Docker Intro

DevOps Course

JOHN BRYCE
Leading in IT Education
matrix company

JOHN BRYCE
Leading in IT Education
matrix company

Yaniv Cohen

PortFolio

https://goo.gl/9ivmRP

Linkedin

https://www.linkedin.com/in/yanivos/

Objectives

JOHN BRYCE
Leading in IT Education
matrix company

- By the end of this session
  - You'll be familiar with Docker Concepts & Base Commands
  - Configure Dockers using DockerFile And Passing Properties To It
  - Run Standalone Jar in docker
  - Operate Docker Hub (Push)
  - Build Docker Image with Maven

Questions for you...

JOHN BRYCE
Leading in IT Education
matrix company

- What Do You Know About Docker?
- Who Used Docker For Development / QA / STG / PROD?
- Who Tried & Failed Implementing Docker

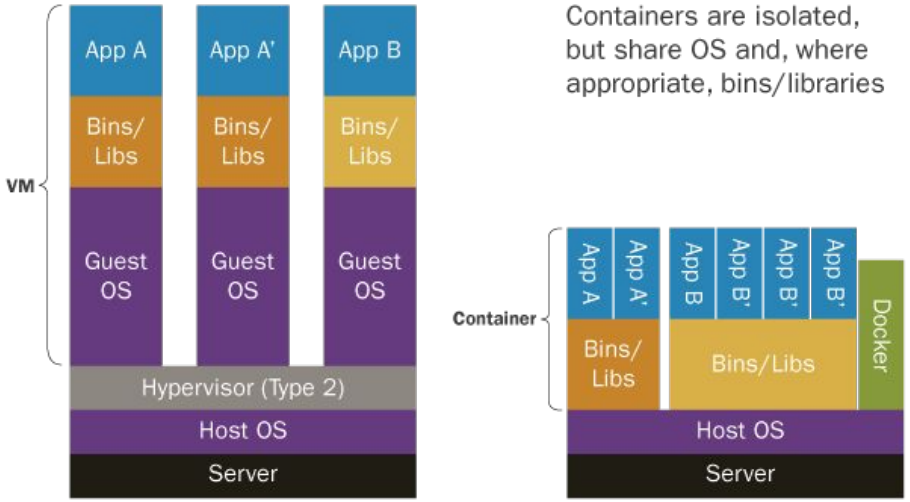
What is Docker

JOHN BRYCE
Leading in IT Education
matrix company

Docker is an open platform for developing, shipping, and running applications.

Docker allows you to package an application with all of its dependencies into a standardized unit for software development.

Containers VS. VMs



Virtual Machines

Containers

Docker Benefits Upon VMs

JOHN BRYCE
Leading in IT Education
matrix company

- Small to tiny images - Few hundred MB's for OS + Application (5MB for full OS - Alpine) VS. Gigabytes in VM's
- Very small footprint on the host machine (CPU, RAM Impact) as Docker only use what it required instead of building a complete Operating system per VM.
- Containers use up only as many system resources as they need at a given time. VMs usually require some resources to be permanently allocated before the virtual machine starts.
- Direct hardware access. Applications running inside virtual machines generally cannot access hardware like graphics cards on the host in order to speed processing. Containers Can (ex. Nvidia)
- Microservice in nature and integrations (API's) for whatever task required.
- Portable, Fast (Deployments , Migration , Restarts and Rollbacks) and Secure
- Can run anywhere and everywhere
- Simplify DevOps
- Version controlled
- Open Source

Common Use Cases for Docker

JOHN BRYCE
Leading in IT Education
matrix company

- CI / CD
- Fast Scaling application layers for overcoming application performance limitations.
- For Sandboxed environments (Development, Testing , Debugging)
- Local development environment (no more “ It run on my laptop…”)
- Infrastructure as a CODE made easy with docker
- Multi-Tier applications (Front End , Mid Tier (Biz Logic) , Data Tier) / Microservices
- Building PaaS , Saas

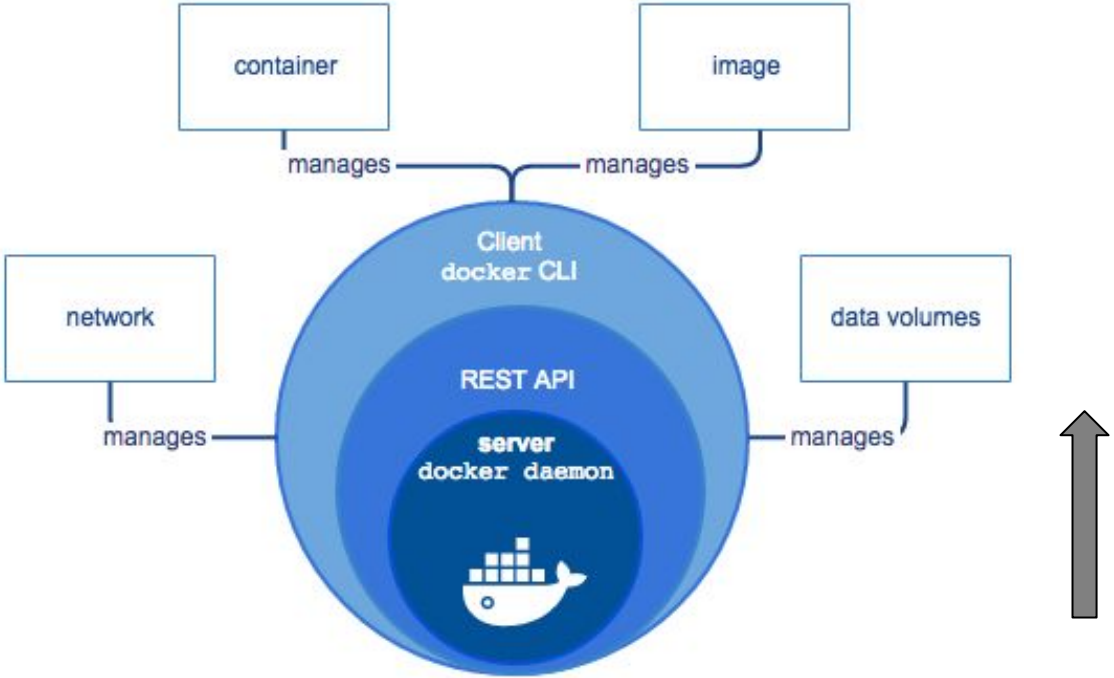
Under The Hood

- Architecture: Linux X86-64
- Written in: GoLang (On March 13, 2014, with the release of version 0.9, Docker dropped LXC as the default execution environment which an operating system level virtualization and replaced it with its own libcontainer library written in the Go programming language)
- Engine: Client - Server (Daemon) Architecture
- Namespace: Isolation of process in linux where one process cant “See” the other process
- Control Groups: Linux Kernel capability to limit and isolate the resource usage (CPU, RAM, disk I/O, network etc..) of a collection of process
- Container format: libcontainer - Go implementation for creating containers with namespaces, control groups and File system capabilities access control

Docker Architecture

JOHN BRYCE
Leading in IT Education
matrix company

Overview



What is docker - Technical Aspect

JOHN BRYCE
Leading in IT Education
matrix company

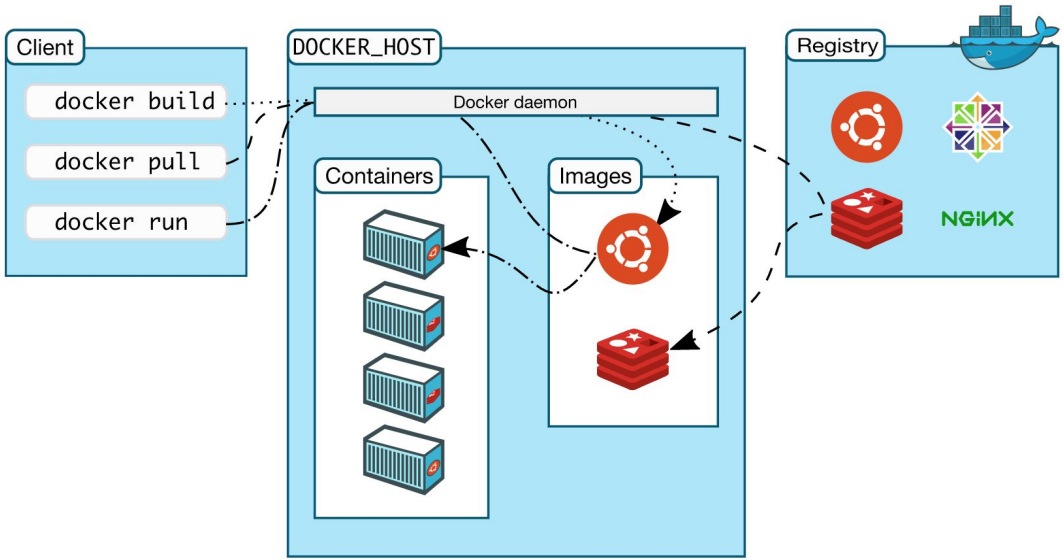
Docker Architecture

Docker uses a client-server architecture. The Docker *client* talks to the Docker *daemon*, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.

Docker Architecture

JOHN BRYCE
Leading in IT Education
matrix company

Docker Architecture



Docker Components

- Engine
- Daemon
- (Docker) Client
- Docker Registries
- Docker Objects
- Machine
- Compose
- Swarm

Docker Components

JOHN BRYCE
Leading in IT Education
matrix company

Engine

- A server which is a type of long-running program called a daemon process (the dockerd command).
- A REST API which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
- A command line interface (CLI) client (the docker command).

Docker Components

JOHN BRYCE
Leading in IT Education
matrix company

Daemon

- The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with **other daemons** to manage Docker services.

Docker Components

JOHN BRYCE
Leading in IT Education
matrix company

Docker Client

- The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The **Docker client can communicate with more than one daemon.**

Docker Components

JOHN BRYCE
Leading in IT Education
matrix company

Docker Registries

- A Docker registry stores Docker images. Docker Hub and Docker Cloud are public registries that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.
- When one use “docker pull / push / run” commands, the required images are pulled from the configured registry.

Docker Components

JOHN BRYCE
Leading in IT Education
matrix company

Docker Objects

- Images
  a. Read Only template with instruction for creating a Docker Container. Often, an Image is based on another image with some additional customization.
  b. Self own images that are fully created by you using DockerFile with a simple syntax where every instruction control a different Layer in the image. Once a change is made to a specific layer, a rebuild of the image will change only the updated layers. This what makes images small, fast and lightweight in compared to other virtualization solutions

Docker Components

JOHN BRYCE
Leading in IT Education
matrix company

Docker Objects

- Containers
  a. A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.
  b. Container is defined by its image as a well as any configuration options we provide to it when created or when we start it

Docker Components

JOHN BRYCE
Leading in IT Education
matrix company

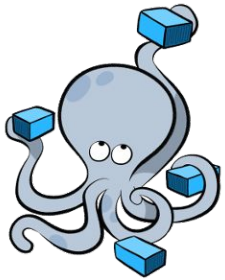
Docker Objects

- Services
    a. Allow you to scale containers across multiple Docker daemons, which all work together as a **swarm** with multiple managers and workers. Each member of a swarm is a Docker daemon, and the daemons all communicate using the Docker API. A service allows you to define the desired state, such as the number of replicas of the service that must be available at any given time. By default, the service is load-balanced across all worker nodes. To the consumer, the Docker service appears to be a single application. Docker Engine supports swarm mode in Docker 1.12 and higher.

Docker Components

JOHN BRYCE
Leading in IT Education
matrix company

Docker Compose



A tool for defining and running complex applications with

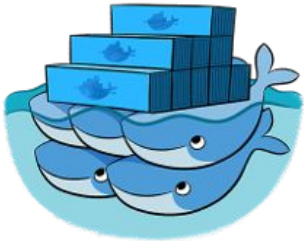
Docker (eg multi-container application ex. LAMP)

With a single file

Docker Components

JOHN BRYCE
Leading in IT Education
matrix company

Docker Swarm



A Native Clustering tool for Docker. Swarm pools together several Docker hosts and exposed them as a single virtual Docker host. It scale up to multiple hosts

Docker Components

JOHN BRYCE
Leading in IT Education
matrix company

Good to know:

Docker Machine

A Tool which makes it easy to create Docker Hosts on Operating systems that does not support docker natively, or on cloud providers and inside your datacenter.

JOHN BRYCE
Leading in IT Education
matrix company

INSTALLING DOCKER

Windows 10 Enterprise / Educational

DOWNLOAD HERE

JOHN BRYCE
Leading in IT Education
matrix company

Let's Start

Docker Flow

```
docker run -i -t -d --name dockerlearning -p 8080:80 alpine:latest sh
```

- ‘docker run’ will run the container
- This will not restart an already running container, just create a new one
- docker run [options] IMAGE [command] [arguments]
    a. [options] modify the docker process for this container
    b. IMAGE is the image to use
    c. [command] is the command to run inside the container (entry point to hold the container running)
    d. [arguments] are arguments for the command

Docker Flow

```
docker run -i -t -d --name dockerlearning -p 8080:80 alpine:latest sh
```

- ‘docker run’ will run the container
  a. -i - Interactive mode
  b. -t - Allocate pseudo TTY - or not Terminal will be available
  c. -d - Run in the background (Daemon style)
  d. --name - Give the container a name or let Docker to name it
  e. -p [local port] : [container port] - Forward local port to the container port

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                    NAMES
98debfed4458        alpine:latest       "sh"                Less than a second ago   Up 1 second         0.0.0.0:8080->80/tcp     dockerlearning
```

Docker Flow

```
docker run -i -t -d --name dockerlearning -p 8080:80 alpine:latest sh
```

- Pulls the alpine:latest image from the registry (if not existed on our station)
  a. Run “docker images” to see what images already downloaded / in use locally
- Creates new container
- Allocate FS and Mounts a read-write Layer
- Allocates network/bridge interface
- Set up an IP Address
- Executes a process that we specify (in this scenario - “sh” as alpine release doesn't have bash)
- Captures and provides application outputs

Docker Examples

JOHN BRYCE
Leading in IT Education
matrix company

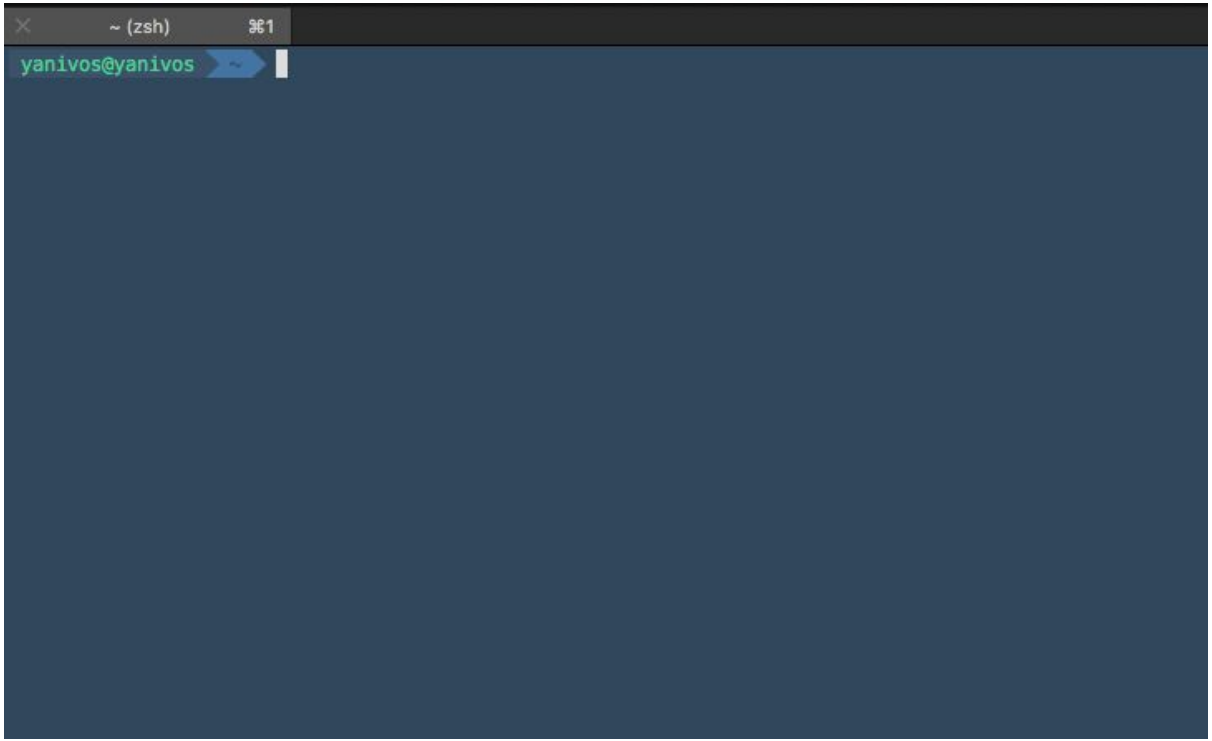
- Pull / Run an image
- SSH into a container
- View Logs
- Docker Volume
- Using Dockerfile - Building our own Jar
- Package an app and push it to a repo

Common Docker Commands

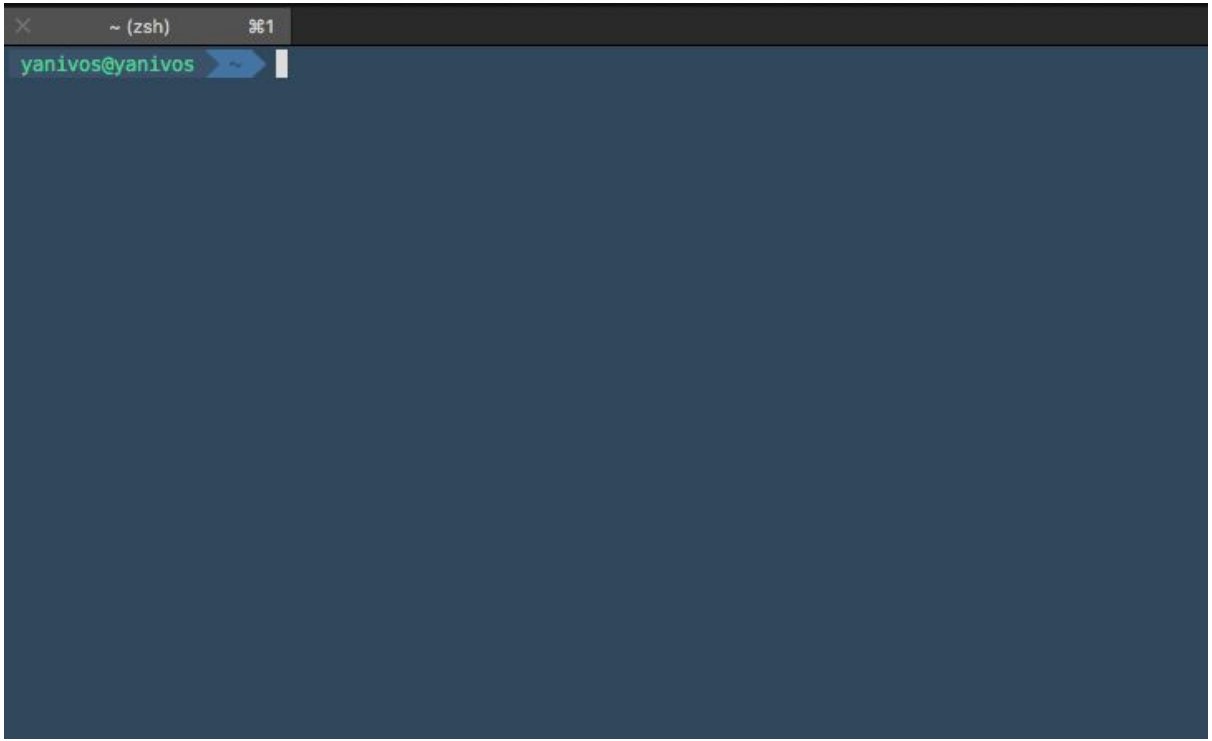
JOHN BRYCE
Leading in IT Education
matrix company

```
// General info
man docker // man docker-run
docker help // docker help run
docker info
docker version
docker network ls
// Images
docker images // docker [IMAGE_NAME]
docker pull [IMAGE] // docker push [IMAGE]
// Containers
docker run
docker ps // docker ps -a, docker ps -l
docker stop/start/restart [CONTAINER]
docker stats [CONTAINER]
docker top [CONTAINER]
docker port [CONTAINER]
docker inspect [CONTAINER]
docker inspect -f "{{ .State.StartedAt }}" [CONTAINER]
docker rm [CONTAINER]
```

Running simple shell



Building & Running Mysql On docker



Why not to run SSH inside a container

JOHN BRYCE
Leading in IT Education
matrix company

- We can…
- Docker is designed for one command per container - Now we run two
- If any update or modification is needed, We need to change our setup and not the docker image..
- If you still want to review something… SSH it.

JOHN BRYCE
Leading in IT Education
matrix company

Building Our Own Jar Application

In a Docker Container

Building Our Own Jar Application

JOHN BRYCE
Leading in IT Education
matrix company

In a Docker Container

For this example we will build a simple SpringBoot-rest application downloaded from here

Later in your LAB you will use your own SpringBoot artifact that you created in previous class.

Building Our Own Jar Application

JOHN BRYCE
Leading in IT Education
matrix company

Creating sample Jar

```
## Java Class
public class HelloWorld {
 public static void main(String[] args){
   System.out.println("Hello World :) ");
 }
}

# Compile
javac HelloWorld.java

# Create  manifest.txt

    Manifest-Version: 1.0
    Created-By: Me
    Main-Class: HelloWorld
```

```
# Create Jar file

jar cfm HelloWorld.jar manifest.txt HelloWorld.class

# Check if working
java -jar HelloWorld.jar
```

Spring Boot Application On Docker

JOHN BRYCE
Leading in IT Education
matrix company

Creating dockerfile with a demo war / jar

```
# Java8 Alpine Release
FROM frolvlad/alpine-oraclejdk8:slim

# configure WorkDir
WORKDIR /app

# Mount HOST Folder
VOLUME /app

# Copy Spring Boot File to target
COPY jar/spring-boot-rest-example/target/spring-boot-rest-example-0.4.0.war
/app/spring-boot-rest-example-0.4.0.war

#Expose Ports - ONLY EXPOSED - IT'S NOT Mapped. -p will be needed on run
EXPOSE 8091
EXPOSE 8090

# Command to run (Entry Point)
CMD java -jar -Dspring.profiles.active=test /app/spring-boot-rest-example-0.4.0.war
```

Spring Boot Application On Docker

JOHN BRYCE
Leading in IT Education
matrix company

Building the dockerfile in CLI

```
# Build dockerfile
# docker build -t [repo/imagename:tag] [dockerfile location]
docker build -t yanivomc/dockerspringboot .

# Run image created above
docker run -p 8091:8091 -p 8090:8090 --rm -t -i --name springboot
yanivomc/dockerspringboot:latest
```

Spring Boot Application On Docker

JOHN BRYCE
Leading in IT Education
matrix company

Running the Image

```
# Command to run (Entry Point)
docker run -p 8091:8091 -p 8090:8090 --rm -t -i --name springboot yanivomc/dockerspringboot:latest
```

```
ramework.boot.actuate.endpoint.mvc.EndpointMvcAdapter.invoke()
2018-01-13 22:52:12.211  INFO 7 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public java.util.Map<j
2018-01-13 22:52:12.217  INFO 7 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler of t
2018-01-13 22:52:12.217  INFO 7 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [cla
2018-01-13 22:52:12.231  INFO 7 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframew
arent: org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@4aa8f0b4
2018-01-13 22:52:12.318  INFO 7 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8091 (http)
2018-01-13 22:52:12.323  INFO 7 --- [           main] o.s.c.support.DefaultLifecycleProcessor  : Starting beans in phase 2147483647
2018-01-13 22:52:12.324  INFO 7 --- [           main] d.s.w.p.DocumentationPluginsBootstrapper : Context refreshed
2018-01-13 22:52:12.341  INFO 7 --- [           main] d.s.w.p.DocumentationPluginsBootstrapper : Found 1 custom documentation plugin(s)
2018-01-13 22:52:12.351  INFO 7 --- [           main] s.d.s.w.s.ApiListingReferenceScanner     : Scanning for api listing references
2018-01-13 22:52:12.517  INFO 7 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8090 (http)
```

Push our docker image to docker hub

JOHN BRYCE
Leading in IT Education
matrix company

1. Create new Repo in docker hub
2. Register your newly created repo and login to it in CLI

   “docker login”
3. Push your created image to your repo

   “docker push repo/image:tag” >> “docker push yanivomc/learningdocker:latest”

JOHN BRYCE
Leading in IT Education
matrix company

Maven style

Building and Pushing Docker image to automate build process

JOHN BRYCE
Leading in IT Education
matrix company

Clone Git

git clone https://github.com/yanivomc/seminars.git

MAVEN & DOCKER

JOHN BRYCE
Leading in IT Education
matrix company

Using Spotify Maven Plugin, Build , Deploy and Push Docker Image post build becomes extremely easy

Once configured we can run: mvn clean package docker:build

```
    <!-- Docker Build -->
<plugins>
    <plugin>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.0.0</version>
    </plugin>
    <plugin>
        <groupId>com.spotify</groupId>
        <artifactId>docker-maven-plugin</artifactId>
        <version>0.4.10</version>
        <configuration>
            <imageName>yanivomc/${project.dockerArtifactId}</imageName>
            <dockerDirectory>dockerfile</dockerDirectory>
            <resources>
                <resource>
                    <targetPath>/</targetPath>
                    <directory>${project.build.directory}</directory>
                    <include>${project.artifactId}-${project.version}.${project.packaging}</include>
                </resource>
            </resources>
        </configuration>
    </plugin>
```

Where to go next ?

JOHN BRYCE
Leading in IT Education
matrix company

Type	Software
Clustering/orchestration	Swarm, Kubernetes, Marathon, MaestroNG, decking, shipyard
Docker registries	Portus, Docker Distribution, hub.docker.com, quay.io, Google container registry, Artifactory, projectatomic.io
PaaS with Docker	Rancher, Tsuru, dokku, flynn, Octohost, DEIS
OS made of Containers	RancherOS

JOHN BRYCE
Leading in IT Education
matrix company

QUESTIONS ?

END

DevOps Course

JOHN BRYCE
Leading in IT Education
matrix company