XSEDE
Extreme Science and Engineering Discovery Environment

Anvil 101

New User Tutorial

FORGING THE FUTURE OF COMPUTING

The Anvil Team, Purdue Research Computing

NSF

PURDUE UNIVERSITY | Information Technology

XSEDE
Extreme Science and Engineering Discovery Environment

Code of Conduct

This external code of conduct for XSEDE-sponsored events represents XSEDE's commitment to providing an inclusive and harassment-free environment in all interactions regardless of race, age, ethnicity, national origin, language, gender, gender identity, sexual orientation, disability, physical appearance, political views, military service, health status, or religion. The code of conduct below extends to all XSEDE-sponsored events, services, and interactions.

Webpage: www.xsede.org/codeofconduct

XSEDE ombudspersons:

Linda Akli, Southeastern Universities Research Association (akli@sura.org)

Lizanne Destefano, Georgia Tech (lizanne.destefano@ceismc.gatech.edu)

Ken Hackworth, Pittsburgh Supercomputing Center (hackworth@psc.edu)

Bryan Snead, Texas Advanced Computing Center (jbsnead@tacc.utexas.edu)

PURDUE UNIVERSITY | Information Technology

Terminology Statement

Words matter! XSEDE's commitment to fostering and promoting an inclusive environment for all users, staff, and the wider community extends to all language and terminology in all of our materials. As a result of this commitment, XSEDE's Terminology Task Force (TTF) was formed to review, address, and define processes to eliminate offensive terms in our materials.

Webpage: www.xsede.org/terminology

XSEDE TTF email address: terminology@xsede.org

XSEDE
Extreme Science and Engineering Discovery Environment

Acknowledgement

“This material is based upon work supported by the National Science Foundation under Grant No. 2005632.”

Disclaimer: “Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.”

PURDUE UNIVERSITY | Information Technology

Full Agenda

- Anvil system architecture including node types, storage, interconnects, and networking.
- Getting started with accounts and allocations
- Compilation and programing environment on Anvil
- Running Jobs on Anvil
- Data management and transfer on Anvil
- Q&A

Agenda

1. Anvil overview

- Introduction to Anvil
- Hardware
- Anvil group and consulting

XSEDE
Extreme Science and Engineering Discovery Environment

About Anvil

- Category I: A national composable advanced computational resource for the future of science and engineering

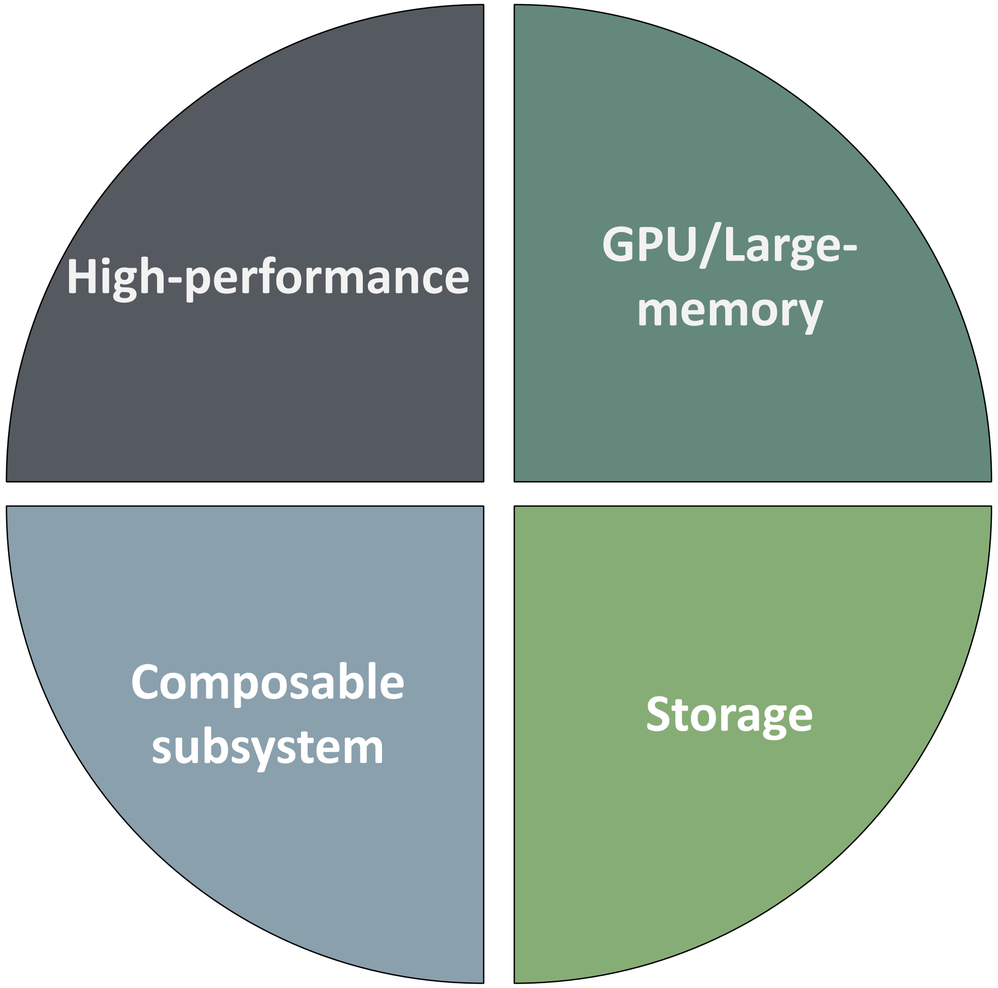
- By the Purdue research computing team. Full access starts February, 2022.

- NSF award #2005632; 5 years of operations; allocated via NSF XSEDE

PURDUE UNIVERSITY | Information Technology

System Resources

- **1000** compute nodes
- **128** core AMD 3rd Gen EPYC 7763 processors
- **5.3** PF peak performance



- **16** nodes with **4** NVIDIA A100 GPUs each
- **32** large memory nodes with **1** TB of RAM

- **8** large memory & storage nodes
- Kubernetes – Rancher for DevOps

- Multi-tier storage (including object storage)
- **10** PB of parallel filesystem, and **3** PB of all-flash storage
- Globus data transfer

Service & Support

Quick turnaround via XSEDE support tickets

[portal.xsede.org/help-desk]

Support team

[comprising domain experts from multiple disciplines]

Advanced user support

[data science consulting, HPC performance optimization, science gateway development]

Multimodal Training Delivery

[live lessons, online tutorials, video lessons]

Agenda

2. Getting started

- Get anvil account and allocation
- Logging in
- Check account usage

Agenda

2. Getting started

- **Get anvil account and allocation**
- Logging in
- Check account usage

Obtaining an Account

As an XSEDE computing resource, Anvil is accessible to XSEDE users who are given an allocation on the system. To obtain an account, users may submit a proposal through the **XSEDE Allocation Request System:** https://portal.xsede.org/allocations/announcements

Anvil is available for allocation now

Submission Period	Meeting Date	Users Notified	Allocation Begins
Dec 15 thru Jan 15	Early March	March 15	April 1
Mar 15 thru Apr 15	Early June	June 15	Jul 1
Jun 15 thru Jul 15	Late Aug/Early Sep	Sept 15	Oct 1
Sep 15 thru Oct 15	Early Dec	Dec 15	Jan 1

Agenda

2. Getting started

- Get anvil account and allocation
- **Logging in**
- Check account usage

Example: Logging in via SSO Hub

Anvil can be accessed via the **XSEDE Single Sign-On (SSO) hub:** https://portal.xsede.org/single-sign-on-hub

```
[localhost]$ ssh -l XUPusername login.xsede.org
login as: XUPusername
Using keyboard-interactive authentication.
Please login to this system using your XSEDE username and password:
Duo two-factor login for XUPusername

Enter a passcode or select one of the following options:

 1. Duo Push to XXX-XXX-XXXX
 2. Phone call to XXX-XXX-XXXX

Passcode or option (1-2): 1
Success. Logging you in...
#  Welcome to the XSEDE Single Sign-On (SSO) Hub!
#  ...
```

Use your SSH client to start an SSH session on login.xsede.org with your *XSEDE User Portal* **username** and **password**.

XSEDE requires the XSEDE Duo service for additional authentication.

ewa@ssohub:~ ⌥⌘1

```
Last login: Thu Jan 27 13:58:48 on console

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
dyn-nat-10-162-17-209:~ ewa$ ssh -l ewa login.xsede.org
(ewa@login.xsede.org) Please login to this system using your XSEDE username and
password:
password:
(ewa@login.xsede.org) Duo two-factor login for ewa

Enter a passcode or select one of the following options:

 1. Duo Push to XXX-XXX-7181
 2. Phone call to XXX-XXX-7181

Passcode or option (1-2): 1
Success. Logging you in...
Last login: Fri Jan 14 14:19:41 2022 from 128.210.106.177

#  Welcome to the XSEDE Single Sign-On (SSO) Hub!
```

Example: Logging in via SSO Hub

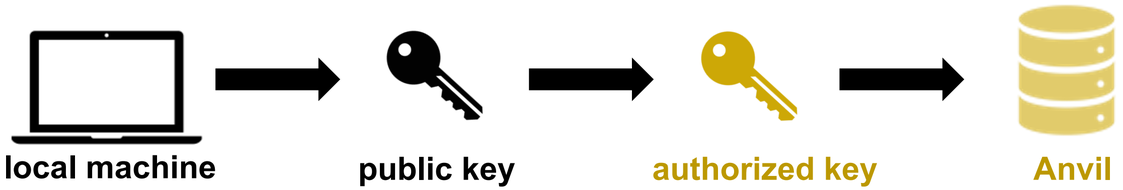
Once logged into the hub, then use the ***gsissh*** utility to login to Anvil.

```
[XUPusername@ssohub ~]$ gsissh anvil
================================================
==                  Welcome to the Anvil Cluster                  ==
...
[x-anvilusername@login01:~]$ whoami   # print effective user id
x-anvilusername
```

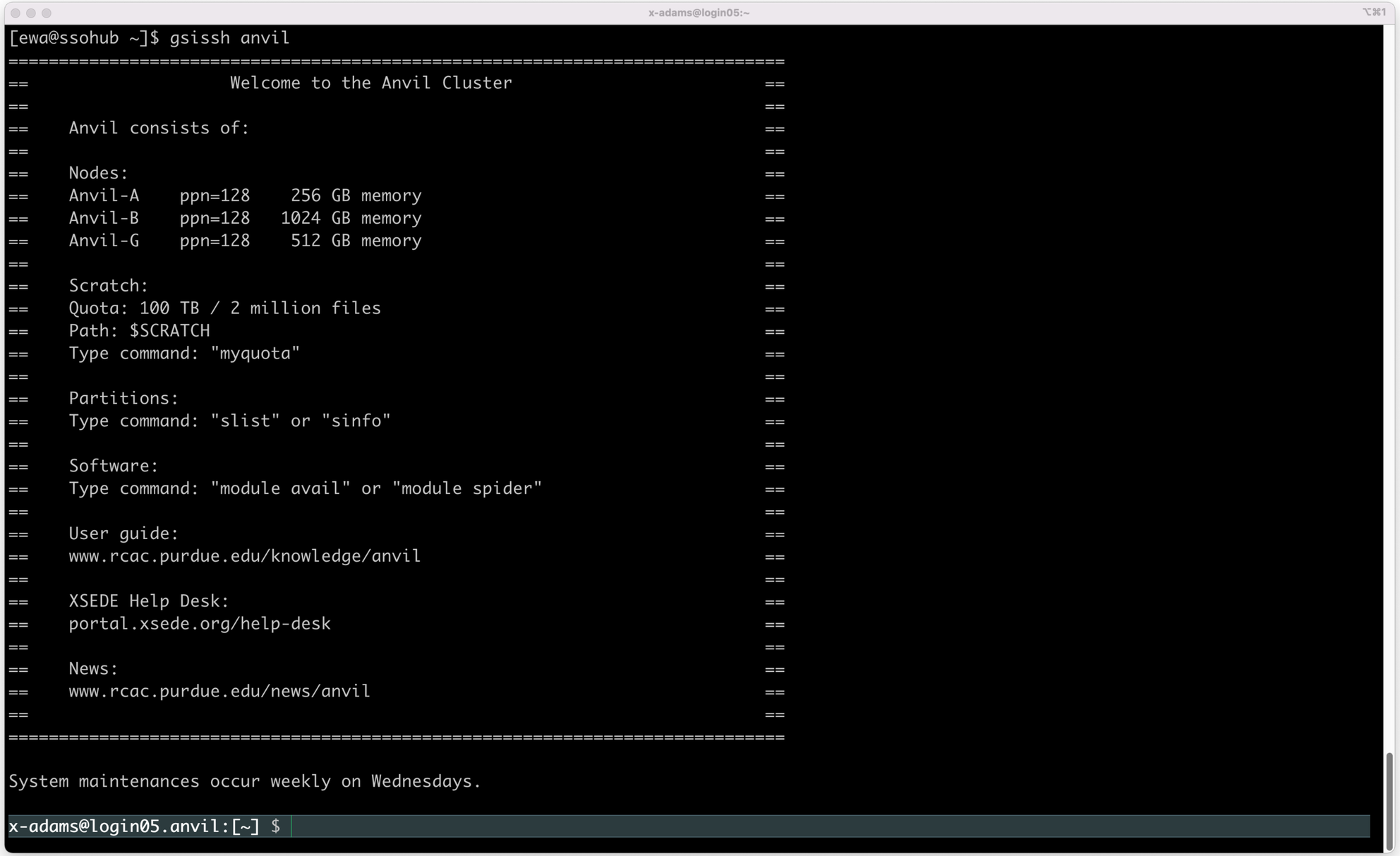
When reporting a problem to help desk, please execute ***gsissh -vvv*** and include the verbose output in your problem description.

SSH keys is also a good way to connect to Anvil.

Please see Appendix or **Anvil user guide** (www.rcac.purdue.edu/knowledge/anvil/access/login/sshkeys) for more detail.



XSEDE
Extreme Science and Engineering Discovery Environment

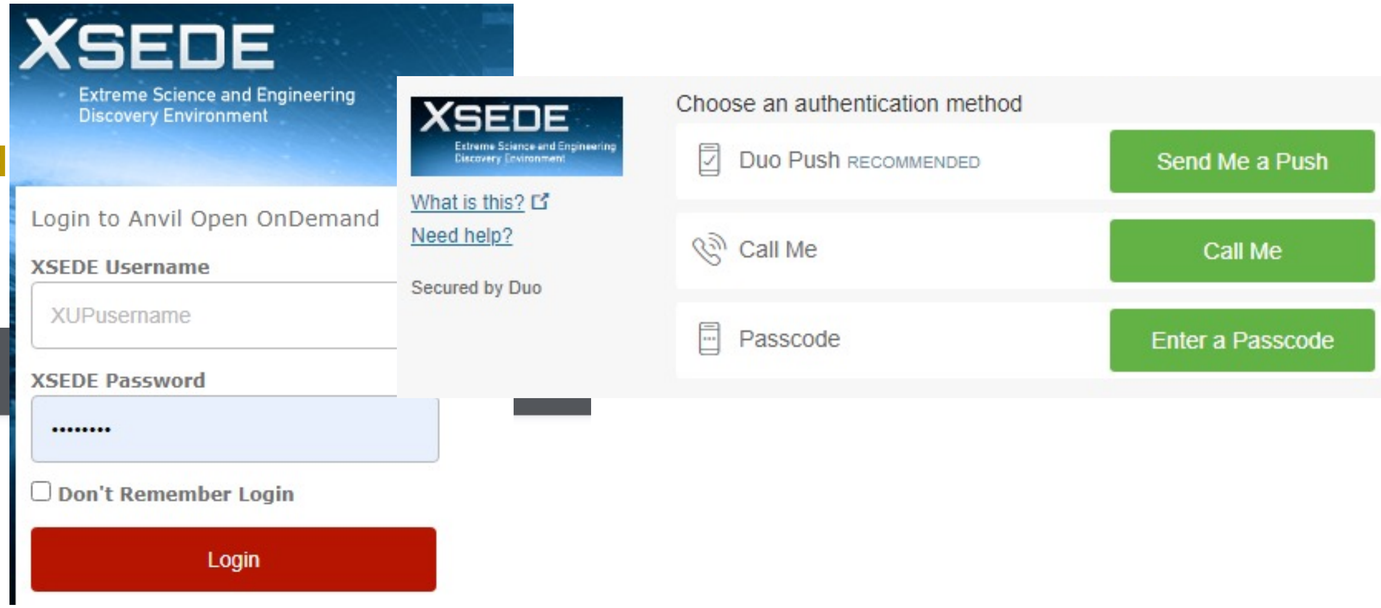


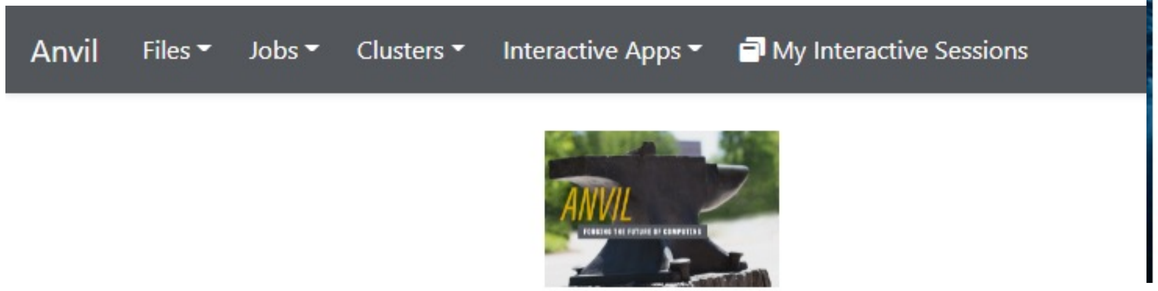
Open OnDemand

Open OnDemand allows one to interact with HPC resources through a web browser and easily manage files, submit jobs, and interact with graphical applications directly in a browser, all with no software to install.

Navigate to ***https://ondemand.anvil.rcac.purdue.edu***

Log in using your **XSEDE portal username and password**





More training section about Open OnDemand will be given by Anvil team in the future.

Dashboard - Anvil

ondemand.anvil.rcac.purdue.edu/pun/sys/dashboard

Apps | Bookmarks | Bookmark Bar | Powerlifting | Instantly Restart... | RCAC | https://www.scho... | Keto | Outriders | CyberAmbassadors | ACM TAPS Author... | The 100-year capi... | My Interactive Ses... | Other Bookmarks | Reading List

Anvil | Files | Jobs | Clusters | Interactive Apps | My Interactive Sessions | Help | Logged in as x-adams | Log Out

OnDemand provides an integrated, single access point for all of your HPC resources.

Message of the Day

powered by OPEN OnDemand

OnDemand version: v2.0.13

Agenda

2. Getting started

- Get anvil account and allocation
- Logging in
- **Check account usage**

Check Allocation Usage

To keep track of the usage of the allocation by your project team, you can use ***mybalance***:

```
[x-anvilusername@login01:~]$ mybalance
```

Allocation Account	Type	SU Limit	SU Usage (account)	SU Usage (user)	SU Balance
xxxxxx-cpu	CPU	1000.0	95.7	3.0	904.3
xxxxxx-gpu	GPU	1000.0	43.5	1.5	956.5

You can also check the allocation usage through **XSEDE User Portal**: https://portal.xsede.org/allocations/managing

You should see at least one allocation.

CPU and GPU nodes use are count separately, so there are using different allocation accounts.

Check Allocation Usage

x-adams@login05:~

```
x-adams@login05.anvil:[~] $ mybalance

Allocation      Type    SU Limit    SU Usage   SU Usage  SU Balance
Account                             (account)    (user)
=============  ====  ==========  ========== ==========  ==========
asc170016       CPU     50000.0     12645.4        1.4     37354.6
asc170016-gpu   GPU      2500.0        25.4        0.0      2474.6
tra220012       CPU      2000.0         0.0        0.0      2000.0

x-adams@login05.anvil:[~] $
```

Agenda

3. Compilation and programing environment

- Module system
- Provide software and software installation policy
- Compiling source code (examples and explanation)

Agenda

3. Compilation and programing environment

- **Module system**
- Provide software and software installation policy
- Compiling source code (examples and explanation)

Modules

- Module system provides for the dynamic modification of a user's environment.
- Module commands allow you to add applications and libraries to your environment.
- This allows us to simultaneously and safely provide several versions of the same software.
- Anvil team makes recommendations for both CPU and GPU stack regarding the CUDA version, compiler, math library, and MPI library. If you have no specific requirements, you can simply load the recommended set by:

```
$ module load modtree/cpu # for CPU
$ module load modtree/gpu # for GPU
```

Modules

- Lmod is a hierarchical module system, a module can only be loaded after loading the necessary compilers and MPI libraries that it depends on. A list of all available modules can be found by:

```
$ module spider
```

- The module spider command can also be used to search for specific module names.

```
$ module spider intel # all modules with names containing 'intel'
```

- To unload a module

```
$ module unload mymodulename
```

Modules

- To unload all loaded modules and reset everything to original state.

```
$ module purge
```

- To see all available modules that are compatible with current loaded modules

```
$ module avail
```

- To display information about a specified module, including environment changes, dependencies, software version and path.

```
$ module show mymodulename
```

- Show all modules currently loaded in my environment:

```
$ module list
```

Example: Modules

$ module list **# Show all modules currently loaded in my environment**

Currently Loaded Modules:

This default environment can be loaded by *$ module load modtree/cpu*

1) gmp/6.2.1 2) mpfr/4.0.2 3) mpc/1.1.0 4) zlib/1.2.11 5) gcc/11.2.0 6) libfabric/1.12.0 7) numactl/2.0.14 8) openmpi/4.0.6 9) modtree/cpu

$ module purge **# To unload all loaded modules and reset everything to original state**

$ module list

No modules loaded

Example: Modules

```
$ module load modtree/cpu          # To load the default CPU environment recommended by the Anvil team

$ module list

Currently Loaded Modules:

  1) gmp/6.2.1   2) mpfr/4.0.2   3) mpc/1.1.0   4) zlib/1.2.11   5) gcc/11.2.0   6) libfabric/1.12.0   7) numactl/2.0.14   8) openmpi/4.0.6   9) modtree/cpu

$ module unload openmpi/4.0.6      # To unload the openmpi/4.0.6  module

$ module list

Currently Loaded Modules:

  1) gmp/6.2.1   2) mpfr/4.0.2   3) mpc/1.1.0   4) zlib/1.2.11   5) gcc/11.2.0   6) modtree/cpu
```

When unload openmpi module, two more dependent modules are removed.

Example: Modules

```
$ module spider openmpi                  # Report all the versions for the modules that match “openmpi ”
----------------------------------------------------------------------------------------------------------------------------------------------------------
openmpi:
----------------------------------------------------------------------------------------------------------------------------------------------------------
Versions:
       openmpi/3.1.6
       openmpi/4.0.6 …


$ module spider openmpi/4.0.6          # Report detailed information on a particular module version openmpi/4.0.6
----------------------------------------------------------------------------------------------------------------------------------------------------------
openmpi: openmpi/4.0.6
----------------------------------------------------------------------------------------------------------------------------------------------------------
  You will need to load all module(s) on any one of the lines below before the "openmpi/4.0.6" module is available to load.
    aocc/3.1.0      gcc/10.2.0      gcc/11.2.0      gcc/8.4.1      intel/19.0.5.281
  Help:
    An open source Message Passing Interface implementation. The Open MPI Project is an open source Message Passing
Interface implementation that
    is developed and maintained by a consortium of academic, research, and industry partners. Open MPI is therefore able to
combine the expertise …
```

Agenda

3. Compilation and programing environment

- Module system
- **Provide software and software installation policy**
- Compiling source code (examples and explanation)

Provide Software

Programming Libraries & Compilers

- Various popular programming languages, GNU, Intel and AOCC compilers, message passing libraries
- Workflow, data management and analysis tools
- Debugging and profiling tools

Scientific Applications

- General purpose mathematics and statistics modeling tools, visualization tools
- Broad application base with installs and modules from various science and engineering domains

Containers and Datasets

- Support for Singularity containerization and execution (e.g. NGC, BioContainers)
- Efficient access to various databases (e.g., NCBI)

Need additional software? Please see the **Software Installation Request Policy**.

Agenda

3. Compilation and programing environment

- Module system
- Provide software and software installation policy
- **Compiling source code (examples and explanation)**

Supported Compilers

CPU nodes

Compilers: GNU, Intel, AOCC (AMD)

MPI implementations: OpenMPI, Intel MPI (IMPI) and MVAPICH2

All compilers installed on Anvil include OpenMP functionality for C, C++, and Fortran

GPU nodes

- The GPU nodes on Anvil support CUDA and OpenCL
- **OpenACC** functionality are support by:
  - **PGI** compilers through the ***nvhpc*** modules
  - **GNU** compiler through ***gcc/11.2.0-openacc*** module
- Some GPU codes may require compiled on the GPU nodes through an interactive session.

Example: Compiling Serial C++ Code

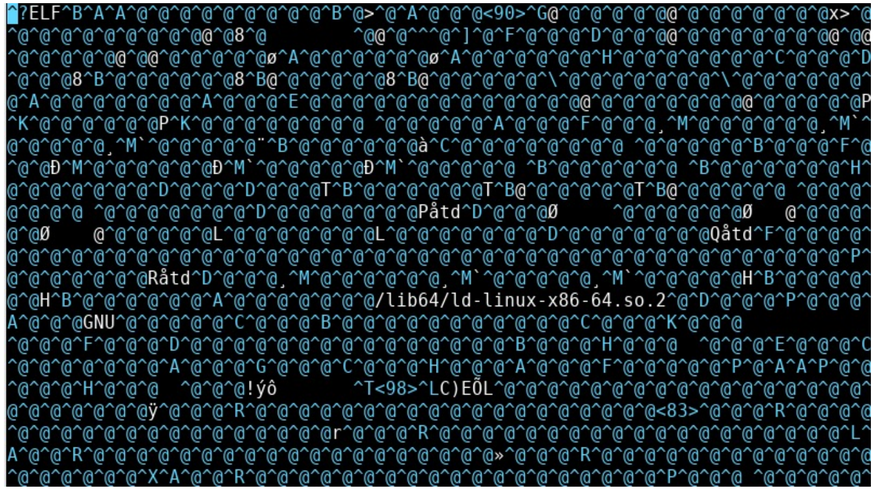
```cpp
//  C++

#include <iostream>
#include <unistd.h>

using namespace std;

int main () {
    int len=30;
    char name[30];
    gethostname(name,len);      // get run-host name
    cout << "Runhost:" << name << "    hello, world\n" << endl;
    return 0;
}
```

Source code: serial_hello.cpp



Machine code: serial

```
$ module list
Currently Loaded Modules:
  1) gmp/6.2.1   2) mpfr/4.0.2   3) mpc/1.1.0   4) zlib/1.2.11   5) gcc/11.2.0   6) libfabric/1.12.0   7) numactl/2.0.14   8) openmpi/4.0.6   9) modtree/cpu

$ g++ serial_hello.cpp -o serial          # Complie the c++ code with GNU compiler

$ ls
serial_hello.cpp    serial                # Executable files generated

$ ./serial

Runhost:a600.anvil.rcac.purdue.edu   hello, world
```

Compiling NVIDIA GPU Programs

Both login and GPU-enabled compute nodes have the CUDA tools and libraries for compiling CUDA programs.

But if code require CUDA drive, you need to submit an interactive job to get to the GPU nodes. The ***gpu-debug*** queue is ideal for this case.

```
$ module load modtree/gpu

$ nvcc gpu_hello.cu -o gpu_hello

./gpu_hello

No GPU specified, using first GPUhello, world
```

Compiling Serial Programs

The following table illustrates how to compile your serial program:			
Language	**Intel Compiler**	**GNU Compiler**	**AOCC Compiler**
Fortran 77	$ ifort myprogram.f -o myprogram	$ gfortran myprogram.f -o myprogram	$ flang program.f -o program
Fortran 90	$ ifort myprogram.f90 -o myprogram	$ gfortran myprogram.f90 -o myprogram	$ flang program.f90 -o program
Fortran 95	$ ifort myprogram.f90 -o myprogram	$ gfortran myprogram.f95 -o myprogram	$ flang program.f90 -o program
C	$ icc myprogram.c -o myprogram	$ gcc myprogram.c -o myprogram	$ clang program.c -o program
C++	$ icc myprogram.cpp -o myprogram	$ g++ myprogram.cpp -o myprogram	$ clang++ program.C -o program

*Intel compiler does not recognize the suffix ".f95". You may use ".f90" to stand for any Fortran code regardless of version as it is a free-formatted form

Compiling MPI Programs

The following table illustrates how to compile your MPI program. Any compiler flags accepted by Intel ifort/icc compilers are compatible with their respective MPI compiler.		
Language	**Intel Compiler with Intel MPI (IMPI)**	**Intel/GNU/AOCC Compiler with OpenMPI/MVAPICH2**
Fortran 77	$ mpiifort myprogram.f -o myprogram	$ mpif77 myprogram.f -o myprogram
Fortran 90	$ mpiifort myprogram.f90 -o myprogram	$ mpif90 myprogram.f90 -o myprogram
Fortran 95	$ mpiifort myprogram.f90 -o myprogram	$ mpif90 myprogram.f90 -o myprogram
C	$ mpiicc myprogram.c -o myprogram	$ mpicc myprogram.c -o myprogram
C++	$ mpiicc myprogram.C -o myprogram	$ mpicxx myprogram.C -o myprogram

*Intel compiler does not recognize the suffix ".f95". You may use ".f90" to stand for any Fortran code regardless of version as it is a free-formatted form

Compiling OpenMP Programs

The following table illustrates how to compile your shared-memory program. Any compiler flags accepted by Intel ifort/icc compilers are compatible with OpenMP.

Language	Intel Compiler	GNU Compiler	AOCC Compiler
Fortran 77	$ ifort -openmp myprogram.f -o myprogram	$ gfortran -fopenmp myprogram.f -o myprogram	$ flang -fopenmp myprogram.f -o myprogram
Fortran 90	$ ifort -openmp myprogram.f90 -o myprogram	$ gfortran -fopenmp myprogram.f90 -o myprogram	$ flang -fopenmp myprogram.f90 -o myprogram
Fortran 95	$ ifort -openmp myprogram.f90 -o myprogram	$ gfortran -fopenmp myprogram.f90 -o myprogram	$ flang -fopenmp myprogram.f90 -o myprogram
C	$ icc -openmp myprogramram.c -o myprogram	$ gcc -fopenmp myprogram.c -o myprogram	$ clang -fopenmp myprogram.c -o myprogram
C++	$ icc -openmp myprogram.cpp -o myprogram	$ g++ -fopenmp myprogram.cpp -o myprogram	$ clang++ -fopenmp myprogram.cpp -o myprogram

*Intel compiler does not recognize the suffix ".f95". You may use ".f90" to stand for any Fortran code regardless of version as it is a free-formatted form

Compiling Hybrid Programs

The following tables illustrate how to compile your hybrid (MPI/OpenMP) program. Any compiler flags accepted by Intel ifort/icc compilers are compatible with their respective MPI compiler.		
Language	**Intel Compiler with Intel MPI(IMPI)**	**Intel/GNU/AOCC Compiler with OpenMPI/MVAPICH2**
Fortran 77	$ mpiifort -qopenmp myprogram.f -o myprogram	$ mpif77 -fopenmp myprogram.f -o myprogram
Fortran 90	$ mpiifort -qopenmp myprogram.f90 -o myprogram	$ mpif90 -fopenmp myprogram.f90 -o myprogram
Fortran 95	$ mpiifort -qopenmp myprogram.f90 -o myprogram	$ mpif90 -fopenmp myprogram.f90 -o myprogram
C	$ mpiicc -qopenmp myprogram.c -o myprogram	$ mpicc -fopenmp myprogram.c -o myprogram
C++	$ mpiicpc -qopenmp myprogram.C -o myprogram	$ mpicxx -fopenmp myprogram.C -o myprogram

*Intel compiler does not recognize the suffix ".f95". You may use ".f90" to stand for any Fortran code regardless of version as it is a free-formatted form

Agenda

4. Running jobs

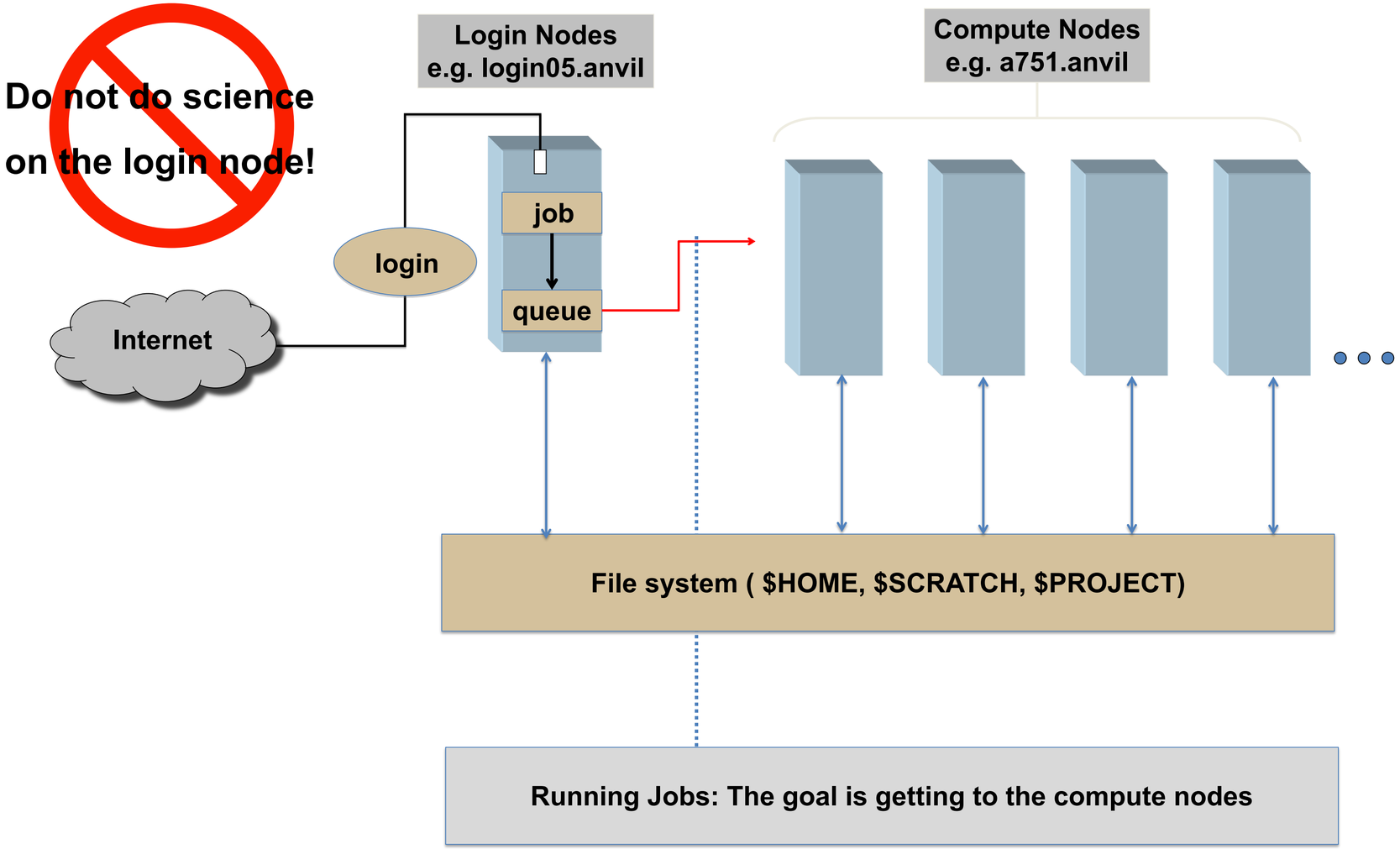
- Accessing to compute node
- Interactive jobs
- Job accounting
- Available queues
- Batch jobs & Examples

Agenda

4. Running jobs

- **Accessing to compute node**
- Interactive jobs
- Job accounting
- Available queues
- Batch jobs & Examples

LOGIN NODE VS COMPUTE NODE



Agenda

4. Running jobs

- Access to compute node
- **Interactive jobs**
- Job Accounting
- Available queues
- Batch jobs & Examples

Interactive Computing

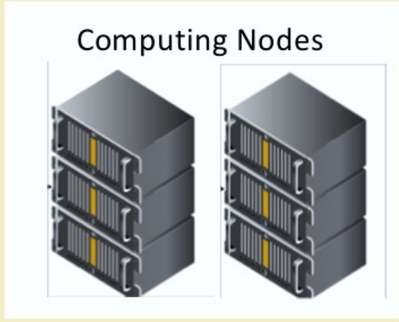
Interactive job: a job that occurs interactively with end users

Run simulations

Acquire simulation results

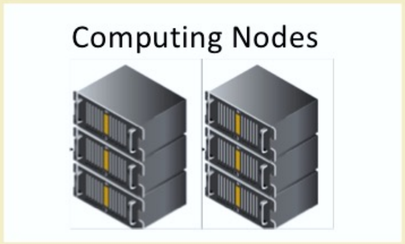
Process input data

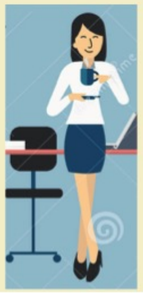
Acquire output data



Batch job: a job that does not need user interactions

Job queue





XSEDE
Extreme Science and Engineering Discovery Environment

Interactive Computing

Gateway

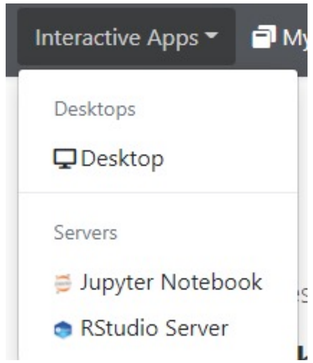
Remote Desktop

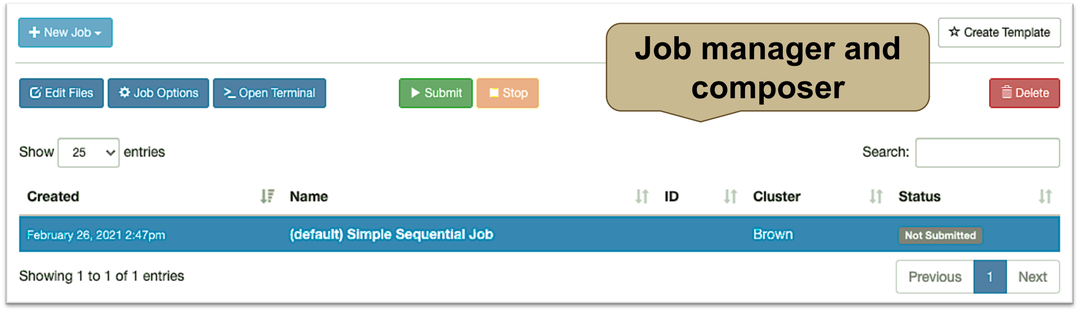
Low Barrier & Familiar Access to Compute

Quicker develop / test / debug cycle

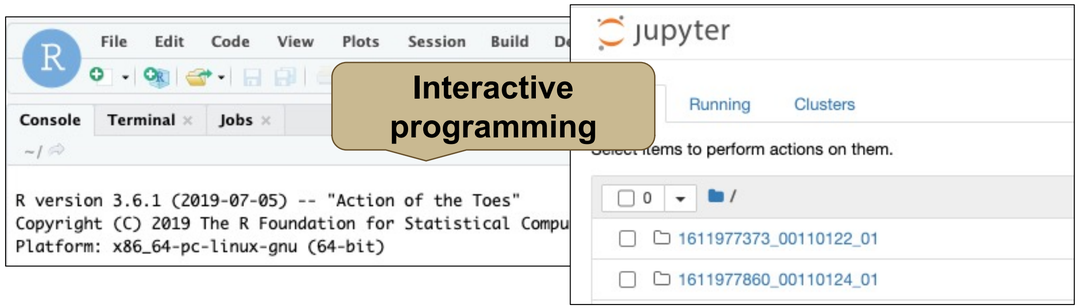
Job manager and composer

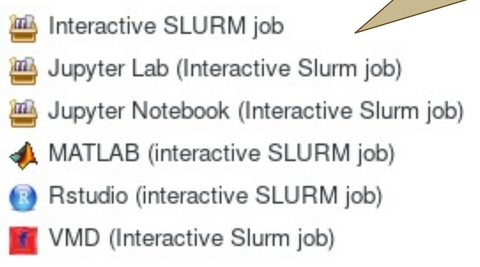
Interactive scientific applications





Interactive programming





Run GPI apps as job: Matlab, Fluent, Windows VM

PURDUE UNIVERSITY | Information Technology

Exercise: Run MPI Code with Interactive Job

1. cd $SCRATCH # go to your scratch directory

2. cp -r /anvil/datasets/training/anvil-101/interactive-test . # copy the test folder to your scratch directory

3. cd interactive-test # go to the interactive-test folder

4. ls

mpi_hello interactive-test-README

Exercise: Run MPI Code with Interactive Job

5. sinteractive -N 2 -n 2 -A tra220012 -t 00:30:00

This example asked for 2 nodes; 1 core on each node. The time limit is 30 mins.

salloc: Granted job allocation 198543

salloc: Waiting for resource configuration

salloc: Nodes a[478-479] are ready for job

- You can use the ***sinteractive*** command to run your job in an interactive session.
- ***sinteractive*** accepts most of the same resource requests as ***sbatch***
- To quit your interactive job: ***exit*** or ***Ctrl-D***

Exercise: Run MPI Code with Interactive Job

6. module purge # To unload all loaded modules and reset everything to original state

7. module load modtree/cpu # To load the default CPU environment recommended by the Anvil team

8. module list

Currently Loaded Modules:

1) gmp/6.2.1 2) mpfr/4.0.2 3) mpc/1.1.0 4) zlib/1.2.11 5) gcc/11.2.0 6) libfabric/1.12.0 7) numactl/2.0.14 8) openmpi/4.0.6 9) modtree/cpu

9. mpirun -np 2 mpi_hello # Use 2 cores with *mpirun* to run the MPI code

Runhost:a479.anvil.rcac.purdue.edu Rank: 1 of 2 ranks hello, world

Runhost:a478.anvil.rcac.purdue.edu Rank: 0 of 2 ranks hello, world

Agenda

4. Running jobs

- Access to compute node
- Interactive jobs
- **Job Accounting**
- Available queues
- Batch jobs & Examples

Job Accounting

- For **CPU** jobs, the charge unit is **Service Unit (SU)**, i.e. 1 CPU or ≤ ~2G memory for 1 hour, based on the actual resources used by your job.
  - If your job used 4 cores and 2 hours:
  - for **shared** queues job, charge = 4 cores x 2 hours = **8 SU**
  - for **node-exclusive** job, all **128** cores will be charged, even if only 4 cores are used, charge = 128 cores x 2 hours = **256 SU**
  - Jobs submitted to the **large memory nodes** will be charged **4 SU per core** (4x standard node charge).
- For **GPU** jobs, 1 SU is 1 GPU or ≤ ~64G memory for 1 hour. 4 GPU on a node. All GPU nodes are **shared**.
- Filesystem storage is not charged.

Agenda

4. Running jobs

- Access to compute node
- Interactive jobs
- Job Accounting
- **Available queues**
- Batch jobs & Examples

Slurm Partitions (Queues)

Anvil Production Queues							
Queue Name	Node Type	Max Nodes per Job	Max Cores per Job	Max Duration	Max running Jobs in Queue	Max running + submitted Jobs in Queue	Charging factor
debug	regular	2 nodes	256 cores	2 hrs	1	2	1
gpu-debug	gpu	1 node	2 gpus	0.5 hrs	1	2	1
standard	regular	16 nodes	2,048 cores	96 hrs	64	128	1
wide	regular	56 nodes	7,168 cores	12 hrs	5	10	1
shared	regular	1 node	128 cores	96 hrs	6400 cores		1
highmem	large-memory	1 node	128 cores	48 hrs	2	4	4
gpu	gpu			48 hrs	8 gpus		1

* For **gpu** queue: max of 12 GPU per job and max of 32 GPU in use by a single group.

Agenda

4. Running jobs

- Access to compute node
- Interactive jobs
- Job Accounting
- Available queues
- **Batch jobs & Examples**

Batch Script example: Serial Job in Standard Queue

```
#!/bin/bash
# FILENAME:  myjobsubmissionfile

#SBATCH -A myallocation              # Allocation name
#SBATCH --nodes=1                    # Total # of nodes (must be 1 for serial job)
#SBATCH --ntasks=1                   # Total # of tasks (should be 1 for serial job)
#SBATCH --time=1:30:00               # Total run time limit (hh:mm:ss)
#SBATCH -J myjobname                 # Job name
#SBATCH -o myjob.o%j                 # Name of stdout output file
#SBATCH -e myjob.e%j                 # Name of stderr error file
#SBATCH -p standard                  # Queue (partition) name
#SBATCH --mail-user=useremailaddress
#SBATCH --mail-type=all              # Send email to above address at begin and end of job
#SBATCH --output=/path/myjob.out     # Redirect job output to somewhere other than the default location
#SBATCH --error=/path/myjob.out      # Redirect job error to somewhere other than the default location

# Manage processing environment, load compilers and applications.
module purge
module load compilername
module load applicationname
module list

# Launch serial code
./myexecutablefiles
```

Common Slurm Commands

- Submit jobs

```
$ sbatch mysubmissionfile

Submitted batch job 188
```

- Check job status

```
$ squeue -u myusername

  JOBID   PARTITION   NAME    USER       ST   TIME   NODES   NODELIST(REASON)
  188       standard     job1   myusername  R    0:14      2         a[010-011]
  189       standard     job2   myusername  PD   0:15      1         a012
```

R -- running
PD -- pending

- Check queued or running job information

```
$ scontrol show job 189

JobId=189 JobName=myjobname
    UserId=myusername GroupId=mygroup MCS_label=N/A
    Priority=103076 Nice=0 Account=myacct QOS=normal
    JobState=RUNNING Reason=None Dependency=(null)
    Requeue=1 Restarts=0 BatchFlag=0 Reboot=0 ExitCode=0:0
    RunTime=00:01:28 TimeLimit=00:30:00 TimeMin=N/A
    SubmitTime=2021-10-04T14:59:52 EligibleTime=2021-10-04T14:59:52
    AccrueTime=Unknown
   StartTime=2021-10-04T14:59:52 EndTime=2021-10-04T15:29:52 Deadline=N/A
...
```

JobState: if the job is Pending, Running, Completed, or Held.

RunTime & TimeLimit: how long the job has run and maximum run time.

SubmitTime: when the job was submitted to the cluster.

WorkDir: the job's working directory.

StdOut & Stderr: locations of stdout and stderr of the job.

Reason: why a PENDING job isn't running.

Common Slurm Commands

- Check historic (completed) job information

```
$ jobinfo 189

Name           : interactive
User           : hong400
Account        : rcac
Partition      : standard
Nodes          : a010
Cores          : 1
GPUs           : 0
State          : TIMEOUT
ExitCode       : 0:0
Submit         : 2021-10-04T14:59:52
Start          : 2021-10-04T14:59:52
End            : 2021-10-04T15:30:20
Waited         : 00:00:00
…
```

- Kill a job

```
$ scancel myjobid
```

Exercise: Submit a batch job

1. cd $SCRATCH # go to your scratch directory

2. cp -r /anvil/datasets/training/anvil-101/sbatch-test . # copy the test folder to your scratch directory

3. cd sbatch-test # go to the sbatch-test folder

4. ls

hello.py myjobsubmitscript sbatch-test-README

5. sbatch myjobsubmitscript # submit a sbatch job

Submitted batch job XXXXXX

Exercise: Submit a batch job

6. squeue -u myusername # check job status under myusername

7. scontrol show job XXXXXX # check queued or running job information with my jobID

8. ls

9. vi myjob.oXXXXXX # check job output with my jobID

10. scancel XXXXXX # kill the job with my jobID

11. jobinfo XXXXXX # check historic (completed) job information with my jobID

MPI Job in Standard Queue

```
#!/bin/bash
# FILENAME:  myjobsubmissionfile

#SBATCH -A myallocation          # Allocation name
#SBATCH --nodes=2                # Total # of nodes
#SBATCH --ntasks=256             # Total # of tasks
#SBATCH --time=1:30:00           # Total run time limit (hh:mm:ss)
#SBATCH -p standard              # Queue (partition) name
#SBATCH --mail-user=useremailaddress
#SBATCH--mail-type=all           # Send email to above address at begin and end of job

# Manage processing environment, load compilers and applications.
module purge
module load compilername
module load mpilibrary
module load applicationname
module list

# Launch MPI code
mpirun -np $SLURM_NTASKS myexecutablefiles
```

OpenMP Job in Standard Queue

```
#!/bin/bash
# FILENAME:  myjobsubmissionfile

#SBATCH -A myallocation        # Allocation name
#SBATCH --nodes=1              # Total # of nodes (must be 1 for OpenMP job)
#SBATCH --ntasks=1             # Total # of tasks
#SBATCH --cpus-per-task=128    # cpu-cores per task (default value is 1, >1 for multi-threaded tasks)
#SBATCH --time=1:30:00         # Total run time limit (hh:mm:ss)
#SBATCH -p standard            # Queue (partition) name

# Manage processing environment, load compilers and applications.
module purge
module load compilername
module load applicationname
module list

# Set thread count (default value is 1).
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Launch OpenMP code
./myexecutablefiles
```

When running OpenMP programs, all threads must be on the same compute node to take advantage of shared memory. The threads cannot communicate between nodes.

Hybrid Job in Standard Queue

```
#!/bin/bash
# FILENAME:  myjobsubmissionfile

#SBATCH -A myallocation          # Allocation name
#SBATCH --nodes=2                # Total # of nodes
#SBATCH --ntasks-per-node=2      # Total # of MPI tasks per node
#SBATCH --cpus-per-task=64       # cpu-cores per task (default value is 1, >1 for multi-threaded tasks)
#SBATCH --time=1:30:00           # Total run time limit (hh:mm:ss)
#SBATCH -p standard              # Queue (partition) name

# Manage processing environment, load compilers and applications.
module purge
module load compilername
module load mpilibrary
module load applicationname
module list

# Set thread count (default value is 1).
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Launch MPI code
mpirun -np $SLURM_NTASKS myexecutablefiles
```

This example asks for **4 MPI tasks** (with **2 MPI tasks per node**), each with **64 OpenMP threads** for a total of **256 CPU-cores**.

GPU job in GPU queue

```
#!/bin/bash
# FILENAME:  myjobsubmissionfile

#SBATCH -A myallocation        # Allocation name for GPU
#SBATCH --nodes=1              # Total # of nodes (must be 1 for serial job)
#SBATCH --ntasks=1             # Total # of tasks
#SBATCH --gres=gpu:1           # Number of GPUs per node
#SBATCH --time=1:30:00         # Total run time limit (hh:mm:ss)
#SBATCH -p gpu                 # Queue (partition) name
#SBATCH --mail-user=useremailaddress
#SBATCH --mail-type=all        # Send email to above address at begin and end of job

# Manage processing environment, load compilers and applications.
module purge
module load modtree/gpu
module load applicationname
module list

# Launch GPU code
./myexecutablefiles
```

When running on multiple GPUs with MPI, you need to ensure one MPI rank per GPU.

Make sure to use ***--gres=gpu*** command, instead of ***--gpu*** or ***–G***. Otherwise, your job may not run properly.

You can use ***sfeatures*** command to see the detailed hardware overview.

NGC GPU Container Job in GPU Queue

XSEDE
Extreme Science and Engineering Discovery Environment

What is NGC?

- Nvidia GPU Cloud (NGC) is a GPU-accelerated cloud platform optimized for deep learning and scientific computing.
- Anvil team provides pre-downloaded NGC containers as convenient modules, so that you can use NGC containers as non-containerized versions of each application. More information can be found at **Anvil NGC containers**: https://www.rcac.purdue.edu/knowledge/anvil/run/examples/slurm/ngc

On Anvil, type the command below to see the lists of NGC containers we deployed:

```
$ module load modtree/gpu
$ module load ngc
$ module avail
------------------------------------------------------------ /opt/spack/ngc ------------------------------------------------------------
 autodock/2020.06             namd/2.13-multinode            pytorch/20.11-py3              rapidsai/0.17              tensorflow/20.06-tf2-py3
 games/17.09-r2-libcchem      namd/2.13-singlenode    (D)    pytorch/20.12-py3              rapidsai/21.06             tensorflow/20.11-tf1-py3
 gromacs/2018.2               namd/3.0-alpha3-singlenode     pytorch/21.06-py3              rapidsai/21.10     (D)     tensorflow/20.11-tf2-py3
 gromacs/2020.2               nvhpc/20.7                     pytorch/21.09-py3       (D)    relion/2.1.b1              tensorflow/20.12-tf1-py3
 gromacs/2021                 nvhpc/20.9                     qmcpack/v3.5.0                 relion/3.1.0               tensorflow/20.12-tf2-py3
 gromacs/2021.3      (D)      nvhpc/20.11                    quantum_espresso/v6.6a1        relion/3.1.2               tensorflow/21.06-tf1-py3
 julia/v1.5.0                 nvhpc/21.5                     quantum_espresso/v6.7   (D)    relion/3.1.3       (D)     tensorflow/21.06-tf2-py3
 julia/v2.4.2                 nvhpc/21.9              (D)    rapidsai/0.12                  tensorflow/20.02-tf1-py3   tensorflow/21.09-tf1-py3
 lammps/10Feb2021             paraview/5.9.0                 rapidsai/0.13                  tensorflow/20.02-tf2-py3   tensorflow/21.09-tf2-py3 (D)
 lammps/15Jun2020             pytorch/20.02-py3              rapidsai/0.14                  tensorflow/20.03-tf1-py3   torchani/2021.04
 lammps/24Oct2018             pytorch/20.03-py3              rapidsai/0.15                  tensorflow/20.03-tf2-py3
 lammps/29Oct2020             pytorch/20.06-py3              rapidsai/0.16                  tensorflow/20.06-tf1-py3
```

NGC GPU Container Job in GPU Queue

```
#!/bin/bash
# FILENAME:  myjobsubmissionfile

#SBATCH -A myallocation        # Allocation name for GPU
#SBATCH --nodes=1              # Total # of nodes (must be 1 for serial job)
#SBATCH --ntasks=1             # Total # of tasks
#SBATCH --gres=gpu:1           # Number of GPUs per node
#SBATCH --time=1:30:00         # Total run time limit (hh:mm:ss)
#SBATCH -p gpu                 # Queue (partition) name

# Manage processing environment, load compilers and applications.
module purge
module load modtree/gpu
module load ngc
module load applicationname
module list

# Launch GPU code
./myexecutablefiles
```

When running on multiple GPUs with MPI, you need to ensure one MPI rank per GPU.

Agenda

5. Data management and transfer

- File system
- Scp, Rsync, SFTP, Globus
- Lost file recovery

Agenda

5. Data management and transfer

- **File system**
- Scp, Rsync, SFTP, Globus
- Lost file recovery

File Systems

Anvil File Systems					
File System	**Mount Point**	**Quota**	**Snapshots**	**Purpose**	**Purge policy**
Anvil ZFS	/home **$HOME**	25 GB	Full schedule*	Home directories: area for storing personal software, scripts, compiling, editing, etc.	Not purged
Anvil ZFS	/apps	N/A	Weekly*	Applications	
Anvil GPFS	/anvil	N/A	No		
Anvil GPFS	/anvil/scratch **$SCRATCH**	100 TB	No	User scratch: area for job I/O activity, temporary storage	Files older than 30-day (access time) will be purged
Anvil GPFS	/anvil/projects **$PROJECT or $WORK**	5 TB	Full schedule*	Per allocation: area for shared data in a project, common datasets and software installation	Not purged while allocation is active. Removed 90 days after allocation expiration
Anvil GPFS	/anvil/datasets	N/A	Weekly*	Common data sets (not allocated to users)	
Versity	N/A (Globus)	20 TB	No	Tape storage per allocation	

* Full schedule keeps nightly snapshots for 7 days, weekly snapshots for 3 weeks, and monthly snapshots for 2 months.

Agenda

5. Data management and transfer

- File system
- **Scp, Rsync, SFTP, Globus**
- Lost file recovery

Transferring Files

Users can transfer files between Anvil and Linux-based systems or Mac or windows terminal using either ***scp*** or ***rsync*** or ***SFTP***.

- **SCP** (Secure CoPy) is a simple way of transferring files between two machines that use the SSH protocol.

  NOTE: SSH Keys is ***required*** for SCP.

  Following is an example of transferring a ***test.txt*** file from Anvil home directory to local machine, make sure to use your anvil user name **x-anvilusername**:

```
localhost> scp x-anvilusername@anvil.rcac.purdue.edu:/home/x-anvilusername/test.txt .
Warning: Permanently added the xxxxxxx host key for IP address 'xxx.xxx.xxx.xxx' to the list of known hosts.
test.txt                                      100%    0     0.0KB/s   00:00
```

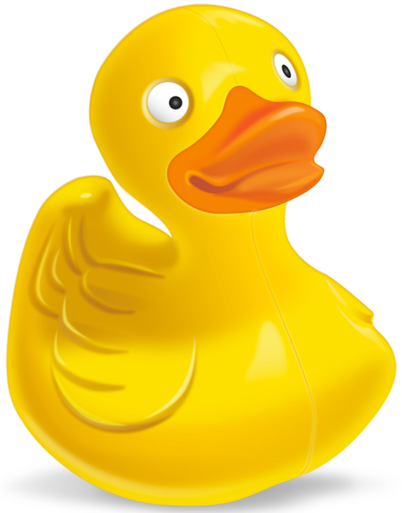
- **Rsync**, or Remote Sync lets you transfer files and directories to local and remote destinations. It allows to copy only the changes from the source and offers customization, use for mirroring, performing backups, or migrating data between different filesystems.

  NOTE: SSH Keys is ***required*** for Rsync. Also make sure to use your anvil user name **x-anvilusername**:

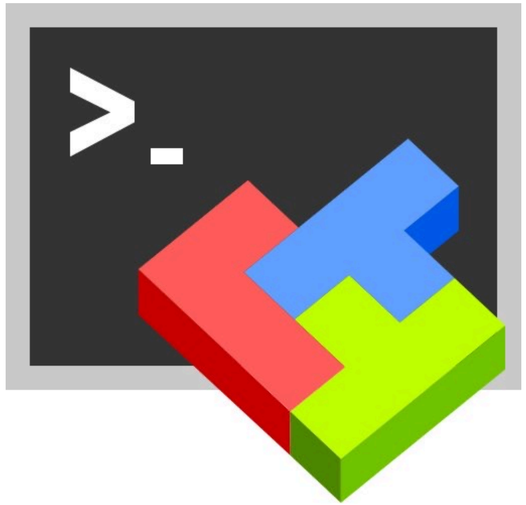
Transferring Files

- **SFTP** (Secure File Transfer Protocol) is available as graphical file transfer programs and as a command-line program. **SFTP** has more features than **SCP** and allows for other operations on remote files, remote directory listing, and resuming interrupted transfers.

- More details can be found at **Anvil File Transfer-SFTP:** www.rcac.purdue.edu/knowledge/anvil/storage/transfer/sftp



Cyberduck for Mac OS X



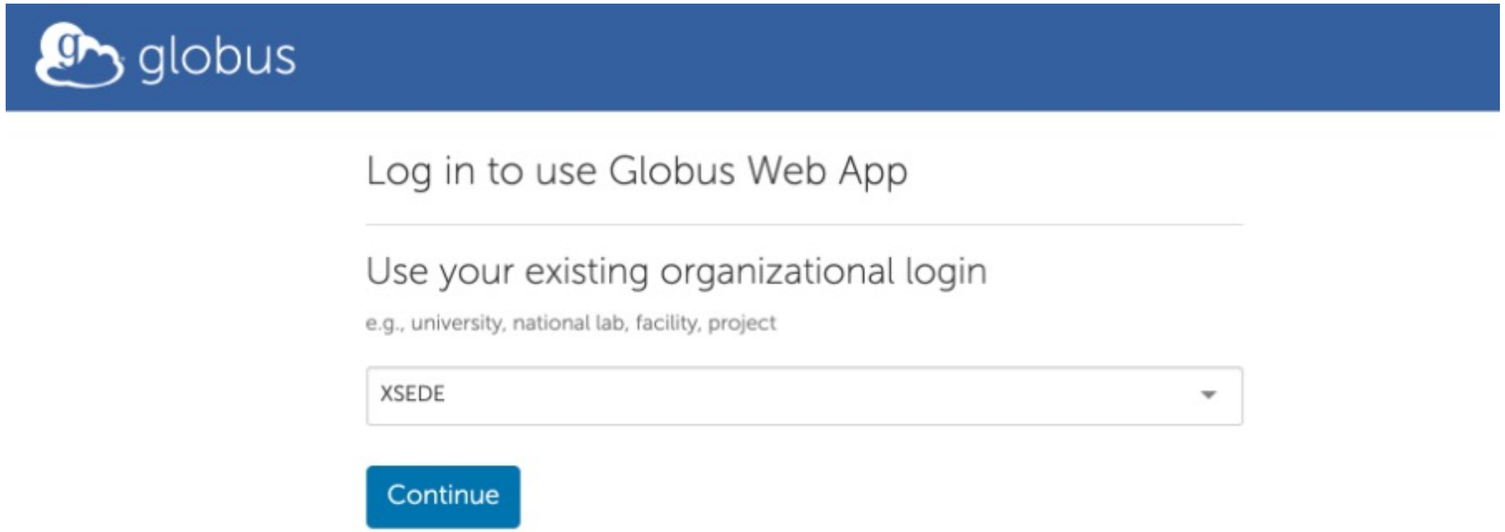
MobaXterm for Microsoft Windows

Transferring Files

- **Globus** is also a powerful and easy to use file transfer. It works between any XSEDE and non-XSEDE sites running Globus, and it connects any of these research systems to personal systems.

  You may use Globus to connect to your home, scratch, and project storage directories on Anvil. Since Globus is web-based, it works on any operating system connected to the internet.

  More details can be found at **XSEDE Data Transfer & Management:** https://portal.xsede.org/data-management



Agenda

5. Data management and transfer

- File system
- Scp, Rsync, SFTP, Globus
- **Lost file recovery**

Lost File Recovery

- Your ***$HOME*** and ***$PROJECT*** directories on Anvil are protected. A series of snapshots are taken every night after midnight. Each snapshot provides the state of your files at the time.

- These snapshots are kept for a limited time at various intervals. Please refer to **Anvil File Systems:** www.rcac.purdue.edu/knowledge/anvil/storage/filesystems for more detail.

- Only files saved during an overnight snapshot are recoverable. If you lose a file the same day you created it, the file is not recoverable.

- Snapshots are **not** a substitute for regular backups. For additional security, you might consider off-site back up important data (e.g. use Globus to transfer to your institution, etc)

Lost File Recovery

- If you know when you lost the file, you can use the ***flost*** command.
- The default location ***flost*** looks at is $HOME directory. For other location (e.g. in $PROJECT), you need to specify where the lost file was with ***-w*** argument.
- If you do not know the date, you may try entering different dates to ***flost***.
- Or you may manually browse the snapshots in ***/home/.zfs/snapshot*** folder for **$HOME** directory or ***/anvil/projects/.snapshots*** folder for $**PROJECT** directory.

Agenda

6. Helpful tips

Helpful Tools

The Anvil cluster provides a list of useful auxiliary tools:

The following table provides a list of auxiliary tools:	
Tools	**Use**
myquota	Check the quota of different file systems
flost	A utility to recover files from snapshots
showpartitions	Display all Slurm partitions and their current usage
myscratch	Show the path to your scratch directory
jobinfo	Collates job information from the *sstat*, *sacct* and *squeue* SLURM commands to give a uniform interface for both current and historical jobs
sfeatures	Show the list of available constraint feature names for different node types.
myproject	print the location of my project directory
mybalance	Check the allocation usage of your project team

THANK YOU!

Contact Us

For user support please submit a ticket at **Help Desk**, by selecting the appropriate Anvil resource to have it routed to us.

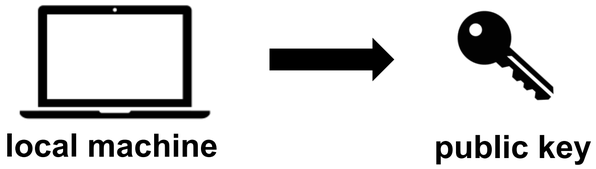
Appendix

Logging in SSH Keys

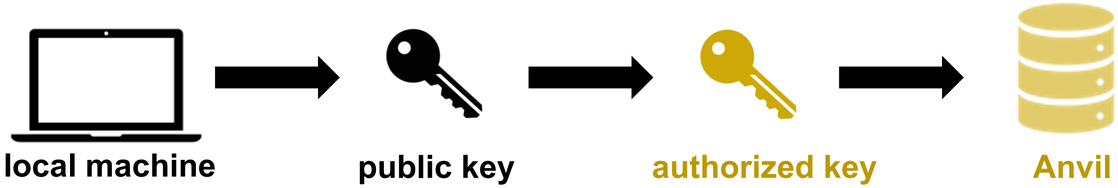
SSH key is an **access credential** in the SSH protocol. It functions similarly to username and password, but the key is primarily used to automated processes and enable single sign-on.

To connect to Anvil using SSH keys, you must follow three high-level steps:

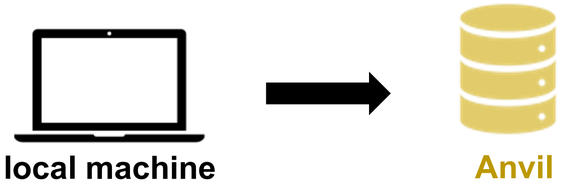
1. **Generate a key pair consisting of a private and a public key on your local machine.**



2. **Copy the public key to the cluster and append it to $HOME/.ssh/authorized_keys file in your account.**



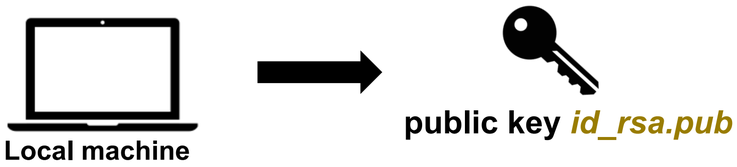
3. **Test if you can ssh from your local computer to the cluster without using XSEDE's Single Sign On (SSO) login hub.**



Logging in SSH Keys

For Mac and Linux:

1. Run ssh-keygen in a terminal on your local machine.



You may supply a filename & a passphrase to protect your private key. Or to accept the default settings, press Enter.

Note: If you do not protect private key with a passphrase, anyone with access to your computer could SSH to your account on Anvil.

By default, the key files will be stored in *~/.ssh/id_rsa* and *~/.ssh/id_rsa.pub* on your local machine.

```
[localhost]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (localhost/.ssh/id_rsa):
```

```
Created directory 'localhost/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in localhost/.ssh/id_rsa.
Your public key has been saved in localhost/.ssh/id_rsa.pub.
The key fingerprint is:
...
The key's randomart image is:
...
```

Logging in SSH Keys

3. Go to the *~/.ssh* folder in your local machine and cat the key information in *id_rsa.pub file*.

```
[localhost/.ssh]$ cat id_rsa.pub
ssh-rsa ... localhost-username@localhost
```

4. Login to Anvil with your XSEDE username and password through XSEDE Single Sign-On (SSO) hub. Then go to the **home** directory on Anvil, make a directory *mkdir -p ~/.ssh* if it does not exist.

```
[x-anvilusername@login01:~]$ cd ~/.ssh
[x-anvilusername@login01:~/.ssh]$ vi authorized_keys
# copy-paste the contents of the public key id_rsa.pub in your local machine to here and save the change of authorized_keys file. Then it is all set!
```

5. Create a file *~/.ssh/authorized_keys* on Anvil and copy the contents of the public key *id_rsa.pub* on your local machine into ~/.ssh/authorized_keys on Anvil.

6. Test the new key by SSH-ing to the server. The login should now complete without asking for a password.

```
[localhost]$ ssh x-anvilusername@anvil.rcac.purdue.edu
================================================
==                  Welcome to the Anvil Cluster
...
================================================[
[x-anvilusername@login06:~]
```

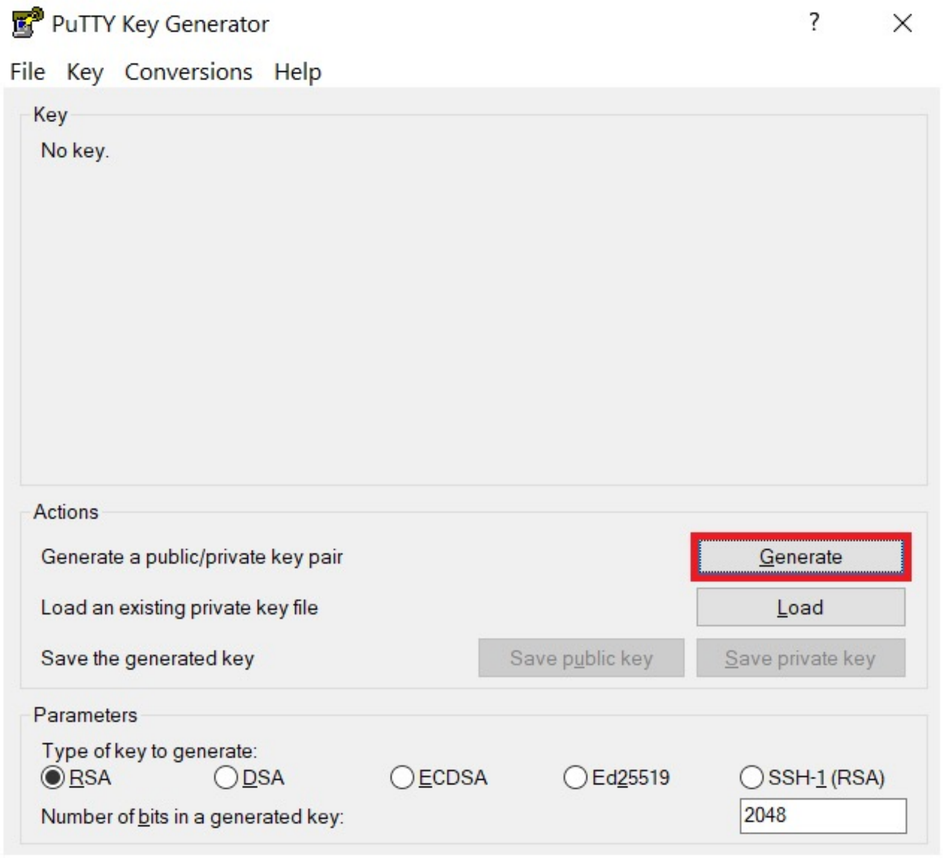
Logging in SSH Keys

For Windows:

Windows SSH Instructions	
Programs	**Instructions**
MobaXterm	Open a local terminal and follow Linux steps
Git Bash	Follow Linux steps
Windows 10 PowerShell	Follow Linux steps
Windows 10 Subsystem for Linux	Follow Linux steps
PuTTY	Follow steps below

Logging in SSH Keys

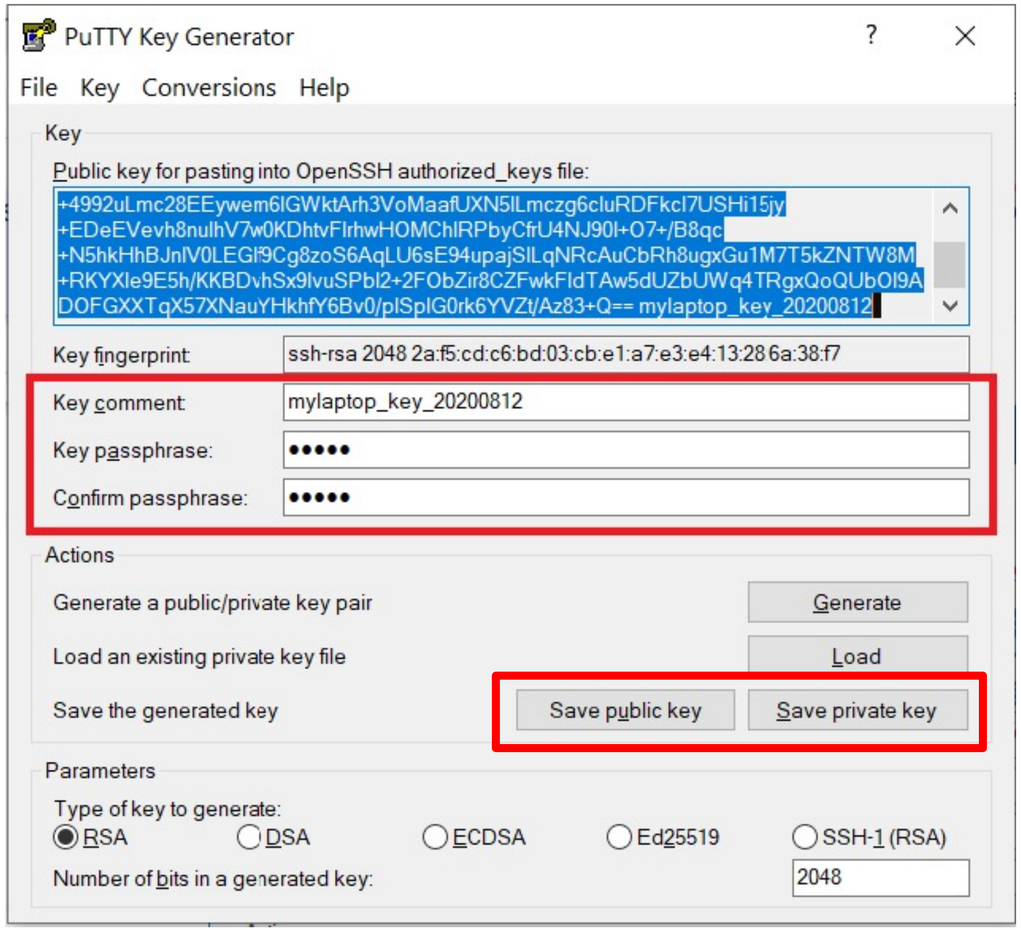
For Windows PuTTY:



1. Launch **PuTTYgen** (another software, not **PuTTY**), keep the default key type (RSA) and length (2048-bits) and click **Generate** button.

Logging in SSH Keys

For Windows PuTTY:



2. Once the key pair is generated:

- Use the **Save public key** button to save the public key, e.g. *Documents\SSH_Keys\mylaptop_public_key.pub*

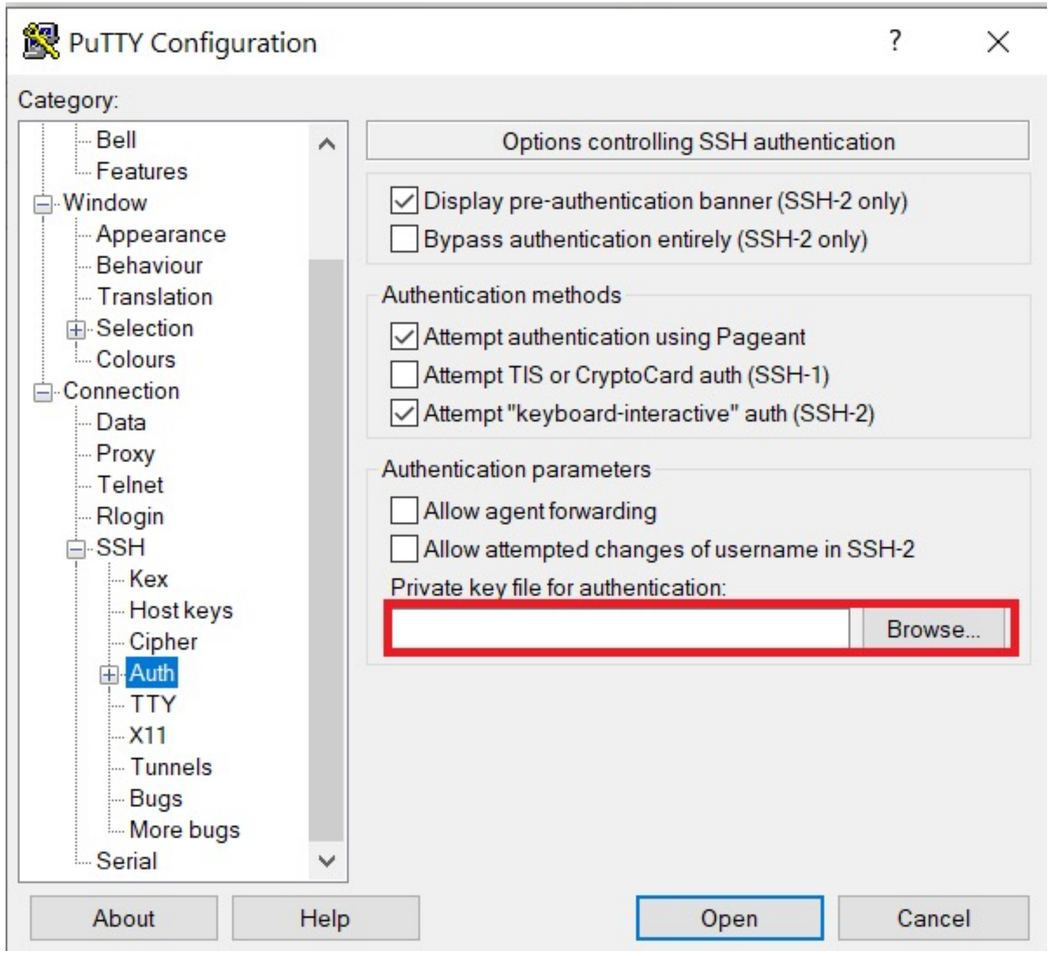
- Use the **Save private key** button to save the private key, e.g. *Documents\SSH_Keys\mylaptop_private_key.ppk*

  When saving the private key, you can also choose a reminder key comment, as well as an optional key passphrase to protect your key.

  Note: If you do not protect your private key with a passphrase, anyone with access to your computer could SSH to your account on Anvil.

Logging in SSH Keys

For Windows PuTTY:



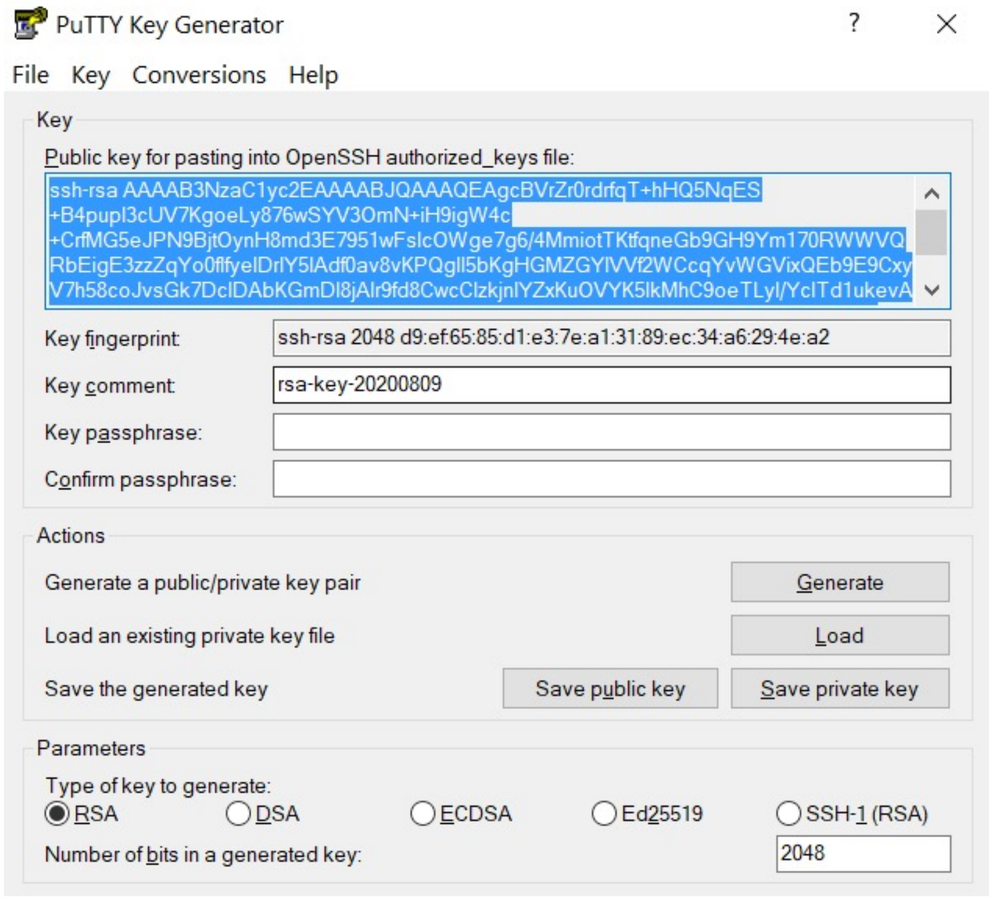
3. Configure **PuTTY** to use key-based authentication:

- Launch PuTTY and navigate to "**Connection** -> **SSH** ->**Auth**" on the left panel, click **Browse** button under the "Authentication parameters" section and choose your **private key**, e.g. mylaptop_private_key.ppk

- Navigate back to "**Session**" on the left panel. Highlight "**Default Settings**" and click the "**Save**" button to ensure the change in place.

Logging in SSH Keys

For Windows PuTTY:

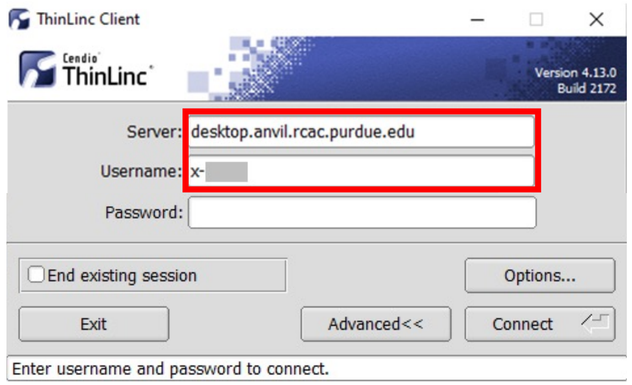


4. Login to Anvil with XSEDE username and password via XSEDE Single Sign-On (SSO) hub. Then go to the **home** directory, make a directory *mkdir -p ~/.ssh* if it does not exist.

5. Create a file *~/.ssh/authorized_keys* on Anvil and copy the contents of the public from PuTTYgen and paste it into *~/.ssh/authorized_keys* on Anvil. Please double-check that your text editor did not wrap or fold the pasted value (it should be one very long line).

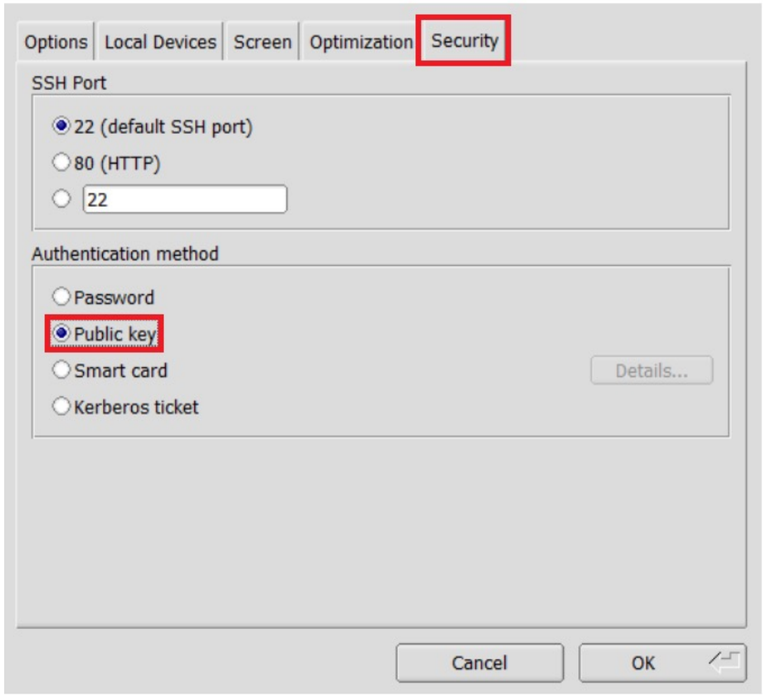
6. **Test** by connecting to the cluster and the login should now complete without asking for a password. If you chose to protect your private key with a passphrase in step 2, you will be prompted to enter the passphrase when connecting

ThinLinc

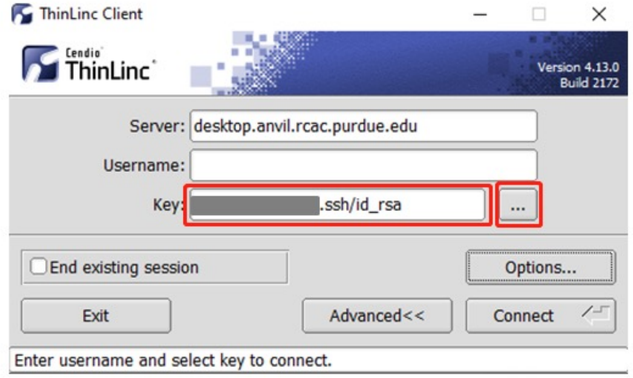


Server: *desktop.anvil.rcac.purdue.edu*

Use your Anvil **username:** *x-anvilusername*



In the **options** dialog, switch to the "**Security**" tab and select the "**Public key**" radio button:



In the **Key** field, type your locally ssh keys path or click ... button to locate and select the key.

Note: If PuTTY is used to generate the SSH Key pairs, please choose the private key in the *openssh* format.

Browser-based Thinlinc access is not supported on Anvil at this moment. Please use native Thinlinc client with SSH keys.