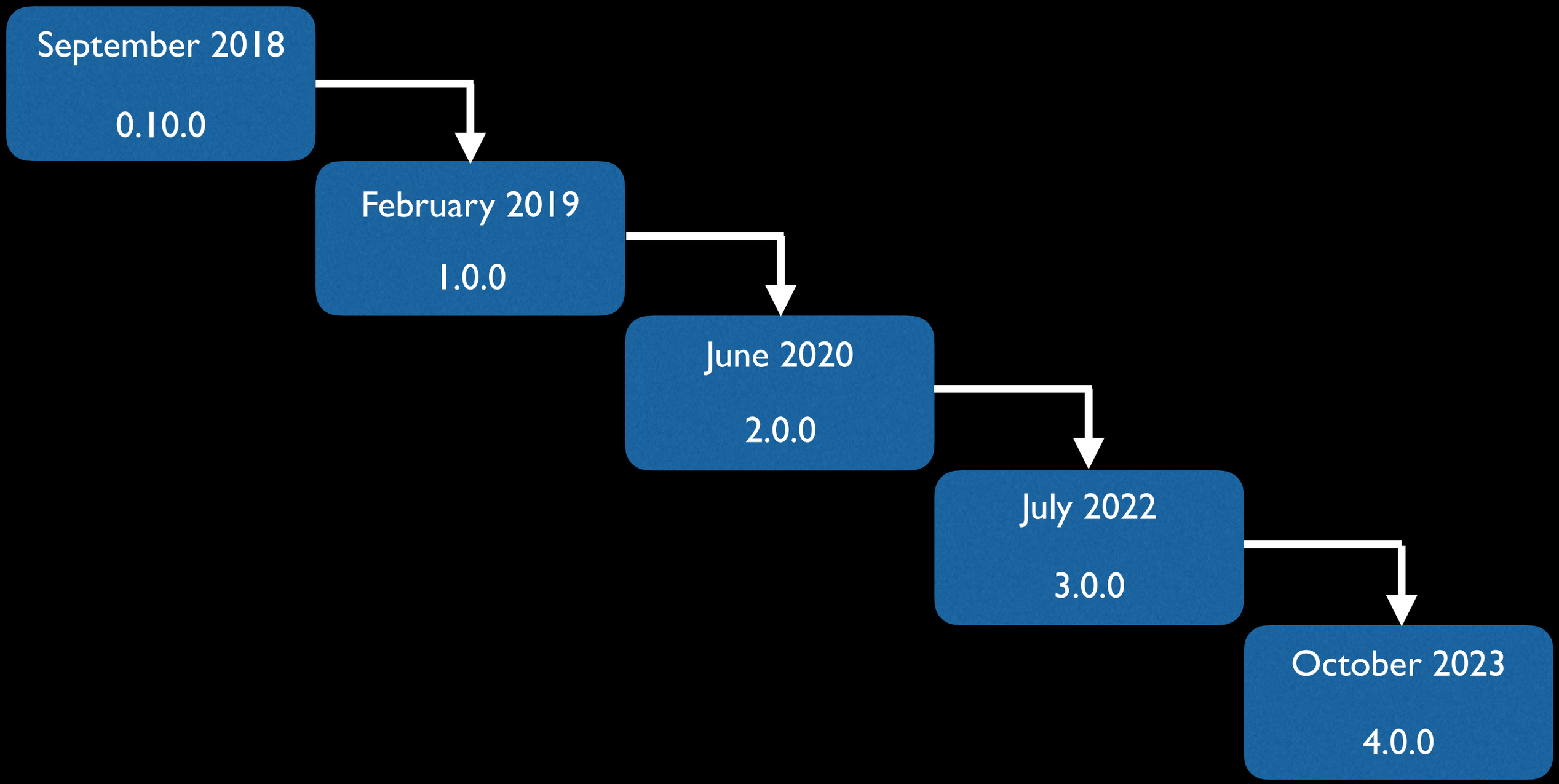
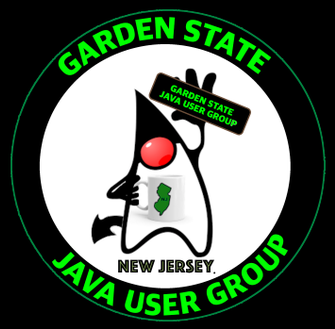
Developing Secure and Robust Java Applications with Helidon

Michael P. Redlich | @mpredli

Evolution of Helidon



Hi, I'm Mike



InfoQ

Lead Java Queue
News Editor

payara®

Contract Developer Advocate
Technical Writer

Java Champion

Co-Director

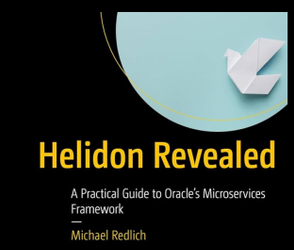
JAKARTA EE

Committer:
Jakarta Data
Jakarta NoSQL

JAKARTA™ EE AMBASSADORS

Vice President

ExxonMobil



Author

Objectives (i)

- Brief History of Helidon
- What is Helidon?
  - Differences between Helidon SE/Helidon MP
- Helidon Architecture
- Helidon and the Java Community

Objectives (ii)

- Getting Started
- Helidon Components
- Demos!

Poll

- I'm a Helidon Expert!
- I've built an application using Helidon
- I'm familiar with Helidon, but haven't used it
- I've heard some things about Helidon
- What the heck is Helidon?

What is Helidon?

What is Helidon? (i)

- A lightweight Java microservices framework
  - introduced by Oracle in September 2018
- A collection of Java libraries designed for creating microservices-based applications
- Originally named J4C (Java for Cloud)
- Helidon == Swallow (Greek)

What is Helidon? (ii)

- Two programming models (flavors)
  - Helidon SE
  - Helidon MP
- Designed to be simple and fast and observable and resilient
- Full support for MicroProfile, GraalVM and Jakarta Persistence specification

What is Helidon? (iii)

- Web servers
  - Reactive
  - Virtual Threads
- Current version: 4.2.0

Reactive Web Server

- Based on Netty
- Versions 1.0 through 3.0 release trains

Virtual Threads Web Server

- Based on JEP 444 (Virtual Threads)
- Versions 4.0+ release trains
- Helidon Níma, the first microservices framework based on virtual threads
- Níma == Thread (Greek)

Helidon SE

Helidon SE

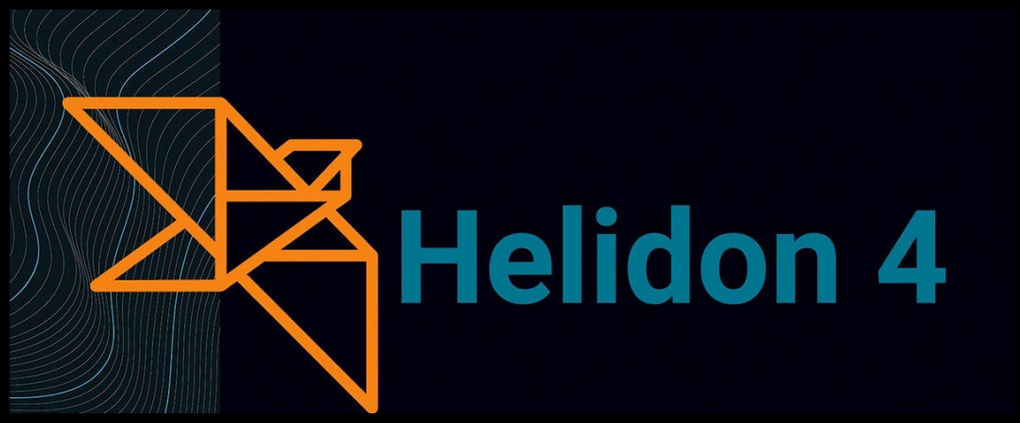
- Provides core functional-style APIs for building microservices-based applications
- An application server is not required as a web server is provided

Functional Style

- Functional style APIs
- Reactive and non-blocking (versions 1.0-3.0)
- Virtual threads (version 4.0+)
- Tiny memory footprint
- No annotations
- No dependency injection
- No enterprise Java standards support

```
Routing routing = Routing.builder()
     .get("/hello", (req, res) -> res.send("Hello World"))
     .build();
WebServer.create(routing)
     .start();
```

Helidon MP



Helidon MP

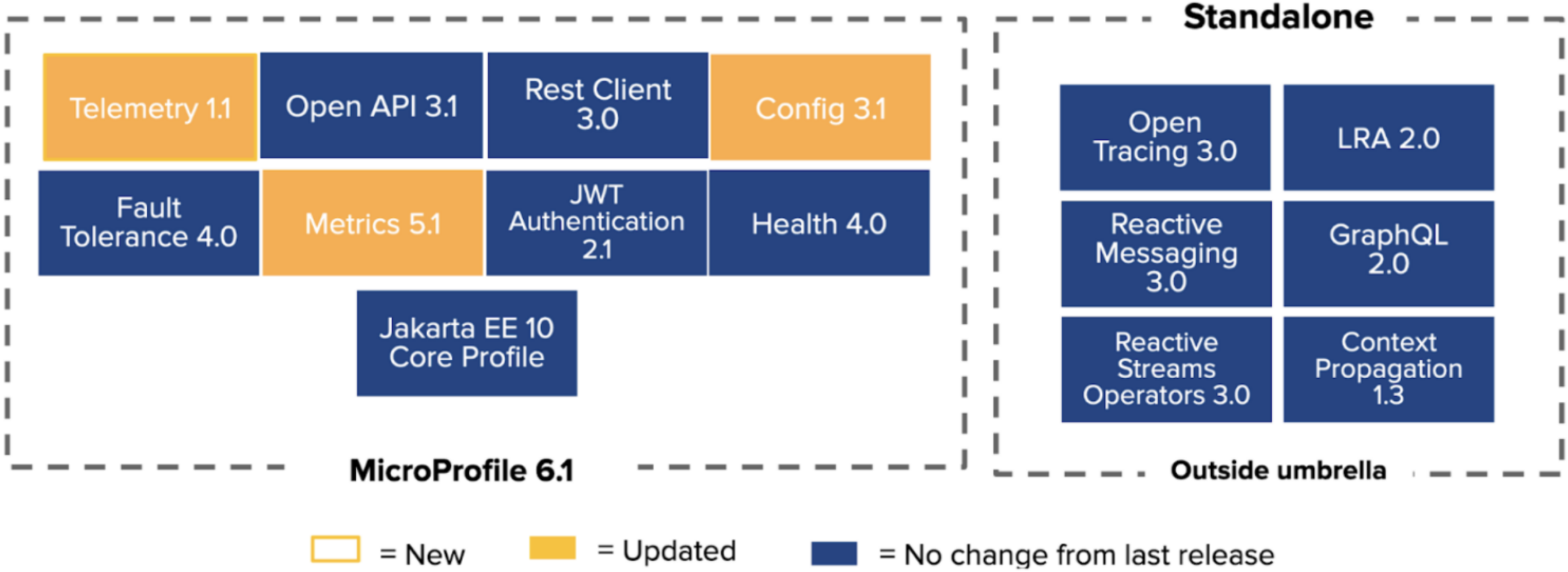
- An implementation of the MicroProfile and Jakarta EE specifications for building microservices-based applications
  - MicroProfile 6.1
  - Jakarta EE 10

Declarative Style

- Declarative style APIs
- Blocking and synchronous
- Small memory footprint
- Heavy use of annotations
- Jakarta Contexts and Dependency Injection (CDI)
- Full support of MicroProfile and partial support of Jakarta EE

```
@Path("hello")
@ApplicationScoped
public class HelloWorld {
    @GET
    public String hello() {
    return "Hello World";
    }
}
```

MicroProfile 6.1



Released October 2023

Helidon Architecture

Helidon Architecture (versions 1.x through 3.x)

Helidon SE

- WebServer
- WebClient
- Config
- Security

Helidon MP

- Metrics
- Health
- Fault Tolerance
- CDI

Netty
(reactive web server)

Helidon Architecture (versions 4.x and beyond)

Helidon SE

- WebServer
- WebClient
- Config
- Security

Helidon MP

- Metrics
- Health
- Fault Tolerance
- CDI

Helidon Níma
(virtual web server)

Helidon and the Java Community

helidon.io

GraalVM

JAKARTA EE

MICRONAUT®

Kotlin

EclipseStore
MicroStream

LOG4J

IJ

GARDEN STATE JAVA USER GROUP
NEW JERSEY

InfoQ

payara®

Getting Started

Ways to Get Started

Quickstarts

Helidon CLI

Project Starter

Quickstarts

```
$ mvn -U archetype:generate -DinteractiveMode=false \
    -DarchetypeGroupId=io.helidon.archetypes \
    -DarchetypeArtifactId=helidon-quickstart-se \
    -DarchetypeVersion=4.2.0 \
    -DgroupId=io.helidon.examples \
    -DartifactId=helidon-quickstart-se \
    -Dpackage=io.helidon.examples.quickstart.se

$ mvn -U archetype:generate -DinteractiveMode=false \
    -DarchetypeGroupId=io.helidon.archetypes \
    -DarchetypeArtifactId=helidon-quickstart-mp \
    -DarchetypeVersion=4.2.0 \
    -DgroupId=io.helidon.examples \
    -DartifactId=helidon-quickstart-mp \
    -Dpackage=io.helidon.examples.quickstart.mp
```

How to Get Helidon CLI

```
$ curl -L -O https://helidon.io/cli/latest/darwin/helidon
chmod +x ./helidon
sudo mv ./helidon /usr/local/bin/
```

```
$ curl -L -O https://helidon.io/cli/latest/linux/helidon
chmod +x ./helidon
sudo mv ./helidon /usr/local/bin/
```

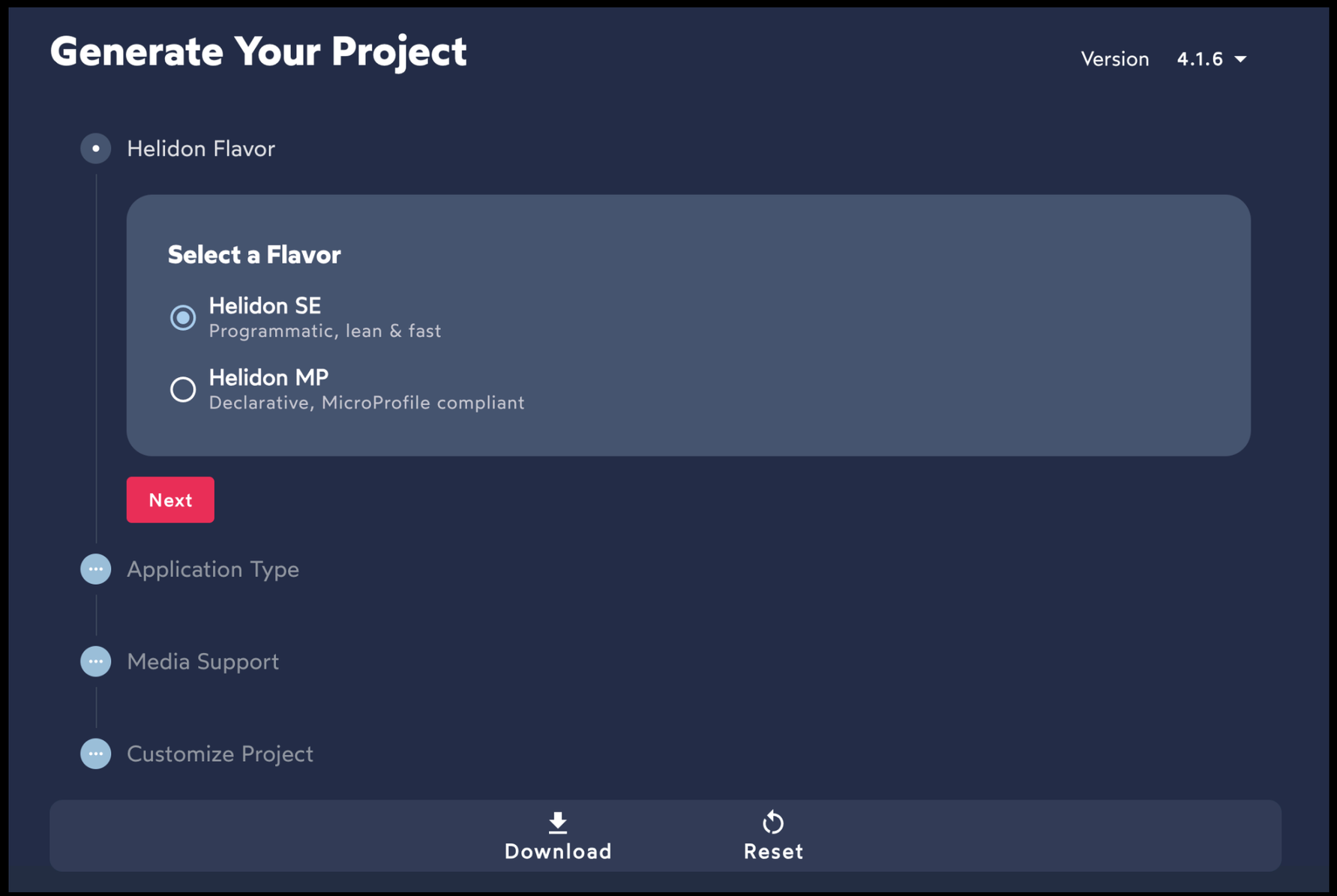
Linux

```
C:> PowerShell -Command Invoke-WebRequest -Uri "https://
helidon.io/cli/latest/windows/helidon.exe" -OutFile "C:
\Windows\system32\helidon.exe"
```

InfoQ

payara®

Project Starter




DEMO

Helidon Web Server

Helidon Web Server

- An immutably configured web server based on virtual threads
  - Previously an asynchronous and reactive API that ran on top of Netty
- **`WebServer`** interface

Helidon Web Client

Helidon Web Client

- An HTTP client for Helidon SE
  - Previously an asynchronous and reactive API
- `WebClient` interface


DEMO

Helidon Config

Helidon Config

- Provides a comprehensive and flexible configuration system to load and process configuration data from various sources into a configuration object that may be used in a Helidon application
- **`Config`** interface


DEMO

Helidon Security

Helidon Security

- Provides authentication, authorization, audit and outbound security
  - Bootstraps security and integrate it with other frameworks
- `Security` interface

Authentication Mechanisms

- HTTP Basic Authentication
- HTTP Digest Authentication
- HTTP Signatures
- Attribute Based Access Control (ABAC) Authorization
- JWT Provider
- Header Assertion
- Google Login Authentication
- OpenID Connect
- IDCS Role Mapping


DEMO

Helidon Metrics

Helidon Metrics

- Provides a unified way for MicroProfile servers to export monitoring data to management agents and also a unified Java API, that all developers can use to expose telemetry data
- A compatible implementation of the MicroProfile Metrics 5.1 specification

Metrics Scopes

base	application	vendor

Metrics Annotations

`@Counted`	`@Gauge`	`@Timed`

@Counted

```
@Counted(name = "countedName")
public String countedName(String name) {
    return "I am a counter named: " + name;
    }

@Counted
public class CounterBean {
    public void countMethod1() {}
    public void countMethod2() {}
    }
```

@Gauge

```
@Gauge(name = "gaugeName")
public int getQueueSize() {
    return queue.size;
}
```

@Timed

```
@Timed(name = "timedName")
public String timedName(String name) {
    return "I am a timer named: " + name;

@Timed
public class TimedBean {
    public void timedMethod1() {}
    public void timedMethod2() {}
    }
```

Metric Registry

- Collects and stores your registered metrics (and related metadata) for a specified scope
- One metric registry for each scope


DEMO

Helidon Fault Tolerance

Helidon Fault Tolerance

- Improves application robustness with the ability to conveniently handle error conditions, or faults, that may occur in real-world applications
- A compatible implementation of the MicroProfile Fault Tolerance 4.0 specification

Fault Tolerance Annotations

`@Retry`	`@Timeout`	`@CircutBreaker`
`@Bulkhead`	`@Falback`	`@Asynchronous`

@Retry

```
@Retry(
    maxRetries=3,
    delay=0,
    delayUnit=ChronoUnit.MILLIS,
    maxDuration=180000,
    durationUnit=ChronoUnit.MILLIS,
    jitter=200,
    jitterDelayUnit=ChronoUnit.MILLIS,
    retryOn={Exception.class},
    abortOn={}
)
```

@Timeout

```
@Timeout(
    value=1000,
    unit=ChronoUnit.MILLIS
)
```

@CircuitBreaker

```
@CircuitBreaker(
    failOn={Throwable.class},
    skipOn={},
    delay=5000,
    delayUnit=ChronoUnit.MILLIS,
    requestVolumeThreshold=20,
    failureRatio=.50,
    successThreshold=1
    )
```

@Bulkhead

```
@Bulkhead(
    value=10,
    waitingTaskQueue=10
)
```

@Fallback

```
@Fallback(
    value=DEFAULT.class,
    fallbackMethod="",
    applyOn={Throwable.class},
    skipOn={}
)
```


DEMO

Helidon Health Checks

Helidon Health Checks

- Defines a runtime mechanism to determine if a computing node needs to be discarded and eventually replaced by a healthy instance
- A compatible implementation of the MicroProfile Health 4.0 specification

Health Checks Types

Liveness	Readiness	Startup
`/health/live`	`/health/ready`	`/health/started`

Health Checks Annotations

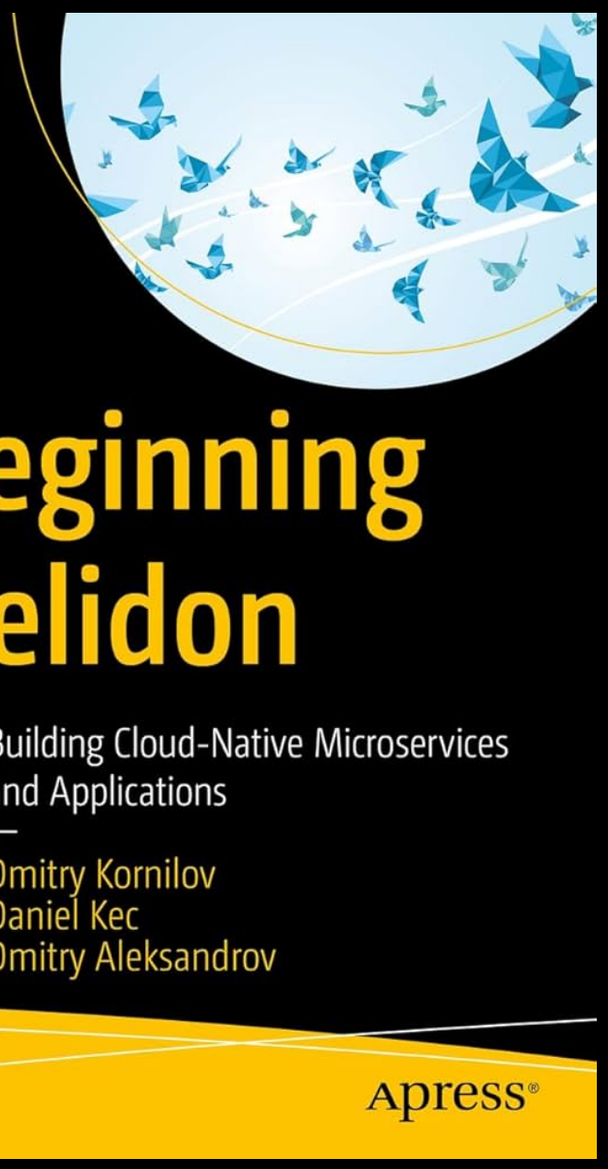
`@Liveness`

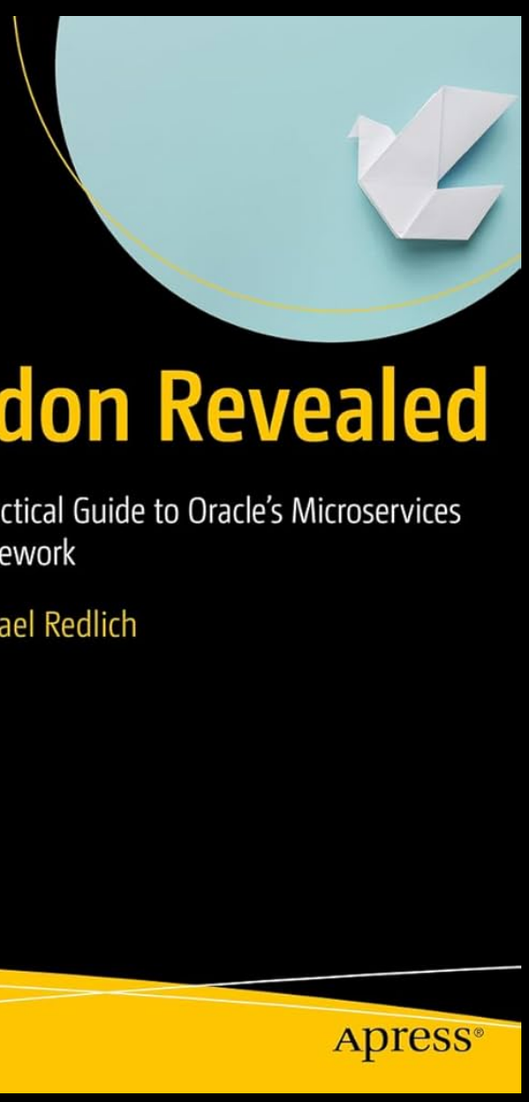
`@Readiness`

`@Startup`


DEMO

Resources


Beginning
Helidon
Building Cloud-Native Microservices
and Applications
—
Dmitry Kornilov
Daniel Kec
Dmitry Aleksandrov
Apress®


Helidon Revealed
A Practical Guide to Oracle's Microservices
Framework
—
Michael Redlich
Apress®

Want to Get Involved?

- Contribute to open source!
  - Share
  - Embrace
  - Engage
  - Develop

Tanja Obradovic
Jakarta EE Program Manager at
Eclipse Foundation

- Share your own Jakarta EE experiences!

Acknowledgements

- Rowena Redlich
- Barry Burd
- Dmitry Kornilov

Contact Info

mike@redlich.net

@mpredli

redlich.net

redlich.net/portfolio

github.com/mpredli01

Thanks!