Secure Shell (SSH)

- Users connect to a cluster by logging into the submit host (i.e., the login node) via the `ssh` command, which stands for Secure Shell (you will sometimes see it capitalized when not a command). SSH is an encrypted network communication protocol that allows network services, including remote login and remote command execution, to operate securely over an unsecured network.
- In SSH, all data is encrypted to protect it against spoofing, interception, and even intermediate data manipulation on the systems you connect through.
- The submit host (which is where you land when you `ssh` into one of our clusters) can be used for medium-performance tasks such as editing files or compiling smaller programs. Resource-demanding (i.e., high-performance) tasks must be submitted as jobs to the batch queuing system. The system automatically determines the appropriate allocation of computing resources for your task, which then runs on one or multiple compute nodes. Even a compile task that is expected to have a long runtime should be submitted as a job on a compute node instead of running it on the login node. Following these instructions is crucial; the performance of the login node is affected for all users when resource-intensive or long-running tasks are performed on the submit host.
- There are two main components in the SSH protocol: the client and the server. The SSH client is a program on your local computer that connects you to our remote systems, which are running the SSH server. In essence, the client requests a connection, and the server accepts or rejects the request.
- The request-acceptance process is facilitated through the use of public/private SSH key pairs. You can think of a user's public and private keys as corresponding to a username and password, respectively. For security, the user should never share their private keys, and should instead keep these stored on their local computer. As you will see below, when you create a key pair, you will be prompted to create a passphrase to protect your private key. In other words, you can think as your passphrase as being an extra barrier to your password.
- SSH key pairs serve as a means of identifying a user to an SSH server. These key pairs are created on your local computer, and can later be used to authenticate your login to our clusters. We ask for your public SSH key in the HPC account creation form because our SSH server uses it to verify your identity.

- Unix-based systems such as macOS and Linux distributions have at least the client version of OpenSSH pre-installed. If you have any of these operating systems, you can get started with generating a key pair right away. Users having Windows 10 or later can also take advantage of the pre-installed OpenSSH client. If you have earlier versions of Windows (or if you would like an alternative for OpenSSH if you have Windows 10 or later), you might find it easiest to download and install PuTTY, which is a free SSH client. We detail the SSH key pair creation process for all of these systems below.

SSH setup process for HPC access

These are the steps needed to gain access:

1. Creating a key pair consisting of a private and a public key
2. Adding a public key to your HPC account (the HPC team does this for you when you submit the HPC account creation form)
3. Adding your key to a keychain (if you have MacOS/Unix/Linux)
4. Setting up an agent for automatic login (optional)
5. Setting up authentication forwarding (optional but recommended)
6. Setting up a server alias (optional)

MacOS/Unix/Linux/Windows 10+

Creation of key pair using OpenSSH:

1. Open your computer's command line tool. For MacOS/Unix/Linux, this is Terminal. For Windows 10/Windows 11, this is Command Prompt.
2. Copy and paste the code below into your Terminal. You can substitute your own comment in the text after the -C tag to label this particular key pair (we recommend that you specify your email and the date for future reference). Hit Enter to proceed.

```
ssh-keygen -t rsa -b 4096 -C "email address and date"
```

3. You should see the output below. We suggest that you keep the default name and directory of your key pair file as they are (`/Users/<user>/.ssh/id_rsa`), so at this point you can press Enter. Otherwise, you can specify a different name and/or directory if you'd like.

```
Generating public/private rsa key pair.
Enter file in which to save the key (the default is
/Users/<user>/.ssh/id_rsa):
```

4. You will now be prompted to create and confirm a secure passphrase to protect your private key. **Remember: this passphrase is important, since it protects your private key and therefore your identity. Your passphrase should be strong (at least 12 characters; not all letters or numbers). Pick something that you will remember because there is no way to recover a key if you forget the passphrase!**

```
Enter passphrase: [Type a passphrase]
Enter same passphrase again: [Type passphrase again]
```

!!! types caution ""
Remember to always keep your private keys private! Share only public keys, never share your private key. DO NOT email us the fingerprint that your terminal will display to you.

5. Our HPC account creation form asks for your public key. To show the public key of the key pair you just generated, type in the following command and hit Enter:

```
cat ~/.ssh/id_rsa.pub
```

6. Copy the entire output (including the label, if you added one), and paste it into the corresponding field of the HPC account creation form.

Setting up agent, authentication forwarding, and server alias

Once the HPC team creates your account using the public key you provide us, you will be able to login to our servers using the passphrase you created. If you would like to avoid typing in your passphrase every time you connect to our servers, you can set up an agent to do this automatically for you. The process is slightly different between MacOS and Linux/Unix. For these systems, it is possible to set up authentication forwarding in the same step.

MacOS

For MacOS X 10.11 "El Capitan" and earlier, when you first access an encrypted SSH private key, the operating system will ask whether you want to save its passphrase in the Apple keychain. If you decide to set this up, your private key will be automatically

decrypted and available upon request (e.g., when logging into our servers). When this is set up correctly, running `ssh-add -L` in Terminal should show at least one key available on your local computer.

For MacOS 10.12 "Sierra" and later, the above does not apply. To add your passphrase to the Apple keychain, you have to edit your SSH config file. First, run `ssh-add --apple-use-keychain ~/.ssh/id_rsa`, assuming `~/.ssh/id_rsa` is where you saved your private key during the key pair creation process. For some versions of MacOS, you may need to run `ssh-add -K ~/.ssh/id_rsa`. You only need to do this once per key.

Once the Apple keychain has been successfully updated, it is then possible to set up authentication forwarding and a server alias in the same step. Run the command `vim ~/.ssh/config` in Terminal (or replace `vim` with your text editor of choice). Add the following to this file (we provide an example below this template):

```
Host <cluster-name alias>
    Hostname <cluster-name>.mskcc.org
    User <username>
    ForwardAgent yes
    AddKeysToAgent yes
    ServerAliveInterval 60
    UseKeychain yes
```

If you are having trouble setting up your Apple keychain, you can replace the last line with `IdentityFile ~/.ssh/id_rsa` (or wherever you saved your private key).

Example `~/.ssh/config` file:

```
Host lilac
    Hostname lilac.mskcc.org
    User joe7
    ForwardAgent yes
    AddKeysToAgent yes
    ServerAliveInterval 60
    UseKeychain yes
```

Using the above, the user `joe7` would be able to login to the `lilac` submit host just by running `ssh lilac` in the Terminal.

Linux/Unix

For these systems, you can disregard the aspects related to the Apple keychain above. Instead, run the command `ssh-add ~/.ssh/id_rsa` (or wherever you saved your key). Most Linux installations will automatically start `ssh-agent` when you log in. The rest of the process to set up the agent, authentication forwarding, and a server alias is the same as above.

Windows 8.1 and older

Creation of key pair using PuTTY

If you have Windows 8.1 or older, or if you have Windows 10/Windows 11 and would like an alternative to OpenSSH, you can follow the steps below.

1. Download and run the PuTTY installer
2. Launch PuTTYgen and click "Generate" to create a key pair
3. Click "Save private key", and save your new public/private key pair in a `.ppk` file in a safe directory (such as `C:\keys\YOURNAME.ppk`) for PuTTY to use later
4. Create a strong passphrase to encrypt your `.ppk` file. The same guidelines as for OpenSSH apply; namely, your passphrase should be at least 12 characters, not all numbers or letters, and should be strong but easy for you to remember, because it cannot be retrieved if forgotten.
5. Select the complete contents of the text area at the top of the window (which should start with "ssh-rsa"). Make sure you copy the entire public key, which is likely to extend below the bottom of the text area. Paste the public key into the HPC account request form.
6. Create a new configuration in PuTTY and save it. This new configuration should include username@hostname in the server field (e.g., `joe7@lilac.mskcc.org`).

Setting up the SSH agent

Once the system administrators have created your HPC account, you will be able to connect to our servers. You will now have a couple different options to do so: either by entering your passphrase each time you want to log in, or by caching your private key to connect automatically. If you find yourself entering your encryption passphrase too often, you can use the latter method.

Method 1: Logging in manually

Select "SSH:Auth" from the left-side menu and specify your `.ppk` file in the bottom right. Scroll back to "Session" at the top of the left-side menu and save your configuration. Once

the system administrators have created your HPC account, you will be able to connect to our servers by double-clicking this new configuration and entering the passphrase you used to encrypt your `.ppk` file.

Method 2: Using an agent to remember your passphrase

Make sure that the `.ppk` file type is assigned to open with Pageant (the "agent") in Windows. Remove any `.ppk` keys from SSH:Auth in PuTTY for all configurations to force the software to use Pageant for authentication. To load your key into Pageant when you log in to Windows, place a shortcut to your key inside your local Startup folder (All Programs: Startup (right-click):Open).

Authentication forwarding

To set up authentication forwarding using PuTTY's Pageant service:

1. Open PuTTY
2. Load the PuTTY session (the appropriate configuration)
3. Check the authentication forwarding box in Putty:SSH:Auth
4. Save the PuTTY session

This should now allow you to `ssh` into a second server once you have already authenticated your identity on a different server.

X11-forwarding for applications with graphical interfaces

When using applications with graphical interfaces, it is sometimes necessary to enable X11-forwarding. You can enable X11-forwarding by using the `-Y` flag during your login process:

```
ssh -Y <alias>
```

For example (once a server alias has been set up, as detailed above):

```
ssh -Y lilac
```

You can test whether the forwarding process was successful by running the command `xterm` on the login node, which should open a new window. Keep in mind that your local

operating system must have an X server running; e.g., Xming on Windows or XQuartz on MacOS.