OpenStack Federation met Nikhef en Sara

HPC Cloud

rijksuniversiteit groningen

snow
UNIX SPECIALISTS

Metagegevens

Opdrachtgever

Naam	Organisatie
Henk-Jan Zilverberg	RuG CIT

Opdrachtnemer

Naam	Organisatie

Belanghebbenden

Naam	Organisatie	Rol	Belang

Reviewers namens belanghebbenden

Naam	Organisatie	Rol

Auteurs

Naam	Organisatie	Rol
Rein van Weerden	Snow	Consultant

Documenthistorie

Versie	Datum	Status
		nieuw

Goedkeurder

Naam	Organisatie	Akkoord

Management samenvatting

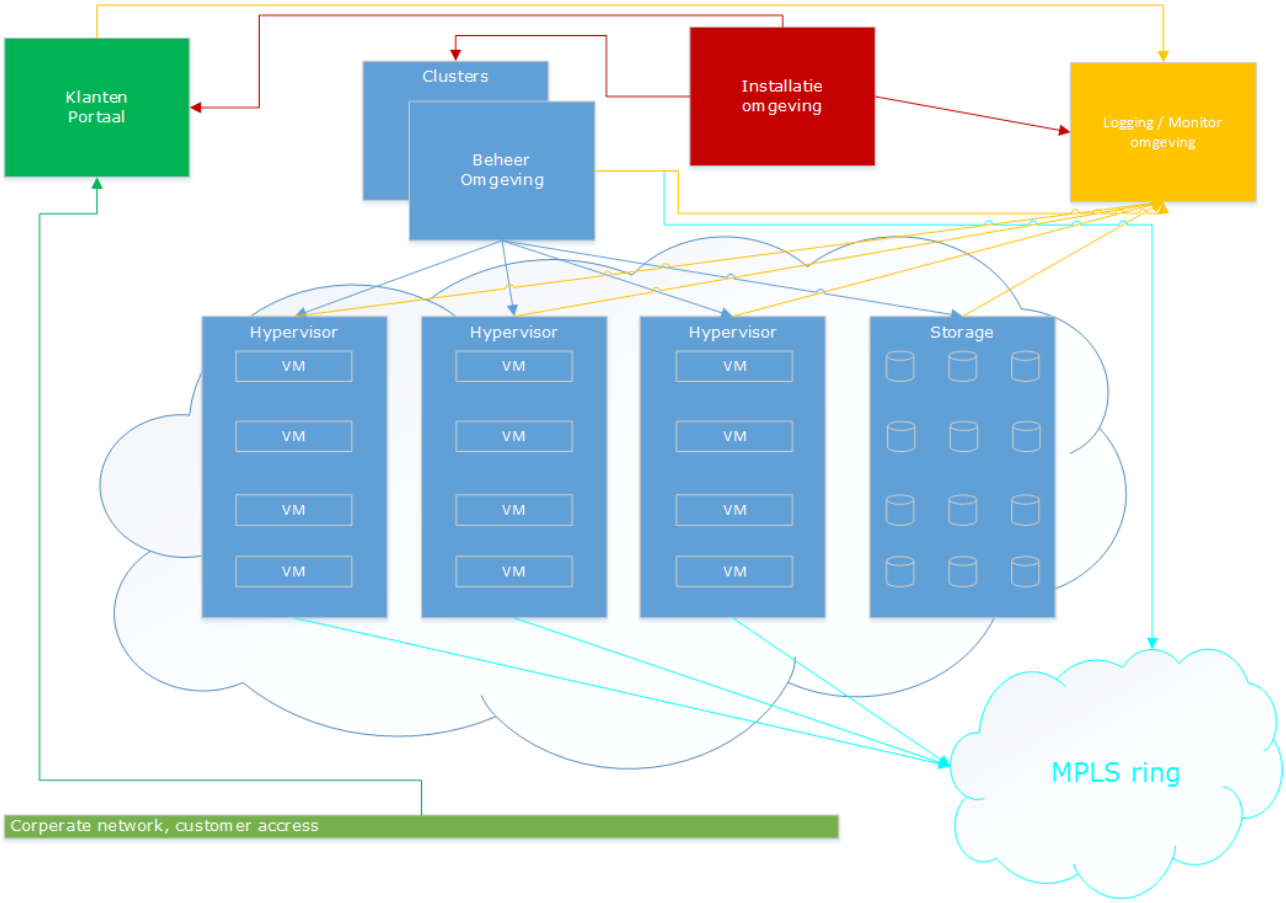
Het project “HPC Cloud” is geïnitieerd om een betere en soepelere inzet te bewerkstelligen van de High Performance Computing hardware. Mede door de cloud, doormiddel van federatie, te koppelen met andere organisaties. Hierdoor kan er optimaal gebruikt worden gemaakt van de hardware en in samenwerking met andere organisaties, de hardware van de partners. De virtuele machines, die bij de partner organisatie worden geplaatst blijven echter bij het project dat de virtuele machines in gebruik heeft en worden niet door de partner organisatie in beheer genomen.

Basis Architectuur

Een cloud binnen een organisatie vraag om een organisatorische wijziging van de werkwijze. De werkwijze binnen een beheer afdeling zorgt vaak voor een uitlever tijd van meerdere dagen en een focus op de server wat betreft beheer en backup. Binnen een cloud omgeving kan een server binnen enkele minuten worden uitgeleverd, indien er gebruik gemaakt wordt van vooraf gedefinieerde configuraties. Dit vraagt een goede planning vooraf over wat de klant zou willen doen op de virtuele hosts. De focus van beheer en backup verschuift naar de data en de configuratie.

Een cloud voor een HPC omgeving is in principe niet anders dan een standaard cloud, met die uitzondering dat een cloud environment voor HPC een grotere stabiliteit moet leveren, dan een gemiddelde cloud. Dit vraagt extra aandacht voor hardware en beheer.

Voor een cloud omgeving zijn naast de hypervisor hosts, enkele beheer en installatie hosts noodzakelijk. Het woord 'host' staat in dit geval equivalent aan een hardware machine, virtuele machine of een container. Ook de afkorting 'vm' kan staan voor een Virtuele machine of een container.



Gobale Architectuur

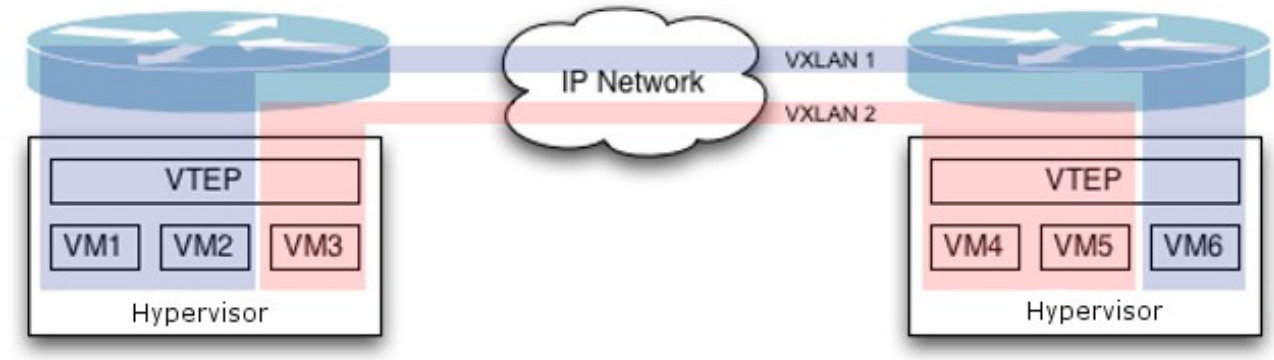
De hypervisor node's zijn voor het ondersteunen van Virtuele Hosts. Deze virtuele hosts worden gebruik voor high performance reken werk. In de nabij toekomst zal hier gebruik van GPU's vereist worden.

Op de beheer machines draaien de cloud beheer tools, van elke tool draaien er normaliter 3 in een cluster. Indien deze tools niet in een container draaien, zijn er 3 bare metal machines of virtuele hosts noodzakelijk. Wanneer deze tools in een container draaien, KAN het aantal bare metal machines gereduceerd worden en kunnen er, in principe, 3 containerised tools op een host draaien, echter dit is geen wenselijke situatie voor een productie omgeving. Voor een ontwikkel dan wel test omgeving is het mogelijk om de containers op een host te draaien. Het is wel van belang om het netwerk gedeelte dan zo op te zetten dat de containers op willekeurige hosts kunnen draaien. (Er zit een verschil in de netwerk structuur, afhankelijk of de containers lokaal of remote draaien).

De installatie hosts zijn niet specifiek voor de cloud, maar bevatten de tools en gegevens om de cloud beheer omgeving te installeren en te onderhouden. Deze hosts hoeven echter niet continue bereikbaar te zijn, uiteraard tijdens de installatie moeten ze wel bereikbaar zijn. Om het beheertijd zo laag mogelijk te

houden, wordt er tijd gestoken in het automatiseren van de installatie en onderhouds taken.

Voor het federated draaien van de cloud is het noodzakelijk om met VxLAN netwerking te draaien. Het is dan mogelijk dat een vm, die in de partner cloud draait, binnen zijn tenant ip range blijft. De authentificatie zal ook federated worden.

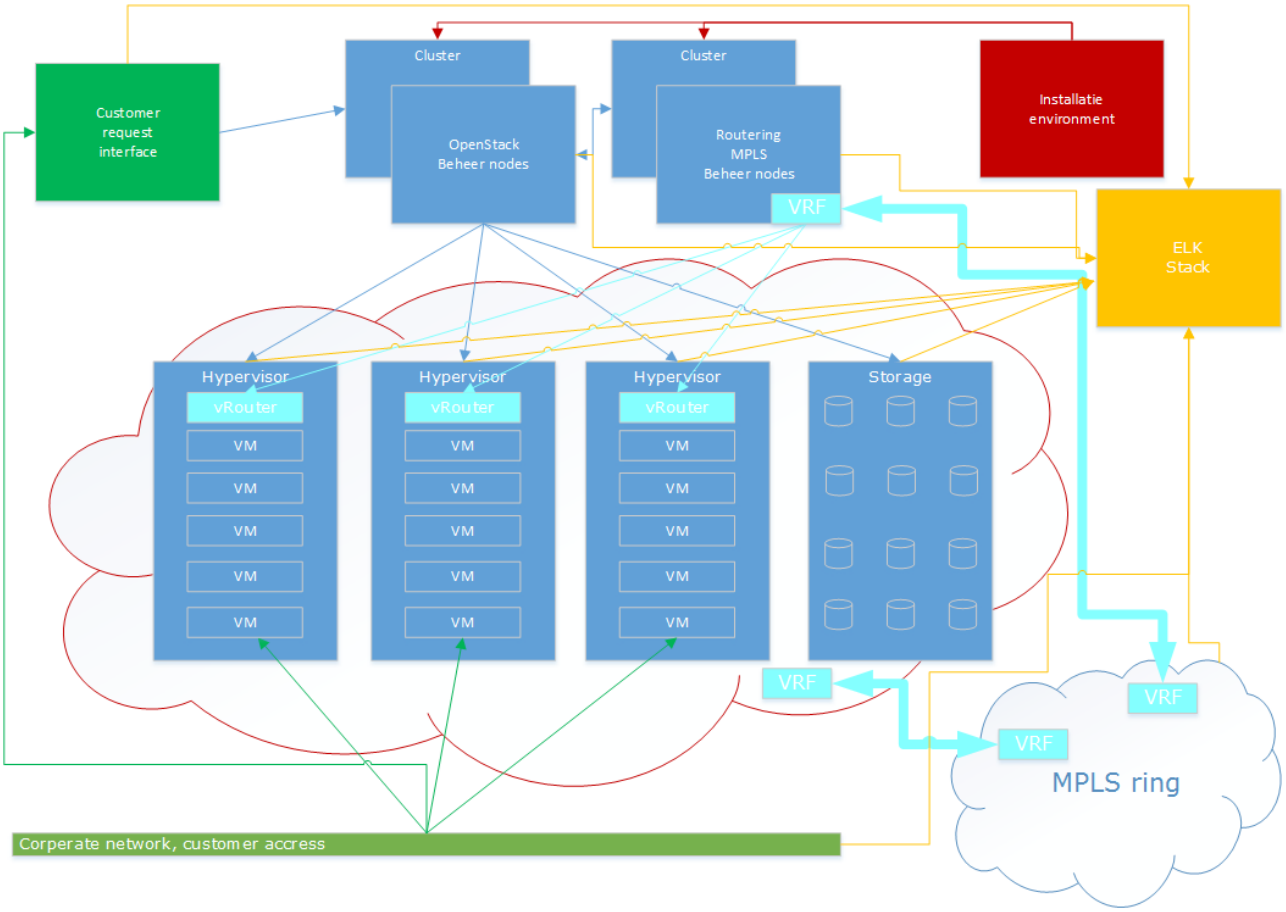


VxLAN

Ontwerp

De keuzes in dit ontwerp zijn gemaakt op basis van de, hier boven omschreven, architectuur.

De installatie en beheer omgeving.
Voor version management is gekozen voor git. Er is een git omgeving binnen de CIT deze heeft echter geen grafische of web based interface. Een grafische interface maakt het mergen van een branch naar een master een stuk eenvoudiger. Hieruit volgt dat er wordt gewerkt in een branch en indien het team de wijzigingen goedkeurt wordt de branch in de master ge-merged, waarna de webhook wordt geactiveerd.
Delen die in git worden opgenomen zijn:
DokerFile's, Ansible playbooks, de hypervisor kernel, de kernel .config, cassandra, de contrail environment, a ubuntu 16.04 (voor docker builds)...

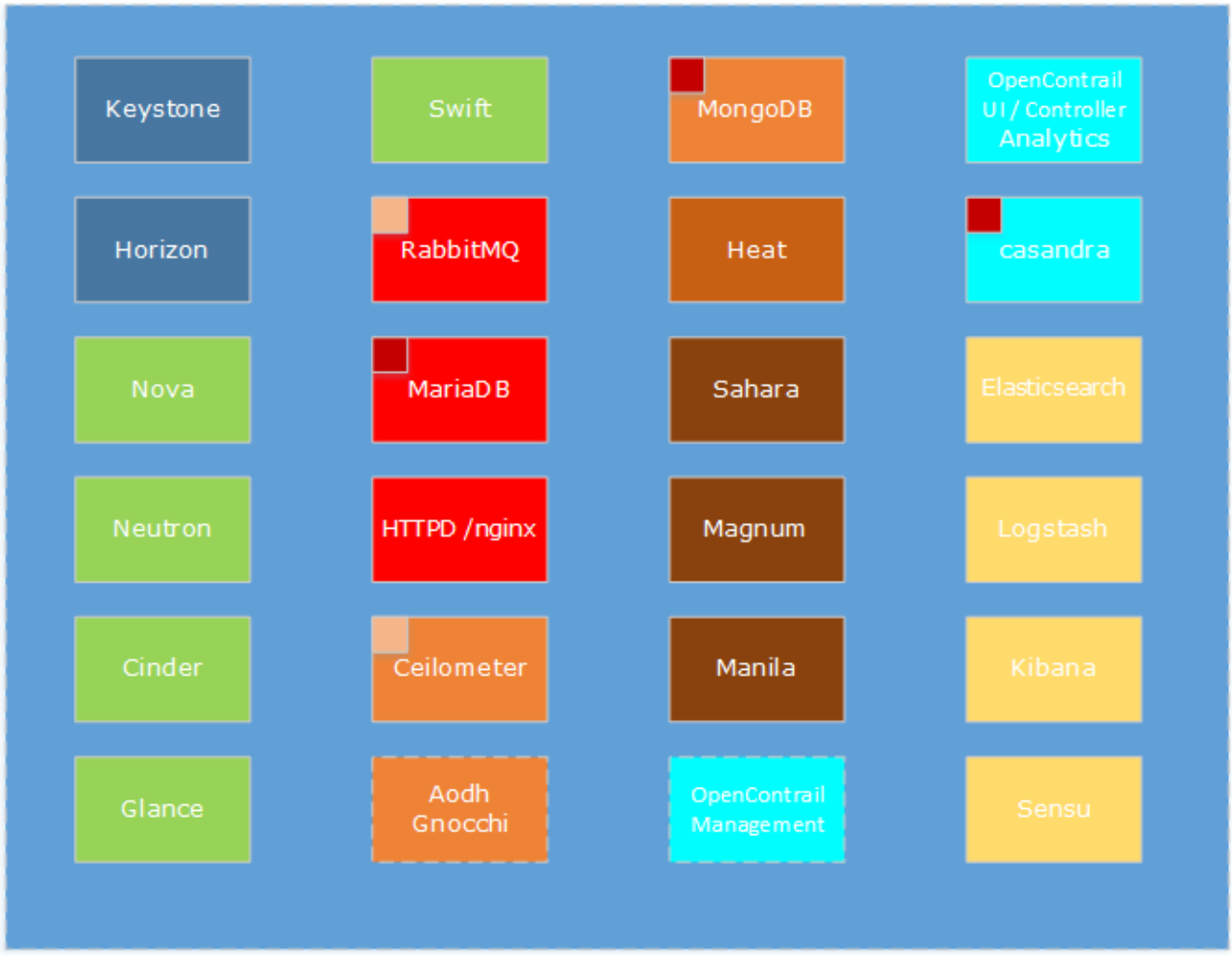


De te bouwen omgeving

Voor de containers wordt een DTR (Docker Trusted Registry) opgebouwd. Deze kan vanuit de community omgeving worden opgebouwd. De DTR bevat de uit de git DockerFile's opgebouwde containers.

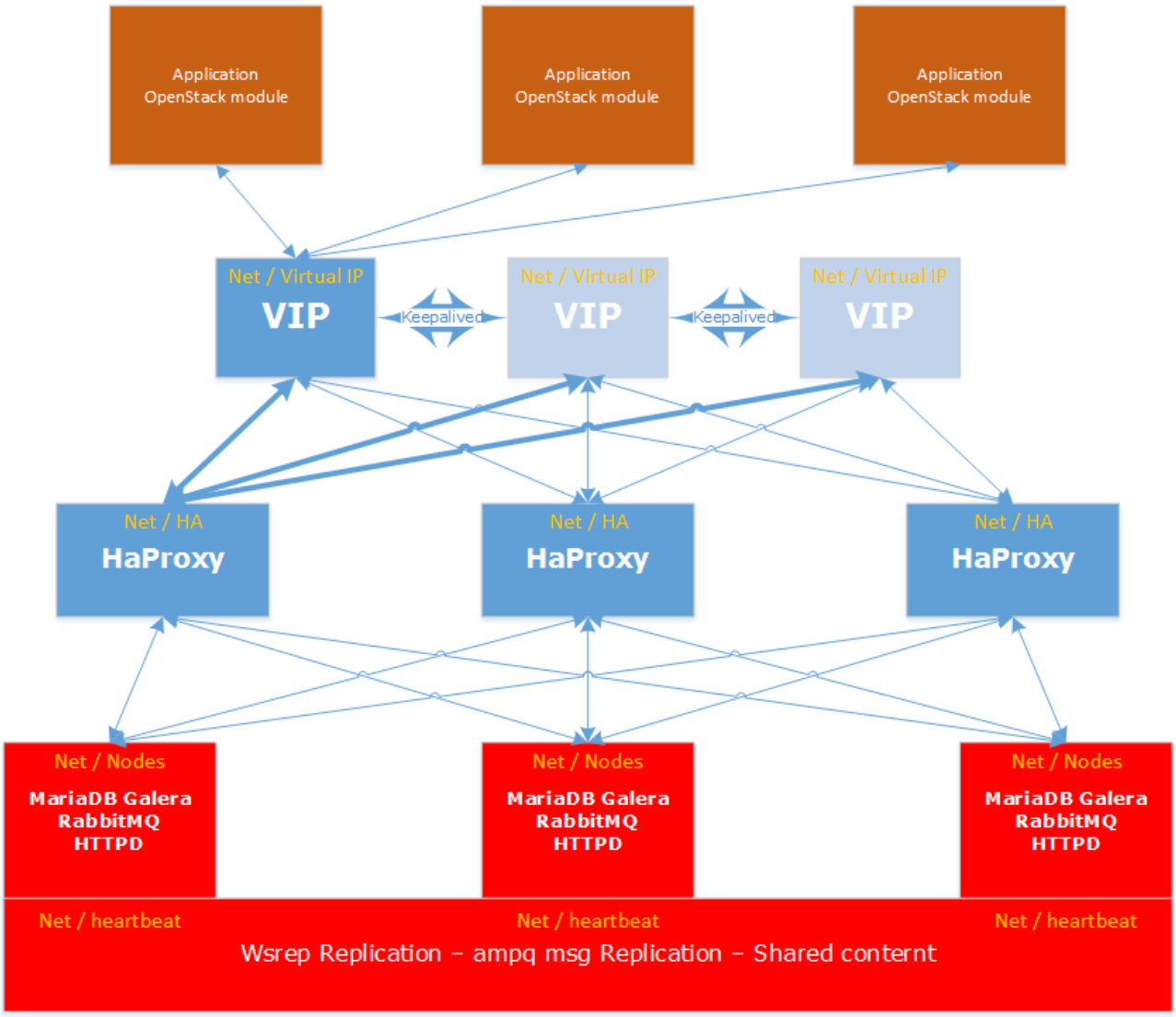
De voornaamste reden voor deze setup, is de stabiliteit van de cloud omgeving. De vm's kunnen snel aangemaakt worden en weer worden verwijdert, echter het fundament zal zo stabiel mogelijk moeten zijn, zonder directe invloed van buitenaf.

De cloud-beheer omgeving.



De te installeren tools

Van de te installeren tools, elk in een container, wordt er 3 maal een set geïnstalleerd over 1 tot 3 stuks hardware.



HAProxy & keepalived

De containers zijn elk doormiddel van haproxy and keepalived, in de nabije toekomst zou deze combinatie vervangen kunnen worden door consul, aan elkaar gekoppeld. Indien er minder dan 1 stuks hardware wordt gebruikt, zal de performance goed in de gaten gehouden moeten worden. Dit geld zeker voor de database servers MariaDB, cassandra en MongoDB, maar ook voor tools met veel I/O, zoals RabbitMQ en Ceilometer.

De keuze voor de samenwerking met partner organisaties geeft als extra dat de 'standaard' SDN (Software Defined Network) van Linux/OpenStack niet kan worden gebruikt. De keuze voor een OpenContrail is gemaakt vanwege de optie tot routering vs switching, die maakt het makkelijker om clouds te federeren. In samenwerking met een MPLS (Multiprotocol Label Switching) ring en VxLAN (Virtual Extensible LAN (Local Area Network)).

Standaard draait de cloud beheer tooling op 3 baremetal machines. Contrail maakt gebruik van een routerings database en deze is standaard cassandra. De cassandra configuratie voor contrail is een cluster van 3 nodes. En voor contrail management 3 nodes, control, analytics en configuratie management. Dit zijn in totaal 9 nodes. Door gebruik te maken van containers kan dit aantal terug gebracht worden tot 1 a 2 hosts, met die uitzondering van deze hosts voldoende capaciteit bezitten om zoveel 'zware' programma's te draaien. Voor ontwikkeling zijn twee hosts, voor de beheer tooling, voldoende. Voor test/acceptatie is het wenselijk is om voor minimaal 3 a 4 hosts te gaan. Voor een productie omgeving zijn, waarschijnlijk, 6 of meer hosts nodig.

De hypervisor omgeving.
De keuze voor contrail zorgt er voor dat er een kernel module, vrouter.ko, op de hypervisor moeten worden geïnstalleerd. Deze module heeft twee netwerk paden nodig een naar de controller en een naar het database cluster. Voor het gebruik van GPU (Graphics Processing Unit) slicing, binnen KVM (Kernel-based Virtual Machine), is een van de laatste kernel's nodig. De minimale Kernel versie is 4.7, de 4.11 kernel is stabieler echter vanaf 4.10 is het geheugen management gewijzigd. Indien deze Kernel gebruikt gaat worden dan is het noodzakelijk dat de contrail vrouter.ko voor deze kernel gebouwd wordt.

De installatie van de cloud-beheer tools.
Alle cloud-beheer tools worden doormiddel van docker containers geinstalleerd, Dit is de eerste serie stappen van een ansible playbook waarna er in de daarop volgende stappen een configuratie aan gekoppeld.
De uitzonderingen zijn de gedistribueerde delen van Nova en Neutron.

Van de cloud-beheer tools word de configuratie als een template file(s) opgeslagen in het des betreffende ansible playbook.
Het playbook voert als eerste enkele mysql/mariadb opdrachten uit. Zoals het aanmaken van de database en het zetten van de permissies voor de gebruiker.

```
CREATE DATABASE keystone;
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
IDENTIFIED BY 'KEYSTONE_DBPASS';
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
IDENTIFIED BY 'KEYSTONE_DBPASS';
```

Na het installeren van de container en het van template naar config file wijzigen, moeten er nog een aantal commando's worden uitgevoerd.

```
su -s /bin/sh -c "keystone-manage db_sync" keystone
keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone
keystone-manage credential_setup --keystone-user keystone --keystone-group
keystone
```

Hierna kan de beheer-tool gebootstraped worden.

```
keystone-manage bootstrap --bootstrap-password ADMIN_PASS \
  --bootstrap-admin-url http://controller:35357/v3/ \
  --bootstrap-internal-url http://controller:5000/v3/ \
  --bootstrap-public-url http://controller:5000/v3/ \
  --bootstrap-region-id RegionOne
```

Deze stappen zijn voor alle beheer-tools vergelijkbaar en zullen in het betreffende playbook worden opgenomen.

Default ansible playbook layout.

```
.
|____development
|____group_vars
| |____all
| |____group1
| |____group2
|____host_vars
| |____hostname1
| |____hostname2
|____production
|____README.md
|____roles
| |____common
| | |____defaults
| | | |____main.yml
| | |____files
| | | |____bar.txt
| | | |____foo.sh
| | |____handlers
| | | |____main.yml
| | |____meta
| | | |____main.yml
| | |____tasks
| | | |____main.yml
| | |____templates
| | | |____ntp.conf.j2
| | |____vars
| | | |____main.yml
| |____roleset1
| | |____defaults
| | | |____main.yml
| | |____files
| | | |____bar.txt
| | | |____foo.sh
| | |____handlers
| | | |____main.yml
| | |____meta
| | | |____main.yml
| | |____tasks
| | | |____main.yml
| | |____templates
| | | |____ntp.conf.j2
| | |____vars
| | | |____main.yml
| |____roleset2
| | |____defaults
| | | |____main.yml
| | |____files
| | | |____bar.txt
| | | |____foo.sh
| | |____handlers
| | | |____main.yml
| | |____meta
```

```
|  |  |  |____main.yml
|  |  |____tasks
|  |  |  |____main.yml
|  |  |____templates
|  |  |  |____ntp.conf.j2
|  |  |____vars
|  |  |  |____main.yml
|  |____roleset3
|  |  |____defaults
|  |  |  |____main.yml
|  |  |____files
|  |  |  |____bar.txt
|  |  |  |____foo.sh
|  |  |____handlers
|  |  |  |____main.yml
|  |  |____meta
|  |  |  |____main.yml
|  |  |____tasks
|  |  |  |____main.yml
|  |  |____templates
|  |  |  |____ntp.conf.j2
|  |  |____vars
|  |  |  |____main.yml
|____servergroup1.yml
|____servergroup2.yml
|____site.yml
```

DockerFile example

Keystone DockerFile:

```
FROM ubuntu:16.04

RUN apt-get -y install git \
    python-setuptools \
    python-pip \
    python-lxml \
    python-greenlet-devel \
    python-ldap \
    sqlite-devel \
    openldap-devel \
    gcc \
    python-devel

WORKDIR /root
RUN git clone https://git.hpc.mgmt.rug.nl/openstack/keystone

WORKDIR /root/keystone
RUN pip install -r requirements.txt
RUN python setup.py install
RUN mkdir -p /etc/keystone
RUN cp etc/keystone.conf.sample /etc/keystone/keystone.conf
RUN cp etc/keystone-paste.ini /etc/keystone/
RUN cp etc/policy.json /etc/keystone/

WORKDIR /

EXPOSE 5000 35357

CMD keystone-manage --config-file /etc/keystone/keystone.conf db_sync && \
    keystone-all --config-file /etc/keystone/keystone.conf
```

Deze DockerFile wordt opgeslagen in git.
Als er een wijziging wordt ge-commit dan wordt er een webhook geactiveerd.
Deze webhook kan dat het volgende set commando's activeren:

```
git clone <url>/dockerostools/keystone.git
docker build -t testing/keystone - < Dockerfile
```

Het resultaat wordt opgeslagen in de DTR.

Vanuit de DTR kan deze dan worden gedistribueerd.

Om een test te starten, wordt het volgende commando gebruikt:

```
docker run -d -p 5000:5000 -p 35357:35357 --name keystone1 testing/keystone
```

Om te testen of the keystone container werkt kan via curl of httpie connectie gemaakt worden met de container. BV:

```
http <Keystone container URL of IP adres>:5000
```

Het resultaat zal dan vergelijkbaar zijn met:

```
HTTP/1.1 300 Multiple Choices
Content-Length: 753
Content-Type: application/json
Date: Fri, 06 Jun 2014 19:35:05 GMT
Vary: X-Auth-Token

{
    "versions": {
        "values": [
            {
                "id": "v3.0",
                "links": [
                    {
                        "href": "http://localhost:5000/v3/",
                        "rel": "self"
                    }
                ],
                "media-types": [
                    {
                        "base": "application/json",
                        "type": "application/vnd.openstack.identity-v3+json"
                    },
                    {
                        "base": "application/xml",
                        "type": "application/vnd.openstack.identity-v3+xml"
                    }
                ],
                "status": "stable",
                "updated": "2013-03-06T00:00:00Z"
            },
            {
                "id": "v2.0",
                "links": [
                    {
                        "href": "http://localhost:5000/v2.0/",
                        "rel": "self"
                    },
                    {
                        "href": "http://docs.openstack.org/",
                        "rel": "describedby",
                       "type": "text/html"
                }
                ],
                "media-types": [
                    {
                        "base": "application/json",
                        "type": "application/vnd.openstack.identity-v2.0+json"
                    },
                    {
                        "base": "application/xml",
                        "type": "application/vnd.openstack.identity-v2.0+xml"
                    }
                ],
                "status": "stable",
```

```
                "updated": "2014-04-17T00:00:00Z"
            }
        ]
    }
}
```

RabbitMQ configuratie directory

Let op: De node namen, zo als in dit voorbeeld, moeten opzoekbaar zijn en dus voorkomen in de DNS of host file.

/etc/rabbitmq/rabbitmq.config

```
[
    {mnesia,                    [{dump_log_write_threshold, 1000}]},
    {rabbit,                    [{tcp_listeners, [{"192.168.<Not
Heartbeat>.214",5672}]},
                                    {cluster_nodes,
{['rabbit@r1hb','rabbit@r2hb','rabbit@r3hb'], disc}}]
                            },
    {rabbitmq_management,       [{listener, [{ip, "192.168.<Not Heartbeat>.214"},
{port, 15672}]}]},
    {rabbitmq_stomp,            [{tcp_listeners, [{"192.168.<Not
Heartbeat>.214",61613}]}]}
].
```

/etc/rabbitmq/enabled_plugins

```
[rabbitmq_federation,rabbitmq_federation_management,rabbitmq_jsonrpc,rabbitmq_js
onrpc_channel,rabbitmq_management,rabbitmq_management_agent,rabbitmq_management_
visualiser,rabbitmq_mqtt,rabbitmq_priority_queue,rabbitmq_shovel,rabbitmq_shovel
_management,rabbitmq_stomp,rabbitmq_tracing].
```

/etc/rabbitmq/rabbitmq-env.conf

```
# Comment lines start with a hash character.
# This is a /bin/sh script file - use ordinary envt var syntax

# use only one of 'NODENAME' and 'NODE_IP_ADDRESS'
NODENAME=rabbit@r1hb
#NODE_IP_ADDRESS=192.168.<not heartbeat>.214

CONFIG_FILE=/etc/rabbitmq/rabbitmq
```

Maak van een standaard cassandra een cluster.
Hiervoor
Stop het cassandra programma
```
systemctl stop cassandra
```

Maak de systeem tabellen leeg
```
sudo rm -rf /var/lib/cassandra/data/system/*
```

Pas de cassandra configuratie file aan. Deze staat standaard in de "/etc/cassandra" directory en heet "cassandra.yaml"
```
vim cassandra.yaml
```
(Use the editor you like, 'namo', 'joe', 'vi' of 'vim' of …)

Verander de cluster name "`cluster_name: '<name>'`" naar, bij voorbeeld, '`RuG HPC cloud Contrail cluster 01`'

Voeg the IP adressen toe aan de seeds
```
seed_provider:
      - class_name: org.apache.cassandra.locator.SimpleSeedProvider
        parameters:
             - seeds: "your_server_ip,your_server_ip_2,...your_server_ip_n"
```

Seeds wordt dan
```
             - seeds: "192.168.70.101,192.168.70.102,192.168.70.103"
```

Wijzig het listen adres
`listen_address: your_server_ip` Naar, bij voorbleed, '`192.168.70.101`'

Doe met het
`rpc_address: your_server_ip` Het zelfde

Zorg dat de endpoint_snitch praat via het Gossip protocol
```
endpoint_snitch: GossipingPropertyFileSnitch
```

En voeg aan het eind van de .yaml file de lijn
```
auto_bootstrap: false
```
toe.

Standaard staat er in cassandra-rackdc.properties file een dc en een rack waarde van `dc=dc1 en rack=r1`. Deze worden door het Gossip protocol gebruikt. Hier moet een zinvolle waarde in gezet worden. Bijvoorblled:
```
dc=dcHPCRuG
rack=ContrailRack
```

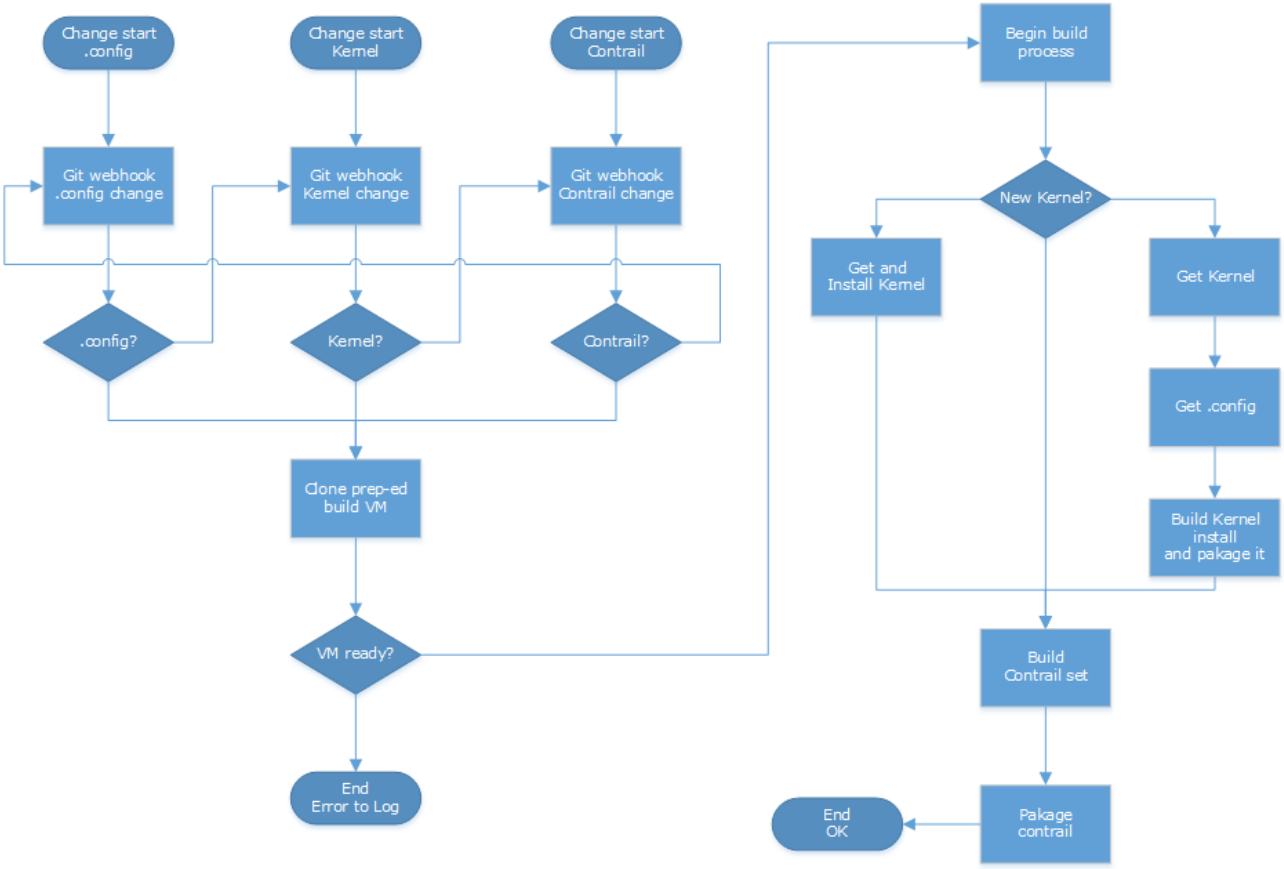
De file cassandra-topology.properties bevat summiere topologie gegevens. Deze gegevens worden als 'backup' worden gebruikt. Indien cassandra juist is geconfigureerd zal de inhoud niet worden gebruik. Voor de juistheid van de installatie worden er wel de juiste gegevens in gezet.

```
# Cassandra Node IP=Data Center:Rack
192.168.70.101=dcHPCRuG:ContrailRack
192.168.70.102=dcHPCRuG:ContrailRack
192.168.70.103=dcHPCRuG:ContrailRack

# default for unknown nodes
#default=DC1:r1
default=dcHPCRuG:ContrailRack
```

Building

De hypervisor environment draait op een kernel versie hoger dan 4.7.x. De Contrail vRouter is een Kernel module en het is noodzakelijk dat deze op de Kernel wordt gebouwd die ook gebruikt gaat worden.



Binnen gogs wordt een webhook aan gemaakt die op een aangewezen bouw host een executabel start "http://<Bouw server address>/cgi-bin/build". In dit geval is de executabel een bash script.

```
#!/bin/bash
#
# Gemaakt voor RuG-CIT 2017-05-27                 Door: Rein van Weerden
# ========================================================================
# build                                       Copyright dVR H&S Sys. 2017
# ========================================================================
#
# build is used as a webhook building script to build a 'new' kernal
# in a virtualbox
```

```
#
#     ____
#    <.   .\
#      \..|
#       \ |___(\--/)
#     __/     (  . . )
#    "'._.     '-.O.'
#         '-.  \ "|\
#            '.,,/'.,     dVR
#
if [[ -n ${DX} ]]
 then
  exec 1>>/tmp/${0##*/}.log 2>&1
  set -${DX}    # Are you going to debug? (DX is set to 'x' (set DX='x') or
unset)
fi
#

DATUM=$(/bin/date '+%a %d %b (%V) %Y %H:%M:%S %Z')
EPOCH=$(/bin/date '+%s')
echo "Content-type: text/html"
echo ""
echo "<html>"
echo "  <head>"
echo "    <title>"
echo "      Build webhook script"
echo "    </title>"
echo "  </head>"
echo "  <body>"
JQ=$(which jq)
JQ=${JQ:-"/usr/local/bin/jq"}
if [[ ! -x "${JQ}" ]]
 then
  echo "Command 'jq' not found, bailing out"
  echo "  </body>"
  echo "</html>"
  exit 1
fi
TR=$(which tr)
TR=${TR:-"/usr/bin/tr"}
if [[ ! -x "${TR}" ]]
 then
  echo "Command 'tr' not found, bailing out"
  echo "  </body>"
  echo "</html>"
  exit 1
fi

# The '/Users/rein/build' directory should be writeble by the user '_www'

INPUT=$(cat)
NAME=$(echo ${INPUT} |${JQ} '.repository.name')
FULL_NAME=$(echo ${INPUT} |${JQ} '.repository.full_name')
DEFAULT_BRANCH=$(echo ${INPUT} |${JQ} '.repository.default_branch')
CLONE_URL=$(echo ${INPUT} |${JQ} '.repository.clone_url')
SSH_URL=$(echo ${INPUT} |${JQ} '.repository.ssh_url')
VERSION=$(echo ${INPUT} |${JQ} '.repository.default_branch')
```

```
D=$(echo "${DATUM}" |${TR} -d '"')
N=$(echo "${NAME}" |${TR} -d '"')
F=$(echo "${FULL_NAME}" |${TR} -d '"')
B=$(echo "${DEFAULT_BRANCH}" |${TR} -d '"')
C=$(echo "${CLONE_URL}" |${TR} -d '"')
S=$(echo "${SSH_URL}" |${TR} -d '"')
V=$(echo "${VERSION}" |${TR} -d '"')
echo "Handed build of '${N}' over to 'rein' on ${D}"
if [[ ${N} == *"Kernel"* ]]
 then
  echo "go build a ${N} branch '${B}' you need also a .config"
elif [[ ${N} == *"build"* ]]
 then
  echo "to use a .config you need also the Kernel repo."
else
  echo "to use the contrail-vnc you need also a 4.7 or greater Kernel."
fi
echo "to get the ${N} do: git clone '${S} --branch ${B}'"

echo "${D}" > /tmp/flg.build
echo "${N}" >> /tmp/flg.build
echo "${B}" >> /tmp/flg.build
echo "${S}" >> /tmp/flg.build
echo "${V}" >> /tmp/flg.build
echo "" >> /tmp/flg.build
echo "${INPUT}" >> /tmp/flg.build

echo "  </body>"
echo "</html>"
exit 0
```

Het bouw script op de bouw host. Dit script wordt via een cron gestart.

```
#!/bin/bash
#
# Gemaakt voor RuG-CIT 2017-05-27                 Door: Rein van Weerden
# ========================================================================
# buildKern                                  Copyright dVR H&S Sys. 2017
# ========================================================================
#
# build is used to check a flag file set by a webhook and if
# go and build a kernel in a virtualbox
#
#     ____
#   <.  .\
#     \..|
#      \ |___(\--/)
#    __/    (  . . )
#   "'._.    '-.O.'
#        '-.  \ "|\
#           '.,,/'.,     dVR
#
if [[ -n ${DX} ]]
 then
  exec 1>>/tmp/${0##*/}.log 2>&1
  set -${DX}    # Are you going to debug? (DX is set to 'x' (set DX='x') or unset)
fi
#
d ${HOME}/build
```

```
if [[ ! -f /tmp/flg.build ]]
 then
  exit
else
  if [[ -f flg.building ]]
   then
    echo "Busy with a build. Try again later"
    exit
  elif [[ -f flg.building.busy ]]
   then
    echo "There is, probably, a VM running with a build..."
    echo "Prees enter to continue or Ctrl-C to quit"
    read
    sudo mv /tmp/flg.build flg.building
  else
    sudo mv /tmp/flg.build flg.building
  fi
fi
DATUM="$(/bin/date '+%a %d %b (%V) %Y %H:%M:%S %Z')"
EPOCH="$(/bin/date '+%s')"

VBOXMANAGE=$(which VBoxManage)
VBOXMANAGE=${VBOXMANAGE:-"/usr/local/bin/VBoxManage"}
if [[ ! -x "${VBOXMANAGE}" ]]
 then
  echo "Command 'VBoxManage' not found, bailing out"
  exit 1
fi

CLONENAME=$(/usr/local/bin/uuidgen -t)
CLONENAME=${CLONENAME%%-*}

cat << EOF > .gitconfig
[user]
    name = plReynaerde
    email = plreynaerde@dvrhss.net
[push]
    default = simple
[filter "media"]
    clean = git media clean %f
    smudge = git media smudge %f
    required = true
[http]
    sslVerify = false

EOF

REPONAME=$(cat flg.building |sed -n '2p')
BRANCHNAME=$(cat flg.building |sed -n '3p')
CONFIGFILE=$(cat flg.building |sed -n '4p')
VERSION=$(cat flg.building |sed -n '5p')
# CONFIGFILE=$(echo ${CONFIGFILE} |sed -e 's/:2242//')
KERNELSET=${CONFIGFILE%/*}
KERNELSET=${KERNELSET}/Kernel.git

cat << EOF > getAndBuild
#!/bin/bash
#
# Gemaakt voor RuG-CIT 2017-05-27                 Door: Rein van Weerden
# ========================================================================
# getAndBuild                                Copyright dVR H&S Sys. 2017
# ========================================================================
#
# getAndBuild is used to check a flag file set by a webhook and if
```

```
# go and build a kernel in a virtualbox
#
#     ____
#    <.   .\\
#      \\..|
#       \\ |___(\\--/)
#     __/     (  . . )
#    "'._.     '-.O.'
#          '-.  \\ "|\\
#             '.,,/'.,     dVR
#
if [[ -n \${DX} ]]
 then
  exec 1>>/tmp/\${0##*/}.log 2>&1
  set -\${DX}    # Are you going to debug? (DX is set to 'x' (set DX='x') or unset)
fi
#

function logDate {
  DAT=\$(date)
  FIRST=\${2}
  if [[ -z "\${FIRST}" ]]
   then
    echo "\${DAT} getAndBuild: \${1}" >> /home/rein/getAndBuild.run.time
  else
    echo "\${DAT} getAndBuild: \${1}" > /home/rein/getAndBuild.run.time
  fi
}

logDate Start first
cd /home/rein/bin
mv mvGet2here mvGet2here.done
sudo mv /media/sf_VM-sfolder/build/vr_genetlink.c.new /home/rein/src
sudo chown rein:rein /home/rein/src/vr_genetlink.c.new
sudo chmod a-wx,a+r,u+w /home/rein/src/vr_genetlink.c.new
cd /home/rein
echo 'debconf debconf/frontend select Noninteractive' | sudo debconf-set-selections
sudo dpkg --configure -a
sudo apt-get update
# sudo apt-get -y install autoconf automake bison debhelper flex libcurl4-openssl-dev
libexpat1-dev libgettextpo0 libprotobuf-dev libtool libxml2-utils make protobuf-compiler
python-all python-lxml python-setuptools python-sphinx ruby-ronn scons unzip vim-common
python-netsnmp librdkafka-dev librdkafka1
sudo apt-get -y install autoconf automake bison debhelper flex libcurl4-openssl-dev
libexpat1-dev libgettextpo0 libprotobuf-dev libtool libxml2-utils make protobuf-compiler
python-all python-lxml python-setuptools python-sphinx ruby-ronn scons unzip vim-common
python-netsnmp
sudo apt-get -y install python-dev
sudo apt-get -y install libfixbuf3-dev
sudo apt-get -y install xz-utils cpio pax
sudo apt-get -y install libncurses5-dev
sudo apt-get -y install libssl-dev
sudo mv /media/sf_VM-sfolder/build/libssl1.0.0_1.0.2g-1ubuntu4_amd64.deb .
sudo apt-get -f install
sudo apt-get -y install git
sudo apt-get -y install g++ g++-multilib
# sudo apt-get -y install gcc-4.9-plugin-dev
sudo apt-get -y install gcc-5-plugin-dev
sudo apt-get -y --no-install-recommends install dselect build-essential fakeroot
sudo apt-get -y install libboost-dev libboost-chrono-dev libboost-date-time-dev libboost-
filesystem-dev libboost-program-options-dev libboost-python-dev libboost-regex-dev
libboost-system-dev libcurl4-openssl-dev google-mock libgoogle-perftools-dev
liblog4cplus-dev libtbb-dev libhttp-parser-dev libxml2-dev libicu-dev
# sudo apt-get -y install libuv1 libuv1-dev
```

```
sudo apt-get -y install libpcap-dev
sudo apt-get -y install libnl-3-200 libnl-3-dev libnl-genl-3-200 libnl-genl-3-dev
sudo apt-get -y install cmake cmake-data cmake-extras
sudo apt-get -y install w3m
sudo apt-get -y install repo
sudo apt-get -y install nmon
sudo apt-get -y -f install
sudo apt-get update
sudo apt-get -y dist-upgrade
sudo apt-get -y autoremove
sudo apt-get -y --no-install-recommends install kernel-package
sudo rm -f /media/sf_VM-sfolder/build/dpkg-gencontrol
sudo mv -f /media/sf_VM-sfolder/build/kernel-pkg.conf /etc/
sudo chmod a+r /etc/kernel-pkg.conf
logDate "apt-get's done"
MAKEOPTIES="--jobs \$(nproc) V=1 -d"
mkdir -p src/Linux
cd src/Linux
git clone ${KERNELSET}
logDate "Kernel clone done"
git clone ${CONFIGFILE}
logDate ".config cloned"
cd ~/src/Linux/build
ln -s ../Kernel source
if make \${MAKEOPTIES} prepare > make.prepare.lst 2>&1
 then
  logDate "make prepare done"
  if make \${MAKEOPTIES} modules > make.modules.lst 2>&1
   then
    logDate "make modules done"
    if make \${MAKEOPTIES} all > make.all.lst 2>&1
     then
      logDate "make all done"
      if sudo make \${MAKEOPTIES} modules_install > make.modules_install.lst 2>&1
       then
        logDate "make modules_install done"
        if sudo make \${MAKEOPTIES} headers_install > make.headers_install.lst 2>&1
         then
          logDate "make headers_install done"
          if sudo make \${MAKEOPTIES} install > make.install.lst 2>&1
           then
            logDate "make install done"
            VERSION=\$(cat include/config/kernel.release)
            cd /boot
            sudo update-grub
            sudo update-initramfs -c -k \${VERSION}
            date >> /home/rein/getAndBuild.run.time
            logDate "update-initramfs -c -k \${VERSION} done"
            sudo update-grub
            sudo update-initramfs -u -k \${VERSION}
            logDate "update-initramfs -u -k \${VERSION} done"
            sudo update-grub
          fi
        fi
      fi
    fi
  fi
fi
logDate "all makes done"
cd ~/src/Linux/build
KR=\$(cat include/config/kernel.release)
# KR=\$(uname -r)
echo \${KR} > ~/kernel.version
sudo mkdir -p /usr/src/linux-headers-\${KR}
```

```
sudo cp -a * /usr/src/linux-headers-\${KR}/
cd /usr/src/
sudo chown -R root:root linux-headers-\${KR}
date >> /home/rein/getAndBuild.run.time
logDate "headers copied"
echo "JuniperControllerBuild" > ~/next
logDate "finisched"
sleep 5
sudo reboot
EOF

chmod 777 getAndBuild
mkdir -p ~/VM-sfolder/build
cp getAndBuild ~/VM-sfolder/build
cp .gitconfig ~/VM-sfolder/build
cp dpkg-gencontrol ~/VM-sfolder/build
cp kernel-pkg.conf ~/VM-sfolder/build
cp vr_genetlink.c.new ~/VM-sfolder/build
#cp JuniperVRouterBuild ~/VM-sfolder/build
cp openssl-1.1.0f.tar.gz ~/VM-sfolder/build
cp libipfix_110209.tgz ~/VM-sfolder/build
cp JuniperControllerBuild ~/VM-sfolder/build
cp mpackages.xml ~/VM-sfolder/build
cp *.patch ~/VM-sfolder/build
cp doScons ~/VM-sfolder/build
cp doPack ~/VM-sfolder/build


BUILDERNAME="build-${CLONENAME}"
UUID=$(${VBOXMANAGE} showvminfo builder|grep 'Hardware UUID:'|cut -f5 -d' ')
${VBOXMANAGE} clonevm --mode all --name "${BUILDERNAME}" --register "${UUID}"
${VBOXMANAGE} startvm "${BUILDERNAME}" --type headless
echo "To connet to the VM use: ssh \$(getVMIPAddress ${BUILDERNAME})"

sudo rm -f .gitconfig getAndBuild
mv flg.building flg.building.busy
```

Op de voorbereide VM host staan enkele scripts om het proces automatisch te starten. In de root environment staat het script "`mvCMDs`". Dit script wordt door een cron task uitgevoerd. De scripts "`doGetAndBuild`" en "`doNext`", staan in de build eigenaars environment en worden tevens door een cron task uitgevoerd.

De root cron task ziet er als volgt uit.

```
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

MAILTO=""

* * * * * ~/bin/mvCMDs >/dev/null 2>&1
```

De build eigenaars cron task ziet er als volgt uit.

```
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

MAILTO=""

* * * * * ~/bin/doNext >/dev/null 2>&1
```

De scripts worden elke minuut gestart. Er is geen logging voor de scripts.

Het "mvCMDs" script

```
#!/bin/bash
#
# Gemaakt voor RuG-CIT 2017-06-10                 Door: Rein van Weerden
# ========================================================================
# mvCMDs                                      Copyright dVR H&S Sys. 2017
# ========================================================================
#
# mvCMDs is used to check, if not Busy, if a script "getAndBuild"
# is on a share...
# If so mv it to the build users bin and set the next file...
#
#     ____
#    <.  .\
#      \..|
#       \ |___(\--/)
#     __/    (  . . )
#    "'._.    '-.O.'
#         '-.  \ "|\
#            '.,,/'.,     dVR
#
if [[ -n ${DX} ]]
 then
  exec 1>>/tmp/${0##*/}.log 2>&1
  set -${DX}    # Are you going to debug? (DX is set to 'x' (set DX='x') or unset)
fi
#

export USER=rein
export HOME=/home/${USER}

GCMD=".gitconfig"
DCMD="getAndBuild"
GFILE="/media/sf_VM-sfolder/build/${GCMD}"
DFILE="/media/sf_VM-sfolder/build/${DCMD}"

if [[ ! -f ${HOME}/Busy ]]
 then
  LIST=$(sudo ls ${DFILE})
  if [[ "${LIST}" == "${DFILE}" ]]
   then
    mv ${GFILE} ${HOME}
    if [[ -f ${HOME}/${GCMD} ]]
     then
      chown ${USER}:${USER} ${HOME}/${GCMD}
      chmod 444 ${HOME}/${GCMD}
    fi

    mv ${DFILE} ${HOME}/bin/
    if [[ -f ${HOME}/bin/${DCMD} ]]
     then
      chown ${USER}:${USER} ${HOME}/bin/${DCMD}
      chmod ug+rx,a-w,o-rx ${HOME}/bin/${DCMD}
      echo ${DCMD} > ${HOME}/next
    fi

    for CMD in $(file /media/sf_VM-sfolder/build/*|grep 'shell script'|grep executable|
cut -f1 -d' '|tr -d ':')
     do
```

```
      mv ${CMD} ${HOME}/bin/
      NAME=${CMD##*/}
      if [[ -f ${HOME}/bin/${NAME} ]]
       then
        chown ${USER}:${USER} ${HOME}/bin/${NAME}
        chmod ug+rx,a-w,o-rx ${HOME}/bin/${NAME}
      fi
    done

    echo "Busy" > ${HOME}/Busy
    chown ${USER}:${USER} ${HOME}/Busy
    chmod 444 ${HOME}/Busy

  fi
fi
```

Het "doNext" script

```
#!/bin/bash
#
# Gemaakt voor RuG-CIT 2017-07-21                  Door: Rein van Weerden
# ========================================================================
# doNext                                        Copyright dVR H&S Sys. 2017
# ========================================================================
#
# doNext is used to check for the next command to execute.
# It checks for ths file ~/next and executes the command written in there
# if it exsists and is executable
#
#    ____
#   <.  .\
#     \..|
#      \ |___(\--/)
#    __/    (  . . )
#   "'._.    '-.O.'
#        '-.  \ "|\
#           '.,,/'.,     dVR
#
if [[ -n ${DX} ]]
 then
  exec 1>>/tmp/${0##*/}.log 2>&1
  set -${DX}    # Are you going to debug? (DX is set to 'x' (set DX='x') or
unset)
fi
#

if [[ -f ~/next ]]
 then
  NXTCMD=$(cat ~/next)
  NEXTCMD="~/bin/${NXTCMD}"
  if [[ -x ${NEXTCMD} ]]
   then
    rm -f ~/next
    exec ${NEXTCMD} > ~/${NXTCMD}.log 2>&1 &
  fi
fi
```

Het eerste “script” dat door doNext wordt uitgevoerd, wordt door “`mvCMDs`” op “`getAndBuild`” gezet. “`getAndBuild`” zet, aan het eind, de volgende stap in '`~/next`'. In dit geval “`JuniperControllerBuild`”.

Het “`JuniperControllerBuild`” script

```
#!/bin/bash -x
#
# Gemaakt voor RuG-CIT 2017-06-10                 Door: Rein van Weerden
# ========================================================================
# JuniperControllerBuild                     Copyright dVR H&S Sys. 2017
# ========================================================================
#
# JuniperControllerBuild is used
# go and build a Contrail Controlller in a virtualbox
#
#     ____
#    <.  .\
#      \..|
#       \ |___(\--/)
#     __/    (  . . )
#    "'._.    '-.O.'
#         '-.  \ "|\
#            '.,,/'.,     dVR
#
if [[ -n ${DX} ]]
 then
  exec 1>>/tmp/${0##*/}.log 2>&1
  set -${DX}    # Are you going to debug? (DX is set to 'x' (set DX='x') or unset)
fi
#

function logDate {
  DAT=$(date)
  FIRST=${2}
  if [[ -z "${FIRST}" ]]
   then
    echo "${DAT} JuniperControllerBuild: ${1}" >>
/home/rein/JuniperControllerBuild.run.time
  else
    echo "${DAT} JuniperControllerBuild: ${1}" >
/home/rein/JuniperControllerBuild.run.time
  fi
}

logDate "Start" "first"
rm -Rf .repo
rm -Rf ~/.repo*
cd /home/rein
echo 'debconf debconf/frontend select Noninteractive' | sudo debconf-set-selections
sudo apt-get update
sudo apt-get -y install pkgconf
sudo apt-get -y install autoconf automake bison debhelper flex libexpat-dev libgettextpo0
libtool libxml2-utils make python-all python-dev python-lxml python-setuptools python-
sphinx ruby-ronn scons unzip vim-common libsnmp-python libfixbuf3 libfixbuf3-dev zlib1g-
dev
sudo apt-get -y install glibc-source
sudo apt-get -y install libssl-dev
cd ~/src
# sudo cp /media/sf_VM-sfolder/build/openssl-1.0.2d.tar.gz .
# sudo chown rein:rein openssl-1.0.2d.tar.gz
# tar xzf openssl-1.0.2d.tar.gz
```

```
# cd openssl-1.0.2d
# ./config --prefix=/usr threads zlib shared
# make all > make.all.lst 2>&1
# find /usr -name "libssl.*" -exec rm -f {} \;
# find /usr -name "libcrypto.*" -exec rm -f {} \;
# sudo make install > make.install.lst 2>&1
cd /usr/lib
# sudo ln -s libssl.so.1.0.0 libssl.so.1.0.0
# sudo ln -s libssl.so.1.0.0 libssl.so.1.0.2
# sudo ln -s libssl.so.1.0.0 libssl.so
# sudo ln -s libcrypto.so.1.0.0 libcrypto.so.1.0.0
# sudo ln -s libcrypto.so.1.0.0 libcrypto.so.1.0.2
# sudo ln -s libcrypto.so.1.0.0 libcrypto.so
sudo cp -a libssl.* libcrypto.* /lib/x86_64-linux-gnu/
mkdir -p ~/src/Juniper/build/lib
sudo cp -a libssl.* libcrypto.* ~/src/Juniper/build/lib
sudo ldconfig
logDate "First set off apt-get's done"
cd ~/src
sudo apt-get -y install libboost-dev libboost-chrono-dev libboost-date-time-dev libboost-
filesystem-dev libboost-program-options-dev libboost-python-dev libboost-regex-dev
libboost-system-dev google-mock libgoogle-perftools-dev liblog4cplus-dev libtbb-dev
libhttp-parser-dev libxml2-dev libicu-dev libpcap-dev libsasl2-dev
sudo apt-get -y install libcurl4-gnutls-dev
sudo apt-get -y install libzookeeper-java libzookeeper2 libzookeeper-mt2 libzookeeper-mt-
dev
sudo apt-get -y install cmake cmake-data cmake-extras
sudo apt-get -y install python-virtualenv python-libxml2 libxslt1-dev
sudo apt-get -y install w3m
sudo apt-get -y install nmon
sudo apt-get -y install python3-pip
sudo pip3 install pip --upgrade
sudo rm -Rf /home/rein/.cache
sudo apt-get -y install python-pip
sudo pip2 install pip --upgrade
sudo apt-get -y install ant default-jdk javahelper libcommons-codec-java libhttpcore-java
sudo apt-get -y install --no-install-recommends nodejs
sudo apt-get update
sudo apt-get -y dist-upgrade
sudo apt-get -y autoremove
logDate "All apt-get's done"
sudo ln -s /usr/bin/libtoolize /usr/bin/libtool
cd ~/src
# git clone https://github.com/tubav/libipfix.git
sudo mv /media/sf_VM-sfolder/build/libipfix_110209.tgz .
sudo chown rein:rein libipfix_110209.tgz
tar xzf libipfix_110209.tgz
cd libipfix_110209/
./configure --prefix=/usr > configure.lst 2>&1
make all > make.all.lst 2>&1
sudo make install > make.install.lst 2>&1
logDate "libipfix done"
cd ~/src
git clone https://github.com/edenhill/librdkafka.git
cd librdkafka
./configure --prefix=/usr
make all > make.all.lst 2>&1
sudo make install > make.install.lst 2>&1
sudo ldconfig
logDate "librdkafka done"
cd /usr/lib
if [[ ! -f libnl-3.so ]]
 then
  sudo ln -s /lib/x86_64-linux-gnu/libnl-3.so.200 libnl-3.so
```

```
fi
if [[ ! -f libnl-genl-3.so ]]
 then
  sudo ln -s /lib/x86_64-linux-gnu/libnl-genl-3.so.200 libnl-genl-3.so
fi
cd ~/src
sudo ldconfig
cat << EOF > ~/.gitconfig.new
[user]
    name = plReynaerde
    email = plreynaerde@dvrhss.net
[push]
    default = simple
[filter "media"]
    clean = git media clean %f
    smudge = git media smudge %f
    required = true
[http]
    sslVerify = false

EOF
sudo mv -f ~/.gitconfig.new ~/.gitconfig
logDate "Extra's done"
cd ~/src
git clone https://github.com/google/protobuf.git -b 3.3.x
cd protobuf/
sed -i 's#curl $curlopts -O https://googlemock.googlecode.com/files/gmock-1.7.0.zip#curl
$curlopts -O http://pkgs.fedoraproject.org/repo/pkgs/gmock/gmock-
1.7.0.zip/073b984d8798ea1594f5e44d85b20d66/gmock-1.7.0.zip#' autogen.sh
./autogen.sh
./configure --prefix=/usr
make > make.all.lst 2>&1
sudo make install > make.install.lst 2>&1
mkdir -p ~/src/Juniper/build/lib
cd ~/src/Juniper/build/lib/
cp -a ~/src/protobuf/src/.libs/* .
rm -f libprotobuf.la libprotobuf-lite.la libprotoc.la
cp -a ~/src/protobuf/src/libprotobuf.la .
cp -a ~/src/protobuf/src/libprotobuf-lite.la .
cp -a ~/src/protobuf/src/libprotoc.la .
sudo ldconfig
logDate "protobuf done"
cd ~/src
mkdir -p casdrvr
cd casdrvr
DISTRO=$(sudo lsb_release -is|tr '[:upper:]' '[:lower:]')
THISONE=$(sudo lsb_release -rs)
ALIST=$(w3m -dump -no-graph http://downloads.datastax.com/cpp-driver/${DISTRO}/$
{THISONE}/dependencies/libuv/|tail -n +6|head -n -2|cut -f2 -d'v'|sed 's/\/.*$//'|tr -d
'.')
for ALN in ${ALIST}
 do
  if [[ $((${ALNO})) -lt $((${ALN})) ]]
   then
    export ALNO=${ALN}
   fi
done
export ALNN=v
for ((i=0; i<${#ALNO}; i++))
 do
  if [[ i -ne 1 ]]
   then
    export ALNN+="${ALNO:$i:1}."
  else
```

```
    export ALNN+="${ALNO:$i:1}"
  fi
done
ALEN=$((${#ALNN} - 1))
AVN=${ALNN:0:${ALEN}}
ADRVLST=$(w3m -dump -no-graph http://downloads.datastax.com/cpp-driver/${DISTRO}/$
{THISONE}/dependencies/libuv/${AVN}/|tail -n +7|head -n -2|awk -F ' ' '{ print $3 }'|grep
-v dbg)
for AFN in ${ADRVLST}
 do
  wget http://downloads.datastax.com/cpp-driver/${DISTRO}/${THISONE}/dependencies/libuv/$
{AVN}/${AFN}
done
sudo dpkg --force-all --install libuv_*
sudo dpkg --force-all --install libuv-dev_*
logDate "libuv done"
cd ~/src
# git clone https://github.com/datastax/cpp-driver.git -b 2.0
git clone https://github.com/datastax/cpp-driver.git
cd cpp-driver/
mkdir -p build
cd build/
cmake .. > cmake..lst 2>&1
make > make.all.lst 2>&1
sudo make install > make.install.lst 2>&1
mkdir -p ~/src/Juniper/build/lib/
cd ~/src/Juniper/build/lib/
cp ~/src/cpp-driver/build/libcassandra.so.2.7.0 .
cp ~/src/cpp-driver/build/libcassandra_static.a .
ln -s libcassandra.so.2.7.0 libcassandra.so.2
ln -s libcassandra.so.2 libcassandra.so
sudo ldconfig
logDate "cpp-driver (libcassandra) done"
cd ~/src
git clone https://gerrit.googlesource.com/git-repo
cd git-repo
sudo cp repo /usr/bin/
logDate "repo installed"
cd ~/src
mkdir -p Juniper
cd Juniper
#repo init -u https://github.com/Juniper/contrail-vnc -b R4.0
repo init -u git@github.com:Juniper/contrail-vnc -b R4.0
#find .repo/manifests -iname "*.xml" -exec sed -i 's/project name="/project
name="Juniper\//' {} \;
logDate "repo init done"
repo sync
logDate "repo sync done"
#sed -i 's#third_party/rapidjson#build/third_party/rapidjson#'
controller/lib/rapidjson/SConscript
#   <package>
#     <name>Apache Thrift 0.8</name>
#     <url>http://archive.apache.org/dist/thrift/0.8.0/thrift-0.8.0.tar.gz</url>
#     <patches>
#       <patch strip="2">thrift_patch1.diff</patch>
#       <patch strip="2">thrift_autoconf.patch</patch>
#     </patches>
#     <format>tgz</format>
#     <md5>d29dfcd38d476cbc420b6f4d80ab966c</md5>
#     <autoreconf>true</autoreconf>
#   </package>
cd third_party
cp packages.xml packages.xml.org
# sed -i 's/0.8.0\/thrift-0.8.0.tar.gz/0.10.0\/thrift-0.10.0.tar.gz/' packages.xml
```

```
# sed -i 's/d29dfcd38d476cbc420b6f4d80ab966c/795c5dd192e310ffff38cfd9430d6b29/'
packages.xml
# sed -i '/thrift_patch1.diff/d' packages.xml
# sed -i '/thrift_autoconf.patch/d' packages.xml
# sed -i '/autoreconf/d' packages.xml
python fetch_packages.py --file packages.xml
# mv thrift-0.10.0 thrift-0.8.0
cd thrift-0.8.0
autoconf
# sudo cp /media/sf_VM-sfolder/build/THRIFT-2039-thrift_config_h.patch .
# sudo chown rein:rein THRIFT-2039-thrift_config_h.patch
# patch -p1 < THRIFT-2039-thrift_config_h.patch
mkdir -p ~/src/Juniper/build/include/
cd ~/src/Juniper/build/include/
# sudo cp /media/sf_VM-sfolder/build/thrift_include.txz .
# sudo chown rein:rein thrift_include.txz
# tar xJf thrift_include.txz
mkdir -p ~/src/Juniper/third_party/thrift-0.8.0/compiler/cpp
cd ~/src/Juniper/third_party/thrift-0.8.0/compiler/cpp
# sudo cp /media/sf_VM-sfolder/build/thrift_include.txz .
# sudo chown rein:rein thrift_include.txz
# tar xJf thrift_include.txz
cd ~/src/Juniper
if [[ -f third_party/.cache/vijava55b20130927src.jar ]]
 then
  if [[ -d ~/src/Juniper/third_party/vijava ]]
   then
    cd ~/src/Juniper/third_party/vijava
    if [[ ! -f README.txt ]]
     then
      unzip ../.cache/vijava55b20130927src.jar
      cd ~/src/Juniper
      patch -p1 < third_party/vijava_patch1.diff
    fi
  fi
fi
cd ~/src/Juniper
if [[ -f third_party/.cache/pugixml-1.2.tar.gz ]]
 then
  if [[ -d ~/src/Juniper/third_party/pugixml ]]
   then
    cd ~/src/Juniper/third_party/pugixml
    if [[ ! -f readme.txt ]]
     then
      tar xzf ../.cache/pugixml-1.2.tar.gz
    fi
  fi
fi
logDate "third party packages done"
cd ~/src/Juniper/third_party
sudo mv /media/sf_VM-sfolder/build/mpackages.xml .
sudo chown rein:rein mpackages.xml
python fetch_packages.py --file mpackages.xml
logDate "third party mpackages done"
export GOROOT=/home/rein/src/Juniper/third_party/go
export GOPATH=/home/rein/go
export GO=${GOROOT}/bin/go
mkdir -p ${GOPATH}/src/github.com/containernetworking
cd ${GOPATH}/src/github.com/containernetworking
rm -Rf cni
git clone https://github.com/containernetworking/cni.git
git clone https://github.com/containernetworking/plugins.git
cd -
${GO} get "github.com/docker/docker/client"
```

```
${GO} get "github.com/natefinch/lumberjack"
${GO} get "github.com/vishvananda/netlink"
${GO} get "github.com/coreos/go-iptables/iptables"
mkdir -p ~/src/Juniper/third_party/go/src
cd ~/src/Juniper/third_party/go/src
ln -s ~/go/src/github.com github.com
mkdir -p ~/src/Juniper/third_party/go/pkg/linux_amd64
cd ~/src/Juniper/third_party/go/pkg/linux_amd64
ln -s ~/go/pkg/linux_amd64/github.com github.com
cd ~/src/Juniper/controller/src/container/cni/cni/common
# sudo cp macvlan.go /media/sf_VM-sfolder/build/macvlan.go.org
# sudo cp /media/sf_VM-sfolder/build/macvlan.go .
# sudo cp interface.go /media/sf_VM-sfolder/build/interface.go.org
# sudo cp /media/sf_VM-sfolder/build/interface.go .
# sudo cp veth.go /media/sf_VM-sfolder/build/veth.go.org
# sudo cp /media/sf_VM-sfolder/build/veth.go .
sudo mv /media/sf_VM-sfolder/build/macvlan.go.patch .
sudo mv /media/sf_VM-sfolder/build/interface.go.patch .
sudo mv /media/sf_VM-sfolder/build/veth.go.patch .
sudo chown rein:rein *.patch
sudo chmod 664 *.patch
for P in $(ls -1 *.patch)
 do
  F=${P%.*}
  patch -lst ${F} ${P}
done
cd ~/go/src/github.com/containernetworking/plugins/vendor
rm -Rf github.com
logDate "go cni done"
# cd ~/src
# git clone https://github.com/Juniper/contrail-controller.git
# cd ~/src/Juniper/controller/src
# rm -Rf analytics
# cp -a ~/src/contrail-controller/src/analytics .
cd ~/src/Juniper/controller/src/analytics
ln -s ../../../build/debug/analytics/analytics_request_skeleton.cpp
analytics_request_skeleton.cpp
sed -i "s/        'nodeinfo',/        'nodeinfo',\n        'dl',/" SConscript
cd ~/src/Juniper
sudo -H pip install generateDS --upgrade
logDate "python update done"
echo "doScons" > ~/next
logDate "finisched"
sleep 5
sudo reboot
```

Het "doScons" script

```
#!/bin/bash
#
# Gemaakt voor RuG-CIT 2017-07-17                 Door: Rein van Weerden
# ========================================================================
# doScons                                      Copyright dVR H&S Sys. 2017
# ========================================================================
#
# doScons is used to build the contrail environment
#
#      ____
#    <.   .\
#       \..|
#        \ |___(\--/)
#      __/     (  . . )
#    "'._.     '-.O.'
#          '-.  \ "|\
```

```
#           '.,,/'.,     dVR
#
if [[ -n ${DX} ]]
 then
  exec 1>>/tmp/${0##*/}.log 2>&1
  set -${DX}    # Are you going to debug? (DX is set to 'x' (set DX='x') or unset)
fi
#

export USER=rein
# export GOROOT=/home/${USER}/src/Juniper/third_party/go
# export GOPATH=/home/${USER}/go
date >> /home/${USER}/doScons.run.time
cd /home/${USER}/src/Juniper
#export C_INCLUDE_PATH="/usr/include"
#export CPLUS_INCLUDE_PATH="/usr/include"
export NUM_CPU="$(nproc)"
if scons -Q debug=1 > /home/${USER}/doScons.log 2>&1
 then
  echo "doPack" > ~/next
fi
date >> /home/${USER}/doScons.run.time
sudo reboot
```

Het "doPack" script

```
#!/bin/bash
#
# Gemaakt voor RuG-CIT 2017-07-24                 Door: Rein van Weerden
# ========================================================================
# doPack                                      Copyright dVR H&S Sys. 2017
# ========================================================================
#
# doPack is used to build the deb packages after scons was completed
#
#     ____
#   <.  .\
#     \..|
#      \ |___(\--/)
#    __/    (  . . )
#   "'._.    '-.O.'
#        '-.  \ "|\
#           '.,,/'.,     dVR
#
if [[ -n ${DX} ]]
 then
  exec 1>>/tmp/${0##*/}.log 2>&1
  set -${DX}    # Are you going to debug? (DX is set to 'x' (set DX='x') or unset)
fi
#

export USER=rein
date >> /home/${USER}/doPack.run.time
cd /home/${USER}/src/Juniper
make -f packages.make all > make.packages.make.all.lst 2>&1
date >> /home/${USER}/doPack.run.time
```

Het "JuniperVRouterBuild" script

```
#!/bin/bash -x
#
# Gemaakt voor RuG-CIT 2017-06-10                 Door: Rein van Weerden
# ========================================================================
# JuniperVRouterBuild                         Copyright dVR H&S Sys. 2017
```

```
# ====================================================================
#
# JuniperVRouterBuild is used to check a 4.11 release kernel and if
# go and build a Contrail VRouter in a virtualbox
#
#     ____
#    <.   .\
#      \..|
#       \ |___(\--/)
#     __/    (  . . )
#    "'._.    '-.O.'
#          '-.  \ "|\
#             '.,,/'.,     dVR
#
if [[ -n ${DX} ]]
 then
  exec 1>>/tmp/${0##*/}.log 2>&1
  set -${DX}    # Are you going to debug? (DX is set to 'x' (set DX='x') or unset)
fi
#
date > /home/rein/JuniperVRouterBuild.run.time
rm -Rf .repo
rm -Rf ~/.repo*
```

Voer de repo sync en scons opdracht uit als user:

```
export USER=<user>
```

Zorg dat een aantal libs op de juiste locatie bereikbaar zijn.

```
cd /usr/lib
if [[ ! -f libnl-3.so ]]
 then
  sudo ln -s /lib/x86_64-linux-gnu/libnl-3.so.200 libnl-3.so
fi
if [[ ! -f libnl-genl-3.so ]]
 then
  sudo ln -s /lib/x86_64-linux-gnu/libnl-genl-3.so.200 libnl-genl-3.so
fi
cd -
sudo ldconfig
```

Maak het bouwen inzichtelijk.

```
MAKEOPTIES="--jobs $(nproc) V=1"
cd ~/src
```

Haal enkele, door Contrail, gebruikte libs op en bouw deze.

```
git clone https://github.com/libuv/libuv.git
cd libuv
./autogen.sh > autogen.lst 2>&1
./configure > configure.lst 2>&1
make ${MAKEOPTIES} > make.all.lst 2>&1
sudo make ${MAKEOPTIES} install > make.install.lst 2>&1
sudo ldconfig
cd ~/src
# git clone https://github.com/datastax/cpp-driver.git -b 2.0
git clone https://github.com/datastax/cpp-driver.git
cd cpp-driver/
mkdir -p build
cd build/
cmake .. > cmake..lst 2>&1
make ${MAKEOPTIES} > make.all.lst 2>&1
sudo make ${MAKEOPTIES} install > make.install.lst 2>&1
mkdir -p ~/src/Juniper/build/lib/
cd ~/src/Juniper/build/lib/
cp ~/src/cpp-driver/build/libcassandra.so.2.7.0 .
cp ~/src/cpp-driver/build/libcassandra_static.a .
ln -s libcassandra.so.2.7.0 libcassandra.so.2
```

```
ln -s libcassandra.so.2 libcassandra.so
sudo ldconfig
cd ~/src

mkdir -p Juniper
cd Juniper
```

Gebruik het repo tool om de Contrail source op te halen. Indien de user geregistreerd is bij github, met een ssh key, kan onderstaande commando regel worden vervangen door "`repo init -u git@github.com:Juniper/contrail-vnc -b R4.0 -m vrouter-manifest.xml`" en kan het "`find`" opdracht komen te vervallen.

```
repo init -u https://github.com/Juniper/contrail-vnc -b R4.0 -m vrouter-manifest.xml
sudo cp /home/rein/.repo/repo/repo /usr/bin/repo
find .repo/manifests -iname "*.xml" -exec sed -i 's/project name="/project
name="Juniper\//' {} \;
```

Synchroniseer de bouw omgeving met de gegevens van 'contrail-vnc'.

```
repo sync
```

Corrigeer enkele locaties en inhoud.

```
sed -i 's/#include "vr_genetlink.h"/#include "genetlink.h"\n#include "vr_genetlink.h"/'
~/src/Juniper/vrouter/linux/vr_genetlink.c
# sed -i "s/env.Append(CPPPATH = \['#tools\/sandesh\/library\/c'\])/env.Append(CPPPATH
= \['#tools\/sandesh\/library\/c'\])\nenv.Append(CPPPATH = \['\/usr\/include'\])/"
~/src/Juniper/vrouter/SConscript
# sed -i "s/env.Append(CPPPATH = \['\/usr\/include'\])/env.Append(CPPPATH = \
['\/usr\/include'\])\n\nenv.Append(CFLAGS = \['\/usr\/include'\])/"
~/src/Juniper/vrouter/SConscript
sed -i 's/make -C /make V=1 -C -I\/usr\/include /' ~/src/Juniper/vrouter/SConscript
sed -i "s/' \&\& make'/' \&\& make V=1 -I\/usr\/include '/"
~/src/Juniper/vrouter/SConscript
# sed -i "s/env = VRouterEnv.Clone()/env = VRouterEnv.Clone()\n\nenv.Append(CPPPATH = \
['\/usr\/include'\])/" ~/src/Juniper/vrouter/linux/SConscript
date >> /home/rein/JuniperVRouterBuild.run.time
```

Zet het aantal concurrent processen en start de bouw in verbose mode.

```
export NUM_CPU="$(nproc)"
scons -Q debug=1
date >> /home/rein/JuniperVRouterBuild.run.time
```

vvvvv

vvvvv

vvvvv

OpenStack extra's

Om te voorkomen dat een hypervisor 'vol' loopt of dat er extra tools, op de hypervisor, noodzakelijk zijn, kan er een mail forwarder, bv “ssmtp”, en log server worden ingezet, bv rsyslog.

Voor “ssmtp” zijn twee configuratie file in '/etc/ssmtp/' gezet. Deze worden, doormiddel van ansible worden geconfigureerd.

Default files:
File '`/etc/ssmtp/revaliases`'

```
# sSMTP aliases
#
# Format:   local_account:outgoing_address:mailhub
#
# Example: root:your_login@your.domain:mailhub.your.domain[:port]
# where [:port] is an optional port number that defaults to 25.
```

File '`/etc/ssmtp/ssmtp.conf`'

```
#
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
# Make this empty to disable rewriting.
root=postmaster

# The place where the mail goes. The actual machine name is required no
# MX records are consulted. Commonly mailhosts are named mail.domain.com
mailhub=mail

# Where will the mail seem to come from?
#rewriteDomain=

# The full hostname
hostname=build

# Are users allowed to set their own From: address?
# YES - Allow the user to specify their own From: address
# NO - Use the system generated From: address
#FromLineOverride=YES
```

Voor log files wordt, onder Ubuntu, standaard “rsyslog” geïnstalleerd. Deze sysloger is eenvoudig te configureren voor gebruik van een syslog server. Deze opzet maak de stap naar een ELK (Elasticsearch, logstash en Kibana) stack eenvoudiger.

Voor een opzet van rsyslog naar ELK zie oa:
https://www.digitalocean.com/community/tutorials/how-to-centralize-logs-with-rsyslog-logstash-and-elasticsearch-on-ubuntu-14-04

vvvv