サーバー管理ガイド

第 1 章: アカウント管理

計算資源を必要とする学生にアカウントを作成し、サーバーへのログインとリソースの利用を許可します。

ユーザーの追加

新しいユーザーを追加するには、以下のコマンドを使用します。

```
useradd -m <ユーザー名>
```

ユーザーパスワードの設定

新しいユーザーのパスワードを設定します。

```
passwd <ユーザー名>
```

注意: sudo 権限の付与

sudo 権限を付与すると、ユーザーは管理者権限でコマンドを実行できます。慎重に以下のコマンドを使用してください。

```
usermod -aG sudo <ユーザー名>
```

SSH　認証の設定

セキュリティを強化するため、SSH の鍵認証を設定し、パスワード認証を無効にすることを検討してください。

1. **クライアント側で SSH　を生成**

   ユーザーのクライアントマシンで、SSH 鍵ペアを生成します。

   ```
   ssh-keygen
   ```

2. **サーバーに公開　をアップロード**

   公開鍵をサーバーにアップロードし、鍵認証によるログインを有効にします。

   ```
   ssh-copy-id -i <公開鍵ファイル> <ユーザー名>@<サーバーのIPアドレス
   >
   ```

   または、サーバーの authorized_keys ファイルに公開鍵を手動で追加します。

```
echo <公開鍵の内容> >> /home/<ユーザー名>/.ssh/authorized_keys
```

第 2 章: 環境構築

ユーザーの Python 環境管理に **conda** を使用します。

conda は Python 環境に特化したパッケージマネージャーで、複数の独立した（仮想）Python 環境を作成・管理できます。パッケージは「チャネル」または「ソース」と呼ばれるリポジトリからダウンロードされ、オープンソース（例: conda-forge）やプロプライエタリ（例: Anaconda）があります。

Miniforge のインストール

1. **Miniforge インストールスクリプトのダウンロード**

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

2. **インストールスクリプトの実行**

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

3. **ベース環境の自動アクティベーションを無効化**

   システムの Python に依存するソフトウェアとの競合を避けるため、conda がベース環境を自動的にアクティブにしないよう設定します。

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

新しい環境の作成とアクティベーション

1. **新しい環境の作成**

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

2. **新しい環境のアクティブ化**

```
conda activate py310
```

PyTorch のインストール

CUDA バージョンとの互換性を確認しながら、公式の手順に従って PyTorch をインストールします。(pip を使用してください。)

- PyTorch 公式サイトにアクセスし、適切なインストールガイドに従ってください。

第 3 章: パッケージ管理

この章は、システムパッケージをインストールおよび管理する管理者向けです。

Ubuntu は Debian ベースのディストリビューションで、`.deb`パッケージ形式を使用します。パッケージのインストール、削除、クエリには、**dpkg**（バックエンド）と **apt**（フロントエンド）のパッケージマネージャーを使用します。

apt の一般的なコマンド

- **パッケージインデックスの更新**

```
sudo apt update
```

- **インストール済みパッケージのアップグレード**

```
sudo apt upgrade
```

- **特定のパッケージのインストール**

```
sudo apt install <パッケージ名>
```

- **ローカルの.deb ファイルのインストール**

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

- **インストール済みパッケージの削除（設定ファイルを保持）**

```
sudo apt remove <パッケージ名>
```

- **インストール済みパッケージとその設定ファイルの削除**

```
sudo apt purge <パッケージ名>
```

- **不要なパッケージの自動削除**

```
sudo apt autoremove
```

- **パッケージの検索**

```
apt search <パッケージ名>
```

依存関係の問題解決

システムパッケージは複雑な依存関係を持つことが多く、依存関係が満たされない場合、インストールや削除、アップデート中に問題が発生することがあります。

依存関係問題を解決する手順

1. **パッケージインデックスの更新**

   パッケージインデックスを最新の状態に保ちます。

   ```
   sudo apt update
   ```

2. **`aptitude`を使用して自動的に依存関係を解決**

   `aptitude`がインストールされていない場合は、以下でインストールします。

   ```
   sudo apt install aptitude
   ```

   `aptitude`で自動解決を試みます。

   ```
   sudo aptitude install <パッケージ名>
   ```

3. **エラーメッセージに基づいて手動で競合を解決**

   - **エラーメッセージを注意深く読む**

     どのパッケージが競合しているかを特定し、原因を理解します。

   - **考えられる原因を検討**

     - 循環依存
     - リポジトリに比べて古いパッケージインデックス
     - 手動でインストールしたサードパーティ製パッケージ

   - **必要なパッケージバージョンを特定**

     必要なバージョンを確認し、取得方法を決定します（例: インデックスを更新してリポジトリからダウンロード、循環依存のパッケージを一旦削除してリポジトリからダウンロード、またはpkgs.orgで検索）。

参考資料

- **Debian パッケージングチュートリアル**
  - Debian パッケージングチュートリアル（PDF）
- **Debian パッケージ管理ツール**
  - Debian FAQ: パッケージツール
  - Debian FAQ: パッケージ管理システムの基本

第 4 章: 注意: リバースプロキシの設定

この章は、学外からサーバーにアクセスする必要があるユーザー向けです。

特定の環境では、サーバーがローカルエリアネットワーク（LAN）内にあり、ファイアウォールによって保護されています。外部から内部マシンへのアクセスはブロックされていますが、内部マシンはインターネットにアクセスできます。LAN の外部からサーバーにアクセスするには、ネットワークトンネルを使用する必要があります。

概要

パブリック IP アドレスを持つ仮想プライベートサーバー（VPS）を利用して接続を確立します。

- **計算サーバー**: アクセスしたい内部サーバー
- **VPS**: 公開 IP アドレスを持つサーバー
- **クライアント**: 計算サーバーにアクセスするマシン

接続フロー

1. **計算サーバーと VPS 間で接続aを確立**
   - 計算サーバーが VPS の指定ポートに持続的な接続を確立します。
2. **VPS がポート`8080`でリスニング**
   - VPS はポート`8080`で受信したトラフィックを接続aを通じて計算サーバーに転送します。
3. **計算サーバーがポート`22`（SSH ポート）でトラフィックを受信**
   - クライアントは VPS のポート`8080`に接続し、それが計算サーバーのポート`22`に転送されます。

図:

```
(クライアント)(local_port) <------> (8090)(VPS)(8080) <---[接続a]---> (local_port)(計算サーバー)(22)
```

frp を使用したネットワークトンネルの構築

ネットワークトンネルの設定には、**frp**（Fast Reverse Proxy）を使用します。

VPS サーバー上での frps のデプロイ　`frps.ini`という設定ファイルを作成し、以下の内容を記述します。

```
[common]
bind_addr = 0.0.0.0
bind_port = 8080
authentication_method = token
authenticate_heartbeats = true
authenticate_new_work_conns = true
token = "あなたの安全なトークン"
log_file = "./frps.log"
```

frps の起動:

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

計算サーバー上での frpc のデプロイ　`frpc.ini`という設定ファイルを作成し、以下の内容を記述します。

```
[common]
server_addr = "<VPSのパブリックIPアドレス>"
server_port = 8080
authentication_method = token
token = "あなたの安全なトークン"

[ssh]
type = tcp
local_ip = 127.0.0.1
local_port = 22
remote_port = 8090
```

frpc の起動:

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

クライアントからの接続　ローカルマシンから、VPS 経由で計算サーバーに接続します。

```
ssh -p 8090 <ユーザー名>@<VPSのパブリックIPアドレス>
```

注意: セキュリティ対策

- **強力な認証トークンを設定**

  tokenには強力でユニークな値を設定し、frp サービスのセキュリティを確保します。例:940142B3-6653-4000-ACB7-F203CA3E7456

- **frp を定期的に更新**

  セキュリティパッチや改善を受けるため、frp を最新の状態に保ちます。

- **パスワード認証を無効化**

  第 1 章で述べたように、SSH のパスワード認証を無効化し、鍵認証を強制します。

  計算サーバーの/etc/ssh/sshd_configファイルで以下を設定します。

  ```
  PasswordAuthentication no
  ```

  SSH サービスを再起動します。

  ```
  sudo systemctl restart ssh
  ```

注記: 内部サービスを外部ネットワークに公開する際は常に注意が必要です。すべての設定が組織のセキュリティポリシーに準拠していることを確認してください。
