

- 容器技术是继大数据和云计算之后又一热门技术，而且未来相当一段时间内都会非常流行
- 对 IT 从业者来说，掌握容器技术是市场的需要，也是提升自我价值的重要途径
- 每一轮新技术的兴起，无论对公司还是个人既是机遇也是挑战



Production-Grade Container Orchestration

每天5分钟 玩转Kubernetes

CloudMan 著

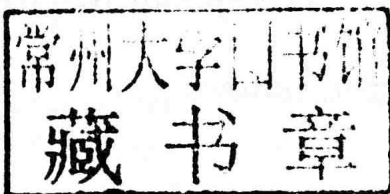
清华大学出版社





每天5分钟 玩转Kubernetes

CloudMan 著



清华大学出版社
北京

内 容 简 介

Kubernetes 是容器编排引擎的事实标准,是继大数据、云计算和 Docker 之后又一热门技术,而且未来相当一段时间内都会非常流行。对于 IT 行业来说,这是一项非常有价值的技术。对于 IT 从业者来说,掌握容器技术既是市场的需要,也是提升自我价值的重要途径。

本书共 15 章,系统介绍了 Kubernetes 的架构、重要概念、安装部署方法、运行管理应用的技术、网络存储管理、集群监控和日志管理等重要内容。书中通过大量实操案例深入浅出地讲解 Kubernetes 核心技术,是一本从入门到进阶的实用 Kubernetes 操作指导手册。读者在学习的过程中,可以跟着教程进行操作,在实践中掌握 Kubernetes 的核心技能。在之后的工作中,则可以将本教程作为参考书,按需查找相关知识点。

本书主要面向微服务软件开发人员,以及 IT 实施和运维工程师等相关人员,也适合作为高等院校和培训学校相关专业的教学参考书。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

每天 5 分钟玩转 Kubernetes / CloudMan 著. — 北京:清华大学出版社,2018(2018.6重印)

ISBN 978-7-302-49667-0

I. ①每… II. ①C… III. ①Linux 操作系统—程序设计 IV. ①TP316.85

中国版本图书馆 CIP 数据核字(2018)第 033859 号

责任编辑:夏毓彦

封面设计:王翔

责任校对:闫秀华

责任印制:董瑾

出版发行:清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址:北京清华大学学研大厦 A 座 邮 编:100084

社总机:010-62770175

邮 购:010-62786544

投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn

质量反馈:010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者:北京富博印刷有限公司

装 订 者:北京市密云县京文制本装订厂

经 销:全国新华书店

开 本:190mm×260mm 印 张:11.5 字 数:294千字

版 次:2018年4月第1版 印 次:2018年6月第2次印刷

印 数:3501~5000

定 价:39.00元

产品编号:079113-01

前言

写在最前面

《每天 5 分钟玩转 Kubernetes》是一本系统学习 Kubernetes 的教程，有下面两个特点：

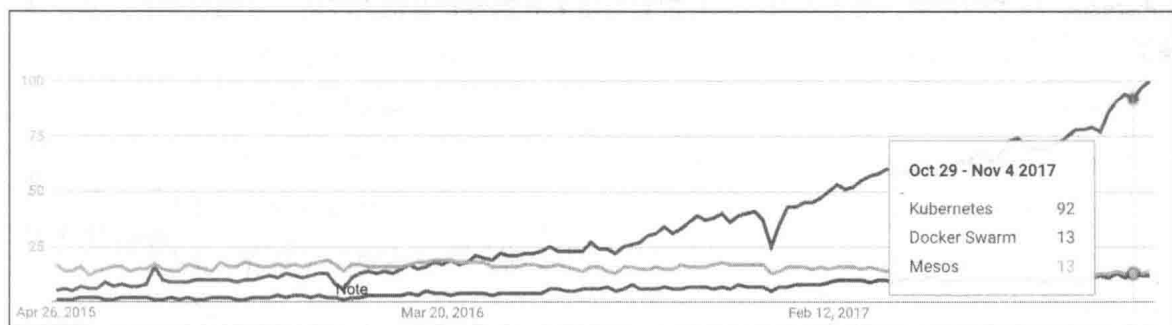
- 系统讲解当前最流行的容器编排引擎 Kubernetes
包括安装部署、应用管理、网络、存储、监控、日志管理等多个方面。
- 重实践并兼顾理论
通过大量实验和操作带领大家学习 Kubernetes。

为什么要写这个

因为 Kubernetes 非常热门，但学习门槛高。

2017 年 9 月，Mesosphere 宣布支持 Kubernetes；10 月，Docker 宣布将在新版本中加入对 Kubernetes 的原生支持。至此，容器编排引擎领域的三足鼎立时代结束，Kubernetes 赢得全面胜利。

其实早在 2015 年 5 月，Kubernetes 在 Google 上的搜索热度就已经超过了 Mesos 和 Docker Swarm，从那之后便是一路飙升，将对手“甩开了十几条街”。



目前，AWS、Azure、Google、阿里云、腾讯云等主流公有云提供的是基于 Kubernetes 的容器服务。Rancher、CoreOS、IBM、Mirantis、Oracle、Red Hat、VMWare 等无数厂商也在大力研发和推广基于 Kubernetes 的容器 CaaS 或 PaaS 产品。可以说，Kubernetes 是当前容器行业最热门的。

每一轮新技术的兴起，无论对公司还是个人既是机会也是挑战。这项新技术未来必将成为主流，那么作为 IT 从业者，正确的做法就是尽快掌握。因为：

(1) 新技术意味着新的市场和新的需求。初期掌握这种技术的人不是很多，而市场需求会越来越大，因而会形成供不应求的卖方市场，物以稀为贵，这对技术人员将是一个难得的价值提升机会。

(2) 学习新技术需要时间和精力，早起步早成材。

机会讲过了，咱们再来看看挑战。

新技术往往意味着技术上的突破和创新，会有不少新的概念和方法。

对于 Kubernetes 这项平台级技术，覆盖的技术范围非常广，包括计算、网络、存储、高可用、监控、日志管理等多个方面，要掌握这些新技术对 IT 老兵尚有不小难度，更别说新人了。

写给谁看

这套教程的目标读者包括：

IT 实施和运维工程师

越来越多的应用将以容器的方式在开发、测试和生产环境中运行。掌握基于 Kubernetes 的容器平台运维能力将成为实施和运维工程师的核心竞争力。

软件开发人员

基于容器的微服务架构（Microservice Architecture）会逐渐成为开发应用系统的主流，而 Kubernetes 将是运行微服务应用的理想平台，市场将需要大量具备 Kubernetes 技能的应用程序开发人员。

我自己

CloudMan 坚信最好的学习方法是分享。编写这本教程的同时也是对自己学习和实践 Kubernetes 技术的总结。对于知识，只有把它写出来并能够让其他人理解，才能说明自己真正掌握了。

著 者

2018 年 1 月

目 录

第 1 章 先把 Kubernetes 跑起来	1
1.1 先跑起来	1
1.2 创建 Kubernetes 集群	2
1.3 部署应用	4
1.4 访问应用	5
1.5 Scale 应用	6
1.6 滚动更新	7
1.7 小结	8
第 2 章 重要概念	9
第 3 章 部署 Kubernetes Cluster	13
3.1 安装 Docker	14
3.2 安装 kubelet、kubeadm 和 kubectl	14
3.3 用 kubeadm 创建 Cluster	14
3.3.1 初始化 Master	14
3.3.2 配置 kubectl	16
3.3.3 安装 Pod 网络	16
3.3.4 添加 k8s-node1 和 k8s-node2	16
3.4 小结	18
第 4 章 Kubernetes 架构	19
4.1 Master 节点	19
4.2 Node 节点	20
4.3 完整的架构图	21
4.4 用例子把它们串起来	22
4.5 小结	24

第 5 章 运行应用	25
5.1 Deployment	25
5.1.1 运行 Deployment	25
5.1.2 命令 vs 配置文件	29
5.1.3 Deployment 配置文件简介	30
5.1.4 伸缩	31
5.1.5 Failover	33
5.1.6 用 label 控制 Pod 的位置	33
5.2 DaemonSet	36
5.2.1 kube-flannel-ds	36
5.2.2 kube-proxy	37
5.2.3 运行自己的 DaemonSet	38
5.3 Job	40
5.3.1 Pod 失败的情况	41
5.3.2 Job 的并行性	43
5.3.3 定时 Job	45
5.4 小结	48
第 6 章 通过 Service 访问 Pod	49
6.1 创建 Service	49
6.2 Cluster IP 底层实现	51
6.3 DNS 访问 Service	53
6.4 外网如何访问 Service	55
6.5 小结	58
第 7 章 Rolling Update	59
7.1 实践	59
7.2 回滚	61
7.3 小结	63
第 8 章 Health Check	64
8.1 默认的健康检查	64
8.2 Liveness 探测	65
8.3 Readiness 探测	67

8.4	Health Check 在 Scale Up 中的应用	69
8.5	Health Check 在滚动更新中的应用	71
8.6	小结	75
第 9 章	数据管理	76
9.1	Volume	76
9.1.1	emptyDir	76
9.1.2	hostPath	78
9.1.3	外部 Storage Provider	79
9.2	PersistentVolume & PersistentVolumeClaim	81
9.2.1	NFS PersistentVolume	81
9.2.2	回收 PV	84
9.2.3	PV 动态供给	86
9.3	一个数据库例子	87
9.4	小结	91
第 10 章	Secret & Configmap	92
10.1	创建 Secret	92
10.2	查看 Secret	93
10.3	在 Pod 中使用 Secret	94
10.3.1	Volume 方式	94
10.3.2	环境变量方式	96
10.4	ConfigMap	97
10.5	小结	100
第 11 章	Helm—Kubernetes 的包管理器	101
11.1	Why Helm	101
11.2	Helm 架构	103
11.3	安装 Helm	104
11.3.1	Helm 客户端	104
11.3.2	Tiller 服务器	105
11.4	使用 Helm	106
11.5	chart 详解	109
11.5.1	chart 目录结构	109

11.5.2	chart 模板	113
11.5.3	再次实践 MySQL chart	115
11.5.4	升级和回滚 release	118
11.5.5	开发自己的 chart	119
11.6	小结	126
第 12 章	网 络	127
12.1	Kubernetes 网络模型	127
12.2	各种网络方案	128
12.3	Network Policy	129
12.3.1	部署 Canal	129
12.3.2	实践 Network Policy	130
12.4	小结	135
第 13 章	Kubernetes Dashboard	136
13.1	安装	136
13.2	配置登录权限	137
13.3	Dashboard 界面结构	139
13.4	典型使用场景	140
13.4.1	部署 Deployment	140
13.4.2	在线操作	141
13.4.3	查看资源详细信息	142
13.4.4	查看 Pod 日志	142
13.5	小结	143
第 14 章	Kubernetes 集群监控	144
14.1	Weave Scope	144
14.1.1	安装 Scope	144
14.1.2	使用 Scope	145
14.2	Heapster	151
14.2.1	部署	151
14.2.2	使用	152
14.3	Prometheus Operator	155
14.3.1	Prometheus 架构	159

14.3.2	Prometheus Operator 架构	161
14.3.3	部署 Prometheus Operator	162
14.4	小结	167
第 15 章	Kubernetes 集群日志管理	168
15.1	部署	168
15.2	小结	173
写在最后	174

第 1 章

先把 Kubernetes 跑起来

Kubernetes (K8s) 是 Google 在 2014 年发布的一个开源项目。

据说 Google 的数据中心里运行着 20 多亿个容器，而且 Google 十年前就开始使用容器技术。

最初，Google 开发了一个叫 Borg 的系统（现在命名为 Omega）来调度如此庞大数量的容器和工作负载。在积累了这么多年的经验后，Google 决定重写这个容器管理系统，并将其贡献到开源社区，让全世界都能受益。

这个项目就是 Kubernetes。简单地讲，Kubernetes 是 Google Omega 的开源版本。

从 2014 年第一个版本发布以来，Kubernetes 迅速获得开源社区的追捧，包括 Red Hat、VMware、Canonical 在内的很多有影响力的公司加入到开发和推广的阵营。目前 Kubernetes 已经成为发展最快、市场占有率最高的容器编排引擎产品。

Kubernetes 一直在快速地开发和迭代。本书我们将以 v1.7 和 v1.8 为基础学习 Kubernetes。我们会讨论 Kubernetes 重要的概念和架构，学习 Kubernetes 如何编排容器，包括优化资源利用、高可用、滚动更新、网络插件、服务发现、监控、数据管理、日志管理等。

下面就让我们开始 Kubernetes 的探险之旅。

1.1 先跑起来

按照一贯的学习思路，我们会在最短时间内搭建起一个可用系统，这样就能够尽快建立起对学习对象的感性认识。先把玩起来，快速了解基本概念、功能和使用场景。

越是门槛高的知识，就越需要有这么一个最小可用系统。如果直接上来就学习理论知识和概念，很容易从入门到放弃。

当然，要搭建这么一个可运行的系统通常也不会太容易，不过很幸运，Kubernetes 官网已经为大家准备好了现成的最小可用系统。

kubernetes.io 开发了一个交互式教程，通过 Web 浏览器就能使用预先部署好的一个 Kubernetes 集群，快速体验 Kubernetes 的功能和应用场景，下面我就带着大家去玩一下。

打开 <https://kubernetes.io/docs/tutorials/kubernetes-basics/>。

页面左边就能看到教程菜单，如图 1-1 所示。



图 1-1

教程会指引大家完成创建 Kubernetes 集群、部署应用、访问应用、扩展应用、更新应用等最常见的使用场景，迅速建立感性认识。

1.2 创建 Kubernetes 集群

点击教程菜单 1. Create a Cluster → Interactive Tutorial - Creating a Cluster，如图 1-2 所示。



图 1-2

显示操作界面，如图 1-3 所示。

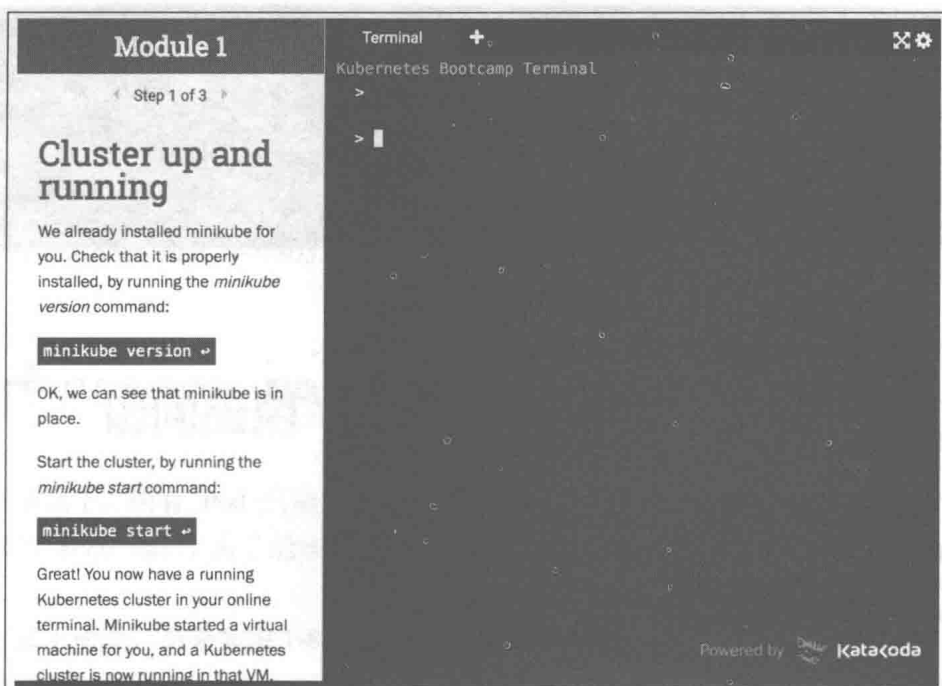


图 1-3

左边部分是操作说明。右边是 Terminal，即命令终端窗口。

按照操作说明，我们在 Terminal 中执行 `minikube start`，然后执行 `kubectl get nodes`，这样就创建好了一个单节点的 kubernetes 集群，如图 1-4 所示。

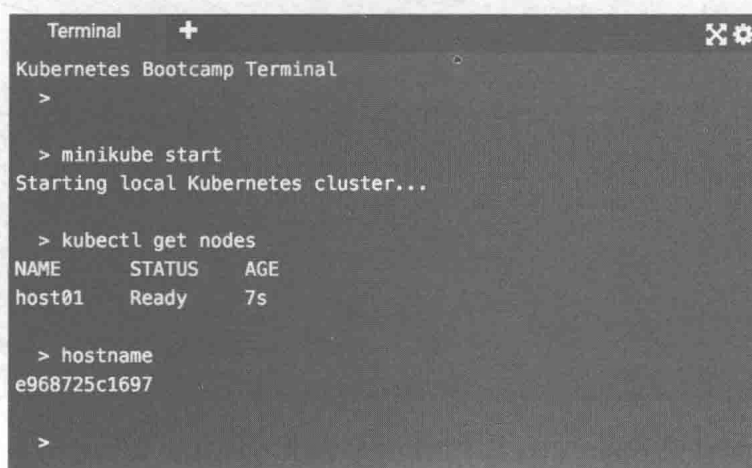


图 1-4

集群的唯一节点为 `host01`，需要注意的是当前执行命令的地方并不是 `host01`。我们是通过 Kubernetes 的命令行工具 `kubectl` 远程管理集群。

执行 `kubectl cluster-info` 查看集群信息，如图 1-5 所示。

```

> kubectl cluster-info
Kubernetes master is running at http://host01:8080
heapster is running at http://host01:8080/api/v1/proxy/namespaces/kube-system/services/heapster
kubernetes-dashboard is running at http://host01:8080/api/v1/proxy/namespaces/kube-system/services/kubernetes-dashboard
monitoring-grafana is running at http://host01:8080/api/v1/proxy/namespaces/kube-system/services/monitoring-grafana
monitoring-influxdb is running at http://host01:8080/api/v1/proxy/namespaces/kube-system/services/monitoring-influxdb

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

>

```

图 1-5

heapster、kubernetes-dashboard 都是集群中运行的服务。

注意：为节省篇幅，在后面的演示中，我们将简化操作步骤，详细的说明和完整步骤请参考官网在线文档。

1.3 部署应用

执行命令：

```

kubectl run kubernetes-bootcamp \
  --image=docker.io/jocatalin/kubernetes-bootcamp:v1 \
  --port=8080

```

这里我们通过 `kubectl run` 部署了一个应用，命名为 `kubernetes-bootcamp`，如图 1-6 所示。Docker 镜像通过 `--image` 指定。
`--port` 设置应用对外服务的端口。

```

Terminal +
> kubectl run kubernetes-bootcamp \
>   --image=docker.io/jocatalin/kubernetes-bootcamp:v1 \
>   --port=8080
deployment "kubernetes-bootcamp" created

> █

```

图 1-6

这里 Deployment 是 Kubernetes 的术语，可以理解为应用。

Kubernetes 还有一个重要术语 Pod。

Pod 是容器的集合，通常会将紧密相关的一组容器放到一个 Pod 中，同一个 Pod 中的所有容器共享 IP 地址和 Port 空间，也就是说它们在一个 `network namespace` 中。

Pod 是 Kubernetes 调度的最小单位，同一 Pod 中的容器始终被一起调度。

运行 `kubectl get pods`，查看当前的 Pod，如图 1-7 所示。

```
> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-390780338-q9p1t  1/1     Running   0           11m
>
```

图 1-7

kubernetes-bootcamp-390780338-q9p1t 就是应用的 Pod。

1.4 访问应用

默认情况下，所有 Pod 只能在集群内部访问。对于上面这个例子，要访问应用只能直接访问容器的 8080 端口。为了能够从外部访问应用，我们需要将容器的 8080 端口映射到节点的端口。

执行如下命令，结果如图 1-8 所示。

```
kubectl expose deployment/kubernetes-bootcamp \
  --type="NodePort" \
  --port 8080
```

```
> kubectl expose deployment/kubernetes-bootcamp \
>   --type="NodePort" \
>   --port 8080
service "kubernetes-bootcamp" exposed
>
```

图 1-8

执行命令 `kubectl get services`，可以查看应用被映射到节点的哪个端口，如图 1-9 所示。

```
> kubectl get services
NAME            CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes     10.0.0.1     <none>        443/TCP          2m
kubernetes-bootcamp 10.0.0.131   <nodes>       8080:32320/TCP  1m
>
```

图 1-9

这里有两个 service，可以将 service 暂时理解为端口映射，后面我们会详细讨论。

Kubernetes 是默认的 service，暂时不用考虑。kubernetes-bootcamp 是我们应用的 service，8080 端口已经映射到 host01 的 32320 端口，端口号是随机分配的，可以执行如下命令访问应用，结果如图 1-10 所示。

```
curl host01:32320
```

```
> curl host01:32320
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-390780338-q9p1t | v=1
>
```

图 1-10

1.5 Scale 应用

默认情况下应用只会运行一个副本，可以通过 `kubectl get deployments` 查看副本数，如图 1-11 所示。

```
> kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp 1          1         1             1           14m
>
```

图 1-11

执行如下命令将副本数增加到 3 个，如图 1-12 所示。

```
kubectl scale deployments/kubernetes-bootcamp --replicas=3
```

```
> kubectl scale deployments/kubernetes-bootcamp --replicas=3
deployment "kubernetes-bootcamp" scaled
>

> kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp 3          3         3             3           17m
>
```

图 1-12

通过 `kubectl get pods` 可以看到当前 Pod 增加到 3 个，如图 1-13 所示。

```
> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-390780338-12sbg 1/1     Running   0           1m
kubernetes-bootcamp-390780338-q9p1t 1/1     Running   0           19m
kubernetes-bootcamp-390780338-swp7   1/1     Running   0           1m
>
```

图 1-13

通过 `curl` 访问应用，可以看到每次请求发送到不同的 Pod，3 个副本轮询处理，这样就实现了负载均衡，如图 1-14 所示。


```

> curl host01:32320
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-390780338-12sbg | v=1

> curl host01:32320
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-390780338-swpv7 | v=1

> curl host01:32320
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-390780338-q9p1t | v=1

> curl host01:32320
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-390780338-12sbg | v=1

>

```

图 1-14

要 scale down 也很方便，执行下列命令，结果如图 1-15 所示。

```
kubectl scale deployments/kubernetes-bootcamp --replicas=2
```

```

> kubectl scale deployments/kubernetes-bootcamp --replicas=2
deployment "kubernetes-bootcamp" scaled

> kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp 2          2         2             2           25m

> kubectl get pods
NAME                READY     STATUS    RESTARTS   AGE
kubernetes-bootcamp-390780338-12sbg 1/1       Running   0           8m
kubernetes-bootcamp-390780338-q9p1t 1/1       Running   0           25m
kubernetes-bootcamp-390780338-swpv7 1/1       Terminating 0           8m

```

图 1-15

从图 1-15 中可以看到，其中一个副本被删除了。

1.6 滚动更新

当前应用使用的 image 版本为 v1，执行如下命令将其升级到 v2，结果如图 1-16 所示。

```
kubectl set image deployments/kubernetes-bootcamp
kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2
```

```

> kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2
deployment "kubernetes-bootcamp" image updated

> kubectl get pods
NAME                READY     STATUS    RESTARTS   AGE
kubernetes-bootcamp-2100875782-2q5k8 0/1       ContainerCreating 0           6s
kubernetes-bootcamp-2100875782-sbwkc 0/1       ContainerCreating 0           4s
kubernetes-bootcamp-390780338-12sbg 1/1       Terminating 0           19m
kubernetes-bootcamp-390780338-q9p1t 1/1       Running      0           36m

> kubectl get pods
NAME                READY     STATUS    RESTARTS   AGE
kubernetes-bootcamp-2100875782-2q5k8 1/1       Running     0           15s
kubernetes-bootcamp-2100875782-sbwkc 1/1       Running     0           13s
kubernetes-bootcamp-390780338-12sbg 1/1       Terminating 0           19m
kubernetes-bootcamp-390780338-q9p1t 1/1       Terminating 0           37m

> kubectl get pods
NAME                READY     STATUS    RESTARTS   AGE
kubernetes-bootcamp-2100875782-2q5k8 1/1       Running     0           1m
kubernetes-bootcamp-2100875782-sbwkc 1/1       Running     0           1m

>

```

图 1-16