Contents

1 Building a new QubesOS package 2
1.1 References 2
1.2 Goal 2
1.3 The software 2
1.4 Packaging 3
1.5 Building 4

1. BUILDING A NEW QUBESOS PACKAGE

1.1 References

- QubesOS minisubmit video (QubesOS - 3mdeb), by Frédéric Pierret
- qubes-skeleton

1.2 Goal

The goal will be to create simple Qube software that do the following:

- Provide a binary named "qvm-druide-antidote"
- The binary will create a new disposableVM, and send file to it
- The disposableVM will open the software "Antidote" from "Druide" '(it is a proprietary spellchecker) on the file it received
- The user fix the spelling issue and close "Antidote"
- The corrected file is sent back to the original VM

We will only explain the "packaging" and "building" part. My repository: qubes-app-linux-druide-antidote

1.3 The software

To be run on the "client" vm:

```
#!/bin/bash
FILE=${1?file required}
TMPFILE=$(mktemp --tmpdir druide-antidote-XXXXXXXX)
BACKUP=$(mktemp --tmpdir druide-antidote-backup-XXXXXXXX)
cp "$FILE" "$BACKUP"
echo "Backup file is: $BACKUP" >&2

cat "$FILE"
exec >&-
cat > "$TMPFILE"
[ -s "$TMPFILE" ] && mv "$TMPFILE" "$FILE"
```

To be run on the "server" vm:

```
#!/bin/bash

FILE=$(mktemp --tmpdir druide-antidote-XXXXXXXX)

cat > "$FILE"
/opt/Druide/Antidote10/Application/Bin/Antidote10 -f "$FILE" 2>
↪  /dev/null > /dev/null
```

```
cat "$FILE"

rm -f "$FILE"
```

1.4 Packaging

I want it to be:

- compatible with the following templates:
  - Fedora
  - Debian
  - Archlinux
- easily reusable for other software that use this kind of client-server RPC

For the easily reusable part, lots of packaging variables come from the README.md. Read the file "vars.sh" and "archlinux/PKGBUILD" to understand better. PKGBUILD:

```
pkgname=(qubes-$(./vars.sh name)-vm)
pkgver=$(cat version)
pkgrel=$(cat rel)
arch=(x86_64)
pkgdesc=$(./vars.sh description)
url=$(git remote get-url origin)
license=(GPL3)
depends=()
makedepends=(pandoc git)

build() {
 ln -s "$srcdir"/../ "$srcdir/src"
}
check(){
 src/tests/all
}
package() {
 cd src
 make install-vm DESTDIR="$pkgdir/"
}
```

vars.sh:

```
#!/bin/bash

name(){
 head -n 1 README.md | cut -d ' ' -f 2 | cut -d '(' -f 1 | cut -c5-
 ↪  | tr [:upper:] [:lower:]
}
```

```
description(){
 awk '/DESCRIPTION/,!//' ./README.md | tail -n +3 | awk
 ↪  '1;/=======/{exit}' | head -n -3
}

variable=${1:?Variable name is required. name, description or
↪  camelname}
case "$variable" in
 "name"):
  name
  ;;
 "description"):
  description
  ;;
 "camelname"):
  name | sed 's/^\([A-Za-z0-9]\)\([A-Za-z0-9]\+\)-\([A-Za-z0-9]\)\([
  ↪  A-Za-z0-9]\+\)$/\U\1\L\2\U\3\L\4/g'
  ;;
 "summary"):
  description | head -n 1
  ;;
 *):
  echo "This variable doesn't exist" >&2
  exit 1
 ;;
esac
```

For the building flow, it start from the template packaging (./debian/, ./archlinux/, ./rpm/), then the build script of the template will call the Makefile

1.5 Building

I learned everything in this section from the minisubmit video I referenced before.

We need to build this package using the standard qubes builder. So the most important things are in the next file named "app-builder.conf".

- How to build a QubesOs package that doesn't come from the official repository
- How to build only that package and not everything related to QubesOS

app-builder.conf :

```
# vim: ft=make ts=4 sw=4

GIT_BASEURL ?= https://github.com
GIT_PREFIX ?= QubesOS/qubes-
NO_SIGN ?= 1
#BRANCH ?= release4.0
```

```
BACKEND_VMM=xen

DIST_DOM0 ?= fc32
DISTS_VM ?= fc33+minimal buster+minimal bullseye+minimal

COMPONENTS ?= \
app-linux-druide-antidote \
builder \
builder-debian \
builder-rpm

INSECURE_SKIP_CHECKING = app-linux-druide-antidote
GIT_URL_app_linux_druide_antidote =
↪  https://github.com/neowutran/qubes-app-linux-druide-antidote
BRANCH_app_linux_druide_antidote = master

BUILDER_PLUGINS = builder-rpm builder-debian
BUILDER_PLUGINS += mgmt-salt

DISTS_VM += archlinux+minimal
COMPONENTS += builder-archlinux
BUILDER_PLUGINS += builder-archlinux
```

app-builder.sh

```
#!/bin/bash
BASE="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 &&
↪  pwd )"
directory=$BASE/qubes-builder
sudo rm -Rf "$directory"
sudo dnf install wget make git qubes-gpg-split
git clone "https://github.com/QubesOS/qubes-builder.git"

key1=$(curl -s
↪  https://keys.qubes-os.org/keys/qubes-master-signing-key.asc |
↪  sha512sum | cut -d " " -f 1)
key2=$(sha512sum /usr/share/qubes/qubes-master-key.asc | cut -d " "
↪  -f 1)

if [ "$key1" != "$key2" ]; then
echo "CRITICAL SECURITY FAILURE: qubes master signing key is not the
↪  same on different source (local and official qubes os website)"
↪  >&2
exit 1
fi
```

```
gpg --import /usr/share/qubes/qubes-master-key.asc
echo "Check the key, if it is good for you, set the trust to 5 and
↪  exit"
echo "fpr" | gpg --edit-key
↪  0x427F11FD0FAA4B080123F01CDDFA1A3E36879494

wget https://keys.qubes-os.org/keys/qubes-developers-keys.asc
gpg --import qubes-developers-keys.asc
rm qubes-developers-keys.asc

commit_data=$(cd "$directory" && git tag -v $(git describe) 2>&1 |
↪  grep "gpg: ")
echo "$commit_data"
echo "$commit_data"  | tail -n 1 | grep "Good signature from "
success=$?

if (( $success == 1 )); then
echo "CRITICAL SECURITY FAILURE: last commit from qubes-builder is
↪  not signed with an approved gpg key" >&2
exit 1
fi

echo "Does this seems good to you ?"
read trash

cp ./app-builder.conf $directory/builder.conf

cd "$directory"
make get-sources

make install-deps
make qubes-vm
```