指定労働日レポート

シェルスクリプトから

SSHログインして別端末で作業させること

2019/08/17

社員番号：719755

Tuvdendorj　Gantulga

◆目的◆

私たち毎日顧客さんのサーバと社内サーバ、他の機器にレモートでログインする必要があります。リモートサーバに入るときTeratermプログラム使ってsshとTelnetで接続して色々な作業しています。ここで毎日ログインする時サーバに入るたびにユーザー名とパスワードを入力するのは辛いからシエルスクリプトを作成して保存して置いたらまた入るときクリックだけで入れるのは早くて便利です。

◆作成 ◆

・SSHログイン先では実行スクリプトと停止スクリプトを対にしておくこと

・trapによる処理を書くこと

・watch\_dogのスクリプトを用意してタイムアウトさせること

パスフレーズ無しの鍵を作ってリモートへログインできるようにする。まずはログイン元の端末で下記のコマンドを実行して秘密鍵・公開鍵のペアを作成

◆SSH設定◆

まずは基本として，SSHでコマンドを送り込むためのメモ。今回の環境はUbuntu OSでやってます。

CentOSの場合はパーミッションで蹴られることが多いので注意ください。

$ ssh-keygen -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (...): [ファイル名を指定する]

Enter passphrase (empty for no passphrase): [何も入れずにエンター]

Enter same passphrase again: [何も入れずにエンター]

Your identification has been saved in [ファイル名].

your public key has been saved in [ファイル名].pub.

The key fingerprint is:

[フィンガープリントが表示される]

ファイル名にyumeを指定した場合，秘密鍵のyumeと公開鍵のyume.pubが生成される。yumeは厳重に管理、yume.pubは適当に扱います。

デフォルトはid\_rsaになっている。デフォルトで使われる上に他の端末で作った鍵と混じったりしたときに訳が分からなくなる。鍵ファイル名はほとんど使わないので長い名前にしても大丈夫です。

続いてconfigファイルの編集、.ssh/configファイルに下記を追記する。

Host [remote\_name]

User [user\_name]

Port [22]

Hostname [hostname]

IdentityFile [path/to/keyfile]

IdentitiesOnly yes

[ ProxyCommand ssh ssh\_server\_name -W %h:%p ]

[remote\_name]は適当に使いやすい名前を指定。keyfileには秘密鍵を指定。.ssh/keyとかの下に置いてディレクトリごと700にしておく必要かあります。最後のProxyCommandは踏み台サーバが必要なときの設定。直接SSHログインできるなら何も書かなくて良いです。

◆ログイン先の端末での作業◆

$ cat yume.pub >> .ssh/authorized\_keys

コマンドを実行

基本的には公開鍵の中身をauthorized\_keysに追記したほうがよいですリダイレクトを使ってるが，別にコピペでもいいし見ながら頑張ってキーボードを叩いても良い。

とにかく公開鍵の中身のテキストがauthorized\_keysに追記されていれば良いです。

**＊ログイン元の端末から**

$ ssh [remote\_name]

パスワードを入れずにログインできれば成功。ログイン出来なかったりパスワードを求められたりした場合は何かが間違えてる。

成功したら，ログイン先では/etc/ssh/sshd\_configを編集してパスワード認証を切っておく。

**成功したかどうか確認せずにパスワード認証を切ってしまうとログイン出来なくなる場合がありますので注意ください。**

/etc/ssh/sshd\_config

# Change to no to disable tunnelled clear text passwords

PasswordAuthentication no

後はsudo service sshd restartとかでSSHデーモンをリスタートしたらSSH設定が完了です。

◆SSHを使ったコマンド投入◆

sshはコマンドを後ろにくっつければそれを実行させます。

$ ssh remote\_name ls

Document

Download

Picture

この ls コマンドはファイルの一覧見せるコマンドです。上記のコマンドを実行したら上のフォルダが見せてあげます。また他の色々このようなコマンドを実行をできます。

◆ シェルスクリプトからSSHでコマンド投入する◆

下記のようなスクリプトを作る。control.shがremoteマシンのremote.shを呼び出す。

|  |
| --- |
| control.sh  #!/bin/sh  echo "Start"  ssh remote ~/remote.sh  echo "Done" |

|  |
| --- |
| remote.sh  #!/bin/sh  echo "Remote Start"  sleep 10  echo "Remote Done" |

レモートでは単に10秒待つだけのプログラム。まずこの時点でcontrol.shに実は誤りがあります。

シェルスクリプト中で指定した、ローカルのホームディレクトリを指し示すため，ユーザ名が異なる端末にログインする場合はフルパスで指定する必要があります。全部同じユーザにしておくとその辺を気にしなくて良いですが，例えばrootログインさせる場合などは注意してほうがいいです。

実行すると下記のようになる。実行権限を付けるのを忘れないように。

|  |
| --- |
| $ ./control.sh  Start  Remote Start  [ 10秒待機 ]  Remote Done  Done  $ |

◆control-Cで止めてみる◆

実行したが何か様子がおかしいので止めよう，というときにほぼ大半の人はcontrol-Cを押すと停止します。以下のプログラムで止めて見ます。

|  |
| --- |
| $ ./control.sh  Start  Remote Start  [ control-Cを押下 ]  ^C　　　　　　　　　　　　　　　　　　⇒ctrl+c  $ |

一見止まったように見える。しかし，remote側でps axをしてみたら

|  |
| --- |
| ...  23069 ? Ss 0:00 /bin/sh /home/user/remote.sh  ... |

といった形でしっかり動いている。

control-Cの押下で行われるのはkillコマンドでSIGINT(2)を送ったことと同様になるのだが，**SSHログイン先で実行中のコマンドに対して送られるわけではなく，手元で動いているシェルスクリプトに対して送られる**。SSHで作業中にcontrol-Cを押してもSSHセッションが切れるわけではないのｓで誤解しやすいと思います。

◆停止用スクリプトの作成◆

SSHログイン先のスクリプトを作成した場合，必ず対となる停止用スクリプトを作成したほうがいいと思います。

例えばremote.shを止めるには下記のようなスクリプトを用意する。

|  |
| --- |
| stop。sh  #!/bin/sh  ps ax | grep [r]emote.sh | awk '{print $1}' | xargs kill |

一番正しいお作法はremote.shが起動直後に自身のPIDを$から取得してそれを呼び出し元に教えたりしてkillする方法だがあります。

control.shは下記のようにしてシグナルを取得，スクリプトを停止する。

|  |
| --- |
| control.sh  #!/bin/sh  echo "Start"  trap "ssh remote ~/stop.sh" 2 15  ssh remote ~/remote.sh  echo "Done" |

trapコマンドはシグナルを受けたときのコマンド，シグナルの種類を指定する。2はSIGINTで，ついでにkillコマンドが実行されたときのSIGTERM(15)も追加しておく。

これでcontrol-Cによる停止やkillコマンドによる停止をしたときにSSH先のコマンドがちゃんと止まるようになります。

◆タイムアウトさせる◆

スクリプトが完成して3日ほど色々な項目を勝手に試させる，みたいなことをするとよくありますが途中で勝手に止まって残りの項目を消化できていない場合に備えて下記のようなwatch\_dog機能を用意しておきます。

|  |
| --- |
| watch\_dog.sh  #!/bin/sh  if [ -z "$2" ]  then  echo "Usage: $0 time pid"  exit 1  fi  sleep $1 &  SPID=$!  trap "kill ${SPID} && exit" 15  wait  trap 15  kill -10 $2 |

。/watch\_dog.sh 10 12345といった形で実行すると，10秒後にPID=12345のプロセスに対してシグナル10を送る。止める場合は普通にkillするだけでいいです。

このwatch\_dogスクリプトをcontrol。shで使うには以下のようにする。

|  |
| --- |
| control.sh  #!/bin/sh  echo "Start"  ./watch\_dog.sh 3 $$ &  WD\_PID=$!  trap "ssh remote ~/stop。sh && exit 1" 10  trap "kill ${WD\_PID} && ssh remote ~/stop.sh" 2 15  ssh remote ~/remote.sh &  wait $! && kill ${WD\_PID}  echo "Done" |

sshのコマンド発行する時バックグラウンドに回してwaitで待機しているが，このようにしないと正常に動作しない。また，正常終了時やcontrol-Cによる処理のときはちゃんとwatch\_dogを停止する必要があります。

◆懸案事項◆

レモート先でSSHして他の端末に指示するような場合はそれぞれで停止スクリプトなってタイムアウトなどが発火させる必要があります、とにかく大事なのは動かすスクリプトと止めるスクリプトを対で用意して，異常時は止めるスクリプトを確実に動作させること大事と思います。例えば測定や試験の自動化を進める時これをちゃんと書いておかないと異常時に爆発的にスレッドが増えてサーバがハングアップできないとか、何かあったら意地でも止めるという気持ちが大事だと思います。