Write an automation that accepts username as a command line argument and enables SSH based remote access for that account on five different machines

Username input can be done via commandline as argument to the program, this time don't ask for the keys, just username.

Authorized_keys file can be used. One can create a pem file for the user for ssh access using ssh-keygen

Then ssh can be used like this by the user -

ssh –i <path to pem file> <username>@<Machine IP>

So for user to be able to connect the following must be available -

- A way to execute remote commands on the 5 machines

  (ssh can be used for remote command execution like ssh –i <path to pem file> <admin user>@<Machine IP> <command>)

  (The program should have a pem file of it's own so that the program can access the 5 machines.)

- All 5 machines must have a account with same username as entered via commandline

  (The program has to create one if it doesn't exist, and the program can use the useradd command on the remote machine using syntax in A)

- On each machine the authorized_keys file needs to be updated. so generate a pem file and public file for the user in the program itself using ssh-keygen, no need to ask user for it

(the program can download the authorized_keys , append it with the new key for the user, and update it on the remote machine, using syntax in point A)

After this the keys are setup in 5 machines for the user. Now he just has to do from his personal machine -

ssh –i <path to pem file> <username>@<Machine IP>

To generate an RSA key pair for version 2 of the SSH protocol, follow these steps:

1. Generate an RSA key pair by typing the following at a shell prompt:

```
$ ssh-keygen   or

$ ssh-keygen   -t  rsa   -b  2048  -v
```

Optional: To **increase the security of your key,** increase the size with the **–b** flag. The minimum value is **768 bytes** and the default, if you do not use the flag, is **2048 bytes. We recommend a 4096 byte key:**

- And when asked to enter file in which to save the key, type **linux_point** and when asked to **enter passphrase**, press **Enter** (empty passphrase) and confirm by

```
$ ls

    linux_point      linux_point.pub
```

- Here we will get two files generated, one will be my-certificate and one will be **pub**, rename the my-certificate to **linux_point.pem,** so you will have two files, **linux_point.pub** and **linux_point.pem**

```
    $ mv   linux_point     linux_point.pem
```

- Change the permissions of the **~/.ssh/** directory

```
$ chmod   700   ~/.ssh
```

- Create a file **~/.ssh/authorized_keys** **if already exist ignore this step**

```
$ vim    ~/.ssh/authorized_keys
```

- Changes are made in file **~/.ssh/authorized_keys** such as copy the pub in file **~/.ssh/authorized_keys** on the machine to which you want to connect, appending it to its end if the file already exists.
- And Change the permissions of the **~/.ssh/authorized_keys** file using the following command:

```
$ chmod   600  ~/.ssh/authorized_keys
```

Now **download** the **pem** file (**linux_point.pem) in your drive or system from where you want to Access the Server.**

References :

https://linuxaws.wordpress.com/

https://www.debian.org/