**山东大学 计算机科学与技术 学院**

**云计算技术 课程实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **学号：**201900130133 | **姓名：**施政良 | | **班级：**四班 |
| **实验题目：** Linux环境下Docker环境的搭建 | | | |
| **实验学时：**2 | | **实验日期：**2020-04-11 | |
| **实验目的：**在Linux环境下，熟悉Docker虚拟化环境。  **具体包括：**了解Docker虚拟化环境的配置和部署，完成实验环境及实验工具的熟悉，构建镜像，撰写实验报告。 | | | |
| **硬件环境：**  联网计算机一台 | | | |
| **软件环境：**  Windows or Linux（ubuntu 18.04） | | | |
| **实验步骤与内容：**  **实验步骤概述：**  本次实验主要涉及Docker环境的搭建以及容器虚拟化技术的了解，实验步骤大致分为如下几步：   1. 了解docker的基本概念 2. 实际搭建docker环境 3. 测试环境搭建是否正确 4. 扩展部分：在docker内配置ubuntu和python开发环境   **具体实验内容**   1. **Docker介绍**   Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的镜像中，然后发布到任何流行的 Linux或Windows操作系统的机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口.  一个完整的Docker有以下几个部分组成：   * DockerClient客户端 * Docker Daemon守护进程 * Docker Image镜像 * DockerContainer容器   Docker 使用客户端-服务器 (C/S) 架构模式，使用远程API来管理和创建Docker容器。Docker 容器通过 Docker 镜像来创建。容器与镜像的关系类似于面向对象编程中的对象与类   1. **容器的概念**   包含相应应用程序组件的服务实例即为容器(Container)。在一个容器中运行的程序无法看到容器外的程序进程，包括那些直接运行在宿主机(host)上的应用和其它容器中的应用。对一个应用程序而言，容器往往容纳了该程序运行所需要的全部文件，它可能包含自己的库、自己的/boot目录、/usr目录、/home目录等。然而，如果需要的话，运行中的容器甚至可能仅包含一个文件，比如运行一个不依赖任何文件的二进制程序。   1. **Docker中的重要概念**   Docker是一种容器技术，其中有镜像、数据卷、哈希函数等相关概念，具体介绍如下所示：   * Docker是一个工具，能帮助我们方便的创建、运行、部署软件。 * Docker允许我们将一个软件和其依赖(运行环境)打包成一个单独的库，更利于移植可运行的软件。   在容器虚拟化技术汇总常常会出现如下术语，其概念如下：  （1）镜像   * Docker镜像是一个构建容器的只读模板，提供了容器应用打包的标准格式，容器即为一个通过Docker镜像创建的运行时实例。 * Docker 镜像包含Dockerfile、依赖和程序的代码 * Dockerfile中包含一系列的指令用来创建Docker Image     （2）容器——镜像运行时的实体   * 容器是镜像运行时的实体。容器可以被创建、启动、停止、删除、暂停等 。     （3）数据卷：数据卷的提出是为了解决重要数据不能随意丢弃的问题   * 数据卷是一个特殊设计数据访问接口，可以将其看作Docker宿主文件系统下的一个目录或文件，可直接加载到一个容器上。 * 数据卷不受Docker存储驱动的管理，所有指向数据卷的读写操作都会绕过Union File System文件系统和存储驱动，直接以宿主机器的性能运行。 * 当一个容器被删除时，任何存储在数据卷上的数据会在Docker宿主机器上持续保存。   （4）哈希函数：如果两个输入串的hash函数的值一样，则称这两个串是一个碰撞 (Collision)。既然是把任意长度的字符串变成固定长度的字符串，所以必有一个输出串对应无穷多个输入串，碰撞是必然存在的。   1. **Docker的工作原理**   当启动一个容器时，Docker会在镜像栈的顶部增加一个新的、薄的读写层，这一层即“容器层”。当前运行容器的所有操作（比如写新文件、修改现有文件、删除文件）都写到这一读写层中。当这一容器被删除时，其读写层也被删除，而底层的镜像保持原状，而重新利用该镜像创建的应用也不保留此前的更改。这种只读层结合顶部读写层的组合被称为Union File System。在这样的架构下，多个容器可以安全的共享一个底层镜像。     1. **容器技术和虚拟机技术的区别**   容器技术不是虚拟化的替代方案，它还不能取代全系统的服务器虚拟化技术，二者对比如下所示：   |  |  |  | | --- | --- | --- | | 特性 | 容器 | 虚拟机 | | 硬盘使用 | 一般为MB | 一般为GB | | 启动 | 秒级 | 分钟级 | | 性能 | 接近原生 | 弱于原生 | | 系统支持 | 单级支持上千个容器 | 一般为几十个 |   **二、具体安装过程**  Docker 并非是一个通用的容器工具，它依赖于已存在并运行的 Linux 内核环境。Docker 实质上是在已经运行的 Linux 下制造了一个隔离的文件环境，因此它执行的效率几乎等同于所部署的 Linux 主机。  因此，Docker 必须部署在 Linux 内核的系统上。如果其他系统想部署 Docker 就必须安装一个虚拟 Linux 环境。   1. 首先在Docker的官网[Docker Hub](https://hub.docker.com/?overlay=onboarding)中下载软件包，待下载完成之后开始安装。   安装完毕之后如下图所示     1. 在下载过程中会出现相应提出，需要安装基于ubuntu系统的wsl工具。，具体过程如下：   打开 PowerShell（或 Windows 命令提示符）并输入：wsl --install执行以下操作：   * 启用可选的 WSL 和虚拟机平台组件 * 下载并安装最新 Linux 内核 * 将 WSL 2 设置为默认值 * 下载并安装 Ubuntu Linux 发行版（可能需要重新启动）   使用 WSL 安装 Linux 发行版的过程完成后，使用“开始”菜单打开该发行版（默认情况下为 Ubuntu）。 系统将要求你为 Linux 发行版创建“用户名”和“密码”。   * 此**用户名**和**密码**特定于安装的每个单独的 Linux 分发版，与 Windows 用户名无关。 * 创建**用户名**和**密码**后，该帐户将是分发版的默认用户，并将在启动时自动登录。 * 此帐户将被视为 Linux 管理员，能够运行 sudo (Super User Do) 管理命令  1. Wsl安装完毕之后，重启计算机 2. 重启之后点击docker Desktop 进入主界面并进行登录，实验过程如下所示：      1. 为了验证是否安装正确，可以通过运行基本的docker命令，并检查输出信息。   在本次实验中，以hello world程序为例，通过docker环境运行hello world项目。  首先打开powershell终端    在终端中出入如下命令：docker run hello-world，终端打印信息如下所示    观察输出信息可以发现，docker命令行提示hello world项目运行正确。因此说明docker环境安装正确。  再次查看docker Desktop界面，可以看到有如下新增的image    **三、实验的扩展**  Docker的本质是一个容器，通过对一些环境进行打包和封装便于开发人员的使用。在本次实验中，以安装python环境和ubuntu环境为例进一步体会docker的用法。具体实验过程如下所示：  **1. 安装ubuntu镜像**  （1）首先**拉取最新版的 Ubuntu 镜像,**使用docker pull ubunt命令  （2）**查看本地镜像**    可以发现，确实出现了新拉取的ubuntu镜像。  运行容器，并且可以通过 exec 命令进入 ubuntu 容器，依次执行以下命令   * docker run -itd --name test ubuntu      * docker exec -it test /bin/bash     可以看到，通过docker进入ubuntu环境，此时测试一些基本的Linux命令行如ls,ps，cd等指令    可以看到对于Linux的指令，均可以正常执行，因此ubuntu镜像的使用正确。  **2. 安装python环境**  （1）首先搜索python环境    （2）使用docker pull python:3.6安装python3.6对应的环境，待安装完之后查看docker desktop如下所示：    （3）在命令行中进行查看    可以看到镜像可以被显示，因此python环境安装正确。 | | | |
| **结论分析与体会：**  **（1）Docker对云的扩展**   1. Docker的轻量级特性使其成为未来云计算的重要拓展方向之一——边缘计算(Edge Computing)的重要使能技术 2. 基于地理分布的边缘设备的计算方式，需要一种灵活可扩展的平台来实现应用和服务的部署。 3. 该平台应当能够部署适合小型边缘设备的轻量级可重用的服务，并且不依赖于异构硬件架构。 4. 该平台还应当能够有效的与云端进行交互，从而实现分布式大数据的边缘计算和云端存储。 5. 这一平台还应易于安装、配置、管理、升级、以及调试。   **（2）容器技术优缺点**  **分析：**  1. 技术优点   * 轻量级、易扩展：虚拟机自身是一个完备系统，拥有虚拟化的硬件和特定资源，如果每个VM有2GB容量，则10个虚拟机就需要20GB；若采用容器，因为共享其操作系统内核，因此并不会占据20GB空间。 * 资源利用率高：虚拟机需要借助虚拟化软件层模拟硬件行为，而容器则直接运行在主机操作系统上。其启动时间也短。 * 简化配置、提升效率：降低了硬件资源和应用环境的耦合度，并且可以给开发者提供理想的开发环境，提升开发效率。   2. 容器技术缺点   * 安全性：容器极度依赖其主机操作系统，所以任何针对主机操作系统的攻击都会造成其安全问题。同时，主机操作系统能够看到容器中运行的一切资源。 * 隔离型相比虚拟机差   **（3）Docker的局限性**  分析：Docker并不是全能的，设计之初也不是KVM之类虚拟化手段的替代品，简单总结几点：   * Docker是基于Linux 64bit的，无法在32bit的linux/Windows/unix环境下使用 * LXC是基于cgroup等linux kernel功能的，因此container的guest系统只能是linux base的 * 隔离性相比KVM之类的虚拟化方案还是有些欠缺，所有container公用一部分的运行库 * 网络管理相对简单，主要是基于namespace隔离 * cgroup的cpu和cpuset提供的cpu功能相比KVM的等虚拟化方案相比难以度量(所以dotcloud主要是按内存收费) * Docker对disk的管理比较有限 * container随着用户进程的停止而销毁，container中的log等用户数据不便收集   **体会**  本次实验主要涉及到了Docker环境的配置以及容器虚拟化技术的有关知识。在实验中，我了解到Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中,然后发布到任何流行的Linux或Windows操作系统的机器上,也可以实现虚拟化,容器是完全使用沙箱机制,相互之间不会有任何接口。正是由于Docker轻量、快速、方便的特点使得Docker备受开发者青睐。  综上，通过本次实验，我主要熟悉了docker的基本概念，并基本掌握了docker环境的搭建。同时，通过实际的测试并在docker内搭建ubuntu和python环境，使我对容器虚拟化技术有了更加深刻的理解。 | | | |