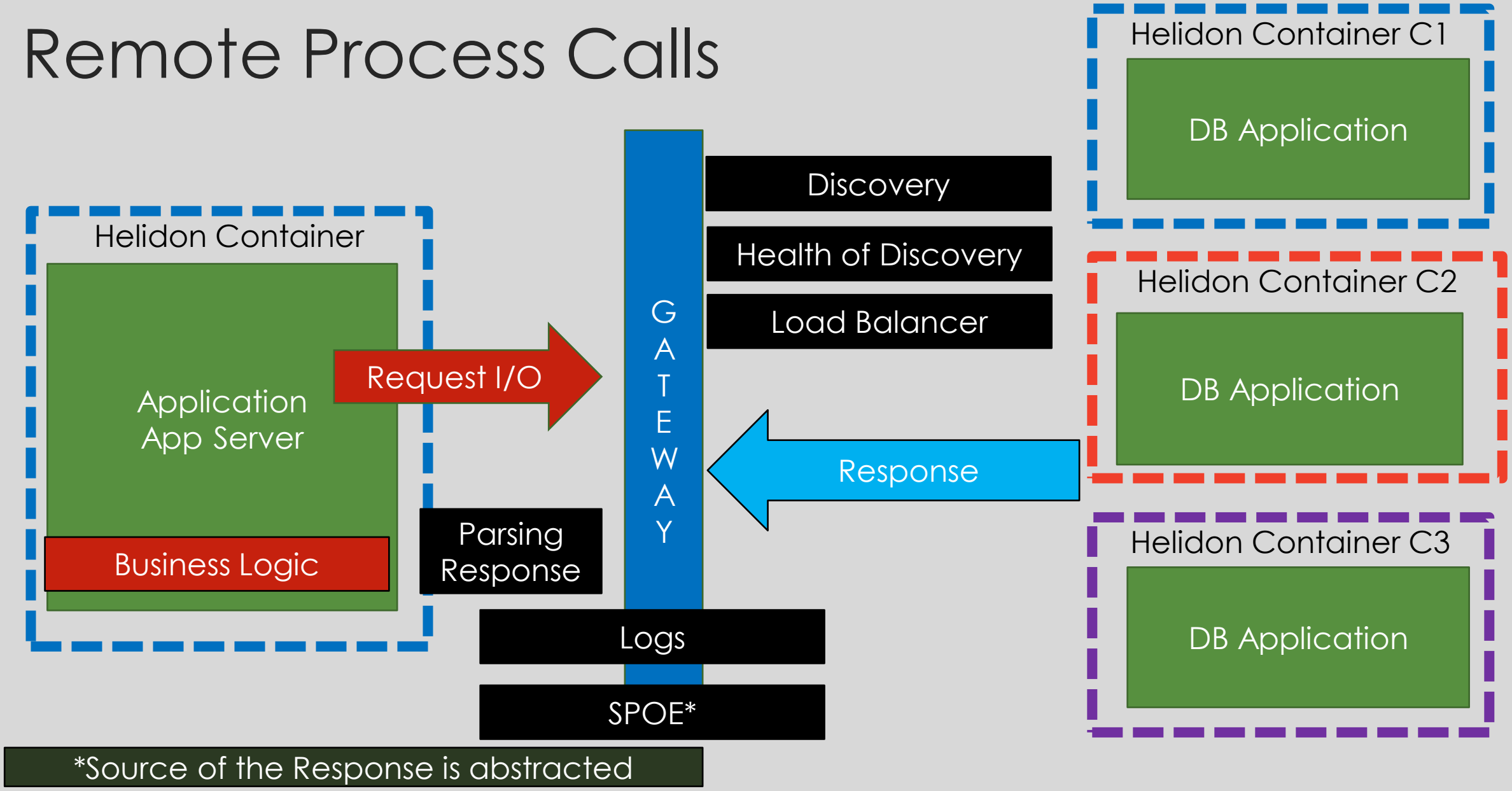

MICROPROFILE
WITH HELIDON
dhananjayan

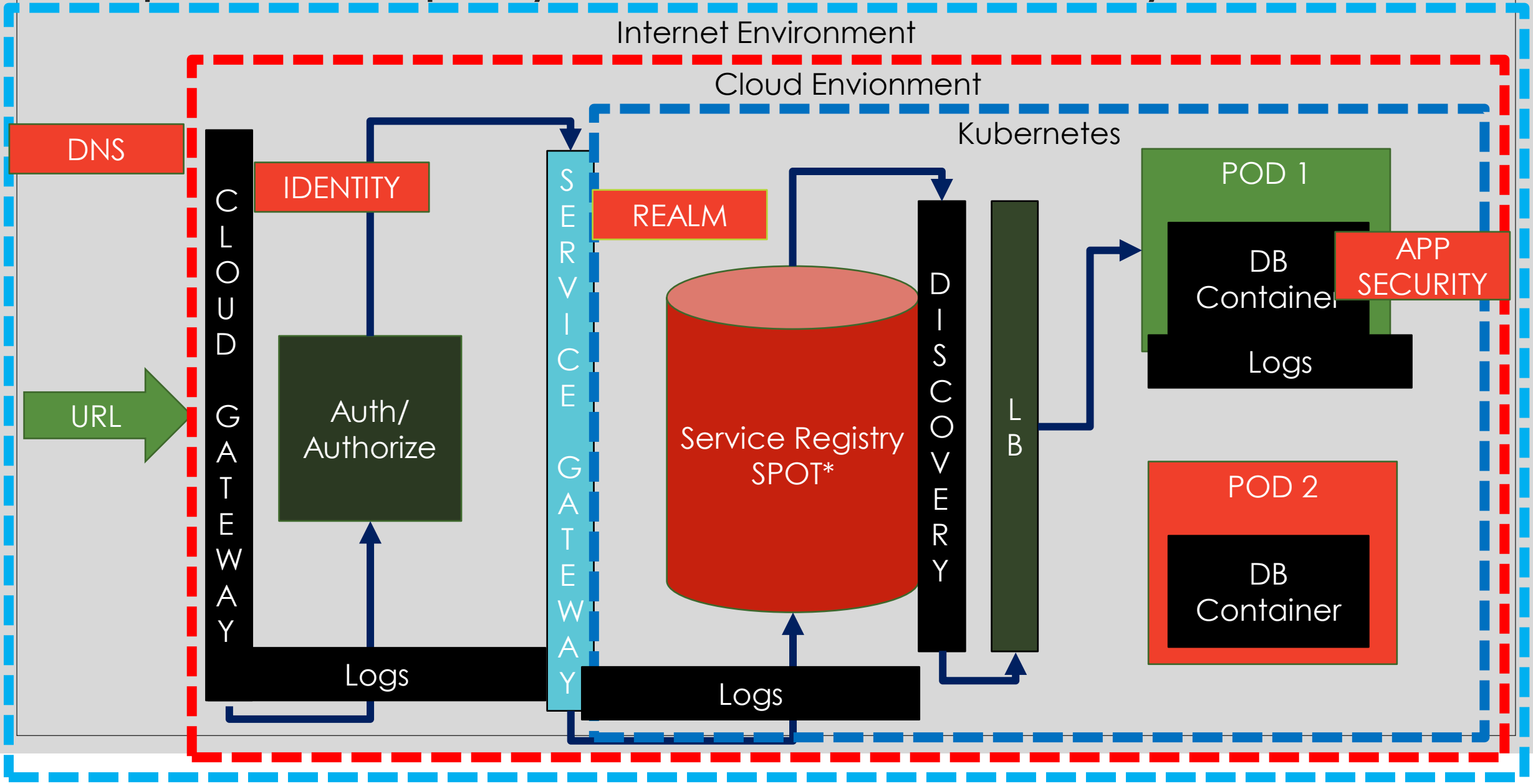
Day 3

- RESTAPI with CDI
- Server Side Discovery
- Implement REST API (Nested API – use case)
- OAS – Open API Specification
- Filter EndPoints
- Implement Swagger in Open API (UI)
- Interceptors
- Health Checks, (Probes) , Metrics (Prometheus)

Remote Process Calls



Scope of Deployments and Security



Types of RPC Calls

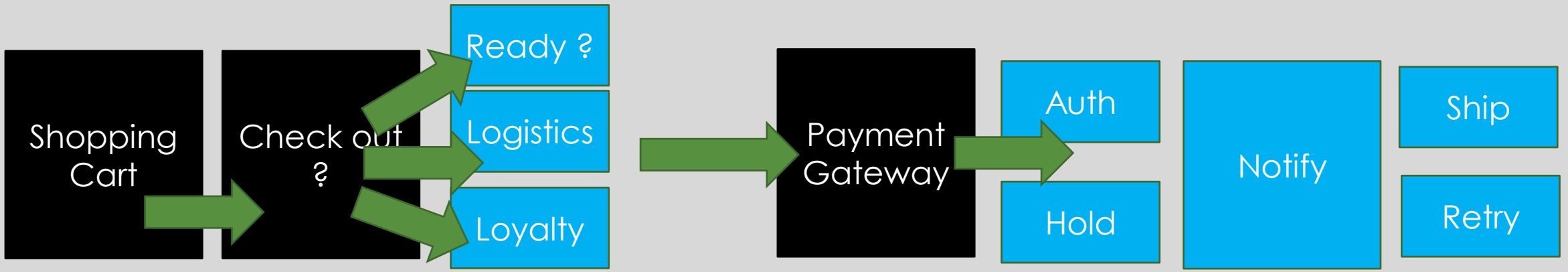
Sync+ Async
Hybrid

Light weight
HTTP
Blocked
1:1
Synchronized (REST API, Thrift)

Broadcast
1:#
ASYNCHRONOUS
Pub – Sub
APMQ , Kafka, .Active MQ, Rabbit MQ

Security - External Requests

Internal – Messages



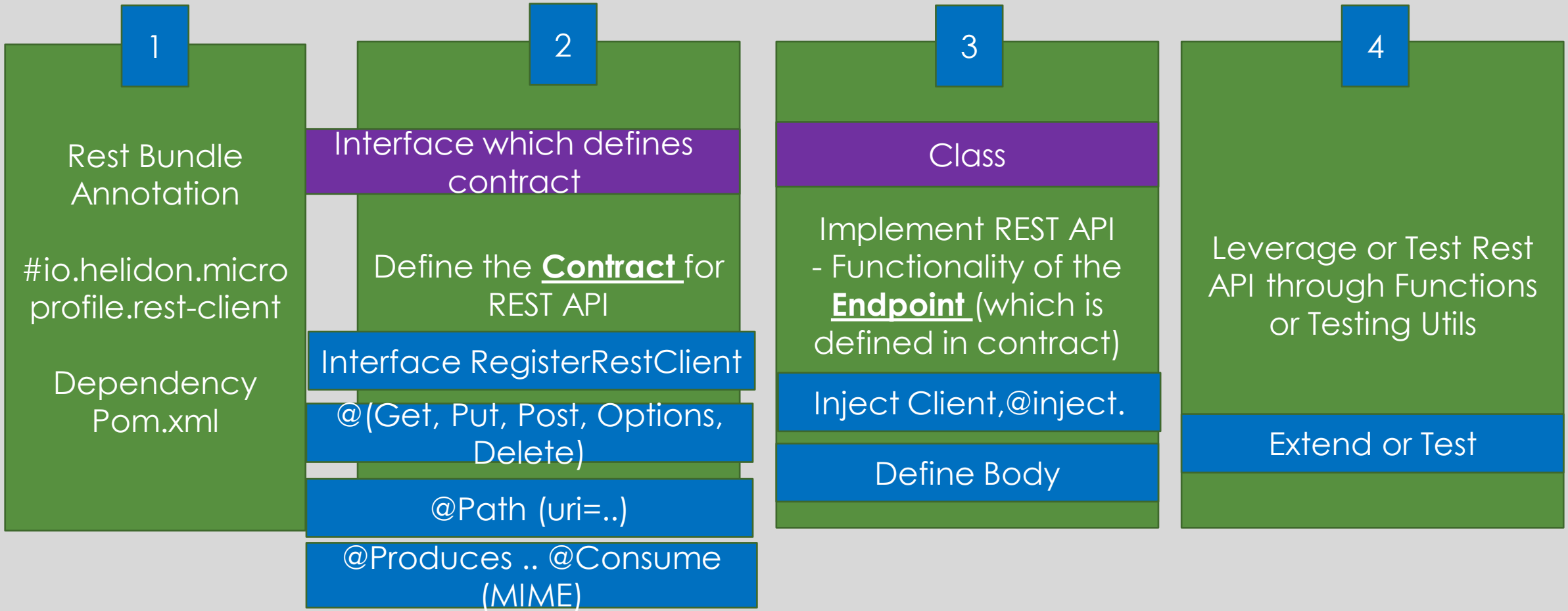
REST API

Define the **Contract** for REST API

Implement REST API - Functionality of the **Endpoint** (which is defined in contract)

Leverage or Test Rest API through Functions or Testing Utils

Implement REST API in Helidon MP



Implement Slide 68

Defined Contract – Step 2 of Slide 68

- http://localhost:8080/greet/outbound/name
  - Invoke default /greet (getDefaultMessage() + name as parameter.

- RestClient Interface

```
@RegisterRestClient(uri=.
@ GET
JsonObject getDefaultMessage()

@Path("/{name}")
@GET
 JSonObject getMessage (@PathParameter("{name}")
```

1

2

Implement Slide 68 (2)

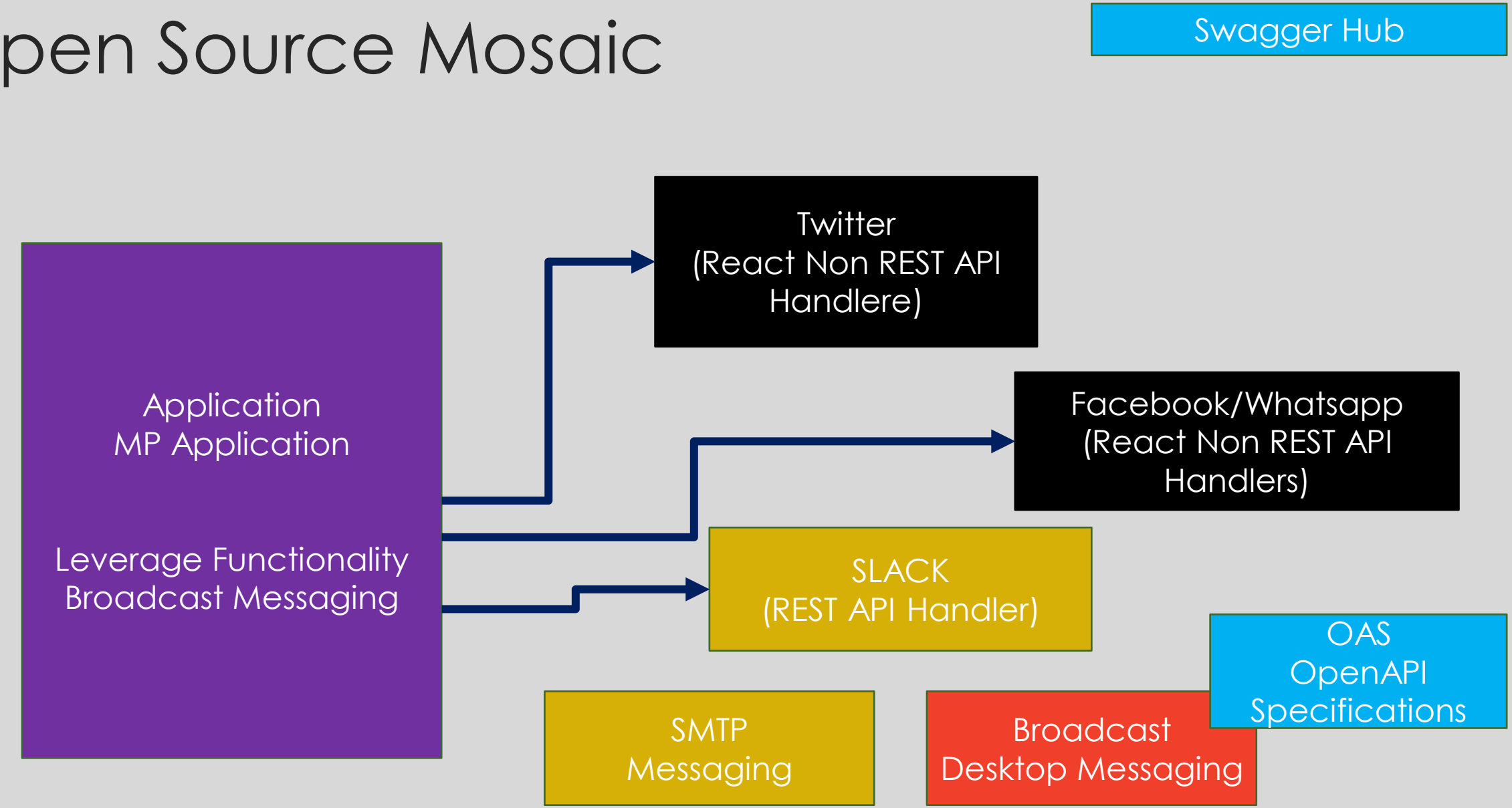
```
@Inject
@RestClient
GreetRestClient restclient;
```

Implementing Functionality of Contract – Step 3 of Slide 68

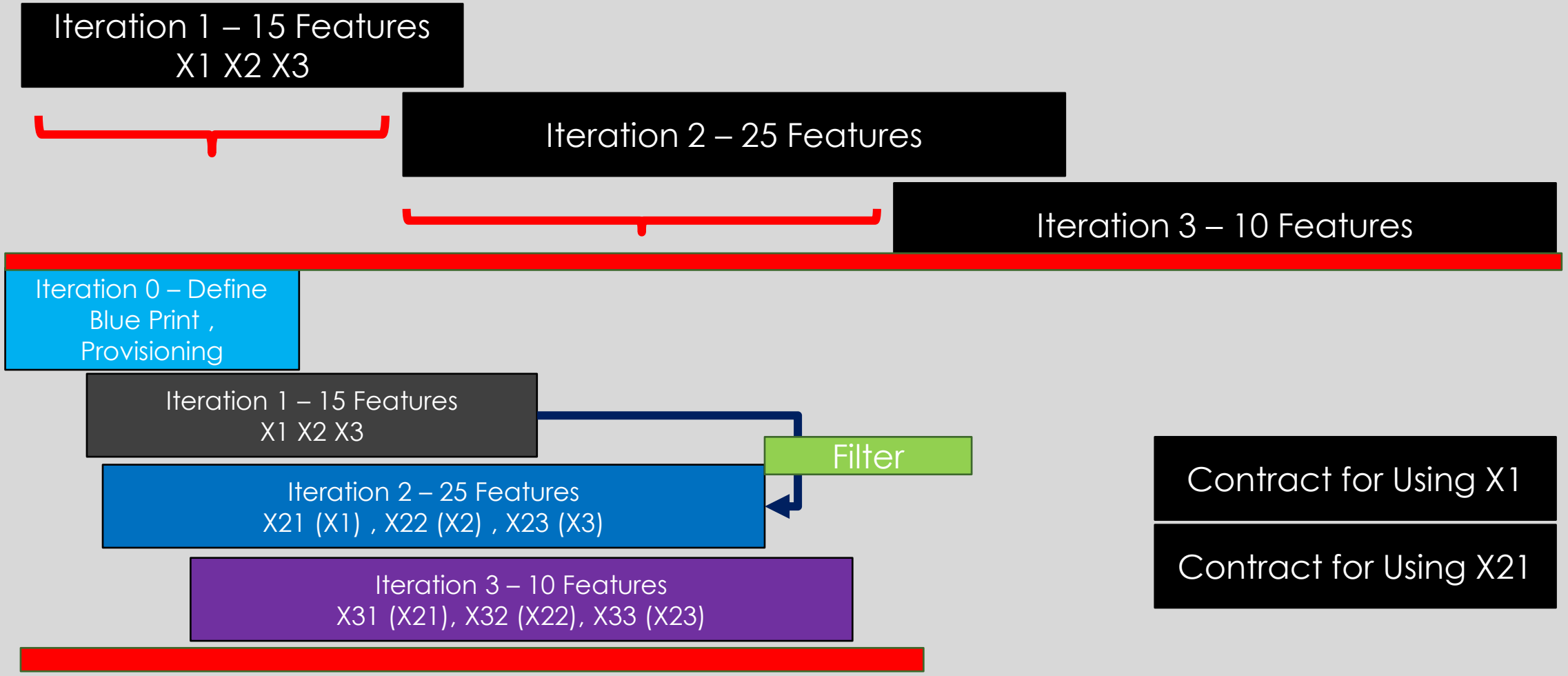
http://localhost:8080/greet/outbound/name

```
@GET
@Path("/outbound/{name}")
public jsonObject outbound (@PathParam("{name"} String x) {
 return restclient.getMessage(x);
 //getMessage is already defined }
```

Open Source Mosaic



Parallel Development



OAS3 Objectives

Support from helidon MP

Parallel Development

Integrate with Open Source

Generalize Standard

Non API to API (Further Incremental)

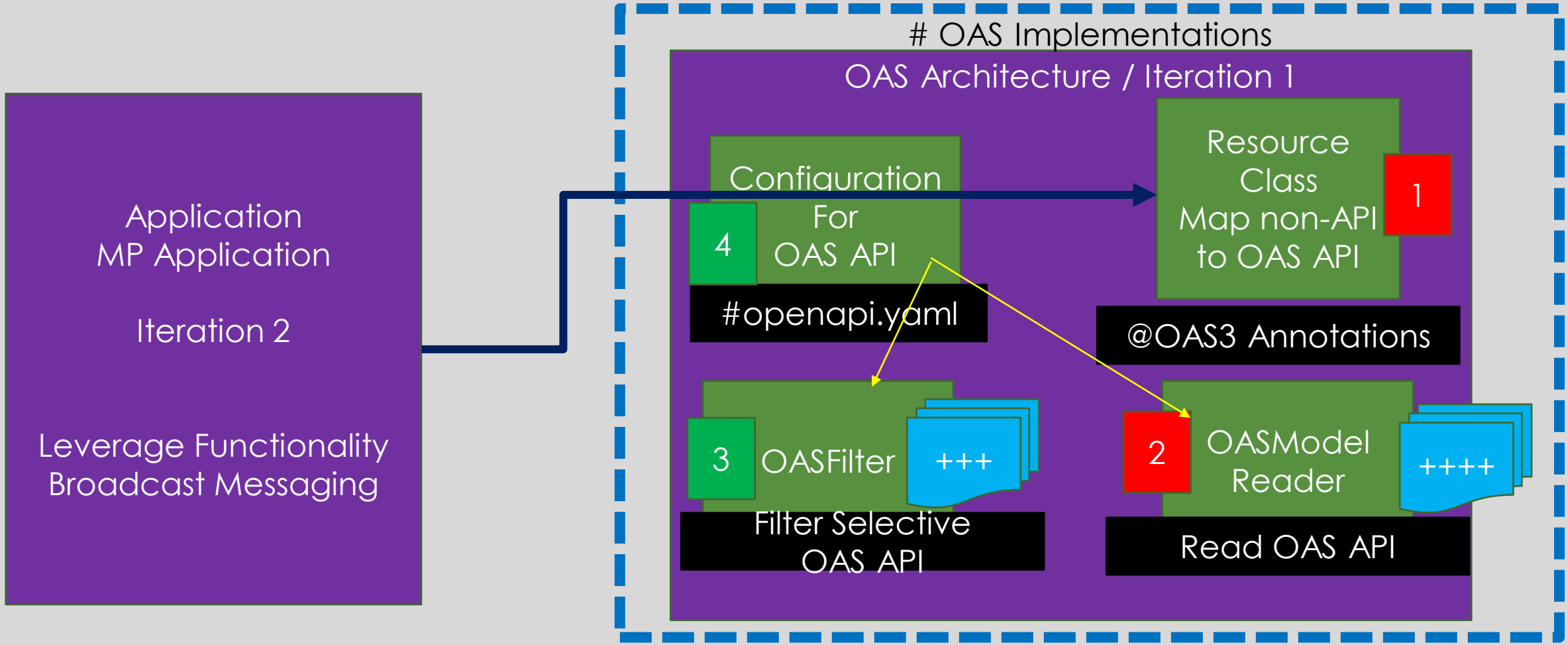
OAS Contracting

Reader, Filter

UI Notation for Test

@OAS3 Annotations

OAS Architecture



Support Systems for Slide 74

- #microprofile.openapi.model.reader
- #mp.openapi.filter
- #openapi.scan.packages
- #openapi.scan.classes
- #openapi.scan.servers
- #mp.openapi.extensions (vendor extensions)

OAS3 Annotations

1

- @Operation (summary="……read files from server")
- @APIResponse (despn="…", @Content(media_type=application/json")
- @APIResponse(code=400,descpn=)
- @Parameters
- @CallBack
- @APIResponses (@APIResponse(…))

Dependencies

- Eclipse Libraries Globally
  - Org.eclipse.microprofile.openapi
    - Microprofile.openapi-api
  - Org.eclipse.microrprofile.openapi-ui
- Helidon MP Library
  - Io.helidon.microrprofile.openapi

OAS-Model-Reader

- Traverse Programattically add Elements (/endpoints) to API
- Allows Developer to bootstrap open API (Big tree graph)

2

OAS.Filter

- Logic of Certain End Points to added to graph
- Implement – Interface – Override equals or compareTo method

3

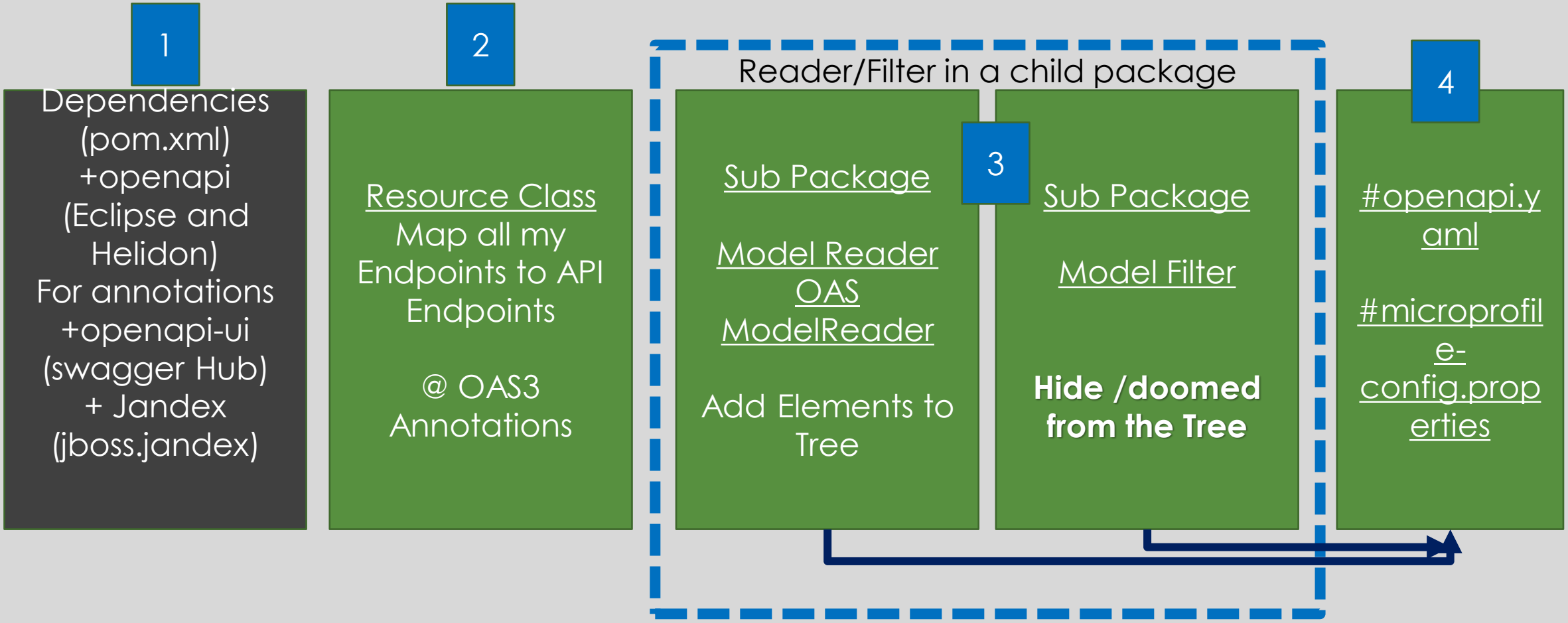
Jandex Plugin

- Performance for JVM /runtime can be expedited
- Index showing information which attributes, methods, annotation – classes
- CDI Discovery Beans – Process and Search annotation faster

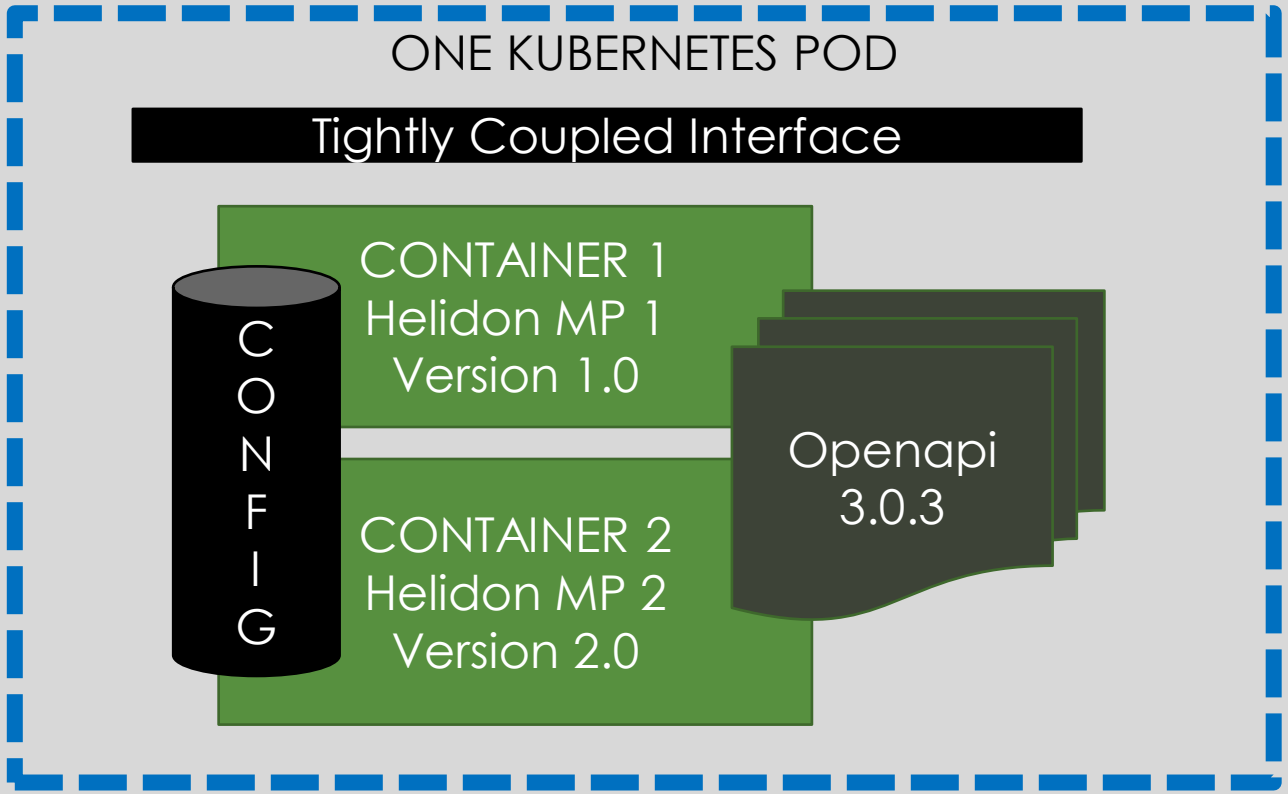
Use case 4 : Implement Open API

- Define Two API Contracts
  - /test/newpath
  - /doomed
- Model Reader for Traversing all /endpoints
- Model Filter to filter for /doomed ➔ Not be in /openapi documentation
- /openapi-ui (UI for openAPI documentation)

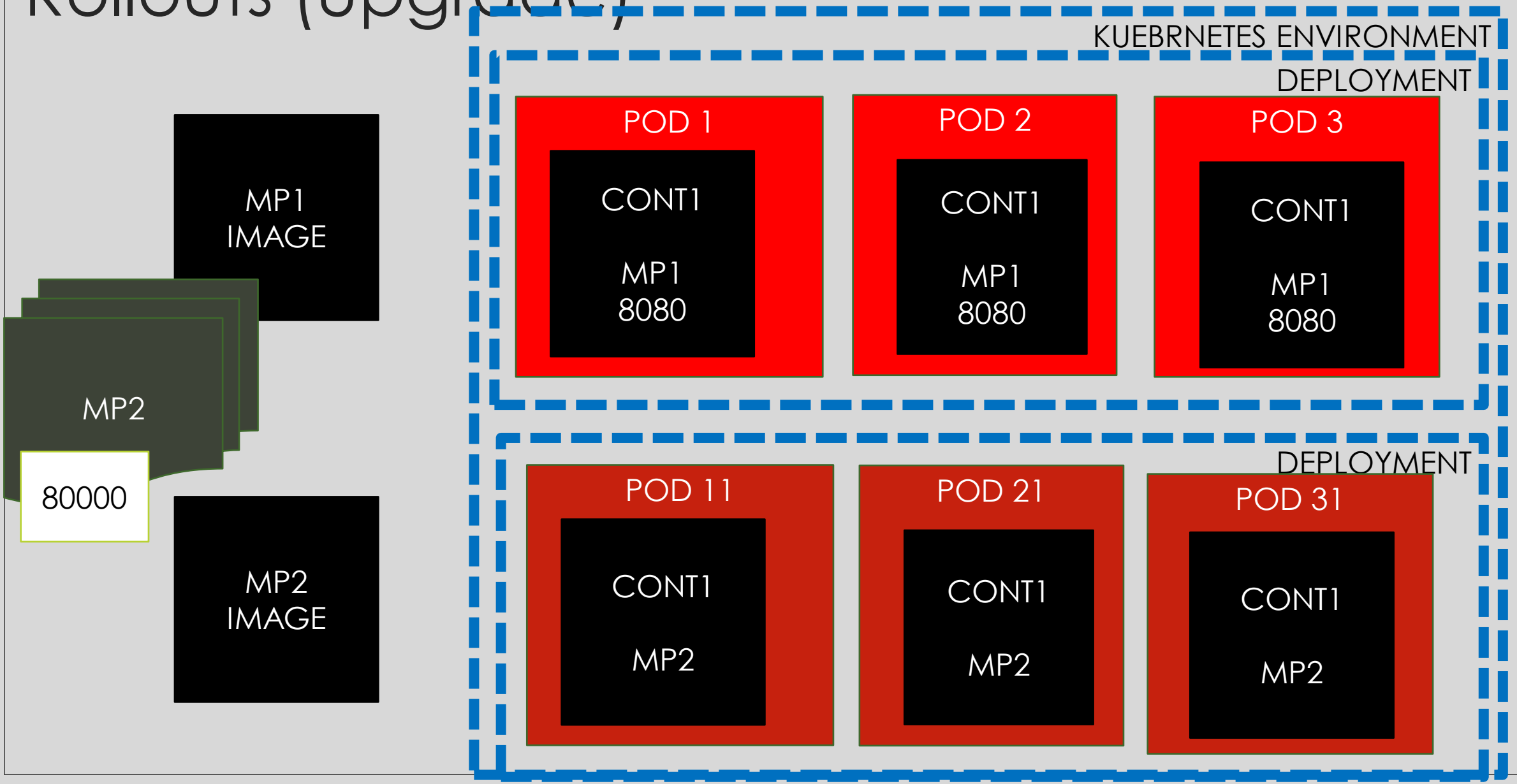
Implementation of Slide 80 (usecase)



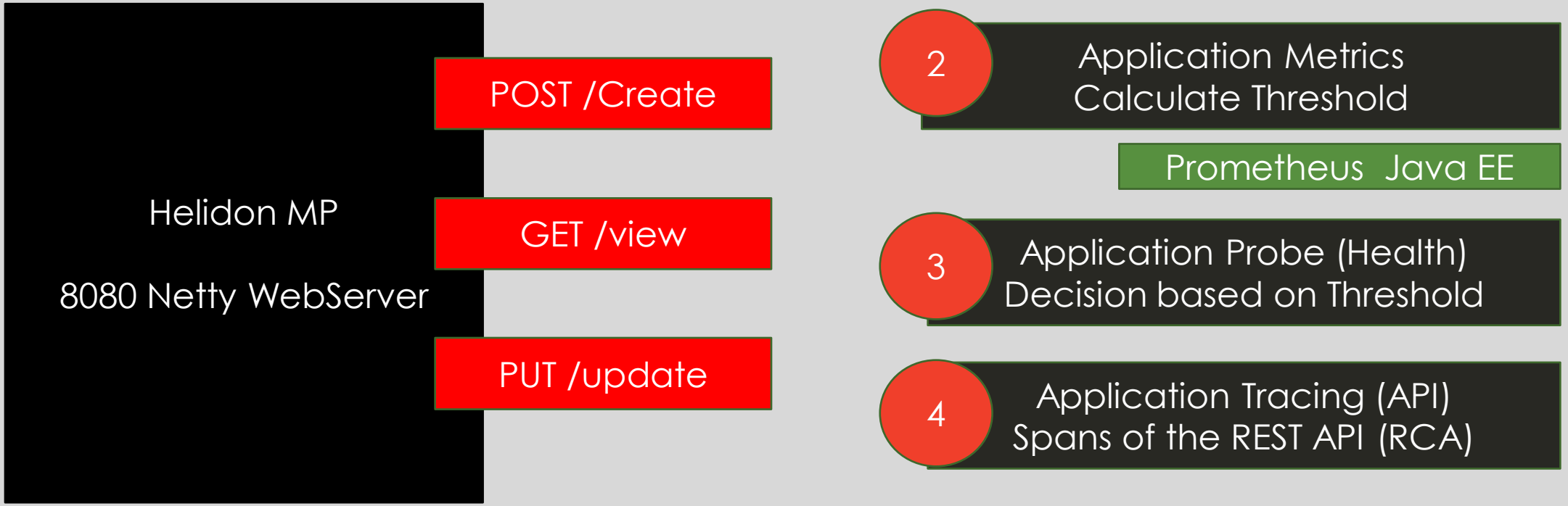
Dependency Services or Attached Services



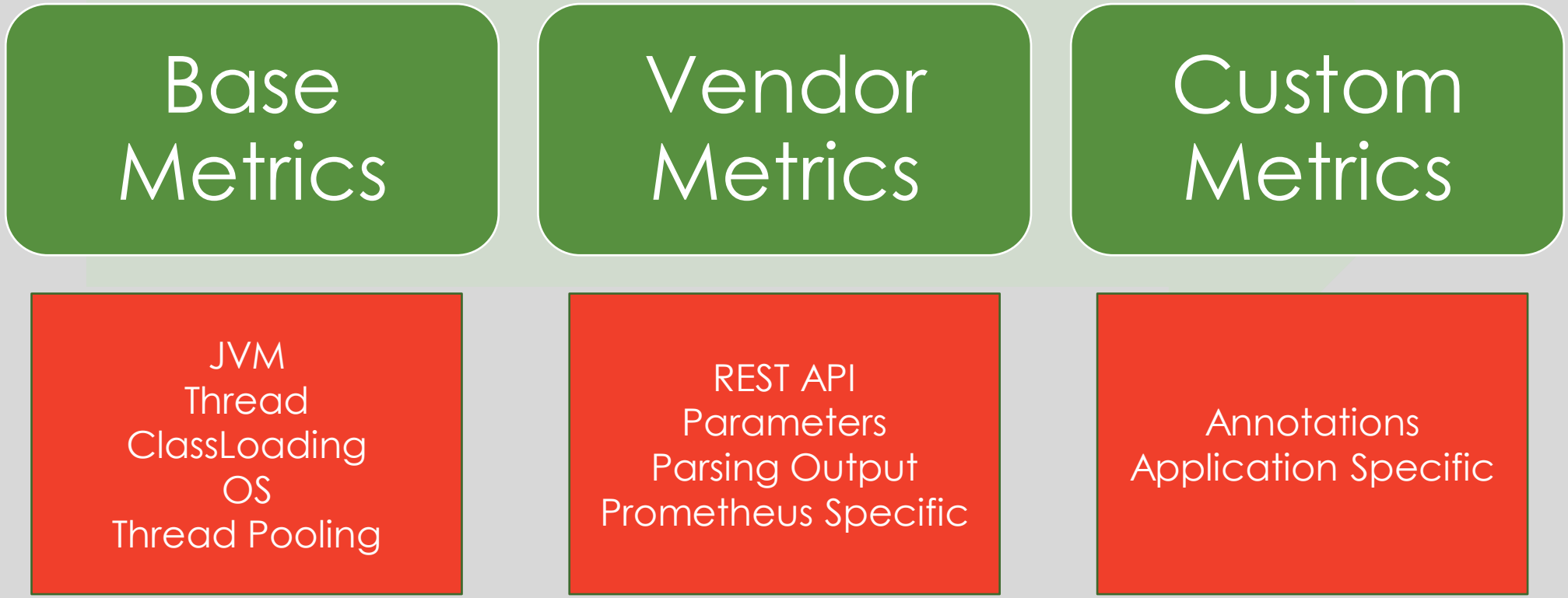
Rollouts (Upgrade)



Application **Interceptors (Shared Model)**



Metrics


Base Metrics
Vendor Metrics
Custom Metrics
JVM
Thread
ClassLoading
OS
Thread Pooling
REST API
Parameters
Parsing Output
Prometheus Specific
Annotations
Application Specific

Metrics View


```
//GreetResource.class

@Counted(name="greet-count")
@Timed (name="timer-metrics-greet",absolute=true)
@ GET
public jsonobject getmessage()
{
}
```

Implementing Metrics (Slide 87)

Helidon

+pom. Xml
(helidon-metrics)*
Helidon 2.4 inclusivity metrics++

//Greet Resource class

Definition of End Points

+Counted
+Timed
+Metered

Import Packages

Build Applications

Resilience.. **Shared Model**

- Ability to Self Heal
- Self –repair
- Monitor core Capabilities below metrics count
  - Probes (Self Probes)

Health Probes

Am I Living ?

LIVENESS PROBE ?

#Restart the Application

Am I ready for Traffic ?

READINESS PROBE ?

No Traffic to Application

Use case 6 : Implementing Probes

Liveness Activity

#mylivenessprobe
---How much time in milliseconds ()

-- up () /down

Readiness Activity

#myreadiness probe
10 seconds to be Ready

+ > **10 seconds – Ready for traffic**

Supporting Systems

Implement **Health Check Interface**

Call() → HealthCheckResponse

OnStartup() → void

Class Annotation

+@Liveness / Liveness Probe

+@ Readiness / Readiness Probe

Libraries

Org.eclipse.microrprofile.health.*
Org.eclipse.microrprofile.health.Liveness
Org.eclipse.microrprofile.health.Readiness

Implementing Health Probes

Helidon

+pom. Xml
Helidon-health
2.4 – Inclusive of
Health Probes

Liveness Class
implements Health
Check
@Liveness
Override call ()

Readiness Class
implements Health
Check

@Readiness
onStartup()
//Record time
Override call
Delta of 10 seconds

Build Applications