TUTORIAL

OpenSSL Essentials: Working with SSL Certificates, Private Keys and CSRs

FAQ Security

By Mitchell Anicas
Last Validated on October 13, 2021 • Originally Published on September 12, 2014
1.6m

Introduction

OpenSSL is a versatile command line tool that can be used for a large variety of tasks related to Public Key Infrastructure (PKI) and HTTPS (HTTP over TLS). This cheat sheet style guide provides a quick reference to OpenSSL commands that are useful in common, everyday scenarios. This includes OpenSSL examples for generating private keys, certificate signing requests, and certificate format conversion. It does not cover all of the uses of OpenSSL.

How to Use This Guide:

- If you are not familiar with certificate signing requests (CSRs), read the first section
- Aside from the first section, this guide is in a cheat sheet format: a list of self-contained command line snippets
- Jump to any section that is relevant to the task you are trying to complete (Hint: use the *Contents* menu or your browser's *Find* function)
- Most of the commands are one-liners that have been expanded to multiple lines (using the `\` symbol) for clarity

About Certificate Signing Requests (CSRs)

If you would like to obtain an SSL certificate from a commercial certificate authority (CA), you must generate a certificate signing request (CSR). A CSR consists mainly of the public key of a key pair, and some additional information. Both of these components are inserted into the certificate when it is signed.

Whenever you generate a CSR, you will be prompted to provide information regarding the certificate. This information is known as a Distinguished Name (DN). An important field in the DN is the **Common Name** (CN), which should be the exact Fully Qualified Domain Name (FQDN) of the host that you intend to use the certificate with. It is also possible to skip the interactive prompts when creating a CSR by passing the information via command line or from a file.

The other items in a DN provide additional information about your business or organization. If you are purchasing an SSL certificate from a certificate authority, it is often required that these additional fields, such as "Organization", accurately reflect your organization's details.

Here is an example of what the CSR information prompt will look like:

```
---
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:New York
Locality Name (eg, city) []:Brooklyn
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Example Brooklyn Company
Organizational Unit Name (eg, section) []:Technology Division
Common Name (e.g. server FQDN or YOUR name) []:examplebrooklyn.com
Email Address []:
```

If you want to non-interactively answer the CSR information prompt, you can do so by adding the `-subj` option to any OpenSSL commands that request CSR information. Here is an example of the option, using the same information displayed in the code block above:

```
-subj "/C=US/ST=New York/L=Brooklyn/O=Example Brooklyn Company/CN=examplebrooklyn.com"
```

Now that you understand CSRs, feel free to jump around to whichever section of this guide covers your OpenSSL needs.

Generating CSRs

This section covers OpenSSL commands that are related to generating CSRs (and private keys, if they do not already exist). CSRs can be used to request SSL certificates from a certificate authority.

Keep in mind that you may add the CSR information non-interactively with the `-subj` option, mentioned in the previous section.

Generate a Private Key and a CSR

Use this method if you want to use HTTPS (HTTP over TLS) to secure your Apache HTTP or Nginx web server, and you want to use a Certificate Authority (CA) to issue the SSL certificate. The CSR that is generated can be sent to a CA to request the issuance of a CA-signed SSL certificate. If your CA supports SHA-2, add the `-sha256` option to sign the CSR with SHA-2.

This command creates a 2048-bit private key (`domain.key`) and a CSR (`domain.csr`) from scratch:

```
$ openssl req \
$        -newkey rsa:2048 -nodes -keyout domain.key \
$        -out domain.csr
```

Answer the CSR information prompt to complete the process.

The `-newkey rsa:2048` option specifies that the key should be 2048-bit, generated using the RSA algorithm. The `-nodes` option specifies that the private key should *not* be encrypted with a pass phrase. The `-new` option, which is not included here but implied, indicates that a CSR is being generated.

Generate a CSR from an Existing Private Key

Use this method if you already have a private key that you would like to use to request a certificate from a CA.

This command creates a new CSR (`domain.csr`) based on an existing private key (`domain.key`):

```
$ openssl req \
$        -key domain.key \
$        -new -out domain.csr
```

Answer the CSR information prompt to complete the process.

The `-key` option specifies an existing private key (`domain.key`) that will be used to generate a new CSR. The `-new` option indicates that a CSR is being generated.

Generate a CSR from an Existing Certificate and Private Key

Use this method if you want to renew an existing certificate but you or your CA do not have the original CSR for some reason. It basically saves you the trouble of re-entering the CSR information, as it extracts that information from the existing certificate.

This command creates a new CSR (`domain.csr`) based on an existing certificate (`domain.crt`) and private key (`domain.key`):

```
$ openssl x509 \
$        -in domain.crt \
$        -signkey domain.key \
$        -x509toreq -out domain.csr
```

The `-x509toreq` option specifies that you are using an X509 certificate to make a CSR.

Generating SSL Certificates

If you would like to use an SSL certificate to secure a service but you do not require a CA-signed certificate, a valid (and free) solution is to sign your own certificates.

A common type of certificate that you can issue yourself is a *self-signed certificate*. A self-signed certificate is a certificate that is signed with its own private key. Self-signed certificates can be used to encrypt data just as well as CA-signed certificates, but your users will be displayed a warning that says that the certificate is not trusted by their computer or browser. Therefore, self-signed certificates should only be used if you do not need to prove your service's identity to its users (e.g. non-production or non-public servers).

This section covers OpenSSL commands that are related to generating self-signed certificates.

Generate a Self-Signed Certificate

Use this method if you want to use HTTPS (HTTP over TLS) to secure your Apache HTTP or Nginx web server, and you do not require that your certificate is signed by a CA.

This command creates a 2048-bit private key (`domain.key`) and a self-signed certificate (`domain.crt`) from scratch:

```
$ openssl req \
$        -newkey rsa:2048 -nodes -keyout domain.key \
$        -x509 -days 365 -out domain.crt
```

Answer the CSR information prompt to complete the process.

The `-x509` option tells `req` to create a self-signed certificate. The `-days 365` option specifies that the certificate will be valid for 365 days. A temporary CSR is generated to gather information to associate with the certificate.

Generate a Self-Signed Certificate from an Existing Private Key

Use this method if you already have a private key that you would like to generate a self-signed certificate with it.

This command creates a self-signed certificate (`domain.crt`) from an existing private key (`domain.key`):

```
$ openssl req \
$        -key domain.key \
$        -new \
$        -x509 -days 365 -out domain.crt
```

Answer the CSR information prompt to complete the process.

The `-x509` option tells `req` to create a self-signed certificate. The `-days 365` option specifies that the certificate will be valid for 365 days. The `-new` option enables the CSR information prompt.

Generate a Self-Signed Certificate from an Existing Private Key and CSR

Use this method if you already have a private key and CSR, and you want to generate a self-signed certificate with them.

This command creates a self-signed certificate (`domain.crt`) from an existing private key (`domain.key`) and (`domain.csr`):

```
$ openssl x509 \
$        -signkey domain.key \
$        -in domain.csr \
$        -req -days 365 -out domain.crt
```

The `-days 365` option specifies that the certificate will be valid for 365 days.

View Certificates

Certificate and CSR files are encoded in PEM format, which is not readily human-readable.

This section covers OpenSSL commands that will output the actual entries of PEM-encoded files.

View CSR Entries

This command allows you to view and verify the contents of a CSR (`domain.csr`) in plain text:

```
$ openssl req -text -noout -verify -in domain.csr
```

View Certificate Entries

This command allows you to view the contents of a certificate (`domain.crt`) in plain text:

```
$ openssl x509 -text -noout -in domain.crt
```

Verify a Certificate was Signed by a CA

Use this command to verify that a certificate (`domain.crt`) was signed by a specific CA certificate (`ca.crt`):

```
$ openssl verify -verbose -CAFile ca.crt domain.crt
```

Private Keys

This section covers OpenSSL commands that are specific to creating and verifying private keys.

Create a Private Key

Use this command to create a password-protected, 2048-bit private key (`domain.key`):

```
$ openssl genrsa -des3 -out domain.key 2048
```

Enter a password when prompted to complete the process.

Verify a Private Key

Use this command to check that a private key (`domain.key`) is a valid key:

```
$ openssl rsa -check -in domain.key
```

If your private key is encrypted, you will be prompted for its pass phrase. Upon success, the unencrypted key will be output on the terminal.

Verify a Private Key Matches a Certificate and CSR

Use these commands to verify if a private key (`domain.key`) matches a certificate (`domain.crt`) and CSR (`domain.csr`):

```
$ openssl rsa -noout -modulus -in domain.key | openssl md5
$ openssl x509 -noout -modulus -in domain.crt | openssl md5
$ openssl req -noout -modulus -in domain.csr | openssl md5
```

If the output of each command is identical there is an extremely high probability that the private key, certificate, and CSR are related.

Encrypt a Private Key

This takes an unencrypted private key (`unencrypted.key`) and outputs an encrypted version of it (`encrypted.key`):

```
$ openssl rsa -des3 \
$        -in unencrypted.key \
$        -out encrypted.key
```

Enter your desired pass phrase, to encrypt the private key with.

Decrypt a Private Key

This takes an encrypted private key (`encrypted.key`) and outputs a decrypted version of it (`decrypted.key`):

```
$ openssl rsa \
$        -in encrypted.key \
$        -out decrypted.key
```

Enter the pass phrase for the encrypted key when prompted.

Convert Certificate Formats

All of the certificates that we have been working with have been X.509 certificates that are ASCII PEM encoded. There are a variety of other certificate encoding and container types; some applications prefer certain formats over others. Also, many of these formats can contain multiple items, such as a private key, certificate, and CA certificate, in a single file.

OpenSSL can be used to convert certificates to and from a large variety of these formats. This section will cover a some of the possible conversions.

Convert PEM to DER

Use this command if you want to convert a PEM-encoded certificate (`domain.crt`) to a DER-encoded certificate (`domain.der`), a binary format:

```
$ openssl x509 \
$        -in domain.crt \
$        -outform der -out domain.der
```

The DER format is typically used with Java.

Convert DER to PEM

Use this command if you want to convert a DER-encoded certificate (`domain.der`) to a PEM-encoded certificate (`domain.crt`):

```
$ openssl x509 \
$        -inform der -in domain.der \
$        -out domain.crt
```

Convert PEM to PKCS7

Use this command if you want to add PEM certificates (`domain.crt` and `ca-chain.crt`) to a PKCS7 file (`domain.p7b`):

```
$ openssl crl2pkcs7 -nocrl \
$        -certfile domain.crt \
$        -certfile ca-chain.crt \
$        -out domain.p7b
```

Note that you can use one or more `-certfile` options to specify which certificates to add to the PKCS7 file.

PKCS7 files, also known as P7B, are typically used in Java Keystores and Microsoft IIS (Windows). They are ASCII files which can contain certificates and CA certificates.

Convert PKCS7 to PEM

Use this command if you want to convert a PKCS7 file (`domain.p7b`) to a PEM file:

```
$ openssl pkcs7 \
$        -in domain.p7b \
$        -print_certs -out domain.crt
```

Note that if your PKCS7 file has multiple items in it (e.g. a certificate and a CA intermediate certificate), the PEM file that is created will contain all of the items in it.

Convert PEM to PKCS12

Use this command if you want to take a private key (`domain.key`) and a certificate (`domain.crt`), and combine them into a PKCS12 file (`domain.pfx`):

```
$ openssl pkcs12 \
$        -inkey domain.key \
$        -in domain.crt \
$        -export -out domain.pfx
```

You will be prompted for export passwords, which you may leave blank. Note that you may add a chain of certificates to the PKCS12 file by concatenating the certificates together in a single PEM file (`domain.crt`) in this case.

PKCS12 files, also known as PFX files, are typically used for importing and exporting certificate chains in Microsoft IIS (Windows).

Convert PKCS12 to PEM

Use this command if you want to convert a PKCS12 file (`domain.pfx`) and convert it to PEM format (`domain.combined.crt`):

```
openssl pkcs12 \
       -in domain.pfx \
       -nodes -out domain.combined.crt
```

Note that if your PKCS12 file has multiple items in it (e.g. a certificate and private key), the PEM file that is created will contain all of the items in it.

OpenSSL Version

The `openssl version` command can be used to check which version you are running. The version of OpenSSL that you are running, and the options it was compiled with affect the capabilities (and sometimes the command line options) that are available to you.

The following command displays the OpenSSL version that you are running, and all of the options that it was compiled with:

```
openssl version -a
```

This guide was written using an OpenSSL binary with the following details (the output of the previous command):

```
OpenSSL 1.1.1  11 Sep 2018
built on: Mon Aug 23 17:02:39 2021 UTC
platform: debian-amd64
options:  bn(64,64) rc4(16x,int) des(int) blowfish(ptr)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -Wa,--noexecstack -g -O2 -fdebug-
OPENSSLDIR: "/usr/lib/ssl"
ENGINESDIR: "/usr/lib/x86_64-linux-gnu/engines-1.1"
Seeding source: os-specific
```

Conclusion

That should cover how most people use OpenSSL to work with SSL certs. It has many other uses that were not covered here, so feel free to ask or suggest other uses in the comments.

If you are having issues with any of the commands, be sure to comment (and include your OpenSSL version output).

Was this helpful? Yes No 27

Report an issue

About the authors

Mitchell Anicas

Software Engineer @ DigitalOcean. Former Señor Technical Writer (I no longer update articles or respond to comments).

Still looking for an answer?

Ask a question

Search for more help

Comments

27 Comments

Leave a comment...

Sign In to Comment

schumann September 26, 2014

2 Great summary, was recently looking for exactly something like that. Thank you for the write up.

I recently stumbled across https://shaaaaaaaaaaaaa.com/ summarizing the soon to come retirement of SHA-1.

It may be of use also for this tutorial to add the option

```
-sha256
```

to create the CSR.

I did that for a recent request from StartSSL and they offered a certificate accommodating the more secure requirements (passing also A+ on ssllabs.com)

Since you especially describe how to generate CSR for obtaining a certificate, it may be worth adding the option in order to be more future proof.

Best regards
Sebastian

Reply Report

manicas September 29, 2014
1
Added. Thanks!

Reply Report

gauravkhandelwa November 5, 2014
0
Great Summary. I want to know how can I add a key usage extension to a certificate. Specifically to make it act as a local-CA to sign other certificates?

Reply Report

shlepshitz May 12, 2015
0
I need to copy paste the Certificate Signing Request (CSR) how do i get a hold of it?

Reply Report

manicas May 12, 2015
0
One way to view the CSR is to type `cat domain.csr`. This will print the contents of the file to the screen (which you then can copy).

Make sure you are in the same directory that you generated the CSR in.

Reply Report

shlepshitz May 12, 2015
0
[deleted]

Reply Report

shlepshitz May 12, 2015
0
thanks i see it now but cant do Ctrl+c

Reply Report

tiangolo May 12, 2015
1
You are probably using a terminal, Putty, Cygwin or something similar. In those programs you commonly can `Ctrl + Insert` to copy and `Shift + Insert` to paste.

Reply Report

manicas May 13, 2015
0
In addition to what @tiangolo mentioned, you can probably right-click and select copy from the menu.

Reply Report

rahoolm July 24, 2015
0
Too good. I was very happy after going through the articles. It helped me a lot. Especially the verification part.

Reply Report

daverick August 20, 2015
0
Someone else created a csr request, and we got the final mail from CA which gave the X509 Certificates and intermediates only certificates. Now I am not sure that whether I am supposed to generate another private key based on the certificate, it would be great if you can explain about this part.

Thanks

Reply Report

manicas August 20, 2015
0
The CSR is created from the private key. Whoever created the CSR should also have the private key.

Reply Report

luissquall October 15, 2015

0 The command provided in section "Generate a CSR from an Existing Certificate and Private Key" generates a file with the plaintext csr and encoded version:

```
Certificate Request:
    Data:
...
-----BEGIN CERTIFICATE REQUEST-----
MIICozCCAYsCAQAwXjEhMB8GA1UECxMYRG9tYW1uIENvbnRyb2wgVmFsaWRhdGVk
```

Is there any option to output only the encoded version?

Reply Report

fazalabbas May 18, 2016

0 Great article. Learnt a lot. Thanks for sharing.

Abbas

Reply Report

Anita May 31, 2016

1 Mitchell - Fantastic post! Just one slight correction:

```
openssl verify -verbose -CAFile ca.crt domain.crt
```

The option uses a lowercase "f", as in:

```
-CAfile
```

Reply Report

KS7000 September 30, 2019

0 **I saw that too!** Maybe here we can use like RFC «*Errata exists*» but like `my way`, at top: «*Errata existed, see bottom*» (I mean, text fixed with a numbered superscript and the last-last-last section for explain every errata, if exists).

That "little" change will need consensus between moderators and after write the code (and remember, don't deploy CSM's code on Friday, ja, ja, ja)

P.S.: I have read and translated into Spanish (with some tutorials have learned so much, like this) and *erratas* are very-little-few! Congratulations for a good job!

Reply Report

anujaggarwal June 22, 2016

0 Thanks for this wonderful article, DO has always been of great article.

I am facing an issue with my SSL certificate installation, if you could help me.

I bought a Rapid SSL and used the below command to generate the .csr and .key files:

```
    sudo openssl req -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/server.ke
```

I answered all questions which this command asked. I then provided the .csr to name.com and successfully generated the server/intermediate certificates. I then followed the steps mentioned at https://knowledge.rapidssl.com/support/ssl-certificate-support/index?page=content&actp=CROSSLINK&id=SO17664 and installed this certificate at my nginx server. I was able to open the HTTPS version of my site as well.

Now, to try something else, I run the command (sudo openssl req…) again with different answers this time and regenerated a new server.key file. Unfortunately, I didn't save the first server.key file. Post modification of nginx .conf file, when I tried to restart the server, I got the below error:

```
nginx: [emerg] SSL_CTX_use_PrivateKey_file("/etc/nginx/ssl/server.key") failed (SSL: er
```

Seeing this error, I realized I have overwritten the server.key file. I tried to generate the key again with same answers I gave for the first time but still the key mismatch error is coming.

My nginx server is still running and I am able to access the HTTPS version of the site but my life is in trouble without the private key. I have gone through the below links but still stuck:

1. http://stackoverflow.com/questions/26191463/ssl-error0b080074×509-certificate-routinesx509-check-private-keykey-values
2. http://serverfault.com/questions/408112/nginx-ssl-certificate-issue-key-values-mismatch

I confirmed by running the below commands that my certificate (issued by name.com) and private keys don't match:

```
openssl x509 -noout -modulus -in server_orig.cert | openssl md5
openssl rsa -noout -modulus -in server.key | openssl md5
```

Is there anything which I can do to find out the private key since Nginx is still up and running? In case not, should I get the certificate re-issued by Geotrust?

Any help would be deeply appreciated.

Reply Report

volch October 6, 2016

0 Thanks!

It will be very useful to explain creation of self-signed local CA pairs, signing CSR and install this CA certt on clients.

Reply Report

mckennatim January 19, 2017

0 Normally one would want the baddest certifcate you can get, un-deciferable cifers, un-stealable keys …

I have a different need, a dumbed down tls1.1 small key/cipher/cert that will work on esp8266's, these tiny, wifi enabled, mqtt protoocol running, $2, iot devices. OK so I made a 512byte private key. Now I need to make a cert and a sha-1 thumbprint that will work with small memory devices using tls1.1, TLS*RSA*WITH*AES*128*CBC*SHA or RC4-MD5 ciphers.

Any ideas on an oppenssl command to get that?

Reply Report

sgarg31 December 11, 2017

0 Thanks DO for the great article. Can you explain the command to create the intermediate cert with the csr and private key

Reply Report

aamir February 16, 2018

1
```
openssl verify -verbose -CAFile ca.crt domain.crt
```

Should be

```
openssl verify -verbose -CAfile ca.crt domain.crt
```

Reply Report

soomro March 15, 2018

0 Great and concise guide. Saved a lot of my time, thanks for sharing!

Reply Report

ChAshok April 20, 2018

0 Great article. For https ,I had created client private key and certificate key. When i had tried connect the https server below logs were coming. But finally able to connect the server(Taking more time).

Verify requested for (Depth 2):

ceThis certificate has no flags

Verify requested for (Depth 1):

ceThis certificate has no flags

Verify requested for (Depth 0):

ceThis certificate has no flags

connected to server
……………………………………………
I had used "openssl verify -verbose -CAFile ca.crt domain.crt" for to create the client certificate and "openssl genrsa -des3 -out domain.key 2048 " for to create privare key.

Could you please tell me if I missed something in my configuration of SSL?

Am i doing wrong?? any suggestions ??please

Reply Report

nparthibann September 3, 2018

0 I have created a sample project using python to create and manage SSL certificates, if anyone interested can have a look at it. https://github.com/parthibann/shield

Reply Report

NutzAz July 5, 2019

0 How to enter into terminal:

openssl rsa -des3 \
-in unencrypted.key \
-out encrypted.key

Total noob question sorry:

I've followed through and been able to create my rsa domain.key, domain.csr and domain.crt THANK YOU SO MUCH! :)

I'm using windows terminal to ssh into my Ubuntu droplet.

Once you started using multi-line commands which are indented I haven't been able to run these. Can I have your advice please.

I've tried copying and pasting, typing, and I read the comment below regarding Ctrl+ins and Shift+ins from community/users/tiangolo with no success. I've tried stringing your command lines together into one line but I can't get a result. I can't encrypt my private.key for the same reason.

Thank you all in advance,
NutAz

Reply Report

NutzAz July 5, 2019

1 SOLVED
Odly, I just reset, ssh logged back in and it works fine just copying and right click pasting!
Thanks again:)

Reply Report

carlusia17 November 24, 2019

0 Hello, great article! I just want to ask you, maybe you know or you can give me a hint or suggestion. How can I create a CSR by passing the information from a json file, not with -subj? Thank you for your time.

Reply Report

starrychloe January 16, 2020

0 You can also create your own certificate authority and sign your own certificates, then trust your own certificate authority so you don't get warnings.

https://www.mirthcorp.com/community/forums/showpost.php?p=22124&postcount=4

Reply Report

GET OUR BIWEEKLY NEWSLETTER

Sign up for Infrastructure as a Newsletter.

HOLLIE'S HUB FOR GOOD

Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

BECOME A CONTRIBUTOR

You get paid; we donate to tech nonprofits.

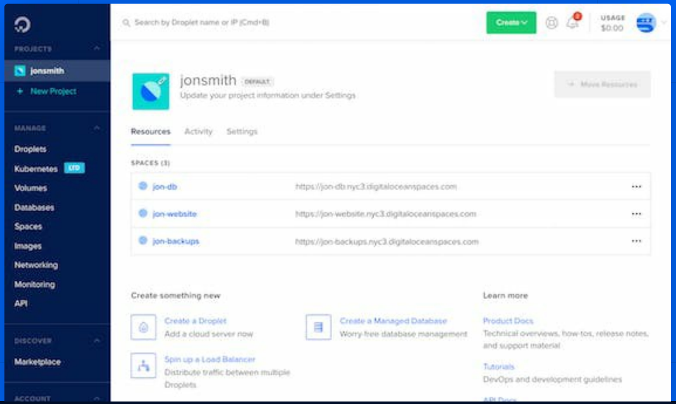
Featured on Community Kubernetes Course Learn Python 3 Machine Learning in Python Getting started with Go Intro to Kubernetes

DigitalOcean Products Virtual Machines Managed Databases Managed Kubernetes Block Storage Object Storage Marketplace VPC Load Balancers

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow – whether you're running one virtual machine or ten thousand.

Learn More



PROJECTS
jonsmith
New Project
MANAGE
Droplets
Kubernetes
Volumes
Databases
Spaces
Images
Networking
Monitoring
API
DISCOVER
Marketplace
ACCOUNT
Search by Droplet name or IP (Cmd+B)
Create
USAGE $0.00
jonsmith
Update your project information under Settings
Move Resources
Resources
Activity
Settings
SPACES (3)
jon-db
https://jon-db.nyc3.digitaloceanspaces.com
jon-website
https://jon-website.nyc3.digitaloceanspaces.com
jon-backups
https://jon-backups.nyc3.digitaloceanspaces.com
Create something new
Create a Droplet
Add a cloud server now
Create a Managed Database
Worry-free database management
Spin up a Load Balancer
Distribute traffic between multiple Droplets
Learn more
Product Docs
Technical overviews, how-tos, release notes, and support material
Tutorials
DevOps and development guidelines

Company	Products	Community	Contact
About	Pricing	Tutorials	Get Support
Leadership	Products Overview	Q&A	Trouble Signing In?
Blog	Droplets	Tools and Integrations	Sales
Careers	Kubernetes	Tags	Report Abuse
Partners	Managed Databases	Product Ideas	System Status
Referral Program	Spaces	Write for DigitalOcean	
Press	Marketplace	Presentation Grants	
Legal	Load Balancers	Hatch Startup Program	
Security & Trust Center	Block Storage	Shop Swag	
	API Documentation	Research Program	
	Documentation	Open Source	
	Release Notes	Code of Conduct	

Sign up for our newsletter

Get the latest tutorials on SysAdmin and open source topics.

Enter your email address

Sign Up

SCROLL TO TOP