**Multi-Server Automation Documentation Using Python**

**Introduction**

Managing and configuring servers remotely and simultaneously is essential for top-tier clients, demanding efficiency, security, and speed. In this documentation, we detail a solution that leverages the Python language and the paramiko library to perform simultaneous tasks on different servers.

**Prerequisites**

* Python 3 installed.
* paramiko library installed:
* pip install paramiko
* SSH key-based authentication set up for each server.

**Solution Design**

A diagram of a software system

Description automatically generated

**Our solution has the following features**:

* SSH connections to each server (CentOS, Ubuntu, Alpine).
* Execution of specific commands tailored to the server's OS.
* Parallel execution, ensuring all servers are processed at the same time.
* Automated scheduling to initiate the process daily at 6 am.

**Import Dependencies:**

* import paramiko
* import time
* import logging

**paramiko**: A Python library facilitating SSH protocol operations. It provides an interface to interact remotely and execute commands on servers.

**time**: Used for time-based operations, notably for scheduling script activities.

**logging**: Provides logging capabilities, allowing the script to record its actions, outputs, and errors.

**Setting Up Logging**:

logging.basicConfig(filename='deployment\_log.log', level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

This configures the logging module to write logs to deployment\_log.log. It captures the timestamp, log level (e.g., INFO, ERROR), and the actual message.

**SSH Command Execution Function:**

def ssh\_execute(hostname, port, username, key\_filename, commands):

This is a generic function that establishes an SSH connection and executes a list of provided commands.

**Key Operations within the Function:**

**Client Setup and Connection:**

Set up an SSH client, decide on the policy for missing host keys, and then connect using the provided credentials.

**Command Execution:**

Iterates through the list of commands, executing them one by one. Outputs and errors are captured and logged.

**Operations on CentOS 7:**

def run\_on\_centos7():

**Key Steps:**

**System Update:**

Ensures the system is up-to-date with the latest patches and security updates.

**Installation of Required Packages**:

Installs the Apache web server (httpd), its SSL module, and certbot for Let's Encrypt SSL management.

**Server Management**:

Stops the Apache server to perform subsequent operations safely.

Starts the Apache server after all operations are complete.

**SSL Verification**:

Checks the validity of SSL certificates using a certbot dry-run.

**Operations on Ubuntu:**

def run\_on\_ubuntu():

**Key Steps**:

**System Update:**

Ensures the Ubuntu system has the latest updates.

**MariaDB Installation & Configuration:**

* Installs MariaDB server.
* Enables the MariaDB service to start on boot.
* Sets the root password for database security.

**Operations on Alpine:**

def run\_on\_alpine():

**Key Steps:**

**System Update:**

Updates all installed packages to their latest versions.

**Docker Installation & Test:**

* Installs Docker.
* Starts the Docker service.
* Tests the Docker installation by running a simple "Hello World" container.

**Script Execution Timing**:

if \_\_name\_\_ == "\_\_main\_\_":

The script uses a continuous loop to check the current time.

At 6 am, it will invoke the defined functions to perform tasks on CentOS 7, Ubuntu, and Alpine.

Post-execution, the script sleeps for an hour to prevent re-triggering within the 6 am hour.

**Security Measures:**

SSH Key-Based Authentication: The script uses SSH keys, a more secure method than password-based authentication. The private keys must be stored securely, with strict permissions (chmod 600).

**Error Handling & Logging:** All potential errors during command execution are captured, logged, and reported back to the administrator. This ensures transparency and allows swift troubleshooting.

**Conclusion:**

For our valued client, this script offers a robust and automated solution to manage multiple servers with diverse OS environments. With its logging, error handling, and secure mechanisms, the script delivers reliability and peace of mind. Always ensure to perform a backup and test any modifications in a controlled environment before deploying them in production.