1.1 习题

1. 填空题

(1) 资源管理

(2) 宿主机

(3) 虚拟化

(4) 容器引擎

(5) 系统 程序

2. 选择题

(1) B

(2) D

(3) A

(4) A

(5) D

3. 思考题

(1) 传统的虚拟化技术是模拟出一套硬件，在其上运行一套完整的操作系统，拥有自己独立的内核，虚拟机包含应用程序，必须的库或二进制文件，以及一个完整的 Guest 操作系统；而容器没有进行硬件虚拟，容器包含应用程序和它所有的依赖，容器中的应用进程直接运行在宿主机的内核上，与宿主机共享内核，因此容器要比传统的虚拟机更加轻便。

容器技术与虚拟化技术都将需要运行的东西进行隔离，形成一个独立的运行空间，与宿主机系统互不干扰，但又相辅相成。虚拟化技术是基于系统的隔离，它将物理层面的资源进行隔离。而容器技术与之不同，容器的隔离空间中运行的是应用程序，是基于程序的隔离，不需要将系统隔离。

相较于虚拟化技术，容器是更加快捷方便的技术，它的部署与迁移都十分快速，结构更加精简，运行速率更高。而虚拟化技术需要系统隔离，每次创建都要新建系统，为用户的操作带来不便，它的结构臃肿，无论是部署还是迁移都要消耗大量时间。

由于 Docker 容器的操作系统是共享的，虚拟化的操作系统是独立的，所以它的隔离性更强，但也注定它结构复杂，无法被广泛的应用到企业中。

- (2) **编排有序**

在以往的项目交付过程中，开发与运维常常出现问题，总会出现在开发这里能够正常运行，到了运维人员那里却无法正常运行的情况，使业务不能在第一时间完成上线，导致整个交付过程效率低下。

Docker 提供了一种全新的发布机制。这种发布机制，是使用 Docker 镜像作为统一的软件制品载体，使用 Docker 容器统一环境运行，通过 docker Hub 提供镜像统一协作，最重要的是使用 Dockerfile 定义容器内部行为和容器关键属性来做支撑，从而使整个开发交付周期都保持了环境的统一，大大的提高了产品交付效率。

Dockerfile 处于整个机制的核心位置。因为在 docker file 中，不仅能够定义使用者要在容器中进行的操作，而且能够定义容器中的运行软件需要的配置，实现了软件开发和运维能够在一个配置文件上达成统一。运维人员能够使用 docker file 在不同场合下部署出与开发环境一模一样的 Docker 容器出来。

- **高效易迁移**

Docker 容器基于开放式标准，几乎可以在任意的平台上运行，包括物理机、虚拟机、公有云、私有云、个人电脑、服务器等。这种兼容性可以轻松的让用户把一个应用程序从一个平台直接迁移到另外一个平台。

Docker 是一款轻量级应用，在一台机器上运行的多个 Docker 容器可以共享这台机器的操作系统内核；能够做到快速启动，只需占用很少的服务器资源。镜像是通过文件系统层进行构造的，并共享一些公共文件。这样就能有效的降低磁盘用量，并能够更快的下载镜像。

- **快速部署**

当公司需要进行业务迁移的时候（例如公司的 IDC 机房要进行搬迁，业务要迁移到云服务器上），通常需要将所有应用在云服务器上重新部署，这些繁琐的工作极大的浪费了人力，并降低了工作效率。利用 Docker 容器可以极大的简化这一工作，只需要在云服务器上运行相应的容器就可以了，无需考虑环境的因素，达到了快速部署。

Docker 赋予应用的隔离性不仅限于隔离，还独立于底层的基础设施。Docker 默认提供最强的隔离，因此应用出现问题，也只是单个容器的问题，而不会影响到整台机器。

4. 操作题

略

2.1 习题

1. 填空题

(1) docker CE　docker EE

(2) 企业

(3) 社区

(4) Edge　Stable

(5) 提升访问 Docker 官方网站的速度

2. 选择题

(1) B

(2) D

(3) A

(4) D

(5) B

3. 思考题

(1) docker Toolbox 是一个 Windows 系统专属的 Docker 客户端。在用户的条件无法使用 docker for Windows 安装 Docker，例如用户系统是 win10 家庭版，这时，就可以使用 docker Toolbox 安装。

(2) Linux 系统安装 Docker 建议满足两个条件，首先，用户所使用的 Linux 是 64 位的操作系统，其次，系统内核版本至少要在 3.10 版本以上。

4. 操作题

略

3.1 习题

1. 填空题

(1) 系统

(2) 宿主机

(3) 只读

(4) 描述

(5) 127

2. 选择题

(1) D

(2) A

(3) B

(4) D

(5) A

3. 思考题

(1) Docker 镜像是一个只读的文件系统，是由一层一层的文件系统组成，每一层仅包含了前一层的差异部分，这种层级的文件系统被称为 UnionFS。大多数 Docker 镜像都是在 base 镜像的基础进行构建，每进行一次新的创建就会在镜像上构建一个新的 UnionFS。

(2) 用户向 docker Hub 上推送镜像之前，需要有一个 docker Hub 账号，在 Docker 客户端使用 docker login 命令登录 docker Hub 账号。然后使用 docker tag 命令给镜像添加 tag 标签，再使用 docker push 命令向 docker Hub 推送镜像。

4. 操作题

略

4.1 习题

1. 填空题

(1) docker create　　docker start

(2) 虚拟的终端

(3) 运行（Up）　暂停（Paused）　终止（Exited）

(4) docker attach　　docker exec

(5) docker rm

2. 选择题

(1) A

(2) B

(3) A

(4) C

(5) D

3. 思考题

(1) 通过 attach 命令进入容器之后，如果直接在容器中使用 exit 命令或者 Ctrl+d 快捷键退出容器，会导致容器终止。想要退出当前容器，并且不终止容器时，可以使用 Ctrl+P+Q 快捷键退出终端。

exec 与 attach 不同，在使用 exec 进入的容器中执行 exit 命令不会终止容器，只会退出当前 bash 终端。

(2) 限制容器的资源可通过三个方面，分别是，限制容器的内存、CPU 与 Block IO。

限制容器内存时，可以在容器启动命令中添加如下参数。

- -m，--memory

设置容器可使用的最大内存，最小值是 4M。

- --memory-swap

设置容器可使用内存 swap 的最大值。

- --memory-swapiness

默认情况下，用户可以设置一个 0-100 的值，代表允许内存与交换分区置换的比例。

- --memory-reservation

设置一个内存使用的 soft limit，如果 Docker 发现主机内存不足，会执行 OOM 操作。这个值必须小于—memory 设置的值。

- --kernel-memory

容器能够使用的内核内存的大小，最小值为 4M。

- --oom-kill-disable

设置是否在运行 OOM 时终止容器进程。宿主机会在内存不足时，随机关闭一些进程，而该参数会保护容器进程不被关闭。只有通过设置-memory 限制容器内存，才可以使用该参数，否则容器会耗尽主机内存，而且导致主机应用被终止。

Docker 为容器设置 CPU 资源限制的参数是-c 或—cpu-shares，，用于配置使用 CPU 的权重，它的值是一个整数。

Block IO 表示磁盘的读写，Docker 可以通过对权重的配置，以配置 bps（每秒读写的数据量）和 iops（每秒读写的次数）的方式限制容器对磁盘读写的带宽。

限制容器 Block IO 时，可以在容器启动命令中添加如下参数。

- --device-read-bps：限制读某个设备的 bps；
- --device-write-bps：限制写某个设备的 bps；
- --device-read-iops：限制读某个设备的 iops；
- --device-write-iops：限制写某个设备的 iops。

默认情况下，所有容器对磁盘读写的带宽是相同的，通过配置-blkio-weight 参数的值（10-1000）可以指定容器 Block IO 的优先级。--blkio-weight 与-cpu-shares 类似，默认的相对权重值都是 500。

4. 操作题

略

5.1 习题

1. 填空题

(1) C/S

(2) 模块化

(3) 客户端

(4) Namespace

(5) 分配器

2. 选择题

(1) A

(2) A

(3) B

(4) C

(5) D

3. 思考题

(1) Linux 操作系统中，容器用来实现“隔离”的技术称为 Namespace。Namespace 技术实际上修改了应用进程看待整个计算机的“视图”，即它的“视线”被操作系统做了限制，只能“看到”某些指定的内容。但对于宿主机来说，这些被进行“隔离”的进程跟其他进程并没有太大区别。

Cgroups 全称是 Control group，是 Linux 内核提供的一种可以限制单个进程或者多个进程所使用资源并进行分组化管理的机制。已经通过 Linux Namespace 创建的容器，Cgroups 将对其做进一步“限制”。

(3)