Cloud Pak for Applications

Raghavendra Deshpande
Developer Advocate
IBM

IBM Developer

IBM

Contents

- App Modernization
- Cloud Native – Why?
- Cloud Paks
- Cloud Pak for Applications
  - Overview
  - Components
  - Why?

App Modernization is inevitable

Evolution of application architectures

Late 90’s	**Enterprise Application (EAI) Services and Models** Addressed integration and transactional challenges primarily by using message oriented middleware. Mostly proprietary systems needing a proliferation of custom interfaces.
Mid 00’s	**Service Oriented Architectures** Based on open protocols like SOAP and WSDL making integration and adoption easier. Usually deployed on an Enterprise ESB which is hard to manage and scale.
Early 10’s	**API Platforms and API Management** REST and JSON become the defacto standard for consuming backend data. Mobile apps become major consumers of backend data. New Open protocols like OAuth become available further simplifying API development .
2015 and beyond	**Cloud Native and Microservice Architecture** Applications are composed of small, independently deployable processes communicating with each other using language-agnostic APIs and protocols.

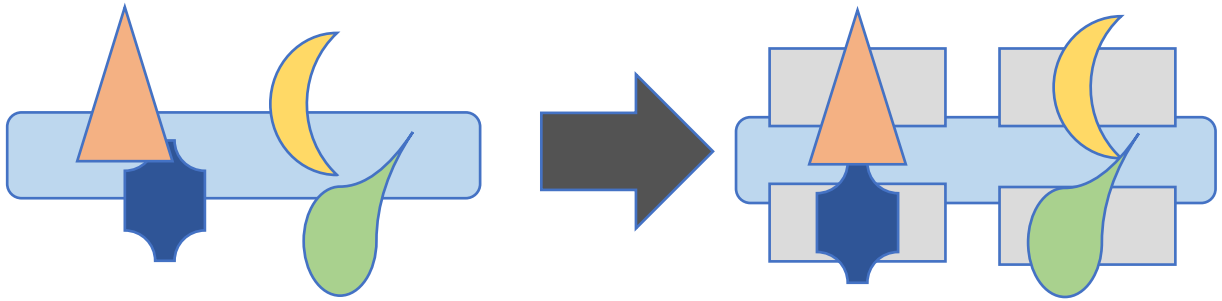
Key tenets of a cloud native application

1. Packaged as light weight **containers**
2. Developed with best-of-breed languages and frameworks
3. Designed as loosely coupled **microservices**
4. Centered around **APIs** for interaction and collaboration
5. Architected with a clean separation of stateless and stateful services
6. Isolated from server and operating system dependencies
7. Deployed on self-service, elastic, **cloud infrastructure**
8. Managed through agile **DevOps** processes
9. Automated capabilities
10. Defined, policy-driven resource allocation

https://thenewstack.io/10-key-attributes-of-cloud-native-applications/

Key tenets of a microservices architecture

1. Large monoliths are broken down into many small services
2. Services are optimized for a single function or business capability
3. Teams that write the code should also deploy the code
4. Smart endpoints, dumb pipes (message brokers)
5. Decentralized governance
6. Decentralized data management

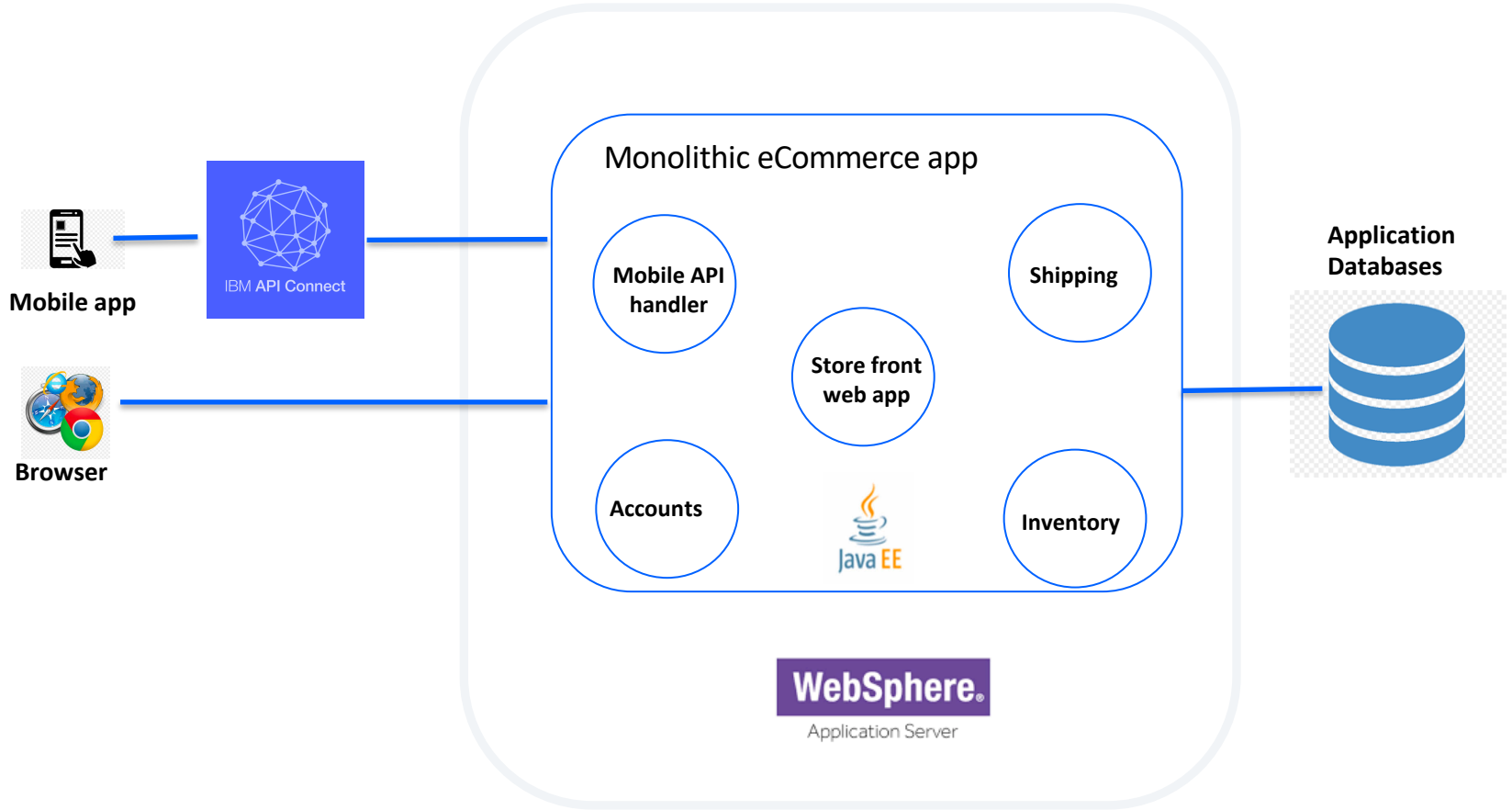


https://martinfowler.com/articles/microservices.html

Example monolithic application

eCommerce app

- Store front web interface
- Customer Accounts
- Inventory
- Shipping
- Back end for mobile app

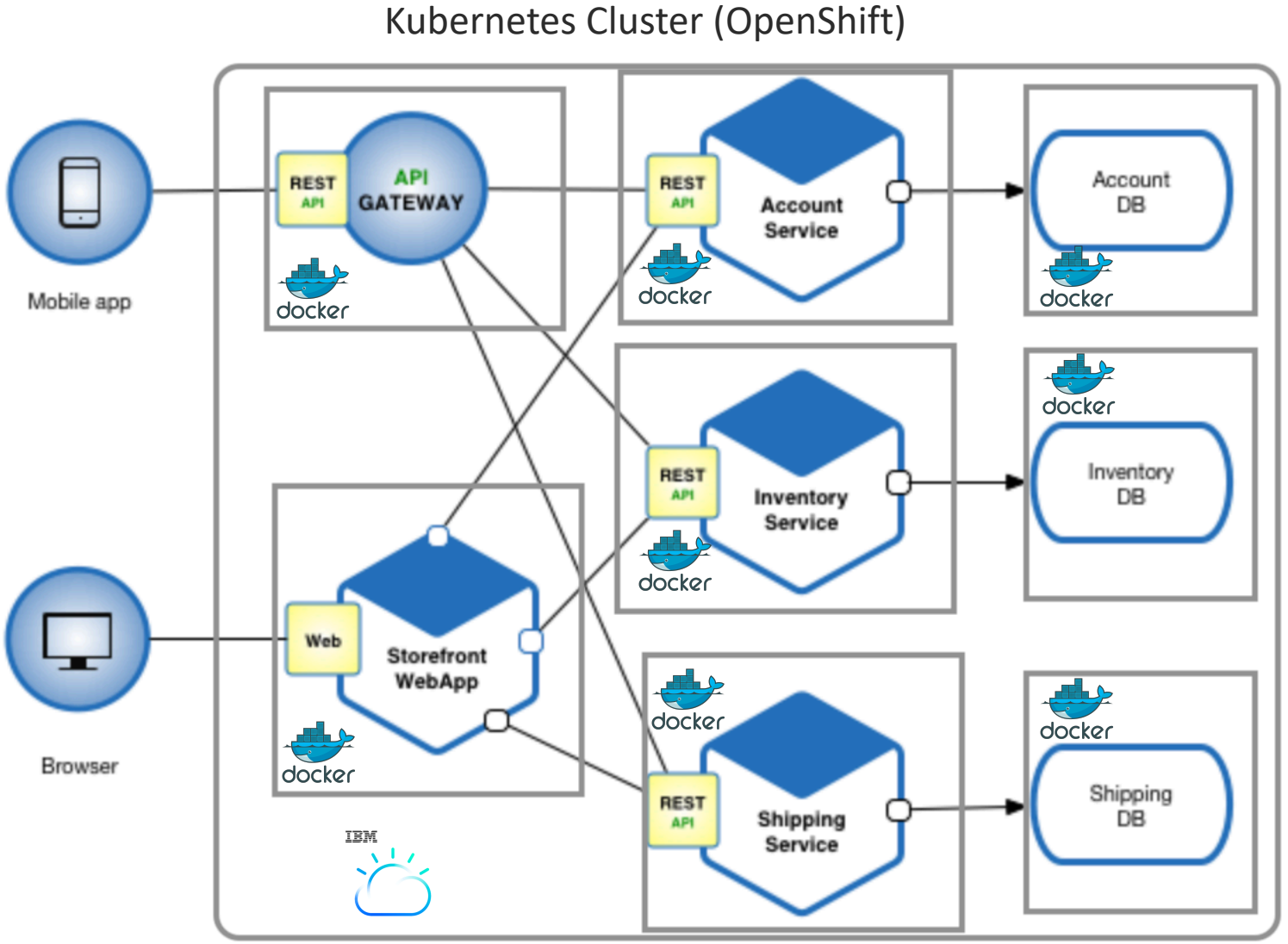


An eCommerce Java EE app on Websphere

Transformed application

Key technologies

- Containers (Docker)
- Container orchestration (Kubernetes)
- Transformation Advisor
- 12-Factor Best Practices
- CI/CD tools (e.g Jenkins)



An eCommerce microservices app on a Kubernetes cluster

Why microservices and cloud native?

Efficient teams	Simplified deployment	Right tools for the job	Improved application quality	Scalability
• End to end team ownership of relatively small codebases	• Each service is individually changed, tested, and deployed without affecting other services	• Teams can use best of breed technologies, libraries, languages for the job at hand	• Services can be tested more thoroughly in isolation	• Services can be scaled independently at different rates as needed
➢Teams can innovate faster and fix bugs more quickly	➢Time to market is accelerated.	➢Leads to faster innovation	➢Better code coverage	➢Leads to better overall performance at lower cost

Cultural change considerations

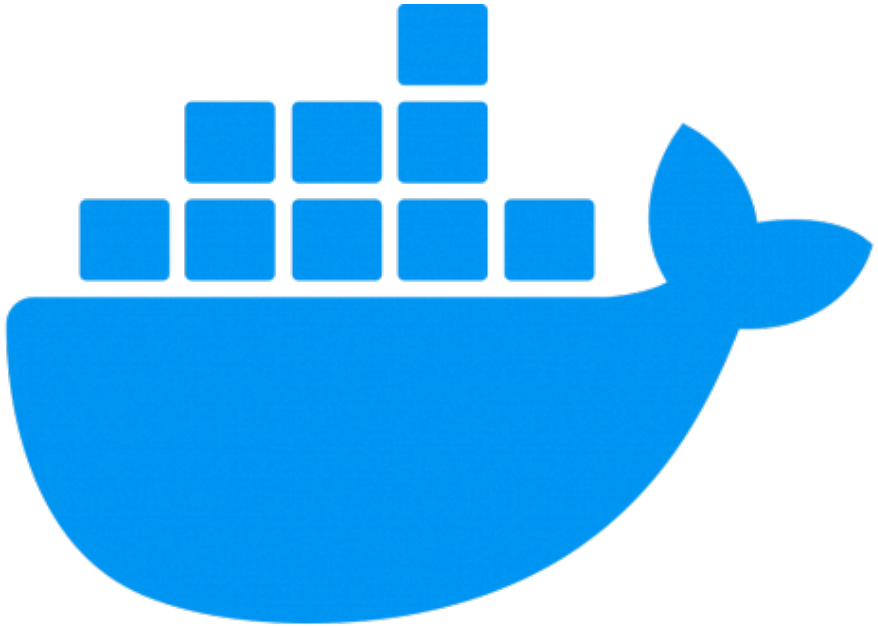
- **Smaller teams with broader scope**
  - Mini end to end development orgs in each team vs large silos across the entire development team
- **Top down support with bottom up execution**
  - Change can't happen effectively w/o executive sponsorship
  - Change needs to be executed at the smallest organizational unit to take hold
- **Teams own all metrics related to operations and development**
  - Have to minimize downtime + number of bugs while also maximizing the rate at which needed features are added and minimizing the time to market of those new features
- **Trust**
  - Teams need to build trust with other teams that they collaborate with rather than relying on one size fits all checklists and rules
- **Reward based on results not compliance**
  - Cultures only change when people are measured and rewarded for outcomes consistent with the changes
  - Smaller more autonomous teams work better with less central micromanagement and more focus on broad measurable goals

Docker Overview

Containers provide process isolation.

Docker is a container runtime.

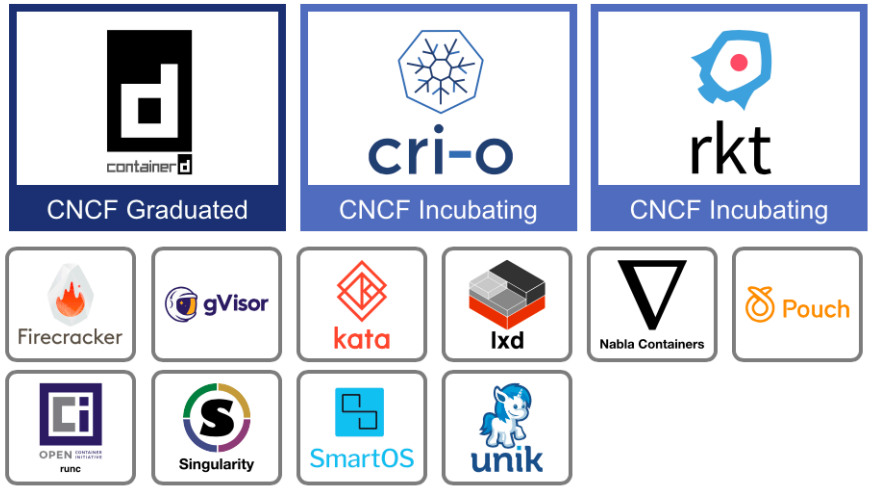
Docker and containers



- **Docker** is one of many **container runtimes**
  - containerD, cri-o, rkt, are other examples

- Docker has the **best developer tools**
- Other runtimes more suited for **production**

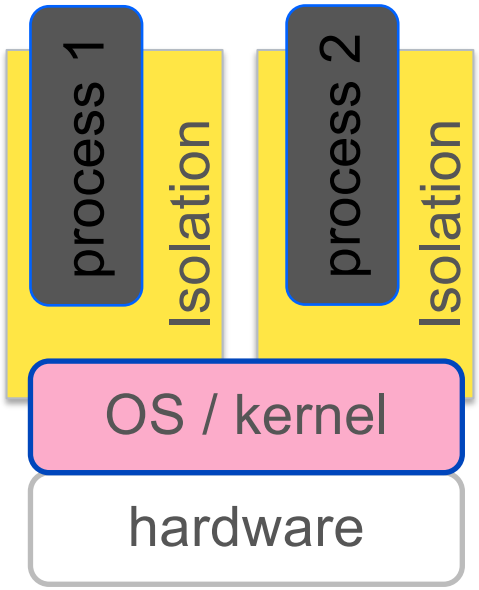
- Containers provide **process isolation**
- Containers are **not virtual machines**



Containers are not a new idea

- Other process isolation technologies:
  - chroot ('80s) process spawned in isolated file space
  - FreeBSD jails
  - OS-level virtualization (user-mode-linux, virtuozzo)
  - Solaris Containers
  - LinuX Containers (LXC)
  - Cloud Foundry (Warden, Garden)

- Docker provided an ecosystem approach that transformed perception
  - Building application-centric containers
  - Mechanism for sharing images (Docker Registry)
  - Open-source enabled



Docker Images and Image Registry

Docker Images

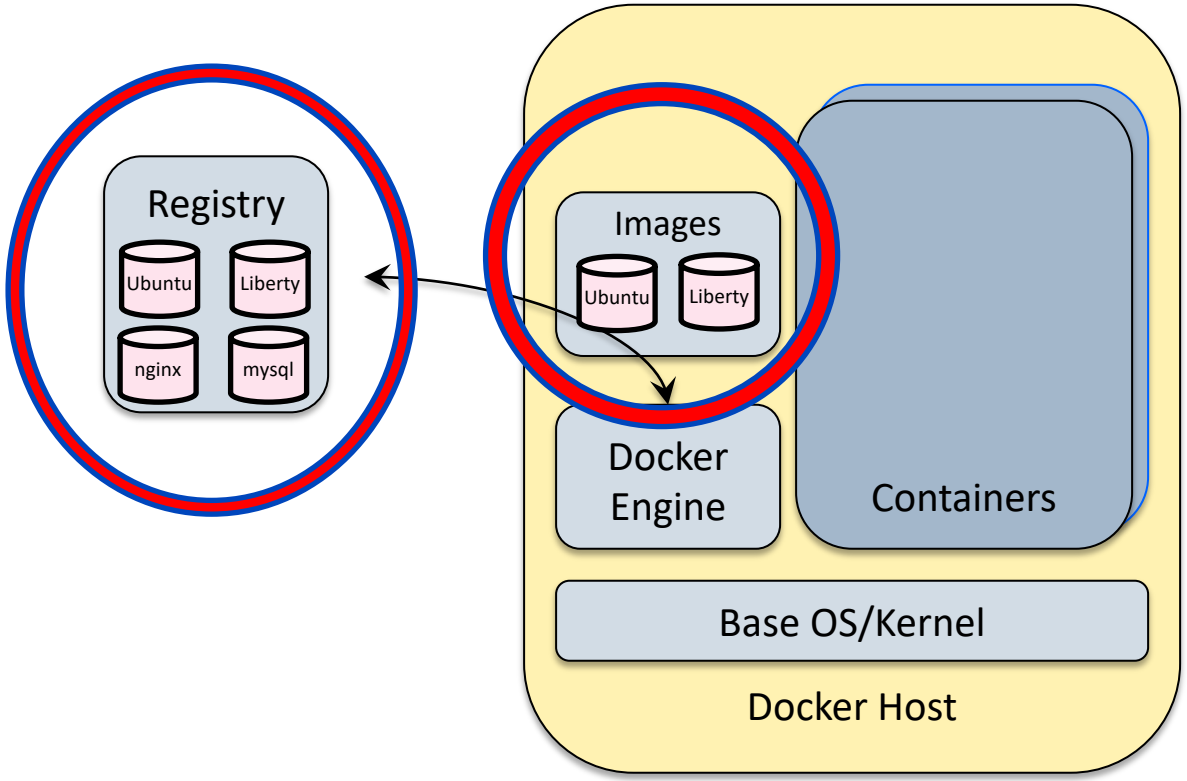
- Tar file containing a container's filesystem and metadata

Docker Registry

- The central place to share images
- Enterprises will want to use a private registry
- OpenShift has a built-in registry

DockerHub - http://hub.docker.com

- Public registry of Docker Images
- Also useful to find prebuilt images for web servers, databases, and much more



Build your own image with a **Dockerfile**

1. Create a **Dockerfile** to script how you want the image to be built
2. Run **docker build** to build an image
3. Run **docker run** to run it locally
4. Run **docker push** to push it to a registry
5. Run **docker pull** to download an image

```
FROM java:8 # This might be an ubuntu or...
COPY *.jar app.jar
CMD java -jar app.jar
```

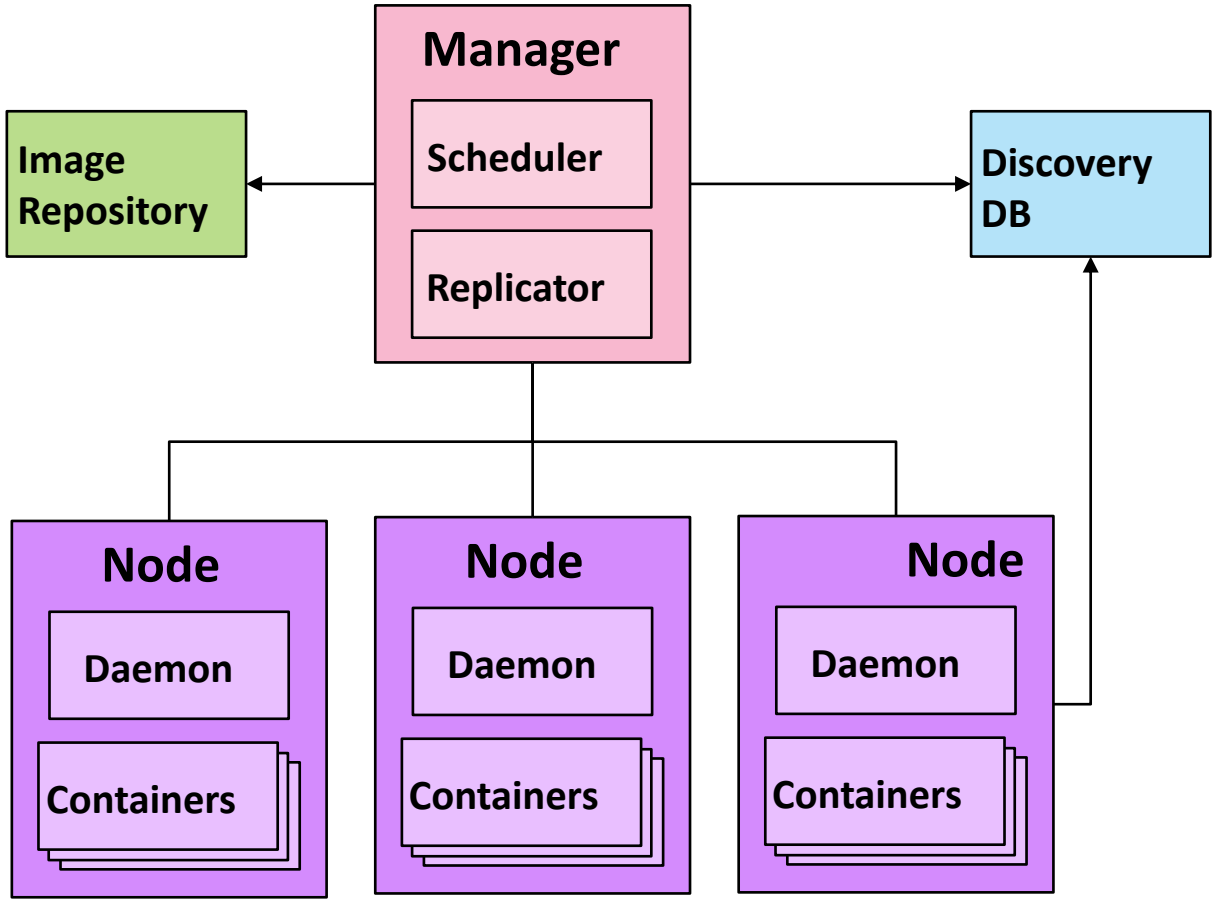
Kubernetes Overview

Container orchestration with Kubernetes unlocks value of containers as application components

What is container orchestration?

Container Orchestration handles the following:

- Cluster management
- Scheduling
- Service discovery
- Replication
- Health management



Kubernetes and Container Orchestration

Kubernetes is a Container Orchestrator

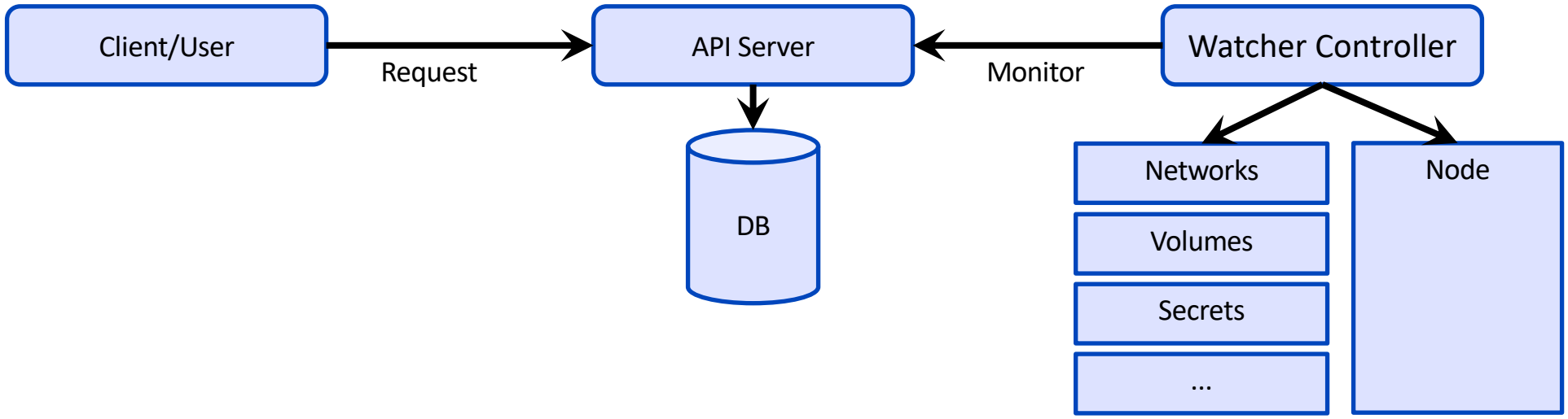
- Provisions, manages, and scales containerized applications
- Supports:
  - Automated scheduling and scaling
  - Zero downtime deployments
  - High availability and fault tolerance
  - A/B deployments
- Manage infrastructure resources needed by applications
  - Volumes
  - Networks
  - Secrets
  - And many many many more..
- Declarative model
  - Provide the "desired state" and Kubernetes will make it happen

Kubernetes fun facts:

- Builds upon 15 years of experience of running production workloads at Google
- Open Governance via CNCF
- Adopted by IBM, Amazon, Microsoft, Red Hat, Google,
- Means “helmsman” in Greek

Kubernetes Architecture

- At its core, Kubernetes is a database (etcd) with "watchers" & "controllers" that react to changes in the DB. The controllers are what make it Kubernetes. This extensibility is part of its "secret sauce".
- The database represents the user's desired state. Watchers attempt to make reality match the desired state
- The API server is the HTTP/REST front-end to the DB



Kubernetes Clients (CLI and Dashboard)

```
> kubectl
```

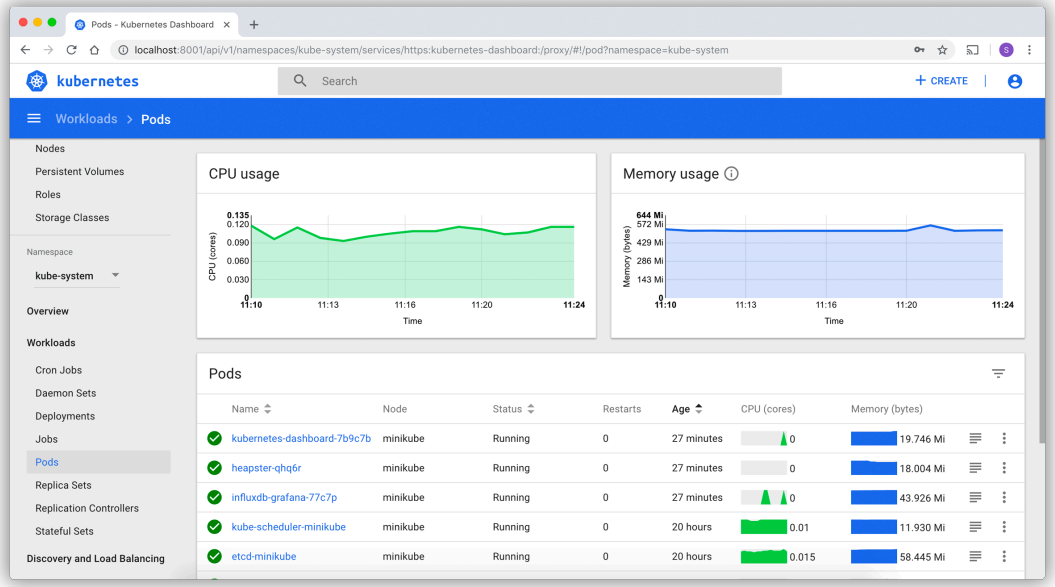
Kubernetes CLI

- Directly manipulate YAML
  - kubectl (create|get|apply|delete) -f myResource.yaml

- https://kubernetes.io/docs/tasks/tools/install-kubectl

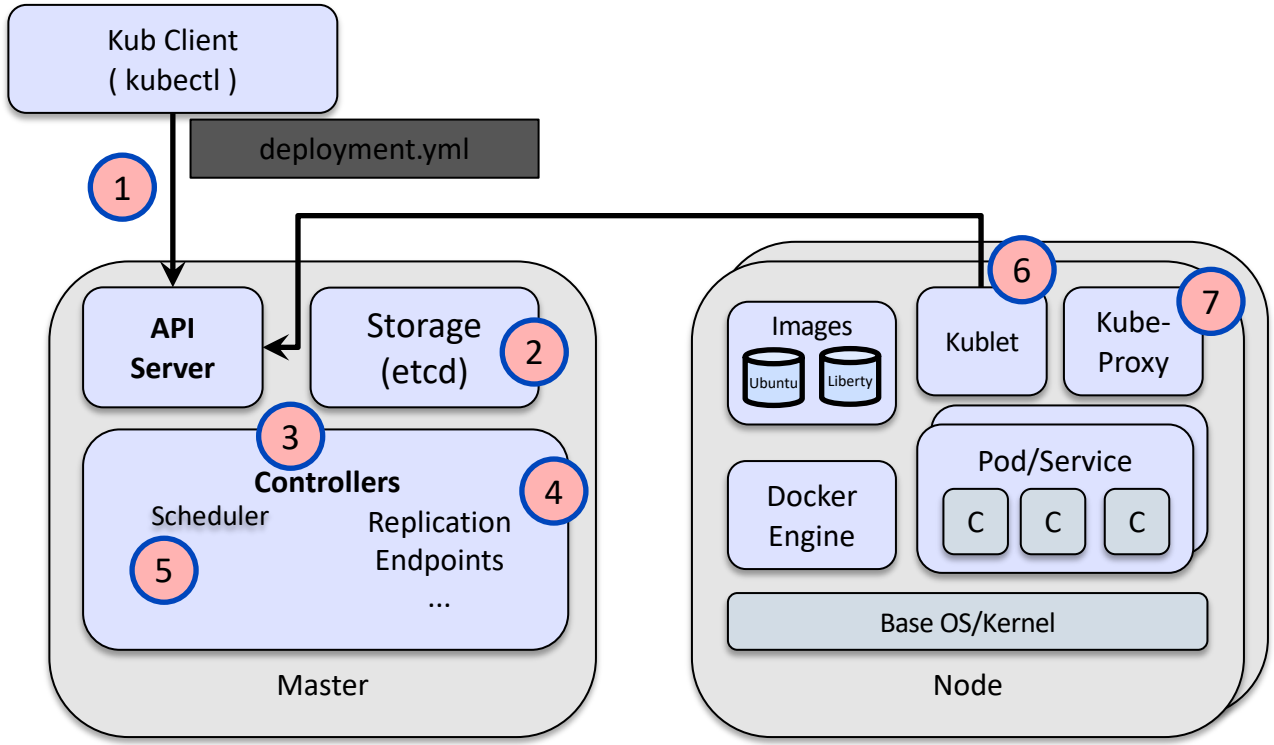
Kubernetes Dashboard

- Another way to view and modify resources



Kubernetes in Action!

1. User via "**kubectl**" deploys a new application
2. API server receives the request and stores it in the DB (etcd)
3. Watchers/controllers detect the resource changes and act upon it
4. ReplicaSet watcher/controller detects the new app and creates new pods to match the desired # of instances
5. Scheduler assigns new pods to a kubelet
6. Kubelet detects pods and deploys them via the container runtime (e.g. Docker)
7. Kubeproxy manages network traffic for the pods – including service discovery and load-balancing



12 Factor Apps

12 Factor is a methodology for building software

Tenets for a 12 Factor App

1. Codebase
2. Dependencies
3. Config
4. Backing Services
5. Build, Release, Run
6. Processes
7. Port Binding
8. Concurrency
9. Disposability
10. Dev/Prod Parity
11. Logs
12. Admin processes

https://12factor.net

Cloud Pak For Application

Cloud Paks are a collection of enterprise-grade container offerings that run on OpenShift

Cloud Paks: Middleware anywhere

A faster, more secure way to move your core business applications to any cloud through enterprise-ready containerized software solutions

Complete yet simple
Application, data and AI services, fully modular and easy to consume

IBM certified
Full software stack support, and ongoing security, compliance and version compatibility

Run anywhere
On-premises, on private and public clouds, and in pre-integrated systems

Cloud Paks: Pre-integrated for cloud use cases

Today, IBM offers clients the first five Cloud Paks…

Reduce dev time up to 84%*

Cloud Pak for Applications

Build, deploy, and run applications

IBM containerized software

Container platform and operational services — Red Hat OpenShift

Make data ready for AI in days

Cloud Pak for Data

Collect, organize, and analyze data

IBM containerized software

Container platform and operational services — Red Hat OpenShift

Eliminate 33% of integration cost

Cloud Pak for Integration

Integrate applications, data, cloud services, and APIs

IBM containerized software

Container platform and operational services — Red Hat OpenShift

Reduce manual processes up to 80%*

Cloud Pak for Automation

Transform business processes, decisions, and content

IBM containerized software

Container platform and operational services — Red Hat OpenShift

Reduce IT op expense by up to 75%*

Cloud Pak for Multicloud Management

Multicloud visibility, governance, and automation

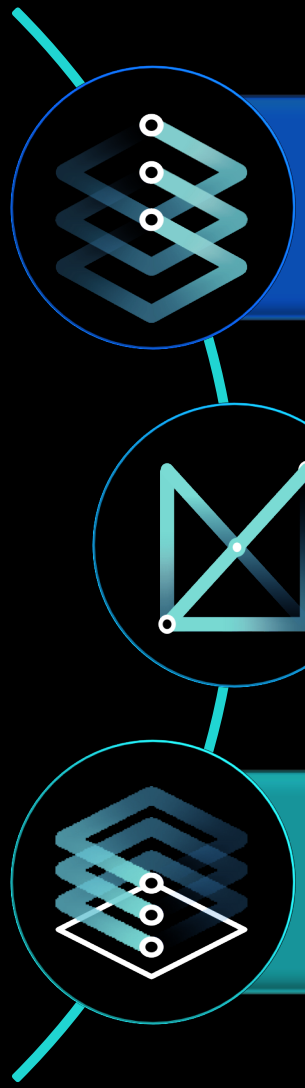
IBM containerized software

Container platform and operational services — Red Hat OpenShift

IBM Cloud · aws · Azure · Google Cloud · Edge · Private · Systems

Cloud Pak for Applications uses cloud-native methodologies to create new apps and build or run existing apps

Cloud Pak for Applications: 3 use cases, 1 offering



Run existing apps | Continue to run your apps, where they are.

Modernize existing apps | **When apps need to move, IBM has the most experience, tools, and experts to move them**

Building new apps | **New apps are automatically ready for hybrid-cloud deployment, using the best of open source, fully supported**

... resulting in lower costs with tangible benefits!

9.5x
more likely to see improvement in **product quality**

5x
more likely to see improvement in **cost reduction**

1.7x
more likely to see improvement in **user/employee satisfaction**

1.6x
more likely to see improvement in **customer retention**

Build once, deploy anywhere
For optimized data and workload placement

Open and integrated approach
Visibility, governance, and secure data access

Culture and skill transformation
Best practices, proven methods, and tools

Cloud Pak for Apps: What you need for today and tomorrow

Run Existing Apps and Build New Apps

Continue to run your apps, where they are.

New apps are automatically ready for hybrid-cloud deployment, using the best of open source, fully supported.

MODERNIZE APPS

When apps need to move, IBM has the right experience, tools, and experts to move them.

WebSphere Application Server

WebSphere ND

WebSphere Base

Liberty Core

Codewind

Kabanero Enterprise

Red Hat Runtimes

IBM Integrated Application Runtimes

LoopBack

Appsody

OPERATOR FRAMEWORK

Integrated DevOps

Digital App Builder

IBM Mobile Foundation

Mobile Services

IBM Cloud Private

Red Hat OpenShift

IBM Modernization & Developer Tools

Transformation Advisor

WebSphere Migration Toolkit

Enterprise Dev tools extensions for local IDE's

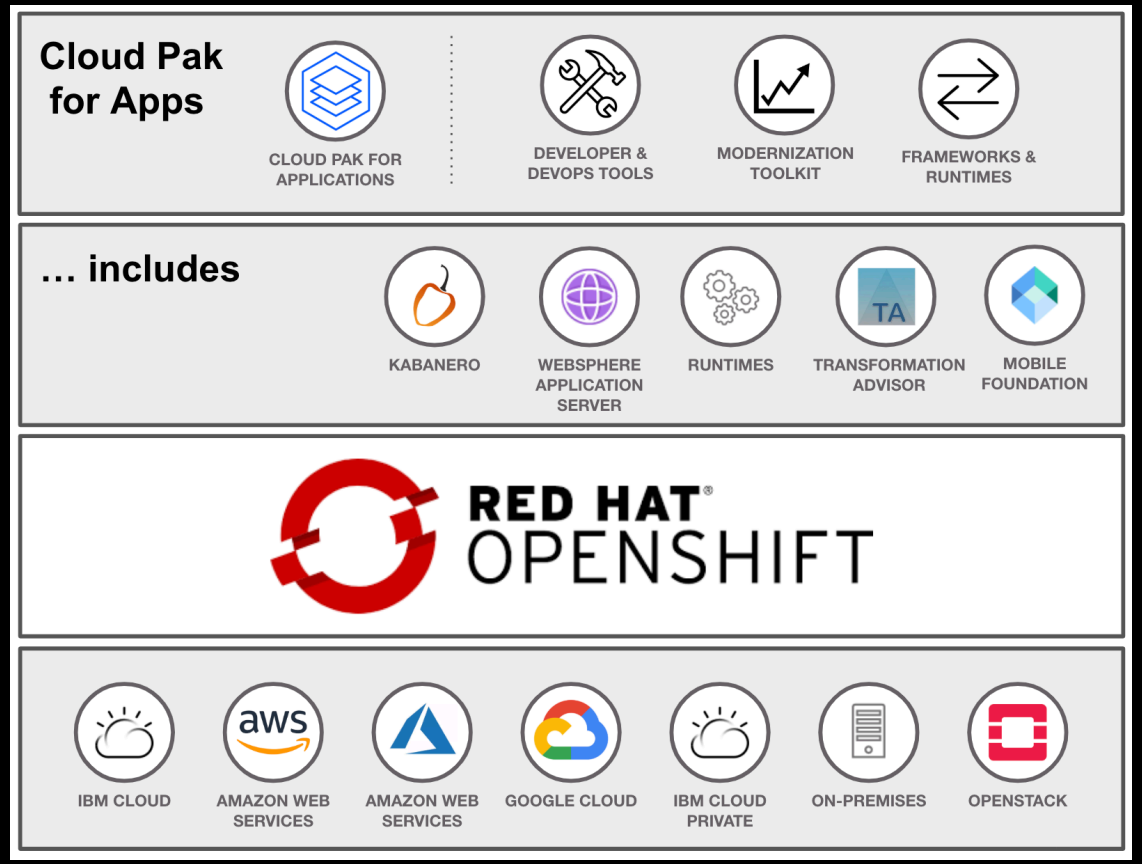
Included with all components

Demo1 - Deploy IBM Cloud Pak for Applications on Managed OpenShift on IBM Cloud

https://www.ibm.com/cloud/garage/dte/producttour/deploy-ibm-cloud-pak-application-managed-openshift-ibm-cloud

Cloud Pak for Apps Components

- **Kabanero Enterprise**
  - The enterprise-ready implementation of the Kabanero.io open source project. Kabanero is an open source project that brings together foundational open source technologies into a modern microservices-based framework.
- **IBM WebSphere Application Server**
  - Cloud Pak for Applications include WebSphere Application Server ND, WebSphere Application Server, and the WebSphere Liberty Core editions with the ability to easily mix and match between them.
- **Red Hat Runtimes**
  - Provides a set of open runtimes, tools, and components for developing and maintaining cloud-native applications.
- **Transformation Advisor**
  - Used to quickly evaluate on-premises Java™ EE apps for deployment to the cloud.
- **IBM Mobile Foundation**
  - A platform to build the next generation of digital apps, including mobile, wearables, conversation, web, and PWAs. Includes containerized mobile back-end services covering comprehensive security, application life cycle management, push notifications, off-line sync, and back-end integration.



Cloud Pak for Apps Components: Kabanero

KABANERO

APPSODY

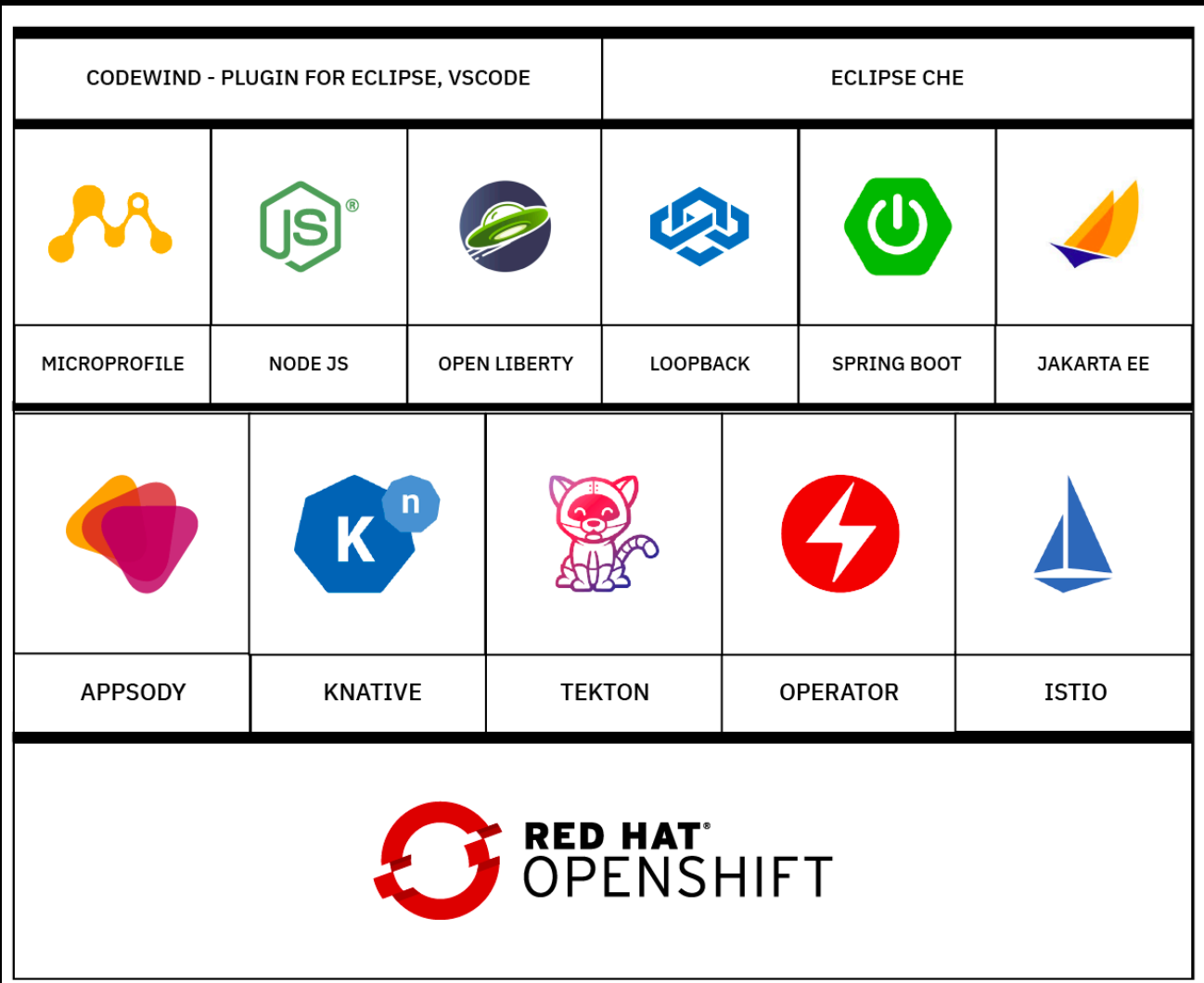
CODEWIND

Kabanero is an open source project that brings together foundational open source technologies into a modern microservices-based framework.

Appsody provides cloud-native technology stacks and starter templates; a central repository of stacks; and a CLI to create, test, run and deploy apps that utilize these stacks.

Codewind provides plugins for VS Code, Eclipse, Eclipse Che

Cloud Pak for Apps Components: Kabanero, 100% open source



PROGRAMMING TOOLS

PROGRAMMING FRAMEWORKS

JAVA AND NODE.JS TECHNOLOGIES

CLOUD DEVOPS TECHNOLOGIES

APPSODY	CONSISTENT DEVELOPMENT TEMPLATES
KNATIVE	SERVERLESS ARCHITECTURES
TEKTON	CI/CD PIPELINES
OPERATOR	APPLICATION MANAGEMENT
ISTIO	MICROSERVICE COMMUNICATION

CONTAINER PLATFORM

CAN DEPLOY ON KUBERNETES, OR PREFERRED ON RED HAT OPENSHIFT

Kabanero's projects, Codewind and Appsody, leverage open source projects such as Knative, Istio, Tekton, and the Operator framework to run cloud-native applications written in Node.JS (Loopback, Express), and Java (Liberty, Spring, Microprofile).

Cloud Pak for Apps Components: WebSphere Application Server

Cloud Pak for Applications includes both WebSphere Liberty (WAS) and traditional WebSphere (often called "tWAS"). WebSphere Liberty is a highly composable, dynamic application server runtime environment. Thanks to its fast startup and minimal footprint, it's perfect for cloud-native microservices application development. The traditional WebSphere edition continues the evolution from the original release in June 1998.

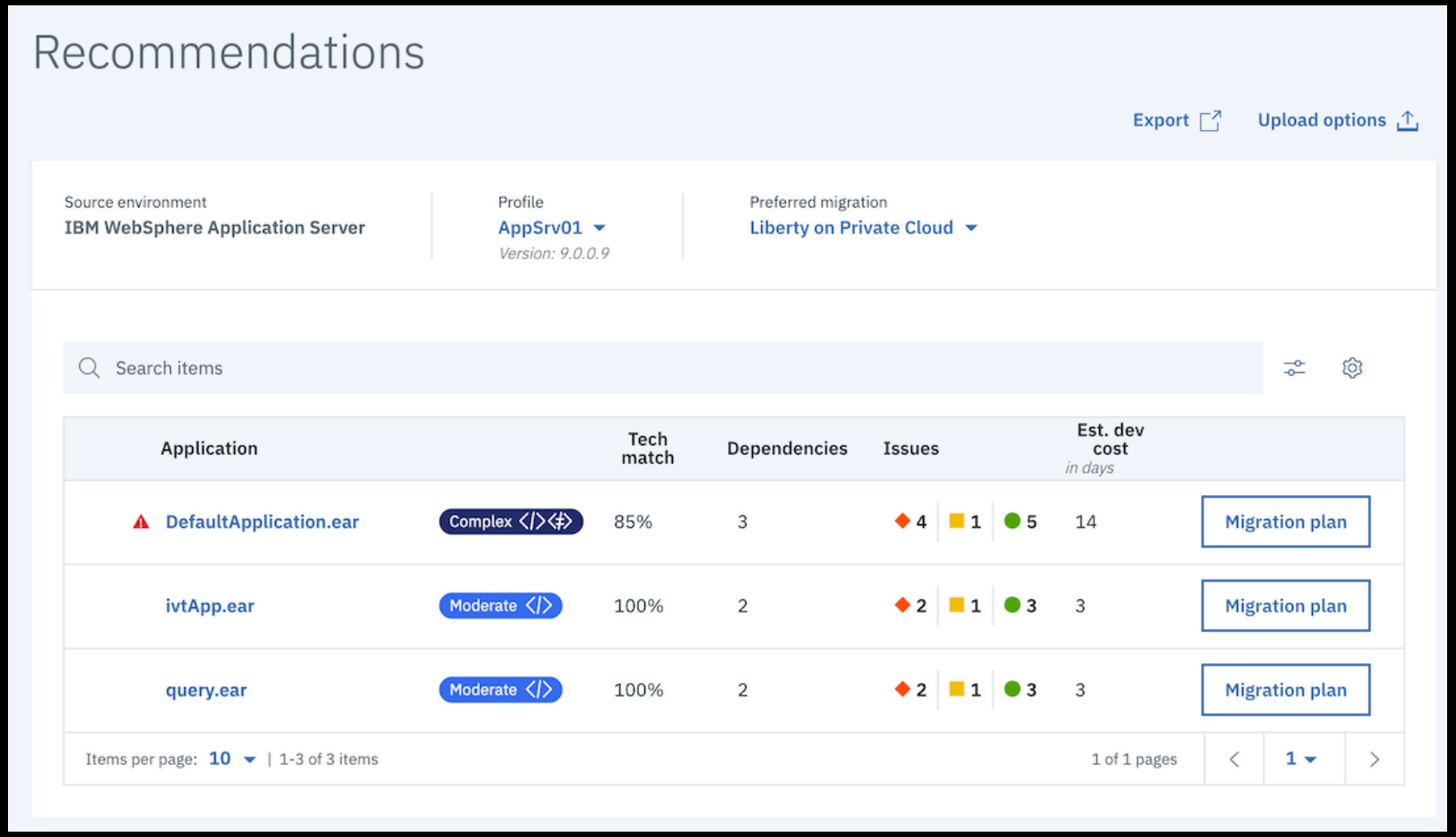
Cloud Pak for Apps Components: Red Hat Runtimes

Product or Service		Product Information & Documentation
Node.js		Node.js
Eclipse Vert.x		Eclipse Vert.x
Thorntail		Thorntail
Spring Boot		Spring Boot
OpenJDK		OpenJDK
Red Hat JBoss® Enterprise Application Platform		Red Hat JBoss® Enterprise Application Platform
Red Hat JBoss® Web Server		Red Hat JBoss® Web Server
Red Hat Data Grid		Red Hat Data Grid
Red Hat AMQ Broker		Red Hat AMQ Broker
Red Hat Single Sign-On		Red Hat Single Sign-On
Red Hat Core Services		Red Hat Core Services

Red Hat Runtimes provides a set of open runtimes, tools, and components for developing and maintaining cloud-native applications. It offers lightweight runtimes and frameworks for highly distributed cloud architectures, such as microservices.

https://access.redhat.com/articles/4394291

Cloud Pak for Apps Components: Transformation Advisor



Transformation Advisor runs data collector agents that analyze apps running on IBM WebSphere Application Server, Apache Tomcat, or WebLogic application servers, and IBM MQ queue managers. The collector automatically uploads the results to Transformation Advisor and generates reports to help you understand issues and where code changes might be required when moving that application to the cloud.

https://www.ibm.com/us-en/marketplace/cloud-transformation-advisor

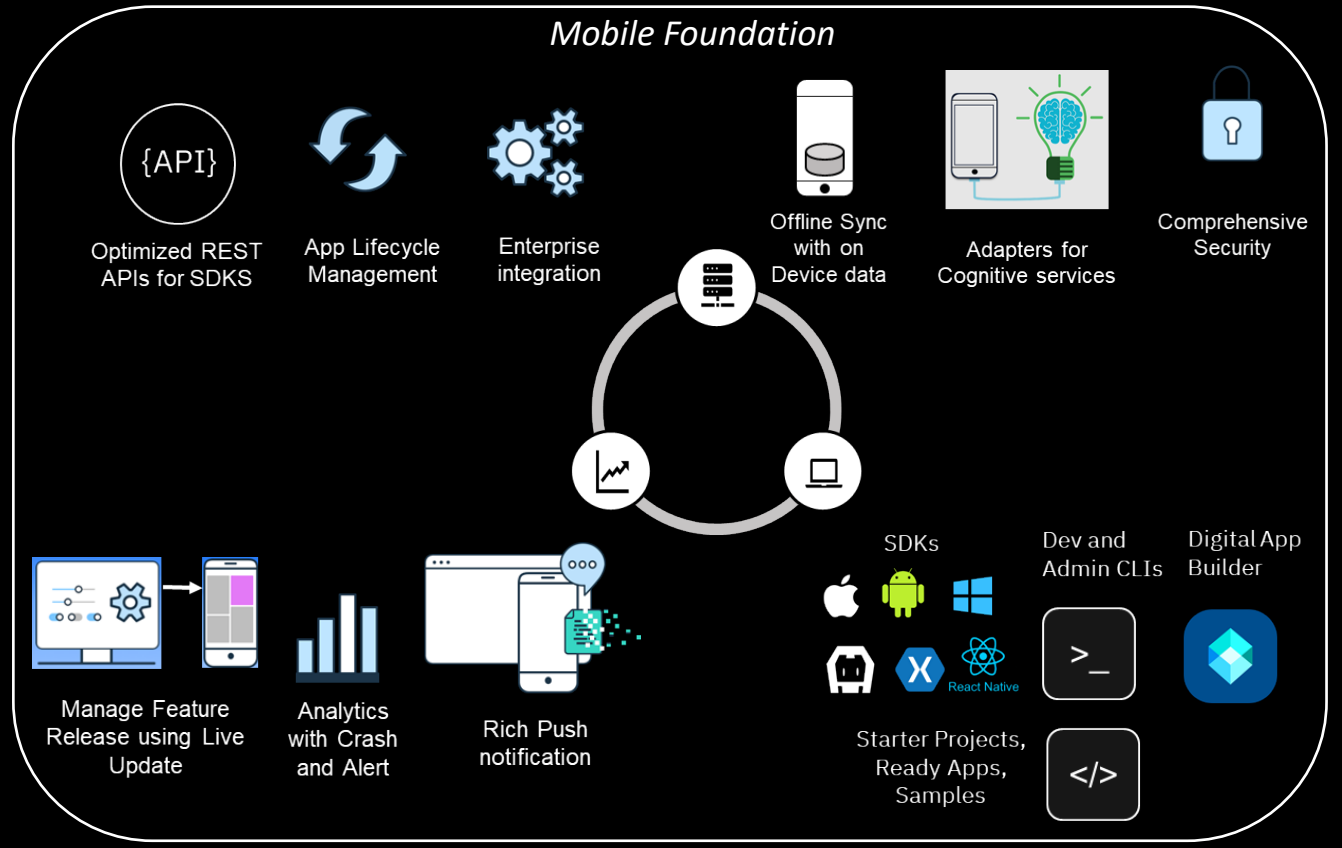
Demo2 - Evaluate an on-prem WebSphere application for migration

https://www.ibm.com/cloud/garage/dte/producttour/evaluate-prem-websphere-application-migration

Cloud Pak for Apps Components: IBM Mobile Foundation

1. Mobile Foundation Services:
   - Deliver comprehensive tooling and runtime for Mobile developers on OpenShift
   - Assist developers innovate (AI/AR/VR/Engaging) iOS, Android, Wearables, PWAs, Smart Web Apps
   - 5G-ready developer platform for low latency, secured AR/VR, off-line faster Mobile use cases

2. Digital App Builder:
   - Removes the skills barrier to address IT's huge App backlogs
   - Complements Kabanero tooling for front-end developers
   - Introduces easy to use AI services for App Developers

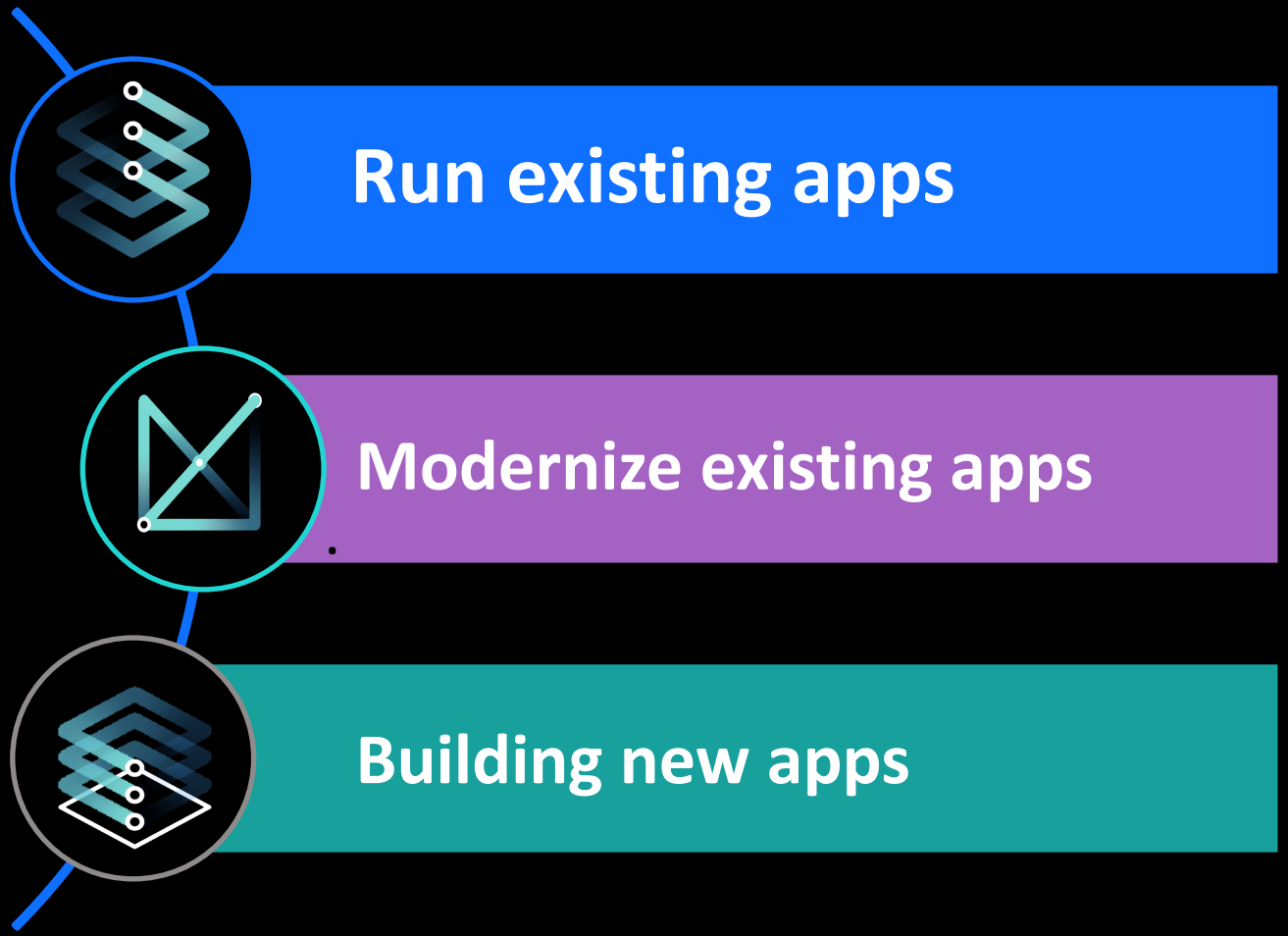


Demo 3 – Mobile Foundation

https://www.ibm.com/cloud/garage/dte/tutorial/creating-hybrid-mobile-application-ibm-digital-app-builder

Cloud Pak for Applications: **WHY?**

Flexibly rebalance entitlement over time: from what you need today, to what you need tomorrow



Run existing apps

Modernize existing apps

Building new apps

WebSphere Application Server

WebSphere ND
WebSphere Base
Liberty Core

**includes latest version, WAS 9.0.5*

IBM Mobile Foundation

IBM Modernization & Developer Tools

OpenShift + Kabanero Enterprise

Full OpenShift offering – not restricted

Perpetual and Term options available


IBM