Modernization: How do I begin?

Kyle Brown

IBM **Cloud**

IBM

Why is it so hard to start modernization projects?

Many Modernization Projects die before they even get going – WHY?

- Differing and Competing Goals
- Lack of clarity and agreement on approach
- Lack of the long-term commitment (for funding and priorities) it takes to complete the job

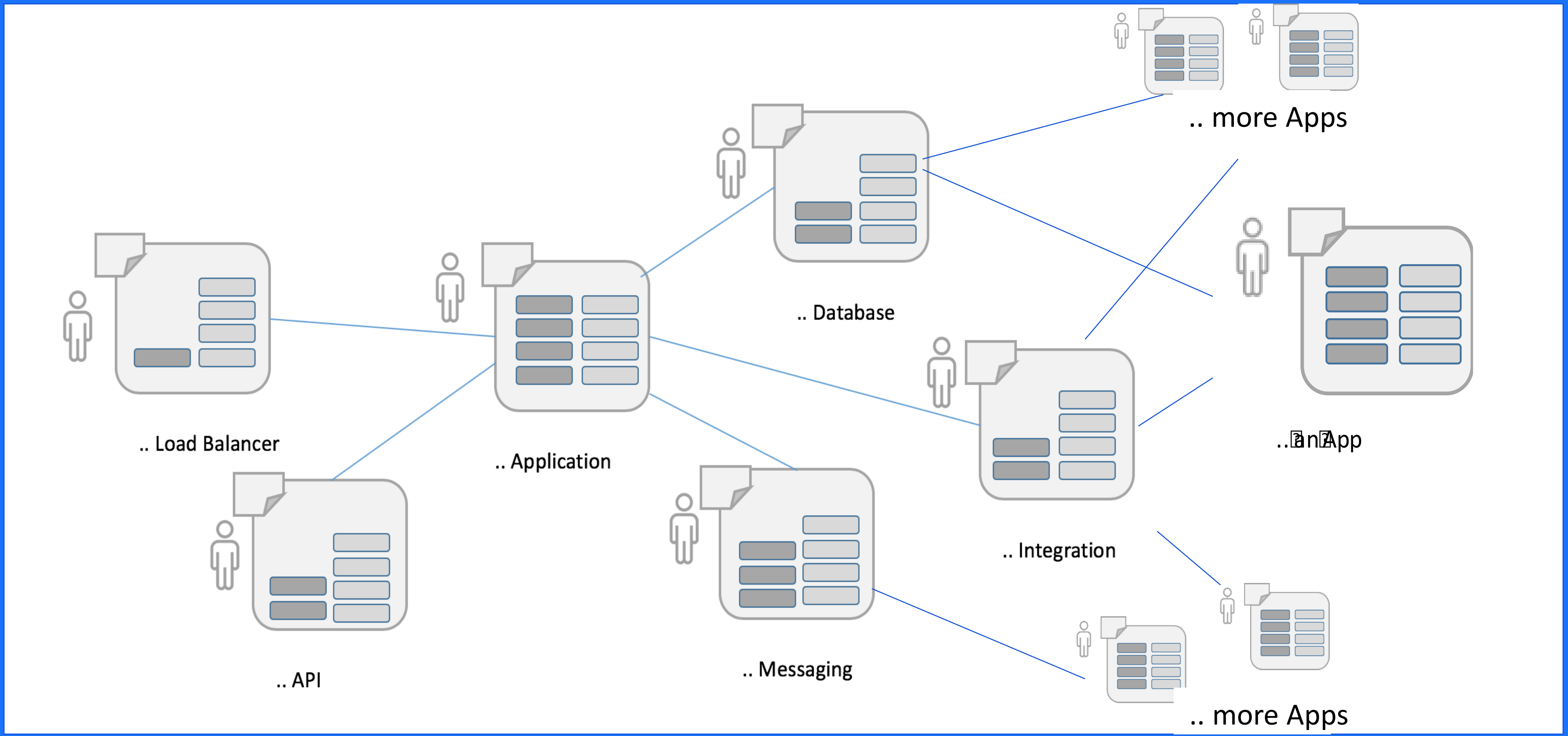
Using Case Studies and examples of our approach we'll show how you can align on all of these in order to succeed

What does modernization even mean?

An ***Application*** *consists of one (or usually more) custom-written program components together with the supporting infrastructure, application middleware, middleware services (such as databases, queuing software, integration software and other aspects) and other components that make up a complete solution.*

Application Modernization *is the process of updating an application so that it can be maintained, extended, deployed and managed in a way that allows the application to meet the business' current and future needs.*

The Enterprise Hairball



Why Modernize?

#1 You can’t develop features at the pace your business requires - and it’s the technology choices and architecture (and not your processes or team constraints) that is causing that.

#2 The architecture of your application is hindering you from being able to add functionality because of fragility (you can’t test it) or constraints arising from technology choices (technical debt)

#3 Your application is expensive to maintain and extend because either the infrastructure is excessively costly (e.g. older versions of middleware that require special support contracts) or the skills required are too expensive to maintain.

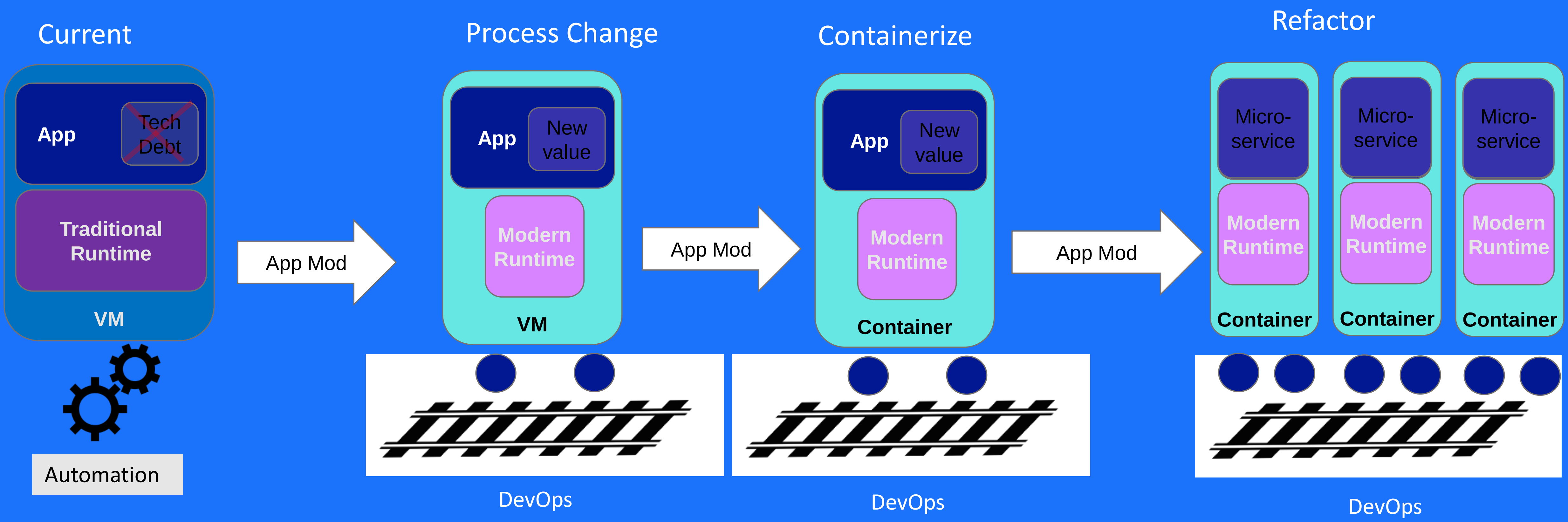
Secrets of successful modernization projects

#1 The organizations that succeed are able to ***put in the required investment and sustain it*** *over the period of time (usually measured in years) that a successful enterprise application modernization will take.*

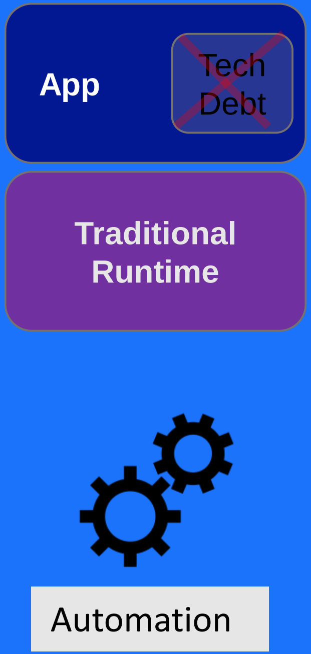
#2 The organizations that succeed are willing ***to make the organizational changes necessary*** *to succeed in the cloud.*

#3 The organizations that succeed are willing ***to change their architecture, development and operational processes*** *to match the pace and type of effort that a modernization will take.*

Our Proven Approach



The Current State



Some applications will NEVER leave their current state.

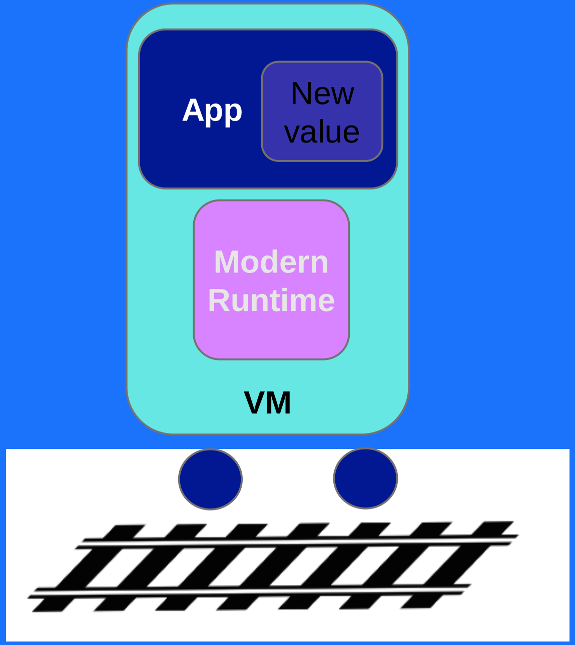
#1 The application may have a limited lifetime.
#2 The application may be replaceable by SaaS.
#3 The application may be one that is supported by third party that is resistant to changing their implementation, automation or management.

For companies that consider themselves to be “technology companies” this total number of these types of apps may be in the low double digits, perhaps 20-30% - however in very traditional companies this can go as high as 80%.

For these applications, the best approach may be a tactical lift and shift to VM’s in the cloud (for instance, IBM’s VMWare offerings)

Step 1: Process Improvement

Process Improvements



DevOps

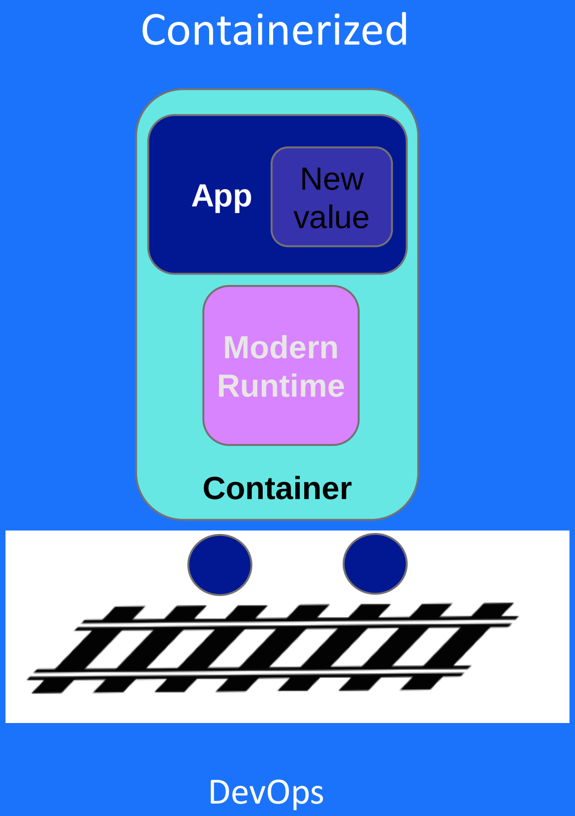
It's usually not your application that is going to be the hard part of modernization **- it's your own processes and organizational structures.**

The two most important changes you can put in place are:

- **DevOps pipelines and the principles surrounding them** (e.g. CI/CD and Automated Testing)
- The principle of **Infrastructure as Code and automation technologies** like Ansible and Terraform

Removing process barriers in deployment (handoffs between Dev and Ops, and slow-moving processes like change boards) can likewise significantly improve your ability to deliver code in smaller increments more often.

Step 2: Containerization

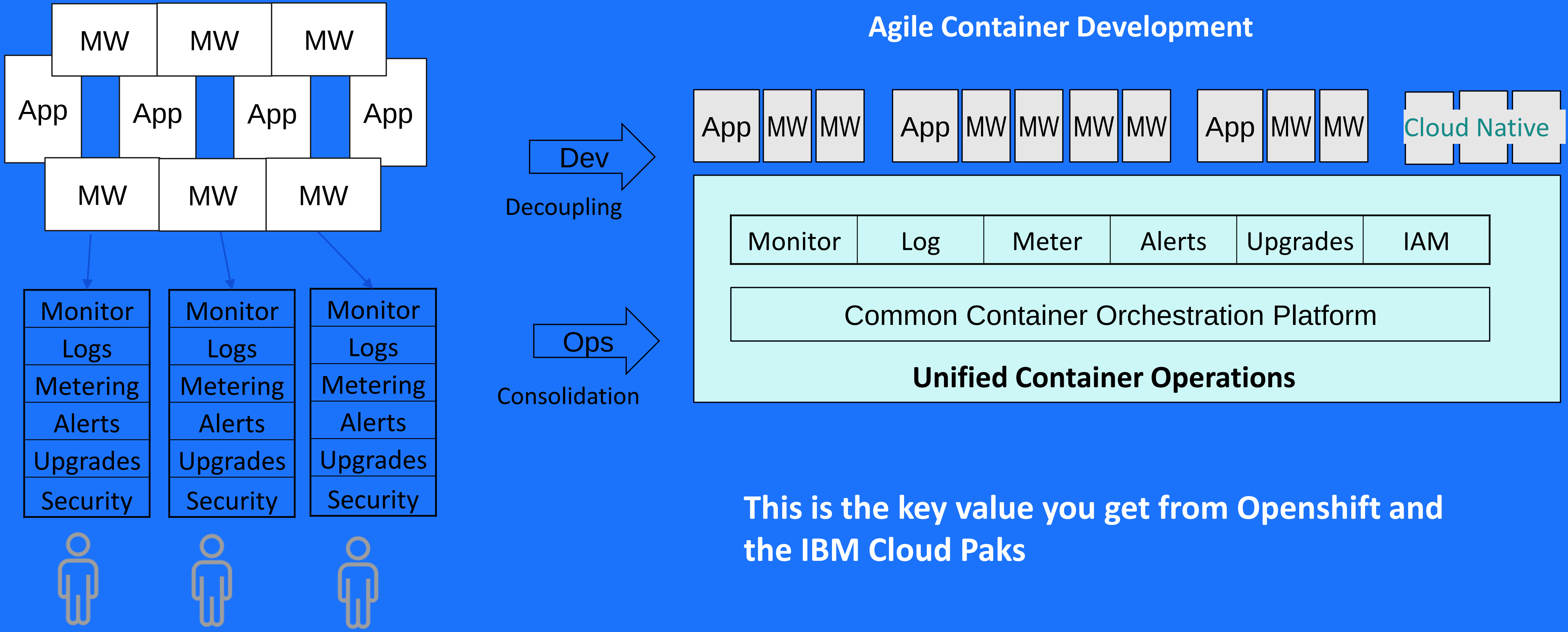


Containers offer significant benefits; faster startup, smaller runtime footprint, denser packing in the same amount of hardware.

- Containerization can also bring the the benefits of a **limited blast radius**
- The key concept to grow to - the idea of **immutability** and **replacement**.

The most important gains from Containerization are when it is used as part of adopting a common platform for operational services

A Common Platform for Operational Services



This is the key value you get from Openshift and the IBM Cloud Paks

Step 3: Refactoring

Decouple application complexity

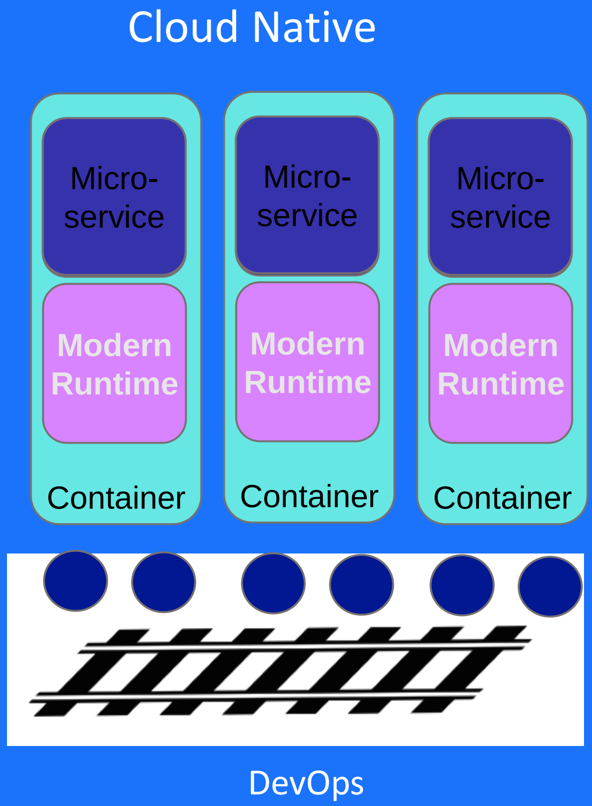
- 12 factor rules
- Microservice architecture
- Strangler pattern
- CQRS Pattern

Decrease scope of release

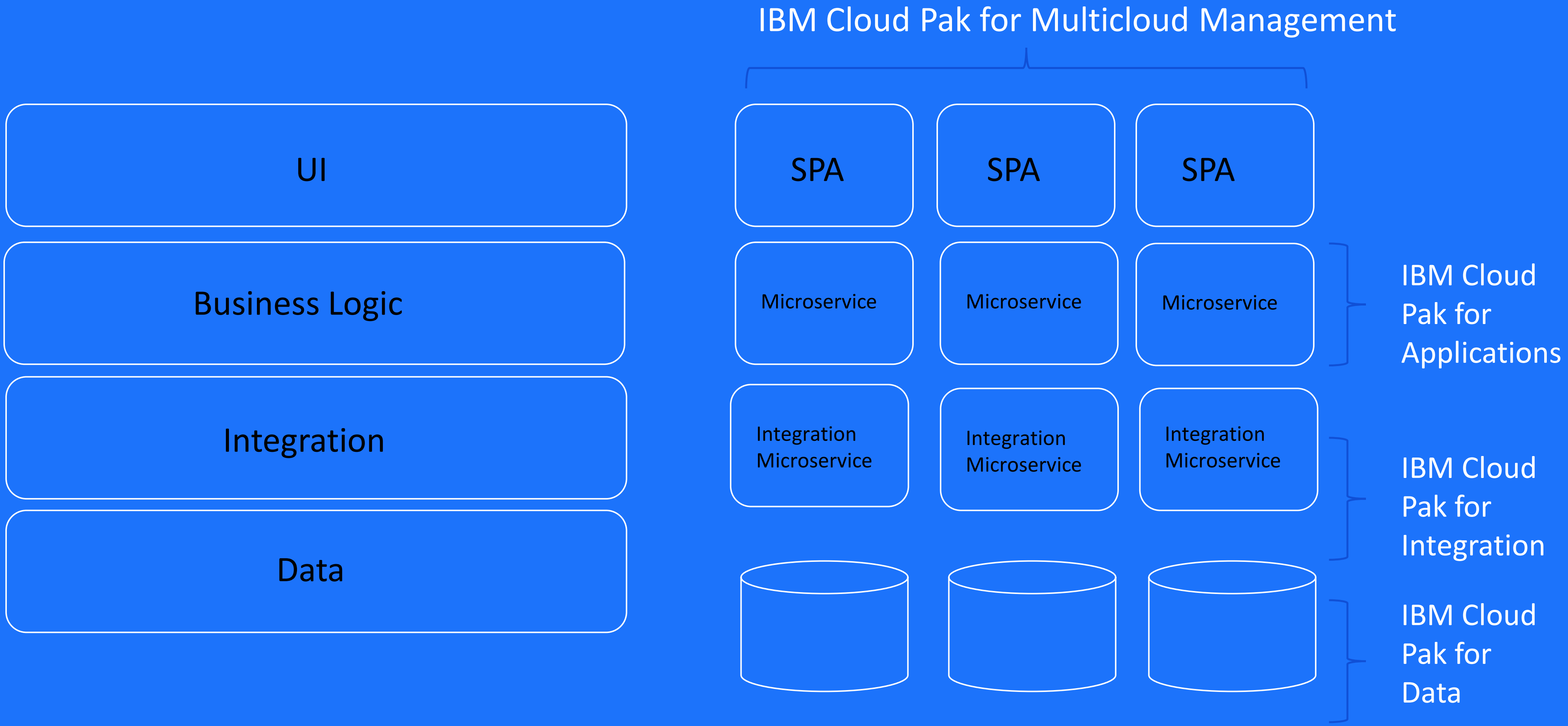
- Deliver as minimal viable product
- Release new features more frequently

Improve Delivery Capability

- Introduce Test Driven Development
- Introduce Site Reliability Engineering



Containerization and detangling the hairball



Why microservices shouldn’t be your goal

Microservices are a wonderful technique for building loosely coupled applications

Each Microservice is independent in scaling, deployment, data control and team ownership

But you don’t always need Microservices

- How large is the application?
- Does it all change at the same rate?
- Are you reintroducing coupling by building complex microservice networks?

Can your application be maintained, can you sustain rapid, incremental releases, and do your operational approaches allow you to identify problems quickly and return to service immediately?

If breaking a monolith into Microservices helps achieve these goals, then that is a tool you can use.

Engaging The organizations

There are as many different ways to fail at a Modernization project as there are different teams in your organization

Development Leadership	What skills are needed? Should I hire new developers? How does this change our development processes?
Operations Leadership	What new toolchains will be needed? How does this change my operational staffing needs? What new roles will be needed and how do we prepare?
Architecture Leadership	What is the role of Architecture in the cloud? How do architects work with Agile teams?
Business Leadership	How does the relationship between business and IT change?

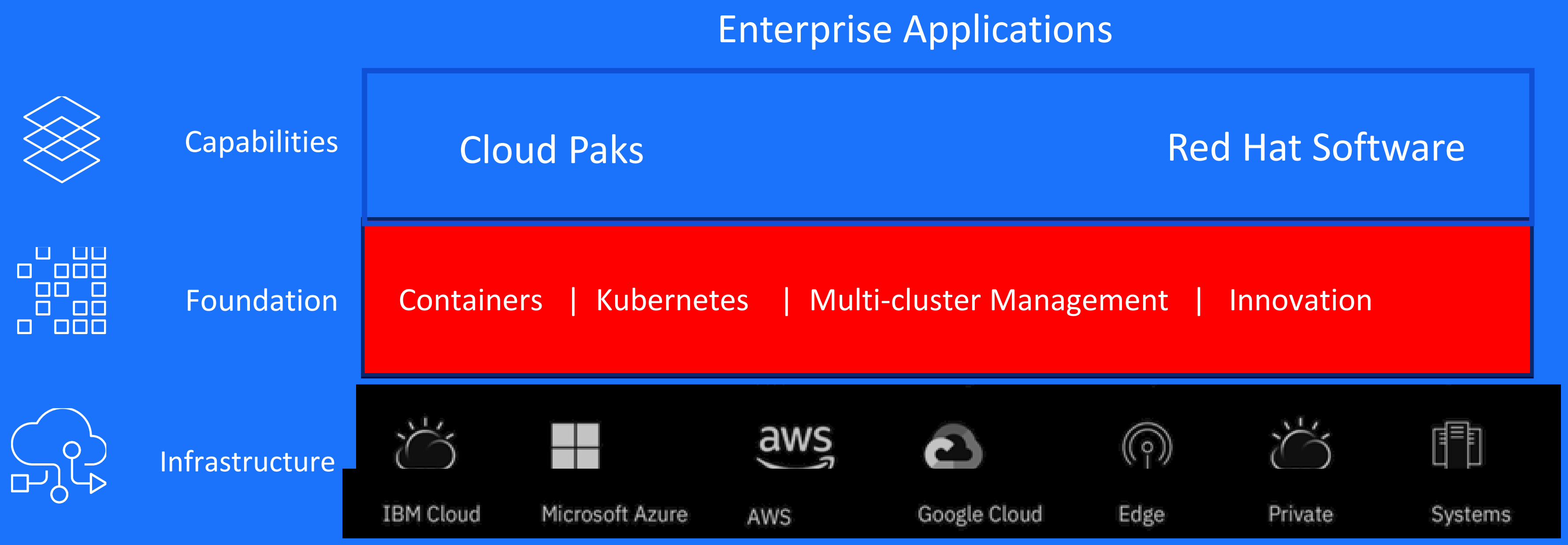
Organizational Engagement Case Study

American Airlines

Development Leadership	Design Thinking exercises to quickly understand the business problem and Rapid MVP iterations to address the most problematic issues. Co-development in squads to help bring skills of teams up to speed.
Operations Team	Design Thinking to describe to-be operational model. Engagement with our CSMO team to teach new SRE principles and understand changes to ITIL processes.
Architecture Team	Experienced Agile Architects as mentors to help understand the new role. Help with setting up new Guild structure to foster embedding
Business Leadership	Intensive training and mentoring in Design Thinking techniques to help understand how to build the Product Owner discipline.

The need for a Hybrid Platform (and a Platform team)

IBM's hybrid multicloud platform



A Common platform lets you manage working in a complex hybrid environment
But it doesn't set itself up – you have to have a team responsible for the platform

US Insurance Company Modernization Workstreams

Container Foundation

- CP dev/test environment
  CP enablement
  Validate full deployment scope
- CP full deployment
  CP integration
- Compliance &
  Production readiness
- **Factory Scale :**
  self-service, multi-tenant, charge-back..

DevOps Foundation

- Refactor pipeline for containers
  Add Liberty containers pipeline
- Cross-env deployment pipelines
- Pipelines deployment
- **Factory Scale:**
  Open source consumption governance

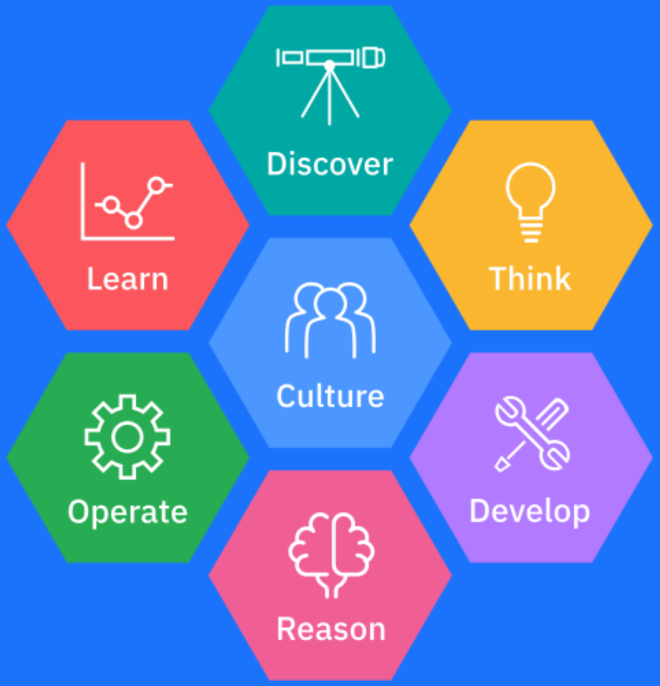
Application Modernization – Back-end

- Prove Liberty in containers
  CP and pipeline integration
  Validate full scope & timeline
- Critical mass containerization
- Full containerization
- **Factory Scale**:
  re-use/publish/Arch Center

Application Modernization - Front-end

- Sync up with Garage best practices and integrated with DevOps foundations
- Accelerate
  Front-end refactoring
- Complete Front-end refactoring
- **Factory Scale:**
  Cross-org template

The IBM Garage Method



Combining industry best practices for **Design Thinking, Lean Startup, Agile Development, DevOps** and **Site Reliability Engineering** to deliver innovative solutions in a consistent and repeatable way

- Application Modernization cannot proceed within traditional organizational and cultural models
- Handoffs between teams hamper organizational agility
- New Skills are required for development in new technologies
- Operational approaches change with new technologies
- So as part of any application modernization project you also need to consider a holistic approach to organization/application development/DevOps and Management.
- That's the **IBM Garage Method for Cloud**

Using a COC to kickstart the transformation

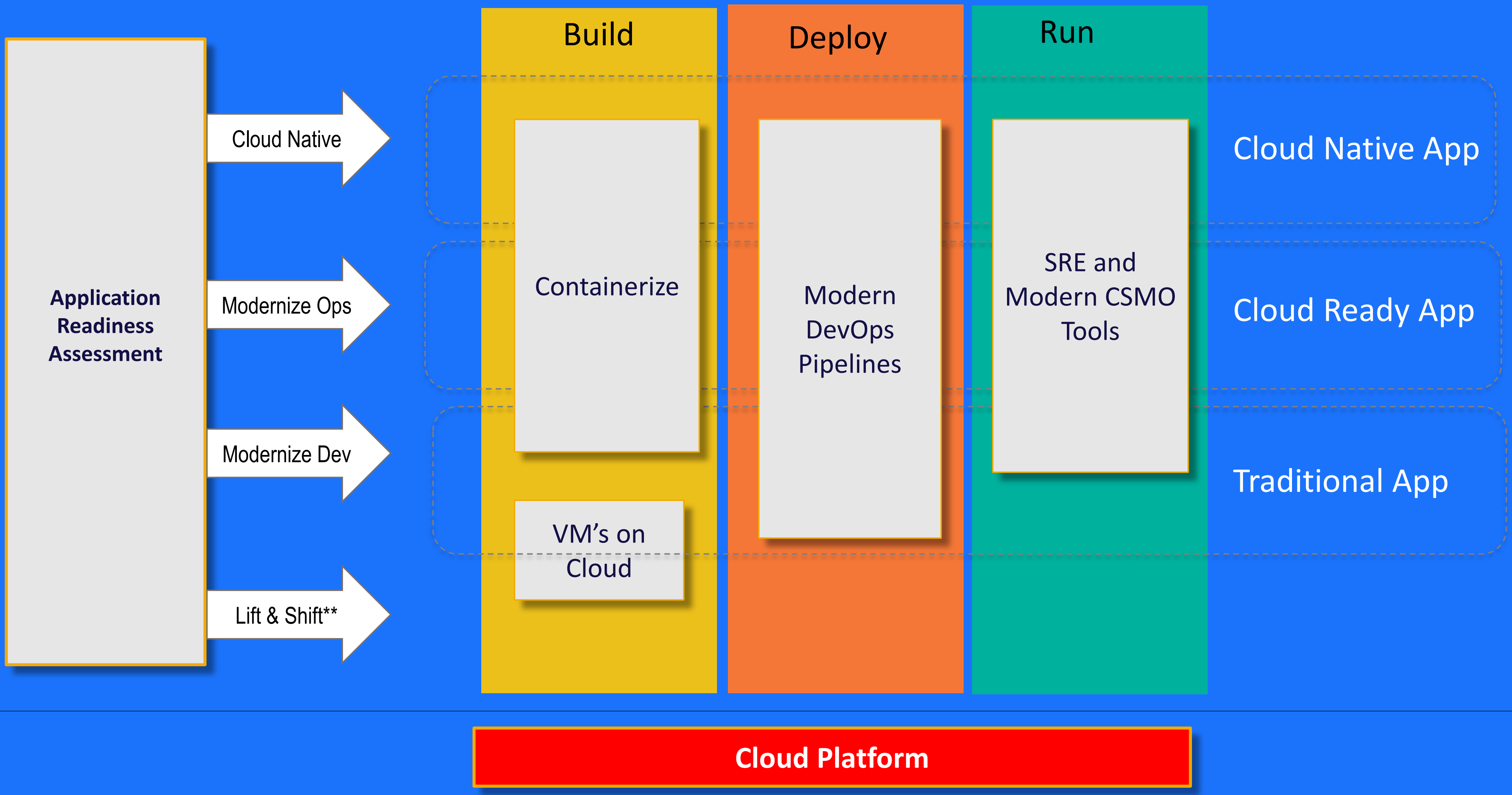
*A **Center of competency*** (**COC**) is an independent body that develops common solutions and acquires new skills that are then spread throughout the organization. This approach increases the likelihood of success of each new modernization project.

A COC has four goals:

- Promote best practices and standard processes that enable repeatable success
- Provide as-needed expertise to solve specific problems that are related to development and deployment
- Help teams become self-sufficient in knowledge and expertise
- Create a focal point in the enterprise for looking ahead to new disruptive technologies and issues beyond the immediate focus of current projects

An overall solution approach for modernization

https://www.ibm.com/cloud/garage/architectures/application-modernization



Start with a visit. Change your perspective. Partner with us to make your ideas into reality. Fast.

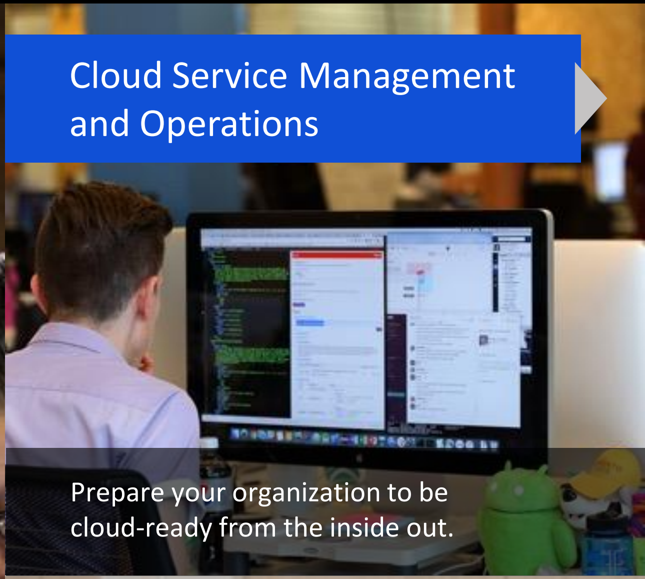
Come to our unique innovation space and see first-hand how you can transform your business.

IBM Design Thinking Workshop

Bring your business and technical leadership together to align around your big idea and define the vision for turning it into reality.

Minimum Viable Product Build-Up

Rapidly iterate on your chosen idea to create a production-ready code that you will be proud to call yours.



Cloud Service Management and Operations

Prepare your organization to be cloud-ready from the inside out.

IBM Cloud Garage Academy

Bring your team to speed with the latest skills and technologies to thrive in today’s digital age.

Super Regional U.S. Bank

Modernization of core services in a monolithic codebase applying microservices architecture to accelerate time to value of new capabilities

• Problem Statement

- A super regional bank with over 15 million transactions per day had their core retail banking services in a legacy SOA resulting in:
- Delayed time to market and unpredictable application stability resulting in negative customer experiences
- Unable to onboard new products due to architectural complexity
- Outdated technology landscape and complexity of service orchestration

IBM Value Proposition

- IBM garage drove innovation at scale through co-creation of a modernization strategy to migrate business capabilities onto a next gen microservices architecture on RedHat OpenShift platform
- Introduced new ways of working to skill-up Bank team members in Garage Method

Outcomes achieved

- **4 weeks to launch MVP into production (from API Design Thinking Workshop to production deployment)**
- **Developed modernization strategy and implementation approach for 200+ services from legacy to target state**

Engagement Profile

- Joint squad of Bank and IBM resources (Squad Lead, Developers, SRE, Architect)
- Defined Modernization strategy and rationalization of existing code to inner/outer APIs aligned to BIAN service domains
- Upskilled Bank resources through immersion into IBM Garage Method (pair programming, test driven development, XP practices, design thinking and hypothesis driven design)
- Accelerated development through microservices API generation, test driven development and test automation; vertical slicing of business capabilities to plan/transition consumers onto target architecture

Southeast Asian Bank

Modernization of payment integration services from a monolithic codebase applying microservices architecture

• Problem Statement

- Bank among the fastest growing commercial banks in India; transforming from a traditional institution to offering state-of-the-art products and services to a diverse group of more than 50,00,000 customers.
- Built out at scale, leveraging new platform & robust technology
- Complex monolithic API Architecture hampers new payment business initiatives

IBM Value Proposition

- Garage Method and expertise in cloud-native approaches
- Deep integration expertise
- IBM Cloud Pak for Integration

Outcomes achieved

- **Integration Modernization Workshop and Services Design achieved in 3 weeks**
- **Implementation of new containerized microservices-based design in 4 weeks**
- **Number of cores required dropped from 102 (monolith) to 60 (microservices) and TPS increased from 120 (monolith) to 250 (microservices)**

Engagement Profile

- Design and Implementation follow Garage Method
- Existing Monolithic Payment Application refactored to microservices architecture.
- Single Payment Application designed to cover all use cases
  - Single Payment(NEFT, RTGS, IMPS, NEFT, UPI)
  - Multi Payment
  - Batch Payments
  - Corporate Payment
- Microservices built in Node JS, Java, App Connect and DataPower.
- Databases redesigned for microservices.
- New Kubernetes-based development, QA, UAT and Prod environments built all with containerized ACE and API Management

Resources

- Video of this presentation for external consumption (2019 App Mod Technical Conference) https://www.ustream.tv/recorded/124074109

- Article Series Part 1https://medium.com/ibm-garage/the-steps-to-application-modernization-for-the-cloud-part-1-7ac07515dc16

- Article Series Part 2https://medium.com/ibm-garage/maps-for-the-journey-950153ed39ce

Thank You