Common Terms and Concepts of Selfhosting

Servers

Servers are machines whose purpose is to provide a service or content over a network. They are typically administered remotely and only connect physically to power and a network. They "serve" content or services using software daemons. Bare metal servers are not virtualized. Any service or content they offer is configured on the host system. They are not new *per-se*, but with the introduction of containerization and virtualization, the phrase has been coined to differentiate the old-school server tradition from newer techniques. Their natural habitat is the datacenter, where they live in racks to survive off electricity and network data. While they are not able to reproduce, they have no natural predators, so their population is stable. Some breeds of server can be found in network/data closets where they live in a business. Fewer are still kept in captivity in private homes. Virtual servers are servers that are run under an emulator or hypervisor to provide a server-like environment using a software envelope which may be augmented with hardware support.

Daemons

Daemons are software packages that run perpetually to provide content or a service. They differentiate servers from clients. Examples of daemons are webservers, email servers, file servers, authentication services (AD, LDAP), database servers, and many more.

Webservers

Webservers are daemons that accept HTTP requests and serve set content based on the requested host (IP address or domain name). The content can be static HTML/XML or it can be dynamic (JavaScript, PHP, FCGI, WSGI). Webservers commonly offer reverse proxy functionality, it is common to use webservers for this purpose instead. Common webservers include: Apache, Cherokee, LiteSpeed, Lighttpd, nginx, and IIS. Apache and nginx are the top webservers by market share respectively, with IIS coming in third.

Domain Names

Domain names are a word, phrase, or string that is used for navigating the Internet. They are registered to individuals or legal entities in lengths of years for a set fee. They are divided into levels, where each level is separated by a period (dot). Domain registrations include the top-level and second-level portion of a domain. All levels below are controlled by DNS at the discretion of the domain registrant. Top-level domains (TLDs) are .com, .net, .info, .edu, .org, etc. The customizable part of the domain name you can register is called the second-level domain. Third-level domains are referred to as *subdomains*.

Structure: subdomain.*secondleveldomain*.**tld**

E.g.: fr.*vainsta*.**fr**

Domain Registration

Registering a domain name is done with a *Domain Registrar*. Prices are based on the top-level domain, but all registrations are for a period of one year minimum. Registrars come in two flavors:

- Accredited: These registrars work directly with ICANN or other regional Internet registries for domain registrations.
- Domain Resellers: These companies work with a "white label" registrar to resell domain registrations for a small markup.

Accreditation requires quite a bit of infrastructure and vetting to make sure you can handle all aspects of registering and maintaining domains on behalf of the registrant. Resellers are popular because of low overhead and easy implementation. Many "white label" registrars have turnkey solutions for resellers to appear as independent registrars while actually reselling domain names.

Which Is Right For You?

Choosing a domain registrar is easy. Picking a domain registrar that is trustworthy and reputable is less so. Many domain registrars also offer to handle the DNS records for the domains registered with them. Many registrars have domain registration as a part of their business. Registration is usually bundled with webhosting or other related services. You may even get a domain registration for free if you agree to a year-long hosting contract with a webhost. While bundling related services together under one roof may sound convenient, it is generally not a good idea. It is recommended to have your domain registration with a registrar, DNS records with another company, and hosting with yet another entity. Common reasoning for this piece of advice is that if your service provider has a serious outage or other technical problem, it can only affect one aspect of your online presence. If you have all services under one provider, a technical issue could prevent your DNS from resolving and your website/service from being served.

You can find the list of ICANN-accredited domain registrars here.

As far as finding a reputable, trustworthy service provider, we must insist on your own research.

One of the most popular forums for discussing hosting and related services is **Web Hosting Talk**. If a relevant service provider has a bad reputation in the industry, you can surely find out about it here.

Domain Name System

The Domain Name System (DNS) is the method of defining what unique machines serve content for your domain. The important parts of DNS you have to worry about are *nameservers* and *DNS records*.

Nameservers

Nameservers are a way to declare which servers are responsible for answering record requests for your domain. Most registrars provide DNS services, but if you have your DNS provided elsewhere, you will want to provide your primary and secondary nameservers to your registrar. The nameservers to use will be provided by your DNS service provider.

E.g.: `ns1.dnsnameserver.net`, `ns2.dnsnameserver.net`

DNS Records

DNS records are part of your domain name configuration called a DNS *zone*.

- **SOA**: Start of Authority Records is generally handled by your DNS service provider automatically. They define:
  - **MNAME**: Master nameserver for the zone.
  - **RNAME**: Email for the domain administrator. Does not support "@", use periods. Periods before the domain name are escaped. E.g.: `some.one@example.com` => `some\\.one.example.com`.
  - **SERIAL**: The DNS zone **serial**, used to indicate when a zone has changed.
  - **REFRESH**: Time to wait for secondary nameservers to query the master.
  - **RETRY**: Timeout for refreshing.
  - **EXPIRE** Threshold time for secondary nameservers to stop attempting to reach an unresponsive master server.
  - **TTL**: The time to live to use for NXDOMAIN responses.

Example SOA:

```
$TTL 86400
@   IN  SOA     ns1.nameserver.com. postmaster.sumdomain.com. (
```

```
    2020080302  ;Serial
    7200        ;Refresh
    3600        ;Retry
    1209600     ;Expire
    3600        ;Negative response caching TTL
)
```

The fields of a DNS zone record are:

- **Domain**: Either the domain name or subdomain to create a record for.
- **Time to live**: The time in seconds for a record to be cached before a new copy is requested.
- **Class**: Indicative of the namespace. Usually IN (Internet namespace).
- **Type**: The type of record to define.
- **Content**: The content of the record. What is acceptable in this field is dependent on the type of record.

There are many types of DNS records, let's go over some common ones. This list is **not** exhaustive.

- **A**: A records tie the domain or subdomain to an IPv4 address.
- **AAAA**: AAAA records tie the domain or subdomain to an IPv6 address.
- **CNAME**: CNAME records tie the domain or subdomain to another domain or subdomain.
- **MX**: MX records are used to define how mail is handled for your domain. The content of an MX record is the priority and answering server domain name. Lower preference number indicates higher priority.
- **TXT**: Text records associate text data with your domain. They are used for a variety of reasons, notable for SPF or DKIM.

Domain	Time To Live	Class	Type	Content
example.com.	86400	IN	A	192.168.1.240
ipv6.example.com.	86400	IN	AAAA	feef:00bb:2005:1eef:fbca:544d
www.example.com.	86400	IN	CNAME	example.com.
example.com.	86400	IN	MX	10 mail.mailserver.com
example.com.	86400	IN	TXT	"Reserved for a purpose I am not legally required to disclose."

Reverse Proxies

Reverse proxies are daemons that accept connections and then connect to another service based on port or host to facilitate the request. They act as a middleman instead of a traffic redirector.

Typical use cases for reverse proxies are to provide a unified frontend for multiple backends or hosts. Another common use is for high-availability to provide failover or distribute load between multiple backends serving the same content.

Examples of popular software capable of performing as a reverse proxy are: Squid, HAProxy, Apache, nginx, and Caddy.

Port Forwarding

Port forwarding is the function of inspecting traffic on an incoming port and redirecting it to another port or host with minimal modification. Primary purposes of this are to forward traffic to a service behind a firewall/router.

Common for hosting game servers from home when running dedicated servers before developers moved to match-making. Another use for this is to open ports for BitTorrent so that you can share your vast and innumerable collection of Linux ISOs.

The difference between port forwarding and a reverse proxy is that the reverse proxy will accept, process, and establish a new connection to the backend service to fulfil the request.

Port forwarding inspects and alters packet headers before it is routed to its new destination. The connection is otherwise untouched.

Port forwarding is a function of your firewall. Commonly at the router or other network gateway.

Linux has two firewalls called `iptables` and `nftables`, with many frontends or management packages available for them. BSD-based firewalls are `pf`, `ipfw`, and `IPFilter`. The Windows firewall consists of a scarecrow holding a sign saying: "plz no tresspass".

Containers

Containers are software envelopes to isolate a piece or bundle of software and their dependencies. Containers come in many forms. A container could contain a PHP-based forum with an AMP stack (Apache, Maria DB, PHP) as dependencies. This is useful if you want an easy way to deploy software without configuring dependent software/libraries manually. Containers can also resolve

software conflicts when running multiple services which depend on different versions of the same software/libraries.

Popular containers are Linux Containers (LXC), jails (BSD UNIX), Kubernetes, and Docker.

Virtualization

Virtualization is a lower level form of containerization. There are many forms of virtualization that provide different sets of features/tradeoffs. In practice, it often virtualizes whole or major parts of an operating system.

Full virtualization

Full virtualization is generally understood as the containerization of a full, unmodified operating system with virtualized hardware. The virtualized OS is not host-aware. Fully virtualized guests require more overhead than paravirtualized guests. This can be mitigated with hardware support (Intel VT, AMD SVM) for virtualization instructions.

Examples of this are Hyper-V, Xen, KVM/Qemu, VMware ESXi.

Paravirtualization

Paravirtualization is the practice of running a modified kernel/OS where privileged instructions are sent through an API shared with the host. It does not require the virtualization of hardware, but it does require an operating system that is modified to be used with the specific API used by your chosen virtualization method. This can be in the form of source code modifications or specialized device drivers.

> Microsoft Windows cannot be paravirtualized.

Examples: Xen, Oracle VM, OpenVZ.

Virtual private networks

Virtual private networks (VPNs) are a way of networking individual machines together in software regardless of their physical or network proximity. A typical use case is for networking corporate locations together to share network resources such as file shares, intranet webservers, on-premises

services, etc. Another use for a VPN is to tunnel traffic destined for a public service through to another endpoint, usually to bypass geo-location restrictions or state-imposed censorship of the Internet.

Some people use VPNs so they can access their services that are behind a restrictive ISP or firewall.

Operating Systems

An operating system is the software that is responsible for running and managing your physical machine. It provides the kernel, hardware drivers, low-level software packages, libraries, and userland applications for the end-user to provide basic functions.

Most consumers or end-users will use Microsoft Windows or macOS as their operating system on their desktop or laptop computers.

You will commonly find that corporate IT infrastructure used to serve employees is Microsoft-based, using ActiveDomain (AD) for authentication, Exchange for email and groupware, IIS for serving websites, and MSSQL for databases.

Linux or UNIX-based operating systems are the popular choice for hosting services and serving content to end-users. Examples are webhosts for serving websites, Netflix for serving movie and TV streams, DNS services for domain records, and most if not all other infrastructure needed to keep the Internet operational.

Android is a Linux-based operating system used in the majority of the smartphone market. iOS is a BSD-based mobile OS used by Apple for iPhones.

Embedded Linux and BSD are also used in devices like set-top boxes, smart TVs, routers, smart switches, medical equipment, flight telemetry controllers for aerospace, navigation equipment, industrial automation, etc...

Created 2025-06-24 19:01:03 UTC by Nicolas
Updated 2025-06-24 19:03:33 UTC by Nicolas