goto; chicago

Supersonic, Subatomic Eclipse MicroProfile

Scott M Stark

goto; chicago

Click ‘Rate Session’
to rate session
and ask questions.

follow us @gotochgo

QUARKUS

Introduction

Quarkus Features

Developer Joy ⌨

- Live Reload
- Imperative and Reactive
- ⚛ Serverless and Microservices

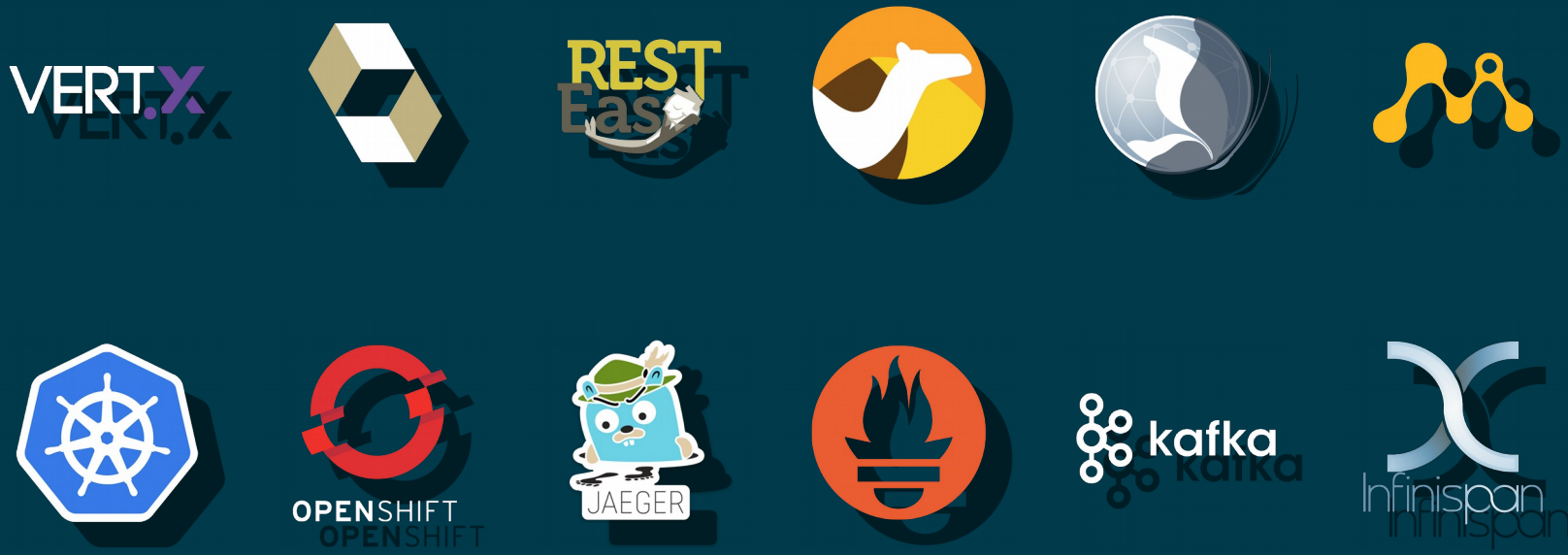
No compromises ✨

- Fat Jars and Native Executables
- ☯ Optimized for JAX-RS & JPA

Optimized for the Cloud

- Lower memory usage
- Faster startup
- ⏱ Optimized for short-lived processes
- ☸ Kubernetes Native

Supported Libraries and Standards



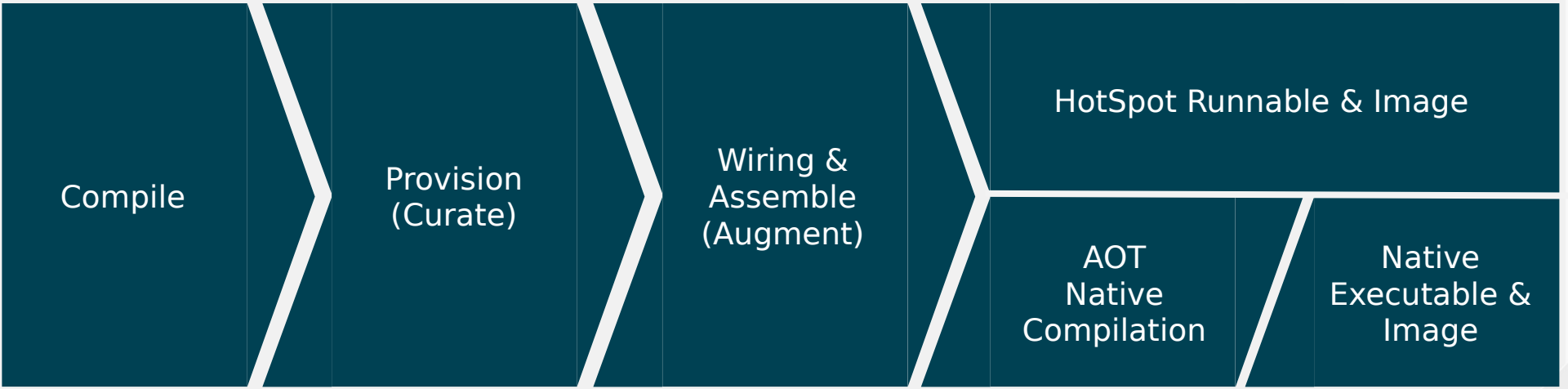
Framework Optimizations

- Moved as much as possible to build phase
- Minimized runtime dependencies
- Maximize dead code elimination
- Introduced clear metadata contracts
- Spectrum of optimization levels

(all -> some -> no runtime reflection)

Optimizations benefit both GraalVM native image and HotSpot

QUARKUS BUILD PROCESS

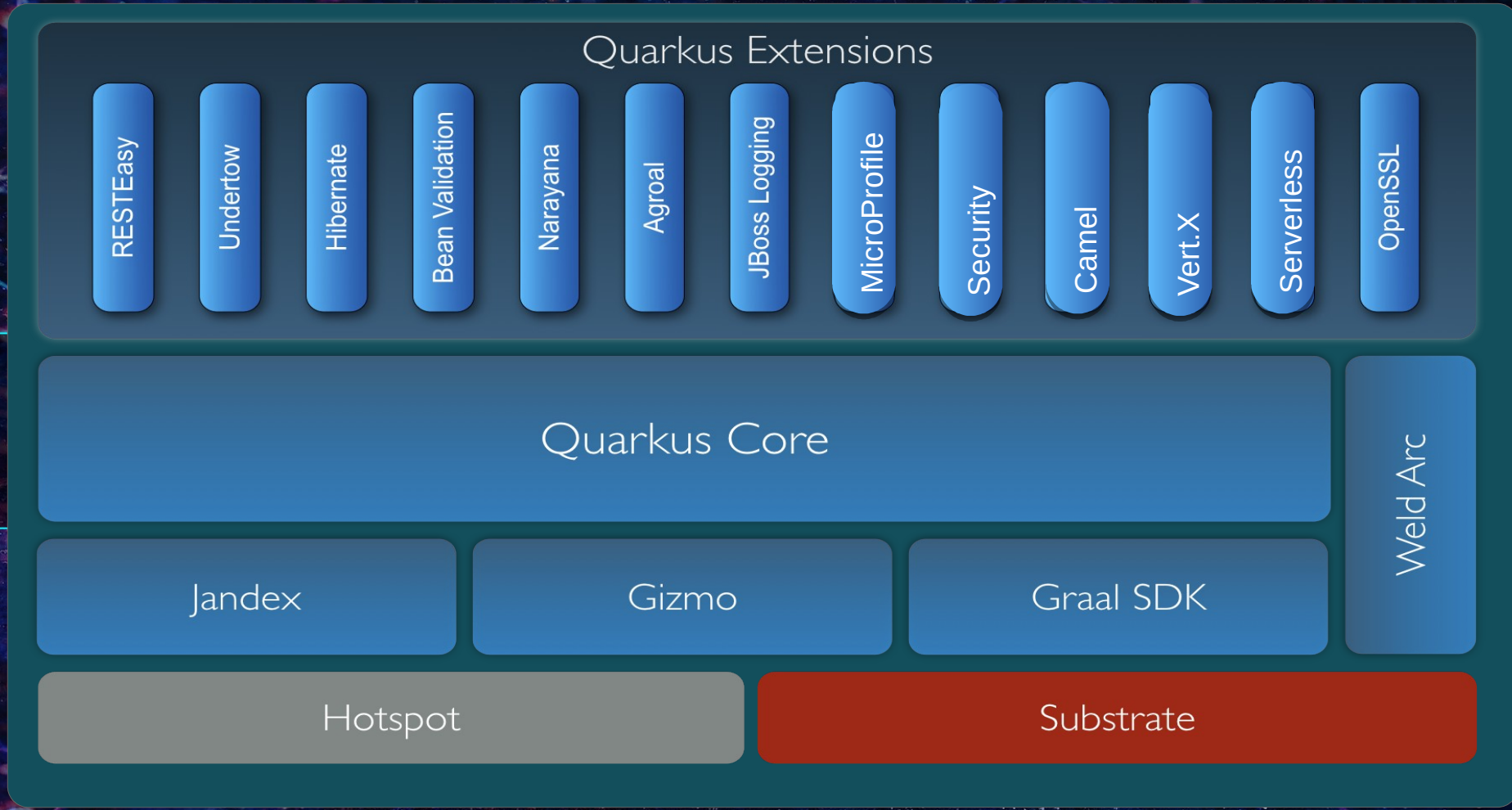


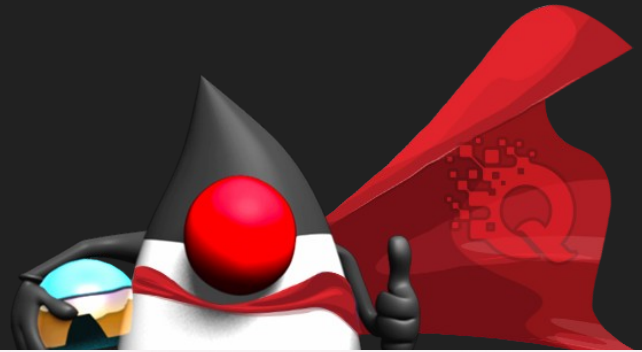
app.jar

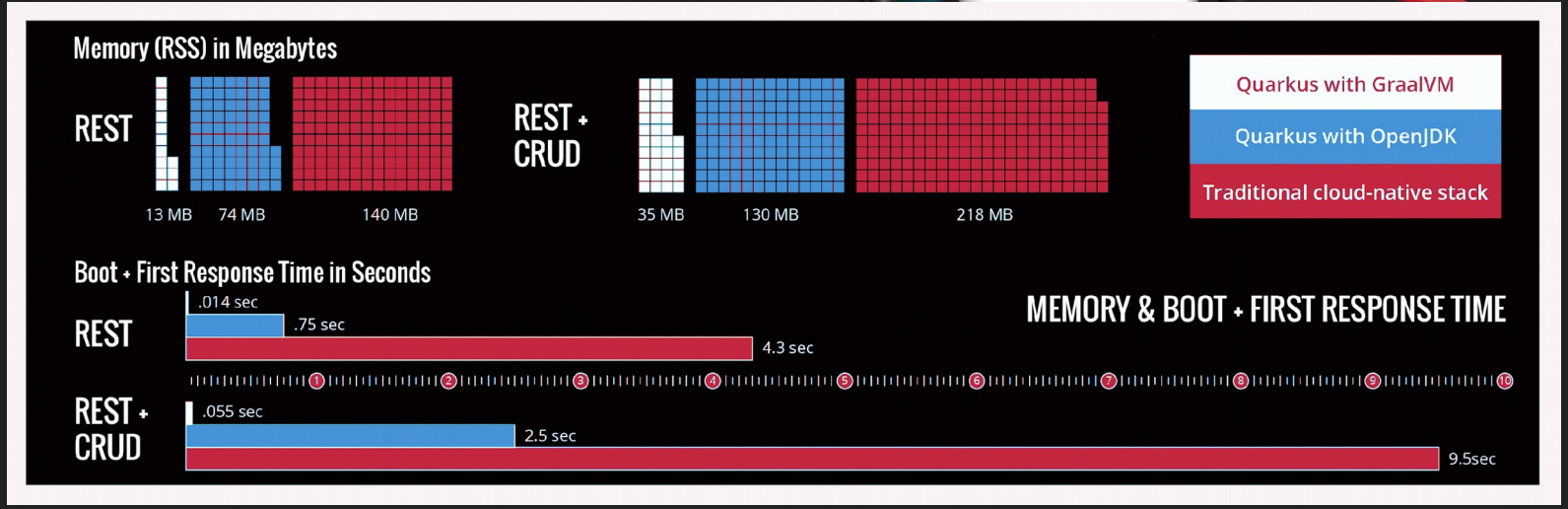
Frameworks

Runnable Java App

native-app

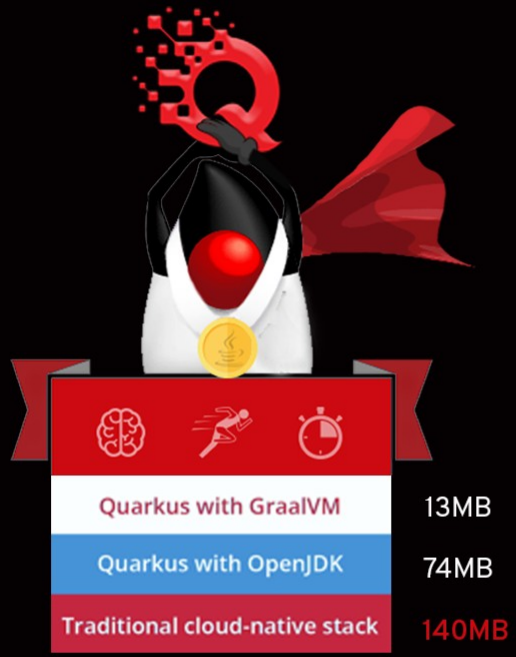

Quarkus Extensions
RESTEasy
Undertow
Hibernate
Bean Validation
Narayana
Agroal
JBoss Logging
MicroProfile
Security
Camel
Vert.X
Serverless
OpenSSL
Quarkus Core
Weld Arc
Jandex
Gizmo
Graal SDK
Hotspot
Substrate



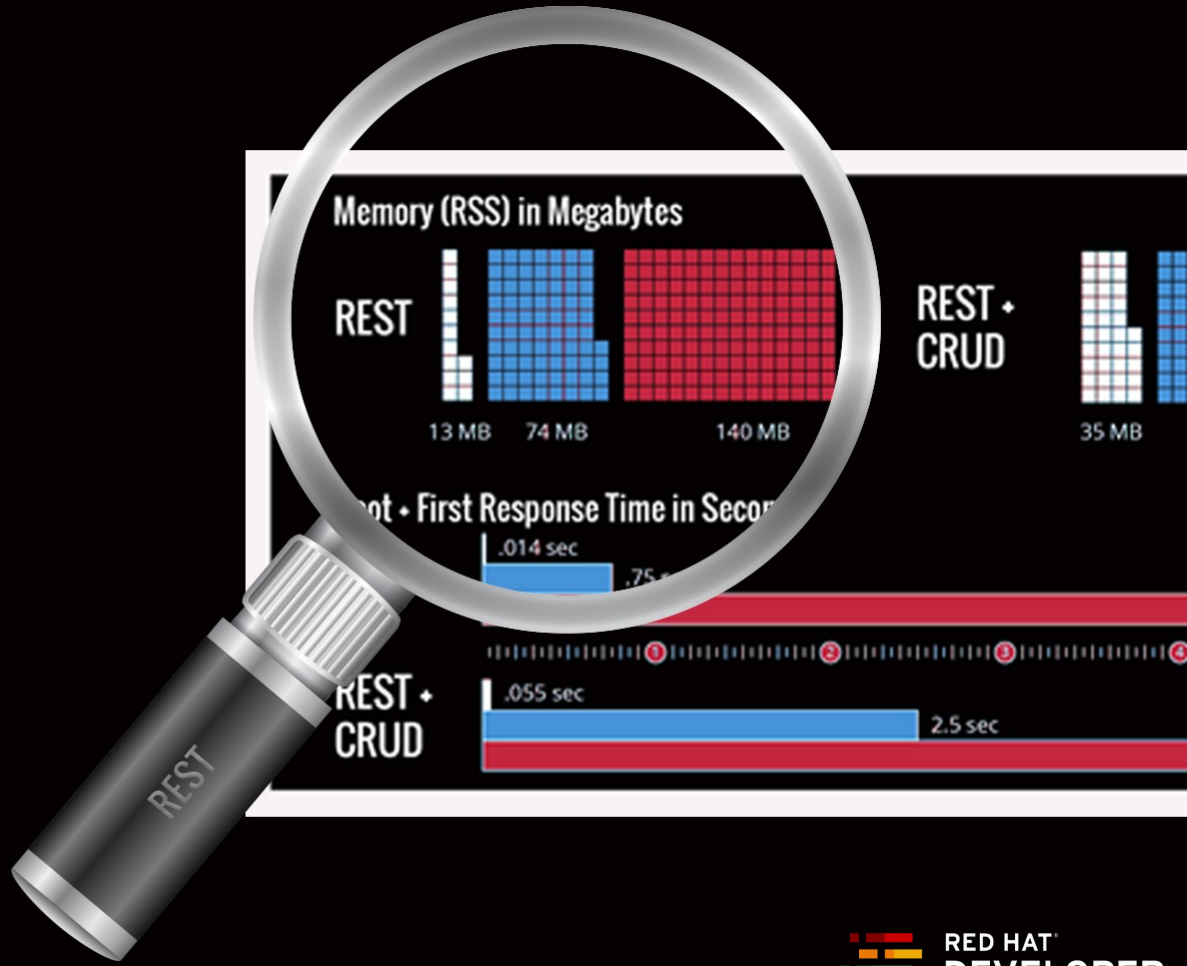


@burrsutter @jtgreene
@vanaga

RED HAT DEVELOPER

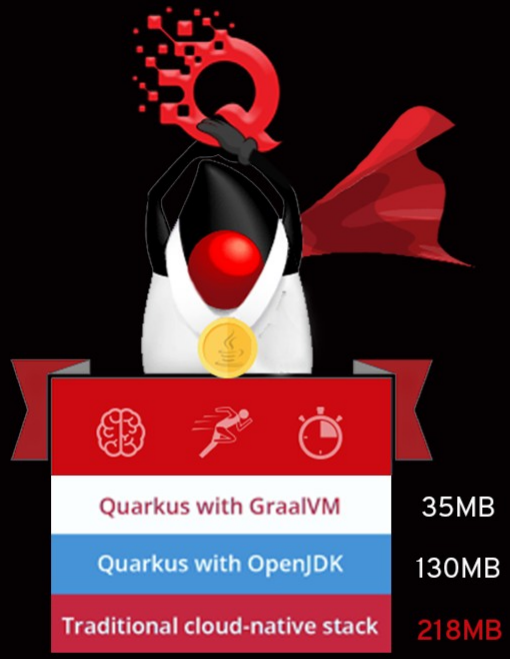


Memory (RSS) in Megabytes

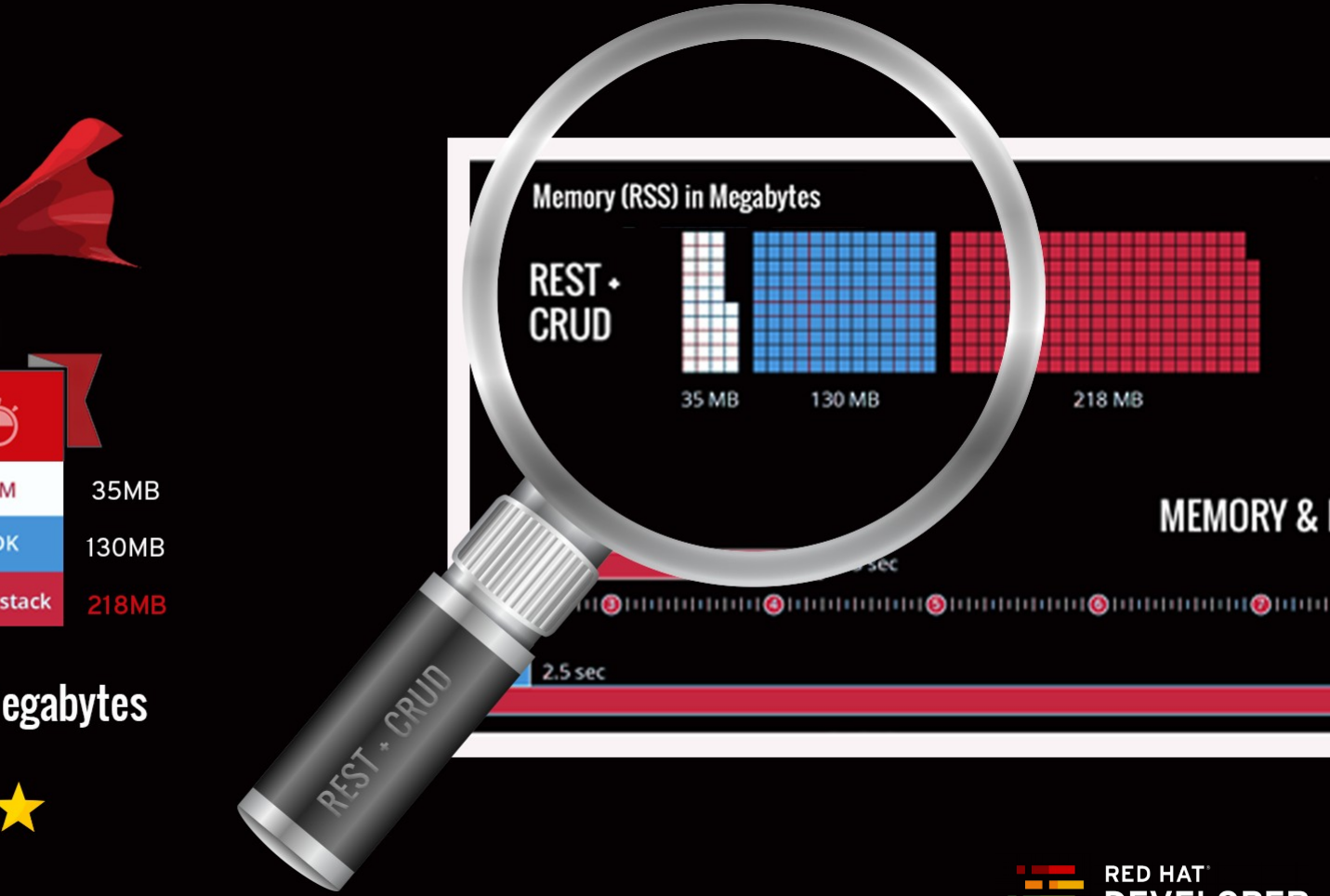


@burrsutter @jtgreene
@vanaga

RED HAT DEVELOPER



Memory (RSS) in Megabytes



★★★★★

@burrsutter @jtgreene
@vanaga

RED HAT DEVELOPER

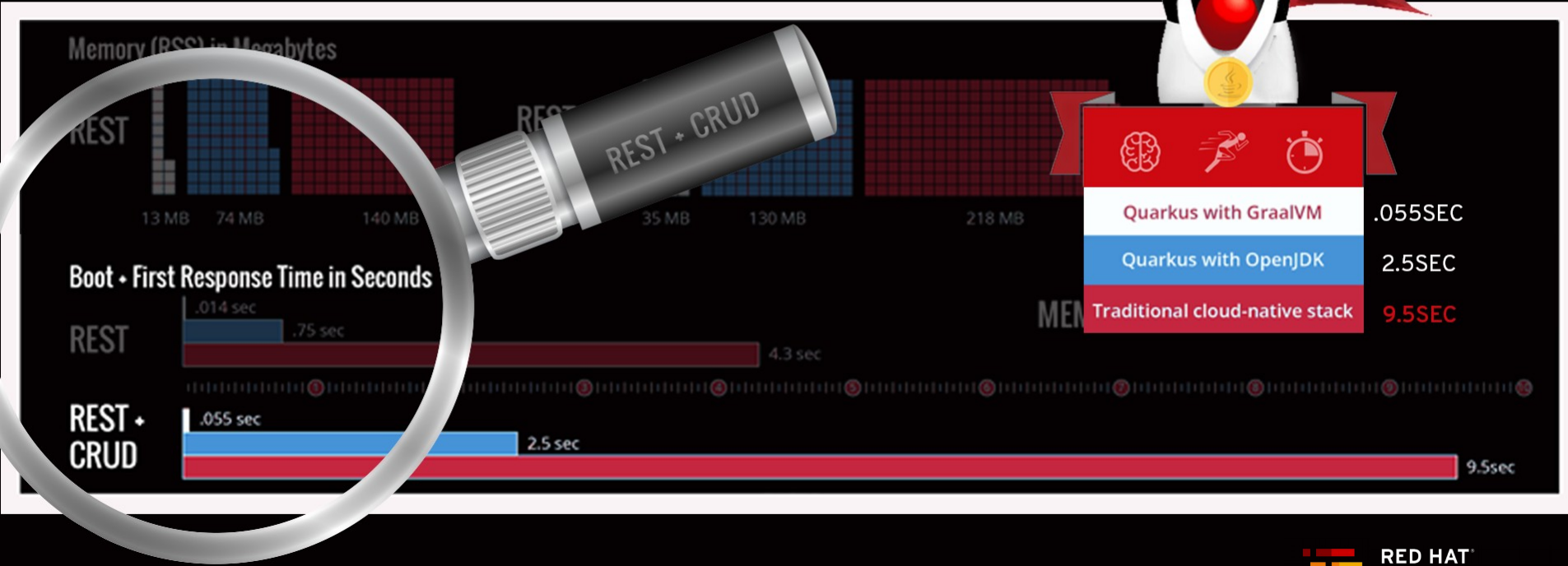
Boot + First Response Time in Seconds

★★★★★

@burrsutter @jtgreene
@yanaga

RED HAT DEVELOPER

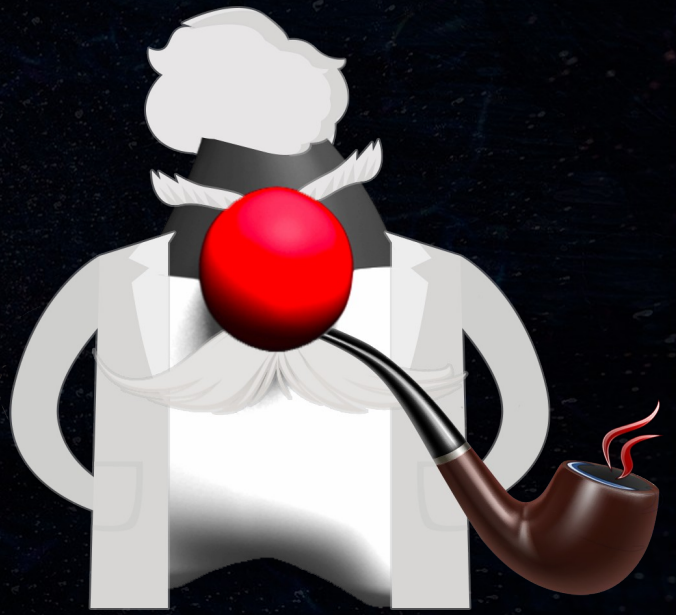
Boot + First Response Time in Seconds



Quarkus Website : https://quarkus.io/

- Getting started projects
  - Over 30+ quick start projects and guides
- Extensions
  - Over 45 extensions and guides
- Links to community and developer resources

“The reports of my death in the cloud are greatly exaggerated”



ECLIPSE FOUNDATION

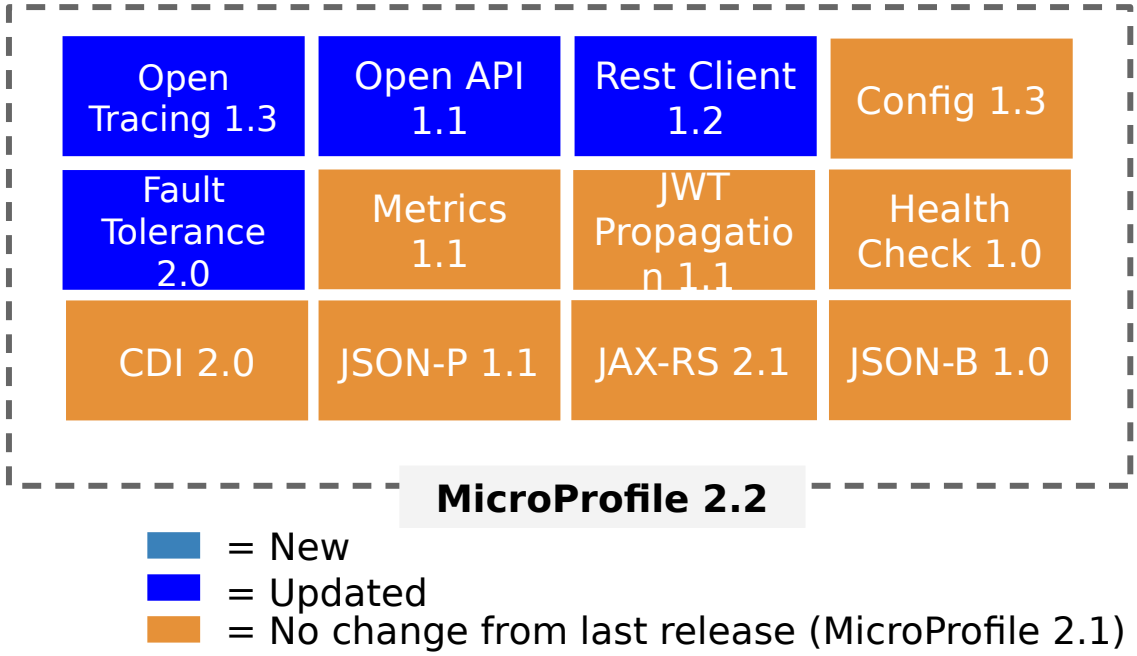
Eclipse MicroProfile 2.2

12th February 2019

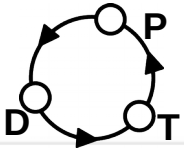
Eclipse MicroProfile Community

Community - individuals, organizations, vendors

Eclipse MicroProfile 2.2 (Feb 2019)



Configuration



- Support for SPI and CDI based usage
- Extensible configuration sources
  - Environment variables, System properties and META-INF/microprofile-config.properties default sources
  - Alternative sources based on ConfigMaps, YAML, JDBC, etc. possible
- Injection or programmatic lookup of config values with standard and custom type conversions

MP-Config Example


```
@Path("/config")
@RequestScoped
public class ConfigTestController {

    @Inject
    @ConfigProperty(name = "injected.value")
    String injectedValue;
    @Inject
    @ConfigProperty(name = "injected.piValue", defaultValue = "pi5=3.14159")
    Double piValue;

    @Path("/injected")
    @GET
    public String getInjectedConfigValue() {
       return "Config value as Injected by CDI " + injectedValue;
    }

    @Path("/injectedPi")
    @GET
    public String getInjectedPiValue() {
       return String.format("Injected Pi value: %.10f", piValue.doubleValue());
    }
...
```

MP-Config Example

...

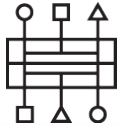
```
@Path("/lookup")
@GET
public String getLookupConfigValue() {
   Config config = ConfigProvider.getConfig();
    String value = config.getValue("value", String.class);
   return "Config value from ConfigProvider " + value;
}
```

MP-Config Example

- /application.properties

```
injected.value=Injected value
injected.piValue=3.1415926532
lookup.value=lookup value
```

OpenAPI



- Aims at providing a unified Java API for the OpenAPI v3 specification (OAS3), that all application developers can use to expose their API documentation.
- Uses annotations on the JAX-RS components to declare the corresponding OAS3 objects
  - Only needed to augment base model derived from the JAX-RS info
- The final OAS3 document can be augmented with static content, custom model readers and filters
- Swagger-UI integration in Quarkus dev/test mode

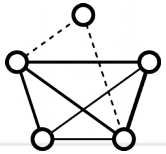
MP OpenAPI Example


```
@ApplicationPath("/demo1")
@LoginConfig(authMethod = "MP-JWT", realmName = "quarkus-quickstart")
@OpenAPIDefinition(
    info = @Info(
            title = "Quarkus MicroProfile 2.2 Extensions Demo",
            version = "1.0",
            contact = @Contact(
                    name = "QUARKUS - COMMUNITY",
                    url = "https://quarkus.io/community/",
                    email = "quarkus-dev+subscribe@googlegroups.com"),
            license = @License(
                    name = "Apache 2.0",
                    url = "http://www.apache.org/licenses/LICENSE-2.0.html"))
)
public class DemoRestApplication extends Application {
...
}
```

MP OpenAPI Example


```
@Path("/time")
@ApplicationScoped
public class TimeService {
...

    @Produces(MediaType.APPLICATION_JSON)
    @Tag(name = "time", description = "time service methods")
    @ExternalDocumentation(description = "Basic World Clock API Home.",
            url = "http://worldclockapi.com/")
    @Operation(summary = "Queries the WorldClockApi using the MP-RestClient",
            description = "Uses the WorldClockApi type proxy injected by the MP-
RestClient to access the worldclockapi.com service")
    public Now utc() {
...
}
```

MP OpenTracing



MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

- The MicroProfile OpenTracing (MP-OT) implements the https://opentracing.io/specification/ for instrumenting microservices for distributed tracing.
- This can be done via configuration of an io.opentracing.Tracer implementation without any changes to application code using MP Config
- This can be enhanced using MP-OT annotations in endpoint code
  - allows fine-tuned control over which classes and methods create OpenTracing spans
  - Can inject configured io.opentracing.Tracer for more complex tracing requirements, such as creating spans inside business methods.

MP OpenTracing Example

```
@ApplicationScoped
@Path("/traced")
public class TracedEndpoint {
    @GET
    @Path("/randomDelay")
    @Produces(MediaType.TEXT_PLAIN)
    @Traced(operationName = "TracedEndpoint#demoRandomDelay")
    public String randomDelay() {
...
    }

    @GET
    @Path("/untraced")
    @Produces(MediaType.TEXT_PLAIN)
    @Traced(false)
    public String untraced() {
...
}
```

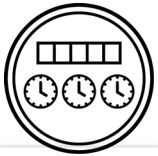
MP Health Check

- Health checks are used to probe the state of a computing node from another machine
  - Typically a service mesh environment like Kube or OpenShift
- MP-HC defines a Java API for specifying logical checks that should be invoked to determine overall health of application
- MP-HC defines a JSON response payload and HTTP status codes for the logical AND of the application health check procedures

MP Health Example


```
@Health
@ApplicationScoped
public class CheckDiskspace implements HealthCheck {
    @Inject
    @ConfigProperty(name = "health.pathToMonitor")
    String pathToMonitor;
    @Inject
    @ConfigProperty(name = "health.freeSpaceThreshold")
    long freeSpaceThreshold;

    @Override
    public HealthCheckResponse call() {
        HealthCheckResponseBuilder builder = HealthCheckResponse.named("diskspace");
        checkDiskspace(builder);
        return builder.build();
    }

    private void checkDiskspace(HealthCheckResponseBuilder builder) {
        File root = new File(pathToMonitor);
        long usableSpace = root.getUsableSpace();
        long freeSpace = root.getFreeSpace();
        builder.withData("usableSpace", usableSpace)
                .withData("freeSpace", freeSpace)
                .state(freeSpace >= freeSpaceThreshold);
    }
}
```

MP Metrics



MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

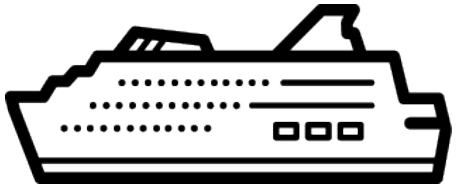
- To ensure reliable operation of software it is necessary to monitor essential system parameters.
- Metrics adds well-known monitoring endpoints and metrics for each microservice
  - base: metrics that all MicroProfile vendors have to provide
  - vendor: vendor specific metrics (optional)
  - application: application-specific metrics (optional)
- Counted, Gauge, Metered, Histogram and Timed metrics types are defined
- Both JSON and Prometheus text formats for query output is defined

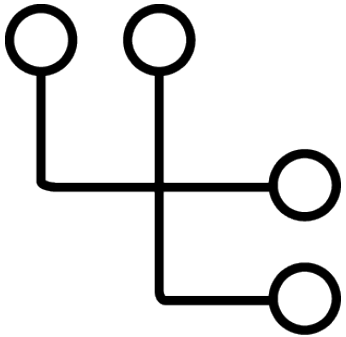
MP Metrics Example

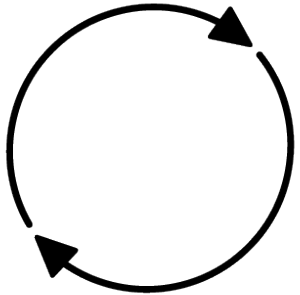

```
@Path("/metric")
@ApplicationScoped //Required for @Gauge
public class MetricController {

    @Inject
    @Metric(name = "endpoint_counter")
    private Counter counter;

    @Path("timed")
    @Timed(name = "timed-request")
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String timedRequest() {
        int wait = new Random().nextInt(1000);
...

        getCustomerCount();
        return "Request is used in statistics, check with the Metrics call.";
    }


    @Path("increment")
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public long doIncrement() {
        counter.inc();
        return counter.getCount();
    }

    @Gauge(name = "counter_gauge", unit = MetricUnits.NONE)
    private long getCustomerCount() {
        return counter.getCount();
    }
}
```

MP Fault Tolerance

Fault tolerance is about leveraging different strategies to guide the execution and result of some logic. Retry policies, bulkheads, and circuit breakers are popular concepts in this area. They dictate whether and when executions should take place, and fallbacks offer an alternative result when an execution does not complete successfully







MP FaultTolerance Example


```
@Path("/resilience")
@ApplicationScoped
public class ResilienceController {

    @Fallback(fallbackMethod = "fallback")
    @Timeout(500)
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String checkTimeout() {
        try {
            Thread.sleep(700L);
        } catch (InterruptedException e) {
            //
        }
        return "Never from normal processing";
    }

    public String fallback() {
        return "Fallback answer due to timeout";
    }
}
```

REST Client

MICROPROFILE
OPTIMIZING ENTERPRISE JAVA

- MicroProfile Rest Client API provides a type-safe approach to invoke RESTful services over HTTP in a consistent and easy-to-reuse fashion.
  - Standardizes non-standard proxy features that implementations have had
- Supports both CDI and builder API styles
  - Integrates with the JAX-RS providers configuration to support advanced setup of the invocation client
- Integrates with the fault tolerance, tracing and metrics projects to do the right thing when used in the context of those features

MP RestClient Example


```
@RegisterRestClient(baseUri = WorldClockApi.BASE_URL)
public interface WorldClockApi {
   static final String BASE_URL = "http://worldclockapi.com/api/json";

   @GET
   @Path("/utc/now")
   @Produces(MediaType.APPLICATION_JSON)
   Now utc();

   @GET
   @Path("{tz}/now")
   @Produces(MediaType.APPLICATION_JSON)
   Now tz(@PathParam("tz") String tz);
}
```

MP RestClient Example


```
@Path("/time")
@ApplicationScoped
public class TimeService {
    @Inject
    @RestClient
    WorldClockApi clockApi;

    @GET
    @Path("/now")
    @Produces(MediaType.APPLICATION_JSON)
    public Now utc() {
        return clockApi.utc();
    }
}
```

MP JWT Propagation

MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

- In a RESTful architecture style, services are usually stateless and any security state associated with a client is sent to the target service on every request in order to allow services to re-create a security context for the caller and perform both authentication and authorization checks
- MP-JWT standardizes expected claims and algorithms defined in the RFC-7519[JSON Web Tokens(JWT)] and related
- Defines a Bearer token authentication scheme RS256 signatures
- Defines a groups claim for RBAC authorization
- Adds support for injecting the JWT as a java.security.Principal extension
  - Also integrates this JWT with the various container security APIs
- Adds support for injection individual JWT claims, including the raw token value

MP-JWT Example

```
@Path("/jwt")
@DenyAll
public class JwtEndpoint {
    @Inject
    private JsonWebToken jwt;
    @Inject
    @Claim(standard = Claims.upn)
    private ClaimValue<String> upn;
    @Context
    private SecurityContext context;


    @GET
    @Path("/secureHello")
    @Produces(MediaType.TEXT_PLAIN)
    @RolesAllowed("user")
    public String secureHello() {
        String user = jwt == null ? "anonymous" : jwt.getName();
        String scheme = context.getAuthenticationScheme();
        boolean isUserInRole = context.isUserInRole("GOTO-attendee");
        return String.format("Hello[secure] user=%s, upn=%s, scheme=%s, isUserInRole(GOTO-attendee)=%s",
user, upn.getValue(), scheme, isUserInRole);
    }
```

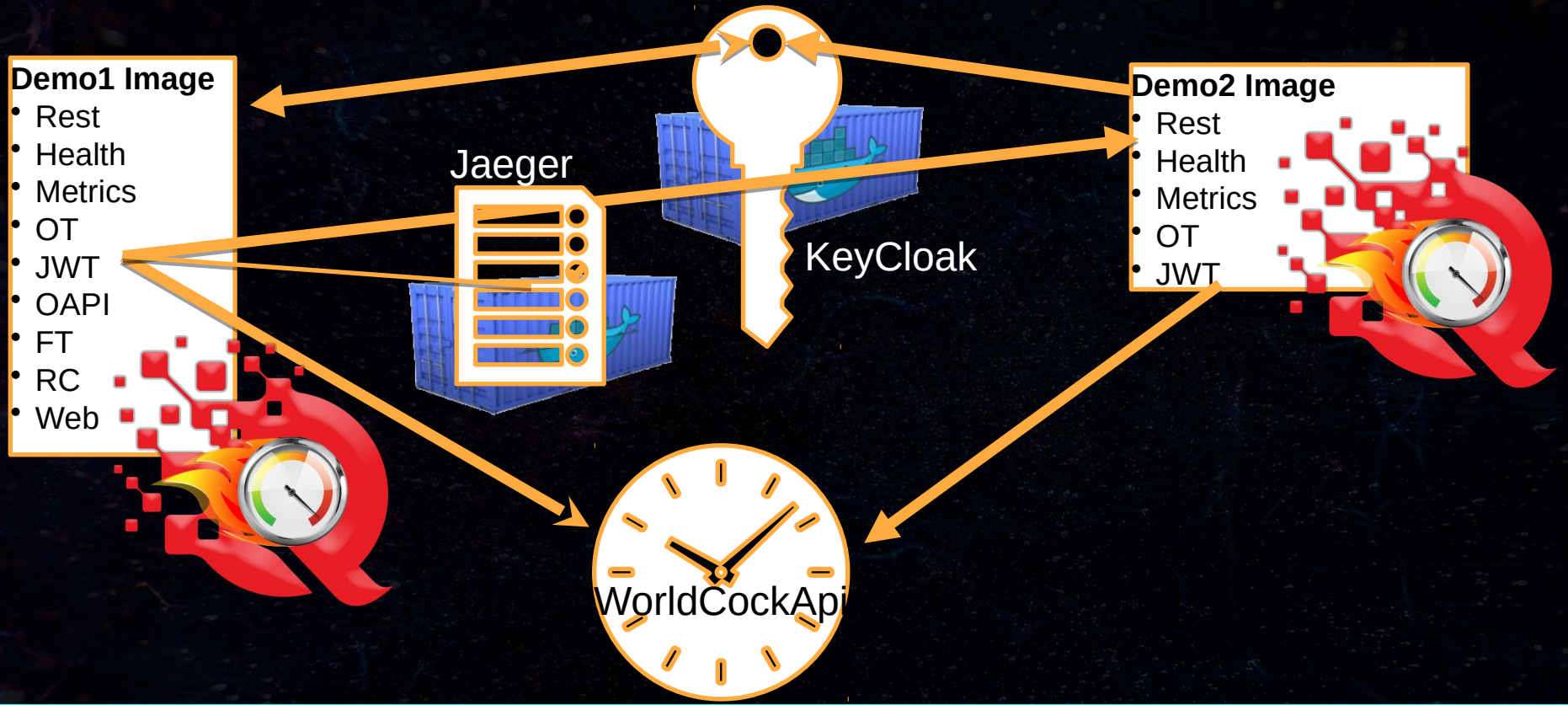
MP-JWT Example


```
@Path("/protected")
@RequestScoped
public class ZoneInfoEndpoint {

    @Inject
    @Claim("zoneinfo")
    private String zoneinfo;
    @Inject
    JsonWebToken jwt;

    @GET
    @Path("/userTZ")
    @RolesAllowed("WorldClockSubscriber")
    @Produces(MediaType.TEXT_PLAIN)
    @Timed
    public String getSubscriberZoneInfo() {
        System.out.printf("Zoneinfo for %s: %s\n", jwt.getName(), zoneinfo);
        return zoneinfo;
    }
}
```

MP-JWT Example

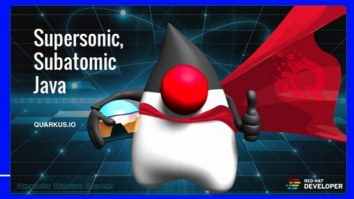
```
# MP-JWT Config

## JWK for public key
mp.jwt.verify.publickey.location=
http://localhost:8180/auth/realms/quarkus-quickstart/protocol/openid-connect/certs
## Expected issuer
mp.jwt.verify.issuer=http://localhost:8180/auth/realms/quarkus-quickstart
quarkus.smallrye-jwt.auth-mechanism=MP-JWT
quarkus.smallrye-jwt.enabled=true
```

Demo Architecture



Quarkusd MicroProfile Demo

A demo illustrating the MicroProfile 2.2 features in the Quarkus runtime.



Hello
Config
Health
Metrics
OpenTracing
OpenAPI
RestClient
FaultTolerance
JWT

Login Logout

Refresh ? 03:27 before token refresh

Auth Refresh Success

Token and Info

IDToken


```
"jti": "5a1bf023-52fc-4c1b-bc1d-c82d16b143c8",
"exp": 1556262960,
"nbf": 0,
"iat": 1556262660,
"iss": "http://localhost:8180/auth/realms/quarkus-quickstart",
"aud": "quarkus-front",
"sub": "a19b2afc-e96e-4939-82bf-aa4b589de136",
"typ": "ID",
"azp": "quarkus-front",
"nonce": "fd966bcc-eb40-48b6-8a6f-804cb63299d1",
"auth_time": 1556259891,
"session_state": "8241dac5-57af-4502-a14b-a0a931b4036f",
"acr": "0",
"upn": "test",
"email_verified": true,
"name": "Demo Tester",
"preferred_username": "test",
"given_name": "Demo",
"family_name": "Tester",
```


Access Token


```
{
    "jti": "13742402-417c-439b-9dcf-55c7d93feaaa",
    "exp": 1556262960,
    "nbf": 0,
    "iat": 1556262660,
    "iss": "http://localhost:8180/auth/realms/quarkus-quicks
    "aud": "account",
    "sub": "a19b2afc-e96e-4939-82bf-aa4b589de136",
    "typ": "Bearer",
    "azp": "quarkus-front",
    "nonce": "fd966bcc-eb40-48b6-8a6f-804cb63299d1",
    "auth_time": 1556259891,
    "session_state": "8241dac5-57af-4502-a14b-a0a931b4036f",
    "acr": "0",
    "allowed-origins": [
        "http://localhost:8080"
    ],
    "realm_access": {
        "roles": [
            "offline_access",
```

Code

https://github.com/starksm64/Qproj4MP.git

Ack

- Burr Sutter
- Jason Green
- Sebastien Blanc

goto; chicago

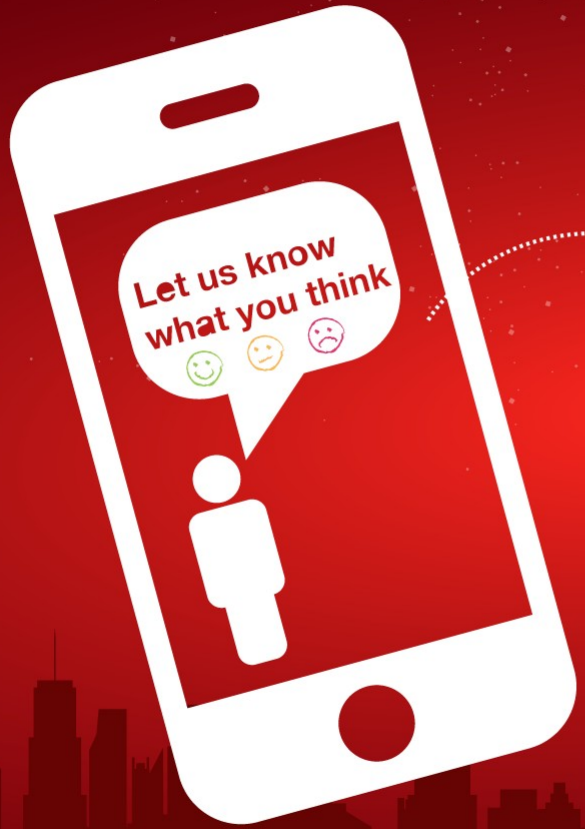
Please

Remember to rate this session

Thank you!

follow us @gotochgo

goto; chicago



Click ‘Rate Session’

Rate **5** sessions to get the supercool GOTO reward

follow us @gotochgo