JavaCro'15

Service Discovery in OSGi

Beyond the JVM using Docker and Consul

JavaCro'15

About me

CTO @ Dexels

Architect at Sendrato Wearables

OSGi tinkerer

NoSQL advocate

JavaCro'15

Service Discovery

- As old as distributed systems
- Find system components
- Prevent hard-coding network addresses
- Important for micro service architectures

Service Discovery

- Finding services by exploring systems
- Use a well-known central registry where every service registers

JavaCro'15

Service Repository

- Watch out for a single point of failure
- Distributed key-value stores
- Low volume
- Resilience and availability are most important

Service Discovery (SOA)

- Can change providers at will
- Access to many 3rd party services
- Services go shopping for other services
- Services can respond to ‘markets’

JavaCro'15

Docker containers

- Light-weight virtual machine
- Process with a filesystem and a network
- Universal, polyglot application deployment mechanism
- DockerHub image store

JavaCro'15

Mini demo

Docker from a service perspective

- An image
- An id
- Exposed ports
- ENV variables (& labels since Docker 1.6)

Docker

- Makes reasoning about services much easier

JavaCro'15

Docker

- Restful HTTP daemon on each host
- CLI tools use the HTTP interface to the daemon

Service discovery

So how do we ‘discover’ these services?

We ask the Docker daemon

What's missing?

HostIp: 192.168.59.103

HostPort: 49212

ContainerPort: 3306

Protocol: TCP

… we need more metadata

Metadata

- Which service is this (name, tags)
- What protocol? How do I consume the service?

We can add them as environment parameters

JavaCro'15

Monitor the Docker Daemon

- Completely generic
- No need for a registry
- Only require a few ‘annotations’

This was the easy part

JavaCro'15

Dynamic Services

- Any service can just appear
- Multiple instances can appear
- Instances can disappear without warning
- Service cascades

JavaCro'15

OSGi

- Java based dynamic service framework
- Since 1999
- Initially for embedded systems
- Now very popular in cloud deployments

OSGi

- Central ‘Service Bus’
- Any Java object can be registered as a service
- Code can redeploy on the fly
- Services can come and go dynamically
- Services can depend on other services

JavaCro'15

Docker & OSGi

- Monitor the Docker daemon
- Inject into OSGi

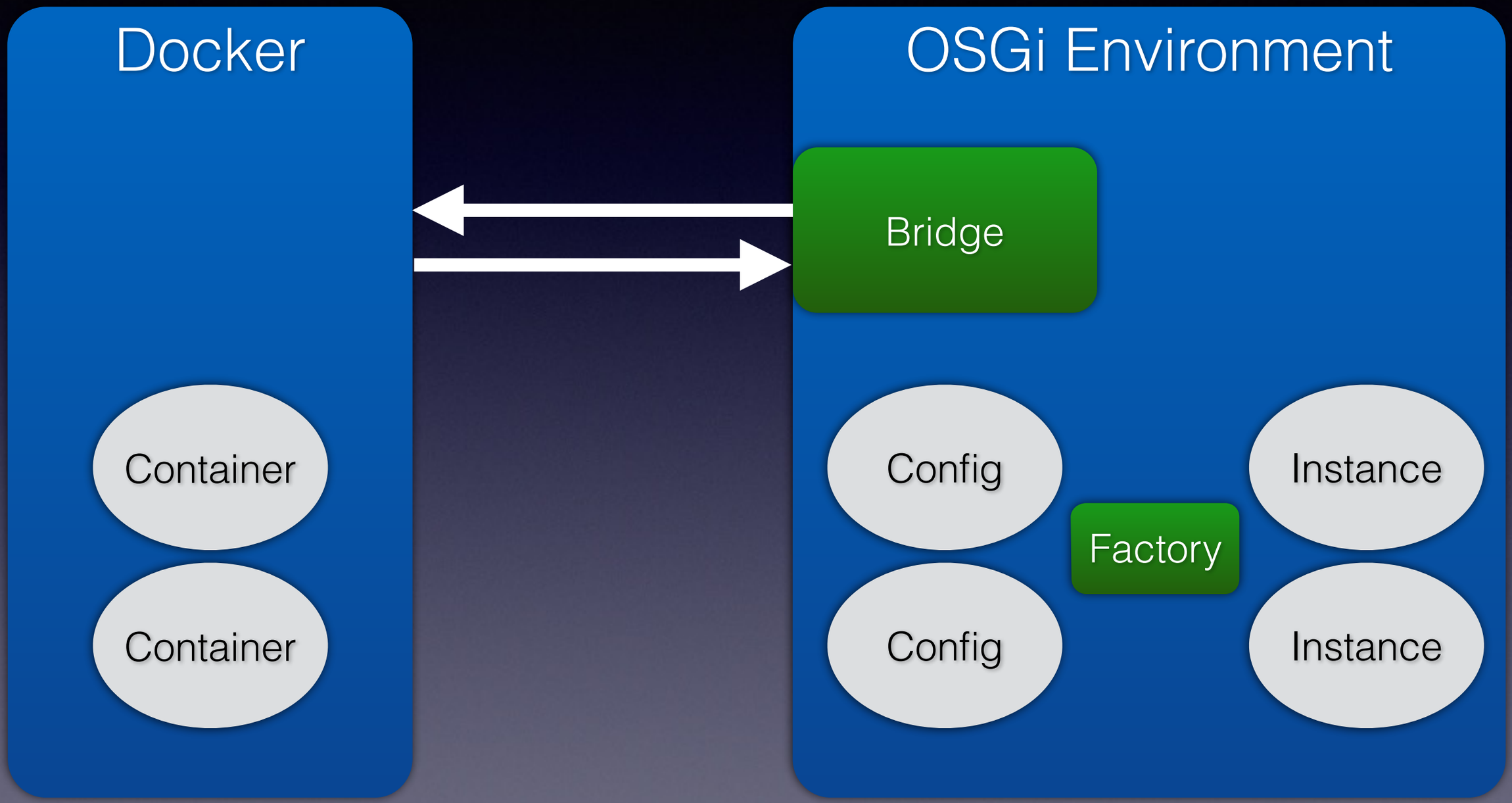
JavaCro'15

Docker & OSGi

- Mismatch between Java objects and Docker connection data
- The Docker ‘monitor’ won’t know what a service is really about

Configuration Admin

- Create Configuration objects based on docker data
- Leave the actual interpretation to ‘driver bundles’


Docker
Container
Container
OSGi Environment
Bridge
Config
Config
Factory
Instance
Instance

Another demo!

JavaCro'15

Limitations

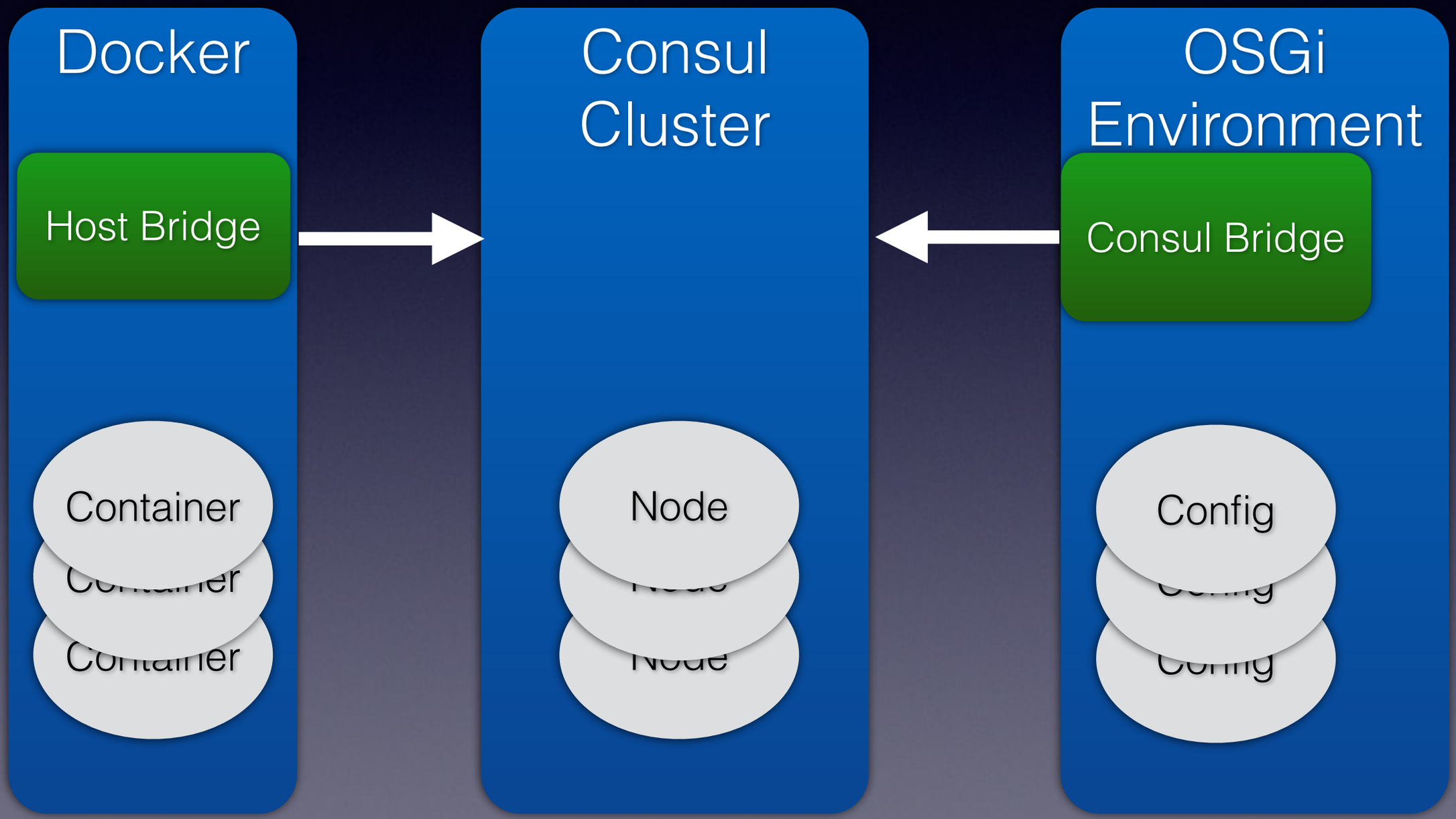
- Single host*
- Docker API Security
- Scalability

JavaCro'15

Consul

- Distributed Key/Value configuration store
- Like Etcd or Zookeeper
- Very robust for node failure
- But specifically built for Service Discovery
- Nice web UI

Consul


Docker
Host Bridge
Container
Container
Container
Consul
Cluster
Node
Node
Node
OSGi
Environment
Consul Bridge
Config
Config
Config

Final demo!

Future work

- Conventions on metadata would be nice
- Better security model for Docker
- Use other sources of configuration like Kubernetes

Thank you!

frank@dexels.com

Twitter @lyaruu

Tasman: https://github.com/flyaruu/tasman

Blog: http://www.codemonkey.nl/