**13-后端工具**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **所有者** | **修改时间** | **创建时间** |
| [14-git工具](https://cqbk0n8o1h.feishu.cn/wiki/SNgfwL5gHi9Oe9ke1yHcpotYnfh) | 李2017 | 2024-05-24 09:34 | 2024-05-14 09:25 |
| [15-docker](https://cqbk0n8o1h.feishu.cn/wiki/Pla1wxcRuiE9tZktkB0cIY2wnxe) | 李2017 | 2024-05-25 18:07 | 2024-05-14 09:25 |
| [16-数据库](https://cqbk0n8o1h.feishu.cn/wiki/Tulsw3Gq7iHuyWkAgxjc9htUnnb) | 李2017 | 2024-05-14 09:38 | 2024-05-14 09:25 |

1. **Kubernetes使用**

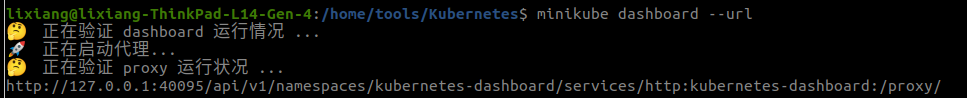
1. [📚 Kubernetes(K8S)简介 - K8S 教程 - 易文档](https://k8s.easydoc.net/docs/dRiQjyTY/28366845/6GiNOzyZ/9EX8Cp45)
2. 一篇文章教会你使用kubernetes的基本使用\_kubernetes使用-CSDN博客
3. [Install and Set Up kubectl on Linux](https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/)
4. [Minikube 安装和简单使用 - 江湖小小白 - 博客园](https://www.cnblogs.com/jhxxb/p/15220729.html)

1.1 **minikube 模拟k8s模拟器工具**

1.1.1 **启动**

|  |
| --- |
| Bash minikube start  添加节点 minikube node list minikube node add/remove  可视化 minikube dashboard --url 输出一个网址可以可视化部署k8s # 让其它 IP 可以访问 kubectl proxy --port=8888 --address='0.0.0.0' --accept-hosts='^.\*' |

1.1.2 **可视化**



|  |
| --- |
| Bash 浏览器访问 http://127.0.0.1:40095/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ |

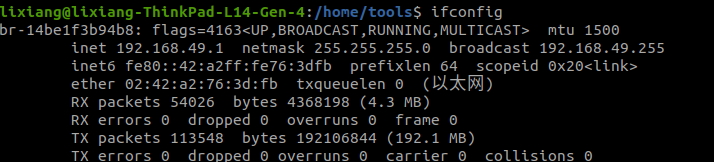
说明：

这个 URL 指向了一个运行在本地计算机上的代理服务器，这个代理服务器被配置为将请求转发到 Minikube 虚拟机中 Kubernetes Dashboard 的实际服务地址。

详细说明如下：

* http://127.0.0.1:40095：代理服务运行在本地主机的 127.0.0.1 地址（也就是 localhost），监听的端口是 40095。这个端口是动态分配的，每次运行 minikube dashboard --url 时都可能不同。
* /api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/：这是 Kubernetes API 的一个路径，用于代理访问名称为 http:kubernetes-dashboard: 的服务，该服务在 kubernetes-dashboard 名命名空间下。这个路径实际上利用 Kubernetes API server 的反向代理功能来访问 Kubernetes Dashboard。

当您访问这个 URL 时，您的 Web 浏览器发送的请求会通过本地监听的代理端口，被转发到 Minikube 虚拟机中 Kubernetes Dashboard 运行的 Pod。这样您就可以查看和管理您的 Kubernetes 集群状态，包括 Pods、Deployments、Services 等 Kubernetes 资源。



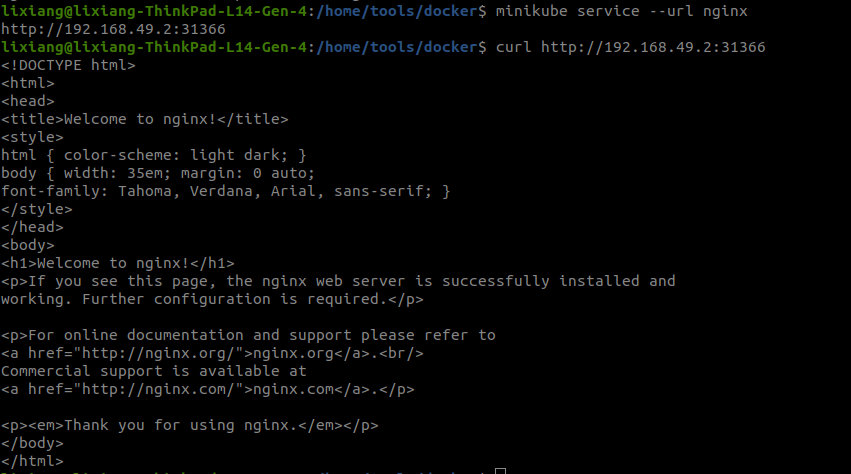


|  |
| --- |
| Bash 浏览器访问 http://127.0.0.1:8888/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/overview?namespace=\_all 或 http://192.168.49.1:8888/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/overview?namespace=\_all |

，并且允许来自任何 host 的连接。这意味着 kubectl proxy 将代理到 Kubernetes API 和任何通过 API 暴露的服务，如 Kubernetes Dashboard。

1.1.3 **部署服务和访问**

|  |
| --- |
| Bash kubectl create deployment nginx --image=nginx kubectl expose deployment nginx --port=80 --type=NodePort # 获取访问地址 minikube service --url nginx |



|  |
| --- |
| Bash curl http://192.168.49.2:31366 也可以 <https://kubernetes.io/zh/docs/tasks/access-application-cluster/access-cluster/#manually-constructing-apiserver-proxy-urls> http://192.168.49.1:8888/api/v1/namespaces/default/services/nginx:80/proxy/ |

1.1.4 **负载均衡访问**

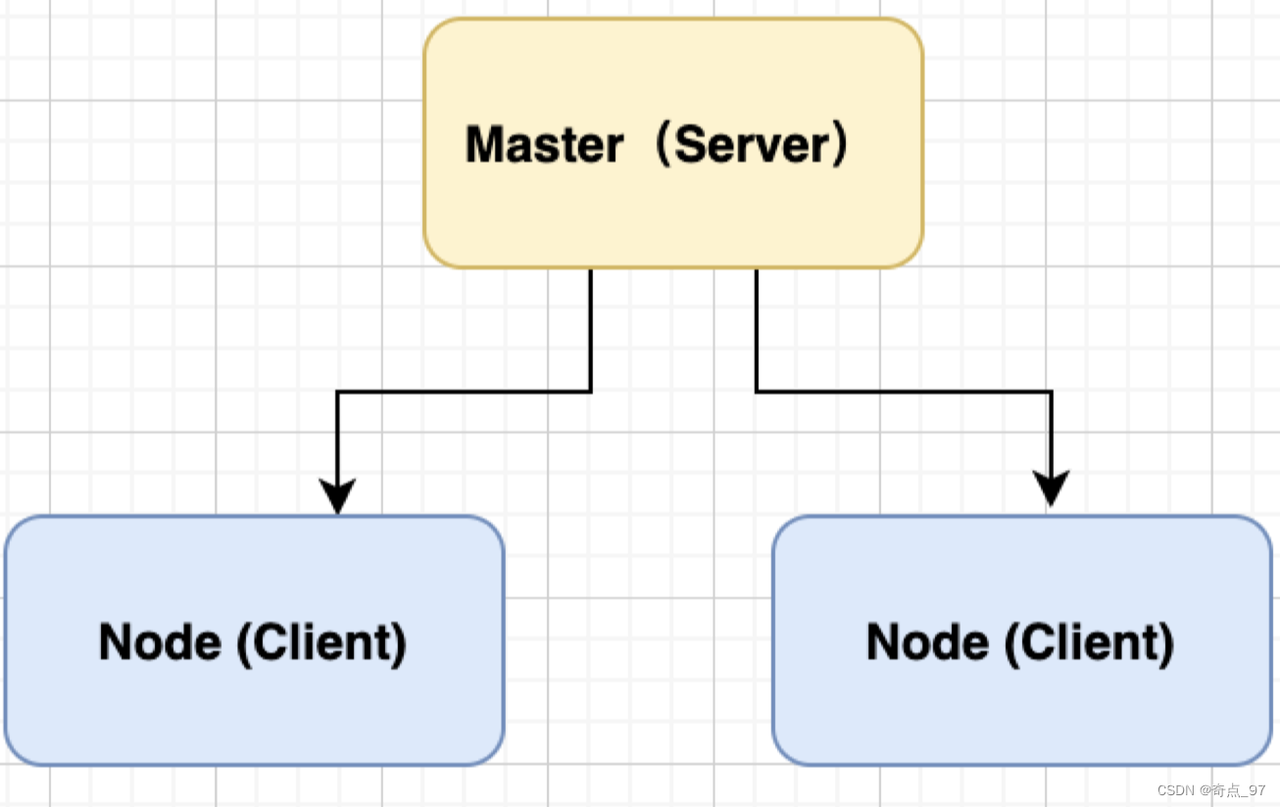
|  |
| --- |
| Bash # 新开窗口运行 minikube tunnel --cleanup=true  # 重新部署 kubectl delete deployment nginx kubectl delete service nginx kubectl create deployment nginx --image=nginx kubectl expose deployment nginx --port=80 --type=LoadBalancer # 查看外部地址 kubectl get svc |

1.1.5 **卸载**

|  |
| --- |
| Bash minikube stop minikube delete --all docker rmi kicbase/stable:v0.0.25rm -rf ~/.kube ~/.minikube sudo rm -rf /usr/local/bin/kubectl /usr/local/bin/minikube docker system prune -a #删除未使用镜像 |

1.2 **k8s名词概念**

常见概念：master、node、pod、podcontroller、service、label、namespace



**Master**

指的是集群控制节点（相当于整个集群的指挥中⼼），在 每个Kubernetes集群⾥都需要有⼀个Master来负责整个集 群的管理和控制

**Node**

除了master，k8s集群中的其他机器被称为Node节点， Node节点才是kubernetes集群中的⼯作负载节点 每个Node节点都会被master分配⼀些⼯作负载（docker 容器），node节点上的docker负责容器的运⾏

除了master，k8s集群中的其他机器被称为Node节点， Node节点才是kubernetes集群中的⼯作负载节点 每个Node节点都会被master分配⼀些⼯作负载（docker 容器），node节点上的docker负责容器的运⾏

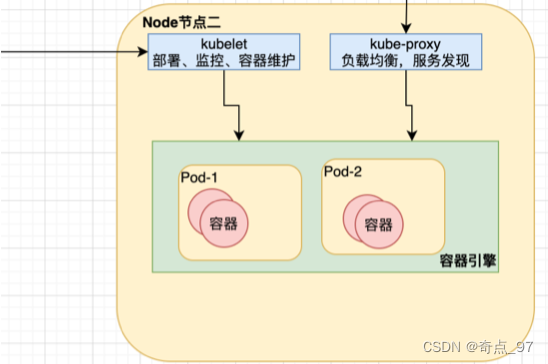
**Pod**

Pod是⼀组容器, 在K8S中，最⼩的单位是Pod, ⼀个Pod可 以包含多个容器，但通常情况下我们在每个Pod中仅使⽤ ⼀个容器。可以把Pod理解成豌⾖荚, Pod内的每个容器是⼀颗颗豌⾖

**创建分类**

⾃主创建：直接创建出来的Pod，这种pod删除后就没 有了，也不会⾃动重建

控制器创建：通过控制器创建的pod，这类Pod删除了 之后还会⾃动重建



2. **ngnix**

1. [Docker 安装 Nginx | 菜鸟教程](https://www.runoob.com/docker/docker-install-nginx.html)
2. 在docker容器中使用nginx，相关配置\_nginx 判断容器域名可用-CSDN博客

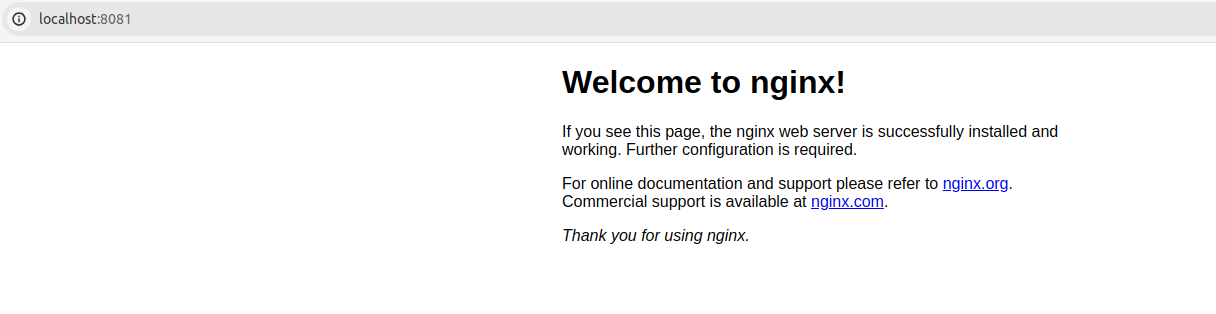
2.1 **下载镜像**

|  |
| --- |
| Bash $ docker pull nginx:latest |

2.2 **启动容器**

|  |
| --- |
| Bash docker run --name my\_nginx -p 8081:80 -d nginx |

2.3 **浏览器访问**



2.4 **ngnix使用**