Linux 操作系统 - 简答 100 题

1.简述开源软件的基本特征及其在 Linux 发展中的作用。

- 开源（Open Source）即开放源代码，开源软件就是源代码开放的软件。
- 开源软件同样有版权，受到法律保护。
- 开源软件要遵循开源协议，常见的开源协议有 GPL、LGPL、MPL、Apache、MIT、BSD 等。
- 任何人可依源代码修改后，依照同一授权条款再散布，促进技术创新。
- 在 Linux 发展中，开源软件模式使得 Linux 内核能够由全球开发者共同维护和优化，加速了其技术演进与稳定性提升。
- 开源生态为 Linux 提供了丰富的应用软件和工具，奠定了其在服务器、嵌入式系统等领域的广泛成功基础。

2.简述虚拟化技术的基本工作原理。

- 虚拟化是指通过软件技术对物理硬件资源进行抽象，将一台物理计算机“分割”为多台逻辑上的虚拟计算机，使多个操作系统和应用程序能够同时运行在同一硬件平台上，并且彼此隔离、互不影响。
- **虚拟化的两种主要架构类型：**I 型（裸机型）：虚拟机直接运行在系统硬件上，创建硬件全仿真实例，Hypervisor 直接管理调用硬件资源，不需要底层操作系统，如 VMware ESXi、Xen、KVM 等。II 型（宿主型）：虚拟机运行在传统操作系统上，作为应用程序运行，依赖宿主系统进行硬件管理，如 VirtualBox、VMware Workstation 等。
- VirtualBox 是一种常见的桌面级虚拟化软件，属于宿主型（Type 2）虚拟机监控器，它运行在宿主操作系统之上，负责创建和管理虚拟机，并将虚拟机对硬件的访问请求转交给宿主系统处理。
- 虚拟化通过多种网络模式实现虚拟机的网络通信，如 NAT、桥接和仅主机模式等，不同模式决定了虚拟机与宿主机、局域网及外部网络之间的连接方式，从而满足隔离测试或对外服务等不同应用场景。
- 虚拟化中常见术语包括宿主机（Host）、虚拟机（VM）、客户机操作系统（Guest OS）以及虚拟硬件等，这些术语共同描述了虚拟环境中各组件的角色与关系。
- 通过虚拟机监控器对硬件资源的统一管理、对指令和资源访问的控制，以及网络和存储等虚拟设备的支持，虚拟化技术实现了多操作系统并行运行，提高了资源利用率和系统灵活性。

3.简述容器技术的工作原理，并说明其三个主要特点。

- 容器技术是一种操作系统级虚拟化技术，它不是模拟完整的硬件环境，而是在同一操作系统内核之上，将应用及其依赖打包成相互隔离的运行环境，实现应用的快速部署和运行。
- 容器引擎（如 Docker）负责容器镜像的构建、存储和运行，它通过镜像分层技术快速创建容器实例，并统一管理容器的生命周期，包括启动、停止、迁移等操作。
- **容器技术的第一个主要特点 - 轻量级：**容器不需要运行独立的操作系统内核，直接共享宿主机内核，因此占用资源少、启动速度快，能够显著提高系统资源利用率和应用部署效率。
- **容器技术的第二个主要特点 - 可移植性：**容器将应用及其运行环境整体封装，保证了开发、测试和生产环境的一致性，同时容器可以在不同平台间快速迁移和部署，具有良好的可移植性和灵活性。
- **容器技术的第三个主要特点 - 隔离性：**容器依托 Linux 内核的 **Namespace** 和 **Cgroup** 机制实现，其中 Namespace 用于实现进程、网络、文件系统等资源的隔离，Cgroup 用于限制和管理 CPU、内存等资源的使用，从而保证容器之间互不干扰。

4.简述 Linux Shell 的基本功能及其在系统管理中的作用。

- Shell 是在 openEuler 操作系统中运行的一种特殊程序，位于用户与操作系统内核之间，负责接收用户输入的命令并进行解释，将需要执行的操作传递给系统内核执行。
- Shell 命令主要分为命令行式 Shell、脚本式 Shell、函数式 Shell 三类
- 命令行式 Shell 是以 Bourne Shell（sh）和 C Shell（csh）为代表，将用户输入解释为命令并执行。
- 脚本式 Shell 是以 Bash Shell（bash）为代表，读取并执行包含 Shell 命令的脚本文件。
- 函数式 Shell 是以 Korn Shell（ksh）和 Z Shell（zsh）为代表，支持函数定义和执行，可以更灵活地组织和控制脚本。

在系统管理中的作用：

- **用户与权限管理：**Shell 提供完整的用户管理命令集，包括 useradd（创建用户）、usermod（修改用户）、userdel（删除用户）、groupadd（创建用户组）等，支持用户生命周期管理和权限配置。
- **进程管理：**Shell 支持进程查看、控制和优先级调整，通过 ps、top 等命令查看进程状态，使用 nice、renice 调整进程优先级，支持前台和后台进程切换，实现系统资源的合理分配。

- **文件与目录管理：**Shell 提供丰富的文件操作命令，如 ls（列出目录）、mkdir（创建目录）、cp（复制文件）、mv（移动文件）、rm（删除文件）等，支持文件权限管理（chmod、chown、chgrp），实现文件系统的完整管理。
- **系统信息查看：**Shell 提供系统信息查询命令，如 uname（查看系统信息）、hostnamectl（查看或设置主机名）、date（查看日期时间）、whoami（查看当前用户）等，帮助管理员了解系统运行状态。
- **自动化运维：**Shell 脚本编程支持自动化或半自动化的操作系统维护管理，通过编写 Shell 脚本实现批量操作、定时任务、系统监控等功能，显著提高运维效率和系统稳定性。

5.YUM 包管理器的主要功能是什么？简述其在 openEuler 中的工作原理。

- YUM（Yellowdog Updater, Modified）是 Linux 系统中的软件包管理工具，主要用于软件包的安装、升级、卸载和查询，能够自动处理软件包之间的依赖关系，简化系统软件管理过程。

主要功能：

- **软件包安装与卸载：**YUM 以 RPM 软件包为管理对象，通过统一的命令接口对系统中的软件进行集中管理，避免了手动安装 RPM 时可能出现的依赖缺失或版本冲突问题。
- **软件包搜索与信息查询：**YUM 提供了查找、安装、删除某一个、一组甚至全部软件包的命令，支持 yum search keywords 搜索软件包、yum info package_name 查看 RPM 包信息、yum list all 输出所有软件包的信息等功能。
- **系统更新与升级：**YUM 支持系统升级更新，包括 yum update package_name 升级单个软件包、yum group update group_name 升级软件包组、yum update 更新所有软件包及其依赖的软件包、yum check-update 列出所有可更新的软件清单等操作。

工作原理：

- 在 openEuler 中，YUM 作为兼容工具存在，其底层实际采用的是 DNF（Dandified YUM）机制，但仍保留 YUM 的使用方式和配置结构，保证用户操作习惯的延续性。
- YUM 通过配置好的软件仓库（Repository）获取软件包元数据，先分析用户请求，再解析所需软件包及其依赖关系，自动从仓库下载并安装合适版本的 RPM 包。
- 借助自动依赖解析和集中仓库管理，YUM 在 openEuler 中实现了高效、安全的软件维护，提高了系统更新与运维管理的可靠性和便捷性。

6.简述 Linux 内核的核心功能及其在操作系统中的地位。

- Linux 内核是操作系统的核心组成部分，直接运行在硬件之上，负责管理和调度系统的各种资源，是连接应用程序与计算机硬件的桥梁，决定了操作系统的性能与稳定性。

核心功能：

- **进程管理功能：**内核负责进程和线程的创建、调度与终止，通过调度算法合理分配 CPU 资源，实现多任务并发执行，保证系统在高负载情况下仍能高效运行。
- **内存管理功能：**Linux 内核通过虚拟内存机制对物理内存进行统一管理，实现内存分配、回收、地址映射和交换，既提高了内存利用率，又为应用程序提供了安全、独立的运行空间。
- **文件系统管理功能：**内核通过设备驱动程序控制硬件设备的运行，并通过文件系统对磁盘等存储资源进行统一管理，使用户和应用能够以文件的方式访问各种硬件资源。

地位：

- Linux 内核提供系统调用接口，允许用户程序安全地请求内核服务，同时通过权限控制和访问控制机制保障系统的安全性和可靠性，在操作系统中处于不可替代的核心地位。
- 作为硬件抽象层，将物理设备抽象为统一接口；作为资源调度中枢，协调 CPU、内存、存储等硬件资源的高效利用；作为安全隔离边界，保障系统稳定性与安全性

7.Linux 中的 Bond（网卡绑定）技术用于实现什么目的？简述其常见应用场景。

目的：

- Bond 即网卡绑定，是将两个或者多个物理网卡绑定为一个逻辑网卡，实现本地网卡的冗余、带宽扩容和负载均衡。
- Bond 技术的主要目的是提高网络链路的可靠性和带宽利用率，通过多网卡协同工作，避免单块网卡或单条链路故障导致网络中断。

常见应用场景：

- 在服务器或关键业务系统中，Bond 常用于实现链路冗余，当某一网卡或交换链路发生故障时，系统可以自动切换到其他网卡，保障网络的连续性。
- 在高并发、高流量的业务环境中，Bond 可将多块网卡的带宽进行聚合，提升整体吞吐能力，常用于数据库服务器、文件服务器或虚拟化宿主机。
- Bond 技术广泛应用于数据中心、云平台和生产服务器环境中，与交换机的链路聚合技术配合使用，提高网络性能、稳定性和系统整体可用性。

8.RAID 技术常见的实现级别有哪些？请简述 RAID 0、RAID 1 和 RAID 5 的工作原理及适用场景。

实现级别：

- RAID（Redundant Array of Independent Disks）是一种将多块硬盘组合成一个逻辑存储单元的技术，常见实现级别包括 RAID 0、RAID 1、RAID 5、RAID 6 和 RAID 10 等，用于提升性能、可靠性或两者兼顾。

RAID0 的工作原理及适用场景：

- RAID 0 采用数据条带化方式，将数据分散写入多块硬盘中，不提供任何冗余或校验机制，可显著提高读写性能，但一旦任意一块硬盘损坏，全部数据都会丢失
- RAID 0 适用于对性能要求高、但对数据安全性要求不高的场景，如临时数据处理、缓存系统或对数据可重建的测试环境。

RAID5 的工作原理及适用场景：

- RAID 1 采用数据镜像方式，将相同的数据同时写入两块或多块硬盘，提供较高的数据安全性，适用于对数据可靠性要求较高的系统，如系统盘或关键业务服务器。

RAID6 的工作原理及适用场景：

- RAID 5 通过数据条带化与分布式奇偶校验实现数据冗余，允许任意一块硬盘损坏而不丢失数据，兼顾性能与可靠性，常用于文件服务器和中小型数据库系统。

9.磁盘 I/O 性能受哪些因素影响？简述提升磁盘 I/O 效率的常用方法。

影响因素：

- **存储介质层面：**磁盘类型是影响 I/O 性能的关键因素，不同存储介质（HDD、SSD、NVMe）在转速、延迟和吞吐能力上差异明显，同时磁盘接口类型（SATA、SAS、PCIe）也直接决定数据传输速度。
- **操作系统层面：**操作系统的 I/O 调度算法、文件系统类型（如 ext4、XFS）以及磁盘分区与对齐方式都会影响磁盘读写效率，不合理的文件系统或调度策略可能导致 I/O 瓶颈。
- **应用程序层面：**应用程序的 I/O 模式（顺序读写或随机读写）、并发访问数量以及读写块大小，都会对磁盘性能产生影响，高并发随机 I/O 通常会显著降低磁盘效率。

提升 I/O 效率的常用方法：

- 通过采用高速存储设备（如 SSD）、配置 RAID、增加磁盘数量或使用缓存设备，可以有效提升磁盘整体吞吐能力和可靠性，适用于对性能要求较高的场景。
- 在软件层面，可通过调整 I/O 调度策略、合理选择文件系统、启用缓存机制以及优化应用的读写方式，减少不必要的磁盘访问，从而提高磁盘 I/O 效率。

10.简述 LVM（逻辑卷管理）的基本组成及其在磁盘管理中的优势。

LVM 定义：

- LVM（Logical Volume Manager）是一种灵活的磁盘管理机制，它在物理磁盘与文件系统之间引入逻辑层，使磁盘空间的管理不再受限于固定分区结构。

基本组成：

- LVM 主要由物理卷（PV）、卷组（VG）和逻辑卷（LV）三部分组成，其中物理卷是实际磁盘或分区，卷组用于整合多个物理卷的存储空间，逻辑卷则是最终供用户使用的存储单元。

磁盘管理中的优势：

- **动态调整容量：**逻辑卷从卷组中按需分配空间，可灵活调整大小，并可在其上创建文件系统，使存储资源能够根据实际需求动态使用。
- **灵活整合存储：**LVM 支持在线扩容和缩减、磁盘空间动态分配以及多磁盘统一管理，避免了传统分区方式在容量调整时需要重新分区或迁移数据的问题。
- **简化管理操作：**LVM 广泛应用于服务器和企业级系统中，特别适合存储需求变化频繁、对系统稳定性和管理灵活性要求较高的磁盘管理环境。
- **数据冗余与快照：**LVM 能实现数据冗余和快照，提高可靠性。通过镜像卷（mirror）或快照（lvcreate -s）保护数据，快速恢复或克隆，为关键业务数据提供额外的安全保障。

11.Linux 支持哪些主流文件系统？请简述 ext4、XFS 的基本特性。

- Linux 支持多种文件系统，常见的主流类型包括 ext4、XFS、Btrfs、FAT32、NTFS 等，其中 ext4 和 XFS 是服务器和桌面系统中应用最为广泛的两种文件系统。

ext4 基本特性：

- ext4 是 ext 系列文件系统的第四代版本，支持大容量文件和分区，具有成熟稳定、兼容性好等特点，是目前 Linux 系统中默认和最常用的文件系统之一。

- ext4 采用日志（Journal）机制，能够在系统异常断电或崩溃后快速恢复文件系统状态，同时在顺序读写和日常使用场景中表现稳定，维护成本较低。

XFS 基本特性：

- XFS 是一种高性能 64 位日志文件系统，擅长处理大文件和高并发 I/O 操作，支持大容量存储和在线扩展，常用于服务器和数据密集型应用环境。
- ext4 适合通用场景和系统盘，强调稳定性和兼容性；XFS 更适合大容量磁盘和高并发访问场景，如数据库和文件服务器，两者在 Linux 系统中各有侧重。

12.Apache 模块（Module）的作用是什么？举例说明两个常用模块的功能。

作用：

- Apache 大多数功能被分散到各模块中，各模块在系统启动时按需载入，安装 Apache 时会默认安装一些模块，通过加载不同模块，可以为 Web 服务器增加新的处理能力，而无需修改 Apache 核心程序。
- Apache 采用模块化设计，管理员可根据实际需求选择性启用或禁用模块，从而减少资源消耗、提升安全性和灵活性，使服务器配置更加可控。

举例说明：

- mod_actions 模块用于将特定类型的请求或错误状态映射到指定的脚本或程序处理，常用于实现对 CGI 程序的调用或自定义错误处理逻辑。
- mod_alias 模块主要用于 URL 路径映射与重定向，通过别名（Alias）或重定向（Redirect）机制，将访问请求映射到服务器上的其他目录或资源。
- 通过合理配置 mod_actions、mod_alias 等模块，Apache 能够实现更灵活的请求处理和资源映射，从而满足多样化的 Web 服务部署需求。

13.什么是代理服务？简述其在企业网络中的典型应用。

- 代理服务（Proxy Service）是一种通过中间服务器（代理服务器）转发网络请求或数据的技术，其核心作用是隔离客户端与目标服务器的直接通信，并在转发过程中提供额外功能（如访问控制、缓存、负载均衡、安全防护等）。
- 代理服务可以实现互联网与局域网之间的通信，分为正向代理和反向代理两种。
- 正向代理位于客户端一侧，代表客户端向外部服务器发起访问，常用于企业内部员工统一上网出口、访问控制、内容过滤以及隐藏内部用户真实地址。

- 反向代理位于服务器一侧，代表服务器接收客户端请求，并将请求转发给后端真实服务器，客户端通常并不感知其存在，常用于对外服务的统一入口。

应用：

- 在企业环境中，反向代理常用于负载均衡、访问加速和安全防护，如隐藏后端服务器、抵御部分网络攻击、提升系统可用性。
- 通过正向代理与反向代理的合理部署，企业可以同时实现网络安全管控、访问审计、性能优化和高可用架构，是企业网络中重要的基础服务组件。

14.数据库服务在 Linux 系统中通常以何种方式部署？简述其基本管理流程。

部署方式：

- 在 Linux 系统中，数据库服务通常以独立服务进程的方式部署，常见形式包括通过系统软件包（如 RPM/YUM）安装、源码编译安装或容器化方式部署。
- 数据库安装完成后，会以守护进程（Daemon）的形式在后台运行，并监听指定端口，对外提供数据存储和访问服务。

管理流程：

- 数据库服务通常由系统服务管理工具（如 systemd）统一管理，管理员可通过相关命令对数据库进行启动、停止、重启和状态查看操作。
- 数据库管理包括配置参数调整、用户与权限管理、数据备份与恢复等操作，以保证数据库服务的安全性、稳定性和数据完整性。
- 在生产环境中，还需对数据库服务进行性能监控和日志管理，并通过访问控制、防火墙和定期更新等措施，保障数据库在 Linux 系统中的可靠运行。

15.文件共享服务的三种常见协议是什么？简述其各自适用场景。

常见协议：

- 文件共享服务用于实现网络中多台主机之间的文件访问与交换，Linux 服务器中常见的文件共享协议主要包括 **NFS、SMB/CIFS 和 FTP**。

适用场景：

- NFS（Network File System）主要应用于 Linux/Unix 系统之间的文件共享，可将远程目录挂载为本地目录使用，适合局域网环境下的服务器集群和内部文件共享。
- SMB/CIFS 协议常用于 Windows 与 Linux 混合网络环境，通过 Samba 服务实现跨平台文件共享，适合企业办公网络中的部门文件服务器。
- FTP（File Transfer Protocol）主要用于文件传输而非实时共享，适合大文件上传下载、软件分发和数据交换等场景。
- 在实际应用中，应根据操作系统类型、网络环境和安全需求选择合适的文件共享协议，以满足不同业务场景下的文件共享需求。

16.FTP 服务有哪两种连接模式？请说明主动模式与被动模式的工作机制及适用场景。

连接模式：

- FTP 服务在进行文件传输时，采用**控制连接**和**数据连接**分离的工作方式，根据数据连接建立方式的不同，FTP 分为主动模式（Active）**和**被动模式（Passive）两种。

主动模式的工作机制及适用场景：

- 在主动模式下，客户端首先主动连接服务器的 21 端口建立控制连接，随后服务器从 20 端口主动向客户端发起数据连接完成文件传输。
- 主动模式适用于客户端位于公网、无防火墙或端口限制较少的网络环境，但在客户端存在防火墙或 NAT 的情况下，服务器主动连接客户端可能会被阻断。

被动模式的工作机制及适用场景：

- 在被动模式下，客户端先连接服务器 21 端口建立控制连接，服务器再开放一个随机高端口并通知客户端，由客户端主动连接该端口建立数据连接。
- 被动模式适用于客户端位于防火墙或 NAT 之后的网络环境，是当前互联网中最常用的 FTP 连接方式，安全性和兼容性较好。

17.KVM 虚拟化技术的基本架构包含哪些核心组件？简述其协作关系。

- KVM（Kernel-based Virtual Machine）是一种基于 Linux 内核的虚拟化技术，它将 Linux 内核本身转变为虚拟机监控器，实现硬件资源的虚拟化管理。
- KVM 以内核模块的形式存在，主要包括 kvm 和 kvm-intel 或 kvm-amd 模块，负责利用 CPU 的硬件虚拟化特性，实现虚拟机的创建和运行。

- QEMU 运行在用户空间，为虚拟机提供设备模拟和 I/O 处理功能，如磁盘、网络和显卡等虚拟硬件，并与内核态的 KVM 模块协同工作。
- KVM 通常通过 libvirt、virt-manager 等管理工具进行统一管理，这些工具负责虚拟机的配置、生命周期管理和资源调度。
- 在 KVM 架构中，Linux 内核提供虚拟化基础，KVM 模块负责 CPU 虚拟化，QEMU 提供设备模拟，管理工具负责统一控制，各组件协同完成虚拟机的高效运行。

18.Docker Engine 架构由哪三个核心部分组成？说明它们之间的协作关系。

组成部分：

- Docker Engine 是 Docker 容器运行的核心平台，采用客户端—服务器架构，由多个组件协同工作，完成容器的构建、运行和管理。三个核心部分是 Docker client、Docker daemon、Docker registries。

协作关系：

- Docker Client 是用户与 Docker Engine 交互的入口，用户通过 docker 命令向 Docker 服务端发送请求，用于创建、启动、停止和管理容器(Docker Client 将发送命令到 Docker daemon，由其执行)。
- Docker Daemon（dockerd）运行在后台，负责接收客户端请求，管理镜像、容器、网络和存储等资源，是 Docker Engine 的核心执行组件。
- Docker Registry 用于集中存储和分发 Docker 镜像，Docker Daemon 可从镜像仓库拉取镜像或将镜像推送到仓库，常见的有 Docker Hub 和私有仓库。
- Docker Client 负责发出操作指令，Docker Daemon 负责具体执行并管理资源，而 Docker Registry 提供镜像存储与获取支持，三者协同完成容器应用的全流程管理。

19.PROC 文件系统的作用是什么？它如何为用户提供内核运行时信息？

PROC 文件系统的作用：

- PROC 文件系统（/proc）是一种由 Linux 内核提供的**虚拟文件系统**，并不占用磁盘空间，而是以内存中的数据结构形式存在，用于反映系统和内核的运行状态。
- PROC 文件系统的主要作用是向用户空间提供内核运行时信息，使用户和系统管理员能够实时查看系统硬件、内核参数、进程状态等关键运行数据。

提供内核运行时信息方式：

- 内核将自身维护的运行数据以“文件”的形式映射到 /proc 目录中，例如 CPU 信息、内存使用情况、系统负载等，用户可通过普通文件操作命令进行访问。
- 在 /proc 目录下，每个正在运行的进程都会对应一个以进程号命名的子目录，目录中包含该进程的状态、资源占用和运行参数等信息，便于进程管理与监控。
- 通过 PROC 文件系统，用户无需直接操作内核即可获取系统运行状态，并可配合监控和管理工具实现系统调优与故障排查，是 Linux 系统管理的重要信息接口。

20.SELinux 支持哪三种运行模式？请说明每种模式的特点及适用场景。

- SELinux（Security-Enhanced Linux）是 Linux 系统中的强制访问控制机制，用于增强系统安全性，其运行模式决定了安全策略的执行方式。
- 在 Enforcing 模式下，SELinux 会严格执行安全策略，对所有不符合策略的访问行为进行阻止并记录日志，适用于对安全性要求较高的生产环境。
- 在 Permissive 模式下，SELinux 不会阻止违规访问，只会记录相关日志，便于管理员分析和调试安全策略，常用于测试和排错阶段。
- 在 Disabled 模式下，SELinux 完全关闭，不进行任何访问控制，也不记录相关日志，系统安全性降低，一般不推荐在生产环境中使用。
- 在实际部署中，通常在调试阶段使用 Permissive 模式，生产环境启用 Enforcing 模式，Disabled 模式仅在特殊需求或临时情况下使用。

21.简述 SELinux 如何通过安全上下文和策略规则实现强制访问控制。

强制访问控制机制（Mandatory Access Control，MAC）是一种由系统管理员从全系统的角度定义和实施的访问控制机制，通过标记系统中的主客体，强制性地限制信息的共享和流动，使用户只能访问与其相关的、指定范围的信息，防止信息泄密，杜绝访问权限的交叉混乱。
SELinux 是访问控制机制在 Linux 内核上的实现。

SELinux 的工作主要通过安全上下文（核心）和安全策略协同实现。

安全上下文是 SELinux 为所有主体（进程）、客体（文件、端口等）分配的标识，由 4 个部分组成：

a. user：SELinux 的用户类型，如 user_u（普通用户登录系统后的预设）、system_u（开机过程中系统进程的预设）、root（root 用户登录后的预设）、unconfined_u（多数本地进程运行的预设）。标记主体 / 客体的所属范畴。
b. role：角色标识。定义文件（object_r）、进程和用户（system_r）的角色，角色可以限制“type”的使用范围。
c. type：数据类型，是定义何种进程类型访问何种文件对象目标的策略，是访问控制的关键依据。
d. security level：安全等级，每个对象有且只有一个级别，等级为 s0~s15，s0 等级最低。策略默认等级为 s0。

安全策略是定义“主体（进程）以何种方式访问客体（文件、端口等）”的权限集合，明确操作的“允许 / 拒绝”。openEuler 内置 3 种安全策略：

1. targeted：默认值，表示部分程序受到 SELinux 的保护，对系统中目标网络的进程进行访问控制，如 dhcpd、httpd、named、nscd、ntpd、portmap、snmpd、squid 以及 syslogd 等。
2. minimum：targeted 的简化版，仅选定的程序受到保护。
3. mls（strict）：Multi-Level Security，多级安全限制。对系统中所有的进程与操作进行严格访问控制，属于较严格的规则集合。

当主体尝试访问客体时，SELinux 会先提取双方的安全上下文，再匹配当前启用的策略规则：若规则允许 “主体 type 访问客体 type”（同时满足 user、role、security level 的限制），则放行访问；否则直接阻断操作，并记录审计日志，实现系统级的强制访问管控。

22.主机防火墙与网络防火墙在部署位置和防护粒度上有何区别?

- 防火墙是服务器安全的重要保障系统，遵循允许或拒绝业务往来的网络通信机制，提供网络通信过滤服务。从保护对象上区分，可以分为主机防火墙和网络防火墙。
- 主机防火墙：是部署在单个主机操作系统中的软件防火墙，属于典型的包过滤防火墙，仅作用于所在的单台主机。
- 主机防火墙防护粒度较细：将网络层作为数据监控对象，对每个数据包的头部、协议、地址端口及类型信息进行规则分析与数据包的处理（如进入、丢弃或拒绝等），从而实现针对单个主机进行防护。
- 网络防火墙：是部署在两个网络（如内网与公网、不同子网）之间的设备或一整套装置，通常部署于网络边界位置，作用于整个网络区域。

- 网络防火墙防护粒度较粗：它以网段间的整体流量为监控对象，会将网络划分为可信与不可信区域，对区域间流入、流出的流量进行过滤；仅能基于地址段、端口范围、协议类型等粗维度开展管控，从而实现对整个网络的边界防护。

23.防火墙在实际应用中存在哪些局限性？请列举五项。

- 防火墙可以阻断攻击，但不能消灭攻击源。
- 防火墙不能抵抗最新的未设置策略的攻击漏洞。
- 防火墙的并发连接数限制容易导致服务拥塞或溢出。
- 防火墙对针对服务器合法开放的端口的攻击无法阻止。
- 防火墙对系统内部发起的攻击无法阻止。
- 防火墙本身也会出现问题或受到攻击。
- 防火墙无法防御病毒。

24.安全审计通常包括哪四项主要内容？

安全审计机制：虽然 openEuler 不能预测何时服务器会遭受攻击，但是可以记录入侵者的行踪，记录事件信息和网络连接情况，信息保存到日志文件中，为后续复查提供支持。通过安全审计，可以持续性、周期性的对操作系统进行安全评估，及时发现安全漏洞并进行修复，提高主机的安全性。安全审计包含以下 4 项：

1. 端口扫描与服务探测。在目标主机处于开机状态时，通过扫描与探测，获取其开放端口、运行的服务程序及版本、操作系统及内核版本等关键信息。
2. 以攻击方式进行探测。根据获取到的目标主机服务程序及版本信息，检索安全漏洞数据库，获取针对性的攻击脚本，开展对目标主机系统的尝试性攻击，并记录目标主机对攻击的响应信息。
3. 对数据进行分析并形成报告。对获取的响应信息进行分析，比对安全漏洞信息数据库，明确目标主机确实存在的安全漏洞信息，形成安全审计报告。
4. 安全风险处理。系统管理员根据安全审计报告的内容，逐项对照解决安全风险。

25.简述安全检测在系统安全防护中的作用，并列举两种常见手段。

作用：

安全检测是使用工具对系统进行扫描检测，验证是否存在安全风险或漏洞，对系统与业务进行整体的安全评估。作用如下：

1. 提前识别安全隐患：主动发现系统漏洞、配置缺陷、违规开放端口等风险点，避免被恶意利用。
2. 实时监控异常行为：持续监测系统的网络连接、进程运行、文件操作等动态，及时捕捉暴力破解、恶意扫描等攻击迹象。
3. 支撑安全加固与应急响应：为系统加固提供精准方向（如针对漏洞修复补丁），同时为攻击事件后的溯源分析提供数据依据。

常见的安全检测手段：

- 端口扫描与主机探测（以 Nmap 为例）：通过发送网络数据包，探测目标主机的存活状态、开放端口、运行服务及版本信息，判断是否存在未授权开放的高危端口（如未授权的 22、3306 端口）。
- 系统漏洞扫描（以 OpenVAS 为例）：基于已知漏洞数据库，对 openEuler 系统内核、已安装软件（如 Apache、MySQL）、配置文件等进行自动化检测，识别是否存在可被利用的安全漏洞（如缓冲区溢出、权限绕过漏洞）。

26.简述 Linux 操作系统版本升级的主要优势。

操作系统版本升级是为了保持系统的最新状态，适应不断变化的应用和硬件环境，提高系统的性能和稳定性，并保障数据安全和应用体验。优点如下：

1. **增强硬件支持与兼容性：**增加对硬件的支持，优化系统性能，提升用户体验。新版本通常包含最新的硬件驱动程序，支持新型号的 CPU、显卡、网卡等设备，确保系统在新硬件环境下的稳定运行
2. **修复安全漏洞与提升稳定性：**修复漏洞，提高系统的稳定性和安全性。Linux 社区会定期发布安全补丁，以修复已知漏洞，升级系统可以确保系统受到最新的安全修复的保护，防止黑客攻击和数据泄露。
3. **获取新功能与工具：**提供更多功能和工具，方便系统使用和管理。Linux 社区不断创新，为用户带来新的功能和改进，通过升级可以获得最新的功能和性能提升，提高系统体验。
4. **性能优化与资源管理改进：**升级 Linux 系统通常会带来性能的提升，包括更好的内存管理、文件系统优化以及更高效的资源利用。内核版本的更新对 CPU 调度、内存管理、文件系统等方面都有显著改进，能够增加系统的响应速度和整体的运行效率。

5. **应用程序兼容性保障：**通过升级系统，可以确保系统与最新的应用程序和库文件保持兼容，从而避免可能的兼容性问题。新版本通常提供更好的软件包管理和依赖关系处理，确保应用程序的正常运行

27.简述 Linux 操作系统版本升级可能带来的主要风险。

1. **兼容性：**可能会引入新的编程接口或更改现有的接口，导致某些应用程序和外部设备无法正常工作。
2. **成本增加：**可能需要额外的技术支持。
3. **数据丢失风险：**可能会导致数据丢失或配置问题。
4. **服务中断与业务影响：**升级过程需要重启系统，导致服务中断，影响业务连续性。升级后某些服务可能无法正常启动或工作异常，需要额外时间进行故障排查和修复，对生产环境造成严重影响。
5. **性能回归与资源消耗增加：**新版本可能存在性能回归问题，某些优化可能不适合特定硬件环境，导致系统性能下降。同时新版本通常对硬件资源要求更高，老旧设备可能出现资源不足的情况，影响系统响应速度。

因此，在生产环境中进行操作系统版本升级，需要进行充分的评估和测试。

28.简述 openEuler 操作系统的主要特点。

1. **开源开放与社区驱动**

   openEuler 基于 Linux 内核，遵循开源协议，由开放原子开源基金会托管。其开发与演进依托全球开源社区协作，具备良好的开放性与可持续性。

2. **全面架构支持与硬件适配**

   系统支持鲲鹏（ARMv8）、x86 等多种主流处理器架构，具备优秀的跨平台兼容性，可灵活部署于服务器、云计算、边缘计算及嵌入式等多种硬件环境。

3. **增强的安全与可信能力**

   内置安全框架，支持强制访问控制、安全审计、防火墙等机制，为企业级应用提供从启动到运行的全栈可信保障。

4. **高可靠与高可用支持**

   提供存储高可用方案（如 LVM、RAID），保障系统在关键业务场景中的稳定运行与快速恢复。

5. **云原生与容器化支持**

原生支持容器技术和云原生应用生态，提供完整的容器运行时、编排和管理工具。支持 Kubernetes、Docker 等主流容器技术，为云原生应用提供了理想的运行平台。

6. **企业级维护与技术支持**

提供 LTS（长生命周期支持）版本，确保企业用户获得长期稳定的技术支持。建立了完善的技术服务体系，包括文档、培训、技术支持等，为企业级部署提供了可靠保障。

29.简述 openEuler 操作系统的典型应用场景。

1. **企业级业务服务器场景**

openEuler 可作为 Web 服务器（Apache/Nginx 网站服务）、数据库服务器（MySQL 数据库服务）、文件服务器（FTP 文件服务、NFS 文件服务、Samba 文件服务）的底层操作系统，支撑企业核心业务应用的稳定运行；结合 LVM、RAID 等存储高可用技术，进一步提升业务连续性。

2. **云计算基础设施场景**

openEuler 作为云平台的宿主机系统，支持虚拟化集群（KVM 虚拟化）、容器化微服务（Docker 容器）的部署与管理，可适配公有云、私有云的基础设施构建，同时兼容多架构硬件（鲲鹏、x86 等），满足云环境的异构算力需求。

3. **边缘计算节点场景**

依托自身多架构兼容（鲲鹏、ARM、x86）与系统组件可裁剪的特性，openEuler 可作为工业物联网终端、智能网关等边缘设备的操作系统，支撑边缘数据的本地低延迟处理，匹配边缘场景的算力与响应需求。

4. **企业级运维管理平台场景**

openEuler 可搭建统一的运维管理平台，通过 Cockpit 可视化工具、系统日志审计、SELinux 安全策略配置等功能，实现对企业异构 IT 设备的集中监控、故障排查与安全管控，降低跨设备运维成本。

5. **开源软件开发与测试场景**

基于 openEuler 的开源特性与多架构兼容能力，可作为开源软件的开发与测试环境，支持 C/C++、Python 等主流开发语言，适配内核驱动开发、云原生应用开发等场景；同时依托开源社区生态，开发者可快速获取技术支持，验证软件在不同硬件架构下的兼容性。

30.Linux 命令通常由哪三个部分组成？请分别说明其含义。

命令的基本要素为命令、选项和参数。

1. 命令表示要执行的操作，其语法格式如下：

> 👋 `command [选项] [参数]`
>
> []表示可选项，有些命令不写选项和参数也可以执行，有些命令需同时附带选项和参数。命令执行需附带参数指定操作对象，若省去参数，则使用命令默认参数。

2. 选项表示要如何执行该操作。不添加选项，命令只能执行最基本的功能；增加选项，则能执行更多功能或显示更丰富的数据。
3. 选项分为短格式选项和长格式选项两种。短格式选项是长格式选项的简写，用一个减号“-”和一个字母表示，例如 `ls -l`。长格式选项是完整单词，用两个减号“--”和一个单词表示，例如 `ls --all`。
4. 参数是命令的操作对象。通常情况下，文件、目录、用户和进程等都可以作为参数。命令一般都需要参数，用于指定命令操作对象是谁。
5. 必选参数是命令的强制操作对象，是命令必须指定的操作目标（如 `cp 源文件 目标路径` 中的 “源文件”“目标路径”），若省略则命令无法执行。例如 `rm` 命令必须指定要删除的文件 / 目录，否则会提示参数缺失。可选参数是命令的默认 / 扩展操作对象。是命令可省略的操作目标，省略时命令会使用默认参数执行。例如 `ls` 省略参数时，默认操作对象是 “当前工作目录”； `cd` 省略参数时，默认切换到当前用户的家目录。

31.Linux 系统如何通过文件权限位实现对文件的访问控制？说明读、写、执行权限在文件和目录中的不同含义。

Linux 为每个文件 / 目录定义了三类访问主体（所有者 u、所属组 g、其他用户 o），并为每类主体分配读（r）、写（w）、执行（x）三个权限位，通过权限位的组合限制不同主体的访问行为，如 `rwxr-xr--` （所有者拥有全部权限，所属组可读可执行，其他用户仅可读）用数字表示为 `754` （r=4、w=2、x=1，权限组合对应数字之和。如 `rwx` 对应 7， `r-x` 对应 5）。

访问控制的实现：系统会先识别访问者属于哪类主体（所有者 / 所属组 / 其他），再检查该主体对应的权限位是否允许目标操作（如读取、修改），从而决定是否放行访问；可通过 chmod（修改权限）、chown（修改所有者）等命令调整权限规则。

读、写、执行权限在文件与目录中的不同含义：

权限类型	文件中的含义	目录中的含义
读（r）	可读取文件的内容（如用 `cat`、`less`、`more` 查看）	可列出目录内的文件 / 子目录（如用 `ls` 查看目录内容）
写（r）	可修改文件的内容（如用 vi 编辑）、删除文件	可在目录内创建 / 删除 / 重命名文件 / 子目录（如 `touch` 新建文件、`rm` 删除文件）
执行（x）	可运行文件（如执行脚本、二进制程序）	可进入该目录（如用 `cd` 切换到目标目录）

32.Linux 系统中用户权限是如何通过用户、组和权限模型进行管理的?

1. **用户（User）：权限的独立主体**

每个用户对应唯一的 UID（用户 ID），`root` 用户 UID 固定为 0，拥有系统最高权限；普通用户 UID 默认从 1000 开始，仅具备有限操作权限。可通过 `useradd`（创建）、`usermod`（修改属性）、`userdel`（删除）等命令管理，是权限分配的最小独立单元（对应书中项目二“用户管理” 实操）。

2. **组（Group）：权限的批量管理载体**

每个组对应唯一的 GID（组 ID），用于批量管理多个用户的权限。每个用户默认属于 1 个**主组**（创建用户时自动生成同名主组），还可加入多个**附加组**（通过 `usermod -G 组名 用户名` 添加）；可通过 `groupadd`（创建）、`groupmod`（修改）、`groupdel`（删除）命令管理，实现“一组多用户” 的权限批量分配（书中 “组管理” 任务核心内容）。

3. **权限位：文件 / 目录的访问规则定义**

系统为每个文件 / 目录配置三类主体的权限，格式为 `rwxr-xr--`（依次对应**所有者 u、所属组 g、其他用户 o**），每类主体对应读（r）、写（w）、执行（x）三种权限（r=4、w=2、x=1，支持数字 / 符号两种配置方式），通过 `chmod` 命令修改（书中 “授权管理” 实操核心）。

4. **权限匹配优先级：访问请求的判定逻辑**

用户访问文件 / 目录时，系统按 “**所有者 > 所属组 > 其他用户**” 的优先级匹配权限：优先判断是否为所有者，匹配则应用所有者权限；否则判断是否属于所属组，匹配则应用所属组权限；均不匹配则应用其他用户权限，最终决定是否允许操作（书中权限生效规则的核心）。

5. **权限与用户 / 组的协同管理：实操落地流程**

典型流程为：① `groupadd webgroup` 创建组；② `useradd -g webgroup user1` 创建用户并指定主组；③ `usermod -G webgroup user2` 将其他用户加入组；④ `chmod u=rwx,g=rx,o=r /var/www/html` 配置目录权限；⑤用户访问时，系统自动按优先级匹配权限（书中 “用户 - 组 - 权限联动配置” 的实操流程）

33.列举 vi 文本编辑器的三种工作模式，并说明各模式之间的转换方式。

1. 命令模式：vi 默认工作模式，可转换为文本编辑模式和末行模式。在命令模式下，从键盘上输入任何字符都被当作命令来解释，而不会在屏幕上显示。如果输入的字符是合法的 vi 子命令，则 vi 就会完成相应的操作。
2. 文本编辑模式：用于字符编辑。在命令模式下输入 i（插入命令）、a（附加命令）等命令后进入文本编辑模式。按“Esc”键可从文本编辑模式返回到命令模式。
3. 末行模式：也称 ex 转义模式。在命令模式下，按“:”键进入末行模式，此时 vi 会在屏幕底部显示“:”符号作为末行模式的提示符，等待用户输入相关命令。命令执行完毕后，vi 自动回到命令模式。
4. **模式转换关系**：命令模式是核心转换枢纽，可通过 i/a/o 等命令进入文本编辑模式，通过":"键进入末行模式；文本编辑模式和末行模式都可通过"Esc"键返回命令模式。
5. **各模式核心功能**：命令模式主要用于光标移动、文本删除复制等操作；文本编辑模式专门用于文本输入和修改；末行模式主要用于文件保存退出、搜索替换等高级操作。

34.openEuler 系统支持哪两种任务计划方式？请分别说明其特性与使用方法。

1. openEuler 中的任务计划包括一次性定时任务（at）和周期性任务（crontab）。允许用户在特定的时间自动执行命令或脚本，实现各种自动化的操作，从而提高工作效率和系统的稳定性。
2. **一次性定时任务（at）：**默认没有安装。用于在指定时间执行一次的临时任务（如临时文件清理、单次数据同步）；基于 `atd` 服务运行，仅执行一次后自动失效。
3. **使用方法：**

- 直接指定时间：执行 `at 22：00` （表示今天 22 点执行），输入要执行的命令后按 `Ctrl+D` 确认；
- 指定脚本文件：执行 `at 22:00 -f /root/temp_clean.sh` （调用脚本执行）；

4. **周期性任务（crontab）：**默认已经安装。用于周期性、重复执行的任务（如每日备份、定时日志清理），是 openEuler 中最常用的任务计划方式；基于 crond 服务运行，支持精确到分钟的周期配置
5. **使用方法：**

- 按 “分 时 日 月 周 命令 / 脚本路径”的格式编写，例如 “`0 3 * * * /root/backup.sh`” 表示每天凌晨 3 点执行备份脚本；

35.列举 Apache HTTP Server 的五项主要特征。

1. 支持最新的 HTTP 协议和多种方式的 HTTP 认证。
2. 支持基于文件的配置。
3. 支持基于 IP 和域名的虚拟网站配置。
4. 支持通用网关接口，支持 PHP、FastCGI、Perl、JavaServlets 等。
5. 支持服务器状态监控。
6. 支持服务器日志记录和日志格式自定义设置。
7. 支持服务器端包含指令（SSI）。
8. 支持安全 Socket 层（SSL）。
9. 集成代理服务器模块。

36.Apache HTTP Server 支持哪三种多处理模块（MPM）工作模式？简述各自的工作原理。

Apache HTTP Server 有 prefork、worker 和 event 三种工作模式，具体描述如下：

1. **prefork 工作模式**

- 是稳定模式，Apache 启动之初预先派生一些子进程，每个子进程同一时间仅能处理一个请求，请求会先放入队列，等待可用进程后才被处理。
- 可通过 StartServers （初始工作进程数）、 MinSpareServers （空闲子进程最小数量）、MaxSpareServers （空闲子进程最大数量）、 MaxRequestWorkers （最大空闲线程数）四个指令调节子进程生成。
- 优点是减少频繁创建和销毁进程的开销；缺点是并发请求高时需增大 MaxRequestWorkers 值，内存小的服务器需减少该值避免崩溃。

2. **worker 工作模式**

- 采用多进程和多线程混合模式，启动时先创建少量子进程，每个子进程再创建一些线程，线程共享父进程内存空间，内存占用更少，处理更轻量。
- 高并发下引入多进程，比 prefork 有更多可用进程；但单进程内线程出错可能导致进程及子线程出错，仅部分服务受影响。

3. **event 工作模式**

- 和 worker 模式相似，解决了 keep-alive 场景下线程长期被占用的资源浪费问题。
- 由专门线程管理 keep-alive 类型线程，真实请求到来时才将请求传递给服务线程，执行完毕后释放，增强了高并发场景下的请求处理能力。

4. **event 模式优势**：

专门用于处理高并发场景，通过分离连接管理和请求处理，增强了高并发场景下的请求处理能力。特别适合长连接和大量并发连接的场景，是 Apache 2.4 版本后的推荐模式。

5. **模式选择建议**：

prefork 适用于兼容性要求高的传统应用；worker 适用于内存有限但需要较好并发性能的场景；event 适用于现代高并发 Web 应用，特别是大量 keep-alive 连接的场景。

37.为提升 Apache HTTP Server 的安全性，可采取哪些措施？请列举五项。

1. 基于 IP / 用户认证限制敏感目录访问。
2. 隐藏服务器敏感信息。
3. 开启 openEuler 的 SELinux 强制访问控制，并配合防火墙进行安全防护。
4. 禁止 Apache 的网站目录浏览。
5. 禁用不必要的模块与危险请求方法 .

38.什么是反向代理？请列举其四项主要作用。

反向代理一般是为服务器架设的。位于网站服务器与互联网之间。在客户端看来它就像是一个普通的的网站服务器，客户端不需要做任何配置。客户端发送请求到代理服务器，代理服务器决定将这些请求转发到何处。作用：

1. 隐藏服务器真实 IP，客户端只能看到代理服务器地址。
2. 实现业务负载均衡，代理服务器可根据网站的负载情况，将客户端请求分发到不同的网站服务器。

3. 提高业务访问速度，代理服务器提供缓存服务、提高网站等业务的访问速度。
4. 提供安全保障，代理服务器可作为应用层防火墙，为内部网站提供安全防护。

39.列举 Nginx 的五项主要特性。

1. 基于模块化的结构。
2. 基于 EPOLL 事件驱动模型。
3. 提供反向代理服务，可使用缓存加速反向代理，支持简单的负载均衡和容错。
4. 支持基于文件的配置。
5. 支持基于 IP 和域名的虚拟网站配置。
6. 支持 MP4、FLV 视频流式服务。
7. 支持嵌入 Perl 语言。
8. 支持 FastCGI、Uwsgi、SCGI。
9. 支持 HTTP/2、HTTP/3。
10. 支持 IMAP、POP3、SMTP 代理。
11. 支持 TCP 和 UDP 的通用代理。
12. 支持 Windows、Linux、UNIX 操作系统。
13. 兼容 X86 与 ARM 架构。
14. 采用 2-clause BSD-like 开源协议，可以免费使用并可以商业化。

40.Nginx 通过哪个模块实现负载均衡？请简述其支持的四种负载均衡策略。

Nginx 主要通过 `ngx_http_upstream_module` 和 `ngx_http_proxy_module` 模块实现网站的负载均衡，支持轮询（round-robin）、最少连接优先（least-connected）、持续会话（ip-hash）及权重负载均衡（Weighted load balancing）等负载均衡方式。

1. 轮询（round-robin）：Nginx 将客户端请求循环发送给各网站服务器节点，各网站服务器节点接收到的请求数量基本是一样的。Nginx 默认为轮询模式。
2. 最少连接优先（least-connected）：Nginx 将避免把请求发送到繁忙的网站服务器节点，而是将请求发送给不太繁忙的网站服务器节点。
3. 持续会话（ip-hash）：Nginx 将客户端的会话一直保持在同一台网站服务器节点，直到该网站服务器节点不可用，一般用于需要维持 session 会话的网站业务。

4. 权重负载均衡（Weighted load balancing）：Nginx 根据设置的网站服务器权重信息，将客户端请求按照权重进行分发，权重值与访问比率成正比，一般用于服务器性能不均的情况。

41.Nginx 实现后端服务健康检测的两种方式是什么？简述其工作原理。

- Nginx 通过主动和被动两种方式对参与负载均衡的各网站服务器节点进行健康检查。
- 被动模式：Nginx 默认基于对代理请求的响应状态来判断后端健康，如连续返回特定次数的连接失败、超时或 5xx 错误代码，则将该节点标记为“不健康”并临时停止向其转发流量。
- 主动模式：Nginx 会周期性地探测各网站服务器节点，同时对探测结果进行标记。主动模式是 NGINX Plus 版本的独有功能，NGINX Open Source 版本仅支持被动检查模式。
- 配置关键：主动检测需在 `upstream` 块中定义 `health_check` 指令及参数（如检测间隔、通过条件等），而被动检测通常通过 `max_fails` 和 `fail_timeout` 参数进行调节。
- 工作原理区别：被动检测依赖真实业务流量的结果，存在一定滞后性和风险；主动检测则通过独立的探测请求提前发现故障，确保流量只分发给健康节点，可靠性更高。

42.为提升 Nginx 的安全性，可采取哪些措施？请列举五项。

- **资源层防护（防 DDoS 攻击）**：防 DDoS 攻击可避免服务器资源被大量无用请求占用，有效地提升网站服务的稳定性。通过 ngx_http_limit_req _module 和 ngx_http_limit_conn_module 模块分别限制每秒请求数、单个即的并发请求数以实现防 DDoS 攻击。
- **应用层防护（防 SQL 注入）**：防 SOL 注入可有效地保证网站服务的数据库管理系统的安全。可通过筛选客户编求将其重定向至 404 页面，从而实现防 SQL 注入。
- **网络层防护（访问控制）**：设置访问范围可有效地阻隔恶意主机的访问，极大地提升网站的安全性。可通过 ngx_htp_access_module 模块实现代理服务器的访问范围限制。
- **启用 HTTPS 并强化 TLS/SSL 配置**：使用 SSL 证书对传输数据进行加密，并禁用不安全的协议（如 SSLv2、SSLv3）和弱密码套件，配置安全的加密算法、启用 HSTS（HTTP Strict Transport Security）头部，防止中间人攻击和数据窃听。
- **隐藏服务器指纹并安全配置响应头**：在配置中设置 server_tokens off; 以隐藏 Nginx 版本信息，防止攻击者利用特定版本漏洞。

43.列举五种常见的关系型数据库管理系统，并简要说明各自的优缺点。

- **MySQL Server**
  - **优点**：性能卓越服务稳定，很少出现异常宕机；体积小、易于维护、安装及维护成本低；支持多种操作系统提供多种 API 接口
  - **缺点**：不易扩展；部分开源。
- **MariaDB**
  - **优点**：遵循 GPL 协议完全开源，支持处理更多的并发连接和查询，支持 XtraDB、Aria、MyRocks 等存储引擎
  - **缺点**：由于 IDX 日志文件会实时占用资源；相对删除较慢；集群版本不稳定。
- **Oracle SQL**
  - **优点**：可移植性好，能在所有主流平台上运行；安全性高，获得最高认证级别的 ISO 标准认证；性能最高，保持着开放平台下 TPCD 和 TPC-C 世界纪录；支持多种工业标准，支持 ODBC、JDBC、OCI 等连接；完全向下兼容
  - **缺点**：对硬件要求高；操作复杂，管理维护麻烦。
- **PostgreSQL**
  - **优点**：遵循 BSD 协议完全开源；源码清晰易读，易于二次开发；支持丰富数据类型；多进程并发处理速度快；查询优化器强大。
  - **缺点**：在简单而繁重的读取操作上性能较低；缺乏报告和审计工具。
- **SQL Server**
  - **优点**：与 Windows 操作系统兼容性极好；事务处理功能强，保证数据完整性；支持多种处理器结构、存储过程，具有自主 SQL 语言；文档和社区资源丰富。
  - **缺点**：仅支持 Windows 操作系统。
- **openGauss**
  - **优点**：内核源自 PostgreSQL；兼容 ARM、x86 架构；支持多核并发控制、NUMA-Aware 存储引擎、SQL-Bypass 智能执行等技术；支持 RTO＜10 秒的快速故障倒换与全链路数据保护；提供智能参数调优、慢 SQL 诊断等自治运维能力。
  - **缺点**：国产开源数据库，生态与管理工具仍在发展中，部分企业环境适配可能需更多优化。

44.列举五种常见的非关系型数据库管理系统，并说明其典型应用场景。

- 非关系型数据库包括键值数据库、列族数据库、文档数据库和图形数据库。
- **Redis**（键值数据库）
  - **典型应用场景**：高频读写场景，如会话缓存、消息队列、实时排行榜、热点数据缓存、分布式锁等。
- **MongoDB**（文档数据库）
  - **典型应用场景**：Web 应用、面向文档或半结构化的数据。
- **HBase**（列族数据库）
  - **典型应用场景**：大规模分布式数据存储，如历史记录查询、时序数据存储（如物联网传感器数据）、日志分析、大数据平台的海量数据持久化存储。
- **Cassandra**（列族数据库）
  - **典型应用场景**：高可用、可扩展的分布式数据存储，尤其适用于跨地域部署、写密集型应用，如日志记录、消息系统、推荐系统数据存储等。
- **Neo4j**（图数据库）
  - **典型应用场景**：社交网络关系分析、实时推荐引擎、欺诈检测系统、知识图谱构建、网络与 IT 基础设施管理等需要高效处理复杂关联关系的场景。

45.什么是数据库集群？使用数据库集群有哪些主要优势？请列举三项。

- 数据库集群即利用两台或者多台数据库服务器，构成一个虚拟单一数据库逻辑映像，像单个数据库系统那样，提供透明的数据服务。

优势：

- **高可用性：**数据库集群可以实现在主服务器上完成所有写入和更新操作，在一个或多个从服务器上完成读操作，以提高性能。
- **负载均衡：**在数据库主节点发生故障时，从节点能够自动接管主数据库，从而保证业务不中断和数据的完整性。
- **备份协助：**数据库备份可能会对数据库服务器产生重大影响，从服务器运行备份能够很好的规避该问题，关闭或锁定从属服务器执行备份并不会影响到主服务器。

- **系统扩展性提升**：通过增加节点数量来线性扩展系统的处理能力，支持横向扩展，满足业务增长需求。

46.简述数据库主从复制模式的工作原理及其优势。

主从模式的工作原理：

- 主数据库开启二进制日志记录，将所有操作作为 binlog 事件写入二进制日志中。
- 从数据库读取主数据库的二进制日志并存储到本地的中继日志(relay log)，然后通过中继日志重现主数据库的操作，从而保持数据的一致性。

优势：

- **读写分离**：写操作走主节点，读操作分散至从节点，显著提升系统并发能力与吞吐量。
- **高可用基础**：从节点作为实时热备，故障时可快速切换为主节点，缩短业务中断时间。
- **异地容灾**：从节点可跨地域部署，实现就近读取与异地数据备份，保障业务连续性。

47.非关系型数据库具有哪些通用特性？请列举四项并简要说明。

- 非关系型数据库是相对于关系型数据库来讲的，不遵循二维数据模型。针对应用不同，其选用的数据模型也不同。

通用特性：

- **高性能：**通过简化的数据模型（如键值、文档）、内存优先架构及弱化强一致性约束，在**海量数据读写**与**高并发访问**场景下，相比传统关系型数据库可提供更优的吞吐量与更低延迟。
- **分布式架构：**设计上原生支持**多节点集群部署**，能够在普通商用服务器上运行，通过数据分片、多副本复制与自动故障转移等机制，实现**大规模数据存储**与**高可用性**服务。
- **易于水平扩展：**主要采用**横向扩展（Scale-out）**方式，通过向集群中动态添加节点即可灵活提升存储容量与处理能力，扩展过程通常平滑、成本可控，且具有良好的线性扩展潜力。
- **灵活模式与弱事务性：**采用**无模式或动态模式**的数据结构，可灵活适应数据模型的变化；多数 NoSQL 数据库**不支持多行跨节点的 ACID 事务**（尤其早期产品），常以最终一致性为默认保证，以此换取更高的并发性能与扩展能力。

48.列举 MySQL Server 的五项主要特性。

- 基于 C 和 C++ 语言编写，可移植性强。
- 支持广泛的平台部署，如 Windows、Linux、Mac OS 等：
- 支持多线程、存储过程。
- 提供事务和非事务性存储引擎。
- 支持多种数据类型。
- 提供 C、C++、Java、Perl、PHP、Python、Ruby 等编程语言的 API，支持 ODBC、JDBC 等连接。
- 支持灵活的权限和密码验证，并支持基于主机的验证。
- 支持大型数据库。
- 提供 mysqladmin、mysqlcheck、mysqidump、mysqlimport 等实用工具

49.MySQL Workbench 提供哪些主要功能？请列举五项并说明其具体用途。

- 设计：MySOL Workbench 使 DBA(教据库管理员)、开发人员或效据架构师能够直接地设计。建模、创建和管理数据库。包括创建 ER 模型、进行正向和反向工程等功能。
- 开发：MySOL Workbench 提供了用于创建、执行和优化 SOL 查询的可视化工具。
- 管理：MySOL Workbench 提供了一个可视化平台，可以轻松管理 MySOL，可以使用可视化工具来配置服务器、管理用户、执行备份和恢复及查看数据库运行状况。
- 仪表盘：MySOLWorkbench 提供了一套仪表盘，可通过性能仪表板查看关键性能指标，通过性能报告查看 JO 瓶颈、慢查询 SQL 语句等。
- 数据库迁移：MySOL Workbench 可将 Microsoft SOL Server、Microsoft Access、Sybase ASE、PostreSQL 或其他数据库数据迁移到 MySOL。

50.列举 MongoDB 数据库的五项主要特性。

- MongoDB 支持多文档事务、连接查询，是较为接近关系型数据库的非关系型数据库。

MongoDB 的主要特性如下：

- **灵活的文档数据模型**：采用 BSON（二进制 JSON）格式存储数据，支持嵌套文档和数组结构，无需预定义表结构，能够灵活适应业务数据模型的变化。
- **强大的查询语言和索引支持**：提供丰富的查询操作符，支持复杂查询、聚合管道、全文搜索等功能，支持多种索引类型（单字段、复合、地理空间、文本索引等）。

- **多语言 API 驱动支持**：提供官方驱动程序支持 Python、Java、Node.js、C++、Go、Ruby 等多种编程语言，便于开发者集成到各种应用系统中。
- **水平扩展能力**：通过分片（Sharding）技术实现数据的水平分布，支持动态添加分片节点，能够处理大规模数据存储和高并发访问需求。
- **高可用性和故障恢复**：支持副本集（Replica Set）架构，提供数据多副本存储、自动故障转移、读写分离等功能，确保系统的高可用性和数据安全性。

51.MongoDB Compass 工具的主要功能有哪些？请列举五项并说明其作用。

- 数据管理：创建、查看、修改和删除数据库、数据表、视图、列、索引等
- 索引管理：创建、删除索引
- 数据聚合：创建和执行聚合管道
- 数据导入导出：支持从 SQL、CSV、XML 等格式文件导入数据；支持以 CSV、XML、PDF、SOL 等格式导出数据
- 监控：监控数据库服务器的流量、连接、进程、查询统计、数据库变量状态、主机状态等
- 集群管理：查看集群状态、查看集群成员、添加集群成员
- 统一身份验证：支持 Kerberos、LDAP 和 x.509 身份验证
- 文档模型分析：提供对指定集合中文档的字段和值的分析

52.什么是 MongoDB 副本集？其包含哪三种成员角色？

- 副本集：副本集是一组维护相同数据集的 mongod 实例。一个副本集包含多个数据承载节点和一个仲裁器(Arbitcr，可选)。在数据承载节点中，只有一个成员被当作主节点，其他成员皆为从节点。副本集中的节点数最好为奇数(为了选举顺利进行)，成员个数最少为 3 个，不超过 50 个(最多有 7 个投票成员)。
- MongoDB 副本集中的成员可分为 3 种：主节点(Primary)、从节点(Secondaries)和仲裁器(Arbiter)，每种成员都在副本集上起着不同的作用。
- 主节点：主节点是副本集中唯一能够接收写操作的成员。副本集只能有一个主节点，如果当前的主节点不可用，则通过选举确定新的主节点。
- 从节点：从节点作为主节点数据集的副本，在副本集中起着数据备份、主节点候选人和找均衡的作用，尽管客户端无法通过从节点写入数据，但是客户端可以选择从节点读取数据。

- 仲裁器：该成员没有数据副本，也不会成为主节点，主要用来选举投票，当副本集的节点数据为偶数时，需要添加一个仲裁节点。仲载节点因为没有数据，只参与投票，所以仲裁节点需要的资源很少，但官方不建议将仲裁节点部署在副本集的其他节点上。

53.在哪些情况下 MongoDB 副本集会触发选举？影响选举结果的因素有哪些?

触发副本集选举的情况：

- 向副本集添加新节点中
- 副本集初始化
- 指定主节点为从节点或副本集重新配置
- 主节点响应超时(默认 10s)。

影响选举因素：

- Heartbeats：副本集成员每两秒钟都会向彼此发送一次 Heartbeats(类似 ping)。如果某个成员在 10s 内未响应，则其他成员将其标记为不可访问，该成员将不能成为主节点或被降低优先级
- 优先级：优先级高的成员将优先获取投票权
- 票数：得票数最多的成员将成为主节点

54.什么是 MongoDB 的写关注（Write Concern）？请简述其三种常见模式。

写关注：写关注影响了客户端在向 mongod 实例、副本集写数据时的返回结果即在满足什么条件时返回操作成功，在满足什么条件时返回操作失败。

选项	值类型	描述
w	<number>	值为 0：表示对客户端的写操作不使用写关注 值为 1：默认值，表示数据写入到主节点就向客户端返回操作成功 值为 majority：表示数据写入到副本集大多数成员后向客户端返回操作成功 值大于 1：表示数据写入到 number 个节点才向客户端返回操作成功
j	<boolean>	表示选项 w 中指定的节点将操作记录到操作日志时向客户端返回操作成功
wtimeout	<number>	超过 number 毫秒仍未向客户端返回操作成功，则客户端确认写入失败

55.FTP 协议支持哪两种数据传输方式？简述其工作原理并比较差异。

- FTP 的传输方式：在 Linux/UNIX 系统中，FTP 支持文本(ASCII)和二进制(Binary)两种方式的文件传输，选择合适的传输方式可以有效地避免文件乱码。

（1)在文本传输模式下，其传输方式会进行调整，主要体现为对不同操作系统的回车、换行结束符等进行转译，将其自动文件转译成目的主机的文件格式。

（2)在二进制传输模式下，会严格保存文件的位序，原始文件和复制文件逐位一一对应，该传输方式不对文件做任何修改。

对比分析

- ASCII 模式用于纯文本文件，会自动转换不同系统的换行符（如 Windows 的 \r\n 与 Linux 的 \n），以保证文本在目标系统上正常显示。
- Binary 模式则按原始字节逐位传输，不作任何修改，适用于图片、压缩包、可执行文件等所有非文本文件。
- 若用 ASCII 模式传输二进制文件，会因错误转译导致文件损坏；而用 Binary 模式传输文本文件通常不会出错，因为现代编辑器能识别各种换行格式。

56.比较 FTP、NFS 和 Samba 三种文件共享协议在传输机制、平台兼容性和安全性方面的异同。

对比维度	FTP	NFS	Samba
传输机制	基于 TCP（21/20 端口），仅文件级传输，不支持挂载，无原生文件锁	NFSv3（UDP/TCP）、NFSv4（TCP），支持远程挂载，基于 RPC，有文件锁，适合并发	基于 TCP（445/139 端口），支持挂载，兼容 SMB 事务 / 锁机制，跨系统协同友好
平台兼容性	全平台支持（Windows/Linux/macOS），无系统限制	原生支持类 Unix，Windows 需额外软件，兼容性差	完美兼容 Windows 与 Linux/Unix/macOS，跨系统无缝共享
安全性	原生明文传输，无加密，需 FTPS/SFTP 增强，仅账户密码控制	原生弱（IP 授权），NFSv4 支持 Kerberos，权限依赖 UID/GID，易混乱	支持加密、多认证方式，细粒度 ACL 权限，原生安全性

相同点

- 均为 C/S 架构，依赖 TCP/IP，支持基本文件操作，可额外加固安全。

不同点

- 定位：FTP 侧重文件传输，NFS/Samba 侧重文件系统挂载；
- 场景：FTP 跨平台传输，NFS 类 Unix 高效共享，SambaWindows-Linux 混合共享；
- 安全：Samba 原生最强，FTP/NFS 需额外配置。

57.虚拟化技术主要有哪三种实现方式？请分别说明其特点。

虚拟化的实现方式。根据实现方式的不同，虚拟化技术可以分为全虚拟化、半虚拟化和操作系统级虚拟化等。

全虚拟化：

- 在全虚拟化中，虚拟机(guest，客户机）和硬件之间，安装有“Hypervisor （超级管理器)”。Hypervisor 是一切硬件资源的管理者，并将其虚拟成各种设备，客户机操作系统无须做任何修改，就能直接对虚拟化的硬件发出请求。客户机操作系统内核执行的任何有特权的指令都需要经过 Hypervisor 翻译，才能正确地被处理。
- 全虚拟化是最为安全的一种虚拟化技术，因为客户机操作系统和底层硬件之间已被隔离。客户机操作系统的内核不要求做任何修改，可以在不同底层体系结构之间自由移植客户机操作系统。只要有虚拟化软件，客户机就能在任何体系结构的处理器上运行，但是在翻译 CPU 指令时会有一定的性能损失。

半虚拟化：

- 半虚拟化技术也叫作准虚拟化技术，是在全虚拟化的基础上，对客户机操作系统进行修改，增加一个专门的 API，使用 API 将客户机操作系统发出的指令进行最优化处理，不需要 Hypervisor 耗费一定的资源进行翻译操作，因此 Hypervisor 的工作负担变得非常小，系统整体的性能会有较大提升。
- 半虚拟化技术的缺点是需要修改操作系统以包含 API，不能实现对通用操作系统的支持。

操作系统级虚拟化：

- 操作系统级虚拟化并不是在硬件系统里创建多个虚拟机环境，而是让一个操作系统创建多个彼此相互独立的应用环境，这些应用环境访问同一内核。操作系统级的虚拟化可以想象是内核的一种功能，而不是抽象成一层独立的软件。
- 因为不存在实际的翻译层或者虚拟化层，所以操作系统级的虚拟机开销很小，大多数都能达到原本的性能。该类型不能使用多种操作系统，所有虚拟机需要共享一个内核。

58.openEuler 通过 KVM 实现虚拟化需要哪三个核心组件？简述各组件的功能。

- openEuler 通过 KVM 实现虚拟化需要 KVM 内核模块、QEMU 硬件模拟器、Libvirt 管理工具集这三个核心组件
- KVM 内核模块：KVM 是 Linux 内核的一部分，提供底层虚拟化功能。它将 Linux 内核转变为一个虚拟机监控器（Hypervisor），允许内核直接支持运行多个隔离的虚拟机（VM）。KVM 负责 CPU 和内存的虚拟化，利用 Intel VT-x 或 AMD-V 等硬件辅助虚拟化技术提升性能。
- QEMU：QEMU 是纯软件实现的虚拟化模拟器，几乎可以模拟任何硬件设备。QEMU 可配合 KVM 实现虚拟化，其主要负责 IO 拟化。
- Libvirt：libvint 提供统一、稳定、开放的源代码应用程序接口(AP1)、守护进程(Iibvirtd)和一个默认命令行管理工具(virsh)，可实现对宿主机及其虚拟化设备、网络和存储的管理。
- openEuler 通过 KVM 实现虚拟化的核心逻辑是：KVM 内核模块借助 CPU 硬件虚拟化扩展（Intel VT-x/AMD-V）将 openEuler 内核转化为 Hypervisor，负责虚拟机的 CPU 和内存虚拟化；搭配 QEMU 完成 I/O 设备的模拟，再通过 Libvirt 实现对虚拟机、虚拟网络和存储的统一管理，最终让多个不同操作系统的虚拟机在同一物理机上高效、安全地并行运行。

59.KVM 虚拟机在其生命周期中可能处于哪六种状态？请简要描述。

KVM 虚拟机主要有如下几种状态。

- 未定义(undefined): 虚拟机未定义或未创建，即虚拟机不存在。
- 关闭状态(shutoff): 虚拟机已经被定义但未运行，或者虚拟机被终止：
- 运行中(running): 虚拟机处于运行状态。
- 暂停(paused): 虚拟机运行被挂起，其运行状态被临时保存在内存中，可以恢复到运行状态。
- 保存(saved): 与暂停(paused)状态类似，其运行状态被保存在持久性存储介质中，可以恢复到运行状态。
- 崩溃(crashed): 通常是由于内部错误导致虚拟机崩溃，不可恢复到运行状态。

60.容器技术的主要应用场景有哪些？简述其相较于传统部署的优势。

- **微服务架构**：将单体应用拆分为多个独立、轻量级的微服务，每个服务打包为一个容器，实现独立开发、部署与扩展。
- **持续集成与持续部署**：作为 CI/CD 流程中的标准交付单元，确保应用在开发、测试、生产环境中的**一致性**，实现快速、可靠的自动化发布。
- **DevOps 与平台工程**：统一开发与运维环境，提供可重复、版本化的基础设施，支撑**敏捷开发、自动化运维和自助式平台服务**。
- **弹性伸缩与云原生**：与编排系统（如 Kubernetes）结合，可根据负载自动伸缩容器实例，是构建云原生应用的基础。
- **混合云与多环境部署**：封装应用及其依赖，实现在物理机、虚拟机、私有云和公有云之间的**无缝迁移与统一管理**。
- **快速实验与隔离测试**：为短期任务、一次性作业（如批处理、数据转换）或隔离的测试环境提供**轻量、即启即用的沙箱**。

相较于传统部署的优势：

- **轻量高效**：共享宿主机内核，资源占用极少，实现秒级启动与更高部署密度，显著提升硬件利用率。
- **环境一致**：通过镜像打包应用及其全部依赖，确保开发、测试、生产环境完全一致，彻底消除“在我机器上能运行”的问题。
- **敏捷交付**：支持基于镜像的快速构建、一键部署、滚动更新和即时回滚，极大提升发布速度和运维自动化水平。
- **可移植性强**：实现“一次构建，处处运行”，使应用能在任意支持容器的平台上无缝迁移与扩展，真正实现跨云和混合云部署。

61.从隔离性、启动速度、资源开销、镜像管理和适用场景五个方面，对比虚拟化与容器技术的异同。

相同点：虚拟化和容器都是虚拟化技术，都能动态分配资源提高利用率，实现环境隔离和标准化部署。

不同点：

- **隔离性**：传统虚拟化通过 Hypervisor 实现**操作系统级的强隔离**，安全性更高；容器技术则利用内核的 Namespace 和 CGroup 实现**进程级的弱隔离**，共享宿主机操作系统内核。
- **启动速度**：虚拟机需启动完整的客户操作系统，过程为**分钟级**；容器本质是宿主机上的一个进程，启动为**秒级甚至毫秒级**，速度极快。
- **资源开销**：虚拟机包含独立的内核和系统进程，**CPU、内存和存储开销大**；容器共享内核，无额外系统开销，**资源占用极小**，部署密度高。
- **镜像管理**：虚拟机镜像（如 qcow2 文件）体积**庞大（GB 级）**，包含整个操作系统；容器镜像采用分层构建，**轻量（MB 级）**，易于分发、版本控制和持续集成。
- **适用场景**：虚拟化更适合运行**异构操作系统、需强隔离或遗留系统**的工作负载；容器则天生契合**微服务、云原生应用和 DevOps 流程**，追求敏捷与弹性。

62.操作系统运维管理主要包括哪五项内容?

- **系统运行监控**：负责查看服务器硬件信息、操作系统运行状态及业务部署情况，确保系统正常运行。
- **资源管理**：包括 CPU 管理（监控负载、优化资源利用）和 内存管理（监控使用情况、合理调配资源），保障系统高性能运转。
- **磁盘管理**：检查磁盘状态、更换硬件、监控磁盘使用率、IO 读写速率等，确保数据存储效率。
- **网络管理**：监控主机网络流量、规划网络结构、及时发现并解决网络故障。
- **权限与日志管理**：
  - 权限管理：设置用户权限、增删账号、保障系统安全。
  - 日志管理：记录系统日志，便于操作追溯与审查分析。

63.操作系统监控应关注哪五个核心方面？简述系统监控的意义。

五个核心监控方面：

- **CPU 监控**：关注处理器使用率、负载队列及核心利用率，防止计算瓶颈。
- **内存监控**：监控物理内存、交换空间使用率及缓存情况，避免内存耗尽导致系统卡顿或崩溃。
- **存储监控**：检查磁盘空间使用率、I/O 读写速率及健康状态，确保数据存储可靠与访问性能。
- **网络监控**：跟踪网络流量、连接状态、带宽使用及错误包率，保障网络通信的稳定性与安全。
- **进程监控**：监视关键进程的运行状态、资源占用及异常行为，及时处理僵尸进程或资源泄露问题。

系统监控的意义：

通过系统监控，可以实时了解系统的运行状态，快速发现异常（如资源瓶颈、服务中断等），并及时解决，从而保障业务服务的可靠性和稳定性，提升运维效率，避免因系统故障导致业务中断或数据损失。

64.系统监控的两种主要实现方式是什么？分别列举常用的工具软件。

- 系统监控的两种主要实现方式是**命令监控**和**软件监控**。
- **命令监控**
  - **方式描述**：通过对操作系统的命令实现对系统运行情况的监控。
  - **常用工具**：主要包括 `top`（监控进程与 CPU/ 内存）、`netstat`（查看网络连接与性能）、`iostat`（监控 CPU 与磁盘 I/O）等命令行工具。
- **软件监控**
  - **方式描述**：通过部署专业的监控软件，以周期性的方式自动采集、记录和分析系统数据，并提供历史数据查询、可视化图表及告警通知功能。
  - **常用工具**：这类软件通常借助 **SNMP 协议**、**Agent 代理**或**探针**进行数据采集。典型的代表软件包括：**Zabbix**、**Prometheus**（配合 Grafana）、**Nagios** 以及各类云服务商提供的监控服务（如阿里云 CloudMonitor）。

65.列举五种可用于查看 openEuler 系统 CPU 负载的命令行工具。

- **top：**

实时监控系统中各个进程的资源占用状况（包括 CPU、内存、进程状态等），类似 Windows 任务管理器，默认 5 秒刷新一次进程列表

- **mpstat：**

监控系统 CPU 的实时状态和单 CPU 详情，是 sysstat 工具集中的核心组件，支持多核 CPU 分析。

- **vmstat：**

统计系统整体存储情况，包括内核进程、内存使用、虚拟内存、磁盘 IO 和 CPU 状态等信息。

- **htop：**

交互式 CPU 监控工具，支持彩色显示和动态刷新，可按 CPU 使用率、内存占用等排序。

- **nmon：**

能够动态地展示 openEuler 的多项性能，也可手动输入命令单项查看 CPU 性能。

66.使用虚拟内存机制有哪些主要优势？请列举五项。

- **扩展内存空间：**提供比物理内存更大的连续地址空间，缓解物理内存不足。
- **程序隔离**：不同进程的虚拟地址相互独立，单个进程的操作不会影响其他进程，提高了系统稳定性。
- **数据保护：**通过为虚拟内存设置读写属性，保护代码段不被修改、数据段不被执行，增强了系统安全性。
- **内存映射：**可将磁盘文件直接映射到虚拟地址空间，实现按需加载，减少物理内存的长期占用。
- **共享内存：**多个进程可通过映射同一块物理内存到各自的虚拟空间来实现内存共享，提升进程间通信效率。

67.操作系统面临的安全风险主要来自哪三个方面？请分别说明其成因。

操作系统面临的安全风险主要来自**硬件设备**、**交互过程**以及**网络病毒漏洞**三个方面。

- **硬件设备的安全风险**：成因于硬件本身的**物理故障**（如部件损坏）及**运行环境不稳定**（如断电、温湿度异常）。
- **交互过程的安全风险**：成因于系统内部的**管理疏漏**与**人为失误**，例如权限分配混乱、弱口令或服务配置错误。
- **网络病毒漏洞的安全风险**：成因于系统暴露在网络上，**软件存在未修补的漏洞**，从而遭受外部恶意扫描、攻击与入侵。

68.openEuler 内置了哪四种主要安全保护机制？简述其功能。

openEuler 内置了 PAM 机制、安全审计机制、强制访问控制机制以及防火墙机制这四种主要安全保护机制。

- **PAM 机制**
  - **功能**：提供一个统一的框架和编程接口，将应用程序的**认证工作从程序员转移给系统管理员**。管理员可以在不重新编译程序的情况下，灵活选择和更换系统的本地认证方法（如密码、令牌等）。
- **安全审计机制**
  - **功能**：用于**记录和追踪安全事件**。它可以记录入侵者的行为、系统事件详情以及网络连接情况，并将这些信息保存到日志文件中，为事后**审计、追溯和取证**提供关键数据支持。
- **强制访问控制（MAC）机制**
  - **功能**：由系统管理员**从全局角度强制实施**的访问控制策略。它通过对系统中的主体（如用户、进程）和客体（如文件、资源）进行安全标记，**严格限制信息的流动与共享**，确保用户只能访问被明确授权的资源，从而有效防止信息泄露和权限滥用。
- **防火墙机制**
  - **功能**：通过预设的**控制策略、行为审计和抗攻击**等功能，对进出服务器的网络流量进行监控、过滤和阻挡，构成一道网络边界防线，以**保障服务器自身免受外部网络攻击**。

69.列举可以查看 openEuler 的 CPU 负载的五种工具。

- **top：**

实时监控系统中各个进程的资源占用状况（包括 CPU、内存、进程状态等），类似 Windows 任务管理器，默认 5 秒刷新一次进程列表

- **mpstat：**

监控系统 CPU 的实时状态和单 CPU 详情，是 sysstat 工具集中的核心组件，支持多核 CPU 分析。

- **vmstat：**

统计系统整体存储情况，包括内核进程、内存使用、虚拟内存、磁盘 IO 和 CPU 状态等信息。

- **htop：**

交互式 CPU 监控工具，支持彩色显示和动态刷新，可按 CPU 使用率、内存占用等排序。

- **nmon：**

能够动态地展示 openEuler 的多项性能，也可手动输入命令单项查看 CPU 性能。

70.列举操作系统运维管理的五项主要内容。

- **系统运行监控**：负责查看服务器硬件信息、操作系统运行状态及业务部署情况，确保系统正常运行。
- **资源管理**：包括 CPU 管理（监控负载、优化资源利用）和 内存管理（监控使用情况、合理调配资源），保障系统高性能运转。
- **磁盘管理**：检查磁盘状态、更换硬件、监控磁盘使用率、IO 读写速率等，确保数据存储效率。
- **网络管理**：监控主机网络流量、规划网络结构、及时发现并解决网络故障。
- **权限与日志管理**：
  - 权限管理：设置用户权限、增删账号、保障系统安全。
  - 日志管理：记录系统日志，便于操作追溯与审查分析。

71.列举操作系统监控的五项主要内容并阐述监控的意义。

五个核心监控方面：

- **CPU 监控**：关注处理器使用率、负载队列及核心利用率，防止计算瓶颈。
- **内存监控**：监控物理内存、交换空间使用率及缓存情况，避免内存耗尽导致系统卡顿或崩溃。
- **存储监控**：检查磁盘空间使用率、I/O 读写速率及健康状态，确保数据存储可靠与访问性能。

- **网络监控**：跟踪网络流量、连接状态、带宽使用及错误包率，保障网络通信的稳定性与安全。
- **进程监控**：监视关键进程的运行状态、资源占用及异常行为，及时处理僵尸进程或资源泄露问题。

系统监控的意义：

通过系统监控，可以实时了解系统的运行状态，快速发现异常（如资源瓶颈、服务中断等），并及时解决，从而保障业务服务的可靠性和稳定性，提升运维效率，避免因系统故障导致业务中断或数据损失。

72.列举系统监控的两种实现方式及其常用的工具软件。

- **命令监控**
  - **方式描述**：通过对操作系统的命令实现对系统运行情况的监控。
  - **常用工具**：主要包括 `top`（监控进程与 CPU/ 内存）、`netstat`（查看网络连接与性能）、`iostat`（监控 CPU 与磁盘 I/O）等命令行工具。
- **软件监控**
  - **方式描述**：通过部署专业的监控软件，以周期性的方式自动采集、记录和分析系统数据，并提供历史数据查询、可视化图表及告警通知功能。
  - **常用工具**：这类软件通常借助 **SNMP 协议**、**Agent 代理**或**探针**进行数据采集。典型的代表软件包括：**Zabbix**、**Prometheus**（配合 Grafana）、**Nagios** 以及各类云服务商提供的监控服务（如阿里云 CloudMonitor）。

  两种方式相辅相成，分别适用于**快速诊断**与**长期运维管理**。

73.列出五种常见的关系型数据库管理系统，并说明其主要优缺点。

- **MySQL Server**
  - **优点**：性能卓越、服务稳定（很少异常宕机）；体积小、易于维护且安装维护成本低；支持多种操作系统并提供多种 API 接口。
  - **缺点**：（图片中描述不完整，据常见短板补充）早期版本对复杂事务、存储过程等高级功能支持较弱，且部分存储引擎不支持完整的事务特性。
- **Oracle SQL**

- **优点**：可移植性好，可在所有主流平台运行；安全性高（获最高级别 ISO 认证）；性能顶尖（保持 TPC-C/-D 世界纪录）；支持多种工业标准（ODBC、JDBC 等）并完全向下兼容。
- **缺点**：对硬件要求高；操作复杂，管理维护成本高。

- **PostgreSQL**
  - **优点**：遵循 BSD 协议完全开源，代码清晰易读、易于二次开发；支持丰富的数据类型；采用多进程架构，并发处理速度快；具备强大的查询优化器，擅长处理复杂查询。
  - **缺点**：对于简单的繁重读取操作，其性能可能较低；且相对缺乏成熟的商业报告和审计工具。
- **SQL Server**
  - **优点**：在 Windows 操作系统上最易于使用；具备强壮的事务处理功能，能通过多种方法保证数据完整性；支持存储过程并有自主的 SQL 语言；拥有丰富的文档和社区帮助。
  - **缺点**：主要支持 Windows 操作系统（跨平台能力受限）。
- **MariaDB**
  - **优点**：遵循 GPL 协议完全开源；支持处理更多的并发连接和查询；支持 XtraDB、Aria、MyRocks 等多种存储引擎。
  - **缺点**：（图片中描述不完整，据常见评价补充）作为 MySQL 的分支，某些新特性可能不及 MySQL 更新及时；在极高负载的复杂集群场景下，其稳定性可能面临考验。

74.列出五种常见的非关系型数据库管理系统，并说明其主要应用场景。

- **Redis**（键值数据库）
  - **应用场景**：主要用于**内容缓存**和**频繁访问**的数据存储，例如会话缓存、排行榜、消息队列等需要高速读写的场景。
- **HBase**（列族数据库）
  - **应用场景**：适用于**分布式大数据存储与管理**，特别适合需要存储海量稀疏数据、进行随机实时读写的场景，如日志系统、历史数据查询。
- **Cassandra**（列族数据库）
  - **应用场景**：同样面向**分布式大数据**环境，尤其适合需要高可用性、可扩展性和跨地域部署的应用，如物联网数据、时间序列数据存储。
- **MongoDB**（文档数据库）

    - **应用场景**：广泛用于 **Web 应用**和内容管理系统，存储**面向文档或半结构化**的数据，如用户配置、博客文章、产品目录等 JSON/BSON 格式数据。
- **Neo4j**（图形数据库）
    - **应用场景**：专为处理**复杂关系**而设计，用于**社交网络分析、推荐引擎、知识图谱、网络拓扑**等需要高效遍历实体间关联的场景。

75.什么是数据库集群？列出使用数据库集群的三个优势。

数据库集群是指**利用两台或多台数据库服务器，构成一个虚拟的、逻辑上统一的数据库映像**。它对外表现得像单个数据库系统一样，为用户和应用程序提供透明的数据服务。

使用数据库集群的三个优势：

- **高可用性**
    - 通过在多台服务器之间分配写入和更新操作，即使部分服务器出现故障，集群仍能继续提供服务，从而显著提高系统的整体可用性和业务连续性。
- **故障转移与数据完整性**
    - 当主节点发生故障时，从节点能够**自动接管**主数据库的工作，最小化服务中断时间，并确保数据的完整性和一致性不受破坏。
- **备份协同与运维隔离**
    - 数据库备份操作通常会消耗大量资源，影响在线服务性能。在集群中，可以在**从服务器上执行备份任务**，从而完全避免对主服务器的性能干扰，实现运维操作与生产服务的隔离。

76.阐述数据库主从模式的工作原理及优势。

工作原理

主从模式的核心是**基于二进制日志的数据复制**。主数据库开启二进制日志功能，将所有数据更改操作记录为日志事件。从数据库则读取主数据库的二进制日志，将其存储为本地的中继日志，并通过重放中继日志中的事件来重现主数据库的操作，从而确保从数据库与主数据库的数据最终一致。

三个主要优势

- **实现高可用性**：通过在**一系列服务器上**共同处理写入和更新操作，提高了系统的整体处理能力和服务可用性。

- **保障服务连续性**：当主节点发生故障时，从节点能够**自动接管**主数据库的工作，从而最大限度地减少服务中断时间，并保证数据的完整性。
- **优化备份操作**：由于数据库备份对服务器资源消耗大，在从服务器上执行备份任务可以完全规避对主服务器性能的影响，实现运维与生产的隔离。

77.什么是非关系型数据库，阐述其通用的四个特性。

非关系型数据库是一种不遵循传统关系模型（无固定表结构）的数据库。它没有统一的架构，不同类别之间的差异远大于各类关系型数据库之间的差异。其设计核心是**针对特定应用场景和存储数据类型进行高度优化**，例如为时间序列、文档或图形关系等场景提供专门的存储和查询模型。

非关系型数据库的四个通用特性：

1. **存储的伸缩性更强**：数据存储模型更贴近底层的物理存储方式，针对读写效率进行优化，而非强制遵循关系代数的逻辑结构。
2. **数据模型的开发性能更强**：其灵活的数据模型（如键值、文档、列族）能更好地匹配现代应用（特别是敏捷开发和迭代快速）的数据结构需求，提升开发效率。
3. **更容易通过多节点部署提高可用性**：天生支持分布式架构，能够方便地通过横向扩展（增加节点）来提升系统整体的可用性、容错性和扩展性。
4. **数据模型更加灵活**：通常采用无模式或动态模式，允许在同一个数据集合中存储结构不一致的记录，适应数据结构频繁变化或半结构化 / 非结构化的数据存储需求。

78.列出 MySQL Server 的五种主要特性。

- **基于 C/C++ 编写，可移植性强**：MySQL 使用 C 和 C++ 语言开发，具有良好的跨平台移植能力。
- **支持广泛的平台部署**：可以在多种操作系统上运行，包括 Windows、Linux 和 Mac OS 等。
- **支持多线程与存储过程**：采用多线程架构以支持高并发，并提供了存储过程功能，允许在服务器端执行复杂的业务逻辑。
- **提供事务性和非事务性存储引擎**：支持多种存储引擎（如 InnoDB、MyISAM），用户可根据应用需要在支持 ACID 事务的引擎与追求更高性能的非事务引擎之间进行选择。
- **支持多种数据类型及编程语言 API**：内置丰富的数据类型，并提供了面向多种编程语言（如 C、C++、Java、Perl、PHP、Python、Ruby）的 API，同时支持 ODBC、JDBC 等标准数据库连接方式。

79.列出 MySQL WorkBench 的五个功能并说明其具体作用。

- **设计**
  - **作用**：为数据库管理员（DBA）、开发人员或数据架构师提供**可视化工具**，用于直接进行数据库的设计、建模、创建和管理。具体包括绘制**实体关系（ER）模型**，以及执行**正向工程**（从模型生成数据库）和**反向工程**（从现有数据库生成模型）。
- **开发**
  - **作用**：提供一套**可视化工具**，用于**编写、执行和优化 SQL 查询**。这使得开发者可以更直观、高效地进行数据库操作和 SQL 调试。
- **管理**
  - **作用**：提供一个**图形化界面**来简化 MySQL 服务器的日常管理。用户可以通过它轻松完成**服务器配置、用户账户管理、执行数据备份与恢复**，以及**监控数据库的整体运行健康状况**。
- **仪表盘**
  - **作用**：集成了一套**监控仪表板**，用于展示数据库的关键性能指标。通过性能仪表板，用户可以快速查看系统状态，并通过性能报告来诊断 **I/O 瓶颈、发现慢查询语句**等。
- **数据库迁移**
  - **作用**：内置**数据迁移工具**，支持将其他数据库系统（如 Microsoft SQL Server, Microsoft Access, Sybase ASE, PostreSQL 等）中的数据**迁移和转换到 MySQL** 数据库中，简化了数据库迁移的工作流程。

80.列出 MongoDB 数据库五个主要特性。

- **高性能**
  - MongoDB 通过**内存映射文件**、**丰富的索引支持**（可在任意字段上建立索引）以及**数据内嵌**（减少关联查询）等机制，在数据读写，尤其是海量数据查询上，提供了出色的性能表现。
- **易扩展**
  - 这是其核心设计目标。通过**分片**技术，可以轻松地将数据水平切分并分布到多个服务器集群中，从而实现存储容量和处理能力的**线性、无缝扩展**，以应对业务的快速增长。
- **分布式架构**
  - MongoDB 原生就是为分布式环境设计的。其**分片**和**副本集**机制共同构成了分布式系统的基础，确保数据可以安全、可靠地分布在多台机器上，并实现负载均衡。

- **灵活的数据模型（文档模型）**
  - 这是 MongoDB 的立身之本。它使用 **BSON**（类似 JSON）格式存储数据，采用无预定义结构的文档模型。这种模型允许字段动态变化、支持嵌套和数组，能够自然地映射现代应用中的对象，极大提升了开发迭代速度。
- **对事务支持的权衡（早期为“不支持事务”）**
  - 在早期版本中，MongoDB **只保证单个文档内的原子性**，不支持多文档的复杂事务。
  - **重要更新**：自 **4.0 版本**起，MongoDB 已支持**多文档 ACID 事务**。

81.列出 MongoDB Compass 的五个功能并说明其具体作用。

- 数据管理：创建、查看、修改和删除数据库、数据表、视图、列、索引等
- 索引管理：创建、删除索引
- 数据聚合：创建和执行聚合管道
- 数据导入导出：支持从 SOL、CSV、XML 等格式文件导入数据
- 支持以 CSV、XML、PDF、SQL 等格式导出数据
- 监控：监控数据库服务器的流量、连接、进程、查询统计、数据库变量状态、主机状态等
- 集群管理：查看集群状态、查看集群成员、添加集群成员
- 统一身份验证：支持 Kerberos、LDAP 和 x.509 身份验证
- 文档模型分析：提供对指定集合中文档的字段和值的分析

82.什么是 MongoDB 副本集？副本集的三种成员是什么？

- 副本集：副本集是一组维护相同数据集的 mongod 实例。一个副本集包含多货据承载节点和一个仲裁器(Arbiter，可选)。在数据承载节点中，只有一个成员被当作主节点，共他成员皆为从节点。副本集中的节点数最好为奇数（为了选举顺利进行)，成员个数最少为了个，不超过 50 个（最多有 7 个投票成员)。
- 副本集中的成员 :MongoDB 副本集中的成员可分为了种：主节点(Primary)、从节点(Secondaries)和仲裁器(Arbiter)，每种成员都在副本集上起着不同的作用。

1)主节点。主节点是副本集中唯一能够接收写操作的成员。副本集只能有一个主节点，如果当前的主节点不可用，则通过选举确定新的主节点。

2)从节点。从节点作为主节点数据集的副本，在副本集中起着数据备份、主节点候选人和负载均衡的作用。尽管客户端无法通过从节点写入数据，但是客户端可以选择从节点读取数据。

3)仲裁器。该成员没有数据副本，也不会成为主节点，主要用来选举投票。当制本集的节点数据为偶数时，需要添加一个仲裁节点。仲裁节点因为没有数据，只参与投票，所以仲载节点需要的资源很少，但官方不建议将仲裁节点部署在副本集的其他节点上。

83.MongoDB 副本集在哪些情况下进行选举？影响选举的因素有哪些?

选举：副本集通过选举来决定哪个节点为主节点。

1)以下事件可以触发副本集选举。

- 向副本集添加新节点
- 副本集初始化
- 指定主节点为从节点或副本集重新配置
- 主节点响应超时（默认 10s)

2)以下因素影响选举。

- Heartbeats，副本集成员每两秒钟都会向彼此发送一次 Heartbeats（类似 ping)，如果某个成员在 10s 内未响应，则其他成员将其标记为不可访问，该成员将不能成为主节点或被降低优先级
- 优先级。优先级高的成员将优先获取投票权
- 票数。得票数最多的成员将成为主节点

84.什么是 MongoDB 写关注（Write Concern）？并阐述写关注的三种模式。

- 什么是写关注：写关注影响了客户端在向 mongod 实例、副本集写数据时的返回结果，即在满足什么条件时返回操作成功，在满足什么条件时返回操作失败。
- MongoDB 写关注模式：

选项	值类型	描述
w	<number>	值为 0：表示对客户端的写操作不使用写关注 值为 1：默认值，表示数据写入到主节点就向客户端返回操作成功 值为 majority：表示数据写入到副本集大多数成员后向客户端返回操作成功 值大于 1：表示数据写入到 number 个节点才向客户端返回操作成功
j	<boolean>	表示选项 w 中指定的节点将操作记录到操作日志时向客户端返回操作成功
wtimeout	<number>	超过 number 毫秒仍未向客户端返回操作成功，则客户端确认写入失败

85.FTP 文件服务的传输方式有哪两种，阐述其原理并对比分析。

FTP 的传输方式。在 Linux/UNIX 系统中，FTP 支持文本(ASCII)和二进制(Binary)两种方式的文件传输。选择合适的传输方式可以有效地避免文件乱码。

1)在文本传输模式下，其传输方式会进行调整，主要体现为对不同操作系统的回车、换行结束符等进行转译，将其自动文件转译成目的主机的文件格式。

2)在二进制传输模式下，会严格保存文件的位序，原始文件和复制文件逐位一一对应，该传输方式不对文件做任何修改

对比分析

- ASCII 模式用于纯文本文件，会自动转换不同系统的换行符（如 Windows 的 \r\n 与 Linux 的 \n），以保证文本在目标系统上正常显示。
- Binary 模式则按原始字节逐位传输，不作任何修改，适用于图片、压缩包、可执行文件等所有非文本文件。
- 若用 ASCII 模式传输二进制文件，会因错误转译导致文件损坏；而用 Binary 模式传输文本文件通常不会出错，因为现代编辑器能识别各种换行格式。

86.FTP 文件服务的工作模式有哪两种，阐述其原理并对比分析。

- 1) FTP 有 Standard 和 Passive 两种工作模式。

a. Standand 模式，即主动模式 .FrP 客户端首先与 FTP 服务器的 TCP21 端口创建连接，客户端通过该通道发送用户名和密码进行登录，登录成功后要展示文件清单列表或该取数据时，客户

端随机开放一个临时端口（也称为自由端口，端口号在 1024~-65535 之间），发送 PORT 命令到 p 服务器，“告证”服务器客户端采用主动模式并开放端口。FTP 服务器收到 PORT 主动模式命今方端口号后，服务器的 TCP20 端口和客户端开放的端口连接，在主动模式下，FTP 服务器和客户说必须创建一个新的连接进行数据传输。

b. Passive 模式，即被动模式，FTP 客户端连接到 FTP 服务器的 TCP21 端口，发送用户名和密码进行登录，登录成功后要展示文件清单列表或者读取数据时，发送 PASV 命令到 EP 服务器，服务器在本地随机开放一个临时端口，然后把开放的端口告诉客户端后，客户端连接到服务器开放的端口进行数据传输，在被动模式下，不再需要创建一个新的 FIP 服务器和客户端的连接。

- 2)主动模式和被动模式的区别可概述为两个方面。

a 主动模式传输数据是服务器连接到客户端的端口，被动模式传输数据是客户端连接到服务器的端口。

b. 主动模式需要客户端必须开放端口给服务器，很多客户端都是在防火墙内，开放端口给 FTP 服务器访问比较困难，被动模式只需要服务器端开放端口给客户端连接即可。

87.比较 FTP、NFS、Samba 三种文件服务协议的特性与差异。

- FTP 采用 C/S 结构，服务端常用 vsftpd，客户端常用 FileZilla Client。
- NFS（Network File System）由 Sun 公司于 1985 年推出，允许不同操作系统通过 TCP/IP 网络透明地共享文件和目录。NFS 依赖 RPC（远程过程调用协议）协调通信，RPC 通过固定端口 111 告知客户端 NFS 各功能的实际端口号，因此必须先启动 RPC 再启动 NFS。
- Samba 是 GPL 开源软件，基于 SMB 协议实现 Linux/UNIX 与 Windows 之间的资源共享，默认使用 TCP 139 或 445 端口。Samba 由 smbd（管理共享资源）和 nmbd（提供 NetBIOS 名称解析与浏览服务）两个核心进程组成。
- 三者中，FTP 适合文件上传下载，NFS 适用于类 UNIX 系统间高效挂载共享，Samba 则专用于跨平台（尤其是 Windows 与 Linux）文件共享。

88.阐述虚拟化技术的工作原理。

虚拟化是指通过软件技术对物理硬件资源进行抽象，将一台物理计算机“分割”为多台逻辑上的虚拟计算机，使多个操作系统和应用程序能够同时运行在同一硬件平台上，并且彼此隔离、互不影响。

- **虚拟化的两种主要架构类型：**I 型（裸机型）：虚拟机直接运行在系统硬件上，创建硬件全仿真实例，Hypervisor 直接管理调用硬件资源，不需要底层操作系统，如 VMware ESXi、Xen、KVM 等。II 型（宿主型）：虚拟机运行在传统操作系统上，作为应用程序运行，依赖宿主系统进行硬件管理，如 VirtualBox、VMware Workstation 等。
- VirtualBox 是一种常见的桌面级虚拟化软件，属于宿主型（Type 2）虚拟机监控器，它运行在宿主操作系统之上，负责创建和管理虚拟机，并将虚拟机对硬件的访问请求转交给宿主系统处理。
- 虚拟化通过多种网络模式实现虚拟机的网络通信，如 NAT、桥接和仅主机模式等，不同模式决定了虚拟机与宿主机、局域网及外部网络之间的连接方式，从而满足隔离测试或对外服务等不同应用场景。
- 虚拟化中常见术语包括宿主机（Host）、虚拟机（VM）、客户机操作系统（Guest OS）以及虚拟硬件等，这些术语共同描述了虚拟环境中各组件的角色与关系。
- 通过虚拟机监控器对硬件资源的统一管理、对指令和资源访问的控制，以及网络和存储等虚拟设备的支持，虚拟化技术实现了多操作系统并行运行，提高了资源利用率和系统灵活性。

89.列出虚拟化技术的三种实现方式并阐述其特性。

虚拟化的实现方式。根据实现方式的不同，虚拟化技术可以分为全虚拟化、半虚拟化和操作系统级虚拟化等。

1)全虚拟化。在全虚拟化中，虚拟机(guest，客户机）和硬件之间，安装有“Hypervisor （超级管理器）”。Hypervisor 是一切硬件资源的管理者，并将其虚拟成各种设备，客户机操作系统无须做任何修改，就能直接对虚拟化的硬件发出请求。客户机操作系统内核执行的任何有特权的指令都需要经过 Hypervisor 翻译，才能正确地被处理。

全虚拟化是最为安全的一种虚拟化技术，因为客户机操作系统和底层硬件之间已被隔离。客户机操作系统的内核不要求做任何修改，可以在不同底层体系结构之间自由移植客户机操作系统。只要有虚拟化软件，客户机就能在任何体系结构的处理器上运行，但是在翻译 CPU 指令时会有一定的性能损失。

2)半虚拟化，半虚拟化技术也叫作准虚拟化技术，是在全虚拟化的基础上，对客户机操作系

统进行修改，增加一个专门的 API，使用 API 将客户机操作系统发出的指令进行最优化处理，不需要 Hypervisor 耗费一定的资源进行翻译操作，因此 Hypervisor 的工作负担变得非常小，系统整体的性能会有较大提升。

半虚拟化技术的缺点是需要修改操作系统以包含 API，不能实现对通用操作系统的支持。

3)操作系统级虚拟化。操作系统级虚拟化并不是在硬件系统里创建多个虚拟机环境，而是让一个操作系统创建多个彼此相互独立的应用环境，这些应用环境访问同一内核。操作系统级的虚

拟化可以想象是内核的一种功能，而不是抽象成一层独立的软件。

因为不存在实际的翻译层或者虚拟化层，所以操作系统级的虚拟机开销很小，大多数都能达到原本的性能。该类型不能使用多种操作系统，所有虚拟机需要共享一个内核。

90.列出 openEuler 通过 KVM 实现虚拟化需要的 3 个组件并阐述其功能。

- KVM 内核模块：KVM 是 Linux 内核的一部分，提供底层虚拟化功能。它将 Linux 内核转变为一个虚拟机监控器（Hypervisor），允许内核直接支持运行多个隔离的虚拟机（VM）。KVM 负责 CPU 和内存的虚拟化，利用 Intel VT-x 或 AMD-V 等硬件辅助虚拟化技术提升性能。
- QEMU：QEMU 是一个开源的用户态模拟器和虚拟化工具，在 KVM 架构中负责设备模拟（如磁盘、网卡、显卡等）和 I/O 虚拟化。当与 KVM 结合时，QEMU 利用 KVM 提供的内核接口处理 CPU 和内存操作，自身则专注于模拟硬件设备，从而实现完整虚拟机的运行。
- Libvirt：Libvirt 是一套用于管理虚拟化平台的 API、守护进程（libvirtd）和命令行工具（如 virsh）。它为 KVM/QEMU 提供统一的管理接口，支持虚拟机的创建、启动、停止、迁移、快照等生命周期操作，并可通过 XML 配置文件定义虚拟机资源。Libvirt 还支持多种编程语言绑定和图形化管理工具（如 Virt-Manager）集成。

91.列出 KVM 虚拟机的 6 种状态。

KVM 虚拟机主要有如下几种状态。

- 未定义(undefined)：虚拟机未定义或未创建，即虚拟机不存在。
- 关闭状态(shut off)：虚拟机已经被定义但未运行，或者虚拟机被终止。
- 运行中(running)：虚拟机处于运行状态。
- 暂停(paused)：虚拟机运行被挂起，其运行状态被临时保存在内存中，可以恢复到运行状态。
- 保存(saved)：与暂停(paused)状态类似，其运行状态被保存在持久性存储介质中，可以恢复到运行状态。
- 崩溃(crashed)：通常是由于内部错误导致虚拟机崩溃，不可恢复到运行状态。KVM 虚报机不同状态之间可以相互转化，但必须满足一定规则。

92.容器的工作原理，并阐述其三个主要特点。

容器的工作原理基于操作系统级别的虚拟化技术，主要利用 Linux 内核的命名空间（Namespaces）和控制组（Cgroups） 机制：

- 命名空间（Namespaces） 实现资源隔离，为每个容器提供独立的进程、网络、文件系统、用户、IPC 等视图，使其“以为”自己运行在独立系统中；
- 控制组（Cgroups） 用于限制、记录和隔离进程对 CPU、内存、磁盘 I/O 等系统资源的使用，实现资源配额与优先级控制；
- 容器共享宿主机的操作系统内核，无需像虚拟机那样运行完整的客户操作系统，因此启动快、开销小。

三个特点：

- 容器是轻量级的可执行独立软件包，包含应用程序运行所需的所有内容，如代码、运行环境、系统工具、系统库与设置等。
- 容器适用于基于 Linux 和 Windows 的应用程序，在任何环境中都能够始终如一地运行。
- 容器赋予了应用程序的独立性，使其免受外在环境差异的影响，有助于减少相同基础设施
- 上运行不同应用程序时的冲突。

93.列出 Docker 架构的 3 个组成部分并阐述其关系。

- Docker daemon:Docker 守护进程(dockerd)，用于监听 DockerAPI 请求并管理 Docker 对象，如镜像、容器、网络和存储卷。Docker daemon 还可与其他守护程序通信以管理 Docker 服务。
- Docker client: Docker 客户端(docker), 用户与 Docker 交互的主要方式，当使用诸如 docker run 之类的命令时，Docker client 将发送命令到 Docker daemon，由其执行。Docker 客户端可与多个 Docker daemon 通信。
- Docker registries: Docker 仓库，用于存储 Docker 镜像。Docker Hub(https://hub.docker.com)
- 是 Docker 官方提供的镜像仓库，当使用 docker pull 之类的命令时，Docker 会从 Docker 仓库中花取镜像，当使用 dockerpush 命令时，Docker 会将本地镜像推送至 Docker 仓库中。
- **组件间协作关系：**

Client 作为用户接口，通过 API 调用与 daemon 交互；Daemon 负责实际执行，必要时从 Registry 拉取所需镜像；Registry 提供镜像存储服务，支持跨环境的镜像共享。

94.列出 Docker 支持的 3 种数据存储方式并阐述其特性。

Docker 支持了 3 种方式的数据存储：

- **数据卷(volume)：**由 Docker 管理，是 Docker 宿主机文件系统中的一部分，在 Linux 系统中默认的目录为 varlib/docker/volumes/，非 Docker 进程不应修改此文件系统，数据卷是 Docker 推荐的数据持久化方式。
- **目录挂载(bind mount)：** 可存储数据至 Docker 宿主机的任何地方。Docker 宿主机或者 Docker 容器中的非 Docker 进程可随时修改此文件系统。
- **内存文件系统挂载(tmpfs)：**仅存储在 Docker 宿主机系统的内存中，且永远不会写入 Docker 宿主机的文件系统。
- **存储方式选择指导：**

生产环境数据持久化优先选择数据卷，开发调试和配置文件管理适合使用目录挂载，临时数据处理和敏感信息存储推荐使用 tmpfs。

- **安全性和权限管理：**

数据卷提供细粒度的权限控制和访问管理，目录挂载需要谨慎处理宿主机文件权限，tmpfs 天然提供数据隔离，增强安全性。

95.虚拟化的应用场景及其优势。

主要应用场景：

- 服务器整合与资源优化：将多个物理服务器的工作负载整合到一台物理主机的多个虚拟机中，提高硬件利用率，降低能耗和运维成本。
- 异构操作系统支持：在同一台物理机上同时运行 Windows、Linux、macOS（受限）等不同操作系统，满足多平台开发或测试需求。
- 传统应用迁移与遗留系统维护：无需修改旧有应用即可在新硬件上通过虚拟机继续运行，延长关键业务系统的生命周期。
- 数据中心与云计算基础设施：公有云（如阿里云、AWS）和私有云普遍基于虚拟化技术提供计算实例（如 ECS），实现资源池化和按需分配。
- 安全隔离与沙箱环境：用于恶意软件分析、安全测试等场景，通过虚拟机实现与宿主机的强隔离，防止系统被破坏。

虚拟化的优势：

- 灵活性和可扩展性。用户可根据需求进行动态资源的分配和回收，满足动态变化的业务需求，同时也可根据不同的产品需求，规划不同的虚拟机规格，并可在不改变物理资源配置的情况下调整规模。

- 更高的可用性和更好的运维方式。虚拟化可提供热迁移、快照、热升级、容灾自动恢复等运维手段，可在不影响用户的情况下对物理资源进行删除、升级或变更，可提高业务的连续性，同时实现自动化运维。
- 提高安全性。虚拟化提供操作系统级隔离，同时实现基于硬件提供的处理器操作特权级控制，相比简单的共享机制具有更高的安全性，可实现数据和服务的可控和安全访问。
- 更高的资源利用率。虚拟化可支持实现物理资源和资源池的动态共享，提高资源利用率。

96.容器化的应用场景及其优势。

主要应用场景

- 微服务架构：将单体应用拆分为多个独立服务，每个服务打包为容器，便于开发、部署和弹性伸缩。
- 持续集成与持续交付（CI/CD）：容器提供一致的构建和运行环境，加速自动化测试、构建和发布流程。
- 跨平台部署：开发、测试、生产环境使用相同容器镜像，避免“在我机器上能跑”的问题。
- 云原生应用：在 Kubernetes 等编排平台中，容器是云原生应用的基本单元，支持高可用、自动扩缩容和故障自愈。
- 快速原型与临时环境：可秒级启动隔离环境，适用于演示、教学或短期任务。

核心优势

- 环境一致性：应用及其依赖被打包在镜像中，确保在任何支持容器的平台上行为一致。
- 资源高效：共享宿主机内核，无需虚拟机开销，启动快、占用内存 /CPU 少，密度更高。
- 敏捷开发与运维：支持 DevOps 实践，提升开发、测试、部署效率，缩短交付周期。
- 弹性与可扩展性：配合编排工具可快速扩缩容，适应流量波动，提高系统韧性。
- 标准化与生态成熟：基于 OCI（开放容器标准），拥有 Docker、Kubernetes 等强大开源生态支持。

97.对比分析虚拟化与容器的功能与特性，列出不少于 5 条。

- 架构层级不同：虚拟化（如 KVM、VMware）基于硬件级虚拟化，需在物理机上运行 Hypervisor，再在其上创建完整的客户操作系统；而容器基于操作系统级虚拟化，直接共享宿主机内核，无需独立操作系统。

- 资源开销与启动速度：虚拟机因包含完整 OS，启动慢（通常需数十秒至分钟级），内存和存储占用大；容器仅打包应用及依赖，启动快（毫秒至秒级），资源开销小，密度更高。
- 隔离性强度：虚拟机提供强隔离，各 VM 之间内核完全独立，安全性高；容器通过命名空间（Namespaces）和 Cgroups 实现进程级隔离，若内核存在漏洞，可能存在逃逸风险，隔离性相对较弱。
- 可移植性与兼容性：容器镜像遵循 OCI 标准，可在任何支持容器引擎的 Linux 系统上运行，跨环境一致性极佳；虚拟机虽也可迁移，但因包含完整 OS，镜像体积大，且可能受 Hypervisor 类型限制，可移植性较差。
- 适用场景不同：虚拟化适合运行异构操作系统（如 Windows 与 Linux 共存）、对安全隔离要求高的传统应用；容器更适合微服务、云原生、CI/CD 等敏捷开发场景，强调快速部署、弹性伸缩和高效资源利用。

98.列出操作系统安全风险的 3 个方面，并详细阐述风险原因。

- **硬件设备的安全风险：**外部硬件设备的运行情况是否正常，硬件设备所处的环境是否长期正常稳定，在使用过程中应防止因异常关机或设备零件故障造成操作系统的无法正常使用。
- **具体表现**：异常关机导致文件系统损坏、硬件零件故障引发系统服务中断、环境因素（温度、湿度）影响设备稳定性。
- **交互过程的安全风险：**系统使用过程中，存在用户权限混乱、服务进程异常等安全风险。
- **具体表现**：权限提升攻击、进程注入、非法用户访问、服务配置错误导致的安全漏洞。
- **网络病毒漏洞的安全风险：**当操作系统在网络中提供服务时，将会面临着服务攻击、口令破解攻击、欺骗用户攻击、网络监听攻击、端口扫描攻击等网络安全风险。
- **具体表现**：服务攻击（DDoS）、口令破解攻击、欺骗用户攻击（钓鱼）、网络监听攻击、端口扫描攻击。

99.列出 openEuler 内置的四种安全保护机制，并详细阐述其内容。

openEuler 的安全机制。目前 openEuler 中已经内工多种安全保护机制，具体如下。

- PAM 机制：PAM(Pluggable Authentication Modules)机制是一套共享库，其目的是提供一个框架和一套编程接口，将认证工作由程序员交给管理员。PAM 允许管理员在多种认证方法之间进行选择，它能够在不重新编译与认证相关应用程序的情况下改变本地认证方法。

- 安全审计机制：虽然 openEuler 不能预测何时服务器会遭受攻击，但是可以记录入侵者的行踪，记录事件信息和网络连接情况，信息保存到日志文件中，为后续复查提供支持。
- 强制访问控制机制：强制访问控制(Mandatory Access Control, MAC)是一种由系统管理员从全系统的角度定义和实施的访问控制机制，它通过标记系统中的主客体，强制性地限制信息的共享和流动，使用户只能访问与其相关的、指定范围的信息，防止信息泄密，杜绝访问权限的交叉混乱。
- 防火墙机制：通过防火增的控制策略、行为审计、抗攻击等功能，保障服务器的自身安全。
- (补充)SELinux 机制：SELinux（安全增强型 Linux）是实现强制访问控制的核心技术，它基于域 - 类型模型，为系统内的进程和文件、端口等资源设置安全上下文，严格限定进程的资源访问范围。该机制不受进程所有者身份影响，即便是 root 进程也需遵循策略限制，能有效防范 root 权限被窃取带来的系统性风险，同时支持强制、宽容、禁用三种运行状态，可适配不同业务场景需求。

100.列出 openEuler 安全加固的五项措施。

安全加固方式。在 openEuler 中，提供了两种安全加固的方式，具体如下。

- 手动修改配置或执行命令。可以通过修改 /etc/openEuler_security/security.conf 配置文件进行系统安全加固。
- 使用工具批量修改加固项。openEuler 的安全加固工具 security-tool 以 openEulersecurity service 服务的形式运行。系统首次启动时会自动远行该服务去执行默认加固策略，且自动设置后续开机不启动该服务。

加固内容，在 openEuler 中，最基本的安全加固有 5 个方面，具体如下。

- 系统服务。将系统中运行的服务配置进行调整修改，提高服务配置的安全性。
- 文件权限。通过修改文件和目录的权限和属主提升系统安全性。
- 内核参数。内核参数决定配置和应用特权的状态，可通过参数配置进行提升系统的安全性。
- 授权认证。将通过限制授权系统的操作权限、用户权限等内容提升系统的安全性。
- 账号口令。通过删除所有测试账号、共享账号，设置合理的用户权限策略，制定复杂的用户密码并定期检查等提升系统的安全性。