Antti Käyhkö

FreePBX:n lähdekoodista kääntämisen ja sen asentamisen automatisointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

2.11.2021

Tiivistelmä

Tekijä:	Antti Käyhkö
Otsikko:	FreePBX:n lähdekoodista kääntämisen ja sen asentamisen automatisointi
Sivumäärä:	42 sivua
Aika:	2.11.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Senior System Architect Antti Kerola Lehtori Simo Silander

---

Tämä insinöörityö tehtiin Installe. Installa on tarve luoda uusi FreePBX-asennuspaketti asiakkaalle aina, kun FreePBX:n asennuksesta luodaan uusi versio. Uudet FreePBX-asennuspaketit täytyi aikaisemmin luoda käsin. FreePBX-asennuspakettien luonti käsin on hidasta ja turhaa työtä. Se on mahdollista korvata skriptillä, joka luo asennuspaketit nopeammin ja helpommin.

Tarkoituksena oli kehittää skripti FreePBX-asennuspakettien automaattiseen rakentamiseen. Skriptin täytyisi pakata kaikki FreePBX:n asentamisen vaaditut tiedostot, jotta asennuspaketilla on mahdollista asentaa FreePBX tietokoneelle ilman internetyhteyttä. Tämä tarkoittaa käytännössä sitä, että kaikki FreePBX-asennukseen kuuluva toiminnallisuus ja niiden riippuvuudet täytyy pakata luotavaan asennuspakettiin. Tällaisia riippuvuuksia ovat esim. FreePBX:n ja sen moduulien riippuvuudet, Asteriskin riippuvuudet ja riippuvuuksien riippuvuudet yms.

Rakentamalla FreePBX:n pääriippuvuuden Asteriskin ja muutaman muun FreePBX:n asennukseen tarvittavan riippuvuuspaketin avulla on mahdollista vähentää asennuspakettiin pakattavien tiedostojen määrää. Rakennettaville riippuvuuksille luotiin skriptit siten, että riippuvuuden rakentamisen yhteydessä skripti lataa rakentamisesta puuttuvat lähdetiedostot skriptille määritellyn version mukaan. Eli skriptillä pystyy rakentamaan riippuvuuden, vaikka kaikki lähdetiedostot puuttuisivat, kunhan on skriptille määritelty haluttu versio, jonka mukaan lähdetiedostot ladataan.

Skripti kehitettiin Bash-komentotulkille. Bash-komentotulkki valittiin, koska Bash-komentotulkki sisältyy suureen osaan Linux-jakeluista. Näin saadaan mahdollistettua luodun skriptin käyttö suuressa osassa Linux-jakeluista.

Tässä Insinöörityössä kuvataan FreePBX:n asennuspaketin luonnin eri vaiheita ja sitä, vaiheet on mahdollista rakentaa yhdeksi kokonaisuudeksi, joka mahdollistaa automaattisen FreePBX-asennuspakettien luonnin. Lopuksi kuvataan luodun FreePBX-asennuspakettien rakentajan rakennetta ja käyttöä.

Avainsanat: FreePBX, Asterisk, asennuspaketti, automatisointi, skripti

Abstract

Author:	Antti Käyhkö
Title:	Automating Compilation and Installation of Freepbx From Source Code
Number of Pages:	42 pages
Date:	2 November 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Software Engineering
Supervisors:	Antti Kerola, Senior System Architect Simo Silander, Senior Lecturer

---

This thesis was made for Insta. Insta had a need to create FreePBX installation packages for their customer. New FreePBX installation packages all needed to be created by hand. Creating FreePBX installation packages by hand is slow and unnecessary work that can be replaced with a script that does the same thing only faster and easier.

The goal was to create a script to automate the creation of FreePBX installation packages. The script would need to package all files needed to install FreePBX on an offline computer. This means that all functionality within the FreePBX installation and their dependencies need to be packaged into the created installation package.

By building FreePBXs main dependency Asterisk and some other dependencies it is possible to lower the number of files packaged into the created installation package. Scripts were created for the built dependencies, so that while building a dependency the script will download all missing source files. This means that it is possible to build a dependency even if all its souce files are missing, as long as the script is configured with the needed version, by which the source files are downloaded.

The script was developed for Bash shell. Bash shell was selected because it is included in a large number of Linux distributions. As such it is possible to enable the usage of the created script in a large number of Linux distributions.

In this thesis the different phases of the FreePBX install package creation are described and also how the phases can be combined in one whole that allows the automated creation of FreePBX install packages. Lastly the usage of the created builder is described.

Keywords: FreePBX, Asterisk, package, automation, script

Sisällys

Lyhenteet

RPM: Redhat package manager on pakettienhallintajärjestelmä, jota käytetään useassa Linux-jakelussa.

NPM: Node package manager on Nodejs:n mukana tuleva pakettienhallintajärjestelmä, joka hallitsee noden npm-paketteja.

SSH: SSH on protokolla, joka mahdollistaa turvallisen yhteyden muodostamisen palvelimeen epäturvallisessa verkossa.

1 Johdanto

Tämä insinöörityö tehtiin Installe. Installa on tarve luoda uusi FreePBX-asennuspaketti asiakkaalle aina, kun FreePBX:n asennuksesta luodaan uusi versio. Aikaisemmin FreePBX-asennuspaketit täytyi joka kerta luoda käsin. FreePBX-asennuspakettien luonti käsin on hidasta ja turhaa työtä, joka on mahdollista korvata skriptillä, joka luo asennuspaketit nopeammin ja helpommin.

Tarkoituksena oli kehittää skripti FreePBX-asennuspakettien automaattiseen rakentamiseen. Skriptin täytyisi pakata kaikki FreePBX:n asentamisen vaaditut tiedostot, jotta on asennuspaketilla mahdollista asentaa FreePBX-tietokoneelle ilman internetyhteyttä. Tämä tarkoittaa käytännössä sitä, että kaikki FreePBX-asennukseen kuuluva toiminnallisuus ja niiden riippuvuudet täytyy pakata luotavaan asennuspakettiin. Tällaisia riippuvuuksia ovat esim. FreePBX:n ja sen moduulien riippuvuudet, Asteriskin riippuvuudet ja riippuvuuksien riippuvuudet yms.

Rakentamalla FreePBX:n pääriippuvuuden Asteriskin ja muutaman muun FreePBX:n asennukseen tarvittavan riippuvuuspaketin avulla on mahdollista vähentää asennuspakettiin pakattavien tiedostojen määrää. Rakennettaville riippuvuuksille luotiin skriptit siten, että riippuvuuden rakentamisen yhteydessä skripti lataa rakentamisesta puuttuvat lähdetiedostot skriptille määritellyn version mukaan. Eli skriptillä pystyy rakentamaan riippuvuuden, vaikka kaikki lähdetiedostot puuttuisivat, kunhan on skriptille määritelty haluttu versio, jonka mukaan lähdetiedostot ladataan.

Luotavaa skriptiä tullaan ajamaan CentOS 7 Linux -jakelulla, joten täytyy luotavan skriptin olla ajettavissa Linux-ympäristössä. Tämä tarkoittaa, että skripti täytyy luoda jollekin CentOS 7 Linux -jakelulta löytyvälle komentotulkille. Lähdettiin skriptiä kehittämään Bash-komentotulkille. Bash-komentotulkki valittiin, koska Bash-komentotulkki sisältyy suureen osaan Linux-jakeluista. Näin saadaan mahdollistettua luodun skriptin käyttö suuressa osassa Linux-jakeluista.

Tässä Insinöörityössä kuvataan FreePBX:n asennuspaketin luonnin eri vaiheita ja sitä, miten vaiheet on mahdollista rakentaa yhdeksi kokonaisuudeksi, joka mahdollistaa automaattisen FreePBX-asennuspakettien luonnin.

2 Tausta

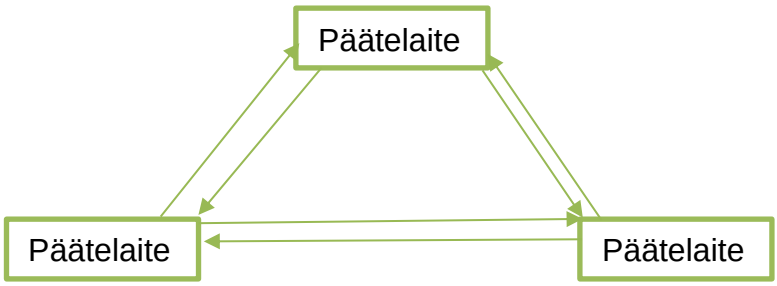
Installa oli tarve saada FreePBX-asennuspakettien luonti automatisoitua. Joka kerta, kun FreePBX-asennusta muutettiin, täytyi manuaalisesti luoda uusi asennuspaketti. Uuden asennuspaketin luonti käsin on turhaa työtä, jonka voi helposti korvata skripteillä, jotka tekevät saman asian helpommin ja nopeammin.

Installa oli jo tapa manuaalisesti tehdä asennuspaketteja FreePBX:ää varten, mutta tätä tapaa ei enää ollut mahdollista käyttää, koska asennuspakettien manuaalinen luonti oli huonosti dokumentoitu, koska se jätti muutaman valmiin asennuspaketin ja jotakin dokumentaatiota. Aluksi täytyi lähteä selvittämään, mikä on tilanne ja paljonko jo tehtyä materiaalia voi käyttää.

2.1 Kommunikaatiojärjestelmä

Esimerkiksi yrityksellä voi olla monta konttoria ja jokaisessa konttorissa on useita päätelaitteita. Näillä päätelaitteilla pitäisi pystyä soittamaan muihin päätelaitteisiin, vaikka päätelaite sijaitsee toisella konttorilla. Kommunikaatiojärjestelmä mahdollistaa useiden päätelaitteiden, kuten puhelimien yhdistämisen useaan muuhun päätelaitteeseen.

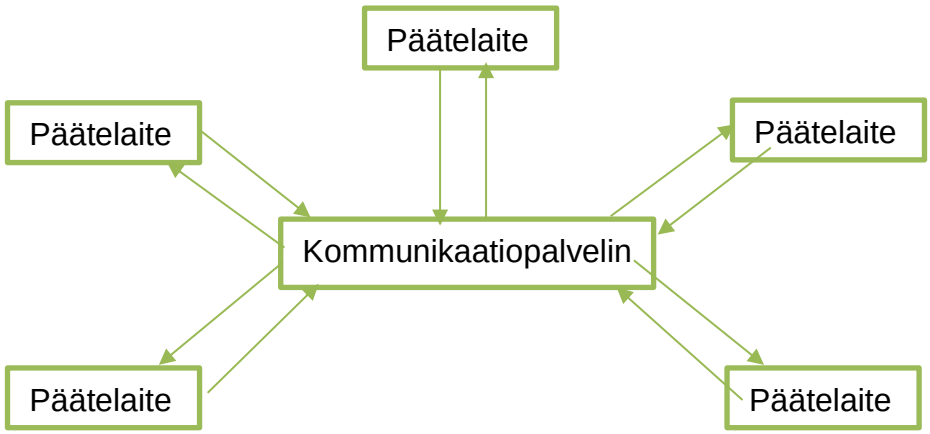
On mahdollista yhdistää päätelaitteet suoraan toisiin päätelaitteisiin, mutta luodun verkon monimutkaisuus kasvaa verkkoon lisättyjen päätelaitteiden lisääntyessä. Kuvassa 1 on kaavio kolmen päätelaitteen verkosta.



Kuva 1: Kaavio kolmen päätelaitteen verkosta

Jokaisen päätelaitteen täytyy tuntea kaikki muut päätelaitteet, eli kolmen päätelaitteen verkossa jokaisella päätelaitteella on 2 yhteyttä. Jos verkko kasvaa kymmeneen päätelaitteeseen, niin jokainen päätelaite tarvitsee yhteyden yhdeksään päätelaitteeseen.

Verkon monimutkaisuuden kasvu päätelaitteita lisättäessä on mahdollista korjata lisäämällä päätelaitteiden väliin kommunikaatiopalvelin, joka keskustelee päätelaitteiden kanssa. Näin päätelaitteiden ei tarvitse tuntea kaikkia muita päätelaitteita, ja verkon monimutkaisuus pysyy pienempänä. Jokaisen päätelaitteen täytyy vain tuntea kommunikaatiopalvelin ja kommunikaatiopalvelin yhdistää päätelaitteet toisiinsa tarpeen mukaan. Kuvassa 2 on kaavio viiden päätelaitteen verkosta, jossa kommunikaatiopalvelin yhdistää päätelaitteet toisiinsa.



Kuva 2: Kaavio viiden päätelaitteen verkosta kommunikaatioserverillä

Kommunikaatiopalvelin lisääminen päätelaiteverkkoon helpottaa työtä myös verkkoon päätelaitteita lisättäessä ja poistettaessa. Ilman kommunikaatiopalvelinta päätelaitetta lisättäessä täytyy lisättävälle päätelaitteelle listata kaikki verkossa olevat päätelaitteet. Kommunikaatioserveriä käytettäessä on mahdollista lisätä uusia päätelaitteita helpommin. Yhdistettäessä uusi päätelaite kommunikaatiopalvelimeen päätelaite lisätään kommunikaatiopalvelimen päätelaitelistaan ja nyt on mahdollista yhdistää uusi päätelaite muihin verkossa oleviin päätelaitteisiin.

2.2 Private branch exchange (PBX)

Private branch exchange (PBX) -termi tulee ajalta, kun puhelimella soitettaessa ihminen manuaalisesti yhdistivät kaksi puhelinta käyttäen puhelinvaihdetta. PBX on järjestelmä, joka yhdistää päätelaitteet toisiinsa käyttäen jotain sille määriteltyä protokollaa. PBX voi olla fyysinen laitteisto tai ohjelmisto, joka automaattisesti hallitsee yhteyksiä sille määriteltyjen päätelaitteiden välillä. (1.)

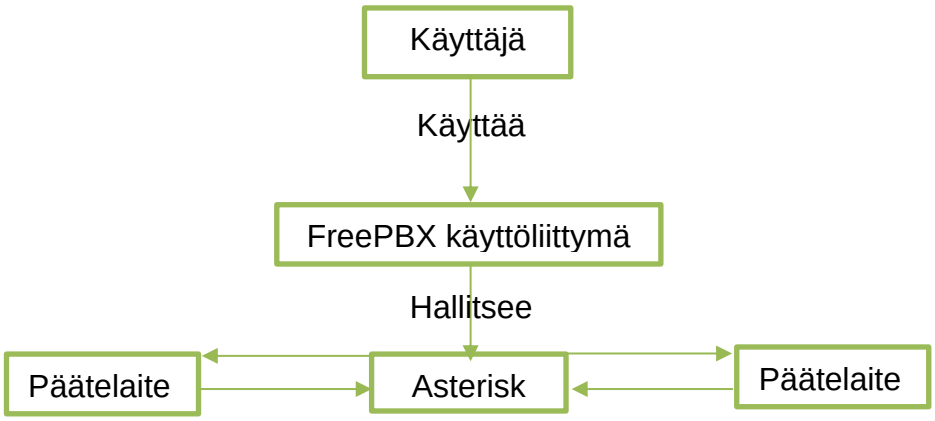
2.3 Asterisk

Asterisk on private branch exchange (PBX) -ohjelmistototeutus, joka mahdollistaa yhteyksien luomisen ja hallinnan eri päätelaiteiden välillä käyttäen useita eri protokollia, kuten VoIP:ia. VoIP:n avulla on Asteriskin mahdollista yhdistää kaksi päätelaitetta internetin välityksellä. Asterisk on myös mahdollista konfiguroida käyttämään ulkoista laitteistoa tarvitsevia protokollia, kuten ISDN:ää ja SS7:ää. (2.)

Asteriskilla on mahdollista luoda kommunikaatiojärjestelmä, joka käyttää samanaikaisesti eri protokollia ja koodekkeja eri päätelaitteiden välillä. Asterisk on mahdollista konfiguroida haluamakseen lisäämällä tai poistamalla Asteriskin käyttämiä koodekkeja ja protokollia. Asteriskin asennus asentaa tietokoneelle Asterisk-kommunikaatiopalvelimen, joka toimii PBX:nä päätelaitteiden välillä.

2.4 FreePBX

FreePBX on avoimen lähdekoodin IP PBX, joka mahdollistaa erilaisten puhelinverkkojen rakentamisen eri vaatimuksien mukaan. Tämän muokattavuuden mahdollistaa FreePBX:n modulaarisuus. Eri moduuleita lisäämällä tai poistamalla on mahdollista muokata puhelinverkko haluamakseen. (3;4.)



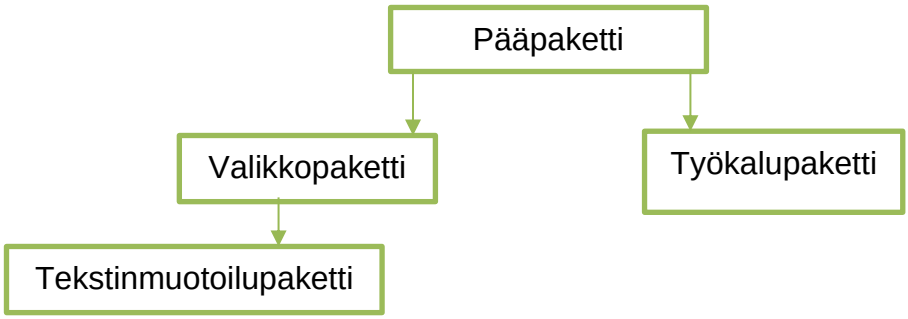
Kuva 3: Käyttäjän, FreePBX:n ja Asteriskin välinen yhteys

FreePBX:n avoin lähdekoodi mahdollistaa uusien moduulien kehityksen helposti. FreePBX-yhteisö on kehittänyt moduuleja eri koodekeille ja protokollille. Kun tietää kehitettävän puhelinverkon vaatimukset, voi valita tarvittavat moduulit jokaista tuettavaa koodekkia ja protokollaa varten. On myös mahdollista itse lähteä kehittämään jotakin moduulia FreePBX:lle, jos valmista moduulia ei löydy. (5;6.)

2.5 Rpm-paketti

Rpm-paketit ovat asennuspaketteja, jotka asentavat jonkin sovelluksen tai jotain toiminnallisuutta tietokoneelle. Paketilla voi olla useita riippuvuuspaketteja, jotka vaaditaan pääpaketin toimimiseen.

Paketti käyttää riippuvuuksia toiminnallisuuden lisäämiseen. Yksi riippuvuuspaketti voi tuoda esimerkiksi käyttöliittymän valikoita pääpakettiin. Pääpaketti voi rakentaa muiden pakettien toiminnallisuuden päälle uutta toiminnallisuutta. Tämä nopeuttaa pakettien kehitystä, kun ei tarvitse itse koodata kaikkea alusta lähtien, kunhan ei tuo kaikkea toiminnallisuutta luotavaan pakettiin riippuvuuspaketteina. (7.)



Kuva 4: Paketin riippuvuuspuu

Yhdellä paketilla voi olla riippuvuuspaketteja ja riippuvuuspaketilla useita riippuvuuspaketteja. Lopuksi paketin riippuvuudet luovat puumaisen rakenteen, jota havainnollistetaan kuvassa 4.

3 Työssä tarvittavat työkalut

Tässä luvussa kuvataan työhön tarvitut työkalut.

3.1 Rpm

Rpm on pakettienhallintajärjestelmä. Pakettienhallintajärjestelmä automatisoi pakettien version, tiedostojen hallinnan ja pakettien validoinnin. Jos ei paketinhallintajärjestelmää olisi, täytyisi itse pitää kirjaa kunkin paketin tiedostoista ja itse selvittää konfliktit pakettien tiedostojen välillä. Yhdellä paketilla voi olla useita satoja tiedostoja ja yhdellä tietokoneella yli sata pakettia. Paketin päivittäminen ilman pakettienhallintajärjestelmää on haasteellista: täytyy selvittää paketin riippuvuuspaketit ja niiden riippuvuuspaketit, selvittää, mitä paketteja täytyy päivittää, selvittää, päivitettävien pakettien tiedostot ovat ja korvata ne uusilla tiedostoilla. On mahdollista, että paketin manuaaliseen päivitykseen menee useita tunteja. Pakettienhallintajärjestelmää käyttäen tämän usean tunnin prosessin saa tehtyä yhdellä komennolla alle minuutissa. (8.)

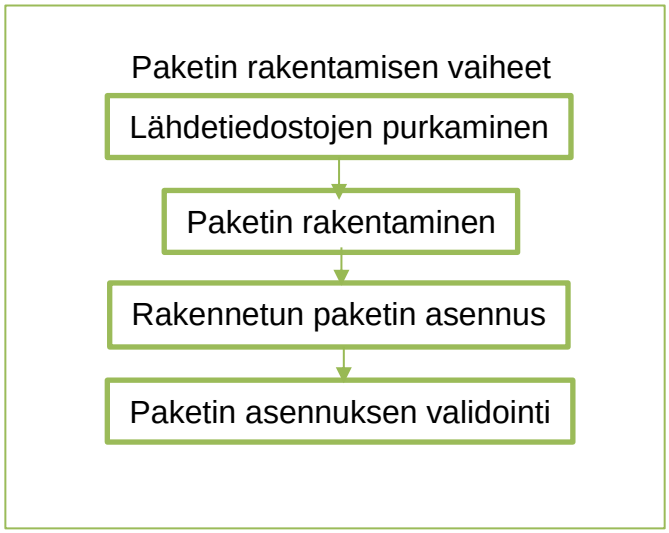
Rpm tulee sanoista Redhat package manager. Se on pakettienhallintajärjestelmä, joka pitää kirjaa pakettien versioista ja siitä, mitä riippuvuuksia kullakin paketilla on. Rpm pitää kirjaa pakettien asentamista tiedostoista ja ratkaisee ristiriitoja eri pakettien välillä. (9.)

3.2 Rpmbuild

Rpmbuild on osa Redhat package manageria. Rpmbuild mahdollistaa pakettien luomisen käyttäen .spec-tiedostoa paketin rakentamisohjeena.

3.2.1 Paketin rakentamisen vaiheet

Paketin rakentamisessa rpmbuild-komennolla on neljä askelta, joita nähdään kuvassa 5.



Kuva 5: Paketin rakentamisen vaiheet

Paketin rakentaminen alkaa lähdetiedostojen purkamisella. Tämän jälkeen tehdään tarvittavat muutokset purettuihin tiedostoihin käyttäen PATCH-tiedostoja. Alkuperäisiin tiedostoihin ei tehdä muutoksia, koska näin on mahdollista tehdä paketin rakentaminen toistettavaksi ja voidaan tarvittavat muutokset tehdä tiedostoihin rakentamisen yhteydessä käyttäen PATCH-tiedostoja muutosten tekemiseksi. (10.)

Patch-tiedostoja luodaan käyttämällä diff-komentoa. Komennolla pystyy vertaamaan kahta tiedostoa ja näkemään erot tiedostojen välillä. PATCH-tiedostoja saadaan luotua seuraavasti: Tehdään jotakin muutoksia johonkin tiedostoon ja verrataan sitä diff-komennolla alkuperäiseen tiedostoon, tallennetaan muutokset näiden kahden tiedostojen välillä PATCH-tiedostona. (11.)

Lähdetiedostojen purkamisen jälkeen rakennetaan paketti yleensä käyttäen make-komentoa. Make-komento ajaa makefile-tiedostossa olevat komennot paketin rakentamiseksi. (10.)

Paketin rakentamisen jälkeen rakennettu paketti asennetaan tietokoneelle käyttäen make install -komentoa. Tämä mahdollistaa paketin asennuksen varmistamisen. (10.) Paketin asennuksen jälkeen katsotaan, löytyvätkö kaikki tiedostot oikeilta paikoiltaan. Jos jotain tiedostoa ei löydy, epäonnistuu paketin rakentaminen. (10.)

Rmpbuild luo rpm-tiedostot, jos kaikki rakentamisen vaiheet ovat onnistuneet. Luoduilla rpm-tiedostoilla on mahdollista asentaa paketti halutulle tietokoneelle. (10.)

3.2.2 Spec-tiedosto

Spec-tiedosto on rpmbuild:n käyttämä määrittelytiedosto, jossa määritellään kaikki paketin rakentamisvaiheet ja miten ja minne rakennetun paketin tiedostot asennetaan. Spec-tiedosto sisältää myös kaikki paketin asennettujen tiedostojen paikat. Tätä listaa vasten verrataan, onnistuiko paketin asennus. (12.)

Spec-tiedosto koostuu neljästä osasta, jotka ovat: %prep, %build, %install, %files (12). Prep-osassa valmistellaan rakentamisen tiedostot, puretaan lähdetiedostot ja tehdään tarvittavat muutokset käyttäen PATCH-tiedostoja.

Build-osassa spec-tiedostoa rakennetaan paketti, joko kirjoittamalla build-osaan skripti paketin rakentamiseksi tai käyttämällä make-komentoa ja ajamalla makefile-tiedostosta löytyvät skriptit. (12.)

Install-osassa sijaitsee skripti paketin asentamiseksi tietokoneelle. On mahdollista käyttää make-komentoa tai kirjoittaa skripti tähän osaan paketin asentamiseksi. (12.)

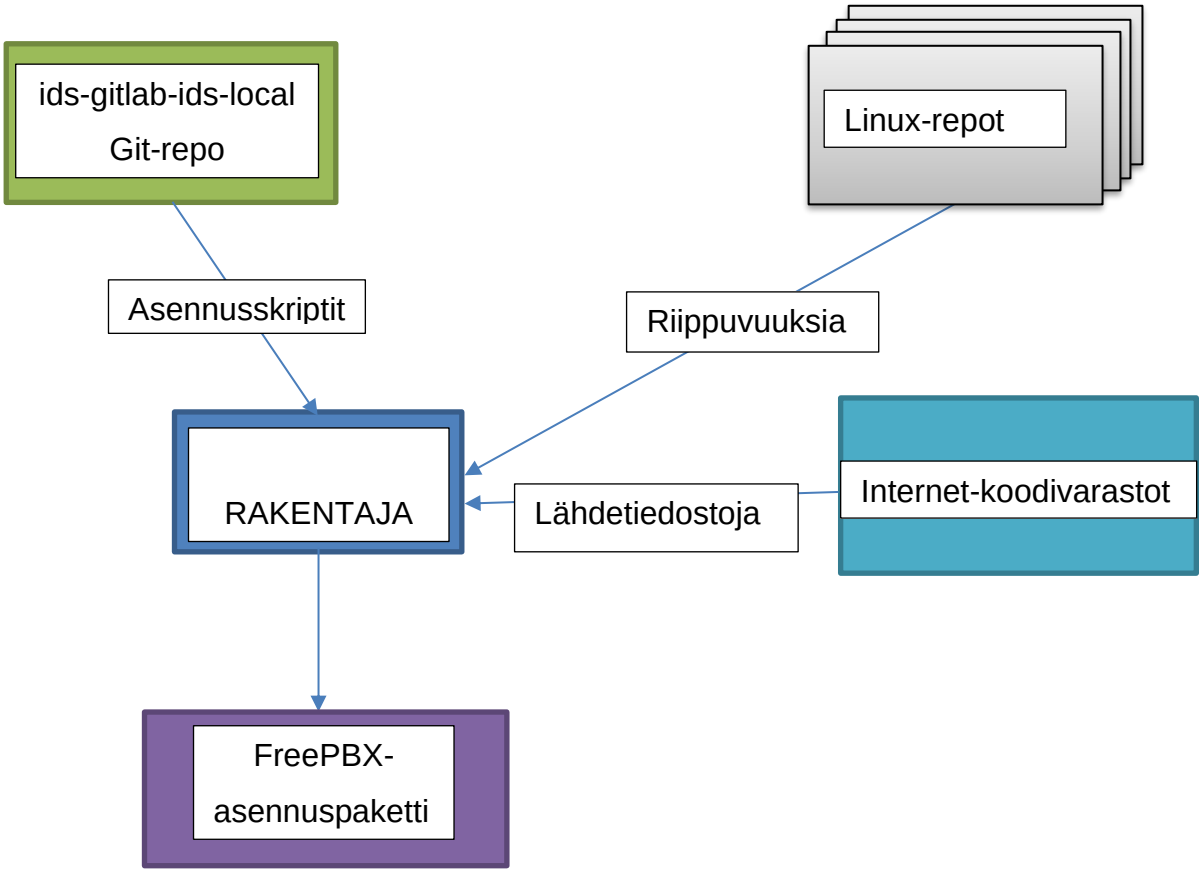
Files-kohdassa on listaus kaikista paketin tiedostoista ja niiden sijainnista tietokoneella, kun se on asennettuna tietokoneelle (12). Kuvassa 6 näytetään esimerkki spec-tiedostoa.

```
Name: eject
Version: 2.1.5
Release: 1%{?dist}
Summary: A program that ejects removable media using software control

License: GPLv2+
URL: http://www.pobox.com/~tranter
Source0: http://www.ibiblio.org/pub/Linux/utils/disk-management/%{name}-%{version}.tar.gz

BuildRequires: gettext
BuildRequires: libtool

%description
The eject program allows the user to eject removable media (typically
CD-ROMs, floppy disks or Iomega Jaz or Zip disks) using software
control. Eject can also control some multi-disk CD changers and even
some devices' auto-eject features.

Install eject if you'd like to eject removable media using software
control.

%prep
%setup -q -n

%build
%configure
make %{?_smp_mflags}

%check
make check

%install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT

install -m 755 -d $RPM_BUILD_ROOT/%{_sbindir}
ln -s ../bin/eject $RPM_BUILD_ROOT/%{_sbindir}

%find_lang %{name}

%files -f %{name}.lang
%doc README TODO COPYING ChangeLog
%{_bindir}/*
%{_sbindir}/*
%{_mandir}/man1/*

%changelog
* Wed Oct 20 2011 John Doe <jdoe@example.com> 0.8.18.1-0.1
- Initial RPM release
```

Kuva 6: Spec-tiedostoesimerkki (13)

4 Toteutettava työ

Tässä luvussa kuvataan projektiin toteutettava työ. Kuvataan projektin eri työvaiheet projektin alusta loppuun. Kuvassa 7 kuvataan luodun rakentajan toimintaa.



Kuva 7: Kuvaus luodun rakentajan toiminnasta

- Rakentaja rakentaa FreePBX-asennuspaketin käyttämällä erilaisia resursseja internetistä ja rpm-koodivarastoista.

- Lokaalista Git-koodivarastosta rakentaja lataa asennusskriptit, joita käytetään FreePBX:n asennuksessa.

- Rakentaja käyttää useaa Linux-koodivarastoa kaikkien tarvittavien FreePBX-riippuvuuksien lataamiseen.

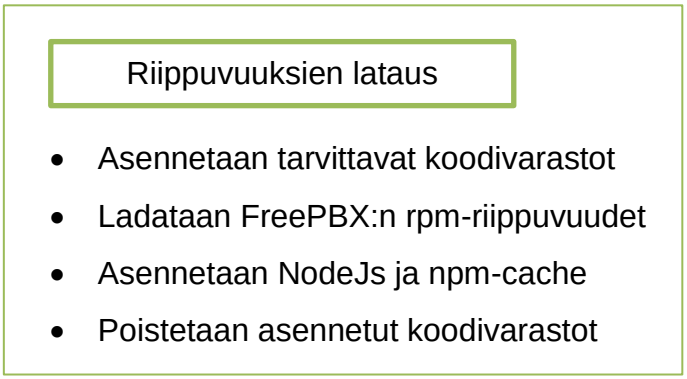
- Rakentaja lataa lähdetiedostoja internetistä löytyvistä koodivarastoista, esim. Asterisk-lähdetiedostoja ladataan Git-koodivarastoista.

- Rakentaja rakentaa Asteriskin ja MariaDB-connectorin käyttäen internetistä ladattavia lähdetiedostoja.

- Rakentaja rakentaa mvgw-apin käyttäen lokaalia Git-koodivarastoa.

4.1 Riippuvuuksien lataaminen

FreePBX:n riippuvuuksien lataamisessa on viisi pääkohtaa, jotka kuvataan kuvassa 8. Käydään seuraavaksi läpi riippuvuuksien latauksen pääkohdat tarkemmin.



Kuva 8: Riippuvuuksien latauksen pääkohdat

Riippuvuudet ladataan luotavaan asennuspakettiin, koska luodun asennuspaketin täytyy pystyä asentamaan FreePBX offline-ympäristössä ilman yhteyttä internetiin. Eli täytyy ladata FreePBX:n kaikki riippuvuuspaketit, joita kuvattiin luvussa 2.5.

Koska kaikkia FreePBX:n riippuvuuksia ei ole mahdollista ladata samasta koodivarastosta, tarvitaan useita rpm-koodivarastoja. Tarvittavat koodivarastot ovat MariaDB, epel, remi, nyx dextop ja NodeJs.

- MariaDB foundation on tunnettu avoimen lähdekoodin MySQL-relaatio-tietokannasta. Ladataan MariaDB-koodivarastosta MySQL-relaatiotieto-kanta ja client tietokantaan. (14)

- Remi tarjoaa viimeisimmän version PHP-pinosta, joka on saatavilla Re-min ylläpitämästä koodivarastossa. Ladataan Remi koodivarastosta PHP7. (15)

- Nux dextop on kolmannen osapuolen rpm-koodivarasto, joka sisältää multimedia- ja komentorivipaketteja. (16)

- NodeJs-koodivarastosta tarvitaan npm eli noden pakettienhallintajärjes-telmä ja NodeJs.

- Epel tulee sanoista Extra Packages for Enterprise Linux. Tästä koodiva-rastosta löytyy useita FreePBX riippuvuuksia CentOs 7 Linux-jakelulle. (17.)

Koodivarastojen asentamisen jälkeen ladataan kaikki FreePBX-tarvittavat riip-puvuudet. Latauksen jälkeen poistetaan asennetut koodivarastot koneelta.Siir-retään vielä lopuksi ladatut rpm-tiedostot rakentajan rpms-kansioon.

Skripti käyttää hyväkseen Yum-komentoa. Yum-komennolla asentaessa paket-teja Yum lataa paketin rpm-tiedoston /var/cache/yum-kansioon. Skripti tyhjentää aluksi kansion mahdollisista rpm-paketeista, koska ei tarvita olemassa olevia paketteja FreePBX:n asennukseen.

Koodivarastojen asennus tapahtuu siirtämällä .repo-tiedosto /etc/yum.repos.d/-kansioon. Tämä kansio sisältää kaikki yum-komennon käyttämät koodivarastot. Jokaiselta yum-komennon käyttämältä koodivarastolta löytyy .repo-tiedosto tästä kansiosta. On mahdollista manuaalisesti siirtää tarvittavien koodivarastojen .repo-tiedostot /etc/yum.repos.d/-kansioon tai käyttää rpm-pakettia, joka asentaa koodivaraston samalla tavalla ja asentaa koodivaraston validointiavaimen, joka mahdollistaa koodivarastosta ladattujen pakettien validoinnin. Validoinnissa katsotaan, onko paketti muuttunut sen allekirjoituksen jälkeen. Jos paketti on muuttunut, ladataan se uudestaan, koska latauksen aikana muuttunut paketti voi sisältää haittaohjelmia, tai osa paketista on voinut korruptoitua latauksen aikana. Skripti asentaa tarvittavat koodivarastot:

- MariaDB-koodivarasto asennetaan siirtämällä .repo-tiedosto /etc/yum.repos.d/-kansioon. Tämä tiedosto luotiin hakemalla .repo-tiedosto MariaDB:n sivuilta CentOS 7 Linux-jakelulle.

- Epel-koodivarasto asennetaan käyttämällä yum-komentoa. Yum-komento lataa rpm-tiedoston, joka asentaa Epel-koodivaraston.

- Asennetaan Nux dextop -koodivarasto lataamalla sen validointiavain ja asennetaan koodivarasto itsessään rpm-komennolla käyttäen rpm-pakettia.

- Asennetaan Remi-koodivarasto yum-komennolla käyttäen rpm-pakettia.

- Asennetaan lopuksi nodejs-koodivarasto, josta saadaan ladattua nodejs ja npm.

Yum install -komento toimii vain, jos pakettia ei ole jo asennettu tietokoneelle. Jos paketti löytyy jo tietokoneelta, ei yum lataa pakettia uudelleen. Tämän takia on tärkeää, että skripti ajetaan CentOS 7 -koneella, joka sisältää vain peruspaketit. Muuten luotu asennuspaketti hajoaa puuttuvien riippuvuuksien vuoksi.

```
sudo yum -y install --downloadonly httpd httpd-tools mod_ssl \
php73 php73-php-cli php73-php-gd php73-php-imap php73-php-intl php73-php-json php73-php-odbc php73-php-pecl-zip php73-php-pear php73-php-ldap php73-php-mbstring
php73-php-pecl-imagick php73-php-pecl-mysql php73-php-bcmath php73-php-xml php73-php-process php73-php-snmp php73-php-common php73-php php73-php-pdo \
MariaDB-server MariaDB-client \
gcc-c++ make nodejs \
mpg123 sox lame wavpack ffmpeg \
gmime codec2 cpptest libevent libmodman libproxy libsrtp neon pakchois unbound-libs uriparser \
NetworkManager-config-routing-rules

sudo find /var/cache/yum/ -name *.rpm -exec mv {} $HOME/ \;
```

Kuva 9: FreePBX:n rpm-riippuvuudet

Riippuvuuksien lataamisen jälkeen siirretään ladatut rpm-tiedostot rpms-kansioon. Tähän kansioon kerätään kaikki FreePBX:n asennukseen vaaditut rpm-tiedostot. Kuvassa 9 kuvataan koodia riippuvuuksien latauksesta ja siirrosta rpms-kansioon.

Seuraavaksi asennetaan nodejs ja npm-cache. Nodejs-pakettia asennettaessa asentuu myös npm, joka on noden paketinhallintajärjestelmä. Täytyy sekä npm että npm-cache olla asennettuna globaalisti, jotta saadaan FreePBX moduulin pm2 npm riippuvuudet ladattua. Globaalissa asennuksessa npm-paketti asennetaan kaikille käyttäjille. Lokaalissa asennuksessa npm-paketti asennetaan vain asennuskomennon ajavalle käyttäjälle.

Npm-cache mahdollistaa npm-paketin ja sen riippuvuuksien cachettamisen, jotta se voidaan asentaa offline-ympäristössä ilman internetyhteyttä (18). Kuvassa 10 kuvataan luotua skriptiä, jossa asennetaan npm, nodejs ja npm-cache.

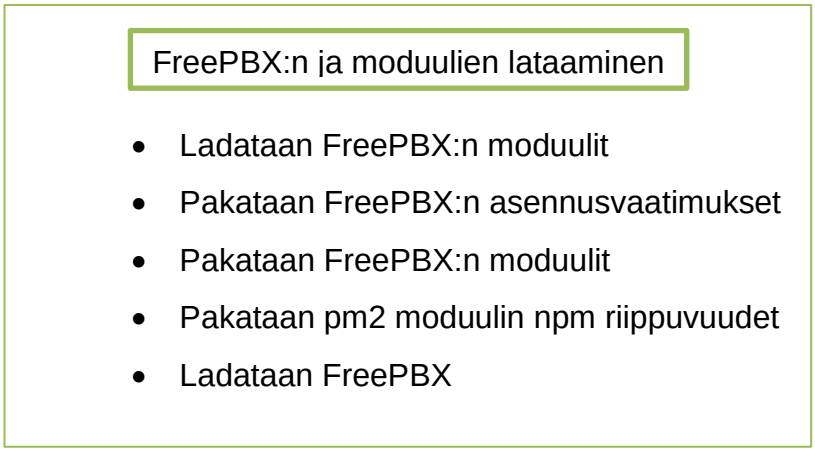
Lopuksi skripti poistaa asennetut koodivarastot, koska usea koodivarasto voi sisältää samalla nimellä olevia paketteja. Näin tämän jälkeiset skriptit toimivat ilman konflikteja pakettien nimien kanssa. Kuvassa 10 poistetaan asennetut koodivarastot.

```
# Installing npm and npm-cache for downloading pm2 npm dependencies
sudo yum -y install nodejs
sudo npm install -g npm-cache

# Removing installed repositories and modules so they don't mess with other scripts.

sudo yum -y remove epel-release MariaDB-common

# .repo files are located /etc/yum.repos.d/. Deleting these files will remove given repo.
if [[ -e "/etc/yum.repos.d/MariaDB.repo" ]]; then
    sudo rm /etc/yum.repos.d/MariaDB.repo
fi

if [[ -e "/etc/yum.repos.d/nodesource-el7.repo" ]]; then
    sudo rm /etc/yum.repos.d/nodesource-el7.repo
fi

sudo yum clean packages

date "+%Y.%m.%d %H.%M:%S" > $HOME/PACKAGES_VER


echo "Packages downloaded, see $HOME"
```

Kuva 10 FreePBX npm- ja nodejs-asennus

4.2 FreePBX:n ja sen moduulien lataaminen.

FreePBX:n ja sen moduulien lataamisessa on viisi pääkohtaa. Pääkohdat on kuvattu kuvassa 11. Käydään seuraavaksi läpi pääkohdat yksi kerrallaan.



Kuva 11: FreePBX:n ja moduulien lataaminen pääkohdat

FreePBX:n moduuleja ladattaessa moduulit jaetaan kahteen kansioon.

Ensimmäiseen kansioon asetetaan moduulit, jotka vaaditaan FreePBX:n asennukseen ja täten täytyy asentaa ennen FreePBX:n asennusta. Kansio pakataan ja käytetään pakattua kansiota yhtenä asentajan lähdetiedostona.

Toiseen kansioon menevät kaikki moduulit, joilla lisätään toiminnallisuutta jo asennettuun FreePBX-järjestelmään. Toinen kansio pakataan ja käytetään pakattua kansiota yhtenä asentajan lähdetiedostona.

Moduulit ladataan allekirjoittamattomina, koska näin ne ovat saatavilla kauemmin, eikä tarvitse pelätä latauslähteiden muuttumista. Moduuleille on kaksi lähdettä: FreePBX:n GitHub-sivut ja FreePBX:n oma verkkopalvelin, johon on useasti tehty muutoksia.

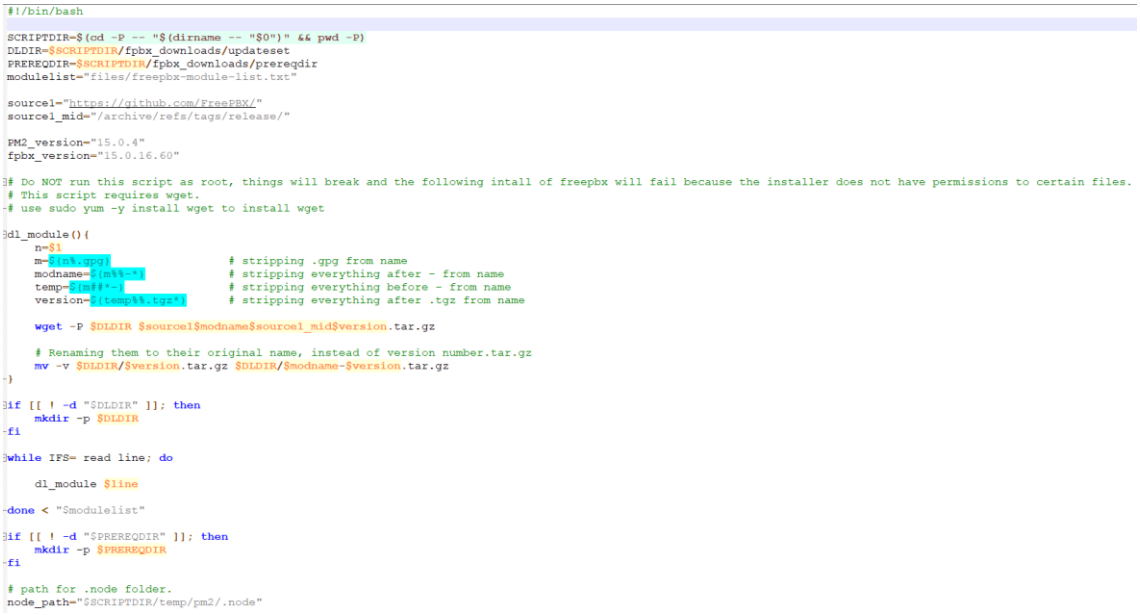
Moduulit ladataan käyttämällä wget-pakettia. Moduulien latausta varten luodaan funktio, jossa sille annettava parametri on ladattavan moduulin nimi. Moduulin nimi on rakenteeltaan:

```
moduulinNimi-moduulinVersio.tar.gz.
```

Parametrina annetusta moduulin nimestä pilkotaan kaksi muuttujaa: moduulin nimi ja sen versio. Seuraavaksi funktiossa rakennetaan moduulin latausosoite, joka on rakenteeltaan:

```
github.com/FreePBX/moduulinNimi/archive/refs/tags/release/moduulinVersio.tar.gz
```

Moduulien nimet on kirjoitettu erilliseen tekstitiedostoon, jossa jokaisella rivillä on moduulin nimi. Moduulien nimet syötetään yksi kerrallaan edellämainittuun funktioon, jossa luodaan moduulin latausosoite ja ladataan moduuli käyttäen luotua latausosoitetta. Kuvassa 12 näytetään kooditoteutus moduulien lataamisesta.



```
#!/bin/bash

SCRIPTDIR=$(cd -P -- "$(dirname -- "$0")" && pwd -P)
DLDIR=$SCRIPTDIR/fpbx_downloads/updateset
PREREQDIR=$SCRIPTDIR/fpbx_downloads/prereqdir
modulelist="files/freepbx-module-list.txt"

source1="https://github.com/FreePBX/"
source1_mid="/archive/refs/tags/release/"

PM2_version="15.0.4"
fpbx_version="15.0.16.60"

# Do NOT run this script as root, things will break and the following intall of freepbx will fail because the installer does not have permissions to certain files.
# This script requires wget.
# use sudo yum -y install wget to install wget

dl_module(){
    n=$1
    n=${n%.gpg}                 # stripping .gpg from name
    modname=${n%%-*}            # stripping everything after - from name
    temp=${n##*-}               # stripping everything before - from name
    version=${temp%%.tgz*}      # stripping everything after .tgz from name

    wget -P $DLDIR $source1$modname$source1_mid$version.tar.gz

    # Renaming them to their original name, instead of version number.tar.gz
    mv -v $DLDIR/$version.tar.gz $DLDIR/$modname-$version.tar.gz
}

if [[ ! -d "$DLDIR" ]]; then
    mkdir -p $DLDIR
fi

while IFS= read line; do

    dl_module $line

done < "$modulelist"

if [[ ! -d "$PREREQDIR" ]]; then
    mkdir -p $PREREQDIR
fi

# path for .node folder.
node_path="$SCRIPTDIR/temp/pm2/.node"
```

Kuva 12 FreePBX-moduulien lataus

Moduulien lataamisen jälkeen pakataan pm2 ja soundlang erilliseen kansioon, koska nämä moduulit täytyy asentaa ennen FreePBX:n asennusta. Pm2 on kriittinen FreePBX-moduuli, jota tarvitaan FreePBX:n asennuksessa. Muut ladatut moduulit pakataan keskenään samaan kansioon.

Pm2-moduuli tarvitsee npm-riippuvuuksia. Nämä riippuvuudet ladataan käyttämällä npm-cache-pakettia. Npm-cache mahdollistaa npm-pakettien ja pakettien riippuvuuksien pakkaamisen.

Npm-cache pakkaa jonkin paketin käyttäen sen package.json-tiedostoa. Package.json-tiedosto sisältää jonkin paketin tarvitsemat pakettiriippuvuudet. Kun pakkaus on suoritettu, npm-cache luo package_cache-kansion, joka sisältää pakatun paketin ja sen riippuvuuspaketit. Käyttäen pakattua pakettia on mahdollista asentaa se offline-tietokoneelle ilman yhteyttä internetiin.

On mahdollista löytää npm-cachen tarvitsema package.json-tiedosto pm2 FreePBX-moduulista. Puretaan moduulista package.json-tiedosto ja käytetään purettua tiedostoa pm2-moduulin riippuvuuksien pakkaamiseen. Npm-cache laskee package.json-tiedostosta hash-arvon, jota se käyttää pakatun paketin tiedostonimenä.

```
# Creating .node folder with bin, lib and share subfolders
sudo -u $(logname) mkdir -p $node_path/{bin,lib,share}
pm2_temp="$SCRIPTDIR/temp/pm2/"

# Moving FreePBX install prerequisites.
#pm2-15.0.4.tar.gz
#soundlang-15.0.5.10.tar.gz
mv -v $DLDIR/pm2-* $PREREQDIR
mv -v $DLDIR/soundlang-* $PREREQDIR

# Packaging pm2 installation and npm dependencies for offline install.
if [[ -e "$PREREQDIR/pm2-$PM2_version.tar.gz" ]]; then
    # Searching the tar for package.json file
    pm2_package=$(tar -tzf $PREREQDIR/pm2-$PM2_version.tar.gz | grep package.json)

    # Untarring package.json file
    sudo -u $(logname) tar -xzf $PREREQDIR/pm2-$PM2_version.tar.gz -C temp $pm2_package

    # Using npm-cache to cache pm2 installation dependencies using untarred package.json file.
    cd temp/$(dirname $pm2_package);sudo -u $(logname) npm-cache install pm2; cd -

    # copy created package cache to pm2 temp folder.
    cp -ar /home/$(logname)/.package_cache $pm2_temp
    cp -ar /home/$(logname)/.npm $pm2_temp
fi

# Copying all globally installed npm packages. In this case npm and npm-cache. Install npm packages globally with: npm install package -g
cp -ar /usr/lib/node_modules $node_path/lib

# Copying npm man files under share
cp -ar $node_path/lib/node_modules/npm/man $node_path/share

# Creating soft symbolic links in .node/bin
ln -s ../lib/node_modules/npm-cache/index.js $node_path/bin/npm-cache
ln -s ../lib/node_modules/npm/bin/npm-cli.js $node_path/bin/npm
ln -s ../lib/node_modules/npm/bin/npx-cli.js $node_path/bin/npx

find $pm2_temp -printf "%P\n" -type f -o -type l -o -type d | tar -czf $SCRIPTDIR/fpbx_downloads/freepbx-15.0-pm2-15.0.4.tar.gz --no-recursion -C $pm2_temp -T -

if [[ -d "temp" ]]; then
    rm -r temp
fi

tar -czvf $SCRIPTDIR/fpbx_downloads/freepbx-15.0-installreq-2.tar.gz -C $PREREQDIR .
```

Kuva 13 FreePBX-asennuksen moduuliriippuvuuksien ja pm2-npm-riippuvuuksien lataus

Nyt on pm2-moduulin npm-riippuvuudet pakattu. Seuraavaksi luodaan .node-kansio, koska asennuksen aikana asentaja etsii npm-cache-toiminnallisuutta paikallisesta .node-kansiosta. Kuvassa 13 kuvataan koodia pm2-npm-riippuvuuksien pakkauksesta.

Tämän toiminnallisuuden tuottamiseksi oli tarpeellista aikaisemmin asentaa npm ja npm-cache globaalisti. Kopioimme npm- ja npm-cache-globaalin asennuksen /usr/lib/node_modules-kansiosta .node/lib-kansioon. Seuraavaksi täytyy linkittää kopioidun kansion toiminnallisuus .node/bin-kansioon, josta asentaja hakee toiminnallisuutta.

Linkitetään toiminnallisuus luomalla pehmeitä symbolisia linkkejä. Pehmeät symboliset linkit osoittavat tiedostoihin, joista toiminnallisuus löytyy. Luodut linkit ovat Windows-maailmassa tunnettu pikakuvakkeina, jotka osoittavat samalla tavalla johonkin muuhun tiedostoon. Luodaan linkit npm-, npx- ja npm-cache-tiedostoille. Lopuksi vielä kopioidaan npm:n man-sivut .node/share-kansioon. Man-sivut ovat ohjesivuja komentojen käyttöön.

Seuraavaksi pakataan luotu .node-kansio ja npm-cachen käytön yhteydessä luodut .npm ja .package_cache yhteen tiedostoon. Tällä tiedostolla on FreePBX:n asennuksen aikana mahdollista asentaa pm2-moduulin npm-riippuvuudet offline-tietokoneelle ilman internetyhteyttä.

Asennuksen aikana asentajalla ei ollut tarvittavia käyttöoikeuksia .node-kansioon pakatun npm-cachen käyttöön. Tämä vika korjattiin tekemällä konfiguraatiotiedostoa, jonka mukaan FreePBX:n asennuksen yhteydessä asetetaan tiedostoille konfiguraatiossa määritetyt käyttöoikeudet. Kuvassa 14 esitetään FreePBX:n chown-komennon konfiguraatiotiedostoa. (19.)

```
1  [custom]
2  file = /var/lib/asterisk/.node/lib/node_modules/npm/bin/npm-cli.js,777,asterisk,asterisk
3  file = /var/lib/asterisk/.node/lib/node_modules/npm/bin/npx-cli.js,777,asterisk,asterisk
4  file = /var/lib/asterisk/.node/lib/node_modules/npm-cache/index.js,777,asterisk,asterisk
```

Kuva 14: FreePBX chown -konfiguraatiotiedosto

4.3 FreePBX-riippuvuuksien rakentaminen lähdekoodista

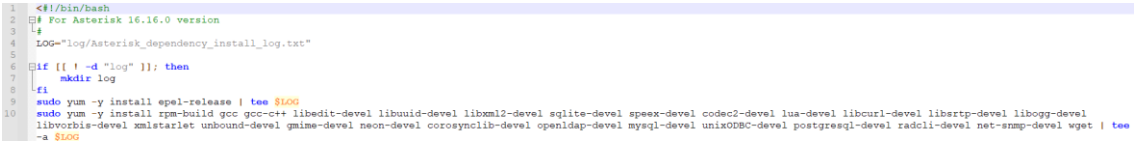
FreePBX:llä on kolme riippuvuutta, joita ei ole mahdollista ladata internetistä. Tässä luvussa kuvataan Asteriskin, MariaDB-connectorin ja mvgw-apin rakentaminen. Rakentamisessa käytetään rpmbuild-pakettia, joka mahdollistaa muiden pakettien rakentamisen. Rpmbuild käyttää rpm-pakettimanageria eri pakettien rakentamisessa. Rpm pitää kirjaa, mitä riippuvuuksia mikäkin paketti tarvitsee ja seuraa pakettien tiedostoja estääkseen konflikteja eri pakettien välillä helpottaakseen pakettien rakentamista.

4.3.1 Asteriskin rakentaminen lähdekoodista

Kuvassa 15 kuvataan Asteriskin lähdekoodista rakentamisen pääkohdat. Käydään seuraavaksi läpi pääkohdat yksi kerrallaan.

Kuva 15: Asteriskin rakentamisen pääkohdat

Ennen Asteriskin rakentamista lähdekoodista täytyy asentaa rakentamiseen vaaditut paketit. Kaikkia vaadittuja paketteja ei ole mahdollista asentaa CentOS 7:n Linux-jakelun mukana tulleista koodivarastoista. Asennetaan epel-koodivarasto, josta saadaan osa tarvittavista paketeista. Kuvassa 16 kuvataan koodia Asteriksin rakentamiseen vaadittujen riippuvuuksien asentamisesta.



```
#!/bin/bash
# For Asterisk 16.16.0 version
#
LOG="log/Asterisk_dependency_install_log.txt"

if [[ ! -d "log" ]]; then
    mkdir log
fi
sudo yum -y install epel-release | tee $LOG
sudo yum -y install rpm-build gcc gcc-c++ libedit-devel libuuid-devel libxml2-devel sqlite-devel speex-devel codec2-devel lua-devel libcurl-devel libsrtp-devel libogg-devel
libvorbis-devel xmlstarlet unbound-devel gmime-devel neon-devel corosynclib-devel openldap-devel mysql-devel unixODBC-devel postgresql-devel radcli-devel net-snmp-devel wget | tee
-a $LOG
```

Kuva 16: Asteriskin rakentamisen riippuvuudet

Asteriskin rakentamisen riippuvuudet on nyt asennettu. Tämän jälkeen ladataan Asteriskin .spec-tiedostossa määritelty versio Asteriskista.

Ladatusta Asterisk-tiedostosta on mahdollista löytää sen vaatimat kolmannen osapuolen riippuvuudet. Tiedot vaadituista kolmannen osapuolen riippuvuuksien versioista on mahdollista löytää Asteriskin kansiosta third-party. Kansiossa olevassa versions.mak-tiedostosta löytyy Asteriskin tarvitsemat versiot pjprojectista ja janssonista.

Muokataan tiedostoa, jotta sitä on mahdollista käyttää skriptissä. Poistetaan tiedostosta välilyönnit ja lisätään jokaisen rivin alkuun export-sana. Export tuo janssonin ja pjprojectin versiot käytettäväksi skriptille. Kuvassa 17 kuvataan versions.mak-tiedostoa.

```
1  JANSSON_VERSION = 2.12
2  PJPROJECT_VERSION = 2.10
3
```

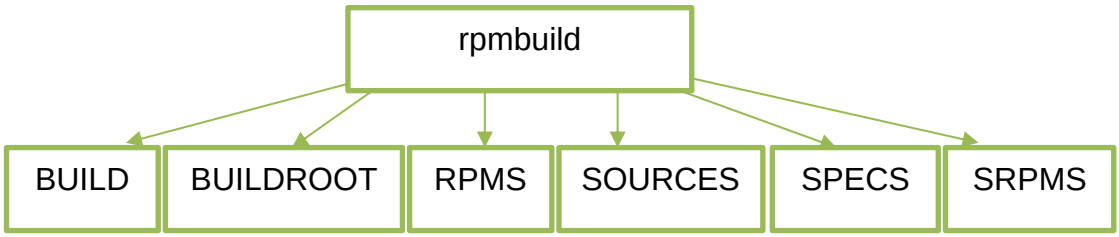
Kuva 17: Versions.mak-tiedosto

Ladataan versions.mak-tiedostossa kuvatut versiot janssonista ja pjprojektista. Seuraavaksi ladataan Codec opus Asteriskin version mukaan. Jokaiselle Asteriskin pääversiolla on saatavilla Codec opus -versio.

Ei Asteriskille ole vielä ladattu ääniä, joten ladataan Asteriskin äänet viidellä eri koodekilla, mikä mahdollistaa äänien toiston käyttäen kyseisiä koodekkeja.

Siirretään kaikki ladatut tiedostot dlcache-kansioon Asteriskia lukuun ottamatta. Pakataan kansio. Nyt on luotu yksi Asteriskin .spec-tiedostoissa määritellyistä lähdetiedostoista.

Mpg123 on MPEG-tiedostojen dekooderi, joka mahdollistaa mp3-tiedostojen soittamisen ja dekoodauksen (18). Ladataan Mpg123:n tiedostot kansioon addons/mp3 ja pakataan kansio. Käytetään pakattua kansiota yhtenä Asteriskin .spec-tiedostosta löytyvänä lähteenä.



Kuva 18: Rpmbuild-puu

Nyt on ladattu kaikki tarvittavat lähdetiedostot Asteriskin rakentamista varten. Seuraavaksi luodaan rpmbuild-puu, johon asetetaan Asteriskin lähdetiedostot. Siirretään ladatut lähdetiedostot rpmbuild-puun SOURCES-kansioon. Tästä kansiosta rpmbuild etsii Asteriskin lähdetiedostoja sen rakentamisen yhteydessä. Siirretään Asteriskin .spec-tiedosto SPECS-kansioon. Tässä kansiossa säilytetään kaikki rakentamiseen liittyvät .spec-tiedostot. Kuvassa 18 havainnollistetaan rpmbuild-puun rakennetta.

Varmistetaan ennen Asteriskin rakentamista, että on ladattu kaikki Asteriskin .spec-tiedostossa määritellyt lähdetiedostot.

```
Name:           asterisk
Version:        16.16.0
Release:        1%{?dist}
Summary:        Asterisk, The Open Source PBX
License:        GPLv2
URL:            https://www.asterisk.org/
#Source:        https://downloads.asterisk.org/pub/telephony/asterisk/releases/asterisk-%{version}.tar.gz
Source0:        asterisk-16.16.0.tar.gz
Source1:        asterisk-16-dlcache.tar.gz
Source2:        asterisk-16-mpg123_rev202.tar.gz
Source3:        asterisk.logrotated
Source4:        asterisk.systemd
Source5:        asterisk.sysconfig
Patch0:         asterisk16160_00-remove_default_voicemails.patch
BuildArch:      x86_64
Vendor:         Insta DefSec Oy
Packager:       Markus Hakkinen <markus.hakkinen@insta.fi>
Group:          MVGW/Backend

%description
Asterisk is an open source PBX and telephony development platform.
Asterisk can both replace a conventional PBX and act as a
platform for the development of custom telephony applications for
the delivery of dynamic content over a telephone; similar to how
one can deliver dynamic content through a web browser using CGI
and a web server.

Asterisk supports a variety of telephony hardware including BRI,
PRI, POTS, and IP telephony clients using the Inter-Asterisk
eXchange (IAX) protocol (e.g. gnophone or miniphone).  For more
information and a current list of supported hardware, see
http://www.asterisk.org
```

Kuva 19: Asteriskin .spec-tiedosto

Kuvassa 19 kuvataan Asteriskin .spec-tiedostoa. Näemme .spec-tiedostosta katsomalla, että kaikki tarvittavat lähdetiedostot on ladattu. Kuvassa 19 olevassa .spec-tiedostossa logrotated-, systemd- ja sysconfig-tiedostot ovat rakennuksessa käytettäviä konfiguraatiotiedostoja. PATCH-tiedostot muokkaavat Asteriskin lähdetiedostoja paketin rakentamisen prep-osassa, katso luku 3.2.2.

Asteriskin rakentamisessa käytetään rpmbuild-pakettia. Tällä paketilla on mahdollista rakentaa muita paketteja. Rpmbuild käyttää .spec-tiedostoja paketin rakentamisohjeena. Rpmbuild seuraa .spec-tiedostoon määriteltyjä vaiheita, jotka on kuvattu luvussa 3.2.2.

Aloitetaan Asteriskin rakentaminen seuraavalla komennolla:

```
rpmbuild –define ”_topdir $(pwd -P)” -ba /SPECS/asterisk.spec
```

Rakentamisen päätyttyä löytyy rakennuspuun RPMS-kansiosta Asteriskin rakennuksen aikana luodut rpm-tiedostot.

Luodaan skriptit aikaisemmin käsiteltyjen askeleiden automatisoimiseksi. Luodaan kaksi skriptiä. Ensimmäinen skripti asentaa Asteriskin rakentamiseen vaaditut riippuvuudet. Toinen skripti lataa määritellyn Asterisk-version mukaan Asteriskin rakentamiseen vaaditut lähdetiedostot.

Viimeinen skripti tulee aikaisemmin toteutetusta työtilasta. Viimeinen skripti rakentaa Asteriskin käyttäen rpmbuild-komentoa ja logittaa koko asteriskin rakennusprosessin. Viimeinen skripti myös pilkkoo tuotettua rpm-tiedostoa pienemmiksi rpm-tiedostoiksi eri Asteriskin ominaisuuksien mukaan. Muokkasin tätä skriptiä, jotta se toimii paremmin muiden skriptien kanssa. Nyt skripti hakee kansion, jossa se sijaitsee ja toimii sen mukaan, missä skripti sijaitsee, eikä mistä sitä kutsutaan.

Nyt on Asteriskin rakentamisen automatisointiin kolme skriptiä, jotka asentavat tarvittavat riippuvuudet Asteriskin rakentamista varten ja lataavat rakentamiseen vaaditut lähdetiedostot ja rakentavat Asteriskin sen .spec-tiedoston mukaan.

Luodaan skripti, joka ajaa nämä skriptit oikeassa järjestyksessä. Tämän skriptin luonnin jälkeen on mahdollista rakentaa Asterisk ajamalla vain yksi skripti. Kuvassa 20 kuvataan toteutettua skriptiä, joka yhdistää Asteriskin rakentamisen yhden skriptin alle.

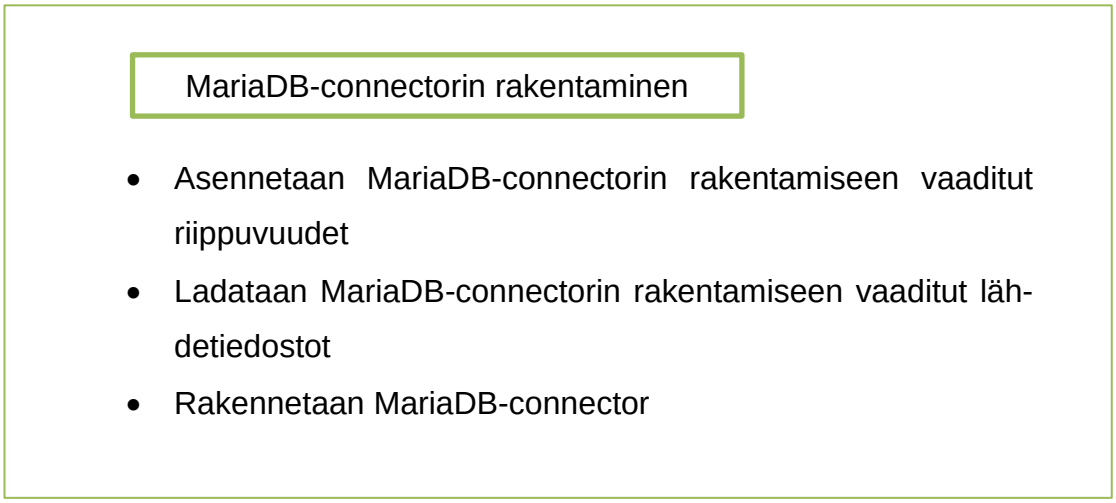
```
#!/bin/bash
# For Asterisk 16.16.0 version
# Set the working folder to the same folder the script is in.
BASEPATH=$(cd -P -- "$(dirname -- "$0")" && pwd -P)

echo "##>> Beginning Asterisk dependency install..."
echo "##>> Please insert your password to continue installing dependencies...(yum install)"
bash $BASEPATH/rpmbuild/install_asterisk_dependencies.sh
echo "##>> Installation of Asterisk dependencies complete."
echo "##>> Beginning download of needed files for Asterisk 16.16.0 build."
bash $BASEPATH/rpmbuild/download_needed_asterisk_files.sh
echo "##>> Beginning Asterisk 16.16.0 build."
bash $BASEPATH/rpmbuild/build.sh SPECS/asterisk-16.16.0.spec
```

Kuva 20 Build Asterisk -skripti

Kuvassa 20 kuvattu skripti hakee kansion, jossa se sijaitsee käyttämällä BASE-PATH-muuttujaan sijoitettua komentoa. Näin skripti toimii sen mukaan, missä kansiossa se on, eikä mistä kansiosta sitä ajetaan.

4.3.2 MariaDB-connectorin rakentaminen lähdekoodista.

Seuraavaksi rakennetaan toinen FreePBX:n riippuvuuksista. Kuvassa 21 kuvataan MariaDB-connectorin rakentamisen pääkohdat. Käydään seuraavaksi pääkohdat läpi yksi kerrallaan.



Kuva 21: MariaDB-connectorin rakentamisen pääkohdat

Ennen MariaDB-connectorin rakentamista täytyy asentaa rakentamiseen vaaditut riippuvuuspaketit. Kuvassa 22 kuvataan koodia MariaDB-connectorin rakentamiseen vaadituista riippuvuuksista.

```
11  echo "##>> Installing dependencies for mariadb connector build."
12  sudo yum -y install cmake make gcc openssl-devel unixODBC rpm-build wget
13  echo "##>> Installation of dependencies complete."
14  # Create the rmpbuild tree.
15  echo "##>> Creating rpmbuild tree.."
16  mkdir -p $SCRIPTDIR/rpmbuild/{BUILD,BUILDROOT,RPMS,SOURCES,SPECS,SRPMS}
17  echo "##>> Rpmbuild tree created"
18
```

Kuva 22: MariaDB-connectorin rakentamisen riippuvuudet

Luodaan MariaDB-connectorin rakentamista varten rpmbuild-puu. Kuvassa 22 kuvataan rpmbuild-puun luontikoodia.

Seuraavaksi ladataan MariaDB-connectorin rakentamiseen vaaditut lähdetiedostot. Lähdetiedostoja on kaksi mariadb-c ja mariadb-odbc. Ladataan sama versio molemmista lähdetiedostoista. Siirretään ladatut lähdetiedostot SOURCES-kansioon, josta rpmbuild hakee MariaDB-connectorin lähdetiedostoja.

Ennen MariaDB-connectorin rakentamista täytyy vielä siirtää MariaDB-connectorin .spec-tiedosto rakennuspuun SPECS-kansioon.

Aloitetaan MariaDB-connectorin rakentaminen ajamalla rpmbuild-komento:

```
rpmbuild –define ”_topdir $(pwd -P)” -ba /SPECS/mariadb-connector.spec
```

Rakentamisen päätyttyä löytyy rakennuspuun RPMS-kansiosta luotu MariaDB-connectorin rpm-tiedosto.

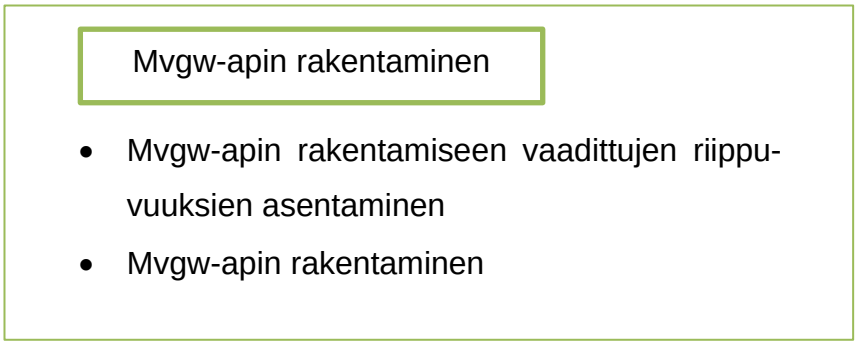
Luodaan skripti aikaisemmin käsiteltyjen askeleiden automatisoimiseksi. Skripti asentaa MariaDB-connectorin rakentamiseen vaaditut riippuvuudet. Seuraavaksi skripti lataa MariaDB-connectorin lähdetiedostot. Lopuksi skripti rakentaa MariaDB-connectorin. Kuvassa 23 kuvataan luodun skriptin koodia.

```
#!/bin/bash
# Build script for building mariadb connector.

SCRIPTDIR=$(cd -P -- "$(dirname -- "$0")" && pwd -P)
BASEPATH=$SCRIPTDIR/rpmbuild
SOURCES="${BASEPATH}/SOURCES"

mariadb_c_version="3.1.11"
mariadb_odbc_version="3.1.11"

echo "##>> Installing dependencies for mariadb connector build."
sudo yum -y install cmake make gcc openssl-devel unixODBC rpm-build wget
echo "##>> Installation of dependencies complete."
# Create the rmpbuild tree.
echo "##>> Creating rpmbuild tree.."
mkdir -p $SCRIPTDIR/rpmbuild/{BUILD,BUILDROOT,RPMS,SOURCES,SPECS,SRPMS}
echo "##>> Rpmbuild tree created"



# Moving possible spec file to rpmbuild specfile folder.
if [[ -e "$SCRIPTDIR/mariadb-connector-odbc.spec" ]]; then
    cp $SCRIPTDIR/mariadb-connector-odbc.spec ${BASEPATH}/SPECS
elif [[ -e "$BASEPATH/mariadb-connector-odbc.spec" ]]; then
    cp $BASEPATH/mariadb-connector-odbc.spec ${BASEPATH}/SPECS
else
    echo "No spec file found for build exiting..."
    exit 1
fi

echo "##>> Downloading required files for mariadb connector build. Here we rename the files to their original names."
if [[ ! -e "$SOURCES/mariadb-connector-c-3.1.11.tar.gz" ]]; then
    wget -P $SOURCES https://github.com/mariadb-corporation/mariadb-connector-c/archive/refs/tags/v$mariadb_c_version.tar.gz
    mv $SOURCES/v$mariadb_c_version.tar.gz $SOURCES/mariadb-connector-c-$mariadb_c_version.tar.gz
fi
if [[ ! -e "$SOURCES/mariadb-connector-odbc-3.1.11.tar.gz" ]]; then
    wget -P $SOURCES https://github.com/mariadb-corporation/mariadb-connector-odbc/archive/refs/tags/$mariadb_odbc_version.tar.gz
    mv $SOURCES/$mariadb_odbc_version.tar.gz $SOURCES/mariadb-connector-odbc-$mariadb_odbc_version.tar.gz
fi
echo "##>> Download complete"

echo "##>> Starting maridb connector build."
rpmbuild --define "_topdir $BASEPATH" -ba ${BASEPATH}/SPECS/mariadb-connector-odbc.spec
echo "##>> Build finished."

```

Kuva 23: MariaDB-connectorin rakentamisskripti

4.3.3 Mvgw-apin rakentaminen lähdekoodista

Kuvassa 24 kuvataan mvgw-apin rakentamisen pääkohdat. Käydään seuraavaksi läpi pääkohdat yksi kerrallaan.



Kuva 24: Mvgw-apin rakentamisen pääkohdat

Ennen mvgw-apin rakentamista täytyy asentaa rakentamiseen vaaditut riippuvuuspaketit. Kuvassa 25 kuvataan koodia mvgw-apin riippuvuuksien rakentamisesta.

```
11  echo "##>> Installing mvgw-api dependencies..." | tee -a $LOG
12  sudo yum -y install rpm-build |  tee -a $LOG
13  echo "##>> Installation of dependencies complete." |  tee -a $LOG
```

Kuva 25: Mvgw-apin rakentamisen riippuvuudet

Mvgw-apin lähdetiedostot ovat vain ladattavissa Instan sisäiseltä Git-serveriltä. Ladataan mvgw-apin lähdetiedostot.

Luodaan rpmbuild-puu, asetetaan lähdetiedostot SOURCES-kansioon ja .spec-tiedosto SPECS-kansioon.

Tämän jälkeen on kaikki tarvittavat tiedostot mvgw-apin rakentamiseen. Aloitetaan rakentaminen rpmbuild-komennolla:

```
rpmbuild –define ”_topdir $(pwd -P)” -ba /SPECS/mvgw-api.spec
```

Rakentamisen jälkeen luodut rpm-tiedostot löytyvät rakennuspuun RPMS-kansiosta.

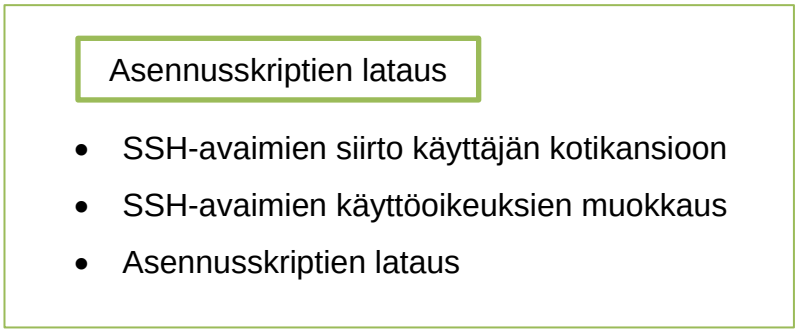
Luodaan skripti edellä mainittujen askeleiden automatisoimiseksi. Skripti asentaa mvgw-apin rakentamiseen vaaditut riippuvuuspaketit ja rakentaa mvgw-apin. Kuvassa 26 kuvataan luotua mvgw-apin rakentamisskriptiä

```
#!/bin/bash

SCRIPTDIR=$(cd -P -- "$(dirname -- "$0")" && pwd -P)
BASEPATH=$SCRIPTDIR/rpmbuild
LOG="log/mvgw-api_build_log.txt"

if [[ ! -d "log" ]]; then
    mkdir log
fi

echo "##>> Installing mvgw-api dependencies..." | tee -a $LOG
sudo yum -y install rpm-build |  tee -a $LOG
echo "##>> Installation of dependencies complete." |  tee -a $LOG



if [[ ! -d "$SCRIPTDIR/rpmbuild" ]]; then
    echo "##>> Creating rpmbuild tree..." |  tee -a $LOG
    mkdir -p $SCRIPTDIR/rpmbuild/{BUILD,BUILDROOT,RPMS,SOURCES,SPECS,SRPMS} |  tee -a $LOG
    echo "##>> Rpmbuild tree created." |  tee -a $LOG
fi

echo "##>> Copying .spec file and SOURCES to rpmbuild folders..." |  tee -a $LOG
if [[ -e "$SCRIPTDIR/mvgw-api.spec" ]]; then
    echo "##>> Found $SCRIPTDIR/mvgw-api.spec copying to $BASEPATH/SPECS" |  tee -a $LOG
    cp $SCRIPTDIR/mvgw-api.spec $BASEPATH/SPECS |  tee -a $LOG
fi
if [[ -d "$SCRIPTDIR/SPECS" ]]; then
    echo "##>> Found SPECS directory copying to $BASEPATH/SPECS" |  tee -a $LOG
    cp -a $SCRIPTDIR/SPECS/. $BASEPATH/SPECS |  tee -a $LOG
fi
if [[ -d "$SCRIPTDIR/SOURCES" ]]; then
    cp -a $SCRIPTDIR/SOURCES/. $BASEPATH/SOURCES |  tee -a $LOG
fi
echo "###>> Copying of .spec and SOURCES complete." |  tee -a $LOG

echo "###>> Starting mvgw-api build..." |  tee -a $LOG
rpmbuild --define "_topdir $BASEPATH" -ba ${BASEPATH}/SPECS/mvgw-api.spec |  tee -a $LOG
echo "###>> mvgw-api build complete." |  tee -a $LOG


```

Kuva 26: Mvgw-apin rakentamisskripti

4.4 Asennusskriptien lataus

Asennusskriptien lataamisessa on kolme pääkohtaa, jotka kuvataan kuvassa 27. SSH-avaimien siirto, joka mahdollistaa asennusskriptien lataamisen Instan lokaalilta Git-serveriltä. SSH-avaimien käyttöoikeuksien muokkaaminen, jotta SSH-avaimia voi käyttää. Asennusskriptien lataaminen Instan lokaalilta Git-serveriltä käyttäen SSH-avaimia.



Kuva 27: Asennusskriptien lataamisen pääkohdat

Asennusskriptien lataus lokaalilta Git-serveriltä onnistuu vain käyttämällä SSH-yhteyttä. SSH-yhteys luo turvallisen yhteyden tietokoneen ja palvelimen välille, vaikka tietokone olisi suojaamattomassa verkossa. Git-palvelimella olevalle koodivarastolle on määritelty SSH-avain, joka mahdollistaa koodivaraston kloonaamisen käyttäen koodivarastolle määriteltyä SSH-avainta.

Siirretään ssh-avaimet ja ssh config-tiedosto /home/käyttäjänimi/.shh-kansioon. Tämä kansio on SSH:n oletuskansio, josta se etsii SSH-avaimia ja konfiguraatiotiedostoja.

Käyttäjillä on tiedostoihin kolme oikeutta: luku-, kirjoitus- ja suoritusoikeus. Näitä oikeuksia on mahdollista rajoittaa käyttämällä chmod-komentoa. Chmod-komennolla on mahdollista rajoittaa käyttäjän, ryhmän ja muiden oikeuksia johonkin tiedostoon.

SSH-avaimen siirron jälkeen täytyy muuttaa avainten käyttöoikeuksia. SSH-avaimille vaaditaan tietyt käyttöoikeudet. SSH-avaimille vaadituissa oikeuksissa rajoitetaan muiden käyttäjien oikeuksia. Muuten SSH-avaimia käytettäessä tulee virheilmoitus SSH-avaimien liian avoimista oikeuksista.

Oikeudet asetetaan chmod-komennolla. Komennolla asetetaan julkisille avaimille ja config-tiedostolle oikeudet 644, joka tarkoittaa, että käyttäjä voi lukea, kirjoittaa ja muut voivat lukea, kukaan ei voi suorittaa. Yksityisille avaimille asetetaan oikeudet 600, joka tarkoittaa, että vain käyttäjä voi lukea ja kirjoittaa tiedostoon, kenelläkään muulla ei ole oikeuksia lukea tai kirjoittaa tiedostoon.

SSH-avainten käyttöoikeuksien muokkauksen jälkeen voidaan ladata asennusskriptit Git-komennolla:

```
git clone git@mvgw-installer:ext-antti.kayhko/mvgw-installer.git installer
```

Komento lataa asennusskriptit installer-nimiseen kansioon.

Seuraavaksi luodaan skripti, joka automatisoi aikaisemmin käsitellyt askeleet. Skripti siirtää SSH-avaimet käyttäjän .ssh-kansioon. Skripti rajoittaa SSH-avaimien käyttöoikeuksia muille käyttäjille ja ryhmille. Kuvassa 28 kuvataan luodun skriptin koodia.

```
#!/bin/bash
# Gets the user who executed the script.
user=$(logname)

SSH="/home/$user/.ssh"
echo "##>> Copying ssh keys..."
cp -vr .ssh /home/$user/
chmod 644 $SSH/config
chmod 644 $SSH/Build_mvgw_deploy_key.pub
chmod 644 $SSH/mvgw-install_deploy_key.pub
chmod 600 $SSH/Build_mvgw_deploy_key
chmod 600 $SSH/mvgw-install_deploy_key
echo "##>> Copying of ssh keys complete."

echo "##>> Starting the creation of installer..."

echo "##>> Downloading install scripts..."

# git clone git@build-mvgw:ext-antti.kayhko/build-mvgw-dependencies.git builder

# mvgw-install repo: https://ids-gitlab.ids.local/ext-antti.kayhko/mvgw-installer.git

git clone git@mvgw-installer:ext-antti.kayhko/mvgw-installer.git installer
echo "##>> Download of install scripts complete."
```

Kuva 28 Asennusskriptien lataus lokaalista Git-koodivarastosta

Kuvassa 28 kuvattu skripti käyttää logname-komentoa sisään kirjautuneen käyttäjän käyttäjänimen hankintaan. Muuten ajettaessa skriptiä sudo-komennolla siirtyvät ssh-avaimet /home/root/.ssh-kansioon. Näin sisään kirjautuneen käyttäjä ei pysty käyttämään siirrettyjä SSH-avaimia, koska ssh-avaimet ovat väärässä kansiossa ja sisään kirjautuneella käyttäjällä ei ole oikeuksia root-käyttäjän SSH-avaimiin.

Kuvassa 28 kuvatun Git-komennon ”mvgw-installer” osa määritellään SSH:n config-tiedostosta. Jos SHH:n config-tiedostoa ei ole määritelty oikein, ei Git-komento toimi.

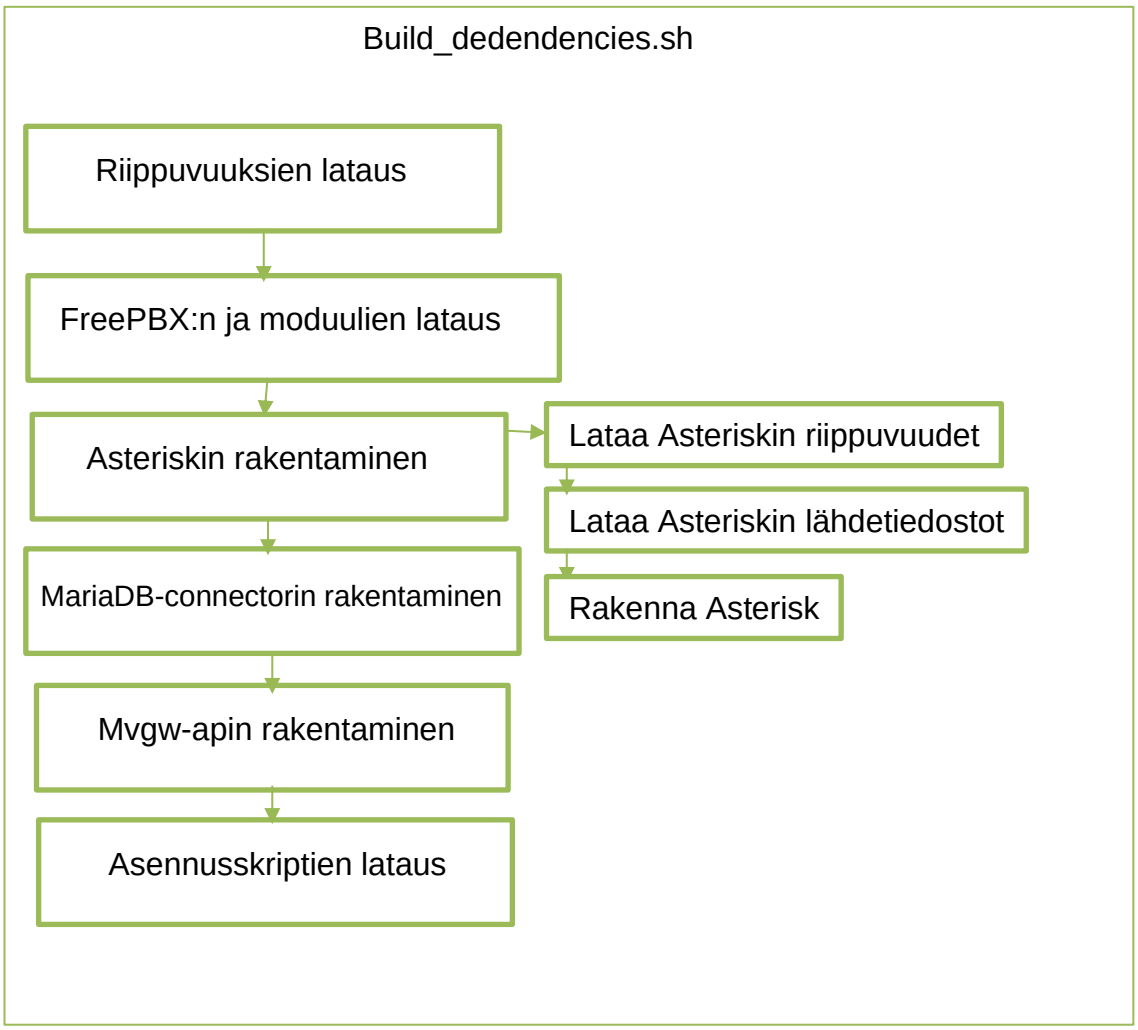
```
1  Host build-mvgw
2    HostName ids-gitlab.ids.local
3    User git
4    IdentityFile=/home/fpbx/.ssh/Build_mvgw_deploy_key
5    IdentitiesOnly yes
6
7  Host mvgw-installer
8    HostName ids-gitlab.ids.local
9    User git
10   IdentityFile=/home/fpbx/.ssh/mvgw-install_deploy_key
11   IdentitiesOnly yes
```

Kuva 29: SSH-konfiguraatiotiedosto

Kuvassa 29 näytetään SHH-konfiguraatiotiedostoa, jossa konfiguroidaan ”mvgw-installer” osa Git-komennosta.

4.5 Skriptien yhdistäminen yhden skriptin alle

Tähän mennessä on esitetty jokaisessa alaluvussa skripti automatisoimaan kyseisen alaluvun askeleet. Tässä luvussa yhdistetään kaikki aikaisemmin esitetyt skriptit yhden skriptin alle. Luodaan skripti, joka ajaa kaikki muut luodut skriptit oikeassa järjestyksessä, jotta tuloksena on FreePBX-asennuspaketti, joka sisältää kaikki tarvittavat tiedostot. Kuvassa 30 esitetään työssä tehdyn skriptin rakennetta.



Kuva 30: Build_dependencies.sh skripti

Ensimmäisenä ladataan FreePBX-asennuksen riippuvuudet, koska FreePBX:n riippuvuuksia rakentaessa täytyy asentaa rakentamiseen vaaditut riippuvuuspaketit tietokoneelle. Jos FreePBX:n riippuvuuspaketteja ladatessa on jokin tarvittavista paketeista jo asennettuna, ei Yum lataa pakettia, koska se on jo asennettuna tietokoneella. Tämä johtaa siihen, että kaikkia FreePBX:n riippuvuuksia ei ladata ja luotava asennuspaketti ei toimi puuttuvien riippuvuuksien takia

Ladataan seuraavaksi FreePBX ja sen moduulit. Ajetaan skripti sisään kirjautuneena käyttäjänä mahdollisten käyttöoikeusongelmien vuoksi.

Seuraavaksi ajetaan skriptit riippuvuuksien rakentamiseksi. Skriptit asentavat riippuvuuksien rakentamiseen vaaditut riippuvuuspaketit, lataavat tarvittavat lähdetiedostot ja rakentavat kyseisen riippuvuuden. Rakennetaan Asterisk, MariaDB-connector ja mvgw-api. Rakentamisen jälkeen siirretään luodut rpm-tiedostot Rakentajan rpms-kansioon. Kuvissa 31 ja 32 esitetään luodun skriptin koodia ja rakennettujen rpm-tiedostojen kopiointia rpms-kansioon, josta ne siirretään luotavaan asennuspakettiin.

FreePBX:n riippuvuuksien rakentamisen jälkeen ladataan asennusskriptit. Ajetaan asennusskriptien latausskripti sisään kirjautuneena käyttäjänä, jotta lataukseen tarvitut SSH-avaimet siirtyvät oikean käyttäjän .ssh-kansioon. Jos skriptiä ei ajettaisi sisään kirjautuneena käyttäjänä, siirtyisivät SSH-avaimet root-käyttäjän .ssh-kansioon. Kuvassa 31 kuvataan skriptin koodia, jossa ladataan asennusskriptit käyttäen niille tehtyä skriptiä.

Lopuksi siirretään luodut rpm-tiedostot asennuspaketin rpms-kansioon. Siirretään Freepbx-paketti ja sen riippuvuudet asennuspaketin packages-kansioon. Kuvassa 32 näytetään koodia rpm-tiedostojen ja FreePBX-tiedostojen siirrosta.

```
#!/bin/bash
LOG="log/dependency-build-log.txt"

if [[ ! -d "log" ]]; then
    mkdir log
fi

echo "##>> Downloading dependency rpms for FreePBX..." | tee -a $LOG
sudo bash dl-packages.sh | tee -a $LOG
echo "##>> Download of rpms complete." | tee -a $LOG

echo "##>> Downloading Freepbx and its modules..." | tee -a $LOG
sudo -u $(logname) bash dl-freepbx.sh | tee -a $LOG
echo "##>> Download of FreePBX and its modules complete." | tee -a $LOG

echo "##>> Installing git..."
sudo yum -y install git
echo "##>> Installation of git complete."

echo "##>> Starting the build of dependencies." | tee -a $LOG

echo "##>> Starting Asterisk build." | tee -a $LOG
bash dependencies/Asterisk/build_asterisk.sh | tee -a $LOG
echo "##>> Asterisk build finished." | tee -a $LOG

echo "##>> Starting Mariadb connector build." | tee -a $LOG
bash dependencies/Mariadb-connector/build_mariadb_connector.sh | tee -a $LOG
echo "##>> Mariadb connector build finished." | tee -a $LOG

echo "##>> Starting mvgw api build." | tee -a $LOG
bash dependencies/mvgw-api/build_mvgw-api.sh | tee -a $LOG
echo "##>> Mvgw api build finished." | tee -a $LOG

echo "##>> Build of dependencies finished." | tee -a $LOG

if [[ ! -d "rpms" ]]; then
    echo "##>> No rmps folder found. Creating rpms folder..." | tee -a $LOG
    mkdir rmps | tee -a $LOG
    echo "##>> Folder rpms created." | tee -a $LOG
fi

echo "##>> Copying created rpms to rpm folder..." | tee -a $LOG
if [[ -d "dependencies/Asterisk/rpmbuild/RPMS/noarch/" ]]; then
    cp -a dependencies/Asterisk/rpmbuild/RPMS/noarch/. rpms | tee -a $LOG
else
    echo "The build of asterisk failed or created rmps not found at dependencies/Asterisk/rpmbuild/RPMS/noarch/." | tee -a $LOG
fi
```

Kuva 31 Build_depenencies-skripti

```
if [[ -d "dependencies/Asterisk/rpmbuild/RPMS/x86_64/" ]]; then
    cp dependencies/Asterisk/rpmbuild/RPMS/x86_64/asterisk-16.16.0-1.el7.x86_64.rpm rpms | tee -a $LOG
    cp dependencies/Asterisk/rpmbuild/RPMS/x86_64/asterisk-sqlite-16.16.0-1.el7.x86_64.rpm rpms | tee -a $LOG
    cp dependencies/Asterisk/rpmbuild/RPMS/x86_64/asterisk-mysql-16.16.0-1.el7.x86_64.rpm rpms | tee -a $LOG
    cp dependencies/Asterisk/rpmbuild/RPMS/x86_64/asterisk-odbc-16.16.0-1.el7.x86_64.rpm rpms | tee -a $LOG

else
    echo "The build of asterisk failed or created rmps not found at dependencies/Asterisk/rpmbuild/RPMS/x86_64/." | tee -a $LOG
fi

if [[ -d "dependencies/Mariadb-connector/rpmbuild/RPMS/x86_64/" ]]; then
    cp -a dependencies/Mariadb-connector/rpmbuild/RPMS/x86_64/mariadb-connector-odbc-3.1.11-1.el7.x86_64.rpm rpms | tee -a $LOG

else
    echo "The build of Mariadb-connector failed or created rmps not found at dependencies/Mariadb-connector/rpmbuild/RPMS/x86_64/." | tee -a $LOG
fi

if [[ -d "dependencies/mvgw-api/rpmbuild/RPMS/noarch/" ]]; then
    cp -a dependencies/mvgw-api/rpmbuild/RPMS/noarch/. rpms | tee -a $LOG

else
    echo "The build of mvgw-api failed or created rpms not found at dependencies/mvgw-api/rpmbuild/RPMS/noarch/." | tee -a $LOG
fi

echo "##>> Copying of created rpms complete. Copied files can be found at $(pwd -P)/rpms" | tee -a $LOG

echo "##>> Build of dependencies complete." | tee -a $LOG

# Executes script as logged in user, who launched the build_dependencies script.
sudo -u $(logname) bash dl-installer.sh | tee -a $LOG

echo "##>> Copying files to installer..."
cp -a rpms/. installer/rpms

cp -a fpbx_downloads/*gz installer/packages

# This file fixes FreePBX installation permission issues.
if [[ -e "files/freepbx_chown.conf" ]]; then
    cp files/freepbx_chown.conf installer/packages

fi
echo "##>> Copying of files complete."
echo "##>> Creation of installer complete."
```

Kuva 32 Build_dependencies-skriptin asennuspaketin luonti

4.6 FreePBX-asennuspaketin rakentaja

Edellisessä luvussa esiteltiin pääskripti, joka luo FreePBX-asennuspaketin. Tämä skripti on osa rakentajaa, jolla täytyy olla jokin kansiorakenne, johon skripti vie ja tuo tiedostoja. Kuvataan seuraavaksi rakentajan rakennetta.

Rakentajalla on yksinkertainen rakenne: pääkansiossa on kolme kansiota: dependencies, files ja rpms-kansio. Dependencies-kansiossa on jokaiselle FreePBX:n rakennettavalle riippuvuudelle kansio, joka sisältää skriptit riippuvuuden rakentamiseksi ja riippuvuuden rakentamiseen tarvittavat tiedostot.

Rpms-kansioon kopioidaan rakennetut rpm-tiedostot, josta ne myöhemmin kopioidaan ladattuihin asennusskripteihin. Tästä kansiosta saa helposti myös kaikki tuotetut rpm-tiedostot ilman, että täytyy niitä hakea yksi kerrallaan jokaiselta riippuvuudelta.

Files-kansio sisältää asennuspaketin aikana luodut lokitiedostot ja muita sekalaisia tiedostoja, kuten listan FreePBX:n moduuleista, jonka mukaan moduulit ladataan.

4.6.1 Rakentajan käyttö

Rakentaja vaatii toimiakseen tuoreen CentOS 7 Linux -jakelun, johon ei ole asennettu muita paketteja kuin jakelun mukana tulleet paketit. Tietokoneella täyttyy olla internetyhteys ja tietokoneen päivän täytyy olla lähellä nykyistä päivää sertifikaattien vanhenemisen vuoksi. Vanhentuneilla sertifikaateilla ei ole mahdollista ladata joitakin FreePBX:n asennuksen vaatimia tiedostoja, mikä voi johtaa luotavan asennuspaketin hajoamiseen.

Rakentajaa on helppo käyttää: Ladataan rakentaja Instan lokaalilta Git-serveriltä CentOS 7 -tietokoneelle. Siirrytään rakentajan kansioon ja ajetaan rakentaja komennolla:

```
sudo bash build_dependencies.sh
```

Tämä komento aloittaa FreePBX-asennuspaketin luonnin. Komennon ajon jälkeen rakentajan kansiossa on installer-kansio. Tämä on luotu FreePBX-asennuspaketti.

4.6.2 FreePBX:n asentaminen käyttäen asennuspakettia

Luotua FreePBX-asennuspakettia käyttäen FreePBX:n asennuskoneelle on helppoa. Siirretään Installer-kansio CenOS 7 -tietokoneelle. Siirrytään installer-kansioon käyttämällä komentoa:

```
cd installer
```

Asennetaan install.sh-tiedosto suoritettavaksi käyttämällä seuraavaa komentoa:

```
chmod +x install.sh
```

Aloitetaan FreePBX-asennus ajamalla seuraava komento:

```
sudo ./install.sh
```

FreePBX:n asennus alkaa ja kysyy asennukseen tarvittavia tietoja. Annetaan tarvitut tiedot, ja kun asennus kysyy, haluatko aloittaa asennuksen, vastataan kyllä.

5 Yhteenveto ja jatkokehitys

Tämä insinöörityö tehtiin Installe. Installa on tarve luoda uusi FreePBX-asennuspaketti asiakkaalle aina, kun FreePBX-asennuksesta luodaan uusi versio. Aikaisemmin FreePBX:n asennuspaketit täytyi joka kerta luoda käsin. FreePBX-asennuspakettien luonti käsin on hidasta ja turhaa työtä, joka on mahdollista korvata skriptillä, joka luo asennuspaketit nopeammin ja helpommin.

Tarkoituksena oli kehittää skripti FreePBX-asennuspakettien automaattiseen rakentamiseen. Luotu skripti pakkaa kaikki FreePBX:n asentamiseen vaadittavat tiedostot, jotta asennuspaketilla on mahdollista asentaa FreePBX tietokoneelle ilman internetyhteyttä.

Luotavaa skriptiä ajetaan CentOS 7 Linux -jakelulla, joten skripti on ajettavissa Linux-ympäristössä. Skripti kehitettiin Bash-komentotulkille. Bash-komentotulkki valittiin, koska Bash-komentotulkki sisältyy suureen osaan Linux-jakeluista. Näin saadaan mahdollistettua luodun skriptin käyttö suuressa osassa Linux-jakeluista.

Ensimmäiseksi pakattiin FreePBX:n ladattavat riippuvuudet hyväksikäyttäen yum-komentoa. Seuraavaksi pakattiin FreePBX ja sen moduulit. Samalla pakattiin myös pm2-moduulin npm-riippuvuudet ja pakattiin npm- ja npm-cache-toiminnallisuus yhteen pakettiin pm2-moduulin riippuvuuksien kanssa.

Luotiin rakennettaville FreePBX:n riippuvuuksille skriptit, joilla on kolme vaihetta: Ensiksi asennetaan rakentamiseen vaaditut riippuvuudet. Seuraavaksi skripti lataa rakentamisesta puuttuvat lähdetiedostot skriptille määritellyn version mukaan. Lopuksi skripti rakentaa riippuvuudet käyttäen rpmbuild-komentoa.

Seuraavaksi kuvattiin asennusskriptien lataamista lokaalista Git-koodivarastosta käyttäen SSH-avaimia. Tämän jälkeen pakattiin kaikki luodut rpm-tiedostot ja kaikki muut FreePBX-asennuksen tarvitsemat tiedostot asennusskriptien kanssa.

Lopuksi kuvattiin luodun FreePBX-asennuspakettien rakentajan rakennetta ja sitä, miten rakentajaa on mahdollista käyttää. Viimeiseksi kuvattiin lyhyesti luodun FreePBX-asennuspaketin käyttöä.

Insinöörityössä luotiin rakentaja, joka mahdollistaa FreePBX-asennuspakettien luonnin yhden skriptin ajamalla. Rakennetulla asennuspaketilla on mahdollista asentaa FreePBX ajamalla yksi skripti haluamalleen CentOS 7 -tietokoneelle.

Rakentajan kehitystä voi jatkaa luomalla rakentajalle yhden config-tiedoston, josta haetaan halutut pakettiversiot eri FreePBX-riippuvuuksille. On myös mahdollista jatkaa rakentajan kehitystä, että rakentaja osaa version mukaan muuttaa .spec-tiedostoja ja muutettujen tiedostojen mukaan rakentaa riippuvuudet.

Lähteet

1. Private branch exchange. Wikipedia. Verkkoaineisto. <https://en.wikipedia.org/wiki/Business_telephone_system#Private_branch_exchange> Päivitetty 3.9.2021. Luettu 31.10.2021.

2. Asterisk (PBX). Wikipedia. Verkkoaineisto. <https://en.wikipedia.org/wiki/Asterisk_(PBX)> Päivitetty 3.7.2021. Luettu 31.10.2021.

3. FreePBX. FreePBX. Verkkoaineisto. <https://www.freepbx.org/> Luettu 31.10.2021.

4. FreePBX. FreePBX. Verkkoaineisto. <https://www.freepbx.org/get-started/> Luettu 31.10.2021.

5. FreePBX. FreePBX add-ons. Verkkoaineisto. <https://www.freepbx.org/add-ons/> Luettu 31.10.2021.

6. Harshit Baluja. 2021. 10 Best Free Open Source PBX Software Solutions. Verkkoaineisto. <https://techstorify.com/best-open-source-pbx-software/> Luettu 31.10.2021.

7. Edward C. Bailey. 2000. Maximum RPM. Verkkoaineisto. <http://ftp.rpm.org/max-rpm/s1-rpm-depend-manual-dependencies.html> Luettu 31.10.2021.

8. Edward C. Bailey. 2000. Maximum RPM. Verkkoaineisto. <http://ftp.rpm.org/max-rpm/ch-intro-to-rpm.html#S1-INTRO-TO-RPM-WHAT-ARE-PACKAGES> Luettu 31.10.2021.

9. Edward C. Bailey. 2000. Verkkoaineisto. <http://ftp.rpm.org/max-rpm/s1-intro-to-rpm-package-management-how.html> Luettu 31.10.2021.

10. Edward C. Bailey. 2000. Verkkoaineisto. <http://ftp.rpm.org/max-rpm/s1-rpm-build-starting-build.html> Luettu 31.10.2021.

11. Bob Cromwell. How to Create and Use Patch Files for RPM Packages. Verkkoaineisto. <https://cromwell-intl.com/open-source/rpm-patch.html> Luettu 31.10.2021.

12. Edward C. Bailey. 2000. Verkkoaineisto. < http://ftp.rpm.org/max-rpm/s1-rpm-build-creating-spec-file.html> Luettu 31.10.2021.

13. Petr Kovář. 2012. Packager's Guide. Verkkoaineisto. <https://docs.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/Packagers_Guide/sect-Packagers_Guide-Creating_a_Basic_Spec_File.html> Luettu 31.10.2021.

14. MariaDB Server: The open source relational database. MariaDB foundation. Verkkoaineisto. <https://mariadb.org/> Luettu 31.10.2021.

15. Remi. 2015. FAQ. Verkkoaineisto. <https://blog.remirepo.net/pages/English-FAQ> Luettu 31.10.2021.

16. SK. Nux Dextop: A Desktop And Multimedia Oriented RPM Repository. Verkkoaineisto. <https://www.unixmen.com/nux-dextop-a-desktop-and-multimedia-oriented-rpm-repository/> Luettu 31.10.2021.

17. Extra Packages for Enterprise Linux (EPEL). EPEL. Verkkoaineisto. <https://docs.fedoraproject.org/en-US/epel/> Luettu 31.10.2021.

18. swaraj. 2017. npm-cache. Verkkoaineisto. <https://www.npmjs.com/package/npm-cache> Luettu 31.10.2021.

19. James Finstrom. FreePBX Chown Conf. Verkkoaineisto. <https://wiki.freepbx.org/display/fop/freepbx+chown+conf> Luettu 31.10.2021.

20. mpg123 - Fast console MPEG Audio Player and decoder library. mpg123. Verkkoaineisto. <https://www.mpg123.de/> Luettu 31.10.2021.