Introduction vagrant, docker, docker-compose

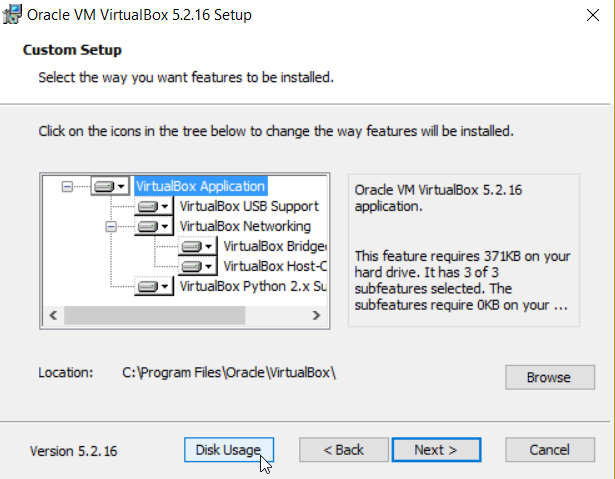
# Preparations

## VirtualBox

Install VirtualBox 5.2.16: <https://www.virtualbox.org/wiki/Downloads>.

* For Windows: <https://download.virtualbox.org/virtualbox/5.2.16/VirtualBox-5.2.16-123759-Win.exe>
* For OSX: <https://download.virtualbox.org/virtualbox/5.2.16/VirtualBox-5.2.16-123759-OSX.dmg>
* For Linux the installation differs per distribution: <https://www.virtualbox.org/wiki/Linux_Downloads>

Make sure you select both the VirtualBox Networking options: Bridged and Host-Only. They should be selected by default.



After the VirtualBox installation, install the Extension Pack.

<https://download.virtualbox.org/virtualbox/5.2.16/Oracle_VM_VirtualBox_Extension_Pack-5.2.16.vbox-extpack>

## Vagrant

Install Vagrant: <https://www.vagrantup.com/> 2.1.2.

# Getting started with Vagrant

## Getting started

Create a new empty directory to work in

Inside this directory create a file named Vagrantfile with the following content:

VAGRANTFILE\_API\_VERSION = "2"

Vagrant.configure(VAGRANTFILE\_API\_VERSION) do |config|

config.vm.define "course" , primary: true do |course|

course.vm.box = "ubuntu/bionic64"

course.vm.hostname = "ubuntusts"

course.vm.provider :virtualbox do |vb|

vb.customize ["modifyvm" , :id, "--memory", "8192"]

vb.customize ["modifyvm" , :id, "--name" , "UbuntuSTS"]

vb.customize ["modifyvm" , :id, "--cpus" , 4]

vb.customize ["modifyvm" , :id, "--clipboard","bidirectional"]

vb.customize ["modifyvm" , :id, "--vram","16"]

end

course.vm.provision :shell, path: "provision.sh"

end

end

Create a file provision.sh with the following contents:

apt-get update

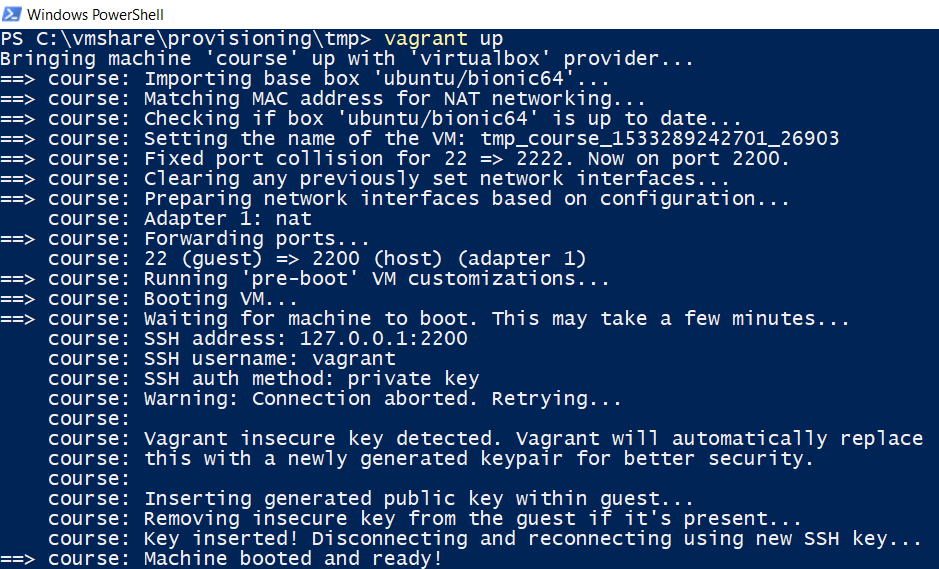
apt-get -y install aptitude apt-transport-https ca-certificates curl software-properties-common

This will install some basic packages which are often required.

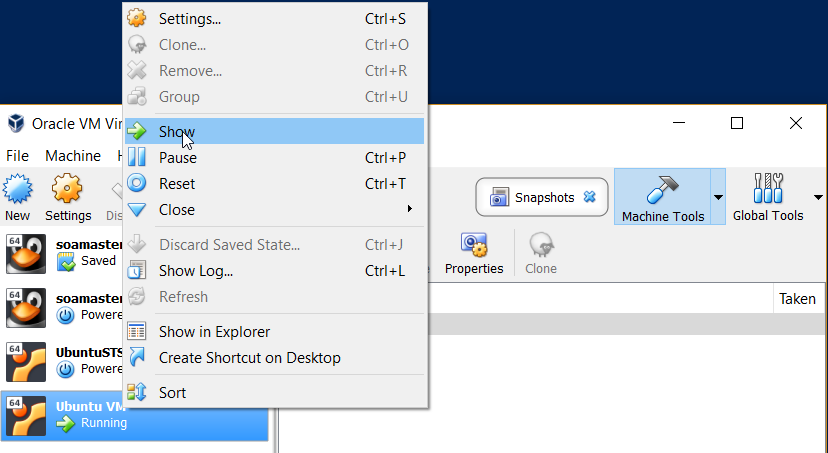
Start a commandshell in the directory with Vagrantfile and provision.sh

Create the VirtualBox VM

vagrant up

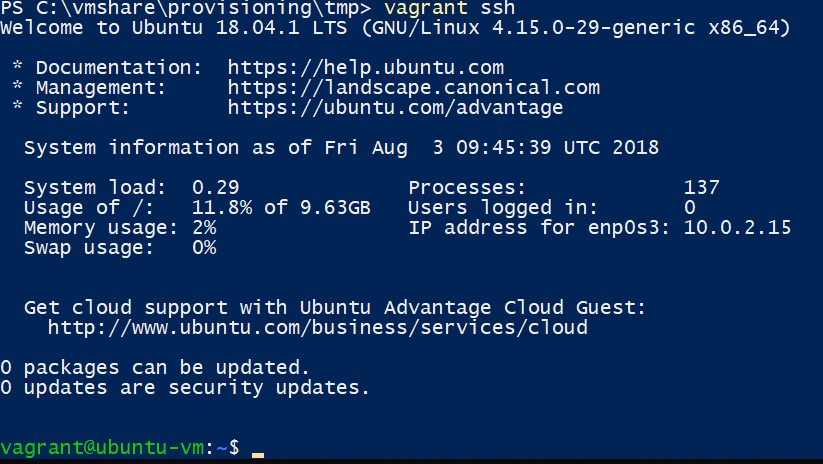


Confirm the VM is running



Login

vagrant ssh

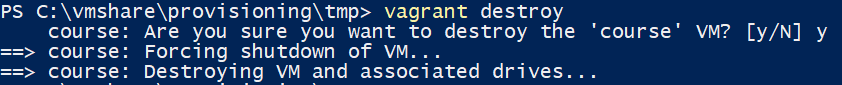


Exit the shell

exit

Remove the VM you’ve just created.

vagrant destroy



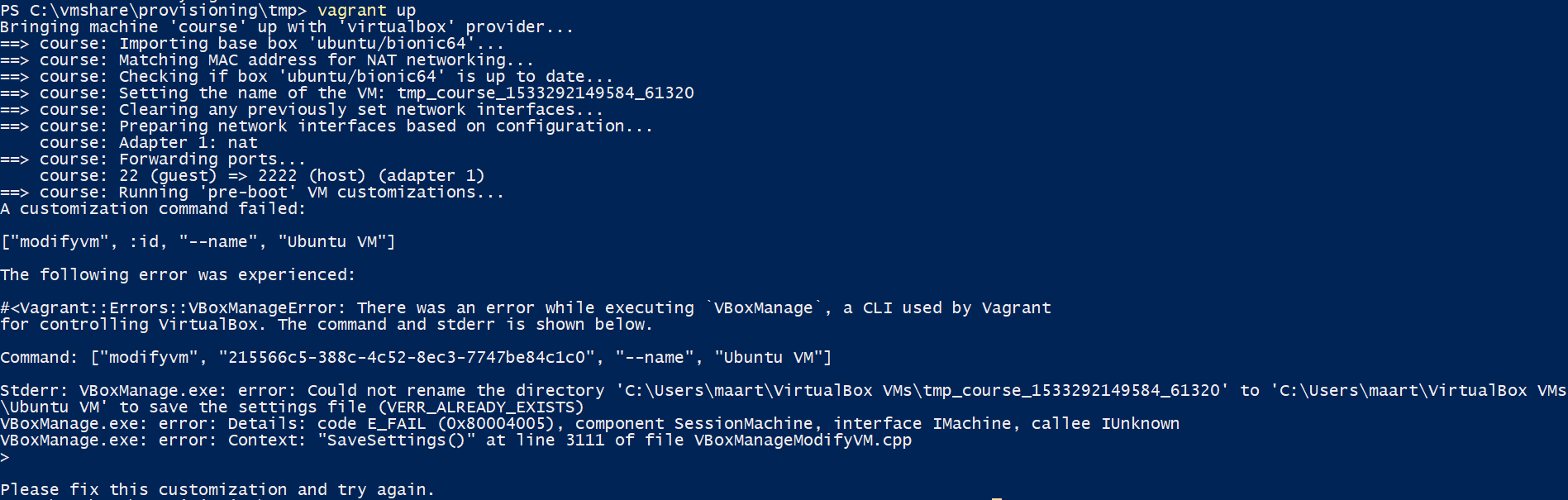
## Assignments

* What is ‘ubuntu/bionic64’ and where did it come from?
* How do you specify a specific version of the base box used in the Vagrantfile? Which versions are available?
* The default disk size is of the VM is approximately 14Gb. Increase the disk size to 50Gb by using configuration in the Vagrantfile and a Vagrant plugin.
* The installed VirtualBox guest additions might not be up to date. Install a Vagrant plugin and add configuration to the Vagrantfile in order to always have guest additions with the same version as your VirtualBox version

You will need to confirm you have completed the assignments by creating a VM using a specific version of the base box with 50Gb Disk and up to date guest additions.

## Troubleshooting

If the following happens:



Remove from the directory where your VM’s are stored (in my case: C:\Users\maart\VirtualBox VMs) the directory called ‘Ubuntu VM’ and try again. If you cannot remove it because it says it is in use, make sure no VirtualBox processes are running and try to delete it again.

# Provisioning Java, Docker, GUI

In order to create a useful development VM, it requires some software to be installed. Luckily this is easily automated.

## Java

From: https://technology.amis.nl/2018/07/27/automate-the-installation-of-oracle-jdk-8-and-10-on-rhel-and-debian-derivatives/

Edit the provision.sh script and add the following:

add-apt-repository ppa:linuxuprising/java

apt-get update

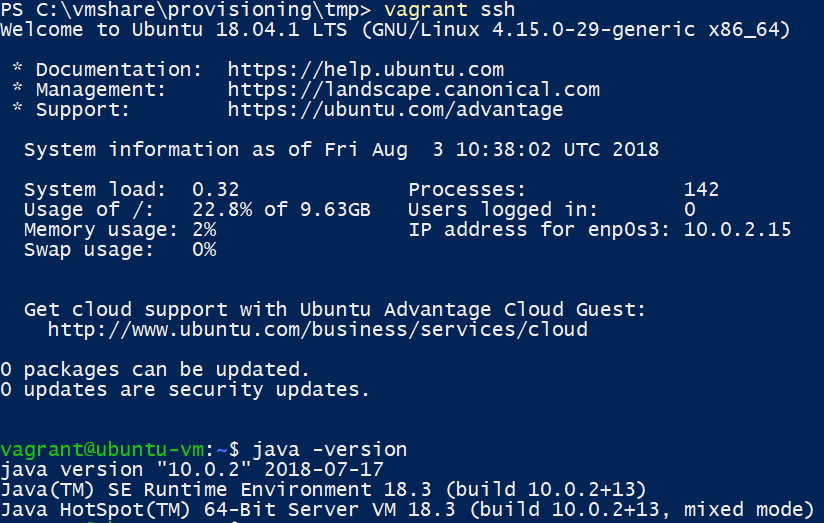
echo debconf shared/accepted-oracle-license-v1-1 select true | sudo debconf-set-selections

echo debconf shared/accepted-oracle-license-v1-1 seen true | sudo debconf-set-selections

apt-get -y install oracle-java10-installer

apt-get -y install oracle-java10-set-default

Confirm this installs Java by recreating the VM with vagrant up.



Why would you not use a similar mechanism in production to install and keep the JDK updated?

## Docker

Installing Docker can be done by adding the following to the provision.sh script and executing it or by running the below commands manually (add sudo in front of the commands in that case).

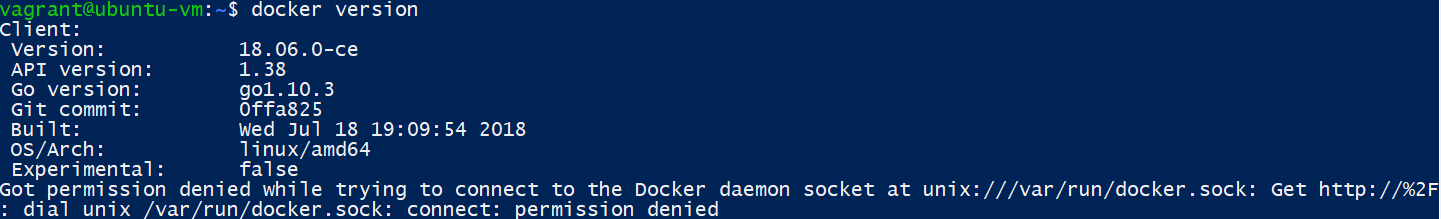
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

apt-get update

apt-get -y install docker-ce docker-compose

Confirm Docker is installed (from within the VM):



Why do you get a permission denied message? How do you fix this?

There is an easier way to install Docker and docker-compose with Vagrant inside a VirtualBox VM. What is it and give it a try. Hint: <https://technology.amis.nl/2018/05/21/rapidly-spinning-up-a-vm-with-ubuntu-and-docker-on-my-windows-machine-using-vagrant-and-virtualbox/>

## GUI

In you want to install a development environment (IDE) inside VirtualBox, having a GUI is key.

Add the following to provision.sh

aptitude -y install --without-recommends ubuntu-desktop

apt-get install firefox terminator

Recreate the VM and confirm you now have a GUI Ubuntu

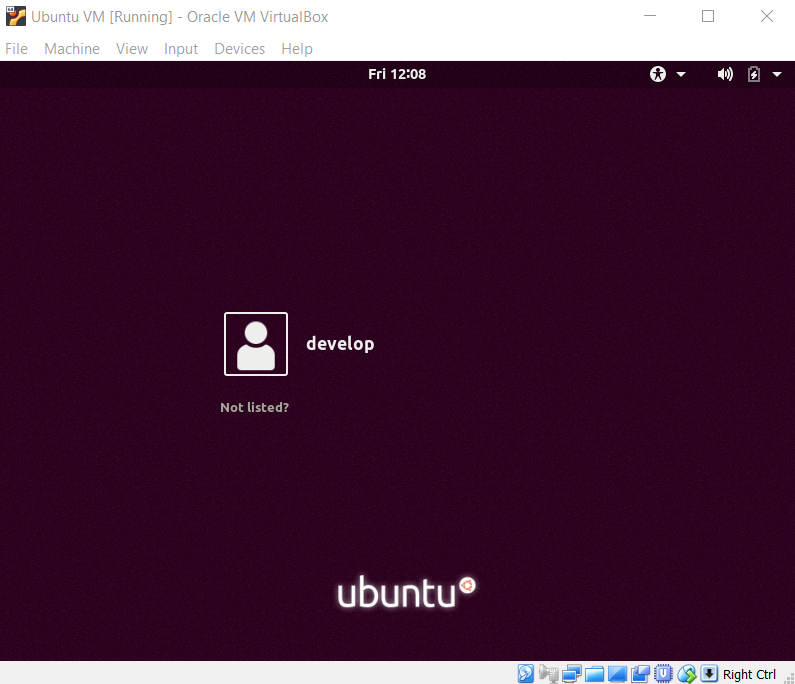
You cannot login to this GUI. Why is that? How would you fix this?

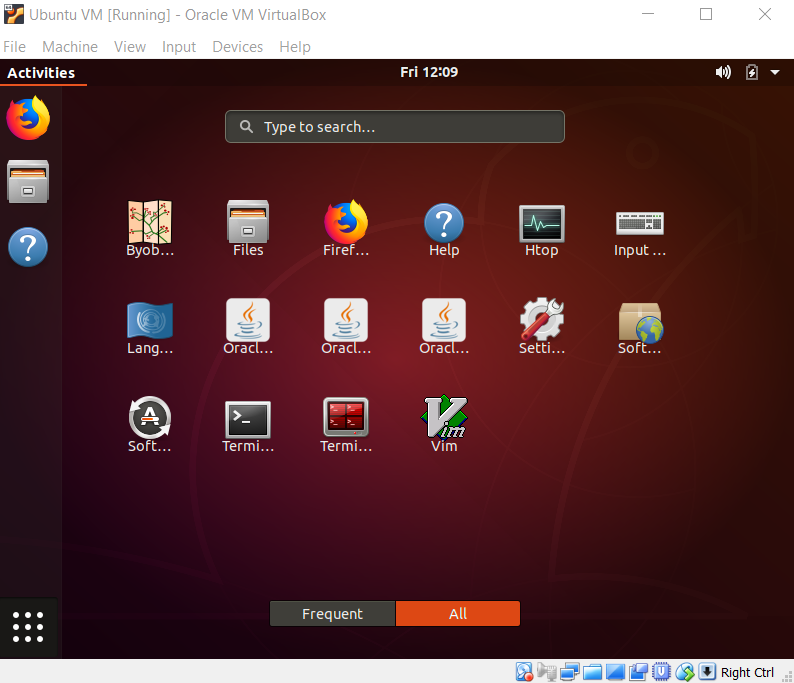
If you experience screen flickering when running the GUI, add the following to provision.sh

#Fix screen flickering issue

sudo perl -e '$^I=".backup";while(<>){s/#(WaylandEnable=false)/$1/;print;}' /etc/gdm3/custom.conf

Perform the actions described in the solutions section before continuing. The result should be a VM where you can login with user develop password Welcome01 which has Firefox, Terminator, docker, JDK10 installed.





# Intermezzo

If you have done the assignments so far, your Vagrantfile and provision.sh should look something like below. You can use these files to continue with the next assignments or download them from <https://github.com/MaartenSmeets/provisioning/tree/master/vagrant/workshop23aug2018AMIS>.

## Vagrantfile

VAGRANTFILE\_API\_VERSION = "2"

Vagrant.configure(VAGRANTFILE\_API\_VERSION) do |config|

config.vm.define "course" , primary: true do |course|

course.vm.box = "ubuntu/bionic64"

course.vm.box\_version = "20180508.0.0"

course.disksize.size = "50GB"

course.vbguest.auto\_update = true

course.vm.hostname = "ubuntu-vm"

course.vm.provider :virtualbox do |vb|

vb.customize ["modifyvm" , :id, "--memory", "8192"]

vb.customize ["modifyvm" , :id, "--name" , "Ubuntu VM"]

vb.customize ["modifyvm" , :id, "--cpus" , 4]

vb.customize ["modifyvm" , :id, "--clipboard","bidirectional"]

vb.customize ["modifyvm" , :id, "--vram","16"]

end

config.vm.provision :docker

config.vm.provision :docker\_compose

course.vm.provision :shell, path: "provision.sh"

end

end

## provision.sh

apt-get update

apt-get -y install aptitude apt-transport-https ca-certificates curl software-properties-common

add-apt-repository ppa:linuxuprising/java

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

apt-get update

echo debconf shared/accepted-oracle-license-v1-1 select true | sudo debconf-set-selections

echo debconf shared/accepted-oracle-license-v1-1 seen true | sudo debconf-set-selections

apt-get -y install oracle-java10-installer

apt-get -y install oracle-java10-set-default

apt-get -y install docker-ce docker-compose

aptitude -y install --without-recommends ubuntu-desktop

apt-get -y install firefox terminator

#Add user develop with password Welcome01

useradd -d /home/develop -m develop

echo -e "Welcome01\nWelcome01" | passwd develop

usermod -a -G vboxsf develop

usermod -a -G docker develop

usermod -a -G sudo develop

usermod --shell /bin/bash develop

#Hide user vagrant

echo '[User]' > /var/lib/AccountsService/users/vagrant

echo 'SystemAccount=true' >> /var/lib/AccountsService/users/vagrant

#Fix screen flickering issue

sudo perl -e '$^I=".backup";while(<>){s/#(WaylandEnable=false)/$1/;print;}' /etc/gdm3/custom.conf

# Spring Tool Suite

We also want to install an IDE to develop in. For Spring Boot Spring Tool Suite (STS) is a common choice. In order to keep Spring Tool Suite isolated, we are going to run it from a docker container. The files shown below can also be downloaded from: <https://github.com/MaartenSmeets/provisioning/tree/master/docker/STS>. Read the following blog post to understand what you are doing and why: <https://technology.amis.nl/2018/08/06/running-spring-tool-suite-and-other-gui-applications-from-a-docker-container/>

Add to provision.sh the following (or execute as root in the existing VM):

xhost local:root

Next do the following:

export UID=$UID

Create the following Dockerfile:

FROM ubuntu:18.04

MAINTAINER Maarten Smeets <maarten.smeets@amis.nl>

ARG uid

LABEL nl.amis.smeetsm.ide.name="Spring Tool Suite" nl.amis.smeetsm.ide.version="3.9.5"

ADD https://download.springsource.com/release/STS/3.9.5.RELEASE/dist/e4.8/spring-tool-suite-3.9.5.RELEASE-e4.8.0-linux-gtk-x86\_64.tar.gz /tmp/ide.tar.gz

RUN adduser --uid ${uid} --disabled-password --gecos '' develop

RUN mkdir -p /opt/ide && \

tar zxvf /tmp/ide.tar.gz --strip-components=1 -C /opt/ide && \

ln -s /usr/lib/jvm/java-10-oracle /opt/ide/sts-3.9.5.RELEASE/jre && \

chown -R develop:develop /opt/ide && \

mkdir /home/develop/ws && \

chown develop:develop /home/develop/ws && \

rm /tmp/ide.tar.gz && \

apt-get update && \

apt-get install -y libxslt1.1 libswt-gtk-3-jni libswt-gtk-3-java && \

apt-get autoremove -y && \

apt-get clean && \

rm -rf /var/lib/apt/lists/\* && \

rm -rf /tmp/\*

USER develop:develop

WORKDIR /home/develop

ENTRYPOINT /opt/ide/sts-3.9.5.RELEASE/STS -data /home/develop/ws

Create the following docker-compose.yml file:

version: '3'

services:

sts:

build:

context: .

dockerfile: Dockerfile

args:

uid: ${UID}

container\_name: "sts"

volumes:

- /tmp/.X11-unix:/tmp/.X11-unix

- /home/develop/ws:/home/develop/ws

- /home/develop/.m2:/home/develop/.m2

- /usr/lib/jvm/java-10-oracle:/usr/lib/jvm/java-10-oracle

- /etc/java-10-oracle:/etc/java-10-oracle

environment:

- DISPLAY

user: develop

ports:

- "8080:8080"

Build the composition

docker-compose build

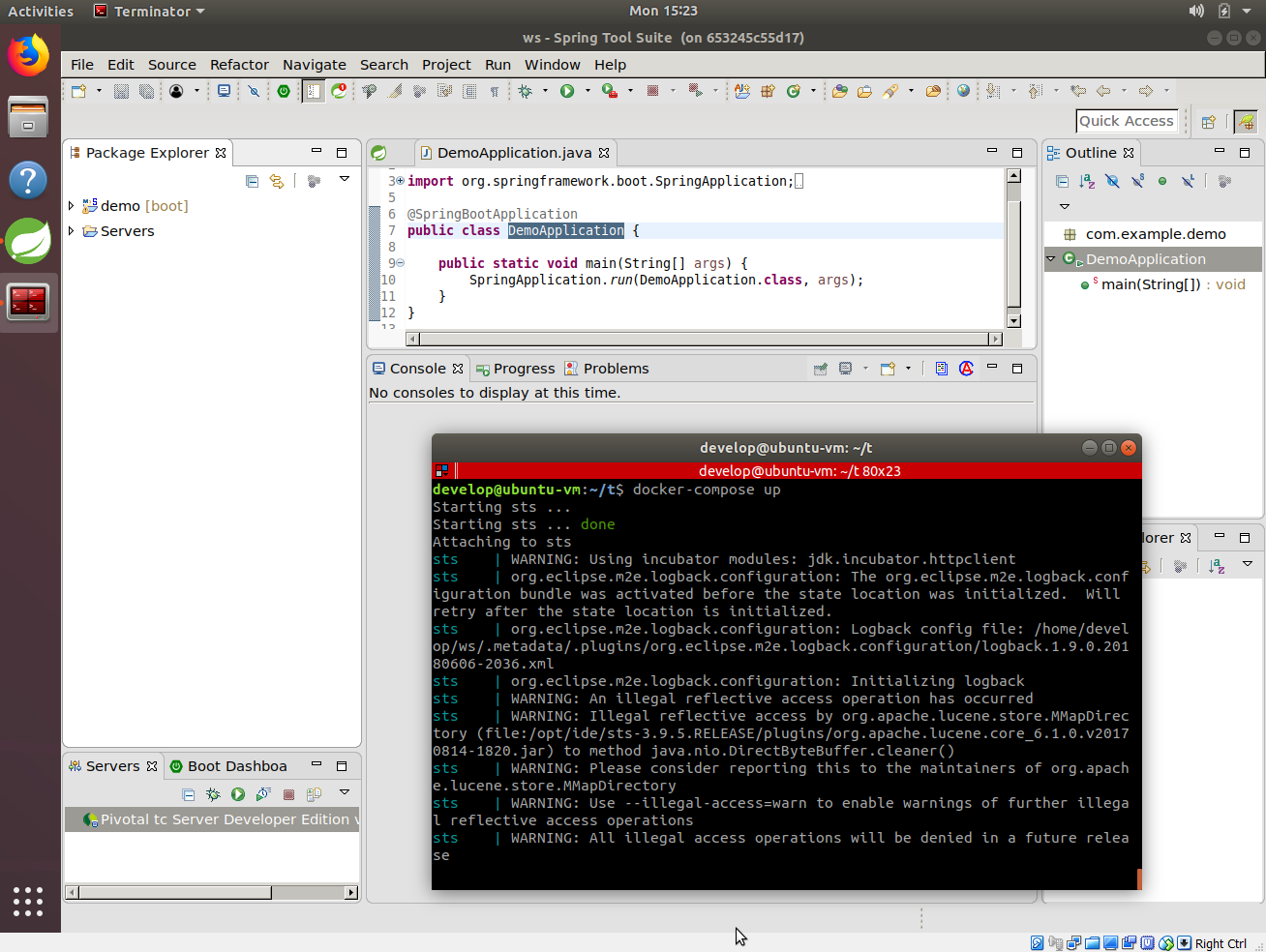
Create the following directories to hold the Maven repository and workspace:

mkdir /home/develop/.m2

mkdir /home/develop/ws

Start Spring Tool Suite

docker-compose up



## Assignment: Bringing it all together (OPTIONAL!)

I want to use create a VM based on Ubuntu 16.04 (Xenial Xerus).

Note 1: the screen flickering issue does not occur in Ubuntu 16.04.

Note 2: to make Unity (the menu) work, the following additional packages are required:

apt-get install unity-lens-applications unity-lens-files

I want to have Spring Tool Suite 3.9.4 (download link can be found on https://spring.io/tools/sts/legacy) installed running on Java 8 under the (host) user myuser. Automate the creation of the VM with Vagrant and Docker.

# kafka and docker networking

We are going to ‘install’ and run Kafka and connect to it from Spring Tool Suite. First go to the home directory of your Linux user:

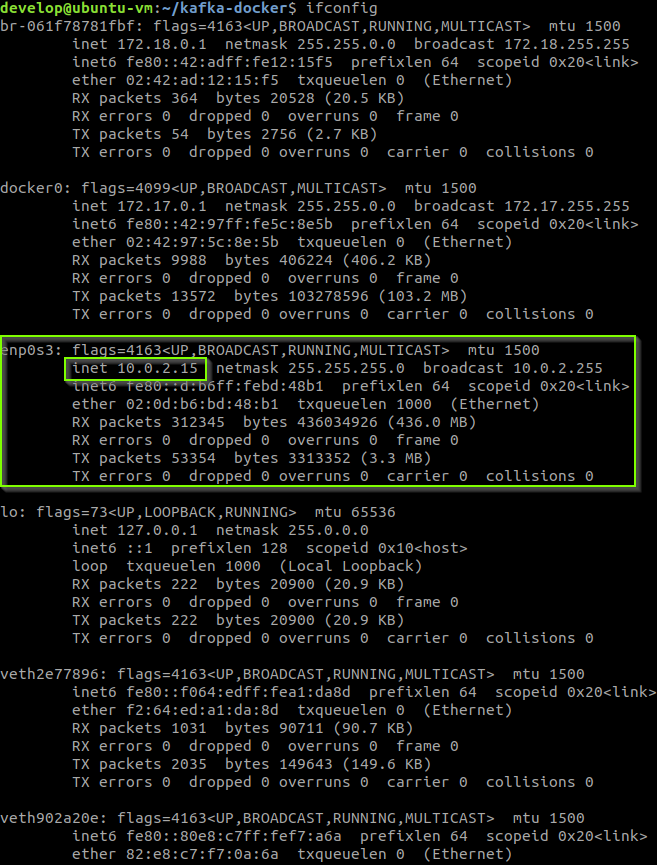
cd ~

git clone https://github.com/wurstmeister/kafka-docker.git

Update docker-compose.yml and replace the value of KAFKA\_ADVERTISED\_HOST\_NAME with your host IP. The host IP can be determined with:

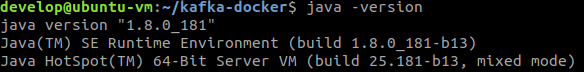
ifconfig | grep -Eo 'inet (addr:)?([0-9]\*\.){3}[0-9]\*' | grep -Eo '([0-9]\*\.){3}[0-9]\*' | grep -v '127.0.0.1'

If you see multiple IPs, determine the right one by looking at the ifconfig output yourself.



docker-compose-up

Install Java 8 as described on: <https://technology.amis.nl/2018/07/27/automate-the-installation-of-oracle-jdk-8-and-10-on-rhel-and-debian-derivatives/>



Download kafkatool

wget <http://www.kafkatool.com/download2/kafkatool.sh>

Install it:

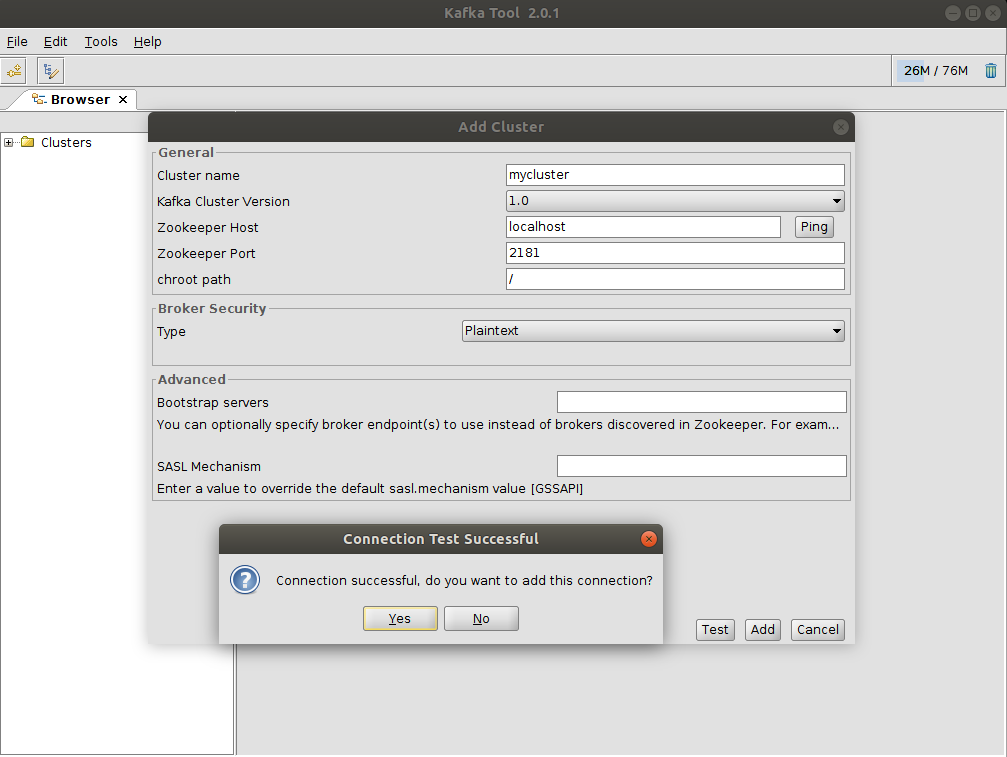
sh ./kafkatool.sh

Next, Accept, Next, Next, Next, Finish

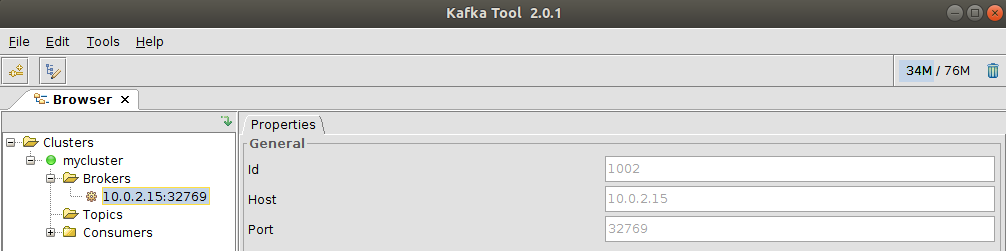
Start it:

/home/develop/kafkatool2/kafkatool

Configure it and confirm Kafka is running



Confirm Zookeeper specifies the correct hostname for the broker so Kafkatool can access it (KAFKA\_ADVERTISED\_HOST\_NAME has been set correctly)



What kind of docker network is used? How can Kafkatool connect to the Kafka broker?

docker network ls

docker network inspect kafkadocker\_default

Update the Spring Tool Suite docker-compose.yml so it connects to the same network and start it. Hint: <https://docs.docker.com/compose/networking/>

If you have rebuild your VM (lost these files), create a directory and fetch them:

cd ~

mkdir sts

cd sts

wget <https://raw.githubusercontent.com/MaartenSmeets/provisioning/master/docker/STS/Dockerfile>

wget <https://raw.githubusercontent.com/MaartenSmeets/provisioning/master/docker/STS/docker-compose.yml>

To run the docker-compose.yml file after having updated it:

export UID=$UID

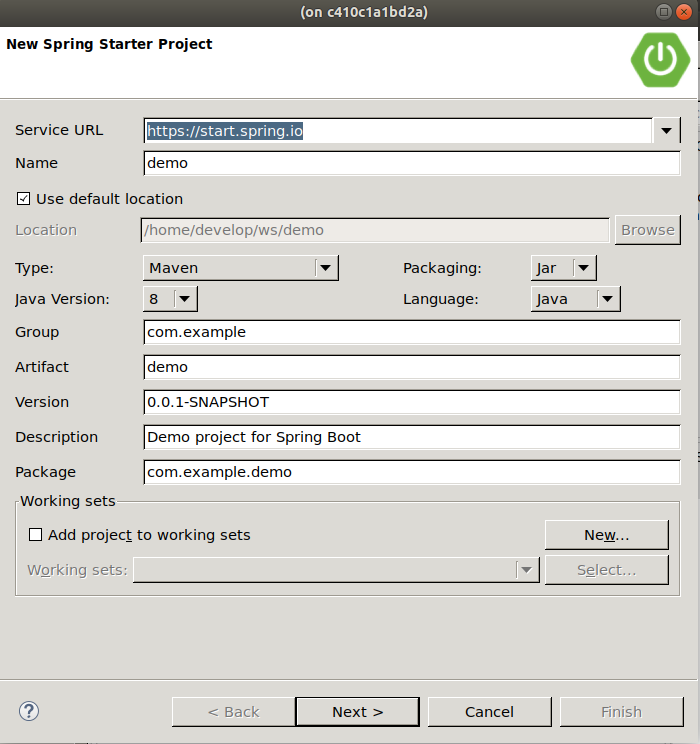
xhost local:root

mkdir ~/ws

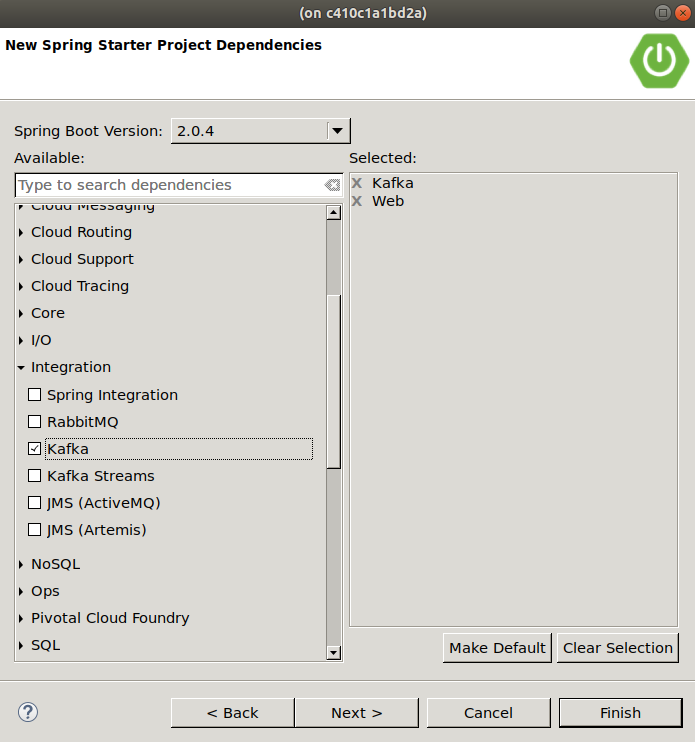
mkdir ~/.m2

docker-compose up

Create a new Spring Starter Project (from within STS, New, Spring Starter Project)



Add web and Kafka dependencies



Use the following code for the DemoApplication.java:

package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ConfigurableApplicationContext;

import org.springframework.kafka.core.KafkaTemplate;

@SpringBootApplication

public class DemoApplication {

@Autowired

private KafkaTemplate<String, String> kafkaTemplate;

public void send(String topic, String payload) {

kafkaTemplate.send(topic, payload);

System.out.println("Message: "+payload+" sent to topic: "+topic);

}

public static void main(String[] args) {

SpringApplication myApp = new SpringApplication(DemoApplication.class);

ConfigurableApplicationContext sa = myApp.run();

DemoApplication bean = sa.getBean(DemoApplication.class);

bean.send("test", "Spring Boot application has started");

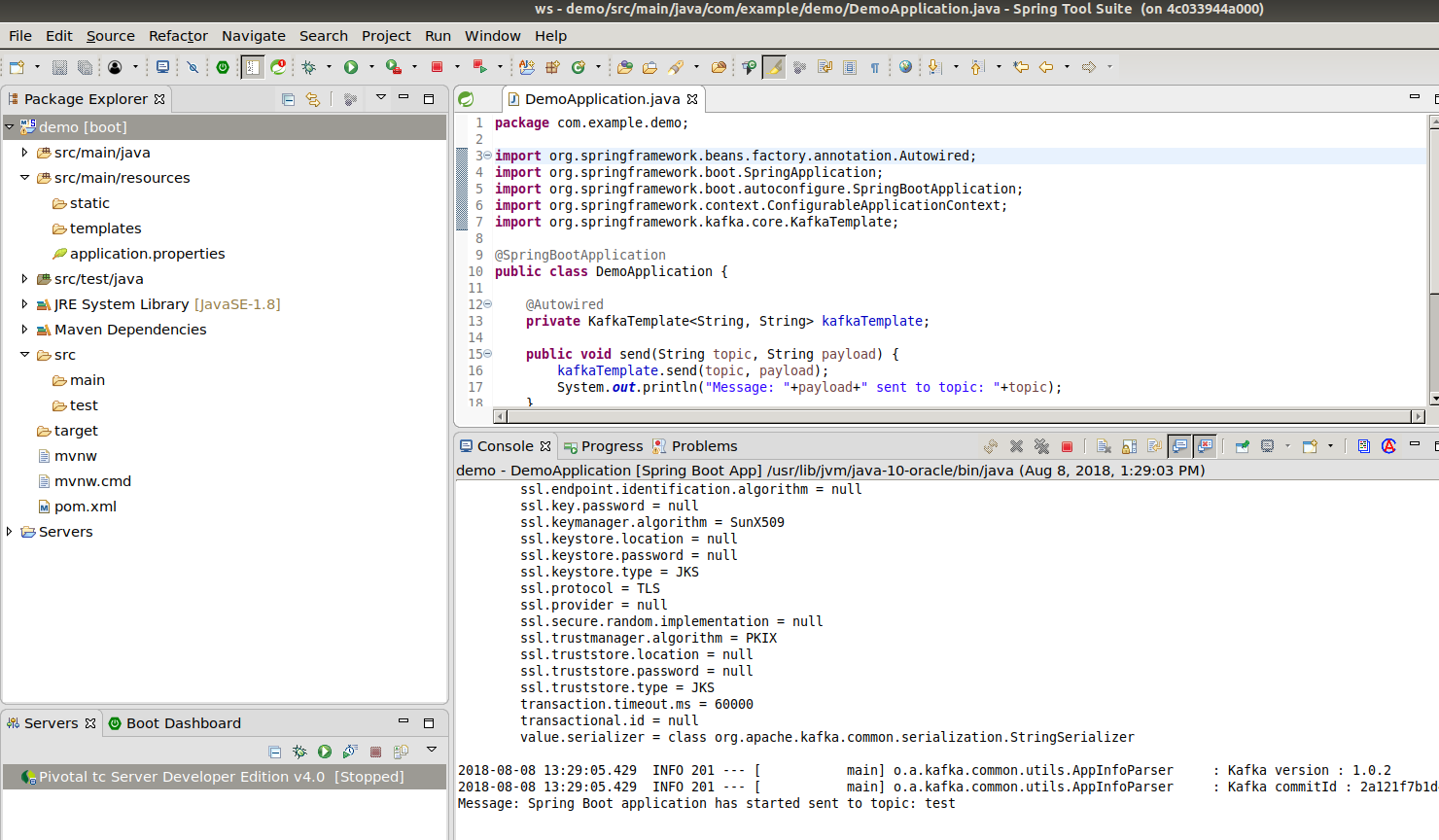
}

}

Edit src/main/resources/application.properties and add the following line:

spring.kafka.bootstrap-servers = kafka:9092

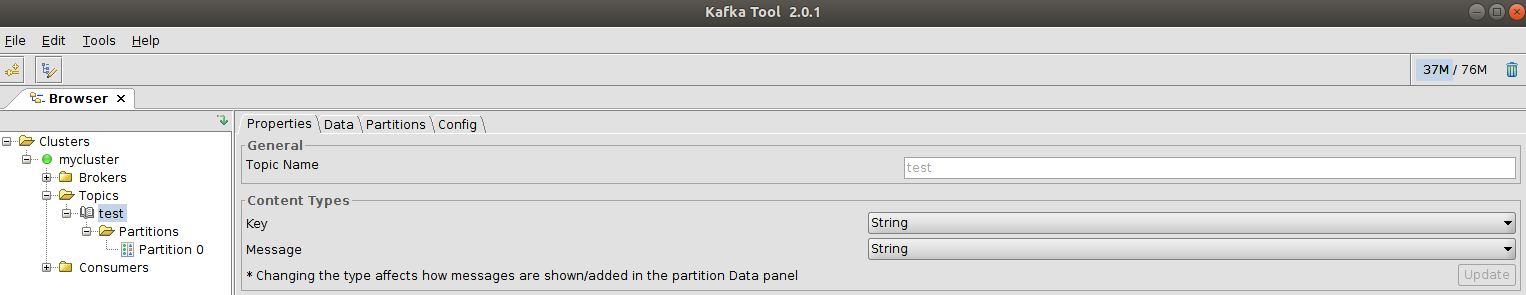
Start the application: Run, Run As, Spring Boot App



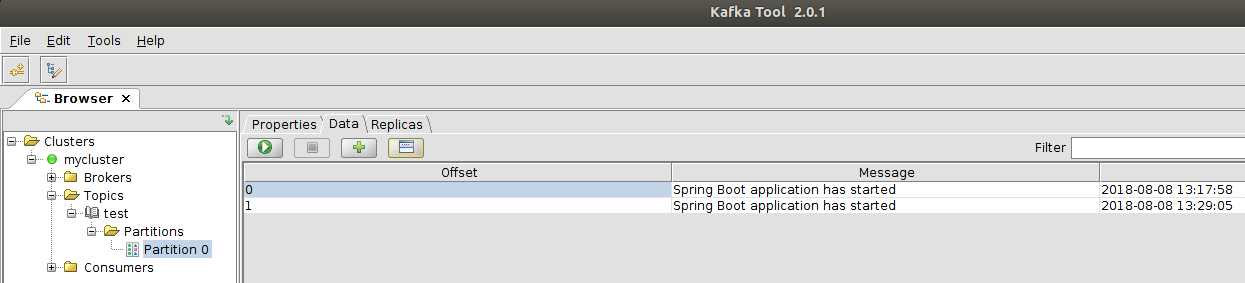
How does Spring Tool Suite know where ‘kafka’ is?

Confirm with Kafkatool the message has been posted to the test topic

Set the properties of the key and message to String

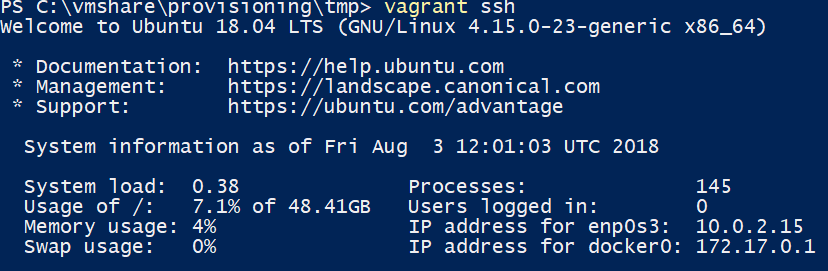


Open Partition 0, click data, click the green play icon to query

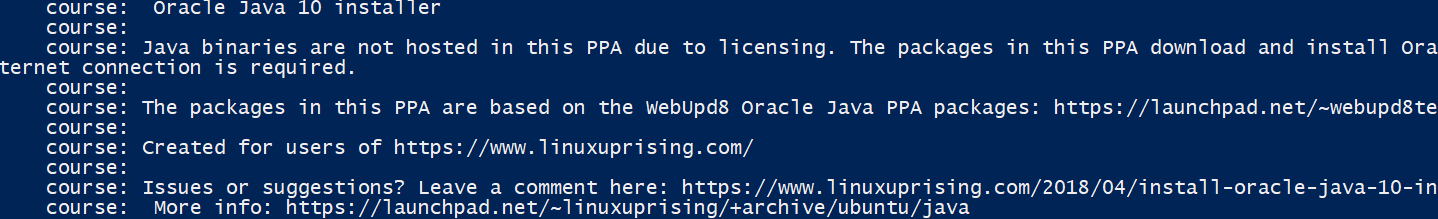


# Solutions

## Getting started with Vagrant

* Canonical (Ubuntu supplier) provides Vagrant base boxes for you to use. <http://cloud-images.ubuntu.com/>. These images are available through the Vagrant Cloud: <https://app.vagrantup.com/ubuntu/boxes/bionic64>
* Add a line like: course.vm.box\_version = "20180508.0.0"  
  The available versions can be determined from <https://app.vagrantup.com/ubuntu/boxes/bionic64>
* First install the plugin: vagrant plugin install vagrant-disksize  
  Next add to the Vagrantfile: course.disksize.size = “50GB”  
  After creating a VM you can confirm the disk size has been increased (Usage of /).  
  
* First install the plugin: vagrant plugin install vagrant-vbguest  
  Next add to the Vagrantfile: course.vbguest.auto\_update = true

## Provisioning Java, Docker, Gui

* Java: Why would you not use a similar mechanism in production to install and keep the JDK updated?   
    
  The installation is provided not by Oracle but by a 3rd party without support/guarantees. Updating the JDK automatically in production might break things which could compromise availability and stability. You might want to test a new version before automatically installing it.  
    
  Production environments often run on a RedHat derivatives. RedHat and Oracle have their own trusted repositories to keep the JDK up to date.
* Docker: Why do you get a permission denied message? How do you fix this?  
  A user is not allowed access to the Docker daemon by default. The Docker daemon runs as root and a user who can access the daemon can abuse this. You can add users to the docker group (add ‘usermod -a -G docker course’ to provision.sh). Read the following: <https://docs.docker.com/install/linux/linux-postinstall/> and understand the risk involved in adding a user to the Docker group. In production you would take specific measures to make using Docker more secure.
* There is an easier way to install Docker and docker-compose with Vagrant. What is it and give it a try.  
  vagrant plugin install vagrant-docker-compose and add the following 2 lines to the Vagrantfile:  
  config.vm.provision :docker  
  config.vm.provision :docker\_compose  
  Above course.vm.provision :shell, path: "provision.sh"
* GUI: You cannot login to this GUI. Why is that? How would you fix this?  
  The vagrant user is configured to use a private key to enter the VM. The user does not have a password set. Usually you would create a separate user in a VM to distribute. For example add the following to provision.sh

#Add user develop with password Welcome01

useradd -d /home/develop -m develop

echo -e "Welcome01\nWelcome01" | passwd develop

usermod -a -G vboxsf develop

usermod -a -G docker develop

usermod -a -G sudo develop

usermod --shell /bin/bash develop

#Hide user vagrant

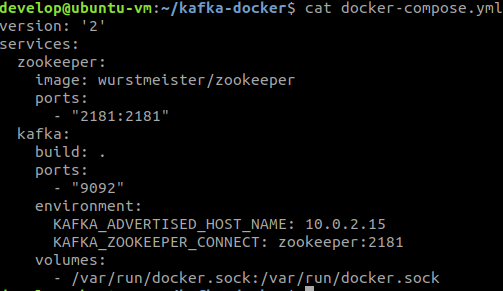
echo '[User]' > /var/lib/AccountsService/users/vagrant

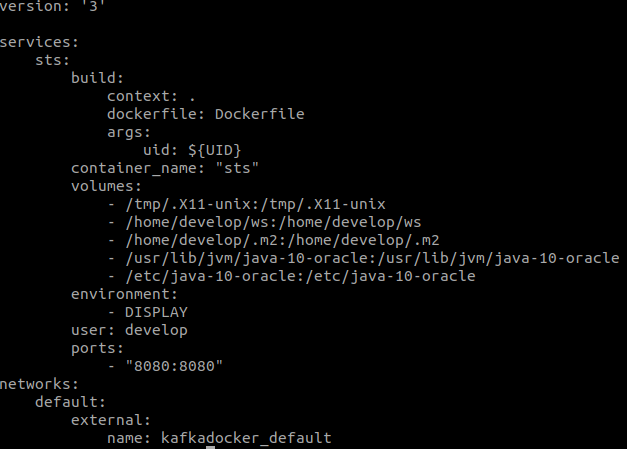
echo 'SystemAccount=true' >> /var/lib/AccountsService/users/vagrant

## Bringing it all together

See <https://github.com/MaartenSmeets/provisioning/tree/master/vagrant/workshop23aug2018AMIS/solutionlastassignment> for the solution files.

## Kafka and Docker networking

* What kind of docker network is used? How can Kafkatool connect to the Kafka broker?  
  A bridge network named kafkadocker\_default is used  
    
  Kafkatool can connect to the ports which are exposed on the host as specified in the docker-compose.yml:  
  
* Specifying to use a pre-existing network can be done like:



* How does Spring Tool Suite know where ‘kafka’ is?  
  In the kafka-docker folder in docker-compose.yml kafka is specified as service. Because STS is connected to the same bridge network, the services which are defined within the same network become available to each other (see automatic DNS resolution between containers for the named bridge network: https://docs.docker.com/network/bridge/#differences-between-user-defined-bridges-and-the-default-bridge)