Steffen Probst

Blog

Debian - apt pinning

In diesem kleinen Beitrag arbeite ich das LUG Thema vom 06.03.2024 auf. Wir hatten dort das Thema Pinning von deb Paketen unter Debian. Eine weiterführende Dokumentation ist auch direkt auf der Debian Seite [2] zu finden.

Was ist apt

`apt` oder auch `apt-get` ist die Paketverwaltung unter Debian basierte Linux Distribution [3]

Was ist das Ziel?

Mit `apt` pinning möchte man in der Paketverwaltung bestimmte Programme bei der Installation priorisieren.
Beispiel:
Du möchtest als Standarddistribution Debian 12 bookworm verwenden, aber bestimmt Pakete aus einer anderen Version von Debian (oder einer anderen Distribution) dazu installieren.

Wichtig: Wenn du Pakete aus anderen Distributionen oder Versionen mischst, kann es zu ungewollten Seiteneffekte kommen. Daher gebe ich auf die Anleitung keine Funktionsgarantie und jeder führt diese auf eigenes Risiko durch.

Wie gehe ich vor?

- Erstelle unter `/etc/apt/source.list.d` eine `.list` mit dem entsprechenden Repo, was du verwenden willst. → Als Vorlage kann die `/etc/apt/source.list` dienen. → Kopiere diese in den Ordner `/etc/apt/source.list.d/` und bennen diese vorzugsweise um. Beispiel: `cp /etc/apt/source.list /etc/apt/source.list.d/foo.list` → Ändere die Einträge in der Datei mit einem Editor deiner Wahl auf das passende Repo ab.
- Lege unter `/etc/apt/preferences.d/` eine Datei an, in der du das Pinning konfigurierst. → Ich gebe der Datei immer voran gestellt eine Nummer und dann, was diese enthalten soll. → Beispiel: `99-debian`
- Öffne die Datei mit dem Editor Deiner Wahl und ändere die Einträge wie folgt.

99-debian

```
Package: *
Pin: release a=unstable
Pin-Priority: 500

Package: *
Pin: release a=stable
Pin-Priority: 900

Package: firefox*
Pin: release a=unstable
Pin-Priority: 1000

Package: keepass*
Pin: release a=unstable
Pin-Priority: 1000

Package: flameshot*
Pin: release a=unstable
Pin-Priority: 1000

Package: filius*
Pin: release a=unstable
Pin-Priority: 1000

Package: remmina*
Pin: release a=unstable
Pin-Priority: 1000

Package: libreoffi*
Pin: release a=unstable
Pin-Priority: 1000

Package: podman*
Pin: release a=unstable
Pin-Priority: 1000

Package: docker*
Pin: release a=unstable
Pin-Priority: 1000

Package: distrobox*
Pin: release a=unstable
Pin-Priority: 1000

Package: remmina*
Pin: release a=unstable
Pin-Priority: 1000

Package: qflipper*
Pin: release a=unstable
Pin-Priority: 1000
```

```
Package: mfcuk
Pin: release a=unstable
Pin-Priority: 1000

Package: mfoc
Pin: release a=unstable
Pin-Priority: 1000

Package: fritzing*
Pin: release a=unstable
Pin-Priority: 1000

Package: arduino*
Pin: release a=unstable
Pin-Priority: 1000
```

- Die ersten dreier Block setzen die Priorität für `sid/unstable` auf den Wert 500. Das bedeutet, dass die Pakete aus diesem Repo installiert werden, wenn Sie benötigt werden.
- Der zweite dreier Block setzte `stable` auf die Priorität 900. Dadurch liegt die Priorität für stable höher als 500 von `unstable` und dem Paket aus dem stabilen Zweck vorrangig installiert, wenn die dort Pakete mit einer niedrigeren Versionsnummer vorhanden sind.
- Die nachfolgenden Blöcke priorisieren die Pakete aus `unstable` mit einer 1000. Die Priorität 1000 sagt, dass die Pakete explizit aus `unstable` installiert werden sollen und die nötigen Abhängigkeiten zwingend nach gezogen werden sollen. Dabei prüft apt ob er Abhängigkeiten aus dem `stable` und `unstable` Zweig der Repositorys auflösen kann.

Schlusswort

Wenn jemand einen Fehler findet, bitte sendet mir diesen zu. Ich korrigiere dann den Eintrag. Falls ich noch auf etwas eingehen soll, dann bitte auch eine Info. Ansonsten findet man sehr gute Anleitungen im Netz dazu, Ich hoffe ich konnte etwas Licht ins Dunkel bringen.

2024/03/07 07:33 · SProbst
debian, linux, apt, pinning

openwrt in docker -> Das geht?

Ja. Das geht. Sogar ziemlich gut!

Was benötigt wird, ist einfach ein `rootfs.tar.gz` von openwrt. Aktuell sind die für **x86/amd64** und **arm** verfügbar. Ab Version **23.5.x** steht in dem Sinne kein **armvirt** mehr zu Verfügung. Aber der aufmerksame Leser der Release Notes findet den Hinweis das, dass entsprechende Release in **armsr/armv7** und **armsr/armv8** gewandert ist.

Als erste lädt man sich entsprechend der Architektur das `rootfs.tar.gz` herunter. Idealerweise mit einem Download wie `wget` oder `aria2c`.

Das Dockerfile sieht wie folgt aus:

Dockerfile

```
FROM scratch

ADD openwrt-23.05.2-armsr-armv8-rootfs.tar.gz /

RUN     mkdir -p /var/lock/ && opkg update && \
        mkdir -p /var/run

EXPOSE 80/tcp

USER root

CMD ["/sbin/init"]
```

Der Rest sollte plausibel sein und funktioniert wie bei anderen Dockerfiles/-containern. Das Ergebnis ist ein verdammt schmaler Container.

2023/12/16 17:50 · SProbst
docker, dockerfile, shell, bash, openwrt

post-checkout mit git

Ich pflege meinen `.ssh` Ordner mit git. Im Klartext, ich habe mehr **Branches** für unterschiedliche Szenarien. Der Hintergrund: egal, weil es geht. Dabei habe ich immer das Problem gehabt, dass die Keys nicht die korrekten Zugriffsberechtigungen hatten (`0644` auf Dateien). Nach etwas Recherche stellt sich heraus, dass im git Repo, es möglicht ist, Script nach bestimmten Positionen/Stadien automatisch ausführen zu lassen. In meinem Fall ist das `post-checkout` Script unter `$REPODIR/.git/hooks/` spannend. Kurzerhand dort eine Shellscript mit dem Namen angelegt. Ein passenden Find/While Befehl gebaut und schwups, sind die Berechtigung zukünftig korrekt. Hier das Beispiel:

post-checkout

```
#!/usr/bin/bash

#find pubkey and set file.key permissions
find $HOME/.ssh -name '*.pub' -print0  | while IFS= read -r -d '' line;
do chmod 600 $HOME/.$(echo $line | cut -f 2 -d ".") || true; done
```

2023/12/16 17:07 · SProbst
ssh, pub, key, shell, bash, git

Backup PC für 5¼ und 3.5 Floppys

Nach dem Floppy-Experiment habe ich mir einen Pentium MMX (Socket 7/i586) als Disketten Rettungssystem zusammen geschraubt. Dabei habe ich ein aktuelles Betriebssystem für den Rechner gesucht. Leider musste ich feststellen das Debian 12 erst mit einem i686 (Pentium II/Pro) funktioniert,

da entsprechend ein CPU-Befehl im Pentium MMX fehlt. Tenor: Da ich seit längerem schonmal mit netbsd befassen wollte, habe ich netbsd 9.3 probiert. Und siehe da, es bootet sogar recht schnell auf dem Rechner. Da der PC mit üppigen 256 MB SD-RAM ausgestattet ist, läuft so auch ein schmales Xorg drauf.
Daher, ein aktuelles Linux bekommt man bedauerlicherweise nicht mehr auf solch alter Hardware am Laufen. Aber ein netbsd tut im Zweifel seinen Dienst.

2023/08/23 08:29 · SProbst
floppy, pc, netbsd, linux

Retro: 5¼ Laufwerk(floppy) unter Linux

Bei unserem letzten LUG Treffen haben wir zwei 5¼ Laufwerk reaktiviert. Nachdem wir die hardwaretechnischen Themen gelöst hatten. Einen PC zu finden, der überhaupt einen 34 poligen Floppyanschluss hat(danke an Thomas), haben wir ein Debian 5.0 herausgesucht und gebootet (auch hier, danke an Thomas).

Nachdem wir erfolgreich die Einstellungen im BIOS konfiguriert haben, mussten wir nun die 5¼ Disketten formatieren. Dabei ist wichtig, das Disketten Format zu wissen. Die Wikipedia DE Seite hat ein schöne **Tabellenübersicht_der_Diskettenformate** auf der Disketten Seite.

Vorab muss man mit `mknod` entsprechend eine **blockorienntiertes** Gerät anlegen. Die **minor major** Nummer sind in der Dokumentation[4] zu `fdformat` zu finden. Dank an Crissi, er hat uns den nötigen Hinweis gegeben. **Hier sollte in jedem Fall auch die manapages zu `fdformat` und wikipedia konsultiert werden, um entsprechend das Diskettenformat zu ermitteln.**
Beispiel:

```
#Floppy 0/A: mit 1.2 MB
mknod /dev/fd0h1200 b 2 8
#Flooppy 0/A: mit 360kB
mknod /dev/fd0d360 b 2 4
```

Hintergrund ist wohl, dem Controller mit zuteilen, auf welches Diskettenformat man zugreifen möchte. Hier greift wohl nicht die Autoerkennung wie bei einem 3½ Diskettenlaufwerk, welches bei einem Testzweck adhoc funktioniert hat, ohne entsprechend die Geräte anzulegen.

Dann formatiert man entweder eine frische Diskette mit dem `fdformat` Befehl oder `mounted` die Diskette.
Beispiel:

```
# LowLevel Formatierung der Diskette
fdformat /dev/fd0h1200
```

Wichtig: Ein LowLevel Formatierung löscht alle Daten unwiederbringlich! Daher nur ausführen, wenn die Diskette frisch verwendet werden soll!
Nach der LowLevel Formatierung muss dann noch ein Dateisystem entweder mit `mkfs.minix` oder ähnlichen oder `mformat` für ein **FAT12** bzw. **FAT16** angelegt werden. ext2 wäre ggf. auch möglich würde aber ggf. zu für Speicher auf der Diskette nehmen. Hier sollte mein meines Erachtens FAT12 dem Vorzug geben. Somit bewahrt man sich auch die Kompatibilität mit älteren Betriebssystemen.

Der `mount` Befehl sieht dann entsprechend aus.

Beispiel:

```
#1.2MB Diskette einhängen/mounten
mount /dev/fd0h1200 /mnt
```

Jetzt sollte der Datenrettung nichts im Wege stehen, wenn die Diskette noch lesbar ist und/oder sie nicht vorher formatiert wurde. ;)

Nun kann man entsprechend Daten sichern. Wichtig: nach dem man sein To-do erledigt hat, klassisch wieder `unmounten`.

2023/07/06 09:44 · SProbst
retro, floppy, diskette, disk

Ältere Einträge >>

Index

1)
Namensnennung - Nicht-kommerziell - Keine Bearbeitung
2)
https://wiki.debian.org/AptConfiguration
3)
Wie Linux Mint der Ubuntu zum Beispiel.
4)
manpages > von Debian bullseye