azure-cmaq

Liz Adams

Feb 22, 2024

CONTENTS:

Warning: This documentation is under continuous development. The most recent version is available here: CMAQv5.4 on Azure

This document provides tutorials and information on using Microsoft Azure Online Portal to create either a single Virtual Machine or a Cycle Cloud Cluster to run CMAQ. The tutorials are aimed at users with cloud computing experience that are already familiar with Azure. For those with no cloud computing experience we recommend reviewing the Additional Resources listed in *chapter 13* of this document.

CHAPTER

ONE

FORMAT OF THIS DOCUMENTATION

This document provides three hands-on tutorials that are designed to be read in order. The Introductory Tutorial will walk you through setting up an Azure Account and logging into the Azure Portal Website. You will learn how to set up your Azure Resource ID, configure and create a demo virtual machine, and exit and delete the virtual machine and all of the resources associated with it by deleting resource group. The Intermediate Tutorial steps you through running a CMAQ test case on a single Virtual Machine with instructions to install CMAQ, libraries, and input data. The Advanced Tutorial explains how to create a CycleCloud (High Performance Cluster) for larger compute jobs and install CMAQ, requried libraries and input data. The remaining sections provide instructions on post-processing CMAQ output, comparing output and runtimes from multiple simulations, and copying output from CycleCloud to an Amazon Web Services (AWS) Simple Storage Service (S3) bucket.

CHAPTER

TWO

AZURE SUBSCRIPTIONS

The ability to use resources available in the Microsoft Azure Cloud is limited by quotas that are set at the subscription level. This tutorial was developed using UNC Chapel Hill's Enterprise account. Additional effort is being made to identify how to use a pay-as-you-go account, but these instructions have not been finalized. There may also be differences in how managed identies and user level permissions are set by the administrator of your enterprise level account that are not covered in this tutorial.

CHAPTER

THREE

WHY MIGHT I NEED TO USE AZURE VIRTUAL MACHINE OR CYCLECLOUD?

An Azure Virtual Machine may be configured to run code compiled with Message Passing Interface (MPI) on a single high performance compute node. The intermediate tutorial demonstrates how to run CMAQ interactively on a single virtual machine running CMAQ with OpenMPI on multiple cpus.

The Azure CycleCloud may be configured to be the equivalent of a High Performance Computing (HPC) environment, including using job schedulers such as Slurm, running on multiple nodes/virtual machines using code compiled with Message Passing Interface (MPI), and reading and writing output to a high performance, low latency shared disk. The advantage of using the slurm scheduler is that the number of compute nodes that will be provisioned can be adjusted to meet requirements of a given simulation. In addition, the user can reduce costs by using Spot instances rather than On-Demand for the compute nodes. CycleCloud also supports submitting multiple jobs to the job submission queue.

Our goal is make this user guide to running CMAQ on either a single Virtual Machine or the CycleCloud Cluster as helpful and user-friendly as possible. Any feedback is both welcome and appreciated.

Additional information on Azure CycleCloud:

CycleCloud HPC Scalabilty

Azure CycleCloud

3.1 Introductory Tutorial

Introductory Tutorial

3.1.1 Create an Azure Account

Create an account and configure your azure cyclecloud credentials. Create Free Azure Account

3.1.2 Sign up for a Developer Azure Support Plan

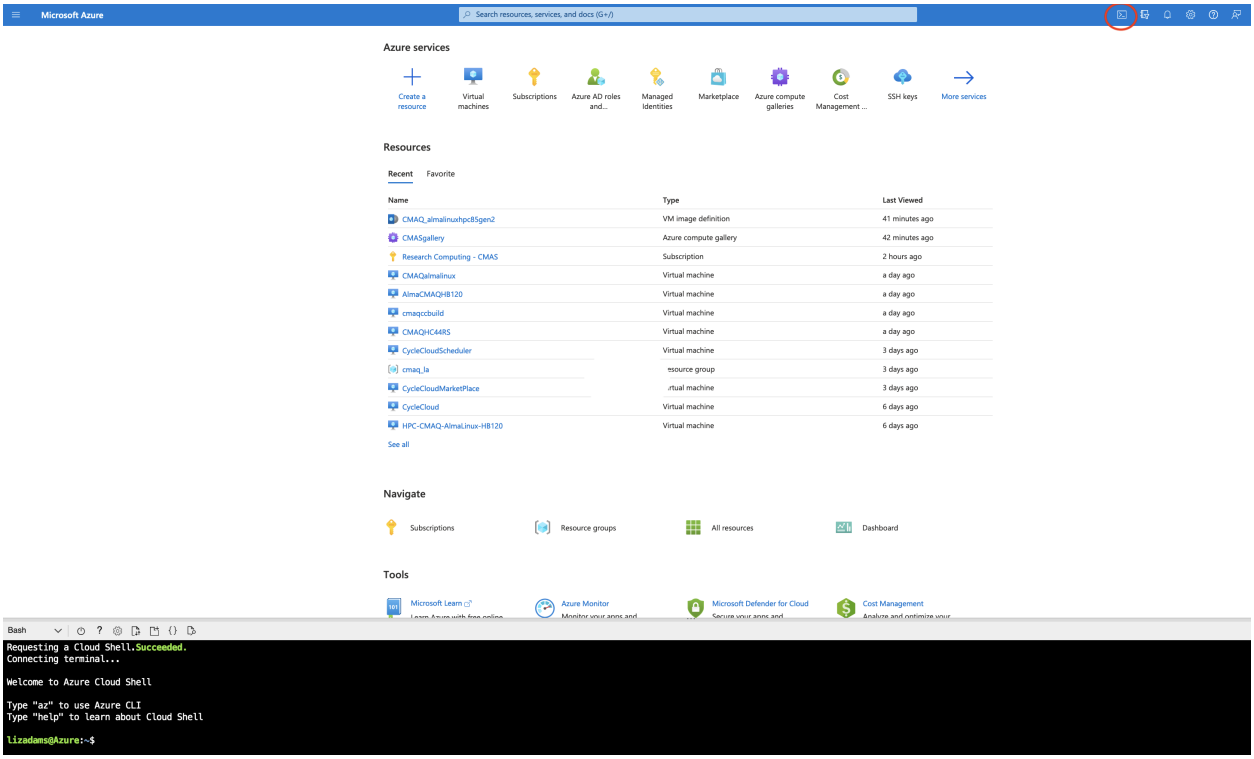
New accounts may be restricted to what virtual machines can be created by quota. With a pay-as-you go account or a free account, you need to sign up for the $29.99 per month support account in order to create a support request to increase the quota limit for the HC44rs or the HBv120 machines that are used in this tutorial. With an enterprise account, the support plan is included.

3.1.3 Use Azure CLI to examine your quota

Login to the Azure Portal

In the upper right corner, click on the icon for “Cloud Shell”

A new shell will be created at the bottom of your portal.



Enter the following at the command prompt to check your quota for the East US Region:

```
az vm list-usage --location "East US" -o table
```

Output:

```
Name                                         CurrentValue    Limit
-------------------------------------------  --------------  -------
Availability Sets                            0               2500
Total Regional vCPUs                         0               10
Virtual Machines                             0               25000
Virtual Machine Scale Sets                   0               2500
Dedicated vCPUs                              0               3000
Cloud Services                               0               2500
Total Regional Low-priority vCPUs            0               3
Standard LSv3 Family vCPUs                   0               0
Standard LASv3 Family vCPUs                  0               0
Standard DPLDSv5 Family vCPUs                0               0
Standard DPLSv5 Family vCPUs                 0               0
Standard DPDSv5 Family vCPUs                 0               0
Standard DPSv5 Family vCPUs                  0               0
Standard EPDSv5 Family vCPUs                 0               0
Standard EPSv5 Family vCPUs                  0               0
```

(continues on next page)

(continued from previous page)

```
Standard NCADS_A100_v4 Family vCPUs       0                 0
Basic A Family vCPUs                      0                 10
Standard A0-A7 Family vCPUs               0                 10
Standard A8-A11 Family vCPUs              0                 10
Standard D Family vCPUs                   0                 10
Standard Dv2 Family vCPUs                 0                 10
Standard DS Family vCPUs                  0                 10
Standard DSv2 Family vCPUs                0                 10
Standard G Family vCPUs                   0                 10
Standard GS Family vCPUs                  0                 10
Standard F Family vCPUs                   0                 10
Standard FS Family vCPUs                  0                 10
Standard NV Family vCPUs                  0                 12
Standard NC Family vCPUs                  0                 12
Standard H Family vCPUs                   0                 8
Standard Av2 Family vCPUs                 0                 10
Standard LS Family vCPUs                  0                 10
Standard Dv2 Promo Family vCPUs           0                 10
Standard DSv2 Promo Family vCPUs          0                 10
Standard MS Family vCPUs                  0                 0
Standard Dv3 Family vCPUs                 0                 10
Standard DSv3 Family vCPUs                0                 10
Standard Ev3 Family vCPUs                 0                 10
Standard ESv3 Family vCPUs                0                 10
Standard Dv4 Family vCPUs                 0                 10
Standard DDv4 Family vCPUs                0                 10
Standard DSv4 Family vCPUs                0                 10
Standard DDSv4 Family vCPUs               0                 10
Standard Ev4 Family vCPUs                 0                 10
Standard EDv4 Family vCPUs                0                 0
Standard ESv4 Family vCPUs                0                 0
Standard EDSv4 Family vCPUs               0                 10
Standard BS Family vCPUs                  0                 10
Standard FSv2 Family vCPUs                0                 10
Standard NDS Family vCPUs                 0                 0
Standard NCSv2 Family vCPUs               0                 0
Standard NCSv3 Family vCPUs               0                 0
Standard LSv2 Family vCPUs                0                 10
Standard PBS Family vCPUs                 0                 6
Standard EIv3 Family vCPUs                0                 10
Standard EISv3 Family vCPUs               0                 10
Standard DCS Family vCPUs                 0                 8
Standard NVSv2 Family vCPUs               0                 0
Standard MSv2 Family vCPUs                0                 0
Standard HBS Family vCPUs                 0                 0
Standard HCS Family vCPUs                 0                 0
Standard NVSv3 Family vCPUs               0                 0
Standard NV Promo Family vCPUs            0                 12
Standard NC Promo Family vCPUs            0                 12
Standard H Promo Family vCPUs             0                 8
Standard DAv4 Family vCPUs                0                 0
Standard DASv4 Family vCPUs               0                 10
```

(continues on next page)

(continued from previous page)

```
Standard EAv4 Family vCPUs                  0               0
Standard EASv4 Family vCPUs                 0               10
Standard NDSv3 Family vCPUs                 0               0
Standard DCSv2 Family vCPUs                 0               8
Standard NVSv4 Family vCPUs                 0               8
Standard NDSv2 Family vCPUs                 0               0
Standard NPS Family vCPUs                   0               0
Standard HBrsv2 Family vCPUs                0               0
Standard NCASv3_T4 Family vCPUs             0               0
Standard NDASv4_A100 Family vCPUs           0               0
Standard EIDSv4 Family vCPUs                0               0
Standard XEISv4 Family vCPUs                0               0
Standard EIASv4 Family vCPUs                0               0
Standard HBv3 Family vCPUs                  0               0
Standard MDSMediumMemoryv2 Family vCPUs     0               0
Standard MIDSMediumMemoryv2 Family vCPUs    0               0
Standard MSMediumMemoryv2 Family vCPUs      0               0
Standard MISMediumMemoryv2 Family vCPUs     0               0
Standard DASv5 Family vCPUs                 0               0
Standard EASv5 Family vCPUs                 0               0
Standard Ev5 Family vCPUs                   0               0
Standard EIv5 Family vCPUs                  0               0
Standard EDv5 Family vCPUs                  0               0
Standard EIDv5 Family vCPUs                 0               0
Standard ESv5 Family vCPUs                  0               0
Standard EISv5 Family vCPUs                 0               0
Standard EDSv5 Family vCPUs                 0               0
Standard EIDSv5 Family vCPUs                0               0
Standard Dv5 Family vCPUs                   0               0
Standard DDv5 Family vCPUs                  0               0
Standard DSv5 Family vCPUs                  0               0
Standard DDSv5 Family vCPUs                 0               0
Standard DCSv3 Family vCPUs                 0               0
Standard DDCSv3 Family vCPUs                0               0
Standard DADSv5 Family vCPUs                0               0
Standard EADSv5 Family vCPUs                0               0
Standard FXMDVS Family vCPUs                0               0
Standard NDAMSv4_A100Family vCPUs           0               0
Standard DCASv5 Family vCPUs                0               0
Standard ECASv5 Family vCPUs                0               0
Standard ECIASv5 Family vCPUs               0               0
Standard DCADSv5 Family vCPUs               0               0
Standard ECADSv5 Family vCPUs               0               0
Standard ECIADSv5 Family vCPUs              0               0
Standard NVADSA10v5 Family vCPUs            0               0
Standard EBDSv5 Family vCPUs                0               10
Standard EBSv5 Family vCPUs                 0               10
Standard EIASv5 Family vCPUs                0               0
Standard EIADSv5 Family vCPUs               0               0
Standard NCADSA10v4 Family vCPUs            0               0
Standard Storage Managed Disks              0               50000
Premium Storage Managed Disks               0               50000
```

(continues on next page)

(continued from previous page)

```
StandardSSDStorageDisks                 0                50000
StandardSSDZRSStorageDisks              0                50000
PremiumZRSStorageDisks                  0                50000
UltraSSDStorageDisks                    0                1000
PremiumV2StorageDisks                   0                1000
StandardStorageSnapshots                0                75000
StandardSSDStorageSnapshots             0                75000
PremiumStorageSnapshots                 0                75000
ZrsStorageSnapshots                     0                75000
UltraSSDTotalSizeInGB                   0                32768
PremiumV2TotalDiskSizeInGB              0                65536
DiskEncryptionSets                      0                1000
DiskAccesses                            0                1000
Gallery                                 0                100
Gallery Image                           0                1000
Gallery Image Version                   0                10000
```

Review list of regions and virtual machines available in each region.

Azure Regions

Follow the instructions in the link below to increase your vCPU quota to allow you to create a virtual machine and run CMAQ.

With a pay-as-you-go account this request to increase a quota for virtual machines may take 3-5 business days.

Azure Regions

Review the virtual machines available from each region

Request a quota increase for the HTC Queue - HC Family of vCPUs for a region where they are available.

From the Azure Portal, search for quotas, select the Quotas Service. Click on Compute.

Use the Search Box to search for HC. Select the HC Standard Family vCPUs in the region nearest to your location. (For North Carolina select - East US) by clicking on the check box to the left of that selection.

Click on Request quota increase > Select Enter a new limit. In the sidebar menu on the righ hand side, enter 44 in the text box under new limit.

Request a quota increase for the HPC Queue - HBv3 Family of vCPUs

From the Azure Portal, search for quotas, select the Quotas Service. Click on Compute.

Use the Search Box to search for HB Select the HBv3 Family vCPUs in the region nearest to your location. (For North Carolina select - East US) by clicking on the check box to the left of that selection.

Click on Request quota increase > Select Enter a new limit. In the sidebar menu on the righ hand side, enter 120 in the text box under new limit.

Request a quota increase for the scheduler node D4s_v3

From the Azure Portal, search for quotas, select the Quotas Service. Click on Compute.

Use the Search Box to search for Dv3 Select the Standard Dv3 Family vCPUs in the region nearest to your location. (For North Carolina select - East US) by clicking on the check box to the left of that selection.

Click on Request quota increase > Select Enter a new limit. In the sidebar menu on the righ hand side, enter 4 in the text box under new limit to request an increase in the quota to 4 vcpu.

Request a quota increase for the F2sV2 HTC Compute Node (part of the Fsv2-series instances)

From the Azure Portal, search for quotas, select the Quotas Service. Click on Compute.

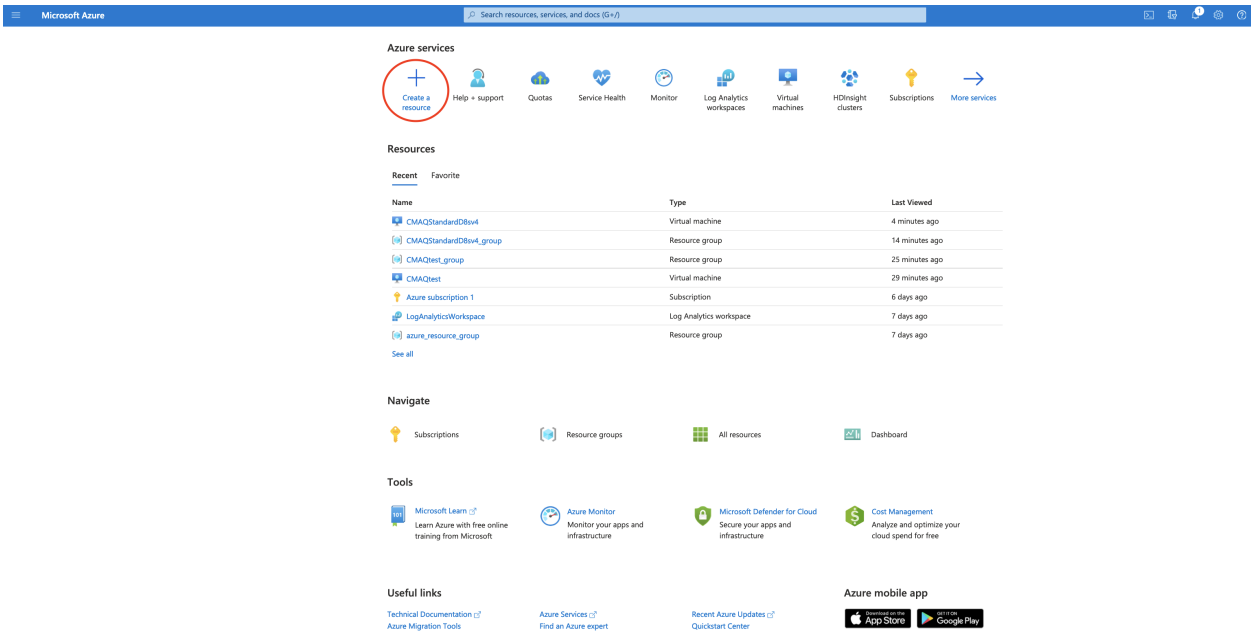
Use the Search Box to search for Select the HBv3 Family vCPUs in the region nearest to your location. (For North Carolina select - East US) by clicking on the check box to the left of that selection.

Click on Request quota increase > Select Enter a new limit. In the sidebar menu on the righ hand side, enter 44 in the text box under new limit.

3.1.4 Create a virtual machine

Once your quota limit has been approved, then you will be able to select a virtual machine

From the Azure Portal Click on Create a resource



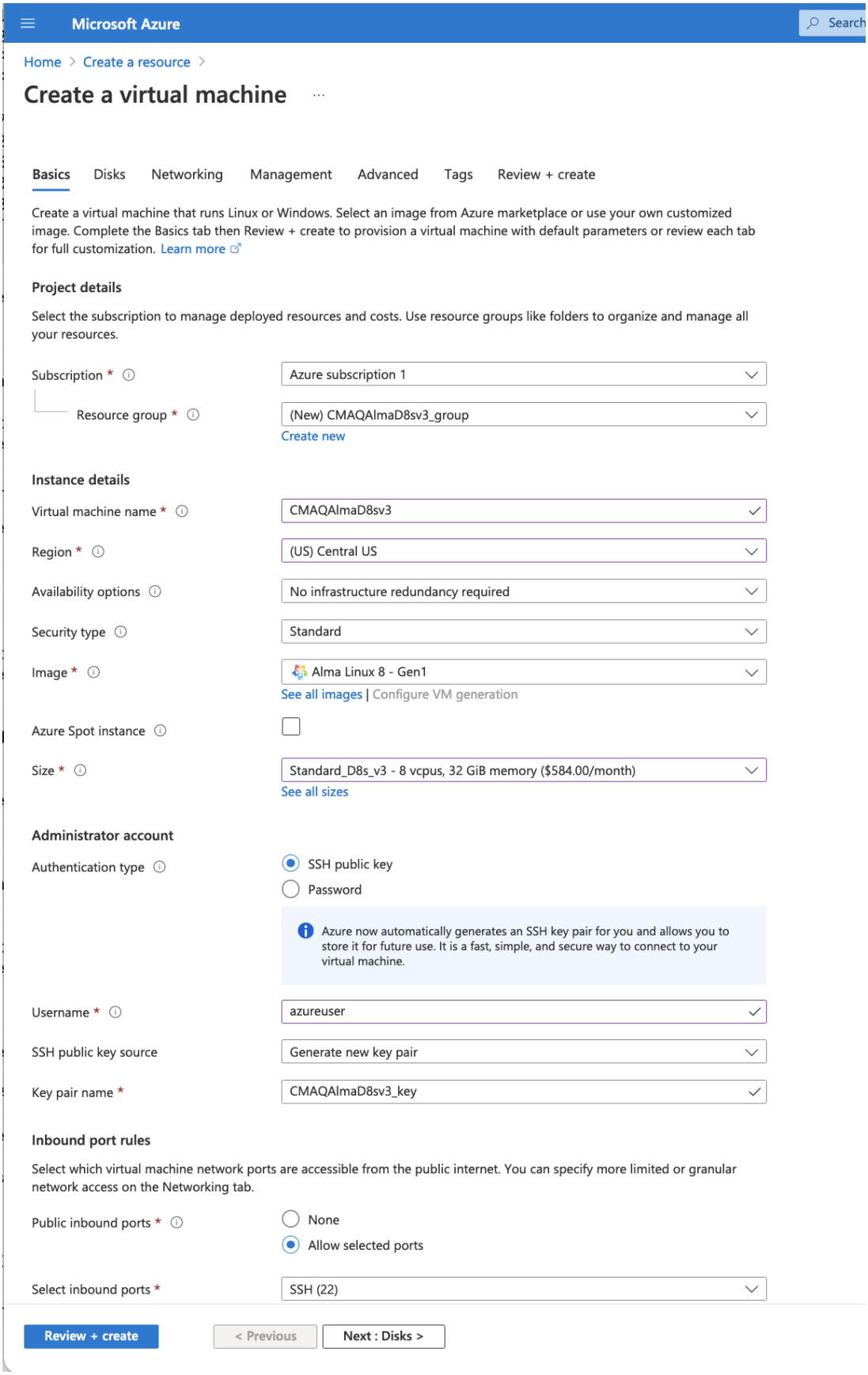
The availability of Images that can be selected depends on the region.

For high performance computing applications, the recommended operating system is Alma Linux 8 - Gen 2, but not all regions have Gen 2, so you may be limited to Gen 1.

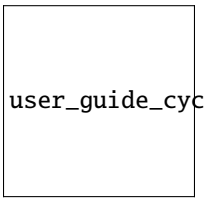
Enter Name, then Select Region, Select Image and Select Size

Recommend selecting a machine name that indicates what you will be using it for, the Operating System, and the Machine Size

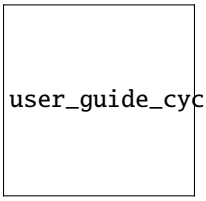
Virtual Machine Name: CMAQAlmaD8sv3 Region: Central US Image: Alma Linux Gen 1 - first select see all images, then search on HPC, then select Alma Linux. If Gen 2 is available, then select that, if not, select Gen 1. Size: Standard_D8_v3 - first select see all sizes, then select an image that is large enough to run CMAQ CONUS Domain


Microsoft Azure
Search
Home > Create a resource >
Create a virtual machine
Basics Disks Networking Management Advanced Tags Review + create
Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. Learn more
Project details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription * Azure subscription 1
Resource group * (New) CMAQAlmaD8sv3_group
Create new
Instance details
Virtual machine name * CMAQAlmaD8sv3
Region * (US) Central US
Availability options No infrastructure redundancy required
Security type Standard
Image * Alma Linux 8 - Gen1
See all images | Configure VM generation
Azure Spot instance
Size * Standard_D8s_v3 - 8 vcpus, 32 GiB memory ($584.00/month)
See all sizes
Administrator account
Authentication type SSH public key
Password
Azure now automatically generates an SSH key pair for you and allows you to store it for future use. It is a fast, simple, and secure way to connect to your virtual machine.
Username * azureuser
SSH public key source Generate new key pair
Key pair name * CMAQAlmaD8sv3_key
Inbound port rules
Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.
Public inbound ports * None
Allow selected ports
Select inbound ports * SSH (22)
Review + create
< Previous
Next : Disks >

Select Next Disks Click on Create and attach new disk of size 1TB Click on Delete Disk with VM


user_guide_cyclecloud/cyclecloud-cmaq/docs/Create_Virtual_Machine_Create_New_Disk.png

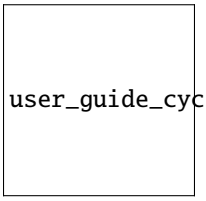
Click Next Networking - accept default settings Click Next Management - Select the System Managed Identity


user_guide_cyclecloud/cyclecloud-cmaq/docs/azure_web_interface_images/Create_virtual_machine_select_Sys

Click on Review and Create - Click on Create

A pop-up window titled “Generate a new key pair” will appear.

Click on Download private key and create resource.


user_guide_cyclecloud/cyclecloud-cmaq/docs/azure_web_interface_imagesVirtual_Machine_Connect_with_SSH.p

Once your resource is created in the upper right corner of the new screen, there will be a menu optione titled Connect.

Select Connect then SSH

The screen will then provide instructions for you to login to the newly created virtual machine.

3.1.5 Login to Virtual Machine

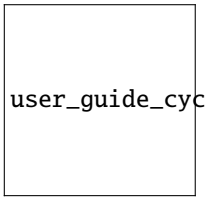
```
ssh -i ./CMAQStandardD8sv4_key.pem azureuser@13.89.128.245
```

Verify that the 1024 GiB size disk is not listed as being available

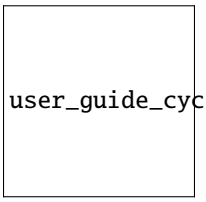
```
df -h
```

In the intermediate tutorial, instructions are provided to find the disk and mount it as a /shared volume to the virtual machine.

3.1.6 Delete the virtual machine and all of the associated resources by deleting the resource group.

Deleting the resource group will delete the virtual machine and will also delete the associated resources that were created for the virtual machine. 

A pop-up window will appear on the right side of the Azure Portal to confirm that you want to delete the resource group.



This tutorial was developed using UNC's enterprise account. It is unknown if Azure will grant access to these virtual machines on a credit card account.

3.2 System Requirements

Description of the compute node and head nodes used for the CycleCloud

3.2.1 System Requirements for a Single Virtual Machine or Cycle Cloud Cluster

Please set up a alarm on Azure

Set alarm to receive an email alert if you exceed $100 per month (or what ever monthly spending limit you need). It may be possible to set up daily or weekly spending alarms as well.

Azure Documentation on selecting the right VM for your workloads

Description of Azure Virtual Machines

For CMAQ, it is recommended that the user select a High Performance Compute Virtual Machine.

Virtual Machines in Azure

Your options on selecting the right VMs for your workloads

Type	Sizes	Description
General purpose	B, Dsv3, Dv3, Dasv4, Dav4, DSv2, Dv2, Av2, DC, DCv2, Dv4, Dsv4, Ddv4, Ddsv4, Dv5, Dsv5, Ddv5, Ddsv5, Dasv5, Dadsv5	Balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers.
Compute optimized	F, Fs, Fsv2, FX	High CPU-to-memory ratio. Good for medium traffic web servers, network appliances, batch processes, and application servers.
Memory optimized	Esv3, Ev3, Easv4, Eav4, Ebdsv5, Ebsv5, Ev4, Esv4, Edv4, Edsv4, Ev5, Esv5, Edv5, Edsv5, Easv5, Eadsv5, Mv2, M, DSv2, Dv2	High memory-to-CPU ratio. Great for relational database servers, medium to large caches, and in-memory analytics.
Storage optimized	Lsv2, Lsv3, Lasv3	High disk throughput and IO ideal for Big Data, SQL, NoSQL databases, data warehousing and large transactional databases.
GPU	NC, NCv2, NCv3, NCasT4_v3, ND, NDv2, NV, NVv3, NVv4, NDasrA100_v4, NDm_A100_v4	Specialized virtual machines targeted for heavy graphic rendering and video editing, as well as model training and inferencing (ND) with deep learning. Available with single or multiple GPUs.
High performance compute	HB, HBv2, HBv3, HC, H	Our fastest and most powerful CPU virtual machines with optional high-throughput network interfaces (RDMA).

3.2.2 Software Requirements for CMAQ on Single VM or CycleCloud Cluster

The software requirements to run CMAQ on Azure are split into three tiers. The first tier includes the software that is provided with the operating system, the second tier includes the libraries required by CMAQ, the third tier includes the CMAQ code and associated pre and post processors, and the third tier includes the R software and packages requried by the analysis scripts for verifying output or doing a quality assurance of CMAQ.

Tier 1: Native Operating System (OS) and associated system libraries, compilers for both Single VM or Cycle Cloud Cluster

- Tcsh shell
- Alma Linux Gen. 2
- Git
- Compilers (C, C++, and Fortran) - GNU compilers version gcc (GCC) 9.2.0 (need to use module load gcc-9.2.0)
- MPI (Message Passing Interface) - OpenMPI 4.1.0 (need to use module load mpi/openmpi-4.1.0)

Tier 1: For the Cycle Cloud Cluster

- Slurm Scheduler

Tier 2: additional libraries required for installing CMAQ

- NetCDF (with C, C++, and Fortran support)
- I/O API

Tier 3: Software distributed thru the CMAS Center

- CMAQv533
- CMAQv533 Post Processors

Tier 4: R packages and Scripts

- R QA Scripts

Hardware Requirements

Recommended Minimum Requirements

The size of hardware depends on the domain size and resolution for your CMAQ case, and how quickly your turn-around requirements are. Larger hardware and memory configurations are also required for instrumented versions of CMAQ incuding CMAQ-ISAM and CMAQ-DDM3D.

Azure Single Virtual Machine

Azure offers generalized, compute, and high performance machines of various sizes. The amount of memory and the number of cpus required to run CMAQ depends on the domain size and resolution of the case that is being run. For this tutorial that uses a two day run of the CONUS2 domain, a minimum size recommended is a HC44rs (44 cpus) or HBv120 (120 cpus) compute node, to allow CMAQ to be run on up to 44 or 120 cpus.

HC Series Virtual Machine Overview

Physically, an HC-series server is 2 * 24-core Intel Xeon Platinum 8168 CPUs for a total of 48 physical cores. Each CPU is a single pNUMA domain, and has unified access to six channels of DRAM. Intel Xeon Platinum CPUs feature a 4x larger L2 cache than in prior generations (256 KB/core -> 1 MB/core), while also reducing the L3 cache compared to prior Intel CPUs (2.5 MB/core -> 1.375 MB/core).

The above topology carries over to the HC-series hypervisor configuration as well. To provide room for the Azure hypervisor to operate without interfering with the VM, we reserve pCores 0-1 and 24-25 (that is, the first 2 pCores on each socket). We then assign pNUMA domains all remaining cores to the VM. Thus, the VM will see:

```
(2 vNUMA domains) * (22 cores/vNUMA) = 44 cores per VM
```

HBv3-series Software Specification

An HBv3-series server features 2 * 64-core EPYC 7V73X CPUs for a total of 128 physical “Zen3” cores with AMD 3D V-Cache. Simultaneous Multithreading (SMT) is disabled on HBv3. 448 GB of RAM, and no hyperthreading with 350 GB/sec of memory bandwidth, up to 32 MB of L3 cache per core, up to 7 GB/s of block device SSD performance, and clock frequencies up to 3.675 GHz.

Azure CycleCloud Cluster

Azure CycleCloud Provides the simplest way to manage HPC workloads using any scheduler (like Slurm, Grid Engine, HPC Pack, HTCondor, LSF, PBS Pro, or Symphony).

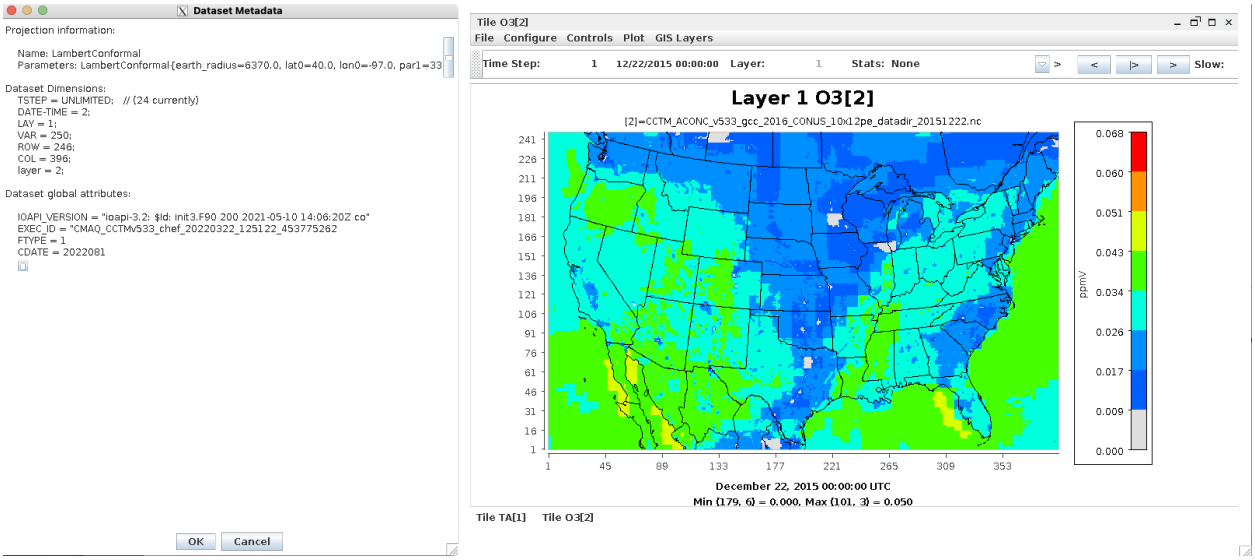
CycleCloud allows you to:

- Deploy full clusters and other resources, including scheduler, compute VMs, storage, networking, and cache
- Orchestrate job, data, and cloud workflows
- Give admins full control over which users can run jobs, as well as where and at what cost
- Customize and optimize clusters through advanced policy and governance features, including cost controls, Active Directory integration, monitoring, and reporting
- Use your current job scheduler and applications without modification
- Take advantage of built-in autoscaling and battle-tested reference architectures for a wide range of HPC workloads and industries

Azure CycleCloud

12US2 Benchmark Domain Description

```
GRIDDESC
'12US2'
'12CONUS'      -2412000.0 -1620000.0 12000.0 12000.0 396 246 1
```



3.2.3 Storage Options

CMAQ requires low-latency storage, especially if you are running CMAQ on a large domain and using more than 200 processors.

Azure File Storage account for premium file shares is required.

Quote from following link: "Provisioned file shares can be dynamically scaled up or down depending on your storage and IO performance characteristics. The provisioned size of the file share can be increased at any time but can be decreased only after 24 hours since the last increase. After waiting for 24 hours without a quota increase, you can decrease the share quota as many times as you like, until you increase it again. IOPS/throughput scale changes will be effective within a few minutes after the provisioned size change."

Azure Premium File Shares

3.2.4 Recommended Cycle Cloud Configuration for CONUS Domain 12US2

Note, first create a VM using the image: CycleCloud 8.2, and from that VM, the Cycle Cloud is built. VM:

*F4sV2 (4vcpus, 8 GiB memory) - VM image: CycleCloud 8.2

CycleCloud Configuration:

Scheduler node:

- D4s_v3

Compute Node for HTC Queue - used for Post-Processing (combine, etc):

- F2sV2 (part of the Fsv2-series instances)

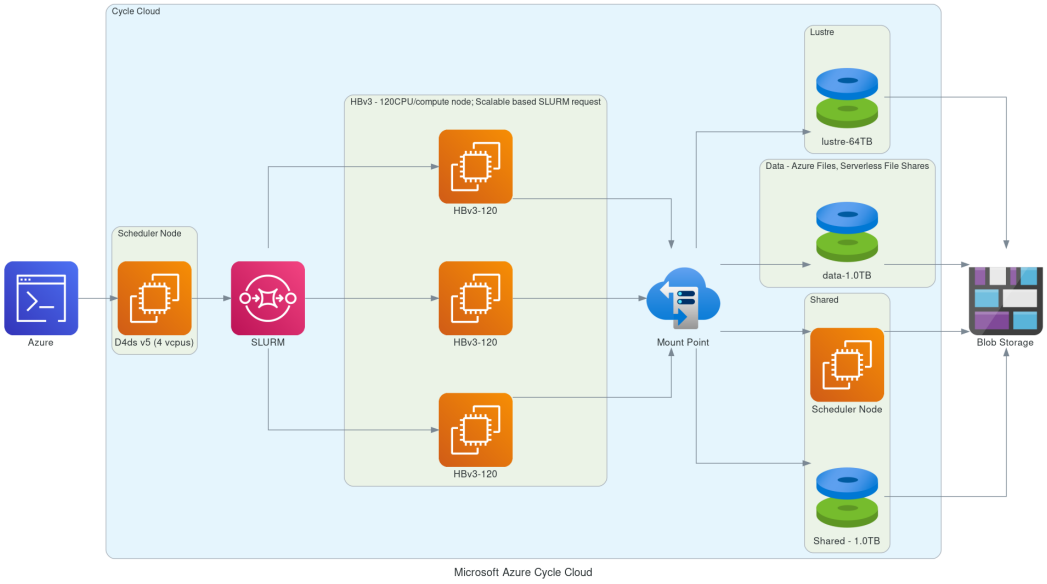
Compute Node for HPC Queue - used to run CMAQ:

- HBv3-120 instance running AlmaLinux

HBv3-series Software Specification

448 GB of RAM, and no hyperthreading with 350 GB/sec of memory bandwidth, up to 32 MB of L3 cache per core, up to 7 GB/s of block device SSD performance, and clock frequencies up to 3.675 GHz.

Figure 1. Cycle Cloud Recommended Cluster Configuration (Number of compute nodes depends on setting for NPCOLxNPROW and #SBATCH –nodes=XX #SBATCH –ntasks-per-node=YY)



Azure CycleCloud specifies what resource to use for disks, scheduler node, and compute nodes.

Cycle Cloud simply tries to schedule the job according to the slurm scheduler instructions. Slurm controls the launch, terminate, and maintain resources. If you try to allocate more nodes than are available in the Cycle Cloud Configuration, then you will need to edit the HPC config in the cyclecloud web interface to set the CPUs to 480 or more and then run the following on the scheduler node the changes should get picked up:

```
cd /opt/cycle/slurm
```

```
sudo ./cyclecloud_slurm.sh scale
```

Number of compute nodes dispatched by the slurm scheduler is specified in the run script using #SBATCH –nodes=XX #SBATCH –ntasks-per-node=YY where the maximum value of tasks per node or YY limited by many CPUs are on the compute node.

For HBv3-120, there are 120 CPUs, so maximum value of YY is 120 or –ntask-per-node=120.

If running a job with 180 processors, this would require the –nodes=XX or XX to be set to 2 compute nodes, as 90x2=180.

The setting for NPCOLxNPROW must also be a maximum of 180, ie. 18 x 10 or 10 x 18 to use all of the CPUs in the CycleCloud HPC Node.

HBv3-120 instance

Software:

- Alma Linux
- Spot or OnDemand Pricing
- /shared/build volume install software from git repo

- 1. TB Shared file system
- Slurm Placement Group enabled
- Elastic Fabric Adapter Enabled on HBv3-120

3.3 Intermediate Tutorial

Run CMAQ on a single Virtual Machine (VM) using HBv120 and AlmaLinux 8.5 HPC - Gen2.

Intermediate Tutorial: Run CMAQv533 from HBv120 Compute Node

Instructions are provided to build and install CMAQ on HBv120 compute node installed from HPC AlmaLinux 8.5 HPC-Gen2 Image that contains modules for git, openmpi and gcc. The compute node does not have a SLURM scheduler on it, so jobs are run interactively from the command line.

Instructions to install data and CMAQ libraries and model are provided along with sample run scripts to run CMAQ on 16, 36, 90, and 120 processors on a single HBv120 instance.

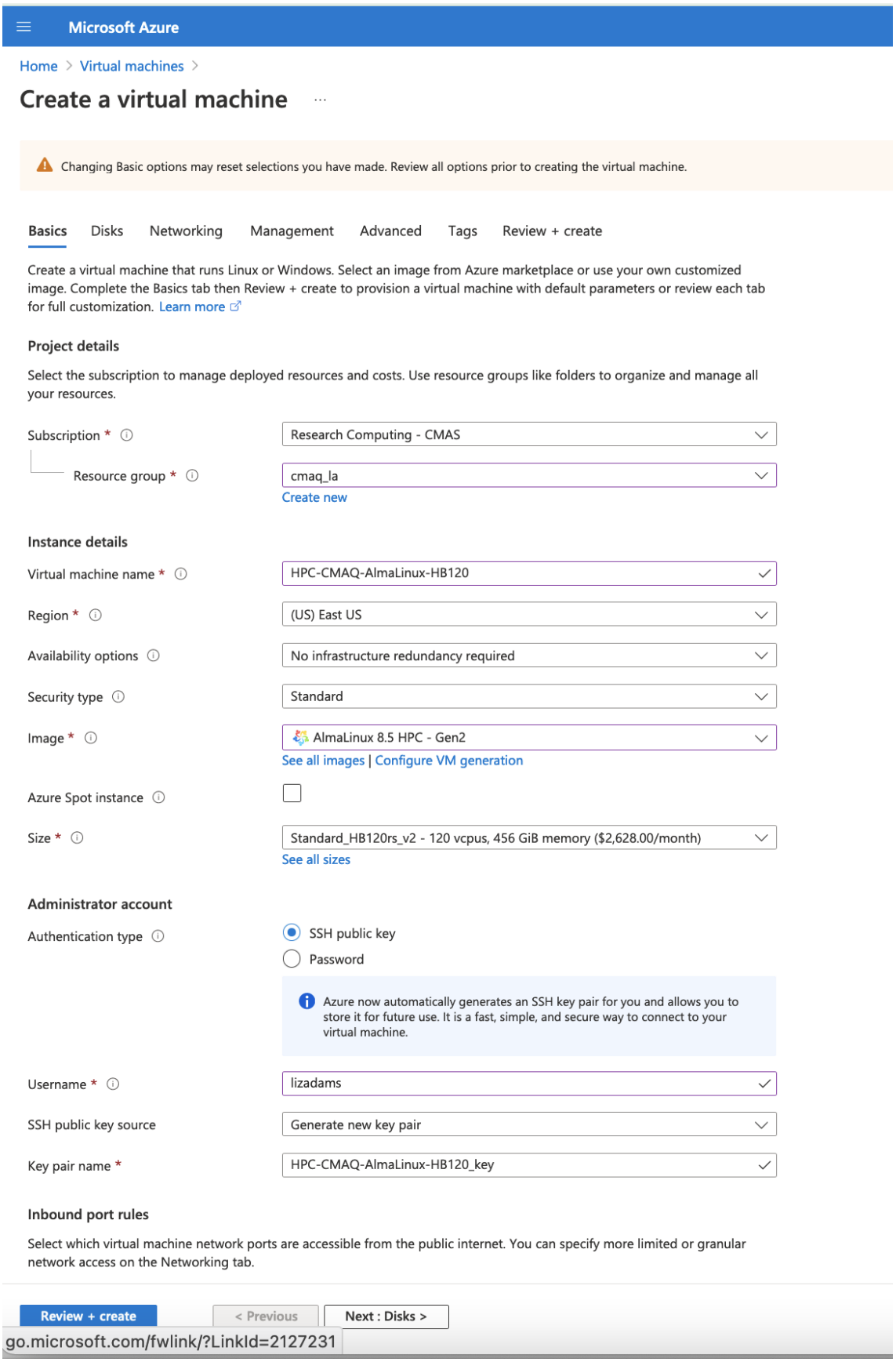
This will provide users with experience using the Azure Portal to create a Virtual Machine, select AlmaLinux 8.5 HPC - Gen2 as the image, select the size of the VM as HB120rs_v2 - 120 vcpus, 456 GiB memory, using an SSH private key to login and install and run CMAQ.

Using this method, the user needs to be careful to start and stop the Virtual Machine and only have it run while doing the intial installation, and while running CMAQ. The full HBv120 instance will incur charges as long as it is on, even if a job isn't running on it.

This is different than the Azure Cycle-Cloud, where if CMAQ is not running in the queue, then the HBv120 Compute nodes are down, and not incurring costs.

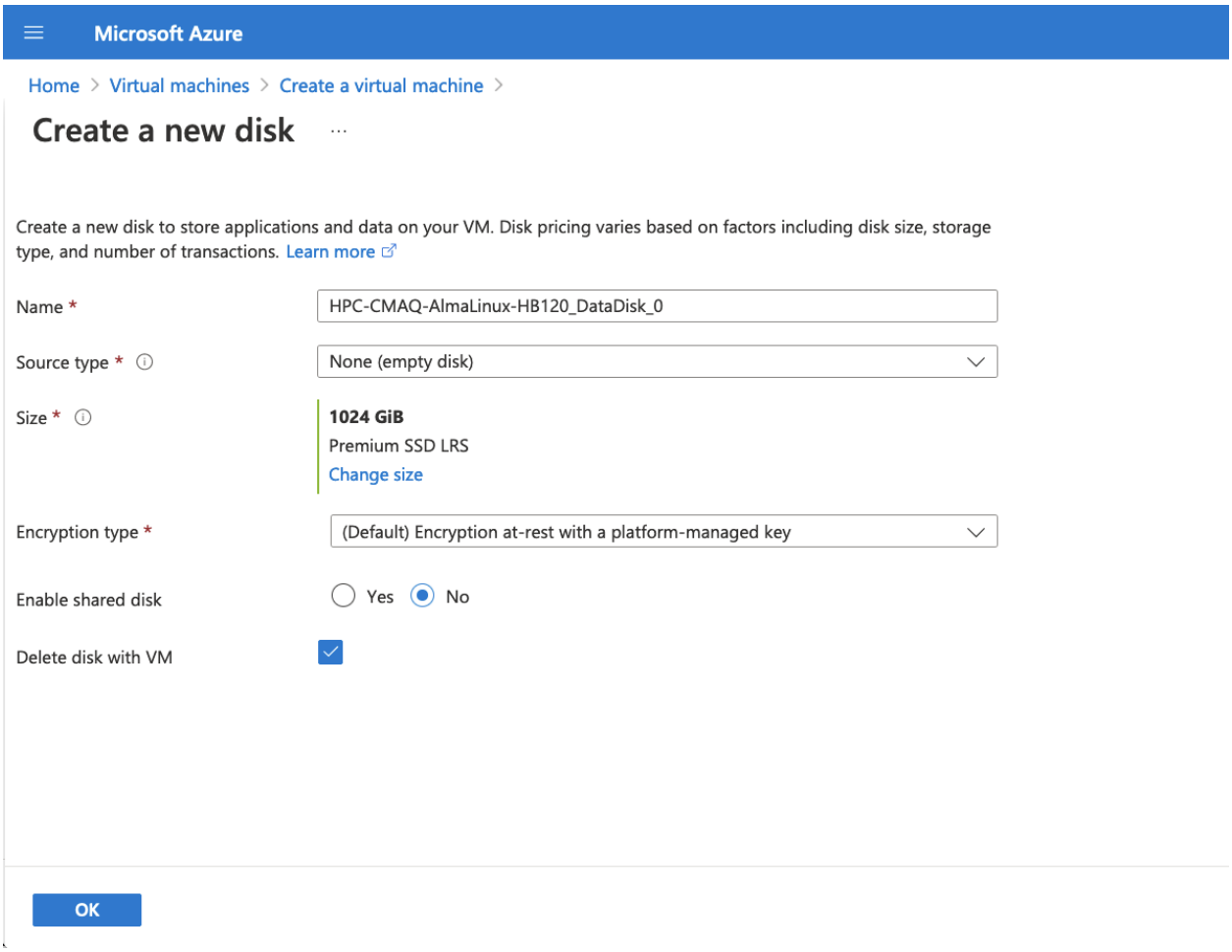
3.3.1 Create a HB120rs_v2 Virtual Machine

1. Login to Azure Portal
2. Select Create a Virtual Machine
3. Click on See all images next to Image and use the search bar to search for HPC. Look for the AlmaLinux 8.5 HPC. Select either the Gen 1 or Gen 2, and click. That option should now pre-populate the form.
4. Select Size - Standard_HB1120rs_v2 - 120 vcpus, 456 GiB memory ($2,628.0/monthly)
5. Enter a Virtual Machine Name in the text box
6. Use your username or azureuser
7. Select Authentication type - SSH public key
8. Select SSH public key source - Generate new key pair


Microsoft Azure
Home > Virtual machines >
Create a virtual machine
Changing Basic options may reset selections you have made. Review all options prior to creating the virtual machine.
Basics Disks Networking Management Advanced Tags Review + create
Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. Learn more
Project details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription * Research Computing - CMAS
Resource group * cmaq_la
Create new
Instance details
Virtual machine name * HPC-CMAQ-AlmaLinux-HB120
Region * (US) East US
Availability options No infrastructure redundancy required
Security type Standard
Image * AlmaLinux 8.5 HPC - Gen2
See all images | Configure VM generation
Azure Spot instance
Size * Standard_HB120rs_v2 - 120 vcpus, 456 GiB memory ($2,628.00/month)
See all sizes
Administrator account
Authentication type
SSH public key
Password
Azure now automatically generates an SSH key pair for you and allows you to store it for future use. It is a fast, simple, and secure way to connect to your virtual machine.
Username * lizadams
SSH public key source Generate new key pair
Key pair name * HPC-CMAQ-AlmaLinux-HB120_key
Inbound port rules
Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.
Review + create
< Previous
Next : Disks >
go.microsoft.com/fwlink/?LinkId=2127231

Click on Next > Disks

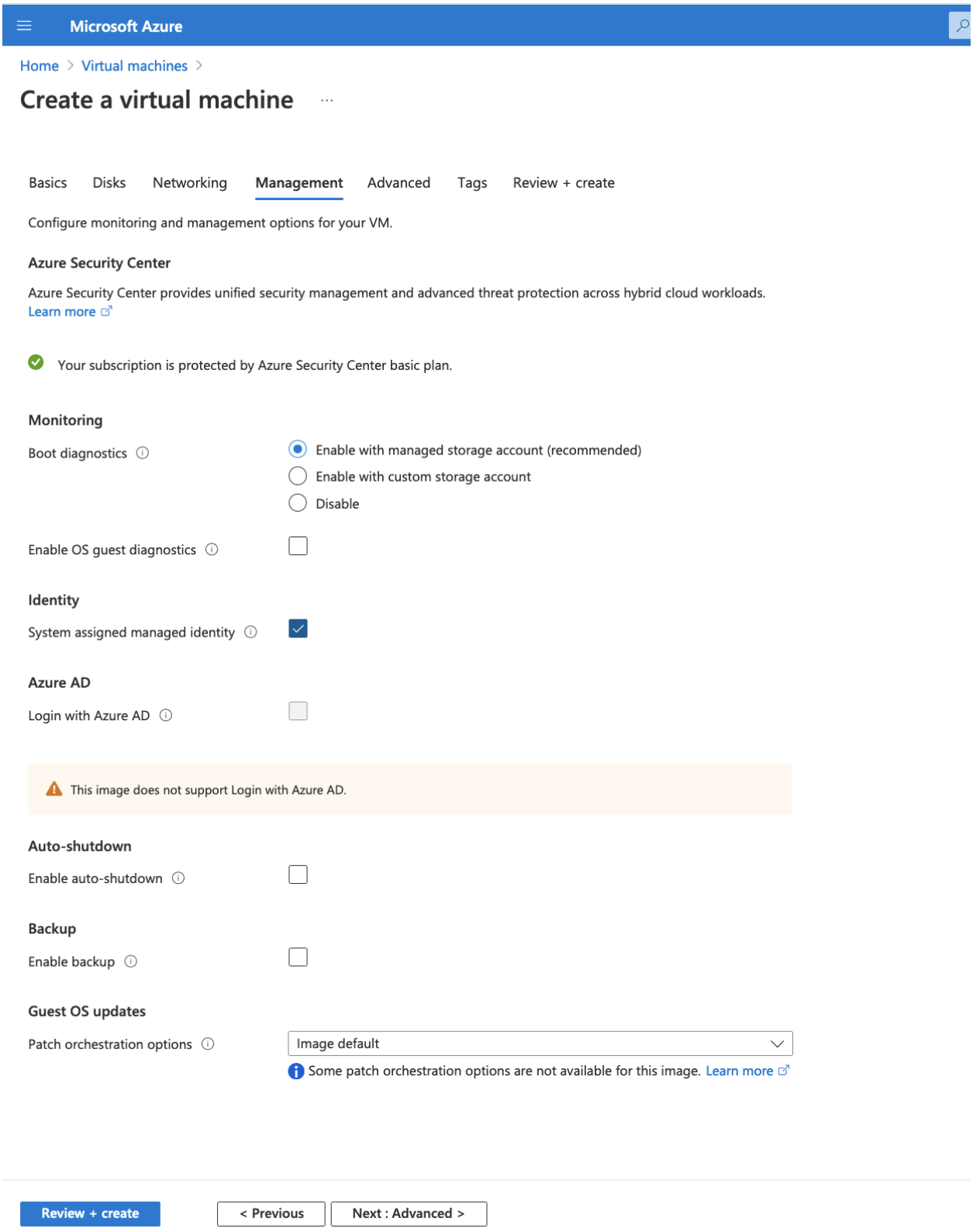
1. Click on Create and attach a new disk - select a 1TB disk
2. Select Checkbox to Delete disk with VM



(note, this will create the disk, but you will need to login and mount the disk as the shared volume following the instructions below.)

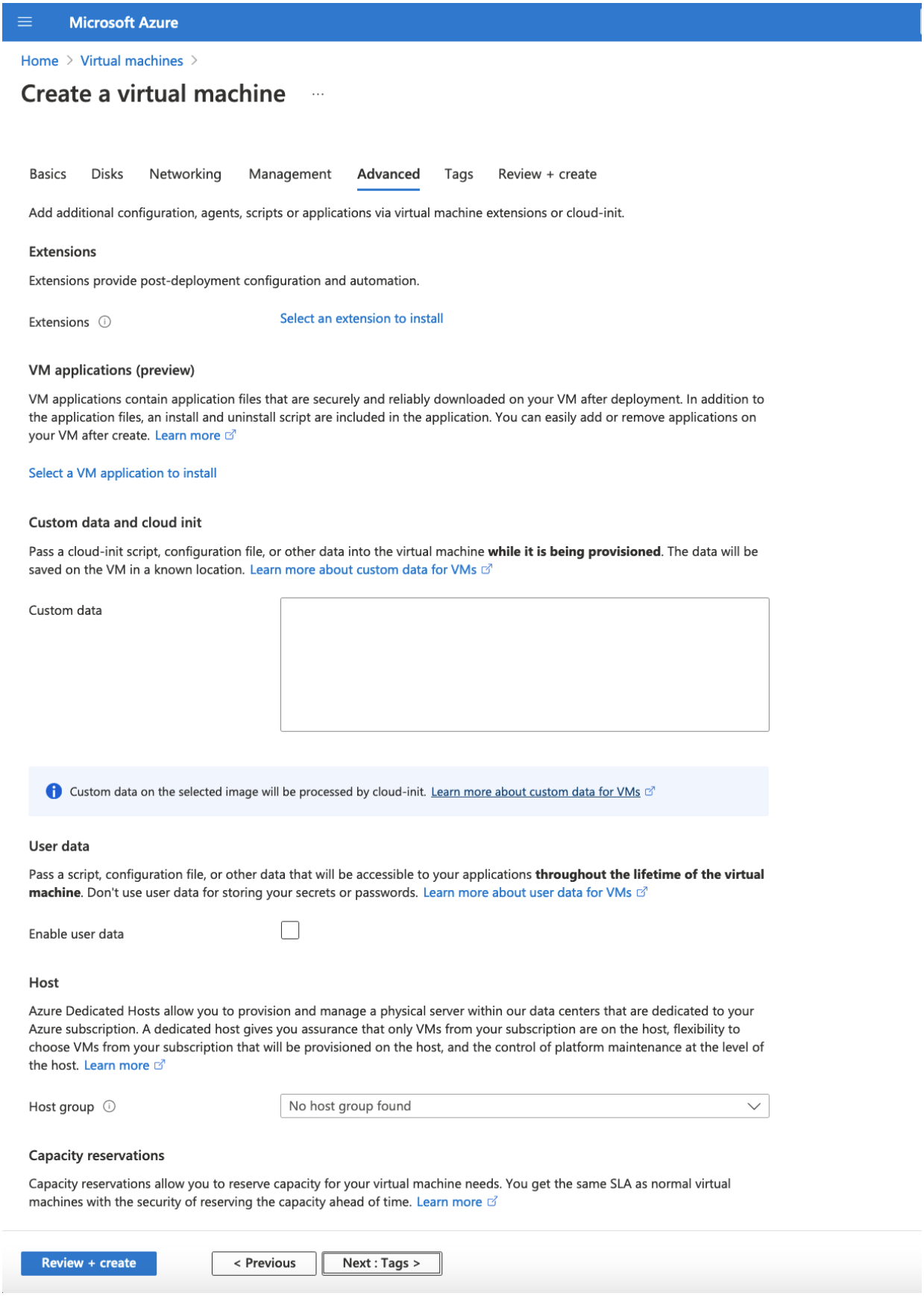
Click on Next > Management

1. Select check box for Identity > System assigned managed identity



Click on Next > Advanced

don't need to change anything


Microsoft Azure
Home > Virtual machines >
Create a virtual machine ...
Basics Disks Networking Management Advanced Tags Review + create
Add additional configuration, agents, scripts or applications via virtual machine extensions or cloud-init.
Extensions
Extensions provide post-deployment configuration and automation.
Extensions
Select an extension to install
VM applications (preview)
VM applications contain application files that are securely and reliably downloaded on your VM after deployment. In addition to the application files, an install and uninstall script are included in the application. You can easily add or remove applications on your VM after create. Learn more
Select a VM application to install
Custom data and cloud init
Pass a cloud-init script, configuration file, or other data into the virtual machine while it is being provisioned. The data will be saved on the VM in a known location. Learn more about custom data for VMs
Custom data
Custom data on the selected image will be processed by cloud-init. Learn more about custom data for VMs
User data
Pass a script, configuration file, or other data that will be accessible to your applications throughout the lifetime of the virtual machine. Don't use user data for storing your secrets or passwords. Learn more about user data for VMs
Enable user data
Host
Azure Dedicated Hosts allow you to provision and manage a physical server within our data centers that are dedicated to your Azure subscription. A dedicated host gives you assurance that only VMs from your subscription are on the host, flexibility to choose VMs from your subscription that will be provisioned on the host, and the control of platform maintenance at the level of the host. Learn more
Host group
No host group found
Capacity reservations
Capacity reservations allow you to reserve capacity for your virtual machine needs. You get the same SLA as normal virtual machines with the security of reserving the capacity ahead of time. Learn more
Review + create
< Previous
Next : Tags >

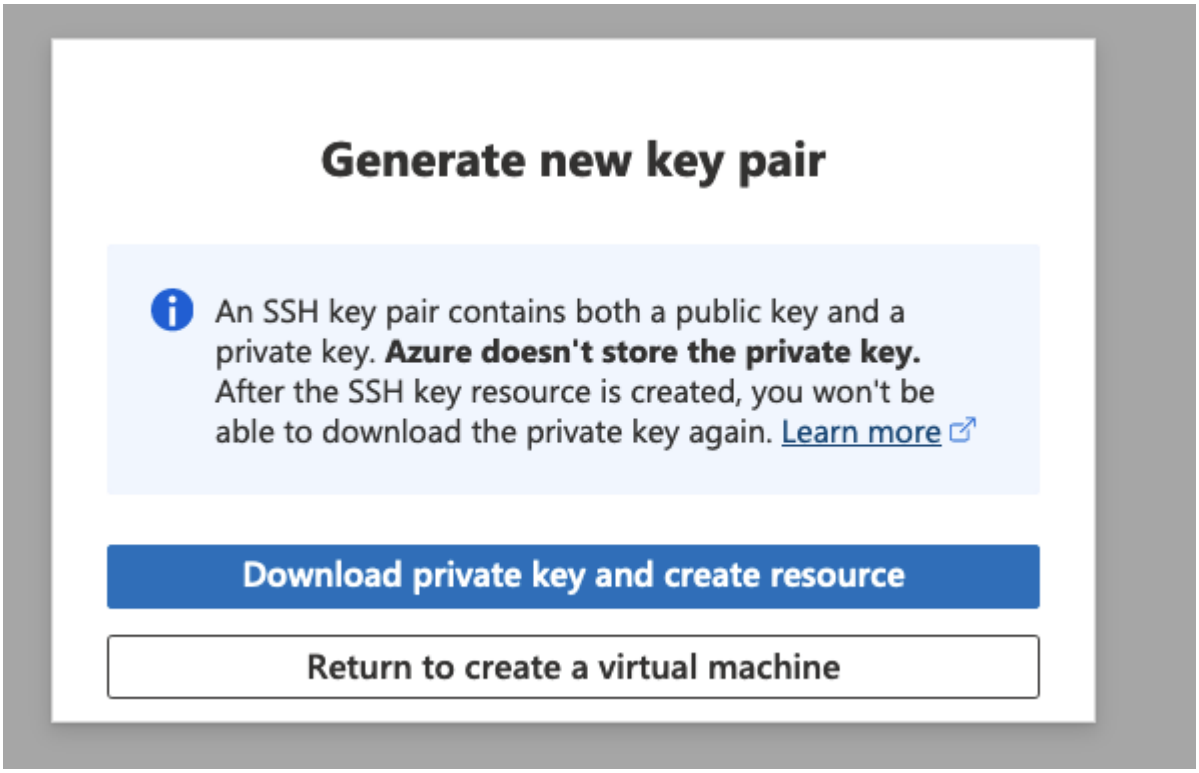
Click on Next > Tags

don't change anything

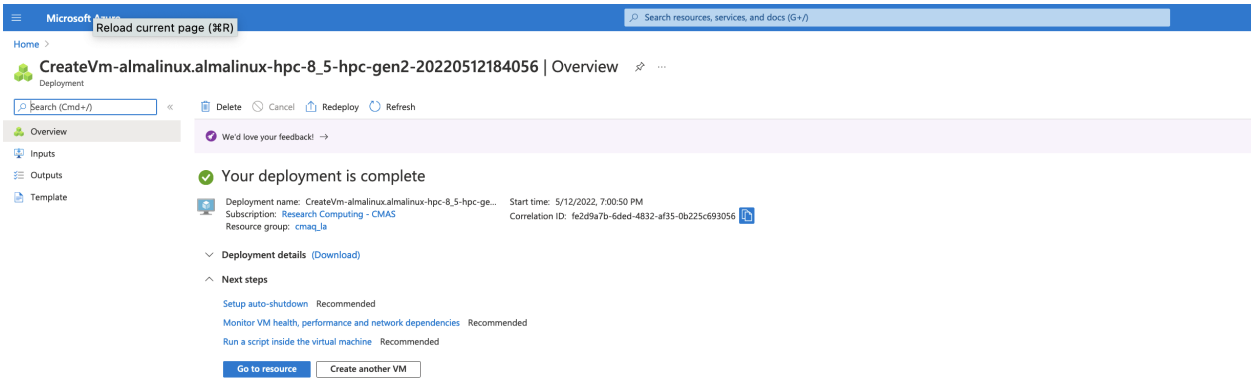
Click on Next > Review and create

!!!azure_web_interface_images/create_virtual_machine_HCv120/Azure_Create_Virtual_Machine_Review_and_crea

Click on download private key and provision resource



Click on Go to Resource once the deployment is completed to get the IP address.



3.3.2 Login to the Virtual Machine

Change the permissions on the public key using command

`chmod 400 HPC-CMAQ-AlmaLinux-HB120_key.pem`

Login to the Virtual Machine using ssh to the IP address using the public key.

`ssh -Y -i ./xxxxxxx_key.pem username@xx.xx.xx.xx`

3.3.3 Mount the disk on the server as /shared using the instructions on the following link:

Mount Disk on Azure Linux Virtual Machine

Find the disk

```
lsblk -o NAME,HCTL,SIZE,MOUNTPOINT | grep -i "sd"
```

Output:

```
sda     0:0:0:0       30G
├─sda1                500M /boot
├─sda2                 29G /
├─sda14                 4M
└─sda15               495M /boot/efi
sdb     0:0:0:1      480G
└─sdb1                480G /mnt
sdc     1:0:0:0        1T
```

In the above case, the 1 Terrabyte (1T) disk was added as sdc

Format the disk

```
sudo parted /dev/sdc --script mklabel gpt mkpart xfspart xfs 0% 100%
sudo mkfs.xfs /dev/sdc1
sudo partprobe /dev/sdc1
```

Mount the disk

```
sudo mkdir /shared
```

Use mount to mount the filesystem

```
sudo mount /dev/sdc1 /shared
```

Persist the mount

```
sudo blkid
```

Output

```
/dev/sdb1: UUID="109f262f-36bb-431d-b1c0-9a9dad39b894" BLOCK_SIZE="4096" TYPE="ext4"␣
↪PARTUUID="111c5d20-01"
/dev/sda1: UUID="9643d043-09b2-4bfa-9842-079b985d4d15" BLOCK_SIZE="512" TYPE="xfs"␣
↪PARTUUID="cd39399b-65c3-4a21-89d1-129b241d7e4d"
/dev/sda2: UUID="943e4d7b-9391-47b5-916c-f51afcdc512f" BLOCK_SIZE="512" TYPE="xfs"␣
↪PARTUUID="0a95f633-83e4-41bf-b6b1-da9ebc5bf5d7"
/dev/sda15: SEC_TYPE="msdos" UUID="3DB8-F6B8" BLOCK_SIZE="512" TYPE="vfat" PARTLABEL=
↪"EFI System Partition" PARTUUID="75d05630-884d-4f11-abb5-6ce3331c7528"
/dev/sdc1: UUID="09e461c7-2ac6-4e07-b3c8-6e7f593dfba2" BLOCK_SIZE="4096" TYPE="xfs"␣
↪PARTLABEL="xfspart" PARTUUID="649e7f66-057a-4460-ab92-661542ae9196"
/dev/sda14: PARTUUID="14abf57d-419d-4263-8078-aa7a849c1d58"
```

Edit fstab

Next, open the /etc/fstab file in a text editor as follows:

```
sudo nano /etc/fstab
```

In this example, use the UUID value for the /dev/sdc1 device that was created in the previous steps, and the mountpoint of /shared. Add the following line to the end of the /etc/fstab file:

```
UUID=09e461c7-2ac6-4e07-b3c8-6e7f593dfba2 /shared xfs defaults,nofail 1 2
```

Verify the /shared directory

Change directories and verify that you see the /shared directory with Size of 1T

```
cd /shared
```

```
df -h
```

Output

```
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        213G     0  213G   0% /dev
tmpfs           213G     0  213G   0% /dev/shm
tmpfs           213G   17M  213G   1% /run
tmpfs           213G     0  213G   0% /sys/fs/cgroup
/dev/sda2        30G   11G   19G  37% /
/dev/sda1       495M  193M  302M  39% /boot
/dev/sda15      495M  5.8M  489M   2% /boot/efi
/dev/sdb1       472G   73M  448G   1% /mnt
tmpfs            43G     0   43G   0% /run/user/1000
/dev/sdc1       1.0T  7.2G 1017G   1% /shared
```

Create subdirectories on /shared

Create a /shared/build, /shared/data and /shared/cyclecloud-cmaq directory and change the permissions from root to your username.

```
cd /
sudo chown azureuser shared
sudo chgrp azureuser shared

cd /shared
mkdir build
mkdir data
mkdir cyclecloud-cmaq
```

3.3.4 Alternatively, you can create an nvme stripped disk that has faster performance.

```
mkdir -p /mnt/nvme

mdadm --create /dev/md10 --level 0 --raid-devices 2 /dev/nvme0n1 /dev/nvme1n1

mkfs.xfs /dev/md10

mount /dev/md10 /mnt/nvme

chmod 1777 /mnt/nvme
```

That should create a file system with about 1.8TiB

3.3.5 Obtain the Cyclecloud-cmaq code from github

Load the git module

```
module load module-git
```

If you do not see git available as a module, you may need to install it as follows:

```
sudo yum install git
```

Load the openmpi module

```
module load mpi/openmpi-4.1.1
```

Install Cycle Cloud Repo

```
git clone -b 5.3.3 https://github.com/CMASCenter/cyclecloud-cmaq.git
```

Install and build netcdf C, netcdf Fortran, I/O API, and CMAQ

```
cd /shared/cyclecloud-cmaq
```

Install netcdf-C and netcdf-Fortran

```
./gcc_install.csh
```

If successful, you will see the following output, that at the bottom shows what versions of the netCDF library were installed.

```
+-------------------------------------------------------------+
| Congratulations! You have successfully installed the netCDF |
| Fortran libraries.                                          |
|                                                             |
| You can use script "nf-config" to find out the relevant     |
| compiler options to build your application. Enter           |
|                                                             |
|     nf-config --help                                        |
|                                                             |
| for additional information.                                 |
|                                                             |
| CAUTION:                                                    |
|                                                             |
| If you have not already run "make check", then we strongly  |
| recommend you do so. It does not take very long.            |
|                                                             |
| Before using netCDF to store important data, test your      |
| build with "make check".                                    |
|                                                             |
| NetCDF is tested nightly on many platforms at Unidata       |
| but your platform is probably different in some ways.       |
|                                                             |
| If any tests fail, please see the netCDF web site:          |
| https://www.unidata.ucar.edu/software/netcdf/                |
|                                                             |
| NetCDF is developed and maintained at the Unidata Program   |
| Center. Unidata provides a broad array of data and software |
| tools for use in geoscience education and research.         |
| https://www.unidata.ucar.edu                                 |
+-------------------------------------------------------------+

make[3]: Leaving directory '/shared/build/netcdf-fortran-4.5.4'
make[2]: Leaving directory '/shared/build/netcdf-fortran-4.5.4'
make[1]: Leaving directory '/shared/build/netcdf-fortran-4.5.4'
netCDF 4.8.1
netCDF-Fortran 4.5.4
```

Install I/O API

```
./gcc_ioapi.csh
```

Find what operating system is on the system:

```
cat /etc/os-release
```

Output

```
NAME="AlmaLinux"
VERSION="8.5 (Arctic Sphynx)"
ID="almalinux"
ID_LIKE="rhel centos fedora"
VERSION_ID="8.5"
PLATFORM_ID="platform:el8"
PRETTY_NAME="AlmaLinux 8.5 (Arctic Sphynx)"
ANSI_COLOR="0;34"
CPE_NAME="cpe:/o:almalinux:almalinux:8::baseos"
HOME_URL="https://almalinux.org/"
DOCUMENTATION_URL="https://wiki.almalinux.org/"
BUG_REPORT_URL="https://bugs.almalinux.org/"

ALMALINUX_MANTISBT_PROJECT="AlmaLinux-8"
ALMALINUX_MANTISBT_PROJECT_VERSION="8.5"
```

3.3.6 Change shell to use tcsh

```
sudo usermod -s /bin/tcsh azureuser
```

Log out and then log back in to have the shell take effect.

Copy a file to set paths

```
cd /shared/cyclecloud-cmaq
```

```
cp dot.cshrc.vm ~/.cshrc
```

3.3.7 Create Environment Module for Libraries

There are two steps required to create your own custome module:

1. write a module file
2. add a line to your ~/.cshrc to update the MODULEPATH

Create a new custom module that will be loaded including any dependencies using the following command:

```
module load ioapi-3.2_20200828/gcc-9.2.1-netcdf
```

Step 1: Create the module file.

First, create a path to store the module file. The path must contain /Modules/modulefiles/ and should have the general form //Modules/modulefiles// where is typically numerical and is the actual module file.

```
mkdir /shared/build/Modules/modulefiles/ioapi-3.2_20200828
```

Next, create the module file and save it in the directory above.

```
cd /shared/build/Modules/modulefiles/ioapi-3.2_20200828
vim gcc-9.2.1-netcdf
```

Contents of gcc-9.2.1-netcdf:

```
#%Module

proc ModulesHelp { } {
   puts stderr "This module adds ioapi-3.2_20200828/gcc-9.2.1 to your path"
}

module-whatis "This module adds ioapi-3.2_20200828/gcc-9.2.1 to your path\n"

set basedir "/shared/build/ioapi-3.2_branch_20200828/"
prepend-path PATH "${basedir}/Linux2_x86_64gfort"
prepend-path LD_LIBRARY_PATH "${basedir}/ioapi/fixed_src"
module load mpi/openmpi-4.1.1
module load gcc-9.2.1
module load netcdf-4.8.1/gcc-9.2.1
```

The example module file above sets two evironment variables and loads two system modules and a custom module (that we also need to define).

The modules update the PATH and LD_LIBRARY_PATH.

Now create the custom module to define the netCDF libraries that were used to build I/O API.

```
mkdir /shared/build/Modules/modulefiles/netcdf-4.8.1
vim gcc-9.2.1
```

Contents of gcc-9.2.1

```
#%Module

proc ModulesHelp { } {
   puts stderr "This module adds netcdf-4.8.1/gcc-9.2.1 to your path"
}

module-whatis "This module adds netcdf-4.8.1/gcc-9.2.1 to your path\n"

set basedir "/shared/build/netcdf"
prepend-path PATH "${basedir}/bin"
prepend-path LD_LIBRARY_PATH "${basedir}/lib"
module load mpi/openmpi-4.1.1
module load gcc-9.2.1
```

Step 2: Add the module path to MODULEPATH.

Now that the two custom module files have been created, add the following line to your ~/.cshrc file so that they can be found:

```
module use --append /shared/build/Modules/modulefiles
```

Step 3: View the modules available after creation of the new module

The module avail command shows the paths to the module files on a given cluster.

```
module avail
```

Step 4: Load the new module

```
module load ioapi-3.2_20200828/gcc-9.2.1-netcdf
```

Output:

```
Loading ioapi-3.2_20200828/gcc-9.2.1-netcdf
  Loading requirement: gcc-9.2.1 mpi/openmpi-4.1.1 netcdf-4.8.1/gcc-9.2.1
```

Verify that the libraries required for netCDF and I/O API have been added to the $LD_LIBRARY_PATH environment variable

```
echo $LD_LIBRARY_PATH
```

Output:

```
/shared/build/netcdf/lib:/opt/openmpi-4.1.1/lib:/opt/rh/gcc-toolset-9/root/lib64:/shared/
↪build/ioapi-3.2_branch_20200828//ioapi/fixed_src::
```

Verify that the I/O API bin directory and netCDF bin directory that you specified in the custom module has been added to the $PATH environment variable

```
 echo $PATH
```

Output

```
/shared/build/netcdf/bin:/opt/openmpi-4.1.1/bin:/opt/rh/gcc-toolset-9/root/bin:/shared/
↪build/ioapi-3.2_branch_20200828//Linux2_x86_64gfort:/usr/share/Modules/bin:/usr/local/
↪bin:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/slurm/bin/:/usr/local/bin:/opt/slurm/bin/:/
↪usr/local/bin
```

see Custom-Modules from Princeton Research Computing

3.3.8 Install and Build CMAQ

`./gcc_cmaq.csh`

Verfify that the executable was successfully built.

`ls /shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts/BLD_CCTM_v533_gcc/*.exe`

Output

```
/shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts/BLD_CCTM_v533_gcc/CCTM_v533.exe
```

3.3.9 Copy the run scripts from the repo to the run directory

`cd /shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts`

`cp /shared/cyclecloud-cmaq/run_scripts/HB120v3/*pe.csh .`

List the scripts available

`ls -rlt *pe.csh*`

Output

```
run_cctm_2016_12US2.90pe.csh
run_cctm_2016_12US2.36pe.csh
run_cctm_2016_12US2.16pe.csh
run_cctm_2016_12US2.120pe.csh
```

3.3.10 Download the Input data from the S3 Bucket

Install aws command line

see Install AWS CLI

```
cd /shared/build
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

Install the input data using the s3 script

```
cd /shared/cyclecloud-cmaq/s3_scripts/
./s3_copy_nosign_conus_cmas_opendata_to_shared.csh
```

Note, this Virtual Machine does not have Slurm installed or configured.

3.3.11 Run CMAQ interactively using the following command:

```
cd /shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts
./run_cctm_2016_12US2.120pe.csh |& tee ./run_cctm_2016_12US2.120pe.log
```

When the run has completed, record the timing of the two day benchmark.

```
tail -n 30 run_cctm_2016_12US2.120pe.log
```

Output:

```
=====================================
  ***** CMAQ TIMING REPORT *****
=====================================
Start Day: 2015-12-22
End Day:   2015-12-23
Number of Simulation Days: 2
Domain Name:               12US2
Number of Grid Cells:      3409560  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       120
   All times are in seconds.

Num  Day        Wall Time
01   2015-12-22   2458.35
02   2015-12-23   2205.08
```

(continues on next page)

(continued from previous page)

```
     Total Time = 4663.43
      Avg. Time = 2331.71
```

If runs are submitted immediately after a successful completion of a run, then you may skey the scaling results. It would be ideal to wait 30 minutes before running a second job.

Run second job interactively using the following command:

```
./run_cctm_2016_12US2.90pe.csh | & tee ./run_cctm_2016_12US2.90pe.log
```

Output

```
================================
  ***** CMAQ TIMING REPORT *****
================================
Start Day: 2015-12-22
End Day:   2015-12-23
Number of Simulation Days: 2
Domain Name:               12US2
Number of Grid Cells:      3409560  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       90
   All times are in seconds.

Num  Day        Wall Time
01   2015-12-22   2786.21
02   2015-12-23   2417.74
     Total Time = 5203.95
      Avg. Time = 2601.97
```

3.4 Advanced Tutorial

- Learn how to install CMAQ software and underlying libraries, copy input data, and run CMAQ.

3.4.1 Create Cyclecloud CMAQ Cluster

Documentation for Azure CycleCloud Documentation

Configure the Cycle Cloud Application Host using the Azure Portal

Log into the Azure Portal

In the search bar, enter “Marketplace”, Click on Marketplace Icon.

In the Marketplace search bar, enter “CycleCloud”.

Click on the heart in the Azure CycleCloud box to add this as a favorite resource.

Use the Create pulldown menu to select `Azure CycleCloud 8.2`

Customize your Host Virtual Machine for the CycleCloud Application

1. Choose your Subscription
2. Select or create a new Resource Group that your CycleCloud instance will run in: note, leave this blank initially, as it will be named after the instance name below by appending _group to the instance name
3. Name your CycleCloud instance using Virtual Machine name : example name: CycleCloudHost
4. Select Region: example name: US East
5. Verify Image is Azure CycleCloud 8.2 - Gen 1
6. Select Size, click on see all sizes, enter D4s into the search button and select Standard_D4s_v3- 4cpus, 16GiB memory ($140.16/month)
7. Select Authentication Type `SSH public key`
8. Create the Username that you will use to log into the instance: example name: azureuser
9. SSH public key source - select `Generate new key pair`
10. Select the Management tab and enable `System assigned managed identity`
11. Click on the `Review` button and then the `Create` button

When a pop-up menu is displayed: click on option to Download private key and create resource.

You will see a message `... Deployment is in progress`

Wait until the resource has been deployed before proceeding to the next step.

Figure 1. Create a virtual Machine - Customize Host Virtual Machine Note: this virtual machine will be used to host the CycleCloud Application that is used to create the Cycle Cloud Cluster from it's Web located at: UI https://IP-address/home

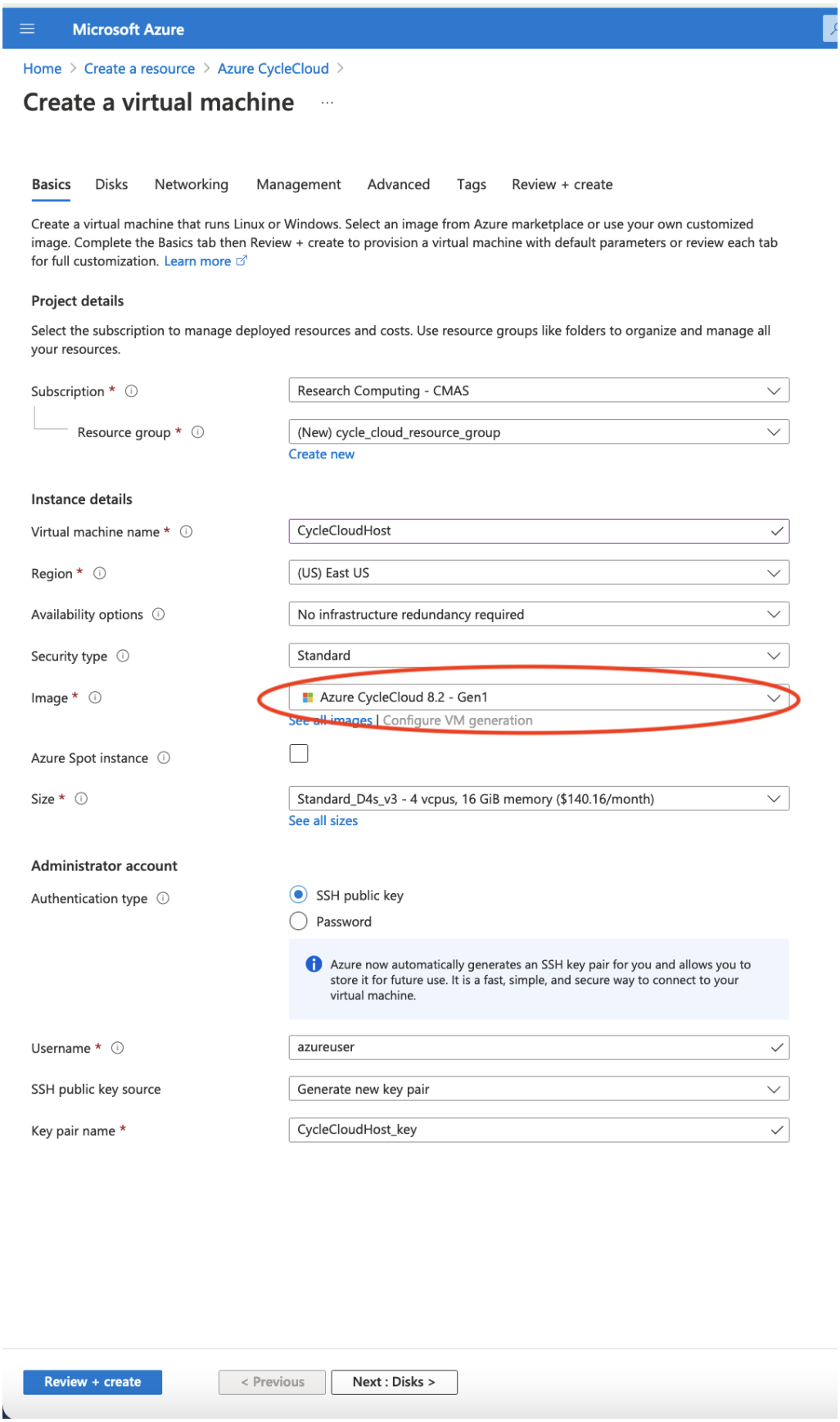

Microsoft Azure
Home > Create a resource > Azure CycleCloud >
Create a virtual machine
Basics Disks Networking Management Advanced Tags Review + create
Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. Learn more
Project details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription * Research Computing - CMAS
Resource group * (New) cycle_cloud_resource_group
Create new
Instance details
Virtual machine name * CycleCloudHost
Region * (US) East US
Availability options No infrastructure redundancy required
Security type Standard
Image * Azure CycleCloud 8.2 - Gen1
See all images | Configure VM generation
Azure Spot instance
Size * Standard_D4s_v3 - 4 vcpus, 16 GiB memory ($140.16/month)
See all sizes
Administrator account
Authentication type SSH public key
Password
Azure now automatically generates an SSH key pair for you and allows you to store it for future use. It is a fast, simple, and secure way to connect to your virtual machine.
Username * azureuser
SSH public key source Generate new key pair
Key pair name * CycleCloudHost_key
Review + create
< Previous
Next : Disks >

Figure 2. Select Disks for the Azure Virtual Machine - use default options

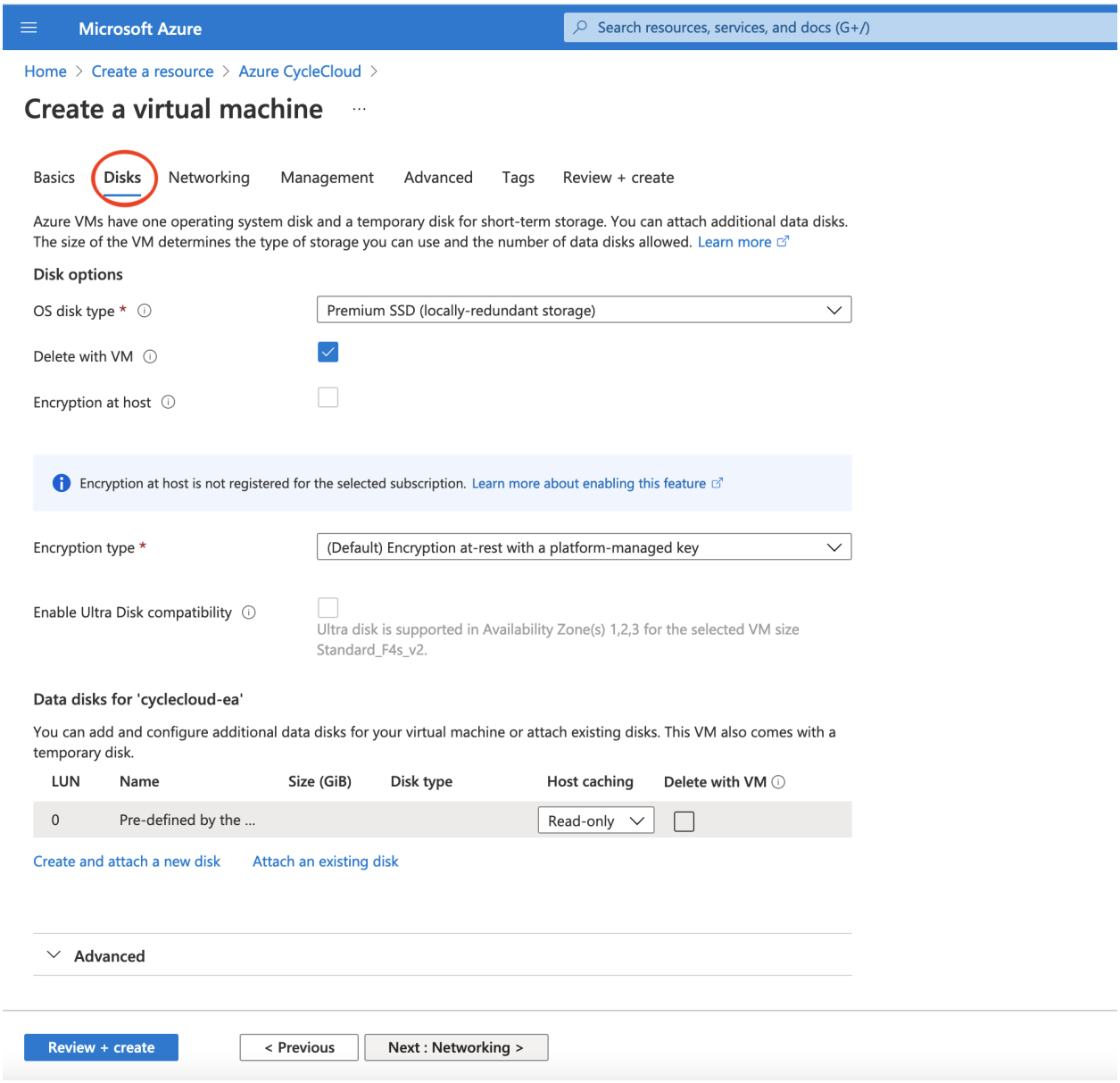


Figure 3. Select Network Interface for the Azure Virtual Machine - use default options

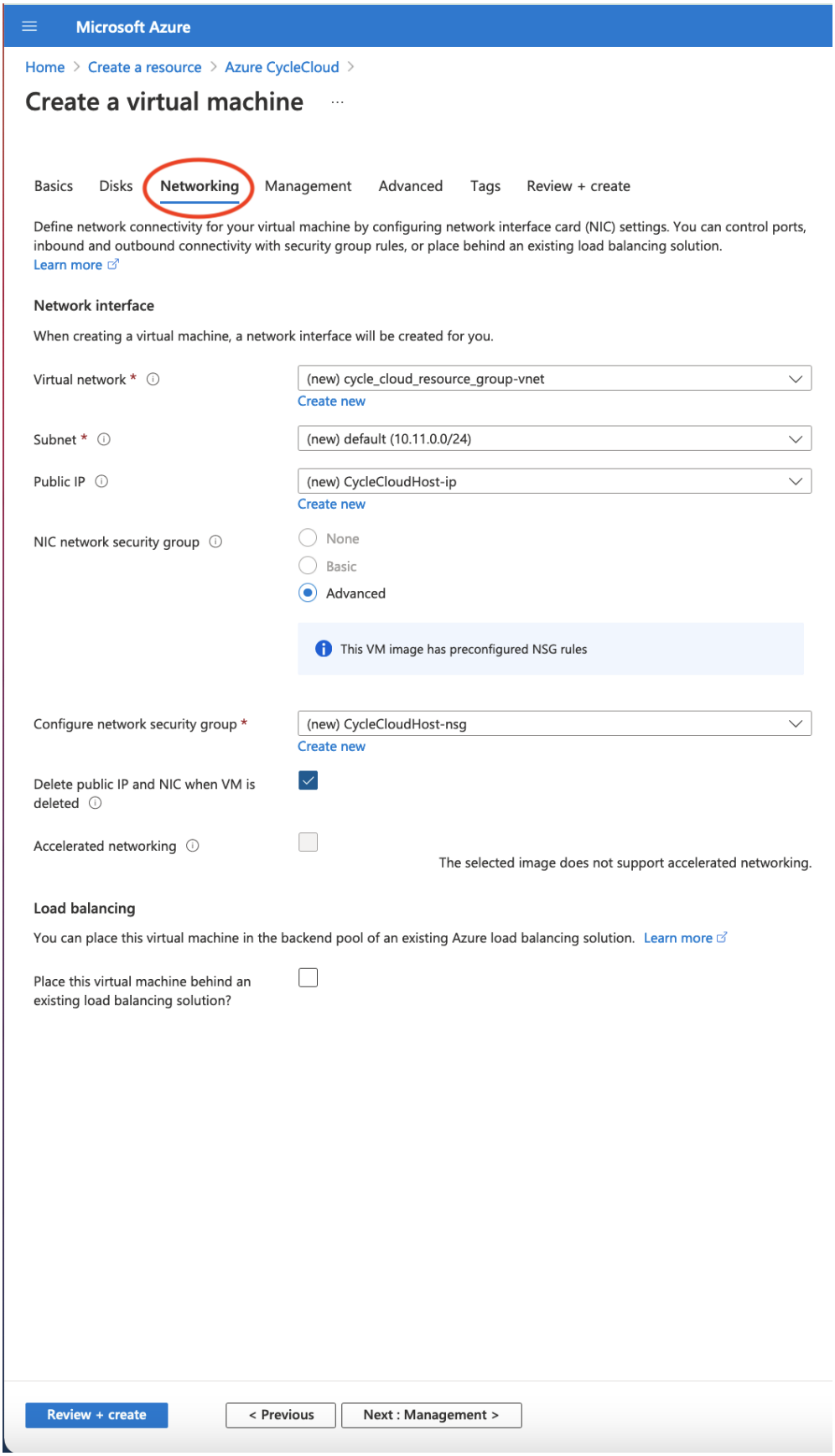

Microsoft Azure
Home > Create a resource > Azure CycleCloud >
Create a virtual machine
Basics
Disks
Networking
Management
Advanced
Tags
Review + create
Define network connectivity for your virtual machine by configuring network interface card (NIC) settings. You can control ports, inbound and outbound connectivity with security group rules, or place behind an existing load balancing solution.
Learn more
Network interface
When creating a virtual machine, a network interface will be created for you.
Virtual network *
(new) cycle_cloud_resource_group-vnet
Create new
Subnet *
(new) default (10.11.0.0/24)
Public IP
(new) CycleCloudHost-ip
Create new
NIC network security group
None
Basic
Advanced
This VM image has preconfigured NSG rules
Configure network security group *
(new) CycleCloudHost-nsg
Create new
Delete public IP and NIC when VM is deleted
Accelerated networking
The selected image does not support accelerated networking.
Load balancing
You can place this virtual machine in the backend pool of an existing Azure load balancing solution. Learn more
Place this virtual machine behind an existing load balancing solution?
Review + create
< Previous
Next : Management >

Figure 4. Select System assigned Managed Identity

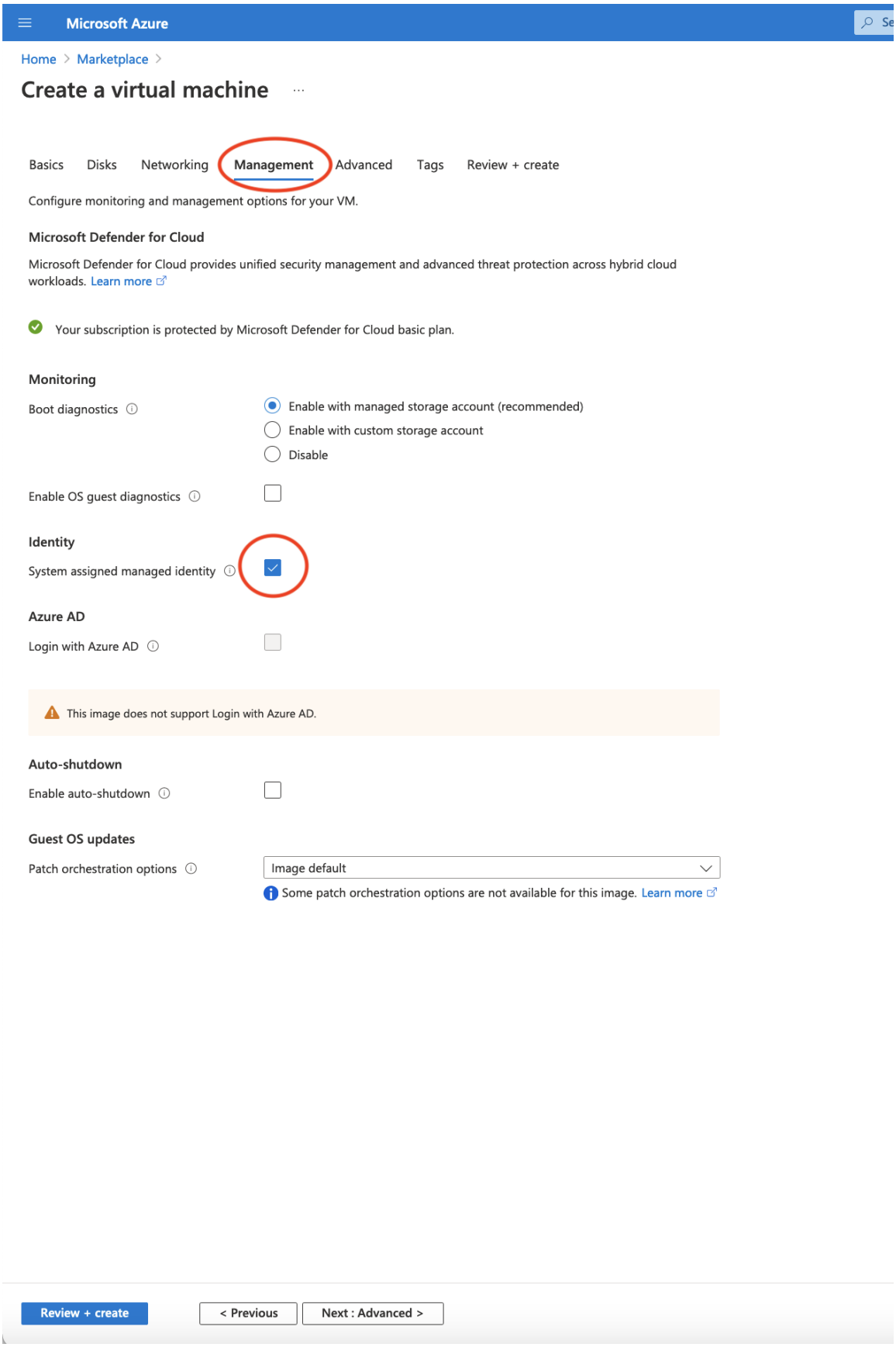

Microsoft Azure
Home > Marketplace >
Create a virtual machine
Basics
Disks
Networking
Management
Advanced
Tags
Review + create
Configure monitoring and management options for your VM.
Microsoft Defender for Cloud
Microsoft Defender for Cloud provides unified security management and advanced threat protection across hybrid cloud workloads. Learn more
Your subscription is protected by Microsoft Defender for Cloud basic plan.
Monitoring
Boot diagnostics
Enable with managed storage account (recommended)
Enable with custom storage account
Disable
Enable OS guest diagnostics
Identity
System assigned managed identity
Azure AD
Login with Azure AD
This image does not support Login with Azure AD.
Auto-shutdown
Enable auto-shutdown
Guest OS updates
Patch orchestration options
Image default
Some patch orchestration options are not available for this image. Learn more
Review + create
< Previous
Next : Advanced >

Figure 5. Create a Virtual Machine - Deployment is in Progress

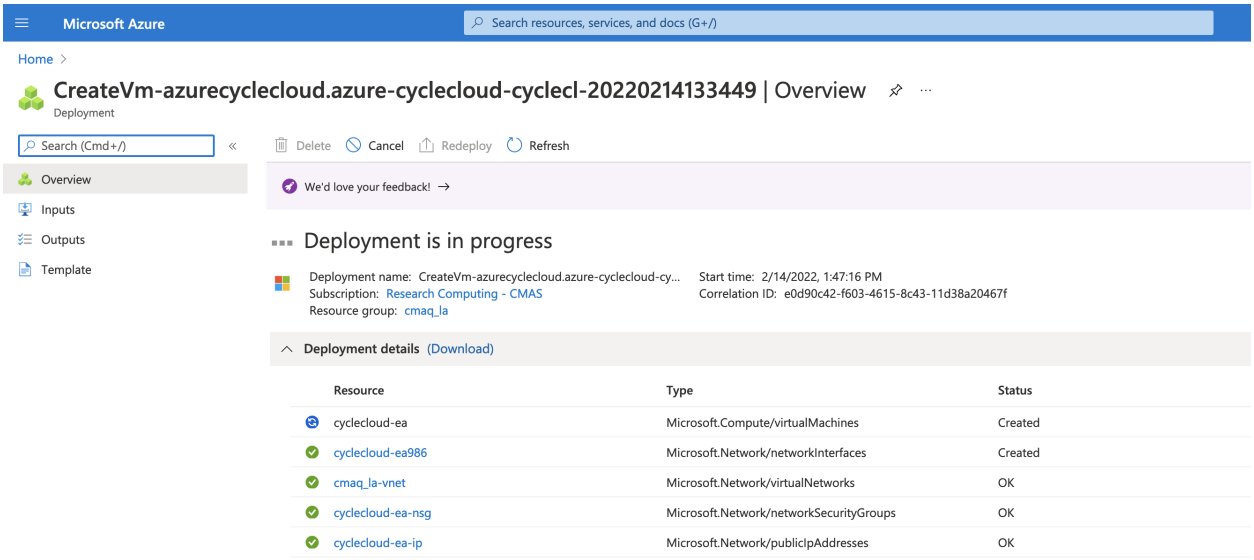
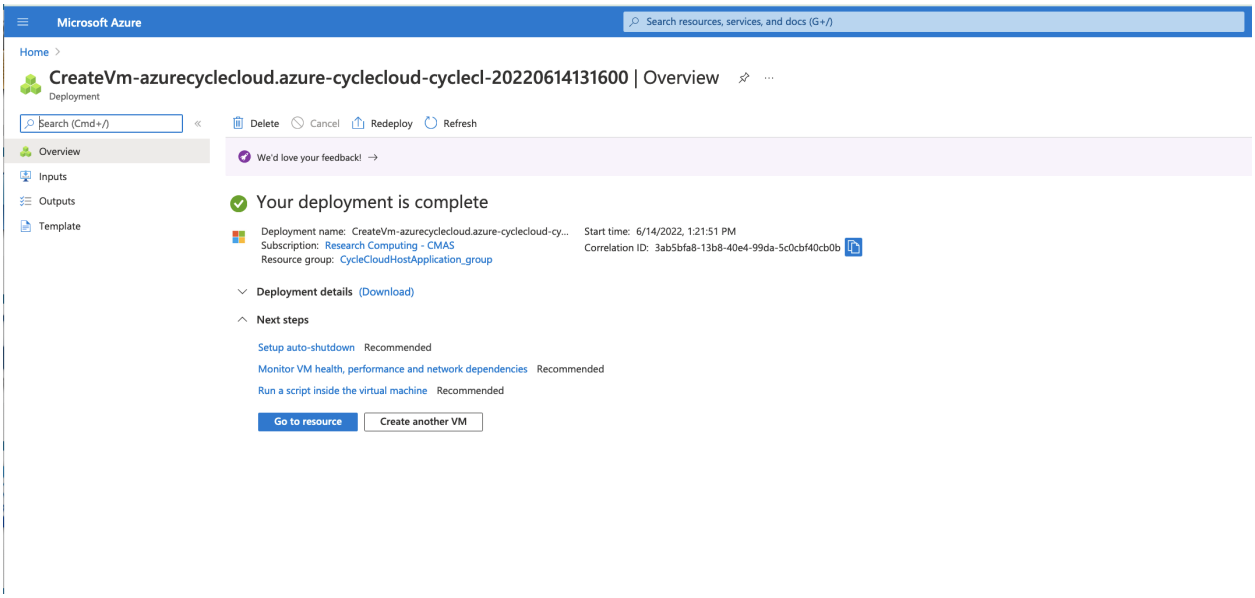


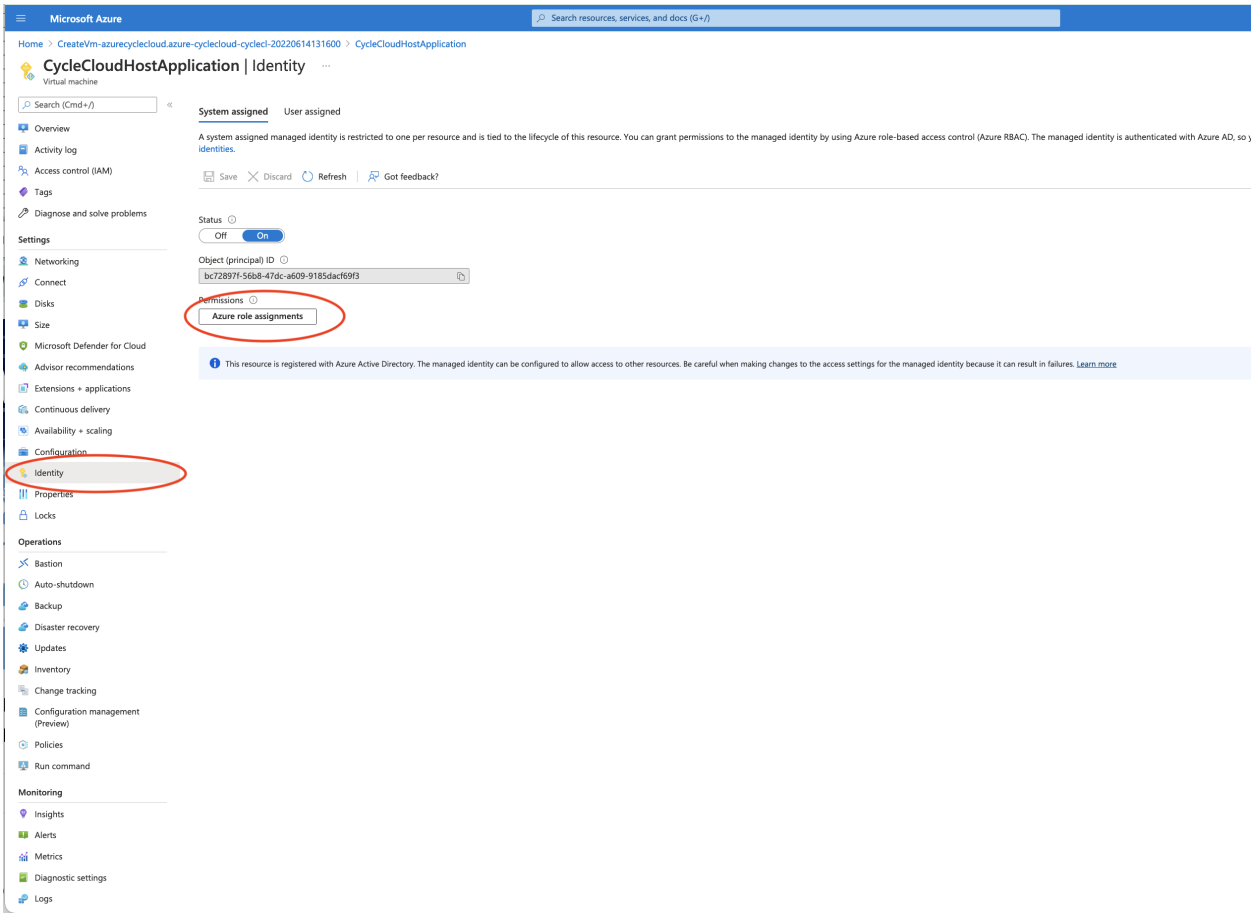
Figure 6. Your Deployment is complete - click on blue button `Go to resource`



After the CycleCloud Host Machine has been deployed click on `Go to resource`

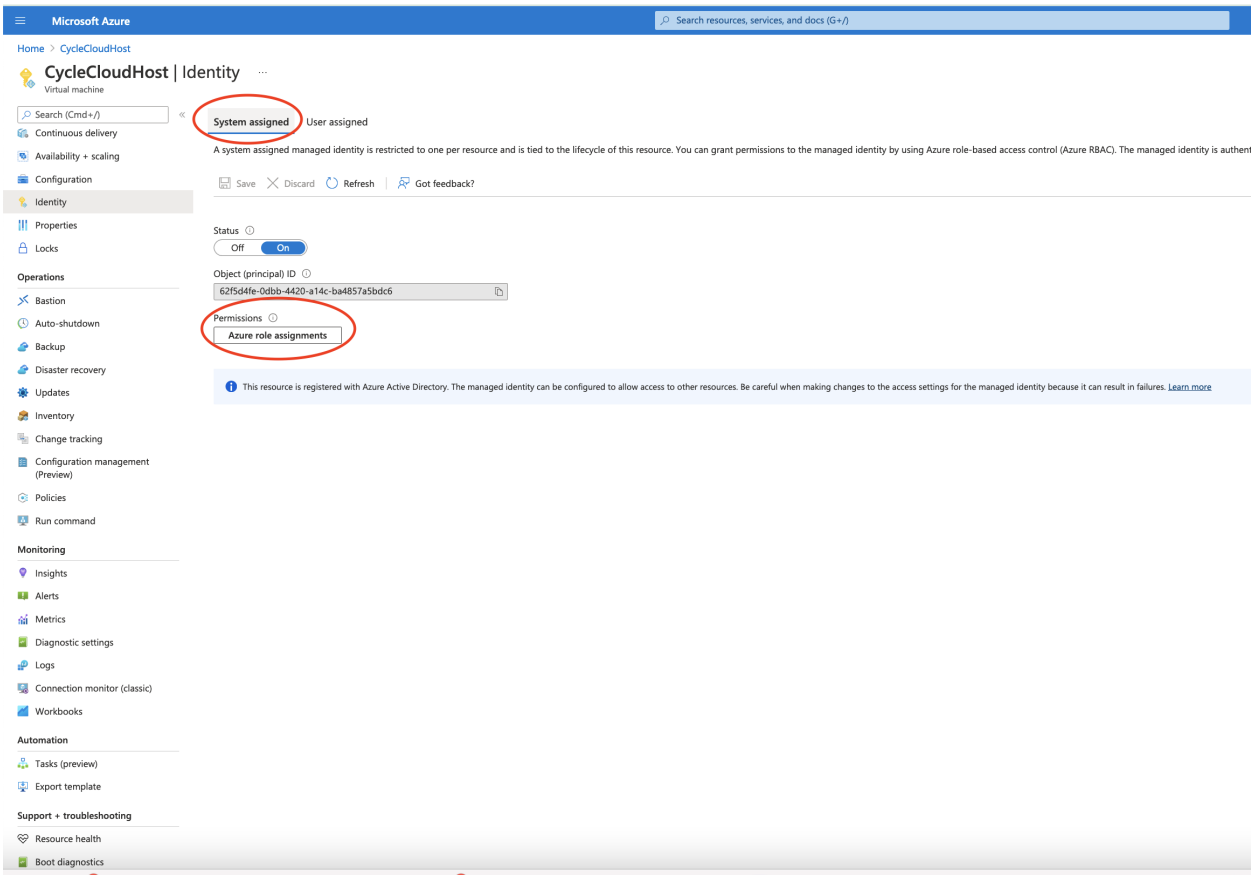
Add Contributor Role to Virtual Machine -

Figure 7. Click on Identity Icon on left side of CycleCloudHost Application Virtual Machine



Click on the Identity Menu on the left side of the newly created virtual machine. Make sure you select the System Assigned Tab at the top of the window. Click on the button `Azure Role Assignments`

Figure 8. Make sure you select the System Assigned Tab at the top of the window.



On the Azure role assignments window click on the + Add role assignment(Preview)

Figure 9. Add System Assigned Role Assignment - Management Identity

Click on Azure role assignements Search for Managed Identity Operator

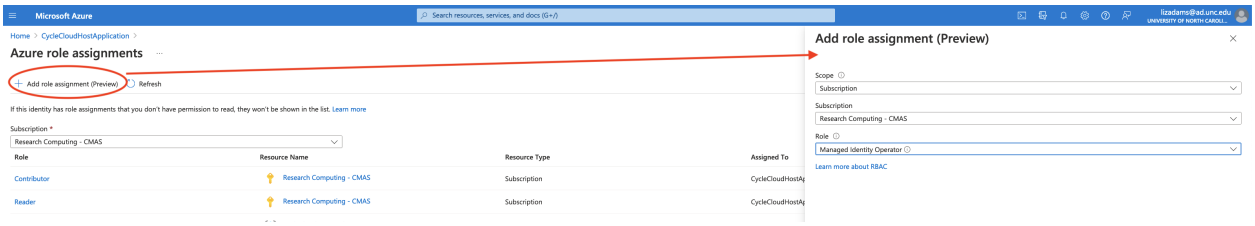


Figure 10. Add Role Assignment

1. Click Identity Icon under Settings on the left side menu
2. Click Azure role assignments
3. Click Add role assignment
4. Search for Contributor

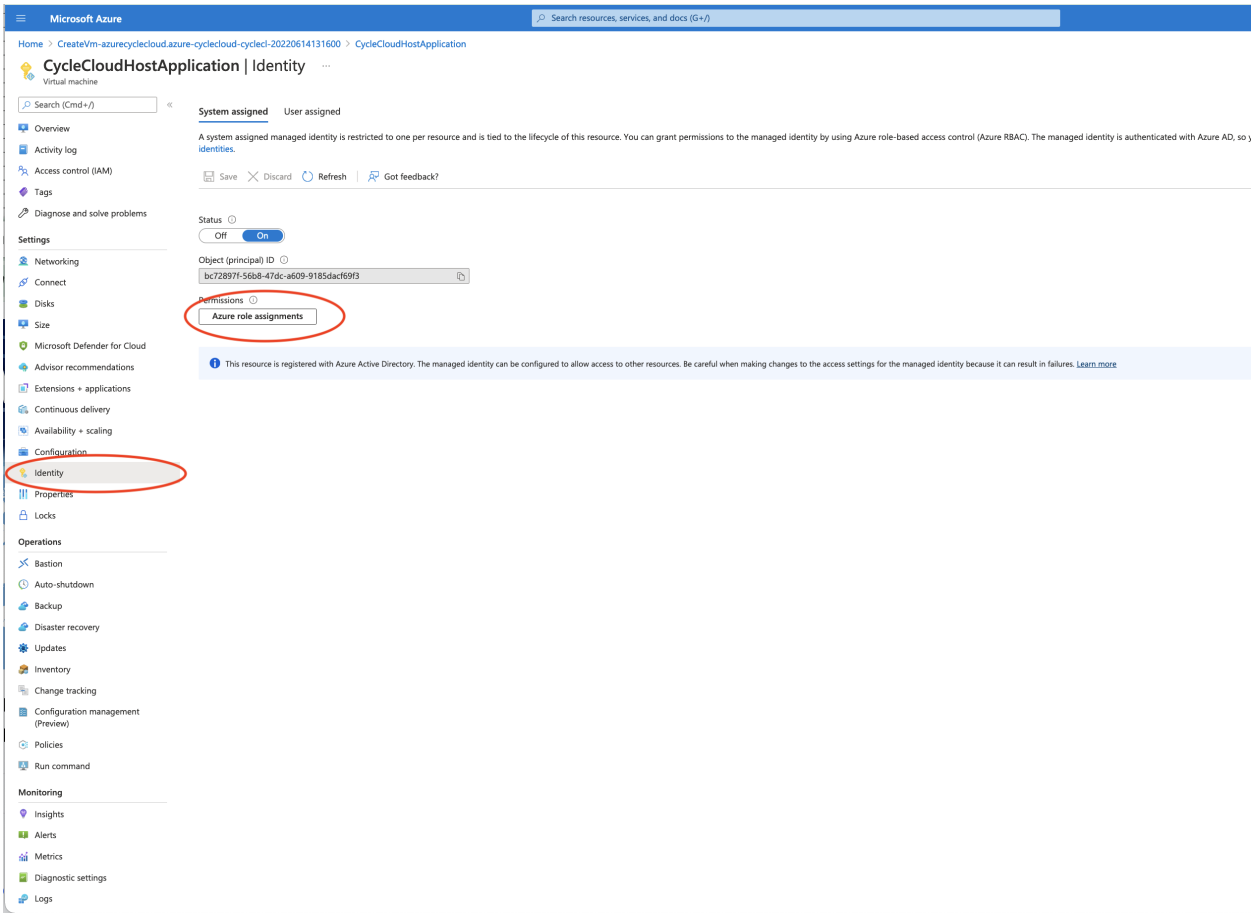
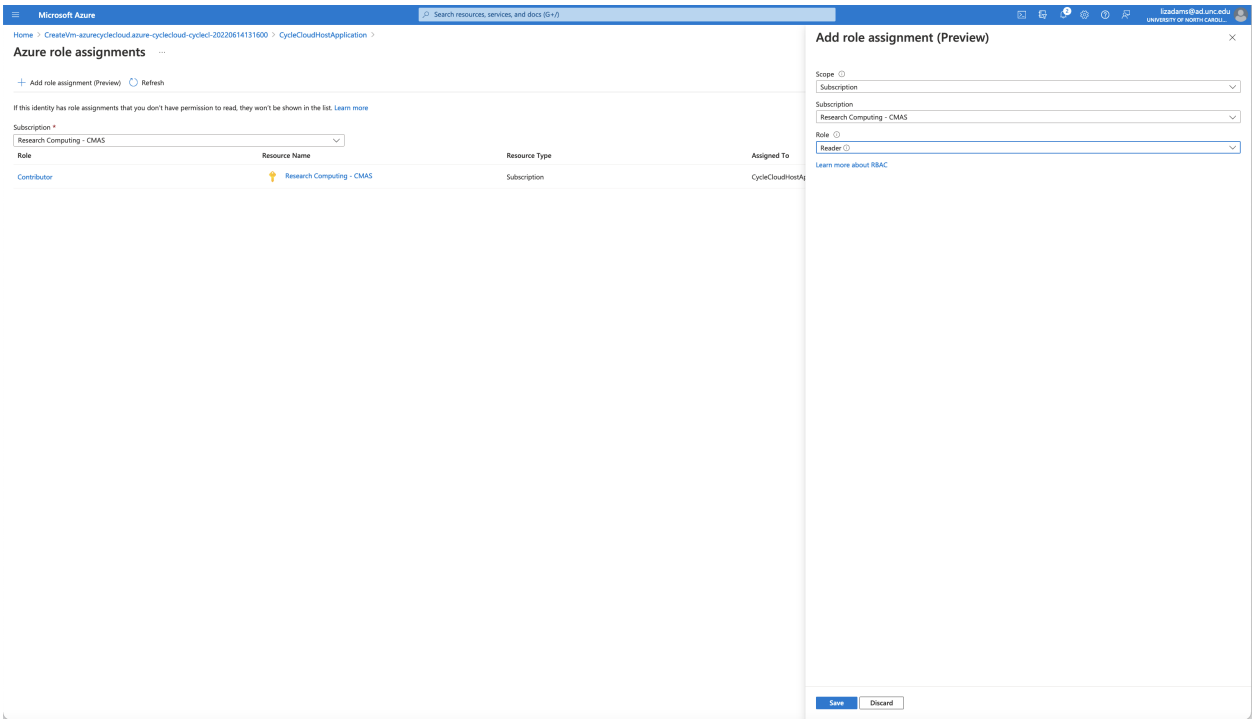


Figure 11. Add Reader Role to Virtual Machine

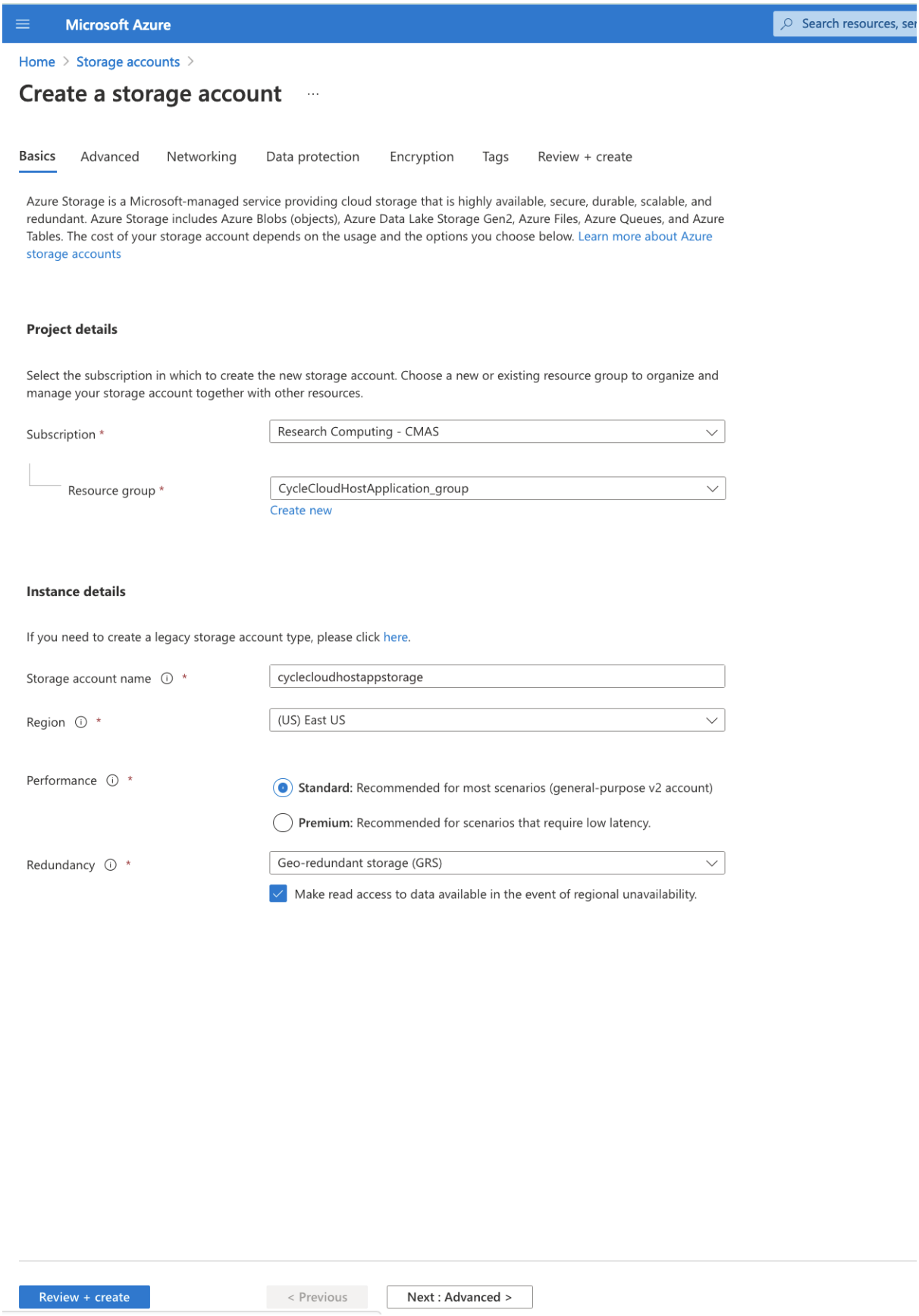


Create Storage Account

In the search bar, enter Storage Account, the select + Create Storage Account

Select the resource group associated with the CycleCloudHost that you created CycleCloudHost_group Select a lowercase name Then switch from the Basics tab to the Advanced Tab Uncheck the box next to Enable blob public access. Click Review and Create After the verification passed message is received Click Create

Figure 12. Azure Create Storage Account Details


Microsoft Azure
Search resources, ser
Home > Storage accounts >
Create a storage account
Basics
Advanced
Networking
Data protection
Encryption
Tags
Review + create
Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. Learn more about Azure storage accounts
Project details
Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.
Subscription *
Research Computing - CMAS
Resource group *
CycleCloudHostApplication_group
Create new
Instance details
If you need to create a legacy storage account type, please click here.
Storage account name *
cyclecloudhostappstorage
Region *
(US) East US
Performance *
Standard: Recommended for most scenarios (general-purpose v2 account)
Premium: Recommended for scenarios that require low latency.
Redundancy *
Geo-redundant storage (GRS)
Make read access to data available in the event of regional unavailability.
Review + create
< Previous
Next : Advanced >

For Redundancy choose Local Redundant Storage instead of Geo-Redundant Storage to reduce costs.

Figure 13. Azure Storage Account disable Public Blob Access

Disable the Public Blob Access by unclicking the box next to `Enable blob public access`

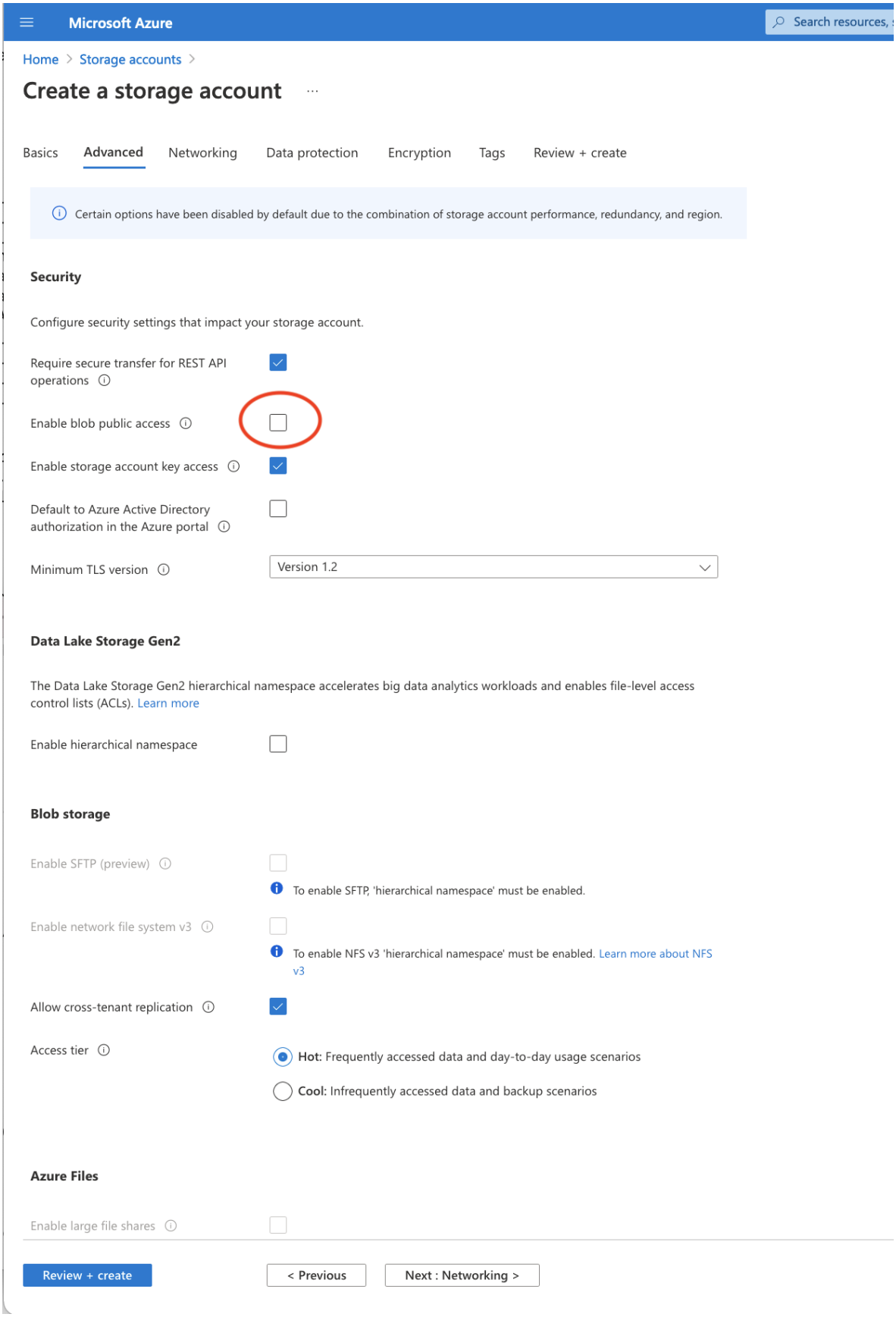
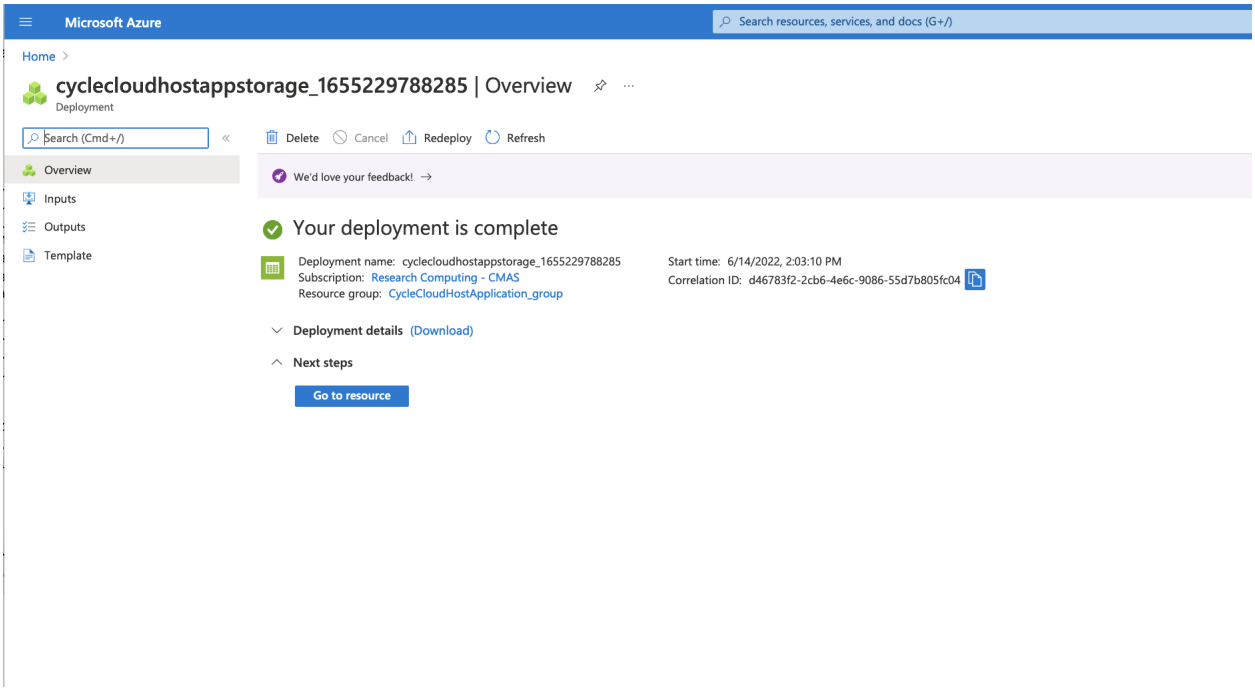

Microsoft Azure
Search resources,
Home > Storage accounts >
Create a storage account
Basics
Advanced
Networking
Data protection
Encryption
Tags
Review + create
Certain options have been disabled by default due to the combination of storage account performance, redundancy, and region.
Security
Configure security settings that impact your storage account.
Require secure transfer for REST API operations
Enable blob public access
Enable storage account key access
Default to Azure Active Directory authorization in the Azure portal
Minimum TLS version
Version 1.2
Data Lake Storage Gen2
The Data Lake Storage Gen2 hierarchical namespace accelerates big data analytics workloads and enables file-level access control lists (ACLs). Learn more
Enable hierarchical namespace
Blob storage
Enable SFTP (preview)
To enable SFTP, 'hierarchical namespace' must be enabled.
Enable network file system v3
To enable NFS v3 'hierarchical namespace' must be enabled. Learn more about NFS v3
Allow cross-tenant replication
Access tier
Hot: Frequently accessed data and day-to-day usage scenarios
Cool: Infrequently accessed data and backup scenarios
Azure Files
Enable large file shares
Review + create
< Previous
Next : Networking >

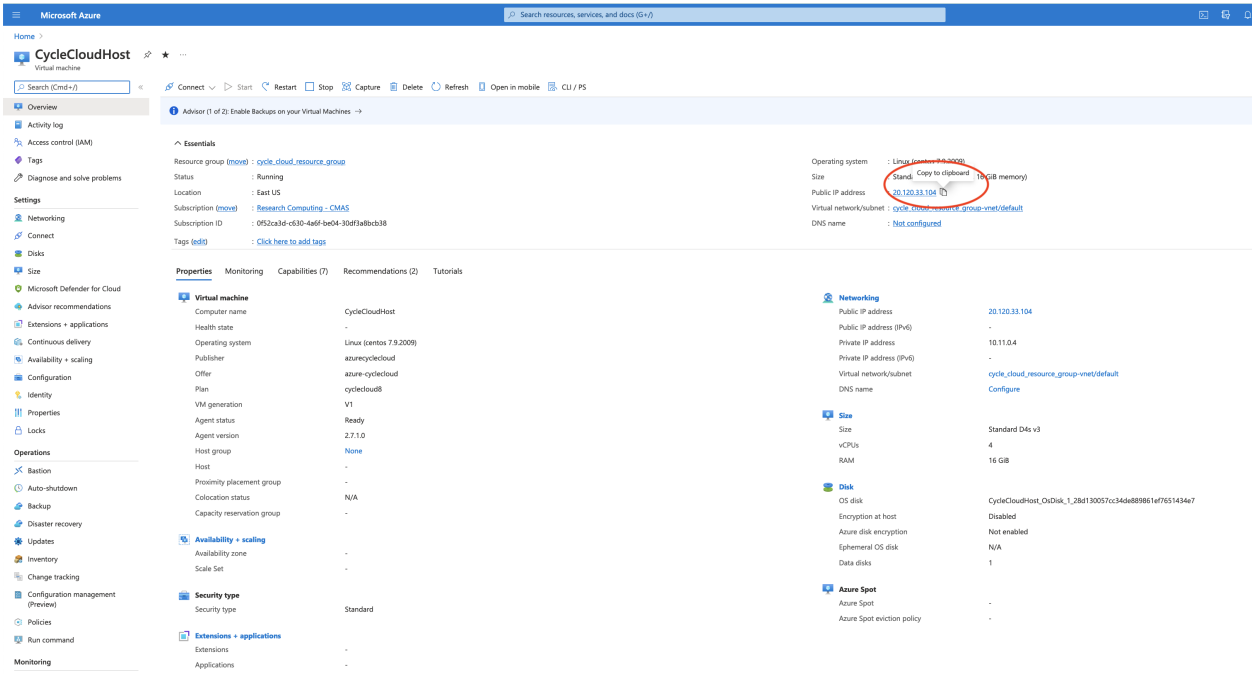
Figure 14. Storage Account Deployment is complete



Click on Home to return to the Azure Portal and then Click on the CycleCloudHostApplication Virtual Machine

Click on Copy next to the Public IP address to copy it.

Figure 15. Azure Cycle Cloud Host Machine IP address



Connect to Cyclecloud Web Interface

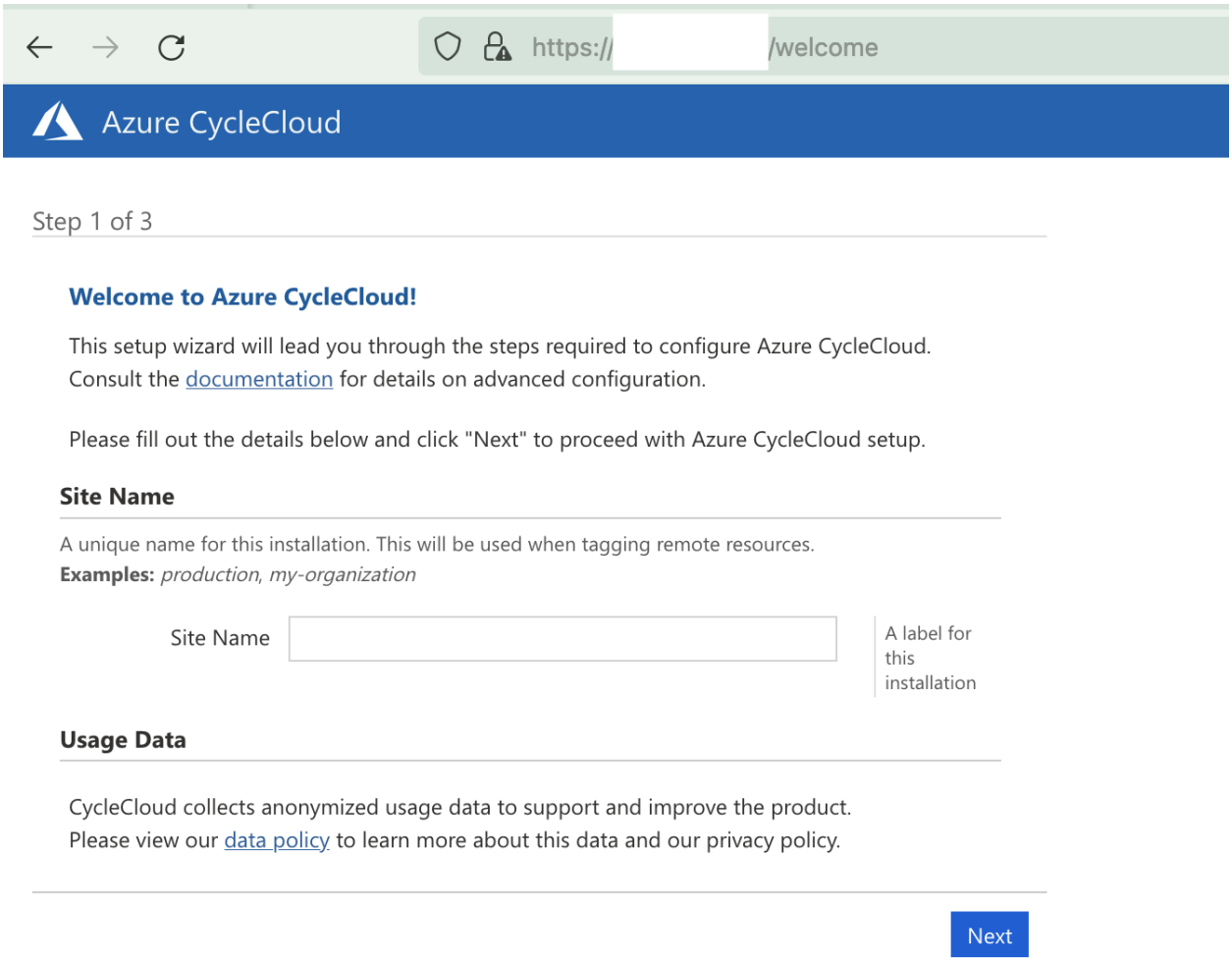
In your web browser, create a new tab, and enter the IP address that you copied from the step above.

`https://-IP-ADDRESS/welcome`

If you get a warning, potential security risk ahead, click on Advanced, then accept risk and continue.

1. Enter a Site Name - a unique name for the CycleCloud. example CycleCloudHostApplicationManager
2. Read and click that you agree to the CycleCloud Software License Agreement
3. Create your CycleCloud Administrator Account. This requires a public rsa key. Instructions for creating this are available here

Figure 16. Web Interface to CycleCloud - connect using the ip address for the Scheduler Node above http://-IP-ADDRESS/welcome



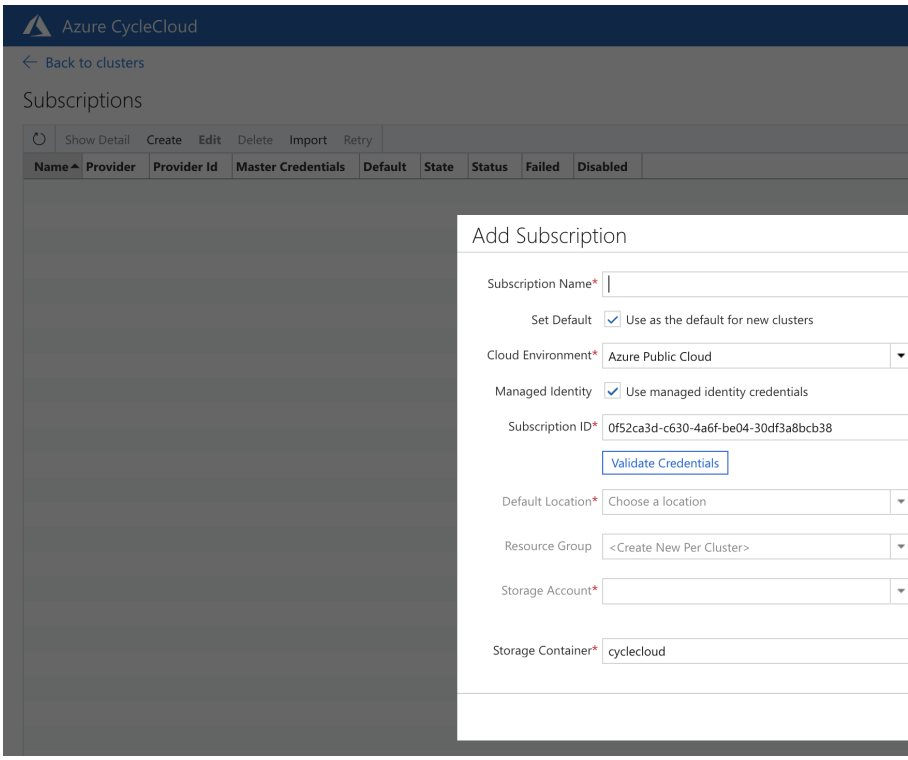


Figure 17. Azure CycleCloud Add Subscription ID

The Subscriptions page will show if the cluster subscription was created. You may need to pull the State window to enlarge it.

When it says created, with nothing under the Failed column, then it was successful, click Back to Clusters.

Figure 18. Check Cluster Creation Status in Subsriptions Table

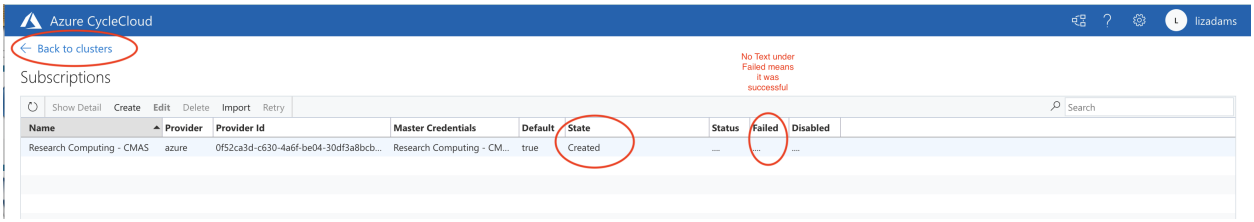


Figure 19. Azure CycleCloud Create a New Cluster - Select SLURM workload Manager

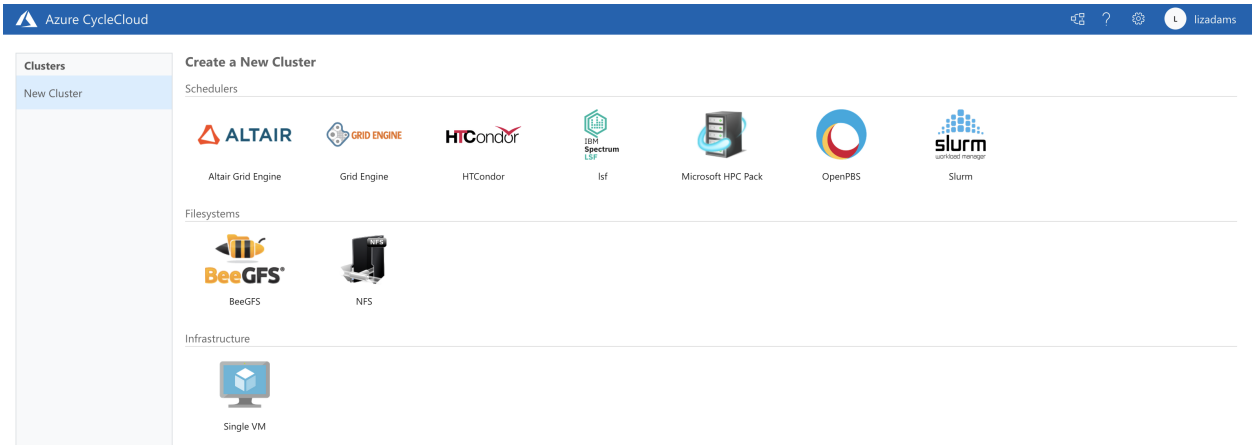


Figure 20. Azure CycleCloud New Slurm Cluster - add a Cluster Name

Example name: CMAQSlurmHC44rsAlmaLinux

Figure 21. Azure CycleCloud HPC Queue Select Machine

In the Min Cores box, input 44 In the Compute Type, select High Performance Compute Select HC44rs, then select Apply

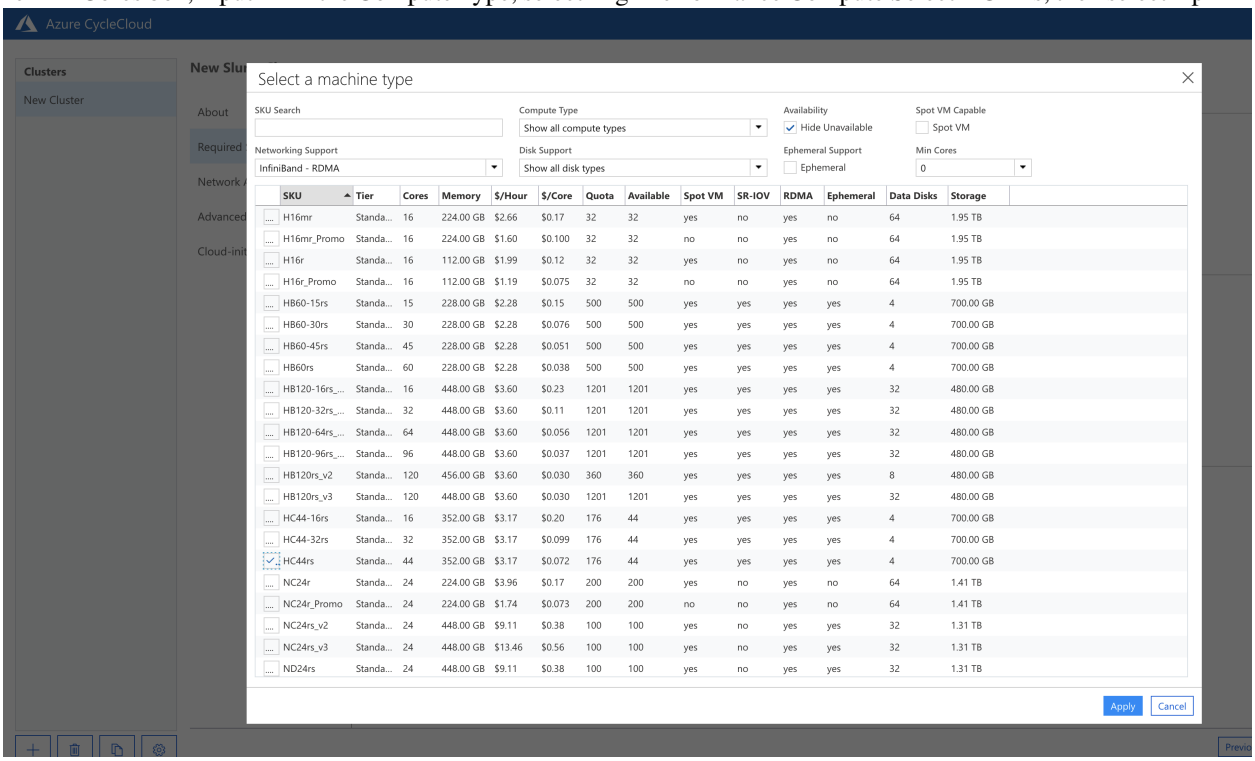


Figure 22. Select Max HPC Cores

Select Auto-Scaling Max HPC Cores to be a multiple of the number of cpus available on the compute node. For HC44rs for a maximum of 5 nodes, it would be 44 x 5 = 220 Max HPC Cores Choose the Networking SubnetID that was created for the CycleCloud.

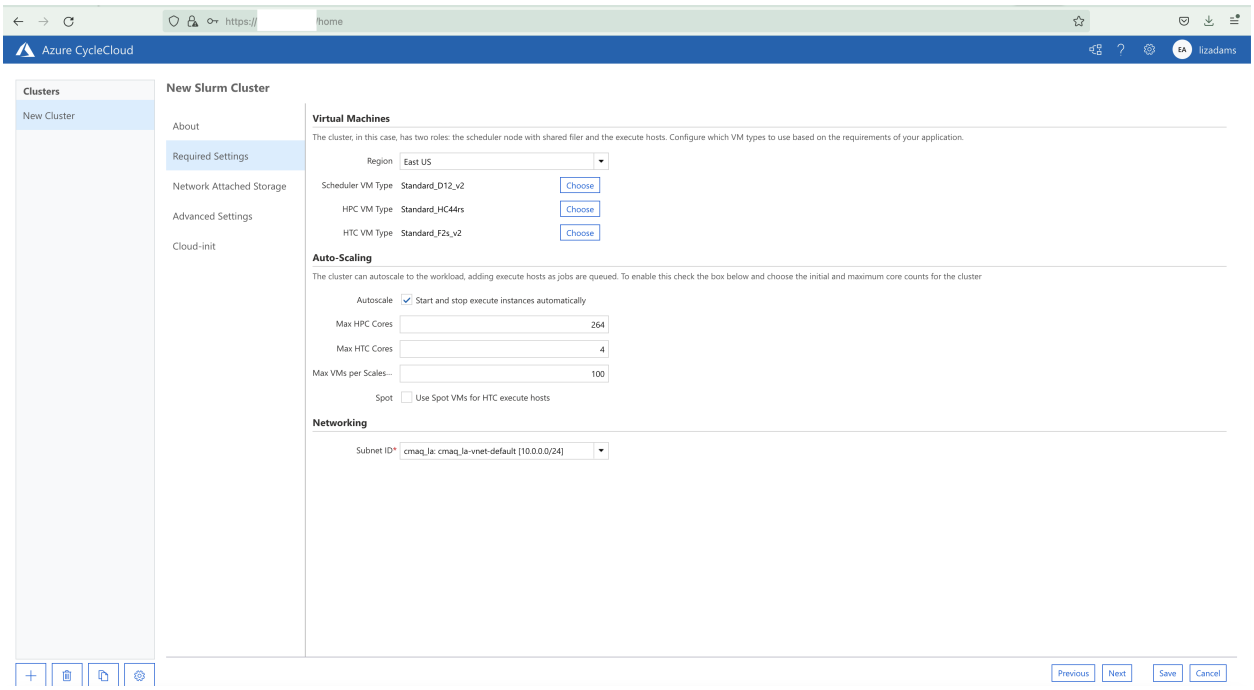


Figure 23. Azure CycleCloud Network Attached Storage

Change the size from 100 GB of network attached storage to 1000 GB of network attached storage for the /shared directory, where CMAQ and the input data will be installed.

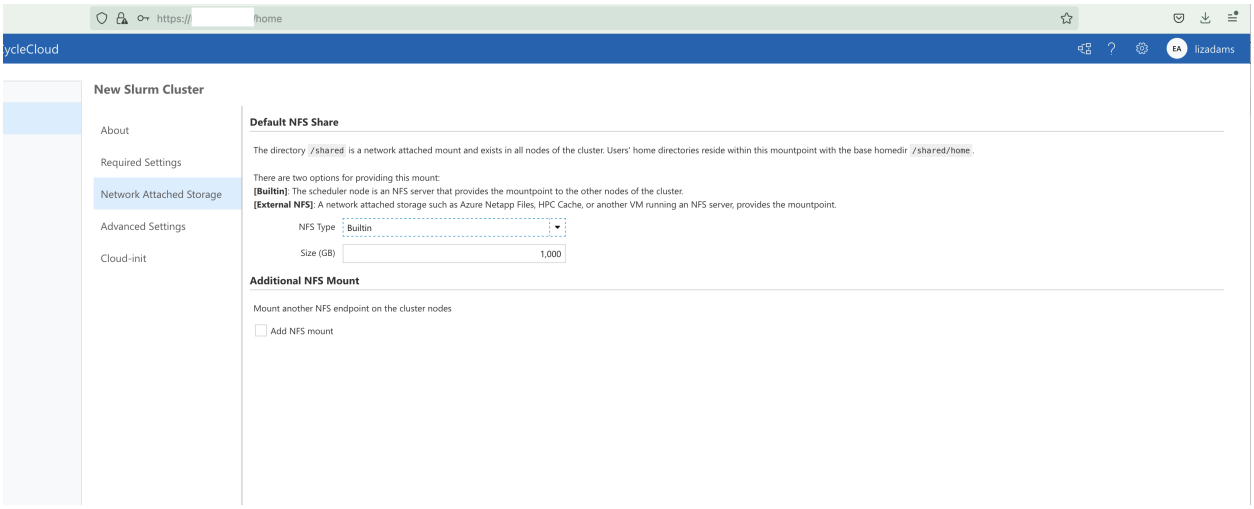


Figure 24. Azure CycleCloud Select OS and Uncheck Name as HostName

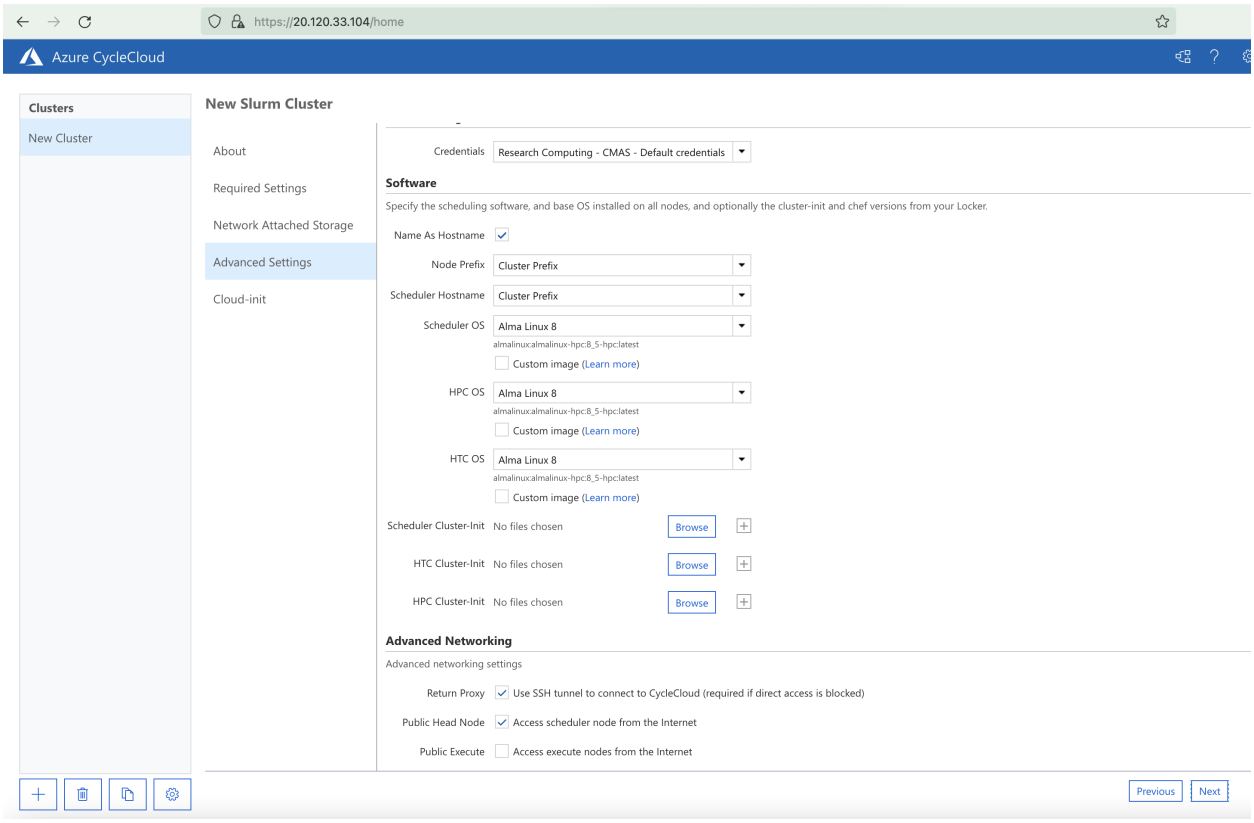


Figure 25. Azure CycleCloud Select Machine Type for HPC Node

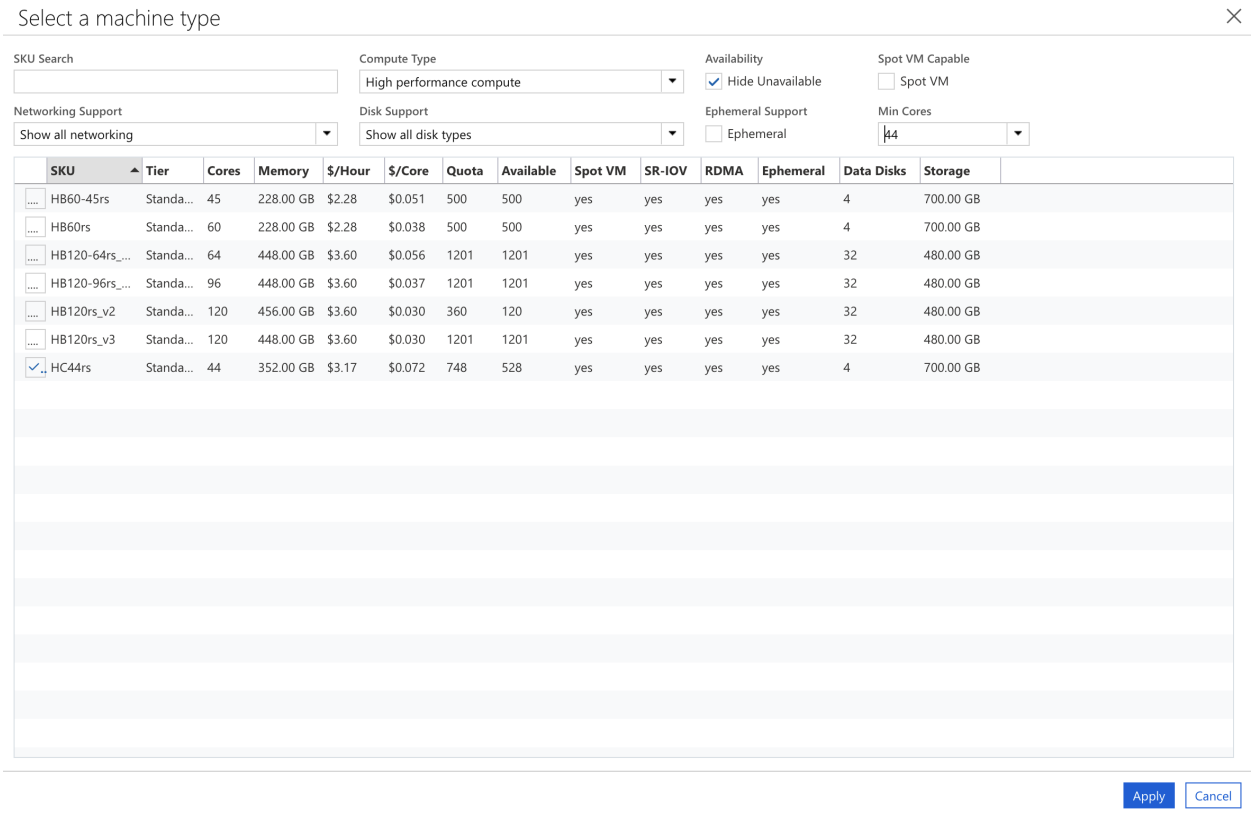
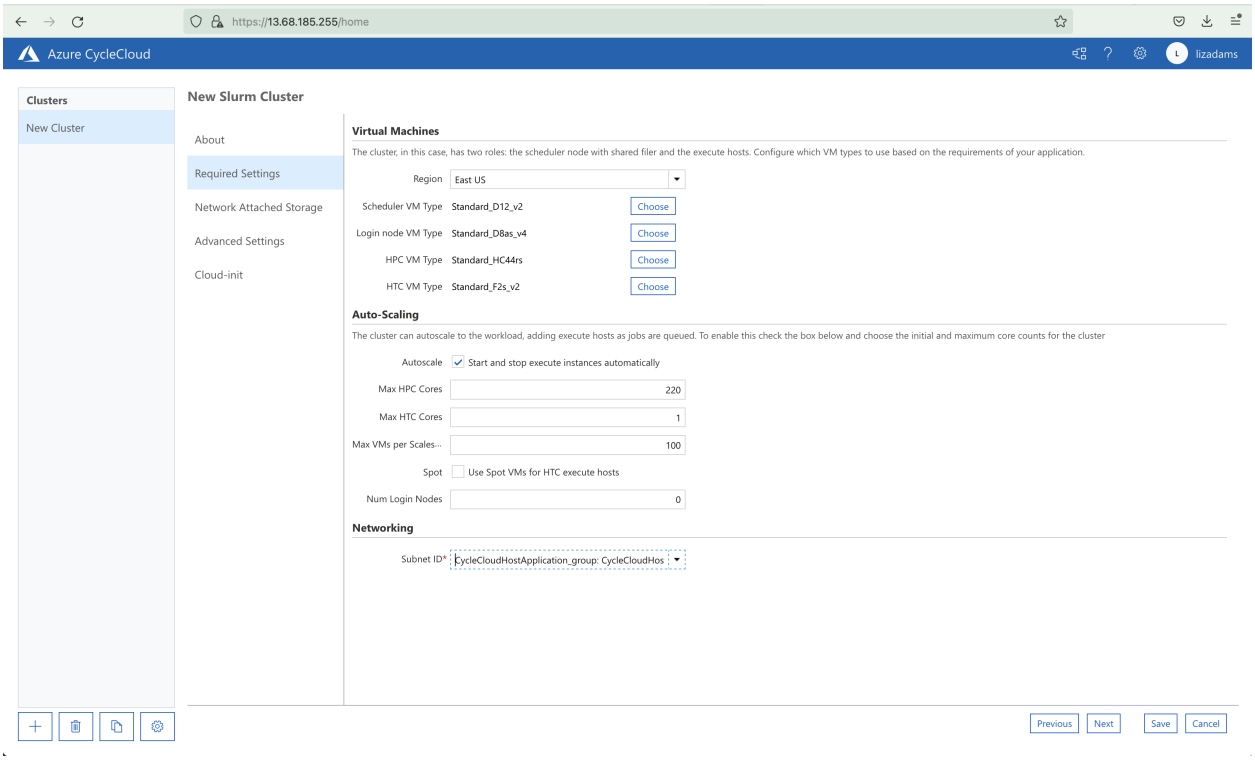


Figure 26. Azure Cycle Cloud Required Settings HPC VM Select HC44rs



Note: the maximum number of CPUs specified for the HPC Compute node can be changed after the cluster has been created. See section 4.1.4 for the command line commands.

Figure 27. Azure Cycle Cloud Subscriptions Registering Service Providers

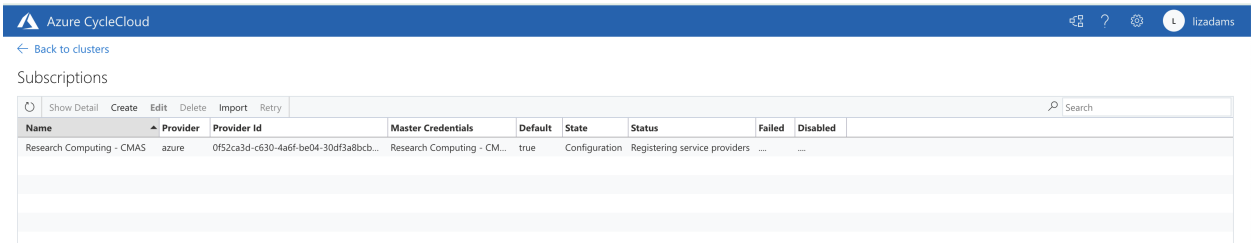


Figure 28. Azure cycle Cloud Subscription Created Successsfully

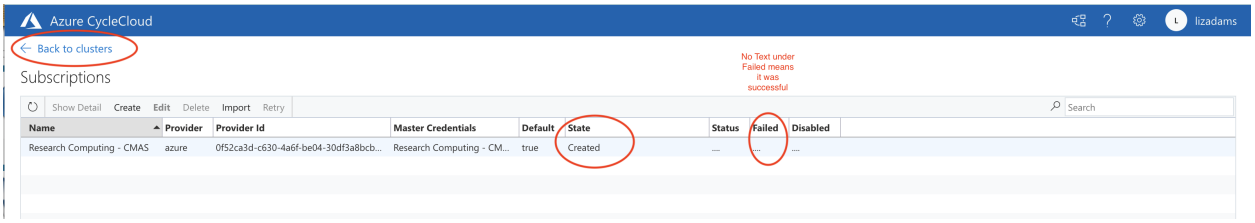


Figure 29. Azure cycle cloud Nodes Tab Shows Status of Scheduler

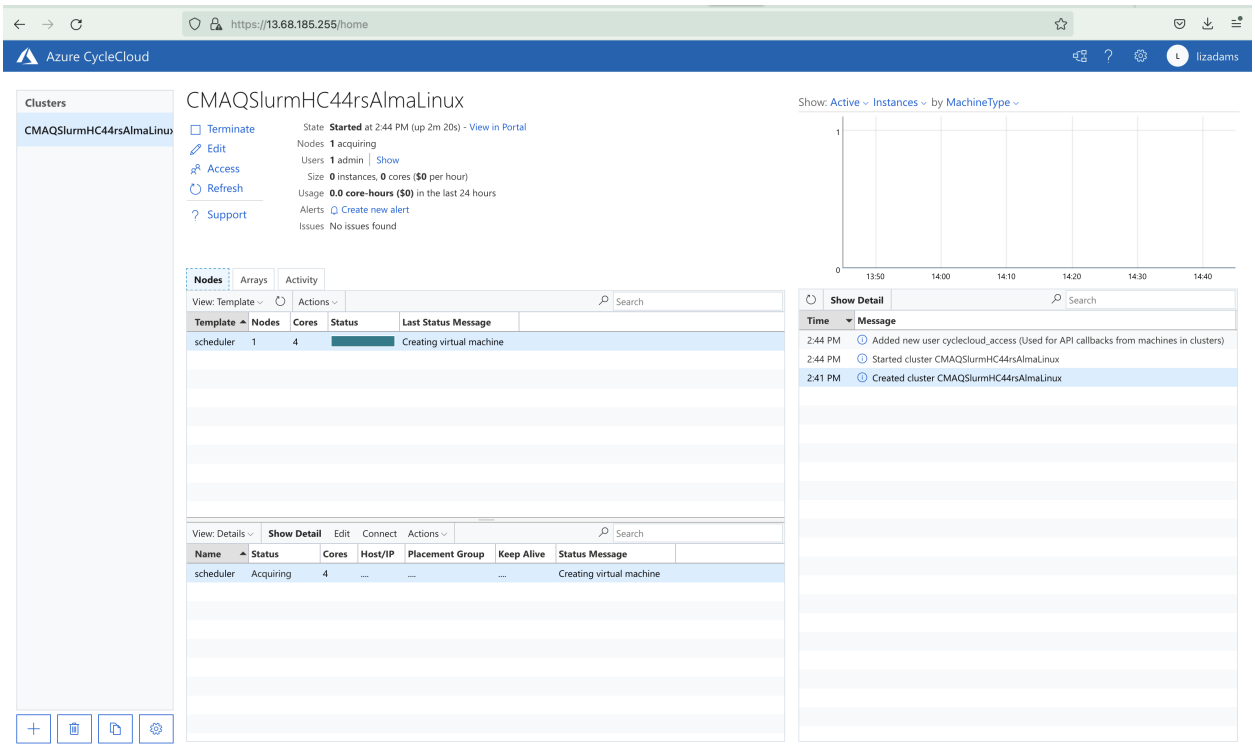
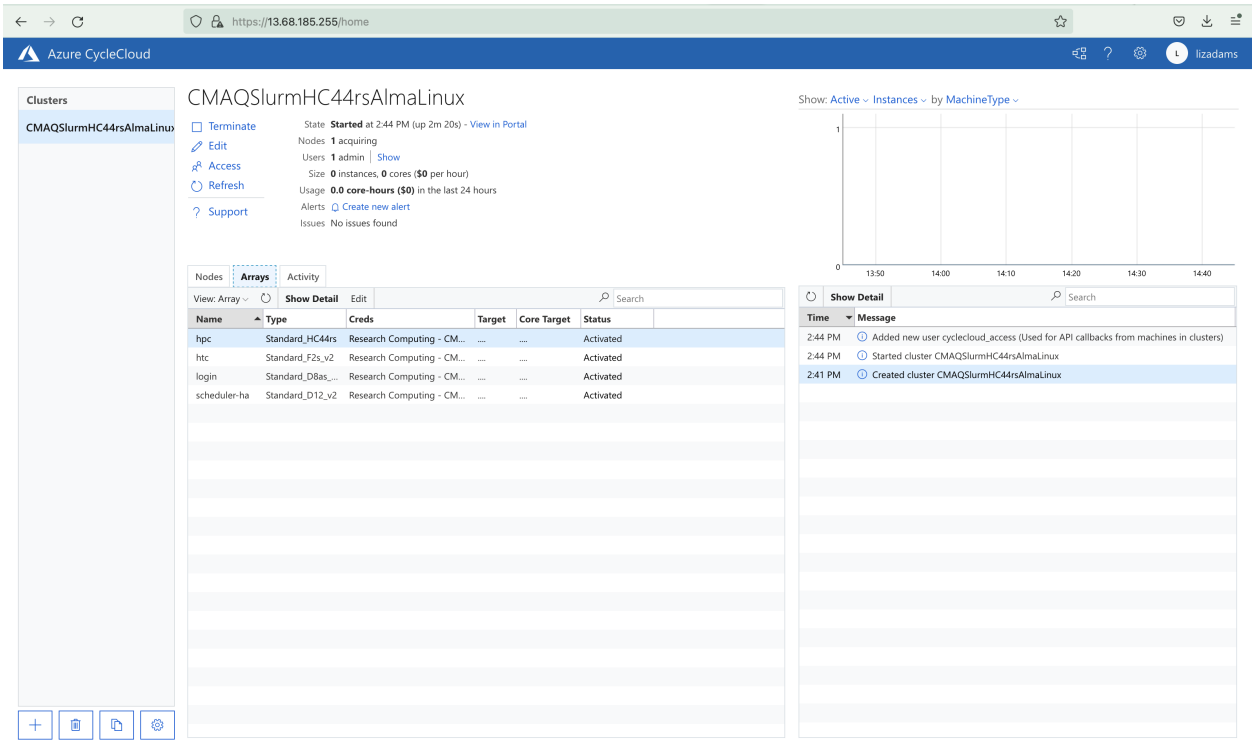


Figure 30. Azure Cycle Cloud Cluster Arrays Tab Shows HPC Queue Machine Type



Azure Cycle Cloud Start Cluster In the Nodes table, it will say scheduler 1 node, 4 cores, Status Message: Staging Resources

Login to Azure Cycle Cloud and verify that the following command works.

Click on the Scheduler node, and obtain the IP address, then login using

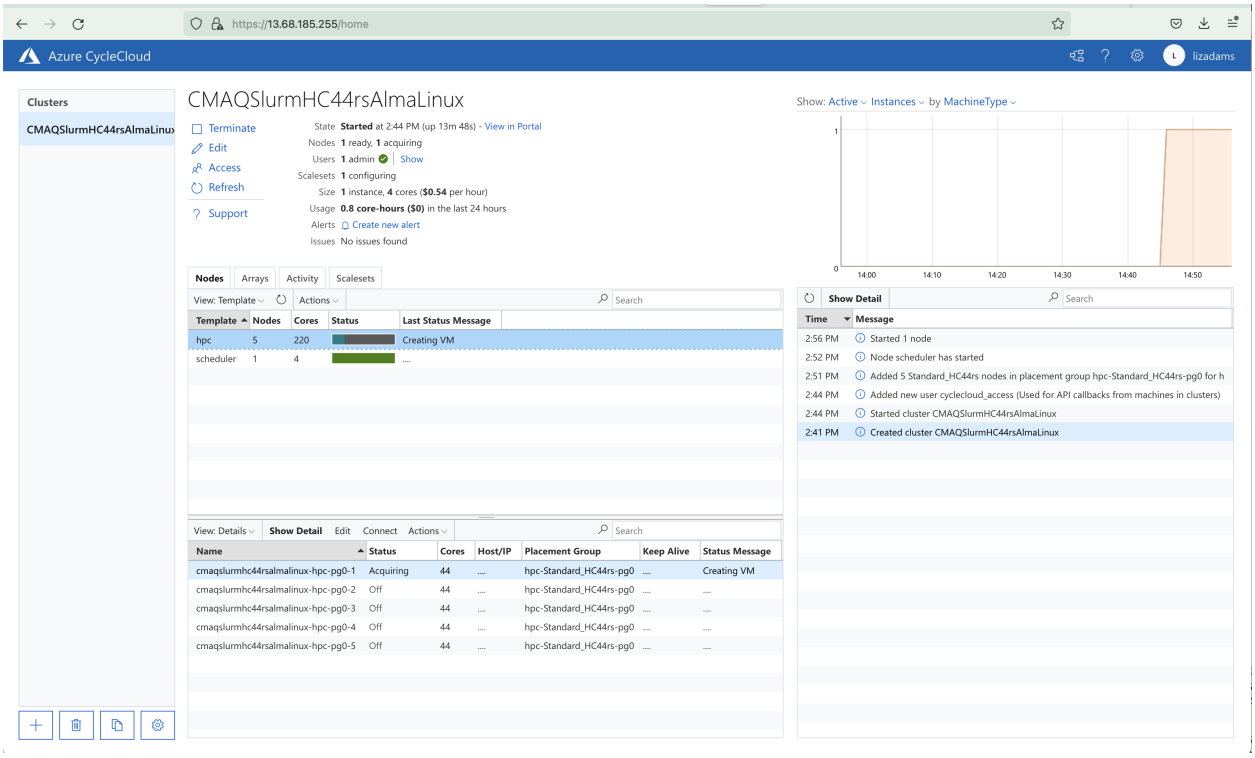
ssh -Y azureuser@IP-ADDRESS

Run a bash script for 1 minute by submitting to the hpc node using srun.

```
srun -t 1:00 -n 2 --pty /bin/bash
```

You should see the hpc acquiring a single node.

Figure 31. Azure CycleCloud Acquiring Compute Node after running srun command.



After the compute node is created and the srun command is completed, the compute node will be shut down automatically, after it has been idle for a period of time.

You can use the slurm commands to monitor the status of the compute nodes.

```
qstat
```

```
Job id                    Name             Username        Time Use S Queue
------------------------- ---------------- --------------- -------- - ---------------
2                         bash             lizadams        00:00:05 R hpc
```

for additional detail:

```
qstat -f
```

Output:

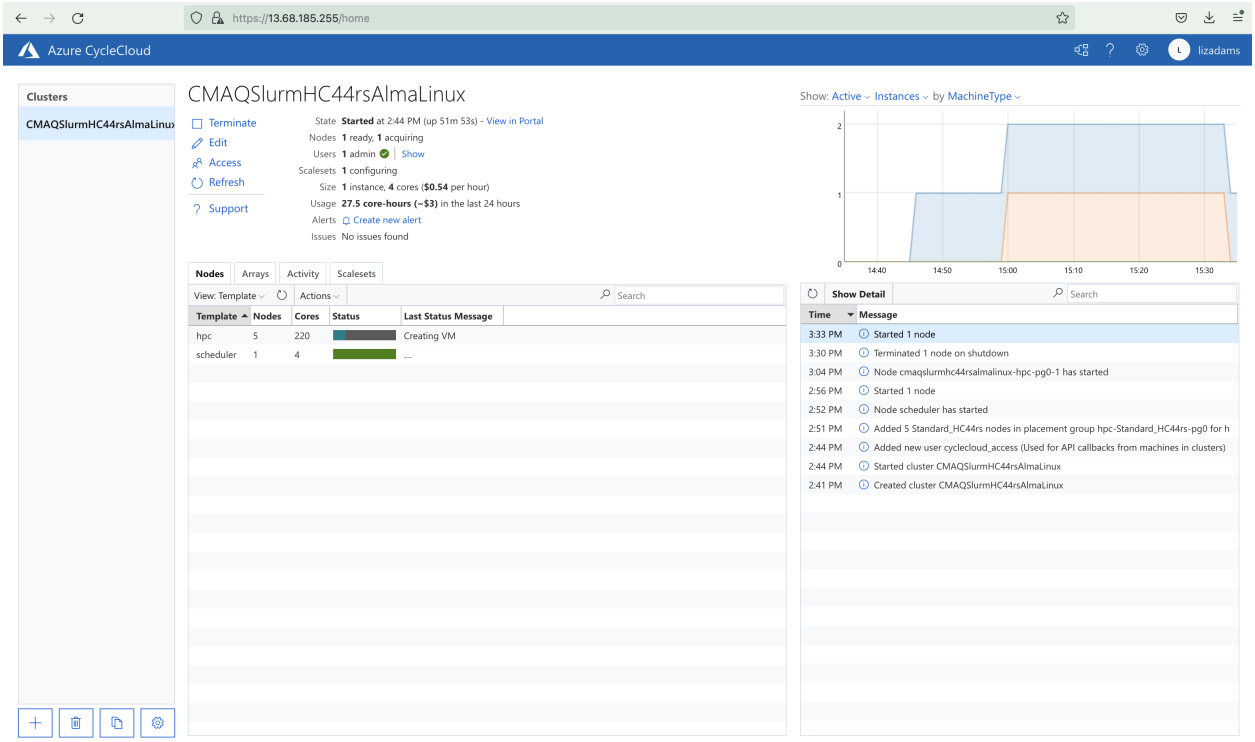
```
Job Id:         2
        Job_Name = bash
        Job_Owner = lizadams@CMAQSlurmHC44rsAlmaLinux-scheduler
        interactive = True
        job_state = R
```

(continues on next page)

(continued from previous page)

```
        queue = hpc
        qtime = Tue Jun 14 18:56:17 2022
        mtime = Tue Jun 14 19:04:13 2022
        ctime = Tue Jun 14 20:34:13 2022
        exec_host = cmaqslurmhc44rsalmalinux-hpc-pg0-1/2
        Priority = 4294901759
        euser = lizadams(20001)
        egroup = lizadams(20001)
        Resource_List.walltime = 01:30:00
        Resource_List.nodect = 1
        Resource_List.ncpus = 2
```

Figure 32. Azure Cycle Cloud Showing usage of Scheduler Node and Compute Nodes for Srun command



Instructions to upgrade the number of processors available to the Cycle Cloud Cluster (only needed if you want to modify the number of nodes in the HPC queue)

Edit the HPC config in the cyclecloud web interface to set the CPUs to 480 Run the following on the scheduler node the changes should get picked up:

```
cd /opt/cycle/slurm
```

```
sudo ./cyclecloud_slurm.sh scale
```

3.4.2 Modify Cyclecloud CMAQ Cluster

If you make changes to the nodes you must run the following commands

```
cd /opt/cycle/slurm
sudo ./cyclecloud_slurm.sh remove_nodes
sudo ./cyclecloud_slurm.sh scale
```

The above commands will remove the hpc nodes from the cluster and then rescale the cluster to use the new nodes that were specified when you edited the cluster.

This is required if you change the identity of the hpc VM. An example: if you change Standard_HB120-64rs_v3, a EPYC virtual machine containing 64 pes, to Standard_HB120rs_v3, an EPYC virtual machine containing 120 pes.

3.4.3 Install CMAQ and pre-requisite libraries on linux

Login to updated cluster

(note, replace the centos.pem with your Key Pair)

```
ssh -v -Y -i ~/[your_azure].pem [scheduler-node-ip-address]
```

Change shell to use .tcsh

```
sudo usermod -s /bin/tcsh azureuser
```

Log out and then log back in to activate the tcsh shell

Optional Step to allow multiple users to run on the CycleCloud Cluster

Add group name to users

```
sudo groupadd cmaq
```

Add the new group for each user

```
sudo usermod -a -G cmaq azureuser
```

Logout and log back in to reset the new group

Set the group to be default group for files created by the user

```
sudo usermod -g cmaq azureuser
```

logout and log back in to have it take effect

Check to see if the group is added to your user ID

```
id
```

Make the /shared/build directory

```
sudo mkdir /shared/build
```

Change ownership to your username

```
sudo chown azureuser /shared/build
```

Make the /shared/cyclecloud-cmaq directory

```
sudo mkdir /shared/cyclecloud-cmaq
```

Change ownership to your username

```
sudo chown azureuser /shared/cyclecloud-cmaq
```

Install git

```
sudo yum install git
```

Install the cluster-cmaq git repo to the /shared directory

```
cd /shared
git clone -b main https://github.com/CMASCenter/cyclecloud-cmaq.git cyclecloud-cmaq
```

```
cd cyclecloud-cmaq
```

Optional - Change the group to cmaq recursively for the /shared directory/build

```
sudo chgrp -R cmaq /shared/build
```

Check what modules are available on the cluster

```
module avail
```

Load the openmpi module

```
module load mpi/openmpi-4.1.0
```

Load the gcc copiler - note, this may have been automatically loaded by the openmpi module

```
module load gcc-9.2.0
```

Verify the gcc compiler version is greater than 8.0

```
gcc --version
```

output:

```
gcc (GCC) 9.2.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Change directories to install and build the libraries and CMAQ

```
cd /shared/cyclecloud-cmaq
```

Build netcdf C and netcdf F libraries - these scripts work for the gcc 8+ compiler

```
./gcc_netcdf_cluster.csh
```

A .cshrc script with LD_LIBRARY_PATH was copied to your home directory, enter the shell again and check environment variables that were set using

```
cat ~/.cshrc`
```

If the .cshrc wasn't created use the following command to create it

```
cp dot.cshrc.cyclecloud  ~/.cshrc
```

Execute the shell to activate it

```
csh
env
```

Verify that you see the following setting

```
echo $LD_LIBRARY_PATH
```

output:

```
/opt/openmpi-4.1.0/lib:/opt/gcc-9.2.0/lib64
```

Build I/O API library

```
./gcc_ioapi_cluster.csh
```

Build CMAQ

note, the primary difference is the location of the openmpi libraries on cyclecloud, /opt/openmpi-4.1.0/lib and include, /opt/openmpi-4.1.0/include

```
./gcc_cmaq_cyclecloud.csh
```

Check to see that the cmaq executable has been built

```
ls /shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts/BLD_CCTM_v533_gcc/*.exe
```

If it fails due to an issue with finding mpi, you will need to edit the gcc_cmaq_cyclecloud.csh script to point to the location of the mpi library and bin directory.

The following path is specified in the config_cmaq_cyclecloud.csh script, and may need to be updated. To find the mpi paths, use the command:

```
which mpirun
```

```
        setenv MPI_INCL_DIR     /opt/openmpi-4.1.0/include              #> MPI Include␣
↪directory path
        setenv MPI_LIB_DIR      /opt/openmpi-4.1.0/lib             #> MPI Lib directory␣
↪path
```

3.4.4 Configuring selected storage and obtaining input data

Install AWS CLI to obtain data from AWS S3 Bucket

see https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

```
cd /shared/build
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

edit .cshrc file to add /usr/local/bin to path

```
vi ~/.cshrc
```

add /usr/local/bin to the set path line

Run csh at the command line

Verify you can run the aws command

```
 aws --help
```

If not, you may need to logout and back in.

Set up your credentials for using s3 copy (you can skip this if you do not have credentials)

```
aws configure
```

Azure Cyclecloud install input on the /shared/data directory

```
sudo mkdir /shared/data
```

Change ownership

```
sudo chown azureuser /shared/data
ls /shared/data
df -h
```

Output:

```
/dev/mapper/vg_cyclecloud_builtinshared-lv0 1000G 66G 935G 7% /shared
```

Use the S3 script to copy the CONUS input data from the CMAS s3 bucket

Modify the script if you want to change where the data is saved to. Script currently uses /shared/data

`/shared/cyclecloud-cmaq/s3_scripts/s3_copy_nosign_conus_cmas_opendata_to_shared.csh`

check that the resulting directory structure matches the run script

Note, this input data requires 44 GB of disk space (if you use the yaml file to import the data to the lustre file system rather than copying the data you save this space)

`cd /shared/data/CMAQ_Modeling_Platform_2016/CONUS/12US2`

`du -sh`

output:

```
44G     .
```

CMAQ Cycle Cloud is configured to have 1 Terrabytes of space on the /shared filesystem, to allow multiple output runs to be stored.

3.4.5 Copy the run scripts from the CycleCloud repo

Note, the run scripts are tailored to the Compute Node. This assumes the cluster was built with HC44rs compute nodes.

Change directories to where the run scripts are available from the git repo.

`cd /shared/cyclecloud-cmaq/run_scripts/HC44rs`

Copy the run scripts to the run directory

`cp * /shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts/`

3.4.6 Run the CONUS Domain on 180 pes

`cd /shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts/`

`sbatch run_cctm_2016_12US2.180pe.csh`

Note, it will take about 3-5 minutes for the compute notes to start up This is reflected in the Status (ST) of PD (pending), with the NODELIST reason being that it is configuring the partitions for the cluster

3.4.7 Check the status in the queue

`squeue`

output:

```
[lizadams@CMAQSlurmHC44rsAlmaLinux-scheduler scripts]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 6       hpc     CMAQ lizadams CF       5:03      5␣
↪cmaqslurmhc44rsalmalinux-hpc-pg0-[1-5]
```

After 5 minutes the status will change once the compute nodes have been created and the job is running

`squeue`

output:

```
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 6       hpc     CMAQ lizadams  R       0:37      5␣
↪cmaqslurmhc44rsalmalinux-hpc-pg0-[1-5]
```

The 180 pe job should take 60 minutes to run (30 minutes per day)

Note, if the job does not get scheduled, examine the slurm logs

`sudo vi /var/log/slurmctld/slurmctld.log`

`sudo vi //var/log/slurmctld/resume.log`

3.4.8 check the timings while the job is still running using the following command

`grep 'Processing completed' CTM_LOG_001*`

output:

```
   Processing completed...    4.6 seconds
            Processing completed...    4.8 seconds
            Processing completed...    4.8 seconds
            Processing completed...    5.2 seconds
            Processing completed...    4.4 seconds
            Processing completed...    5.0 seconds
            Processing completed...    4.6 seconds
            Processing completed...    4.7 seconds
            Processing completed...    4.7 seconds
            Processing completed...    5.1 seconds
```

3.4.9 When the job has completed, use tail to view the timing from the log file.

`tail /shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts/run_cctmv5.3.3_Bench_2016_12US2.2x90.10x18pe.2day.log`

output:

```
================================
  ***** CMAQ TIMING REPORT *****
================================
Start Day: 2015-12-22
End Day:   2015-12-23
Number of Simulation Days: 2
Domain Name:               12US2
Number of Grid Cells:      3409560  (ROW x COL x LAY)
Number of Layers:          35
Number of Processes:       180
   All times are in seconds.

Num  Day        Wall Time
01   2015-12-22   2097.37
02   2015-12-23   1809.84
     Total Time = 3907.21
```

(continues on next page)

(continued from previous page)

```
      Avg. Time = 1953.60
```

3.4.10 Check whether the scheduler thinks there are cpus or vcpus

`sinfo -lN`

output:

```
Thu Feb 17 14:53:19 2022
NODELIST             NODES PARTITION       STATE CPUS    S:C:T MEMORY TMP_DISK WEIGHT␣
↪AVAIL_FE REASON
cmaq-hbv3-hpc-pg0-1      1      hpc*       idle% 120   120:1:1 435814        0      1   ␣
↪cloud none
cmaq-hbv3-hpc-pg0-2      1      hpc*       idle% 120   120:1:1 435814        0      1   ␣
↪cloud none
cmaq-hbv3-hpc-pg0-3      1      hpc*       idle% 120   120:1:1 435814        0      1   ␣
↪cloud none
cmaq-hbv3-htc-1          1       htc       idle~ 1       1:1:2   3072        0      1   ␣
↪cloud none
```

#Post-process and QA

Post-processing CMAQ Run, Install R and packages Instructions to install R and packages for QA of CMAQ difference in output between two runs.

3.5 Scripts to run combine and post processing

3.5.1 Build the POST processing routines

Copy the buildit script from the repo, as it was corrected to use CMAQv533 rather than CMAQv532

```
cd /shared/build/openmpi_gcc/CMAQ_v533/POST/combine/scripts
cp /shared/cyclecloud-cmaq/run_scripts/bldit_combine.csh .
```

Run the bldit script for combine.

```
cd /shared/build/openmpi_gcc/CMAQ_v533/POST/combine/scripts
./bldit_combine.csh gcc |& tee ./bldit_combine.gcc.log
```

Copy the bldit script from the repo, as it was corrected to use CMAQv533 rather than CMAQv532.

```
cd /shared/build/openmpi_gcc/CMAQ_v533/POST/calc_tmetric/scripts
cp /shared/cyclecloud-cmaq/run_scripts/bldit_calc_tmetric.csh .
```

Run the bldit script for calc_tmetric

```
cd /shared/build/openmpi_gcc/CMAQ_v533/POST/calc_tmetric/scripts
./bldit_calc_tmetric.csh gcc |& tee ./bldit_calc_tmetric.gcc.log
```

Copy the bldit script from the repo.

```
cd /shared/build/openmpi_gcc/CMAQ_v533/POST/hr2day/scripts
cp /shared/cyclecloud-cmaq/run_scripts/bldit_hr2day.csh .
```

Run the bldit script for hr2day

```
cd /shared/build/openmpi_gcc/CMAQ_v533/POST/hr2day/scripts
./bldit_hr2day.csh gcc |& tee ./bldit_hr2day.gcc.log
```

Copy the bldit script from the repo.

```
cd /shared/build/openmpi_gcc/CMAQ_v533/POST/bldoverlay/scripts
cp /shared/cyclecloud-cmaq/run_scripts/bldit_bldoverlay.csh .
```

Run the bldit script for bldoverlay.

```
cd /shared/build/openmpi_gcc/CMAQ_v533/POST/bldoverlay/scripts
./bldit_bldoverlay.csh gcc |& tee ./bldit_bldoverlay.gcc.log
```

3.6 Scripts to post-process CMAQ output

3.6.1 Note, the post-processing analysis should be done on the head node

Ideally the post-processing would be done on the HTC queue, but the R packages were installed to the head node system library and were not acccessible to the compute nodes. Installing the R software and packages to the /shared volume will be investigated in future work.

Verify that the compute nodes are no longer running if you have completed all of the benchmark runs

`squeue`

You should see that no jobs are running.

Show compute nodes

`scontrol show nodes`

3.6.2 Edit, Build and Run the POST processing routines

You need to run the post processing scripts for every benchmark case.

`cp /shared/cyclecloud-cmaq/run_scripts/run_combine_conus.csh .`

Examine the run script

`cat run_combine_conus.csh`

The post processing scripts are set up for a specific case, example:

setenv APPL 2016_CONUS_10x18pe

note, you will need to change the sed command to a different configuration if you ran another case, example:

setenv APPL 2016_CONUS_12x9pe

If you used the CMAQ Benchmark Option 1, with the pre-loaded software, then these scripts have already been modified.

Run the following scripts

```
cd /shared/build/openmpi_gcc/CMAQ_v533/POST/combine/scripts
sbatch run_combine_conus.csh
```

```
cd /shared/build/openmpi_gcc/CMAQ_v533/POST/calc_tmetric/scripts
sbatch run_calc_tmetric_conus.csh
```

```
cd /shared/build/openmpi_gcc/CMAQ_v533/POST/hr2day/scripts
sbatch run_hr2day_conus.csh
```

```
cd /shared/build/openmpi_gcc/CMAQ_v533/POST/bldoverlay/scripts
sbatch run_bldoverlay_conus.csh
```

If you used the CMAQ Bechmark Option 2 to install CMAQ yourself, you will need to save and modify the scripts using the instructions below.

```
setenv DIR /shared/build/openmpi_gcc/CMAQ_v533/

cd $DIR/POST/combine/scripts
sed -i 's/v532/v533/g' bldit_combine.csh
cp run_combine.csh run_combine_conus.csh
sed -i 's/v532/v533/g' run_combine_conus.csh
sed -i 's/Bench_2016_12SE1/2016_CONUS_10x18pe/g' run_combine_conus.csh
sed -i 's/intel/gcc/g' run_combine_conus.csh
./bldit_combine.csh gcc |& tee ./bldit_combine.gcc.log
sed -i 's/2016-07-01/2015-12-22/g' run_combine_conus.csh
sed -i 's/2016-07-14/2015-12-23/g' run_combine_conus.csh
setenv CMAQ_DATA /shared/data
./run_combine_conus.csh

cd $DIR/POST/calc_tmetric/scripts
./bldit_calc_tmetric.csh gcc |& tee ./bldit_calc_tmetric.gcc.log
cp run_calc_tmetric.csh run_calc_tmetric_conus.csh
sed -i 's/v532/v533/g' run_calc_tmetric_conus.csh
sed -i 's/Bench_2016_12SE1/2016_CONUS_10x18pe/g' run_calc_tmetric_conus.csh
sed -i 's/intel/gcc/g' run_calc_tmetric_conus.csh
sed -i 's/201607/201512/g' run_calc_tmetric_conus.csh
setenv CMAQ_DATA /shared/data
./run_calc_tmetric_conus.csh

cd $DIR/POST/hr2day/scripts
sed -i 's/v532/v533/g' bldit_hr2day.csh
./bldit_hr2day.csh gcc |& tee ./bldit_hr2day.gcc.log
cp run_hr2day.csh run_hr2day_conus.csh
sed -i 's/v532/v533/g' run_hr2day_conus.csh
sed -i 's/Bench_2016_12SE1/2016_CONUS_10x18pe/g' run_hr2day_conus.csh
sed -i 's/intel/gcc/g' run_hr2day_conus.csh
sed -i 's/2016182/2015356/g' run_hr2day_conus.csh
sed -i 's/2016195/2015357/g' run_hr2day_conus.csh
sed -i 's/201607/201512/g' run_hr2day_conus.csh
setenv CMAQ_DATA /shared/data
./run_hr2day_conus.csh
```

(continues on next page)

(continued from previous page)

```
cd $DIR/POST/bldoverlay/scripts
sed -i 's/v532/v533/g' bldit_bldoverlay.csh
./bldit_bldoverlay.csh gcc |& tee ./bldit_bldoverlay.gcc.log
cp run_bldoverlay.csh run_bldoverlay_conus.csh
sed -i 's/v532/v533/g' run_bldoverlay_conus.csh
sed -i 's/Bench_2016_12SE1/2016_CONUS_10x18pe/g' run_bldoverlay_conus.csh
sed -i 's/intel/gcc/g' run_bldoverlay_conus.csh
sed -i 's/2016-07-01/2015-12-22/g' run_bldoverlay_conus.csh
sed -i 's/2016-07-02/2015-12-23/g' run_bldoverlay_conus.csh
setenv CMAQ_DATA /shared/data
./run_bldoverlay_conus.csh
```

3.7 Install R, Rscript and Packages

How to install R on Centos7

May need to install on head node into a local mylibs directory, and then copy to the compute nodes, in order to run post processing R scripts on HTC node using slurm..

Using R on HPC Clusters

Use the following commands, and also install packages - note, see website above for full details:

Install R

```
sudo yum install epel-release
sudo yum config-manager --set-enabled powertools
sudo yum install R
R --version
```

Install packages as root - to make them available to all users

```
sudo -i R
install.packages("stringr")
install.packages("patchwork")
```

Had an issue installing ncdf4 see: <ahref="https://cirrus.ucsd.edu/~pierce/ncdf/">install ncdf4

ncdf4 REQUIRES the netcdf library be version 4 or above, AND installed with HDF-5 support (i.e., the netcdf library must be compiled with the –enable-netcdf-4 flag). If you do not want to install the full version of netcdf-4 with HDF-5 support, then please install the old, deprecated ncdf package instead.

ERROR: configuration failed for package ‘ncdf4’

- removing ‘/usr/lib64/R/library/ncdf4’

building netcdf with HDF5 support requires curl.

```
sudo yum install curl
sudo yum install libcurl-devel
```

Load the gcc and openmpi module before building the hdf5 enabled netcdf libraries.

```
module avail
module load mpi/openmpi-4.1.1
module load gcc-9.2.1
```

```
cd /shared/cyclecloud-cmaq
./gcc_install_hdf5.cyclecloud.csh
```

Need to specify the location of nc-config in your .cshrc

set path = ($path /shared/build/install/bin /shared/build/ioapi-3.2/Linux2_x86_64gfort /shared/build-hdf5/install/bin)

Run command to install ncdf4 package

```
cd /shared/cyclecloud-cmaq/qa_scripts/R_packages
```

```
sudo R CMD INSTALL ncdf4_1.13.tar.gz --configure-args="--with-nc-config=/shared/
build-hdf5/install/bin/nc-config"
```

or to install to local directory

```
R CMD INSTALL ncdf4_1.13.tar.gz --configure-args="--with-nc-config=/shared/build-hdf5/
install/bin/nc-config" -l "/shared/build/R_libs"
```

Install additional packages as root so that all users will have access.

```
sudo -i R
install.packages("fields")
install.packages("mapdata")
```

M3 package requires gdal

```
sudo yum install gdal
sudo yum install gdal-devel proj-devel
```

```
sudo -i R
install.packages("rgdal")
install.packages("ggplot2")
```

```
sudo R CMD INSTALL M3_0.3.tar.gz
```

3.8 QA CMAQ

Quality Assurance: Comparing the output of two CMAQ runs.

3.8.1 Quality Assurance Checks for Successful CMAQ Run on CycleCloud

run m3diff to compare the output data for two runs that have different values for NPCOL

```
cd /shared/data/output
ls */*ACONC*
```

```
setenv AFILE output_CCTM_v533_gcc_2016_CONUS_10x18pe_remove_native_sleep/CCTM_ACONC_v533_
↪gcc_2016_CONUS_10x18pe_remove_native_sleep_20151222.nc
setenv BFILE output_CCTM_v533_gcc_2016_CONUS_9x10pe_remove_native_sleep/CCTM_ACONC_v533_
↪gcc_2016_CONUS_9x10pe_remove_native_sleep_20151222.nc
```

```
m3diff
```

hit return several times to accept the default options

```
grep A:B REPORT
```

Should see all zeros. There are some non-zero values. It appears to have all zeros if the domain decomposition is the same NPCOL, here, NPCOL differs (10 vs 16) Did a test to determine if removing the compiler option -march=native would result in zero differences if NPCOL differs. This seems to work on CycleCloud, but did not work on Parallel Cluster.

Verify that you have loaded the gcc and openmpi modules.

`module avail`

`module load gcc-9.2.0`

`module load mpi/openmpi-4.1.0`

Verfiy the compiler version:

`gcc --version`

Output

```
gcc (GCC) 9.2.0
```

Comparison of the Makefiles on Cyclecloud:

`cd /shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts`

`diff BLD_CCTM_v533_gcc/Makefile BLD_CCTM_v533_gcc_remove_native/Makefile`

Output:

```
36c36
<   FSTD = -O3 -funroll-loops -finit-character=32 -Wtabs -Wsurprising -march=native -
↪ftree-vectorize  -ftree-loop-if-convert -finline-limit=512
---
>   FSTD = -O3 -funroll-loops -finit-character=32 -Wtabs -Wsurprising -ftree-vectorize  -
↪ftree-loop-if-convert -finline-limit=512
```

```
NPCOL  =  10; @ NPROW = 18
NPCOL  =  9; @ NPROW = 10
```

```
grep A:B REPORT
```

output

```
 A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
 A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
 A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
```

(continues on next page)

(continued from previous page)

```
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
A:B  0.00000E+00@(  1,  0, 0)  0.00000E+00@(  1,  0, 0)  0.00000E+00  0.00000E+00
```

Use m3diff to compare two runs that have different NPCOL but were compiled with -march=native

```
setenv AFILE output_CCTM_v533_gcc_2016_CONUS_10x18pe/CCTM_ACONC_v533_gcc_2016_CONUS_
↪10x18pe_20151222.nc
setenv BFILE output_CCTM_v533_gcc_2016_CONUS_12x20pe/CCTM_ACONC_v533_gcc_2016_CONUS_
↪12x20pe_20151222.nc
m3diff
```

```
grep A:B REPORT
```

```
NPCOL   =  10; @ NPROW = 18
NPCOL   =  12; @ NPROW = 20
```

Resulted in differences in the output

```
A:B  1.39698E-09@(280, 83, 1) -1.16415E-10@( 58,208, 1)  1.79255E-14  4.74000E-12
A:B  2.79397E-09@(320,150, 1) -3.72529E-09@(300,145, 1) -2.61413E-16  2.41309E-11
A:B  3.72529E-09@(292,140, 1) -5.58794E-09@(273,157, 1) -1.07441E-13  9.05464E-11
A:B  1.27475E-08@(246, 60, 1) -1.44355E-08@(353,166, 1)  1.10951E-13  2.19638E-10
A:B  5.12227E-08@(322,183, 1) -9.73232E-08@(384,201, 1) -4.47347E-12  9.37281E-10
A:B  2.44007E-07@(383,201, 1) -1.62516E-07@(357,171, 1) -1.39055E-11  2.33763E-09
A:B  2.99886E-07@(291,150, 1) -2.14204E-07@(321,183, 1) -1.30589E-11  3.38753E-09
A:B  4.89876E-07@(316,190, 1) -2.59839E-07@(368,178, 1)  1.53742E-11  5.59802E-09
A:B  5.34113E-07@(281,157, 1) -3.59956E-07@(293,156, 1)  2.77145E-11  7.09438E-09
A:B  5.22472E-07@(317,190, 1) -4.34928E-07@(293,157, 1)  1.76922E-11  8.39259E-09
A:B  4.05125E-06@(295,160, 1) -5.04777E-07@(317,184, 1)  4.79654E-11  1.63639E-08
A:B  7.26432E-07@(308,159, 1) -1.67079E-06@(295,160, 1)  1.09544E-11  1.36729E-08
A:B  7.61822E-07@(262,167, 1) -2.97464E-06@(295,161, 1) -5.31256E-11  1.96557E-08
```

(continues on next page)

(continued from previous page)

```
A:B  1.52830E-06@(252,170, 1) -2.21282E-06@(339,201, 1) -1.21567E-12  2.49501E-08
A:B  2.96161E-06@(296,160, 1) -2.02283E-06@(300,169, 1)  1.20804E-10  3.65410E-08
A:B  2.82843E-06@(132, 98, 1) -1.64099E-06@(134, 98, 1)  2.63695E-10  3.76774E-08
A:B  1.87941E-06@(310,163, 1) -1.13249E-06@(279,167, 1) -2.02132E-10  3.36560E-08
A:B  2.50991E-06@(349,206, 1) -1.35228E-06@(297,165, 1) -1.04908E-10  3.62385E-08
A:B  1.50874E-06@(348,204, 1) -3.92273E-06@(298,165, 1) -1.10150E-10  3.89420E-08
A:B  3.27453E-06@(352,208, 1) -6.02473E-06@(259,178, 1) -4.80990E-10  4.45810E-08
A:B  2.69525E-06@(259,182, 1) -4.68642E-06@(259,179, 1) -3.80682E-10  4.53800E-08
A:B  2.86289E-06@(259,182, 1) -4.98630E-06@(259,180, 1) -7.44821E-11  4.62413E-08
A:B  2.29664E-06@(354,208, 1) -3.09758E-06@(259,181, 1) -2.24090E-10  4.51584E-08
A:B  1.93343E-06@(354,208, 1) -1.84402E-06@(309,158, 1) -2.71079E-10  4.94183E-08
```

Run the following R script to create box plots and spatial plots showing difference between two CMAQ runs.

Note: requires that the R scripts and packages. See earlier instructions.

edit the R script to specify the sim1.dir, sim1.file and sim2.dir, sim2.file to correspond to the Benchmark cases that have been run.

Then run the R scripts!

```
cd /shared/cyclecloud-cmaq/qa_scripts
Rscript compare_EQUATES_benchmark_output_CMAS_cyclecloud.r
```

View the Operating System

`cat /etc/os-release`

Output:

```
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"

CENTOS_MANTISBT_PROJECT="CentOS-7"
CENTOS_MANTISBT_PROJECT_VERSION="7"
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="7"
```

To view the script, install imagemagick

```
sudo yum groupinstall "Development Tools" -y
sudo yum install ImageMagick
sudo yum install ImageMagick-devel
sudo yum install xauth
```

Other ideas for fixing display back to local host. how-to-fix-x11-forwarding-request-failed-on-channel-0

Make sure that you have Xquartz running on your local machine, and that you have given permission to display back from the cyclecloud server.

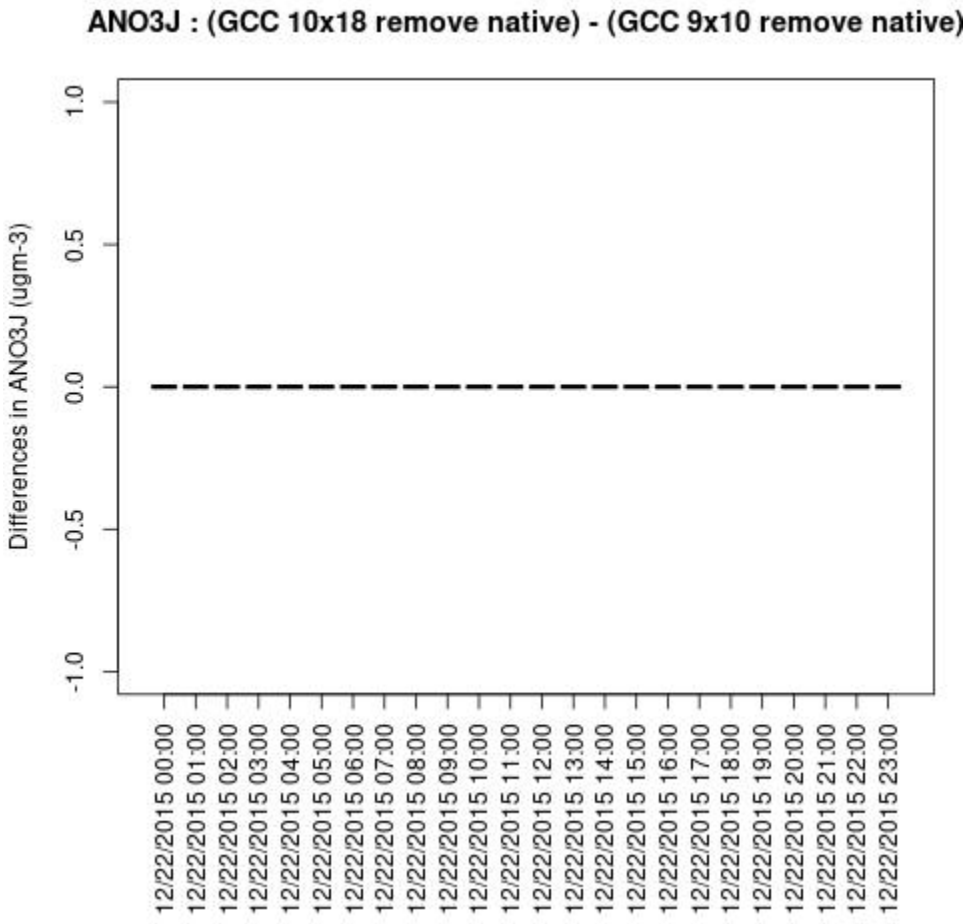
On your local terminal: `host +`

Example output plots are available for the CONUS Benchmark in the following directory

When NPCOL is fixed, we are seeing no difference in the answers.

Example comparison of 10x18 compared to 9x10 with the -march=native compiler flag removed

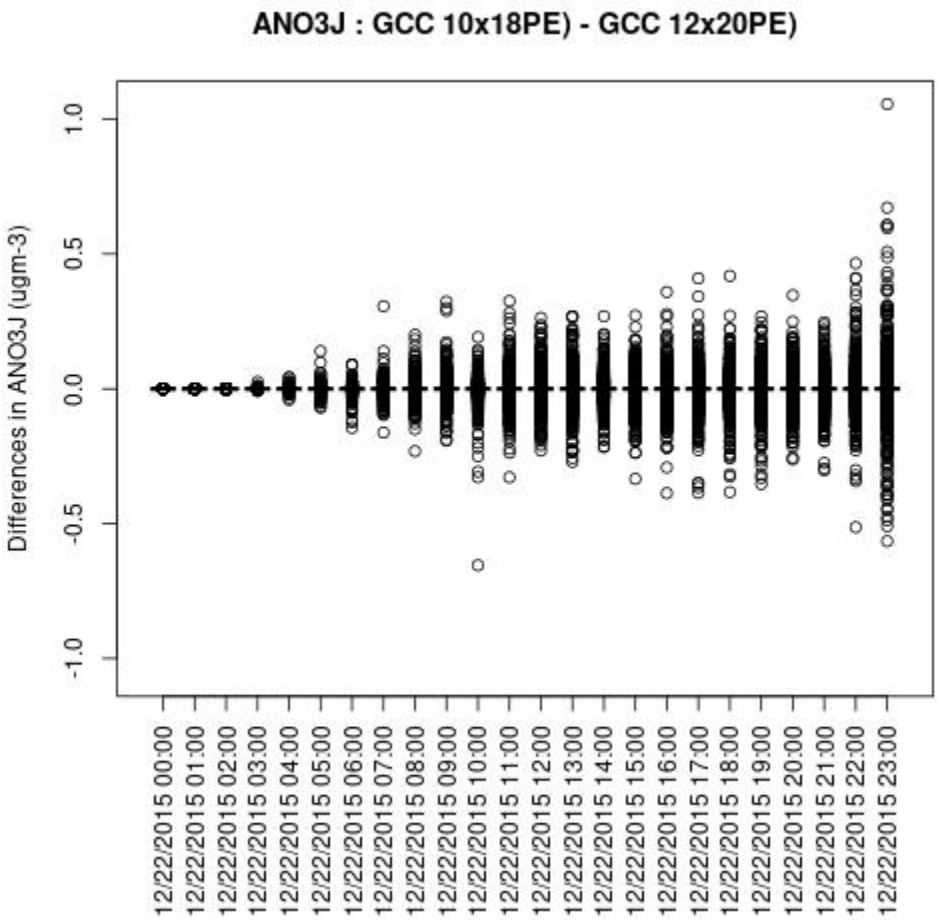
Box Plot for ANO3J when -march=native compiler flag is removed



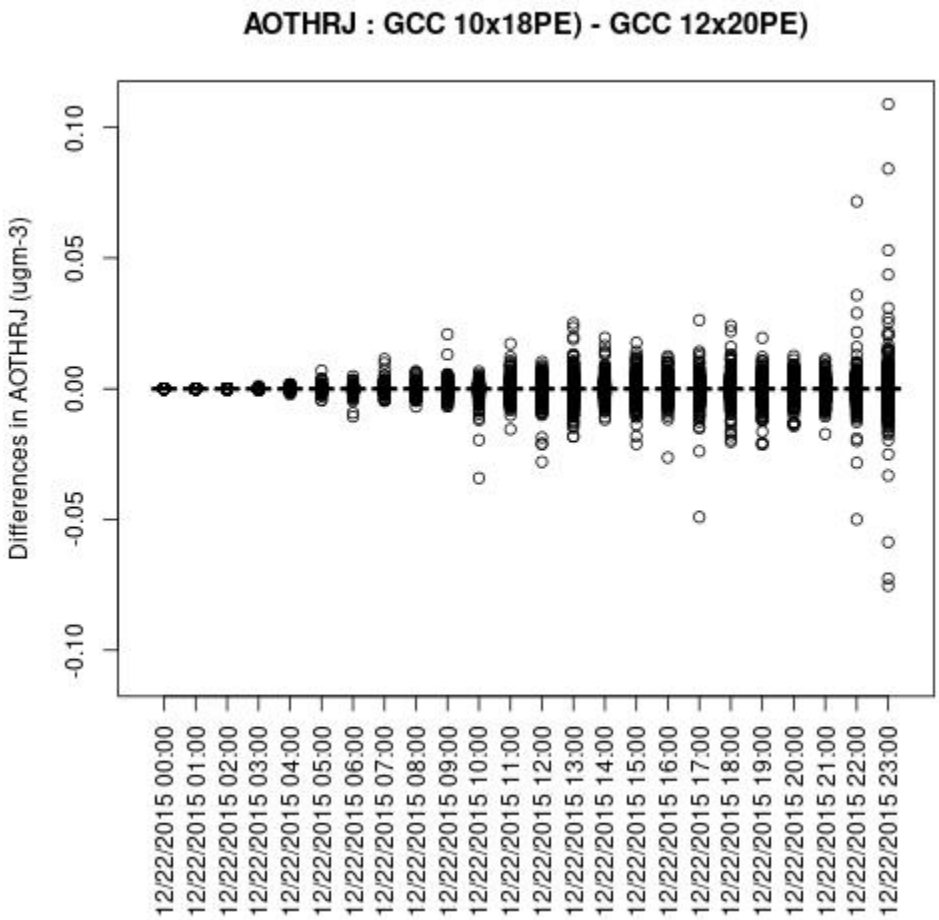
Box plot shows no difference between ACONC output for a CMAQv5.3.3 run using different PE configurations as long as NPCOL is fixed (this is true for all species that were plotted (AOTHRJ, CO, NH3, NO2, O3, OH, SO2), or when not using march=native in the compiler flag

Box plot shows a difference betweeen ACONC output for a CMAQv5.3.3 run using different PE configurations when NPCOL is different, if march=native option is used

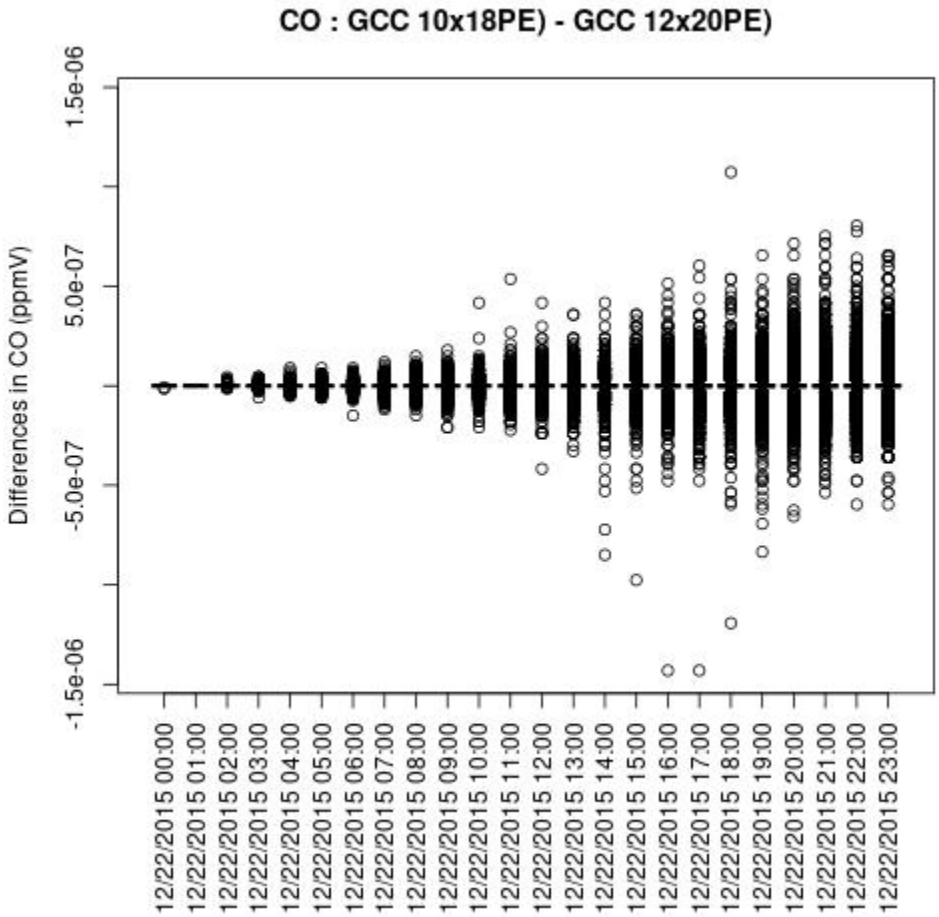
ANO3J


ANO3J : GCC 10x18PE) - GCC 12x20PE)
Differences in ANO3J (ugm-3)
1.0
0.5
0.0
-0.5
-1.0
12/22/2015 00:00
12/22/2015 01:00
12/22/2015 02:00
12/22/2015 03:00
12/22/2015 04:00
12/22/2015 05:00
12/22/2015 06:00
12/22/2015 07:00
12/22/2015 08:00
12/22/2015 09:00
12/22/2015 10:00
12/22/2015 11:00
12/22/2015 12:00
12/22/2015 13:00
12/22/2015 14:00
12/22/2015 15:00
12/22/2015 16:00
12/22/2015 17:00
12/22/2015 18:00
12/22/2015 19:00
12/22/2015 20:00
12/22/2015 21:00
12/22/2015 22:00
12/22/2015 23:00

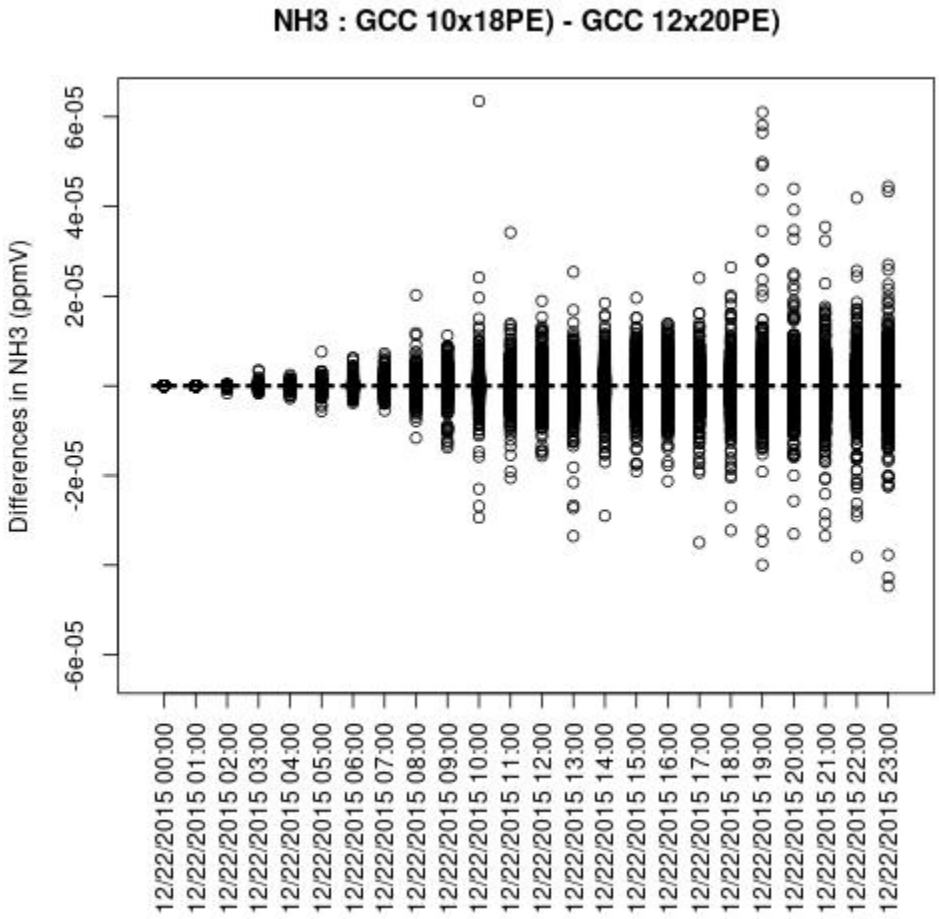
AOTHRJ


AOTHRJ : GCC 10x18PE) - GCC 12x20PE)
Differences in AOTHRJ (ugm-3)
0.10
0.05
0.00
-0.05
-0.10
12/22/2015 00:00
12/22/2015 01:00
12/22/2015 02:00
12/22/2015 03:00
12/22/2015 04:00
12/22/2015 05:00
12/22/2015 06:00
12/22/2015 07:00
12/22/2015 08:00
12/22/2015 09:00
12/22/2015 10:00
12/22/2015 11:00
12/22/2015 12:00
12/22/2015 13:00
12/22/2015 14:00
12/22/2015 15:00
12/22/2015 16:00
12/22/2015 17:00
12/22/2015 18:00
12/22/2015 19:00
12/22/2015 20:00
12/22/2015 21:00
12/22/2015 22:00
12/22/2015 23:00

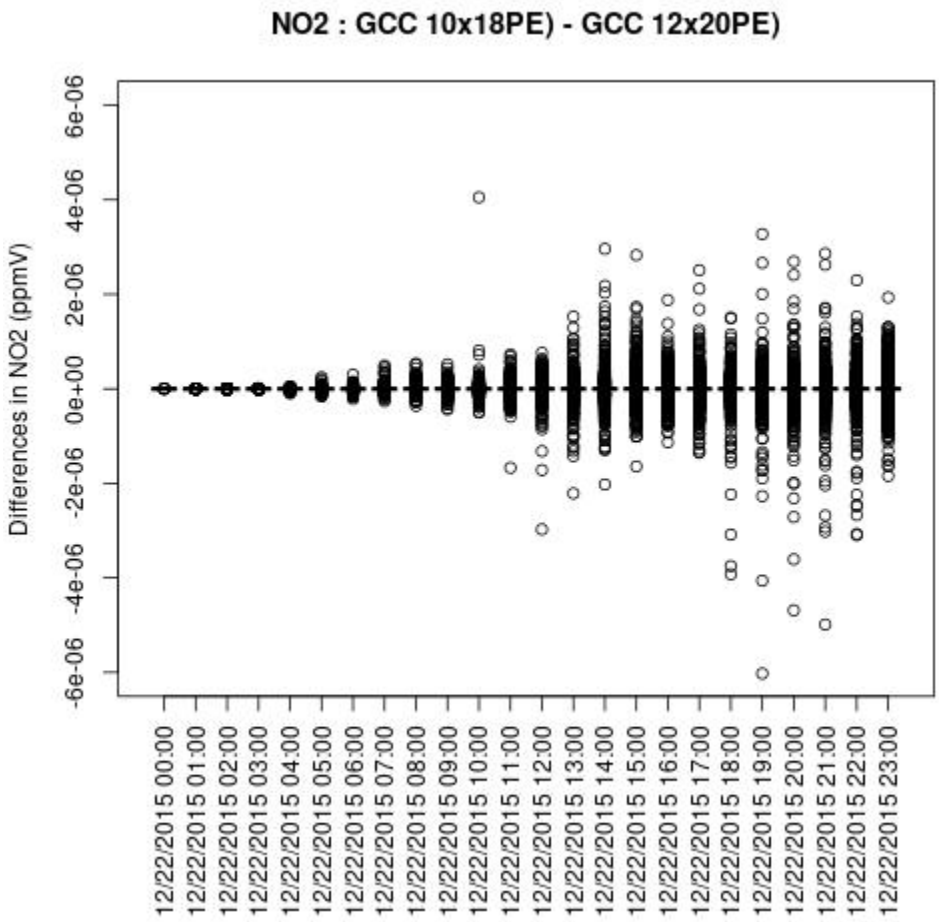
CO


CO : GCC 10x18PE) - GCC 12x20PE)
Differences in CO (ppmV)
1.5e-06
5.0e-07
-5.0e-07
-1.5e-06
12/22/2015 00:00
12/22/2015 01:00
12/22/2015 02:00
12/22/2015 03:00
12/22/2015 04:00
12/22/2015 05:00
12/22/2015 06:00
12/22/2015 07:00
12/22/2015 08:00
12/22/2015 09:00
12/22/2015 10:00
12/22/2015 11:00
12/22/2015 12:00
12/22/2015 13:00
12/22/2015 14:00
12/22/2015 15:00
12/22/2015 16:00
12/22/2015 17:00
12/22/2015 18:00
12/22/2015 19:00
12/22/2015 20:00
12/22/2015 21:00
12/22/2015 22:00
12/22/2015 23:00

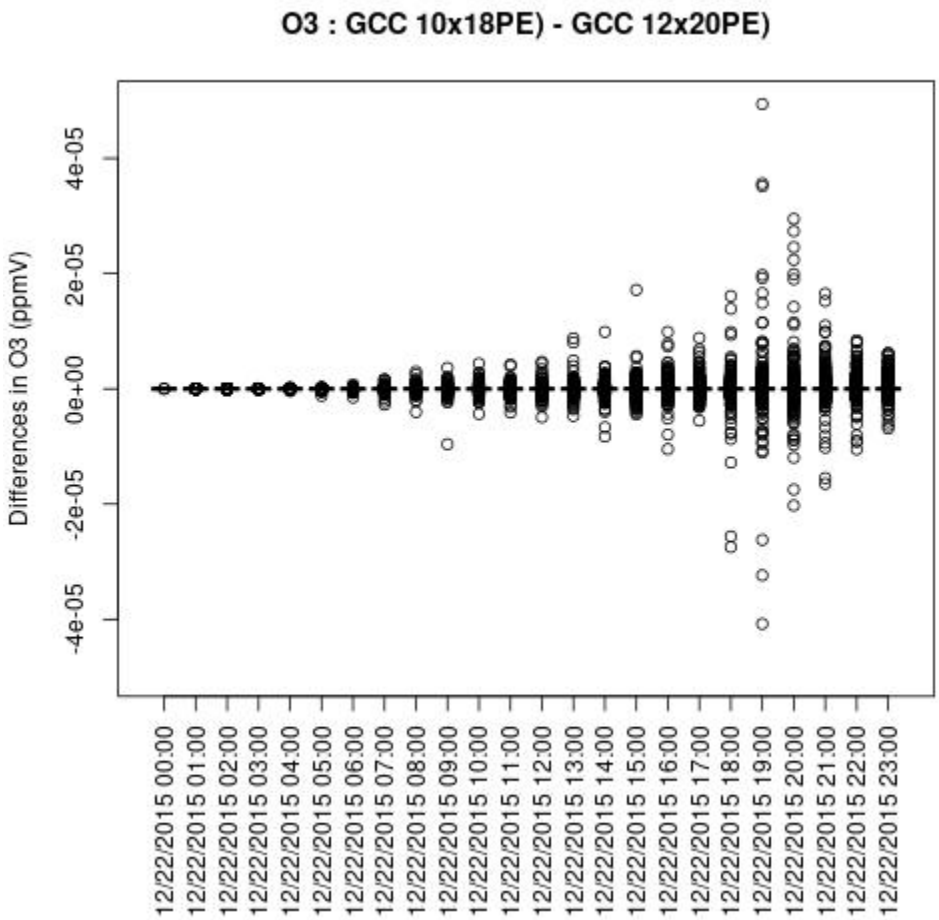
NH3


NH3 : GCC 10x18PE) - GCC 12x20PE)
Differences in NH3 (ppmV)
-6e-05
-2e-05
2e-05
4e-05
6e-05
12/22/2015 00:00
12/22/2015 01:00
12/22/2015 02:00
12/22/2015 03:00
12/22/2015 04:00
12/22/2015 05:00
12/22/2015 06:00
12/22/2015 07:00
12/22/2015 08:00
12/22/2015 09:00
12/22/2015 10:00
12/22/2015 11:00
12/22/2015 12:00
12/22/2015 13:00
12/22/2015 14:00
12/22/2015 15:00
12/22/2015 16:00
12/22/2015 17:00
12/22/2015 18:00
12/22/2015 19:00
12/22/2015 20:00
12/22/2015 21:00
12/22/2015 22:00
12/22/2015 23:00

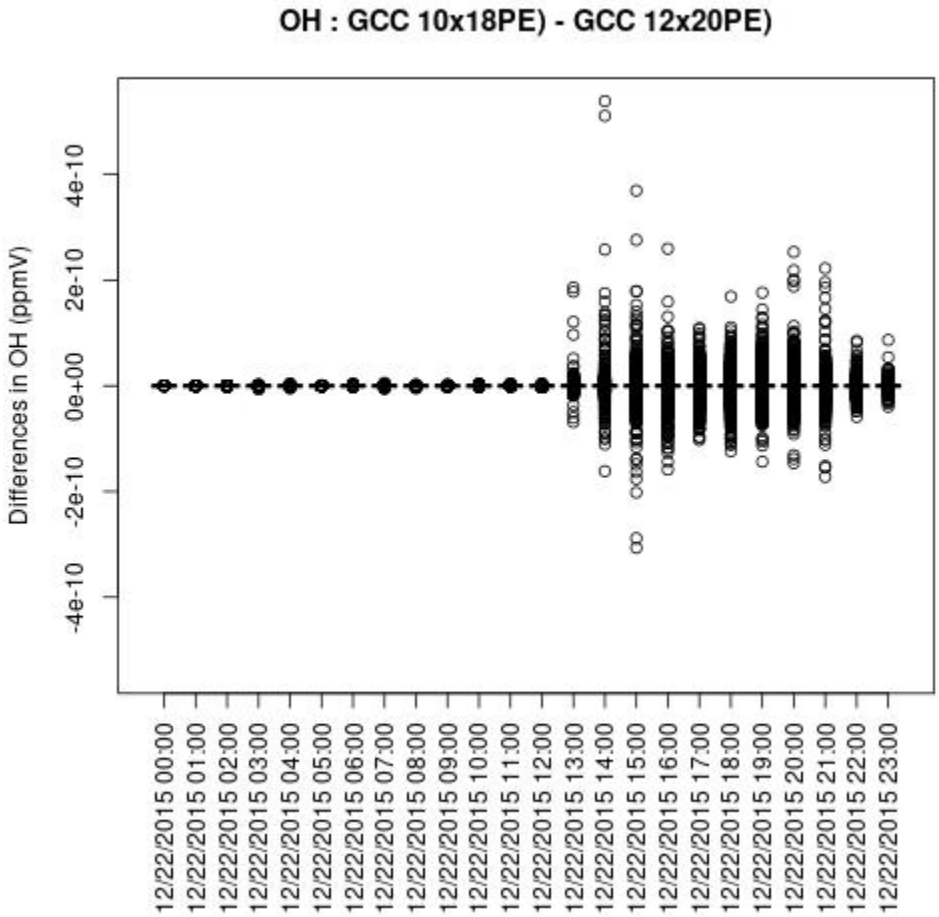
NO2


NO2 : GCC 10x18PE) - GCC 12x20PE)
Differences in NO2 (ppmV)
-6e-06
-4e-06
-2e-06
0e+00
2e-06
4e-06
6e-06
12/22/2015 00:00
12/22/2015 01:00
12/22/2015 02:00
12/22/2015 03:00
12/22/2015 04:00
12/22/2015 05:00
12/22/2015 06:00
12/22/2015 07:00
12/22/2015 08:00
12/22/2015 09:00
12/22/2015 10:00
12/22/2015 11:00
12/22/2015 12:00
12/22/2015 13:00
12/22/2015 14:00
12/22/2015 15:00
12/22/2015 16:00
12/22/2015 17:00
12/22/2015 18:00
12/22/2015 19:00
12/22/2015 20:00
12/22/2015 21:00
12/22/2015 22:00
12/22/2015 23:00

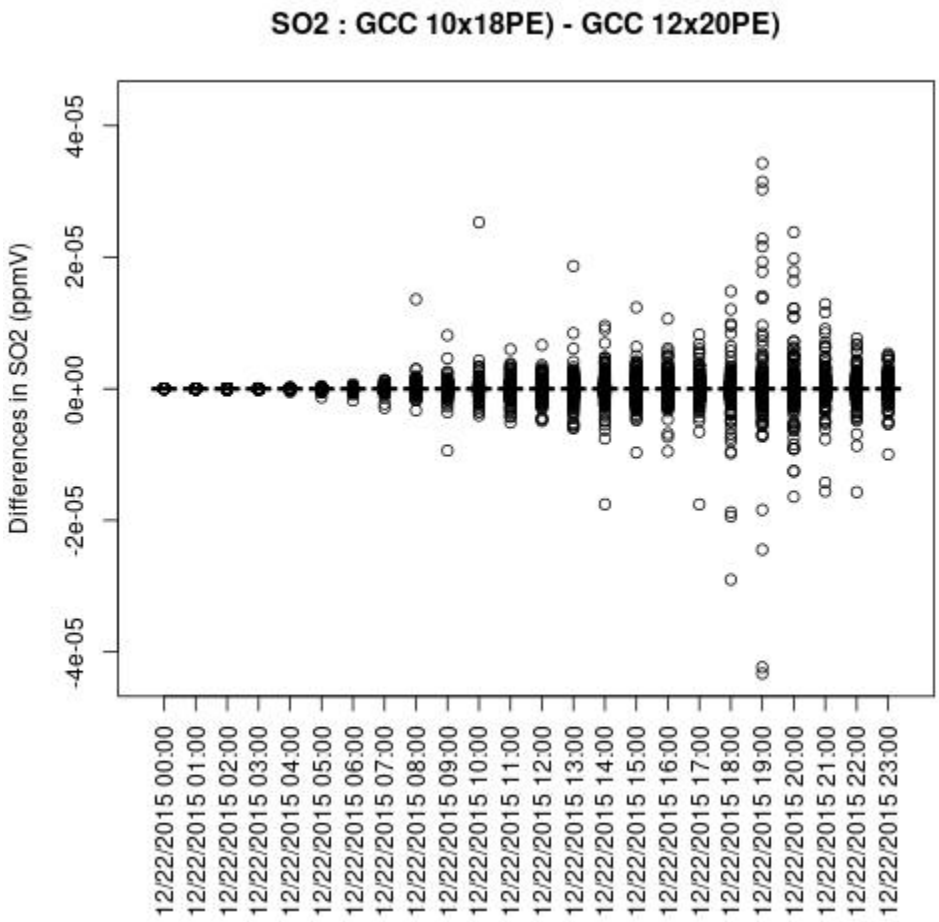
O3


O3 : GCC 10x18PE) - GCC 12x20PE)
Differences in O3 (ppmV)
-4e-05
-2e-05
0e+00
2e-05
4e-05
12/22/2015 00:00
12/22/2015 01:00
12/22/2015 02:00
12/22/2015 03:00
12/22/2015 04:00
12/22/2015 05:00
12/22/2015 06:00
12/22/2015 07:00
12/22/2015 08:00
12/22/2015 09:00
12/22/2015 10:00
12/22/2015 11:00
12/22/2015 12:00
12/22/2015 13:00
12/22/2015 14:00
12/22/2015 15:00
12/22/2015 16:00
12/22/2015 17:00
12/22/2015 18:00
12/22/2015 19:00
12/22/2015 20:00
12/22/2015 21:00
12/22/2015 22:00
12/22/2015 23:00

OH

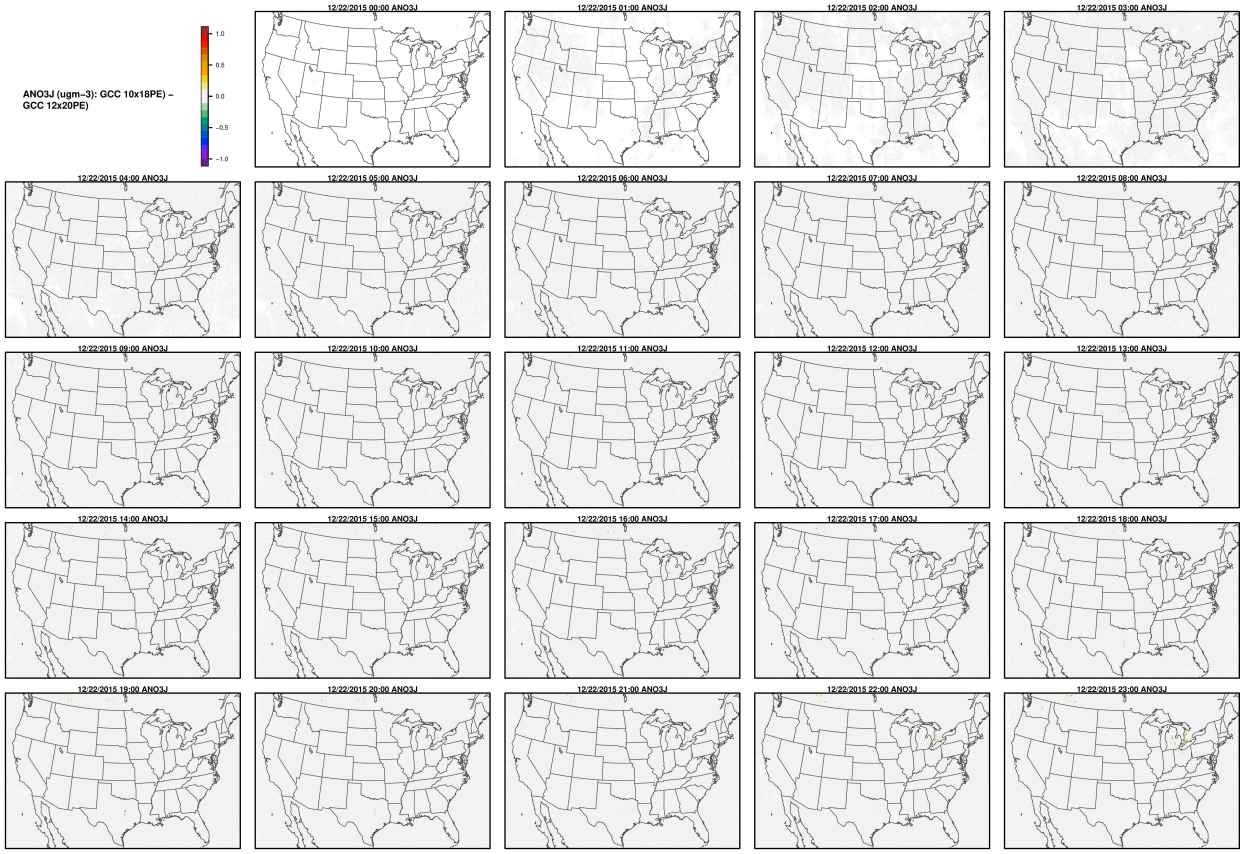

OH : GCC 10x18PE) - GCC 12x20PE)
Differences in OH (ppmV)
-4e-10
-2e-10
0e+00
2e-10
4e-10
12/22/2015 00:00
12/22/2015 01:00
12/22/2015 02:00
12/22/2015 03:00
12/22/2015 04:00
12/22/2015 05:00
12/22/2015 06:00
12/22/2015 07:00
12/22/2015 08:00
12/22/2015 09:00
12/22/2015 10:00
12/22/2015 11:00
12/22/2015 12:00
12/22/2015 13:00
12/22/2015 14:00
12/22/2015 15:00
12/22/2015 16:00
12/22/2015 17:00
12/22/2015 18:00
12/22/2015 19:00
12/22/2015 20:00
12/22/2015 21:00
12/22/2015 22:00
12/22/2015 23:00

SO2

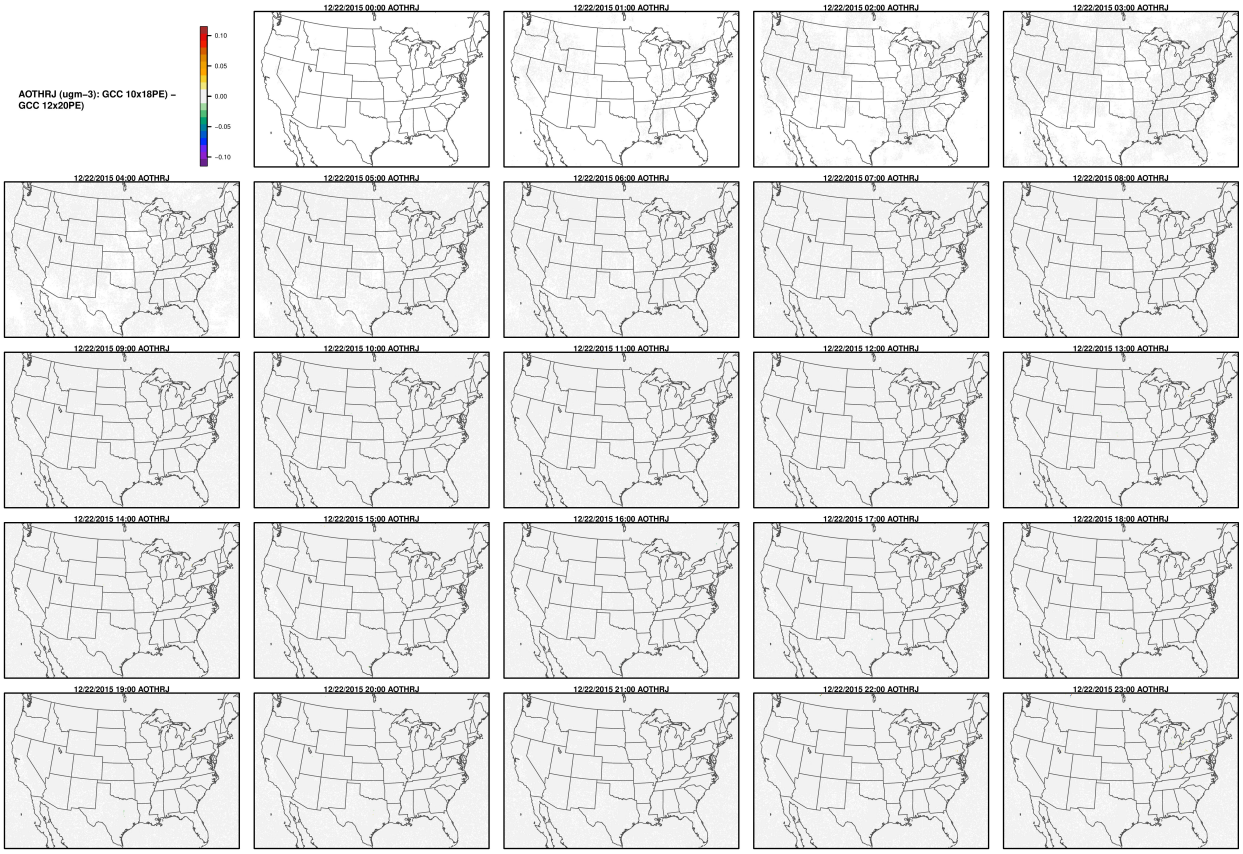


Spatial Plot for when NPCOL is different and when -march=native compiler flag is used

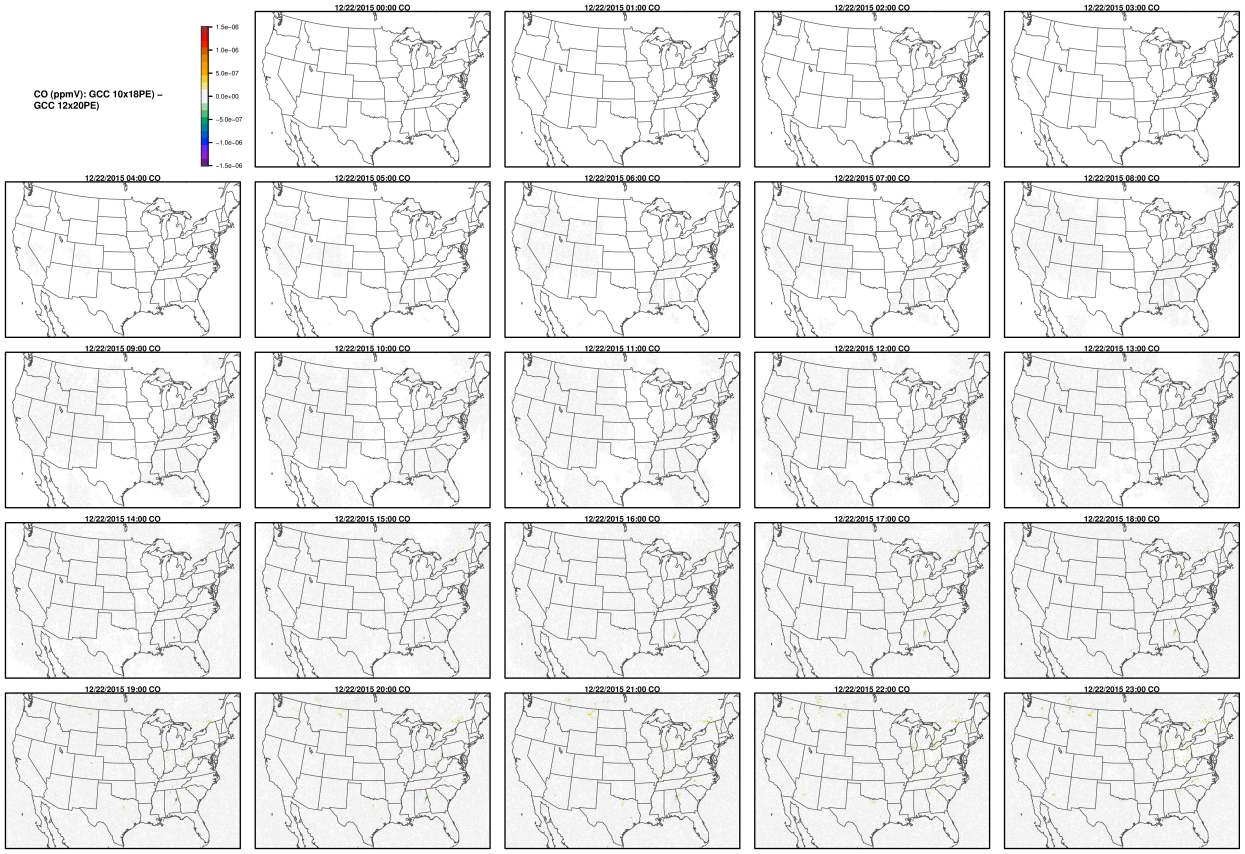
ANO3J


ANO3J (ugm−3): GCC 10x18PE) – GCC 12x20PE)
1.0
0.5
0.0
−0.5
−1.0
12/22/2015 00:00 ANO3J
12/22/2015 01:00 ANO3J
12/22/2015 02:00 ANO3J
12/22/2015 03:00 ANO3J
12/22/2015 04:00 ANO3J
12/22/2015 05:00 ANO3J
12/22/2015 06:00 ANO3J
12/22/2015 07:00 ANO3J
12/22/2015 08:00 ANO3J
12/22/2015 09:00 ANO3J
12/22/2015 10:00 ANO3J
12/22/2015 11:00 ANO3J
12/22/2015 12:00 ANO3J
12/22/2015 13:00 ANO3J
12/22/2015 14:00 ANO3J
12/22/2015 15:00 ANO3J
12/22/2015 16:00 ANO3J
12/22/2015 17:00 ANO3J
12/22/2015 18:00 ANO3J
12/22/2015 19:00 ANO3J
12/22/2015 20:00 ANO3J
12/22/2015 21:00 ANO3J
12/22/2015 22:00 ANO3J
12/22/2015 23:00 ANO3J

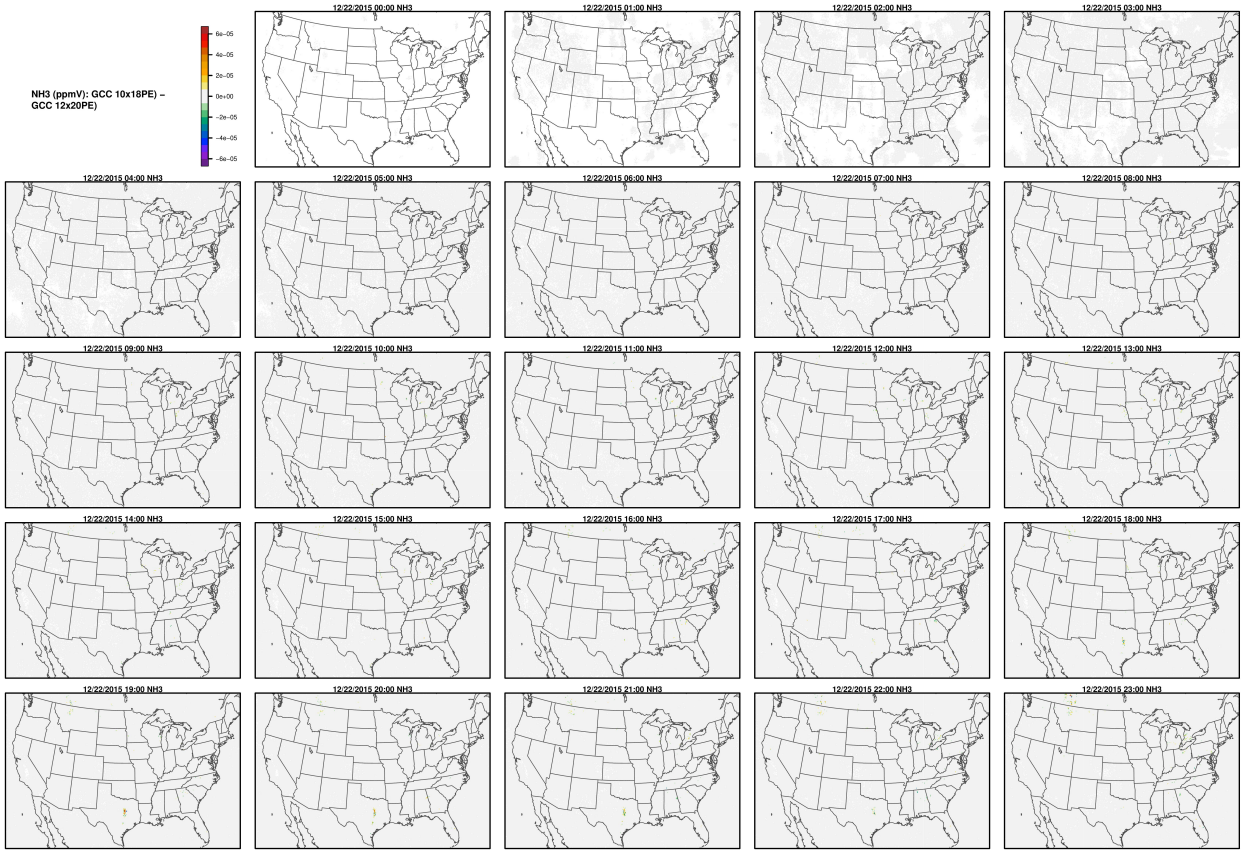
AOTHRJ


AOTHRJ (ugm−3): GCC 10x18PE) − GCC 12x20PE)
0.10
0.05
0.00
−0.05
−0.10
12/22/2015 00:00 AOTHRJ
12/22/2015 01:00 AOTHRJ
12/22/2015 02:00 AOTHRJ
12/22/2015 03:00 AOTHRJ
12/22/2015 04:00 AOTHRJ
12/22/2015 05:00 AOTHRJ
12/22/2015 06:00 AOTHRJ
12/22/2015 07:00 AOTHRJ
12/22/2015 08:00 AOTHRJ
12/22/2015 09:00 AOTHRJ
12/22/2015 10:00 AOTHRJ
12/22/2015 11:00 AOTHRJ
12/22/2015 12:00 AOTHRJ
12/22/2015 13:00 AOTHRJ
12/22/2015 14:00 AOTHRJ
12/22/2015 15:00 AOTHRJ
12/22/2015 16:00 AOTHRJ
12/22/2015 17:00 AOTHRJ
12/22/2015 18:00 AOTHRJ
12/22/2015 19:00 AOTHRJ
12/22/2015 20:00 AOTHRJ
12/22/2015 21:00 AOTHRJ
12/22/2015 22:00 AOTHRJ
12/22/2015 23:00 AOTHRJ

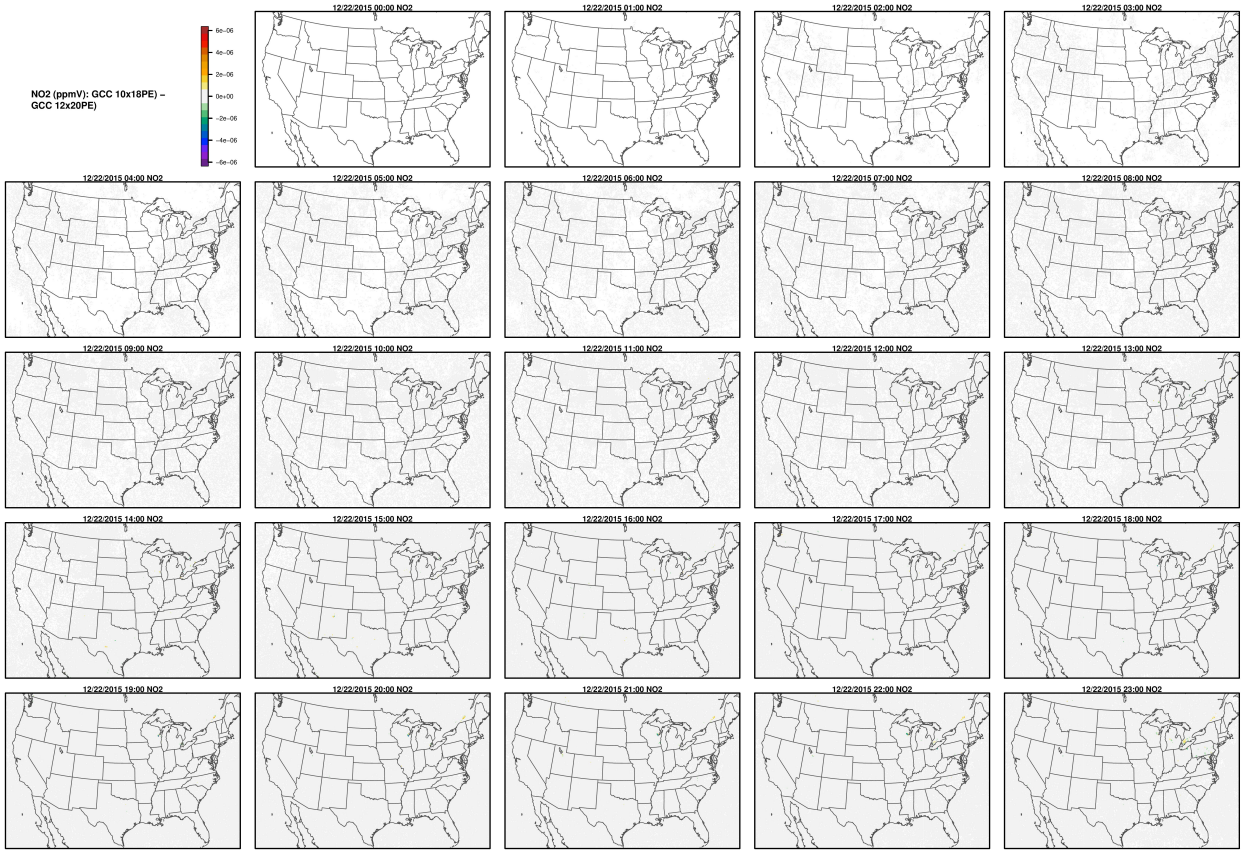
CO


CO (ppmV): GCC 10x18PE) – GCC 12x20PE)
1.5e−06
1.0e−06
5.0e−07
0.0e+00
−5.0e−07
−1.0e−06
−1.5e−06
12/22/2015 00:00 CO
12/22/2015 01:00 CO
12/22/2015 02:00 CO
12/22/2015 03:00 CO
12/22/2015 04:00 CO
12/22/2015 05:00 CO
12/22/2015 06:00 CO
12/22/2015 07:00 CO
12/22/2015 08:00 CO
12/22/2015 09:00 CO
12/22/2015 10:00 CO
12/22/2015 11:00 CO
12/22/2015 12:00 CO
12/22/2015 13:00 CO
12/22/2015 14:00 CO
12/22/2015 15:00 CO
12/22/2015 16:00 CO
12/22/2015 17:00 CO
12/22/2015 18:00 CO
12/22/2015 19:00 CO
12/22/2015 20:00 CO
12/22/2015 21:00 CO
12/22/2015 22:00 CO
12/22/2015 23:00 CO

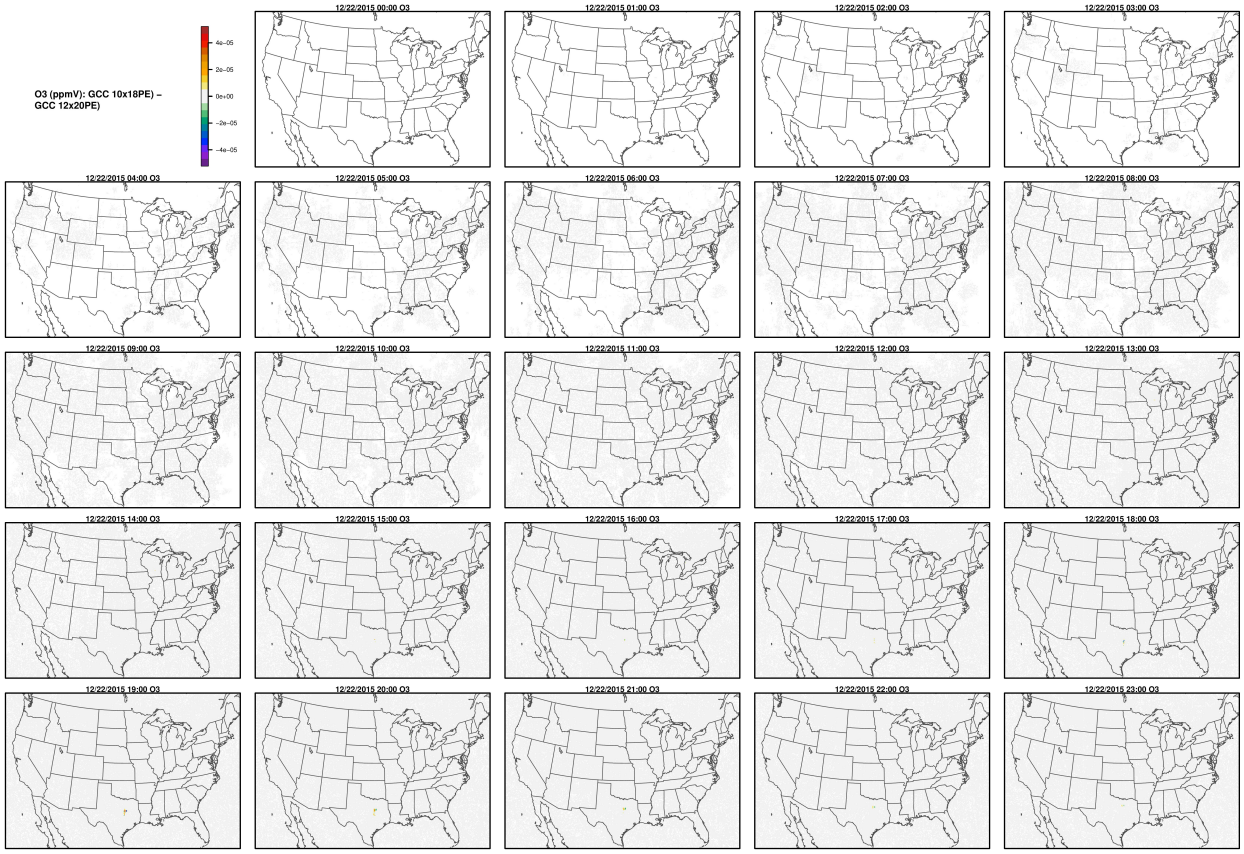
NH3


NH3 (ppmV): GCC 10x18PE) – GCC 12x20PE)
6e−05
4e−05
2e−05
0e+00
−2e−05
−4e−05
−6e−05
12/22/2015 00:00 NH3
12/22/2015 01:00 NH3
12/22/2015 02:00 NH3
12/22/2015 03:00 NH3
12/22/2015 04:00 NH3
12/22/2015 05:00 NH3
12/22/2015 06:00 NH3
12/22/2015 07:00 NH3
12/22/2015 08:00 NH3
12/22/2015 09:00 NH3
12/22/2015 10:00 NH3
12/22/2015 11:00 NH3
12/22/2015 12:00 NH3
12/22/2015 13:00 NH3
12/22/2015 14:00 NH3
12/22/2015 15:00 NH3
12/22/2015 16:00 NH3
12/22/2015 17:00 NH3
12/22/2015 18:00 NH3
12/22/2015 19:00 NH3
12/22/2015 20:00 NH3
12/22/2015 21:00 NH3
12/22/2015 22:00 NH3
12/22/2015 23:00 NH3

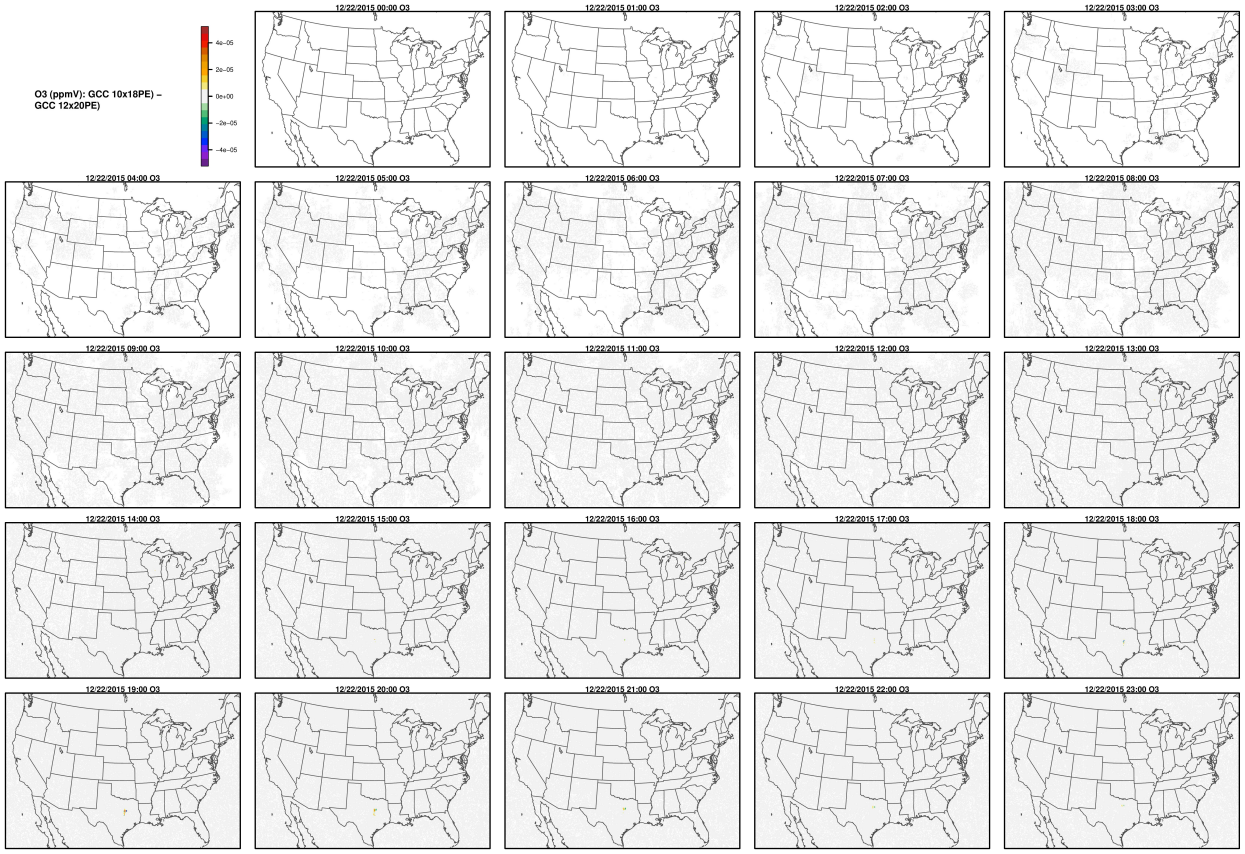
NO2


NO2 (ppmV): GCC 10x18PE) – GCC 12x20PE)
6e-06
4e-06
2e-06
0e+00
-2e-06
-4e-06
-6e-06
12/22/2015 00:00 NO2
12/22/2015 01:00 NO2
12/22/2015 02:00 NO2
12/22/2015 03:00 NO2
12/22/2015 04:00 NO2
12/22/2015 05:00 NO2
12/22/2015 06:00 NO2
12/22/2015 07:00 NO2
12/22/2015 08:00 NO2
12/22/2015 09:00 NO2
12/22/2015 10:00 NO2
12/22/2015 11:00 NO2
12/22/2015 12:00 NO2
12/22/2015 13:00 NO2
12/22/2015 14:00 NO2
12/22/2015 15:00 NO2
12/22/2015 16:00 NO2
12/22/2015 17:00 NO2
12/22/2015 18:00 NO2
12/22/2015 19:00 NO2
12/22/2015 20:00 NO2
12/22/2015 21:00 NO2
12/22/2015 22:00 NO2
12/22/2015 23:00 NO2

O3


O3 (ppmV): GCC 10x18PE) – GCC 12x20PE)
4e-05
2e-05
0e+00
-2e-05
-4e-05
12/22/2015 00:00 O3
12/22/2015 01:00 O3
12/22/2015 02:00 O3
12/22/2015 03:00 O3
12/22/2015 04:00 O3
12/22/2015 05:00 O3
12/22/2015 06:00 O3
12/22/2015 07:00 O3
12/22/2015 08:00 O3
12/22/2015 09:00 O3
12/22/2015 10:00 O3
12/22/2015 11:00 O3
12/22/2015 12:00 O3
12/22/2015 13:00 O3
12/22/2015 14:00 O3
12/22/2015 15:00 O3
12/22/2015 16:00 O3
12/22/2015 17:00 O3
12/22/2015 18:00 O3
12/22/2015 19:00 O3
12/22/2015 20:00 O3
12/22/2015 21:00 O3
12/22/2015 22:00 O3
12/22/2015 23:00 O3

OH



SO2 qa_plots/spatial_plots/16x8_vs_8x16/SO2_MAPS_CMAQv5.3.3wGCC16x8pe_vs_CMAQv5.3.3wGCC8x16pe.jpg

3.9 Compare Timing of CMAQ Routines

Compare the timing of CMAQ Routines for two different run configurations.

3.9.1 Parse timings from the log file

Compare CONUS CycleCloud Runs

Note: CycleCloud Configurations can impact the model run times.

It is up the the user, as to what model run configurations are used to run CMAQ on the CycleCloud. The following configurations may impact the run time of the model.

- For different PE configurations, using 36 cpus out of 44 cpus on HC44rs

```
NPCOL x NPROW
```

  - [ ] 6x6 #SBATCH –nodes=1, #SBATCH –ntasks-per-node=36
  - [ ] 9x12 #SBATCH –nodes=3, #SBATCH –ntasks-per-node=36
  - [ ] 12x12 #SBATCH –nodes=4, #SBATCH –ntasks-per-node=36
  - [ ] 16x16 #SBATCH –nodes=8, #SBATCH –ntasks-per-node=36
  - [ ] 16x18 #SBATCH –nodes=8, #SBATCH –ntasks-per-node=36
- For different PE configurations, using 18 cpus out of 44 cpus on HC44rs
  - [ ] 3x6 #SBATCH –nodes=1, #SBATCH –ntasks-per-node=18
  - [ ] 9x14 #SBATCH –nodes=7, #SBATCH –ntasks-per-node=18
- For different compute nodes
  - [ ] HC44rs (44 cpus) - with Elastic Fabric Adapter (see above)
  - [ ] HBv120 (120 cpus - with Elastic Fabric Adapter
- For different PE configurations, using 36 cpus out of 120 cpus on HBv120s
  - [ ] 6x6 #SBATCH –nodes=2, #SBATCH –ntasks-per-node=18
  - [ ] 6x6 #SBATCH –nodes=1, #SBATCH –ntasks-per-node=36
- For different PE configurations, using 90 cpus out of 120 cpus on HBv120s
  - [ ] 9x10 #SBATCH –nodes=1, #SBATCH –ntasks-per-node=90
  - [ ] 10x18 #SBATCH –nodes=2, #SBATCH –ntasks-per-node=90
  - [ ] 15x18 #SBATCH –nodes=3, #SBATCH –ntasks-per-node=90
  - [ ] 20x18 #SBATCH –nodes=4, #SBATCH –ntasks-per-node=90
- For with and without Elastic Fabric and Elastic Netaork Adapter
- For with and without network placement
- For /shared versus /data
  - [ ] input data copied to /shared
  - [ ] input data copied to /data
  - [ ] input data copied to /mnt resource (local to each node)

Edit the R script

First check to see what log files are available:

`ls -lrt /shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts/*.log`

Modify the name of the log file to match what is avaible on your system.

`cd /shared/pcluster-cmaq/qa_scripts`

`vi parse_timing_pcluster.r`

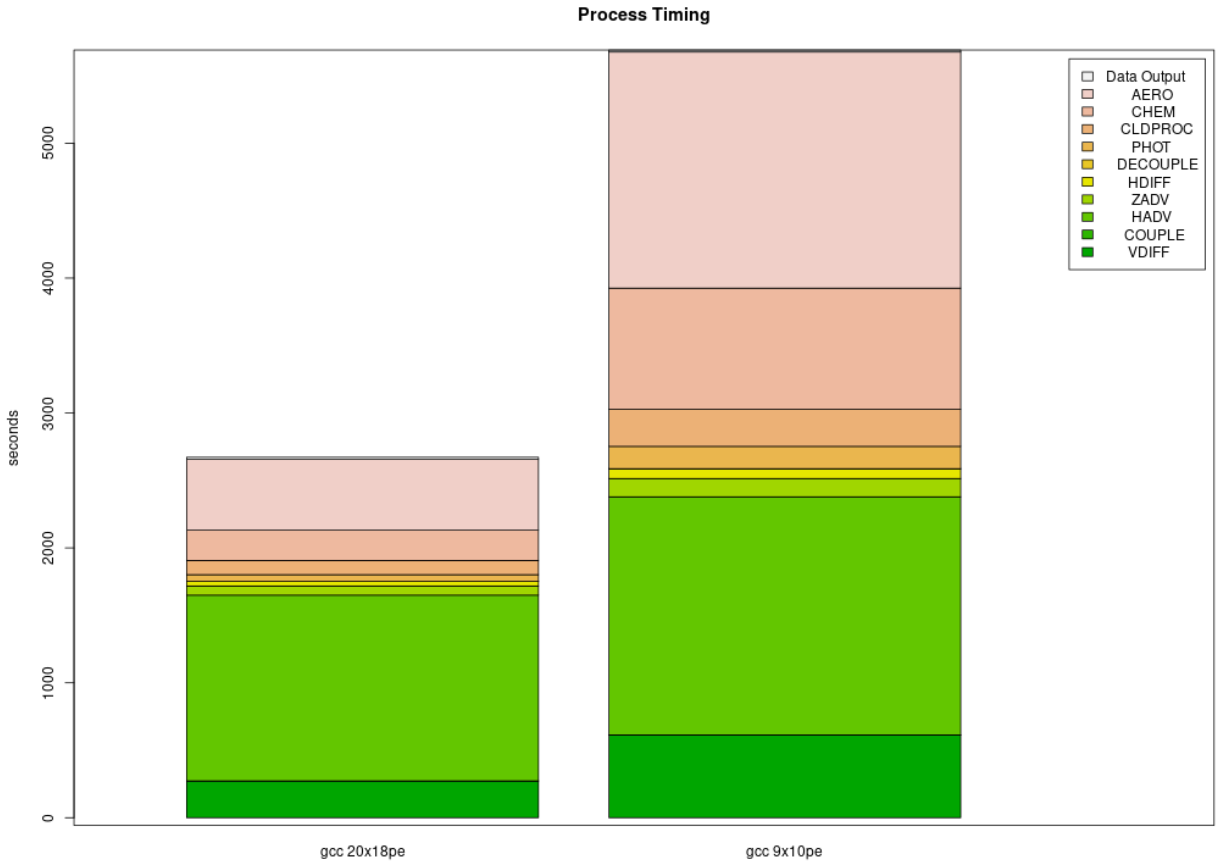
Edit the following section of the script to specify the log file names available on your ParallelCluster

```
sens.dir  <- '/shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts/'
base.dir  <- '/shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts/'
files     <- dir(sens.dir, pattern ='run_cctmv5.3.3_Bench_2016_12US2.20x18pe.2day.sleep.
↪cyclecloud.log' )
b.files <- dir(base.dir,pattern='run_cctmv5.3.3_Bench_2016_12US2.9x10pe.2day_remove_
↪native_sleep.cyclecloud.log')
#Compilers <- c('intel','gcc','pgi')
Compilers <- c('gcc')
# name of the base case timing. I am using the current master branch from the CMAQ_Dev␣
↪repository.
# The project directory name is used for the sensitivity case.
base.name <- '9x10pe'
sens.name <- '20x18pe'
```

Use parse_timing.r script to examine timings of each process in CMAQ

```
cd qa_scripts
Rscript parse_timing.r
```

Timing Plot Comparing GCC run on 20x18 versus 9x10.



3.10 Copy Output to S3 Bucket

Copy output from CycleCloud to an S3 Bucket

3.10.1 Copy Output Data and Run script logs to S3 Bucket

Note, you will need permissions to copy to a S3 Bucket. see S3 Access Control

Currently, the bucket listed below has ACL turned off see S3 disable ACL

See example of sharing bucket across accounts. see Bucket owner granting cross-account permissions

Copy scripts and logs to /shared

The CTM_LOG files do not contain any information about the compute nodes that the jobs were run on. Note, it is important to keep a record of the NPCOL, NPROW setting and the number of nodes and tasks used as specified in the run script: #SBATCH –nodes=16 #SBATCH –ntasks-per-node=8 It is also important to know what volume was used to read and write the input and output data, so it is recommended to save a copy of the standard out and error logs, and a copy of the run scripts to the OUTPUT directory for each benchmark.

```
cd /shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts
cp run*.log /shared/data/output
cp run*.csh /shared/data/output
```

Examine the output files

```
cd /shared/data/output/output_CCTM_v533_gcc_2016_CONUS_16x18pe_full
ls -lht
```

output:

```
total 173G
drwxrwxr-x 2 ubuntu ubuntu 145K Jan  5 23:53 LOGS
-rw-rw-r-- 1 ubuntu ubuntu 3.2G Jan  5 23:53 CCTM_CGRID_v533_gcc_2016_CONUS_16x18pe_full_
↪20151223.nc
-rw-rw-r-- 1 ubuntu ubuntu 2.2G Jan  5 23:52 CCTM_ACONC_v533_gcc_2016_CONUS_16x18pe_full_
↪20151223.nc
-rw-rw-r-- 1 ubuntu ubuntu  78G Jan  5 23:52 CCTM_CONC_v533_gcc_2016_CONUS_16x18pe_full_
↪20151223.nc
-rw-rw-r-- 1 ubuntu ubuntu 348M Jan  5 23:52 CCTM_APMDIAG_v533_gcc_2016_CONUS_16x18pe_
↪full_20151223.nc
-rw-rw-r-- 1 ubuntu ubuntu 1.5G Jan  5 23:52 CCTM_WETDEP1_v533_gcc_2016_CONUS_16x18pe_
↪full_20151223.nc
-rw-rw-r-- 1 ubuntu ubuntu 1.7G Jan  5 23:52 CCTM_DRYDEP_v533_gcc_2016_CONUS_16x18pe_
↪full_20151223.nc
-rw-rw-r-- 1 ubuntu ubuntu 3.6K Jan  5 23:22 CCTM_v533_gcc_2016_CONUS_16x18pe_full_
↪20151223.cfg
-rw-rw-r-- 1 ubuntu ubuntu 3.2G Jan  5 23:22 CCTM_CGRID_v533_gcc_2016_CONUS_16x18pe_full_
↪20151222.nc
-rw-rw-r-- 1 ubuntu ubuntu 2.2G Jan  5 23:21 CCTM_ACONC_v533_gcc_2016_CONUS_16x18pe_full_
↪20151222.nc
```

(continues on next page)

(continued from previous page)

```
-rw-rw-r-- 1 ubuntu ubuntu  78G Jan  5 23:21 CCTM_CONC_v533_gcc_2016_CONUS_16x18pe_full_
↪20151222.nc
-rw-rw-r-- 1 ubuntu ubuntu 348M Jan  5 23:21 CCTM_APMDIAG_v533_gcc_2016_CONUS_16x18pe_
↪full_20151222.nc
-rw-rw-r-- 1 ubuntu ubuntu 1.5G Jan  5 23:21 CCTM_WETDEP1_v533_gcc_2016_CONUS_16x18pe_
↪full_20151222.nc
-rw-rw-r-- 1 ubuntu ubuntu 1.7G Jan  5 23:21 CCTM_DRYDEP_v533_gcc_2016_CONUS_16x18pe_
↪full_20151222.nc
-rw-rw-r-- 1 ubuntu ubuntu 3.6K Jan  5 22:49 CCTM_v533_gcc_2016_CONUS_16x18pe_full_
↪20151222.cfg
```

Check disk space

```
 du -sh
173G    .
```

Copy the output to an S3 Bucket

Examine the example script

```
cd s3_scripts
cat s3_upload.HBv3-120.csh
```

output:

```
#!/bin/csh -f
# Script to upload output data to S3 bucket
# NOTE: a new bucket needs to be created to store each set of cluster runs

cd /shared/build/openmpi_gcc/CMAQ_v533/CCTM/scripts
cp run*.log /shared/data/output
cp run*.csh /shared/data/output

aws s3 mb s3://hbv3-120-compute-conus-output
aws s3 cp --recursive /shared/data/output/ s3://hbv3-120-compute-conus-output/
aws s3 cp --recursive /shared/data/POST s3://hbv3-120-compute-conus-output/
```

If you do not have permissions to write to the s3 bucket listed above, you will need to edit the script to specify the s3 bucket that you have permissions to write to. In addition, edit the script to include a new date stamp, then run the script to copy all of the CMAQ output and logs to the S3 bucket.

```
./s3_upload.HBv3-120.csh
```

3.11 Logout and Delete CycleCloud

Logout and delete the CycleCloud when you are done to avoid incurring costs.

3.11.1 Link to Azure Instructions on how to logout and delete cyclecloud

How to Terminate Cluster Resources

Tutorial to Clean-up Cluster Resources

3.12 Performance Optimization

Timing information and scaling plots to assist users in optimizing the performance of their Cycle Cloud HPC Cluster.

Performance Optimization for Single Virtual Machine

3.12.1 Right-sizing Compute Nodes for a Single Virtual Machine.

Selection of the compute nodes depends on the domain size and resolution for the CMAQ case, and what your model run time requirements are. Larger hardware and memory configurations may also be required for instrumented versions of CMAQ incuding CMAQ-ISAM and CMAQ-DDM3D. Running on a single virtual machine requires that the user know how CMAQ scales for the domain of interest.

3.12.2 An explanation of why a scaling analysis is required for Single Node

Quote from the following link.

"IMPORTANT: The optimal value of –nodes and –ntasks for a parallel code must be determined empirically by conducting a scaling analysis. As these quantities increase, the parallel efficiency tends to decrease. The parallel efficiency is the serial execution time divided by the product of the parallel execution time and the number of tasks. If multiple nodes are used then in most cases one should try to use all of the CPU-cores on each node."

Note: For the scaling analysis that was performed with CMAQ, the parallel efficiency was determined as the runtime for the smallest number of CPUs divided by the product of the parallel execution time and the number of additional cpus used. If smallest NPCOLxNPROW configuration was 18 cpus, the run time for that case was used, and then the parallel efficiency for the case where 36 cpus were used would be parallel efficiency = runtime_18cpu/(runtime_36cpu*2)*100

See also:

Scaling Analysis - see section on Multinode or Parallel MPI Codes

Azure HBv3-120 Pricing

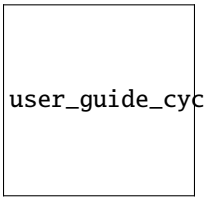


Table 1. Azure Instance On-Demand versus Spot Pricing (price is subject to change)

Instance Name	CPUs	RAM	Memory Bandwidth	Network Bandwidth	Linux On-Demand Price	Linux Spot Price
HBv3-120	120	448 GiB	350 Gbps	200 Gbps(Infiniband)	$3.6/hour	$1.4/hour

Table 2. Timing Results for CMAQv5.3.3 2 Day CONUS2 Run on Single Virtual Machine HBv120 (120 cpu per node) I/O on /shared directory

CPUs	Nodes-by-CPU	NPCOL x NPROW	Day1 Timing (sec)	Day2 Timing (sec)	Total-Time	CPU Hours/day	SBATCH exclusive	Data Imported or Copied	Equation using Spot Pricing	Spot-Cost	Equation using On Demand Pricing	On-Demand-Cost	compiler flag	i/o dir
16	1x16	4x4	10374.66	6610.67	19685.33	734	no	copied	$1.44/hr * 1 nodes * 5.468 =	7.87	3.6/hr * 1 nodes * 5.468 =	19.68	with -march=native compiler flag	shared/data
36	1x36	6x6	5102.89	4714.96	9817.85	1.36	no	copied	$1.44/hr * 1 nodes * 2.72 =	3.92	3.6/hr * 1 nodes * 2.72 =	9.79	with -march=native compiler flag	/shared/data
72	1x72	8x9	3130.73	2747.3	5878.03	815	no	copied	$1.44/hr * 1 nodes * 1.63 =	2.35	3.6/hr * 1 nodes * 1.63 =	5.87	with -march=native compiler flag	/shared/data
90	1x90	9x10	2739.38	2417.26	5156.64	715	no	copied	$1.44/hr * 1 nodes * 1.43 =	2.06	3.6/hr * 1 nodes * 1.43 =	5.15	with -march=native compiler flag	/shared/data
120	1x120	10x12	2646.52	2374.25	5020.73	6973	no	copied	$1.44/hr * 1 nodes * 1.3946 =	2.01	3.6/hr * 1 nodes * 1.39 =	5.00	with -march=native compiler flag	/shared/data

Total HBv3-120 compute cost of Running Benchmarking Suite using SPOT pricing = $1.4/hr

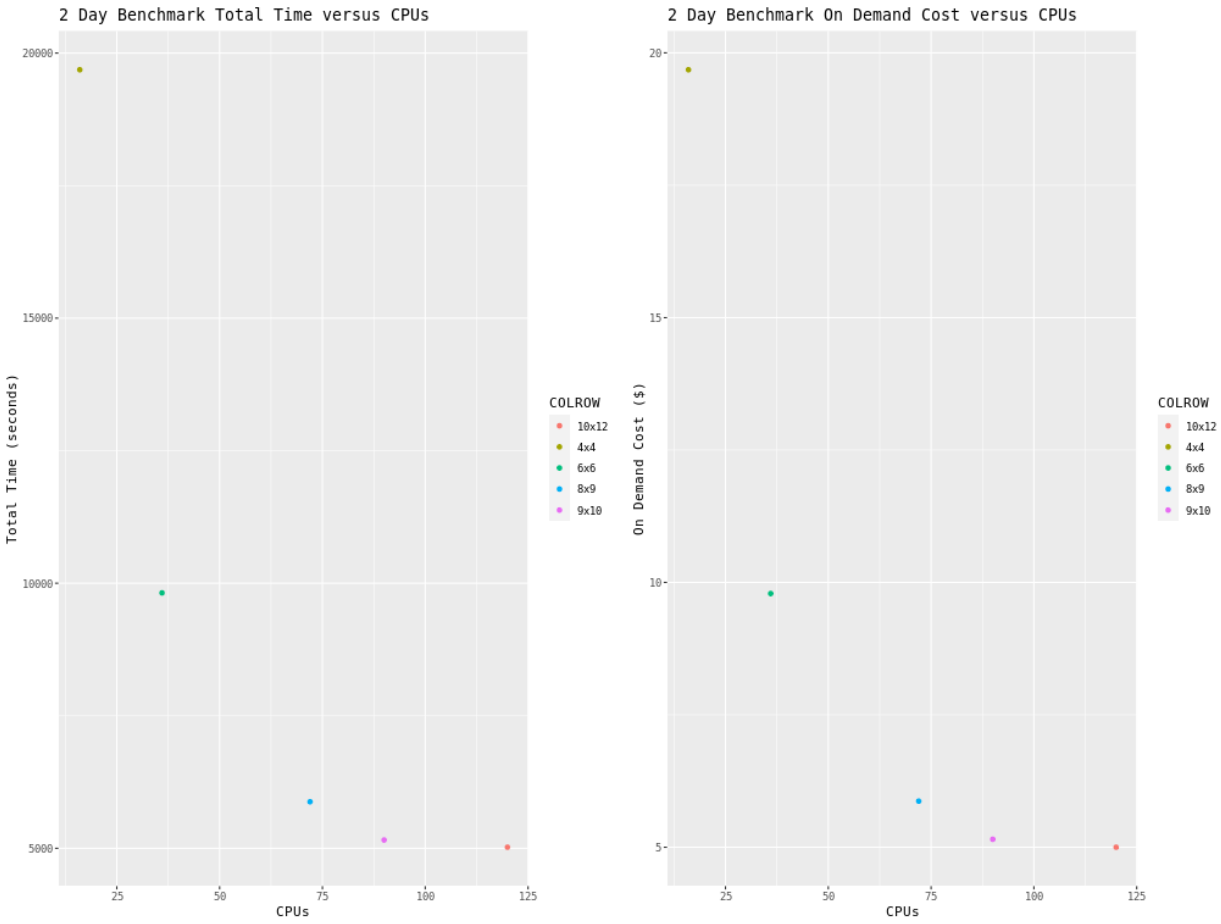
Total HBv3-120 compute cost of Running Benchmarking Suite using ONDEMAND pricing = $3.6/hr

Savings is ~ 60% for spot versus ondemand pricing for HBv3-120 compute nodes.

Azure Spot and On-Demand Pricing

3.12.3 Benchmark Scaling Plots using Single Virtual Machine HBv120

Figure 1. Plot of Time and On Demand Cost versus CPU



Performance Optimization for Cycle Cloud

3.12.4 Right-sizing Compute Nodes for the CycleCloud

Selection of the compute nodes depends on the domain size and resolution for the CMAQ case, and what your model run time requirements are. Larger hardware and memory configurations may also be required for instrumented versions of CMAQ incuding CMAQ-ISAM and CMAQ-DDM3D. The CycleCloud allows you to run the compute nodes only as long as the job requires, and you can also update the compute nodes as needed for your domain.

3.12.5 An explanation of why a scaling analysis is required for Multinode or Parallel MPI Codes

Quote from the following link.

"IMPORTANT: The optimal value of –nodes and –ntasks for a parallel code must be determined empirically by conducting a scaling analysis. As these quantities increase, the parallel efficiency tends to decrease. The parallel efficiency is the serial execution time divided by the product of the parallel execution time and the number of tasks. If multiple nodes are used then in most cases one should try to use all of the CPU-cores on each node."

Note: For the scaling analysis that was performed with CMAQ, the parallel efficiency was determined as the runtime for the smallest number of CPUs divided by the product of the parallel execution time and the number of additional cpus used. If smallest NPCOLxNPROW configuration was 18 cpus, the run time for that case was used, and then the parallel efficiency for the case where 36 cpus were used would be parallel efficiency = runtime_18cpu/(runtime_36cpu*2)*100

See also:

Scaling Analysis - see section on Multinode or Parallel MPI Codes

Example Slurm script for Multinode Runs

3.12.6 Slurm Compute Node Provisioning

Azure CycleCloud relies on SLURM to make the job allocation and scaling decisions. The jobs are launched, terminated, and resources maintained according to the Slurm instructions in the CMAQ run script. The CycleCloud Web Interface is used to set the identity of the head node and the compute node, and the maximum number of compute nodes that can be submitted to the queue.

Number of compute nodes dispatched by the slurm scheduler is specified in the run script using #SBATCH –nodes=XX #SBATCH –ntasks-per-node=YY where the maximum value of tasks per node or YY limited by many CPUs are on the compute node.

As an example:

For HC44rs, there are 44 CPUs/node, so maximum value of YY is 44 or –ntask-per-node=44. For many of the runs that were done, we set –ntask-per-node=36 so that we could compare to the c5n.9xlarge on Parallel Cluster

If running a job with 180 processors, this would require the –nodes=XX or XX to be set to 5 compute nodes, as 36x5=180.

The setting for NPCOLxNPROW must also be a maximum of 180, ie. 18 x 10 or 10 x 18 to use all of the CPUs in the Cycle Cloud HPC Node.

For HBv120, there are 120 CPUS/node, so maximum value of YY is 120 or –ntask-per-node=120.

If running a job with 240 processors, this would require the –nodes=XX or XX to be set to 2 compute nodes, as 120x2=240.

Azure HBv3-120 Pricing

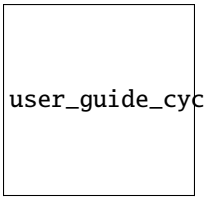


Table 1. Azure Instance On-Demand versus Spot Pricing (price is subject to change)

Instance Name	CPUs	RAM	Memory Bandwidth	Network Bandwidth	Linux On-Demand Price	Linux Spot Price
HBv3-120	120	448 GiB	350 Gbps	200 Gbps(Infiniband)	$3.6/hour	$1.4/hour

Table 2. Timing Results for CMAQv5.3.3 2 Day CONUS2 Run on Cycle Cloud with D12v2 schedulare node and HBv3-120 Compute Nodes (120 cpu per node) I/O on /shared directory

Note, two different CPUs were used,

Old CPU (logs between Feb. 16 - March 21, 2022)

```
Vendor ID:             AuthenticAMD
CPU family:            25
Model:                 1
Model name:            AMD EPYC 7V13 64-Core Processor
Stepping:              0
CPU MHz:               2445.405
BogoMIPS:              4890.81
```

New CPU (logs after March 22, 2022)

```
Vendor ID:             AuthenticAMD
CPU family:            25
Model:                 1
Model name:            AMD EPYC 7V73X 64-Core Processor
Stepping:              2
CPU MHz:               1846.530
BogoMIPS:              3693.06
```

CPUs	Nodes	Nodes-CPU	COL-ROW	Day1 Timing (sec)	Day2 Timing (sec)	Total-Time	CPU Hours/day	SBATCH-exclusive	Equation using Spot Pricing	Spot-Cost	Equation using On Demand Pricing	On-Demand-Cost	compiler flag	Input-Data	cpuMhz
90	1	1x90	9x10	3153.33	2758.13	5911.45	1.821	no	$1.4/hr * 1 nodes * 1.642 hr =	$2.29	$3.6/hr * 1 nodes * 1.642 hr =	5.911	without -march=native compiler flag	shared	2445.402
120	1	1x120	10x12	2829.82	2516.05	5345.87	1.742	no	$1.4/hr * 1 nodes * 1.484 hr =	$2.08	$3.6/hr * 1 nodes * 1.484 hr =	5.34	without -march=native compiler flag	shared	2445.400
180	2	2x90	10x18	2097.37	1809.84	3907.21	2.542	no	$1.4/hr * 2 nodes * 1.08 hr =	$3.03	$3.6/hr * 2 nodes * 1.08 hr =	7.81	with -march=native compiler flag	shared	2445.395
180	2	2x90	10 x 18	1954.20	1773.86	3728.06	2.518	no	$1.4/hr * 2 nodes * 1.036 hr =	$2.9	$3.6/hr * 2 nodes * 1.036 hr =	7.46	without -march=native compiler flag	shared	2445.405
180	5	5x36	10x18	1749.80	1571.50	3321.30	4.461	no	$1.4/hr * 5 nodes * .922 hr =	$6.46	$3.6/hr * 5 nodes * .922 hr =	16.596	without -march=native compiler flag	shared	1846.529
240	2	2x120	20x12	1856.50	1667.68	3524.18	4.895	no	$1.4/hr * 2 nodes * .97 hr =	$2.716	$3.6/hr * 2 nodes * .97 hr =	6.984	without -march=native compiler flag	shared	2445.409
270	3	3x90	15x18	1703.19	1494.17	3197.36	5.444	no	$1.4/hr * 3 nodes * .888hr =	$3.72	3.6/hr * 3 nodes * .888 =	9.59	with -march=native compiler flag	shared	2445.400
360	3	3x120	20x18	1520.29	1375.54	2895.83	7.402	no	$1.4/hr * 3 nodes * .804 =	$3.38	3.6/hr * 3 nodes * .804 =	8.687	with -march=native compiler flag	shared	2445.399
3.12.	**Performance Optimization**														**101**
360	3	3x120	20x18	1512.31	1349.54	2861.87	7.397	no	$1.4/hr * 3 nodes	$3.33	93.6/hr * 3 nodes	8.586	with -march=native com-	shared	1846.530

Total HBv3-120 compute cost of Running Benchmarking Suite using SPOT pricing = $1.4/hr

Total HBv3-120 compute cost of Running Benchmarking Suite using ONDEMAND pricing = $3.6/hr

Savings is ~ 60% for spot versus ondemand pricing for HBv3-120 compute nodes.

Azure Spot and On-Demand Pricing

Table 3. Timing Results for CMAQv5.3.3 2 Day CONUS2 Run on Cycle Cloud with D12v2 schedulare node and HBv3-120 Compute Nodes (120 cpu per node), I/O on mnt/resource/data2 directory

CPUs	Nodes	NodesxCPU	COLROW	Day1 Timing (sec)	Day2 Timing (sec)	TotalTime	CPU Hours/day	Sl
18	1	1x16	3x6	10571.20	9567.43	20138.63	2.80	no
36	1	1x36	6x6	5933.48	5230.05	11163.53	1.55	no
36	1	1x36	6x6	5841.81	5153.47	10995.28	1.52	no
96	1	1x96	12x8	3118.91	2813.86	5932	.82	?
96	1	1x96	12x8	2470.94	2845.32	5316.26	.738	?
96	1	1x96	12x8	2835.37	2474.28	5309.65	.737	ye
96	1	1x96	12x8	2683.51	2374.71	5058.22	.702	ye
120	1	1x120	10x12	2781.89	2465.87	5247.76	.729	no
120	1	1x120	10x12	3031.81	2378.64	5410.45	.751	no
120	1	1x120	10x20	2691.40	2380.51	5071.91	.704	no
120	1	1x120	12x10	3028.54	2741.83	5770.37	.801	ye
120	1	1x120	12x10	2594.57	2371.46	4966.03	.698	ye
120	1	1x120	12x10	2405.62	2166.42	4572.04	0.635	ye
192	2	2x96	16x12	2337.53	fail			
192	2	2x96	16x12	2148.09	fail			
192	2	2x96	16x12	2367.27	2276.14	4643.41	.645	ye
192	2	2x96	16x12	2419.51	2243.45	4662.96	.648	ye
192	2	2x96	16x12	1898.92	1748.17	3647.09	.5065	ye
240	2	2x120	16x15	2522.3	2172.21	4694.51	0.652	ye
240	2	2x120	16x15	1920.57	1767.07	3687.64	0.512	ye
288	3	3x96	16x18	1923.52	fail			
288	3	3x96	16x18	1967.16	1639.55	3606.71	1.00	?
288	3	3x96	16x18	2206.73	fail			
288	3	3x96	16x18	2399.31	fail			
288	3	3x96	16x18	2317.68	fail			
288	3	3x96	16x18	2253.63	2183.55	4437.18	.616	ye
288	3	3x96	16x18	1673.15	1581.15	3254.3	.452	ye
360	3	3x120	20x18	1966.37	300.73	fail		ye
360	3	3x120	20x18	1976.24	300.73	fail		ye
360	3	3x120	20x18	1950.84	294.06	fail		ye
360	3	3x120	20x18	1722.43	1630.6	3353.03	.466	ye
360	3	3x120	20x18	1404.04	1337.72	2741.76	.381	ye
384	4	4x96	24x16	1575.88	256.47	fail		
384	4	4x96	24x16	1612.54	283.36	fail		
384	4	4x96	24x16	1808.31	4.83	fail		
384	4	4x96	24x16	1043.02	258.11	fail		
384	4	4x96	24x16	1072.87	204.27	fail		
384	4	4x96	24x16	1894.96	1664.72	3559.68	.4944	ye
384	4	4x96	24x16	1631.05	1526.87	3157.92	.4386	ye
960	8	8x120	30x32	1223.52	1126.19	2349.71	.326	no
960	8	8x120	30x32	1189.21	1065.73	2254.94	.313	no

Table 4. Timing Results for CMAQv5.3.3 2 Day CONUS2 Run on Cycle Cloud with D12v2 schedulare node and HBv3-120 Compute Nodes (120 cpu per node), I/O on /lustre

CPUs	Nodes	Nodes&CPU	COL-ROW	Day1 Timing (sec)	Day2 Timing (sec)	Total-Time	CPU Hours/day	SBATCH Expensive	Equation using Spot Pricing	Spot-Cost	Equation using On Demand Pricing	On-Demand-Cost	compiler flag	Input-Data	Pin
96	1	1x96	12x8	3053.34	2753.47	5806.81	1.61	no	$.8065/hr * 1 nodes * $? =	$?	.8065/hr * 1 nodes * 3.6 =	2.90	no	shared	yes
96	1	1x96	12x8	2637.54	2282.20	4919.74	1.36	no	$.683/hr * 1 nodes * $? =	$?	.883/hr * 1 nodes * 3.6 =	2.46	no	data	yes
96	1	1x96	12x8	2507.99	2713.59	5221.58	1.45	no	$.725/hr * 1 nodes * $? =	$?	.725/hr * 1 nodes * 3.6 =	2.61	no	lustre	yes
192	2	2x96	16x12	2066.07	1938.85	4004.92	1.11	no	$.556/hr * 2 nodes * $?	$?	.556/hr * 2 nodes * 3.6 =	4.00	no	shared	yes
192	2	2x96	16x12	1608.48	1451.76	3060.24	.850	no	$.425/hr * 2 nodes * $?	$?	.425/hr * 2 nodes * 3.6 =	3.06	no	data	yes
192	2	2x96	16x12	1481.03	1350.2	2831.23	0.786	no	$.393/hr * 2 nodes * $?	$?	.393/hr * 2 nodes * 3.6 =	2.83	no	lustre	yes
288	3	3x96	16x18	1861.91	1783.59	3645.50	1.01	no	$.506/hr * 3 nodes * $? =	$?	.506/hr * 3 nodes * 3.6 =	5.46	no	shared	yes
288	3	3x96	16x18	1295.17	1182.85	2478.02	.688	no	$.344/hr * 3 nodes * $? =	$?	.344/hr * 3 nodes * 3.6 =	3.78	no	data	yes
288	3	3x96	16x18	1239.03	1127.45	2366.48	.657	no	$.328/hr * 3 nodes * $? =	$?	.328/hr * 3 nodes * 3.6 =	3.61	no	data	yes
384	4	4x96	24x16	1670.79	1595.90	3266.69	.907	no	$.454/hr * 4 nodes * $? =	$?	.453/hr * 4 nodes * 3.6 =	6.53	no	shared	yes
384	4	4x96	24x16	1095.16	1012.95	2108.11	.586	no	$.292/hr * 4 nodes * $? =	$?	.292/hr * 4 nodes * 3.6 =	4.21	no	data	yes
384	4	4x96	24x16	962.67	877.46	1840.13	.511	no	$.256/hr	$?	.256/hr	3.68	no	lus-	yes

Table 5. Timing Results for CMAQv5.3.3 2 Day CONUS2 Run on Cycle Cloud with D12v2 schedular node and HC44RS Compute Nodes (44 cpus per node)

Note, the CPU Mhz values are reported in the table below.

```
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz
Stepping:              4
CPU MHz:               2693.763
BogoMIPS:              5387.52
```

CPUs	Nodes	Nodes&CPU	COL-ROW	Day1 Timing (sec)	Day2 Timing (sec)	Total-Time	CPU Hours/day	SBATCH Exclusive	Equation using Spot Pricing	Spot-Cost	Equation using On Demand Pricing	On-Demand-Cost	compiler flag	Input-Data
18	1	1x18	3x6	13525.66	12107.02	25632.68	7.1256	no	$.3168/hr * 1 nodes * 7.12 =	$2.26	3.186/hr * 1 nodes * 7.12 =	22.68	with -march=native compiler flag	shared
36	1	1x36	6x6	7349.06	6486.37	13835.43	3.8432	no	$.3168/hr * 1 nodes * 3.84 =	$1.22	3.186/hr * 1 nodes * 3.84 =	12.23	with -march=native compiler flag	/shared
40	1	1x40	4x10	6685.74	5935.01	12620.75	3.5055	no	$.3168/hr * 1 nodes * 3.5 =	$1.11	3.168/hr * 1 nodes * 3.5 =	11	with -march=native compiler flag	/shared
72	2	2x36	8x9	4090.80	3549.60	7640.40	2.1233	no	$.3168/hr * 2 nodes * 2.12 =	$1.34	3.168/hr * 2 nodes * 2.12 =	13.4	with -march=native compiler flag	/shared
108	3	3x36	9x12	2912.59	2551.08	5463.67	1.5175	no	$.3168/hr * 3 nodes * 1.517 =	$1.44	3.168/hr * 3 nodes * 1.517 =	14.41	with -march=native compiler flag	/shared
126	7	7x18	9x14	2646.52	2374.21	5020.73	2.7836	no	$.3168/hr * 7 nodes * 1.517 =	$3.36	3.168/hr * 7 nodes * 1.517 =	33.64	with -march=native compiler flag	/shared
144	4	4x36	12x12	2449.39	2177.28	4626.67	1.854	no	$.3168/hr * 4 nodes * 1.285 =	$1.63	3.168/hr * 4 nodes * 1.285 =	16.28	with -march=native compiler flag	/shared
180	5	5x36	10x18	2077.22	1851.77	3928.99	1.5545	no	$.3168/hr * 5 nodes * 1.09 =	$1.72	3.168/hr * 5 nodes * 1.09 =	17.26	with -march=native compiler flag	/shared
216	6	6x36	18x12	1908.15	1722.07	3630.22	1.5504	no	$.3168/hr * 6 nodes * 1.01 =	$1.92	3.168/hr * 6 nodes * 1.01 =	19.16	with -march=native compiler flag	/shared
288	8	8x36	16x18	1750.36	1593.29	3343.65	1.5464	no	$.3168/hr * 8 nodes * .928 =	$2.35	3.168/hr * 8 nodes * .928 =	39.54	with -march=native compiler flag	/shared

3.12.7 Benchmark Scaling Plots using CycleCloud

Benchmark Scaling Plot for CycleCloud using HC44rs Compute Nodes

Figure 4. Scaling per Node on HC44rs Compute Nodes (44 cpu/node)

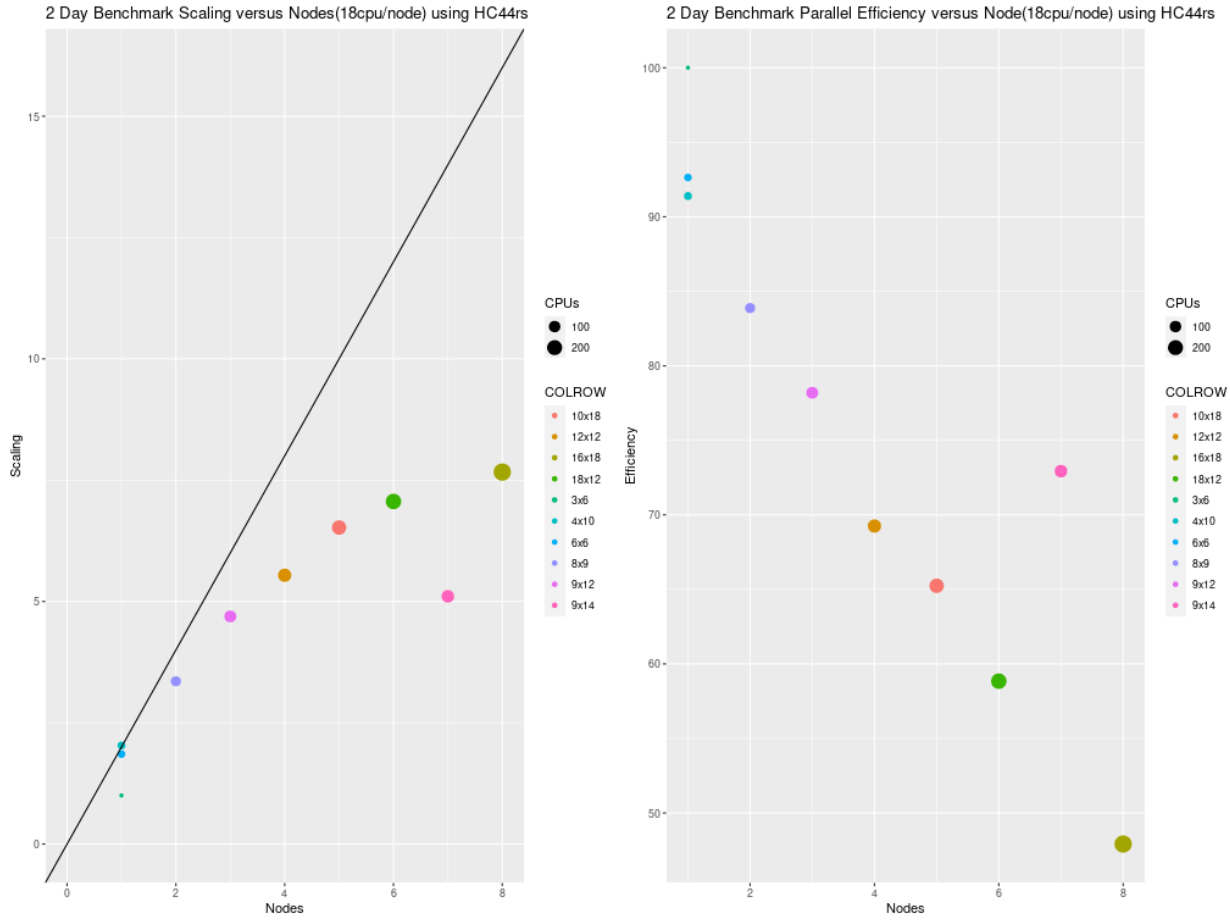


Figure 5. Scaling per CPU on HC44rs Compute Nodes (44 cpu/node)

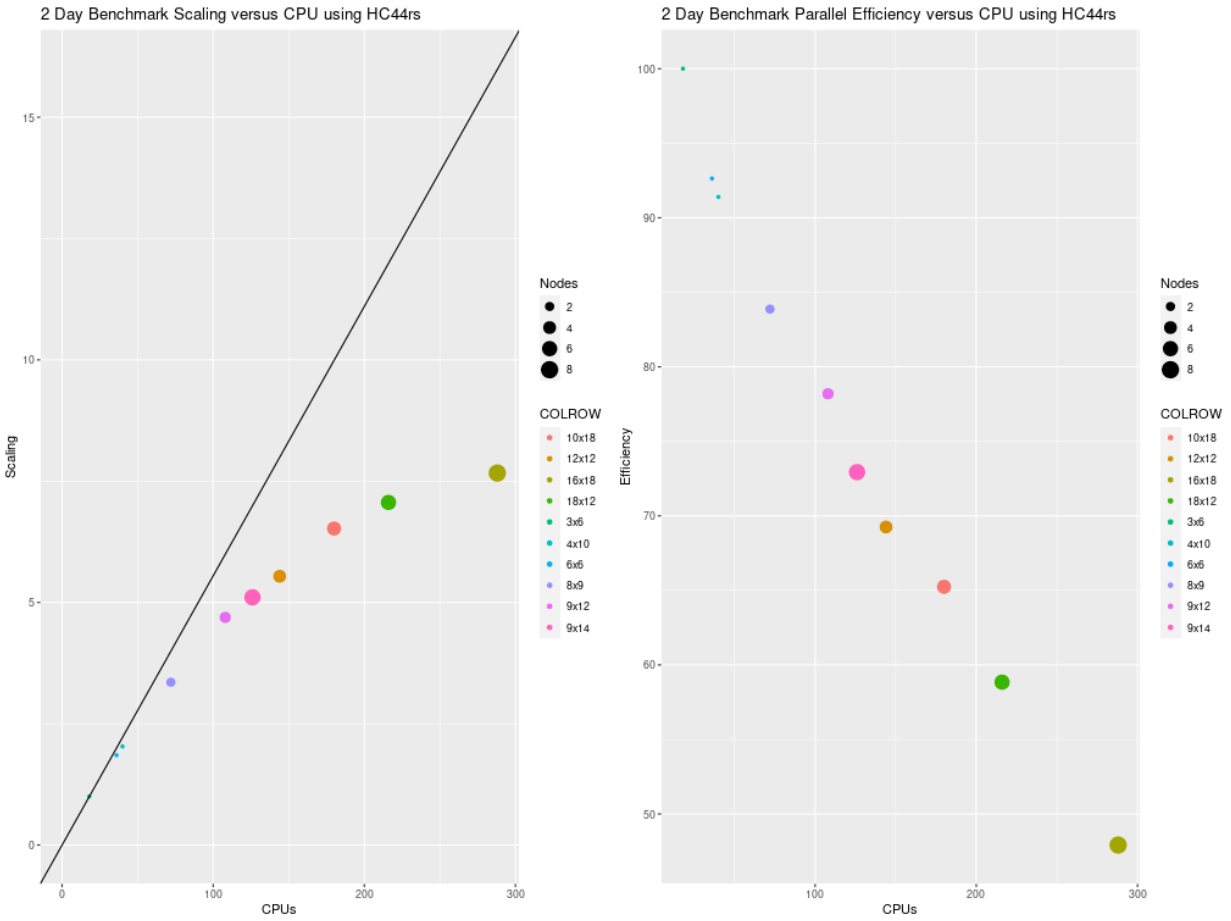


Figure 6. Scaling per Node on HBv120 Compute Nodes (120 cpu/node)

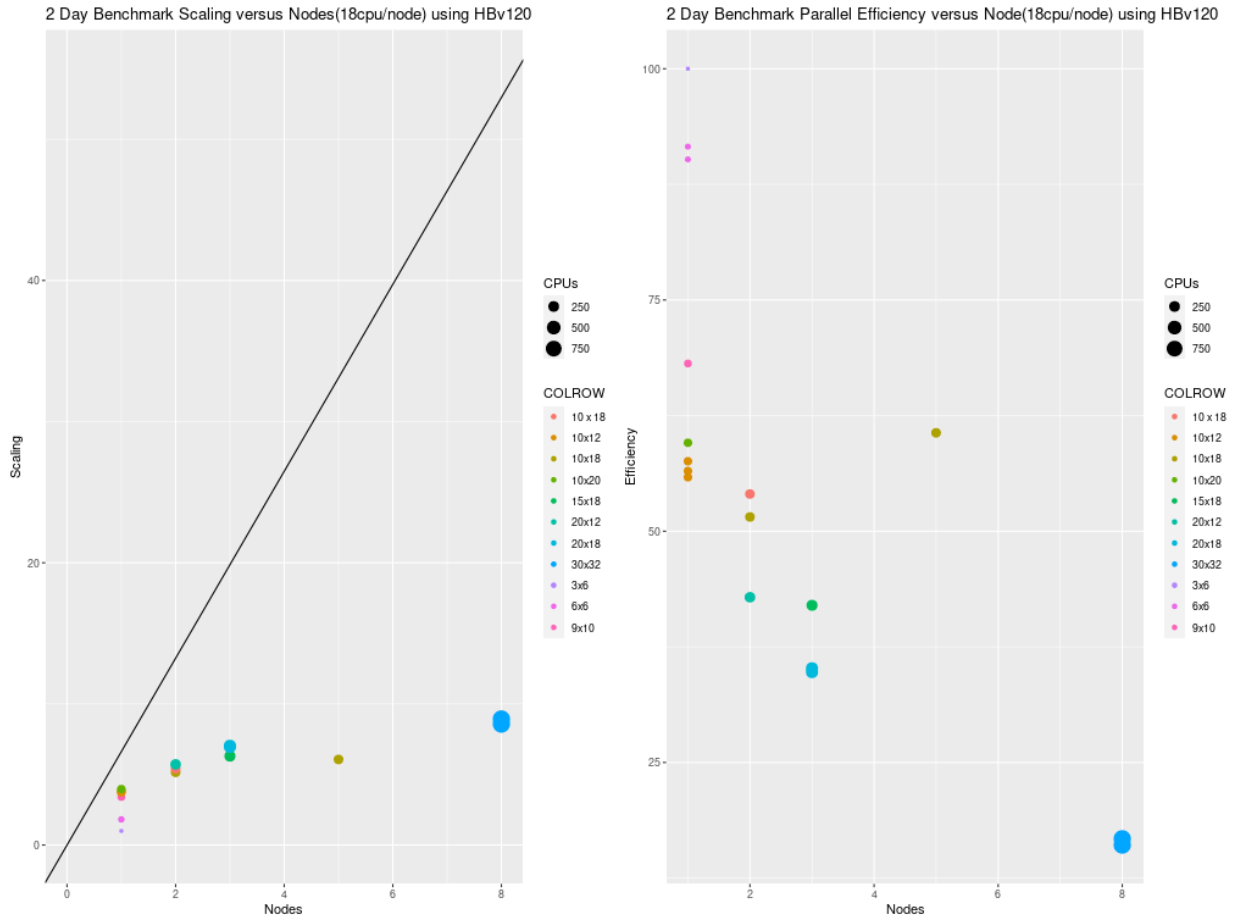


Figure 7. Scaling per CPU on HBv120 Compute Node (120 cpu/node)

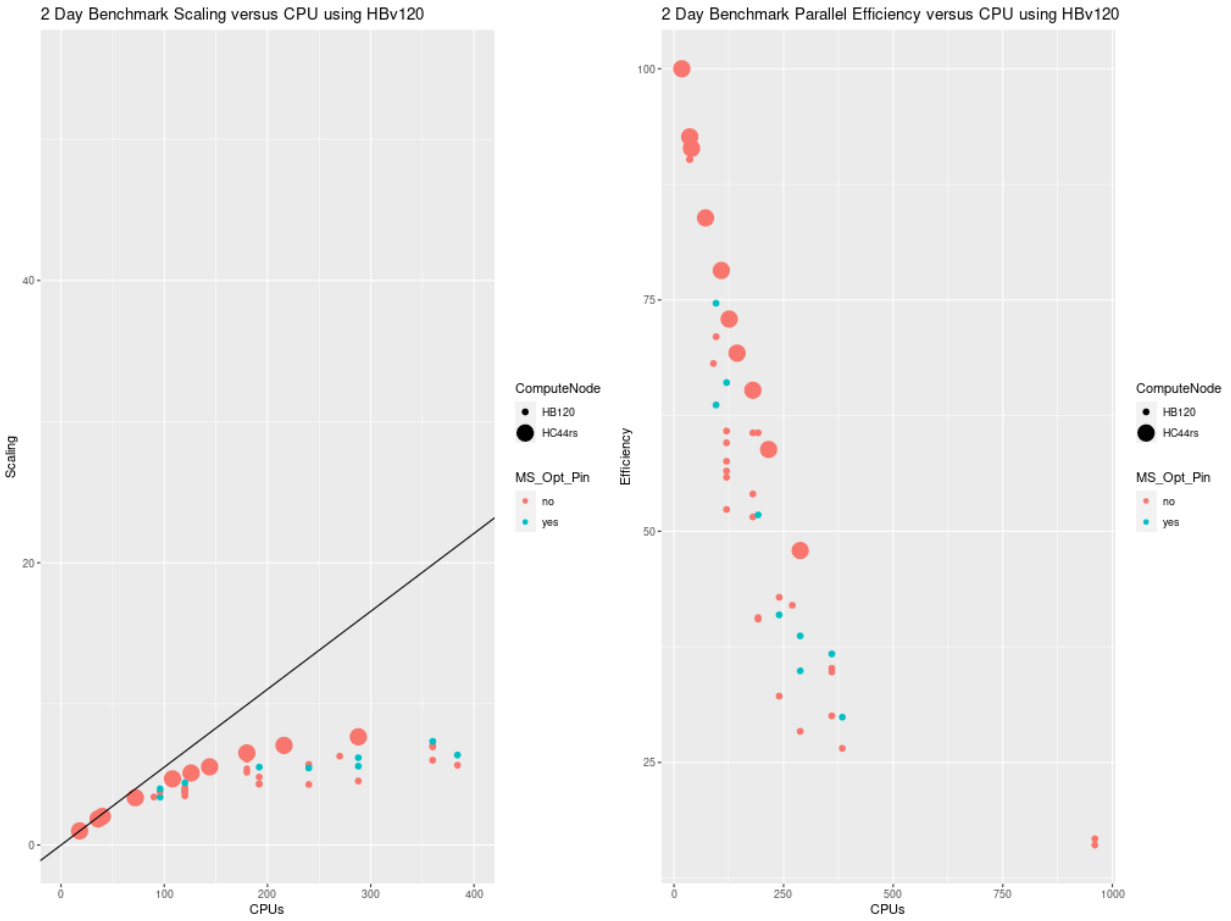


Figure 8 shows the scaling per-node, as the configurations that were run were multiples of the number of cpus per node. CMAQ was not run on a single cpu, as this would have been costly and inefficient.

Figure 9. Plot of Total Time and On Demand Cost versus CPUs for both HC44rs and HBv120

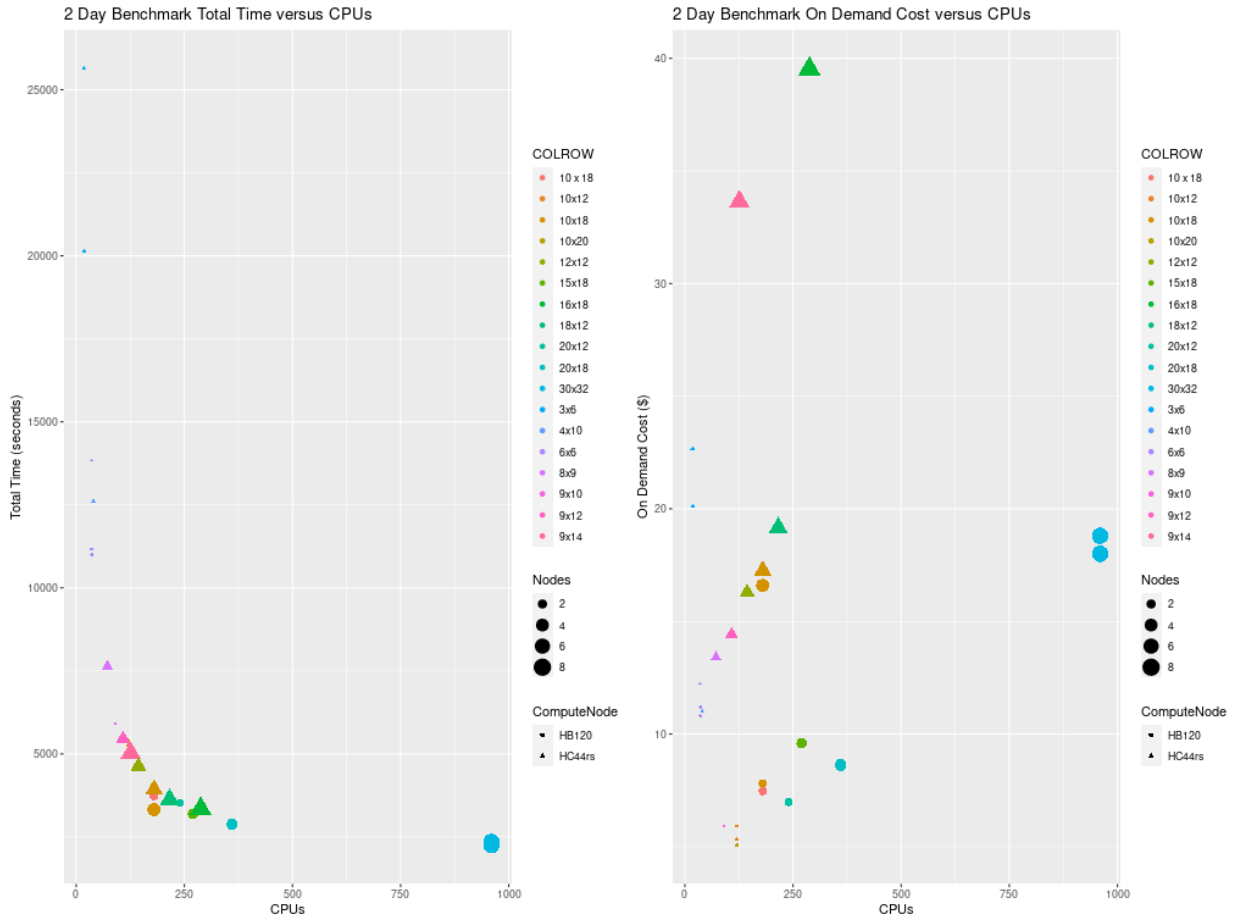
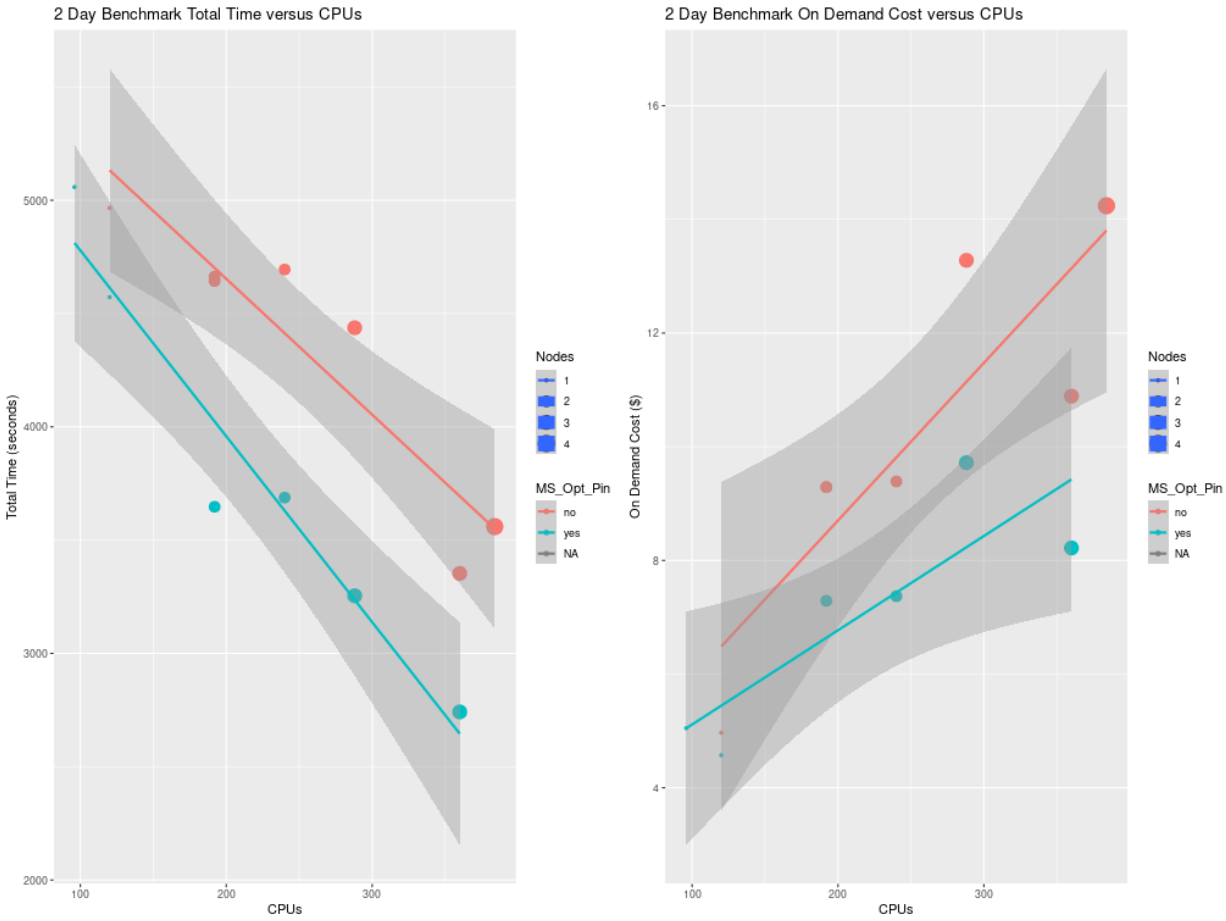


Figure 10. Plot of Total Time and On Demand Cost versus CPUs for HBv120


2 Day Benchmark Total Time versus CPUs
Total Time (seconds)
5000
4000
3000
2000
100
200
300
CPUs
Nodes
1
2
3
4
MS_Opt_Pin
no
yes
NA
2 Day Benchmark On Demand Cost versus CPUs
On Demand Cost ($)
16
12
8
4
100
200
300
CPUs
Nodes
1
2
3
4
MS_Opt_Pin
no
yes
NA

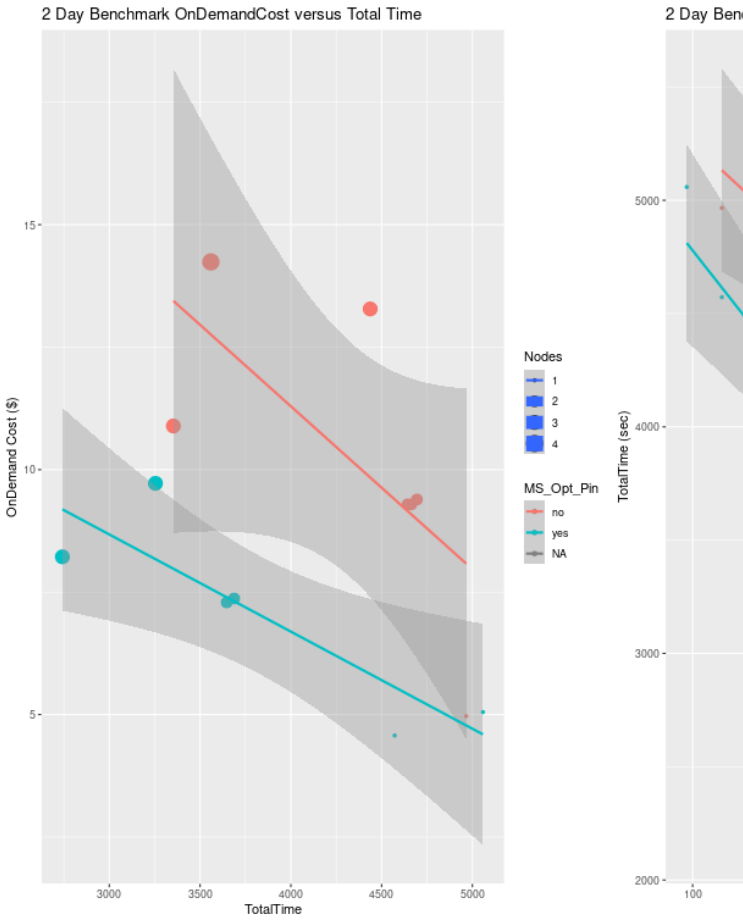


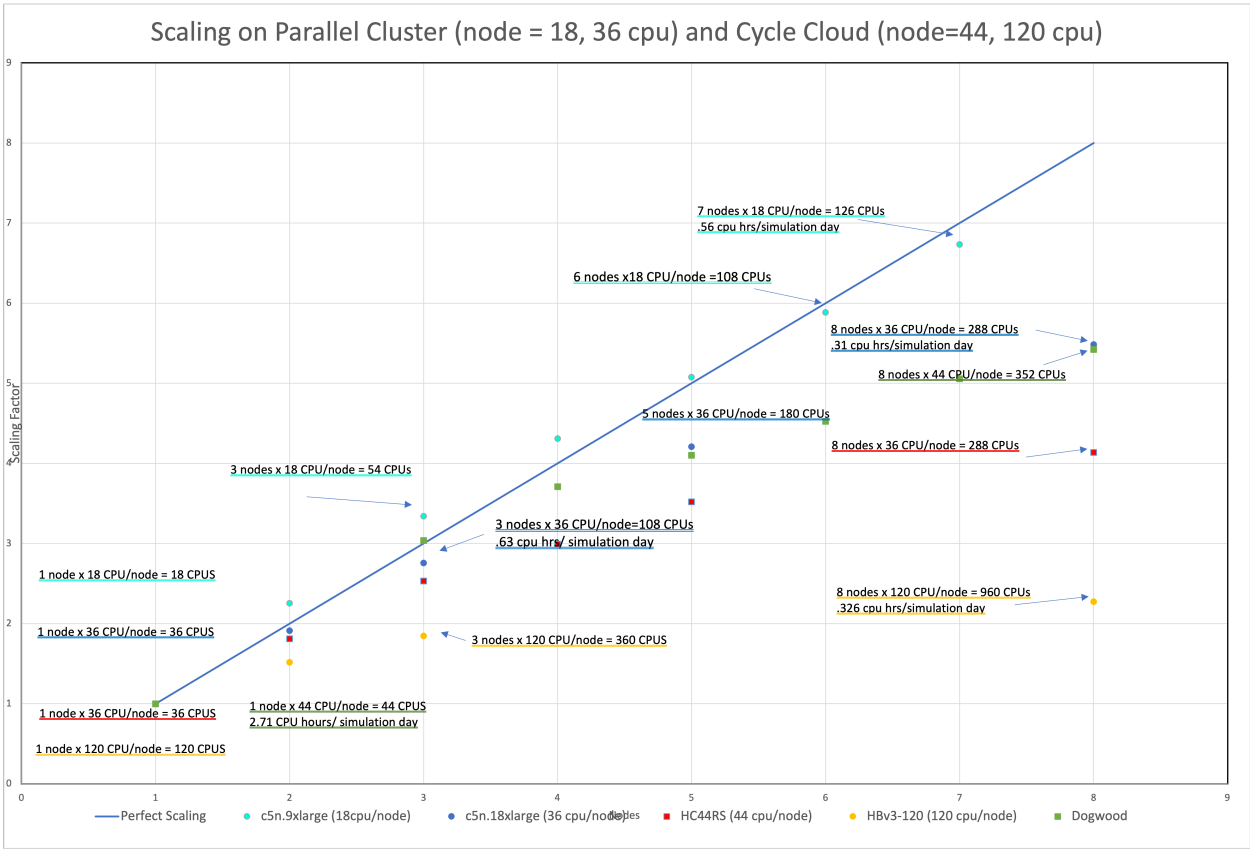
Figure 11. Plot of On Demand Cost versus Total Time for HBv120

HC44RS SPOT Pricing $.3168

HC44RS ONDEMAND pricing $3.168

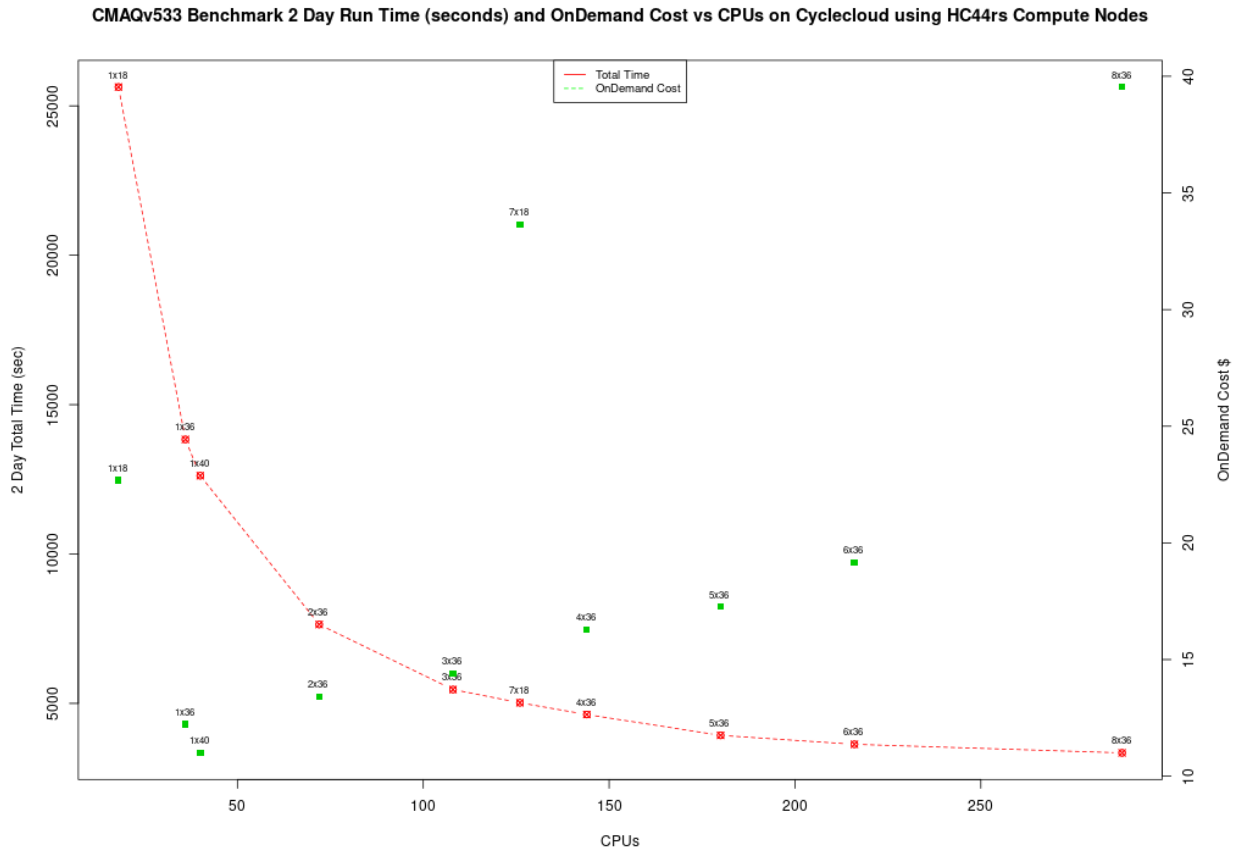
Savings is ~ 90% for spot versus ondemand pricing for HC44RS compute nodes.

Figure 11. Scaling Plot Comparison of Parallel Cluster and Cycle Cloud



Note CMAQ scales well up to ~ 200 processors for the CONUS domain. As more processors are added beyond 200 processors, the CMAQ gets less efficient at using all of them. The Cycle Cloud HC44RS performance is similar to the c5n.18xlarge using 36 cpus/node on 8 nodes, or 288 cpus. cost is $39.54 for Cycle Cloud compared to $19.46 for Parallel Cluster for the 2-Day 12US2 CONUS Benchmark.

Figure 12. Plot of Total Time and On Demand Cost versus CPUs for HC44RS.



Figures: todo - need screenshots of Azure Pricing

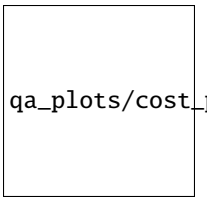
Fost by Instance Type - update for Azure

qa_plots/cost_plots/Azure_Bench_Cost.png

Figure 13. Cost by Usage Type - Azure Console

qa_plots/cost_plots/Azure_Bench_Usage_Type_Cost.png

Figure 14. Cost by Service Type - Azure Console



Scheduler node D12v2 compute cost = entire time that the CycleCloud HPC Cluster is running (creation to deletion) = 6 hours * $0.?/hr = $? using spot pricing, 6 hours * $?/hr = $? using on demand pricing.

Using 360 cpus on the Cycle Cloud Cluster, it would take ~6.11 days to run a full year, using 3 HBv3-120 compute nodes.

Table 6. Extrapolated Cost of HBv3-120 used for CMAQv5.3.3 Annual Simulation based on 2 day CONUS2 benchmark

Benchmark Case	Number of PES	Compute Nodes	Number of HBv3-120 Nodes	Pinning	Pricing	Cost per node	Time to completion (hour)	Extrapolate Cost for Annual Simulation	Annual Cost	Days to Complete Annual Simulation
2 day CONUS	360	HBv3-120	3	SPOT	No	1.4/hour	2895.83/3600 = .8044	8044/2 * 365 = 147 hours/node * 3 nodes = 441 * $1.4 =	$617.4	18.4
2 day CONUS	360	HBv3-120	3	ON-DEMAND	No	3.6/hour	2895.83/3600 = .8044	8044/2 * 365 = 147 hours/node * 3 nodes = 441 * $3.6 =	$1,587.6	18.4
2 day CONUS	96	HBv3-120	1	SPOT	Yes	1.4/hour	5221.58/3600 = 1.45	1045/2 * 365 = 264.7 hours/node * 1 node = 264.7 * $1.4 =	$370.6	11.03
2 day CONUS	96	HBv3-120	1	ON-DEMAND	Yes	3.6/hour	5221.58/3600 = 1.45	1045/2 * 365 = 264.7 hours/node * 1 node = 264.7 * $3.6 =	$952.9	11.03
2 day CONUS	192	HBv3-120	2	SPOT	Yes	1.4/hour	2831.23/3600 = .786	786/2 * 365 = 143.5 hours/node * 2 nodes = 287.1 * $1.4 =	$401.9	4.87
2 day CONUS	192	HBv3-120	2	ON-DEMAND	Yes	3.6/hour	2831.23/3600 = .786	786/2 * 365 = 143.5 hours/node * 2 nodes = 287.1 * $3.6 =	$1033.3	4.87
2 day CONUS	180	HC44RS	5	SPOT	No	.3168/hour	3928.99/3600 = 1.09	1009/2 * 365 = 190 hours/node * 5 nodes = 950 * $.3168 =	$301	39.5
2 day CONUS	180	HC44RS	5	ON-DEMAND	No	3.168/hour	3928.99/3600 = 1.09	1009/2 * 365 = 190 hours/node * 5 nodes = 950 * $3.168 =	$3,009	39.5

Azure SSD Disk Pricing Azure SSD Disk Pricing

Table 7. Shared SSD File System Pricing

Storage Type	Storage options	Max IOPS (Max IOPS w/ bursting)	Pricing (monthly)	Pricing	Price per mount per month (Shared Disk)
Persistant 1TB	200 MB/s/TB	5,000 (30,000)	$122.88/month	$6.57	

Table 8. Extrapolated Cost of File system for CMAQv5.3.3 Annual Simulation based on 2 day CONUS benchmark

Need to create table

Also need estimate for Archive Storage cost for storing an annual simulation

Recommended Workflow

Post-process monthly save output and/or post-processed outputs to archive storage at the end of each month.

Goal is to develop a reproducable workflow that does the post processing after every month, and then copies what is required to archive storage, so that only 1 month of output is stored at a time on the /shared/data scratch file system. This workflow will help with preserving the data in case the cluster or scratch file system gets pre-empted.

3.13 Additional Resources

- Additional resources for Cycle Cloud

3.13.1 Cycle Cloud Resources

Links to additional resources

1. Paper on HPC Computing on Azure using Cycle Cloud: High Performance Computing on Azure using CycleCloud
2. Link on how to run GEOS-Chem on Cloud: Geos-Chem on the Cloud
3. Paper on HPC on Cloud: Enabling High-Performance Cloud Computing for Earth Science Modeling on Over a Thousand Cores: Application to the GEOS-Chem Atmospheric Chemistry Model
4. Comparitive Benchmarking on Cloud: Comparative benchmarking of cloud computing vendors with high performance linpack
5. Information about Azure Open Dataset Program: Azure Open Datasets
6. Tutorial on Getting Started with GEOS Chem: Getting Started with GEOS Chem Tutorial
7. Git repo for Auto-scaling Slurm CycleCloud Cluster Git Repo to set up Auto-scaling Slurm CycleCloud Cluster
8. WRF, CMAQ & CAMx VM Image on Azure HPC available for fee on Azure Marketplace (not using Cycle Cloud) WRF, CMAQ & CAMx VM Image on Azure HPC (not using Cycle Cloud)

3.13.2 Help Resources for CMAQ

1. CMAS Center Forum
2. EPA CMAQ Website
3. UNC CMAS Center Website

3.13.3 Resources from Azure for diagnosing issues with running the Cycle Cloud

Issues

Please open a GitHub issue for any feedback or issues:

There is also an active community driven Q&A site that may be helpful:

All Git Repositories matching CycleCloud

CycleCloud itself is installed as an application server on a Virtual Machine (VM) in Azure that requires outbound access to Azure Resource Provider APIs. CycleCloud then starts and manages VMs that form the High Performance Computing (HPC) systems — these typically consist of the HPC scheduler head node(s) and compute nodes.

Azure CycleCloud Documentation

Create an Virtual Machine for the CycleCloud 8.2 Image and then from that VM configure a cycle cloud host instance which will create a Cycle Cloud Scheduler. Use your credentials to ssh into the scheduler.

Follow these quickstart instructions to create CycleCloud.

Quickstart CycleCloud from Marketplace VM

Set up and use Managed Identities

Set up and use Managed Identities

List of error messages encountered on Cycle Cloud

List of error message one encounters on Cycle Cloud.

3.13.4 Frequently Asked Questions

Q. How do you figure out why a job is not successfully running in the slurm queue

A. Check the logs available in the following link

Slurm on CycleCloud

`vi /var/log/slurmctld/resume.log`

If the resume step is failing, there will also be a `vi /var/log/slurmctld/resume_fail.log`

Q. Is there a tutorial on how to use SLURM?

A. Yes

Surm Tutorial

3.13.5 Computing on the Cloud References

WRF Cloud Computing Paper

NOAA Cloud-based Warn-on-Forecast

3.14 Future Work

3.14.1 List of ideas for future work

Azure Cycle Cloud

1. Test creation of Cycle Cloud on new account to verify instructions for user account permission settings and cycle cloud options to login, build and run CMAQ.
2. Figure out how to install slurm scheduler on AlmaLinux Virtual machine for HBv120. Determine if there is an overhead cost or penalty for running jobs within slurm versus interactively.
3. Figure out how to streamline install scripts and have them load while the Virtual Machine or Cycle Cloud instance is procurred using cloud-init.
4. Figure out how to save the machine image with the software installed and re-use for the creation of a new Virtual Machine.
5. Learn how to set alarms or alerts or automatic shut-down of Virtual Machine if costs exceed budget.
6. Learn how to use spot instances for running CMAQ for both a single Virtual Machine and also for the compute nodes in Cycle Cloud. I ran into difficulty using a spot node machine for the Cycle Cloud VM Application Server. (it may make it difficult to terminate the Cycle Cloud)
7. Look into the Azure Open Data Program. Is there any reason to save the input data for the benchmark there in addition to on the AWs S3 Bucket?

Documentation

1. Finalize documentation and implement a version for the documentation in github.
2. Learn how to use the Cost Explorer in Azure and create screenshots showing costs for the single Virtual Machine HBv120 benchmarks and also for the Azure Cycle Cloud.
3. Document the screenshot that shows the Cycle Cloud VM that is the Application Server. This VM has a public IP address from which you can see the Cycle Cloud and on the left column see all of the Clusters that have been created, and in the center panel, see the status of those clusters. Note, that even if all of the clusters listed in the Cycle Cloud Application Server have been terminated, you can restart them, and have access to the input data and software.
4. I think it is a best practice to leave the Application Server running, but terminate the Cycle-Cloud Clusters. I don't know what happens if you "Stop" the Application Server, Can you later restart the Application Server? It would likely use a new IP address, but would it allow you to see all of the Cycle-Cloud Clusters again?
5. Add a poll to gauge usefulness of AWS and Azure Tutorials
6. Does your group have an azure account,
7. What is your experience level with Virtual Machine on Azure
8. What is your experience level with CycleCloud on Azure
9. What is your experience with using the AWS Parallel Cluster Tutorial

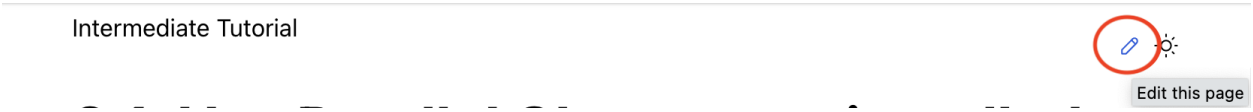
3.15 Contribute to this Tutorial

The community is encouraged to contribute to this documentation. It is open source, created by the CMAS Center, under contract to EPA, for the benefit of the CMAS Community.

3.15.1 Contribute to Azure-cmaq Documentation

Please take note of any issues and submit to Github Issue

Note: At the top of each page of the documentation, there is also an pencil icon, that you can click. It will create a fork of the project on your github account that you can make edits and then submit a pull request.



If you are able to create a pull request, please include the following in your issue:

- pull request number

If you are not able to create a pull request, please include the following in your issue:

- section number
- description of the issue encountered
- recommended fix, if available