Taming the Beast

Modernizing with containers and cloud native technologies

Markus Eisele

Developer Adoption Lead EMEA

Red Hat

Markus Eisele

Developer Adoption Lead EMEA
15 years developer and architect with Enterprise Java (Automotive, Finance, Insurance)
6 years Developer Relations
150+ presentations, 200+ articles

twitter.com/myfear
https://www.linkedin.com/in/markuseisele/

Java Champions

JavaLand

Red Hat

I am going to talk about..

- Why are we here? What are the challenges?
- Why Buzzwords won’t solve your problems.
- Where to even start? Migration challenges.
- Define Success! Begin with the End in Mind.
- It’s not (only) about technology!
- Where to go from here?

Creating value depends on your ability to deliver applications faster

Cloud-native applications

AI & machine learning

Analytics

Internet of Things

Innovation culture

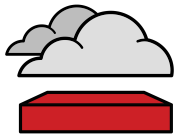
Red Hat

Shifting investment to innovation

It's about efficiency, agility, & speed

IT optimization

Gain greater efficiency while building a cloud-ready foundation

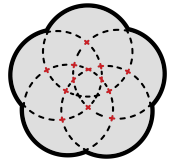


Hybrid cloud infrastructure

Enable data and application portability across cloud platforms

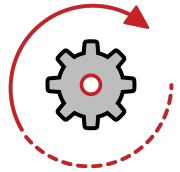
Cloud-native development

Quickly build and run scalable applications in dynamic environments



Agile integration

Integrate applications and data to identify and act on opportunities



Automation

Reduce costs, complexity, and errors deploying infrastructure and applications

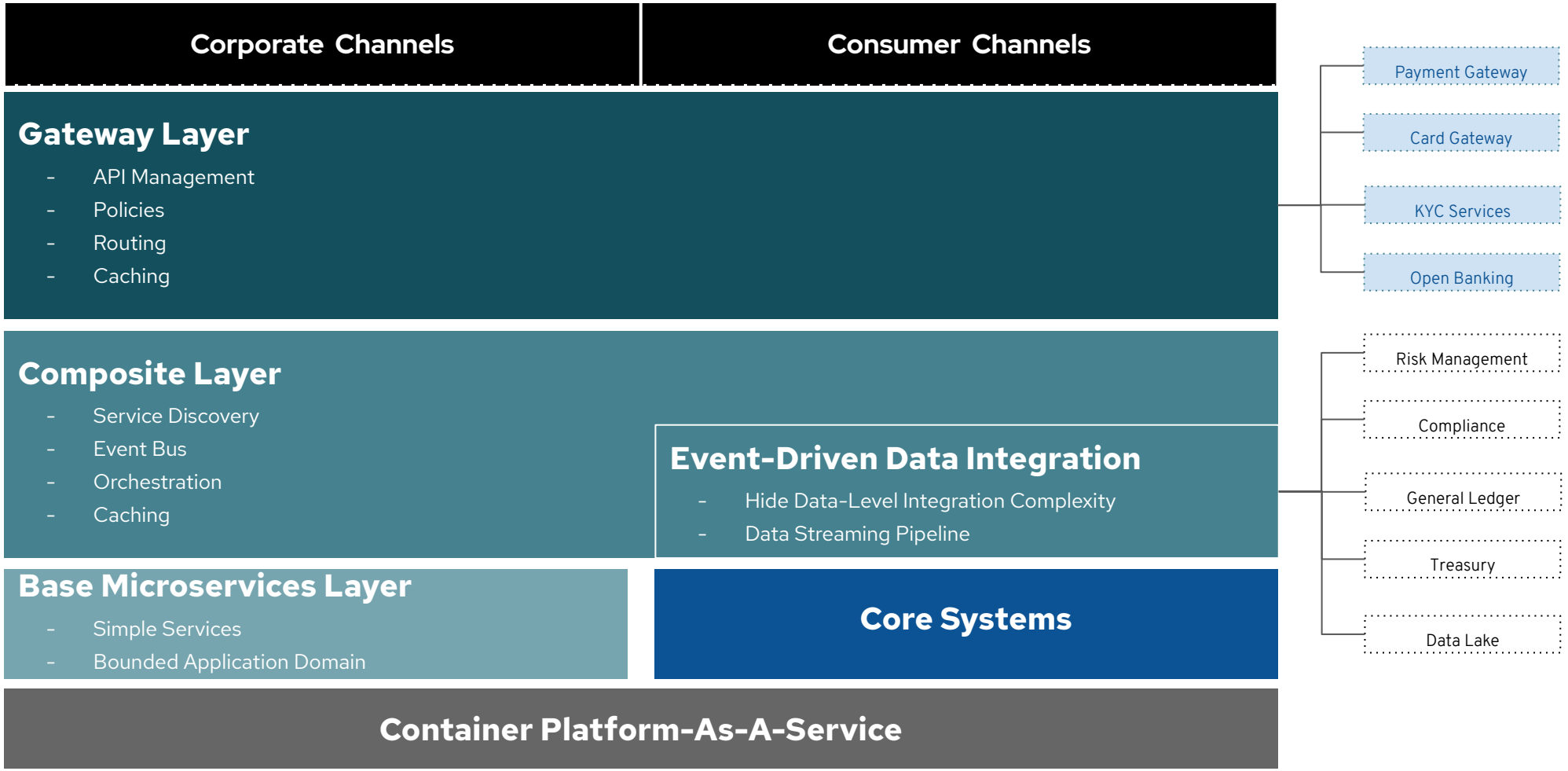
Red Hat

Careful about Buzzwords

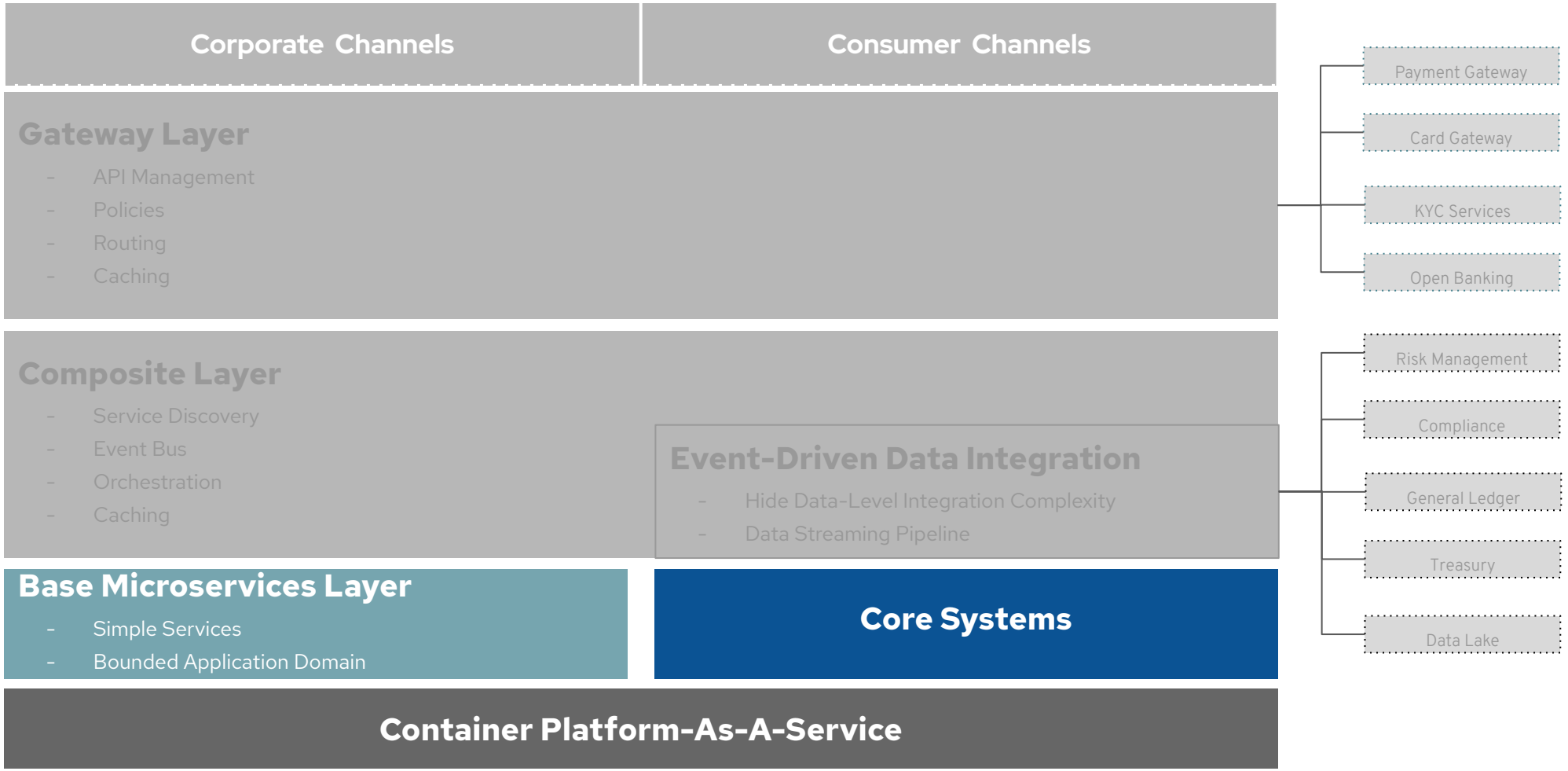
Or: Only using Microservices doesn’t solve your problems.

Red Hat

There is sooo much more that is needed!



And I don't have time for all of this today...



Red Hat

High Level Approach to define your ideal Platform

Define the Core

- Existing Core Capabilities
- Functional Gap Assessment
- Target Deployment Model
- Core Modernization Approach

Define the Customer Experience Layer

- Define Customer Centric requirements
- Assess Existing CX Framework vs. Build
- Integration Requirements
- Data Gap Analysis

Define the Integration

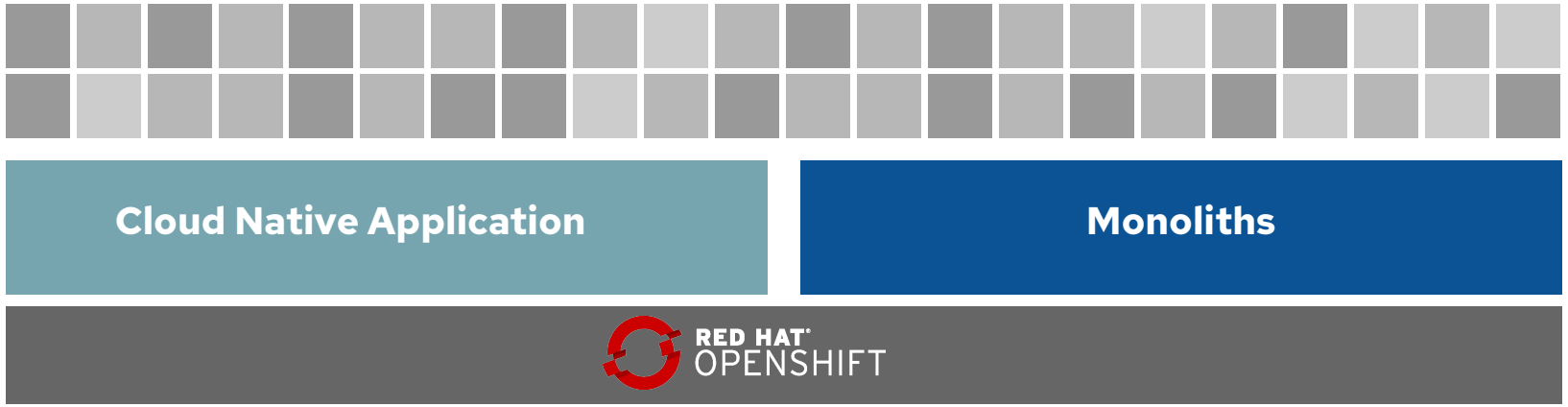
- Existing Integration Capabilities
- Define Partner Solution Ecosystem
- Define Integration requirements (Data Sources, Service Integration, Messaging, APIs)

Define the Technology Stack

- Technology Stack Assessment across Core, CX, and external services
- Deployment model (IaaS, PaaS, Hybrid)
- Define Target Development Platform
- Development skills Gap Analysis

Let's just assume it's OpenShift and we want to...

- Moving existing apps to containers
- Creating modern applications
- The developer lifecycle around it.



Red Hat

Where to even start? Migration challenges.

REHOST

Containerize existing workloads

Deploy them on a PaaS

Keep external integrations and data on legacy

Legacy applications have to be well written and suited

REPLATFORM

Similar to Rehost

Augment with new layers - new capabilities

Deploy on PaaS

New integration points between legacy and new layers

REFACTOR

Legacy is totally replaced

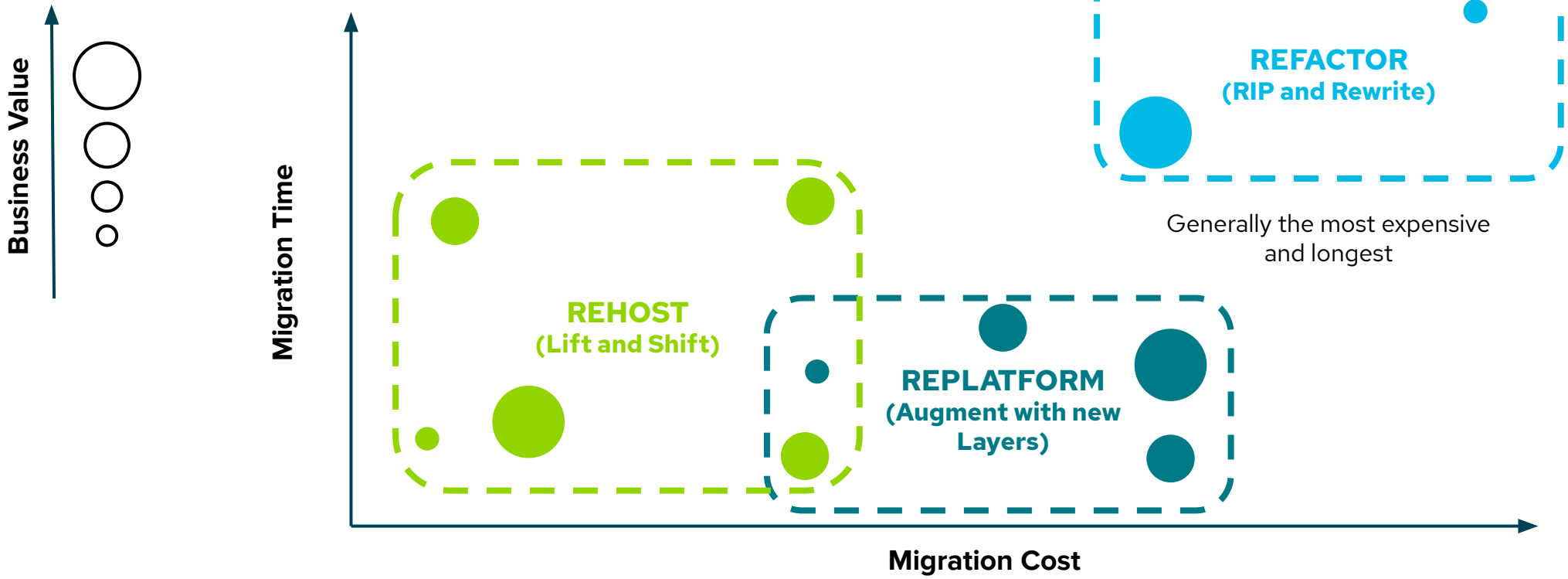
New interfaces and data

Use PaaS to run

Some data and features can be re-wrapped, but mostly are retired.

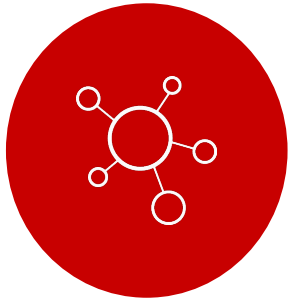
Workload Migration Patterns

No single best pattern

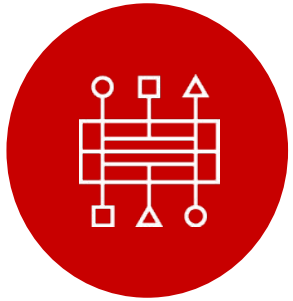


Red Hat

WHAT are Cloud Native Applications?

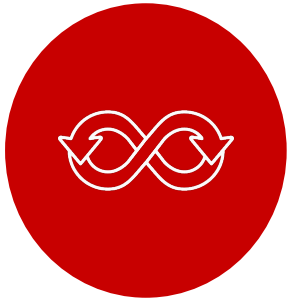


SERVICE-BASED ARCHITECTURE



API

CONTAINERS



DEVOPS

Cloud-Native architecture is NOT magic pixie dust

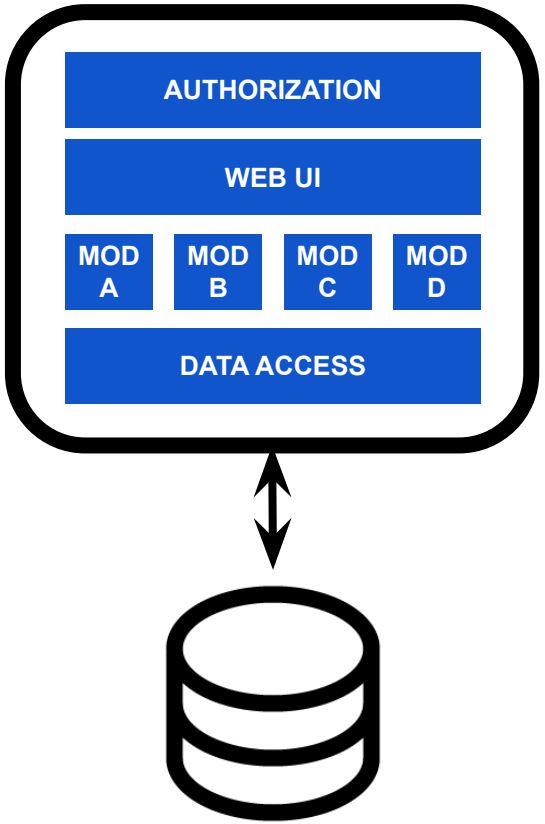
Adopting microservices won't address:

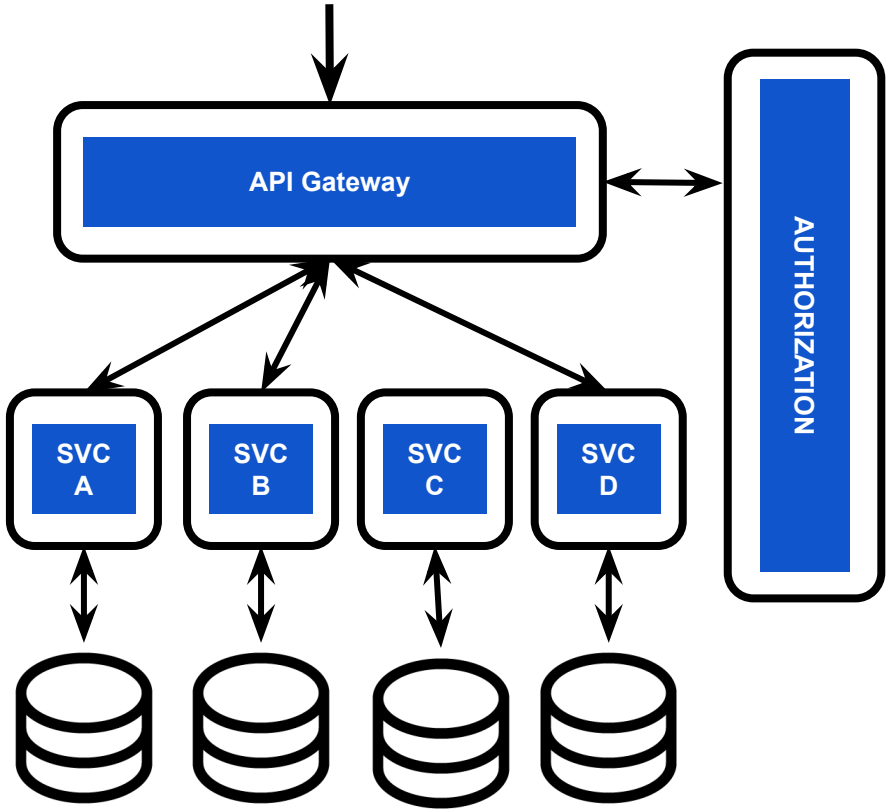
- Poor code quality
- Lack of automated testing
- Poor development process
- ....

And might make things worse!

Do it incrementally

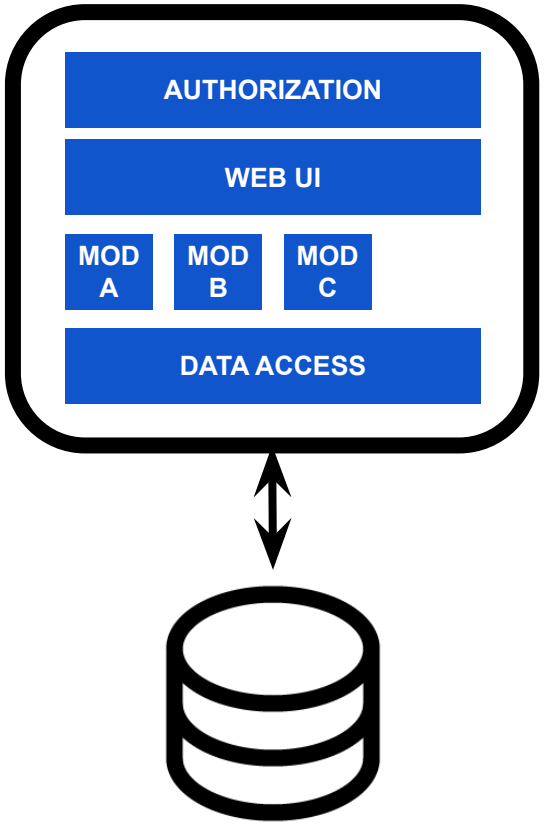
Incrementally migrate functionality from existing application to new (strangler) application

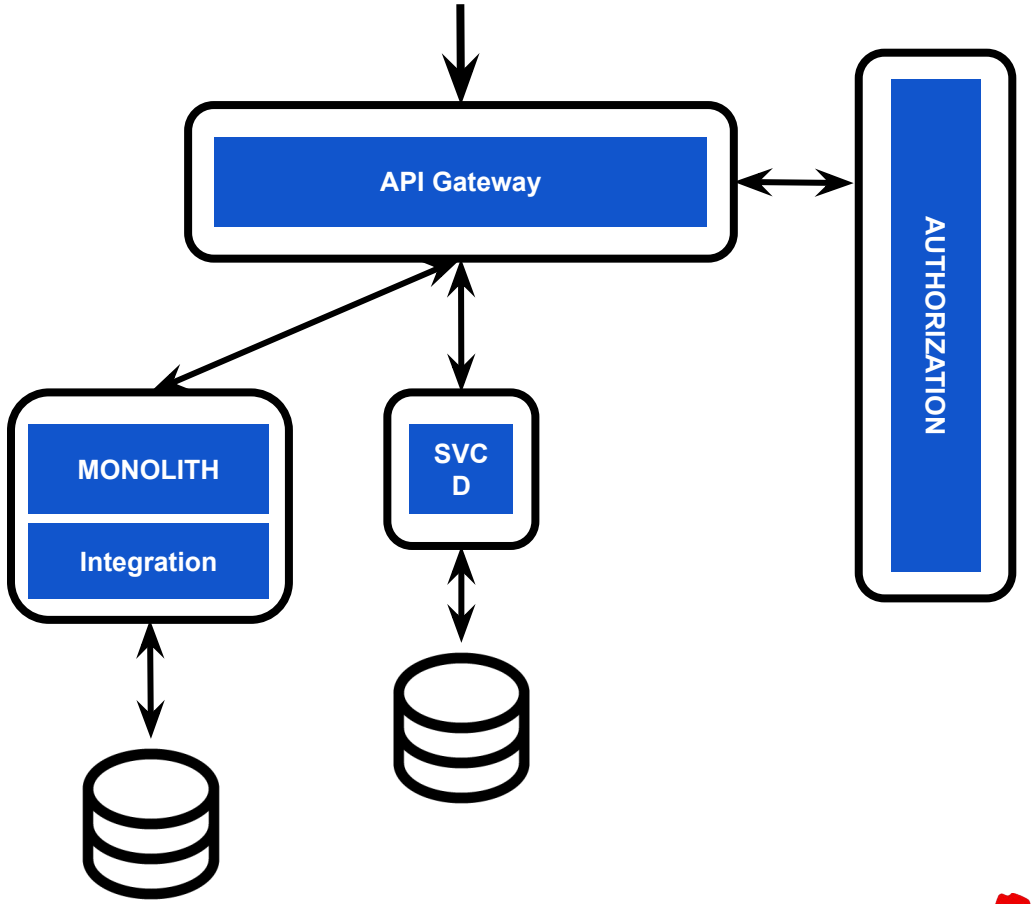




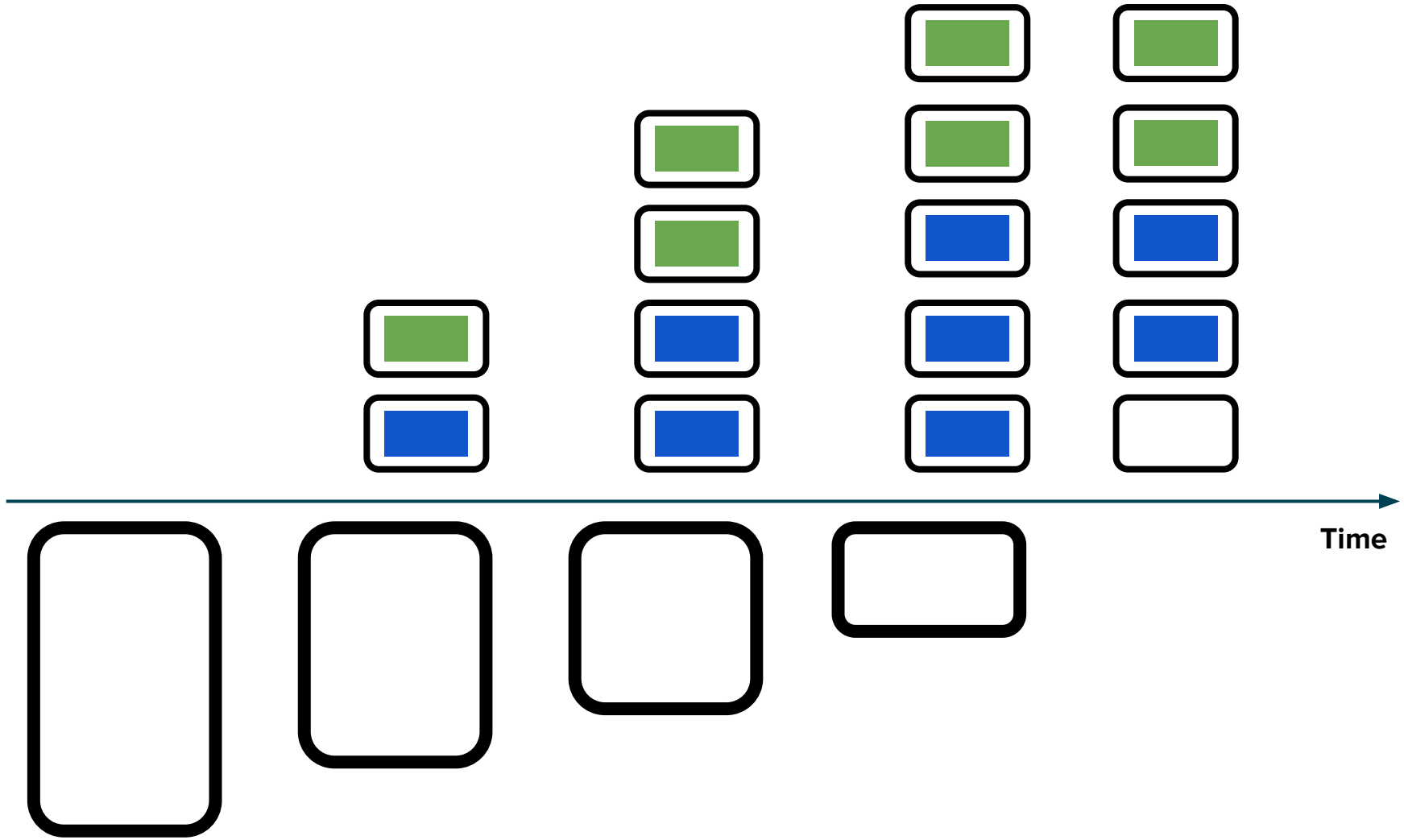
Let new features become new services

Instead of new modules



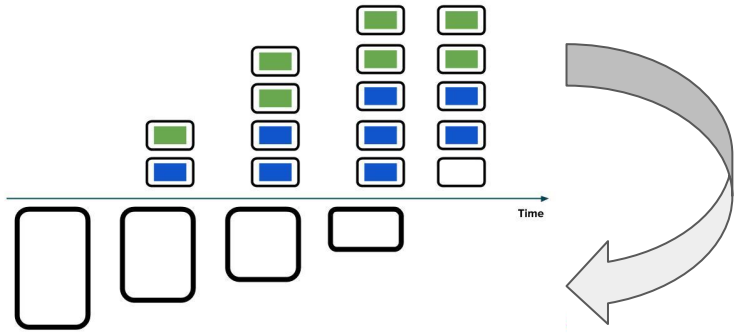


Let the monolith shrink over time



And go on until...

- **The monolith is eliminated**
- **Solved software delivery problems**
- **Higher priority work**



Red Hat

Cost vs. Benefit

Benefit

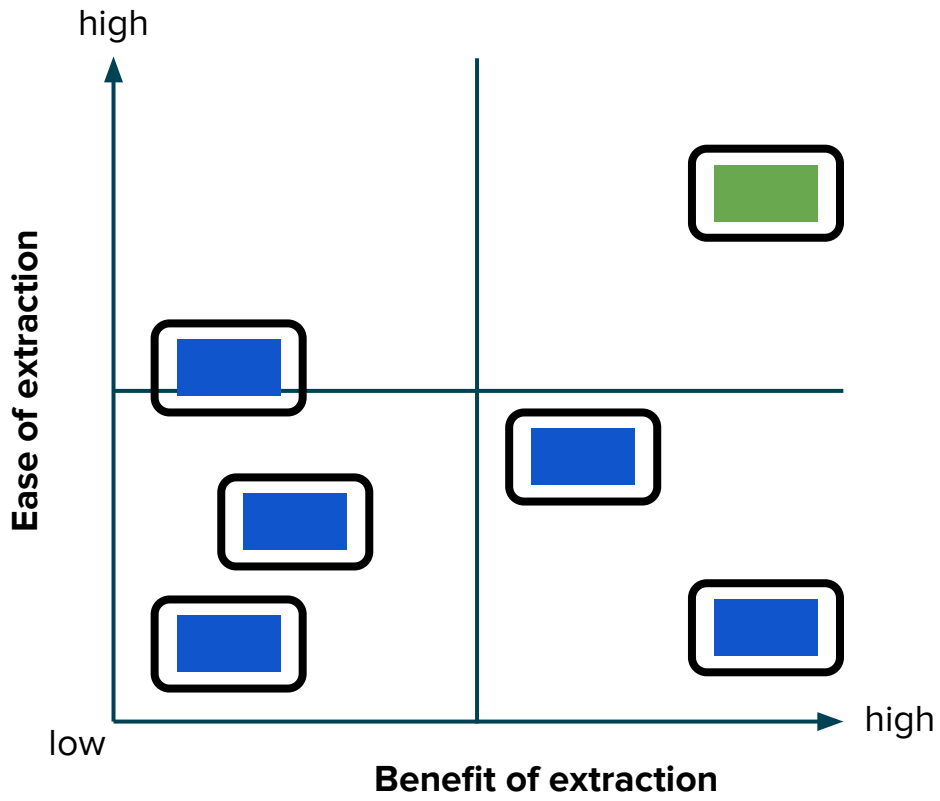
- Solves a significant problem
- Velocity ⇒ frequently updated
- Scalability ⇒ Conflicting resource requirements
- ...

Cost

- Changing the monolith and adapting/rewriting module
- Difficulty in decoupling/ breaking dependencies
- Need to participate in sagas/compensating transactions
- Decoupling inbound dependencies
- ...

Cost vs. Benefit

Find the right candidates first



Red Hat

Software Design Implications

Architecture Principles

Single Responsibility Principle

Service Oriented Architecture

Encapsulation

Separation of Concern

Loose Coupling

Hexagonal Architecture

Design Patterns

Domain-driven Design

Bounded Contexts

Event Sourcing

CQRS

Eventual Consistency

Context Maps

Best Practices

Design for Automation

Designed for failure

Service load balancing and automatic scaling

Design for Data Separation

Design for Integrity

Design for Performance

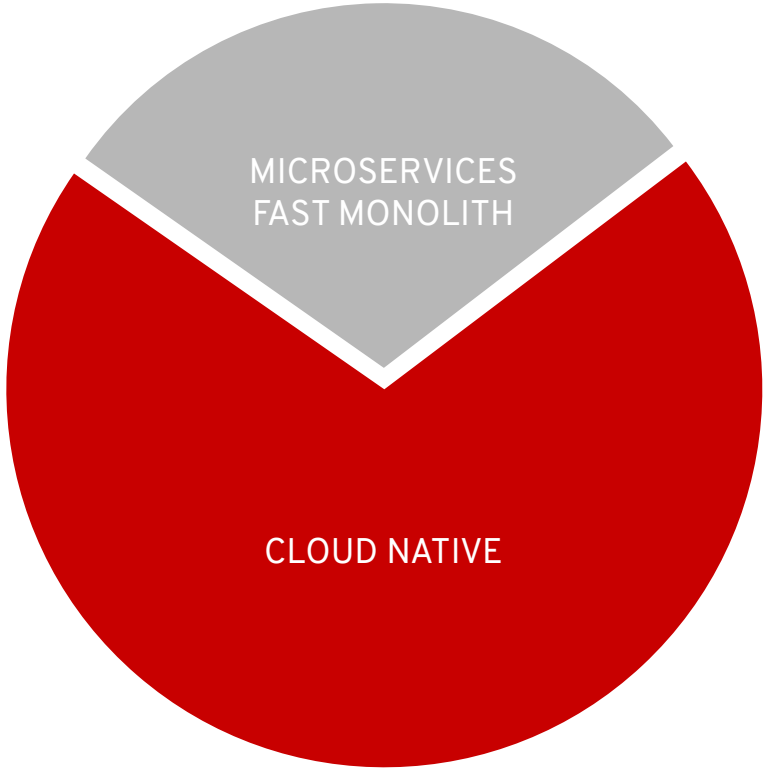
(Micro) Service Approach

Focus on reducing time to value

- Small and simple
- API Focused
- Smaller and faster to test
- Fast start-up time
- Deployed independently
- Design for failure
- Foster new technologies adoption

Define Success! Begin with the End in Mind.

- Success != Number of Microservices
- Improved metrics:
  - Reduced lead time
  - Increased deployment frequency
  - Reduced changed failure rate
- ...



Cloud Native is MUCH MORE

than your application

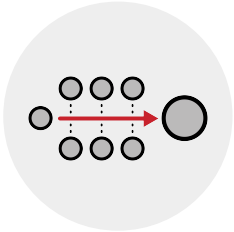
architecture

Red Hat

Evolving Legacy the Red Hat Way.

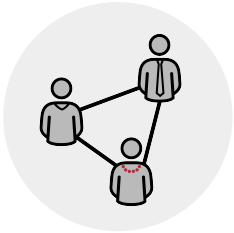
Platform

- Treat integration and process automation like code.
- Implement CI/CD pipelines for faster releases and A/B testing.
- Modern, cloud-native application development requires more agile integration. Use container-based, distributed architectures to deploy business logic, integration, data stream processing across environments.



Process

- Move integration and process automation to agile development teams that cross business and technical users.
- Define overall design goals and guidelines to decentralize centers of excellence.
- Define and measure metrics around dependability, speed, flexibility and cost.



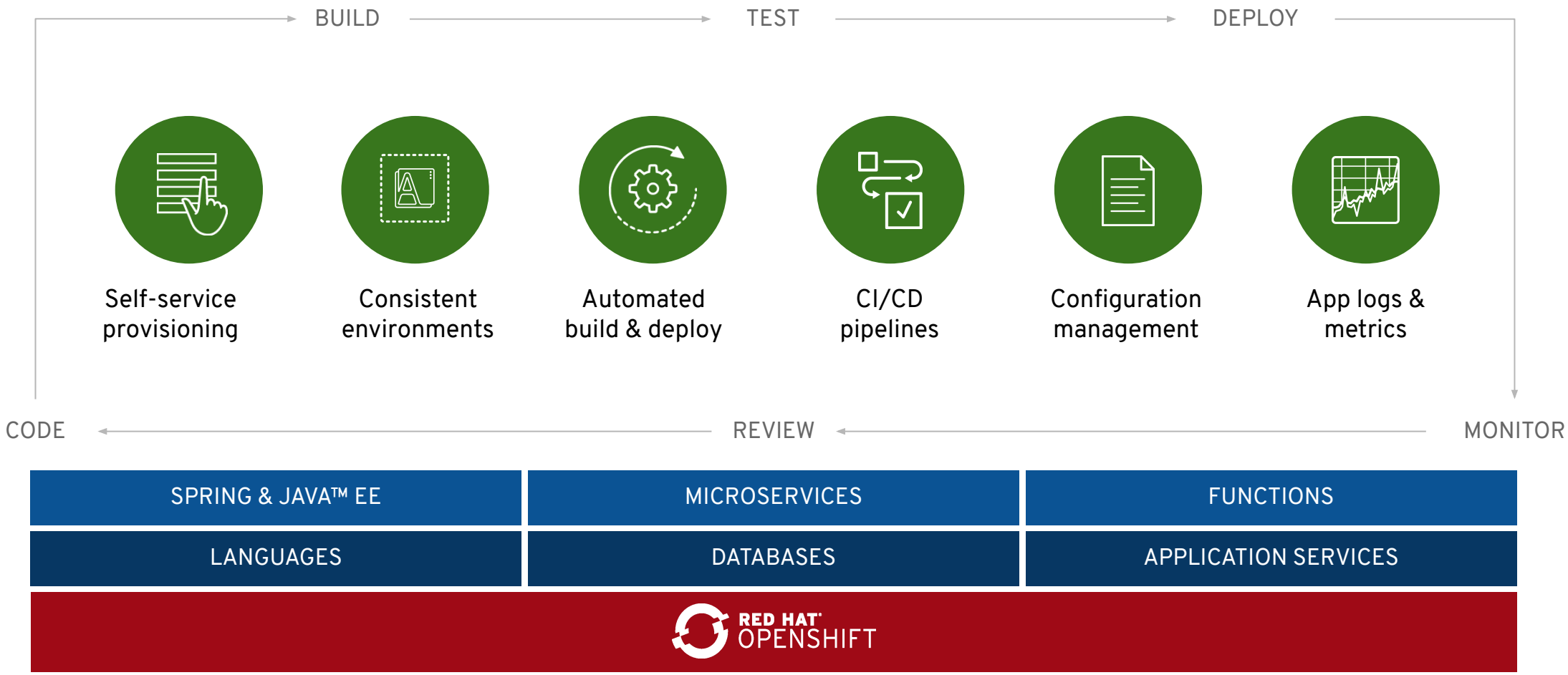
People

- Align integration or decision logic with strategic business goals.
- Define an integration or data strategy.
- Communicate goals and cultivate buy-in for the process and outcomes.

Communication, Collaboration and Openness – People, Process and Technology

- It's common sense, but well worth investing in good practices
- Little steps
- Information is shared and publicized
- Self organizing teams for agility
- Development is paired
- Regular feedback is essential
- Involve all the stakeholders in the project - no silos
- Develop for CI
- Build for production
- Red Hat are well placed to help with DevOps culture

Full Application Lifecycle

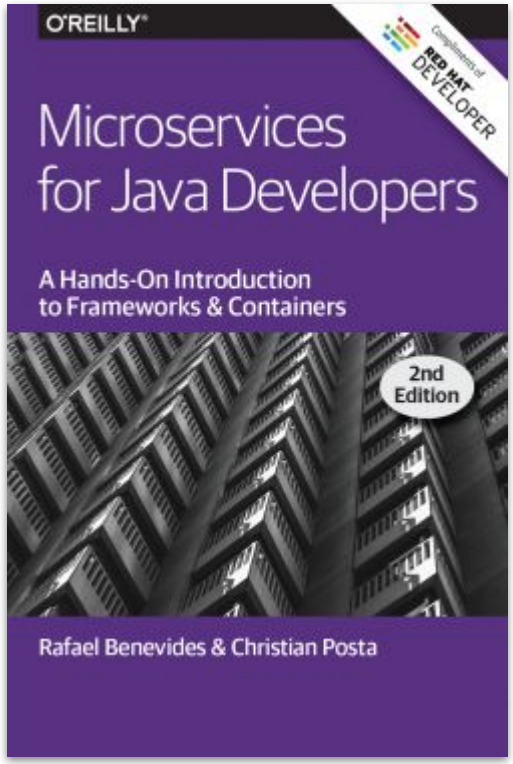


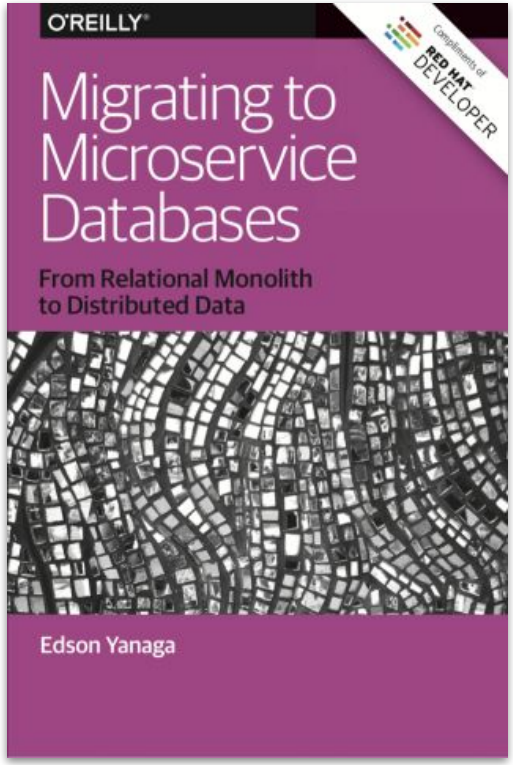
Red Hat

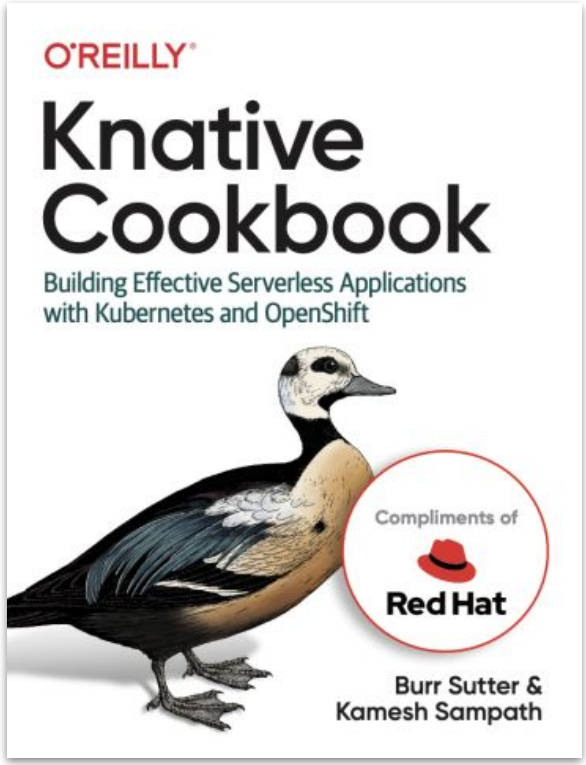
Red Hat Developer

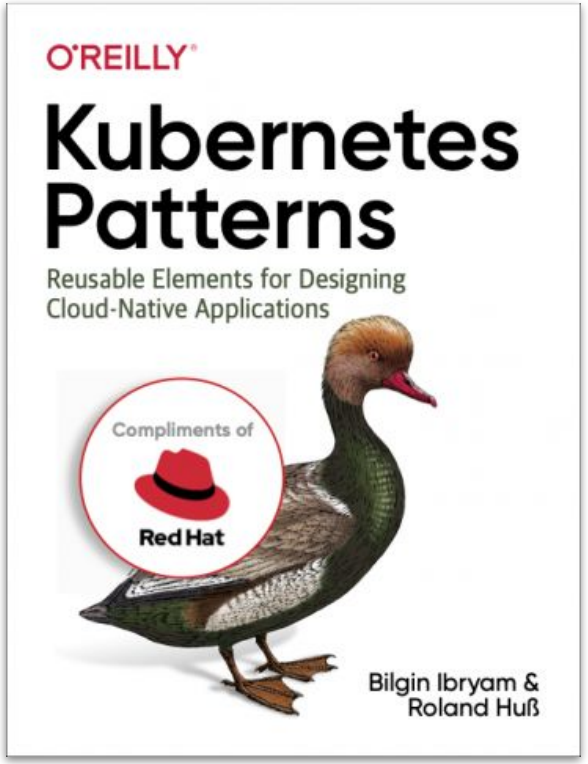
https://developers.redhat.com

Further Reading









https://developers.redhat.com/ebooks

Red Hat

Red Hat

A monthly webinar series for developers.

#OpenShift #Microservices #Knative #Quarkus #Kafka #Cloudnative #Container

OpenDevHour

with Markus & Guests

Sharing best practices in designing and building modern applications.

Join me and guests - no subscription, no signup!

https://red.ht/OpenDevHour

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat