NixOS For Large Businesses

By John Umbriac

NixOS was originally released in 2006 by Eelcro Dolstra as a master's thesis project. (1) The project gained popularity over many years, and the NixOS Foundation was created in 2015 to help it and associated projects grow (2). Today, NixOS has not only the most packages of any repository by a wide margin, but also the most "fresh" packages (ones that include recent updates and security fixes) by far (5).

Traditional linux distributions have users run a sequence of commands to install and configure packages. This causes that computer to obtain a unique state (configuration) that is extremely difficult to reproduce again. What makes NixOS unique is its use of the Nix package manager which does all package management, and most of the configuration for those packages, through the configuration.nix file. This allows a specific state to be replicated across many different machines by simply transferring one file (3). Beyond that, each time the file is changed, the user runs the "nixos-rebuild" command which generates an entirely new self contained collection of the packages declared in that file, and adds that new state as a boot option. If a breaking change is made to the configuration.nix file, rolling back is as easy as choosing the previous entry in grub (6).

There are options to install NixOS through either a GUI installer, or through the command line on a minimal bootable ISO. This example uses that minimal ISO (6):

1. Download minimal ISO from https://nixos.org/download/
2. Flash that ISO file to an empty USB drive using the following command(run this command on an unmounted USB drive directly e.g. of="/dev/sdb" not "/dev/sdb1")

```
sudo dd bs=4M conv=fsync oflag=direct status=progress
if=<path-to-file>/nixos-minimal-24.05.5919.32e940c7c420-x86_64-linux.iso
of=<path-to-USB-device>
```

3. Boot into the usb device (different for every computer)
4. Select the first option in the boot loader
5. Configure networking by enabling and configuring NetworkManager or wpa_supplicant
6. Mount the partition to install NixOS on to /mnt

```
sudo mount /dev/<path-to-partition>
```

7. Make a boot directory and mount the boot partition to it

```
mkdir -p /mnt/boot
```

then

```
sudo mount -o umask=077 /dev<boot-parition> /mnt/boot
```

8. Enable swap (optional, but useful if adding large packages to configuration.nix)

```
swapon /dev/<path-to-swap-parition>
```

9. Generate an initial configuration.nix
   Note: If this is done before the boot directory is mounted, it will not auto detect some important settings!

```
nixos-generate-config --root /mnt
```

10. Edit the configuration.nix to your liking
    Note: The configuration generated is not usable as is, so make sure to set up things like a non nano text editor, another user in addition to root, and most importantly some form of networking!

```
sudo vim /mnt/etc/nixos/configuration.nix
```

11. When complete, install that configuration on the partition. Look out for a promp asking to set the root password

```
sudo nixos-install
```

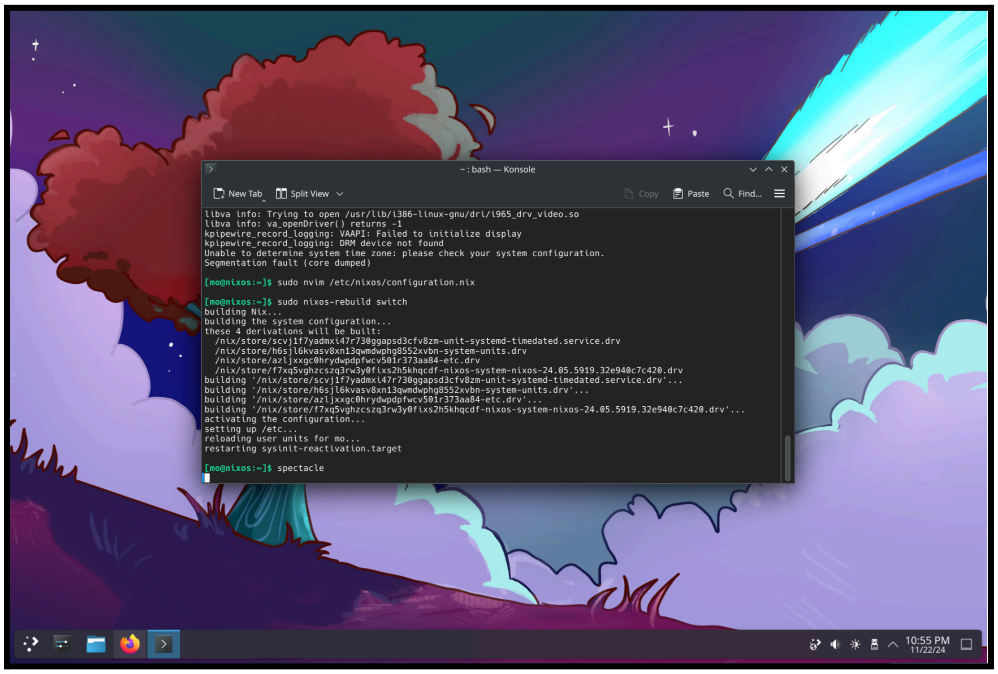
12. If a non-root user was created, a password must be created with this command

```
nixos-enter --root /mnt -c 'passwd <username>'
```

13. Reboot to enter the new system

```
reboot
```

When those steps are complete, if KDEPlasma is set as the desktop manager, then the desktop will look like Fig 1. KDEPlasma is a popular and functional desktop manager. Its main advantages over other desktop managers is its similarity to Windows, and its clean aesthetic.



```
libva info: Trying to open /usr/lib/i386-linux-gnu/dri/i965_drv_video.so
libva info: va_openDriver() returns -1
kpipewire_record_logging: VAAPI: Failed to initialize display
kpipewire_record_logging: DRM device not found
Unable to determine system time zone: please check your system configuration.
Segmentation fault (core dumped)

[mo@nixos:~]$ sudo nvim /etc/nixos/configuration.nix

[mo@nixos:~]$ sudo nixos-rebuild switch
building Nix...
building the system configuration...
these 4 derivations will be built:
  /nix/store/scvj1f7yadmxi47r730ggapsd3cfv8zm-unit-systemd-timedated.service.drv
  /nix/store/h6sjl6kvasv8xn13qwmdwphg8552xvbn-system-units.drv
  /nix/store/azljxxgc0hrydwpdpfwcv501r373aa84-etc.drv
  /nix/store/f7xq5vghzcszq3rw3y0fixs2h5khqcdf-nixos-system-nixos-24.05.5919.32e940c7c420.drv
building '/nix/store/scvj1f7yadmxi47r730ggapsd3cfv8zm-unit-systemd-timedated.service.drv'...
building '/nix/store/h6sjl6kvasv8xn13qwmdwphg8552xvbn-system-units.drv'...
building '/nix/store/azljxxgc0hrydwpdpfwcv501r373aa84-etc.drv'...
building '/nix/store/f7xq5vghzcszq3rw3y0fixs2h5khqcdf-nixos-system-nixos-24.05.5919.32e940c7c420.drv'...
activating the configuration...
setting up /etc...
reloading user units for mo...
restarting sysinit-reactivation.target

[mo@nixos:~]$ spectacle
```

Fig 1. Default Dark Mode KDEPlasma on NixOS

Nix, the NixOS Package manager is quite different from other options. Available packages can be searched on https://search.nixos.org/packages. These packages can then be **declared** in /etc/nixos/configuration.nix, or other linked .nix files to add, modify, or remove packages. When the user is doing editing their configuration, they simply run

```
sudo nixos-rebuild switch
```

To generate a build matching the new configuration, and switch to it. One advantage of this system is that builds are **atomic**. This means that if a build fails due to an error in the configuration or a power outage, it can safely be discarded, but if the build succeeds then it is set as the default build. This means there is no risk of system corruption during an upgrade. In addition to this, the new build is added to the boot menu alongside all previous builds. This makes it trivial to revert to the previous working build if something suddenly breaks.

NixOS supports temporarily trying out packages in a specific shell through the nix-shell command. This feature is useful for developers who want to try out a tool in a sandboxed environment before committing to it. Once that shell session is closed, the package is gone. Additionally, it is possible to point the nix-shell command at a separate configuration environment to effortlessly generate a completely sandboxed environment. This is extremely useful for developers who might want to use different versions of packages for different projects, but don't want to mess with the often lacking language specific tools that attempt to handle dependency versioning.

Overall, NixOS has the following advantages:

- Free - free to download and use
- Atomic - no broken states in between updates. A new build either fully succeeds and is used, or fails and is not
- Declarative - system's setup is exactly as specified in the config.
- Portable - one file will make the same setup on almost any machine
- Easy to fix - rolling back is as easy as one command, also packages will not break other packages
- Secure - Can't break something important or install something bad without root access

And the following disadvantages:

- Rigid - requires root access to install/remove packages (aka to deploy any modifications to configuration.nix)
- Unfamiliar - It's not widely used, even inside the linux community. Could be configured to act like Windows or MacOS or Ubuntu, but would require a lot of work
- Not friendly to new users - requires a competent user and good configuration.nix to get everything set up

NixOS was initially quite difficult for me to learn as I decided to essentially create my configuration.nix file from scratch, but once I got the basic things like a desktop manager, a browser, and a decent text editor in there I found it extremely easy to make changes. If anything went wrong during these upgrades, it was trivial to revert to a previous build. Because of this I spent a lot less time fiddling around trying to fix things that were broken.

There are two use cases where I think NixoS makes a lot of sense. The first is power users because it rewards people who take the time to set everything up exactly as the way they want it, and things won't just randomly break in a way that blocks you from doing what you need to. The second is large IT departments as they would have the resources to really understand and make the most of nix's configuration options, and imaging a machine is as simple as loading a text file, and running a command.

Overall, my experience with NixOS was positive. While the package management scheme is novel and requires some getting used to, it quickly becomes effortless, and much easier to organize than any other solution. This is enhanced by the large number of quality, up to date packages available. I think it is the future of package management.

Bibliography

1. https://www.usenix.org/legacy/event/hotos07/tech/full_papers/dolstra/dolstra_html/
2. https://en.wikipedia.org/wiki/NixOS
3. https://nixos.org/
4. https://nixos.org/blog/
5. https://repology.org/
6. https://nixos.org/manual/nixos/stable/#sec-installation