Podman in Windows: Installation Process and Comparison to Docker

Introduction

Podman is an open-source container engine designed to be a daemonless alternative to Docker, It provides the capability to run containers without needing a central daemon process, which can enhance security and simplify management. This report outlines the process of installing Podman directly on Windows without using a virtual machine and compares Podman to Docker.

Installation Process of Podman on Windows

Podman can now be installed directly on Windows using a native Windows installer. Here are the steps to install Podman on Windows:

Prerequisites

1. Windows 10 or Later: Ensure your Windows version is 10 or later with the latest updates installed.

2. Windows Subsystem for Linux 2 (WSL2): Although Podman can run natively, some features might still rely on WSL2.

Step-by-Step Installation

1. Download the Installer:

- Podman can be downloaded from the [Podman.io](https://podman.io/) website.

- Download the latest Windows installer (e.g., `podman-setup.exe`).

2. Run the Installer:

- Locate the downloaded installer and double-click to run it.

- Follow the installation prompts, which will guide you through the setup process.

3. Configure Podman:

- After installation, you may need to configure Podman. Open PowerShell and initialize the Podman environment:

podman machine init

podman machine start

- This sets up a Podman machine that uses a lightweight VM in the background for container management.

4. Verify the Installation:

- Check that Podman is installed correctly by running the following command in PowerShell:

podman --version

podman info

- You should see the installed Podman version output.

5. Running a Container:

- To test Podman, try running a simple container:

podman run --rm -it hello-world

- This should pull the `hello-world` image and run it, displaying a welcome message.

Other Podman Commands

podman pull [image-name], Pull a container image from a container registry.

podman images, List all downloaded container images.

podman run [options] image [command] [args...], Create and start a container from an image.

podman ps, List all currently running containers.

podman ps –a, List all containers (both running and stopped).

podman stop container\_name\_or\_id, Stop a running container.

podman rm container\_name\_or\_id, Remove a container.

podman rmi image\_name\_or\_id, Remove a container image.

podman build -t image\_name, Build a container image from a Dockerfile.

Comparison to Docker

1. Architecture:

- Podman: Daemonless architecture. Each container runs as a child process of the Podman command, which can enhance security and reduce the single point of failure.

- Docker: Relies on a central daemon (`dockerd`) to manage containers, which acts as a single point of management but also as a potential single point of failure.

2. Rootless Containers:

- Podman: Supports rootless containers natively, allowing non-privileged users to run containers without root permissions, improving security.

- Docker: Requires additional configuration to support rootless containers and traditionally runs with elevated privileges, which can be a security concern.

3. Compatibility:

- Podman: Aims to be compatible with Docker CLI commands, allowing many Docker commands to be used interchangeably with Podman. However, some Docker-specific features might not be available.

- Docker: The industry standard with broad compatibility and a wide range of features and integrations.

4. Use Cases:

- Podman: Ideal for environments where security and rootless operation are critical, such as in multi-user systems and development environments where users do not have root access.

- Docker: Preferred in environments requiring extensive tooling and integrations, such as CI/CD pipelines, and where ease of use and wide community support are important.

Using a daemonless container management system like Podman can be considered better or safer compared to daemon-based systems like Docker for several reasons:

Security

1. Rootless Operation:

- Podman: Allows containers to run as non-root users, reducing the risk associated with running processes with elevated privileges. This minimizes the potential impact of a security breach.

- Docker: Typically requires the Docker daemon to run with root privileges, which can be a security risk if the daemon or any container running under it is compromised.

2. Reduced Attack Surface:

- Podman: Without a central daemon, there are fewer potential targets for attacks. Each command runs in user space, reducing the chance of a single point of failure being exploited.

- Docker: The central daemon (`dockerd`) must be protected, as it has broad control over the system and can become a single point of attack.

Stability and Reliability

1. No Single Point of Failure:

- Podman: Since there is no daemon, the failure of one container or command does not affect others. Each container is managed independently.

- Docker: The Docker daemon is a single point of failure; if it crashes, all containers managed by it can be affected.

2. Independent Operations:

- Podman: Commands are executed independently, so issues with one operation do not cascade to others. This isolation increases overall system stability.

- Docker: The daemon's state can impact all container operations, meaning issues with the daemon can have widespread effects.

Flexibility

1. Simpler Permission Management:

- Podman: Users can run containers with their own user permissions, making it easier to manage in multi-user environments without giving all users elevated privileges.

- Docker: Requires careful management of user permissions and access controls to ensure security when running the daemon as root.

2. Easier Troubleshooting:

- Podman: Troubleshooting is often simpler because each command and container operates independently. Logs and errors are specific to individual commands.

- Docker: Troubleshooting can be more complex due to the central daemon, which handles all container operations and can produce more complex logs and error states.

Compliance and Policy

1. Policy Compliance:

- Podman: Easier to comply with policies that restrict root access and require minimal privileges for containerized applications.

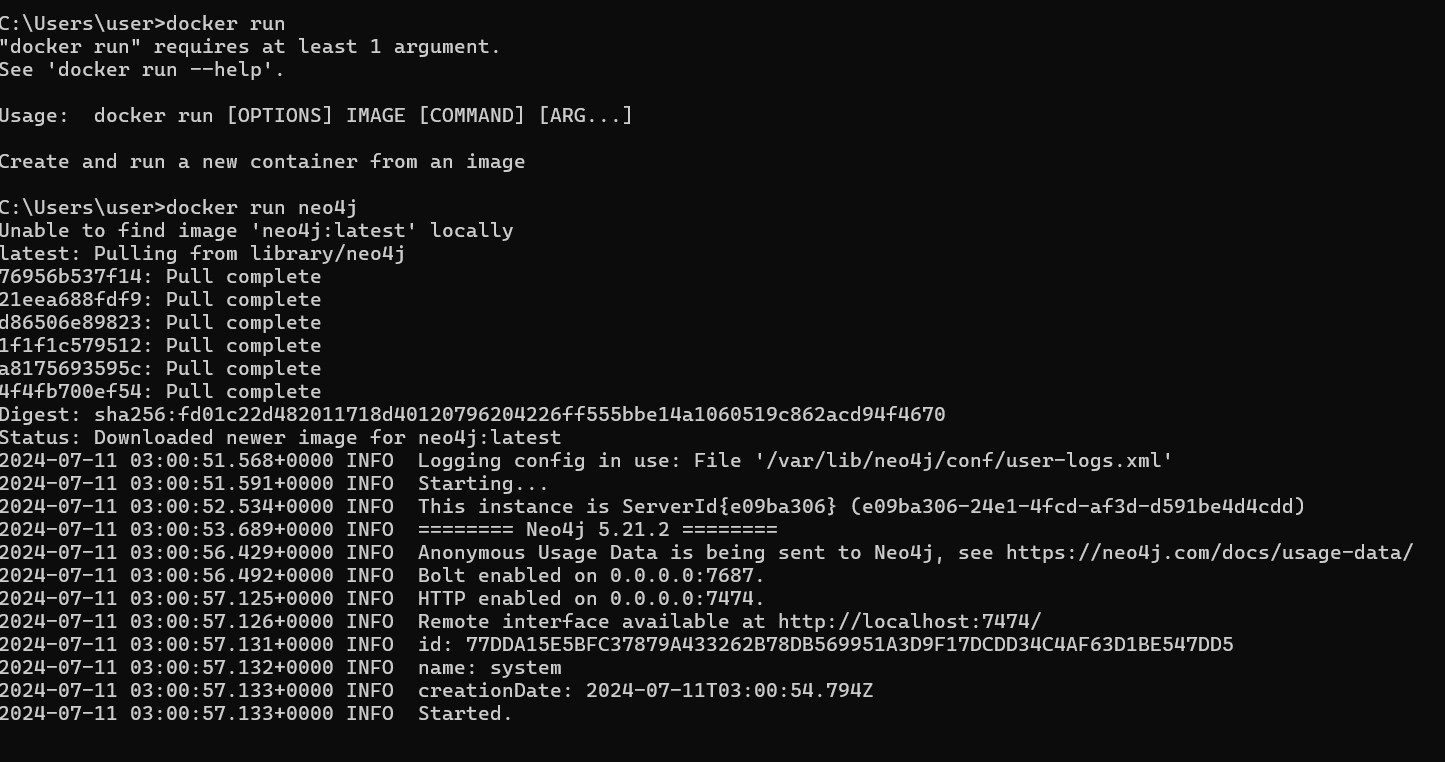
- Docker: Requires more effort to configure and maintain compliance with such policies due to the root-requiring daemon.

2. Audit and Monitoring:

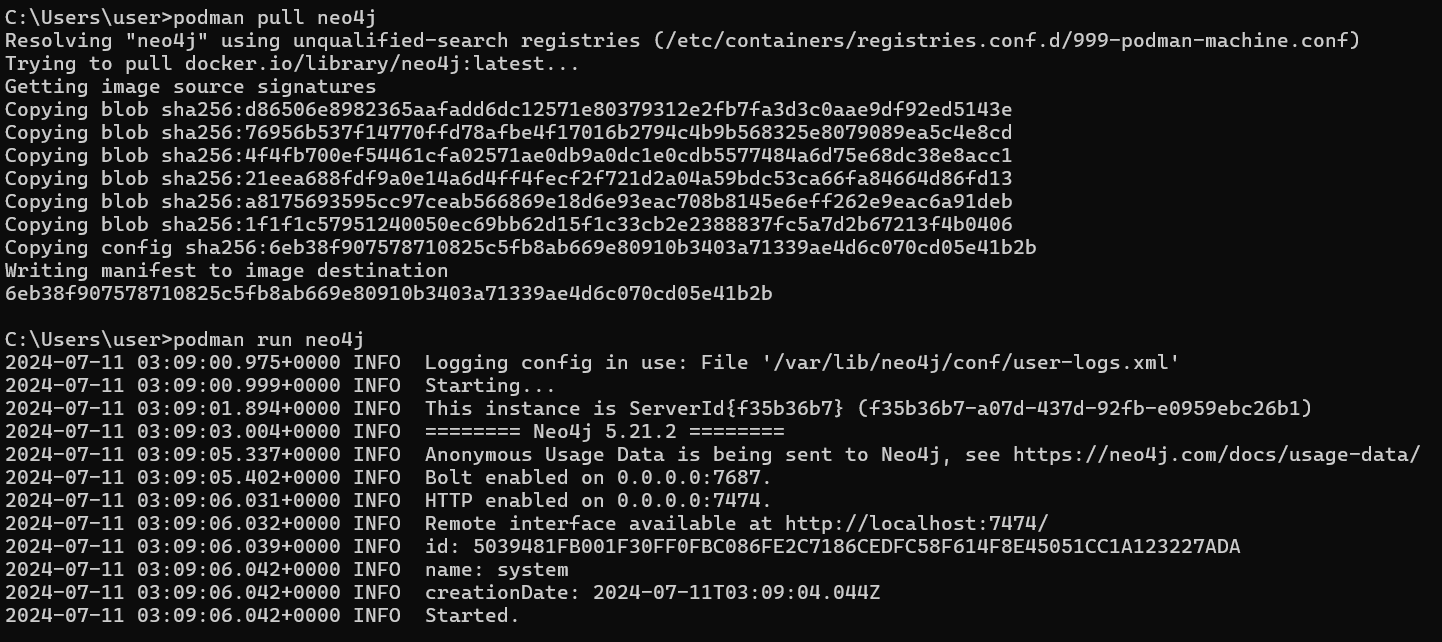
- Podman: Each container process can be audited and monitored individually, making it straightforward to track activities and ensure compliance with security policies.

- Docker: Centralized logging and monitoring of the daemon can complicate audit processes and make it harder to isolate specific container actions.

Pulling a container in Docker and Podman(NEO4J)



Podman



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Step** |  |  | | --- | |  | | |  | | --- | | **Docker 🐳** |  |  | | --- | |  | | |  | | --- | | **Podman 🍶** |  |  | | --- | |  | | | **Notes ⭐** | | --- |  |  | | --- | |  | |
| Command to Pull an Image | docker pull <image> | podman pull <image> | Both use similar commands |
| Example Command | docker pull neo4j | podman pull neo4j | Both pull images from Docker Hub by default |
| Default Image Source | Docker Hub | Docker Hub, but can configure other registries | Can configure other registries for both |
| Time to Pull an Image | Similar, depends on network and image size | Similar, depends on network and image size | Performance may vary slightly based on implementation details and caching |
| Command to Run an Image | docker run <options> <image> | podman run <options> <image> | Both use similar commands |
| Example Command | docker run -d -p 7474:7474 -p 7687:7687 neo4j | podman run -d -p 7474:7474 -p 7687:7687 neo4j | Both support similar options for running containers |
| Underlying Technology | Docker Daemon | Direct container runtime (no daemon) | Podman runs in rootless mode by default |
| Time to Start a Container | Quick startup time, depends on image size | Quick startup time, depends on image size | Similar performance, but Podman might be faster without daemon overhead |

Conclusion

Podman provides a secure and flexible alternative to Docker, particularly appealing in scenarios requiring rootless containers and enhanced security. While Docker remains the more popular and widely supported option, Podman’s compatibility with Docker commands and its daemonless architecture make it an attractive choice for certain use cases. The installation of Podman on Windows has become more straightforward with native support, making it accessible for users looking for a Docker alternative.