

CONTENTS

Prerequisites

Step 1 – Following Best Practices for Application Management

Step 2 – Configuring Automatic Upgrades

Step 3 – Updating and Livepatching the Kernel

Conclusion

RELATED

How To Automate Your Node.js Production Deployments with Shipit on CentOS 7

[View](#) 

How to Set Up SSH Keys on Rocky Linux 9

[View](#) 

// Tutorial //

How to Keep Rocky Linux 8 Servers Updated

Published on August 2, 2022

Automated Setups

Rocky Linux

Rocky Linux 8

By [Alex Garnett](#)

Senior DevOps Technical Writer





Not using Rocky Linux 8?

Choose a different version or distribution.

Rocky Linux 8 ▼

Introduction

In this tutorial, you'll explore some best practices for keeping your Rocky Linux 8 server stack up to date. Just as with [network security hardening](#), there are many steps you can take to ensure your servers will continue to work securely without future intervention.

There are a number of tools and configurations that you can apply to most Rocky Linux servers beyond what is configured for you automatically. If you're doing your own server administration, it can be quite disruptive and error-prone to patch all of your environments manually.

This tutorial will cover:

- Testing graceful reboots following best practices for application management, to minimize any complications from maintenance updates
- Configuring automatic updates for most of the packages and libraries running on your machine
- Live kernel patching, and other best practices around kernel updates

Prerequisites



- A Rocky Linux 8 server and a non-root user with sudo privileges. You can learn more about how to set up a user with these privileges in our [Initial](#)



COOKIE PREFERENCES

[with Rocky Linux 8](#) guide.

Step 1 – Following Best Practices for Application Management

A fundamental part of configuring your server for automatic upgrades is ensuring that all of the applications running on the server are able to restart correctly after unplanned downtime or a reboot. Linux package managers are designed to run non-disruptively in the background so that necessary maintenance does not come with any additional overhead. Despite this, one of the most common reasons for not having a good update strategy in place is being concerned about how your server will behave after being restarted.

Wherever possible, the applications in your stack should be managed by your server's init system, which on most modern Linux distributions including Rocky Linux is `systemd`. `Systemd` provides the [systemctl](#) command for interacting with your running services and automatically restarting them as needed. Virtually all software that is installed via your package manager and designed to run in the background should automatically provide a `systemd` service and a configuration unit file as a best practice.

When running your own software, or software deployed from Git repositories, it is not a bad idea to write your own unit files to integrate with `systemd`. As a lightweight alternative, you may prefer to use a tool like [supervisor](#). You can also use your system's [cron](#) scheduler with the `@reboot` syntax.

After your configuration is in place, make sure to test it through reboots. You can reboot by running `sudo shutdown now -r`, which will cleanly stop your running processes and reboot immediately. You can also specify a time in `hh:mm`, or a number of minutes from now, instead of `now` to schedule a reboot in the future. Production deployments should generally not require your attention after any unplanned outages, and all necessary services and endpoints should come back up automatically.

Now that you've ensured that your environment won't have any trouble persisting through maintenance reboots, in the next step, you'll learn how to schedule automatic upgrades.

Step 2 – Configuring Automatic Upgrades

Rocky's package manager, `dnf`, has two main ways of performing a full system upgrade. You can run `dnf upgrade` without specifying a package to upgrade every package on your system. You can also run `dnf upgrade-minimal` to upgrade every package only to the latest bugfix or security patch release, which will perform necessary maintenance

while avoiding any potential breaking changes upstream. You can read more about `dnf` commands from the [dnf documentation](#).

Rocky also provides a unique tool called `dnf-automatic` to automatically retrieve and install security patches and other essential upgrades for your server. You can install it using `dnf`:

```
$ sudo dnf install dnf-automatic -y
```

[Copy](#)

`dnf-automatic` is not enabled automatically after installation. Instead, it provides several different services that you can register with Systemd to customize its behavior. These are:

- **dnf-automatic** – this service follows the `dnf-automatic` configuration file options in `/etc/dnf/automatic`.
- **dnf-automatic-notifyonly** – this overrides the configuration file by only notifying you of available updates and not installing them.
- **dnf-automatic-download** – this overrides the configuration file by only downloading packages and not installing them.
- **dnf-automatic-install** – this overrides the configuration file by always installing downloaded packages.

For this tutorial, you will be enabling the `dnf-automatic-install` service, but first, you'll make one change to the `dnf-automatic` configuration file. Open the file using `vi` or your favorite text editor:

```
$ sudo vi /etc/dnf/automatic
```

[Copy](#)

`/etc/dnf/automatic.conf`

```
[commands]
# What kind of upgrade to perform:
# default                               = all available upgrades
# security                             = only the security upgrades
upgrade_type = security
random_sleep = 0

# Maximum time in seconds to wait until the system is on-line and able to
# connect to remote repositories.
network_online_timeout = 60

# To just receive updates use dnf-automatic-notifyonly.timer

# When updates should be downloaded when they are available, by
# dnf-automatic.timer, notifyonly.timer, download.timer and
# install.timer override this setting.
download_updates = yes
```



```
# Whether updates should be applied when they are available, by
# dnf-automatic.timer, notifyonly.timer, download.timer and
# install.timer override this setting.
apply_updates = no
...
```

Most of the options in this file correspond to the various overrides provided by the different Systemd services. An exception is the `upgrade_type` option, which is set to `default` by default. If you're going to be implementing automatic upgrades (the most proactive configuration), it's a good idea to only install security upgrades by default, to avoid unexpected changes in functionality. Change the `upgrade_type` value to `security` as in the above example, then save and close the file. If you are using `vi`, you can save and quit by entering `:x`.

Now you can enable the service using `systemctl`:

```
$ sudo systemctl enable dnf-automatic-install.timer
```

Copy

You can check to ensure that the `dnf-automatic-install` service is running correctly using `systemctl`:

```
$ sudo systemctl status dnf-automatic-install
```

Copy

Output

```
• dnf-automatic-install.service - dnf automatic install updates
  Loaded: loaded (/usr/lib/systemd/system/dnf-automatic-install.service; static; ven
  Active: inactive (dead)

Jul 14 21:01:03 droplet-name dnf-automatic[40103]: No security updates needed, but 15
Jul 14 21:01:03 droplet-name systemd[1]: dnf-automatic-install.service: Succeeded.
```

Web hosting without headaches. Try Cloudways with \$100 in free credit! Sign up →

We're
hiring

Blog

Docs

Get
Support

Sales



Tutorials Questions Learning Paths For Businesses Product Docs Social Impact

You can check on the details of that timer using `systemctl cat`.

```
$ sudo systemctl cat dnf-automatic-install.timer
```

Copy



Output



```
[Unit]
Description=dnf-automatic-install timer
# See comment in dnf-makecache.service
ConditionPathExists=!/run/ostree-booted
Wants=network-online.target

[Timer]
OnCalendar=*-*-* 6:00
RandomizedDelaySec=60m
Persistent=true

[Install]
WantedBy=timers.target
```

By default the service is configured to run around 6:00 every day. You should not need to change this value.

You should now have solutions in place to ensure all of the packages on your server receive essential security updates without any additional intervention. In the last step, you'll learn how to keep your kernel updated, and how best to handle server reboots when they are necessary.

Step 3 – Updating and Livepatching the Kernel

Less often than other packages, you will need to update your system's kernel. The Linux kernel contains (almost) all running hardware drivers and is responsible for most low-level system interactions. Kernel updates are usually only necessary if there is a high-profile vulnerability to address, if you need to make use of a publicized new kernel feature, or if your kernel has become so old that there is a greater risk of accumulated bugs and vulnerabilities that you may not be aware of.

There is no universal method of automatically scheduling Linux kernel updates. This is because kernel updates have historically required a full system reboot, and scheduling reboots is impossible without making assumptions about your environment. Many servers are expected to provide as close to 24/7 availability as possible, and a reboot can take an unknown amount of time, or require manual intervention.

Most production deployments require additional complexity around rebooting like this to ensure service availability. For example, you might use a load balancer to automatically redirect traffic to servers that can provide identical functionality in a [horizontally scaled deployment](#) while they are individually rebooted in sequence, to avoid any visible downtime.

Enabling Livepatch to Ensure Server Uptime During Kernel Updates



To avoid downtime during kernel upgrades, you can use a feature of the Linux kernel called live patching. This feature makes it possible to implement kernel updates without rebooting. There are two major maintainers for kernel live patches in the Rocky Linux ecosystem: Red Hat's `kpatch`, which provides live patching for Red Hat Enterprise Linux, and [KernelCare](#) who support Rocky Linux in addition to most other major Linux distributions. Both require registration to use.

Due to the way that Red Hat Enterprise Linux's licensing model works, you will not be able to receive live kernel patches via `kpatch` when running Rocky Linux — Rocky acts in many ways like an unlicensed version of Red Hat, and there is no way to enroll in kernel live patching without running a full licensed Red Hat instance. However, the method to enable it is the same on Rocky or RHEL. First, install the `kpatch-dnf` package:

```
$ sudo dnf install kpatch-dnf
```

[Copy](#)

Next, run `dnf kpatch auto` to automatically subscribe to live patching services:

```
$ sudo dnf kpatch auto
```

[Copy](#)

Output

```
Last metadata expiration check: 0:00:06 ago on Thu 14 Jul 2022 09:12:07 PM UTC.  
Dependencies resolved.  
Nothing to do.  
Complete!
```

`kpatch` will check for configured patching services, and, finding none, will exit gracefully. On Red Hat, this would instead perform a license check and register with the Red Hat `kpatch` servers. On Rocky, you may want to investigate KernelCare as a commercial support option instead.

Conclusion

In this tutorial, you explored multiple strategies to keep your Rocky Linux servers updated automatically. You also learned some of the nuances of package repositories, kernel updates, and handling server reboots. These are all important fundamentals of DevOps and of working with servers more broadly, and nearly all production configurations build on these core concepts.

Next, you may want to learn to use [Watchtower](#) in order to automatically update Docker container images.



Thanks for learning with the DigitalOcean Community. Check out our offerings for compute, storage, networking, and managed databases.

[Learn more about us](#) →

About the authors



[Alex Garnett](#) Author

Senior DevOps Technical Writer

Still looking for an answer?

[Ask a question](#)

[Search for more help](#)

Was this helpful?

[Yes](#)

[No](#)



Comments

Leave a comment

B *I* U ☺ 📎 🖼️ ✎ H₁ H₂ H₃ ☰ ☷ “,” ⓘ 🗑️ <>



Leave a comment...



This textbox defaults to using **Markdown** to format your answer.

You can type `!ref` in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

[Sign In or Sign Up to Comment](#)



This work is licensed under a Creative Commons Attribution-NonCommercial- ShareAlike 4.0 International License.

Try DigitalOcean for free

Click below to sign up and get **\$200 of credit** to try our products over 60 days!

[Sign up →](#)

Popular Topics

[Ubuntu](#)

[Linux Basics](#)

[JavaScript](#)

[Python](#)

[MySQL](#)




[Docker](#)

[Kubernetes](#)

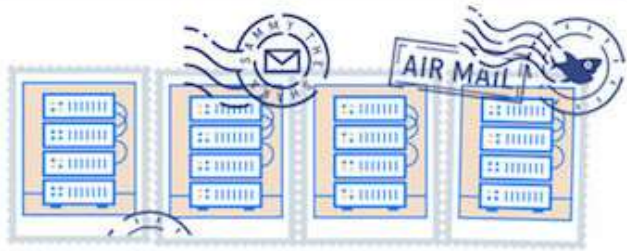
[All tutorials →](#)

[Free Managed Hosting →](#)



-  Congratulations on unlocking the whale ambience easter egg! Click the whale button in the bottom left of your screen to toggle some ambient whale noises while you read.
-  Thank you to the [Glacier Bay National Park & Preserve](#) and [Merrick079](#) for the sounds behind this easter egg.
-  Interested in whales, protecting them, and their connection to helping prevent climate change? We recommend checking out the [Whale and Dolphin Conservation](#).

Reset easter egg to be discovered again / Permanently dismiss and hide easter egg



Get our biweekly newsletter

Sign up for Infrastructure as a Newsletter.

[Sign up →](#)




Hollie's Hub for Good

Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

[Learn more →](#)



Become a contributor

You  aid; we donate to tech nonprofits.

[Learn more →](#)

Featured on Community

[Kubernetes Course](#) [Learn Python 3](#) [Machine Learning in Python](#)
[Getting started with Go](#) [Intro to Kubernetes](#)

DigitalOcean Products

[Cloudways](#) [Virtual Machines](#) [Managed Databases](#) [Managed Kubernetes](#)
[Block Storage](#) [Object Storage](#) [Marketplace](#) [VPC](#) [Load Balancers](#)

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow – whether you're running one virtual machine or ten thousand.

[Learn more →](#)[About](#)[Products](#)[Products Overview](#)[Community Solutions](#)[Tutorials](#)[Website Hosting](#)[Contact](#)[Support](#)

Leadership	Droplets	Q&A	VPS Hosting	Sales
Blog	Kubernetes	CSS-Tricks	Web & Mobile Apps	Report Abuse
Careers	App Platform	Write for DOnations	Game Development	System Status
Customers	Functions	Currents Research	Streaming	Share your ideas
Partners	Cloudways	Hatch Startup Program	VPN	
Channel Partners	Managed Databases	deploy by DigitalOcean	SaaS Platforms	
Referral Program	Spaces	Shop Swag	Cloud Hosting for Blockchain	
Affiliate Program	Marketplace	Research Program	Startup Resources	
Press	Load Balancers	Open Source		
Legal	Block Storage	Code of Conduct		
Security	Tools & Integrations	Newsletter Signup		
Investor Relations	API	Meetups		
DO Impact	Pricing			
	Documentation			
	Release Notes			
	Uptime			

© 2023 DigitalOcean, LLC. All rights reserved.

