --image quay.io/rhdevelopers/knative-tutorial-greeter:quarkus
```



Cashless payment systems
Transaction processing auditing
Fraud Detection
Credit checks
Check signature validation through OCR



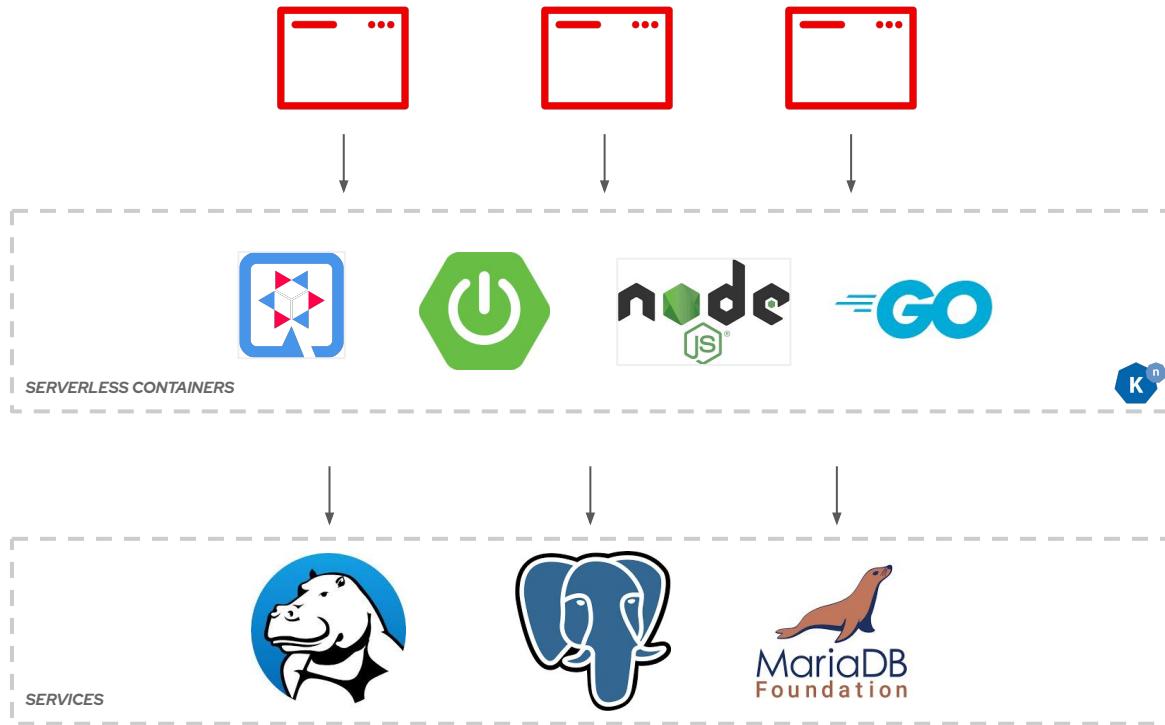
Product thumbnail generation
Chatbots and CRM functions
Marketing Campaign notifications
Sales Audit
Content Push



Network Anomaly detection (VNF)
Victim Identification
Network Feature enablement
Traffic Manipulation
Media processing (5G and VNFs)



Web Applications and APIs



Language or runtime of your choice:

OpenJDK



python™



django



VERT.X



JS



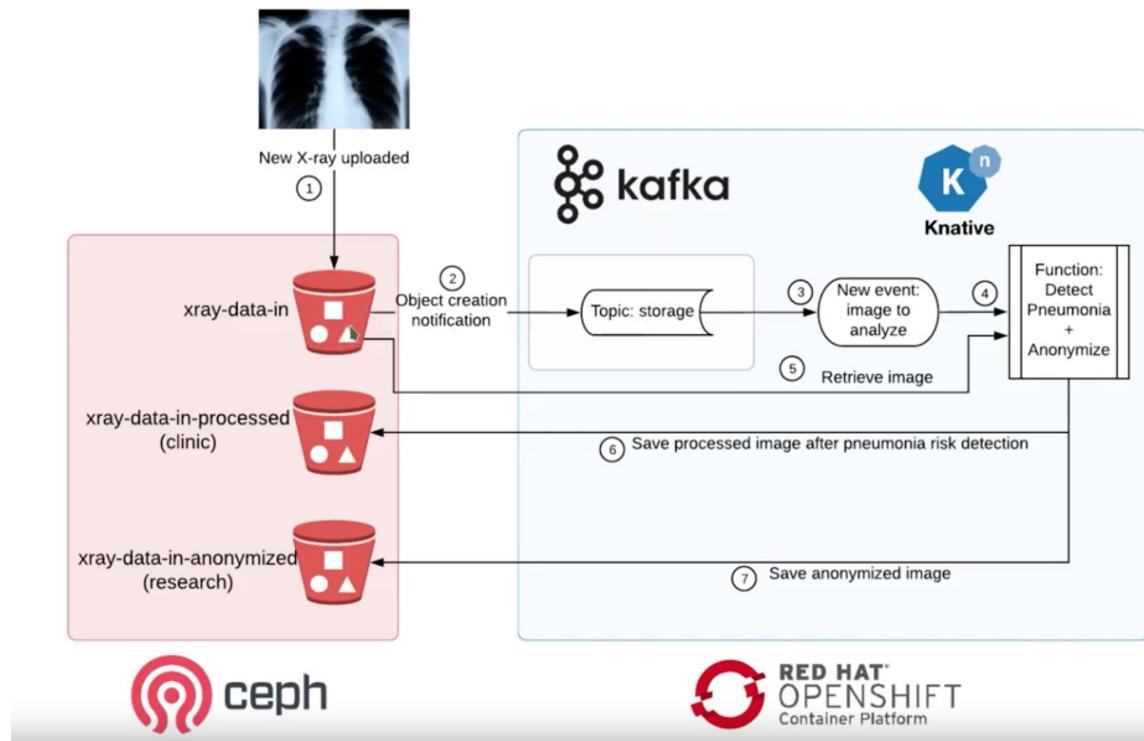
RAILS



Data Transformation

Common Use cases

- Image analysis* (medical, AI/ML, media)
- Video format transcoding
- File type conversion (financial, medical)
- Data extraction
- Lightweight Data Transformation
- Invoice Generation



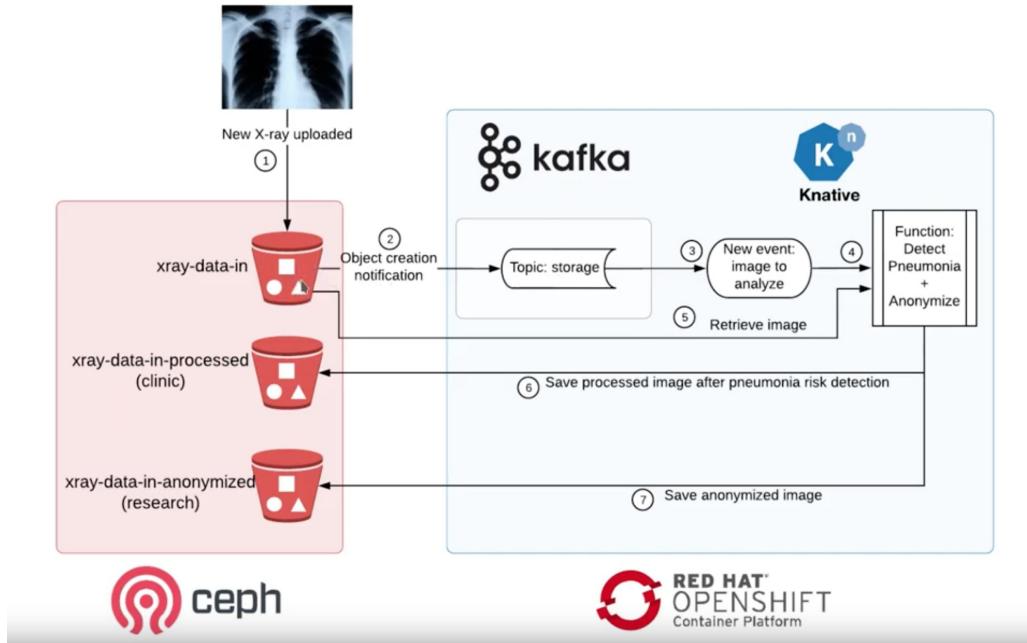


Ceph + Kafka + Serverless

"Automated AI/ML Data Pipelines"

Common Use cases

- Image analysis*
(medical, AI/ML, media)
- Video format transcoding
- File type conversion
(financial, medical)
- Data extraction
- Data Transformation
- Invoice Generation





Building on OpenShift Serverless with Red Hat Services

Connected Services

How Knative services interact with the outside world.



Service Orchestrator

Composing multiple services together into an application.



Event Streaming

All modern architectures need some Kafka.



API Gateway

Next gen APIs still require management.



Implementing Services

Functions, languages, and the vagaries of cold starts.



The Dirty Word in Serverless

Yep, you still need state to handle long-lived orchestration.

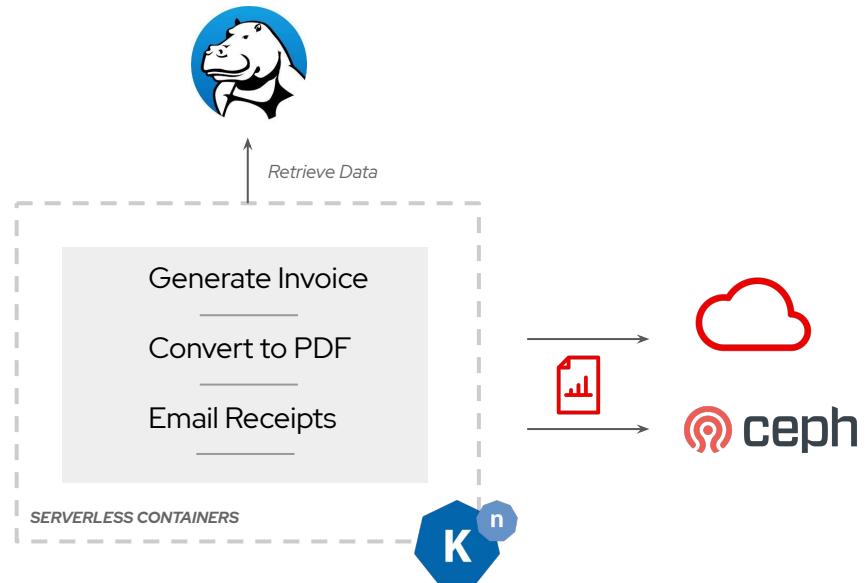




Scheduled Apps Cron Jobs



- Every Hour →
- Every Day →
- Every Week →
- Every Month →



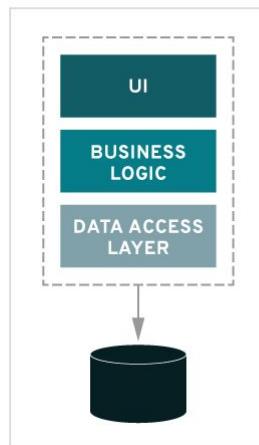
Optional - ServiceMesh / Istio

What are Microservices?

an architectural style that structures an application as a collection of services

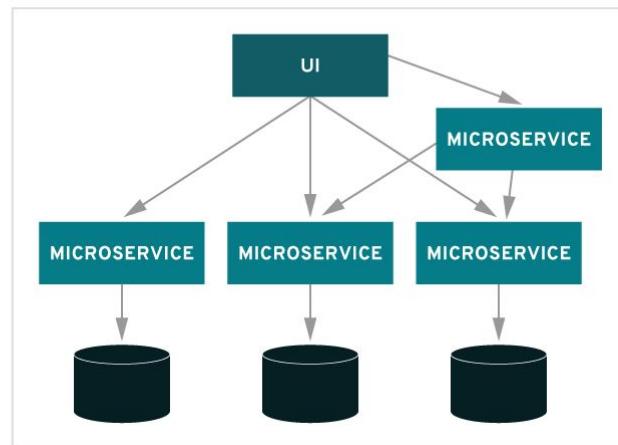
- Single purpose
- Independently deployable
- Have their context bound to a biz domain
- Owned by a small team
- Often stateless

MONOLITHIC



VS.

MICROSERVICES



Benefits of Microservices



Agility

Deliver updates faster and react faster to new business demands

Highly scalable

Scale independently to meet temporary traffic increases, complete batch processing, or other business needs

Can be purpose-built

Use the languages and frameworks best suited for the

Many orgs have had success with Microservices - Netflix, Amazon, eBay, The Guardian

Resilience

Improved fault isolation restricts service issues, such as memory leaks or open database connections, to only



There is inherent complexity in adopting microservices

Some common areas where organizations stumble when adopting microservices

Tolerance to Faults

Cascading failure, partial outages, traffic spikes

Services Communication Needs

Latency, concurrence, distributed transactions

Securing Services

Malicious requests, DoS, id & access control

DevOps and Deployments

More failure surface, version incompatibility, untracked svcs

Inability to Monitor & Understand Performance

More to monitor & different types of monitoring required

Highly Distributed Logs

Scattered logs, lots more logs to manage, access control

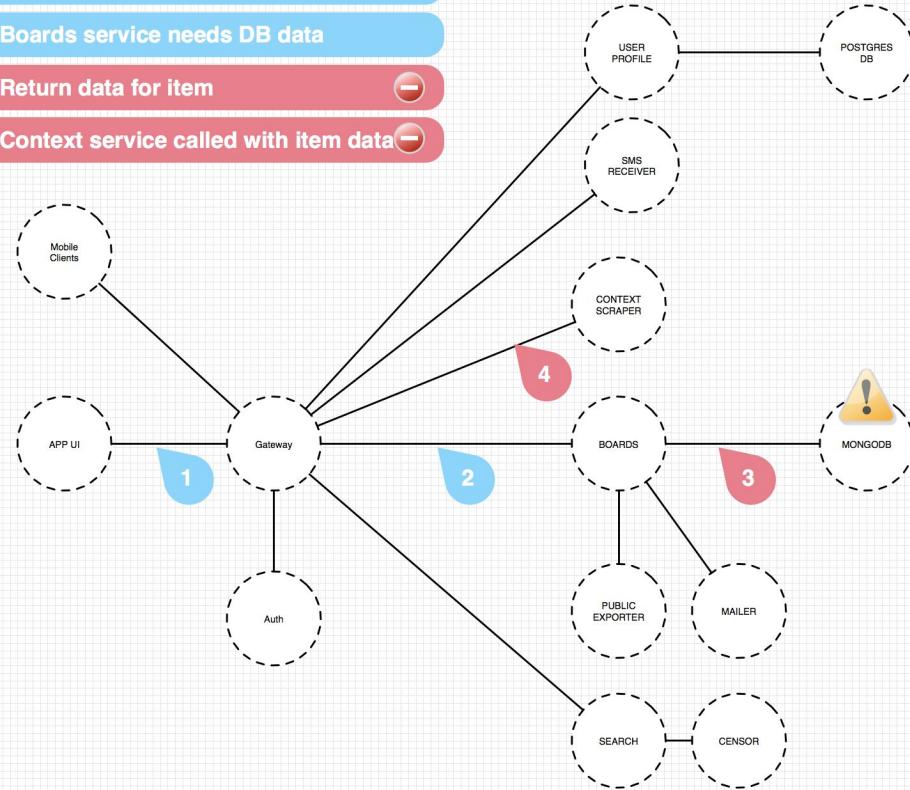
Biggest challenge:
What used to be internal now
needs to go across a network

Partial Failure → Cascading Failures

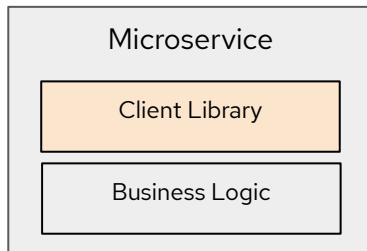
- Route service's database goes offline
- Now no item data can be returned
- Default timeout for developer's HTTP framework is 2 minutes
 - This wait is happening repeatedly
- Item data is needed to pass to context scraper service - so it fails too
- User experience is poor

- Leads to unexpected case of users repeatedly mashing the refresh button
- Now the boards service begins to get more requests than than it can handle

1. User click on board item
2. Boards service needs DB data
3. Return data for item
4. Context service called with item data



Language Specific Libraries and Tools



Narrow Scope

Built to address a specific problem such as fault tolerance

Language Specific

Client libraries are tied to programming language

Thick Clients

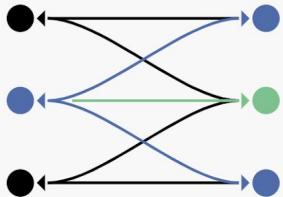
Solution results in clients with bulk of capabilities

Don't force extra work
on developers

There is a better way

Istio Service Mesh

A modern way to manage the complexity of microservice applications



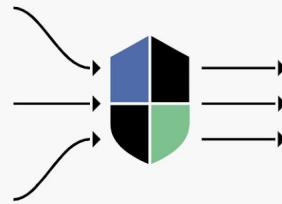
Connect

Intelligently control the flow of traffic and API calls between services, conduct a range of tests, and upgrade gradually with red/black deployments.



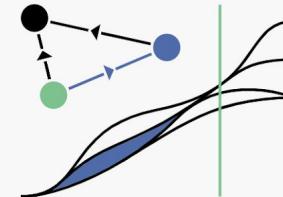
Secure

Automatically secure your services through managed authentication, authorization, and encryption of communication between services.



Control

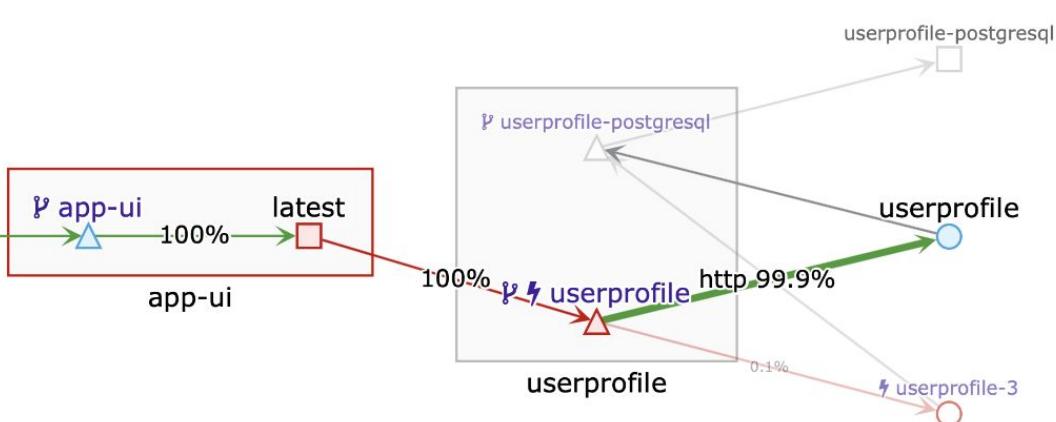
Apply policies and ensure that they're enforced, and that resources are fairly distributed among consumers.



Observe

See what's happening with rich automatic tracing, monitoring, and logging of all your services.

Handling Partial Failures with the Service Mesh

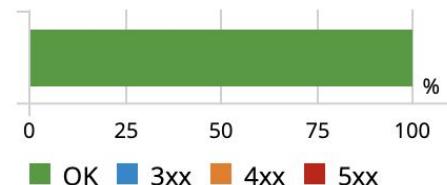


Traffic Response Codes

HTTP requests per second:

Total	%Success	%Error
-------	----------	--------

35.58	100.00	0.00
-------	--------	------



HTTP Request Traffic min / max:

RPS: 27.73 / 31.07 , %Error 0.00 / 0.00

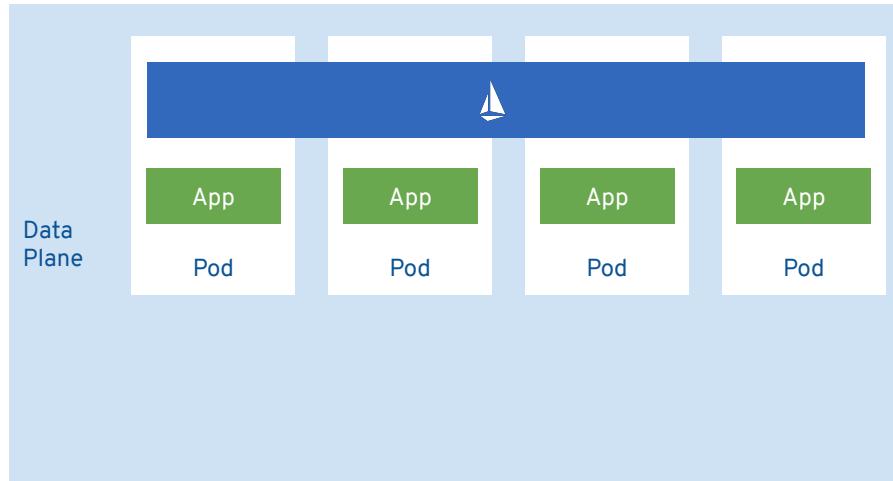


```
26  trafficPolicy:  
27    connectionPool:  
28      http:  
29        http1MaxPendingRequests: 1  
30        maxRequestsPerConnection: 1  
31    outlierDetection:  
32      consecutiveErrors: 1  
33      interval: 1s  
34      baseEjectionTime: 10m  
35      maxEjectionPercent: 100
```

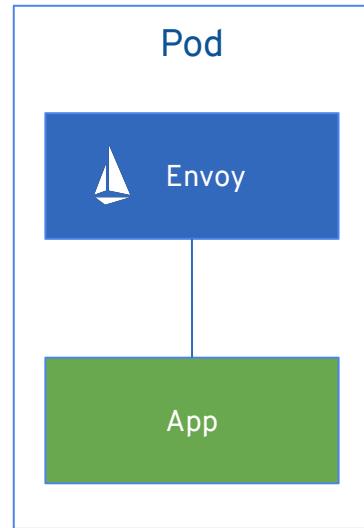
YAML configurable,
(Kubernetes native)

The service mesh is critical in addressing the inherent complexity of microservices

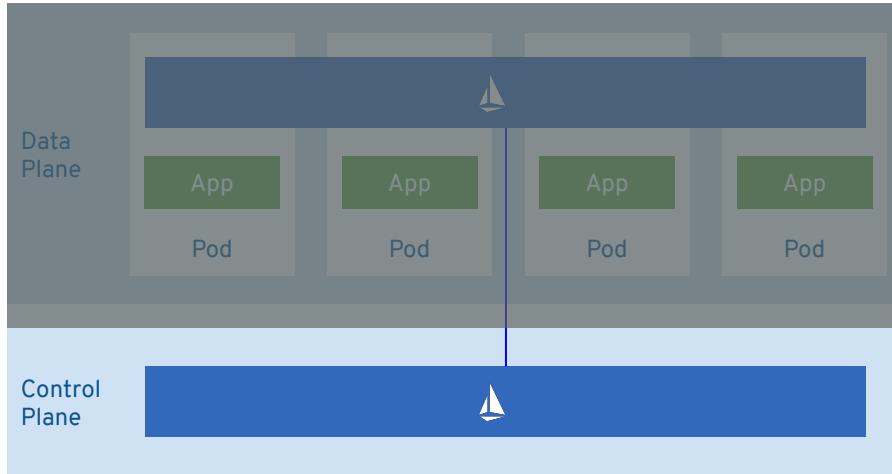
Your Services are in a “Data Plane”



What's a Sidecar have to do with containers?



Your Policy Makes Up a “Control Plane”



Istio - Caveats

- Caveats
 - Be careful, istio is NOT a ServiceBus
 - Be careful, istio is NOT an API Manager
 - Be careful, istio is NOT meant to be a router
 - Be careful, istio is NOT a BPM tool
- Istio is meant to help solving some challenges
 - Centrally encryption / management of internal service communication
 - Distributed networking / communication of services
 - Increase Fault tolerance
 - Fault simulation

... and now it's
time to prove
your knowledge!

Week #4 - Homework

- You have 2 different options to complete this course
 - **Either** create a presentation and upload it **OR**
 - Create a demo, make a screencast and upload it

Please upload your work until end of this week (Sunday, 14th of March 2021)

(if you have urgent other important things to do this week, please get in touch with me)

Homework - The presentation

- Think about a use case / a customer scenario
- Describe the use case in 1-2 slides
- Describe your solution in 1-2 slides
- Create an architecture diagram of the solution and describe it in 1-2 slides
- Describe the benefits of using your solution in 1-2 slides
- Upload it to OPEN as PDF file

Homework - The Demo

- Create your own demo script, which should contain
 - A scenario you'd like to address
 - Take one of the technologies you've learned over the course
 - Odo, Helm, Kustomize, Tekton, GitOps, Serverless, ServiceMesh...
 - Describe, why you have chosen this technology and why you think this would help in the scenario
 - Use a screen recorder tool to record your demo with your voice
 - The whole recording should not be longer than 5 minutes
- Upload it to OPEN

Homework - Upload your work

The screenshot shows the Red Hat Partner Connect interface. At the top, there's a navigation bar with links for 'Partner Home', 'Catalog', 'My Learning', and 'Reporting'. A user profile for 'Wanja Pernath' is visible on the right.

The main content area displays the 'OpenShift Developer Distance Learning Program - Sales Engineer (EMEA)' course. It includes a progress bar at 73% completion. Below the progress bar, there's a brief description of the program, stating it's a new OpenShift learning experience for Developers. It highlights that Red Hat has put together a comprehensive learning program for developers to learn everything they need to understand the concepts, architectural principles, and components of Red Hat products, for successfully discussing and developing OpenShift at a customer site from a developer perspective.

Under the description, there's a list of what participants will get upon completion, including benefits like plain Kubernetes, how to effectively and efficiently demo OpenShift, and how to start coding with OpenShift in their preferred language. It also mentions how to use OpenShift for managing releases and making use of advanced features like Istio and Serverless.

At the bottom of the main course page, there's a link to 'OpenShift Developer Distance Learning Program - Detailed Overview'.

Below the main course page, there's a section titled 'WEEK 1: Introduction to OpenShift for Developers'. This section lists five mandatory modules:

- Introduction to Red Hat OpenShift for Developers**: 2 hours, Not Enrolled. This module provides details on the agenda and structure of the Distance Learning Program.
- Red Hat Foundations**: 2 hours, Incomplete. This module covers Red Hat's history and its role in enterprise customers.
- Getting Started with Red Hat OpenShift for Developers**: 15 minutes, Completed Feb-14-21. This module teaches how to use the OpenShift Container Platform to build and deploy an application with a data backend and a web frontend.
- Using the CLI to Manage Resource Objects**: 15 minutes, Incomplete. This module shows how to enumerate, describe, and update application resource objects on OpenShift.
- Deploying Applications From Source**: 15 minutes, Completed Mar-03-21. This module demonstrates how to deploy an application from its source code using a Source-to-Image build on OpenShift.

At the bottom of the week 1 section, there are 'RED HAT' and 'GUIDED HELP' buttons.

- Open your OPEN progress
- Scroll down to Week #4

Homework - Upload your work

The screenshot shows a web-based learning platform interface for Red Hat OPEN - WPNRATH. The main content area displays a list of learning modules categorized by week:

- WEEK 3: CI/CD** (All Mandatory, 4 items):
 - Operator SDK with Helm (30 minutes, Completed Mar-05-21)
 - Operator SDK with Go (30 minutes, Completed Mar-05-21)
 - CI/CD with Red Hat OpenShift (2 hours, Not Enrolled)
 - GITOPS Integration with Red Hat OpenShift (30 minutes, Completed Mar-05-21)
- WEEK 4: Summary and Wrap Up** (All Mandatory, 2 items):
 - Summary and Wrap up and Next Steps (2 hours, Not Enrolled)
 - OpenShift Developer Distance Learning Program (EMEA) - Homework Assignment (In Progress)A red box highlights the "View" button next to the homework assignment entry.
- OPTIONAL LEARNING** (All Optional, 2 items):
 - Getting Started with OpenShift Serverless (30 minutes)
 - Red Hat OpenShift ServiceMesh Interactive Learning Scenarios (9 hours)

At the bottom of the interface, there are two buttons: "RED HAT" and "GUIDED HELP".

- Click on “View” of the entry “Homework Assignment”

Homework - Upload your work

The screenshot shows the Red Hat Partner Connect interface. At the top, there's a navigation bar with links for 'Partner Home', 'Catalog', 'My Learning', and 'Reporting'. Below the navigation, the main content area has a title 'OpenShift Developer Distance Learning Program (EMEA) - Homework Assignment'. There are two sections: 'Submission Files' and 'Submission Comments'. Under 'Submission Files', there is a button labeled 'Upload' with a red box and arrow highlighting it. Below 'Submission Files' is a section for 'Learning Path' containing items like 'OpenShift Developer Distance Learning Program - Sales Engineer (EMEA)' and 'My Personal Learning Path'. At the bottom of the page, there are links for 'Copyright © 2021 Red Hat Inc.', 'Privacy Statement', 'Terms of Use', and 'All policies and guidelines'. There are also 'RED HAT' and 'GUIDED HELP' buttons.

- Click on Upload
- Choose your File (presentation or video)

Homework - Upload your work

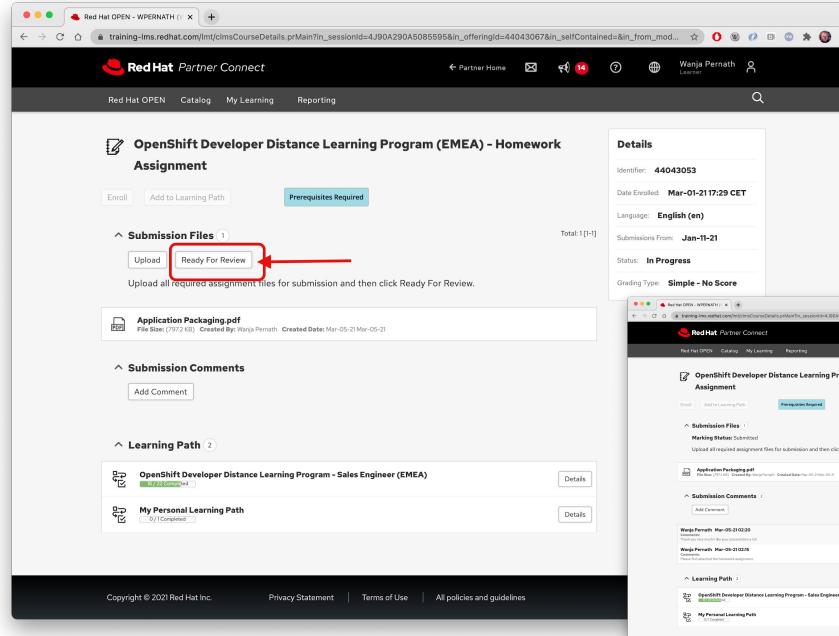
The screenshot shows a web browser window for 'Red Hat OPEN - WPERNATH'. The URL is https://training-lms.redhat.com/lmt/clmsCourseDetails.prMain?in_sessionId=4J90A290A5085595&in_offeringId=44043067&in_selfContained=&in_from_mod.... The page title is 'Red Hat Partner Connect'.

The main content area displays a 'Homework Assignment' for the 'OpenShift Developer Distance Learning Program (EMEA)'. It includes sections for 'Submission Files' (with a PDF file attached) and 'Submission Comments' (with a message from 'Wanja Pernath'). A red box highlights these two sections, with red arrows labeled 1 and 2 pointing to them respectively.

At the bottom of the page, there are links for 'Copyright © 2021 Red Hat Inc.', 'Privacy Statement', 'Terms of Use', 'All policies and guidelines', 'RED HAT', and 'GUIDED HELP'.

- Upload your file
- You should see it here
- Optionally, add comments to explain what your homework is about

Homework - Upload your work



The screenshot shows the Red Hat Partner Connect interface with the following details:

- Details:**
 - Identifier: 44043053
 - Date Enrolled: Mar-01-21 17:29 CET
 - Language: English (en)
 - Submissions From: Jan-11-21
 - Status: In Progress
 - Grading Type: Simple - No Score
- Submission Files:** 1 item
 - Upload (button)
 - Ready For Review (button, highlighted with a red box)

Upload all required assignment files for submission and then click Ready For Review.
- Submission Comments:**
 - Add Comment (button)
- Learning Path:**
 - OpenShift Developer Distance Learning Program - Sales Engineer (EMEA)
 - My Personal Learning Path

- Upload your file
- Click on “Ready for Review” once you’re done

Homework - Let me have a look at your work

Now I need to have a look at your work. If I need to know anything, have questions or something, I am going to comment it. → You'll be getting an email

If I am accepting it, you're also going to get an email

This should be done by end of the week (depending on how many of you are going to submit your homework)

Homework - Upload your work

The screenshot shows a web browser window titled "Red Hat OPEN - WPERNATH" displaying the "Optional Learning" section of the Red Hat OpenShift Distance Learning Program. The page lists various learning paths and webinars, categorized by week.

WEEK 1: Kubernetes Fundamentals

- Kubernetes API Fundamentals
- Operator Lifecycle Manager
- ETCD Operator
- Red Hat Ansible Operator Overview
- Operator SDK with Helm
- Operator SDK with Go

WEEK 3: CI/CD

- All Mandatory (4)
- Ci/CD with Red Hat OpenShift
- GitOps Introduction with Red Hat OpenShift
- Multi-Cluster GitOps with Red Hat OpenShift
- Getting Started with Red Hat OpenShift Pipelines

WEEK 4: Summary and Wrap Up

- All Mandatory (2)
- Summary and Wrap up and Next Steps
- OpenShift Developer Distance Learning Program (EMEA) - Homework Assignment

At the bottom of the page, there is a red box highlighting the "OpenShift Developer Distance Learning Program (EMEA) - Homework Assignment" link. The page also features the Red Hat logo and a "GUIDED HELP" button.

Summary

Summary

- You've learned a lot over the past weeks
- Goal was to help you understanding the huge ecosystem of OpenShift from a Developer / User perspective

Optional section marker or title

57



Thank you



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[facebook.com/redhatinc](https://www.facebook.com/redhatinc)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



twitter.com/RedHat