OpenSUSE Serwer SSH

OpenSSH – zestaw programów komputerowych zapewniających szyfrowaną komunikację w sieci komputerowej dzięki protokołowi SSH

Instalacja i konfiguracja serwera OpenSSH w systemie OpenSUSE 15.2 wersja serwer (bez GUI)

Instalacja

```
sudo zypper update
sudo zypper install openssh
```

Pakiet OpenSSH instaluje wiele komend w systemie, poniższa tabela przedstawia niektóre z nich.

Komenda	***Opis***
`scp`	Bezpieczna alternatywa dla rcp.
`sftp`	Bezpieczna alternatywa dla ftp.
`slogin`	Bezpieczna alternatywa dla rlogin, alias dla klienta ssh.
`ssh`	Bezpieczna alternatywa dla rlogin i telnet.
`ssh-add`	Dodaje charakterystykę RSA/DSA/ECDSA do ssh-agent.
`ssh-agent`	Agent uwierzytelniania. Zarządza kluczami używanymi przez RSA/DSA/ECDSA do uwierzytelniania.
`ssh-copy-id`	Kopiuje klucze RSA/DSA/ECDSA na zdalne systemy.
`ssh-keygen`	Generuje prywatne i publiczne klucze.

Status / uruchamianie / zatrzymanie/ restart

```
sudo systemctl status sshd
sudo systemctl start sshd
sudo systemctl stop sshd
sudo systemctl restart sshd
```

Konfiguracja

Plik konfiguracyjny znajduje się w katalogu `/etc/ssh/` i nosi nazwę sshd_config`

Opcje konfiguracji

`Port`	**Port, na którym nasłuchuje sshd, domyślnie 22.**
`ListenAddress`	Adres IP, na którym ma nasłuchiwać demon.
`Protocol`	Wersja protokołu, która ma być używana. Domyślnie 2.
`SyslogFacility`	Domyślnie `AUTHPRIV`. Definiuje kod udogodnienia, jakie ma być używane podczas logowania komunikatów do `/var/log/secure`. Oparte o konfigurację w pliku `/etc/rsyslog.conf`.
`LogLevel`	Domyślnie INFO. Poziom krytyczności logowanych komunikatów.
`LoginGraceTime`	Domyślnie 2 minuty. Czas, po którym rozłączane jest połączenie z użytkownikiem, który się nie zalogował.
`PermitRootLogin`	Domyślnie tak (yes). Pozwala na bezpośrednie logowanie roota do systemu.
`MaxAuthTries`	Domyślnie 6. Maksymalna liczba dozwolonych prób uwierzytelnienia użytkownika. Po tej ilości prób zrywane jest połączenie.
`MaxSessions`	Domyślnie 10. Maksymalna ilość otwartych sesji SSH równolegle.
`RSAAuthentication`	Domyślnie yes. Określa czy pozwalać na uwierzytelnienie RSA.
`PubKeyAuthentication`	Domyślnie yes. Określa czy ma być włączona możliwość uwierzytelnienia przez klucz publiczny.
`AuthorizedKeysFile`	Domyślnie `~/.ssh/authorized_keys`. Określa położenie pliku z autoryzowanymi kluczami użytkowników.
`PasswordAuthentication`	Domyślnie yes. Określa czy ma być włączona możliwość uwierzytelnienia w oparciu o hasło.
`PermitEmptyPasswords`	Domyślnie no. Określa czy zezwalać na puste hasła.
`ChallengeResponseAuthentication`	Domyślnie yes. Określa czy pozwalać na uwierzytelnianie metodą challenge-response.
`UsePAM`	Domyślnie yes. Włącza lub wyłącza uwierzytelnianie przez PAM. Jeżeli włączone tylko root może uruchomić demona.
`AllowAgentForwarding`	Domyślnie yes. Włącza lub wyłącza komendę ssh-agent służącą do forwardowania kluczy prywatnych do zdalnych systemów.
`AllowTCPForwarding`	Domyślnie yes. Określa czy pozwalać na forwardowanie komunikacji TCP przez kanał ssh.

`X11Forwarding`	Domyślnie yes. Pozwala lub zakazuje na zdalny dostęp do aplikacji graficznych (z GUI).
`TCPKeepAlive`	Domyślnie yes. Określa czy wysyłać sygnały TCP keepalive do serwera ssh aby sprawdzić jego dostępność.
`UseLogin`	Domyślnie no. Pozwala lub zakazuje na użycie komendy login dla sesji interaktywnych.
`Compression`	Domyślnie delayed. Określa czy pozwolić na kompresję lub opóźnić kompresję do czasu aż użytkownik zostanie uwierzytelniony z powodzeniem.
`ClientAliveInterval`	Domyślnie 0. Określa interwały czasowe (timeout interval) w sekundach dla serwera aby wysyłał komunikaty do klienta oczekując na odpowiedź.
`ClientAliveCountMax`	Domyślnie 3. Jeżeli ta dyrektywa jest ustawiona na inną wartość niż 0, określa ona maksymalną ilość komunikatów, jakie mają być wysłane do klienta.
`AllowUsers`	Zezwala na dostęp do serwera tylko umieszczonym na liście użytkownikom. Składnia wygląda następująco: `AllowUsers user1 user2` `AllowUsers user3@server2.example.com` `AllowUsers user4@192.168.0.110 user5@192.168.0.120`
`AllowGroups`	Zezwala na dostęp do serwera tylko członkom grup umieszczonym na liście. Składnia wygląda następująco: `AllowGroups dba unixadmins` `AllowGroups dba@server2.example.com`
`DenyUsers`	Nie pozwala umieszczonym na liście użytkownikom na dostęp do serwera. Składnia wygląda następująco: `DenyUsers user1 user2` `DenyUsers user3@server2.example.com` `DenyUsers user4@192.168.0.110 user5@192.168.0.120`
`DenyGroups`	Nie pozwala członkom grup umieszczonym na liście na dostęp do serwera. Składnia wygląda następująco: `DenyGroups dba unixadmins` `DenyGroups dba@server2.example.com`

Konfiguracje użytkownika

Osobne pliki konfiguracyjne użytkowników.

Plik o nazwie config z konfiguracją ssh dla danego użytkownika składowany jest w katalogu ***~/.ssh*** (podkatalog katalogu domowego tego użytkownika).

Format pliku ***~/.ssh/config*** jest taki sam jak pliku ***/etc/ssh/ssh_config***. Katalog ~/.ssh nie istnieje domyślnie, zakładany jest gdy użytkownik po raz pierwszy loguje się na serwer.

Katalog ~/.ssh zawiera ponadto plik o nazwie ***known_host***. Plik ten zawiera adres IP i kopię klucza publicznego systemu, do którego uzyskaliśmy dostęp. Przykładowo może wyglądać tak:

```
192.168.1.4 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBJUiDpnI4TxEOuAHmGU8MHhbDsmEraxp38qfOPIK1aj
lVzeJPb0Vwoim1PwAcQQGVbE+1xfKmXShqEjUQm4ew7M=
```

W katalogu, o którym mowa mogą także znajdować się pliki takie jak:

- `id_rsa, id_dsa, id_ecdsa` – składujące klucze prywatne
- `id_rsa.pub, id_dsa.pub, id_ecdsa.pub` – składujące klucze prywatne
- `authorized_keys` – wykorzystywany gdy ustawiamy relację zaufania między dwoma systemami.

Generowanie kluczy i logowanie z wykorzystaniem kluczy

Konfiguracja uwierzytelnienia w oparciu o klucz prywatny i publiczny.

1. Logujemy się na *server1* jako *user*. Generujemy klucze RSA. Wybieramy domyślne miejsce składowania kluczy, tj. wciskamy ENTER przy wyborze nazwy pliku. Hasło ustawiamy puste.

```
# ssh-keygen`
`Generating public/private rsa key pair.`
`Enter file in which to save the key (~/.ssh/id_rsa):`
`Enter passphrase (empty for no passphrase):`
`Enter same passphrase again:`
`Your identification has been saved in ~/.ssh/id_rsa.`
`Your public key has been saved in /root/.ssh/id_rsa.pub.`
`The key fingerprint is:`
`46:87:a9:c8:6b:bf:2c:f1:99:ab:77:1d:05:1f:2a:dc user@centos.host.pl`
`The key's randomart image is:`
`+--[ RSA 2048]----+`
`|         |`
`|     o. .  |`
`|    .+..+ .  |`
`|  . . oo.E o  |`
`|  o . S. .   |`
`|  .. . .   |`
`|  oo o . .   |`
`|  ..o= . .   |`
`|  .+=+     |`
`+-----------------+
```

Klucze zapisane zostają w katalogu `~/.ssh`.
Klucz publiczny: `~/.ssh/id_rsa.pub`
Klucz prywatny: `~/.ssh/id_rsa`

2. Plik z kluczem publicznym kopiujemy na serwer2 (adres IP=192.168.10.2) do katalogu domowego odpowiedniego użytkownika na drugim serwerze. Przy pierwszym logowaniu na serwer2 wyświetli się

zapytanie czy akceptujemy fingerprint (odcisk palca) dla drugiego serwera. Potwierdzamy *yes*.

```
# ssh-copy-id user@192.168.10.2
`The authenticity of host '192.168.10.2 (192.168.10.2)' can't be established.`
`ECDSA key fingerprint is a6:28:81:79:a2:8c:f9:0d:5c:39:a2:aa:f9:3f:12:6a.`
`Are you sure you want to continue connecting (yes/no)? yes`
`/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed`
`/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
```

3. Teraz możemy zalogować się na zdalnym serwerze bez użycia hasła.

```
$ ssh 192.168.10.2
```

Wykonywanie komend na zdalnym serwerze przy użyciu ssh.

```
$ ssh 192.168.10.2 /bin/ls -l
```

Kopiowanie plików przy użyciu scp.

Kopiowanie z lokalnego serwera pliku plik_lokalny na serwer host2:

```
$ scp plik_lokalny user@host2:/sciezka/do/pliku
```

Kopiowanie z serwera zdalnego pliku pliku2 na serwer lokalny:

```
$ scp user@host2:/sciezka/do/pliku2 plik_lokalny
```

Przesyłanie plików przy użyciu sftp.

```
$ sftp server2
`Enter passphrase for key '/home/user1/.ssh/id_rsa':`
`Connected to server2.`
`sftp>
```

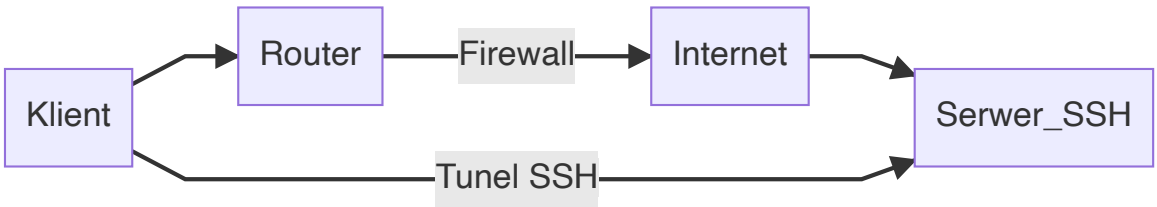
Tutaj możemy wpisać znak zapytania ? aby wyświetlić listę dostępnych komend.

```
sftp> ?`
```

```
`Available commands:`
`bye Quit sftp`
`cd path Change remote directory to 'path'`
`chgrp grp path Change group of file 'path' to 'grp'`
`chmod mode path Change permissions of file 'path' to 'mode'`
`chown own path Change owner of file 'path' to 'own'`
`df [-hi] [path] Display statistics for current directory or filesystem containing 'path'`
`exit Quit sftp`
`get [-Ppr] remote [local] Download file`
`reget remote [local] Resume download file`
`help Display this help text`
`lcd path Change local directory to 'path'`
`lls [ls-options [path]] Display local directory listing`
`lmkdir path Create local directory`
`ln [-s] oldpath newpath Link remote file (-s for symlink)`
`lpwd Print local working directory`
`ls [-1 afhlnrSt] [path] Display remote directory listing`
`lumask umask Set local umask to 'umask'`
`mkdir path Create remote directory`
`progress Toggle display of progress meter`
`put [-Ppr] local [remote] Upload file`
`pwd Display remote working directory`
`quit Quit sftp`
`rename oldpath newpath Rename remote file`
`rm path Delete remote file`
`rmdir path Remove remote directory`
`symlink oldpath newpath Symlink remote file`
`version Show SFT P version`
`! command Execute 'command' in local shell`
`! Escape to local shell`
`? Synonym for help
```

Dodatek

Tunelowanie SSH



Mówiąc o tunelowaniu SSH mamy na myśli forwardowanie jakiegoś portu przez port szyfrowany SSH. W tym przypadku musimy posiadać dostęp do Serwera_SSH z zewnętrznym adresem IP.

Istnieją trzy typy forwardowania SSH: **local, remote, dynamic.**

Remote Forwarding

Ten typ połączenia pozwala na połączenie z Twojego zdalnego komputera do innego lokalnego komputera w sieci lokalnej. Domyślnie ten typ forwardowania jest zablokowany i, aby go odblokować, należy dodać do pliku konfiguracyjnego sshd (np. w

/etc/ssh/sshd_config) dyrektywę:

GatewayPorts yes

Aby przekierować ruch i dostać się na port 80 zdalnej maszyny, wykonamy poniższą linijkę. Port 80 zdalnej maszyny będzie przekierowany na port 8080 naszej maszyny lokalnej.

```
$ ssh -f -N user@Serwer_SSH -R 80:localhost:8080
```

Local Forwarding

Ten typ forwardowania pozwala na połączenie się z Twojej maszyny lokalnej, na jakiś zasób na zdalnej maszynie, na której port, do którego chcesz się dostać jest zablokowany. Dla przykładu, jeśli jesteś w sieci lokalnej, zabezpieczonej, w której wyjście na dany port w sieci zdalnej jest zablokowane, to dzięki forwardowaniu typu local możesz do tego zasobu się dostać.

Jeśli zatem z twojej sieci lokalnej nie możesz połączyć się na port 80 w jakimś zdalnym zasobie (serwerze), to możesz przekierować przez port SSH ten ruch na lokalny port 8080 Twojego komputera. Dzięki temu otworzysz przeglądarkę, wpiszesz http://localhost:8080 i uzyskasz dostęp do zasobu zlokalizowanego na http://Serwer_SSH

```
$ ssh -f -N user@Serwer_SSH -L 8080:Serwer_SSH:80
```

Dynamic Forwarding

Jest ostatnim z trzech typów forwardowania i umożliwia, w przeciwieństwie do local i remote forwarding, komunikację na większej ilości portów. Jest najbardziej ciekawym i użytecznym z perspektywy użytkownika typem forwardowania/tunelowania. Za jego pomocą możemy omijać często restrykcje w sieciach, o czym w dalszej części. Poprzednie dwa typy umożliwiały przekierowanie tylko na jeden konkretny port. Dynamic forwarding umożliwia przekierowanie całego zakresu TCP. Ten typ forwardowania tworzy na Twoim komputerze serwer socks proxy, dzięki któremu na wyjściu twojego komputera możesz przyjąć IP serwera np. serwer.pl i przez niego technicznie komunikować się z innymi zasobami.

```
$ ssh -f -N -D 9000 user@Serwer_SSH
```