1 eGov Testing Machine Manual

1.1 About the project

The Autonomous Province of Bolzano currently offers important digital services (for eg, online applications to start businesses, pay property taxes and initiate e-Payments etc), the number is expected to increase as eGoverment services continue to grow.

The FSCRS project intends to contribute to the creation and improvement of eGovernment services through an innovative process that verifies and tests functionality. The underlying idea of the project is that the accessibility of digital public services is necessary to increase the innovation potential of the region. .

The project takes its name from the Free Software Client Reference System, a specific reference system (OS + defined set of software applications) that during testing simulates a user accessing the services.

1.2 About the software

Testingmachine is used to test eGovernment services. The eGovernment services can vary from country to country but generally it grants citizens access to important documents and information. In most EU countries there are additional services like paying property tax that can be done online. Most governments in Europe are leaning in this direction in order to decrease administrative overhead.

Another challenge to overcome will be testing eGov services on mobile platforms, more specifially automating tests under Android. Good news is this is possible using Selenium. The one major obstacle that we face is getting around the smart card login. So far I am unaware of getting this to work under Android unless you patch the kernel and this is of course no an option. More documentation and research is needed concerning this.

1.3 The eGov Testing Machine

The expected result is the development of a systematic, auto validation process, currently not available on the market, which allows the testing of eGovernment services without the manual intervention of an operator

The eGov Testing Machine can be thought of as a virtual group of people, sitting at the computer and using the eGov services and checking if they work properly, allowing the local Public Administration to test eGov services on a daily basis that are being offered to all citizens.

1.4 Software Overview

The Testing Machine is currently made up by Virtual Machine Manager (tm-vmm) and documentation on how to wite, execute and automate tests of eGov sites in particular but also other softwares.

1.4.1 Virtual Machine Manager (VMM)

tm-vmm is made up by bash scripts that let the user manage various virtual machine software in a general way. See the tm-vmm manual for more information.

1.4.2 eGov Manuals

Writing tests of eGov sites is not hard but we believe that some help will still be useful for most people. The manuals and guides provided together with tm-vmm make it possible to test eGov sites automatically and unattended.

2 Features

2.1 Virtual Machine Management

With Testing Machine you can:

- start/pause/stop virtual machines
- take screenshots of running virtual machines
- execute commands in running virtual machines # Installation

2.2 Software requirement

- ssh (client)
- at least one virtualization software (see list of supported softwares below)
- bash

2.2.1 Additional requirements for VMM developers

- pandoc - to generate documentation
- pdflatex - to generate documentation

2.3 Supported Virtualization software

- Virtualbox
- Android
- qemu

We're looking into supporting: vmware,

2.4 Downloading Virtual Machine Manager

2.4.1 Via git

- Download git code

```
git clone git://github.com/tis-innovation-park/vmm.git
```

2.4.2 Via a dist file (.tar.gz)

Coming later

2.5 Building and installing Virtual Machine Manager

- Go to the vvm directory

```
cd vvm
```

- Configure the software

```
./configure --prefix <installationdir>
```

- Build the software

```
make
```

- Install the software

```
sudo make install
```

- Verify the installation

```
<installationdir>/bin/tm-vmm --list-clients
```

3 Setup

- Create the directory $HOME/.testingmachine

```
mkdir $HOME/.testingmachine
```

- Create tm-vmm.conf in .testingmachine, typically with your favorite editor (emacs?)

```
emacs ~/.testingmachine/tm-vmm.conf
```

In this file you can configure settings you want to use as default in your clients. It is perfectly possible to override these settings in your individual client configurations.

For a list of variables, see section Configuration syntax below.

3.1 Creating a machine

To create a machine vvm relies on the virtualization software. So if you want to manage a Virtualbox machine you (at least for now) create it with Virtualbox. For information about how to do this, read the chapter “Creating a Virtualbox machine” or “Creating a Android Virtual Device”.

3.2 Preparing for creating a client

First of all you need to decide what machine you want to use with your client.

- Create a directory for all clients:

```
mkdir ~/.testingmachine/clients
```

3.3 Create a VirtualBox client

In the example below we will assume it is called Debian-6.0

3.3.1 Create a client configuration file

You can create client configuration in two different ways:

- Manually
- Automatically

Create a configuration file manually

- Manually create a configuration file for the client (bound to a virtual machine):

  [CLIENT NAME].conf

  - Set the variables as you find suitable for your project.

Create a configuration file automatically

- Use the command line option:

  `--create-client-conf`

Example usage of the option.

```
`tm-vmm --create-client-conf  Debian-6.0`
```

It is assumed that Debian-6.0 is a VirtualBox image.

3.3.2 Example of a client configuration:

```
VM_NAME=Debian6.0
VM_TYPE="VirtualBox"
VM_IP_ADDRESS=192.168.1.2
VM_USER=$USER
VM_SUPERUSER=root
SSH_PORT=22
SSH_SHUTDOWN_COMMAND="shutdown -h now"
```

For more variables see section Configuration syntax below

- Copy your public ssh key to the machine's root account, e.g

```
ssh-copy-id -p 2256
```

3.4 Create an Android client

In the example below we will assume it is called Nexus-10

3.4.1 Create a client configuration file

You can create client configuration in two different ways:

- Manually
- Automatically

Create a configuration file manually

- Manually create a configuration file for the client (bound to a virtual machine):

  [CLIENT NAME].conf

  - Set the variables as you find suitable for your project.

Create a configuration file automatically

- Use the command line option:

  `--create-client-conf`

Example usage of the option.

```
`tm-vmm --create-client-conf  Nexus-10`
```

It is assumed that Debian-6.0 and Nexus-10 is the name either a VirtualBox or Android image.

3.4.2 Example of a client configuration:

```
VM_NAME="eGov-android-machine"
VM_TYPE="Android"
ANDROID_SYS=emulator64-arm
```

For more variables see section Configuration syntax below

4 Using Virtual Machine Manager

Regardless of the underlying virtual machine you manage the machine in one way. We provide the most basic uses of tm-vmm here and encourage you to read the complete list of command line options in the last section of the manual.

4.1 Starting

4.1.1 Starting with guest operating system visible

```
tm-vmm --start-client <CLIENTNAME>
```

Example:

```
tm-vmm --start-client Ubuntu-10
```

4.1.2 Starting without showing guest operating system (headless)

```
tm-vmm --start-client-headless <CLIENTNAME>
```

Example:

```
tm-vmm --start-client-headless Ubuntu-10
```

4.2 Checking status

```
tm-vmm --check-client-status <CLIENTNAME>
```

Example:

```
tm-vmm --check-client-status Ubuntu-10
```

4.3 Stopping

```
tm-vmm --stop-client <CLIENTNAME>
```

Example:

```
tm-vmm --stop-client Ubuntu-10
```

5 Selenium

5.1 About Selenium

Selenium is a portable software testing framework for web applications. Selenium provides a record/playback tool for authoring tests without learning a test scripting language (Selenium IDE). It also provides a test domain-specific language (Selenese) [1] to write tests in a number of popular programming languages, including Java, C#, Groovy, Perl, PHP, Python and Ruby. The tests can then be run against most modern web browsers. Selenium deploys on Windows, Linux, and Macintosh platforms.

This text is quoted from: http://en.wikipedia.org/wiki/Selenium_(software)

Selenium is run on the guest so some tweaks are needed to get things working, but don't worry we've done almost everything for you.

5.2 Writing Selenium tests

6 Miscallaneous other softwares

6.1 Sikuli

Installing Java is a prerequisite for running Sikuli. Install it with apt-get:

```
sudo apt-get install openjdk-6-jdk
```

The software can be installed on Ubuntu 12.04 LTS with the following command:

```
sudo apt-get install sikuli
```

6.2 GNU Xnee

6.3 wmctrl

The package "wmctrl" is used to resize windows.

```
sudo apt-get install wmctrl
```

7 Launching tests using VMM

7.1 Using crontab

7.2 Using Jenkins

8 Testing eGov sites

8.1 Problems we've encountered

8.1.1 Logging in to a web site using a smart card

When setting up the Testingmachine in South Tyrol to test the *burger card* the biggest obstacle we had to overcome was the login process. Since this is done using a smart card and a pin we found no way to do this using Selenium and so Sikuli was used. Once logged in, Selenium can be used to test the web services.

Getting Sikuli to perform an automatic login can be done in the following way:

8.2 Sikuli

In order to use Sikuli a test case has to be made. This is done within Sikuli to emulate the login process step by step. A possible test case could look like this:

```
wait("KJ2.png",60)
click("1352889207953.png")
sleep(5)
wait("Pleaseentert.png",60)
click("Pleaseentert-1.png")
type("96414648")
sleep(1)
click("OK.png")
sleep(2)
wait("Thlssltehasr.png",60)
click("1352890307684.png")
sleep(2)
wait("Tl2fIlHlKJlI.png",60)
```

The .png files are created with Sikuli. When running Sikuli you can take 'snapshots' of certain parts of the screen that are captured and later used as a reference point where to perform certain functions like emulating button presses or entering text. Sikuli then 'looks' for parts of the screen that matches the

'snapshots' before it performs actions. This works well with web pages since Sikuli can wait until a page has loaded before executing a command.

Note: *We've also written some words on how to setup smart cards. See *

9 Creating virtual machines

9.1 Creating a VirtualBox machine with Ubuntu 12.10

Basically you should follow the normal procedure, as described in the VirtualBox manual, on how to create a new virtual machine. However we provide a guide below to make this easier. You can choose to use our settings or change some at your will.

Note: *If you're on a Ubuntu system you might have to add your user to the vboxusers group.*

9.1.1 Start Virtualbox

In a terminal, type: `virtualbox`

In the VM Virtualbox Manager window click "New"

9.1.2 Name and operating system

- Name: eGov testing machine
- Type: Linux
- Version: Ubuntu (64 bit)

9.1.3 Memory size

- 2048 MB

9.1.4 Hard drive

- Choose to create a virtual hard drive now

9.1.5 Hard drive file type

- Choose VDI

Figure 1: Name and os

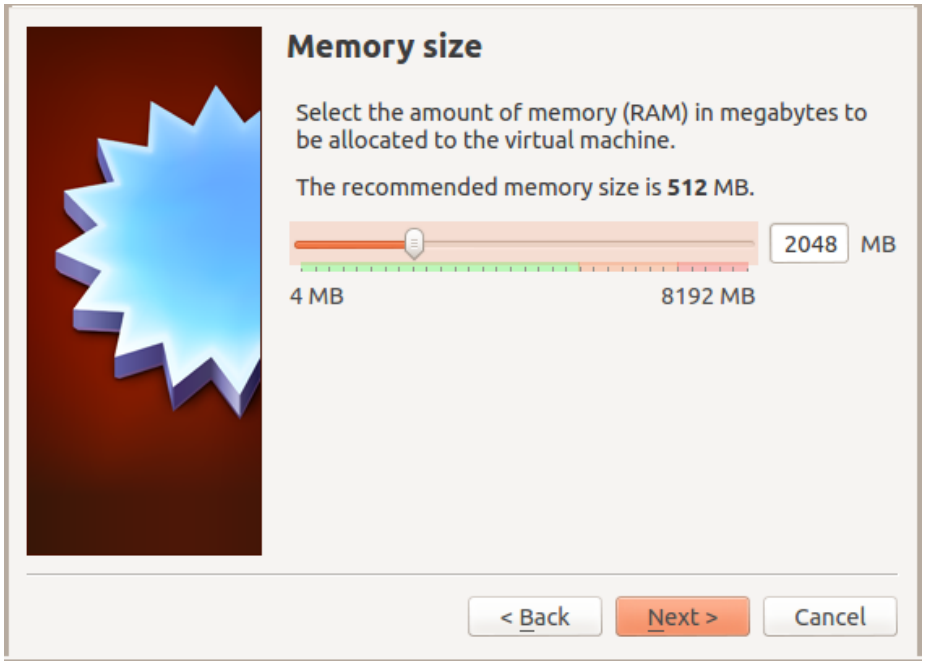

Figure 2: Memory size

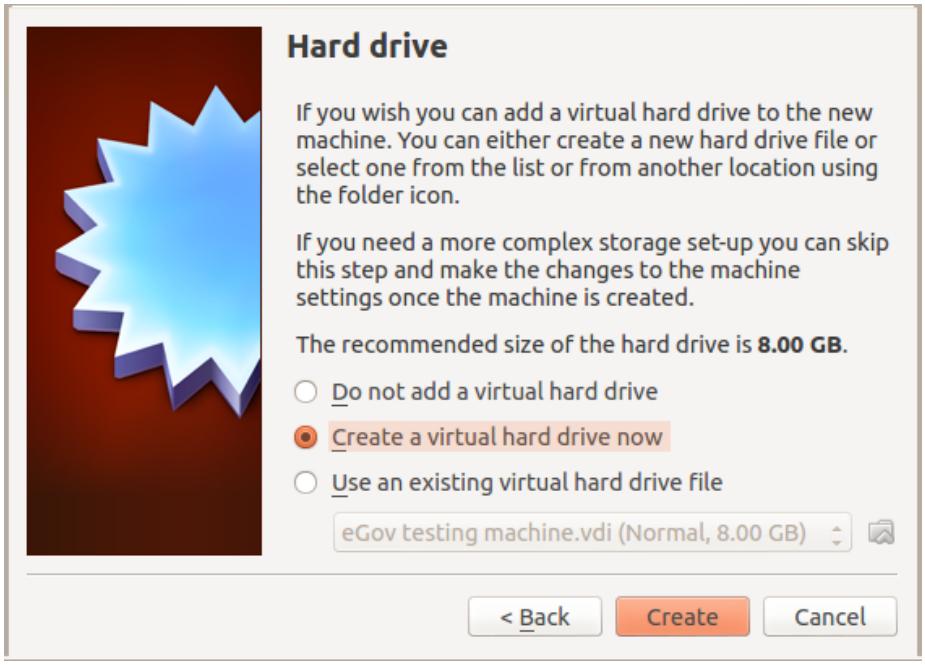

Figure 3: Hard drive

Figure 4: Hard drive file type

9.1.6 Storage on physical hard drive

- Choose Dynamically allocated

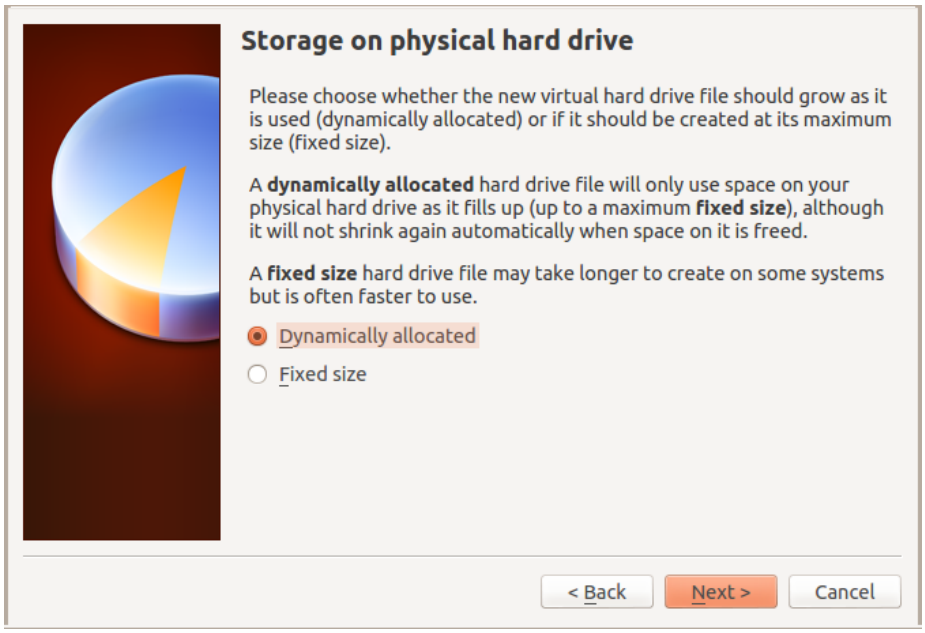

Figure 5: Hard drive file type

9.1.7 File location and size

- 8 GB should be enough

Your disk has now been created. Before starting it we need to do some additional settings.

9.1.8 Network

Setup the network card

- Open up the VirtualBox Manager
- Choose your Virtual machine and click Settings
- Make sure Network Adapter 1 is enabled.

The following settings should be applied to Adapter 1.

Attached to: NAT (previously Bridged adaptor was advised)

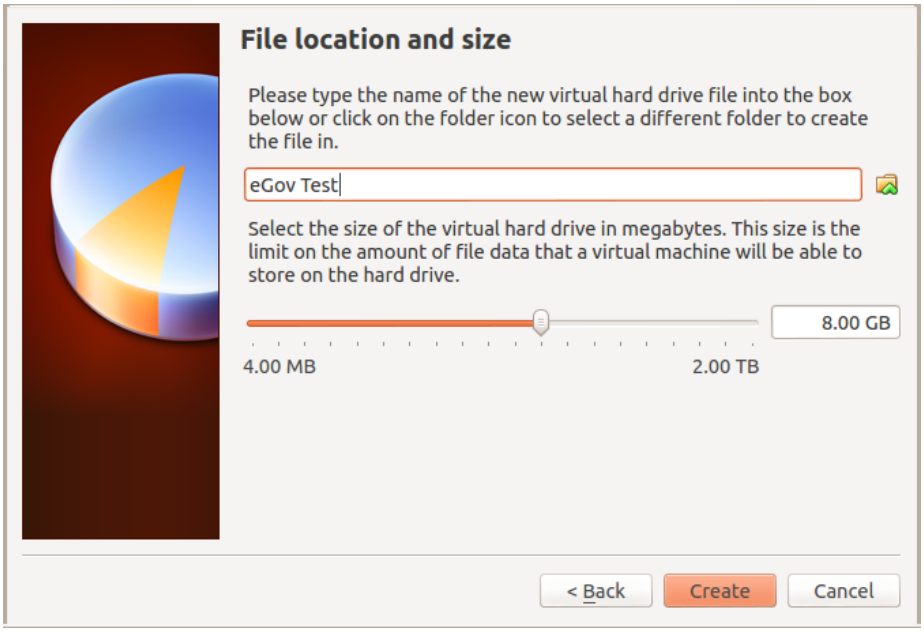

Figure 6: File location and size

- **Name**: eth0
- **Advanced**:

- **Adapter type**: Intel PRO/1000 MT Desktop (...)
- **Promiscuous mode**: Deny
- **MAC Address**: use the suggested
- **Cable connected**: should be checked

This gives your virtual machine an IP address on the same subnet as the host computer. See http://www.virtualbox.org/manual/ Chapter 6 for additional information on bridged networking.

Allowing SSH logins to your virtual machine Open up the VirtualBox Manager

Choose your Virtual machine and click Settings

Choose Port forwarding

Add a new rule by clicking the + sign. Enter

- **Rule**: ssh

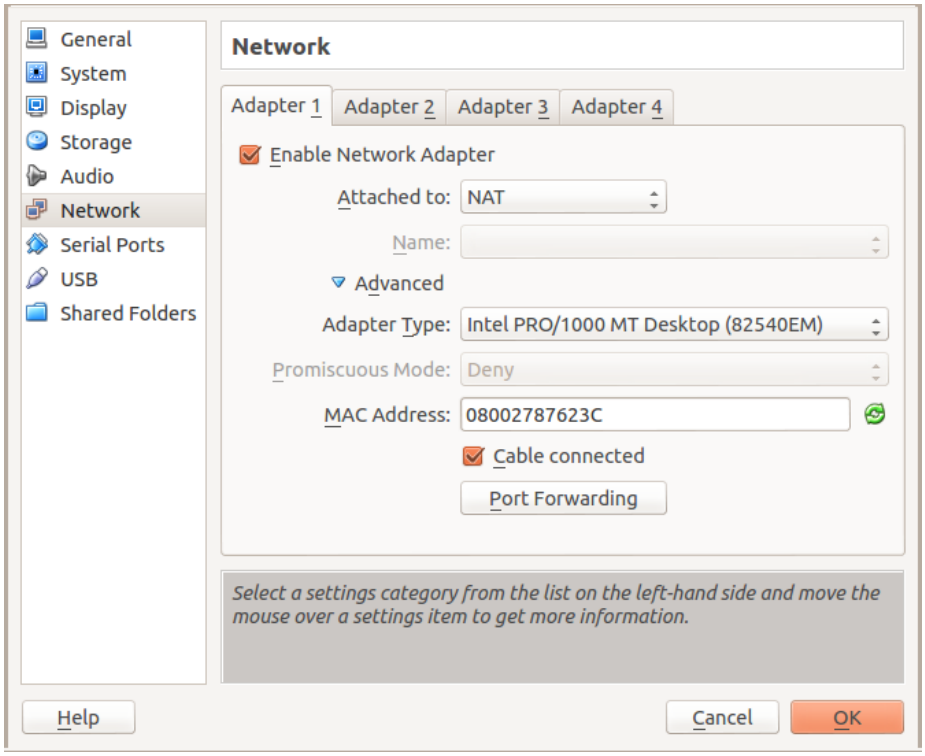

Figure 7: Network

- **Protocol**: TCP
- **Host IP**:
- **Host Port**: 2256 (the value can basically be any free port on your host computer)
- **Guest IP**:
- **Guest Port**: 22

Figure 8: Network

USB Make sure USB is enabled if you plan to use a smart card reader or another USB device

To use a smart card reader in your virtual machine you have to do the following:

- Attach the smart card reader to your computer
- Add Filter from Device (Alt + Ins) and choose the device that is attached to the physical machine.
- The same procedure applies to other peripherals, like a USB storage device.

See http://www.virtualbox.org/manual/ Chapter 3 for additional information

9.1.9 Installing Ubuntu

Download the preferred Ubuntu iso image from ubuntu.com

Click Settings and choose storage.

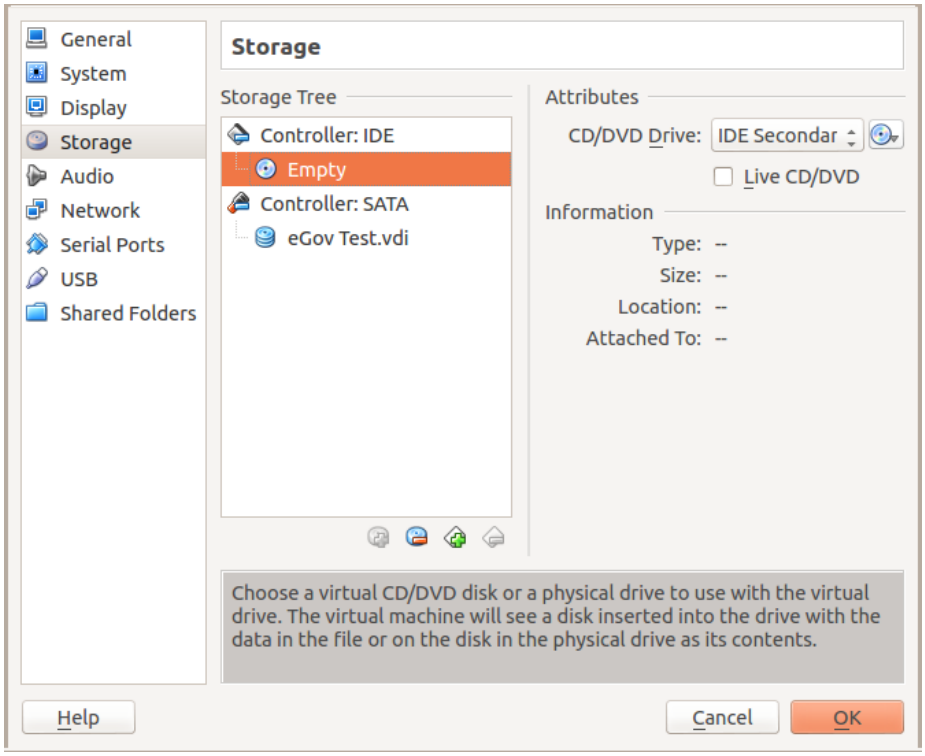

Figure 9: Storage

Add the downloaded iso image to the virtual machine as a CD-ROM by clicking the Empty icon (image missing) under Controller: IDE and then click on the disc icon right of the text CD/DVD Drive: .

Choose a virtual CD/DVD disk file ... and point out the downloaded iso image (e.g. ubuntu-12.10-desktop-amd64.iso).

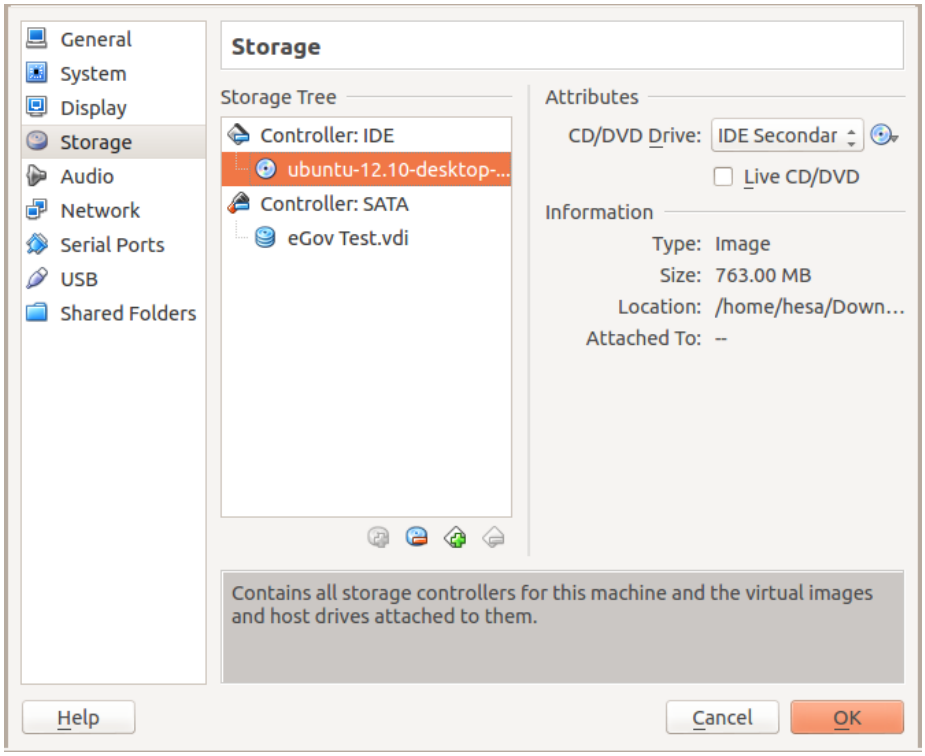

Figure 10: Storage

Click Start

Follow the installation instructions to install Ubuntu in your new virtual machine.

Upgrading your installation To upgrade your system you need to:

- Click on the Software updater icon to your left
- Click install now
- Enter password

Installing necessary tools in your virtual machine The following tools need to be installed on your virtual machine

- openssh-server

Install them by typing the following command in a terminal window, assuming you're logged in as your first user (created during the Ubuntu installation):

sudo apt-get install openssh-server

Setting up users on the virtual machine Log in to your virtual machine as the user you created during installation.

Setting up a new user VMM puts no restrictions or requirements on the name of the user in your virtual machine. The user name "vmm" is given here as an example and will be used in all manual text below.

Log in as the user you created during the Ubuntu installation.

Open up a user management tool by pressing the dasher (logo missing) and type user accounts.

- Press the unlock symbol and type in the password of the first user you created.
- Press the + symbol. You will now see a new window, called Create new account, in which you should fill in:

Account Type: Standard **Full name**: Virtual Machine Manager **Username** vmm

- Click on Enable password and type in a password.

Setting up the root account

- Permit root to login via the ssh server.
- Launch a terminal window
- Open up the ssh server configuration file

```
sudo gedit /etc/ssh/sshd_config
```

- Make sure that PermitRootLogin is set to yes
- Restart the ssh server

```
sudo /etc/init.d/ssh restart
```

- Add the ssh key of your host user to vmm account

```
ssh-copy-id "-p 2256 vmm@localhost"
```

- Test your user acoount

```
ssh -p 2256 vmm@localhost whoami
```

- Add the ssh key of your host user to the root account

```
ssh -p 2256 vmm@localhost -t "sudo mkdir /root/.ssh && sudo cp
/home/vmm/.ssh/authorized_keys /root/.ssh/"
```

- Test your root acoount

```
ssh -p 2256 root@localhost whoami
```

We assume here that you're using port 2256 for ssh and that your user is called vmm. Change it accordingly if not. **Note**: Make sure that your port (-p 2256) and host arguments (vmm@localhost) are enclosed between the same parenthesises.

9.2 Creating a Android Virtual Device

Basically you should follow the normal procedure, as described on the Android Developer pages manual: http://developer.android.com/tools/devices/index.html.

We do, however, provide a guide below to make this easier. You can choose to use our settings or change some at your will. In this guide we will setup a Nexus 7 device.

9.2.1 Start Android Virtual Device Manager

In a terminal, type:

```
android avd
```

In the Android Virtual Device Manager window click "New"

9.2.2 Create new Android Virtual Device (AVD)

- Name: eGov-android-machine
- Device: Nexus 7
- Target: Android 4.2

You can keep the default values for all other settings.

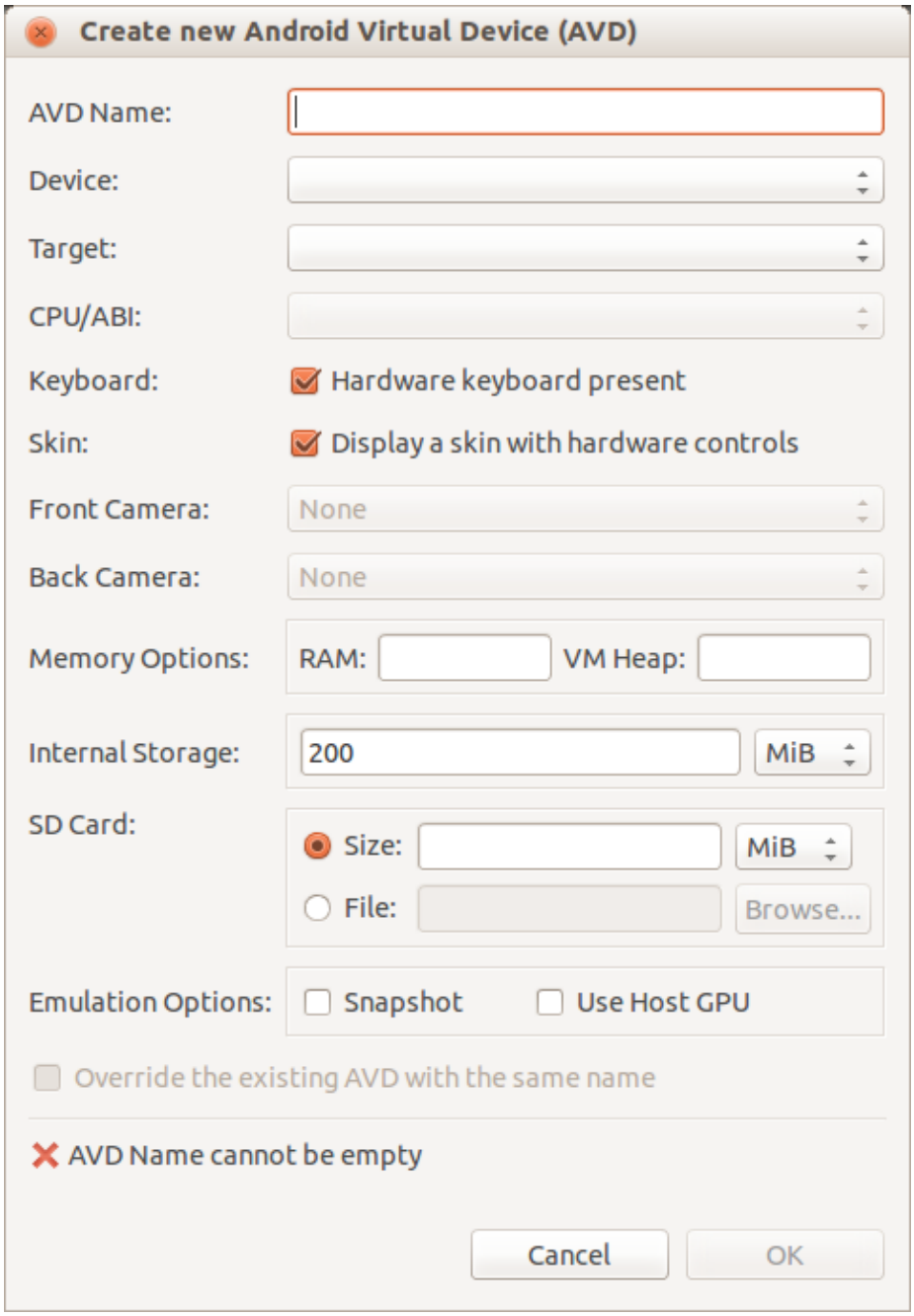

Figure 11: Create new Android Virtual Device

10 Example use:

/opt/bin/tm-vmm –check-client-status Ubuntu-12.10

/opt/bin/tm-vmm –check-client-status Ubuntu-12.10

/opt/bin/tm-vmm –check-client-ssh Ubuntu-12.10

/opt/bin/tm-vmm –client-exec Ubuntu-12.10 “pkcs15-tool -L”

11 Project Communication

11.1 Reporting bugs

Try to be as precise as possible when reporting bugs. The more information we get the bigger chance we have of fixing the problem.

Use the mailing list below to report bugs.

11.2 Getting involved

How to join: clone the repo, try it out – join the mailing list :)

For more information read the developer guidelines.

11.3 Mailing list

We have one mailing list for the project: `community@testingmachine.eu`

Join this list here: `https://lists.testingmachine.eu/cgi-bin/mailman/listinfo/community`

If you send emails to this list as a non subscriber chances are it will get list.

If you want to report a bug: * use a github account and add an issue * subscribe to the mailing and send the report to the list

11.4 Blog

https://testingmachine.eu/blog

11.5 Home page

https://testingmachine.eu/

Source code is located here: `https://github.com/tis-innovation-park/vmm`

11.6 Social media

11.6.1 Twitter

https://twitter.com/FSCRS

12 Testing Machine VMM configuration

12.1 Configuration file syntax

The syntax for setting a variable is the same as in bash scrips (no coincidence!). Basically you write:

`VARIABLE=VALUE`

12.1.1 Variables

`LOG_FILE_DIR=/tmp/vmm/log` - sets the log file base directory to /tmp/vmm/log. This means that all logs can be found here.

`VM_STARTUP_TIMEOUT=10` - the time to wait for a virtual machine to start up before considering it to be 'dead'.

`VM_STOP_TIMEOUT=20` - the time to wait for a virtual machine to start up before taking more drastic actions to take down the machine. Ultimately VMM will take down a machine with a `kill`.

`SSH=ssh` - the SSH program to use

`SSH_TEST_OPTIONS=" -o connectTimeout=3"` - the options to pass to the SSH program (see SSH variable above) when testing of the machine is up (or not).

`VM_NAME="Debian6.0"` - the name of the machine to use in this client. It is important that your Virtualization software can find this machine.

`VM_TYPE="VirtualBox"` - what type of Virtulization software this machine belongs to. Currently you can use: VirtualBox and qemu

`VM_IP_ADDRESS=192.168.x.x` - the IP address of the virtual machine

`VM_USER=$USER` - the user you want to use when accessing the virtual machine

`VM_SUPERUSER=root` - the name of the super user/administrator account, typically used to reboot the machine

`SSH_PORT=22` - the port where the SSH server is running on the client

`SSH_SHUTDOWN_COMMAND="shutdown -h now"` - VVM will do its very best to shut down a machine as gracefully as possible. One way to do this is to try to shut it down using SSH. The command in this variable will be used to do that.

13 tm-vmm Command line options

13.1 Client options

`--list-clients` - lists all configured clients

`--start-client CLIENT` - starts client name CLIENT

`--start-client-headless CLIENT_NAME` - Start client called CLIENT_NAME as headless (no screen)

`--stop-clients CLIENT` - stops client named CLIENT

`--list-running-clients` - Lists all clients currently running

`--check-client-ssh CLIENT` - Checks if ssh is up on CLIENT

`--check-client-status CLIENT` - Checks if clients is up and running

`--client-exec CLIENT cmd` - Execeute cmd on client

`--client-exec-as-root` - Execeute cmd on client as root

`--client-x11 CLIENT` - Checks if X11 is up and running on CLIENT

`--client-screenshot CLIENT` - Take a screenshot on CLIENT (not 100% ready)

`--print-client CLIENT` - Print the configuration for CLIENT

`--wait-for-ssh CLIENT` - Wait until ssh is up and running on CLIENT

`--open-ssh CLIENT` - Open an interactive shell (using ssh) on CLIENT

`--check-client-online CLIENT` - Check if CLIENT can ping the outside world

13.2 Machine options

`--list-machines` - lists all machines known to VVM

`--start-machine MACHINE` - starts machine named MACHINE

`--start-machine-headless VM_NAME` - Start machine called VM_NAME as headless (no screen)

`--stop-machine MACHINE` - stops machine named MACHINE

`--check-machine MACHINE` - checks status on machine named MACHINE