# Accessing the Prince cluster

<table>
<thead>
<tr>
<th>Quick Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPC Home</td>
</tr>
<tr>
<td>Getting an account</td>
</tr>
<tr>
<td>Getting started on Prince</td>
</tr>
<tr>
<td>Prince How-to Articles</td>
</tr>
<tr>
<td>Logging in</td>
</tr>
<tr>
<td>Windows</td>
</tr>
<tr>
<td>Mac / Linux</td>
</tr>
<tr>
<td>Clusters and Storage</td>
</tr>
<tr>
<td>Prince (HPC)</td>
</tr>
<tr>
<td>Dumbo (Hadoop)</td>
</tr>
<tr>
<td>Brooklyn (OpenStack)</td>
</tr>
<tr>
<td>Dalmatia (NYU Abu Dhabi)</td>
</tr>
</tbody>
</table>
Transferri ng data to/from the clusters

Transferri ng data to/from Prince cluster using Globus

Submitting jobs with sbatch

Available software

Licensed software available on the HPC Cluster

Software Tutorial

Building

Slurm

Cleaning

FAQs

Tutorials

Programming for Biologists

Acknowledgment

Statement

Research

Gallery

People

HPC

Cleanup

Scratch

Area

Clean up

Area

Clean-up

Cleanup

Program
Running jobs on the Prince Cluster

Accessing the Prince Cluster

From Windows workstation

From Mac workstation

Software and Environment Module

Job script and resource request

Introduction to job scheduling

Submitting jobs with sbatch

Requesting resources

Using computing nodes interactively

Monitoring batch jobs

Monitoring batch jobs - squeue
The HPC clusters cannot be accessed directly from the Internet - if you are outside of the NYU network, access is a two-step process:

1. Log in to `hpc2.nyu.edu`
   This is a "bastion host", providing a secure gateway between the NYU HPC clusters and the Internet. You will find that on `hpc2.nyu.edu` itself you can do very little save but to log in to an NYU HPC cluster

2. Log in to the cluster you wish to use
   This will put you on a "login" node for that cluster, from where you can manipulate files and initiate jobs. This is not the place for heavy compute work: you submit that as a job to the compute nodes via the batch queuing system. Instructions for using the batch system can be found here (but you don't need to worry about that yet).

**What is a SSH tunnel**

In computer networking, a computer decides what to do with an incoming network packet according to the "port" it arrived on. The port is simply a number attached to the packet. Certain ports are reserved for specific functions, for example packets arriving on port 22 are assumed to be intended for the SSH handler, so the computer passes those packets to SSH to interpret. Other port numbers are available to use for whatever you like, and as long as the same port is not used for different things on the same computer, everything works.

With SSH Tunneling, you will start an SSH session between your workstation and the bastion host `hpc2.nyu.edu`, and instruct that session to create a tunnel. Your workstation will make one end of the tunnel, at "localhost, port 8026" ("localhost" is the computer's name for itself, so packets arriving at your workstation port 8026 will be sent into the tunnel). The bastion host will make the other end of the tunnel, at "prince.hpc.nyu.edu, port 22", so anything coming through the tunnel will be forwarded to the normal SSH port (22) of Prince. The fact that your workstation cannot see Prince does not matter, it only needs to see its end of the tunnel.

The following diagram illustrates the process. It looks complex, but only requires 2 steps: the blue text shows what happens when you create the tunnel (step 1) and the green arrows indicate using the tunnel (step 2).
You only need to do step 1 once, and then you can use the tunnel (step 2) as many times as you like - for example, you might have two terminal sessions and a WinSCP session all using the same tunnel created with step 1.

In these instructions we are using port 8026. If it happens that another program on your computer is watching this port (which is fairly unlikely) then it won’t work, and you’ll need to choose a different port number, eg 9020, and substitute that throughout these instructions. 4-digit numbers starting with an 8 or a 9 are usually good ones to choose.