Setup Linux ssh for key based login

We need to create a private/public key set to allow passwordless login via ssh. To do this run the sshkey-ge command:

```
$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/testuser1/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/testuser1/.ssh/id_ecdsa
Your public key has been saved in /home/testuser1/.ssh/id_ecdsa.pub
The key fingerprint is:
SHA256:xb4Rs37UbXt3Wn5cHkdKWy2ZDBbor9F83IYNLhjsfIU
testuser1@bree.strw.leidenuniv.nl
The key's randomart image is:
+---[ECDSA 256]---+
|           ...   |
|         .. o    |
|         o=. + o.|
|         o++E.O.+|
|        So+*.=.@o|
|         .=+* BoB|
|          o+.o =O|
|           ..   +B|
|              . o|
+----[SHA256]-----+
```

For both questions about passphrase, just could hit enter (in that case we will not be using passphrases). Security wise it is better though you do fill in a passphrase. This will have generated two files in your personal .ssh directory:

```
$ ls -ltr id_ecdsa*
-rw------- 1 testuser1 users 537 Mar 22 12:13 id_ecdsa
-rw-r--r-- 1 testuser1 users 195 Mar 22 12:13 id_ecdsa.pub
```

The file `id_ecdsa` (without `.pub`) is the **private key**. You will have to keep this file private, ie: no one should have access to it (so don't copy it on removable media, share it with anyone, leave it in any unprotected place, etc). **Treat it as an actual key**; anyone with access to this private key, has access to your account, as if you had handed over your house key to others.

The file `id_ecdsa.pub` must be transferred to the remote host. For this we can use ssh-copy-id:

```
$ ssh-copy-id -i ~/.ssh/id_ecdsa.pub username@remote-host
```

This may produce the following message:

```
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/username/.ssh/id_ecdsa.pub"
```

```
The authenticity of host 'remote-host (123.123.123.123)' can't be
established.
ECDSA key fingerprint is SHA256:tygMarTe3SOjTcY9HzldKThxQzsTeiYHg5JmjB2bxeg.
Are you sure you want to continue connecting (yes/no)? yes
```

Having confirmed the access key to remote-host, the copy operation will commence:

```
/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
username@remote-host's password:
One-time password (OATH) for `username`:
```

Type your password (and the 2FA passcode) to actually start the file copy.

```
Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'username@remote-host'"
and check to make sure that only the key(s) you wanted were added.
```

The passwordless ssh login is now in place.

Alternative public key copy

It may be that ssh-copy-id is not available with your version of OpenSSH. In that case you need to copy the information yourself. This can be done following the below procedure.

It is possible to copy the public key directly to the remote host. The command below pipes the content of the public key through the ssh login to the remote host. On the remote host we first create the .ssh directory (if not there) and then append the public key content to the authorized_keys file, all in one command:

```
cat ~/.ssh/id_ecdsa.pub | ssh username@remote-host "mkdir -p ~/.ssh && chmod
700 ~/.ssh && cat >>  ~/.ssh/authorized_keys"
```

where username@remote-host can also me replaced by the logical name you have defined while configuring the ssh.

After successful execution of above command you can login to remote-host without specifying a password.