# Teamspeak Installation

Es gibt zwar schon zahlreiche Tutorials für das Installieren eines Teamspeak 3 Server unter Linux im Internet, aber leider noch nicht auf ScriptzBase. Bevor ich jetzt einfach "Copy & Past" mache, schreibe ich lieber selber ein kleines Tutorial;)  
Ich verwende als Linux System Debian, Ubuntu User müssen vor den Befehlen ein „sudo“ verwenden.

**Wichtig: Zuerst das ganze Tutorial durchlesen und dann los legen!**​

**1.) Es ist für die Sicherheit wichtig, das wir den Teamspeak 3 Server nicht unter root rechten Installieren und ausführen!!! Daher legen wir für den Teamspeak Server im SSH einen eignen Benutzer an. Ich benenne diesen als „teamspeakux3“, kann aber auch anders heißen:**

adduser teamspeakux3

Das Verzeichnis für den neuen Benutzer lautet /home/teamspeakux3 in diesem Verzeichnis werden wir den TeamSpeak Server installieren.  
  
  
 **2.) Bevor wir jetzt den Teamspeak 3 Server runterladen, wechseln wir zuerst zum teamspeak SSH-Account, den wir zuvor angelegt haben.**

su teamspeakux3

Und wechseln in das Verzeichnis /home/teamspeakux3

cd /home/teamspeakux3

**3.) Jetzt müssen wir mit dem Befehl "uname -m" noch herausfinden, welche bit Version installiert ist, in der Regel ist das meistens die 64 bit Version.**

uname -m

**4.) Nachdem wir jetzt Wissen welche bit Version unser System hat, laden wir die aktuell Version von TeamSpeak 3 auf dem Server.**  
 *64bit Version*

wget http://dl.4players.de/ts/releases/3.0.12.4/teamspeak3-server\_linux\_amd64-3.0.12.4.tar.bz2

*32bit Version*

wget http://dl.4players.de/ts/releases/3.0.12.4/teamspeak3-server\_linux\_x86-3.0.12.4.tar.bz2

*Ich versuche die hier angegebenen Downloadlinks immer aktuell zu halten, eine Übersicht über die aktuellen Teamspeak 3 Versionen findet Ihr* [***hier***](http://antiref.com?https:/www.teamspeak.com/downloads)*.*  
  
**5.) Sobald der Download abgeschlossen ist, wird das .tar.bz Archiv entpackt.**  
  
*64bit Version*

tar -xjvf teamspeak3-server\_linux\_amd64-3.0.13.3.tar.bz2

*32bit Version*

tar -xjvf teamspeak3-server\_linux\_x86-3.0.13.3.tar.bz2

Die Dateien liegen nun im Unterverzeichnis teamspeak3-server\_linux\_amd64. Falls es euch stört, könnt ihr die Dateien auch eine Ebene höher legen, ist aber nicht wirklich notwendig.  
  
**6.) Nun sind wir auch schon fast so weit. Als letzten Schritt starten wir das Installation Script von Teamspeak.**

teamspeak3-server\_linux\_amd64/ts3server\_minimal\_runscript.sh

In der Ausgabe sind die Daten für den Serveradmin Query Account und der Token für Administration Berechtigung für den TeamSpeak 3 Channel. Der Teamspeak 3 Server wurde automatisch gestartet.  
*Die Daten für den Serveradmin Query Account unbedingt notieren, da Ihr diese Später eventuell noch benötigt.*  
  
**Mit der Installation wären Wir so weit fertig, bitte den SSH-Client noch nicht schließen!**  
Da wir den Teamspeak 3 Server nur mit dem Start Script gestartet haben, würde dieser beim schließen der SSH-Console einfach offline gehen.  
  
**7.) Via Client auf den Teamspeak Server verbinden und konfigurieren**  
Die Teamspeak Clients können für so ziemlich jedes Betriebssystem downgeloaded werden. Egal ob Windows, Linux, Mac, FreeBSD oder Android. Ich verwende in diesem Tutorial den Client für Windows.  
  
Client Starten und auf den Server verbinden („Verbindungen“ -> „Verbinden“). Unter „Server Adresse:“ muss die IP Adresse oder der Hostnamen des Servers eingetragen werden. Der Standardport für den ersten TeamSpeak Channel auf dem Server lautet "9987"  
  
Der Server erwartet nun die Eingabe des Berechtigungsschlüssels "tokens" der nach der Installation im SSH ausgegeben wurde. (sehe schritt 6)  
  
  
  
**8.)** **Ihr solltet nun auf eure Server Konsole zurück wechseln und mit „STRG + C“ die Ausführung des Teamspeak Servers beenden. Keine Angst, der Server wird gleich wieder gestartet, aber dieses mal mit dem korrekten Startup Script, sodass der Dienst im Hintergrund läuft und nicht nach dem Logout gekillt wird.**  
  
**Teamspeak Server starten:**

/home/teamspeakux3/teamspeak3-server\_linux\_amd64/ts3server\_startscript.sh start

**Mit folgenden Befehl kann man den Teamspeak Server wieder stoppen**

/home/teamspeakux3/teamspeak3-server\_linux\_amd64/ts3server\_startscript.sh stop

**TeamSpeak Server neu starten:**

/home/teamspeakux3/teamspeak3-server\_linux\_amd64/ts3server\_startscript.sh restart

Und damit wären wir auch schon fertig, der TeamSpeak Server ist nun voll Einsatzbereit und die Konsole kann geschlossen werden.  
  
**TeamSpeak 3 Startup Script**  
  
**Damit der Server nicht immer manuell gestartet werden muss, wenn der root Server mal abstürtzt oder neu gestartet werden muss, empfehle ich die Verwendung eines Teamspeak 3 Server Startup Script**  
Im Verzeichniss /etc/init.d müsst Ihr mit eurem Lieblingseditor eine Datei erstellen und befüllen. Ich verwende hier den namen teamspeakserver. Die Spalte USER und DIR muss gff. noch angepasst werden.

nano /etc/init.d/teamspeakserver

#!/bin/sh

### BEGIN INIT INFO

# Provides: teamspeak3

# Required-Start: $local\_fs $network

# Required-Stop: $local\_fs $network

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Description: Teamspeak 3 Server

### END INIT INFO

# Customize values for your needs: "User"; "DIR"

USER="teamspeakux3"

DIR="/home/teamspeakux3/teamspeak3-server\_linux\_amd64"

###### Teamspeak 3 server start/stop script ######

case "$1" in

start)

su $USER -c "${DIR}/ts3server\_startscript.sh start"

;;

stop)

su $USER -c "${DIR}/ts3server\_startscript.sh stop"

;;

restart)

su $USER -c "${DIR}/ts3server\_startscript.sh restart"

;;

status)

su $USER -c "${DIR}/ts3server\_startscript.sh status"

;;

\*)

echo "Usage: {start|stop|restart|status}" >&2

exit 1

;;

esac

exit 0

Nun müsst ihr das Script noch ausführbar machen, dies machen wir mit dem befehl:

chmod 755 /etc/init.d/teamspeakserver

Das Script ist soweit fertig und muss nun noch als Autostart definiert werden:

update-rc.d teamspeakserver defaults

**Teamspeak Server starten:**

/etc/init.d/teamspeakserver start

**Teamspeak Server stoppen:**

/etc/init.d/teamspeakserver stop

**Teamspeak Server neu starten:**

/etc/init.d/teamspeakserver restart

**Teamspeak Server Status anzeigen lassen:**

/etc/init.d/teamspeakserver status

Beim nächsten Reboot wird der Teamspeak 3 Server automatisch gestartet.  
  
**So das war es für das erste. Der Teamspeak 3 Server läuft und Startet nun automatisch.  
  
TeamSpeak 3 Server deinstallieren**  
Mit diesen einfachen Schritten könnt Ihr den TeamSpeak Server von eurem Server entfernen.  
  
**1.) Teamspeak Server löschen**  
Einfach den Ordner vom Teamspeak Server löschen

Rmdir -r /home/teamspeakux3/teamspeak3-server\_linux\_amd64

**2.) Root User löschen**

deluser teamspeakux3

Das war es eigentlich schon, der Teamspeak Server ist nun vom Server entfernt.  
  
**Startup Script löschen**  
Wer das Startup Script nutzt muss noch folgende dinge erlidgen.  
  
**1.) TeamSpeak 3 Startup Script löschen**

/etc/init.d/teamspeakserver

**2.) Autostart entfernen**

update-rc.d teamspeakserver remove

Mfg

# Easy-Wi Installation VPS:

Hallo in diesem video wird [Easy-wi](http://antiref.com?https:/easy-wi.com/) vorgestellt und installiert.

* [Zum Video](https://www.youtube.com/watch?v=7tyvkJSOEwc)  
  Homepage: [https://easy-wi.com/](http://antiref.com?https:/easy-wi.com/)  
    
  Es handelt sich dabei um einen Webpanel mit REST API und vielen erleichterungen beim server verwalten. Es ist freeware!!!  
    
    
  Viel Spass:  
    
    
    
    
  Für alle die es gesucht haben.  
    
  Die Deutsche Download Seite befindet sich an dieser stelle: [https://easy-wi.com/de/downloads/](http://antiref.com?https:/easy-wi.com/de/downloads/)  
    
  Die Englishe hier: [https://easy-wi.com/uk/downloads/](http://antiref.com?https:/easy-wi.com/uk/downloads/)  
    
  viel erfolg!

# Hugo auf Ubuntu installieren:

Hugo is written in [Go](http://golang.org/) with support for multiple platforms.

The latest release can be found at [Hugo Releases](https://github.com/spf13/hugo/releases). We currently provide pre-built binaries for  Windows,  Linux,  FreeBSD and  OS X (Darwin) for x64, i386 and ARM architectures.

Hugo may also be compiled from source wherever the Go compiler tool chain can run, e.g. for other operating systems including DragonFly BSD, OpenBSD, Plan 9 and Solaris. See <http://golang.org/doc/install/source> for the full set of supported combinations of target operating systems and compilation architectures.

## Installing Hugo (binary)

Installation is very easy. Simply download the appropriate version for your platform from [Hugo Releases](https://github.com/spf13/hugo/releases). Once downloaded it can be run from anywhere. You don’t need to install it into a global location. This works well for shared hosts and other systems where you don’t have a privileged account.

Ideally, you should install it somewhere in your PATH for easy use. /usr/local/bin is the most probable location.

On macOS, if you have [Homebrew](http://brew.sh/), installation is even easier: just run brew update && brew install hugo.

For a more detailed explanation follow the corresponding installation guides:

* [Installation on macOS](https://gohugo.io/tutorials/installing-on-mac/)
* [Installation on Windows](https://gohugo.io/tutorials/installing-on-windows/)

### Installing Pygments (optional)

The Hugo executable has one optional external dependency for source code highlighting (Pygments).

If you want to have source code highlighting using the [highlight shortcode](https://gohugo.io/extras/highlighting/), you need to install the Python-based Pygments program. The procedure is outlined on the [Pygments home page](http://pygments.org/).

## Upgrading Hugo

Upgrading Hugo is as easy as downloading and replacing the executable you’ve placed in your PATH.

## Installing Hugo on Linux from native packages

### Arch Linux

You can install Hugo from the [Arch user repository](https://aur.archlinux.org/) on Arch Linux or derivatives such as Manjaro.

sudo pacman -S yaourt

yaourt -S hugo

Be aware that Hugo is built from source. This means that additional tools like [Git](https://git-scm.com/) and [Go](https://golang.org/doc/install) will be installed as well.

### Debian and Ubuntu

Hugo has been included in Debian and Ubuntu since 2016, and thus installing Hugo is as simple as:

sudo apt install hugo

Pros:

* Native Debian/Ubuntu package maintained by Debian Developers
* Pre-installed bash completion script and man pages for best interactive experience

Cons:

* Might not be the latest version, especially if you are using an older stable version (e.g., Ubuntu 16.04 LTS). Until backports and PPA are available, you may consider installing the Hugo snap package to get the latest version of Hugo, as described below.

### Fedora and Red Hat

* <https://copr.fedorainfracloud.org/coprs/spf13/Hugo/> (updated to Hugo v0.16)
* <https://copr.fedorainfracloud.org/coprs/daftaupe/hugo/> (updated to Hugo v0.19)

See also [this discussion](https://discuss.gohugo.io/t/solved-fedora-copr-repository-out-of-service/2491).

### Snap package for Hugo

In any of the [Linux distributions that support snaps](http://snapcraft.io/docs/core/install):

snap install hugo

Note: Hugo-as-a-snap can write only inside the user’s $HOME directory—and gvfs-mounted directories owned by the user—because of Snaps’ confinement and security model. More information is also available [in this related GitHub issue](https://github.com/spf13/hugo/issues/3143).

## Installing from source

### Prerequisite tools for downloading and building source code

* [Git](http://git-scm.com/)
* [Go](http://golang.org/) 1.8+
* [govendor](https://github.com/kardianos/govendor)

### Vendored Dependencies

Hugo uses [govendor](https://github.com/kardianos/govendor) to vendor dependencies, but we don’t commit the vendored packages themselves to the Hugo git repository. Therefore, a simple go get is not supported since go get is not vendor-aware. You **must use govendor** to fetch Hugo’s dependencies.

### Fetch from GitHub

go get github.com/kardianos/govendor

govendor get github.com/spf13/hugo

govendor get will fetch Hugo and all its dependent libraries to $HOME/go/src/github.com/spf13/hugo, and compile everything into a final hugo (or hugo.exe) executable, which you will find sitting inside $HOME/go/bin/, all ready to go!

Windows users: where you see the *$HOME* environment variable above, replace it with *%USERPROFILE%*.

## Contributing

Please see the [contributing guide](https://gohugo.io/doc/contributing/) if you are interested in working with the Hugo source or contributing to the project in any way.

[[1]](#endnote-1)

1. Copyright by CGNKev [↑](#endnote-ref-1)