↖ Topic Index

The ssh command 🚧

Operation			
The SSH commands 🚧 Last updated on:	The suite of SSH commands is ubiquitously useful whenever you need to work on remote hosts. This table provides some useful information. It is a WIP, as more info will be added.🚧 2025-09-18		
Create a SSH relationship with remote server (such as Gitlab, Github, etc...)	• Make sure ssh is installed:	`ssh -V`	This will print the version of SSH and TLS/SSL used.

	• Ensure that the ~/.ssh directory exists and has the correct permission (700), otherwise create it:	`cd`
		`mkdir .ssh`
		`chmod 700 .ssh`
			The permission of the **~/.ssh** directory is important. An invalid permission may cause issue with several operations.
			Use the command `ls -lda ~/.ssh` to confirm the permission should show: **drwx------**

	• Create a pair of SSH keys	The <comment> is anything useful for you to remind you the purpose of the key from the point of view of the SSH server. If you use several computers you may want to identify the name of the computer from where you'd be using this key to access the server. • The ed25519 key type is often the preferred format. Use something else if identified by the SSH server. • This will prompt for a passphrase. Although enhancing security, automatic ssh operations might be difficult to implement and therefore the passphrase is often not used. • It will also prompt for the **key name** and will propose the name based on the key type. It's best to identify the purpose of the key in the name. I normally use the purpose, an underscore, and the key type identifier.	
		`ssh-keygen -t ed25519 -C "<comment>"`	
	• Check if the SSH agent is running • In most situations it is best to use only **one** SSH agent.	The **ssh-agent-isit-running** is a shell script part of **USRHOME project**. • It checks if the SSH agent is running and report a problem if none or more than one agent is running. • The exit code is 0 if there is only one ssh agent running.	
		`ssh-agent-isit-running`	
	• If more than one SSH agent is running, stop any supplemental SSH agent: • The **ssh-agent-isit-running** command lists the process IDs (2nd column) of all running SSH agents and the start date/time of each (9th column). • Identify the process ID of the SSH agents you want to stop. • Run the following commands for each process ID of SSH agent you want to stop. • Replace <PID> by the process ID number of the SSH agent you want to stop:		
		`SSH_AGENT_PID=<PID>; export SSH_AGENT_PID;` `eval $(ssh-agent -k)`	
	• Add the keys locally	`cd ~/.ssh` `ssh-add <key name>`	The <key name> is the name of the key file that was created by the ssh-keygen command above. • That name excludes any extension.
	• Update the **~/.ssh/config** file, ensuring that the new key and the remote server is identified. • It is also useful to write a comment identifying the purpose of the key, the date it was created, and the expiration date.		

	For example, I add something like the following to the file for a key created for accessing Gitlab, with the <date> and <date/time> fields set to the appropriate values.	`# Gitlab General - Added <date>. Expires <date/time>`
		`Host gitlab.com`
		`  AddKeysToAgent yes`
		`  UseKeychain yes`
		`  IdentityFile ~/.ssh/gitlab_id_ed25519`
		`# ----------------------------------------------------------------------------`	

	• Copy the content of the **public** key into the appropriate location in the SSH server (something like Gitlab, Github, a remote host, etc...)		
	• In the commands, replace <keyname> by the name of the key file you created above. • The copied data must be free of any new line character. • Copy the data to the SSH server	• On macOS, use **pbcopy** to copy stdout to the OS pasteboard.	`tr -d '\n' < ~/.ssh/<keyname>.pub | pbcopy`
		• On Linux, use the **xclip** command to copy the data to the X clipboard.	`xclip -sel clip < ~/.ssh/<keyname>.pub`
	• At this point you should be able to perform a git clone of a remotely hosted repository via ssh.		
Setting up remote host	On the remote host, do the following:		
• **Allowing SSH password-less connections to that host.**	• Make sure ssh is installed:	`ssh -V`	This common will print the version of SSH and TLS/SSL used.

	• Ensure that the ~/.ssh directory exists and has the correct permission (700), otherwise create it:	`cd`
		`mkdir .ssh`
		`chmod 700 .ssh`
			The permission of the **~/.ssh** directory is important. An invalid permission may cause issue with several operations.
			Use the command `ls -lda ~/.ssh` to confirm the permission should show: **drwx------**

	• From the **client host**, where you will use the ssh client, establish a SSH trust relationship with the remote host. • Use the **ssh-copy-id** command for this: • specify the username@REMOTE-HOST-IP-ADDRESS in the commands	`ssh-copy-id REMOTE-HOST-USERNAME@REMOTE-HOST-IP-ADDRESS` • The REMOTE-HOST-USERNAME is the username you use on the remote host. • The REMOTE-HOST-IP-ADDRESS is the IP address or recognized DNS name of the remote host. If the IP address of the remote host is statically allocated, then using the IP address is often the best choice, otherwise use the DNS name.	
	• From the server host, the remote host you will connect to:	Edit the file `~/.ssh/authorized_keys` : remove any public key in excess and leave only the keys that are necessary.	
	• At this point you should be able to access the remote host via ssh using the following command, with `<username>` replaced by your user name on the remote host and `<hostname-or-ip>` by the DNS hostname or it's IP address.	`ssh -l <username> <hostname-or-IP>`	

SSH References

SSH	• ssh (secure shell protocol) @ Wikipedia • ssh-file transfer protocol @ Wikipedia • ssh-agent @ Wikipedia • ssh-keygen @ Wikipedia • sshfs @ Wikipedia • ssh-copy-id @ ssh.com • Web-based SSH @ Wikipedia	• Comparison of SSH clients @ Wikipedia • OpenSSH @ Wikipedia • DropBear @ Wikipedia • Comparison of SSH Servers @ Wikipedia • OpenSSH @ Wikipedia • DropBear @ Wikipedia • GNU lsh @ Wikipedia
SSL & TLS	• SSL @ Wikipedia • TLS @ Wikipedia	• Comparison of TLS implementations @ Wikipedia • OpenSSL @ Wikipedia • LibreSSL @ Wikipedia • GnuTLS @ Wikipedia
	• What is a passphrase & how to use it @ ssh.com	
Using SSH with Gitlab	• SSH keys @ Gitlab provides a good overview of SSH keys and how to set up a SSH connection with your Gitlab account.	
SSH Key Types		
• **ED25519**	ED25519 @ Wikipedia	
• **ED25519_SK**	ED25519_SK	
• **ECDSA_SK**	ECDSA_SK	

Operation							
• **RSA**	RSA @ Wikipedia						
• **ECDSA**	ECDSA @ Wikipedia						