EOTK in drei LXD-Containern: Debian 13, Ubuntu 20.04 und Ubuntu 24.04

01.11.2026 · @Sebastian Ebner · Quelle: KI Werkzeug Cloude

Beispielnamen: Container `debian13ax`, `ubuntu2004ax`, `ubuntu2404ax`. In allen drei heißen **Benutzer und eotk-Projekt `ax`**. Domain `xxx.com`, Onion-Adresse `onion3-adresse`. Durch die eigenen ersetzen. Alle drei Systeme sind getestet und laufen mit openresty 1.31.8.1 (nginx 1.31) und Tor 0.4.9.15.

1. So funktioniert das mit den drei Containern

Ebene	Was dort läuft
V-Server (Host)	Ubuntu mit **LXD** (über snap). Hier laufen alle `lxc`-Befehle. Der Kernel des V-Servers wird von allen Containern mitbenutzt.
Speicher-Pool	Ein Ordner bzw. eine Abbilddatei, in dem alle Container und ihre Snapshots liegen (`lxc storage list`).
Container `debian13ax`, `ubuntu2004ax`, `ubuntu2404ax`	Drei getrennte Linux-Systeme mit eigenen Programmen, eigenen Benutzern und eigener IP im LXD-Netz. Was in einem Container passiert, betrifft die anderen nicht.
Benutzer `ax` (in jedem Container)	Eigener Benutzer pro Container. Unter ihm laufen eotk, Tor und nginx, nicht unter root.
eotk-Projekt `ax`	Konfiguration `/home/ax/eotk/ax.conf`, Schlüssel in `/home/ax/eotk/secrets.d/`.

Wichtig für den Betrieb

- Jeder Container kann eine **eigene** Onion-Adresse haben, zum Beispiel einen Test-Schlüssel. Dann laufen alle drei gleichzeitig.
- Dieselbe Onion-Adresse darf immer nur in **einem** Container laufen. Sonst kämpfen zwei Tor-Instanzen um die Adresse und sie springt hin und her. Vor dem Umschalten im alten Container stoppen: `lxc exec altername -- su - ax -c "cd ~/eotk && ./eotk stop ax"`.
- Tor verbindet nur nach außen. Die Container brauchen deshalb keine Portfreigabe und keine öffentliche IP.
- Für den Dauerbetrieb eignen sich **Debian 13** oder **Ubuntu 24.04**. Ubuntu 20.04 bekommt seit 2025 keine kostenlosen Sicherheits-Updates mehr und taugt nur noch zum Testen.

	Debian 13 „Trixie“	Ubuntu 20.04	Ubuntu 24.04
Container anlegen	`lxc launch images:debian/13 debian13ax`	`lxc launch ubuntu:20.04 ubuntu2004ax`	`lxc launch ubuntu:24.04 ubuntu2404ax`
Bauskript in `opt.d`	`debian-13-trixie.sh`	`build-ubuntu-20.04.sh`	`build-ubuntu-24.04.sh`
OpenSSL	3.5	1.1.1	3.0

	Debian 13 „Trixie“	Ubuntu 20.04	Ubuntu 24.04
`lib.sh` Zeile nginx / Tor	57 / 92	55 / 101	55 / 101
`lib.sh` Zeile Schlüssel nginx / Tor	58 / 96	57 / 103	57 / 103

2. LXD: Images und Container

Befehl (auf dem V-Server)	Zweck
`lxd init`	Nur einmal bei der Ersteinrichtung des V-Servers
`lxc remote list`	Quellen: `ubuntu:` (Canonical) und `images:` (alle anderen Systeme)
`lxc image list images: type=container architecture=x86_64 -c l --format csv | sort`	Alle Systeme von `images:` (Debian, Alpine, Fedora, Rocky …), **ohne Ubuntu**
`lxc image list ubuntu: type=container architecture=x86_64 -c ld`	Ubuntu-Versionen (Kurzname = Anfangsbuchstabe: f = 20.04, j = 22.04, n = 24.04)
`lxc list`	Alle Container mit Status
`lxc exec containername -- bash`	In einen Container wechseln (`exit` = zurück)
`lxc restart containername`	Nur diesen Container neu starten
`lxc config set containername boot.autostart true`	Container startet mit dem V-Server
`lxc delete containername --force`	Container löschen

Regeln für `lxc launch quelle:version name` - `debian:` gibt es nicht. Debian immer über `images:debian/13`, Ubuntu immer über `ubuntu:24.04`. - `/cloud` hinter einem Image bedeutet „mit cloud-init“, für die Anleitung nimmst du die Version ohne `/cloud`. - Containername: Buchstaben, Ziffern und `-`, mit einem Buchstaben beginnen, kein Punkt, kein Unterstrich, höchstens 63 Zeichen.

3. EOTK installieren (gleicher Ablauf in allen drei Containern)

Grundregel: **als root bauen, ganz am Ende die Rechte auf ax setzen, dann als ax starten.** Für `bauskript.sh` den Namen aus der Tabelle in Abschnitt 1 einsetzen.

Nr.	Befehl	Wer / wo
1	`lxc launch …` (siehe Abschnitt 1), dann `lxc exec containername -- bash`	V-Server
2	`apt update && apt full-upgrade -y`	root im Container

Nr.	Befehl	Wer / wo
3	`apt install -y nano curl git cron tor` und `systemctl disable --now tor`	root
4	`timedatectl set-timezone Europe/Berlin`	root
5	`adduser ax`	root
6	`cd /home/ax && git clone https://github.com/alecmuffett/eotk.git`	root
7	angepasste `lib.sh` und `bauskript.sh` nach `/home/ax/eotk/opt.d/` hochladen	root (SFTP)
8	`cd /home/ax/eotk/opt.d && chmod 700 lib.sh bauskript.sh`	root
9	`./bauskript.sh 2>&1 | tee /root/build.log`	root
10	`ls -l /home/ax/eotk/opt.d/nginx /home/ax/eotk/opt.d/tor` (beide müssen da sein)	root
11	`secrets.d` und `ax.conf` (bzw. `ax.tconf`) nach `/home/ax/eotk/` hochladen	root (SFTP)
12	`chown -R ax:ax /home/ax/eotk && chmod 700 /home/ax/eotk/secrets.d`	root
13	`find /home/ax/eotk ! -user ax` (keine Ausgabe = alles richtig)	root
14	`su - ax`, dann `cd ~/eotk && ./eotk version`	ax
15	`./eotk config ax.conf && ./eotk syntax ax`	ax
16	`./eotk start ax && ./eotk status ax`	ax
17	`(crontab -l 2>/dev/null; echo "@reboot cd ~/eotk && ./eotk start ax") | crontab -`	ax
18	`exit`, `exit`, dann `lxc config set containername boot.autostart true`	V-Server
19	`lxc restart containername`, nach einer Minute `lxc exec containername -- su - ax -c "cd ~/eotk && ./eotk status ax"` (Autostart-Test)	V-Server
20	`lxc snapshot containername eotk-laeuft`	V-Server

Bedienung danach

```
su - ax -c "cd ~/eotk && ./eotk restart ax"                          # als root im Contai
ner
lxc exec containername -- su - ax -c "cd ~/eotk && ./eotk status ax" # vom V-Server
```

Statt `status` gehen auch `start`, `stop`, `restart` und `version`.

4. eotk-Konfiguration für das Projekt ax

ax.conf (bei vorhandenen Schlüsseln in `secrets.d`)

```
set project ax
set force_https 0
set preserve_csv save,xxx\\.com,i,xxx.com wwwsave,www\\.xxx\\.com,i,www.xxx.com
set nginx_cache_seconds 0
set nginx_resolver 9.9.9.9 1.1.1.1 ipv6=off
hardmap onion3-adresse xxx.com
```

Zeile	Bedeutung
`set project ax`	Projektname, steht auch in `./eotk start ax`
`set force_https 0`	Tor-Besucher werden nicht auf https umgeleitet
`set preserve_csv …`	Domain und E-Mail im sichtbaren Text (davor Leerzeichen, `>` oder `@`) bleiben normal. Links werden weiter auf .onion umgeschrieben.
`set nginx_cache_seconds 0`	kein Zwischenspeicher, Änderungen an der Website sind sofort sichtbar
`set nginx_resolver …`	DNS-Server, mit denen nginx die Domain auflöst
`hardmap onion3-adresse xxx.com`	Onion-Adresse (ohne `.onion`) zur Hauptdomain. Alle `set`-Zeilen stehen davor.

tconf und conf

- Für einen **neuen** Schlüssel schreibst du in eine `ax.tconf` die Zeile `hardmap %NEW_V3_ONION% xxx.com`. `./eotk config ax.tconf` erzeugt dann den Schlüssel in `secrets.d` und schreibt die fertige **`ax.conf`**.
- Gibt es die `ax.conf` schon, wird die tconf **nicht mehr gelesen** („using existing ax.conf“). Ab dann änderst du nur noch die `ax.conf`.
- Die `ax.conf` nie löschen, solange in der tconf `%NEW_V3_ONION%` steht. Sonst gibt es beim nächsten `config` eine neue Adresse.
- Die vollständige Vorlage mit allen Einstellungen liegt in `/home/ax/eotk/demo.d/example.tconf`.

Erkenntnisse aus den Tests

- `www mail` hinter der Domain wirkt laut eotk-Quellcode nur auf das selbst erstellte SSL-Zertifikat, nicht auf das Umschreiben.
- Zeigt die hardmap auf eine andere Domain (zum Beispiel einen Spiegel), wird die Hauptdomain im Text nicht umgeschrieben. Das sieht dann aus wie „ausgeblendet“.
- Kostenlose Spiegel mit Bot-Schutz liefern „502 Bad Gateway“. Die hardmap muss auf die Hauptdomain zeigen.
- Mit `preserve_csv` ist eine geänderte `nginx.conf` nicht mehr nötig. Sie würde ohnehin bei jedem `config` überschrieben.

5. Änderungen an der lib.sh

Die Original-`lib.sh` aus `git clone` (eotk 1.5.0) baut **Tor 0.4.5.8** (2021) und **openresty 1.15.8.3** (2020). Tor 0.4.5.8 wird vom Tor-Netz nicht mehr angenommen und baut nicht mit

OpenSSL 3. Tor stellt außerdem keine `.tar.gz.asc` mehr bereit, sondern unterschreibt eine Prüfsummendatei `.tar.gz.sha256sum`.

5.1 Debian 13

Stelle	Original (git clone)	Angepasst
openresty `tool_version` (Zeile 57)	1.15.8.3	**1.31.8.1**
openresty-Schlüssel (Zeile 58)	25451EB0…	unverändert
Tor `tool_version` (Zeile 92)	0.4.5.8	**0.4.9.15**
Tor-Schlüssel (Zeile 96)	2 alte Kurz-IDs	3 aktuelle Fingerabdrücke
Tor-Dateien	`….tar.gz` + `.tar.gz.asc`	`….tar.gz` + `.tar.gz.sha256sum` (`tool_checksum_url`) + `.sha256sum.asc`
Prüfung	`SetupForBuild`: `gpg --verify` Archiv	neue Funktion **`SetupForBuildWithChecksum`**: `gpg --verify` Prüfsummendatei, danach `sha256sum -c`
`CustomiseVars` / `BuildAndCleanup`	–	Prüfsummendatei wird ermittelt und nach dem Bauen gelöscht
PCRE1 (`pcre-8.45`)	–	`FetchPcreSource` / `CleanupPcreSource` bleiben nur als Reserve für openresty 1.15.8.3

5.2 Ubuntu 20.04 und 24.04 (gleiche lib.sh)

Stelle	Original (git clone)	Angepasst
openresty `tool_version` (Zeile 55)	1.15.8.3	**1.31.8.1**
openresty-Schlüssel (Zeile 57)	25451EB0…	unverändert
Tor `tool_version` (Zeile 101)	0.4.5.8	**0.4.9.15**
Tor-Schlüssel (Zeile 103)	2 alte Kurz-IDs	4 aktuelle Fingerabdrücke
Tor-Dateien	`….tar.gz` + `.tar.gz.asc`	zusätzlich `tool_sum_url` (`.tar.gz.sha256sum`), Signatur `.sha256sum.asc`
`SetupForBuild`	`gpg --verify` Archiv, Abbruch wenn Schlüssel nicht ladbar	bei Tor: `gpg --verify` Prüfsummendatei und `sha256sum -c`; Schlüssel laden über `hkps` ohne Abbruch; `curl -f` bricht bei fehlender Datei ab

Stelle	Original (git clone)	Angepasst
`CustomiseVars` / `BuildAndCleanup`	–	Prüfsummendatei `tool_sum` wird ermittelt und nach dem Bauen gelöscht
`SetupOpenRestyVars`	–	`tool_sum_url=""` (openresty wird wie bisher über `.asc` geprüft)

6. Änderungen am Bauskript

Ab nginx 1.21.5 wird PCRE2 benutzt, deshalb überall `libpcre2-dev` statt `libpcre3-dev`. `git` braucht das Zusatzmodul `ngx_http_substitutions_filter_module`, das die Adressen umschreibt. Als root installieren die Skripte die Pakete direkt mit `apt`, sonst über `sudo`.

6.1 Debian 13: `debian-13-trixie.sh`

Stelle	Vorlage (`build-ubuntu-20.04.sh`)	`debian-13-trixie.sh`
PCRE	`libpcre3-dev` (gibt es unter Debian 13 nicht)	**`libpcre2-dev`**
git	fehlt	**neu**
libssl1.1	ja	entfernt
Paketinstallation	immer `sudo apt`	als root direkt `apt`, sonst `sudo`
Tor bauen	`SetupForBuild`	**`SetupForBuildWithChecksum`**
PCRE1 aus dem Quellcode	–	`FetchPcreSource`, `--with-pcre=../pcre-8.45` und `CleanupPcreSource` wieder entfernt (nur für openresty 1.15.8.3 nötig)

6.2 Ubuntu 20.04: `build-ubuntu-20.04.sh`

Stelle	Original	Angepasst
PCRE	`libpcre3-dev` (PCRE1)	**`libpcre2-dev`**
git	fehlt	**neu**
libssl1.1	ja	bleibt (unter 20.04 vorhanden, aber nicht nötig)
Paketinstallation	immer `sudo apt`	als root direkt `apt`, sonst `sudo`

6.3 Ubuntu 24.04: `build-ubuntu-24.04.sh` (neu)

Stelle	Vorlage (`build-ubuntu-20.04.sh`)	`build-ubuntu-24.04.sh`
PCRE	`libpcre3-dev` (PCRE1)	**`libpcre2-dev`**
git	fehlt	**neu**

Stelle	Vorlage (`build-ubuntu-20.04.sh`)	`build-ubuntu-24.04.sh`
libssl1.1	ja	**entfernt**, gibt es ab 24.04 nicht mehr (`apt` würde abbrechen)
Paketinstallation	immer `sudo apt`	als root direkt `apt`, sonst `sudo`

Im eotk-Projekt gibt es kein Bauskript für 24.04, deshalb ist diese Datei neu. Sie funktioniert auch unter 20.04.

7. Versionen später von Hand ändern

Wo die Versionen stehen

System	lib.sh nginx	Schlüssel nginx	lib.sh Tor	Schlüssel Tor	Bauskript
Debian 13	Zeile **57**	Zeile 58	Zeile **92**	Zeile 96	`debian-13-trixie.sh`
Ubuntu 20.04	Zeile **55**	Zeile 57	Zeile **101**	Zeile 103	`build-ubuntu-20.04.sh`
Ubuntu 24.04	Zeile **55**	Zeile 57	Zeile **101**	Zeile 103	`build-ubuntu-24.04.sh`

Wo die neuesten Versionen stehen

Programm	Adresse
nginx (openresty)	https://openresty.org/download/ (höchste Nummer ohne `rc`)
Tor	https://dist.torproject.org/ (höchste Nummer ohne `alpha`)
Tor Sicherheits-Updates	https://forum.torproject.org/c/news/11

7.1 Debian 13 (`debian13ax`)

1. `lxc snapshot debian13ax vor-update` (V-Server)
2. `lxc exec debian13ax -- bash`, dann `apt update && apt full-upgrade -y`
3. `grep -n 'tool_version' /home/ax/eotk/opt.d/lib.sh`
4. `nano +57 /home/ax/eotk/opt.d/lib.sh` (nginx) bzw. `nano +92 …` (Tor), Nummer ändern, Strg+O, Enter, Strg+X
5. `cd /home/ax/eotk/opt.d && ./debian-13-trixie.sh 2>&1 | tee /root/update.log`
6. `chown -R ax:ax /home/ax/eotk`
7. `su - ax -c "cd ~/eotk && ./eotk version && ./eotk syntax ax && ./eotk restart ax"`

7.2 Ubuntu 20.04 (`ubuntu2004ax`)

1. `lxc snapshot ubuntu2004ax vor-update` (V-Server)
2. `lxc exec ubuntu2004ax -- bash`, dann `apt update && apt full-upgrade -y`
3. `grep -n 'tool_version' /home/ax/eotk/opt.d/lib.sh`

4. `nano +55 /home/ax/eotk/opt.d/lib.sh` (nginx) bzw. `nano +101 …` (Tor), Nummer ändern, Strg+O, Enter, Strg+X
5. `cd /home/ax/eotk/opt.d && ./build-ubuntu-20.04.sh 2>&1 | tee /root/update.log`
6. `chown -R ax:ax /home/ax/eotk`
7. `su - ax -c "cd ~/eotk && ./eotk version && ./eotk syntax ax && ./eotk restart ax"`

7.3 Ubuntu 24.04 (`ubuntu2404ax`)

1. `lxc snapshot ubuntu2404ax vor-update` (V-Server)
2. `lxc exec ubuntu2404ax -- bash` , dann `apt update && apt full-upgrade -y`
3. `grep -n 'tool_version' /home/ax/eotk/opt.d/lib.sh`
4. `nano +55 /home/ax/eotk/opt.d/lib.sh` (nginx) bzw. `nano +101 …` (Tor), Nummer ändern, Strg+O, Enter, Strg+X
5. `cd /home/ax/eotk/opt.d && ./build-ubuntu-24.04.sh 2>&1 | tee /root/update.log`
6. `chown -R ax:ax /home/ax/eotk`
7. `su - ax -c "cd ~/eotk && ./eotk version && ./eotk syntax ax && ./eotk restart ax"`

Hinweise für alle drei

- Das Bauskript baut immer nginx **und** Tor neu, auch wenn sich nur eine Nummer geändert hat.
- `apt full-upgrade` bringt OpenSSL, libevent, zlib und PCRE2 auf Stand. Danach eotk immer neu starten.
- Bei großem nginx-Versionssprung vorher `rm -rf /home/ax/eotk/opt.d/openresty.d` .
- Bei „No public key“ oder „BAD signature“ den neuen Signaturschlüssel in der Schlüssel-Zeile ergänzen.
- Geht etwas schief: `lxc restore containername vor-update` .
- eotk selbst nicht mit `git pull` aktualisieren. Das Projekt wird kaum gepflegt, und `git pull` stößt mit den angepassten Dateien zusammen.
- Außerdem regelmäßig den V-Server aktualisieren (`apt full-upgrade` , bei Kernel-Update neu starten) und LXD prüfen (`snap list lxd`).

Versionen prüfen

Befehl	Zeigt
`su - ax -c "cd ~/eotk && ./eotk version"`	nginx und Tor (die Meldung zu onionbalance ist harmlos)
`/home/ax/eotk/opt.d/nginx -v` und `/home/ax/eotk/opt.d/tor --version`	installierte Versionen
`ps -o user,lstart,cmd -C tor,nginx`	laufender Benutzer und Startzeit

8. Rechtevergabe

Regel	Grund
Bauen und Pakete installieren **als root**	ax hat kein sudo („may not run sudo“ ist gewollt). Außerdem fehlt ax `/usr/sbin` im Suchpfad, dann bricht openresty mit „you need to have ldconfig in your PATH“ ab.

Regel	Grund
`chmod 700 lib.sh bauskript.sh`	Beide Dateien müssen ausführbar sein, sonst werden nginx und Tor nicht gebaut.
`chown -R ax:ax /home/ax/eotk` **am Ende** und nach jedem Hochladen	Was root baut oder hochlädt, gehört root. Nur root darf den Besitzer ändern.
`chmod 700 secrets.d`	Nur ax darf an die Onion-Schlüssel, Tor verlangt das so.
`./eotk …` **nur als ax**	Als root gehören `/tmp/nginx-*-ax` root, über Tor kommt Fehler 500. Abhilfe: stoppen, `rm -rf /tmp/nginx-*-ax`, als ax starten.
Als root `su - ax -c "…"`	Bei root zeigt `~` auf `/root`, nicht auf `/home/ax`.

chmod-Zahlen: Jede Ziffer steht für Besitzer, Gruppe und alle anderen. 4 = lesen, 2 = schreiben, 1 = ausführen.

chmod	Anzeige	Bedeutung
`700`	`rwx------`	nur der Besitzer, alles
`755`	`rwxr-xr-x`	Besitzer alles, andere lesen und ausführen
`644`	`rw-r--r--`	Besitzer lesen und schreiben, andere nur lesen
`600`	`rw-------`	nur der Besitzer lesen und schreiben

`find /home/ax/eotk ! -user ax` zeigt alle Dateien, die **nicht** ax gehören. Keine Ausgabe heißt: alles gehört ax. Kommen Zeilen, hilft `chown -R ax:ax /home/ax/eotk`.

9. Snapshots und Sicherung

Befehl (V-Server)	Zweck
`lxc snapshot containername name`	Sicherungspunkt (vorher am besten `lxc stop`)
`lxc restore containername name`	auf den Sicherungspunkt zurücksetzen
`lxc info containername`	vorhandene Snapshots
`lxc copy containername/name sicherung-tmp` und `lxc export sicherung-tmp /root/sicherung.tar.gz --instance-only` und `lxc delete sicherung-tmp`	Sicherung als Datei, auch für einen anderen Server
`lxc import /root/sicherung.tar.gz neuername`	Sicherung einspielen

Snapshots liegen auf derselben Festplatte. Für eine echte Sicherung die Datei aus `lxc export` herunterladen. Sie enthält die Onion-Schlüssel und muss sicher aufbewahrt werden.

10. Häufige Fehler

Meldung	Lösung
`lxc: command not found`	Befehl auf dem V-Server ausführen, nicht im Container
`The remote "debian" doesn't exist`	`images:debian/13` statt `debian:13`
`chmod: … Operation not permitted`	als root ausführen
`[ 1000 -eq 0 ]`, danach `sudo apt …`	Bauskript als root starten
`start.sh: nginx: not found`	`/root/build.log` prüfen und neu bauen
`cd: /root/eotk: No such file or directory`	als root `su - ax -c "…"` benutzen
Hilfe-Text von eotk erscheint	Dateiname prüfen: `ls *.conf *.tconf`, `./eotk projects`
Kein neuer Schlüssel bei `config`	`ax.conf` existiert schon oder `%NEW_V3_ONION%` fehlt
502 Bad Gateway	hardmap auf die Hauptdomain setzen
Fehler 500 über Tor	eotk wurde als root gestartet, siehe Abschnitt 8
`onionbalance: not found`	harmlos

Quellen

1. EOTK (Enterprise Onion Toolkit), Quellcode mit `opt.d/lib.sh`, `templates.d/nginx.conf.txt`, `lib.d/do-configure.pl`, `lib.d/expand-config.pl`, `demo.d/example.tconf`, `docs.d/QUICK-START.md`, `docs.d/HOW-TO-INSTALL.md`: https://github.com/alecmuffett/eotk
2. Tor Project, Quellcode prüfen (Dateinamen, Signaturschlüssel): https://support.torproject.org/little-t-tor/verify-little-t-tor
3. Tor Project, Download-Verzeichnis: https://dist.torproject.org/
4. Tor Project Forum, News (Security Release 0.4.9.15): https://forum.torproject.org/c/news/11
5. OpenResty, Download-Verzeichnis (1.31.8.1 vom 26.10.2026): https://openresty.org/download/
6. OpenResty, Ankündigung 1.31.1.1 (nginx 1.31): https://openresty.org/en/ann-1031001001.html
7. nginx, Änderungen (PCRE2 ab 1.21.5): https://nginx.org/en/CHANGES
8. LXD Dokumentation (Canonical): https://documentation.ubuntu.com/lxd/en/latest/
9. LXD, Sicherung von Instanzen (Snapshots, Export, Import): https://documentation.ubuntu.com/lxd/en/latest/howto/instances_backup/
10. Debian 13 „Trixie“: https://www.debian.org/releases/trixie/
11. Ubuntu 24.04 LTS: https://releases.ubuntu.com/24.04/
12. Ubuntu Release-Zyklus und Unterstützungszeiträume: https://ubuntu.com/about/release-cycle
13. Zeitzonen (tz database): https://en.wikipedia.org/wiki/List_of_tz_database_time_zones
14. Eigene Tests und Protokolle von Sebastian Ebner (Build-Logs, eotk version, nginx-error.log, curl-Tests über die Socket-Datei, lxc image list)