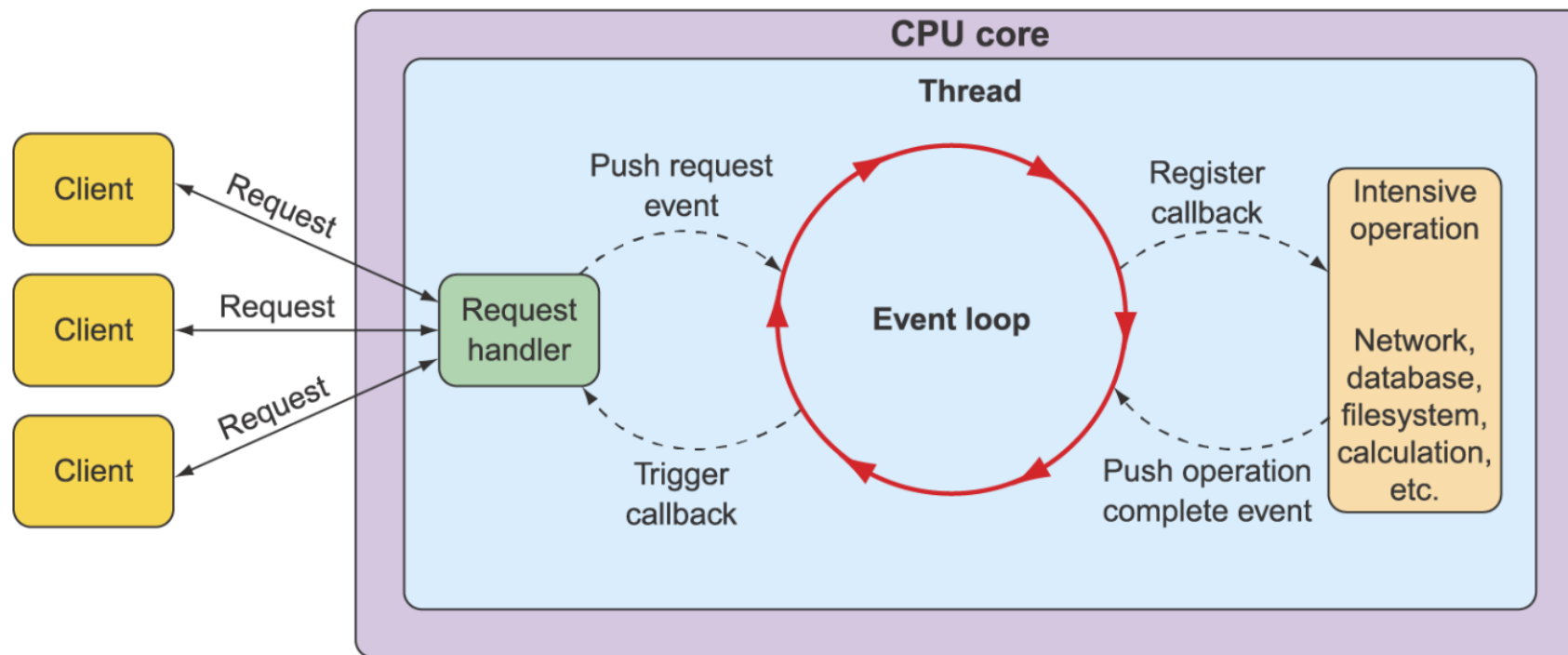


# 服务端开发-Spring WebFlux

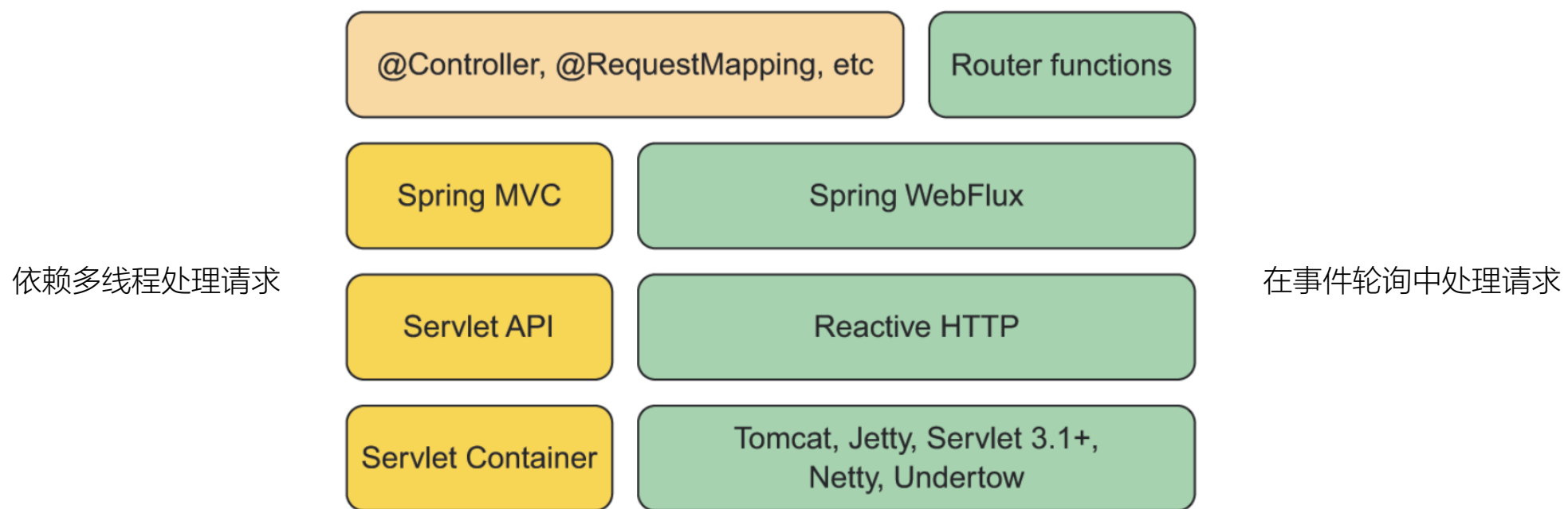
陶召胜

# 异步Web框架的事件轮询(event looping)机制

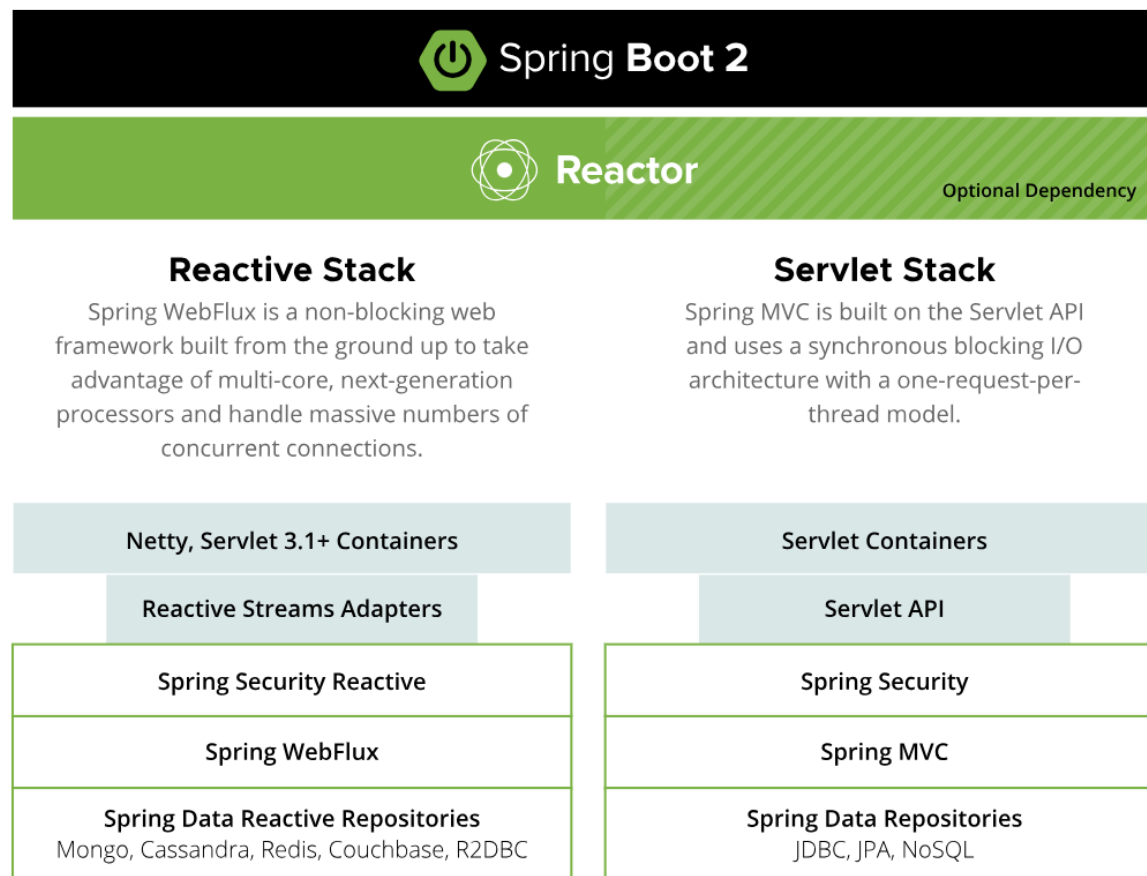
- 用更少的线程处理更多的请求，从而减少线程管理的开销



# Spring MVC与Spring WebFlux的共性与不同



# Reactive Microservices With Spring Boot



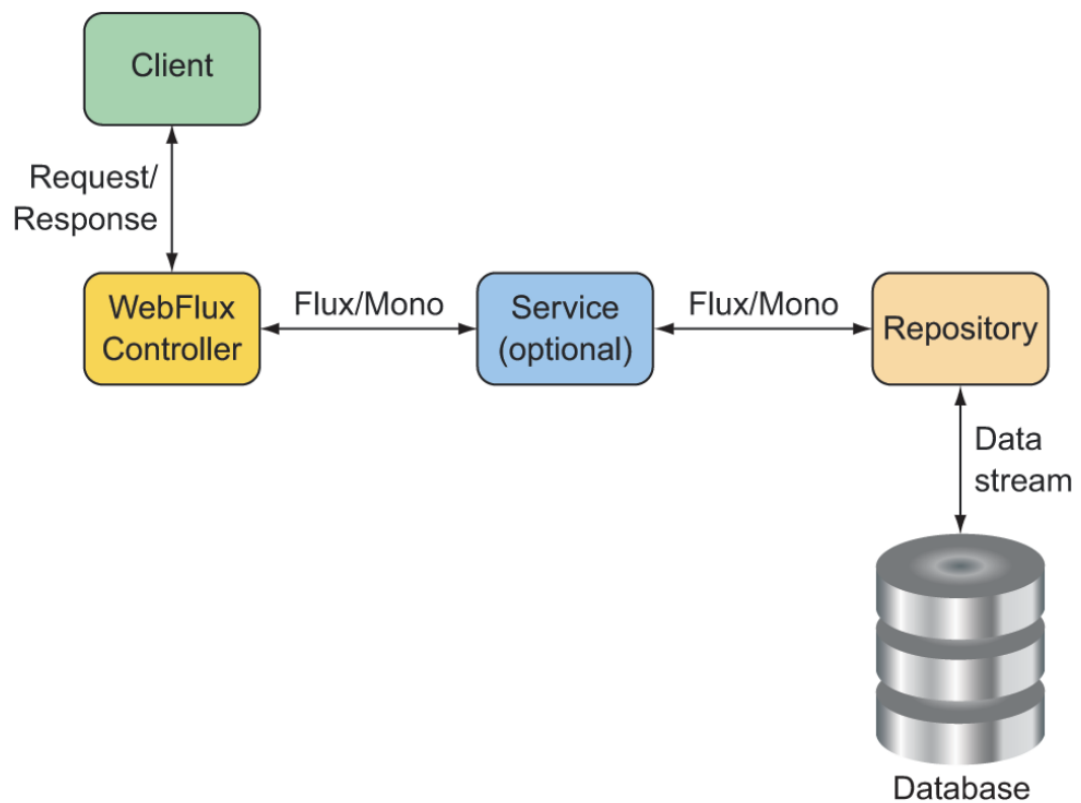
# 依赖

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-webflux</artifactId>  
</dependency>
```

# 编写反应式控制器

- Ingredient: Get、Post

# 端到端反应式栈



# R2DBC

- 反应式关系型数据库连接 (reactive relational database connectivity)
- 是JDBC的替代方案，实现非阻塞的持久化操作
- 依赖

```
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-data-r2dbc</artifactId>  
</dependency>
```

- H2数据库和驱动

```
<dependency>  
    <groupId>com.h2database</groupId>  
    <artifactId>h2</artifactId>  
    <scope>runtime</scope>  
</dependency>  
<dependency>  
    <groupId>io.r2dbc</groupId>  
    <artifactId>r2dbc-h2</artifactId>  
    <scope>runtime</scope>  
</dependency>
```



# 使用函数式编程范式来定义控制器

- 使用函数式编程风格来定义endpoints的功能
- 框架引入了两个基本组件：HandlerFunction 和 RouterFunction
- HandlerFunction 表示处理接收到的请求并生成响应的函数
- RouterFunction 替代了 @RequestMapping 注解。它用于将接收到的请求路由到处理函数

# 函数式编程模型涉及的4个类型

- RouterFunction Bean
- [org.springframework.web.reactive.function.server](https://org.springframework.web.reactive.function.server)
  - ✓ RequestPredicate, 声明要处理的请求类型
  - ✓ RouterFunction, 声明如何将请求路由到处理器代码中
  - ✓ ServerRequest, 代表一个http请求, 包括对请求头和请求体的访问
  - ✓ ServerResponse, 代表一个http响应, 包括响应头和响应体信息

# 测试反应式控制器

- 处理器函数使用单独的Bean实现, GreetingHandler
- WebClient

# 反应式消费Rest API

- RestTemplate=>WebClient
- GreetingClient

# 控制器的快速返回

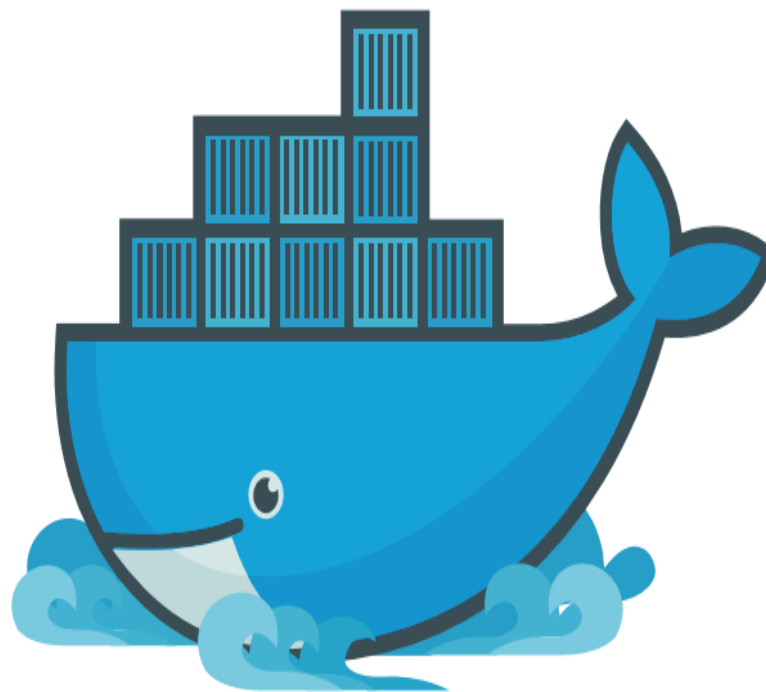
- /mono
- /flux, text/event-stream

---

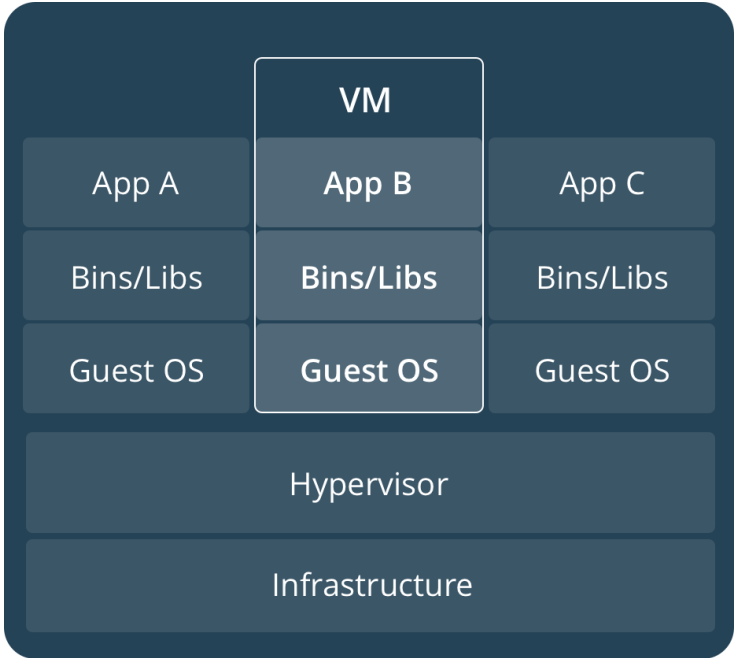
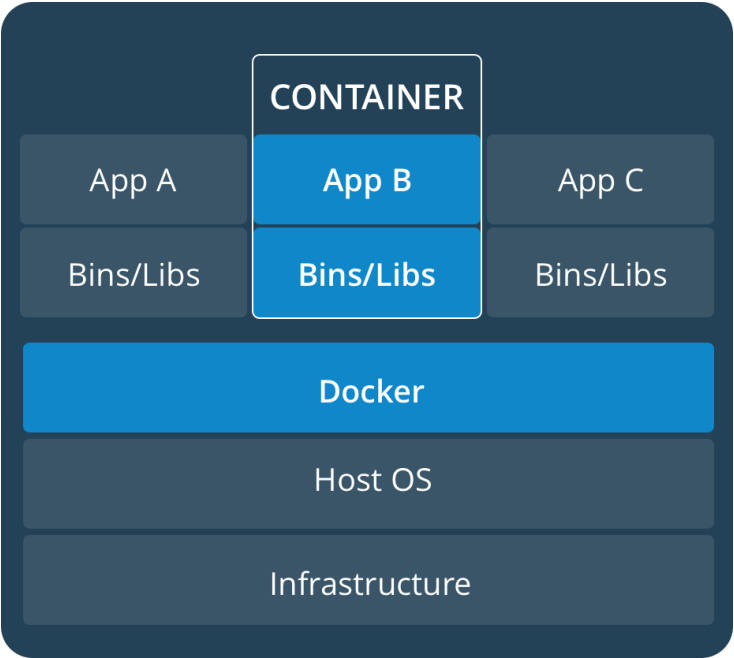
# 下节课前请安装Docker

## Build, Ship, Run

Docker is the world's leading software  
containerization platform



# 容器与虚拟机

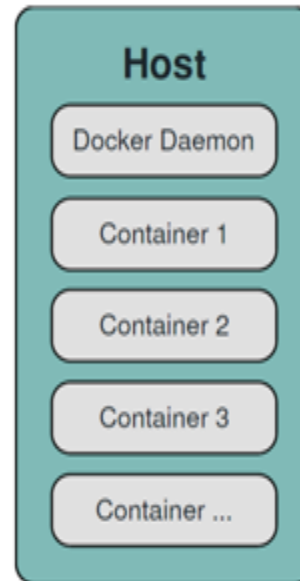


# Docker的3部分

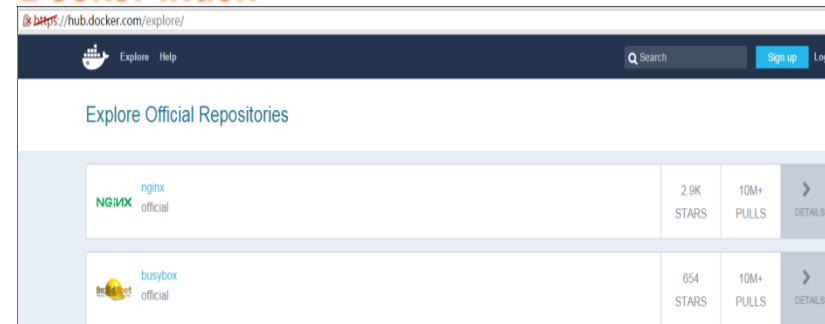
- Docker daemon就是Docker Engine, 一般在宿主主机后台运行
- 用户使用client通过pipe、unix socket或tcp直接跟daemon交互
- Docker index指向Docker registries, 也叫docker仓库, 可以用来让你上传和下载images。Hub.docker.com为docker官方仓库

## Docker Client

```
docker pull
docker run
docker ...
```



## Docker Index



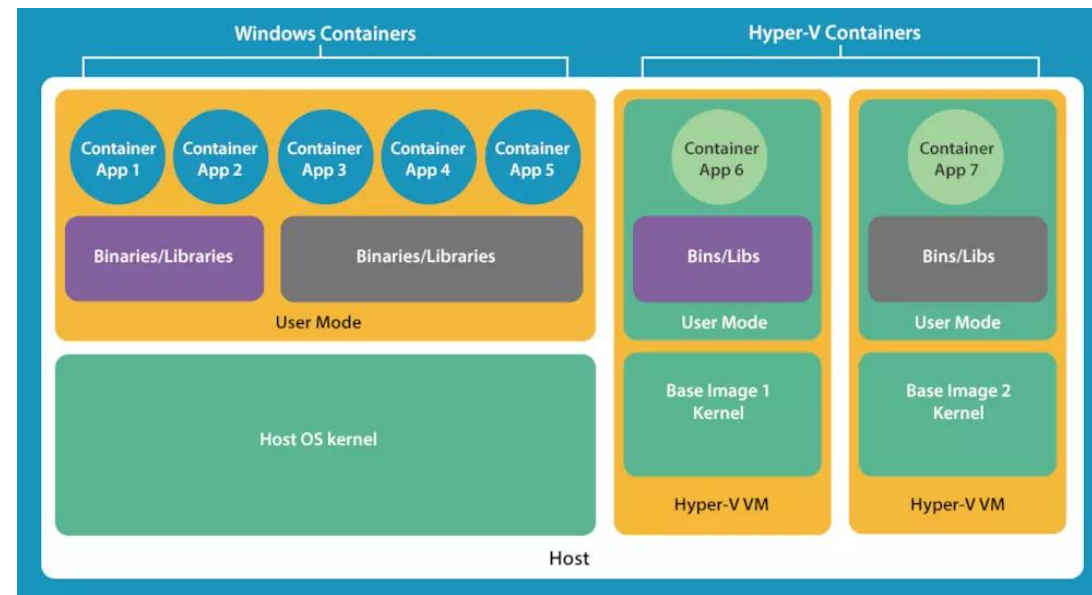


# Docker可运行在以下操作系统

- Windows
- OS X
- Linux

# Windows下的两类容器

- Windows Container
- linux container



# 作业：Docker安装请反馈截图

- 1、docker version 来查看版本号
- 2、docker run hello-world 载入测试镜像

```
C:\Users\tzs91>docker version
Client:
 Cloud integration: v1.0.20
 Version: 20.10.10
 API version: 1.41
 Go version: go1.16.9
 Git commit: b485636
 Built: Mon Oct 25 07:47:53 2021
 OS/Arch: windows/amd64
 Context: default
 Experimental: true

Server: Docker Engine - Community
 Engine:
  Version: 20.10.10
  API version: 1.41 (minimum version 1.12)
  Go version: go1.16.9
  Git commit: e2f740d
  Built: Mon Oct 25 07:41:30 2021
  OS/Arch: linux/amd64
  Experimental: false
 containerd:
  Version: 1.4.11
  GitCommit: 5b46e404f6b9f661a205e28d59c982d3634148f8
 runc:
  Version: 1.0.2
  GitCommit: v1.0.2-0-g52b36a2
 docker-init:
  Version: 0.19.0
  GitCommit: de40ad0
```

```
命令提示符
C:\Users\tzs91>docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Already exists
Digest: sha256:f9dfddf63636d84ef479d645ab5885156ae030f611a56f3a7ac7f2fdd86d7e4e
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

C:\Users\tzs91>
```

谢谢观看！

