# Practice M4: Network Services (CentOS)

This practice assumes that you are working in an on-premise environment

All tasks can be achieved under different configurations (host OS and/or virtualization solution) with the appropriate adjustments

This practice is oriented towards **CentOS 8.4** and **Fedora Server 34** and other similar distributions

The infrastructure will vary during the practice but in its most complete stage will include up to three machines:

Chart, waterfall chart

Description automatically generated

## Part 1: Web Servers. Printing Services

For this part we will need an infrastructure with all three machines

Machines can be with or without graphical environment

Network settings shown on the picture reflect the ones, used during the demonstration. You should adjust them according to your setup

### Apache

Let us log on to the **M1** machine with a regular user account

#### Installation and Configuration

The installation of **Apache** is straight forward. Execute

**sudo dnf install httpd**

*Basically, we can also install the* ***tree*** *command to explore the* ***/etc/httpd*** *hierarchy*

Once, it is installed, we can turn off the default (welcome) web page

We can remove the **/etc/httpd/conf.d/welcome.conf** file but it will be recreated during an upgrade

Instead, we will comment out its content

**sudo sed -i 's/^/#/g' /etc/httpd/conf.d/welcome.conf**

Now, we can modify a little bit the main configuration file **/etc/httpd/conf/httpd.conf**

**sudo vi /etc/httpd/conf/httpd.conf**

(line 89) Set the **ServerAdmin** for example to **root@lsaa.lab**

(line 98) Set the **ServerName** for example to **m1.lsaa.lab:80**

(line 147) Change **Options** to **FollowSymLinks** *(remove the Indexes directive)*

(line 167) And add **index.php** to **DirectoryIndex**

Save and close the file

More information on the core **Apache** features can be found here: <https://httpd.apache.org/docs/2.4/mod/core.html>

Check the configuration with

**apachectl configtest**

Enable and start the **httpd** service

**sudo systemctl enable --now httpd**

Check the status of the **httpd** service

**systemctl status httpd**

Don’t forget to open the appropriate port in the firewall if running

**sudo firewall-cmd --add-service http --permanent**

**sudo firewall-cmd --reload**

Set a custom **index.html** file

**echo '<h1>Hello from M1</h1>' | sudo tee /var/www/html/index.html**

And finally, test it with

**curl http://localhost**

#### Virtual Hosts

Quite often we would want to have multiple sites on one server. One way to achieve this is to use **virtual hosts**

##### Preparation

Create a file **/etc/httpd/conf.d/main.conf** with the following content

**<VirtualHost \*:80>**

**DocumentRoot /var/www/html**

**ServerName m1.lsaa.lab**

**</VirtualHost>**

Save and close the file

##### Virtual Hosts (by port)

Create a file **/etc/httpd/conf.d/vhost-port.conf** with the following content

**Listen 8080**

**<VirtualHost \*:8080>**

**DocumentRoot /var/www/vhost-port**

**ServerName m1.lsaa.lab**

**</VirtualHost>**

Save and close the file

The **Listen 8080** instruction can be placed in the main configuration file instead in this one

##### Virtual Hosts (by name)

Create a file **/etc/httpd/conf.d/vhost-name.conf** with the following content

**<VirtualHost \*:80>**

**DocumentRoot /var/www/vhost-name**

**ServerName www.demo.lab**

**ServerAdmin admin@demo.lab**

**ErrorLog logs/vhost-name-error.log**

**CustomLog logs/vhost-name-access.log combined**

**</VirtualHost>**

Save and close the file

##### Finalization

Create the corresponding **DocumentRoot** folders

**sudo mkdir /var/www/vhost-{name,port}**

Crete two new **index.html** files

**echo '<h1>Hello from vhost by port</h1>' | sudo tee /var/www/vhost-port/index.html**

And

**echo '<h1>Hello from vhost by name</h1>' | sudo tee /var/www/vhost-name/index.html**

Test the new configuration with

**apachectl configtest**

Restart the **httpd** service

**sudo systemctl restart httpd**

Open port **8080/tcp** in the firewall

**sudo firewall-cmd --add-port 8080/tcp --permanent**

**sudo firewall-cmd --reload**

Add new record in the **/etc/hosts** file

**echo '<m1-ip> www.demo.lab www' | sudo tee -a /etc/hosts**

Finally, test both virtual hosts with

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

**curl http://www.demo.lab**

More information and samples about **Apache** virtual hosts can be found here: <https://httpd.apache.org/docs/2.4/vhosts/examples.html>

#### TLS/SSL

We can use certificates issued from a trusted certificate authority or create self-signed certificate

As this is a demo, let us create a self-signed certificate

##### Preparation

Install the necessary packages with

**sudo dnf install mod\_ssl openssl**

Generate the private key

**openssl genrsa -out ca.key 2048**

Create a certificate signing request (CSR)

**openssl req -new -key ca.key -out ca.csr**

Generate the self-signed certificate

**openssl x509 -req -days 365 -in ca.csr -signkey ca.key -out ca.crt**

We can we the result with

**openssl x509 -text -in ca.crt**

Copy the files to the appropriate folders

**sudo cp ca.crt /etc/pki/tls/certs/ca.crt**

**sudo cp ca.key /etc/pki/tls/private/ca.key**

**sudo cp ca.csr /etc/pki/tls/private/ca.csr**

##### Apache settings

Open the **/etc/httpd/conf.d/ssl.conf** file and navigate to the section of interest with

**sudo vi +/SSLCertificateFile /etc/httpd/conf.d/ssl.conf**

Change the **SSLCertificateFile** and **SSLCertificateKeyFile** to match our path *(change the name to* ***ca.key*** *or* ***ca.crt****)*

Save and close the file

Restart the **httpd** service

**sudo systemctl restart httpd**

Ask for the status

**systemctl status httpd**

You can see that **Apache** is listening on port **443** as well

Test the default site with

**curl** [**https://localhost**](https://localhost)

Because the certificate is self-signed you will see an error. Try again with

**curl -k** [**https://localhost**](https://localhost)

Now, you should see our default site

##### Final touches

We must open the appropriate port in the firewall

**sudo firewall-cmd --add-service https --permanent**

**sudo firewall-cmd --reload**

Should we want to be automatically redirected to **https** when visiting **http**, we can modify the virtual host configuration for our default site

**sudo vi /etc/httpd/conf.d/main.conf**

Add the following three lines just before the closing **</VirtualHost>**

**RewriteEngine on**

**RewriteCond %{HTTPS} off**

**RewriteRule ^(.\*)$ https://%{HTTP\_HOST}%{REQUEST\_URI} [R=301,L]**

Save and close the file

Restart the **httpd** service

**sudo systemctl restart httpd**

Open a browser tab on the host and navigate to **http://<m1-ip>/**

A warning should appear, accept it. Now, you should see our default site

More information about the **mod\_rewrite** module of **Apache** can be found here: <https://httpd.apache.org/docs/2.4/mod/mod_rewrite.html>

#### PHP

Let us install the necessary packages. This time we will utilize the new **module** functionality

**sudo dnf module install php**

Once the installation is complete, we can check with

**php -v**

Then we can restart the **httpd** service

**sudo systemctl restart httpd**

Remove the existing **index.html** file

**sudo rm /var/www/html/index.html**

Create a new **index.php** file with

**echo '<?php phpinfo(); ?>' | sudo tee /var/www/html/index.php**

Do test with either **curl** locally or a with browser tab on the host

#### Reverse Proxy

##### Preparation

Log on to the **M2** machine and install **httpd**

**sudo dnf install httpd**

Then create a custom **index.html** page

**echo '<h1>Hello from M2</h1>' | sudo tee /var/www/html/index.html**

Don’t forget to enable and start the service

**sudo systemctl enable --now httpd**

And last, but not least, don’t forget to adjust the firewall

**sudo firewall-cmd --add-service http --permanent**

**sudo firewall-cmd --reload**

##### Configuration

Return to the **M1** machine

**mod\_proxy** is installed as part of the **httpd** module

We can check the content of the module with

**dnf module info httpd**

And we can check the **Apache** configuration as well with

**grep mod\_proxy /etc/httpd/conf.modules.d/\***

Now, let us create a configuration for the reverse proxy module

**sudo vi /etc/httpd/conf.d/reverse-proxy.conf**

Enter the following

**<IfModule mod\_proxy.c>**

**ProxyRequests Off**

**<Proxy \*>**

**Require all granted**

**</Proxy>**

**ProxyPass / http://m2.lsaa.lab/**

**ProxyPassReverse / http://m2.lsaa.lab/**

**</IfModule>**

Save and close the file

Test the configuration

**apachectl configtest**

Restart the **httpd** service

**sudo systemctl restart httpd**

Test by opening a browser tab on the host and navigating to **https://<m1-ip>/**

You will see an error

Return on **M1** and adjust the following **SELinux Boolean**

**sudo setsebool -P httpd\_can\_network\_connect on**

Refresh the browser tab, now it must work as expected

You should see the index page of **M2**

More information about the **mod\_proxy** module of **Apache** can be found here: <https://httpd.apache.org/docs/2.4/mod/mod_proxy.html>

#### Load Balancing

##### Preparation

Log on to the **M3** machine and install **httpd**

**sudo dnf install httpd**

Then create a custom **index.html** page

**echo '<h1>Hello from M3</h1>' | sudo tee /var/www/html/index.html**

Don’t forget to enable and start the service

**sudo systemctl enable --now httpd**

And last, but not least, don’t forget to adjust the firewall

**sudo firewall-cmd --add-service http --permanent**

**sudo firewall-cmd --reload**

##### Configuration

We are one step away from turning our setup into a fully functional load balancing solution

Return on **M1**

Open the proxy module configuration file

**sudo vi /etc/httpd/conf.d/reverse-proxy.conf**

And modify the file by deleting the **ProxyPass** and **ProxyPassReverse** lines

Then enter the following

**ProxyPass / balancer://demo/**

**ProxyPassReverse / balancer://demo/**

**<Proxy balancer://demo>**

**BalancerMember http://m2.lsaa.lab**

**BalancerMember http://m3.lsaa.lab**

**ProxySet lbmethod=bytraffic**

**</Proxy>**

Save and close the file

Check the configuration

**apachectl configtest**

Restart the **httpd** service

**sudo systemctl restart httpd**

Test by opening a browser tab on the host and navigating to **https://<m1-ip>/**

Refresh a few times. Everything should work as expected

More information about the **mod\_proxy** module of **Apache** can be found here: <https://httpd.apache.org/docs/2.4/mod/mod_proxy.html>

And about the load balancing extension, here:

<https://httpd.apache.org/docs/2.4/mod/mod_proxy_balancer.html>

### NGINX

We can continue with the same infrastructure, just reset the machines. Alternatively, spin up a new set

Let us log on to the **M1** machine with a regular user account

#### Installation and Configuration

The installation of **NGINX** is straight forward, execute

**sudo dnf install nginx**

Once, it is installed, we can change the default (welcome) web page

Execute the following to change it

**echo '<h1>Hello from M1</h1>' | sudo tee /usr/share/nginx/html/index.html**

Now, we can modify a little bit the main configuration file **/etc/nginx/nginx.conf**

**sudo vi /etc/nginx/nginx.conf**

(line 41) Set the **server\_name** for example to **m1.lsaa.lab**

Save and close the file

Test the **NGINX** configuration with

**sudo nginx -t**

Enable and start the **nginx** service

**sudo systemctl enable --now nginx**

Check the status of the **nginx** service

**systemctl status nginx**

Don’t forget to open the appropriate port in the firewall if running

**sudo firewall-cmd --add-service http --permanent**

**sudo firewall-cmd --reload**

And finally, test with

**curl http://localhost**

#### Virtual Hosts

Quite often we would want to have multiple sites on one server. One way to achieve this is to use **virtual hosts**

##### Virtual Hosts (by port)

Create a file **/etc/nginx/conf.d/vhost-port.conf** with the following content

**server {**

**listen 8080;**

**location / {**

**root /usr/share/nginx/vhost-port;**

**index index.html;**

**}**

**}**

Save and close the file

##### Virtual Hosts (by name)

Create a file **/etc/nginx/conf.d/vhost-name.conf** with the following content

**server {**

**listen 80;**

**server\_name www.demo.lab;**

**location / {**

**root /usr/share/nginx/vhost-name;**

**index index.html;**

**}**

**}**

Save and close the file

##### Finalization

Create the corresponding **root** folders

**sudo mkdir /usr/share/nginx/vhost-{name,port}**

Crete two new **index.html** files

**echo '<h1>Hello from M1 (vhost by port)</h1>' | sudo tee /usr/share/nginx/vhost-port/index.html**

And

**echo '<h1>Hello from M1 (vhost by name)</h1>' | sudo tee /usr/share/nginx/vhost-name/index.html**

Test the new configuration with

**sudo nginx -t**

Restart the **nginx** service

**sudo systemctl restart nginx**

Open port **8080/tcp** in the firewall

**sudo firewall-cmd --add-port 8080/tcp --permanent**

**sudo firewall-cmd --reload**

Add new record in the **/etc/hosts** file

**echo '<m1-ip> www.demo.lab www' | sudo tee -a /etc/hosts**

Finally, test both virtual hosts with

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

**curl http://www.demo.lab**

#### TLS/SSL

We can use certificates issued from a trusted certificate authority or create self-signed certificate

As this is a demo, let us create a self-signed certificate

##### Preparation

Repeat the same steps as with **Apache**

Install the necessary packages with

**sudo dnf install openssl**

Generate the private key

**openssl genrsa -out ca.key 2048**

Create a certificate signing request (CSR)

**openssl req -new -key ca.key -out ca.csr**

Generate the self-signed certificate

**openssl x509 -req -days 365 -in ca.csr -signkey ca.key -out ca.crt**

We can we the result with

**openssl x509 -text -in ca.crt**

Copy the files to the appropriate folders

**sudo cp ca.crt /etc/pki/tls/certs/ca.crt**

**sudo cp ca.key /etc/pki/tls/private/ca.key**

**sudo cp ca.csr /etc/pki/tls/private/ca.csr**

##### NGINX settings

Open the **/etc/nginx/nginx.conf** file

**sudo vi +59 /etc/nginx/nginx.conf**

Uncomment lines between **60** and **87**

You can do it automatically with the following

**.,$s/^#//g**

Then adjust the **server\_name**, **ssl\_certificate**, and **ssl\_certificate\_key** to match your settings

Save and close the file

Test the new configuration with

**sudo nginx -t**

Restart the **nginx** service

**sudo systemctl restart nginx**

Test the default site with

**curl -k** [**https://localhost**](https://localhost)

Now, you should see our default site

##### Final touches

We must open the appropriate port in the firewall

**sudo firewall-cmd --add-service https --permanent**

**sudo firewall-cmd --reload**

Should we want to be automatically redirected to **https** when visiting **http**, we can modify the virtual host configuration for our default site

**sudo vi /etc/nginx/nginx.conf**

Add the following line between **listen** (line 40) and **server\_name** (line 41)

**return 301 https://$host$request\_uri;**

Save and close the file

Test the new configuration with

**sudo nginx -t**

Restart the **nginx** service

**sudo systemctl restart nginx**

Open a browser tab on the host and navigate to **http://<m1-ip>/**

A warning should appear, accept it. Now, you should see our default site

#### PHP

Let us install the necessary packages

**sudo dnf module install php**

Once the installation is complete, we can check with

**php -v**

Then we can restart the **nginx** service

**sudo systemctl restart nginx**

Create a new **index.php** file with

**echo '<?php phpinfo(); ?>' | sudo tee /usr/share/nginx/html/index.php**

Do test with either **curl** locally or a with browser tab on the host

#### Reverse Proxy

##### Preparation

Log on to the **M2** machine and install **nginx**

**sudo dnf install nginx**

Change the **/etc/nginx/nginx.conf** file by adding the following after the **root** directive (line 42)

**set\_real\_ip\_from <machines-network>/24; 🡨 for example 192.168.81.0/24**

**real\_ip\_header X-Forwarded-For;**

And adjusting the **server\_name** line

Save and close the file

Test the new configuration with

**sudo nginx -t**

Start the **nginx** service

**sudo systemctl enable --now nginx**

Then create a custom **index.html** page

**echo '<h1>Hello from M2</h1>' | sudo tee /usr/share/nginx/html/index.html**

Don’t forget to adjust the firewall

**sudo firewall-cmd --add-service http --permanent**

**sudo firewall-cmd --reload**

##### Configuration

Return to the **M1** machine

Open the main configuration file for editing

**sudo vi /etc/nginx/nginx.conf**

Enter the following lines just above the **location** (line 48)

**proxy\_redirect off;**

**proxy\_set\_header X-Real-IP $remote\_addr;**

**proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;**

**proxy\_set\_header Host $http\_host;**

Add the following to the **location** block

**proxy\_pass** [**http://m2.lsaa.lab/**](http://m2.lsaa.lab/)**;**

Repeat the above steps for the **TLS** section as well

Save and close the file

Test the configuration

**sudo nginx -t**

Restart the **nginx** service

**sudo systemctl restart nginx**

Adjust the following **SELinux Boolean**

**sudo setsebool -P httpd\_can\_network\_connect on**

Open a browser tab on the host and navigate to **https://<m1-ip>/**

You should see the index page of **M2**

#### Load Balancing

##### Preparation

Log on to the **M3** machine and install **nginx**

**sudo dnf install nginx**

Change the **/etc/nginx/nginx.conf** file by adding the following after the **root** directive (line 42)

**set\_real\_ip\_from <machines-network>/24; 🡨 for example 192.168.81.0/24**

**real\_ip\_header X-Forwarded-For;**

And adjusting the **server\_name** line

Save and close the file

Test the new configuration with

**sudo nginx -t**

Start the **nginx** service

**sudo systemctl enable --now nginx**

Then create a custom **index.html** page

**echo '<h1>Hello from M3</h1>' | sudo tee /usr/share/nginx/html/index.html**

Don’t forget to adjust the firewall

**sudo firewall-cmd --add-service http --permanent**

**sudo firewall-cmd --reload**

##### Configuration

We are one step away from turning our setup into a fully functional load balancing solution

Return on the **M1** machine

Open the main configuration file for editing

**sudo vi /etc/nginx/nginx.conf**

Add the following lines in the top of **http** section

**upstream backend {**

**server m2.lsaa.lab;**

**server m3.lsaa.lab;**

**}**

Then change the **location** of both sections for plain http and TLS to

**proxy\_pass http://backend;**

Save and close the file

Check the configuration

**sudo nginx -t**

Restart the **nginx** service

**sudo systemctl restart nginx**

Test by opening a browser tab on the host and navigating to **https://<m1-ip>/**

Refresh a few times. Everything should work as expected

### CUPS

We must either reset the existing infrastructure or prepare a new set of machines. In fact, we will need just one (M1)

#### Installation

Install the necessary packages with

**sudo dnf install cups**

Enable and start the **cups** service

**sudo systemctl enable --now cups**

Check its status

**systemctl status cups**

#### Configuration

Now, let us adjust a few settings in the **/etc/cups/cupsd.conf** configuration file

**sudo vi /etc/cups/cupsd.conf**

Change line 13 from **Listen localhost:631** to **Listen 0.0.0.0:631** to set CUPS listen on all interfaces

Add **Allow @LOCAL** to both **<Location />** and **<Location /admin>** to allow access from the local network

Save and close the file

Restart the **cups** service

**sudo systemctl restart cups**

Open the appropriate firewall port

**sudo firewall-cmd --add-port 631/tcp --permanent**

**sudo firewall-cmd --reload**

Now, you can open a browser tab on the host and navigate to **http://<m1-ip>:631**

#### Installing a PDF printer

Return on **M1**

We will need the **cups-pdf** package which is not present in **CentOS 8.x** repositories

So, we will install it using another approach

**sudo dnf install** [**https://download-ib01.fedoraproject.org/pub/epel/7/x86\_64/Packages/c/cups-pdf-2.6.1-7.el7.x86\_64.rpm**](https://download-ib01.fedoraproject.org/pub/epel/7/x86_64/Packages/c/cups-pdf-2.6.1-7.el7.x86_64.rpm)

Next, we can explore its configuration

**sudo vi /etc/cups/cups-pdf.conf**

Once done exploring, restart the **cups** service

**sudo systemctl restart cups**

Let us ask for the list of printers

**lpstat -p -d**

And now, that we have a printer, let us make it the default printer with

**lpoptions -d Cups-PDF**

#### Printing

We can either create a text file, or use on existing

In order to find one, we can execute

**find /usr/share/doc -name README**

We can pick up for example

**cat /usr/share/doc/sudo/README**

And print it with

**lp /usr/share/doc/sudo/README**

Then, if we list the content of our folder, we will see a **README.pdf** file

We can copy it on the host and open it

## Part 2: Directory Services

For this part we will need an infrastructure with just two machines

Machines can be with or without graphical environment

Network settings shown on the picture reflect the ones, used during the demonstration. You should adjust them according to your setup

### BIND

#### Caching Server

Log on to the **M1** machine

Start with installing the required packages

**sudo dnf install bind bind-utils**

Open the main configuration file

**sudo vi /etc/named.conf**

Add the following block above the **options** instruction

**acl trusted-clients {**

**localhost;**

**<machines-network>/24; 🡨 for example 192.168.81.0/24**

**}**

This will allow us to grant all stations in our internal network the right to query the DNS server

Next, change the interfaces the **bind** service listens on

For example, change the IPv4 **localhost** (**127.0.0.1**) to **any** (to listen on all interfaces)or to **none** (to stop listen)

Same applies for IPv6but instead of **127.0.0.1** the address is **::1**

Let us set both **listen-on** and **listen-on-v6** to **any**

Now, change the **allow-query** to **{ trusted-clients; }**

Save and close the file

Check that everything with the configuration is okay

**sudo named-checkconf**

It appears that have an error due to a missing semicolon

Open again the file and correct the error

Check the configuration again. Now all should be fine

It is a good idea to check the ownership of the file

**ls -l /etc/named.conf**

It must be owned by the **root** user and the **named** group

In the same manner, we must check the **SELinux** label for the file

**ls -Z /etc/named.conf**

The type part must be **named\_conf\_t**

Enable and start the service

**sudo systemctl enable --now named**

And check its status

**systemctl status named**

Additionally, the status can be checked with

**sudo rndc status**

Don’t forget to allow the DNS service in the firewall

**sudo firewall-cmd --add-service dns --permanent**

**sudo firewall-cmd --reload**

Now, log on to the **M2** machine

Install the required packages

**sudo dnf install bind-utils**

Set **M1** as a DNS server

Change local settings for the DNS server

**sudo nmcli conn modify ens192 ipv4.dns <m1-ip>**

**sudo nmcli conn down ens192; sudo nmcli conn up ens192**

*Please note that you may need to adjust the network interface name as well*

Let us look up information for a domain, for example **opensuse.org**

**dig opensuse.org**

We can see plenty of information

Notice the time it took to answer the query

Repeat the command once more

Now, the answer is returned fast, because of the cache

Let us try a reverse lookup query with one of the returned IP addresses

**dig -x 195.135.221.140**

#### Forwarding Server

Now, we can modify the settings of our server and turn it into a forwarding server

Return on **M1** machine

Open the main configuration file

**sudo vi /etc/named.conf**

Insert the following block after the **allow-query** instruction

**forwarders {**

**8.8.8.8;**

**8.8.4.4;**

**};**

**forward only;**

Save and close the file

Then, check it for errors

**sudo named-checkconf**

Execute the following to make **bind** reload its configuration

**sudo rndc reload**

And this one to flush the cache

**sudo rndc flush**

Switch to the **M2** machine and repeat the lookup queries

Even the first attempt resolves much faster now

#### Internal DNS server

Return on the **M1** machine

Open the main configuration file for editing

**sudo vi /etc/named.conf**

Position the cursor after the last **zone** block and type the following for the forward lookup zone

**zone "lsaa.lab" IN {**

**type master;**

**file "lsaa.lab.zone";**

**allow-update { none; };**

**};**

And then add the following for the reverse lookup zone

**zone "81.168.192.in-addr.arpa" IN {**

**type master;**

**file "81.168.192.zone";**

**allow-update { none; };**

**};**

Save and close the file

Check the configuration with

**sudo named-checkconf**

Create a new forward lookup zone file

**sudo vi /var/named/lsaa.lab.zone**

And enter the following

**$ORIGIN lsaa.lab.**

**$TTL 86400**

**@ IN SOA m1.lsaa.lab. root.lsaa.lab. (**

**2021092401 ; serial**

**3600 ; refresh in 1 hour**

**1800 ; retry in 30 minutes**

**604800 ; expires after 7 days**

**86400 ; minimum TTL of 1 day**

**)**

**IN NS m1.lsaa.lab.**

**IN A <m1-ip>**

**IN MX 10 m1.lsaa.lab.**

**m1 IN A <m1-ip>**

**m2 IN A <m2-ip>**

**client IN CNAME m2.lsaa.lab.**

Save and close the file

Ensure that the file permissions and ownership are as expected

**sudo chmod 640 /var/named/lsaa.lab.zone**

**sudo chown root:named /var/named/lsaa.lab.zone**

Check the zone with

**sudo named-checkzone lsaa.lab /var/named/lsaa.lab.zone**

Create a new reverse lookup zone file

**sudo vi /var/named/81.168.192.zone**

And enter the following

**$TTL 86400**

**81.168.192.in-addr.arpa. IN SOA m1.lsaa.lab. root.lsaa.lab. (**

**2021092401 ; serial**

**3600 ; refresh in 1 hour**

**1800 ; retry in 30 minutes**

**604800 ; expires after 7 days**

**86400 ; minimum TTL of 1 day**

**)**

**IN NS m1.lsaa.lab.**

**<last-part-ip-m1> IN PTR m1.lsaa.lab.**

**<last-part-ip-m2> IN PTR m2.lsaa.lab.**

Save and close the file

Check the zone

**sudo named-checkzone 81.168.192.in-addr.arpa /var/named/81.168.192.zone**

Adjust the permissions and ownership

**sudo chmod 640 /var/named/81.168.192.zone**

**sudo chown root:named /var/named/81.168.192.zone**

Restart the **named** service

**sudo systemctl restart named**

And check its status

**systemctl status named**

Change local settings for the DNS server

**sudo nmcli conn modify ens192 ipv4.dns <m1-ip>**

**sudo nmcli conn down ens192; sudo nmcli conn up ens192**

*Please note that you may need to adjust the network interface name as well*

Do a forward query for the **m2.lsaa.lab** machine

**dig m2.lsaa.lab**  
Then do a reverse lookup

**dig -x <m2-ip>**

You can switch to **M2** machine and test the same

### FreeIPA

Do not forget to reset the machines used during the previous part or spin a new set

Please note, that for this part, the **M1** machine should have at least 2 GB of RAM

Log on to the **M1** machine with a regular user account

We must make sure that our **/etc/hosts** file is updated and contains the FQDN of the machine

#### Installation

First, let us check the modules list

**dnf module list**

There is the **idm** module with multiple profiles. Let us check more about the profiles

**dnf module info idm --profile**

As, we want an all-in-one solution, we will install the **idm** with **dns** profile

**sudo dnf module install idm:DL1/dns**

Once the installation is complete, we can set the initial configuration

**sudo ipa-server-install --setup-dns**

Confirm the hostname by pressing **Enter**

Confirm the domain name

Confirm the **Kerberos Realm** name

Enter and confirm password for the **Directory Manager**

Enter and confirm the **IPA** server administrator (admin) password

Confirm that you want to configure **DNS** forwarders

Confirm the proposed forwarder server

If there are not any more forwarder server candidates, press **Enter** to continue

Confirm that you want to search for missing reverse zones

Confirm to add the proposed reverse zone

Confirm the zone name

If you do not have any preferred **NTP** server address or pool in mind, confirm that you will not provide one

Enter **yes** to confirm that you want to apply the settings

Once the installation is complete (it can take 10 minutes or more), we can adjust the firewall

**sudo firewall-cmd --add-service={freeipa-4,dns,ntp} --permanent**

**sudo firewall-cmd --reload**

Now, obtain a **Kerberos** ticket for the **admin** user

**kinit admin**

And list the available tickets

**klist**

#### Users and Groups

Let us create one user named **ivan**

**ipa user-add ivan --first=Ivan --last=Petkov --password**

We can use interactive mode, by executing just **ipa user-add**

Add second user

**ipa user-add mariana --first=Mariana --last=Petkova --password**

Let us check what commands are there

**ipa help**

Okay, and what topics are there

**ipa help topics**

Hm, let us see what commands are in topic **user**

**ipa help user**

To receive the list of users, execute

**ipa user-find**

Information for a user can be extracted with

**ipa user-find ivan**

Should we want to see the actual attribute names, we can add **--raw**

We can change the password of a user with

**ipa passwd ivan**

We can disable/enable user with

**ipa user-disable mariana**

Groups can be added with

**ipa group-add --desc='IT Dept.' it**

Let us add our two users to the group we just created

**ipa group-add-member --users={ivan,mariana} it**

We can list the available groups with

**ipa group-find**

Or check who is part of a group

**ipa group-show it**

Of course, we can delete and modify users and groups

#### Clients

First, we must add a **DNS** record for the client

Log on to the **M1** machine and execute

**ipa dnsrecord-add lsaa.lab m2 --a-rec <m2-ip>**

Return information for the zone

**ipa dnszone-show lsaa.lab**

Show zone entries

**ipa dnsrecord-find lsaa.lab**

Now, log on to the **M2** machine

Every client should have its time synchronized with the **IPA** server

First, we will install and configure **NTP** client

**sudo dnf install chrony ntpstat**

We can adjust configuration setting with

**sudo vi /etc/chrony.conf**

If you made any changes restart the **chronyd** service

We can check what is going on with

**chronyc sources**

**ntpstat**

You may need to allow clients to connect to the **NTP** of **M1** (by editing **/etc/chrony.conf**)

We are ready to move forward

Now, we must install the **FreeIPA** client module with

**sudo dnf module install idm:DL1/client**

Once the installation is complete, we must modify the local **DNS** settings. Our **FreeIPA** server must be the first (in our case the only) **DNS** server

Check the name of the connection

**nmcli conn show**

Modify the **DNS** setting for the connection, for example

**sudo nmcli conn modify ens192 ipv4.dns <m1-ip>**

Restart the connection

**sudo nmcli conn down ens192; sudo nmcli conn up ens192**

*Please note that you may need to adjust the network interface name as well*

Let us join our station to the **FreeIPA** server

**sudo ipa-client-install --server=m1.lsaa.lab --domain=lsaa.lab**

Answer **yes** to the first question

As our **NTP** client is already configured, we can answer with **no** on the second question

Confirm the configuration with **yes**

Enter **admin** and its password when asked for credentials

After a while, the process will finish, and our station will be joined to the **FreeIPA** server

Before we continue, let us check how the system identity and authentication sources are configured

**authselect current**

Check what features are available for **sssd**

**authselect list-features sssd**

Should we want home folders to be created automatically on first logon, we ca enable the corresponding feature

**sudo authselect enable-feature with-mkhomedir**

Enable and start the **oddjobd** service

**sudo systemctl enable --now oddjobd**

Close the session and start a new one with one of the users created earlier

**ssh ivan@<m2-ip>**

As this is the first login attempt, you will be prompted to change the password

Congratulations! You made it

#### Web Console

On the host machine open a browser tab and navigate to <https://m1.lsaa.lab/ipa/ui/>

You should have the appropriate record in your **hosts** file

Alternatively, if you have GUI installed on one of the VMs (for example, the one you joined to the Realm), you can open a browser tab there

Enter the **admin** credentials and log on

You can see the list of users we created earlier

There are plenty of functions here, take your time and explore the interface

More information can be found here: <https://www.freeipa.org/page/Main_Page>

## Part 3: Mail Services

For this part we will need an infrastructure with just two machines

Machines can be with or without graphical environment

Network settings shown on the picture reflect the ones, used during the demonstration. You should adjust them according to your setup

### Sendmail

Log on to the **M1** machine

#### Installation

Install the required package

**sudo dnf install postfix**

Once it is installed, we can explore what configuration files are there

**rpm -qc postfix**

In a similar way, we can see what executable utilities we have got

**rpm -ql postfix | grep bin**

Enable and start the **postfix** service

**sudo systemctl enable --now postfix**

Check its status

**systemctl status postfix**

Now, check open TCP ports and what is listening on them

**sudo ss -ltnp**

#### Configuration

Let us configure **postfix** to listen on all interfaces. First get current value and then change it

**postconf inet\_interfaces**

**sudo postconf -e "inet\_interfaces = all"**

Examine and set the **inet\_protocols** parameter

**postconf inet\_protocols**

**sudo postconf -e "inet\_protocols = ipv4"**

Examine and set if needed (it should be set already to the correct value) the **myhostname** parameter

**postconf myhostname**

**sudo postconf -e "myhostname = m1.lsaa.lab"**

Examine and set if needed (it should be set already to the correct value) the **mydomain** parameter

**postconf mydomain**

**sudo postconf -e "mydomain = lsaa.lab"**

Examine and set the **myorigin** parameter

**postconf myorigin**

**sudo postconf -e "myorigin = \$mydomain"**

Examine and change the **mydestination** parameter

**postconf mydestination**

**sudo postconf -e "mydestination = \$myhostname, localhost.\$mydomain, localhost, \$mydomain"**

All above changes could have been done with direct changes in **/etc/postfix/main.cf** file

We can list the available parameters and their default values with

**postconf -d**

We can check how many parameters are there

**postconf -d | wc -l**

Wow, we have more than 900 parameters

Alternatively, we can get the list of explicitly set parameters

**postconf -n**

We can check the configuration for any errors with

**sudo postfix check**

Restart the **postfix** service

**sudo systemctl restart postfix**

Alternatively, we can ask the daemon to reload its configuration with

**sudo systemctl reload postfix**

Check again the open port and listening processes with

**sudo ss -ltnp**

Open the **smtp** service in the firewall

**sudo firewall-cmd --add-service smtp --permanent**

**sudo firewall-cmd --reload**

#### Aliases

Let us explore the alias list

**cat /etc/aliases**

Now, add an alias redirecting mail from **root** to the current user

**sudo vi /etc/aliases**

Go to the end of the file and add

**root: <current-user>**

Save and close the file

Now, we must apply the changes by executing either the **sendmail** emulation command

**sudo newaliases**

Or the native one

**sudo postalias /etc/aliases**

And then restart the service

**sudo systemctl restart postfix**

#### Message and Mailbox Limits

Message size limit we can check and change via the following parameter

**postconf -p message\_size\_limit**

And the mailbox size limit, via the following

**postconf -p mailbox\_size\_limit**

The second should be bigger than the first one

#### Testing Locally

Install a command line mail client

**sudo dnf install mailx**

Let us create a simple message with subject **test** for our current user

**mailx -s test <current-user>**

Then enter some text, for example

**This is a test message**

Once done, on a new line enter the dot (**.**) symbol and hit **Enter** or press **Ctrl+D**

We can compose and send another one with

**mailx -s 'Test message' postmaster**

Enter something for the body of the message and press **Ctrl+D** to send it

Then enter the mail program with

**mailx**

We can see a list of the messages in our mailbox

In order to read a message, we must enter its number and hit **Enter**

If you hit **Enter** again, the next message will be displayed

Should you want to see the list of messages, execute **header**

List of all commands can be invoked with **list**

In order to leave the application, you must execute **quit**

We can explore the log as well

**sudo tail /var/log/maillog**

#### Sending Between Servers

Log on to **M2** machine

Install **postfix** and do a basic configuration

**sudo dnf install postfix mailx**

**sudo postconf -e "inet\_interfaces = all"**

**sudo postconf -e "inet\_protocols = ipv4"**

**sudo postconf -e "myorigin = \$mydomain"**

**sudo postconf -e "mydestination = \$myhostname, localhost.\$mydomain, localhost, \$mydomain"**

**sudo systemctl enable --now postfix**

**sudo firewall-cmd --add-service smtp --permanent**

**sudo firewall-cmd --reload**

Now, we are ready to try to send an email from **M2** to **M1**

**mailx -s 'From M2 to M1 #1' user@m1.lsaa.lab**

Enter some body text and send the message

Ask for the message queue with

**postqueue -p**

It appears to be empty. After a while we will receive a notification that our message was not delivered

Let us check the log file

**sudo cat /var/log/maillog**

It appears that there is a name resolution problem

In our case this is due to the way we have set up the name resolution through **/etc/hosts** file

We can ask for the list of all related settings with

**postconf -p | grep lookup**

We must change the following set of parameters

**sudo postconf -e 'ignore\_mx\_lookup\_error = yes'**

**sudo postconf -e 'lmtp\_host\_lookup = native'**

**sudo postconf -e 'smtp\_host\_lookup = native'**

Now, restart the **postfix** service

**sudo systemctl restart postfix**

And try again to send a message to the **M1** machine

**mailx -s 'From M2 to M1 #2' user@m1.lsaa.lab**

Check the log again

**sudo tail /var/log/maillog**

It appears that the message was delivered

Log on to the **M1** machine and check with **mailx**

Indeed, the message has arrived

#### TLS with Postfix

Return to **M2** machine and create a self-signed certificate

**openssl req -x509 -nodes -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -subj "/CN=m2.lsaa.lab"**

Alternatively, we can use the ones generated during **postfix** installation (go with this one)

We can explore the certificate with

**openssl x509 -in /etc/pki/tls/certs/postfix.pem -text**

Now, let us check a set of parameters and their values

**postconf -p smtpd\_use\_tls**

**postconf -p smtpd\_enforce\_tls**

**postconf -p smtpd\_tls\_loglevel**

**postconf -p smtpd\_tls\_key\_file**

**postconf -p smtpd\_tls\_cert\_file**

If **smptd\_use\_tls** is set to **no**, then change it to **yes**. In addition, set the **smtpd\_tls\_loglevel** to **1** and restart the service

**sudo postconf -e 'smtpd\_use\_tls = yes'**

**sudo postconf -e 'smtpd\_tls\_loglevel = 1'**

**sudo systemctl restart postfix**

Now, let us test the **TLS/SSL** communication to our **SMTP** server with

**openssl s\_client -connect m2.lsaa.lab:25 -starttls smtp**

We can browse the information here, in the client

We can type the following to receive information from the server

**ehlo m2.lsaa.lab**

Let us exit with **quit** and check the logs

**sudo tail /var/log/maillog**

Here, we can see that an anonymous **TLS** connection had been established

It appears that we have **TLS** working out of the box without significant changes

Furthermore, now we are using just one port - **25/tcp** for both plain and **TLS** connections. We can check

**sudo ss -ltnp**

Let us see how we can switch or expose the **TLS** service on a separate port

Open for editing the **/etc/postfix/master.cf** file

**sudo vi /etc/postfix/master.cf**

And find and uncomment the **smtps** section (lines 29 to 39)

Save and close the file

Check the configuration with

**sudo postfix check**

Add the following to the end of the **/etc/postfix/main.cf** file especially if they are not defined (you see warnings when use **postconf -n** for example)

**smtpd\_restriction\_classes = mua\_sender\_restrictions, mua\_client\_restrictions, mua\_helo\_restrictions**

**mua\_client\_restrictions = permit\_sasl\_authenticated, reject**

**mua\_sender\_restrictions = permit\_sasl\_authenticated, reject**

**mua\_helo\_restrictions = permit\_mynetworks, reject\_non\_fqdn\_hostname, reject\_invalid\_hostname, permit**

Add the following to enable the **SASL** (**Simple Authentication and Security Layer**) authentication

**smtpd\_sasl\_auth\_enable = yes**

**broken\_sasl\_auth\_clients = yes**

**smtpd\_recipient\_restrictions = permit\_sasl\_authenticated, permit\_mynetworks, reject\_unauth\_destination**

Save and close the file

Check the configuration again with

**sudo postfix check**

Install the **saslauthd** service

**sudo dnf install cyrus-sasl cyrus-sasl-plain cyrus-sasl-gssapi**

Enable and start **saslauthd** service

**sudo systemctl enable --now saslauthd**

Restart the **postfix** service

**sudo systemctl restart postfix**

If we check again the open ports and the listening processes

**sudo ss -ltnp**

We will see that we have the **postfix master** process listening on port **465**

Let us check again by establishing an interactive session

**openssl s\_client -connect m2.lsaa.lab:465**

We can type the following to receive information from the server

**ehlo m2.lsaa.lab**

And the type **quit** to exit

### Dovecot

Continue on **M2** machine

#### Preparation

In order to be able to send emails from (desktop) clients, we must enable the **submission** service

Open for editing the **/etc/postfix/master.cf** file

**sudo vi /etc/postfix/master.cf**

And uncomment lines **17** to **28**

Save and close the file

Check the values and change them if needed

**postconf -p smtpd\_sasl\_type**

**sudo postconf -e 'smtpd\_sasl\_type = dovecot'**

**postconf -p smtpd\_sasl\_path**

**sudo postconf -e 'smtpd\_sasl\_path = private/auth'**

Restart the **postfix** service

**sudo systemctl restart postfix**

And check the list of open ports and listening processes

**sudo ss -tlpn**

#### Installation

Let us install the required packages

**sudo dnf install dovecot**

Now, open the main configuration file for editing

**sudo vi /etc/dovecot/dovecot.conf**

And set the protocols in use. Uncomment and adjust line **24** (remove **submission** as we turned it on via postfix)

To change the IP version in use, uncomment and adjust line **30**

Save and close the file

Check the **mail\_spool\_directory** parameter

**postconf -p mail\_spool\_directory**

It will return **/var/mail**

Now, open the **/etc/dovecot/conf.d/10-mail.conf** file for editing

**sudo vi /etc/dovecot/conf.d/10-mail.conf**

Copy the text from line **25** and set it on line **30**

On line **114** set **mail\_privileged\_group = mail**

Save and close the file

Add the **dovecot** user to the **mail** group

**sudo usermod -a -G mail dovecot**

Next, adjust the authentication mechanisms

Open for editing the **/etc/dovecot/conf.d/10-auth.conf** file

**sudo vi /etc/dovecot/conf.d/10-auth.conf**

Adjust any authentication related settings if needed

Save and close the file

Open for editing the **/etc/dovecot/conf.d/10-master.conf** file

**sudo vi /etc/dovecot/conf.d/10-master.conf**

Uncomment lines **101** to **103** and add bellow the **mode** line

**user = postfix**

**group = postfix**

Save and close the file

Open the file **/etc/dovecot/conf.d/15-mailboxes.conf** for editing

**sudo vi /etc/dovecot/conf.d/15-mailboxes.conf**

Add **auto = create** to every mailbox definition you like

Save and close the file

Change permissions of the folder where mail is stored

**sudo chmod 0600 /var/mail/\***

Enable and start the **dovecot** service

**sudo systemctl enable --now dovecot**

**systemctl status dovecot**

Restart the **postfix** service

**sudo systemctl restart postfix**

Open the appropriate ports in the firewall

**sudo firewall-cmd --add-service={pop3,pop3s,imap,imaps} --permanent**

**sudo firewall-cmd --reload**

#### Remote Testing

Log on to the **M1** machine

Install terminal mail client

**sudo dnf install mutt**

Create a settings file

**vi .muttrc**

With the following content

**set folder="imap://user@m2.lsaa.lab"**

**set spoolfile="imap://user@m2.lsaa.lab/INBOX"**

Save and close the file

Start the client with **mutt**

You will be asked if you accept the certificate. Confirm with **a**

Then you will be asked for a password

Now, if everything is okay, you will see the content of your mailbox

Try to send an email