Running NGINX Plus as non-root user

The following article describes how to run NGINX Plus as a non-root user:

<https://www.exratione.com/2014/03/running-nginx-as-a-non-root-user/>

This article says that the following filesystem path configuration options need to be changed, and set to locations to which the user has write access:

• error\_log (in the main scope as well as lower scopes)

• access\_log

• pid

• client\_body\_temp\_path

• fastcgi\_temp\_path

• proxy\_temp\_path

• scgi\_temp\_path

• uwsgi\_temp\_path

Notes from Moscow:

Apart from the filesystem permissions there is an important note:

* Note that as Nginx is not launched as root, it cannot bind to privileged ports lower than 1024. So, you should verify that all listen directives have ports > 1024 - alternatively, use  
  setcap CAP\_NET\_BIND\_SERVICE /usr/sbin/nginx
* Also, you should check permissions to the unix sockets in the configuration (if any).
* To avoid warnings at the start the "user" directive should be commented out because non-root master process has no ability to make setuid(2)-like calls.

## Running Nginx as a Non-Root User

By ReasonMarch 1st, 2014[Permalink](https://www.exratione.com/2014/03/running-nginx-as-a-non-root-user/)

There are times when you may find yourself wanting to launch [Nginx](http://nginx.org/) as a non-root user. Perhaps you are running an automated test suite against a local instance of a web application, for example, and the whole process - including starting up and shutting down Nginx - is operated by a bash script. The simple approach in this sort of situation is to set up the local environment so that you can sudo without password entry:

|  |  |
| --- | --- |
| 1 | sudo nginx -c /absolute/path/to/my/nginx.conf |

To enable password-free sudo access for any command for a specific user in Ubuntu create the file **/etc/sudoers.d/username** with the following contents:

|  |  |
| --- | --- |
| 1 | username ALL=(ALL) NOPASSWD:ALL |

What if you cannot do this, however? Your test automation script might be running on build system servers under a user that by design does not have sudo rights. This is where you will start to find that Nginx really doesn't like to be launched as a non-root user: there are a number of settings that default to assuming that the user has write access to places like **/var/log** and **/var/run**. Fortunately all of these settings can be changed. It's just a little painful to wander through the documentation to find them one by one if you don't have a list to hand already.

The following filesystem path configuration options need to be changed, and set to locations to which the user has write access:

* error\_log (in the main scope as well as lower scopes)
* access\_log
* pid
* client\_body\_temp\_path
* fastcgi\_temp\_path
* proxy\_temp\_path
* scgi\_temp\_path
* uwsgi\_temp\_path

Here is an example configuration file that allows the package installation of Nginx on Ubuntu 12.04 to be launched by a non-root user without using sudo. You can find this [set up with a Vagrant VM at GitHub](https://github.com/exratione/non-root-nginx):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71 | #  # A very simple example configuration showing how to launch Nginx as a non-root  # user without sudo access.  #  # Adjust the paths and other settings for your specific circumstances. They are  # currently configured for transient usage - you'd want to pick more permanent  # locations in the filesystem if intending this to run for a while.  #  # Note that as Nginx is not launched as root, it cannot bind to privileged  # ports lower than 1024.  #  # Usage: nginx -c /path/to/this/nginx.conf  #    # This error log will be written regardless of server scope error\_log  # definitions, so we have to set this here in the main scope.  #  # Even doing this, Nginx will still try to create the default error file, and  # log a non-fatal error when it fails. After that things will work, however.  error\_log /tmp/error.log;    # The pidfile will be written to /var/run unless this is set.  pid /tmp/nginx.pid;    worker\_processes 1;    events {    worker\_connections 1024;  }    http {    # Set an array of temp and cache file options that will otherwise default to    # restricted locations accessible only to root.    client\_body\_temp\_path /tmp/client\_body;    fastcgi\_temp\_path /tmp/fastcgi\_temp;    proxy\_temp\_path /tmp/proxy\_temp;    scgi\_temp\_path /tmp/scgi\_temp;    uwsgi\_temp\_path /tmp/uwsgi\_temp;      tcp\_nopush on;    tcp\_nodelay on;    keepalive\_timeout 65;    types\_hash\_max\_size 2048;      include /etc/nginx/mime.types;    index index.html index.htm index.php;      log\_format   main '$remote\_addr - $remote\_user [$time\_local] $status '      '"$request" $body\_bytes\_sent "$http\_referer" '      '"$http\_user\_agent" "$http\_x\_forwarded\_for"';      default\_type application/octet-stream;      server {      # IPv4.      listen 8080;      # IPv6.      listen [::]:8080 default ipv6only=on;        root /var/www;        access\_log /tmp/access.log;      error\_log /tmp/error.log;        location / {        # First attempt to serve request as file, then as directory, then fall        # back to index.html.        try\_files $uri $uri/ /index.html;      }    }  } |