Network-attached-Storage-Demo

Network-attached storage (NAS) is a file-level (as opposed to block-level storage) computer data storage server connected to a computer network providing data access to a heterogeneous group of clients. NAS is specialized for serving files either by its hardware, software, or configuration. It is often manufactured as a computer appliance – a purpose-built specialized computer. NAS systems are networked appliances that contain one or more storage drives, often arranged into logical, redundant storage containers or RAID. Network-attached storage removes the responsibility of file serving from other servers on the network. NAS uses file-based protocols such as NFS (popular on UNIX systems), SMB (Server Message Block) (used with MS Windows systems), AFP (used with Apple Macintosh computers), or NCP (used with OES and Novell NetWare). NAS units rarely limit clients to a single protocol. Potential benefits of dedicated network-attached storage, compared to general-purpose servers also serving files, include faster data access, easier administration, and simple configuration.

The hard disk drives with "NAS" in their name are functionally similar to other drives but may have different firmware, vibration tolerance, or power dissipation to make them more suitable for use in RAID arrays, which are often used in NAS implementations.

NAS provides both storage and a file system. This is often contrasted with SAN (storage area network), which provides only block-based storage and leaves file system concerns on the "client" side. SAN protocols include Fibre Channel, iSCSI, ATA over Ethernet (AoE) and HyperSCSI.

One way to loosely conceptualize the difference between a NAS and a SAN is that NAS appears to the client OS (operating system) as a file server (the client can map network drives to shares on that server) whereas a disk available through a SAN still appears to the client OS as a disk, visible in disk and volume management utilities (along with client's local disks), and available to be formatted with a file system and mounted.

In this project, we will implement a tooling website solution which makes access to DevOps tools within the corporate infrastructure easily accessible. We will implement a solution that consists of following components:

Infrastructure: Azure

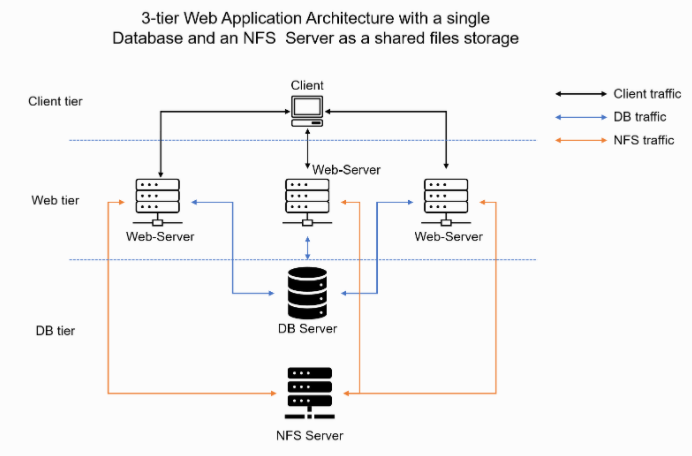
Webserver(s): Red Hat Enterprise Linux 7

Database Server: Ubuntu 20.04 + MySQL

Storage Server: Red Hat Enterprise Linux 8 + NFS Server

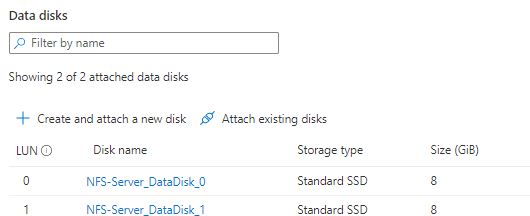
Programming Language: PHP

Code Repository: GitHub



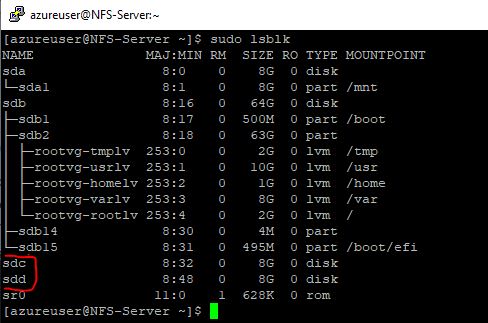
Step 1: Prepare the NFS server

Create a RHEL Linux 7 VM that will serve as the NFS server named *NFS-Server*. Create 2 volumes (disks) for the VM of the same size.

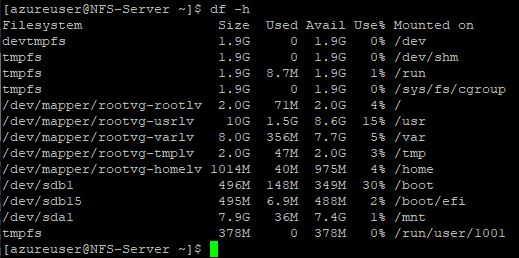


SSH to the VM after creation (using Putty in a Windows OS).

Use *lsblk* command to inspect what block devices (disks) are attached to the server. Notice names of the newly created devices. All devices in Linux reside in /dev/ directory. Inspect it with ls /dev/ and make sure all 2 newly created block devices are seen there



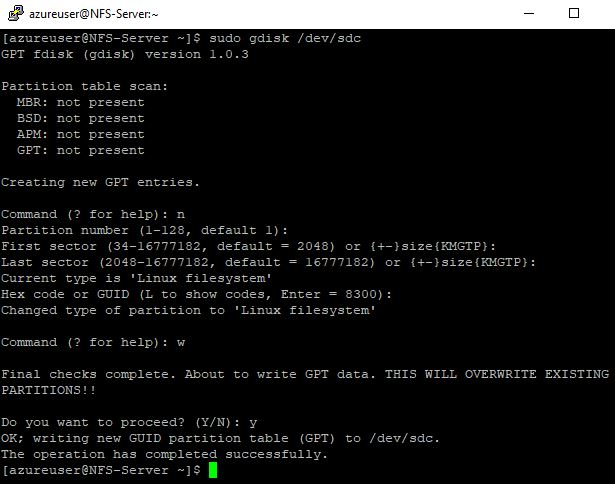
Use *df -h* command to see all mounts and free space on the server



Use *gdisk* utility to create a single GPT partition on each of the 3 disks.

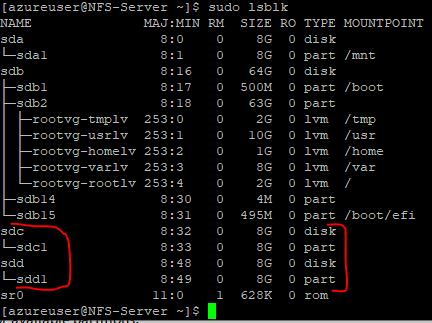
To create a partition on the first disk i.e. /dev/sdc, run the command and follow the prompt:

*$ sudo gdisk /dev/sdc*



Create a partition on the other disk i.e. /dev/sdd in the same way

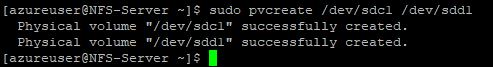
Use *lsblk* to view the newly configured partition on the 2 disks



From the partition created for each disk, create a Physical Volume (PV).

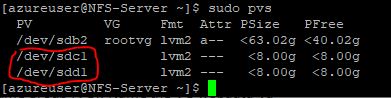
Use *pvcreate* utility to mark each of the disks as physical volumes (PVs) to be used by the LVM (Logical Volume):

*$ sudo pvcreate /dev/sdc1 /dev/sdd1*



Run *sudo pvdisplay* to view the info about the created PVs.

Run *sudo pvs* to view the PVs summary.



Physical volumes are combined into volume groups (VGs). It creates a pool of disk space out of which logical volumes can be allocated.

Use *vgcreate* utility to add the 2 PVs to a volume group (VG). Name the VG *nfsdata-vg*

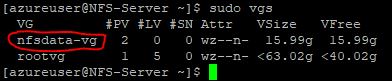
*$ sudo vgcreate nfsdata-vg /dev/sdc1 /dev/sdd1*

C:\Users\osygroup\Desktop\Darey\Project 7\vg1.JPG

Run *sudo vgdisplay* to view the info about the created VG.

Run *sudo vgs* to view the VG summary.

Run *sudo vgdisplay –v* to view all info about a VG including the PVs and LVs of a VG.

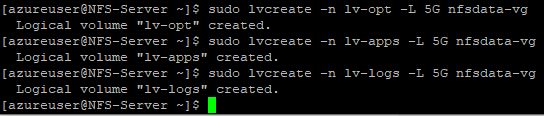


Use *lvcreate* utility to create 3 logical volumes: *lv-opt*, *lv-apps* and *lv-logs each of 5gb (*VG size is 16gb).

*$* *sudo lvcreate -n lv-opt -L 5G nfsdata-vg*

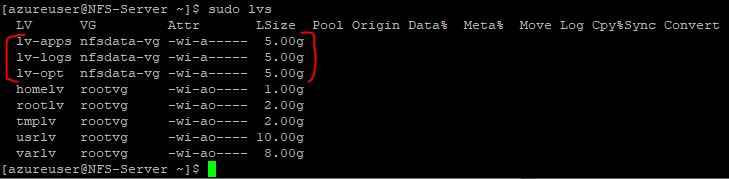
*$* *sudo lvcreate -n lv-apps -L 5G nfsdata-vg*

*$* *sudo lvcreate -n lv-logs -L 5G nfsdata-vg*



Run *sudo lvdisplay* to view the info about the created LVs.

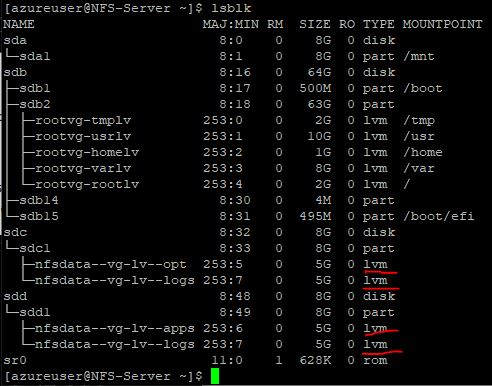
Run *sudo lvs* to view the LVs summary.



Verify the entire setup

*$ sudo vgdisplay -v* (view complete setup - VG, PV, and LV)

*$ sudo lsblk*



Use mkfs.xfs (an xfs file system) to format the logical volumes with xfs filesystem

*$ sudo mkfs.xfs /dev/nfsdata-vg/lv-opt*

Or

*$ sudo mkfs -t xfs /dev/nfsdata-vg/lv-opt*



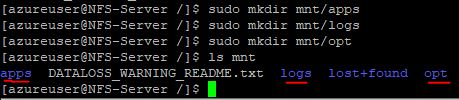
Format *lv-apps and lv-logs* in the same way:

*$ sudo mkfs.xfs /dev/nfsdata-vg/lv-apps*

*$ sudo mkfs.xfs /dev/nfsdata-vg/lv-logs*

Create /mnt/apps directory to be used by the webserver(s), create /mnt/logs directory to be used by the webserver(s) logs, and create /mnt/opt directory to be used by Jenkins server.

NOTE: /mnt directory is mounted to ephemeral storage in Azure RHEL 8. After reboot or instance stop/start, all data is lost. It is advised to create a new mount directory for the apps, logs and opt directories and not use an ephemeral storage like /mnt. See <https://github.com/Azure/WALinuxAgent/issues/1971> and <https://docs.microsoft.com/en-us/azure/virtual-machines/managed-disks-overview#temporary-disk> for more details.



Mount lv-apps logical volume on /mnt/apps

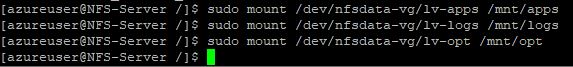
*$ sudo mount /dev/nfsdata-vg/lv-apps /mnt/apps*

Mount lv-logs logical volume on /mnt/logs

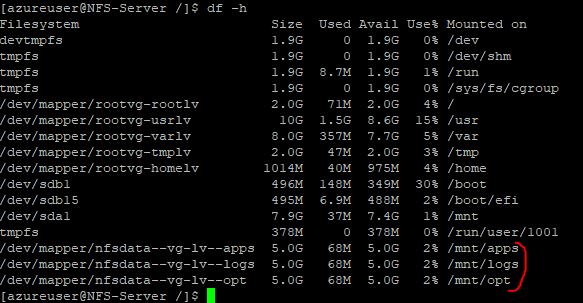
*$ sudo mount /dev/nfsdata-vg/lv-logs /mnt/logs*

Mount lv-opt logical volume on /mnt/opt

*$ sudo mount /dev/nfsdata-vg/lv-opt /mnt/opt*

**

Run *df –h* to confirm that the logical volumes were mounted on their correct paths



Note that if the server is restarted, the mount configurations will be lost.

Update /etc/fstab file so that the mount configurations will persist after restart of the server.

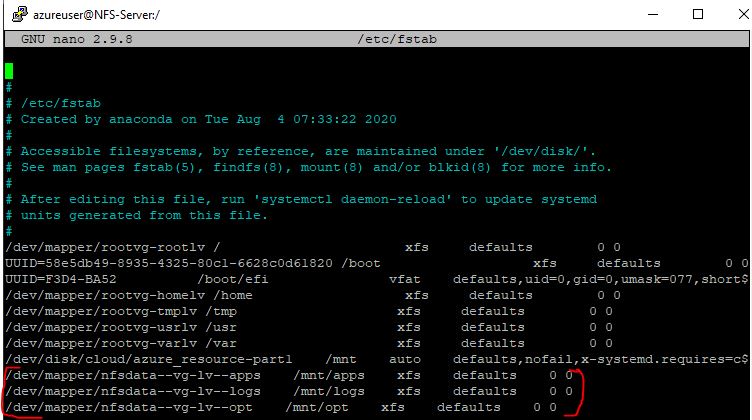
Add the below three lines at the end of the contents of the /etc/fstab with any editor of choice

/dev/mapper/nfsdata--vg-lv--apps /mnt/apps xfs defaults 0 0

/dev/mapper/nfsdata--vg-lv--logs /mnt/logs xfs defaults 0 0

/dev/mapper/nfsdata--vg-lv--opt /mnt/opt xfs defaults 0 0

*$ sudo nano /etc/fstab*



Restart the server and run *df -h* to confirm if the mount configurations persisted.

Install NFS server, configure it to start on reboot and make sure it is up and running

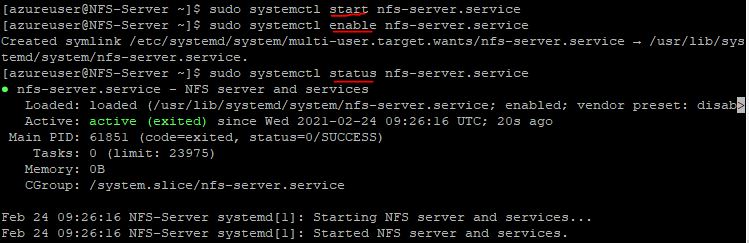
*$ sudo yum -y update*

*$ sudo yum install nfs-utils -y*

*$ sudo systemctl start nfs-server.service*

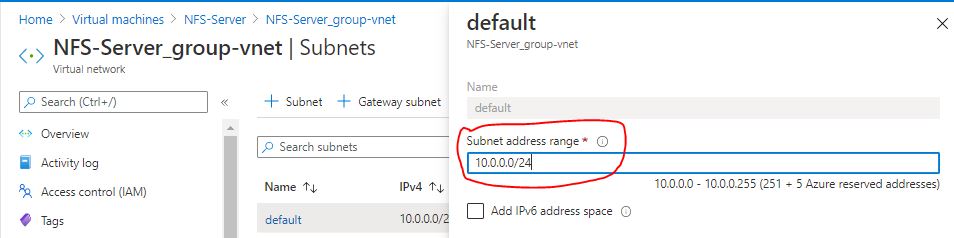
*$ sudo systemctl enable nfs-server.service*

*$ sudo systemctl status nfs-server.service*



Export the mounts for webserver(s)’ subnet cidr to connect as clients.

For simplicity, the webserver(s) will be inside the same subnet as the NFS server, but in production setup each tier may be separated inside its own subnet for higher level of security. To check the subnet CIDR of the NFS server, click on the Virtual network/subnet link on the Overview of the NFS-Server, and then click on Subnets pane of the VNET and copy the subnet CIDR:



Make sure to set up the permission that will allow the Web server(s) to read, write and execute files on NFS:

*$ sudo chown -R nobody: /mnt/apps*

*$ sudo chown -R nobody: /mnt/logs*

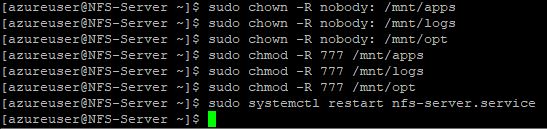
*$ sudo chown -R nobody: /mnt/opt*

*$ sudo chmod -R 777 /mnt/apps*

*$ sudo chmod -R 777 /mnt/logs*

*$ sudo chmod -R 777 /mnt/opt*

*$ sudo systemctl restart nfs-server.service*



Configure access to NFS for clients within the same subnet.

Create a file ‘exports’ in /etc with any text editor of choice.

*$ sudo nano /etc/exports*

Then copy the following into the file and save:

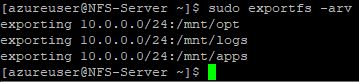
/mnt/apps 10.0.0.0/24(rw,sync,no\_all\_squash,no\_root\_squash)

/mnt/logs 10.0.0.0/24(rw,sync,no\_all\_squash,no\_root\_squash)

/mnt/opt 10.0.0.0/24(rw,sync,no\_all\_squash,no\_root\_squash)

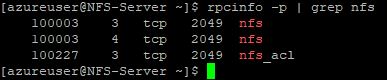
Then run:

*$ sudo exportfs -arv*

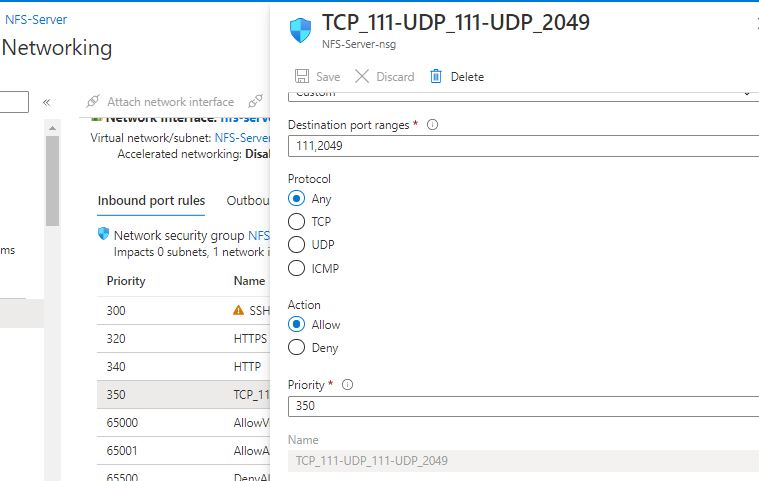


Check which port is used by NFS and open it using Security Groups (add new Inbound Rule)

*$ rpcinfo -p | grep nfs*



In order for NFS server to be accessible from the webserver(s), open following ports in the NSG of the NFS-Server: TCP 111, UDP 111, UDP 2049



NOTE: Check that there are no other firewalls like UFW or Firewalld running in the server. If there is any firewall enabled, ensure that it is disabled.

Run the following to disable Firewalld if it is enabled:

*$ sudo systemctl stop firewalld*

*$ sudo systemctl disable firewalld*

*$ sudo systemctl mask --now firewalld*

Step 2: Configure the database server

Create an Ubuntu 20.04 VM that will serve as the database server named *MySQL-Server*. MySQL Server is to be installed in the server. Run the following commands:

*$ sudo apt update*

*$ sudo apt install mysql-server*

When prompted, confirm installation by typing Y, and then ENTER.

Confirm the MySQL installation:

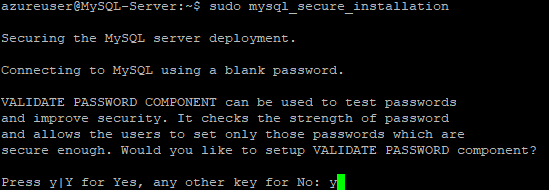
*$ mysql -V*



To configure the MySQL installation, it’s recommended to run a security script that comes pre-installed with MySQL. This script will remove some insecure default settings and lock down access to the database system. Start the interactive script by running:

*$ sudo mysql\_secure\_installation*

This will ask to configure the VALIDATE PASSWORD PLUGIN. Answer ‘y’ for yes.



The configuration will ask to select a level of password validation. Note that if ‘2’ is selected, there will be errors when attempting to set any password which does not contain numbers, upper and lowercase letters, and special characters, or which is based on common dictionary words.

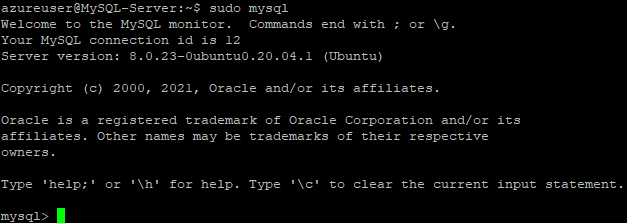
The configuration will next ask to set and confirm a password for the MySQL root user. This is not to be confused with the system root. The database root user is an administrative user with full privileges over the database system.

If password validation is enabled, the password strength for the root password just entered will be shown and the server will ask to continue with that password. If satisfied with the current password, enter Y for “yes” at the prompt. For the rest of the questions, press ‘Y’ and hit the ‘ENTER’ key at each prompt. This will remove some anonymous users and the test database, disable remote root logins, and load these new rules so that MySQL immediately respects the changes made:

When finished, test the MySQL console login by typing:

*$ sudo mysql*

This will connect to the MySQL server as the administrative database user root, which is inferred by the use of *sudo* when running this command. The output looks like this:



Create a database named *tooling* and a user named *webaccess*.

To create a new database, run the following command from your MySQL console:

*mysql> CREATE DATABASE tooling;*

Create a new user and grant the account full privileges on the database just created.

The following command creates a new user named *webaccess*, using mysql\_native\_password as default authentication method. This user’s password is ‘Password11#’. The password chosen should be in line with the level of password validation policy chosen when the VALIDATE PASSWORD PLUGIN was configured.

*mysql> CREATE USER 'webaccess'@'%' IDENTIFIED WITH mysql\_native\_password BY 'Password11#';*

Give this user permission over the *tooling* database:

*mysql> GRANT ALL ON tooling.\* TO 'webaccess'@'%';*

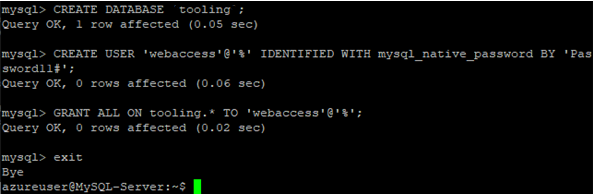
This will give the *webaccess* user full privileges over the *tooling* database, while preventing this user from creating or modifying other databases on your server.

Run this also:

*mysql> FLUSH PRIVILEGES;*

Exit the MySQL shell with:

*mysql> exit*



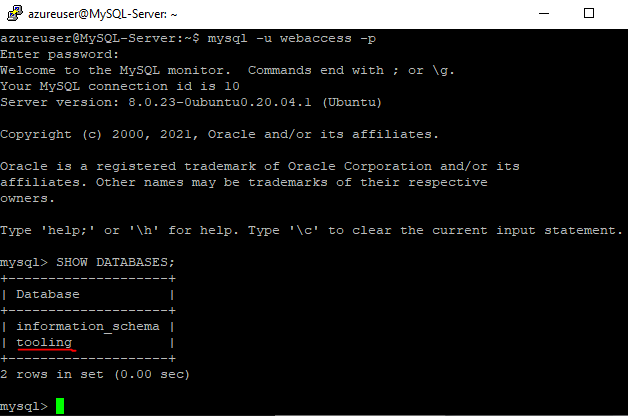
Test if the new user has the proper permissions by logging in to the MySQL console again, this time using the custom user credentials:

*$ mysql -u webaccess -p*

Notice the *-p* flag in this command, which will prompt for the password used when creating the *ernesto* user. After logging in to the MySQL console, confirm access to the *tooling* database:

*mysql> SHOW DATABASES;*

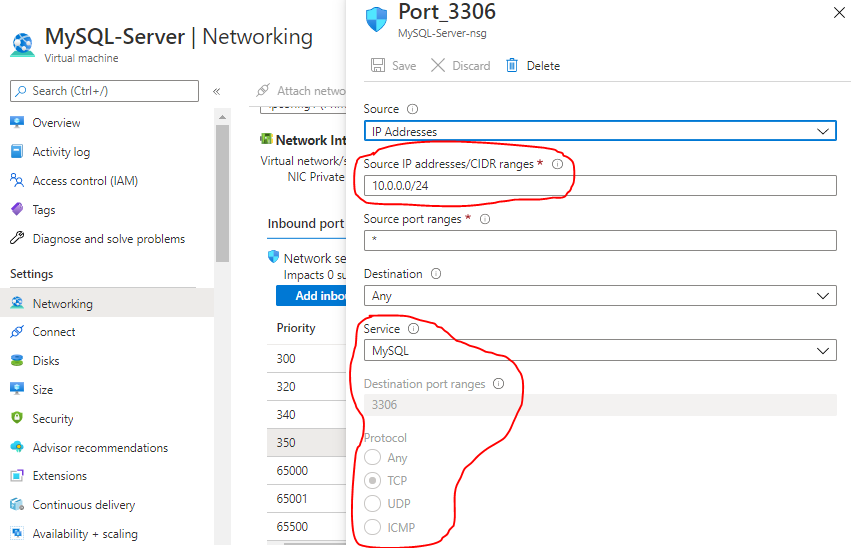
This will give the following output:



Exit the MySQL shell with:

*mysql> exit*

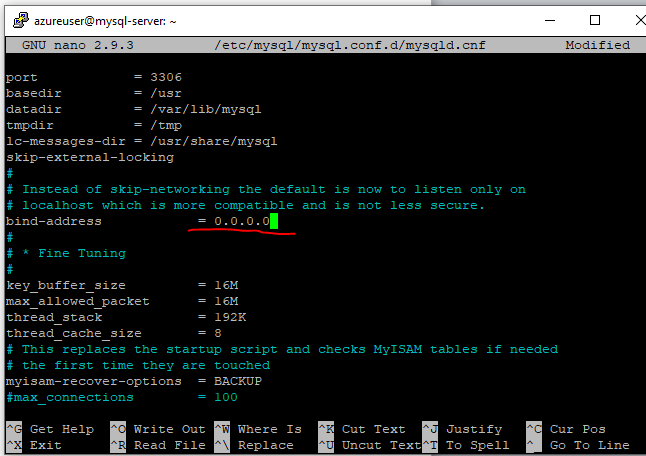
MySQL server uses TCP port 3306 by default. Add an inbound security rule to the Network Security Group of MySQL-Server VM to open inbound connection through port 3306. For extra security, do not allow all IP addresses to reach the MySQL server. Allow access only to the subnet CIDR of the webserver(s) i.e. the subnet CIDR of the NFS-Server (all the VMs are created in the same subnet of the NFS-Server VM).



Next, configure the MySQL server to allow connections from remote hosts, in this case, the webserver(s). Edit the MySQL server configuration file by using a text editor:

*$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf*

Replace ‘127.0.0.1’ with ‘0.0.0.0’ on the bind-address line



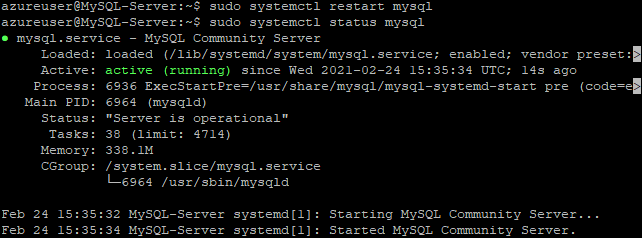
NOTE: If you wish to access a MySQL from more than one, but less than all the interfaces, you should bind to 0.0.0.0 and firewall off the interfaces you don't want to be accessed through. In this case, NSG was sued as the firewall.

Restart the MySQL server

*$ sudo systemctl restart mysql*

Confirm the status of the MySQL server

*$sudo systemctl status mysql*



Step 3: Prepare the Web Server(s)

One or more Web Server VMs can be created and pointed to the same NFS server and also connect to the same Database Server with the following instructions. During this step, NFS Client will be configured and a tooling application will be deployed to the web server(s).

Launch a new RHEL 8 Operating System VM and update the VM

*$ sudo yum update*

Install NFS client

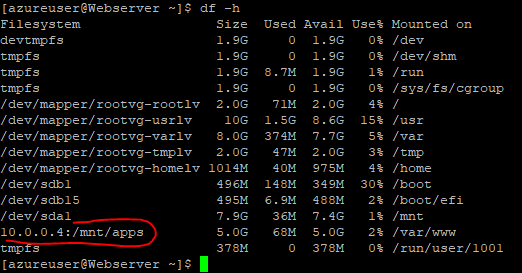
*$ sudo yum install nfs-utils nfs4-acl-tools -y*

Mount /var/www/ and target the NFS server’s export for apps

*$ sudo mkdir /var/www*

*$ sudo mount -t nfs -o rw,nosuid 10.0.0.4:/mnt/apps /var/www*

Verify that NFS was mounted successfully by running *df -h*.



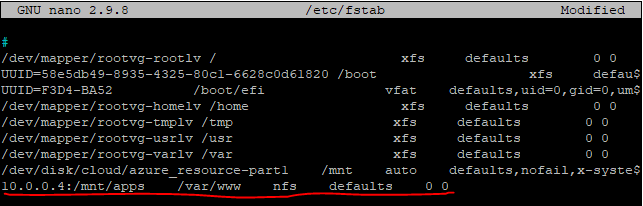
Make sure that the changes will persist on Web Server after reboot.

Update /etc/fstab file so that the mount configurations will persist after restart of the server.

Add the below line at the end of the contents of the /etc/fstab with any editor of choice:

10.0.0.4:/mnt/apps /var/www nfs defaults 0 0

*$ sudo nano /etc/fstab*



Install Apache:

*$ sudo yum install httpd*

Verify that Apache files and directories are available on the Web Server in /var/www and also on the NFS server in /mnt/apps. If the same files are seen, it means NFS is mounted correctly.





Locate the log folder for Apache (/var/log/httpd) on the Web Server and mount it to NFS server’s export for logs. Also make sure that the changes will persist on Web Server after reboot.

*$ sudo mount -t nfs -o rw,nosuid 10.0.0.4:/mnt/logs /var/log/httpd*

Add the below line at the end of the contents of the /etc/fstab with any editor of choice:

10.0.0.4:/mnt/logs /var/log/httpd nfs defaults 0 0

Install PHP (the DevOps tooling website is a PHP application).

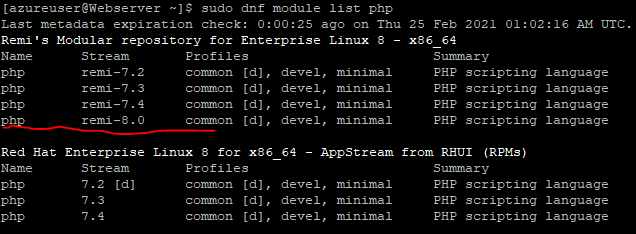
PHP 8 is not available in the default CentOS 8 and RHEL 8 package repositories. So EPEL and Remi repositories will be enabled:

*$ sudo dnf install -y epel-release* or *$ sudo dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm*

*$ sudo dnf install -y http://rpms.remirepo.net/enterprise/remi-release-8.rpm*

Run below command to list the available versions of PHP,

$ sudo dnf module list php



Run the following commands to reset PHP module and install PHP 8 from remi-8.0 module.

*$ sudo dnf module reset php*

*$ sudo dnf module install -y php:remi-8.0*

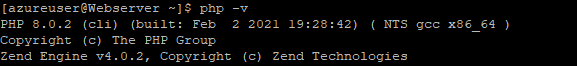
Install the following additional modules and packages:

*$ sudo dnf install -y php-{mysqlnd,xml,xmlrpc,curl,gd,imagick,mbstring,opcache,soap,zip}*

By enabling these modules, the PHP code can become more versatile and adaptable, allowing seamless interaction with things such as MySQL.

Once the PHP packages are installed successfully then execute below command to verify PHP version:

*$ php -v*



Disable SELinux:

*$ sudo setenforce 0*

Also set SELinux to ‘disabled’ in the */etc/sysconfig/selinux* file:

*$ sudo nano /etc/sysconfig/selinux*

*SELINUX=disabled*

Restart and enable Apache server:

*$ sudo systemctl restart httpd*

*$ sudo systemctl enable httpd*

Fork the tooling source code from Darey.io Github Account to a personal Github account:

<https://github.com/darey-io/tooling>

Install Git:

*$ sudo yum install git*

Clone the tooling website’s code to the Webserver

*$ git clone https://github.com/YOUR\_USERNAME/tooling.git*

cd into the tooling directory and recursively copy the contents of the html directory into /var/www/html.

*$ cd tooling && sudo cp -r html/\* /var/www/html*

Run the following to disable Firewalld if it is enabled:

*$ sudo systemctl stop firewalld*

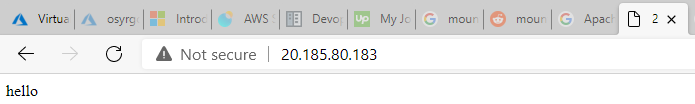
*$ sudo systemctl disable firewalld*

*$ sudo systemctl mask --now firewalld*

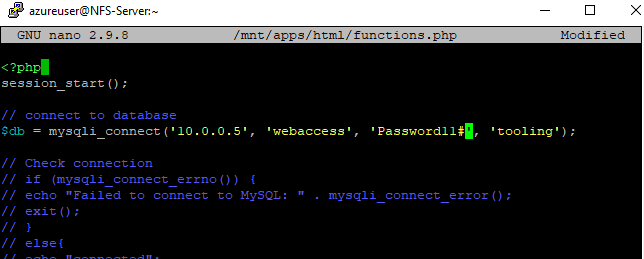
To ensure that the apache server can now serve content on the web server, simply create an index.html page with any chosen content within;

sudo bash -c "echo hello >> /var/www/html/index.html"

Head over to a web browser and type in the IP address of the web server to verify.



Within the /mnt/apps/html directory of the NFS server, open the *functions.php* file to edit it.



Where ’10.0.0.5’ is the private IP address of the MySQL VM, ‘webaccess’ is the username in the MySQL server, ‘Password11#’ is the password of the user, and ‘tooling’ is the database that the user ‘webaccess’ has full permissions in. This will set the database credentials to be used when connecting to the tooling database.

Back to the *tooling* directory cloned from the Github repository, there is a file *tooling-db.sql*. This is a backup of a database table that would be needed for the tooling website, and a user account that will be able to login to the tooling website. Users created in the tooling website will be added into this table.

Mysqldump (i.e. dump) the database table into the empty *tooling* database in the MySQL server:

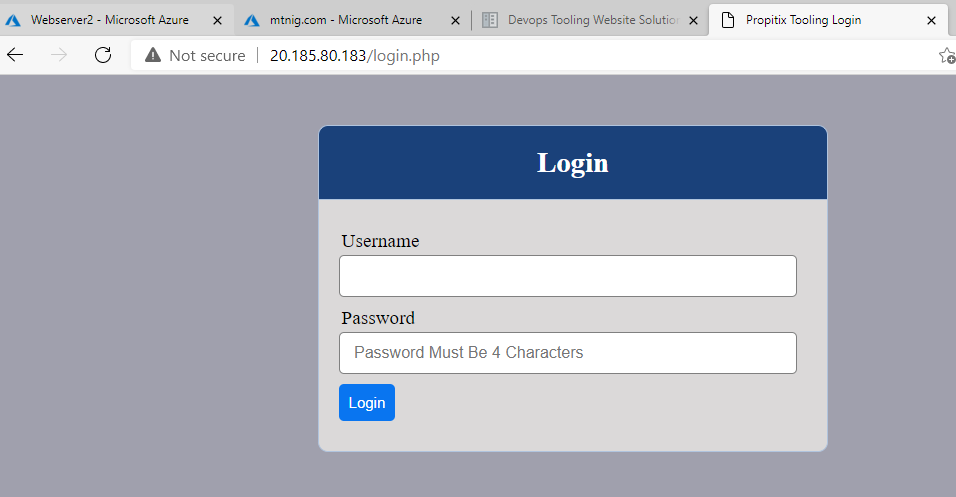
*$ sudo yum install mysql-server*

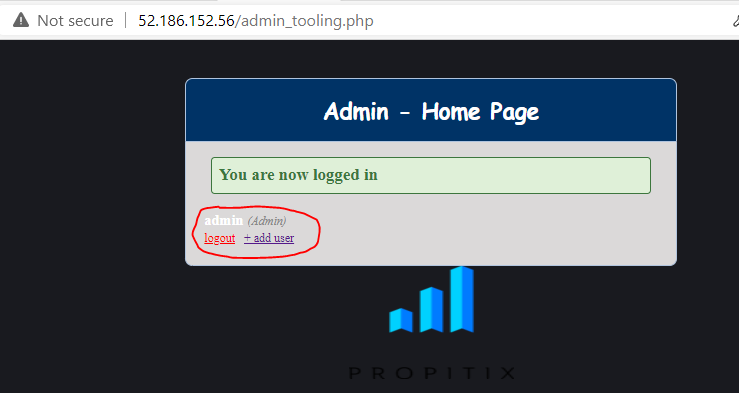
*$ mysql -u webaccess -p tooling -h 10.0.0.5 < tooling-db.sql*

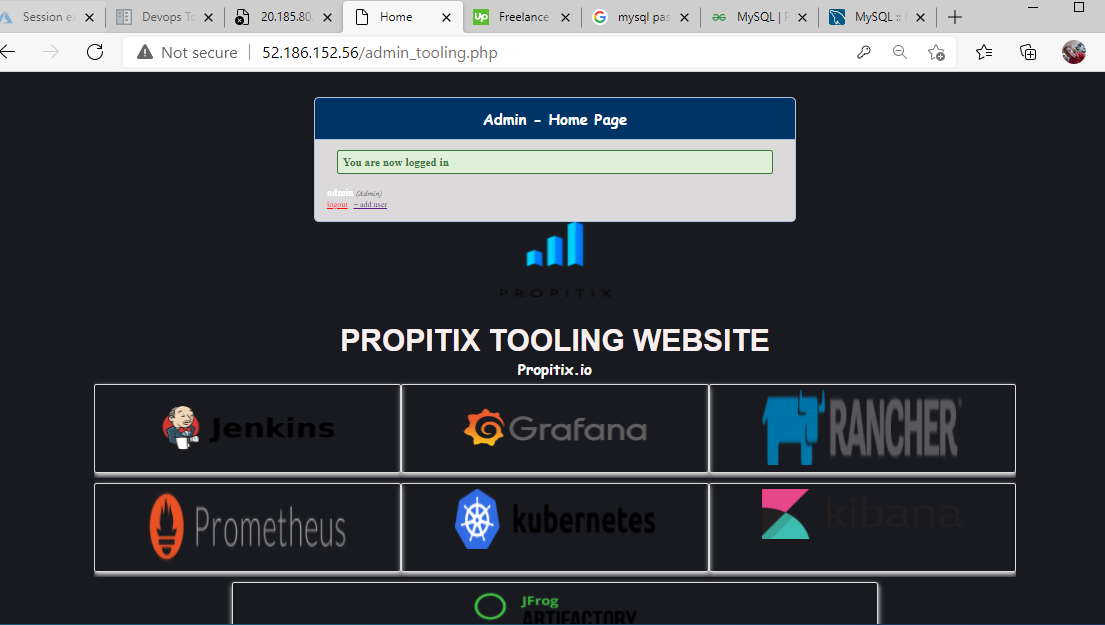
A user with username ‘*admin*’ and password ‘*admin*’ is automatically added to the table.

Blocker: There is a missing semicolon in the *tooling-db.sql* file on line 44, after the closed bracket. Edit the file and add the semicolon.

Head over to a web browser and type in the IP address of the web server to view the tooling website:







Users can be created in the database table. For example, a new admin user with username ‘*myuser’* and password ‘*password*’ can be created by running the following in the MySQL database:

*mysql> USE tooling;*

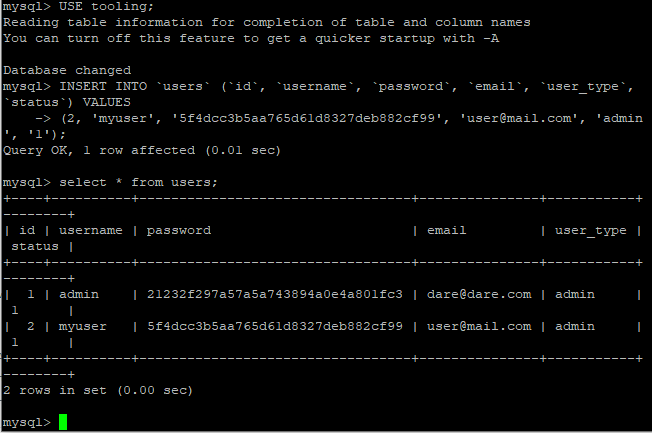
*mysql> INSERT INTO `users` (`id`, `username`, `password`, `email`, `user\_type`, `status`) VALUES*

*(2, 'myuser', '5f4dcc3b5aa765d61d8327deb882cf99', 'user@mail.com', 'admin', '1');*

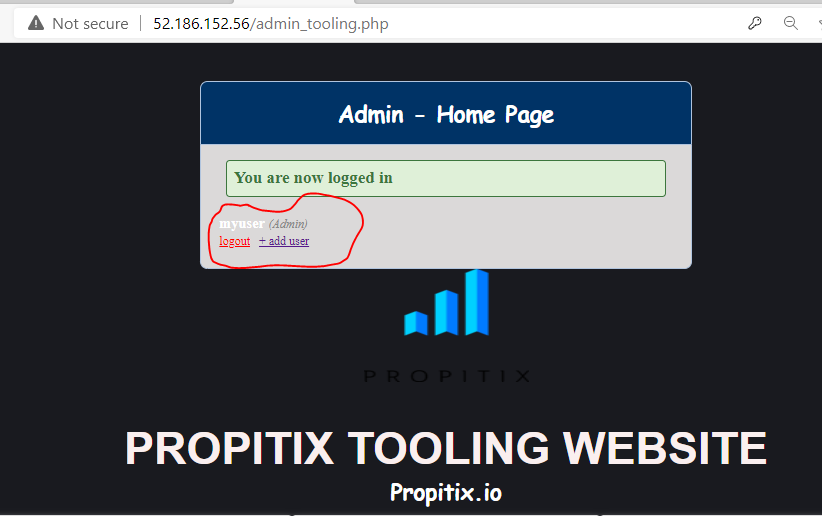
Don’t forget to change the *‘id’* to a different number, as it is the primary key in the table and it has to be unique.

To see a list of the users in the table, run:

*mysql> select \* from users;*

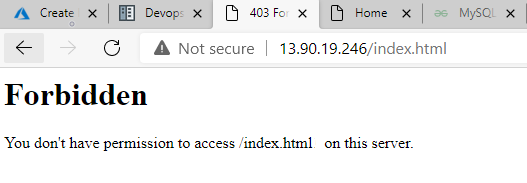


Head over to a browser and login to the tooling website using the new credentials:



The above instructions are for setting up a webserver for the tooling app. To add subsequent webservers, just create the webservers with the same instructions, but skip the part of cloning the tooling website repository to the server and copying the html directory into /var/www/. Also skip creating an index.html page. Once the mount is successful, these contents will be automatically copied from the NFS-Server.

If ‘Forbidden’ permission error is encountered when visiting the index page or the tooling website on a new webserver:



The above output indicates that SELinux is also preventing access to the web contents in the /var/www/html directory (recall the nfs share was mounted there).

Kindly run the following in the webserver to get permission to the files:

*$ sudo setsebool -P httpd\_use\_nfs=1*

The above command will modify any working SELinux policy behaviour to allow apache (httpd) to use nfs files, etc.

Kindly note that SSH into the webservers is not possible if the NFS-Server is not running. Also, the tooling website will be offline until the NFS-Server is running.

Encryption functions are used for the generation of a hashed password using a plain-text password string. It uses hashing techniques to generate the hashed password so as to hide a password in a table e.g.

*mysql> SELECT MD5('testing');*

-> 'ae2b1fca515949e5d54fb22b8ed95575'

Conclusion

NAS is useful for more than just general centralized storage provided to client computers in environments with large amounts of data. NAS can enable simpler and lower cost systems such as load-balancing and fault-tolerant email and web server systems by providing storage services. The potential emerging market for NAS is the consumer market where there is a large amount of multimedia data. Such consumer market appliances are now commonly available. Unlike their rack-mounted counterparts, they are generally packaged in smaller form factors. The price of NAS appliances has fallen sharply in recent years, offering flexible network-based storage to the home consumer market for little more than the cost of a regular USB or FireWire external hard disk. Many of these home consumer devices are built around ARM, x86 or MIPS processors running an embedded Linux operating system.

Credits

<https://en.wikipedia.org/wiki/Network-attached_storage>

<https://www.linuxtechi.com/create-extend-xfs-filesystem-on-lvm/>

<https://linuxize.com/post/how-to-stop-and-disable-firewalld-on-centos-7/>

<https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/sect-using_the_mount_command-mounting>

<https://blog.codeasite.com/how-do-i-find-apache-http-server-log-files/#:~:text=Default%20apache%20access%20log%20file,log>

<https://www.linuxtechi.com/install-php-8-centos-8-rhel-8/>

<https://www.tecmint.com/install-php-8-on-centos/>

<https://linuxs.info/how-to-uninstall-php-7-3-on-centos-7/>

<https://forum.remirepo.net/viewtopic.php?id=3991>

<https://linuxize.com/post/show-tables-in-mysql-database/#:~:text=To%20get%20a%20list%20of,run%20the%20SHOW%20TABLES%20command.&text=The%20optional%20FULL%20modifier%20will,as%20a%20second%20output%20column>.

<https://www.mysqltutorial.org/mysql-show-columns/>

<https://stackoverflow.com/questions/15989529/unknown-column-in-field-list-but-column-does-exist>

<https://www.tutorialspoint.com/how-to-fix-error-you-have-an-error-in-your-syntax-check-the-manual-that-corresponds-to-your-mysql-server-version-for-the-right-syntax-to-use-near>

<https://www.sqlshack.com/how-to-backup-and-restore-mysql-databases-using-the-mysqldump-command/>

<https://www.mysqltutorial.org/mysql-delete-statement.aspx>

<https://dev.mysql.com/doc/refman/8.0/en/encryption-functions.html>

<https://access.redhat.com/discussions/5473561>