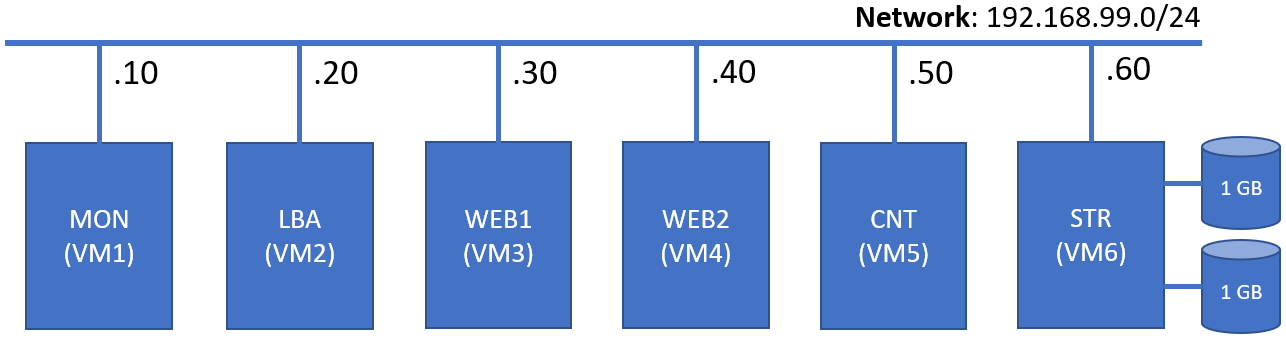
# Practice M8: Exam Preparation #1 (Ubuntu)

## Sample Exam

### Infrastructure

You will have to accomplish a set of tasks in the following infrastructure



All machines are of the same type (in terms of OS and hardware parameters)

Pease note that if your IP address is **A.B.C.D** then the following port-forwarding rules are in place:

|  |  |  |
| --- | --- | --- |
| A.B.C.D:10122 -> 192.168.99.10:22 | A.B.C.D:10222 -> 192.168.99.20:22 | A.B.C.D:10322 -> 192.168.99.30:22 |
| A.B.C.D:10422 -> 192.168.99.40:22 | A.B.C.D:10522 -> 192.168.99.50:22 | A.B.C.D:10622 -> 192.168.99.60:22 |
| A.B.C.D:20080 -> 192.168.99.10:80 | A.B.C.D:38080 -> 192.168.99.20:8080 | A.B.C.D:48080 -> 192.168.99.50:8080 |

Be sure to **follow** the **naming** **conventions** specified in the tasks checklist

Tasks execution order should not be derived from the order in which they are listed below. Please note that there are tasks that depend on the successful completion of one or more other tasks

Usually, all steps could be achieved following different paths and using different tools. In the end, not the means, but the **end results** are being **measured**

### Tasks checklist

#### Containers [12 pts]

Demonstrate knowledge and readiness to work with application containers:

* (T101 / 3 pts) Install **Docker** on the **CNT** machine
* (T102 / 2 pts) Configure **Docker** to start on boot and allow the **exam** user to use it without the need of **sudo**
* (T103 / 2 pts) Start (execute) a container based on the **hello-world** image

Start a container based on the **nginx:latest** that meets the following requirements:

* (T104 / 1 pts) It is run in **detached** (background)mode
* (T105 / 1 pts) It is named **exam-web**
* (T106 / 1 pts) Port **80** of the container is forwarded to port **8080** on the **CNT** machine
* (T107 / 2 pts) The default web page (or the whole folder) in the container is mounted from a local folder (**~/html**) containing **index.html** with your **SoftUni ID** (username)

#### Storage [16 pts]

Demonstrate knowledge and readiness to work with storage related technologies (mostly working on **STR**)

Create a simple software **RAID** with **MD**:

* (T201 / 2 pts) Utilize the two extra drives (each 1 GB) to create a **RAID0** device **/dev/md0** with **128K** chunk
* (T202 / 1 pts) Make the configuration persistent (alter the **/etc/mdadm.conf** file)
* (T203 / 1 pts) Create an **ETX4** file system on the **RAID** device
* (T204 / 2 pts) Mount it at **/storage/raid** and add a record in the **/etc/fstab**

Create an **iSCSI** storage configuration:

* (T205 / 1 pts) Install the necessary **iSCSI**-related packages on the **STR** machine
* (T206 / 1 pts) Create an **iSCSI** target pointing to a **1 GB** disk image file named **D1GB.img**
* (T207 / 1 pts) Register the **CNT** machine as an initiator for the disk
* (T208 / 2 pts) Create an **EXT4** file system on the **iSCSI** disk on the **CNT** machine and add an entry in the **/etc/fstab** to mount it to **/storage/iscsi**

Create a simple **Samba** share:

* (T209 / 1 pts) Install **Samba** on the **STR** machine
* (T210 / 2 pts) Create a public read-write share named **exam** pointing to the **/storage/samba** folder
* (T211 / 2 pts) Mount the share on the **CNT** machine at **/storage/share** and add a record to the **/etc/fstab**

#### Load Balancing [8 pts]

Demonstrate knowledge and readiness to work with load balancing solutions:

* (T301 / 2 pts) Install **HAProxy** on the **LBA** machine
* (T302 / 2 pts) Create a front-end configuration block to listen on port **8080**
* (T303 / 2 pts) Create a back-end configuration block that contains **WEB1** and **WEB2** with **roundrobin** algorithm
* (T304 / 2 pts) Make sure that when accessed the load balancer will redirect to the added virtual host (port 8080) on both web servers

#### Web Servers [11 pts]

Demonstrate knowledge and readiness to work with web servers (on both **WEB1** and **WEB2** machines):

* (T401 / 3 pts) Install **Apache HTTP Server**
* (T402 / 2 pts) Make sure that the service is started and is configured to start on boot
* (T403 / 1 pts) Change the default web page to a page, containing the name of the Linux distribution in use
* (T404 / 4 pts) Create a virtual host and set it to listen on port **8080**
* (T405 / 1 pts) Create index page for the virtual host, that shows your **SoftUni ID** (username) and the name of the host, for example ***ivan on WEB1***

#### Monitoring [13 pts]

Demonstrate knowledge and readiness to work with monitoring solutions:

* (T501 / 3 pts) Install **Nagios** on the **MON** machine and make sure it is started and set to start on boot
* (T502 / 1 pts) Create a **nagiosadmin** user with password set to **LSAA-Exam**
* (T503 / 2 pts) Create a file **/etc/nagios/objects/exam-xxx.cfg** for every machine except the **MON**, where **xxx** is the name of the machine. For example, **exam-cnt.cfg**, **exam-str.cfg**, etc. Include those files in the main configuration file
* (T504 / 5 pts) Adjust configurations made in ***T503*** to include the nodes (all except **MON**) for monitoring by adding their object definitions
* (T505 / 2 pts) Setup monitoring of the **HTTP** service running on **WEB1** and **WEB2** in the appropriate configuration files created earlier (**exam-web1.cfg** and **exam-web2.cfg**)

## Sample Solution

Okay, for starters we must read the tasks and prepare our strategy how we will tackle them

Let’s imagine, that we came up with this:

### Web Servers (WEB1 and WEB2)

Log in to **WEB1** with the provided credentials

Install the required packages

**sudo apt-get update**

**sudo apt-get install apache2**

Create a custom **index.html** page

**echo 'Ubuntu 21.04' | sudo tee /var/www/html/index.html**

Create a virtual host configuration **/etc/apache2/sites-available/vhost-exam.conf**

**sudo vi /etc/apache2/sites-available/vhost-exam.conf**

With the following content

**Listen 8080**

**<VirtualHost \*:8080>**

**DocumentRoot /var/www/exam**

**ServerName <server-fqdn-name>**

**</VirtualHost>**

Save and close the file

Create the target folder

**sudo mkdir /var/www/exam**

Test the configuration

**apachectl configtest**

Enable the site

**sudo a2ensite vhost-exam.conf**

Restart the service

**sudo systemctl restart apache2**

Create the custom index page

**echo '@student on <server-name>' | sudo tee /var/www/exam/index.html**

Test the two sites – the default one and the virtual host-based one locally

**curl http://localhost**

**curl http://localhost:8080**

Open a port or service (or in this case – both) in the firewall if running

**sudo ufw allow Apache**

**sudo ufw allow 8080/tcp**

**sudo ufw reload**

Repeat the steps on **WEB2** as well

### Load Balancing (LBA)

Log in to **LBA** with the provided credentials

Install the required packages

**sudo apt-get update**

**sudo apt-get install haproxy**

Let’s adjust the configuration file **/etc/haproxy/haproxy.cfg** to match our needs

**sudo vi /etc/haproxy/haproxy.cfg**

Add **frontend** and **backend** sections at the end of the file

Create a new **frontend** section like this

**frontend http-in**

**bind \*:8080**

**default\_backend web\_servers**

**option forwardfor**

Create a new **backend** section like this

**backend web\_servers**

**balance roundrobin**

**server web1 <ip-address-web1>:8080 check**

**server web2 <ip-address-web2>:8080 check**

Save and close the file

Restart the **haproxy** service

**sudo systemctl restart haproxy**

Open the appropriate port in the firewall if running

**sudo ufw allow 8080/tcp**

**sudo ufw reload**

Test locally the result

**curl http://localhost:8080**

### Storage (STR)

Log in to **STR** with the provided credentials

#### Software RAID

Check the available disks

**lsblk**

Prepare partitions on both extra (**sdb** and **sdc**) disks

**sudo parted -s /dev/sdb -- mklabel msdos mkpart primary 2048s -0m set 1 raid on**

**sudo parted -s /dev/sdc -- mklabel msdos mkpart primary 2048s -0m set 1 raid on**

Let’s make sure that we have the necessary tools installed

**sudo apt-get update**

**sudo apt-get install mdadm**

Now, we are ready to create the required **RAID0** array

Use the following command to create a new array with **128k** chunks

**sudo mdadm --create /dev/md0 --level 0 --chunk 128 --raid-devices 2 /dev/sd{b,c}1**

Make the configuration persistent with

**sudo mdadm --detail --brief /dev/md0 | sudo tee -a /etc/mdadm.conf**

Let’s create the required file system

**sudo mkfs.ext4 /dev/md0**

Create the mount point

**sudo mkdir -p /storage/raid**

Open the **/etc/fstab** file

**sudo vi /etc/fstab**

And append the following line to the end

**/dev/md0 /storage/raid ext4 defaults 0 0**

Of course, we may use the **UUID** way. It is up to us

Test that everything with the configuration is okay and the file system is mounted correctly

**sudo mount -av**

Update the **initramfs**

**sudo update-initramfs -u**

If you like, reboot the system to check if the array will mount on boot

#### iSCSI

Let’s continue the same machine with the **iSCSI** related tasks

Install the required packages

**sudo apt-get update**

**sudo apt-get install targetcli-fb**

Create a folder to store the **iSCSI** disk files

**sudo mkdir /var/lib/iscsi-images**

Start the administration tool

**sudo targetcli**

Switch to the **fileio** backend

**cd backstores/fileio**

Create an **iSCSI** disk

**create D1 /var/lib/iscsi-images/D1GB.img 1G**

Switch to the **iscsi** functions

**cd /iscsi**

Define a new target

**create iqn.2021-09.lab.lsaa:str.tgt1**

Enter the target

**cd iqn.2021-09.lab.lsaa:str.tgt1/tpg1/luns**

Create a **LUN** using the disk created earlier

**create /backstores/fileio/D1**

Adjust the access to the resource

**cd ../acls**

Register the initiator (**CNT**)

**create iqn.2021-09.lab.lsaa:cnt.init1**

Enter the record

**cd iqn.2021-09.lab.lsaa:cnt.init1/**

Set user and password

**set auth userid=demo**

**set auth password=demo**

Exit the administrative tool

**exit**

Adjust the firewall if running

**sudo ufw allow 3260/tcp**

**sudo ufw reload**

Enable and start the service

**sudo systemctl enable --now rtslib-fb-targetctl.service**

#### Samba

Now, let’s continue with the **Samba** related tasks

Install the required packages

**sudo apt-get update**

**sudo apt-get install samba smbclient**

Now, we can check if the services are running

**systemctl status nmbd smbd**

Add firewall exception for **Samba** if the firewall is running

**sudo ufw allow samba**

**sudo ufw reload**

Prepare the folder to be shared

**sudo mkdir -p /storage/samba**

Set the ownership of the folder

**sudo chown nobody:nogroup /storage/samba**

Set the permissions

**sudo chmod 1777 /storage/samba**

Now, open the main configuration file (**/etc/samba/smb.conf**) for **Samba**

**sudo vi /etc/samba/smb.conf**

Add the following at the end

**[exam]**

**comment = Exam share**

**path = /storage/samba**

**writable = yes**

**browseable = yes**

**public = yes**

Save and close the file

Execute the following to test the configuration changes

**sudo testparm**

Restart the **Samba** services

**sudo systemctl restart smbd nmbd**

Let’s check for the available shares locally

**sudo smbclient -L //localhost**

When prompted for the **root** password just hit **Enter**

Now, let’s connect to our public share locally

**sudo smbclient -N //localhost/exam**

We can test with a few commands and finally quit the share with **quit**

We are done here, now must either switch to the client part or start another set of tasks

Let’s finish first the client part

### Storage on the Client (CNT)

Log in to **CNT** with the provided credentials

#### iSCSI

Let’s finalize the **iSCSI** related part first

Install the initiator package

**sudo apt-get update**

**sudo apt-get install open-iscsi**

Open the initiator configuration file for editing

**sudo vi /etc/iscsi/initiatorname.iscsi**

Set the name to match to your situation, for example **iqn.2021-09.lab.lsaa:cnt.init1**

Save and close the file

Adjust the authentication settings in **/etc/iscsi/iscsid.conf** file

**sudo vi /etc/iscsi/iscsid.conf**

Change the mode on line 46 to **automatic**

Uncomment **node.session.auth.authmethod = CHAP** (line 59)

Uncomment and adjust **node.session.auth.username** and **node.session.auth.password** (lines 70 and 71)

Save and close

Initiate a target discovery with

**sudo iscsiadm -m discovery -t sendtargets -p str**

Confirm what we have discovered

**sudo iscsiadm -m node -o show**

Login to the target

**sudo iscsiadm -m node --login**

Confirm the established session

**sudo iscsiadm -m session -o show**

We can check the available block devices either with either

**lsblk**

Or

**cat /proc/partitions**

Let’s create a partition on the **sdb** device

**sudo parted -s /dev/sdb -- mklabel msdos mkpart primary 16384s -0m**

Create a filesystem

**sudo mkfs.ext4 /dev/sdb1**

Prepare the mount point

**sudo mkdir -p /storage/iscsi**

Get the **UUID** of the file system

**sudo blkid /dev/sdb1**

Then open the **/etc/fstab** file

**sudo vi /etc/fstab**

And append to the end the following line

**UUID="<copied-value>" /storage/iscsi ext4 \_netdev 0 0**

Save and close the file

Test that everything with the configuration is okay and the file system is mounted correctly

**sudo mount -av**

If you like, reboot the system to check if the filesystem on the **iSCSI** disk will mount on boot

#### Samba

Now, let’s tackle the **Samba** related task

First, install the required packages

**sudo apt-get update**

**sudo apt-get install smbclient cifs-utils**

Check what is provided by the server

**smbclient -NL //str**

Try to connect to the **exam** share

**smbclient -N //str/exam**

List the contents with **ls**

Try to create a new folder named **client** with

**mkdir client**

And list again with **ls**

Finally, quit the **Samba** client with **quit**

Prepare the mountpoint

**sudo mkdir -p /storage/share**

Now, that we know the share is working, let’s try to mount it

**sudo mount -o guest,noperm //str/exam /storage/share**

We can verify the successful mount with either **df -hT** or **mount -l**

Unmount it

**sudo umount /storage/share**

Open the **/etc/fstab** file for editing

**sudo vi /etc/fstab**

Append the following line to the end

**//str/exam /storage/share cifs guest,noperm 0 0**

Save and close the file

Test that everything with the configuration is okay and the file system is mounted correctly

**sudo mount -av**

If you like, reboot the system to check if the remote filesystem will mount on boot

### Containers (CNT)

Log in to **CNT** with the provided credentials

First, we must add **Docker’s** official **GPG** key

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -**

Then, we must add the official **Docker** repository

**sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"**

Now, install the **Docker**

**sudo apt-get install docker-ce docker-ce-cli containerd.io**

Add our user to the **docker** group

**sudo usermod -aG docker <username>**

Close and re-open the session

Start and enable the service if not already started

**sudo systemctl enable --now docker**

Check that **Docker CLI** is accessible and the **Docker** daemon reachable

Execute either one or both

**docker version**

**docker info**

Now, that we have **Docker** up and running, let’s execute required hello world container

**docker run hello-world**

We should see a message that our installation is correctly set up

Create a folder in our home folder to store the custom index file

**mkdir html**

And then create the custom index file

**echo '@student' > ~/html/index.html**

Now, start the container that fulfills the requirements

**docker container run -d --name exam-web -p 8080:80 -v ~/html:/usr/share/nginx/html:ro nginx**

Check that the container is indeed running

**docker container ls**

Check that the web page is reachable from the host

**curl http://localhost:8080**

Open the appropriate port in the firewall if running

**sudo ufw allow 8080/tcp**

**sudo ufw reload**

### Monitoring (MON)

Log in to **MON** with the provided credentials

Install the required packages

**sudo apt-get update**

**sudo apt-get install -y apache2 autoconf bc build-essential dc gawk gcc gettext libapache2-mod-php7.4 libc6 libgd-dev libmcrypt-dev libmysqlclient-dev libnet-snmp-perl libssl-dev make php snmp unzip wget**

Enter the **/tmp** folder

**cd /tmp**

Download the source

**wget -O nagioscore.tar.gz https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.6.tar.gz**

Extract it

**tar xzf nagioscore.tar.gz**

Enter the folder

**cd /tmp/nagioscore-nagios-4.4.6/**

Configure it

**sudo ./configure --with-httpd-conf=/etc/apache2/sites-enabled**

Compile it

**sudo make all**

Create the required groups and users

**sudo make install-groups-users**

**sudo usermod -a -G nagios www-data**

Install binaries

**sudo make install**

Install the service

**sudo make install-daemoninit**

Install and configure external command file

**sudo make install-commandmode**

Install sample configuration files

**sudo make install-config**

Install **Apache** configuration files and enable modules

**sudo make install-webconf**

**sudo a2enmod rewrite**

**sudo a2enmod cgi**

Configure the firewall

**sudo ufw allow Apache**

**sudo ufw reload**

Create **Nagios** admin user

**sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin**

Restart the **Apache** service

**sudo systemctl restart apache2.service**

Return to the **/tmp** folder

**cd /tmp**

Download the plugins source package

**wget --no-check-certificate -O nagios-plugins.tar.gz https://github.com/nagios-plugins/nagios-plugins/archive/release-2.3.3.tar.gz**

Extract it

**tar zxf nagios-plugins.tar.gz**

And enter the folder

**cd /tmp/nagios-plugins-release-2.3.3/**

Configure the build process

**sudo ./tools/setup**

**sudo ./configure**

Compile it

**sudo make**

Install the plugins

**sudo make install**

Start the **Nagios** service

**sudo systemctl start nagios.service**

And then open a browser tab on your machine and navigate to **http://A.B.C.D:20080/nagios** where the **A.B.C.D** should be substituted with the IP address you received for the exam environment

Check that all the standard items and functions are accessible and working

Return to the console session on the **MON** machine

Create the required configuration files (one for each machine without the **MON** machine)

**sudo touch /usr/local/nagios/etc/objects/exam-lba.cfg**

**sudo touch /usr/local/nagios/etc/objects/exam-web1.cfg**

**sudo touch /usr/local/nagios/etc/objects/exam-web2.cfg**

**sudo touch /usr/local/nagios/etc/objects/exam-cnt.cfg**

**sudo touch /usr/local/nagios/etc/objects/exam-str.cfg**

Next, edit the main configuration file to include them

**sudo vi /usr/local/nagios/etc/nagios.cfg**

Add the following lines

**cfg\_file=/usr/local/nagios/etc/objects/exam-lba.cfg**

**cfg\_file=/usr/local/nagios/etc/objects/exam-web1.cfg**

**cfg\_file=/usr/local/nagios/etc/objects/exam-web2.cfg**

**cfg\_file=/usr/local/nagios/etc/objects/exam-cnt.cfg**

**cfg\_file=/usr/local/nagios/etc/objects/exam-str.cfg**

Save and close the file

Now, it is time to add the host object definition in each of the above files

Open them one by one and add the following

**define host {**

**use linux-server**

**host\_name <host-name>**

**alias <host-alias>**

**address <host-ip-address>**

**}**

Where **<host-name>** is the **FQDN** of a machine, for example **lba.lsaa.lab**

**<host-alias>** is the short name of the machine, for example **LBA**

Do this for all the machines except the **MON** machine

Once done, test the **Nagios** configuration

**sudo nagios -v /usr/local/nagios/etc/nagios.cfg**

And if there are not any, we can restart **nagios** service

**sudo systemctl restart nagios**

And return to the browser to check the result

We should see our newly added hosts there

The only task that remains is to add monitoring for the **HTTP** service on the two web servers

Let us first add a check for the web server running on **WEB1**

Open the **exam-web1.cfg** file for editing

**sudo vi /usr/local/nagios/etc/objects/exam-web1.cfg**

And add the following set of lines

**define service {**

**use generic-service**

**host\_name web1.lsaa.lab**

**service\_description HTTP**

**check\_command check\_http!$HOSTADDRESS$**

**}**

Save and close the file

Repeat the same for the configuration file of the **WEB2** machine

Then test **nagios** configuration

**sudo nagios -v /usr/local/nagios/etc/nagios.cfg**

Restart the service

**sudo systemctl restart nagios**

Return to the browser and check again

Congratulations, you made it! You finished the exam successfully! 😊