

[Show Submission Credentials](#)

P0. Secure Shell (SSH) Secure Shell (SSH)

✓ Secure Shell (SSH)

✓ Remote Screen Management

Secure Shell (SSH)

Secure Shell (SSH)

Secure Shell (SSH) is a network protocol used for secure data communication, remote login and remote command execution. You should be quite familiar with SSH already, but there are some quirks to using SSH with EC2 instances, which are covered below.

In order to get SSH working with an instance on EC2, you must ensure that the SSH clients and servers are installed correctly, that the instance's security group allows incoming connections on port 22, and that your authentication mechanism is working correctly.

```
$ ssh -i myfirstinstance.pem ubuntu@ec2-54-242-38-85.compute-1.amazonaws.com
The authenticity of host 'ec2-54-242-38-85.compute-1.amazonaws.com (54.242.38.85)' can't be established.
ECDSA key fingerprint is SHA256:1F1z2m30/R8KVaS9AMy9PzZQhLaQZv5w25CCkgYBw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-242-38-85.compute-1.amazonaws.com,54.242.38.85' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-1022-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun Jan  9 13:44:15 UTC 2022

System load:  0.61               Processes:    101
Usage of /:   18.2% of 7.69GB    Users logged in:  0
Memory usage: 20%              IPv4 address for eth0: 172.31.21.109
Swap usage:   0%

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-21-109:~$
```

Figure 1: Connecting to AWS EC2 Instances through SSH

Setting up SSHD (If not installed already)

All EC2 images come with an ssh daemon (`openssh`) pre-installed. In case you have a fresh linux install on your own machine, please follow the steps below to install and configure `openssh` :

```
sudo apt-get install openssh-server
```

Configuration settings for ssh are on `/etc/ssh/sshd.config`

Depending on your version of Ubuntu, sshd can be managed either using `init.d` :

```
sudo /etc/init.d/sshd
```

or by using service:

```
sudo service sshd [start|stop|restart]
```

Authentication Mechanisms

SSH supports a number of authentication mechanisms. You might be most familiar with password authentication (where you enter a username/password when connecting to a remote machine). Password authentication security is based on the length and complexity of the password, and is prone to brute-force or dictionary-based attacks. This is especially dangerous on a publicly-accessible EC2 instance.

To plug the vulnerability and to increase security, all EC2 instances are configured to use “key-based” authentication instead. This is based on public-key cryptography. A key pair consists of two keys: one private and one public. The public and private keys are mathematically

linked; they are generated using algorithms such as RSA or DSA. However, from the knowledge of only one of the keys (only private or only public), it is either impossible or very expensive to calculate the other key.

You can also generate your own key pair. In Linux, ssh key pairs are typically stored in the `~/.ssh` directory. We recommend using this directory as it has the correct permissions (read-write for user and no permissions for anyone else - `700`). The following command can be used to generate a key-pair in Linux using the `ssh-keygen` tool in standard OpenSSH installation (You are free to create an RSA key using any other tool):

```
# Generate a new key with your email id as a label
ssh-keygen -t rsa -b 4096 -C "email_id@domain.com"

# Enter the file where you want to save the key: (recommended - choose default)

# You will be asked enter a pass-phrase for your key twice.
# (Use a strong pass-phrase. Longer pass-phrases are more secure than shorter ones.)

# After you provide the pass-phrase. The console will print the location of the
key and the key fingerprint.
```

You can now import your key on Amazon EC2 and use this to create an SSH connection to a remote instance. You can see how to do this here (<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html#how-to-generate-your-own-key-and-import-it-to-aws>).

As an example, the following `ssh` command will execute the command `uname -a` on the remote instance `some-instance.ec2.amazonaws.com`. This is especially useful to run commands on many remote instances in a script.

```
ssh -i private_key_file.pem some-instance.ec2.amazonaws.com 'uname -a'
```

Information

Troubleshooting SSH Problems

If you receive a permission-denied error when trying to connect to an EC2 instance, verify that you are using the private key to connect. In addition, verify that your private key files have unix permissions `400` and the directory containing the key is `700`.

Running `ssh -v` will provide more verbose output, which may be useful for debugging connection issues.

Information

SSH Timeout

Sometimes you need to execute long-running scripts on your EC2 instance, however SSH terminal sessions will typically timeout after being inactive for a certain amount of time. If the sessions timeout, all your running scripts will be terminated. There are several

ways you could avoid SSH timeout:

1. Disable SSH timeout on your SSH Client
(<https://docs.oseems.com/general/application/ssh/disable-timeout>).
2. Use `nohup` to run your scripts.
3. Preserve your terminal sessions by using remote screen management (as explained in the next section).

Remote Screen Management

Remote Screen Management

When working with remote machines over SSH, the `bash` (or similar linux shells) are used. These shells work in interactive mode, allowing the user to run commands and launch processes. However, due to network connectivity issues, remote SSH shells can be volatile and `bash` by default will terminate all launched processes if the session is disconnected. This can be especially frustrating when running long jobs or scripts that get terminated due to a network issue. The `nohup` command is useful to run a program in the background and continue to run the command even if the session is disconnected.

Another useful tool to manage a remote SSH session is `byobu`. `byobu` is a multi-terminal manager application for Linux, similar to `screen` or `tmux`. We highly recommend using `byobu` to keep the state of your remote sessions alive, even if you disconnect from the SSH session. You can install `byobu` on your instance if it's not already present.

Launch `byobu` by running the command `byobu`. This will launch a new shell session in your instance. You should see a screen similar to the image below. You can create multiple shell sessions by pressing `F2`. To toggle between shell sessions, press `F3/F4`. To disconnect from `byobu` (and keep the shells alive), press `F6`. To terminate a shell session, simply exit the shell.

```

kirkland@x230 () - byobu
Alt-Left/Right      Move focus among windows
Alt-Up/Down         Move focus among sessions
Shift-Left/Right/Up/Down Move focus among splits
Shift-F3/F4         Move focus among splits
Ctrl-F3/F4          Move a split
Ctrl-Shift-F3/F4    Move a window
Shift-Alt-Left/Right/Up/Down Resize a split
F5                  Reload profile, refresh status
Alt-F5             Toggle UTF-8 support, refresh status
Shift-F5           Toggle through status lines
Ctrl-F5            Reconnect ssh/gpg/dbus sockets
Ctrl-Shift-F5      Change status bar's color randomly
F6                Detach session and then logout
Shift-F6           Detach session and do not logout
Alt-F6            Detach all clients but yourself
Ctrl-F6           Kill split in focus
F7               Enter scrollback history
Alt-PageUp/PageDown Enter and move through scrollback
Shift-F7          Save history to $BYOBU_RUN_DIR/printscreens

en
F8               Rename the current window
Ctrl-F8          Rename the current session
Shift-F8         Toggle through split arrangements
Alt-Shift-F8     Restore a split-pane layout
Ctrl-Shift-F8    Save the current split-pane layout
F9              Launch byobu-config window
Ctrl-F9          Enter command and run in all windows
Shift-F9         Enter command and run in all splits
F10             * Used by X11 *
F11             * Used by X11 *
Alt-F11         Expand split to a full window
Shift-F11       Zoom into a split, zoom out of a split
Ctrl-F11       Join window into a vertical split
F12            Escape sequence
Shift-F12      Toggle on/off Byobu's keybindings
Alt-F12        Toggle on/off Byobu's mouse support
Ctrl-Shift-F12 Mondrian squares

(END)

kirkland@x230:~$ # Byobu is great for disconnecting from sessions
kirkland@x230:~$ # and coming back later, perhaps over SSH
kirkland@x230:~$ # F6 will disconnect from the current session
kirkland@x230:~$ # while leaving all of your programs running
kirkland@x230:~$ # F6...
kirkland@x230:~$

top - 10:37:25 up 18:55, 2 users, load average: 0.13, 0.19, 0.21
Tasks: 250 total, 1 running, 249 sleeping, 0 stopped, 0 zombie
%Cpu(s): 4.8 us, 1.8 sy, 0.0 ni, 93.3 id, 0.0 wa, 0.0 hi, 0.0 st,
KiB Mem: 16122772 total, 1952948 used, 14169824 free, 86944 buffers
KiB Swap: 0 total, 0 used, 0 free, 1115156 cached

  PID USER   PR  NI    VIRT    RES    SHR   S  %CPU  MEM%   TIME+
15521 kirkland 20   0 163980 12104 10000  S   7.3   0.1   0:48.45
1426  root    20   0 336008 58504 46776  S   3.3   0.4   2:13.18
12577 kirkland 20   0 660876 20976 13576  S   2.0   0.1   0:13.70
2832  kirkland 20   0 1591140 106008 35324  S   1.7   0.7   3:45.57
2581  kirkland 20   0 362240 9020 2920  S   1.0   0.1   0:19.37
12611 kirkland 20   0 25128 2596 1272  S   1.0   0.0   0:06.11
2650  kirkland 20   0 205152 3304 2728  S   0.7   0.0   0:06.12
2873  kirkland 20   0 450956 29388 12100  S   0.7   0.2   0:08.35
7     root    20   0 0 0 0  S   0.3   0.0   0:04.28
2615  kirkland 20   0 22440 900 532  S   0.3   0.0   0:14.59
2616  kirkland 20   0 30784 700 392  S   0.3   0.0   0:03.17

```

Figure 2: Byobu Screenshot. source (<https://www.byobu.org>)

