EECS Tutorial: cslab Linux Environment SSH Access

FAQ for SSH access into cslab Linux environment

General Information regarding accessing cslab Linux environment using SSH

- **Please only connect into the cslab environment with an SSH client if you have previous experience in using SSH and the Linux command-line. If you are new to Linux, please use the cslab *Guacamole* web-browser interface.**
- SSH uses network port 22. If you are accessing cslab via SSH on WSU campus, ensure you are connected wirelessly to "WSU Secure" or using an Ethernet connected computer. "WSU Guest" wireless prohibits port 22 connections and your SSH session will fail to connect.
- You can only access cslab using SSH public key authentication. You cannot access cslab over SSH using your myWSU password.
- The cslab-nodes are located inside a virtual private network. These internal nodes are only SSH accessible via the `cslab-bastion.cs.wichita.edu` jumphost. Any external connection into the cslab environment must proxy connect through the `cslab-bastion`.
- `cslab-bastion.cs.wichita.edu` and `cslab-sftp.cs.wichita.edu` can be used as SCP or SFTP file-server hosts with SSH clients, such as *OpenSSH*, or graphical SFTP clients, such as *Filezilla*.
- This tutorial will help you configure your SSH client to make a proxy connection into cslab via `cslab-bastion` and to set up SSH public key authentication.
- **ONLY USE THE CSLAB-BASTION AS A PROXY/JUMPHOST INTO THE CSLAB-NODES OR AS AN SCP/SFTP SERVER. DO NOT DIRECTLY SSH INTO THE CSLAB-BASTION FOR OTHER TASKS!**

How do I access cslab Linux environment via *PuTTY* on Microsoft Windows?

Configuring the *PuTTY* SSH client for cslab access:

1. Download the full *PuTTY* package from Simon Tatham's official download page and install all *PuTTY* utilities on your local Windows computer. You cannot connect to cslab with just the *PuTTY* client.
2. Run the *PuTTY Key Generator*, which is listed in your start menu as *PuTTYgen*.
3. In the *PuTTY Key Generator* window, generate a new RSA key with 4096 bits. You may need to change the settings at bottom of window before clicking **Generate**.
4. Follow the prompts to create randomness and generate your RSA key.
5. Once your key has been generated, change the following fields:
   - Key comment: `your_mywsu_id@your_local_computer_name`
   - Key passphrase: `choose_passphrase_you_will_remember`
   - Confirm passphrase: `same_passphrase_as_above`

   It is highly recommended to use a passphrase for your RSA key to keep your Linux user account secure.
6. Select all the text displayed within the box titled *Public key for pasting into OpenSSH authorized_keys file*, starting with `ssh-rsa` and ending with name of your computer.
7. Copy the selected text either by right-clicking within the same box and selecting [copy] or by pressing the key combination **Ctrl+C**.
8. Open a web-browser application and follow the *eecs_tutorial_cslab_web_access* document to access cslab-gateway.cs.wichita.edu
9. Once logged into *guacamole*, open a [cslab_SSH_CLI_terminal] connection.
10. Within the cslab SSH terminal session in your browser, open the *Guacamole* menu sidebar by pressing the key combination **Ctrl+Alt+Shift**.
11. Paste the copied text to the remote *Guacamole* **Clipboard** field using your preferred method, i.e. **Ctrl+V**.
12. Close the *Guacamole* menu sidebar by pressing the key combination **Ctrl+Alt+Shift**.
13. Within the cslab SSH terminal session, open your `authorized_keys` file by typing `nano ~/.ssh/authorized_keys`

14. Paste your locally copied SSH public key into the terminal session by right-clicking on the browser window with your mouse or by pressing the key combination **Ctrl+Shift+V**.

15. Ensure you have a blank line at end of the text file by pressing **Enter** if required.

16. Quit *nano* by pressing **CTRL+X** and follow the prompts at the bottom of the screen to ensure you save the `authorized_keys` file.

17. Ensure correct permissions are set on the `authorized_keys` file by typing `chmod 600 ~/.ssh/authorized_keys`

18. NOTE: If you wish to set up more than one local computer with different SSH keys for accessing cslab, then you can append additional SSH public keys in your cslab user `authorized_keys` file. Make sure to remove no longer used SSH public keys from this file.

19. Once you are done with the above steps, make sure to properly disconnect and log out of the cslab *guacamole* web-interface.

20. Back in the *PuTTY Key Generator* window, click **Save private key**.

21. In the *Save private key* window, browse to a local directory where you wish to securely store your new SSH key and in the **File name** field enter the name `cslab_rsa`. Clicking on **Save** will store your SSH key to your local computer as the `cslab_rsa.ppk` *PuTTY* key file. **Keep this file safe and private!**

22. OPTIONAL: If you wish to also store your SSH public key as a simple text file for future reference, then click **Save public key**, browse to the same directory, and in the **File name** field enter the name `cslab_rsa_public_key.txt`

23. Once you are done with the above steps, exit/close the *PuTTY Key Generator* program.

24. Download the `cslab_putty_ssh_client_full_configuration` registry file from Ben Roose's GitHub tutorials repository.

25. Open this downloaded file from within your web-browser or by double-clicking on the file in your file explorer, and click **Yes** when asked if you wish to add this new information to your Windows registry. You will not be able to continue configuration of PuTTY until this file has been added into your Windows registry.

26. Run the *PuTTY* program and you should now see two new **Saved Sessions** entries in your *PuTTY* Configuration Window listed as *cslab access* and *cslab last node accessed*.

27. Click on *cslab access* and click on **Load**. DO NOT click **Open**.

28. Click on the *Connection–Data* Category tab on the left of the window and enter your myWSU ID in the **Auto-login username** field.

29. Click on the *Connection–Proxy* Category tab on the left of the window and enter your myWSU ID in the **Username** field.

30. Click on the *Session* Category tab on the left of the window and click on **Save**. DO NOT click **Open**.

31. Perform the last 4 steps again for the *cslab last node accessed* entry.

32. Once you are done with the above steps, exit/close the *PuTTY* program.

33. Congratulations! You have now configured your local Windows computer to directly access cslab via *PuTTY*. However, you will need to first add your new SSH key to the *Pageant (PuTTY Authentication agent)* prior to logging into cslab. Running *Pageant* and adding your SSH key will be explained in the next section!

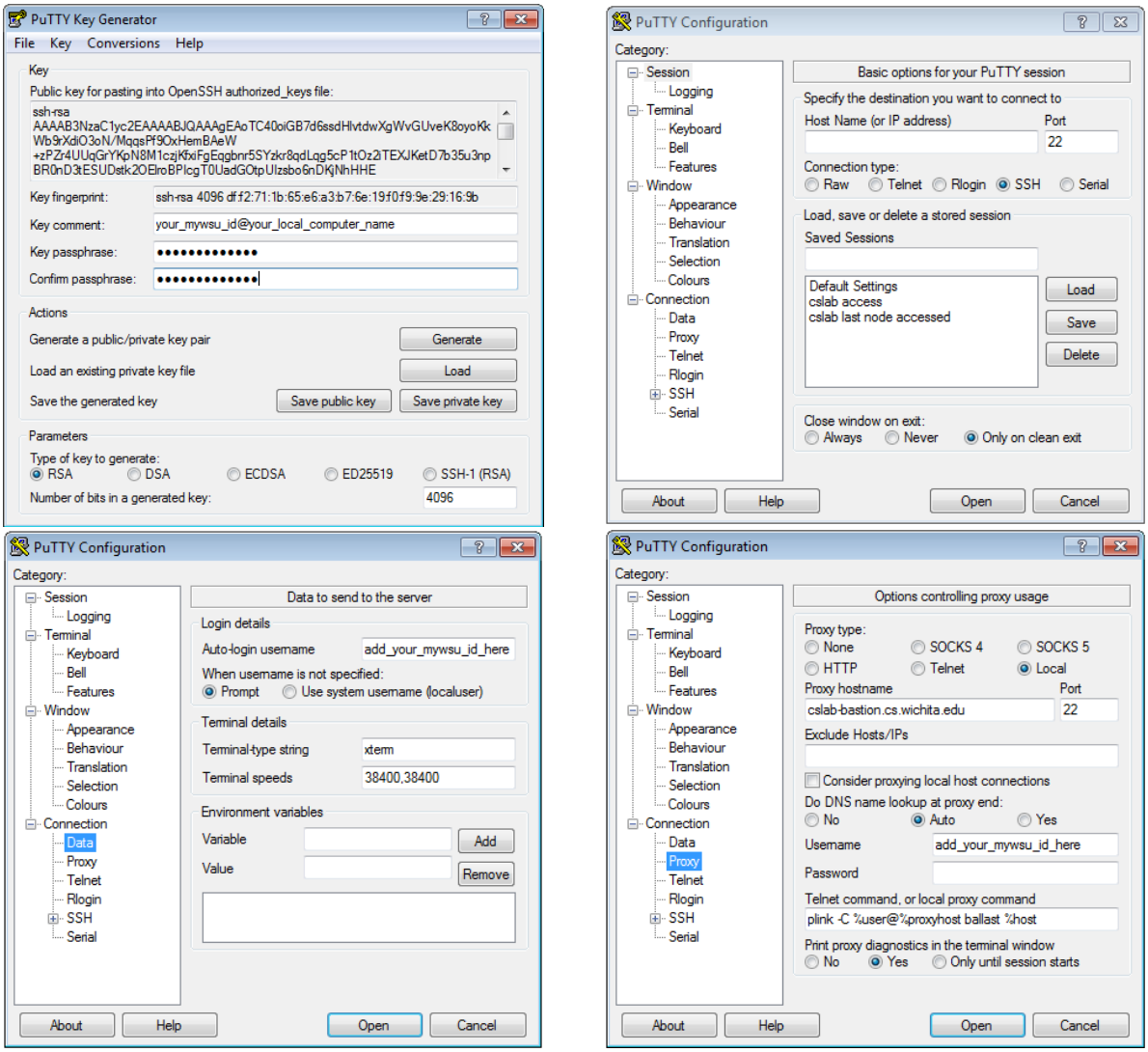

Figure 1: *PuTTY* screenshots for cslab configuration

Logging into cslab using the *PuTTY Pageant* client:

1. Run the *Pageant (PuTTY Authentication agent)*, listed in your start menu as *Pageant*.
2. On the far right of your Windows taskbar, you should see a new notification icon for the running *Pageant* program.
3. Right-click on the *Pageant* notification icon and select the menu option **Add key**.
4. In the *Select private Key File* window, browse to the local directory with your stored `cslab_rsa.ppk` file. Select the `cslab_rsa` key file and click on **Open**.
5. When prompted, enter the previously defined passphrase for your key.
6. OPTIONAL: if you wish to see the SSH keys already added into *Pageant*, right-click on the *Pageant* notification icon and select the menu option **View keys**.
7. Right-click on the *Pageant* notification icon and, under the **Saved Sessions** sub-menu, select **cslab remote access**.
8. If everything is correctly configured, a new *PuTTY* terminal emulator window will open and automatically log you into an available cslab node!
9. Each time you shutdown or reboot your computer, you will need to manually run *Pageant* and then add your SSH key before you can connect to the cslab environment.
10. OPTIONAL: You can have *Pageant* run automatically at startup by copying the *Pageant* Start menu shortcut file into your Windows **Startup** directory.
11. OPTIONAL: To have your cslab SSH key automatically added into *Pageant* when it runs, open the **Properties** box for the *Pageant* Start menu shortcut (or the **Startup** shortcut). In the **Target** field enter the full filepath of your `cslab_rsa.ppk` key file to the end of the `pageant.exe` line, i.e.

```
"C:\Program Files\PuTTY\pageant.exe" "C:\Users\localuser\cslab_rsa.ppk"
```

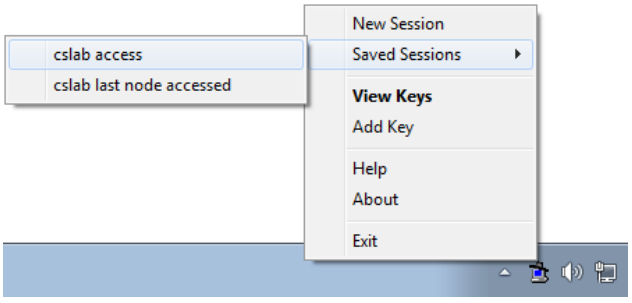

Figure 2: *Pageant* taskbar menu for cslab access

Logging into your last used cslab-node-# using the *PuTTY* client:

- When connecting to the cslab Linux environment using an SSH client, the ballast load-balancer will redirect you to one of the available and least used cslab-nodes at time of connection. Since load-balancing is calculated by ballast on a one minute cycle, you may not be redirected to the same cslab-node each time you connect.
- **Use the *cslab last node accessed* saved session to connect to the last used cslab-node only when you need access into a previously running SSH session. For normal use, the best option is to connect using the *cslab access* saved session and let the ballast load-balancer automatically connect to an available node.**

Running graphical (GUI) applications using the *PuTTY* client:

1. SSH allows for graphical applications to run on a local computer from the remote cslab Linux environment using X11 forwarding.
2. You will need to install the Xming software before using X11 forwarding with *PuTTY*. Download the Xming X Server for Windows and install the software package.
3. Run the *PuTTY* program, click on **cslab access** in Saved Sessions, and click on **Load**.
4. Click on the *Connection–SSH–X11* Category tab on the left of the window.
5. Turn on the **Enable X11 Forwarding** check box and ensure the **X display location** contains `localhost:0.0`. If you change the display location within *Xming Launch* settings, then also change the **X display location** in *PuTTY* to ensure consistency.
6. Click on the *Session* Category tab on the left of the window. Save this change as a different **Saved Session** by entering a new name for the session, such as `cslab with graphical access`, and click on **Save**.
7. Once you are done with the above steps, exit/close the *PuTTY* program.
8. To use cslab with X11 forwarding run *Xming* in the background first, then select your new **Saved Session** from the *Pageant* notification menu.
9. Once a connection into cslab has been established, you can type the name of a GUI application you wish to run, such as `geany`.

Why is *PuTTY* asking me to confirm the authenticity of cslab host?

- SSH host keys are essential to securing an SSH connection into a remote server. If you see an incorrect host key, then it may mean a cyber-attacker has attempted to compromise the remote server.
- The `cslab_putty_ssh_client_full_configuration` file from Ben Roose's GitHub tutorials repository will automatically add the cslab host keys into your Windows registry when you add this file during the initial configuration and set up of *PuTTY*. However, if your configuration lost the host keys for any reason, then it may ask you to confirm the authenticity of cslab hosts when you try to open an SSH connection.
- You can also download the `cslab_putty_ssh_client_host_keys_only` file from Ben Roose's GitHub tutorials repository and re-add the host keys into your Windows registry.
- Alternatively, you can manually add SSH host keys to your *PuTTY* Session configurations. When adding new host keys you should always ensure the host key fingerprint is correct. cslab-bastion and cslab host key fingerprints must match one of the following SHA256 or MD5 hashes before you add or accept the host key:

```
ECDSA key fingerprint is
SHA256:X6dBKj4sqYYPWol6MXSQvGhpIQ6qBxh7mBQhnSw8n64
MD5:d8:ba:c6:1c:86:fa:7f:f6:92:4f:c1:02:30:ce:ab:99

ED25519 key fingerprint is
SHA256:zzozIV7cP1T9C77PLRaevzdzCu21k44lbjd8jaJKS8Q
MD5:6d:3d:8e:3a:db:f6:de:33:af:77:01:40:f3:71:1d:14

RSA key fingerprint is
SHA256:0CUyGZAYMdOd8vTOK3AtM2XTX3lMaGA2NP73rR7s6Ns
MD5:75:5a:16:53:1a:7c:c2:4b:99:66:2d:e3:1e:76:f9:c9

DSA key fingerprint is
SHA256:7zW122xr+aoBb5yiRI96nvdx8Ml07qLKHYwG2Wu6jIM
MD5:27:59:53:18:5a:67:71:f6:32:f1:e1:15:e9:e5:fe:b1
```

How do I access cslab Linux environment via *OpenSSH* on Linux or Mac?

Configuring the *OpenSSH* client for cslab access:

1. On your Linux or Mac desktop/laptop computer, copy the following host entry into your local user ~/.ssh/config file:

```
Host cslab cslab-last cslab.cs.wichita.edu cslab-last.cs.wichita.edu
  ProxyCommand ssh your_mywsu_id@cslab-bastion.cs.wichita.edu ballast %h
  User your_mywsu_id
  IdentityFile ~/.ssh/cslab_rsa
  HostKeyAlias cslab.cs.wichita.edu

Host cslab-sftp cslab-sftp.cs.wichita.edu
  HostName cslab-sftp.cs.wichita.edu
  User your_mywsu_id
  IdentityFile ~/.ssh/cslab_rsa
  HostKeyAlias cslab.cs.wichita.edu
```

(Note: replace "your_mywsu_id" with your own 8 character ID number in both `ProxyCommand` and `User` lines.)

2. The above text is also available for download in the file `cslab_openssh_client_config` from Ben Roose's GitHub tutorials repository.

3. Open a command-line terminal emulator window and generate a new SSH public/private key pair for your local user by typing
   `ssh-keygen -t rsa -b 4096 -f ~/.ssh/cslab_rsa`

4. Follow the prompts in the command-line as your new SSH key is generated and enter a passphrase you will remember!
   It is highly recommended to use a passphrase for your SSH key to keep your Linux user account secure.

5. Open the newly generated SSH public key by typing
   `less ~/.ssh/cslab_rsa.pub`

6. Select all the text displayed within *less*, starting with `ssh-rsa` and ending with hostname of your computer.

7. Copy the selected text either by right-clicking on the terminal emulator window and selecting [copy] or by pressing the key combination **Ctrl+Shift+C**.

8. Open a web-browser application and follow the *eecs_tutorial_cslab_web_access* document to access cslab-gateway.cs.wichita.edu

9. Once logged into *guacamole*, open a [cslab_SSH_CLI_terminal] connection.

10. Within the cslab SSH terminal session in your browser, open the *Guacamole* menu sidebar by pressing the key combination **Ctrl+Alt+Shift**.

11. Paste the copied text to the remote *Guacamole* **Clipboard** field using your preferred method, i.e. **Ctrl+V**.

12. Close the *Guacamole* menu sidebar by pressing the key combination **Ctrl+Alt+Shift**.

13. Within the cslab SSH terminal session, open your `authorized_keys` file by typing `nano ~/.ssh/authorized_keys`

14. Paste your locally copied SSH public key into the terminal session by right-clicking on the browser window with your mouse or by pressing the key combination **Ctrl+Shift+V**.

15. Ensure you have a blank line at end of the text file by pressing **Enter** if required.

16. Quit *nano* by pressing **CTRL+X** and follow the prompts at the bottom of the screen to ensure you save the `authorized_keys` file.

17. Ensure correct permissions are set on the `authorized_keys` file by typing `chmod 600 ~/.ssh/authorized_keys`

18. NOTE: If you wish to set up more than one local computer with different SSH keys for accessing cslab, then you can append additional SSH public keys in your cslab user `authorized_keys` file. Make sure to remove no longer used SSH public keys from this file.

19. Congratulations! You have now configured your local Linux or Mac computer to directly access cslab via *OpenSSH*. Make sure to properly disconnect and log out of the cslab *guacamole* web-interface once you are done with the above steps.

Logging into cslab using the *OpenSSH* client:

1. Ensure you have first followed the directions in the previous section to configure your *OpenSSH* client.
2. In your local computer open a command-line terminal emulator window, such as *lxterminal* or *terminator*.
3. Start the ssh-agent in the background by typing `eval "$(ssh-agent -s)"`
4. Add your cslab_rsa private key to the ssh-agent by typing `ssh-add ~/.ssh/cslab_rsa`
5. When prompted, enter your previously created passphrase to unlock your SSH private key.
6. Connect to the cslab Linux environment by typing `ssh cslab`
7. The first time you connect to the cslab environment using SSH, you will be asked to confirm the authenticity of each remote host, i.e.

   ```
   The authenticity of host 'cslab.cs.wichita.edu' can't be established.
   ECDSA key fingerprint is [SHA256 or MD5 hash value].
   Are you sure you want to continue connecting (yes/no)?
   ```

8. Ensure the cslab-bastion and cslab host key fingerprints match one of the following SHA256 or MD5 hashes before typing `yes`:

   ```
   ECDSA key fingerprint is
   SHA256:X6dBKj4sqYYPWol6MXSQvGhpIQ6qBxh7mBQhnSw8n64
   MD5:d8:ba:c6:1c:86:fa:7f:f6:92:4f:c1:02:30:ce:ab:99

   ed25519 key fingerprint is
   SHA256:zzozIV7cP1T9C77PLRaevzdzCu21k44lbjd8jaJKS8Q
   MD5:6d:3d:8e:3a:db:f6:de:33:af:77:01:40:f3:71:1d:14

   RSA key fingerprint is
   SHA256:0CUyGZAYMdOd8vTOK3AtM2XTX3lMaGA2NP73rR7s6Ns
   MD5:75:5a:16:53:1a:7c:c2:4b:99:66:2d:e3:1e:76:f9:c9

   DSA key fingerprint is
   SHA256:7zW122xr+aoBb5yiRI96nvdx8Ml07qLKHYwG2Wu6jIM
   MD5:27:59:53:18:5a:67:71:f6:32:f1:e1:15:e9:e5:fe:b1
   ```

9. If the SSH connection completed successfully, then you will be presented with a standard shell prompt within a cslab node:
   `your_mywsu_id@cslab-node-#:~$`

10. NOTE: Mac OSX and many Linux distributions can run the ssh-agent and unlock SSH keys during user login using a *keychain* program. Check online for specific details on how to store SSH keys in your distribution. If not, then there are many script functions that can be added to your local user `.bashrc` file for running the ssh-agent at login, such as tstellanova's start_ssh_agent.sh on GitHubGist.

Logging into your last used cslab-node-# using the *OpenSSH* client:

- When connecting to the cslab Linux environment using `ssh cslab`, the ballast load-balancer will redirect you to one of the available and least used cslab-nodes at time of connection. Since load-balancing is calculated by ballast on a one minute cycle, you may not be redirected to the same cslab-node each time you connect.
- **Only connect to the last used cslab-node if you need access into a previously running SSH session. For normal use, best option is always let the ballast load-balancer automatically connect to an available node.**
- To connect to the last node you previously accessed using SSH type
  `ssh cslab-last`

Running graphical (GUI) applications using the *OpenSSH* client:

- SSH allows for graphical applications to run on a local computer from the remote cslab Linux environment using X11 forwarding.
- If you are using Mac OSX, then you may need to install the XQuartz software before using X11 forwarding. Download XQuartz for Mac and install the software package.
- To use X11 forwarding on a per session basis append the `-X` option flag to your SSH command:
  `ssh -X cslab`
- To always use X11 forwarding for connections into cslab, instead of using the `-X` option flag, add the following line to the `Host cslab cslab-last` entry in your local user `~/.ssh/config` file:
  `ForwardX11 yes`

Copying files to/from cslab via SCP:

- To copy a file from your local computer to your user home directory on cslab using Secure Copy (SCP), type
  `scp local_filename_or_path cslab-sftp:~`
- To copy a file from your user home directory on cslab to a local directory on your local computer, type
  `scp cslab-sftp:~/remote_filename_or_path local_directory`