

By: Justin Ellingwood

 Subscribe Share Contents ▾

An Introduction to DNS Terminology, Components, and Concepts

Posted Feb 18, 2014  158.9k DNS Linux Basics FAQ

Tutorial Series

This tutorial is part 1 of 7 in the series: [An Introduction to Managing DNS](#)

Introduction

DNS, or the Domain Name System, is often a very difficult part of learning how to configure websites and servers. Understanding how DNS works will help you diagnose problems with configuring access to your websites and will allow you to broaden your understanding of what's going on behind the scenes.

In this guide, we will discuss some fundamental DNS concepts that will help you hit the ground running with your DNS configuration. After tackling this guide, you should be ready to [set up your domain name with DigitalOcean](#) or [set up your very own DNS server](#).

Before we jump into setting up your own servers to resolve your domain or setting up our domains in the control panel, let's go over some basic concepts about how all of this actually works.

Domain Terminology

We should start by defining our terms. While some of these topics are familiar from other contexts, there are many terms used when talking about domain names and DNS that aren't used too often in other areas of computing.

Let's start easy:

Domain Name System

The domain name system, more commonly known as "DNS" is the networking system in place that allows us to resolve human-friendly names to unique addresses.

Domain Name

A domain name is the human-friendly name that we are used to associating with an internet resource. For instance, "google.com" is a domain name. Some people will say that the "google" portion is the domain, but we can generally refer to the combined form as the domain name.

The URL "google.com" is associated with the servers owned by Google Inc. The domain name system allows us to reach the Google servers when we type "google.com" into our browsers.

IP Address

An IP address is what we call a network addressable location. Each IP address must be unique within its network. When we are talking about websites, this network is the entire internet.

IPv4, the most common form of addresses, are written as four sets of numbers, each set having up to three digits, with each set separated by a dot. For example, "111.222.111.222" could be a valid IPv4 IP address. With DNS, we map a name to that address so that you do not have to remember a complicated set of numbers for each place you wish to visit on a network.

Top-Level Domain

A top-level domain, or TLD, is the most general part of the domain. The top-level domain is the furthest portion to the right (as separated by a dot). Common top-level domains are "com", "net", "org", "gov", "edu", and "io".

Top-level domains are at the top of the hierarchy in terms of domain names. Certain parties are given management control over top-level domains by ICANN (Internet Corporation for Assigned Names and Numbers). These parties can then distribute domain names under the TLD, usually through a domain registrar.

Hosts

Within a domain, the domain owner can define individual hosts, which refer to separate computers or services accessible through a domain. For instance, most domain owners make their web servers accessible through the bare domain (example.com) and also through the "host" definition "www" (www.example.com).

You can have other host definitions under the general domain. You could have API access through an "api" host (api.example.com) or you could have ftp access by defining a host called "ftp" or "files" (ftp.example.com or files.example.com). The host names can be arbitrary as long as they are unique for the domain.

SubDomain

A subject related to hosts are subdomains.

DNS works in a hierarchy. TLDs can have many domains under them. For instance, the "com" TLD has both "google.com" and "ubuntu.com" underneath it. A "subdomain" refers to any domain that is part of a larger domain. In this case, "ubuntu.com" can be said to be a subdomain of "com". This is typically just called the domain or the "ubuntu" portion is called a SLD, which means second level domain.

Likewise, each domain can control "subdomains" that are located under it. This is usually what we mean by subdomains. For instance you could have a subdomain for the history department of your school at "www.history.school.edu". The "history" portion is a subdomain.

The difference between a host name and a subdomain is that a host defines a computer or resource, while a subdomain extends the parent domain. It is a method of subdividing the domain itself.

Whether talking about subdomains or hosts, you can begin to see that the left-most portions of a domain are the most specific. This is how DNS works: from most to least specific as you read from left-to-right.

Fully Qualified Domain Name

A fully qualified domain name, often called FQDN, is what we call an absolute domain name. Domains in the DNS system can be given relative to one another, and as such, can be somewhat ambiguous. A FQDN is an absolute name that specifies its location in relation to the absolute root of the domain name system.

This means that it specifies each parent domain including the TLD. A proper FQDN ends with a dot, indicating the root of the DNS hierarchy. An example of a FQDN is "mail.google.com.". Sometimes software that calls for FQDN does not require the ending dot, but the trailing dot is required to conform to ICANN standards.

Name Server

A name server is a computer designated to translate domain names into IP addresses. These servers do most of the work in the DNS system. Since the total number of domain translations is too much for any one server, each server may redirect request to other name servers or delegate responsibility for a subset of subdomains they are responsible for.

Name servers can be "authoritative", meaning that they give answers to queries about domains under their control. Otherwise, they may point to other servers, or serve cached copies of other name servers' data.

Zone File

A zone file is a simple text file that contains the mappings between domain names and IP addresses. This is how the DNS system finally finds out which IP address should be contacted when a user requests a certain domain name.

Zone files reside in name servers and generally define the resources available under a specific domain, or the place that one can go to get that information.

Records

Within a zone file, records are kept. In its simplest form, a record is basically a single mapping between a resource and a name. These can map a domain name to an IP address, define the name servers for the domain, define the mail servers for the domain, etc.

How DNS Works

Now that you are familiar with some of the terminology involved with DNS, how does the system actually work?

The system is very simple at a high-level overview, but is very complex as you look at the details. Overall though, it is a very reliable infrastructure that has been essential to the adoption of the internet as we know it today.

Root Servers

As we said above, DNS is, at its core, a hierarchical system. At the top of this system is what are known as "root servers". These servers are controlled by various organizations and are delegated authority by ICANN (Internet Corporation for Assigned Names and Numbers).

There are currently 13 root servers in operation. However, as there are an incredible number of names to resolve every minute, each of these servers is actually mirrored. The interesting thing about this set up is that each of the mirrors for a single root server share the same IP address. When requests are made for a certain root server, the request will be routed to the nearest mirror of that root server.

What do these root servers do? Root servers handle requests for information about Top-level domains. So if a request comes in for something a lower-level name server cannot resolve, a query is made to the root server for the domain.

The root servers won't actually know where the domain is hosted. They will, however, be able to direct the requester to the name servers that handle the specifically requested top-level domain.

So if a request for "www.wikipedia.org" is made to the root server, the root server will tell not find the result in its records. It will check its zone files for a listing that matches

"www.wikipedia.org". It will not find one.

It will instead find a record for the "org" TLD and give the requesting entity the address of the name server responsible for "org" addresses.

TLD Servers

The requester then sends a new request to the IP address (given to it by the root server) that is responsible for the top-level domain of the request.

So, to continue our example, it would send a request to the name server responsible for knowing about "org" domains to see if it knows where "www.wikipedia.org" is located.

Once again, the requester will look for "www.wikipdia.org" in its zone files. It will not find this record in its files.

However, it will find a record listing the IP address of the name server responsible for "wikipedia.org". This is getting much closer to the answer we want.

Domain-Level Name Servers

At this point, the requester has the IP address of the name server that is responsible for knowing the actual IP address of the resource. It sends a new request to the name server asking, once again, if it can resolve "www.wikipedia.org".

The name server checks its zone files and it finds that it has a zone file associated with "wikipedia.org". Inside of this file, there is a record for the "www" host. This record tells the IP address where this host is located. The name server returns the final answer to the requester.

What is a Resolving Name Server?

In the above scenario, we referred to a "requester". What is the requester in this situation?

In almost all cases, the requester will be what we call a "resolving name server". A resolving name server is one configured to ask other servers questions. It is basically an intermediary for a user which caches previous query results to improve speed and knows the addresses of the root servers to be able to "resolve" requests made for things it doesn't already know about.

Basically, a user will usually have a few resolving name servers configured on their computer system. The resolving name servers are usually provided by an ISP or other organizations. For instance Google provides resolving DNS servers that you can query. These can be either configured in your computer automatically or manually.

When you type a URL in the address bar of your browser, your computer first looks to see if it can find out locally where the resource is located. It checks the "hosts" file on the computer and a few other locations. It then sends the request to the resolving name server and waits back to receive the IP address of the resource.

The resolving name server then checks its cache for the answer. If it doesn't find it, it goes through the steps outlined above.

Resolving name servers basically compress the requesting process for the end user. The clients simply have to know to ask the resolving name servers where a resource is located and be confident that they will investigate and return the final answer.

Zone Files

We mentioned in the above process the idea of "zone files" and "records".

Zone files are the way that name servers store information about the domains they know about. Every domain that a name server knows about is stored in a zone file. Most requests coming to the average name server are not something that the server will have zone files for.

If it is configured to handle recursive queries, like a resolving name server, it will find out the answer and return it. Otherwise, it will tell the requesting party where to look next.

The more zone files that a name server has, the more requests it will be able to answer authoritatively.

A zone file describes a DNS "zone", which is basically a subset of the entire DNS naming system. It generally is used to configure just a single domain. It can contain a number of records which define where resources are for the domain in question.

The zone's `$ORIGIN` is a parameter equal to the zone's highest level of authority by default.

So if a zone file is used to configure the "example.com." domain, the `$ORIGIN` would be set to `example.com.`

This is either configured at the top of the zone file or it can be defined in the DNS server's configuration file that references the zone file. Either way, this parameter describes what the

zone is going to be authoritative for.

Similarly, the `$TTL` configures the "time to live" of the information it provides. It is basically a timer. A caching name server can use previously queried results to answer questions until the TTL value runs out.

Record Types

Within the zone file, we can have many different record types. We will go over some of the more common (or mandatory types) here.

SOA Records

The Start of Authority, or SOA, record is a mandatory record in all zone files. It must be the first real record in a file (although `$ORIGIN` or `$TTL` specifications may appear above). It is also one of the most complex to understand.

The start of authority record looks something like this:

```
domain.com.  IN SOA ns1.domain.com. admin.domain.com. (  
                                12083   ; serial number  
                                3h       ; refresh interval  
                                30m      ; retry interval  
                                3w       ; expiry period  
                                1h      ; negative TTL  
)
```

Let's explain what each part is for:

- **domain.com.:** This is the root of the zone. This specifies that the zone file is for the `domain.com.` domain. Often, you'll see this replaced with `@`, which is just a placeholder that substitutes the contents of the `$ORIGIN` variable we learned about above.
- **IN SOA:** The "IN" portion means internet (and will be present in many records). The SOA is the indicator that this is a Start of Authority record.
- **ns1.domain.com.:** This defines the primary master name server for this domain. Name servers can either be master or slaves, and if dynamic DNS is configured one server needs

to be a "primary master", which goes here. If you haven't configured dynamic DNS, then this is just one of your master name servers.

- **admin.domain.com.:** This is the email address of the administrator for this zone. The "@" is replaced with a dot in the email address. If the name portion of the email address normally has a dot in it, this is replaced with a "\" in this part (your.name@domain.com becomes your\name.domain.com).
- **12083:** This is the serial number for the zone file. Every time you edit a zone file, you must increment this number for the zone file to propagate correctly. Slave servers will check if the master server's serial number for a zone is larger than the one they have on their system. If it is, it requests the new zone file, if not, it continues serving the original file.
- **3h:** This is the refresh interval for the zone. This is the amount of time that the slave will wait before polling the master for zone file changes.
- **30m:** This is the retry interval for this zone. If the slave cannot connect to the master when the refresh period is up, it will wait this amount of time and retry to poll the master.
- **3w:** This is the expiry period. If a slave name server has not been able to contact the master for this amount of time, it no longer returns responses as an authoritative source for this zone.
- **1h:** This is the amount of time that the name server will cache a name error if it cannot find the requested name in this file.

A and AAAA Records

Both of these records map a host to an IP address. The "A" record is used to map a host to an IPv4 IP address, while "AAAA" records are used to map a host to an IPv6 address.

The general format of these records is this:

host	IN	A	IPv4_address
host	IN	AAAA	IPv6_address

So since our SOA record called out a primary master server at "ns1.domain.com", we would have to map this to an address to an IP address since "ns1.domain.com" is within the "domain.com" zone that this file is defining.

The record could look something like this:

```
ns1      IN  A      111.222.111.222
```

Notice that we don't have to give the full name. We can just give the host, without the FQDN and the DNS server will fill in the rest with the \$ORIGIN value. However, we could just as easily use the entire FQDN if we feel like being semantic:

```
ns1.domain.com.  IN  A      111.222.111.222
```

In most cases, this is where you'll define your web server as "www":

```
www      IN  A      222.222.222.222
```

We should also tell where the base domain resolves to. We can do this like this:

```
domain.com.  IN  A      222.222.222.222
```

We could have used the "@" to refer to the base domain instead:

```
@      IN  A      222.222.222.222
```

We also have the option of resolving anything that under this domain that is not defined explicitly to this server too. We can do this with the "*" wild card:

```
*      IN  A      222.222.222.222
```

All of these work just as well with AAAA records for IPv6 addresses.

CNAME Records

CNAME records define an alias for canonical name for your server (one defined by an A or AAAA record).

For instance, we could have an A name record defining the "server1" host and then use the "www" as an alias for this host:

```
server1    IN  A      111.111.111.111
www        IN  CNAME  server1
```

Be aware that these aliases come with some performance losses because they require an additional query to the server. Most of the time, the same result could be achieved by using additional A or AAAA records.

One case when a CNAME is recommended is to provide an alias for a resource outside of the current zone.

MX Records

MX records are used to define the mail exchanges that are used for the domain. This helps email messages arrive at your mail server correctly.

Unlike many other record types, mail records generally don't map a host to something, because they apply to the entire zone. As such, they usually look like this:

```
IN  MX  10  mail.domain.com.
```

Note that there is no host name at the beginning.

Also note that there is an extra number in there. This is the preference number that helps computers decide which server to send mail to if there are multiple mail servers defined. Lower numbers have a higher priority.

The MX record should generally point to a host defined by an A or AAAA record, and not one defined by a CNAME.

So, let's say that we have two mail servers. There would have to be records that look something like this:

```
      IN  MX  10  mail1.domain.com.
      IN  MX  50  mail2.domain.com.
mail1  IN  A      111.111.111.111
mail2  IN  A      222.222.222.222
```

In this example, the "mail1" host is the preferred email exchange server.

We could also write that like this:

```
      IN  MX  10  mail1
      IN  MX  50  mail2
mail1  IN  A      111.111.111.111
mail2  IN  A      222.222.222.222
```

NS Records

This record type defines the name servers that are used for this zone.

You may be wondering, "if the zone file resides on the name server, why does it need to reference itself?". Part of what makes DNS so successful is its multiple levels of caching. One reason for defining name servers within the zone file is that the zone file may be actually being served from a cached copy on another name server. There are other reasons for needing the name servers defined on the name server itself, but we won't go into that here.

Like the MX records, these are zone-wide parameters, so they do not take hosts either. In general, they look like this:

```
      IN  NS      ns1.domain.com.
      IN  NS      ns2.domain.com.
```

You should have at least two name servers defined in each zone file in order to operate correctly if there is a problem with one server. Most DNS server software considers a zone file to be invalid if there is only a single name server.

As always, include the mapping for the hosts with A or AAAA records:

```

      IN      NS      ns1.domain.com.
      IN      NS      ns2.domain.com.
ns1      IN      A      111.222.111.111
ns2      IN      A      123.211.111.233

```

There are quite a few other record types you can use, but these are probably the most common types that you will come across.

PTR Records

The PTR records are used to define a name associated with an IP address. PTR records are the inverse of an A or AAAA record. PTR records are unique in that they begin at the `.arpa` root and are delegated to the owners of the IP addresses. The Regional Internet Registries (RIRs) manage the IP address delegation to organizations and service providers. The Regional Internet Registries include APNIC, ARIN, RIPE NCC, LACNIC, and AFRINIC.

Here is an example of a PTR record for 111.222.333.444 would look like:

```
444.333.222.111.in-addr.arpa. 33692 IN PTR host.example.com.
```

This example of a PTR record for an IPv6 address shows the *nibble* format of the reverse of Google's IPv6 DNS Server 2001:4860:4860::8888.

```
8.8.8.8.0.0.0.0.0.0.0.0.0.0.0.0.6.8.4.0.6.8.4.1.0.0.2.ip6.arpa. 86400IN
```

The command line tool `dig` with the `-x` flag can be used to look up the reverse DNS name of an IP address.

Here is an example of a dig command. The `+short` is appended to reduce the output to the reverse DNS name.

```
$ dig -x 8.8.4.4 +short
```

The output for the dig command above will be the domain name in the PTR record for the IP address:

```
google-public-dns-b.google.com.
```

Servers on the Internet use PTR records to place domain names within log entries, make informed spam handling decisions, and display easy-to-read details about other devices.

Most commonly-used email servers will look up the PTR record of an IP address it receives email from. If the source IP address does not have a PTR record associated with it, the emails being sent may be treated as spam and rejected. It is not important that the FQDN in the PTR matches the domain name of the email being sent. What is important is that there is a valid PTR record with a corresponding and matching forward A record.

Normally network routers on the Internet are given PTR records that correspond with their physical location. For example you may see references to 'NYC' or 'CHI' for a router in New York City or Chicago. This is helpful when running a traceroute or MTR and reviewing the path Internet traffic is taking.

Most providers offering dedicated servers or VPS services will give customers the ability to set a PTR record for their IP address. **DigitalOcean will automatically assign the PTR record of any Droplet when the Droplet is named with a domain name.** The Droplet name is assigned during creation and can be edited later using the settings page of the Droplet control panel.

Note: It is important that the FQDN in the PTR record has a corresponding and matching forward A record. Example: 111.222.333.444 has a PTR of server.example.com and server.example.com is an A record that points to 111.222.333.444.

Conclusion

You should now have a pretty good grasp on how DNS works. While the general idea is relatively easy to grasp once you're familiar with the strategy, this is still something that can be

difficult for inexperienced administrators to put into practice.

For an overview check out [How To Set Up Domains](#) within the DigitalOcean Control Panel.

By: Justin Ellingwood

♡ Upvote (121)

📄 Subscribe

🔗 Share

Tutorial Series

An Introduction to Managing DNS

DNS, or the domain name system, is an essential component of modern internet communication. It allows us to reference computers by names instead of IP addresses. In this series, we will cover the basic ideas behind DNS so that you feel comfortable working with it. Afterwards, we will walk through various ways that you can gain greater control over your domains and DNS resolution.

- 1 An Introduction to DNS Terminology, Components, and Concepts February 18, 2014
- 2 A Comparison of DNS Server Types: How To Choose the Right DNS Configuration June 30, 2014
- 3 How To Set Up a Host Name with DigitalOcean August 28, 2012
- 4 How To Configure Bind as a Caching or Forwarding DNS Server on Ubuntu 14.04 June 25, 2014
- 5 How To Configure Bind as an Authoritative-Only DNS Server on Ubuntu 14.04 June 27, 2014
- 6 How To Configure BIND as a Private Network DNS Server on Ubuntu 14.04 August 12, 2014
- 7 How To Use NSD, an Authoritative-Only DNS Server, on Ubuntu 14.04 July 3, 2014

SCROLL TO TOP

Give back to open source this October

Celebrate open source software by contributing to GitHub-hosted open source projects for the chance of getting your own limited-edition Hacktoberfest T-shirt.

[Learn more about Hacktoberfest](#)

Related Tutorials

DNS Tips and Tricks

[How to Create an Intranet with OpenVPN on Ubuntu 16.04](#)

[How To Configure BIND as a Private Network DNS Server on Ubuntu 16.04](#)

[How To Configure Bind as a Caching or Forwarding DNS Server on Ubuntu 16.04](#)

[Blueprint: How Ghost Migrated From Dedicated Servers to DigitalOcean](#)



9 Comments

[SCROLL TO TOP](#)

Leave a comment...

[Log In to Comment](#)



 [robin.joseph](#) *March 1, 2014*
 0 Technically, 111.222.333.444 isn't a valid IPv4 address since each grouping of numbers is 8 bits. Each has a range of 0-255.



 [luissquall](#) *March 3, 2014*
 0 Great article!

Just a minor fix, trailing dot is missing in the CNAME RDATA of MX & NS Records examples, e.g.

IN MX 10 mail1.domain.com.

IN NS ns1.domain.com.

 [jellingwood](#) MOD *March 3, 2014*
 0 Thank you both for the feedback. You are both correct, and I've updated the article accordingly. Thanks!

 [tzickell](#) *July 8, 2014*
 0 Without question the best DNS available. Currently, the pricing has changed actually to make it more accessible. You can get a domain on the full a anycast DynECT light platform starting at five dollars a month
you are

[SCROLL TO TOP](#)

I am impressed with the unicast DNS system you have set up. However honestly there are better options than this. Of course, it depends on what you want to spend and what type of uptime guarantee you need.

If you are hosting a large group of clients managed DNS by third-party is the way to go 9/10 times including adding a second managed DNS server to complete the redundancy.
DynECT is the world's best DNS if you ask me. I mix it EdgeCast route DNS both are able to resolve GEO-IP and make a blazing fast extremely redundant setup.

Other very fast options are to use DNS made easy if you need to secure it run AWS route 53 as the secondary DNS.

To give you an idea DNS made easy is considered one of the fastest DNS service in the world up there with DynECT, edge cast & ultraDNS.

If you have a lot of clients spend USD25 a year on 10 domains and 30 million queries per month if you need a larger amount USD50 a year gets you 25 domains 50 million queries per month with failover.

AWS route 53 is unbelievably capable for its price. USD.50 a domain a dollar per 10 million queries. (It is fast and reliable but not as fast and reliable as DNS made easy)

Last but not least Akamai's DNS is sold through Zerego at USD25 a month for 25 domains.

DynECT light has dropped dramatically in price getting you a single domain with just under 1 million queries per second at roughly USD10 a month.

DNS is tracked continuously by SolveDNS you can type a name server into the top right search box and it will give you the stats on it and you can also look at all any cast managed DNS hosting companies speeds.

Last but not least remember to lower your TTL (time to live) the more queries you will use for the same amount of traffic. I know people that set their TTL for 60 min. (that is a good time by the way enough to cache and enough to change quickly) and a site with 20,000 unique visitors will be able to deal with under 1 million queries per month.

I should tell you that DynECT enterprise or just what they call DynECT DNS is about USD200 a month with 1 million queries a month you can lower the price and discuss the queries which are much less than you would think because it is an enterprise plan. When all is said and done if you want the ability to route CDN's through Geo-IP you are looking at somewhere between \$1,000 or \$2,000 month.

With UltraDNS you can purchase their SMB offering for USD30 a month with 2 million queries and 50 domains.

For enterprise UltraDNS the price begins at USD1500 a month.

Edge Cast route is not as sophisticated as DynECT or UltraDNS however it offers Geo-IP routing, fail over and load balancing for under USD100 a month.

The fastest are DynECT & edge cast followed by DNS made easy they are all in the under 10 ms club which is something that cannot be said for very many DNS hosts. No matter what system you can get something much faster for USD25 a year

Go to solveDNS.com to see statistics and run speed tests on any name server you want.

Remember CloudFlare offers anycast DNS for free however it has terrible uptime and is not very fast most of the time because the entire anycast network absorbs all the attacks. Your better off using a system described above and having no issues with your DNS. the reason it is fast and the site I have referenced is because they test for only 2 to 3 min. a month not of the other services use their pops as a punching bag for free Subscribers and paid subscribers wanting the DDOS protection.

PS if you want to have a remarkably powerful firewall for very little money cloudproxy.sucuri.net is an amazing WAF and makes an excellent match with digital ocean.



[trycatch9264](#) July 17, 2014

Simple and Clear, easy to understand, Great article!
Thank you Justin!



[vincent.osullivan](#) July 17, 2014

I think you may have meant "Pedantic" instead of "Semantic" above?
Great article!



[sashashakun](#) April 15, 2015

Hello! Thank you for the article, but as I see you may have a problem with FB sharing:
<http://tinypic.com/r/ilb0qw/8>
Sorry for language)
Translation of the pic's text:
"The specified URL-address configuration is not allowed Applications: One or more of the URL-address is blocked application settings. The address must match the address of the website or canvas, or domain should be a subdomain of a domain application."



[PabloAssembler](#) May 2, 2015

Hi. Great article! In the **Zone File** explanation, shouldn't it be "...about the domains they have authority over" or "... are authoritative" instead of "...about the domains they know about" ?



[rami2](#) September 15, 2015

Hi, in the process off learning all this from scratch i wrote a tutorial for setting up multiple sites which people may find helpful, some things are missing from the digital ocean community are included here.

Setting-up-3-wordpress-sites-on-Apache-server-and-Ubuntu-14-04

<https://sunnahmuakadasupplementary.files.wordpress.com/2015/09/setting-up-3-wordpress-sites-on-apache-server-and-ubuntu-14-04.pdf>

SCROLL TO TOP



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2016 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#) 

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Get Paid to Write](#)

[SCROLL TO TOP](#)