Modernizing Complex Legacy Applications

Pranjal Bathia
Mike Moore
9-May-2019

Legacy Application Modernization

legacy adjective

leg·a·cy | \ ˈle-gə-sē \

of, relating to, or being a previous or outdated computer system
(Merriam-Webster)

denoting or relating to software or hardware that has been superseded but is difficult to replace because of its wide use (Dictionary.com)

Should you modernize?

There are reasons to change (some better than others)

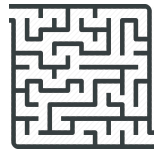
- Improved functionality
- Expansion of system capabilities (scalability, performance, maintainability, flexibility, etc.)
- Opportunity and Risk Management

There are also reasons for caution

- Cost
- Business disruption
- Strategic drift and feature decay

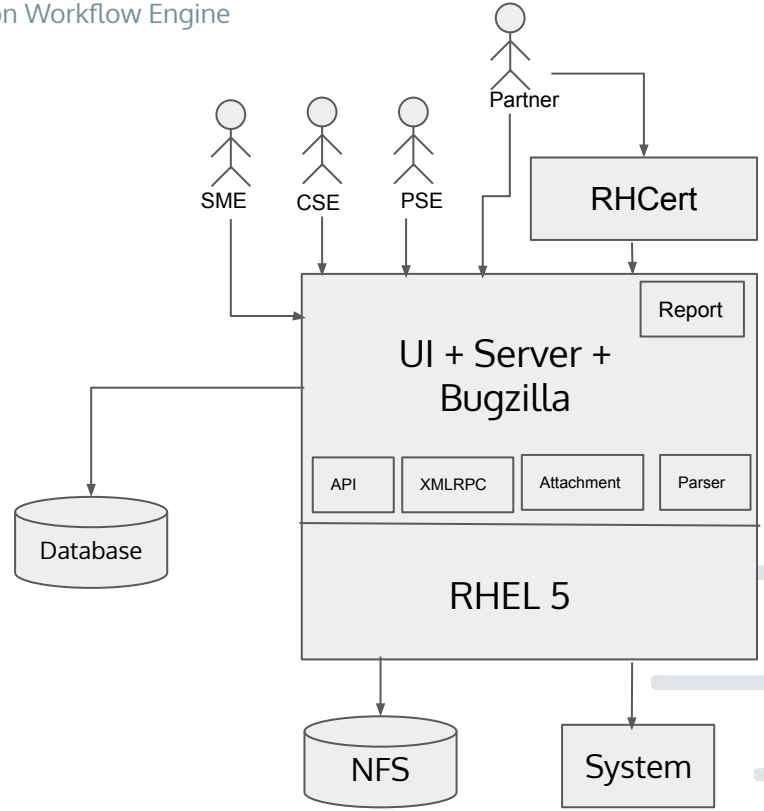
From 2001: A Space Odyssey (MGM)

Legacy CWE system

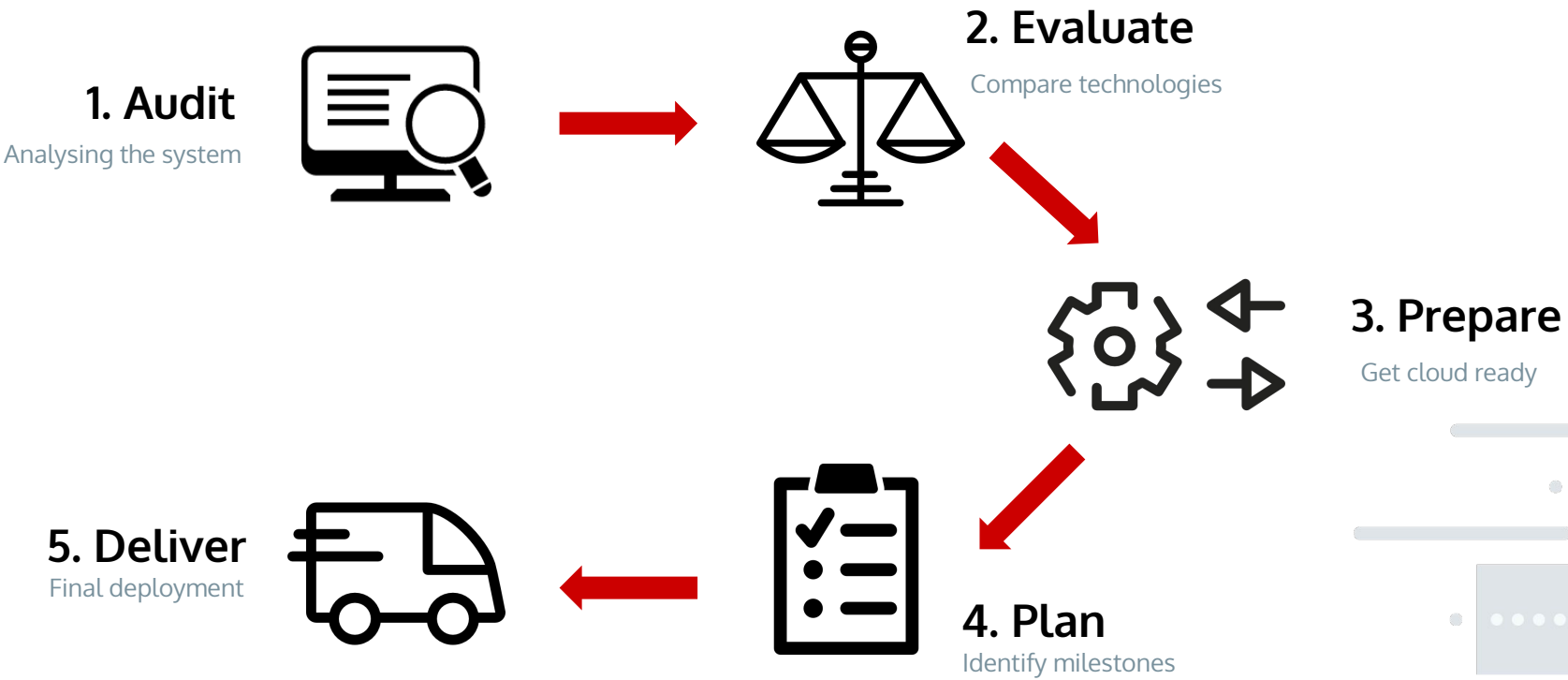


Certification Workflow Engine

- Usecases for solution certification
- Used by partners and Internal associates
- Decade old app, organically developed
- Undocumented. Ambiguous feature set
- Supportability Gaps
- Performance issues
- Restriction on developing more features

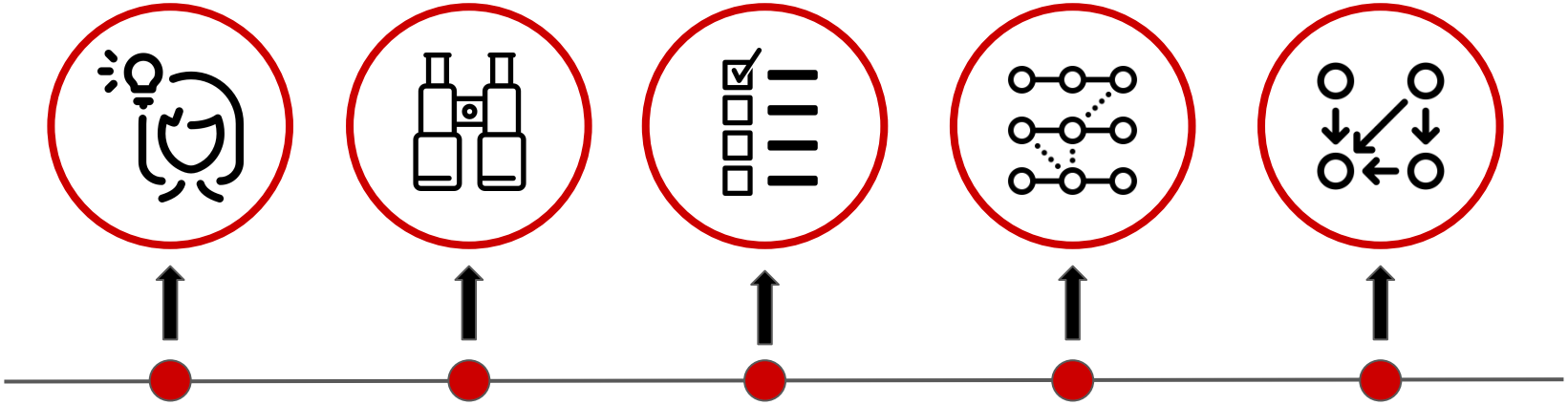


How to start ?


1. Audit
Analysing the system
2. Evaluate
Compare technologies
3. Prepare
Get cloud ready
4. Plan
Identify milestones
5. Deliver
Final deployment

Audit

Analysing the system



Understand

Understand business and the use case of application

Identify

List down discrete applications and infrastructure

Assess

Examine access logs to list down functions being used and which one can be retired

Map

Map relationship between applications and check which one can be combined

Dependency

Understand business dependency to make decision on disruption it can handle

Evaluate

Compare against different parameters

	Option 1	Option 2	Option 3
Operating System	✓	✗	✓
Database	✗	✓	✗
Application Server	✓	✓	✓
Programming Language	✓	✗	✓
Monitoring tools	✗	✓	✗
Storage	✓	✗	✓
Cloud	✗	✓	✓
Cost	✗	✓	✗

Evaluate

We used open source technology

RED HAT® OPENSHIFT

RED HAT® ENTERPRISE LINUX®

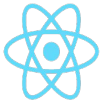
RED HAT® JBOSS® DATA GRID

RED HAT® JBOSS® A-MQ

RED HAT® JBOSS® FUSE

RED HAT® JBOSS® BRMS

Redux



React

PATTERNFLY

Prepare

Get cloud ready

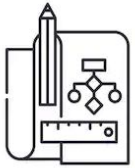
Rehosting

Replatforming

Repurchasing

Retain

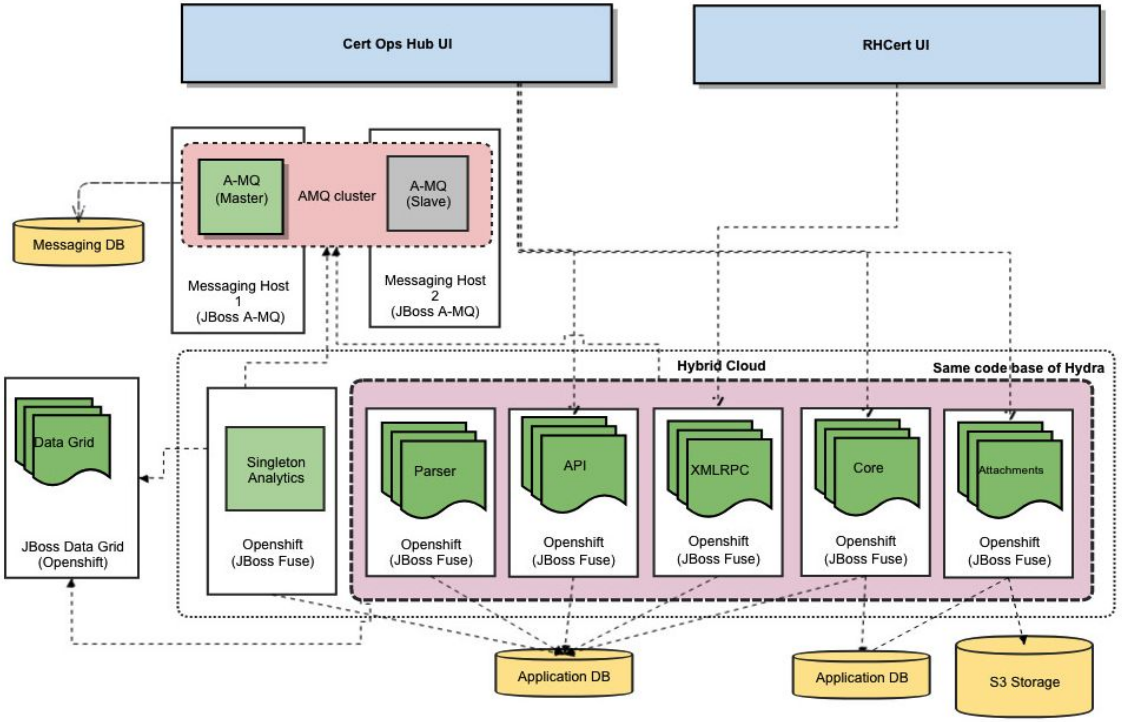
Retire



Rearchitecting

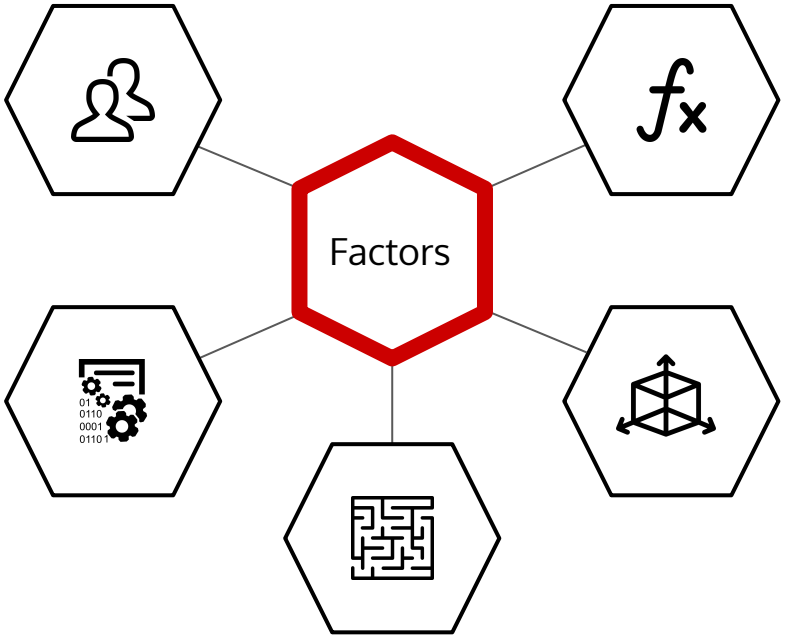
Prepare

Microservice Based architecture



Plan

Identifying milestones allows minimal disruption to business and most effective method for migration



Users

Release MVP for set of users to minimize the business impact

Business Functions

Chose simpler functions which increases chances of success

Sensitivity of Data

Pick up function that deals with less sensitive data to start

Size of application

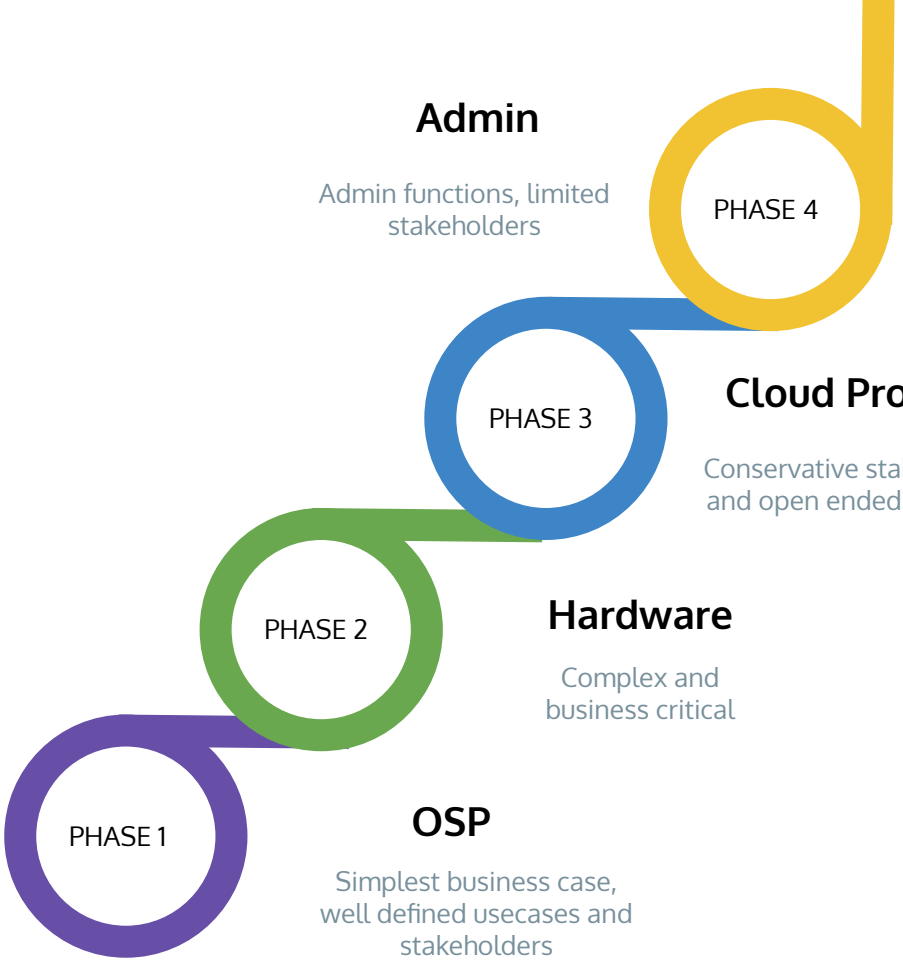
Opt for a smaller application which helps to deliver & learn faster

Complexity of Data

Go for a service which has less complex structure of Data

Plan

Phase based approach



OSP

Simplest business case, well defined usecases and stakeholders

Hardware

Complex and business critical

Cloud Program

Conservative stakeholders and open ended usecases

Admin

Admin functions, limited stakeholders

Plan

Phase based approach

Change Management

Develop Champions

Align Teams

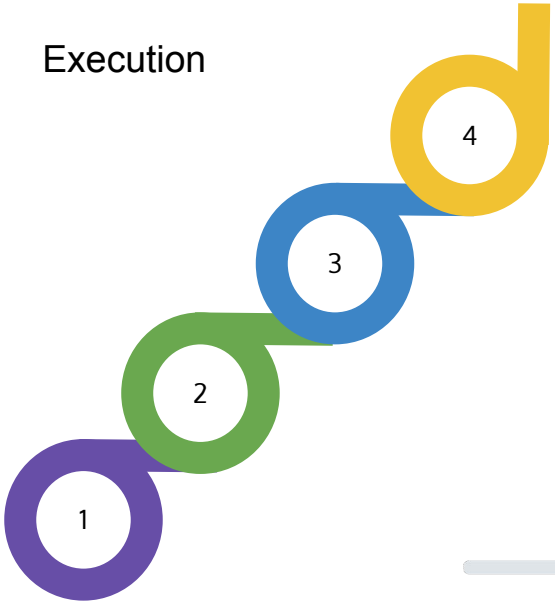
Communication

Process Transformation

Front load Impacts

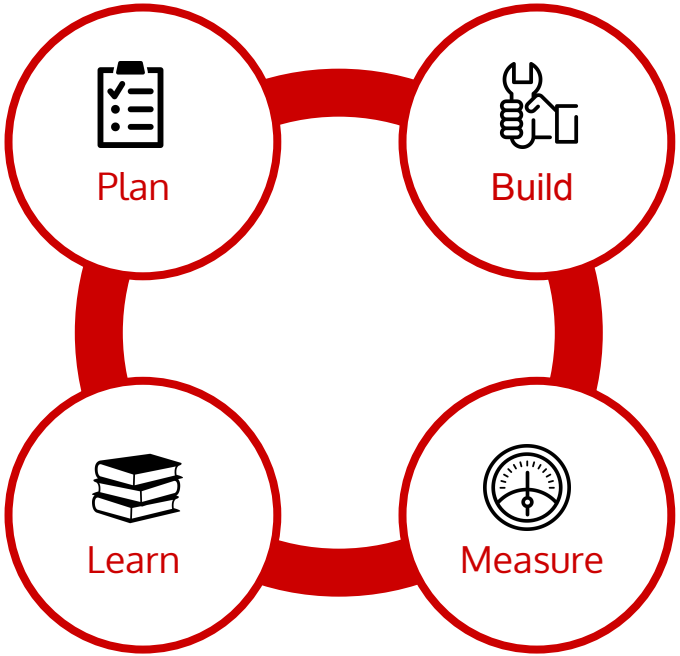
Skate to where the puck will be

Execution



Deliver

First release should be MVP (Minimal Viable Product)



Tips

1. Do a thorough **impact analysis** as part of your audit
2. **Containerization** allows you to deploy your application anywhere
3. Building **smaller services** helps efficient decoupling of system components
4. Focus on the **business processes** and **users** in your transitional states
5. A **phased migration** can help mitigate unforeseen pitfalls
6. Identify opportunities to include **automation** in every phase

RED HAT SUMMIT

THANK YOU


linkedin.com/company/Red-Hat


facebook.com/RedHatinc

youtube.com/user/RedHatVideos


twitter.com/RedHat