**SSH Configuration Parameters [ /etc/ssh/sshd\_config ]**

**Working of SSH**

1. The user initiates an SSH connection by running the ssh command on their local machine followed by the server’s IP address or hostname
2. The client connects to the SSH server and receives the server’s public key, which it checks against the **known\_hosts** file on the client machine. If it is the first time connecting, the client may be asked to verify the server’s identity.
3. Once the server’s identity is verified, the authentication process begins. The user can authenticate in two ways: **password-based authentication** or **public key-based authentication**.

* **Password-Based Authentication**: If this method is used, the server will ask for the user’s password. The client sends the password over the encrypted connection, and the server checks the password against its stored credentials. If the password is correct, the client is authenticated and granted access.
* **Public Key Authentication**: In this method, the client uses a **public/private key pair** for authentication. The client keeps the **private key** on their machine, while the **public key** is stored in the ~/.ssh/authorized\_keys file on the server. During the SSH connection, the server sends a challenge encrypted with the client’s public key. The client decrypts it using their private key and responds by signing the data using its private key. If the server can verify the response with the corresponding public key, authentication is successful.

1. After authentication, a secure and encrypted session is established using a **key exchange** algorithm (shared key is generated). This ensures that both the client and server can securely communicate and establish a shared secret key.
2. The communication between the client and the server is then encrypted using symmetric encryption (such as AES), ensuring the confidentiality of data.

**Port** **<port\_number>** - Change the default port (22) to something else for security.

**PermitRootLogin <yes/no>** - To enable or disable root login for security.

**PasswordAuthentication <yes/no>** - To enable or disable password login and use keys to login instead.

**AllowUsers <usernames>** - To give ssh access to specific users in the system.

**AllowGroups <groupnames>** - To give ssh access to specific groups in the system.

**DenyUsers <usernames>** - To deny ssh access to specific users in the system.

**DenyGroups <groupnames>** - To deny ssh access to specific groups in the system.

**MaxAuthTries <value>** - To limit the number of authentication attempts per connection.

**LoginGraceTime <time>** - Specifies the time allowed for a user to authenticate. If the user fails to log in within this time, the connection is closed.

**PubkeyAuthentication <yes/no>** - To enable or disable public key authentication.

**MaxSessions <value>** - To Limit the number of sessions per user, (i.e.) number of terminal tabs with which a single user can use ssh to connect to remote system.

**MaxStartups <value>** - To control how many unauthenticated connections the SSH server allows at once.

**X11Forwarding <yes/no>** - Allows SSH to forward graphical applications (web browser, etc) from a remote server to your local machine (i.e.) though ssh is CLI based when i open graphical apps it open up in my local system instead of opening in the ssh server.

**AllowTcpForwarding <yes/no>** - To control whether TCP forwarding (port forwarding) is allowed.

**UseDNS <yes/no>** -To allow/deny the SSH server perform DNS lookups on connecting clients. Disabling it can speed up login times.

**Banner <banner path>** -Specifies a text file to display a legal banner (e.g., warning message) before login.

**Compression <yes/no>** -Specifies whether to enable compression.

**HostKey <host key path>** -Lists the private key files used by the server for host authentication.

**LogLevel <value>** -It determines how much detail the SSH server includes in the log file (/var/log/auth.log). It can take The following values - QUIET, FATAL, ERROR, DEBUG, INFO, VERBOSE.

**ClientAliveInterval <time>** -It sets a timeout interval (in seconds) for the SSH server to check if the client is still responsive. The server sends a keepalive message to the client after the specified interval of inactivity. If the client responds, the connection stays open. If not, the server waits for additional retries (controlled by ClientAliveCountMax) before closing the connection.

**ClientAliveCountMax <value>** - It works with ClientAliveInterval and specifies how many keepalive messages the server sends to the client before deciding the client is unresponsive and closing the connection.