Openshift on IBM Cloud Workshop

Jaffa Sztejnbok
Developer Advocate
IBM

jaffas@il.ibm.com

IBM **Developer**

IBM

Presentation and Lab

https://github.com/jaffaszt/jaffa-openshift101

Register IBM Cloud

1. Go to https://github.com/jaffaszt/jaffa-openshift101
2. **Run Step 1 : Register to IBM Cloud https://github.com/jaffaszt/jaffa-openshift101/tree/master/workshop**

IBM Developer

IBM Alpha Zone accelerator

Startup & Developer – IBM Israel

Agenda

Containers

Docker

Orchestration

Kubernetes

Red Hat OpenShift

Lab Materials

CNCF SURVEY 2019

Deployments are getting larger as cloud native adoption becomes mainstream

© 2020 IBM Corporation

https://www.cncf.io/wp-content/uploads/2020/03/CNCF_Survey_Report.pdf

Container Usage is growing

What are Containers?

OS level virtualization

- Isolate user space instances on a single host machine
- Gives isolated view of processes, user space and file system for user owner
- Share host Linux kernel
- A self-contained sealed unit of software
- Contains everything required to run the code

A Container Includes :
Code, configs, Processes, Networking, Dependencies, OS (just enough)

VM vs Container

Traditional Deployment **Virtualized Deployment** **Container Deployment**

Containers: Dev vs Ops

DEV	**OPS**
Code	Logging
Libraries	Remote Access
Config	Network Config
Runtime	Monitoring
OS	

Why Containers?

- Agile
- Continuous Deployment
- Dev and Ops
- Observability
- Consistency
- Management
- Microservices
- Resource Isolation
- Resource Utilization
- Portable
- Easy to manage
- Containers provide “just enough” isolation
- Immutable

Containers

Any language

Any library

Any binary

Ecosystem of base images

Industry standard

MONOLITHIC ARCHITECTURE

MICROSERVICE ARCHITECTURE

Martin Fowler: Microservices

“In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies. “

Microservices and Containers

- are nearly synonymous
- microservices must be separately deployable, scalable independent units
- a microservice can be packed into a **Container image** and isolated as a **container**
- this way, you can build an application that is independent of your host environment.

Open Container Initiative – OCI

THE LINUX FOUNDATION PROJECTS

OPEN CONTAINER INITIATIVE

About Community Join Blog News FAQ Contact Us Release Notices

OPEN CONTAINER INITIATIVE

AN OPEN GOVERNANCE STRUCTURE FOR THE EXPRESS PURPOSE OF CREATING OPEN INDUSTRY STANDARDS AROUND CONTAINER FORMATS AND RUNTIME

What is Docker?

Docker is the world's leading containerization standard platform

Docker is a platform for developing, shipping and running applications using **container** virtualization technology.

What is Docker?

Image
Template to use for creating containers. Snapshot of Read-Only stored in Docker Hub。

Container
Basic unit on which application service runs。

Docker Hub
Store, distribute and share images of containers.
Establish and use in SaaS or enterprise.

Docker Repository
Pulling an image from Docker Trusted Registry is the same as pulling an image from Docker Hub or any other registry. Since DTR is secure by default, you always need to authenticate before pulling images.。

Docker Engine
A program for creating containers and running them.
Run on a physical server or virtual machine.
User can operate with command。

Docker :

Docker makes it really easy to install and run software without worrying about setup or dependencies

Common images available on the cloud

Dockerfile

```
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

Each instruction creates one layer:

- `FROM` creates a layer from the `ubuntu:18.04` Docker image.
- `COPY` adds files from your Docker client's current directory.
- `RUN` builds your application with `make`.
- `CMD` specifies what command to run within the container.

artifactory
npm
$ docker build
Dockerfile
WorkDIR
Images
node
Run
Containers
local docker instance
My Computer
Docker registry

Images
Run
Containers
local docker instance
$ docker run hello-world
Docker registry
My Computer

Docker Commands

Example：

command	description
run	Run a new container from the image
start	Start a container that has stopped
stop	Stop a running container
exec	Execute a **second** command in the running container
ps	Browse the system's container
images	Browse the system image
build	Build the container from the build script Dockerfile
inspect	Check the container's system information
logs	Check the container's log

https://docs.docker.com/engine/reference/commandline/docker/

What is podman and buildha?

podman

buildah

https://developers.redhat.com/blog/2019/02/21/podman-and-buildah-for-docker-users/

Kaniko

How to manage containers?

How to manage containers?

amazon.com

NETFLIX

Container Orchestration !

- Scheduling
- Cluster management
- Service discovery
- Provisioning
- Monitoring
- Configuration management

What is Kubernetes?

- **Container Orchestrator**
  - Provision, manage, scale applications
- **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
- **What's in a name?**
  - Kubernetes (K8s/Kube): "Helmsman" in ancient Greek

What is Kubernetes?

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 pluggability and extensibility is part of its "secret sauce".

DB represents the user's desired state
Watchers attempt to make reality match the desired state

"API Server" is the HTTP/REST front-end to the DB

More on controllers later...

Kubernetes Architecture: How apps are accessed

Pod

- Smallest deployment unit – runs containers
- Each pod has its own IP
- Shares a PID namespace, network, and hostname

Service

- Collection of pods exposed as an endpoint
  - state and networking info propagated to all worker nodes
- Types of service exposure
  - ClusterIP – Exposes cluster-internal IP
  - NodePort – Exposes the service on each Node's IP at a static port
  - LoadBalancer – Exposes externally using a cloud provider's load balancer
  - ExternalName – Maps to an external name (such as foo.bar.example.com)

Kubernetes resource model

- A resource for every purpose

Config Maps
Daemon Sets
Deployments
Events
Endpoints
Ingress
Jobs
Nodes
Namespaces
Pods
Persistent Volumes
Replica Sets
Secrets
Service Accounts
Services
Stateful Sets, and more...

Kubernetes aims to have the building blocks on which you build a cloud native platform.

Therefore, the internal resource model **is** the same as the end user resource model.

Key Resources

Pod: set of co-located containers

- Smallest unit of deployment
- Several types of resources to help manage them
- **Replica Sets, Deployments**, Stateful Sets, ...

Services

- Define how to expose your app as a DNS entry
- Query based selector to choose which pods apply

Kubernetes in Action – Deployment

Kubernetes in Action – Deployment

Users > jaffa > istio101 > workshop > guestbook > v2 > ! guestbook-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: guestbook-v2
  labels:
    app: guestbook
    version: "2.0"
spec:
  selector:
    matchLabels:
      app: guestbook
  replicas: 3
  template:
    metadata:
      labels:
        app: guestbook
        version: "2.0"
    spec:
      containers:
      - name: guestbook
        image: ibmcom/guestbook:v2
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        ports:
        - name: http
          containerPort: 3000
```

Kubernetes in Action – Service

Users > jaffa > istio101 > workshop > guestbook > v2 > ! guestbook-service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: guestbook
  labels:
    app: guestbook
spec:
  # comment or delete the following line if you want to use a LoadBalancer
  # type: NodePort
  # if your cluster supports it, uncomment the following to automatically create
  # an external load-balanced IP for the frontend service.
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 3000
    name: http
  selector:
    app: guestbook
```

What Does Kubernetes Do?

- Service Discovery and Load Balancing
- Storage Orchestration
- Rollouts/Rollbacks
- Best use of your resources
- Self-Healing
- Secret/Config Management

What Doesn’t Kubernetes Do?

- Define Application Types
- Deploy Code CI/CD
- Application-Level Services
- Logging/Monitoring/Alerting
- Config
- Machine Management

Why Openshift ?

Red Hat OpenShift

OpenShift Architectural Overview

OpenShift Overview

- Container Host & Runtime
- Enterprise Kubernetes Product
- Validated Integrations
- Integrated Container Registry
- Developer Workflows
- Access to Services
- Security
- Support

Why Openshift ?

- OKD (Origin Community Distribution)
- github.com/openshift/origin
- 30,000+ commits, 350+ contributors

Workshop website

https://github.com/jaffaszt/jaffa-openshift101

IBM