

08 | 搭建私有Serverless（一）：K8s和云原生CNCF

2020-05-04 蒲松洋

Serverless入门课

[进入课程 >](#)



讲述：蒲松洋

时长 20:28 大小 18.75M



你好，我是秦粤。上节课我们只是用 Docker 部署了 index.js，如果我们将所有拆解出来的微服务都用 Docker 独立部署，我们就要同时管理多个 Docker 容器，也就是 Docker 集群。如果是更复杂一些的业务，可能需要同时管理几十甚至上百个微服务，显然我们手动维护 Docker 集群的效率就太低了。而容器即服务 CaaS，恰好也需要集群的管理工具。我们也知道 FaaS 的底层就是 CaaS，那 CaaS 又是如何管理这么多函数实例的呢？怎么做才能提升效率？

我想你应该听过 Kubernetes，它也叫 K8s（后面统一简称 K8s），用于自动部署、打 ☆，管理容器化应用程序的开源系统，是 Docker 集群的管理工具。为了解决上述问题，其实我们就可以考虑使用它。K8s 的好处就在于，它具备跨环境统一部署的能力。

这节课，我们就试着在本地环境中搭建 K8s 来管理我们的 Docker 集群。但正常情况下，这个场景需要几台机器才能完成，而通过 Docker，我们还是可以用一台机器就可以在本地搭建一个低配版的 K8s。

下节课，我们还会在今天内容的基础上，用 K8s 的 CaaS 方式实现一套 Serverless 环境。通过这两节课的内容，你就可以完整地搭建出属于自己的 Serverless 了。

话不多说，我们现在就开始，希望你能跟着我一起多动手。

PC 上的 K8s

那在开始之前，我们先得把安装问题解决了，这部分可能会有点小困难，所以我也给你详细讲下。

首先我们需要安装 [🔗 kubectl](#)，它是 K8s 的命令行工具。

你需要在你的 PC 上安装 K8s，如果你的操作系统是 MacOS 或者 Windows，那么就比较简单了，桌面版的 Docker 已经自带了 [🔗 K8s](#)；其它操作系统的同学需要安装 [🔗 minikube](#)。


不过，要顺利启动桌面版 Docker 自带的 K8s，你还得解决国内 Docker 镜像下载不了的问题，这里请你先下载 [🔗 第 8 课的代码](#)。接着，请你跟着我的步骤进行操作：

1. 开通阿里云的 [🔗 容器镜像仓库](#)；
2. 在阿里云的容器镜像服务里，找到镜像加速器，复制你的镜像加速器地址；
3. 打开桌面版 Docker 的控制台，找到 Docker Engine。

 复制代码

```
1 {
2   "registry-mirrors" : [
3     "https://你的加速地址.mirror.aliyuncs.com"
4   ],
5   "debug" : true,
6   "experimental" : true
7 }
```

4. 预下载 K8s 所需要的所有镜像，执行我目录下的 docker-k8s-prefetch.sh，如果你是 Windows 操作系统，建议使用 gitBash[1];

 复制代码

```
1 chmod +x docker-k8s-prefetch.sh
2 ./docker-k8s-prefetch.sh
```

5. 上面拉取完运行 K8s 所需的 Docker 镜像，你就可以在桌面版 Docker 的 K8s 项中，勾选启动 K8s 了。


现在，K8s 就能顺利启动了，启动成功后，请你继续执行下面的命令。

查看安装好的 K8s 系统运行状况。

 复制代码


```
1 kubectl get all -n kube-system
```

查看 K8s 有哪些配置环境，对应 ~/.kube/config。

 复制代码

```
1 kubectl config get-contexts
```

查看当前 K8s 环境的整体情况。

 复制代码

```
1 kubectl get all
```

按照我的流程走，在大部分的机器上本地运行 K8s 都是没有问题的，如果你卡住了，请在留言区告知我，我帮你解决问题。

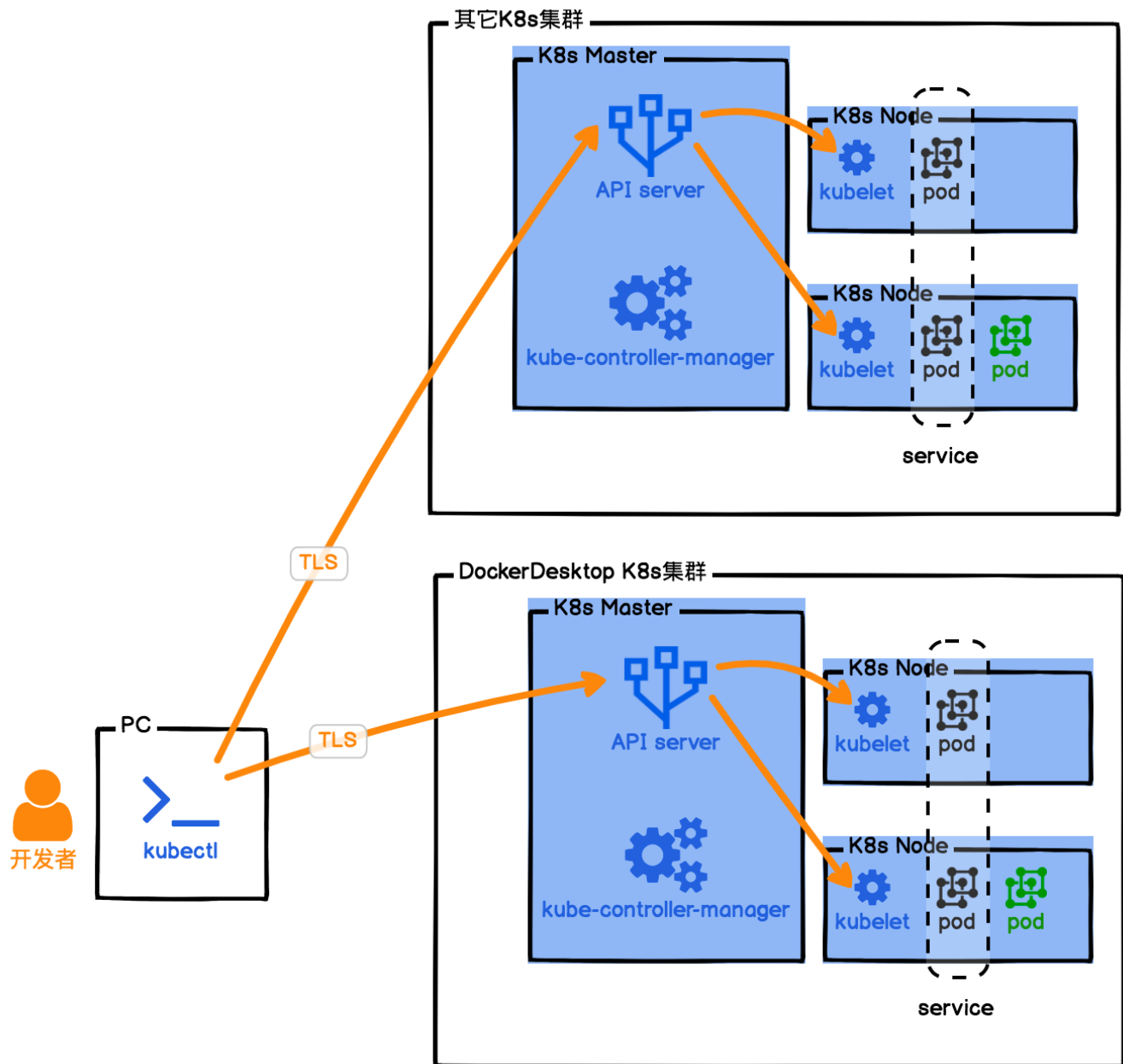
K8s 介绍

安装完 K8s 之后，我们其实还是要简单地了解下 K8s，这对于你后面应用它有重要的意义。

我想你应该知道，K8s 是云原生 Cloud Native[2] 的重要组成部分（目前，K8s 的文档[3]已经全面中文化，建议你多翻阅），而云原生其实就是一套通用的云服务商解决方案。在我们的前几节课中，就有同学问：“如果我使用了某个运营商的 Serverless，就和这个云服务商强制绑定了怎么办？我是不是就没办法使用其他运营商的服务了？”这就是服务商锁定 vendor-lock，而云原生的诞生就是为了解决这个问题，通过云原生基金会 CNCF(Cloud Native Computing Foundation)[4]，我们可以得到一整套解锁云服务商的开源解决方案。

那 K8s 作为云原生 Cloud Native 的重要组成部分之一，它的作用是什么呢？这里我先留一个悬念，通过理解 K8s 的原理，你就能清楚这个问题，并充分利用好 K8s 了。

我们先来看看 K8s 的原理图：



通过图示我们可以知道，PC 本地安装 kubectl 是 K8s 的命令行操作工具，通过它，我们就可以控制 K8s 集群了。又因为 kubectl 是通过加密通信的，所以我们可以在一台电脑上同时控制多个 K8s 集群，不过需要指定当前操作的上下文 context。这个也就是云原生的重要理念，我们的架构可以部署在多套云服务环境上。

在 K8s 集群中，Master 节点很重要，它是我们整个集群的中枢。没错，Master 节点就是 Stateful 的。Master 节点由 API Server、etcd、kube-controller-manager 等几个核心成员组成，它只负责维持整个 K8s 集群的状态，为了保证职责单一，Master 节点不会运行我们的容器实例。

Worker 节点，也就是 K8s Node 节点，才是我们部署的容器真正运行的地方，但在 K8s 中，运行容器的最小单位是 Pod。一个 Pod 具备一个集群 IP 且端口共享，Pod 里可以运

行一个或多个容器，但最佳的做法还是一个 Pod 只运行一个容器。这是因为一个 Pod 里面运行多个容器，容器会竞争 Pod 的资源，也会影响 Pod 的启动速度；而一个 Pod 里只运行一个容器，可以方便我们快速定位问题，监控指标也比较明确。

在 K8s 集群中，它会构建自己的私有网络，每个容器都有自己的集群 IP，容器在集群内部可以互相访问，集群外却无法直接访问。因此我们如果要从外部访问 K8s 集群提供的服务，则需要通过 K8s service 将服务暴露出来才行。

案例：“待办任务”K8s 版本

现在原理我是讲出来了，但可能还是有点不好理解，接下来我们就还是套进案例去看，依然是我们的“待办任务”Web 服务，我们现在把它部署到 K8s 集群中运行一下，你可以切身体验。相信这样，你就非常清楚这其中的原理了。不过我们本地搭建的例子中，为了节省资源只有一个 Master 节点，所有的内容都部署在这个 Master 节点中。

还记得我们上节课构建的 Docker 镜像吗？我们就用它来部署本地的 K8s 集群。

我们通常在实际项目中会使用 YAML 文件来控制我们的应用部署。YAML 你可以理解为，就是将我们在 K8s 部署的所有要做的事情，都写成一个文件，这样就避免了我们要记录大量的 kubectl 命令执行。不过，K8s 也细心地帮我们准备了 K8s 对象和 YAML 文件互相转换的能力。这种能力可以让我们快速地将一个 K8s 集群中部署的结构导出 YAML 文件，然后再在另一个 K8s 集群中用这个 YAML 文件还原出同样的部署结构。

我们需要先确认一下，我们当前的操作是在正确的 K8s 集群上下文中。对应我们的例子里，也就是看当前选中的集群是不是 docker-desktop。

```
1 kubectl config get-contexts
```

 复制代码

```
→ todolist-backend git:(lesson08) kubectl config get-contexts
CURRENT  NAME                CLUSTER             AUTHINFO             NAMESPACE
*        docker-desktop      docker-desktop      docker-desktop
         docker-for-desktop  docker-desktop      docker-desktop
         minikube          minikube            minikube
```

如果不对，则需要执行切换集群：

[📄 复制代码](#)

```
1 kubectl config use-context docker-desktop
```

然后需要我们添加一下拉取镜像的证书服务：

[📄 复制代码](#)

```
1 kubectl create secret docker-registry regcred --docker-server=registry.cn-shan
```

这里我需要解释一下，通常我们在镜像仓库中可以设置这个仓库：公开或者私有。如果是操作系统的镜像，设置为公开是完全没有问题的，所有人都可以下载我们的公开镜像；但如果是我们自己的应用镜像，还是需要设置成私有，下载私有镜像需要验证用户身份，也就是 Docker Login 的操作。因为我们应用镜像仓库中，包含我们的最终运行代码，往往会有我们数据库的登录用户名和密码，或者我们云服务的 ak/sk，这些重要信息如果泄露，很容易让我们的应用受到攻击。

当你添加完 secret 后，就可以通过下面的命令来查看 secret 服务了：

[📄 复制代码](#)

```
1 kubectl get secret regcred
```

另外我还需要再啰嗦一下，secret 也不建议你用 YAML 文件设置，毕竟放着用户名和密码的文件还是越少越好。

做完准备工作，对我们这次部署的项目而言就很简单了，只需要再执行一句话：

[📄 复制代码](#)

```
1 kubectl apply -f myapp.yaml
```

这句话的意思就是，告诉 K8s 集群，请按照我的部署文件 myapp.yaml，部署我的应用。具体如下图所示：

```
→ todolist-backend git:(lesson08) kubectl get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/myapp-759f79bd58-mjwwh	1/1	Running	2	21h

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	29h
service/myapp	NodePort	10.109.56.77	<none>	3001:30512/TCP	21h

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/myapp	1/1	1	1	21h

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/myapp-759f79bd58	1	1	1	21h

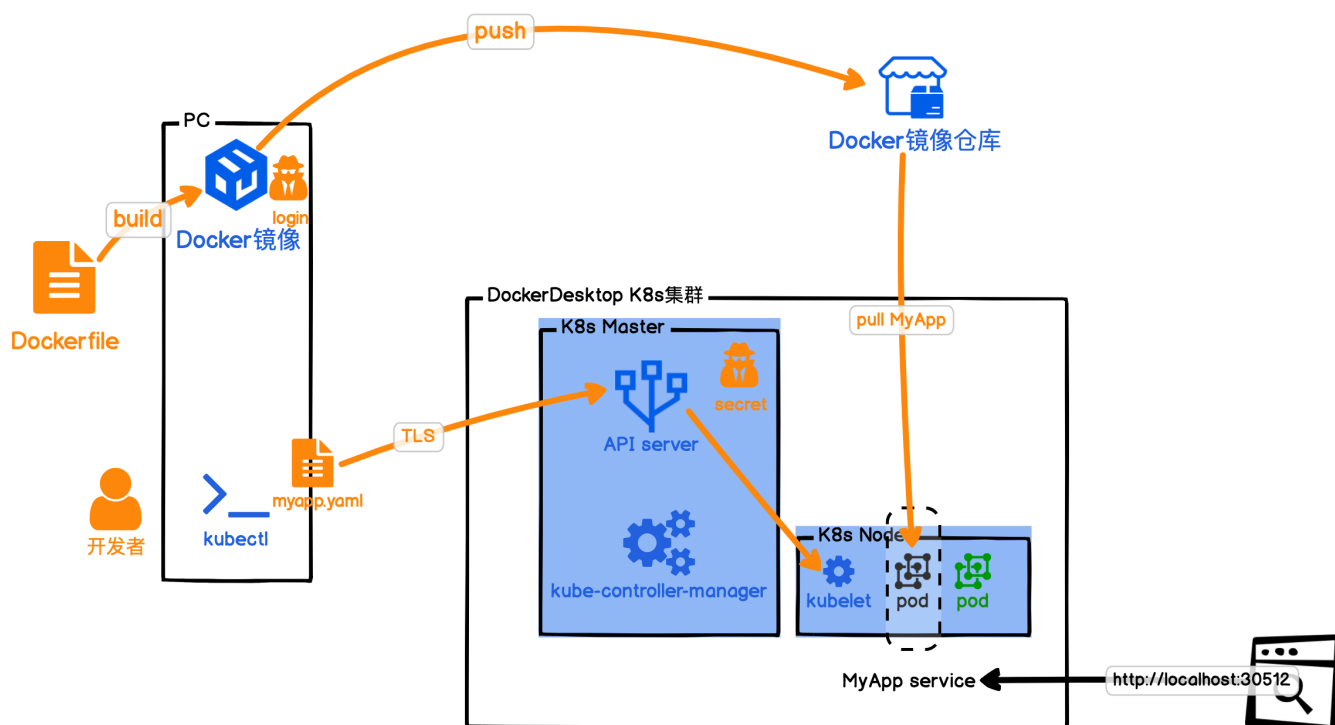
通过获取容器的运行状态，对照上图我粗略地讲解一下我们的 myapp.yaml 文件吧。

首先我们指定要创建一个 service/myapp，它选中打了"app:myapp"标签的 Pod，集群内访问端口号 3001，并且暴露 service 的端口号 30512。

然后我们创建了部署服务 deployment.apps/myapp，它负责保持我们的 Pod 数量恒久为 1，并且给 Pod 打上"app:myapp"的标签，也就是负责我们的 Pod 持久化，一旦 Pod 挂了，部署服务就会重新拉起一个。

最后我们的容器服务，声明了自己的 Docker 镜像是什么，拉取镜像的 secret，以及需要什么资源。

现在我们再回看 K8s 的原理图，不过这张是实现“待办任务”Web 服务版本的：



首先我们可以看出，使用 K8s 仍然需要我们上节课的 Docker 发布流程：build、ship、run。不过现在有很多云服务商也会提供 Docker 镜像构建服务，你只需要上传你的 Dockerfile，就会帮你构建镜像并且 push 到镜像仓库。云服务商提供的镜像构建服务的速度，比你本地构建要快很多倍。

而且相信你也发现了，K8s 其实就是一套 Docker 容器实例的运行保障机制。我们自己 Run 一个 Docker 镜像，会有许多因素要考虑，例如安全性、网络隔离、日志、性能监控等等。这些 K8s 都帮我们考虑到了，它提供了一个 Docker 运行的最佳环境架构，而且还是开源的。

还有，既然我们本地都可以运行 K8s 的 YAML 文件，那么我们在云上是不是也能运行？你还记得前面我们留的悬念吧，现在就解决了。

通过 K8s，我们要解开云服务商锁定 vendor-lock 就很简单了。我们只需要将云服务商提供的 K8s 环境添加到我们 kubectl 的配置文件中，就可以让我们的应用运行在云服务商的 K8s 环境中了。目前所有的较大的云服务商都已经加入 CNCF，所以当你掌握 K8s 后，就可以根据云服务商的价格和自己喜好，自由地选择将你的 K8s 集群部署在 CNCF 成员的云服务商上，甚至你也可以在自己的机房搭建 K8s 环境，部署你的应用。

到这儿，我们就部署好了一个 K8s 集群，那之后我们该如何实时监控容器的运行状态呢？K8s 既然能管理容器集群，控制容器运行实例的个数，那应该也能实时监测容器，帮我们解决扩缩容的问题吧？是的，其实上节课我们已经介绍到了容器扩缩容的原理，但并没有给你讲如何实现，那接下来我们就重点看看 K8s 如何实现实时监控和自动扩缩容。

K8s 如何实现扩缩容？

首先，我们要知道的一点就是，K8s 其实还向我们隐藏了一部分内容，就是它自身的服务状态。而我们不指定命名空间，默认的命名空间其实是 default 空间。要查看 K8s 集群系统的运行状态，我们可以通过指定 namespace 为 kube-system 来查看。K8s 集群通过 namespace 隔离，一定程度上，隐藏了系统配置，这可以避免我们误操作。另外它也提供给我们一种逻辑隔离手段，将不同用途的服务和节点划分开来。

```

→ todolist-backend git:\(lesson08\) kubectl get all -n kube-system
NAME                                READY    STATUS    RESTARTS   AGE
pod/coredns-5c98db65d4-4rn67       1/1     Running   1           29h
pod/coredns-5c98db65d4-xd2fz       1/1     Running   1           29h
pod/etcd-docker-desktop             1/1     Running   0           29h
pod/kube-apiserver-docker-desktop   1/1     Running   0           29h
pod/kube-controller-manager-docker-desktop 1/1     Running   0           29h
pod/kube-proxy-d9kts                1/1     Running   0           29h
pod/kube-scheduler-docker-desktop   1/1     Running   0           29h
pod/storage-provisioner             1/1     Running   0           29h

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
service/kube-dns                    ClusterIP     10.96.0.10    <none>         53/UDP,53/TCP,9153/TCP 29h

NAME                                DESIRED    CURRENT    READY    UP-TO-DATE    AVAILABLE    NODE SELECTOR    AGE
daemonset.apps/kube-proxy           1          1          1        1              1            beta.kubernetes.io/os=linux 29h

NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/coredns             2/2      2              2            29h


NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/coredns-5c98db65d4  2          2          2        29h

```

没错，K8s 自己的服务也是运行在自己的集群中的，不过是通过命名空间，将自己做了隔离。这里需要你注意的是，这些服务我不建议你尝试去修改，因为它们涉及到了 K8s 集群的稳定性；但同时，K8s 集群本身也具备扩展性：我们可以通过给 K8s 安装组件 Component，扩展 K8s 的能力。接下来我先向你介绍 K8s 中的性能指标 metrics 组件 [5]。

我的代码根目录下已经准备好了 metric 组件的 YAML 文件，你只需要执行安装就可以了：

```
1 kubectl apply -f metrics-components.yaml
```

 复制代码

安装完后，我们再看 K8s 的系统命名空间：

```

→ todolist-backend git:(lesson08) kubectl get all -n kube-system
NAME                                READY    STATUS    RESTARTS   AGE
pod/coredns-5c98db65d4-4rn67       1/1     Running   1           30h
pod/coredns-5c98db65d4-xd2fz       1/1     Running   1           30h
pod/etcd-docker-desktop             1/1     Running   0           29h
pod/kube-apiserver-docker-desktop   1/1     Running   0           29h
pod/kube-controller-manager-docker-desktop 1/1     Running   0           29h
pod/kube-proxy-d9kts                1/1     Running   0           30h
pod/kube-scheduler-docker-desktop   1/1     Running   0           29h
pod/metrics-server-74657b4dc4-t979q 1/1     Running   0           100s
pod/storage-provisioner              1/1     Running   0           29h

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)                                AGE
service/kube-dns                    ClusterIP     10.96.0.10    <none>          53/UDP,53/TCP,9153/TCP                30h
service/metrics-server               ClusterIP     10.97.152.64  <none>          443/TCP                                100s

NAME                                DESIRED    CURRENT    READY    UP-TO-DATE    AVAILABLE    NODE SELECTOR    AGE
daemonset.apps/kube-proxy            1          1          1        1              1            beta.kubernetes.io/os=linux    30h

NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/coredns              2/2      2              2            30h
deployment.apps/metrics-server        1/1      1              1            100s

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/coredns-5c98db65d4   2          2          2        30h
replicaset.apps/metrics-server-74657b4dc4 1          1          1        100s

```

对比你就能发现，我们已经安装并启动了 metrics-server。那么 metrics 组件有什么用呢？我们执行下面的命令看看：

```
1 kubectl top node
```

 复制代码


```

→ todolist-backend git:(lesson08) kubectl top node
NAME                CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
docker-desktop      316m         7%     1249Mi           66%

```

安装 metrics 组件后，它就可以将我们应用的监控指标 metrics 显示出来了。没错，这里我们又可以用到上一讲的内容了。既然我们有了实时的监控指标，那么我们就可以依赖这个指标，来做我们的自动扩缩容了：

```
1 kubectl autoscale deployment myapp --cpu-percent=30 --min=1 --max=3
```

 复制代码

上面这句话的意思就是，添加一个自动扩缩容部署服务，cpu 水位是 30%，最小维持 1 个 Pod，最大维持 3 个 Pod。执行完后，我们就发现会多了一个部署服务。

```
1 kubectl get hpa
```

[复制代码](#)

```
→ todolist-backend git:(lesson08) kubectl get hpa
```

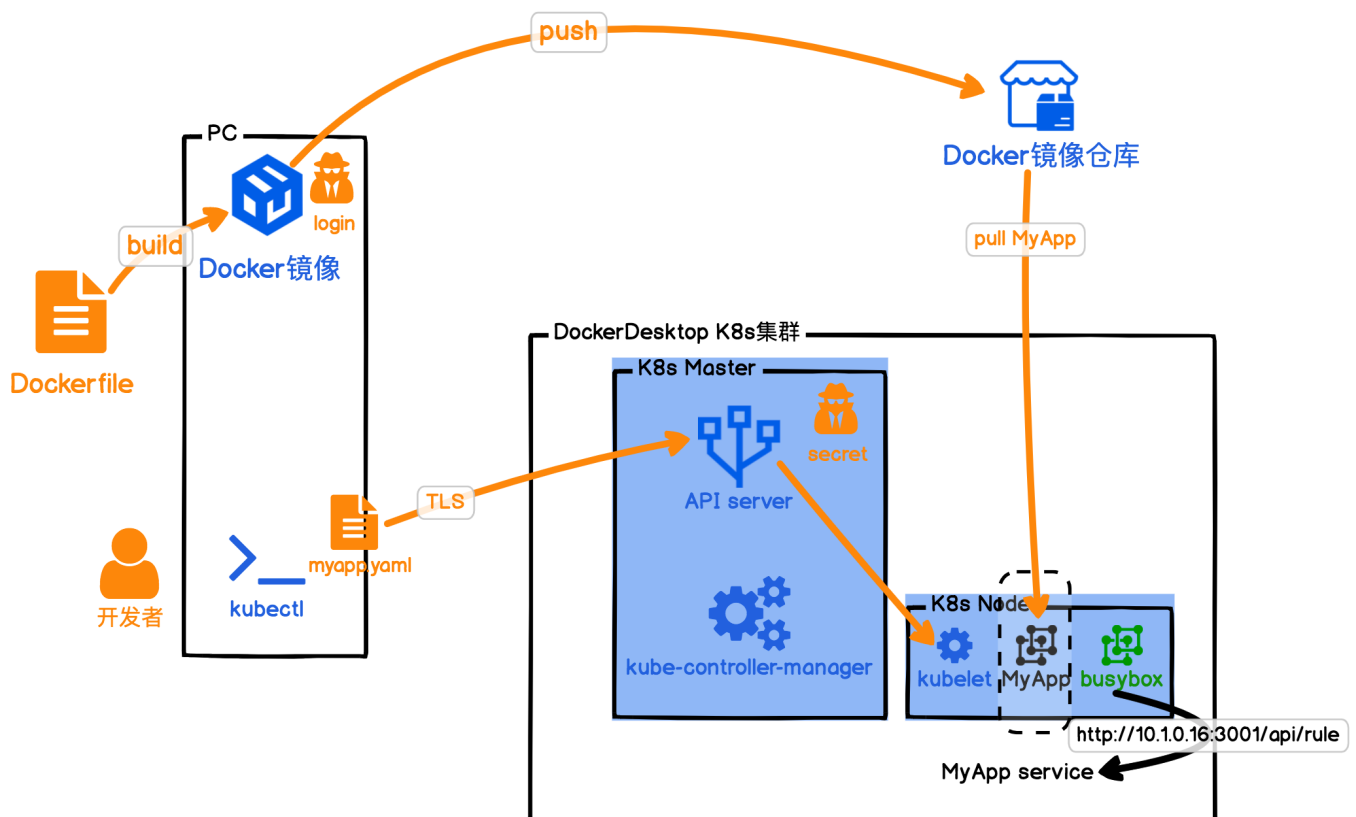
NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
myapp	Deployment/myapp	0%/30%	1	3	1	22h

接下来，我们就可以模拟压测了：

```
1 kubectl run -i --tty load-generator --image=busybox /bin/sh
2 $ while true; do wget -q -O- http://10.1.0.16:3001/api/rule; done
```

[复制代码](#)

这里我们用一个 K8s 的 Pod，启动 busybox 镜像，执行死循环，压测我们的 MyApp 服务。不过我们目前用 Node.js 实现的应用可以扛住的流量比较大，单机模拟的压测，轻易还压测不到扩容的水位。



总结

这节课我向你介绍了云原生基金会 CNCF 的重要成员：Kubernetes。**K8s 是用于自动部署、扩展和管理容器化应用程序的开源系统。**云原生其实就是一套通用的云服务商解决方案。

然后我们一起体验了在本地 PC 上，通过 Docker desktop 搭建 K8s。搭建完后，我还向你介绍了 K8s 的运行原理：K8s Master 节点和 Worker 节点。其中，Master 节点，负责我们整个 K8s 集群的运作状态；Worker 节点则是具体运行我们容器的地方。

之后，我们就开始把“待办任务”Web 服务，通过一个 K8s 的 YAML 文件来部署，并且暴露 NodePort，让我们用浏览器访问。

为了展示 K8s 如何通过组件 Component 扩展能力，接着我们介绍了 K8s 中如何安装使用组件 metrics：我们通过一个 YAML 文件将 metrics 组件安装到了 K8s 集群的 kube-system 命名空间中后，就可以监控到应用的运行指标 metrics 了。给 K8s 集群添加上监控指标 metrics 的能力，我们就可以通过 autoscale 命令，让应用根据 metrics 指标和水位来扩缩容了。

最后我们启动了一个 BusyBox 的 Docker 容器，模拟压测了我们的“待办任务”Web 服务。

总的来说，这节课我们的最终目的就是在本地部署一套 K8s 集群，通过我们“待办任务”Web 服务的 K8s 版本，让你了解 K8s 的工作原理。我们已经把下节课的准备工作做好了，下节课我们将在 K8s 的基础上部署 Serverless，可以说，实现属于你自己的 Serverless，你已经完成了一半。

作业


这节课是实战课，所以作业就是我们今天要练习的内容。请在你自己的电脑上安装 K8s 集群，部署我们的“待办任务”Web 服务到自己的 K8s 集群，并从浏览器中访问到 K8s 集群提供的服务。

另外，你可以尝试一下，手动删除我们部署的 MyApp Pod。

 复制代码

```
1 kubectl delete pod/你的pod名字
```


但你很快就会发现，这个 Pod 会被 K8s 重新拉起，而我们要清除部署的 MyApp 的所有内容其实也很简单，只需要告诉 K8s 删除我们的 myapp.yaml 文件创建的资源就可以了。

 复制代码

```
1 kubectl delete -f myapp.yaml
```

快来动手尝试一下吧，期待你也能分享下今天的成果以及感受。另外，如果今天的内容让你有所收获，也欢迎你把它分享给身边的朋友，邀请他加入学习。

参考资料

[1] [🔗 https://gitforwindows.org/](https://gitforwindows.org/)

[2] [🔗 https://www.cncf.io/](https://www.cncf.io/)

[3] [🔗 https://kubernetes.io/zh/](https://kubernetes.io/zh/)

[4] [🔗 https://github.com/cncf/landscape](https://github.com/cncf/landscape)

[5] [🔗 https://github.com/kubernetes-incubator/metrics-server/](https://github.com/kubernetes-incubator/metrics-server/)

五一计划 📅

晒学习姿势 「免费」领课程



【点击】图片, 立即参加 >>>

© 版权归极客邦科技所有, 未经许可不得传播售卖。页面已增加防盗追踪, 如有侵权极客邦将依法追究其法律责任。

上一篇 07 | 后端BaaS化 (下) : Container Serverless

精选留言 (2)

写留言



一步

2020-05-04

在 K8s 中 容器不是部署在 Node 节点上吗? 怎么文中部署在了Master 节点了? Master 节点也可以充当 Node节点吗?

展开 ▾

2



我来也

2020-05-04

由于我有现成的k8s环境,所以就不用重新搭建,可以直接使用了.

最近两天在折腾阿里云的弹性容器实例(ECI).

想不到这东西还能使用抢占式实例 (Spot) 的模式,2CPU-2G的实例,不到0.05元每小时的价格....

展开 ✓

