实验二：Service job in cloud

目录

实验概述

在实验二中，我们学习与 Kubernetes 的相关知识。Kubernetes 常被简写为 K8s，是谷歌基于 Borg 推出的开源容器编排管理系统。K8s 经常用于管理面向用户型应用，具有很高的灵活性、可扩展性和容灾性。在实验二中，我们要在华为云 ECS 上搭建一个简单的单节点 Kubernetes 集群，并在这个集群上运行一个简单的 wordpress 博客系统，并在搭建的过程中学习 K8s 的相关概念，了解云原生时代的应用组织方式。

Minikube 是 Kubernetes 的单节点版，用于学习和本地开发。使用 minikube 搭配 docker，我们可以在一台服务器上搭建一个由多个本地容器组成的微型 Kubernetes 集群。

Wordpress 是一个用 PHP 语言开发的博客平台。使用 wordpress 搭配 mysql 数据库，我们可以搭建起一个简易的博客网站。

通过本实验，同学们可以了解到 Kubernetes 的基本组件和基本概念，并动手实操，部署一

个典型的面向用户型服务应用。

相关背景资料

建议同学们在开始实验前，先大致浏览以下材料，做到对本实验涉及的基本概念有所了解。

Container：

官方文档（docker）：https://www.docker.com/resources/what-container/

视频解释：https://www.youtube.com/watch?v=_dfLOzuIg2o

Kubernetes:

官方文档：https://kubernetes.io/zh-cn/docs/concepts/overview/

视频解释：https://dev.to/techworld_with_nana/kubernetes-simply-explained-for-beginners-33em

Minikube:

官方文档：https://minikube.sigs.k8s.io/docs/

视频解释：https://dev.to/techworld_with_nana/what-is-minikube-and-kubectl-setup-a-minikube-cluster-for-kubernetes-beginners-5gj3

Wordpress:

官方文档：https://wordpress.com/zh-cn/

视频解释：https://www.youtube.com/watch?v=71EZb94AS1k

安装与准备

1. 购买华为云 ECS

在本实验中，为了确保同学们实验环境的统一性，我们仍然通过华为云 ECS 完成实验。但所有的实验内容也可以稍加调整后在同学们的电脑上本地运行。

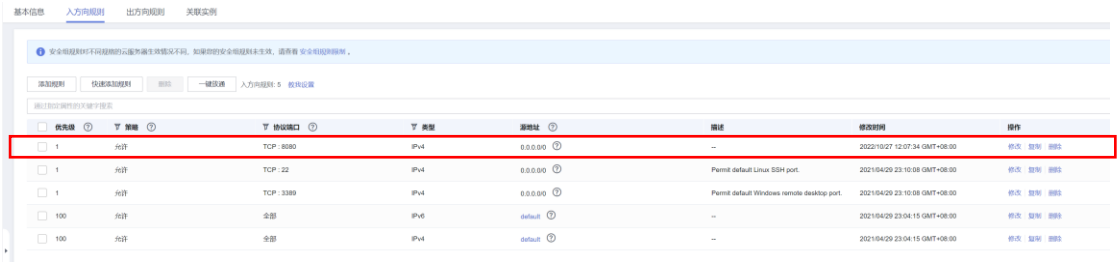
#购买 ECS 的部分与实验一完全相同

额外地，我们需要设置安全组入方向规则，允许远程访问 ECS 的 8080 端口。通过这个端口，我们将能够访问 ECS 上运行的博客系统。

首先，在华为云控制台中“弹性云服务器”选项卡中，点击已经购买的 ECS 的名称：

在跳转到的页面右侧，点击安全组

在弹出的页面中选择“入方向规则”，并点击“快速添加规则”，勾选 HTTP_ALT(8080)，点击确定，可以发现添加了一条新规则：



2. 安装 minikube

我们使用 1.23.1 版本的 minikube。

远程连接到华为云 ECS 服务器后，在终端中输入以下命令安装 minikube：

```
curl -Lo minikube https://kubernetes.oss-cn-hangzhou.aliyuncs.com/minikube/releases/v1.23.1/minikube-linux-amd64 && chmod +x minikube && sudo mv minikube /usr/local/bin/
```

检查安装是否正常：

```
minikube version
```

此时应显示如下内容

```
root@test-ecs:~/experiment2# minikube version
minikube version: v1.23.1
commit: 9e2f8cb489d9b3e871ba206d40ae92c7521b7e76-dirty
```

3. 安装 docker

为了实现更好的隔离与封装，Minikube 组件（包括 minikube 控制平面和用户启动的 pod）本身并不运行在主机中，而是运行在一个从主机上启动的虚拟机或 docker 容器中。具体的虚拟机 hypervisor 或者容器类型被 minikube 称为 driver。

华为云 ECS 本身就是一台虚拟机，因此我们无法在虚拟机内启动虚拟机。我们采用 docker 作为 minikube driver。

按照链接中给出的方法安装 docker：
https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository

Install using the repository

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

Set up the repository

1. Update the `apt` package index and install packages to allow `apt` to use a repository over HTTPS:

```
$ sudo apt-get update

$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

2. Add Docker's official GPG key:

```
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

3. Use the following command to set up the repository:

```
$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Install Docker Engine

1. Update the `apt` package index:

```
$ sudo apt-get update
```

Receiving a GPG error when running `apt-get update` ?

Your default umask may be incorrectly configured, preventing detection of the repository public key file. Try granting read permission for the Docker public key file before updating the package index:

```
$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
$ sudo apt-get update
```

2. Install Docker Engine, containerd, and Docker Compose.

Latest | Specific version

To install the latest version, run:

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

3. Verify that the Docker Engine installation is successful by running the `hello-world` image:

```
$ sudo docker run hello-world
```

This command downloads a test image and runs it in a container. When the container runs, it prints a confirmation message and exits.

You have now successfully installed and started Docker Engine. The `docker` user group exists but contains no users, which is why you're required to use `sudo` to run Docker commands. Continue to Linux post-install to allow non-privileged users to run Docker commands and for other optional configuration steps.

4. 安装 kubectl

Kubectl 是 Kubernetes 集群的命令行管理工具。本实验中，我们要借助 kubectl 检查集群状态，设置端口代理等。与 minikube 1.23.1 版本对应的 kubectl 版本是 1.22.1。

更新 apt 源，并设置 https 支持

```
apt update && apt install -y apt-transport-https
```

添加阿里云 kubernetes 源的访问公钥

```
curl https://mirrors.aliyun.com/kubernetes/apt/doc/apt-key.gpg | apt-key add -
```

添加阿里云 kubernetes 源

```
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb https://mirrors.aliyun.com/kubernetes/apt/ kubernetes-xenial main
EOF
```

安装 kubectl

```
apt install -y kubectl=1.22.1-00
```

检查安装是否正常

```
kubectl version
```

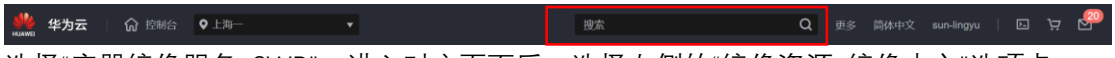
此时应显示如下内容

```
root@test-ecs:~/experiment2# kubectl version
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.1", GitCommit:"632ed300f2c3
4f6d6d15ca4cef3d3c7073412212", GitTreeState:"clean", BuildDate:"2021-08-19T15:45:37Z", GoVersion:
"go1.16.7", Compiler:"gc", Platform:"linux/amd64"}
The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

5. 准备镜像地址

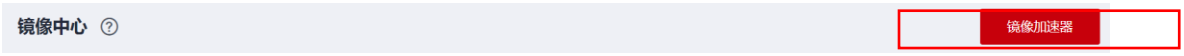
在国内，Minikube 拉取容器镜像的速度比较慢。因此一般需要使用镜像加速器。在本实验中，我们使用华为云提供的容器镜像加速器服务。

首先登录华为云控制台 https://console.huaweicloud.com/，在搜索栏中输入“容器镜像服务”



选择“容器镜像服务 SWR”。进入对应页面后，选择左侧的“镜像资源-镜像中心”选项卡

选择页面右上角的红色按钮“镜像加速器”



记录下弹出页面中的加速器地址，并按照弹出页面中的要求修改“/etc/docker/daemon.json”文件。

6. 预下载 coredns 镜像

由于网址名称问题，在启动 minikube 时可能会发生无法拉取 coredns 镜像的问题。因此我们使用 docker 预下载 coredns 镜像，并将其重命名为 minikube 需要的格式。

使用如下命令预下载 coredns 1.8.4：

```
    docker pull registry.cn-hangzhou.aliyuncs.com/google_containers/coredns:1.8.4
```

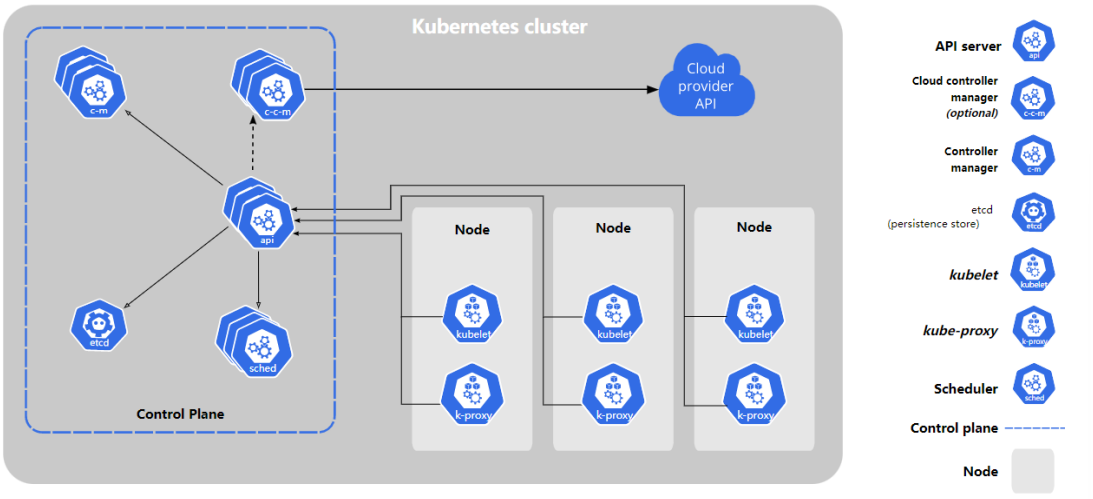
将其重命名为 minikube 需要的格式（coredns v1.8.4）

```
    docker tag registry.cn-hangzhou.aliyuncs.com/google_containers/coredns:1.8.4
registry.cn-hangzhou.aliyuncs.com/google_containers/coredns/coredns:v1.8.4
```

实验二 Part A：认识 Kubernetes 集群

1. Kubernetes 集群的结构

一个 kubernetes 集群的基本结构包括一个 control plane 和若干 node。每个 node 都是一个独立的服务器，上面运行 kubelet 和 k-proxy 两个管理组件。其大致结构如下：



同学们可以浏览 kubernetes 的官方文档，了解这些管理组件的大致用途：
https://kubernetes.io/zh-cn/docs/concepts/overview/components/
这些管理组件全部以 pod 的形式运行。一个 pod 由一个或者多个容器组成：
https://kubernetes.io/docs/concepts/workloads/pods/
当我们要在 kubernetes 集群上运行应用时，应用最终也会以 pod 的形式被部署并加以管理。

2. 启动 minikube 集群

使用如下命令启动 minikube 集群

```
minikube start --image-mirror-country='cn' --image-mirror="加速器地址"  --force
```

加速器地址可以在“安装与准备”部分第 5 节中找到。
如果你看到如下的输出，那么恭喜你，你已经拥有了一个正常工作的 minikube 集群！

```
root@test-ecs:~/experiment2# minikube start --image-mirror-country='cn' --registry-mirror="https://0bd[redacted]0.mirror.swr.myhuaweicloud.com" --force
😄  minikube v1.23.1 on Ubuntu 18.04 (amd64)
❗  minikube skips various validations when --force is supplied; this may lead to unexpected behavior
✨  Automatically selected the docker driver. Other choices: none, ssh
🛑  The "docker" driver should not be used with root privileges.
💡  If you are running minikube within a VM, consider using --driver=none:
📘    https://minikube.sigs.k8s.io/docs/reference/drivers/none/
✅  Using image repository registry.cn-hangzhou.aliyuncs.com/google_containers
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.22.1 on Docker 20.10.8 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔎  Verifying Kubernetes components...
    ▪ Using image registry.cn-hangzhou.aliyuncs.com/google_containers/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

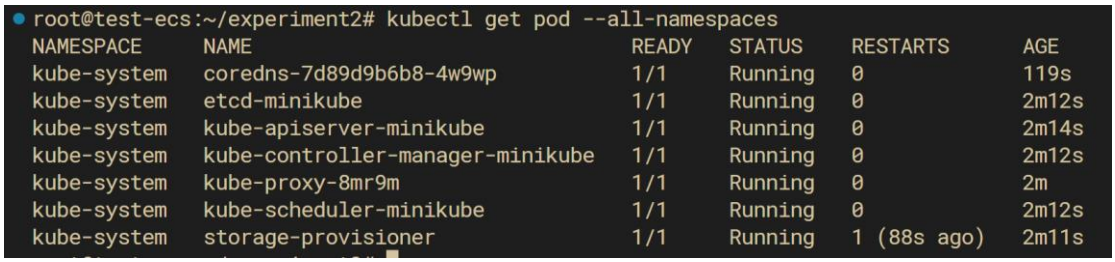
3. 观察 minikube 集群

通过 kubectl，我们可以在命令行检查这些组件的运行情况。

在命令行输入如下命令：

```
kubectl get pod --all-namespaces
```

可以看到 kube-system namespace 下的 pod 以及他们的运行状态：



*有些 pod 的状态可能显示为 Pending，等待一段时间后即可转变为 Running。

上图中，大多数 pod 都运行着控制平面的管理组件，但由于 minikube 是一个单节点的 kubernetes 集群，kube-proxy-xxx 这个 pod 并不属于控制平面。它运行着 Node 上的管理组件。

4. 实验二 Part A 报告

- 请你给出运行如下两个命令时的结果截图

```
minikube start --image-mirror-country='cn' --image-mirror="加速器地址"  --force
kubectl get pod --all-namespaces
```

- 请你在实验报告中描述你对 kubernetes 集群的认识。（可以从用途、架构等角度阐述，鼓励同学们与课上内容相结合）
- 请你任意选择一个 Kubernetes 管理组件，学习这个组件的具体功能，并在报告中描述。

注意：实验二 PartA 报告的总篇幅不得超过一页（五号字，不含截图），若超过一页则按不及格处理。

实验二 Part B：在 Kubernetes 上搭建博客系统

在 PartB 中，我们在 minikube 上搭建一个由 wordpress 和 mysql 两个组件组成的博客系统，并远程访问博客网站，创建一篇博客。

1. 准备 yaml 配置文件

在 K8s 中，我们通过提供 yaml 格式配置文件的方式部署应用。K8s 会按照配置文件的内容拉取镜像，构建应用。我们准备了 wordpress 和 mysql 对应的两个 yaml 文件。同学们可以直接使用。

首先建立一个新文件夹，后续的操作均在此目录下进行

```
mkdir experiment2 && cd experiment2
```

创建 kustomization.yaml，这个文件将包含部署一个博客系统的全部配置内容。

```
cat <<EOF >./kustomization.yaml
secretGenerator:
- name: mysql-pass
  literals:
  - password=你的密码
EOF
```

注意把"你的密码"改成一个你自己喜欢的密码。现在，kustomization.yaml 包含了一个 Secret generator，这是一个用于管理密码的组件，我们不做进一步介绍，感兴趣的同学可以参考这个链接：https://kubernetes.io/docs/concepts/configuration/secret/

下载两个 yaml 文件，我们不对 yaml 文件的具体内容和格式进行进一步介绍，感兴趣的同学可以自行了解。

```
curl -LO https://k8s.io/examples/application/wordpress/mysql-deployment.yaml
curl -LO https://k8s.io/examples/application/wordpress/wordpress-deployment.yaml
```

将两个 yaml 文件的名字加入到 kustomization.yaml 中：

```
cat <<EOF >>./kustomization.yaml
resources:
  - mysql-deployment.yaml
  - wordpress-deployment.yaml
EOF
```

现在，kustomization.yaml 文件的内容应该如下图所示：

```
root@test-ecs:~/experiment2# cat kustomization.yaml
secretGenerator:
- name: mysql-pass
  literals:
  - password=sunlingyu
resources:
  - mysql-deployment.yaml
  - wordpress-deployment.yaml
```

2. 部署博客系统

只需要把 kustomization.yaml 提供给 minikube，就可以将博客系统自动部署。

执行如下命令：

```
kubectl apply -k ./
```

你应该看到如下输出

```
root@test-ecs:~/experiment2# kubectl apply -k ./
secret/mysql-pass-54c225hgm8 created
service/wordpress created
service/wordpress-mysql created
persistentvolumeclaim/mysql-pv-claim created
persistentvolumeclaim/wp-pv-claim created
deployment.apps/wordpress created
deployment.apps/wordpress-mysql created
```

具体来说，在 kubernetes 内部，我们的博客系统通过 wordpress 和 wordpress-mysql 两个 deployment 进行部署，每个 deployment 创建一个对应的 pod，并通过 service 互相通信。

3. 远程访问博客系统

到上一步为止，博客系统已经搭建好了。我们还需要进行一些网络配置，才能从远程访问它。
Kubernetes 容器的网络关系比较复杂，这里我们不做相详细介绍。同学们只需要使用 kubectl 这一工具设置端口代理，把运行 wordpress 容器的 80 端口开放给 ECS 的 8080 端口，即可远程访问。

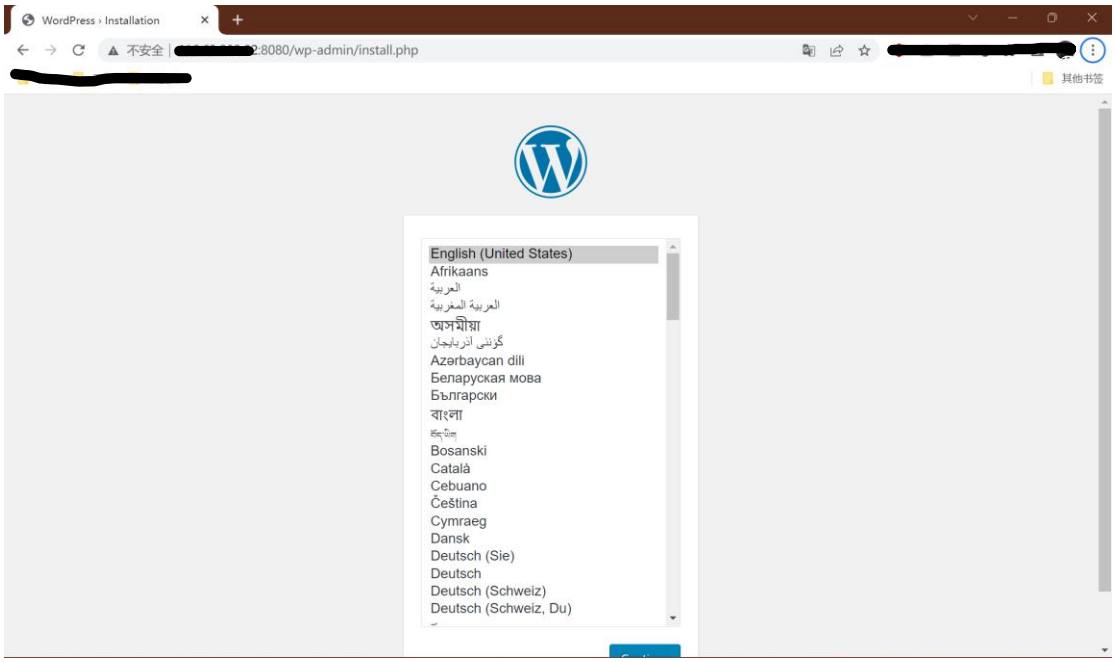
新开启一个 bash，并执行

```
kubectl port-forward service/wordpress --address 0.0.0.0 8080:80
```

接下来，在本机打开浏览器，输入

```
[IP of ECS]:8080
```

你将看到如下页面



接下来，我们就可以跟随 wordpress 的指引，创建第一篇博文了。

*本实验参考：https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/

4. 实验二 Part B 报告

- 在上面的流程中，我们在执行 kubectl apply 后，出现了如下输出：

```
root@test-ecs:~/experiment2# kubectl apply -k ./
secret/mysql-pass-54c225hgm8 created
service/wordpress created
service/wordpress-mysql created
persistentvolumeclaim/mysql-pv-claim created
persistentvolumeclaim/wp-pv-claim created
deployment.apps/wordpress created
deployment.apps/wordpress-mysql created
```

  请你解释其中 service, persistentvolumeclaim 与 deployment 的含义与用途。
- 我们想要了解部署博客系统后集群中运行的所有 service、deployment 和 pod 的状态。请你给出执行相关命令的结果截图，以及创建的第一个博客内容的截图（注意包含网址）。
- 请你谈谈部署 kubernetes 应用与部署传统应用的不同之处

注意：实验二 PartB 报告的总篇幅不得超过 1 页（五号字，不含截图），若超过一页则按不及格处理。