Welcome

So, you've heard about self-hosting. Or maybe you have come to realize that you want to opt out of the cloud but are not exactly sure where to start. Or maybe someone sent you this website in order to convince you that you should.

Maybe you're approaching this topic as an individual, or maybe you represent a small group or collective with similar ideas. Maybe you're looking for some recommendations for docker images—or maybe you have no idea what that means.

This publication aims to pick you up wherever it is that you are right now, and to point you in a direction where you can continue your journey towards self-hosting. It's inspired by squatting manuals, texts that have codified and politicized the process of squatting since the 1970s, and that have been openly available to all that are interested. The authors of this publication think that setting up your own server is a political act, and no so unlike the action of squatting a house to live and work in.

The goal of this publication is to make self-hosting, a topic that comes with several technical hurdles, accessible to a more general public and beginners, while still being a useful resource to more advanced readers. No prerequisites are required from the reader.

> Today, we use more and more cloud services in our daily lives. These are often run by software companies with questionable ethical integrity. They often lock us in walled gardens and limit our agency as users. Our internet and technology usage has devastating consequences for the environment.
> Self-hosting can be a more ethical approach to digital infrastructure in your personal or professional life. The cloud is just a bunch of servers, and you can run your own server—in many cases, retired consumer hardware like an old laptop is enough! It requires some technical know-how, but most of the information is freely available and much can be learned along the way.

The first version of this text has been developed by Lukas Engelhardt with Paul Bille and Ada Reinthal as part of the Artsformation program on occasion of the Digital Shadows exhibition and funded by Waag in Amsterdam, NL. A print version that can be generated from this website was riso printed by Tjobo Kho in Amsterdam.

> You can also print pages by clicking Export to PDF at the bottom of the page!

This text is a work in progress, and we would like to open it up to others. **Especially the Practical Guides section needs a lot of work!** If you would like to contribute to this wiki, please email info (at) lukasengelhardt (.) net for access.

Problems with the Cloud

1.1 What is a cloud?

Today, in many professions, we are dependent on tools that work in the cloud. This becomes especially true when we work in teams. We have to write collaboratively, share files, conduct video calls and invite each other to calendar events. What might have seemed almost impossible a few decades ago, has become so simple that it has become almost invisible to us as users: Right click, share, send, done.

What a lot of us are not always aware of is that this seamless experience is enabled by an expansive network of digital and physical infrastructure that runs around the clock for our convenience. Or, in more simple terms: For us to collaborate in the cloud, somewhere in the world there is a very fancy, very fast computer that facilitates this collaboration. Usually these computers are located in vast data centers around the world, many of which are controlled by a small number of hard- and software conglomerates which, in turn, have come to use a significant percentage of the world's energy production.

1.2 Why is that a problem?

As mentioned above, this infrastructure is incredibly expensive from an energy usage/environmental point of view. Even though many data centers around the world are run with 100% renewable energies, in many cases they use so much power, that there simply is no more green energy left for the rest of the power grid, which remains powered through the use of fossil fuels.

As mentioned above, large parts of `the cloud` are run by large corporations like Amazon, Apple, Google and the like. Companies (and monopolies) with histories of

- worker exploitation in the global south,
- of supply chains that include child labor and extraction from indigenous lands,
- of producing abhorrent amounts ofe-waste that ends up on land fills in the global south,
- of spying on their users to sell them ads,
- of sharing this information with governments,
- of profiting from political division and right wing radicalization,
- of intentionally making their products addictive,
- of locking their users in incomprehensible terms and conditions.
- and of being aware of all of the above while not changing any of it.

We urgently need to ask ourselves if we really want one of these companies to facilitate the infrastructure on which we run our businesses, collectives, foundations, associations or individual practices. But besides these ethical considerations, there is the more practical issue of being dependent on them.

Most cloud services give the users little to no control about the way their files and data are handled. Instead, they intentionally lock users into so-called *walled gardens*, beautifully designed interfaces that are difficult to escape from. In practice, this often means

- users can't edit or access their files and documents when they are not connected to the internet,
- users don't know where files are physically saved, or how to recover them when they are lost,
- users can't easily switch to another service and take their documents, files and data with them,
- by excluding users from the way things work behind the scenes, they are kept dependent on these services and unable to set things up by themselves,
- users have to pay monthly fees in order to be allowed to keep accessing their data (see: Software as a Service (SaaS)),
- if there is an issue (like lost or accidentally deleted data), there is no customer service to turn to,
- users have to trust that any sensitive data is handled appropriately and have no control over security measures taken by these companies. Large corporations have large amounts of data, which makes them a target. They get hacked all the time, data is leaked all the time.
- As was the case with Twitter in 2022, companies can change owners, business models or Terms of Service within a few weeks, potentially limiting access to users' data or forcing them into new payment plans or dependencies. There is no stability in these (often very young!) companies.

1.3 Some notes on expense

As mentioned above, running services in the cloud is computationally expensive. All data is constantly fragmented, distributed and redundant, so that even if a whole data center were to break down, users wouldn't notice. But the cost (in compute power, bandwidth, energy consumption) of maintaining this level of availability is hidden behind the smooth interfaces that we as end users interact with: the edges are round and everything seems light, immediate, easy and immaterial.

As a result of this, our notion of the expense of the cloud is warped. Or, in other words, we are spoiled: any amount of *friction*—of something not happening immediately, or something not working exactly as intended—is perceived as an inconvenience, perceived as a divergence from the smooth *norm*. From this perspective, an effort like self-hosting (which almost certainly entails a process full of friction) can seem unfeasible or unpractical, while in reality it is the status quo that is unsustainable.

We urgently need to question whether everything really needs to be available 99.99% of the time and more. But we also need to rethink our notion of expense and accept that cloud services don't just happen by themselves. If we don't want companies like Amazon and Google to provide them to us at the aforementioned costs and ethical compromises, we need to invest the time and energy to set them up, and to maintain them ourselves. Maybe we have to stop looking at this as additional time that we have to invest on top of all the work we already do, but look at this as a key component of organizing—similarly to how we need to take care of finances, and how we would often have a dedicated member of any group, collective, foundation or association, that facilitates these tasks.

Luckily, there are actually a number of services and tools freely available for those that are willing and able to put in the necessary time and energy, which we will explore in this guide.

Hosting Yourself

At this point, it's probably good to explain what a server is in a few words: A server is a computer that *serves* information to a user (or *client*) machine. It's an essential part of everything *online*—arguably, the internet is just a bunch of servers.

When you visit a website, somewhere in the world there is a computer that has the website stored on it—that computer is one example of what we call a server. When you enter a URL into your browser, you are being connected to this server, and it will send you back (*serve* you) the content of the website. Your computer then displays this website in your browser, similar to the way a TV receives the signal from the TV station and renders it into an image on the screen.

Similarly, if you edit a Google Docs file together with someone else, somewhere in the world there is a server on which the content of this document is stored. Every time you make a change in the document, this change is sent to the server, which then passes on (*serves*) the changes you just made to all the other users, instantly. This way everyone is always in sync and can see what the other people are writing, in real time.

Obviously, on a practical level things are a bit more complicated than that but when people say *the cloud is just someone else's computer*, this is what they mean: the so-called cloud is just a bunch of servers that store everyone's data and keep everyone connected. In the case of big tech, these servers are part of huge data centers, incredibly complex operations that exist in specially designed buildings that use the energy equivalent of small cities. This scale is "necessary" because they serve millions of people simultaneously, all over the world, all of whom expect things to happen immediately and without a moment of delay or a second of downtime.

Self hosting is the opposite of that. Instead of having all your data on someone else's computer, as is the case with the cloud, it's setting up your own computer to do the same thing. At least theoretically, any computer can function as a server and while it requires some technical know-how, time and effort, it's possible to set up a small server in your own home or work place that can replace some or all of the cloud services you (and your colleagues?) use. Since you most likely don't need to serve millions of people at the same time, the hardware, software and energy requirements for this can be surprisingly low. If you want to supply file sharing, collaborative writing, video calling and a shared calendar for yourself and the people in your surroundings (let's assume this is up to 15 or 20 people), chances are that an old laptop that you or someone you have lying around would be up to the task.

2.1 What are the benefits of self-hosting?

As previously mentioned, self-hosting comes with a set of ethical and practical advantages:

- it allows people to gain (more) control over their data, both in terms of privacy and access to their files
- it allows them to become more self-sufficient in their digital infrastructure,
- it lets them build systems that can adapt and develop along with their needs while simultaneously enabling them to plan for longevity,
- it often allows them to build systems that are more closely adapted to their specific needs,
- in the process, users will learn a lot about the way digital infrastructures work, making them more adapt at avoiding mistakes and fixing problems in the future,

- It allows users to minimize their ecological footprint through lower energy usage and the recycling of old hardware,
- It makes them less dependent on companies that are diametrically opposed to their ethical values

2.2 What are the challenges of self-hosting?

But nothing is perfect and life is not fair, and like everything else, self-hosting comes at a cost. The trade-off here is mostly about the time and energy that an individual or group has to spend on it (more on this in our notes on expense).

- Self-hosting means maintenance. Software has to be updated, hardware has to be upgraded or exchanged, issues need to be solved every once in a while. This is a commitment that you need to be aware of before deciding to self-host (parts of) your digital infrastructure.
- While most things we describe here are really not rocket science, self-hosting always comes with a certain risk of just *fucking up*—of breaking something, accidentally deleting data, or of hardware failure. While there are ways to prevent irreversable damage, it's good practice to consider the consequences of an interruption of service before deciding to switch to self-hosting for a particular service.
- While it's possible to reduce your ecological footprint through practices like self-hosting (where you can use computers that are extremely energy efficient and/or reuse old hardware), there are limits. From a radical ecological point of view, computing, and especially approaches that are not offline-first, are not ecologically sustainable per definition. The materials that go into the production of chipsets are extracted at considerable costs to the environment, and the amount of energy that goes into the production of a modern computer cannot be compensated for, no matter how energy efficient it is. This is not even accounting for the fact that eventually it will probably end up in a landfill somewhere.

2.3 List of things that can be replaced by self-hosting

The list of self-hostable services is long and ever-growing. If there is a cloud service provided by some start-up, there is probably a way of hosting something similar yourself. This includes, but is not limited to:

- File hosting, like Dropbox or Google Drive
- Collaborative office suites, like Google Docs
- Shared calendars like Google Calendar,
- Email services like Gmail
- Web hosting like AWS

But there are many more applications to discover or play around with. You can self-host gaming servers (for example for minecraft), or media servers to connect to your TV.

Managing Expectations

If you are thinking about self-hosting some of your digital infrastructure, there are a few things to consider beforehand.

3.1 Becoming a sysadmin

Firstly, depending on your current level of expertise, you might have to prepare to take in a whole bunch of new information, and learn (a lot of) new things about computers. While you don't really need to know or learn programming itself, becoming a system administrator, or sysadmin for short, involves learning how to install and use Linux, and using the command line.

These things can seem scary and confusing to people that so far have only worked with graphical user interfaces (GUIs). But it's just about getting your hands dirty, learning by doing, and with a little bit of practice these things will seem much less daunting than they might do now! This guide includes a small introduction to the command line, and there are many articles on the internet that give beginners an introduction on how to interact with their computers through text based interfaces (see Graphical user interfaces, shells, consoles, and terminals for more information).

3.2 Time (well?) spent

As previously mentioned, self-hosting can be time-consuming (even though there are good reasons to think of it as necessary time-commitments rather than extra unnecessary work!). There is the initial time to set up a functioning system, starting with choosing the right hardware, installing an operating system, and setting up and configuring all the services that you want to run. We recommend taking some time to really dive into this, and give yourself enough time and space to figure things out.

And then there is maintenance. You should make a habit of updating your system and software periodically. You might have to install new services, and figure out ways in which they don't conflict with other services you have set up (even though using docker, as we strongly recommend, solves many of those issues). And you might have to switch out, or upgrade hardware. Something might break, and stop working, and you might have to figure out what it is and how to fix it. If you are the person in charge of the server, people in your surroundings might ask you to solve issues for them, sometimes at inconvenient times.

This does **not** mean that you will have to spend your days to make sure your emails still send—often it can be closer to one hour per week. But it's important to manage expectations, yours and those of others, and to realize that maintenance will require continuous engagement—similarly to how you have to clean your kitchen every so often, or how you have to keep your finances organized. In a lot of cases, it can make sense that these become the tasks of a dedicated individual or group of people.

Hardware

> **TL;DR**Basically, any computer that you have previously used as a workstation (for example, an old laptop) can become a server. If it doesn't have an Ethernet port, get an adapter (PCIe Ethernet for a desktop, USB-Ethernet adapter for a laptop). Depending on the scale of your operation, even single board computers like a Raspberry Pi can work, but remember: the only ethical hardware is second hand hardware. Except for hard drives, don't buy second hand hard drives.

1.1 System Requirements

There are a few factors that go into choosing the correct hardware for your project, mainly:

- Budget/availability
- Use case
- Environmental Footprint

The more concurrent users you expect (people that are accessing the server at the same time) the better your hardware needs to be. That being said, most web apps are not very compute intensive and rather rely on having a lot of RAM and the ability to process a lot of requests in parallel—CPUs with multiple cores are definitely beneficial here.

Most well refined systems use PHP, an old and reliable programming language for the web that is part of most websites on the internet today. However, since it is an old programming language, it is not very well optimized for a lot of parallel users. Most services that you might want to run on your server specify minimum system requirements, so it's probably good to read up on a few before making a choice.

1.2 Picking a machine

Basically any computer can be turned into a server, even an old smartphone! For very old or very small computers, you will have to assess if they are fast enough for what you have in mind. If you are reading this guide, any working old laptop is probably enough.

Some people like to work with single board computers (SBCs), like Raspberry Pis. While a Raspberry Pi will reach its limits quite quickly if you plan on running several services at the same time, there are other single board computers that are much more powerful and can handle ever larger jobs. Single board computers are often relatively cheap, use very low amounts of energy, and come in a small and convenient form factor.

If the form factor is not important, however, we recommend using an old laptop. Due to much fewer restrictions in terms of size, even an old laptop will likely outperform a new single board computer. The built-in battery, if it is still functional, can protect against fluctuations in the power grid. The built-in screen and keyboard, if still functional, eliminate the necessity for additional peripherals during set up and allow direct interaction with the machine. Due to their portability, laptops are usually designed to be much more power efficient than desktop machines. If you or someone you know doesn't already have an old laptop that you can use, chances are you can find one for free or very cheap online.

1.3 New or Used Hardware

It takes a lot of energy to build computers, and buying new hardware has a big impact on the environment. Even the most energy efficient new computer will not be able to offset the amount of energy that had to be used in its production. **We strongly encourage everyone to consider used hardware wherever possible.** A used laptop will always be more environmentally friendly than a new Raspberry Pi!

Apple

While it's certainly possible to install Linux on an old Apple computer like an old MacBook, and to turn it into a server, Apple has complicated this process significantly (basically: the newer the computer, the more hurdles you will have to overcome). Because the process changes so much depending on your model, we currently have not included a manual for how to install Linux on old Apple computers.

For the purpose of this guide, we will assume that you have access to an old Windows or Linux computer, as these are much easier and cheaper to come buy than old Apple computers. If you do have an Apple computer which you want to install Linux on, there are many guides such as this one which will walk you through the installation process. After the installation process is complete, you should be able to follow the guides in this manual as usual.

1.4 Storage

No matter what hardware you chose, you will often need to add additional storage, for example if you want to set up a file sharing service where each user has multiple Gigabytes or even Terabyte of storage allocated to them. But also in other cases, it's quite common to use different storage options for different purposes. A classic solution would be to use a small but fast disk to run your operating system from, and to save all the user data on a bigger, but potentially slower drive. There are multiple considerations here, but mainly it boils down to **speed**, **capacity**, **durability** and **expense**.

Types of storage

> HDDs are very cheap but slow and not durable. SSDs are fast and reliable, but expensive. In most cases, a hybrid solution is the most efficient: run the system on a small SSD that is fast and reliable, and store all data on multiple large and cheap HDDs.

The main types of storage are Hard Disk Drives(HDD) and Solid State Drives (SSD), though some single board computers also work with other storage mediums like **flash drives** or **SD cards**. While those are ok for testing and tinkering, it is not recommended to run a server that uses an SD card as data storage—SD cards deteriorate with every read/write operation, and they **will** fail eventually, leading to downtime and potentially data loss. It is possible to use an SD card to boot the system from, but to set it to read-only, and store all relevant data on another volume, but this is something that we will not cover in this guide as it is a bit of an edge case.

Hard Disk Drives (HDD)

Hard Disk Drives have been around for a long time. Simply put, they operate similar to a CD or DVD, with all data stored on a physically spinning disk. They are the slowest, type of storage, but also the cheapest. They are very big in comparison to other solutions and not very durable: Hard drives have a limited amount of read/write operations they can do before they will inevitably die, and when they do, they probably cannot be fixed without spending a considerable amount of money. Most servers are always on and especially when you have multiple users, this means that your hard drives will undergo a lot of read/write operations. **The question is not *if* an HDD will fail, but *when*.**

There are server-grade HDDs which are more resistant to wear, but cost more money. For the scope of self-hosting, those are probably not required, though. Even under continuous use, a new HDD should last a few years, and there are setups to prevent data loss even in case of an HDD dying.

> While we recommend buying used hardware wherever possible, we don't recommend this with HDDs.

> The longer a hard drive is in use, the more likely it is to fail. With used hard drives, it's difficult to know how much life they have left in them. There is also a chance that the second hard drives you find online have been used in crypto mining at some point, further increasing the likelihood of failure in the near future.

RAID configurations

Nothing good lasts forever, and that is especially true for HDDs. In anticipation of hard drive failure and data loss, you probably don't want to save all your data on one HDD. Fixing or recovering data from an old hard disk is magnitudes more expensive than a new disk, and not guaranteed to work. Instead, it's recommended to set up what is called a RAID configuration.

This means that multiple hard drives are set up in such a way that every piece of data is present on multiple disks, so that if one of the drives fails, no data is lost, and it can simply be replaced. The most simple RAID configuration is RAID-0, which is two hard drives that are exact mirrors of each other. Every time data is written to one of the drives, it's written to the other one as well. If one of the drives fails, you can simply exchange it without any data lost.

Obviously, this doubles the cost of storage, as you will have to buy double the amount of hard disks. But, especially if you plan to set something up for multiple users with the goal of maintaining service for a long period of time, you should seriously consider this. Also, (at the point of writing) two HDDs are still cheaper than the equivalent amount of storage on an SSD.

Network Attached Storage (NAS)

While you can set up your own RAID configuration with multiple hard drives, there are many solutions out there that do exactly this already, the most popular ones are probably by the brand Synology. They are actually small servers themselves that can connect to your computer over USB or Ethernet, and can hold multiple hard disks. Depending on where you live, you might be able to find a second hand NAS for relatively cheap online, potentially saving you time and effort. If one of the drives in

your NAS fails, simply replace it with a new one.

Solid State Drives (SSD)

Solid State Drives are a newer technology than HDDs that address most of their shortcomings: they are much smaller, much faster and much more durable. They are, however, also much more expensive. Unless money is not a factor for you, it's probably not very feasible to use SSDs to store large amounts of data. SSDs are however the preferred medium to run an operating system from, as this will have a big impact on performance while requiring relatively little space. As opposed to HDDs, SSDs are probably safe to buy second hand.

1.5 Network connection

While it's certainly possible to run a server over Wifi, Ethernet is much faster and much more reliable. Unless you really don't have access to an Ethernet port we would always recommend you to connect your server to the internet over Ethernet. Most single board computers and Windows laptops come with Ethernet ports, if you have a computer without one, you should probably look into finding an adapter!

1.6 Where to get old hardware

Computers quickly lose value. Improvements in performance and capacities quickly render older machines obsolete, and computers that are older than 10 years often can't even receive updates anymore. This means that very few people would invest much money into old hardware, which keeps the prices low. This is good news for the happy self-hoster, because the requirements for self-hosting are comparatively low.

With a little bit of luck, on websites like Marktplaats (in the Netherlands) and eBay (in many other parts of the world) it should be possible to find old laptops for cheap or even for free. It's also worth asking in your friend circle and/or professional environment if people have old computers that they don't use anymore.

System

> When setting up a server, you want to use Linux. And you want to do so without a GUI.

2.1 Who's the Penguin?

If setting up single board computers and servers is not something you do regularly, chances are your knowledge of (non-mobile) operating systems is mostly limited to Apple's macOS and/or Microsoft Windows. When it comes to consumer hardware, both have their advantages and disadvantages (for example that they are proprietary software developed by evil tech conglomerates), but neither are very well suited to run servers. This is where Linux comes in.

Linux is a family of free and operating systems. Open source means that anyone can download and modify the code of Linux and make their own version of it. Because of this, there are countless different versions of Linux, that are commonly called *distros*. While you might not be aware of it, you almost certainly have come in contact with it before, since Android, the operating system that powers most non-Apple smartphones, is based on Linux.

There are many advantages to using Linux over other operating systems, but while its user base has been steadily growing in recent years, on Laptops and Desktop PCs it's probably still a bit of a niche phenomenon. This however is not true at all when it comes to servers: the cloud (more or less) runs on Linux, and for good reasons:

- It's free. That means you can spin up a new server any time without acquiring a license before
- It's secure
- It's stable
- It runs on basically any kind of hardware
- It's customizable: due to its open source character, there are countless *distros*, each with different flavors, features and focuses, and optimized for different scenarios.
- It's lightweight: as opposed to a consumer PC, which should be good at all kinds of things, a server is usually meant to do only one thing, but do that thing really well. Linux gives you the possibility to build an operating system with a focus on one specific task without wasting resources on anything else.

2.2 Headlessness

Probably the most prominent example of cutting out unnecessary things is the GUI, or rather the lack thereof. We are used to interacting with so-called graphic user interfaces: Windows, Mouse cursors, background images, things to be dragged into folders that contain them, etc. This great to organize photos, browse the web, watch movies. But it also costs a lot of resources. We don't want to have the same level of interaction with a server. We want it to sit in a room somewhere, headless (meaning without a screen, keyboard and mouse connected to it), doing its job.

This is why most servers run on operating systems that have reduced their interfaces to the bare minimum: a text based console. You type in a command, the server gives a response—all via text, like in the old days.

To many, the command line still has an air of Y2K-action-movie-hacker to it, and it can seem a bit

daunting. And it does take a bit of getting used to—sometimes it can be disorientating or confusing. Things that take a simple gesture on a consumer PC suddenly take multiple commands and a lot of troubleshooting. But once you got the hang of it, you will realize that it's actually far more powerful than the GUIs we normally interact with.

Currently, the scope of this wiki has not yet allowed us to write an in-depth guide to the command line. We would like to do this in the future, though. For now, we will just tell you to search the internet for things like "command line for beginners", you will find countless guides such as this one that should give you a pretty good overview. Further than that, we think the best way to learn these kinds of things is to get your hands dirty and follow one of our guides.

2.3 DietPi

As mentioned above, there are countless Linux Distributions, each with their own sets of pros and cons. In our guide, we recommend DietPi: originally developed as a lightweight system for the Raspberry Pi, it now supports most SBCs as well as native PCs. It has an active community, is straight forward, and super light weight, meaning that it's very reduced in functionality and that it will use very few resources.

There are many other distros that can make sense, depending on preference and situation. If you prefer something else, feel free to install that instead; since our guides recommend using docker, you should be able to follow along nonetheless.

Software

> Don’t argue, use docker

When using a consumer PC, installing software has become so easy we barely notice it. You either open an installer and click yes—yes—yes until the installation is complete (without reading the Terms and Conditions, of course), or you simply drag an icon into another icon, or you just click `install`.

But under the hood what is happening is a bit more complicated than that. This is where docker comes in and make our lives better. But let’s start by explaining what exactly the problem is that docker solves.

3.1 The problem: dependencies

When writing software, you try not to reinvent the wheel. Instead, you *depend* on things that other people have written before you—a programming language maybe, a database, an image library. *Dependencies* are a sysadmins nightmare, especially when trying to run multiple services simulataneously.

For example: a file server might run on PHP. This usually means that it requires a specific version of PHP, a programming language that gets updated continuously. But this file server is also supposed to have a web interface—a website where users can look at their files. This requires a web server to be installed. The sysadmin of this server now wants to install a second service on the same server, a note-taking app. This note-taking app might also require PHP, but a different version, so there is a conflict here. On top of that, it also requires a web server, so that users can edit their notes in the browser, but this specific app needs a different type of web server. So there is another conflict here!

3.2 The solution: containers

The more things you want to run in parallel, the more you will run into things like this. That’s where containers come in.

Containers, as the term suggests, “contain” every little thing that an application may need to be executed. It’s like a transportable box, filled with tools and libraries and settings and all that other good stuff that we need to run the application. Docker, which we strongly recommend, is a container platform. In the example above, when using docker, both the file server and the note-taking app would come in containers, packed up together with their own PHP versions and web servers, and without the risk of interference. In fact, it becomes so easy, you don’t even have to know what dependencies the things you install actually need.

3.3 I’m sold, where do I find containers?

A good place to start looking for docker containers is the website linuxserver.io, though if you have an application you would like to run in mind, you can probably simply search it up online by adding

“docker” to your query.

Related Projects

Guides

Homebrew Server Club

The homebrewserver.club is a monthly gathering for those who (wish to) host their own online services from home, rather than using commercial and privacy unfriendly alternatives. (…)

Without Homebrew Server Club this wiki might not exist (yet). It's an immensely useful resource on self-hosting (and other approaches to computing) and serves as one of the biggest inspirations for this wiki. Link

Scrapism

A guide on how to scrape the web, written by the artist Sam Lavigne. Web-scraping is the act of accumulating large amounts of data. It can be done by hand, but usually the process is automated with a computer program. Link

Theory

Permacomputing Wiki

Permacomputing is a more sustainable approach to computer and network technology inspired by permaculture. Permacomputing is both a concept and a community of practice oriented around issues of resilience and regenerativity in digital technology. (…)

Started in May 2022, the permacomputing wiki is where we maintain a set of principles, collectively document practical perspectives and interpretations of these principles, and offer more speculative reflections on permacomputing. Link

Elephant in the Room

Constant wrote an open letter to cultural institutions about their use of commercial platforms and proprietary technology. Constant is a non-profit organization based in Brussels since 1997 and active in the fields of art, media and technology.

Constant is a non-profit organization based in Brussels since 1997 and active in the fields of art, media and technology. Link

Feminist Server Manifesto

In 2013, *Constant*, a non-profit, artist-run organization in Brussels, hosted the workshop *Are you being served?* During the session: *First Feminist Server Summit*, artists and activists reflected on questions around the potential of a Feminist Server practice. Link

A wishlist for trans*feminist servers

A 2022 rewriting of the 2013 text, retitling it “Trans*feminist servers…”, and removing the denotation “manifesto”. Instead of a straightforward declaration of intentions, this text is an ambiguous ongoing wishlist for techno-ecologies in the making; an ongoing set of spells for a different tech for this world, for different tech for different worlds. Link

Praxis

Low Tech Magazine

Low-tech Magazine underscores the potential of past and often forgotten technologies and how they can inform sustainable energy practices. Technology has become the idol of our society, but technological progress is—more often than not—aimed at solving problems caused by earlier technical inventions. Link

Glossary

Most of the information on this page has been taken from the simple English Wikipedia. It’s a bit of an incomplete list and a work in progress.

Machine

Booting

Booting is what happens when a computer starts. This happens when the power is turned on. It is called “reboot” if it happens at other times. When you boot a computer, your processor looks for instructions in system ROM (Read only memory, the BIOS) and executes them. They normally wake up peripheral equipment and search for the boot device. The boot device either loads the operating system or gets it from someplace else.

CPU (Central Processing Unit)

A central processing unit (CPU) is an important part of every computer. The CPU sends signals to control the other parts of the computer, almost like how a brain controls a body.

ARM

ARM architecture is a computer CPU architecture used in computers of all sizes up to supercomputers; commonly used in embedded systems and mobile devices such as cell phones, tablet computers, and handheld game consoles such as the Game Boy Advance.

ARM CPUs use very little electricity and produce very little heat. Most ARM CPUs run on battery power and don’t need a cooling fan. The Linux operating system is used most on ARM CPUs.

x86

x86 is a term used to describe a CPU instruction set compatible with the Intel 8086 and its successors, including the Pentium and others made by Intel and other companies.

This is the CPU architecture used in most desktop and laptop computers. Many 21st century workstations and servers also use x86 processors.

SBC (Single Board Computer)

A single-board computer (SBC) is a complete computer built on a single circuit board, with microprocessor(s), memory, input/output (I/O) and other features required of a functional computer. Single-board computers are commonly made as demonstration or development systems, for

educational systems, or for use as embedded computer controllers.

System

File Systems

A File system (or filesystem) is a way of storing all data on a data storage device. The data is usually organized in computer files in directories. Below the file system there is usually a physical device where the files are stored.

This might be a hard disk, USB flash drive, compact disc, or DVD. The file system might also talk to a remote server over a network where the file is stored. The file system might also only use RAM to store the files.

Kernel

A kernel is the central part of an operating system. It manages the operations of the computer and the hardware, most notably memory and CPU time.[1] Kernels also provide services which programs can use through system calls.

Linux

Linux or GNU/Linux is a Unix-like operating system (or family of) for computers. An operating system is a collection of the basic instructions that manage the electronic parts of the computer allowing running applications and programs.

The Linux kernel (the basis of the operating system) is free software, meaning everyone has the freedom to use it, see how it works, change it, or share it.

Ubuntu

Ubuntu is a free operating system that uses the Linux kernel. The word “ubuntu” is an African word meaning “humanity to others”. It is pronounced “oo-boon-too”.

It is one of the most popular Linux distributions and it is based on Debian Linux computer operating system.

Debian

Debian is a free operating system. It is a distribution of an operating system known as the GNU operating system, which can be used with various kernels, including Linux, kFreeBSD, and Hurd. In combination with these kernels, the operating system can be referred to as Debian GNU/Linux, Debian

GNU/kFreeBSD, and Debian GNU/Hurd, respectively. Debian GNU/Linux is one of the most complete and popular GNU/Linux distributions, on which many others, like Ubuntu, are based.

DietPi

DietPi is an extremely lightweight Debian-based OS. It is highly optimised for minimal CPU and RAM resource usage, ensuring your SBC always runs at its maximum potential.

GNU (GNU's Not Unix)

GNU is the name of a computer operating system. The name is short for GNU's Not Unix. Richard Stallman leads the GNU Project. The popular Linux operating systems made using the Linux kernel have many GNU tools too. So, many projects and developers call the Linux-based operating systems GNU/Linux.

The GNU project was started by Richard Stallman in 1983. He wanted to create a computer system that was all free and open-source software. Users could change, share and publish new work based on GNU. He and a group of developers started by creating copies of each piece of UNIX software. The rest of GNU was the kernel, called the GNU Hurd, which is not yet finished. The more popular Linux kernel is often used instead.

Software

Docker (software)

Docker is a technology that bundles a software program with all of the other software that application needs to run, such as an operating system, third-party software libraries, etc. Software bundled like this is called a **container**.

The benefit of using Docker to put applications in containers is that they can be run on different kinds of computers (for example, both a laptop and a web server), without the risk of a missing software library or a different operating system causing the application to not work.

Graphical user interfaces, shells, consoles, command-lines and terminals

Computers can display information and let the user give commands to it using two methods: a command line interface (CLI) or a graphical user interface (GUI).

In a command line interface, the user types commands using the keyboard to tell the computer to take an action. For example, the more command available in most operating systems will display the contents of a file. Sometimes people call CLI "Console", but consoles can be GUI, too.

Shell

An operating system shell is a user interface that enables the user to interact with and access the services offered by the operating system. The user gives commands to the operating system through its shell.

There are various types of shells:

- Command line shells: the user types commands at the prompt.
- Menu driven shells: the user selects commands from menus.
- Graphical user interface shells: the user selects graphical menus and icons.

Examples of command line operating systems are UNIX and Disk Operating System (DOS). Examples of menu driven operating systems are the DOS shell. Finally, examples of graphical user interface (GUI) operating system are Linux and Microsoft Windows.

Sytem Console

One meaning of system console, computer console, root console, operator’s console, or simply console is the text entry and display device for system administration messages, particularly those from the BIOS or boot loader, the kernel, from the init system and from the system logger.

It is a physical device consisting of a keyboard and a screen, and traditionally is a text terminal, but may also be a graphical terminal. System consoles are generalized to computer terminals, which are abstracted respectively by virtual consoles and terminal emulators.

Text Terminal

A text terminal, or often just terminal (sometimes text console) is a serial computer interface for text entry and display. Information is presented as an array of pre-selected formed characters. When such devices use a video display such as a cathode-ray tube, they are called a “video display unit” or “visual display unit” (VDU) or “video display terminal” (VDT).

The system console is often a text terminal used to operate a computer. Modern computers have a built-in keyboard and display for the console. Some Unix-like operating systems such as Linux and FreeBSD have virtual consoles to provide several text terminals on a single computer.

Command Line Interface (CLI)

A means of interacting with a computer program where the user (or client) issues commands to the program in the form of successive lines of text (command lines).

Linux Console

The Linux console is a system console internal to the Linux kernel.[1] A system console is the device which receives all kernel messages and warnings and which allows logins in single user mode.

Command Line Cheat Sheets

A list of cheat sheets for different command-lines environments.

- Linux Cheat Sheet
- Mac OS Terminal Cheat Sheet
- Bash Cheat Sheet
- SSH Cheat Sheet

Network

Server

In computing, a server is a piece of computer hardware or software (computer program) that provides functionality for other programs or devices, called “clients”. This architecture is called the client–server model. Servers can provide various functionalities, often called “services”, such as sharing data or resources among multiple clients, or performing computation for a client.

A single server can serve multiple clients, and a single client can use multiple servers. A client process may run on the same device or may connect over a network to a server on a different device. Typical servers are database servers, file servers, mail servers, print servers, web servers, game servers, and application servers.

Cloud

The cloud is a metaphor for the Internet based on how it is described in computer network diagrams. Just as how in the real world, clouds hide parts of the sky from sight, the cloud in computing hides the complex infrastructure that makes the Internet work.

Software as a Service (SaaS)

Software as a service is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted. SaaS is also known as “on-demand software” and Web-based/Web-hosted software.

Web hosting service

A web hosting service is a type of Internet hosting service. It allows people and companies to make their website available on the World Wide Web. Web hosts are companies which provide space on a server which is owned or leased for use by clients. These clients store their Web site on the server. The server feeds the web pages to the Internet.

Domain name

A domain name is a human-readable web address that points to an IP address and helps users to access websites or other resources in a convenient way.

IP address

An IP address (short for **Internet Protocol** address) is a label which is used to identify one or more devices on a computer network, such as the internet. It can be compared to a postal address. An IP address is a long number written in binary.

Since such numbers are difficult to communicate, IP addresses are usually written as a set of numbers in a given order. Devices using IP addresses use the internet protocol to communicate.

TCP/IP

The TCP/IP model (Transmission Control Protocol/Internet Protocol) is a model with four layers which is for both modelling current Internet architecture, as well as providing a set a rules that govern all forms of transmission over a network.

DARPA, an agency of the United States Department of Defense,created it in the 1970s. It evolved from ARPANET, which was an early wide area network and a predecessor of the Internet. The TCP/IP Model is sometimes called the Internet Model or less often the DoD Model.

Administration

Production Ready

When we refer to an environment being production-ready we mean, that the server and the software you installed on it, runs, is backed-up, stable, maintainable and that it satisfies the needs for whatever you are doing on a regular basis.

Maintenance

The technical meaning of maintenance involves functional checks, servicing, repairing or replacing of necessary devices, equipment, machinery, building infrastructure, and supporting utilities in industrial, business, and residential installations.

Software maintenance in software engineering is the modification of a software product after delivery to correct faults, to improve performance or other attributes.

In a broader context maintenance can simply mean taking care of something, often over a prolonged period of time.