MICROPROFILE WITH HELIDON

dhananjayan

Pre-requisite

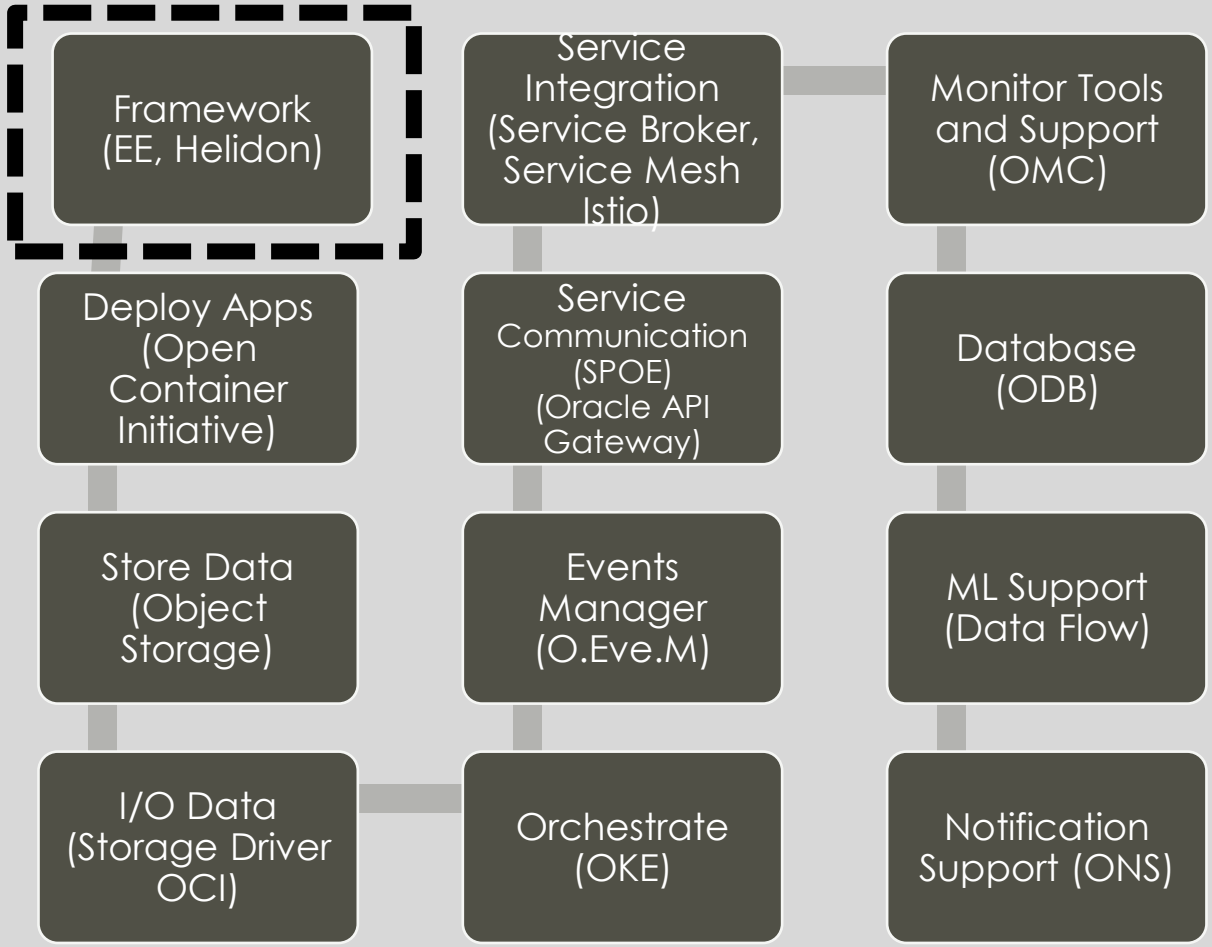
- Name, Role and Objective
- Working Knowledge of Java EE
- REST / Services Architecture - Developing /Deploying /Debugging
- Monitoring Tools – Awareness
- Awareness to Cloud Computing
- Awareness to Container Architecture

Day 2

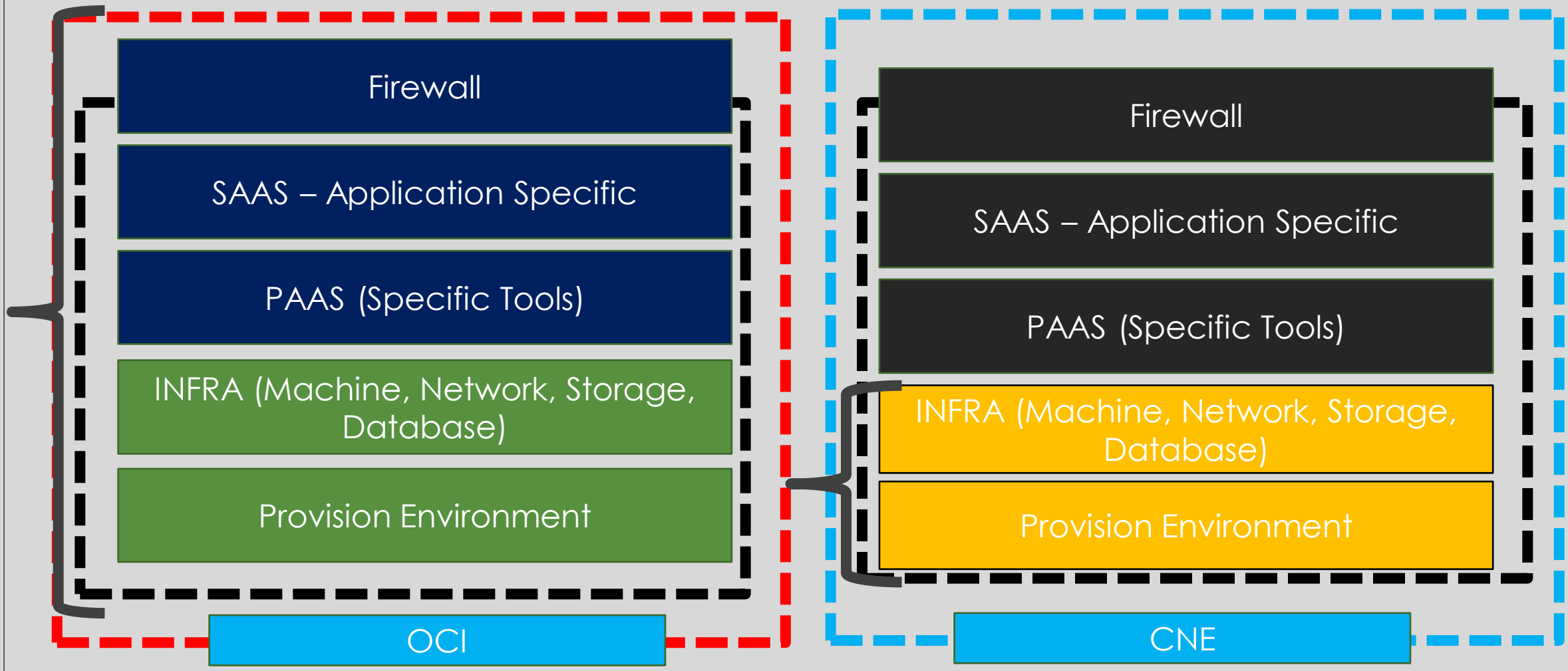
- Microservices with OCI
- Polygot Approach
- Scale Cube
- Maturity Journey
- Helidon Architecture
- Helidon Approach
- Helidon Components – CDI , JSON, Healthchecks (Microprofile)
- Configure Server /Environments

Yield Microservices

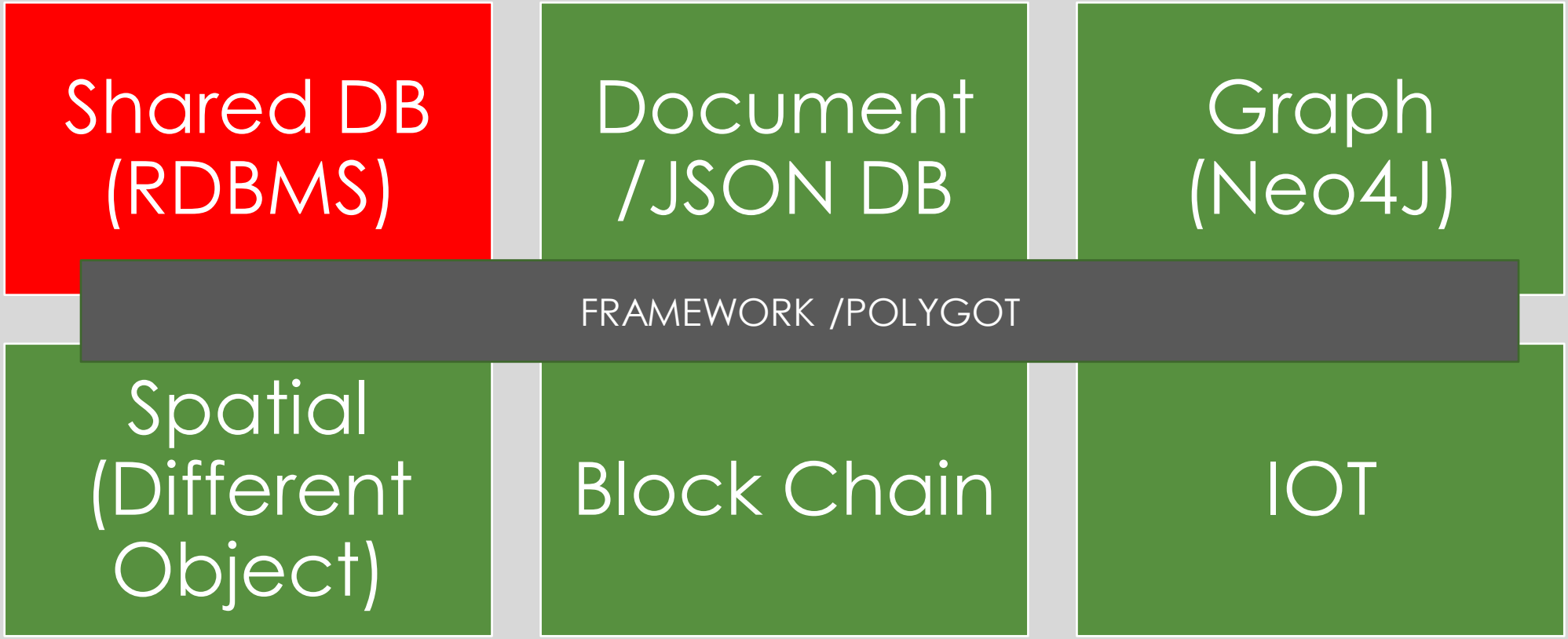
Integration with OCI (MSA)



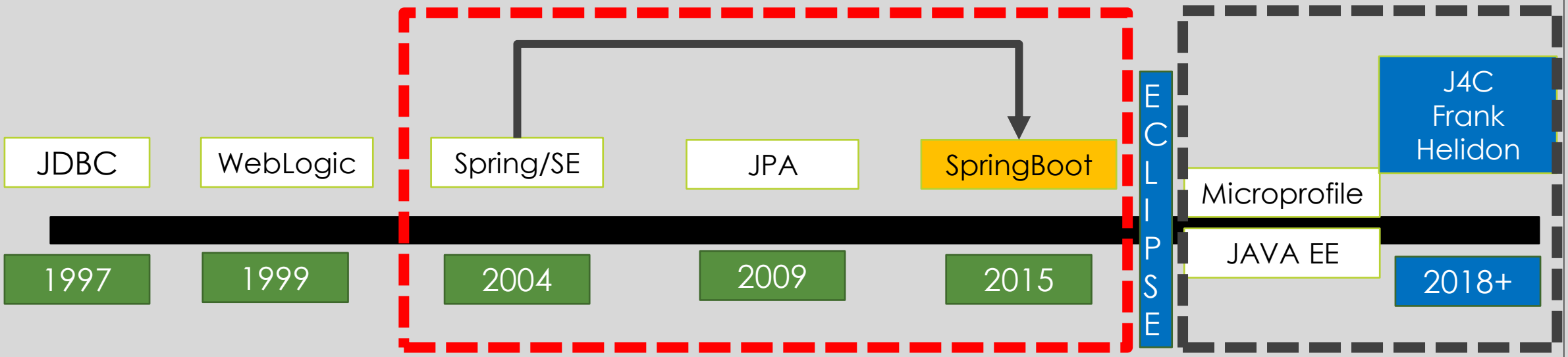
OCI vs CNE


Firewall
SAAS – Application Specific
PAAS (Specific Tools)
INFRA (Machine, Network, Storage, Database)
Provision Environment
OCI
Firewall
SAAS – Application Specific
PAAS (Specific Tools)
INFRA (Machine, Network, Storage, Database)
Provision Environment
CNE

Polygot Approach



Helidon


JDBC
WebLogic
Spring/SE
JPA
SpringBoot
ECLIPSE
Microprofile
JAVA EE
J4C
Frank
Helidon
1997
1999
2004
2009
2015
2018+

SE vs MP (properties)

Standard Edition	Micro-profile
Procedural	Declarative Format
Define Milestones → Achieve Execution Plan → Rescue Factor (Exception Handling) → Achieve Outcome	**Define Milestones →Define Outcome** → **Runtime** will create an Internal SE code to reach this outcome
Clear Transparency – Defined API / Methods, Parameters	**Annotations** (CDI) ➔ Outcome OCI (Resource manager)
Portability of Applications /Rewrite	**Portability / Rewrite on Business changes**

Technology Levers

Hybrid

FULL STACK DEVELOPMENT : MONOLOTHIC, MICROSERVICES, LAYERED ARCHITECTURE
Spring , SpringBoot, Procedural (SE) , OpenJDK

CNA

CLOUD NATIVE APPLICATIONS : MICROSERVICES
Declarative , Fast Track : Helidon MP, Open Liberty

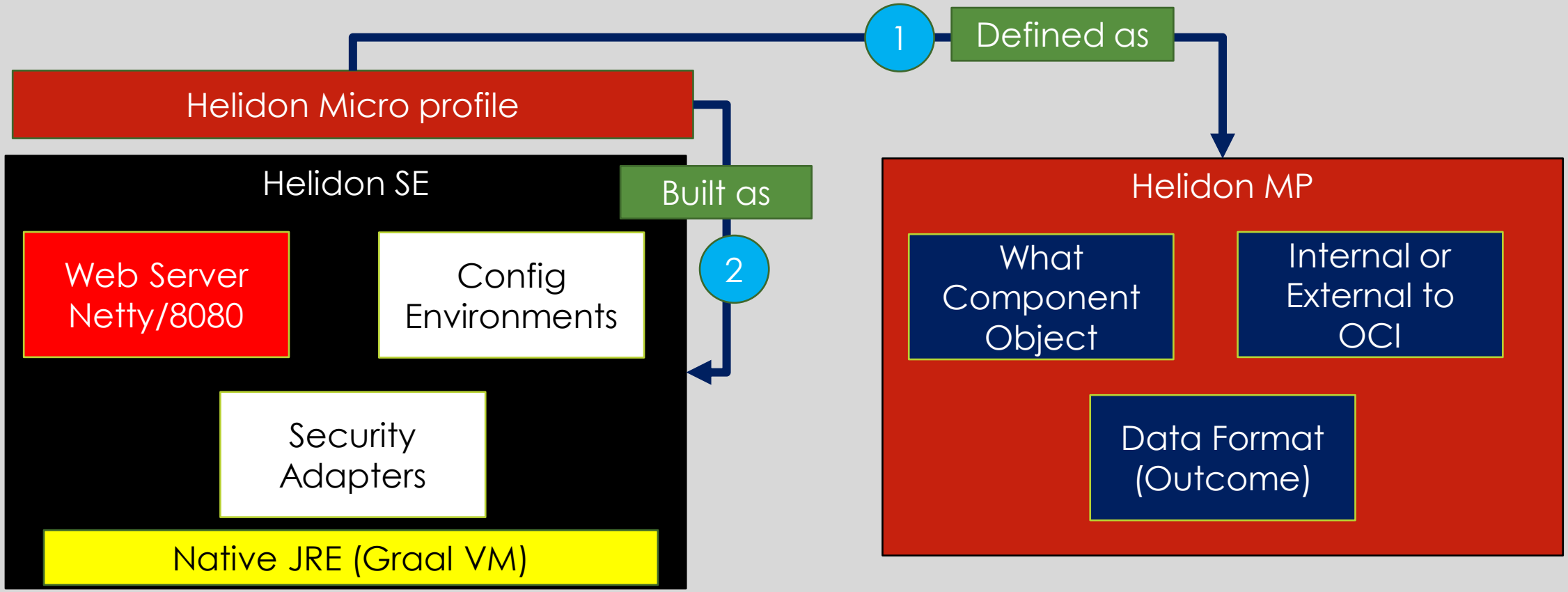
Micro

Non-CLOUD , CLOUD NATIVE APPLICATIONS : MICROSERVICES++
Procedural : Helidon SE ,Javalin , SPARK , Micronaut (Serverless)

Helidon SE vs Helidon MP

Helidon SE	Helidon MP
Functional/Procedural Development	Declarative Development
Micro framework (CNA, Layered)	CAN, Microprofile
Reactive (Execution Plan, Exceptions Rescue) – Avoid Risks	Capability Model – Handle Risks
API , Binaries , SE	CDI (Dependency Injection), @Annotations, Java EE
Config, Database , Kafka Connectors, Streams, Jedis, Redis, Connection Pool, OCI Storage Driver , JSON Library (Explicit API to connect to External Product)	CDI , CDI Extensions – Explicit Automated Annotations , JSON (P), JSON (B)

Helidon Architecture (Java EE)



Helidon API

- Default JDK Runtime – JDK 11+, Maven 3.6+
- Other Runtime – Go, Ruby, Python, java Script (node js) , R ,(.NET)
- Simple by Structure, Developed in Modern React way
- Exclusive for Microservices , Light weight

Helidon SE

Configuration	Security	WebServer
- **Configure Environments** - Data Sources - **Formats (Logs/Outcome)** - Extensions (Ports, Bindings) - Dynamic updates (Monitoring – tracing, metrics)	- Connect to Security Provider (LDAP, Exchange) - Authentication - Authorization - Auditing - Keys Support	X Netty - Tracing API - Reactive API - Static content support

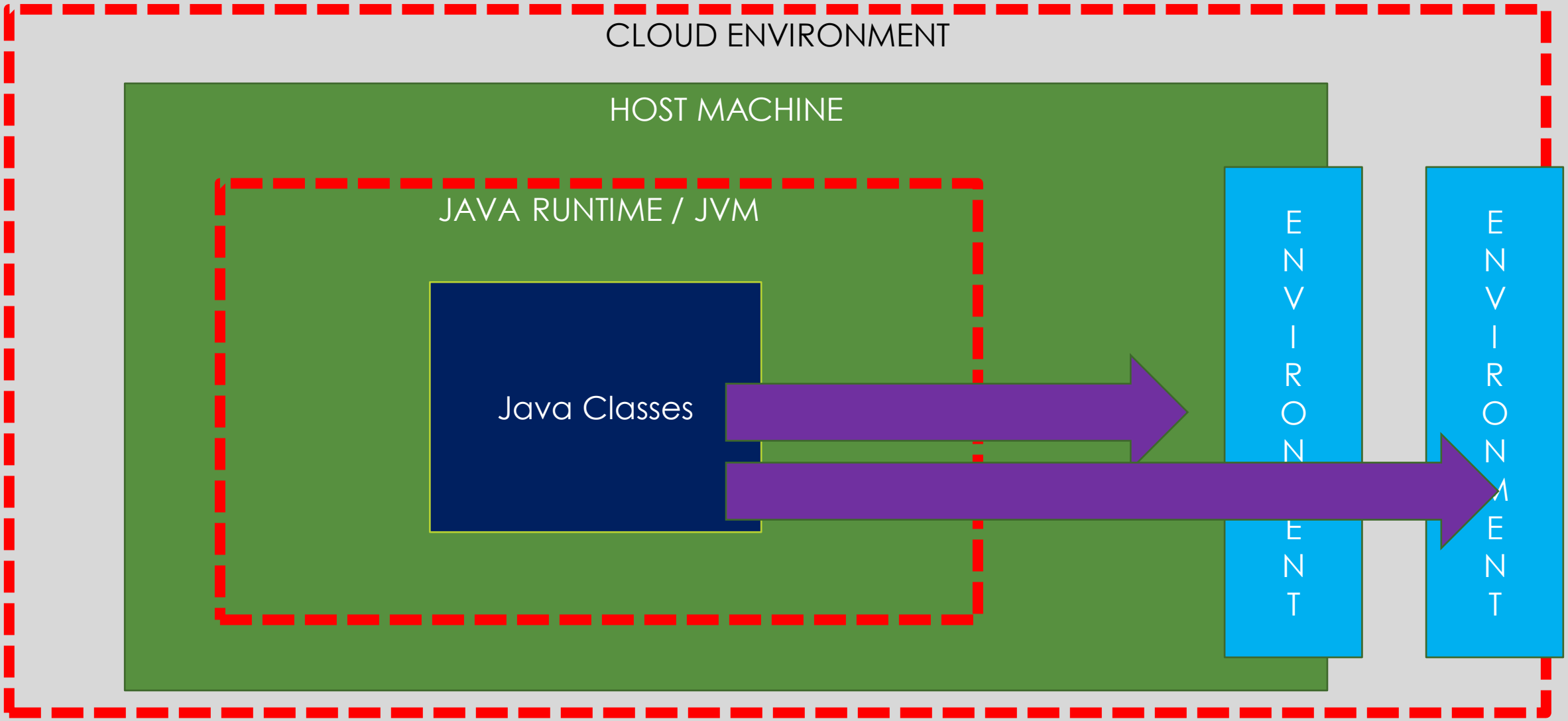
Helidon MP

- Open source for Enterprise Java
- Light weight , Microservice Approach
- Open Tracing API (RCA)
- Open API
- Rest Clients (REST API)
- Configure (MP)
- gRPC /RPC
- Fault Tolerance
- Health Checks, Metrics, Probes

- Integration with OCI (CDI Extension)
- JWT Authentication
- Integration with JPA
- Integration with JTA
- JSON Parsing and JSON Binding
- JAX RS (REST API)

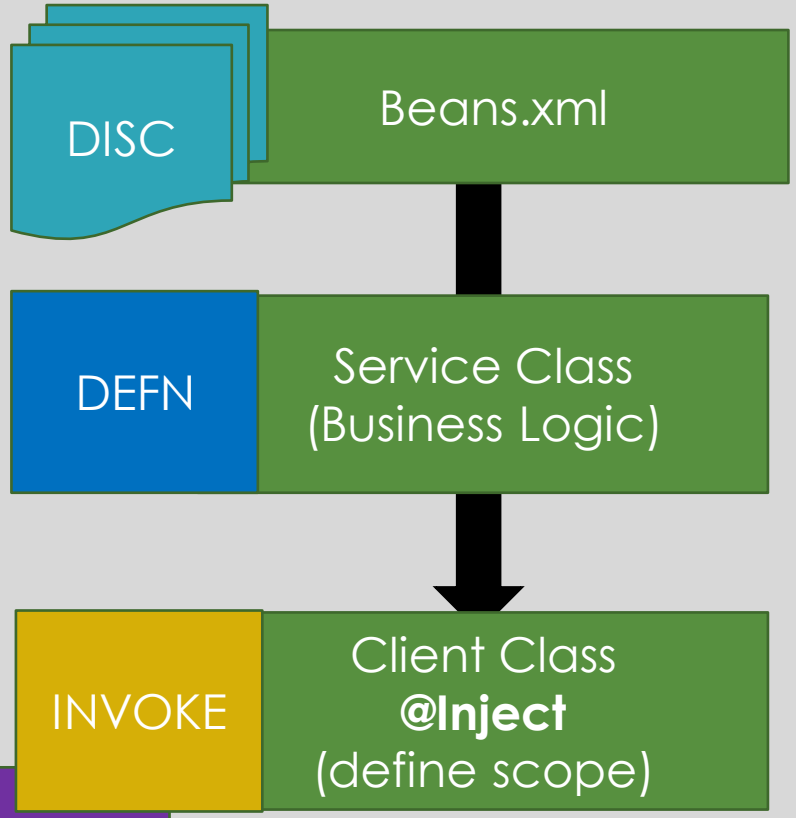
CDI

CDI


CLOUD ENVIRONMENT
HOST MACHINE
JAVA RUNTIME / JVM
Java Classes
ENVIRONMENT
ENVIRONMENT

Context and Dependency Injection

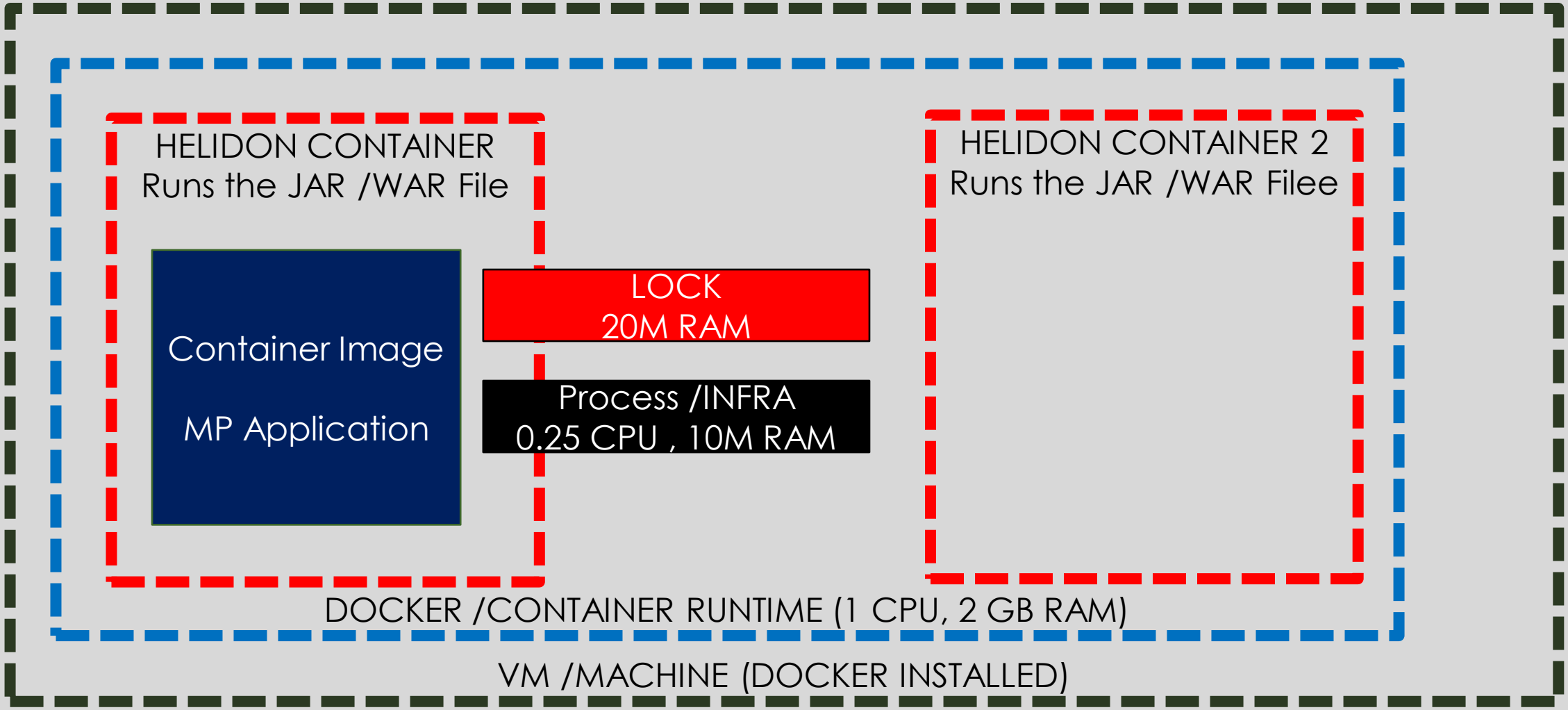
- Loosely Coupled
- Standard Dependency Injection as object
- Allows you to manage lifecycle of stateful components
- - Fields
- - Services
- - Components (Objects)
- DYDI (Do it yourself Dependency Inject)



Context
Life cycle , Interaction

Dependency Injection
At Scope (Field, Method, Constructor)

Deploying Helidon Containers



Maven Lifecycle

Create or Generate Sources from central Repository

~/.m2

Download of the Dependencies

#pom.xml

Resource Graph

Compile and Validate

 # mvn package

Build Environment

Jar file

Container Image
VM Application
(makefile)

Setting up Helidon Microprofile Application

Maven Project Template

Microservices Architecture Format

SE/**MP**

Review Project Resources in Eclipse

Life cycle Configured

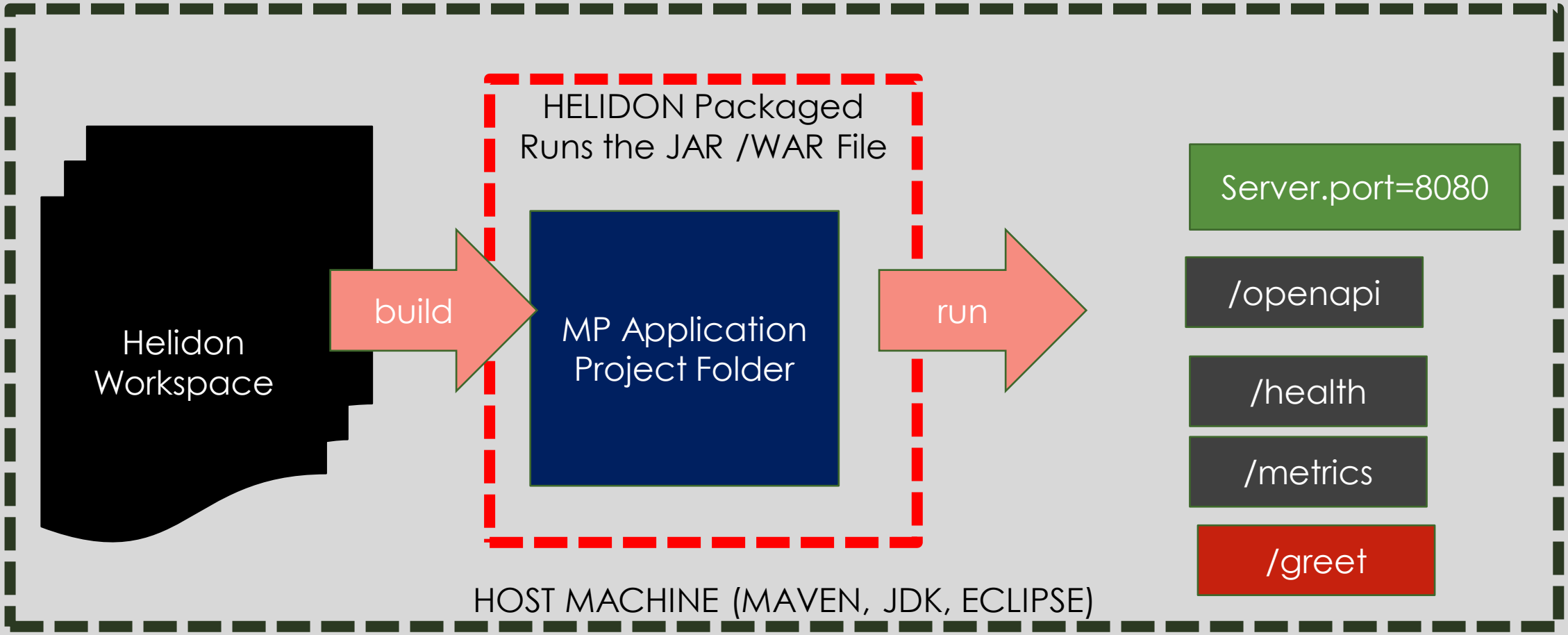
Compile the package

#mvn package

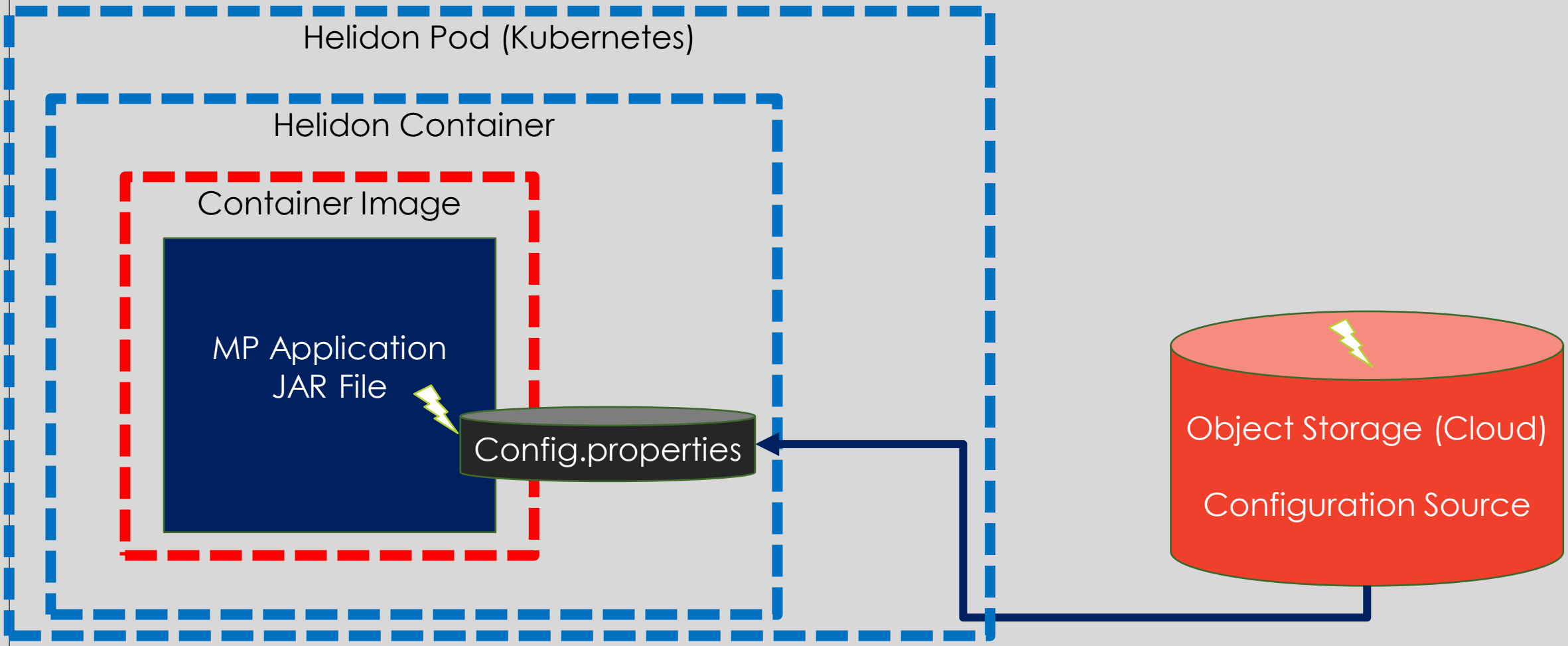
Review End Points

/greet
/health
/metrics
/openapi

Launch Micro-profile Application (Slide 55)



Configure Properties



Configuration Scope

- **Config**
  - Comprehensive Configuration
    - INI Notation (/etc/hosts)
    - JSON Notation
    - YAML Notation
    - .Properties Notation
  - **Customize Precedence of Resources**
  - **Make the precedence – optional or mandatory**

Configuring the Main Server

1 Configuring Environment Variables

2 Order of Precedence

Xyz.properties

1

Microrprofile-config.properties

//Local to project

2

Environment Variables

java –D APP_GREETING=

Configure Server

- Io.helidon.microprofile.server.Server
- Server Builder → Static Server Instance
- Start builder → builder().start () (CRUD on Server)
- Configure Server → Server.builder().config (Config()). build().start()
- Config() → Config
  - Disableenvironmentsource() /stop default configuration path
    sources (1) //ClassPath
    source (2)

Use case : Start and Configure Server

Helidon MP Template

Create Custom Build Main () //Start Server Instance

Without Config Scope

Custom Config Resource Scope

Custom Start Server Instance with Configuration

T E S T

Use case 2 – Extending

Helidon MP Template

Custom Config Resource Scope

Custom Start Server Instance with Configuration And Precedence

T
E
S
T