Linux Operating System and Application

SSH Service

Learning Objectives

- Understand the concept and role of SSH.
- Learn to use the ssh command for remote connection.
- Install and configure the SSH service (sshd).
- Enhance SSH security.
- Practice connecting and transferring files via SSH.

What is SSH?

- SSH (Secure Shell) is a network protocol and a set of standards that establishes a secure channel between two computers for remote login and command execution.
- It was originally designed to replace insecure applications like Telnet, Rlogin, rsh, and rcp.
- Key features of SSH:
  - Encrypts the entire session to prevent eavesdropping (e.g., network sniffing, MITM).
  - Provides mutual authentication, data confidentiality, and data integrity.
- Traditional applications had drawbacks such as authentication based on IP address or reusable passwords, clear text data transmission, and vulnerability of the X protocol.

How SSH Works

- Client-Server Architecture:
  - A SSH server (sshd daemon) runs on the server side, listening for incoming connections.
  - A client executes the ssh command to connect to the remote host.
- Default Port: SSH connects via port 22/TCP.
- Authentication: Users are authenticated using either a username and password or public key authentication.
- Encryption: The session is encrypted using strong algorithms like AES, RSA, etc.
- The SSH-2 Protocol is built upon three layers:
  - Transport Layer: Handles initial key exchange, server authentication, data confidentiality, and integrity.
  - User Authentication Layer: Manages client authentication, offering various methods.
  - Connection Layer: Defines logical channels for services like interactive shell sessions, TCP port forwarding, and X11 forwarding

Basic SSH Command

- Syntax: *ssh user@remote_host*
  - *user*: The username on the remote host.
  - *remote_host*: The hostname or IP address of the remote server.
- Example: *ssh student@192.168.1.10*
- Options:
  - -p <port>: Specify a non-default port for the connection.
  - -i <private_key_file>: Use a specific private key file for authentication.
- You can also execute commands directly on the remote host:

  ssh [username@hostname_or_IP_address] "command"

Install SSH Server

- OpenSSH is often installed by default, even with a minimal CentOS installation, so additional package installation might not be necessary.
- Installation commands:
  - On Debian/Ubuntu:

    sudo apt update

    sudo apt install openssh-server
  - On CentOS/RHEL:

    sudo yum install openssh-server
    - For the client, *yum -y install openssh-clients* can be used.
- Start and Enable the SSH Service:

  sudo systemctl start ssh

  sudo systemctl enable ssh
- Firewall Configuration: If Firewalld is running, allow the SSH service (which uses port 22/TCP):

  firewall-cmd --add-service=ssh --permanent

  firewall-cmd --reload

SSH Configuration

- The main configuration file for the SSH daemon (sshd) is located at: /etc/ssh/sshd_config.
- Key options to configure:
  - **Port 22**: Specifies the port sshd listens on (default is 22). You can change this for security.
  - **PermitRootLogin no**: Prohibits direct remote root login for enhanced security.
  - **PasswordAuthentication yes/no**: Controls whether password authentication is allowed. For better security, it should be set to no if key-pair authentication is used.
  - **ChallengeResponseAuthentication no**: Should be set to no if password authentication is disabled.
  - **PubkeyAuthentication yes**: Enables public key authentication.
  - **AllowUsers** *student*: Limits SSH access to specific users.
- After editing the configuration file, you must restart the SSH service for changes to take effect:

  sudo systemctl restart ssh (or *systemctl restart sshd* on CentOS 7)

File /etc/ssh/sshd_config

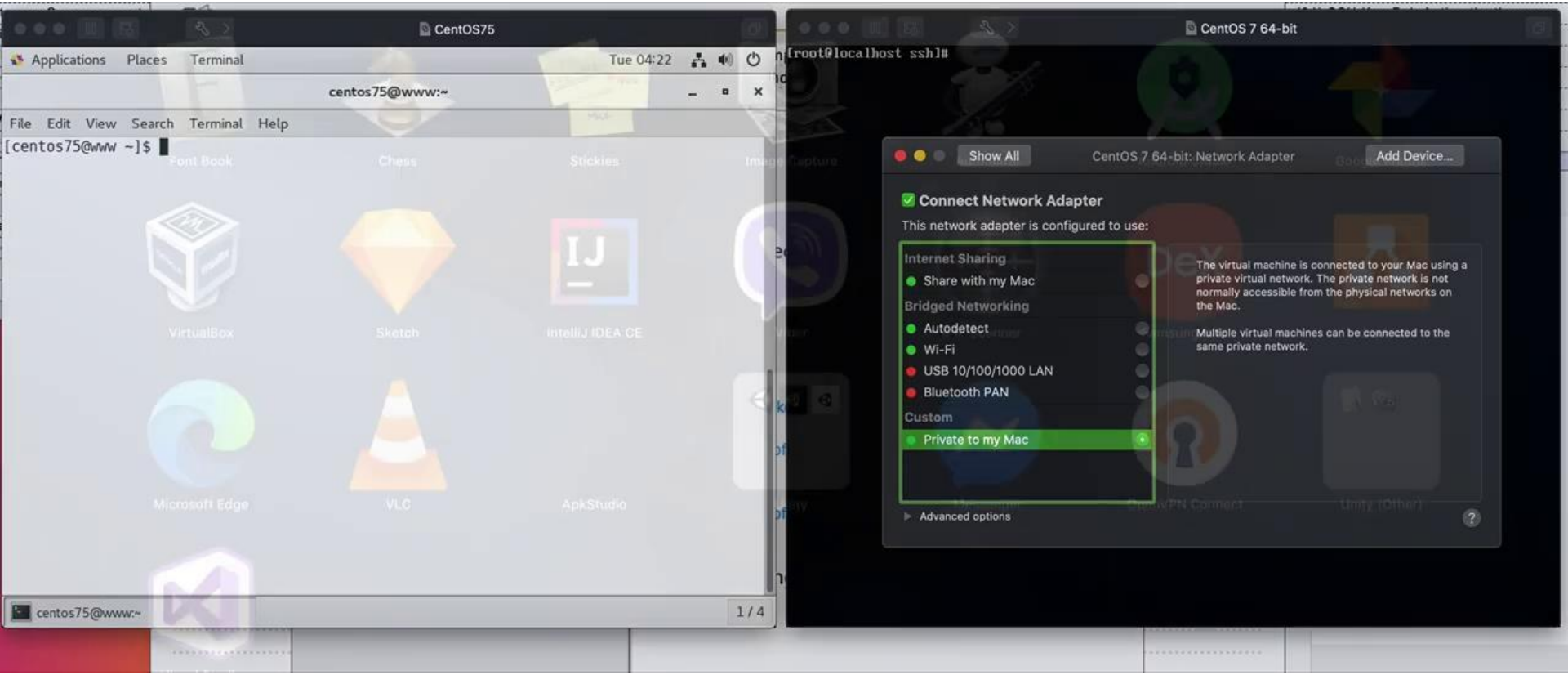
```
#Port 22
#Protocol 2,1
Protocol 2
SyslogFacility AUTHPRIV
# Authentication:
PasswordAuthentication yes
ChallengeResponseAuthentication no
# GSSAPI options
GSSAPIAuthentication yes
GSSAPICleanupCredentials yes
UsePAM yes
# Accept locale-related environment variables
AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL
X11Forwarding yes
# override default of no subsystems
Subsystem       sftp    /usr/libexec/openssh/sftp-server
```

File /etc/ssh/ssh_config

```
# Host *
#   ForwardAgent no
#   ForwardX11 no
#   RhostsRSAAuthentication no
#   RSAAuthentication yes
#   PasswordAuthentication yes
#   HostbasedAuthentication no
#   BatchMode no
#   CheckHostIP yes
#   AddressFamily any
#   ConnectTimeout 0
#   StrictHostKeyChecking ask
#   IdentityFile ~/.ssh/identity
#   IdentityFile ~/.ssh/id_rsa
#   IdentityFile ~/.ssh/id_dsa
#   Port 22
#   Protocol 2,1
#   Cipher 3des
#   Ciphers aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc
#   EscapeChar ~
#   Tunnel no
#   TunnelDevice any:any
#   PermitLocalCommand no
Host *
        GSSAPIAuthentication yes
        ForwardX11Trusted yes
# Send locale-related environment variables
        SendEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
        SendEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
        SendEnv LC_IDENTIFICATION LC_ALL
```

Demo

SSH: Password Authentication



SSH Security

- Do not use the **root** account for direct remote logins. Create and use a regular user account instead.
- Disable password authentication and exclusively rely on SSH **key authentication**.
- Change the default SSH port (22) to a **non-standard port** (e.g., 8022) to reduce automated attacks.
- Utilize **firewall** rules to restrict SSH access to specific IP addresses.
- Consider installing intrusion prevention tools like **fail2ban** to prevent brute-force attacks.
  - Fail2Ban reads log files for defined patterns (e.g., "authentication failure").
  - If a threshold is exceeded (e.g., 5 failed logins in 10 mins), it triggers a ban. The ban is usually done via firewall (iptables, firewalld, etc.).

File Transfer via SSH

- SSH provides secure methods for file transfer
- **scp** (Secure Copy):
  - Used for copying files and directories between local and remote hosts.
  - Syntax: *scp [Option] Source Target*
  - Example (local to remote):

    scp ./local_file.txt user@remote_host:~/remote_directory/
  - Example (remote to local):

    scp user@remote_host:/path/to/remote_file.txt ./local_directory/

File Transfer via SSH

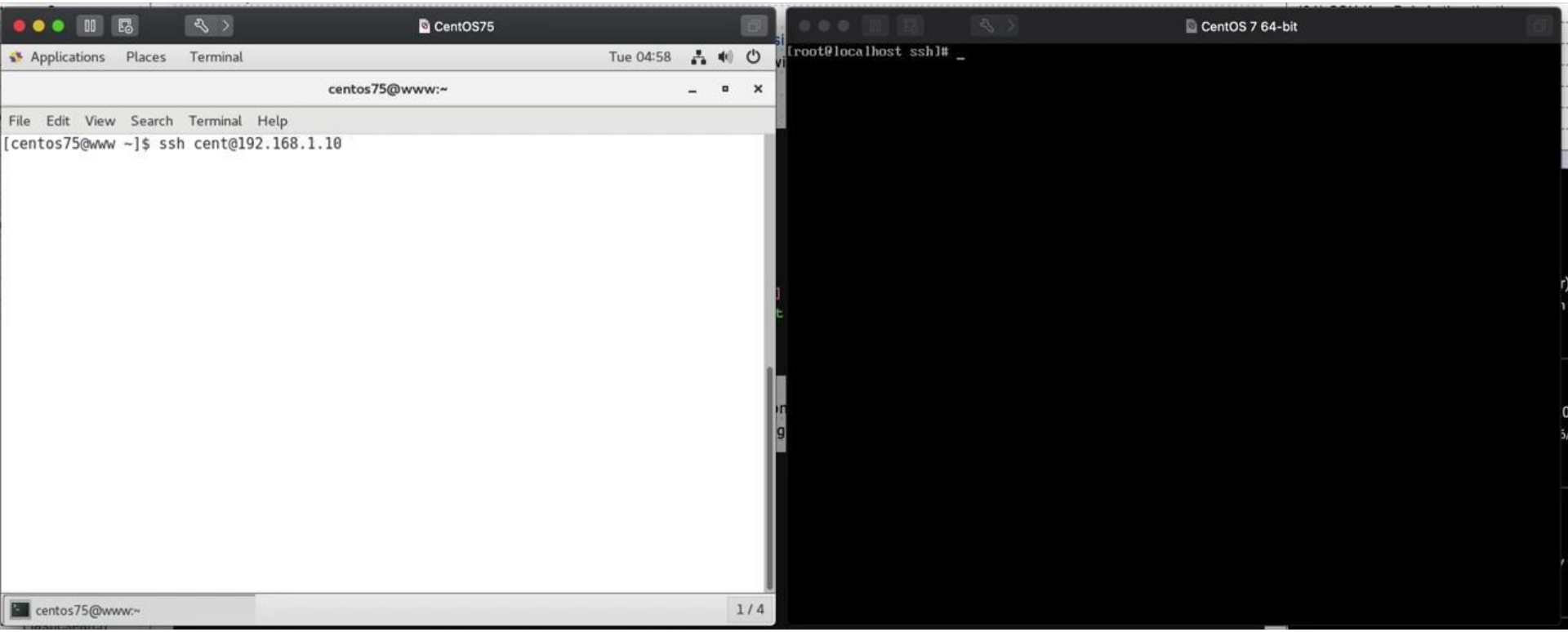
- **sftp** (SSH File Transfer Protocol):
  - Provides an interactive command-line interface for file transfer over SSH, similar to FTP but secure.
  - Usage: *sftp user@remote_host*
  - Common sftp commands include **pwd** (remote), **!pwd (local)**, **ls -l** (remote), **!ls -l** (local), **cd**, **put** (upload), **get** (download), **mkdir**, **rmdir**, **rm**.
  - SFTP server functionality is enabled by default in OpenSSH, but can be explicitly configured via Subsystem sftp **/usr/libexec/openssh/sftp-server** in **/etc/ssh/sshd_config**.
  - SFTP can be configured with **Chroot** to restrict users to specific directories

File Transfer via SSH

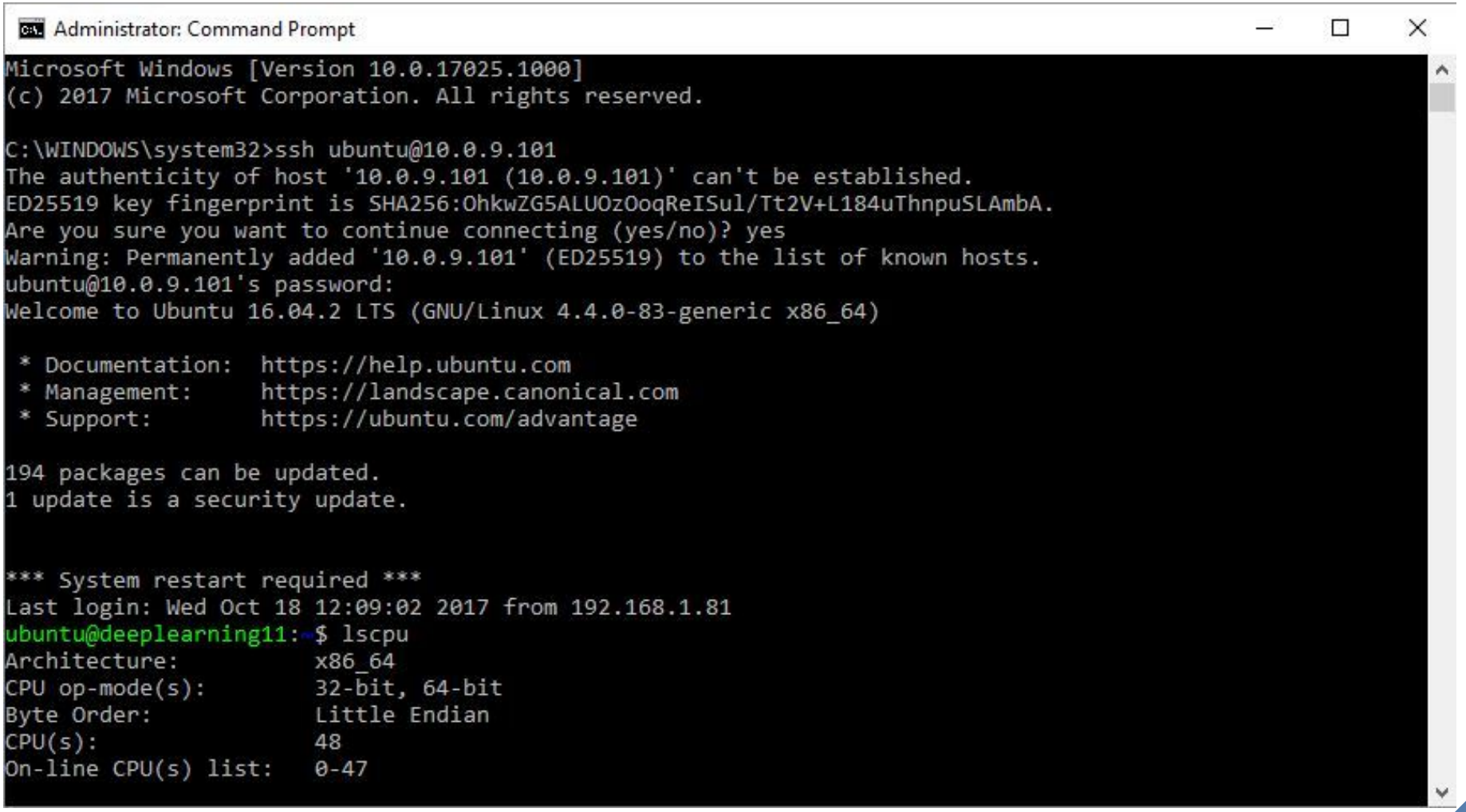
- **rsync**: Another tool for efficient file transfer and synchronization over SSH.
  - Usage: *rsync -avz file.txt user@remote:/home/user/*
- **Windows Clients**:
  - Tools like **WinSCP** provide a graphical interface for SFTP.
  - Windows 10 Version 1803 and later include a built-in OpenSSH client, allowing the use of scp and sftp commands directly from the command prompt.

Demo

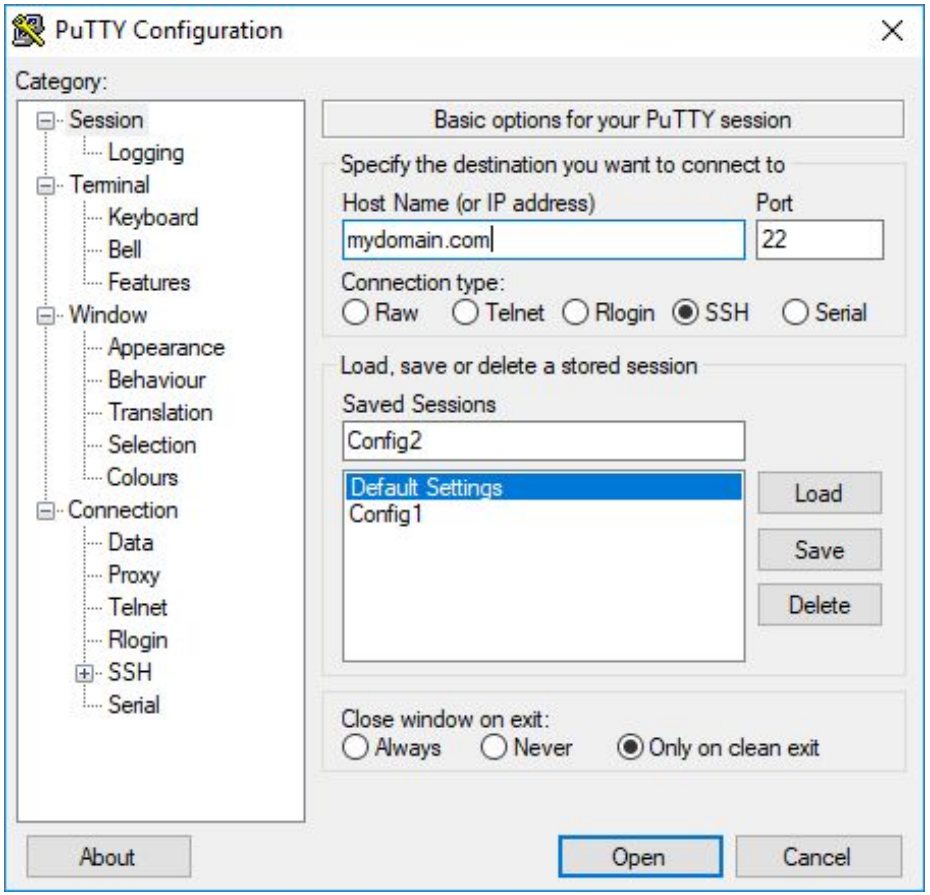
scp

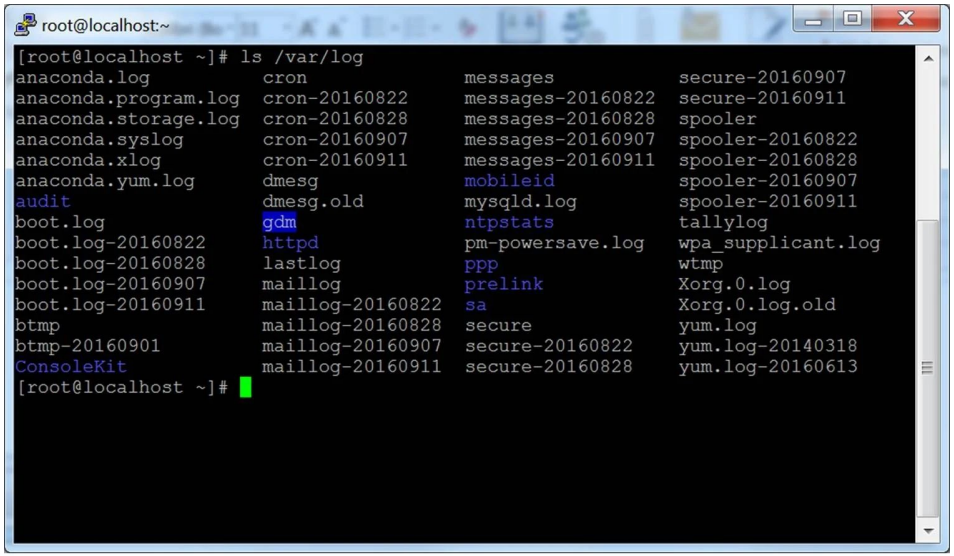

CentOS75
Applications Places Terminal
Tue 04:58
centos75@www:~
File Edit View Search Terminal Help
[centos75@www ~]$ ssh cent@192.168.1.10
centos75@www:~
1 / 4
CentOS 7 64-bit
[root@localhost ssh]# _

OpenSSH


Administrator: Command Prompt
Microsoft Windows [Version 10.0.17025.1000]
(c) 2017 Microsoft Corporation. All rights reserved.
C:\WINDOWS\system32>ssh ubuntu@10.0.9.101
The authenticity of host '10.0.9.101 (10.0.9.101)' can't be established.
ED25519 key fingerprint is SHA256:OhkwZG5ALUOzOoqReISul/Tt2V+L184uThnpuSLAmbA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.9.101' (ED25519) to the list of known hosts.
ubuntu@10.0.9.101's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-83-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
194 packages can be updated.
1 update is a security update.
*** System restart required ***
Last login: Wed Oct 18 12:09:02 2017 from 192.168.1.81
ubuntu@deeplearning11:~$ lscpu
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 48
On-line CPU(s) list: 0-47

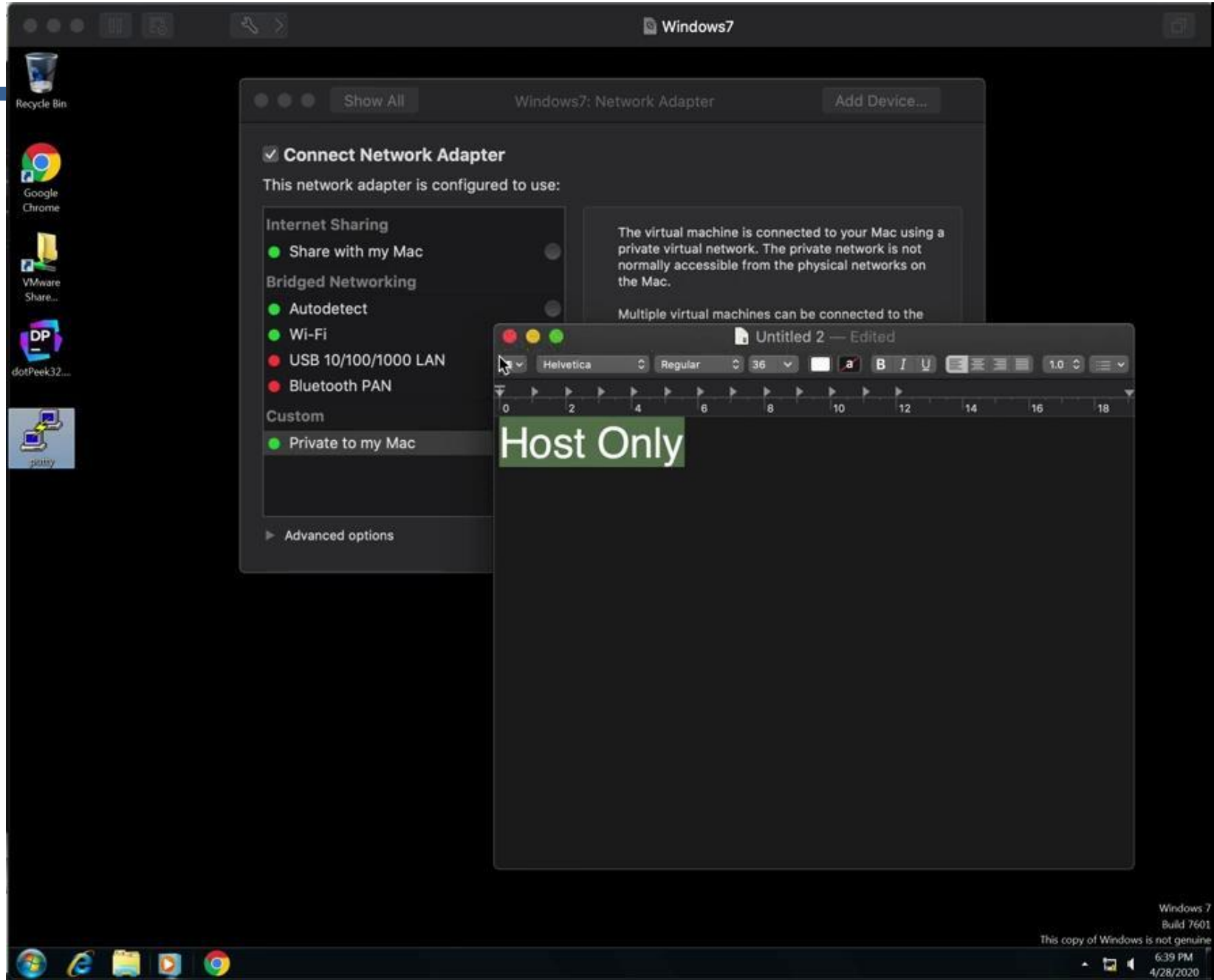
Putty


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
mydomain.com
22
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
Config2
Default Settings
Config1
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel

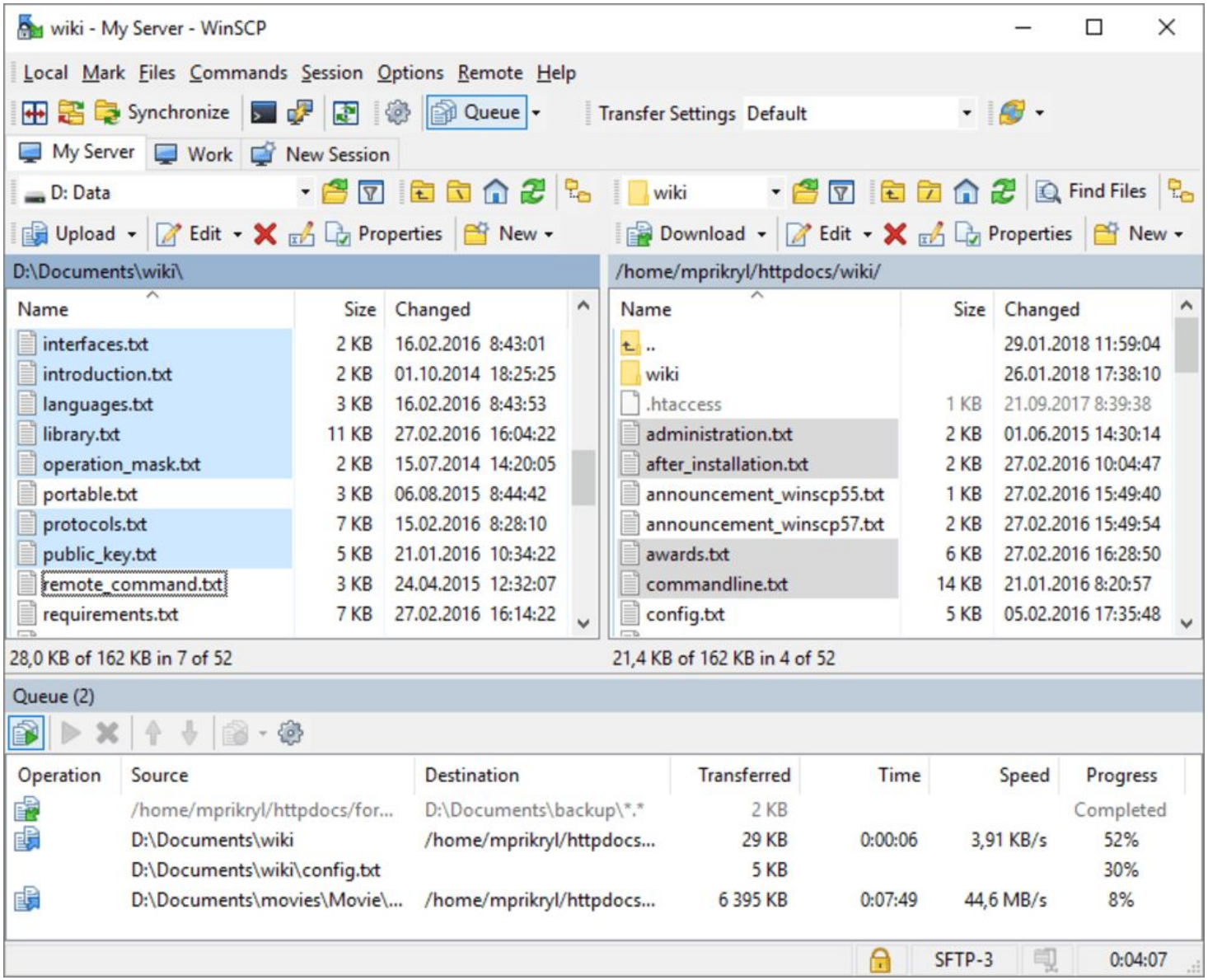

root@localhost:~
[root@localhost ~]# ls /var/log
anaconda.log
anaconda.program.log
anaconda.storage.log
anaconda.syslog
anaconda.xlog
anaconda.yum.log
audit
boot.log
boot.log-20160822
boot.log-20160828
boot.log-20160907
boot.log-20160911
btmp
btmp-20160901
ConsoleKit
cron
cron-20160822
cron-20160828
cron-20160907
cron-20160911
dmesg
dmesg.old
gdm
httpd
lastlog
maillog
maillog-20160822
maillog-20160828
maillog-20160907
maillog-20160911
messages
messages-20160822
messages-20160828
messages-20160907
messages-20160911
mobileid
mysqld.log
ntpstats
pm-powersave.log
ppp
prelink
sa
secure
secure-20160822
secure-20160828
secure-20160907
secure-20160911
spooler
spooler-20160822
spooler-20160828
spooler-20160907
spooler-20160911
tallylog
wpa_supplicant.log
wtmp
Xorg.0.log
Xorg.0.log.old
yum.log
yum.log-20140318
yum.log-20160613
[root@localhost ~]#

Demo

Putty


Windows7
Recycle Bin
Google Chrome
VMware Share...
dotPeek32...
putty
Show All
Windows7: Network Adapter
Add Device...
Connect Network Adapter
This network adapter is configured to use:
Internet Sharing
Share with my Mac
Bridged Networking
Autodetect
Wi-Fi
USB 10/100/1000 LAN
Bluetooth PAN
Custom
Private to my Mac
Advanced options
The virtual machine is connected to your Mac using a private virtual network. The private network is not normally accessible from the physical networks on the Mac.
Multiple virtual machines can be connected to the
Untitled 2 — Edited
Helvetica
Regular
36
Host Only
Windows 7
Build 7601
This copy of Windows is not genuine
6:39 PM
4/28/2020

WinSCP


wiki - My Server - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize
Queue
Transfer Settings Default
My Server
Work
New Session
D: Data
wiki
Find Files
Upload
Edit
Properties
New
Download
Edit
Properties
New
D:\Documents\wiki\
/home/mprikryl/httpdocs/wiki/
Name
Size
Changed
interfaces.txt 2 KB 16.02.2016 8:43:01
introduction.txt 2 KB 01.10.2014 18:25:25
languages.txt 3 KB 16.02.2016 8:43:53
library.txt 11 KB 27.02.2016 16:04:22
operation_mask.txt 2 KB 15.07.2014 14:20:05
portable.txt 3 KB 06.08.2015 8:44:42
protocols.txt 7 KB 15.02.2016 8:28:10
public_key.txt 5 KB 21.01.2016 10:34:22
remote_command.txt 3 KB 24.04.2015 12:32:07
requirements.txt 7 KB 27.02.2016 16:14:22
.. 29.01.2018 11:59:04
wiki 26.01.2018 17:38:10
.htaccess 1 KB 21.09.2017 8:39:38
administration.txt 2 KB 01.06.2015 14:30:14
after_installation.txt 2 KB 27.02.2016 10:04:47
announcement_winscp55.txt 1 KB 27.02.2016 15:49:40
announcement_winscp57.txt 2 KB 27.02.2016 15:49:54
awards.txt 6 KB 27.02.2016 16:28:50
commandline.txt 14 KB 21.01.2016 8:20:57
config.txt 5 KB 05.02.2016 17:35:48
28,0 KB of 162 KB in 7 of 52
21,4 KB of 162 KB in 4 of 52
Queue (2)
Operation
Source
Destination
Transferred
Time
Speed
Progress
/home/mprikryl/httpdocs/for... D:\Documents\backup*.* 2 KB Completed
D:\Documents\wiki /home/mprikryl/httpdocs... 29 KB 0:00:06 3,91 KB/s 52%
D:\Documents\wiki\config.txt 5 KB 30%
D:\Documents\movies\Movie\... /home/mprikryl/httpdocs... 6 395 KB 0:07:49 44,6 MB/s 8%
SFTP-3
0:04:07

SSH Key-Pair Authentication

- A more secure alternative to password authentication.
- Involves a pair of cryptographic keys: a private key (kept secret on the client machine) and a public key (stored on the server).
- Process:
  - 1. **Generate Key-Pair**: On the client machine, use ssh-keygen to create the keys.
    - *ssh-keygen -t rsa -b 4096 -o -a 100* for a strong RSA key.
    - (Better) *ssh-keygen -t ed25519 -o -a 100* for modern Ed25519 keys.
    - Always protect your private key with a strong passphrase

SSH Key-Pair Authentication

- Process:
  - 2. **Deploy Public Key**: Copy the public key (id_rsa.pub or id_ed25519.pub) to the server's ~/.ssh/authorized_keys file
  - 3. **Client Authentication**: When connecting, the server challenges the client, which then proves its identity using the private key.
- Important: The private key (id_rsa or id_ed25519) must be kept SECRET and never shared.
- The first time a client connects to a server, the server's public key is saved in the client's ~/.ssh/known_hosts file to verify authenticity on subsequent connections

```
C:\Users\leham>ssh cent@192.168.47.128
The authenticity of host '192.168.47.128 (192.168.47.128)' can't be established.
ED25519 key fingerprint is SHA256:MYx7dl+NQTXxw5RMt046NCvQPHyZq1kMBSuxzwlQ444.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Client Authentication with Public Key in SSH: How Server Challenges Client

1. **Client says:** "Hi, I want to log in as userX, and here's my **public key** (or its fingerprint)."
2. **Server checks:**
   a. Does this public key exist in ~/.ssh/authorized_keys for userX?
   b. If **yes**, it proceeds to the challenge.
3. **Server generates a challenge:**
   a. It creates a **random piece of data** (a challenge string or nonce).
   b. Then it **encrypts** this data using the **client's public key**.
4. **Server sends the encrypted challenge** to the client.
5. **Client receives it and decrypts** the challenge using its **private key**.
6. **Client responds** with the decrypted challenge as proof of ownership of the private key.
7. **Server verifies** that the decrypted data matches what it originally sent.

Demo

- SSH Key Authentication: https://www.youtube.com/watch?v=5Fcf-8LPvws
- https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server

SSH Port Forwarding (SSH Tunneling)

- SSH port forwarding allows tunneling application ports securely over an encrypted SSH connection.
- This adds security to applications that don't natively support encryption.
- Types of Port Forwarding:
  - **Local Forwarding (-L)**: Forwards a port from your local machine to a port on the remote server.
  - Purpose: Access a service on the remote server as if it were running on your local machine.
  - Example: Securely access a remote MySQL server (port 3306):
  - Then connect your MySQL client to localhost:3306 locally

SSH Port Forwarding (SSH Tunneling)

- **Remote Forwarding (-R)**: Forwards a port from the remote server to a port on your local machine.
  - Purpose: Allow a service on the remote server to access a service running on your local machine.
  - Example: Allow a remote server to access your local web server (port 8080):
  - On the remote server, access localhost:9090.
- **Dynamic Forwarding (-D)**: SSH acts as a SOCKS proxy for versatile traffic.
  - Purpose: Create a secure proxy for web browsing or other network traffic through the SSH tunnel.
  - Example: Create a SOCKS proxy on local port 1080:
  - Then configure your browser or applications to use a SOCKS5 proxy at localhost:1080.
- Client tools like Putty on Windows also support port forwarding configuration

X11 Forwarding (Graphical Interface Forwarding)

- Purpose: Allows you to run graphical user interface (GUI) applications on a remote server and have their display forwarded to your local client machine via SSH.
- Requirements:
  - Your local client machine must have an X server installed (e.g., XQuartz on macOS, Xming on Windows, Linux distributions usually have it by default).
  - The remote machine must have X11 system and libraries installed.
  - The remote machine does NOT need to be logged into a graphical desktop session. SSH X11 forwarding creates a virtual display for GUI applications over SSH, even if the server is running in text mode.
- Usage:
  - Connect to the remote host with the -X option:
  - Once connected, run a GUI application (e.g., xclock) on the remote server, and its window will appear on your local desktop

X11 Forwarding

Example