SSH クイックリファレンス

接続、鍵、設定、トンネル、SCP、SFTP

接続

基本的な接続

```
ssh user@host                   # ホストに接続
ssh -p 2222 user@host           # カスタムポート
ssh user@host command             # リモートコマンドを実行
ssh -t user@host "top"          # TTY 割り当てを強制
```

接続フラグ

`-p port`	指定ポートに接続
`-i key`	特定の秘密鍵を使用
`-t`	擬似端末の割り当てを強制
`-v / -vv / -vvv`	詳細デバッグ出力（レベルが増すほど詳細）
`-q`	クワイエットモード（警告を抑制）
`-N`	リモートコマンドなし（トンネル用）
`-f`	コマンド前にバックグラウンドに移行
`-J jump`	ジャンプホスト（ProxyJump）

鍵の管理

鍵の生成

```
ssh-keygen -t ed25519 -C "you@example.com"
ssh-keygen -t rsa -b 4096 -C "you@example.com"
ssh-keygen -t ed25519 -f ~/.ssh/mykey
ssh-keygen -p -f ~/.ssh/id_ed25519  # パスフレーズを変更
```

公開鍵の配置

```
ssh-copy-id user@host
ssh-copy-id -i ~/.ssh/mykey.pub user@host
# 手動: リモートの authorized_keys に .pub を追記
cat ~/.ssh/id_ed25519.pub | ssh user@host \
  "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys"
```

鍵ファイル

`~/.ssh/id_ed25519`	秘密鍵（秘匿にする）
`~/.ssh/id_ed25519.pub`	公開鍵（共有可能）
`~/.ssh/authorized_keys`	リモート: 許可された公開鍵
`~/.ssh/known_hosts`	既知のホストフィンガープリント

設定ファイル

~/.ssh/config の基本

```
Host myserver
  HostName 192.168.1.100
  User deploy
  Port 2222
  IdentityFile ~/.ssh/deploy_key

# 以下で接続できるようになる:
# ssh myserver
```

便利な設定オプション

```
Host *
  ServerAliveInterval 60
  ServerAliveCountMax 3
  AddKeysToAgent yes
  IdentitiesOnly yes

Host bastion
  HostName bastion.example.com
  User admin
```

設定ディレクティブ

`Host`	エントリのエイリアスパターン
`HostName`	実際のホスト名または IP
`User`	ログインユーザー名
`Port`	リモートポート（デフォルト 22）
`IdentityFile`	秘密鍵へのパス
`ProxyJump`	別のホストを経由して接続
`ServerAliveInterval`	キープアライブ間隔（秒）
`IdentitiesOnly`	指定した鍵のみを使用

ポートフォワーディング

ローカルフォワーディング（-L）

```
# リモートポート 5432 をローカルポート 5432 でアクセス
ssh -L 5432:localhost:5432 user@host
# ssh ホスト経由で remote-db:3306 にアクセス
ssh -L 3306:remote-db:3306 user@host
# すべてのインターフェースにバインド
ssh -L 0.0.0.0:8080:localhost:80 user@host
```

リモートフォワーディング（-R）

```
# ローカルポート 3000 をリモートポート 8080 で公開
ssh -R 8080:localhost:3000 user@host
# リモートの任意のインターフェースから接続を許可
ssh -R 0.0.0.0:8080:localhost:3000 user@host
```

ダイナミックフォワーディング（-D）

```
# ローカルポート 1080 に SOCKS5 プロキシを設定
ssh -D 1080 user@host
# バックグラウンド SOCKS プロキシ
ssh -D 1080 -fN user@host
```

SCP と SFTP

SCP（セキュアコピー）

```
scp file.txt user@host:/remote/path/
scp user@host:/remote/file.txt ./local/
scp -r dir/ user@host:/remote/path/
scp -P 2222 file.txt user@host:/path/
```

SFTP（インタラクティブ転送）

```
sftp user@host
# sftp セッション内:
# put local.txt    – ファイルをアップロード
# get remote.txt   – ファイルをダウンロード
# ls / lcd / cd    – ディレクトリの一覧表示・移動
```

転送フラグ

`-r`	再帰的（ディレクトリをコピー）
`-P port`	ポートを指定（SCP は -P を使用、-p ではない）
`-C`	圧縮を有効化
`-l limit`	帯域幅を Kbit/s で制限
`-i key`	特定の秘密鍵ファイルを使用

エージェント転送

SSH エージェント

```
eval "$(ssh-agent -s)"          # エージェントを起動
ssh-add ~/.ssh/id_ed25519       # エージェントに鍵を追加
ssh-add -l                      # 読み込まれた鍵を一覧表示
ssh-add -D                      # すべての鍵を削除
```

エージェントの転送

```
ssh -A user@host                # エージェントを転送
# または ~/.ssh/config で:
# Host myserver
#   ForwardAgent yes
```

エージェントに関する注意

エージェント転送により、リモートホストはローカルの鍵をコピーせずに使用できる。信頼できるホストのみで使用すること。可能な場合はエージェント転送より ProxyJump を優先する。

トンネル

永続的なトンネル

```
# バックグラウンドで維持されるトンネル
ssh -fNT -L 5432:localhost:5432 user@host
# 自動再接続トンネル（autossh を使用）
autossh -M 0 -fNT -L 5432:localhost:5432 user@host
```

ジャンプホスト・踏み台サーバー

```
ssh -J bastion user@internal-host
ssh -J user1@hop1,user2@hop2 user@target
# 設定での同等記述:
# Host internal
#   HostName 10.0.0.5
#   ProxyJump bastion
```

トンネル管理

`-f` 認証後にバックグラウンドに移行
`-N` リモートコマンドなし
`-T` 擬似端末を無効化
`~.` スタックした SSH セッションを終了（エスケープ）
`~C` フォワーディング用コマンドラインを開く
`~#` 転送中の接続を一覧表示

トラブルシューティング

接続のデバッグ

```
ssh -vvv user@host              # 最大詳細度
ssh -G user@host                # 設定をダンプ（ドライラン）
ssh-keyscan host                # ホスト鍵を取得
ssh-keygen -R host              # known_hosts から削除
```

よくある問題

`Permission denied`	鍵、ユーザー、または ~/.ssh のパーミッション（700/600）が間違っている
`Host key changed`	ssh-keygen -R host を実行して再接続
`Connection timed out`	ファイアウォール、ポート、ホストの到達可能性を確認
`Too many auth failures`	-i で鍵を指定するか IdentitiesOnly を使用
`Broken pipe`	設定に ServerAliveInterval を追加

ファイルパーミッション

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/id_ed25519     # 秘密鍵
chmod 644 ~/.ssh/id_ed25519.pub # 公開鍵
chmod 600 ~/.ssh/authorized_keys
chmod 644 ~/.ssh/known_hosts
```

セキュリティのベストプラクティス

サーバーの強化

`PasswordAuthentication no`	パスワードログインを無効化
`PermitRootLogin no`	root の SSH アクセスを無効化
`AllowUsers deploy`	許可するユーザーをホワイトリストに登録
`Port 2222`	デフォルト以外のポート（スキャナを回避）
`MaxAuthTries 3`	認証試行回数を制限

鍵の運用

Ed25519 鍵を優先すること（小さく、速く、より安全）。
秘密鍵には常にパスフレーズを設定すること。
ssh-agent を使用してパスフレーズの再入力を避ける。
定期的に鍵をローテーションし、未使用の鍵は失効させる。
IdentitiesOnly を使用して提示する鍵を制御する。

多重化

接続の共有

```
# ~/.ssh/config 内
Host *
  ControlMaster auto
  ControlPath ~/.ssh/sockets/%r@%h-%p
  ControlPersist 600

# ソケットディレクトリを作成
mkdir -p ~/.ssh/sockets
```

多重化のメリット

同一ホストへの複数の SSH セッションで 1 つの TCP 接続を再利用。
繰り返しのハンドシェイクを排除し、接続が高速になりオーバーヘッドが低減。
ControlPersist でマスターを維持する時間（秒）を設定。

エスケープシーケンス

SSH エスケープコマンド

`~.` 接続を終了（スタックしたセッションを削除）
`~^Z` SSH セッションをサスペンド
`~C` コマンドラインを開く（フォワーディングを追加）
`~#` 転送中の接続を一覧表示
`~&` SSH をバックグラウンドに移行（接続待機中）
`~?` エスケープヘルプを表示
`~~` チルダをそのまま送信