# 微服务部署到DOCKER容器和K8S集群

## 课程简介

本课程以工作坊形式，以实战hands-on的方式，分步骤介绍如何把一个具体的微服务产品从最初的产品代码最终落地部署到K8S集群并利用其强大的编排和容错特性提供高可用服务。课程目的是让学员通过一个微服务示例，渐进的掌握并且实践其中每个知识点，包括使用DOCKER COMPOSE功能快速搭建本地运行环境，转换DOCKER COMPOSE的配置文件成为K8S的服务文件，K8S的架构组件, 到 由K8S统一编排来为产品的容错和强伸缩性赋能。

课程示例中使用的微服务是典型的分布式面向服务的系统，不仅继承现在传统的RESTFUL API和消息队列作为服务间主要的交互机制，也扩展使用SignalR技术开启WebSocket通道提供服务器对客户端的反向信息发起，提供非阻塞（non-blocking）技术 实现。和微服务并行工作的还有DevOps中广泛使用的工具服务，Seq被应用于日志收集展示，Grafana服务应用于监测以及报警，Jaeger服务应用于分布式追踪（distributed tracing）

课程配以视频作为自行配置所需工具如K8S集群，DOCKER本地工具，目的是高效利用课程时间用于讲解知识点而不是基本配置。

## 适合人群

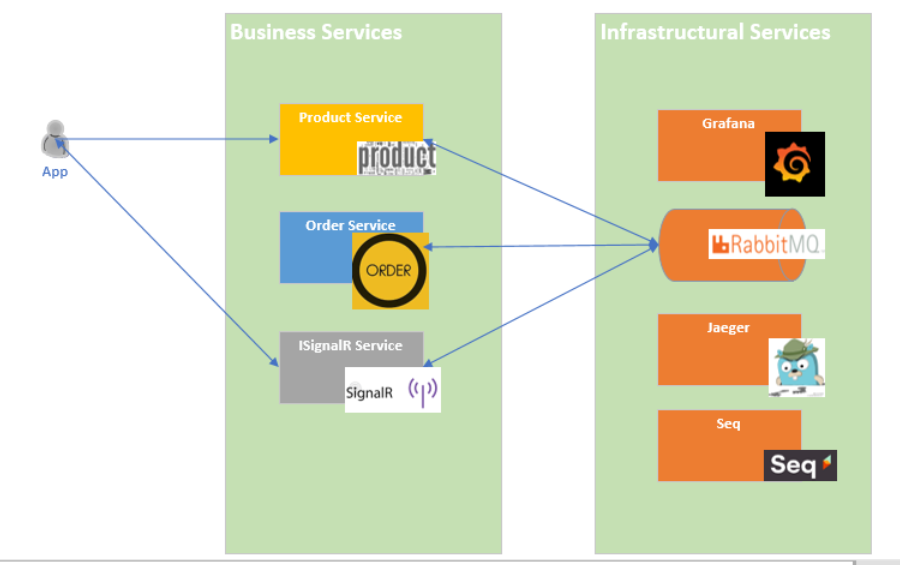
所有对持续集成，DevOps和基础架构为代码（Infrastructure As Code）有兴趣的人员都适合参加

## 学完收获

本课程易于操作，并能保证每个参加者都可以在结束时都能实践, 获取并运行K8S上运行的实例，

## 课程章节

## 课程示例的微服务介绍

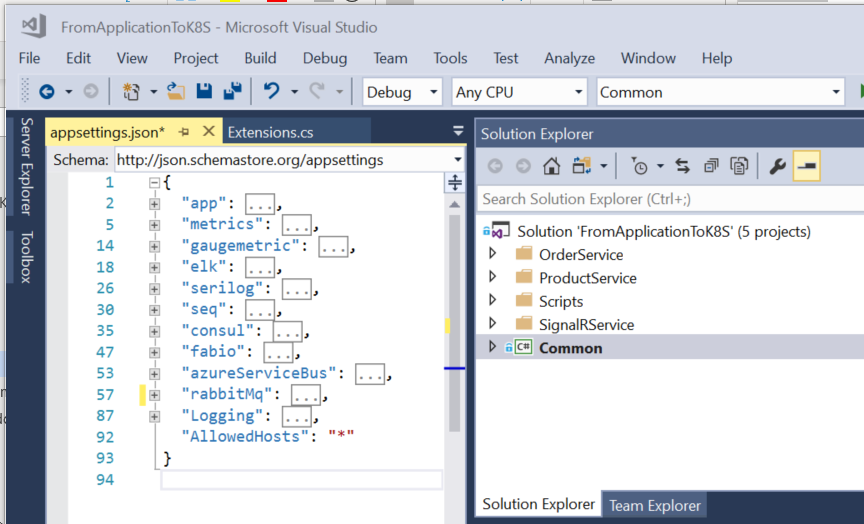


### 服务演示

### 相关资源

* SignalR (基于websocket) **[SignalR - Wikipedia](https://en.wikipedia.org/wiki/SignalR)**
* Jaeger (分布式跟踪) **[Jaeger: open source, end-to-end distributed tracing](https://www.jaegertracing.io/)**
* Grafana (持续监控) **[Grafana: The open observability platform | Grafana Labs](https://grafana.com/)**
* InfluxDB(时序型数据库) **[InfluxDB: Purpose-Built Open Source Time Series Database](https://www.influxdata.com/)**
* Redis
* RabbitMQ

### 代码结构



## 第一步 安装Docker For Windows

### 下载Docker For Windows

链接：

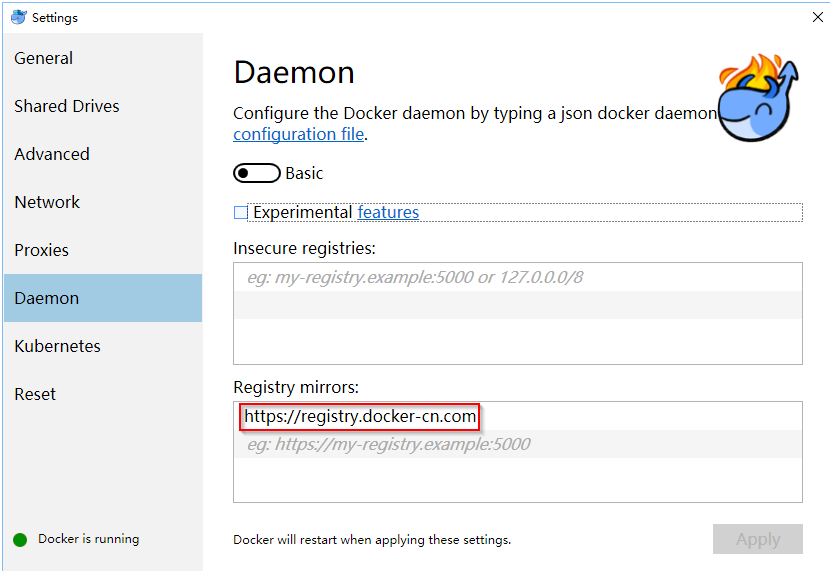
<https://download.docker.com/win/stable/31259/Docker%20for%20Windows%20Installer.exe>

### 安装Docker功能

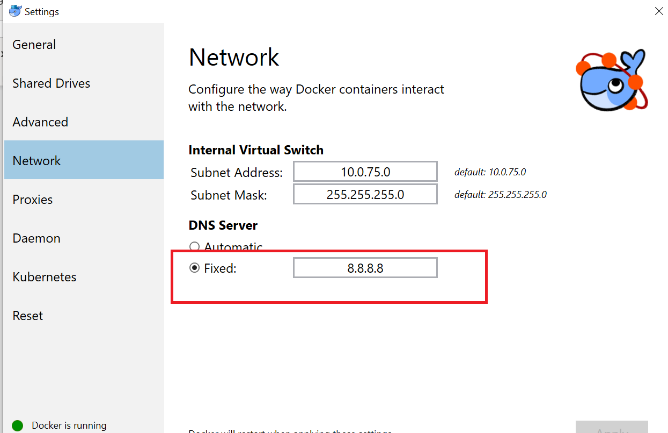
### 安装Kubernetes功能

国内用户需要预设Kubernetes镜像

* 为 Docker daemon 配置 Docker Hub 的中国官方镜像加速 <https://registry.docker-cn.com>



* 在Network选项中按如下配置选择Fixed 8.8.8.8



* 下载<https://github.com/AliyunContainerService/k8s-for-docker-desktop/archive/v2.0.0.2.zip> 并解压
* “以管理员身份运行” 的 PowerShell 中执行 set-ExecutionPolicy RemoteSigned -f

PowerShell 中执行 .\load\_images.ps1

## 第二步 下载课程所需代码

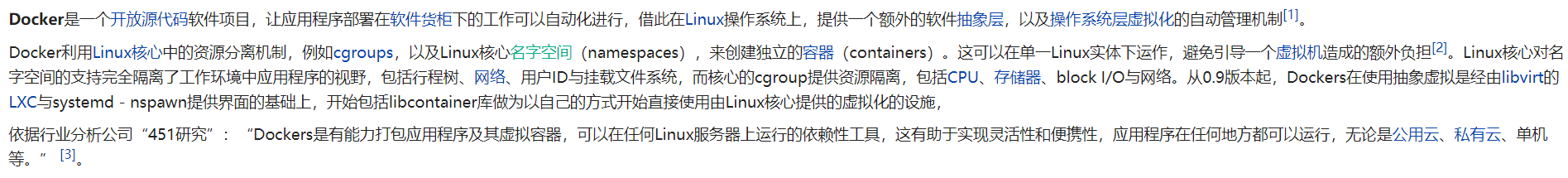
直接下载 <https://github.com/mattcoder2017/FromAppToK8S/archive/master.zip>

并解压缩

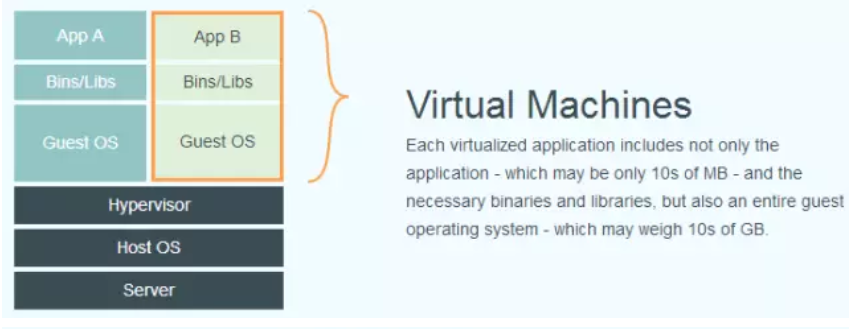
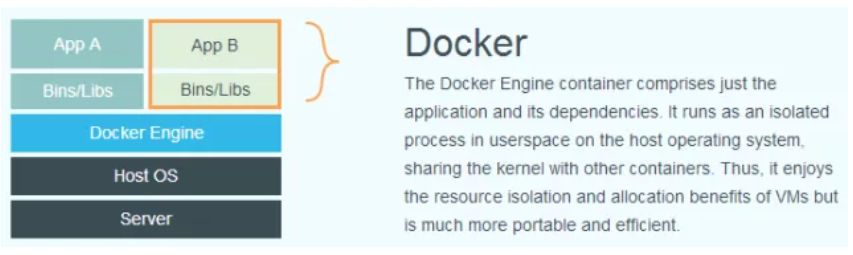
## 第三步 运行Docker Build命令创建镜像（docker image）

### 什么是Docker



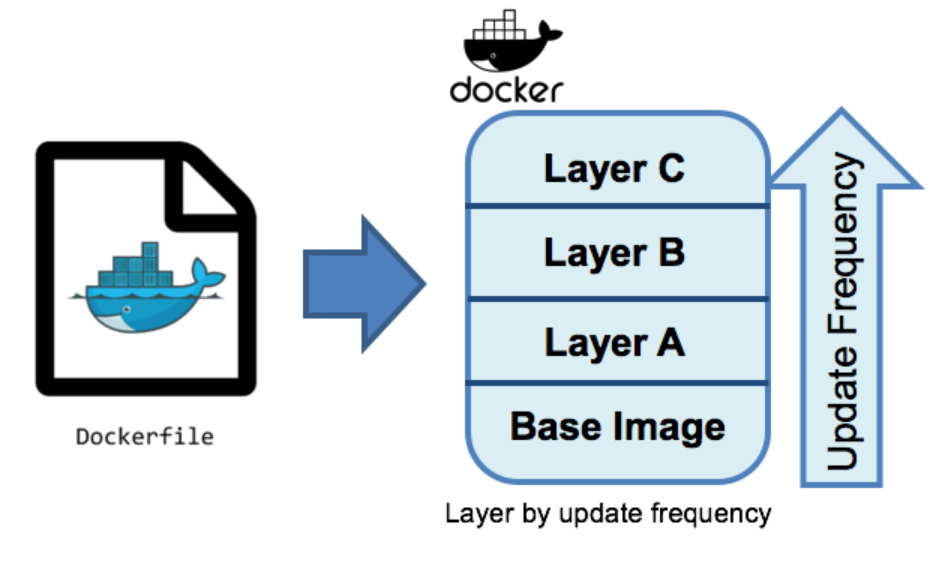


### Docker 与虚拟机的区别

### Docker的工作原理

* + Docker的分层结构



* + 浏览Docker Hub

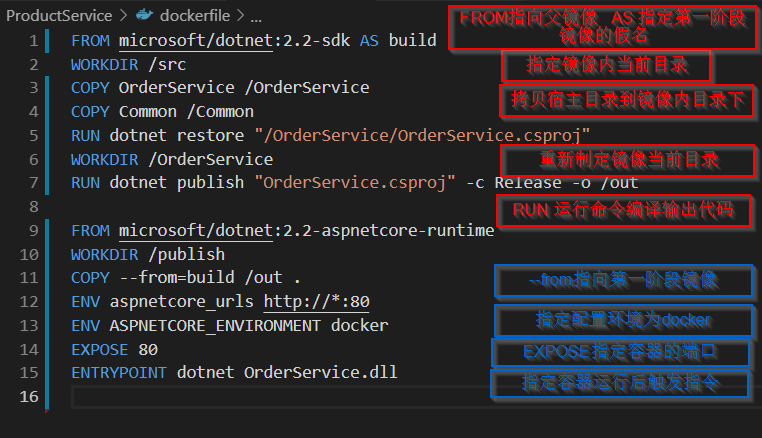
目前 Docker 官方维护了一个公共仓库 Docker Hub，其中已经包括了数量超过 2,650,000 的镜像。大部分需求都可以通过在 Docker Hub 中直接下载镜像来实现

### 执行docker build创建3个服务的docker image

* + Docker build命令



* + DockerFile的结构



### docker run

## 第四步 运行Docker Compose命令批量创建容器（docker containers）

### Docker Compose 是什么

Docker compose 是一个用来把 docker 自动化的组件。

以docker-compose 为指令执行

docker-compose [-f <arg>...] [options] [COMMAND] [ARGS...]有了 docker-compose 你可以把所有繁复的 docker 操作全都一条命令，批量完成。

### Docker compose 的优点

在工作中，经常会碰到需要多个容器相互配合来完成某项任务的情况。例如要实现一个Web项目，除了Web服务容器本身，往往还需要再加上后端的数据库服务容器，甚至还包括负载均衡容器等。  
Compose允许用户通过一个单独的docker-compose.yml模板文件（YAML 格式）来定义一组相关联的应用容器为一个项目（project）。

### 执行docker compose创建第三方服务实例



### 执行docker compose创建业务微服务实例

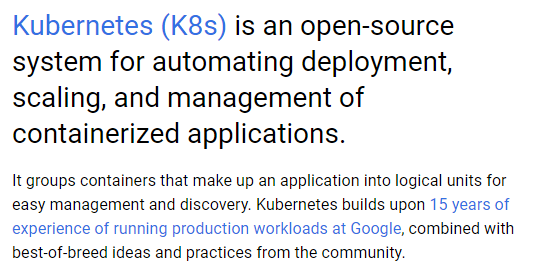


### 检查docker virtual network

### 检查服务状态

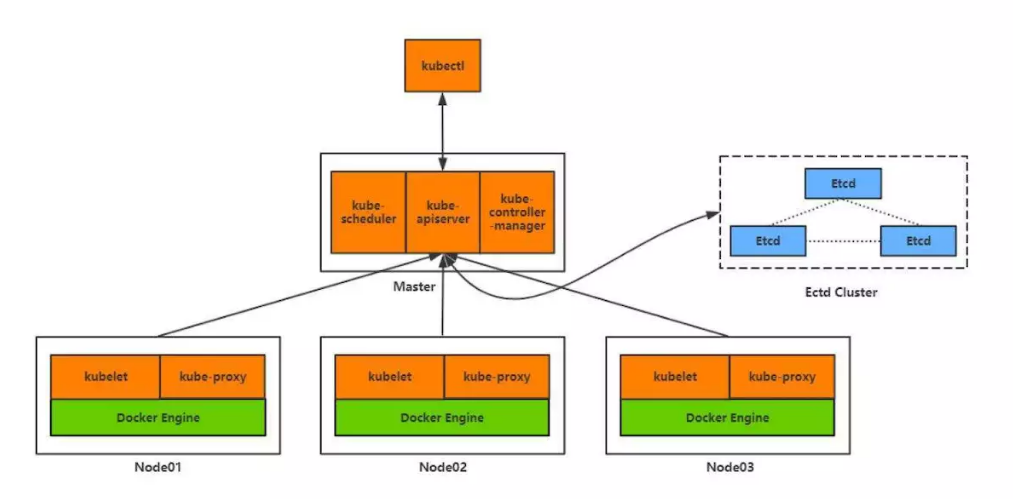
## 第五步 深入K8S pod, deployment, service, ingress等配置技术

### 什么是K8S



Kubernetes（简称K8S）是开源的容器集群管理系统，可以实现容器集群的自动化部署、自动扩缩容、维护等功能。它既是一款容器编排工具，也是全新的基于容器技术的分布式架构领先方案。在Docker技术的基础上，为容器化的应用提供部署运行、资源调度、服务发现和动态伸缩等功能，提高了大规模容器集群管理的便捷性。

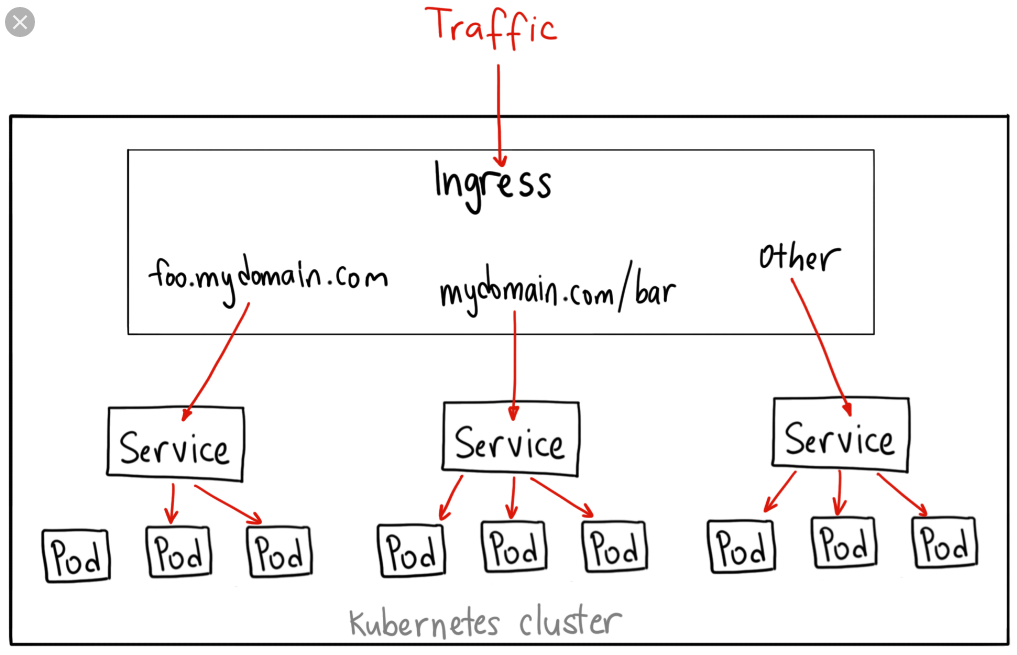
### K8S架构



### K8S基本资源

* pod：的最小调度单元，一个pod中可以有多个容器，多个容器共享网络和存储卷
* deployment：封装了pod/replicaset，可以实现指定pod副本数量，滚动更新，扩容pod，一般一个应用（服务）一个deployment
* service：kubernetes抽象出来一个概念，可以理解为负载均衡器，后端接pod
* ingress： 对集群外部暴露集群内部service的一种方式

### 分层结构图



## 第六步 使用KOMPOSE 轻松转换Docker Compose到K8S配置

### KOMPOSE的用途

Kompose工具能够自动把 Docker Compose 应用转换为 Kubernetes 描述文件。利用简单的 kompose up 命令，可以在 Kubernetes 集群上启动 Compose 应用。

<http://kompose.io/>

下载KOMPOSE: <https://github.com/kubernetes/kompose/releases/download/v1.17.0/kompose-windows-amd64.exe>

### 修改Service用以支持外部访问

### 节点类型

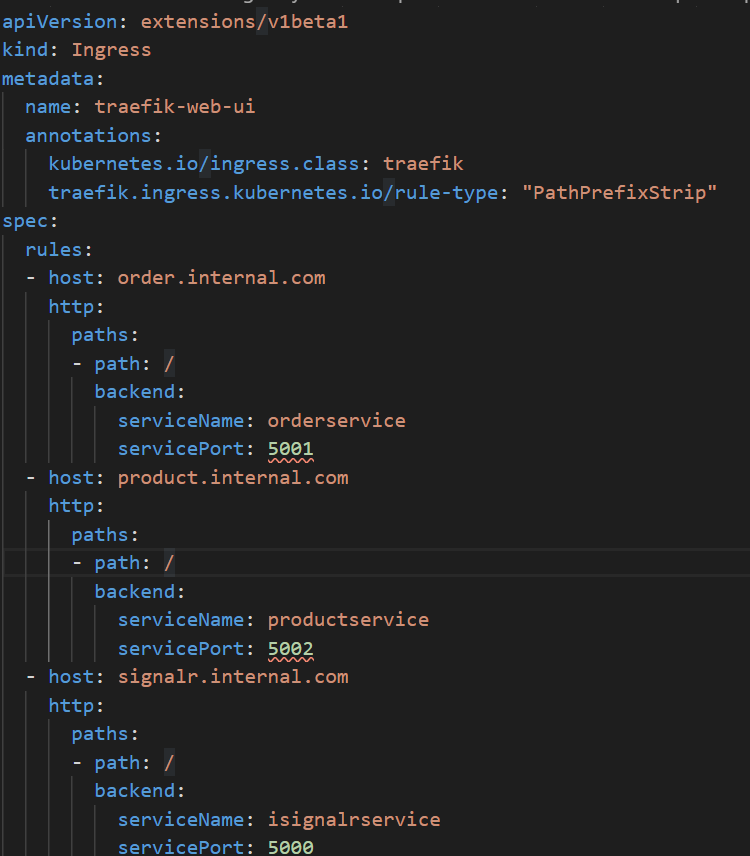
* cluster ip：service在集群中的ip，相当负载均衡器的ip
* nodeport： 对集群外部暴露服务的第二种方式，跟随service配置，让集群中的node节点都监听相应的端口，可以通过node节点访问集群内部service
* loadbalancer： 对集群外部暴露集群内部service的一种方式，一般只有在云平台才能使用

## 第七步 使用TRAEFIK作为Ingress控制器

### 什么是Ingress

### ingress就是从kubernetes集群外访问集群的入口，将用户的URL请求转发到不同的service上。Ingress相当于nginx、apache等负载均衡反向代理服务器，其中还包括规则定义，即URL的路由信息，路由信息的刷新由 [Ingress controller](https://kubernetes.io/docs/concepts/services-networking/ingress/#ingress-controllers) 来提供。

### Ingress 示例



* backend中要配置 default namespace 中启动的 service 名字
* path就是URL地址后的路径，如traefik.frontend.io/path
* host 最好使用 service-name.filed1.filed2.domain-name 这种类似主机名称的命名方式，方便区分服务

根据你自己环境中部署的 service 名称和端口自行修改，有新 service 增加时，修改该文件后可以使用kubectl replace -f traefik.yaml来更新。

### 什么是Traefik

[Traefik](https://traefik.io/) 是一款开源的反向代理与负载均衡工具，它监听后端的变化并自动更新服务配置。Traefik 最大的优点是能够与常见的微服务系统直接整合，可以实现自动化动态配置。目前支持 Docker、Swarm,Marathon、Mesos、Kubernetes、Consul、Etcd、Zookeeper、BoltDB 和 Rest API 等后端模型。

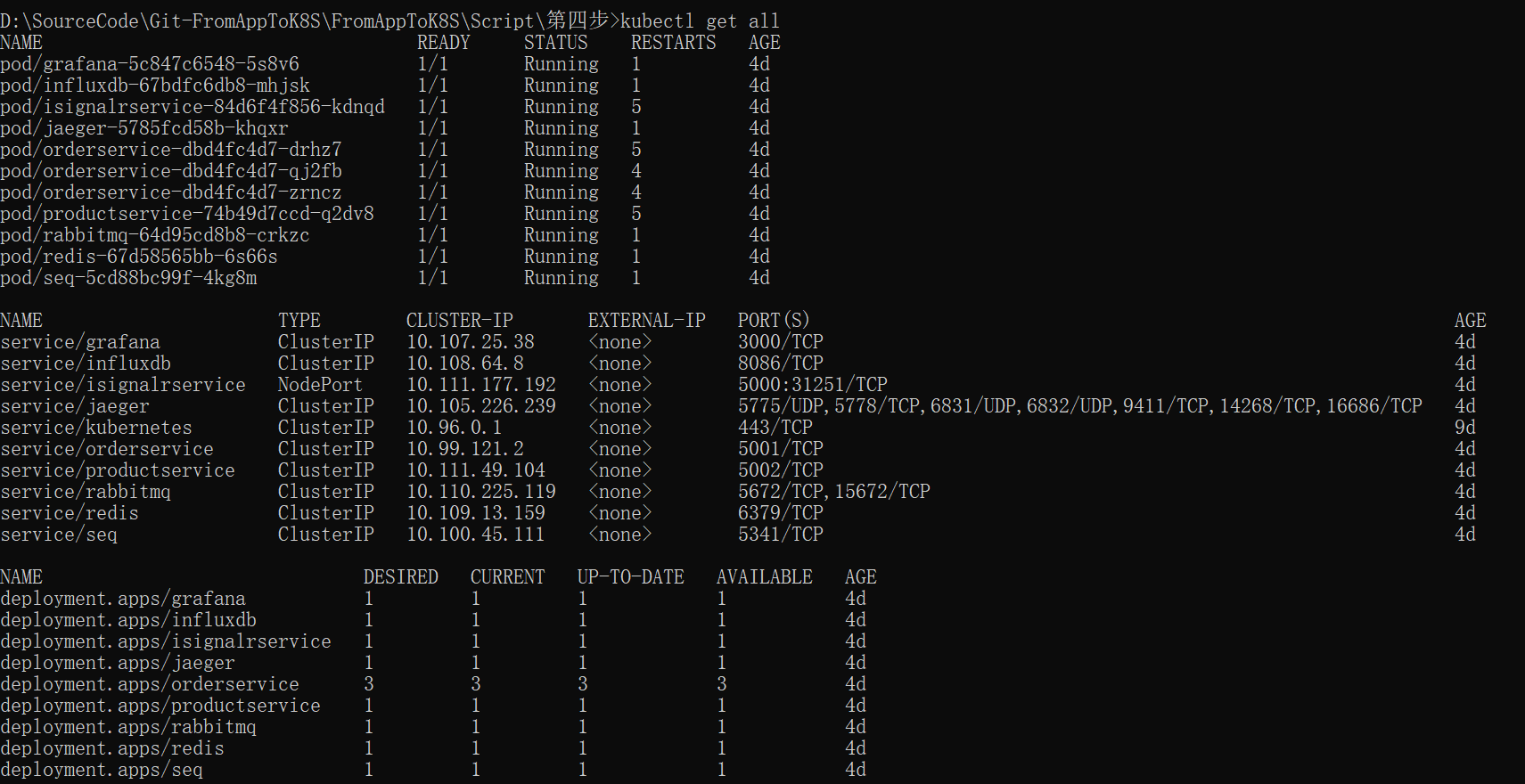
### 安装Traefik Ingress

通过K8S资源文件命令行安装

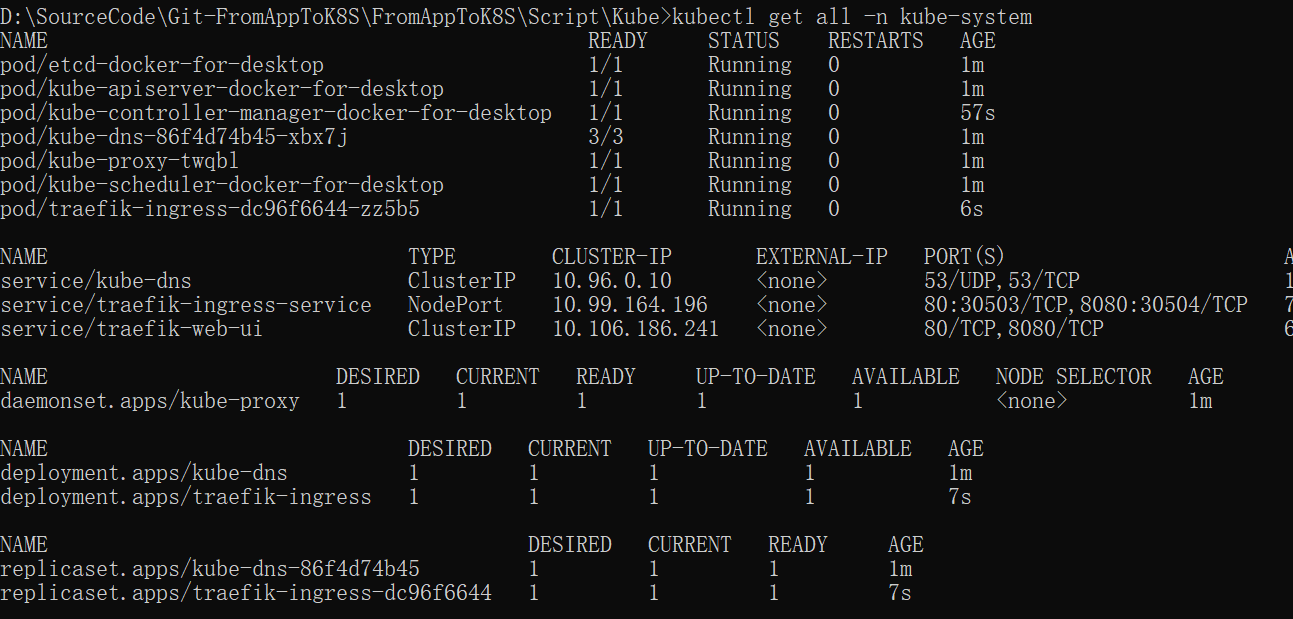
文件系统定位到“第七步“文件夹并运行”kubectl apply -f ./“

## 第八步 部署产品到单机（Windows）的K8S

### 确保第六步的微服务相关K8S资源都已经在运行



### 确保第七步Traefik配置的资源都已经在运行



### 在etc/host文件中加入以下主机名称

127.0.0.1 product.internal.com

127.0.0.1 order.internal.com

127.0.0.1 signalr.internal.com

127.0.0.1 manage.rabbitmq.internal.com

127.0.0.1 manage.seq.internal.com

127.0.0.1 manage.jaeger.internal.com

127.0.0.1 manage.rabbitmq.internal.com

127.0.0.1 manage.grafana.internal.com

127.0.0.1 manage.influxdb.internal.com

### 测试产品

## 第九步 K8S指令对产品进行滚动升级

1. 修改相关代码
2. Build一个新版本的docker image
3. 使用以下指令格式通知K8S针对某个service做滚动升级

Kubectl set image [deployment name] [old image]=[new image]

## 第十步 创建K8S主从集群(Linux based)

详情参考视频

## 第十一步 部署产品到K8S主从集群(Linux based)

### 把所有K8S资源文件拷贝至Master机器

### 运行kubectl apply -f [directory]创建所有服务

### 运行 以下命令观察服务实例的分布情况

### 在etc/host文件中加入以下主机名称

127.0.0.1 product.internal.com

127.0.0.1 order.internal.com

127.0.0.1 signalr.internal.com

127.0.0.1 manage.rabbitmq.internal.com

127.0.0.1 manage.seq.internal.com

127.0.0.1 manage.jaeger.internal.com

127.0.0.1 manage.rabbitmq.internal.com

127.0.0.1 manage.grafana.internal.com

127.0.0.1 manage.influxdb.internal.com

### 测试产品

kubectl get pods -o wide --sort-by="{.spec.nodeName}"

## 第十二步 K8S的数据收集以及横向伸缩

### Metrics server是Kubernetes集群资源使用情况的聚合器

### 从1.8版本开始，Metrics server默认可以通过kube-up.sh脚本以deployment的方式进行部署，也可以通过yaml文件的方式进行部署

### Metrics server收集所有node节点的metrics信息

### Metrics API：

### 通过Metrics API我们可以获取到指定node或者pod的当前资源使用情况，API本身不存储任何信息，所以我们不可能通过API来获取资源的历史使用情况。

### Metrics API的获取路径位于：/apis/metrics.k8s.io/

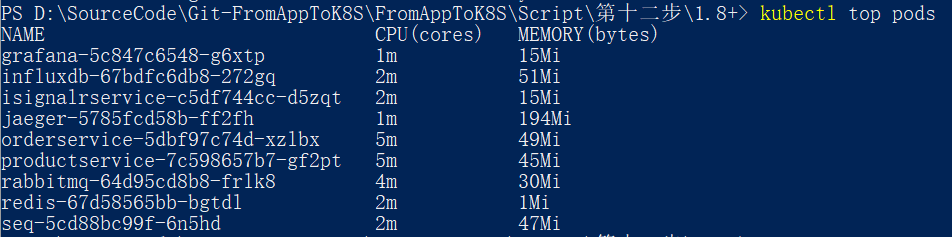
### Horizontal Pod Autoscaler（HPA，Pod水平自动伸缩），根据资源利用率或者自定义指标自动调整replication controller、deployment 或 replica set，实现部署的自动扩展和缩减，让部署的规模接近于实际服务的负载。



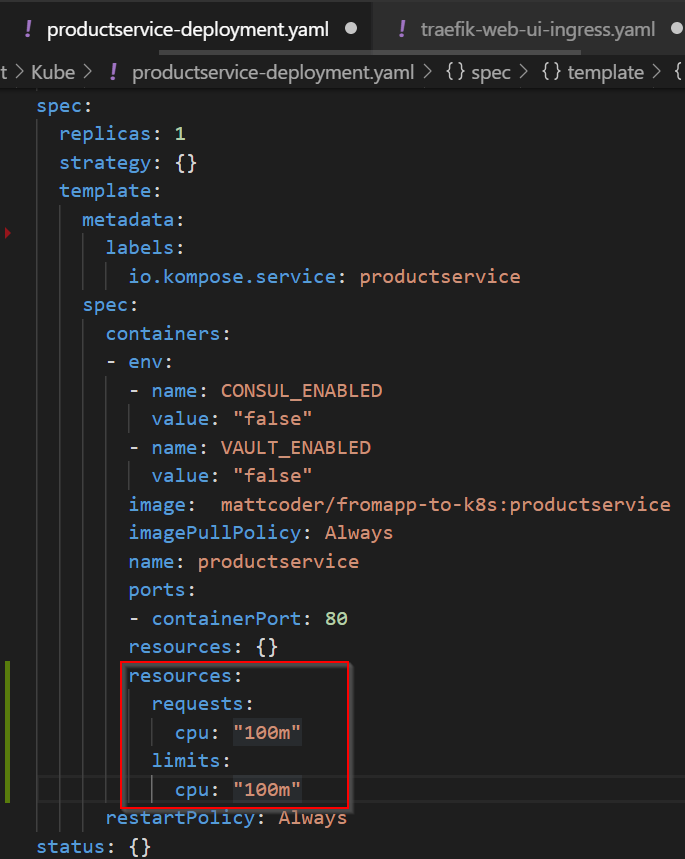
### 部署metrics-server

使用教程文件夹的以yaml文件方式部署

1. 测试metric-server



### 在deployment文件定义pod的指定资源分配



## 第十三步 大功告成，测试K8S的伸缩特性