Linux Academy

Study Guide

RHCE

Contents

System Configuration and Management

Use Network Teaming/Bonding to Configure Aggregated Network Links Between Two Red Hat Enterprise Systems or Interfaces

- Command line network interface management utility: `nmcli`
- Create a new active/backup team connection:
  - `nmcli con add type team con-name myteam0 ifname myteam0 config '{"runner": {"name": "activebackup"}}'`
- Add a new IP address to the team adapter:
  - `nmcli con mod myteam0 ipv4.addresses '192.168.1.100/24'`
    - Substitute *192.168.1.100* with your desired IP
- Indicate that the teamed interface is manually configured with a static IP rather than DHCP:
  - `nmcli con mod myteam0 ipv4.method manual`
- Assign two existing interfaces to the myteam0 connection:
  - `nmcli con add type team-slave con-name myteam0-port1 ifname eno1 master myteam0`
  - `nmcli con add type team-slave con-name myteam0-port2 ifname eno2 master myteam0`
- Check the state/configuration of the teamed interfaces:
  - `teamdctl myteam0 state`
- Test failover/teaming by disconnecting the primary/active interface:
  - `nmcli dev dis en01`
  - `teamdctl myteam0 state`
- Bring it back online:
  - `nmcli  con up myteam0-port1`

Configure IPv6 Addresses and Perform Basic IPv6 Troubleshooting

- IPv6 Address
  - » 128-bit number
  - » Eight (8) colon-separated groups of four hexadecimal "nibbles" (half-bytes)
  - » Each one (four bits) represents 16 bits of the IPv6 address
  - » Example: **3022:0cd8:0000:0100:0000:0000:0000:0001**
- Short Hand
  - » A group of all zeros in a nibble can be represented by colons alone, like so: **3022:0cd8:0:01::1**
  - » Double zeros in groups must be suppressed
  - » Use **::** to shorten when possible
  - » Cannot have two **::** substitute shortcuts in any one address, regardless of grouping of zeros
  - » Typically do not use **::** to simply shorten ONE group of zeros
  - » Hex number values are always lowercase
- `nmcli` is otherwise used as normal to manage IPv6 addresses

Route IP Traffic and Create Static Routes

- `ip route`
  - » Displays the current routing table, including static routes restored on boot
  - » Replaces the `netstat -rn` command for showing routes, although that still works on all distributions, as well, at this time
- `ip route add [network/subnet] via [gateway ip] dev [interface]`
- `ip route add 192.168.1.0/24 via 192.168.0.1 dev eth1`
  - » This adds a static route to the 192.168.1.0 network (addresses .1 through .254) via the 192.168.0.1 IP through device eth1
- **/etc/sysconfig/network-scripts**
  - » Directory where files exist for interface and network routing persistence
  - » Example:
    - –
```
route-eth1
ADDRESS0=192.168.1.0
```

```
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.1
```

  - These will persistently create the same static route that was added dynamically with the `route` command above

Use Firewalld and Associated Mechanisms to Implement Packet Filtering and Configure Network Address Translation

- Firewalld is the firewall default daemon
- Managed at the command line with `firewall-cmd`
- Allows working with services, zones and rules, as well as importing custom rules from XML files
- Service configurations are stored in XML files located at **/usr/lib/firewalld/services** and **/etc/firewalld/services** (depending on user-defined [`/usr/lib`] or system-level [`/etc/firewalld`] definitions)
- Restart the firewall:
  - `systemctl restart firewalld`
- Reloading persistent firewall rules:
  - `firewall-cmd -reload`
- Zone management parameters:
  - `-get-default-zone`
  - `-set default-zone`
  - `-get-active-zones`
  - `-get-zones`
  - `-list-all`
  - `-list-all-zones`
  - `-new-zone`
  - `-delete-zone`
  - `-permanent`
  - `-zone`
- Current vs permanent firewall rule changes:
  - Commands making rule changes of any kind are NOT persistent (in other words, will not persist across reboots or service restarts) UNLESS the `-permanent` option is specified
  - Commands making rule changes take immediate effect UNLESS specifying the `-permanent`

parameter, rules specifying permanent must be applied with a subsequent `-reload`

- Service management options:
  - `-get-services`
  - `-list-services`
  - `-query-service`
  - `-add-service`
  - `-remove-service`
  - `-new-service`
  - `-delete-service`
- Port management options:
  - `-list-ports`
  - `-add-port`
  - `-remove-port`
  - `-query-port`
- When adding ports, the port number must specify TCP/UDP for application
- Ranges can be added with dashes:
  - `-add-port 500-599/TCP for example`
- Rich rule management:
  - `-list-rich-rules`
  - `-add-rich-rule`
  - `-remove-rich-rule`
  - `-query-rich-rule`
- Sample using rich rules to allow inbound HTTP access from a network IP range 10.0.1.0/24, logging each one at the info log level and making the change permanent:
  - `firewall-cmd -add-rich-rule 'rule-family="ipv4" source address="10.0.1.0/24" service name="http" log prefix="HTTP Allow Rule" level="info" accept' -permanent`
  - This will update the default zone (public by default) XML rule file in /etc/firewalld/zones/public.xml
- Port forwarding options:
  - `-list-forward-ports`
  - `-add-forward-port`

  - » `-remove-forward-port`
  - » `-query-forward-port`
- Forwarding example for redirecting SSH from port 22 to port 2222 in the DMZ zone:
  - » `firewall-cmd -zone DMZ -permanent -add-forward-port port=22:proto=tcp:toport=2222`

Use /proc/sys and systctl to Modify and Set Kernel Runtime Parameters

- The **/proc** filesystem contains system level configuration items
- Making changes to a running kernel parameter can be done by manipulating the values within the file pertaining to the section you wish to change; for example, to change the IPv4 forwarding value in order to allow your system to function as a network gateway, you can do the following to effect an immediate change:
  - » `echo 1 > /proc/sys/net/ipv4/ip_forward`
  - » NOTE: this does NOT persist on a reboot
    - – You can add that command to **/etc/rc.d/rc.local**
- Using `sysctl` and **sysctl.conf**:
  - » `sysctl` will allow you to view available kernel runtime parameters:
    - – `sysctl -a`
  - » `sysctl` will allow you to set a value:
    - – `sysctl -w vm.pagesize = 4096`
- To permanently storing changes, write them to **/etc/sysctl.conf**
- For example, to add the ip_forward change persistently:
  - » `net.ipv4.ip_forward = 1`
  - » Applying the change:
    - – `sysctl -p`
    - – *or* `reboot`

Configure a System to Authenticate Using Kerberos

- Kerberos authentication requires:
  - » KDC server

  - » Client
  - » Packages:
    - – `krb5-server`
    - – `krb5-workstation`
    - – `pam_krb5`

- KDC server:
  - » **/var/kerberos/krb5kdc/kdc.conf**
    - – Replace *EXAMPLE.COM* with your own domain/realm
    - – Option: To make Kerberos 5 compatible only, comment out the *default_principal_flags = +preauth*
  - » **/etc/krb5.conf**
    - – Replace *EXAMPLE.COM/example.com* with your realm/domain (case sensitive)
    - – Replace server name with your KDC server name
  - » **/var/lib/krbkdc/kadm5.acl**
  - » Replace *EXAMPLE.COM* with your domain
- Fully Qualified Domain Names:
  - » Kerberos *requires* FQDN references; if you do not have FQDN or are on a private network, be sure all clients and servers have the FQDN set to localhost (**/etc/hosts**) for those authenticating to the KDC
- Creating the Kerberos authentication database:
  - » `kdb5_util create -s -r EXAMPLE.COM`
    - – Replace **EXAMPLE.COM** as appropriate for your configuration
- Managing the Kerberos services:
  - » `systemctl enable krb5kdc kadmin`
  - » `systemctl start krb5kdc`
  - » `systemctl start kadmin`
- Kerberos administration tool:
  - » If local: `kadmin.local`
  - » If remote: `kadmin`
- Adding a principal admin account for Kerberos:

  - » `addprinc root/admin`
- Adding a user account:
  - » `addprinc user01`
- Adding hostname allowed to authenticate:
  - » `addprinc -randkey host/kdcserver.example.com`
    - – Replace the server name and *example.com* with your own information
- Creating keytab file once complete:
  - » `ktadd host/kdcserver.example.com`
- Configuring the host to allow Kerberos user authentication via `ssh`:
  - » **/etc/ssh/ssh_config**
    - – `GSSAPIAuthentication yes`
    - – `GSSAPIDelegateCredentials yes`
  - » `systemctl reload sshd`
- Allow PAM authentication at the command line:
  - » `authconfig -enablekrb5 -update`
- Client and user authentication (local KDC or separate client after user and host added as above):
  - » `su - user01`
  - » `kinit`
    - – Password for user when added to KDC
  - » `klist`
    - – Will show host and user associated to principle
  - » `ssh kdcserver.example.com`

Configure a System as Either an iSCSI Target or Initiator That Persistently Mounts an iSCSI Target

- Server configuration, called iSCSI target:
  - » `yum install -y targetcli`
- Enable and activate the service:
  - » `systemctl enable target`
  - » `systemctl start target`

  - Note: No configuration has been made, target will start and stop

- Two types of target storage:
  - `fileio backstore`
    - Specially formatted block file; less performant
  - `block backstore`
    - Associated with partition, device or LVM; more performant
- `fileio backstore creation`
  - `targetcli`
  - `backstores/fileio/ create [NAME OF BACKSTORE] [LOCATION OF IMG FILE] [SIZE in M/G/T]`
- `block backstore`
  - `targetcli`
  - `backstores/block/ create [NAME OF BACKSTORE] [DEVICE AND PARTITION]`
    - Note: No size is needed as it is determined by the device and partition size references
- Create an IQN (iSCSI Qualified Name) and target:
  - Standard format, in our example:
    - `iqn.2016.05.com.example:targetname`
    - Note: Will create a TPG (Target Profile Group)
- Create a LUN:
  - `fileio backstore`
    - `luns/ create /backstores/fileio/[NAME OF BACKSTORE]`
  - `block backstore`
    - `luns/ create /backstores/block/[NAME OF BACKSTORE]`
- Create an ACL for your client and the initiator name:
  - `acls/ create iqn.2016.05.com.example:client`
- Optional: Create userid and password:
  - `set suth userid=username`
  - `set auth password=password`
- Client configuration, called iSCSI Initiator:
  - `yum install -y iscsi-initiator-utils`
- **/etc/iscsi/initiatorname.iscsi**

  - » Replace with initiator name created on server (in our example):
    - – `InitiatorName=iqn.2016.05.com.example:client`
- Additionally, if userid and password set: **/etc/iscsci/iscsid.conf**
  - » `node.session.auth.authmethod = CHAP`
  - » `node.session.auth.username = username`
  - » `node.session.auth.password = password`
- Enable and start the service:
  - » `systemctl enable iscsi`
  - » `systemctl start iscsi`
- Discover the remote target:
  - » `iscsiadm -mode discovery -type sendtargets -portal [server ip]`
- Get the connection for the target and establish connection:
  - » `iscsiadm -mode node -targetname iqn.2016.05.com.example:targetname -portal [server ip] -login`
- Check the type:
  - » `lsblk -iscsi`
- Make the filesystem (no fdisk/gdisk, simple mkfs command)
- **/etc/fstab** mounting:
  - » `get block id, blkid`
  - » `UUID=[uuid from blkid cmd]  /mnt/location  ext4  _netdev 0 0`
    - – */mnt/location* can be wherever your mount is intended
    - – *ext4* can be whatever filesystem you choose
    - – *_netdev* indicates during boot to wait for full network availability before attempting mount

Produce and Deliver Reports on System Utilization

- Common system utilization utilities covered in other courses or certifications valid here:
  - » `ps`
  - » `pgrep`
  - » `top`
  - » `ss`
  - » `nmap`

- `yum install -y sysstat`
- Enable and activate:
  - » `systemctl enable sysstat`
  - » `systemctl start sysstat`
- Reporting options:
  - » Example: `sadf -d /var/log/sa/sa15 -- -u -r -dp -n DEV`
    - – Will produce a report for the 15th of the month for CPU, memory, disk activity and network (where DEV is the interface you want the report to apply to for activity)

HTTP/HTTPS

- Install and configure Apache:
  - » `httpd`
  - » Apache service to install
  - » `systemctl enable httpd`
  - » `systemctl start httpd`
  - » By default, listening on port 80
- Key configuration files and directories:
  - » **/etc/httpd**
    - – Primary http configuration directory, all httpd configuration files and directories root here
  - » **/etc/httpd/conf**
    - – **magic** – mime configuration file for filetype definitions
    - – **httpd.conf** – primary overall apache configuration, default location for vhosts
  - » **/etc/httpd/conf.d**
    - – User and index configuration file (user directory, welcome message and indexes)
  - » **/etc/httpd/conf.d.modules**
    - – Module configuration files (as referenced in **httpd.conf**)
  - » **/etc/httpd/logs**
    - – Links to **/var/log/httpd** by default
  - » **/etc/httpd/modules**

  - Apache modules, links to **/usr/lib64/httpd/modules** by default
- **/etc/httpd/run**
  - Temporary run files for httpd process, links to **/run/httpd** by default
- **/var/www/html**
  - Default directory for website content

Configure Private Directories

- Allowing the web server to provide browser-based access to user directories.
- **/etc/httpd/conf/httpd.conf**
  - `<Directory "/var/www/html/private">    # Example`
  - `AuthType Basic`
  - `AuthName "Password Protected Directory"    # Example`
  - `AuthUserFile /etc/httpd/conf/passwd   # Can be elsewhere`
  - `Require user USERNAME    # Replace USERNAME as appropriate`
  - `</Directory>`
- Test configuration:
  - `apachectl configtest`
- Create the password for access for user USERNAME:
  - `htpasswd -c /etc/httpd/conf/passwd USERNAME`
- Set permissions:
  - `chown 600 /etc/httpd/conf/passwd`
- Ownership to Apache:
  - `chown apache:apache /etc/httpd/conf/passwd`
- **.htpasswd** file can be used locally instead
- `systemctl restart httpd`
- Can be accessed via URL:
  - `http://[SERVNAME]/private`
  - Should prompt for username and password

Configure Group Managed Content

- Similar to the private directory, for groups
- **/etc/httpd/conf/httpd.conf**
  - `<Directory "/var/www/html/private">    # Example`
  - `AuthType Basic`
  - `AuthName "Password Protected Directory"    # Example`
  - `AuthGroupFile /etc/httpd/conf/group   # Can be elsewhere`
  - `AuthUserFile /etc/httpd/conf/passwd   # Can be elsewhere`
  - `Require group GROUPNAME    # replace GROUPNAME as appropriate`
  - `</Directory>`
- Test configuration:
  - `apachectl configtest`
- Create the password for access for group GROUPNAME:
  - `htpasswd -c /etc/httpd/conf/passwd GROUPNAME`
- Set permissions:
  - `chown 600 /etc/httpd/conf/passwd`
  - `chown 600 /etc/httpd/conf/group`
- Ownership to Apache:
  - `chown apache:apache /etc/httpd/conf/passwd`
  - `chown apache:apache /etc/httpd/conf/group`
- **.htpasswd** file can be used locally instead
- `systemctl restart httpd`
- Can be accessed via URL:
  - `http://[SERVNAME]/private`
  - Should prompt for username and password

Configure a Virtual Host

- Create virtual host directory:
  - `mkdir /etc/httpd/vhost.d`
- Add the directory to **/etc/httpd/conf/httpd.conf**

  - » Include **vhost.d/*.conf**

- Externalizes vhost configuration; by default, vhost configurations can be added directly to **httpd.conf**
- Sample configuration for domain **myhost.sample.com**:
  - » **/etc/httpd/vhost.d/myhost.sample.com_http.conf**
    - – `<VirtualHost *:80>`
    - – `ServerAdmin admin@myhost.sample.com`
    - – `DocumentRoot /var/www/html/myhost.sample.com`
    - – `ServerName myhost.sample.com`
    - – `ServerAlias myhost`
    - – `ErrorLog logs/myhost.sample.com-error_log`
    - – `CustomLog logs/myhost.sample.com-access_log common`
    - – `</VirtualHost>`
- Testing the configuration:
  - » `apachectl configtest`
- Apache must be restarted or reloaded gracefully to read the new virtual host:
  - » `systemctl restart httpd`
  - » `systemctl reload httpd`
- Show a dump of the virtual host configurations for Apache, in general:
  - » `httpd -D DUMP_VHOSTS`

Deploy a basic CGI application

- **/var/www/cgi-bin**
  - » Default, preconfigured CGI Apache directory
  - » Files must have '755' permissions, owned by Apache
- SELinux considerations:
  - » `getsebool httpd_enable_cgi`
  - » If off: `setsebool httpd_enable_cgi on` (as root or sudo)
- Using non-default CGI directory:
  - » `mkdir /mycgi`
- SELinux considerations for custom directory for CGI:

  - » `semanage fcontext -a -t httpd_sys_script_exec_t "/mycgi(/*)?"`
  - » `restorecon -R /mycgi`
- Changes for custom directory, **/etc/httpd/conf/httpd.conf**
  - » Replace *ScriptAlias*
  - » `ScriptAlias /cgi-bin/ "/mycgi/"`
- In virtual host file for site:
  - » `<Directory "/mycgi">`
  - » `AllowOverride None`
  - » `Options None`
  - » `Require all granted`
  - » `</Directory>`
- Testing the configuration:
  - » `apachectl configtest`
- Apache must be restarted or reloaded gracefully to read the new virtual host
  - » `systemctl restart httpd`
  - » `systemctl reload httpd`

Configure TLS Security

- Make sure openssl is installed:
  - » `yum -y install openssl`
- Generating a certificate keyfile:
  - » `openssl req -new -x509 -nodes -keyout /etc/httpd/ssl-keys/myhost.sample.com.key -days 730`
- Generating a self-signed certificate using the generated key:
  - » `openssl req -new -x509 -nodes -out /etc/httpd/ssl-certs/myhost.sample.com.crt -keyout /etc/httpd/ssl-certs/myhost.sample.com.key -days 730`
- System-wide certificate:
  - » **/etc/httpd/conf.d/ssl.conf**
    - – `SSLCertificateFile /etc/httpd/ssl-certs/myhost.sample.com.crt`
    - – `SSLCertificateKeyFile /etc/https/ssl-certs/myhost.sample.com.key`
    - – Search and replace: `ServerName myhost.sample.com:443`

- Test configuration:
  - » `httpd -t`
  - » `apachectl configtest`
- Certificates for specific virtual hosts: Add the following to virtual host file (in our configuration, in **/etc/httpd/vhost.d/myhost.sample.com_https.conf**):
  - » Make sure the following exist:
    - – `<VirtualHost *:443>`
    - – typical virtual host configurations for directories, server names, etc (see above)
    - – `SSLEngine on`
    - – `SSLCertificateFile /etc/httpd/ssl-certs/myhost.sample.com.crt`
    - – `SSLCertificateKeyFile /etc/https/ssl-certs/myhost.sample.com.key`
- Testing the configuration:
  - » `apachectl configtest`
- Apache must be restarted or reloaded gracefully to read the new virtual host:
  - » `systemctl restart httpd`
  - » `systemctl reload httpd`
- Show a dump of the virtual host configuration for apache in general:
  - » `httpd -D DUMP_VHOSTS`

DNS

Configure a Caching-Only Name Server

- Install the bind name server:
  - » `yum install -y bind`
- **/etc/named.conf**
  - » Primary configuration service file
  - » Change:
    - – `listen-on port 53 { 127.0.0.1; }; # replace 127.0.0.1 with 'any'`
    - – `allow-query { localhost };  # replace localhost with 'any'`
    - – `dnssec-validation no;`
  - » Validate the changes:

  - `named-checkconf`
- » Enable and start the service:
  - `systemctl enable named`
  - `systemctl start named`
- » Testing:
  - `nslookup google.com 127.0.0.1 # or local IP of server`
  - `dig@127.0.0.1 google.com # or local IP of server`
- » Client utilities and files for troubleshooting DNS
  - `nslookup`
  - `dig`
  - `ping`
  - **/etc/resolv.conf**: Client configuration pointing to DNS server(s)

NFS

Provide Network Shares to Specific Clients

- Group installation for server:
  - » `yum groupinstall -y file-server`
- Enable services for NFS server:
  - » `systemctl enable rpcbind nfs-server`
- Sample directory to provide for file sharing:
  - » `mkdir /home/fileshare`
- Apply permissions:
  - » `chmod 0777 /home/fileshare`
- SELinux considerations:
  - » View all NFS related booleans:
    - `semanage Boolean -l | grep nfs`
  - » Set booleans for our configuration:
    - `setsebool -P nfs_export_all_ro on`
    - `setsebool -P nfs_export_all_rw on`
    - `setsebool -P use_nfs_home_dirs on`

- **/etc/exports**
  - » Defines the shares to export for use:
    - – `/home/fileshare client.sample.com(rw,no_root_squash)`
    - – Note: no space between domain/IP and options in parentheses
- Exporting the filesystem:
  - » `exportfs -avr`
- Checking the exported filesystems on the server:
  - » `showmount -e localhost`
- Client installation:
  - » `yum install -y nfs-utils`
- Mounting the filesystem on the **/mnt/shared** directory:
  - » `mount -t nfs nfsserver.sample.com:/home/fileshare /mnt/shared`
- **/etc/fstab**
  - » `[servername/ip]:[/path/to/share] [/path/local/mountpoint] nfs options 0 0`
    `# example below`
  - » `nfsserver.sample.com:/home/fileshare /mnt/shared nfs _`
    `netdev,wsize=8192,rsize=8192,timeo=14,intr 0 0`

Provide Network Shares Suitable for Group Collaboration

- Same server packages as above for NFS server
- Same services enabled and started
- Create a directory to be exported:
  - » `mkdir /mygroup`
- Create a group that will have access to this share:
  - » `groupadd -g 50000 grpshare`
- Add the group ownership to the shared filesystem:
  - » `chgrp grpshare /mygroup`
- Change permissions, including special permissions bit:
  - » `chmod 2770 /mygroup`
- **/etc/exports**
  - » Add the line for the export:

  - `/mygroup client.sample.com(rw,no_root_squash)`

- SELinux considerations
  - » View all NFS-related booleans
    - `semanage Boolean -l | grep nfs`
  - » Set booleans for our configuration:
    - `setsebool -P nfs_export_all_ro on`
    - `setsebool -P nfs_export_all_rw on`
    - `setsebool -P use_nfs_home_dirs on`
- Exporting the filesystem:
  - » `exportfs -avr`
- Checking the exported filesystems on the server:
  - » `showmount -e localhost`
- Client installation:
  - » `yum install -y nfs-utils`
- Mounting the filesystem on the **/mnt/shared** directory:
  - » `mount -t nfs nfsserver.sample.com:/mygroup /mnt/shared`
- **/etc/fstab**
  - » `[servername/ip]:[/path/to/share] [/path/local/mountpoint] nfs options 0 0 # example below`
  - » `nfsserver.sample.com:/mygroup /mnt/shared nfs _ netdev,wsize=8192,rsize=8192,timeo=14,intr 0 0`

Use Kerberos to Control Access to NFS Shares

- NFSshould be set up as above on the server and client
- Add the package:
  - » `yum install -y nfs-secure-server`
- Enable and start the servers:
  - » `systemctl enable nfs-server nfs-secure-server`
  - » `systemctl start nfs-server`
  - » `systemctl start nfs-secure-server`
- KDC server should already be configured

- Using the KDC admin utility to add the appropriate NFS server configuration:
  - `kadmin` (or `kadmin.local` if KDC and NFS services on same system)
  - `ktadd nfs/nfsserver.sample.com`
- **/etc/exports**
  - Add the option `sec=krb5` insider the options for any export to be protected via Kerberos
    - `/mygroup client.sample.com(rw,no_root_squash,sec=krb5)`
- Exporting the filesystem:
  - `exportfs -avr`
- Checking the exported filesystems on the server:
  - `showmount -e localhost`
- NFS secure client should have the pam and Kerberos packages
- Using the KDC admin utility, add the appropriate client configuration:
  - `kadmin`
  - `ktadd nfs/client.sample.com`
- RHEL 7.1+ systems need to enable and start the special nfs-client.target:
  - `systemctl enable nfs-client.target`
  - `systemctl start nfs-client.target`
- Mounting the remote directory:
  - `mount -t nfs4 -o sec=grb5 nfsserver.sample.com:/mygroup /mnt/grpshare`
- **/etc/fstab**
  - `[servername/ip]:[/path/to/share] [/path/local/mountpoint] nfs4 options [sec=krb5] 0 0 # example below`
  - `nfsserver.sample.com:/mygroup /mnt/shared nfs4 _ netdev,wsize=8192,rsize=8192,timeo=14,intr,sec=krb5 0 0`
- Switch to the user (in our original example, user01):
  - `su - user01`
- Obtain the Kerberos ticket:
  - `kinit`
- You can now read/write files in the mounted **/mnt/shared** directory.

SMB

Provide Network Shares to Specific Client

- Group installation for server:
  - `yum groupinstall -y file-server`
- **/etc/samba/smb.conf**
  - Primary configuration file for Samba service and shares
- Settings needed to change:
  - `workgroup = YOURDOMAIN`
  - `netbios name = YOURSERVER`
  - `interfaces = lo en01 10.0.1.0/24 # sample device and network`
  - `hosts allow = 127. 10.0.1.  # sample network`
- Adding a shared filesystem:
  - `[shared]`
  - `browseable = yes`
  - `path = /path/to/your/share`
  - `valid users = user01 ▯ sample user`
  - `writable = yes`
- Testing the syntax and configuration:
  - `testparm`
  - Will show output for the configuration and share destination section
- Create the directory you indicated in your configuration for the share.
- Change the permissions to *rwx* for everyone
- SELinux considerations:
  - On a directory named */myshare* do the following:
    - `semanage fcontext -a -t samba_share_t "/myshare(/.*)?"`
    - `restorecon -R /myshare`
- Enable the services needed:
  - `systemctl enable smb nmb winbind`
- Start the services:
  - `systemctl start smb`
  - `systemctl start nmb`

  - » `systemctl start winbind`
- Create the user (if needed), create the samba password for the user account:
  - » `smbpasswd -a user01`

SMTP

Configure a System to Forward All Email to a Central Mail Server

- Install Postfix:
  - » `yum install -y postfix`
- Enable and start it:
  - » `systemctl enable postfix`
  - » `systemctl start postfix`
- Build what is called a "null client" configuration.
- **/etc/postfix/main.cf**
  - » Changes needed for scenario of network 10.0.1.0/24 and the mail server called mail.sample.com on IP 10.0.1.100:
    - myhostname = mail.sample.com
    - mydomain = sample.com
    - myorigin = $mydomain
    - inet_interfaces = loopback-only
    - mydestination =
    - relayhost = 10.0.1.100
  - » All other items can be left at default.
- Checking the configuration:
  - » `postfix check`
  - » `postconf -a`
- Reload the configuration:
  - » `systemctl restart postfix`
- Testing:

  - » `echo "Some test" | mail -s "Testing Mail Forwarding" user01@sample.com`
  - » `mailq`
  - » Log in as user01 and should see a message you have mail

DATABASE SERVICES

Install and Configure MariaDB

- Install the service and client:
  - » `yum install -y mariadb mariadb-server`
- Enable and start the service:
  - » `systemctl enable mariadb`
  - » `systemctl start mariadb`
- Run the secure installation, setting root password:
  - » `mysql_secure_installation`
- **/etc/my.cnf**
  - » MariaDB configuration file
- Enabling remote access in **/etc/my.cnf**:
  - » `bind-address [server ip]`

Backup and Restore a Database

- Back up utility:
  - » `mysqldump`
- Backing up sample database called *mydb* using the root user on the localhost:
  - » `mysqldump -user=root password="password" -result-file=mydb.sql mydb`
- Restoring the same database (note that the database itself MUST already exist, it will not recreate the database with the above command backup):
  - » `mysql -user=root -password="password" mydb < mydb.sql`

Create a Simple Database Schema

- Connect to MariaDB server, as root:
  - » `mysql -u root -p`

  - Will prompt for root password as defined in the `mysql_secure_installation` set up above

- Creating the database:
  - `CREATE DATABASE MyDB;  # example`
- Granting permission on that DB to a user called *user01* from the localhost:
  - `GRANT ALL ON MyDB.* TO user01@localhost IDENTIFIED BY 'password';`
  - `FLUSH PRIVILEGES; # reloads DB permissions tables`
- Reconnect as the user just created:
  - `mysql -u user01 -p`
    - Will prompt for password from SQL GRANT statement above
- Indicate the DB with which to work:
  - `USE MyDB;`
- Create a simple user table with firstname, lastname and emails address fields, all string fields with 50 character limits
  - `CREATE TABLE tblUserInfo(firstname varchar(50),lastname varchar(50),email varchar(50));`
- Show the table just created:
  - `SHOW TABLES;`
- Show the fields in the table called *tblUserInfo* just created:
  - `DESC tblUserInfo;`

Perform Simple SQL Queries Against a Database

- Reconnect as the user just created:
  - `mysql -u user01 -p`
    - Will prompt for password from SQL GRANT statement above
- Indicate the DB to with which to work:
  - `USE MyDB;`
- Insert records:
  - `INSERT tblUserInfo VALUES("John","Smith","jsmith@sample.com");`
  - `INSERT tbl UserInfo VALUES("Bruce","Wayne","thebatman@justiceleague.com");`
- Select records from the table:
  - `SELECT * FROM tblUserInfo;`

  - Will display both records

- Select certain records:
  - » `SELECT * FROM tblUserInfo WHERE firstname="Bruce";`
    - Will only show Batman's name