# Homework #1 – Ansible

# Matan Levi, Jacob Jasser, Sushma Maryada, Samina Qazi, Khurram Shahzad

# CMPE 272 – Enterprise Software Platforms

# 11/1/2017

Below are the main goals for our Ansible assignment:

* Install Ansible on any Linux Distribution
* Configure Ansible to deploy Apache web server
* Bring up web page with “Hello World”
* Uninstall Apache web server

**Introduction:** What is Ansible? Ansible is an open source platform that can help you with configuration management, application deployment, and task automation. It can also do IT orchestration, where you have to run tasks in sequence and create a chain of events which must happen on several different servers or devices. Unlike Puppet or Chef it doesn’t use an agent on the remote host. Instead Ansible uses SSH which is assumed to be installed on all the systems you want to manage. Also it’s written in Python which needs to be installed on the remote host. This means that you don’t have to setup a client server environment before using Ansible, you can just run it from any of your machines and from the client's point of view there is no knowledge of any Ansible server. The best part about Ansible is that it is available for free and runs on Linux, Mac or BSD.

Below are the high-level steps our team took to accomplish using Ansible to configure and deploy a webserver using Apache:

1. **Decide which distribution system to use for installing Ansible.** Our team originally approached install Ansible through various methods. At first we tried to setup Ansible through Mac, but noticed there isn’t as much support/resources available to perform the tasks we were looking to accomplished. We eventually pursued Ubuntu 16.04.3 LTS as our Linux Distribution to install Ansible. Our method of approach was using VirtualBox (a cross-platform virtualization application) to deploy two Ubuntu Virtual Machines on a Windows machine.
2. Here is the website we used to download Ubuntu 16.04.3 LTS (Long Term Support) ISO Image: <https://www.ubuntu.com/download/desktop>. The reason we used LTS is because it is the most stable release for Ubuntu.
3. Here is the website where we downloaded VirtualBox: <https://www.virtualbox.org/wiki/Downloads>

Once we installed VirtualBox on our Windows Machine, we went through the motions of creating two Virtual Machines. One was our controller (where Ansible is installed) and the other was our remote/target machine where we would install/configure/orchestrate various pieces of software packages:

* Creating Virtual Machine by selecting our Virtual Machine name, Type of Virtual Machine (Linux), and Version (Ubuntu 64-bit).
* Selecting Memory Size of Virtual Machine – in our case, each VM has 2048MB of memory.
* Create a Virtual Hard Disk and select Virtual Disk Image.
* Use “Dynamically Allocated” hard disk file which will only use space on physically hard disk as it fills up.
* Selecting the size of the Virtual Hard Disk (20.00 GB) since we wanted to be on the same side with our package installations.
* We then went into settings of each Virtual Machine → Network Tab → Under “Adapter 1” Tab → Select “Bridged Adapter”. Under the Bridged Adapter, your virtual machines behave as any other computer on the network where the hosting system resides; it bridges the virtual and physical networks. The outside world can directly communicate with the guest machine. We wanted this option since we would have an active network to download packages and software on our Ubuntu VM’s.
* We then select “Start” in VirtualBox for each Virtual Machine and go through the normal process of installing Ubuntu 16.04.3 LTS.

After installing both Virtual Machines, we then checked both Virtual Machine’s network to make sure each one is connected to the internet. The command we used in Terminal was “ifconfig” to observer eth0. We confirmed that each VM had an IP Address (this is important for our hosts file).

1. **Install Ansible on Ubuntu 16.04.3 LTS Controller Virtual Machine.** Our team figured out early on that there is a correct process to install Ansible. Here are the proper steps to install Ansible on our controller system.
2. Sudo apt-get update: This command updates the package lists for upgrades for packages that need upgrading, as well as new packages that have just come to the repositories.
3. Sudo apt-get install software-properties-common: This was called “python-software-properties”
4. Sudo apt-add-repository ppa:ansible/ansible Adding the Personal Package Archive.
5. Sudo apt-get update:
6. Sudo apt-get install ansible: Installs Ubuntu onto VirtualMachine.
7. FIGURE 1.1 displays that we typed “ansible –version” to ensure we have the latest version to play with.

FIGURE 1.1:Using command line to output Ansible Version.

1. **Install SSH Keys on both Controller Machine and Target Machine.** Ansible primarily communicates with client computers through SSH. While it certainly has the ability to handle password-based SSH authentication, SSH keys help keep things simple. We will create an SSH key pair on our Ubuntu Virtual Machines to authenticate with the hosts that it will administer. Below are the steps to properly install SSH Keys on both Virtual Machines:
2. Open Terminal and type “ssh-keygen” on both Virtual Machine instances. This is a standard component of the Secure Shell (SSH) protocol suite found on Unix and Unix-like computer systems used to establish secure shell sessions between remote computers over insecure networks, through the use of various cryptographic techniques. A public and private key will be generated.
3. The public and private keys are automatically created and stored in the “.ssh/” directory on the system. Using the command “cd .ssh/” to change directories, and “cat id\_rsa.pub” (FIGURE 1.2) will allow us to view the content of that file. In this example, we will be able to view the public key generated. To view the private key generated, we type “cat id\_rsa” in the Terminal (FIGURE 1.3).
4. We then use the command “ssh-copy-id username@host” in Terminal to transfer public keys to each Virtual Machine. This will allow us to the public key to Lock a message, and private key to unlock a message.

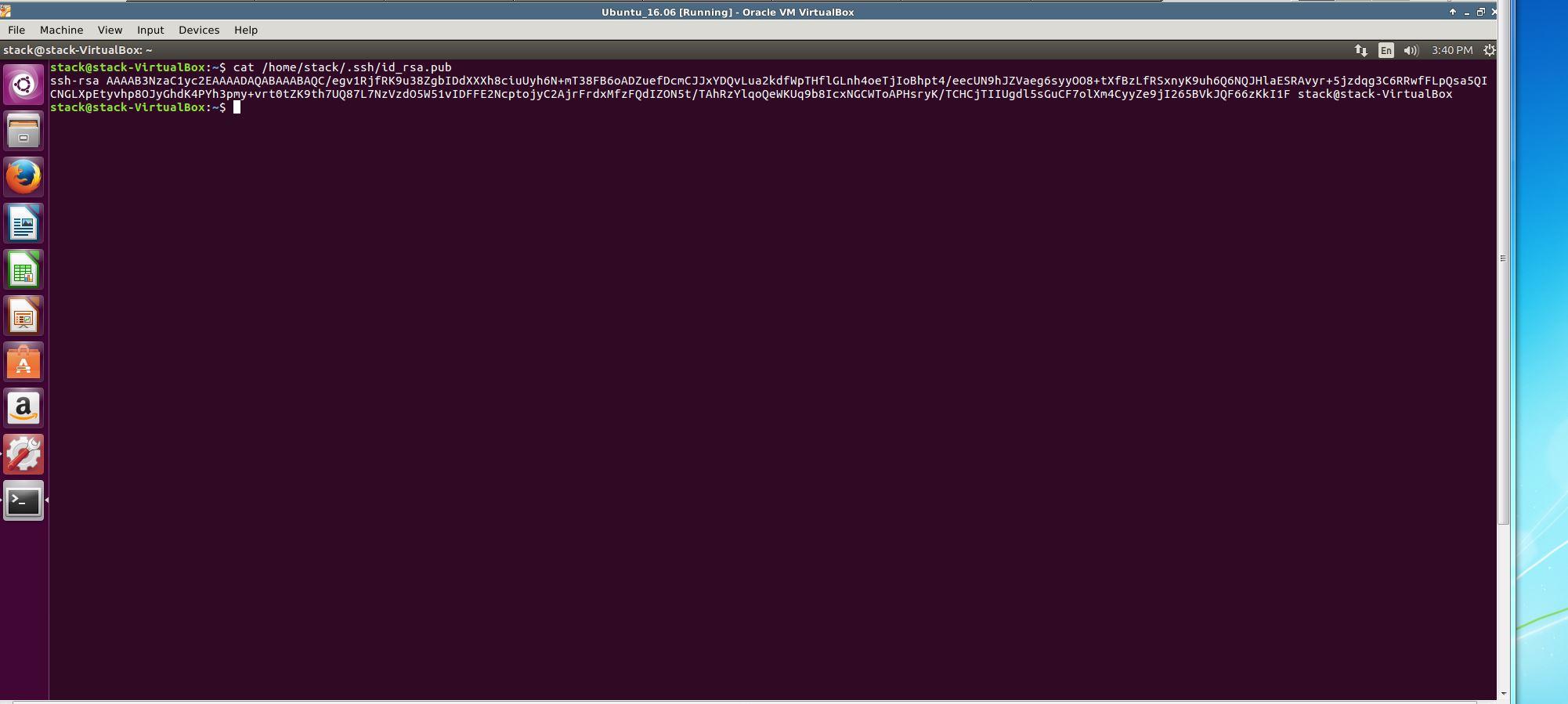
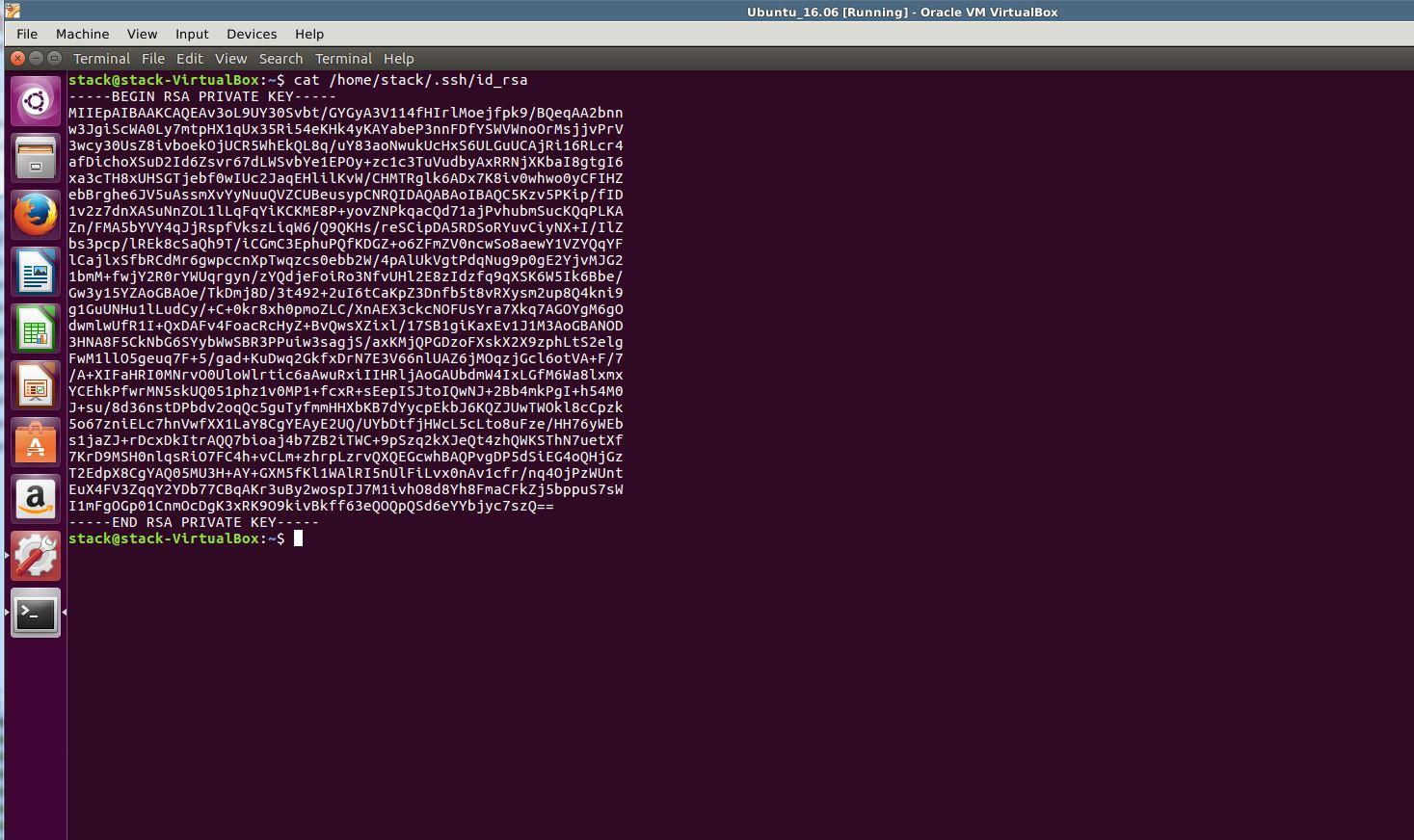


FIGURE 1.2: Using “cat” command to display generated Public Key.

FIGURE 1.3: Using “cat” command to display generated Public Key.

1. **Manage servers using Ansibles Host file.** Ansible has a default inventory file used to define which servers it will be managing. The default location can be referenced and managed in the “/etc/ansible/hosts” directory on the Controller Virtual Machine. This step allows us to run Ansible Tasks directly on the target server. The benefit here is that we can control multiple servers simultaneously, and not setup multiple servers or virtual machines. Here are the steps we took to accomplish this task:
2. We copied and moved the default hosts file so we can reference it later with the command “sudo mv /etc/ansible/hosts /etc/ansible/hosts.orig”.
3. After moving the original inventory file, we created a new “/etc/ansible/hosts” file with the command “sudo nano hosts”.
4. Since we know the IP Address of the Target Virtual Machine to deploy our Apache Webserver, we will define our group called “[web]” with the IP Address of “10.240.160.25”.
5. We use “Control + O” to write to the “hosts” file, and “Control + X” to exit the Nano terminal.
6. Using the command “cat /etc/ansible/hosts” we are able to view the contents of our newly formed hosts file.
7. **Create index.html file for “Hello World”.**
8. **Create Ansible Playbooks for installing Apache webserver, and uninstalling Apache Webserver.**   
   Playbooks are Ansible’s configuration, deployment, and orchestration language. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process. At a basic level, playbooks can be used to manage configurations of and deployments to remote machines. At a more advanced level, they can sequence multi-tier rollouts involving rolling updates, and can delegate actions to other hosts, interacting with monitoring servers and load balancers along the way. Playbooks are written in YAML, which simplistically could be viewed as XML but human readable.

Before utilizing Ansible Playbooks, we will ensure that we can ping the target server from our hosts file using the command “ansible –m ping all”. This confirms whether or not our server is online. The “-m” switch allows us to use the “ping” module which simply runs the ping command and returns the result. “All” ensures we are pinging all servers in our host file. FIGURE 1.4 displays the successful ping to the target Virtual Machine.

FIGURE 1.4: Successful ping to Target Virtual Machine using Ansible command “ansible –m ping all”.

**--------------------------------------------------------------------------------------------------------------------------------------------------**

**Below is the code we used to install and configure apache (FIGURE 1.5). Comments to describe each action will be in Orange font:**

**--- #This is syntax used to start an Ansible Playbook**

**- hosts: web #Tasks will be performed/installed onto the servers configured in “web” group in hosts file.**

**sudo: yes #Run ansible playbook as superuser.**

**vars: #Creates a variable to be used in our index.html file.**

**MyMessage: "HELLO WORLD" #Outputs HELLO WORLD when invoking our webpage through the browser.**

**tasks: #Playbooks exist to run tasks. Tasks combine an action (a module and its arguments) with a name and optionally some other keywords (like looping directives).**

**- name: install apache2 #This installs Apache2 webserver.**

**apt: name=apache2 update\_cache=yes state=latest Install latest Apache2 Webserver”**

**- name: index.html copy #Lists the action performed when running Ansible Playbook.**

**template: src=index.html.j2 dest=/var/www/html/index.html #Copies index.html file to different destination; this is the location used**

**- name: images.JPG copy #Lists the action performed when running Ansible Playbook.**

**template: src=ansible.JPG dest=/var/www/html #Put an image file into our html file.**

**- name: apache2 listen on port 80 #Lists the action performed when running Ansible Playbook.**

**lineinfile: dest=/etc/apache2/ports.conf regexp="^Listen 80" line="Listen 80" state=present**

**- name: apache2 virtualhost on port 80 #Lists the action performed when running Ansible Playbook.**

**lineinfile: dest=/etc/apache2/sites-available/000-default.conf regexp="^<VirtualHost \\*:80>" line="<VirtualHost \*:80>" state=present #By default Apache listens on port 80 for all HTTP traffic.**

**#With the default Apache configuration on Ubuntu 16.04 x64, there are two files that need to be #updated: To do this, we can use the lineinfile module. This module is incredibly powerful and through #the use of it's many different configuration options, it allows you to perform all sorts of changes to an #existing file on the host. For this example, we will use the following options:**

**#dest -- The file to be updated as part of the command.**

**#regexp -- Regular Expression to be used to match an existing line to be replaced.**

**#line -- The line to be inserted into the file, either replacing the regexp line or as a new line on the end.**

**#state -- Either present or absent.**

**- name: enabled mod\_rewrite #Lists the action performed when running Ansible Playbook.**

**apache2\_module: name=rewrite state=present #Using the apache2\_module, we enable #mod\_rewrite. Now that Apache is installed, we need to enable a module to be used by Apache.**

**#Let us make sure that the mod\_rewrite module is enabled for Apache. Via SSH, this can be done easily #by using a2enmod and restarting Apache. However, we can also do it very easily with Ansible using the #apache2\_module module. The apache2\_module module takes two options:**

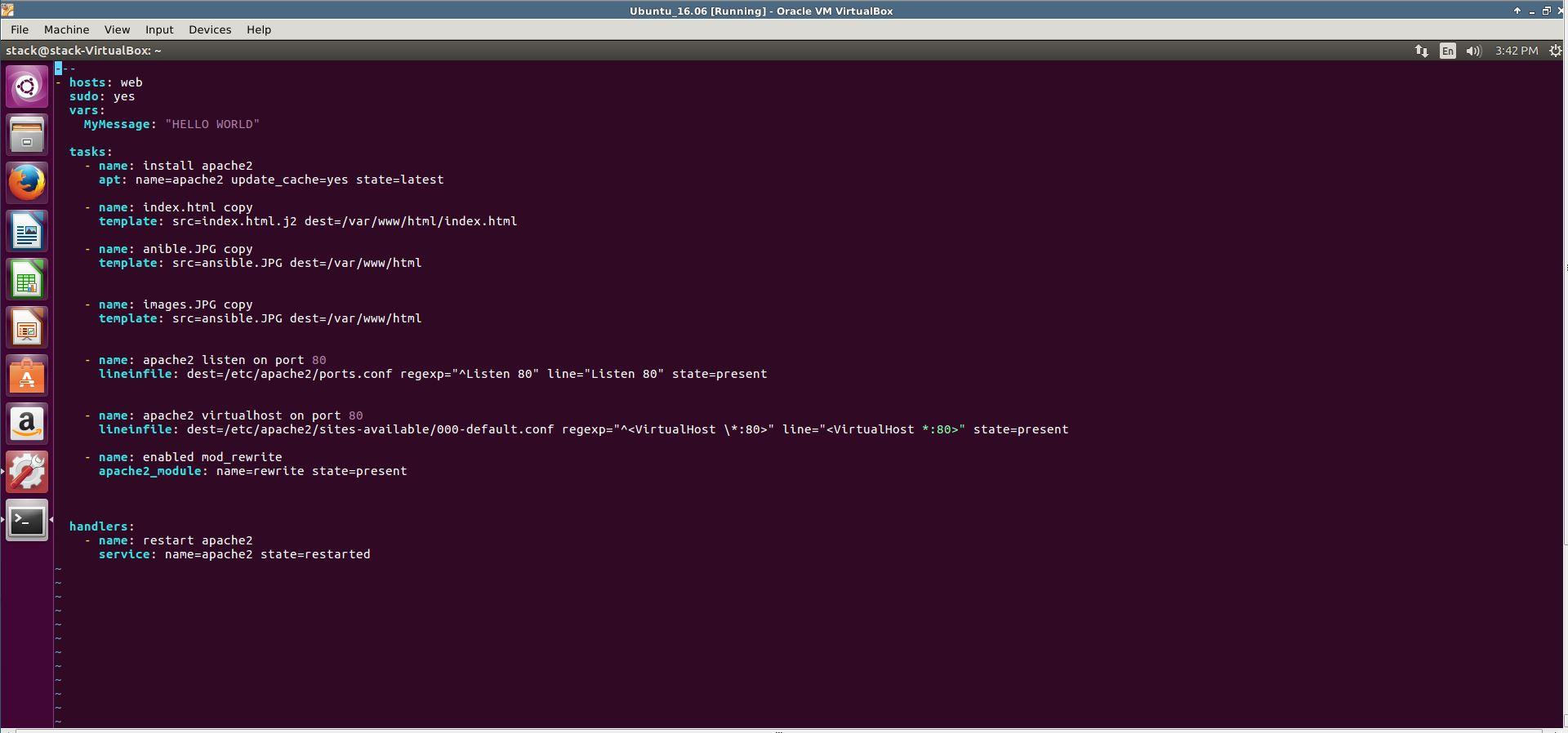
**#name -- The name of the module to enable, such as rewrite.**

**#state -- Either present or absent, depending on if the module needs to be enabled or disabled.**

**handlers: #Handlers are just like regular tasks in an Ansible playbook (see Tasks) but are only run if the #Task contains a notify directive and also indicates that it changed something.**

**- name: restart apache2 #Lists the action performed when running Ansible Playbook.**

**service: name=apache2 state=restarted #** **This particular handler uses the Service module, which can #start, stop, restart, reload (and so on) system services. Here we simply tell Ansible that we want #apache2 to be started.**

FIGURE 1.5: YML Code developed to install Apache2 webserver.

**--------------------------------------------------------------------------------------------------------------------------------------------------**

**Below is the code we used to uninstall apache (FIGURE 1.6). Comments to describe each action will be in Orange font:**

**--- #This is syntax used to start an Ansible Playbook**

**- hosts: web #Tasks will be performed/installed onto the servers configured in “web” group in hosts file.**

**sudo: yes #Run ansible playbook as superuser.**

**tasks: #Playbooks exist to run tasks. Tasks combine an action (a module and its arguments) with a name and optionally some other keywords (like looping directives).**

**- name: remove apache2 #Lists the action performed when running Ansible Playbook.**

**apt: name=apache2 update\_cache=yes state=absent purge=yes #Uses the apt module to uninstall Apache by purging it completely.**

**- name: delete our Hello World website folder #Lists the action performed when running Ansible Playbook.**

**file: #Lists the file path where we have Hello World website folder.**

**path: /var/www/ #Sets the path where to delete Hello World website filder.**

**state: absent #Sets the desired state of the file to absent.**

**- name: delete our virtual host file #Lists the action performed when running Ansible Playbook.**

**lineinfile: dest=/etc/apache2/sites-available/000-default.conf regexp="^<VirtualHost \\*:80>" line="<VirtualHost \*:80>" state=absent #Deletes VirtualHost File using “absent” argument.**

**- name: delete apache www folder #Lists the action performed when running Ansible Playbook.**

**file: #Lists the file path where we have Hello World website folder.**

**path: /var/www #Sets the path where to delete Hello World website filder.**

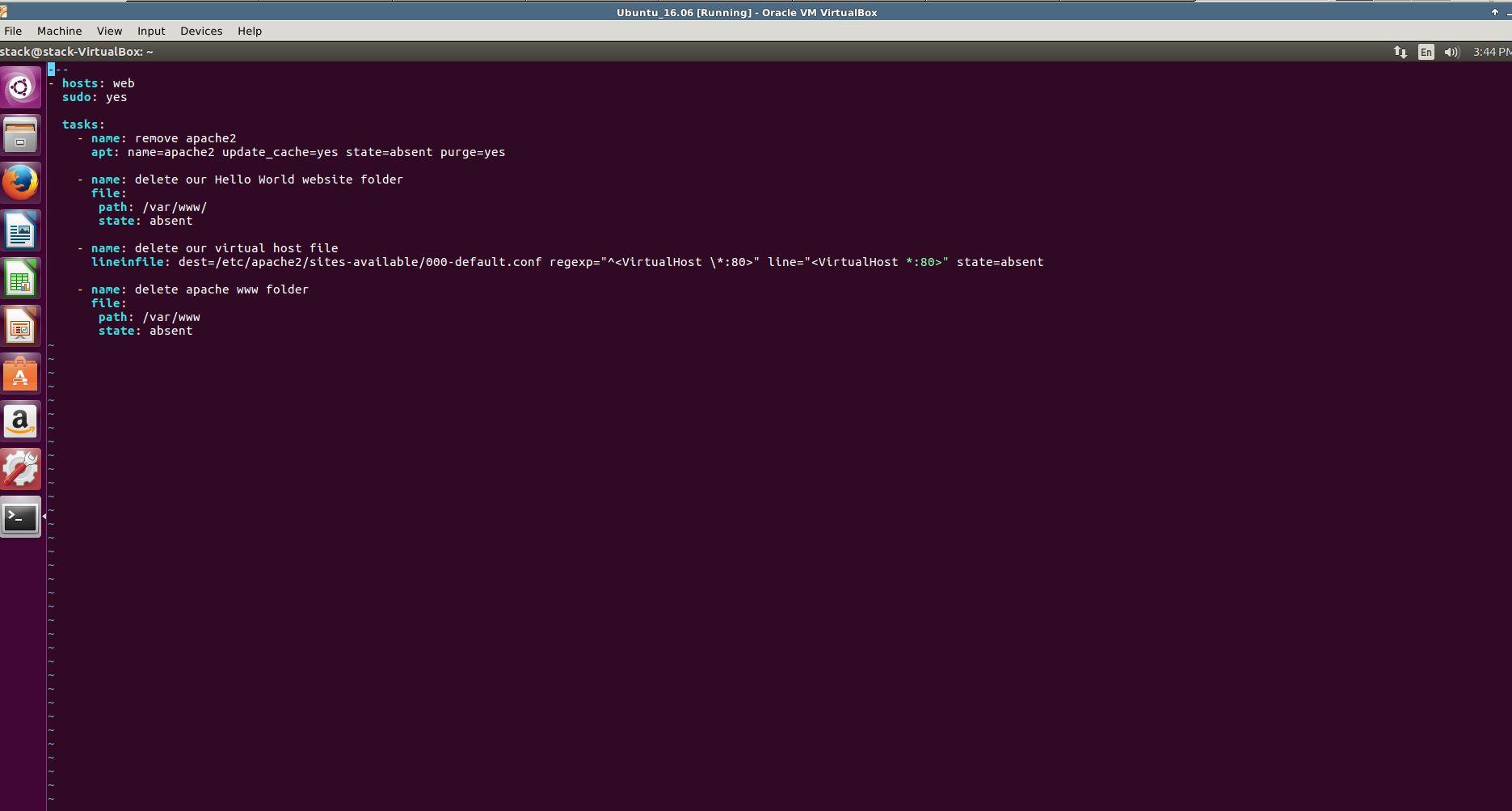
**state: absent #Sets the desired state of the file to absent.**

FIGURE 1.6: YML Code developed to uninstall Apache2 webserver.

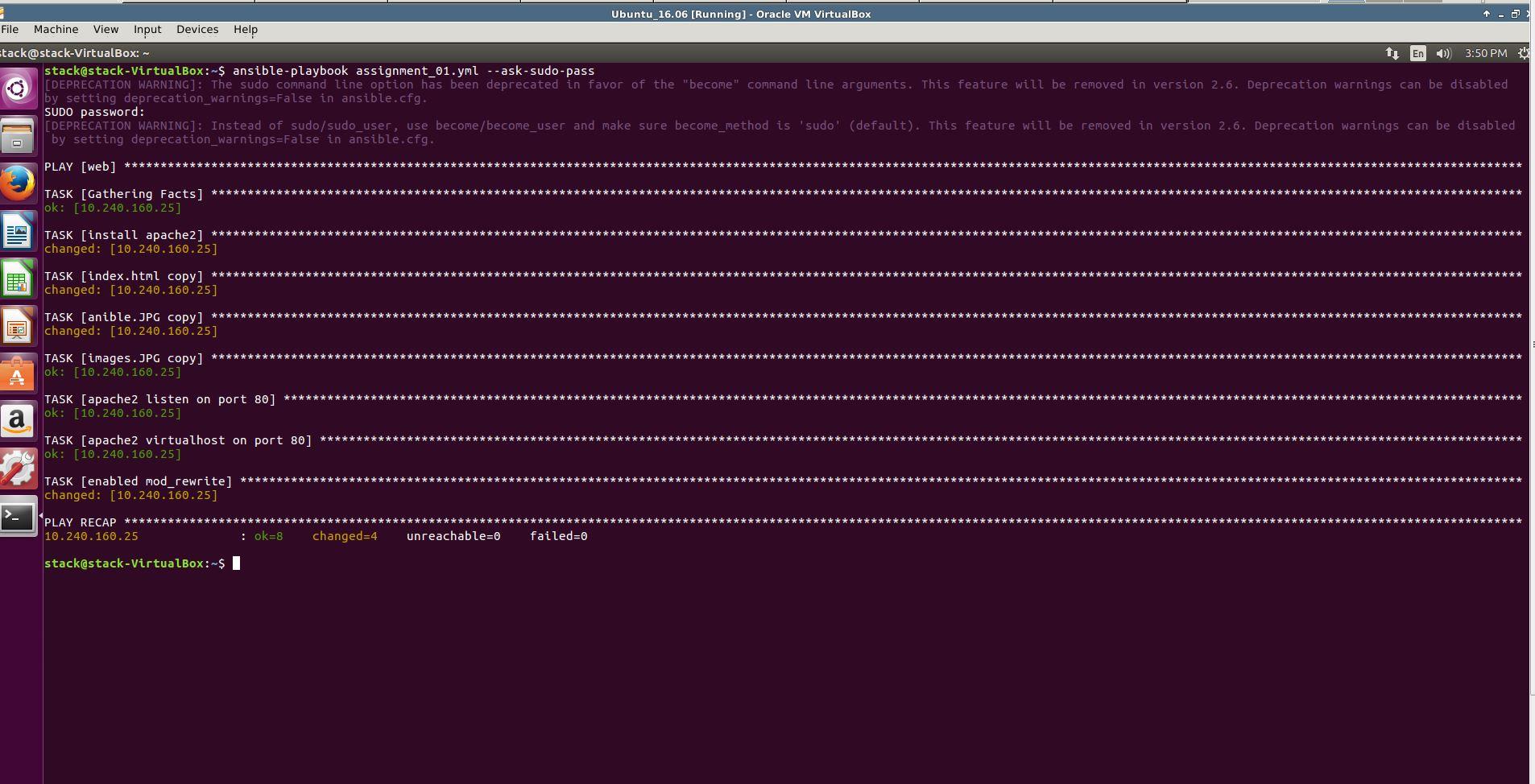
**--------------------------------------------------------------------------------------------------------------------------------------------------**

Each play contains a list of tasks. Tasks are executed in order, one at a time, against all machines matched by the host pattern, before moving on to the next task. It is important to understand that, within a play, all hosts are going to get the same task directives. It is the purpose of a play to map a selection of hosts to tasks. The goal of each task is to execute a module, with very specific arguments. Variables, as mentioned above, can be used in arguments to modules. In our example, Apache will be installed onto the “web” host which was defined in our hosts file.

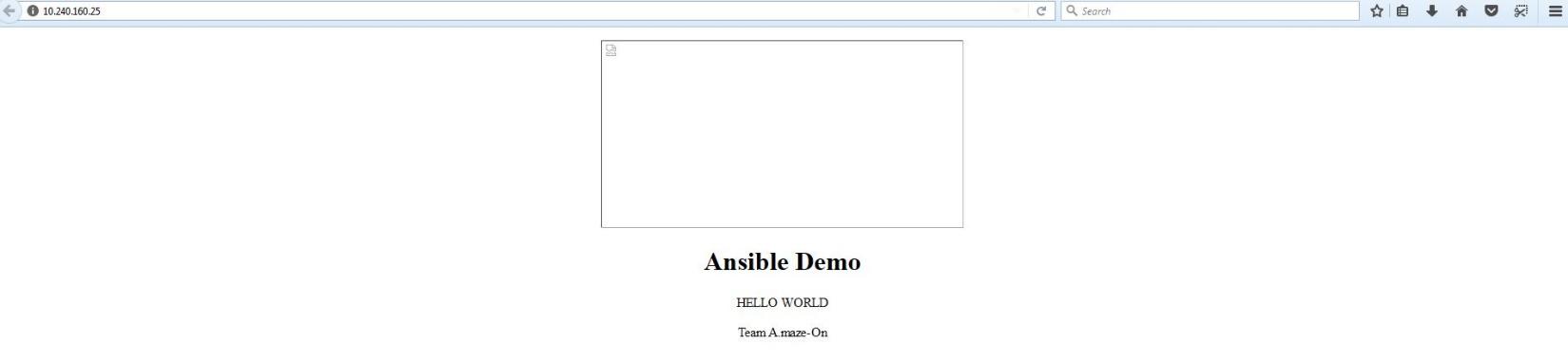
Every task should have a name, which is included in the output from running the playbook. This is human readable output, and so it is useful to provide good descriptions of each task step. If the name is not provided though, the string fed to ‘action’ will be used for output. Sudo is to run the ansible playbook as a superuser, allowing additional privileges. We then added a variable called “HELLO WORLD” which will be used tin our index.html file for launching our webpage.

The basic format for declaring a Task is to use the “module: option” format. Each option will have a “key=value” argument to describe the action performed. For instance to install Apache, we use “apt: name=apache2 update\_cache=yes state=latest”. This invokes the apt module which manages apt packages in Debian/Ubuntu. The “name” key is used to describe the package name being installed, “update\_cache” key runs “apt-get update” before the operation is performed, and “state” key indicates the desired package state (in our case latest version installed). The important thing to note is that each module contains options which can be run to perform certain actions.

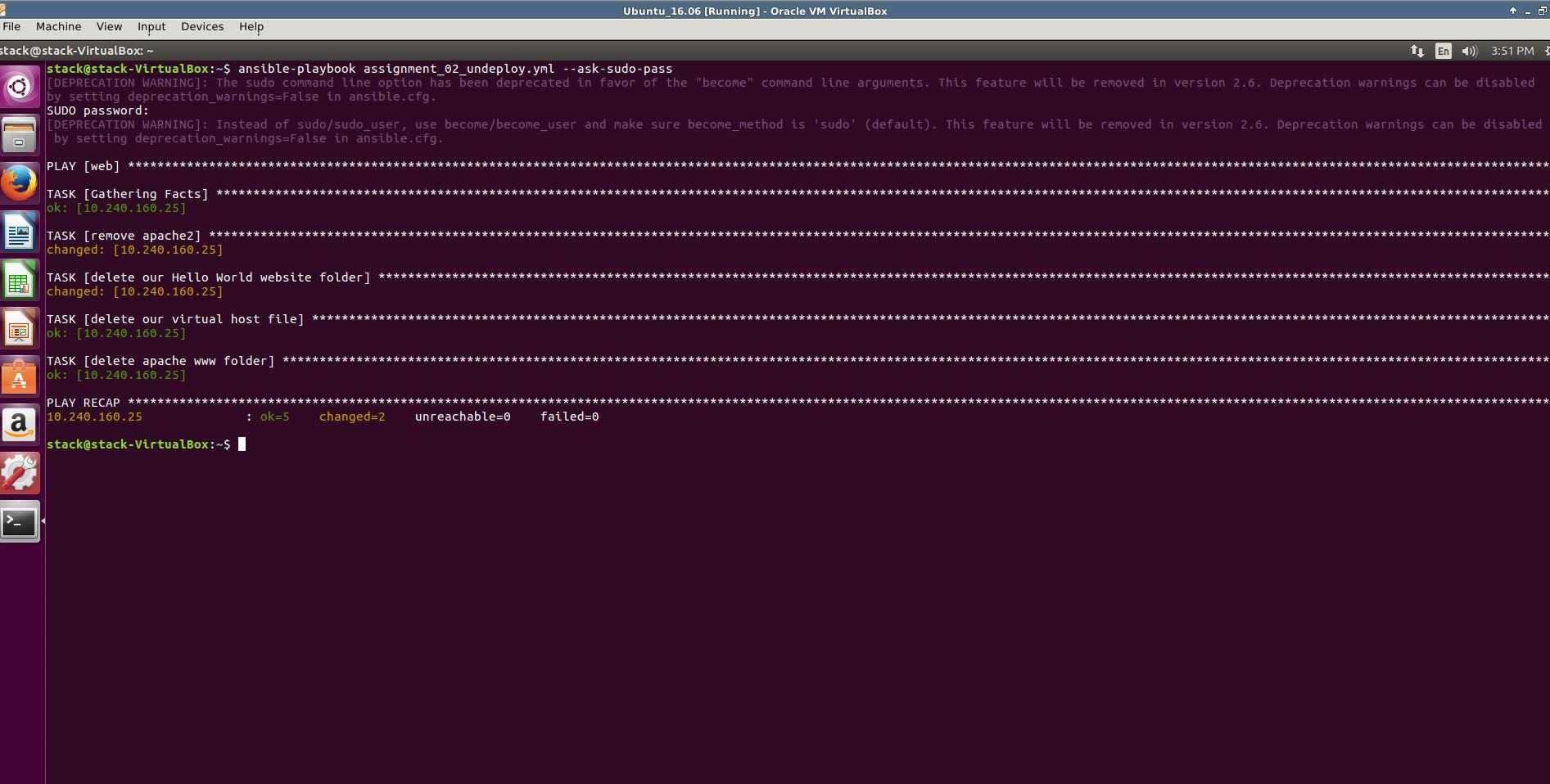
1. **Run Ansible Playbook to install Apache by using the command “ansible-playbook assignment\_01.yml –ask-sudo-pass”.** “Ansible playbook” is a command used to execute our playbooks, “assignment\_01.yml” is the name of our playbook, and “—ask-sudo-pass” ensures only the superuser has access to run this playbook. In FIGURE 1.7 we can see that the playbook runs successfully to install Apache onto our target Virtual Machine.

FIGURE 1.7: Running Ansible Playbook to install Apache2 webserver on Target Machine.

1. **Type in IP Address of our Target System (10.240.160.25) into browser to ensure “HELLO WORLD” html works as intended (FIGURE 1.8).** As we can see, we have successfully orchestrated apache with Ansible to output HELLO WORLD, our team name, and Ansible Demo.

FIGURE 1.8: Successful deployment of Hello World webpage.

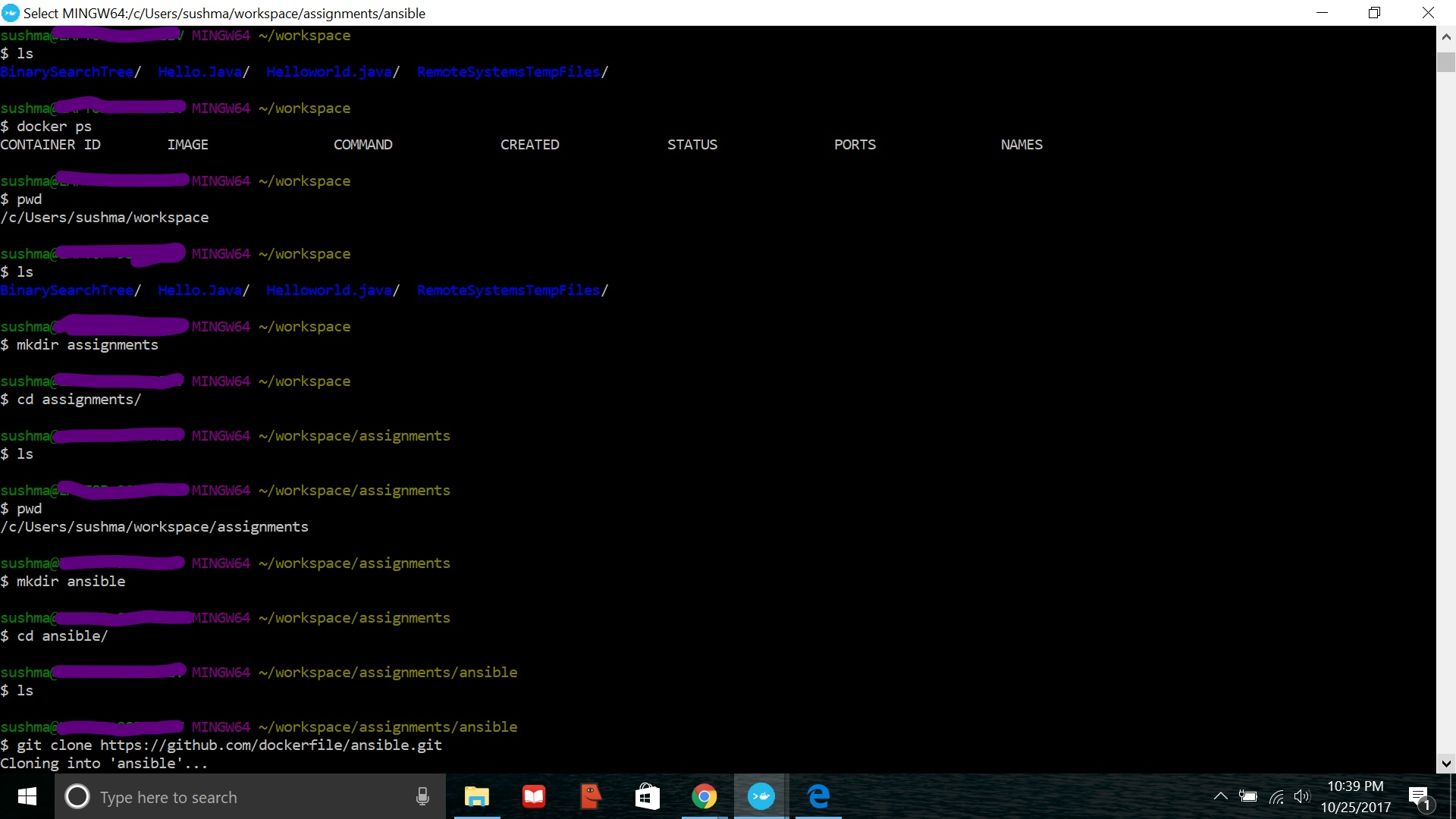
1. **Last but not least, uninstall Apache from Target Virtual Machine using Ansible Controller Virtual Machine.** In command line we type “ansible playbook assignment\_02\_undeploy.yml –ask-sudo-pass”. As mentioned before “Ansible playbook” is a command used to execute our playbooks, “assignment\_01.yml” is the name of our playbook, and “—ask-sudo-pass” ensures only the superuser has access to run this playbook. FIGURE 1.9 shows that we have successfully made changes to removing apache2 and deleting our Hello World website folder.

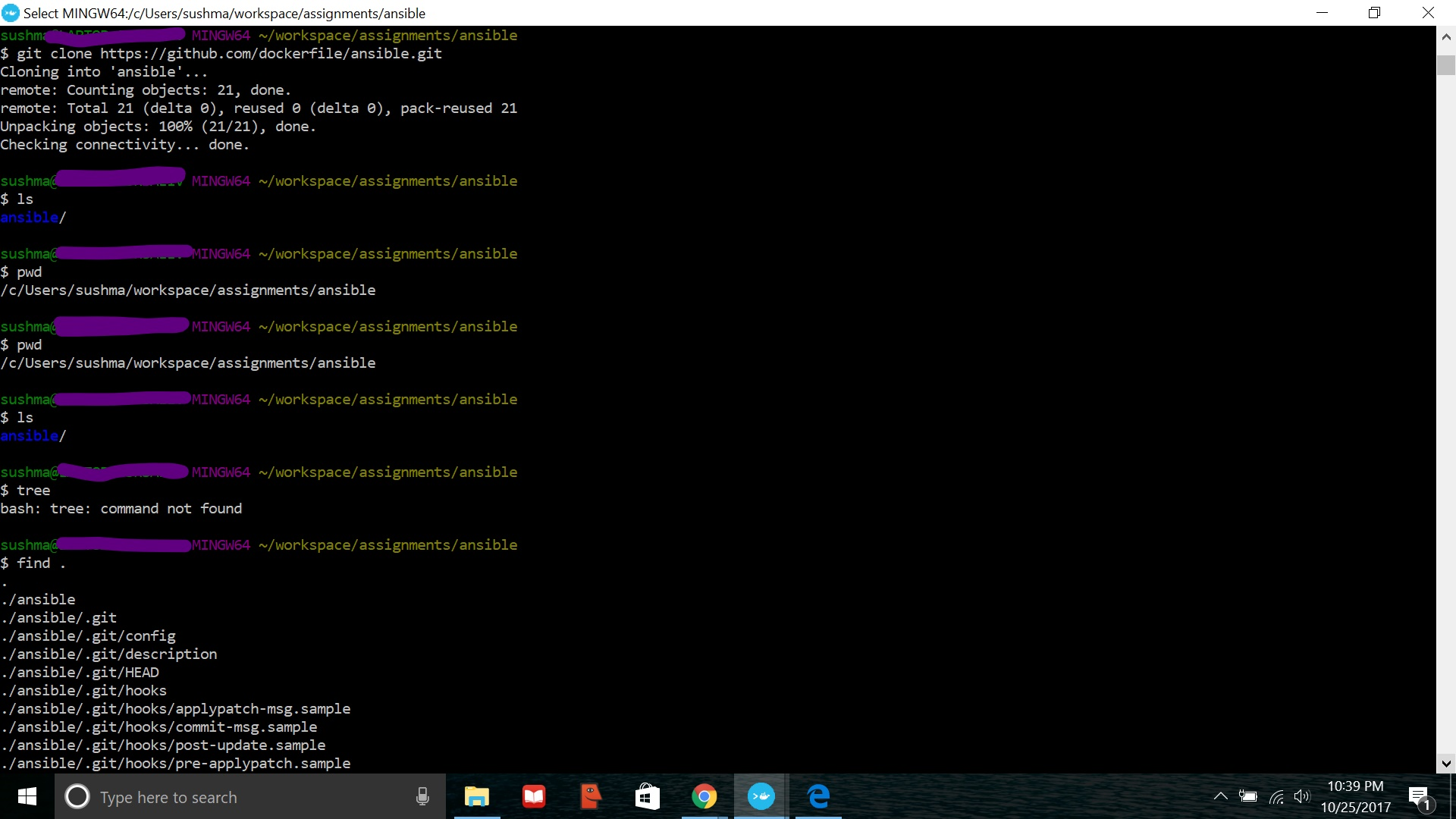
FIGURE 1.9: Successfully removed Apache2 and Hello World website folder.

**Conclusion:**

Ansible is a very powerful service orchestration and IT Automation Tool, and is capable of providing invaluable power to System Administrators and Corporations.

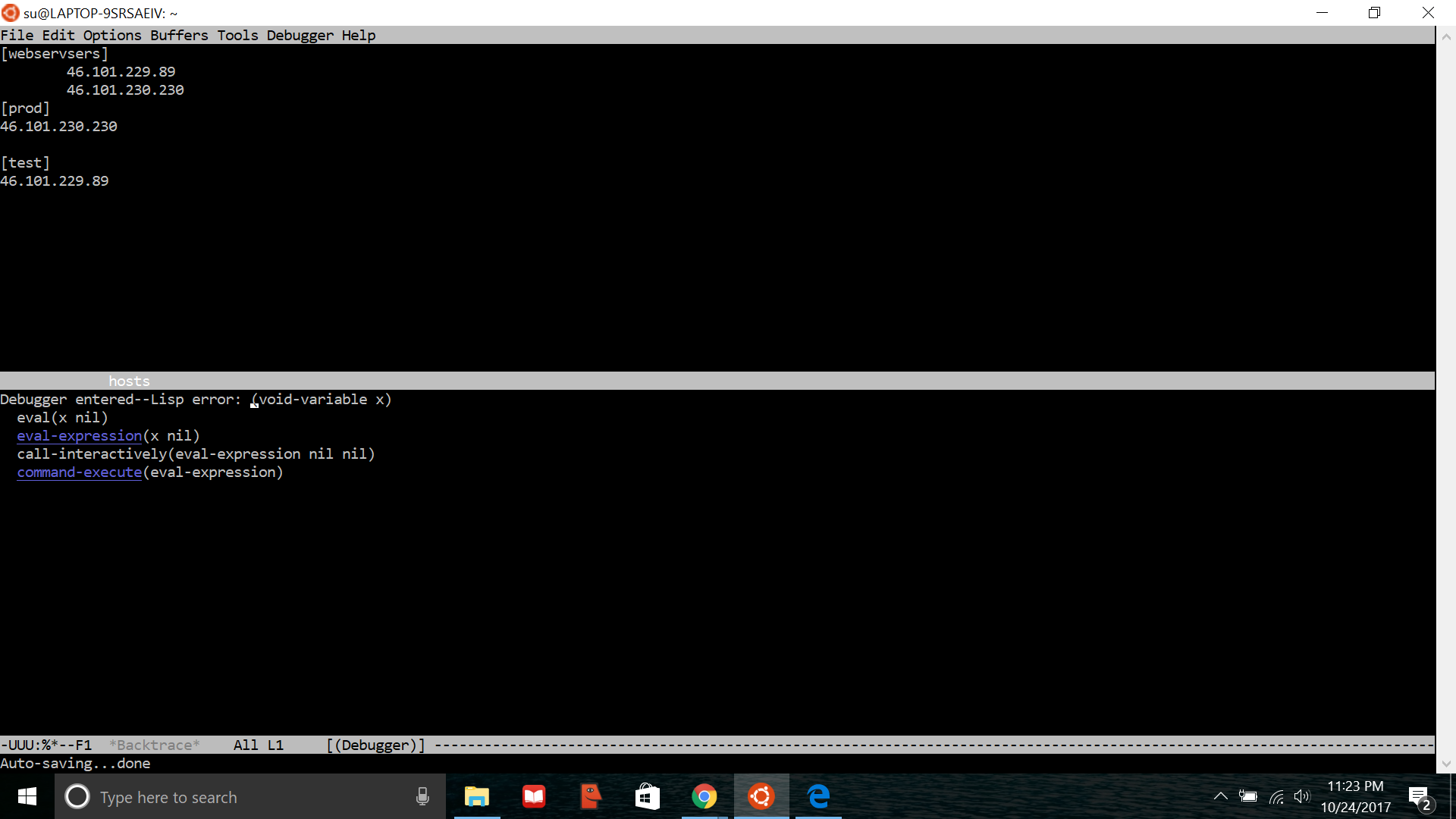
And also tried with docker container by cloning ansible to docker machine using git clone.

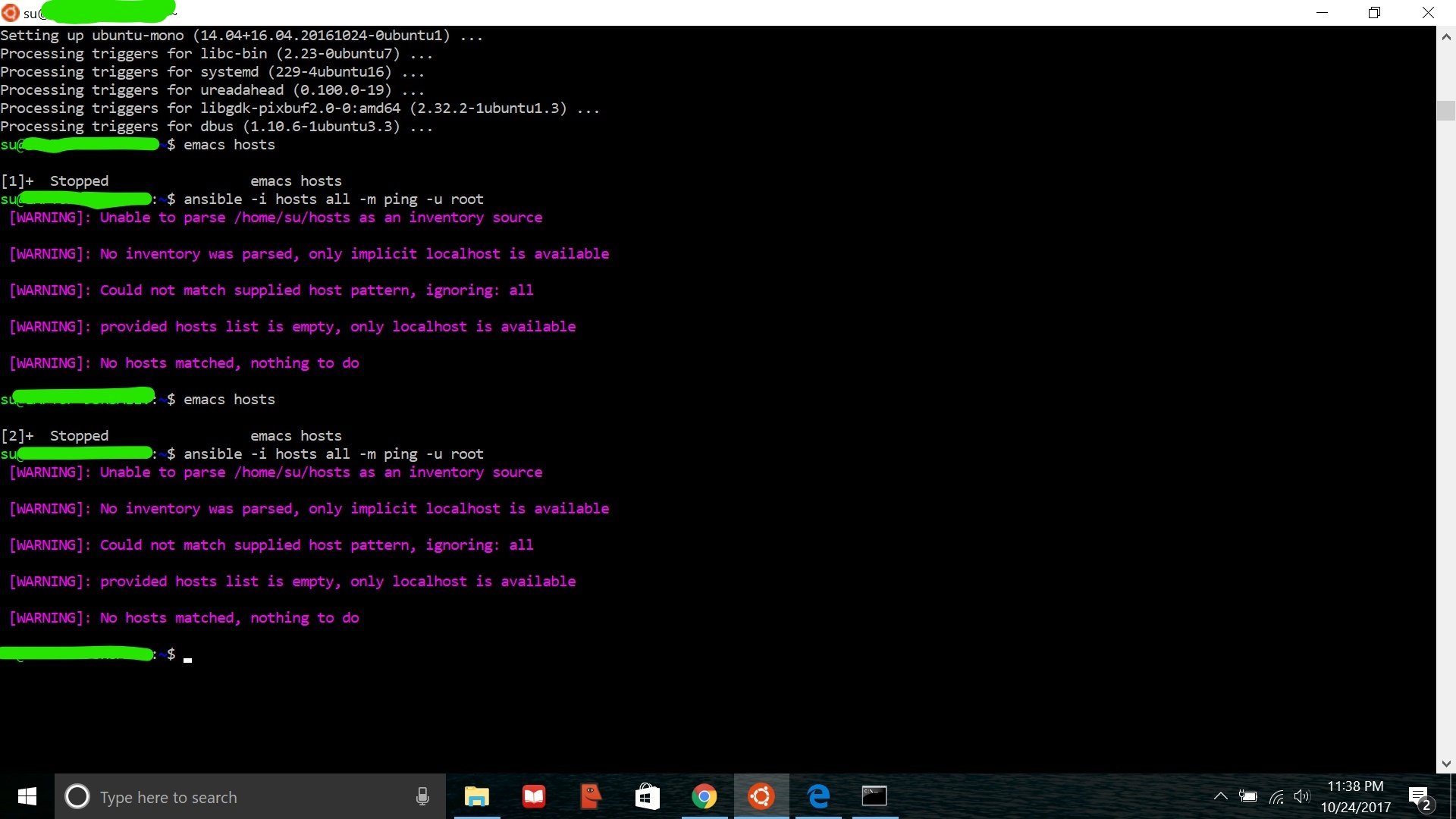




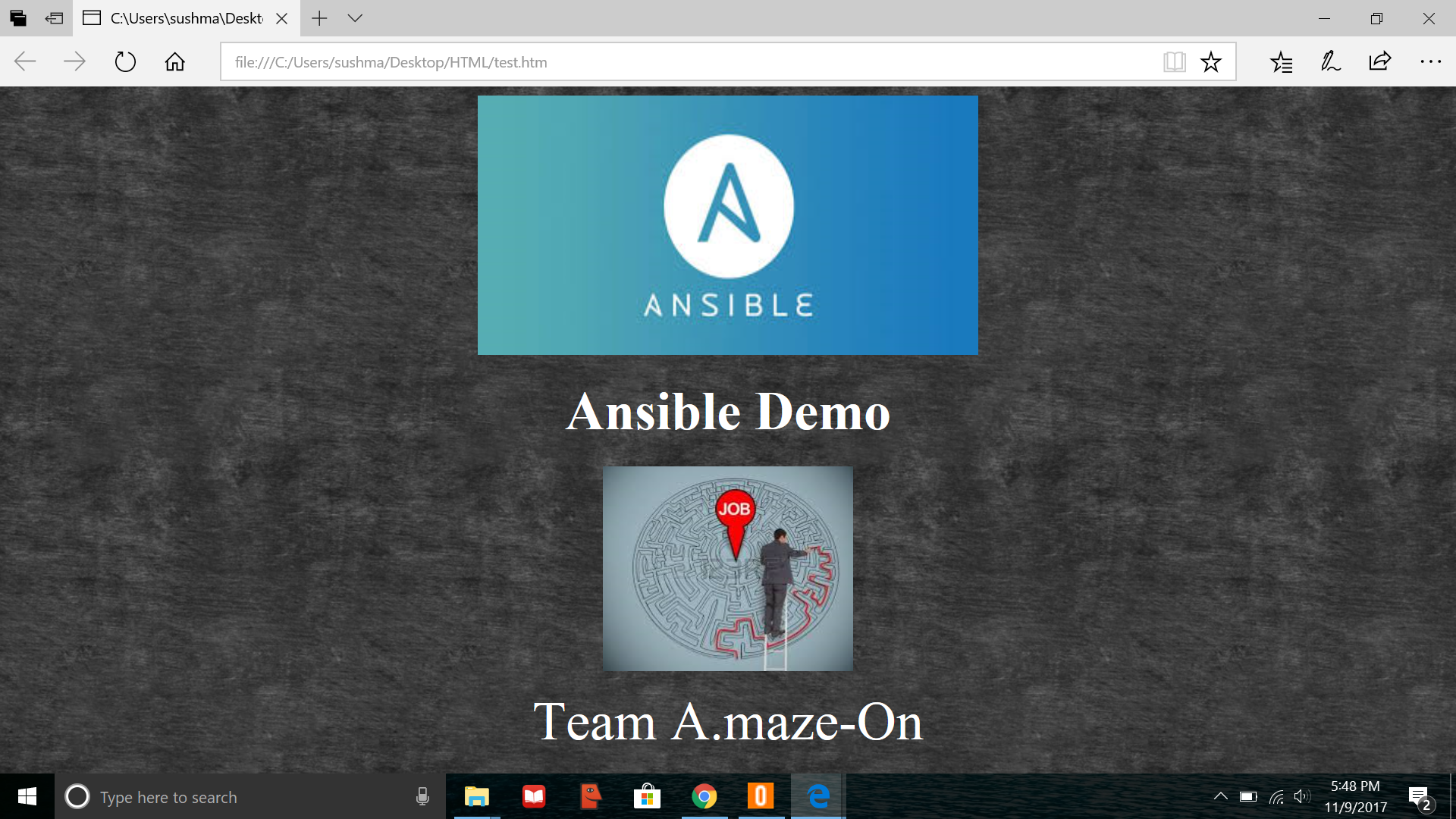
**Difficulties Faced :**

While trying with windows machine using BASH on ubuntu on windows, *Vagrant Up* didn’t work properly, Unable to ping.





And also we tried to get this output:



It Did not work because of the images, and when we tried with other images it worked.

