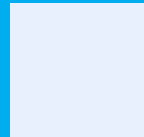




How to Provision Azure Clusters for Big Compute POCs: Step by Step

Azure Big Compute



Instructor Led lab

Azure Big Compute opportunities for AI, Deep Learning, Visualization and HPC not only have the highest potential for transformational impact but also drive the largest consumption of all Azure services. This session, delivered by C+E Big Compute Black Belts, will walk you through the steps to provision a Big Compute cluster to use for a POC.

This document supports a preliminary release of a software product that may be changed substantially prior to final commercial release. This document is provided for informational purposes only and Microsoft makes no warranties, either express or implied, in this document. Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results from the use of this document remains with the user. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Copyright 2012 © Microsoft Corporation. All rights reserved.

Microsoft Windows, Azure are trademarks of the Microsoft group of companies.

Ubuntu and Canonical are registered trademarks of Canonical Ltd. The CentOS Marks are trademarks of Red Hat, Inc. ("Red Hat").

All other trademarks are property of their respective owners.

Introduction

Estimated time to complete this lab

60 minutes

Objectives

After completing this lab, you will be able to:

- Determine which github repository to use for provisioning a Big Compute cluster on Azure
- Initialize the Cloud Shell in the Azure Portal
- Provision a cluster of Linux Virtual Machines from a Github JSON template using the Azure Portal
- Run an MPI Ping Pong test on the cluster
- Re-do the exercise using the CLI commands in the Cloud Shell using a parameter file

Time permitting --

- Install the BASH on Ubuntu on Windows Shell on the Windows 10 VM
- Install the Azure CLI 2.0 tools

Prerequisites

Before working on this lab, you must have:


- Login ID and Password for an Azure subscription. There will be pre-activated subscriptions available for all lab participants, however you are welcome to use your own subscription.
- Sufficient core quota for the VM type in the location you want to use.

Overview of the lab

Most Big Compute jobs run on Linux clusters. To manage the cloud burst scenario, the Azure clusters need to be provisioned on demand. This process is simplified by the use of JSON templates. Once the machines are provisioned, the protocol for making a remote connection is called SSH. There is no native SSH client for Windows, so in the past, it was necessary to install a 3rd party solution like putty or MobaXTerm. A new capability has recently been added to the Azure Portal called Cloud Shell. This lab will walk through the process of deploying a Big Compute cluster from an ARM Template in a github repository using the Azure Portal. After the machines are provisioned, we will run an MPI Ping Pong test to verify that the Infiniband is working properly. Then, we will repeat the deployment process using the CLI commands in the Cloud Shell. Finally, if time permits, we will step through the process of installing and configuring the Bash on Ubuntu on Windows Shell and install the CLI tools on it.

Scenario

Customers who run Big Compute jobs on premises typically will have requirements for bursting some of that workload to a public cloud. That way, they can absorb the additional workload without having to invest in new hardware. The provisioning of those machines needs to be simplified so that every end user can spin up a cluster on demand.

 Because cutting and pasting from this document into the Virtual Machine can be cumbersome, the longer phrases are contained in this text file:

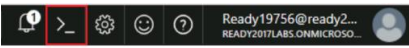
<https://techcadill300.blob.core.windows.net/docs/commands.txt>

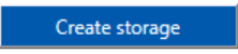
So, type that address into the browser and leave the tab open and copy the strings you need from there.

Exercise 1: Deploy the Master Node

The Azure Subscriptions for this lab have been pre-activated, so all you should need for logging in to the Azure portal should automatically show up in the upper right corner of the screen in your lab environment. Of course, you are welcome to use your own Azure Subscription if you have one.

Task 1.1 – Prepare the Cloud Shell

1. Open a browser window and go to <http://portal.azure.com>
2. Login using the credentials that were provided at the start of the lab.
3. Click on the  Cloud Shell symbol near the top right corner of the browser window.

4. Then, when prompted, click on the  button.

SSH is able to use public/private key pairs to enable password-less connections between systems. We will use this to connect to the Azure Virtual Machines later in the lab. These keys can be generated by issuing the following command.

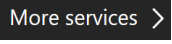



5. At the \$ prompt, type the following command, and then press ENTER:

```
↩ ssh-keygen
```

6. Press ENTER to accept the default location
7. Press ENTER twice so as not to create a passphrase
8. You can close the Cloud Shell for now by clicking on the X in the top right corner.


Task 1.2 – Verify Core Quota

Usually, before you can deploy a large number of Virtual Machines into an Azure Subscription, it is necessary to increase the core quota. It is always a good idea to verify if the quota limits for given VM type in a given region are sufficient for size of the cluster being deployed.

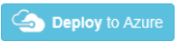
1. Select  from the bottom left corner of the browser window.
2. Then, select  Subscriptions from the pane that pops up.
3. Click on the Subscription Name  Azure Pass.
4. Then, click on  Usage + quotas in the middle of the screen. (You may have to scroll the center section down to find it).
5. If you are using a new subscription and haven't deployed anything yet, this screen will be blank. So, you will need to change the dialog box in the top right corner (you may have to scroll right to find it) from "Show only items with usage" to "Show all".

- For this lab, we will be using the A8 machines, so click on the drop down arrow beside the “All service quotas” dialog box and Un-Check the “Select all” box. Then, scroll down and check the box next to “Standard A8-A11 Family Cores”.
- We will be using the South Central US Region for this lab, so click on the drop down arrow beside the “All Locations” dialog box and Un-Check the “Select all” box. Then scroll down and check the box next to “South Central US”.

The right hand column should now be displaying the current core quota for that subscription. All of the subscriptions for this lab should already have a 16 core quota for the A8 machines in the South Central US Region.




- If the quota shown is not sufficient for the cluster you want to deploy, then you will need to request a quota increase. This can be done by clicking on the  button in the top right corner and filling in the requested info.

Task 1.3 – Launch the Deployment from the Portal


- Open a new tab in the browser and navigate to:
<https://github.com/grandparoch/azure-hpc/tree/Ready/Compute-Grid-Infra>
- Scroll down and scan through the description of what the templates do. You can click on any of the artifacts in the repository to examine the JSON templates and bash scripts, but that is not the focus of this lab, so we will not dwell on them.
- Click on the first  button that you encounter about half-way down the screen. This will launch the main deployment template through the Azure management portal.
- The Custom deployment pane presents dialog boxes for capturing all of the input parameters.
- For the Resource Group, leave the “Create new” radio button selected and write “ready” (or any other name you like) in the box below it.
- Make sure that the location is set to “South Central US”.
- The JSON template specifies the allowed values and default values for each of the parameters. For this lab, leave the defaults as shown:

V Msku	Standard_DS2_v2
Master Image	CentOS_7.3
Shared Storage	nfsonmaster
Data Disk Size	P10
Nb Data Disks	4
Monitoring	ganglia
Scheduler	pbspro
VM Prefix	ready

Azure Big Compute: Step by Step

- For the “Admin User Name”, it could be any name you want, but for this lab we suggest that you use the same ID as the one which is created in the Cloud Shell. (the name to the left of the “@”). That way, you will not have to specify a different user name when you connect to the machines with SSH from the Cloud Shell.
- For the “Admin Password”, enter any 12 character, complex password – the password provided with the Azure subscription will not be long or complex enough.
- You can leave the SSH Key Data dialog box blank.
- Scroll down until you see the line that says, “I agree to the terms and conditions stated above” and check the box next to it.
- Then click on the  button at the bottom. This launches the deployment.
- To monitor the progress, select  Resource groups from the menu on the left.
- Click on the new Resource Group name - “ready” (or the name you used) and notice all the resources which have already been deployed.
- Click on the  Deployments link in the middle of the menu on the left side of the window. This will show the deployment status for each of the templates and how long each of them took to complete. When you click on the name of any of them, you can observe the status of each of the resources. Close that window (you may have to scroll right to find the X in top right corner).

Task 1.4 – Login to the Master node

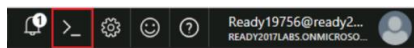
- Scroll left to the main Resource Group Overview pane and click on  readymaster
- Find the Public IP address in the middle of the right column. If you hover over the Public IP Address, a clipboard icon will appear on the right side.

Public IP address/DNS name label
137.117.15.28/<none>



When it does, click on the icon to copy the address to the clipboard.

- If the Cloud Shell is not still open in another browser tab, then launch it again by clicking on the



icon.

- At the prompt, type “ssh “ and then right-click to paste the address that was copied from the portal then press “Enter”.
- Type “yes” in response to the question about wanting to continue. When prompted, type the password that you entered at the parameters screen. The command prompt should change to “[readyusr19755@readymaster ~]\$”
- To monitor the progress of the configuration script, you may enter the following commands:

```
↩ sudo su - root
↩ <retype the password>
```

```
↪ cd /var/lib/waagent/custom-script/download
↪ ls -al
```

7. You should see directories named 0, 1, 2, 3, etc. – one for each of the template scripts which have been downloaded and run as part of the provisioning process. In each of those directories, you will see the script file and files named “stderr” and “stdout” which capture the console activity during the execution of the scripts. Pick the largest number and examine the contents.

```
↪ cd 3
↪ tail -f stdout
```

8. If the screen is not scrolling, then the script could be finished. To stop the “tail” command, type:

```
↪ ^C (Ctrl-C)
```

9. Display the stderr and stdout files with the “cat” command.

```
↪ cat stderr | more      (hit the space bar to advance the screen)
↪ cat stdout | more
```

10. To return to root’s home directory, type

```
↪ cd ~
```

11. Remember we specified 4 – P10 attached disks. (the P10 is 128GB). The install_nfs.sh script should have formatted the attached disks and created a RAID 0 (striped) array which should total 512GB.

Verify that the attached disks were formatted and striped properly for NFS by typing:










```
↪ df -h
```


You should see this row in the resulting table:

/dev/md10	512G	33M	512G	1%	/data
-----------	------	-----	------	----	-------

If it does not display this row, then it may not be finished running the NFS Setup script yet. You can check the status by returning to the Deployments screen in the portal. (Steps 13-15 in Task 1.3) If any of the icons are still blue, then it is not finished yet.

Azure Big Compute: Step by Step


storage	 Deploying
scheduler	 Succeeded
monitoring	 Succeeded
masterExtension	 Succeeded
master-OS-resources	 Deploying
master-shared-resources	 Succeeded
master-nsg	 Succeeded
shared-resources	 Succeeded
Microsoft.Template	 Deploying

You can hit the  Refresh button at the top of that window until they all turn green. Then, try the “df -h” command again to verify the presence of the row shown above.

12. To check if PBSPro is installed properly, run the command:

```
↩ pbsnodes -a
```


This will not return any nodes, but it should run without an error. You should stay logged in to the Master Node. We will come back to it after the Compute Nodes are provisioned.

13. One of the other parameters we selected during the deployment was “ganglia” for monitoring. Ganglia is a very popular tool which is used for monitoring performance metrics across a cluster of machines. Return to the Resource Group view in the Azure Portal. Click on the  Deployments link in the middle of the menu on the left side of the window.
14. Click on [Microsoft.Template](#) and copy the GANGLIAURI to the clipboard by clicking on the icon to the right of the address.
15. Open a new tab in the browser and paste in the address from the clipboard. This will show you the Ganglia charts. Notice in the table on the left that there is only 1 Host with a total of 2 cores being monitored, since the DS2_v2 is the only machine we have provisioned so far.

Exercise 2: Deploy the Compute Nodes

The compute nodes for this cluster are implemented as multiple instances of a single VM Scale Set. They are provisioned in a manner similar to the Master Node.

Task 2.1 – Launch the Deployment from the Portal


1. Navigate the browser back to the github repository:
<https://github.com/grandparoch/azure-hpc/tree/Ready/Compute-Grid-Infra>
2. This time, scroll past the first two occurrences of the “Deploy to Azure” button. The middle section it optional. It will provision a distributed BeeGFS file system. We will not be doing that for this lab, but it is supported by the templates.
3. Click on the  button after the “Provision the Compute Nodes” section. This will launch the Compute Nodes deployment template through the Azure management portal.
4. The Custom deployment pane presents dialog boxes for capturing all of the input parameters.
5. For the Resource Group, click on the “Use existing” radio button click on the drop down arrow in the dialog box to select the same Resource Group you specified for the Master node deployment (“ready” for this lab).
6. Make sure that the location is set to “South Central US”.
7. The JSON template specifies the allowed values and default values for each of the parameters. For this lab, leave the defaults as shown:

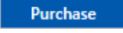


V Msku	Standard_A8
Shared Storage	nfsonmaster
Scheduler	pbspro
Monitoring	ganglia
Compute Node Image	CentOS-HPC_7.3
Instance Count per VMSS	2
Number of VMSS	1
Authentication Type	password
Master Name	readymaster
Post Install Command	hostname

8. For the “RGvnetName”, you should use “ready” (or whatever name you used when you deployed the master node). The VNet was created in the same Resource Group as the master node. Even though for this lab, we chose to use the same Resource Group for the compute nodes, we could have chosen a different one. So, this field is required for the template to know what Resource Group contains the VNet.
9. For the “Admin User Name”, you should use the same one you did for the Master node.



Azure Big Compute: Step by Step

10. For the “Admin Password”, use the same one that you did for the Master node.
11. You can leave the “Ssh Key Data” dialog box blank.
12. The “Post Install Command” should have the default “hostname” already in the box. This field will enable the machine to run a command after it is provisioned.

 **Note:** Please do not erase it. There is a bug which will prevent any of the VM Extension scripts from running if that field is blank.


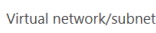
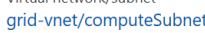
13. Scroll down until you see the line that says, “I agree to the terms and conditions stated above” and check the box next to it.
14. Then click on the  button at the bottom. This launches the deployment.
15. To monitor the progress, select  Resource groups from the menu on the left.
16. Click on the Resource Group name - “ready” (or the name you used) and click the  button until the name of the VMSS shows up on the list. It will be named <VMSS Prefix>00.

17. Click on the VMSS name  ready00

18. Then, click on  Instances from the menu on the left and wait for the Status to show  Running .

Next, we want to find the IP addresses of the Compute Nodes so that we will be able to connect to them from the Master Node. There are methods for basically crawling across a subnet to discover all of the machines that are attached, and there is a script which will do this located here:

<https://github.com/tanewill/AHOD-HPC/blob/master/hn-setup.sh> , but for just a couple of nodes, we will use the portal. Unfortunately, the instance view of the Compute Nodes does not include the IP address. So,

if you click on the  Overview at the top of the menu on the left, you will see  Virtual network/subnet  grid-vnet/computeSubnet on the right side. When you click on that, you will see the IP Addresses of these instances. In our case, they will be 10.0.0.4, and 10.0.0.5.


- ✦ Also note that the instance names are not the same as the machine names for these Compute Nodes. We’ll discover those by logging in to the machines.

Task 2.2 – Login to the Master node to test the compute nodes

1. If you are not still logged in to the Master Node, return to the Cloud Shell and log in again.

The ARM templates and deployment scripts created a user ID on all the machines named “hpcuser”. The home directory for the “hpcuser” is on the NFS share so that all the machines have the same home directory. The deployment script also executed the ssh-keygen command for the “hpcuser”, so all the machines share the same keys. That way, it is possible to perform SSH connections between all the machines without requiring a password.

2. Change to the “hpcuser” user by typing:

```
 sudo su - hpcuser
```

Azure Big Compute: Step by Step

You may be prompted to enter the password again. After you do, you should see the prompt change to: [hpcuser@readymaster readyusr19755]\$

3. Connect to the first Compute Node by typing:

```
↪ ssh 10.0.0.4
```

You should see a warning message about that address being permanently add, and the prompt should change to something like this: [hpcuser@ready00cw000000 ~]\$

✦ Note that the machine name is made up of the prefix you specified in the input parameters, plus 00 which the template inserted to allow for more than one VMSS's with the same prefix, then a 3digit code, and then 5 digits representing the instance number. That 3 digit code will be unique to the deployment. This naming scheme is described in this article: <https://msfstack.wordpress.com/2017/05/10/figuring-out-azure-vm-scale-set-machine-names/>

4. Connect to the other Compute Node by typing:

```
↪ ssh 10.0.0.5
```

After the same warning message, the prompt should change to something like this: [hpcuser@ready00cw000001 ~]\$

5. Use the “df” command to verify that the nodes are able to mount the NFS files

```
↪ df -h
```

You should see these two rows near the bottom of the list:

```
readymaster:/share/home 30G 2.4G 28G 8% /share/home
readymaster:/data 512G 33M 512G 1% /data
```

6. Disconnect from the second Compute Node by typing:

```
↪ exit
```

You should see the prompt change back to: [hpcuser@ready00cw000000 ~]\$

7. Now, if you revisit the Ganglia web site (steps 14-15 of Task 1.4), you will see that it now shows 18 CPUs Total, and 3 Hosts up.

Exercise 3: Run the MPI Ping Pong Test

Now that we have verified that the cluster has been configured properly, the next step is to test the nodes to make sure that they are leveraging the Infiniband. This is done with an MPI Ping Pong test. Sometimes Big Compute jobs do not use a job scheduler, but are run with a simple script on one of the compute nodes. Other times, the jobs are submitted to a job scheduler like PBSpro. We will use both approaches to run the MPI Ping Pong test.

Task 3.1 – Run the test with a script on one of the Compute Nodes.

1. You should still be logged in to one of the Compute Nodes as the “hpcuser”. You will need to create a file with the list of IP addresses of the Compute Nodes you want to test.

Create this list by typing:

```
↪ echo 10.0.0.4 10.0.0.5 > nodeips.txt
```

2. Retrieve the script from github with a “wget” command:

```
↪ wget https://raw.githubusercontent.com/tanewill/AHOD-HPC/master/full-pingpong.sh
```

3. Add “execution” ability to the script:

```
↪ chmod +x full-pingpong.sh
```

4. Run the script:

```
↪ ./full-pingpong.sh
```

This script runs through every combination of nodes in the list and would be useful in determining if one of the nodes was not performing as well as the others. The script filters the output and only displays the latency number for each of the combinations. For the instances where a node is communicating with itself, the latency is expected to be a very small number because the MPI traffic is handled in RAM and never leaves the machine.

- ✎ If the Infiniband is working properly, then the other numbers should be slightly more or less than 3.0. If those numbers are closer to 30, then we know that the Infiniband is not working, so something must not be configured properly.

Task 3.2 – Run the test by submitting a job to PBSpro

1. Exit from the Compute Node by typing:

```
↪ exit
```

You should see the prompt change back to: [hpcuser@readymaster ~]\$

2. Make sure you are at the home directory of the “hpcuser” by typing:

```
↪ cd ~
```

Azure Big Compute: Step by Step

3. Now, run the same PBSPro command that was run after just the Master Node had been provisioned:

```
↪ pbsnodes -a
```

This time, it will list the attributes of the two Compute Nodes.

4. PBSPro requires an input script which specifies the parameters for the job. Retrieve the script from github with a “wget” command:

```
↪ wget https://raw.githubusercontent.com/grandparoach/azure-hpc/Ready/Compute-Grid-Infra/apps/pallas/pingpong.sh
```

5. Then submit the job to PBSPro with this command:

```
↪ qsub pingpong.sh
```

It should respond with the message “0.readymaster”. It may respond with a “no such file or directory” error, but just repeat the qsub command.

6. To view the results, look at the output file:

```
↪ cat pallas-pingpong.o0 | more
```


The numbers in the 3rd column should be comparable to those from the other test.

Congratulations! You have successfully completed the main portion of this lab

Clean Up

Even if you still have time to work on the remainder of the lab, you will still need to remove the Resource Group and start fresh, because the lab subscriptions do not have sufficient core quotas to provision more than one cluster at a time.

The entire cluster may be removed by simply deleting the Resource Group.

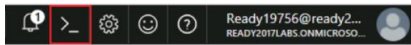
1. In the portal, go to the Resource Group view and click on the  Delete icon at the top.
2. It will then ask you to verify by typing the Resource Group name and then clicking “Delete” at the bottom.

Exercise 4: Repeat the entire process using the CLI tools in the Cloud Shell (Time permitting)

This exercise will familiarize you with the process of deploying a JSON template using the azure CLI tools in the Cloud Shell in the Azure Portal. This time, we will use the SSH keys rather than a Password to authenticate.

Task 4.1 Authorize the Azure CLI tools to manage your Azure Subscription

1. If the Cloud Shell is not still open in another browser tab, then launch it again by clicking on the



icon in the Azure portal.

2. You will need to authorize the CLI tools to manage your Azure subscription.

At the \$ prompt in the BASH shell, type the following command, and then press ENTER:

```
↩ az login
```

Then open a web browser and navigate to <https://aka.ms/devicelogin> then copy the code from the Linux command line (highlight it, then right-click) and paste it into the browser window. Follow the normal authentication process as if you were logging in to the Azure management portal using the credentials that you used when you logged into the portal. If it shows that you are already logged in, then just click on that box.

Task 4.2 Create the parameters files for the master deployment

1. To reduce the typing, we have posted the parameters file to github. You can download it directly to your BASH shell by typing the following command:

```
↩ wget https://raw.githubusercontent.com/grandparoach/azure-hpc/Ready/Compute-Grid-Infra/Samples/4nodes-centos71-pbs-IB/master.param.json
```

2. You can verify that it downloaded correctly by typing this command:

```
↩ cat master.param.json
```

The output should look like this:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "adminUsername": { "value": "" },
    "authenticationType": { "value": "sshPublicKey" },
    "sshKeyData": { "value": "" }
  }
}
```

3. Copy the SSH public key (which we generated in Task 1.1) to the clipboard.

```
↪ cd .ssh
↪ cat id_rsa.pub
```

The key will look something like this:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDQ80lphpjSN254BECXW1bU/A/CRRbJdwZwh604BYfhuc7rJU7oCqk93fdtR+b/Ay9Fc
EPnuwOyI6VXu/mZrnV3YKFRiafNgB4IEJaXF5uYzIkynIbWVi0o8+t5s4lrXt3Xd80o7R06No0RXJg19kSr7m1qUomAUpcK51f9tAnpo1
1eyL1uzJ2mhnu0QteeQ360gsSJF27wTqRfpi5kPx4dno1L2Bvii7eLFHhBnaZ+nnQIvphWME3vXfDdIQ3di1VUcn+JxFvq4Qgj1hTxeYu
xjqQoVauzdBvEpJH3DU504anxJ+zXw16i/ds603y6bMZrBxr15xmmB5SiWwOPaOBL sroach@MININT-VCLH4DM
```

Highlight the entire key by dragging with the mouse, then copy it to the clipboard with a single right mouse click.

4. Insert the Admin User Name and SSH key into the param.json file.

```
↪ vi master.param.json
```

Move the cursor to the 2nd quote mark in the value field of the “adminUsername” parameter as shown below by using the keyboard arrow keys (just clicking with the mouse will not work since the vi editor is character based).

```
"parameters": {
  "adminUsername": { "value": "O" },
  "authenticationType": { "value": "sshPublicKey" },
  "sshKeyData": { "value": "O" }
}
```

5. In the vi editor, switch to insert mode by pressing the “i” key.
6. Type the Admin User Name. (The same name that you used before – the name to the left of the @ in the Cloud Shell prompt).
7. Exit the Insert mode by hitting the “Esc” key
8. Move the cursor to the 2nd quote mark in the value field of the sshKeyData parameter.
9. Switch to insert mode by pressing the “i” key.
10. Paste the SSH key from the clipboard into the vi window with a Right-Click
11. Exit the Insert mode by hitting the “Esc” key.
12. Save the file by entering the following command, and then press ENTER:

```
↪ :wq
```

Task 4.3 Create the parameters files for the nodes deployment

13. To reduce the typing, we have posted the parameters file to github. You can download it directly to your BASH shell by typing the following command:

```
↪ wget https://raw.githubusercontent.com/grandparaoach/azure-hpc/Ready/Compute-Grid-Infra/Samples/4nodes-centos71-pbs-IB/nodes.param.json
```

14. You can verify that it downloaded correctly by typing this command:

```
↪ cat nodes.param.json
```

The output should look like this:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "adminUsername": { "value": "" },
    "authenticationType": { "value": "sshPublicKey" },
    "sshKeyData": { "value": "" },
    "RGvnetName": { "value": "ready1" }
  }
}
```

15. Copy the SSH public key (which we generated in Task 1.1) to the clipboard.

```
↪ cd .ssh
↪ cat id_rsa.pub
```

The key will look something like this:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDAQ801phpjSN254BECXW1bU/A/CRRbJdwZwh604BYFhuc7rJU7oCqk93fdtR+b/Ay9Fc
EPnuwOyI6VXu/mZrnV3YKFRiafNgB4IEJaXF5uYzIkynIbWVi0o8+t5s4lrXt3Xd80o7R06No0RXJg19kSr7m1qUomAUck51f9tAnpo1
1eyL1uzJ2mhnu0QteeQ360gsSJF27wTqRfpI5kPx4dno1L2Bvii7eLFHhBnaZ+nnQIvphWME3vXfDdIQ3di1VUcn+JxFvq4Qgj1hTxeYU
xjqQoVauzdBvEpJH3DUso4anxJ+zXw16i/ds603y6bMZrBxr15xmmB5SiWwOPaOBL sroach@MININT-VCLH4DM
```

Highlight the entire key by dragging with the mouse, then copy it to the clipboard with a single right mouse click.

16. Insert the Admin User Name and SSH key into the param.json file.

```
↪ vi nodes.param.json
```

Move the cursor to the 2nd quote mark in the value field of the “adminUsername” parameter as shown below by using the keyboard arrow keys (just clicking with the mouse will not work since the vi editor is character based).

```
"parameters": {
  "adminUsername": { "value": "O" },
  "authenticationType": { "value": "sshPublicKey" },
  "sshKeyData": { "value": "O" }
}
```

17. In the vi editor, switch to insert mode by pressing the “i” key.
18. Type the Admin User Name. (The same name that you used before – the name to the left of the @ in the Cloud Shell prompt).

19. Exit the Insert mode by hitting the “Esc” key
20. Move the cursor to the 2nd quote mark in the value field of the sshKeyData parameter.
21. Switch to insert mode by pressing the “i” key.
22. Paste the SSH key from the clipboard into the vi window with a Right-Click
23. Exit the Insert mode by hitting the “Esc” key.
24. Save the file by entering the following command, and then press ENTER:

```
↩ :wq
```

Task 4.4 Create a BASH script to launch the deployment

25. The deployment script is also available for downloading from github. Use this command to retrieve it:

```
↩ wget https://raw.githubusercontent.com/grandparoach/azure-hpc/Ready/Compute-Grid-Infra/deploycluster.sh
↩ cat deploycluster.sh
```

The result should display this:

```
az group create -n ready1 -l southcentralus

az group deployment create -g ready1 -n deploymaster --template-uri
https://raw.githubusercontent.com/grandparoach/azure-hpc/Ready/Compute-Grid-Infra/deploy-
master.json --parameters @master.param.json

az group deployment create -g ready1 -n deploynodes --template-uri
https://raw.githubusercontent.com/grandparoach/azure-hpc/Ready/Compute-Grid-Infra/deploy-
nodes.json --parameters @nodes.param.json
```

- ✎ You will notice that this script performs 3 separate operations. First it creates the Resource Group, then it invokes the “deploy-master.json” template, and when that operation is completely finished, it will then launch the deploy-nodes.json template.

26. Make the script executable by entering the following command and then press ENTER:

```
↩ chmod +x deploycluster.sh
```

27. Finally, launch the deployment by entering the following command and then press ENTER:

```
↩ ./deploycluster.sh
```

- ✎ You will notice that the Resource Group being created for this deployment has a different name than the one we used for the first 2 exercises. That is because the old one may not be completely deleted, and we don’t want to wait while it finishes, so we simply create a new one with a different name.

Also, the cli deployment is not verbose by default, so there are no status updates until the operation succeeds or fails. So, to monitor the progress, you can repeat the same steps we did in steps 13-15 of Task 1.3 and Task 1.4.

Azure Big Compute: Step by Step

- ✎ This time, when you log in to the master node, you should not have to enter a password, because it should use the SSH keys to authenticate.

Task 4.4 Clean Up

The entire cluster may be removed by simply deleting the Resource Group.

```
↪ az group delete -n ready1
```

Extra Credit: Windows Subsystem for Linux (Bash on Windows)

This exercise will familiarize you with the new BASH shell on Windows.

Enabling the Windows Subsystem for Linux

The Subsystem for Linux is not enabled by default. A good description of the process can be found at this website https://msdn.microsoft.com/en-us/commandline/wsl/install_guide

Installing the Azure CLI 20.0 tools

Besides the portal and PowerShell, Azure has a rich management toolset which runs on Linux, MAC, and Windows. By installing these tools on the BASH shell, we will be able to incorporate Azure management functions into BASH scripts.

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>