Eclipse MicroProfile with Thorntail (formerly WildFly Swarm)

John Clingan
Senior Principal Product Manager

Ken Finnigan
Senior Principal Software Engineer
EVOLUTION OF MICROSERVICES (2014 - ?)

Support Services
> Smart Routing
> API Management
> Caching Service
> Configuration
> Messaging
> SSO
> Registry

Application Logic
> Client-side Load Balancing
> Service Registration
> Circuit Breaker
> Distributed Tracing

NetFlix

Ribbon

Eureka

Zuul

Config Server

Hystrix

Zipkin
WHAT WE ARE HEARING FROM CUSTOMERS

1. Cloud and microservices distributed architectures leads to the need for multiple runtimes, frameworks and languages

2. Microservices and distributed application development is hard. Need a platform that can abstract away complexity and simplify development.

3. While enterprise developers want choice of runtime/framework, the enterprise needs a way to standardize, support, and plan

4. Cannot just turn off or replace existing(legacy) apps. Need modernization ramp to harvest more value from existing apps.

5. Do not want cloud lock-in and restricted to specific application framework, architecture style, languages etc.
RED HAT OPENSSHIFT APPLICATION RUNTIMES
Providing curated set of integrated runtimes and frameworks that standardizes Cloud Native App

✓ Simplified development
✓ Strategic flexibility
✓ DevOps automation

YOUR APPS AND SERVICES

RED HAT OPENSIFT APPLICATION RUNTIMES
Launch Service
- Reactive Vert.x
- MicroProfile WildFly Swarm
- Spring Boot Tomcat
- JavaScript Node.js
- Java EE JBoss EAP

APPLICATIONS LIFECYCLE MANAGEMENT
SUPPORTING MIDDLEWARE SERVICES
CONTAINER ORCHESTRATION & MANAGEMENT (KUBERNETES)

RED HAT OPENSIFT

YOUR INFRASTRUCTURE

Laptop
Datacenter
OpenStack
Amazon Web Services
Microsoft Azure
Google Cloud
EVOLUTION OF MICROSERVICES (2014 - Current)

Application Logic
- Client-side Load Balancing
- Service Registration
- Circuit Breaker
- Distributed Tracing

Support Services
- Smart Routing
- API Management
- Caching Service
- Configuration
- Messaging
- SSO
- Registry

Support Services
- Distributed Tracing
- API Management
- Caching Service
- Messaging
- SSO

2014

Current
EVOLUTION OF MICROSERVICES (NEAR FUTURE)

Support Services
- Smart Routing
- API Management
- Caching Service
- Configuration
- Messaging
- SSO
- Registry

Application Logic
- Client-side Load Balancing
- Service Registration
- Circuit Breaker
- Distributed Tracing

Application Logic
- Client-side Load Balancing
- Circuit Breaker

Support Services
- Distributed Tracing
- API Management
- Caching Service
- Messaging
- SSO

Support Services
- Registry
- Configuration
- Server-side Load Balancing

Application Logic
- Registry
- Configuration
- Server-side Load Balancing

Support Services
- API Management
- Caching Service
- Messaging
- SSO

Application Logic
- Server-side Load Balancing
- Circuit Breaker
- Fault Injection

2014 Current Near Future
For Java EE, how does this change how I write my code?
<table>
<thead>
<tr>
<th>Architecture vs API vs Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microservices</strong> is an architectural approach</td>
</tr>
<tr>
<td><strong>Java EE</strong> and MicroProfile are specifications enabling innovation on implementation</td>
</tr>
<tr>
<td><strong>Java EE</strong> applications can be built as collaborating microservices on a lightweight runtime</td>
</tr>
</tbody>
</table>
● Defines open source Java microservices specifications
● Industry Collaboration - Red Hat, IBM, Payara, Tomitribe, London Java Community, SouJava, Oracle, Hazelcast, Fujitsu, SmartBear...
● WildFly Swarm is Red Hat’s implementation
● Minimum footprint for Enterprise Java cloud-native services (v1.3):
Thorntail  (formerly WildFly Swarm)
What's Thorntail?

- Formerly WildFly Swarm
- Rename process in community
- New name announced last week
- Rename work still to be done
Demo
What’s in the Demo?

- Showcases Eclipse MicroProfile specifications
- Three services: Greeting, Boring Name, Cute Name

https://github.com/kenfinnigan/RedHatSummit-2018
What’s in the Demo?

Greeting

Greeting

Boring Name

Cute Name
Eclipse MicroProfile
Roadmap

● January 2018 - 1.3
  ○ MP 1.2 + OpenTracing + OpenAPI + Type Safe REST Client

● June 2018 - 1.4/2.0
  ○ MP 1.4
    ■ Updates to Config, JWT Propagation, Fault Tolerance, OpenTracing
  ○ MP 2.0
    ■ MP 1.4 + Java EE 8 base for CDI, JAX-RS, JSON-P, (+JSON-B?)
Config

@Inject
@ConfigProperty(name = "myapp.default-name")
private String defaultName;
@Health
@ApplicationScoped
public class GoodHealthCheck implements HealthCheck {
    public HealthCheckResponse call() {
        return HealthCheckResponse
                .named("MyHealthCheck")
                .withData("someKey", "aValue")
                .up()
                .build();
    }
}
Metrics

- @Counted
- @Gauge
- @Metered
- @Timed
Metrics

@Counted(name = "hCount",
    absolute = true,
    description = "# calls to /myendpoint",
    monotonic = true)

@GET
@Path("/myendpoint")
Public Response myEndpoint {}
Open API

- @OpenAPIDefinition
- @Operation
- @APIResponse
- @Content
- @Parameter
Open API

@OpenAPIDefinition(info = @Info(
    title = "My REST Application",
    version = "1.0"
))

@Operation(
    operationId = "getSomething",
    summary = "Get some data over REST"
)
Fault Tolerance

- @Asynchronous
- @Retry
- @Fallback
- @Timeout
- @CircuitBreaker
- @Bulkhead
Fault Tolerance

@Retry(maxRetries = 2)
@Fallback(fallbackMethod = "greetingFallback")
public Response greeting() {}
Type Safe REST Client

- Define interface of external service. As JAX-RS would
- RestClientBuilder instead of JAX-RS ClientBuilder
@Path("/api")

public interface CuteNameService {
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    @Path("/name")
    String getName();
}
Type Safe REST Client

CuteNameService service =

    RestClientBuilder.newBuilder()
    .baseUrl(new URL(NAME_SERVICE_URL))
    .build(CuteNameService.class);

    String name = service.getName();
Open Tracing

- Add `jaeger-core` dependency to activate
- `@Traced` for fine grain control
- Set jaeger configuration
JWT Propagation

@Inject
private JwtWebToken jwtPrincipal;

@Inject
@Claim(standard = Claims.raw_token)
private ClaimValue<String> rawToken;
Thorntail
Thorntail Rename

- “org.wildfly.swarm” -> “io.thorntail”
- 2018.6.0 -> 2.0.0.Final
- Update documentation, examples, etc
- Follow Google Group community for updates
Not current WildFly Swarm!
Thorntail Roadmap

- 2.0.0.Final release in June
- PoC -> Thorntail 4.x [timing tbd]
- JDK 10 / 11 support
- Java EE 8 / Jakarta EE
- MicroProfile 1.4 / 2.0
THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHat