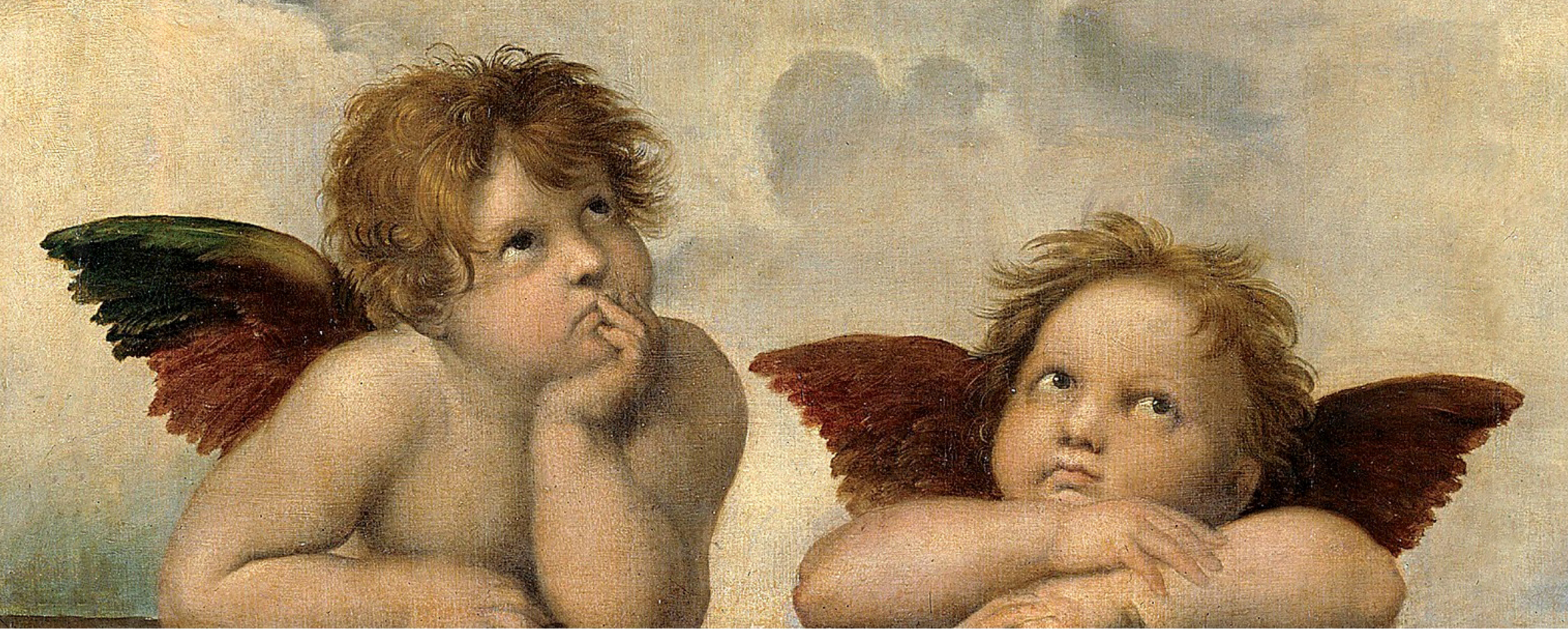
101 AND FUN

JEE ON DC/OS

Dr. Josef Adersberger (@adersberger)


DC/OS = #GIFEE
Google's
(and Facebook's, Twitter's, Airbnb's, ...)
Infrastructure
For
Everyone
Else

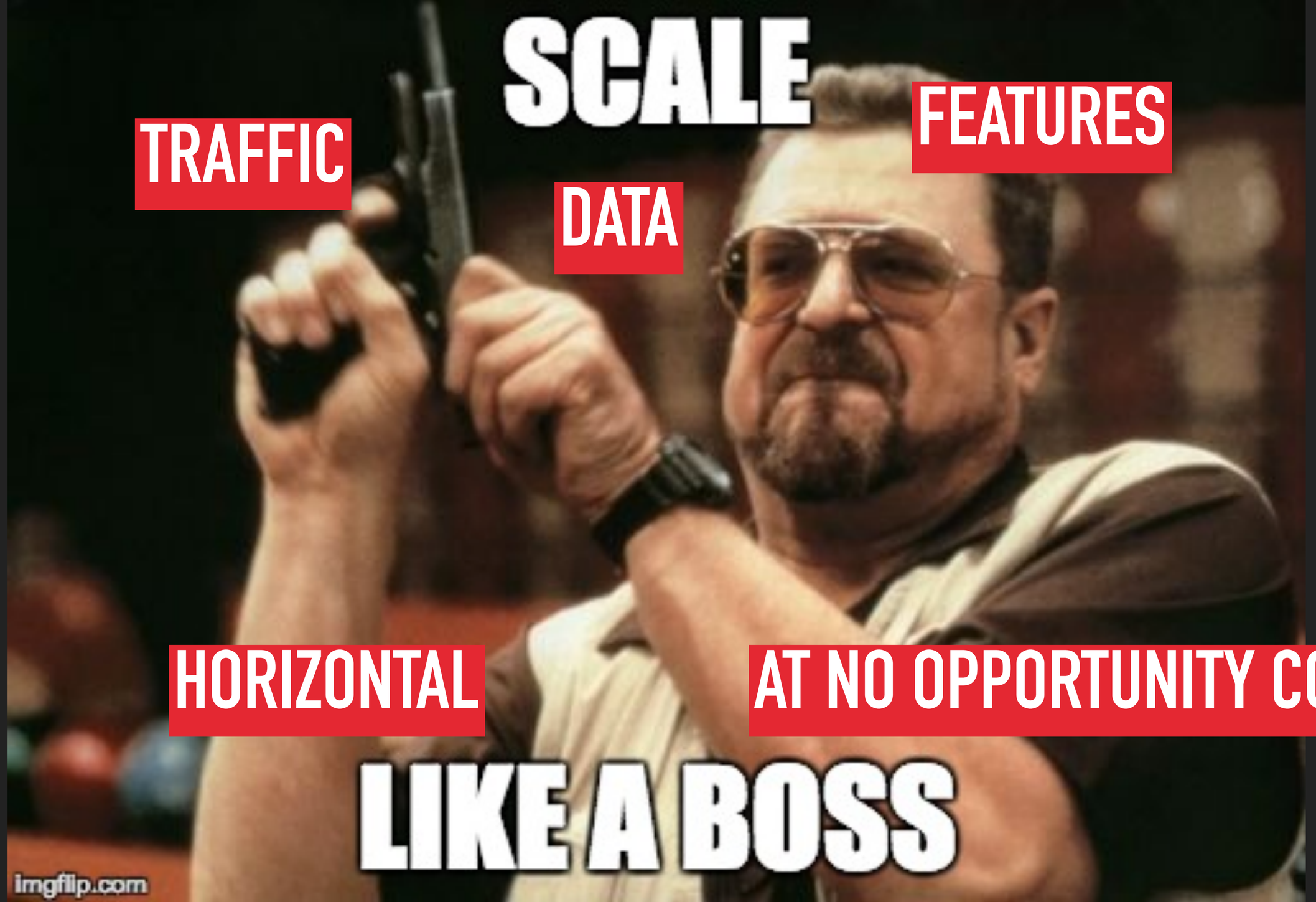


#GIFEE => Cloud Native Applications

(apps like Google's, Facebook's, ...)


Why?!


I HIRED A CONSULTANT TO HELP US EVOLVE OUR PRODUCTS TO CLOUD COMPUTING.
Dilbert.com DilbertCartoonist@gmail.com
BLAH BLAH CLOUD.
BLAH BLAH CLOUD.
BLAH BLAH CLOUD.
BLAH BLAH CLOUD.
©2011 Scott Adams, Inc./Dist. by UFS, Inc.
IT'S AS IF YOU'RE A TECHNOLOGIST AND A PHILOSOPHER ALL IN ONE!
BLAH BLAH PLATFORM.


SCALE
TRAFFIC
DATA
FEATURES
HORIZONTAL
AT NO OPPORTUNITY COSTS
LIKE A BOSS
imgflip.com


ELIZABE
CLOUD NATIVE JEE?


The second largest monolith on earth!*
*) only beaten by the guys who are writing PHP code at large scale

JEE

+

CLOUD NATIVE

instagram.com/toby_littledude

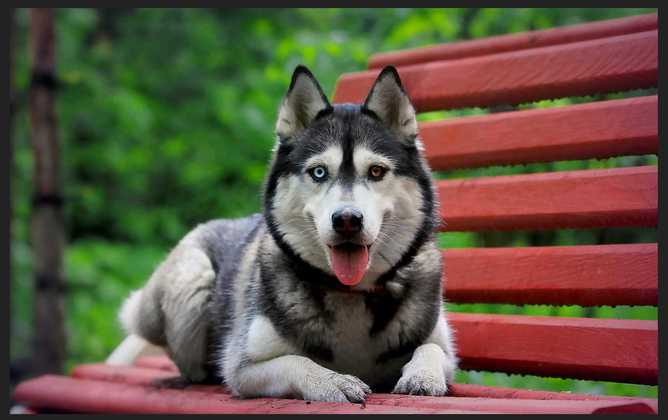
=

CLOUD NATIVE JEE

JEE IS NEITHER UGLY NOR FOCUSED ON MONOLITHIC APPS

- **Java EE by itself is modular und lightweight in the recent versions.** (to be honest: Spring feels more heavyweight than JEE to me from time to time)
- **Java EE micro containers** allow to modularize applications into self-contained runnable units.
- How to develop enterprise applications based on Java EE is **well understood and knowledge is widespread**.
- API is **mature and standardized**. Implementations are battle proven.

JEE



+

CLOUD NATIVE

instagram.com/toby_littledude

=

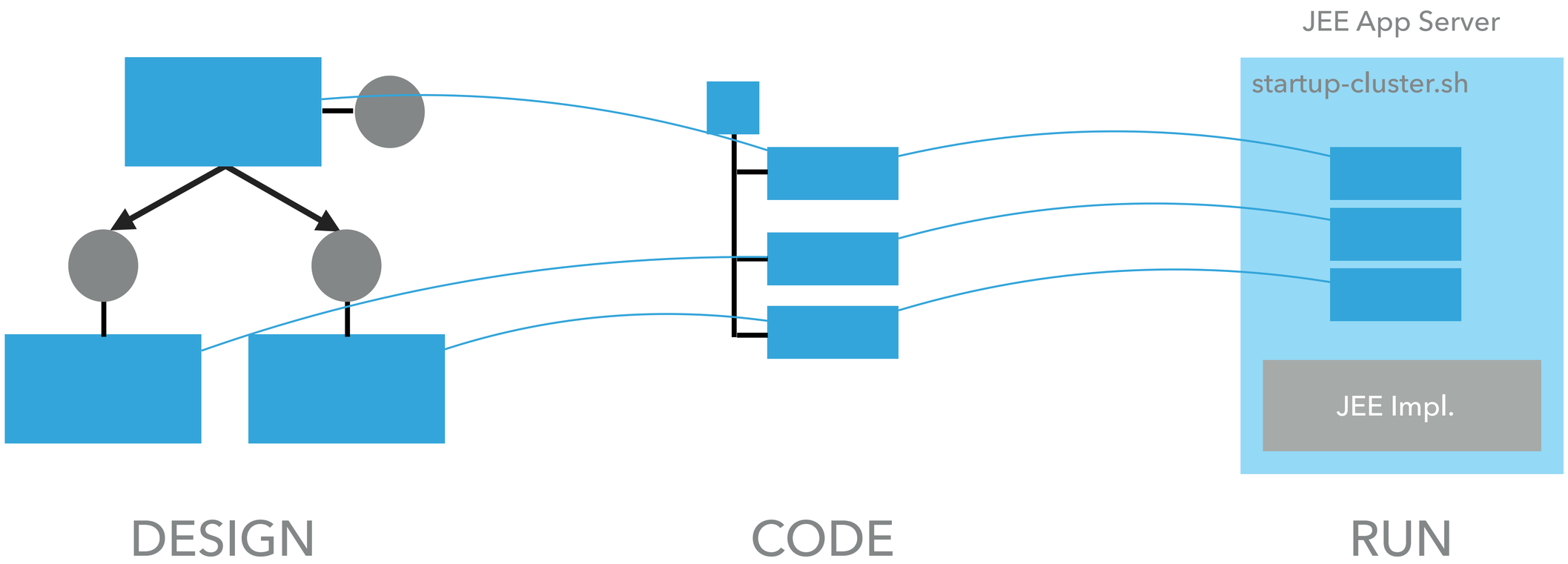
CLOUD NATIVE JEE


LET'S GET INTO CLOUD
NATIVE JEE!

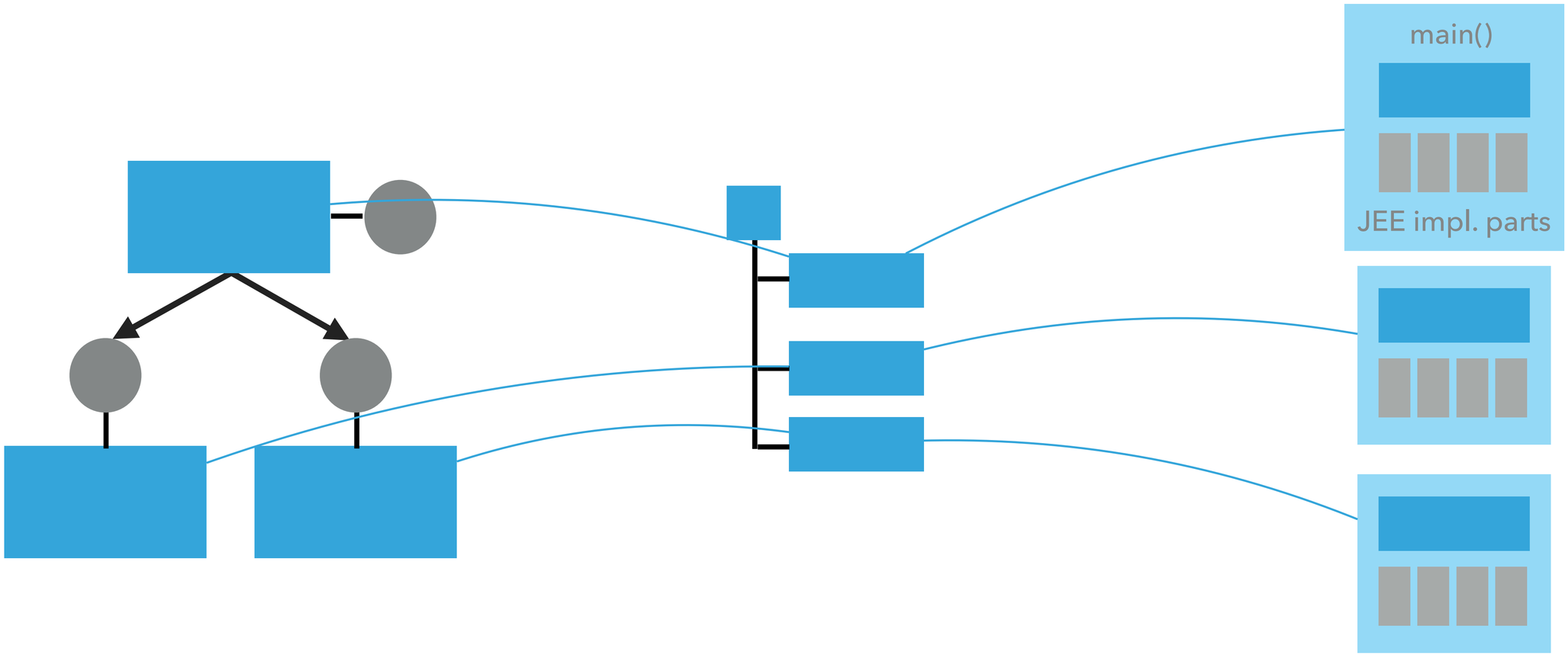
CLOUD NATIVE JEE

ARCHITECT'S VIEW

JEE IS NOT AN OVERALL MONOLITH. BUT PEOPLE TEND TO RUN JEE APPS AS MONOLITHS WITHIN HEAVY WEIGHT APP SERVERS.



JEE MICRO CONTAINER TO THE RESCUE: ENABLING MICROSERVICES ON JEE



DESIGN CODE RUN

A CLOUD OF JEE MICRO CONTAINERS. MISSION ACCOMPLISHED?

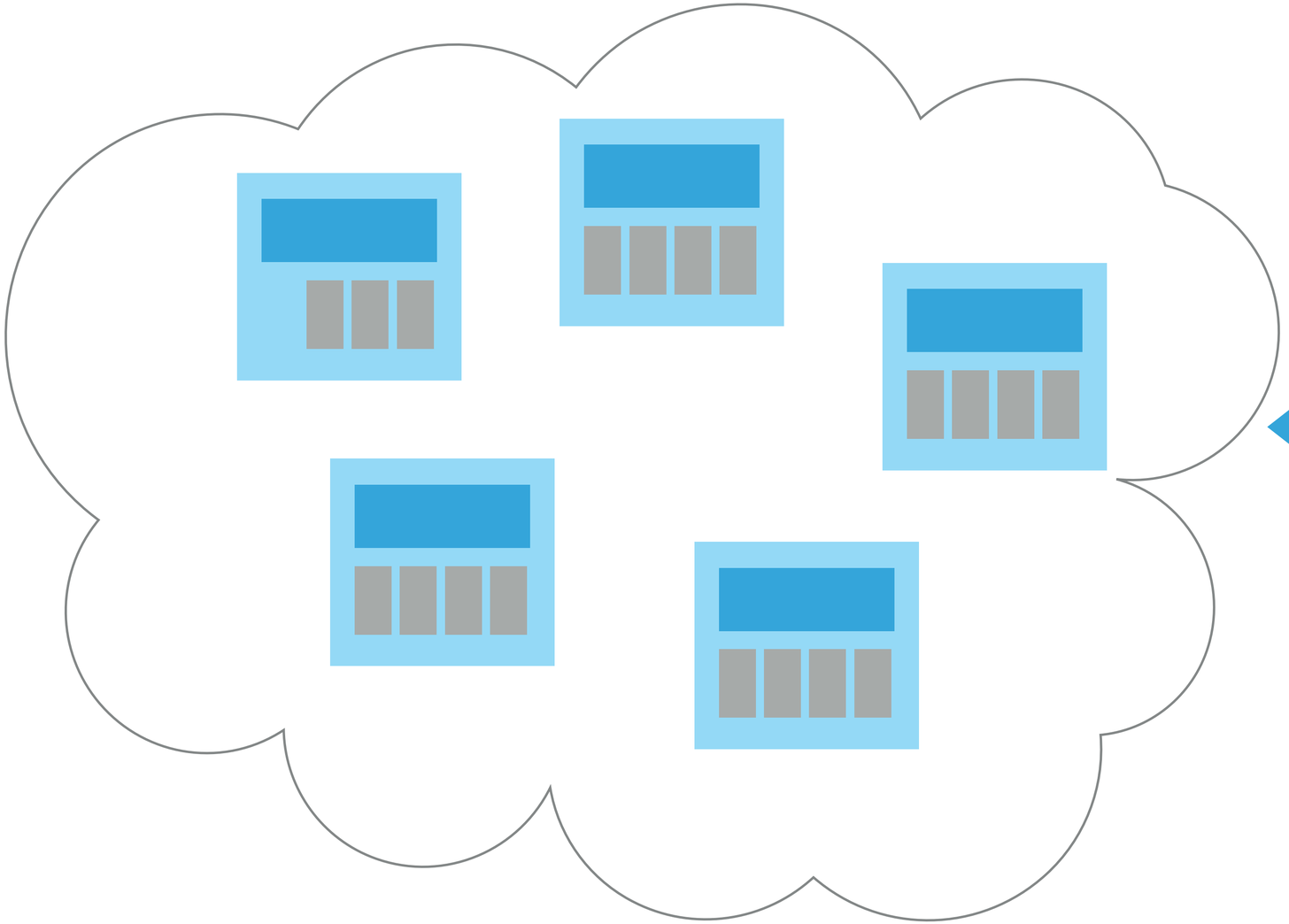
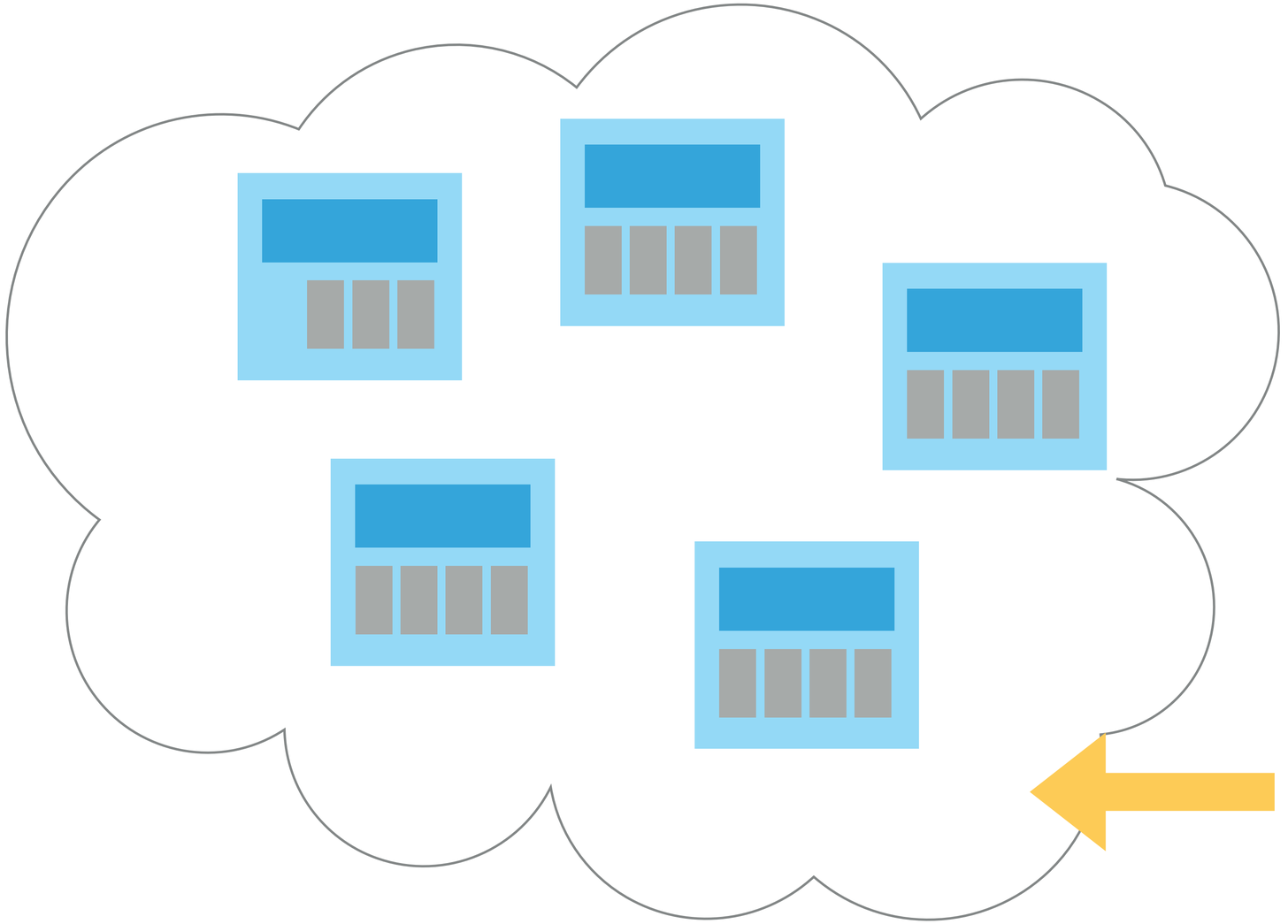
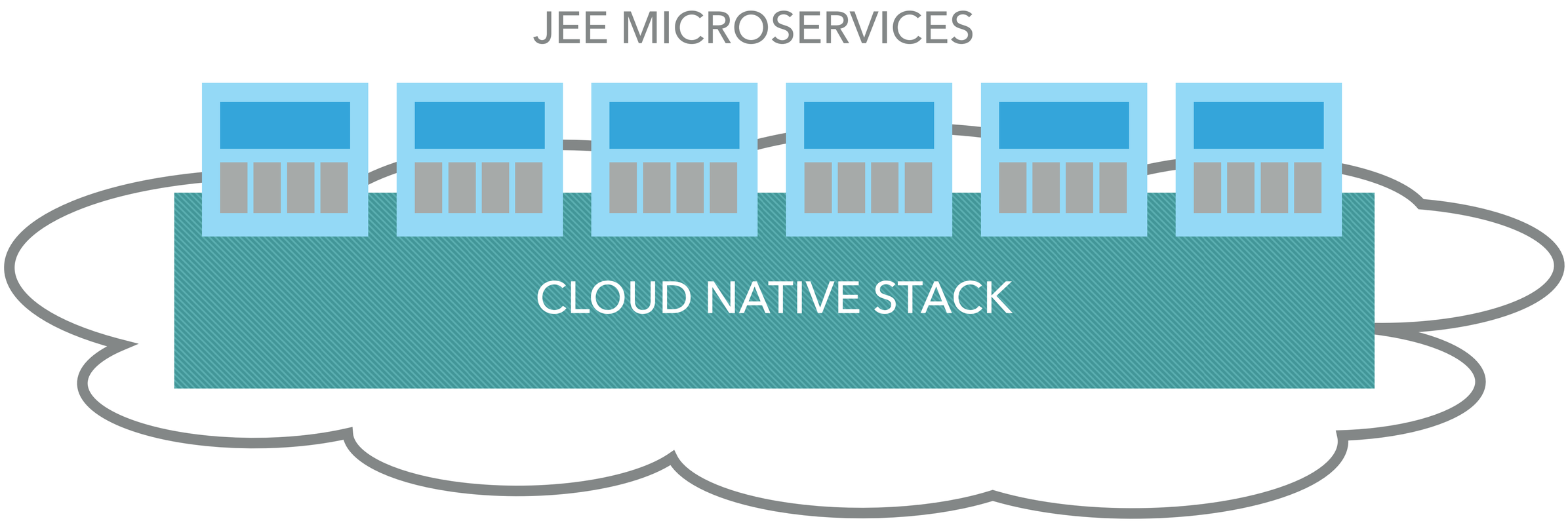


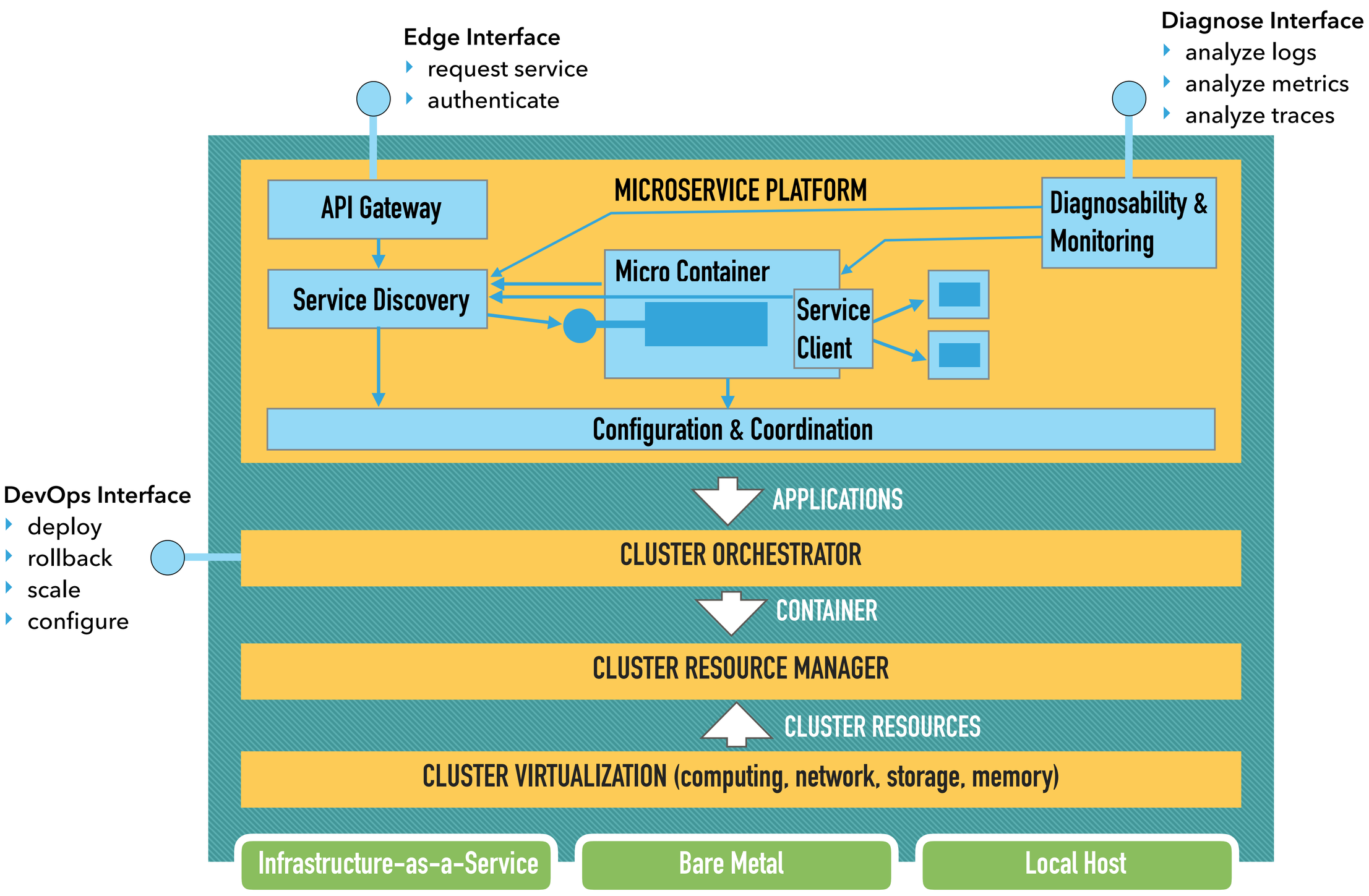
image: https://www.dreamstime.com

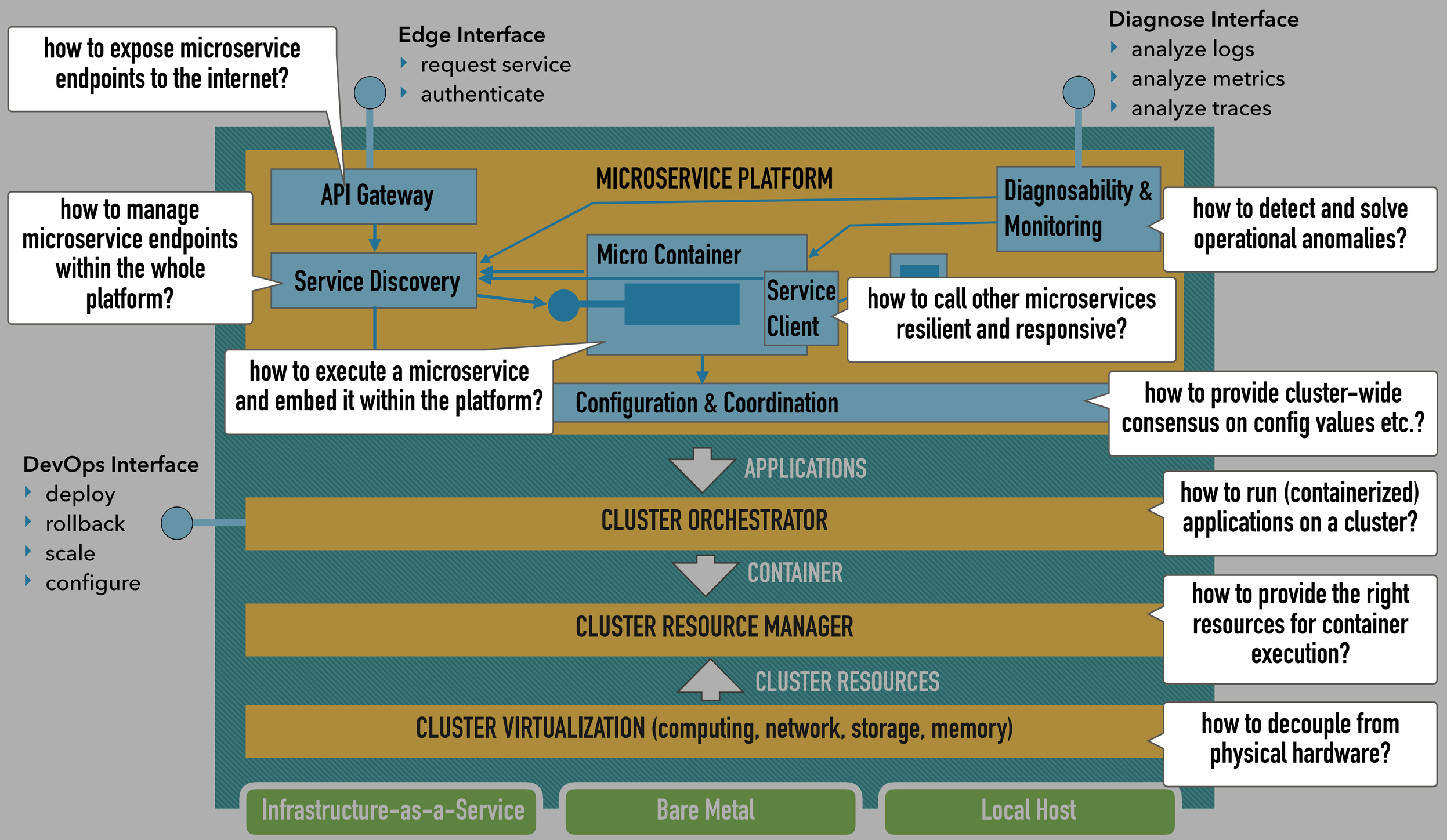


a cloud is not nothing!

what's inside our cloud?


JEE MICROSERVICES
CLOUD NATIVE STACK


Edge Interface
request service
authenticate
Diagnose Interface
analyze logs
analyze metrics
analyze traces
API Gateway
MICROSERVICE PLATFORM
Diagnosability & Monitoring
Service Discovery
Micro Container
Service Client
Configuration & Coordination
DevOps Interface
deploy
rollback
scale
configure
APPLICATIONS
CLUSTER ORCHESTRATOR
CONTAINER
CLUSTER RESOURCE MANAGER
CLUSTER RESOURCES
CLUSTER VIRTUALIZATION (computing, network, storage, memory)
Infrastructure-as-a-Service
Bare Metal
Local Host


how to expose microservice endpoints to the internet?
Edge Interface
request service
authenticate
Diagnose Interface
analyze logs
analyze metrics
analyze traces
MICROSERVICE PLATFORM
API Gateway
Diagnosability & Monitoring
how to detect and solve operational anomalies?
how to manage microservice endpoints within the whole platform?
Service Discovery
Micro Container
Service Client
how to call other microservices resilient and responsive?
how to execute a microservice and embed it within the platform?
Configuration & Coordination
how to provide cluster-wide consensus on config values etc.?
DevOps Interface
deploy
rollback
scale
configure
APPLICATIONS
CLUSTER ORCHESTRATOR
how to run (containerized) applications on a cluster?
CONTAINER
CLUSTER RESOURCE MANAGER
how to provide the right resources for container execution?
CLUSTER RESOURCES
CLUSTER VIRTUALIZATION (computing, network, storage, memory)
how to decouple from physical hardware?
Infrastructure-as-a-Service
Bare Metal
Local Host

Cloud Native Landscape 2016

goo.gl/xrVg3J

For more cloud native know-how see:
qaware.de/news/cloudnative

CODE

APPLICATION PLATFORM

Microservices

spring Spring Cloud, hapi, lagom

Dataservices*

Spark, spring Spring Cloud

APPLICATIONS

CLUSTER ORCHESTRATOR

kubernetes, Helios, docker COMPOSE, Kontena, vamp, apache brooklyn, Chronos, Lattice, RANCHER, Marathon, Apache AURORA, PFNFmax, Cloudify, decking

CONTAINER

CLUSTER SCHEDULER

MESOS, docker SWARM, Nomad, hadoop YARN, kubernetes, CoreOS fleet

CLUSTER RESOURCES

CLUSTER VIRTUALIZATION

OS Virtualization

docker, rkt, ZEROVM

Network Virtualization

flannel, OPENCONTRAIL, CALICO, Open Switch, weaveworks Weave Net, ROMANA, SocketPlane

Storage Virtualization

Flocker, GlusterFS, TACHYON, ceph, REX-Ray, MAPR

Memory Virtualization

hazelcast, Infinispan, Ignite

NODE RESOURCES

OPERATING SYSTEM

ATOMIC, RANCHEROS, PHOTON OS™, Snappy Ubuntu, CoreOS

PaaS

Flynn, DEIS, PaaSTA, CLOUDFOUNDRY, Apollo, OPENSHIFT, MANTL, Singularity

IaaS

Private / Hybrid

cloudstack, openstack, OpenNebula.org, HPE Helion public

Public

Microsoft Azure, Google Cloud Platform, amazon web services

CaaS

docker DOCKER DATACENTER, MESOSPHERE DCOS, TECTONIC by CoreOS

CLUSTER INFRASTRUCTURE

Configuration & Coordination:

etcd, Consul, Archaius, Zookeeper

Service Discovery:

Consul, Eureka, SkyDNS

Edge & API Server:

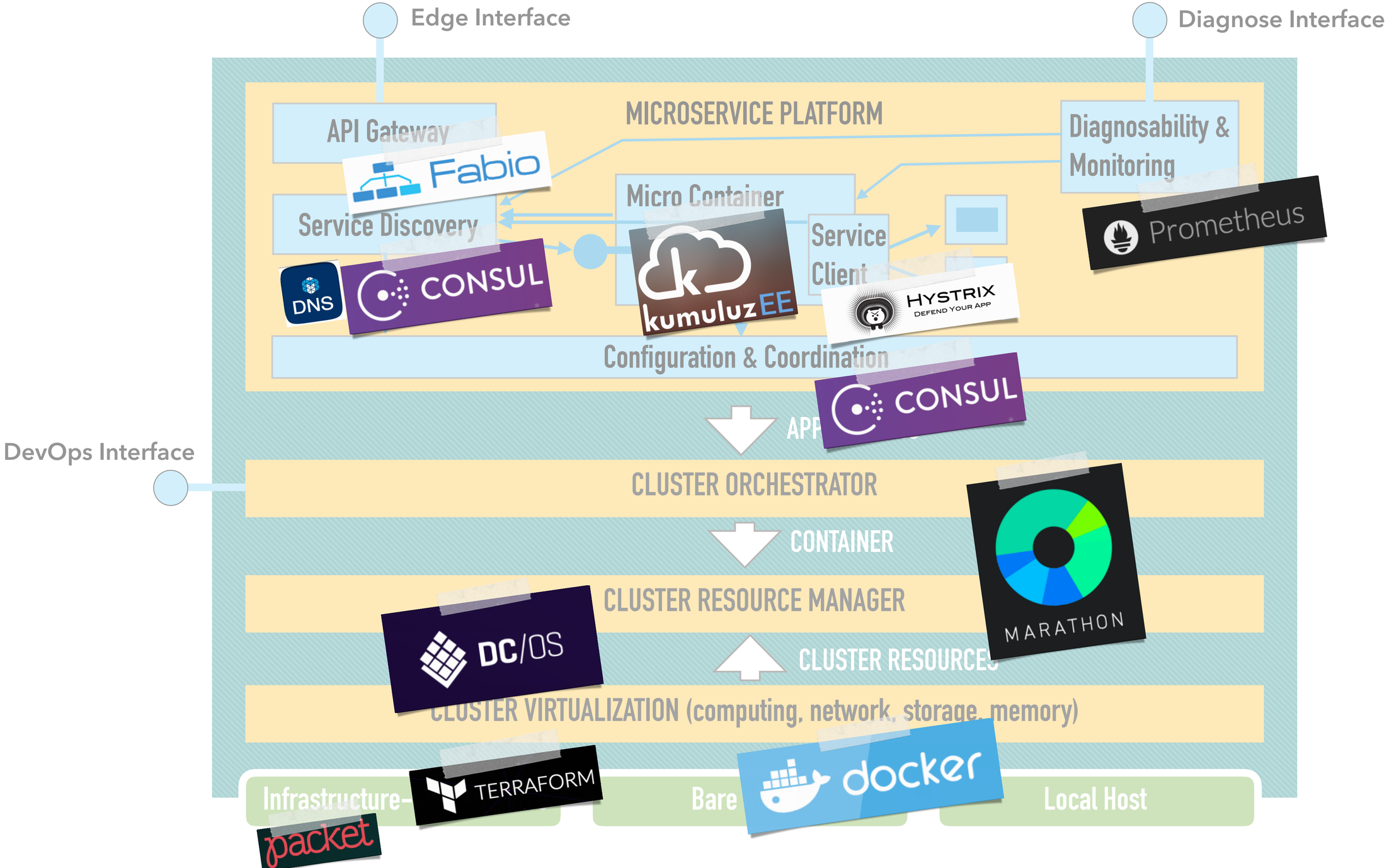
Tyk, KONG, træfik

Weave Flux, Vulkan, Zuul

Cluster Diagnosability:

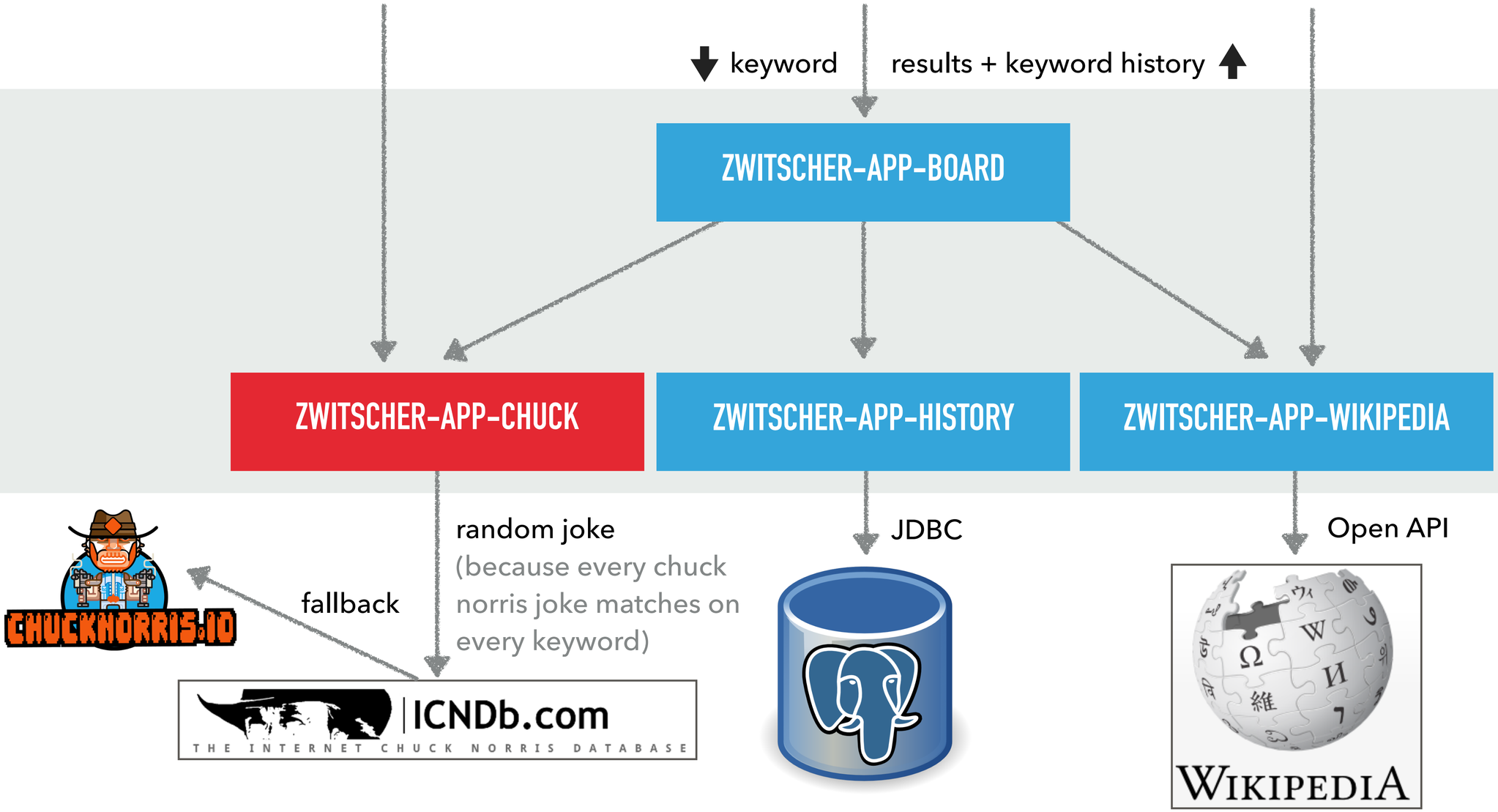
Grafana, sensu, ZIPKIN, sysdig, elastic + + kibana, Prometheus, NETFLIX OSS, Atlas, Turbine

* See our Big Data Landscape: qaware.de/news/big-data-landscape


Edge Interface
Diagnose Interface
MICROSERVICE PLATFORM
API Gateway
Fabio
Service Discovery
DNS
CONSUL
Micro Container
kumuluzEE
Service Client
Diagnosability & Monitoring
Prometheus
HYSTRIX
DEFEND YOUR APP
Configuration & Coordination
CONSUL
APP
DevOps Interface
CLUSTER ORCHESTRATOR
MARATHON
CONTAINER
CLUSTER RESOURCE MANAGER
DC/OS
CLUSTER RESOURCES
CLUSTER VIRTUALIZATION (computing, network, storage, memory)
Infrastructure-
TERRAFORM
packet
Bare
docker
Local Host

THE ZWITSCHER JEE SHOWCASE

https://github.com/adersberger/cloud-native-zwitscher-jee



CLOUD NATIVE JEE 101

DEVELOPER'S VIEW

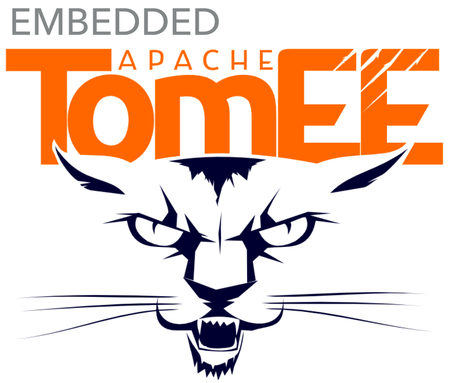
TALK
CHEAP
SHOW ME
THE
CODE

MICRO CONTAINER

DEPENDENCY INJECTION
APPLICATION LIFECYCLE
ENDPOINT EXPOSITION

THE JEE MICRO CONTAINER ECOSYSTEM

https://ee.kumuluz.com



http://tomee.apache.org

http://wildfly-swarm.io

implements (unstable)

http://microprofile.io (to come)

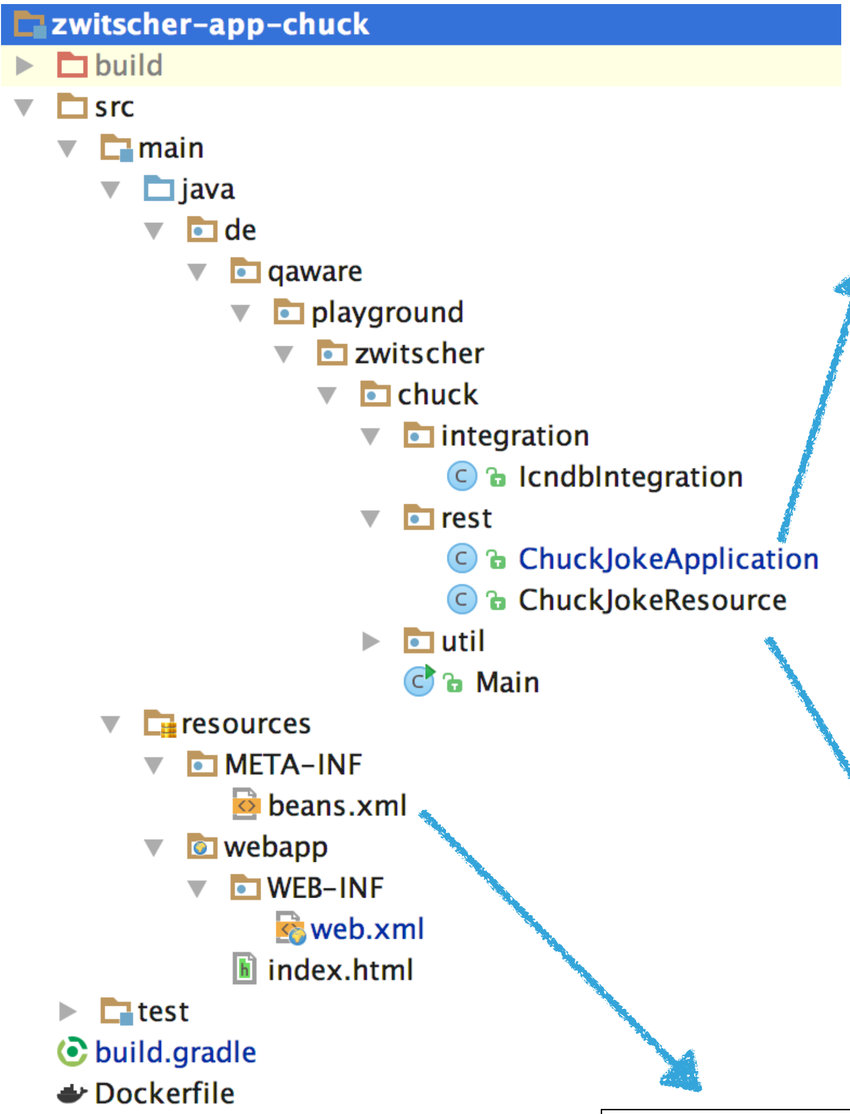
http://www.payara.fish/payara_micro

*1) http://tomee.apache.org/javaee7-status.html

*2) http://stackoverflow.com/questions/13487987/where-to-use-ejb-3-1-and-cdi

JEE MICRO CONTAINER COMPARISON CHART

	Payara Micro	Wildfly Swarm	TomEE+	kumuluzEE
Servlets et al.	x	x	x	x
WebSockets	x	x	x	
JSF	x	x	x	
JAX-RS	x	x	x	x
JAX-WS		x	x	
EJB *2	x	x	x	
CDI	x	x	x	x
JTA	x	x	x	
JCA		x	x	
JMS		x	x	
JPA	x	x	x	x
Bean Validation	x	x	x	x
JBatch	x	x		
Concurrency	x	x		
JCache	x	x		
JEE version	JEE 7	JEE 7	JEE 6, JEE 7 part.*1	JEE 7 part.
Packaging	WAR + JAR	JAR	JAR	classes + JARs
Startup time	4s	4s	2s	1s
Size	57MB	83 MB	44 MB	15 MB

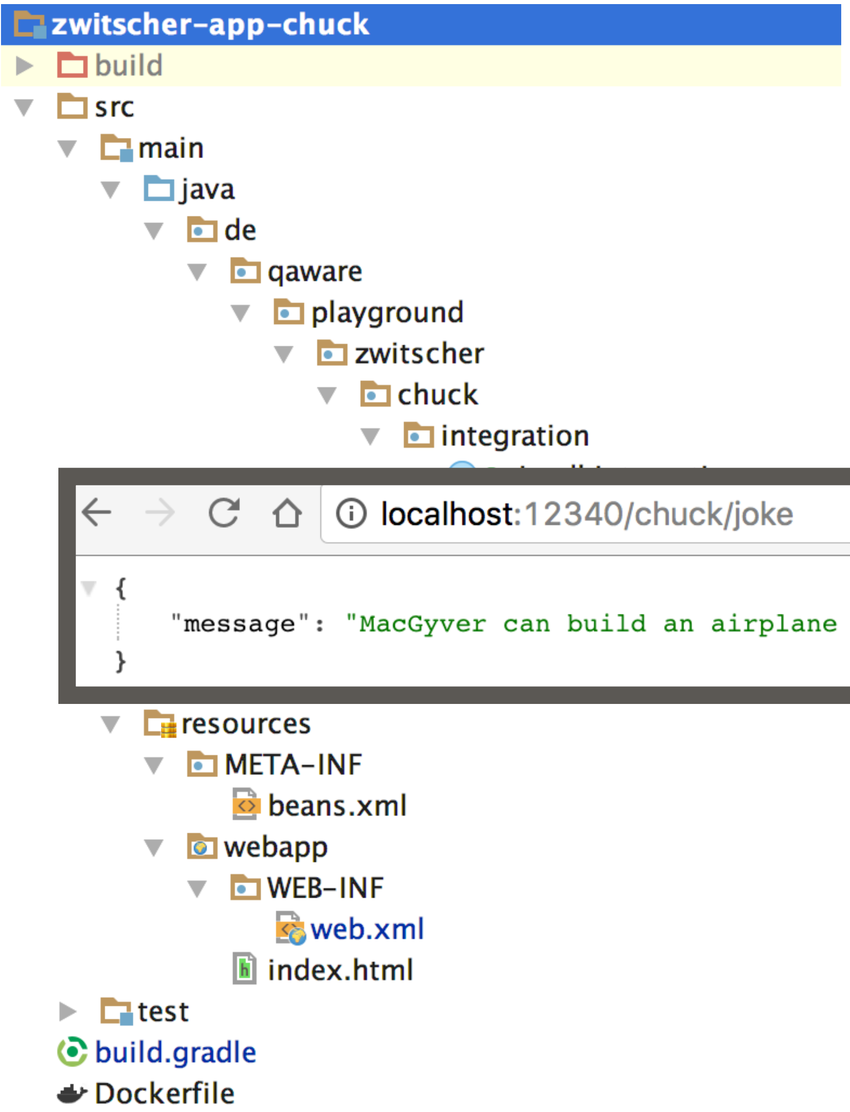



```java
@ApplicationPath("/chuck")
public class ChuckJokeApplication extends ResourceConfig {
    public ChuckJokeApplication() {
        super();
        register(ChuckJokeResource.class);
    }
}
```

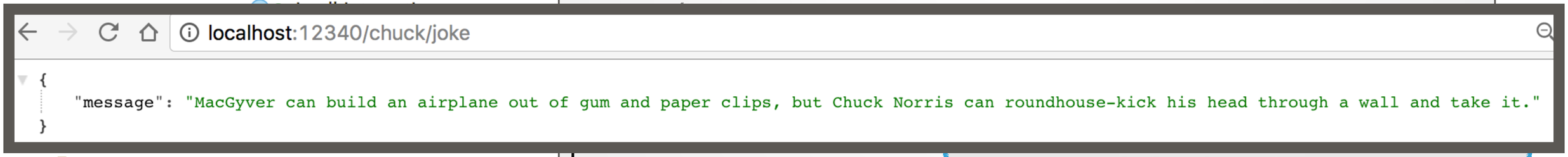
```java
@Path("/joke") @RequestScoped
public class ChuckJokeResource {
    @Inject
    private IcndbIntegration icndb;
    @GET
    @Produces("application/json")
    public Response getJoke() {
        Map<String, String> json = new HashMap<>();
        json.put("message", icndb.getRandomJoke());
        return Response.ok(json).build();
    }
}
```

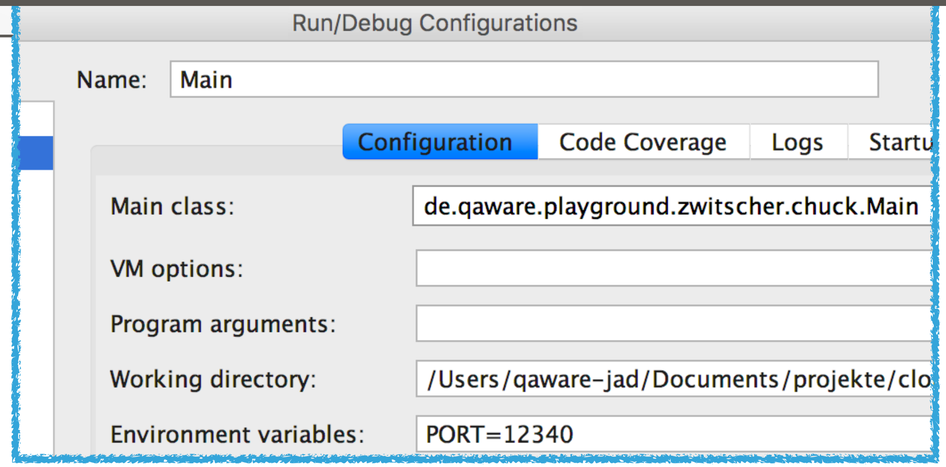
```
bean-discovery-mode="all"
```

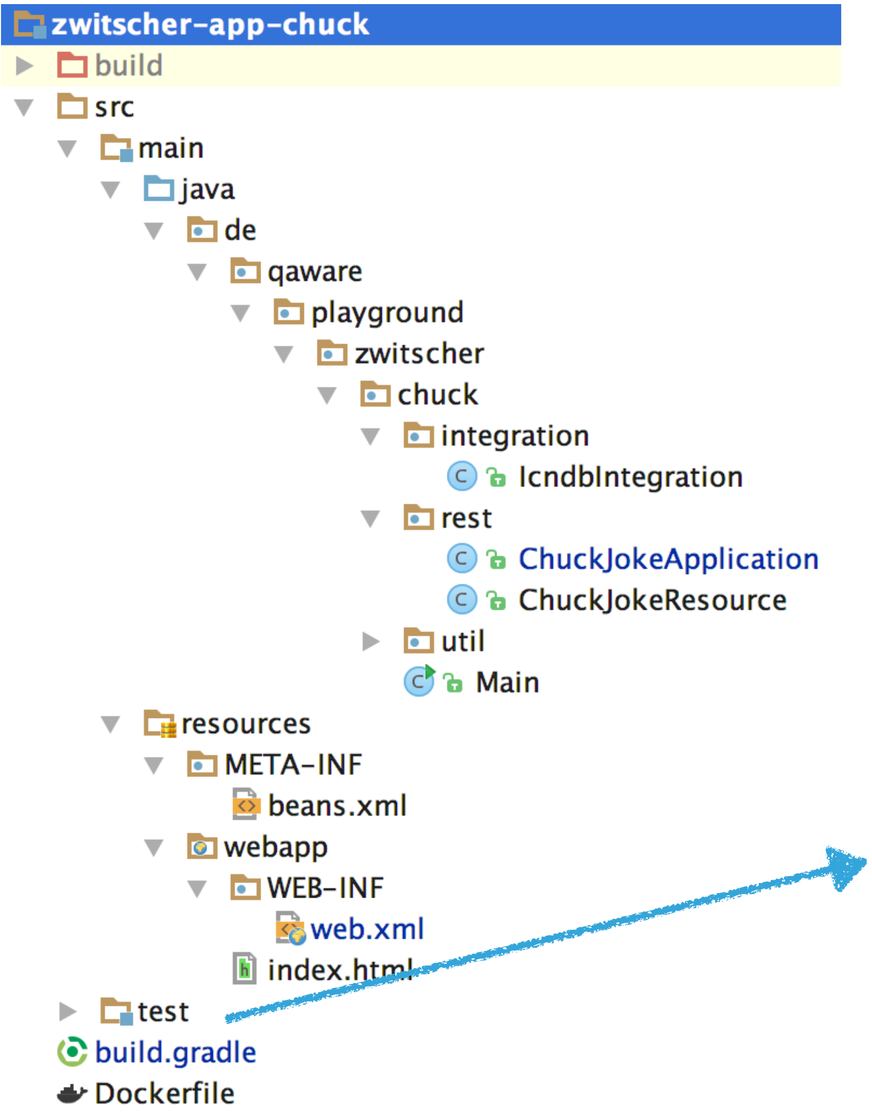
JEE MICRO CONTAINER STARTUP




```java
public class Main {

    /**
     * Starts the microservice container.
     *
     * Requires environment variable "PORT" according
     * on what port to listen.
     *
     * @param args no arguments evaluated
```








```java
@RunWith(CdiTestRunner.class)
public class TestChuckJokeResource {

    @Inject
    private ChuckJokeResource chuckJokeResource;

    @Test
    public void testChuckJokeResource() {
        Response response = chuckJokeResource.getJoke();
        assertThat(
            response.getStatusInfo().getStatusCode(),
            equalTo(200));
    }
}
```

SERVICE CLIENT

- SERVICE DISCOVERY
- LOAD BALANCING
- REQUEST MONITORING
- CIRCUIT BREAKING

SERVICE CLIENT WITH JERSEY, RXJAVA AND HYSTRIX

- Circuit Breaker (Resiliency)
- Request Monitoring

- Parallel & async execution (Responsive)

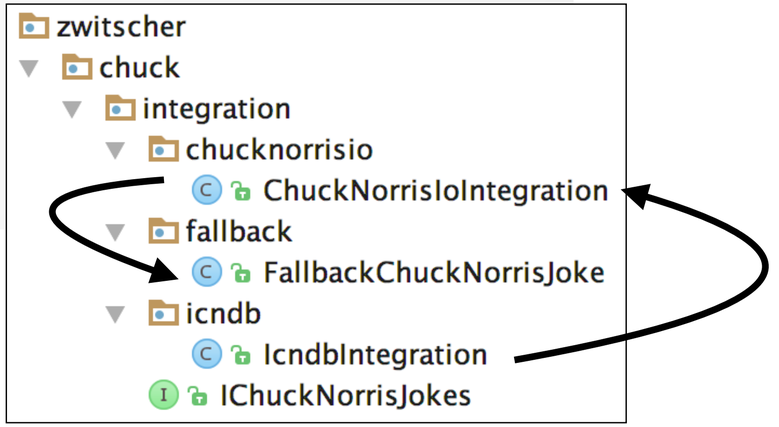
- JAX-RS 2.0 compliant REST clients

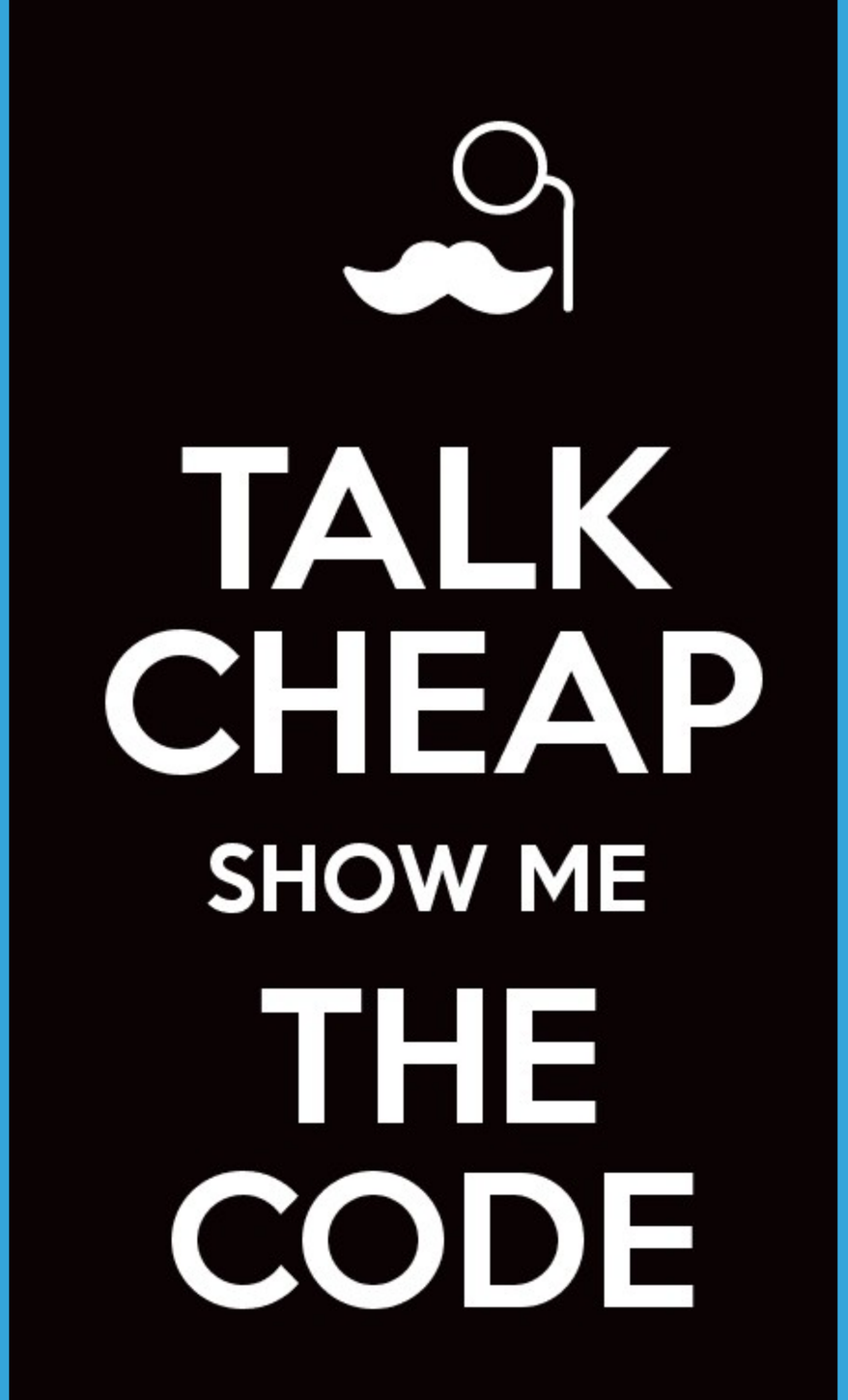
SERVICE CLIENT WITH JERSEY, RXJAVA AND HYSTRIX

```
@RequestScoped
public class IcndbIntegration implements IChuckNorrisJokes {

    @Override public String getRandomJoke() {
        IcndbIntegrationCommand cmd = new IcndbIntegrationCommand();
        return cmd.observe().toBlocking().toFuture().get();
    }

    private class IcndbIntegrationCommand extends HystrixObservableCommand<String> {

        IcndbIntegrationCommand() {
            super(Setter.withGroupKey(HystrixCommandGroupKey.Factory.asKey("zwitscher"))
                    .andCommandPropertiesDefaults(HystrixCommandProperties.Setter()
                            .withExecutionTimeoutInMilliseconds(3000)));
        }

        @Override protected Observable<String> construct() {
            return RxObservable.newClient()
                    .target("http://api.icndb.com").path("jokes/random").request(MediaType.APPLICATION_JSON_TYPE)
                    .rx().get()
                    .map(response -> {
                        Map<String, Map<String, String>> json = response.readEntity(Map.class);
                        return json.get("value").get("joke");
                    });
        }
    }
}
```

FALLBACK JOKES

```
@RequestScoped
public class IcndbIntegration implements IChuckNorrisJokes {

    @Inject @Named("chucknorrisjoke-chucknorrisio")
    private IChuckNorrisJokes fallback;

    @Override public Observable<String> getRandomJokeObservable() {
        return new IcndbIntegrationCommand().observe();
    }

    //…

    private class IcndbIntegrationCommand extends HystrixObservableCommand<String> {

        //…

        @Override
        protected Observable<String> resumeWithFallback() {
            return fallback.getRandomJokeObservable();
        }
    }
}
```

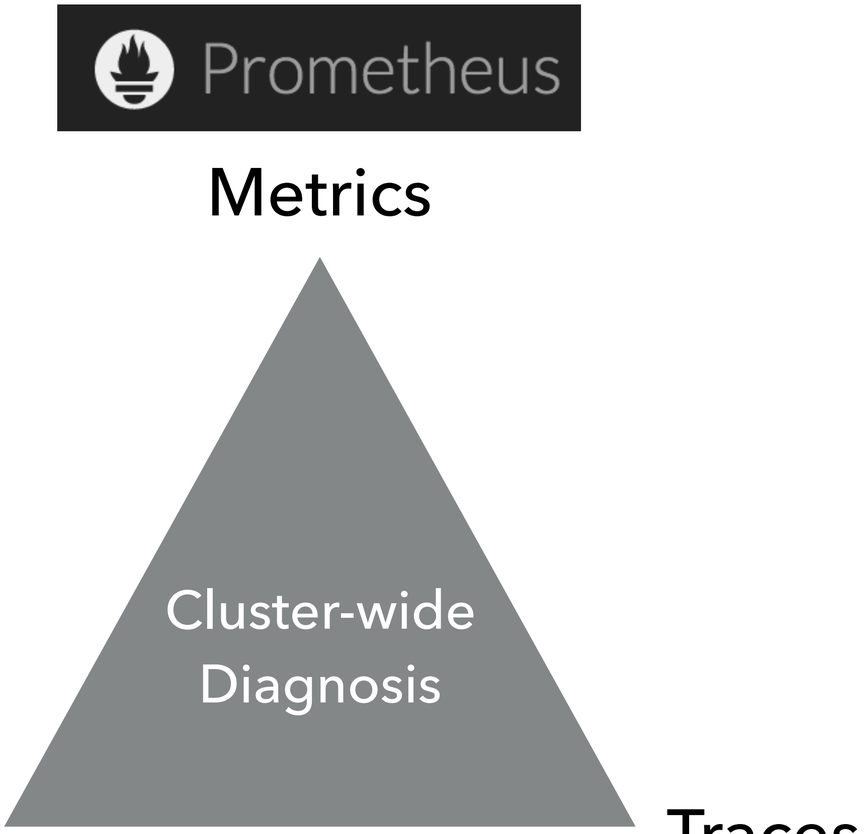




MONITORING & DIAGNOSABILITY

- COLLECT, STORE, ANALYZE METRICS
- COLLECT, STORE, ANALYZE LOGS
- COLLECT, STORE, ANALYZE TRACES
- DASHBOARDING & ALERTING

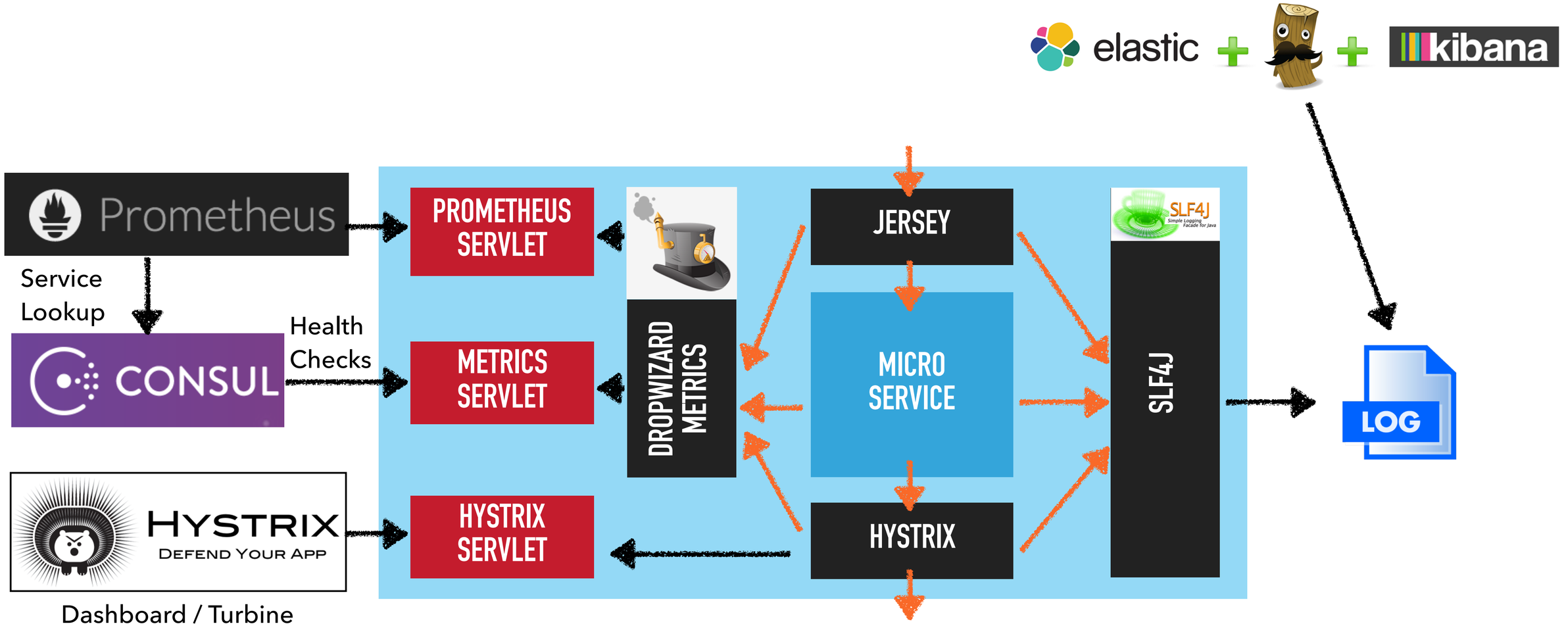
THE MAGIC DIAGNOSABILITY TRIANGLE



elastic + + kibana

ZIPKIN

DIAGNOSABILITY BIG PICTURE



INSTRUMENTING THE MICROSERVICE WITH DROPWIZARD METRICS

Obtain the singleton Metric Registry

Instrument inbound REST calls

```
@ApplicationPath("/chuck")
public class ChuckJokeApplication extends ResourceConfig {
    public ChuckJokeApplication() {
        super();
        //Instrument application with metrics
        MetricRegistry METRIC_REGISTRY = MetricsProvider.getMetricRegistryInstance();
        register(new InstrumentedResourceMethodApplicationListener(METRIC_REGISTRY));
        HystrixPlugins.getInstance().registerMetricsPublisher(
                new HystrixCodaHaleMetricsPublisher(METRIC_REGISTRY));
        //Register Prometheus metric exporter
        CollectorRegistry.defaultRegistry.register(new DropwizardExports(METRIC_REGISTRY));
    }
}
```

Instrument outbound REST calls

Export all metrics to Prometheus

Prometheus Alerts Graph Status Help

de_qaware_playground_zwitscher_chuck_rest_ChuckJokeResource_getJoke

Load time: 32ms
Resolution: 1s

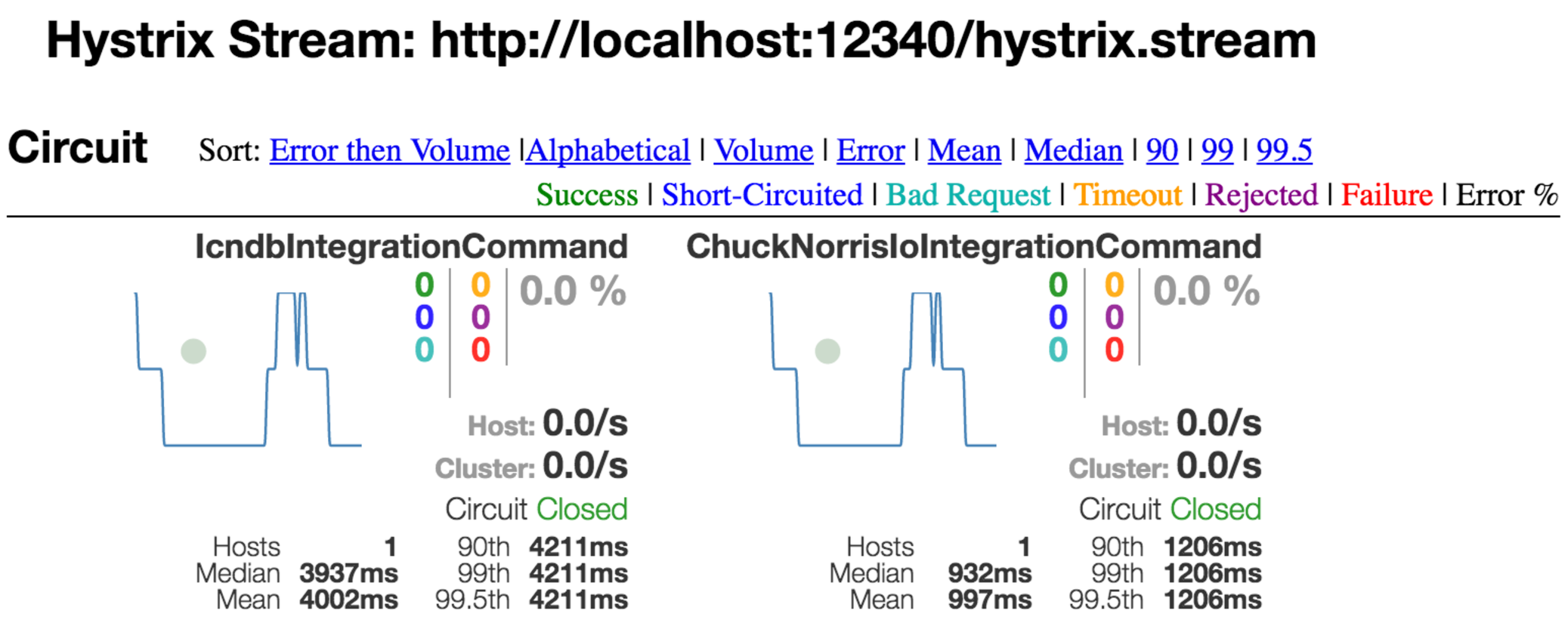
Execute - insert metric at cursor -

Graph Console

- 5m + Until Res. (s) stacked

de_qaware_playground_zwitscher_chuck_rest_ChuckJokeResource_getJoke{instance="localhost:12340",job="prometheus",quantile="0.999"}
de_qaware_playground_zwitscher_chuck_rest_ChuckJokeResource_getJoke{instance="localhost:12340",job="prometheus",quantile="0.99"}
de_qaware_playground_zwitscher_chuck_rest_ChuckJokeResource_getJoke{instance="localhost:12340",job="prometheus",quantile="0.98"}
de_qaware_playground_zwitscher_chuck_rest_ChuckJokeResource_getJoke{instance="localhost:12340",job="prometheus",quantile="0.95"}
de_qaware_playground_zwitscher_chuck_rest_ChuckJokeResource_getJoke{instance="localhost:12340",job="prometheus",quantile="0.75"}
de_qaware_playground_zwitscher_chuck_rest_ChuckJokeResource_getJoke{instance="localhost:12340",job="prometheus",quantile="0.5"}

HYSTRIX DASHBOARD



TALK CHEAP

SHOW ME

THE CODE

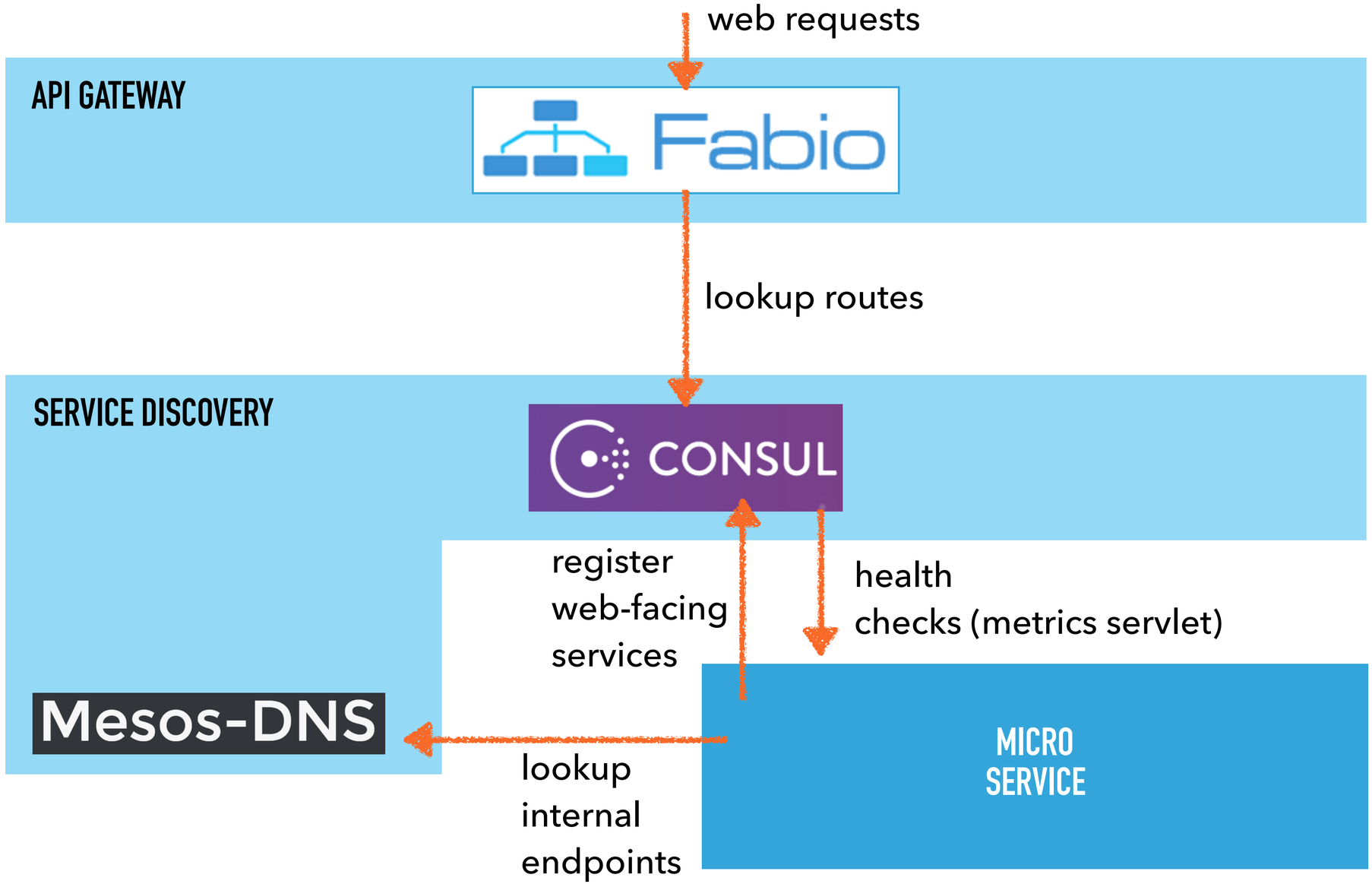
SERVICE DISCOVERY

- SERVICE REGISTRATION
- SERVICE LOOKUP
- SERVICE HEALTH CHECKING

& API GATEWAY

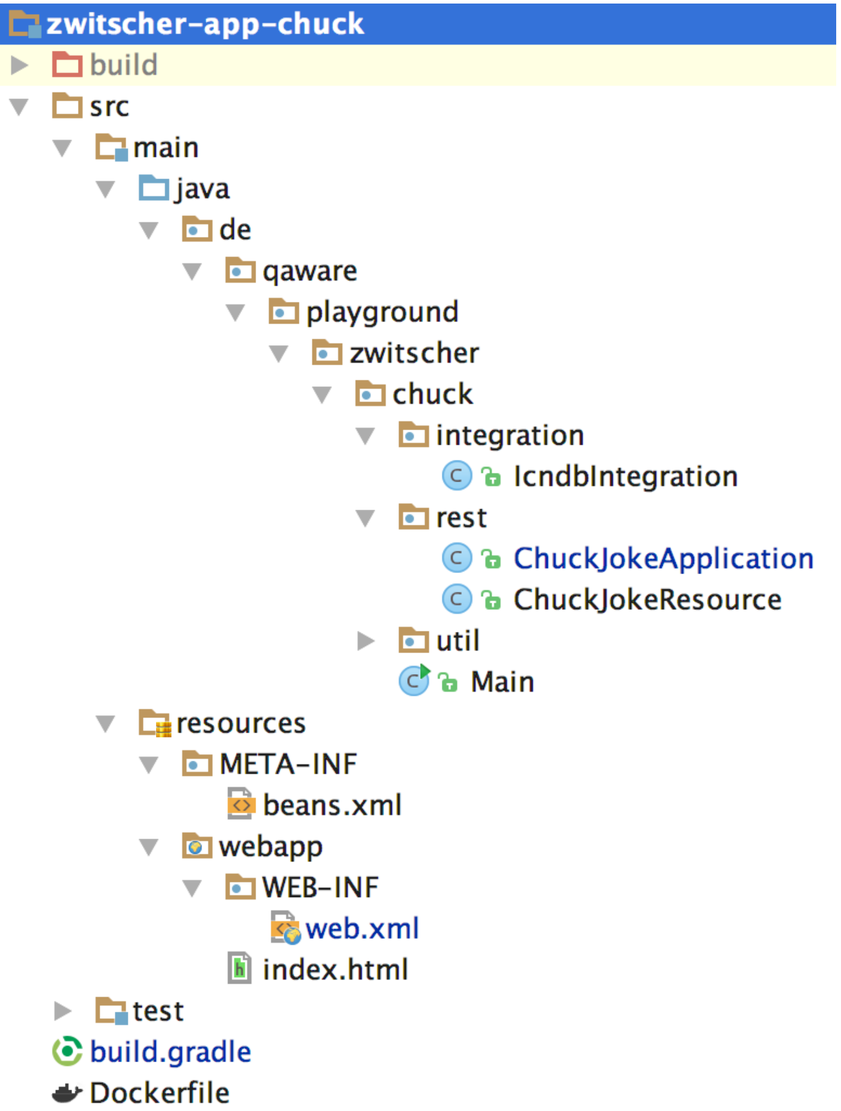
- API EXPOSITION & REQUEST ROUTING
- AUTHENTICATION & AUTORISATION
- LOAD BALANCING & SHEDDING
- RATE LIMITING
- REQUEST MONITORING & AUDITING

THE BIG PICTURE




```
java.security.Security.setProperty("networkaddress.cache.ttl", "0" );
```

SERVICE REGISTRATION WITHIN A JEE MICROSERVICE




```
@ApplicationPath("/chuck")
public class ChuckJokeApplication extends ResourceConfig {

    @Inject
    public ChuckJokeApplication(
            @ConsulFabio IServiceDiscovery serviceDiscovery) {
        super();
        //Register service
        serviceDiscovery.registerService(
            "zwitscher-chuck", "/chuck/joke");
    }
}
```

Service Name

URL Path to Service

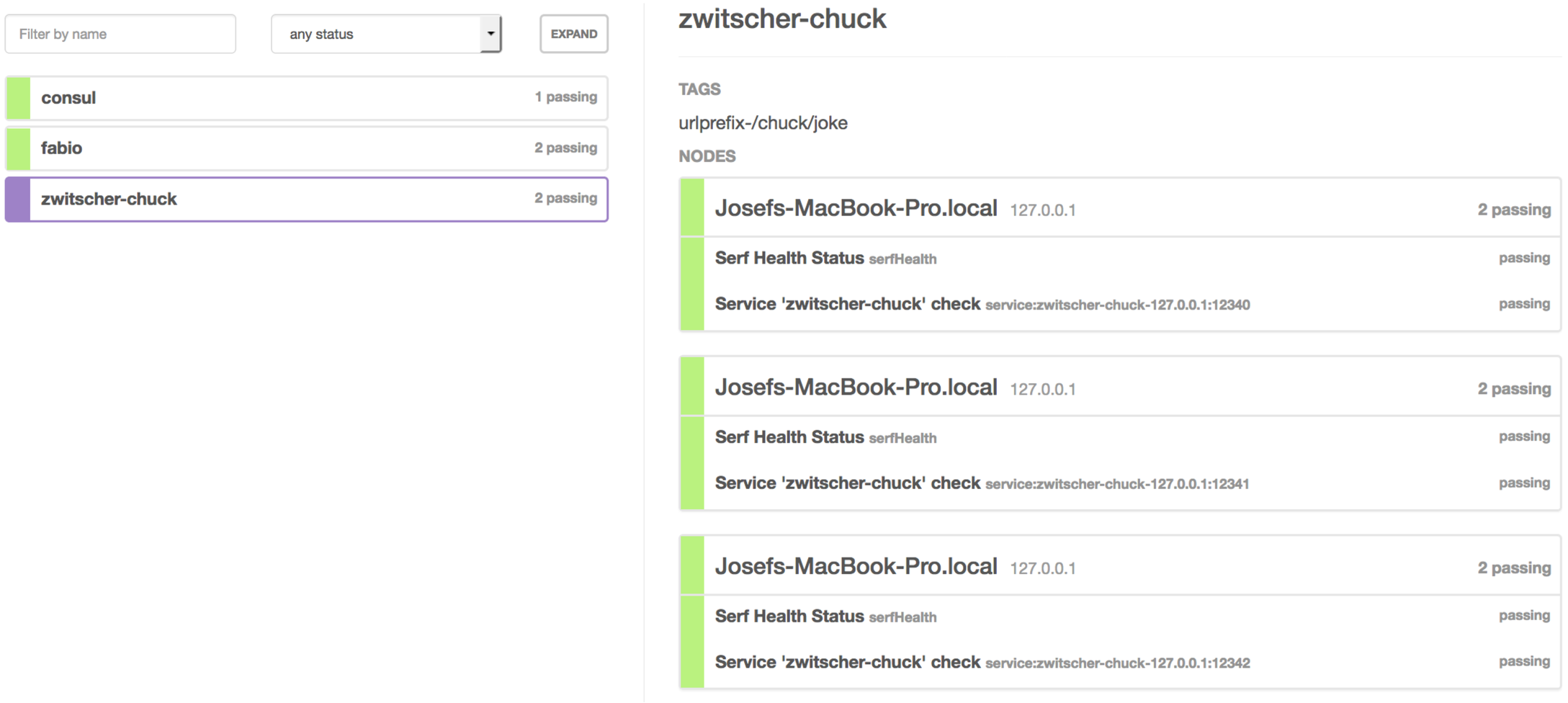
SERVICE REGISTRATION: THE HEAVY LIFTING

```
@ConsulFabio
@ApplicationScoped
public class ConsulFabioServiceDiscovery implements IServiceDiscovery {
```

```
/**
 * Registeres a service
 *
 * see https://github.com/eBay/fabio/wiki/Service-Configuration
 */
public synchronized void registerService(String serviceName, String servicePath) {

    String applicationHost = getOutboundHost();
    int applicationPort = getOutboundPort();
    HostAndPort consulEndpoint = getConsulHostAndPort();
    logger.info("Will register service on host {} and port {} at consul endpoint {}",
                applicationHost, applicationPort, consulEndpoint.toString());

    //generate unique serviceId
    String serviceId = serviceName + "-" + applicationHost + ":" + applicationPort;
    String fabioServiceTag = "urlprefix-" + servicePath;

    //point healthcheck URL to dropwizard metrics healthcheck servlet
    URL serviceUrl = UrlBuilder.empty()
            .withScheme("http")
            .withHost(applicationHost)
            .withPort(applicationPort)
            .withPath("/metrics/ping").toUrl();

    // Service bei Consul registrieren inklusive einem Health-Check auf die URL des REST-Endpunkts.
    logger.info("Registering service with ID {} and NAME {} with healthcheck URL {} and inbound ROUTE {}",
            serviceId, serviceName, serviceUrl, fabioServiceTag);

    //use consul API to register service
    ConsulClient client = new ConsulClient(consulEndpoint.toString());
    NewService service = new NewService();
    service.setId(serviceId);
    service.setName(serviceName);
    service.setPort(applicationPort);
    service.setAddress(applicationHost);
    List<String> tags = new ArrayList<>();
    tags.add(fabioServiceTag);
    service.setTags(tags);
    //register health check
    NewService.Check check = new NewService.Check();
    check.setHttp(serviceUrl.toString());
    check.setInterval(ConsulFabioServiceDiscovery.HEALTHCHECK_INTERVAL + "s");
    service.setCheck(check);
    client.agentServiceRegister(service);
}

public static String getOutboundHost() {
    String hostName = System.getenv(HOSTNAME_ENVVAR);
    String host = System.getenv(HOST_ENVVAR);
    if (hostName == null && host == null) return DEFAULT_HOST;
    else if (host != null) return host;
    else {
        File etcHosts = new File("/etc/hosts");
        List<String> lines;
        try {
            lines = Files.readLines(etcHosts, Charset.defaultCharset());
        } catch (IOException e) {
            return DEFAULT_HOST;
        }
        for (String line: lines){
            if (!line.trim().startsWith("#") && !line.trim().isEmpty()) {
                String[] etcEntry = line.split("\\s+");
                if (etcEntry[1].equals(hostName)) return etcEntry[0];
            }
        }
        return DEFAULT_HOST;
    }
}

public static int getOutboundPort() {
    String portEnv = System.getenv(PORT_ENVVAR);
    if (portEnv == null) return DEFAULT_PORT;
    return Integer.valueOf(portEnv);
}

public static HostAndPort getConsulHostAndPort() {
    String consulEnv = System.getenv(CONSUL_ENVVAR);
    if (consulEnv == null) return HostAndPort.fromString(CONSUL_DEFAULT_HOSTANDPORT);
    else return HostAndPort.fromString(consulEnv);
}
```

- ☑ figure out Consul-visible host name and port
- ☑ compose health check
- ☑ add meta data for Fabio
- ☑ register service at Consul
  (as well as API doc and diagnosability endpoints)

ET VOILA: CONSUL (STARTING 3 INSTANCES)



ET VOILA: FABIO

./fabio Overrides

Routing Table

type to filter routes

#	Service	Host	Path	Dest	Weight
1	zwitscher-chuck		/chuck/joke	http://127.0.0.1:12342/	33.33333333333333%
2	zwitscher-chuck		/chuck/joke	http://127.0.0.1:12341/	33.33333333333333%
3	zwitscher-chuck		/chuck/joke	http://127.0.0.1:12340/	33.33333333333333%

CLOUD NATIVE JEE

APPOP'S VIEW

SELF-DELIVERING SOFTWARE

EVERYTHING AS CODE:

- codes **application**
- codes **tests**
- codes **infrastructure**
- codes **delivery workflow**

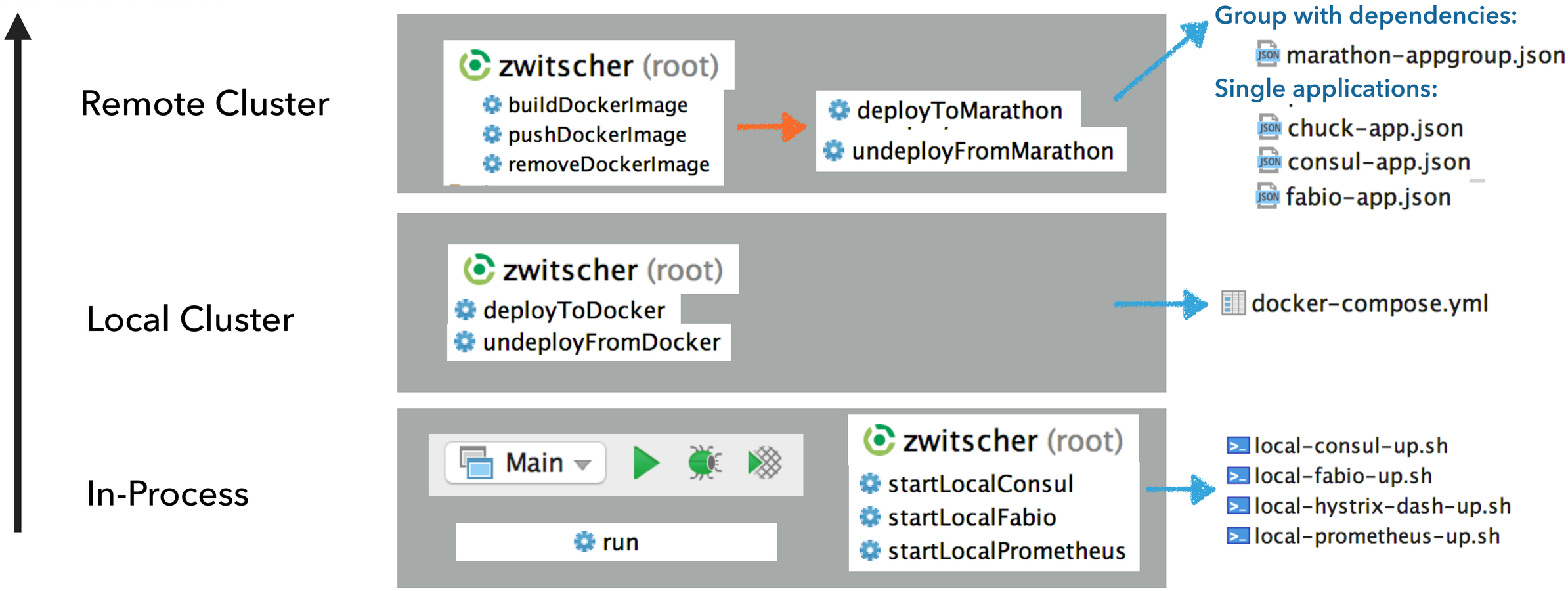
Gradle

git

RUNNING
TESTED
SOFTWARE
INCREMENT

FOCUS ON SHORT ROUND TRIPS WITH MULTIPLE RUN MODES

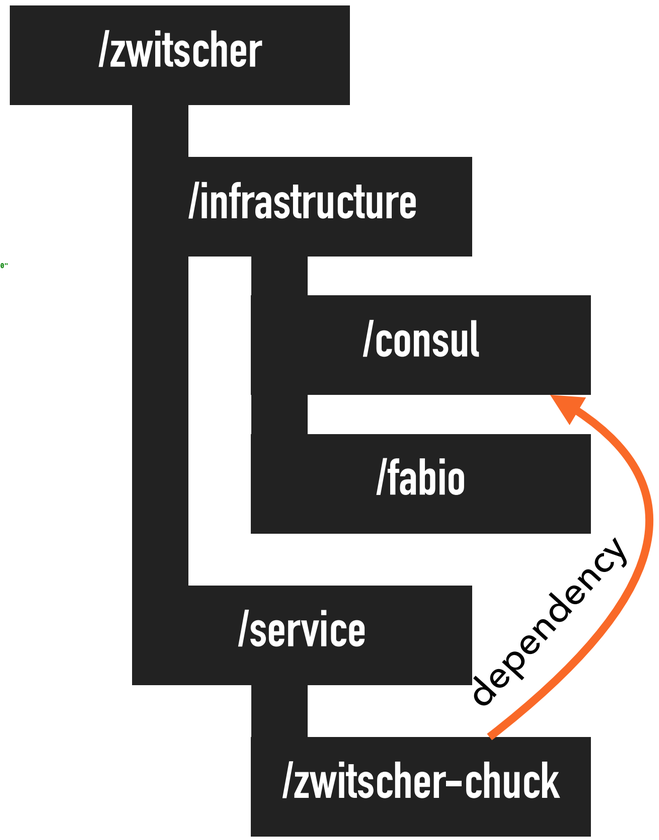
- longer round trip time
- but: more realistic



MARATHON APP DEFINITION

marathon-appgroup.json


```
{
  "id": "/zwitscher",
  "groups": [
    {
      "id": "/zwitscher/infrastructure",
      "apps": [
        {
          "id": "consul",
          "cpus": 1,
          "mem": 256,
          "disk": 0,
          "instances": 1,
          "cmd": "/bin/consul agent -server -ui -advertise=$HOST -config-dir=/config -data-dir=/tmp/consul -bootstrap-expect=1 -node=consul-server -client=0.0.0.0",
          "container": {
            "docker": {
              "image": "gliderlabs/consul-server:0.6",
              "forcePullImage": true,
              "privileged": false,
              "network": "HOST",
              "portDefinitions": [
                { "port": 8300, "protocol": "tcp", "name": "server-rpc" },
                { "port": 8301, "protocol": "tcp", "name": "serf-lan" },
                { "port": 8302, "protocol": "tcp", "name": "serf-wan" },
                { "port": 8400, "protocol": "tcp", "name": "cli-rpc" },
                { "port": 8500, "protocol": "tcp", "name": "http-api" },
                { "port": 8600, "protocol": "udp", "name": "dns" }
              ],
              "requirePorts" : true
            }
          },
          "env": {
            "GOMAXPROCS": "10"
          },
          "healthChecks": [
            {
              "protocol": "HTTP",
              "port": 8500,
              "path": "/v1/status/leader",
              "intervalSeconds": 10,
              "timeoutSeconds": 10,
              "maxConsecutiveFailures": 3
            }
          ]
        },
        {
          "id": "fabio",
          "cpus": 1,
          "mem": 256,
          "disk": 0,
          "instances": 1,
          "env": {
            "registry_consul_addr": "consul.infrastructure.zwitscher.marathon.mesos:8500"
          },
          "container": {
            "docker": {
              "image": "magiconair/fabio:latest",
              "forcePullImage": true,
              "privileged": false,
              "network": "HOST",
              "portDefinitions": [
                { "port": 9998, "protocol": "tcp", "name": "web-ui" },
                { "port": 9999, "protocol": "tcp", "name": "proxy-port" }
              ],
              "requirePorts" : true
            }
          },
          "acceptedResourceRoles":["slave_public"],
          "healthChecks": [
            {
              "protocol": "HTTP",
              "port": 9998,
              "path": "/health",
              "intervalSeconds": 10,
              "timeoutSeconds": 10,
              "maxConsecutiveFailures": 3
            }
          ]
        }
      ]
    },
    {
      "id": "/zwitscher/services",
      "apps": [
        {
          "id": "zwitscher-chuck",
          "cpus": 1,
          "mem": 256,
          "disk": 0,
          "instances": 1,
          "container": {
            "docker": {
              "image": "adersberger/zwitscher-app-chuck:1.0.0-SNAPSHOT",
              "forcePullImage": true,
              "privileged": false,
              "network": "HOST",
              "portDefinitions": [
                { "port": 12340, "protocol": "tcp", "name": "rest-api" }
              ],
              "requirePorts" : true
            }
          },
          "env": {
            "PORT": "12340",
            "CONSUL": "consul.infrastructure.zwitscher.marathon.mesos:8500",
            "CONFIG_ENV" : "zwitscher"
          },
          "args": [
            "-Xmx256m"
          ],
          "healthChecks": [
            {
              "protocol": "HTTP",
              "port": 12340,
              "path": "/metrics/ping",
              "intervalSeconds": 10,
              "timeoutSeconds": 10,
              "maxConsecutiveFailures": 3
            }
          ],
          "dependencies": [
            "/zwitscher/infrastructure/consul"
          ]
        }
      ]
    }
  ]
}
```

Define health checks for every app:

```
"healthChecks": [
  {
    "protocol": “HTTP", "port": 9998, "path": "/health",
    "intervalSeconds": 10, "timeoutSeconds": 10, "maxConsecutiveFailures": 3
  }
]
```

HOST networking (so that Mesos-DNS works) with fixed ports:

```
"network": "HOST",
"ports": [9998, 9999],
"requirePorts" : true
```

Run API gateway on public slave (accessible from www):

```
"acceptedResourceRoles":["slave_public"]
```

Configuration @ startup with env vars and args:

```
"env": {
  "PORT": "12340",
  "CONSUL": "consul.infrastructure.zwitscher.marathon.mesos:8500"
},
"args": ["-Xmx256m"]
```

forcePullImage = true to get the latest pushed docker image:

```
"container": { "docker": {
    "image": "adersberger/zwitscher-app-chuck:1.0.0-SNAPSHOT",
    "forcePullImage": true
```

Yes, sometimes you need dependencies (but you should avoid them to get more resilient):

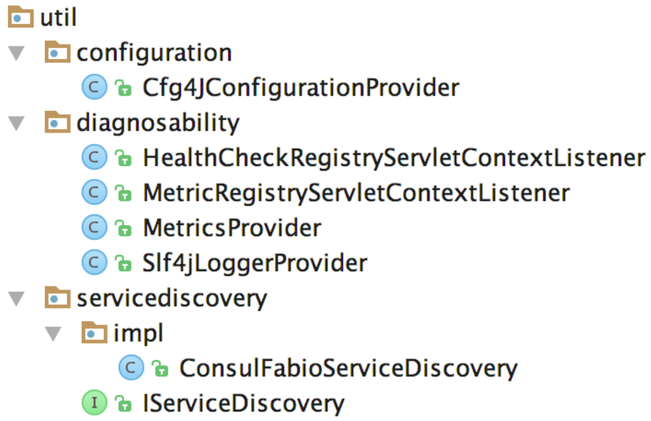
```
"dependencies": [ "/zwitscher/infrastructure/consul"]
```

CLOUD NATIVE JEE

SUMMARY

SUMMARY

- Implementing JEE microservices on DC/OS is simple & fun.
- The JEE APIs are out of the box not cloud native but can easily be enhanced with the required functionality. You can use our JCloudEE util classes to glue JEE together with cloud native tech.

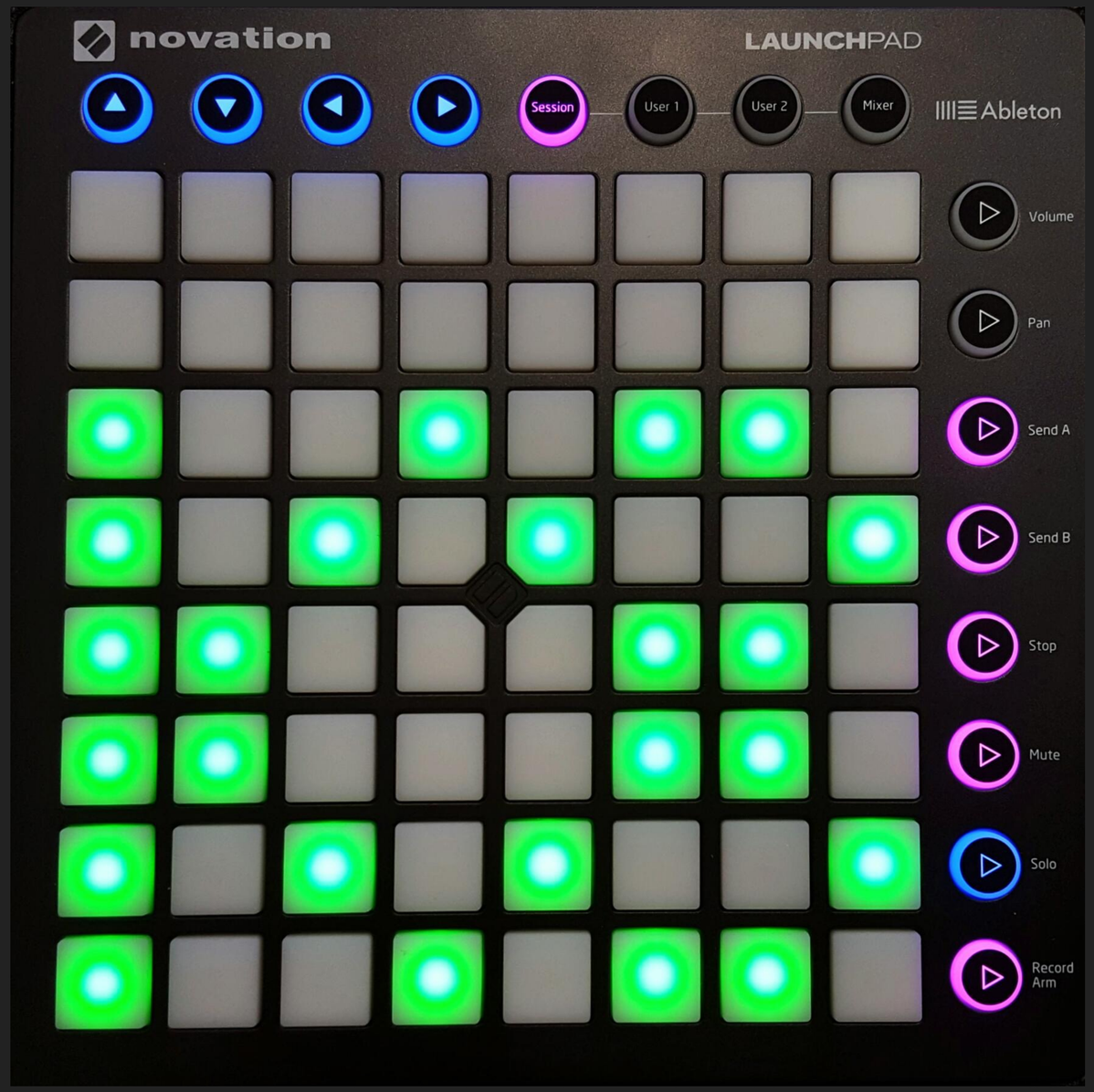


- Short round trip times are essential to be productive. You need full portability and automation from local host up to the cloud.

CLOUD NATIVE JEE

FUN!

https://github.com/qaware/kubepad



- First things first: Will be renamed to CloudPad soon!
- Controlling a DC/OS cluster with a DJ pad.
- Written in fancy Kotlin.
- Turned out to be really helpful for cloud native newbies to better grasp cluster orchestration.
- Kicky colored lights and well-hidden snake game easter egg. Set colors with app label.

JEE ON DC/OS 201

SNEAK PREVIEW

MESSAGING

WEB USER INTERFACE

SECURITY

STATEFUL SERVICES

Thank you!

Questions?

josef.adersberger@qaware.de
@adersberger
https://github.com/adersberger/cloud-native-zwitscher-jee

TWITTER.COM/QAWARE – SLIDESHARE.NET/QAWARE

BONUS SLIDES

TALK
CHEAP
SHOW ME
THE
CODE

CONFIGURATION & COORDINATION

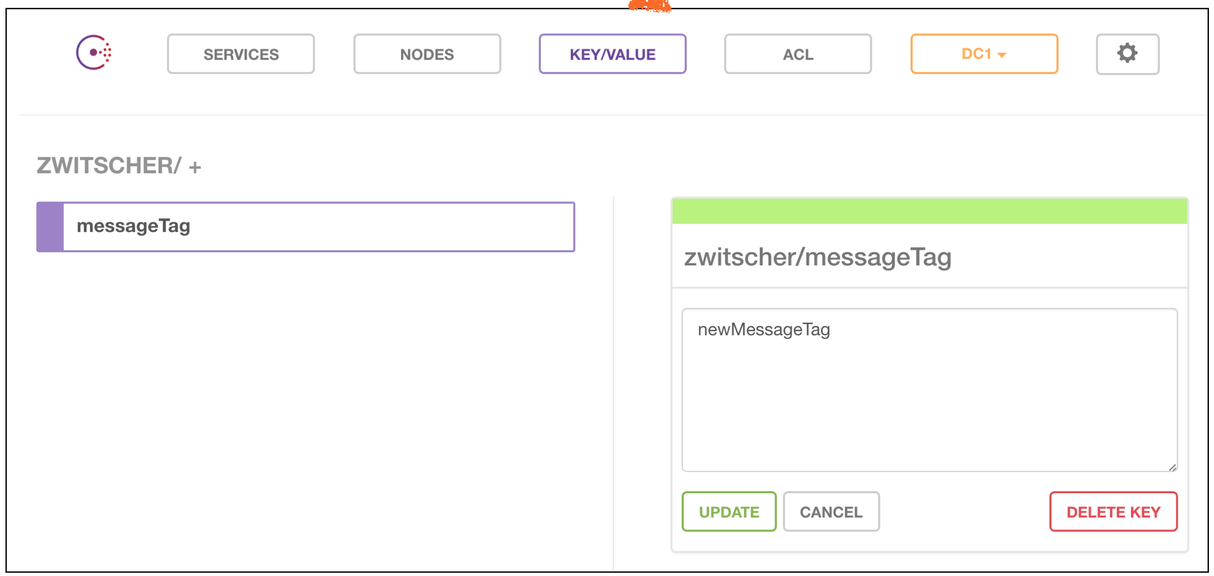
- KEY-VALUE STORE
- NOTIFICATIONS, EVENTS
- LOCKS
- LEADER ELECTIONS

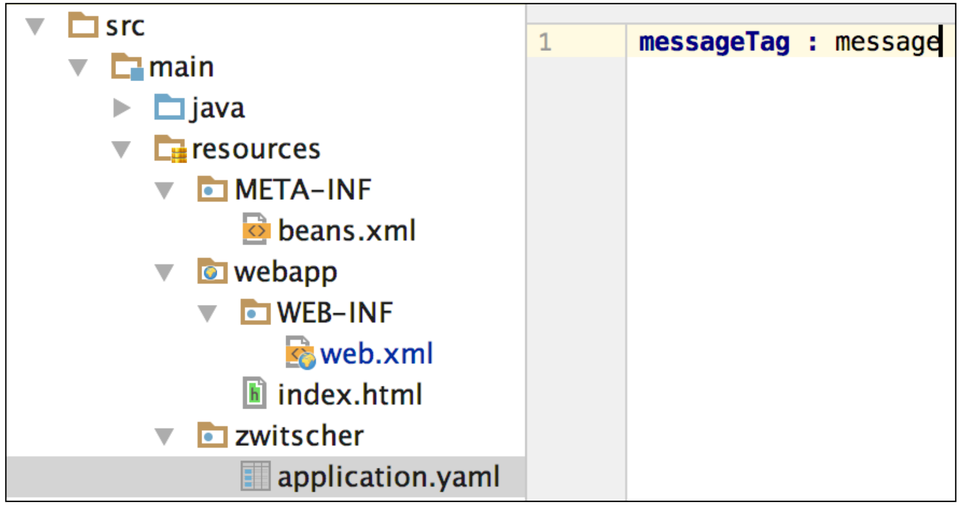
READING CONFIGURATION VALUES

```
@Inject
private ConfigurationProvider config;
//…
String messageTag = config.getProperty("messageTag", String.class);
```

try to read config value in Consul

fallback to file if Consul is not available or no config value is set in Consul

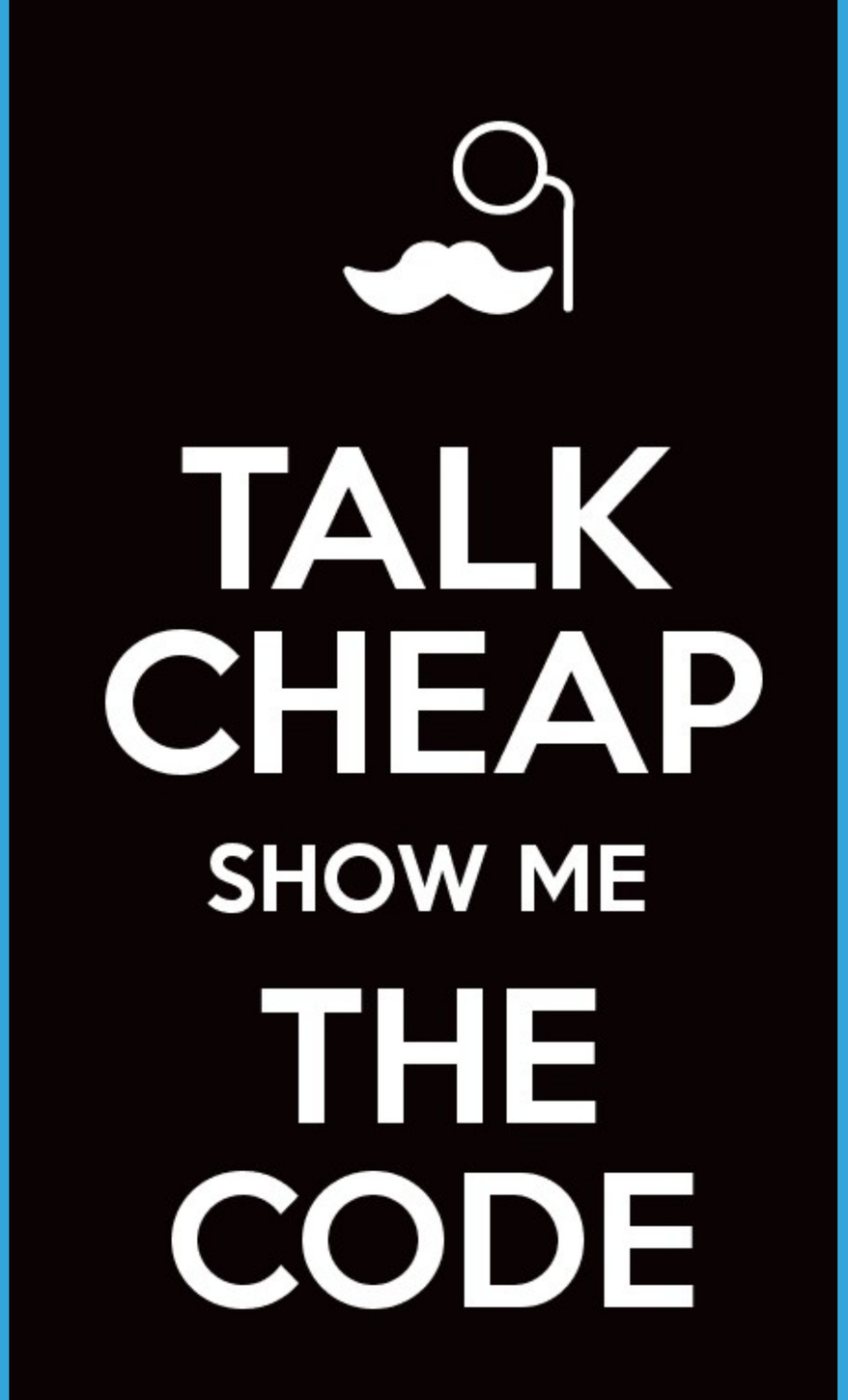




THE CONSUL AGENT AND CONFIG ENVIRONMENT CAN BE SET WITH A ENV VAR.

```
"env": {
  "CONFIG_ENV" : "zwitscher"
}
```

The config environment (e.g. “zwitscher”, “zwitscher-test”, “zwitscher-prod”). Refers to a path in Consul or to a local path where the config file is.



MONITORING & DIAGNOSABILITY

- COLLECT, STORE, ANALYZE METRICS
- COLLECT, STORE, ANALYZE LOGS
- COLLECT, STORE, ANALYZE TRACES
- DASHBOARDING & ALERTING

Logging

```
@Inject
private Logger logger;
```

```
@ApplicationScoped
public class Slf4jLoggerProvider {

    @Produces
    public Logger produceLogger(InjectionPoint injectionPoint) {
        return LoggerFactory.getLogger(
              injectionPoint.getMember()
                            .getDeclaringClass()
                            .getName());
    }

}
```

Business Metrics

```
@Inject
MetricRegistry metrics;

//…

metrics.counter("fun counter").inc();
```

Timers

```
@GET
@Timed
@Produces("application/json")
public Response getJoke() { //…
```