

Modularizacija SquashFS Linux datotecnog sistema

by

Dajan Brackovic

Submitted to the Odsjek za racunarstvo i informatiku
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

ELEKTROTEHNICKI FAKULTET U SARAJEVU

June 2020

© Dajan Brackovic, MMXX. All rights reserved.

The author hereby grants to ETF permission to reproduce and to
distribute publicly paper and electronic copies of this thesis document
in whole or in part in any medium now known or hereafter created.

Author

Odsjek za racunarstvo i informatiku

May 18, 2020

Certified by

Samir Ribic

dr. sc.

Thesis Supervisor

Accepted by

Jasmin Velagic

Dean of the Faculty of Electrical Engineering Sarajevo

Contents

Abstract	3
Uvod	4
Sistemske zahtjevi	5
Priprema radnog okruzenja	6
SquashFS paket	6
Modul NodeJS	7
Rezultat Modul NodeJS	11
Modul MongoDB	14
Modul Chrome	19

Abstract

This work is describing the process of modularization of Linux SquashFS filesystem. Modularization is performed by manual customization of packages in the live system, then extracting that customized system as a separate module. This thesis will show how to create 3 separate modules out of the same base image, which will be the ubuntu-18.04.4-desktop-amd64.iso. We will be using the SquashFS tools to make the modifications inside the ubuntu-18.04.4-desktop-amd64.iso image

Uvod

Zasto uopce mijenjati instalacioni iso image operativnog sistema? Postoji nekoliko razloga:

1. Da bismo napravili svoju distribuciju mijenjajuci postojecu iso datoteku
2. Da bismo predstavili odredjenu aplikaciju
3. Radi lokalizacije na odredjeni jezik
4. Da bismo uklonili odredjene softverske pakete
5. S ciljem dodavanja novih softverskih paketa
6. U svrhu azuriranja softverkih paketa
7. Radi mijenjanja systemske konfiguracije kao sto su teme, ikone, fontovi, pozadina...

Najlaksi nacin modifikacije iso image-a baziranih na Ubuntu distribuciji je koriscenjem "Ubuntu Customization Kit" alata. Medjutim ovaj rad ce obuhvatiti drukciju princip, manualni.

Svaki od modula koji su kreirani su bazirani na istom base image-u, ubuntu-18.04.4-desktop-amd64.iso. Modifikacijom istog dobit cemo tri modula:

- 1 Modul NodeJS - ubuntu-with-nodejs-18.04-amd64.iso
- 2 Modul MongoDB - ubuntu-with-mognoadb-18.04-amd64.iso
- 3 Modul Java - ubuntu-with-java-18.04-amd64.iso

Sistemske zahtjevi

Da biste se uputili u ovaj zadatak postoji prije svega nekoliko hardverskih minimuma koje vaša radna mašina treba da ispunjava:

1. Najmanje 5GB slobodnog prostora na disku, mada poželjno bi bilo mnogo više od 5GB, pogotovo ukoliko pravite različitih modula.
2. Najmanje 512MB RAM memorije i 1GB alocirane swap memorije.
3. Linux kernel sa squashfs podrskom.
4. QEMU/KVM || VirtualBox || VMWare - bilo koji od ova 3 alata za testiranje kreiranih modula.
5. genisoimage - paket za generisanje novog iso image-a

Priprema radnog okruzenja

Instalirati squashfs-tools i genisoimage:

```
1 sudo apt-get install squashfs-tools genisoimage
```

SquashFS paket

Paket squashfs-tools implementira 2 funkcije koje se koriste u ovom radu a koje pruža SquashFS <http://tldp.org/HOWTO/SquashFS-HOWTO/whatis.html>. Radi se o funkcijama **mksquashfs** i **unsquashfs**. Prva od navedenih koristi se za kreiranje squashfs datoteke, dok se druga funkcija koristi za raspakivanje kompresovane squashfs datoteke.

SquashFS je moguće instalirati kao dodatak na linux jezgro. Prema tome moguće ga je instalirati na različite linux distribucije. Za Debian distribuciju njegov naziv je squashfs-tools.

Modul NodeJS

NodeJS Modul ce biti kreiran od istog baznog modula kao i svi ostali moduli. To je ubuntu-18.04.4-desktop-amd64.iso datoteka:

```
1 mkdir ~/squashfs/livedtmp
2 mkdir ~/squashfs/livedtmp/isoimgs
3 mv ~/Downloads/ubuntu-18.04.4-desktop-amd64.iso ~/squashfs/livedtmp/isoimgs
4 cd ~/squashfs/livedtmp
```

Napraviti mnt direktorij unutar livedtmp direktorija u koji ce biti mount-an ubuntu-18.04.4-desktop-amd64.iso image:

```
1 mkdir mnt
2 sudo mount -o loop ./isoimgs/ubuntu-18.04.4-desktop-amd64.iso mnt
```

Napraviti direktorij extract-cd u kojeg cemo kopirati mnt direktorij izostavljajuci filesystem.squashfs datoteku unutar /casper direktorija:

```
1 mkdir extract-cd
2 sudo rsync --exclude=/casper/filesystem.squashfs -a mnt/ extract-cd
```

Napraviti direktorij za modul nodejs i kopirati u njega extract-cd direktorij:

```
1 mkdir modul-nodejs
2 sudo rsync -a extract-cd/ modul-nodejs
```

U ovom trenutku cemo upotrijebiti unsquashfs funkciju iz squashfs-tools paketa. Te cemo kopirati raspakovani squashfs-root direktorij u edit direktorij. Ovaj edit direktorij cemo kasnije koristiti da unutar njega instaliramo nodejs pakete:

```
1 sudo unsquashfs mnt/casper/filesystem.squashfs
2 sudo mv squashfs-root/ edit
```

Da bi imali mrežnu konekciju unutar edit direktorija jedno rješenje je kopirati /run direktorij unutar edit direktorija. Najbolje manuelno popuniti resolv.conf unutar edit direktorija, isto i za etc/hosts datoteku:

```
1 sudo cp /etc/resolv.conf edit/etc/
2 sudo mount -o bind /run/ edit/run
```

Kopirati i hosts direktorij/:

```
1 sudo cp /etc/hosts edit/etc/
```

Namjestiti edit/dev direktorij kopirajući /dev/ direktorij sa hosta, zatim chroot u edit direktorij. Obaviti mount instrukcije navedene ispod. Ukoliko korisnik odluči da obriše edit direktorij iz nekog razloga, bilo bi potrebno uraditi unmount edit direktorija da sistem domaćin ne bi postao neupotrebljiv:

```
1 sudo mount --bind /dev/ edit/dev
2 sudo chroot edit
3 mount -t proc none /proc
4 mount -t sysfs none /sys
5 mount -t devpts none /dev/pts
```

Takodje potrebno je izvršiti sljedeće komande da bi se izbjegli problemi sa lokalizacijom:

```
1 export HOME=/root
2 export LC_ALL=C
```

Za ispis svih instaliranih paketa:

```
1 dpkg-query -W --showformat='${Installed-Size}\t${Package}\n' | sort -nr | less
```

Instalacija nodejs paketa:

```
1 apt-get update
```



```
2 apt-get install curl
3 curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash -
4 apt-get install -y nodejs
```

Nakon zavrsetka instalacije izvršiti unutar chroot:

```
1 apt-get clean
2 rm -rf /tmp/* ~/.bash_history
3 rm -rf /tmp/* ~/.bashrc
4 rm /var/lib/dbus/machine-id
5 rm /sbin/initctl
6 dpkg-divert --rename --remove /sbin/initctl
7 umount /proc || umount -lf /proc
8 umount /sys
9 umount /dev/pts
10 umount /dev
11 exit
```

Ponovno generisati filesystem.manifest:

```
1 chmod +w extract-cd/casper/filesystem.manifest
2 sudo su
3 chroot edit dpkg-query -W --showformat='${Package} ${Version}\n' > extract-cd/
  casper/filesystem.manifest
4 exit
5 sudo cp extract-cd/casper/filesystem.manifest extract-cd/casper/filesystem.manifest-
  desktop
6 sudo sed -i '/ubiquity/d' extract-cd/casper/filesystem.manifest-desktop
7 sudo sed -i '/casper/d' extract-cd/casper/filesystem.manifest-desktop
```

Sada ćemo upotrijebiti drugu funkciju iz squashfs-tools, a to je mksquashfs. S tom funkcijom ćemo kompresovati edit direktorij u novu filesystem.squashfs datoteku. U kodu ispod je potrebno izvršiti komandu iz linije 1 i jednu od preostale 3, pri čemu prva (komanda na liniji 2) daje najslabiju kompresiju, ali je najbrža. Druga komanda

se duze izvrsava ali je veci procenat kompresije u odnosu na prvu komandu. Dok je kod trece komande procenat kompresije najveći, a vrijeme izvršenja najduže:

```
1 sudo rm extract-cd/casper/filesystem.squashfs
2 sudo mksquashfs edit extract-cd/casper/filesystem.squashfs -no lzma
3 sudo mksquashfs edit extract-cd/casper/filesystem.squashfs -b 1048576
4 sudo mksquashfs edit extract-cd/casper/filesystem.squashfs -comp xz -e edit/boot
```

Naredni korak je da azuriramo filesystem.size datoteku:

```
1 sudo su
2 printf $(du -sx --block-size=1 edit | cut -f1) > extract-cd/casper/filesystem.size
3 exit
```

Nakon toga upisati naziv image-a unutar README.diskdefines. Upisati 'Ubuntu with NodeJS 18.04.4 LTS "Bionic Beaver" - Release amd64' u polje DISKNAME:

```
1 cd extract-cd
2 sudo rm md5sum.txt
3 find -type f -print0 | sudo xargs -0 md5sum | grep -v isolinux/boot.cat | sudo tee md5sum
    .txt
```

Azurirati md5sum.txt datoteku:

```
1 sudo gedit extract-cd/README.diskdefines
```

Napokon mozemo napraviti iso image koji ce da sadrzi NodeJS modul. Za ovu operaciju koristimo funkciju genisoimage. Neke linux distribucije nude mkisofs funkciju. Tako da ukoliko ne radi jedna trebala bi druga:

```
1 sudo genisoimage -D -r -V "$IMAGE_NAME" -cache-inodes -J -l -b isolinux/isolinux.
    bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o ../
    ubuntu-with-nodejs-18.04-amd64.iso .
```

Sada cemo napraviti virtuelni hard disk pomocu qemu-img komande da bismo pokrenuli na njemu nas novi modul NodeJS Ubuntu.

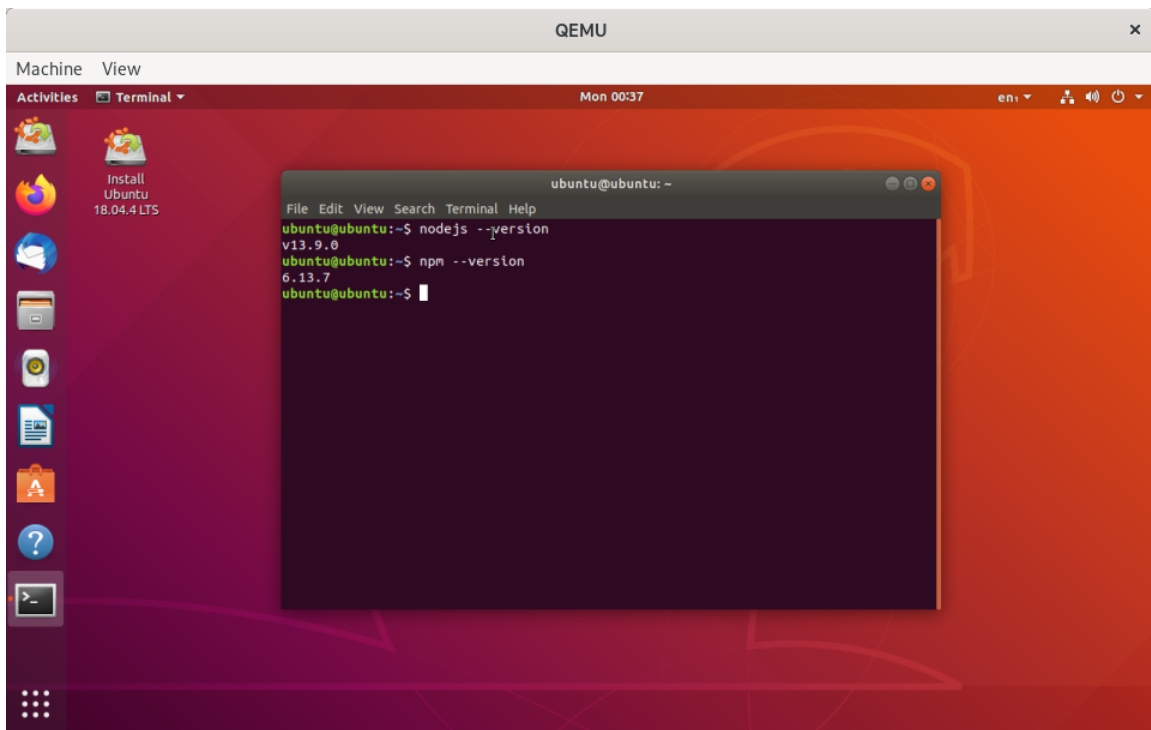
```
1 cd ~
```

```
2 qemu-img create ubuntu-nodejs.img 5G
```

Pokrenemo modul pomoću KVM-a:

```
1 sudo kvm -hda ubuntu-nodejs.img -cdrom ~/zavrzni/livecdtmp/ubuntu-with-nodejs  
-18.04-amd64.iso -boot d -m 2048
```

Rezultat Modul NodeJS



Unutar ove live instalacije možemo upotrijebiti nodeJS biblioteku te kreirati jednostavnu web aplikaciju.

Prateći uputstvo na linku:

<https://docs.microsoft.com/en-us/azure/app-service/app-service-web-get-started-nodejs>

unutar naše live distribucije sa preinstaliranim NodeJS bibliotekama izvršimo sljedeće komande koristeći Terminal:

```
1 git clone https://github.com/Azure-Samples/nodejs-docs-hello-world  
2 cd nodejs-docs-hello-world  
3 npm start
```

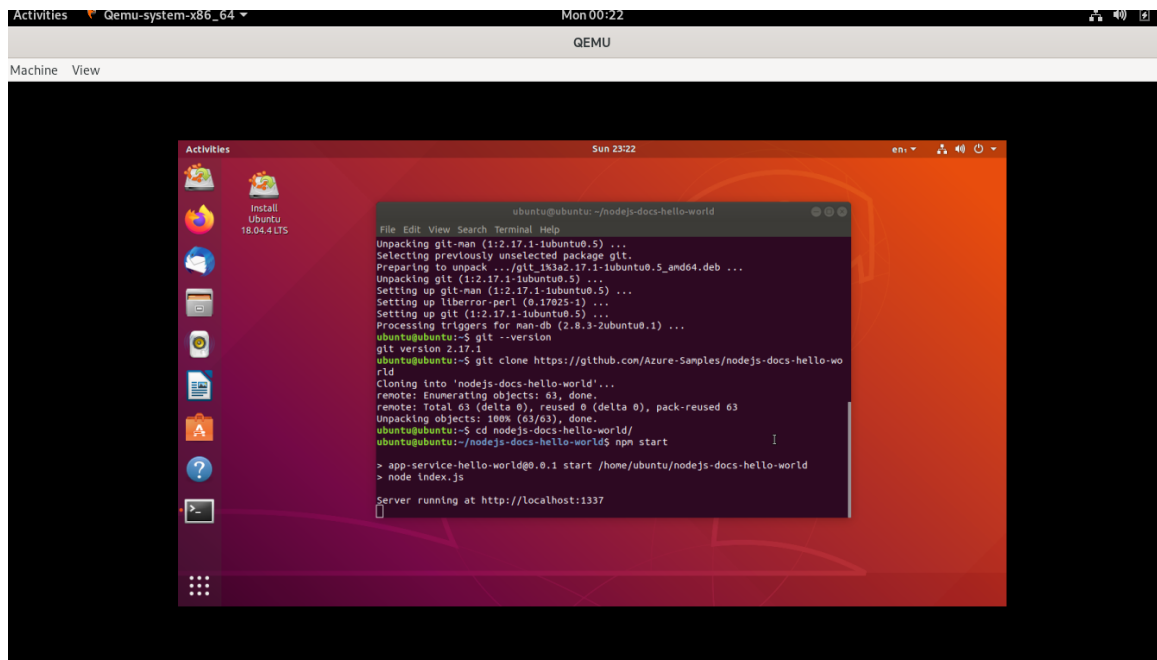
Ukoliko git program nije instaliran potrebno je instalirati git koristeći komandu:

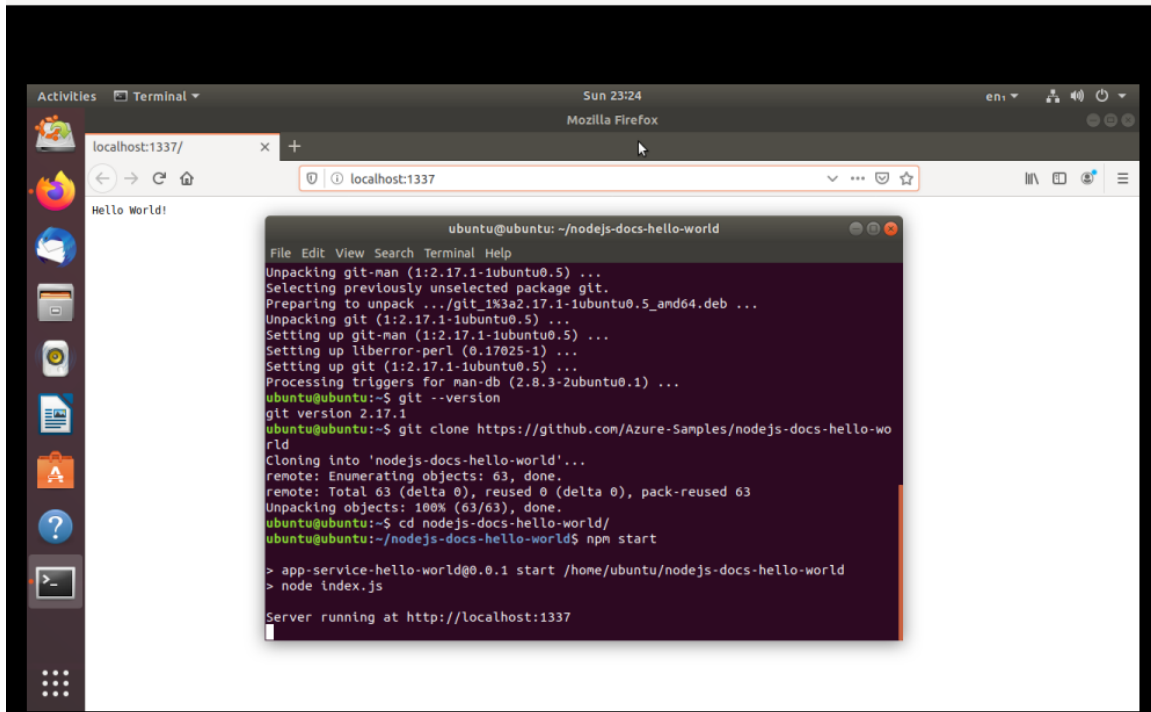
```
1 sudo apt install git
```

Nakon toga NodeJS bi trebao pokrenuti server kojeg mozemo provjeriti web pregledniku na URL-u:

```
1 http://localhost:1337
```

Za potrebe rada nije radjena modifikacija ove web aplikacije, ali moguće je iskoristiti aplikaciju kao bazu za nadogradjivanje po želji. HTTP web server se kreira unutar index.js datoteke te bi početna modifikacija bila svakako nadogradnja ove datoteke za dodatnim funkcionalnostima.





Modul MongoDB

MongoDB je nerelaciona baza podataka napisana u C++ programskom jeziku. Koristi JSON format za spremanje podataka.

To je cini pogodnom za povezivanje sa NodeJS bibliotekama, ciji smo modul vec napravili u prethodnom paragrafu.

Postupak kreiranja ovog modula ce biti gotovo identican postupku kreiranja NodeJS modula, izuzev dijela u kojem se vrši instaliranje novih paketa unutar raspakovanog squashfs datotecnog sistema.

```
1 cd ~/squashfs/livecdtmp
2 sudo mount -o loop ./isoimgs/ubuntu-18.04.4-desktop-amd64.iso mnt
3 mkdir extract-mongodb-cd
4 sudo rsync --exclude=/casper/filesystem.squashfs -a mnt/ extract-mongodb-cd
5 mkdir modul-mongodb
6 sudo rsync -a extract-mongodb-cd/ modul-mongodb
```

Zatim slijedi korak u kojem se opet raspakuje filesystem.squashfs direktorij. Ova operacija moze potrajati par minuta tako da je ne treba prekidat:

```
1 sudo unsquashfs mnt/casper/filesystem.squashfs
```

Te prekopiramo zadrzaj novonastalog squashfs-root direktorija u edit-mongodb direktor:

```
1 sudo mv squashfs-root/ edit-mongodb
```

Da bi imali mrežnu konekciju unutar edit-mongodb direktorija jedno rješenje je kopirati /run direktorij unutar edit-mongodb direktorija. Najbolje manuelno popuniti resolv.conf unutar edit direktorija:

```
1 sudo gedit edit-mongodb/etc/resolv.conf
```

Te unijeti sljedeći sadržaj i spasiti promjene: (*nameserver 1.1.1.1*
nameserver 8.8.8.8).

Isto vazi i za etc/hosts datoteku:

```
1 sudo gedit edit-mongodb/etc/hosts
```

Kopirati sadržaj iz /etc/hosts datoteke na sistemu domaćinu unutar edit-mongodb/etc/hosts datoteke:

```
127.0.0.1          localhost
127.0.1.1          debianjoda.joda.net      debianjoda
```

Namjestiti edit-mongodb/dev direktorij kopirajući /dev/ direktorij sa hosta, zatim chroot u edit-mongodb direktorij. Obaviti mount instrukcije navedene ispod. Ukoliko korisnik odluči da obriše edit-mongodb direktorij iz nekog razloga, bilo bi potrebno uraditi unmount edit-mongodb direktorija da sistem domaćin ne bi postao neupotrebljiv:

```
1 sudo mount --bind /dev/ edit-mongodb/dev
2 sudo chroot edit-mongodb
3 mount -t proc none /proc
4 mount -t sysfs none /sys
5 mount -t devpts none /dev/pts
```

Neophodno je podesiti sistemske varijable pomoću sljedeće komande da bi se izbjegli problemi sa lokalizacijom:

```
1 export HOME=/root
2 export LC_ALL=C
```

Za ispis svih instaliranih paketa:

```
1 dpkg-query -W --showformat='${Installed-Size}\t${Package}\n' | sort -nr | less
```

Naime da bismo mogli pokrenuti mongoDB, neophodno je instalirati neophodne pakete:

```
1 sudo apt-get install libcurl4 openssl
```

Preuzimanje mongoddb paketa sa interneta:

```
1 wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-ubuntu1804-4.2.5.tgz
```

Ekstrakcija paketa:

```
1 tar -zxvf mongodb-linux-x86_64-ubuntu1804-4.2.5.tgz
```

Da bismo izbjegli potrebu da postavimo putanju u PATH sistemsku varijablu, kopiracemo mongoddb bin direktorij u /usr/local/bin/ direktorij:

```
1 sudo cp mongodb-linux-x86_64-ubuntu1804-4.2.5/bin/* /usr/local/bin/
```

Konfiguracija mongoddb paketa:

Prvo napravimo direktorij u koji ce mongoddb spremati podatke:

```
1 sudo mkdir -p /var/lib/mongo
```

Takodjer potrebno je napraviti direktorij u koji ce se spremat logovi:

```
1 sudo mkdir -p /var/log/mongodb
```

Potrebno je azurirati privilegije pristupa na novokreirane direktorije:

```
1 chown 'whoami' /var/lib/mongo
```

```
2 chown 'whoami' /var/log/mongodb
```

Sada mozemo pokrenuti mongod proces:

```
1 mongod --dbpath /var/lib/mongo --logpath /var/log/mongodb/mongod.log --fork
```

Provjera instalacije:

```
1 mongo --version
```

Rezultat komande bi trebao potvrditi uspješno instaliran mongod:

```
1 MongoDB shell version v4.2.5
2 git version: 2261279b51ea13df08ae708ff278f0679c59dc32
3 OpenSSL version: OpenSSL 1.1.1 11 Sep 2018
4 allocator: tcmalloc
5 modules: none
6 build environment:
7   distmod: ubuntu1804
8   distarch: x86_64
9   target_arch: x86_64
```

Bilo bi poželjno mongoDB pokrenuti povezujući je sa drugom IP adresom jer je po defaultu povezana na localhost tj. 127.0.0.1 te može primiti zahtjeve samo od aplikacija koje su na toj masini na kojoj je instaliran mongo.

```
1 Start the server with --bind_ip <address> to specify which IP addresses it should serve
   responses from, or with --bind_ip_all to bind to all interfaces.
```

Nakon završetka instalacije izvršiti unutar chroot "ciscenje":

```
1 apt-get clean
2 rm -rf /tmp/* ~/.bash_history
3 rm -rf /tmp/* ~/.bashrc
4 rm /var/lib/dbus/machine-id
5 rm /sbin/initctl
6 dpkg-divert --rename --remove /sbin/initctl
7 umount /proc || umount -lf /proc
8 umount /sys
9 umount /dev/pts
10 umount /dev
11 exit
```

Ponovno generisati filesystem.manifest:

```
1 sudo chmod +w extract-mongodb-cd/casper/filesystem.manifest
```

```

2 sudo su
3 chroot edit-mongodb dpkg-query -W --showformat='${Package} ${Version}\n' >
    extract-mongodb-cd/casper/filesystem.manifest
4 exit
5 sudo cp extract-mongodb-cd/casper/filesystem.manifest extract-mongodb-cd/casper/
    filesystem.manifest-desktop
6 sudo sed -i '/ubiquity/d' extract-mongodb-cd/casper/filesystem.manifest-desktop
7 sudo sed -i '/casper/d' extract-mongodb-cd/casper/filesystem.manifest-desktop

```

Sada cemo upotrijebiti drugu funkciju iz squashfs-tools, a to je mksquashfs. S tom funkcijom cemo kompresovati edit-mongodb direktorij u novu filesystem.squashfs datoteku. U kodu ispod je potrebno izvršiti komandu iz linije 1 i jednu od preostale 3, pri čemu prva (komanda na liniji 2) daje najslabiju kompresiju, ali je najbrža. Druga komanda se duže izvršava ali je veći procenat kompresije u odnosu na prvu komandu. Dok je kod treće komande procenat kompresije najveći, a vrijeme izvršenja najduže:

```

1 sudo rm extract-mongodb-cd/casper/filesystem.squashfs
2 sudo mksquashfs edit-mongodb extract-mongodb-cd/casper/filesystem.squashfs -no lzma
3 sudo mksquashfs edit-mongodb extract-mongodb-cd/casper/filesystem.squashfs -b
    1048576
4 sudo mksquashfs edit-mongodb extract-mongodb-cd/casper/filesystem.squashfs -comp xz
    -e edit-mongodb/boot

```

Naredni korak je da azuriramo filesystem.size datoteku:

```

1 sudo su
2 printf $(du -sx --block-size=1 edit-mongodb | cut -f1) > extract-mongodb-cd/casper
    /filesystem.size
3 exit

```

Nakon toga upisati naziv image-a unutar README.diskdefines. Upisati 'Ubuntu with MONGODB 18.04.4 LTS "Bionic Beaver" - Release amd64' u polje DISKNAME:

```

1 sudo gedit extract-mongodb-cd/README.diskdefines

```

Azurirati md5sum.txt datoteku:

```
1 cd extract-mongodb-cd
2 sudo rm md5sum.txt
3 find -type f -print0 | sudo xargs -0 md5sum | grep -v isolinux/boot.cat | sudo tee md5sum
   .txt
```

Napokon mozemo napraviti iso image koji ce da sadrzi MONGODB modul. Za ovu operaciju koristimo funkciju genisoimage. Neke linux distribucije nude mkisofs funkciju. Tako da ukoliko ne radi genisoimage trebala bi raditi funkcija mkisofs:

```
1 sudo genisoimage -D -r -V "$IMAGE_NAME" -cache-inodes -J -l -b isolinux/isolinux.
   bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o ../
   ubuntu-with-mongodb-18.04-amd64.iso .
```

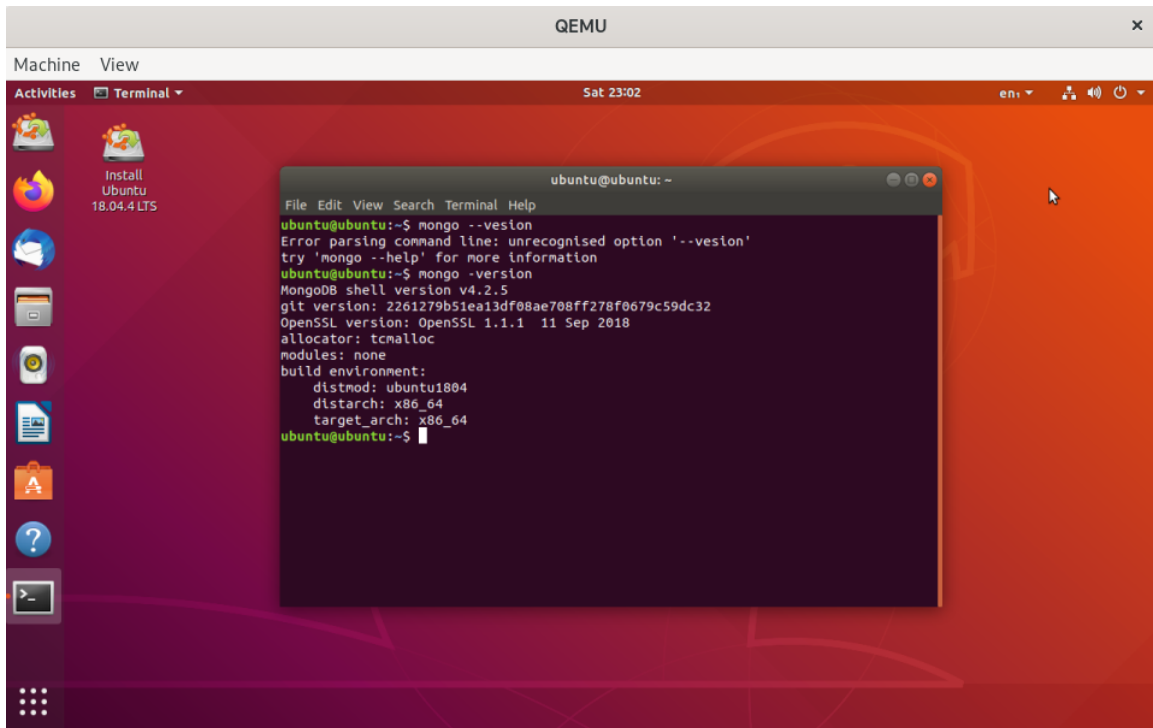
Sada cemo napraviti virtuelni hard disk pomocu qemu-img komande da bismo pokrenuli na njemu nas novi modul MYSQL Ubuntu.

```
1 cd ~
2 qemu-img create ubuntumongodb.img 5G
```

Pokrenucemo modul pomocu KVM-a:

```
1 sudo kvm -hda ubuntumongodb.img -cdrom ~/zavrsni/livecdtmp/ubuntu-with-mongodb
   -18.04-amd64.iso -boot d -m 2048
```

Rezultat Modul MongoDB



```
Machine View
Activities Terminal Sat 23:02 en1
Install
Ubuntu
18.04.4 LTS

ubuntu@ubuntu: ~
File Edit View Search Terminal Help
ubuntu@ubuntu:~$ mongo --vesion
Error parsing command line: unrecognised option '--vesion'
try 'mongo --help' for more information
ubuntu@ubuntu:~$ mongo -version
MongoDB shell version v4.2.5
git version: 2261279b51ea13df08ae708ff278f0679c59dc32
OpenSSL version: OpenSSL 1.1.1 11 Sep 2018
allocator: tcmalloc
modules: none
build environment:
  distmod: ubuntu1804
  distarch: x86_64
  target_arch: x86_64
ubuntu@ubuntu:~$
```

Modul Java

Kao treci primjer modularizacije squashfs datotecnog sistema, kreiran je modul Java. Kao sto mu ime kaze, rijec je o modulu sa instaliranim Java paketima. Vecinom koraci su identicni kao u prethodna 2 slucaja, izuzev koraka instaliranja dodatnih paketa unutar modula.

```
1 cd ~/squashfs/livecdtmp
2 sudo mount -o loop ./isoimgs/ubuntu-18.04.4-desktop-amd64.iso mnt
3 mkdir extract-chrome-cd
4 sudo rsync --exclude=/casper/filesystem.squashfs -a mnt/ extract-chrome-cd
5 mkdir modul-chrome
6 sudo rsync -a extract-chrome-cd/ modul-chrome
```

Zatim slijedi korak u kojem se opet raspakuje filesystem.squashfs direktorij i kopiramo ga u edit-chrome direktorij. Ovaj put cemo edit direktorij imenovati edit-chrome da ne izgubimo prethodni sadrzaj edit direktorija.

```
1 sudo unsquashfs mnt/casper/filesystem.squashfs
2 sudo mv squashfs-root/ edit-chrome
```

Da bi imali mreznu konekciju unutar edit-chrome direktorija jedno rjesenje je kopirati /run direktorij unutar edit-chrome direktorija. Najbolje manuelno popuniti resolv.conf unutar edit-chrome direktorija

nameserver 1.1.1.1

nameserver 8.8.8.8.

Isto vazi i za `etc/hosts` datoteku. Najbolje je provjeriti nakon izvršenih komandi da li je upisan sadržaj u `resolv.conf` i `hosts` datoteke, te ukoliko nije dopuniti nedostatke:

```
1 sudo cp /etc/resolv.conf edit-chrome/etc/
2 sudo mount -o bind /run/ edit-chrome/run
```

Kopirati i `hosts` direktorij/:

```
1 sudo cp /etc/hosts edit-chrome/etc/
```

Namjestiti `edit-chrome/dev` direktorij kopirajući `/dev/` direktorij sa hosta, zatim `chroot` u `edit-chrome` direktorij. Obaviti `mount` instrukcije navedene ispod. Ukoliko korisnik odluči da obrisu `edit-chrome` direktorij iz nekog razloga, bilo bi potrebno uraditi `umount` `edit-chrome` direktorija da sistem domaćin ne bi postao neupotrebljiv:

```
1 sudo mount --bind /dev/ edit-chrome/dev
2 sudo chroot edit-chrome
3 mount -t proc none /proc
4 mount -t sysfs none /sys
5 mount -t devpts none /dev/pts
```

Takodje potrebno je izvršiti sljedeće komande da bi se izbjegli problemi sa lokalizacijom:

```
1 export HOME=/root
2 export LC_ALL=C
```

Za ispis svih instaliranih paketa:

```
1 dpkg-query -W --showformat='${Installed-Size}\t${Package}\n' | sort -nr | less
```

Instalacija `google-chrome` paketa:

```
1 sudo nano /etc/apt/sources.list.d/google-chrome.list
```

Te upisati u ovu datoteku sljedeći sadržaj

```
1 deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main
```

Zatim spasiti datoteku unutar nano editora sa **CTRL+O**, **ENTER** za potvrdu i **CTRL+X** za izlaz iz nano editora.

Sljedeća komanda preuzima Google javni ključ da bismo mogli instalirati google-chrome. Zatim komandom apt-key dodajemo ključ u prsten javnih ključeva da bi apt mogao potvrditi integritet Google Chrome paketa.

```
1 wget https://dl.google.com/linux/linux_signing_key.pub
2 sudo apt-key add linux_signing_key.pub
```

Sada izvršimo azuriranje liste paketa i instaliramo google-chrome-stable paket:

```
1 sudo apt update
2 sudo apt install google-chrome-stable
```

Provjera instalacije:

```
1 google-chrome-stable --version
```

Nakon završetka instalacije izvršiti unutar chroot:

```
1 apt-get clean
2 rm -rf /tmp/* ~/.bash_history
3 rm -rf /tmp/* ~/.bashrc
4 rm /var/lib/dbus/machine-id
5 rm /sbin/initctl
6 dpkg-divert --rename --remove /sbin/initctl
7 umount /proc || umount -lf /proc
8 umount /sys
9 umount /dev/pts
10 exit
11 sudo umount edit-chrome/dev
```

Ponovno generisati filesystem.manifest:

```
1 sudo cp extract-chrome-cd/casper/filesystem.manifest extract-chrome-cd/casper/
   filesystem.manifest-desktop
```

```
2 sudo sed -i '/ubiquity/d' extract-chrome-cd/casper/filesystem.manifest-desktop
3 sudo sed -i '/casper/d' extract-chrome-cd/casper/filesystem.manifest-desktop
4 sudo umount edit/dev
```

Sada cemo upotrijebiti drugu funkciju iz squashfs-tools, a to je mksquashfs. S tom funkcijom cemo kompresovati edit-chrome direktorij u novu filesystem.squashfs datoteku. U kodu ispod je potrebno izvršiti komandu iz linije 1 i jednu od preostale 3, pri čemu prva (komanda na liniji 2) daje najslabiju kompresiju, ali je najbrža. Druga komanda se duže izvršava ali je veći procenat kompresije u odnosu na prvu komandu. Dok je kod treće komande procenat kompresije najveći, a vrijeme izvršenja najduže:

```
1 sudo rm extract-chrome-cd/casper/filesystem.squashfs
2 sudo mksquashfs edit-chrome extract-chrome-cd/casper/filesystem.squashfs -nolzma
3 sudo mksquashfs edit-chrome extract-chrome-cd/casper/filesystem.squashfs -b 1048576
4 sudo mksquashfs edit-chrome extract-chrome-cd/casper/filesystem.squashfs -comp xz -e
  edit/boot
```

Naredni korak je da azuriramo filesystem.size datoteku:

```
1 sudo su
2 printf $(du -sx --block-size=1 edit-mysql | cut -f1) > extract-chrome-cd/casper/
  filesystem.size
3 exit
```

Nakon toga upisati naziv image-a unutar README.diskdefines. Upisati 'Ubuntu with Google Chrome 18.04.4 LTS "Bionic Beaver" - Release amd64' u polje DISKNAME:

```
1 sudo gedit extract-chrome-cd/README.diskdefines
```

Azurirati md5sum.txt datoteku:

```
1 cd extract-chrome-cd
2 sudo rm md5sum.txt
3 find -type f -print0 | sudo xargs -0 md5sum | grep -v isolinux/boot.cat | sudo tee md5sum
  .txt
```

Napokon mozemo napraviti iso image koji ce da sadrzi Google Chrome modul. Za ovu operaciju koristimo funkciju genisoimage. Neke linux distribucije nude mkisofs funkciju. Tako da ukoliko ne radi genisoimage trebala bi raditi funkcija mkisofs:

```
1 sudo genisoimage -D -r -V "$IMAGE_NAME" -cache-inodes -J -l -b isolinux/isolinux.  
   bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o ../  
   ubuntu-with-chrome-18.04-amd64.iso .
```

Sada cemo napraviti virtuelni hard disk pomocu qemu-img komande da bismo pokrenuli na njemu nas novi modul Google Chrome Ubuntu.

```
1 cd ~  
2 qemu-img create ubuntuchrome.img 5G
```

Pokrenucemo modul pomocu KVM-a:

```
1 sudo kvm -hda ubuntuchrome.img -cdrom ~/zavrsni/livecdtmp/ubuntu-with-chrome  
   -18.04-amd64.iso -boot d -m 2048
```
