# #90DaysOfDevOps Challenge - Day 7 - Understanding package manager and systemctl

Welcome to Day 7 of the #90DaysOfDevOps challenge. In today's challenge, we will explore package managers in Linux, learn about the powerful `systemctl` command, and understand the role of `systemd` in managing system services. Let's dive into these topics and enhance our understanding of package management and service management in Linux.

## Package Managers in Linux

Package managers are essential tools for installing, updating, and removing software packages on Linux distributions. They provide a centralized repository of packages and handle dependencies, ensuring smooth and efficient software installation.

### APT (Advanced Package Tool)

APT (Advanced Package Tool) is a widely used package management system primarily used in Debian-based distributions such as Ubuntu. It offers a user-friendly command-line interface to interact with the package management system.

With APT, you can perform various operations, including package installation, update, and removal. APT automatically resolves dependencies, ensuring that all required packages are installed correctly.

### YUM (Yellowdog Updater Modified)

YUM (Yellowdog Updater Modified) is another popular package management system used in Red Hat-based distributions like CentOS. It simplifies software package management and dependency handling.

YUM provides a command-line interface for managing packages. You can search for packages, install, update, and remove them using the YUM commands. It takes care of resolving package dependencies, ensuring a smooth installation process.

Both APT and YUM greatly simplify the process of managing software packages, making it efficient and hassle-free for DevOps engineers working on different Linux distributions.

## Systemctl and Systemd

`Systemctl` is a command-line utility used to manage system services in Linux distributions that adopt the `systemd` init system. `Systemd` is a system and service manager that provides advanced features such as process management, logging, and service dependencies.

`Systemd` replaces the traditional SysV init system and offers enhanced functionality and control over system services. It enables parallel service startup, efficient dependency management, and better monitoring and logging capabilities.

Key concepts related to `systemctl` and `systemd` include:

- **Service Units**: Systemd uses service units, which are configuration files with a `.service` extension, to define and manage services. These units describe the properties and behaviour of a service.
- **Service Management**: Systemctl offers commands like `start`, `stop`, `restart`, `enable`, and `disable` to manage services. These commands allow you to control the lifecycle and behaviour of services on your system.
- **System Status**: You can use `systemctl status` to obtain detailed information about a service, including its current status, enabled or disabled state, and recent log entries. This feature facilitates troubleshooting and monitoring of services.

`Systemctl` and `systemd` provides a modern and efficient approach to service management on Linux systems. They offer greater control, flexibility, and consistency in managing services, making it easier for DevOps engineers to handle various service-related tasks.

## Task 1: Installing Docker and Jenkins Using Package Managers

Let's now put our knowledge of package managers into action by installing Docker and Jenkins on Ubuntu using package managers.

### Installing Docker on Ubuntu

To install Docker on Ubuntu using APT, follow these steps:

### Set up the repository

1. Update the `apt` package index and install packages to allow `apt` to use a repository over HTTPS:

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
```

2. Add Docker's official GPG key:

```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

3. Use the following command to set up the repository:

```
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-
by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

4. Update the `apt` package index:

```
sudo apt-get update
```

5. Install Docker Engine, containerd, and Docker Compose.

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-
buildx-plugin docker-compose-plugin
```

6. Start Docker:

```
sudo systemctl start docker
```

## Installing Jenkins on Ubuntu using APT

To install Jenkins on Ubuntu using APT, follow these steps:

1. Jenkins requires **Java** in order to run, so we'll update the Debian apt repositories, install OpenJDK 11, and check the installation with the commands:

```
sudo apt update
sudo apt install openjdk-11-jre
java -version
openjdk version "11.0.12" 2021-07-20
OpenJDK Runtime Environment (build 11.0.12+7-post-Debian-2)
OpenJDK 64-Bit Server VM (build 11.0.12+7-post-Debian-2, mixed mode,
sharing)
```

2. Next, let's install the Jenkins Long Term Support release by using the below commands:

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key |
sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

3. Start the Jenkins service:

```
sudo systemctl start jenkins
```

4. Enable Jenkins to start on system boot:

```
sudo systemctl enable jenkins
```

5. Jenkins should now be installed and running on your Ubuntu system. You can access it by opening your web browser and navigating to `http://localhost:8080`.

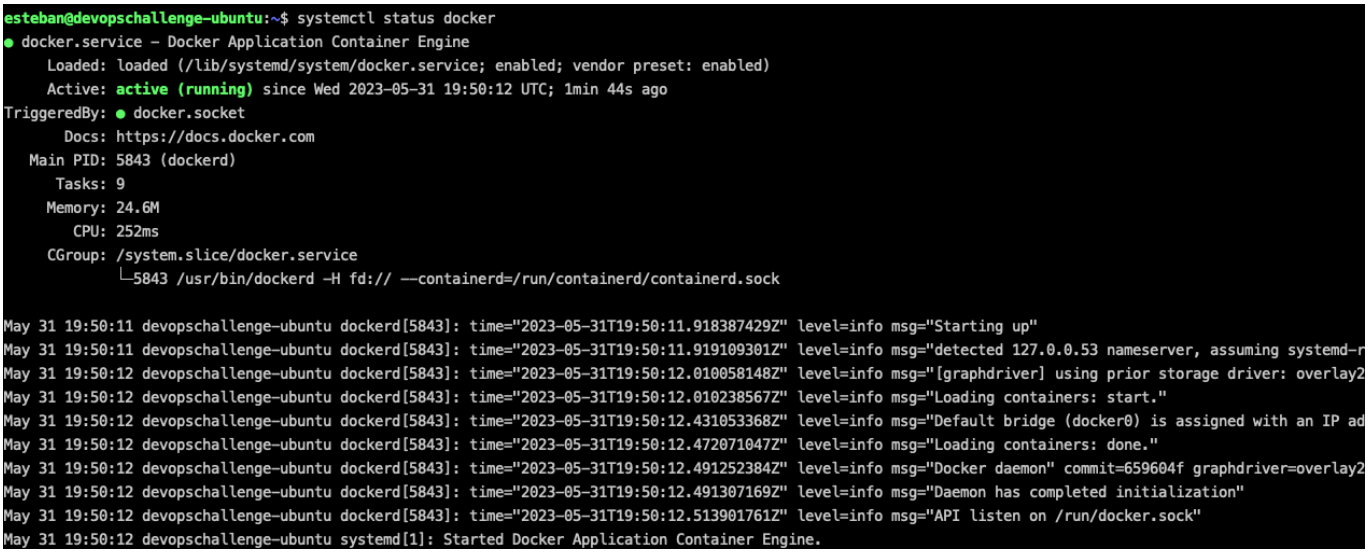
## Task 2: Checking the Status of the Docker Service

Now that Docker is installed, let's check the status of the Docker service using `systemctl`.

To check the status of the Docker service, run the following command:

```
systemctl status docker
```

The output will provide information about the current status of the Docker service, including whether it is running, enabled, and any recent log entries.




## Task 3: Stopping the Jenkins Service

Next, let's stop the Jenkins service and capture before and after screenshots.

To stop the Jenkins service, run the following command:

```
sudo systemctl stop jenkins
```

Before stopping Jenkins:

```
esteban@ubuntu-server-template:~$ systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-06-01 20:10:41 UTC; 7min ago
   Main PID: 12211 (java)
      Tasks: 48 (limit: 4571)
     Memory: 1.2G
        CPU: 48.585s
     CGroup: /system.slice/jenkins.service
             └─12211 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Jun 01 20:10:23 ubuntu-server-template jenkins[12211]: a4e9dc9974af489da1aa28815d552def
Jun 01 20:10:23 ubuntu-server-template jenkins[12211]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jun 01 20:10:23 ubuntu-server-template jenkins[12211]: *************************************************************
Jun 01 20:10:23 ubuntu-server-template jenkins[12211]: *************************************************************
Jun 01 20:10:23 ubuntu-server-template jenkins[12211]: *************************************************************
Jun 01 20:10:41 ubuntu-server-template jenkins[12211]: 2023-06-01 20:10:41.413+0000 [id=29]        INFO        jenkins.InitReactorRunner$1#onAttained: Complete
Jun 01 20:10:41 ubuntu-server-template jenkins[12211]: 2023-06-01 20:10:41.489+0000 [id=22]        INFO        hudson.lifecycle.Lifecycle#onReady: Jenkins is f
Jun 01 20:10:41 ubuntu-server-template systemd[1]: Started Jenkins Continuous Integration Server.
Jun 01 20:10:42 ubuntu-server-template jenkins[12211]: 2023-06-01 20:10:42.055+0000 [id=46]        INFO        h.m.DownloadService$Downloadable#load: Obtained
Jun 01 20:10:42 ubuntu-server-template jenkins[12211]: 2023-06-01 20:10:42.056+0000 [id=46]        INFO        hudson.util.Retrier#start: Performed the action
```

After stopping Jenkins:

```
esteban@ubuntu-server-template:~$ systemctl status jenkins
○ jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: inactive (dead) since Thu 2023-06-01 20:18:55 UTC; 1s ago
    Process: 12211 ExecStart=/usr/bin/jenkins (code=exited, status=143)
   Main PID: 12211 (code=exited, status=143)
     Status: "Jenkins stopped"
        CPU: 48.821s

Jun 01 20:18:55 ubuntu-server-template jenkins[12211]: 2023-06-01 20:18:55.411+0000 [id=24]        INFO        jenkins.model.Jenkins$16#onAttained: Completed t
Jun 01 20:18:55 ubuntu-server-template jenkins[12211]: 2023-06-01 20:18:55.411+0000 [id=24]        INFO        jenkins.model.Jenkins#_cleanUpDisconnectComputer
Jun 01 20:18:55 ubuntu-server-template jenkins[12211]: 2023-06-01 20:18:55.428+0000 [id=24]        INFO        jenkins.model.Jenkins#_cleanUpShutdownPluginMana
Jun 01 20:18:55 ubuntu-server-template jenkins[12211]: 2023-06-01 20:18:55.429+0000 [id=24]        INFO        jenkins.model.Jenkins#_cleanUpPersistQueue: Pers
Jun 01 20:18:55 ubuntu-server-template jenkins[12211]: 2023-06-01 20:18:55.488+0000 [id=24]        INFO        jenkins.model.Jenkins#_cleanUpAwaitDisconnects:
Jun 01 20:18:55 ubuntu-server-template jenkins[12211]: 2023-06-01 20:18:55.489+0000 [id=24]        INFO        hudson.lifecycle.Lifecycle#onStatusUpdate: Jenki
Jun 01 20:18:55 ubuntu-server-template jenkins[12211]: 2023-06-01 20:18:55.496+0000 [id=24]        INFO        o.e.j.s.handler.ContextHandler#doStop: Stopped w
Jun 01 20:18:55 ubuntu-server-template systemd[1]: jenkins.service: Deactivated successfully.
Jun 01 20:18:55 ubuntu-server-template systemd[1]: Stopped Jenkins Continuous Integration Server.
Jun 01 20:18:55 ubuntu-server-template systemd[1]: jenkins.service: Consumed 48.821s CPU time.
```

## Task 4: Understanding systemctl vs. service

`systemctl` and service are both command-line utilities used for managing services in Linux. However, there are some differences between them.

- `systemctl` is the primary command-line utility for managing services in Linux distributions that use `systemd`. It provides more advanced features and functionalities, such as service dependency management and improved logging.
- `service` is a backward-compatible command that works with both `systemd` and the older SysV init system. It is still available on many systems but is gradually being replaced by `systemctl`.

To compare the output of systemctl and service for a specific service, let's take the example of Docker.

Run the following commands to check the status of the Docker service using both systemctl and service:

```
systemctl status docker
service docker status
```

Observe the differences, if any, in the output of these commands. This exercise will help you understand the variations between systemctl and service and their usage in different Linux distributions.

Congratulations! You've completed Day 7 of the #90DaysOfDevOps challenge. Today, you gained insights into package managers, learned how to install Docker and Jenkins using APT, explored systemctl and systemd, and performed various tasks related to service management.

Stay tuned for Day 8, where we'll dive into Basic Git & GitHub for DevOps Engineers. Keep up the excellent work in your DevOps journey!