# AppImage

an AppImage is a container for a single application. It is a single binary containing an entire application. It provides everything the application needs from the OS within the appimage itself, so it has no dependencies. Basically in a way similar to a docker container.

You can simply download any AppImage to your linux Machine, make it executable and run it. That's fantastic but there are some downsides compared to applications installed via your OS's packet manager:

- You need to manually copy the app image somewhere useful
- you need to manually integrate it into your desktop launcher
- you need to manually download and install updates of your appimage

So basically, using AppImage is like using windows XP, where you download shady setup exes and manually look for updates etc. So why bother? Well, the big upside of AppImages is what it brings to the developer: It makes it a lot simpler for a developer to provide a binary version of his application, because he does not have to familiarize himself with the various build systems of all those many linux distributions out there. He does not have rely on stone-age library versions, just to be sure his app will compile on an old RHEL or Debian as well to make sure he can package it for those distributions as well. He can simply write his application on his favorite Linux environment, compile it and package it and it will run on all other Linux variants and most versions right away. Docker does the same for services, Snap does almost exactly the same as AppImage but it provides a packet manager, security features etc.. That's better than AppImage, but it's also more complicated, because Snap is not broadly accepted by all Distributions yet, an app developer would only reach users of very few distributions yet. Because AppImage needs nothing, anybody can run it without installing an entire ecosystem first.

So in the end, AppImage is a workaround for an age old issue of Linux: that it is hard for a developer to supply distribution agnostic binaries.

I see more and more apps being supplied as app images only being available as an app image. This is okay as long as it's something you use occationally, but as some regularly used apps are being packed that way, we need a way to integrate app images easily into our desktop environment and to keep them updated as well.

## App Image helper programs

Here are some helper programs i have found so far, that help to overcome the Windows XP feeling of AppImages:

### Appimaged

appimaged is a tool provided by the AppImage people themselves which helps integrating app images into your desktop environment. It monitors your directories where you usualy store excutables (directories listed in your $PATH environment variable) and if it finds new AppImages, it adds them to

your desktop launcher. It also removes apps that where removed from those directories. If you right click an app in your launcher, you get further options to do various things with the appimage, including its removal or creating a portable home for it etc. It also makes any appimage it finds executable.

what I don't like about it is, it also searches your Downloads directory, to detect an app image as soon as its downloaded. This is of course very convenient, however, it makes your system a mess! also keep in mind, that it automatically marks everything executable. I see my Downloads folder as a temporary storage for stuff I download. It is the first folder i delete when i run out of diskspace, so not really the place where i want to keep my AppImage files that i need every day.

Of course i can manually copy them to a different folder, but there we ware, i need to open a file browser again and start to manage my appimages.

Sadly there is no configuration option that would allow us to change the directories in which appimages are searched, and looking at a comment from the developer to a request of a user to have a configuration option to disable the automatic creation of the Applications folder, he answered saying, that configurability adds complexity and he wants to avoid that.. a valid thaught, however, it does not help in this case. I personally believe that configurability does not need to add complexity, as long as everything runs with good default settings. but that's an opinion just like any other ;)

### Installation

To install it, you simply download it, **copy it to its permanent location**, mark it executable and run it. it will then create a user systemd service in `~/.config/systemd/user/appimaged.service` which makes sure it autostarts every time you boot your machine. It will then scan various directories for you and add all the appimages it found to your launcher.

It is important that you copy it first to the location where you want to keep it, as it will link to itself in the systemd service it creates. So if you run it the first time from your download folder, it will link to there and as soon as you clean up your downloads the next time, appimaged is gone! A good location for appimaged would bi in `~/Applications`. If you don't have this directory yet, just create it, appimaged will do it for you anyway once it has been started for the first time.

if you have started appimaged already from a folder where you didn't want it to stay, simply stop the systemd service

```
systemctl --user stop appimaged.service
```

then copy the appimage file to the defintive location and start it from there, it will automatically re-create the systemd entry

### Workaround for Download folder issue

looking at the sourcecode of appimaged.go here is the list of directories which are searched for appimages:

```
var candidateDirectories = []string{
    xdg.UserDirs.Download,
```

```
    xdg.UserDirs.Desktop,
    home + "/.local/bin",
    home + "/bin",
    home + "/Applications",
    "/opt",
    "/usr/local/bin",
}
```

I agree to all of them except for the “Download” and “Desktop” which at least I personally use as more a temporary kind of storage. Luckily the path for these two folders is determined via XDG and XDG uses some environment variables to customize it. Namely `XDG_CONFIG_DIRS` which points to directories holding further path information. so maybe we can mess with that.. this has not worked yet..

```
mkdir -p ~/.config/appimaged/{Empty,xdg}
cat > ~/.config/appimaged/xdg/user-dirs.dirs <<EOF
XDG_DESKTOP_DIR="\$HOME/.config/appimaged/Empty"
XDG_DOCUMENTS_DIR="\$HOME/.config/appimaged/Empty"
XDG_DOWNLOAD_DIR="\$HOME/.config/appimaged/Empty"
XDG_MUSIC_DIR="\$HOME/.config/appimaged/Empty"
XDG_PICTURES_DIR="\$HOME/.config/appimaged/Empty"
XDG_PUBLICSHARE_DIR="\$HOME/.config/appimaged/Empty"
XDG_TEMPLATES_DIR="\$HOME/.config/appimaged/Empty"
XDG_VIDEOS_DIR="\$HOME/.config/appimaged/Empty"
EOF
mkdir -p ~/.config/systemd/user/appimaged.service.d
cat > ~/.config/systemd/user/appimaged.service.d/override.conf <<EOF
[Service]
Environment="XDG_CONFIG_DIRS=/home/psuter/.config/appimaged/xdg"
EOF
```

## AppImageLauncher

AppImageLauncher is another take on installing and integrating app images. It makes use uf the fact, that fresly downloaded files don't have the executable flag set. So when you double-click or “open” them, your OS is looking for an application to handle this type of files. AppImageLauncher will set it self up as the default handler for AppImages. Once it is started, it provides you with options to install the app image or run it only once. It will do the integration for you and store it to a reasonable path (`~/Applications`) for you. That's more like it :) Now we are at least at the level of a `setup.exe` from the old windows days :)

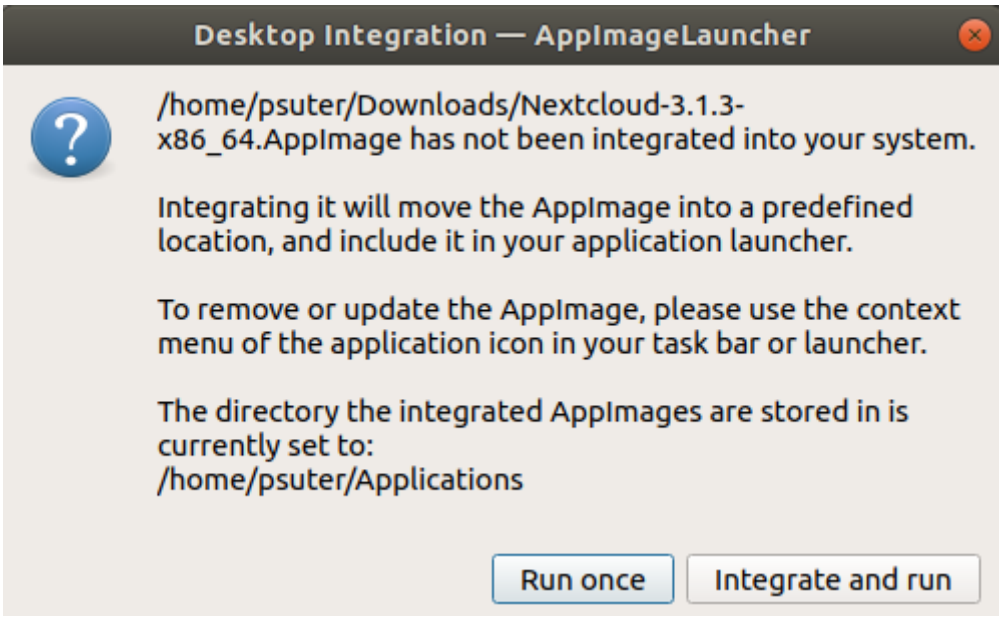
covneiently, AppImageLauncher itself comes packaged for many distributions, because it needs to mess with your distribution's specific file paths etc. anyway. I chose to go with the PPA for ubuntu so `apt` can take care of updates etc.:

```
sudo add-apt-repository ppa:appimagelauncher-team/stable
sudo apt update
sudo apt install appimagelauncher
```

Ironitcally, the PPA does not provide packages for Ubuntu bionic (18.04) the now second newest LTS

release even though the deb file which can be manually downloaded and installed from the GitHub page is named `bionic` because it is built for bionic and newer releases of Ubuntu.. This demonstrates again why AppImages are used and are a good thing after all :) So if you ware on Bionic, simply download the `.deb` from the latest release and install it. you will have to update it yourself.. or simply update your ubuntu and then use the PPA :)

once you have it installed, simply double clicke an appimage and you will be presented with a screen that says it all:




if your appimage has embedded udpate information, you will see a option to check for updates if you right-click the app in your launcher, otherwise there is only the uninstall option.

### Uninstalling

AppImageLauncher can be uninstalled via `apt` or whatever package manager you used to install it. However, it can be that it leaves a little bit of garbage behind, which might get in your way if you want to use the beforementioned `appimaged`. after uninstalling it, run

```
rm ~/.config/systemd/user/default.target.wants/appimagelauncherd.service
```

to remove a left over symlink from your system. you need to reboot before you can try to run appimaged

From:
http://wiki.psuter.ch/ - **pswiki**

Permanent link:
**http://wiki.psuter.ch/doku.php?id=appimages&rev=1613898370**

Last update: **21.02.2021 10:06**