



# Container hybrid deployment to the EDGE powered by Openshift & RHEL

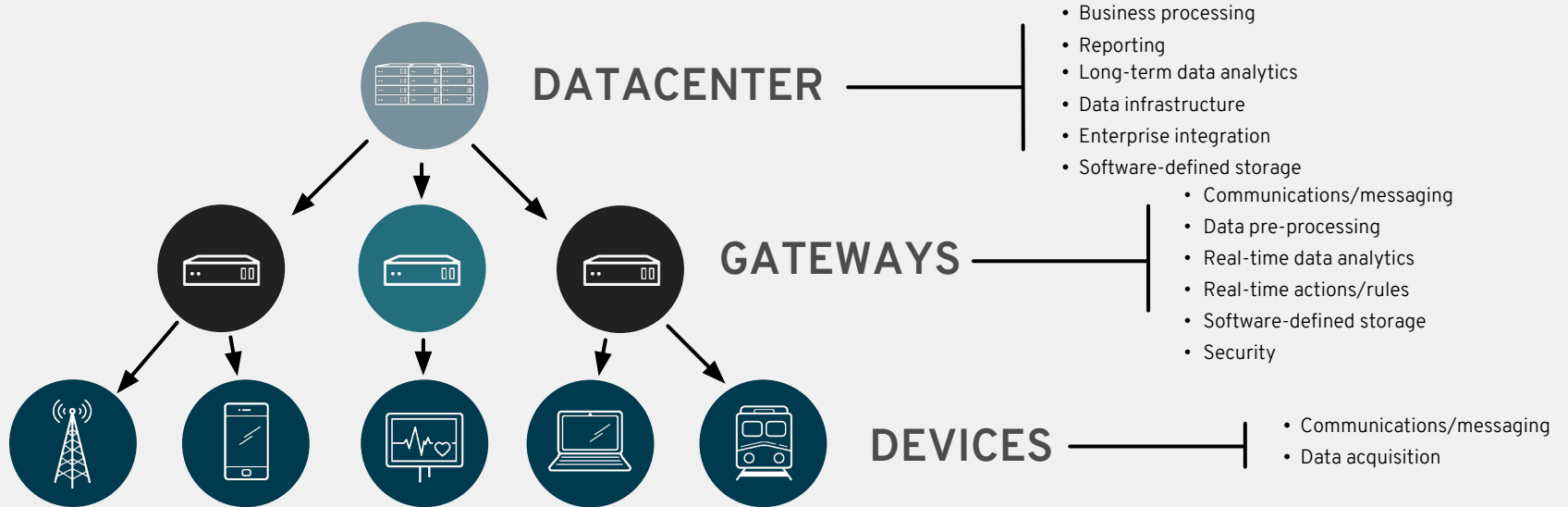
**Alessandro Arrichiello**  
Solution Architect  
ale@redhat.com



#RedHatOSD

# ENTERPRISE IoT ARCHITECTURE

Driving datacenter function to the edge



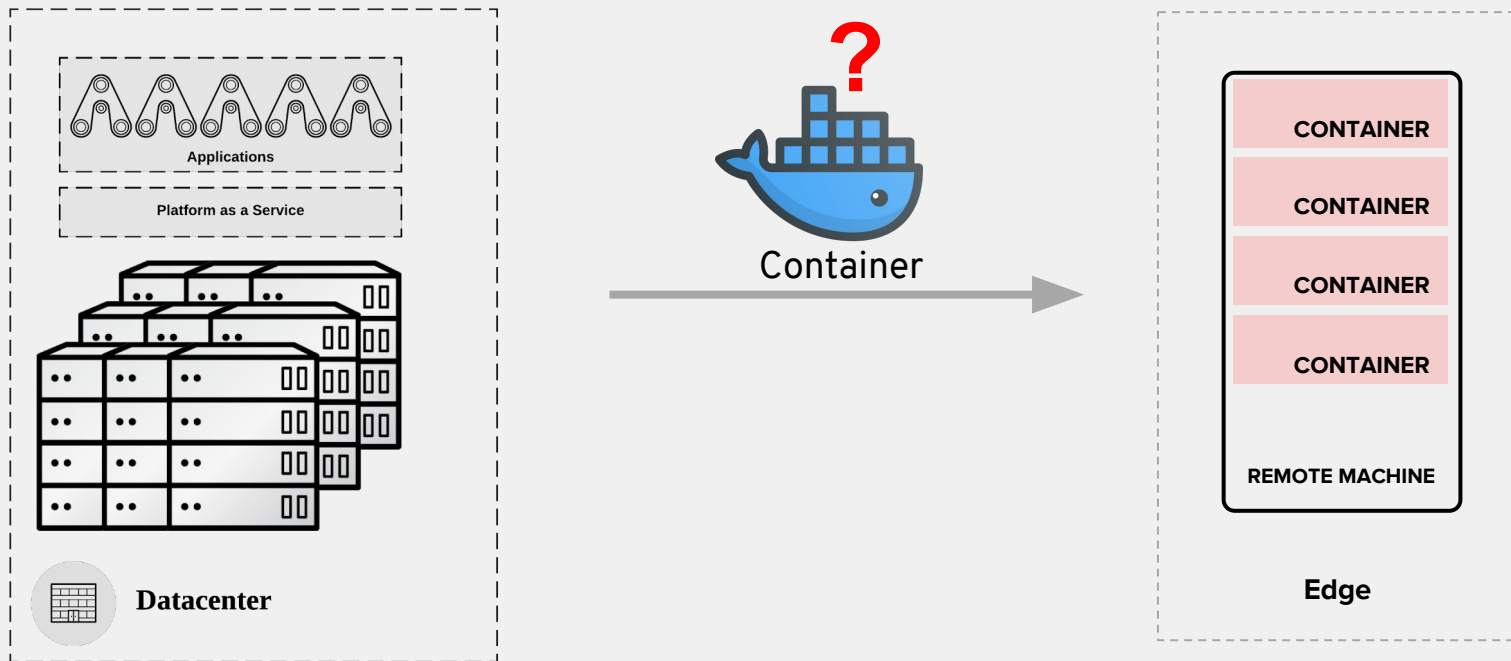
# EXAMPLES OF TARGETED USE CASES

Remote factories, disconnected ferries, trains, oil stations

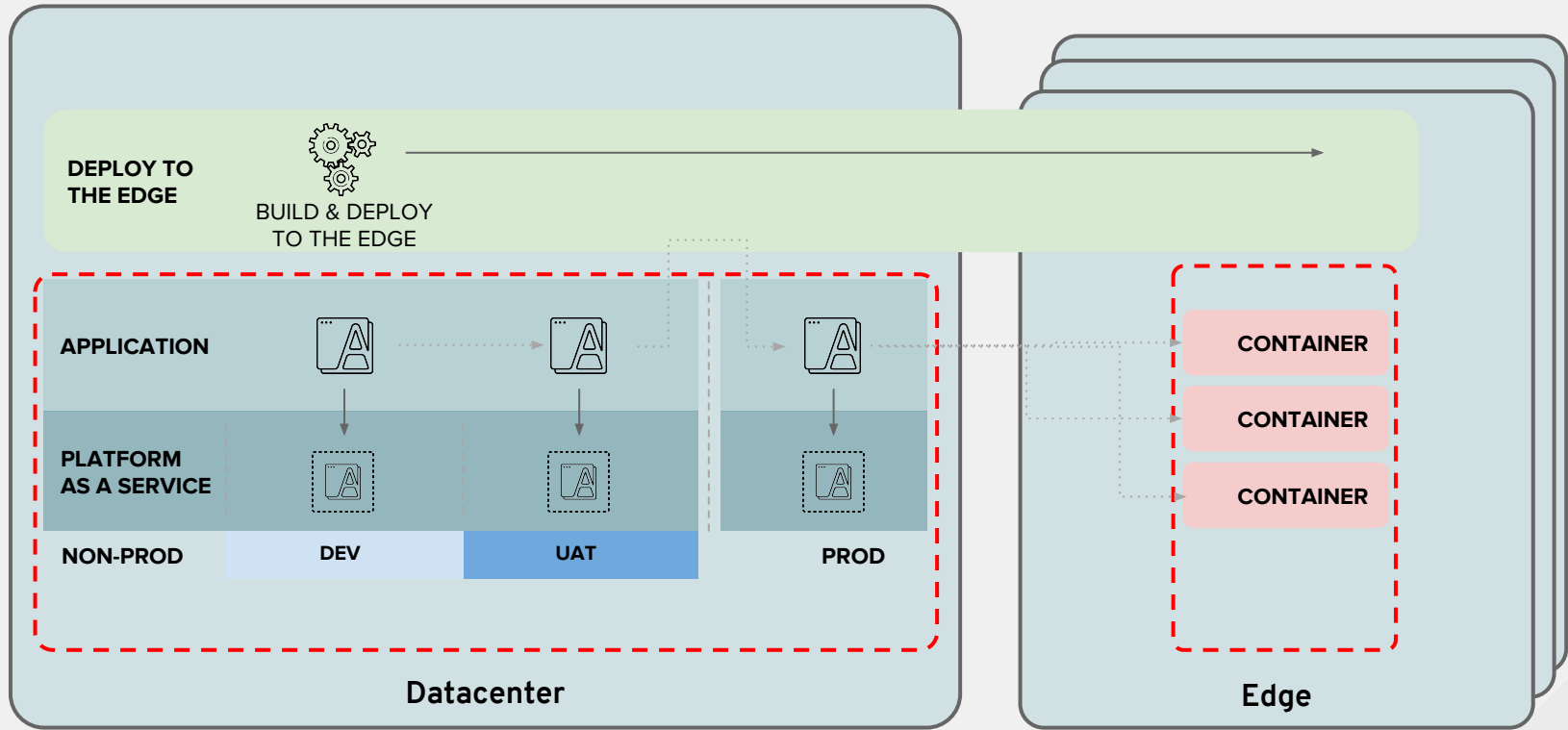


# CUSTOMER NEEDS

How to handle containers deployment to the edge?

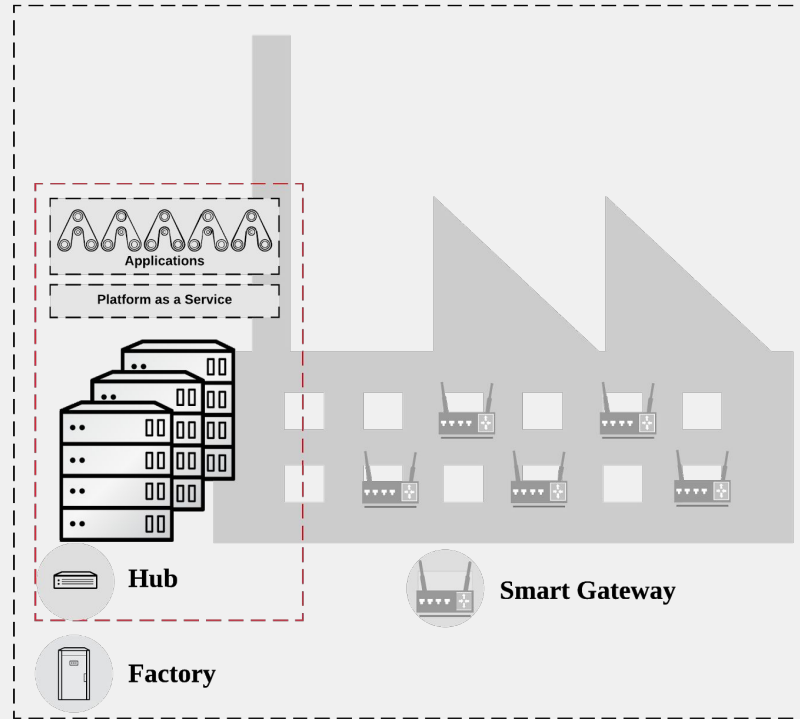
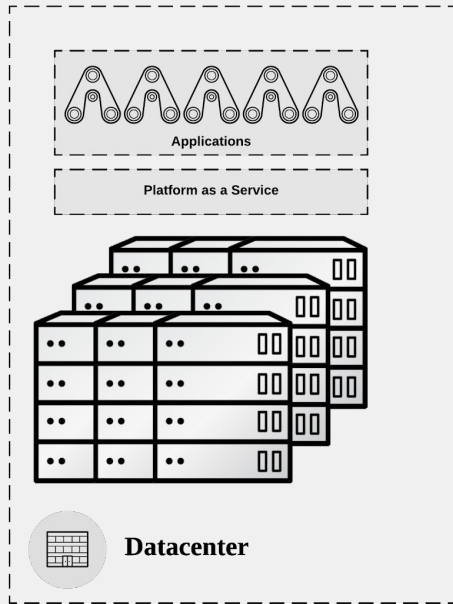


# FROM DEVELOPMENT TO THE EDGE

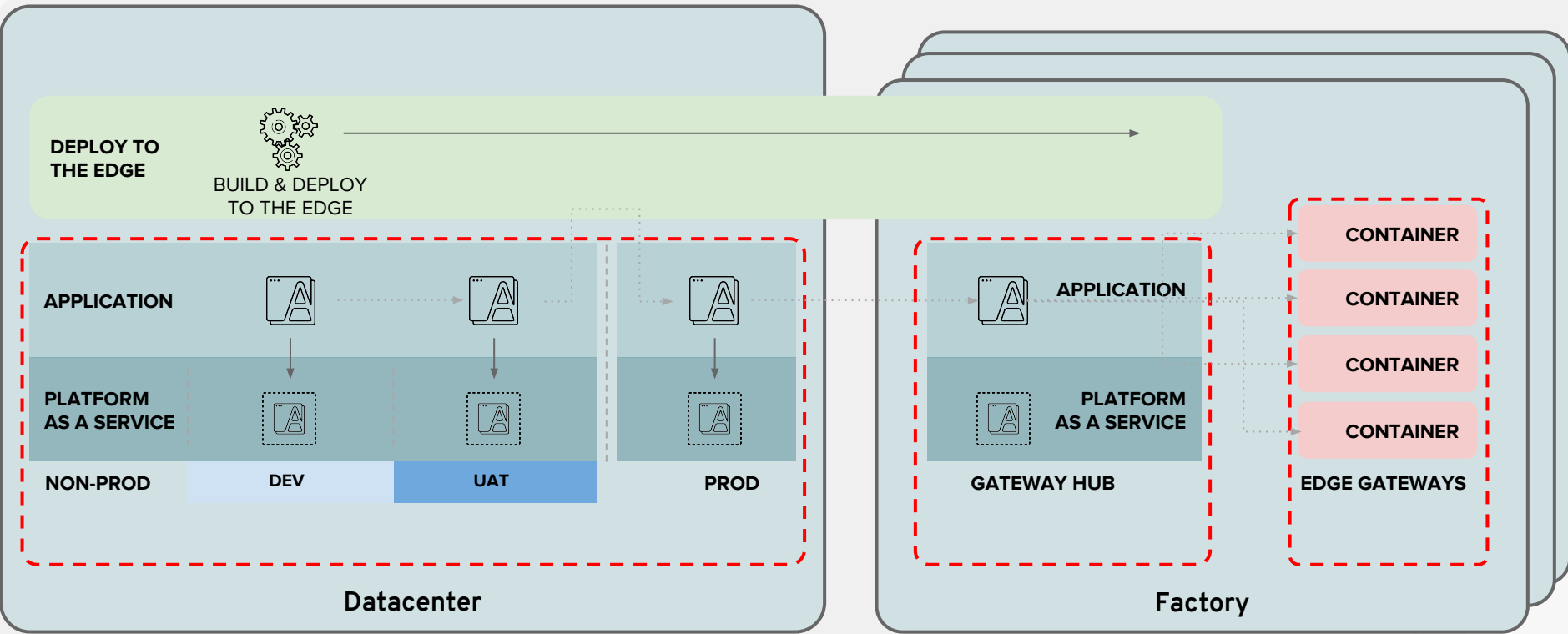


# USE CASE SCENARIO

From Datacenter to the Factory

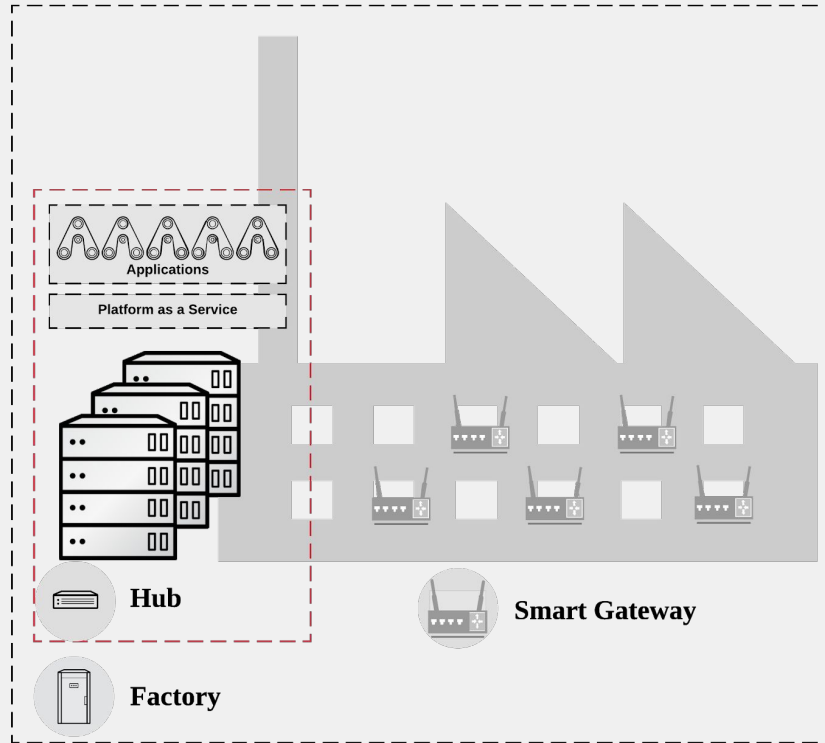
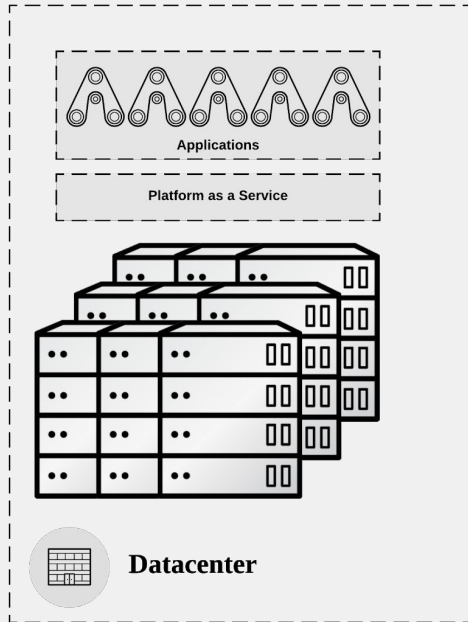


# FROM DEVELOPMENT TO EDGE DEPLOYMENTS



# USE CASE SCENARIO

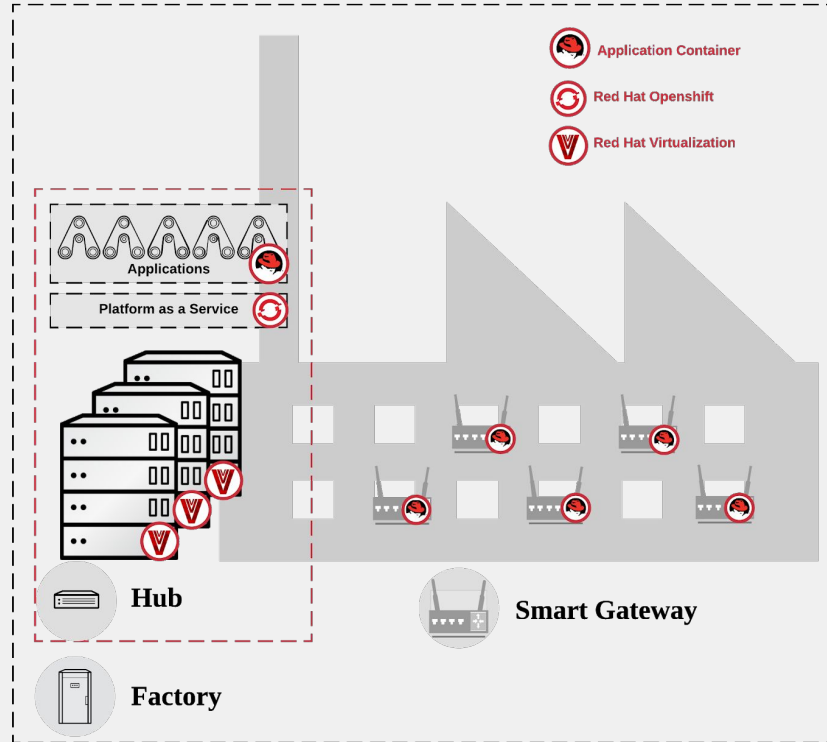
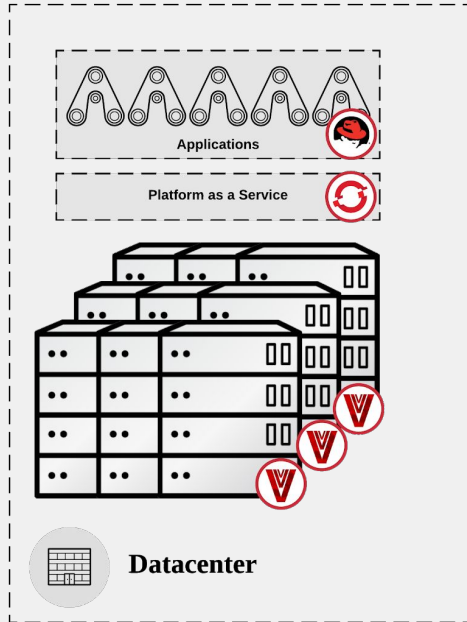
From Datacenter to the Factory





# THE RED HAT STACK

Can support your edge deployments



# HOW DO YOU HANDLE IT?



#RedHatOSD



# MULTIPLE EDGE DEPLOYMENTS SCENARIOS

## Corporate Node



## Plant Node

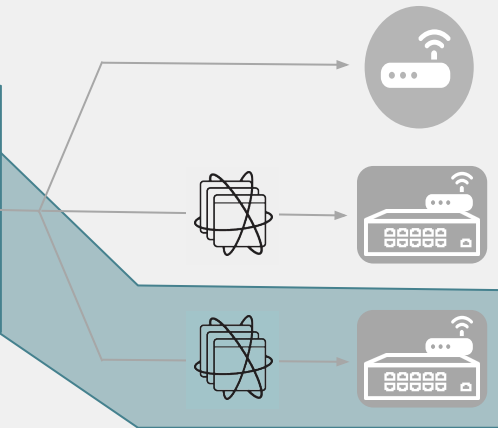
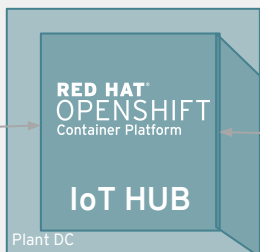


## Edge Node



**Deployment Scenarios**  
(Based on resource (CPU/Memory) and connectivity (Bandwidth/Latency) availability)

**Available capabilities**



### SCENARIO 1

Low resource (Edge Gateway)  
Non reliable connectivity

- Data gathering
- Basic analytics remotely managed

### SCENARIO 2

High resource (Edge Server)  
Reliable connectivity

- Data gathering
- Dynamic deployed containerized business applications

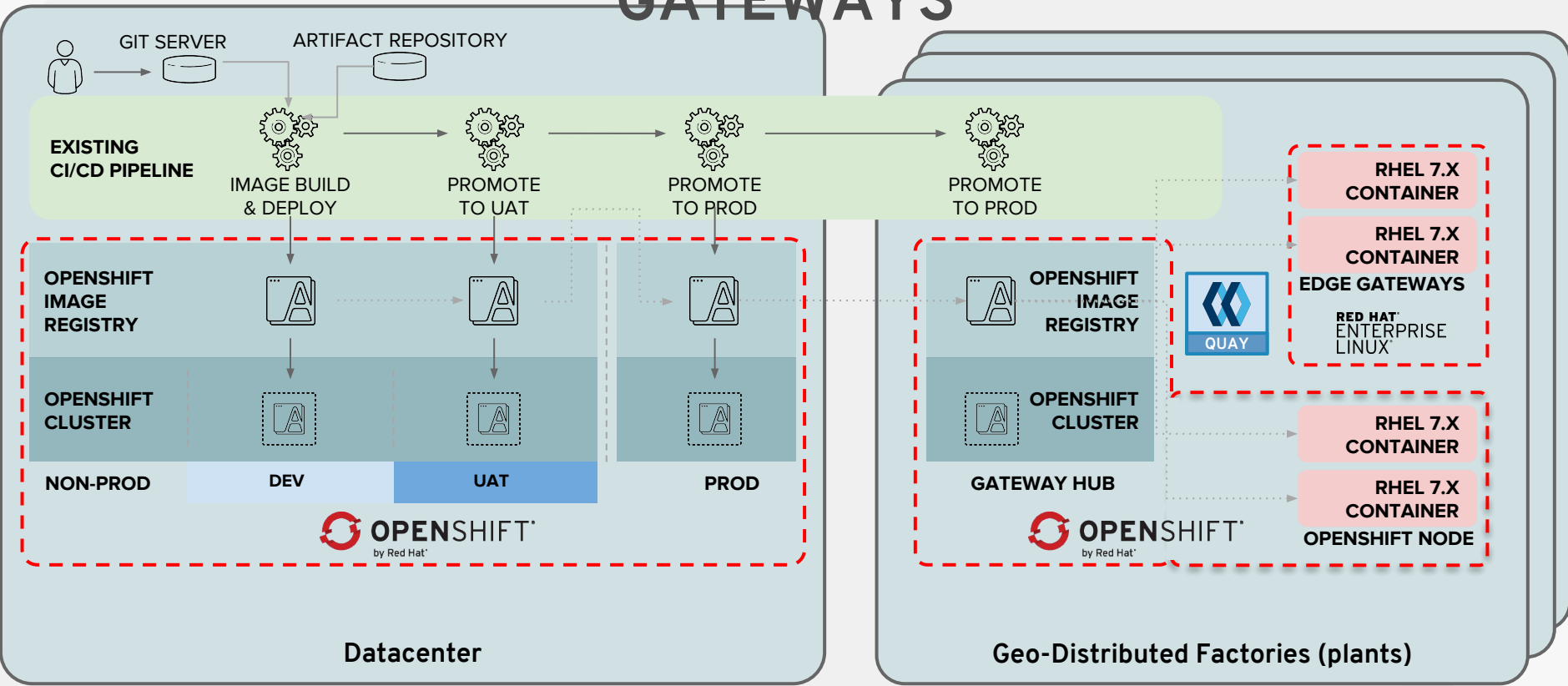
### SCENARIO 3

Available resource (Edge Server)  
High affidability connectivity

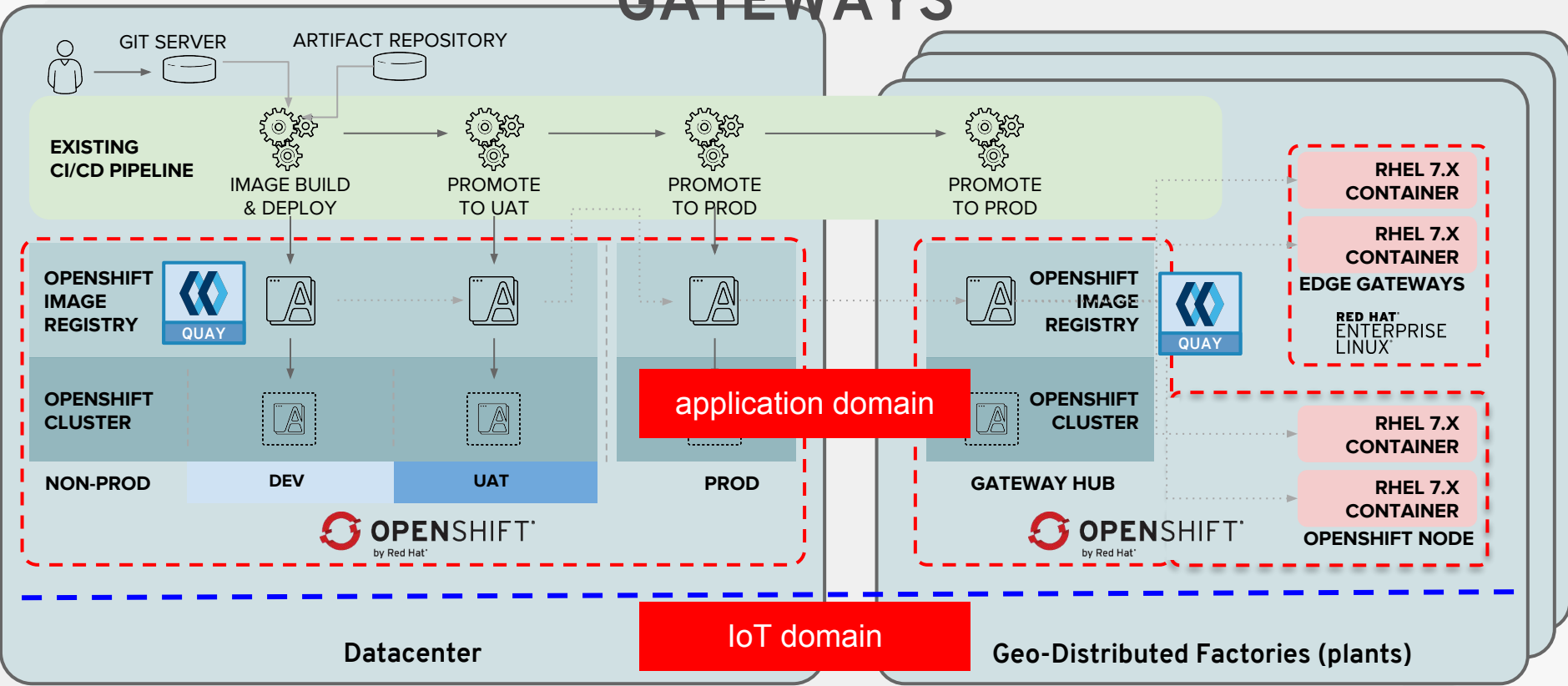
- Data gathering
- Dynamic deployed containerized business applications
- Centralized Management



# CI/CD THROUGH DATACENTERS & GATEWAYS

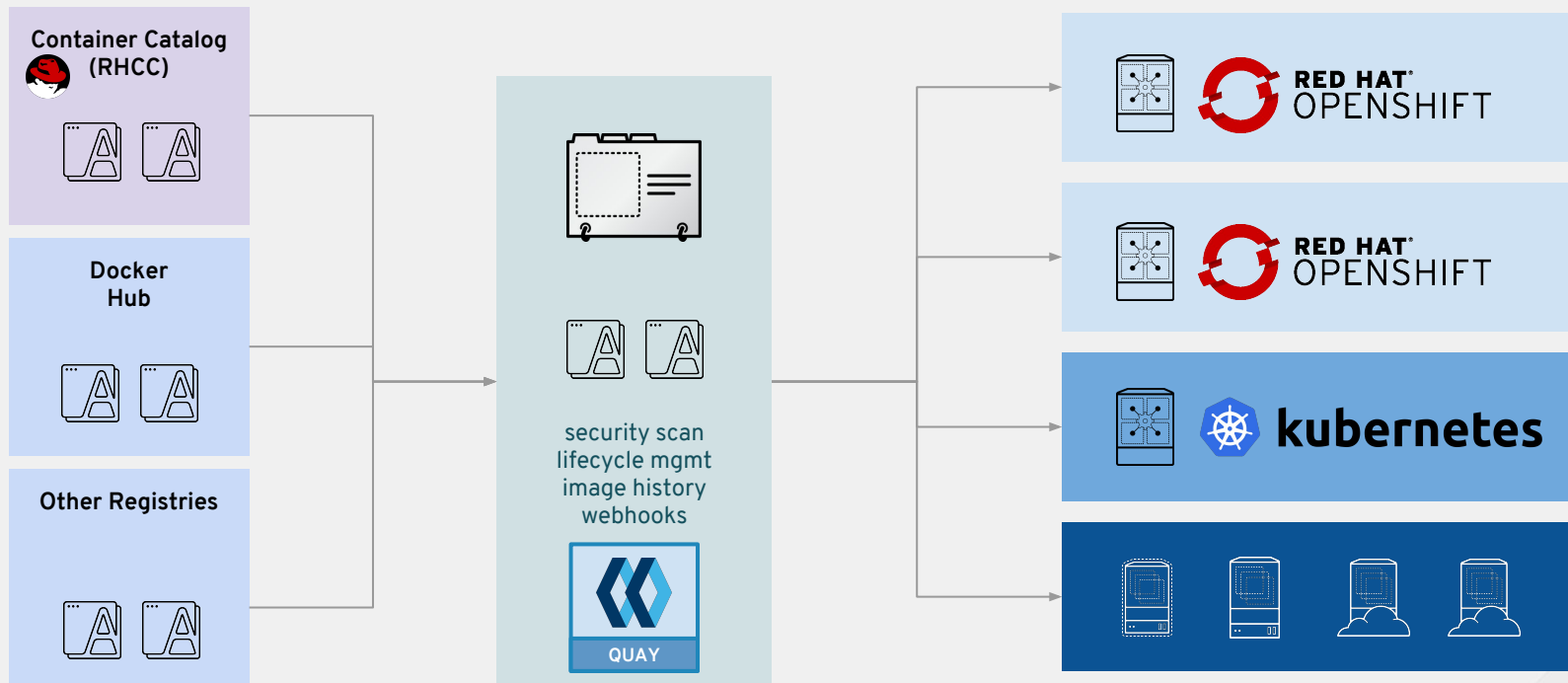


# CI/CD THROUGH DATACENTERS & GATEWAYS



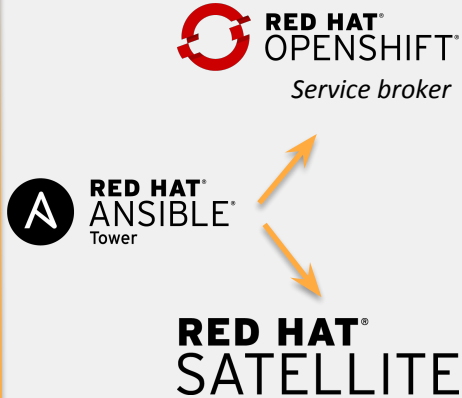
# IMAGES DISTRIBUTION w/ RED HAT QUAY

Content governance and ingress for OpenShift / Kubernetes / Edge

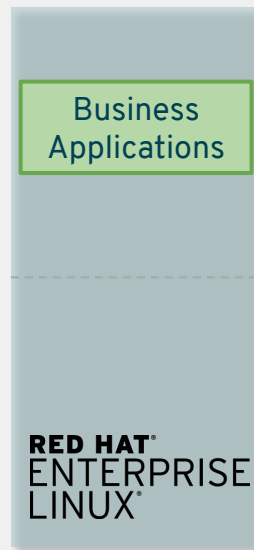
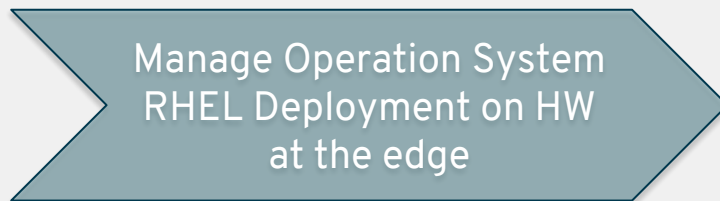
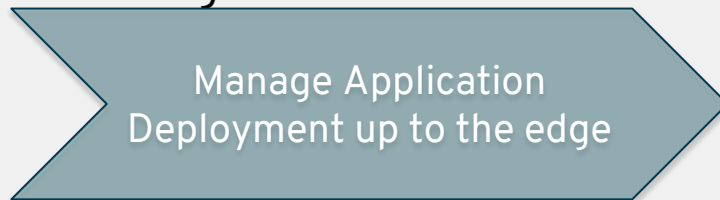


# SCALE IoT SOLUTION THROUGH CENTRALIZED AUTOMATION PROCESSES

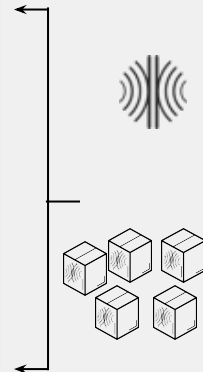
Centralized Management for...



...Automatized processes to the edge



Gateway/  
Edge Server



# WHAT ABOUT THE TECHNOLOGY?

## Six Building Blocks

1. ENTERPRISE READY  
VIRTUALIZATION LAYER
2. CONTAINER BASED  
PLATFORM AS A SERVICE
3. AUTOMATION ENGINE  
ENTERPRISE FRAMEWORK
4. CRITICAL RELIABLE &  
MILITARY-GRADE SECURE  
OS
5. O.S. INSTALLATION &  
MANAGEMENT

**RED HAT®**  
VIRTUALIZATION

**RED HAT®**  
OPENSIFT  
Container Platform



**RED HAT®**  
ENTERPRISE LINUX®

**RED HAT®**  
SATELLITE

**RED HAT®**  
QUAY



#RedHatOSD





# A REAL USE CASE: INTELLIGENT IoT GATEWAY

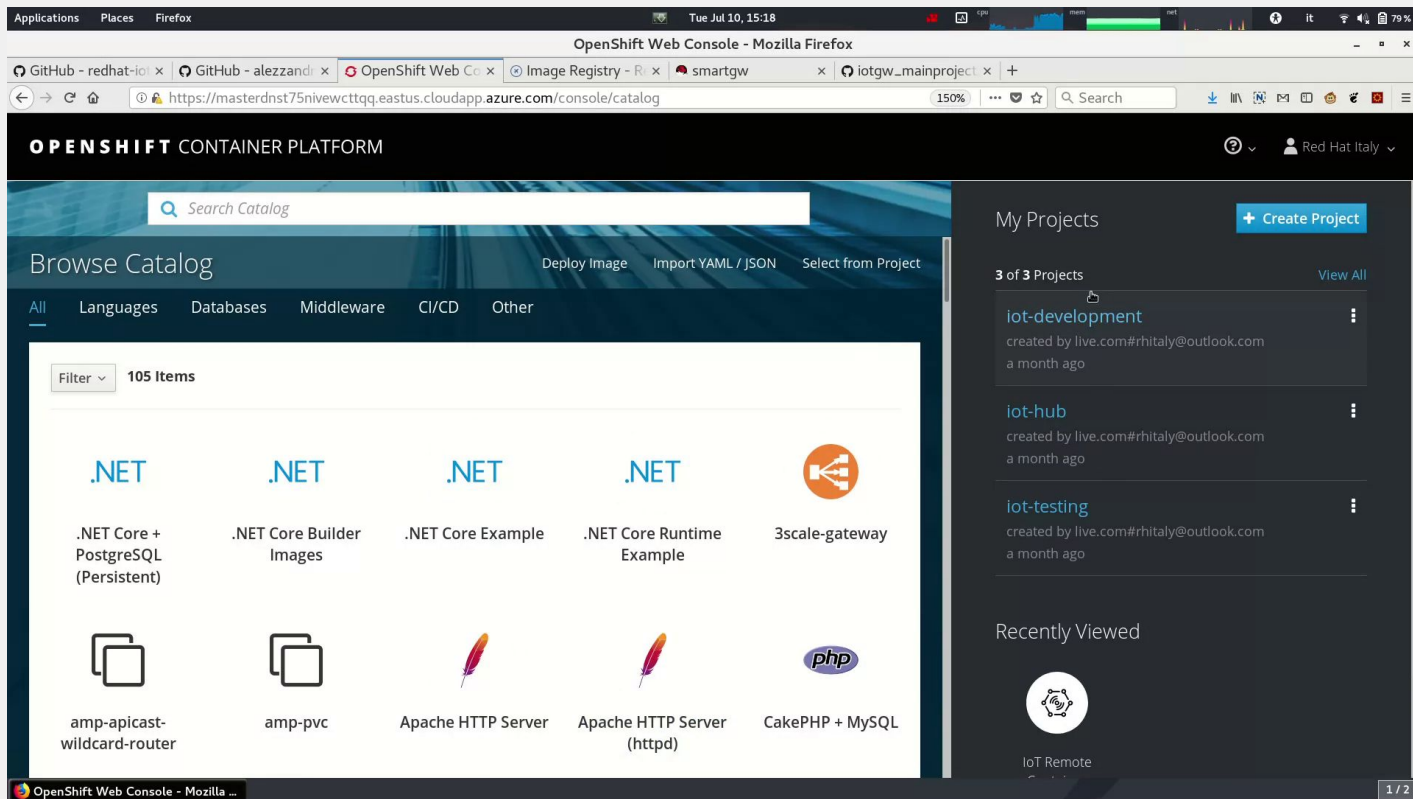
<https://www.youtube.com/watch?v=bNipu5OA1o4>



#RedHatOSD



# THE PLATFORM: OPENSSHIFT



The screenshot displays the OpenShift Web Console interface. At the top, the browser address bar shows the URL `https://masterdnst75nivevcttq.eastus.cloudapp.azure.com/console/catalog`. The main header reads "OPENSSHIFT CONTAINER PLATFORM" with a search bar and navigation options like "Deploy Image", "Import YAML / JSON", and "Select from Project". Below this, the "Browse Catalog" section is active, showing a filter for "105 Items" and a grid of application templates. The visible templates include:

- .NET Core + PostgreSQL (Persistent)
- .NET Core Builder Images
- .NET Core Example
- .NET Core Runtime Example
- 3scale-gateway
- amp-apicast-wildcard-router
- amp-pvc
- Apache HTTP Server
- Apache HTTP Server (httpd)
- CakePHP + MySQL

On the right side, a "My Projects" sidebar lists three projects: "iot-development", "iot-hub", and "iot-testing", each created by "live.com#rhitaly@outlook.com" a month ago. Below this is a "Recently Viewed" section showing an "IoT Remote" project.



# RED HAT LAB: INTELLIGENT IoT GATEWAY

Demo firstly developed for Red Hat Summit 2016

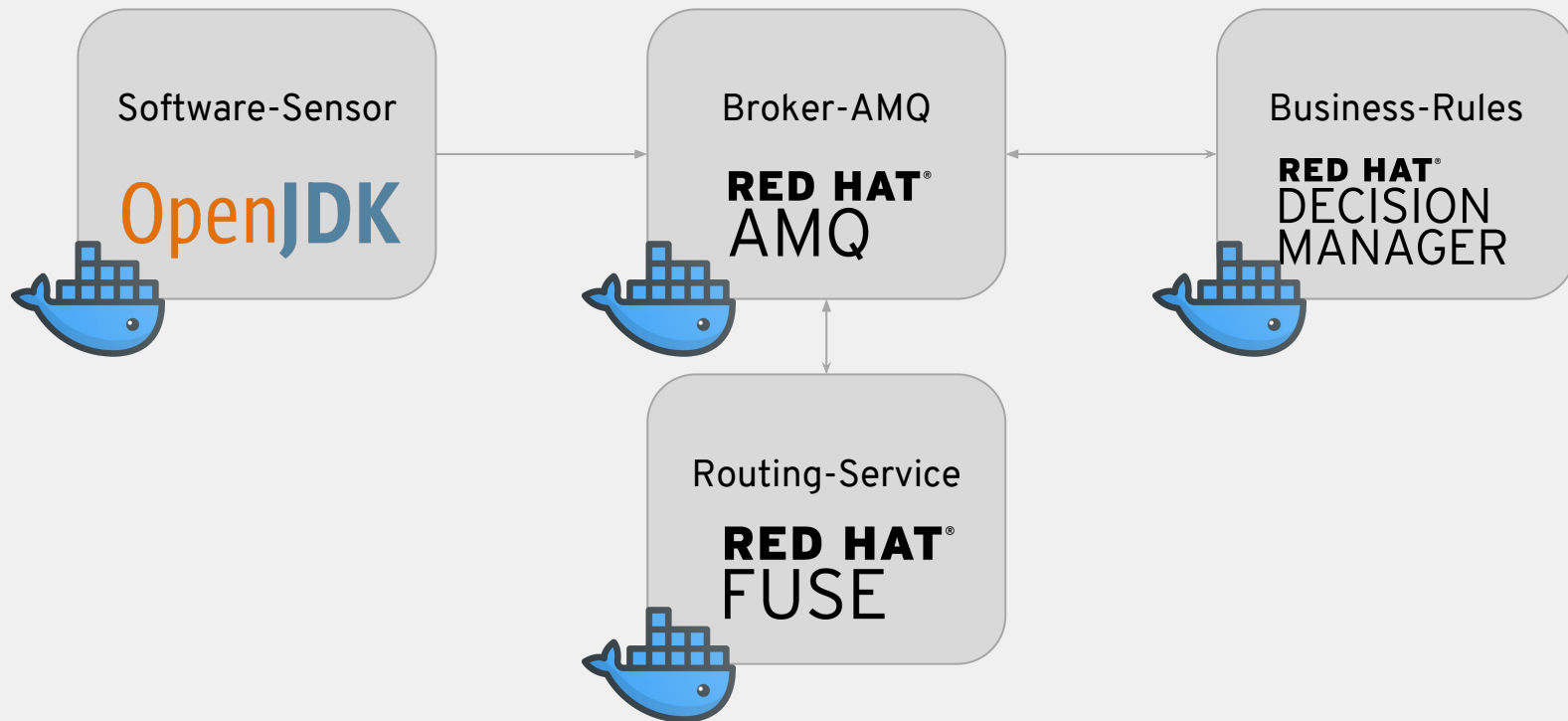
([https://github.com/redhat-iot/Virtual\\_IoT\\_Gateway](https://github.com/redhat-iot/Virtual_IoT_Gateway) )

- ❖ Build the Intelligent IoT Gateway with open source software in a few simple steps
- ❖ Main components of the Gateway are:
  - **Red Hat Enterprise Linux** to provide Enterprise class foundation
  - **Red Hat Fuse** to transform sensor data and route it to end points
  - **Red Hat Decision Manager** to enable real-time decision making at the edge
  - **Red Hat AMQ** to arbitrate sensor data
- ❖ Red Hat Fuse integrate sensor app and a business rules service
- ❖ Sensor app sends temperature data using MQTT to the Red Hat AMQ broker, these messages will be forwarded to the earlier services
- ❖ Finally the business rules will trigger desired action when the sensor value reaches a threshold



# INTELLIGENT IoT GATEWAY: CONTAINERIZED

Demo refactoring for OpenShift deployment



# BUILD THE CONTAINERS

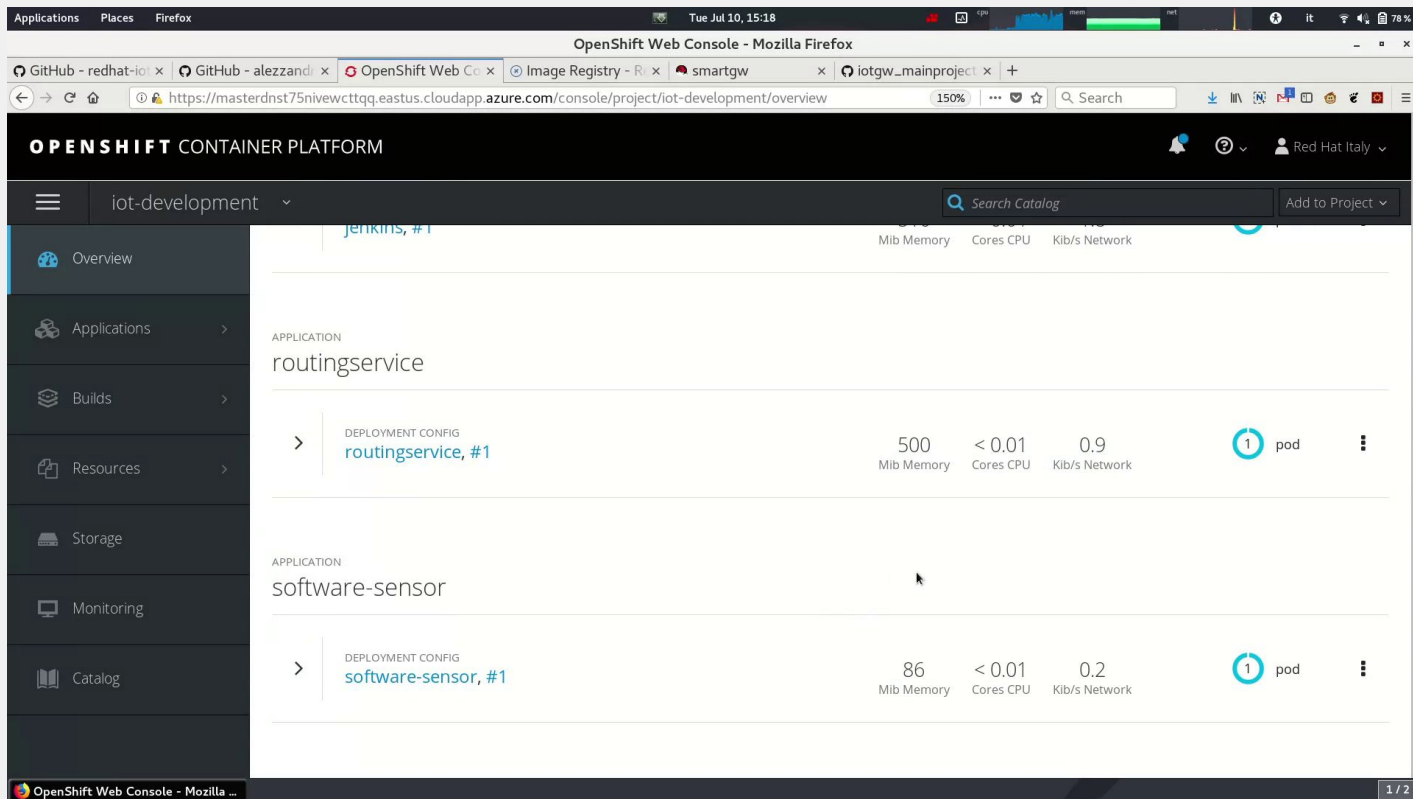
The screenshot displays the OpenShift Container Platform web console interface. The top navigation bar includes 'Applications', 'Places', and 'Firefox'. The browser address bar shows the URL: `https://masterdnst75nivecctqg.eastus.cloudapp.azure.com/console/project/iot-development/browse/pipelines`. The main content area is titled 'OPENSIFT CONTAINER PLATFORM' and shows the 'iot-development' project. A sidebar on the left contains navigation links for Overview, Applications, Builds, Resources, Storage, Monitoring, and Catalog. The main view displays two pipeline runs:

- businessrules-pipeline** (created a month ago):
  - Recent Runs: Build #5 (18 minutes ago)
  - Stages: preamble (3s), build (6s), test (0s), deploy (1s), tag (0s)
  - Average Duration: 1m 7s
- routingservice-pipeline** (created a month ago):
  - Recent Runs: Build #2 (a month ago)
  - Stages: preamble (1s), build (7s), test (0s), deploy (20s), tag (0s)
  - Average Duration: 10h 20m

Each pipeline run includes a 'View Pipeline Runs' and 'Edit Pipeline' link. The bottom of the screenshot shows the browser's address bar with the URL: `https://masterdnst75nivecctqg.eastus.cloudapp.azure.com/console/project/iot-development/edit/pipelines/routingservice-pipeline`.



# DEPLOY THE CONTAINERS

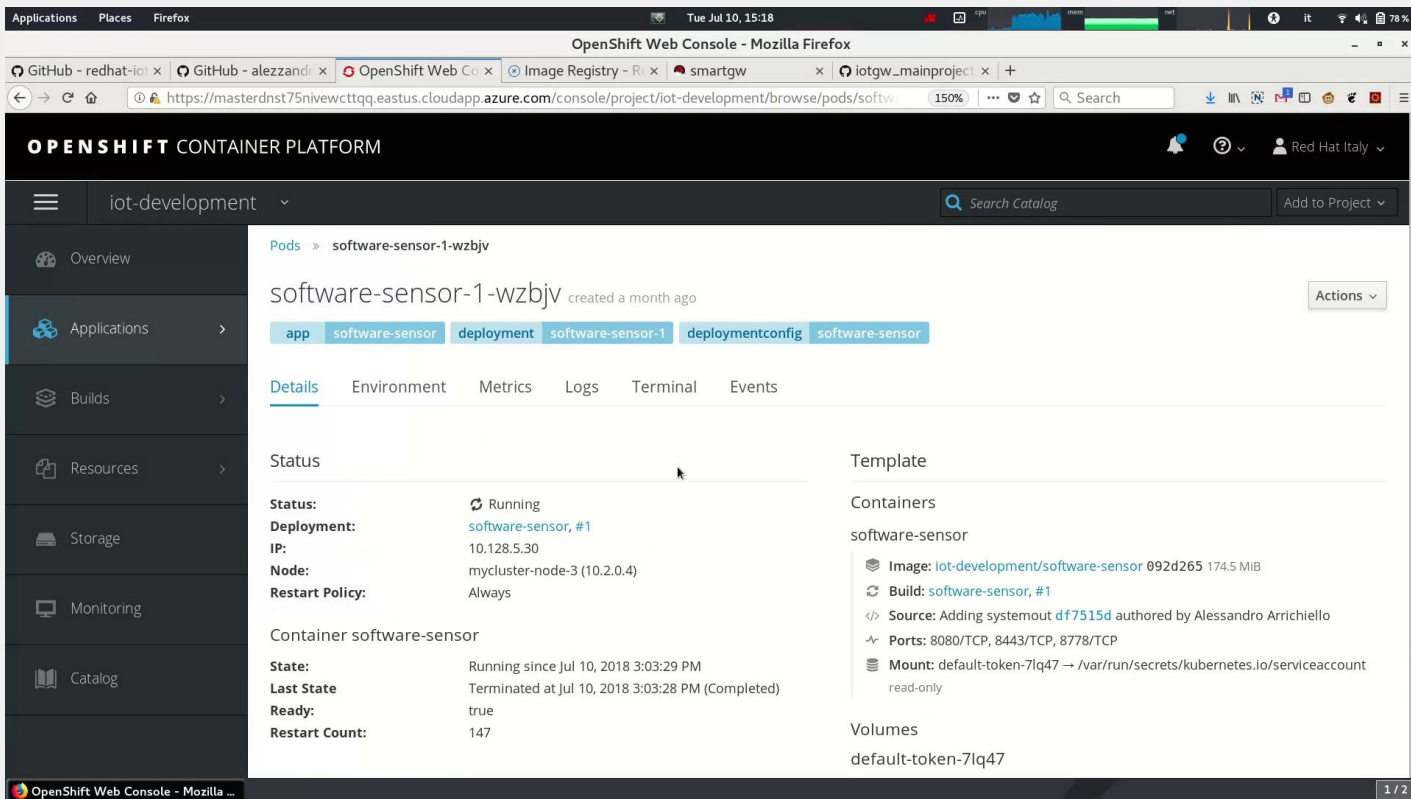


The screenshot displays the OpenShift Web Console interface for the 'iot-development' project. The left sidebar contains navigation options: Overview, Applications, Builds, Resources, Storage, Monitoring, and Catalog. The main content area shows a list of applications with their respective deployment configurations and resource usage.

| Application     | Deployment Config   | Mib Memory | Cores CPU | Kib/s Network | Pod Count |
|-----------------|---------------------|------------|-----------|---------------|-----------|
| jenkins, #1     |                     |            |           |               |           |
| routing-service | routing-service, #1 | 500        | < 0.01    | 0.9           | 1 pod     |
| software-sensor | software-sensor, #1 | 86         | < 0.01    | 0.2           | 1 pod     |



# TEST THE CONTAINERS



The screenshot displays the OpenShift Web Console interface. The main content area shows the details for a pod named 'software-sensor-1-wzbjv', which was created a month ago. The pod is currently in a 'Running' state. The console provides a detailed overview of the pod's configuration, including its deployment, IP address, node, and restart policy. It also shows the container's state, last state, readiness, and restart count. The 'Template' section details the container's image, build, source code, ports, and mount points. The 'Volumes' section lists the default-token-71q47 volume.

**OPENSIFT CONTAINER PLATFORM**

iot-development

Pods » software-sensor-1-wzbjv

software-sensor-1-wzbjv created a month ago

app software-sensor deployment software-sensor-1 deploymentconfig software-sensor

Details Environment Metrics Logs Terminal Events

**Status**

**Status:** Running

**Deployment:** software-sensor, #1

**IP:** 10.128.5.30

**Node:** mycluster-node-3 (10.2.0.4)

**Restart Policy:** Always

**Container software-sensor**

**State:** Running since Jul 10, 2018 3:03:29 PM

**Last State:** Terminated at Jul 10, 2018 3:03:28 PM (Completed)

**Ready:** true

**Restart Count:** 147

**Template**

**Containers**

software-sensor

- Image: iot-development/software-sensor 092d265 174.5 MiB
- Build: software-sensor, #1
- Source: Adding systemout df7515d authored by Alessandro Arrichiello
- Ports: 8080/TCP, 8443/TCP, 8778/TCP
- Mount: default-token-71q47 → /var/run/secrets/kubernetes.io/serviceaccount read-only

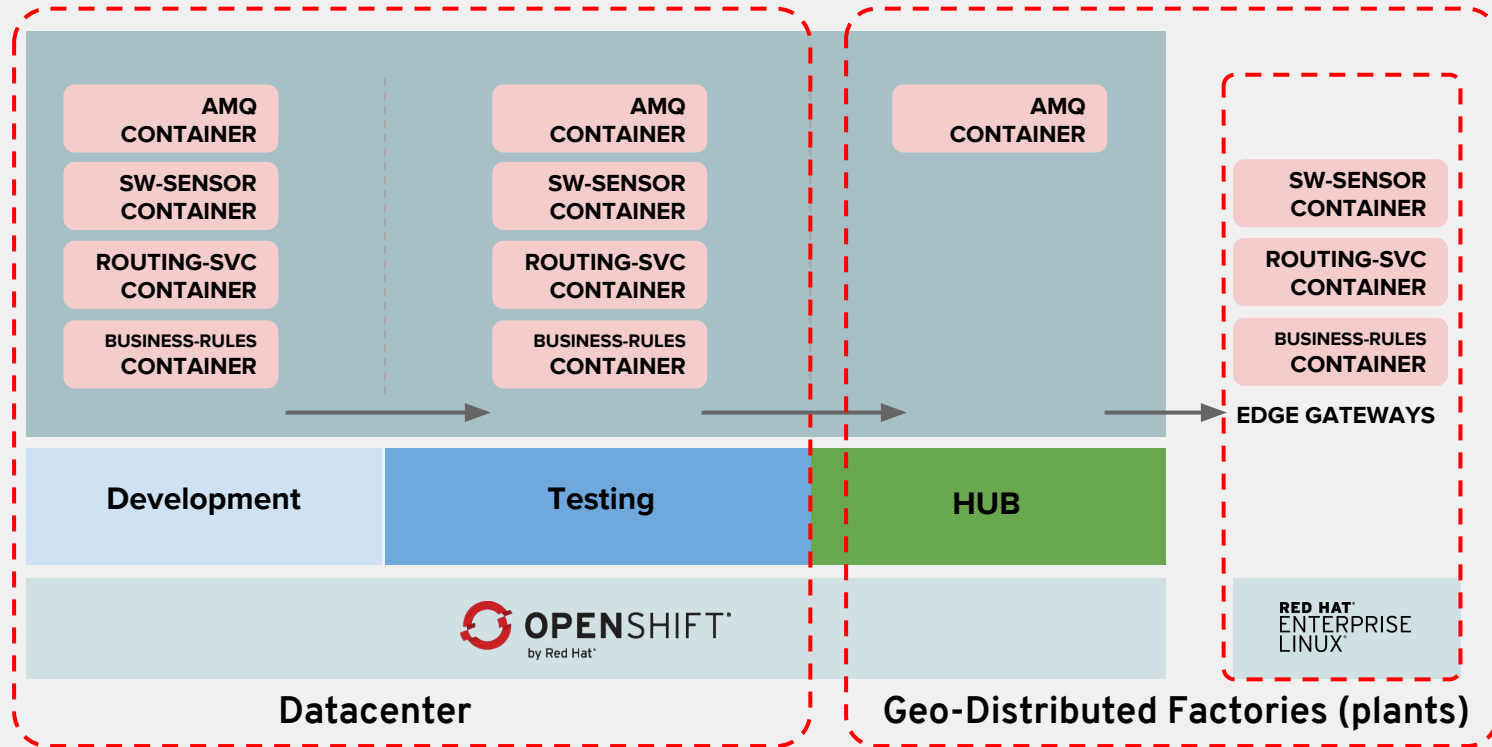
**Volumes**

default-token-71q47



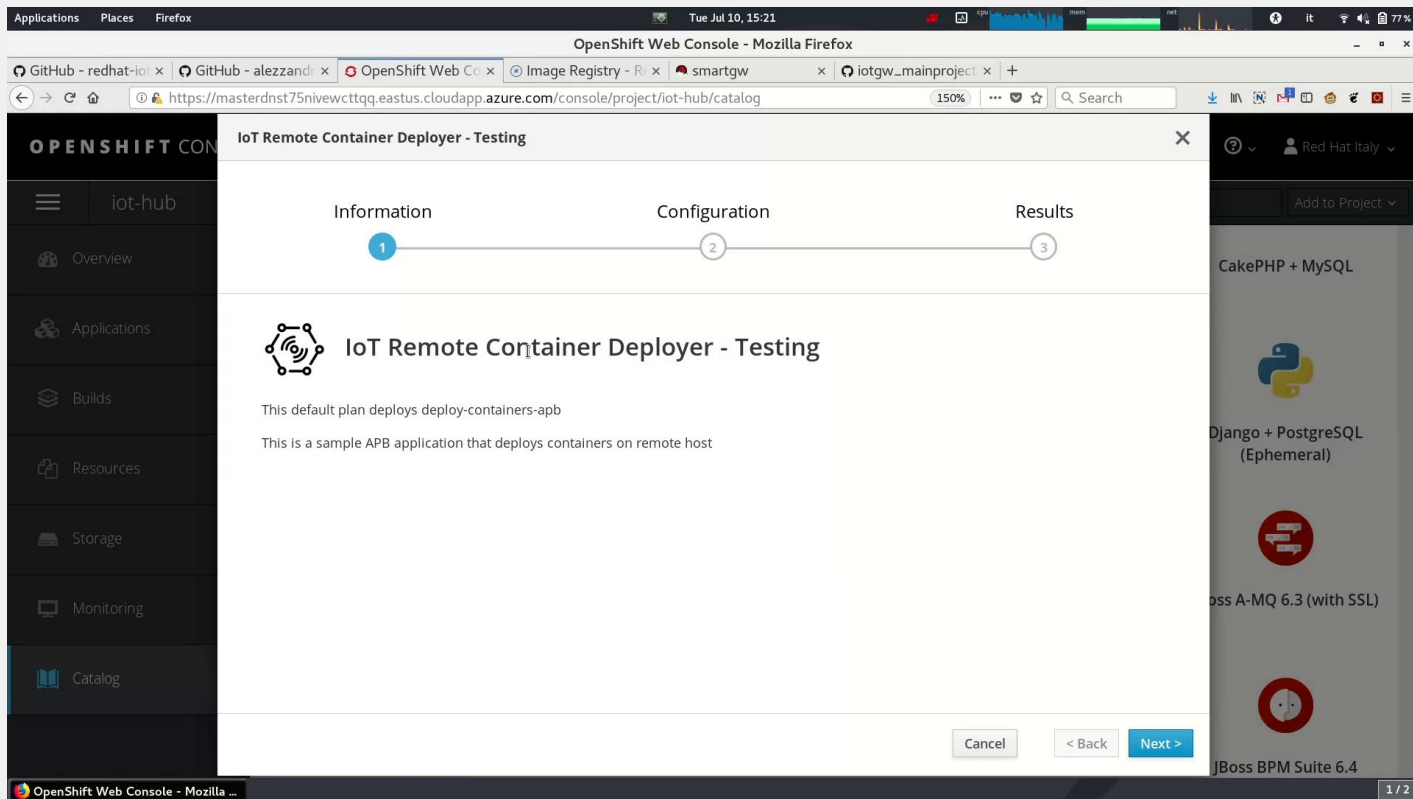
# DEMO's CONTAINERS DEPLOYMENT TOPOLOGY

Multiple Openshift projects simulating DC and HUB





# DEPLOY REMOTE CONTAINERS



Applications Places Firefox Tue Jul 10, 15:21

OpenShift Web Console - Mozilla Firefox

GitHub - redhat-iot x | GitHub - alezzandi x | OpenShift Web Co x | Image Registry - Ri x | smartgw x | iotgw\_mainproject x | +

https://masterdnst75nivewcttq.eastus.cloudapp.azure.com/console/project/iot-hub/catalog

150% Search

OPENSIFT CON IoT Remote Container Deployer - Testing

iot-hub

Overview

Applications

Builds

Resources


Storage

Monitoring

Catalog

Information Configuration Results

1 2 3

 IoT Remote Container Deployer - Testing

This default plan deploys deploy-containers-apb

This is a sample APB application that deploys containers on remote host

Cancel < Back Next >

OpenShift Web Console - Mozilla ... 1 / 2

Red Hat Italy

Add to Project

CakePHP + MySQL

Django + PostgreSQL (Ephemeral)

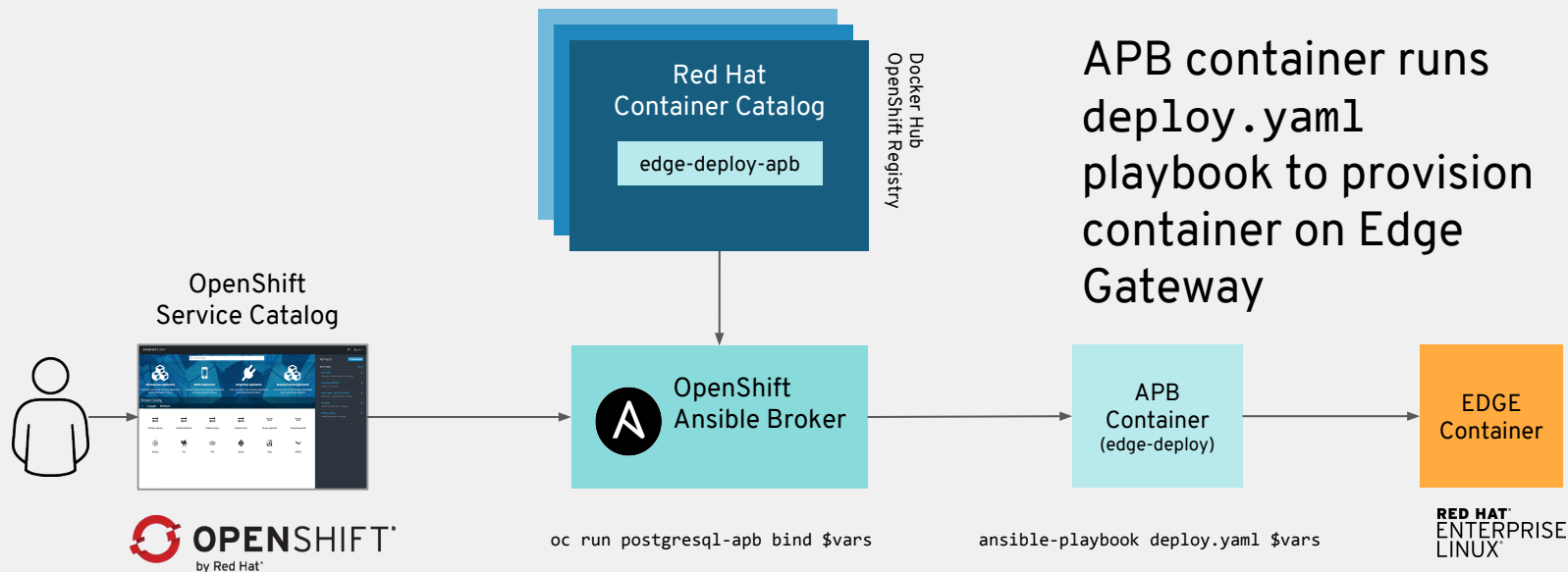
JBoss A-MQ 6.3 (with SSL)

JBoss BPM Suite 6.4

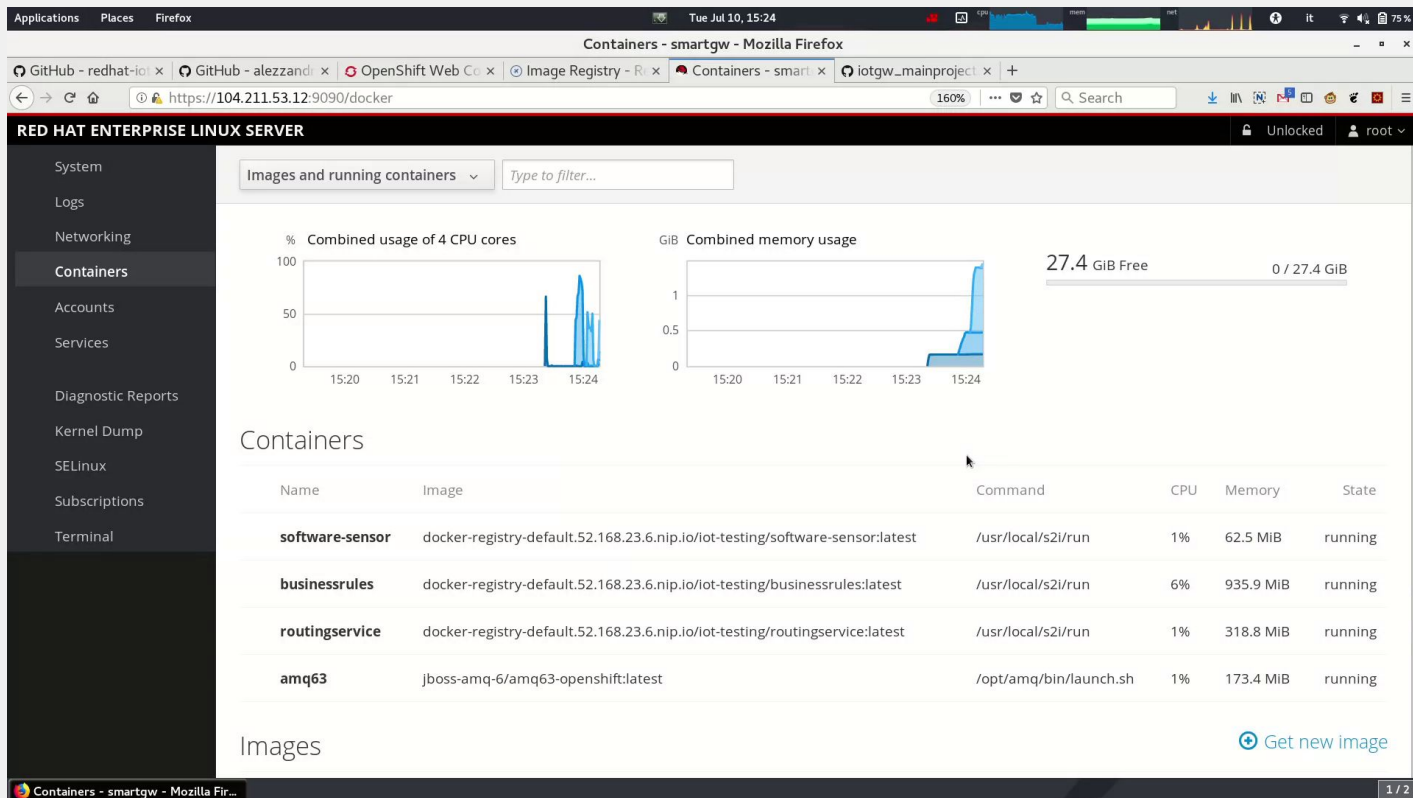


# OPENSIFT ANSIBLE PLAYBOOK BUNDLE

Deploy the container just built through OpenShift



# COCKPIT: RHEL MANAGEMENT



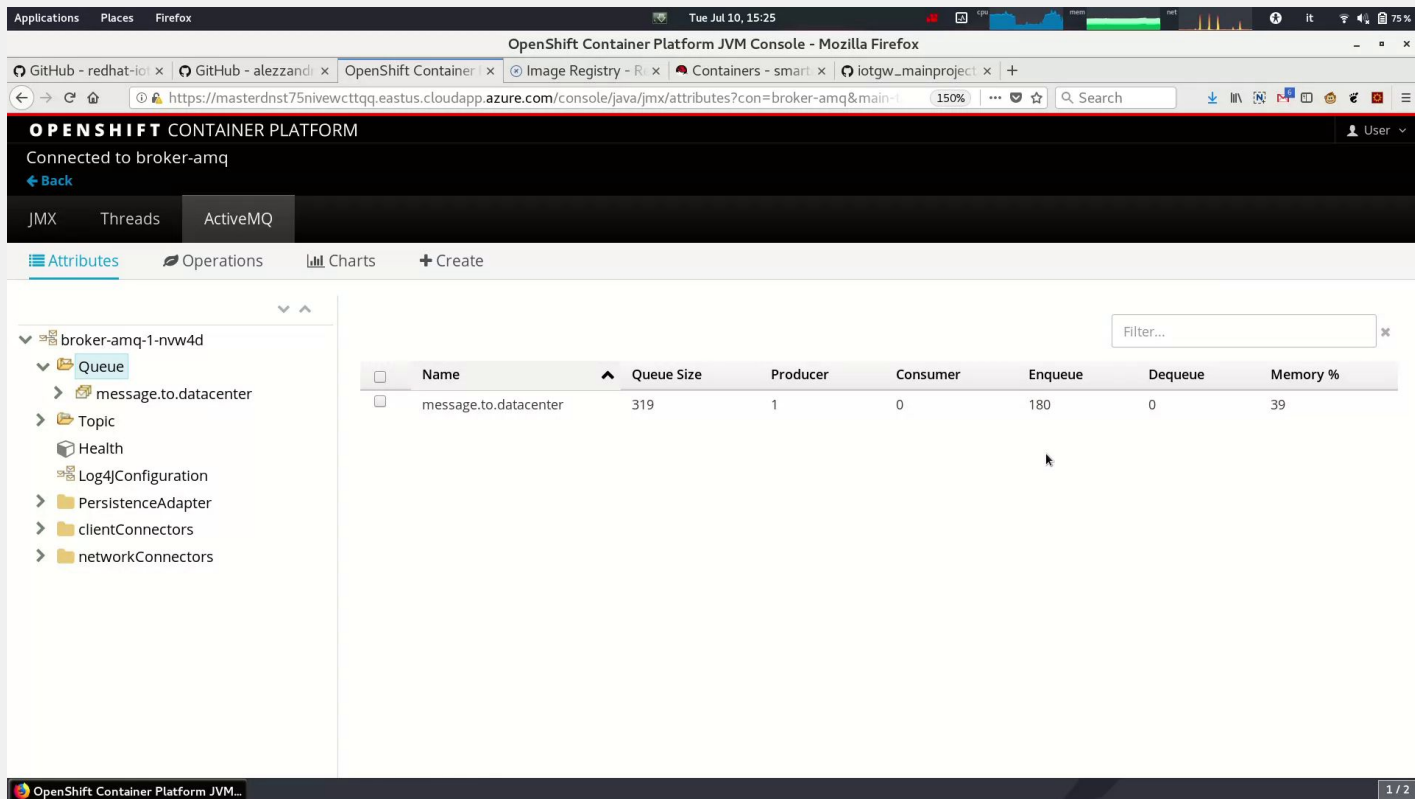
The screenshot displays the Cockpit web interface for a Red Hat Enterprise Linux server. The top navigation bar includes 'System', 'Logs', 'Networking', 'Containers', 'Accounts', 'Services', 'Diagnostic Reports', 'Kernel Dump', 'SELinux', 'Subscriptions', and 'Terminal'. The 'Containers' section is active, showing 'Images and running containers' with a search filter. Two line graphs are visible: '% Combined usage of 4 CPU cores' and 'GiB Combined memory usage'. The memory usage graph shows a bar chart with a value of 27.4 GiB Free and 0 / 27.4 GiB used. Below the graphs is a table of running containers:

| Name            | Image                                                                         | Command                | CPU | Memory    | State   |
|-----------------|-------------------------------------------------------------------------------|------------------------|-----|-----------|---------|
| software-sensor | docker-registry-default.52.168.23.6.nip.io/iot-testing/software-sensor:latest | /usr/local/s2i/run     | 1%  | 62.5 MIB  | running |
| businessrules   | docker-registry-default.52.168.23.6.nip.io/iot-testing/businessrules:latest   | /usr/local/s2i/run     | 6%  | 935.9 MIB | running |
| routingservice  | docker-registry-default.52.168.23.6.nip.io/iot-testing/routingservice:latest  | /usr/local/s2i/run     | 1%  | 318.8 MIB | running |
| amq63           | jboss-amq-6/amq63-openshift:latest                                            | /opt/amq/bin/launch.sh | 1%  | 173.4 MIB | running |

At the bottom, there is a section for 'Images' with a '+ Get new image' button.



# DATA FLOWS BACK TO THE HUB



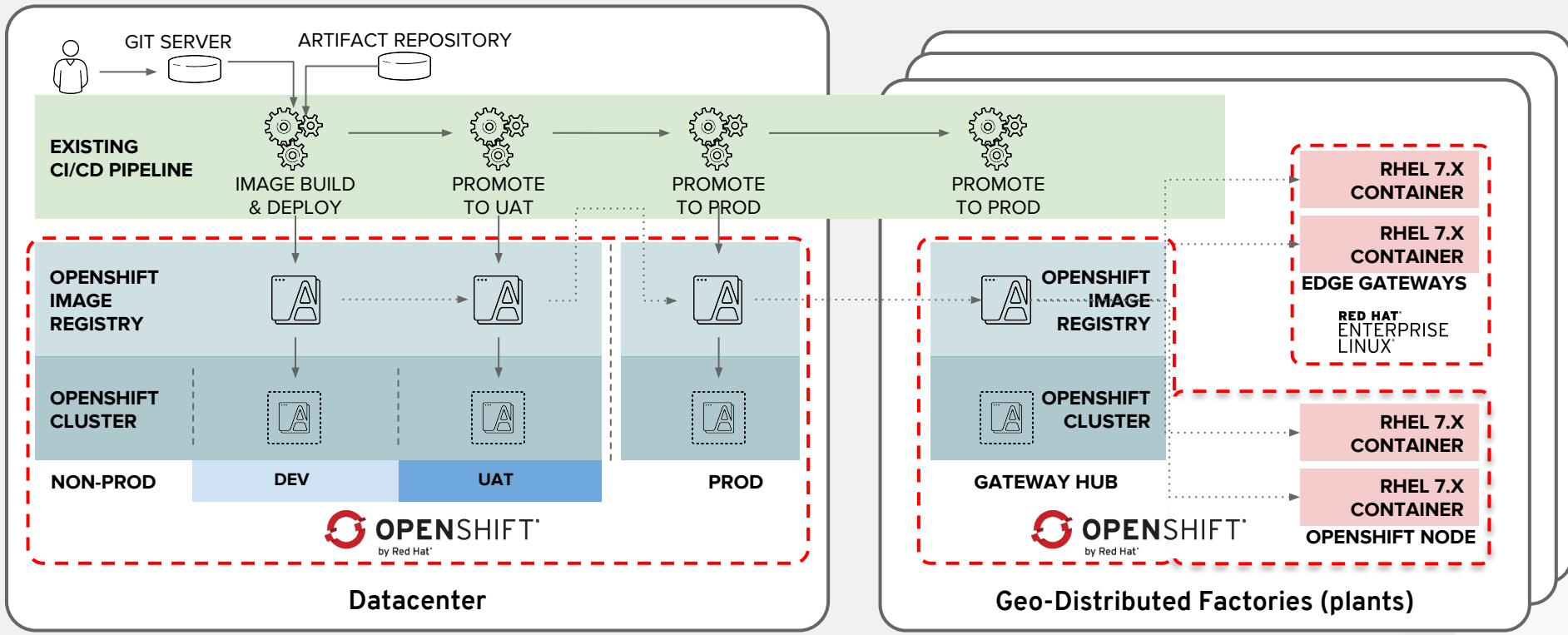
The screenshot displays the OpenShift Container Platform JVM Console in a Mozilla Firefox browser. The console is connected to a broker-amq and shows the ActiveMQ tab selected. The left sidebar lists the broker's components, with the 'Queue' section expanded to show a 'message.to.datacenter' queue. The main panel displays a table of queue statistics:

| Name                  | Queue Size | Producer | Consumer | Enqueue | Dequeue | Memory % |
|-----------------------|------------|----------|----------|---------|---------|----------|
| message.to.datacenter | 319        | 1        | 0        | 180     | 0       | 39       |



# CI/CD THROUGH DATACENTERS & GATEWAYS

[https://github.com/alezzandro/iotgw\\_mainproject](https://github.com/alezzandro/iotgw_mainproject)



# MULTIPLE EDGE DEPLOYMENTS SCENARIOS

## Corporate Node



## Plant Node

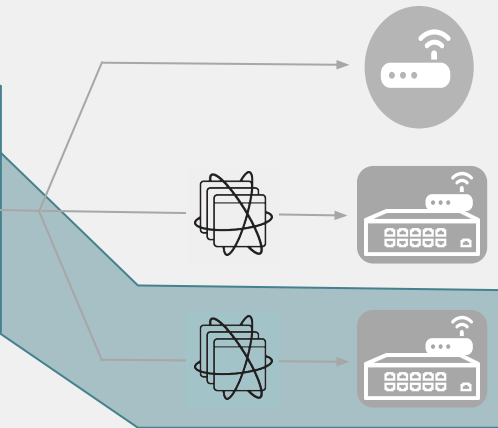
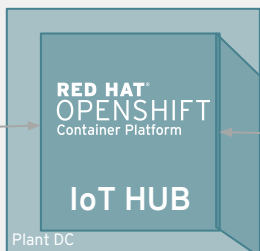


## Edge Node



**Deployment Scenarios**  
(Based on resource (CPU/Memory)  
and connectivity (Bandwidth/Latency)  
availability)

**Available capabilities**



### SCENARIO 1

Low resource (Edge Gateway)  
Non reliable connectivity

- Data gathering
- Basic analytics remotely managed

### SCENARIO 2

High resource (Edge Server)  
Reliable connectivity

- Data gathering
- Dynamic deployed containerized business applications

### SCENARIO 3

Available resource (Edge Server)  
High affidability connectivity

- Data gathering
- Dynamic deployed containerized business applications
- Centralized Management





GRAZIE PER L'ATTENZIONE



#RedHatOSD