

Anvil 101

New User Tutorial

The Anvil Team, Purdue Research Computing

Pre-Survey



Code of Conduct

This external code of conduct for ACCESS-sponsored events represents ACCESS'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 ACCESS-sponsored events, services, and interactions.

Webpage: <https://support.access-ci.org/code-conduct>

How to Submit a Ticket: <https://support-access-ci.org>

If you feel your safety is in jeopardy or the situation is an emergency, contact local law enforcement before making a report to ACCESS. (In the U.S., dial 911.)

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.”

Full Agenda

- **Anvil system architecture including node types, storage, interconnects, and networking.**
- **Sign up for Access/Anvil accounts and apply for allocations**
- **Install software and programming environment on Anvil**
- **Data management and transfer on Anvil**
- **Running Jobs on Anvil**
- **Q&A**

About Anvil

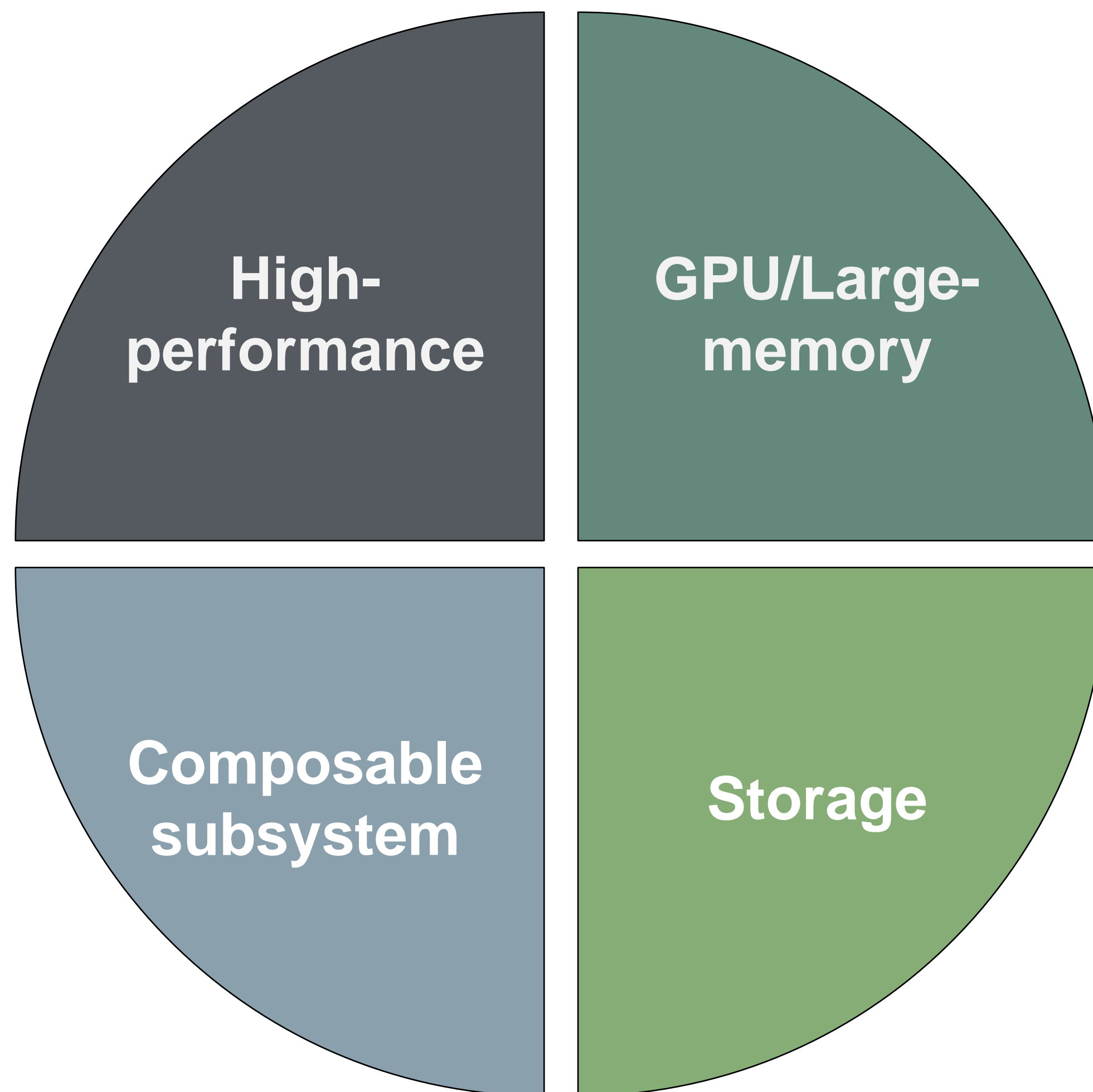
- **Category I:** A national composable advanced computational resource for the future of science and engineering
- By the Purdue research computing team. Full access started **February, 2022**
- NSF award **#2005632**; **5 years** of operations; allocated via NSF ACCESS



System Resources

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

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



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

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



Starting 4/11/24
every other
Thursday 2-3pm

Service & Support



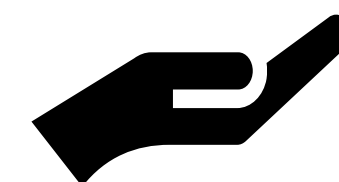
**Quick
turnaround
via ACCESS
support
tickets**

support.access-ci.org/



**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**

Obtaining ACCESS and Anvil Account

Anvil is accessible to ACCESS users who are given an allocation on the system. To obtain an account, users may submit a proposal through:

[ACCESS Allocation Request System:](https://allocations.access-ci.org/)
<https://allocations.access-ci.org/>

- Sign up for an ACCESS account (if you don't have one already) at <https://allocations.access-ci.org>
- Prepare an allocation request with details of your proposed computational workflows (science, software needs), resource requirements, and a short CV. See the individual "Preparing Your ... Request" pages for details on what documents are required:

Obtaining an Allocation

[How do I get onto Anvil through ACCESS?](#)

| Allocation | Credit Threshold |
|--|-------------------------|
| <u>Explore ACCESS</u> | 400,000 |
| <u>Discover ACCESS</u> | 1,500,000 |
| <u>Accelerate ACCESS</u> | 3,000,000 |
| <u>Maximize ACCESS</u> | Not awarded in credits. |

Obtaining an **Allocation**

When your request is approved, you only get ACCESS **credits** awarded. You still need to go through the step of exchanging these credits for time on Anvil.

You need not use up all your credits and may also use part of your credits for time on other ACCESS resources.

Exchange calculator (https://allocations.access-ci.org/exchange_calculator)

You will also need to go to the allocations page and add any users you would like to have access to these resources.

Note that they will need to sign up for ACCESS accounts as well before you can add them.

For other questions you may have, take a look at the FAQs on the ACCESS page here:

(<https://allocations.access-ci.org/ramps-policies-faqs>)

Obtaining an **Allocation**

When your ACCESS allocation is approved, you will receive an email from ACCESS.

After you transfer your credit to Anvil, it takes a bit of time for ACCESS to send the information to Anvil.

Agenda

2. Getting started

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

Logging in via **SSO Hub**

Anvil accepts standard SSH connections with public keys-based authentication to anvil.rcac.purdue.edu using your Anvil username:

```
localhost$ ssh my-x-anvil-username anvil.rcac.purdue.edu
```

Please note:

- Your Anvil username is not the same as your ACCESS portal username. Anvil usernames starts with an x-.
- Password-based authentication is not supported on Anvil (in favor of **SSH keys**). There is **NO "Anvil password"**, and your ACCESS password will not be accepted by Anvil's SSH either.

Logging in via SSH

Anvil accepts standard SSH connections with **public keys-based** authentication to *anvil.rcac.purdue.edu* using your Anvil username:

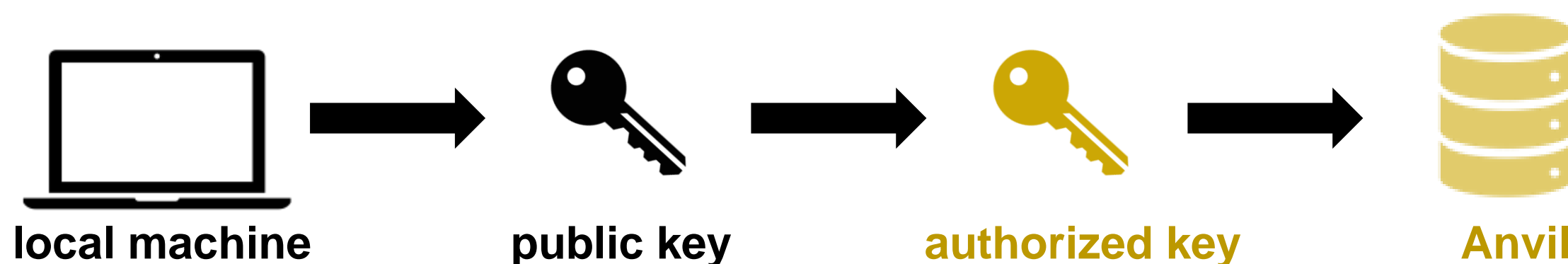
```
localhost$ ssh -l my-x-anvil-username anvil.rcac.purdue.edu
```



Your Anvil username is **not** the same as your ACCESS Portal username.
Anvil usernames look like x-ACCESSusername or similar, starting with an x-.

Password-based authentication is not supported on Anvil (in favor of SSH keys). There is no "Anvil password", and your ACCESS User Portal password will not be accepted by Anvil's SSH either.

Please see Appendix or [Anvil user guide](http://www.rcac.purdue.edu/knowledge/anvil/access/login/sshkeys) (www.rcac.purdue.edu/knowledge/anvil/access/login/sshkeys) for more detail about **SSH keys**.

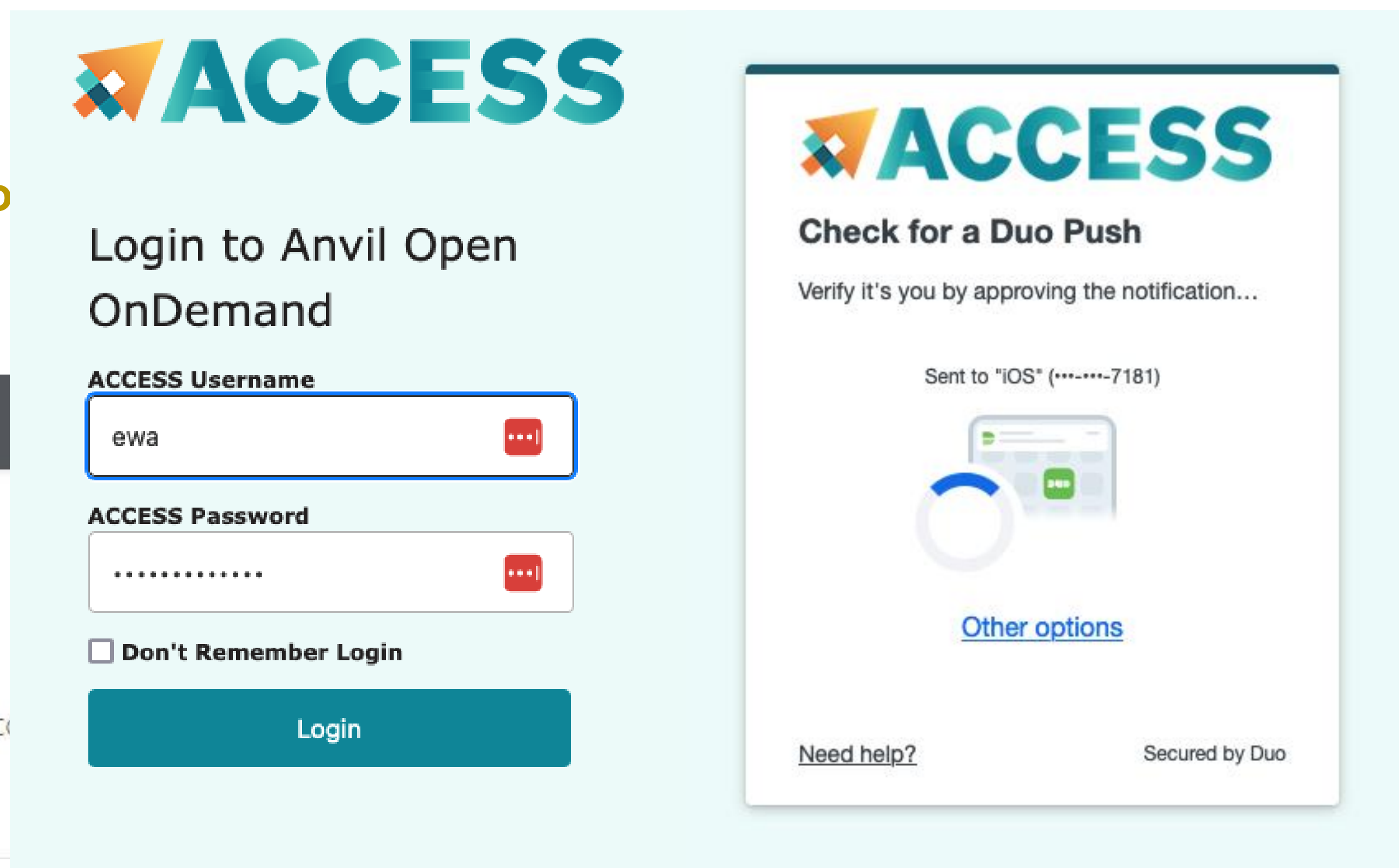
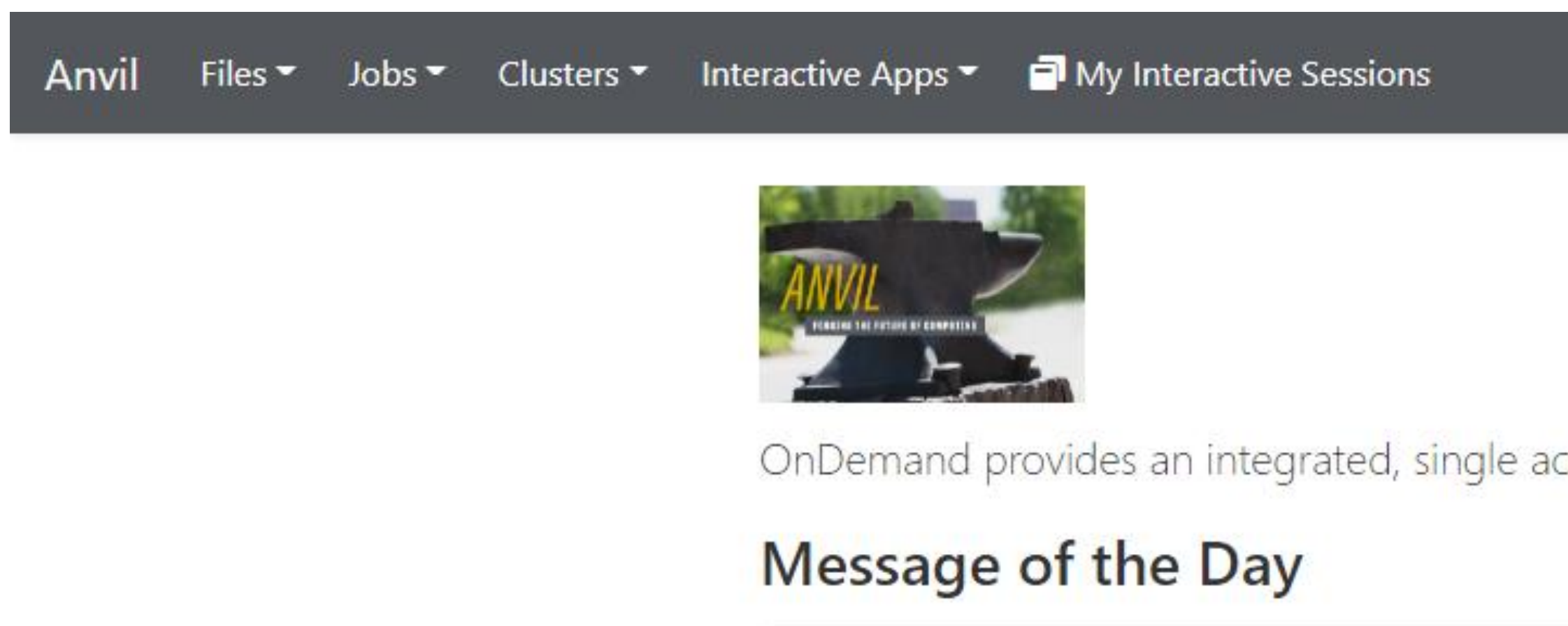


Open OnDemand

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 **ACCESS portal username and password**



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

The screenshot shows a web browser window with the URL `ondemand.anvil.rcac.purdue.edu/pun/sys/dashboard`. The page features a dark navigation bar with the following items: Anvil, Files, Jobs, Clusters, Interactive Apps, My Interactive Sessions, Help, Logged in as x-adams, and Log Out. Below the navigation bar is a banner for ANVIL with the text "OnDemand provides an integrated, single access point for all of your HPC resources." and a "Message of the Day" section. The main content area contains four dropdown menus, each highlighted with a yellow box:

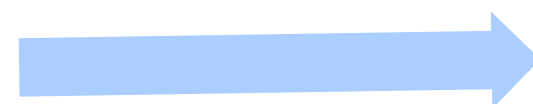
- Files**: Home Directory, Scratch `/anvil/scratch/x-`, Project `x- /anvil/projects/x-`, Project `x- /anvil/projects/x-`, Project `-`
- Jobs**: Active Jobs, Job Composer
- Clusters**: `_Anvil Shell Access`
- Interactive Apps**: Desktops, Desktop, Servers, Jupyter Notebook, RStudio Server

At the bottom left, it says "powered by OPEN OnDemand". At the bottom right, it says "OnDemand version: v2.0.13".

ACCESS Account

- Sign up at allocations.access-ci.org
- ACCESS username, password and 2FA
- Submit your proposal
- ACCESS credits

Transfer to SU



Anvil Account

- [Open On Demand](https://ondemand.anvil.rcac.purdue.edu): ondemand.anvil.rcac.purdue.edu
ACCESS username + password
- [ThinLinc](https://desktop.anvil.rcac.purdue.edu): desktop.anvil.rcac.purdue.edu
Anvil username (e.g. x-wu1) + **SSH key** (No password)
- [SSH](https://anvil.rcac.purdue.edu): anvil.rcac.purdue.edu
Anvil username + **SSH key** (No password)

First time
login



local machine



public key



authorized key



Anvil

```
ewa — x-adams@login04:~ — ssh x-adams@anvil.rcac.purdue.edu — 112x47
Last login: Thu Feb 23 13:06:28 on ttys000
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.
pal-nat187-12-232:~ ewa$ ssh x-adams@anvil.rcac.purdue.edu
=====
Welcome to the Anvil Cluster
=====
Anvil consists of:
Nodes:
Anvil-A   ppn=128   256 GB memory (standard, wide, shared, debug)
Anvil-B   ppn=128   1024 GB memory (highmem)
Anvil-G   ppn=128   512 GB memory (gpu, gpu-debug)
          + 4 NVIDIA A100 GPUs
Scratch:
Quota: 100 TB / 2 million files
Path: $SCRATCH
Type command: "myquota"
Partitions:
Type command: "showpartitions" or "sinfo -s"
Software:
Type command: "module avail" or "module spider"
User guide:
www.rcac.purdue.edu/knowledge/anvil
XSEDE Help Desk:
portal.xsede.org/help-desk
News:
www.rcac.purdue.edu/news/anvil
=====
Tip of the day (use "touch $HOME/.no.tips" to stop):
~~~~~
On Anvil 'mybalance' prints a list of your allocations and their SU balances.
x-adams@login04.anvil:[~] $
```

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 [ACCESS User Portal](https://allocations.access-ci.org/):

<https://allocations.access-ci.org/>

You should see at least one allocation.

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

Agenda

3. Compilation and programming environment

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

Agenda

3. Compilation and programming environment

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

Modules

- 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:

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

```
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
```

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 programming 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)

Provide Software: <https://purduercac-applications.readthedocs.io/en/latest/>

Need additional software? Please see the [Software Installation Request Policy](#).

Agenda

3. Compilation and programming 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.

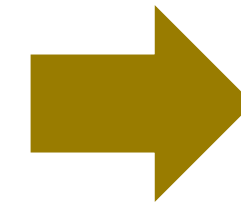
Example: Compiling Serial C++ Code

```
// 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

Compile and link the c++ code and generate the executable

\$ ls

serial_hello.cpp **serial**

Executable files generated

\$./serial

Runhost:a600.anvil.rcac.purdue.edu hello, 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 myprogram.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

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
```


Agenda

4. Running jobs

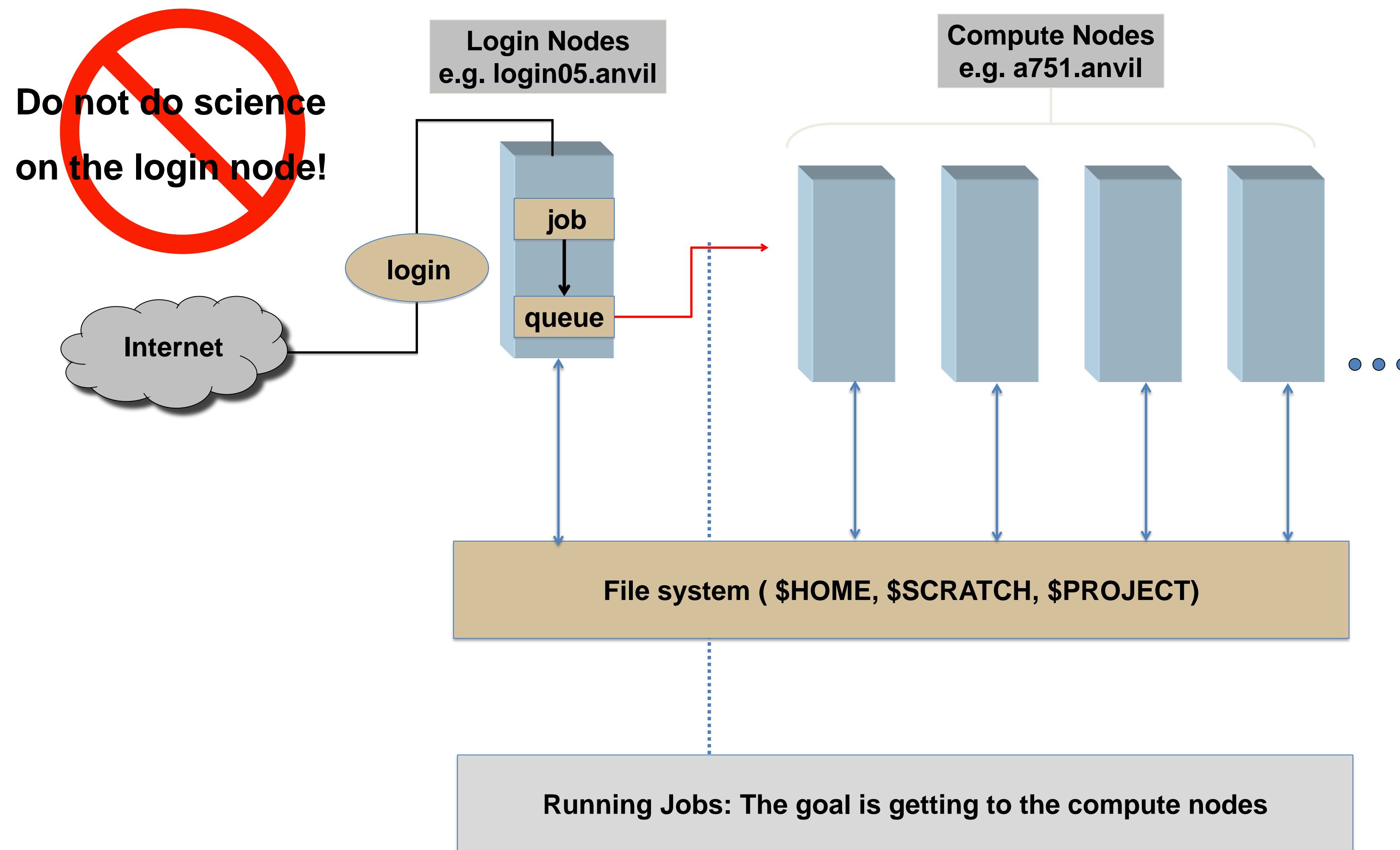
- **Accessing to compute node**
- **Interactive jobs**
- **Job accounting**
- **Available queues**
- **Batch jobs & Examples**

Agenda

4. Running jobs

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

LOGIN NODE VS COMPUTE NODE



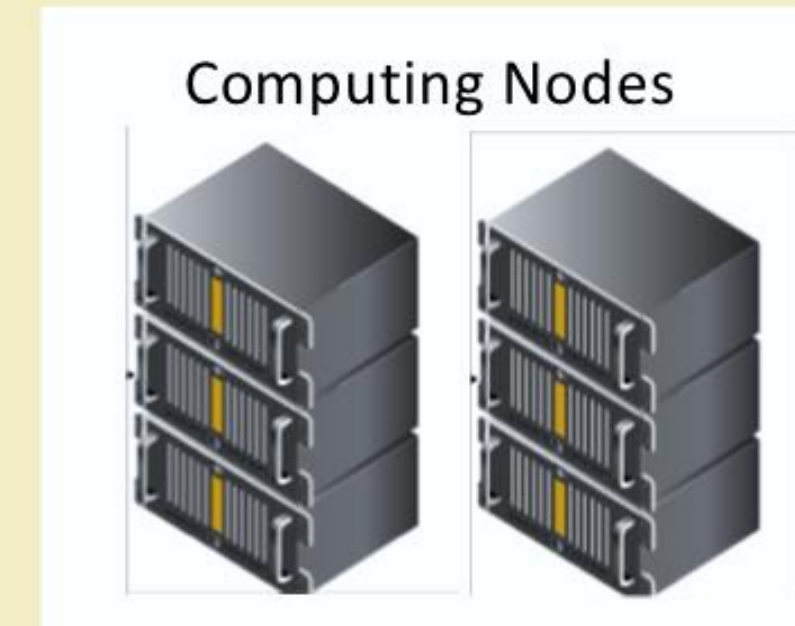
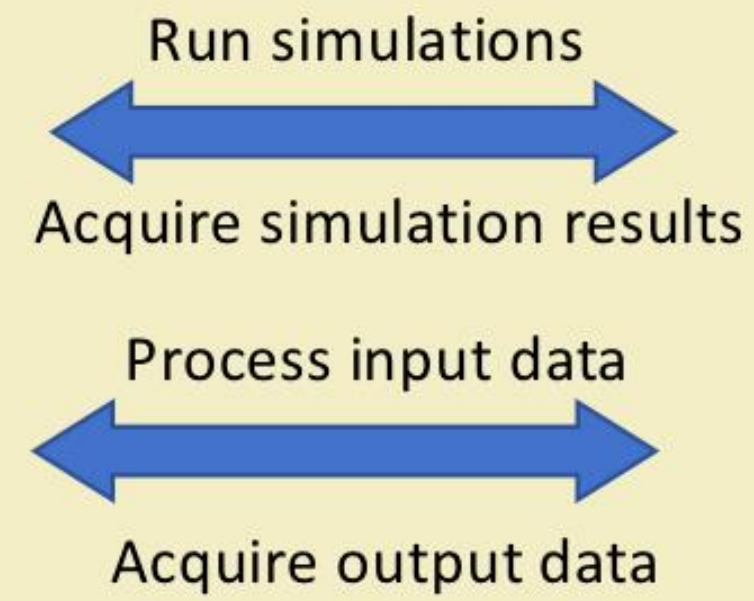
Agenda

4. Running jobs

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

Interactive Computing

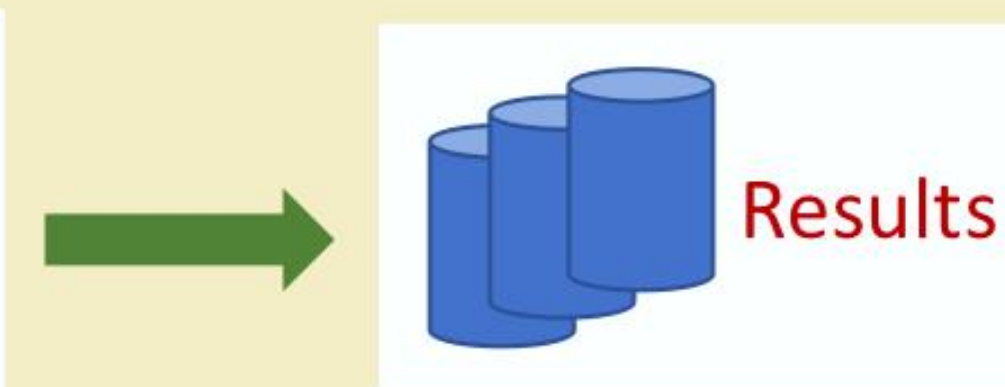
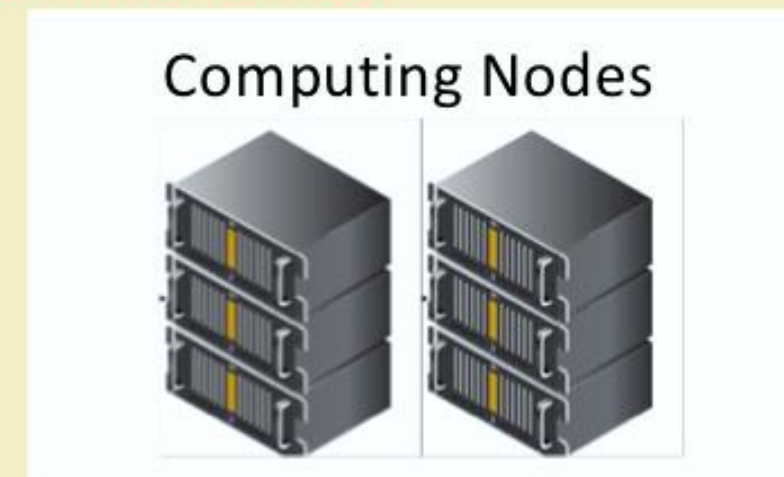
Interactive job: a job that occurs interactively with end users



Batch job: a job that does not need user interactions



Job queue



Interactive 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***

```
$ sinteractive -N 2 -n 256 -A myallocation -t 00:30:00
```

```
salloc: Granted job allocation 198543
```

```
salloc: Waiting for resource configuration
```

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

This example asked for 2 nodes.

128 cores on each node.

The time limit is 30 mins.

Interactive Computing

Gateway



Remote Desktop



Low Barrier & Familiar Access to Compute

Quicker develop / test / debug cycle

Job manager and composer

| Created | Name | ID | Cluster | Status |
|--------------------------|---------------------------------|----|---------|---------------|
| February 26, 2021 2:47pm | (default) Simple Sequential Job | | Brown | Not Submitted |

Interactive scientific applications

- Interactive SLURM job
- Jupyter Lab (Interactive Slurm job)
- Jupyter Notebook (Interactive Slurm job)
- MATLAB (interactive SLURM job)
- Rstudio (interactive SLURM job)
- VMD (Interactive Slurm job)

Interactive programming

```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
```

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

Agenda

4. Running jobs

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

Batch Script example: Serial Job in Shared 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 shared                 # Queue (partition) name
#SBATCH --mail-user=useremailaddress

# 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
```

- Kill a job

```
$ scancel myjobid
```

- Check job status

```
$ squeue -u myusername (or squeue --me)
```

| JOBID | PARTITION | NAME | USER | ST | TIME | NODES | |
|-------|-----------|------|------------|----|------|-------|---------------|
| 188 | wholenode | job1 | myusername | R | 0:14 | 2 | R -- running |
| 189 | wholenode | job2 | myusername | PD | 0:00 | 1 | PD -- pending |

Common Slurm Commands

- Check queued or running job information

```
$ scontrol show jobid
```

```
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
```

```
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  : wholenode  
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
```

```
...
```

Example: Submit a batch job

1. **cd sbatch-test**

go to the sbatch-test folder

2. **ls**

hello.py myjobsubmitscript

3. **sbatch myjobsubmitscript**

submit a sbatch job

Submitted batch job XXXXXX

4. **squeue -u myusername** or **squeue -me**

check job status under myusername

Example: Submit a batch job

5. **scontrol show job XXXXXX**

check queued or running job information with my jobID

6. **scancel XXXXXX**

kill the job with my jobID

7. **jobinfo XXXXXX**

check historic (completed) job information with my jobID

8. **vi myjob.oXXXXXX**

check job output file

9. **vi myjob.eXXXXXX**

check job error file

MPI Job in Wholenode 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 wholenode        # Queue (partition) name

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

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

OpenMP Job in Wholenode 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 wholenode       # 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 Wholenode 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 wholenode            # 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

2 MPI tasks per node

Each with 64 OpenMP threads

Total of 256 CPU-cores

GPU job in GPU queue

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

#SBATCH -A myGPUallocation # Allocation name for GPU
#SBATCH --nodes=1          # Total # of nodes (must be 1 for serial job)
#SBATCH --ntasks=1        # Total # of tasks
#SBATCH --gpus-per-node=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 ***gpus-per-node=1***. 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

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):
<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
gamess/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
lammmps/10Feb2021        paraview/5.9.0            rapidsai/0.13            tensorflow/20.02-tf2-py3 tensorflow/21.09-tf2-py3 (D)
lammmps/15Jun2020        pytorch/20.02-py3         rapidsai/0.14            tensorflow/20.03-tf1-py3 torchani/2021.04
lammmps/24Oct2018        pytorch/20.03-py3         rapidsai/0.15            tensorflow/20.03-tf2-py3
lammmps/29Oct2020        pytorch/20.06-py3         rapidsai/0.1659         tensorflow/20.06-tf1-py3
```

NGC GPU Container Job in GPU Queue

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

#SBATCH -A myGPUallocation # Allocation name for GPU
#SBATCH --nodes=1          # Total # of nodes (must be 1 for serial job)
#SBATCH --ntasks=1        # Total # of tasks
#SBATCH --gpus-per-node=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

4. Running jobs

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

Job Accounting

- For **CPU** jobs, the charge unit is **Service Unit (SU)**, i.e. 1 CPU using \leq ~2G memory for 1 hour, based on the actual resources tied up by your job.

Example: a 4 cores + 2 hours job:

- Submitted to **shared** queues job $\left\{ \begin{array}{l} \text{if mem} \leq \sim 8\text{G, charge} = 4 \text{ cores} \times 2 \text{ hours} = \mathbf{8 \text{ SU}} \\ \text{if mem} = 9\text{G, charge} = 5 \text{ cores} \times 2 \text{ hours} = \mathbf{10 \text{ SU}} \end{array} \right.$
- Submitted to **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 wholenode charge).

- For **GPU** jobs, 1 SU is 1 GPU using \leq ~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
- Batch jobs & Examples
- Job Accounting
- **Available queues**

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 |
| DEFAULT wholenode | 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.

Slurm Partitions (Queues)

\$ showpartitions

Partition statistics for cluster anvil at Tue Jun 21 11:02:14 EDT 2022

| Partition | | #Nodes | | #CPU_cores | | Cores_pending | | Job_Nodes | | MaxJobTime | Cores | Mem/Node |
|---------------------|---------------|----------------|----------------|------------------|------------------|---------------|------------------|--------------|------------------|---------------------|----------------|----------------|
| Name | State | Total | Idle | Total | Idle | Resorc | Other | Min | Max | Day-hr:mn | /node | (GB) |
| wholenode:* | up | 750 | 637 | 96000 | 81536 | 0 | 897 | 1 | infin | infinite | 128 | 257 |
| standard | up | 750 | 637 | 96000 | 81536 | 0 | 20676 | 1 | infin | infinite | 128 | 257 |
| shared | up | 250 | 245 | 32000 | 31551 | 0 | 0 | 1 | infin | infinite | 128 | 257 |
| wide | up | 750 | 637 | 96000 | 81536 | 0 | 0 | 1 | infin | infinite | 128 | 257 |
| highmem | up | 32 | 32 | 4096 | 4096 | 0 | 0 | 1 | infin | infinite | 128 | 1031 |
| debug | up | 17 | 17 | 2176 | 2176 | 0 | 0 | 1 | infin | infinite | 128 | 257 |
| gpu | up | 16 | 8 | 2048 | 1911 | 0 | 96 | 1 | infin | infinite | 128 | 515 |
| gpu-debug | up | 16 | 8 | 2048 | 1911 | 0 | 0 | 1 | infin | infinite | 128 | 515 |

* wholenode is the default partition.

standard partition will be removed soon.

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.

File Systems

To check the quota of different file systems, type *myquota* at the command line.

```
x-anvilusername@login03.anvil:[~] $ myquota
```

| Type | Location | Size | Limit | Use | Files | Limit | Use |
|----------|-----------------|---------|---------|-------|-------|--------|-------|
| home | x-anvilusername | 261.5MB | 25.0GB | 1% | - | - | - |
| scratch | anvil | 6.3GB | 100.0TB | 0.01% | 3k | 1,048k | 0.36% |
| projects | accountname1 | 37.2GB | 5.0TB | 0.73% | 403k | 1,048k | 39% |
| projects | accountname2 | 135.8GB | 5.0TB | 3% | 20k | 1,048k | 2% |

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
```

Transferring Files

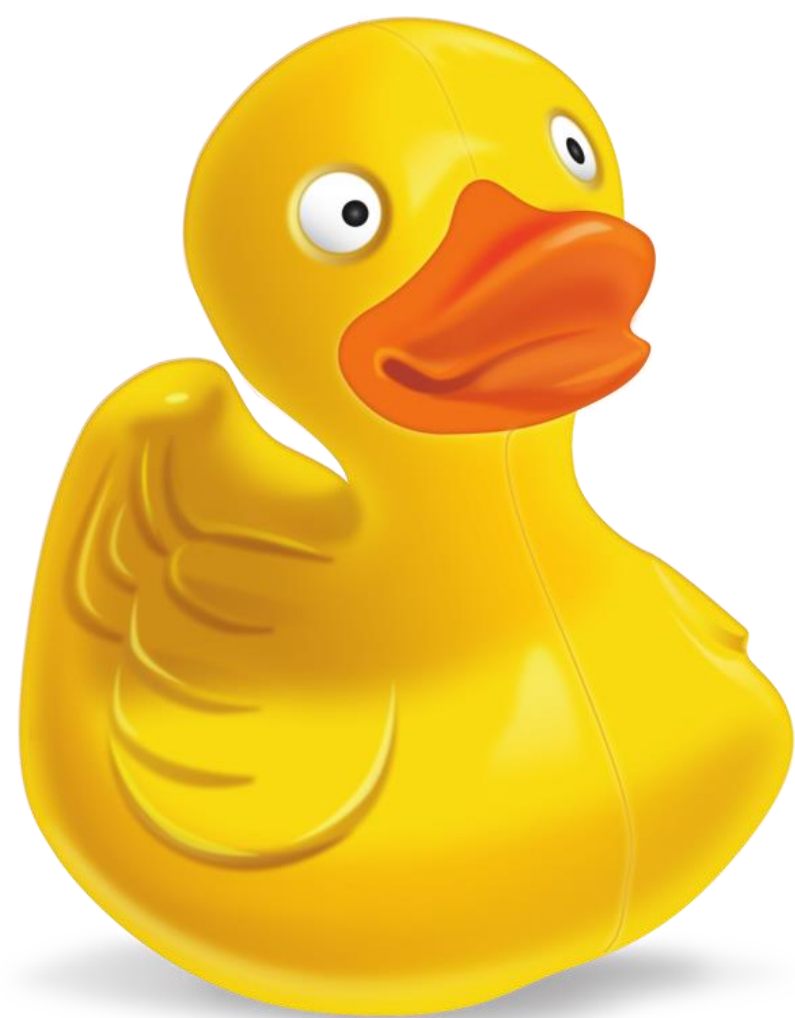
Users can transfer files between Anvil and Linux-based systems or Mac or windows terminal using either *scp* or *rsync* or *SFTP*.

- **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.

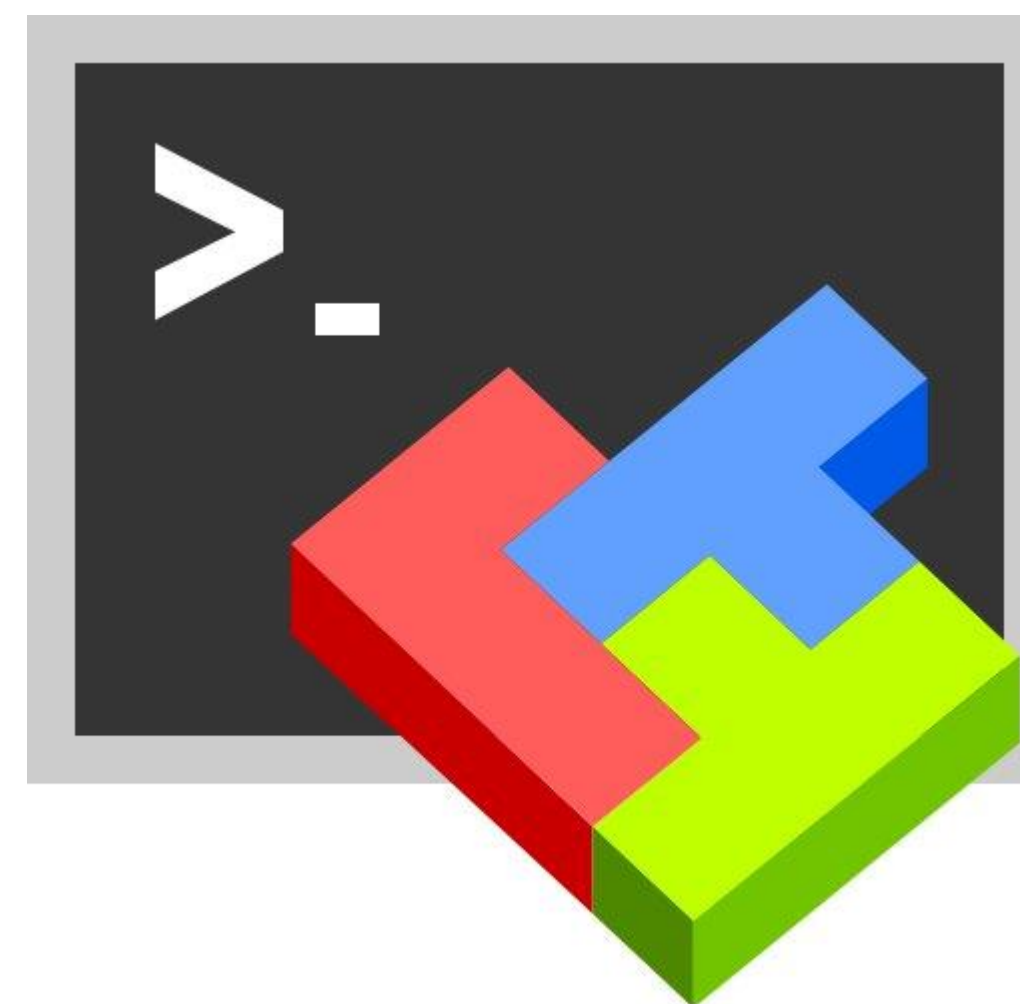
NOTE: **SSH Keys** is *required* for Rsync. Also make sure to use your anvil user name **x-anvilusername**:

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](http://www.rcac.purdue.edu/knowledge/anvil/storage/transfer/sftp)



[Cyberduck](#) for Mac OS X



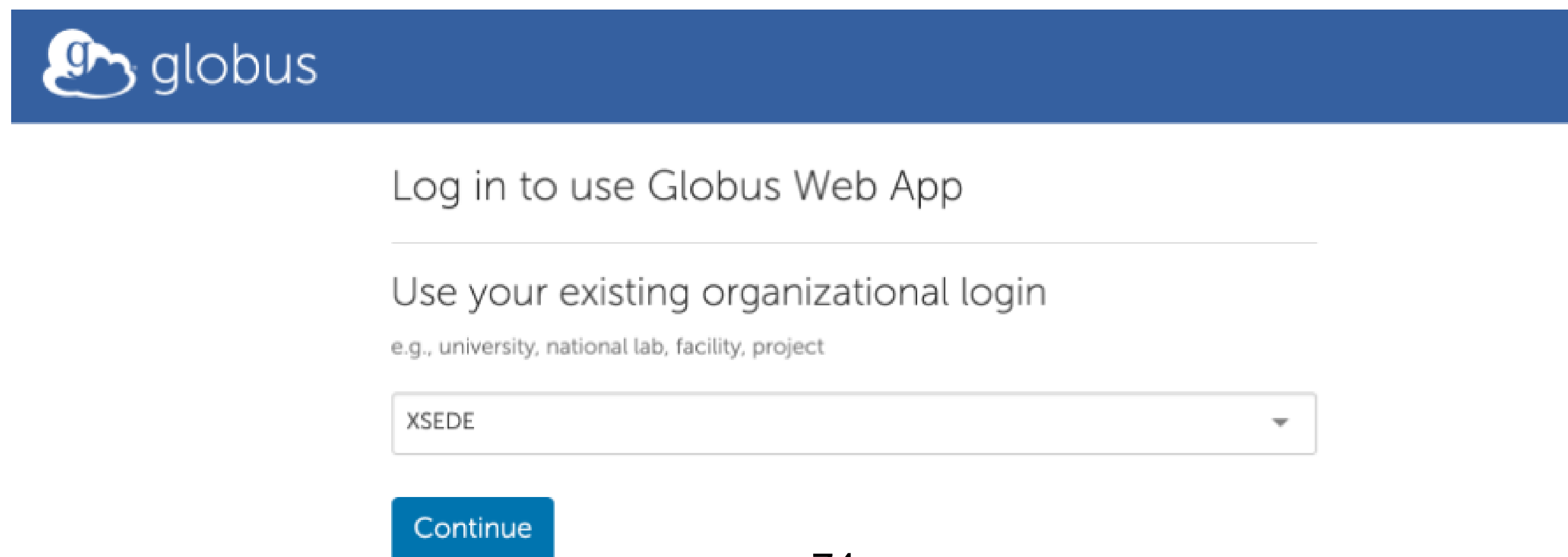
[MobaXterm](#) for Microsoft Windows

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>



The screenshot shows the Globus Web App login interface. At the top left is the Globus logo, which consists of a stylized 'g' inside a cloud shape followed by the word 'globus'. Below the logo, the text 'Log in to use Globus Web App' is displayed. Underneath this is a horizontal line, followed by the text 'Use your existing organizational login' and a smaller line of text 'e.g., university, national lab, facility, project'. A dropdown menu is shown with 'XSEDE' selected. At the bottom of the form is a blue button labeled 'Continue'.

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](http://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 <i>sstat</i> , <i>sacct</i> and <i>squeue</i> 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 |

Anvil

Purdue University is the home of Anvil, a powerful new supercomputer that provides advanced computing capabilities to support a wide range of computational and data-intensive research spanning from traditional high-performance computing to modern artificial intelligence applications.

FORGING THE FUTURE OF COMPUTING

Overview

Scientific Highlights

Documentation

Training

Advisory Board

Contact Us

News & Events

[Anvil Maintenance](#)

June 2, 2022 8:00am - 6:00pm EDT

Archive

Overview

Anvil, built through a \$10 million system acquisition from the [National Science Foundation \(NSF\)](#), and provides resources to the NSF's [Extreme Science and Engineering Discovery Environment \(XSEDE\)](#), which serves tens of thousands of researchers across the U.S., and in which Purdue has been a partner for the past nine years. Anvil entered production in February 2022 and will serve researchers for five years. Additional funding from the NSF will support Anvil's operations and user support.

The name "Anvil" reflects the Purdue Boilermakers' strength and workmanlike focus on producing results, and the Anvil supercomputer will enable important discoveries across many different areas of science and engineering. Anvil also will serve as an experiential learning laboratory for students to gain real-world experience using computing for their science, and for student interns to work with the Anvil team for construction and operation. We will be training the research computing practitioners of the future. Learn more about Anvil's mission in the [Anvil press release](#).

Anvil is funded under NSF award number 2005632. Carol Song is the principal investigator and project director. Preston Smith, executive director of Research Computing, Xiao Zhu, computational scientist and senior research scientist, and Rajesh Kalyanam, data scientist, software engineer, and research scientist, are all co-PIs on the project.

www.rcac.purdue.edu/anvil

Documentation



ARCHITECTURE

Look over the specs and architecture to get an idea of what Anvil can do.



GETTING STARTED

Learn how to set up an account and connect to the system.



SUBMITTING JOBS

Learn how to submit jobs, check status, and output.



SOFTWARE

See the initial list of pre-installed software.



DATA MANAGEMENT

Learn about our file system and the ways to transfer files.



POLICIES & TIPS & FAQs

Know what to expect and what we expect from you.

▼ Anvil User Guide

- > Overview of Anvil
- > Accessing the System
- > System Architecture

▼ Running Jobs

- > Accessing the Compute Nodes
- > Job Accounting
- > Slurm Partitions (Queues)
- > Batch Jobs
- > Interactive Jobs
- > Example Jobs

- > Managing and Transferring Files
- > Software
- > Policies, Helpful Tips and FAQs
- > Anvil Composable Subsystem

▼ Software

- > Module System
- > Compiling, performance, and optimization on Anvil
- > [Compiling Source code](#)
- > Provided Software
- ▼ Installing applications
 - > VASP
 - > 🔒 LAMMPS

Post-Survey



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.