COURSEWARE

Managing systemd Services

Contents

Overview

A service, or daemon, is a background application that waits to be used, or carries out essential tasks, rather than being in direct control of an interactive user.
An example of this is all of the processes running in the background on a machine, such as: the networking services, graphical interface, etc.
We can use the systemd software suite as a unified and consistent way of controlling all of these daemons, and even use it to manage our own applications, such as web and API servers.

systemctl

Within the systemd suite there is a tool called `systemctl` (System Control), which can be used to interact with services.

Listing Services

The `list-units` command can be used to see everything being managed by systemctl.
To see the services specifically, which is what we are interested in here, we can use the `--type` option:

```
sudo systemctl list-units --type service
```

Starting a Service

Use the `start` command to start a service.
The service that has been configured will be started in the background:

```
# sudo systemctl start [SERVICE_NAME]
sudo systemctl start my-service
```

Stopping a Service

Use the `stop` command to stop a service.
The process managed by systemd will be terminated for you:

```
# sudo systemctl stop [SERVICE_NAME]
sudo systemctl stop my-service
```

Restarting a Service

Stop and then start a service.
This can be used if a new version of the service has been installed, or if configurations for the application have been updated:

```
# sudo systemctl restart [SERVICE_NAME]
sudo systemctl restart my-service
```

Reloading a Service Configuration

Some services have the capability of reloading their configuration without needing to restart.
This is the preferred option (if it's available) because if the configurations are invalid, the service will continue to use the configuration that was already in place and continue running.
Lets say a new configuration was entered for a service, and the service was then restarted, the service could crash if the configuration is invalid; however, this could be avoided by using the reload command:

```
# sudo systemctl reload [SERVICE_NAME]
sudo systemctl reload my-service
```

Enabling a Service

It's common to need a service to be running once the system turns on.
We can use the enable command to make a service run on start up:

```
# sudo systemctl enable [SERVICE_NAME]
sudo systemctl enable my-service
```

Enable and Start

The `--now` option can also be used to start the service.
You can think of this as running the `start` and `enable` commands at the same time:

```
# sudo systemctl enable --now [SERVICE_NAME]
sudo systemctl enable --now my-service
```

Disable a Service

This command may sound a little misleading, as it wont actuallly "disable" the service. It will actually just stop it running on startup (so the opposite of enabling a service):

```
# sudo systemctl disable [SERVICE_NAME]
sudo systemctl disable my-service
```

Check the Service Status

To make sure everything is running well, you can check the status of your service using the `status` command:

```
# sudo systemctl status [SERVICE_NAME]
sudo systemctl status my-service
```

Tutorial

Here are some tasks that will give you a chance to use the commands shown above.
NGINX will be used as an example for these commands; you don't need to fully

understand what NGINX is, just that it's a web-server and, in this case, is managed as a systemd service.
For these tasks, its best to have a fresh install of Debian or Ubuntu

Install NGINX

We can use `apt` to get our NGINX server downloaded and configured as a systemd service. We'll also install curl, for getting a response from the NGINX server:

```
sudo apt install -y nginx curl
```

Check the Service Status

Make sure that NGINX is running by checking its status:

```
sudo systemctl status nginx
```

Stop the Service

First, lets get a response from the server:

```
curl localhost
```

You should see a basic HTML (web page) response.
Now, stop the service:

```
sudo systemctl stop nginx
```

If we use curl to try and get a response from the server, it should fail:

```
curl localhost
```

Check the Service Status

We can now see that the service has been stopped, by checking its status:

```
sudo systemctl status nginx
```

Enable and Start the Service

Make sure that the NGINX service will run on startup by enabling it.
The service can be started back up at the same time too:

```
sudo systemctl enable --now nginx
```

Check the status to see that the service is back up and running:

```
sudo systemctl status nginx
```

Make a Configuration Change

Edit the `/etc/nginx/nginx.conf` using `sudo`, or as the `root` user, and enter the following:

```
events {}
http {
    server {
        location / {
            return 200 "Reload Worked!\n";
        }
    }
}
```

Then, reload the NGINX service:

```
sudo systemctl reload nginx
```

We can then check that the configration has taken effect on the server, by making a request to it (again using curl):

```
curl localhost
```

The server should reply with `Reload Worked!`

Cleanup

Undo the changes we made by running the following:

```
sudo systemctl disable nginx
sudo systemctl stop nginx
sudo apt purge -y nginx
```

Exercises

There are no exercises for this module.