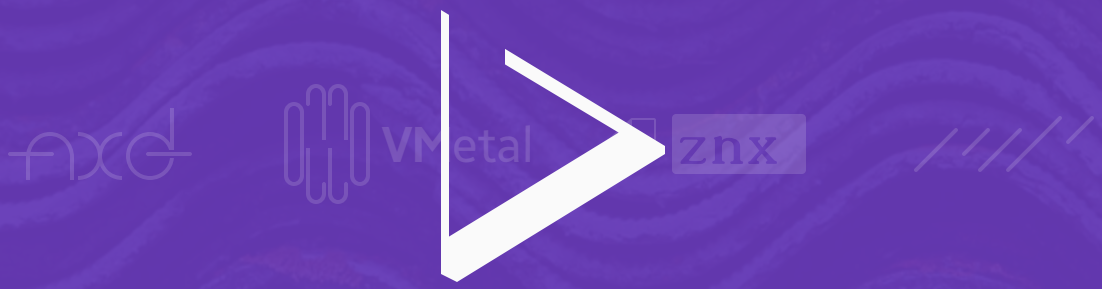
Frequently Asked Questions.

Reference guide

for the **Nitrux Operating System**.



First Edition April (2020)

Table of contents

What is Nitrux?

If you are new to Nitrux, or only casually acquainted with Linux based distributions, it can be difficult to understand how a Linux operating system compares with other computer systems that you may already be familiar. Hopefully, this page will help demystify Nitrux for new-comers.

Nitrux is an operating system based on Linux. Nitrux can be deployed without any need for a traditional installation. The operating system as a whole resides in a single file and directory on the partition ZNX_DATA of your device, making it easier to organize with your other data. We provide an operating system already preconfigured for the most common tasks, included is a web browser, an office suite, various utilities including a text editor, a file manager, a calculator, etc. Unlike a Live system, when Nitrux is deployed using znx *it keeps all the data in your home folder even after the next time you boot*. This feature is known as *persistence*. Using this feature of znx allows Nitrux to offer an **immutable** operating system. In other words, *no changes will take place in the root directory*.

Why Nitrux ≠ Ubuntu

Ubuntu is one of the largest Linux based desktop operating systems in the world. Linux is at the heart of Ubuntu and makes it possible to create versatile operating systems.

Nitrux takes the Ubuntu sources and brings it to a whole new level. We have built Nitrux with the goal of not depending on the Debian package manager (dpkg) or its extended set of tools APT (Advanced Packaging Tool) to manage the operating system. For example, to obtain new software users don't have to worry about missing dependencies, out of date repositories or conflicts with packages and when upgrading the operating system users won't have to worry about conflicting dependency problems, partitioning or data loss because of the inherent stability of performing transactional updates.

Contrary to what many people might think, we are not taking the vanilla Ubuntu distribution and performing a facelift.

With our constant focus on AppImages as *the primary method of obtaining software* now and in the future and znx as the *only* method of managing the operating system, the use of a package manager is not a central part of this distribution; such is the case that we have removed APT and dpkg from our operating system; instead we utilize the AppImage CLI Tool which is available from the Terminal by using the command *app* (lowercase). And, another notable difference is the shell that we employ in Nitrux, we do not make use of the Debian Almquist shell (dash) we utilize the MirBSD™ Korn Shell (mksh) and the Z Shell (zsh). Finally, users that aren't able to find software distributed as an AppImage can make use of *Homebrew* to install new software.

To make it clear though, *we are not focusing on building a Linux distribution centered around using a package manager, no matter which one it is, our focus is on pushing forward the use of AppImages.*

All of this has a massive impact on new users who expect Nitrux to be a replica of the Ubuntu experience with only minor aesthetic differences, realizing this is not correct users may feel confused and at odds when trying out this distribution for the first time. Our changes run deep, and even though we build Nitrux from Ubuntu sources, *Nitrux is not Ubuntu neither does it work like Ubuntu, nor it will be Ubuntu under the hood forever.*

What is the difference between the Stable and Development ISO?

The only difference between our two ISO is that the stable release is left untouched for a month unless it is to update it for security reasons. The development branch is merged into the stable branch at the end of the month as per our schedule.

As we mention in this FAQ, we don't guarantee stability for the development build as it is the branch where we commit changes for the next version of Nitrux. However, we do not force updates unto users, so if an update were to cause breakage unless the user applies the update, the development build will not break.

If the user decided to apply the updated and found it to cause breakage, they can use znx to rollback to the previous version of the ISO.

Why are the system requirements listed so high?

First, it must be understood that what is listed as the minimum requirements are not the same as what is listed as the recommended requirements. Minimum requirements indicate what we are considering to be the least powerful hardware setup to run Nitrux and still be able to use it without hurting the user experience that much. Yes, Nitrux could run in even lower-end hardware than what we list. However, the user experience would be utterly inadequate due to hardware limitations.

Therefore the recommended requirements are so that the user experience of Nitrux is optimal and the minimum requirements are so that user experience of Nitrux to be good enough. We have also included a list of the hardware that we used to create these measurements.

Why is there no support for 32-bit processors in Nitrux?

A big difference between 32-bit processors and 64-bit processors is the number of calculations per second they can perform, which affects the speed at which they can complete tasks. 64-bit processors can come in dual-core, quad-core, six-core, and eight-core versions for home computing. Multiple cores allow for an increased number of calculations per second that can be performed, which can improve the processing power and help make a computer run faster. Software programs that require many calculations to function smoothly can operate quicker and more efficiently on the multi-core 64-bit processors, for the most part.

Another big difference between 32-bit processors and 64-bit processors is the maximum amount of memory (RAM) that is supported. 32-bit computers support a maximum of 4 GB (232 bytes) of memory, whereas 64-bit CPUs can address a theoretical maximum of 18 EB (264 bytes). However, the practical limit of 64-bit CPUs (as of 2018) is 8 TB of addressable RAM.

High amounts of RAM are especially useful for software used in graphic design, engineering, and video editing as these programs have to perform many calculations to render their images.

One thing to note is that 3D graphics programs and games do not benefit much, if at all, from switching to a 64-bit computer, unless the program is a 64-bit program. A 32-bit processor is adequate for any application written for a 32-bit processor. In the case of computer games, you'll get a lot more performance by upgrading the video card instead.

64-bit processors have been available for almost twenty years and EFI motherboards since nearly ten. *We build Nitrux for the present and the future, not for the past*. If you are running a 32-bit computer, we strongly recommend using other Linux or BSD distributions that focus on supporting this type of hardware.

Will Nitrux support processor architectures other than x64?

Yes. It is in our plans to support processor architectures other than x86-64 (or *amd64*). We will work towards adding support for the following processor architectures in the future:

- ARM (AArch64).
- IBM's PowerPC/Power ISA (POWER8 and POWER9).
- RISC-V.

Will Nitrux support MBR storage devices for initialization?

You can plugin MBR storage devices to Nitrux without a problem and utilize them, however, when we refer to Nitrux not supporting MBR devices we are *explicitly talking about the storage device that is intended to be used with znx*.

znx only supports GPT drives, when znx initializes a new device it creates a new partition table utilizing the GUID Partition Table and then creates the ESP and the data partition.

Unlike MBR partitioned disks, data critical to platform operation is located in partitions instead of unpartitioned or hidden sectors. Also, GPT partitioned disks have redundant primary and backup partition tables for improved partition data structure integrity.

Though UEFI supports the traditional master boot record (MBR) method of hard drive partitioning, it doesn't stop there. It's also capable of working with the GUID Partition Table (GPT), which is free of the limitations the MBR places on the number and size of partitions.

MBR-based partition table schemes insert the partitioning information for (usually) four "primary" partitions in the master boot record (MBR) (which on a BIOS system is also the container for code that begins the process of booting the system). In a GPT, the first sector of the disk is reserved for a "protective MBR" such that booting a BIOS-based computer from a GPT disk is supported, but the boot loader and O/S must both be GPT-aware. Regardless of the sector size, the GPT header begins on the second logical block of the device.

GPT uses current logical block addressing (LBA) in place of the cylinder-head-sector (CHS) addressing used with MBR. Legacy MBR information is contained in LBA 0, the GPT header is in LBA 1, and the partition table itself

follows.

In summary, **no, Nitrux will not support MBR storage devices for initialization**.

Why does Nitrux not support Legacy BIOS and only support UEFI or EFI?

The lack of support for Legacy BIOS devices is a decision that we made based on what kind of features we wanted to add to our Linux distribution and the sort of hardware that we wanted to support for these features to work. The availability of EFI and subsequently UEFI devices has been mainstream for the past ten years, meaning that any future hardware that vendors release to the public will support UEFI exclusively and not Legacy BIOS.

As we explain further in this FAQ, it all comes down to how our distribution works; its management method and the features that we include which either make use of or depend on the hardware to support the use of EFI or UEFI firmware and the characteristics that either of these brings.

We are knowledgeable of hardware that would otherwise run this operating system without a problem but is otherwise unsupported because it did not include EFI firmware at least. Our decision is technical in that the limitations of Legacy BIOS did not allow for the support of our intended features in our Linux distribution to be feasible.

The UEFI specification describes the layout of an MBR partition table but does not mention the ESP.

It is theoretically possible to use the ESP on an MBR device. This method, however, is untested and it seems to produce more problems related to bootloader naming and NVRAM entries managed by efibootmgr. Furthermore, Intel plans to phase out support for the CSM by 2020. As such, we do not have any intention to spend resources on pursuing this method.

Also, EFI and subsequently UEFI motherboards have been available ca. 2010–2011 when Intel introduced its Nehalem architecture in 2008 and then when AMD launched its 800 chipset series in 2009 and later released it in 2010. These motherboards have been commonplace ever since, and this means that the kind of hardware that we are targeting is at least ten years old.

While rare there were EFI compatible motherboards for older platforms such as Socket 775 Intel processors.

Other reasons why we decided to support only UEFI devices are the following:

- **UEFI enables better use of bigger hard drives**. Though UEFI supports the traditional master boot record (MBR) method of hard drive partitioning, it doesn't stop there. It's also capable of working with the GUID Partition Table (GPT), which is free of the limitations the MBR places on the number and size of partitions. GPT ups the maximum partition size from 2.19TB to 9.4 zettabytes.

- **UEFI may be faster than the BIOS**. Various tweaks and optimizations in the UEFI may help your system boot more quickly it could before. For example: With UEFI you may not have to endure messages asking you to set up hardware functions (such as a RAID controller) unless your direct input is required, and UEFI can choose to initialize only individual components. The degree to which a boot is sped up will depend on your system configuration and hardware so that you may see a major or a minor speed increase.
- **Technical changes abound in UEFI.** UEFI has room for more useful and usable features than could ever be crammed into the BIOS. Among these are cryptography, network authentication, support for extensions stored on non-volatile media, an integrated boot manager, and even a shell environment for running other EFI applications such as diagnostic utilities or flash updates. Also, both the architecture and the drivers are CPU-independent, which opens the door to a wider variety of processors (including those using the ARM architecture, for example).

Therefore, Nitrux only supports systems that are classified as Class 2 and Class 3, for reference:

- **Class 0**: Non-UEFI platforms, the set of platforms based on traditional legacy BIOS. These platforms are not UEFI aware.
- **Class 1**: In the earlier days of EFI/UEFI when all the leading OSs were not EFI/UEFI aware a Compatibility Support Module (CSM) was used to present the traditional BIOS like interface. These platforms only booted to conventional, legacy OSs.
- **Class 2**: These platforms came about when EFI was adapted as UEFI industry standard and OS started adding support for UEFI. These platforms support booting using the traditional method of int19h, wherein BIOS loads the boot sector and hands off execution to the boot loader, as well as loading a UEFI boot loader application. Majority of platforms shipping today are Class 2 platforms.
- **Class 3**: These platforms support booting only using the UEFI defined method of loading the boot loader application from a specific location. Class 3 platforms do not sport a Compatibility Support Module. Any class 2 platform with CSM turned off functions like a Class 3 platform.

UEFI is the newer name and latest development of EFI.

To verify if your computer supports UEFI execute the following sequence of commands within a Terminal window:

```
1. [ -d /sys/firmware/efi ] && echo UEFI || echo BIOS
```

Does Nitrux support 32-bit UEFI?

The main thing we need to understand is a core concept of the UEFI Specification: the OS and firmware architecture need to match. For most non-PC processors, this isn't a big deal since they only have one architecture. The "x86 compatible" processor is a bit different and can switch into various 16/32/64 bit modes for application compatibility. Because most PC platforms support both 32-bit & 64-bit architecture along with old

school Intel 8086/80286 16-bit instructions, the UEFI firmware on these platforms can be compiled in IA32 or x64 mode.

On a legacy BIOS system, the firmware-OS hand-off happened in 16-bit real mode, and the OS was expected to transition to 32/64-bit protected mode. Aside from several technical limitations and security problems in the legacy BIOS boot model, this caused problems between the 16-bit BIOS runtime interfaces and 32/64-bit OS services. It's easier for the OS to get along with the firmware if they run at the same bitness (32-bit OS boots from 32-bit firmware, 64-bit OS boots from a 64-bit firmware) and this is reflected as a fundamental principle of the UEFI Specification.

Therefore, currently, Nitrux does not support 32-bit UEFI systems.

What is NX Desktop? (previously Nomad Desktop)

https://nx-desktop.org

Nitrux received many comments for creating yet another desktop environment instead of contributing to more extensive and established projects or worse that we have forked Plasma 5, unbeknownst to some people NX Desktop is NOT a desktop environment and we never have, once, referred to NX Desktop as such. This misunderstanding was caused by our use of the word Desktop in its name which is a tradition in many FOSS projects to apply to desktop shells.

What is NX Desktop and what it isn't?

- **What it is**:NX Desktop is our set of applied customizations to the Plasma 5 Desktop and this includes new plasmoids (or widgets), a new look and feel package (wallpapers, Plasma themes, Konsole theme and profile, Aurorae themes, SDDM themes, cursors, and color schemes) an icon theme and initially two desktop applications, NX Software Center and Nomad Firewall, all of which is original work. The NX Software Center became its own thing, and Nomad Firewall is a KCM (KDE Control Module) which is not part of the plasmoids package or the look and feel package. The icon theme Lüv is also not part of the look-and-feel package.
- **What it isn't**: NX Desktop is not a desktop environment. A desktop environment by concept would be a complete desktop shell and a suite of essential applications. These include a file manager, web browser, multimedia player, email client, address book, PDF reader, photo manager, and system preferences application. NX Desktop does not provide any of these which is why we have never referred to it as such.
- *NX Desktop is not a fork or a derivative or a spin-off of Plasma Shell.*

The *Desktop* suffix is because that's where everything that comprises it is used, in the desktop. Contrary to what many people might think, we did not create a distribution around "just a bunch of themes." **Nitrux** is about offering a user experience of our own that includes but it's not limited to the artwork and the overall aesthetics,

and our intention was never to provide a *vanilla* Plasma 5 Desktop. There are already sufficient Linux and BSD distributions that offer an unmodified Plasma 5 desktop experience.

How can I disable the welcome message when I log in?

To disable the NX Welcome Wizard from appearing every time that you log in, open System Settings, then click on Startup and Shutdown, click on Autostart and uncheck the box labeled *nx-welcome-wizard*.

Will Nitrux add another variant with other desktop environments?

No. We do not have it in our plans to add a variant of Nitrux that features a different desktop environment, e.g., Gnome. We are fully committed to using KDE technologies and the Qt framework for our graphical interface.

Why focusing on AppImages?

https://appimage.org/

> An AppImage is a downloadable file for Linux that contains an application and everything the application needs to run (e.g., libraries, icons, fonts, translations, etc.) that cannot be reasonably expected to be part of each target system. — AppImage wiki.

Nitrux is a desktop Linux distribution meant to be used in desktop computers (that includes laptops et al.). Based on this, AppImage is the format that most adequately befits our vision. It is truly portable, doesn't require anything else to run, and it is aimed at desktop computing. AppImage provides a very innovative and simplistic solution to software distribution. It is abundantly clear that a one-click solution to obtain software would be instantly accessible to anyone. That is not to say that package managers aren't smart (or powerful!) when resolving conflicts but it's not the same to manage a dozen if not hundreds of files to handle only one.

To be noted is that the AppImage format does not try to replace package managers but we want to make it our **primary method of distribution for end-user software.**

For this purpose, we have created a GUI software store and software to integrate AppImage files to the desktop environment. Also, we have included proper tools designed by the AppImage developers such as appimaged and AppImageUpdate. These tools are entirely independent of the Linux distribution where they

are running and are also independent of each other. The mobile segment proved that this unprecedented simplicity was vital for people to obtain new software let alone actually using it, why limiting that simplicity to our phones?.

We firmly believe in the idea behind AppImages where an application is not installed but instead deployed, and this idea is from where we have created znx, our operating system manager.

What is znx?

https://github.com/Nitrux/znx/wiki

Znx is a tool that lets you use multiple operating systems and keep them updated without having to repartition the drives. It's all about simplicity and reliability. Znx is an operating system manager. It manages the lifetime of OSes that are deployed with it. Znx is not an installer, a container, a program to flash USBs (it's not a replacement for 'dd' or anything similar), or virtualization software.

It's designed to work on UEFI systems. znx will create a GPT partition table on a device and write a bootloader into the EFI System Partition. From that moment, you can deploy, update and remove operating systems as single-file ISO images. The bootloader (GRUB2) will make them available on boot. znx will help you keep them updated while making the update process safe.

znx deploys the ISO image into the storage device, then makes the device bootable and creates the data partition where the user data is kept across reboots. Once that the ISO has been deployed the user can restart and boot into the storage device to select the system.

znx follows the concept of AppImages which is that software isn't "installed," but it is instead deployed. That means that you don't get lots of files and folders spread in a filesystem but just one file, same with znx, instead of unpacking the squashfs file (the compressed filesystem that holds the actual OS) and extracting the contents to the target device (thus "installing" the OS the traditional way). The OS remains as one file which can be updated using a zsync file for differential updates (so-called delta updates or transactional updates which is what znx does, similarly, AppImageUpdate performs updates to AppImages only updating the bits that changed).

Any file that the user creates is kept in the data partition in ZNX_DATA. To install new software (by the software we mean desktop applications like Firefox, LibreOffice, etc.) the go-to method is to download an AppImage. These AppImages would go in "/Applications" where they would be kept when the user updates the deployed ISO to a new version.

znx is a significant overhaul in Nitrux and together with the improved AppImage integration and our UI framework Mauikit forms the basis of what makes this distribution Nitrux and not Ubuntu.

znx supports both local ISO files and remote URLs.

What is the difference between installing and deploying in Nitrux?

In the context of utilizing Nitrux, we differ from conventional Linux distributions. Typically, when the user installs software, this is done with the use of a package manager. In general terms, the package manager will download and extract the contents of an archive and will place these contents inside the directory structure of the distribution in question. The archive contains files and folders that are placed in the corresponding path in the filesystem (the binaries in /bin, the configuration files to /etc, libraries to /lib and so on), and the package manager keeps track of where these files are located.

This concept is the same when an operating system is installed. Current Linux distributions are distributed as ISO files and make use of an installer. The installer will launch from a "Live" session and will extract the contents of the SquashFS file contained in the ISO file and place its contents on the storage device, creating the directory tree. Nitrux does not do this.

Deploying the operating system in our case means that you are "copying" the ISO file to the storage device as a single file and enabling persistence on the storage device using OverlayFS instead of extracting its contents and creating the standard directory tree on the storage device, this is why we refer to say that Nitrux does not do a traditional install.

In other words, Nitrux does a frugal installation (the operating system files are stored in just a couple of files in a directory, rather than spread out over a drive partition).

When you install a current Linux distribution, your storage device will contain a directory tree like this: /boot /bin /dev /lib /usr /opt /var and so on, each of these directories with files and more folders on as many partitions as you created during installation.

What znx does is frugal installations. *"A frugal installation only occupies one folder in a partition, and the rest of the partition can be used for anything else. Other Linux distributions, for example."* Meanwhile, traditional Linux distributions do a full installation, *"A full installation is where Linux occupies an entire partition, and in that partition you will see the folders /bin, /sbin, /opt, /etc/, /sys, /proc, /tmp, /dev, /usr, /run, /lib, and more."*

An installer such as Ubiquity, Calamares (KPM Core), Anaconda, and every other installer works the same, they extract the contents of the SquashFS file inside the ISO and place the contents on a partition of the storage device. That is why we don't refer to znx "installing" an OS; instead, we use the word **deploy**. *Because znx isn't extracting the SquashFS file from the ISO, it's booting the ISO directly, and data is preserved on the storage device using OverlayFS.*
With Nitrux you only have two partitions, a boot partition with the ESP and a data partition with the ISO file (containing the operating system) and the user folders, so the directory tree in your storage device is comprised

of only /STORE.

znx utilizes it, but what is the OverlayFS?

OverlayFS provides a great way to merge directories or filesystems such that one of the filesystems (called the “lower” one) never gets written to, but all changes are made to the “upper” one. Brought into the Linux kernel mainline with version 3.18, OverlayFS allows you to overlay the contents (both files and directories) of one directory onto another.

The source directories can be on different volumes and can even be different file systems, which creates an exciting mechanism for allowing temporary modification of read-only files and folders, this also allows you to quickly add some storage to an existing filesystem that is running out of space, without having to alter any structures. It could also be a useful component of a backup or snapshot system.

Modifications to files in the “upper” directory will take place as usual. Any modification to a file from the “lower” folder will create a copy in the upper directory, and that file will be the one modified, this leaves the base files untouched and available through direct access to the “lower” folder.

Interestingly, a second task could copy modified files from the “upper” folder to the “lower” when modifications are complete.

A file removed from the OverlayFS directory would directly transfer a file from the “upper” directory, and simulate that removal from the “lower” directory by creating what is called a “whiteout” file. This file exists only within the OverlayFS directory, without physically appearing in either the “upper” or “lower” directories. When the OverlayFS is dismounted, this state information will be lost, so care should be taken to reflect any necessary changes to the “lower” directory.

A subdirectory can also be deleted from the “lower” directory, which creates what is known as an “opaque directory” in the OverlayFS directory. Behind the scenes, OverlayFS uses the “trusted” extended attribute class or namespace to record “whiteouts” and “opaque directories.”

How does znx store data?

With znx it’s preferred that images are deployed to the computer’s primary storage device, instead of USB storage; but it’s not mandatory. We store data directly on the device, without intermediate files. At boot time, we create an overlay mount on /etc and /home. Our focus is to have a clean core system, and our preferred application distribution format is AppImage, so this made sense to us.

Znx further extends the concept of frugal installations (we call them “deployments,” not installations) by also serving the updates (which is a very critical state). It does not just manage the deployment, but also the whole lifetime of an image; this is, updating it safely (and keeping a backup of the last working version) and, possibly, discarding it. znx is designed to work only on UEFI.

The following diagram illustrates how znx stores data in a device.

```
####------------------------####
####    znx new structure   ####
####------------------------####

device
├── ZNX_BOOT (partition)
│     ├── /boot
│     │     └── grub
│     ├── /efi
│     │     ├── boot
├─────└─────└─────└── bootx64.efi
├── ZNX_DATA (partition)
│     ├── /store
│     │     ├── /os_1
│     │     │    ├── /version_1
│     │     │    │    ├── /data
│     │     │    │    │    ├── /user_1_data
│     │     │    │    │    └── /user_2_data
│     │     │    │    ├── image
│     │     │    │    └── image_bak
│     │     │    ├── /version_2
│     │     │    │    ├── /data
│     │     │    │    │    └── user_1_data
│     │     │    │    ├── image
│     │     │    │    └── image_bak
│     │     │    └──...
│     │     ├── /os_2
│     │     │    ├── /version_1
│     │     │    │    ├── /data
│     │     │    │    │    └── /user_1_data
│     │     │    │    ├── image
│     │     │    │    └── image_bak
│     │     │    └──...
└─────└─────└─────...


####------------------------####
####    znx old structure   ####
####------------------------####

device
├── ZNX_BOOT (partition)
│     ├── /boot
│     │     └── grub
│     ├── /efi
│     │     ├── boot
├─────└─────└─────└── bootx64.efi
├── ZNX_DATA (partition)
│     ├── /data
│     │     ├── /boot_images
│     │     │     ├── /os_1
```

```
52.  │    │    │    │   ├── /version_1
53.  │    │    │    │   │   ├── image
54.  │    │    │    │   │   └── image_bak
55.  │    │    │    │   ├── /version_2
56.  │    │    │    │   │   ├── image
57.  │    │    │    │   │   └── image_bak
58.  │    │    │    │   └──...
59.  │    │    │    ├── /os_2
60.  │    │    │    │   ├── /version_1
61.  │    │    │    │   │   ├── image
62.  │    │    │    │   │   └── image_bak
63.  │    │    │    └──...
64.  │    │    ├── /data
65.  └────└────└────└── /user_data
```

How does znx do updates to the operating system?

Znx upgrades the operating system by doing a transactional update, also known as atomic upgrades, delta updates, binary updates, or partial updates.

At its heart, znx uses Zsync (hence the z). Zsync will calculate the difference between the old and new file in a remote server and download only the different parts. By utilizing Zsync, znx can save the bandwidth that would be otherwise used to download a new ISO file. To make use of this feature the vendor has to embed the update information in the ISO, then znx can adequately make use of the Zsync file that would be generated during the creation process to update the ISO image.

A “transactional update” is a kind of update that can be defined as:

- **A transactional update** is atomic — the update does not influence your running system
- **A transactional update** can be rolled back — if the upgrade fails or if the newer software version is not compatible with your infrastructure, you can quickly restore the situation as it was before the update.

Unlike other Linux distribution methods, the information where to look for updates is not contained in separate repository description files such as sources.list that need to be managed by the user but is embedded inside the ISO file using a file called .INFO.

This method has the advantage that the update information always travels alongside the ISO file so that the end user does not have to do anything special to be able to check for updates. During the upgrade process, znx will keep a backup of the ISO file, called IMAGE.0.zs-old, which is a backup of the former file that only exists after an update is performed.

So znx uses all the device, does that mean that I can't use other operating systems but Nitrux?

You can entirely use other operating systems with znx, in fact, **you can deploy other Linux distributions using it**. Znx is not exclusively a tool created as a whim from us to not use an installer. We developed znx to serve as a more straightforward approach to manage operating systems. As we have explained, znx is not an installer as it performs other tasks too.

We developed znx to use it with Nitrux, but it's not exclusively made to work only with Nitrux. To deploy other Linux distributions using znx, the initramfs has to be modified accordingly to support the use of the OverlayFS mount points and also the ISO needs to have a grub.cfg file and support for EFI, most distributions have this file, while others don't, typically these distributions use ISOLINUX.

How do I create new partitions with znx?

One of the points of using znx is to deploy multiple distributions in one run and easily manage them. As we have explained in this FAQ, znx creates two partitions boot and data (ZNX_BOOT, and ZNX_DATA). With znx, all the operating systems deployed with it are residing as a single archive, and their data is stored using OverlayFS directly to the storage device without any intermediate files in the machine.

Therefore, by utilizing znx there's no need to create multiple partitions.

Does a Linux distribution, other than Nitrux, have persistence too when deployed using znx?

If you were to deploy an ISO file of a random Linux distribution *today* with znx no, it wouldn't have persistence; this is because *the initramfs file requires to be modified to make use of OverlayFS for the data to persist across reboots*. Officially, there are no other Linux distributions that we know of that support znx at the moment; however, we hope that soon changes. Unofficially, we provide ISO files that are compatible with znx and which use OverlayFS, these files can be found in our Developments build section.

As we mention in this FAQ, we make use of OverlayFS to store data. We do this by modifying our initramfs accordingly during its build process. This process isn't complicated, and it's supported directly by using initramfs-tools. However, we're aware that not all Linux distributions make use of initiramfs-tools, in that case, a custom solution may need to be created for example if the Linxus distribution uses dracut instead.

All Linux distributions release their ISO files as installation media, so currently they are not built with this use in mind. However, it's entirely possible to make these changes, just like we did.

How can I add support for znx to my Linux distribution?

To add support for znx to your Linux distribution, please check the following information:

- First, to make your ISO file compatible with znx, It must be able to boot with a **loopback.cfg** file, the original link to the documentation is broken: http://www.supergrubdisk.org/wiki/Loopback.cfg; but here are some links that work:
  - How to boot Linux from ISO file using loopback.cfg?.
  - Make a generic grub2 boot menu (using loopback.cfg).
  - In our case the file loopback.cfg points to grub.cfg

```
source /boot/grub/grub.cfg
```

- Second, to enable data persistence across reboots, it should implement a mechanism in the initramfs. Here's an example to achieve this:

```
#! /bin/sh

for o in $(cat /proc/cmdline); do

  case $o in

    # -- Determine the overlay mountpoints from /proc/cmdline.

    ZNX_OVERLAYS=*)

      ZNX_OVERLAYS=${o#ZNX_OVERLAYS=}
      ZNX_OVERLAYS=$(printf $ZNX_OVERLAYS | sed 's/,/ /g')

    ;;

    # -- Find the persistent storage directory.

    iso-scan/filename=*)

      PERSISTENT_DATA=${o#iso-scan/filename=}
      PERSISTENT_DATA=${PERSISTENT_DATA%/*}/DATA

    ;;

  esac

done


# -- Mount the requested overlays.

for k in $ZNX_OVERLAYS; do

  mkdir -p \
    /root/$k \
    /root/isodevice/.overlay-tmp/$k.w \
    /root/isodevice/$PERSISTENT_DATA/$k \
    /root/isodevice/.overlay-tmp/$k.w

  mount -t overlay \
    -o lowerdir=/root/$k \
    -o upperdir=/root/isodevice/$PERSISTENT_DATA/$k \
    -o workdir=/root/isodevice/.overlay-tmp/$k.w \
    . /root/$k

```

```
done
```

- Finally, you will need to provide a usable grub.cfg file. Note that the line ZNX_OVERLAYS=/ is added as a GRUB parameter. We have modified znx to make it easier to define what directories in the filesystem will use the OverlayFS driver. In this example, setting the GRUB parameter to ZNX_OVERLAYS=/ will use the OverlayFS driver for the entire root filesystem.

```
set gfxmode=auto
insmod efi_gop
insmod efi_uga
insmod gfxterm
terminal_output gfxterm
loadfont /boot/grub/themes/ubuntu/dejavu_sans_mono_bold.pf2

set theme=/boot/grub/themes/your/theme.txt

menuentry "Boot MyDistro " {
  set gfxpayload=keep
  linux /boot/kernel boot=casper quiet splash elevator=noop iso-scan/filename=$iso_path username=ubuntu hostname=ubuntu ZNX_OVERLAYS=/
  initrd /boot/initramfs
}

menuentry "Boot MyDistro debug mode" {
  set gfxpayload=keep
  linux /boot/kernel boot=casper xforcevesa elevator=noop iso-scan/filename=$iso_path username=ubuntu hostname=ubuntu ZNX_OVERLAYS=/
  initrd /boot/initramfs
}
```

Please note that due to the differences between the different families of Linux distributions in the creation of the initramfs, the method above is not guaranteed to work without modifications.

Deploying Nitrux from an existing Linux (Live or already installed) using znx

1. Download the znx AppImage from https://github.com/Nitrux/znx/releases
2. Make it executable, open a Terminal window and run the following commands

```
chmod +x znx-*
```

- And for clarity, rename the file

```
mv znx-* znx
```

- Alternatively, use a file manager (right-click>properties>permissions.)

Since znx is a command line utility, all the steps below will be done on a Terminal window.

- First, initialize a storage device running:

```
sudo /path_to/./znx init /dev/sdX #where X is the device, do not enter a partition
```

If you're using an NVME device use **/dev/nvmeXnY** where X is the port and Y is the device number, as mentioned before, do not enter a partition.

- Now deploy Nitrux, run:

```
sudo /path_to/./znx deploy /dev/sdX nitrux/$version $isopath #where $version can be anything (no spaces or hyphens) and $isopath can be a local file or a remote URL
```

Depending on the speed of the storage device where the ISO is being deployed to it will take between 5-15 minutes or a couple of seconds in a virtual machine. During this time the command prompt will not be available, so it's critical that the window where the command is running is not closed.

- After the command prompt is available once again make sure that the write cache is empty, run the command:

```
sync
```

And that's it you can begin to use Nitrux. If the ISO were deployed to a USB 2.0 device, it would take several minutes to load due to the limits of the transfer speed of these devices.

Subsequent deployments of Nitrux can be done using znx GUI from the Nitrux deployment. You can skip downloading the ISO altogether since the ISO would be already available from the device that Nitrux has booted from. The file is at "/isodevice/STORE/$image".

Deploying Nitrux from Windows using znx

Unfortunately for Windows users, the AppImage of znx does not run in WSL 1 or Cygwin because ***neither can access the block devices in a computer and the kernel used by WSL 1 does not have the FUSE module included***. This situation represents a problem that znx can't fix since it isn't a problem of znx but a problem with Windows not being a POSIX system. *WSL 2 may have support for FUSE; however, it hasn't been officially released yet.*

For Windows users, please create a bootable Live USB using balenaEtcher or Rufus and proceed to deploy Nitrux afterward. *See How can I flash the ISO to a USB?.*

How can I verify that znx successfully deployed Nitrux?

To confirm that the deployment was successful, you can do the following:

- Verify that the partitions were created using the commands lsblk, blkid, or fdisk.

```
lsblk /dev/sdX

lsblk -o NAME,SIZE,MOUNTPOINT,LABEL

sudo blkid

sudo fdisk -l
```

It's not necessary to use all of these commands at once.

How can I burn the ISO to a DVD?

Please don't write our ISO file to any optical media.

How can I flash the ISO to a USB?

These instructions will walk you through the process of creating a bootable Nitrux USB stick on Windows or Linux. You can use a USB stick to boot and test out or install Nitrux on any computer that supports booting from USB and uses EFI as a minimum.

If you are using Windows or Linux and **balenaEtcher**, follow these steps:

1. Download a Nitrux ISO file, download and install balenaEtcher.
2. Insert the USB flash drive into the USB port and launch balenaEtcher.
3. Click on the Select image button and locate your Nitrux .iso file. If you downloaded the file using a web browser, then it should be stored in the downloads folder located in your user account.
4. Click on the Flash image button, and the flashing process will start.

The process may take several minutes, depending on the size of the ISO file and the USB stick speed. Once completed, the following screen will appear informing you that the image is successfully flashed.

If you are using Windows and **Rufus**, follow these steps:

1. Download a Nitrux ISO file, download and install Rufus.
2. Insert the USB flash drive into the USB port and launch Rufus.
3. Rufus will update to set the device within the **Device** field. If the **Device** selected is incorrect (perhaps you have multiple USB storage devices), select the correct one from the device field's drop-down menu
4. Now choose the Boot selection. Choices will include *Non-bootable, FreeDOS, or Select ISO image file* since you are creating a bootable Nitrux device, select **Non-bootable**. The default selections for the Partition

scheme (*GPT*) and Target system (*BIOS or UEFI*) are appropriate (and are the only options available).

5. To select the Nitrux ISO file you downloaded previously, click **SELECT** to the right of "Boot selection". If this is the only ISO file present in the Downloads folder you will only see one file listed. Select the appropriate ISO file and click on **Open**.
6. The *Volume label* will be updated to reflect the ISO selected. Leave all other parameters with their default values and click **START** to initiate the write process.
7. The ISO will now be written to your USB stick, and the progress bar in Rufus will give you some indication of where you are in the process. With a reasonably modern machine, this should take around 10 minutes. Total elapsed time is shown in the lower right corner of the Rufus window.
8. When Rufus has finished writing the USB device, the Status bar will be green filled and the word **READY** will appear in the center. Select **CLOSE** to complete the write process.

If you are using Windows or Linux and **ROSA Image Writer**, follow these steps:

1. Download a Nitrux ISO file, download, and install ROSA Image Writer.
2. Insert the USB flash drive into the USB port and launch ROSA Image Writer.
3. Click the folder icon to browse for the Nitrux ISO file and select it.
4. Select the USB device from the dropdown menu.
5. Click **Write** to initiate the writing process.

If you are using Linux and **KDE ISO Image Writer**, follow these steps:

1. Download a Nitrux ISO file, download, and install KDE ISO Image Writer.
2. Insert the USB flash drive into the USB port and launch KDE ISO Image Writer.
3. Click the folder icon to browse for the Nitrux ISO file and select it.
4. Select the USB device from the dropdown menu.
5. Click **Create** to initiate the writing process.

If you are using Linux and **the command 'dd'** follow these steps:

1. Download a Nitrux ISO file.
2. Insert the USB flash drive into the USB port and open a Terminal window.
3. Enter the following command in the Terminal window and hit Enter.

```
sudo dd if=nitrux.iso of=/dev/sdX oflag=sync bs=4M status=progress
```

How do I try Nitrux on a virtual machine?

The single best way to test Nitrux is to deploy it in real hardware, but maybe you don't want to do that yet so your second best option is to use a virtual machine.

- **Oracle VirtualBox**. To use the ISO in VirtualBox all that you have to do is create a new virtual machine. Select *Linux and Ubuntu (64-bit)* after you have customized the rest of the settings for your virtual machine to boot the ISO check *Enable EFI (special OSes only)* under System and ***VBoxVGA*** under Graphics controller.
- **VMware Workstation Pro 15 and Workstation Player 15**. To use the ISO in VMware Workstation all that you have to do is create a new virtual machine. Select *Linux and Other Linux 4.x or later kernel 64-bit.* And just like with VirtualBox enable EFI, in this case, it would be UEFI. *Select the UEFI checkbox under Firmware type in the Options tab in the virtual machine settings.* For Workstation Player 15 add the following line to the vmx file of your virtual machine, **firmware = "efi".**
- **Parallels Desktop 13+**. Following this article in the Parallels knowledge base, to enable EFI the boot flag must be set to *vm.bios.efi=1* under Advanced Settings in the Boot Order category.

I deployed Nitrux to a USB drive using znx. How do I redeploy it to my internal storage?

To redeploy Nitrux from a USB drive, all that you need to do is the following:

```
sudo znx deploy /dev/sdX $vendor/$version /isodevice/STORE/$first_deployment/IMAGE.0
```

- Where "/dev/sdX" is the device that you want to redeploy (and where X can be a, b, c…). If the device is an NVME drive that would change to "/dev/nvmeXnY" (where X and Y can be 0, 1, 2….).
- Where "$first_deployment" means the name that you gave to the deployment in the USB, e.g., if when you deployed Nitrux to the USB, you called it "nx/dev."
- The path where znx deploys the ISO files is always "/isodevice/STORE" followed by the name, e.g. if the name was "nx/dev" the path would be "/isodevice/STORE/nx/dev/IMAGE.0."

I deployed Nitrux using znx, but I got one of the following errors

- **"No Operating System" or "unknown filesystem."**
- **"Selected boot image did not Authenticate. Press Enter to Continue."**
- **"Uncompression Error. System Halted" or "no bootable medium found."**
- **"Reboot and select a proper Boot device or Insert Boot Media in selected Boot device and press a key."**

Please make sure that your motherboard supports UEFI or EFI. If you are running a VM make sure to enable EFI support. Also, make sure that Secure Boot and any other Windows-specific startup options in your UEFI settings are disabled.

- **"znx: Error: Bad image name (must match '^[a-zA-Z0-9_]+/[a-zA-Z0-9_]+$')"**

  Please make sure that the image name does not use spaces or hyphens. e.g., "nitrux/1-1-7" or "nitrux/1 1 7" is invalid while "nitrux/1_1_7" or "nitrux/117" is valid.

- **"znx: Error: /dev/sdX is mounted! Unmount it before continuing."**

  Please make sure that the device where you are trying to deploy Nitrux is not mounted. Devices that are mounted cannot be modified while in use.

- **"wipefs: /lib/x86_64-linux-gnu/libsmartcols.so.1: version `SMARTCOLS_2.33` not found required by wipefs."**

  Please make sure that the operating system where you are trying to run the AppImage comes with a recent version of util-linux, specifically, version 2.32 and newer are preferred. If your operating system does include a more recent version of util-linux and you're still receiving this error, please contact us to assist you.

I have XYZ operating system installed on my computer, how can I dual-boot Nitrux?

While it is possible to dual-boot NItrux with other operating systems, *there are some things to consider.*

1. Nitrux must be deployed using znx before any other operating system is installed on the computer.
2. znx will create two partitions, the ESP and the Data partition.
3. The included installers in most Linux and BSD distributions can do manual partitioning for their operating systems, during this phase the user can select the ESP that znx created and use it as mount point "/boot/efi." The installer will add the EFI files for the operating system in question.
4. During the manual partitioning phase of these installers, the user can shrink the Data partition that znx created to install the new operating system.

It is possible that the files in the ESP partition of znx are overwritten by the installer of the other operating system. ***We have confirmed that this is the case with the Ubiquity installer found in Ubuntu and its derivatives****. By using znx, it is possible to deploy more Linux distributions the same way as Nitrux. However, like Nitrux other Linux distributions have to make use of OverlayFS in their initramfs to enable persistence.*

For Windows users.

1. If Windows is already installed in the computer and Nitrux is deployed to the same storage device, the storage device will be wiped.
2. The Windows installer should be able to select the EFI system partition (ESP) automatically. However, it won't be able to shrink the Data partition since Windows does not support BTRFS by default (this third-party driver adds some support for it), so to accommodate Windows in the Data partition that znx created the partition has to be resized before installing Windows.
3. If Nitrux is deployed first, then the Data partition is resized, then Windows is installed on the remaining space, it's very likely that access to Nitrux is lost. Windows is not Linux, Windows does not use

GRUB so it can't be deployed using znx and its ISO file is for installation and recovery only.

In summary, to avoid any of these complications we recommend users to use *separate storage devices.*

I have a previous version of Nitrux installed, how do I upgrade to the most recent release?

Starting from Nitrux version 1.1.0 we introduced znx, our tool to manage operating systems. As we have explained in this FAQ, our stated goals do not include the use of a package manager and a traditional installer to manipulate the operating system, as such, to upgrade Nitrux you have to use znx.

To upgrade Nitrux using znx follow the steps in the Compendium.

If you have installed a version of Nitrux such as the release candidate, Alpha 1 and Alpha 2, and any release from the 1.0.x series (1.0.0, 1.0.1, 1.0.2, 1.0.3, 1.0.4, 1.0.5, 1.0.6, 1.0.7, 1.0.8, 1.0.9, 1.0.10, 1.0.11, 1.0.12, 1.0.13, 1.0.14, 1.0.15 and 1.0.16) ***you cannot upgrade to the newest release of Nitrux due to the fundamental changes the operating system.***

I can't find XYZ application in the software center, how can I add new software?

Some of the significant software is already available as an AppImage, GIMP, Inkscape, LibreOffice, ONLYOFFICE, Blender, Krita, among others. The software center loads its listings from https://www.linux-apps.com. Alternatively, you can also download new AppImages from the web from sites like https://appimage.github.io or https://www.appimagehub.com/. Move the downloaded AppImage file to "/home/$USER/Applications", and the operating system will add it to the application menu.

Our Software Center was never designed to support anything else other than portable formats at first Snaps and then AppImages *only*. As mentioned before, the idea is not to use a package manager. However, *we acknowledge that the software center should display a lot more results*.

Recently we have removed the Software Center from our release images as it is in our plans to update it entirely. However, we have added the console-based AppImage CLI Tool. This tool is available as the command **app** and can be used in the following way.

- Open Konsole and type the command "app" without quotes followed by the argument then press Enter.

```
app --help
Usage: app [options] command
Command details
   search <query>              list applications available in the store
   install <STORE ID>          install the application with the given store id
```

```
    list                        list applications available on your system
    update <APP ID>             update if possible the given application
    remove <APP ID>             remove the application from your system

Options:
  -h, --help  Displays this help.

Arguments:
  command     Command to be executed: search | install | list | update | remove

app search firefox
1168996 Firefox - Appimage 63.0.1 by AJSlye

app install 1168996
Getting 1168996 from
https://dl.opendesktop.org/api/files/download/id/1530021194/s/0cb9c26e57f7011261ca000b778cf021/t/1553641278/o/1/Firefox-60.0.glibc2.7-x86_64.AppImage
```

- *The AppImages that are downloaded using this command are retrieved from AppImagehub.com.*
- *Providing AppImages is entirely up to the developer, so, if the software that you are looking for is not available as an AppImage ask its developer to provide one and it will be available.*

How can I add new software that isn't available as an AppImage to Nitrux?

The use of an AppImage by a developer to distribute their software allows them to target Linux as a whole instead of targeting a specific distribution. We emphasize the use of AppImages in Nitrux as it is a more straightforward way of managing end-user applications. However, not all developers make use of this packaging format, most software that is targeting Linux available as packages created for package managers like APT, rpm, etc.

There are other instances where the developer releases the software as a TAR or ZIP compressed file; it is often the case that this software can directly be extracted to and doesn't need to be installed using the distribution's package manager. An example of this software is Android Studio and the itch.io game and assets storefront.

Other software is released as .run script files which depending on the software, allow you to select the installation path.

Recently we have added Homebrew to Nitrux. Homebrew is a macOS package manager that works on Linux, which allows users to install software to their home directory.

Homebrew features are the following:

- Homebrew can install software to your home directory and so does not require *sudo*.
- Install software not packaged by your host distribution.
- Install up-to-date versions of software when your host distribution is old.
- Use the same package manager to manage your macOS, Linux, and Windows systems

Homebrew requires configuration from the user, follow these instructions to configure it.

A package in Homebrew is called a formula.

Third-party repositories in Homebrew are called a tap.

- To begin, open Konsole and type the command **brew.**

```
brew
```

- When first run, this command will update brew and will add the necessary folders to the /home directory. Follow the instructions on the screen.
- Once you have successfully set up **Homebrew** on your machine, you can start using it.

To list all installed formulae, run.

```
brew list
```

To install a formula using Linuxbrew, use the command **brew**. For instance, hello, which is a Hello World! example package.

```
brew install hello
```

You can search for packages using the following syntax.

- The first command will show all the available formula, while the second will show formulae that match a keyword.

```
brew search
brew search --desc <keyword>
```

You can also add third-party repositories to **Homebrew**. To list all current repositories run.

```
brew tap
```

To know more about **Homebrew** usage options, type:

```
brew help
```

We strongly recommend reading the **Homebrew** documentation at https://docs.brew.sh.

I updated Nitrux using the package manager but nothing changed?

As we have mentioned already, to manage the operating system, we use znx; this is normal and expected behavior.

How can I update software in Nitrux?

To update software in Nitrux we make use of AppImageUpdate. Just like znx updates ISOs using transactional updates or delta updates, AppImageUpdate updates AppImage files using a zsync file, which means that only the bits that changed are updated.

The AppImage developers describe it as "*AppImageUpdate lets you update AppImages in a decentral way using information embedded in the AppImage itself. No central repository is involved. This enables upstream application projects to release AppImages that can be updated easily. Since AppImageKit uses delta updates, the downloads are very small and efficient.*"

AppImages do not update themselves, or automatically. AppImages are updatable on-demand by the user, to do this follow these steps:

- Navigate to "/home/$USER/Applications" using the file manager.
- Then right-click an AppImage and select *Actions>Update AppImage*.

Note: For AppImageUpdate to do its magic, the update information must be embedded inside the AppImage. As indicated in "Update information overview" at the AppImageUpdate repository.

How can I remove software in Nitrux?

To remove software in Nitrux, all that you need to do is delete the AppImage file.

I installed XYZ packages using the package manager, but after rebooting, they disappeared?

Nitrux is an immutable operating system; this is normal and expected behavior. To obtain new software, please use AppImages.

How can I update operating systems deployed with znx other than Nitrux?

To update other operating systems that were deployed using znx follow the instructions to update Nitrux.

I created a new user in System Settings and enabled autologin, but it's not working?

This issue occurs due to the way that Casper works out of the box. As a workaround, press "E" after selecting the operating system in the znx boot menu, and use the arrow keys to navigate to the line marked user and hostname; here you can enter the username and the name of the computer after making the desired changes press "F10" to boot the operating system.

We recommend that after creating a new user using System Settings logout from the current session, in the login screen, use the new user credentials and reboot from the graphical session.

How long is Nitrux supported for?

Nitrux is a continuously evolving operating system, as such, we only support the latest version of the system, *that is currently the 1.2.x series of releases.*

Earlier versions of Nitrux such as the release candidate, Alpha 1 and 2, the 1.0.x series of releases, and the 1.1.x series of releases are not supported anymore.

Do I have to deploy Nitrux again after a new release is available?

No. Once you have deployed Nitrux to your storage device all you need to do is execute znx to update it. *You only need to deploy the operating system once.*

What is Mauikit?

https://mauikit.org

A set of templated controls and tools based off QQC2 and Kirigami shared among the Maui set of applications. MauiKit helps to quickly build UIs that follow the Maui HIG and bring ready to go tools for different platforms, such as Android and Linux. Seamless transition between mobile and desktop technology — where the line between desktop and mobile is blurred. Using the same codebase, Maui Apps provide users with one app for multiple form factors.

- Ship your app faster with less code. Accomplish in a few lines what would otherwise take hundreds, from concept to the end user's screen — the fastest way to create convergent apps.
- Build apps for any device. Use the same code whether you're developing for Android or desktop. Hot push new features without forcing users to download a new app.

- Integrate technologies you use. Use popular frameworks and tools, right out-of-the-box. Focus on building features instead of disparate wiring components together yourself.

Some of the Maui applications are:

- Index, a file manager.
- Pix, a gallery manager.
- VVave, a music library manager.
- Nota, a simple text editor.
- Station, a terminal emulator.

It is in our plans to exclusively make use of Maui applications and software in the operating system. Currently, we only include Pix, the gallery manager. However, some of the Maui applications are available to test and troubleshoot as AppImages here.

Is MauiKit a fork of Kirigami?

No. MauiKit can be seen as an add-on to Kirigami. Mauikit is not forking Kirigami; it builds upon it. In reality, some MauiKit components have been upstreamed to Kirigami. MauiKit differentiates from Kirigami in some of its interaction models and its styling.

What differentiates MauiKit applications from others?

MauiKit applications aim towards convergent use-cases, that is, the use of a single application across different form factors. MauiKit applications are designed in a way that the desktop modality offers traditional users a familiar workflow when utilizing their computer, while the same app can be recompiled for mobile experiences without code differences. The use of a single code-base and a single interface with adaptive user experience are the major differentiators.

What is VMetal, and why are the system requirements so high?

VMetal allows users to run Windows in parallel to Nitrux to provide users of access to Windows software; meanwhile, they can still use their Linux desktop at the same time. VMetal is not a wrapper or a compatibility layer; it makes use of QEMU and KVM (Kernel-based Virtual Machine) on the software side and of VFIO and IOMMU on the hardware side, meaning that Windows is accessing directly the hardware that it utilizes.

For this feature in Nitrux to work the hardware where VMetal will be running must support specific features; otherwise, the software will not work, and you won't be able to enjoy the benefits.

- **CPU must support virtualization extensions** (Intel VT-x, AMD-Vx)
- **CPU must support Directed I/O** (VT-d for Intel or AMD-Vi, generically known as **IOMMU**)
- **The motherboard must support VT-x and VT-d (or AMD equivalents).**
- **The motherboard and the graphics cards must support UEFI.** When buying new hardware, check the motherboard and the graphics card user manual.
- **Should have one or more discrete PCIe graphics cards**. A combination of an iGPU and a discrete GPU is also usable.

The listed system requirements to use VMetal in Nitrux are high because they are considering two things

- The computer will be effectively running two operating systems at the same time. A piece of sufficiently robust computer equipment is needed.
- Therefore, you need a CPU that has enough cores to provide an optimal configuration (4+2 or 6+2 or 4+4) for VMetal with optimal IPC performance, enough RAM (8GB or more) to run both operating systems and software within them, and two graphics cards with individual outputs (compatible with UEFI). Additionally, all the core hardware components must support features tailored for virtualization.

How does VMetal work?

VMetal is as its name suggests, a virtualization feature built into Nitrux. VMetal works by utilizing the following technologies.

QEMU is a generic and open source machine emulator and virtualizer. When used as a machine emulator, QEMU can run OSes and programs made for one machine (e.g., an ARM board) on a different computer (e.g., your PC). By using a dynamic translation, it achieves outstanding performance.

KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko.

The **VFIO** driver is an IOMMU/device agnostic framework for exposing direct device access to userspace, in a secure, IOMMU protected environment. In other words, this allows safe, non-privileged, userspace drivers.

An **input-output memory management unit** (**IOMMU**) is a memory management unit (MMU) that connects a direct-memory-access–capable (DMA-capable) I/O bus to the main memory.

Utilizing VMetal is also possible thanks to AMD-Vi and Intel VT-d.

- *AMD-Vi, Intel VT-d. AMD-Vi/VT-d is a processor feature that enables virtualization of I/O resources (directed I/O).*
- *The IOMMU extends the concept of protection domains (or domains, for short) first introduced with the AMD64 DEV. The IOMMU allows each I/O device in the system to be assigned to a specific area and a distinct set of I/O page tables.*
- *The I/O Memory Management Unit (IOMMU) extends the AMD64 system architecture by adding support for address translation and system memory access protection on DMA transfers from peripheral devices. IOMMU also helps filter and remap interrupts from peripheral devices.*

In addition to this, VMetal is aimed to support UEFI operating systems and computer hardware such as Nitrux and Windows 10 utilizing **OVMF**.

OVMF is an EDK II based project to enable UEFI support for Virtual Machines. OVMF contains sample UEFI firmware for QEMU and KVM.

VMetal does not make use of libvirt, Virt Manager and neither does it make use of the ACS override patch for the Linux kernel, as using it would compromise the security of the system. VMetal also does not do binding and rebinding of the PCI Express device that would result in the graphical session of the host presenting problems.

What kind of performance can I expect from VMetal?

VMetal, unlike other popular virtualization options for end-users, provides almost native performance when it comes to games which are the primary focus of this feature.

This reasoning is considering that Windows would be accessing the hardware such as the graphics card directly and it wouldn't be using an emulated piece of hardware or software rendering using the CPU as do most popular virtualization software.

Performance, however, will be directly tied to the hardware in the computer, e.g., A high-end CPU, with high IPC, higher clock speeds, more cache, etc. will perform better than a low-end CPU with lower IPC, lower clock speeds and less cache, and the same applies for the rest of the components, i.e., RAM and graphics cards.

As we have stated before, *We build Nitrux for the present and the future, not for the past*. Trends in hardware manufacturing indicate that future components will be more efficient, robust, and in the case of CPUs will have more cores in a single package as it has been the case over the past decade, these trends also indicate that consumer motherboards will not make use of Legacy BIOS in the future but exclusively UEFI.

How can I check if my hardware is compatible with VMetal?

You can check the compatibility of your system in the following ways:

1. Check your motherboard manual for any indication of IOMMU or VT-d enablement.
2. Check the following links in Wikipedia List of IOMMU-supporting hardware, List of Intel chipsets and List of AMD chipsets and use your browser search function (Ctrl+F) to look for the terms VT-d or IOMMU.
3. Due to the extensive portfolio of processors by Intel and AMD, we recommend that you search for your specific model including keywords such as “vt-d,” “iommu,” “amd-v” or “amd -vi.”
4. If you’re on Linux run the following script in a terminal window. Copy the code, paste it on a blank document, and give executable permissions to the file.

```
#!/bin/bash

# Find IOMMU Groups
for d in /sys/kernel/iommu_groups/*/devices/*; do
  n=${d#*/iommu_groups/*}; n=${n%%/*}
  printf 'IOMMU Group %s ' "$n"
  lspci -nns "${d##*/}"
done

# Did this PC boot in UEFI or BIOS?
[ -d /sys/firmware/efi ] && echo UEFI || echo BIOS

# Find if CPU is 64-bit
grep -w -o lm /proc/cpuinfo | uniq

# Find if CPU uses SVM or VMX extensions
egrep -o '(vmx|svm)' /proc/cpuinfo | sort | uniq
```

Other things to consider are the following:

- The graphics cards don’t necessarily have to be the same model or brand. But it’s imperative that they are in separate IOMMU groups, to verify this check the output from the script.
- In our testing, hyperthreading works for AMD Ryzen processors without any problems. However, your mileage may vary depending on your motherboard.
- *Some motherboards’ UEFI firmware may cause problems with PCI power management*. Unfortunately, there’s little we can do about this.
- Some configuration may be required for Windows post-installation, for assistance, please contact us.
- If your needs for VMetal are outside of gaming and include other activities like audio recording, we strongly recommend that you use an external audio device like a USB adapter or a PCI-Express sound card to avoid latency problems.
- Overclocking your CPU and RAM will benefit the performance of VMetal; however, only do so if your computer has sufficient cooling. Once booted into Windows you can overclock your graphics card too.

If you intend to use VMetal on a laptop with hybrid graphics, consider the following:

- The use of VMetal requires that *two completely independent GPUs are found in the system*. It is the case that with most hybrid graphics found in laptops that there are indeed two graphics chips, however, they are not

independent of each other. Typically, the low-powered GPU will drive the main display and the external display outputs, when the user switches to the discrete GPU, the low-powered GPU takes the role of a display adapter with the discrete GPU taking the part of the 3D renderer.

- You may check the output of the script above to verify this. For example, a GPU will output the following string: VGA compatible controller while a 3D renderer outputs this string: 3D controller. The main difference between these two values is how they're reporting to the operating system. A *3D controller does not drive displays by itself, and it uses the VGA controller as a passthrough to output the video to the connected screens*.
- In addition to this, you must be able to select one of the two GPUs in your system as a *primary display output* or as an *initial display output* in the UEFI of the laptop. Given the way that hybrid graphics work, it's very likely that this is not an option.
- Switching the GPU that is used after the operating system has booted up makes no difference since the operating system has to take over the device during boot time.

Is VMetal already available?

Yes. VMetal is available in Nitrux starting with version 1.2.0.

How can I use VMetal?

To start using VMetal in Nitrux, first, make sure that your hardware meets the requirements to run it. We have detailed these requirements in our blog here and also in this FAQ.
If your computer is capable of supporting the features that are necessary for VMetal to run the next thing that you want to do is start it. You can do this by going to the application launcher, then click on "Utilities" and "VMetal." Keep in mind that VMetal will run in the background; *it's not a graphical application*.

Alternatively, you can run the command "vmetal" (without sudo) from the Terminal for the startup process and with sudo to boot the virtual environment.

```
1. vmetal
```

- The setup process is automated except for downloading the Windows 10 ISO file and installing Windows.
- Make sure that your computer has two individual video outputs, e.g., Integrated graphics and a discrete graphics card or two separate graphics cards.
- If your computer has integrated graphics, make sure to select them as the primary display output in your UEFI firmware.

Next, we will detail the setup process for VMetal. To start using VMetal, you will need the following:

- You will need a Windows 10 ISO file to install Windows and the VirtIO ISO file. Both files have to be present in the following path “/home/$USER/VMetal/iso/.”
  - If VMetal detects that there’s no Windows ISO present, it will redirect you to download Windows 10 from Microsoft’s official site.
  - If you already have a Windows 10 ISO file that you want to you use to put it in the following path “/home/$USER/VMetal/iso/” with the filename “win_install.iso.”
- After downloading the Windows ISO file, VMetal will download the VirtIO ISO with the drivers in the background. These drivers will be necessary for the virtual environment so that you can use the virtual storage devices and the virtual network. VMetal will save this file as “virtio_drivers.iso” in “/home/$USER/VMetal/iso/.”
- **If neither of these files exists VMetal will exit and not continue.**

Note: If you already have installed Windows, you can do the following to skip this check.

Open Konsole and type these commands:

```
cd /home/$USER/VMetal/iso
touch win_install.iso
touch virtio_drivers.iso
```

Once these files are present in the right path, start VMetal again to continue the setup.

1. After these two files are present on your computer, VMetal will automatically create and configure the main virtual drive (50GB).
2. And, If there’s enough space VMetal will also create a secondary virtual drive (100GB).

- Please note that this process can take a long time, depending on the speed of your storage device. For example, *it can take up to 1 hour in a SATA 6 SSD to create the first virtual drive.*

To avoid permission errors do the following:

```
sudo -i
ln -sv /home/$USER/VMetal/ VMetal #where $USER is your username
exit
```

After VMetal has verified the existence of the two ISO files and the existence of its virtual drives, it will boot the virtualized environment. The virtual disks will be present at “/home/$USER/VMetal/images.”

- Please note that to interact with the virtual environment that VMetal creates, you need to use your USB input devices. VMetal will look for these devices in the ports 1-7 of the USB Bus 1 of your motherboard.
- *We strongly recommend that if you don’t know which USB ports belong to which Bus that you verify this in the manual of your motherboard*.
- If VMetal doesn’t boot up after this process is complete click the launcher again.

You can also run VMetal from the Terminal using the following command:

```
LANG=C sudo -Sk vmetal
```

Using that command, you can check what VMetal is doing.

When you start VMetal, you will notice that the input devices you are using will not respond in the host; this means that the virtual environment has booted.

From this point forward you can install and configure Windows as you would in any computer. To visualize the virtual environment, switch the input of your monitor to the correct entry that connects into your non-primary display output.

- If you require further help with the configuration of Windows post-installation, please contact us at hello@nxos.org if you're a single user or at oem@nxos.org if you're an organization.
- *Please note that this kind of support may incur in charges in both cases*.

I have successfully booted VMetal, but my input devices don't work

VMetal will make use of the USB Bus #1 of your motherboard and the ports 1 through 7. While we are knowledgable that motherboards have more than one USB Bus, we can't be confident of how many more do they have. However, all motherboards will always have a USB Bus #1.

It's important to mention too that in some motherboards USB ports may belong to a USB Hub instead. It's important to understand because QEMU (currently) does not support using USB Hubs in a virtual environment. So if your motherboard groups USB ports in a USB Hub you won't be able to use input devices with VMetal.

This particularity is the case, for example for some older motherboards where USB 2.0 ports are grouped in a USB Hub, but USB 3.0 ports weren't.

I have tested VMetal, but I got an error: Unable to power on device, stuck in D3.

This error is caused by the motherboard firmware not being able to manage the power states of the device properly.

```
PCI devices supporting the PCI PM Spec can be programmed to go to any of the
supported low-power states (except for D3cold).  While in D1-D3hot the
standard configuration registers of the device must be accessible to software
(i.e. the device is required to respond to PCI configuration accesses), although
its I/O and memory spaces are then disabled.  This allows the device to be
programmatically put into D0.  Thus the kernel can switch the device back and
```

```
forth between D0 and the supported low-power states (except for D3cold) and the
possible power state transitions the device can undergo are the following:

+---------------------------+
| Current State | New State  |
+---------------------------+
| D0            | D1, D2, D3 |
+---------------------------+
| D1            | D2, D3     |
+---------------------------+
| D2            | D3         |
+---------------------------+
| D1, D2, D3    | D0         |
+---------------------------+

The transition from D3cold to D0 occurs when the supply voltage is provided to
the device (i.e. power is restored).  In that case, the device returns to D0 with
a full power-on reset sequence and the power-on defaults are restored to the
device by hardware just as at initial power-up.
```

More information about PCI power management can be found here.

A workaround for this problem is to either upgrade or downgrade the current UEFI firmware in your motherboard whatever your case may be.

How can I use VMetal in XYZ dsitribution?

At the moment VMetal is only available for Nitrux. We will eventually offer VMetal for other Linux, BSD, Solaris, and Unices systems.

My computer meets the requirements for VMetal, but I can't use it?

It's important to note that VMetal relies heavily on the hardware supporting the necessary features for it to work. This a list of issues that are known to affect the functionality of VMetal:

- **IOMMU grouping**. Most modern motherboard chipsets from 2011+ support IOMMU, VT-d, Vt-x or AMD-V, and AMD-Vi, respectively. However, even if the chipset supports it, the firmware used by the manufacturer may not allow for adequate IOMMU grouping. For example, VMetal requires two GPUs to be present, it may be the case that you have two individual GPUs in your system, but because of the firmware of your motherboard, these are grouped in the same IOMMU group, meaning that for VMetal to work both cards would have to be used by it. And this isn't an adequate scenario; as a user, to work around this, you have to downgrade or upgrade the motherboard firmware in the hope that a different version does not have this behavior.
- **AMD GCN 1st Gen. GPUs**. We often test hardware to check for any quirks in the user experience when utilizing VMetal. We have corroborated that while AMD Radeon GPUs utilizing the GCN 1st Gen. architecture can be used with VMetal, there is a case where a system that has two GCN 1st Gen. GPUs suffer from an issue where both cards are supported by both the AMDGPU and Radeon drivers in the Linux kernel and prevent the vfio-pci driver from being used. In this scenario, instead of one of the cards being used by the

vfio-pci driver, and the other being used by the AMDGPU or Radeon driver both GPUs are utilizing the enabled display driver (by default we force these cards to use the AMDGPU driver). To work around this problem, the user will have to proceed to do unbinding and rebinding of the desired GPU to use with VMetal. See the instructions below to achieve this.

- *Select AMD entry in GRUB.*
- *Press "E" and edit the boot entry not to blacklist the **amdgpu** module.*
- *After reaching the graphical interface, logout, and switch to a different virtual terminal (TTY) and stop SDDM. By default, Nitrux starts at TTY 1.*

```
sudo systemctl stop sddm.service
```

- *Find the vendor ID and slot number of the desired GPU to use with VMetal, and unbind it:*

```
lspci -nn | grep VGA
08:00.0 VGA compatible controller [0300]: Lorem ipsum dolor sit amet, consectetur adipiscing elit. [10de:1b80] (rev a1)

09:00.0 VGA compatible controller [0300]: Lorem ipsum dolor sit amet, consectetur adipiscing elit. [10de:1b80] (rev a1)

...

sudo echo -n "0000:08:00.0" > /sys/bus/pci/drivers/amdgpu/unbind
```

- *Add the device to vfio-pci to rebind it:*

```
sudo echo 10de:1b80 > /sys/bus/pci/drivers/vfio-pci/new_id
```

- *Restart SDDM.*

Can I use Windows software in Nitrux?

We include Wine in Nitrux for those users interested in running Windows software without using VMetal. Wine describes itself as "*Wine (originally an acronym for "Wine Is Not an Emulator") is a compatibility layer capable of running Windows applications on several POSIX-compliant operating systems, such as Linux, macOS, & BSD. Instead of simulating internal Windows logic like a virtual machine or emulator, Wine translates Windows API calls into POSIX calls on-the-fly, eliminating the performance and memory penalties of other methods and allowing you to integrate Windows applications into your desktop cleanly.*"

Do note that by using Wine alone, not all software written for Windows will work right away, further configuration by the user is necessary. We include Wine as an AppImage (version 3.10), and this means that it isn't an official release by the Wine developers. *The AppImage does include the necessary 32-bit libraries for Wine to work*.

We encourage users to visit Wine official site for support in running a specific Windows application; however, as our primary method of running Windows software is to use VMetal, which unlike Wine does offer full compatibility with Windows software albeit with higher system requirements.

Does Nitrux collect any data from my system?

No. We do not collect any form of data from your computer, and we do not have any intention or interest to do so in the future.

I have a question, where do I ask for help?

Please check https://nxos.org/en/nitrux/compendium.

I've found an issue with Nitrux, where do I report it?

Our bug tracker is here https://github.com/Nitrux/nitrux-bug-tracker.

Source code and license of Nitrux

Nitrux is made of hundreds of projects that are distributed as free and open-source software: you can redistribute and modify these projects in accordance to their respective licenses which include the terms of the GNU General Public License (GPL) as published by the Free Software Foundation. Nitrux is distributed in the hope that it will be useful, but without any warranty; use at your own risk.

The GNU GPL license requires that all source codes are published so others could reuse it, modify or learn from it.

For details about each license, please check each projects specific page.