Configuring Unattended Upgrades on Debian


benheater.com/configuring-unattended-upgrades-on-debian

0xBEN April 21, 2023

Installation

```
sudo apt install -y unattended-upgrades apt-listchanges
```

Bash

With the installation complete, we're primarily concerned with the configuration of two files:

- `/etc/apt/apt.conf.d/50unattended-upgrades`
- `/etc/apt/apt.conf.d/20auto-upgrades`

When you manually install upgrades with `apt`, you'd probably run something like, `apt update && apt upgrade -y --autoremove && apt autoclean`. All you're really doing is automating this process with the use of `unattended-upgrades`.

Configuring the Service

```
sudo nano /etc/apt/apt.conf.d/50unattended-upgrades
```

Bash

Configuring Origin Patterns

Making Sense of Origin Patterns

You use ***Origin Patterns*** to control ***which sources*** `unattended-upgrades` will read and install upgrades from. You can technically use a wildcard pattern in the origin pattern to upgrade from all, but this will also upgrade the operating system major version, unless ***apt pinning*** is configured to prevent it.

```
Unattended-Upgrade::Origins-Pattern {
                // Comments
                  // ...
                  // ...
"origin=${distro_id},codename=${distro_codename}";
                }
```

Plain text

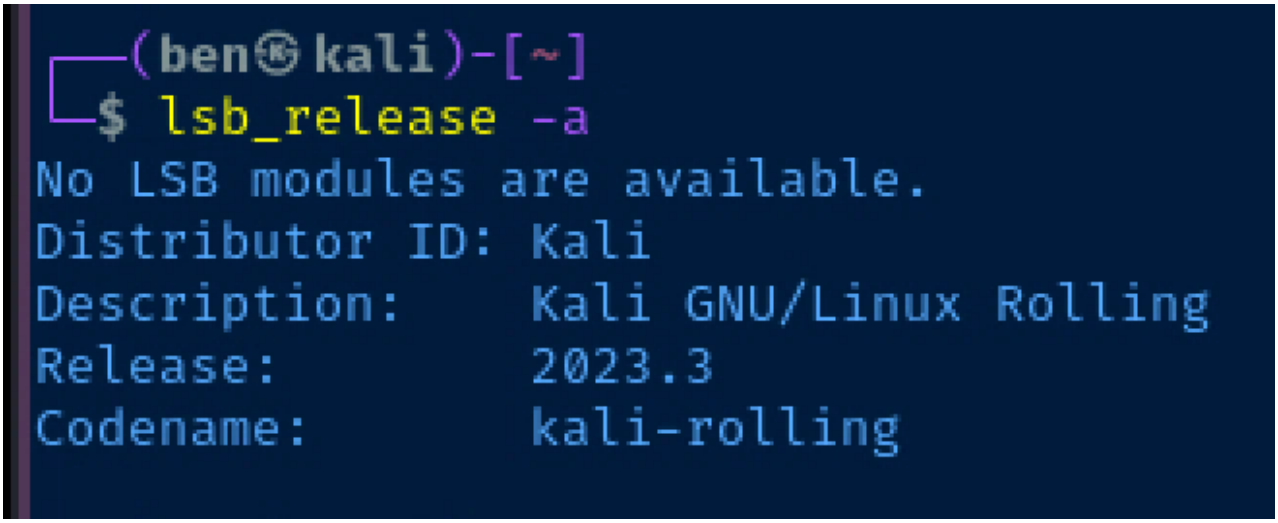
You'd probably see something like this in a default installation of Unattended Upgrades

Analyzing an Origin Pattern String

What are the two variable names we see here?

- `${distro_id}`
- `${distro_codename}`

These will be sourced in from `lsb_release` (using Kali as an example):



- `${distro_id}` = `Kali`
- `${distro_codename}` = `kali-rolling`
- So effectively, we've got an origin pattern of: `"origin=Kali,codename=kali-rolling";`

Forming an Origin Pattern String

If you look at the ***comments at the top of the config file***, it spells out pretty clearly:

- The parts of an origin pattern
- How to find the origin patterns for your sources in `/etc/apt/sources.list` and `/etc/apt/sources.list.d/`

```
//   a,archive,suite (eg, "stable")
//   c,component     (eg, "main", "contrib", "non-free")
//   l,label         (eg, "Debian", "Debian-Security")
//   o,origin        (eg, "Debian", "Unofficial Multimedia Packages")
//   n,codename      (eg, "jessie", "jessie-updates")
//     site          (eg, "http.debian.net")
// The available values on the system are printed by the command
// "apt-cache policy", and can be debugged by running
```

Plain text

You can use the following values in your origin pattern:

- `a=` or `archive=`

- `c=` or `component=`
- `l=` or `label=`
- `o=` or `origin=`
- `n=` or `codename=`
- or a straight `site` URL

It also notes that you can see your configured sources by running `apt-cache policy` (again an example from Kali):

```
┌──(ben㉿kali)-[~]
└─$ apt-cache policy
Package files:
 100 /var/lib/dpkg/status
     release a=now
 500 http://packages.microsoft.com/repos/code stable/main armhf Packages
     release o=code stable,a=stable,n=stable,l=code stable,c=main,b=armhf
     origin packages.microsoft.com
 500 http://packages.microsoft.com/repos/code stable/main arm64 Packages
     release o=code stable,a=stable,n=stable,l=code stable,c=main,b=arm64
     origin packages.microsoft.com
 500 http://packages.microsoft.com/repos/code stable/main amd64 Packages
     release o=code stable,a=stable,n=stable,l=code stable,c=main,b=amd64
     origin packages.microsoft.com
 500 https://packages.microsoft.com/repos/edge stable/main amd64 Packages
     release o=edge stable,a=stable,n=stable,l=edge stable,c=main,b=amd64
     origin packages.microsoft.com
 500 http://http.kali.org/kali kali-rolling/non-free amd64 Packages
     release o=Kali,a=kali-rolling,n=kali-rolling,c=non-free,b=amd64
     origin http.kali.org
 500 http://http.kali.org/kali kali-rolling/contrib amd64 Packages
     release o=Kali,a=kali-rolling,n=kali-rolling,c=contrib,b=amd64
     origin http.kali.org
 500 http://http.kali.org/kali kali-rolling/main amd64 Packages
     release o=Kali,a=kali-rolling,n=kali-rolling,c=main,b=amd64
     origin http.kali.org
Pinned packages:
```

ℹ️

If you haven't noticed, you can pretty much just copy and paste straight from each source. ***HOWEVER, PLEASE NOTE:***unattended-upgrades does not accept the b= (branch) of the specifc source. So, while it may seem there are duplicates of certain sources, it's almost certainly likely due to multiple branches.

Example 1: Visual Studio Code

Let's use this output from `apt-cache policy` as an example:

```
500 http://packages.microsoft.com/repos/code stable/main armhf Packages
    release o=code stable,a=stable,n=stable,l=code stable,c=main,b=armhf
    origin packages.microsoft.com
500 http://packages.microsoft.com/repos/code stable/main arm64 Packages
    release o=code stable,a=stable,n=stable,l=code stable,c=main,b=arm64
    origin packages.microsoft.com
500 http://packages.microsoft.com/repos/code stable/main amd64 Packages
    release o=code stable,a=stable,n=stable,l=code stable,c=main,b=amd64
    origin packages.microsoft.com
```

Plain text

These are the branches of the ***Visual Studio Code*** `apt` repositories on my Kali Linux box. I ***do not*** need to configure three separate origin patterns for these. These sources ***are all the same*** with the exception of the branch denoting the CPU architecture.

So, if I wanted to ***create an origin pattern*** that will ***automatically upgrade Visual Studio Code*** to the latest version, I can ***copy and paste*** this single origin pattern:

```
Unattended-Upgrade::Origins-Pattern {
        // Comments and other content
           // Removed for clarity
  "origin=${distro_id},codename=${distro_codename}";
"o=code stable,a=stable,n=stable,l=code stable,c=main";
                      };
```

Plain text
Showing origin pattern to automatically upgrade VS Code

Example 2: Microsoft Edge and Brave Browser

Again, looking at the output from `apt-cache policy`:

```
500 https://packages.microsoft.com/repos/edge stable/main amd64 Packages
    release o=edge stable,a=stable,n=stable,l=edge stable,c=main,b=amd64
    origin packages.microsoft.com
500 https://brave-browser-apt-release.s3.brave.com stable/main amd64 Packages
    release o=Brave Software,a=stable,n=stable,l=Brave Browser,c=main,b=amd64
    origin brave-browser-apt-release.s3.brave.com
```

Plain text

To keep my Microsoft Edge and Brave Browser automatically upgraded, I could copy and paste these origin patterns into the config file:

```
Unattended-Upgrade::Origins-Pattern {
        // Comments and other content
            // Removed for clarity
   "origin=${distro_id},codename=${distro_codename}";
  "o=code stable,a=stable,n=stable,l=code stable,c=main";
  "o=edge stable,a=stable,n=stable,l=edge stable,c=main";
"o=Brave Software,a=stable,n=stable,l=Brave Browser,c=main";
                        };
```

Plain text
VS Code, Microsoft Edge, and Brave Browser origin patterns

Configuring Upgrade Options

Now that the origin patterns have been added to `/etc/apt/apt.conf.d/50unattended-upgrades`, it's time to configure the rest of the service. As you scroll down the configuration file, you'll see additional options — some commented out with a `//` prefix.

Additional options that I'd set in the file would be:

```
Unattended-Upgrade::AutoFixInterruptedDpkg "true";
Unattended-Upgrade::InstallOnShutdown "false";
Unattended-Upgrade::Remove-Unused-Kernel-Packages "true";
Unattended-Upgrade::Remove-New-Unused-Dependencies "true";
Unattended-Upgrade::Remove-Unused-Dependencies "true";
Unattended-Upgrade::Automatic-Reboot "true";
Unattended-Upgrade::Automatic-Reboot-WithUsers "true";
Unattended-Upgrade::Automatic-Reboot-Time "04:00";
Unattended-Upgrade::OnlyOnACPower "false";
```

Plain text

A couple things to note here:

- I auto-reboot at `04:00`, change this to a time that suits your needs
- I haven't shown you how to configure `//Unattended-Upgrade::Mail "";`, but I'd encourage you to look into this if you're running a production server and/or want to track changes.
- Read the `// Comments` above each section to gain a better understanding about each option.

Configuring the Schedule

```
sudo nano /etc/apt/apt.conf.d/20auto-upgrades
```

Bash

```
// How often (in days) to apt update
APT::Periodic::Update-Package-Lists "1";

// How often (in days) to download new packages
APT::Periodic::Download-Upgradeable-Packages "1";

// How often (in days) to clean the apt clean
APT::Periodic::AutocleanInterval "7";

// How often (in days) to run unattended-upgrades
APT::Periodic::Unattended-Upgrade "1";
```

Plain text

Enable and Start the Service

```
sudo systemctl enable --now unattended-upgrades
```

Bash

We want the daemon to start at boot and also start immediately

Troubleshooting

Dry Run Mode

```
sudo unattended-upgrades -d --dry-run
```

Bash

Run unattended-upgrades with output in dry run mode

You can use dry run mode to see if there are any problems with your origin patterns, or problems with your overall configuration. Dry run mode will operate like a normal run, but will not install upgrades.

Manual Run

```
sudo unattended-upgrades -d
```

Bash

Run unattended-upgrades manually with debug output

Running unattended-upgrades in debug mode will run a full upgrade of your packages based on your origin patterns. Running it in debug mode manually can be useful when you want to gauge how the program will behave in future unattended runs.

References

UnattendedUpgrades - Debian Wiki

Debian Wiki

Written by

0xBEN

View all posts
More from 0xBEN