Desktop-Anwendungen in Containern bereitstellen

Verpacker

Erkan Yanar

Das Werkzeug Subuser vereint die Vorteile von Containern mit dem Nutzen einer Paketverwaltung.

Beim Verteilen und Paketieren von Desktop-Anwendungen bringen sich neue Paketformate in Stellung. Die Grundidee: Desktop-Images werden lediglich einmal (idealerweise vom Entwickler) gebaut und sind anschließend auf unterschiedlichen Distributionen lauffähig, unter anderem weil die Applikationen alle erforderlichen Komponenten im Image mitbringen. Auf dieses Konzept setzt auch die Containerverwaltung Docker.

Desktop-Applikations-Images laufen eigenständig und sind vom Host, auf dem sie laufen, weitgehend isoliert. Sie können zudem unabhängig von ihm aktualisiert werden. Gerade Anwendungsentwickler haben es damit einfacher, da ein Paketformat für alle Distributionen ausreicht und sie auch beim Weiterentwickeln der Applikation nicht auf unterschiedliche oder veraltete Bibliotheken auf den Hostsystemen Rücksicht nehmen.

Anwendungen erhalten dadurch einen unabhängigen Lebenszyklus und lassen sich kontinuierlich aktualisieren, ohne den Host zu beeinflussen. Distributionen könnten sich auf Kernfunktionen beschränken und solche Paketformate für Anwendungen benutzen (wie CoreOS).

Es gibt drei bekannte Formate für Desktop-Applikations-Images:
– Canonicals Snappy soll Ubuntu und Ubuntu Phone harmonisieren und auch auf anderen Distributionen laufen.
– Flatpak wird laut Red-Hat-Entwickler Adam Williamson von der Community entwickelt. Es definiert für seine Pakete Abhängigkeiten zu Libraries des Systems und delegiert damit die Verantwortung dafür wieder an das Hostsystem, womit die erwähnten Vorteile verloren gehen.
– AppImage besticht durch seine Einfachheit. AppImages sind ISO-Abbilder der Applikation mit einer Linux-Runtime im Header. Man nutzt sie nach dem Herunterladen als ELF-Executable.

Subuser baut lokale Container

Das Open-Source-Projekt Subuser, ein Wrapper um Docker-Container, stellt ebenfalls ein Paketformat für Anwendungen vor und nutzt Xpra zum Isolieren von X11-Applikationen. Jede Subuser-Applikation läuft in einem eigenen Docker-Container. Sie teilt sich einen X11-Socket mit dem Xpra-Server. Dieser kommuniziert mit dem Xpra-Client (der wie der Server in einem Container läuft), der sich den X11-Socket mit dem Host teilt.

Hat man Docker und Git installiert und Letzteres konfiguriert, stellen folgende Schritte das Tool */usr/local/bin/subuser* bereit:

```
$ sudo usermod -a -G docker $USER
$ sudo apt install -y python3-pip
$ sudo pip3 install subuser
```

Ein *subuser list available* checkt per Git das Default-Repository nach *~/.subuser/repositories/default* aus und zeigt verfügbare Anwendungen, etwa *vlc@default* oder *xterm@default*. Xterm installiert man mit

```
subuser subuser add xterm xterm@default
```

Hierbei prüft die Software die Datei *permissions.json* (Listing 1) und definiert daraufhin, welches Kommando im Container ausgeführt werden soll. Zu einem Subuser-Paket gehört außerdem ein Dockerfile *SubuserImagefile* im Verzeichnis *~/.subuser/repositories/default/xterm/image/*. Wobei Subuser für Letzteres mit *FROM-SUBUSER-IMAGE* ein alternatives *FROM* anbietet, das ein Subuser-Image als Quelle nutzt.

Nach dem Bauen des *xterm*-Containers auf dem Host startet man diesen mit *subuser run xterm*. Das Kommando *docker ps* enthüllt, dass drei Container gestartet wurden: der *xterm*- sowie die beiden *Xpra*-Container. Das Kommando *xterm* startet nun die Anwendung.

Da das Ergebnis der Dockerfiles nicht deterministisch ist, unterscheiden sich die Subuser-Images voneinander: Ein *FROM debian* im *SubuserImagefile* bedeutet stets etwas anderes. Daher ist die Software vor allem als Framework zum Isolieren der Pakete vom restlichen System zu sehen, kann jedoch nicht garantieren, dass diese auf allen Systemen gleichermaßen laufen (was in der Docker-Welt die aus den Dockerfiles erstellten Images erfüllen).

Ein klassisches Dockerfile dient dem Erstellen eines Image, erst dieses ist eindeutig und wird idealerweise in einer Registry verwaltet. So ein binäres Repository fehlt Subuser. Dieser Mangel wäre mit einem Kniff zu lösen, indem das *SubuserImagefile* lediglich aus einem *FROM* mit fester Referenz auf ein vom Entwickler erstelltes Image (via Digest) bestünde. Nur ist das zumindest im installierten Default-Repository nicht umgesetzt.

Fazit

Das Verwenden von Docker unter Sicherheitsgesichtspunkten ist kritisch zu sehen, steht doch dem *$USER* mit dem Zugriff auf den Docker-Daemon das ganze System zur Verfügung. Davon abgesehen ist Subuser ein interessantes Werkzeug, das Containertechnik mit einem offenen und portierbaren Paketkonzept vereint. (tiw)

Listing: Berechtigungen des Pakets *xterm*

```
$ cat ~/.subuser/repositories/default/xterm/permissions.json
{
 "description"               : "A trivial terminal emulator"
 ,"maintainer"               : "Timothy Hobbs <timothyhobbs (at) seznam dot cz>"
 ,"executable"               : "/usr/bin/xterm"
 ,"gui"                      : {"clipboard":true,"cursors":true}
 ,"access-working-directory" : true
 ,"allow-network-access"     : true
 ,"basic-common-permissions" : true
}
```

Erkan Yanar

ist Diplom-Soziologe und Open-Source-Enthusiast.

Alle Links: www.ix.de/ix1608135