Paketmanager

- dpkg / APT / Aptitude
  - Alle durch Benutzer geänderten paketierte Konfigurationsdateien ausgeben
  - Alle installierten Pakete auf einen anderen Server übertragen
  - dpkg-Cheat-Sheet
  - dpkg-rc Pakete entfernen
  - Was bedeuten die verschiedenen flags bei dpkg -l?
  - APT / Aptitude - Tipps und Tricks
  - GPG-Key eines Repositorys in Apt hinzufügen

- RPM / YUM
  - RPM Kurzübersicht
  - RPM-Datenbank abfragen
  - RPM-Datenbank defekt
  - RPM-Paket entpacken
  - YUM - Tipps und Tricks

- Häufig genutzte 3rd-Party Repositories
  - download.docker.com (Docker + Compose)
  - packages.icinga.com (Icinga2)
  - deb.sury.org (PHP)
  - repo.mongodb.org (MongoDB Community Edition)
  - repo.percona.com (Percona MySQL and others)
  - packagecloud.io/crowdsec (Crowdsec Security)
  - packages.gitlab.com (Gitlab)
  - nginx.org (nginx)
  - artifacts.elastic.co (Elasticsearch)
  - nodejs.org (nodeJS)

dpkg / APT / Aptitude

Alle durch Benutzer geänderten paketierte Konfigurationsdateien ausgeben

```
dpkg-query -W -f='${Conffiles}\n' '*' | awk 'OFS="  "{print $2,$1}' | LANG=C md5sum -c 2>/dev/nu
```

Alle installierten Pakete auf einen anderen Server übertragen

> Vorher sollten als entfernt markierte Pakete gelöscht werden, siehe hier.

Quelle:

```
dpkg --get-selections > selection.txt
```

dann die Datei per scp oder Copy&Paste auf das Ziel übertragen. Dort:

```
dpkg --set-selections < selection.txt
```

Falls hier Warnings auftreten siehe unten.

Danach können die Pakete installiert werden:

```
aptitude install
# oder
apt-get -u dselect-upgrade
```

Sollte es bei „–set-selections“ Warnungen gegeben haben, muss noch dselect installiert werden:

```
apt-get install dselect
dselect
  -> Update
  -> Install
  -> Quit
```

Folgende Alternative funktioniert auch, wenn man Pakete nur selektiv übertragen möchte. Mit diesem Beispiel erhalte ich eine einfach zu übertragende Liste aller Pakete, die `perl` im Namen oder der Beschreibung haben (läßt sich dann einfach an `aptitude install` übergeben):

```
dpkg -l | grep perl | awk '{ printf "%s ", $2 }'
```

dpkg-Cheat-Sheet

Pakete auf Version festhalten / hold

ein Paket auf „hold“ setzen

```
echo <paketname> hold | dpkg --set-selections
```

ein Paket wieder fürs Updaten freigeben

```
echo <paketname> install | dpkg --set-selections
```

gehaltene Pakete anzeigen

```
dpkg --get-selections | awk '$2 == "hold" { print $1 }'
```

Fremdpakete finden

Die folgenden Tipps sind nicht immer zuverlässig

Pakete finden, die aus keinem der konfigurierten Repositories stammen (auf einem sauberen System sollte diese Liste leer sein):

```
aptitude search '?narrow(?not(?archive("^[^n][^o].*$")),?version(CURRENT))'
```

Pakete finden, die nicht aus einem Debian-Repo stammen

```
dpkg -l | awk '/^.i/ {print $2}' | xargs apt-cache policy | awk '/^[a-z0-9.\-]+:/ {pkg=$1}; /\*\
```

dpkg-rc Pakete entfernen

Auf einem gut gereiften System oder z.B auch nach einem Dist-Upgrade sammeln sich gerne teilweise entfernte Pakete an. Diese erkennt man an dem rc im dpkg Status. Das bedeutet, dass das Paket entfernt wurde, aber die Konfigurationsdateien noch auf dem System verblieben sind.

Am besten vorher noch ein autoremove fahren:

```
apt autoremove
```

Danach kann man sich die Liste der halb-deinstallierten Pakete anschauen:

```
dpkg --list | grep "^rc"
```

Das sieht dann z.B. so aus:

```
rc  dctrl-tools                 2.14.5                      amd64       Command-line tools to process Debian
package information
rc  libaugeas0                   0.7.2-1                     amd64       The augeas configuration editing library and
API
rc  libbind9-60                  1:9.7.3.dfsg-1~squeeze10          amd64       BIND9 Shared Library used by BIND
rc  libdns69                     1:9.7.3.dfsg-1~squeeze10          amd64       DNS Shared Library used by BIND
rc  libisc62                    1:9.7.3.dfsg-1~squeeze10          amd64       ISC Shared Library used by BIND
rc  libisccc60                   1:9.7.3.dfsg-1~squeeze10          amd64       Command Channel Library used by
BIND
```

Das rc steht dabei für als „entfernt“ markierte Pakete, deren Konfigurationsdateien noch auf dem System liegen.

Gelegentlich empfiehlt es sich dann aufzuräumen. Mit folgendem Snippet lassen sich all diese Pakete bequem auf einmal entfernen:

```
dpkg --list | grep "^rc" | cut -d " " -f 3 | xargs dpkg --purge
```

Alternativ mit einem apt Kommando:

```
apt-get purge $(dpkg -l | awk '/^rc/ { print $2 }')
```

Was bedeuten die verschiedenen flags bei dpkg -l?

m Header von „dpkg -l“ steht die Erklärung dazu. Hier eine kurze Übersicht:

```
$ dpkg -l | head -6

Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                                 Version                                 Architecture Description

ii  accountsservice                      0.6.35-0ubuntu7.2                       amd64        query and manipulate
user account information
```

Hier die Beschreibung der einzelnen Felder:

Erster Buchstabe	gewünschter Zustand („selection state“)
u	unknown
i	install
r	remove/deinstall
p	purge (Paket soll inklusive der Konfigurationsdateien entfernt werden)
h	hold

Zweiter Buchstabe	aktueller Zustand
n	not-installed
i	installed
c	config-files (nur Konfigurationsdateien sind installiert)
u	unpacked
f	half-configured (Konfiguration ist aus irgendeinem Grund fehl geschlagen)
h	half-installed (Installation ist aus irgendeinem Grund fehl geschlagen)
w	triggers-awaited (Paket wartet auf Trigger durch ein anderes Paket)

Zweiter Buchstabe	aktueller Zustand
t	triggers-pending (Paket wurde getriggert)
B	broken (Installation/Update fehlgeschlagen)

Dritter Buchstabe	Fehler (da sollte normalerweise nichts stehen)
r	reinst-required (Paket ist kaputt und sollte neu installiert werden)

APT / Aptitude - Tipps und Tricks

- Verfügbare Updates anzeigen, aber nicht installieren:

```
# aptitude -F%p --disable-columns search ~U
# oder:
# apt-get --just-print dist-upgrade
```

- durch „aptitude hold“ gehaltene Pakete:

```
aptitude search ~ahold
```

- Abhängigkeiten, Vorschläge, Empfohlen, Konflikte eines Paketes anzeigen:

```
# apt-cache depends tar
tar
  PreDepends: libc6
  Suggests: bzip2
  Suggests: ncompress
  Suggests: xz-utils
  Conflicts: cpio
  Breaks: dpkg-dev
  Replaces: cpio
```

- warum ist ein bestimmtes Paket installiert worden (required by):

```
# aptitude why tar
i   dpkg PreDepends tar (>= 1.23)
```

- Installierte Pakete anzeigen die kein anderes Paket voraussetzt (depends on):

```
aptitude search '~i ! ~R ~i'

# erweitert auf "Empfohlen" oder "Vorgeschlagen" (recommends / suggests):
aptitude search '~i ! ~R ~i ! ~Rrecommends:~i  ! ~Rsuggests:~i'

# für alle Pakete, installiert oder nicht:
aptitude search '! ~R .'
```

GPG-Key eines Repositorys in Apt hinzufügen

Beim Einbinden eines neuen Repositorys erscheint diese Meldung, wenn der GPG-Key, mit dem die Pakete signiert wurden, nicht in der APT-Datenbank enthalten ist:

```
W: GPG error: http://packages.linuxmint.com sarah Release:
The following signatures couldn't be verified because the
public key is not available: NO_PUBKEY A6616109451BBBF2
```

Das läßt sich ganz einfach beheben:

```
# apt-key adv --recv-keys --keyserver keyserver.ubuntu.com A6616109451BBBF2

Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --homedir /tmp/tmp.in8AoFu6Qi --no-auto-check-trustdb --trust-model always --keyring /etc/apt/trusted.gpg --primary-keyring /etc/apt/trusted.gpg --keyring /etc/apt/trusted.gpg.d/steam.gpg --recv-keys --keyserver keyserver.ubuntu.com A6616109451BBBF2
gpg: requesting key 451BBBF2 from hkp server keyserver.ubuntu.com
gpg: key 451BBBF2: public key "Linux Mint Repository Signing Key <root@linuxmint.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

RPM / YUM

RPM Kurzübersicht

Einige Tips zum Umgang mit dem Redhat Package Manager (RPM)

* Alle installierten Pakete anzeigen (in dieser Liste läßt sich auch suchen, wenn man in der Ausgabe grept)

```
rpm -qa
```

* Informationen zu einem installierten Paket ausgeben:

```
rpm -qi paketname
```

* Herausfinden zu welchem Paket eine bestimmte Datei gehört

```
rpm -qf /bin/cp
```

* Liste aller in einem Paket enthaltenen Dateien anzeigen

```
rpm -ql paketname
```

* Mit der zusätzlichen Angabe des Parameters p kann man auch ein nicht installiertes RPM-Paket abfragen

```
rpm -qlp programm-1.0.rpm
```

* ein installiertes Paket auf Veränderungen prüfen

```
rpm -V paketname
```

* ein runtergeladenes Paket überprüfen

```
rpm -K paketname
```

* Die Dateirechte aller RPM-Pakete auf den Installationszustand zurücksetzen (für Dateien, die NICHT im Päckchen enthalten sind, gilt dies nicht)

```
for package in `rpm -qa` ; do rpm --setperms --setugids "${package}"; done
```

RPM-Datenbank abfragen

Mit `rpm -q` lassen sich detailierte Informationen aus der RPM-Datenbank ziehen. Welche Tags einem hier zur Verfügung stehen zeigt dieser Befehl:

```
[root@ov ~]# rpm --querytags | sort
ARCH
ARCHIVESIZE
BASENAMES
BUILDARCHS
BUILDHOST
BUILDTIME
C
CAPABILITY
CHANGELOGNAME
CHANGELOGTEXT
CHANGELOGTIME
CLASSDICT
CONFLICTFLAGS
CONFLICTNAME
CONFLICTS
CONFLICTVERSION
COOKIE
DEPENDSDICT
DESCRIPTION
DIRINDEXES
DIRNAMES
DISTRIBUTION
DISTTAG
DISTURL
DSAHEADER
E
EPOCH
EXCLUDEARCH
EXCLUDEOS
EXCLUSIVEARCH
EXCLUSIVEOS
FILECLASS
```

FILECOLORS
FILECONTEXTS
FILEDEPENDSN
FILEDEPENDSX
FILEDEVICES
FILEDIGESTALGO
FILEDIGESTS
FILEFLAGS
FILEGROUPNAME
FILEINODES
FILELANGS
FILELINKTOS
FILEMD5S
FILEMODES
FILEMTIMES
FILENAMES
FILEPROVIDE
FILERDEVS
FILEREQUIRE
FILESIZES
FILESTATES
FILEUSERNAME
FILEVERIFYFLAGS
FSCONTEXTS
FSNAMES
FSSIZES
GIF
GROUP
HDRID
HEADERI18NTABLE
HEADERIMAGE
HEADERIMMUTABLE
HEADERREGIONS
HEADERSIGNATURES
ICON
INSTALLCOLOR
INSTALLTID
INSTALLTIME
INSTPREFIXES
LICENSE

LONGARCHIVESIZE
LONGFILESIZES
LONGSIGSIZE
LONGSIZE
N
NAME
O
OBSOLETEFLAGS
OBSOLETENAME
OBSOLETES
OBSOLETEVERSION
OLDFILENAMES
OPTFLAGS
ORIGBASENAMES
ORIGDIRINDEXES
ORIGDIRNAMES
ORIGFILENAMES
OS
P
PACKAGER
PATCH
PATCHESFLAGS
PATCHESNAME
PATCHESVERSION
PAYLOADCOMPRESSOR
PAYLOADFLAGS
PAYLOADFORMAT
PKGID
PLATFORM
POLICIES
POSTIN
POSTINPROG
POSTTRANS
POSTTRANSPROG
POSTUN
POSTUNPROG
PREFIXES
PREIN
PREINPROG
PRETRANS

PRETRANSPROG
PREUN
PREUNPROG
PROVIDEFLAGS
PROVIDENAME
PROVIDES
PROVIDEVERSION
PUBKEYS
R
RECONTEXTS
RELEASE
REMOVETID
REQUIREFLAGS
REQUIRENAME
REQUIRES
REQUIREVERSION
RHNPLATFORM
RPMVERSION
RSAHEADER
SHA1HEADER
SIGGPG
SIGMD5
SIGPGP
SIGSIZE
SIZE
SOURCE
SOURCEPACKAGE
SOURCEPKGID
SOURCERPM
SUMMARY
TRIGGERCONDS
TRIGGERFLAGS
TRIGGERINDEX
TRIGGERNAME
TRIGGERSCRIPTPROG
TRIGGERSCRIPTS
TRIGGERTYPE
TRIGGERVERSION
URL
V

```
VENDOR
VERIFYSCRIPT
VERIFYSCRIPTPROG
VERSION
XPM
```

Nun läßt sich mit Hilfe dieser Tags und dem Parameter `--queryformat` eine ordentliche Ausgabe erzeugen. Will man z.B. wissen für welche Architektur ein Paket gebaut wurde empfiehlt sich etwa dieser Befehl:

```
[root@ov ~]# rpm -q --queryformat "%{NAME}-%{VERSION}.%{ARCH}\n" libxml2
libxml2-2.7.3.x86_64
libxml2-2.7.3.i386
```

dies bedeutet, dass das Paket libxml in der i386 und in der 64bit Version (x86_64) installiert ist.

Eine praktische Anwendung des Queryformat bietet sich auch an. Vor allem auf Systemen, auf denen 64- und 32-bit Pakete installiert sind. Einfach folgenden Code in /etc/profile oder in eine Datei in /etc/profile.d eintragen. Nach erneuter Anmeldung oder sourcen von /etc/profile steht das „rpmqa“-Kommando zur Verfügung, das folgende Ausgabeformat hat:

```
acl-2.2.39-6.el5.x86_64
acpid-1.0.4-9.el5_4.2.x86_64
alsa-lib-1.0.17-1.el5.i386
alsa-lib-1.0.17-1.el5.x86_64
amavisd-new-2.4.5-1.el5.noarch
amtu-1.0.6-1.el5.x86_64
anacron-2.3-45.el5.centos.x86_64
ant-1.6.5-2jpp.2.x86_64
antlr-2.7.6-4jpp.2.x86_64
apr-1.2.7-11.el5_3.1.i386
apr-1.2.7-11.el5_3.1.x86_64
apr-devel-1.2.7-11.el5_3.1.x86_64
apr-util-1.2.7-11.el5.i386
apr-util-1.2.7-11.el5.x86_64
apr-util-devel-1.2.7-11.el5.x86_64
[...]
```

RPM-Datenbank defekt

Wenn rpm oder yum beim Aufruf hängen und keine Abfragen mehr gehen ist mit ziemlicher Wahrscheinlichkeit die RPM-Datenbank defekt. Die Datenbank kann folgendermaßen wiederhergestellt werden:

zuerst alle laufenden rpm-Prozesse abwürgen:

```
ps -ax | grep rpm
kill -9 <pid>
```

mit folgenden Befehlen läßt sich die rpm-Datenbank neu aufbauen:

```
rm /var/lib/rpm/__*
rpm --rebuilddb
rpm --initdb
```

RPM-Paket entpacken

Alle Dateien aus einem RPM entpacken:

```
$ rpm2cpio RPM_file | cpio -idv
```

Ein bestimmtes File aus einem RPM entpacken:

```
$ rpm2cpio RPM_file | cpio -id individual_file(s)
```

z.B. libcrypto.so.0.9.7a und libssl.so.0.9.7a aus openssl-0.9.7a-2.i386.rpm extrahieren:

```
$ rpm2cpio openssl-0.9.7a-2.i386.rpm | cpio -it egrep "libcrypto.so.0.9.7a|libssl.so.0.9.7a"

-rwxr-xr-x   1 root     root       992092 Feb 27 12:10 ./lib/libcrypto.so.0.9.7a
-rwxr-xr-x   1 root     root       216004 Feb 27 12:10 ./lib/libssl.so.0.9.7a

$ rpm2cpio openssl-0.9.7a-2.i386.rpm | cpio -idv ./lib/libssl.so.0.9.7a ./lib/libcrypto.so.0.9.7a
```

Eine cpio Datei entpacken:

```
$ cpio -iv < cpio_file
```

Den Inhalt einer cpio Datei anzeigen:

```
$ cpio -itv < cpio_file
```

Ein cpio file mit den Dateien im aktuellen Verzeichnis erstellen:

```
$ ls | cpio -o > cpio_file
```

YUM - Tipps und Tricks

- Pakete nach Namen und Beschreibung suchen:

```
yum search blabla
```

- Info zu einem bestimmten Paket anzeigen:

```
yum info bind
```

- Pakete nur nach Namen suchen:

```
yum list blabla
```

- Suchen welches Paket eine bestimmte Datei zur Verfügung stellt:

```
yum whatprovides 'libstdc++.so.5'
```

- kürzlich neu hinzugekommene Pakete eines Repos anzeigen

```
yum list recent
```

- Alle Abhängigkeiten eines Paketes anzeigen:

```
yum deplist bind
```

- lokal runtergeladenes Paket mit yum installieren und eventuelle Abhängigkeiten gleich aus dem Repository nachziehen, z.B.:

```
yum --nogpgcheck localinstall opera opera-9.24-20071015.6-shared-qt.i386-en.rpm
```

Häufig genutzte 3rd-Party Repositories

Beschreibt die schnelle Einrichtung von häufig genutzten 3rd-Party Repositories wie z.B. für docker oder des Icinga2-Clients

Häufig genutzte 3rd-Party Repositories

download.docker.com (Docker + Compose)

```
apt-get update
apt-get install ca-certificates curl gnupg
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
chmod a+r /etc/apt/keyrings/docker.gpg
echo "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | tee
/etc/apt/sources.list.d/docker.list > /dev/null
apt-get update
apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

packages.icinga.com (Icinga2)

```
apt update
apt -y install apt-transport-https wget gnupg
wget -O - https://packages.icinga.com/icinga.key | gpg --dearmor -o /usr/share/keyrings/icinga-archive-keyring.gpg
DIST=$(awk -F"[)(]+" '/VERSION=/ {print $2}' /etc/os-release); \
 echo "deb [signed-by=/usr/share/keyrings/icinga-archive-keyring.gpg] https://packages.icinga.com/debian icinga-${DIST} main" > \
 /etc/apt/sources.list.d/${DIST}-icinga.list
 echo "deb-src [signed-by=/usr/share/keyrings/icinga-archive-keyring.gpg] https://packages.icinga.com/debian icinga-${DIST} main" >> \
 /etc/apt/sources.list.d/${DIST}-icinga.list
apt update
```

Häufig genutzte 3rd-Party Repositories

deb.sury.org (PHP)

```
apt update
apt -y install lsb-release ca-certificates curl
curl -sSLo /usr/share/keyrings/deb.sury.org-php.gpg https://packages.sury.org/php/apt.gpg
sh -c 'echo "deb [signed-by=/usr/share/keyrings/deb.sury.org-php.gpg] https://packages.sury.org/php/
$(lsb_release -sc) main" > /etc/apt/sources.list.d/php.list'
apt update
```

repo.mongodb.org (MongoDB Community Edition)

Add the MongoDB apt repository and install the MongoDB packages

Replace the Version (4.4) with the one you want to install.

```
apt-get install gnupg curl
curl -fsSL https://pgp.mongodb.com/server-4.4.asc | gpg -o /usr/share/keyrings/mongodb-server-4.4.gpg --dearmor
echo "deb [ signed-by=/usr/share/keyrings/mongodb-server-4.4.gpg ] http://repo.mongodb.org/apt/debian bullseye/mongodb-org/4.4 main" | tee /etc/apt/sources.list.d/mongodb-org-4.4.list
apt-get update
apt-get install -y mongodb-org
```

repo.percona.com (Percona MySQL and others)

```
apt update
apt install curl
curl -O https://repo.percona.com/apt/percona-release_latest.generic_all.deb
apt install gnupg2 lsb-release
dpkg -i percona-release_latest.generic_all.deb
apt update
percona-release setup ps80
```

packagecloud.io/crowdsec (Crowdsec Security)

```
mkdir -p /etc/apt/keyrings/
curl -fsSL https://packagecloud.io/crowdsec/crowdsec/gpgkey | gpg --dearmor >
/etc/apt/keyrings/crowdsec_crowdsec-archive-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/crowdsec_crowdsec-archive-keyring.gpg]
https://packagecloud.io/crowdsec/crowdsec/debian bookworm main " >
/etc/apt/sources.list.d/crowdsec_crowdsec.list
echo "#deb-src [signed-by=/etc/apt/keyrings/crowdsec_crowdsec-archive-keyring.gpg]
https://packagecloud.io/crowdsec/crowdsec/debian bookworm main" >>
/etc/apt/sources.list.d/crowdsec_crowdsec.list
apt update
```

packages.gitlab.com (Gitlab)

gitlab-ce

```
apt-get update -y
apt-get install -y curl ca-certificates apt-transport-https gnupg2
curl -s https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh | bash
apt-get update -y
```

gitlab-runner

```
curl -L "https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh" | bash
cat <<EOF | sudo tee /etc/apt/preferences.d/pin-gitlab-runner.pref
Explanation: Prefer GitLab provided packages over the Debian native ones
Package: gitlab-runner
Pin: origin packages.gitlab.com
Pin-Priority: 1001
EOF
apt-get update
```

nginx.org (nginx)

```
apt update
apt install curl gnupg2 ca-certificates lsb-release debian-archive-keyring
curl https://nginx.org/keys/nginx_signing.key | gpg --dearmor | tee /usr/share/keyrings/nginx-archive-keyring.gpg
>/dev/null
# stable nginx
echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg] http://nginx.org/packages/debian
`lsb_release -cs` nginx" | tee /etc/apt/sources.list.d/nginx.list
# mainline nginx
#echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg]
http://nginx.org/packages/mainline/debian `lsb_release -cs` nginx" | tee /etc/apt/sources.list.d/nginx.list
echo -e "Package: *\nPin: origin nginx.org\nPin: release o=nginx\nPin-Priority: 900\n" | tee
/etc/apt/preferences.d/99nginx
apt update
apt install nginx
```

artifacts.elastic.co (Elasticsearch)

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg
apt-get install apt-transport-https
echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | tee /etc/apt/sources.list.d/elastic-8.x.list
apt-get update && apt-get install elasticsearch
```

nodejs.org (nodeJS)

> NodeJS bietet viele verschiedene Varianten. Daher ist es am einfachsten direkt auf der Herstellerseite zu schauen:
>
> für Debian und Co: https://github.com/nodesource/distributions
>
> andere: https://nodejs.org/en/download/package-manager