DAILY DAIRY FOR MY TRAINING

15-Aug-2024

Training Report:

What is AWS Cloud?

AWS Cloud, or Amazon Web Services Cloud, is a comprehensive platform of cloud computing services offered by Amazon. It allows individuals, companies, and governments to access computing resources (like servers, storage, databases, networking, and software) over the internet, rather than managing their own hardware and software.

Here's an overview of what AWS Cloud offers and how it works:

Compute

Services that provide virtual servers, serverless computing, and container orchestration.

Examples:

Amazon EC2: Virtual machines in the cloud.

AWS Lambda: Serverless computing (run code without provisioning servers).

Amazon ECS/EKS: Manage and deploy containers using Docker and Kubernetes.

2. Storage

Flexible and scalable storage solutions for different use cases like backups, archiving, or active data storage.

Examples:

Amazon S3: Object storage for storing data securely.

Amazon EBS: Block storage for EC2 instances.

Amazon Glacier: Long-term archival storage.

3. Networking

Tools to create secure, fast, and reliable networks in the cloud.

Examples:

Amazon VPC: Isolated private networks for your resources.

AWS Route 53: Domain name system (DNS) service.

Elastic Load Balancing (ELB): Automatically distributes traffic.

4. Databases

Managed database services for relational, NoSQL, and data warehouse needs.

Examples:

Amazon RDS: Relational databases (e.g., MySQL, PostgreSQL, SQL Server).

Amazon DynamoDB: NoSQL database for high-speed and scalable apps.

Amazon Redshift: Data warehousing for big data analytics.

5. AI and Machine Learning

Tools to build, train, and deploy machine learning models.

Examples:

Amazon SageMaker: ML model development.

Amazon Rekognition: Image and video analysis.

Amazon Lex: Build chatbots and conversational interfaces.

6. Developer Tools

Services for automating software development and delivery.

Examples:

AWS CodePipeline: CI/CD for application deployment.

AWS CodeDeploy: Automates deployment to servers or containers.

7. Analytics

Tools for processing and analyzing large amounts of data.

Examples:

Amazon Athena: SQL queries on data stored in S3.

Amazon EMR: Big data processing using Apache Spark, Hadoop.

Amazon Kinesis: Real-time data streaming.

8. Security and Identity

Secure and control access to AWS resources.

Examples:

AWS IAM: Manage user permissions.

AWS Shield: DDoS attack protection.

AWS WAF: Web Application Firewall for applications.

9. IoT

Tools to connect and manage IoT devices.

Examples:

AWS IoT Core: Connect IoT devices to the cloud.

AWS Greengrass: Run local compute and ML on IoT devices.

10. Hybrid and Edge Computing

Services to extend cloud functionality to on-premises or edge locations.

Examples:

AWS Outposts: Run AWS services on-premises.

AWS Wavelength: Low-latency applications for 5G networks.

11. Migration and Transfer

Tools to help migrate data and applications to AWS.

Examples:

AWS Migration Hub: Centralize and monitor migration tasks.

AWS DataSync: Transfer data between on-premises storage and AWS.

Create AWS Cloud Account:

Here are the steps to create an AWS account:

Step 1: Visit the AWS Sign-Up Page:

-- Go to aws.amazon.com (https://aws.amazon.com/) and click on "Create an AWS

Account" in the top right corner.

Step 2. Enter Your Email and Create a Password:

-- Provide a valid email address, choose an account name, and create a strong password.

This will be used as your root account.

Step 3. Select the AWS Account Type:

-- Choose Personal (for individual use) or Professional (for business use), and fill in the required information, including contact details.

Step 4. Add Payment Information:

-- Enter a valid credit or debit card. AWS will charge a small, refundable fee (around $1) to verify the payment method.

Step 5. Verify Your Identity:

-- Complete the phone verification by receiving a call or SMS with a verification code.

Step 6. Select a Support Plan:

-- Choose a support plan (Basic Support is free, with paid options for advanced support).

Step 7. Activate Your Account:

-- After completing the steps, AWS will process your information and activate your account, which may take a few minutes. Once activated, you can sign in and start using AWS services.

AWS offers a Free Tier for new accounts, allowing you to try many services at no cost for the first 12 months.

16-Aug-2024

Training Day - 2 Report:

What is PuTTy Software?

PuTTY is a free, open-source SSH (Secure Shell) and Telnet client used primarily to remotely connect and manage servers. Available for Windows, it provides a secure way to access command-line interfaces on remote computers, commonly used for managing Linux servers. PuTTY also supports other protocols, like SCP and SFTP, for secure file transfers, and includes tools like PuTTYgen for key generation.

To install PuTTY on Windows:

1. Download the Installer:

-- Go to the official PuTTY website (https://www.putty.org/), navigate to "Download PuTTY," and download the latest Windows version.

2. Run the Installer:

-- Locate the downloaded `.msi` file, double-click to run it, and follow the setup prompts.

3. Choose Installation Options:

-- Select the installation location (default is usually fine), and choose to create a desktop shortcut if desired. You may need admin rights to complete the installation.

4. Complete Installation:

Once installation finishes, click “Finish.”

5. Verify and Launch PuTTY:

-- Open PuTTY from your desktop or Start menu, set up your SSH connection (hostname, port, etc.), and click “Open” to connect.

PuTTY is now installed and ready for SSH connections on your Windows system.

What is WinSCP Software?

WinSCP is a free, open-source file transfer client for Windows, used to securely transfer files

between a local computer and a remote server. It supports SFTP (SSH File Transfer Protocol), SCP (Secure Copy Protocol), and FTP (File Transfer Protocol). WinSCP provides an intuitive graphical interface for file management, allowing users to drag and drop files, and it also integrates with PuTTY for SSH access to remote servers.

To install WinSCP on Windows:

1. Download the Installer:

-- Visit the official WinSCP website (https://winscp.net/eng/download.php) and download the latest version for Windows.

2. Run the Installer:

Locate the downloaded `.exe` file, double-click it, and follow the installation prompts.

3. Choose Setup Options:

Select your preferred installation type (typical setup is recommended), installation location, and interface language.

4. Complete Installation:

-- Click “Finish” once the setup is complete. Optionally, choose to launch WinSCP immediately.

5. Launch WinSCP and Configure Connection:

-- Open WinSCP from the desktop or Start menu, enter your server’s hostname, username, password, and protocol (like SFTP or FTP), then click “Login” to connect.

WinSCP is now installed and ready for secure file transfers on your Windows system.

To launch and connect to an AWS EC2 instance using PuTTY, follow these steps:

1. Launch an EC2 Instance

-- Sign in to AWS Console: Go to AWS Console (https://aws.amazon.com/console/) and open the EC2 Dashboard.

-- Launch Instance: Select Launch Instance, then choose an Amazon Machine

Image (AMI) (e.g., Amazon Linux 2).

-- Instance Type: Choose an instance type (e.g., t2.micro for the free tier)

-- Configure Instance: Set network and storage preferences as needed, then proceed to the Key Pair step.

-- Key Pair: Enter a Key Pair Name (e.g., "main"), download the `.pem` file (AWS format), and Launch the instance.

2. Convert Key Pair for PuTTY

-- Convert .pem to .ppk: Use PuTTYgen (included with PuTTY) to convert the `.pem` file to a `.ppk` file:

-- Open PuTTYgen, click Load, and select your `.pem` file.

-- Click Save Private Key and save the file as `main.ppk`.

3. Get EC2 Instance Details

-- Access Instance ID: Go back to the EC2 dashboard, click on the Instance ID to open instance details.

-- Copy IPv4 Address: Under Public IPv4 address, copy the instance's IP address for later use.

4. Connect to EC2 Using PuTTY

-- Open PuTTY: Launch PuTTY on your computer.

-- Session Configuration:

In Session, enter the IPv4 Address in the Host Name field (e.g., `ec2-user@<publicip>`).

-- Add Private Key:

Go to Connection > SSH > Auth.

Under Credentials, click Browse and select your `main.ppk` file.

5. Save and Open Session

-- Save Session: Go back to Session and enter a name (e.g., "My EC2 Instance") in

Saved Sessions, then click Save.

-- Connect: Click Open to establish an SSH connection to the EC2 instance.

6. SSH Verification and Access

-- Accept Security Alert: If prompted with a security alert, click Yes to accept the host key.

-- Use the Shell: You are now connected via SSH to your EC2 instance and can use the command-line interface for managing your instance.

Day – 18 Aug, 2024

Training Day - 3 Report:

What is Web Server?

A web server is a software or hardware that serves content to the web. It processes incoming requests from clients (usually web browsers) and delivers web pages, images, videos, and other content over the Internet using the Hypertext Transfer Protocol (HTTP). The main function of a web server is to store, process, and deliver web pages to users.

Introduction of Apache web server:

Apache HTTP Server, commonly referred to as Apache, is one of the oldest and most widely used web servers in the world. It was developed by the Apache Software Foundation and first released in 1995. Here are some key features and characteristics of Apache:

Key Features:

1. Open Source: Apache is free to use and distribute, and its source code is available for modification.
2. Cross-Platform: It runs on various operating systems, including Unix, Linux, Windows, and macOS.
3. Modular Architecture: Apache supports a modular architecture, allowing users to enable or disable specific features through modules. This includes support for various programming languages (e.g., PHP, Python) and features like URL rewriting, authentication, and caching.
4. Configuration Flexibility: Apache uses configuration files (httpd.conf, .htaccess) that allow for fine-grained control over server settings, directory permissions, and URL handling.
5. Robust Community Support: Due to its long history, Apache has a large community and extensive documentation, making it easier to find support and resources.
6. Virtual Hosting: Apache supports both name-based and IP-based virtual hosting, allowing multiple domains to be hosted on a single server.
7. Security Features: Apache offers various security modules and features, including

SSL/TLS support, authentication, and access control.

Use Cases:

-- Ideal for serving dynamic content and applications, especially when combined with languages like PHP or Python.

-- Commonly used for hosting websites, web applications, and content management systems (CMS) like WordPress.

Introduction of Nginx web server:

Nginx (pronounced "engine-x") is a high-performance web server and reverse proxy server created by Igor Sysoev and first released in 2004. It has gained significant popularity due to its speed and efficiency. Here are some key features and characteristics of Nginx:

Key Features:

1. Event-Driven Architecture: Nginx uses an asynchronous, event-driven architecture, which allows it to handle many connections simultaneously with low resource consumption. This makes it well-suited for high-traffic websites.
2. Reverse Proxy and Load Balancing: Nginx is commonly used as a reverse proxy server, distributing incoming traffic among multiple backend servers. It can also perform load balancing, caching, and SSL termination.
3. Static File Serving: Nginx excels at serving static content (e.g., images, CSS, JavaScript) quickly and efficiently, making it a popular choice for serving static websites.
4. Configuration Simplicity: Nginx configuration files are straightforward and easy to read, which simplifies server setup and management.
5. High Performance: Nginx is known for its speed and ability to handle thousands of concurrent connections with minimal resource usage.
6. Support for HTTP/2 and HTTP/3: Nginx supports modern web protocols like HTTP/2 and HTTP/3, enhancing performance and security.

7. Modular Design: While Nginx does not have the same level of modularity as

Apache, it does support various modules for extended functionality, including thirdparty modules.

Use Cases:

-- Often used for serving static websites and as a reverse proxy for dynamic

applications.

-- Commonly employed in high-traffic environments, such as content delivery networks

(CDNs) and large-scale web applications.

Step-by-Step Guide to Set Up and Deploy an HTML Website on an AWS EC2 Instance

with Apache

This guide walks through setting up an EC2 instance, installing Apache, and uploading a

website template using WinSCP.

1. Launch an EC2 Instance

-- Login to AWS: Access your AWS console and open EC2 from the dashboard.

-- Launch Instance: Choose an Amazon Machine Image (AMI), such as Amazon Linux

2, and select an instance type (e.g., t2.micro).

-- Key Pair: Enter a Key Pair Name (e.g., "main"), download the `.pem` file, and

launch the instance.

2. Connect to the EC2 Instance Using PuTTY

-- Convert Key Pair: Use PuTTYgen to convert your `.pem` file to `.ppk`.

-- Open PuTTy:

Enter the IPv4 address of your instance in Host Name (e.g., `ec2-user@<publicip>`).

Go to Connection > SSH >Authand upload the `main.ppk` file.

-- Login as ec2-user: When the SSH session opens, type `ec2-user` and press Enter.

3. Install Apache Web Server

-- Update and Install Apache:

Run `sudo yum install httpd -y` to install Apache on your EC2 instance.

-- Status Apache Service:

Use `sudo service httpd status` to start Apache.

Verify by visiting `http://<instance-public-ip>` in a browser.

4. Prepare to Upload HTML Files with WinSCP

-- Download an HTML Template:

Visit ThemeWagon (https://themewagon.com/theme-tag/html5-css3/) and download an HTML5 template to your computer.

-- Configure WinSCP:

Open WinSCP and enter your EC2 instance's IPv4 address with port 22.

-- Go to Advanced Settings > Authentication and upload your `.ppk` file.

-- Connect: Log in, and your local files will appear on the left (LHS), and the server files on the right (RHS).

5. Set Up the Directory for HTML Files

-- Modify Directory Permissions:

In the SSH terminal, run `sudo chown -R -v ec2-user /var/www/html` to change permissions.

-- Upload HTML Template:

In WinSCP, navigate to `/var/www/html` on the RHS.

Drag and drop your HTML files from your local system (LHS) into the `/var/www/html` directory on the server.

6. Start/Stop Apache and View Your Website

1 Control Apache Service:

Start with `sudo service httpd start` and stop with `sudo service httpd stop` as needed.

2 Verify the Website:

Visit `http://<instance-public-ip>` to see your uploaded HTML template live.

Your website should now be hosted on AWS EC2 and accessible via the public IP address!

19-Aug-2024

Training Day - 4 Report:

AWS EC2 Setup with Nginx Website Deployment

1. Creating an EC2 Instance

-- Login to AWS Console.

-- Navigate to EC2 Dashboard.

-- Click “Launch Instance”.

-- Configure Instance:

a) Name: nginx-webserver.

b) AMI: Amazon Linux 2023.

c) Instance Type: t2.micro (free tier).

d) Key Pair: Create new (nginx-key), download .pem.

e) Network Settings: Allow SSH (22), HTTP (80), HTTPS (443).

-- Click “Launch Instance”.

2. Connecting via PuTTY

-- Open PuTTY.

-- Host Name: ec2-user@[Your-EC2-Public-IP].

-- Port: 22.

-- SSH → Auth: Browse for your .ppk file.

-- Click “Open” to connect.

3. Installing and Configuring Nginx

-- sudo yum update -y.

-- sudo yum install nginx -y.

-- sudo systemctl start nginx.

-- sudo systemctl enable nginx.

-- sudo systemctl status nginx.

4. Setting Up Website Directory

-- sudo chmod -R 755 /usr/share/nginx/html

5. Uploading Website Files Using WinSCP

-- Open WinSCP

-- Protocol: SFTP.

-- Host name: [Your-EC2-Public-IP].

-- Port: 22.

-- Username: ec2-user.

-- Advanced → Authentication: Browse for your .ppk file.

-- Connect and upload files to /usr/share/nginx/html.

6. Verifying Setup

-- Open Web Browser

-- Enter your EC2 public IP.

-- Verify your website is visible.

22-Aug-2024

Training Day - 5 Report:

Domain and Hosting Charges?

A domain is the web address (e.g., www.example.com) that users type into a browser to access a website. Hosting refers to the service that stores your website files and makes them accessible on the internet. These are separate services, and each incurs its own charges. Domain registration typically involves an annual fee, while hosting services can be billed monthly or annually based on the type of plan chosen.

Knowledge of Server, SQL, and PHP?

A server is a computer or system that provides resources, data, services, or programs to other computers, known as clients, over a network. SQL (Structured Query Language) is a standard language for managing and manipulating databases. PHP (Hypertext Preprocessor) is a server-side scripting language used for web development, allowing developers to create dynamic web pages. Understanding these technologies is crucial for full-stack development, enabling seamless interaction between the front-end and back-end of a web application.

IIS Server?

IIS (Internet Information Services) is a web server created by Microsoft for hosting websites and applications on Windows servers. It supports various protocols, including HTTP, HTTPS, FTP, and more. IIS is specifically designed for Windows environments, offering features like security, logging, and application management, making it a popular choice for enterprises that use Microsoft technologies.

XAMPP, LAMP, and WAMP?

-- XAMPP: A cross-platform web server solution stack package that includes Apache, MySQL, PHP, and Perl, designed for easy installation and use on multiple operating systems.

-- LAMP: A stack consisting of Linux, Apache, MySQL, and PHP, commonly used for developing and deploying web applications on Linux servers.

-- WAMP: A Windows-based stack that includes Windows, Apache, MySQL, and PHP,

providing a similar development environment for Windows users.

Requirements for Developing a PHP Website?

To develop a PHP website, you typically need:

-- A web server (like Apache or Nginx) to serve your PHP files.

-- A database (like MySQL) to store and manage your data.

-- A PHP interpreter to process the PHP code and generate dynamic content.

-- A development environment (like XAMPP, LAMP, or WAMP) to set up these components easily on your local machine.

Need for Apache Server?

Apache is one of the most widely used web servers in the world, known for its flexibility, power, and community support. It serves web content to users by processing HTTP requests and delivering HTML pages, images, and other resources. Apache supports various modules, allowing for extended functionality like URL rewriting, authentication, and security features, making it a critical component for hosting dynamic websites and applications.

Introduction of Elastic Ip:

An Elastic IP in AWS is a static public IPv4 address that you can allocate to your AWS account. It allows you to maintain a consistent IP address for your resources, even if you stop or restart your EC2 instances. This is particularly useful for applications that require a stable endpoint for users to connect to.

Uses of Elastic IP:

-- Static IP Addressing: Unlike standard public IP addresses that change when an instance is stopped or restarted, an Elastic IP remains constant, ensuring that your applications can always be accessed at the same IP address.

-- Failover Capability: If an instance fails or needs to be replaced, you can quickly remap the Elastic IP to another instance, minimizing downtime and maintaining service continuity.

-- Consistent Endpoint: Elastic IPs serve as stable identifiers for your cloud resources, which is beneficial for configuring external services, such as DNS records or firewall rules.

-- Dynamic Allocation: You can programmatically associate and disassociate Elastic IPs as needed, allowing you to adapt to changing demands and scale your infrastructure efficiently.

Steps to Create and Associate an Elastic IP with a Launch Instance:

Open the Amazon EC2 Console:

-- Navigate to the Amazon EC2 console.

Allocate an Elastic IP Address:

-- In the navigation pane, choose Network & Security and then select Elastic IPs.

-- Click on Allocate Elastic IP address.

-- (Optional) Choose the Network border group if you want to specify where the Elastic IP will be allocated.

-- Click Allocate to create the Elastic IP.

Launch an EC2 Instance:

-- In the EC2 console, click on Instances and then Launch Instance.

-- Follow the prompts to select an Amazon Machine Image (AMI), instance type, and configure instance details.

-- Ensure that the instance is in a public subnet to allow internet access.

Associate the Elastic IP with the Instance:

-- After launching the instance, go back to the Elastic IPs section in the EC2 console.

-- Select the Elastic IP address you allocated.

-- Click on Actions and choose Associate Elastic IP address.

-- For Resource type, select Instance and then choose the instance you just launched.

-- Click Associate to link the Elastic IP with your instance.

Verify the Association:

-- Once associated, you can check the instance details to confirm that the Elastic IP is now linked to your instance.

-- You can also test connectivity by accessing the instance using the Elastic IP address.

Benefits of Associating an Elastic IP:

-- Persistent Connectivity: Your applications can maintain a consistent IP address, even if the underlying instance changes.

-- Easy Failover: In case of instance failure, you can quickly remap the Elastic IP to a new instance, ensuring minimal downtime.

-- Simplified DNS Management: You can point your domain names to the Elastic IP, making it easier to manage your web applications.

23-July-2024

Training Day - 6 Report:

What is the mean of 400 in chmod ?

In the context of Unix and Linux file permissions, the chmod command is used to change the permissions of a file or directory. The permissions are represented using a three-digit octal number, where each digit corresponds to a different set of permissions.

Understanding chmod 400

-- 400 is an octal representation of file permissions.

-- Each digit represents permissions for the owner, group, and others, respectively.

Here's the breakdown of 400:

1. First Digit (4):

-- This represents the permissions for the owner of the file.

-- The value 4 corresponds to read permission. This means the owner can read the file but cannot write to it or execute it.

-- In binary, this is represented as 100, which means:

1. Read (4): Yes

2. Write (2): No

3. Execute (1): No

2. Second Digit (0):

-- This represents the permissions for the group.

-- The value 0 means no permissions are granted to the group. In binary, this is represented as 000.

3. Third Digit (0):

-- This represents the permissions for others (everyone else).

-- Similarly, the value 0 means no permissions are granted to others. In binary, this is also represented as 000.

Summary of chmod 400

When you set permissions to 400 using chmod, you are effectively allowing only the owner

of the file to read it, while the group and others have no permissions at all.

This is a common setting for sensitive files (like private keys or configuration files) that should only be accessible by the user who owns them.

Introduction of Filezilla?

FileZilla is a popular, open-source file transfer protocol (FTP) client that allows users to transfer files between a local computer and a remote server. It supports multiple file transfer protocols, including FTP, SFTP (SSH File Transfer Protocol), and FTPS (FTP Secure).

FileZilla is widely used by web developers, system administrators, and anyone needing to upload, download, or manage files on a server. The interface is user-friendly, providing an easy drag-and-drop system for file management, and it includes features like:

-- Cross-platform compatibility: Works on Windows, macOS, and Linux.

-- File transfer resume: Allows users to resume file transfers if they get interrupted.

-- Support for large files: Handles files larger than 4GB.

-- Directory comparison: Highlights differences between local and remote directories.

-- Remote file editing: Allows users to edit files on the server without downloading them first.

FileZilla also has a server counterpart, FileZilla Server, which enables users to set up their own FTP server for secure file sharing.

Windows Installation

1. Download FileZilla Client from filezilla-project.org (https://filezilla-project.org/) for Windows.

2. Run the downloaded `.exe` installer file.

3. Accept the license agreement in the installation wizard.

4. Choose installation options and select the installation location.

5. Click Install to complete the installation.

Linux Installation (Ubuntu/Debian-based)

1. Update your package list with `sudo apt update`.

2. Install FileZilla using `sudo apt install filezilla'.

3. Verify the installation by typing `filezilla` in the terminal.

Setting Up an Nginx Web Server on an Amazon EC2 Instance and Uploading Files via FileZilla

Step-by-Step Guide

1. Create an EC2 Instance:

-- Log in to the AWS Management Console, navigate to EC2, and click on "Launch Instance" to create a new instance using an Amazon Machine Image (AMI) like Ubuntu or Amazon Linux.

2. Configure Instance Details:

-- Choose instance type (e.g., t2.micro for free tier eligibility), configure instance details, and click "Next" until you reach the "Review and Launch" section.

3. Set Security Group:

-- Create a new security group or select an existing one, ensuring to allow inbound traffic on ports 22 (SSH) and 80 (HTTP) for web access.

4. Launch the Instance:

-- Review your settings and click "Launch," then select or create a new key pair (download the .pem file) to access the instance.

5. Access the Instance via SSH:

-- Open a terminal and use the command ssh -i /path/to/your-key.pem ec2-user@your-instance-public-ip (replace with your key path and instance public IP) to connect.

6. Update Package Repository:

-- Run sudo apt update (for Ubuntu) or sudo yum update (for Amazon Linux) to ensure your package repository is up to date.

7. Install Nginx:

-- Execute sudo apt install nginx (for Ubuntu) or sudo yum install nginx (for Amazon Linux) to install the Nginx web server.

8. Start Nginx Service:

-- Start Nginx with sudo systemctl start nginx and enable it to start on boot with sudo systemctl enable nginx.

9. Verify Nginx Installation:

-- Open a web browser and navigate to http://your-instance-public-ip to see the Nginx welcome page.

10. Prepare for File Upload:

-- Ensure your instance's security group allows inbound traffic on port 21 (FTP) or use SFTP (port 22) for secure file transfer.

11. Open FileZilla:

-- Launch FileZilla on your local machine and go to Site Manager (File > Site Manager).

12. Configure FileZilla Connection:

-- Create a new site with Host as your instance's public IP, Protocol as SFTP, and enter the username (ec2-user for Amazon Linux or ubuntu for Ubuntu) and the path to your private key in the settings.

13. Connect to the Instance:

-- Click "Connect" in FileZilla to establish a connection to your EC2 instance.

14. Navigate to the Web Directory:

-- In FileZilla, navigate to the web root directory (usually /var/www/html for Nginx) in the right panel.

15. Upload Your File:

-- Drag and drop your website files from the left panel (local) to the right panel

(remote) to upload them to the Nginx web directory.

16. Verify File Upload:

-- Go back to your web browser and navigate to http://your-instance-publicip/your-file.html to ensure the uploaded file is accessible.

By following these steps, you will have successfully created an EC2 instance, installed

Nginx, and uploaded files using FileZilla.

24-July-2024

Training Day - 7 Report:

Overview of File Permissions and User Management in Linux

Understanding File Permissions in Linux

File Permissions:

In Linux, each file and directory has associated permissions that determine who can read, write, or execute them. Permissions are divided into three categories:

-- Owner: The user who owns the file.

-- Group: A group of users who share permissions.

-- Others: All other users on the system.

Permission Types:

-- Read (r): Permission to read the file or list the directory.

-- Write (w): Permission to modify the file or add/remove files in a directory.

-- Execute (x): Permission to execute a file or access a directory.

Checking File Permissions

-- Use the command ls -l to list files and their permissions in a directory.

Example Output: -rw-r--r-- 1 user group 0 Oct 1 12:00 file.txt

1. The first character indicates the type (e.g., - for a file, d for a directory).
2. The next three characters represent owner permissions (read, write, execute).
3. The following three represent group permissions.
4. The last three represent permissions for others.

-- Use ls -ltr to list files in long format sorted by modification time (oldest first).

Creating and Managing Users

1. Create a User:

-- Command: sudo useradd -m -s /bin/bash cgoyal

1. -m: Creates a home directory for the user.
2. -s /bin/bash: Sets the default shell to bash.

-- Set a password: sudo passwd cgoyal

2. View User Information:

-- Command: cat /etc/passwd

-- Displays user account information, including username, user ID, group ID, home directory, and shell.

3. Delete a User:

-- Command: sudo userdel ram

-- To remove the user and their home directory: sudo userdel -r ram

4. Create Another User:

-- Command: sudo useradd -m -s /bin/bash chintu

-- Set a password: sudo passwd chintu

-- Switch to the new user: su chintu (enter the password for chintu).

File and Directory Management

1. Create a Directory:

-- Command: sudo mkdir demo_1 (creates a new directory named demo_1).

2. Remove a Directory:

-- Command: sudo rmdir demo_1 (removes an empty directory).

3. Create a File:

-- Command: touch demo_1.txt (creates an empty file named demo_1.txt).

4. Write to a File:

-- Command: echo "my name" > cgoyal.txt (writes "my name" to cgoyal.txt).

-- To enter content interactively, use cat > cgoyal.txt and press Ctrl + D to exit.

5. Edit a File:

-- Use a text editor like vi: vi cgoyal.txt (opens cgoyal.txt for editing).

6. Remove Files:

1) Command: sudo rm *.txt (removes all .txt files in the current directory).

Day – 25 July, 2024

Training Day - 8 Report:

Managing AWS EC2 Instances: Snapshots and Volumes

Definitions:

Volume: A volume in AWS is a durable block storage device that can be attached to an EC2 instance. It is used to store data persistently, even when the instance is stopped or terminated. Volumes are typically used for storing the operating system, application data, and other files that need to be retained.

Snapshot: A snapshot is a point-in-time backup of an EBS (Elastic Block Store) volume. It captures the data stored in the volume at a specific moment, allowing you to restore the volume to that state later. Snapshots are stored in Amazon S3 and can be used to create new volumes or to back up existing volumes.

Steps to Manage EC2 Instances, Volumes, and Snapshots:

1. Create an EC2 Instance: Launch a new instance in the AWS cloud.
2. Access the Instance: Connect to the instance via terminal using SSH.
3. Start Apache HTTP Server: Run the command to start and enable the Apache web server (httpd).
4. Change Ownership: Use the chown command to change file ownership on the instance.
5. Upload Website: Use FileZilla to upload your website files to the instance.
6. Create a Snapshot: In the EC2 dashboard, select the instance, go to the snapshot section, and create a snapshot of the volume.
7. Monitor Snapshot Status: Wait for the snapshot creation to complete and check its status.
8. Create Volume from Snapshot: Use the snapshot to create a new volume.
9. Attach Volume to New Instance: Select the new instance and attach the newly created volume, giving it a name and disabling the automatic IP option.

10. Detach Volume from Running Instance: If the volume is in use, stop the instance, detach the volume, and then attach it to the new instance.

11. Start the New Instance: Run the new instance with the attached volume to ensure the backup website runs successfully.

By following these steps, you can effectively manage your EC2 instances, create backups using snapshots, and utilize volumes for persistent storage.

26-July-2024

Training Day - 9 Report:

Setting Up Amazon Route 53 for Domain and DNS Management

What is Amazon Route 53?

Amazon Route 53 is a scalable and highly available Domain Name System (DNS) web service designed to provide reliable routing of end users to Internet applications by translating human-readable domain names (like www.example.com) into IP addresses. It helps improve the speed and reliability of applications by directing traffic to the nearest endpoint.

Why Use Route 53?

1. DNS Management: Route 53 simplifies the management of DNS records and provides a user-friendly interface for creating and managing DNS configurations.

2. Traffic Routing: It offers various routing policies (like latency-based routing, geolocation routing, etc.) to direct user traffic efficiently.

3. Health Checks: Route 53 can monitor the health of your application endpoints and route traffic away from unhealthy instances.

Types of Routing Policies in Route 53:

-- Simple Routing: Routes traffic to a single resource (e.g., an IP address).

-- Weighted Routing: Distributes traffic across multiple resources based on assigned weights.

-- Latency-based Routing: Directs traffic to the region with the lowest latency for the user.

-- Geolocation Routing: Routes traffic based on the geographic location of users.

-- Failover Routing: Automatically redirects traffic to a backup resource if the primary resource fails.

-- Multivalue Answer Routing: Returns multiple IP addresses in response to DNS queries, enabling load balancing.

Steps to Connect an IP Address to a Domain Using Route 53:

1) Access Route 53: Go to the AWS Management Console and search for Route 53 in the services menu.

2) Create a Hosted Zone: Click on "Create Hosted Zone," enter the domain name you purchased, and provide an optional description.

3) Select Type: Choose "Public" for the hosted zone type.

4) Create Record Set: Click "Create Record Set" to add DNS records.

5) Root Domain Record: Leave the root domain empty, enter the IP address in the value box, and click "Create Record."

6) Create CNAME Record: For the www subdomain, create a new record with the name "www," select CNAME as the type, and enter your domain name (e.g., chandan.com) as the value.

7) Update Domain Registrar: Go to the domain registrar where you purchased the domain, navigate to the DNS management section, and assign the AWS Route 53 name servers (NS) to your domain.

8) Final Steps: Confirm all settings are correct, and the domain should now point to your server.

29-July-2024

Training Report:

MySQL Installation Steps on Linux

What is a Database?

A database is an organized collection of structured information or data, typically stored electronically in a computer system. Databases are managed by Database Management Systems (DBMS), which allow for efficient data storage, retrieval, and manipulation.

Types of Databases:

1) Relational Databases: Use structured query language (SQL) for defining and manipulating data. Data is stored in tables with predefined relationships (e.g., MySQL, PostgreSQL, Oracle).

2) NoSQL Databases: Designed for unstructured data and can store data in various formats (e.g., document-based, key-value pairs). Examples include MongoDB, Cassandra, and Redis.

3) Object-oriented Databases: Store data in the form of objects, similar to objectoriented programming (e.g., db4o, ObjectDB).

4) Distributed Databases: Data is distributed across multiple locations, which can be on the same network or geographically dispersed (e.g., Google Cloud Spanner).

5) Cloud Databases: Databases that run on cloud computing platforms, offering scalability and flexibility (e.g., Amazon RDS, Azure SQL Database).

MySQL Overview:

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) for accessing and managing data. It is widely used for web applications and is known for its reliability, performance, and ease of use. MySQL supports various storage engines, allowing for different data handling methods, and is often used in conjunction with PHP and Apache in web development.

MySQL Installation Steps in Linux (Brief):

1. Update Package Index:

2. Install MySQL Server:

3. Check MySQL Service Status:

4. Secure MySQL Installation:

5. Access MySQL Command Line:

6. Exit MySQL Command Line:

7. Remove MySQL and Dependencies (if needed):

By following these steps, you can effectively install and manage MySQL on a Linux system.

30-july-2024

Training1 Report:

What is IIS Server?

IIS (Internet Information Services) is a web server software created by Microsoft for use with the Windows operating system. It is designed to host and serve web applications and websites, providing a platform for developers to build, manage, and deploy web content. IIS supports various protocols, including HTTP, HTTPS, FTP, FTPS, and more, allowing for flexible and secure web hosting.

Key Features of IIS:

1. Web Hosting: IIS allows for the hosting of static websites (HTML, CSS, JavaScript) and dynamic web applications built using technologies such as ASP.NET.

2. Security: IIS includes built-in security features such as authentication, authorization, and SSL/TLS for secure data transmission.

3. Scalability: It can handle a large number of concurrent users and can be configured to scale with the needs of the application.

4. Management Tools: IIS provides a user-friendly management interface, allowing administrators to configure settings, monitor performance, and manage applications easily.

5. Application Support: IIS supports various application frameworks, including ASP.NET, PHP, and others, enabling developers to create a wide range of web applications.

6. Logging and Diagnostics: It offers robust logging capabilities for tracking user activity and diagnosing issues, helping maintain optimal performance.

7. Extensibility: IIS can be extended with modules and plugins, allowing for additional functionalities and customizations based on specific needs.

Overall, IIS is a powerful and versatile web server solution widely used in enterprise environments for hosting websites and applications.

Introduction od RDP?

A remote desktop client is a software application that allows users to connect to and control another computer over a network or the internet. This enables users to access their desktop environment, applications, and files from a different location, as if they were sitting in front of the remote machine.

Why We use Remote Desktop Clients?

Remote desktop clients are utilized for various reasons, including:

1. Remote Access: They allow users to access their work or home computers from anywhere, facilitating remote work and flexibility.

2. Technical Support: IT professionals can use remote desktop clients to troubleshoot and resolve issues on users' machines without needing to be physically present.

3. Resource Management: Users can access applications and files stored on a remote computer, enabling them to utilize resources that may not be available on their local devices.

4. Collaboration: Remote desktop clients can facilitate collaboration among team members by allowing them to share screens and work together on projects in realtime.

5. Security: Many remote desktop solutions offer secure connections, ensuring that data

transmitted between the client and the remote machine is encrypted and protected from unauthorized access.

6. Cost-Effectiveness: Organizations can reduce costs by allowing employees to work from home or other locations without needing additional hardware or software.

Steps to Backup and Restore Data Using AWS Instance with Remote Desktop Connection

1. Set Up Remote Desktop Connection

-- Connect to your AWS Windows instance using Remote Desktop Protocol (RDP).

2. Prepare and Attach a New Volume

-- Go to AWS Console → EC2 → Volumes.

-- Create a new volume with a desired size in the same Availability Zone as the instance.

-- Attach the volume to your instance, specifying an available device name.

3. Initialize the New Volume in Windows

-- Open the Remote Desktop window for your instance.

-- Go to This PC → Manage or search for Server Manager.

-- In Server Manager, go to File and Storage Services → Disks.

-- Find the new volume (it should be offline initially); set it to Online.

4. Format the Volume and Allocate Space

-- Create a new volume, allocate space, and complete the setup.

-- Close Server Manager and verify the new volume in This PC.

-- Create a folder on the new volume (e.g., on D: drive, not C:).

5. Create a Snapshot of the Volume

-- In the AWS Console → EC2→ Volumes, select the new volume.

-- Create a snapshot to back up the volume’s current state.

6. Detach the New Volume

-- Detach the volume from the instance (e.g., 100GB volume that was attached).

7. Restore Data from Snapshot on a New Instance

-- In AWS Console → Snapshots, create a new volume from the snapshot.

-- Attach this volume to the new instance.

8. Verify Data on the New Instance

-- Open Remote Desktop on the new instance.

-- Go to This PC → Server Manager → File and Storage Services→ Disks.

-- Confirm the restored data is visible and accessible on the new instance.

31-july-2024

Training Report:

What is xfreedp?

xfreerdp is an open-source command-line client for the Remote Desktop Protocol (RDP) on Linux. It allows users to connect to and control Windows systems from Linux machines. It is especially useful for remote access in headless or lightweight Linux setups, without needing a full graphical desktop client.

Why Use xfreerdp in Linux?

-- Cross-platform access: Provides Linux users the ability to access and manage Windows machines.

-- Lightweight: Command-line-based, consuming fewer resources than graphical clients.

-- Customizable: Supports various options for configuring resolution, sound, and clipboard sharing.

Installation Steps

After installation, you can use xfreerdp to initiate an RDP connection. For example:

Steps to Launch a Windows Instance in AWS:

1. Log in to AWS Console: Go to the AWS Management Console and sign in.
2. Navigate to EC2 Service: In the AWS Console, go to Services → EC2.
3. Launch Instance: Click on Launch Instances to start the setup process.
4. Choose an Amazon Machine Image (AMI): Select a Windows Server AMI (e.g., Windows Server 2019 Base).
5. Select Instance Type: Choose an instance type (e.g., t2.micro for free tier or higher based on needs).
6. Configure Instance Details: Adjust settings like number of instances, network, and IAM role (leave as default if unsure).

7. Add Storage: Set the storage size and type for the root volume or add extra volumes if needed.

8. Configure Security Group: Add a rule to allow RDP (port 3389) for remote access; restrict IP access if desired for security.

9. Review and Launch: Review settings and click Launch.

10. Create or Select a Key Pair: Choose or create a key pair for instance access, and download the key file (.pem).

11. Launch the Instance: Click Launch Instances to start it.

12. Connect to the Instance: After it's running, click Connect → RDP Client → Get Password (using the .pem file) to connect via Remote Desktop.

Steps to Connect to an AWS Windows Instance Using xfreerdp on Linux:

To connect to an AWS Windows instance from Linux using xfreerdp, follow these steps:

1. Retrieve the Instance's Public IP and Password:

o Get the Public IP of your instance from the EC2 Console.

o Decrypt the instance password using the .pem key.

2. Connect Using xfreerdp:

1-Aug-2024

Training Report:

Step-by-Step Guide for XAMPP Installation, HTML File Upload, and Database Access

1. XAMPP Installation

-- Download the XAMPP installer from the Apache Friends website.

-- Run the installer and follow the prompts, selecting components (Apache, MySQL, PHP) and choosing the installation directory (e.g., C:\xampp).

-- Complete the installation by clicking “Next” until finished.

2. Setting Up IIS Server (Optional)

-- Open Server Manager, click on "Add roles and features," and proceed with the wizard.

-- Select "Web Server (IIS)" under Server Roles and complete the installation process.

3. Uploading HTML File

-- Open Microsoft Edge to access your server.

-- Extract your HTML file.

-- Copy the extracted files to C:\inetpub\wwwroot (for IIS) or C:\xampp\htdocs (for XAMPP).

-- Obtain your server's IP address from the instance settings.

4. Configuring Apache in XAMPP

-- Install Visual Studio Code if not already installed.

-- Navigate to C:\xampp\apache\conf\httpd.conf and open it in VS Code.

-- Modify line 285 (or similar) to change the DocumentRoot to your project folder.

5. Accessing MySQL (MariaDB)

Method 1 (Command Line):

-- Open Command Prompt: cmd

-- Navigate to MySQL directory: cd C:\xampp\mysql\bin

-- Connect to MySQL: MySQL.exe -uroot -p (Press Enter)

Method 2 (XAMPP Shell):

-- Open XAMPP Control Panel and click on “Shell.”

-- Type: mysql -uroot -p (Press Enter)

Method 3 (Direct Access):

-- Navigate to C:\xampp\mysql\bin in Command Prompt.

-- Run: MySQL.exe -uroot -p (Press Enter)

Method 4 (phpMyAdmin):

-- Open XAMPP Control Panel.

-- Click on “Admin” next to MySQL to access phpMyAdmin.

6. Creating and Accessing a Registration Form

-- Create a new file named index.php in the C:\xampp\htdocs directory.

-- Add the following HTML and PHP code to index.php:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Registration Form</title>
 <style>
 body { font-family: Arial, sans-serif; }
 form { margin-bottom: 20px; }
 input { margin: 5px 0; }
 </style>
</head>
<body>
 <h2>Registration Form</h2>
```

```
<form method="POST">
Name: <input type="text" name="name" required><br>
Email: <input type="email" name="email" required><br>
Mobile: <input type="text" name="mobile" required><br>
<input type="submit" value="Submit">
</form>
<?php

if ($_SERVER["REQUEST_METHOD"] == "POST") {
$conn = new mysqli("localhost", "root", "", "your_database_name");
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
$name = $_POST['name'];
$email = $_POST['email'];
$mobile = $_POST['mobile'];
$sql = "INSERT INTO users (name, email, mobile) VALUES ('$name', '$email',
'$mobile')";
if ($conn->query($sql) === TRUE) {
echo "New record created successfully";
} else {
echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn->close();
}
?>
<h3>Registered Users:</h3>
```

```
<ul>
<?php
$conn = new mysqli("localhost", "root", "", "your_database_name");
$result = $conn->query("SELECT name, email, mobile FROM users");
while ($row = $result->fetch_assoc()) {

echo "<li>{$row['name']} - {$row['email']} - {$row['mobile']}</li>";
}
$conn->close();
?>
</ul>
</body>
</html>
```

7. Testing the Setup

-- Open your web browser and navigate to http://localhost/index.php to access the registration form.

-- Fill in the form and submit to test

2-Aug-2024

Training Report:

Step-by-Step Guide for XAMPP Installation and Creating a Registration Form

1. Create a New Windows Instance

Set up a new Windows instance on your preferred cloud platform or local machine.

2. Install XAMPP

-- Download the XAMPP installer from the Apache Friends website and run it.

3. Save Template in htdocs

-- During installation, specify the installation directory (e.g., C:\xampp) and ensure to save your project files in the htdocs folder (e.g., C:\xampp\htdocs).

4. Create index.php for Registration Form

-- In the C:\xampp\htdocs directory, create a new file named index.php.

5. Add HTML and PHP Code to index.php

-- Insert the following code into index.php to create a registration form that captures name, email, and mobile number, and displays the submitted data:

6. Disable Windows Firewall Defender

-- Open the Run dialog (Windows + R), type Firewall.cpl, and press Enter to open Windows Firewall settings.

-- Turn off Windows Defender Firewall for both private and public networks.

7. Access Localhost

-- Open a web browser and navigate to http://localhost/index.php to access your registration form.

5-Aug-2024

Training5 Report:

Step-by-Step Guide to Upload a Website Online Using Apache Server on Ubuntu

1. Install Apache2

-- Open the terminal on your Ubuntu server and run the following command to install Apache2:

-- This command installs the Apache web server. By default, Apache starts automatically after installation.

2. Access the Apache Server

-- Copy the public IP address of your instance and paste it into your web browser. You should see the Apache2 default welcome page, indicating that the server is running correctly.

3. Navigate to the Web Directory

-- In the terminal, change the directory to the default web root for Apache:

-- This is where you will upload your website files.

4. Install PHP

-- To enable PHP support for your web applications, install PHP by running:

-- This command installs PHP and its necessary modules.

5. Install MySQL

-- Next, install the MySQL server to manage your databases:

6. Access MySQL

-- To access the MySQL prompt, use the following command:

-- Alternatively, you can use:

7. Set MySQL Root Password

-- Once logged in to MySQL, set a password for the root user by executing:

8. Create a Database

-- Show existing databases and create a new database for your application:

-- Replace registration_db with your desired database name.

9. Use the Database

-- Select the database you just created:

-- You can now create tables or import data into this database.

10. Upload Files Using FileZilla

-- Open FileZilla and connect to your server using your .pem key file:

1. Host: Your server's IP address

2. Username: ubuntu

3. Authentication: Use your .pem file

-- Upload your SQL file to the /var/www/html directory through FileZilla.

11. Change Ownership of the Web Directory

-- After uploading your files, change the ownership of the /var/www/html directory to ensure that the ubuntu user has the necessary permissions:

-- This command recursively changes the ownership of the directory.

12. Import the SQL File

-- To import the SQL file into your database, run the following command in the terminal:

mysql -h localhost -u root -p registration_db < /var/www/html/registration_db.sql

-- This command imports the SQL file into the registration_db database.

13. Verify Database Content

-- To check the contents of your database, log in to MySQL again and run:

-- This will display the tables and data within the users table.

14. Upload PHP Files

-- Using FileZilla, upload your PHP files to the /var/www/html directory. Ensure that your PHP files are correctly placed and configured to connect to your database.

15. Configure Apache Virtual Host

-- Finally, change the configuration for your website by editing the 000-default.conf file:

-- Modify this file to set the DocumentRoot or any other configurations necessary for

your site.

6-Aug-2024

Training6 Report:

Step-by-Step Guide to Manage MySQL Users and Configure PHP on Apache Server

1. Check User Information in MySQL

-- To view the current users and their hosts, run the following command in the MySQL prompt:

2. View User Authentication Details

-- To see more detailed information about users, including their authentication strings, execute:

3. Check User Authentication Plugin

-- To view the authentication plugin used by each user, run:

4. Create a New MySQL User

-- To create a new user named chandan with a password, use the following command:

5. Grant Privileges to the New User

-- To grant all privileges on the registration_db database to the new user, execute:

6. Flush Privileges

-- To ensure that MySQL recognizes the changes made to user privileges, run:

-- This command clears any cached privileges and reloads the user permissions.

7. Change Database Credentials in PHP File

-- Open your PHP file using vim or any text editor to update the database username and password for the connection. For example:

-- Change the credentials in the connection string to reflect the new user chandan and the password '123'.

8. Check Apache Error Logs

-- To monitor the Apache error logs in real-time for any issues, use the following

command:This will display the latest error messages, helping you troubleshoot any problems with your

web application.

7-Aug-2024

Training7 Report:

Define Firewall

-- Firewall: Protects a network by controlling incoming and outgoing traffic based on

predetermined security rules. Requires knowledge of port numbers to control access.

OSI (Open System Interconnection) Model

-- Application Layer: Handles human-computer interaction, where applications access

network services.

-- Presentation Layer: Ensures data is in a usable format and handles encryption.

-- Session Layer: Maintains and controls sessions and ports.

-- Transport Layer: Transmits data using transmission protocols, TCP and UDP.

-- Network Layer: Determines the path for data transmission.

-- Data Link Layer: Defines the data format for transmission over the network.

-- Physical Layer: Deals with hardware (processor, RAM, HDD).

Protocols

-- Protocols: A set of rules and regulations governing data exchange.

o Application, Presentation, Session Layers: Associated with port numbers.

o Transport Layer: Involves TCP/IP and UDP protocols.

o UDP: Used for small queries (e.g., DNS).

o TCP: Used for larger work/queries or requests.

DNS in AWS (Route 53)

-- AWS Route 53: Manages DNS, mapping domain names to IP addresses.

-- Domain Name Website: Paste DNS information to link a domain name to AWS.

Ports and Their Functions

-- SSH: Port 22

-- MySQL: Port 3306

-- HTTPS: Port 443

-- SMTP: Port 25

-- HTTP: Port 80

-- POP3: Port 995

-- RDP: Port 3389

-- FTP: Port 21 (data) and 20 (command)

-- POP2: Port 109

-- DNS: Port 53

Role of Port Numbers in Firewalls

-- Port numbers help block or allow traffic through the firewall by controlling access based on protocol and destination.

Network Layer and Data Link Layer

-- Network Layer: Uses IP addresses to route data.

-- Data Link and Physical Layer: Handles network connections via WiFi, LAN, or WAN.

Client-Server Communication

-- Example: Client (IP and port 192.168.1.1:3306) sends a request to Server.

Ping of Death Attack

-- A DOS (Denial of Service) attack where excessive ping requests overload the network, particularly through UDP protocol due to its small request size.

8-Aug-2024

Training8 Report:

Topics in Firewall

a. What is a Firewall?

b. Firewalld service in Linux

c. Enable/disable Firewall

d. How to see the existing firewall rules

e. Adding & deleting Firewall Rules

f. Adding/Removing Ports

g. Block incoming/outgoing Traffic

h. Block ICMP

Types of Firewall:

1. Software Firewall: A software firewall is installed on individual devices, such as computers or servers, and monitors incoming and outgoing traffic at the software level. It's more flexible and often suitable for individual or small network protection.

2. Hardware Firewall: A hardware firewall, on the other hand, is a physical device placed between a network and external traffic sources, providing a strong barrier for larger networks. It offers robust protection for enterprise environments, managing traffic before it reaches internal devices.

Make a Linux instance in AWS and access through xfreedp in linux:

1. Basic Setup and Initial Configuration

Installing Apache with Firewall Protection:

Install Apache

sudo apt install apache2 -y

Install firewalld

sudo apt install firewalld -y

Start and enable firewalld

sudo systemctl start firewalld

sudo systemctl enable firewalld

Verify installations

apache2 -v

firewall-cmd --version

2. Common Firewall Scenarios

Scenario 1: Setting Up a Web Server

Allow HTTP and HTTPS

sudo firewall-cmd --zone=public --add-service=http --permanent

sudo firewall-cmd --zone=public --add-service=https --permanent

sudo firewall-cmd --reload

Verify configuration

curl localhost

Scenario 2: Database Server Setup

Allow MySQL

sudo firewall-cmd --zone=public --add-port=3306/tcp --permanent

sudo firewall-cmd --reload

Test MySQL connection

telnet localhost 3306

3. Advanced Service Management

Working with Custom Services

Create new service

sudo firewall-cmd --new-service=myapp --permanent

Configure service

sudo firewall-cmd --service=myapp --add-port=8080/tcp --permanent

sudo firewall-cmd --zone=public --add-service=myapp --permanent

Multiple Port Configuration

Add port range

sudo firewall-cmd --zone=public --add-port=4000-4100/tcp --permanent

Add multiple individual ports

sudo firewall-cmd --zone=public --add-port=8080/tcp --add-port=8443/tcp --permanent

4. Security Hardening Examples

Blocking Suspicious IP Addresses

Block single IP

sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="10.10.10

.10" reject'

Block IP range

sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="192.168.

1.0/24" reject'

Rate Limiting

Limit SSH connections

sudo firewall-cmd --permanent --add-rich-rule='rule service name="ssh" accept limit value="

3/m"'

5. Real-World Applications

Setting Up a LAMP Stack

Allow required services

sudo firewall-cmd --zone=public --add-service=http --permanent

sudo firewall-cmd --zone=public --add-service=https --permanent

sudo firewall-cmd --zone=public --add-port=3306/tcp --permanent

sudo firewall-cmd --reload

Verify configuration

sudo firewall-cmd --list-all

Securing Remote Access

Configure SSH access

sudo firewall-cmd --zone=public --add-service=ssh --permanent

Allow specific IP for SSH

sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="trusted-i

p" service name="ssh" accept'

6. Monitoring and Troubleshooting

Real-time Monitoring

Watch active connections

watch -n1 "sudo firewall-cmd --list-connections"

Monitor rejected packets

sudo tail -f /var/log/messages | grep REJECT

Common Issues and Solutions

Problem 1: Service Not Accessible

Check if service is allowed

sudo firewall-cmd --list-services

Verify port is open

sudo firewall-cmd --list-ports

Test port locally

nc -zv localhost <port>

Problem 2: Website Blocking Not Working

Clear DNS cache

sudo systemd-resolve --flush-caches

Verify rich rules

sudo firewall-cmd --list-rich-rules

Test blocking

ping blocked-website.com

7. Advanced Configurations

Custom Zone Configuration

Create custom zone

sudo firewall-cmd --permanent --new-zone=customzone

Configure zone

sudo firewall-cmd --zone=customzone --add-service=http --permanent

sudo firewall-cmd --zone=customzone --add-source=192.168.1.0/24 --permanent

Time-Based Rules

Allow service during business hours

sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4" service name="http" acce

pt time start="9:00" end="17:00"'

8. Backup and Recovery

Backup Configuration

Export configuration

sudo firewall-cmd --runtime-to-permanent

sudo cp -r /etc/firewalld /etc/firewalld.bak

Restore configuration

sudo cp -r /etc/firewalld.bak/* /etc/firewalld/

sudo systemctl restart firewalld

9. Regular Maintenance Tasks

Weekly Maintenance Checklist

1. Check firewall status

sudo firewall-cmd --state

2. Review active rules

sudo firewall-cmd --list-all

3. Check logs for suspicious activity

sudo journalctl -u firewalld --since "24 hours ago"

4. Verify services

sudo firewall-cmd --list-services

5. Update firewall

sudo apt update

sudo apt upgrade firewalld

10. Performance Optimization

Optimize Rule Processing

Remove redundant rules

sudo firewall-cmd --remove-rich-rules='rule family="ipv4" source address="0.0.0.0/0" accept

'

Organize rules by frequency

Most used rules should be at the top

sudo firewall-cmd --permanent --add-rich-rule='rule priority=1 service name="http" accept'

9-Aug-2024

Training9 Report:

Comprehensive Firewall Management Guide Continue

apt install apache2

service apache2 status

apt install firewalld –y

ping gndec.ac.in

For Block a website temporary :

firewall-cmd --direct --add-rule ipv4 filter OUTPUT 0 -d 175.176.187.102 -j DROP

To check list which website has been blocked :

iptables -t filter -L -v -n

For UnBlock a website temporary:

firewall-cmd --direct --remove-rule ipv4 filter OUTPUT 0 -d 175.176.187.102 -j DROP

OR

firewall-cmd –reload

For Block a website Permanent :

firewall-cmd --direct --remove-rule ipv4 filter OUTPUT 0 -d 175.176.187.102 -j DROP

12-Aug-2024

Training Day - 20 Report:

Remote Access Guide for Ubuntu Linux

Steps to Remotely Access One Ubuntu PC from Another

1) Ping Google to Check Connectivity:

2) Install Curl:

3) Test Curl with Google

4) Switch to Super User

5) Update Package List

6) Install Firewalld

7) Install Net Tools

8) Install OpenSSH Server

9) Check SSH Status

10)Identify Network Type

-- If the ping is successful, both systems are on the same network.

-- If not, they are on different networks.

11) Check Current User

12) SSH Into Another Ubuntu Machine

13)Enter Password For the Remote Machine

14) Optional Add a Rich Rule to Firewall

-- firewall-cmd --zone=public --add-rich-rule='rule family="ipv4" destination

address="175.176.187.102" reject' --permanent

15)List Current Rich Rules

13-Aug-2024

Training Day - 21 Report:

Amazon VPC: Purpose and Benefits in Steps

What is Amazon VPC?

Definition: Amazon Virtual Private Cloud (VPC) is a service that allows you to create a logically isolated virtual network within the AWS cloud.

Why Use Amazon VPC?

1. Enhanced Security: Provides a secure environment by isolating resources from other AWS users.

2. Flexibility and Scalability: Allows customization of network architecture to meet specific needs and easily scale resources.

3. Cost-Effective: No additional charges for creating a VPC; only pay for specific components like NAT gateways.

4. Integration with AWS Services: Seamlessly integrates with other AWS services (e.g., RDS, ELB) while maintaining network control.

Key Features of Amazon VPC:

-- Isolation and Control: Complete control over your virtual network, including IP address range and subnet configuration.

-- Customizable Network Configuration: Create public and private subnets tailored to your application needs.

-- Security: Utilize security groups and network ACLs to manage traffic, with VPC flow logs for monitoring.

-- Connectivity Options: Connect to the internet via Internet Gateway or to onpremises networks using VPN; supports VPC peering.

Steps to Create a VPC and Launch Instances in AWS:

1) Search for VPC: In the AWS Management Console, search for "VPC" and select it.

2) Create VPC: Click on "Create VPC".

3) Configure VPC: Select "VPC only", enter a name, and specify the IPv4 CIDR block, then click "Create VPC".

4) Select VPC: In the left-hand sidebar, select the VPC you just created.

5) Access Subnets: In the left-hand sidebar, click on "Subnets".

6) Create Subnet: Click on "Create Subnet", select the VPC ID of your created VPC.

7) Configure Subnet: Enter the CIDR block, select an availability zone, enter a name, and click "Create Subnet".

8) Access Internet Gateway: In the left-hand sidebar, click on "Internet Gateways" and then click "Create Internet Gateway".

9) Name Internet Gateway: Enter a name for the internet gateway and click "Create Internet Gateway".

10) Attach Internet Gateway: Click on "Attach to VPC", select your VPC, and click "Attach Internet Gateway".

11) Access Route Tables: In the left-hand sidebar, select "Route Tables" and click "Create Route Table".

12) Configure Route Table: Enter a name, select your VPC, and click "Create Route Table".

13)Edit Route Table: In the route table, click on "Subnet Associations", then "Edit subnet associations", select your subnet, and click "Save".

14) Access Security Groups: In the left-hand sidebar, click on "Security Groups" and click "Create Security Group".

15) Configure Security Group: Enter a name, select your VPC, and click "Create

Security Group".

16) Add Rules: Add a rule for "All Traffic" and select "Custom" for the source security group; add another rule for "RDP" with the source set to "Anywhere", then save the rules.

17)Launch Instances: Create multiple Windows instances, selecting your VPC, enabling automatic IP assignment, and choosing the existing security group, then click "Launch Instance".

18) Name Instances: Assign different names to the instances, ensuring one is named "Server".

19) Connect to Instances: Connect to the instances to access the machines.

20) Adjust Firewall Settings: Turn off firewall settings on the instances.

21) Ping Instances: Ping the IP addresses of all instances to ensure connectivity.

22) Set Up Web Server: On one instance, search for "Server Manager", select "Add Features and Roles", then proceed through the prompts to select "Web Server (IIS)"

and click "Next" until done.

23) Access Server IP: Use the server's IP address to access the website from different PCs or instances.

14-Aug-2024

Training Day - 22 Report:

What is an S3 Bucket in AWS?

-- Definition: An S3 bucket is a container used to store objects in Amazon Simple Storage Service (S3), which is a scalable cloud storage solution.

-- Purpose: Buckets serve as a repository for data, allowing users to store, retrieve, and manage various types of files such as documents, images, and videos.

-- Unique Naming: Each S3 bucket must have a globally unique name, ensuring that no two buckets across all AWS accounts can share the same name.

-- Access Control: Users can configure access permissions for their buckets using policies and access control lists (ACLs), providing security and control over who can access the data.

-- Features: S3 buckets support features like versioning, lifecycle management, and different storage classes to optimize cost and performance based on usage needs.

-- Use Cases: Common use cases for S3 buckets include data backup, content distribution, and hosting static websites, making them versatile for various applications in the cloud.

Steps to Create an S3 Bucket for Static Website Hosting and Link with Route 53 in AWS

Step 1: Create an S3 Bucket

1) Search for S3: In the AWS Management Console, search for "S3" and select it.

2) Create a Bucket: Click on "Create bucket".

3) Enter Bucket Name: Provide a unique bucket name (e.g., school-website) which should match your domain name.

4) Configure ACLs: Under "Block Public Access settings for this bucket", uncheck

"Block all public access".

5) Acknowledge Settings: Check the acknowledgment box and click "Create bucket".

Step 2: Enable Static Website Hosting

1) Access Bucket Properties: Click on the bucket you just created to open its settings.

2) Scroll to Properties: Navigate to the "Properties" tab.

3) Enable Static Website Hosting: Scroll down and click the "Edit" button.

-- Configure Hosting Settings:

-- Select "Enable" for Static Website Hosting

-- Enter the main file name (e.g., index.html).

4) If applicable, enter an error file name (e.g., error.html).

5) Save Changes: Click "Save changes".

Step 3: Set Bucket Policy for Public Access

1) Go to Permissions: Click on the "Permissions" tab.

2) Edit Bucket Policy: Click on "Bucket Policy" and edit the policy.

3) Copy and Modify Policy: Use the following policy template, replacing BucketName with your actual bucket name:

4) Save Changes: Click "Save changes".

Step 4: Upload Website Files

1) Upload Files: Click on the "Upload" button in the bucket.

2) Select Files: Upload all your website files and folders one by one.

3) Complete Upload: After selecting all files, click the "Upload" button.

Step 5: Access Your Website

1) Copy Website Link: Go to the "Properties" tab, scroll down to the "Static website

hosting" section, and copy the endpoint link.

2) Test Your Website: Paste the link in your browser to ensure the website is online.

Step 6: Configure Route 53 for Domain

1. Search for Route 53: In the AWS Management Console, search for "Route 53" and select it.

2. Create Hosted Zone: Click on "Hosted zones" and then "Create hosted zone".

3. Enter Domain Name: Provide your domain name (e.g., example.com) and click "Create hosted zone".

4. Create Record: Click on "Create record".

5. Configure Alias Settings:

-- Set "Alias" to "Yes".

-- Choose the "Endpoint" as "S3 Website".

-- Select the S3 website you created earlier.

6. Save Changes: Click "Create records" to save the settings.

16-Aug-2024

Training Day - 23 Report:

Introduction

AWS Identity and Access Management (IAM) allows you to manage access to AWS services and resources securely. With IAM, you can create multiple users and groups, assign different permissions, and ensure that users have the access they need to perform their jobs while following the principle of least privilege. This document outlines the steps to create IAM users, groups, and policies to manage permissions effectively.

Why Use IAM?

-- Security: IAM helps to secure your AWS environment by allowing you to control who can access your resources.

-- Granular Permissions: You can assign specific permissions to users or groups based on their job roles.

-- Management: Easily manage access for multiple users and groups from a single console.

-- Audit and Compliance: IAM provides features to monitor user activity and maintain compliance with security policies.

Steps to Create IAM Users and Groups

Step 1: Sign in to the AWS Management Console

1. Log In: Sign in to the AWS Management Console using your root user credentials.

Step 2: Create a Normal User

1. Access IAM: In the AWS Management Console, search for "IAM" and select it.

2. Create User: Click on "Users" in the left sidebar and then click the "Add user"

button.

3. Enter User Name: Input a username for the new user and click the "Next: Permissions" button.

Step 3: Create a User Group

1. Create Group: Click on the "Create group" button to create a new group.

2. Enter Group Name: Provide a name for the group (e.g., "RDS-Users") and click "Create group".

Step 4: Assign User to Group

1. Select Group: In the "Set permissions" step, select the group you just created.

2. Click Next: Click the "Next: Tags" button.

Step 5: Review and Create User

1. Review Settings: Review the user details and permissions.

2. Create User: Click the "Create user" button to finalize the process.

Step 6: Create a Policy

1. Access Policies: In the IAM dashboard, click on "Policies" in the left sidebar.

2. Create Policy: Click the "Create policy" button.

3. Select Services: Choose the services you want to grant permissions for (e.g., select "RDS" to give RDS permissions).

4. Set Permissions: Choose "All actions" to grant all permissions for the selected service.

5. Next and Review: Click "Next: Tags", then "Next: Review".

6. Create Policy: Provide a name for the policy (e.g., "RDS-Full-Access") and click "Create policy".

Step 7: Attach Policy to User Group

1. Go to User Groups: Return to the "User groups" section in IAM.

2. Select Group: Click on the group you created earlier.

3. Attach Policy: Click on the "Permissions" tab and then "Attach policies".

4. Search and Attach: Search for the policy you created (e.g., "RDS-Full-Access") and attach it to the group.

Step 8: Enable Console Access for Users

1. Return to Users: Go back to the "Users" section.

2. Select User: Click on the username of the user you created.

3. Enable Console Access: Click on the "Security credentials" tab and select "Manage" next to "Console password"

4. Set Custom Password: Choose "Custom password" and enter a password for the user. Enable console access.

5. Save Changes: Click "Save changes".

Step 9: Share Access Details

1. Copy Access Link: Copy the link provided for console access.

2. Send to Users: Share the link along with the username and password with the user(s) for access.

20-Aug-2024

Training Day - 24 Report:

What is DynamoDB?

1) Amazon DynamoDB: DynamoDB is a fully managed NoSQL database service provided by AWS that offers fast and predictable performance with seamless scalability.

2) Key-Value and Document Store: It supports both key-value and document data structures, making it versatile for various applications.

Why Use DynamoDB?

-- Scalability: DynamoDB automatically scales up or down to adjust for capacity and maintain performance.

-- Performance: It provides low-latency responses, making it suitable for applications that require real-time data access.

-- Fully Managed: As a managed service, it eliminates the administrative burden of operating and scaling a distributed database.

-- Flexible Data Model: The schema-less design allows for storing diverse data types and structures.

-- Integrated with AWS Services: It easily integrates with other AWS services, enabling powerful application architectures.

Getting Started with Amazon DynamoDB: A Step-by-Step Guide for Database Management and Integration with PHP

Step 1: Access DynamoDB

-- Search for DynamoDB: In the AWS Management Console, search for "DynamoDB" and select it.

Step 2: Create a Table

-- Create Table: Click on "Create table".

-- Enter Table Name: Provide a name for your table (e.g., "College").

-- Define Partition Key: Specify the partition key for the table and click "Create table".

Step 3: Add Items to the Table

-- Select Table: Click on the table you created (e.g., "College").

-- Create Item: In the "Actions" dropdown, click on "Create item".

-- Add Attributes: Add new attributes as needed by entering the attribute name and value.

Step 4: Explore Items

-- View Entries: Navigate to "Items" in the left sidebar to see the entry data for your table.

Step 5: Edit PHP Code for Connection

-- Open PHP File: In your code editor (e.g., VS Code), open the PHP file where you want to connect to DynamoDB.

-- Edit Code: Update the PHP code to include the necessary AWS SDK for PHP:

Step 6: Install AWS SDK for PHP

-- Open Terminal: In the VS Code terminal, run the following command to install the AWS SDK for PHP:

21-Aug-2024

Training Day - 25 Report:

Comprehensive Guide to Load Balancers in AWS: Purpose, Types, and Benefits

Understanding Load Balancers in AWS

A load balancer in AWS is a service that distributes incoming application traffic across multiple targets, such as EC2 instances, to ensure high availability and reliability.

Why Use a Load Balancer?

1) High Availability: Ensures that applications remain accessible even in the event of server failures by rerouting traffic to healthy instances.

2) Scalability: Automatically adjusts to varying traffic loads, allowing applications to handle increased demand without performance degradation.

3) Fault Tolerance: Monitors the health of registered targets and only routes traffic to those that are healthy, minimizing downtime.

4) Security: Provides an additional layer of security by hiding the backend instances from direct access and can integrate with AWS WAF for web application security.

Types of Load Balancers in AWS

1. Application Load Balancer (ALB):

-- Operates at Layer 7 (Application Layer) and is ideal for HTTP/HTTPS traffic.

-- Supports advanced routing features based on content, making it suitable for microservices architectures.

2. Network Load Balancer (NLB):

-- Operates at Layer 4 (Transport Layer) and is designed for high-throughput and lowlatency applications.

-- Best for TCP and UDP traffic, handling millions of requests per second.

3. Classic Load Balancer (CLB):

-- The original load balancer that operates at both Layer 4 and Layer 7.

-- Suitable for applications built within the EC2-Classic network but lacks some advanced features of ALB and NLB.

4. Gateway Load Balancer (GWLB):

-- Combines a transparent network gateway with a load balancer.

-- Ideal for deploying third-party virtual appliances, such as firewalls, in a scalable manner.

22-Aug-2024

Training Day - 26 Report:

Site-to-Site VPN in AWS: Steps to Set Up with Virtual Private Gateway

Overview

A Site-to-Site VPN in AWS allows you to securely connect your on-premises network or data center (Customer Gateway) to your AWS Virtual Private Cloud (VPC) through a Virtual Private Gateway. This connection enables secure communication between your local network and AWS resources.

Components

-- Customer Gateway (CGW): Represents your on-premises VPN device (VPN server).

-- Virtual Private Gateway (VGW): The VPN concentrator on the AWS side that connects your VPC to the VPN.

-- Transit Gateway: A service that enables you to connect multiple VPCs and onpremises networks through a central hub.

Steps to Set Up Site-to-Site VPN

1. Create a VPC:

Log in to the AWS Management Console.

Navigate to the VPC dashboard.

Click on "Create VPC" and configure the required settings (CIDR block, name, etc.).

2. Launch an EC2 Instance (Optional):

-- If needed, launch an EC2 instance within the created VPC for further testing or application deployment.

3. Create a Customer Gateway:

-- Go to the VPC dashboard and select "Customer Gateways."

-- Click on "Create Customer Gateway."

-- Enter the public IP address of your on-premises VPN device (CGW).

-- Specify the routing type (static or dynamic) based on your network setup.

4. Create a Virtual Private Gateway:

-- In the VPC dashboard, select "Virtual Private Gateways."

-- Click on "Create Virtual Private Gateway."

-- Provide a name and create the gateway.

5. Attach the Virtual Private Gateway to the VPC:

-- After creating the VGW, select it and click on "Attach to VPC."

-- Choose the VPC you created earlier and attach the VGW.

6. Create a Site-to-Site VPN Connection:

-- In the VPC dashboard, navigate to "VPN Connections."

-- Click on "Create VPN Connection."

-- Select the Virtual Private Gateway you created and the Customer Gateway you set up.

-- Configure the tunnel options (encryption, routing, etc.) and create the connection.

7. Update Routing Tables:

-- Go to the VPC dashboard and select "Route Tables."

-- Choose the route table associated with your VPC.

-- Add a route to direct traffic destined for your on-premises network through the Virtual Private Gateway.

8. Configure the On-Premises VPN Device:

-- Download the VPN configuration file provided by AWS.

-- Use this configuration to set up your on-premises VPN device (CGW) to establish the VPN connection.

9. Testing the Connection:

-- Once configured, test the VPN connection by pinging resources in the VPC from your on-premises network.

-- Monitor the connection status in the AWS Management Console.

10. Using Transit Gateway (Optional):

-- If you have multiple VPCs, consider creating a Transit Gateway.

-- Attach your VPCs and Customer Gateways to the Transit Gateway for centralized management and routing.

23-Aug-2024

Training Day - 27 Report:

AWS CloudFront: Overview and Implementation Steps

Introduction to AWS CloudFront

AWS CloudFront is a content delivery network (CDN) service that provides a way to deliver data, videos, applications, and APIs to users globally with low latency and high transfer speeds. It achieves this by caching content at edge locations close to the users, ensuring faster access and reduced load on the origin server.

How CloudFront Works?

1. Edge Locations: CloudFront uses a network of edge locations around the world. When a user requests content, the request is routed to the nearest edge location.

2. Data Retrieval: If the content is cached at the edge location, it is served directly to the user. If not, CloudFront fetches the content from the origin server (e.g., an EC2 instance) and caches it for future requests.

3. Latency Reduction: By serving content from the nearest edge location, CloudFront

minimizes latency, improving the performance of websites, especially for ecommerce, streaming, and blogs.

4. SEO Benefits: Faster response times can enhance user experience, leading to better

SEO rankings as search engines favor quick-loading websites.

Practical Steps to Set Up CloudFront with an EC2 Instance

Step 1: Create an EC2 Instance

-- Launch an EC2 instance that will serve as your origin server.

Step 2: Create a CloudFront Distribution

-- Navigate to the CloudFront console and create a new distribution.

-- Enter the DNS URL of your EC2 instance as the origin.

Step 3: Select Protocol

-- Choose the appropriate protocol for your distribution (HTTP/HTTPS).

Step 4: Create a Cache Policy

-- Define a cache policy by entering a name and setting the TTL (Time to Live) for how

long the content should be cached at the edge location.

Step 5: Create the Distribution

-- Click "Create Distribution" and wait for it to be deployed.

Step 6: Monitor Last Modified Status

-- The first request will fetch data from the origin server to the edge location, allowing

subsequent requests to be served from CloudFront.

Problem: Direct Access to EC2 Instance

Solution: Restrict Access to CloudFront Only

Find CloudFront IP Addresses:

-- Use the AWS-managed prefix list to find the IP addresses associated with CloudFront.

Modify EC2 Security Group:

-- Go to the security group settings of your EC2 instance.

-- Remove the existing HTTP (port 80) rule.

-- Create a new rule allowing HTTP access only from the CloudFront IP prefix list.

Invalidate CloudFront Cache:

-- To ensure users receive the latest content, perform an invalidation in CloudFront by specifying the paths to delete or using /* to clear all cached content.

Reload CloudFront DNS:

-- After making these changes, reload your CloudFront distribution to ensure it serves the latest content.

Important Notes

-- Always ensure that any updates made to your EC2 instance or load balancer are reflected in CloudFront by performing cache invalidation.

-- Regularly check and update the security group rules to maintain access control and security.

By following these steps, you can effectively utilize AWS CloudFront to enhance the performance and security of your web applications.

27-Aug-2024

Training Day - 28 Report:

Using AWS Simple Email Service (SES) to Send Emails

Overview

AWS Simple Email Service (SES) allows users to send and receive emails securely and reliably. Below are the steps to set up and send emails using AWS SES.

Steps to Use AWS SES

1. Access AWS SES:

Log in to the AWS Management Console and search for "SES."

2. Create Email Identity:

-- Click on "Identities" and then select "Create Identity."

-- Choose "Email Address" and enter the email address you want to verify.

-- Click "Create Identity."

3. Verify Email Address:

-- Check your inbox for a verification email from AWS SES.

-- Click the verification link in the email to confirm your address.

-- Refresh the SES console to see the verified status.

4. Send a Test Email:

a. Click on "Send Test Email" in the SES console.

b. Fill in the required information (recipient, subject, and body).

c. Click "Send."

5. Check Spam Folder:

If the email ends up in the spam folder, note that this can happen with unverified senders.

6. Request Production Access:

-- Go to the Account Dashboard and click on "View the Getting Started Page."

-- Request production access by selecting the mail type, entering your website URL, and providing a description of your email use case.

-- Specify the email addresses that will receive emails and submit the request.

7. Final Notes:

-- Ensure that you have created and verified an email identity.

-- Understand that initially, you can only send emails to verified addresses until you receive production access.

By following these steps, you can effectively set up and use AWS SES to send emails from your applications.

28-Aug-2024

Training Day - 29 Report:

Using AWS CLI: Configuration and Instance Creation Guide

This document outlines the steps to install and configure the AWS Command Line Interface (CLI), create a new user, and launch an AWS EC2 instance using the CLI.

1. Installing the AWS CLI

For Windows:

1. Download the AWS CLI MSI installer from here.

2. Run the installer and follow the on-screen instructions.

3. Verify the installation by opening Command Prompt and running:

For Linux:

1. Download the AWS CLI installation package:

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

2. Unzip the downloaded file:

3. Install the AWS CLI:

4. Verify the installation:

2. Configuring AWS CLI

To configure the AWS CLI, you need to set up your access keys and default region.

1) Create a new IAM user:

-- Go to the AWS Management Console.

-- Navigate to IAM (Identity and Access Management).

-- Click on Users and then Add user.

-- Provide a username and select Programmatic access.

-- Set permissions by adding the user to a group or attaching policies directly.

2) Create access keys:

-- After creating the user, go to the User Security Credentials tab.

-- Scroll down to Access keys and click on Create access key.

-- Select CLI, check the acknowledgment box, and proceed.

-- Download the CSV file containing your access key and secret key.

3) Configure the AWS CLI: Open your terminal and run:

Enter the following information when prompted:

i. AWS Access Key ID

ii. AWS Secret Access Key

iii. Default region name (e.g., ap-south-1)

iv. Default output format (e.g., json)

3. Creating an EC2 Instance Using AWS CLI

1. Describe available instances: To view existing instances, use:

2. Run a new EC2 instance: Use the following command to launch an EC2 instance.

Replace <ami-id> with the actual AMI ID you wish to use:

aws ec2 run-instances --image-id <ami-id> --instance-type t2.micro --region ap-south-1

29-Aug-2024

Training Day - 30 Report:

Taken some tests

30-Aug-2024

Training Day - 31 Report:

Taken some Test

2-Sep-2024

Training Day - 32 Report:

CCNA (Cisco Certified Network Associate)

What is Network?

A computer network can be described as a system of interconnected devices that can communicate using some common standards called the Internet protocol suite or TCP/IP. These devices communicate to exchange network resources, such as files and printers, and network services.

Types of Computer Networks

Listed below are the most common types of computer networks:

1) Local Area Network (LAN) – LANs are commonly used in small to medium size companies, households, buildings, etc., with limited space.

2) Personal Area Network (PAN) – PAN covers a short distance of 10 meters. Bluetooth is an example of PAN.

3) Metropolitan Area Network (MAN) – MANs are used in a single geographic region, such as a city or town.

4) Wide Area Network (WAN) – WANs cover larger areas like different states and countries.

5) Wireless Local Area Network (WLAN) – Wireless LAN is used for wireless networks, connecting wired and wireless devices.

LAB 1: ONE ROUTER ONE SWITCH THREE PC

ROUTER: A router is a networking device that directs data packets between different networks, such as between a local home network (LAN) and the internet (WAN).

Routers can also assign IP addresses, manage traffic within a network, routers help devices

communicate across different networks efficiently and securely.

SWITCH: A switch is a networking device that connects multiple devices within a local area network (LAN) and directs data to the specific device using MAC addresses.

PC: A PC (Personal Computer) is a general-purpose computing device designed for individual use. It typically consists of hardware like a central processing unit (CPU), memory (RAM), storage (hard drive or SSD), and input/output devices (keyboard, mouse, monitor). PCs run operating systems like Windows, macOS, or Linux, allowing users to perform tasks such as browsing the internet, running software applications, and storing data.

Step 1: Open Cisco Packet Tracer

Launch Cisco Packet Tracer on your computer.

Step 2: Add Devices

1. Router:

In the left sidebar, click on the Router icon (the router icon) and drag a router (e.g., Cisco 2901) onto the workspace.

2. Switch:

In the same sidebar, click on the Switch icon and drag a Switch 2960 or any other model) onto the workspace.

3. PCs:

Click on the End Devices icon (the computer icon) and drag 3 PCs onto the workspace.

Step 3: Connect Devices

1. Router to Switch:

-- Click on the Connections icon (the lightning bolt) in the lower-left corner of Packet Tracer.

-- Choose the Copper Straight-Through Cable. (This cable is used for different devices.)

-- Connect one end of the cable to any Ethernet port on the router

-- (FastEthernet0/0) and the other end to any port on the switch

-- (FastEthernet0/1).

2. PCs to Switch:

Use the Copper Straight-Through Cable again to connect each PC to the switch.

For example:

-- PC1 to FastEthernet0/2 on the switch.

-- PC2 to FastEthernet0/3.

-- PC3 to FastEthernet0/4.

Step 4: Assign IP Addresses to PCs

1. Click on PCs one by one.

2. Go to the Desktop tab and select IP Configuration.

3. Assign an IP address with a subnet mask:

-- PC1: IP 1.1.1.2, SUBNET MASK: 255.0.0.0

-- PC2: IP 1.1.1.3, SUBNET MASK: 255.0.0.0

-- PC3: IP 1.1.1.4, SUBNET MASK: 255.0.0.0

Step 5: Configure the Router (Optional for Routing)

If you want to add routing to the router (for example, to enable the router to communicate with the PCs), follow these steps:

1. Click on the Router and go to the CLI tab.

2. Enter the following commands to set up the router:

-- Router> enable: The enable command is used to move from user EXEC mode to privileged EXEC mode on a Cisco router or switch.

-- Router# configure terminal: This command allows you to enter global configuration mode on a Cisco router or switch.

-- Router(config)# interface fa0/0: This command selects the Fast Ethernet 0/0

interface for configuration. It allows you to configure settings like IP addresses, interface status, and other parameters specific to this interface.

-- Router(config-if) # Ip address 1.1.1.1 255.0.0.0: Ip address: This command is used to configure the IP address and subnet mask for an interface.

-- 1.1.1.1: This is the IP address that you're assigning to the interface. In this case, the address.

-- 1.1.1.1 is part of the Class A address space, which traditionally uses a 255.0.0.0 subnet mask.

-- Router(config-if) # no shutdown: This command enables the interface, bringing it out of the shutdown state and allowing it to begin functioning and passing traffic.

-- shutdown is a command used to disable the interface, and when you use no shutdown, you're effectively reversing that action and enabling the interface.

-- Router(config-if) # exit: it is used to exit the interface configuration mode and return to global configuration mode

-- Router(config)# exit: it is used to exit global configuration mode and return to privileged EXEC mode on a Cisco router or switch.

Step 6: Verify Connectivity (Ping Test)

1. On each PC, open the Command Prompt from the Desktop tab.

2. Run a ping test to check the network connectivity:

-- From PC1, ping `1.1.1.1` (the router's IP address).

-- From PC2 and PC3, ping the other PCs (e.g., `ping 1.1.1.3`, `1.1.1.4`).

-- This setup enables basic connectivity between PCs through the switch and router, allowing for communication within the same network.

How To Save Router Configuration commands:

o Router# show running-config

o Router# copy running-config startup-config

o How To Remove Router & Interface Configuration:

o Router# erase startup-config

o Router # reload: The router will prompt to save the current running configuration before reloading. Since you've erased the startup config and want to remove the configuration.

2-Sep-2024

Training Day - 33 Report:

Performed In Lab

04-Sep-2024

Training Day - 34 Report:

LAB 2: STATIC ROUTING

To implement static routing as shown in the diagram, follow these steps:

1. Identify Networks:

-- Network 1: 192.168.1.0/24 (connected to Router R1)

-- Network 2: 192.168.2.0/24 (connected to Router R2)

-- Network 3: 192.168.3.0/24 (connected to Router R3)

-- WAN Links:

-- Link between R1 and R2: 12.0.0.0/24

-- Link between R2 and R3: 23.0.0.0/24

2. Configuring Static Routes: (Always give networked)

1) On Router R1:

- Configure the default route to reach network 192.168.2.0/24 and 192.168.3.0/24 via R2:bash

-- ip route 192.168.2.0 255.255.255.0 12.0.0.2

-- ip route 192.168.3.0 255.255.255.0 12.0.0.2

2) On Router R2:

- Configure routes to reach 192.168.1.0/24 via R1 and 192.168.3.0/24 via R3:bash

-- ip route 192.168.1.0 255.255.255.0 12.0.0.1

-- ip route 192.168.3.0 255.255.255.0 23.0.0.2

3) On Router R3:

- Configure the route to reach network 192.168.1.0/24 and 192.168.2.0/24 via

R2:bash

-- ip route 192.168.1.0 255.255.255.0 23.0.0.1

-- ip route 192.168.2.0 255.255.255.0 23.0.0.1

3. Configure Interfaces: You need to assign the correct IP addresses to the interfaces on

each router as per the diagram:

On Router R1:

-- interface Fa0/0

-- ip address 12.0.0.1 255.255.255.0

-- no shutdown

-- interface Fa0/1

-- ip address 192.168.1.1 255.255.255.0

-- no shutdown

On Router R2:

-- interface Fa0/0

-- ip address 12.0.0.2 255.255.255.0

-- no shutdown

-- interface Fa0/1

-- ip address 23.0.0.1 255.255.255.0

-- no shutdown

-- interface Fa0/3

-- ip address 192.168.2.1 255.255.255.0

-- no shutdown

On Router R3:

-- interface Fa0/0

-- ip address 23.0.0.2 255.255.255.0

-- no shutdown

-- interface Fa0/1

-- ip address 192.168.3.1 255.255.255.0

-- no shutdown

The commands you mentioned are used to configure static routes on Router R1. Here's a detailed explanation of each command:

-- ip route 192.168.2.0 255.255.255.0 12.0.0.2

-- ip route: This is the command used to create a static route.

-- 192.168.2.0: This is the destination network that Router R1 needs to send traffic to (Network 2 in the diagram).

-- 255.255.255.0: This is the subnet mask for the destination network, indicating that it's a /24 network.

-- 12.0.0.2: This is the next-hop IP address, which is the IP address of the interface on Router R2 (the router directly connected to R1 on the 12.0.0.0/24 network).

This command tells Router R1 that to reach the 192.168.2.0/24 network (connected to Router R2), it should forward the traffic to 12.0.0.2 (Router R2).

-- ip route 192.168.3.0 255.255.255.0 12.0.0.2

-- ip route: Again, this is the command to add a static route.

-- 192.168.3.0: This is the destination network that Router R1 needs to send traffic to (Network 3 in the diagram).

-- 255.255.255.0: This is the subnet mask for the destination network, indicating it's a /24 network.

-- 12.0.0.2: This is the next-hop IP address, which is Router R2's interface in the 12.0.0.0/24 network.

This command tells Router R1 that to reach the 192.168.3.0/24 network (connected to Router R3), it should forward the traffic to 12.0.0.2 (Router R2).

Router R2 will then forward the traffic to Router R3.

192.168.2.0/24 (its own local network) and 192.168.3.0/24 (via Router R3).

Key Concepts:

-- Next hop is typically the IP address of a router interface directly connected to the current router.

-- A router doesn't need to know the entire path to the destination, just the IP address of the next device (next hop) that is part of the route toward the destination.

Example:

In your previous configuration:

-- ip route 192.168.2.0 255.255.255.0 12.0.0.2

-- Destination Network: 192.168.2.0/24 (Network 2, located behind Router R2).

-- Next-Hop IP: 12.0.0.2 (This is the IP address of Router R2’s interface that is directly connected to Router R1).

-- So, when Router R1 needs to send packets to the 192.168.2.0/24 network, it forwards those packets to 12.0.0.2 (the next-hop address). Router R2 then takes over and

-- forwards the packet to the correct destination.

Visualizing the Next-Hop:

1) If Router R1 wants to send traffic to a destination that is not directly connected to it, it forwards the packet to the next-hop router (Router R2), which is responsible for forwarding the packet further down the network path.

2) This concept is essential in static routing (where routes are manually defined) as well as in dynamic routing protocols (where routers learn paths to destinations

automatically).

05-Sep-2024

Training Day - 35 Report:

LAB 3: DYANMIC ROUTING USING RIP

To configure RIP v2 (Routing Information Protocol version 2) for the network topology shown in the image, you can follow these steps for each router in your network.

Step 1: Access Each Router's Configuration Mode

Connect to each router and enter the global configuration mode:

Router> enable

Router# configure terminal

Step 2: Enable RIP v2 on Each Router

For each router (R1, R2, and R3), you need to enable RIP and specify the networks that will participate in RIP.

Configuration for R1:

-- R1(config)# router rip

-- R1(config-router)# version 2

-- R1(config-router)# no auto-summary

-- R1(config-router)# network 12.0.0.0

-- R1(config-router)# network 192.168.1.0

-- R1(config-router)# exit

Configuration for R2:

-- R2(config)# router rip

-- R2(config-router)# version 2

-- R2(config-router)# no auto-summary

-- R2(config-router)# network 12.0.0.0

-- R2(config-router)# network 23.0.0.0

-- R2(config-router)# network 192.168.2.0

-- R2(config-router)# exit

Configuration for R3:

-- R3(config)# router rip

-- R3(config-router)# version 2

-- R3(config-router)# no auto-summary

-- R3(config-router)# network 23.0.0.0

-- R3(config-router)# network 192.168.3.0

-- R3(config-router)# exit

Step 3: Save the Configuration

After configuring RIP on all routers, save the configuration to ensure it persists after a reboot.

Router# write memory

Step 4: Verify the Configuration

You can verify that RIP is working and that the routers are exchanging routing information with the following commands:

Router# show ip route

Router# show ip protocols

Explanation of the Configuration:

 version 2: Specifies that the router should use RIP version 2.

 no auto-summary: Disables automatic summarization of networks, allowing RIP to advertise subnet information.

 network [network-address]: Specifies the networks that will be included in the RIP routing process.

By following these steps on all routers in the network, RIP v2 will be set up, allowing the routers to share routing information dynamically. Each router will learn about other networks

connected through the other routers and route traffic accordingly

6-Sep-2024

Training Day-36 Report:

OSPF (Open Shortest Path First):

1. Scalability: OSPF is more scalable and suitable for larger networks compared to RIP. It's widely used in real-world networks.

2. Link-State Protocol: OSPF uses a link-state routing algorithm, which can give you a deeper understanding of how routing decisions are made compared to the distance-vector approach of RIP.

3. Widely Applicable: Knowledge of OSPF is often required for network certifications and is valuable for understanding many enterprise network environments.

Once you're comfortable with OSPF, you might then explore EIGRP to understand Cisco's advanced routing protocol, and eventually move on to BGP for a broader perspective on internet-scale routing.

When it's said that knowledge of OSPF is often required for network certifications and is valuable for understanding many enterprise network environments, it means:

1. Network Certifications: OSPF is frequently covered in networking certification exams, such as the Cisco CCNA (Cisco Certified Network Associate) and CCNP (Cisco Certified Network Professional) exams. These certifications are often pursued by IT professionals to validate their skills and knowledge in network management and routing.

2. Enterprise Networks: Many businesses and organisations use OSPF in their internal network infrastructure due to its efficiency and scalability. Understanding OSPF helps in managing and troubleshooting these networks effectively.

In summary, OSPF is a key concept in both professional certifications and real-world network management, making it a crucial area of knowledge for anyone pursuing a career in networking.

What is link-state routing algorithm?

Link-State Routing Algorithm

Definition: A link-state routing algorithm is a type of routing protocol that builds a complete map of the network by gathering information about the state of each network link. Each router in the network uses this map to compute the best path to each destination.

Key Concepts:

1. Network Map: Each router creates a link-state database (LSDB) that contains information about the network topology, including the state of all links (e.g., up or down) and the cost associated with each link.
2. Link-State Advertisements (LSAs): Routers periodically send out link-state advertisements to inform other routers about the state of their links. This information is used to update the LSDB.
3. Dijkstra's Algorithm: Once each router has a complete map of the network, it uses Dijkstra's Shortest Path First (SPF) algorithm to calculate the shortest path to each destination based on the link costs.
4. Convergence: Link-state protocols generally converge faster than distance-vector protocols because routers update their LSDBs immediately when there is a change in the network, leading to a quicker recalculation of routes.

Advantages:

1. Scalability: Suitable for larger networks due to its efficient use of network resources and faster convergence.
2. Accuracy: Provides a more accurate view of the network topology compared to distance-vector protocols.
3. Reduced Routing Loops: Because each router has a complete view of the network, routing loops are less likely.

Example Protocols: OSPF (Open Shortest Path First) and IS-IS (Intermediate System to

Intermediate System) are common examples of link-state routing protocols.

In summary, link-state routing algorithms help routers understand the entire network topology and calculate optimal routing paths, leading to more efficient and reliable network operation.

LAB 4: OSPF ROUTING

Step-by-Step OSPF Configuration

1. Configure OSPF on Router0 (Chandigarh)

1. Enter global configuration mode:

Router0> enable

Router0# configure terminal

2. Start the OSPF process and assign an OSPF process ID (we'll use 1):

Router0(config)# router ospf 1

3. Assign networks to OSPF area 1:

-- Fa0/0 (192.168.1.0/24)

-- Fa1/0 (10.1.1.0/30)

-- Fa0/2 (23.1.2.0/30)

Router0(config-router)# network 192.168.1.0 0.0.0.255 area 1

Router0(config-router)# network 10.1.1.0 0.0.0.3 area 1

Router0(config-router)# network 23.1.2.0 0.0.0.3 area 1

1. Exit OSPF configuration:

Router0(config-router)# exit

2. Configure OSPF on Router1 (Delhi)

1. Enter global configuration mode:

Router1> enable

Router1# configure terminal

2. Start the OSPF process:

Router1(config)# router ospf 1

3. Assign networks to OSPF area 1:

o Fa0/0 (172.136.4.0/30)

o Fa0/1 (192.168.2.0/24)

o Fa0/2 (17.5.3.0/30)

Router1(config-router)# network 172.136.4.0 0.0.0.3 area 1

Router1(config-router)# network 192.168.2.0 0.0.0.255 area 1

Router1(config-router)# network 17.5.3.0 0.0.0.3 area 1

4. Exit OSPF configuration:

Router1(config-router)# exit

3. Configure OSPF on Router2 (Kurukshetra)

1. Enter global configuration mode:

Router2> enable

Router2# configure terminal

2. Start the OSPF process:

Router2(config)# router ospf 1

3. Assign networks to OSPF area 1:

o Fa0/0 (23.1.2.0/30)

o Fa1/0 (17.5.3.0/30)

Router2(config-router)# network 23.1.2.0 0.0.0.3 area 1

Router2(config-router)# network 17.5.3.0 0.0.0.3 area 1

4. Exit OSPF configuration:

Router2(config-router)# exit

4. Configure OSPF on Router3 (Panipat)

1. Enter global configuration mode:

Router3> enable

Router3# configure terminal

2. Start the OSPF process:

Router3(config)# router ospf 1

3. Assign networks to OSPF area 1:

o Fa0/0 (172.136.4.0/30)

o Fa1/0 (10.1.1.0/30)

Router3(config-router)# network 172.136.4.0 0.0.0.3 area 1

Router3(config-router)# network 10.1.1.0 0.0.0.3 area 1

4. Exit OSPF configuration:

Router3(config-router)# exit

5. Verify OSPF Configuration

On each router, use the following command to verify that OSPF is working and neighbors are established:

1. Check OSPF neighbors:

Router# show ip ospf neighbor

2. Check the OSPF routing table:

Router# show ip route ospf

9-Sep-2024

Training Day - 37 Report:

BGP (Border Gateway Protocol)

1) It is Routing Protocol.

2) It is Exterior Routing Protocol (Means Autonomous System connect outside).(Rip & ospf is a interior routing protocol which is used inside the company to best path to communicate).

3) It provides Routing between Autonomous System (Big Organization different organization To Connect)

4) ISP (Internet Service Provider Also Used this protocol) in National Level or Regional Level that's why it is also called routing protocol of internet.

Who use BGP?

-- ISP

-- VERY big Organisations can use BGP

-- Having two or more internet Connections have in organization then to communicate we use BGP

-- Thats why it is called Multi Homing

BGP (Border Gateway Protocol) History

1. Before BGP there is Another Protocol EGP (Exterior Gateway Protocol) but in 1994 BGP v4 Replace the EGP Since 1994 we are using BGP protocol for Exterior Routing Protocol.

2. It also Supports CIDR means classes or internet domain protocols.

Type of BGP Routing Protocol

a) it is Neither link state or Distance Vector Protocol.

b) It is called path vector routing Protocol.

Routing Decision are made Based on

-- Path

-- Network Polices

-- Rules

Metric

1) Very complex & big

2) Composite Metric

3) Tenable with Attributes

If you are not Tenable there attribute according to the polices by default it will be based on Distance vector Protocol, but difference is that there is no of Hops there is a No of

Autonomous are there

BGP (Border Gateway Protocol) Types:

i) Internal BGP (IBGP): - are those BGP Neighbours that belongs to the same AS & These neighbours needn't to be directly connected. (if they are very close to each other then also connected)

ii) External BGP (EBGP): - are those BGP Neighbours that Belongs to the Different AS Neighbours are need to be directly connected

BGP (Border Gateway Protocol)

1) BGP Peers and Peering

2) BGP Autonomous System

1) BGP Peers and Peering:

1) BGP neighbours: When BGP Router exchange routes with another BGP speaking Device, it is called BGP peer or BGP peering.

2) it is established with the help of by Manuel Conjurations.

2) BGP Autonomous System

1) Group of Routers

2) Share Similar Routing Protocols

3) Operate within a single administrative domain.

4) Typically Belong to one Organisation

5) AS Number can be between 1 to 65,535

LAB 5: BGP (Border Gateway Protocol)

To implement BGP (Border Gateway Protocol) according to the topology in your diagram, you'll need to configure BGP on the routers in Autonomous System (AS) 20, AS 30, and AS 35.

Here's an outline of the BGP configuration steps for each router based on your diagram:

1. Router0 (AS 20) Configuration

-- Router ID: Set the router's BGP process to AS 20.

-- Neighbor: Establish BGP neighbor relationships with Router1 (AS 30) and Router2 (AS 35).

-- Networks: Advertise the local network 192.168.10.0/24 and 10.10.10.0/24.

```
Router(config)# router bgp 20
Router(config-router)# neighbor 10.10.10.2 remote-as 35
Router(config-router)# neighbor 192.168.10.1 remote-as 20
Router(config-router)# network 192.168.10.0 mask 255.255.255.0
Router(config-router)# network 10.10.10.0 mask 255.255.255.0
```

2. Router1 (AS 30) Conguration

o Router ID: Set the router's BGP process to AS 30.

o Neighbor: Establish a BGP neighbor relationship with Router0 (AS 20).

o Networks: Advertise the local network 192.168.20.0/24 and 20.20.20.0/24

```
Router1(config)# router bgp 30
Router1(config-router)# bgp router-id 1.1.1.1 (Compulsory for all router)
Router1(config-router)# neighbor 20.20.20.1 remote-as 35
Router1(config-router)# network 192.168.20.0 mask 255.255.255.0
```

3. Router2 (AS 35) Conguration

o Router ID: Set the router's BGP process to AS 35.

o Neighbor: Establish BGP neighbor relationships with Router0 (AS 20) and Router1 (AS 30).

o Networks: Advertise the local network 10.10.10.0/24 and 20.20.20.0/24.

```
Router2(config)# router bgp 35
Router2(config-router)# neighbor 10.10.10.1 remote-as 20
Router2(config-router)# neighbor 20.20.20.2 remote-as 30
Router2(config-router)# network 10.10.10.0 mask 255.255.255.0
```

Router2(config-router)# network 20.20.20.0 mask 255.255.255.0

Verification Commands

After configuring BGP, verify the BGP session using these commands on each router:

-- Router# show ip bgp summary

-- Router# show ip bgp neighbors

-- Router# show ip route bgp

These commands will confirm that the BGP neighbors are established and that routes are being advertised and received correctly.

By configuring BGP in this manner, Router0, Router1, and Router2 will be able to communicate across the different Autonomous Systems

10-Sep-2024

Training Day - 38 Report:

lab

11-Sep-2024

Training Day - 39 Report:

HSRP (Hot Standby Routing Protocol)

In CCNA have only HSRP, VRRP & GLBP is remove

LAB 6: HSRP (Hot Standby Routing Protocol)

To implement HSRP (Hot Standby Router Protocol) for your network setup, we need to configure the routers to provide gateway redundancy. Here's how you can configure HSRP on Cisco routers, assuming this is a Packet Tracer or similar topology.

The two routers (Router1 and Router2) in your topology will act as the redundant routers for the clients connected to the switch. I'll provide a basic configuration for both routers.

Steps to Configure HSRP on Router1 and Router2:

1. Configure HSRP on Router1:

Assume the virtual IP will be 192.168.2.254 for the 192.168.2.x network and 192.168.3.254 for the 192.168.3.x network.

-- Router1> enable

-- Router1# configure terminal

-- Router1(config)# interface fa0/5 # Assuming fa0/5 is connected to the

-- 192.168.2.x network

-- Router1(config-if)# ip address 192.168.2.1 255.255.255.0

-- Router1(config-if)# standby 1 ip 192.168.2.254

-- Router1(config-if)# standby 1 priority 110 # Assign a higher priority to Router1

-- Router1(config-if)# standby 1 pre-empt # Enable pre-emption so Router1 takes

-- over if it becomes active again

-- Router1(config-if)# standby 1 version 2 # Use HSRP version 2 for enhanced

-- functionality

-- Router1(config-if)# no shutdown

-- Router1(config)# interface fa0/6 # Assuming fa0/6 is

-- connected to the 192.168.3.x network

-- Router1(config-if)# ip address 192.168.3.1 255.255.255.0

-- Router1(config-if)# standby 2 ip 192.168.3.254

-- Router1(config-if)# standby 2 priority 110

-- Router1(config-if)# standby 2 preempt

-- Router1(config-if)# standby 2 version 2

-- Router1(config-if)# no shutdown

-- Router1(config-if)# exitRouter2> enable

Configure HSRP on Router2:

-- Router2> enable

-- Router2# configure terminal

-- Router2(config)# interface fa0/5 # Assuming fa0/5 is connected to the 192.168.2.x

network

-- Router2(config-if)# ip address 192.168.2.2 255.255.255.0

-- Router2(config-if)# standby 1 ip 192.168.2.254

-- Router2(config-if)# standby 1 priority 100 # Lower priority than Router1

-- Router2(config-if)# standby 1 preempt

-- Router2(config-if)# standby 1 version 2

-- Router2(config-if)# no shutdown

-- Router2(config-if)# exit

-- Router2# write memory

-- Router2(config)# interface fa0/6 # Assuming fa0/6 is connected to the

-- 192.168.3.x network

-- Router2(config-if)# ip address 192.168.3.2 255.255.255.0

-- Router2(config-if)# standby 2 ip 192.168.3.254

-- Router2(config-if)# standby 2 priority 100

-- Router2(config-if)# standby 2 preempt

-- Router2(config-if)# standby 2 version 2

-- Router2(config-if)# no shutdown

-- Router2(config-if)# exit

-- Router2(config)# exit

Verify the Configuration:

After configuring HSRP, you can verify the status using the following commands:

Router1# show standby brief

Router2# show standby brief

This will display which router is currently the active router and which one is in standby

mode.

Key Points:

1. Virtual IP Address: Both routers will share a virtual IP (192.168.2.254 and 192.168.3.254) that clients will use as the default gateway.

2. Preemption: Allows a higher-priority router to take over when it comes back online.

3. Priority: Router1 is configured with a higher priority (110) to ensure it is the active router.

With this setup, your routers will provide gateway redundancy using HSRP, and clients on both networks will have uninterrupted service even if one router fails.

12-Sep-2024

Training Day - 40 Report:

HSRP (Hot Standby Routing Protocol) Explanation

HSR Configuring HSRP (Hot Standby Router Protocol) on Router1 involves several steps.

Let's break down the configuration, command by command, so you understand the purpose of each part:

Assumptions:

-- Router1 will act as the primary or active router for the two networks: 192.168.2.x and 192.168.3.x.

-- We will use HSRP group 1 for the 192.168.2.x network and HSRP group 2 for the 192.168.3.x network.

-- Virtual IP addresses:

-- 192.168.2.254 for the 192.168.2.x network (clients will use this as their gateway).

-- 192.168.3.254 for the 192.168.3.x network (clients will use this as their gateway).

Here is the detailed explanation for configuring Router1:

Router1> enable # Switch to privileged mode

Router1# configure terminal # Enter global configuration mode to apply settings

1. Enter Global Configuration Mode:

This allows you to start making configuration changes on Router1.

2. Configure the Interface for the 192.168.2.x Network:

We need to configure the interface that connects to the 192.168.2.x network (which is FastEthernet 0/5 in this case).

-- Router1(config)# interface fa0/5 # Enter interface configuration mode for FastEthernet 0/5P

-- Router1(config-if)# ip address 192.168.2.1 255.255.255.0 # Assign an IP to this interface

Here, we assign the IP address 192.168.2.1 with a subnet mask of 255.255.255.0 to the interface. This IP address will be the physical IP address of Router1 for the 192.168.2.x network.

3. Configure HSRP Group 1 for 192.168.2.x:

Router1(config-if)# standby 1 ip 192.168.2.254

This command configures HSRP group 1 on Router1. The virtual IP 192.168.2.254 is the shared gateway IP that clients will use on the 192.168.2.x network.

This IP address does not belong to any specific router but is used by the active router (Router1 or Router2).

4. Set Priority for Router1:

Router1(config-if)# standby 1 priority 110

HSRP uses a priority value to decide which router will be active. By default, the priority is 100. We set Router1's priority to 110 so it becomes the active router unless it fails.

5. Enable Pre-emption:

Router1(config-if)# standby 1 preempt

Pre-emption allows Router1 to take over as the active router if it has a higher priority and if it

comes back online after a failure. Without this command, if Router2 takes over as the active router (due to Router1 going down), Router1 won't automatically become the active router again when it comes back online, unless pre-emption is enabled.

6. Use HSRP Version 2:

Router1(config-if)# standby 1 version 2

This command congures HSRP version 2, which supports IPv6 and offers improved features over version 1. Version 2 uses a different multicast address (224.0.0.102) and increases the HSRP group numbers range from 0-255 to 0-4095.

7. Activate the Interface:

Router1(config-if)# no shutdown

The no shutdown command ensures that the interface is active (up) and can participate in the network.

8. Configure HSRP for the 192.168.3.x Network: Repeat the same process for the second interface that connects to the 192.168.3.x network (which is Fast Ethernet 0/6 in this case)

Router1(config)# interface fa0/6 # Enter interface configuration mode for Fast Ethernet 0/6

Router1(config-if)# ip address 192.168.3.1 255.255.255.0 # Assign an IP to this interface

9. Save the Configuration: Once you've configured both interfaces, save the configuration to make sure it persists after a reboot.

Router1(config)# exit

Router1# write memory

This saves the configuration to the router's NVRAM.

Summary of Key Commands:

Virtual IP: standby [group number] ip [virtual IP] sets the virtual IP address for the HSRP group.

Priority: standby [group number] priority [value] sets the priority to control which router is active.

Pre-emption: standby [group number] pre-empt enables pre-emption so a higherpriority router takes over if it comes back online.

HSRP Version 2: standby [group number] version 2 uses the newer version of HSRP

for enhanced functionality.

Activate Interface: no shutdown ensures the interface is active and operational. With

these steps, Router1 will act as the active router for the clients in the 192.168.2.x and

192.168.3.x networks, providing high availability and redundancy.

13-Sep-2024

Training Day - 41 Report:

Physically perform in Lab.

17-Aug-2024

Training Day - 42 Report:

LAB 7: DHCP (Dynamic Host Configuration Protocol):

DHCP Configuration on a Cisco Router:

Step 1: Access the Router's CLI

-- Connect to the router via console or remotely using Telnet/SSH.

-- Enter global configuration mode.

Router> enable

Router# configure terminal

Step 2: Exclude IP Addresses (Optional)

-- You can exclude a range of IP addresses that you don't want to assign dynamically.

This is useful for devices with static IPs (like servers or switches).

Router(config)# ip dhcp excluded-address 10.0.0.1 10.0.0.10

In this case, IP addresses from 10.0.0.1 to 10.0.0.10 won't be assigned by DHCP.

Step 3: Create a DHCP Pool

-- You need to define a DHCP pool. This is the range of IP addresses that will be assigned to clients.

Router(config)# ip dhcp pool info

In this example, info is the name of the DHCP pool (you can name it anything).

Step 4: Configure the Network and Subnet

-- Specify the network and subnet mask for the pool.

Router(dhcp-config)# network 10.0.0.0 255.0.0.0

This configures DHCP to assign IPs from the 10.0.0.0/8 network.

Step 5: Specify the Default Gateway (Router)

-- Define the default gateway that will be assigned to clients. This is typically the IP address of the router's interface connected to the network.

Router(dhcp-config)# default-router 10.0.0.1

Here, 10.0.0.1 is the default gateway for clients.

Step 6: Specify the DNS Server (Optional)

-- If needed, configure the DNS server for clients. You can use a public DNS server (e.g., Google’s 8.8.8.8) or a local DNS server.

Router(dhcp-config)# dns-server 8.8.8.8

Step 7: (Optional) Configure Lease Time

-- By default, the DHCP lease is infinite, but you can configure a lease time if needed (e.g., 1 day).

Router(dhcp-config)# lease 1

This sets the lease time to 1 day.

Step 8: Exit and Save Configuration

-- Exit DHCP configuration mode and save the changes.

Router(dhcp-config)# exit

Router(config)# exit

Router# write memory

Step 9: Verify the DHCP Configuration

-- Use the following command to verify the DHCP configuration:

Router# show ip dhcp pool

Router# show ip dhcp binding

18-Sep-2024

Training Day - 43 Report:

Lab done

19-Sep-2024

Training Day - 44 Report:

LAB 8: DHCP (Dynamic Host Configuration Protocol) & DNS Configuration

DHCP Configuration Steps:

Enter global configuration mode:

Router> enable

Router# configure terminal

Create a DHCP pool:

Router(config)# ip dhcp pool NETWORK-POOL

Configure the network and subnet mask (as shown in your CLI):

Router(dhcp-config)# network 10.0.0.0 255.0.0.0

Configure the default gateway (as shown in your CLI):

Router(dhcp-config)# default-router 10.0.0.1

Configure DNS server (assuming Server0 will be the DNS server):

Router(dhcp-config)# dns-server 192.168.1.4

Configure excluded addresses (to prevent DHCP from assigning router interfaces and

static IPs):

Router(config)# ip dhcp excluded-address 10.0.0.1 10.0.0.10

DNS Configuration Steps:

On Server0 (192.168.1.4): Copy- Install DNS server role/service

1) Create a new forward lookup zone for your domain

2) Configure reverse lookup zone for 10.0.0.0 network

3)Add A records for:

-- Router0 (10.0.0.1)

-- Router1 (10.0.0.2)

-- Other network devices as needed

On the DHCP router, verify the DNS configuration is properly pointing to Server0:

Router# show ip dhcp pool

Additional Configuration Tips:

On the interfaces that will be serving DHCP:

Router(config)# interface fa0/1

Router(config-if)# ip helper-address 192.168.1.4

Verify DHCP operation:

Router# show ip dhcp binding

Router# show ip dhcp server statistics

Test DNS resolution:

Router# ping dns-server-name

20-Sep-2024

Training Day - 45 Report:

Physically perform in Lab.

23-Sep-2024

Training Day - 46 Report:

LAB 9: WIRELESS NETWORK USING BASIC ROUTING

To configure a wireless network using a basic router, as shown in the diagram, follow these step-by-step instructions:

Components in the Diagram:

-- Wireless Router (WRT300N): IP address 192.168.0.1

-- Laptop-PT: IP address 192.168.0.2

-- PC-PT: IP address 192.168.0.3

Configuration Steps:

Step 1: Configure the Wireless Router (WRT300N)

1. Access the Router Configuration:

-- Connect to the router by entering its default IP address (`192.168.0.1`) in a web browser from either the laptop or the PC.

-- Log in using the default credentials (usually `admin/admin` for most routers, or refer to the router's manual).

2. Set the Wireless Network:

-- Navigate to the Wireless Settings or Wireless Setup section.

-- Enable wireless networking and set:

-- SSID (Network Name): Choose a name, e.g., Lab9_Wireless

-- Channel: Set to `Auto` or a specific channel to avoid interference.

-- Mode: Choose mixed mode if you have different types of devices.

-- Security: Enable WPA2-PSK encryption for better security.

-- Set a Wi-Fi password for the wireless network.

3. Set the Router's LAN Settings:

-- Go to LAN Settings.

-- Ensure the router's IP address is 192.168.0.1.

-- Subnet Mask should be 255.255.255.0.

-- Enable DHCP Server (if needed) and set the IP address range (e.g., `192.168.0.2

192.168.0.50`).

4. Save and Reboot:

-- Apply the changes and reboot the router if necessary.

Step 2: Configure the Laptop (Laptop-PT)

1. Connect to the Wireless Network:

-- On the laptop, open the wireless network settings.

-- Scan for available networks and select `Lab9_Wireless` (the SSID you set on the

router).

-- Enter the password you configured earlier.

2. Set the IP Address:

-- In the Network and Sharing Centre, go to Change adapter settings.

-- Right-click on the wireless adapter and go to Properties.

-- Select Internet Protocol Version 4 (TCP/IPv4) and click on Properties.

-- Set a Static IP address:

IP: 192.168.0.2

Subnet Mask: 255.255.255.0`

Default Gateway: 192.168.0.1 (the router's IP address)

Set the DNS server as the same as the router's IP (`192.168.0.1`) or any external

DNS server (e.g., `8.8.8.8` for Google DNS).

3. Test the Connection:

-- Open the Command Prompt and type `ping 192.168.0.1` to test connectivity to the

router.

-- You should receive replies, indicating a successful connection.

Step 3: Configure the PC (PC-PT)

1. Set the IP Address:

-- Go to the Network and Sharing Centre, open Change adapter settings, and rightclick on the Ethernet adapter.

-- Set the Static IP address for the PC:

IP: 192.168.0.3

Subnet Mask: 255.255.255.0

Default Gateway: 192.168.0.1 (the router's IP)

DNS Server: 192.168.0.1 or 8.8.8.8.

2. Test the Connection:

-- Open the Command Prompt on the PC and type `ping 192.168.0.1` to check the

connection to the router.

Step 4: Test the Network

1. Test Communication Between Devices:

-- From the laptop, open Command Prompt and type `ping 192.168.0.3` to check

if the laptop can reach the PC.

-- From the PC, type `ping 192.168.0.2` to check if the PC can reach the laptop.

-- Both devices should be able to communicate with each other via the wireless

router.

24-Sep-2024

Training Day - 47 Report:

LAB 10: ACCESS POINT & PRINTER CONNECT WITH NETWORK

Components in the Diagram:

-- Router (2811):

Interfaces:

Fa0/0: 192.168.1.1

Fa0/1: 192.168.2.1

-- Switch (2960-24TT):

Connected to Router (Fa0/0) and Access Point (Fa0/2).

-- Access Point (AP): Connected to the switch.

-- Laptop-PT (Laptop0): 192.168.1.3

-- Laptop-PT (Laptop1): 192.168.1.4

-- Printer-PT: 192.168.1.5

-- Second Switch: Connected to the second subnet 192.168.2.0

-- Laptop-PT (Laptop2): 192.168.2.2

Configuration Steps:

Step 1: Configure the Router (2811)

1. Access the Router:

Connect a console cable to the router and access it through a terminal emulator (e.g.,

PuTTY, Tera Term).

2. Configure the Router Interfaces:

-- Enter global configuration mode:

-- Router> enable

-- Router# configure terminal

-- Fa0/0 (192.168.1.1):

-- Router(config)# interface Fa0/0

-- Router(config-if)# ip address 192.168.1.1 255.255.255.0

-- Router(config-if)# no shutdown

-- Fa0/1 (192.168.2.1):

-- Router(config)# interface Fa0/1

-- Router(config-if)# ip address 192.168.2.1 255.255.255.0

-- Router(config-if)# no shutdown

-- Save the configuration:

-- Router(config)# end

-- Router# write memory

Step 2: Configure the Access Point (Access Point-PT)

1. Access the Access Point Configuration:

Connect to the Access Point through a web browser or console if available.

2. Set Wireless Network Settings:

Go to the Wireless Settings or Wireless Setup section.

Set the following:

SSID: Choose a name like Lab10_Wireless

Channel: Set to Auto or a specific channel to avoid interference.

Security Mode: Set WPA2-PSK for security and provide a password.

3. Configure the Access Point's IP:

Navigate to LAN Settings.

Assign a Static IP in the range of 192.168.1.x, e.g., 192.168.1. x

Subnet Mask: 255.255.255.0

Gateway: 192.168.1.1 (the router's IP)

4. Save and Reboot the Access Point.

Step 3: Configure the Laptops (Laptop-PT0 and Laptop-PT1)

1. Laptop-PT0 (192.168.1.3):

Go to the Network and Sharing Center.

Choose the Wireless network and connect to Lab10_Wireless.

Set a Static IP address:

IP Address: 192.168.1.3

Subnet Mask: 255.255.255.0

Default Gateway: 192.168.1.1

DNS: 192.168.1.1 or a public DNS like 8.8.8.8.

2. Laptop-PT1 (192.168.1.4):

Repeat the same steps as Laptop-PT0 but set the IP address as 192.168.1.4.

Step 4: Configure the Printer (Printer-PT)

1. Assign IP Address to Printer:

Access the printer configuration through its web interface or physical interface.

Set the IP Address as 192.168.1.5, Subnet Mask 255.255.255.0, and Default

Gateway as 192.168.1.1.

Save the settings and restart the printer.

Step 5: Configure the Second Switch and Laptop on the 192.168.2.0 Network

1. Configure Switch1:

Connect a laptop via console and access the CLI.

The switch operates in Layer 2, so no IP address is required on ports.

2. Laptop-PT2 (192.168.2.2):

Set a static IP on the laptop:

IP Address: 192.168.2.2

Subnet Mask: 255.255.255.0

Default Gateway: 192.168.2.1 (the router's Fa0/1 interface).

Test the network connection by pinging the router and other devices.

Step 6: Test the Network Connectivity

1. Test Wireless Laptops:

From Laptop-PT0 and Laptop-PT1, open the command prompt and ping

192.168.1.1 (the router).

Ping the printer at 192.168.1.5.

25-Sep-2024

Training Day - 48 Report:

Physically Perform in Lab.

26-Sep-2024

Training Day - 49 Report:

LAB 11: SIMPLE MAIL TRANSFER PROTOCOL

Components in the Diagram:

-- Routers (2811): Routers interconnecting three different networks:

Network 1 (Left): 192.168.1.0/24

Network 2 (Center): 192.168.2.0/24

Network 3 (Right): 192.168.3.0/24

-- Switches (2960): Two switches connecting laptops and the email server to routers.

-- Email Server (SMTP): 192.168.2.2 configured to provide SMTP services.

-- Laptops: Laptops connected to their respective networks:

Pankaj Laptop: 192.168.1.x

Ram Laptop: 192.168.1.x

Sham Laptop: 192.168.3.x

Rohan Laptop: 192.168.3.x

Configuration Steps:

Step 1: Configure Router 1 (Left Router)

1. Access Router1 using the console or terminal.

2. Configure Interfaces:

Fa0/0 (192.168.1.1):

Router1> enable

Router1# configure terminal

Router1(config)# interface Fa0/0

Router1(config-if)# ip address 192.168.1.1 255.255.255.0

Router1(config-if)# no shutdown

Fa0/1 (10.0.0.1) for interconnection to Router2:

Router1(config)# interface Fa0/1

Router1(config-if)# ip address 10.0.0.1 255.255.255.0

Router1(config-if)# no shutdown

3. Enable Routing between interfaces:

Router1(config)# ip routing

4. Set up Static Routes for other networks:

Router1(config)# ip route 192.168.2.0 255.255.255.0 10.0.0.2

Router1(config)# ip route 192.168.3.0 255.255.255.0 10.0.0.2

5. Save Configuration:

Router1(config)# end

Router1# write memory

Step 2: Configure Router 2 (Central Router)

1. Access Router2 via console.

2. Configure Interfaces:

Fa1/0 (10.0.0.2) (Link to Router1):

Router2> enable

Router2# configure terminal

Router2(config)# interface Fa1/0

Router2(config-if)# ip address 10.0.0.2 255.255.255.0

Router2(config-if)# no shutdown

Fa0/1 (192.168.2.1) (Local network):

Router2(config)# interface Fa0/1

Router2(config-if)# ip address 192.168.2.1 255.255.255.0

Router2(config-if)# no shutdown

3. Set up Static Routes:

Route to 192.168.1.0/24 via 10.0.0.1 (Router1):

Router2(config)# ip route 192.168.1.0 255.255.255.0 10.0.0.1

Route to 192.168.3.0/24 via 40.0.0.1 (Router3):

Router2(config)# ip route 192.168.3.0 255.255.255.0 40.0.0.1

4. Save Configuration:

Router2(config)# end

Router2# write memory

Step 3: Configure Router 3 (Right Router)

1. Access Router3 through the console.

2. Configure Interfaces:

Router3> enable

Router3# configure terminal

Router3(config)# interface Fa0/0

Router3(config-if)# ip address 192.168.3.1 255.255.255.0

Router3(config-if)# no shutdown

Fa0/1 (40.0.0.2) for interconnection to Router2:

Router3(config)# interface Fa0/1

Router3(config-if)# ip address 40.0.0.2 255.255.255.0

Router3(config-if)# no shutdown

3. Enable Routing between interfaces:

Router3(config)# ip routing

4. Set up Static Routes:

Route to 192.168.1.0/24 via 40.0.0.1 (Router2):

Router3(config)# ip route 192.168.1.0 255.255.255.0 40.0.0.1

Route to 192.168.2.0/24 via 40.0.0.1 (Router2):

Router3(config)# ip route 192.168.2.0 255.255.255.0 40.0.0.1

5. Save Configuration:

Router3(config)# end

Router3# write memory

Step 4: Configure the SMTP Server

1. Access the Email Server (192.168.2.2) through the console or terminal.

2. Configure the SMTP Server Settings:

- Go to Services > Email.

- SMTP: Enable the SMTP service.

Set up the following:

Domain Name: example.com

Server Name: mail.example.com

Add user accounts for each laptop:

Pankaj: pankaj@example.com

Ram: ram@example.com

Sham: sham@example.com

Rohan: rohan@example.com

Step 5: Configure the Laptops for Email

1. Access Each Laptop (Pankaj, Ram, Sham, Rohan) and configure email clients:

Open the Email application.

Set up the following details:

- Incoming Mail Server: mail.example.com

- SMTP Server: 192.168.2.2

- Email Address: Use respective accounts:

Pankaj: pankaj@example.com

Ram: ram@example.com

Sham: sham@example.com

Rohan: rohan@example.com

Password: Set the respective password for each user.

2. Send Test Emails:

- From Pankaj's laptop, send an email to Ram at ram@example.com.

- From Sham's laptop, send an email to Rohan at rohan@example.com.

Step 6: Verify SMTP Communication

- Use Packet Tracer's simulation mode to verify that the SMTP packets are transmitted between the email server and the laptops across the different networks. The routers should correctly route the packets based on the static routes configured.

27-Sep-2024

Training Day - 50 Report:

Physically Perform in Lab.

30-Sep-2024

Training Day – 51 Report:

Test Of CCNA Lab

1-Oct-2024

Training Day – 52 Report:

What is Open Source?

Open source software is built on the principles of transparency, collaboration, and user freedom, allowing anyone to use, modify, and share the software, often under licenses that ensure these freedoms while enabling community-driven innovation and continuous improvement.

Key Points on Linux Origins:

-- 1984: The GNU Project and Free Software Foundation were established, aiming to

create an open-source version of UNIX utilities and introducing the General Public

License (GPL), a license enforcing open-source principles.

-- 1991: Linus Torvalds developed an open-source, UNIX-like kernel, also licensed

under the GPL, and integrated GNU utilities, seeking community support online.

-- Today: The Linux kernel combined with GNU utilities forms a complete, opensource, UNIX-like operating system. It's packaged into various distributions tailored

for specific users and needs.

Why Linux?

-- Open Source

-- Community Support

-- Highly Customizable

-- Server Dominance

-- DevOps Integration

-- Automation

-- Security

Architecture of Linux:

Some Importaznt Directories:

-- Home Directories: /root, /home/username

-- User Executable: /bin, /usr/bin, /usr/local/bin

-- System Executables: /sbin, /usr/sbin, /usr/local/sbin

-- Other Mount points: /media, /mnt

-- Configuration: /etc

-- Temporary Files: /tmp

-- Kernels and Bootloader: /boot

-- Server Data: /var, /srv

-- System Information: /proc, /sys

-- Shared Libraries: /lib, /usr/lib, /usr/local/lib

Different Linux Distributions:

Desktop Linux OS:

-- Ubuntu: User-friendly, strong community, frequent updates.

-- Linux Mint: Stable, beginner-friendly, based on Ubuntu.

-- Arch Linux: Minimalist, customizable, rolling-release.

-- Fedora: Cutting-edge, developer-focused, Red Hat-sponsored.

-- Debian: Stable, versatile, widely used in desktop/server.

-- OpenSUSE: Robust system management, developer-friendly.

Server Linux OS:

-- Red Hat Enterprise Linux (RHEL): Enterprise-grade, stable, secure.

-- Ubuntu Server: Scalable, flexible, popular in cloud environments.

-- CentOS: Free RHEL alternative, community-driven, stable.

-- SUSE Enterprise Linux: High performance, enterprise-focused.

Most Used in IT:

-- RPM-based: RHEL, CentOS (enterprise, stability).

-- Debian-based: Ubuntu Server (cloud, DevOps, flexibility).

Linux Commands

File and Directory Operations:

1. ls: Lists all files and directories in the current directory.

2. cd: Changes the current directory to the specified directory.

3. pwd: Prints the current working directory, showing the full path of the directory you are in. Useful for confirming your location within the file system.

4. mkdir: Creates a new directory with the specified name.

5. rm: Remove files

6. rm -r: Recursively removes a directory and all of its contents.

7. cp: Copy files

8. mv: Move or rename files.

9. touch: Create an empty file.

3-Oct-2024

Training Day – 53 Report:

File Viewing and Editing:

-- cat: Displays the entire contents of a file.

-- nano or vim: Command-line text editors for creating and editing files.

Filter Commands:

-- head: Shows the first few lines of a file (default is 10 lines).

-- tail: Shows the last few lines of a file (default is 10 lines).

-- less: Allows you to view file contents page by page with scrolling.

-- more: Displays file content one screen at a time, allowing you to scroll down through

long files using the spacebar.

-- Display First 5 Lines and Pipe to less:

4-Oct-2024

Training Day – 54 Report:

System Information:

-- uname -a: Displays detailed system information, including the kernel name, version,

and system architecture.

-- df: Shows disk space usage for all mounted filesystems; use `df -h` for humanreadable format (e.g., GB, MB).

-- top: Displays a real-time view of running processes, CPU, and memory usage, helping monitor system performance.

-- ps: Lists currently running processes; use `ps aux` for detailed information about each process.

-- whoami: Outputs the current user's username, indicating who is logged into the session.

File Permissions:

-- chmod: Changes file or directory permissions, allowing you to set read, write, and execute permissions for the owner, group, and others.

-- chown: Changes the ownership of a file or directory, assigning a different user or group as the owner.

Network:

-- ping: Sends packets to another network host to check connectivity and measure response time, useful for network troubleshooting.

-- ifconfig or ip addr or ip addr show: Displays network interface configurations, including IP addresses, network masks, and MAC addresses.

-- ssh: Provides secure remote login to another machine over a network, commonly used for remote administration.

Package Management:

-- apt-get (Debian/Ubuntu) or yum (Red Hat/CentOS): Installs, updates, or removes software packages from the system's package repositories.

-- apt-get update: Refreshes the package list, ensuring access to the latest versions available in the repositories.

-- apt-get install: Installs specified packages from the repository, resolving and downloading dependencies automatically.

Other Useful Commands:

-- man: Displays the manual page for a command, providing detailed information and usage options.

-- grep: Searches files for lines that match a specified pattern, often used for text processing and filtering.

-- sudo: Runs commands with superuser privileges, required for tasks that affect system settings or protected files.

-- clear: Clears the terminal screen, providing a clean workspace.

-- history: Lists previously executed commands in the terminal session, allowing easy recall or re-execution of commands.

7-Oct-2024

Training Day – 55 Report:

Managing users and groups:

1. User Management Commands:

-- useradd username: Creates new user account with default settings like home directory, shell, and adds entry to /etc/passwd file.

-- userdel username: Removes user account and can optionally delete their home directory and mail spool.

-- passwd username: Sets or changes password for a user account, stores encrypted password in /etc/shadow.

-- usermod: Modifies existing user account settings including groups, home directory, shell, and account expiry.

2. Group Management:

-- groupadd groupname: Creates a new group in the system and adds an entry to

/etc/group file.

-- groupdel groupname: Removes an existing group from the system, fails if it's a primary group of any user.

-- usermod -aG groupname username: Adds a user to supplementary groups while preserving existing group memberships.

3. Important Files:

-- /etc/passwd: Stores essential user account information including username, UID, GID, home directory, and default shell.

-- /etc/shadow: Contains encrypted password information and password policy settings for user accounts.

-- /etc/group: Maintains group information including group name, GID, and list of group members.

4. User Information:

-- id username: Displays user's UID, GID, and all groups they belong to.

-- whoami: Shows the current username, useful when switching between users or verifying effective user.

-- groups username: Lists all groups that a specific user belongs to, including primary and supplementary groups.

8-Oct-2024

Training Day – 56 Report:

Types of Files in linux:

Here are the main types of files in Linux with their descriptions:

1. Regular Files (-) :

-- Most common file type that contains data, text, or program instructions

-- Examples include text files, images, scripts, and binary executables

2. Directory (d) :

-- Special file that contains a list of filenames and related information

-- Acts as a container for other files and directories, creating the filesystem hierarchy

3. Link Files (l) :

-- Symbolic links (soft links) that point to other files in the filesystem

-- Acts like shortcuts in Windows, containing the path to the target file

4. Character Device Files (c) :

-- Provide interface for serial I/O access to hardware devices

-- Examples include terminals, keyboard, mouse, and serial ports

5. Block Device Files (b) :

-- Provide interface for block I/O access to hardware devices

-- Used for devices that store or hold data like hard drives and USB drives

6. Socket Files (s) :

-- Enable communication between processes on the same machine

-- Used for inter-process communication and networking

7. Named Pipes/FIFO (p) :

-- Special files that allow processes to communicate with each other

-- Acts as a connection between two processes for data transfer

9-Oct-2024

Training Day – 57 Report:

Introduction to Python

Python is a high-level, interpreted programming language that has gained immense popularity

since its inception in the late 1980s. Developed by Guido van Rossum and first released in 1991, Python was designed with code readability and simplicity in mind. Its syntax is clear and intuitive, making it an excellent choice for both beginners and experienced programmers.

Why We Need Python

Ease of Learning:

-- Python's straightforward syntax and structure allow beginners to grasp programming concepts quickly. This makes it an ideal first language for new programmers.

Versatility:

-- Python is a multiparadigm language, supporting object-oriented, imperative, and functional programming styles. This versatility allows developers to choose the best approach for their projects.

Wide Range of Applications:

Python is used in various domains, including:

-- Web Development: Frameworks like Django and Flask make it easy to build robust web applications.

-- Data Science and Machine Learning: Libraries like Pandas, NumPy, and scikitlearn provide powerful tools for data analysis and machine learning.

-- Automation and Scripting: Python is often used for writing scripts to automate repetitive tasks.

-- Game Development: Libraries such as Pygame allow developers to create games.

-- Scientific Computing: Python is widely used in scientific research and simulations with libraries like SciPy and Matplotlib.

Python has a large and active community of developers who contribute to its growth. This means abundant resources, including documentation, tutorials, and forums, are available to help learners and developers troubleshoot issues.

Rich Ecosystem of Libraries and Frameworks:

-- Python has a vast ecosystem of libraries and frameworks that extend its functionality. This allows developers to leverage existing tools to accelerate development and focus

on solving specific problems rather than reinventing the wheel.

Cross-Platform Compatibility:

-- Python is compatible with various operating systems, including Windows, macOS, and Linux. This cross-platform nature allows developers to write code once and run it anywhere.

Integration Capabilities:

-- Python can easily integrate with other languages like C, C++, and Java, allowing developers to use Python as a scripting language within larger applications.

Career Opportunities:

-- With the rise of data science, machine learning, and web development, Python has become one of the most sought-after programming languages in the job market. Proficiency in Python can open doors to numerous career opportunities.

What is JSON?

JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate. It is primarily used for transmitting data between a server and a web application as an alternative to XML.

Key Characteristics of JSON:

-- Text-Based: JSON is a text format that is completely language-independent but uses conventions that are familiar to programmers of the C family of languages (C, C++, Java, JavaScript, etc.).

-- Data Structure: JSON supports simple data structures like objects (key-value pairs) and arrays (ordered lists), making it suitable for representing complex data.

-- Format: JSON data is represented in a key-value pair format, where keys are strings and values can be strings, numbers, objects, arrays, booleans, or null.

What is YAML?

YAML (YAML Ain't Markup Language) is a human-readable data serialization format that is

often used for configuration files and data exchange between languages with different data structures. It is designed to be easy to read and write, making it a popular choice for configuration files and data representation.

Key Characteristics of YAML:

-- Human-Readable: YAML is designed to be easy for humans to read and write, with a focus on clarity and simplicity.

-- Data Structure: YAML supports complex data structures, including scalars (strings, integers), sequences (lists), and mappings (key-value pairs).

-- Indentation-Based: YAML uses indentation to represent structure, which makes it visually intuitive but requires careful attention to spacing.

What is a Variable?

A variable in programming is a named storage location in memory that can hold a value. It acts as a container for data, allowing you to store, modify, and retrieve information during the execution of a program. Variables are fundamental to programming because they enable you to work with dynamic data.

10-Oct-2024

Training Day – 58 Report:

What is an Operator?

An operator is a symbol or keyword in programming that performs a specific operation on one, two, or more operands (variables or values). Operators are fundamental to programming as they allow you to manipulate data and perform calculations.

Types of Operators in Python

Python supports several types of operators, which can be categorized as follows:

1. Arithmetic Operators

Used to perform mathematical operations.

Examples:

-- Addition (+)

-- Subtraction (-)

-- Multiplication (*)

-- Division (/)

-- Modulus (%)

-- Exponentiation (**)

-- Floor Division (//)

2. Comparison Operators

Used to compare two values and return a Boolean result (True or False).

Examples:

-- Equal to (==)

-- Not equal to (!=)

-- Greater than (>)

-- Less than (<)

-- Greater than or equal to (>=)

-- Less than or equal to (<=)

3. Logical Operators:

Used to combine conditional statements.

Examples:

-- Logical AND (and)

-- Logical OR (or)

-- Logical NOT (not)

4. Assignment Operators:

Used to assign values to variables.

Examples:

-- Assignment (=)

-- Add and assign (+=)

-- Subtract and assign (-=)

-- Multiply and assign (*=)

-- Divide and assign (/=)

-- Modulus and assign (%=)

11-Oct-2024

Training Day – 59 Report:

Lists in Python

A list in Python is a built-in data structure that allows you to store an ordered collection of items. Lists are versatile and can hold a variety of data types, including numbers, strings, and even other lists. They are mutable, meaning you can modify them after their creation (e.g., adding, removing, or changing elements).

Characteristics of Lists

1. Ordered: The items in a list have a specific order, and that order is preserved. You can access items by their index, which starts at 0.

2. Mutable: Lists can be changed in place. You can add, remove, or modify items in a list.

3. Heterogeneous: Lists can contain items of different data types.

4. Dynamic: The size of a list can change as you add or remove items.

Creating a List

You can create a list by placing comma-separated values inside square brackets [].

Accessing List Elements

You can access elements in a list using their index.

Modifying Lists

You can modify lists using various methods:

Adding Elements:

-- append(): Adds an item to the end of the list.

-- insert(): Inserts an item at a specified index.

-- extend(): Adds multiple items to the end of the list.

Slicing Lists

You can access a subset of a list using slicing.

14-Oct-2024

Training Day – 60 Report:

Sets in Python

A set in Python is an unordered collection of unique items. Sets are a built-in data type that provides a way to store multiple items in a single variable while ensuring that each item is unique (i.e., no duplicates). Sets are mutable, meaning you can add or remove items after the set is created.

Characteristics of Sets

-- Unordered: The items in a set do not have a defined order, and you cannot access items by index.

-- Unique: Each item in a set must be unique; duplicate items are not allowed.

-- Mutable: You can add or remove items from a set, but the items themselves must be immutable types (like numbers, strings, or tuples).

-- Dynamic: Sets can grow and shrink as you add or remove items.

Creating a Set

You can create a set by placing comma-separated values inside curly braces {} or by using the set() constructor.

Example:

Tuples in Python

A tuple in Python is a built-in data structure that is used to store a collection of items. Tuples are similar to lists but have some key differences. They are immutable, meaning that once they are created, their contents cannot be changed. This makes tuples useful for storing data that should not be modified.

Characteristics of Tuples

1. Ordered: Tuples maintain the order of elements. The items can be accessed by their index, which starts at 0.
2. Immutable: Once a tuple is created, you cannot modify its contents (i.e., you cannot add, remove, or change items).
3. Heterogeneous: Tuples can contain items of different data types, including numbers, strings, lists, and even other tuples.
4. Dynamic: While the contents of a tuple cannot be changed, you can create new tuples from existing ones.

Creating a Tuple

You can create a tuple by placing a comma-separated list of values inside parentheses ().

Example:

15-Oct-2024

Training Day – 61 Report:

Dictionaries in Python

A dictionary in Python is a built-in data structure that allows you to store data in key-value pairs. Dictionaries are mutable, unordered collections that can hold a variety of data types, including other dictionaries, lists, and tuples. They are often used to represent structured data and provide fast access to values based on their keys.

Characteristics of Dictionaries

-- Key-Value Pairs: Each item in a dictionary is stored as a pair, consisting of a key and a value. The key must be unique and immutable (e.g., strings, numbers, or tuples), while the value can be of any data type and can be duplicated.

-- Unordered: Dictionaries do not maintain the order of items. However, as of Python 3.7, dictionaries preserve the insertion order of keys.

-- Mutable: You can change, add, or remove items from a dictionary after it has been created.

-- Dynamic: The size of a dictionary can change as you add or remove key-value pairs.

Key Features of Python

a) Easy to Learn and Use: Simple and readable syntax.

b) Interpreted Language: Code is executed line by line, facilitating debugging.

c) Dynamically Typed: Variable types are determined at runtime.

d) Object-Oriented: Supports encapsulation, inheritance, and polymorphism.

e) Extensive Standard Library: Rich set of modules and functions for various tasks.

f) Cross-Platform Compatibility: Runs on multiple operating systems without modification.

g) Support for Multiple Programming Paradigms: Supports procedural, objectoriented, and functional programming.

h) Community and Ecosystem: Large, active community with abundant resources and libraries.

i) Integration Capabilities: Easily integrates with other programming languages.

j) Rich Ecosystem of Third-Party Libraries: Vast collection of libraries available through PyPI.

k) Generators and Iterators: Efficient looping over large datasets without consuming memory.

l) List Comprehensions: Concise and readable way to create lists.

m) Exception Handling: Robust mechanism for managing errors gracefully.

n) Decorators: Modify the behavior of functions or methods.

16-oct-2024

Training Day - 62 Report:

What is Chocolatey:

Chocolatey is a package manager for Windows that makes it easier to install, update, and manage software. Here's a clear explanation:

Core Concepts:

-- A command-line package manager similar to apt-get (Linux) but for Windows

-- Automates software installation, updates, and uninstallation

-- Uses PowerShell to execute commands

-- Manages dependencies automatically

Key Features:

1. Installation Management:

-- Simple commands like choco install firefox

-- Can install multiple packages at once

-- Handles software dependencies automatically

-- Silent installations (no clicking through installers)

2. Common Commands:

-- choco install packagename - Install a package

-- choco uninstall packagename - Remove a package

-- choco upgrade packagename - Update a package

-- choco list - Show installed packages

-- choco search term - Search for packages

3. Benefits:

-- Saves time on software installation

-- Consistent installation process

-- Easy system maintenance

-- Can script installations for multiple computers

-- Access to thousands of software packages

STEPS TO INSTALL CHOCOLATEY ON WINDOWS:

1. Open PowerShell as Administrator (right-click Windows PowerShell and select "Run as Administrator")

2. First, check your execution policy by running:

COMMAND: Get-ExecutionPolicy

3. If it's restricted, run:

COMMAND: Set-ExecutionPolicy AllSigned

4. Run this command to install Chocolatey:

5. Wait for the installation to complete

6. Verify the installation by running:

COMMAND: choco –version

What is GitBash:

GitBash is a command-line interface for Windows that provides a Unix-like shell environment, combining Git functionality with traditional Bash commands. It enables users to interact with Git repositories while using familiar Unix commands on Windows systems.

Key Features:

-- Unix-style command-line environment for Windows

-- Built-in Git commands integration

-- BASH emulation for Windows

-- SSH client and key generation

-- Unix tools packaged for Windows

Essential GitBash Commands:

1. Basic Navigation:

-- pwd - Current directory

-- ls - List files

-- cd - Change directory

-- mkdir - Create directory

2. File Operations:

-- touch - Create empty file

-- rm - Remove files

-- cp - Copy files

-- mv - Move/rename file

STEPS TO INSTALL GITBASH USING CHOCOLATEY:

1. Open PowerShell as Administrator and run:

COMMAND: choco install git.install (this command for install both git and bit bash)

2. Verify the installation:

COMMAND: git –version

Introduction to Vagrant

Vagrant is a powerful tool used to build and manage virtualized development environments efficiently. By providing a consistent, reproducible environment, it eliminates the common "it works on my machine" issue, making collaboration easier.

Why Use Vagrant?

-- Ease of Setup: Simplifies and automates the setup of development environments, reducing time and effort.

-- Cross-Platform Compatibility: Works on Windows, macOS, and Linux, ensuring consistent development environments across teams.

-- Portable and Reproducible: Configurations are defined in a single `Vagrantfile`, making it easy to reproduce environments.

Core Features

-- Provider Agnostic: Supports various providers like VirtualBox, VMware, AWS, and more.

-- Provisioning Tools: Integrates with provisioning tools like shell scripts, Chef, and Puppet to automate software configuration.

-- Single Workflow: Manages virtual machines through a simple workflow (`vagrant up`, `vagrant halt`, etc.), streamlining the process for all users.

Benefits by Role

1. For Developers:

-- Isolates dependencies and configurations in a consistent environment.

-- Enables easy setup across teams, ensuring everyone works in the same environment.

-- Reduces bugs due to environmental differences.

2. For Operators (DevOps):

-- Offers a disposable environment for testing infrastructure management scripts (e.g., shell, Chef, Puppet).

-- Supports testing on local and cloud environments (e.g., AWS) using the same configuration.

3. For Designers:

-- Simplifies the setup of web applications, allowing designers to focus on design without worrying about setup.

-- Eliminates dependency on developers for environment fixes.

INSTALLATION OF VAGRANT USING CHOCOLATEY:

1. Before install vagrant on your system you have to install virtualbox:

COMMAND: choco install virtualbox -y

2. Install Vagrant using Chocolatey:

COMMAND: choco install vagrant -y

3. Verify the installation:

COMMAND: vagrant –version

18-Oct-2024

Training Day – 63 Report:

Basic Vagrant commands to get started:

-- mkdir vagrant-vms - Used to Create a directory for your project.

-- cd vagrant-vms

-- vagrant init ubuntu/jammy64 - Initialize Vagrant with a base box (Ubuntu in this example).

-- vagrant up - It is used to start and provision a virtual machine based on the configuration defined in the Vagrantfile.

-- vagrant ssh - To connects you to the VM that has been created and is currently running.

-- vagrant destroy - It is used to completely remove the virtual machine and its associated resources.

-- vagrant status - It shows the current state of the virtual machine, indicating whether it is running, stopped, or in another state.

21-Oct-2024

Training Day - 64 Report:

Changes in Vagrant File:

1. First, edit Vagrantfile, open on wordpad or notepad

2. To assign public ip to the virtual m/c:

Remove #(hash) from these lines:-

-- config.vm.network "private_network", ip: "192.168.33.10"

-- config.vm.network "public_network"

Change the Memory or RAM Size in our m/c

we also increase or decrease the size of the memory and ram by removing #(hash) from lines given below:

By shell scripting we update and install nginx server and start their services:

PROVISIONING SHELL SCRIPTING

It show us updates and changes we done in the vagrantfile.

Vagrant reload --provision

22-Oct-2024

Training Day - 65 Report:

Static websites setup using vagrant

1. Create the project directory

o mkdir vagrant-website

o cd vagrant-website

o mkdir site

2. Create the Vagrantfile

o vagrant init ubuntu/jammy64 --box-version 20241002.0.0

3. Open the Vagrantfile and replace the contents with the following

```
# -*- mode: ruby -*-
 # vi: set ft=ruby :
 Vagrant.configure("2") do |config|
 # Use Ubuntu 20.04 LTS as the base box
config.vm.box = "ubuntu/focal64"
# Define a VM name
```

```
config.vm.hostname = "static-site"
# Configure the network. We use port forwarding so the website is accessible via
localhost.
 config.vm.network "forwarded_port", guest: 80, host: 8080
 # Provision the virtual machine


 config.vm.provision "shell", inline: <<-SHELL
 # Update package lists
 sudo apt-get update
 # Install Nginx
 sudo apt-get install -y nginx
 # Remove the default Nginx configuration file
 sudo rm /etc/nginx/sites-enabled/default
 # Create a new Nginx configuration file for the static site
 sudo tee /etc/nginx/sites-available/static-site <<EOL
server {
 listen 80;
 server_name localhost;
 root /vagrant/site;
 index index.html;
}
EOL
 # Enable the configuration by linking it
 sudo ln -s /etc/nginx/sites-available/static-site /etc/nginx/sites-enabled/static-site
 # Restart Nginx to apply the new configuration
 sudo systemctl restart nginx
 SHELL
```

Sync the 'site' directory to /vagrant/site in the VM

config.vm.synced_folder "./site", "/vagrant/site"

end

4. Create the static website

5. Start the Vagrant environment

COMMAND-: vagrant up

6. Access the website

Once the setup is complete, you can access your website by navigating to

http://localhost:8080 in your browser.

7. Managing the virtual machine

-- To stop the VM, run vagrant halt.

-- To destroy the VM, run vagrant destroy.

-- To SSH into the VM, run vagrant ssh.

23-Oct-2024

Training Day - 66 Report:

Website online through Shell Scripting In vagrant File

1. Create the project directory

o mkdir vagrant-shell-scripted-website

o cd vagrant-shell-scripted-website

o mkdir site

2. Create a basic HTML file

3. Create the Vagrantfile

COMMAND-: vagrant init ubuntu/focal64 --box-version 20240821.0.1

Open the Vagrantfile and add the following content:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config|
# Use Ubuntu 20.04 as the base box
 config.vm.box = "ubuntu/focal64"
# Name the virtual machine
 config.vm.hostname = "shell-scripted-site"
# Forward port 80 on the guest machine to port 8080 on the host machine
 config.vm.network "forwarded_port", guest: 80, host: 8080
 # Sync the 'site' folder to the '/vagrant/site' directory in the VM
 config.vm.synced_folder "./site", "/vagrant/site"
 # Provision the VM with a shell script

 config.vm.provision "shell", inline: <<-SHELL
 # Update the package list
 sudo apt-get update
 # Install Nginx
 sudo apt-get install -y nginx
 # Remove the default Nginx config file if it exists
 sudo rm -f /etc/nginx/sites-enabled/default
 # Create a new Nginx configuration to serve the static site
 sudo bash -c 'cat <<EOF > /etc/nginx/sites-available/static-site
server {
 listen 80;
 server_name localhost;
 root /vagrant/site;
 index index.html;
```

```
 location / {
 try_files \$uri \$uri/ =404;
 }
}
EOF'
 # Enable the new site configuration
 sudo ln -s /etc/nginx/sites-available/static-site /etc/nginx/sites-enabled/static-site
 # Restart Nginx to apply changes
 sudo systemctl restart nginx
 # Ensure Nginx starts on boot
 sudo systemctl enable nginx
 SHELL
End
```

4. Start the Vagrant environment

COMMAND-: vagrant up

5. Access your website

Search on browser-: http://localhost:8080

6. Managing the virtual machine

-- To stop the VM, run vagrant halt.

-- To destroy the VM, run vagrant destroy.

-- To SSH into the VM, run vagrant ssh.

24-Oct-2024

Training Day - 67 Report:

Wordpress setup using vagrant:

1) mkdir vagrant-wordpress

2) cd vagrant-wordpress

3) vagrant init ubuntu/jammy64

4) vagrant up

5) vagrant ssh

6) In Google Search, Install and configure wordpress in Ubuntu

7) Link-: https://ubuntu.com/server/docs/how-to-install-and-configure-wordpress

8) Follow all the steps of given above link and copy all the steps on virtual machine

which we are using.

9) After the completion of these all steps

10) Type ip addr show command in your linux

11) And copy the local machine ip and paste in your browser

12) wordpress is live

25-Oct-2024

Training Day- 68 Report:

Automate Deploy Wordpress using Vagrantfile

1. Create the project directory

o mkdir vagrant-wordpress-automation

o cd vagrant-wordpress-automation

2. Initialize Vagrant with the Ubuntu 22.04 (Jammy Jellyfish) box

o vagrant init ubuntu/jammy64

3. Edit the Vagrantfile

o Edit Vagrantfile and adding a new shell script for Install and configure wordpress in

Ubuntu

```
config.vm.provision "shell", inline: <<-SHELL
 apt-get update
#install apache2
```

```
apt-get install -y apache2
#install wget and tr utilities
sudo apt-get install -y wget coreutils
#install MySQL
sudo apt-get install -y mysql-server
sudo mysql -e "CREATE DATABASE chd;"
sudo mysql -e "CREATE USER 'pankaj'@'localhost' IDENTIFIED BY '123';"
sudo mysql -e "GRANT ALL PRIVILEGES ON chd.* TO 'pankaj'@'localhost';"
sudo mysql -e "FLUSH PRIVILEGES;"
#Install PHP and required extension

sudo apt-get install -y php libapache2-mod-php php-mysql
#Download Wordpress and Setup it
cd /var/www/html
sudo rm -rf wordpress
sudo wget https://wordpress.org/latest.tar.gz
sudo tar -xzf latest.tar.gz
sudo mv wordpress/* .
sudo rm -rf wordpress latest.tar.gz
sudo chown -R www-data:www-data /var/www/html
sudo chmod -R 755 /var/www/html
#remove Default Apache Page if it exists
sudo rm -f /var/www/html/index.html
#create wp-config using sample config File
sudo cp /var/www/html/wp-config-sample.php /var/www/html/wp-config.php
sudo sed -i "s/database_name_here/chd/" /var/www/html/wp-config.php
sudo sed -i "s/username_here/pankaj/" /var/www/html/wp-config.php
```

```
 sudo sed -i "s/password_here/123/" /var/www/html/wp-config.php
#configure Apache for Wordpress
 sudo a2enmod rewrite
 sudo systemctl restart apache2
```

4. Provision the Vagrant environment

o vagrant up --provision

o vagrant ssh

5. Access WordPress

o Now, Copy the local m/c ip and paste in your browser

o Wordpress is live through Automation

28-Oct-2024

Training Day - 69 Report:

Multiple Virtual Machines Start using single vagrant File

1. Create a project directory for multiple VMs

-- mkdir vagrant-multiple_m/c's

-- cd vagrant-wordp multiple_m/c's

2. Create and edit the Vagrantfile

```
Vagrant.configure("2") do |config|
 config.hostmanager.enabled = true
 config.hostmanager.manage_host = true

### DB vm ####
 config.vm.define "db01" do |db01|
 db01.vm.box = "eurolinux-vagrant/centos-stream-9"
 db01.vm.box_version = "9.0.43"
 db01.vm.hostname = "db01"
```

```
 db01.vm.network "private_network", ip: "192.168.56.15"
 db01.vm.provider "virtualbox" do |vb|
 vb.memory = "300"
 end

 end

### Memcache vm ####
 config.vm.define "mc01" do |mc01|
 mc01.vm.box = "eurolinux-vagrant/centos-stream-9"
 mc01.vm.box_version = "9.0.43"

 mc01.vm.hostname = "mc01"
 mc01.vm.network "private_network", ip: "192.168.56.14"
 mc01.vm.provider "virtualbox" do |vb|
 vb.memory = "300"
 end

 end

### RabbitMQ vm ####
 config.vm.define "rmq01" do |rmq01|
 rmq01.vm.box = "eurolinux-vagrant/centos-stream-9"
 rmq01.vm.box_version = "9.0.43"
 rmq01.vm.hostname = "rmq01"
 rmq01.vm.network "private_network", ip: "192.168.56.16"
 rmq01.vm.provider "virtualbox" do |vb|
```

```
 vb.memory = "300"
 end


 end

### tomcat vm ###
 config.vm.define "app01" do |app01|
 app01.vm.box = "eurolinux-vagrant/centos-stream-9"
 app01.vm.box_version = "9.0.43"
 app01.vm.hostname = "app01"
 app01.vm.network "private_network", ip: "192.168.56.12"
 app01.vm.provider "virtualbox" do |vb|
 vb.memory = "300"
 end
 end

### Nginx VM ###
 config.vm.define "web01" do |web01|
 web01.vm.box = "ubuntu/jammy64"
 web01.vm.hostname = "web01"
 web01.vm.network "private_network", ip: "192.168.56.11"
# web01.vm.network "public_network"
 web01.vm.provider "virtualbox" do |vb|
 vb.gui = true
 vb.memory = "300"
 end
```

end

end

3. Explanation of the Vagrantfile

This Vagrantfile defines two virtual machines:

-- Web Server VM:

o Uses Ubuntu 22.04 (ubuntu/jammy64).

o Has a private IP of 192.168.56.30.

o Allocates 1 CPU and 1550 MB of memory.

o Provisions Apache and curl via a shell script.

-- Database Server VM:

o Uses Ubuntu 22.04 (ubuntu/jammy64).

o Has a private IP of 192.168.56.40.

o Allocates 1 CPU and 1550 MB of memory.

o Provisions MySQL via a shell script.

4. Bring up the virtual machines:

-- vagrant up

-- vagrant status

-- vagrant up web

-- vagrant up db

29-Oct-2024

Training Day - 70 Report:

Create systemctl Service For Tomcat Web-Server

1. Create a Directory for the Tomcat Server

mkdir TOMCAT-server

cd TOMCAT-server

2. Initialize Vagrant

vagrant init ubuntu/jammy64

3. Bring up the Vagrant Virtual Machine

4. SSH into the VM

5. Install Apache

6. Check Apache Status

7. Download and Install Tomcat

Download Tomcat 10 from the official website:

Extract

8. Install Java

Before running Tomcat, ensure that Java is installed. First, check the current Java version:

If Java is not installed, update the package list and install OpenJDK 17:

9. Navigate to the Tomcat bin Directory

After extracting Tomcat, change to the bin directory of Tomcat.

10. Start Tomcat

Run the Tomcat startup script to start the server

11. Verify Tomcat is Running:

Check if Tomcat is running using the ps command.

12. Stop Tomcat (if needed):

You can stop Tomcat by killing the process ID (PID) found in the previous step.

30-Oct-2024

Training Day - 71 Report:

13. Create a systemctl Service for Tomcat:

13.1 Create a Tomcat User Without Home Directory:

Create a Tomcat user:

13.2 Copy Tomcat Files to the Home Directory:

Copy the necessary Tomcat files to /opt/tomcat:

13.3 Remove the Old Directory:

Optionally, remove the old Tomcat directory:

13.4 Set Ownership for the Tomcat Directory:

Assign ownership of the Tomcat directory to the tomcat user:

13.5 Create Systemd Service File for Tomcat:

Create the systemd service file for Tomcat and reload systemd to apply the configuration changes:

14.Final Steps for Tomcat Server Setup:

14.1 Update Alternatives and Configure Java:

Update the alternatives and configure Java using:

14.2 Systemd Configuration for Tomcat:

Create the systemd service file for Tomcat:

14.3 Reload Systemd Daemon and Start Tomcat Service:

Reload the systemd configuration:

14.4 Start the Tomcat service:

4-Nov-2024

Training Day - 72 Report:

Introduction of Docker

Docker is an open-source platform designed to automate the deployment, scaling, and management of applications using containerization technology. It provides a standardized unit of software, known as a container, that packages an application and all its dependencies,

enabling it to run consistently across various computing environments.

Key Concepts of Docker

1. Containerization

Definition: Containerization is the process of encapsulating an application and its dependencies into a container. This allows applications to run in isolated environments without interfering with each other.

Isolation: Each container operates independently, sharing the host OS kernel but maintaining its own file system, libraries, and configuration files. This isolation helps prevent conflicts between applications.

2. Docker Images

Definition: A Docker image is a lightweight, standalone, executable package that includes everything needed to run a piece of software, including the application code, libraries, dependencies, and runtime.

Layered Architecture: Docker images are built in layers, where each layer represents a set of file changes. This layered structure allows for efficient storage and sharing of common layers across different images.

Dockerfile: Images are created from a Dockerfile, a text document that contains a series of commands and instructions to assemble the image. The Dockerfile specifies the base image, application code, dependencies, and configuration.

3. Docker Containers

Definition: A Docker container is a running instance of a Docker image. It is a lightweight, standalone environment where the application runs.

Lifecycle: Containers can be created, started, stopped, and removed. They are ephemeral by nature, meaning they can be easily destroyed and recreated.

Resource Efficiency: Containers share the host OS kernel, making them more efficient in terms of resource usage compared to traditional virtual machines (VMs).

4. Docker Engine

Definition: The Docker Engine is the core component of Docker that enables users to create, run, and manage containers. It consists of a server (the Docker daemon), a REST API, and a command-line interface (CLI).

Architecture: The Docker daemon manages the containers and images on the host system, while the Docker CLI allows users to interact with the Docker daemon through commands.

5. Docker Hub

Definition: Docker Hub is a cloud-based registry service for sharing and managing Docker images. It allows users to store and distribute images publicly or privately.

Pre-built Images: Docker Hub hosts a wide range of pre-built images, making it easy for developers to find and use existing software packages.

Advantages of Using Docker

1. Consistency Across Environments: Docker ensures that applications run the same way in development, testing, and production environments, reducing "it works on my machine" issues.

2. Isolation: Each container runs in its own environment, preventing conflicts between applications and making it easier to manage dependencies.

3. Portability: Docker containers can run on any system that supports Docker, allowing for seamless movement of applications across different environments.

4. Scalability: Docker makes it easy to scale applications by adding or removing containers based on demand. This is particularly useful in microservices architectures.

5. Resource Efficiency: Containers are lightweight and can start quickly, leading to better resource utilization compared to traditional virtual machines.

6. Rapid Deployment: Docker allows for faster application deployment by enabling developers to package applications and their dependencies together.

7. Microservices Support: Docker is well-suited for microservices architectures, where applications are broken down into smaller, independent services that can be

developed, deployed, and scaled independently.

8. Version Control: Docker images can be versioned, allowing developers to track changes, roll back to previous versions, and maintain multiple versions of an application.

9. Integration with CI/CD: Docker integrates well with Continuous Integration and Continuous Deployment (CI/CD) pipelines, enabling automated testing and

deployment of applications.

10. Community and Ecosystem: Docker has a large community and a rich ecosystem of tools, libraries, and extensions that enhance development workflows.

Use Cases for Docker

1) Development Environments: Developers can create consistent environments for their applications, reducing setup time and conflicts.

2) Microservices: Docker facilitates the development and deployment of microservices by allowing each service to run in its own container.

3) Continuous Integration/Continuous Deployment: Docker enables automated testing and deployment processes in CI/CD pipelines.

4) Cloud Deployments: Docker containers can be easily deployed on cloud platforms, making it an ideal choice for cloud-native applications.

5) Legacy Application Migration: Docker can be used to containerize legacy applications, making them easier to deploy and manage in modern environments.

Conclusion

Docker is a powerful tool that has transformed how applications are developed, deployed, and managed. Its containerization technology provides a consistent, efficient, and scalable

5-Nov-2024

Training Day - 73 Report:

Containers and Docker

1. Through Docker Container deploy a project:

Docker containers are lightweight, standalone packages of software that include everything needed to run an application (code, libraries, dependencies, etc.). This makes deploying projects fast and consistent across different environments.

2. Make folders and put Vagrant file:

-- Vagrant is a tool for managing virtual machine environments.

-- A Vagrantfile is a configuration file that defines the setup for the virtual environment, such as specifying the base operating system, networking, and provisioning tools like Docker.

3. Search docker.com:

-- Visit Docker's official website (https://www.docker.com/) to download and install Docker, and explore its features, documentation, and tutorials.

4. Install Docker:

-- Docker installation steps include setting up Docker's repository, adding its GPG key, and installing Docker packages (as shown in your earlier script).

5. Vagrant up:

-- The vagrant up command initializes and starts the virtual environment defined in the Vagrantfile. If Docker provisioning is included, Docker will be installed and configured within this virtual environment.

Docker Build Cloud:

-- Docker Build Cloud is likely referring to tools like Docker Build or Docker Hub:

-- Docker Build: Command to build Docker images locally or in CI (Continuous Integration) environments.

-- Docker Hub: A cloud-based registry service that allows you to host and share container images.

-- These tools make it easier to create, store, and deploy Docker images.

Commands Explanation:

1. vagrant ssh:

o Connects to the Vagrant-managed virtual machine using SSH. Once connected, you can run commands inside the VM.

2. systemctl status docker:

o Checks the current status of the Docker service (active, inactive, or failed).

3. sudo -i:

o Switches to the root user for elevated privileges, necessary to execute certain Docker commands.

4. docker images:

o Lists all Docker images currently stored on the system, including the image name, tag, and size.

5. docker ps:

o Displays all currently running Docker containers.

6. docker ps -a:

o Shows a list of all containers (both running and stopped).

7. docker run hello-world:

o Runs the hello-world container, which verifies that Docker is installed and functioning correctly.

Alternatively:

o docker pull hello-world: Downloads the hello-world image without running it.

6-Nov-2024

Training Day - 74 Report:

Commands and Explanations

1. docker run --name web01 -p 9080:80 nginx:

-- docker run: Starts a new container.

-- --name web01: Assigns the name web01 to the container for easy reference.

-- -p 9080:80: Maps port 9080 on the host machine to port 80 inside the container. This allows accessing the Nginx server running in the container via http://localhost:9080.

-- nginx: Specifies the Nginx image to run.

Additional Note:

-- If the -d flag is used, the container will run in the background.

2. docker inspect web01:

-- Provides detailed information about the web01 container, such as its configuration, network settings, and status.

3. curl http://172.17.0.2:

-- Sends an HTTP request to the container's IP address (172.17.0.2 is a typical IP for Docker containers using the default bridge network). This checks if the Nginx server is running and serving content.

4. ip addr show:

-- Displays network interfaces and their associated IP addresses on the host machine. Useful for verifying network configurations.

5. mkdir images:

-- Creates a new directory named images. Typically used for organizing Dockerrelated files (like Dockerfiles or scripts).

6. cd images:

-- Changes the working directory to images.

7. vim Dockerfile:

-- Opens or creates a Dockerfile using the vim text editor.

-- The Dockerfile contains instructions to build a custom Docker image.

8. docker build -t testing .:

-- docker build: Builds a Docker image from the current directory (denoted by .) containing the Dockerfile.

-- -t testing: Tags the built image with the name testing.

9. docker run -p 80:80 testing:

-- Runs a container from the testing image and maps port 80 on the host to port 80 in the container. This exposes the application in the container to the host machine on port 80.

10. docker stop web01:

-- Stops the web01 container gracefully.

11. docker rm web01:

-- Removes the stopped container named web01. Containers must be stopped before they can be removed.

12. docker images:

-- Lists all locally available Docker images. Displays repository name, tag, image ID, creation date, and size.

13. docker rmi <image_id>:

-- Removes a Docker image by its ID. This helps free up space by deleting unused images.

Summary of Workflow:

1. Start by running a container (docker run) and mapping ports for accessibility.

2. Use docker inspect and curl to verify that the container is running and accessible.

3. Use mkdir, cd, and vim to create a directory and write a Dockerfile for custom image building.

4. Build a custom image with docker build, run it with docker run, and expose the application via port mapping.

5. Stop and remove containers/images (docker stop, docker rm, docker rmi) as needed to manage resources.

7-Aug-2024

Training Day - 75 Report:

Monolithic Architecture in Docker:

-- The entire application is packaged into a single container, which contains all components like UI, business logic, and database access.

-- Scaling means replicating the entire container.

-- Simpler to set up in Docker but harder to manage and update as the application grows.

Microservices Architecture in Docker:

-- Each service (e.g., user service, payment service, inventory service) is deployed in its own container.

-- Services communicate with each other using APIs (e.g., REST or gRPC).

-- Allows independent scaling, deployment, and updates of each service.

Comparison:

Why Microservices Are More Popular Now:

1. Scalability:

-- Modern applications need to scale specific parts of the system independently.
For instance, an e-commerce platform might scale its product catalog service during a sale, without scaling the user authentication service.

-- Microservices, combined with tools like Docker and Kubernetes, make independent scaling efficient.

2. Flexibility in Technology Stack:

-- Microservices allow developers to use different languages and technologies for different services. For instance, a company could use Python for machine learning services, Node.js for APIs, and Java for backend processing.

3. Cloud-Native Applications:

-- With the rise of cloud platforms (AWS, Azure, GCP), applications are designed to be cloud-native. Microservices are better suited to this architecture, leveraging containerization and orchestration tools for deployment.

4. DevOps and CI/CD:

-- DevOps practices and Continuous Integration/Continuous Deployment pipelines align well with microservices, enabling rapid updates to specific services without affecting the entire application.

5. Fault Tolerance:

-- Microservices provide better fault isolation. If one service fails, it doesn't bring down the entire system.

6. Containerization with Docker:

-- Tools like Docker and Kubernetes have made deploying and managing microservices much easier, allowing businesses to fully embrace the microservices approach.

Where Monolithic Is Still Relevant:

While microservices dominate modern architectures, monolithic architecture is still used in certain scenarios:

1. Small Applications or Startups:

-- For simple applications with limited scope and resources, a monolithic approach is faster and easier to develop and deploy.

2. Legacy Systems:

-- Many organizations still run legacy systems built on monolithic architecture, as transitioning to microservices can be costly and time-consuming.

3. Teams with Limited Expertise:

-- Small teams or those without expertise in managing distributed systems might prefer monoliths for simplicity.

Trends in Industry Usage:

1. Microservices + Docker:

-- Companies like Netflix, Amazon, and Uber rely heavily on microservices and Docker to run their scalable and distributed systems.

2. Hybrid Approach:

-- Some organizations adopt a modular monolith as a middle ground, where they keep the system unified but use Docker to package different modules for easier management.

3. Kubernetes and Orchestration:

-- With Kubernetes becoming the standard for container orchestration, microservices are easier to manage and deploy, further accelerating their adoption.

8-Aug-2024

Training Day - 76 Report:

Micro services Using Docker

Project: E-Mart Project Deploy Using Docker

1. Steps to Deploy

-- Vagrant up:

-- Likely the command to initialize and provision a Vagrant virtual machine.

-- Vagrant ssh → sudo -i:

-- Connect to the Vagrant virtual machine using SSH and switch to the root user

with sudo -i.

-- mkdir compose:

-- Create a directory named compose.

-- cd compose:

-- Change into the newly created compose directory.

-- touch docker-compose.yml:

-- Create an empty docker-compose.yml file.

-- vim docker-compose.yml:

-- Open the

2. Docker Compose Commands

-- docker-compose up -d:

-- Start all the services defined in the docker-compose.yml file in detached

mode (-d means run in the background).

-- Note: "not a folder name, keyword" is likely a clarification to avoid

confusion about the up command.

-- docker-compose ps:

-- Show the status of the running containers/services.

-- ip addr show:

-- Display the network interface details, which can be used to find the IP address

of the machine.

-- Project is live:

-- docker-compose down:

-- Stop and remove all the containers, networks, and services started by dockercompose up.

3. General Notes

-- The notes emphasize basic Docker Compose commands for managing the lifecycle of services.

-- A focus on structured steps ensures proper configuration and deployment.

11-Nov-2024

Training Day - 77 Report:

Understanding Shell Scripting: Definition, Purpose, and Benefits

Definition of Shell Scripting

Shell scripting is a method of automating tasks in Unix/Linux environments by writing a series of commands in a text file, which can be executed as a program. The "shell" refers to the command-line interface that allows users to interact with the operating system. Shell scripts can include various commands, control structures (such as loops and conditionals), functions, and variables, enabling users to create complex workflows and automate repetitive tasks.

Purpose of Shell Scripting

The primary purpose of shell scripting is to simplify and automate a wide range of tasks that would otherwise require manual intervention. This is particularly useful for system administrators, developers, and anyone who frequently interacts with the command line. Shell scripts can be used for:

1. Automation: Automating repetitive tasks like backups, file management, and system monitoring reduces the workload and minimizes the risk of human error.

2. Task Scheduling: Shell scripts can be scheduled to run automatically at specified times using tools like cron, allowing for regular maintenance and updates without manual effort.

3. System Administration: Administrators use shell scripts to manage system configurations, user accounts, and software installations, making it easier to maintain

multiple systems efficiently.

4. Batch Processing: Shell scripts can process large volumes of data in batch mode, enabling efficient data manipulation and analysis.

5. Environment Setup: They can configure the environment for development or deployment, ensuring that all necessary variables and settings are in place.

6. Integration: Shell scripts allow for the integration of various programs and tools, facilitating complex workflows that involve multiple steps and processes.

7. Portability: Shell scripts are often portable across different Unix-like systems, allowing users to run them without modification on various platforms.

8. Simplicity: Writing shell scripts can be simpler and quicker than developing fullfledged applications, especially for straightforward tasks.

Benefits of Shell Scripting

1) Efficiency: Automating tasks saves time and increases productivity by allowing users to focus on more complex and critical issues.

2) Consistency: Scripts ensure that tasks are performed in a consistent manner, reducing the likelihood of errors that can occur with manual execution.

3) Flexibility: Users can customize scripts to fit their specific needs, making it a versatile tool for a variety of tasks.

Cost-Effectiveness: Shell scripting is a cost-effective solution for automating tasks without the need for expensive software tools.

Deploying a Website Live on an Apache Server

Definition of the Script File

This shell script is designed to automate the setup of an Apache web server on a Unix/Linux system. It performs tasks such as updating the system, installing necessary packages, starting the Apache service, downloading a sample website template, and cleaning up temporary files after the installation process is complete.

Use and Benefits:

Use: This script is used to quickly and easily set up an Apache web server and deploy a sample website. It streamlines the process of installation and configuration, making it accessible even for those with limited experience.

Benefits:

-- Time-Saving: Automates repetitive tasks so you don’t have to enter commands manually.

-- Error Reduction: Reduces the chance of mistakes by executing predefined commands consistently.

-- Ease of Use: Allows users to set up a web server with just a single command, making it user-friendly.

-- Clean-up: Automatically removes temporary files after use, keeping the system organized.

Input:

Output:

12-Nov-2024

Training Day - 78 Report:

Deploying a Website Live on an NGINX Server

Description

This script automates the process of deploying a static website on a server using HTTPD (Apache). It installs necessary dependencies, downloads and unzips a website template, copies the files to the web server directory, and starts the HTTPD service to make the website live.

Use

This script can be used to quickly set up a static website for testing or small-scale

deployment. By executing it, a user can automate the installation of dependencies, deploy web files, and ensure the HTTPD service is up and running.

Benefits

1. Efficiency: Automates repetitive steps, making deployment faster.

2. Error Reduction: Minimizes human errors associated with manual setup.

3. Consistency: Ensures the same steps are followed every time, maintaining setup consistency.

4. Convenience: Simplifies server setup for those unfamiliar with manual deployment steps.

Input:

Output:

13-Nov-2024

Training Day - 79 Report:

What is a Variable?

In shell scripting, a variable is a named placeholder used to store data such as strings, numbers, or command outputs. Variables make scripts more dynamic by allowing the reuse of stored values and simplifying modifications.

Key Features:

1. No Type Declaration: Variables in shell scripting do not require data types (e.g., int, string).

2. Assignment: Use = for assignment without spaces around it.

3. Access: Retrieve the value using a $ prefix before the variable name.

Syntax:

Automated Website Deployment Script with Variable Assignments

Description

This script uses assigned variables to streamline the deployment of a static website. Variables like svc for the service name, package for required dependencies, website_url for the web template, and directory_path for the temporary storage location are defined at the beginning. This setup allows easy modification of values without needing to alter the script deeply. The script then installs necessary packages, starts and enables the HTTPD service, deploys web files from the specified URL, and cleans up temporary files after deployment.

Use

This script is useful for automating website deployment tasks on a server with minimal effort. By defining key values as variables, it allows for easy customization of the service name, package dependencies, website template URL, and directory paths. This makes it adaptable for similar deployments where these values might need changing.

Benefits

1. Easy Customization: Variables allow for quick updates to service names, URLs, or directories without modifying the entire script.
2. Efficiency: Automates setup steps like dependency installation and file deployment, reducing time and manual effort.
3. Reduced Errors: By centralizing key values in variables, the script minimizes the chance of typos or inconsistencies during configuration.
4. Reusable: The script can be reused across different deployments by simply adjusting variable values.
5. Scalability: Supports a modular approach, making it easy to add more variables if additional customization is required in future deployments.

Input:

Output:

Arguments ($1 $2 $3) Mean

Purpose

This file is a Bash script designed to demonstrate the handling and output of positional arguments ($1, $2, etc.) in a shell script. Positional arguments allow users to pass input values to the script when it is executed.

Contents and Functionality

1. Introduction Section

o Displays a visual separator ("~" characters) for formatting purposes.

o Outputs the script name ($0) and the provided arguments ($1 and $2).

2. Key Actions

o Prints the name of the script (stored in $0).

o Prints the first ($1) and second ($2) arguments supplied when the script is executed.

Input:

-- Command-line arguments: The script expects two arguments to be passed when running the command.

Example:

Here, arg1 will be $1, and arg2 will be $2.

Output

The script displays:

1. The script name (e.g., ./arguments.sh).

2. The first argument ($1).

3. The second argument ($2).

Example Output:

Use Case

This script is typically used:

-- To understand and debug how command-line arguments are passed to a Bash script.

-- As a foundation for scripts that require user input via arguments.

Package Installation Using User Arguments

Purpose

This script demonstrates how to install software packages using yum (a package manager for RHEL-based Linux systems) based on user-provided arguments. The script allows flexibility to specify additional packages directly as input when executing the script.

Contents and Functionality

1. Introduction Section

o Displays a message indicating that the script handles package installation using arguments.

2. Key Action

o Executes the yum install command with:

 wget (predefined package in the script).

 Additional packages passed as arguments ($1, $2).

3. Automatic Confirmation

o Uses the -y flag to automatically confirm installation prompts.

Input:

-- Command-line arguments: The script expects up to two additional package names passed when running the command.

Example:

Here:

1. package1 corresponds to $1.

2. package2 corresponds to $2.

Output:

The script installs:

1. wget (default).

Packages specified as arguments ($1, $2).

Use Case

This script is useful for:

-- Simplifying the installation of multiple packages with a single command.

-- Automating installations for developers or system administrators.

Automated Website Deployment Script Using Command-Line Arguments

Definition of Arguments

Arguments are values passed to a script at runtime that can be used within the script. In this script, the arguments are represented by $1, $2, and $3, which allow dynamic input for specific actions. Here, they correspond to the URL to download ($1), the file to unzip ($2), and the directory path to copy ($3).

Why We Use Arguments

Arguments make scripts flexible and reusable by allowing the user to specify different values each time the script runs. This avoids hardcoding and makes the script adaptable for various use cases without modification.

Use of the Script

This script automates the process of downloading, unzipping, and copying web files to the server's root directory by accepting URL, file, and directory as command-line arguments. This is especially useful for deploying different web applications or versions of a site on the same server setup.

Benefits

1. Flexibility: The script can handle various inputs, allowing it to be used with different URLs, files, and directory paths.

2. Reusability: The use of arguments makes it easy to reuse the script for different deployments without editing the code.

3. Efficiency: Speeds up deployment by automating dependency installation, file handling, and server setup.

4. Reduced Maintenance: Minimal changes are needed for new deployments, simplifying script maintenance and reducing the risk of errors.

Input:

Output:

14-Nov-2024

Training Day - 80 Report:

Understanding Variables in Shell Scripting

Definition

In shell scripting, variables are used to store data and reuse it throughout the script. This script demonstrates:

1. Standard Variable Substitution: Prints the value of a variable ($a).

2. Literal Output of Variable Name: Shows how to prevent variable substitution using single quotes or escape characters.

Example Breakdown:

-- a="chandan" assigns the string "chandan" to the variable a.

-- "Your Name is $a" displays the value of a.

-- 'Name is $a' treats $a as plain text.

-- "Your Name is \$a" uses the escape character (\) to print $a literally.

INPUT:

OUTPUT:

Using if Conditions in Shell Scripting

Introduction

This script demonstrates the use of the if condition in shell scripting to compare two userprovided values. By reading inputs and applying a conditional check, it determines whether

the values are equal or one is greater than the other, showcasing decision-making capabilities

in shell scripts.

Use

This script is useful for comparing two numerical inputs and performing conditional

operations based on their values. It can be adapted for scenarios such as:

1. Equality Checks: Determine if two user-provided values are the same.

2. Decision Making: Execute specific commands or display messages based on the

result of a comparison.

3. Input Validation: Useful in scripts where user inputs need to be compared or

evaluated for further processing.

Input:

Output:

Runtime Inputs in Shell Scripting

The uploaded file contains a shell script demonstrating how to handle runtime inputs in bash

scripting. It shows the use of various read commands to capture user inputs, such as names,

age, and gender, both with and without masking sensitive input.

Uses of Runtime Inputs in Shell Scripting

1. Interactive Applications: Enables scripts to interact with users during execution.

2. Dynamic Behavior: Allows a script to adapt its behavior based on user-provided

data.

3. Data Collection: Useful for capturing user-specific data like credentials, preferences, and configurations.

4. Automation: Facilitates parameter input for automating tasks without hardcoding values.

Benefits of Using Runtime Inputs

1. Flexibility: Eliminates the need to modify scripts for different inputs, making them reusable.

2. User-Centric: Increases script usability by allowing real-time user interaction.

3. Enhanced Security: Using masked inputs (e.g., read -sp) ensures sensitive data, like passwords, is not displayed on the screen.

4. Ease of Debugging: Simplifies testing by allowing varied inputs during runtime

Input:

Output:

18-Nov-2024

Training Day - 81 Report:

Understanding Loops in Bash Scripting

Loops are fundamental constructs in programming that allow repetitive execution of a block of code. They are essential for automating tasks, processing data, and managing iterative operations efficiently. In Bash scripting, loops help execute commands multiple times without manual intervention, enhancing script functionality and flexibility.

1. Introduction to for Loops in Bash

The for loop in Bash iterates over a list of items, executing a set of commands for each item in the list. This is particularly useful when performing repetitive tasks such as processing files, handling user inputs, or automating system operations.

Syntax of for Loop:

-- variable: A placeholder for the current item in the iteration.

-- item1 item2 item3: The list of items to iterate over.

-- do ... done: The block of commands to execute for each item.

2. Introduction to for While in Bash

The while loop in Bash scripting is a control flow statement that allows code to be executed repeatedly as long as a given condition evaluates to true. It is commonly used for tasks where the number of iterations is not predetermined, making it highly flexible.

Syntax of While Loop:

Example of For Loop:

Bash Script: Input and Output

Input:

The uploaded Bash script iterates over a list of names and outputs them with formatting and a delay. Here is the script content:

Script:

Output:

When the script is executed, it produces the following formatted output:

Example of While Loop:

Bash Script: Input and Output

Input (Script):

Output:

When you run this script, it will:

1. Start with i=1.

2. Continue the loop as long as i is less than or equal to 10.

3. Print the value of i with a delay of 1 second between each iteration.

Example Output:

19-Nov-2024

Training Day - 82 Report:

Introduction to GitHub

GitHub is a web-based platform that facilitates version control and collaborative software development. Built on top of Git, a distributed version control system, GitHub provides a host of tools for managing projects, tracking changes, and collaborating effectively.

Key Features of GitHub

1. Version Control:

o Tracks changes in your code over time.

o Allows reverting to previous versions if needed.

2. Repositories (Repos):

o A repository is a storage space for your project files.

o Can be public (accessible to everyone) or private (restricted access).

3. Collaboration:

o Multiple developers can work together on the same project.

o Features like pull requests, forks, and issues enhance teamwork.

4. Pull Requests:

o Used to propose changes to a codebase.

o Enables discussion, code review, and testing before merging changes.

5. Issues and Bug Tracking:

o A feature for reporting and tracking bugs or suggesting improvements.

6. Branching and Merging:

o Allows developers to work on different features or fixes without affecting the main codebase.

o Changes can be merged back into the main branch after review.

7. Actions:

o Automates workflows such as CI/CD (Continuous Integration and Continuous Deployment).

8. Markdown Support:

o Provides formatting for project documentation, README files, and comments.

9. Community and Open Source:

o A vast library of open-source projects to contribute to or use as inspiration.

10. Integrations:

o Supports integration with tools like Slack, Jira, and third-party CI/CD systems.

1. Make a new repository in GitHub.

2. Open Git Bash and check the Git version:

git –version

3. Create a new folder in your system related to the repository.

4. Set up global configuration:

git config –list

5. Initialize a new Git repository:

git init

6. Create files in your local machine.

7. Check the status of your repository:

git status

8. Add files to the staging area:

git add <file-name>/< . >

9. Check the status again:

git status

10. Copy the repository HTTPS link from GitHub.

11. Add the remote repository:

git remote add origin <repo-link>

12. Commit the changes with a message:

git commit -m "<message>"

13. Check the status:

git status

14. Push changes to the remote repository:

git push -u origin <branch-name>

15. Add a new file to the repository.

16. Modify or make changes to the file.

17. View logs:

git log

git log –oneline

18. Check the repository configuration:

cat .git/config

19. Clone a repository:

1. Create a folder:

mkdir clone-files

2. Clone the repository:

git clone <git-hub-link>

20-Nov-2024

Training Day - 83 Report:

SSH and Branching:

1. Generate an SSH key:

ssh-keygen

2. View SSH keys:

ls .ssh/

3. Display the public key:

cat .ssh/id_ed25519.pub

4. Copy the key and add it to GitHub:

1. Go to the repository settings in GitHub.

2. Click on 'SSH and GPG keys'.

3. Add a new SSH key with a title and paste the key.

5. Create a folder for the SSH repository.

6. View all files including hidden ones:

ls –a

7. Copy the SSH link of the repository.

Branches:

Introduction to Branches in GitHub

In GitHub (and Git), branches are a core feature used to manage different versions of a codebase. A branch represents an independent line of development within a repository. They allow developers to work on new features, bug fixes, or experiments without affecting the main codebase.

Key Concepts of Branches

1. Default Branch:

o Typically named main or master.

o Represents the "production-ready" version of the code.

o It’s the branch that other branches are usually merged into.

2. Feature Development:

o Create branches for individual tasks or features, e.g., feature/login-page or bugfix/error-404.

o Helps in isolating work and avoiding conflicts with other developers.

3. Branching Workflow:

o Developers can switch between branches to work on different features simultaneously.

o Each branch maintains its history of changes.

4. Collaboration:

o Multiple team members can work on their own branches and later merge them into the main branch.

o Encourages parallel development.

Why Use Branches?

-- Isolation: Prevents unfinished code from affecting the stable version.

-- Collaboration: Teams can work on different features independently.

-- Code Review: Pull requests from branches allow for better code review processes.

-- Experimentation: Test new ideas without disrupting the main codebase.

1) Create a new branch:

git branch <new-branch-name>

2) Switch to the new branch:

git switch <branch-name>

3) Push the branch to the remote repository:

git push origin <new-branch-name>

4) Add an untracked file:

1. Create a file using touch.

2. Stage the file:

git add .

3. View status:

git status

4. Unstage the file:

git restore --staged <file-name>

5. View status again:

git status

5) Merge the new branch into the master branch.

6) Unmerge the new branch from the master branch.

7) Add a file and merge changes:

Use the command:

git reset --hard <commit-id>

21-Nov-2024

Training Day - 84 Report:

Introduction to Maven Build Tool

Maven is a tool designed to help with building software. It simplifies the process of turning source code into working programs. It automates tasks like compiling code, running tests, and packaging the program into a format that can be distributed.

Maven is emphasized as a build tool specifically designed for Java projects. It helps automate tasks such as compiling code, running tests, and creating packages (like JAR, WAR files).

Why Do We Need Maven?

Maven simplifies and automates the software build process, managing dependencies, standardizing workflows, and ensuring efficient multi-project handling.

Overview of Build Tools

This page describes different tools used for automating the software build process:

-- Maven: Primarily for Java projects, it uses an XML file to manage the build process.

-- Ant: Another tool for Java with XML configuration.

-- MsBuild: A Microsoft tool for building applications.

-- Gradle: Uses Groovy-based DSL (Domain-Specific Language) for configuration.

-- NANT: A build tool for the Windows .NET platform.

-- Make: Commonly used for creating executable programs from source code.

These tools make software development easier and more consistent by automating repetitive tasks.

Maven Phases:

Maven organizes its tasks into a series of steps called "phases." Each phase accomplishes a specific goal in the software build process:

1. validate: Ensures the project has all necessary information and is correct.

2. compile: Converts the source code into executable code.

3. test: Runs unit tests to check the code without deploying it.

4. package: Packages the compiled code into a distributable format (e.g., JAR).

5. verify: Ensures integration test results meet quality standards.

6. install: Saves the package in a local repository for use in other projects.

7. deploy: Uploads the package to a remote repository for sharing with others.

Installing Maven

To use Maven, you need to:

1. Install it using a package manager (like Chocolatey, Homebrew, yum, or apt) based on your operating system.

2. Have the Java Development Kit (JDK) installed on your system, as Maven depends on Java.

The process is simple and ensures developers can start using Maven efficiently.

Step-by-Step Guide to Install Maven via CLI

1) Update the System Packages

Before installing any software, it's crucial to ensure the system packages are updated.

2) Install Java Development Kit (JDK)

Java is required for Maven to function as Maven runs on the Java Virtual Machine

(JVM).

3) Install Apache Maven

Maven is a build automation tool used for Java projects.

Commands:

4) Install Git and Clone the Repository

Git is required to download the project repository.

Commands:

5) Navigate to the Project Directory

Change into the directory where the project files are stored.

Commands:

6) Run Tests

Use Maven to run the tests in the project.

Commands:

Explanation:

mvn test: Executes the tests defined in the project.

ls target/: Lists the contents of the target directory, where the test outputs are stored.

7) Build the Project

Compile and package the project into a distributable format.

Commands:

Explanation:

mvn install: Builds the project and generates a package (e.g., a JAR or WAR file).

ls target/: Confirms that the build artifacts are in the target directory.

8) Clean the Build

Remove the target directory to clean the build environment.

Commands:

9) Validate Maven Project

Validates that the project's structure and dependencies are correct.

22-Nov-2024

Training Day - 85 Report:

What is Jenkins, Why Do We Need It, and an Explanation of Continuous Integration?

What is Jenkins?

Jenkins is an open-source automation server widely used for Continuous Integration (CI) and Continuous Delivery (CD) in software development. It automates repetitive tasks involved in building, testing, and deploying applications, streamlining the software development process

Why Do We Need Jenkins?

1. Automates Build and Deployment:

o Jenkins automates the process of compiling code, running tests, and deploying applications, saving time and reducing manual errors.

2. Supports Continuous Integration:

o Ensures that code changes are continuously merged and tested, maintaining a stable codebase.

3. Integration with Tools:

o Jenkins integrates with various tools like Git, Maven, Docker, Kubernetes, and others to manage the entire software lifecycle.

4. Cross-Platform:

o Runs on Windows, macOS, Linux, and other platforms, providing flexibility for different environments.

5. Extensible:

o Offers a wide variety of plugins (over 1,500) to customize and extend its functionality.

6. Scalable:

o Can distribute builds across multiple machines to handle large-scale projects efficiently.

Key Features of Jenkins

1. Pipeline as Code:

o Use Jenkinsfiles to define build pipelines as code, enabling better version control.

2. Distributed Builds:

o Execute builds across multiple nodes, improving performance.

3. Integration with SCM:

o Works seamlessly with Source Code Management (SCM) tools like Git, GitHub, or Bitbucket.

4. Automated Testing:

o Runs automated tests to detect and fix issues early.

5. Dashboard and Notifications:

o Provides a web-based dashboard for monitoring builds and integrates with communication tools for alerts.

What is Continuous Integration (CI)?

Continuous Integration (CI) is a software development practice where developers integrate code into a shared repository frequently, often multiple times a day. Automated builds and tests are triggered with every integration to detect issues early.

Why is Continuous Integration Important?

1. Early Bug Detection:

o Frequent integrations catch bugs earlier in the development cycle, reducing costs and effort.

2. Stable Codebase:

o Automated testing ensures that the main branch remains stable and deployable.

3. Faster Development Cycles:

o Developers can focus on writing code instead of spending time on manual testing and debugging.

4. Improved Collaboration:

o Encourages teamwork by providing a centralized repository for all code changes.

Installation of Jenkins:

1. Switch to root: sudo –i

2. Update packages: apt-get update –y

3. Install Java: apt-get install openjdk-21-jdk

4. Verify Java: java –version

5. Download Jenkins GPG key: sudo wget -O /usr/share/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

6. Add Jenkins repo: echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

7. Update packages: sudo apt-get update

8. Install Jenkins: sudo apt-get install jenkins.

9. Check Status: service Jenkins status

10. Check IP Addr: ip a s

11. Copy ip & paste it browser with part no 8080 : See Jenkins Interface

12. View password: cat /var/lib/jenkins/secrets/initialAdminPassword

13. Paste it & Continue

14. Select Plugins to install

15. Select what you want to install and click to install

16. Create First Admin User

17. Save & Finish (if u want to change your url then change it)

18. Click Start using Jenkins

25-Nov-2024

Training Day - 86 Report:

Jenkins is an open-source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration, and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat, or by default as a stand-alone web-application in co-bundled Eclipse Jetty. It supports version control tools, including CVS, Subversion, Git, Mercurial, Perforce, ClearCase, and RTC, and can execute Apache Ant, Apache Maven, and set based projects as well as arbitrary shell scripts and Windows batch commands.

BUILDS

Builds can be triggered by various means, for example:

-- a webhook that gets triggered upon pushed commits in a version control system

-- scheduling via a cron-like mechanism

-- requesting a specific build URL.

-- after the other builds in the queue have completed

-- invoked by other builds

PLUGINS

Plugins have been released for Jenkins that extend its use to projects written in languages other than Java. Plugins are available for integrating Jenkins with most version control systems and bug databases. Many build tools are supported via their respective plugins. Plugins can also

change the way Jenkins looks or add new functionality. There are a set of plugins dedicated for the purpose of unit testing that generate test reports in various formats (for example, JUnit bundled with Jenkins, MSTest, NUnit, etc. and automated testing that supports automated tests. Builds can generate test reports in various formats supported by plugins (JUnit support is currently bundled) and Jenkins can display the reports and generate trends and render them in the GUI.

MAILER

Allows configuring email notifications for build results. Jenkins will send emails to the specified recipients whenever a certain important event occurs, such as:

-- Failed build.

-- Unstable build.

-- Successful build after a failed build, indicating that a crisis is over.

-- Unstable build after a successful one, indicating that there's a regression.

CREDENTIALS

Allows storing credentials in Jenkins. Provides a standardized API for other plugins to store and retrieve different types of credentials.

MONITORING EXTERNAL JOBS

Adds the ability to monitor the result of externally executed jobs

SSH AGENTS

This plugin allows managing agents (formerly known as slaves) running on nix machines over SSH. It adds a new type of agent launch method. This launch method will

-- Open a SSH connection to the specified host as the specified username,

-- Check the default version of Java for that user,

-- [not implemented yet] If the default version is not compatible with Jenkins's agent.jar, try to find a proper version of Java

-- Once it has a suitable version of Java, copy the latest agent.jar via SFTP (falling back to scp if SFTP is not available),

Start the agent process

26-Nov-2024

Training Day - 87 Report:

Click on dashboard & Click Manage Jenkins

1. Menu (Left Side):

-- New Item: Allows users to create a new Jenkins project, which could be a Freestyle project, Pipeline, etc.

-- Build History: Displays a log of past builds. Users can click this to view the results and status of previous jobs.

-- Manage Jenkins: This navigates to the Jenkins management and configuration settings (like in the previous image).

-- My Views: Users can create custom views for managing different jobs or seeing data tailored to their preferences.

2. Build Queue:

No builds in the queue: Indicates that no jobs are currently in the queue for execution. If jobs were queued for execution, they would appear here.

3. Build Executor Status:

Shows the number of available or busy executors (agents that run the build jobs). In this case, it shows 0/2, indicating that both executors are idle and no jobs are running.

4. Main Job List:

This section provides details about Jenkins jobs and their build history.

-- All (+ icon): Allows you to create new jobs or filter the display to show specific jobs.

-- Columns:

-- S (Success Indicator): The green checkmark indicates that the last build was successful.

-- W (Warning Indicator): This column may display a warning symbol if the build encountered issues but was not a total failure.

-- Name: The name of the Jenkins job. In this case, the job name is "ist."

-- Last Success: Indicates the time since the job was last successfully completed. Here, it was 5 hours and 7 minutes ago, and the build number (#4) is provided for reference.

-- Last Failure: Indicates the time since the job last failed. In this case, the job failed 5 hours and 11 minutes ago, with build number (#1).

-- Last Duration: Displays the amount of time it took for the last build to complete. In this case, the last successful build ran for 3.9 seconds.

-- Play Icon: A button that allows the user to manually trigger the job to run again.

5. Icon Size:

S, M, L: Options to change the icon size for better visibility on the Jenkins dashboard (small, medium, or large).

6. Add Description (Top Right):

Allows users to add a description for the current job or dashboard view, which can be helpful for tracking purpose or adding notes.

This dashboard provides a concise overview of your Jenkins jobs, their recent build history, and their current status.

27-Nov-2024

Training Day - 88 Report:

1. Click on dashboard & Click Manage Jenkins

2. Click on Tools

3. click on JDK installations & Click Add jdk : Enter Name & Location of jdk

4. Click on git installations & Click on Add Git

5. Then Fill Details:

6. Click on maven installations & Click Add Maven

7. Enter Name and version

8. After that click on apply & then save

9. Go to dashboard & Click on New Item:

10. Enter New Item name and select freestyle Project & Click Ok

11. Enter Description & then Select JDK

12. Click on source code management & select git then Enter Git repository URL

and Enter Branch name

13. Click on Add Build Setups

14. Select Invoke top-level Maven Targets

15. Select Maven Version & In Goals write (Install) and Click to Save

16. Click Save and Apply

17. Click to Build Now Option

18. Now, Your Build now is done

19. Go to Workspace and see your all project files

20. Click on target and see your .war files

28-Nov-2024

Training Day - 89 Report:

Now if you want to see it your war files in front then follow these steps:

1. Click on Configure and Click Add post-build action

2. Click on Archive the aircrafts

3. Type **/*.war then save & Apply

4. Now you see your file in front

29-Nov-2024

Training Day - 90 Report:

Version Control System (VCS)

In Jenkins, a Version Control System (VCS) is used to manage the source code and configuration files for projects. Jenkins integrates with various VCSs like Git, SVN, Mercurial, and others to automate builds and deployments. By connecting to a VCS, Jenkins can pull the latest code changes, trigger automated pipelines, run tests, and deploy applications based on the version-controlled files.

Jenkins uses VCS to:

-- Fetch code from repositories.

-- Track specific branches, commits, or tags.

-- Automatically trigger builds on code changes (e.g., via webhooks or polling).

This integration ensures consistency and enables continuous integration and delivery (CI/CD) by automating processes around version-controlled code.

DIFFERENT TYPES OF METHOD OF VERSIONS

1. MANUAL VERSIONING

2. PARAMETER VERSIONING

3. PLUGIN VERSIONG

1

st: Implementation of manual versioning

1. Click on dashboard and select job(mavenbuild)

2. Then click on job(mavenbuild) & click on configure

3. After that click on add build & select option “execute shell”

4. Write command in execute shell & save it

5. Click on build now

6. Go to workspace & you see versions folder

7. Then click on versions & you see your .war file

2

nd: Implementation of parameter versioning

1. Go to mavenbuild & click on configure select option “This project is parameterized”

2. After that click on add parameter & select multi-line string parameter

3. Then add name, default value, description

4. Then write command on “execute shell” & click on save

5. Click on build with parameters, enter any versions & click on build

6. After that click on workspace choose versions & click on it you get your file.

3

rd: Implementation of parameter versioning

1. Go to dashboard, click on Jenkins manage & click on plugins

2. Click on Available plugins & search timestamp select it and install it

I already installed so, it doesn't show

3. Go back to Jenkins manage & click on system

4. Select build timestamp & enable build stamp, change pattern then save it.

5. Go back to configure & write command in "execute shell" and save it.

6. After that click on workspace choose versions folder & open it you get your file.

2-Dec-2024

Training Day - 91 Report:

Python Fabric is a library and command-line tool for streamlining the process of executing remote or local shell commands and automating tasks, particularly for system administration and deployment workflows. It enables Python developers to write simple or complex task automation scripts efficiently.

Why Do We Need Python Fabric?

1. Task Automation: Fabric is useful for automating repetitive system tasks like application deployment, server setup, or log analysis.

2. Remote Command Execution: It simplifies the process of running commands on remote machines using SSH.

3. Multi-Host Orchestration: Fabric can execute commands across multiple servers simultaneously, ideal for managing clusters or distributed systems.

4. Integration with Python: It combines Python's scripting capabilities with remote operations, allowing for complex logic in deployment or maintenance scripts.

Key Features of Python Fabric

1. Remote Execution: Execute shell commands on remote servers via SSH with ease.

2. Local Execution: Run commands locally for setup or testing purposes.

3. File Transfers: Upload and download files to and from remote servers.

4. Parallel Execution: Execute tasks on multiple servers concurrently, enhancing efficiency.

5. Custom Scripts: Write Python scripts to define reusable tasks or workflows.

6. Interactive SSH: Support for interactive SSH sessions for real-time debugging or manual intervention.

7. Secure Connection: Utilizes paramiko, a Python SSH library, ensuring encrypted and secure communication.

8. Extensibility: Easy integration with other Python libraries for extended functionality.

Typical Use Cases

1. Application Deployment:

o Automating deployment pipelines for web or database applications.

2. Server Management:

o Setting up or updating software packages, configurations, or services.

3. System Monitoring:

o Collecting logs, monitoring disk usage, or analyzing system health.

4. Backup and Recovery:

o Automating file transfers for backup purposes.

5. Multi-Server Orchestration:

o Managing multiple servers with a single command or script.

3-Dec-2024

Training Day - 92 Report:

To install Fabric on a Linux system, follow these steps:

python3 --version

pip3 --version

sudo apt update

sudo apt install python3 python3-pip

Create a simple Python program in the terminal:

Another example of Python print function:

Exit terminal or file editing:

Create a directory for Python scripts:

Create an empty Python file:

Edit the Python file using vi editor:

Inside the vi editor, add the following code:

Run the Python script:

EXAMPLES:

-- #!/usr/bin/python: This is the shebang that indicates the script should be run with Python.

-- import os: The os module allows you to interact with the operating system.

-- input(): Used to get input from the user, int() converts the input into an integer.

-- os.system("ls"): If a is greater than b, the script will run the ls command (list the directory contents).

-- os.system("pwd"): If a is not greater than b, the script runs the pwd command to print the current working directory.

EXAMPLES:

-- #!/usr/bin/python: Shebang for running the script with Python.

-- import os: The os module provides a way to use operating system-dependent functionality.

-- input(): Prompts the user to input a path (file or directory).

-- os.path.isdir(): Checks if the given path is a directory.

-- If true, it prints "It is a Directory".

4-Dec-2024

Training Day - 93 Report:

What is Ansible?

Ansible is an open-source IT automation tool used for configuration management, application deployment, task automation, and orchestration. It uses simple, human-readable YAML files (called playbooks) to define tasks and does not require agents to be installed on target machines, making it lightweight and easy to use.

Why Use Ansible?

1. Agentless: It uses SSH to manage nodes, so no additional software is required on the managed systems.

2. Simple Configuration: YAML-based configuration makes it easy to learn and use.

3. Idempotent: Ensures tasks are applied only when necessary, avoiding repeated changes.

4. Efficient for Multi-Node Management: Automates repetitive tasks across multiple systems.

5. Wide Use Cases: Can be used for provisioning, configuration management, application deployment, and continuous delivery.

Ansible is a powerful automation tool with a wide range of features that make it suitable for IT configuration management, deployment, and orchestration. Below are its key features:

Features of Ansible

1. Agentless Architecture

-- No agents required: Ansible does not require any software or agents to be installed on

managed nodes.

-- Uses SSH (or WinRM for Windows) to communicate with remote systems.

-- Simplifies setup and reduces resource consumption.

2. Simple and Human-Readable

-- Tasks and configurations are defined in YAML files (playbooks), which are easy to read and write.

-- No need for specialized coding skills; the learning curve is minimal.

3. Idempotency

-- Ensures tasks are applied only if needed, avoiding unnecessary changes.

-- Guarantees that the system state is consistent even if a playbook is run multiple times.

4. Cross-Platform Support

-- Supports a wide variety of systems, including Linux, Windows, macOS, networking devices, and cloud platforms.

-- Can manage hybrid environments seamlessly.

5. Modular and Extensible

-- Comes with a wide range of built-in modules to manage tasks like package management, file operations, user creation, and service management.

-- Easily extended with custom modules or third-party modules from Ansible Galaxy.

6. Declarative and Procedural Support

-- Define the desired state of the system in a declarative way (e.g., "ensure Nginx is installed and running").

-- Alternatively, execute procedural tasks with precise steps.

7. Scalability

-- Can manage a single node or thousands of nodes efficiently.

-- Supports inventory grouping for managing servers based on roles, environments, or other attributes.

8. Built-In Security

-- Uses SSH (secure by design) for Linux systems and WinRM for Windows.

-- Provides Ansible Vault for encrypting sensitive data like passwords and API keys.

9. Integration with DevOps and Cloud

-- Supports popular DevOps tools like Jenkins, Docker, Kubernetes, Terraform, and Git.

-- Integrates with cloud platforms such as AWS, Azure, Google Cloud, and OpenStack

for provisioning and management.

10. Orchestration

-- Handles complex workflows involving multiple servers and services.

-- Allows orchestration across diverse infrastructure components, ensuring they work in

harmony.

5-Dec-2024

Training Day - 94 Report:

Installing Ansible un Ubuntu Machine:

You need to install the Ansible software on the machine that will serve as the Ansible control

node. From your control node, run the following command

Commands:

-- $ sudo apt-get update

-- $ sudo apt install software-properties-common

-- $ sudo add-apt-repository --yes --update ppa:ansible/ansible

-- $ sudo apt-get install ansible –y

Next, check the version of the installed software using command below

-- $ ansible –version

Here it should reflect the ansible version on your console.

Exercise 1:

-- Make a File:

-- Run File:

ansible web001 –m ping -i inverntory

-- Output Screen:

Exercise 2:

-- Make a File:

-- Run File:

ansible web001 –m ping -i inverntory

ansible * –m ping -i inverntory

ansible all –m ping -i inverntory

ansible “web*” –m ping -i inverntory

-- Output Screen:

6-Dec-2024

Training Day - 95 Report:

Exercise 3: (Grouping)

-- Make a File:

-- Run File:

ansible webservers –m ping -i inverntory

ansible dbservers –m ping -i inverntory

ansible dc_oregion –m ping -i inverntory

ansible “web*” –m ping -i inverntory

-- Output Screen:

Exercise 4: (Variables)

-- Make a File:

-- Run File:

ansible webservers –m ping -i inverntory

ansible dbservers –m ping -i inverntory

ansible dc_oregion –m ping -i inverntory

ansible “web*” –m ping -i inverntory

-- Output Screen:

9-Dec-2024

Training Day - 96 Report:

Exercise 5: (Adhoc Commands in Ansible)

https://docs.ansible.com/ansible/latest/command_guide/intro_adhoc.html

change the hostname of your all machines if u want

Manage Packages Commands:

For install:

ansible <machine-name> -m ansible.builtin.yum -a "name=<webserver-name> state=present"

appu nu become lgana penda ehnu run karn leyi (become means sudo/root user)

ansible web001 -m ansible.builtin.yum -a "name=httpd state=present" –become

output ist time:

2

nd time run karn ch khduga eh already done aa

For remove/ delete:

ansible web001 -m ansible.builtin.yum -a "name=httpd state=absent" –become

Server on / off command:

ansible web001 -m ansible.builtin.service -a "name=httpd state=started" –i inventory –

b

Output:

Now hun je appa ek group ch install krni howe services --- and yeh kisi ch install hegi

aa tah thik aa yeh nnhi aa tah install krdu service --- agr sbb ch install aa tah kise ch

kuch nhi kru

Output:

Browser o/p:

For remove service in group:

10-Dec-2024

Training Day - 97 Report:

Exercise 5:

File and project deploy using Ansible :

Machines nu http protocols allow krni security ch

Copy krna project /var/www/html location ch

Output:

Done uploading:

11-Dec-2024

Training Day - 98 Report:

Playbook file

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html

What are playbooks?

Ansible is an orchestration tool. It needs a list of tasks/instructions to perform on the machines listed in the inventory file. You can execute a task with Ansible more than once using a playbook. Playbooks offer repeatable, reusable, simple configuration management. It can be

used for configuration management, orchestrating steps of any manual process on multiple machines in synchronous or asynchronous order.

A playbook is written in YAML format. It is composed of one or more 'plays' in an ordered list. And one play contains a set of tasks that runs on a group of machines.

Different YAML Tags

Let us now go through the different tags in a playbook-

-- name: Logical name of the task which specifies what this playbook will be doing

-- hosts: This specifies the lists of hosts against which we want to run the task

-- vars: This allows you to define and use variables in your playbook

-- tasks: Tasks are a list of actions the playbook will perform

In an ad-hoc command setup, we need to run commands repeatedly to achieve the same configuration on different machines. This means we have to manually execute commands every time we want to make changes or ensure consistency.

However, with a playbook file, we can write the commands in a file and use it to execute the desired tasks on multiple machines in one go. This approach saves effort and ensures consistency across all machines, as the same playbook file can be reused whenever needed. Essentially, by creating and keeping a playbook file, we can streamline the process and use it whenever required instead of running ad-hoc commands repeatedly.

Exercise 6:

1. Make a playbook yaml file

2. Run the command:

Ansible-playbook –I inventory <playbook-file-name>

3. Output:

4. Readable Form:

To make the output more readable and detailed, you can use the -v option (verbose mode) with the ansible-playbook command.

Command:

ansible-playbook -i inventory <playbook-file-name> -v

12-Dec-2024

Training Day - 99 Report:

Exercise 7:

In playbook file add lines For Dbserver

Run the command:

Output:

Exercise 8: (super global variable bnane sikhange)

Make a group_vars directory.. it is predefined !

Ls

Cd group_vars

Touch all (all file is complousary bnania kyuki playbok file nu access krn leyi)

Vi all

Cat

Now hun aapa playbook file edit krange

Debug predefined variable use krange and ek time ch ek hi msg print ho skda

Now run the file:

Output:

13-Dec-2024

Training00 Report:

Main imp:

If ur variables is also present in playbook file (grup_vars – all de nal nal)

Then ansible phle tuhade playbook aale variables chku ga then all file aale super global variables chkkuga

16-Dec-2024

Training01 Report:

INTRODUCTION DOCKER

What is Docker?

Docker is an open-source platform designed to automate the deployment, scaling, and management of applications. It uses containerization technology to package applications and their dependencies into a standardized unit called a container. This ensures that the application runs consistently, regardless of the environment in which it's deployed, such as on a developer's local machine, a testing server, or a production cloud server.

At its core, Docker enables developers to package their applications along with all necessary libraries, frameworks, and system tools into lightweight containers. These containers can run on any system that supports Docker, making it possible to eliminate the classic “it works on my machine” problem in software development.

What is a Docker Container?

A Docker container is a runnable instance of an image. Containers encapsulate the application and its dependencies, running in an isolated environment. Docker containers are lightweight because they share the host operating system's kernel but run independently of each other. Each container has its own filesystem, system resources, and process space, ensuring isolation and security.

Why Use Docker?

Docker is primarily used for:

1. Consistency Across Environments:

-- One of the major challenges in software development is ensuring that an application behaves consistently across different environments. This includes development, staging, and production environments. With Docker, you can package the application along with its entire ecosystem (such as libraries, system tools, and settings) into a container, ensuring that it runs the same way everywhere.

2. Microservices Architecture:

-- In modern application development, the microservices architecture is gaining popularity. Microservices break large, monolithic applications into smaller, independent services that communicate via APIs. Docker containers are ideal for running microservices because each service can be packaged and run in its own isolated environment. This isolation allows developers to update and scale individual services without affecting the entire application.

3. Streamlining Development and Testing:

-- Docker is great for setting up development and testing environments. Developers can quickly spin up containers that mirror the production environment, making it easier to test applications in conditions identical to those in production. Docker Compose, an extension of Docker, allows developers to define multi-container applications and manage them easily during development and testing.

4. Simplified Continuous Integration and Deployment (CI/CD):

-- Docker integrates well with CI/CD pipelines, allowing teams to automate the building, testing, and deployment of applications. By using Docker images in the CI/CD pipeline, developers can build and test code in a container that is identical to the production environment. This minimizes deployment errors and

improves the reliability of releases.

5. Portability and Scalability:

-- Containers built with Docker can run anywhere, on any machine that supports Docker. Whether you're running the container on your local machine, a virtual machine, or a cloud server, it will behave the same way. Docker also integrates with orchestration tools like Kubernetes to help scale applications efficiently.

Advantages of Docker

1. Consistent Development and Production Environments: Docker eliminates the problem of different environments by packaging everything the application needs to run into a container. Whether it's a developer’s laptop, a testing server, or a production server in the cloud, the container ensures that the application behaves the same way. This consistency helps developers and operations teams avoid "environment drift," where minor differences between development and production environments cause unexpected errors.

2. Efficient Resource Utilization: Docker containers are more lightweight than traditional virtual machines (VMs). Instead of creating a separate OS for each container, Docker containers share the host machine's kernel. This leads to reduced overhead, allowing multiple containers to run on the same host without consuming significant additional resources. As a result, Docker can run more containers on a single physical machine than virtual machines.

3. Faster Start-up and Deployment Times: Docker containers start almost instantly compared to virtual machines, which need to boot an entire OS. This rapid startup speed makes Docker ideal for dynamic environments where applications need to scale quickly

in response to changing workloads. This also means that Docker containers are easier to deploy, reducing the time it takes to push changes from development to production.

4. Isolation and Security: Each Docker container runs in isolation, with its own

filesystem, memory, CPU allocation, and network stack. This isolation ensures that issues in one container do not affect others. For example, if a container crashes or encounters a bug, it will not bring down the entire system or other containers. Docker also uses kernel-level security features, such as control groups (cgroups) and namespaces, to enhance isolation between containers.

5. Simplified Application Deployment: Docker simplifies the deployment of complex, multi-component applications. With tools like Docker Compose, you can define an entire multi-container application, including the application server, database, caching server, and load balancer, in a single configuration file (docker-compose.yml). This configuration can be shared across development, testing, and production environments, ensuring consistent deployment with minimal effort.

6. Facilitates Microservices Architecture: Docker is widely adopted in microservicesbased architectures. Each microservice can run in its own container, which can be built, tested, and deployed independently of other services. This allows for greater flexibility in how applications are built, scaled, and maintained.

7. Support for Multi-cloud and Hybrid Cloud Deployments: Docker provides flexibility in terms of where applications can run. Containers are platform-agnostic and can run on any infrastructure that supports Docker, whether it's on a developer's machine, in a data center, or in any cloud provider's environment (AWS, Azure, Google Cloud, etc.). This makes it easier to adopt hybrid or multi-cloud strategies, where applications are deployed across multiple cloud providers or between on-premise and cloud environments.

Key Docker Concepts

1. Docker Images:

-- A Docker image is a read-only template used to create containers. An image contains everything needed to run a container, including the operating system, application code, libraries, and dependencies. Images are built from Dockerfile instructions and can be stored in a Docker registry (such as Docker Hub) for

sharing and reuse.

2. Docker Containers:

-- A Docker container is a runnable instance of a Docker image. Containers are lightweight and portable, providing isolation between applications and their environment. Multiple containers can be run on a single host, and they share the host OS's kernel but have their own process and file namespace.

3. Dockerfile:

-- A Dockerfile is a text file that contains a series of instructions used to build a Docker image. Each instruction in a Dockerfile creates a new layer in the image. For example, you might use a FROM instruction to base your image on an existing image (like python:3.9-slim), and a COPY instruction to copy files from your local machine into the image.

4. Docker Compose:

-- Docker Compose is a tool that allows you to define and manage multi-container applications. Using a docker-compose.yml file, you can specify how to configure and run multiple services (like a web server, database, and caching layer) as a single application. This simplifies the orchestration of complex applications.

INSTALLATION OF DOCKER

-- Search docker.com on chrome & you get interface like that

-- Click on docs mention on the upper right of the interface

-- Go to Docker Engine & click on install

-- Click on Ubuntu

-- Copy that setup a docker commands

-- Go to git bash and run the command Ubuntu

-- Paste the docker installation command & installation start

-- After that check docker install or not / check the version of it

-- Check Docker service status:

This checks the status of the Docker service to verify if it's running.

17-Dec-2024

Training02 Report:

-- List running containers:

This lists all the Docker containers currently running on your machine.

-- Run "hello-world" Docker container:

This runs a simple container that prints "Hello from Docker!" to check if Docker is working.

-- List available Docker images:

This command lists all Docker images that are currently available on your system.

-- Remove Docker images:

This command deletes a Docker image from your system.

-- Start a Docker container:

This command starts an existing Docker container by referencing its container ID.

-- Stop a Docker container:

This command stops a running container.

-- Remove a stopped Docker container:

This command removes a container that is no longer running.

MySQL Container Setup:

-- Pull MySQL Docker image:

This downloads the latest MySQL image from Docker Hub.

-- Run MySQL container:

This runs a MySQL container in the background (-d), exposing port 3306 for MySQL.

-- Connect to a running MySQL container:

This allows you to execute commands inside the running MySQL container. It opens the MySQL prompt where you can log in as root.

-- MySQL connection example (external):

This connects to the MySQL server running in Docker from an external client using the host IP and port 3306.

Build a Docker Image:

-- Steps to create a custom Docker image:

-- Create a directory for your Docker image:

You create a directory for the project where your Docker image files (such as Dockerfile) will reside.

-- Create a Dockerfile:

Create a file named Dockerfile (the blueprint of the Docker image), and open it in an editor like vim to add instructions.

-- Writing the Dockerfile:

-- FROM ubuntu:latest: This tells Docker to base the image on the latest version of Ubuntu.

-- CMD ["echo", "Welcome to Docker"]: This sets the command that will run when the container starts. It will print "Welcome to Docker" to the console.

-- Build the Docker image:

This command builds the Docker image using the instructions in the Dockerfile located in the current directory (.). The image is tagged with a name (<image-name>) and

version (v1).

-- Run the Docker container:

This runs the container from the Docker image you built. The output will display "Welcome to Docker," as defined in the CMD instruction of the Dockerfile.

18-Dec-2024

Training Report:

SOME EXAMPLES:

Project 2

-- Create a directory for Project 2:

Create a new directory named Project2 and navigate into it.

-- Create a Dockerfile:

Create an empty Dockerfile using touch and then open it in an editor (vim) to add instructions.

-- Write the Dockerfile:

FROM ubuntu:latest: This defines the base image for the container, which is the latest version of Ubuntu.

ENTRYPOINT ["echo"]: This sets the default executable to be echo. The ENTRYPOINT instruction ensures that the container runs echo when started.

-- Build the Docker image:

This command builds a Docker image from the Dockerfile in the current directory (.), and tags it as chandan:v2.

-- Run the Docker container:

This runs the Docker container based on the chandan:v2 image, passing Pankaj Sharma as an argument to the echo command.

-- List running containers:

This lists all currently running Docker containers.

-- List all containers (including stopped ones):

This lists all Docker containers, including those that are stopped.

19-Dec-2024

Training Report

Project 3

-- Create a directory for Project 3:

Create a new directory named Project3 and navigate into it.

-- Create a Dockerfile:

Again, create an empty Dockerfile and open it in a text editor to add content.

-- Write the Dockerfile:

FROM ubuntu:latest: Use the latest Ubuntu as the base image.

ENTRYPOINT ["echo"]: Set echo as the command that will always be run when the container starts.

CMD ["welcome to Shimla"]: This provides default arguments to the echo command, so the container will print "welcome to Shimla" when run.

-- Build the Docker image:

This command builds a Docker image from the Dockerfile in the current directory (.), tagging it as :v3.

-- Run the Docker container:

This runs the Docker container based on the :v3 image. It will print "welcome to Shimla" since that is specified in the CMD instruction.

-- Run the container with arguments:

This runs the same Docker container but overrides the default message from CMD with "welcome to Haridwar". When using arguments, Docker replaces the CMD part but still

runs the ENTRYPOINT command.

20-Dec-2024

Training Report

DOCKER COMPOSE

What is Docker Compose?

Docker Compose is a tool that simplifies the management of multi-container Docker applications. It allows developers to define and manage multi-container environments using a single configuration file, typically called docker-compose.yml. With Docker Compose, you can define the services that make up your application, specify their configurations, and manage how they interact with each other.

In essence, Docker Compose enables you to:

-- Define multiple services in one place.

-- Configure each service's environment, networking, and dependencies.

-- Orchestrate starting, stopping, and scaling of services.

-- Run multi-container Docker applications with a single command.

Key Components of Docker Compose

Docker Compose revolves around a few key components:

-- docker-compose.yml file: This file defines the services, networks, and volumes required for a multi-container application. It specifies details such as the Docker images to use, the command to run, exposed ports, environment variables, volumes, and networks.

-- Services: Each service represents a single container within the application. For example, a web service, a database service, and a cache service are all defined as separate services within the Compose file.

-- Volumes: Volumes are used to persist data across container restarts. This is important

for services like databases, where you want to ensure data is retained even if the container stops or is rebuilt.

-- Networks: Networks allow services to communicate with each other. Docker Compose automatically creates a default network for services, but you can also define custom networks for more complex setups.

Why Use Docker Compose?

-- Simplified Multi-Container Application Management

Docker Compose makes it easier to define and manage multiple containers. Instead of manually starting each container with complex docker run commands, you can simply define everything in a docker-compose.yml file and start the entire system with a single docker-compose up command.

-- Declarative Infrastructure

With Docker Compose, the configuration for your application is stored in code (in the YAML file). This ensures consistency and allows teams to manage their infrastructure declaratively. It provides a clear record of how services should be configured and run.

-- Service Isolation and Networking

Docker Compose enables seamless service isolation. Each service runs in its own container, but Docker Compose automatically connects them via internal networking. Services can communicate with each other by referring to their service name (e.g., a web service can connect to a database by referring to the "db" service). This eliminates the need to manage IP addresses or manual port forwarding.

-- Scalability

Docker Compose allows you to scale services easily. For example, if you want to scale the number of web containers in a system, you can simply use docker-compose up --scale web=3. This command will launch three instances of the "web" service, distributing the workload.

-- Environment-Specific Configurations

Docker Compose supports different environments, such as development, testing, and production. Using features like .env files and service profiles, you can customize the configuration of your services depending on the environment. For example, in a development environment, you may want to expose debugging ports or run additional services that are not needed in production.

Training

26 -31 Dec 2024

Qubernetes started

What is Kubernetes?

Kubernetes (often abbreviated as **K8s**) is a platform designed to manage and orchestrate containers. Containers are lightweight, standalone, and executable units of software that include everything needed to run a piece of software, such as code, runtime, libraries, and dependencies.

Key Features of Kubernetes:

1. **Automated Deployment and Scaling**: Kubernetes ensures your applications are properly deployed and scales them up or down automatically based on demand.
2. **Self-Healing**: If a container crashes or fails, Kubernetes can restart or replace it automatically.
3. **Load Balancing**: Distributes traffic across containers to ensure stability and availability.
4. **Storage Orchestration**: Allows applications to use local storage, public cloud storage, or network storage seamlessly.
5. **Configuration Management**: Manages application configurations through secrets and config maps, ensuring sensitive data isn't exposed unnecessarily.
6. **Rolling Updates and Rollbacks**: Allows smooth updates to applications while maintaining service availability.

Use of qubernetes in aws:

1. Amazon EKS (Elastic Kubernetes Service)

Amazon EKS is a fully managed Kubernetes service provided by AWS. It simplifies running Kubernetes on AWS by taking care of the underlying infrastructure and Kubernetes control plane.

Key Features of Amazon EKS:

- **Managed Control Plane**: AWS handles Kubernetes cluster management, upgrades, and maintenance.
- **High Availability**: EKS automatically distributes the Kubernetes control plane across multiple availability zones.
- **Scalability**: Easily scale your applications and worker nodes up or down based on demand.
- **Integration with AWS Services**: Leverage services like IAM (Identity and Access Management), ALB (Application Load Balancer), and CloudWatch for seamless operation.
- **Support for Hybrid Deployments**: Use **Amazon EKS Anywhere** to run Kubernetes clusters on-premises.

---

2. Use Cases for Kubernetes in AWS

- **Microservices Deployment**: Kubernetes allows you to deploy and manage microservices architecture efficiently.
- **CI/CD Pipelines**: Automate application builds, tests, and deployments.
- **Hybrid and Multi-Cloud Strategies**: Run Kubernetes workloads across AWS and on-premises data centers.
- **AI/ML Workloads**: Kubernetes supports containerized AI/ML frameworks like TensorFlow or PyTorch, often combined with AWS GPU instances.

---

3. Benefits of Kubernetes in AWS

- **Ease of Management**: AWS takes care of operational tasks like control plane management and updates.
- **Scalability**: Kubernetes in AWS supports elastic scaling using **Amazon EC2 Auto Scaling** or **AWS Fargate**.
- **Cost Optimization**: Use Kubernetes with **Spot Instances** to run workloads cost-effectively.
- **Security**: AWS provides tools like **IAM roles for service accounts** to secure Kubernetes workloads.
- **Observability**: Integrate with AWS monitoring tools such as **Amazon CloudWatch**, **AWS X-Ray**, and **Prometheus** for visibility.

---

4. Running Kubernetes on AWS: Options

You can run Kubernetes in AWS through different methods:

1. **Amazon EKS**: Fully managed Kubernetes clusters.
2. **Self-Managed Kubernetes on EC2**: Set up and manage Kubernetes on AWS EC2 instances manually.
3. **AWS Fargate with EKS**: Run Kubernetes pods directly on AWS Fargate, removing the need to manage worker nodes.
4. **EKS Anywhere**: Manage hybrid or on-premises Kubernetes deployments.

Example Architecture: Kubernetes on AWS

- **Control Plane**: Managed by Amazon EKS in multiple Availability Zones for high availability.
- **Worker Nodes**: EC2 instances or AWS Fargate running in a Virtual Private Cloud (VPC).
- **Ingress Controller**: Integrated with Application Load Balancer (ALB) for external access.
- **Persistent Storage**: Use Amazon EBS or Amazon EFS for stateful workloads.
- **Networking**: Leverage VPC, AWS Security Groups, and Kubernetes Network Policies.

1. **Step 1: Launch an EC2 instance**
    - Use **AWS EC2** and launch the instance.
2. **Step 2: Log in to the instance**
    - Use SSH with the command:

php

Copy code

ssh -i <keyfile.pem> ubuntu@<Instance Public IP>

    - Example: Replace <keyfile.pem> and <Instance IP> with actual values.
3. **Step 3: Update packages**
    - Run:

sql

Copy code

sudo apt update -y

IAM User Creation

- **Steps to Create an IAM User:**
    1. **Create a user**:
        - Go to AWS IAM and create a new user.
    2. **Attach Policies Directly**:
        - Assign a policy such as **AdministratorAccess**.
    3. **Make Note of Access Key**:
        - Download the access key and secret.

AWS CLI Configuration

1. **Install AWS CLI**:
    - Run the command:

css

Copy code

snap install aws-cli --classic

2. **Configure AWS CLI**:
    - Use the command:

Copy code

aws configure

    - Provide:
        - **Region**: e.g., us-east-1
        - **Output**: e.g., json

Steps for Kubernetes Setup

1. **Generate SSH Key**:
    - Use the command:

bash

Copy code

ssh-keygen

- Save as .ssh/ or ~/.ssh.

2. **Install kubectl**:
    - Search for **kubectl installation on Linux** (use Google).
    - Copy and paste the **curl command** for downloading the binary.
    - Execute the command in the terminal.
3. **Verify Installation**:
    - Open GitBash and run:

bash

Copy code

kubectl version --client

Setting Up Domain Using Route 53

1. **Create Hosted Zones**:
    - Go to AWS Route 53 and create a new hosted zone for your domain.
    - Example:
        - Domain Name: kubernetes.example.com
2. **BigRock DNS Sync**:
    - Go to **BigRock** (or any DNS provider).
    - Sync your domain:
        - Click on **DNS Records**.
        - Update the **Name Servers (NS)** provided by Route 53 in your domain's settings.

Edit Configuration Files

1. **Example File**:
    - File name: tuna.
    - Example Configuration:

.ssh/id_ed25519.sub

2. **Zone Settings**:
    - Zone Name: kubernetes.example.net.
    - Ensure proper settings for hosted zones.

Setting Up AWS Bucket

1. **Create a Bucket**:
    - Bucket Name: Provide a unique bucket name.
2. **Enable Public Access**:
    - Unblock public access.
    - Acknowledge changes.

FINISHED FINISHED FINISHED