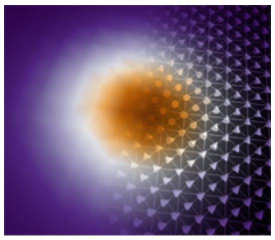
Virtual Micromagnetics Documentation

Release 1.1.0

Mark Vousden

May 01, 2017

Contents

 Welcome to the *Virtual Micromagnetics* project, where we aim to enable accessible and reproducible micromagnetics simulation, without compromise.

Provided by Mark Vousden, Hans Fangohr, and others at the University of Southampton. Funded by EPSRC's DTC grant EP/G03690X/1. The license for this software is available here.

The *Virtual Micromagnetics* project creates *virtual environment*s that run micromagnetic (and in some cases, atomistic) simulations of magnetic behaviour. These environments produce *system virtual machine*s which emulate a configured set of software on your computer. This means you as a user only need to manage the software to support the virtual machine, as opposed to the complicated set of dependencies most simulation packages require. As a result, these virtual environments are far simpler to maintain, meaning you have more time to solve the mysteries of the universe instead of:

- wondering why the latest version of a package is incompatible with earlier simulations.
- wondering how to maintain multiple versions of a package to support old simulation software.
- persuading your high-performance computing system administrator to support your long list of software dependencies.
- setting up user accounts and packages for new students to run simulations.

CHAPTER 1

Contents

Getting Started: As a User

Virtual Machines

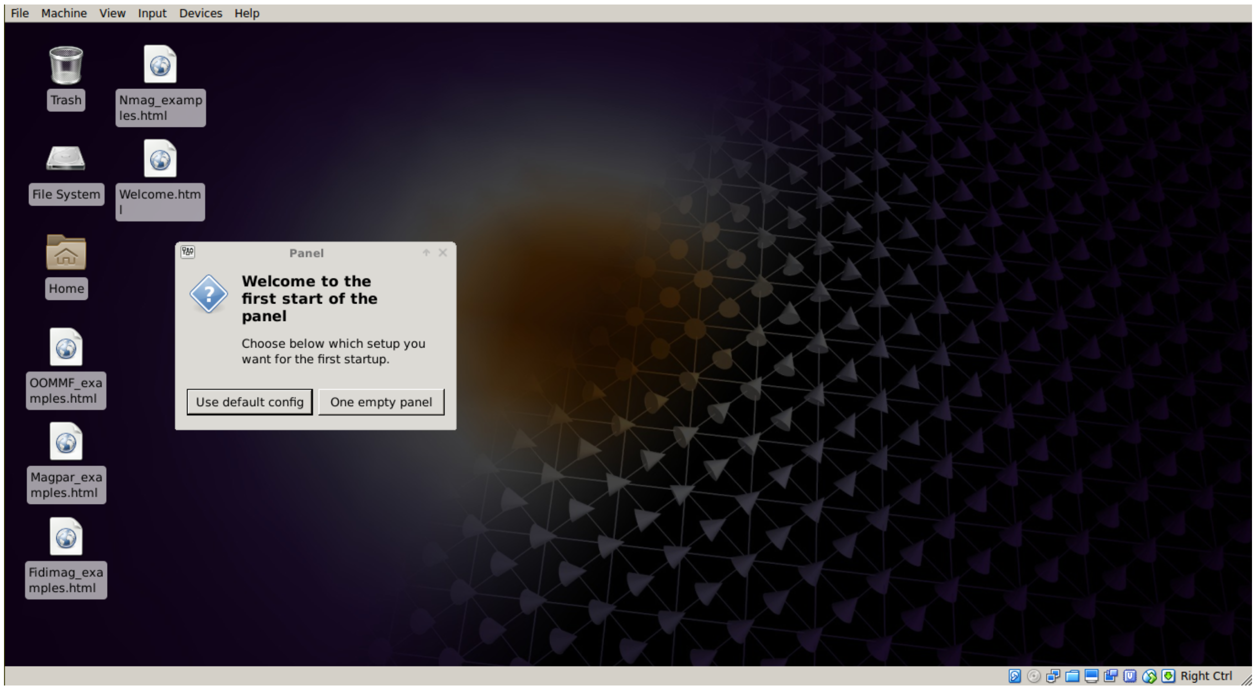
To start a *Virtual Micromagnetics* environment, you will need the following software:

- VirtualBox >= 5.0 (https://www.virtualbox.org/wiki/downloads)
- Vagrant >= 1.7.4 (https://www.vagrantup.com/downloads.html)

After restarting your machine, command the following in an empty directory:

```
vagrant init virtualmicromagnetics/full
vagrant up --provider virtualbox
```

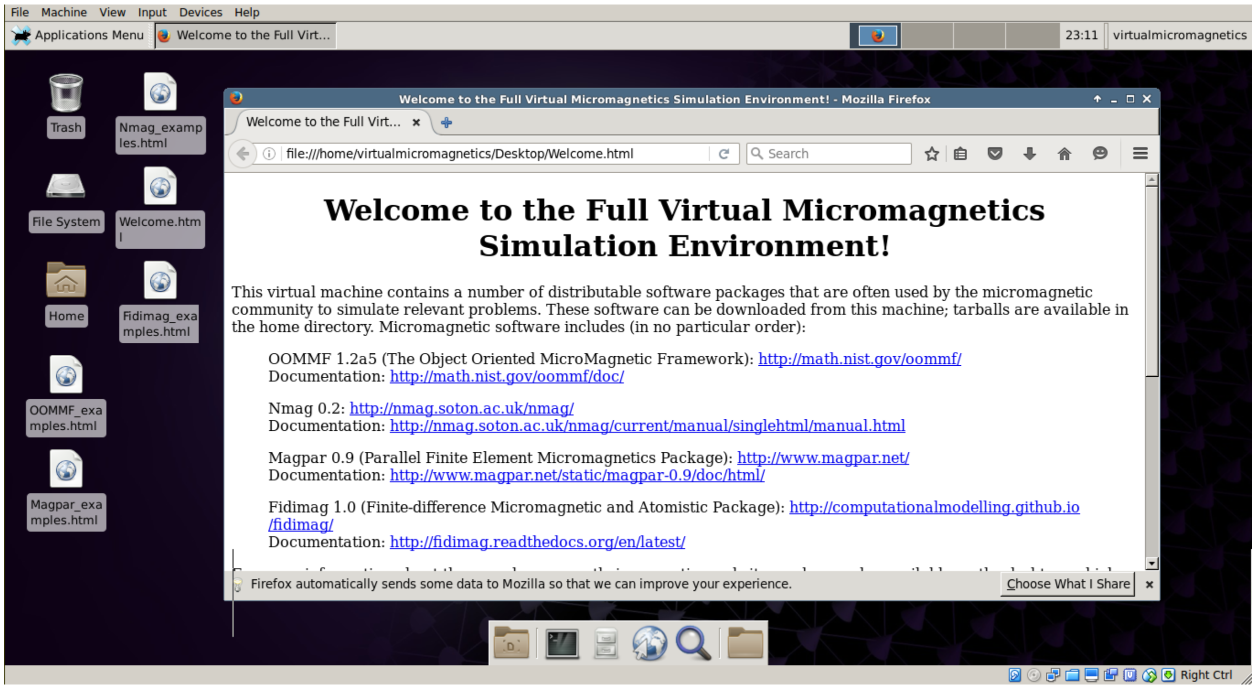
These commands will download the Full Virtual Micromagnetics environment from the Internet to your computer, and load the environment automatically. When complete, you should be greeted with this window:



Welcome to the first start of the panel

Choose below which setup you want for the first startup.

This is output from a *virtual machine* running on your computer! Virtual machines produced in this way run Ubuntu GNU/Linux (https://www.ubuntu.com) with the XFCE window manager. From here, you can follow instructions in the welcome file on the desktop to run simulations with the installed packages. Never worry about software dependencies again!


File Machine View Input Devices Help
Applications Menu
Welcome to the Full Virt...
23:11
virtualmicromagnetics
Trash
Nmag_examples.html
File System
Welcome.html
Home
Fidimag_examples.html
OOMMF_examples.html
Magpar_examples.html
Welcome to the Full Virtual Micromagnetics Simulation Environment! - Mozilla Firefox
Welcome to the Full Virt...
file:///home/virtualmicromagnetics/Desktop/Welcome.html
Search
Choose What I Share
Right Ctrl

Welcome to the Full Virtual Micromagnetics Simulation Environment!

This virtual machine contains a number of distributable software packages that are often used by the micromagnetic community to simulate relevant problems. These software can be downloaded from this machine; tarballs are available in the home directory. Micromagnetic software includes (in no particular order):

OOMMF 1.2a5 (The Object Oriented MicroMagnetic Framework): http://math.nist.gov/oommf/
Documentation: http://math.nist.gov/oommf/doc/

Nmag 0.2: http://nmag.soton.ac.uk/nmag/
Documentation: http://nmag.soton.ac.uk/nmag/current/manual/singlehtml/manual.html

Magpar 0.9 (Parallel Finite Element Micromagnetics Package): http://www.magpar.net/
Documentation: http://www.magpar.net/static/magpar-0.9/doc/html/

Fidimag 1.0 (Finite-difference Micromagnetic and Atomistic Package): http://computationalmodelling.github.io/fidimag/
Documentation: http://fidimag.readthedocs.org/en/latest/

Firefox automatically sends some data to Mozilla so that we can improve your experience.

Next, see *Virtual Micromagnetics Environments and Simulation Software* for the environments that are available besides the Full Virtual Micromagnetics environment, and the software on these environments, or you can read on to learn about *container* virtualisation.

Containers

*Container*s are an alternative virtualization technology to virtual machines. To start *Virtual Micromagnetics* containers, you will need:

- Docker >= 1.6.2 (https://docs.docker.com/engine/installation/)

Your user will need to be in the docker group on the machine you are running on[1]. After restarting your machine, command the following in an empty directory:

```
docker run -ti virtualmicromagnetics/lite:release /bin/bash -l
```

This command will download the Full Virtual Micromagnetics container image from the Internet to your computer, and start bash in your shell. Alternatively, you can run a simulation in the container without connecting to it explicitly by commanding the following, with `square.mif` in your current directory:

```
docker run -v $PWD:/host/ virtualmicromagnetics/oommf:release oommf boxsi /host/
↪square.mif
```

Now take a look at *Virtual Micromagnetics Environments and Simulation Software* for the environments that are available besides the Full Virtual Micromagnetics environment.

Virtual Micromagnetics Environments and Simulation Software

In *Getting Started: As a User*, we created a *virtual machine* based on the Full *Virtual Micromagnetics* environment. Here, we detail the micromagnetic simulation packages and other software supported and used by Virtual Micromagnetics, and we describe the list of available Virtual Micromagnetics environments.

Software

The following micromagnetic simulation packages are supported by Virtual Micromagnetics:

- OOMMF (http://math.nist.gov/oommf/)
- MagPar (http://www.magpar.net/)
- Nmag (http://nmag.soton.ac.uk/nmag/)
- Fidimag (http://computationalmodelling.github.io/fidimag/)

Each environment that uses a simulation package also contains its examples, and a link to its documentation page which can be opened in the virtual machine. The following infrastructure software is also used in all environments:

- Ubuntu (http://www.ubuntu.com/) 14.04
- XFCE4 (http://www.xfce.org/)
- Python (https://www.python.org/) 2.7, IPython (http://ipython.org/)

Environments

The Full environment contains all simulation packages supported by Virtual Micromagnetics, as well as dependencies requested by our users, including:

[1] Note that the docker group is root-equivalent, so you will likely need to own the machine to use Virtual Micromagnetics containers. To avoid this, consider using a different provider that does not require root privileges, but beware, as Docker is the only container provider supported at present.

- Cython(http://cython.org/)
- FEniCS (http://fenicsproject.org/)
- Gmsh (http://gmsh.info/)
- Netgen (https://sourceforge.net/projects/netgen-mesher/)
- ParaView (http://www.paraview.org/)
- Sundials (http://acts.nersc.gov/sundials/)

We recognise that many of our users will not require these tools. To that end, a "Lite" environment can be used instead, which contains all of the simulation packages the Full environment does, without these optional packages. To download and use the Lite Virtual Micromagnetics environment, command:

```
vagrant init virtualmicromagnetics/lite
vagrant up --provider virtualbox
```

Note that this is syntactically similar to the command used in *Getting Started: As a User*, and can be adapted for all other Virtual Micromagnetics environments. Environments exist for specific simulation packages, such as "virtualmicromagnetics/oommf". The following table shows the list of environments available under Virtual Micromagnetics, and the software they contain:

Software vs. Environment		Full	Lite	OOMMF	Magpar	Nmag	Fidimag
Micromagnetic	OOMMF						
	Magpar						
	Nmag						
	Fidimag						
Infrastructure	Ubuntu						
	XFCE						
	Python 2						
	IPython						
Other	Cython						
	FEniCS						
	Gmsh						
	Netgen						
	ParaView						
	Sundials						

See *Getting Started: As a Poweruser* if more fine-grained control over your software interests you. However, we firstly recommend reading *Virtual Machines and Related Software* to understand more about *virtual machine*s, *virtual environment*s, and the software used to create them in the Virtual Micromagnetics project.

Virtual Machines and Related Software

In *Getting Started: As a User*, we created a *virtual machine* based on the Full *Virtual Micromagnetics* environment using a virtual machine *provider* and *manager*. Here, we detail some of the underlying mechanisms of *virtual machine* creation, what providers, managers, and *provisioner*s are, and how virtual machines and *virtual environment*s are related through these virtual machine software.

Virtual Machines

A *virtual machine* is a software that imitates a certain other software, usually an operating system or environment, on a certain hardware. For our purposes however, it can be thought of as an operating system running in an operating system. Virtual machines are useful because they allow software tasks to be undertaken in precisely defined environments with little repercussion on the host system they run on. These machines themselves can be described by single files representing the equivalent of a hard disk of the machine, as well as a description of the hardware that the virtual machine emulates. This makes them simple to distribute.

Software Related to Virtual Machines

Virtual machines must be supported by software in order to function. Only provider software is necessary to run a virtual machine, but managers and provisioners are useful for creating virtual machines and virtual environments respectively.

Providers

Virtual machine providers are virtualiser software that supports creating, running, destroying, and other interaction with *virtual machine*s on your computer. VirtualBox (https://www.virtualbox.org) is an example of a virtual machine provider that is open source and freely available under the GNU GPL (https://www.gnu.org/copyleft/gpl.html), and is the provider supported by this project. Other popular provider software includes VMWare, KVM, and Docker. While many cloud computing organisations use virtual machines, they typically use existing provider software.

Managers

While not essential for starting *virtual machine*s, specialist software is useful for managing *virtual environment*s. Vagrant (https://www.vagrantup.com) is an example of a virtual machine manager. It provides a command-line interface to the creation and provision of virtual machines from virtual environments. HashiCorp, the company behind Vagrant, also provides a framework for sharing virtual environments. Most importantly for our purpose, Vagrant can be automated to generate virtual machines containing an environment without user intervention. This environment can then be used to complete our objectives. To specify this environment however, provisioning software is required.

Provisioners

Provisioning is the action of running select commands on a machine, virtual or otherwise, to reach a desired end state. By "state", here we mean how storage is populated with packages, environment definitions, and more. The user of a provisioner describes the desired state of the system, and the provisioner makes it so. In the absence of a provisioner, shell commands can be executed to specify the state, but this becomes unwieldy for large projects because focus is placed on the instructions needed to obtain the desired state, as opposed to the state itself. Provisioning software, such as Ansible (https://www.ansible.com) alleviates this problem. Ansible uses Yet Another Markup Language (YAML) to describe plays to run on a machine to enact the desired end state. Since the focus is on the end state of the system, *idempotency* is essential.

Summary

*Virtual machine*s imitate hardware and software, but must be hosted on a *host machine*. Virtual machines are provided by *provider* software running on the host machine, and can be provisioned for use by a *provisioner* software. *Manager* software links these two concepts, allows the virtual machine to be preserved and distributed as a *virtual environment*, and simplifies the creation of virtual machines.

You can learn more about *Containers and Related Software*, or you can *get started as a poweruser*, which explains how to create custom environments containing software you choose, as well as instructions for adding new software or configuring your own virtual environment.

Containers and Related Software

In *Getting Started: As a User*, we created a *container* in addition to a *virtual machine* based on the Full *Virtual Micromagnetics* environment. This page talks about containers and how the *provider*-*manager*-*provisioner* model applies to container creation. We recommend reading *Virtual Machines and Related Software*, if you have not already.

Containers and Images

A *container* is a virtualisation mechanism similar to a *virtual machine*, in that it allows users to run software in a controlled environment with a cap on available computing resources (like memory). Where a virtual machine contains the entire software stack above and including the operating system, a container uses the operating system and kernel of the host machine to produce its environment. Containers are created from *image*s (container templates, analogous to *box file*s) to run a single process, and are usually destroyed once that process has been completed.

Images can be distributed to other users so that they can run *Virtual Micromagnetics* environments from a container.

Software Related to Containers

As with *virtual machine*s, *container*s require supporting software to function. The *provider*-*manager*-*provisioner* model outlined for virtual machines in *Virtual Machines and Related Software* also applies to containers as follows:

- Provisioner: Ansible is also used to provision containers. In *Virtual Micromagnetics*, scripts to provision containers and virtual machines with software are very similar, encouraging code reuse.
- Manager: Vagrant is used to manage containers in this project in a similar way to how virtual machines are managed. An exception is that Vagrant can only be used to host *box file*s, meaning another hosting method is needed for images.
- Provider: Docker is used in Virtual Micromagnetics to create containers for simulation (as the user) and for provisioning (as the poweruser). Docker also supports online hosting of images; this is used in Virtual Micromagnetics as a distribution method.

Summary

*Container*s are another virtualisation mechanism, like virtual machines. Containers virtualise fewer elements of the software stack so they are smaller, but consequently impose more requirements on the *host machine*. Like virtual machines, containers can be provisioned, managed, and distributed.

You are now ready to *get started as a poweruser*, which explains how to create custom environments containing software you choose, as well as instructions for adding new software or configuring your own virtual environment.

Getting Started: As a Poweruser

In *Virtual Micromagnetics Environments and Simulation Software*, we learned about the different *Virtual Micromagnetics* environments available to users, which bundle sets of configured software. Here we outline how you can create virtual environments yourself, which you can distribute to others. We recommend reading *Virtual Machines and Related Software*, if you have not already.

To create a new, custom *virtual environment*, you will need the following software in addition to the software list in *Getting Started: As a User*:

- A GNU/Linux operating system (we use Ubuntu, https://www.ubuntu.com)

- 2.7 <= Python < 3.0 (https://www.python.org/)
- 1.9 <= Ansible < 2.0 (https://www.ansible.com/)

With this software:

1. Grab a copy of Virtual Micromagnetics. You can do this via Git by cloning our repository with `git clone -b release https://github.com/computationalmodelling/virtualmicromagnetics.git`, or by grabbing a release version at https://github.com/computationalmodelling/virtualmicromagnetics/releases
2. Install some sub-packages:
   - Ansible role "blockinfile"; command `ansible-galaxy install yaegashi.blockinfile -proles/` from the Virtual Micromagnetics software directory
   - Vagrant plugin "vagrant-vbguest"; command `vagrant plugin install vagrant-vbguest`
3. Make, with `make` from the Virtual Micromagnetics software directory, with an Internet connection.

After time of the order of hours, you should find a *box file* at `./artefacts/virtualmicromagnetics-full-*.box`. If not, see *Troubleshooting* for more help. If so, congratulations on building your first Virtual Micromagnetics environment! This file represents your virtual environment, which you can share with other users. You can use this environment yourself by commanding the following in an empty directory:

```
vagrant init $PATH_TO_BOX_FILE
vagrant up --provider virtualbox
```

where `$PATH_TO_BOX_FILE` is the aforementioned artefact file. Now that you can create *virtual environment*s, see *Developer Notes (valid for version 1.1.0)* to learn how to customise the software you place on them.

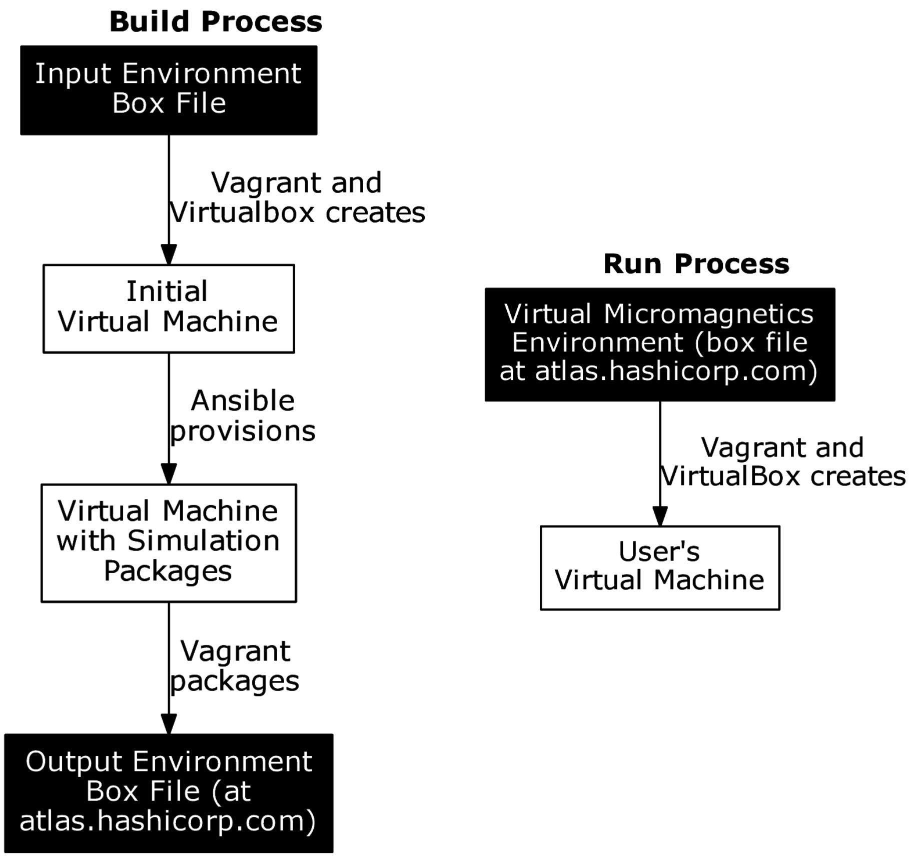
Developer Notes (valid for version 1.1.0)

In *Getting Started: As a Poweruser*, we created a virtual environment from scratch that can be shared with other users. Here, we show how you can completely specify your own environment. Knowledge of Ansible is needed, which can be gleaned from their excellent documentation at http://docs.ansible.com/ansible/.

Build and Run

The Build and Run Processes for Virtual Machines

This graph shows the operations involved in the build and run processes for *virtual machine*s.

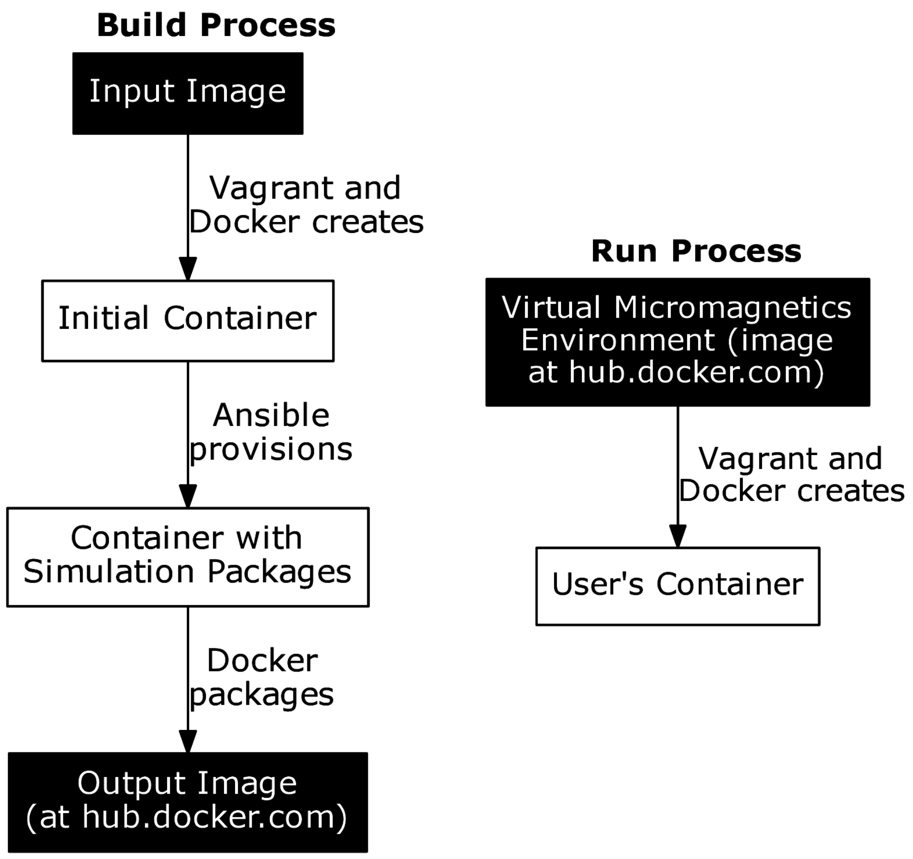

Build Process
Input Environment Box File
Vagrant and Virtualbox creates
Initial Virtual Machine
Ansible provisions
Virtual Machine with Simulation Packages
Vagrant packages
Output Environment Box File (at atlas.hashicorp.com)
Run Process
Virtual Micromagnetics Environment (box file at atlas.hashicorp.com)
Vagrant and VirtualBox creates
User's Virtual Machine

The run process is simple: the user follows the instructions in *Getting Started: As a User* to create a virtual machine for themselves. Lets break down the build process:

- Input Environment Box File -> Initial Virtual Machine: The input environment is a *virtual environment* containing only the operating system and few convenience tools. In development, Vagrant and VirtualBox create a *virtual machine* from this environment in the `create_vm` role (see *Build and Run*).
- Initial Virtual Machine -> Virtual Machine with Simulation Packages: Vagrant commands Ansible to provision this machine using an Ansible playbook.
- Virtual Machine with Simulation Packages -> Output Environment Box File: Vagrant then packages the virtual machine into a new virtual environment, which can be distributed to others. Tagged releases are uploaded by administrators to atlas.hashicorp.com, where they become available to all Vagrant users.

The Build and Run Processes for Containers

In the same way, this graph shows the operations involved in the build and run processes for *container*s.


Build Process
Input Image
Vagrant and Docker creates
Initial Container
Ansible provisions
Container with Simulation Packages
Docker packages
Output Image (at hub.docker.com)
Run Process
Virtual Micromagnetics Environment (image at hub.docker.com)
Vagrant and Docker creates
User's Container

Again the process of creating and running a container as a user is as simple as following the instructions in *Getting Started: As a User*. The build process is also similar; a container template (*image*) is downloaded, a container is created and provisioned, and the container is packaged as an image for download by all Docker users. The key difference is that Docker pushes the image to https://hub.docker.com/ as opposed to Vagrant.

Details

The build (make) process in step 3 in *Getting Started: As a Poweruser* allowed us to create a virtual environment. The Makefile in the software repository can build multiple targets. Each target runs Ansible on the `master.yml` playbook, which in turn runs the `create_vm` or `create_container` role in the roles directory. This creates a virtual machine or container and provisions it with the playbook passed as a command-line argument in `Makefile`, which lives in the jobs directory. It will also do some post-provisioning tasks using the hookbook, again passed as a command-line argument. The fundamental difference between the playbook and the hookbook is that the playbook is run on the guest virtual machine by *Vagrant*, and the hookbook is run on the host machine. Different Makefile targets may place different build artefacts in the artefacts directory.

Roles add or configure software, playbooks describe the roles that must be enacted to provision the machine, hookbooks describe what to do with that machine (like creating a *box file*), and jobs are Makefile targets that produce certain machines. To add a new environment, one needs to add a job that follows the pattern of existing jobs.

Where Things Are

In order to add jobs, one should edit `Makefile`. In order to do that, one would need to know where things are, hence the purpose of this section. The *virtual micromagnetics* repository is structured as follows:

- `Makefile`: This is the Makefile through which all jobs are conducted.
- `ansible.cfg` and `inventory.txt`: These files are used by Ansible when the master.yml playbook is run. They contain configuration information.
- `roles/`: This directory contains roles (obviously). Each role is given a subdirectory, and should not overlap. Each role directory contains tasks, and may also contain the subdirectories:
  - `vars/` (variable definitions),
  - `templates/` (files to duplicate to the guest virtual machine),
  - `meta/` (metadata, such as role dependencies),
  - `files/` (files used by tasks that aren't covered by the usecases of templates)
- `jobs/`: This directory contains playbooks and directories that can be thought of as jobs in `Makefile`. They are either provisioning playbooks, or post-provisioning hookbooks.
- `machines/`: This directory is created by `Makefile`, and houses the vagrant environment for each individual virtual machine. The provision process is recorded to a log file in the machine's directory (for example, the provision log for the lite build job exists in machines/virtualmicromagnetics-lite/virtualmicromagnetics-lite.log)
- `artefacts/`: This directory is created by `Makefile`, and houses build artefacts.

Examples

Create New Machine with Existing Software

Lets create a custom machine called doc-example, that contains Fidimag but no X server. Firstly, we add a target to `Makefile` (append the following to the `Makefile`):

```
# This target builds a virtual hard disk file containing an OOMMF and Fidimag
# installation.
doc-example-vm:
    ansible-playbook master.yml -c local -i localhost, -v -k --extra-vars="type=vm vm_
↪name=virtualmicromagnetics-doc-example playbook=provision_virtualmicromagnetics_doc-
↪example.yml hookbook=hook_vm.yml extra_resources_dir=guest_resources/"
```

Now we need to describe what the state of the machine should be, by writing the playbook *jobs/provision_virtualmicromagnetics_doc-examples.yml*:

```
---
# This Ansible playbook is a provision playbook designed to be used with
# vagrant. This playbook provisions a machine suitable for micromagnetic
# simulation with Fidimag. It is executed by the virtual machine.

- hosts: all

  vars:
    vm_name: virtualmicromagnetics-doc-example

  roles:
    - fidimag
    - fidimag_examples
    - add_super_user
    - { role: set_hostname, HOSTNAME: {{ vm_name }} }
```

Now we are ready to build the environment by commanding (again, from the repository root directory):

```
make doc-example
```

This creates another *virtual environment* in the artefacts directory.

Adding Software

In *Build and Run*, we introduce roles. Roles can add new software to a *virtual environment*. By way of example, we can create a role to install Emacs (https://www.gnu.org/software/emacs/) from the Ubuntu software repository. We firstly create a directory structure:

```
# Create a role for Emacs.
mkdir --parents roles/emacs/tasks
```

Now we introduce some content using information from the Ansible documentation (http://docs.ansible.com/ansible/, and http://docs.ansible.com/ansible/apt_module.html). Write the following to `roles/emacs/tasks/main.yml`:

```
---
# This Ansible playbook installs Emacs.

- name: Install Emacs.
  apt:
    pkg=emacs
    state=latest
    update_cache=yes
    cache_valid_time=86400
  sudo: yes
```

This role, when run, will ensure that the latest version of Emacs and its dependencies are installed on the virtual machine, and updates the apt cache. Roles can be parameterised and have dependencies, which can cause them to become complicated. By way of example, installing Emacs on the new doc-example environment requires us to append the line:

```
- emacs
```

To clarify, playbook `jobs/provision_virtualmicromagnetics_doc-examples.yml` now looks like:

```
---
# This Ansible playbook is a provision playbook designed to be used with
# vagrant. This playbook provisions a machine suitable for micromagnetic
# simulation with fidimag. It is executed by the virtual machine.

- hosts: all

  vars:
    vm_name: virtualmicromagnetics-doc-example

  roles:
    - fidimag
    - fidimag_examples
    - add_super_user
    - { role: set_hostname, HOSTNAME: {{ vm_name }} }
    - emacs
```

Can I have a Container Too?

You certainly can, with minimal changes too. Add this target to your Makefile:

```
# This target builds a container image containing an OOMMF and Fidimag
# installation.
doc-example-container:
    ansible-playbook master.yml -c local -i localhost, -v -k --extra-vars=
↪"type=container container_name=doc-example playbook=provision_virtualmicromagnetics_
↪doc-example.yml hookbook=hook_container.yml extra_resources_dir=guest_resources/"
```

The only differences between this target and the one added previously are:

- The value of "type" is now "container", not "vm".
- The value of "hookbook" is now "hook_container.yml", not "hook_vm.yml".
- "vm_name=virtualmicromagnetics-doc-example" is now "container_name=doc-example".

Further Tinkering with the Virtual Machine

We have explored how a new *virtual environment* can be created, and how new software can be added. In this section, we describe how the virtual machine itself can be configured using Vagrant's parameters. Vagrantfiles are files used by Vagrant written using Ruby syntax. These files specify parameters of the *virtual machine* created from a virtual environment. When running the commands in *Getting Started: As a User*, we create a Vagrantfile in the working directory that describes the virtual machine to Vagrant. Vagrantfiles can also be built into a virtual environment. Built-in Vagrantfiles can be found in `guest_resources/vagrantfiles`.

For example, if you wish to specify that 2048MB of memory must be used in the virtual machine created in *Create New Machine with Existing Software*[1], we can add a builtin Vagrantfile at `guest_resources/vagrantfiles/Vagrantfile_virtualmicromagnetics-doc-example_builtin` with the following content:

```
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

  config.vm.provider :virtualbox do |vb|
    vb.memory = 2048
  end
```

This Vagrantfile will be detected by the hookbook and included automatically when the environment is packaged. For more information on Vagrantfiles, see the Vagrant documentation (https://www.vagrantup.com/docs/vagrantfile/).

Summary and Final Words

To summarise, *virtual environment*s are created from an empty Ubuntu virtual machine after being provisioned and packaged. This build process allows the user to create a Virtual Micromagnetics *virtual machine* using Vagrant and VirtualBox. A similar approach is used to create *image*s for *Docker* containers. We have also presented how a new environment can be created, how the software of that environment can be controlled, and how the virtual machines can be parameterised.

Thank you for using Virtual Micromagnetics! If you create roles for your favourite software, consider sharing them with the community. You can create a pull request at our GitHub repository at https://github.com/computationalmodelling/virtualmicromagnetics, or contacting Mark at mark[dot]vousden[at]soton[dot]ac[dot]uk.

[1] Note that this is not such a good idea if you want to distribute your environment to different users, since they may have a different amount of available memory to you.

Troubleshooting

Currently there is not troubleshooting information for Virtual Micromagnetics. If you have an issue, please add it to our issue tracker at https://github.com/computationalmodelling/virtualmicromagnetics/issues, or email mark[dot]vousden[at]soton[dot]ac[dot]uk. We will endeavour to resolve the issue and add it here if it is widespread.

Glossary

Box File A file used by *Vagrant* to create *Vagrant environment*s. A box file represents a template from which *virtual machine*s can be created.

Container "Containers are a lightweight virtualization method for running multiple isolated Linux systems under a common host operating system"[2]

As with *virtual machine*s, containers are a virtualisation medium. Unlike virtual machines however, they use components of the host operating system. This reduces their size at the cost of reduced isolation from the host machine.

Host

Host Machine A machine that uses virtualisation software to host *virtual machine*s and/or *container*s. Host machines allocate resources, including memory and disk space, to support the running of a virtual machine or container. Note that a virtual machine can host other virtual machines, making it both a host and a virtual machine.

Idempotent

Idempotency Idempotent *provisioning* runs only the commands required to achieve the desired state of the system. Commands that do not change the state are not run. This can save considerable build time when handling dependencies, because a dependency should not be downloaded and reinstalled if it is already in the state it should be in.

Image A *container* image acts as a template from which containers can be created. Images are to containers and *box file*s are to *virtual machine*s.

Manager

Vagrant Manager software provides an interface for managing *virtual machine*s and *container*s. Vagrant is an example of manager software, which provides a command-line interface to virtual machine and container management which can be automated. See *Managers*.

Vagrant Environment The directory structure (including `Vagrantfile`) created by Vagrant when `vagrant init` is commanded. A single *virtual machine* or *container* can be managed in a given environment.

Virtual Environment Virtual environments produce the same *virtual machine* or *container* on all virtualisation-capable computers. Virtual environments are also *box file*s, but are not *Vagrant environment*s.

Provider

Docker

Virtualiser

VirtualBox Provider (or virtualiser) software supports the creation of, and interaction with, *virtual machine*s or *container*s from a host machine. VirtualBox and Docker are examples of free provider software for virtual machines and containers respectively. See *Providers*.

Provisioner

Provisioning

[2] Jacobsen, D.M., Canon, R.S. (2015). "Contain This, Unleashing Docker for HPC". Proceedings of the Cray User Group.

Ansible Provisioner software runs a set of commands on a machine, virtual or otherwise, to ensure it is in a particular state. Ansible is an example of provisioner software that enables *idempotent* provisioning. See *Provisioners*.

System Virtual Machine

Virtual Machine "An efficient, isolated duplicate of a real machine."[1]

Software that imitates certain other software on certain hardware. In this project, this includes a complete operating system, and a combination of one or many simulation packages and dependencies. General system virtual machines are described in brief at *Virtual Machines and Related Software*.

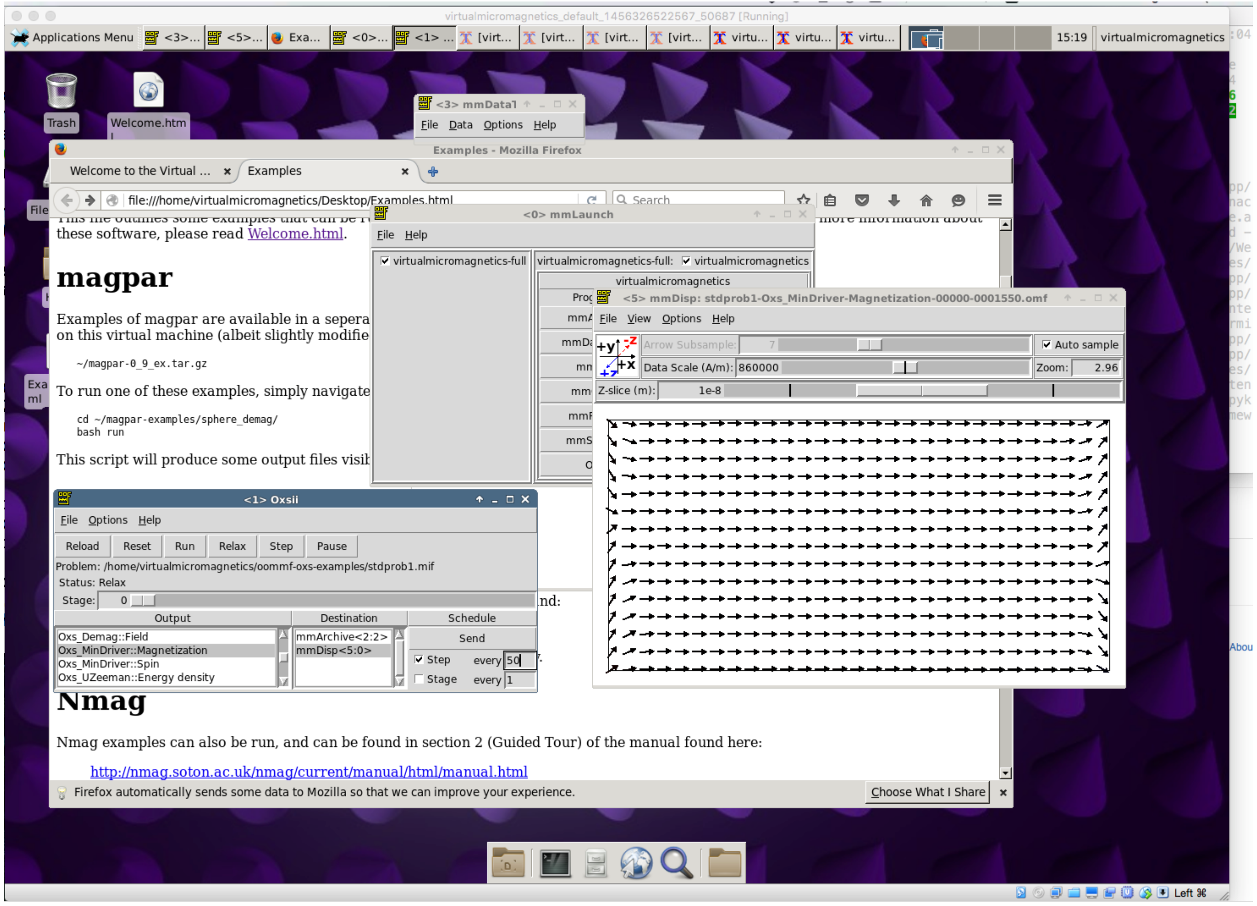
More strictly, a virtual machine is a specific instance of a *virtual environment*. When we build *box file* artefacts, we are creating *virtual environment*s, not virtual machines. Once *Vagrant* creates a *Vagrant environment* from a *box file* and `vagrant up` is commanded, a corresponding virtual machine is created. Virtual machines created on a *host* are managed by virtual machine *provider*s, which typically list the machines they are maintaining.

Virtual Micromagnetics "Enabling accessible and reproducible micromagnetic simulation."

The name of this project, which represents the collection of virtual environments and the software written to create them.

References

[1] Smith, J., Nair, R. (2005). "The Architecture of Virtual Machines". Computer (IEEE Computer Society) 38 (5): 32–38. doi:10.1109/MC.2005.173


virtualmicromagnetics_default_1456326522567_50687 [Running]
Applications Menu
15:19
virtualmicromagnetics
Trash
Welcome.htm
<3> mmData1
Examples - Mozilla Firefox
Welcome to the Virtual ...
Examples
<0> mmLaunch
<5> mmDisp: stdprob1-Oxs_MinDriver-Magnetization-00000-0001550.omf
<1> Oxsii
File Options Help
Reload Reset Run Relax Step Pause
Output Destination Schedule
Send

file:///home/virtualmicromagnetics/Desktop/Examples.html

these software, please read Welcome.html.

magpar

Examples of magpar are available in a separa
on this virtual machine (albeit slightly modifie

```
~/magpar-0_9_ex.tar.gz
```

To run one of these examples, simply navigate

```
cd ~/magpar-examples/sphere_demag/
bash run
```

This script will produce some output files visit

Nmag

Nmag examples can also be run, and can be found in section 2 (Guided Tour) of the manual found here:

http://nmag.soton.ac.uk/nmag/current/manual/html/manual.html

Firefox automatically sends some data to Mozilla so that we can improve your experience. Choose What I Share

Problem: /home/virtualmicromagnetics/oommf-oxs-examples/stdprob1.mif
Status: Relax
Stage: 0

Oxs_Demag::Field
Oxs_MinDriver::Magnetization
Oxs_MinDriver::Spin
Oxs_UZeeman::Energy density

mmArchive<2:2>
mmDisp<5:0>

✓ Step every 50
Stage every 1

Arrow Subsample: 7 ✓ Auto sample
Data Scale (A/m): 860000 Zoom: 2.96
Z-slice (m): 1e-8

Fig. 1.1: OOMMF running in *Virtual Micromagnetics*.

Index