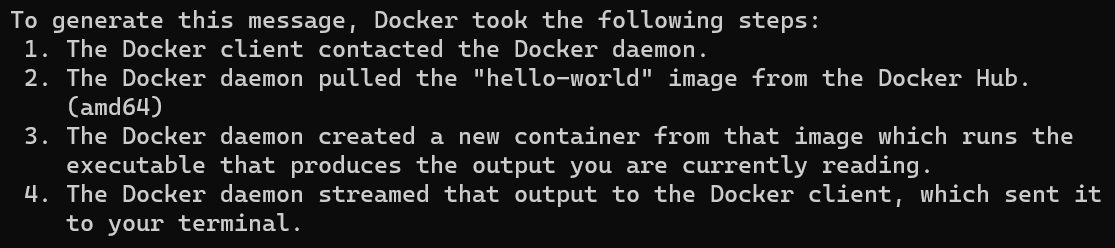
**Docker、K8s配置学习**

**Docker运行原理**

docker run <image>

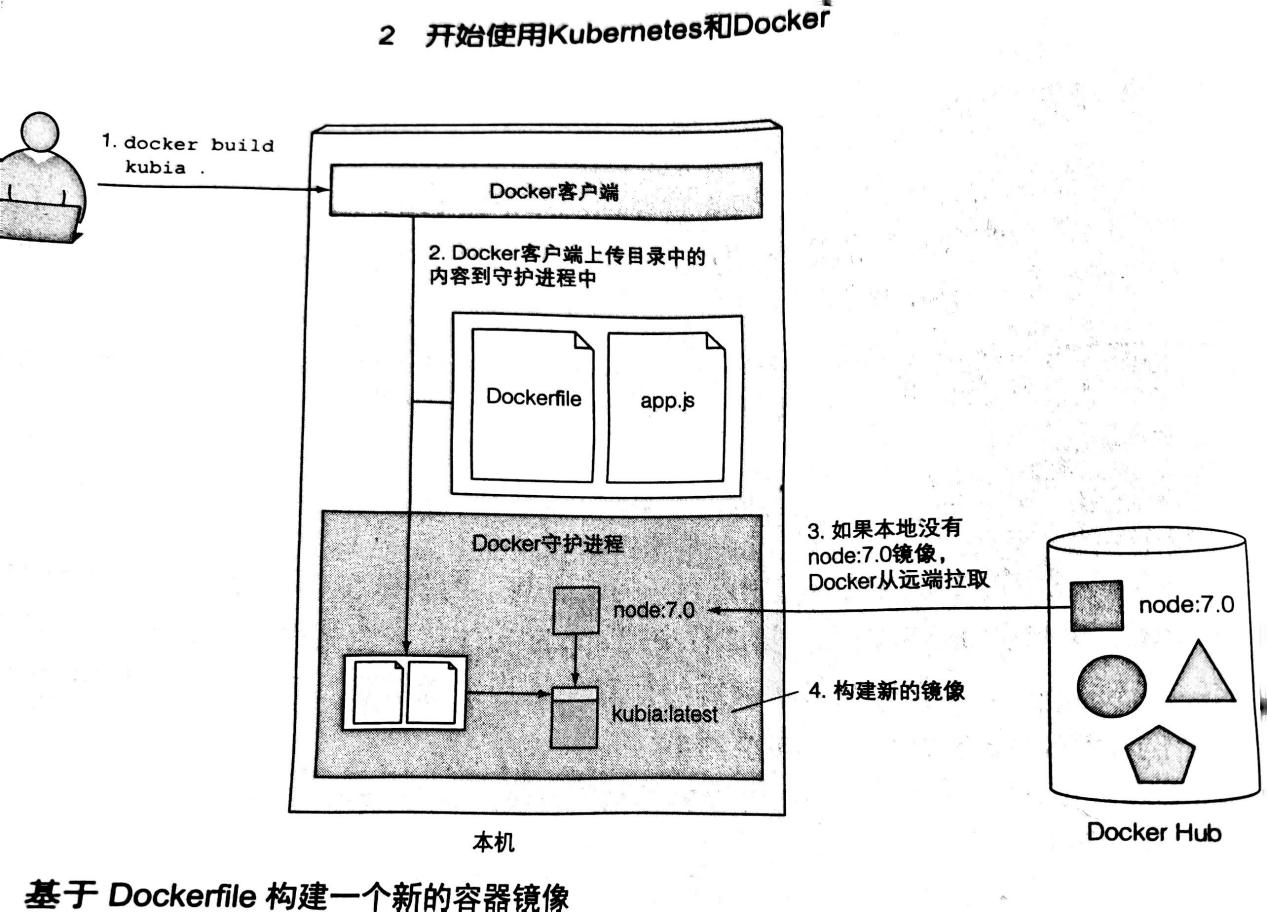


**构建容器镜像**

Docker build -t kubia .

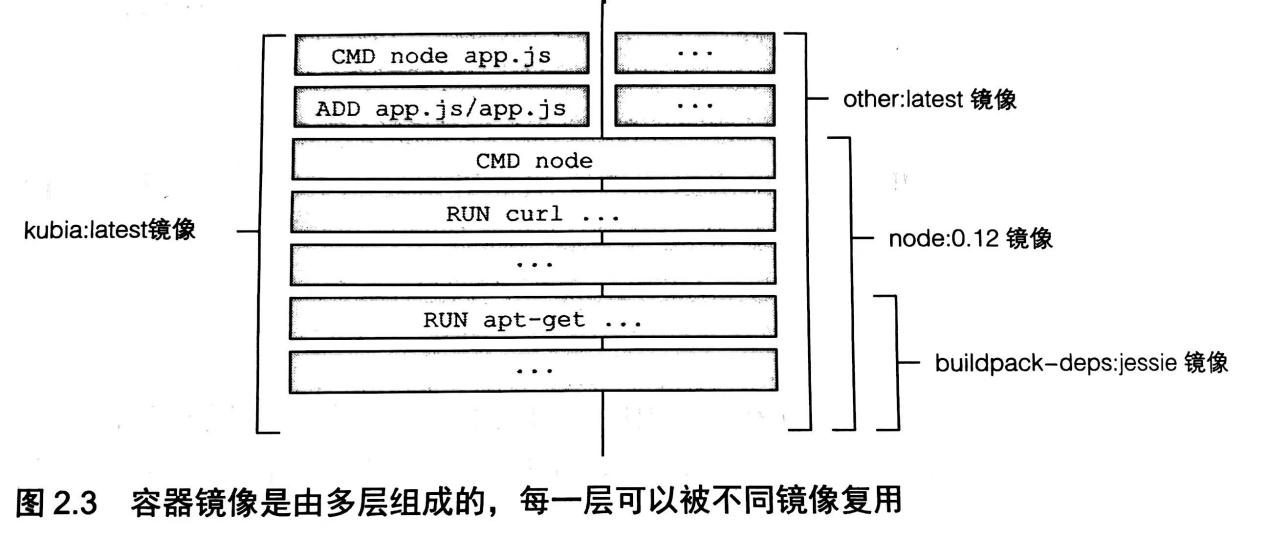
+基于当前目录构建叫kubia的镜像，Docker会在目录中寻找Dockerfile，基于其中指令构建镜像。

（不要在构建目录中包含任何不需要的文件）



**镜像分层**

Dockerfile中每一条单独的指令都创建一个新层，拉取所有分层后创建新层并添加app.js，再创建一层指定镜像运行时执行命令。



**运行容器镜像**

docker run --name kubia-container -p 8080:8080 -d kubia

基于kubia创建一个叫kubia-container的新容器，容器与命令行分离（-d），本机8080端口映射到容器内8080端口（-p 8080:8080），可通过http://localhost:8080访问。

**停止删除容器**

docker stop kubia\_container

docker rm kubia\_container

**向镜像仓库推送镜像**

公开可用的镜像中心Docker Hub（http://hub.docker.com）

先注册Docker Hub ID，再用自己的ID重新标注镜像（创建额外标签，不是重命名）

docker tag kubia ID/kubia

使用docker login和用户ID登录，推送镜像

docker push ID/kubia

**Kubernetes**

使开发者自主部署应用，控制部署频率

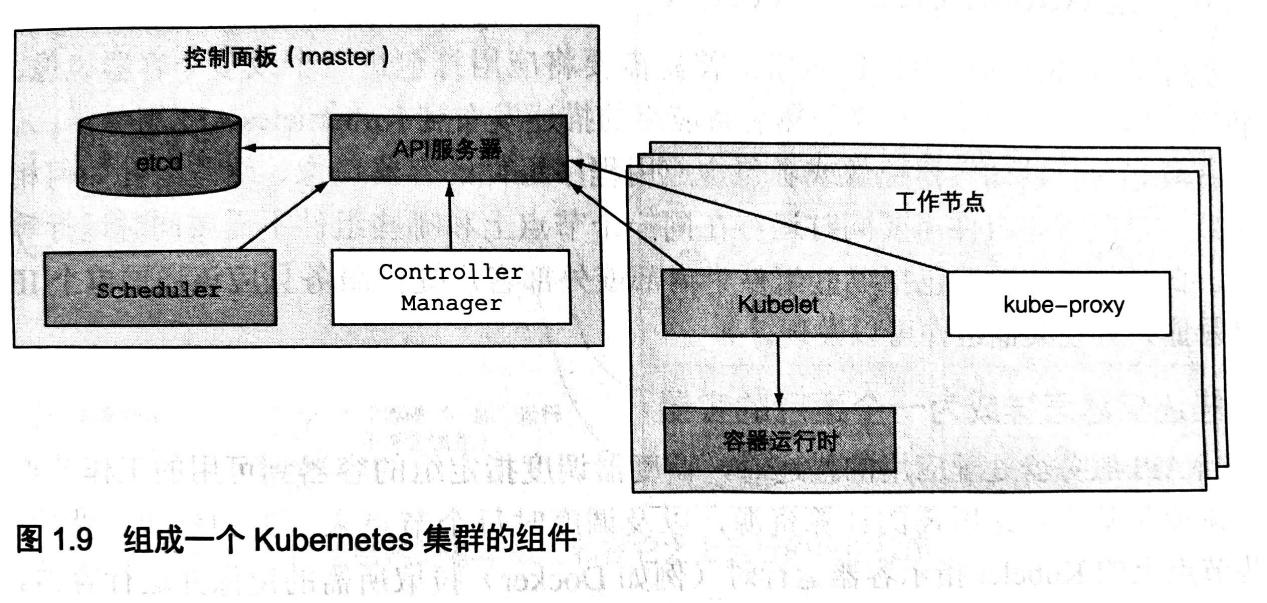
抽象数据中心的硬件基础设施，对外暴露资源池

为每个组件都选择一个合适的服务器，组件间发现并通信

功能：系统由一个主节点和若干工作节点组成，开发者把应用列表提交到主节点，Kubernetes将他们部署到集群的工作节点

提供服务：服务发现、扩容、负载均衡、自恢复、领导者的选举

集群架构：主节点：控制面板，工作节点：运行用户实际部署的应用



**配置Kubernetes集群**

在本地机器上运行单节点Kubernetes集群

在Google Kubernetes Engine上运行托管集群

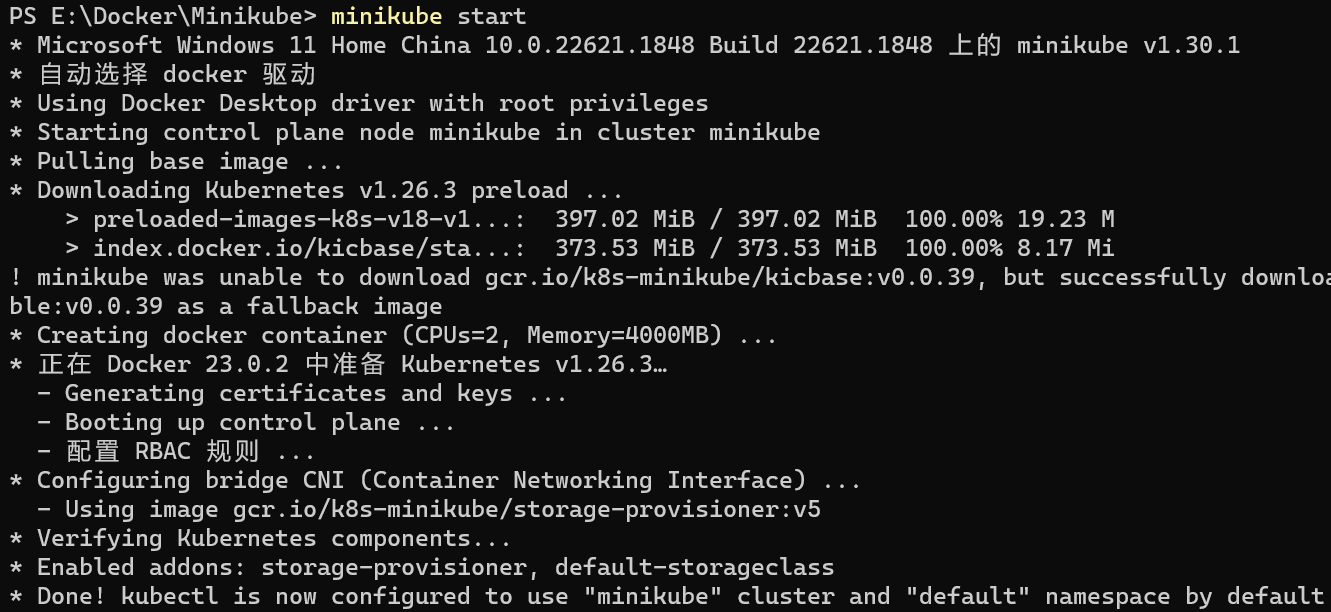
使用kubeadm安装一个集群

**Minikube运行本地单节点Kubernetes集群**

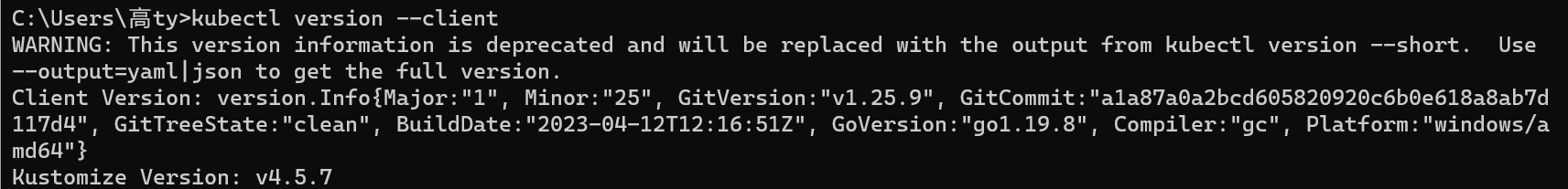
安装minikube

使用minikue启动一个kubernetes集群

minikube start

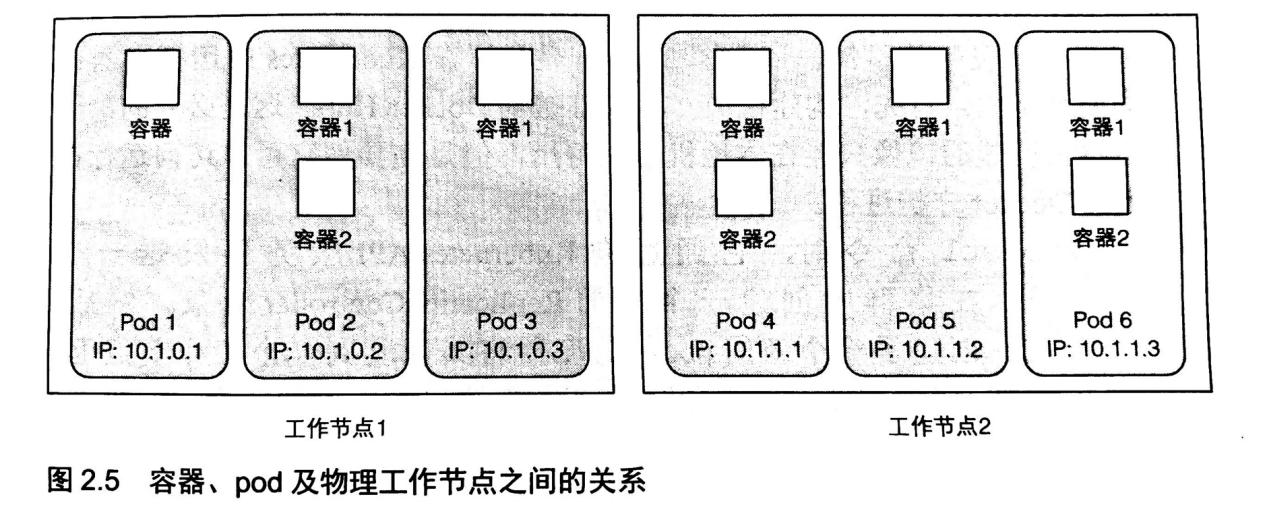


安装客户端kubectl



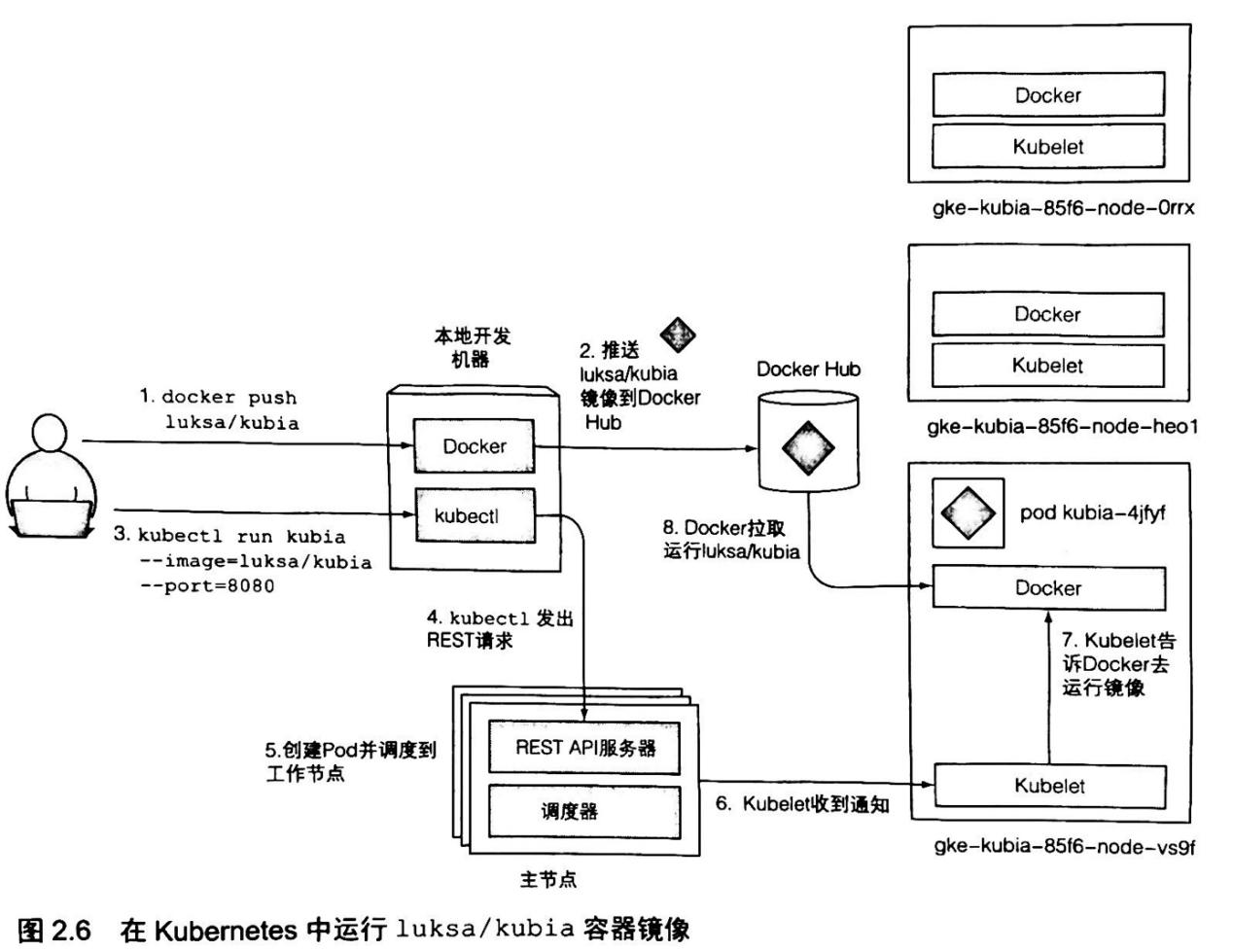
**POD**

一个pod是一组紧密相关的容器，有自己的IP，分布在不同工作节点上，每个容器都运行一个应用进程。



列出pod

kubectl get pods

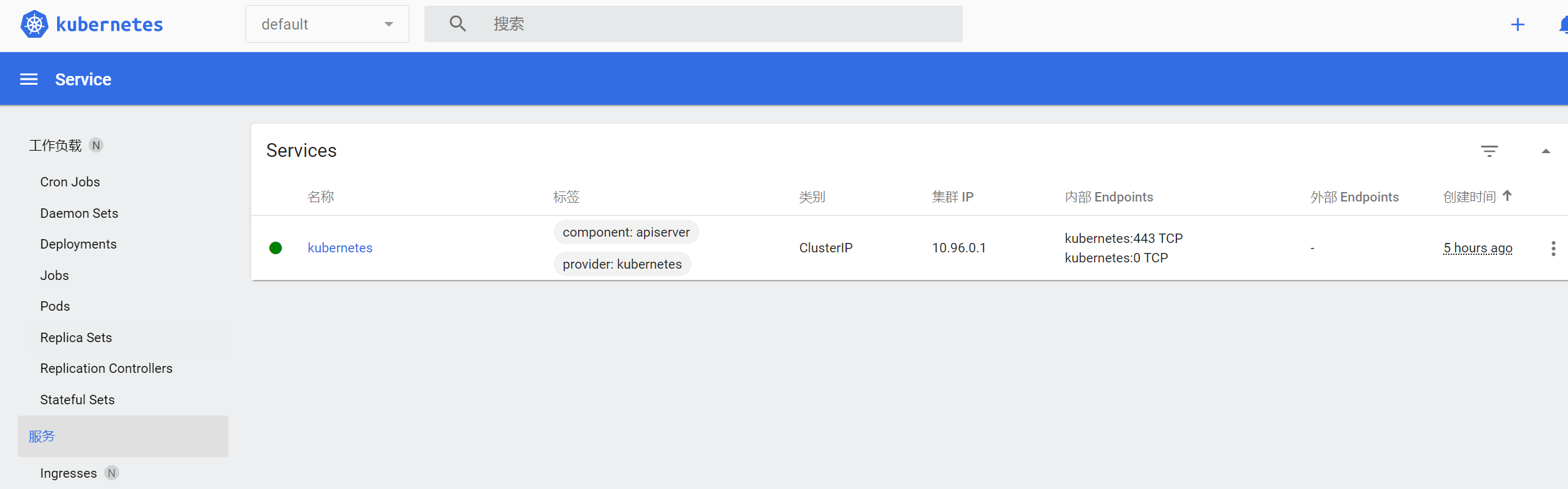


**Kubernetes dashboard**

可以列出部署在集群中的所有pod、ReplicationController、服务和其他部署在集群中的对象，以及创建、修改和删除它们。

minikube dashboard

在默认浏览器中打开。



**案例系统**

常见案例系统：

Hipster Shop：

GitHub 项目地址：https://github.com/GoogleCloudPlatform/microservices-demo

Bookstore：

GitHub 项目地址：https://github.com/ledongthuc/bookstore-microservices

Ticket Booking System：

GitHub 项目地址：https://github.com/lalit527/ticket-booking-microservices

Music Streaming Service：

GitHub 项目地址：<https://github.com/bgogetap/Music-Streaming-Service>

Sock Shop：

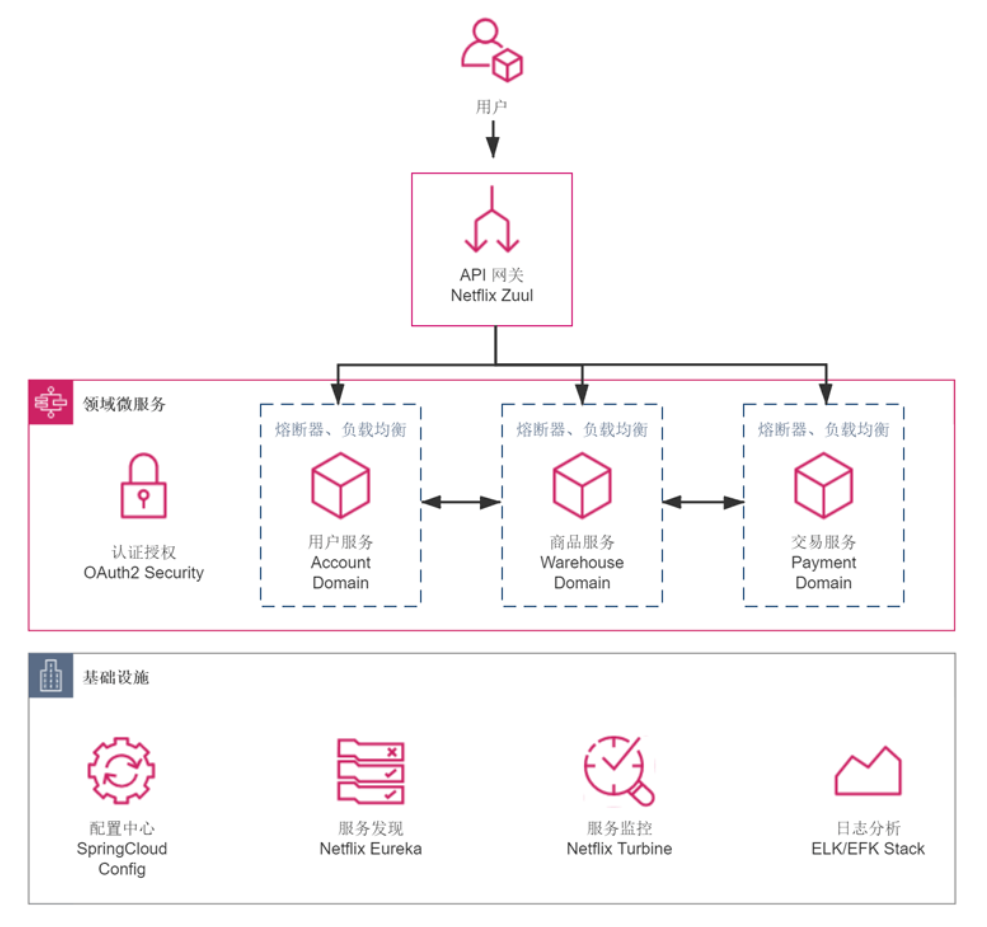
GitHub 项目地址：<https://github.com/microservices-demo/microservices-demo>

选择Fenix's Bookstore

GitHub 项目地址：<https://github.com/fenixsoft/microservice_arch_springcloud>

小书店Fenix's Bookstore生意日益兴隆，客人、货物、营收都在持续增长，业务越发复杂，对信息系统并发与可用方面的要求也越来越高。

准备把单体的Fenix's Bookstore拆分成为“用户”、“商品”、“交易”三个能够独立运行的子系统，它们将在一系列非功能性技术模块（认证、授权等）和基础设施（配置中心、服务发现等）的支撑下互相协作，以统一的API网关对外提供与原来单体系统功能一致的服务，应用视图如下图所示：



通过Docker容器方式运行：

# 下载docker-compose配置文件

$ curl -O [https://raw.githubusercontent.com/fenixsoft/microservice\_arch](https://raw.githubusercontent.com/fenixsoft/microservice_arch_springcloud)

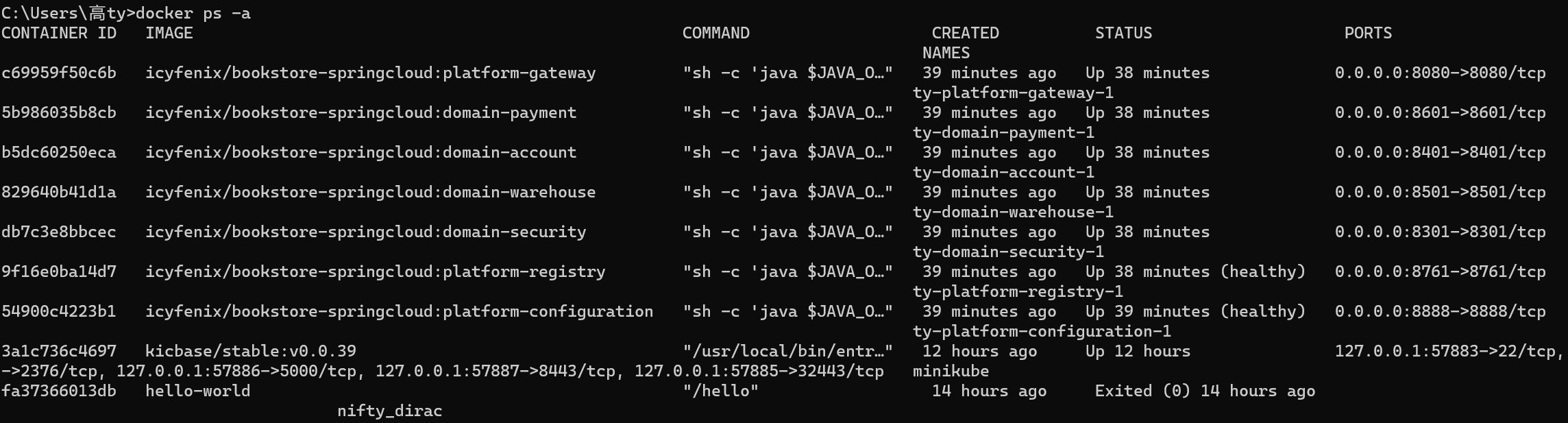
[\_springcloud](https://raw.githubusercontent.com/fenixsoft/microservice_arch_springcloud)/master/docker-compose.yml

# 启动服务

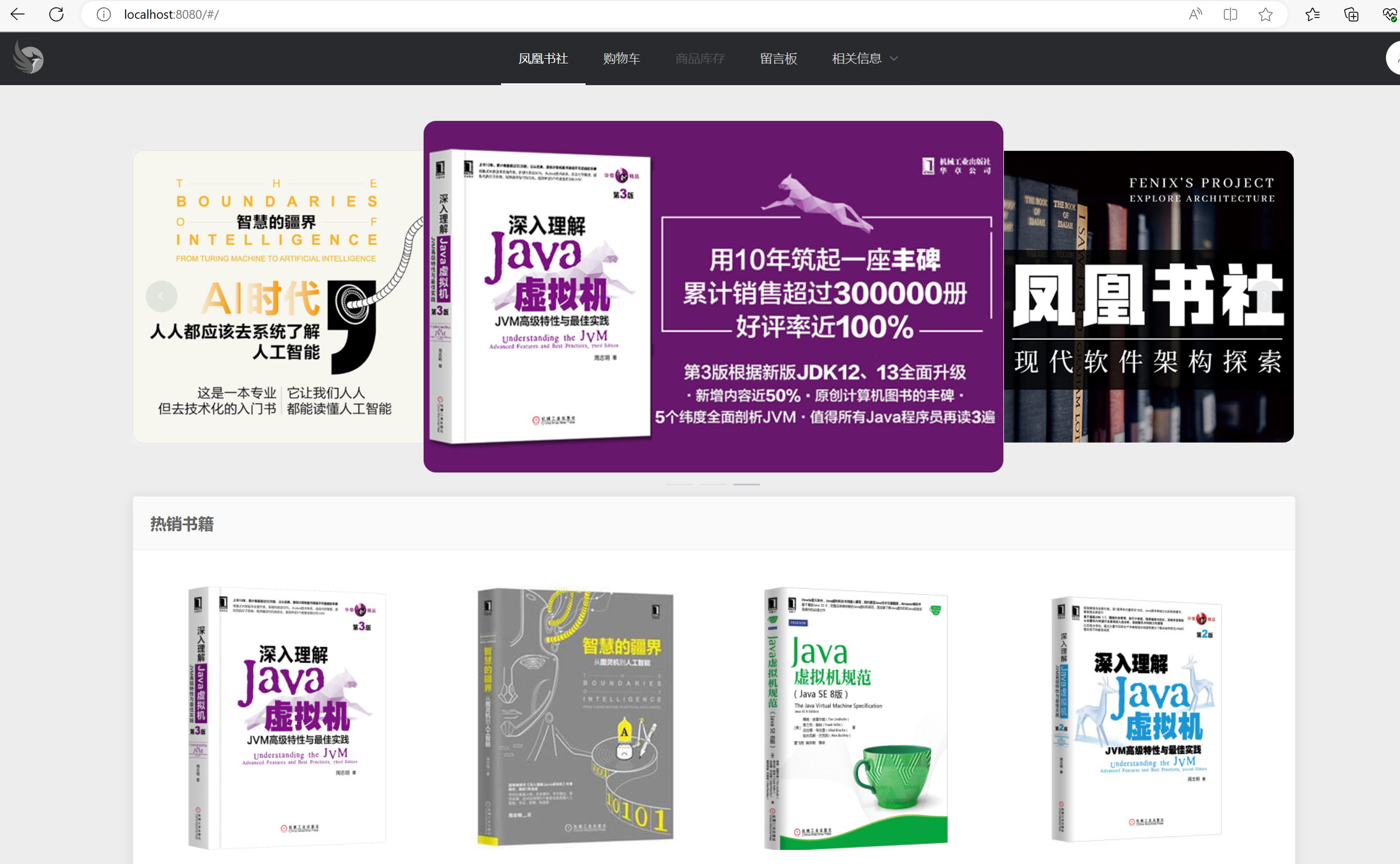
$ docker-compose up

# 查看是否成功运行

$ docker ps -a



在浏览器访问：http://localhost:8080，系统预置了一个用户（user:icyfenix，pw:123456），也可以注册新用户来测试。



**下周计划**

1. 继续研究如何使用minikube部署微服务案例系统
2. 继续阅读《k8s in action》和《凤凰架构》深入了解docker、k8s
3. 学习go语言