In28 Minutes

Master Microservices

Spring Boot in 10(ish) Steps

Getting Started with Spring Boot

In28 Minutes

- **WHY** Spring Boot?
  - You can build web apps & REST API WITHOUT Spring Boot
  - What is the need for Spring Boot?
- **WHAT** are the goals of Spring Boot?
- **HOW** does Spring Boot work?
- **COMPARE** Spring Boot vs Spring MVC vs Spring

Getting Started with Spring Boot - Approach

In28 Minutes

- **1:** Understand the world before Spring Boot (10000 Feet)
- **2:** Create a Spring Boot Project
- **3:** Build a simple REST API using Spring Boot
- **4:** Understand the MAGIC of Spring Boot
  - Spring Initializr
  - Starter Projects
  - Auto Configuration
  - Developer Tools
  - Actuator
  - ...

World Before Spring Boot!

In28 Minutes

- Setting up Spring Projects **before Spring Boot was NOT easy**!
- We needed to configure a **lot of things** before we have a **production-ready** application

World Before Spring Boot - 1 - Dependency Management

In 28 Minutes

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>6.2.2.RELEASE</version>
</dependency>
<dependency>
 <groupId>com.fasterxml.jackson.core</groupId>
 <artifactId>jackson-databind</artifactId>
 <version>2.13.3</version>
</dependency>
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
</dependency>
```

- Manage frameworks and versions
  - **REST API** - Spring framework, Spring MVC framework, JSON binding framework, ..
  - **Unit Tests** - Spring Test, Mockito, JUnit, ...

World Before Spring Boot - 2 - web.xml

In28 Minutes

```
<servlet>
 <servlet-name>dispatcher</servlet-name>
 <servlet-class>

  org.springframework.web.servlet.DispatcherServlet
 </servlet-class>
 <init-param>

  <param-name>contextConfigLocation</param-name>

  <param-value>/WEB-INF/todo-servlet.xml</param-value>
 </init-param>
 <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
 <servlet-name>dispatcher</servlet-name>
 <url-pattern>/*</url-pattern>
</servlet-mapping>
```

- **Example**: Configure **DispatcherServlet** for Spring MVC

World Before Spring Boot - 3 - Spring Configuration

In28 Minutes

```
<context:component-scan base-package="com.in28minutes" />

<bean
 class="org.springframework.web.servlet.view.InternalResourceViewResolver">
 <property name="prefix">

  <value>/WEB-INF/views/</value>
 </property>
 <property name="suffix">

  <value>.jsp</value>
 </property>
</bean>
```

- Define your **Spring Configuration**
  - Component Scan
  - View Resolver
  - ....

World Before Spring Boot - 4 - NFRs

In28 Minutes

```
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
        <path>/</path>
        <contextReloadable>true</contextReloadable>
    </configuration>
</plugin>

<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
</dependency>
```

- Logging
- Error Handling
- Monitoring

World Before Spring Boot!

In28 Minutes

- Setting up Spring Projects **before Spring Boot was NOT easy**!
  - 1: Dependency Management (**pom.xml**)
  - 2: Define Web App Configuration (**web.xml**)
  - 3: Manage Spring Beans (**context.xml**)
  - 4: Implement Non Functional Requirements (NFRs)
- AND repeat this for every new project!
- Typically takes a **few days** to setup for each project (and countless hours to maintain)

Understanding Power of Spring Boot

In28 Minutes

```
// http://localhost:8080/courses
[
  {
 "id": 1,
 "name": "Learn AWS",
 "author": "in28minutes"
  }
]
```

- **1:** Create a Spring Boot Project
- **2:** Build a simple REST API using Spring Boot

What's the Most Important Goal of Spring Boot?

In28 Minutes

- Help you build **PRODUCTION-READY** apps **QUICKLY**
  - Build **QUICKLY**
    - Spring Initializr
    - Spring Boot Starter Projects
    - Spring Boot Auto Configuration
    - Spring Boot DevTools
  - Be **PRODUCTION-READY**
    - Logging
    - Different Configuration for Different Environments
      - Profiles, ConfigurationProperties
    - Monitoring (Spring Boot Actuator)
    - ...

Spring Boot
BUILD QUICKLY

Exploring Spring Boot Starter Projects

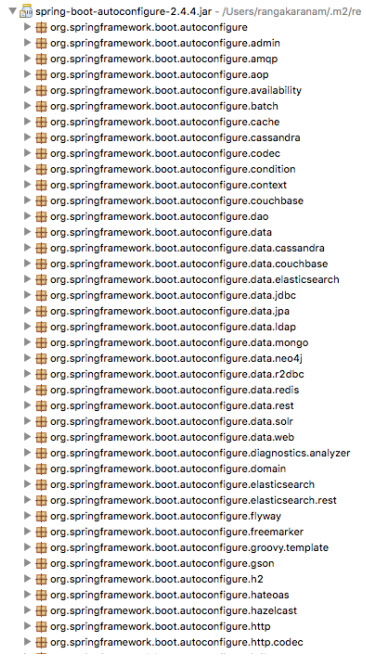
In28 Minutes

- I need a lot of frameworks to build application features:
  - **Build a REST API**: I need Spring, Spring MVC, Tomcat, JSON conversion...
  - **Write Unit Tests**: I need Spring Test, JUnit, Mockito, ...
- How can I group them and make it easy to build applications?
  - **Starters**: Convenient **dependency descriptors** for diff. features
- **Spring Boot** provides variety of starter projects:
  - **Web Application & REST API** - Spring Boot Starter Web (spring-webmvc, spring-web, spring-boot-starter-tomcat, spring-boot-starter-json)
  - **Unit Tests** - Spring Boot Starter Test
  - **Talk to database using JPA** - Spring Boot Starter Data JPA
  - **Talk to database using JDBC** - Spring Boot Starter JDBC
  - **Secure your web application or REST API** - Spring Boot Starter Security
- (REMEMBER) **Starters**: Define all application dependencies

Exploring Spring Boot Auto Configuration

In28 Minutes

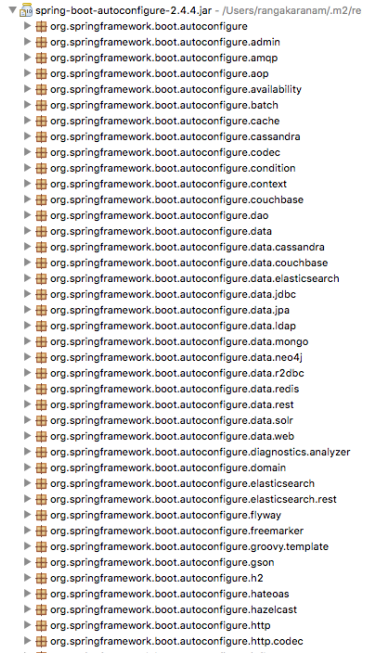
- I need **lot of configuration** to build Spring app:
  - Component Scan, DispatcherServlet, Data Sources, JSON Conversion, ...
- How can I simplify this?
  - **Auto Configuration**: **Automated configuration** for your app
    - **Decided** based on:
      - Which frameworks are in the Class Path?
      - What is the existing configuration (Annotations etc)?
- **Example**: Spring Boot Starter Web
  - Dispatcher Servlet (`DispatcherServletAutoConfiguration`)
  - Embedded Servlet Container - Tomcat is the default (`EmbeddedWebServerFactoryCustomizerAutoConfiguration`)
  - Default Error Pages (`ErrorMvcAutoConfiguration`)
  - Bean<->JSON (`JacksonHttpMessageConvertersConfiguration`)



Understanding the Glue - @SpringBootApplication

In 28 Minutes

- Questions:
  - Who is launching the Spring Context?
  - Who is triggering the component scan?
  - Who is enabling auto configuration?
- Answer: **@SpringBootApplication**
  - 1: **@SpringBootConfiguration**: Indicates that a class provides Spring Boot application @Configuration.
  - 2: **@EnableAutoConfiguration**: Enable auto-configuration of the Spring Application Context,
  - 3: **@ComponentScan**: Enable component scan (for current package, by default)



Build Faster with Spring Boot DevTools

In28 Minutes

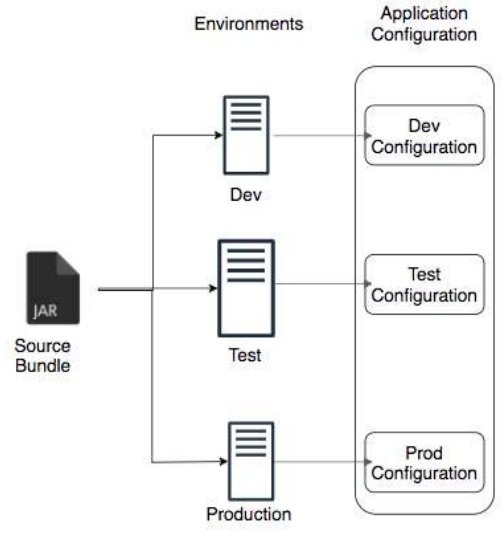
- Increase developer productivity
- Why do you need to restart the server **manually** for every code change?
- **Remember**: For pom.xml dependency changes, you will need to restart server **manually**

Spring Boot
PRODUCTION-READY

Managing App. Configuration using Profiles

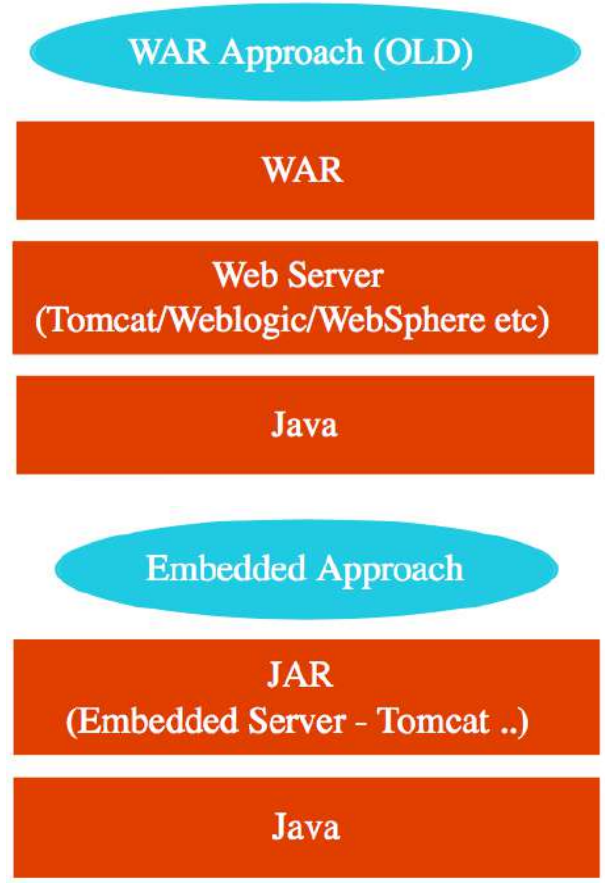
In28 Minutes

- Applications have different environments: **Dev, QA, Stage, Prod**, ...
- Different environments need **different configuration**:
  - Different Databases
  - Different Web Services
- How can you provide different configuration for different environments?
  - **Profiles**: Environment specific configuration
- How can you define externalized configuration for your application?
  - **ConfigurationProperites**: Define externalized configuration



Simplify Deployment with Spring Boot Embedded Servers

- How do you deploy your application?
  - Step 1 : Install Java
  - Step 2 : Install Web/Application Server
    - Tomcat/WebSphere/WebLogic etc
  - Step 3 : Deploy the application WAR (Web ARchive)
    - This is the OLD **WAR** Approach
    - Complex to setup!
- **Embedded Server** - Simpler alternative
  - Step 1 : Install Java
  - Step 2 : Run **JAR** file
  - **Make JAR not WAR** (Credit: Josh Long!)
  - Embedded Server **Examples**:
    - spring-boot-starter-tomcat
    - spring-boot-starter-jetty
    - spring-boot-starter-undertow



Monitor Applications using Spring Boot Actuator

In28 Minutes

- Monitor and manage your application in your production
- Provides a number of endpoints:
  - **beans** - Complete list of Spring beans in your app
  - **health** - Application health information
  - **metrics** - Application metrics
  - **mappings** - Details around Request Mappings

Understanding Spring Boot vs Spring MVC vs Spring

In28 Minutes

- **Spring Boot vs Spring MVC vs Spring**: What's in it?
  - **Spring Framework**: Dependency Injection
    - @Component, @Autowired, Component Scan etc..
    - Just Dependency Injection is NOT sufficient (You need other frameworks to build apps)
      - **Spring Modules and Spring Projects**: Extend Spring Eco System
        - Provide good integration with other frameworks (Hibernate/JPA, JUnit & Mockito for Unit Testing)
  - **Spring MVC** (Spring Module): Simplify building web apps and REST API
    - Building web applications with Struts was very complex
    - @Controller, @RestController, @RequestMapping("/courses")
  - **Spring Boot** (Spring Project): Build **PRODUCTION-READY** apps **QUICKLY**
    - **Starter Projects** - Make it easy to build variety of applications
    - **Auto configuration** - Eliminate configuration to setup Spring, Spring MVC and other frameworks!
    - Enable non functional requirements (NFRs):
      - **Actuator**: Enables Advanced Monitoring of applications
      - **Embedded Server**: No need for separate application servers!
      - Logging and Error Handling
      - Profiles and ConfigurationProperties

Spring Boot - Review

- **Goal**: 10,000 Feet overview of Spring Boot
  - Help you understand the terminology!
    - Starter Projects
    - Auto Configuration
    - Actuator
    - DevTools
- **Advantages**: Get started quickly with production ready features!

In28 Minutes

Building REST API with Spring Boot

Building REST API with Spring Boot - Goals

- **WHY** Spring Boot?
  - You can build REST API WITHOUT Spring Boot
  - What is the need for Spring Boot?
- **HOW** to build a great REST API?
  - Identifying Resources (/users, /users/{id}/posts)
  - Identifying Actions (GET, POST, PUT, DELETE, ...)
  - Defining Request and Response structures
  - Using appropriate Response Status (200, 404, 500, ..)
  - Understanding REST API Best Practices
    - Thinking from the perspective of your consumer
    - Validation, Internationalization - i18n, Exception Handling, HATEOAS, Versioning, Documentation, Content Negotiation and a lot more!

Building REST API with Spring Boot - Approach

In28 Minutes

- **1:** Build 3 Simple Hello World REST API
  - Understand the magic of Spring Boot
  - Understand fundamentals of building REST API with Spring Boot
    - @RestController, @RequestMapping, @PathVariable, JSON conversion
- **2:** Build a REST API for a Social Media Application
  - Design and Build a Great REST API
    - Choosing the right URI for resources (/users, /users/{id}, /users/{id}/posts)
    - Choosing the right request method for actions (GET, POST, PUT, DELETE, ..)
    - Designing Request and Response structures
    - Implementing Security, Validation and Exception Handling
  - Build Advanced REST API Features
    - Internationalization, HATEOAS, Versioning, Documentation, Content Negotiation, ...
- **3:** Connect your REST API to a Database
  - Fundamentals of JPA and Hibernate
  - Use H2 and MySQL as databases

What's Happening in the Background?

- Let's explore some **Spring Boot Magic**: Enable Debug Logging
  - **WARNING**: Log change frequently!
- **1:** How are our requests handled?
  - **DispatcherServlet** - Front Controller Pattern
    - `Mapping servlets: dispatcherServlet urls=[/]`
    - **Auto Configuration** (`DispatcherServletAutoConfiguration`)
- **2:** How does **HelloWorldBean** object get converted to JSON?
  - @ResponseBody + JacksonHttpMessageConverters
    - **Auto Configuration** (`JacksonHttpMessageConvertersConfiguration`)
- **3:** Who is configuring error mapping?
  - **Auto Configuration** (`ErrorMvcAutoConfiguration`)
- **4:** How are all jars available(Spring, Spring MVC, Jackson, Tomcat)?
  - **Starter Projects** - Spring Boot Starter Web (spring-webmvc, spring-web, spring-boot-starter-tomcat, spring-boot-starter-json)

Social Media Application REST API

- Build a REST API for a Social Media Application
- **Key Resources**:
  - Users
  - Posts
- **Key Details**:
  - User: id, name, birthDate
  - Post: id, description

Request Methods for REST API

In28 Minutes

- **GET** - Retrieve details of a resource
- **POST** - Create a new resource
- **PUT** - Update an existing resource
- **PATCH** - Update part of a resource
- **DELETE** - Delete a resource

localhost:8080/users


```
[
  {
    "id": 1,
    "name": "Adam",
    "birthDate": "2022-08-16"
  },
  {
    "id": 2,
    "name": "Eve",
    "birthDate": "2022-08-16"
  },
  {
    "id": 3,
    "name": "Jack",
    "birthDate": "2022-08-16"
  }
]
```

Social Media Application - Resources & Methods

In28 Minutes

- **Users REST API**
  - Retrieve all Users
    - **GET /users**
  - Create a User
    - **POST /users**
  - Retrieve one User
    - **GET /users/{id} -> /users/1**
  - Delete a User
    - **DELETE /users/{id} -> /users/1**
  - **Posts REST API**
    - Retrieve all posts for a User
      - **GET /users/{id}/posts**
    - Create a post for a User
      - **POST /users/{id}/posts**
    - Retrieve details of a post
      - **GET /users/{id}/posts/{post_id}**

Response Status for REST API

In28 Minutes

- Return the **correct response status**
  - Resource is not found => 404
  - Server exception => 500
  - Validation error => 400
- **Important Response Statuses**
  - **200** — Success
  - **201** — Created
  - **204** — No Content
  - **401** — Unauthorized (when authorization fails)
  - **400** — Bad Request (such as validation error)
  - **404** — Resource Not Found
  - **500** — Server Error

Advanced REST API Features

In28 Minutes

- Documentation
- Content Negotiation
- Internationalization - i18n
- Versioning
- HATEOAS
- Static Filtering
- Dynamic Filtering
- Monitoring
- ....

REST API Documentation

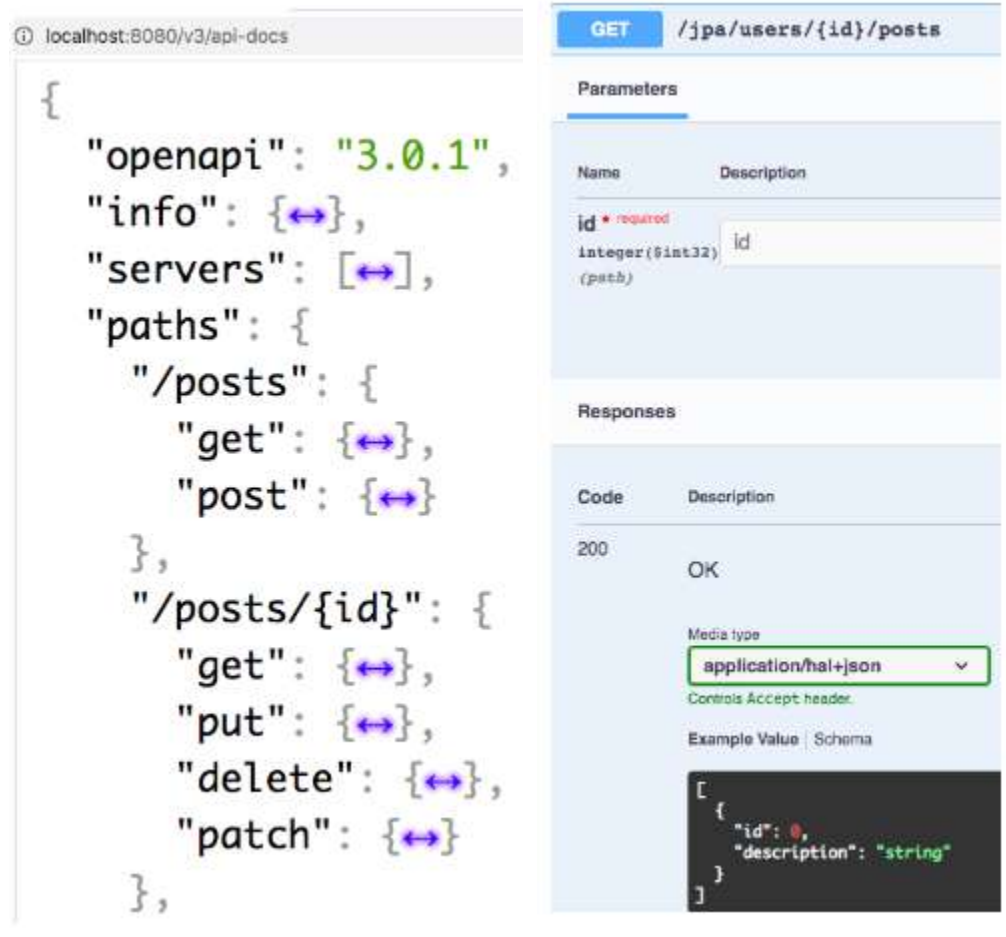
In28 Minutes

- Your REST API consumers need to understand your REST API:
  - Resources
  - Actions
  - Request/Response Structure (Constraints/Validations)
- **Challenges**:
  - Accuracy: How do you ensure that your documentation is upto date and correct?
  - Consistency: You might have 100s of REST API in an enterprise. How do you ensure consistency?
- **Options**:
  - **1**: Manually Maintain Documentation
    - Additional effort to keep it in sync with code
  - **2**: Generate from code Swagger and Open API

REST API Documentation - Swagger and Open API

In28 Minutes

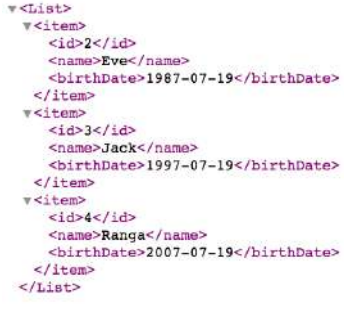
- **Quick overview:**
  - **2011:** Swagger Specification and Swagger Tools were introduced
  - **2016:** Open API Specification created based on Swagger Spec.
    - Swagger Tools (ex:Swagger UI) continue to exist
  - **OpenAPI Specification**: Standard, language-agnostic interface
    - Discover and understand REST API
    - Earlier called Swagger Specification
  - **Swagger UI**: Visualize and interact with your REST API
    - Can be generated from your OpenAPI Specification



Content Negotiation

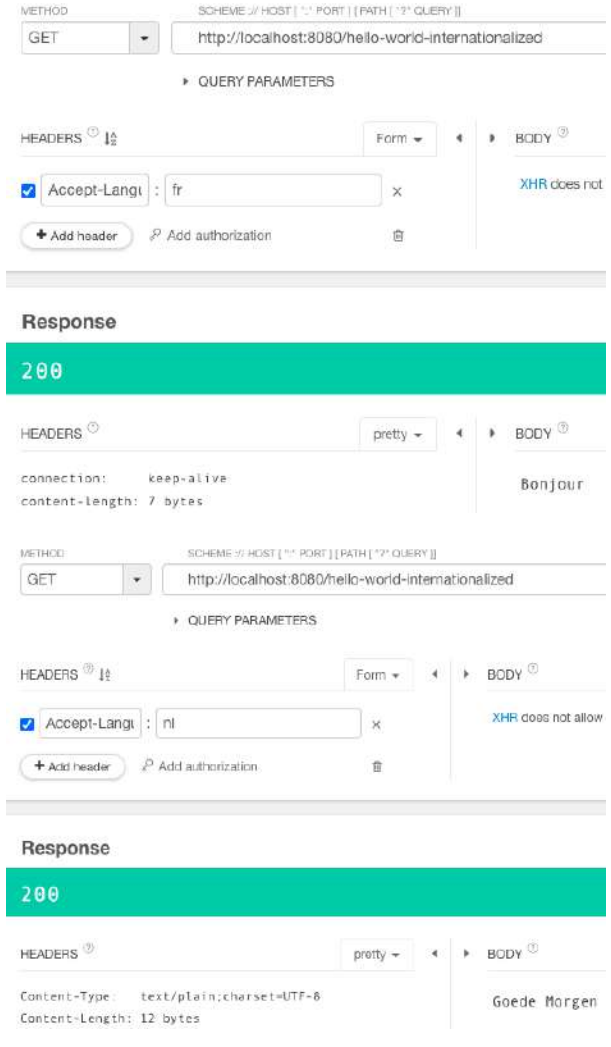
In28 Minutes

- **Same Resource** - Same URI
  - HOWEVER **Different Representations** are possible
    - Example: Different Content Type - XML or JSON or ..
    - Example: Different Language - English or Dutch or ..
- How can a consumer tell the REST API provider what they want?
  - **Content Negotiation**
- Example: Accept header (MIME types - application/xml, application/json, ..)
- Example: Accept-Language header (en, nl, fr, ..)



Internationalization - i18n

- Your REST API might have consumers from around the world
- How do you customize it to users around the world?
  - **Internationalization - i18n**
- Typically **HTTP Request Header - `Accept-Language`** is used
  - `Accept-Language` - indicates natural language and locale that the consumer prefers
  - Example: `en` - English (Good Morning)
  - Example: `nl` - Dutch (Goedemorgen)
  - Example: `fr` - French (Bonjour)
  - Example: `de` - Deutsch (Guten Morgen)



Versioning REST API

In 28 Minutes

- You have built an amazing REST API
  - You have 100s of consumers
  - You need to implement a breaking change
    - Example: Split name into firstName and lastName
- **SOLUTION**: Versioning REST API
  - **Variety of options**
    - URL
    - Request Parameter
    - Header
    - Media Type
  - **No Clear Winner!**

localhost:8080/v1/person

```
{
  "name": "Bob Charlie"
}
```

localhost:8080/v2/person

```
{
  "name": {
    "firstName": "Bob",
    "lastName": "Charlie"
  }
}
```

Versioning REST API - Options

In28 Minutes

- **URI Versioning** - Twitter
  - *http://localhost:8080/v1/person*
  - *http://localhost:8080/v2/person*
- **Request Parameter versioning** - Amazon
  - *http://localhost:8080/person?version=1*
  - *http://localhost:8080/person?version=2*
- **(Custom) headers versioning** - Microsoft
  - SAME-URL headers=[X-API-VERSION=1]
  - SAME-URL headers=[X-API-VERSION=2]
- **Media type versioning** (a.k.a "content negotiation" or "accept header") - GitHub
  - SAME-URL produces=application/vnd.company.app-v1+json
  - SAME-URL produces=application/vnd.company.app-v2+json

Versioning REST API - Factors

In28 Minutes

- **Factors to consider**
  - URI Pollution
  - Misuse of HTTP Headers
  - Caching
  - Can we execute the request on the browser?
  - API Documentation
  - **Summary**: No Perfect Solution
- **My Recommendations**
  - Think about versioning even before you need it!
  - One Enterprise - One Versioning Approach

URI Versioning - Twitter
- *http://localhost:8080/v1/person*
- *http://localhost:8080/v2/person*

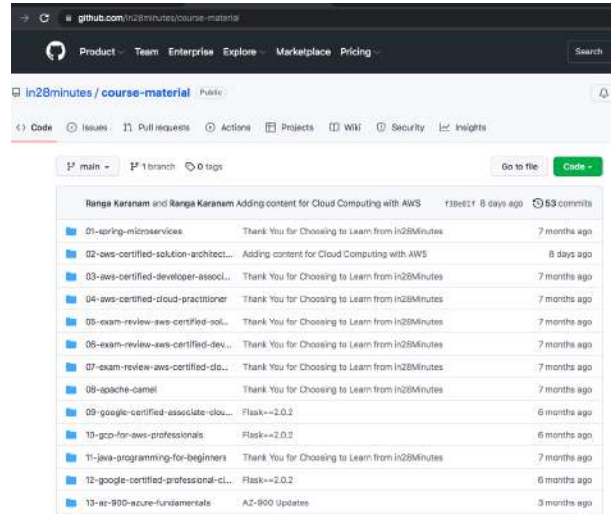
Request Parameter versioning - Amazon
- *http://localhost:8080/person?version=1*
- *http://localhost:8080/person?version=2*

(Custom) headers versioning - Microsoft
- SAME-URL headers=[X-API-VERSION=1]
- SAME-URL headers=[X-API-VERSION=2]

Media type versioning - GitHub
- SAME-URL produces=application/vnd.company.app-v1+json
- SAME-URL produces=application/vnd.company.app-v2+json

HATEOAS

- **Hypermedia as the Engine of Application State (HATEOAS)**
- Websites allow you to:
  - See **Data** AND Perform **Actions** (using links)
- How about enhancing your REST API to tell consumers how to perform subsequent actions?
  - **HATEOAS**
- **Implementation Options**:
  - **1:** Custom Format and Implementation
    - Difficult to maintain
  - **2:** Use Standard Implementation
    - **HAL (JSON Hypertext Application Language)**: Simple format that gives a consistent and easy way to hyperlink between resources in your API
    - **Spring HATEOAS**: Generate HAL responses with hyperlinks to resources




```
{
  "name": "Adam",
  "birthDate": "2022-08-16",
  "_links": {
    "all-users": {
      "href": "http://localhost:8080/users"
    }
  }
}
```

Customizing REST API Responses - Filtering and more..

- **Serialization**: Convert object to stream (example: JSON)
  - Most popular JSON Serialization in Java: Jackson
- How about customizing the REST API response returned by Jackson framework?
- **1:** Customize field names in response
  - @JSONProperty
- **2:** Return only selected fields
  - **Filtering**
  - Example: Filter out Passwords
  - **Two types**:
    - **Static Filtering**: Same filtering for a bean across different REST API
      - @JsonIgnoreProperties, @JsonIgnore
    - **Dynamic Filtering**: Customize filtering for a bean for specific REST API
      - @JsonFilter with FilterProvider

localhost:8080/filtering-list


```
[
  {
    "field2": "value2",
    "field3": "value3"
  },
  {
    "field2": "value5",
    "field3": "value6"
  }
]
```


localhost:8080/filtering


```
{
  "field1": "value1",
  "field3": "value3"
}
```

Get Production-ready with Spring Boot Actuator

In28 Minutes

- **Spring Boot Actuator**: Provides Spring Boot's production-ready features
  - Monitor and manage your application in your production
- **Spring Boot Starter Actuator**: Starter to add Spring Boot Actuator to your application
  - **spring-boot-starter-actuator**
- Provides a number of endpoints:
  - **beans** - Complete list of Spring beans in your app
  - **health** - Application health information
  - **metrics** - Application metrics
  - **mappings** - Details around Request Mappings
  - and a lot more .....

Explore REST API using HAL Explorer

In28 Minutes

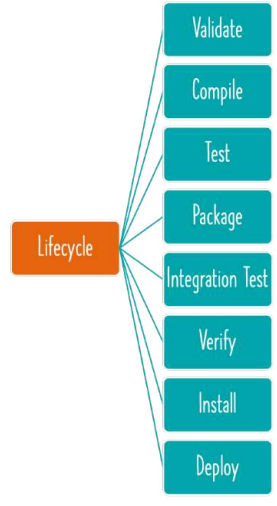
- 1: **HAL (JSON Hypertext Application Language)**
  - Simple format that gives a consistent and easy way to hyperlink between resources in your API
- 2: **HAL Explorer**
  - An API explorer for RESTful Hypermedia APIs using HAL
  - Enable your non-technical teams to play with APIs
- 3: **Spring Boot HAL Explorer**
  - Auto-configures HAL Explorer for Spring Boot Projects
  - spring-data-rest-hal-explorer

Maven

What is Maven?

In28 Minutes

- **Things you do** when writing code each day:
  - Create new projects
  - Manages **dependencies** and their versions
  - Spring, Spring MVC, Hibernate,...
  - Add/modify dependencies
  - **Build** a JAR file
  - Run your application locally in Tomcat or Jetty or ..
  - Run **unit tests**
  - Deploy to a test environment
  - and a lot more..
- Maven helps you do all these and more...



spring

Maven™

Exploring Project Object Model - pom.xml

In28 Minutes

Maven

- Let's explore Project Object Model - pom.xml
  - **1: Maven dependencies**: Frameworks & libraries used in a project
  - Ex: `spring-boot-starter-web` and `spring-boot-starter-test`
  - Why are there so many dependencies in the classpath?
    - Answer: Transitive Dependencies

```
- (REMEMBER) Spring dependencies are DIFFERENT
```

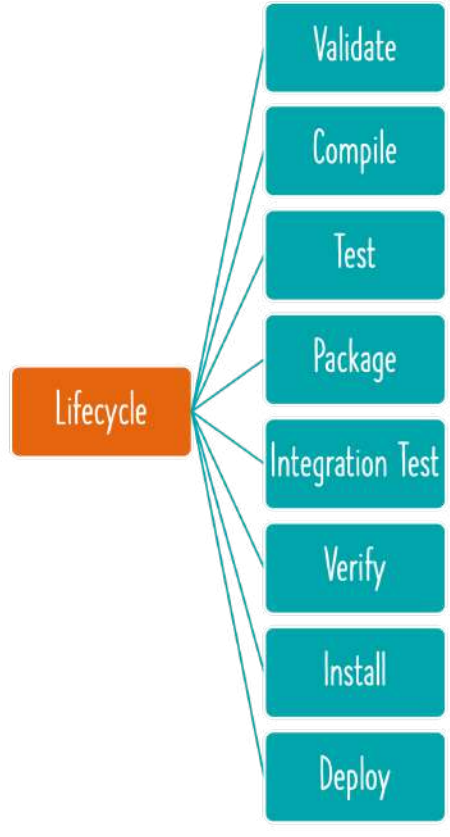
    - **2: Parent Pom**: spring-boot-starter-parent
      - Dependency Management: `spring-boot-dependencies`
      - Properties: `java.version`, plugins and configurations
    - **3: Name of our project**: groupId + artifactId
  - **1: groupId**: Similar to package name
  - **2: artifactId**: Similar to class name
  - **Why is it important?**

- Think about this: How can other projects use our new project?
  - **Activity**: help:effective-pom, dependency:tree & Eclipse UI
  - Let's add a new dependency: spring-boot-starter-web

Exploring Maven Build Life Cycle

In28 Minutes

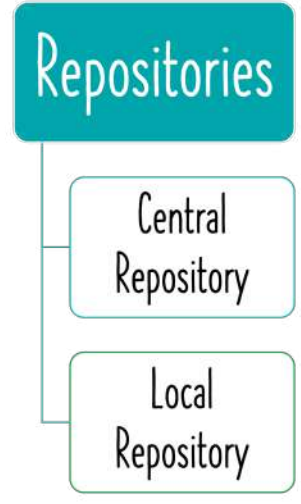
- When we run a maven command, maven build life cycle is used
- Build LifeCycle is a sequence of steps
  - Validate
  - Compile
  - Test
  - Package
  - Integration Test
  - Verify
  - Install
  - Deploy



How does Maven Work?

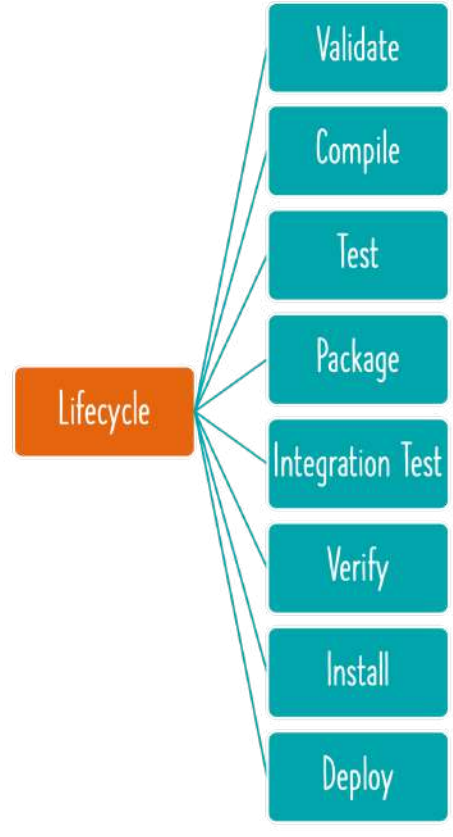
In28 Minutes

- Maven follows **Convention over Configuration**
  - Pre defined folder structure
  - Almost all Java projects follow **Maven structure** (Consistency)
- **Maven central repository** contains jars (and others) indexed by artifact id and group id
  - Stores all the versions of dependencies
  - repositories > repository
  - pluginRepositories > pluginRepository
- When a dependency is added to pom.xml, Maven tries to download the dependency
  - Downloaded dependencies are stored inside your maven local repository
  - **Local Repository** : a temp folder on your machine where maven stores the jar and dependency files that are downloaded from Maven Repository.



Important Maven Commands

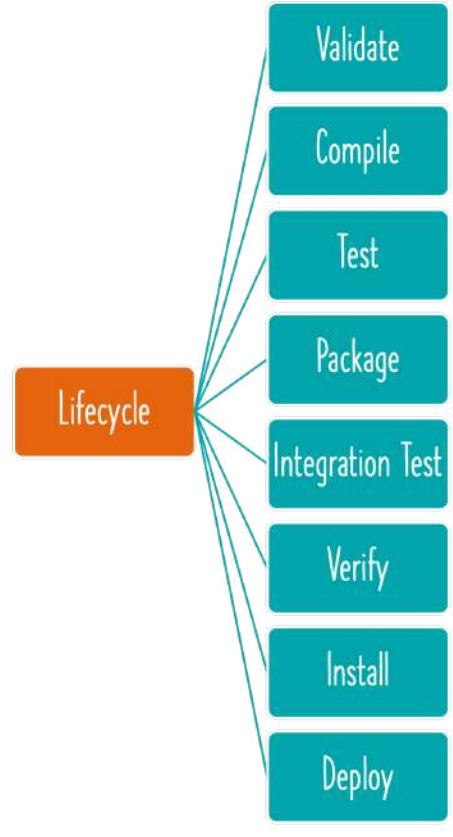
- mvn --version
- mvn compile: Compile source files
- mvn test-compile: Compile test files
  - OBSERVCE CAREFULLY: This will also compile source files
- mvn clean: Delete target directory
- mvn test: Run unit tests
- mvn package: Create a jar
- mvn help:effective-pom
- mvn dependency:tree



Spring Boot Maven Plugin

In28 Minutes

- **Spring Boot Maven Plugin**: Provides Spring Boot support in Apache Maven
  - Example: Create executable jar package
  - Example: Run Spring Boot application
  - Example: Create a Container Image
  - **Commands**:
  - mvn spring-boot:**repackage** (create jar or war)
- Run package using `java -jar`
  - mvn spring-boot:**run** (Run application)
  - mvn spring-boot:**start** (Non-blocking. Use it to run integration tests.)



- mvn spring-boot:**stop** (Stop application started with start command)
- mvn spring-boot:**build-image** (Build a container image)

How are Spring Releases Versioned?

In28 Minutes

- **Version scheme** - MAJOR.MINOR.PATCH[-MODIFIER]
  - **MAJOR**: Significant amount of work to upgrade (10.0.0 to 11.0.0)
  - **MINOR**: Little to no work to upgrade (10.1.0 to 10.2.0)
  - **PATCH**: No work to upgrade (10.5.4 to 10.5.5)
  - **MODIFIER**: Optional modifier
  - **Milestones** - M1, M2, .. (10.3.0-M1,10.3.0-M2)
  - **Release candidates** - RC1, RC2, .. (10.3.0-RC1, 10.3.0-RC2)
  - **Snapshots** - SNAPSHOT
  - **Release** - Modifier will be ABSENT (10.0.0, 10.1.0)
- **Example versions in order**:
  - 10.0.0-SNAPSHOT, 10.0.0-M1, 10.0.0-M2, 10.0.0-RC1, 10.0.0-RC2, 10.0.0, ...
- **MY RECOMMENDATIONS**:
  - Avoid SNAPSHOTs

- Use ONLY Released versions in PRODUCTION

Gradle

Gradle

In28 Minutes

- **Goal**: Build, automate and deliver better software, faster

  - **Build Anything**: Cross-Platform Tool
    - Java, C/C++, JavaScript, Python, ...
  - **Automate Everything**: Completely Programmable
    - Complete flexibility
    - Uses a DSL (domain- specific language)
      - Supports Groovy and Kotlin
  - **Deliver Faster**: Blazing-fast builds
    - Compile avoidance to advanced caching
    - Can speed up Maven builds by up to 90%
      - **Incrementality** — Gradle runs only what is necessary
        - Example: Compiles only changed files
      - **Build Cache** — Reuses the build outputs of other Gradle builds with the same inputs

Maven is a platform-specific tool.
It depends on Java.

If you look at Maven, it was based on pom.xml.
If you'd want to do any programming,you need to create a plugin and create a XML configurationn

- Same project layout as Maven
- IDE support still evolving

Gradle Plugins

- Top 3 Java Plugins for Gradle:
  - **1: Java Plugin**: Java compilation + testing + bundling capabilities
    - **Default Layout**
      - src/main/java: Production Java source
      - src/main/resources: Production resources, such as XML and properties files
      - src/test/java: Test Java source
      - src/test/resources: Test resources
    - **Key Task**: build
  - **2: Dependency Management**: Maven-like dependency management
    - `group:'org.springframework', name:'spring-core', version:'10.0.3.RELEASE'` OR
    - Shortcut: `org.springframework:spring-core:10.0.3.RELEASE`
  - **3: Spring Boot Gradle Plugin**: Spring Boot support in Gradle
    - Package executable Spring Boot jar, Container Image (bootJar, bootBuildImage)
    - Use dependency management enabled by spring-boot-dependencies
      - No need to specify dependency version
        - Ex: `implementation('org.springframework.boot:spring-boot-starter')`

Maven vs Gradle - Which one to Use?

In28 Minutes

- Let's start with a few popular examples:
  - **Spring Framework** - Using Gradle since 2012 (Spring Framework v3.2.0)
  - **Spring Boot** - Using Gradle since 2020 (Spring Boot v2.3.0)
  - **Spring Cloud** - Continues to use Maven even today
  - Last update: Spring Cloud has no plans to switch
- **Top Maven Advantages**: Familiar, Simple and Restrictive
- **Top Gradle Advantages**: Faster build times and less verbose
- **What Do I Recommend**: I'm sitting on the fence for now
  - Choose whatever tool best meets your projects needs
    - If your builds are taking really long, go with Gradle
    - If your builds are simple, stick with Maven

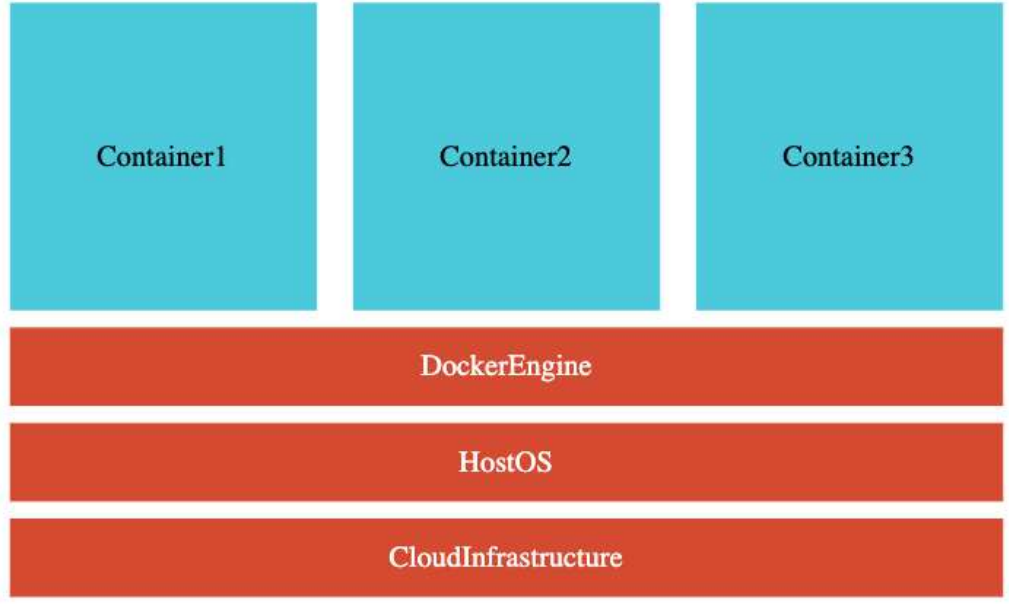
In28 Minutes

Microservices

Microservices - V2

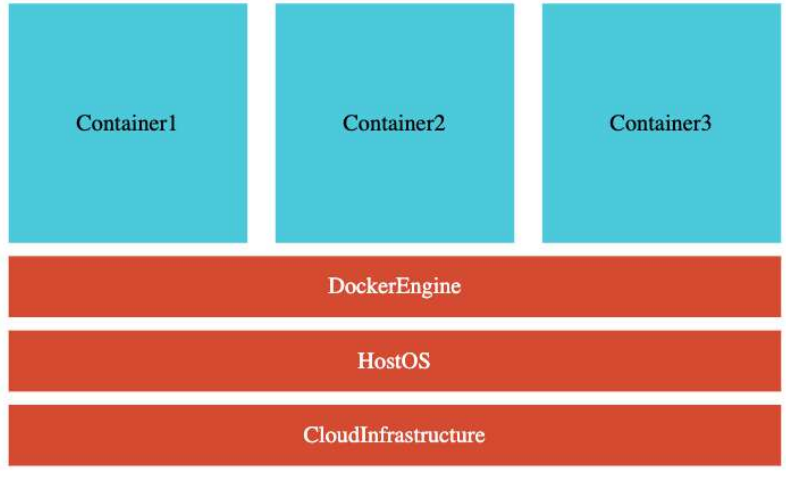
In28 Minutes

- **V2 (2.4+)** - Latest Releases of
  - Spring Boot
  - Spring Cloud
  - Docker and
  - Kubernetes
  - **Skip to Next Section :)**
- **V1** - Old Versions
  - Spring Boot v2.3 and LOWER
  - **Continue on to next lecture :(**



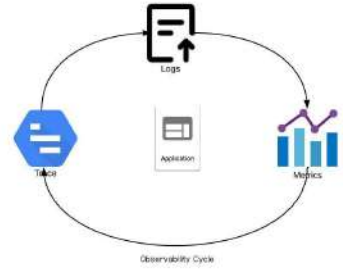
Microservices - Evolution

- **Goal**: Evolve with Microservices
  - **V1** - Spring Boot 2.0.0 to 2.3.x
  - **V2** - Spring Boot 2.4.0 to 3.0.0 to ...
    - Spring Cloud LoadBalancer (Ribbon)
    - Spring Cloud Gateway (Zuul)
    - Resilience4j (Hystrix)
    - NEW: Docker
    - NEW: Kubernetes
    - NEW: Observability
      - NEW: Micrometer (Spring Cloud Sleuth)
      - NEW: OpenTelemetry



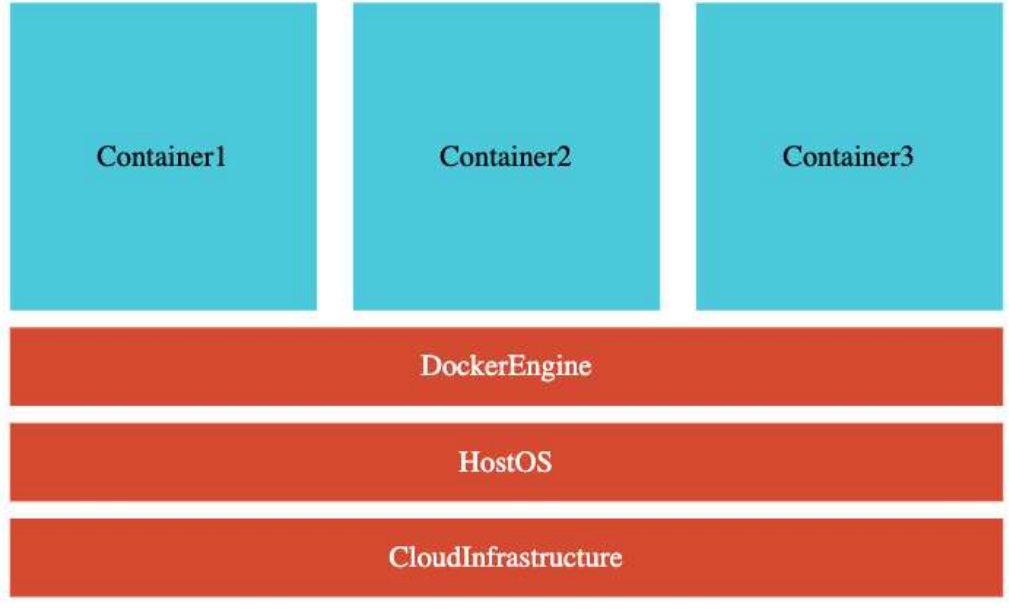
Microservices - Spring Boot 2 vs Spring Boot 3

- **V1**(2.0.0 to 2.3.x)
- **V2** (2.4.x to 3.0.0 to ..)
- Spring Boot 2.4.0+
  - ***https://github.com/in28minutes/spring-microservices-v2***
- Spring Boot 3.0.0+
  - ***https://github.com/in28minutes/spring-microservices-v3***
  - Notes: v3-upgrade.md
  - Key Changes:
    - **Observability** - Ability of a system to measure its current state based on the generated data
      - Monitoring is reactive while Observability is proactive
      - **OpenTelemetry**: One Standard for Logs + Traces + Metrics
        - Cross Language
        - Cross Platform
      - **Micrometer** (Replaces Spring Cloud Sleuth)
        - Collect (Logs + Traces + Metrics)



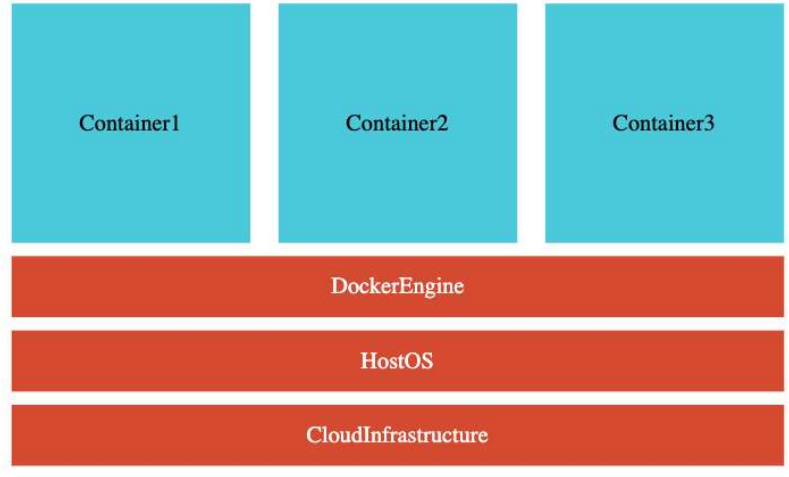
Microservices - V2

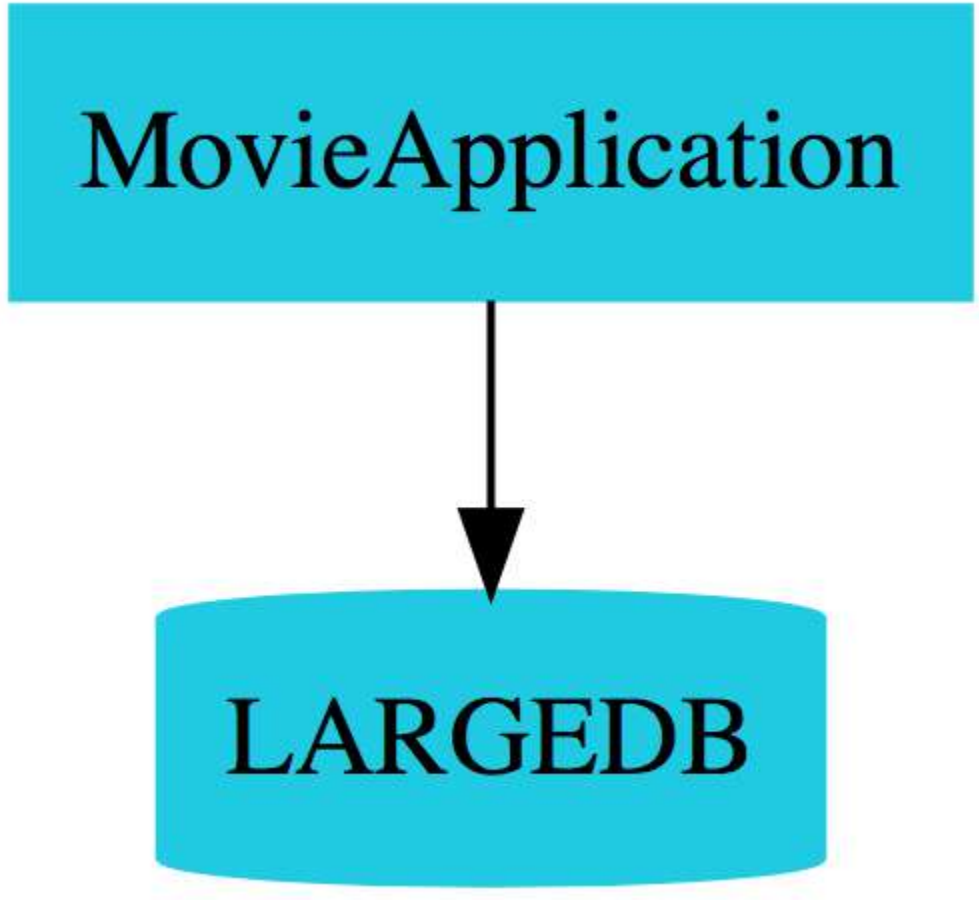
- You have **skipped V1**
  - Go to next lecture!
- You have **completed V1**
  - Option 1: **Start from Zero Again**:
    - Go to the next lecture!
  - Option 2: **Get a Quick Start**:
    - Jump to **"Step 21 - QuickStart by Importing Microservices"**
      - Same microservices as V1: **Currency Exchange** and **Currency Conversion**
      - Very little changes in **Eureka Naming Server**
      - **Step 21** helps you set these up and get started quickly!



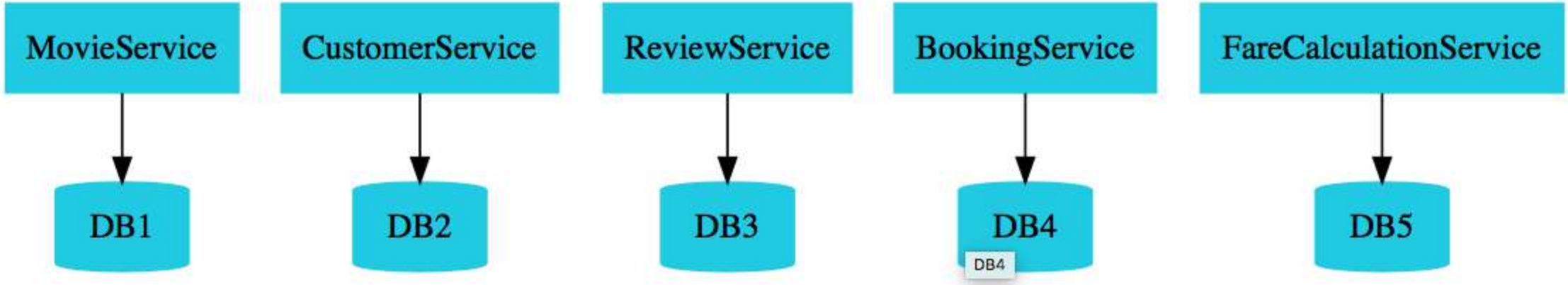
Microservices - V2 - What's New

- Microservices Evolve Quickly
- **V2** (Spring Boot - 2.4.x to 3.0.0 to LATEST)
  - **Spring Cloud LoadBalancer** instead of Ribbon
  - **Spring Cloud Gateway** instead of Zuul
  - **Resilience4j** instead of Hystrix
  - **Docker**: Containerize Microservices
    - Run microservices using Docker and Docker Compose
  - **Kubernetes**: Orchestrate all your Microservices with Kubernetes
  - **OpenTelemetry**: One Standard - Logs, Traces & Metrics
  - **Micrometer** (Replaces Spring Cloud Sleuth)





Monolith



Microservices

What is a Microservice?

In28 Minutes

Small autonomous services that work together

Sam Newman

What is a Microservice?

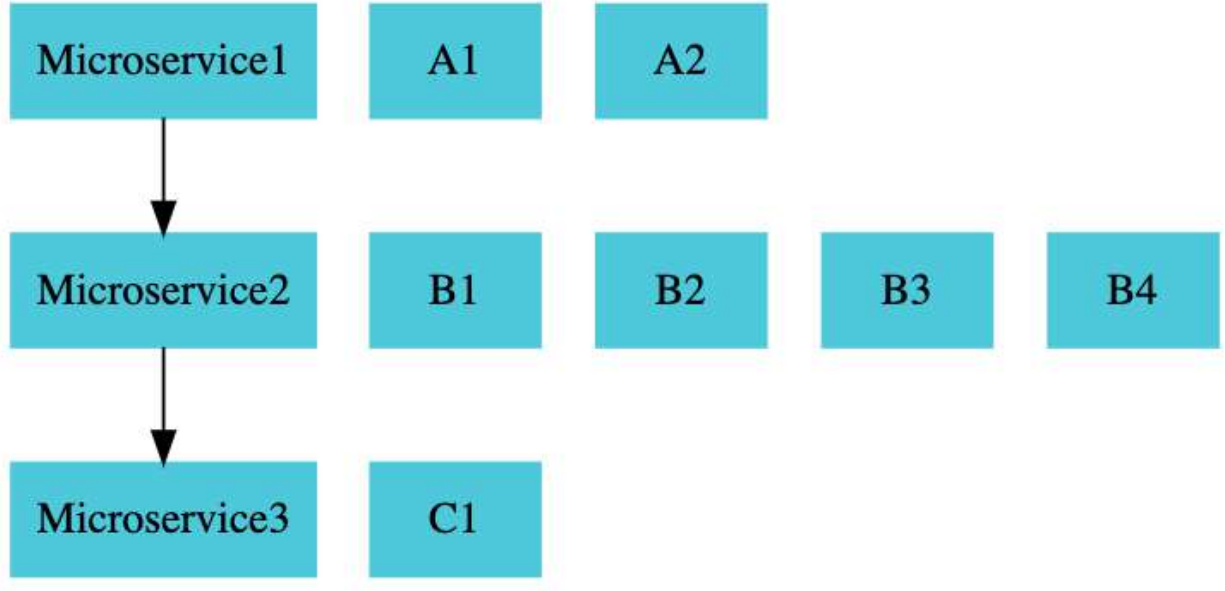
In28 Minutes

Approach to developing a application as a suite of small services, each running in its own process and communicating with lightweight mechanisms often an HTTP resource API.
These services are built around business capabilities and independently deployable by fully automated deployment machinery.
There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

James Lewis and Martin Fowler

Microservices for me

- REST
- Small Well Chosen Deployable Units
- Cloud Enabled



Microservices - Challenges

How do you identify the boundary for each of these microservices?

How do you decide what you should do and what you should not do?

We said we would have five or 50 microservices.
These microservices have multiple instances in each environment,
and there are multiple environments.
So let's say there are 10 microservices with five environments, and let's say 50 instances.
So we are talking about basically tons of configuration and that's a lot of work for the operation team to maintain.

- Bounded Context
- Configuration Management
- Dynamic Scale Up and Scale Down
- Visibility
- Pack of Cards
- Zero Downtime Deployments

The loads on different microservices will be different at different instances of time.
And at particular instance, I might need two instances of microservice two, but later at a different point in time
I might be needing 10 instances of this.
So I should be able to bring the new instances of microservices up and bring down older instances of microservices when they are not really needed.

If I say the functionality is now distributed among 10 microservices and there's a bug,
how do you identify where the bug is?
You need to have a centralized log where I can go and find out what happened for a specific request.
Which microservice caused the problem?
Not just that we also need monitoring around these microservices because we have hundreds of microservices.
We need to be able to identify the microservices which are down.
We would want to be able to automatically identify servers where there is not enough disc space.
All these kind of things need to be automated.
So we need great visibility

Microservices - Challenges

- Bounded Context
- Configuration Management
- Dynamic Scale Up and Scale Down
- Visibility
- Pack of Cards
- Zero Downtime Deployments

you have one microservice calling another, another calling another.
So there would be certain microservices which would be the fundamental for the whole thing.
And if that microservice goes down then the entire application might go down.
So it's like a pack of cards.

they can collapse very easily

therefore it's very important
for you to have fall tolerance in your microservices.

Microservice - Solutions

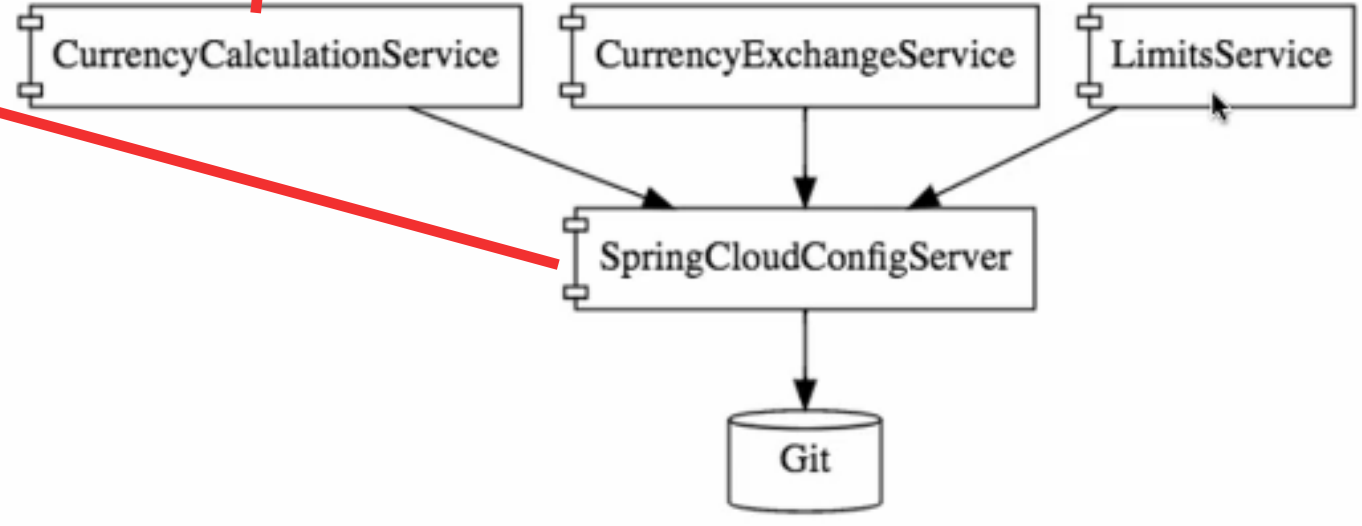
- **Spring Cloud** Umbrella Projects
  - Centralized Configuration Management (Spring Cloud Config Server)
  - Location Transparency - Naming Server (Eureka)
  - Load Distribution (Ribbon, Spring Cloud Load Balancer)
  - Visibility and Monitoring (Zipkin)
  - API Gateway (Zuul, Spring Cloud Gateway)
  - Fault Tolerance (Hystrix, Resilience4j)
- **Docker**: Language Neutral, Cloud Neutral deployable units
- **Kubernetes**: Orchestrate Thousands of Microservices

Config Client SPRING CLOUD CONFIG

Client that connects to a Spring Cloud Config Server to fetch the application's configuration.

Config Server SPRING CLOUD CONFIG

Central management for configuration via Git, SVN, or HashiCorp Vault.



Spring Cloud Config Server

Spring Cloud Config Server provides an approach where you can store all the configuration for all the different environments of all the microservices in a Git repository.
So you can store all the configuration for different environments of different microservicesin just one place, in a centralized location,
and Spring Cloud Config Server
can be used to expose that configuration to all the microservices. This helps us to keep the configuration in one place
and that makes it very easy
to maintain the configuration for all microservices.

DYNAMIC SCALE UP AND DOWN

- Naming Server (Eureka)
- Ribbon (Client Side Load Balancing)
- Feign (Easier REST Clients)

As you can see in the diagram,
there are multiple instances of the CurrencyExchangeService.
And it's possible that at any point in time, new instances can be added in or removed out.
And we would want the CurrencyCalculationService to be able to distribute the load between all the instances of the CurrencyExchangeService.
We would want to be able to dynamically check what are the available instances of the CurrencyExchangeService?
And make sure that their load is distributed among all of them.

In28Minutes

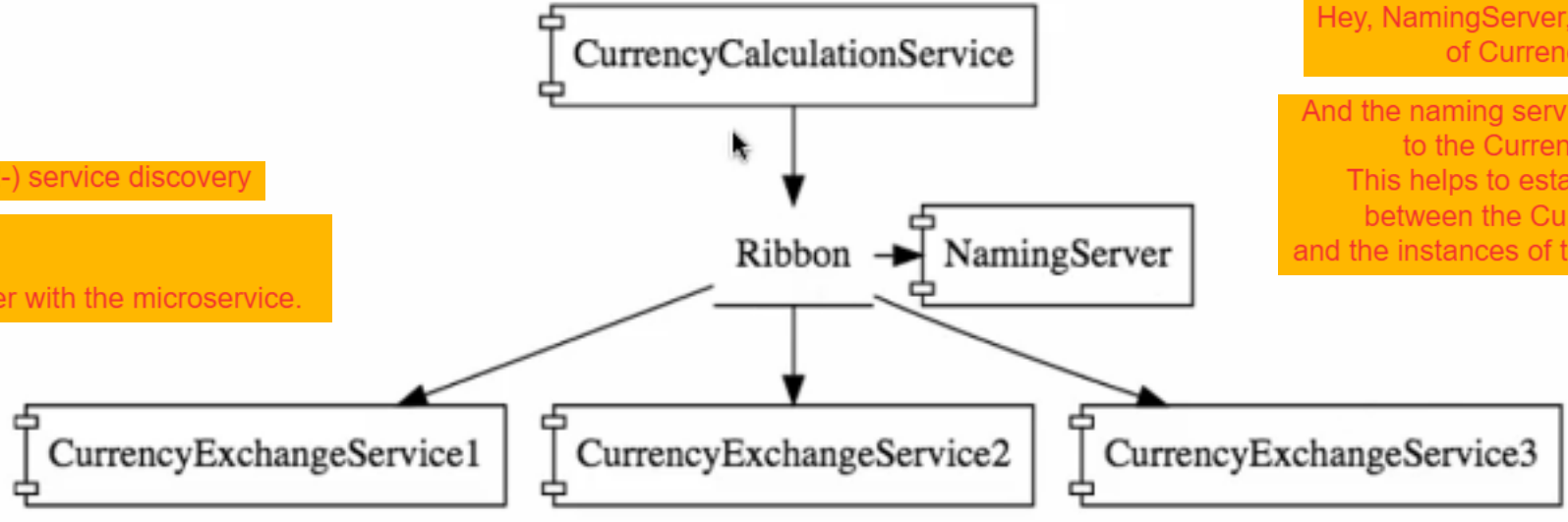
Solution: Eureka (Naming Server)

So all the instances of all microservices
would register with the naming server.

1-) service registration

So all microservices can register with the microservice.

2-) service discovery



the CurrencyCalculationService
can ask the Eureka NamingServer,
Hey, NamingServer, give me the current instances
of CurrencyExchangeService.

And the naming service would provide those URLs
to the CurrencyCalculationService.
This helps to establish dynamic relationship
between the CurrencyCalculationService
and the instances of the CurrencyExchangeService.

Ribbon Load Balancing

We will use Ribbon for client-side load balancing.
That means the CurrencyCalculationService
will host Ribbon.
And it would make sure that the load is evenly distributed
among the existing instances
that it gets from the NamingServer.

We'll also use Feign in the CurrencyCalculationService
as a mechanism to write simple RESTful clients.

we would use Zipkin Distributed Tracing to trace a request across multiple components.
One of the important things about microservices is these microservices have a lot of common features.
For example, logging, security, analytics and things like that.
You don't want to implement all these common features in every microservice.
API Gateways provide great solutions to this kind of challenges.

VISIBILITY AND MONITORING

- Zipkin Distributed Tracing
- Netflix API Gateway

If a service is down,
Hystrix helps us to configure a default response.

In28Minutes

FAULT TOLERANCE

- Hystrix

Microservices - 3 Key Advantages

- New Technology & Process Adoption
- Dynamic Scaling
- Faster Release Cycles

For example, Microservice1 might be Java.
Microservice2 might be Node.js.
Microservice3 might be written in Kotlin
and tomorrow there might be a language xyz
which is really doing well
and which provides a lot of benefits to you.

This means that you can bring new features faster to market
and that's a big advantage to have in the modern world.

Consider an online shopping application like Amazon.
They don't really have the same amount of load
or same amount of traffic, or same amount of users
throughout the year.
Especially during the holiday season,
the load on the applications will be lot
and during the rest of the year
there might not be so much load.
During the Black Friday
there might be a huge amount of load.
If your microservices are cloud enabled
they can scale dynamically and you can procure hardware
and release it dynamically as well.
So you can scale up your applications
and scale them down based on the load.
Because you're developing smaller components
it's much easier to release microservices
compared to monolith applications.

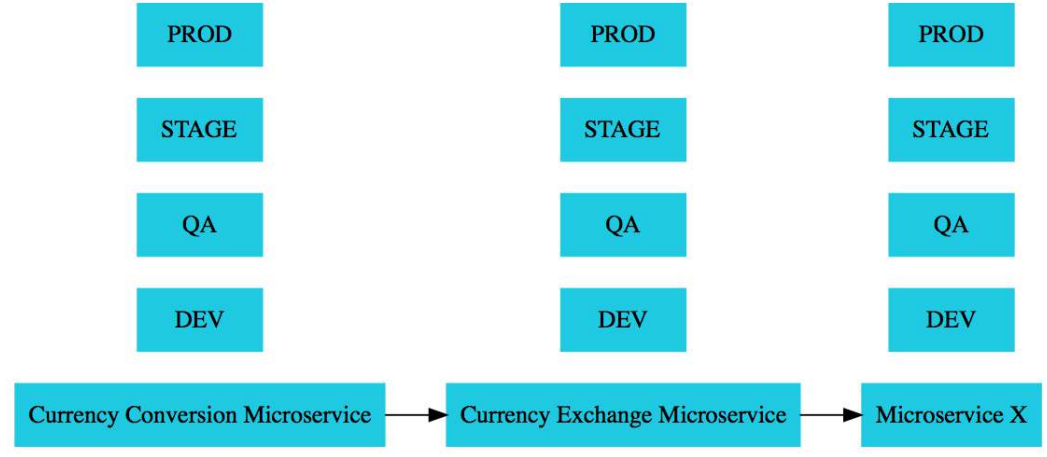
Ports Standardization

In28 Minutes

Application	Port
Limits Microservice	8080, 8081, ...
Spring Cloud Config Server	8888
Currency Exchange Microservice	8000, 8001, 8002, ..
Currency Conversion Microservice	8100, 8101, 8102, ...
Netflix Eureka Naming Server	8761
API Gateway	8765
Zipkin Distributed Tracing Server	9411

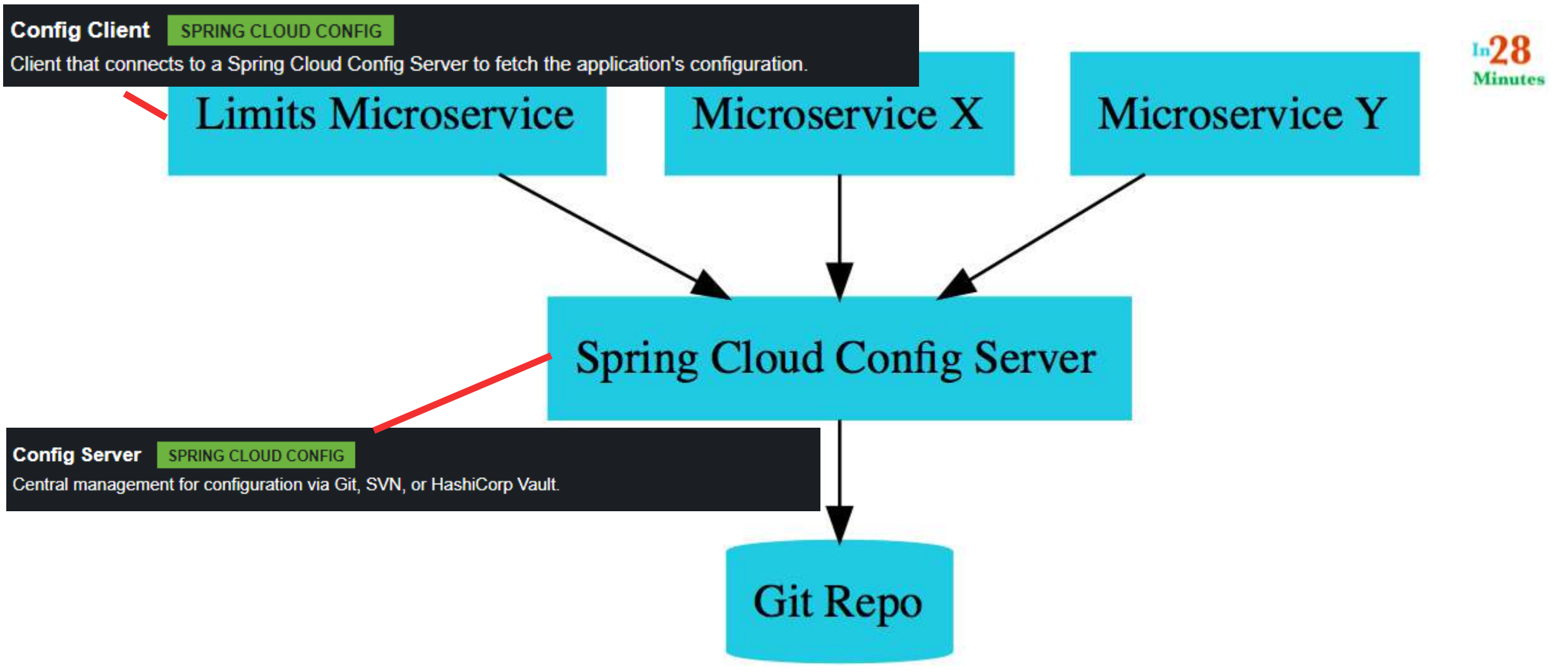
Need for Centralized Configuration

- Lot of configuration:
  - External Services
  - Database
  - Queue
  - Typical Application Configuration
- Configuration variations:
  - 1000s of Microservices
  - Multiple Environments
  - Multiple instances in each Environment
- How do you manage all this configuration?



Config Client SPRING CLOUD CONFIG

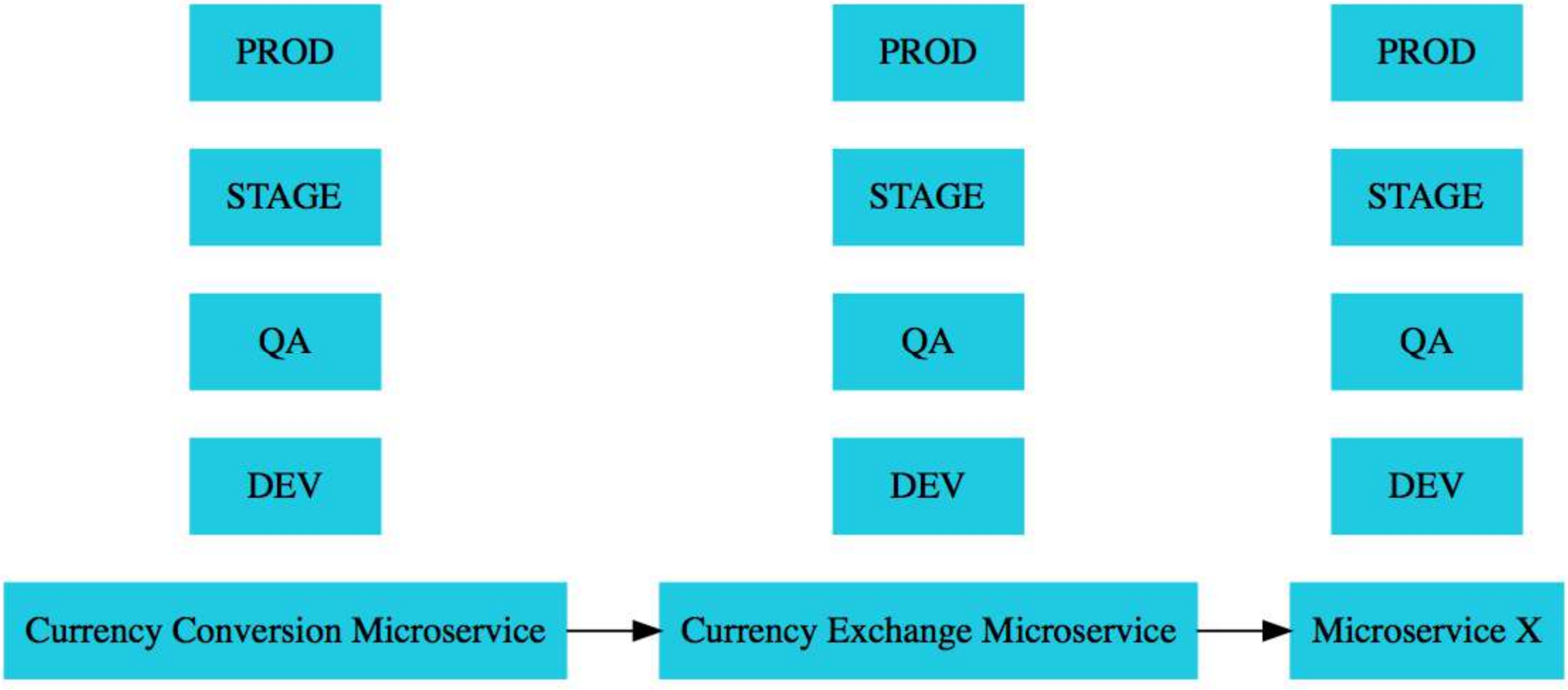
Client that connects to a Spring Cloud Config Server to fetch the application's configuration.



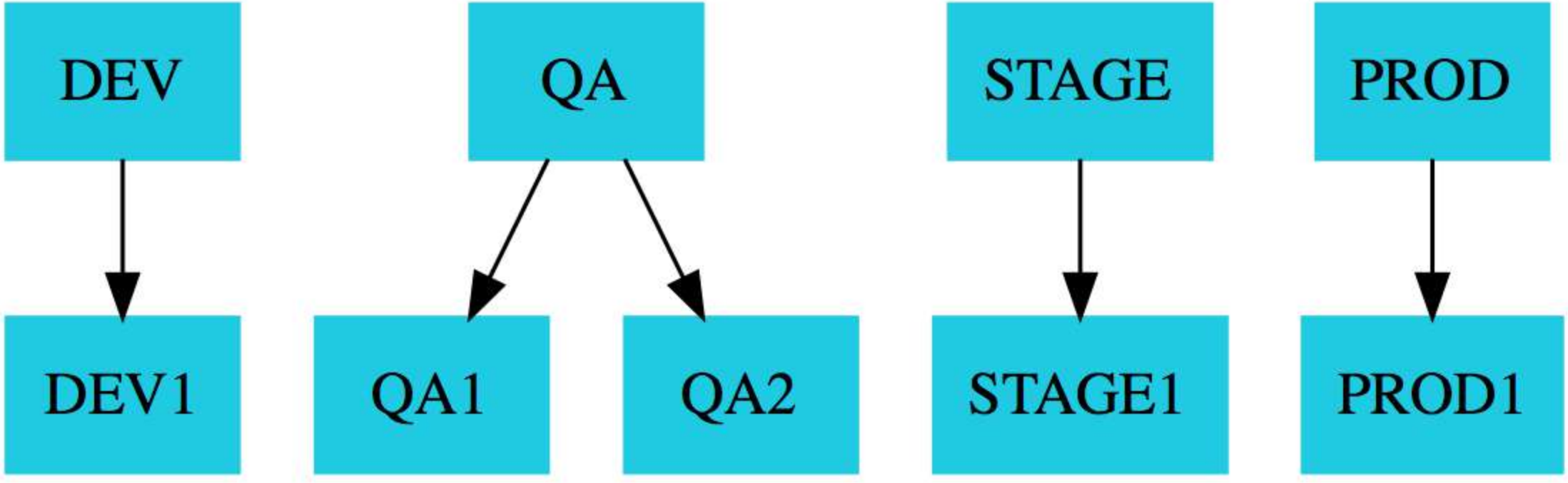
Config Server SPRING CLOUD CONFIG

Central management for configuration via Git, SVN, or HashiCorp Vault.

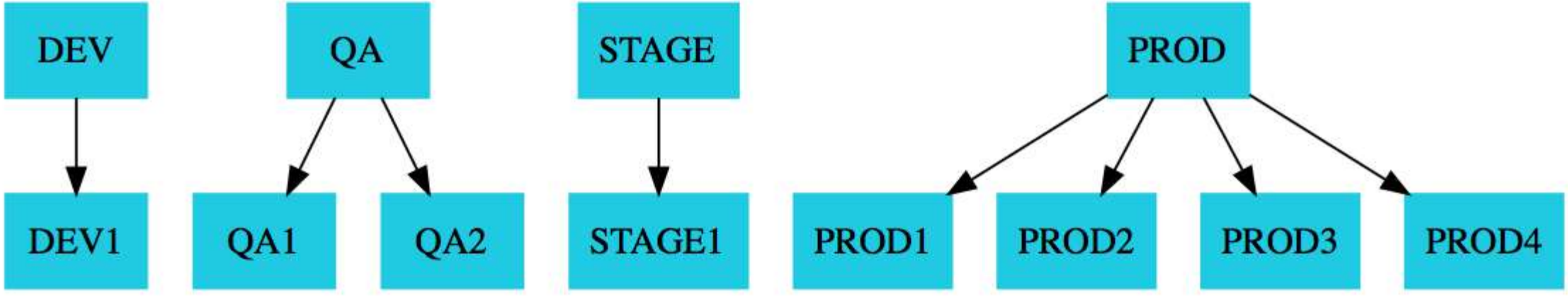
Config Server



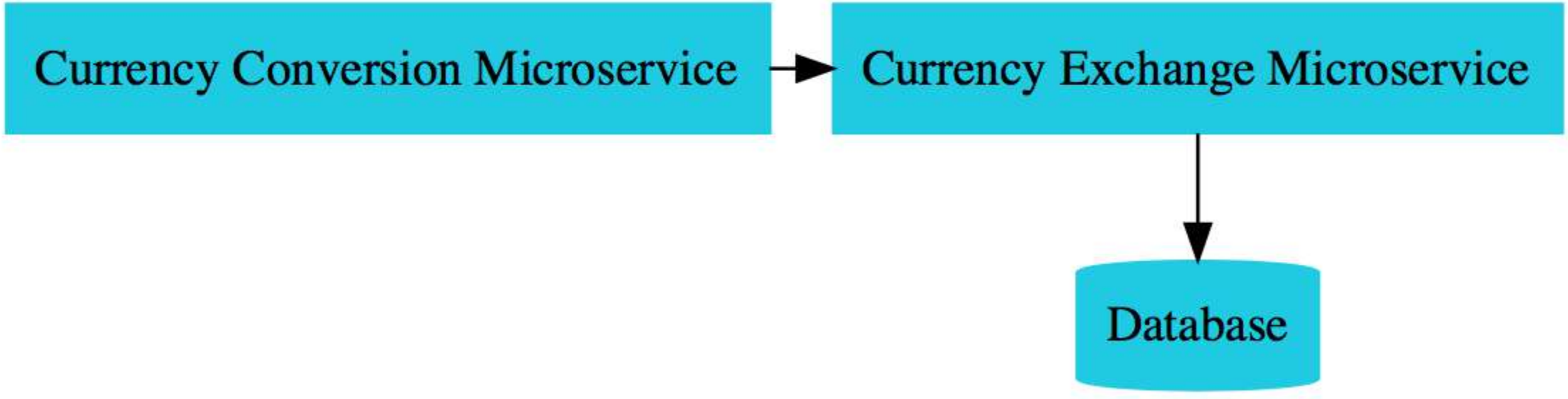
Environments



Environments



Environments



Microservices Overview

Currency Exchange Microservice

In 28 Minutes

What is the exchange rate of one currency in another?

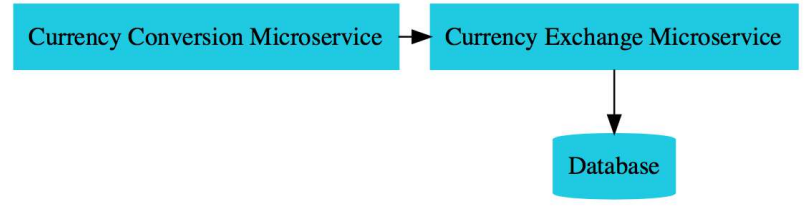
```
http://localhost:8000/currency-exchange/from/USD/to/INR

{
"id":10001,
"from":"USD",
"to":"INR",
"conversionMultiple":65.00,
"environment":"8000 instance-id"
}
```

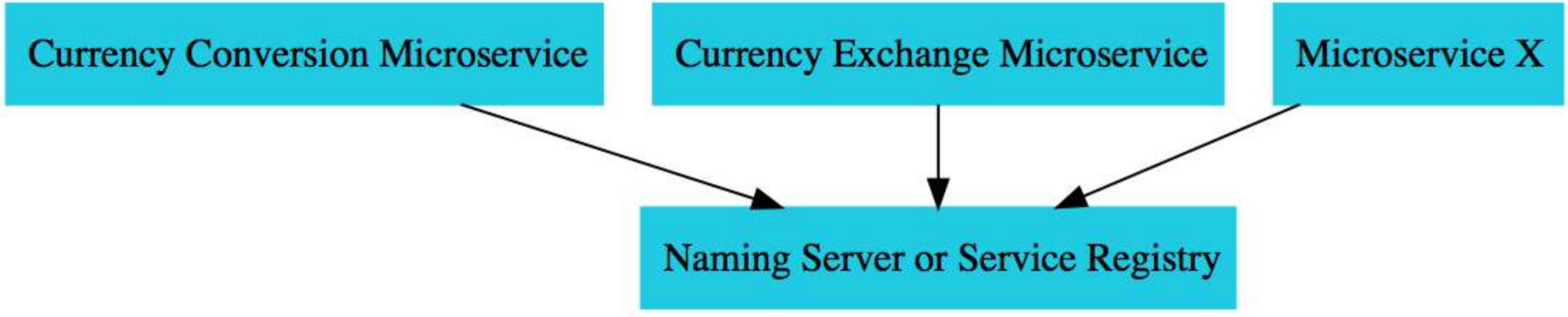
Currency Conversion Microservice

In28 Minutes

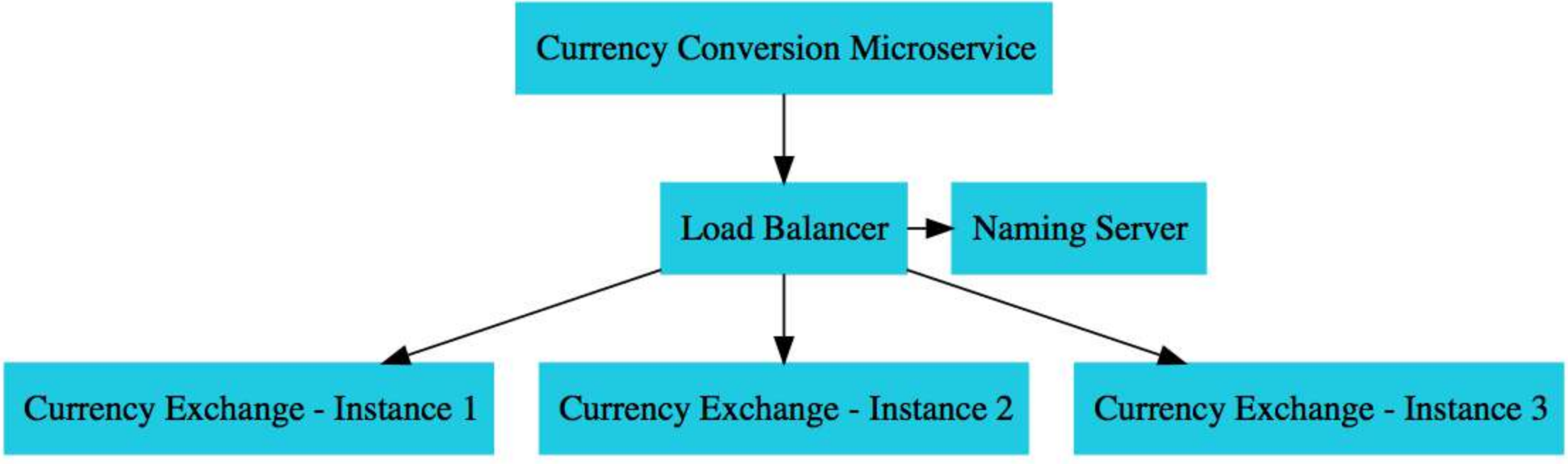
Convert 10 USD into INR

```
http://localhost:8100/currency-conversion/from/USD/to/INR/quantity/10

{
  "id": 10001,
  "from": "USD",
  "to": "INR",
  "conversionMultiple": 65.00,
  "quantity": 10,
  "totalCalculatedAmount": 650.00,
  "environment": "8000 instance-id"
}
```





Naming Server



Load Balancing

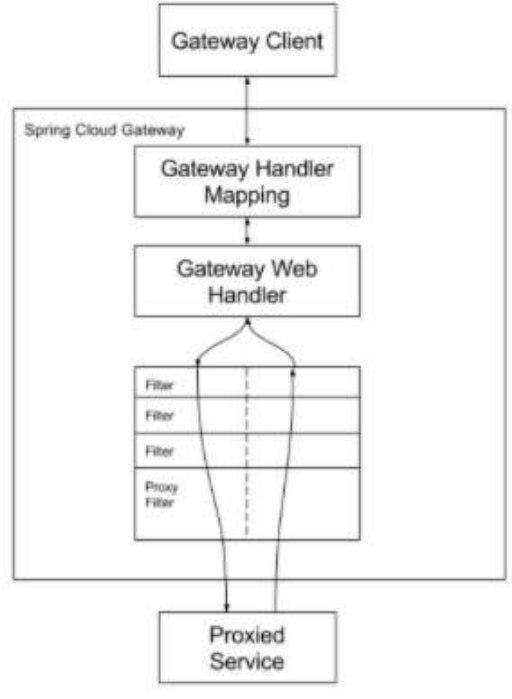
Spring Cloud Gateway

There would be hundreds of microservices like theseand these microservices have a lot of common features like Authentication, authorization, logging, rate limiting.
Solution: API gateway

Zool is not supported by netflix now, Spring Cloud Gateway is now popular!

Minutes

- Simple, yet effective way to route to APIs
- Provide cross cutting concerns:
  - Security
  - Monitoring/metrics
- Built on top of Spring WebFlux (Reactive Approach)
- Features:
  - Match routes on any request attribute
  - Define Predicates and Filters
  - Integrates with Spring Cloud Discovery Client (Load Balancing)
  - Path Rewriting



From *https://docs.spring.io*

Circuit Breaker

There is a complex call chain in a microservices architecture.

- What if one of the services is down or is slow?
  - Impacts entire chain!
- Questions:
  - Can we return a fallback response if a service is down?
  - Can we implement a Circuit Breaker pattern to reduce load?
  - Can we retry requests in case of temporary failures?
  - Can we implement rate limiting?
- Solution: Circuit Breaker Framework - Resilience4j

Can I configure a default response? This might not always be possible. For example, in the case of a credit card transaction, or something of that kind, you do not have any fallback responses possible. But in the case of a shopping application, instead of returning a set of products. You might return a default set of products.

```
Resilience4j - docs
```

```
Resilience4j - docs
```

Setup

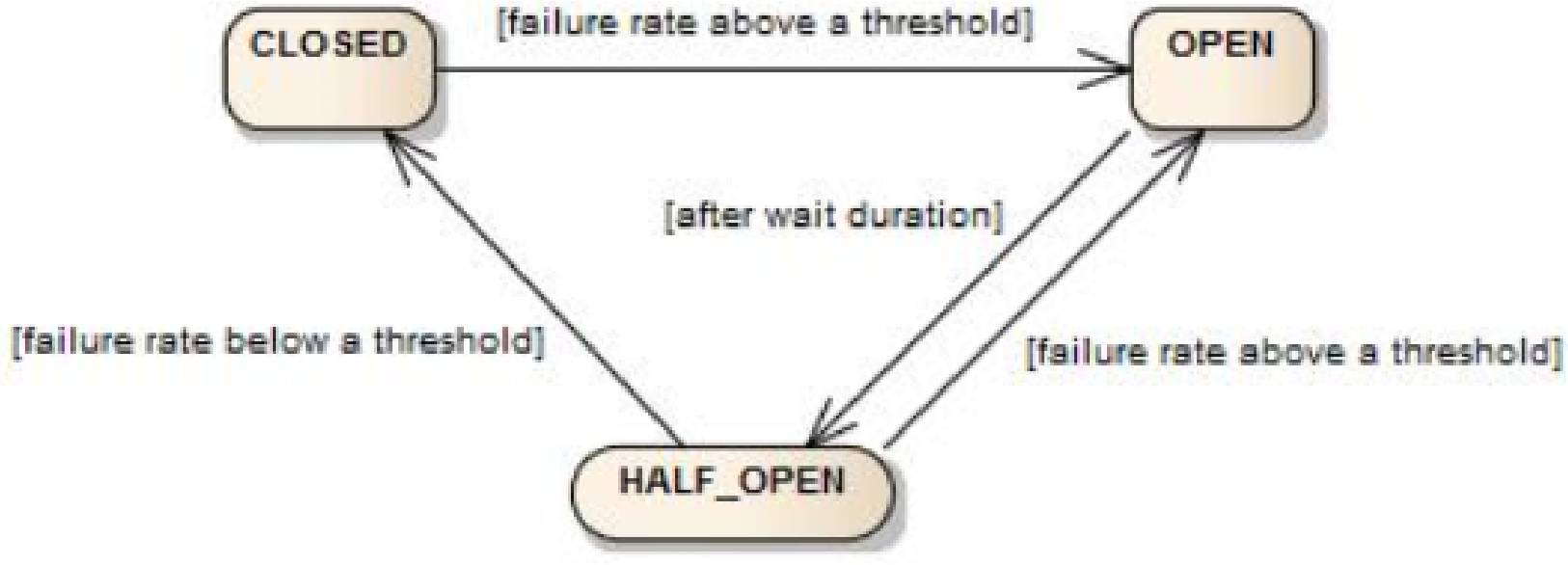
Add the Spring Boot Starter of Resilience4j to your compile dependency.

The module expects that org.springframework.boot:spring-boot-starter-actuatorand org.springframework.boot:spring-boot-starter-aopare already provided at runtime.

If you are using webflux withSpring Boot 2 or SpringBoot 3, you also need io.github.resilience4j:resilience4j-reactor

Closed is when I am calling the dependent microservice continuously.

In a open state, the circuit breaker will not call the dependent microservice. It'll directly return the fallback response.

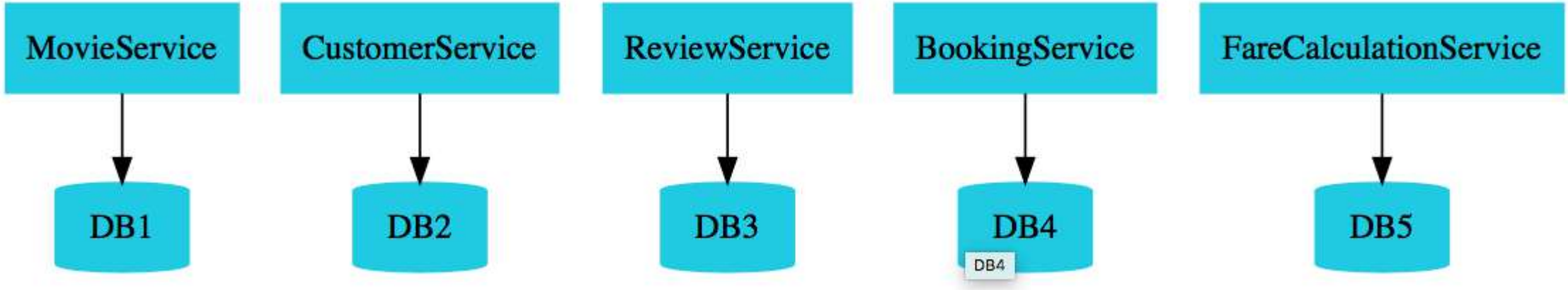


In a half open statea circuit secure breaker would be sendinga percentage of requests to the dependent microservice, and for rest of the requests, it would return the hard coded responseor the fallback response back.

```
2024-08-18T10:54:38.978+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-1] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:39.150+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-2] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:39.322+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-3] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:39.494+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-4] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:39.729+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-5] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:39.932+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-6] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:40.120+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-7] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:40.307+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-8] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:40.463+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-9] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:40.636+03:00  INFO 14448 --- [currency-exchange-server] [io-8001-exec-10] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:40.854+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-1] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:41.076+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-2] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:41.233+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-3] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:41.436+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-4] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:41.624+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-5] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:41.796+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-6] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:41.975+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-7] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:42.163+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-8] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:42.334+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-9] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:54:42.507+03:00  INFO 14448 --- [currency-exchange-server] [io-8001-exec-10] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:55:42.604+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-5] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:55:42.775+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-6] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:55:42.947+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-7] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:55:43.120+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-8] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:55:43.308+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-9] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:55:43.464+03:00  INFO 14448 --- [currency-exchange-server] [io-8001-exec-10] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:55:43.636+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-1] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:55:43.793+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-2] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:55:43.965+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-3] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:55:44.137+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-4] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:56:44.206+03:00  INFO 14448 --- [currency-exchange-server] [io-8001-exec-10] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:56:44.378+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-1] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:56:44.550+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-2] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:56:44.722+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-3] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:56:44.894+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-4] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:56:45.066+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-5] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:56:45.238+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-6] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:56:45.457+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-7] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:56:45.691+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-8] c.i.m.c.CircuitBreakerController       : Sample API call received
2024-08-18T10:56:45.957+03:00  INFO 14448 --- [currency-exchange-server] [nio-8001-exec-9] c.i.m.c.CircuitBreakerController       : Sample API call received
```

The circuit breaker accepts requests for a while and then it stops accepting requests.

Then, it accepts 10 request each minute.

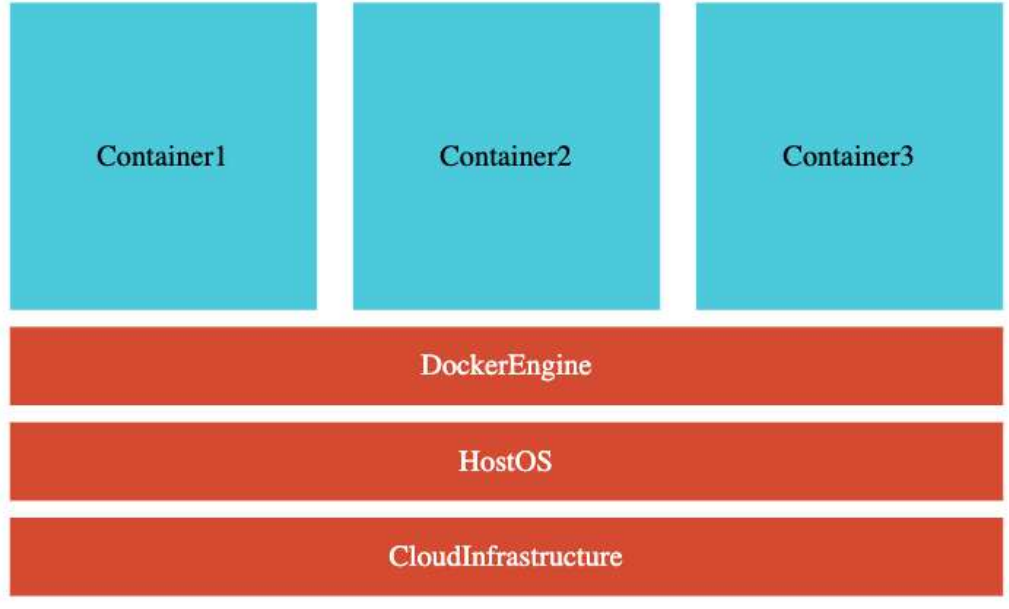


- Enterprises are heading towards microservices architectures
  - Build small focused microservices
  - **Flexibility to innovate** and build applications in different programming languages (Go, Java, Python, JavaScript, etc)
  - BUT **deployments become complex**!
  - How can we have **one way of deploying** Go, Java, Python or JavaScript .. microservices?
    - Enter **containers**!

Docker

In28 Minutes

- Create **Docker images** for each microservice
- Docker image **contains everything a microservice needs** to run:
  - Application Runtime (JDK or Python or NodeJS)
  - Application code
  - Dependencies
- You can run these docker containers **the same way** on any infrastructure
  - Your local machine
  - Corporate data center
  - Cloud



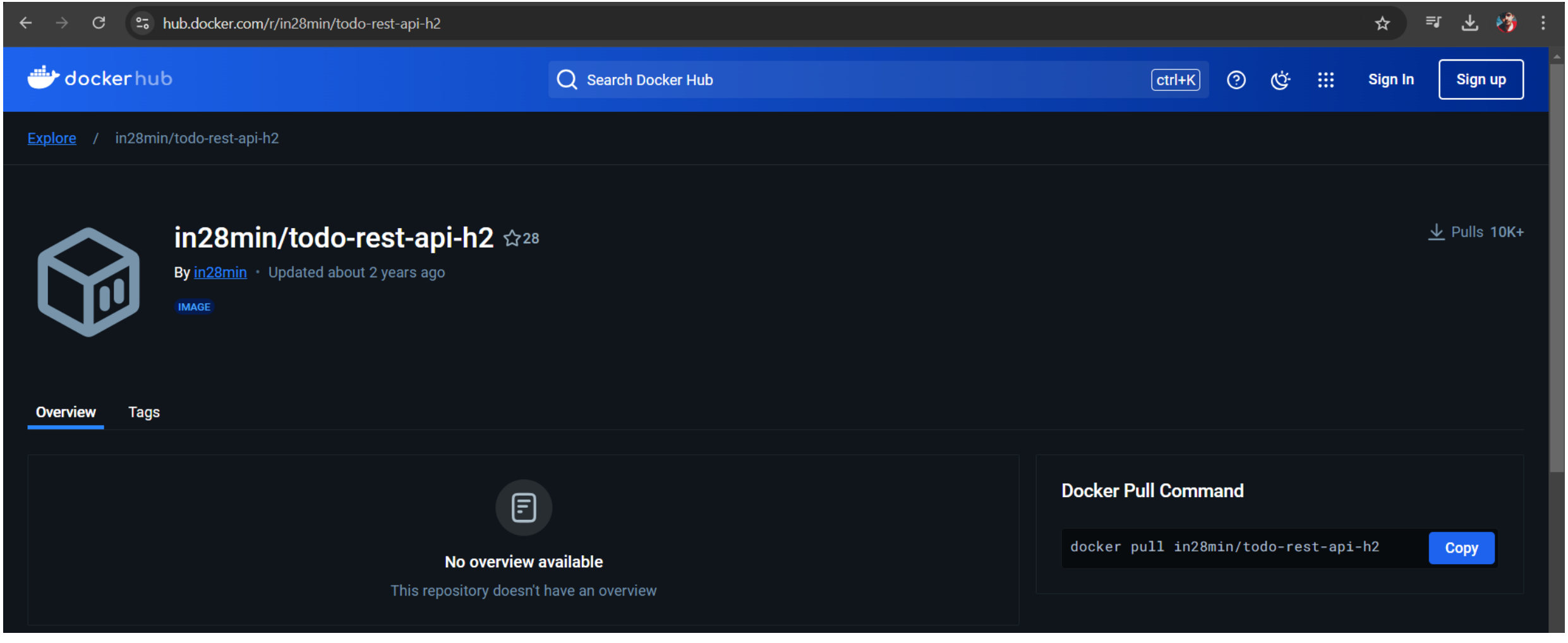
1-You must open docker desktop firstly.

2-docker run in28min/todo-rest-api-h2:1.0.0.RELEASE

```
PS C:\Users\Unfaithful> docker run in28min/todo-rest-api-h2:1.0.0.RELEASE
Unable to find image 'in28min/todo-rest-api-h2:1.0.0.RELEASE' locally
1.0.0.RELEASE: Pulling from in28min/todo-rest-api-h2
e7c96db7181b: Pull complete
f910a506b6cb: Pull complete
c2274a1a0e27: Pull complete
80559632f507: Pull complete
Digest: sha256:b50038f47a4ed68180ecf8d8efc8ec3ec25e97dd4232020ef03574384c48a23c
Status: Downloaded newer image for in28min/todo-rest-api-h2:1.0.0.RELEASE

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.0.RELEASE)

2024-08-18 19:54:13.538  INFO 1 --- [           main] c.i.r.w.r.RestfulWebServicesApplication  : Starting RestfulWebServicesApplication v1.0.0.RELEASE on d5
bd2f1ecc06 with PID 1 (/app.jar started by root in /)
2024-08-18 19:54:13.544  INFO 1 --- [           main] c.i.r.w.r.RestfulWebServicesApplication  : No active profile set, falling back to default profiles: de
fault
2024-08-18 19:54:14.754  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data repositories in DEFAULT mode.
2024-08-18 19:54:14.841  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 77ms. Found 1 r
epository interfaces.
2024-08-18 19:54:15.375  INFO 1 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.transaction.annotation.ProxyTrans
actionManagementConfiguration' of type [org.springframework.transaction.annotation.ProxyTransactionManagementConfiguration$$EnhancerBySpringCGLIB$$eef34fbc]
 is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
```

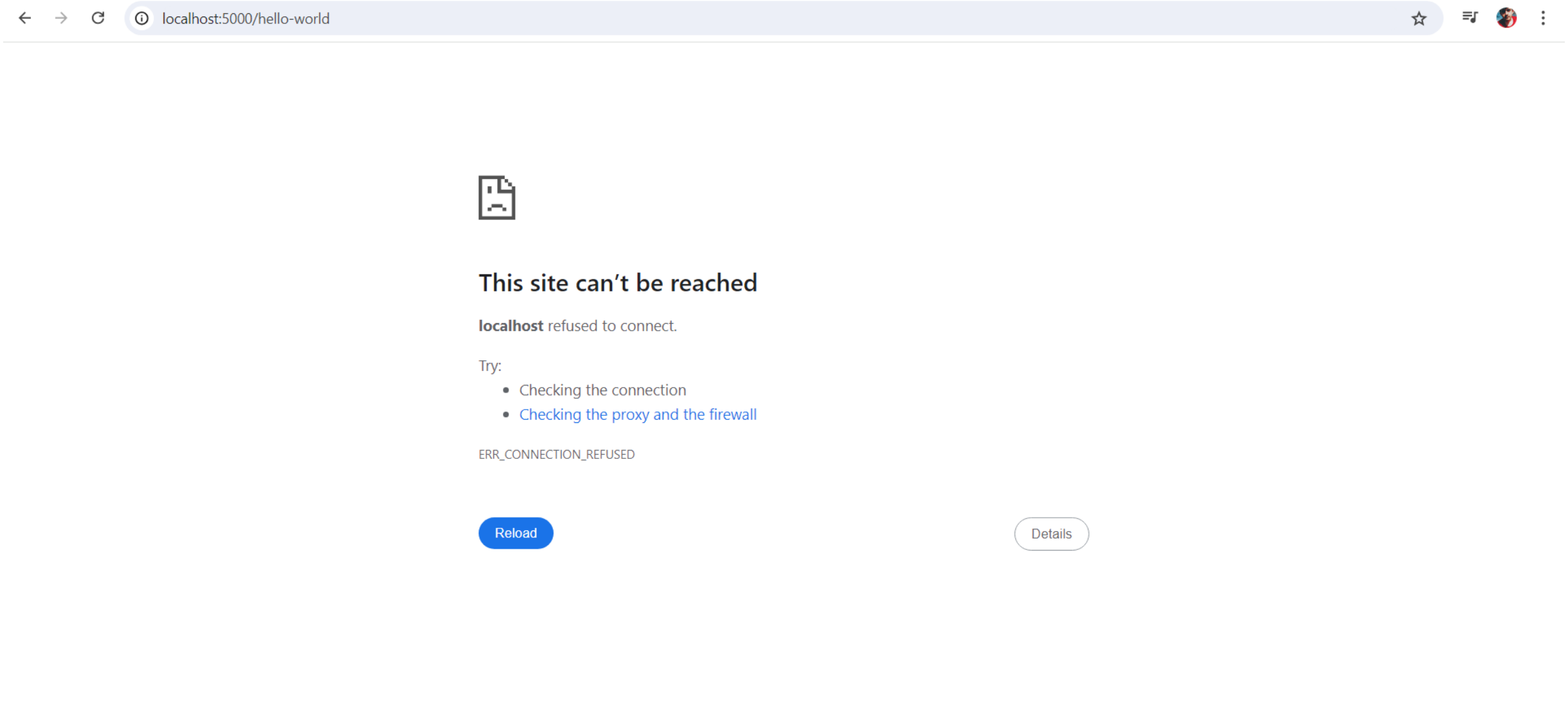
hub.docker.com → Docker Registry

https://hub.docker.com/r/in28min/todo-rest-api-h2 → Repository



```
docker run in28min/todo-rest-api-h2:1.0.0.RELEASE
```

```
http://localhost:5000/hello-world
```

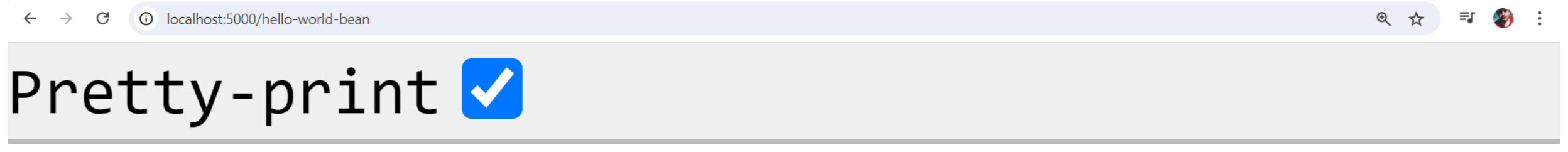


-p 5000:5000 → -p {HostPort}:{ContainerPort}

```
docker run -p 5000:5000 in28min/todo-rest-api-h2:1.0.0.RELEASE
```

http://localhost:5000/hello-world

Hello World



Pretty-print ☑


```
{
  "message": "Hello World"
}
```

Docker Concepts

Tag → In Docker, a tag is a label used to identify a specific version of a Docker image. It helps you specify and manage different versions of the same image.

Docker Image → A Docker image is a blueprint or template. It contains everything needed to run a piece of software: the application code, libraries, environment variables, and configuration files.

Docker Container → A Docker container is a running instance of a Docker image. It's like a virtual environment that runs your application based on the image you've created.

docker run -p 5000:5000 -d in28min/todo-rest-api-h2:1.0.0.RELEASE

detached mode: Docker can run your container in detached mode in the background. To do this, you can use the --detach or -d for short.

work in background if you make ctrl + c in terminal or close terminal

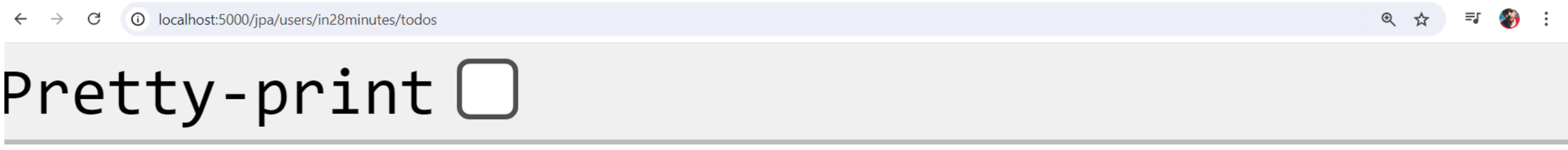
```
PS C:\Users\Unfaithful> docker run -p 5000:5000 -d in28min/todo-rest-api-h2:1.0.0.RELEASE
680cbc81ab5c3d00efc6323eb4078374ca1474a9a0a8155621e9dba3d1d2df17
PS C:\Users\Unfaithful> docker logs 680cbc81ab5c3d00efc6323eb4078374ca1474a9a0a8155621e9dba3d1d2df17


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.0.RELEASE)

2024-08-19 10:07:53.630  INFO 1 --- [           main] c.i.r.w.r.RestfulWebServicesApplication  : Starting RestfulWebServicesApplication v1.0.0.RELEASE on 68
0cbc81ab5c with PID 1 (/app.jar started by root in /)
2024-08-19 10:07:53.635  INFO 1 --- [           main] c.i.r.w.r.RestfulWebServicesApplication  : No active profile set, falling back to default profiles: de
fault
2024-08-19 10:07:54.751  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data repositories in DEFAULT mode.
2024-08-19 10:07:54.844  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 79ms. Found 1 r
epository interfaces.
2024-08-19 10:07:55.376  INFO 1 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.transaction.annotation.ProxyTrans
actionManagementConfiguration' of type [org.springframework.transaction.annotation.ProxyTransactionManagementConfiguration$$EnhancerBySpringCGLIB$$e7f4a697]
 is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
2024-08-19 10:07:56.035  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 5000 (http)
2024-08-19 10:07:56.077  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-08-19 10:07:56.077  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache Tomcat/9.0.12
2024-08-19 10:07:56.095  INFO 1 --- [           main] o.a.catalina.core.AprLifecycleListener   : The APR based Apache Tomcat Native library which allows opt
imal performance in production environments was not found on the java.library.path: [/usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/server:/usr/lib/jvm/java-1.
8-openjdk/jre/lib/amd64:/usr/lib/jvm/java-1.8-openjdk/jre/../lib/amd64:/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib]
2024-08-19 10:07:56.236  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-08-19 10:07:56.237  INFO 1 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 253
6 ms
2024-08-19 10:07:56.286  INFO 1 --- [           main] o.s.b.w.servlet.ServletRegistrationBean  : Servlet dispatcherServlet mapped to [/]
2024-08-19 10:07:56.289  INFO 1 --- [           main] o.s.b.w.servlet.ServletRegistrationBean  : Servlet webServlet mapped to [/h2-console/*]
2024-08-19 10:07:56.294  INFO 1 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'characterEncodingFilter' to: [/*]
2024-08-19 10:07:56.295  INFO 1 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
2024-08-19 10:07:56.295  INFO 1 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'formContentFilter' to: [/*]
2024-08-19 10:07:56.295  INFO 1 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to: [/*]
2024-08-19 10:07:56.660  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-08-19 10:07:56.923  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2024-08-19 10:07:57.019  INFO 1 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [
        name: default
```

```
docker logs -f 680cbc81ab5c3d00efc6323eb4078374ca1474a9a0a8155621e9dba3d1d2df17
```

↓

start tailing the logs (For exit tailing the logs → ctrl + c)

http://localhost:5000/jpa/users/in28minutes/todos



[]

```
 running for 8.245)
2024-08-19 10:18:19.830  INFO 1 --- [nio-5000-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2024-08-19 10:18:19.830  INFO 1 --- [nio-5000-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2024-08-19 10:18:19.848  INFO 1 --- [nio-5000-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 17 ms
2024-08-19 10:18:19.987  INFO 1 --- [nio-5000-exec-1] o.h.h.i.QueryTranslatorFactoryInitiator  : HHH000397: Using ASTQueryTranslatorFactory
Hibernate: select todo0_.id as id1_0_, todo0_.description as descript2_0_, todo0_.is_done as is_done3_0_, todo0_.target_date as target_d4_0_, todo0_.usernam
e as username5_0_ from todo todo0_ where todo0_.username=?
```


You'd be able to see the query in here, in the log.

docker container ls → Running Containers

NOTE: You can write first 5 instead of all!

```
PS C:\Users\Unfaithful> docker container ls
CONTAINER ID   IMAGE                                          COMMAND                  CREATED          STATUS          PORTS                    NAMES
ae4232758b25   in28min/todo-rest-api-h2:1.0.0.RELEASE         "sh -c 'java $JAVA_O…"   2 seconds ago    Up 2 seconds    0.0.0.0:5001->5000/tcp   competent_solomon
680cbc81ab5c   in28min/todo-rest-api-h2:1.0.0.RELEASE         "sh -c 'java $JAVA_O…"   19 minutes ago   Up 19 minutes   0.0.0.0:5000->5000/tcp   inspiring_diffie
PS C:\Users\Unfaithful> docker logs -f 680cb


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.0.RELEASE)

2024-08-19 10:07:53.630  INFO 1 --- [           main] c.i.r.w.r.RestfulWebServicesApplication  : Starting RestfulWebServicesApplication v1.0.0.RELEASE on 680cbc81ab5c wit
h PID 1 (/app.jar started by root in /)
2024-08-19 10:07:53.635  INFO 1 --- [           main] c.i.r.w.r.RestfulWebServicesApplication  : No active profile set, falling back to default profiles: default
2024-08-19 10:07:54.751  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data repositories in DEFAULT mode.
2024-08-19 10:07:54.844  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 79ms. Found 1 repository inte
rfaces.
2024-08-19 10:07:55.376  INFO 1 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.transaction.annotation.ProxyTransactionManageme
ntConfiguration' of type [org.springframework.transaction.annotation.ProxyTransactionManagementConfiguration$$EnhancerBySpringCGLIB$$e7f4a697] is not eligible for getting
 processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
2024-08-19 10:07:56.035  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 5000 (http)
2024-08-19 10:07:56.077  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-08-19 10:07:56.077  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache Tomcat/9.0.12
2024-08-19 10:07:56.095  INFO 1 --- [           main] o.a.catalina.core.AprLifecycleListener   : The APR based Apache Tomcat Native library which allows optimal performan
ce in production environments was not found on the java.library.path: [/usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/server:/usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64:/usr
/lib/jvm/java-1.8-openjdk/jre/../lib/amd64:/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib]
2024-08-19 10:07:56.236  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-08-19 10:07:56.237  INFO 1 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 2536 ms
2024-08-19 10:07:56.286  INFO 1 --- [           main] o.s.b.w.servlet.ServletRegistrationBean  : Servlet dispatcherServlet mapped to [/]
2024-08-19 10:07:56.289  INFO 1 --- [           main] o.s.b.w.servlet.ServletRegistrationBean  : Servlet webServlet mapped to [/h2-console/*]
2024-08-19 10:07:56.294  INFO 1 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'characterEncodingFilter' to: [/*]
2024-08-19 10:07:56.295  INFO 1 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
2024-08-19 10:07:56.295  INFO 1 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'formContentFilter' to: [/*]
2024-08-19 10:07:56.295  INFO 1 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to: [/*]
2024-08-19 10:07:56.660  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-08-19 10:07:56.923  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2024-08-19 10:07:57.019  INFO 1 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [
```

docker images (local)

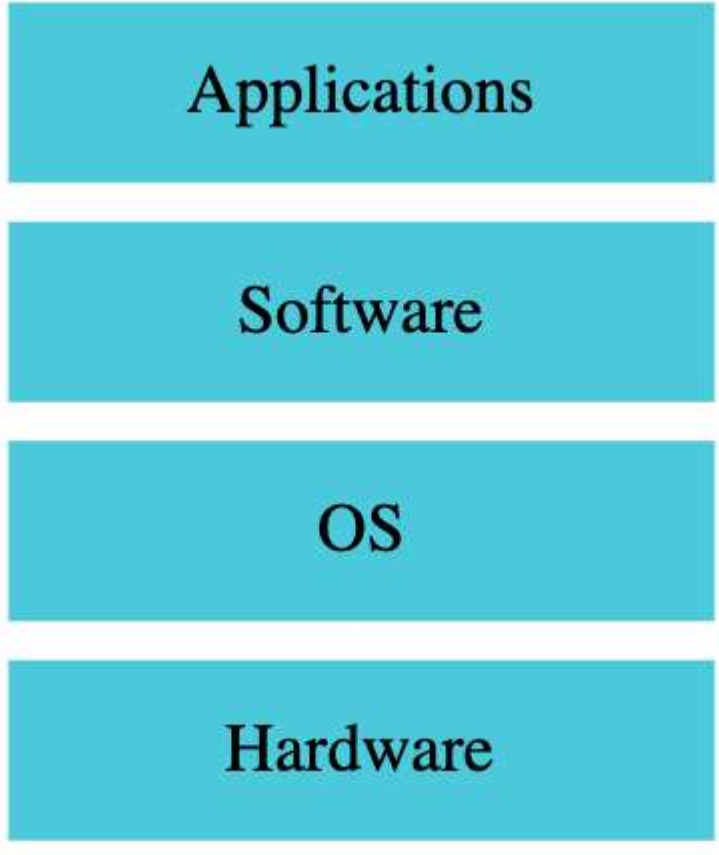
```
PS C:\Users\Unfaithful> docker images
REPOSITORY                TAG             IMAGE ID       CREATED       SIZE
in28min/todo-rest-api-h2  1.0.0.RELEASE   9d05dd98f4a4   2 years ago   143MB
PS C:\Users\Unfaithful>
```


docker container ls -a (all containers → running + stopped)

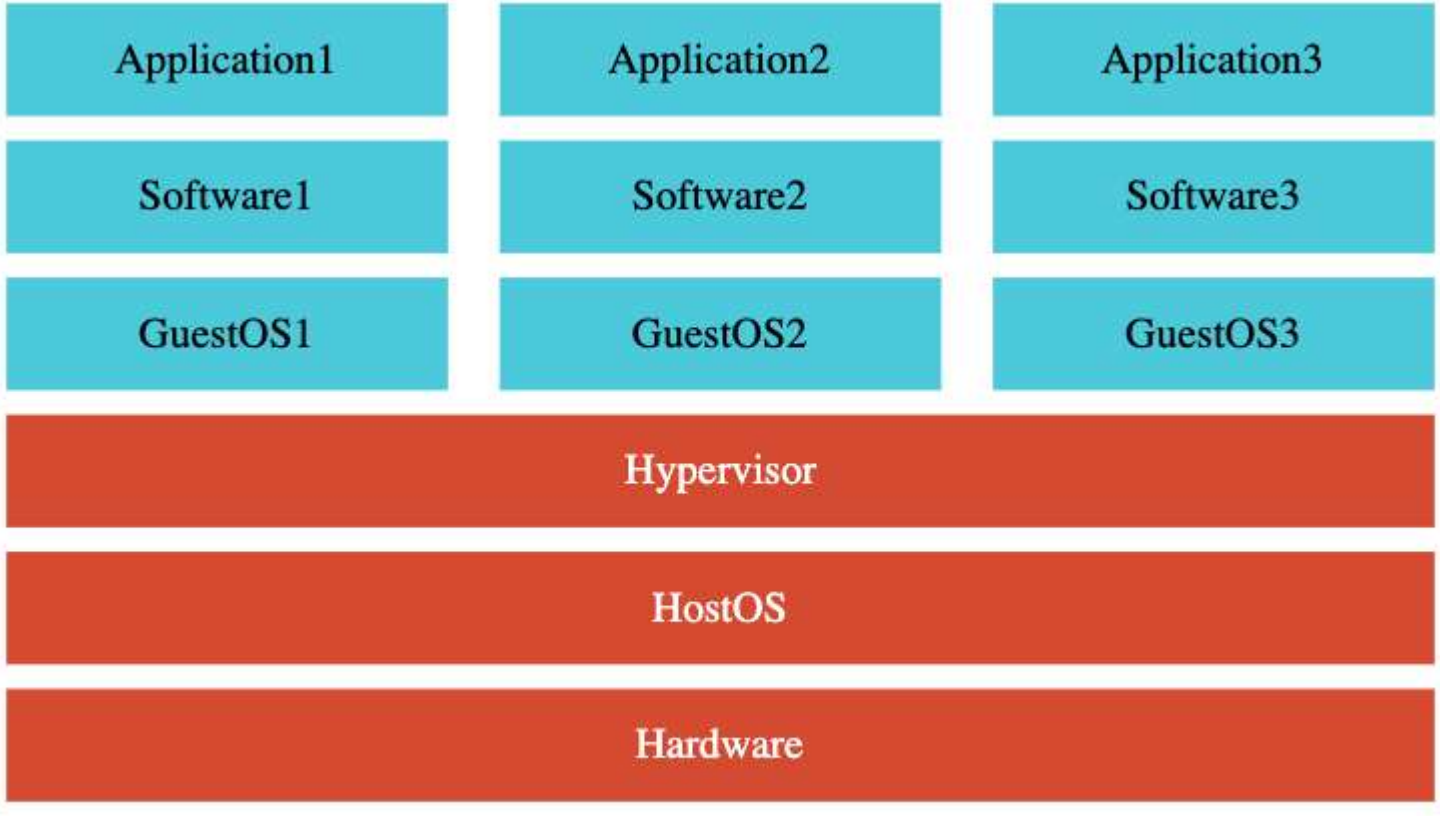
```
PS C:\Users\Unfaithful> docker container ls -a
CONTAINER ID   IMAGE                                      COMMAND                  CREATED          STATUS                          PORTS                    NAMES
ae4232758b25   in28min/todo-rest-api-h2:1.0.0.RELEASE     "sh -c 'java $JAVA_O…"   19 minutes ago   Up 19 minutes                   0.0.0.0:5001->5000/tcp   competent_solomon
680cbc81ab5c   in28min/todo-rest-api-h2:1.0.0.RELEASE     "sh -c 'java $JAVA_O…"   39 minutes ago   Up 39 minutes                   0.0.0.0:5000->5000/tcp   inspiring_diffie
fd9ee4ba4d04   in28min/todo-rest-api-h2:1.0.0.RELEASE     "sh -c 'java $JAVA_O…"   2 hours ago      Exited (130) 52 minutes ago                              peaceful_lehmann
d5bd2f1ecc06   in28min/todo-rest-api-h2:1.0.0.RELEASE     "sh -c 'java $JAVA_O…"   15 hours ago     Exited (130) 2 hours ago                                 affectionate_pros
kuriakova
PS C:\Users\Unfaithful>
```

docker container stop ae423

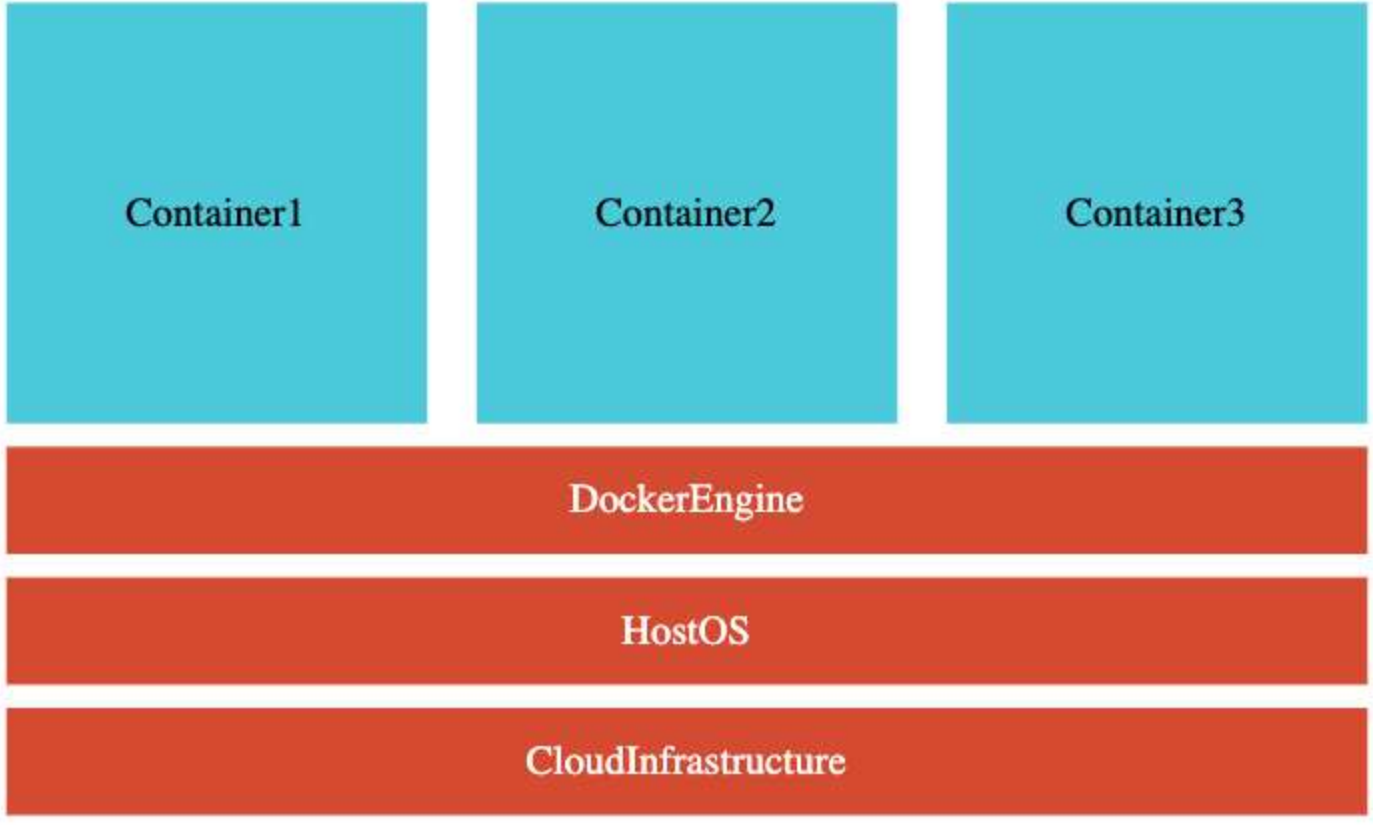
```
PS C:\Users\Unfaithful> docker container ls -a
CONTAINER ID   IMAGE                                          COMMAND                  CREATED          STATUS                          PORTS                    NAMES
ae4232758b25   in28min/todo-rest-api-h2:1.0.0.RELEASE   "sh -c 'java $JAVA_O…"   19 minutes ago   Up 19 minutes                   0.0.0.0:5001->5000/tcp   competent_solomon
680cbc81ab5c   in28min/todo-rest-api-h2:1.0.0.RELEASE   "sh -c 'java $JAVA_O…"   39 minutes ago   Up 39 minutes                   0.0.0.0:5000->5000/tcp   inspiring_diffie
fd9ee4ba4d04   in28min/todo-rest-api-h2:1.0.0.RELEASE   "sh -c 'java $JAVA_O…"   2 hours ago      Exited (130) 52 minutes ago                              peaceful_lehmann
d5bd2f1ecc06   in28min/todo-rest-api-h2:1.0.0.RELEASE   "sh -c 'java $JAVA_O…"   15 hours ago     Exited (130) 2 hours ago                                 affectionate_pros
kuriakova
PS C:\Users\Unfaithful> docker container stop ae423
ae423
PS C:\Users\Unfaithful> docker container stop 680cb
680cb
PS C:\Users\Unfaithful> docker container ls -a
CONTAINER ID   IMAGE                                          COMMAND                  CREATED          STATUS                          PORTS     NAMES
ae4232758b25   in28min/todo-rest-api-h2:1.0.0.RELEASE   "sh -c 'java $JAVA_O…"   23 minutes ago   Exited (143) 33 seconds ago               competent_solomon
680cbc81ab5c   in28min/todo-rest-api-h2:1.0.0.RELEASE   "sh -c 'java $JAVA_O…"   42 minutes ago   Exited (143) 21 seconds ago               inspiring_diffie
fd9ee4ba4d04   in28min/todo-rest-api-h2:1.0.0.RELEASE   "sh -c 'java $JAVA_O…"   3 hours ago      Exited (130) 56 minutes ago               peaceful_lehmann
d5bd2f1ecc06   in28min/todo-rest-api-h2:1.0.0.RELEASE   "sh -c 'java $JAVA_O…"   15 hours ago     Exited (130) 3 hours ago                  affectionate_proskuriakova
PS C:\Users\Unfaithful>
```



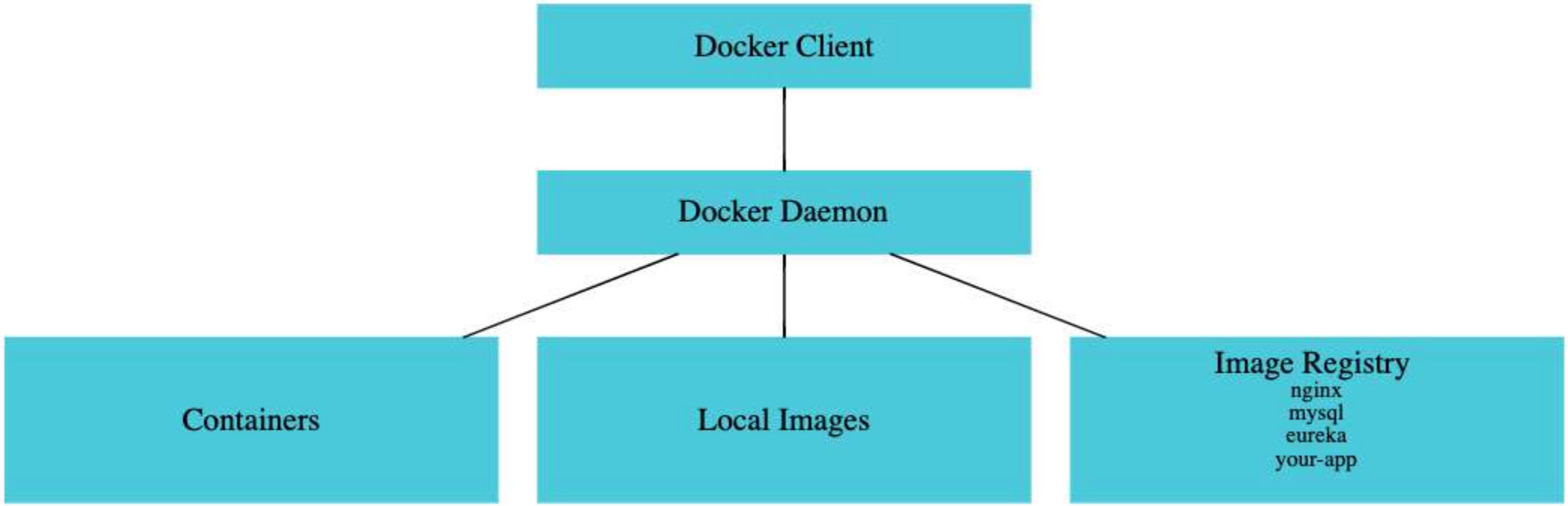
Traditional Deployment



Deployments using Virtual Machines



Deployments using Docker



Docker Architecture

The Docker Daemon is responsible for managing the containers.
It's responsible for managing the locally images and
it is responsible for pulling something from the image registry
if something is not available on our local or pushing a locally
created image to a image registry.

Docker Image Notes:

create a image with different tag

docker tag in28min/todo-rest-api-h2:1.0.0.RELEASE in28min/todo-rest-api-h2:latest

```
PS C:\Users\Unfaithful> docker images
REPOSITORY                TAG              IMAGE ID       CREATED       SIZE
in28min/todo-rest-api-h2  1.0.0.RELEASE    9d05dd98f4a4   2 years ago   143MB
in28min/todo-rest-api-h2  0.0.1-SNAPSHOT   40ee0c4765e0   5 years ago   143MB
PS C:\Users\Unfaithful> docker tag in28min/todo-rest-api-h2:1.0.0.RELEASE in28min/todo-rest-api-h2:latest
PS C:\Users\Unfaithful> docker images
REPOSITORY                TAG              IMAGE ID       CREATED       SIZE
in28min/todo-rest-api-h2  1.0.0.RELEASE    9d05dd98f4a4   2 years ago   143MB
in28min/todo-rest-api-h2  latest           9d05dd98f4a4   2 years ago   143MB
in28min/todo-rest-api-h2  0.0.1-SNAPSHOT   40ee0c4765e0   5 years ago   143MB
PS C:\Users\Unfaithful>
```

docker search mysql ⟶ check image is official or not

```
PS C:\Users\Unfaithful> docker search mysql
NAME                            DESCRIPTION                                     STARS     OFFICIAL
mysql                           MySQL is a widely used, open-source relation…   15311     [OK]
mariadb                         MariaDB Server is a high performing open sou…   5828      [OK]
percona                         Percona Server is a fork of the MySQL relati…   631       [OK]
phpmyadmin                      phpMyAdmin - A web interface for MySQL and M…   1017      [OK]
databack/mysql-backup           Back up mysql databases to... anywhere!         118
circleci/mysql                  MySQL is a widely used, open-source relation…   30
bitwardenrs/server-mysql        [DEPRECATED] Use vaultwarden/server image, i…   12
linuxserver/mysql-workbench                                                     56
drupalci/mysql-5.5              https://www.drupal.org/project/drupalci         3
cimg/mysql                                                                      3
bitnami/mysql                   Bitnami container image for MySQL               114
bitnami/mysqld-exporter         Bitnami container image for MySQL Server Exp…   7
linuxserver/mysql               A Mysql container, brought to you by LinuxSe…   41
drupalci/mysql-5.7              https://www.drupal.org/project/drupalci         0
drupalci/mysql-8                https://www.drupal.org/project/drupalci         0
bitnamicharts/mysql                                                             0
docksal/mysql                   MySQL service images for Docksal - https://d…   0
jumpserver/mysql                                                                1
ddev/ddev-dbserver-mysql-8.0    DDEV's ddev-dbserver image                      0
istio/examples-bookinfo-mysqldb This image is used for Istio samples.          0
drud/ddev-dbserver-mysql-5.7                                                    1
eclipse/mysql                   Mysql 5.7, curl, rsync                          1
dockette/adminer                My most tiniest (10mb) Adminer image with su…   24
airbyte/normalization-mysql                                                     0
drud/mysql                                                                      1
PS C:\Users\Unfaithful>
```

docker history 9d05dd98f4a4

```
PS C:\Users\Unfaithful> docker images
REPOSITORY                TAG             IMAGE ID       CREATED       SIZE
mysql                     latest          7ce93a845a8a   3 weeks ago   586MB
in28min/todo-rest-api-h2  1.0.0.RELEASE   9d05dd98f4a4   2 years ago   143MB
in28min/todo-rest-api-h2  latest          9d05dd98f4a4   2 years ago   143MB
in28min/todo-rest-api-h2  0.0.1-SNAPSHOT  40ee0c4765e0   5 years ago   143MB
PS C:\Users\Unfaithful> docker history 9d05dd98f4a4
IMAGE          CREATED       CREATED BY                                      SIZE      COMMENT
9d05dd98f4a4   2 years ago   ENTRYPOINT ["sh" "-c" "java $JAVA_OPTS -Djav…   0B        buildkit.dockerfile.v0
<missing>      2 years ago   ENV JAVA_OPTS=                                  0B        buildkit.dockerfile.v0
<missing>      2 years ago   ADD target/*.jar app.jar # buildkit             38.1MB    buildkit.dockerfile.v0
<missing>      2 years ago   EXPOSE map[5000/tcp:{}]                         0B        buildkit.dockerfile.v0
<missing>      2 years ago   VOLUME [/tmp]                                   0B        buildkit.dockerfile.v0
<missing>      5 years ago   /bin/sh -c set -x  && apk add --no-cache   o…   99.3MB
<missing>      5 years ago   /bin/sh -c #(nop)  ENV JAVA_ALPINE_VERSION=8…   0B
<missing>      5 years ago   /bin/sh -c #(nop)  ENV JAVA_VERSION=8u212       0B
<missing>      5 years ago   /bin/sh -c #(nop)  ENV PATH=/usr/local/sbin:…   0B
<missing>      5 years ago   /bin/sh -c #(nop)  ENV JAVA_HOME=/usr/lib/jv…   0B
<missing>      5 years ago   /bin/sh -c {   echo '#!/bin/sh';   echo 'set…   87B
<missing>      5 years ago   /bin/sh -c #(nop)  ENV LANG=C.UTF-8             0B
<missing>      5 years ago   /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B
<missing>      5 years ago   /bin/sh -c #(nop) ADD file:a86aea1f3a7d68f6a…   5.53MB
PS C:\Users\Unfaithful>
```

docker image inspect 9d05dd98f4a4 -1

```
PS C:\Users\Unfaithful> docker image inspect 9d05dd98f4a4
[
    {
        "Id": "sha256:9d05dd98f4a4afdbeded8d1b93ee3dbf20ccba12f71bf43340c2e552c369506c",
        "RepoTags": [
            "in28min/todo-rest-api-h2:1.0.0.RELEASE",
            "in28min/todo-rest-api-h2:latest"
        ],
        "RepoDigests": [
            "in28min/todo-rest-api-h2@sha256:b50038f47a4ed68180ecf8d8efc8ec3ec25e97dd4232020ef03574384c48a23c"
        ],
        "Parent": "",
        "Comment": "buildkit.dockerfile.v0",
        "Created": "2022-06-01T10:03:36.454025827Z",
        "ContainerConfig": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": null,
            "Cmd": null,
            "Image": "",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": null
        },
        "DockerVersion": "",
        "Author": "",
        "Config": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "ExposedPorts": {
                "5000/tcp": {}
            },
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/lib/jvm/java-1.8-openjdk/jre/bin:/usr/lib/jvm/java-1.8-openjdk/bin",
                "LANG=C.UTF-8",
                "JAVA_HOME=/usr/lib/jvm/java-1.8-openjdk",
                "JAVA_VERSION=8u212",
                "JAVA_ALPINE_VERSION=8.212.04-r0",
                "JAVA_OPTS="
            ],
            "Cmd": null,
            "ArgsEscaped": true,
            "Image": "",
            "Volumes": {
                "/tmp": {}
            },
            "WorkingDir": "",
            "Entrypoint": [
                "sh",
                "-c",
                "java $JAVA_OPTS -Djava.security.egd=file:/dev/./urandom -jar /app.jar"
            ],
```

docker image inspect 9d05dd98f4a4 -2


```
            "OnBuild": null,
            "Labels": null
        },
        "Architecture": "amd64",
        "Os": "linux",
        "Size": 142956395,
        "VirtualSize": 142956395,
        "GraphDriver": {
            "Data": {
                "LowerDir": "/var/lib/docker/overlay2/a48930b4fe5e0bbf264bd7383dbdc12946e23057c1b3c66bf77571aa1808b49c/diff:/var/lib/docker/overlay2/9e9e5ba2ee7311d80a0782ec4c301f4f8154d7c056fc2eeb55f830cb8de31262/diff:/var/lib/docker/overlay2/9518a2061304c6adab59cd09
142155097bb6e14912c60a101cc4cb490c97bc83/diff",
                "MergedDir": "/var/lib/docker/overlay2/8431ebb2ef6377c7795e86f0d062f025a22448170e9fc25738f7f78e324f35ad/merged",
                "UpperDir": "/var/lib/docker/overlay2/8431ebb2ef6377c7795e86f0d062f025a22448170e9fc25738f7f78e324f35ad/diff",
                "WorkDir": "/var/lib/docker/overlay2/8431ebb2ef6377c7795e86f0d062f025a22448170e9fc25738f7f78e324f35ad/work"
            },
            "Name": "overlay2"
        },
        "RootFS": {
            "Type": "layers",
            "Layers": [
                "sha256:f1b5933fe4b5f49bbe8258745cf396afe07e625bdab3168e364daf7c956b6b81",
                "sha256:9b9b7f3d56a01e3d9076874990c62e7a516cc4032f784f421574d06b18ef9aa4",
                "sha256:ceaf9e1ebef5f9eaa707a838848a3c13800fcf32d7757be10d4b08fb85f1bc8a",
                "sha256:66a4dd35368970e9a4f68bf2cdd5af3959668529613bc875d5bf4cfff55d026c"
            ]
        },
        "Metadata": {
            "LastTagTime": "2024-08-19T18:10:35.138995702Z"
        }
    }
]
PS C:\Users\Unfaithful>
```

docker image remove 7ce93a845a8a

```
PS C:\Users\Unfaithful> docker images
REPOSITORY                TAG             IMAGE ID       CREATED       SIZE
mysql                     latest          7ce93a845a8a   3 weeks ago   586MB
in28min/todo-rest-api-h2  1.0.0.RELEASE   9d05dd98f4a4   2 years ago   143MB
in28min/todo-rest-api-h2  latest          9d05dd98f4a4   2 years ago   143MB
in28min/todo-rest-api-h2  0.0.1-SNAPSHOT  40ee0c4765e0   5 years ago   143MB
PS C:\Users\Unfaithful> docker image remove 7ce93a845a8a
Untagged: mysql:latest
Untagged: mysql@sha256:d8df069848906979fd7511db00dc22efeb0a33a990d87c3c6d3fcdafd6fc6123
Deleted: sha256:7ce93a845a8a98c89bd58a5e45fe71277ca5f52177c88046858a6a0af969ba74
Deleted: sha256:5f272bb3ff021589961b41dfd52486837f2bf55b257ecae518ca58f49f7cd05c
Deleted: sha256:3c053c0c18b0a01fe38376222040192bc70333fd033fdc0dc8982221b8e28094
Deleted: sha256:027b6d1a4884ded3bcd7d3e045501fe77670aa24c55bca0c6cfd32915b2ec0c6
Deleted: sha256:b18d8cac7e324a500bc7da81a37ab8249282ba61dbdeba8595f7b3893a5abfe7
Deleted: sha256:cae0e6c244b3ef3dabdc2e80148c6d816502887a27cdf6c8b9555c9cfecc170d
Deleted: sha256:86f178c90fd041f1b20a9e342ad6af31a0db1b22094c16eae4d1ea3c07a09314
Deleted: sha256:05b2d940624b43f8a3ed710dcaba6bc5427d6ce2853588eddc49f833cce80e2a
Deleted: sha256:9d9ac1b7ddb58fe8de72181da67b1f3d0b2a197f8e4f8fe42af7e108f576fd83
Deleted: sha256:02b4320329b9e205e7090059e0ac630d97395708803d3793f4628f8ecb926294
Deleted: sha256:2606c15a4838dfb909dab29f47c601a797f657f51a35005bd06d01da891d813c
PS C:\Users\Unfaithful>
```

Docker Container Notes:

```
docker run -p 5000:5000 -d in28min/todo-rest-api-h2:1.0.0.RELEASE
```

=

```
docker container run -p 5000:5000 -d in28min/todo-rest-api-h2:
1.0.0.RELEASE
```

docker container pause 25c9

```
PS C:\Users\Unfaithful> docker container run -p 5000:5000 -d in28min/todo-rest-api-h2:1.0.0.RELEASE
25c954c125dee1232592c3e53fcf9cc2d1276bb418a29f876a48d061025d64ef
PS C:\Users\Unfaithful> docker container pause 25c9
25c9
```

Containers

Give feedback

Container CPU usage

No containers are running.

Container memory usage

No containers are running.

Show charts

Search

Only show running containers

Name	Image	Status	Port(s)	CPU ...	Last started	Actions
nostalgic_nobel b2956ab469ef	in28min/todo-rest-api-h2:0.0.1-SNAP	Exited (130)	5000:5000	N/A	17 hours ago	
competent_solomon ae4232758b25	in28min/todo-rest-api-h2:1.0.0.RELEA	Exited (143)	5001:5000	N/A	21 hours ago	
inspiring_diffie 680cbc81ab5c	in28min/todo-rest-api-h2:1.0.0.RELEA	Exited (143)	5000:5000	N/A	22 hours ago	
peaceful_lehmann fd9ee4ba4d04	in28min/todo-rest-api-h2:1.0.0.RELEA	Exited (130)	5000:5000	N/A	23 hours ago	
affectionate_proskuriakova d5bd2f1ecc06	in28min/todo-rest-api-h2:1.0.0.RELEA	Exited (130)		N/A	1 day ago	
nifty_morse 25c954c125de	in28min/todo-rest-api-h2:1.0.0.RELEA	Paused	5000:5000	N/A	6 minutes ago	

docker container unpause 25c9

```
PS C:\Users\Unfaithful> docker container unpause 25c9
25c9
```

docker container prune (remove stopped containers)

```
PS C:\Users\Unfaithful> docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
b2956ab469ef9c16c96b4d16843b3ed4948cf569818fc52f32e1bc42f92e6b61
ae4232758b25cb51f31823369b4d3660e33bc793d94c4fb5b68745e3eb294975
680cbc81ab5c3d00efc6323eb4078374ca1474a9a0a8155621e9dba3d1d2df17
fd9ee4ba4d04d599a964ab26ded8db525661cb9a2c7bf95d18cfaefac1145f1b
d5bd2f1ecc06703cb6bd202a4e3b10c2c74260851e2bc57b6796a7011f84d655

Total reclaimed space: 0B
PS C:\Users\Unfaithful>
```

docker container stop 25c stop => sigterm => graceful shutdown

```
PS C:\Users\Unfaithful> docker container ls
CONTAINER ID   IMAGE                                   COMMAND                  CREATED             STATUS          PORTS                    NAMES
25c954c125de   in28min/todo-rest-api-h2:1.0.0.RELEASE   "sh -c 'java $JAVA_O…"   About an hour ago   Up 34 seconds   0.0.0.0:5000->5000/tcp   nifty_morse
PS C:\Users\Unfaithful> docker container stop 25c
25c
PS C:\Users\Unfaithful>
```

```
2024-08-20 08:53:38.155  INFO 1 --- [           main] org.hibernate.cfg.Environment            : HHH000206: hibernate.properties not found
2024-08-20 08:53:38.458  INFO 1 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.0.4.Final}
2024-08-20 08:53:38.817  INFO 1 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.H2Dialect
Hibernate: drop table todo if exists
Hibernate: drop sequence if exists hibernate_sequence
Hibernate: create sequence hibernate_sequence start with 1 increment by 1
Hibernate: create table todo (id bigint not null, description varchar(255), is_done boolean not null, target_date timestamp, username varchar(255), primary
key (id))
2024-08-20 08:53:39.717  INFO 1 --- [           main] o.h.t.schema.internal.SchemaCreatorImpl  : HHH000476: Executing import script 'org.hibernate.tool.sche
ma.internal.exec.ScriptSourceInputNonExistentImpl@5ddeb7cb'
2024-08-20 08:53:39.721  INFO 1 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit '
default'
2024-08-20 08:53:40.513  INFO 1 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2024-08-20 08:53:40.587  WARN 1 --- [           main] aWebConfiguration$JpaWebMvcConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, d
atabase queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2024-08-20 08:53:41.175  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 5000 (http) with context path ''
2024-08-20 08:53:41.179  INFO 1 --- [           main] c.i.r.w.r.RestfulWebServicesApplication  : Started RestfulWebServicesApplication in 7.916 seconds (JVM
 running for 9.051)
2024-08-20 08:54:07.853  INFO 1 --- [       Thread-3] o.s.s.concurrent.ThreadPoolTaskExecutor  : Shutting down ExecutorService 'applicationTaskExecutor'
2024-08-20 08:54:07.857  INFO 1 --- [       Thread-3] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'defa
ult'
2024-08-20 08:54:07.858  INFO 1 --- [       Thread-3] .SchemaDropperImpl$DelayedDropActionImpl : HHH000477: Starting delayed evictData of schema as part of
SessionFactory shut-down'
Hibernate: drop table todo if exists
Hibernate: drop sequence if exists hibernate_sequence
2024-08-20 08:54:07.875  INFO 1 --- [       Thread-3] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
2024-08-20 08:54:07.883  INFO 1 --- [       Thread-3] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
PS C:\Users\Unfaithful>
```

It takes a bit time like *7s or *10s

docker container kill aee kill => sigkill => immediately terminates the process

```
PS C:\Users\Unfaithful> docker container kill aee
aee
PS C:\Users\Unfaithful>
```

```
2024-08-20 09:01:18.011  INFO 1 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 309
8 ms
2024-08-20 09:01:18.309  INFO 1 --- [           main] o.s.b.w.servlet.ServletRegistrationBean  : Servlet dispatcherServlet mapped to [/]
2024-08-20 09:01:18.311  INFO 1 --- [           main] o.s.b.w.servlet.ServletRegistrationBean  : Servlet webServlet mapped to [/h2-console/*]
2024-08-20 09:01:18.323  INFO 1 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'characterEncodingFilter' to: [/*]
2024-08-20 09:01:18.324  INFO 1 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
2024-08-20 09:01:18.325  INFO 1 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'formContentFilter' to: [/*]
2024-08-20 09:01:18.326  INFO 1 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to: [/*]
2024-08-20 09:01:18.531  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-08-20 09:01:18.860  INFO 1 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2024-08-20 09:01:18.979  INFO 1 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [
        name: default
        ...]
2024-08-20 09:01:19.086  INFO 1 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate Core {5.3.7.Final}
2024-08-20 09:01:19.089  INFO 1 --- [           main] org.hibernate.cfg.Environment            : HHH000206: hibernate.properties not found
2024-08-20 09:01:19.384  INFO 1 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.0.4.Final}
2024-08-20 09:01:19.733  INFO 1 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.H2Dialect
Hibernate: drop table todo if exists
Hibernate: drop sequence if exists hibernate_sequence
Hibernate: create sequence hibernate_sequence start with 1 increment by 1
Hibernate: create table todo (id bigint not null, description varchar(255), is_done boolean not null, target_date timestamp, username varchar(255), primary
key (id))
2024-08-20 09:01:20.680  INFO 1 --- [           main] o.h.t.schema.internal.SchemaCreatorImpl  : HHH000476: Executing import script 'org.hibernate.tool.sche
ma.internal.exec.ScriptSourceInputNonExistentImpl@8462f31'
2024-08-20 09:01:20.684  INFO 1 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit '
default'
2024-08-20 09:01:21.649  INFO 1 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2024-08-20 09:01:21.730  WARN 1 --- [           main] aWebConfiguration$JpaWebMvcConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, d
atabase queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2024-08-20 09:01:22.443  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 5000 (http) with context path ''
2024-08-20 09:01:22.448  INFO 1 --- [           main] c.i.r.w.r.RestfulWebServicesApplication  : Started RestfulWebServicesApplication in 8.366 seconds (JVM
 running for 9.271)
PS C:\Users\Unfaithful>
```

docker container run -p 5000:5000 --restart=always -d in28min/todo-rest-api-h2:1.0.0.RELEASE

```
PS C:\Users\Unfaithful> docker container run -p 5000:5000 --restart=always -d in28min/todo-rest-api-h2:1.0.0.RELEASE
345ddcafe686e2128a04df938b990db600c6e8fb21d3028e812d371fe41ea005
PS C:\Users\Unfaithful> docker container stop 345
345
PS C:\Users\Unfaithful> docker container ls -a
CONTAINER ID   IMAGE                                      COMMAND                  CREATED          STATUS                      PORTS     NAMES
345ddcafe686   in28min/todo-rest-api-h2:1.0.0.RELEASE   "sh -c 'java $JAVA_O…"   53 seconds ago   Exited (143) 7 seconds ago            determined_montalci
ni
PS C:\Users\Unfaithful>
```

The container automatically start when docker is restart

DEFAULT:

docker container run -p 5000:5000 --restart=no -d in28min/todo-rest-api-h2:1.0.0.RELEASE

It is useful for databases.

docker events

```
PS C:\Users\Unfaithful> docker events
2024-08-21T09:57:09.021511717+03:00 network connect c8782885d0f82b2c345dfb9f2d1790d4d274f5a842a8078979b9ce6b9ded9c39 (container=345ddcafe686e2128a04df938b990db600c6e8fb21d3028e812d371fe41ea005, name=bridge, type=bridge)
2024-08-21T09:57:09.025252533+03:00 volume mount 5ec785e2a4b3abd962eadfcb5445c0ffa7decc47c098b666e9b90cf05ad9c1c4 (container=345ddcafe686e2128a04df938b990db600c6e8fb21d3028e812d371fe41ea005, destination=/tmp, driver=local, propagation=, read/write=true)
2024-08-21T09:57:09.351004634+03:00 container start 345ddcafe686e2128a04df938b990db600c6e8fb21d3028e812d371fe41ea005 (image=in28min/todo-rest-api-h2:1.0.0.RELEASE, name=determined_montalcini)
2024-08-21T09:57:24.247875366+03:00 container kill 345ddcafe686e2128a04df938b990db600c6e8fb21d3028e812d371fe41ea005 (image=in28min/todo-rest-api-h2:1.0.0.RELEASE, name=determined_montalcini, signal=15)
2024-08-21T09:57:24.784798691+03:00 network disconnect c8782885d0f82b2c345dfb9f2d1790d4d274f5a842a8078979b9ce6b9ded9c39 (container=345ddcafe686e2128a04df938b990db600c6e8fb21d3028e812d371fe41ea005, name=bridge, type=bridge)
2024-08-21T09:57:24.792943178+03:00 volume unmount 5ec785e2a4b3abd962eadfcb5445c0ffa7decc47c098b666e9b90cf05ad9c1c4 (container=345ddcafe686e2128a04df938b990db600c6e8fb21d3028e812d371fe41ea005, driver=local)
2024-08-21T09:57:24.793336197+03:00 container stop 345ddcafe686e2128a04df938b990db600c6e8fb21d3028e812d371fe41ea005 (image=in28min/todo-rest-api-h2:1.0.0.RELEASE, name=determined_montalcini)
2024-08-21T09:57:24.800637044+03:00 container die 345ddcafe686e2128a04df938b990db600c6e8fb21d3028e812d371fe41ea005 (execDuration=15, exitCode=143, image=in28min/todo-rest-api-h2:1.0.0.RELEASE, name=determined_montalcini)
```

docker top 345ddcafe686 →

The docker top actually is used to check what is the top process, which is running in a specific container.

Display the running processes of a container

```
PS C:\Users\Unfaithful> docker container ls
CONTAINER ID   IMAGE                                    COMMAND                  CREATED       STATUS         PORTS                    NAMES
345ddcafe686   in28min/todo-rest-api-h2:1.0.0.RELEASE   "sh -c 'java $JAVA_O…"   22 hours ago   Up 15 seconds   0.0.0.0:5000->5000/tcp   determined_montalci
ni
PS C:\Users\Unfaithful> docker top 345ddcafe686
UID                 PID                 PPID                C                   STIME               TTY                 TIME                CMD
root                666                 646                 94                  07:02               ?                   00:00:35            java -Djava.secu
rity.egd=file:/dev/./urandom -jar /app.jar
PS C:\Users\Unfaithful>
```

docker stats

```
PS C:\Users\Unfaithful> docker stats
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   3.77%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   3.77%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.18%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.14%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.14%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.14%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.18%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.18%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.17%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.17%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.14%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.14%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.17%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.17%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.14%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.14%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
CONTAINER ID   NAME                    CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O   PIDS
345ddcafe686   determined_montalcini   0.16%     477MiB / 7.664GiB     6.08%     1.05kB / 0B     0B / 0B     38
```

docker run -p 5001:5000 -m 512m --cpu-quota 5000 -d in28min/todo-rest-api-h2:1.0.0.RELEASE

100000 => %100
5000 => %5

```
PS C:\Users\Unfaithful> docker run -p 5001:5000 -m 512m --cpu-quota 5000 -d in28min/todo-rest-api-h2:1.0.0.RELEASE
9fb0fd2e87c4fd3b06e6cd405bbb8fa1600540567eb0b2c9676ad9eef3d3faaa
PS C:\Users\Unfaithful> docker stats
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    0.24%     32.96MiB / 512MiB   6.44%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    6.31%     30.29MiB / 512MiB   5.92%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    6.31%     30.29MiB / 512MiB   5.92%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    3.84%     30.32MiB / 512MiB   5.92%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    3.84%     30.32MiB / 512MiB   5.92%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    3.84%     30.32MiB / 512MiB   5.92%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    5.54%     30.49MiB / 512MiB   5.96%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    5.54%     30.49MiB / 512MiB   5.96%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    4.17%     30.64MiB / 512MiB   5.98%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    4.17%     30.64MiB / 512MiB   5.98%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    5.71%     30.69MiB / 512MiB   5.99%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    5.71%     30.69MiB / 512MiB   5.99%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    4.71%     30.81MiB / 512MiB   6.02%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    4.71%     30.81MiB / 512MiB   6.02%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
9fb0fd2e87c4   nervous_tu    4.53%     30.9MiB / 512MiB    6.03%     746B / 0B   0B / 0B     11
CONTAINER ID   NAME          CPU %     MEM USAGE / LIMIT   MEM %     NET I/O     BLOCK I/O   PIDS
```

docker system df → It helps us to look atall the different things that your Docker daemon manages.

```
PS C:\Users\Unfaithful> docker system df
TYPE            TOTAL     ACTIVE    SIZE      RECLAIMABLE
Images          2         1         181.1MB   143MB (78%)
Containers      2         1         0B        0B
Local Volumes   9         2         65.54kB   32.77kB (50%)
Build Cache     0         0         0B        0B
PS C:\Users\Unfaithful>
```

docker images downloaded

Name	Tag	Status	Created	Size	Actions
in28min/todo-rest-api-h2 9d05dd98f4a4	1.0.0.RELEASE	In use	2 years ago	142.95 MB	
in28min/todo-rest-api-h2 9d05dd98f4a4	latest	In use	2 years ago	142.95 MB	
in28min/todo-rest-api-h2 40ee0c4765e0	0.0.1-SNAPSHOT	Unused	5 years ago	142.95 MB	

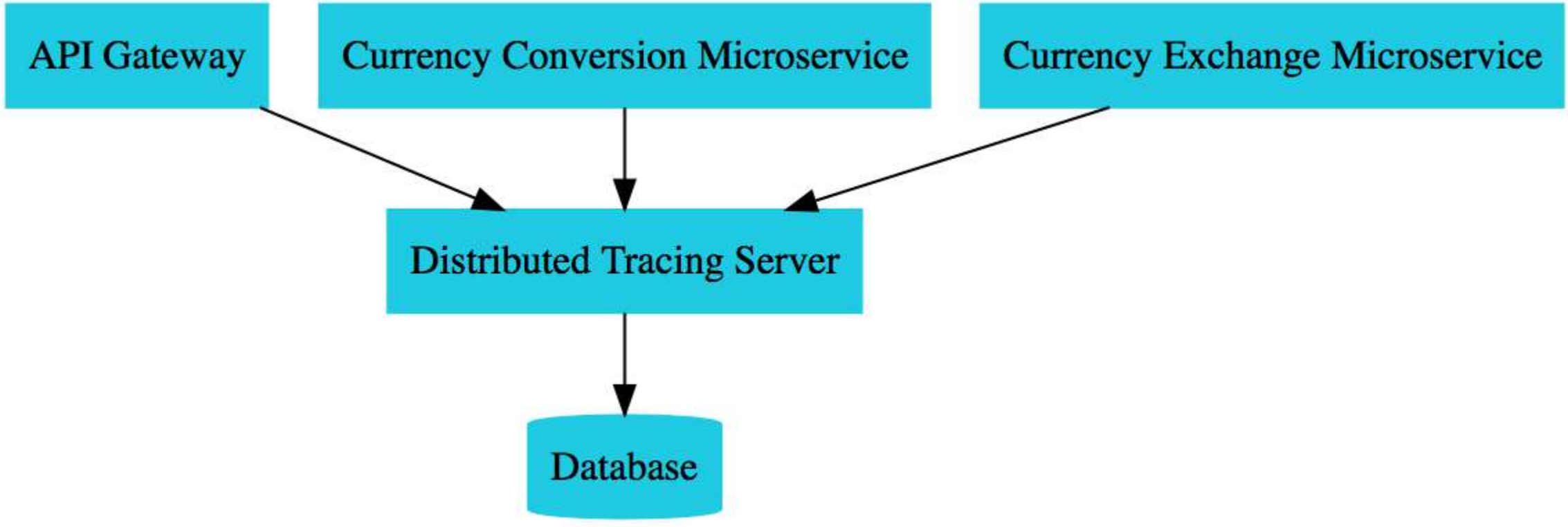
Showing 3 items

I create latest tag one, so it does not downloaded.
(it is different tag same image)

Distributed Tracing

- Complex call chain
- How do you debug problems?
- How do you trace requests across microservices?
- Enter Distributed Tracing

How do you find out where the problem is?



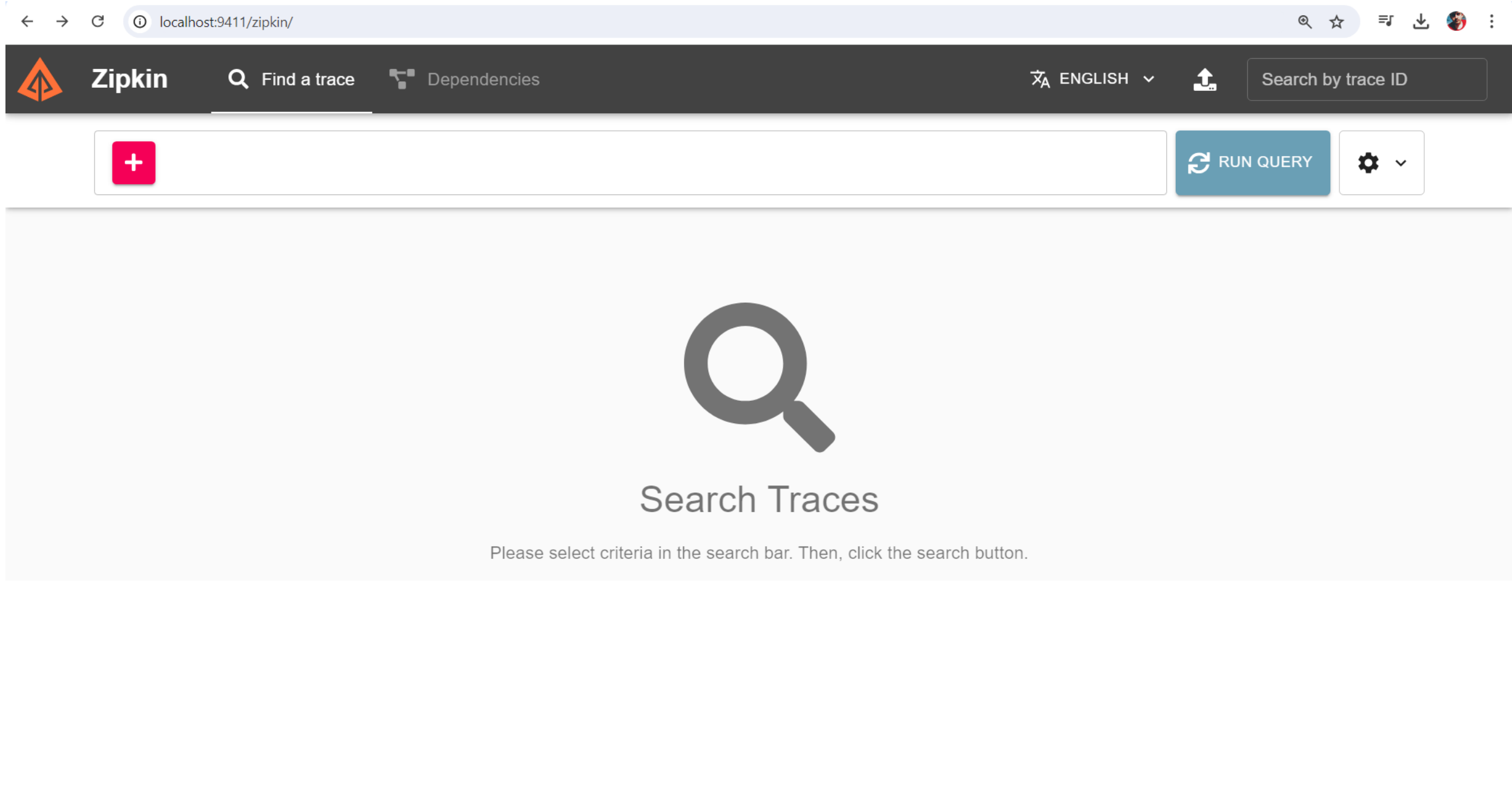
Distributed Tracing

Launching Zipkin Container Using Docker

☐ openzipkin/zipkin web site (click square)

docker run -p 9411:9411 openzipkin/zipkin:2.23

http://localhost:9411/zipkin/



Observability and OpenTelemetry

- **Monitoring vs observability**: Monitoring is reactive. Observability is proactive.
  - Monitoring is a subset of Observability.
- **Observability**: How well do we understand what's happening in a system?
  - **STEP I**: Gather data: metrics, logs, or traces Monitoring
  - **STEP II**: Get Intelligence: AI/Ops and anomaly detection Observability
- **OpenTelemetry**: Collection of tools, APIs, and SDKs to instrument, generate, collect, & export telemetry data (metrics, logs, & traces)
  - All applications have metrics, logs, and traces
    - Why do we need to have a separate standard for each one of these?
  - **OpenTelemetry**: How about one standard for metrics, logs, and traces?
  - Almost all cloud platforms provide support for OpenTelemetry today!

In observability, we focus on getting intelligence from this data,
and hopefully this intelligence would help us to detect problems faster.

Docker Compose

https://docs.docker.com/compose/

Docker Compose is a tool for defining and running multi-container applications.

https://docs.docker.com/compose/install/ → Docker Desktop includes Docker Compose

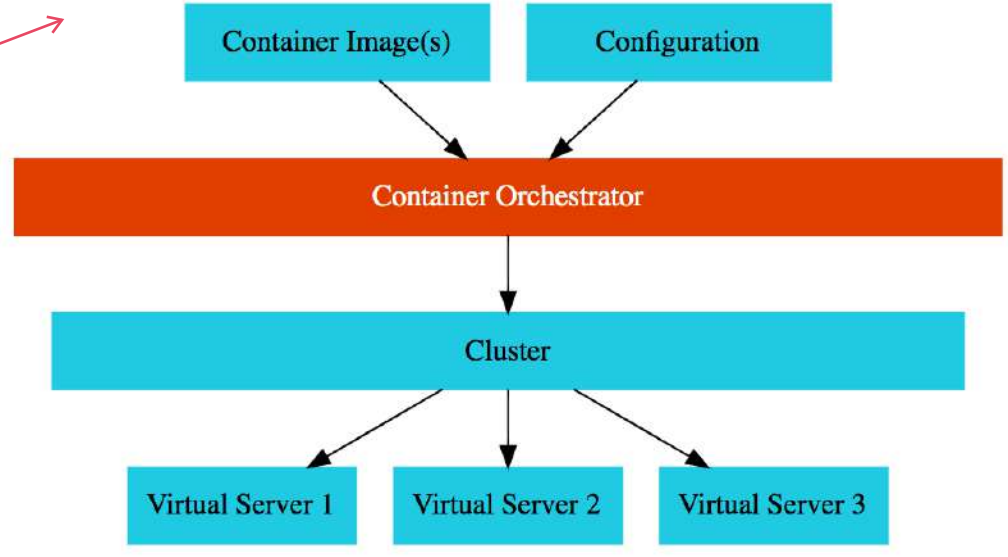
docker-compose --version → Check Docker Compose

Container Orchestration

In28 Minutes

- **Requirement** : I want 10 instances of Microservice A container, 15 instances of Microservice B container and ....
- Typical Features:
  - **Auto Scaling** - Scale containers based on demand
  - **Service Discovery** - Help microservices find one another
  - **Load Balancer** - Distribute load among multiple instances of a microservice
  - **Self Healing** - Do health checks and replace failing instances
  - **Zero Downtime Deployments** - Release new versions without downtime

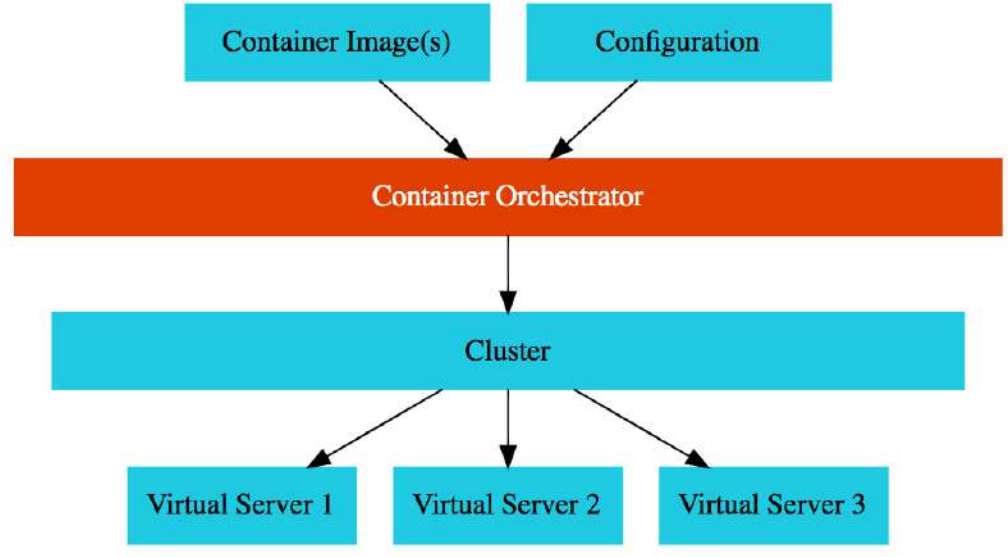
I would want auto scaling. Based on the load which is coming in, I would want to scale the number of containers. If a Microservice A is having a lot of load, then I would want to have five instances, 20 instances, hundred instances basedon the load. So I would want auto scaling. I would want to be able to scale containers basedon the demand.

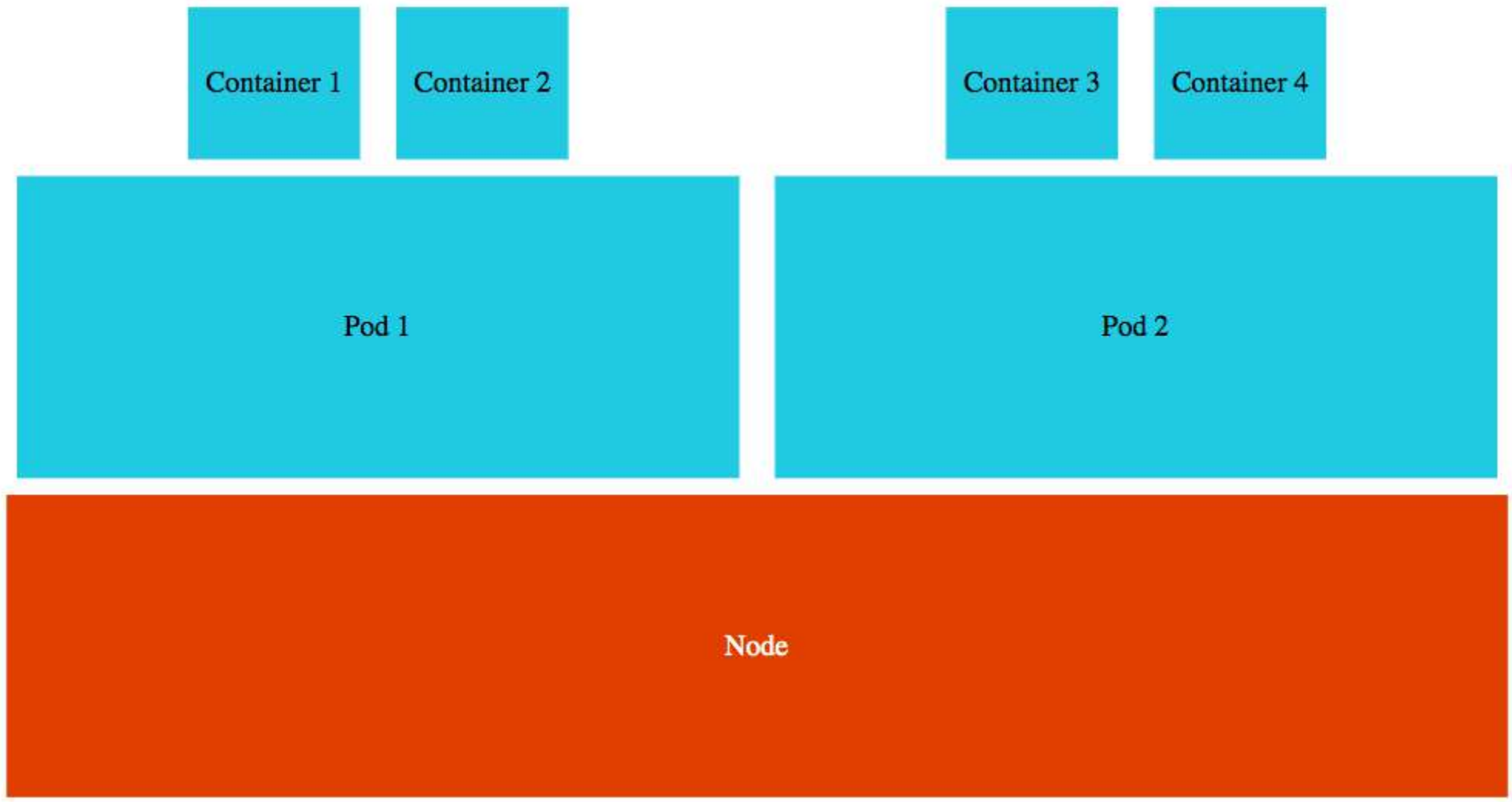


If one of the Microservice instances is down, I would want the container orchestration toolto identify that and automatically launch a new instance. So we want to do health checksand replace failing instances

Container Orchestration Options

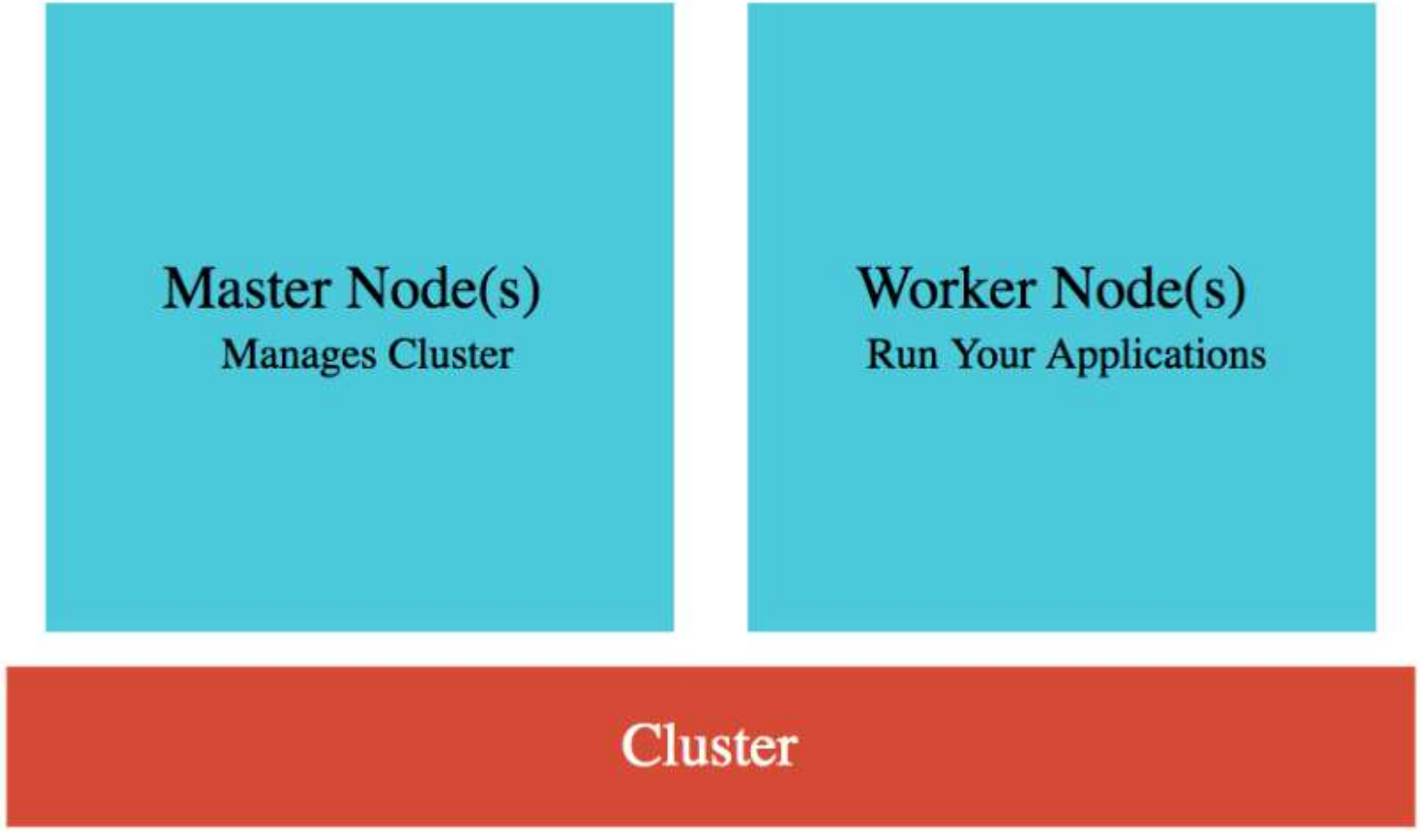
- **AWS Specific**
  - AWS Elastic Container Service (ECS)
  - AWS Fargate : Serverless version of AWS ECS
- **Cloud Neutral** - Kubernetes
  - AWS - Elastic Kubernetes Service (EKS)
  - Azure - Azure Kubernetes Service (AKS)
  - GCP - Google Kubernetes Engine (GKE)
  - EKS/AKS does not have a free tier!
    - We use GCP and GKE!



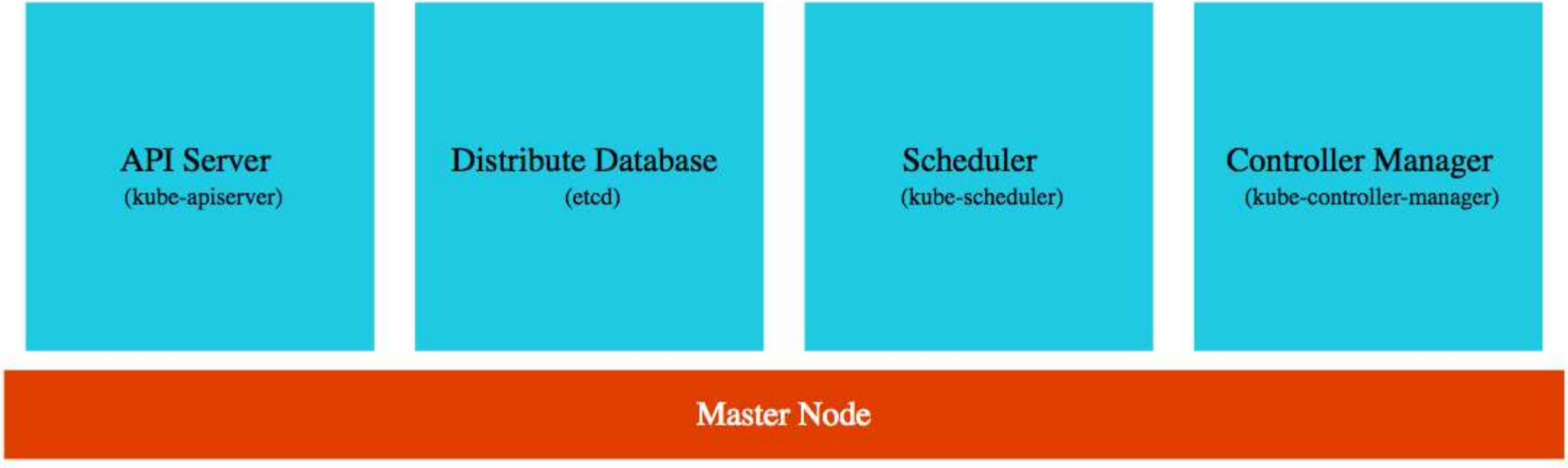

Container 1
Container 2
Container 3
Container 4
Pod 1
Pod 2
Node

Kubernetes Architecture

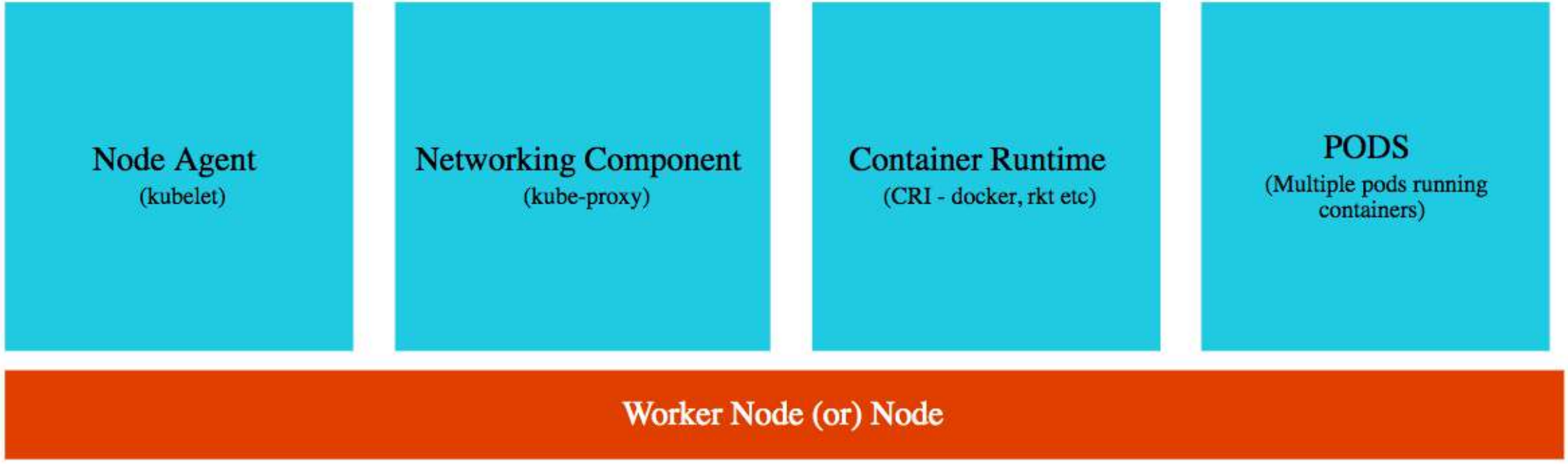
In28 Minutes



Kubernetes Architecture



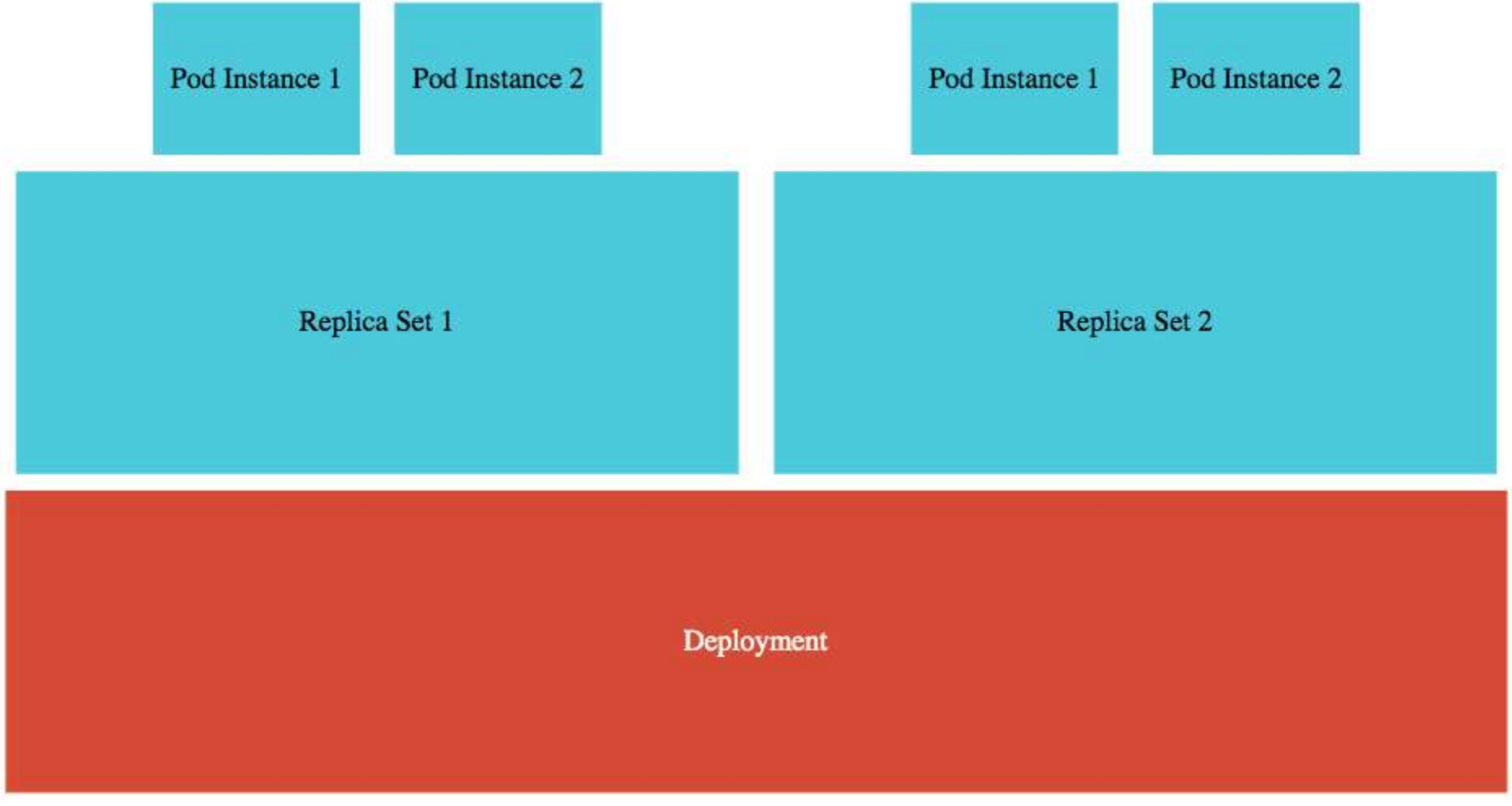
Kubernetes Architecture



Kubernetes Architecture

Kubernete Deployments

Kubernete Deployments



Kubernete Deployments

Kubernete Service

Kubernetes - Liveness and Readiness Probes

- Kubernetes uses probes to check the health of a microservice:
  - If readiness probe is not successful, no traffic is sent
  - If liveness probe is not successful, pod is restarted
- Spring Boot Actuator (>=2.3) provides inbuilt readiness and liveness probes:
  - /health/readiness
  - /health/liveness

What Next?

7 Roadmaps, 1 Million+ Learners

In28 Minutes

- *AWS*
- *Azure*
- *Google Cloud*
- *Spring Boot*
- *Java Microservices*
- *Java Full Stack*
- *DevOps*