Server Management Guide

Section 1: Account Management

Create accounts for students who need to use computing resources, allowing them to log into the computer and utilize those resources.

Adding a User

To add a new user, use the following command:

```
useradd -m <username>
```

Setting a User Password

Set a password for the new user:

```
passwd <username>
```

Caution: Granting sudo Privileges

Granting sudo privileges allows a user to execute commands with administrative rights. Use this command carefully:

```
usermod -aG sudo <username>
```

Setting Up SSH Key Authentication

For enhanced security, set up SSH key-based authentication and consider disabling password authentication.

1. **Generate an SSH Key on the Client Side**

   On the user's client machine, generate an SSH key pair:

   ```
   ssh-keygen
   ```

2. **Upload the Public Key to the Server**

   Upload the public key to the server to enable key-based login:

   ```
   ssh-copy-id -i <public_key_file> <username>@<server_ip_address>
   ```

   Alternatively, manually add the public key to the server's authorized keys file:

```
echo <public_key_contents> >> /home/<username>/.ssh/
    authorized_keys
```

Section 2: Environment Setup

Set up Python environment management for users using **conda**.

Conda is a package manager focused on Python environments. It allows you to create and manage multiple isolated (virtual) Python environments. Packages are downloaded from repositories known as “channels” or “sources,” which can be open-source (e.g., conda-forge) or proprietary (e.g., Anaconda).

Installing Miniforge

1. **Download the Miniforge Installation Script**

   ```
   wget https://github.com/conda-forge/miniforge/releases/latest/
       download/Miniforge3-Linux-x86_64.sh
   ```

2. **Run the Installation Script**

   ```
   bash Miniforge3-Linux-x86_64.sh
   ```

3. **Prevent Auto-Activation of the Base Environment**

   To avoid conflicts with system software that relies on the system’s Python, configure conda not to automatically activate the base environment:

   ```
   conda config --set auto_activate_base false
   ```

Creating and Activating a New Environment

1. **Create a New Environment**

   ```
   conda create -n py310 python=3.10
   ```

2. **Activate the New Environment**

   ```
   conda activate py310
   ```

Installing PyTorch

Install PyTorch following the official instructions, ensuring compatibility with your CUDA version:

- Visit the PyTorch website and follow the appropriate installation guide.(Use `pip`)

Section 3: Package Management

This section is intended for administrators to install and manage system packages.

Ubuntu is a Debian-based distribution that uses the `.deb` package format. It utilizes package managers like **dpkg** (backend) and **apt** (frontend) to install, remove, and query packages.

Common apt Commands

- **Update Package Index**

```
sudo apt update
```

- **Upgrade All Installed Packages**

```
sudo apt upgrade
```

- **Install a Specific Package**

```
sudo apt install <package_name>
```

- **Install a Local .deb File**

```
sudo apt install ./path/to/package.deb
```

- **Remove an Installed Package (Keep Configuration Files)**

```
sudo apt remove <package_name>
```

- **Remove an Installed Package and Its Configuration Files**

```
sudo apt purge <package_name>
```

- **Automatically Remove Unneeded Packages**

```
sudo apt autoremove
```

- **Search for a Package**

  ```
  apt search <package_name>
  ```

Resolving Dependency Issues

System packages often have complex dependency relationships. Dependency issues may arise during installation, removal, or updates if dependencies are not satisfied.

Steps to Resolve Dependency Problems:

1. **Update the Package Index**

   Ensure your package index is up to date:

   ```
   sudo apt update
   ```

2. **Use `aptitude` to Automatically Resolve Dependencies**

   Install `aptitude` if it's not already installed:

   ```
   sudo apt install aptitude
   ```

   Use `aptitude` to attempt automatic resolution:

   ```
   sudo aptitude install <package_name>
   ```

3. **Manually Resolve Conflicts Based on Error Messages**

   - **Read Error Messages Carefully**

     Identify which packages are conflicting and understand why.

   - **Consider Possible Causes**

     - Circular dependencies
     - Outdated package index compared to the repository
     - Manually installed third-party packages

   - **Determine Required Package Versions**

     Find out which package versions are needed and how to obtain them (e.g., by updating the index to download from repositories or searching on pkgs.org).

Reference Materials

- **Debian Packaging Tutorial**
  - Debian Packaging Tutorial (PDF)
- **Debian Package Management Tools**
  - Debian FAQ: Package Tools
  - Debian FAQ: Basics of the Debian Package Management System

---

Section 4: Caution: Reverse Proxy

This section is intended for users who need to access the server from outside the institution.

In certain environments, your server resides within a local area network (LAN) protected by a firewall that controls inbound and outbound traffic. External access to internal machines is blocked, but internal machines can access the public internet. To access the server from outside the LAN, network tunneling is required.

Overview

Use a public virtual private server (VPS) to facilitate the connection.

- **Compute Server**: The internal server you wish to access.
- **VPS**: A server with a public IP address.
- **Client**: The machine from which you wish to access the Compute Server.

Connection Flow

1. **Establish Connection a Between Compute Server and VPS**
   - The Compute Server establishes a persistent connection to the VPS on a specified port.
2. **VPS Listens on Port `8080`**
   - The VPS forwards incoming traffic on port `8080` through connection a to the Compute Server.
3. **Compute Server Receives Traffic on Port 22 (SSH Port)**

- The client connects to the VPS on port 8080, which is forwarded to the Compute Server's port 22.

Diagram:

```
(Client)(local_port) <-------> (8090)(VPS)(8080) <---[Connection a]--->
    (local_port)(Compute Server)(22)
```

Using frp for Network Tunneling

We use **frp** (Fast Reverse Proxy) to set up the tunneling.

Deploying frps on the VPS Server Create a configuration file `frps.ini` with the following content:

```
[common]
bind_addr = 0.0.0.0
bind_port = 8080
authentication_method = token
authenticate_heartbeats = true
authenticate_new_work_conns = true
token = "YOUR_SECURE_TOKEN"
log_file = "./frps.log"
```

Start frps:

```
frps -c frps.ini
```

Deploying frpc on the Compute Server Create a configuration file `frpc.ini` with the following content:

```
[common]
server_addr = "<VPS_PUBLIC_IP>"
server_port = 8090
authentication_method = token
token = "YOUR_SECURE_TOKEN"

[ssh]
type = tcp
local_ip = 127.0.0.1
local_port = 22
remote_port = 6000
```

Start frpc:

```
frpc -c frpc.ini
```

Connecting from the Client On your local machine, connect to the Compute Server via the VPS:

```
ssh -p 6000 <username>@<VPS_PUBLIC_IP>
```

Caution: Security Considerations

- **Set a Strong Authentication Token**

  Ensure `token` is a strong, unique value to secure the frp service.(For example,`940142B3-6653-4000-ACB7-F203CA3E7456`)

- **Regularly Update frp**

  Keep frp up to date to benefit from security patches and improvements.

- **Disable Password Authentication**

  As mentioned in Section 1, disable password-based SSH login and enforce key-based authentication for enhanced security.

  In the `/etc/ssh/sshd_config` file on the Compute Server, set:

  ```
  PasswordAuthentication no
  ```

  Restart the SSH service:

  ```
  sudo systemctl restart ssh
  ```

Note: Always exercise caution when exposing internal services to external networks. Ensure all configurations comply with your organization's security policies.
