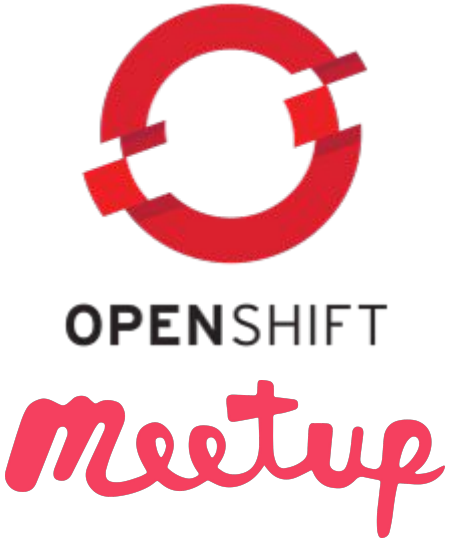


Application Modernization and Migration

Journey to the Cloud

oc whoami

```
$ oc whoami
rmarting, jromanmartin
$ oc describe user rmarting jromanmartin
Name: Jose Roman Martin Gil
Created: 41 years ago
Labels: father, husband, friend, runner, curious, red hatter,
developer (in any order)
Annotations: Senior Middleware Architect @ Red Hat
Identities:
     mailto:   rmarting@redhat.com
     GitHub:   https://github.com/rmarting
     Twitter:  https://twitter.com/jromanmartin
     LinkedIn: https://www.linkedin.com/in/jromanmartin/
```

APPLICATION MODERNIZATION & MIGRATION

DevOps !?

Devs

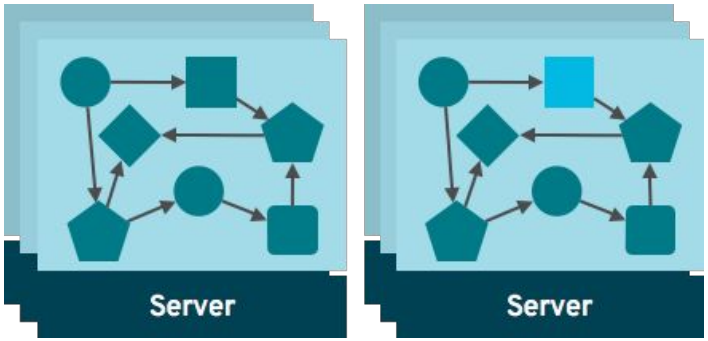
Ops

Application Designs

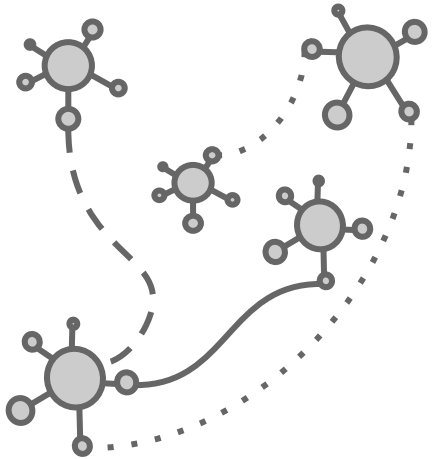
Monoliths

Microservices

Application Designs



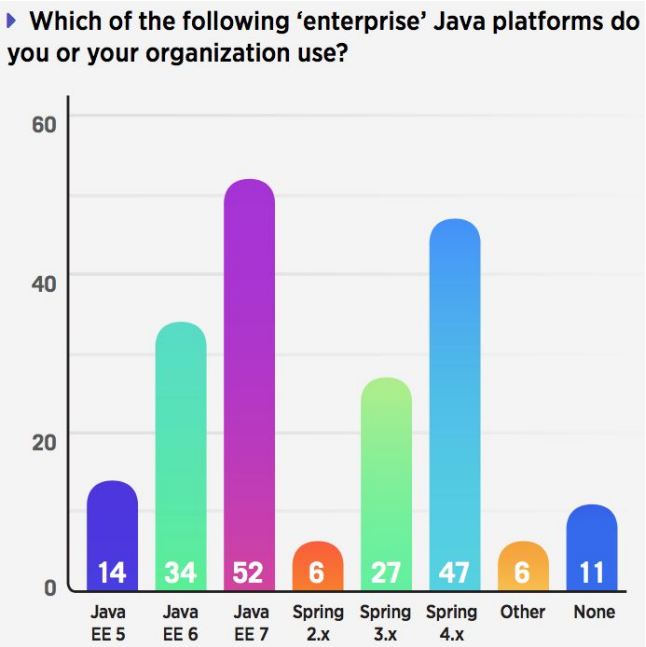
Monoliths



Microservices

Java and JavaEE still very relevant !?

Which of the following 'enterprise' Java platforms do you or your organization use?



What versions of Java are being used at your organization?

	FOR NEW APPS	FOR EXISTING APPS	NOT USING
Java 5 and below	0%	11%	89%
Java 6	2%	35%	65%
Java 7	17%	63%	32%
Java 8	89%	49%	9%

Source: DZone 2017 "Java Development and Evolution" whitepaper

What does AMM mean?

APPLICATION MODERNIZATION & MIGRATION

What does AMM mean?

APPLICATION MODERNIZATION & MIGRATION

Focus on business workloads and solutions.

What does AMM mean?

APPLICATION MODERNIZATION & MIGRATION

Digital transformation. Journey to the future.

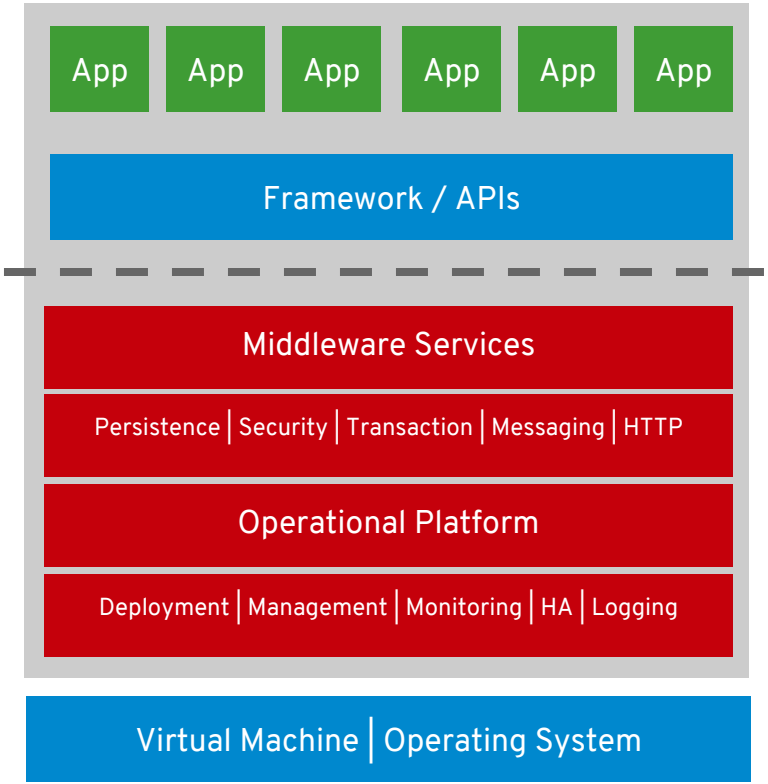
What does AMM mean?

APPLICATION MODERNIZATION & MIGRATION

Making old new again.

APPLICATION

Application Servers



Java™ COMPATIBLE ENTERPRISE EDITION

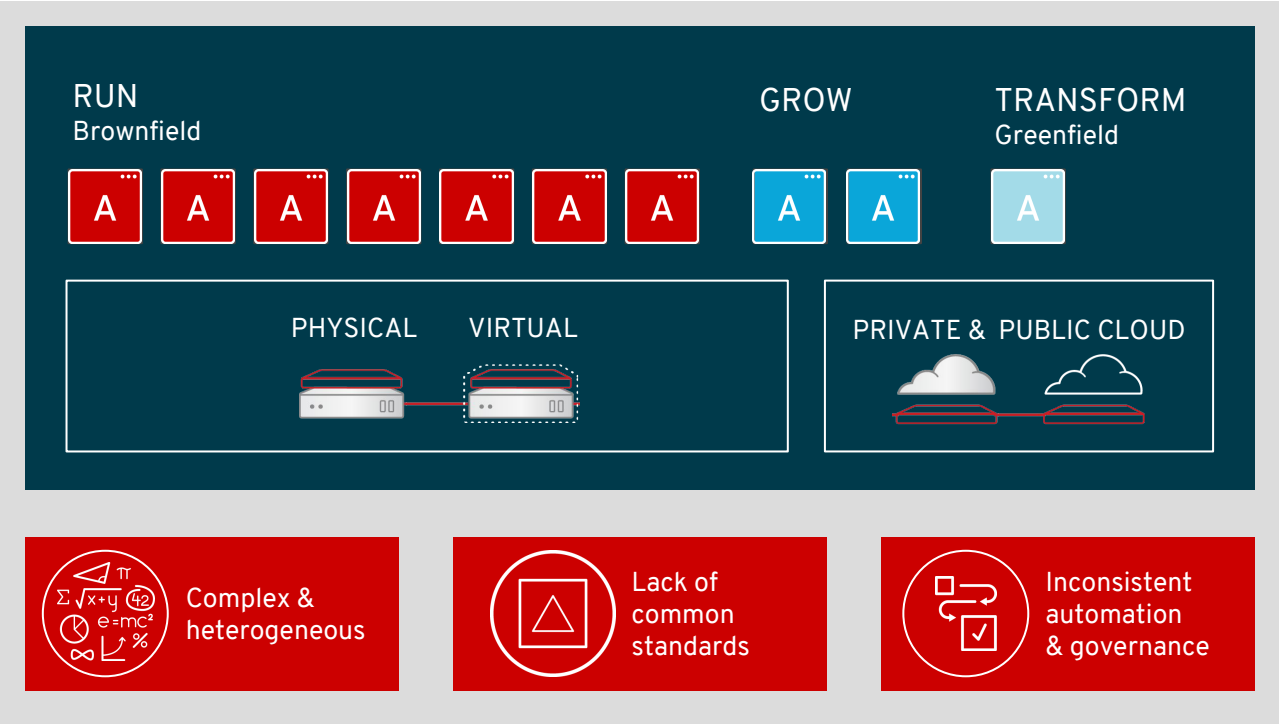
spring by Pivotal

JBoss® by Red Hat

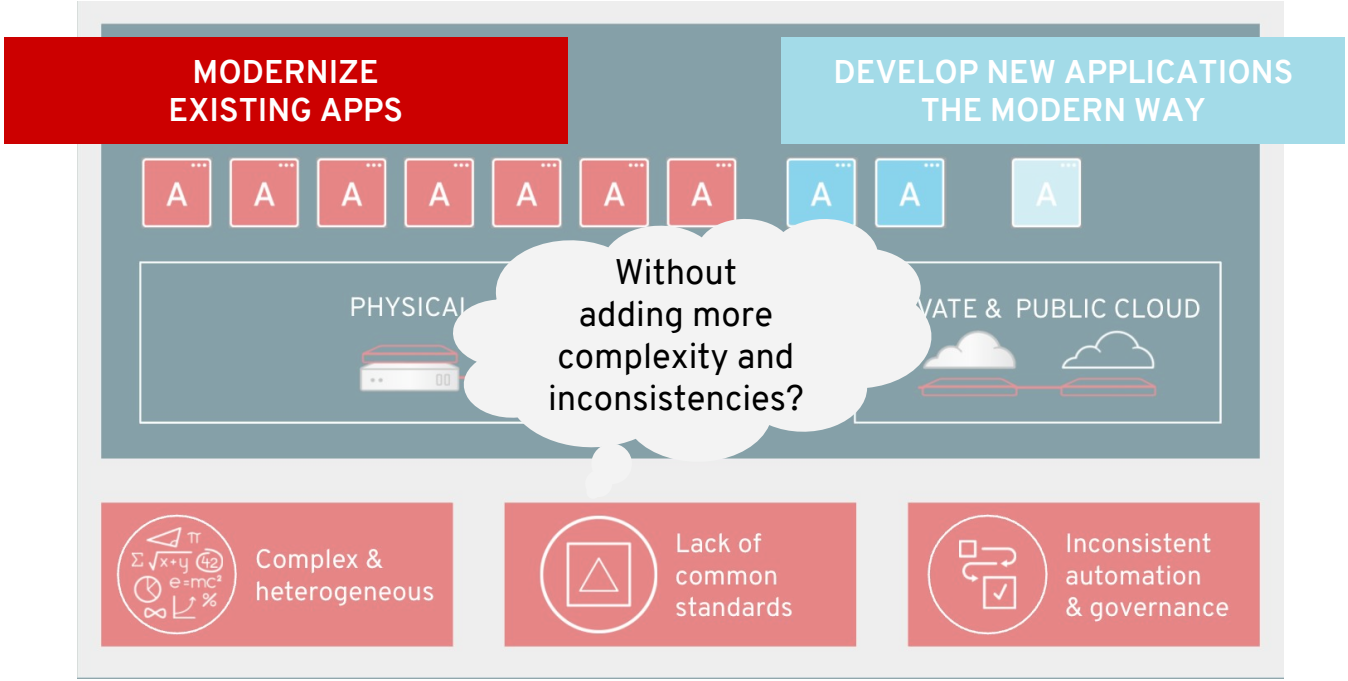
ORACLE®

WebSphere® software

Typical Landscape Today

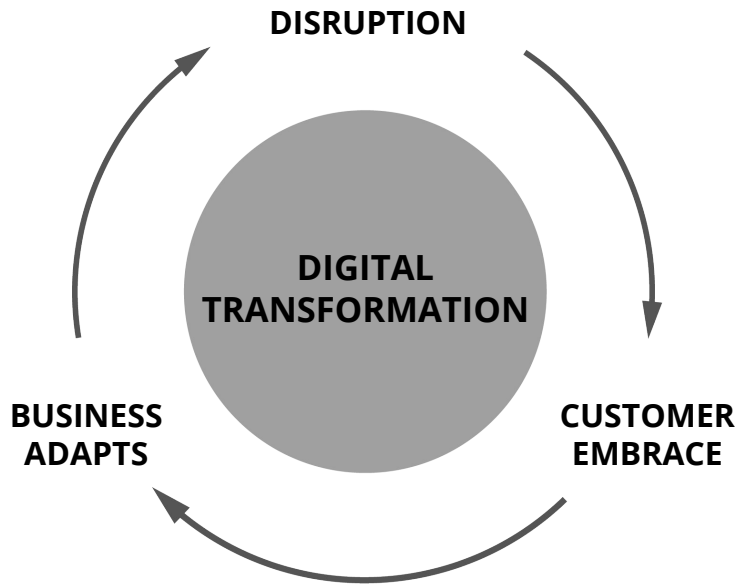


New problems to resolve



MODERNIZATION

Why Modernize?



Why Modernize?

- Every business is a technology business
- Code has no business value until it’s deployed
- Scale and Speed challenges
- Competitive challenges
- Gain Business Agility
- Growth enablement

MIGRATION

Why Migrate?

There is not a “off button” for your Business Applications

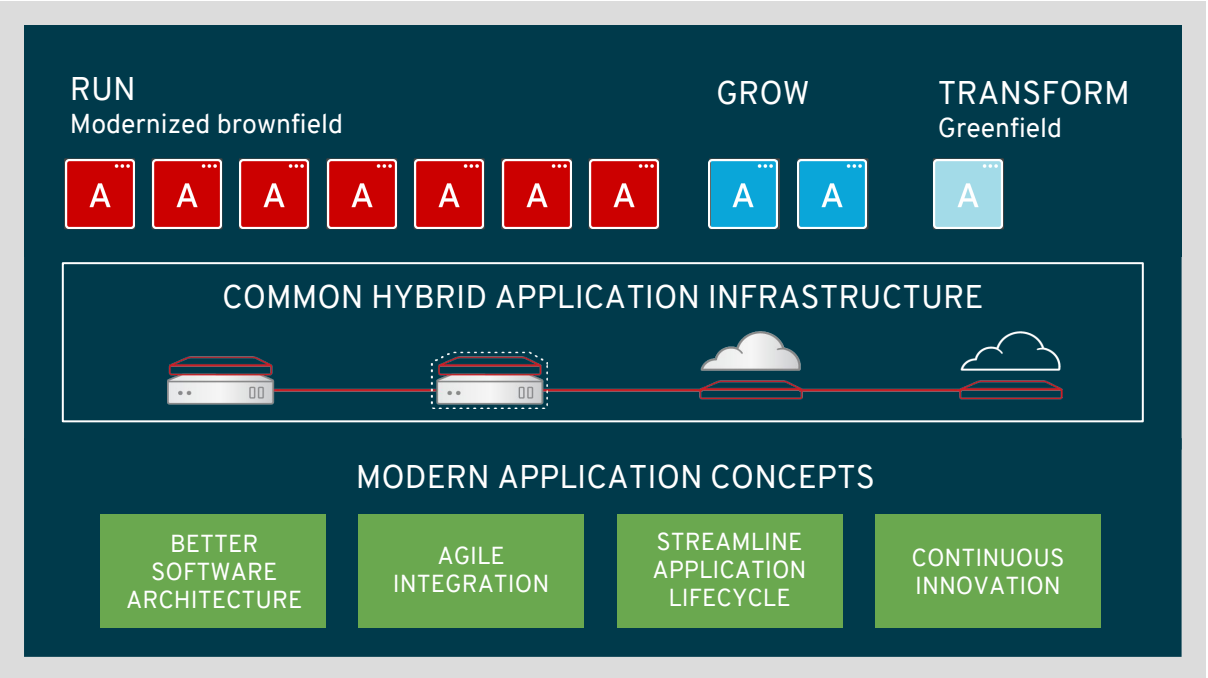
Why Migrate?

- Optimizing and streamlining existing application usage
- Unlocking more value from your IT investments
- Consolidate application instances
- Rehost or replatform the application to newer infrastructure
- Develop new application code to extend the life and utility of the legacy application
- Convert and update existing code into new development languages
- Restructure application code to support a more modular, loosely coupled services architecture
- Retire the existing packaged application and migrate to a new application

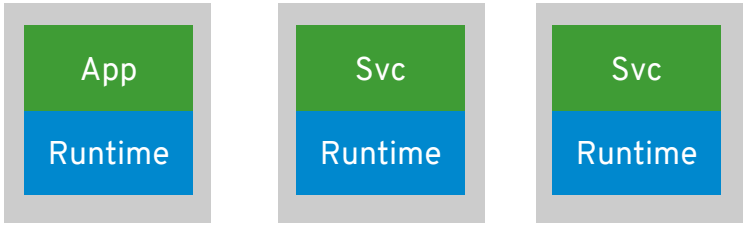
JOURNEY TO OPENSHIFT

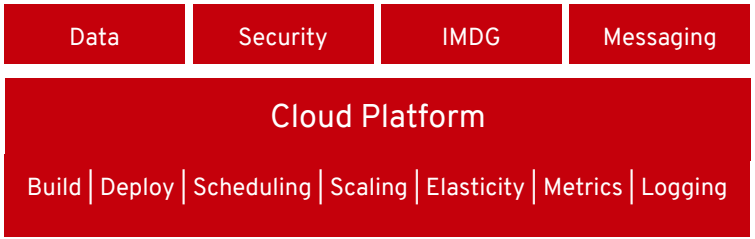
Where would like to be?

- One platform to support you today and tomorrow



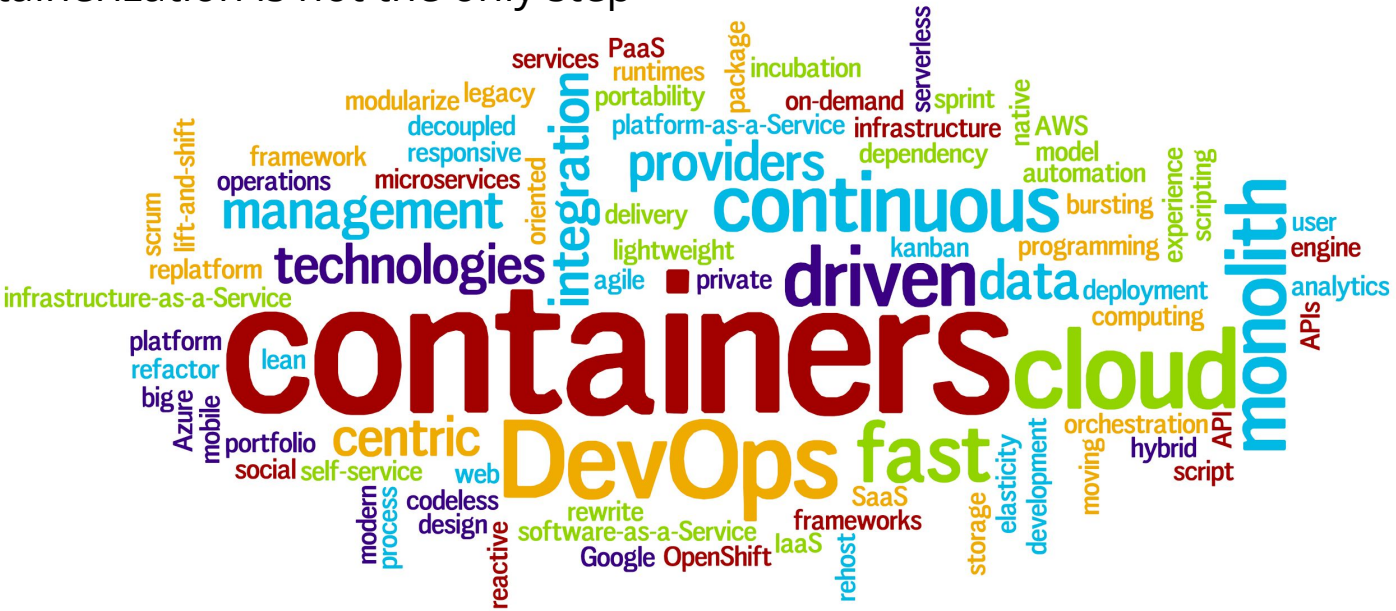
OpenShift is the new Application Server





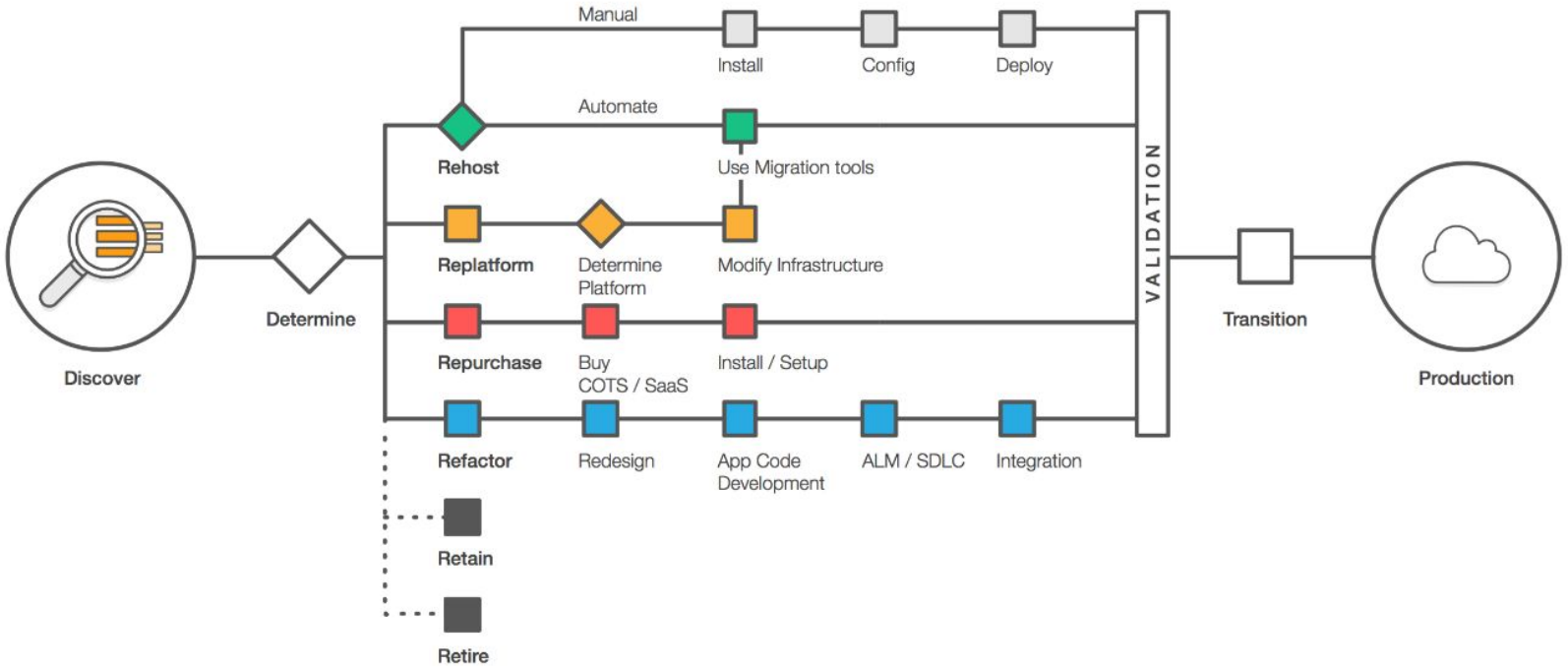
How do it get it?

- How do we build and run applications in the new world?
- Strategy and methodology
- Containerization is not the only step



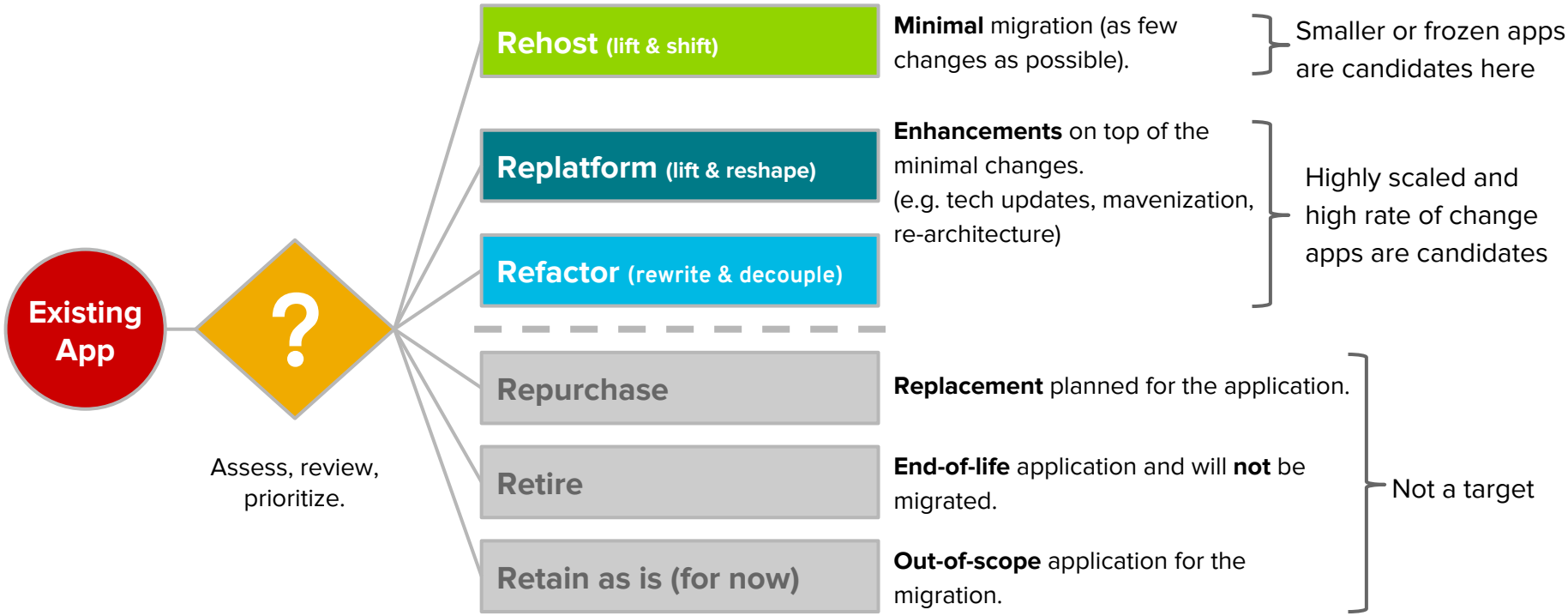
STRATEGIES

The 6 R's



Source: https://aws.amazon.com/cloud-migration/

Decision Tree



Patterns in Modernizing Applications

REHOST
(lift & shift)

- **Containerize** existing workloads
- Deploy them on a **PaaS**
- Keep external integrations and data on legacy
- Legacy applications have to be well written and suited

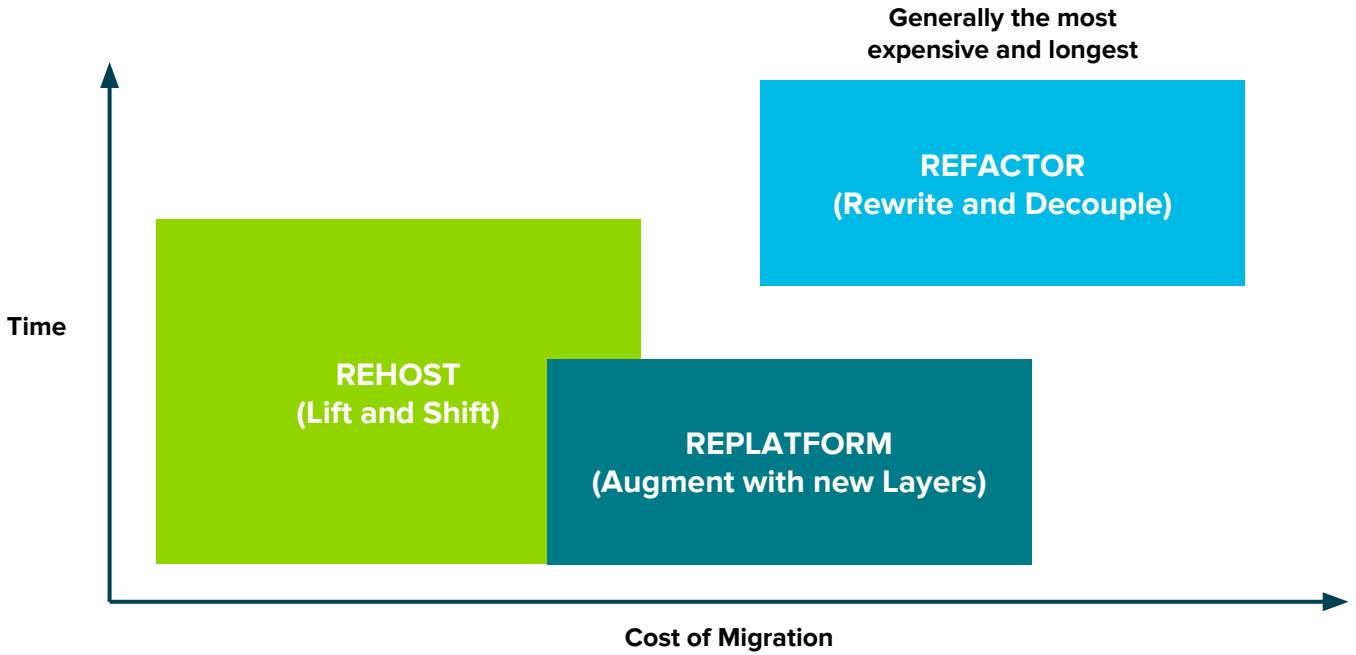
REPLATFORM
(augment)

- **Legacy remains intact**
- New layer - new capabilities
- Deploy on **PaaS**
- **New integration points** between legacy and new layers **(Need for Agile Integration)**

REFACTOR
(rewrite & decouple)

- **Legacy is totally replaced**
- New interfaces and data
- Use **PaaS** to run
- Some data and features can be re-wrapped, but mostly are retired.

Patterns in Modernizing Applications



Modern Application Concepts

Enhancing applications, platform and processes

BETTER SOFTWARE ARCHITECTURE	AGILE INTEGRATION	STREAMLINE APPLICATION LIFECYCLE	CONTINUOUS INNOVATION
Future-proof applications	Bridge old and new	Speed up your business	Foster an agile culture
Modularize "Fast moving monolith" Microservices Clean technical debt	Decouple, expose & integrate APIs, services & applications Need hybrid-cloud-enabled integration platform	Accelerate time from idea to production Continuous Integration & Delivery (CI/CD) Automation & self-service Container technology	Agile methodology DevOps principles Collaboration

CLOUD-READINESS

Cloud-Readiness

- “Cloud-Native”
  - Distinctive architectural characteristics
- “Cloud-Compatible”
  - Represents the minimum viable product
- “Cloud-Ready” / “Container-Ready” / “OpenShift-Ready”

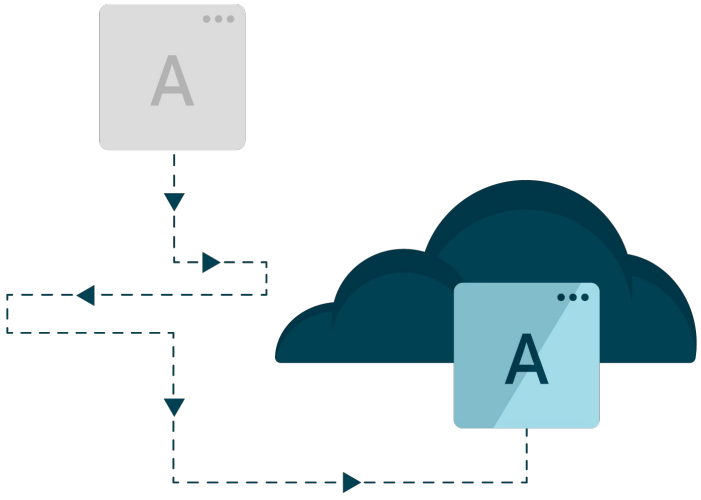
- Not every application **should / can / must** be made “cloud-native”
  - Container-Ready is often enough
  - Primary focus on OpenShift adoption instead on high re-architecturing efforts

Cloud-Native Application Ideals

- Ability to handle dynamic scaling, load configuration in flexible ways, and aggregate logs
- Dynamic discovery of dependencies
- Load balancing of requests to dependencies
- Enable application monitoring and distributed tracing
- Circuit breaker and and bulkhead patterns
- Feature toggles and health checks

Journey from Cloud-Compatible to Cloud-Native

- Start with Cloud-Compatible changes
  - Create support from external configurations
  - Remove IP bindings
  - Run on Linux
  - Ensure logs write to console/stdout
- Progress with subsequent iterations
- Actual end state should be dictated by the needs of the business



CONTAINERIZING APPLICATIONS

What is Containerization?

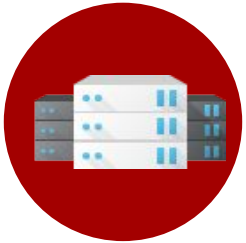
- Packaging of a configured application and all its dependencies into a light, portable, cloud-ready sandbox.

Containerization Challenges

Beyond the code

APPLICATIONS

Dependencies
Concurrency
Log and file system
State and persistence
Environment independance
Backing services
Port bindings



INFRASTRUCTURE

Automation
Admin processes
Configuration management
Application aggregation
Availability and reliability
Elastic scaling
Security

PROCESS

Application lifecycle
Application discovery
Log management
Monitoring and health
Knowledge governance

DEMO TIME

Demo Time - Show me the code

- Standard JEE Application
  - JPA + Hibernate to manage an external MySQL Database
  - JMS Queues to add more information to the application
  - MDB to load data from JMS Queues to store information into MySQL Database
  - Initial Servlets consuming extra startup time

- Main issues
  - Extra time to be ready after redeploy it
  - Monolithic architecture

Demo Time - Show me the code

- Migration
  - Package using Wildfly Swarm
  - Deploy using Maven Fabric8 Plug-In
  - Automate
  - Monitoring
  - Scale up/down

Demo Time - Wildfly Swarm

- An innovate approach to package and run Java EE applications
- *Just enough App Server*: package your app with required runtime dependencies (but nothing more)
- Based in Wildfly Application Server
- Packaged as an Uber Jar (self-contained, executable Java archive)

Demo Time - Maven Fabric8 PlugIn

- Brings your Java applications on to Kubernetes and OpenShift
- Provides tight integration into Maven
- Focus on two tasks:
  - Building Docker Images
  - Creating Kubernetes and OpenShift resource descriptors

Demo Time - Show me the code

- Modernization
  - Deploy Messaging Services
  - Deploy Integration Services
  - Deploy CronJobs
  - Remove old operations in monolithic application

OPENSHIFT

Meetup

Questions?

OPENSHIFT

Meetup

Thank you!