Using Rosetta to run x86-64 Docker Containers and Binaries in Linux virtual machines with Parallels Desktop

- Parallels Desktop for Mac Pro Edition
- Parallels Desktop for Mac Business Edition

Introduced in Parallels Desktop 19, a new feature allows running x86-64 binaries in Linux virtual machines, for purposes such as running x86 Docker containers in Linux virtual machines on Mac computers with Apple silicon, or some of the x86-64 software binaries directly.

Based on Apple's Rosetta capabilities in virtualization framework emulating x86-64 binaries, Parallels Desktop integrates this feature to make it as convenient to use as possible.

Note: the feature is available in Parallels Desktop Pro or Business Edition only.

Overview

Rosetta is a translation layer by Apple, that allows running code written for x86-64 architecture on Apple silicon systems, where it would otherwise not be possible to do so. Starting with macOS Ventura 13 the virtualization framework allows utilizing Rosetta in a Linux virtual machine for a range of applications, primarily for running x86-64 Docker containers in the virtual machine or using the applications that cannot be run on an Arm-based system otherwise.

Setting up automatically

The fastest way to get started using this feature is to download a pre-configured **Ubuntu 22.04.02** virtual machine that will have Rosetta set up, dependencies updated, and Docker ready to go.

To install it, open **Control Center** in Parallels Desktop > create a new virtual machine from the list of Free Systems and select **Download Ubuntu with x86_64 emulation** > click **Continue** to create and start the virtual machine.

You can now start using Docker in the Terminal immediately to create x86-64 containers, as the Docker command-line interface (CLI) is pre-installed in the appliance as well, or try to install software, provided it conforms to the limitations. See the Troubleshooting section should you encounter an issue.

Manual set up

If you're not planning to use Ubuntu, or require a different distribution version, for instance, you can also set it up manually. This option should work for the Debian distributions and requires just a few extra steps to complete, such as adding the amd64 repositories and enabling the architecture. Here, you can examine the manual setup on the Ubuntu example:

1. Create your new Ubuntu virtual machine. Learn more in KB 128445.

2. Let **Parallels Tools** install if created from an image file and **shut down** the virtual machine.

3. Open the virtual machine configuration and enable the **Use Rosetta to run x86-64 binaries** feature in the **Hardware > CPU & Memory** section.

4. Start up your virtual machine and add amd64 repositories. This can be done in one of the following ways:

Adding the repositories manually:

You can add those repositories manually, by editing the /etc/apt/sources.list file. Changes that are required to add [arch=amd64] to the existing list, and add a separate record for the [arch=amd64]. The end result can be noted in the **example** sources.list file, but in general, it should arrive at the following pairs for each repository instance:

```
deb [arch=arm64] http://ports.ubuntu.com/ubuntu-ports/ jammy-updates main
restricted
deb [arch=amd64] http://archive.ubuntu.com/ubuntu/ jammy-updates main
restricted
```

Note: You may note lines starting with **deb-src** in the file. We do not recommend making any changes to them due to the unforeseen consequences that may be caused by changes to repositories for packages in the source format.

One of the ways to edit the **sources.list** file is to use the following command in the Terminal:

```
sudo nano /etc/apt/sources.list
```

After modifying the file, run the following command to add the amd64 architecture:

```
sudo dpkg --add-architecture amd64
```

And then update the changes with:

```
sudo apt update
```

Adding the repositories using the script:

In the case of Ubuntu Linux, we have created a short script that performs the modifications for the sources.list file automatically and enables the amd64 architecture. You can download and run the script, after which only running apt update is required to get the functionality running:

- Download the rosetta_x86_sources.sh file to a folder accessible on your virtual machine.
- In your virtual machine, open the containing folder and right-click anywhere to select "Open in Terminal":

- Execute the following command to run the script that will enable the repositories and add the amd64 architecture:

Note: Running the script with "**disable**" option will revert the changes.

```
sudo ./rosetta_x86_sources.sh enable
```

- And then update the changes with:

```
sudo apt update
```

5. After adding the repositories by using one of the methods, you will be able to run the x86-64 Docker containers and binaries using Rosetta.

Note: if your distribution is different from Ubuntu, the sources records and enabling the architecture command may vary, but the process would remain the same. We recommend using the sources file for Ubuntu as a reference for adding the corresponding repositories for different distributions.

Troubleshooting

Installing x86-64 applications can sometimes return an error in the system due to the mentioned dependencies' conundrum. Here, we will examine this in the example of Microsoft Edge, which is available in Linux only as an x84-64 application. However, in the case of Edge, the procedure allows circumventing the issue caused by the dependencies, and it's recommended in case of any application that exhibits the same behaviour, unfortunately, due to the nature of different dependencies it's not guaranteed to work in every single case.

In the case of the Microsoft Edge example, while running an installation, you may encounter the following issue:

To troubleshoot the dependency issue, you can run the following command in Terminal:

```
sudo apt-get -f install -y
```

The command will detect the failed installation attempt and make some changes, after which try running the installation again, and in the case of Microsoft Edge, it completes successfully.

Known Limitations

Due to the way Rosetta translation functions, while it possesses impressive emulation capabilities, it's not always as simple as running any x86-64-only package on Linux virtual machines and having them work like they would on a native Intel system.

Installing x86-64 Packages

In terms of installing packages, for instance, **Snap Packages** are not supported in this scenario due to them containing all their dependencies in the package, and hence not being able to make use of the translation as it strictly runs off x86-64 base, while the virtual machine remains an Arm-based Linux. There is no possibility of mitigating this at the present time.

Additionally, as the system itself remains an Arm-based Linux, certain packages might have dependencies that may or may not allow compiling them through Rosetta to run on an Arm system. This particular scenario can sometimes

be mitigated by using an additional command before the installation. See the Troubleshooting section for more details.

Distributions Limitations

At this point, due to the overwhelming popularity of usage, this feature is fully tested and integrated on Ubuntu virtual machines, and for that particular distribution, a pre-configured virtual machine is available to download, and by extension, it's likely will also work the same way on other Debian based Linux distributions when set up manually, but due to the fact that are still different in their own way, you may encounter an issue.

We are, however, also not forgetting about Red Hat and other distributions, but due to certain differences in security restrictions, we are still in the process of adapting this process to function as seamlessly as possible, and currently, setting up Rosetta is not completely supported by Red Hat and RHEL based Linux distributions.

If you encounter any additional limitations running specific binaries or distributions, we'd love to hear about them and your usage scenario, and you can use this Forum thread to share your feedback.

© 2024 Parallels International GmbH. All rights reserved. Parallels, the Parallels logo and Parallels Desktop are registered trademarks of Parallels International GmbH. All other product and company names and logos are the trademarks or registered trademarks of their respective owners.