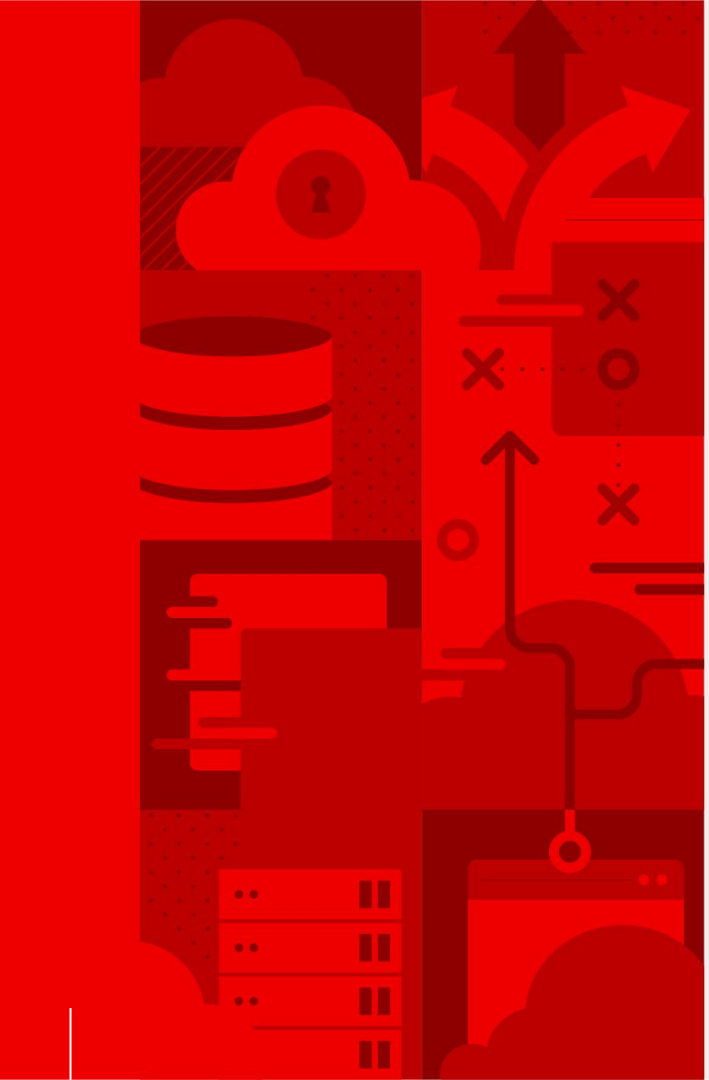


OpenShift Developer

Distance Learning

Optional Learnings and homework assignment

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat

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

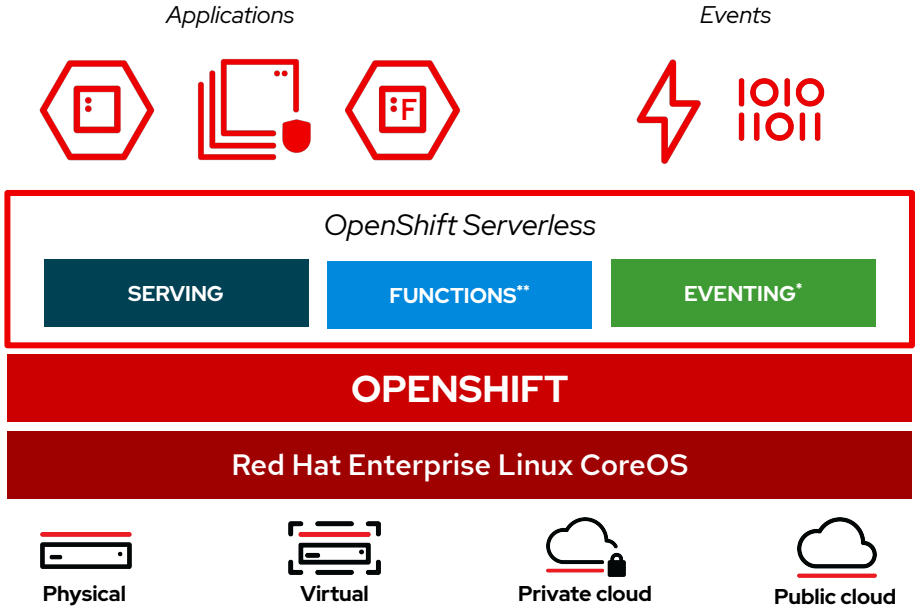
Optional - Serverless / knative

Generally Available

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

Red Hat

OpenShift Serverless

Key Features

Containers made easy

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

Ready for the Hybrid Cloud

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

Immutable revisions

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

Any programming language

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

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.

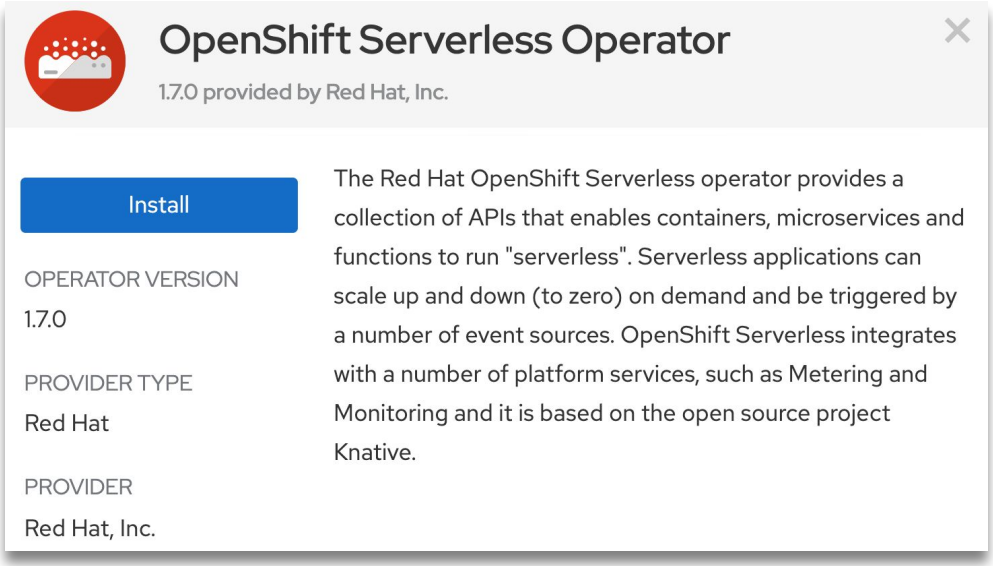
Event Driven Architectures

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

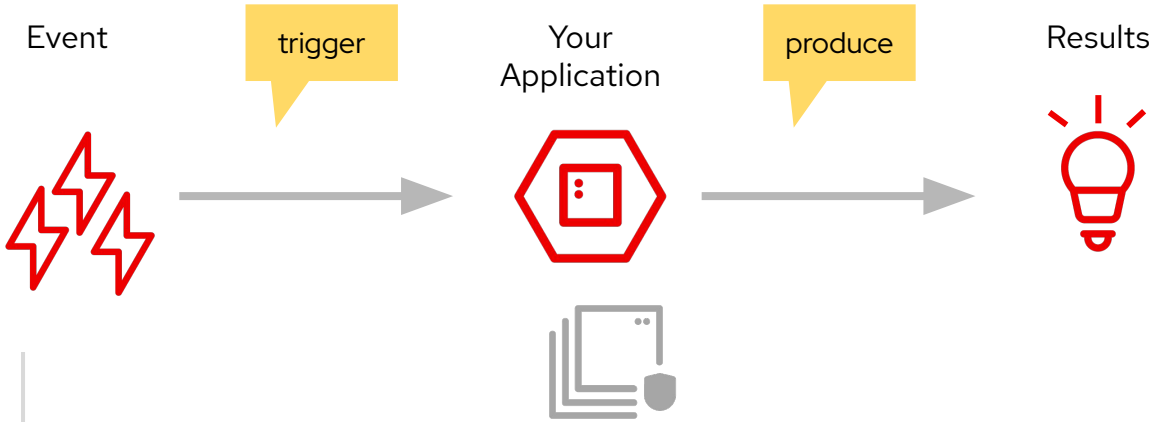
Installation experience

"Easy day 1 and even better for day 2"

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



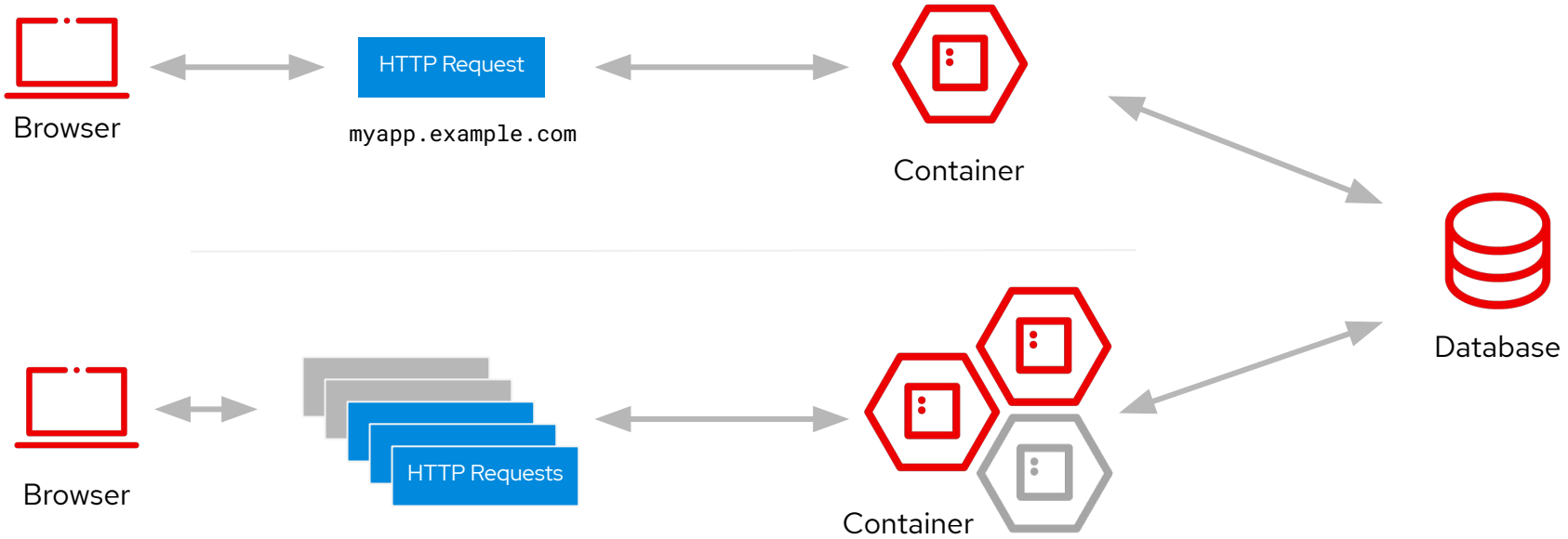
The "Serverless Pattern"



- HTTP Requests
- Kafka Messages
- Image Uploaded
- New Order
- Login from user

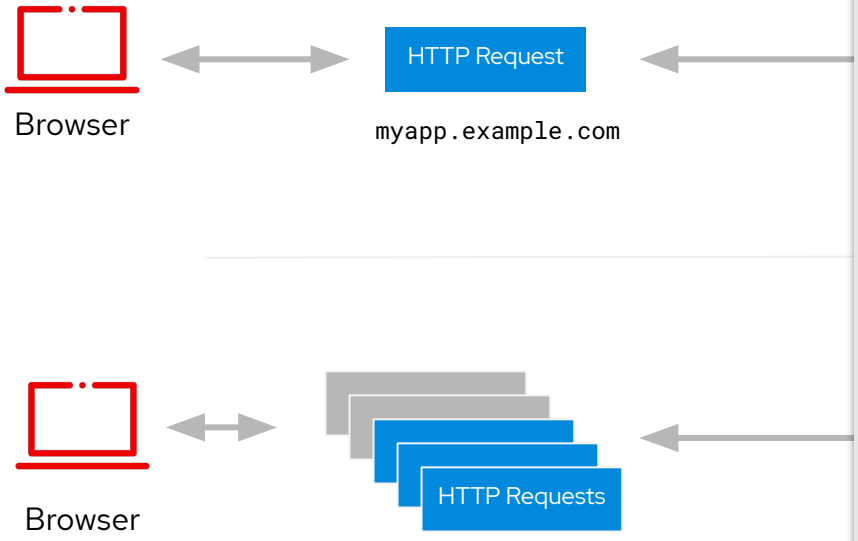
The "Serverless Pattern"

A serverless web application



The "Serverless Pattern"

A serverless web application

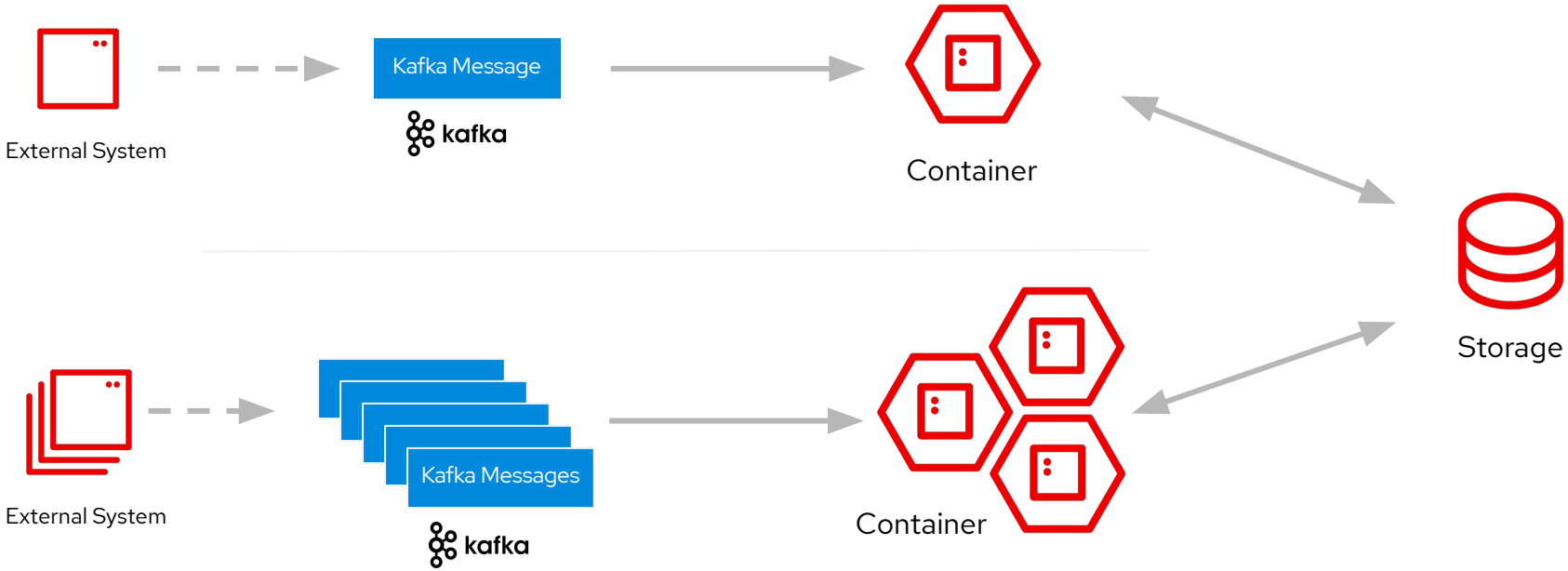


Benefits of this model:

- 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"

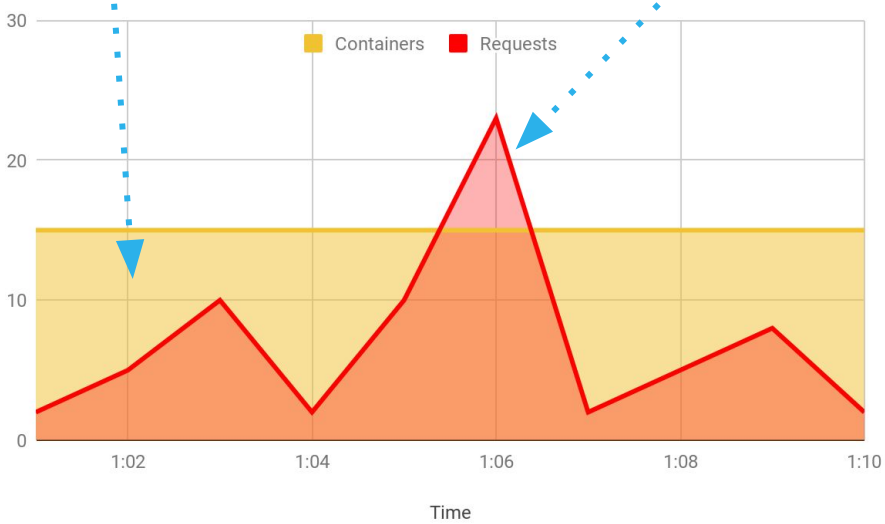
Processing a Kafka message



Serverless Operational Benefits

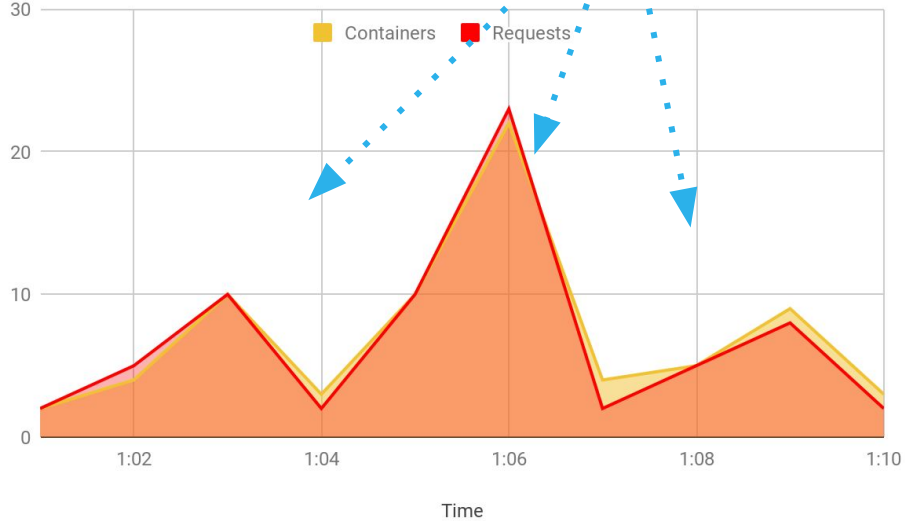
Over provisioning
Time in capacity planning
IT cost of idle resources

Under provisioning
Lost business revenue
Poor quality of service



NOT Serverless

More applications
Direct line between IT
costs & business revenue



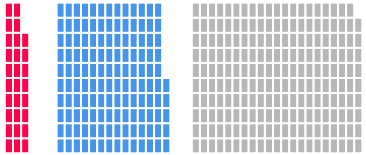
with Serverless

Choosing the Right Tool

Your Team

The Ecosystem

Performance

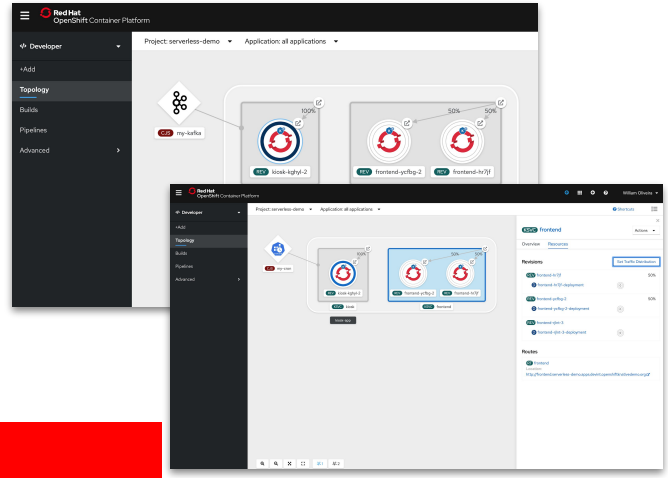


LOGIC

Developer experience

- Java
- Node
- Python
- Go
- Ruby
- Ruby on Rails
- PHP
- Perl
- .NET Core

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



CLI

UI

OpenShift Serverless

OPENSHIFT

Red Hat Enterprise Linux CoreOS

Physical

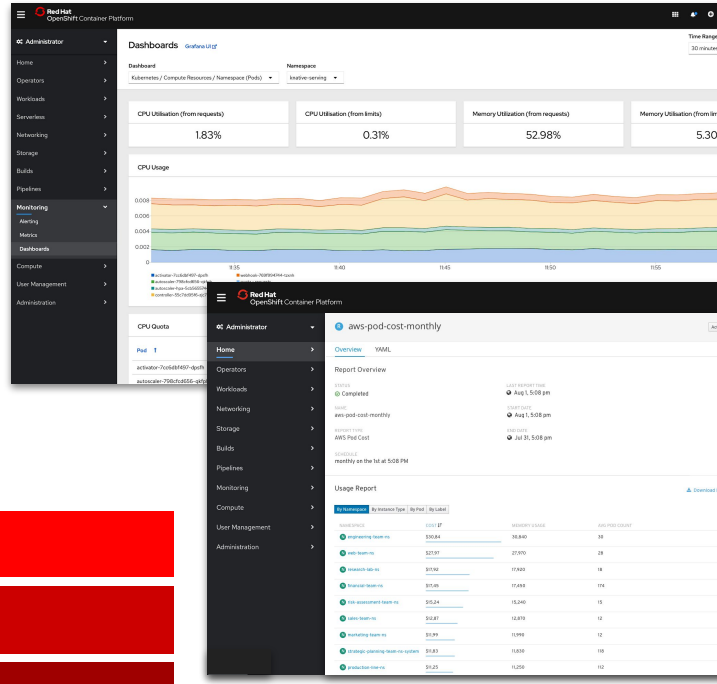
Virtual

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



UI

OpenShift Serverless

OPENSHIFT

Red Hat Enterprise Linux CoreOS

Physical | Virtual | Private cloud | Public cloud

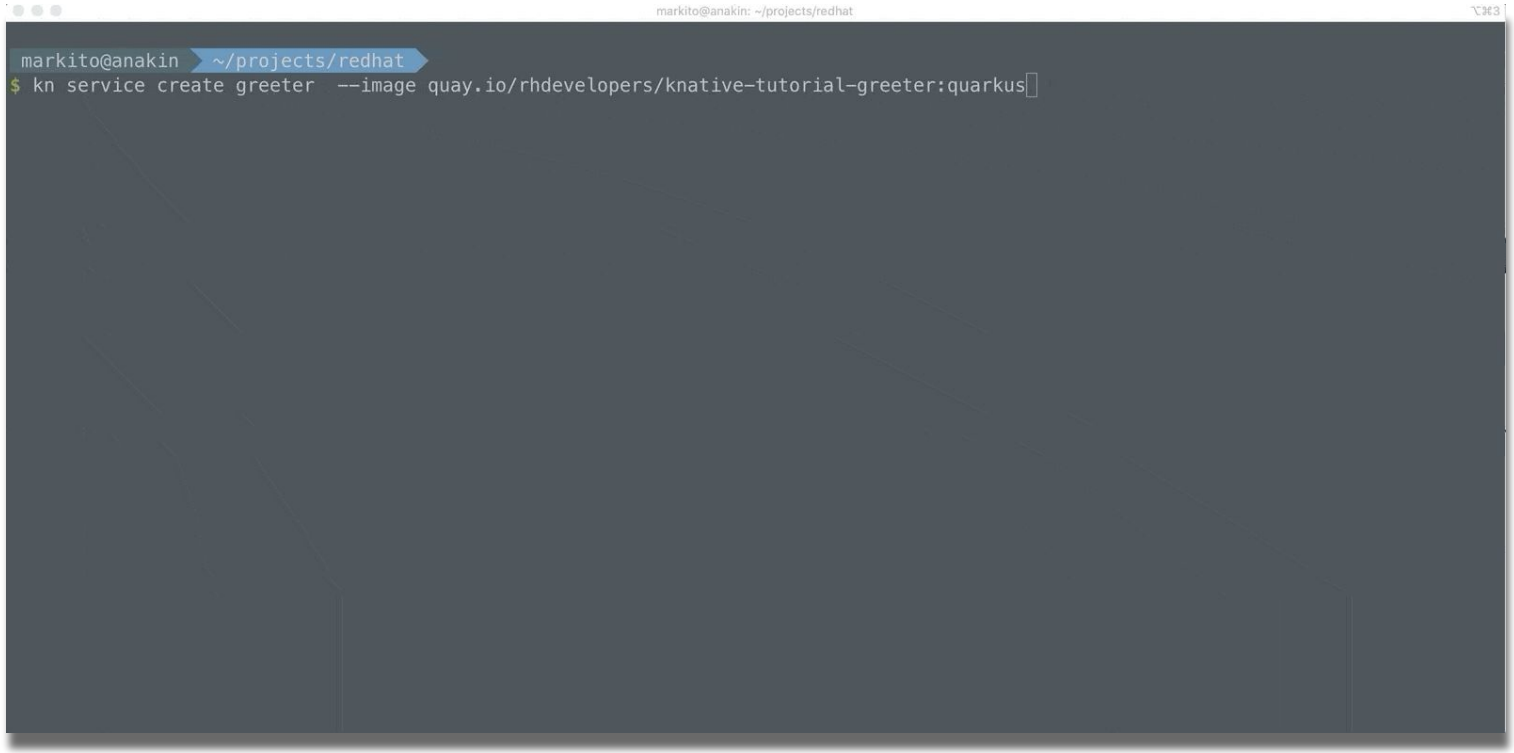
Command Line Experience

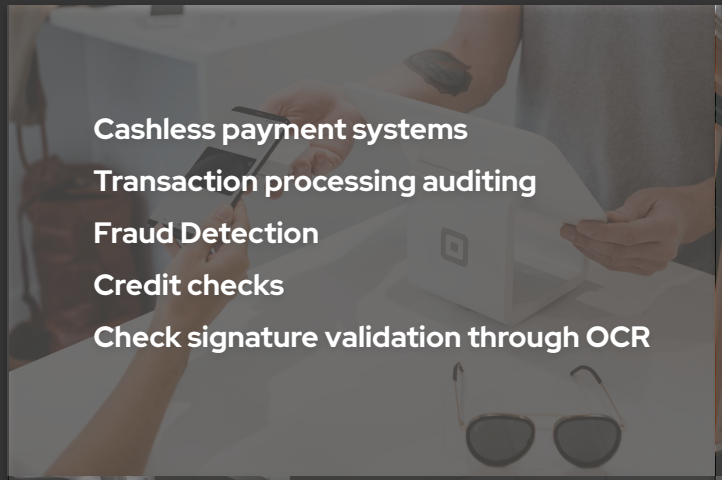
```
$ kn service create myService --image=[registry/mycontainer:v1] --min-scale=1 --max-scale=100
```

```
$ 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!





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

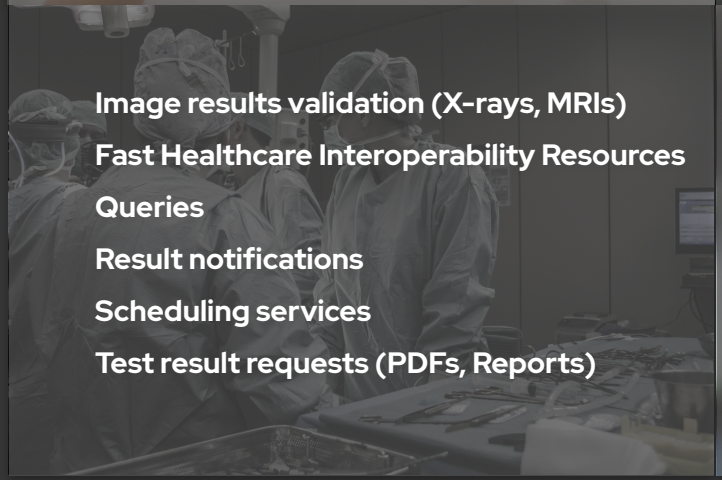
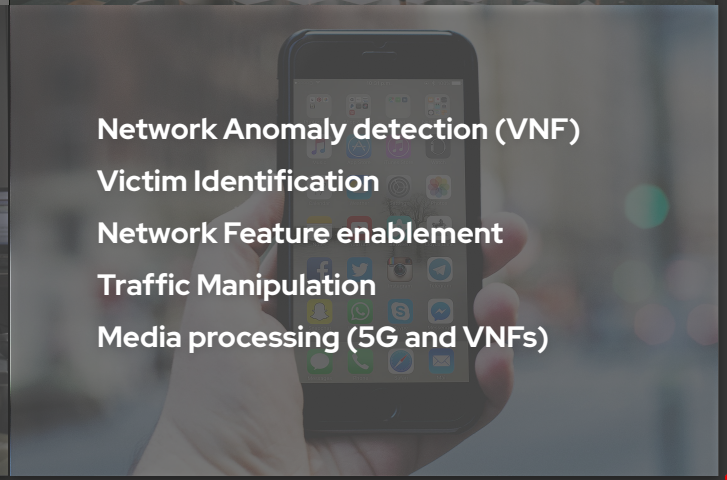
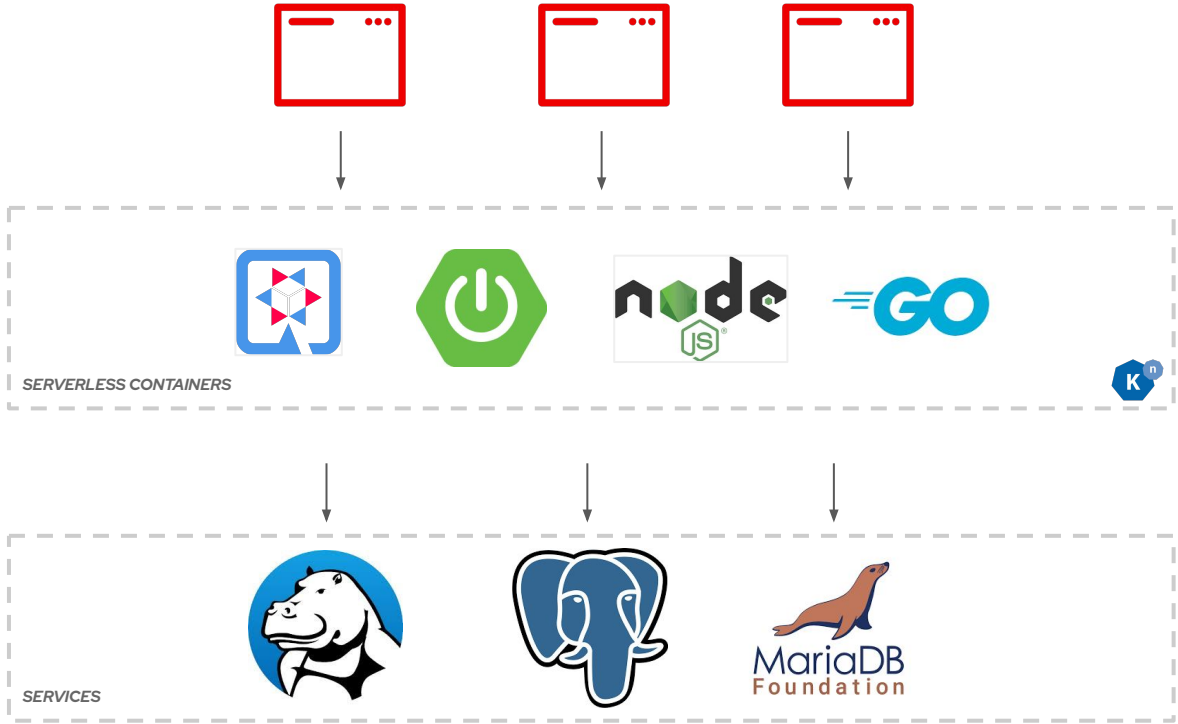


Image results validation (X-rays, MRIs)
Fast Healthcare Interoperability Resources
Queries
Result notifications
Scheduling services
Test result requests (PDFs, Reports)



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:

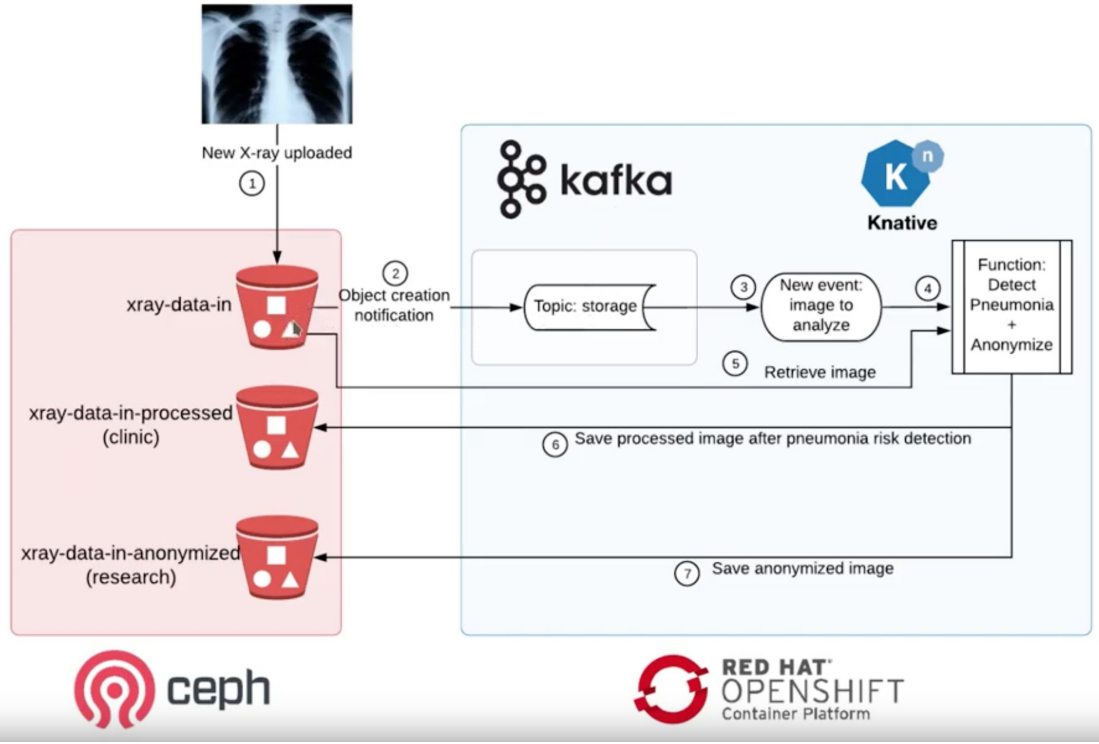
OpenJDK
python
django
VERT.X
JS
RAILS

Red Hat

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



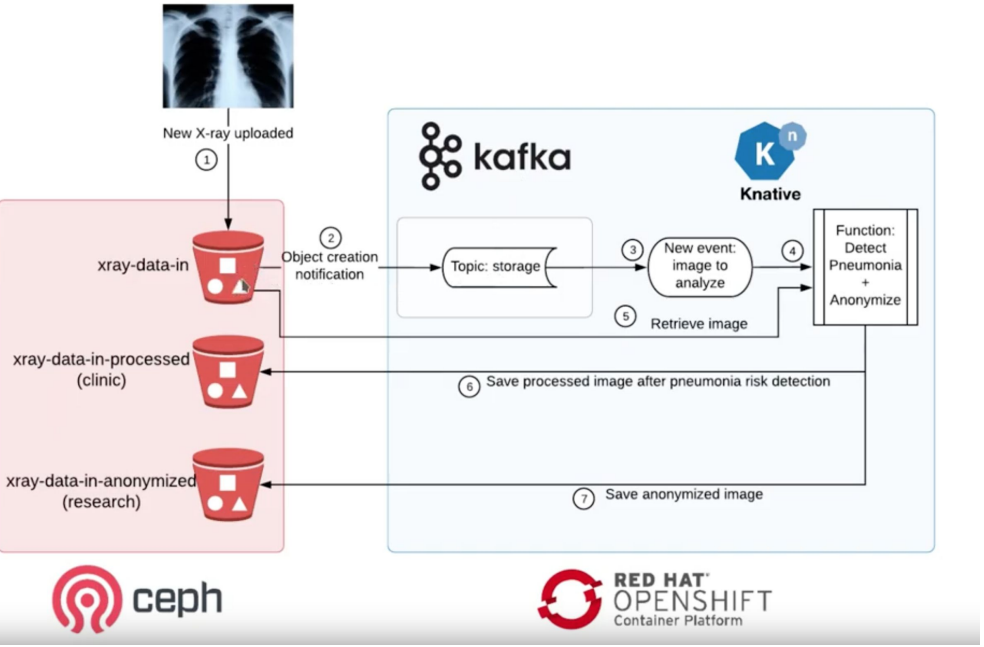
* Article: https://medium.com/analytics-vidhya/automated-data-pipeline-using-ceph-notifications-and-kserving-5e1e9b996661

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



TensorFlow

https://medium.com/analytics-vidhya/automated-data-pipeline-using-ceph-notifications-and-kserving-5e1e9b996661

Red Hat

Building on OpenShift Serverless with Red Hat Services

Connected Services
How Knative services interact with the outside world.

API Gateway
Next gen APIs still require management.

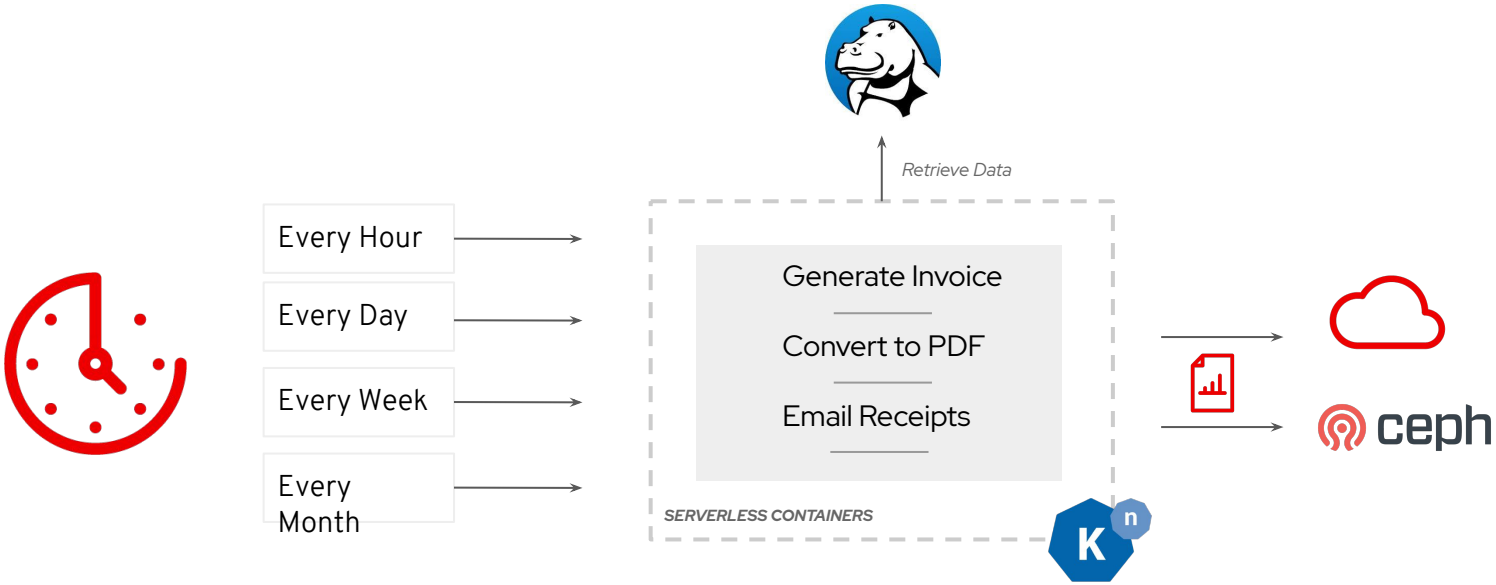
Service Orchestrator
Composing multiple services together into an application.

Implementing Services
Functions, languages, and the vagaries of cold starts.

Event Streaming
All modern architectures need some Kafka.

The Dirty Word in Serverless
Yep, you still need state to handle long-lived orchestration.

Scheduled Apps Cron Jobs

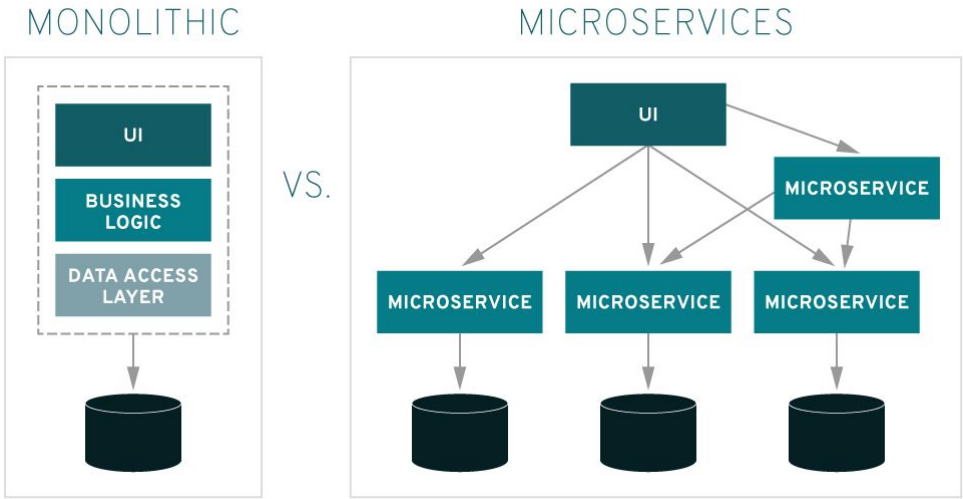


Optional - ServiceMesh / Istio

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



Red Hat

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

DevOps and Deployments
More failure surface, version incompatibility, untracked svcs

Services Communication Needs
Latency, concurrence, distributed transactions

Inability to Monitor & Understand Performance
More to monitor & different types of monitoring required

Securing Services
Malicious requests, DoS, id & access control

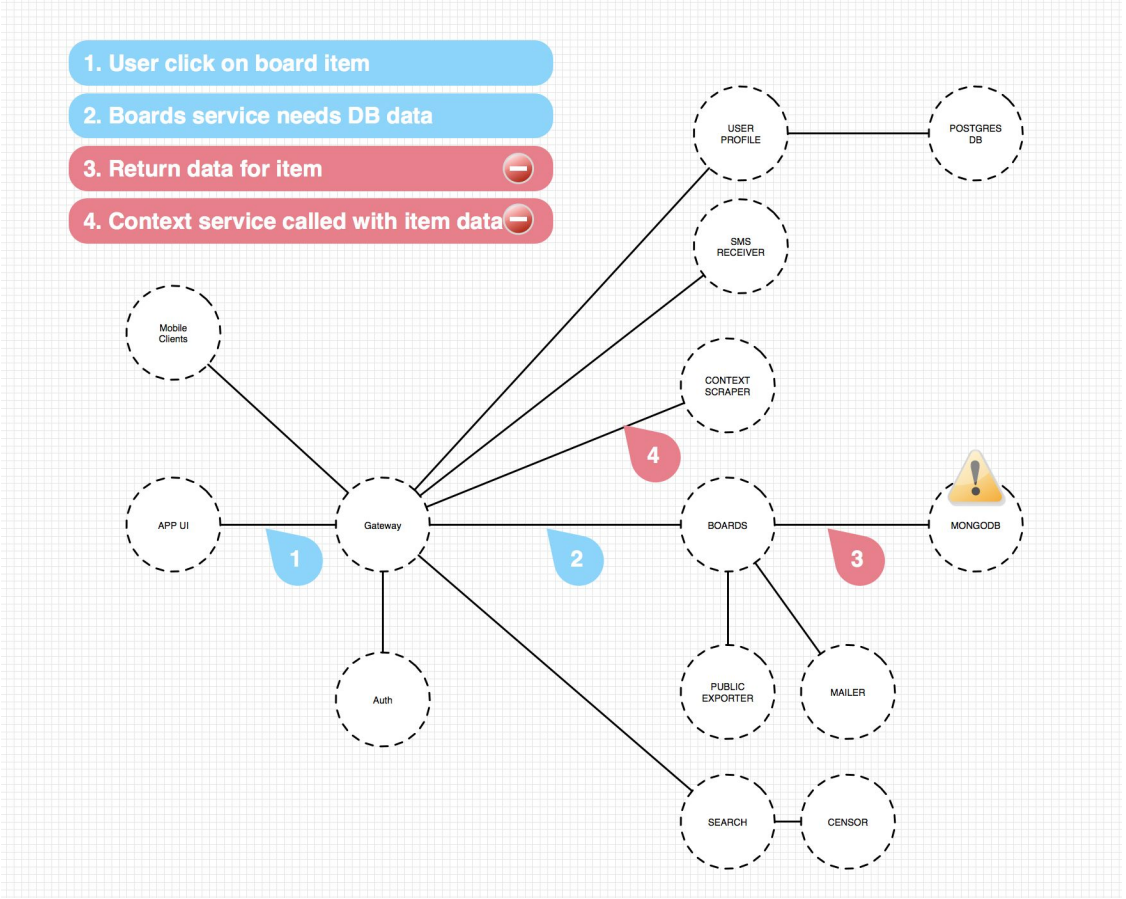
Highly Distributed Logs
Scattered logs, lots more logs to manage, access control

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

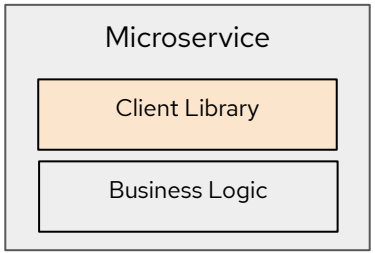
Partial Failure → Cascading Failures

- 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



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

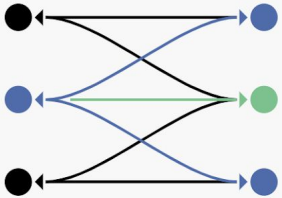
Solution results in clients with bulk of

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

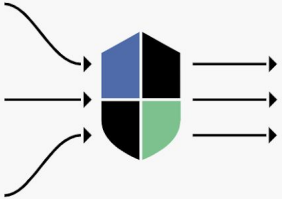


Connect

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

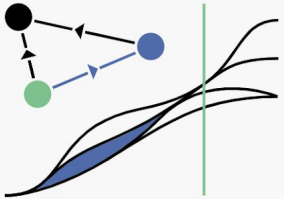
Secure

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



Control

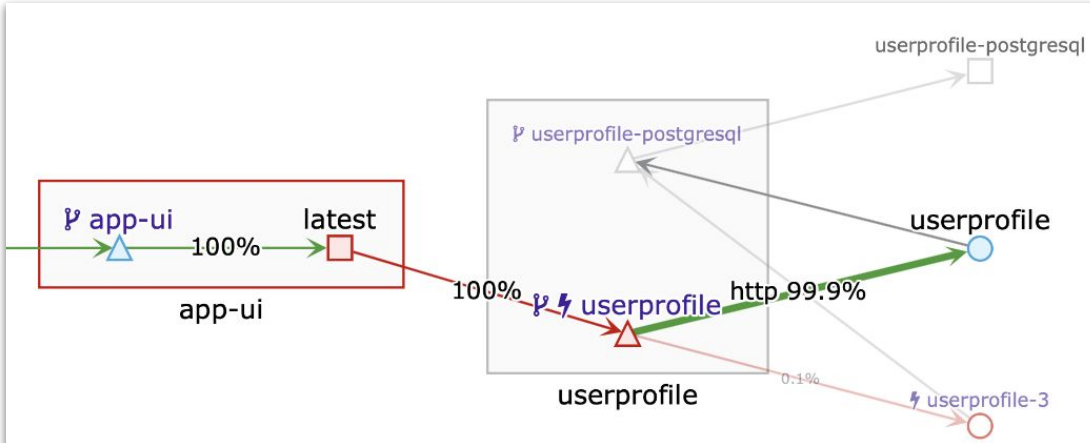
Apply policies and ensure that they're enforced, and that resources are fairly distributed among consumers.

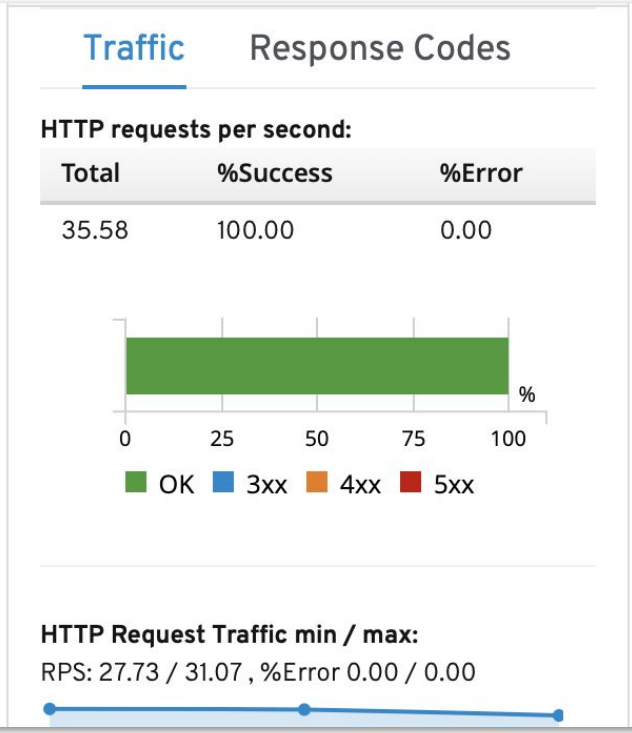


Observe

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

Handling Partial Failures with the Service Mesh





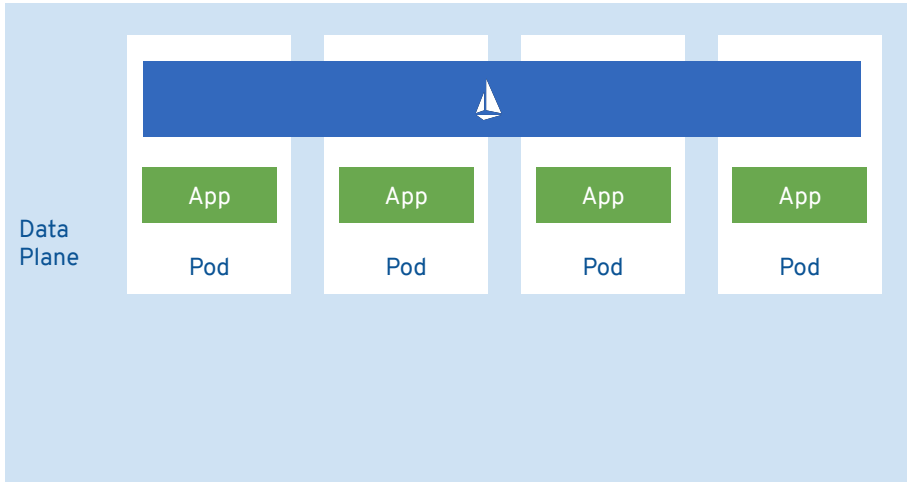

```
26      trafficPolicy:
27        connectionPool:
28          http:
29            http1MaxPendingRequests: 1
30            maxRequestsPerConnection: 1
31        outlierDetection:
32          consecutiveErrors: 1
33          interval: 1s
34          baseEjectionTime: 10m
35          maxEjectionPercent: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

YAML configurable,
(Kubernetes native)

Red Hat

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

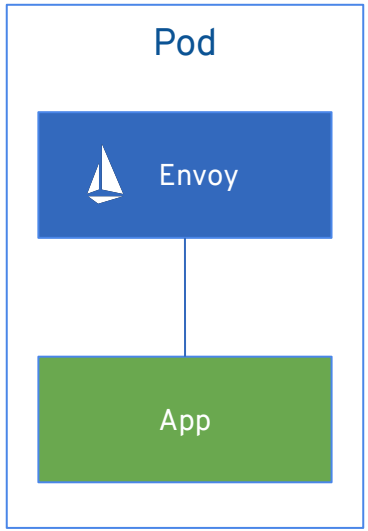
Your Services are in a “Data Plane”



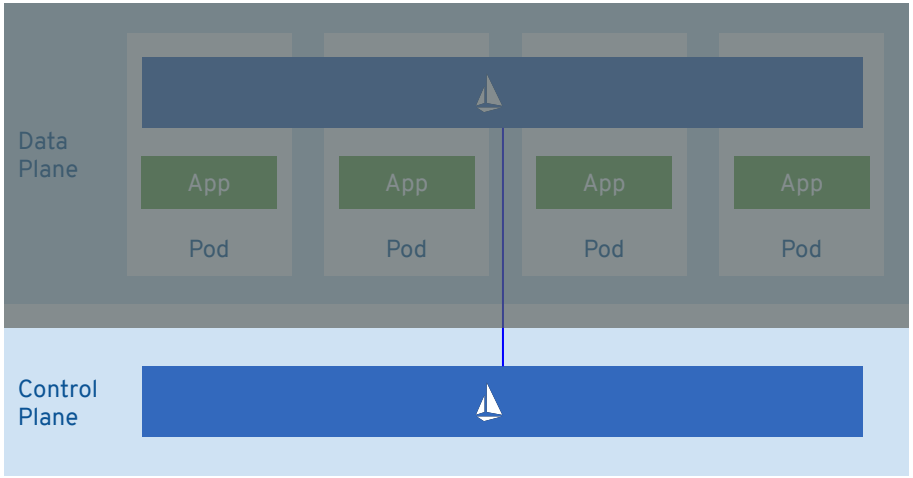
RED HAT® OPENSHIFT

Red Hat

What's a Sidecar have to do with containers?



Your Policy Makes Up a “Control Plane”



RED HAT® OPENSHIFT

Red Hat

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)

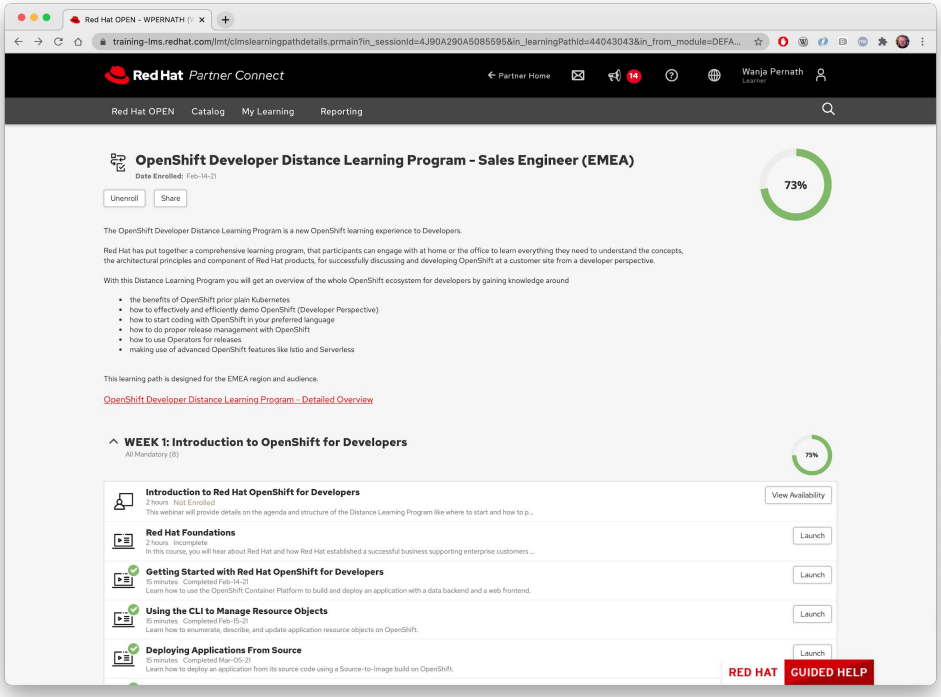
Homework - The presentation

- 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

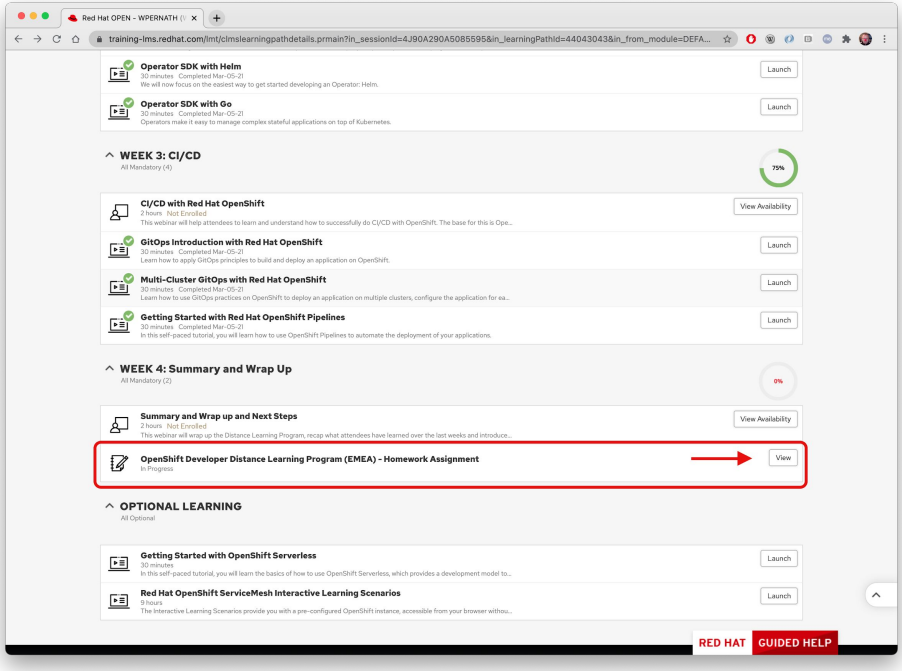
- Create your own demo script, which should contain
  - A scenario you'd like to address
  - Take one of the technologies you've learned over the course
    - Odo, Helm, Kustomize, Tekton, GitOps, Serverless, ServiceMesh…
  - Describe, why you have chosen this technologie and why you think this would 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



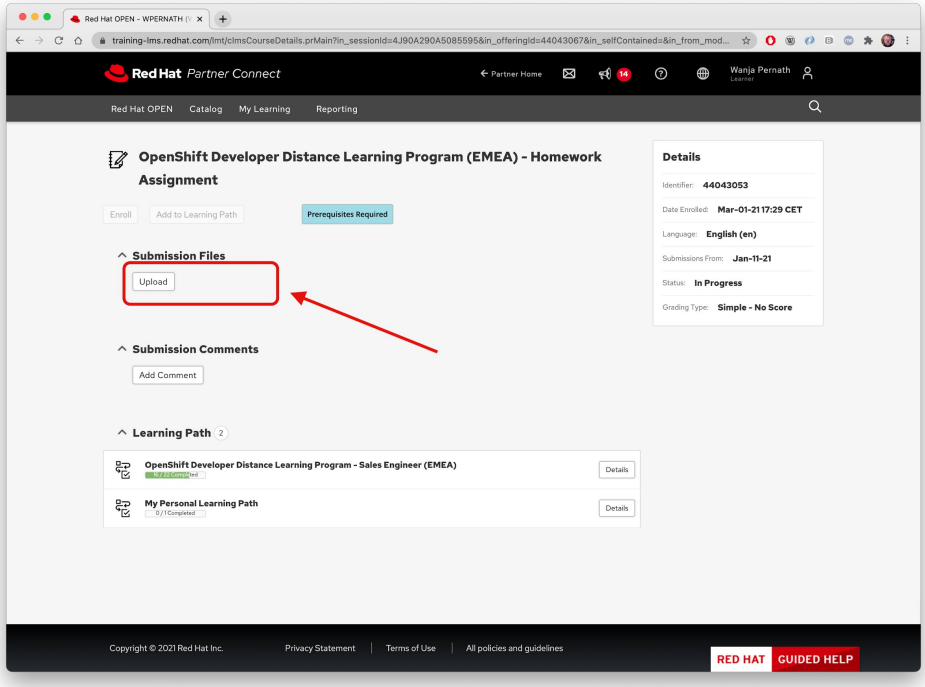
- Open your OPEN progress
- Scroll down to Week #4

Homework - Upload your work



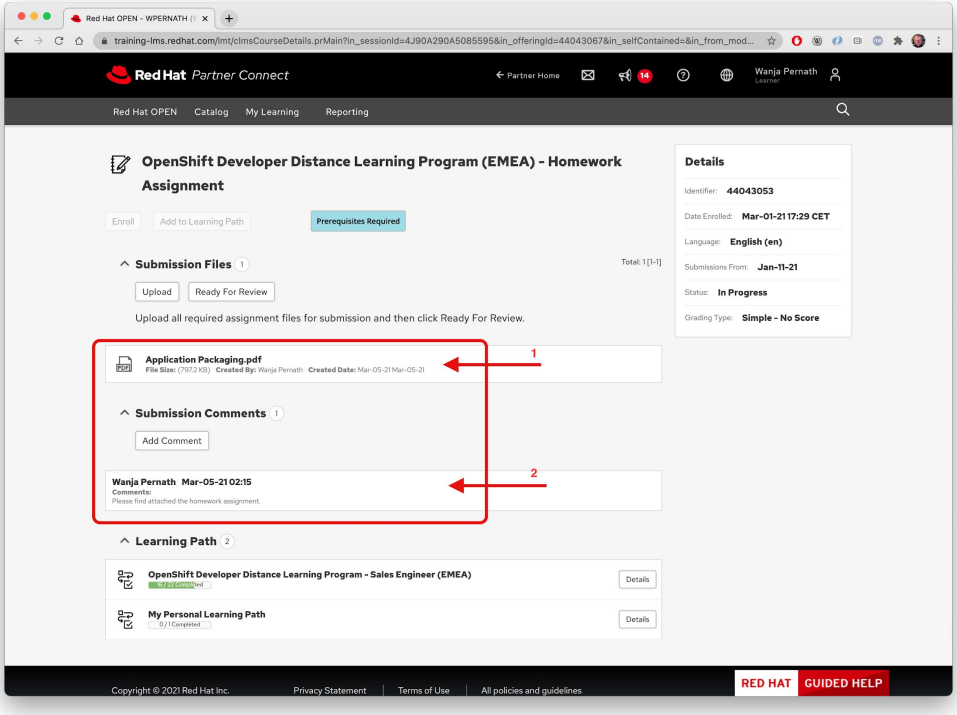
- Click on “View” of the entry “Homework Assignment”

Homework - Upload your work



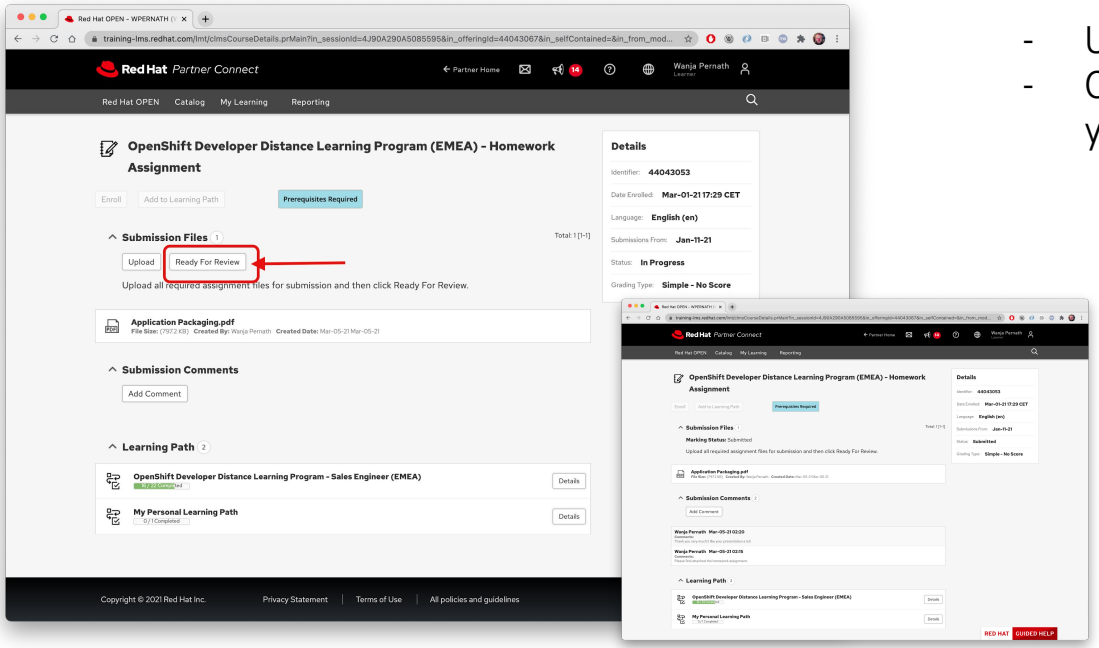
- Click on Upload
- Choose your File (presentation or video)

Homework - Upload your work



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

Homework - Upload your work



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

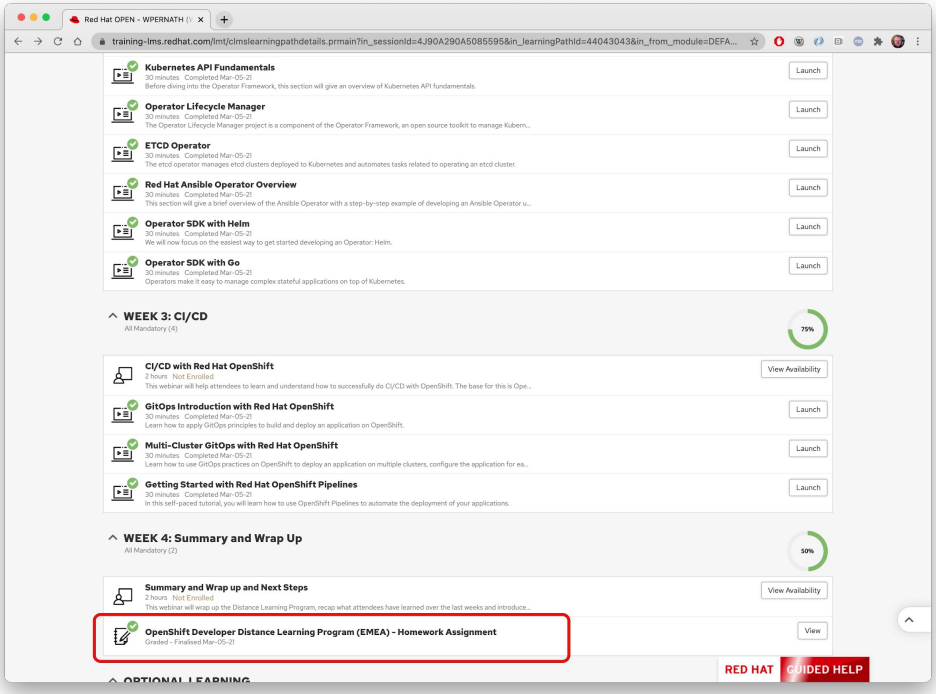
Homework - Let me have a look at your work

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)

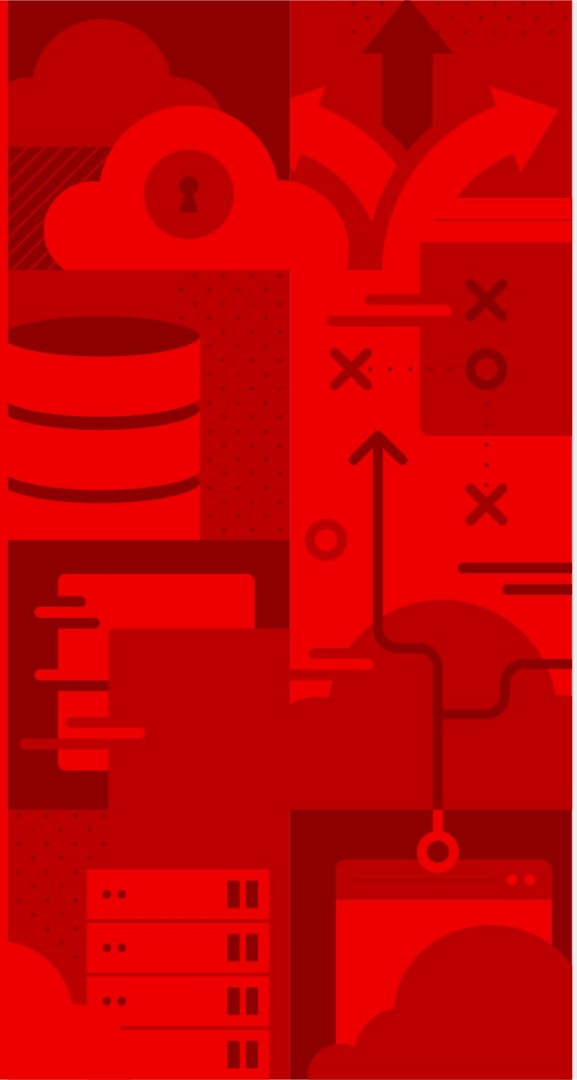
Homework - Upload your work



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



Thank you

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat