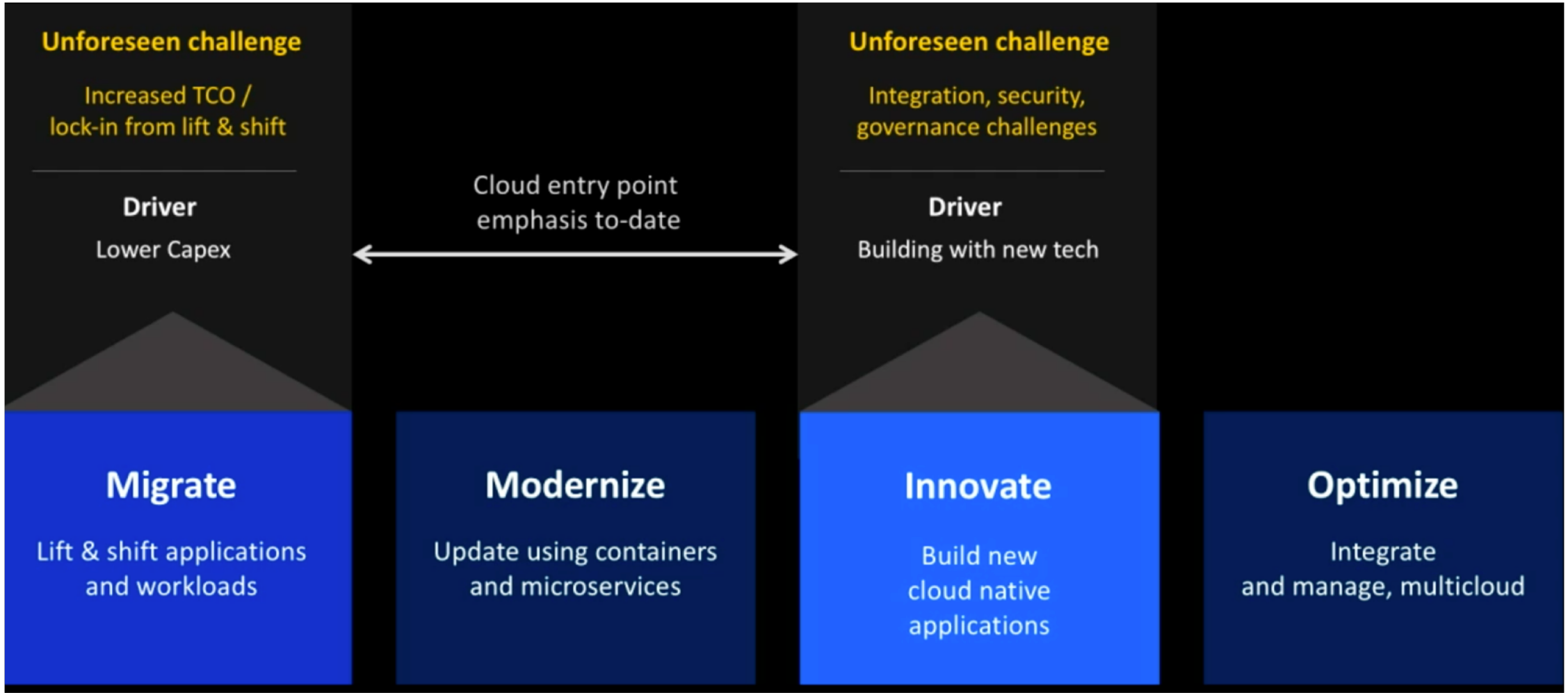
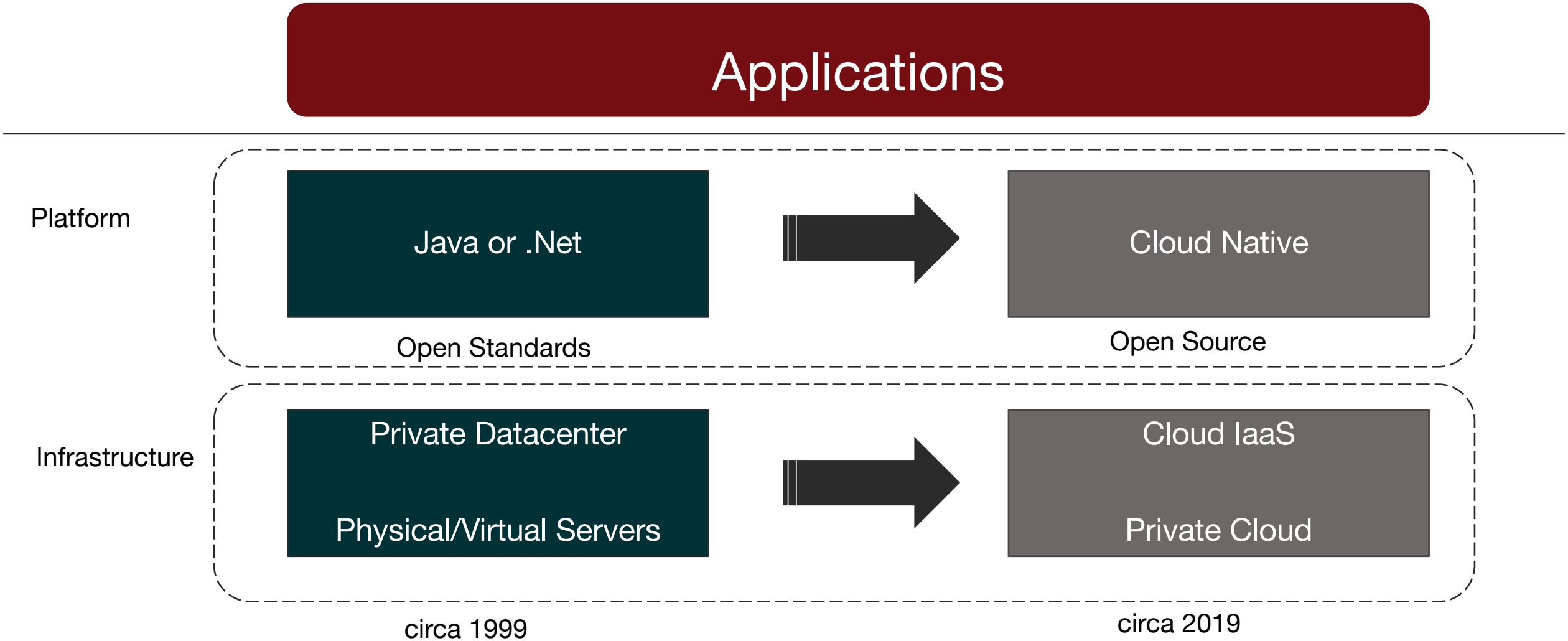
To date, fewer than 20% of workloads have moved to cloud. Why?

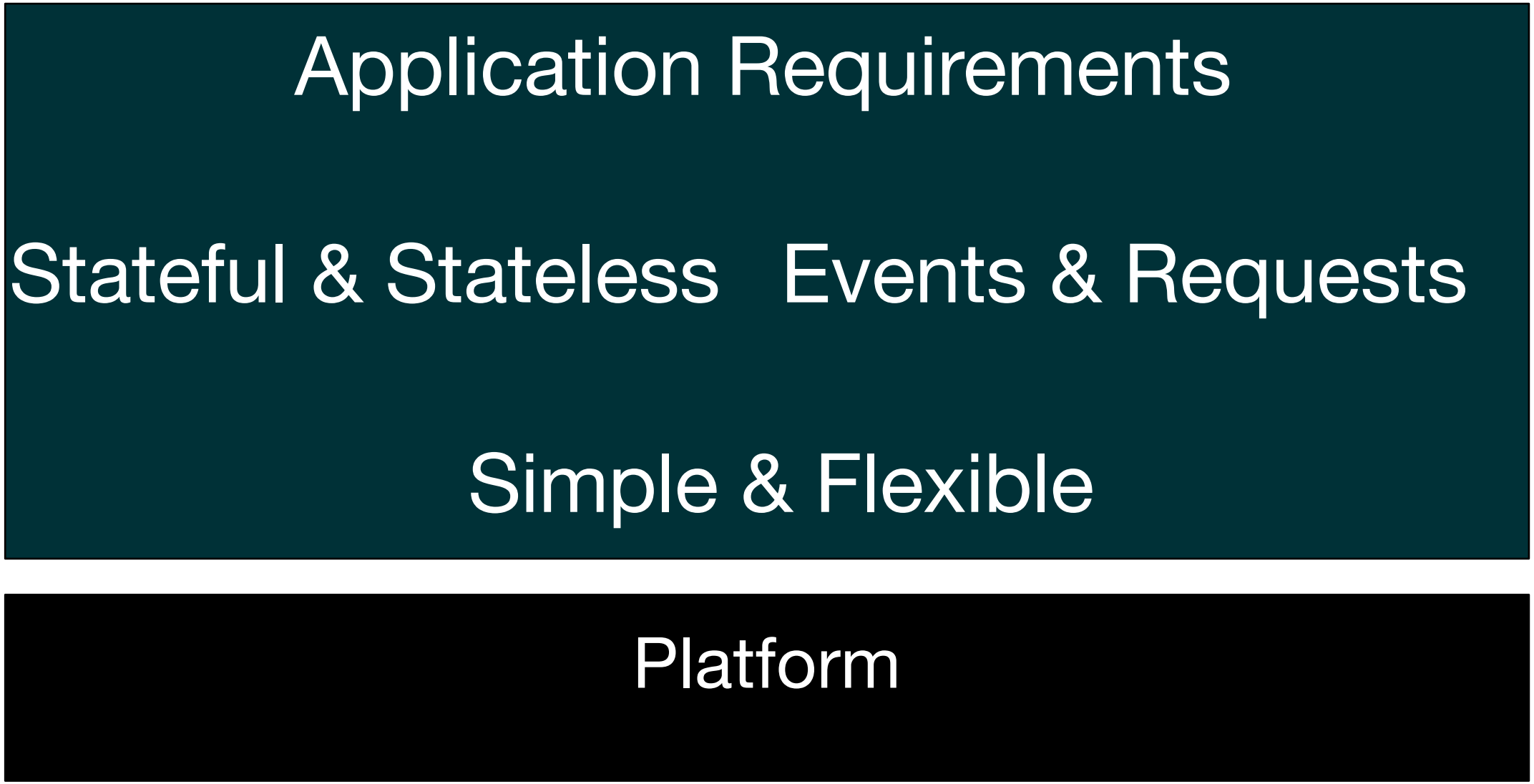


Evolution of the App Platform

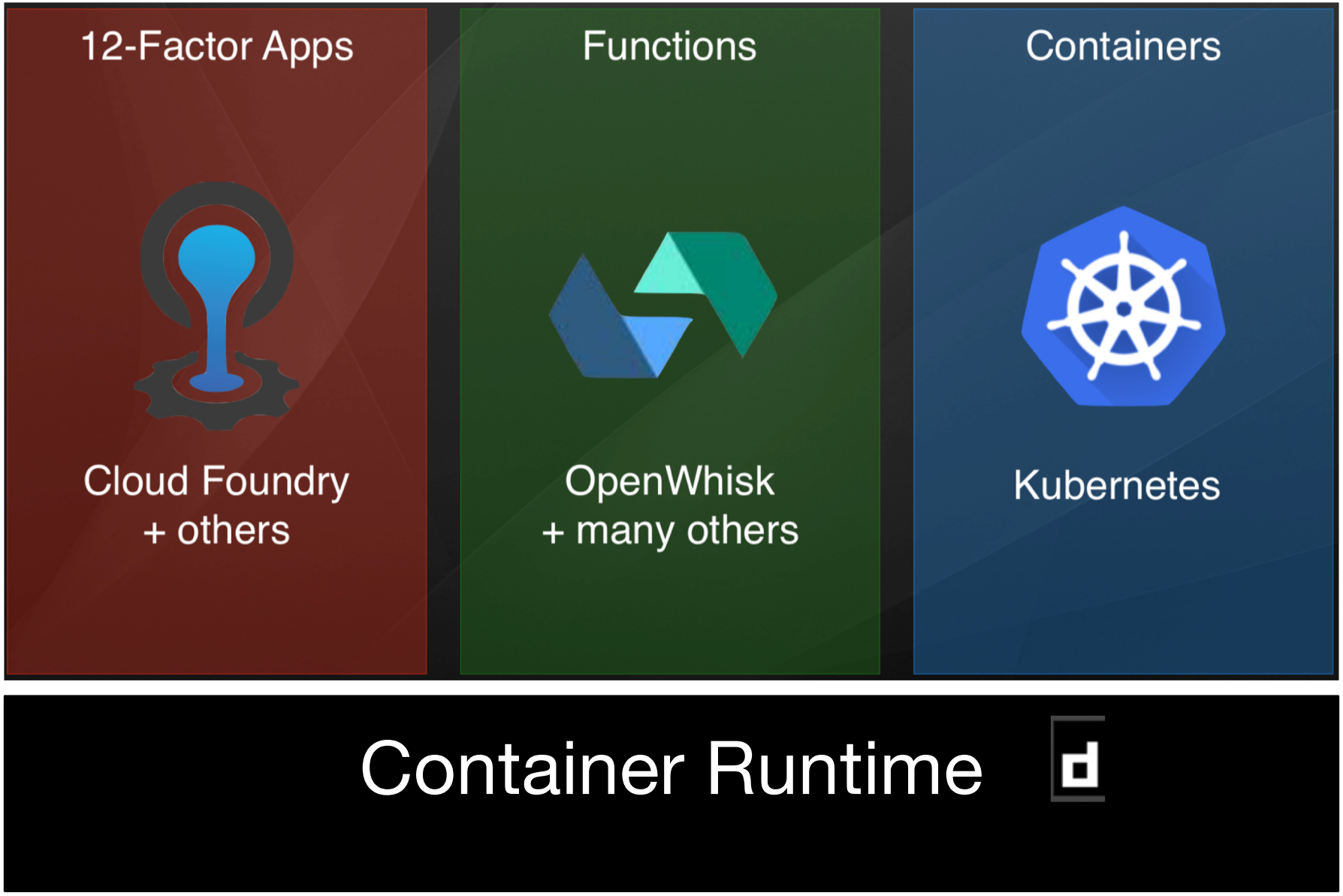


IBM CODE

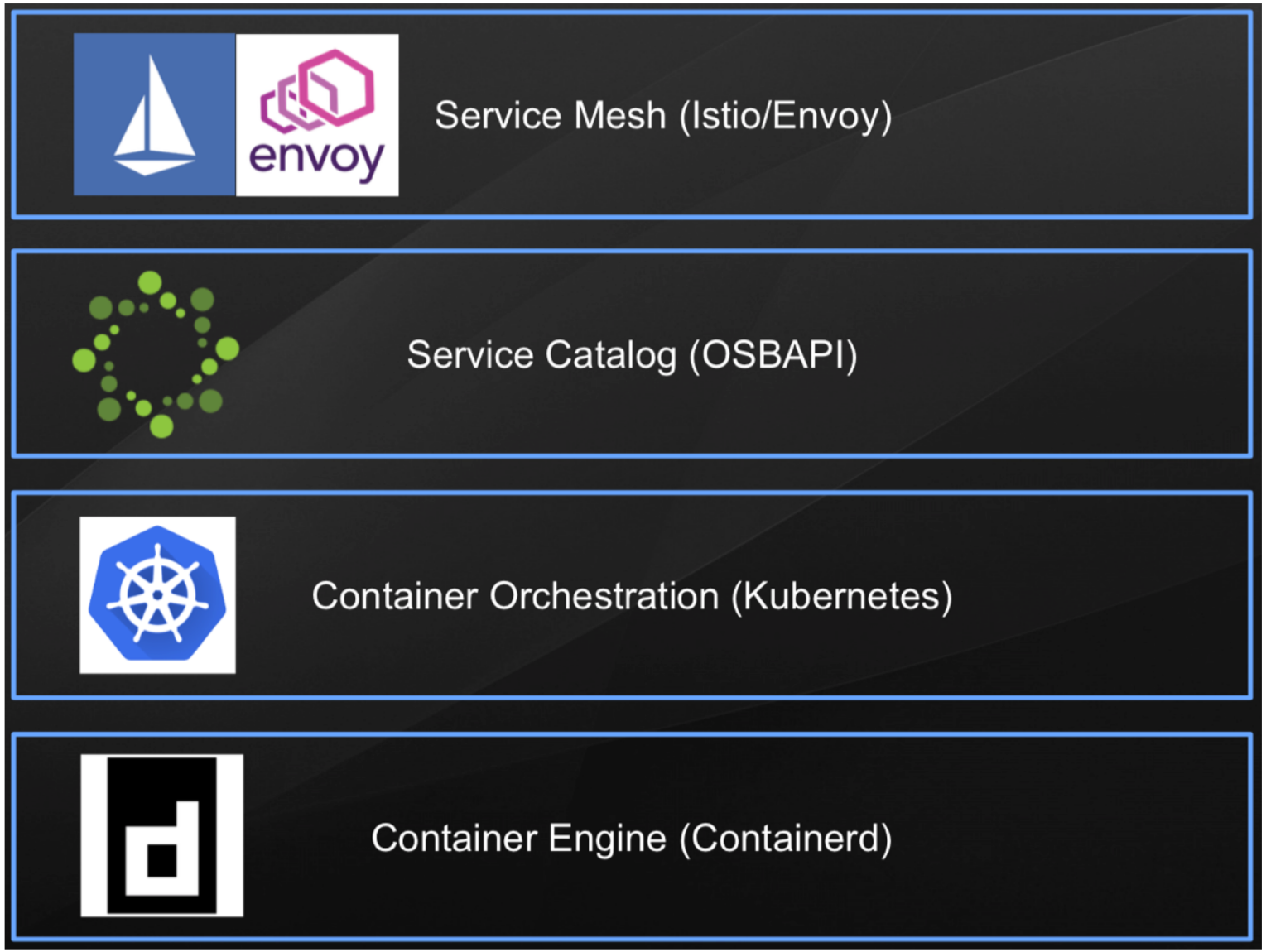
Tradeoffs in App Design



Three stacks for three styles



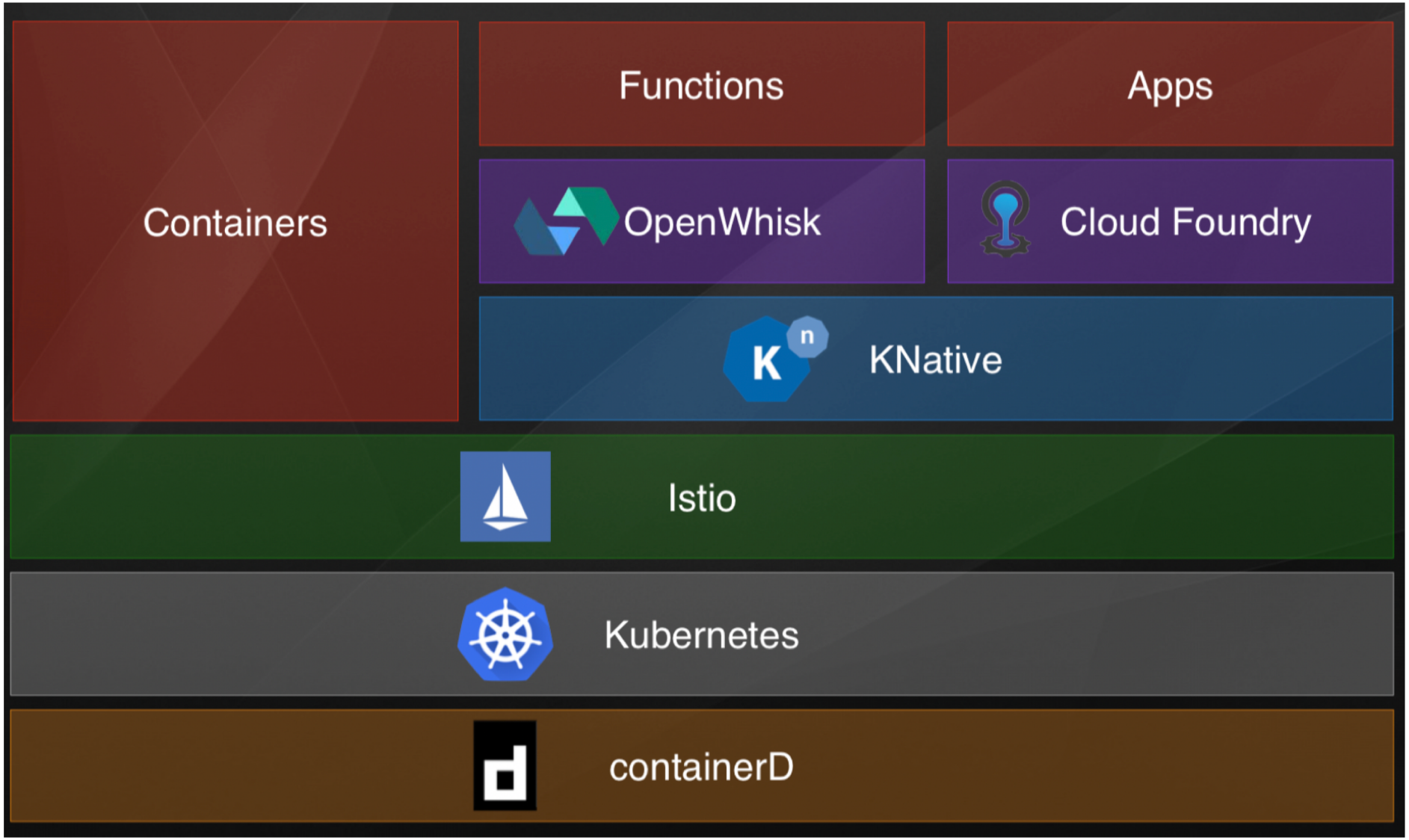
Common elements in the stack



Docker Swarm, Mesos, Nomad, Diego,…

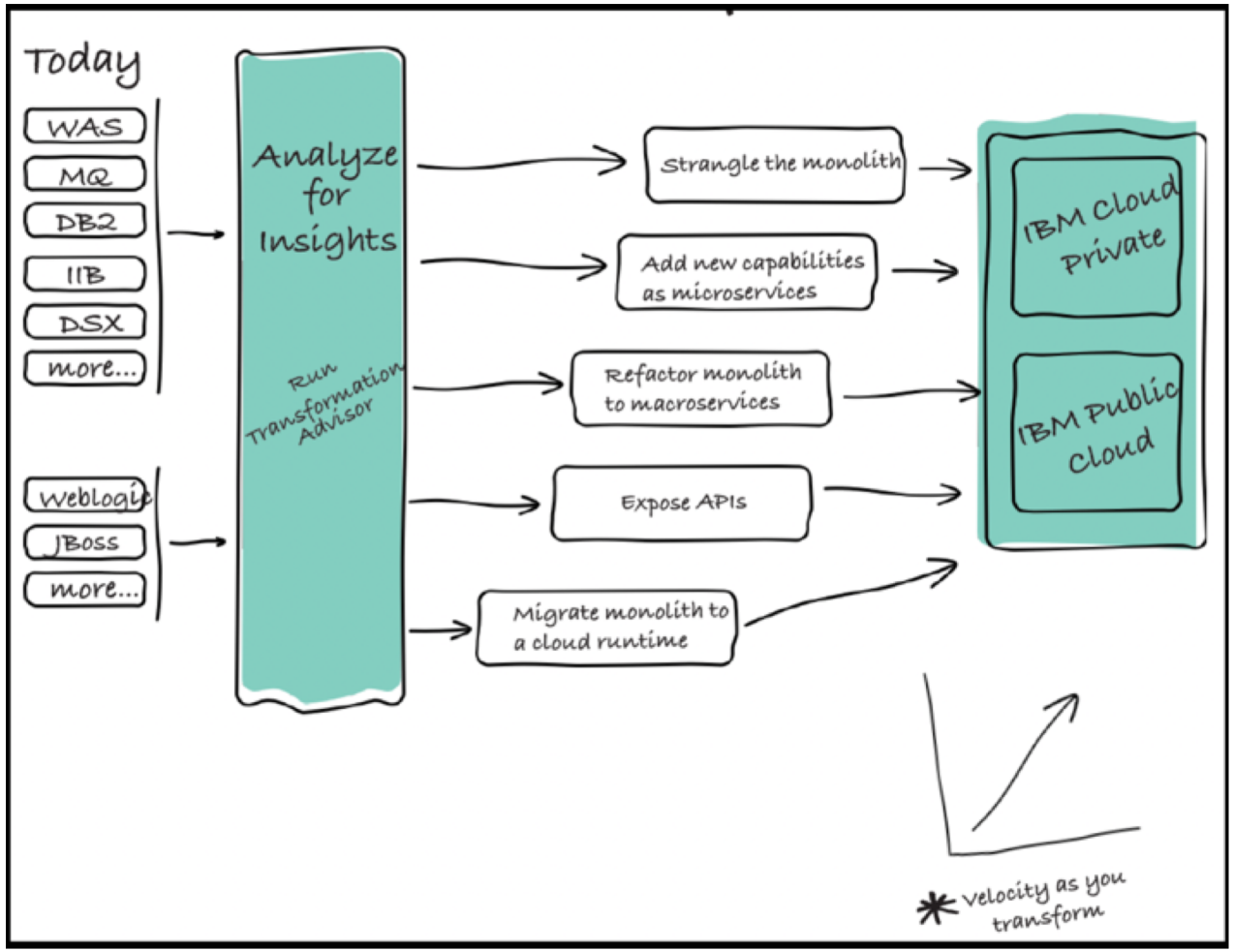
Docker, Garden, runc, Rkt, …

Uniform platform for containers, apps and functions



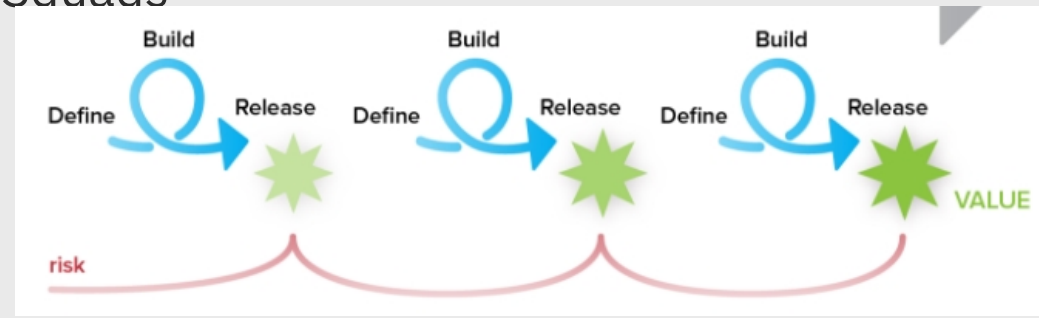
What Does App Modernization Mean Anyway ?

- Containerize an Existing Application / Workload ?
- Refactor Applications into Microservices ?
- Strangle Monolith over time with new Microservices ?
- Completely rewrite into new microservices ?
- Automate deployment ?
- Lift and Shift into a Cloud ?
- Expose Applications through API's ?
- Augment Old Code with new microservices ? APIs / Services (AI, Data Science) ?
- What About My Data !!!!!!!!



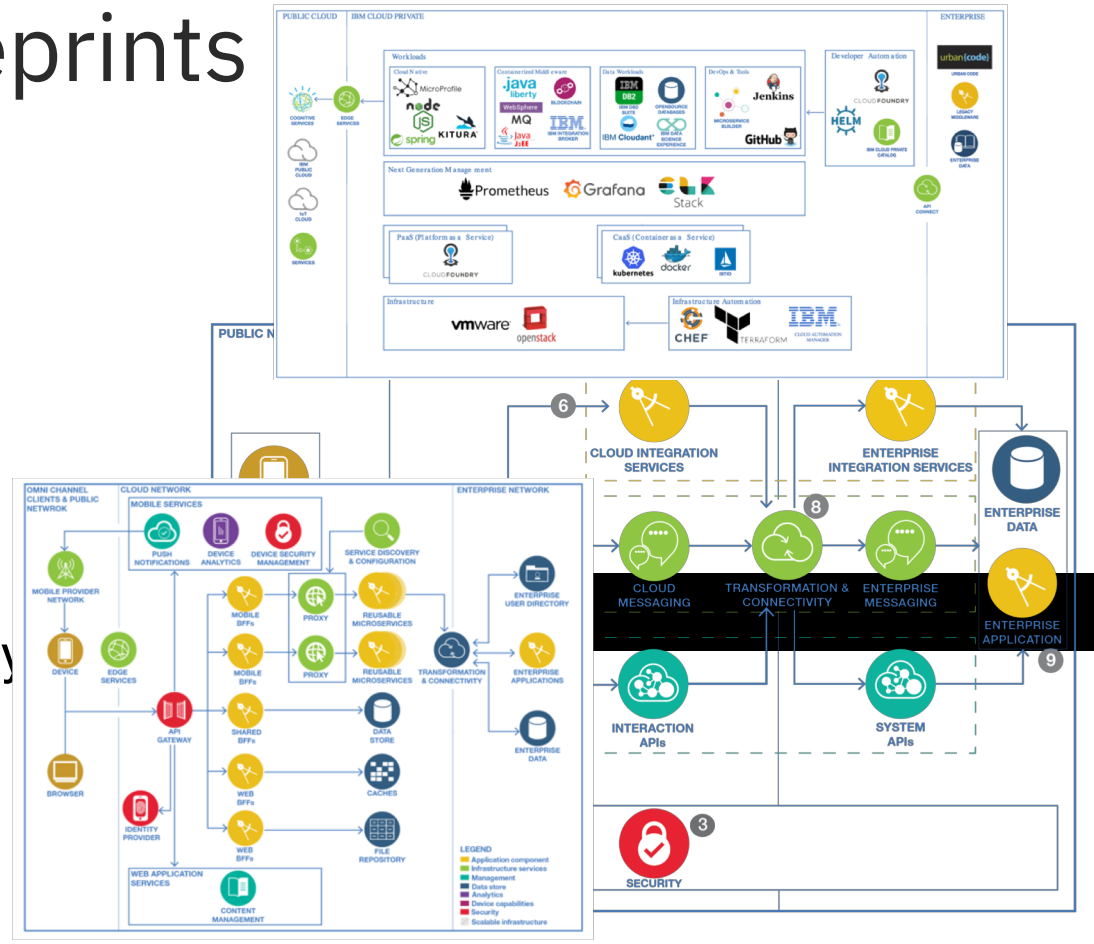
Modernization is difficult without Some Culture Change

Traditional IT	Cloud
Planned, process-oriented	Iterative, agile
Goal to look good - „it‘s not me“	Goal to learn - blame-free, no finger-pointing
Saviour syndrom, „heros“	Learning organization
Goals per business unit	Common goals across all units
Expertise	Collaboration, Sharing
Be protective of information	Transparency
Be comfortable in static environment	Be comfortable with changes and dynamics
Risk-averse	Risk-receptive
Reaction to challenge : helplessness	Reaction to challenge : resilience
Default attitiude is „no“	Default attitude is „yes“
Silos (Dev -> Admins)	Squads



App Modernization Requires Blueprints

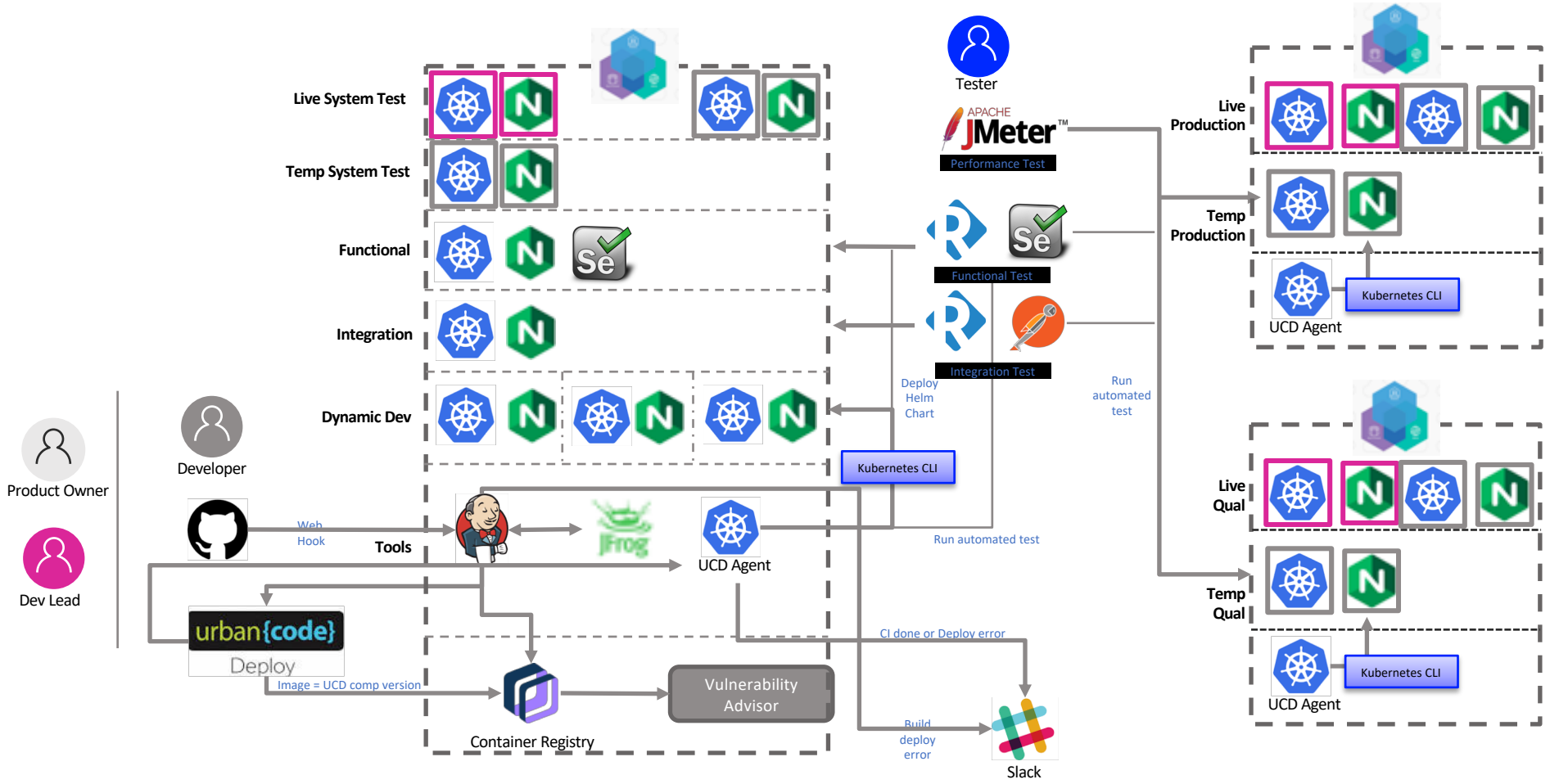
- Provide blueprint for building apps on the cloud
- Reference architectures define the basic pattern, while implementations provide specific technology, practices, and tool choices to build and deploy that pattern
- Categorized by style (eg. Microservices, Web, Mobile..) and for each a detailed view of key aspects (eg. Security Service Management..)
- Technology opinionated but still flexible
- Connected to implementation assets
- Allows organization to standardize on proven blueprints and repeat faster



https://www.ibm.com/cloud/garage/architectures/

https://ibm-cloud-architecture.github.io/deliverables/application-modernization.html

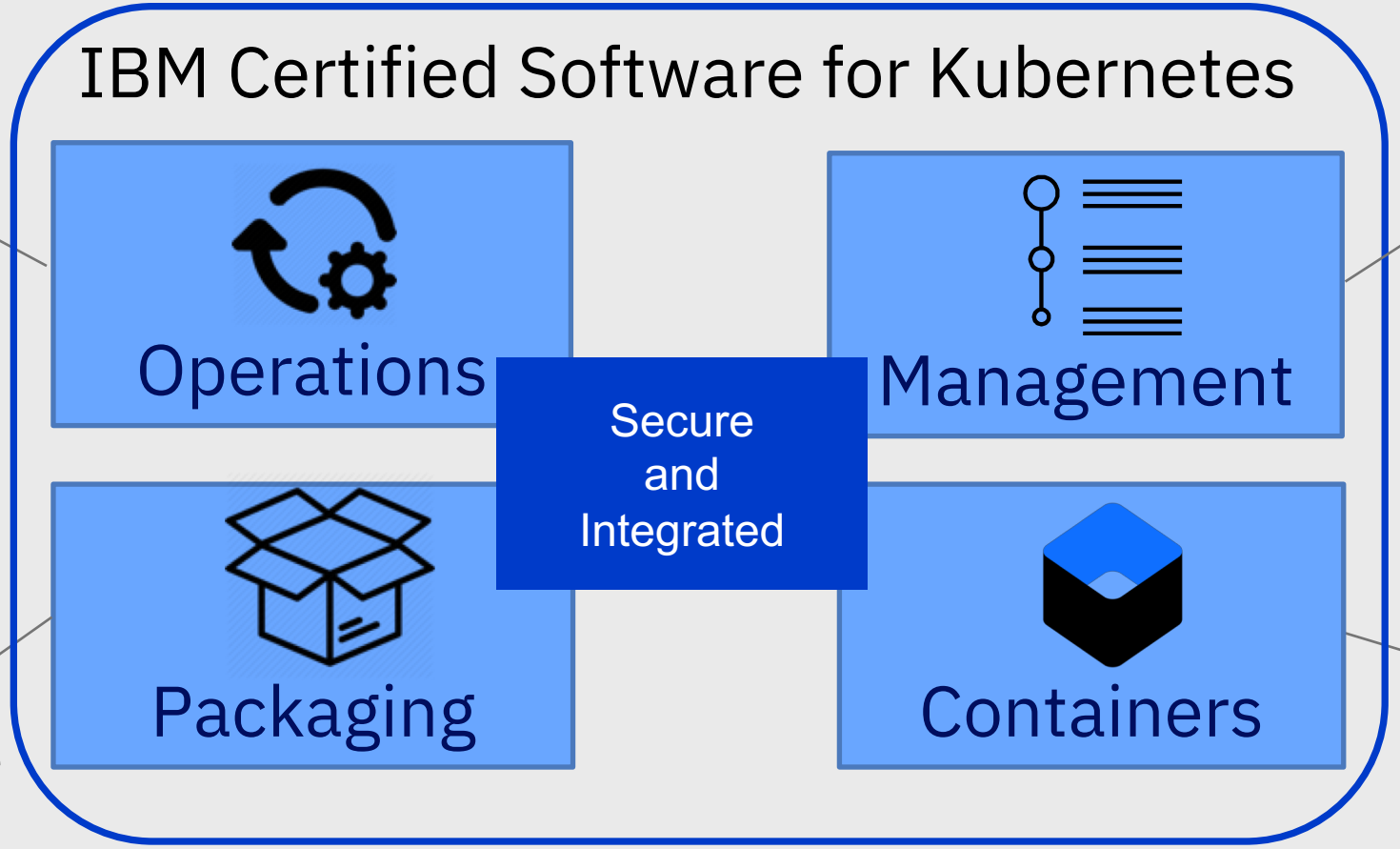
App Modernization Requires Automation



App Modernization is more than wrapping your app in a Docker Image

Upgrade & Rollback
consistent across all IBM Software built for Kubernetes

Enterprise Ready and Simple to Deploy
-Orchestrated by the product experts
-Integrated catalog experience
-Open standards packaging
-Secured by ICP IAM



Pre-integrated:
-Logging (Debug)
-Monitoring (Alerting)
-Usage Metering
-License Management

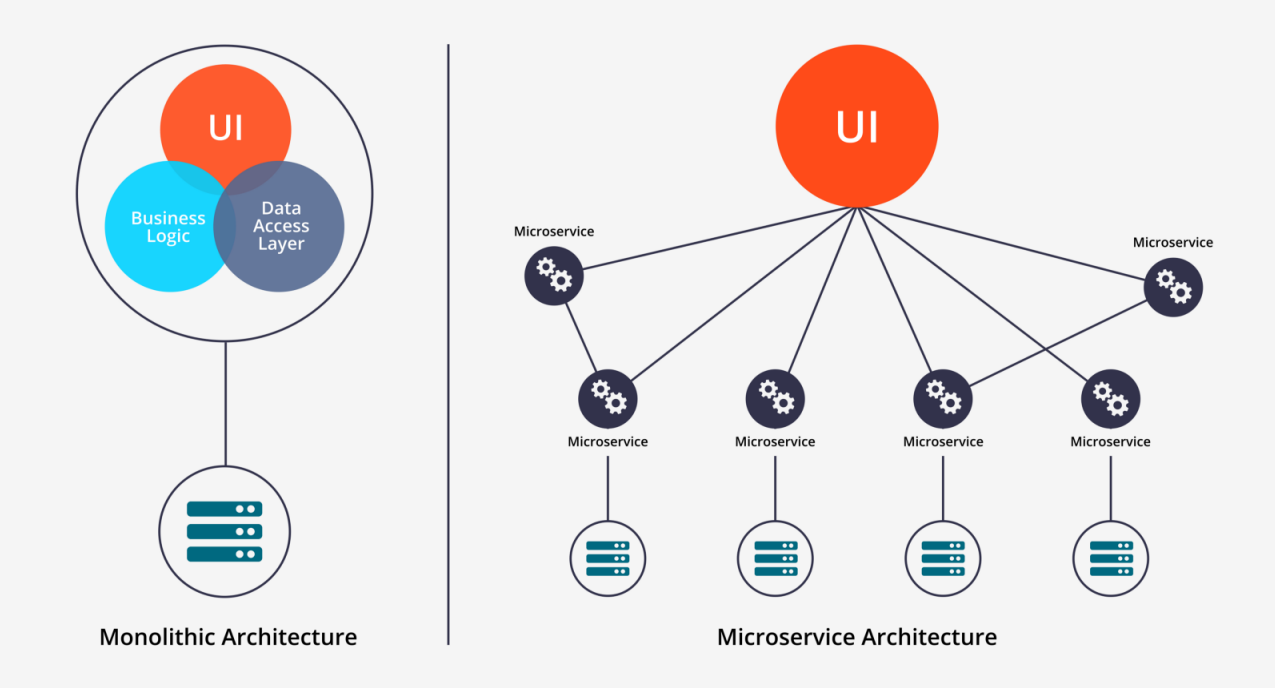
-**Scanned for Vulnerabilities**
-**Extendable** to Redhat Certified with RHEL base image

The entire package is certified to support IBM Cloud Private, not the individual pieces

IBM Cloud Private
The Universal Platform for IBM Middleware and Containers

Architecture Principles

Microservices



An engineering approach focused on decomposing an application into single-function modules with well defined interfaces which are independently deployed and operated by a small team who owns the entire lifecycle of the service.

Microservices accelerate delivery by minimizing communication and coordination between people while reducing the scope and risk of change.

Microservices Fatigue ?

Stats

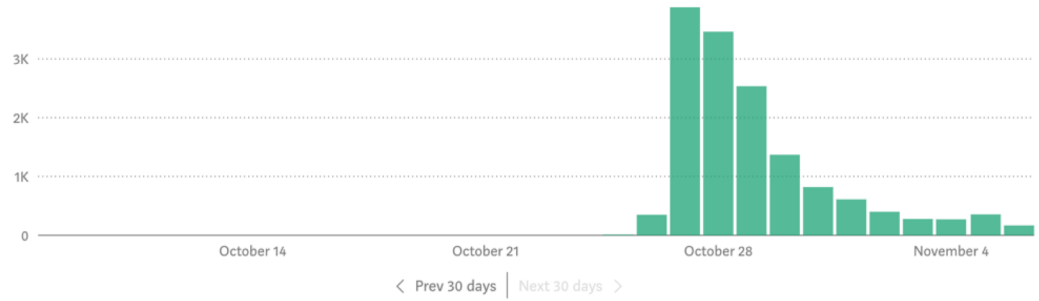
Click story below to view in chart

Learn more about using stats

14,521
Views (30 days)

8,319
Reads (30 days)

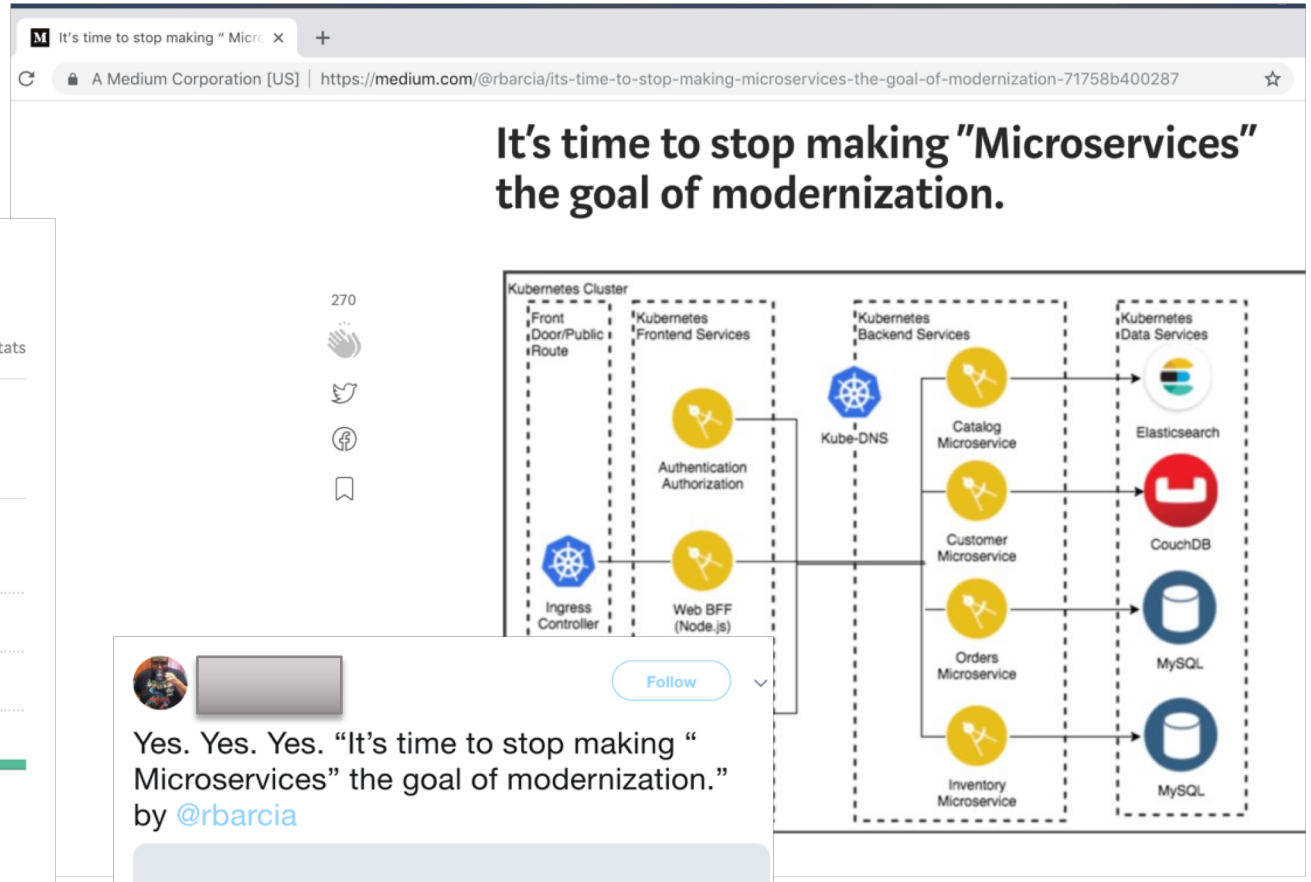
77
Fans (30 days)



Prev 30 days | Next 30 days

Stories Responses Series

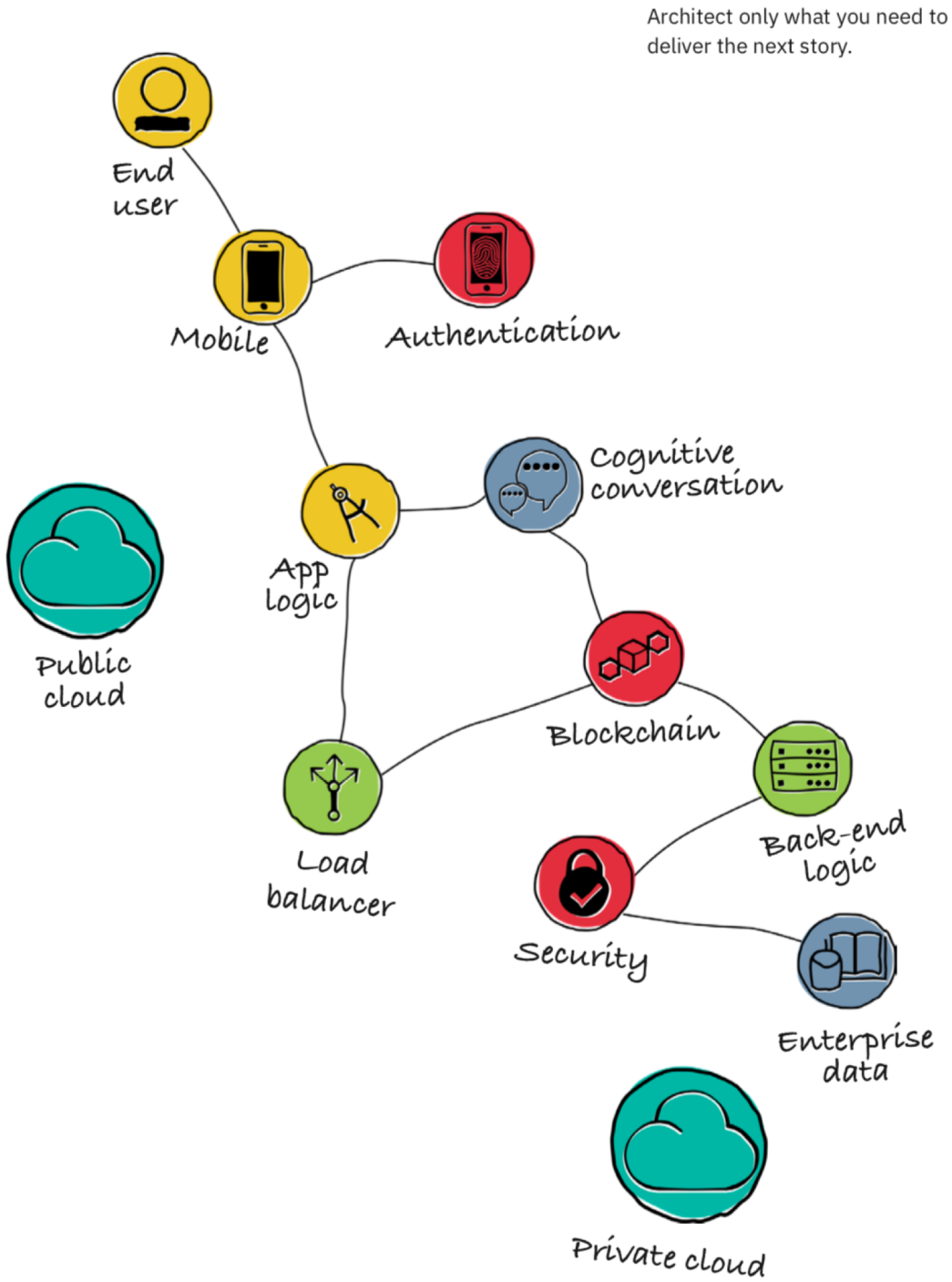
Date ↑	Views	Reads	Read ratio	Fans
It's time to stop making " Microservices" the goal of... 2 min read · View story · Details	14.5K	8.3K	57%	77



The 12 Factors

12 factor App Architecture

- Codebase
- Dependencies
- Config
- Backing Services
- Build, release, run
- Processes
- Port binding
- Concurrency
- Disposability
- Development/Production Parity
- Logs
- Admin Process



https://developer.ibm.com/integration/blog/2017/04/16/12-factor-integration/
https://12factor.net/

Microservice versus Monolith

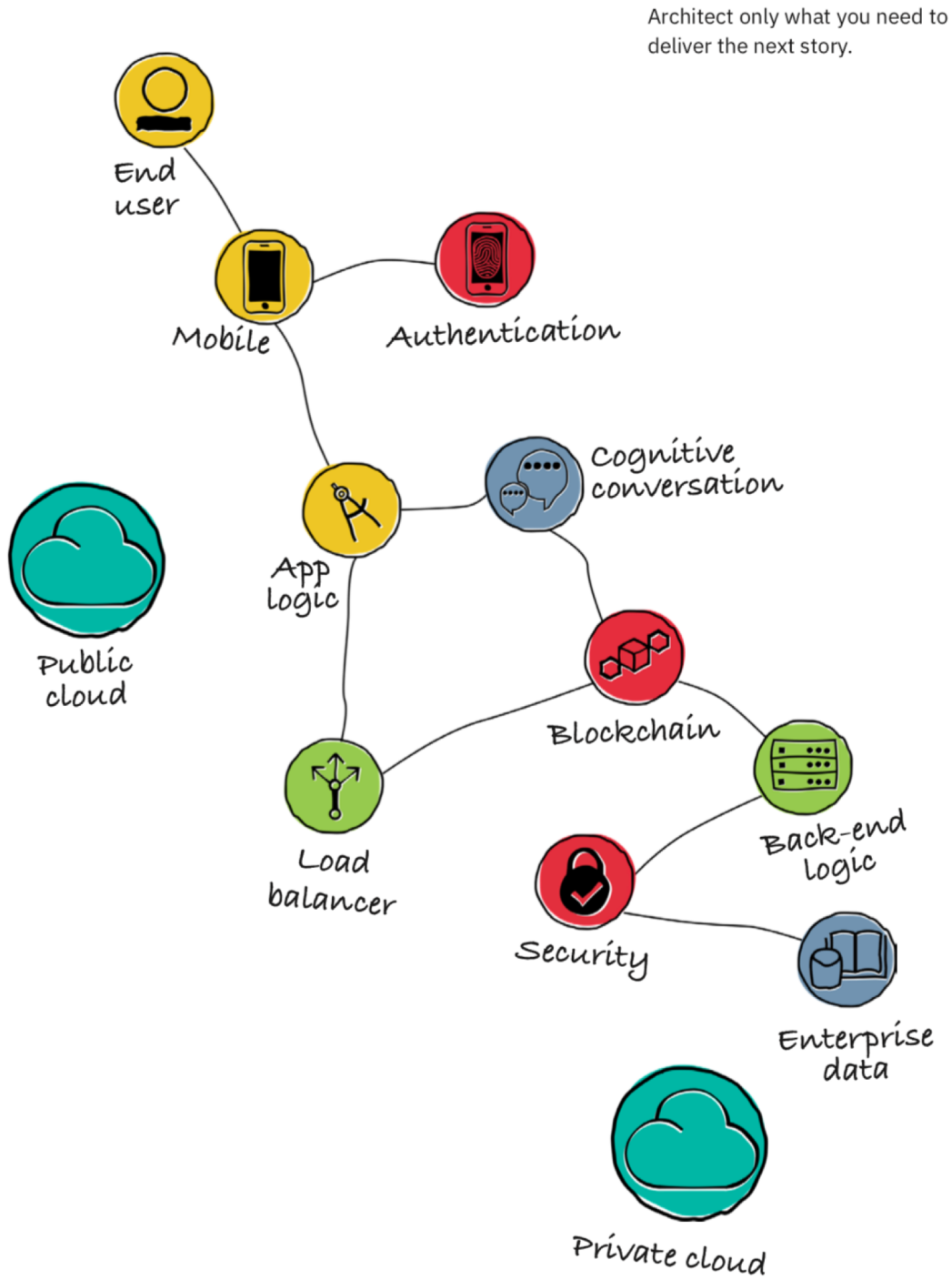
	Monolith	**Microservice**
Architecture	Built as a single logical executable (typically the server-side part of a three tier client-server-database architecture)	Built as a suite of small services, each running separately and communicating with lightweight mechanisms
Modularity	Based on language features	Based on business capabilities
Agility	Changes to the system involve building and deploying a new version of the entire application	Changes can be applied to each service independently
Scaling	Entire application scaled horizontally behind a load- balancer	Each service scaled independently when needed
Implementation	Typically written in one language	Each service implemented in the language that best fits the need
Maintainability	Large code base intimidating to new developers	Smaller code base easier to manage

Just enough architecture

Architect only what you need to produce your MVP

TIMING IS EVERYTHING

- **Reduce risks**
- **Scope your MVP**
- **Avoid rework**



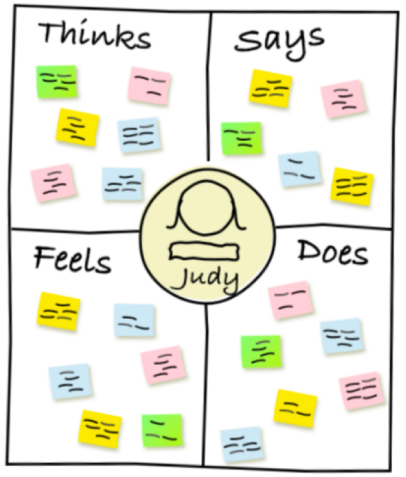
Enterprise Design Thinking

Enterprise Design Thinking helps you focus on your users and their needs to deliver more useful, usable, and desirable solutions.

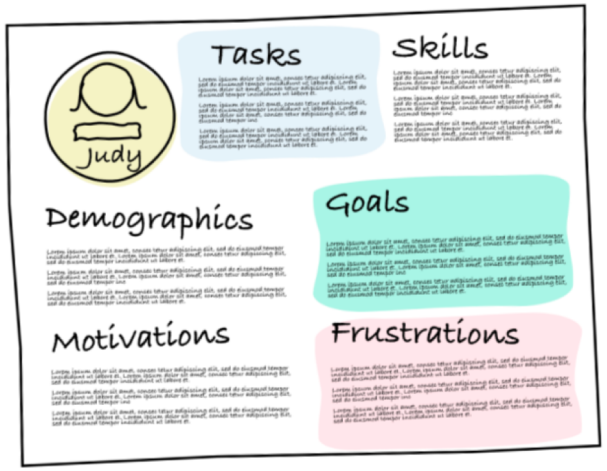
WOW YOUR CUSTOMERS!

- **Understand your users**
- **Define an MVP**
- **Align the team with playbacks**

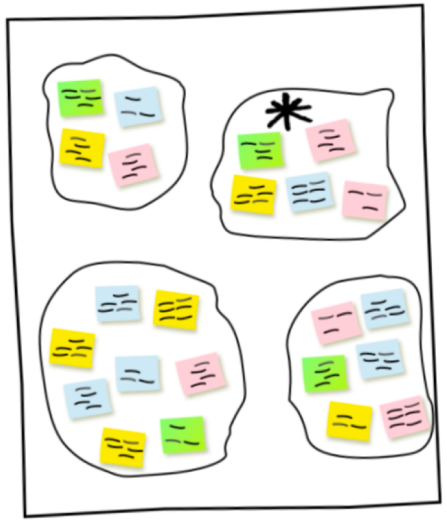
Know your audience and meet their needs faster than your competition.

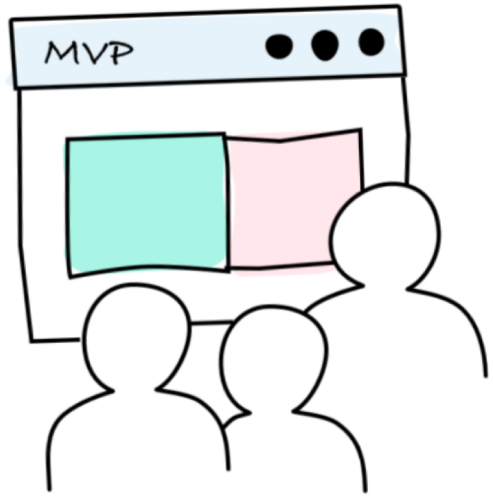


Empathy Maps



Personas / Sponser users


Ideation



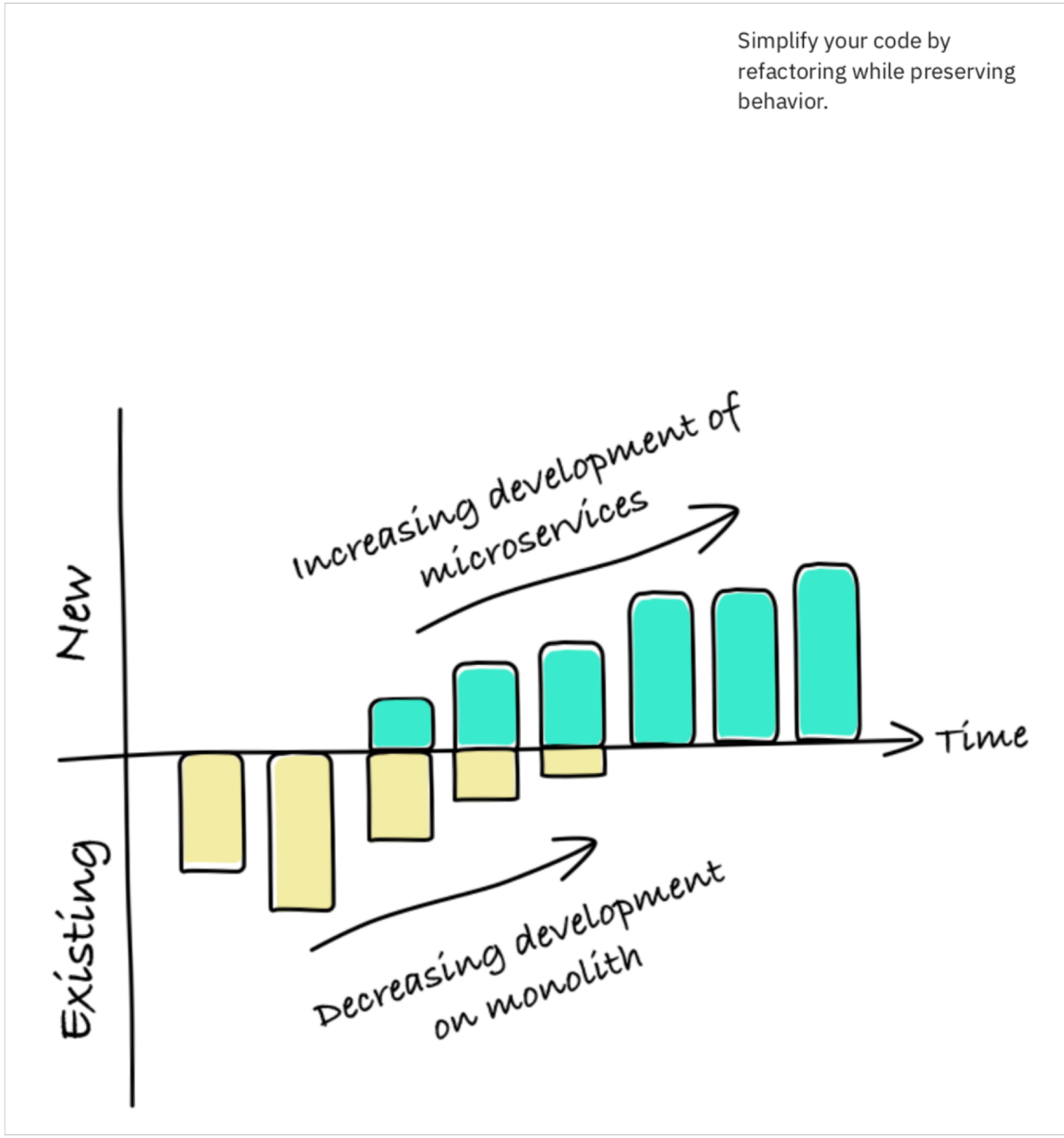
Playback

Simplify through refactoring

Refactor that code to make it simpler, while ensuring that it preserves the behavior and passes your automated tests.

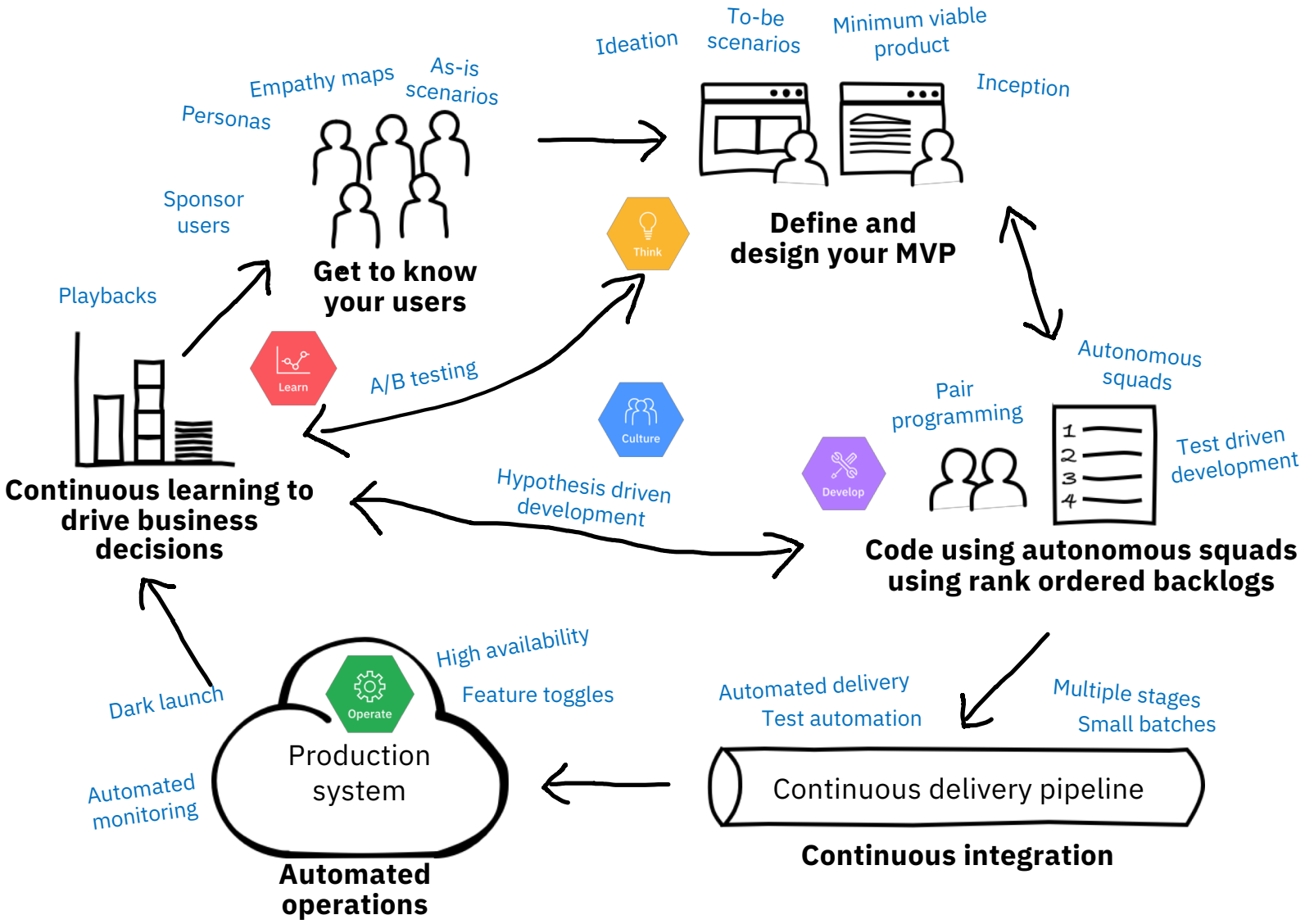
SIMPLIFY, SIMPLIFY, SIMPLIFY!

- **Don’t refactor on speculation**
- **Recognize the signs**
- **Automate tests**



Minimum Viable Product

(MVP)

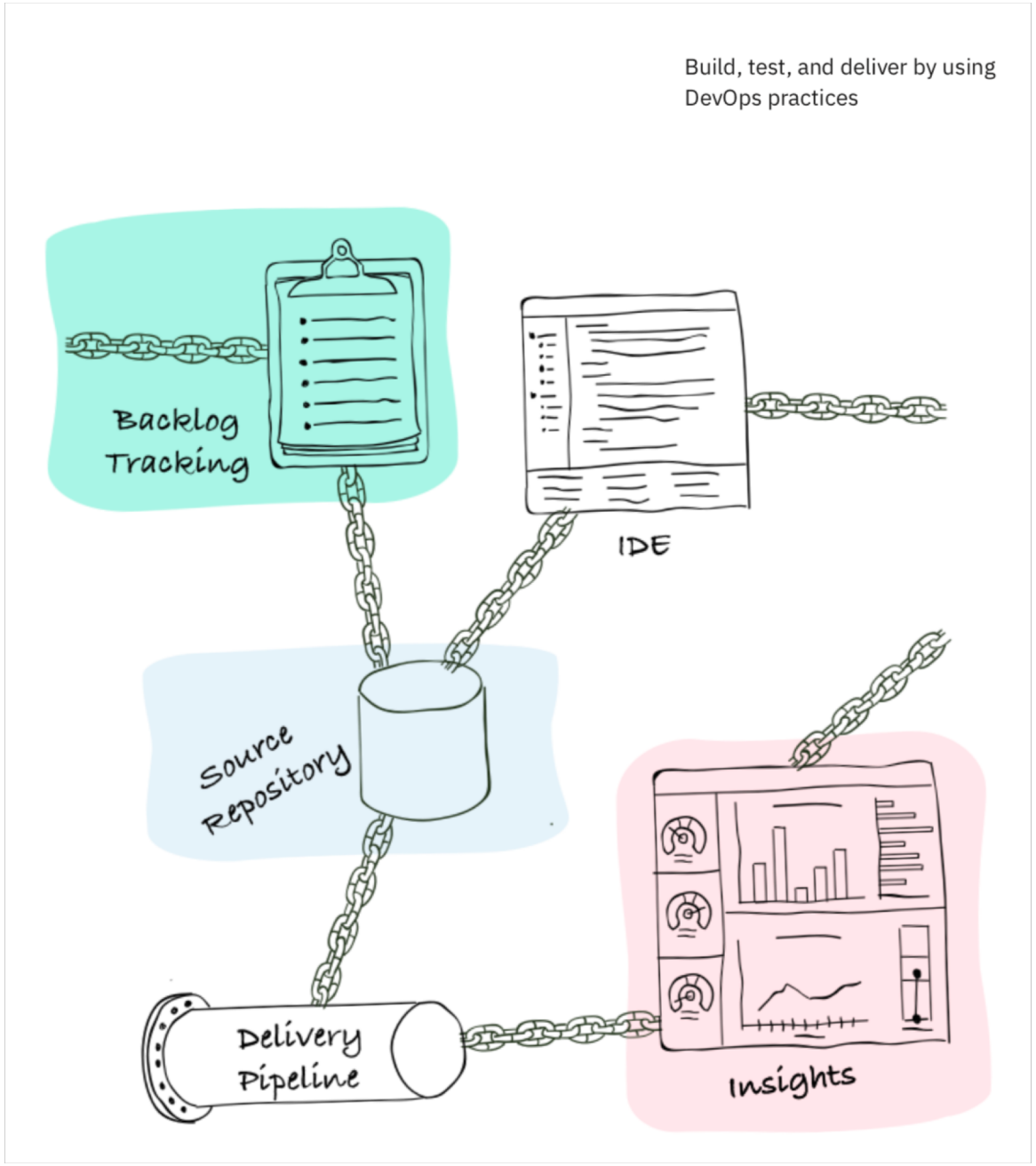


Continuously improve your MVP

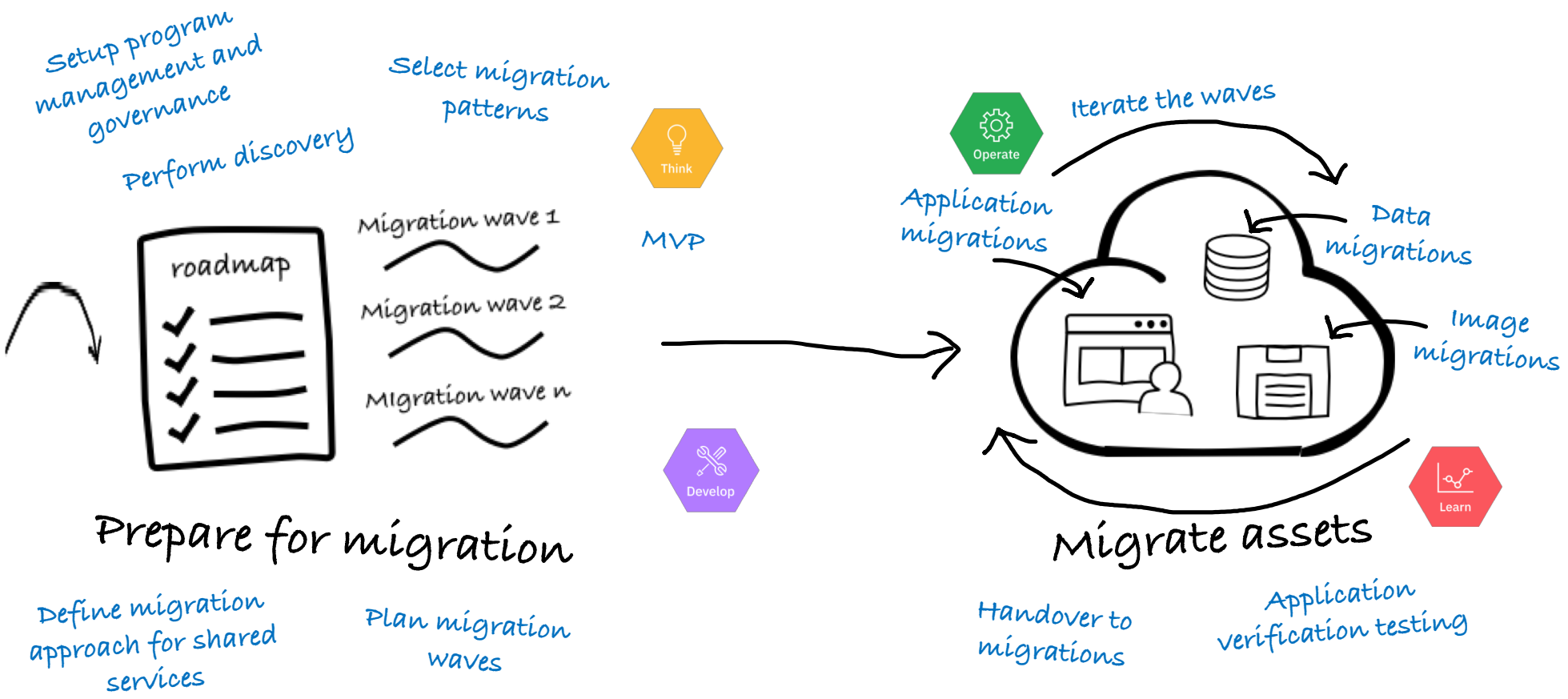
Continuous iterate to deliver quality production ready code.

INTEGRATE IT. BUILD IT. DEPLOY IT.

- **Create an integrated DevOps toolchain**
- **Continuously integrate and deliver using a delivery pipeline**
- **Automate deployments**

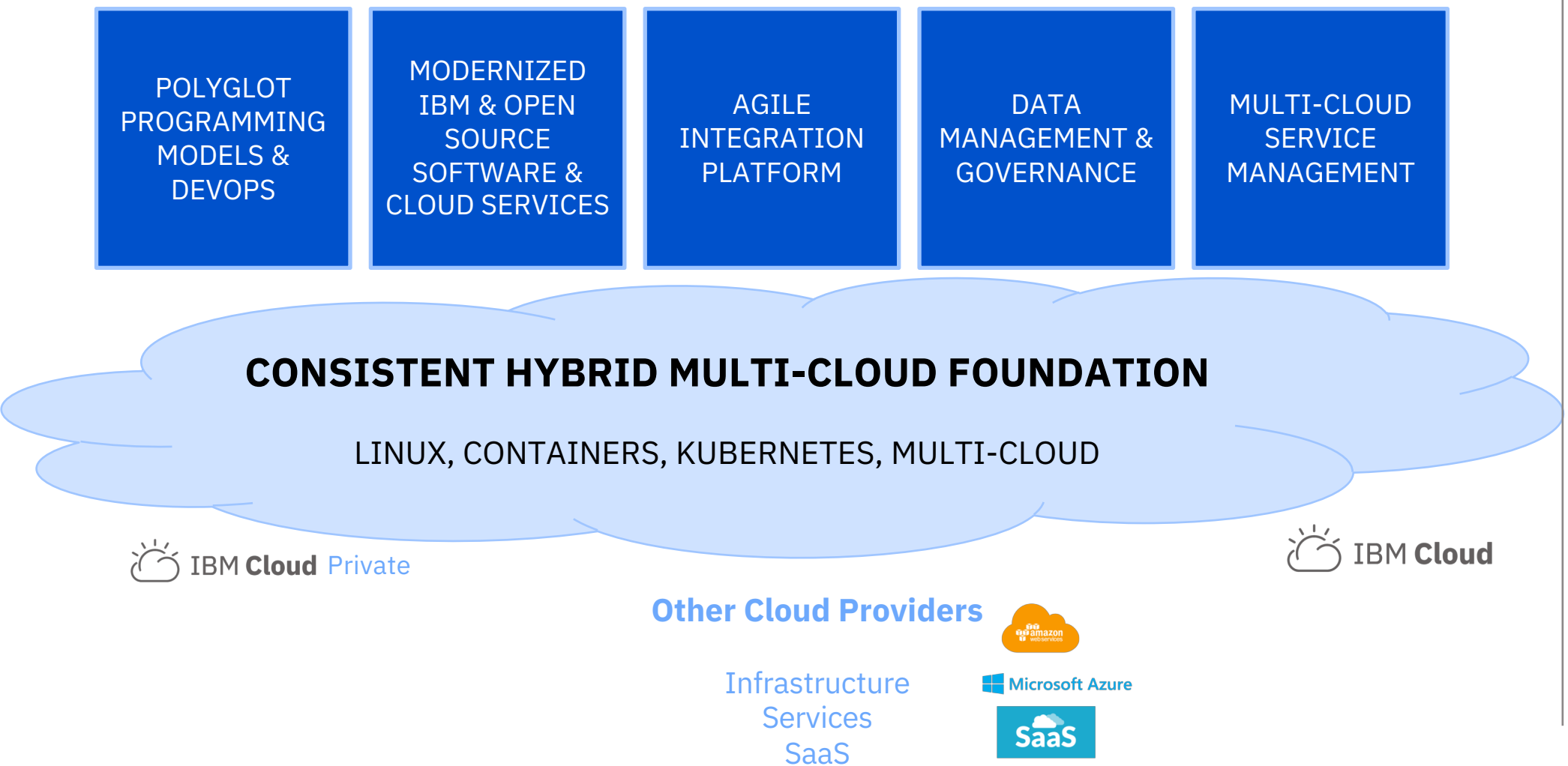


Migration Workflow



IBM’s Approach to Application Modernization

An Integrated Enterprise Hybrid Cloud Platform



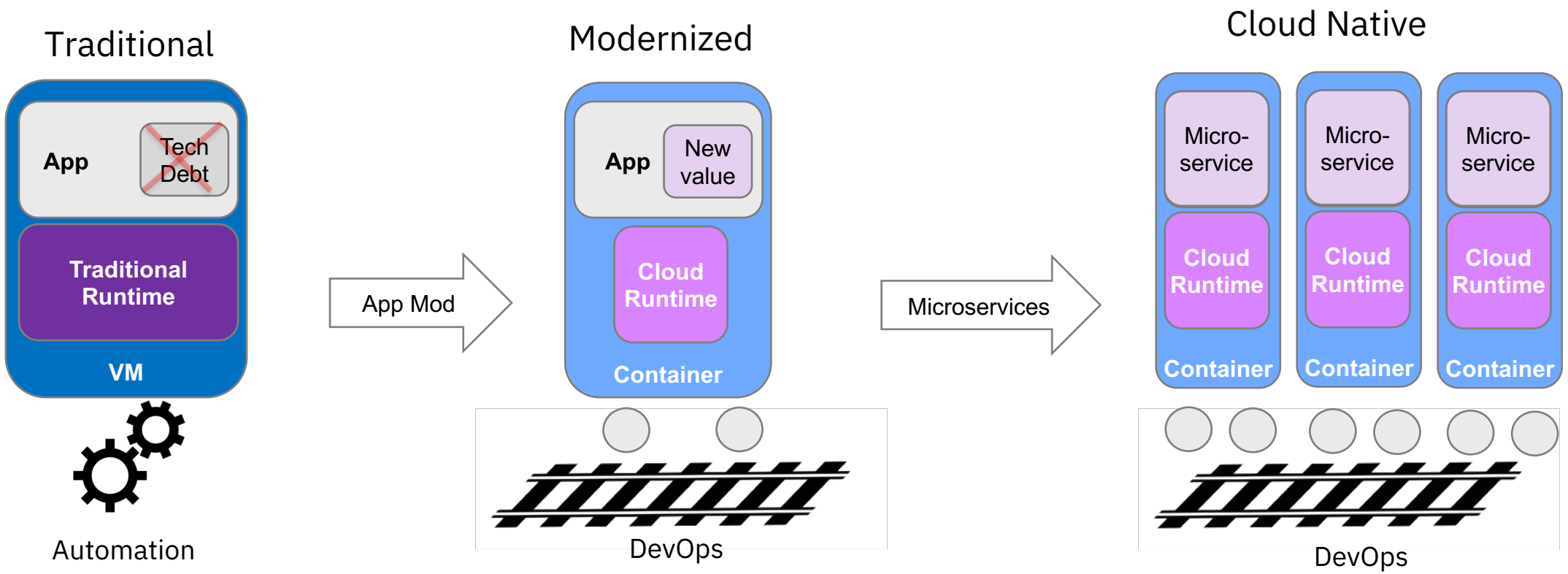
Migration & Transformation Tools

Methods & Reference Architectures

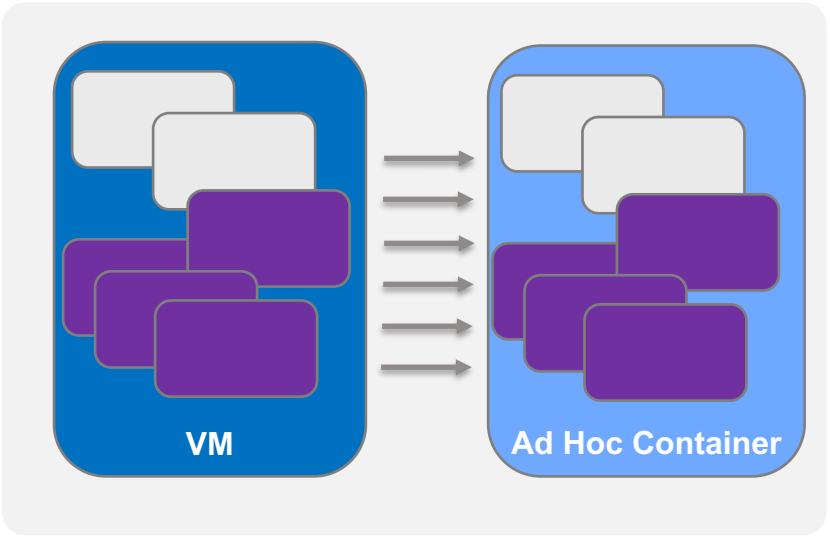
Innovation Garages

Professional Application & Managed Services

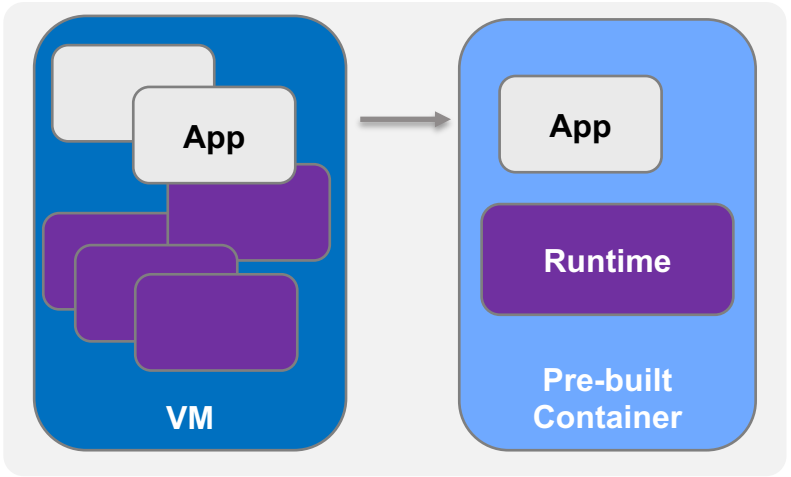
Traditional Java Application Modernization



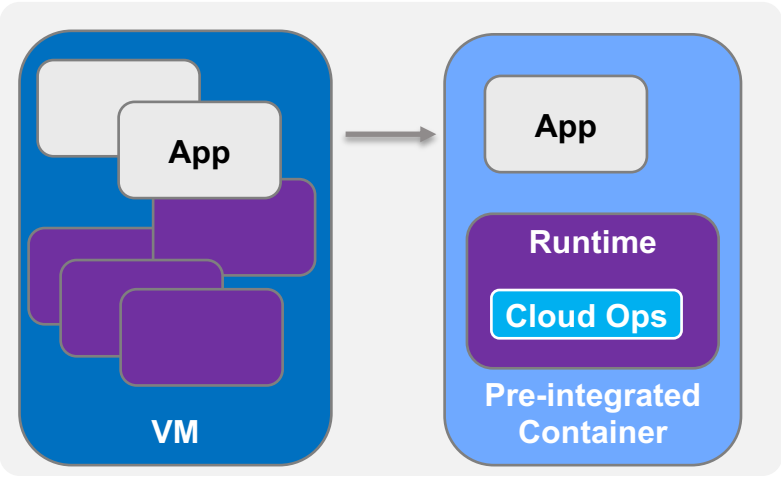
Containerization Approaches



“Lift & Shift” Containerization - DIY

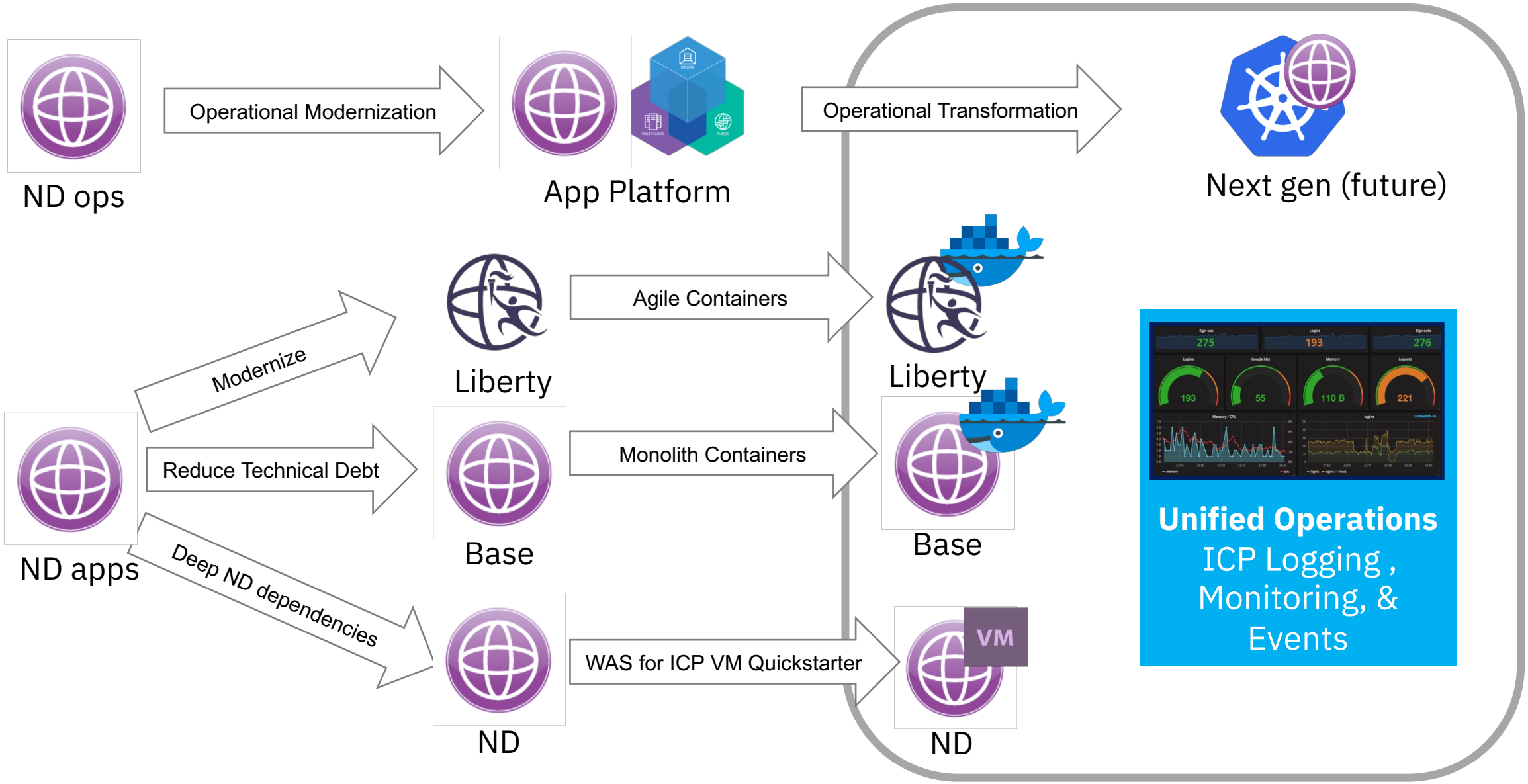


Vendor-provided containers



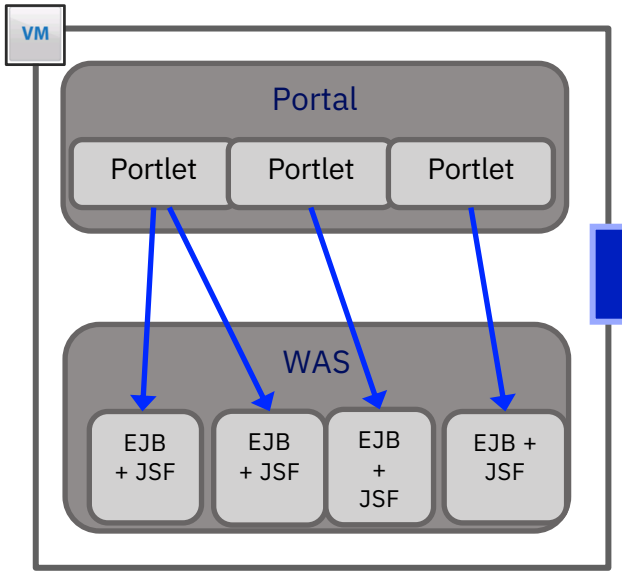
Built-in opinionated Cloud Ops

AppMod for WebSphere



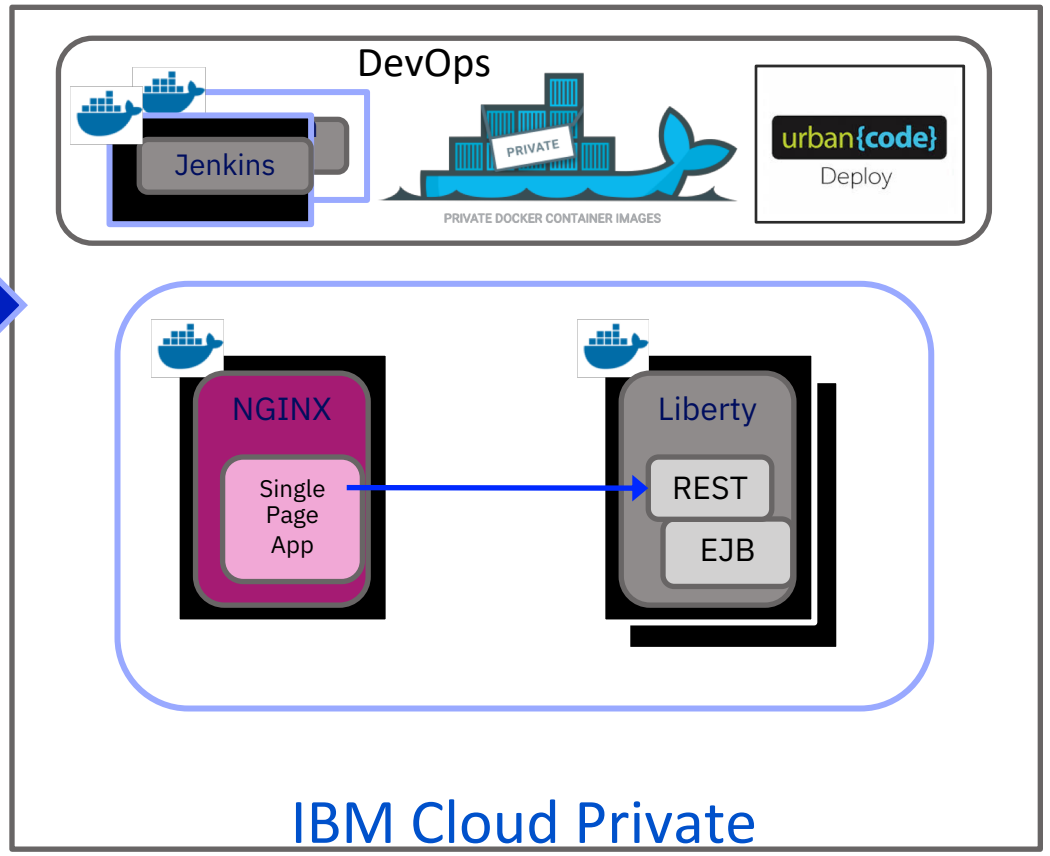
Modernize Mission-Critical Applications

Large US Insurance provider needed to modernize mission critical application for managing insurance claims and processing



Modernization Journey

- Modernize EJBs to run on Liberty and with REST interface
- Create new frontend
- Automate CI/CD with Jenkins and IBM UrbanCode



BENEFITS:

#1 - Modern, supported runtime with zero migration

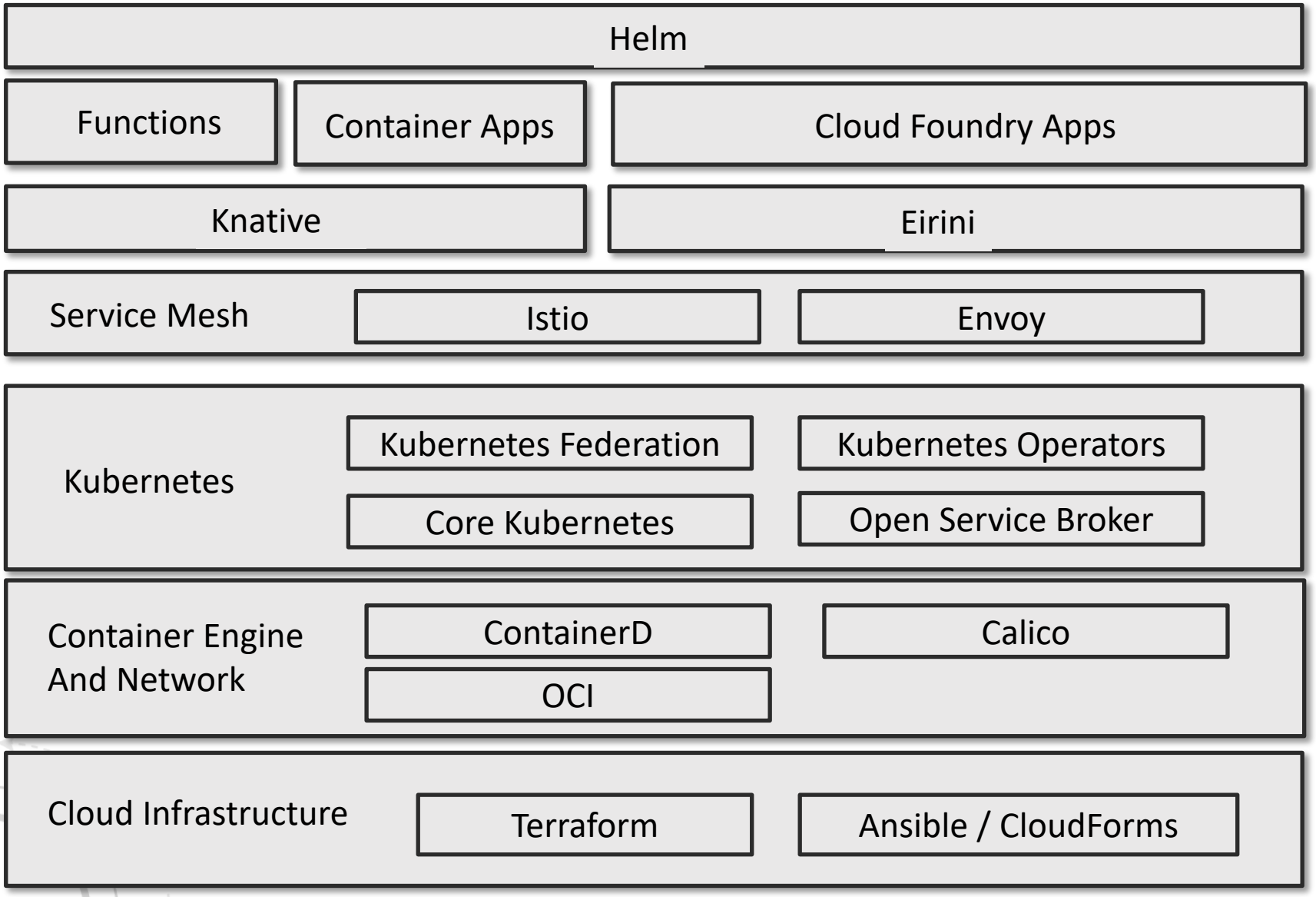
#2 - Automated provisioning of test environments and automated testing

#3 - Reduced business risk by reusing EJB layer

Customer implementation led by IBM Cloud Garage

Open Source Projects and Communities

Priority projects Supporting IBM Hybrid Cloud Containerization and Kubernetes Everywhere Strategy



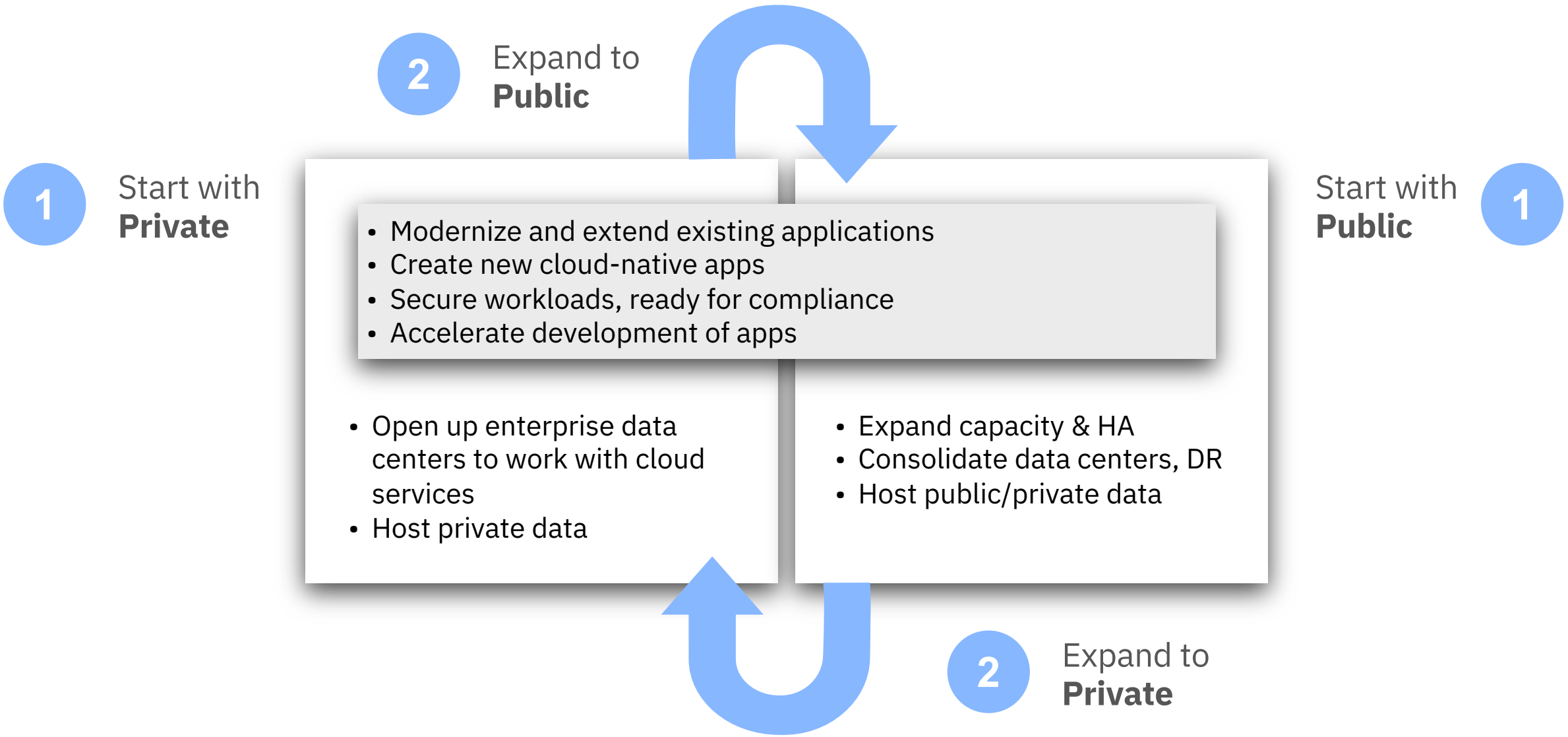
- **Helm:** Kubernetes templating and packaging

- **Knative:** Kubernetes serverless application middleware (serving, eventing, build pipelines, Functions)
- **Eirini:** Kubernetes container scheduling for CloudFoundry
- **Istio:** service mesh
- **Envoy**: Service Mesh Router

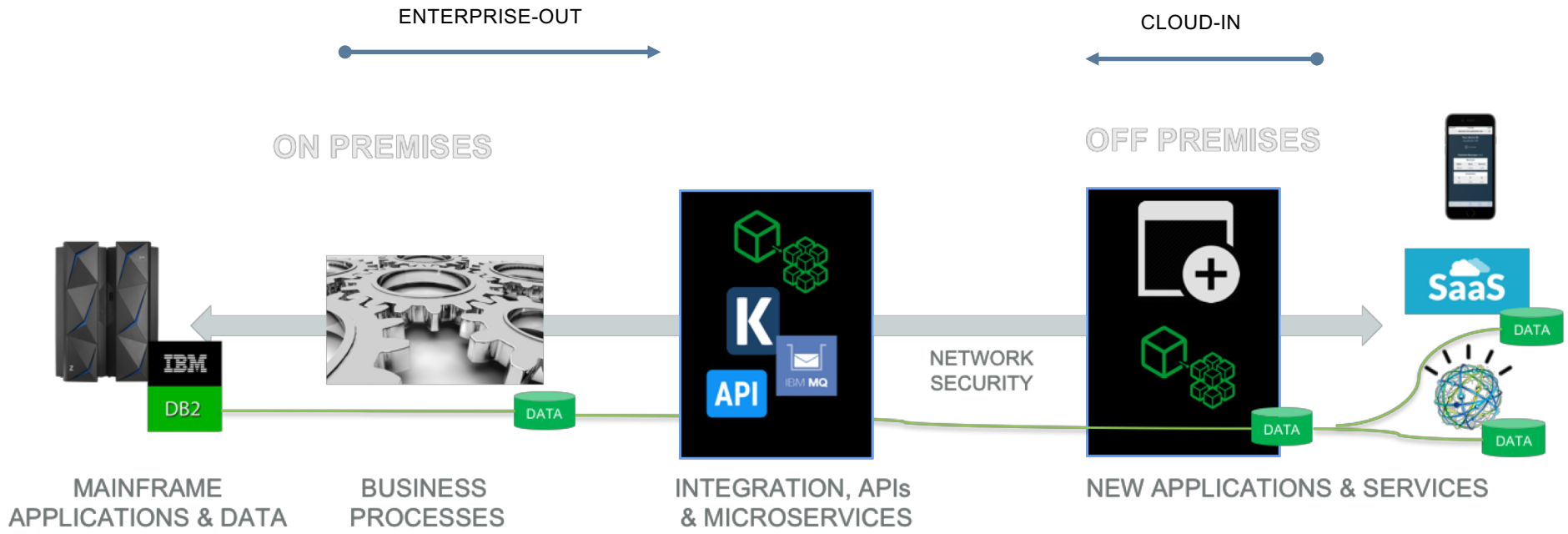
- **Kubernetes:** container orchestration
- **Federation:** multicluster deployment
- **Operators:** Package and deploy apps
- **OCI**: Container Engine
- **ContainerD**: Container Runtime
- **Calico**: Container Networking Overlay

- **Terraform:** Configure and deploy infrastructure as code
- **Ansible / CloudForms:** Infrastructure automation

Hybrid deployments with choice of entry points



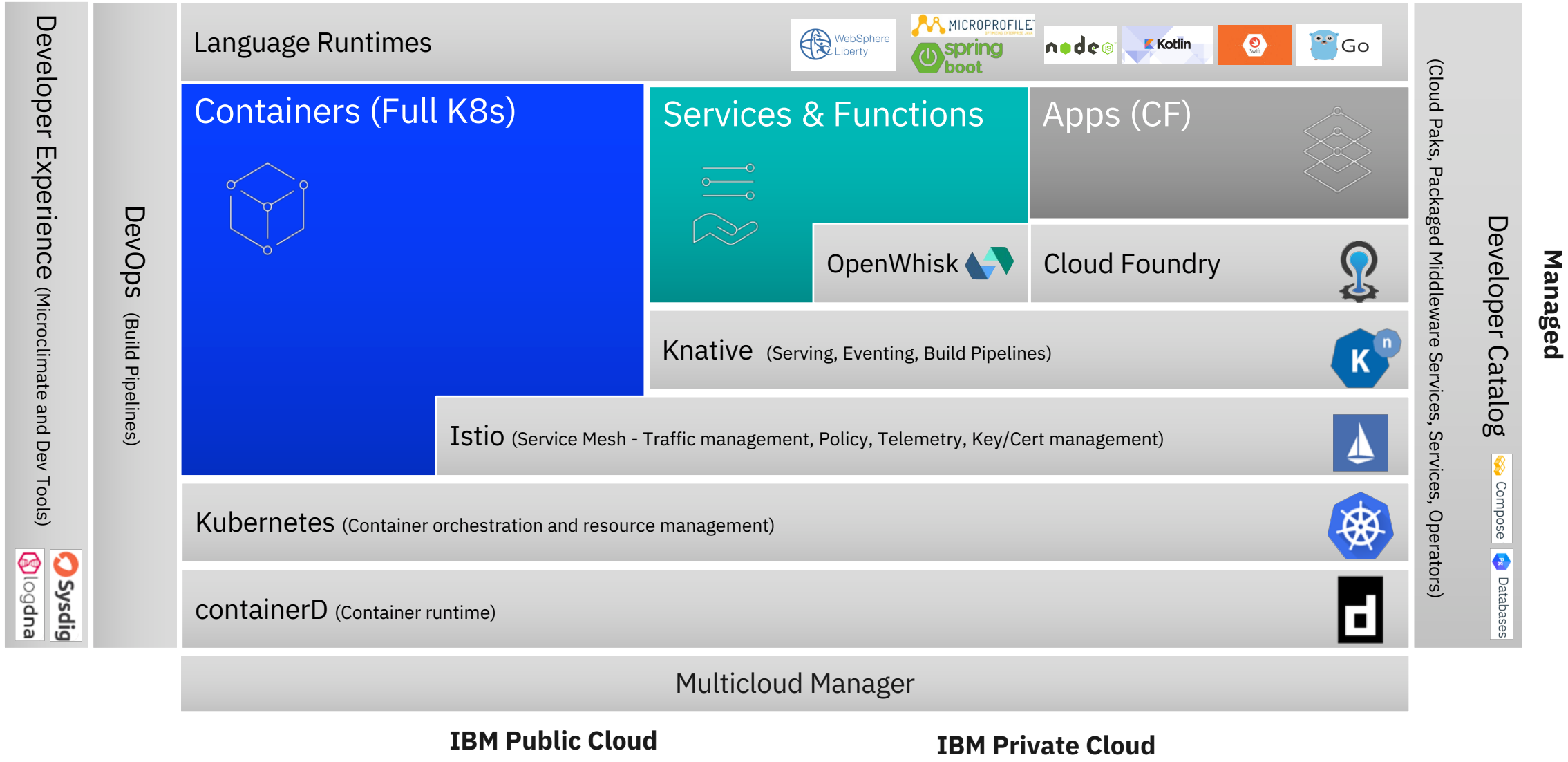
The Evolving Shape of Applications



CHALLENGES

CLOUD NATIVE DEVELOPMENT	CORE MODERNIZATION	INTEGRATION	DATA MOVEMENT & GOVERNANCE	SERVICE MANAGEMENT	SECURITY & COMPLIANCE

Industry Cloud Native Platform



IBM CODE

Innovate with IBM's Cloud Native Platform


Functions


Kubernetes

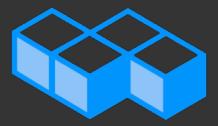

CLOUD FOUNDRY


Istio


Event Streams


Watson


Cloudant


Databases

Continuously delivering innovation to your application

Client Success Stories

Always-On & Automation

IBM Cloud builds on open-source to relieve the pains of security, scale, software, & infrastructure management

IBM CODE