JAVA EE TO MICROSERVICES

AUTOMAGICALLY

ALEXANDRE PORCELLI

Principal Sofware Engineer

JBoss by Red Hat

MICROSERVICES

DEFINITION

IN COMPUTING, MICROSERVICES IS A SOFTWARE ARCHITECTURE STYLE IN WHICH COMPLEX APPLICATIONS ARE COMPOSED OF SMALL, INDEPENDENT PROCESSES COMMUNICATING WITH EACH OTHER USING LANGUAGE-AGNOSTIC APIS.

Wikipedia

MICROSERVICES

ULTIMATE GOAL

- DECOUPLE
- COMPOSE
- SCALE

MICROSERVICES

COMMON IMPLEMENTATION

- REST
- CONTAINERS
- ORCHESTRATION
- PAAS

MICROSERVICES

REGULAR PROBLEM

- BUSINESS LOGIC **MIXED** WITH DISTRIBUTED COMPUTING
  - **COMPLEX ERROR HANDLING**
  - **NETWORK PARTITION**
  - **AVAILABILITY**

WHAT IF...

FLEXIBILITY OF
MICROSERVICES

SIMPLE
PROGRAMMING
MODEL

ABSTRACT THE
COMPLEXITY OF
DISTRIBUTED COMPUTING

KEEP THE VALUES
MICROSERVICES
ARCHITECTURE BRINGS


UF
CLOUD
EXTENSION

WARNING

this might not be for you...

but for 90% of the cases, it works!


JPA
CDI
JSF
JAX-WS
SERVLET
JMS
JTA
JCA
EJB
JSP
MAIL
JAAS
JAX-RS
REST
WEBSOCKET
BEAN VALIDATION
BATCH

CONTEXTS AND DEPENDENCY INJECTION

NATURAL ENABLER OF A GOOD SOFTWARE ARCHITECTURE

DEPENDENCY INJECTION ⊕ EVENTS

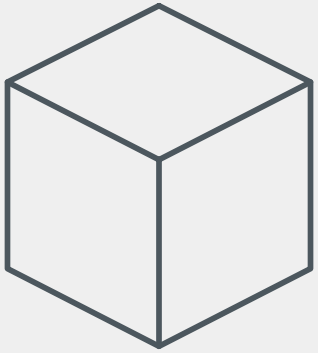
COMMON ATTRIBUTES OF GOOD CDI BASED APPS

- MODULAR
- LOOSELY COUPLED
- TESTABLE

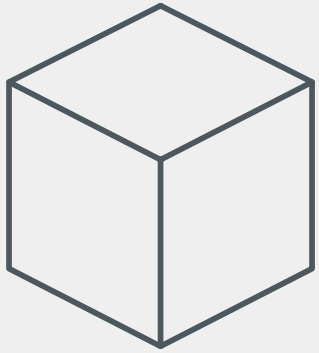
redhat

NOTES DEMO

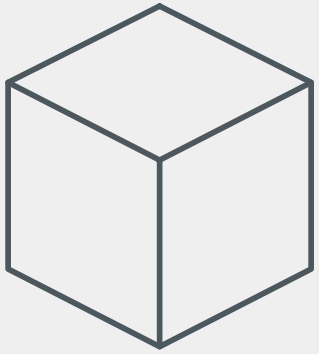
MODULES



PERSISTENCE

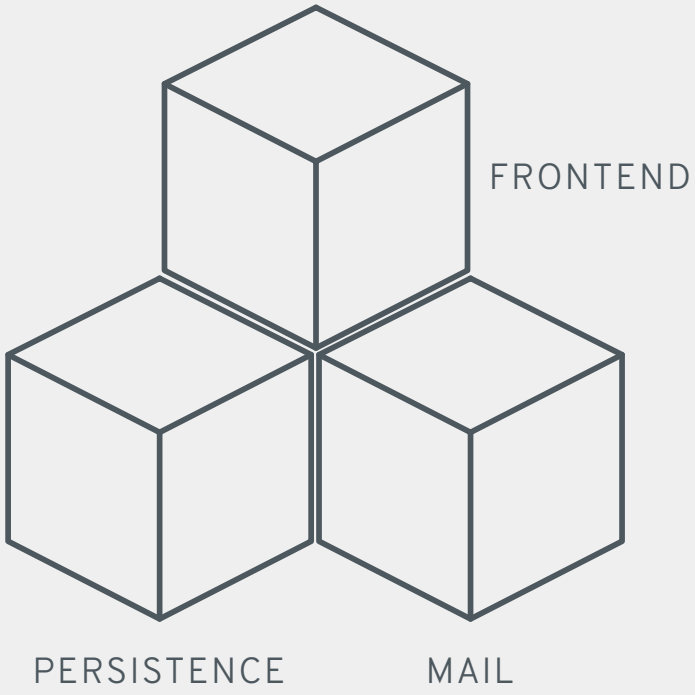


MAIL

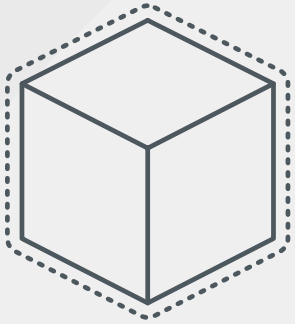


FRONTEND

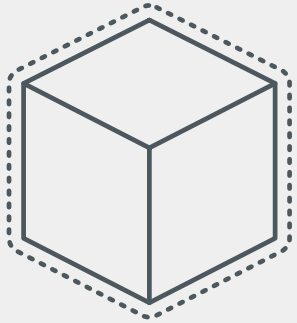
MONOLITH



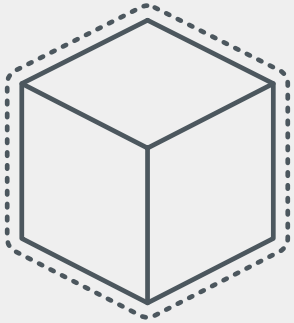
UF CLOUD ENABLEMENT



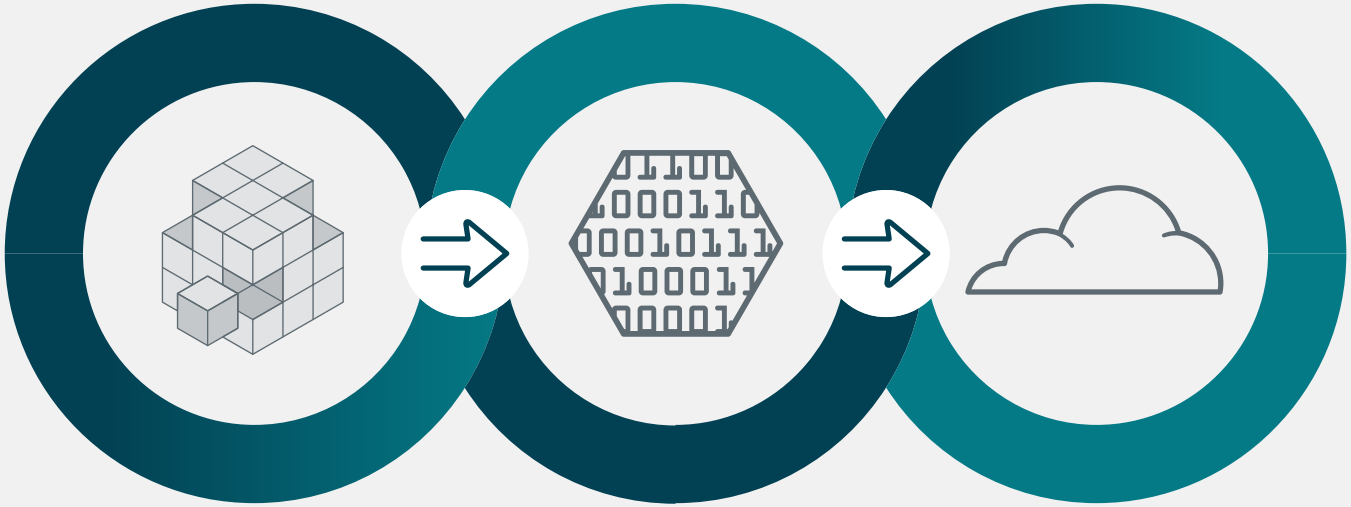
PERSISTENCE



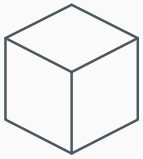
MAIL



FRONTEND


MONOLITH
CDI EXTENSION
CLOUD

ARCHITECTURE



API

@Cloud

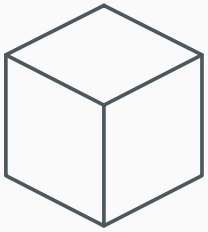
EVENT

OBSERVER

MARSHALLING

EVENT

PROPAGATION

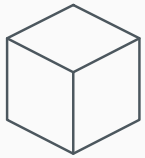


PROXY BEAN

PUBLISHER

CDI

Extension

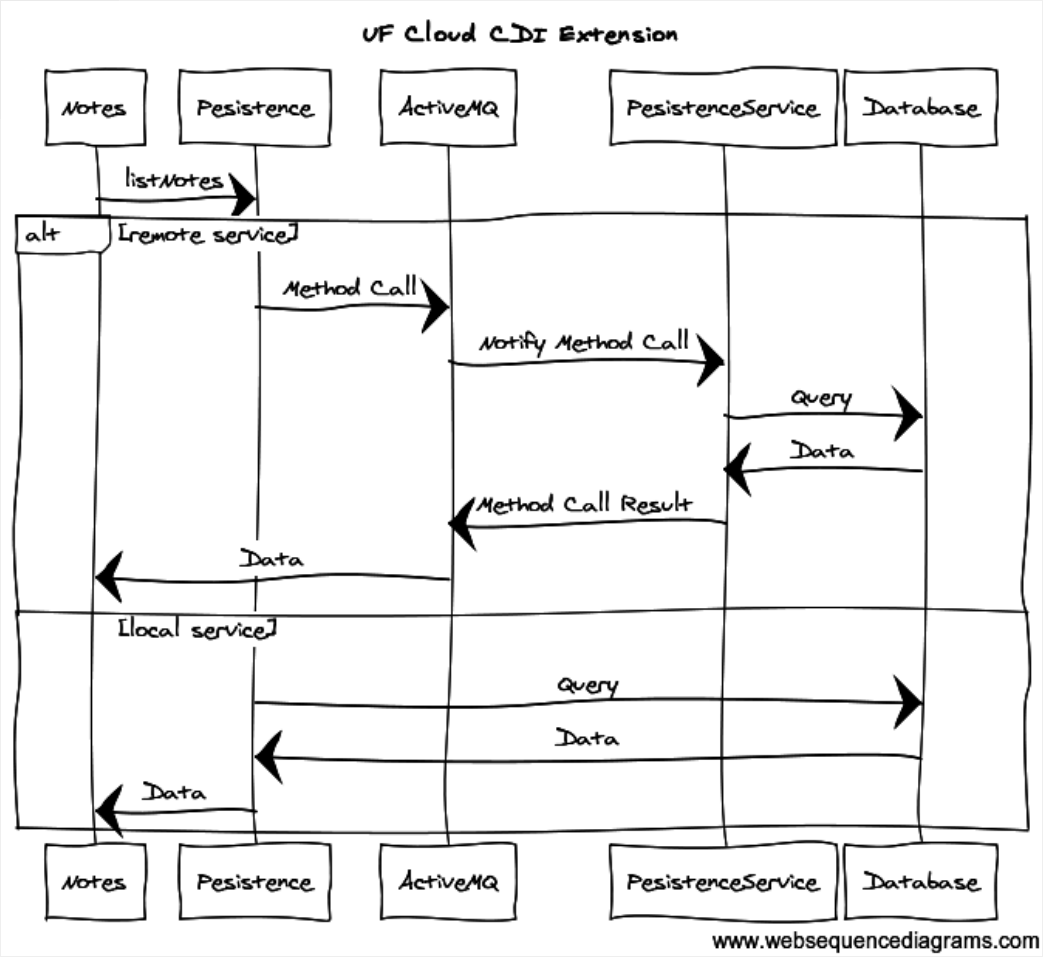


Publishers

ActiveMQ

Rest

ARCHITECTURE

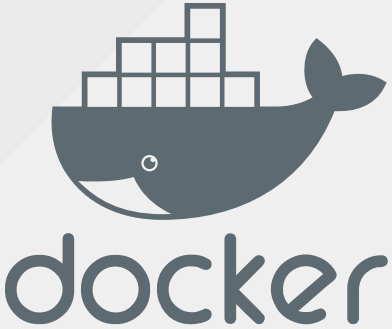


SHOW ME THE CODE!

```
package org.uberfire.cloud.cdi; importjava.lang.annotation.Annotation; importjava.lang.
importjavax.enterprise.inject.spi.AfterBeanDiscovery; importjavax.enterprise.inject.spi
importorg.uberfire.cloud.rpc.RPCService; importorg.uberfire.cloud.rpc.RPCServiceImpl; i
for(finalAnnotationqualifier:methodQualifiers){ eventQualifiers.put(qualifier.annotatio
finalBeanManagerbm){ this.cloudTypeUtil=newCloudTypeUtil(cloudTypes); abd.addBean(newBe
publicvoiddestroy(finalLocalUniqueIdinstance, finalCreationalContext<LocalUniqueId>ctx)
returncloudTypeUtil; } @Override publicvoiddestroy(finalCloudTypeUtilinstance, finalCre
returnpba.getBeanAttributes().isAlternative(); } @Override publicbooleanisNullable(){ r
publicSet<Type>getTypes(){ returnnewHashSet<Type>(){{ add(EventDispatcher.class); add(O
add(newAnnotationLiteral<Any>(){ }); }}; } @Override publicClass<?extendsAnnotation>get
returnCollections.emptySet(); } @Override publicStringgetName(){ return"RoutingServiceI
returnRPCServiceImpl.class; } @Override publicSet<InjectionPoint>getInjectionPoints(){
ctx.release(); } }); } } private<T>TresolveBean(finalClass<T>type, finalBeanManagerbm){
```

redhat

NOTES DEMO


docker


WildFly


OPENSHIFT
origin

Q&A

redhat.

THANK YOU!

porcelli@redhat.com

twitter.com/porcelli