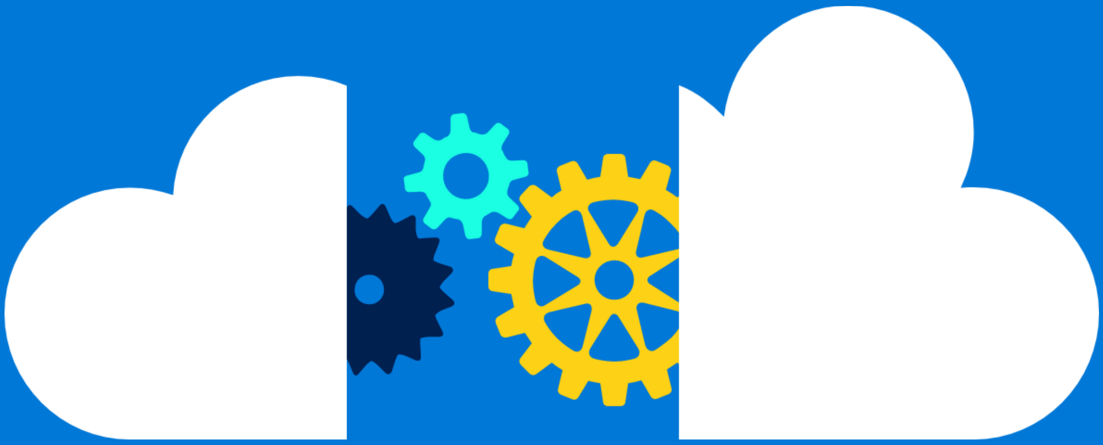
Microsoft



High-performance computing on Microsoft Azure: GlusterFS

Introduction to creating an Azure HPC cluster and HPC storage

Azure Customer Advisory Team (AzureCAT)

April 2018

Contents

Introduction

With large datasets, you need a simple and flexible storage solution. Getting started with high-performance computing (HPC) doesn't have to be confusing. A cloud-based HPC environment offers advantages that traditional HPC setups can't match, including the ability to create and destroy the environment on demand—quickly, completely, and easily. This article and the resources in the accompanying GitHub repository remove the guesswork, so you can implement HPC on Microsoft Azure infrastructure as a service (IaaS) with four types of storage and the GlusterFS file system (GFS).

The scripts and templates in the GitHub repository are designed to be a first step in exploring a cloud-based HPC storage and compute architecture. Figure 1 shows the deployed architecture. Although many different configurations are possible, this deployment creates an RDMA-connected compute cluster to which the GlusterFS file system is attached.

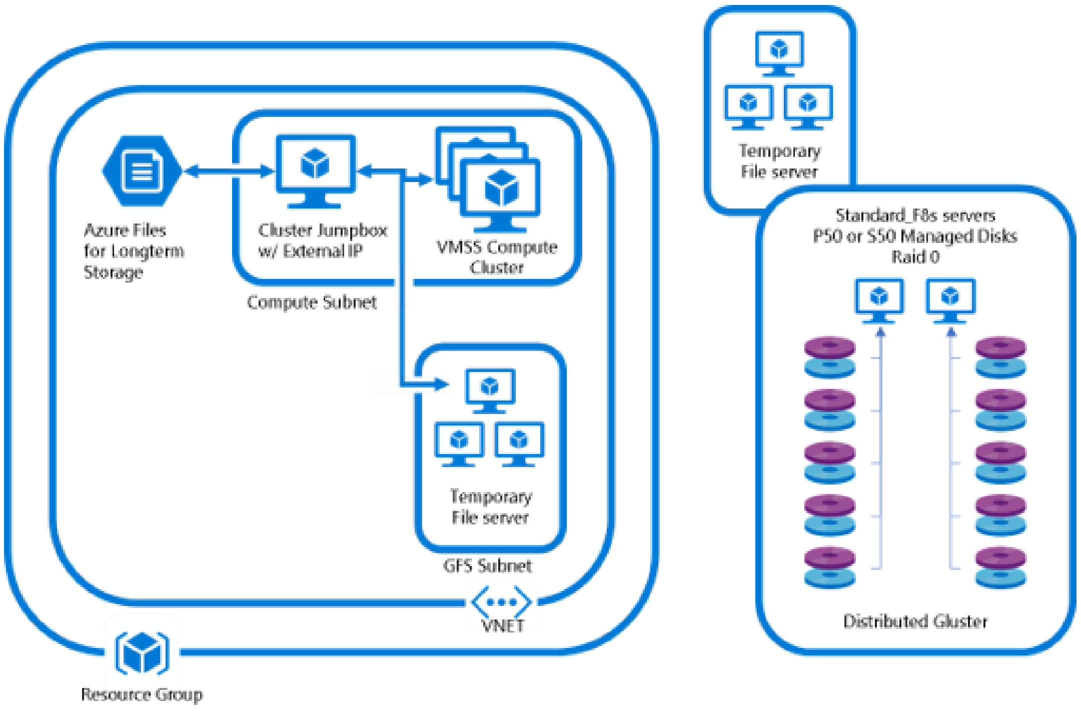


Figure 1. HPC architecture on Azure with GlusterFS

Our test setup was designed for simplicity and performance, but consider the following:

- You might be able to get the required performance from shared storage GlusterFS using Premium disks. If the performance is not adequate, you can also use local storage on each compute node as a burst buffer.
- For long-term storage, you can also use the multi-tiering feature of GlusterFS with a Standard disk pool using 4 TB-capacity disks.
- GlusterFS supports both the Windows and Linux clients if needed—that is, Network File System (NFS) and Common Internet File System (CIFS).
- To get a cost estimate for this architecture, use the Azure pricing calculator.

Quick start

To get started quickly, here are all the steps you need to deploy an InfiniBand-enabled compute cluster with a GlusterFS file server attached and mounted. The rest of this article describes these steps in more detail.

1. Make sure you have sufficient quota for H-series (compute cluster) and F-series (jumpbox and storage cluster) virtual machines.
2. Open the cloud shell from the Azure portal.
3. Clone the repository:

   ```
   git clone https://github.com/az-cat/HPC-az-gfs
   ```

4. Update credentials.yaml with the credentials needed by Batch Shipyard Remote Filesystem to access Azure resources.
5. From inside the cloned repository folder, run:

   ```
   ./create_cluster [Resource Group name] [Compute nodes]
   ```

   For example: `./create_cluster.sh my-test-rg01 3`
6. When the deployment finishes, note the SSH string. You need this to access your cluster.
7. Change directories to the folder that was created for your resource group name, and then run the SSH string. For example:

   ```
   ssh -i id_rsa_batchshipyardkey azureuser@23.45.67.89
   ```

8. To see the storage available on the jumpbox, log on to the jumpbox, and then run:

   ```
   df -h
   ```

9. To see the compute node IP addresses, run:

   ```
   cat bin/hostips
   ```

10. Connect via SSH to the first IP address listed; for example:

    ```
    ssh 10.0.0.6
    ```

    Where **10.0.0.6** represents the first IP address in the output list from the previous command.

    The compute nodes are Azure H16R-size virtual machines that are connected via InfiniBand for low-latency communication.

Deployment process

The deployed architecture includes GlusterFS, a free, scalable, open source distributed file system specifically optimized for cloud storage. You can create large, distributed storage solutions for media streaming, data analysis, and other data-rich and bandwidth-intensive tasks. The GlusterFS architecture includes a server node and a client node in a cluster. GlusterFS stores and locates data using an elastic hash algorithm. Servers can be added or removed whenever required.

This deployment uses four different types of storage:

- Physically attached storage is used as a burst buffer and is located at /mnt/resource on each node.
- A NFS shared from the jumpbox and located at /mnt/scratch is created by the /scripts/hn-setup_gfs.sh script:

```
sudo mount -t cifs //myStorageAccount.file.core.windows.net/mystorageshare
/mnt/mymountdirectory -o
vers=3.0,username=mystorageaccount,password=mystorageaccountkey,dir_mode=0777,fi
le_mode=0777
#Setup the NFS server, mount the gluster, get Long Term Storage Keys
echo "/mnt/scratch $localip.*(rw,sync,no_root_squash,no_all_squash)" | tee -a
/etc/exports
echo "$GFSIP:/gv0       /mnt/gfs  glusterfs   defaults,_netdev  0  0" | tee -a
/etc/fstab

systemctl enable rpcbind
systemctl enable nfs-server
systemctl enable nfs-lock
systemctl enable nfs-idmap
systemctl start rpcbind
systemctl start nfs-server
systemctl start nfs-lock
systemctl start nfs-idmap
systemctl restart nfs-server
mount -a
```

- GlusterFS is shared from the storage cluster mounted at /mnt/gfs, created using Batch Shipyard in the create_cluster.sh script:

```
#BATCH SHIPYARD COMMANDS
echo ------------------------- `date +%F" "%T` Adding disks
SHIPYARD_CONFIGDIR=. shipyard fs disks add
az group update -n $RG --set tags.Type=Compute_with_GFS tags.LaunchTime=`date
+%F_%T`
echo ------------------------- `date +%F" "%T` Creating servers
SHIPYARD_CONFIGDIR=. shipyard fs cluster add -y mystoragecluster
```

- Three 5 TB Azure Files shares are mounted on the jumpbox at /mnt/lts1, /mnt/lts2, and /mnt/lts3. This Common Internet File System (CIFS) share can be mounted to both the Windows and Linux operating systems. These Azure Files shares are subject to the performance limits. The size can be altered by increasing the quota in the following line of the create_cluster.sh script:

```
az storage share create --name longtermstoragetwo --quota 5000 --account-name
$ltsName --account-key $ltsKey
```

Different types of storage are deployed because a workload typically needs multiple storage structures, as Figure 2 demonstrates. In the data life cycle, active high-performance data resides in a hot storage tier like the scratch space with premium solid-state drives (SSDs). Over time, as data is accessed less often like the temporary workload, it can be stored in a warm tier with lower performance, physically attached storage—Network File System (NFS) shares or standard hard disk drives (HDDs). Later in the life cycle when data is rarely accessed by the user but must be retained, it can be archived in a cold tier.

Using this life-cycle model, this deployment uses GlusterFS for the equivalent of the campaign workload. The Azure Files share supports long-term data retention.

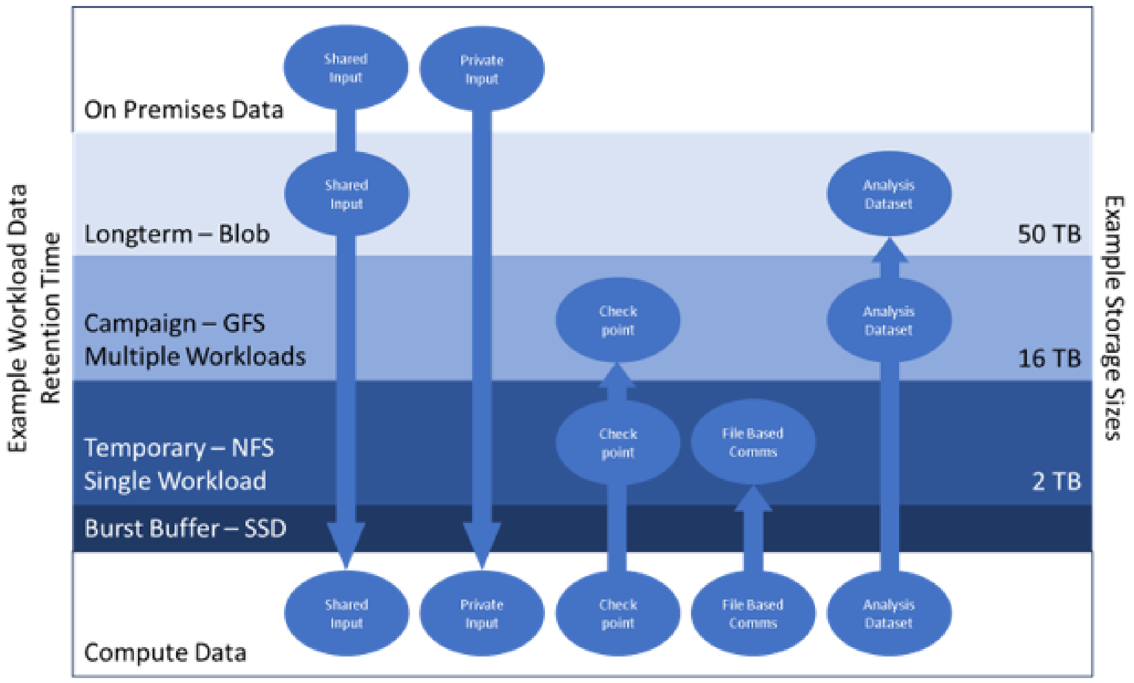


Figure 2. This sample workload data is stored in various shares. GlusterFS is used for campaign data.

Tools in the repo

The GitHub repository includes the scripts, templates, and tools you need to create the HPC cluster on Azure with a GlusterFS file system. This deployment uses Azure Resource Manager templates (Table 1). Batch Shipyard is also used to configure the standalone remote file system because it supports automatic provisioning of a GlusterFS storage cluster for both scale-up and scale-out scenarios.

Table 1. Resource Manager templates used in this deployment

Template	Description
azuredeploy.json	Primary deployment script. It creates all the resources needed for this example. This template does not affect the file server—its deployment is completely managed by Batch Shipyard.
Parameters.json	Specifies the parameters for the azuredeploy.json template.

Unlike many Resource Manager templates, azuredeploy.json is not meant to be run independently. It is designed to be deployed in connection with features called by the create_cluster.sh master script (Table 2). The azuredeploy.json template also calls two other scripts that configure the head node and the compute nodes. These scripts are designed to be used when GlusterFS is deployed in the virtual network where the template is deployed.

Table 2. Scripts in this deployment

Script	Description	Location
create_cluster.sh	The master script that downloads Batch Shipyard, deploys a storage file server and a compute cluster, and then mounts Azure Files.	/HPC-az-gfs
hn-setup_gfs.sh	Head node setup. Performs a number of basic node configuration commands, installs needed packages, and starts the NFS server.	/HPC-az-gfs/scripts
cn-setup_gfs.sh	Compute node setup. Installs a few packages, installs Ganglia, and configures the environment for MPI execution of applications.	/HPC-az-gfs /scripts

In addition, downloading data transfer tools is recommended for this deployment. You can use the following tools to transfer data to the file server, the compute cluster, or the storage blob:

- **Blobxfer** is an advanced data movement tool and library for Azure Storage Blob and Azure Files. It includes a command-line interface (CLI) you can use to copy your files into or out of Azure Storage or to integrate the blobxfer data movement library into your own Python scripts. Install binaries for blobxfer.
- **Secure copy protocol (scp)** is a Linux protocol based on SSH (Secure Shell) that provides quick, secure command-line file transfers between two computers. Command-line functionality can be used in your own batch files and scripts to automate file transfers.

Storage deployment

During deployment, the GLUSTERFS file server is created first using Batch Shipyard. The deployment scripts download the latest Batch Shipyard binary and create the file server based on the credentials in the credentials.yaml file and the configuration definitions in the fs.yaml and configuration.yaml files.

NOTE: Before deploying, make sure to add the required credentials to the credentials.yaml file. You need these to get artifacts from Azure. See the "Credentials configuration" section later in this article.

To deploy the file server, a resource group is created, and then the managed disks defined in fs.yaml are created. Next, the deployment scripts create a virtual network, a subnet, and the nodes used for the file server. Managed disks are then attached to the file server nodes. An F8-size virtual machine is used as the file server with 12 P30 managed disks for 12 TB of storage using a distributed, RAID 0 configuration. You can modify this setup by editing the fs.yaml file.

For information about how many disks can be attached to different virtual machine sizes, see Compute optimized virtual machine sizes.

Compute deployment

When you deploy the HPC cluster using the default configuration, more than 29 TB is available for the compute cluster. The Azure architecture is simple. For the compute cluster, an H16r/H16mr Virtual Machine Scale Set is specified in azuredeploy.json (lines 298 to 350). The nodes are automatically deployed in a single placement group and connected via InfiniBand hardware.

Scale sets do not have external or public-IP addresses, so when the new virtual network is created along with the HPC cluster, a jumpbox must be added for management access. (See azuredeploy.json.) This virtual machine is not the cluster headnode or rank 0—it is simply a way to access your scale sets and can be a small size.

In addition, azuredeploy.json runs a custom script on the jumpbox that downloads and calls hn-setup_gfs.sh. This script configures the head node and installs and starts the NFS server. It also sets up passwordless authentication to allow a separate home directory on each node:

```
for name in `cat /home/$USER/bin/hostips`; do
        sshpass -p "$PASS" ssh $USER@$name "mkdir -p .ssh"
        cat /home/$USER/.ssh/config | sshpass -p "$PASS" ssh $USER@$name "cat >>
.ssh/config"
        cat /home/$USER/.ssh/id_rsa | sshpass -p "$PASS" ssh $USER@$name "cat >>
.ssh/id_rsa"
        cat /home/$USER/.ssh/id_rsa.pub | sshpass -p "$PASS" ssh $USER@$name "cat >>
.ssh/authorized_keys"
        sshpass -p "$PASS" ssh $USER@$name "chmod 700 .ssh; chmod 640
.ssh/authorized_keys; chmod 400 .ssh/config; chmod 400 .ssh/id_rsa"
        cat /home/$USER/bin/hostips | sshpass -p "$PASS" ssh $USER@$name "cat >>
/home/$USER/hostips"
        cat /home/$USER/bin/hosts | sshpass -p "$PASS" ssh $USER@$name "cat >>
/home/$USER/hosts"
        cat /home/$USER/bin/cn-setup_gfs.sh | sshpass -p "$PASS" ssh $USER@$name "cat >>
/home/$USER/cn-setup_gfs.sh"
        sshpass -p $PASS ssh -t -t -o ConnectTimeout=2 $USER@$name 'echo "'$PASS'" | sudo
-S sh /home/'$USER'/cn-setup_gfs.sh '$IP $USER $myhost $GFSIP & > /dev/null 2>&1
done
```

This step protects connectivity from being lost if the NFS server locks up. Also, the jumpbox is not necessarily the same type of virtual machine as the compute nodes.

The host list is populated using nmap (network mapper) commands instead of relying on a response from the Azure CLI.

For monitoring, Ganglia is installed on the jumpbox and all the compute nodes:

```
chmod +x install_ganglia.sh
./install_ganglia.sh $myhost azure 8649
```

The hn-setup_gfs.sh script also launches the cn-setup_gfs.sh script, which mounts the GlusterFS and NFS file server on all the compute nodes. The compute cluster is created in a separate subnet.

Finally, the script installs the selected application (the **solver** parameter) on the share based on the first node in the host list. The azuredeploy.json template also creates an Azure Files storage account, which is used for long-term storage.

After the Resource Manager template has been fully deployed, the create_cluster.sh script gets the storage account keys, and then mounts the storage account to the jumpbox.

Credential configuration

For Batch Shipyard to complete its deployment without security prompts, you must configure an Azure service principal and an authentication key. These credentials are required for Batch Shipyard to operate using Azure resources. If you are a user of a directory that does not allow you to register an application, contact your Azure subscription owner to create a service principal.

After the service principal is created, you must get the Azure AD directory ID, then register the application to use with Azure AD. Copy the application ID that is generated. Next, generate the authentication key and copy its value. You can use Azure portal to get or create a directory ID, register the application, and generate the key. For detailed steps, see Create an Azure Active Directory application.

The values of the directory ID, application ID, and key are specified as the environment variables in the credentials.yaml file during step 4 of the installation in the next section.

Installation

To deploy an InfiniBand-enabled compute cluster with GlusterFS attached and mounted, follow these steps:

1. Make sure your Azure subscription has sufficient quota for the H-series (compute cluster) and F-series (jumpbox and storage cluster) virtual machines used in this deployment.
2. Open the cloud shell from the Azure portal.
3. Clone the repository:

   ```
   git clone https://github.com/az-cat/HPC-az-gfs
   ```

4. In the credentials.yaml file, update the Batch Shipyard RemoteFS credentials file with the required Azure service principal entries.
5. From inside the cloned repository folder, run:

   ```
   ./create_cluster [Resource Group name] [Compute nodes]
   ```

   For example: `./create_cluster.sh my-test-rg01 3`

6. When the deployment finishes, note the SSH string. You need this to access your cluster.

7. Change directories to the folder that was created for your resource group name, and then run the SSH string. For example:

   ```
   ssh -i id_rsa_batchshipyardkey azureuser@23.45.67.89
   ```

8. Log on to the jumpbox, then check the available storage by running:

   ```
   df -h
   ```

9. To see the compute node IP addresses, run:

   ```
   cat bin/hostips
   ```

10. Use SSH to go to the first IP address listed.

    ```
    ssh [address]
    ```

    For example, run:

    ```
    ssh 10.0.0.6
    ```

    Where **10.0.0.6** is the first IP address listed from the output of step 9.

    The compute nodes are Azure H16r-size virtual machines that are connected via InfiniBand for low-latency communication.

Learn more

For more information, see the following resources:

- Availability of H-series VMs in Microsoft Azure. Azure blog. September 26, 2016.
- Cray supercomputers are coming to Azure. Azure blog. October 23, 2017.
- The long rise of HPC in the Cloud. *Inside HPC*, March 1, 2017.
- The cloud is great for HPC: Discuss. *The Register*. June 16, 2017.