serverfault

How to set environment variable in systemd service?

Asked 10 years, 2 months ago Modified 1 month ago Viewed 518k times

322

I have an Arch Linux system with systemd and I've created my own service. The configuration service at `/etc/systemd/system/myservice.service` looks like this:

```
[Unit]
Description=My Daemon

[Service]
ExecStart=/bin/myforegroundcmd

[Install]
WantedBy=multi-user.target
```

Now I want to have an environment variable set for the `/bin/myforegroundcmd` . How do I do that?

systemd

Share Improve this question

Follow

edited Apr 8, 2020 at 5:25

Braiam

622 4 23

asked Aug 1, 2012 at 19:43

lfagundes

3,323 2 13 6

6 Answers

Sorted by: Highest score (default)

420

Times change and so do best practices.

The *current* best way to do this is to run `systemctl edit myservice`, which will create an override file for you or let you edit an existing one.

In normal installations this will create a directory `/etc/systemd/system/myservice.service.d`, and inside that directory create a file whose name ends in `.conf` (typically, `override.conf`), and in this file you can add to or override any part of the unit shipped by the distribution.

For instance, in a file `/etc/systemd/system/myservice.service.d/myenv.conf`:

```
[Service]
Environment="SECRET=pGNqduRFkB4K9C2vijOmUDa2kPtUhArN"
Environment="ANOTHER_SECRET=JP8YLOc2bsNlrGuD6LVTq7L36obpjzxd"
```

Also note that if the directory exists and is empty, your service will be disabled! If you don't intend to put something in the directory, ensure that it does not exist.

For reference, the old way was:

The recommended way to do this is to create a file `/etc/sysconfig/myservice` which contains your variables, and then load them with `EnvironmentFile`.

For complete details, see Fedora's documentation on how to write a systemd script.

Share Improve this answer

Follow

edited Feb 27, 2019 at 12:52

Mikolasan

107 5

answered Aug 1, 2012 at 20:07

Michael Hampton

237k 42 481 941

4 I guess the `sysconfig` path is specific to Fedora but the question is about Arch Linux. The answer by paluh is more interesting I think – Ludovic Kuty Apr 27, 2013 at 8:49

1 `/etc/sysconfig` is Fedora-specific. AFAIR Arch Linux was pushing for having the config files somewhere package-specific rather in `/etc` rather than that Fedora-specific location. Like `/etc/myservice.conf`, though using extra file doesn't seem the right way here. – Michał Górny Apr 23, 2014 at 7:13

6 No, no, no. /etc/sysconfig is not recomended. It is discouraged, along with /etc/default/* from debian, because they are pointless, and the names are meaningless and make sense only for backwards compatibility reasons (all of /etc is about configuration of the system, not just /etc/sysconfig, and /etc/defaults is for overrides, not the defaults). Just put the definitions directly in the unit file, or if it is not possible, in an enviornment file that has a package specific location (like Michał's comment suggests). – zbyszek Oct 4, 2014 at 18:41

3 @MichaelHampton Could you please add documentation link for "current best way"? – jb. Dec 31, 2015 at 13:26

5 Don't use `Environment=` to pass secrets like passwords. See my answer for details. – Don Kirkby May 4, 2018 at 0:30

117

The answer depends on whether the variable is supposed to be constant (that is, not supposed to be modified by user getting the unit) or variable (supposed to be set by the user).

Since it's your local unit, the boundary is quite blurry and either way would work. However, if you started to distribute it and it would end up in `/usr/lib/systemd/system`, this would become important.

Constant value

If the value doesn't need to change per instance, the preferred way would be to place it as `Environment=`, directly in the unit file:

```
[Unit]
Description=My Daemon

[Service]
Environment="FOO=bar baz"
ExecStart=/bin/myforegroundcmd

[Install]
WantedBy=multi-user.target
```

The advantage of that is that the variable is kept in a single file with the unit. Therefore, the unit file is easier to move between systems.

Variable value

However, the above solution doesn't work well when sysadmin is supposed to change the value of the environment variable locally. More specifically, the new value would need to be set every time the unit file is updated.

For this case, an extra file is to be used. How — usually depends on the distribution policy.

One particularly interesting solution is to use `/etc/systemd/system/myservice.service.d` directory. Unlike other solutions, this directory is supported by systemd itself and therefore comes with no distribution-specific paths.

In this case, you place a file like `/etc/systemd/system/myservice.service.d/local.conf` that adds the missing parts of unit file:

```
[Service]
Environment="FOO=bar baz"
```

Afterwards, systemd merges the two files when starting the service (remember to `systemctl daemon-reload` after changing either of them). And since this path is used directly by systemd, you don't use `EnvironmentFile=` for this.

If the value is supposed to be changed only on some of the affected systems, you may combine both solutions, providing a default directly in the unit and a local override in the other file.

Share Improve this answer

Follow

edited Oct 16, 2018 at 18:12

The Guy with The Hat

105 5

answered Apr 23, 2014 at 7:48

Michał Górny

1,420 1 10 7

1 `systemctl daemon-reload` is the command to reload systemd – Dmitry Buzolin Apr 22, 2018 at 23:57

3 `EnvironmentFile=` is better when the values are secrets like passwords. See my answer for details. – Don Kirkby May 4, 2018 at 0:31

Still helpful 7+ years later. Thanks! – Brent Writes Code Dec 11, 2021 at 4:04

56

http://0pointer.de/public/systemd-man/systemd.exec.html#Environment= - you have two options (one already pointed by Michael):

```
Environment=
```

and

```
EnvironmentFile=
```

Share Improve this answer

Follow

edited Aug 29, 2013 at 17:25

answered Oct 16, 2012 at 13:55

paluh

661 5 7

9 Just leaving this here: If you decide to use EnviromentFile= make sure your file does not contain 'export VARIABLE=VALUE' statements but just the 'VARIABLE=VALUE' basic statements to make it work. Thought it might possibly help someone out. – Tommy Bravo Nov 9, 2021 at 15:51

This is helpful if you're converting from sysvinit's `/etc/default/` to systemd. – Tom O'Connor May 23 at 23:14

49

The answers by Michael and Michał are helpful and answer the original question of how to set an environment variable for a systemd service. However, one common use for environment variables is to configure sensitive data like passwords in a place that won't accidentally get committed to source control with your application's code.

If that's why you want to pass an environment variable to your service, **do not** use `Environment=` in the unit configuration file. Use `EnvironmentFile=` and point it to another configuration file that is only readable by the service account (and users with root access).

The details of the unit configuration file are visible to any user with this command:

```
systemctl show my_service
```

I put a configuration file at `/etc/my_service/my_service.conf` and put my secrets in there:

```
MY_SECRET=correcthorsebatterystaple
```

Then in my service unit file, I used `EnvironmentFile=` :

```
[Unit]
Description=my_service

[Service]
ExecStart=/usr/bin/python /path/to/my_service.py
EnvironmentFile=/etc/my_service/my_service.conf
User=myservice

[Install]
WantedBy=multi-user.target
```

I checked that `ps auxe` can't see those environment variables, and other users don't have access to `/proc/*/environ` . Check on your own system, of course.

Share Improve this answer Follow

answered May 4, 2018 at 0:29

Don Kirkby
1,174 3 10 23

Why is using Environment= for secrets bad? – Yngvar Kristiansen Jan 15 at 10:53

4 As I said, `systemctl show my_service` will show the unit configuration file contents to any user, including `Environment=` . Try it out. – Don Kirkby Jan 16 at 3:11

11

Michael gave one clean solution but I wanted to get updated env variable from script. Unfortunately executing bash commands is not possible in systemd unit file. Fortunately you can trigger bash inside ExecStart:

http://www.dsm.fordham.edu/cgi-bin/man-cgi.pl?topic=systemd.service&ampsect=5

> Note that this setting does not directly support shell command lines. If shell command lines are to be used they need to be passed explicitly to a shell implementation of some kind.

Example in our case is then:

```
[Service]
ExecStart=/bin/bash -c "ENV=`script`; /bin/myforegroundcmd"
```

Share Improve this answer Follow

answered Jul 23, 2014 at 13:31

user1830432
243 2 6

10 This won't work for multiple reasons (unless it's a "one-shot" service, which is rather pointless). I managed to get the following to work: `/bin/bash -a -c 'source /etc/sysconfig/whatever && exec whatever-program'` . The `-a` ensures the environment is exported to the sub-process (unless you want to prefix all variables in `whatever` with `export`) – Otheus Apr 29, 2015 at 22:42

why it won't work? It should always trigger entire command which includes executing the script, ain't it? – user1830432 Apr 30, 2015 at 8:29

Maybe `ExecStart=/usr/bin/env ENV=script /bin/myforegroundcmd` is a little better solution in this case. – kstep Nov 26, 2015 at 6:18

2 There IS a way to execute a bash command "in" a systemd service file. See this link: coreos.com/os/docs/latest/... – Mark Lakata Jan 13, 2017 at 0:35

1 @GwynethLlewelyn I found a back up of that page. web.archive.org/web/20190716112314/https://coreos.com/os/docs/... – Mark Lakata Oct 21, 2021 at 0:09

0

Don't use Environment= or EnvironmentFile= for credentials / secrets.

Per https://www.freedesktop.org/software/systemd/man/systemd.exec.html#Environment

You should use `LoadCredential=, LoadCredentialEncrypted= or SetCredentialEncrypted=`

> Note that environment variables are not suitable for passing secrets (such as passwords, key material, ...) to service processes. Environment variables set for a unit are exposed to unprivileged clients via D-Bus IPC, and generally not understood as being data that requires protection. Moreover, environment variables are propagated down the process tree, including across security boundaries (such as setuid/setgid executables), and hence might leak to processes that should not have access to the secret data. Use LoadCredential=, LoadCredentialEncrypted= or SetCredentialEncrypted= (see below) to pass data to unit processes securely.

Share Improve this answer Follow

answered Sep 9 at 17:25

Adrnalnrsh

1