

# Intermediate Level

## Introduction to Computing at CARC

1 hour version with Gaussian

Matthew Fricke

Version 0.1



# Goals

- 1) SLURM scheduler literacy
- 2) Example Gaussian
- We wont cover file transfer, storage systems, module system, conda, PBS. (These are all covered in depth in the video tutorials)

# Logging into Hopper



First login to the Linux **workstation** in front of you. Your CARC username is on the sign in sheet.

If you have logged in before use your **existing password**

Otherwise, your initial password is **Welcome2CARC**

This is an “important step” so don’t let me move on until you have logged in

# Logging into Hopper



```
ssh vanilla@hopper.alliance.unm.edu
```

Should prompt you for a password...

Don't let me move on until you are able to login.

Replace vanilla with your name (unless your last name is Ice)

# Logging into Hopper

Welcome to Hopper

Be sure to review the "Acceptable Use" guidelines posted on the CARC website.

For assistance using this system email [help@carc.unm.edu](mailto:help@carc.unm.edu).

Tutorial videos can be accessed through the CARC website: Go to <http://carc.unm.edu>, select the "New Users" menu and then click "Introduction to Computing at CARC".

Warning: By default home directories are world readable. Use the chmod command to restrict access.

Don't forget to acknowledge CARC in publications, dissertations, theses and presentations that use CARC computational resources:

"We would like to thank the UNM Center for Advanced Research Computing, supported in part by the National Science Foundation, for providing the research computing resources used in this work."

Please send citations to [publications@carc.unm.edu](mailto:publications@carc.unm.edu).

There are three types of slurm partitions on Hopper:

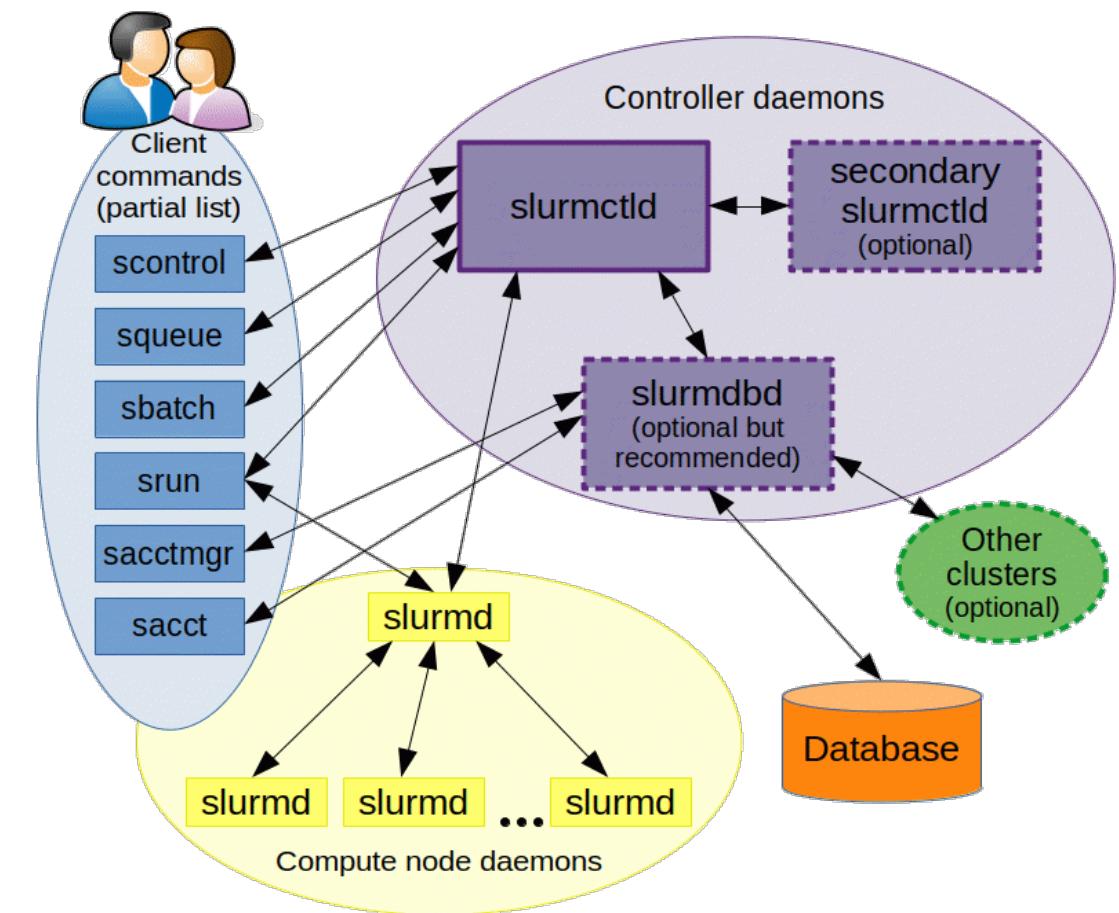
- 1) General - this partition is accessible by all CARC users.
- 2) Condo - preemptable scavenger queue available to all condo users. Your job must use checkpointing to use this queue or you will lose any work you have done if it is preempted by the partition's owner.
- 3) Named partitions - these partitions are available to condo users working under the grant/lab/center that purchased the associated hardware.

Type "qgrok" to get the status of the partitions.

---

Last login: Wed Jul 27 17:46:13 2022 from 129.24.246.68  
mfricke@hopper:~ \$

# Simple Linux Utility for Resource Management



ENJOY

# Slurm

SODA

IT'S HIGHLY ADDICTIVE!

VOTED #1 SOFT DRINK OF THE 31<sup>ST</sup> CENTURY!



SELENE MCKENRE

Slurm  
SOFT DRINK  
OF THE 31<sup>ST</sup> CENTURY

```
[vanilla@hopper ~]$ qgrok
```

queues	free	busy	offline	jobs	nodes	CPUUs	GPUUs	CPUUs/node	GPUUs/node	Memory/node	time_limit	CPU_limit
general	4	6	0	4	10	320	0	32	0	93G	2-00:00:00	64
debug	2	0	0	0	2	64	0	32	0	93G	4:00:00	8
condo	22	25	4	7	51	1632	28	32	2	93G-1.5T	2-00:00:00	192
bugs	2	0	0	0	2	64	0	32	0	93G	7-00:00:00	

```
vanilla@hopper:~ $ qgrok
```

queues	free	busy	offline	jobs	nodes	CPUs	GPUs
general	1	9	0	7	10	320	0
debug	2	0	0	0	2	64	0
condo	18	19	1	7	38	1216	8
bugs	0	2	0	0	2	64	0
pcnc	1	1					
pathogen	1	0					
tc	5	5					
gold	2	0					
fishgen	0	1					
neuro-hsc	8	6					
cup-ecs	0	2					
tid	0	1					
biocomp	0	1					
chakra	1	0					
pna	0	0					
totals:	19	28					

Open partitions for use by  
everyone with a CARC account.

Purchased by the Office for the  
Vice President for Research.

vanilla@hopper:~ \$ qgrok

queues	free	busy	offline	jobs	nodes	CPUs	GPUs
general	1	9	0	7	10	320	0
debug	2	0	0	0	2	64	0
condo	18	19	1	7	38	1216	8
bugs	0	2	0	0	2	64	0
pcnc	1	1	0	0	2	64	0
pathogen	1	0	0	0	1	32	0
tc	5	5	0	3	10	320	0
gold	2	0	0	0	2	64	0
fishgen	0	1	0	0	1	32	0
neuro-hsc	8	6	0	0	14	448	0
cup-ecs	0	2	0	2	2	64	4
tid	0	1	0	0	1	32	2
biocomp	0	1	0	0	1	32	1
chakra	1	0	0	0	1	32	1
pna	0	0	1	0	1	32	0
totals:	19	28	1	14	48	1536	8

```
vanilla@hopper:~ $ qgrok
```

queues	free	busy	offline	jobs	nodes	CPUs	GPUs
general	1	9	0	7	10	320	0
debug	2	0	0	0	2	64	0
condo	18	19	1	7	38	1216	8
bugs	0	2	0	0	2	64	0
pcnc	1	1					
pathogen	1	0					
tc	5	5					
gold	2	0					
fishgen	0	1					
neuro-hsc	8	6					
cup-ecs	0	2					
tid	0	1					
biocomp	0	1					
chakra	1	0					
pna	0	0					
totals:	19	28					

## Private partitions

- Reserved for use by the purchaser.
- Request access by emailing [support@carc.unm.edu](mailto:support@carc.unm.edu) and CC the partition owner.

```
mfricke@hopper:~ $ qgrok
```

queues	free	busy	offline	jobs	nodes	CPUs	GPUs
general	1	9	0	7	10	320	0
debug	2	0	0	0	2	64	0
<b>condo</b>	<b>18</b>	<b>19</b>	<b>1</b>	<b>7</b>	<b>38</b>	<b>1216</b>	<b>8</b>
bugs	0	2	0	0	2	64	0
pcnc	1	1					
pathogen	1	0					
tc	5	5					
gold	2	0					
fishgen	0	1					
neuro-hsc	8	6					
cup-ecs	0	2					
tid	0	1					
biocomp	0	1					
chakra	1	0					
pna	0	0					
totals:	19	28					

## Condo “scavenger” partition

- Allows you to use compute nodes purchased by another group that are currently idle.
- May be interrupted at any time if the owners start to use it.

```
[vanilla@hopper ~]$ quotas
```

```
Home Directory (/users/vanilla):
```

```
quota: Cannot resolve mountpoint path /root/.spack: Permission denied
```

```
Disk quotas for user vanilla (uid 659):
```

Filesystem	space	quota	limit	grace	files	quota	limit	grace
chama:/home/homes	1527M	100G	200G		14913	4295m	4295m	

---

```
Centerwide user scratch (/carc/scratch/users/mfricke)
```

```
Quota information for storage pool Default (ID: 1):
```

user/group	size	chunk	files	
name	used	hard	used	hard
mfricke	592.71 GiB	1024.00 GiB	32784	unlimited

---

```
Centerwide scratch quota for project mfricke2016174 (/carc/scratch/projects/mfricke2016174)
```

```
Quota information for storage pool Default (ID: 1):
```

user/group	size	chunk	files	
name	used	hard	used	hard
mfricke2016174	190.97 GiB	1024.00 GiB	23704	unlimited

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      2  idle  hopper[011-012]
```

sinfo reports information about  
partitions

```
[vanilla@hopper ~]$ sinfo --partition debug
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug        up    4:00:00      2  idle  hopper[011-012]
```

The debug queues are intended  
for testing your programs.

And for interactive jobs.

```
[vanilla@hopper ~]$ sinfo --partition debug  
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST  
debug        up    4:00:00      2  idle hopper[011-012]
```



Name

```
[vanilla@hopper ~]$ sinfo --partition debug  
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST  
debug        up    4:00:00      2  idle  hopper[011-012]
```



You can run a “job” for up to 4 hrs.

```
[vanilla@hopper ~]$ sinfo --partition debug  
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST  
debug        up    4:00:00      2  idle  hopper[011-012]
```



There are two nodes in this partition.

```
[vanilla@hopper ~]$ sinfo --partition debug  
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST  
debug        up    4:00:00      2  idle  hopper[011-012]
```



The names of the nodes in the partition

```
[vanilla@hopper ~]$ sinfo --partition debug  
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST  
debug        up    4:00:00      2  idle  hopper[011-012]
```



The names of the nodes in the partition

```
[vanilla@hopper ~]$ sinfo --partition general
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
general*      up   2-00:00:00      9  alloc  hopper[001-009]
general*      up   2-00:00:00      1  idle   hopper010
```



Hopper001 through 009 are running jobs.

Hopper010 is waiting to be used.

```
[vanilla@hopper ~] $ hostname  
hopper  
[vanilla@hopper ~] $
```



Running on the Head Node.  
The head node's name is “hopper”.

```
[vanilla@hopper ~]$ hostname  
hopper
```

```
[vanilla@hopper ~]$ man hostname
```

```
[vanilla@hopper ~]$ hostname  
hopper  
[vanilla@hopper ~]$ man hostname  
(‘q’ to quit)
```

```
[vanilla@hopper ~]$ man man  
(‘q’ to quit)
```

```
[vanilla@hopper ~]$ man sinfo
```

sinfo(1)  
Commands

Slurm  
sinfo(1)

## NAME

sinfo - View information about Slurm nodes and partitions.

## SYNOPSIS

`sinfo [OPTIONS...]`

## DESCRIPTION

sinfo is used to view partition and node information for a system running Slurm

## OPTIONS

`-a, --all`

Display information about all partitions. This causes information to be displayed about partitions that are configured as hidden and partitions that are unavailable to the user's group.

```
[vanilla@hopper ~]$ sinfo --all
```

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	NODELIST
general*	up	2-00:00:00	9	alloc	hopper[001-009]
general*	up	2-00:00:00	1	idle	hopper010
debug	up	4:00:00	2	idle	hopper[011-012]
condo	up	2-00:00:00	1	down*	hopper045
condo	up	2-00:00:00	3	mix	hopper[018-020]
condo	up	2-00:00:00	16	alloc	hopper[013-015,028-036,049-052]
condo	up	2-00:00:00	18	idle	hopper[016-017,021-027,037-044,053]
bugs	up	7-00:00:00	2	alloc	hopper[013-014]
pcnc	up	7-00:00:00	1	alloc	hopper015
pcnc	up	7-00:00:00	1	idle	hopper016
pathogen	up	7-00:00:00	1	idle	hopper017
tc	up	7-00:00:00	3	mix	hopper[018-020]
tc	up	7-00:00:00	2	alloc	hopper[029-030]
tc	up	7-00:00:00	5	idle	hopper[021-025]
gold	up	7-00:00:00	2	idle	hopper[026-027]
fishgen	up	7-00:00:00	1	alloc	hopper028
neuro-hsc	up	7-00:00:00	6	alloc	hopper[031-036]
neuro-hsc	up	7-00:00:00	8	idle	hopper[037-044]
cup-ecs	up	7-00:00:00	2	alloc	hopper[049-050]
tid	up	7-00:00:00	1	alloc	hopper051
biocomp	up	7-00:00:00	1	alloc	hopper052
chakra	up	7-00:00:00	1	idle	hopper053
pna	up	7-00:00:00	1	down*	hopper045

```
[vanilla@hopper ~]$ srun --partition debug hostname
```



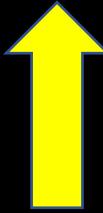
Tell slurm to run a program  
on a compute node...

```
[vanilla@hopper ~]$ srun --partition debug hostname
```



Run the program on a  
compute node in the  
debug partition.

```
[vanilla@hopper ~]$ srun --partition debug hostname
```



The program  
to run.

```
[vanilla@hopper ~]$ srun --partition debug hostname
srun: Account not specified in script or
~/.default_slurm_account, using latest project
You have not been allocated GPUs. To request GPUs,
use the -G option in your submission script.
hopper011
```

```
[vanilla@hopper ~]$ squeue
```

```
[vanilla@hopper ~]$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIM	QOSMaxCpuPerUserLimit
4314	general	PRE	erowland	PD	0:00	2 (QOSMaxCpuPerUserLimit)
4315	general	PRE	erowland	PD	0:00	2 (QOSMaxCpuPerUserLimit)
4317	general	PRE	erowland	PD	0:00	2 (QOSMaxCpuPerUserLimit)
4318	general	PRE	erowland	PD	0:00	2 (QOSMaxCpuPerUserLimit)

PD means programs  
that are waiting their  
turn.

Shows you what the slurm  
scheduler is doing right now.

Here we can see that user  
'erowland' has a lot of programs  
waiting to run.

```
[vanilla@hopper ~]$ squeue
```

The reason these jobs are not running is that ‘erowland’ is already using the maximum number of CPUs they are allowed.

```
[vanilla@hopper ~]$ squeue -t R --all
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
4405	condo	2ndMA	mfricke	R	1-07:48:30	6	hopper[031-036]
5208	condo	NN	kgu	R	5:48:49	1	hopper015
5210	condo	NN	kgu	R	6:30:13	1	hopper014
5209	condo	NN	kgu	R	6:31:13	1	hopper013
5206	condo	NN	kgu	R	6:32:13	1	hopper051
5207	condo	NN	kgu	R	6:32:13	1	hopper052
5205	condo	NN	kgu	R	6:32:43	1	hopper028
4595	cup-ecs	golConfi	aalasand	R	2-06:51:59	1	hopper050
4594	cup-ecs	golConfi	aalasand	R	2-06:52:03	1	hopper049
5120	general	jupyterh	jacobm	R	11:45:47	1	hopper007
4313	general	PRE	erowland	R	1:17:29	2	hopper[003-004]
5111	general	1stMA	mfricke	R	11:15:28	2	hopper[005-006]
5025	general	c2n	jxzuo	R	1:50	1	hopper001
5024	general	c2n	jxzuo	R	31:28	1	hopper002
5203	general	NN	kgu	R	6:37:50	1	hopper009
5201	general	NN	kgu	R	6:38:14	1	hopper008
4390	tc	UCsTpCyd	lepluart	R	2-15:18:18	3	hopper[018-020]
5198	tc	NN	kgu	R	6:40:19	1	hopper030
5196	tc	NN	kgu	R	6:40:31	1	hopper029

```
[vanilla@hopper ~]$ srun --partition debug --ntasks 2 hostname
srun: Account not specified in script or
~/.default_slurm_account, using latest project
You have not been allocated GPUs. To request GPUs, use the -G
option in your submission script.
hopper011
hopper011
```

```
[vanilla@hopper ~]$ srun --partition debug --ntasks 2 hostname
srun: Account not specified in script or
~/.default_slurm_account, using latest project
You have not been allocated GPUs. To request GPUs, use the -G
option in your submission script.
hopper011
hopper011
```

You ran two **copies** of your program.

**ntasks** is the number of copies to run.

```
[vanilla@hopper ~]$ srun --partition debug --ntasks 8 hostname
srun: Account not specified in script or
~/.default_slurm_account, using latest project
hopper011
hopper011
hopper011
hopper011
You have not been allocated GPUs. To request GPUs, use the -G
option in your submission script.
hopper011
hopper011
hopper011
hopper011
hopper011
```

You ran eight **copies** of your program.

**ntasks** is the number of copies to run.

```
[vanilla@hopper ~]$ srun --partition debug --ntasks 8 hostname
srun: Account not specified in script or
~/.default_slurm_account, using latest project
hopper011
hopper011
hopper011
hopper011
You have not been allocated GPUs. To request GPUs, use the -G
option in your submission script.
hopper011
hopper011
hopper011
hopper011
hopper011
```

**By default, each task (copy of your program) is allowed to use one CPU.**

**Many programs are able to use more than one CPU at a time.**

```
[vanilla@hopper ~]$ srun --partition debug --ntasks 2 --cpus-per-task 2 hostname  
srun: Account not specified in script or ~/.default_slurm_account, using latest project  
You have not been allocated GPUs. To request GPUs, use the -G option in your submission  
script.  
hopper011  
hopper011
```

Here we are telling SLURM to run 2 copies of our program and let each copy of our program use 2 CPUs.

```
[vanilla@hopper ~]$ srun --partition debug --nodes 2 -ntasks-per-node 4 hostname  
srun: Account not specified in script or ~/.default_slurm_account, using  
latest project  
hopper012  
You have not been allocated GPUs. To request GPUs, use the -G option in  
your submission script.  
hopper012  
hopper011  
hopper011  
hopper012  
hopper012  
hopper011  
hopper011
```

**Here we are telling SLURM to run 4 copies of our program on 2 different compute nodes.**

**This is useful when our programs need a bigger share of the compute node.**

```
[vanilla@hopper ~]$ srun --partition debug --nodes 2  
--ntasks-per-node 2 --cpus-per-task 2 hostname  
srun: Account not specified in script or  
~/.default_slurm_account, using latest project  
hopper011  
You have not been allocated GPUs. To request GPUs, use  
the -G option in your submission script.  
hopper011  
hopper012  
hopper012
```

And we can combine all three.

```
[vanilla@hopper ~]$ srun --partition debug --mem 4G  
--nodes 2 --ntasks-per-node 2 --cpus-per-task 2  
hostname
```

```
srun: Account not specified in script or  
~/.default_slurm
```

```
hopper012
```

```
hopper012
```

```
You have not be
```

```
the -G option i
```

```
hopper011
```

```
Hopper011
```

**And we can specify how much  
memory we want.**

**--mem 4G means give me 4  
gigabytes of memory per node.**

```
[vanilla@hopper ~]$ srun --partition debug --mem 4G  
--nodes 2 --ntasks-per-node 2 --cpus-per-task 2  
hostname  
srun: Account not specified in script or  
~/.default_slurm  
hopper012  
hopper012  
You have not been  
the -G option in  
hopper011  
Hopper011
```

## Why does all this matter?

The purpose of SLURM is to provide you the hardware your programs need.

So you have to understand what those requirements are really well.

```
[vanilla@hopper ~]$ srun --partition debug --mem 4G  
--nodes 2 --ntasks-per-node 2 --cpus-per-task 2 hostname  
srun: Account not specified in script or  
~/.default_slurm_account using latest project  
hopper012  
hopper012  
You have not been assigned to a project.  
the -G option is required.  
hopper011  
Hopper011
```

- 1) Can my program use multiple CPUs?**
- 2) How much memory does my program need?**
- 3) Can my program use multiple compute nodes (MPI\*, GNU Parallel\*)?**
- 4) Can my program use GPUs?**

```
[vanilla@hopper ~]$ srun --partition debug --mem 4G  
--nodes 2 --ntasks-per-node 2 --cpus-per-task 2 hostname  
srun: Account not specified in script or  
~/.default_slurm_account using latest project  
hopper012  
hopper012  
You have not been  
the -G option in  
hopper011  
Hopper011
```

This command is getting pretty long.

We can use **shell scripts** to automate  
all this in **batch mode**.

# Interactive vs Batch Mode

## Interactive Mode

- Everything so far has been interactive. You request hardware, run your program, and get the output on your screen right away.

## Batch Mode

- Most programs at an HPC center are run in “batch” mode.
- Batch mode means we write a shell script that the SLURM scheduler runs for us. The script requests hardware just like we did with salloc and then runs the commands in the script.
- Whatever would have been written to the screen is saved to a file instead.

```
[vanilla@hopper ~]$ git clone https://lobogit.unm.edu/CARC/workshops.git
Cloning into 'workshops'...
remote: Enumerating objects: 132, done.
remote: Counting objects: 100% (75/75), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 132 (delta 33), reused 74 (delta 32), pack-reused 57
Receiving objects: 100% (132/132), 57.58 KiB | 3.60 MiB/s, done.
Resolving deltas: 100% (51/51), done.
[vanilla@hopper ~]$ cd workshops/intro_workshop
```

Rather than make you write shell scripts lets just download some we wrote for this workshop...

```
[vanilla@hopper intro_workshop]$ pwd  
/users/vanilla/workshops/intro_workshop  
[vanilla@hopper intro_workshop]$ cat slurm/workshop_example1.sh  
#!/bin/bash  
#SBATCH --partition debug  
#SBATCH --ntasks 4  
#SBATCH --time 00:05:00  
#SBATCH --job-name ws_example  
#SBATCH --mail-user your_username@unm.edu  
#SBATCH --mail-type ALL
```

```
srun hostname
```

Let's take a look at the **workshop\_example.sh** script in the slurm directory...

```
[vanilla@hopper intro_workshop]$ sbatch slurm/workshop_example1.sh  
sbatch: Account not specified in script or ~/.default_slurm_account,  
using latest project  
Submitted batch job 5252  
[vanilla@hopper intro_workshop]$
```

We **submit** our slurm  
shell script with the  
**sbatch** command.

```
[vanilla@hopper intro_workshop]$ sbatch slurm/workshop_example1.sh  
sbatch: Account not specified in script or ~/.default_slurm_account,  
using latest project  
Submitted batch job 5252  
[vanilla@hopper intro_workshop]$
```

Notice that the only output we get is a job id.

This indicates that the script was successfully sent to the scheduler.

The commands in the script will run as soon as the hardware requested is available.

We submit our slurm shell script with the sbatch command.

# Workflow

Head Node

User 1

Program A

Script A

User 2

Program B

Script B

Compute Node 01

Compute Node 02

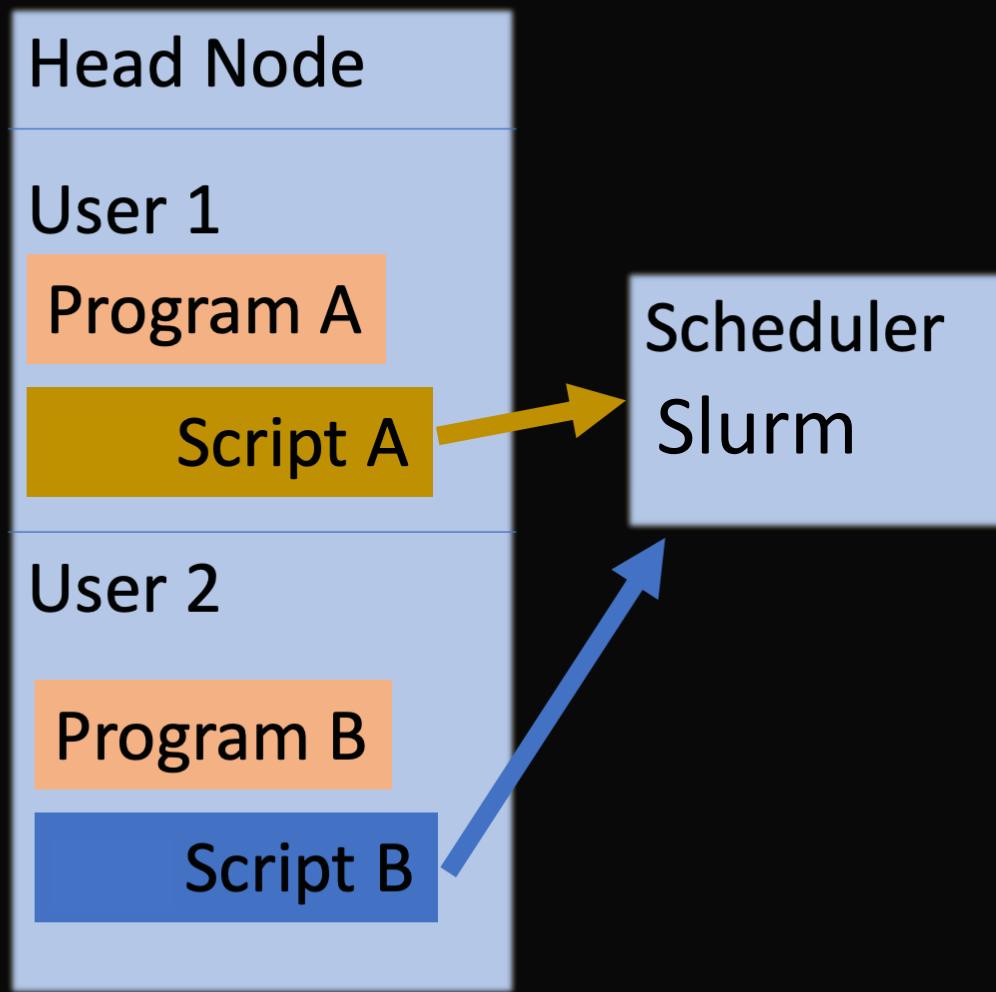
Compute Node 03

Compute Node 04

Compute Node 05

Shared filesystems – All nodes can access the same programs and write output

# Workflow



Compute Node 01

Compute Node 02

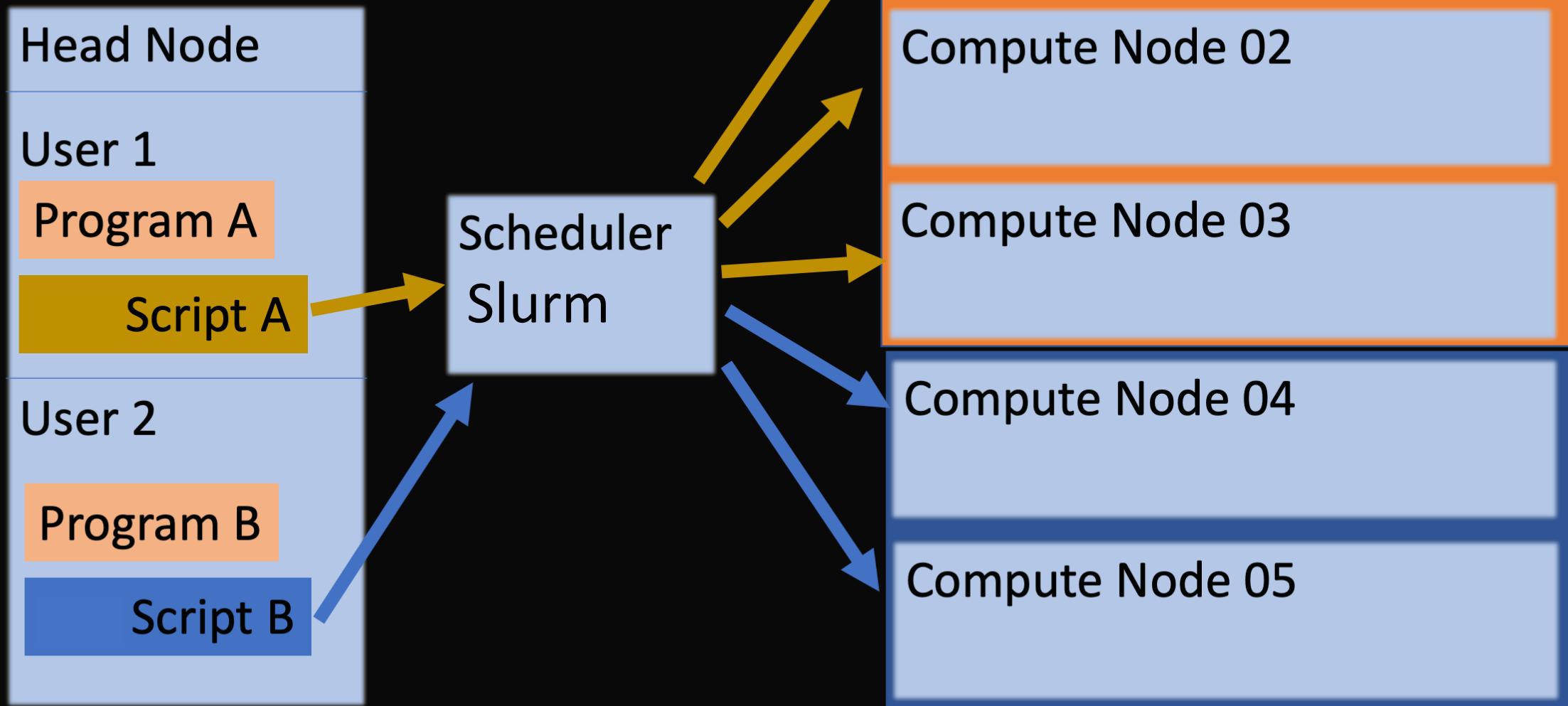
Compute Node 03

Compute Node 04

Compute Node 05

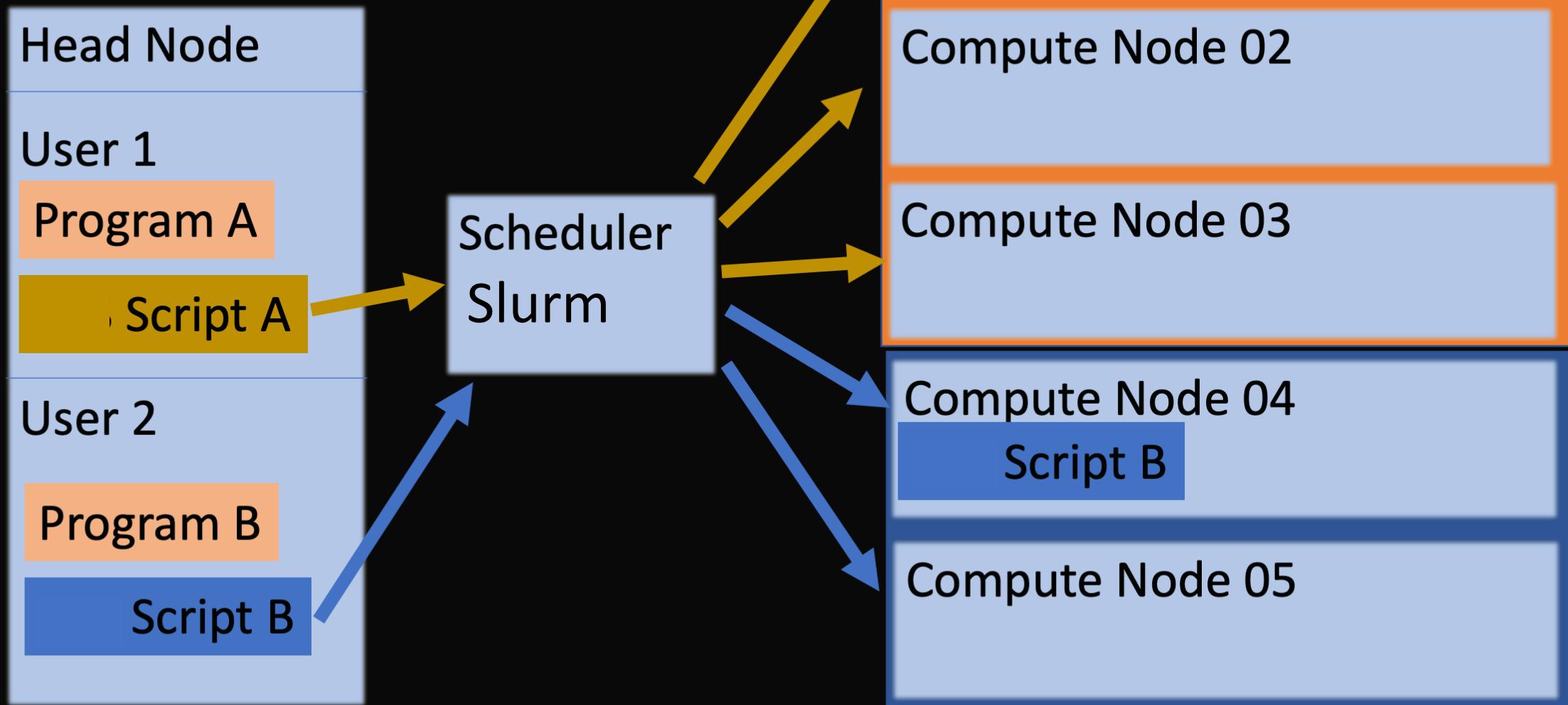
Shared filesystems – All nodes can access the same programs and write output

# Workflow



Shared filesystems – All nodes can access the same programs and write output

# Workflow



Shared filesystems – All nodes can access the same programs and write output

# Workflow

We need something in the script to run the program on all the nodes. E.g. srun.

Head Node

User 1

Program A

Script A

User 2

Program B

Script B

Scheduler  
Slurm

Compute Node 01

Script A

Program A

Compute Node 02

Program A

Program A

Compute Node 03

Program A

Program A

Compute Node 04

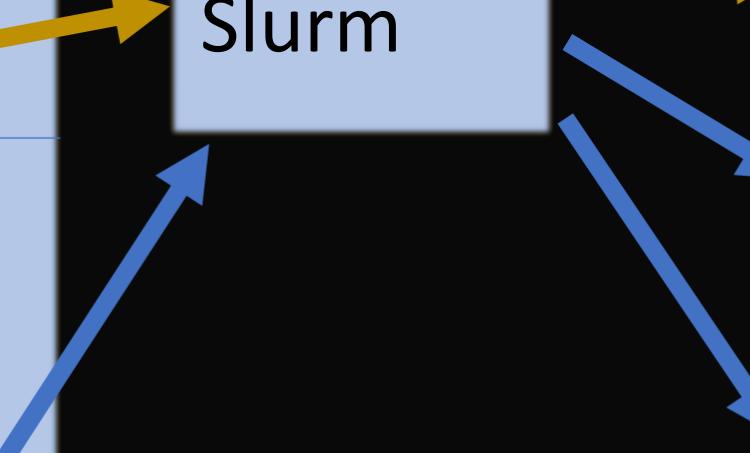
Script B

Program B

Compute Node 05

Program B

Program B



Shared filesystems – All nodes can access the same programs and write output

```
[vanilla@hopper intro_workshop]$ ls  
code  data  pbs  slurm  slurm-5252.out
```

The **hostname** command is very fast so everyone's job should finish in a few seconds.

When it is finished you will have a new file named slurm-{your job id}.out.



```
[vanilla@hopper intro_workshop]$ ls  
code  data  pbs  slurm  slurm-5252.out
```



When it is finished you will have a new file named slurm-{your job id}.out.

```
[vanilla@hopper intro_workshop]$ cat slurm-5252.out  
hopper011
```

General  
Purpose  
Computational  
Chemistry  
Package



We will design a molecule using Avogadro and then optimize its geometry with Gaussian

Favorites



Avogadro2

Accessories



IcedTea-Web Policy Editor

Documentation



OpenJDK 1.8.0 for x86\_64 Policy ...

Internet

Office

Programming

Sound &amp; Video

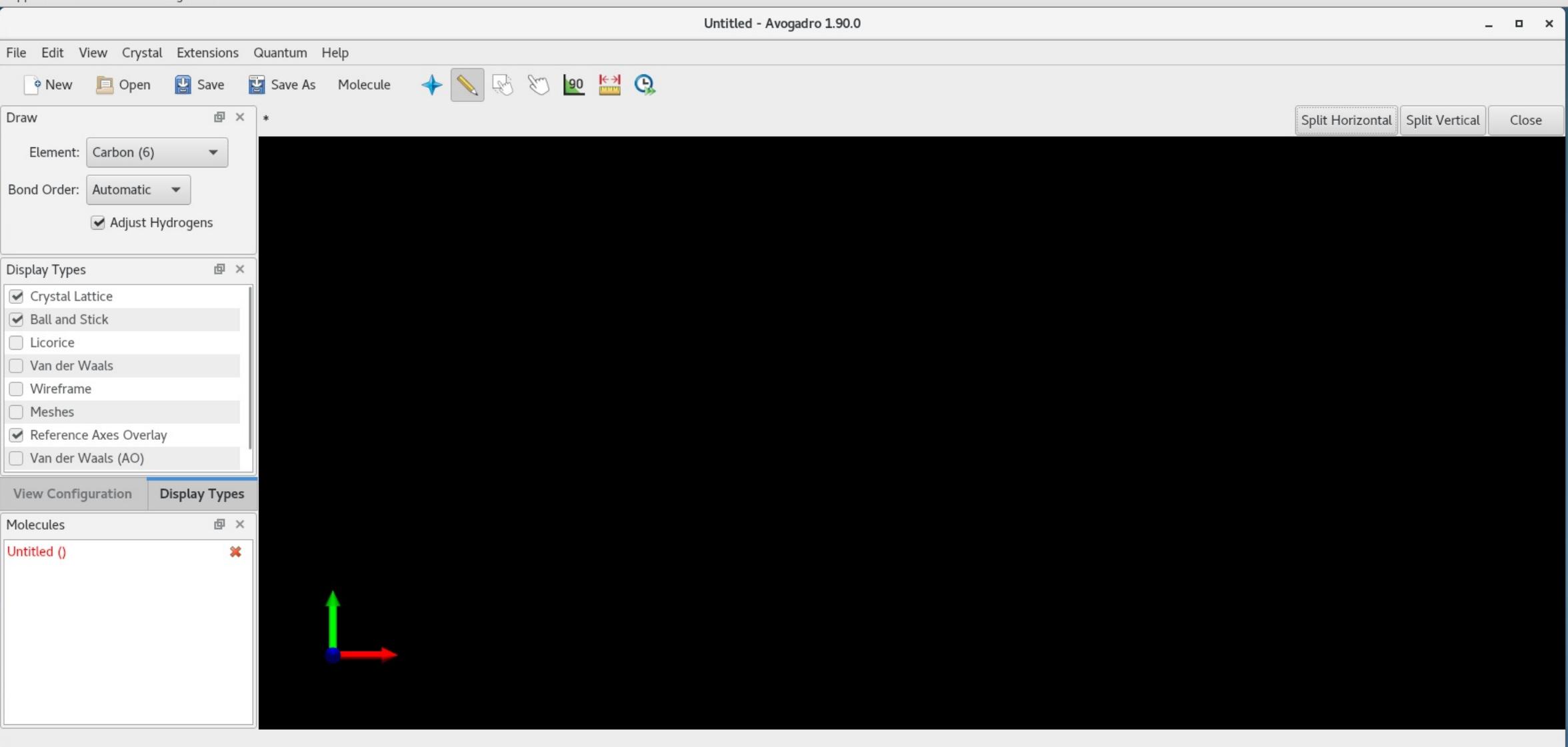
Sundry

System Tools

Utilities

Other

7  
C E N T O S



CENTOS

## Untitled\* - Avogadro 1.90.0

File Edit View Crystal Extensions Quantum Help



Draw

Element: Carbon (6)

Bond Order: Automatic

 Adjust Hydrogens

Display Types

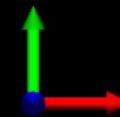
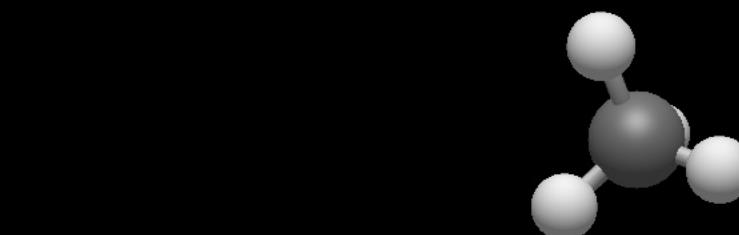
- Crystal Lattice
- Ball and Stick
- Licorice
- Van der Waals
- Wireframe
- Meshes
- Reference Axes Overlay
- Van der Waals (AO)

View Configuration

Display Types

Molecules

Untitled (CH4)



CENTOS

File Edit View Crystal Extensions Quantum Help



## Draw

Element: Carbon (6)

Bond Order: Automatic

 Adjust Hydrogens

## Display Types

- Crystal Lattice
- Ball and Stick
- Licorice
- Van der Waals
- Wireframe
- Meshes
- Reference Axes Overlay
- Van der Waals (AO)

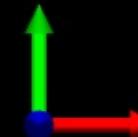
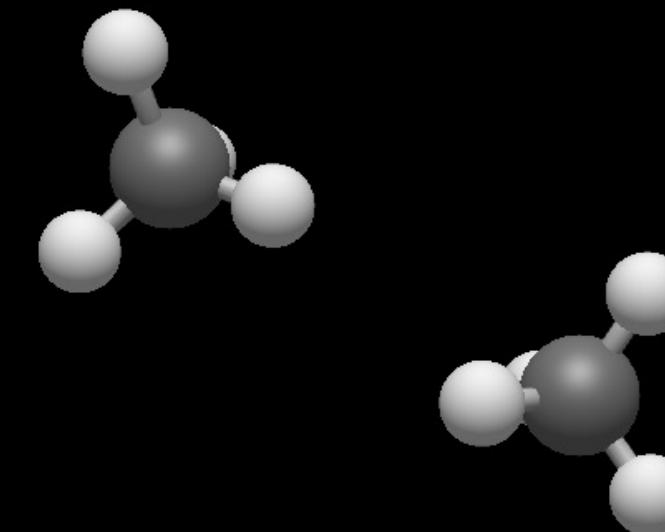
View Configuration

Display Types

## Molecules

Untitled (C2H8)

Split Horizo



Edit View Crystal Extensions Quantum Help

New Open Save Save As Molecule



Element: Carbon (6)

Order: Automatic

Adjust Hydrogens

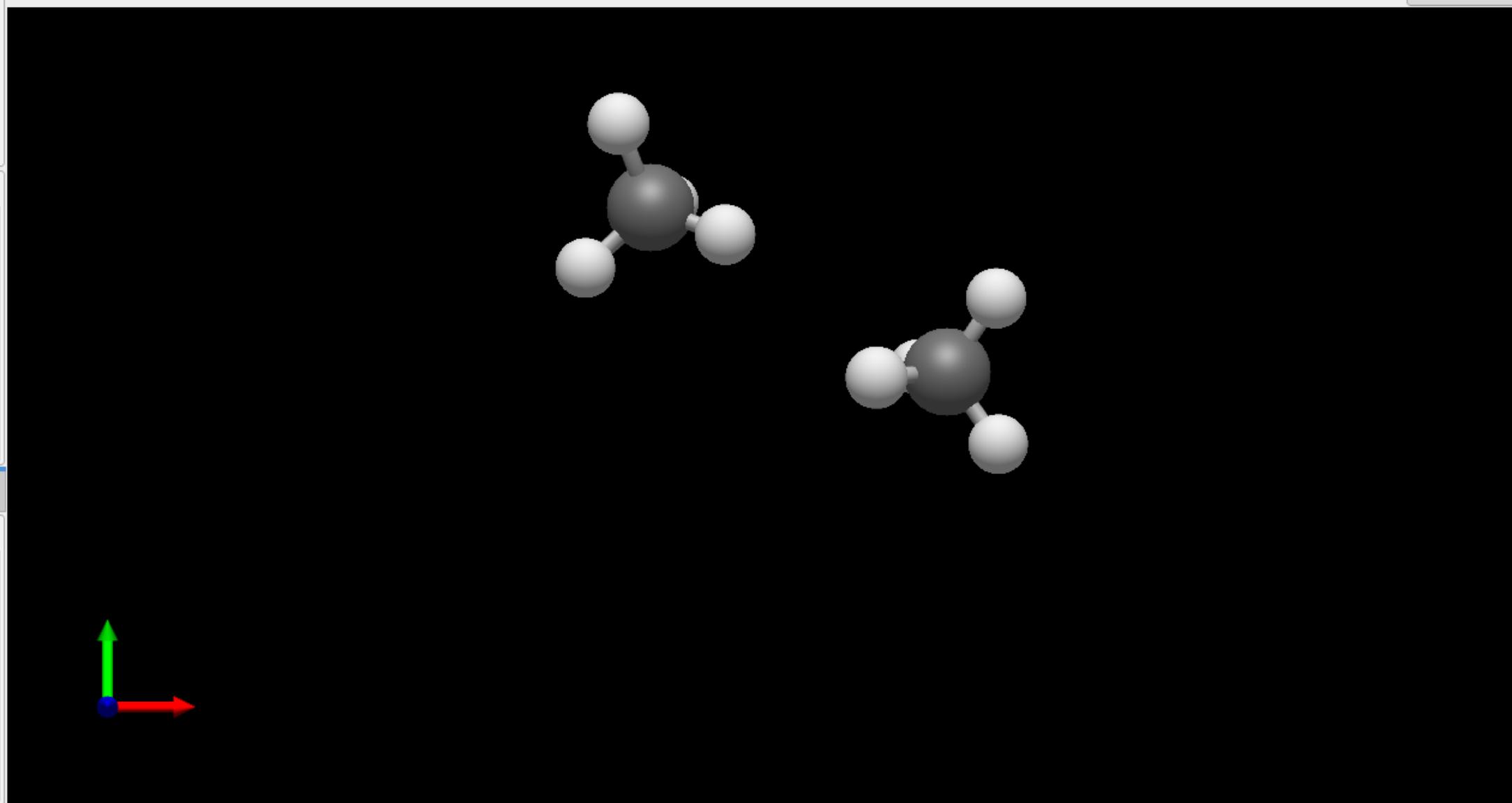
Types

- Crystal Lattice
- Sticks and Stick
- Wireframe
- Surface
- Radius
- Radius
- Reference Axes Overlay
- Radius (AO)

Configuration Display Types

States

1 (C<sub>2</sub>H<sub>8</sub>)





ent: Oxygen (8) ▾

der: Automatic ▾

### Adjust Hydrogens

types

### General Lattice

## Hand Stick

100

11

der Waals

## frame

10

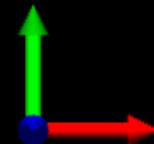
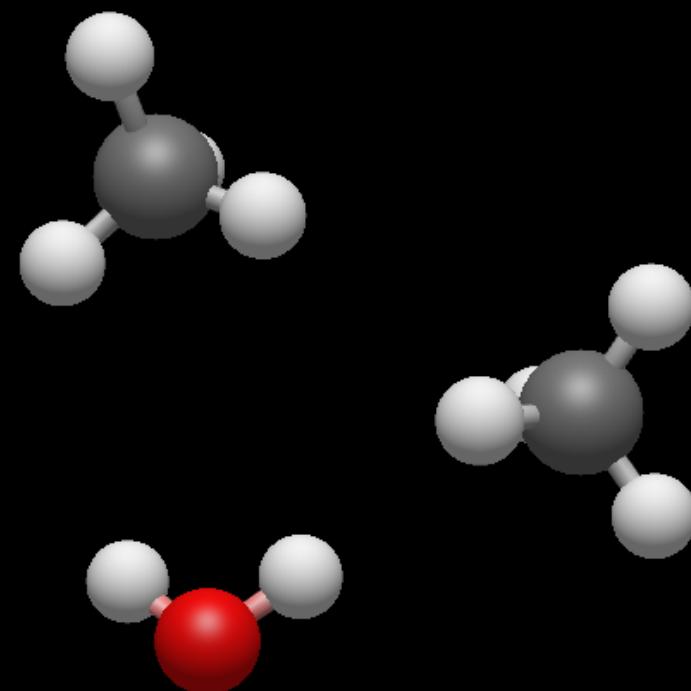
ference A

der Waals (AO)

## Display Types

(C<sub>2</sub>H<sub>10</sub>O)

✗



File View Crystal Extensions Quantum Help

Open Save Save As Molecule



Current: Chlorine (17)

Order: Automatic

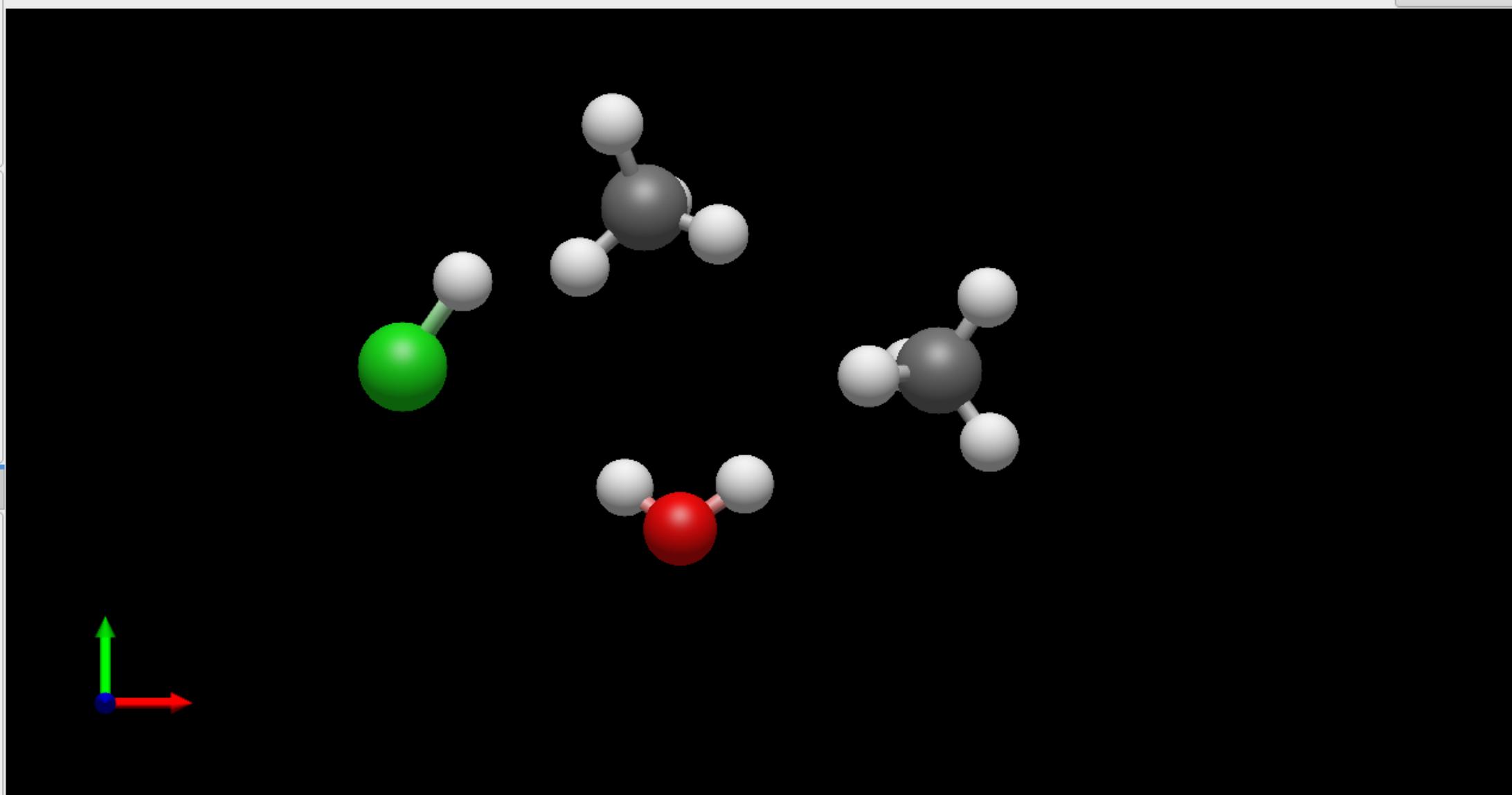
Adjust Hydrogens

Display Types

- Ball Lattice
- Bond Stick
- Wireframe
- Space Filled
- Radius
- Reference Axes Overlay
- Outer Waals (AO)

Configuration Display Types

C2H11OCl





Draw \*

Element: Chlorine (17)

Bond Order: Automatic

Adjust Hydrogens

Display Types

Crystal Lattice

Ball and Stick

Licorice

Van der Waals

Wireframe

Meshes

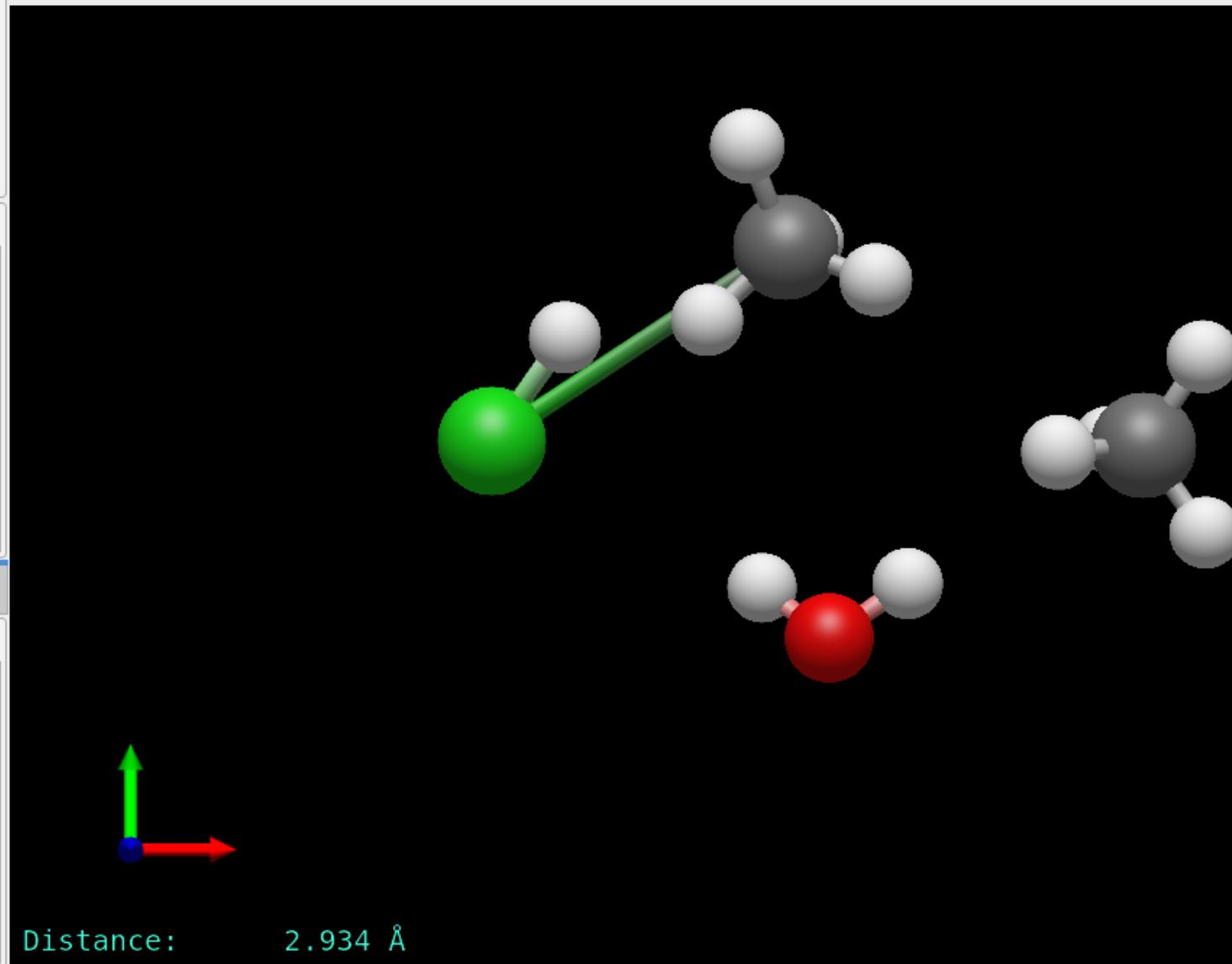
Reference Axes Overlay

Van der Waals (AO)

View Configuration Display Types

Molecules

Untitled (C<sub>2</sub>H<sub>11</sub>OCl) X



Crystal Extensions Quantum Help

Open Save Save As Molecule

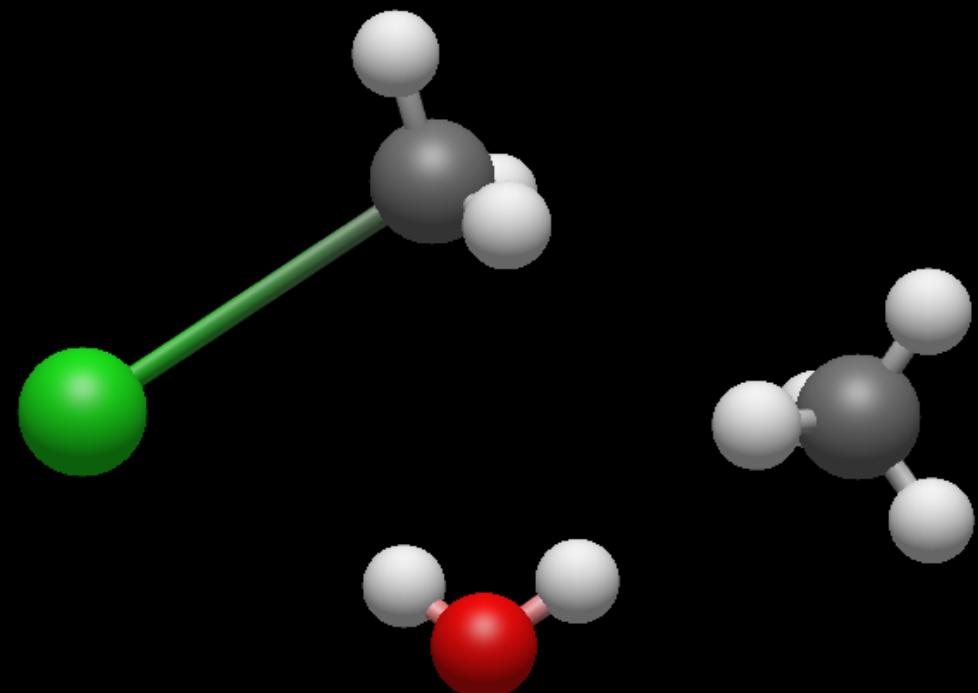


Split Horizontal Split

\*  
chlorine (17)

Aromatic  
Aromatic  
Single  
Double  
Triple  
S  
Bases Overlay  
AOs (AO)  
Display Types

Cl)



File Edit View Crystal Extensions Quantum Help



Draw

Element: Chlorine (17)

Bond Order: Double

 Adjust Hydrogens

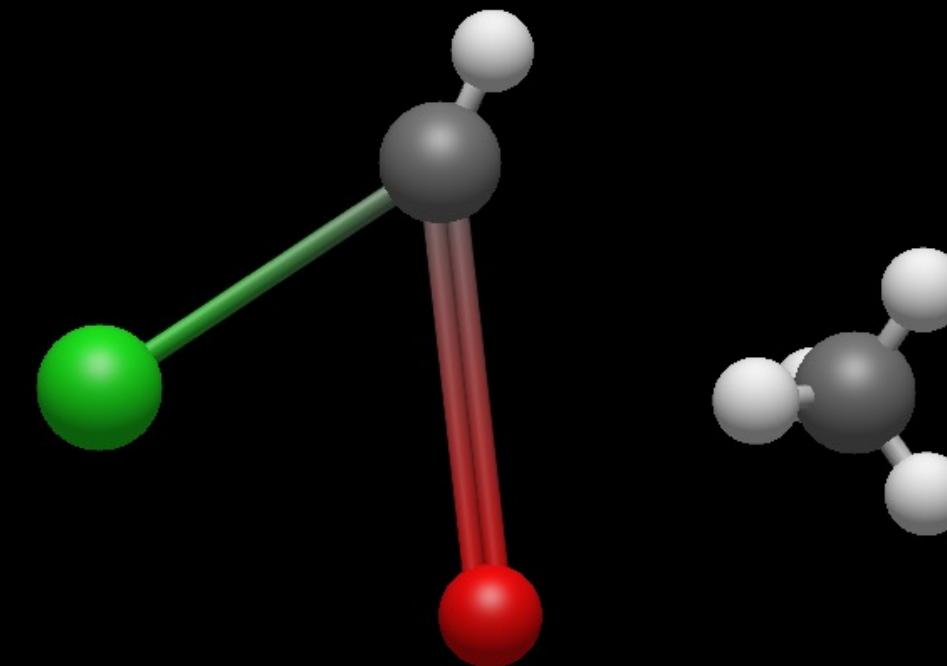
Display Types

 Crystal Lattice Ball and Stick Licorice Van der Waals Wireframe Meshes Reference Axes Overlay Van der Waals (AO)

View Configuration

Display Types

Molecules

Untitled (C<sub>2</sub>H<sub>5</sub>OCl)

Split H

File Edit View Crystal Extensions Quantum Help



raw \*  
Element: Chlorine (17)

ond Order: Single  
 Adjust Hydrogens

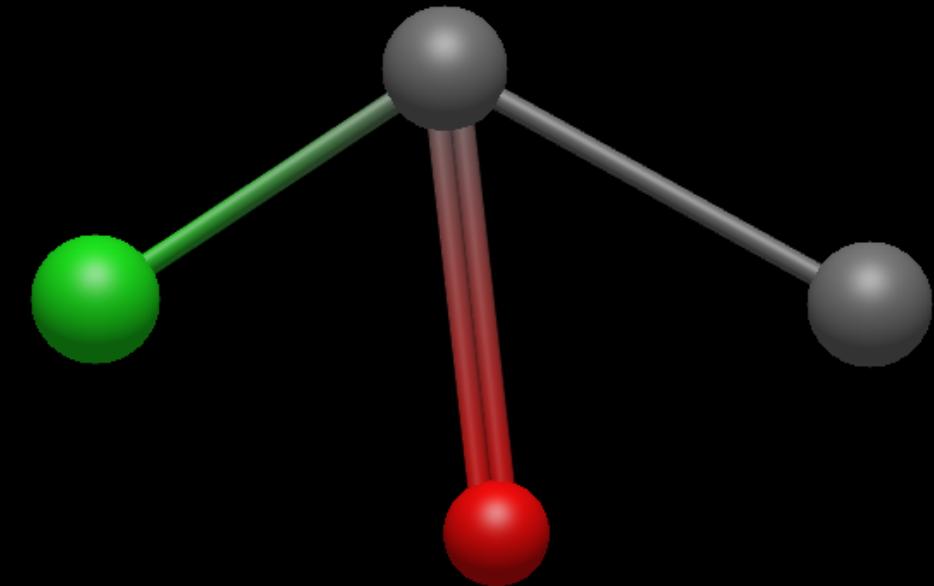
Display Types

- Crystal Lattice
- Ball and Stick
- Licorice
- Van der Waals
- Wireframe
- Meshes
- Reference Axes Overlay
- Van der Waals (AO)

View Configuration Display Types

Molecules

Untitled (C<sub>2</sub>OCl) \*



Split H

Cancel Name AcetylChloride.cml Save

File Edit View Crystal Extensions

New Open Save

Draw Element: Chlorine (17)

Bond Order: Single  Adjust Hydrogens

Display Types

Crystal Lattice  
 Ball and Stick  
 Licorice  
 Van der Waals  
 Wireframe  
 Meshes  
 Reference Axes Overlay  
 Van der Waals (AO)

Molecules Untitled (C2OCl) X

Home Documents Downloads Music Pictures Videos

+ Other Locations

Name	Size	Modified
AcetylChloride.cml	794 bytes	12:20
admin_scripts		5 Aug 2019
AIPSDATA		24 Oct 2022
AIPSDATA_BAK		24 Oct 2022
alphafold		3 Aug
armpl		16 Jun 2022
assembly_index		29 Jul 2022
astrobiology		18 Feb
backups		11 Mar 2022
bad.local		18 Feb
bad_crypto		19 Jul 2022
batchspawner		9 Aug 2019
bin		9 Sep 2021
bin_bak		16 Oct 2019
carpentry		9 Mar
casa_env		18 Feb
casa_example		19 Feb
CATSettings		9 Jun 2022
certs		28 Aug 2019
clblast		16 Jun 2022
climate_change_jupyterhub		15 Sep 2022
climate_change_jupyterhub_old		13 Sep 2022
Complimmunology		27 Apr
conf.d		12 Oct 2022
configs		9 Jun 2022
Coursera		31 Aug 2019
crest		16 Jun 2022
CS499_HPC		Sat
cudnn-training		5 Nov 2020
Desktop		9 Mar

Chemical Markup Language

**File Edit View Crystal Extensions Quantum Help**

- New** Ctrl+N
- Open...** Ctrl+O
- Open Recent
- Save** Ctrl+S
- Save As...** Ctrl+Shift+S
- Export** ►
- Import** ►
- Quit** Ctrl+Q

**Save** Save As
Molecule































































































































































































































































Save as AcetylChloride.com

Compare molecules using InChI  
Copy raw text  
Crystallographic Information File  
DL-POLY CONFIG  
DMol3 coordinates format  
Dock 3.5 Box format  
FASTA format  
FHaims XYZ format  
FPS text fingerprint format  
Fastsearch format  
Feature format  
Fenske-Hall Z-Matrix format  
Fingerprint format  
Free Form Fractional format  
GAMESS Input  
GAMESS-UK Input  
GAMESS-UK Output  
GRO format  
GROMOS96 format  
Gaussian 98/03 Input  
Gaussian Z-Matrix Input  
Gaussian cube format  
Ghemical format  
HyperChem HIN format  
InChI format  
InChIKey  
Jaguar input format  
M.F. Sanner's MSMS input format  
MCDL format  
MDL

scp AcetylChloride.com vanilla@hopper.alliance.unm.edu:~  
AcetylChloride.com 100% 269 114.7KB/s 00:00

ssh vanilla@hopper.alliance.unm.edu:~

---

Welcome to Hopper

Be sure to review the "Acceptable Use" guidelines posted on the CARC website.

For assistance using this system email [help@carc.unm.edu](mailto:help@carc.unm.edu).

Tutorial videos can be accessed through the CARC website: Go to  
<http://carc.unm.edu>, select the "New Users" menu and then click  
"Introduction to Computing at CARC".

vanilla@hopper:~ \$

```
cd workshops/gaussian/
```

```
|ls -l
```

```
total 1192
-rw-r--r-- 1 mfricke users      500 Sep 12 13:22 AcetylChloride.com
-rw-r-xr-- 1 mfricke users     795 Sep 12 14:15 gaussian16_linda_acetylchloride.sh
```

```
cd workshops/gaussian/
```

```
ls -l
```

```
total 1192
-rw-r--r-- 1 mfricke users      500 Sep 12 13:22 AcetylChloride.com
-rw-r-xr-- 1 mfricke users     795 Sep 12 14:15 gaussian16_linda_acetylchloride.sh
```

```
sbatch gaussian16_linda_acetylchloride.sh
```

```
sbatch: Using account 2016199 from ~/.default_slurm_account
Submitted batch job 2047647
```

```
cd workshops/gaussian/
```

```
ls -l
```

```
total 1192
-rw-r--r-- 1 mfricke users      500 Sep 12 13:22 AcetylChloride.com
-rw-r-xr-- 1 mfricke users     795 Sep 12 14:15 gaussian16_linda_acetylchloride.sh
```

```
cat gaussian16_linda_acetylchloride.sh
```

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=4
#SBATCH --time=00:10:00
#SBATCH --job-name=g16_acetylchloride
#SBATCH --mail-user yourusername@unm.edu

module load gaussian/g16

# To make linda print verbose messages
export GAUSS_LFLAGS="-v"

INPUT_MOLECULE=$SLURM_SUBMIT_DIR/AcetylChloride.com
OUTPUT_FILE=$SLURM_SUBMIT_DIR/AcetylChloride.log

# Reformat the SLURM provided list of compute nodes to a format Gaussian can understand
# i.e. remove duplicates and replace newlines with commas
export GAUSS_WDEF=$(cat $CARC_NODEFILE | uniq | sed -z 's/\n/,/g;s/,$/\n/' )

# Tell Linda to use as many nodes as requested by the user
export GAUSS_PDEF=$SLURM_CPUS_PER_TASK
echo "Parallelizing $GAUSS_PDEF processes across $GAUSS_WDEF nodes."

# Set the memory Gaussian variable
export GAUSS_MDEF=4GB

# Run Gaussian
g16 $INPUT_MOLECULE $OUTPUT_FILE
```

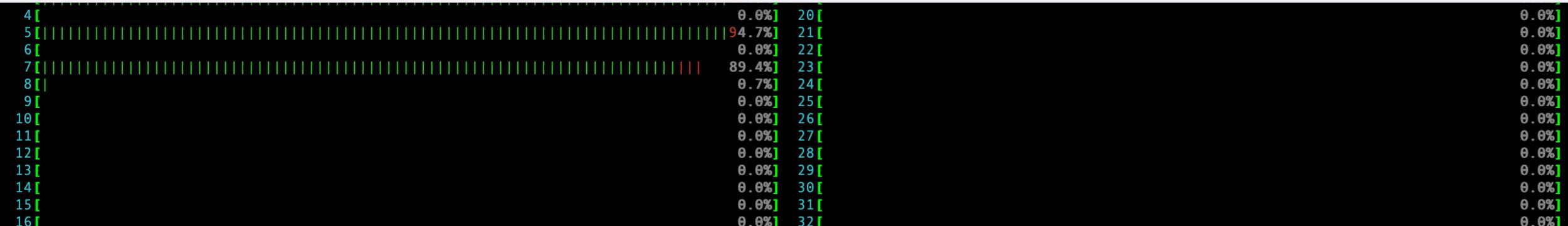
Gaussian uses a helper program called Linda to run multiple copies on multiple nodes. Linda does not understand SLURM so we have to spend some time converting SLURM information into a format Linda understand. That's most of this script.

watch squeue --me

Every 2.0s: squeue --me hopper: Tue  
Sep 12 14:37:13 2023

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
2047652	general	g16_acet	mfricke	R	1:05	2	hopper[003-004]

Output will be written to AcetylChloride.log and slurm-<jobid>.out



PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
240218	mfricke	20	0	6020M	265M	23148	R	371.0	0.3	1:22.65	/opt/local/gaussian/g16/C.01/avx2/g16/linda-exe/l508.exe 536870912 AcetylChloride.chk 1 /canc/scratch/users/mfricke/G
240306	mfricke	20	0	6020M	265M	23148	R	94.9	0.3	0:21.05	/opt/local/gaussian/g16/C.01/avx2/g16/linda-exe/l508.exe 536870912 AcetylChloride.chk 1 /canc/scratch/users/mfricke/G
240305	mfricke	20	0	6020M	265M	23148	R	94.2	0.3	0:20.96	/opt/local/gaussian/g16/C.01/avx2/g16/linda-exe/l508.exe 536870912 AcetylChloride.chk 1 /canc/scratch/users/mfricke/G
240304	mfricke	20	0	6020M	265M	23148	R	93.6	0.3	0:20.86	/opt/local/gaussian/g16/C.01/avx2/g16/linda-exe/l508.exe 536870912 AcetylChloride.chk 1 /canc/scratch/users/mfricke/G
240353	mfricke	20	0	36220	5036	3244	R	1.3	0.0	0:00.10	htop
1	root	20	0	232M	11176	8608	S	0.0	0.0	0:16.95	/sbin/init
506	root	20	0	695M	6364	4740	S	0.0	0.0	1:17.53	/warewulf/bin/wwclient
509	root	20	0	695M	6364	4740	S	0.0	0.0	0:01.80	/warewulf/bin/wwclient
511	root	20	0	695M	6364	4740	S	0.0	0.0	0:03.08	/warewulf/bin/wwclient
512	root	20	0	695M	6364	4740	S	0.0	0.0	0:03.20	/warewulf/bin/wwclient
513	root	20	0	695M	6364	4740	S	0.0	0.0	0:03.33	/warewulf/bin/wwclient
547	root	20	0	103M	23496	23044	S	0.0	0.0	0:02.61	/usr/lib/systemd/systemd-journald
600	root	20	0	97888	7384	6992	S	0.0	0.0	0:01.34	/usr/lib/systemd/systemd-udevd
792	rpc	20	0	67252	5092	4940	S	0.0	0.0	0:03.51	/usr/bin/rpcbind -w -f
805	root	20	0	235M	11356	10844	S	0.0	0.0	0:19.76	/usr/sbin/rsyslogd -n
806	dbus	20	0	76736	5212	4692	S	0.0	0.0	0:01.12	/usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
809	root	20	0	235M	11356	10844	S	0.0	0.0	0:19.39	/usr/sbin/rsyslogd -n
810	root	20	0	235M	11356	10844	S	0.0	0.0	0:00.32	/usr/sbin/rsyslogd -n
822	root	20	0	499M	33752	30424	S	0.0	0.0	23:02.61	/usr/libexec/platform-python -Es /usr/sbin/tuned -l -P
827	root	20	0	109M	3264	3228	S	0.0	0.0	0:00.00	/usr/sbin/gssproxy -D
828	root	20	0	109M	3264	3228	S	0.0	0.0	0:00.00	/usr/sbin/gssproxy -D
829	root	20	0	109M	3264	3228	S	0.0	0.0	0:00.00	/usr/sbin/gssproxy -D
830	root	20	0	109M	3264	3228	S	0.0	0.0	0:00.00	/usr/sbin/gssproxy -D
831	root	20	0	109M	3264	3228	S	0.0	0.0	0:00.00	/usr/sbin/gssproxy -D
832	root	20	0	109M	3264	3228	S	0.0	0.0	0:00.00	/usr/sbin/gssproxy -D
835	munge	20	0	253M	6228	4784	S	0.0	0.0	1:44.83	/usr/sbin/munged
837	munge	20	0	253M	6228	4784	S	0.0	0.0	1:42.73	/usr/sbin/munged
838	munge	20	0	253M	6228	4784	S	0.0	0.0	0:00.78	/usr/sbin/munged
839	munge	20	0	253M	6228	4784	S	0.0	0.0	0:00.77	/usr/sbin/munged
889	root	20	0	499M	33752	30424	S	0.0	0.0	10:33.69	/usr/libexec/platform-python -Es /usr/sbin/tuned -l -P
891	polkitd	20	0	1981M	18832	18284	S	0.0	0.0	0:00.42	/usr/lib/polkit-1/polkitd --no-debug
897	polkitd	20	0	1981M	18832	18284	S	0.0	0.0	0:00.00	/usr/lib/polkit-1/polkitd --no-debug
898	polkitd	20	0	1981M	18832	18284	S	0.0	0.0	0:00.29	/usr/lib/polkit-1/polkitd --no-debug

ssh hopper<number>  
 htop

# You have learned

- how to run programs using the SLURM scheduler
- the difference between interactive and batch jobs
- how to check the status of your jobs
- how to select debug vs general SLURM partitions
- how to ask for the hardware resources you need
- you ran the Gaussian code using SLURM

