Orchestrate Your Services on OpenShift using Spring Cloud Kubernetes

Capgemini Open Day
Poznań, 13th of June 2017

Krzysztof Sobkowiak (@ksobkowiak)

The Apache Software Foundation Member
Senior Solution Architect at Capgemini

Your Speaker

- FOSS enthusiast, evangelist & architect
- Senior Solution Architect at Capgemini
- The Apache Software Foundation
  - Member
  - Apache ServiceMix commiter & PMC chair (V.P. Apache ServiceMix)
  - active at Apache Karaf, CXF, Camel, ActiveMQ
- Member/developer at OASP, OPS4J

Views in this presentation are my personal views and do not necessarily reflect the views of Capgemini.

Creating business value through software is about speed, safety, iteration, and continuous improvement

Typical problems developing microservices

- How to run them all locally?
- How to package them (dependency management)
- How to test?
- Vagrant? VirtualBox? VMs?
- Specify configuration
- Process isolation
- Service discovery
- Multiple versions?

- Simple configuration
- Curated dependencies and transitive dependencies
- Built in metrics, monitoring
- Slim profile for deployment (…micro even?)

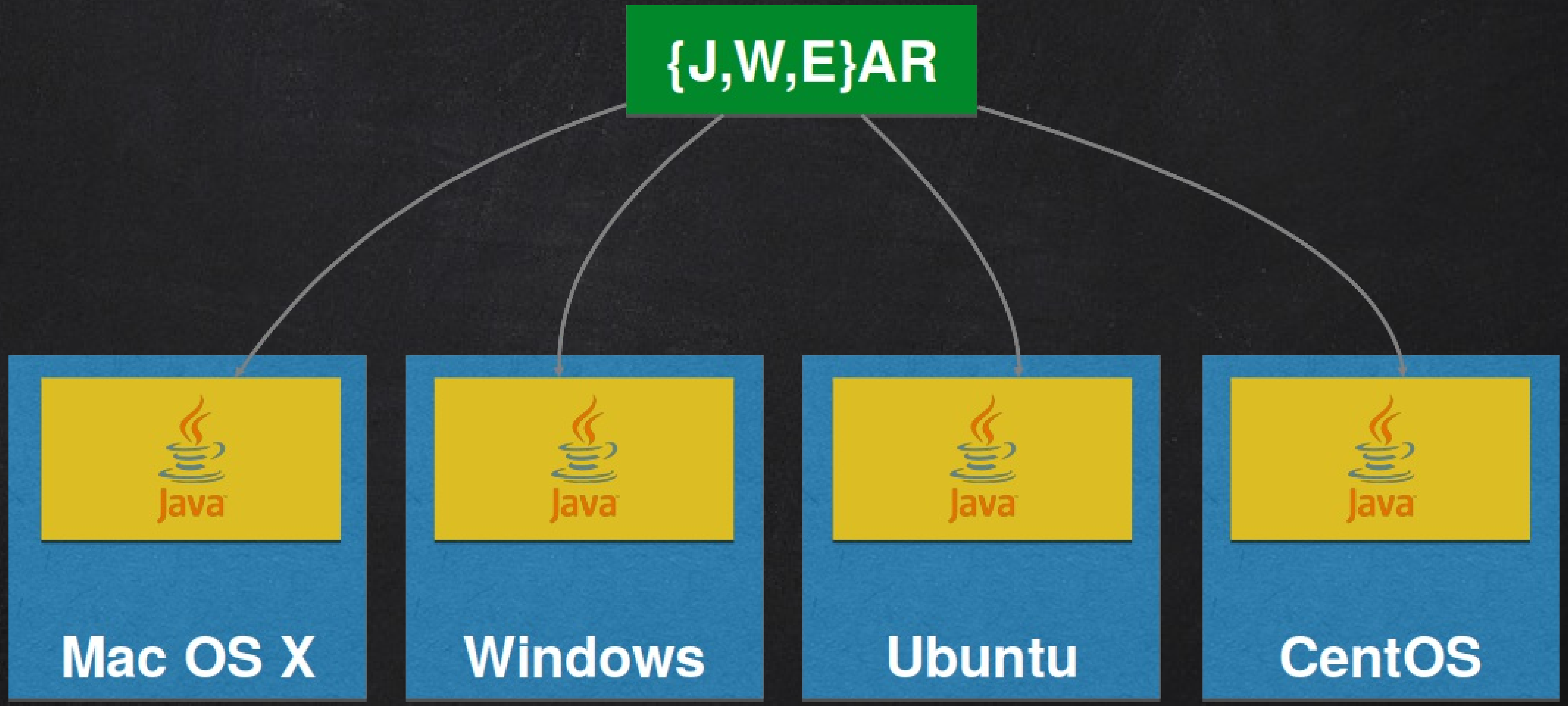
  #microprofile

Spring Boot

- It can be pretty small…
- Predefined packages/starters available
- Can generate WAR or JAR file

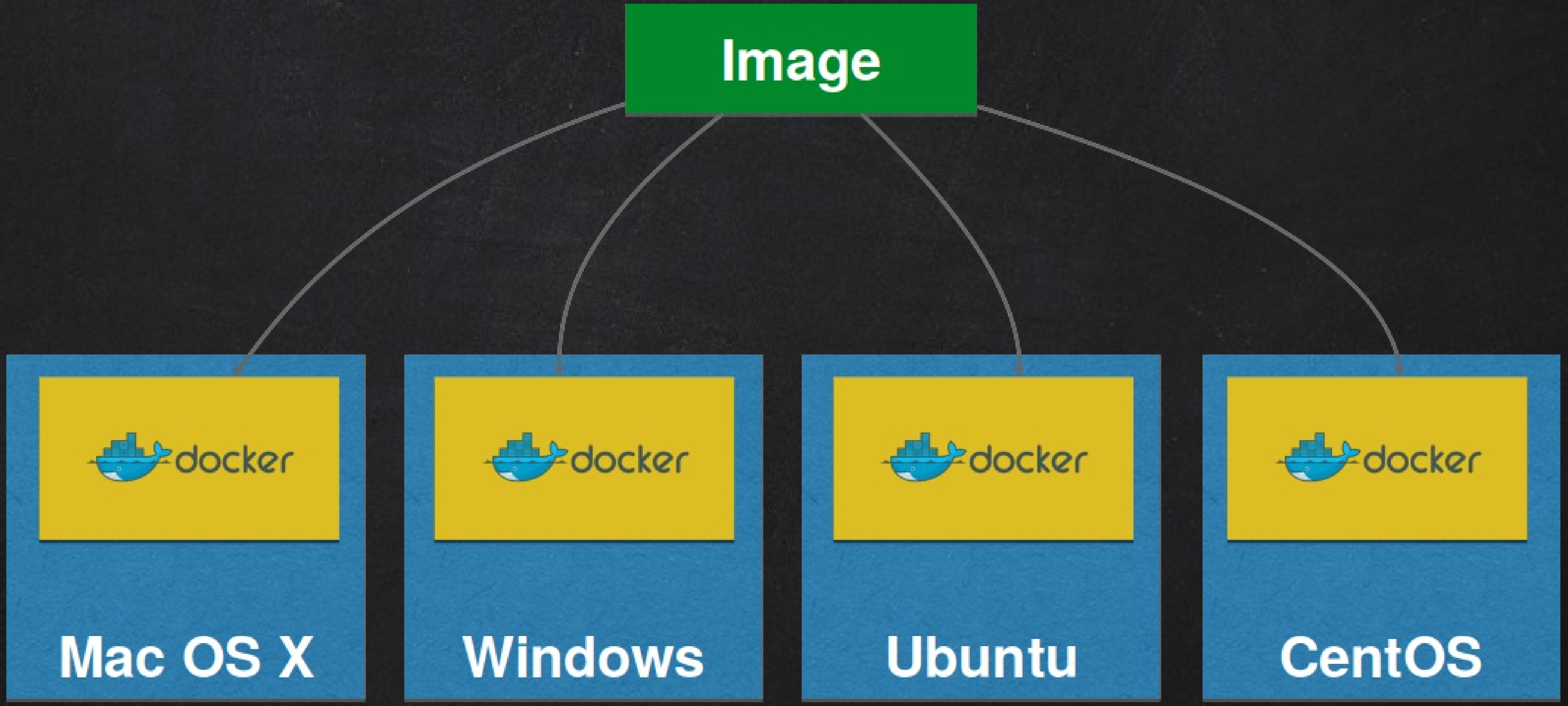
```
@RestController
@SpringBootApplication
public class ControllerAndMain {
  private int counter;

  @Autowired
  private Config config;

  @RequestMapping(value = "/ip", method = RequestMethod.GET)
  public IPAddress ipaddress() throws Exception {
    return new IPAddress(++counter,
      InetAddress.getLocalHost().getHostAddress(),
      config.getMessage());
  }

  public static void main(String[] args) {
    SpringApplication.run(
      ControllerAndMain.class, args);
  }
}
```

WORA = Write Once Run Anywhere

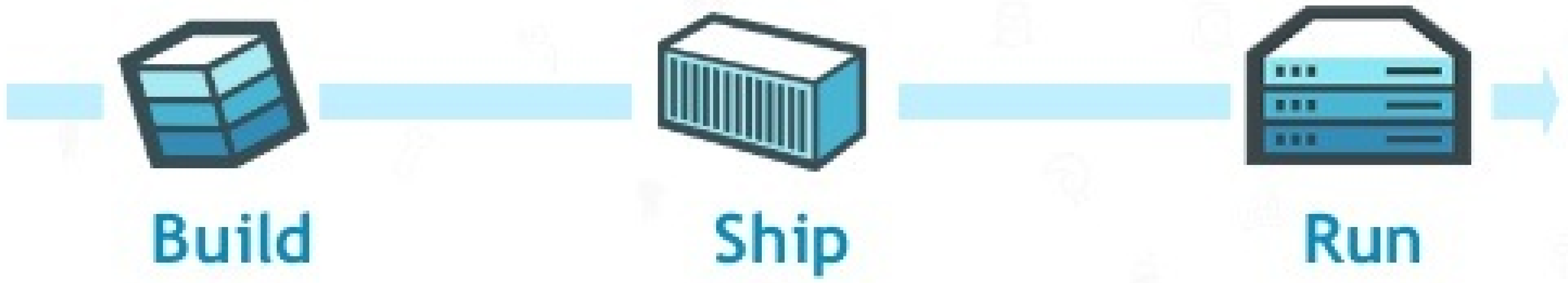


Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

PODA = Package Once Deploy Anywhere

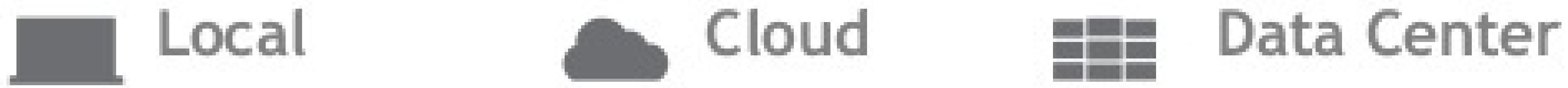


The Docker Mission



Any Application

Anywhere



Docker for Java Developers

- Dockerfile

```
FROM openjdk:latest

ADD target/ipservice-boot-docker-0.0.1-SNAPSHOT.jar .
EXPOSE 8090
CMD /usr/bin/java -Xmx400m -Xms400m -jar ipservice-boot-docker-0.0.1-SNAPSHOT.jar
```

- Build image

```
$ docker build . -t capgemini/ipservice-boot-docker

$ docker images
REPOSITORY                        TAG       IMAGE ID        CREATED          SIZE
capgemini/ipservice-boot-docker   latest    e0e8d458c945    4 seconds ago    625MB
openjdk                           latest    ab0ecda9094c    2 weeks ago      610MB
```

- Run the container

```
$ docker run --name ipservice -d -p 8090:8090 capgemini/ipservice-boot-docker

$ curl http://$(docker-machine ip ms):8090/ip
{"id":1,"ipAddress":"172.17.0.2","message":"Hello from IP Service from Docker"}
```

Maven Plugin

```
<plugin>
  <groupId>io.fabric8</groupId>
  <artifactId>docker-maven-plugin</artifactId>
  <version>0.21.0</version>
  <configuration>
    <images>
      <image>
        <alias>ipservice</alias>
        <name>capgemini/ipservice-boot-docker:latest</name>
        <build>
          <from>openjdk:latest</from>
          <assembly>
            <descriptorRef>artifact</descriptorRef>
          </assembly>
          <cmd>java -jar maven/${project.artifactId}-${project.version}.jar</cmd>
        </build>
      </image>
    </images>
  </configuration>
</plugin>
```

```
$ mvn docker:build
$ mvn docker:start
```

Docker Compose

```
version: "3"

services:
  ipservice:
    image: capgemini/ipservice-boot-docker:latest
    networks:
      - ipservice

  ipclient:
    image: capgemini/ipclient-boot-docker:latest
    ports:
      - "8090:8090"
    networks:
      - ipservice

networks:
  ipservice:
```

```
$ docker-compose up -d
```

```
$ docker stack deploy --compose-file=docker-compose.yml ipdemo
```

Writing a single service is nice…

...but no microservice is an island

Challenges of Distributed Systems

- Configuration management
- Service registration & discovery
- Routing & balancing
- Fault tolerance (Circuit Breakers!)
- Monitoring

- Distributed configuration
- Service Discovery
- Loadbalancing
- Bulkheading
- Circuit Breaker
- Fallback
- Versioning/Routing
- Based on AWS

Eureka Server

- @EnableEurekaServer
- Dependency to cloud-starter-eureka-server

```
@EnableEurekaServer
@EnableAutoConfiguration
public class EurekaApplication {
  public static void main(String[] args) {
    SpringApplication.run(EurekaApplication.class, args);
  }
}
```

Eureka Client

- Registers automatically with the Eureka server under a defined name
- Can access other Microservices
- Integrates Load Balancing with Ribbon using
  - DiscoveryClient, FeignClient
  - Eureka aware RestTemplate (sample later)
- @EnableDiscoveryClient or @EnableEurekaClient
- Dependency to spring-cloud-starter-eureka

```
eureka.client.serviceUrl.defaultZone=http://eureka:8761/eureka/
eureka.instance.leaseRenewalIntervalInSeconds=5
spring.application.name=ipservice
eureka.instance.metadataMap.instanceId=ipservice:${random.value}
eureka.instance.preferIpAddress=true
```

RestTemplate & Load Balancing

- @RibbonClient
- Dependency to spring-cloud-starter-ribbon

```
@RibbonClient("ipclient")
... // Left out other Spring Cloud / Boot Annotations
public class IPAddressController {

  @Autowired
  private RestTemplate restTemplate;

  @RequestMapping(value = "/ip", method = RequestMethod.GET)
  public IPAddress ipaddress() throws Exception {
    return template.getForEntity("http://ipservice/ip", IPAddress.class).getBody();
  }
}
```

Hystrix with Annotations

- Java proxies automaticaly created
- Annotations of javanica library
- @EnableCircuitBreaker or @EnableHystrix, dependency to spring-cloud-starter-hystrix

```java
@RequestMapping(value = "/ip", method = RequestMethod.GET)
@HystrixCommand(fallbackMethod = "localIP")
public IPAddress ipaddress() throws Exception {
    return template.getForEntity("http://ipservice/ip", IPAddress.class).getBody();
}

public IPAddress localIP() throws UnknownHostException {
    return new IPAddress(++counter, InetAddress.getLocalHost().getHostAddress(),
        config.getMessage());
}
```

What about non-java?

DevOps challenges for multiple containers

- How to scale?
- How to avoid port conflicts?
- How to manage them in multiple hosts?
- What happens if a host has a trouble?
- How to keep them running?
- How to update them?
- Where are my containers?

What if you could do all of this right now with an open-source platform?

- 100% open source, ASL 2.0
- Technology agnostic (java, nodejs, python, golang, etc)
- Built upon decades of industry practices
- 1-click automation
- Cloud native (on premise, public cloud, hybrid)
- Complex build/deploy pipelines (human workflows, approvals, chatops, etc)
- Comprehensive integration inside/outside the platform

Meet Kubernetes

Greek for Helmsman; also the root of the word Governor (from latin: gubernator)

- Container orchestration platform
- Supports multiple cloud and bare-metal environments
- Inspired by Google's experience with containers
  - Rewrite of Google's internal framework Borg
- Open source, written in Go
- Manage applications, not machines

Kubernetes

- Distributed configuration
- Service Discovery
- Load-balancing
- Versioning/Routing
- Deployments
- Scaling/Autoscaling
- Liveness/Health checking
- Self healing

Kubernetes Concepts

Pods

colocated group of containers that share an IP, namespace, storage volume, resources, lifecycle

Replica Set

manages the lifecycle of pods and ensures specified number are running (next gen Replication Controller)

Service

Single, stable name for a set of pods, also acts as LB

Label

used to organize and select group of objects

Kubernetes Concepts

Node

Machine (physical or VM) in the cluster

Master

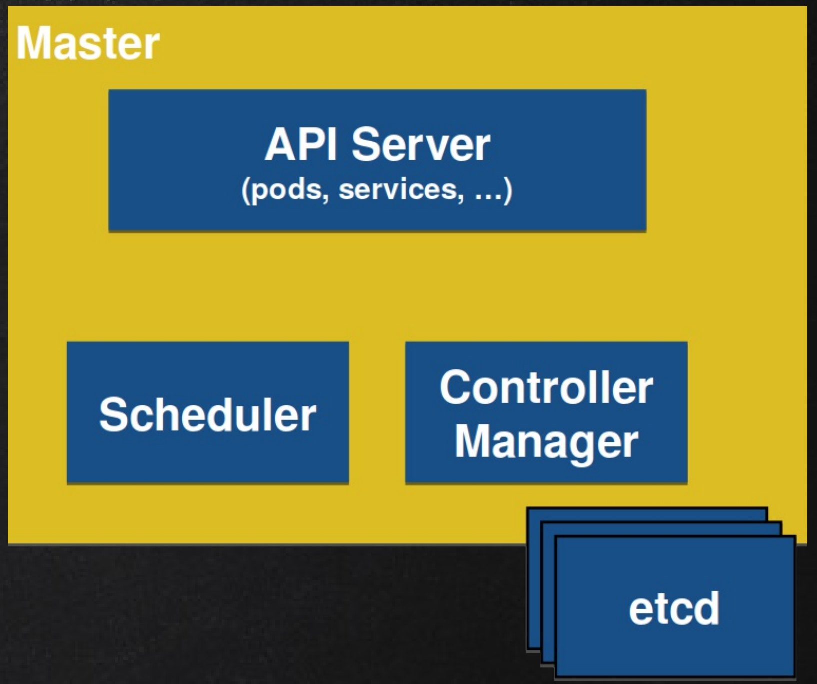
Central control plane, provides unified view of the cluster

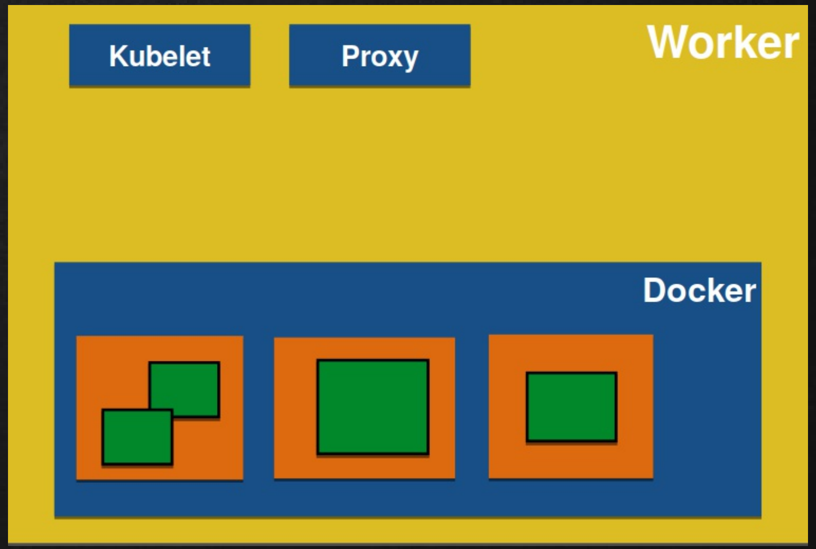
- etcd: distributed key-value store used to persist Kubernetes system state

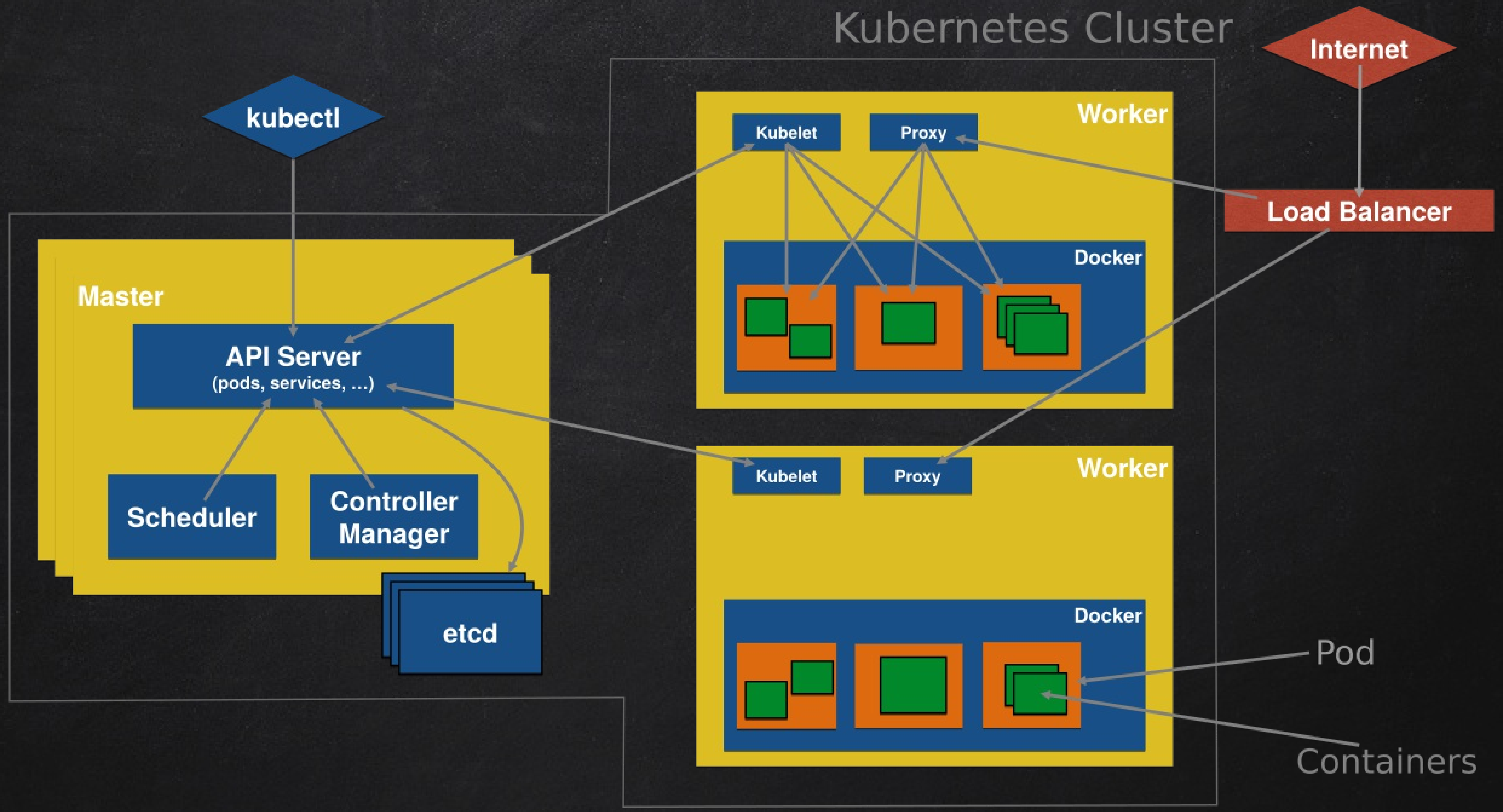
Worker

Docker host running kubelet (node agent) and proxy services

- Runs pods and containers






Kubernetes Cluster
Internet
kubectl
Worker
Kubelet
Proxy
Docker
Load Balancer
Master
API Server
(pods, services, ...)
Scheduler
Controller Manager
etcd
Worker
Kubelet
Proxy
Docker
Pod
Containers

OpenShift is Kubernetes

- Team self service application deployment
- Developer focused workflow
- Enterprise ready (LDAP, RBAC, Oauth, etc)
- Higher level abstraction above containers for delivering technology and business value
- Integrated Docker registry
- Jenkins Pipeline out of the box
- Build/deployment triggers
- Software Defined Networking (SDN)
- Docker native format/packaging
- CLI/IDE/Web based tooling

What about client-side load balancing?

Spring Cloud Kubernetes:

- DiscoveryClient
- Ribbon integration
- Actuator/Health integrations
- Hystrix/Turbine Dashboard integrations (kubeflix)
- Zipkin Tracking
- Configuration via ConfigMaps

Demo

Typical problems developing microservices

- How to run them all locally? ⇒ Minikube, Minishift, CDK
- How to package them ⇒ Docker
- How to test? ⇒ Arquillian
- Vagrant? VirtualBox? VMs? ⇒ Minikube, Minishift, CDK
- Specify configuration ⇒ Templates, EnvVars, ConfigMap
- Process isolation ⇒ Docker
- Service discovery ⇒ Kubernetes
- Multiple versions? ⇒ Kubernetes, API manager

Fabric8 all the things!

- Built on top of Kubernetes
- Wizards to create microservices
- Package as immutable containers
- Rolling upgrade across environments
- 1-Click install of fully configured CI/CD (Jenkins Pipeline, Nexus, Git)
- Feedback loops
- Lots of developer tooling
- ChatOps
- iPaaS/Integration
- Chaos Monkey

Thanks!

Any questions?

You can find me at
@ksobkowiak
krzysztof.sobkowiak@capgemini.com
http://krzysztof-sobkowiak.net

Credits

Special thanks to all the people who made and released their awesome resources for free:

- MicroServices for Java Developers by Christian Posta
- Microservices with Spring Cloud, Netflix OSS and Kubernetes by Christian Posta
- Microservices with Docker, Kubernetes, and Jenkins by Rafael Benevides and Christian Posta
- Kubernetes Introduction by Rafael Benevides and Edson Yanaga
- Integration in age of DevOps by Christian Posta
- Docker for Java Developers by Arun Gupta
- Kubernetes for Java Developers by Arun Gupta