Leveraging Kubernetes-Based Platforms for Microservices

Eric Herness
Hybrid Cloud CTO
IBM Fellow
herness@us.ibm.com

Karri Carlson-Neumann
Hybrid Cloud CTO Office
Introduction

Big Picture

Kubernetes and Where it Fits
• Containers and Container Orchestration
• Kubernetes Basics

Microservices
• Basics
• The Programming Model meets Kubernetes
• Getting the value from the microservices approach → back to kubernetes

Demonstration

Solutions View – Cloud Architecture and Application Architecture

Conclusion
The Big Picture
Software – Evolution

The New Stage
- The role it now plays in everything we do
- The business demands → velocity
- The Cloud

Building

Packaging

Exposing

Running and Managing

Operating

Polyglot
Java, node.js, Swift, ...

Microservices frameworks

APIs
Swagger (Open API)

Containers
Docker (Open Container Initiative)

Container Orchestration
kubernetes

Do we call this cloud-native?

Concepts
Containers

A standard way to package an application and all its dependencies so that it can be moved between environments and run without changes.

Containers work by isolating the differences between applications inside the container so that everything outside the container can be standardized.
What is a Container

• An isolated user space within a running Linux OS
• Shared kernel across containers
• Direct device access
• All packages and data in an isolated run-time, saved as a filesystem.
• Resource management implemented with cgroups
• Resource isolation through namespaces
Container History Lesson

- 1982: Unix/chroot BSD
  - FreeBSD jails/Solaris zones
- 2000: OpenVZ Parallels
- 2005: AIX Wpars IBM
- 2006: Cgroups/Process Containers IBM/Google
- 2007: LXC
- 2008: Today

2018 IBM Corporation
Intro to Docker

Enabling application development **efficiency**, making deployment more **efficient**, eliminating vendor ‘lock-in’ with true **portability**

- Open Software
  - Launched March 2013
  - 2.0+ billion downloads of Docker images

- Open Contribution
  - 3300+ contributors
  - #2 most popular project
  - 185 community meet-up groups in 58 countries

- Open Design
  - Contributors include IBM, Red Hat, Google, Microsoft, VMware, AWS, Rackspace, and others

- Open Governance
  - 12 member governance advisory board selected by the community
Docker Basics – A shipping container for code

**Image**
- A **read-only snapshot** of a container stored in Docker Hub to be used as a template for building containers

**Container**
- The standard unit in which the application service resides or transported

**Docker Hub**
- Available in SaaS or Enterprise to deploy anywhere you choose
- Stores, distributes and shares container images

**Docker Engine**
- A program that creates, ships and runs application containers
- Runs on any physical and virtual machine or server locally, in private or public cloud (*portability*)
- Client communicates with Engine to execute commands
These technologies can become packaging and management models for:

1. New microservice based projects (Cloud Native) – our focus today
2. Existing applications (monoliths?) – Cloud Enabled
3. Existing middleware - Cloud Enabled
Kubernetes

Going a little deeper into Kubernetes basics
Everyone’s container journey starts with one container....
At first the growth is easy to handle....
But soon it is overwhelming... chaos reigns
Regain control with Kubernetes
What is Kubernetes?

- Container orchestrator
- Runs and manages containers
- Supports multiple cloud and bare-metal environments
- Inspired and informed by Google's experiences and internal systems
- 100% Open source, written in Go
- Manage applications, not machines
- Rich ecosystem of plug-ins for scheduling, storage, networking
Container Stack

Layer 6: Development Workflow
Opinionated Containers

Layer 5: Orchestration/Scheduling
Service Model

Layer 4: Container Engine

Layer 3: Operating System

Layer 2: Virtual Infrastructure

Layer 1: Physical Infrastructure
Kubernetes Architecture

UI → API → Kubernetes Master

- Etcd
- API Server
- Controller Manager
- Scheduler

Worker Node 1 → Worker Node 2 → Worker Node 3 → Worker Node n

Registry
**Kubernetes Architecture**

Nodes – hosts that run Kubernetes applications

Master nodes:
- Controls and manages the cluster
- Kubectl (command line)
- REST API (communication with workers)
- Scheduling and replication logic

Worker nodes:
- Hosts the K8s services
- Kubelet (K8s agent that accepts commands from the master)
- Kubeproxy (network proxy service responsible for routing activities for inbound or ingress traffic)
- Docker host
**Kubernetes Architecture**

- **Etcd** - a highly-available key value store which K8s uses for persistent storage of all of its REST API objects.

- **API Server** – Kubernetes API server

- **Controller manager** – Daemon that runs controllers (background threads that handle routine tasks). Includes Node Controller, Replication Controller (ReplicaSet), Endpoints Controller, Service Account * Token Controllers)

- **Scheduler** – schedules pods in worker nodes
**Kubernetes Workloads**

Pods:
- Smallest deployment unit in K8s
- Collection of containers that run on a worker node
- Each has its own IP
- Pod shares a PID namespace, network, and hostname

Service:
- Collections of pods exposed as an endpoint
- Information stored in the K8s cluster state and networking info propagated to all worker nodes
Kubernetes terminology

Resource Types include:
- configmaps
- daemonsets
- deployments
- endpoints
- events
- ingresses
- jobs
- namespaces
- networkpolicies
- nodes
- persistentvolumeclaims
- persistentvolumes
- pods
- replicas
- replicationcontrollers
- resourcequotas
- secrets
- services
- statefulsets
- storageclasses

Configuring resources:
- Label
- Selector
- ConfigMap
- Secrets
- Quota

Network:
- NodePort
- ClusterIP
- LoadBalancer
- ExternalName

Storage:
- Static provisioning
- Dynamic provisioning
- StorageClass

Closely Related

Deploying:
- Helm
- Tiller
- Helm packages
- charts
- public registry
- local private registry
- rolling update
- canary deploy
- blue-green deploy

Network:
- Container Network Interface (CNI)
- isolated subnet
- network isolation
- network policies
- DNS
- virtual IP
- NIC
kubectl examples

- Get the state of your cluster
  $ kubectl cluster-info

- Get all the nodes of your cluster
  $ kubectl get nodes -o wide

- Get info about the pods of your cluster
  $ kubectl get pods -o wide

- Get info about the replication controllers of your cluster
  $ kubectl get rc -o wide

- Get info about the services of your cluster
  $ kubectl get services

- Get full config info about a Service
  $ kubectl get service NAME_OF_SERVICE -o json

- Get the IP of a Pod
  $ kubectl get pod NAME_OF POD -template={{.status.podIP}}

- Delete a Pod
  $ kubectl delete pod NAME

- Delete a Service
  $ kubectl delete service NAME_OF_SERVICE
Microservices
An engineering approach focused on decomposing an application into single-function modules with well defined interfaces which are independently deployed and operated by small teams who own 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 architecture

Simplistically, microservices architecture is about breaking down large silo applications into more manageable, fully decoupled pieces.

A microservice is a granular, decoupled component within a broader application.
Key tenets of a microservices architecture

1. Large monoliths are broken down into many small services
   - Each service runs in its own process
   - The applicable cloud rule is one service per container

2. Services are optimized for a single function
   - There is only one business function per service
   - The Single Responsibility Principle: A class (microservice) should have one, and only one, reason to change
   - Database per service is desired.

3. Communication via REST API and/or message brokers
   - Avoid tight coupling introduced by communication through a database

4. Per-service continuous integration and continuous deployment (CI/CD)
   - Services evolve at different rates
   - You let the system evolve but set architectural principles to guide that evolution

5. Per-service high availability (HA) and clustering decisions
   - One size or scaling policy is not appropriate for all
   - Not all services need to scale; others require autoscaling up to large numbers
Advantages of microservices

- In a word: **Independent**
- Developed by a single team
  - Small enough for a team fewer than 20 people
  - All team members can understand the entire code base
- Developed independently
  - Limited, explicit dependencies on other services
- Developed on its own timetable
  - New versions delivered independently of other services
- Each can be developed in a different language (**Polyglot**)
  - Select the best language
- Manages its own data
  - Select the best technology and schema
- Scales and fails independently
  - Isolates problems

All of these tenets and advantages are only enabled if we have a platform that can take the microservices and allow enterprise grade and enterprise scale level deployment of microservices. This is the next step in what leads us to a **cloud-native** story.
Cloud Native

An application architecture designed to leverage the strengths and accommodate the challenges of a standardized cloud environment, including concepts such as elastic scaling, immutable deployment, disposable instances, and less predictable infrastructure.
Writing microservices when kubernetes is the target

Conceptual Assertions:

• Programmer’s shouldn’t have to know that their source code will run in a docker container orchestrated by kubernetes

Reality might be a little different:

• Experience shows that kubernetes and vendor offerings that include kubernetes:
  • Have more built in features to support microservices (logging, metrics, secrets, config maps, ..) – This is good.
  • Can support your own mesh, but that might start to detract from the advantages of microservices and leave you more cloud-enabled than cloud native

• Your programming framework might need to be integrated or wired down to kubernetes concepts

• Your build pipeline will need to not only create docker images, but also create declarative configuration for kubernetes deployments.

• Some other guidelines are probably appropriate:
  • Use standard DNS lookups for service to service REST calls
  • Keep an eye on things like istio, which will standardize sidecars, offer a number of features like version aware routing.
  • Watch out for solutions that obfuscate or hide kubernetes
Intelligent Routing and Load Balancing

Resiliency across Languages and Platforms

Fleet Wide Policy Enforcement

In-Depth Telemetry and Reporting
Demonstration
IBM Cloud Private - A Cloud Native platform with Enterprise grade content

IBM Middleware, Data, Analytics and Developer Services
Cloud enabled middleware, messaging, databases, analytics, and cognitive services to optimize current investments while rapidly innovating.

Core Operational Services
Simplify Operations Management, Security, and Hybrid integration across your Hybrid IT environment.

Kubernetes-based Container Platform
Industry leading container orchestration platform across private, dedicated and public clouds.

Runs on existing IaaS:
- VMware
- openstack
- Power Systems
- System Z
- IBM Spectrum

Third Party alliances:
- Dell
- Cisco
- NetApp
- Lenovo
- Canonical

Enterprise grade services for Middleware, Data, Analytics, DevOps
Enterprise grade operations, across your Hybrid IT environment
Open by design, preventing vendor lock-in
Leverage existing investments
Stock Trader App – What We Created – Hybrid App

Browser → Web App → Portfolio → DB2 → Stock Quote → Public Cloud

Private Cloud:
- DB2
- Redis
- MQ
- Notification

Public Cloud:
- API Connect
- Quandl
- Open Whisk
- Slack

Integration:
- GET/POST
- JDBC
- JMS
- onMessage
- Quandl
- Slack

Library/Service:
- Quandl
- Slack
Demo Here
IBM WebSphere Liberty delivers the foundation of innovation

- Five years ago IBM reinvented the application server for the cloud with IBM® WebSphere® Liberty
  At that time:
  - Open source was starting to receive widespread acceptance in business
  - Microservices were displaying future potential
  - Container technology was emerging

*Key areas we are evolving with Liberty to bring more disruptive value to developers...*

<table>
<thead>
<tr>
<th>Open source</th>
<th>Microservices</th>
<th>Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>The essential Liberty runtime and IBM’s JVM are now open source projects, still fully supported with commercial licenses.</td>
<td>WebSphere provides full Java™ EE, but cloud-native apps need new fault tolerant capabilities that are not yet available from the Java EE spec.</td>
<td>Available as pre-built containers from IBM, WebSphere Liberty is ready for production deployment into virtually any cloud environment.</td>
</tr>
</tbody>
</table>
Eclipse MicroProfile: Microservice innovation

- Vendor-neutral programming model, designed in the open, for Java microservices
- Provide core capabilities for building fault tolerant, scalable, microservices
- Increasing the rate and pace of innovation beyond Java EE

Standardizing microservices in enterprise Java via the MicroProfile community

<table>
<thead>
<tr>
<th>Config</th>
<th>Fault Tolerance</th>
<th>Health Check</th>
<th>Health Metrics</th>
<th>JWT Propagation</th>
</tr>
</thead>
<tbody>
<tr>
<td>externalize configuration to improve portability</td>
<td>build robust behavior to cope with unexpected failures</td>
<td>common format to determine service availability</td>
<td>common REST endpoints for monitoring service health</td>
<td>interoperable authentication and role-based access control</td>
</tr>
</tbody>
</table>

MicroProfile v1.2 – Released September 2017
http://microprofile.io/
Microservice Builder:
Create, Configure and Deploy microservices for IBM Cloud

In 3 steps
Create and run your microservices, hybrid and containerized apps

- Innovate with SPEED
Set up your environment, fabric and DevOps pipeline in minutes using Microservice Builder.

- Helm charts for fabric and pipeline
  - delivered with IBM Cloud private
  - available for any Kubernetes environment
  - support entitled with WebSphere Application Server licenses.

https://developer.ibm.com/microservice-builder
Open Liberty
A lightweight open source server runtime ideal for building Java™ microservices and cloud-native apps

- Easy to consume
- Deploy on any cloud for Java™
- Seamlessly transition to WebSphere

https://openliberty.io/
Solution View
Hybrid Cloud – What kinds of capabilities?

- **Service on a Cloud**
  - Capabilities to which customer subscribes. (e.g., Watson services, Weather, ..)
  - Components utilize these as a service.
- **Service on private cloud**
  - Capabilities installed on local ICP.
  - Purpose is to support components.
  - Installed via Helm charts
  - Managed via K8s and consoles
  - Components utilize these
  - Probably a shared service
- **New microservice**
  - Surrounds customer-written logic.
  - Usually microservices, but can be large monoliths. (e.g., java, node.js)
  - Created & installed as part of a build
  - Strong affinity to the overall solution of which they are a part

**App Architecture of StockTrader Example**

- Browser -> Web App -> Portfolio -> Stock Quote
- Github (GHE) -> MSB -> MQ
- DB2 -> Redis
- Loyalty Level -> Notification
- API Connect -> Slack
- Open Whisk -> Quandl

**ICP Roles**

- **Todd**
  - Operations / Admin
  - Responsible for infrastructure, security, and management of the environment.

- **Jane**
  - Enterprise Developer
  - Responsible for modernizing existing apps and creating new Cloud Native Workloads.
Application or Solution Architecture – StockTrader Original

This is a demo where the components are built as Java micro-services.

Most services used are also local on the same IBM Cloud private.
MQ and DB2 are now used in their current on-premises formats.

Consider for production, while leveraging MQ and Db2 in IBM Cloud Private for dev/test.
Add voice capabilities

Orchestrator
PII Scrubber
Chat analytics
Redis
Postgres
IBM Voice Gateway

Chat UI
Crawler
API Connect
DataPower
ECM

Watson Conversation
Watson Discovery
Watson Text To Speech
Watson Speech To Text

Existing on premises environment

Get Balance
Pay Card
Activate Card
IIB
MQ and DB2 are now used in their current on-premises formats and there are existing queues and databases in place already for this monolithic application to use.

**Transformation Advisor** will target this type of application, showing modifications to get it to run in Liberty.

**Web Apps** (monolithic but on Liberty via some development work maybe that updated it to fit into liberty programming model)
Instead of putting all web apps onto a single Liberty instance, separation occurs as part of the transformation. This isn’t micro-services, but it is some level of refactoring.

Separate life-cycle management is now possible – This is a better option, assuming there are not a lot of shared dependencies.
Conclusion
Conclusion

Microservices leads to the need for:

- Containers (Docker) and container orchestration (Kubernetes)
- A full platform for Cloud Native
- Considerations for existing middleware and applications (Cloud-Enabled) as part of the solution

- Capitalizing on the microservices approach requires all of the above
- Kubernetes is a key component of building out that cloud-native platform
- Kubernetes combined with other purpose built supporting services and integrations to existing platform operational capabilities gets you further.
- It also requires cultural change
References

• Introductory Blog Entry showing IBM Cloud private
  • https://t.co/ST1FKnkaTM

• Video showing IBM Cloud private
  • https://t.co/KC4H3pSuLL

• Building and running a solution on IBM Cloud Private
  • https://ibm.co/2vGNtZl

• Developing microservices in IBM Cloud Private
  • https://ibm.co/2wjvGJ8

• Transformation Advisor
  • https://www.ibm.com/developerworks/community/blogs/5092bd93-e659-4f89-8de2-a7ac980487f0/entry/Introducing_Transformation_Advisor?lang=en

• Knowledge Center
Learn more and try IBM Cloud Private today

Learn more and try the IBM Cloud Private Community Edition


BackUp
Combining Docker and Kubernetes to deliver powerful tools, an intuitive user experience, and built-in security and isolation to enable rapid delivery of applications - all while leveraging IBM Cloud Services including cognitive capabilities from Watson.

https://www.ibm.com/cloud/container-service