Service Management

1. Installing and configuring a web server

1.1. Installing and starting a web server

1.1.1. Install a web server software:

```
# yum install httpd mod_ssl
```

1.1.2. Enable and start the "httpd" service:

```
# systemctl enable --now httpd
Created symlink from /etc/systemd/system/multi-
user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
```

or

```
# systemctl enable httpd
# systemctl start httpd
```

1.1.3. Verify that the service is active:

```
# systemctl is-active httpd
active
```

or

```
# systemctl status httpd
```

1.1.4. Optionally (for the testing purposes) create a test page:

```
# echo "I am $(hostname)." > /var/www/html/index.html
```

1.2. Enabling web server communication through the firewall

1.2.1. Modify the firewall rules to allow connections on the standard web server ports "80" and "443":

```
# firewall-cmd --add-service=http --add-service=https --permanent
success
```

1.2.2. Reload the firewall configuration:

```
# firewall-cmd --reload
success
```

1.2.3. Verify the firewall configuration for the default (public) zone:

```
# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: dhcpv6-client http https ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

1.2.4. Optionally display the test page executing the following command from another computer:

```
# curl <IP>
I am Arnold.
```

(Use the IP address of the web server.)

1.3. Changing the web server port

1.3.1. Change the default port in the web server configuration file to "8090":

```
# sed -i 's/Listen 80/Listen 8090/' /etc/httpd/conf/httpd.conf
```

1.3.2. Enable the port via the firewall:

```
# firewall-cmd --add-port=8090/tcp --permanent
```

1.3.3. Reload the firewall configuration:

```
# firewall-cmd --reload
```

1.3.4. Enable the port within SELinux (if SELinux is used):

```
# semanage port -at http_port_t -p tcp 8090
```

1.3.5. Restart the web server:

```
# systemctl restart httpd
```

1.3.6. Verify that the service is listening on the new port:

```
# netstat -an | grep :8090
tcp        0      0 0.0.0.0:8090              0.0.0.0:*               LISTEN
```

1.3.7. Optionally display the test page executing the following command from another computer:

```
# curl <IP>:8090
I am Arnold.
```

(Use the IP address of the web server.)

---

2. Creating a custom systemd service as root

2.1. Create a custom systemd service unit file in the "/etc/systemd/system" directory that will execute a script every 5 minutes:

```
# cat > /etc/systemd/system/swap_check.service

[Unit]
Description=Swap check service

[Service]
Type=oneshot
ExecStart=/usr/local/bin/swap_check.sh
StandardOutput=append:/var/log/swap_check.log
StandardError=append:/var/log/swap_check.log

[Install]
WantedBy=default.target

^D
```

(systemd unit files under the "/usr/lib/systemd/system" directory are provided by software packages and any changes to them may be overriden during software updates.)

2.2. Create a custom systemd timer unit file in the "/etc/systemd/system" directory that will trigger the corresponding service:

```
# cat > /etc/systemd/system/swap_check.timer

[Unit]
Description=Activate swap_check.service every 5 minutes

[Timer]
OnCalendar=*:0/5

[Install]
WantedBy=timers.target

^D
```

2.3. Create the associated script and log file:

```
# cat > /usr/local/bin/swap_check.sh
#!/bin/bash

# Get total swap used
total_swap_used=$(free -h | awk '/Swap/ {print $3}')

# Print the result
echo "Date: $(date)"
echo "Total Swap Used: $total_swap_used"
echo

^D
```

```
# chmod +x /usr/local/bin/swap_check.sh
# touch /var/log/swap_check.log
```

2.4. Verify that the custom systemd unit files contain no errors:

```
# systemd-analyze verify /etc/systemd/system/swap_check.*
```

(If the command returns no output, the files have passed the verification successfully.)

2.5. Reload the systemd configuration to recognize the newly created unit files:

```
# systemctl daemon-reload
```

2.6. Start the timer unit that triggers the corresponding service:

```
# systemctl start swap_check.timer
```

2.7. Enable the timer unit after the system boot:

```
# systemctl enable swap_check.timer
Created symlink /etc/systemd/system/timers.target.wants/swap_check.timer →
/etc/systemd/system/swap_check.timer.
```

2.8. Check the status of the timer unit:

```
# systemctl status swap_check.timer
● swap_check.timer - Activate swap_check.service every 5 minutes
     Loaded: loaded (/etc/systemd/system/swap_check.timer; enabled; preset:
disabled)
     Active: active (waiting) since Tue 2024-01-30 23:22:13 CET; 12s ago
    Trigger: Tue 2024-01-30 23:25:00 CET; 2min 33s left
   Triggers: ● swap_check.service

Jan 30 23:22:13 arnold systemd[1]: Started swap_check.timer - Activate
swap_check.service every 5 minutes.
```

2.9. Check the status of the service unit:

```
# systemctl status swap_check.service
○ swap_check.service - Swap check service
     Loaded: loaded (/etc/systemd/system/swap_check.service; disabled;
preset: disabled)
     Active: inactive (dead) since Tue 2024-01-30 23:22:16 CET; 20s ago
   Duration: 2.089s
TriggeredBy: ● swap_check.timer
    Process: 109496 ExecStart=/usr/local/bin/swap_check.sh (code=exited,
status=0/SUCCESS)
   Main PID: 109496 (code=exited, status=0/SUCCESS)
        CPU: 2.525s

Jan 30 23:22:13 arnold systemd[1]: Starting swap_check.service - Swap check
service...
Jan 30 23:22:16 arnold systemd[1]: swap_check.service: Deactivated
successfully.
Jan 30 23:22:16 arnold systemd[1]: Finished swap_check.service - Swap check
service.
Jan 30 23:22:16 arnold systemd[1]: swap_check.service: Consumed 2.525s CPU
time.
```

2.10. View the logs of the timer unit:

```
# journalctl -u swap_check.timer
```

2.11. View the logs of the service unit:

```
# journalctl -u swap_check.service
```

2.12. View the updated output of the service:

```
# tail -f /var/log/swap_check.log
```

---

3. Creating a custom systemd service as a regular user

3.1. Log in directly as a particular user (do not use the "su" or "sudo" command).

3.2. Create the user's systemd configuration directory (if it does not already exist):

```
$ mkdir -p ~/.config/systemd/user
```

3.3. Create a custom systemd service unit file for a sample application, e.g., "myweb.service":

```
$ cat > ~/.config/systemd/user/myweb.service

[Unit]
Description=My sample application

[Service]
WorkingDirectory=/home/dookie/myweb/html
ExecStart=/usr/bin/python3 -m http.server 8080
Restart=on-failure

[Install]
WantedBy=default.target

^D
```

3.4. Create the associated directory and file:

```
$ mkdir -p ~/myweb/html
$ echo "Hello World" > ~/myweb/html/index.html
```

3.5. Verify that the custom systemd service unit file contains no errors:

```
$ systemd-analyze --user verify ~/.config/systemd/user/myweb.service
```

(If the command returns no output, the file has passed the verification successfully.)

3.6. Reload the systemd configuration to recognize the newly created unit file:

```
$ systemctl --user daemon-reload
```

3.7. Start the service:

```
$ systemctl --user start myweb
```

3.8. Enable the service:

```
$ systemctl --user enable myweb
Created symlink
/home/testuser/.config/systemd/user/default.target.wants/myweb.service →
/home/testuser/.config/systemd/user/myweb.service.
```

3.9. Check the status of the service:

```
$ systemctl --user status myweb
● myweb.service - My sample application
     Loaded: loaded (/home/testuser/.config/systemd/user/myweb.service;
disabled; preset: disabled)
     Active: active (running) since Tue 2024-01-30 09:17:13 CET; 14s ago
   Main PID: 75650 (python3)
      Tasks: 1 (limit: 9226)
     Memory: 11.7M
        CPU: 56ms
     CGroup:
/user.slice/user-1001.slice/user@1001.service/app.slice/myweb.service
             └─75650 /usr/bin/python3 -m http.server 8080

Jan 30 09:17:13 arnold systemd[75548]: Started myweb.service - My sample
application.
```

3.10. Enable the service to run independently of an active user session:

```
$ loginctl enable-linger $USER
```

(User-specific systemd services terminate by default when the user logs out.)

3.11. Optionally verify the configuration:

```
$ loginctl show-user $USER | grep -i linger
Linger=yes
```

3.12. View the logs of the service:

```
$ journalctl --user -u myweb
```

3.13. Verify that the web service is working on port 8080:

```
$ curl http://localhost:8080
Hello World
```