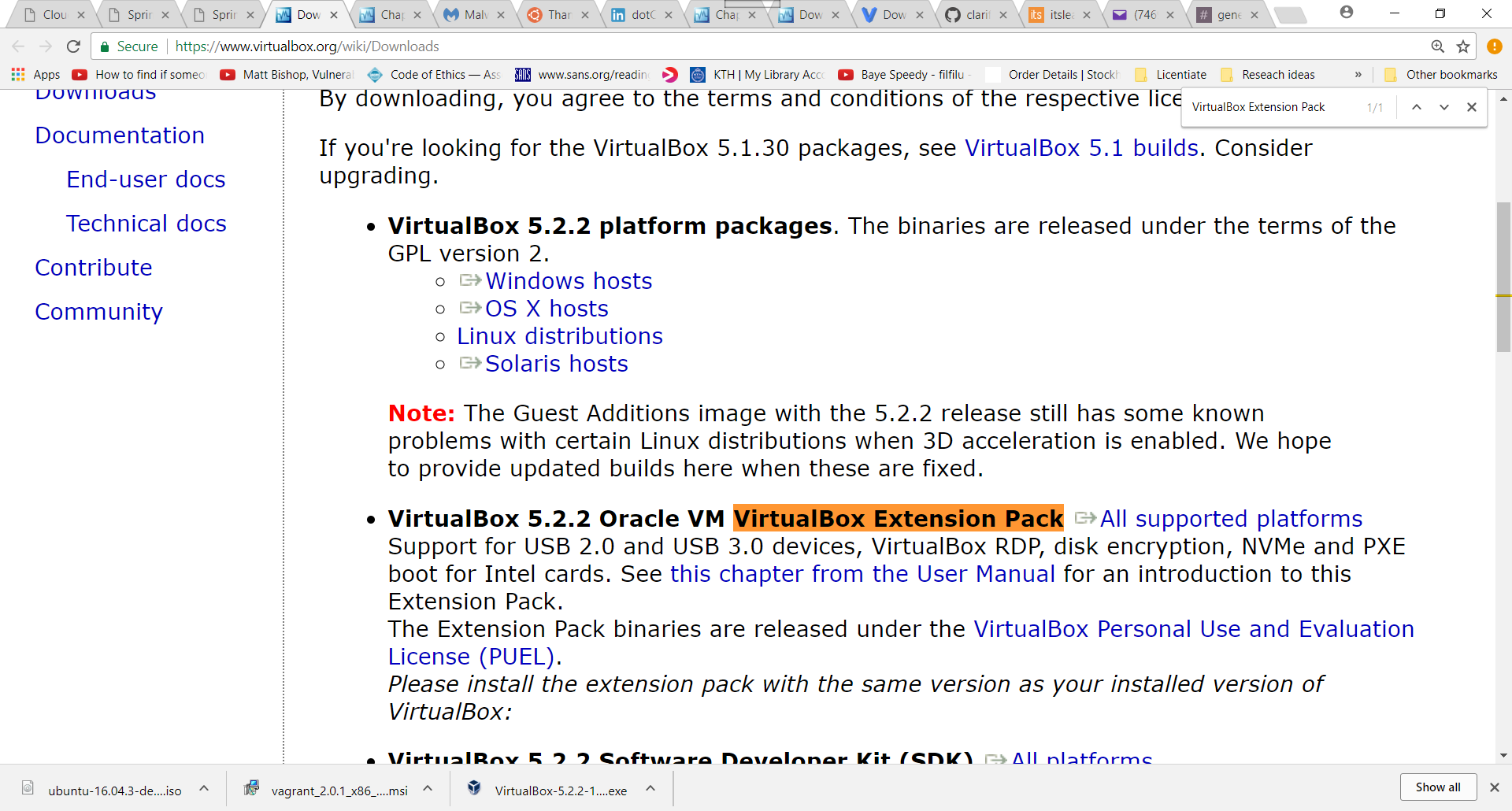
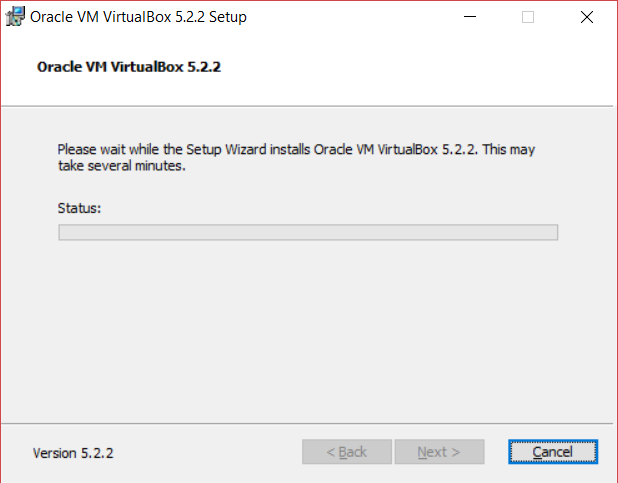
**Workneh Yilma Ayele**

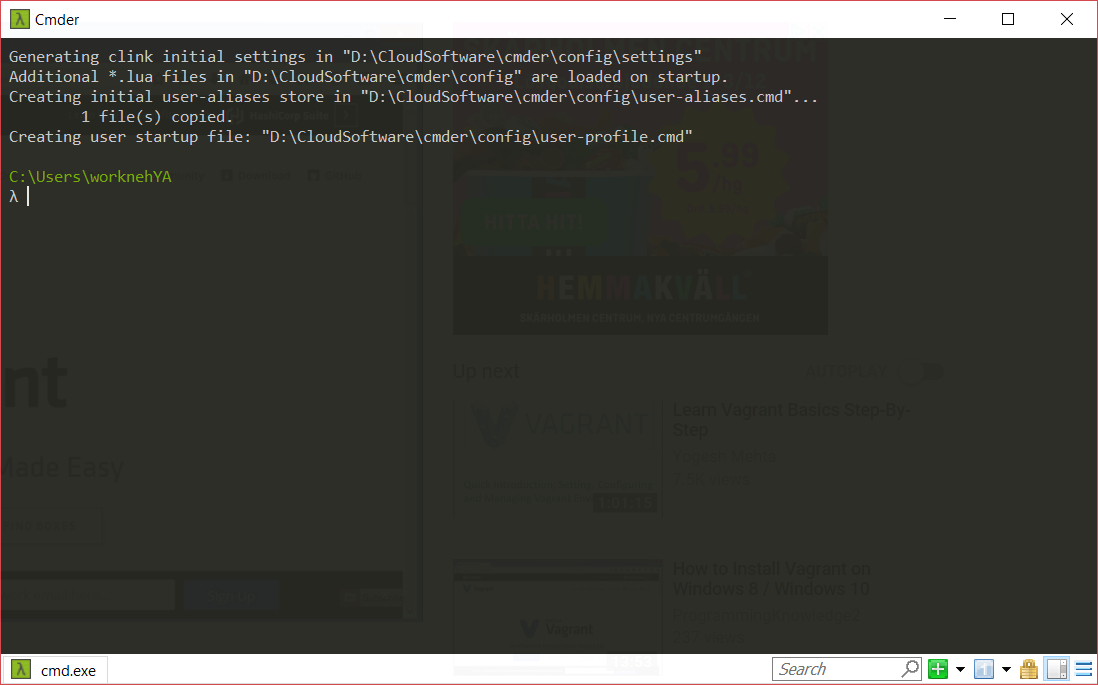
# Sprint: Getting started with Virtualisation

I installed VitualBox and VirtualBox extension pack, downloaded Ubuntu image and then installed Ubuntu as illustrated below.

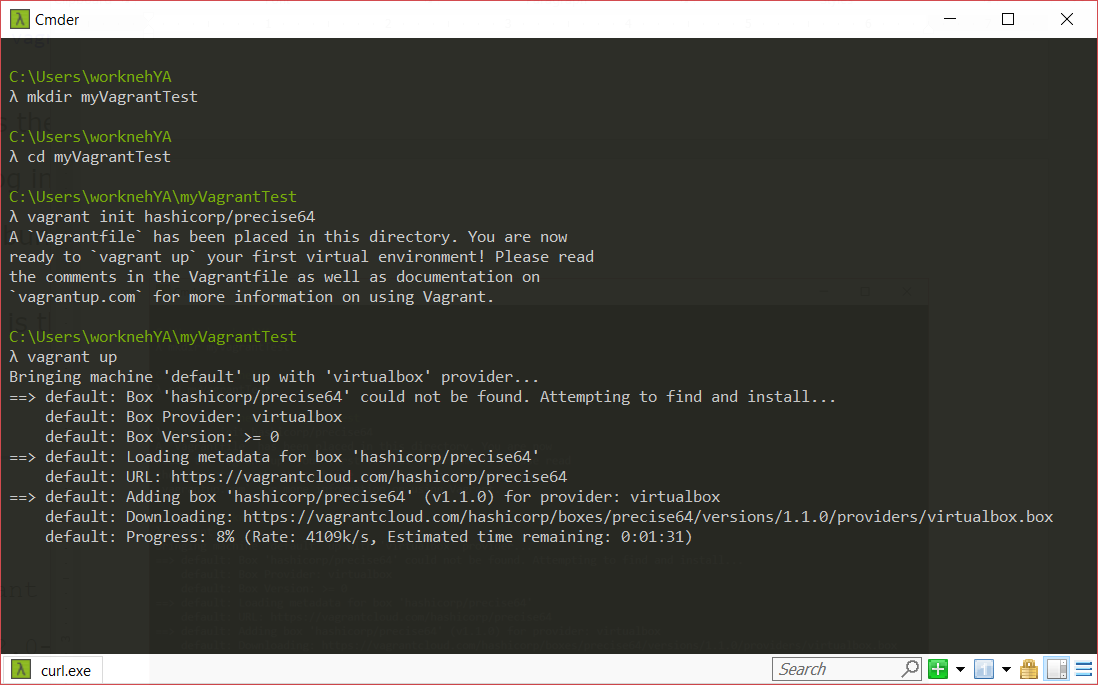


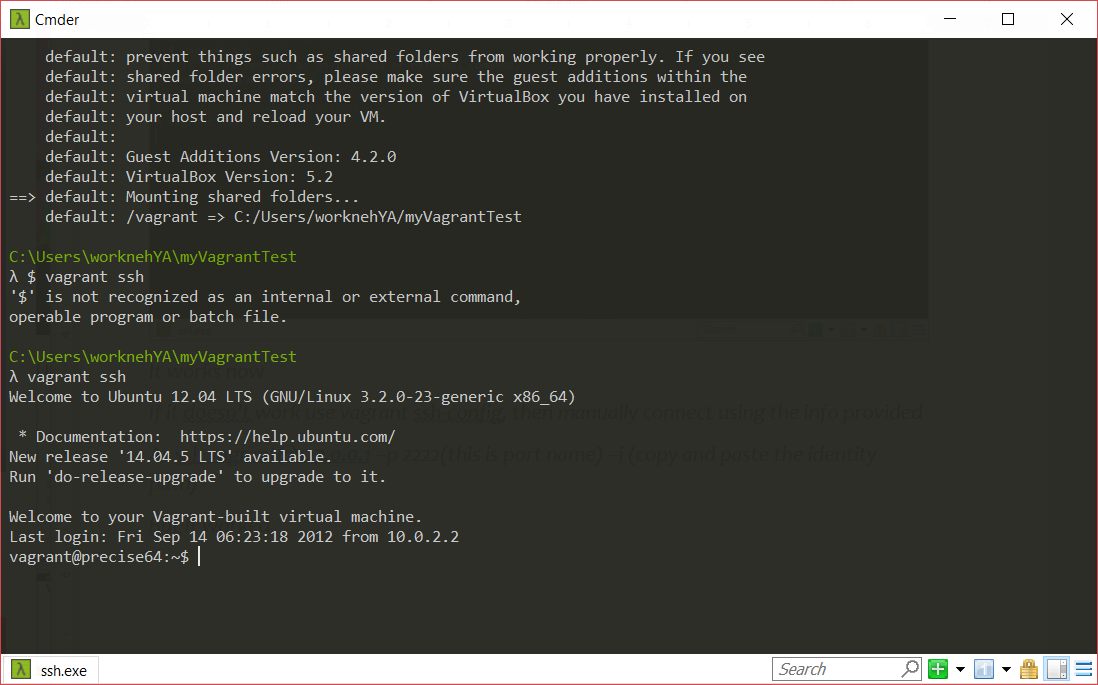
But installed vagrant on my Windows machine using **cmder.net** emulator, **cmder** emulator supports shh and supports unix commands, I used **cmder** emulator to work on my Windows machine to do all sprints.

***Cmder*** *– “is a software package created out of pure frustration over the absence of nice console emulators on Windows. It is based on amazing software, and spiced up with the Monokai color scheme and a custom prompt layout, looking sexy from the start” –* [*http://cmder.net/*](http://cmder.net/)



With **Cmder** – I didn’t install vagrant because the version I downloaded supports vagrant.



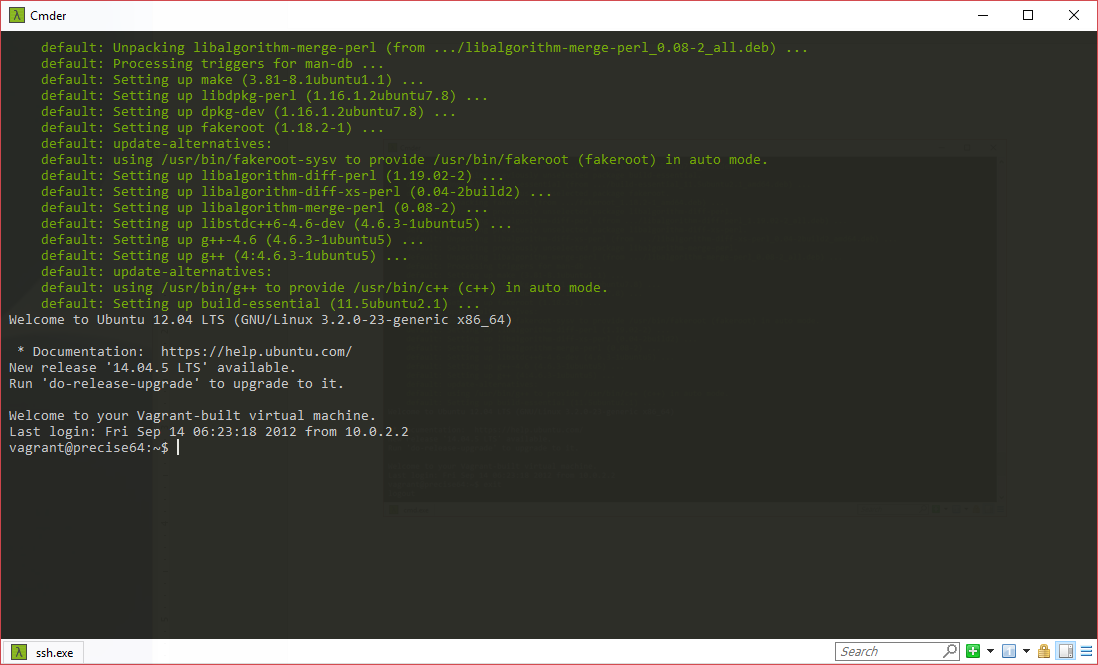


# Sprint: Provisioning

Installed git and nodejs using vagrant and then using puppet, see the illustration below.

**Provisioning with vagrant**

First, I used vagrant to install the packages and run the command (vagrant up && vagrant ssh)



Running the code

vagrant@precise64:~$ node --version && npm --version

*Welcome to your Vagrant-built virtual machine.*

*Last login: Fri Sep 14 06:23:18 2012 from 10.0.2.2*

*vagrant@precise64:~$ vagrant@precise64:~$ node --version && npm --version*

*vagrant@precise64:~$: command not found*

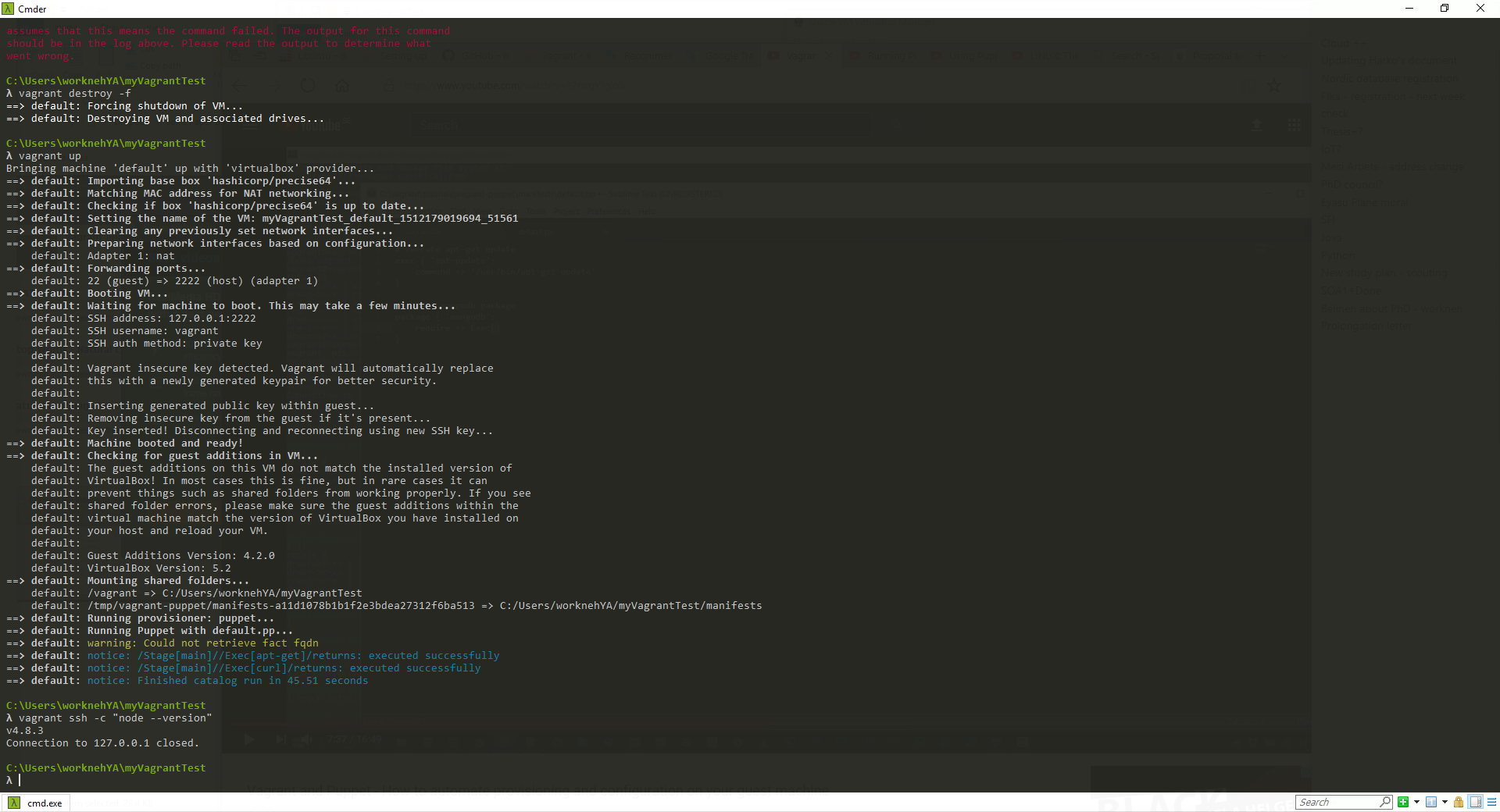
*vagrant@precise64:~$ node --version && npm --version*

*v4.8.3*

*2.15.11*

I exit and destroy the box

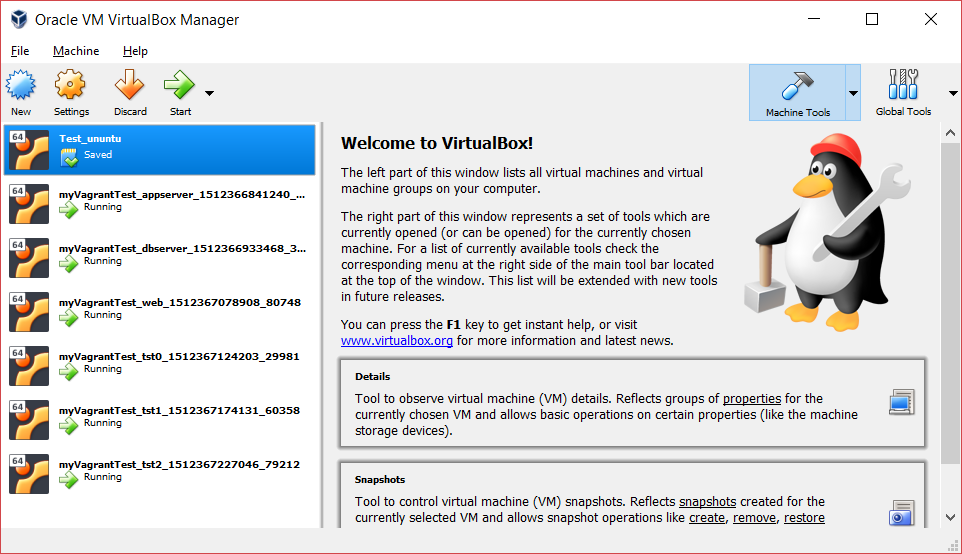
**Provisioning with puppet output**



# Sprint: Provisioning Multiple Machines

**Basic Multi-Machine Setup**

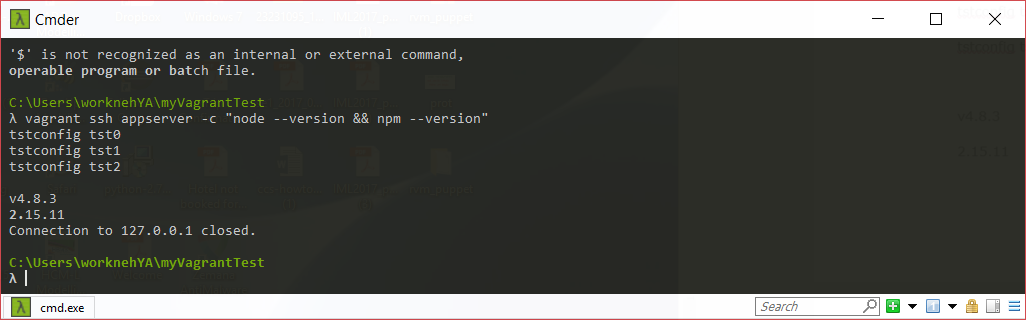
I created a multi-machine configuration to provision, appserver, dbserver, web, and tst0, tst1, tst2 as illustrated below.



I run the following

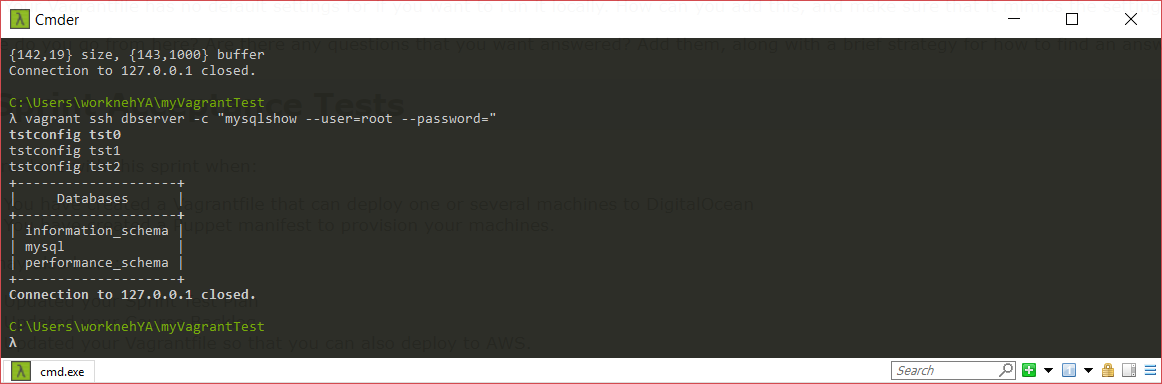
$ vagrant ssh appserver -c "node --version && npm --version"

The output is, the tstconfi tst0, tst1… is fixed in the configuration available in GIThub:



vagrant ssh web -c "node --version && npm --version && curl http://127.0.0.1:8080/"

$ vagrant ssh dbserver -c "mysqlshow --user=root --password=<YOUR\_ROOT\_PASSWORD>"



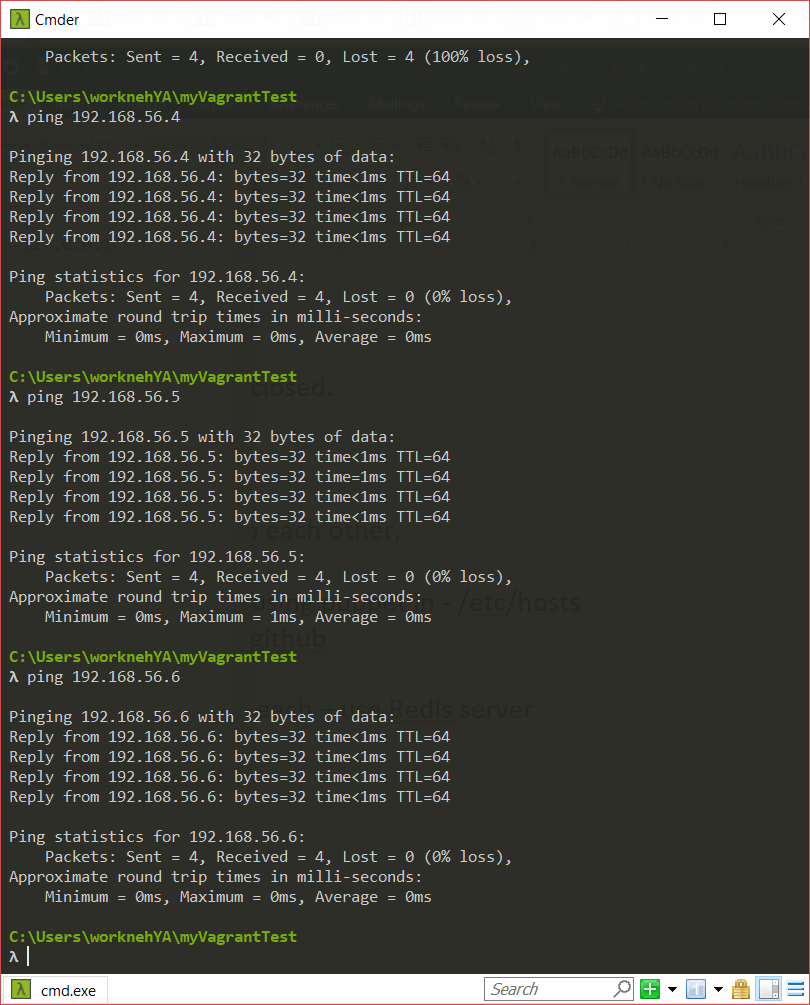
λ vagrant ssh appserver -c "node --version && npm --version"

v4.8.3

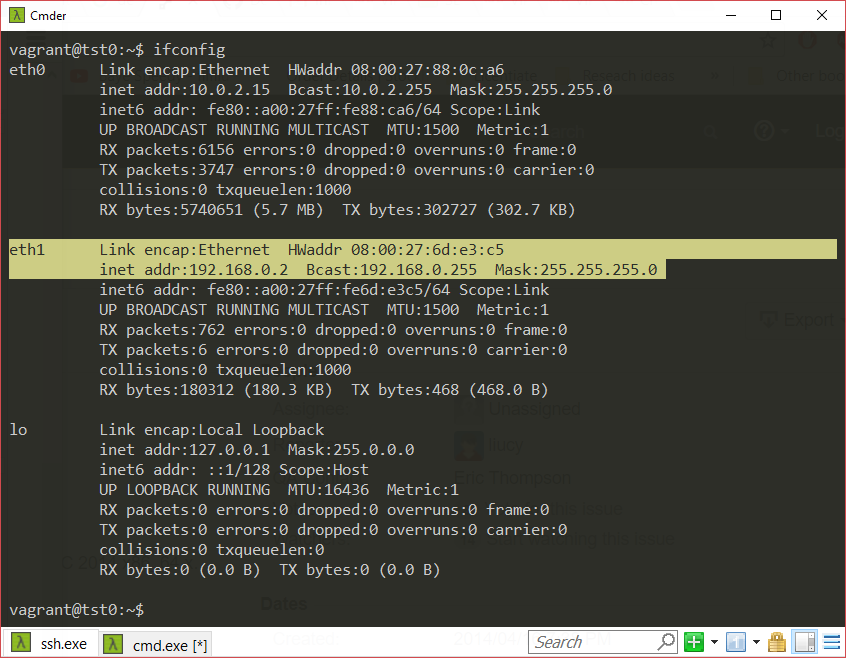
2.15.11

**Machine that talk to each other**

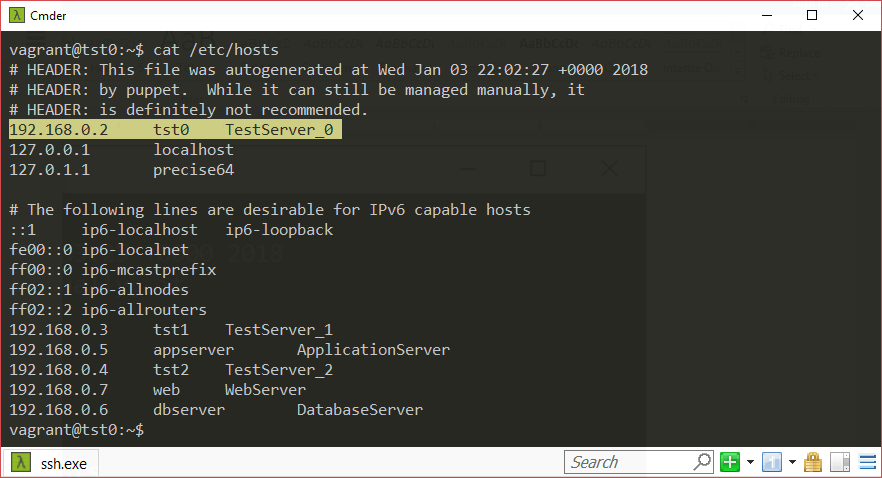
I created all host entries using puppet in - /etc/hosts



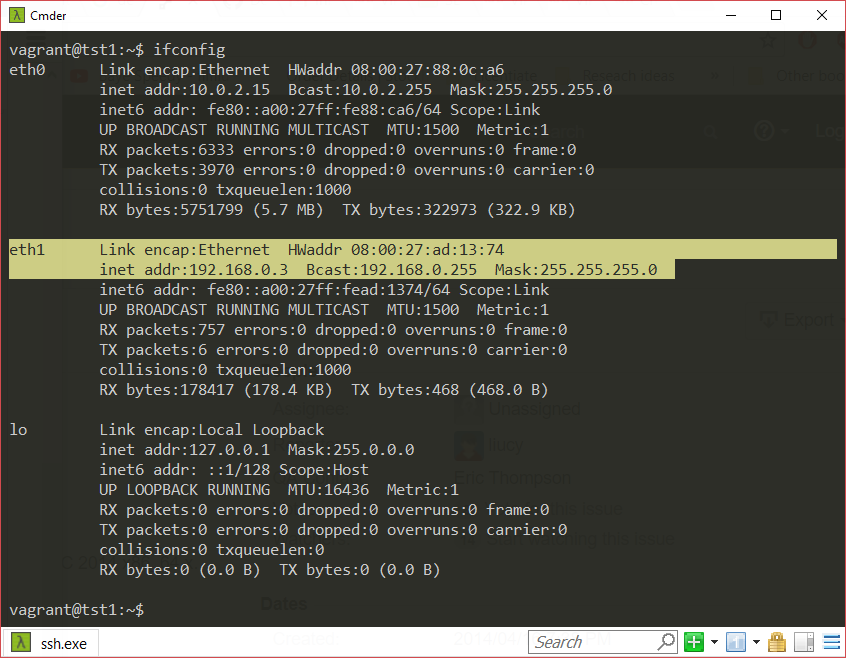
1. **Server: tst0 – ifconfig on tst0**



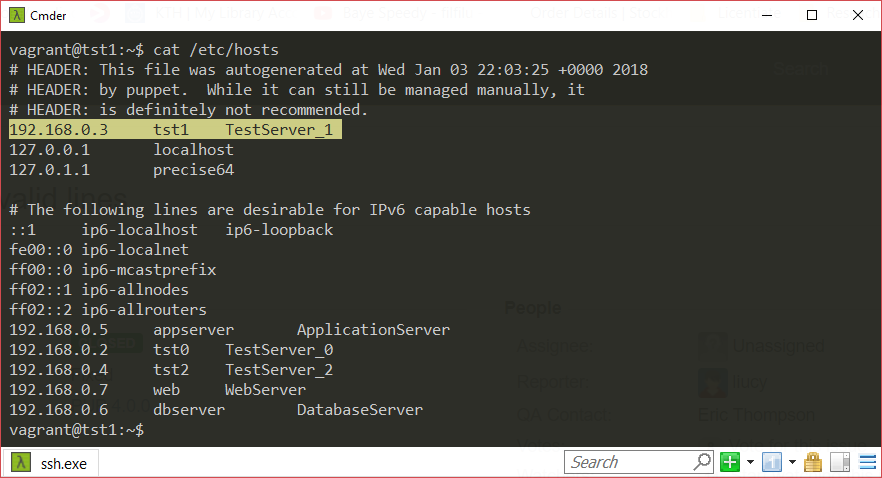
**Server: tst0** – host entries



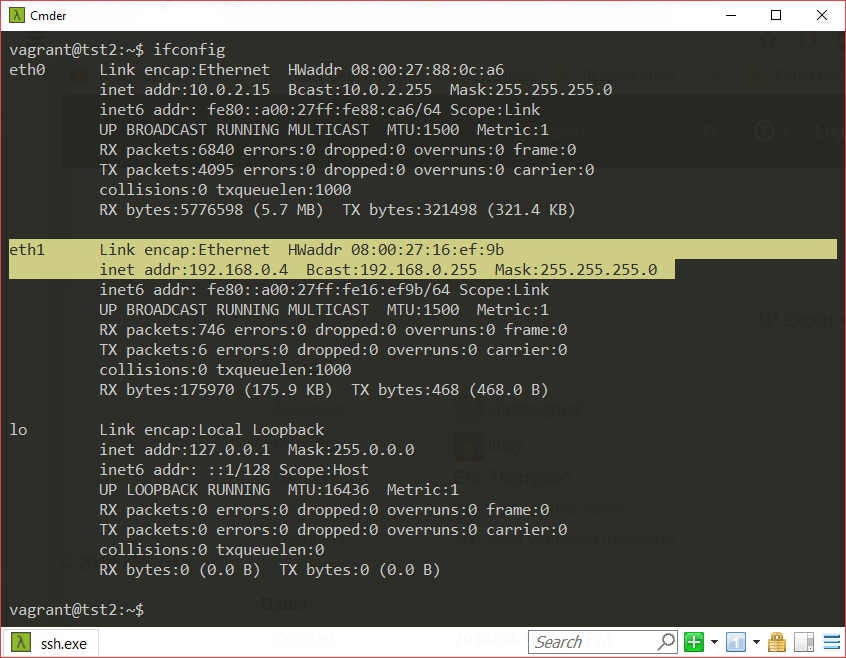
1. **Server: tst1 – ifconfig on tst1**



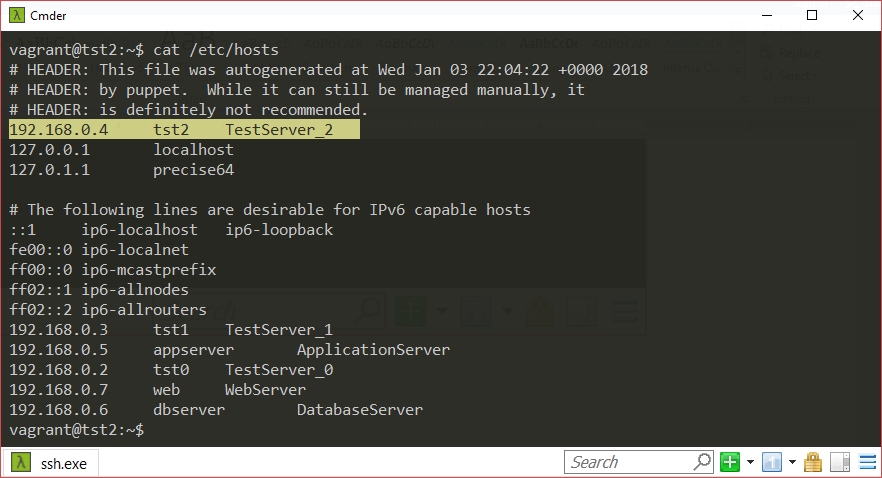
**Server: tst1** – host entries



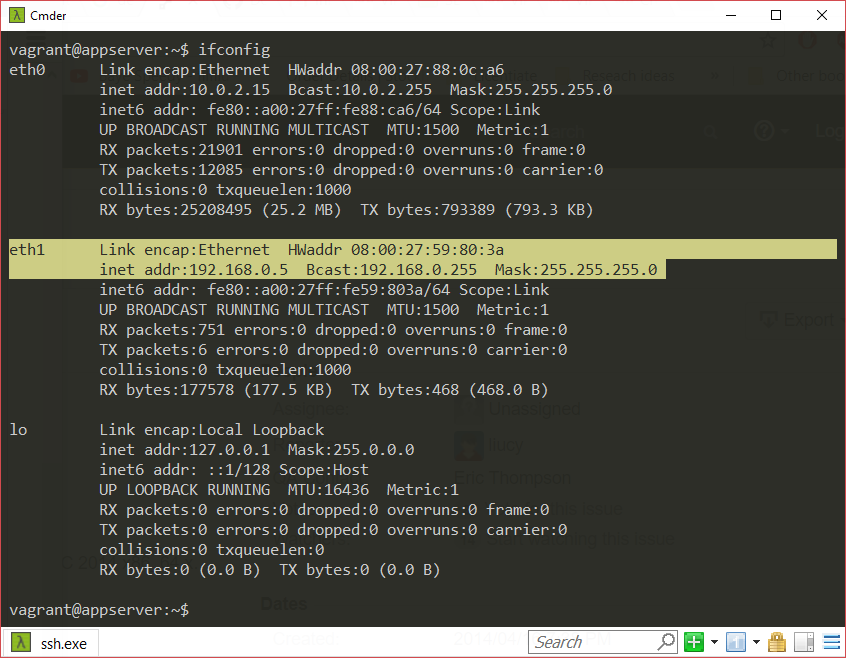
1. **Server: tst2 – ifconfig on tst2**



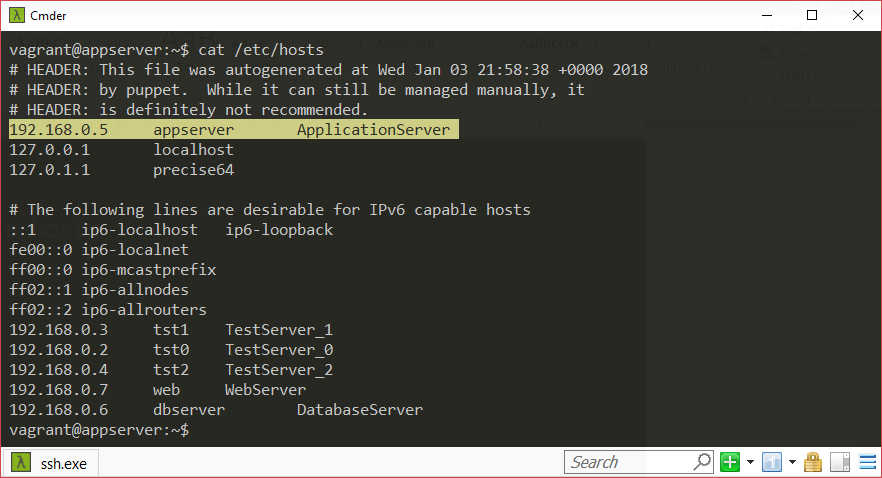
**Server: tst2** – host entries



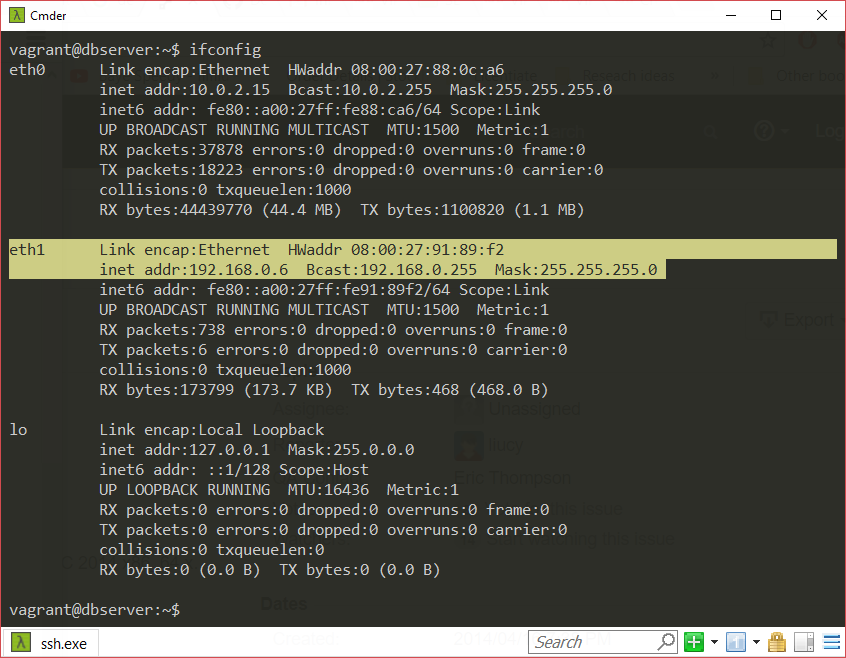
1. **Server: appserver – ifconfig on appserver**



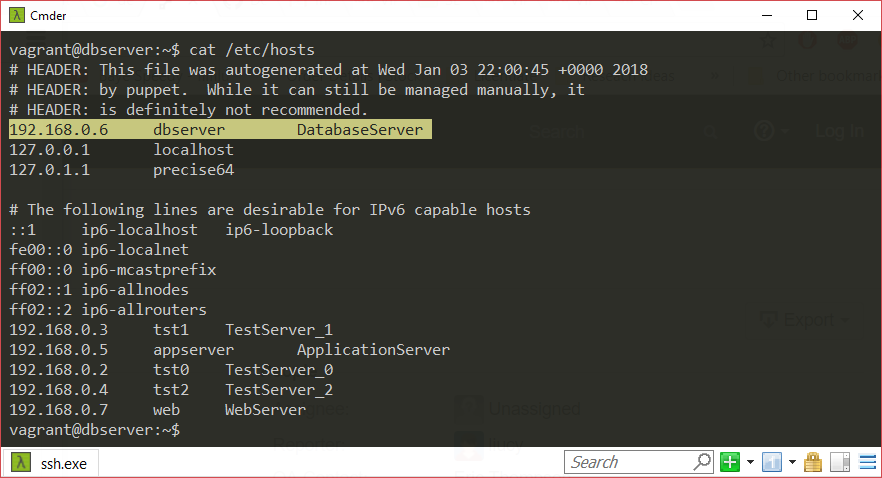
**Server: appserver** – host entries



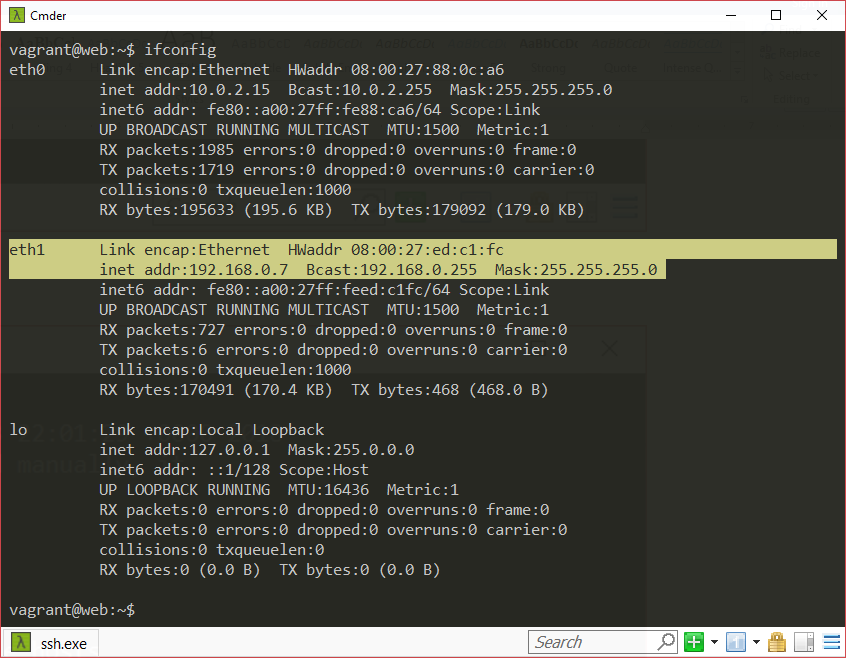
1. **Server: dbserver – ifconfig on dbserver**



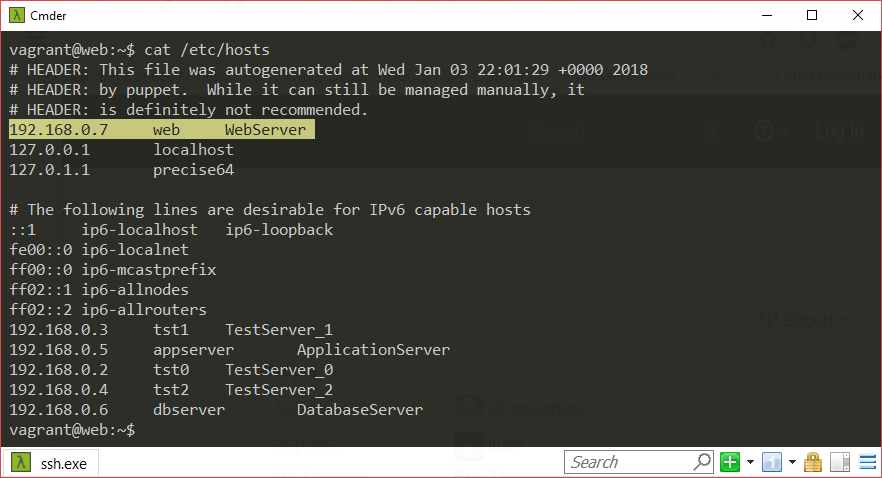
**Server: dbserver**  – ifconfig on dbserver

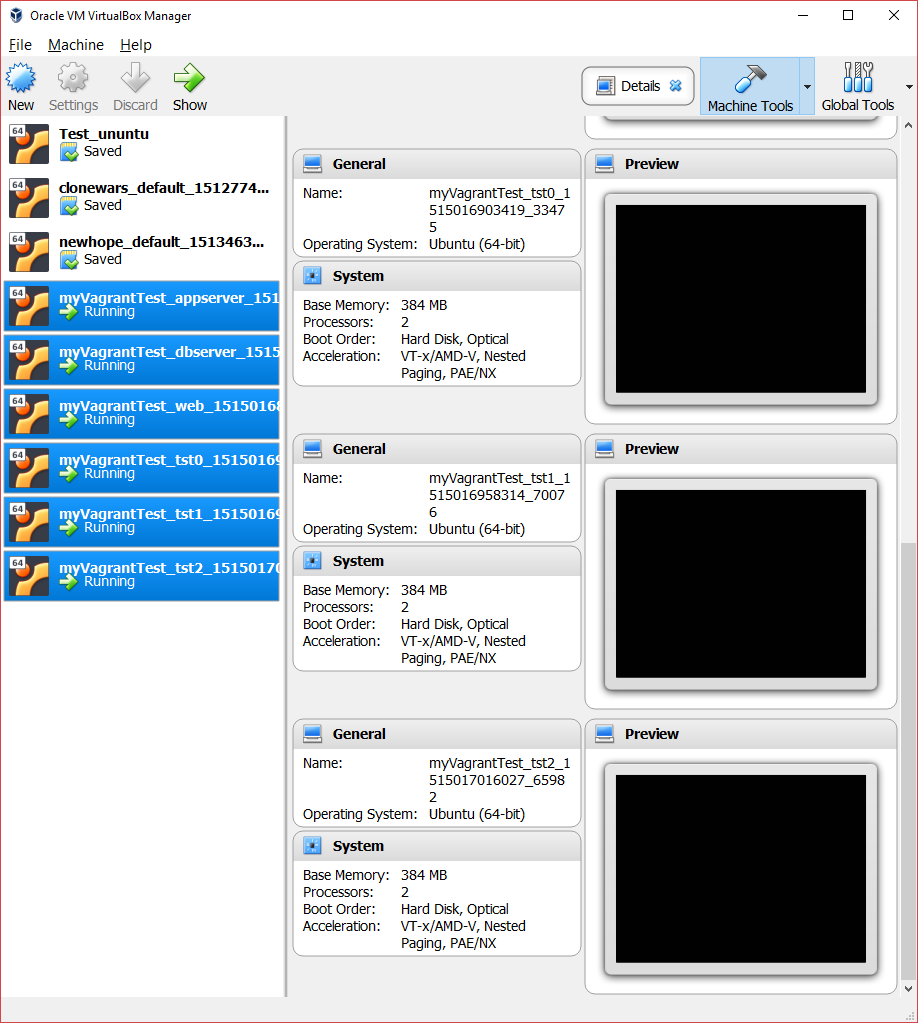


1. **Server: web – ifconfig on web**



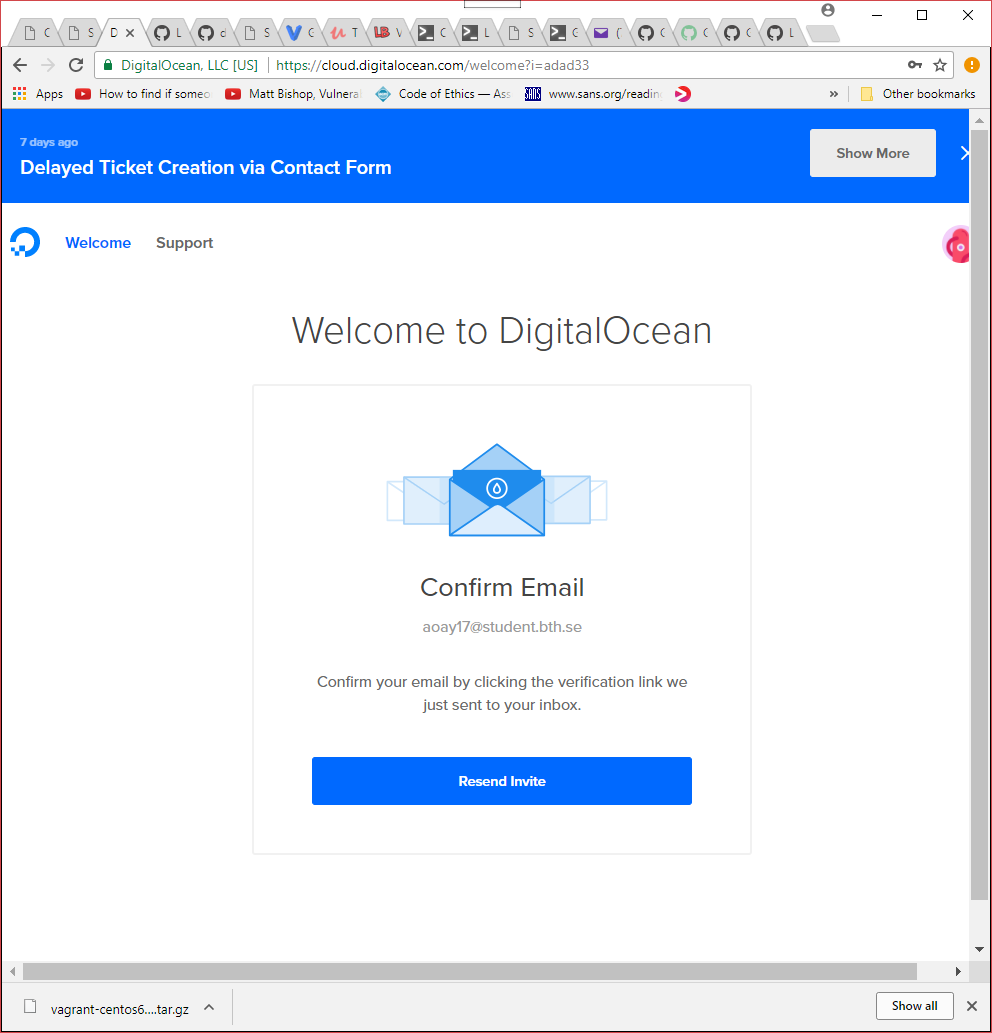
**Server: web** – ifconfig on web





# Sprint: Enter the cloud – digital ocean

I signed up to GitHub education pack and signed up to digital ocean with student and got 50$ bonus.

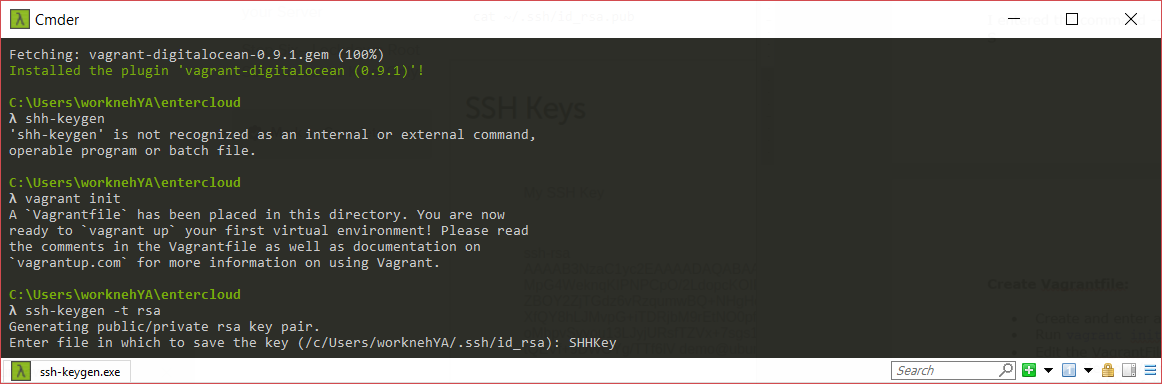


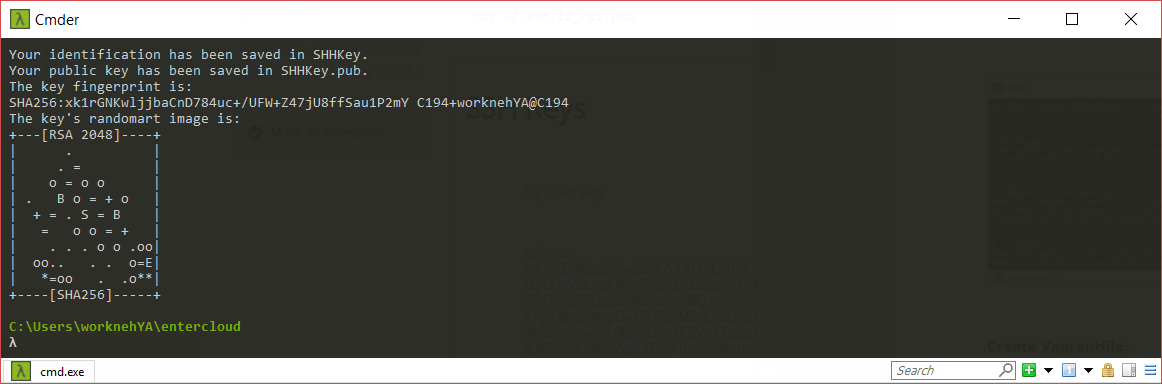
Then setup vagrant for digitalocean

vagrant plugin install vagrant-digitalocean

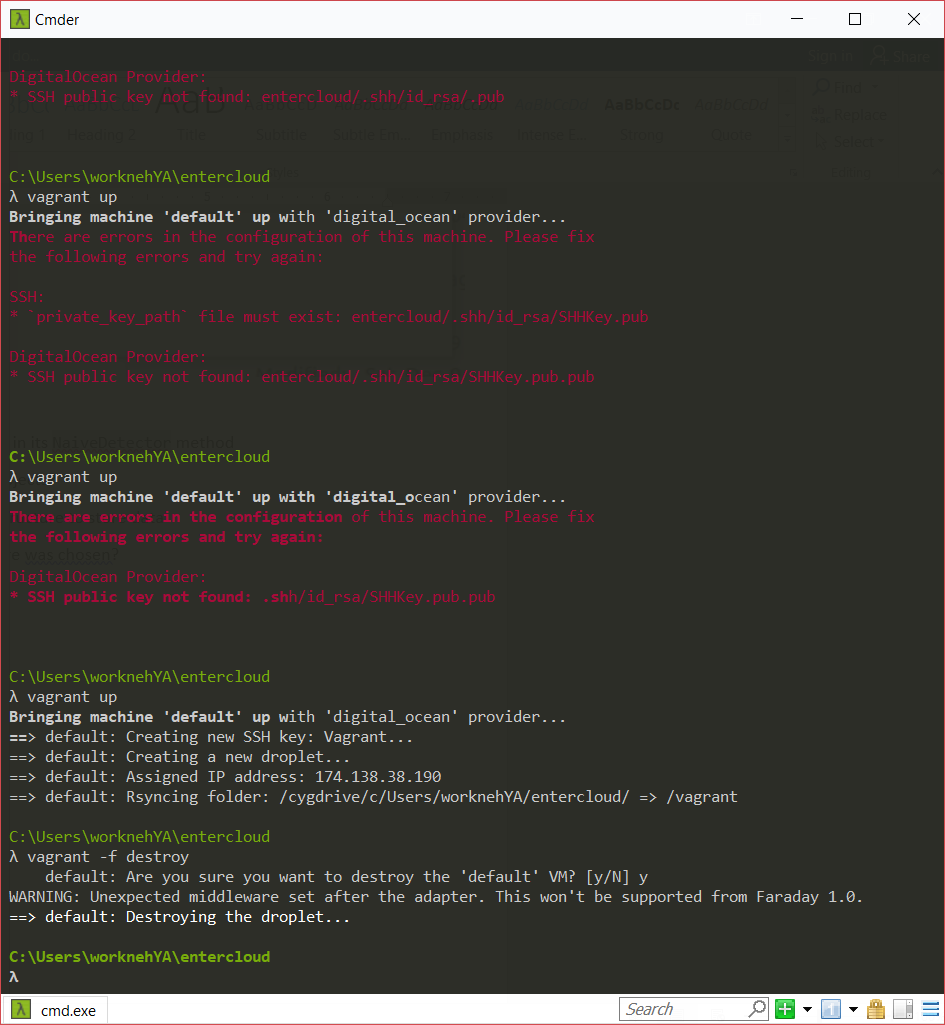


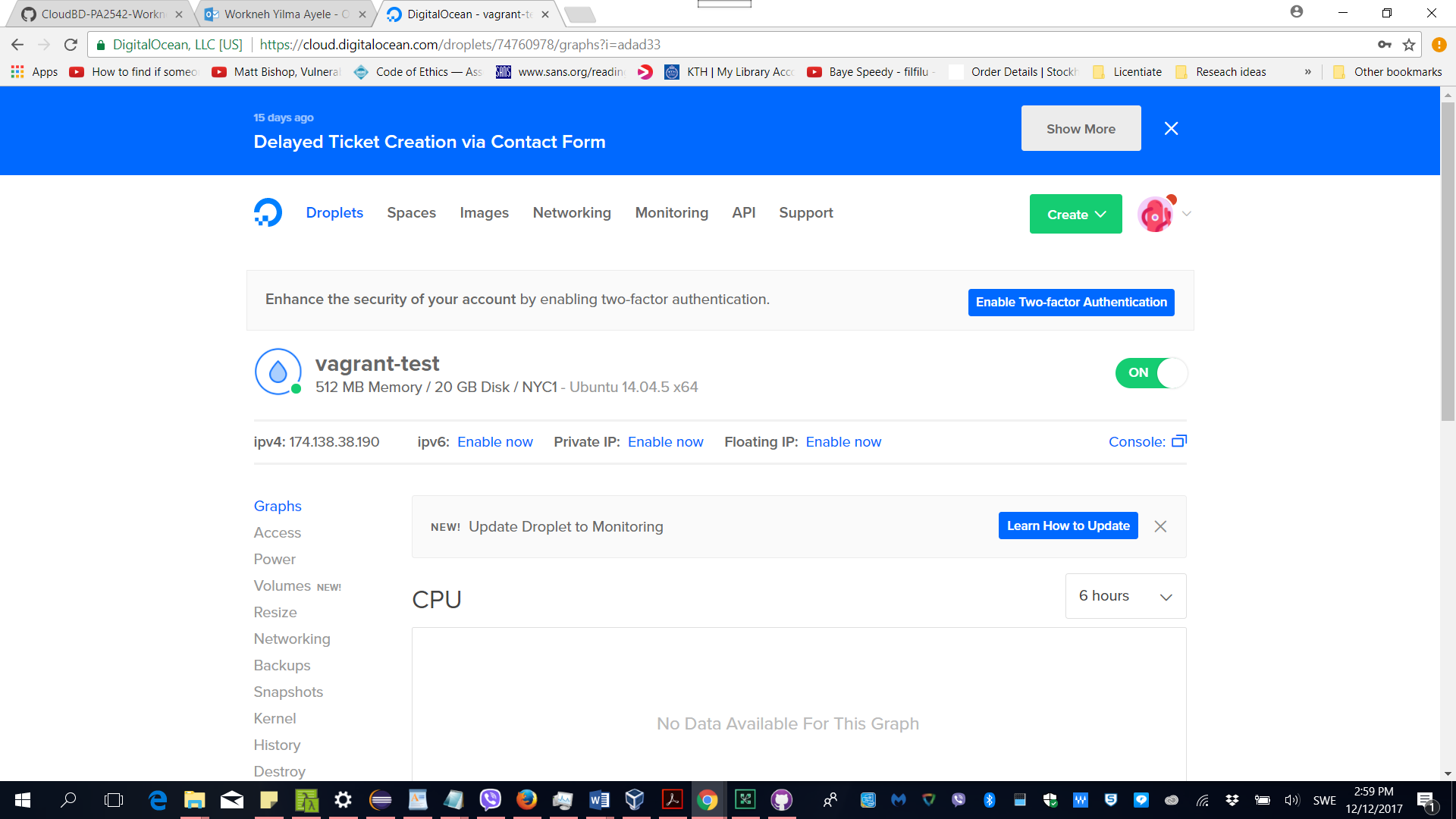
The generated a token from digital ocean and generate SSH key on my machine using **ssh-keygen**



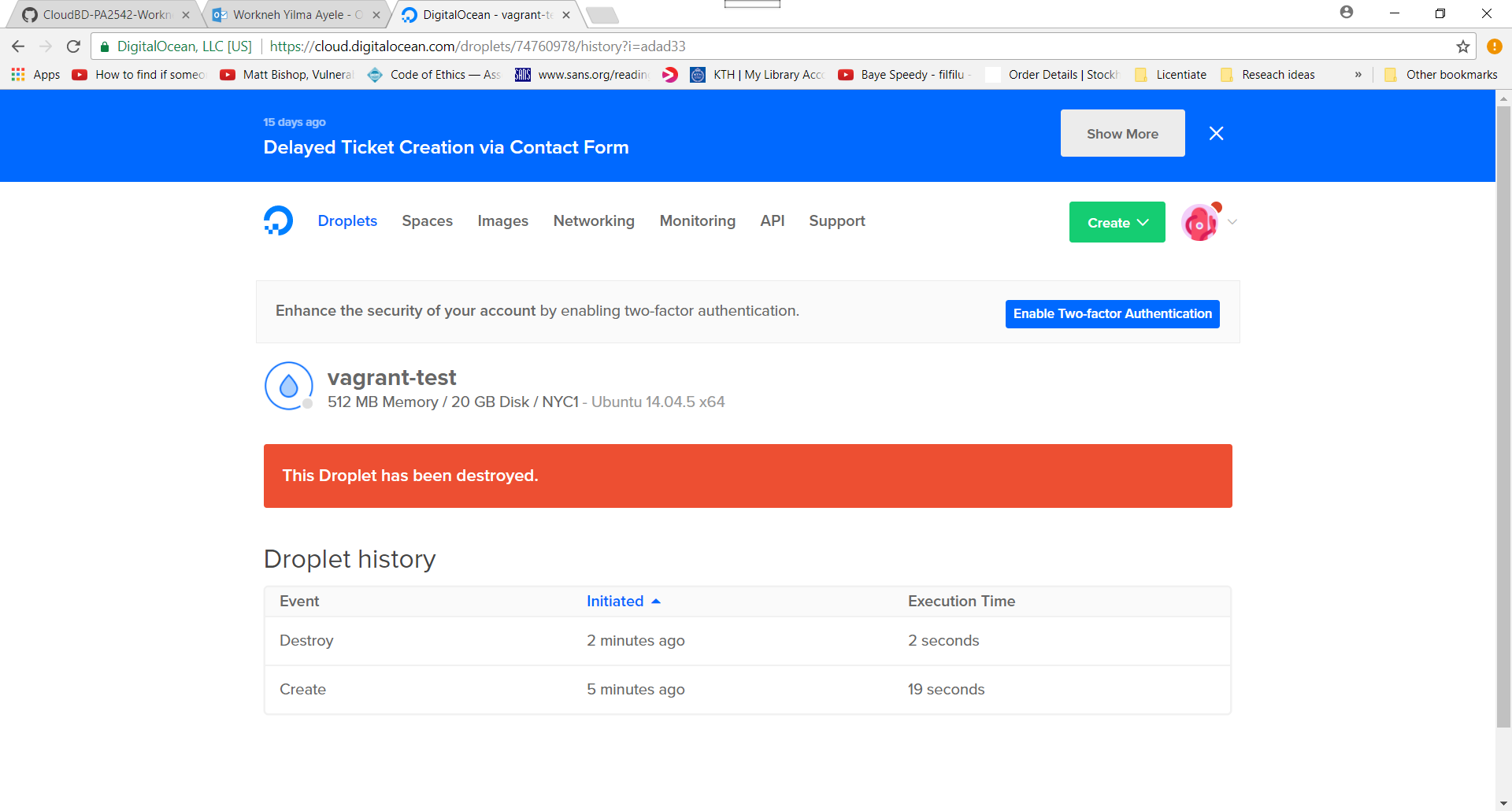


Then provisioned a single machine setup on digitalocean using vagrant



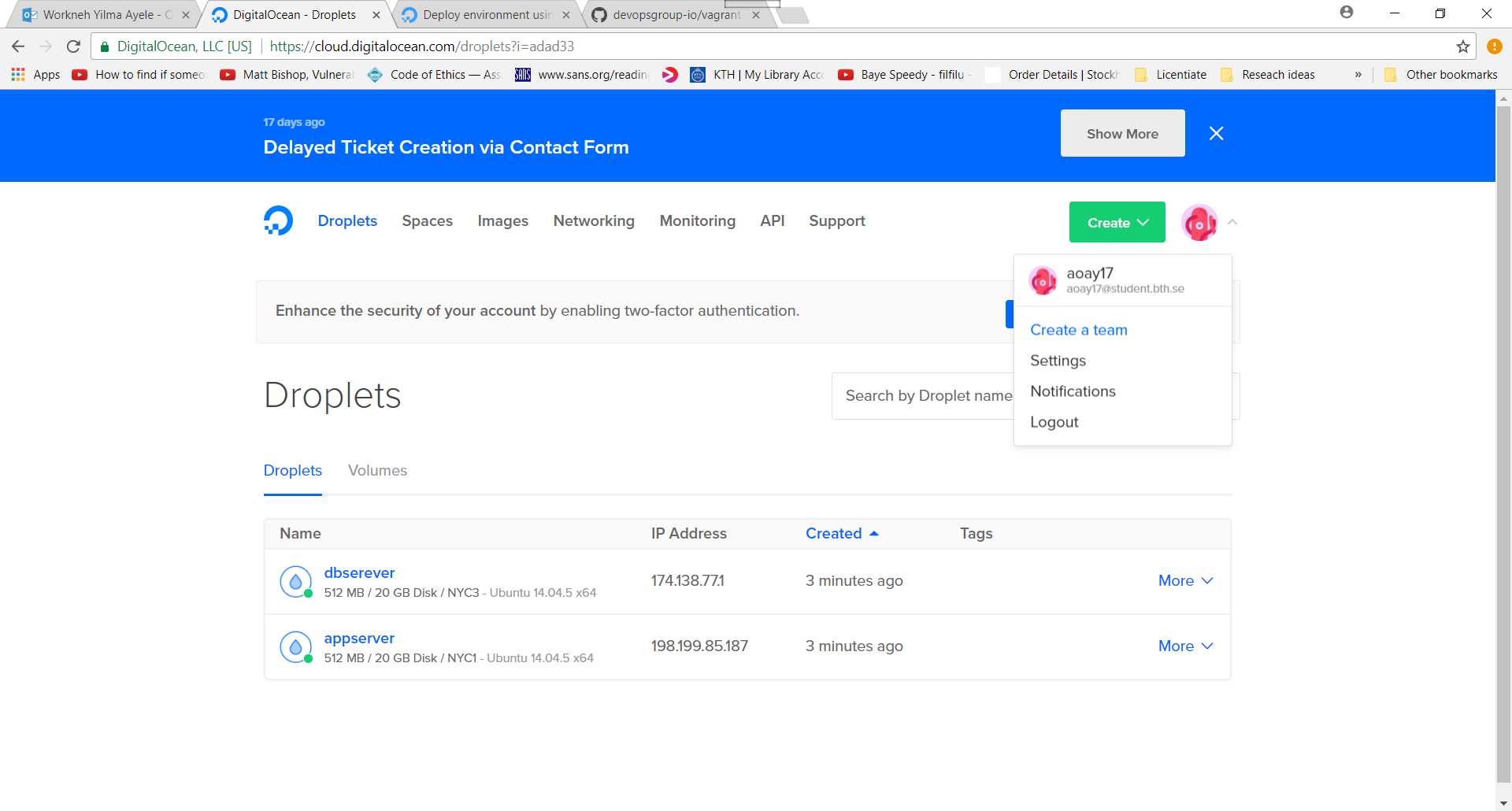


Then destroyed the droplet



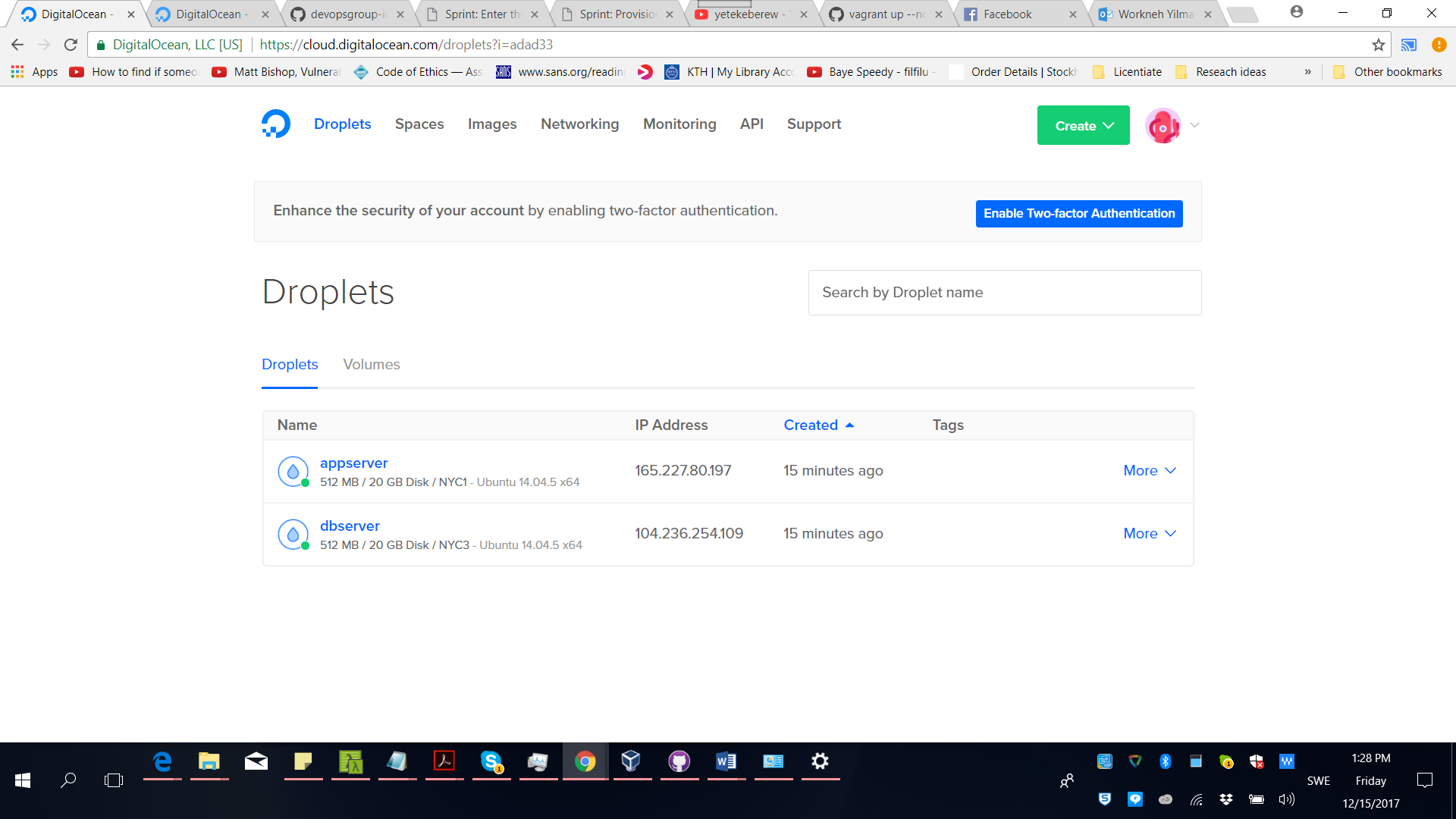
**Multi-machine provisioning**

I modified the vagrant file to include two machines and run vagrant up as illustrated below then two machines were provisioned on digitalocean



**Cloud provisioning and deployment using puppet**

I created a manifest file – default.pp



# Overall reflection

The syntax of Puppet and Vagrant is not like the programing languages I used to do, so there are times when I figured to find tricks to solve some of the problems I encountered. I plan to read more about puppet in the future. The other challenge was, wrting shell scripts in windows or copying and pasting script files into notepad and saving it with .sh extension doesn’t always work. In order to deal with such problem there were time when I copied shell scripts as it is and edited them.

The key to succeeding in configuring cloud provisioning is documentation. One need to know where to search for an issue, once one found what he wanted it is always good to bookmark and also to maintain documentation and best practices for future use. Otherwise, working with provisioning will be unpleasant thing to persue.

The security implications of using images of OSs available in the cloud are:

* Images are available to everyone so black hats can exploit the vulnerabilities of the images
* Developer or manager needs to install security patches and reconfigure the OS by for example changing passwords and changing some of the default ports.

**Reference**

<https://unix.stackexchange.com/questions/268140/applying-hosts-resource-in-puppet-to-update-etc-hosts>

<https://tickets.puppetlabs.com/browse/PUP-2289>

<https://github.com/devopsgroup-io/vagrant-digitalocean>

<https://www.digitalocean.com/community/tutorials/how-to-use-digitalocean-as-your-provider-in-vagrant-on-an-ubuntu-12-10-vps>

<https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys--2>

<https://sysadmincasts.com/episodes/42-crash-course-on-vagrant-revised>

<http://krisjordan.com/essays/goodbye-server-configuration-woes-hello-puppet>

<https://mediatemple.net/community/products/developer/204405534/install-nginx-on-ubuntu>

<https://jamesmcfadden.co.uk/using-vagrant-and-puppet-to-build-a-php-nginx-and-mysql-environment>

<https://www.vagrantup.com/docs/networking/forwarded_ports.html#auto_correct>