Introduction to Virtualization

Virtualization is a transformative technology that revolutionizes the way computing resources are utilized and managed. Virtualization involves creating virtual instances of computing components, such as operating systems, storage, or networks, allowing multiple virtual machines to run on a single physical host. Virtualization liberates us from the shackles of hardware limitations, allowing for the efficient utilization of computing resources. Through abstraction, it decouples the software from the underlying hardware, granting unparalleled flexibility and scalability. This liberation extends beyond servers to encompass desktops, storage, and networks, revolutionizing the landscape of IT operations. Virtualization isn’t merely a technological innovation; it’s a transformative force reshaping the very fabric of modern computing. Its ability to transcend physical constraints, optimize resources, enhance flexibility, and bolster efficiency positions it as an indispensable cornerstone of the digital era.

Virtualization Architecture

The virtualization architecture is composed of three fundamental layers: the physical hardware, the hypervisor, and the virtual machines.

Physical Hardware: The physical hardware serves as the unwavering foundati

Hypervisor: Upon this bedrock rests the hypervisor, the maestro of the vir

Virtual Machines: Virtual machines, the stars of the virtualization show,

Types of Virtualization

There are three primary categories of Virtualization.

Hardware Virtualization is a process of creating a virtual machine (VM) th

Software Virtualization: it is a process of creating a virtual environment

Application Virtualization is a process of making an application run indep

Benefits of virtualization

The benefits of virtualization mentioned below have a prominent role in shaping computing

Increased Flexibility: Virtualization makes it easier to move VMs between

Virtualization makes it easier to add new VMs to an existing infrastructur

```
 Reduced Complexity: Virtualization can simplify IT management by reducing
```

Use cases of virtualization

Following are the use cases of virtualization

```
Server consolidation: Virtualization can be used to consolidate multiple p
```

```
Disaster recovery: Virtualization can be used to create a disaster recover
```

```
Testing and development: Virtualization can be used to create isolated tes
```

```
Desktop virtualization: It can be used to provide users with a virtual des
```

```
Application virtualization: Virtualization can be used to package applicat
```

In addition to these specific use cases, virtualization can also be used to improve the overall efficiency and agility of IT organizations. By virtualizing their infrastructure, businesses can make it easier to provision new resources, deploy applications, and manage change. This can lead to significant cost savings and improvements in productivity.

Study material

https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/

https://insights.sei.cmu.edu/blog/virtualization-via-virtual-machines/

https://www.geeksforgeeks.org/difference-between-full-virtualization-and-paravirtualization/

Introduction to Docker Containers

Docker containers are the virtuosos of modern software development, revolutionizing the way applications are built, shipped, and run across diverse computing environments.

What are containers?

Containers are a form of virtualization that allows you to package and isolate applications along with their dependencies, enabling them to run uniformly across different computing environments.

Containers encapsulate everything an application needs to run, such as libraries, configuration files, and code, into a single unit. This approach ensures consistency in deploying applications across various environments, from development to testing and production.

What is docker?

Docker is a platform for building, shipping, and running containers. It simplifies the process of creating and managing containers by providing tools and a platform that streamlines containerization.

Docker uses a high-level API to provide a lightweight containerization virtualization solution, making it easier to package applications and their dependencies into standardized units called Docker containers.

Docker offers a user-friendly interface and CLI tools that make working with containers more accessible and manageable, abstracting complexities associated with container deployment and allowing developers to focus on building and shipping applications. Docker's methodology ensures consistency across development, testing, and production environments. Developers craft these containers on their local machines, ensuring that what works locally will seamlessly function in any environment, be it a colleague's computer, a test server, or a production cluster. This consistency diminishes the notorious “It works on my machine” discrepancies, streamlining the development pipeline. The magic of Docker doesn’t halt there; it extends to orchestration tools like Kubernetes, which manage the deployment, scaling, and operation of containerized applications. This pairing of Docker with orchestration solutions amplifies its prowess, enabling the management of vast fleets of containers effortlessly.

Docker Architecture

Docker containers rely on a client-server architecture and consist of several key elements

Docker Daemon is a background service that runs on the host system and manages container lifecycle operations.

Docker Client is the primary interface through which users interact with Docker. It communicates with the Docker Daemon, sending commands and receiving information about containers, images, networks, and other Docker objects.

Docker Images serve as read-only templates that contain the application code, libraries, dependencies, and configurations needed to run an application.

Containers are instances of Docker images. They encapsulate applications and their dependencies, providing an isolated environment to run processes. Each container operates independently, having its own filesystem, networking, and isolated system resources while sharing the host OS kernel.

Docker Registry stores Docker images. The Docker Hub is a popular public registry where users can find and share Docker images.

Docker provides networking capabilities that allow containers to communicate with each other and with external networks.

All these components work together cohesively to facilitate the creation, deployment, and orchestration of Docker containers, enabling developers to build, ship, and run applications consistently across different environments.

Use Cases

Docker containers have a wide range of use cases across various industries and applications. The most common use cases of Docker are described below

`Web Development and Deployment`

Docker is widely used for packaging and deploying web applications. It provides a consistent and isolated environment for web applications, ensuring that they run the same way regardless of the deployment environment. This makes it easier to develop, test, and deploy web applications, and it also helps to reduce downtime and improve application performance.

`Microservices Architecture`

Docker is the foundation of microservices architecture, Each microservice runs in its container, which makes it easy to develop, test, and deploy independently. Docker also makes it easy to scale microservices up or down as needed.

`DevOps and IT Operations`

Docker is an essential tool for DevOps teams, which are responsible for bridging the gap between development and operations. Docker containers can be used to create and manage complex IT environments, and they can also be used to automate the deployment and management of applications.

`Software Testing`

Docker is used extensively for software testing. Docker containers can be used to create isolated test environments, making it easy to test applications without interfering with production environments.

These are just a few examples of the many ways that Docker is being used today. Docker is a versatile and powerful tool that can be used to solve a wide range of problems. As the adoption of containerization continues to grow, we can expect to see even more innovative use cases for Docker in the future.

Useful link: https://github.com/docker

Study material:

Docker Official Documentation: https://docs.docker.com/

Docker Hub: https://docs.docker.com/docker-hub/

How to Deploy Docker Containers to Production: https://www.linode.com/docs

Docker container operations

Docker pull

Use the docker pull command to pull an image from a registry.

Syntax

docker pull [options] image_name:Tag

Docker run

To create and start a container from a Docker image, use the docker run command. This command also pulls the image if it's not already available.

Syntax

docker run [options] image [command][args]

Docker ps

To see the list of running containers, use the docker ps command.

Syntax

docker ps [options]

List all containers

Syntax

docker ps -a

Docker exec

You can access the shell of a running container using the docker exec command.

Syntax

docker exec [options] container [command] [args]

Docker Stop

To stop a running container, use the docker stop command.

Syntax

docker stop container

Docker rm

To remove a stopped container, use the docker rm command.

Syntax

docker rm container

Docker Logs

You can view the logs of a running container using the docker logs command.

Syntax

docker logs [options] container

Docker cp

To copy files between the host and a container, you can use docker cp.

Linux Fundamentals

Linux, the open-source operating system, embodies a powerful foundation for computing systems across the globe. Its fundamentals form the bedrock of its widespread use, underpinning everything from personal computers to servers and embedded devices. At its core, Linux champions modularity, security, and versatility.

File Structure

Linux organizes files in a hierarchical structure, starting from the root directory ("/"). Key directories include:

```
/dev: Device files.

/bin: Essential binaries (basic commands like ls, cp).

/etc: System configuration files.

/home: User home directories.

/var: Variable data (logs, caches).

/tmp: Temporary files.

/proc: Virtual filesystem providing system information.
```

Filesystem Types

Linux supports various file system types that offer different features and performance characteristics. File system types include `ext4`

`btrfs`

`xfs`

User and groups

Every user in Linux has a unique identifier (UID) and is a member of at least one group. Groups help manage file access permissions.

Permissions

Linux uses permissions to control access to files and directories. Each file has permissions for the owner, group, and others, you can set permissions to

`read`

`write`

`execute`

Shell

The shell is a command-line interface that interprets user commands and executes programs. Popular shells include

`Bash (Bourne Again SHell)`

`Zsh`

`Fish`

Networking

Linux offers robust networking capabilities, with commands like ifconfig, ip, and tools for managing networks, configuring interfaces, and troubleshooting connections.

Syntax

sudo apt install net-tools

Package Manager

Distributions use package managers to install, update, and remove the software. The package manager changes with distributions, some of them are mentioned below

`apt`

`yum`

`pacman`

Updates

Linux is a constantly evolving operating system, and new updates are released regularly. Users should keep their systems up to date to ensure that they have the latest security patches and bug fixes.

Mastery of these above-mentioned fundamentals not only empowers users with a deeper understanding of the system but also forms the basis for leveraging Linux's vast potential in diverse computing environments.

Study material

Linux Permissions for users and groups: https://www.section.io/engineering-education/user-groups-and-permissions-linux/

Linux Networking: https://itsfoss.com/basic-linux-networking-commands/

Linux Shells: https://phoenixnap.com/kb/linux-shells

Syntax

docker cp [options] container:src_path dest_path

Linux Fundamentals

Linux, the open-source operating system, embodies a powerful foundation for computing systems across the globe. Its fundamentals form the bedrock of its widespread use, underpinning everything from personal computers to servers and embedded devices. At its core, Linux champions modularity, security, and versatility.

File Structure

Linux organizes files in a hierarchical structure, starting from the root directory ("/"). Key directories include:

```
/dev: Device files.

/bin: Essential binaries (basic commands like ls, cp).

/etc: System configuration files.

/home: User home directories.

/var: Variable data (logs, caches).

/tmp: Temporary files.

/proc: Virtual filesystem providing system information.
```

Filesystem Types

Linux supports various file system types that offer different features and performance characteristics. File system types include `ext4`

`btrfs`

`xfs`

User and groups

Every user in Linux has a unique identifier (UID) and is a member of at least one group. Groups help manage file access permissions.

Permissions

Linux uses permissions to control access to files and directories. Each file has permissions for the owner, group, and others, you can set permissions to

`read`

`write`

`execute`

Shell

The shell is a command-line interface that interprets user commands and executes programs. Popular shells include

` Bash (Bourne Again SHell)`

` Zsh`

` Fish`

Networking

Linux offers robust networking capabilities, with commands like ifconfig, ip, and tools for managing networks, configuring interfaces, and troubleshooting connections.

Syntax

sudo apt install net-tools

Package Manager

Distributions use package managers to install, update, and remove the software. The package manager changes with distributions, some of them are mentioned below

` apt`

`yum`

`pacman`

Updates

Linux is a constantly evolving operating system, and new updates are released regularly. Users should keep their systems up to date to ensure that they have the latest security patches and bug fixes.

Mastery of these above-mentioned fundamentals not only empowers users with a deeper understanding of the system but also forms the basis for leveraging Linux’s vast potential in diverse computing environments.

Study material

Linux Permissions for users and groups: https://www.section.io/engineering-education/user-groups-and-permissions-linux/

Linux Networking: https://itsfoss.com/basic-linux-networking-commands/

Linux Shells: https://phoenixnap.com/kb/linux-shells

Linux Command-Line Interface (CLI)

Introduction

The Linux command-line interface (CLI) is a text-based interface that allows users to interact with the Linux operating system. It is a powerful tool that can be used to perform a wide variety of tasks, including managing files and directories, installing software, configuring the system, and troubleshooting problems.

Linux CLI Commands

Print the Current Working Directory

To print the current working directory use pwd command.

Syntax

pwd

List Contents

To list the contents of the current working directory use ls command.

Syntax

ls

Change Directory

To change the current working directory use cd command.

Syntax

cd

Create Directory

To create a new directory use the mkdir command.

Syntax

```
mkdir directory_name
```

Remove Empty Directory

To remove an empty directory use rmdir command.

Syntax

```
rmdir directory_name
```

Remove File/Directory

Use rm command to remove a file or directory.

Syntax

```
rm file_name
```

To delete a directory

Syntax

```
rm -r directory_name
```

Copy File/Directory

Use cp command to copy a file or directory.

Syntax

```
cp directory_name dest_path
```

Move/Rename Directory

To move a file or directory use mv command.

Syntax

```
mv directory_name dest_path
```

To rename a file or directory use mv command.

Syntax

```
mv directory_name new_directory_name
```

These are just some of the basic Linux CLI commands that you should know. There are many other CLI commands that you can learn about as you become more familiar with Linux.

Study materialS

https://www.digitalocean.com/community/tutorials/linux-commands

https://www.hostinger.com/tutorials/linux-commands

Introduction to Linux

What is Linux?

Linux, the brainchild of Linus Torvalds, stands tall as an open-source, Unix-like operating system revered for its power, flexibility, and community-driven ethos. At its core, Linux embodies the spirit of collaboration, offering a robust foundation that powers a myriad of devices, from smartphones and servers to supercomputers and embedded systems. Linux embraces an open philosophy, allowing users to access, modify, and distribute its source code freely. This approach nurtures a vibrant ecosystem where developers, enthusiasts, and enterprises contribute to its evolution, fostering innovation and adaptability. Linux adaptability spans across architectures, supporting a plethora of hardware platforms from x86 and ARM to mainframes, ensuring its ubiquity in an increasingly diverse computing landscape.

Linux Features

The core features of Linux are described below

```
Security Fortitude
```

Linux boasts robust security measures, including stringent user permissions, encrypted file systems, and a robust firewall, earning it a reputation for stability and resilience against cyber threats.

```
 Multiuser and multitasking Capabilities
```

Linux has the Multiuser and multitasking Capabilities, Linux can handle multiple users and numerous processes concurrently.

```
Customizability and adaptability
```

Linux provides vast Customizability and adaptability. it offers a multitude of distributions, each tailored to specific needs, empowering users to select and customize their systems according to preferences.

Linux Architecture

Linux consists of a four-layer Architecture.

```
Hardware: The hardware layer is the physical layer of the computer system.

Kernel: The kernel is the core of the Linux operating system. It is respon

System utilities: The system utility layer provides a set of tools that ar

User applications: The user applications layer is the top layer of the Lin
```

Linux Distributions

The distinctive distributions of Linux are described below

Ubuntu is Recognized for its user-friendly interface and robust community

Fedora is Focused on pioneering software advancements, appealing particula

Debian is Renowned for its stability, frequently serving as the foundation

Arch Linux Offers extensive customization options but demands a higher tec

Linux Applications

Linux is a versatile operating system that can be used for a wide variety of purposes. Here are some of the most common applications of Linux

Web servers

Linux is the most popular operating system for web servers. It is a powerful, stable, and secure platform that can handle a high volume of traffic. Some of the most popular web servers that run on Linux include Apache, Nginx, and Lighttpd.

Desktop Computers

Linux is also a popular choice for desktop computers. It is a powerful and flexible operating system that can be customized to meet the needs of a wide variety of users. Some of the most popular Linux distributions for desktop computers include Ubuntu, Fedora, and Mint.

Other types of Servers

Linux is also used for a variety of other types of servers, including file servers, mail servers, database servers, print servers, and game servers.

Embedded systems

Linux is also a popular choice for embedded systems. Embedded systems are specialized computers that are designed to perform a specific task. Some examples of embedded systems that run on Linux include routers, switches, firewalls, mobile phones, and smart TVs.

These are just a few examples of the many applications of Linux. Linux is powerful and versatile and is used by millions of people around the world. It is a great choice for anyone who wants a stable, reliable, and secure operating system.

Useful link: https://github.com/torvalds/linux

Study material

Linux Introduction: https://www.javatpoint.com/what-is-linux

Linux Distributions: https://itsfoss.com/best-linux-beginners/