INSTRUCTIONS:

Create a professional document showing the completion of each task. Be ready to demonstrate your understanding of each task. Your document must include:

- Title page
- Updateable Table of Contents
- Document introduction
- Section introductions or description, each section must be clearly identified
- Graphics or screenshots MUST include a title with a short description
- A references including a reference page at the end showing where your information was obtained

INTRODUCTION

For the final project, we will be looking into automating aspects of VM deployment. There are entire roles related to the automation and deployment of applications. A common task is creating and provisioning a VM for testing. Provisioning is the process of getting the VM to a functional state. The goal is to be able to rapidly deploy a VM that has the needed configuration. We are going to be use a technology called Vagrant [2] to create two testing VMs.

SUPPLEMENTAL CONTENT AND HELPFUL LINKS.

- Learning Vagrant. This course goes over the basics of Vagrant. All of the examples use VirtualBox but there are some useful concepts.
- Vagrant Docs
- Vagrant Download
- VMWare Utility
- Windows 10 Base Box

PART 1 – VAGRANT BASICS

This part covers the steps to get Vagrant up and running. We are going to install the necessary software and run our first box.

1. Download and install Vagrant on your host machine.
2. Vagrant on its own is there to simplify configuration and VM management. It is designed to be agnostic to the underlying hypervisor, but primarily supports VirtualBox. The hypervisor is reffered to as a provider. We are going to use VMWare as it is already set up on our systems. Download the VMWare utility and install the required plug-in using the following command:

   ```
   vagrant plugin install vagrant-vmware-desktop
   ```

3. Vagrant VMs are based on images known as boxes. Boxes can be created and stored locally, but Hashicorp, the developers of Vagrant, maintain a public repository of boxes. Each box supports at least one or more providers. We are going to use a Windows 10 system as our base.
   (a) Create a new directory for your VM.
   (b) Navigate there using Powershell.

(c) Run the following command:

```
vagrant init gusztavvargadr/windows-10 --box-version 2202.0.2409
```

4. The above command create a "Vagrantfile" configured to use a Windows 10 box. Open the file in a text editor to take a look. In this file we can apply configurations to our VM. For now, the only uncommented configurations are to use the base box. Look through the different configuration options that are commented out. Add the following lines to the Vagrant file [1]:

```
config.vm.provider "vmware_desktop" do |vmw|
    vmw.gui = true
end
```

These are provider specific configurations that tell Vagrant we want the machine's GUI to start when we run it.

5. In your Powershell window type the following command to bring up your machine:

```
vagrant up
```

6. Include a screenshot of your Powershell window and VM running in VMWare to show your completion of this part.

PART 2 – COMPANYINC

Now that we have our system running, we can add additional configurations. We want to modify our system to match some of the CompanyInc configurations we made over the semester.

1. Lets start by bringing our machine down. In the host Powershell type:

```
vagrant halt
```

2. The settings for VMWare machines are defined in their VMX files. We can configure this file from within Vagrant. In the provider specific section of the VagrantFile, add the following lines:

```
vmw.vmx["numvcpus"] = "1"
vmw.vmx["suspend.disabled"] = "TRUE"
```

Compare these setting with the VMX file you have been using up until now. Notice the keys are in the square brackets and the values are assigned.

3. To provision the machine and make changes to the guest OS, Vagrant can be configured to run scripts. Adding a line like the one below to our Vagrantfile will cause a script to be run as part of the provisioning process. Note that the script will run with elevated privileges. To test this feature, try adding your `NewObject.ps1` script from A5. You'll need to place the script on your host system and provide the path to the script.

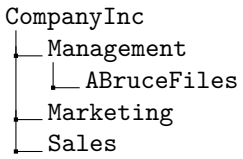
```
config.vm.provision "shell", path: "path/to/myscript.ps1", privileged: true
```

Because we've already started our machine once, we have to tell it to explicitly provision the VM. Type the following line to bring the VM up:

```
vagrant up --provision
```

[1] Notice how these are similar to the provider specific configurations for VirtualBox.

4. Adjust your configuration and provisioning to match the following objectives. Your VM should use Vagrant and a provisioning script to achieve these settings on startup.
   - Memory 8GB (or less if host system demands)
   - 1 CPU
   - Add VMX settings from Brightspace.
   - Create your CompanyInc directory structure on your C drive.



   - Groups:
     - Management
     - Marketing
     - Sales
   - Users:
     - ABruce – Member of: Users, Management
     - FirstInitialLastname – Member of: Users, Administrator
5. Write a script called `CompanyInc_test.ps1` to demonstrate your provisioning was successful. Your script should show the CompanyInc folder structure and membership in the correct groups for your users.
6. Synced folders allow our VM to easily access files on our host machine. Windows does this through shared folders. You provide a path on your host sytem and a directory for your VM. Add a line like the following to your Vagrantfile to ensure your script is available on your VM:

```
config.vm.synced_folder "path/to/testscript/folder", "/test"
```

   Check the "VMware Shared Folders" shortcut on your VM desktop to make sure that your test script is available.
7. Include a screenshot of your test script running and copies of your Vagrantfile and scripts as text in your document. Be ready to demo bringing up your VM and running your test script.

PART 3 – DEMO APP

Create a second VM based on the same base box as your CompanyInc test VM. This new VM will act as a test machine for an application of your choosing. For example, you could create a test VM for a program you have written in another class, or to host a game server. You have flexibility provided it meets the following requirements:

- Installs some additional software automatically as part of the provisioning process.
- Uses an artifact shared through synced folders.

Define a specific configuration for your application VM. It should include any VM settings, required software. Make sure to include the version of the software you are using.

Include your configuration requirements, scripts and Vagrantfile in your document. You should also add a screenshot that shows your application running on your VM. Be ready to demonstrate creating a new VM with your Vagrantfile.

Hints: It may help to use a package manager to make installing your software easier, e.g., WinGet, NuGet or Chocolatey. Your deployment does not have to be that complex. It can be as simple as running helloworld.py or helloworld.js

PART 4 – REFLECTION

There is considerable evidence that reflection enhances learning and understanding [3, 1]. Include a brief one paragraph reflection on what you learned in the course. Try to answer the following questions:

- What? *What did you learn in the course.*
- So What? *Why is it important or relevant.*
- What now? *How might you use what you've learned in the future.*

MARKING

Value	**Task**
Part 1	
1	Vagrant VM running
1	GUI enabled
1	Windows 10 Enterprise
Part 2	
2	VMX settings
1	Folders
1	Users
1	Groups
3	ACLs
2	Provision Script
4	Test Script
2	Synced Folder
Part 3	
3	Requirements Defined
6	Provisioning
2	Synced Folder
Part 4	
2	What?
3	So What?
3	What now?
2	Document
40	**Total**

Table 1: Grading Scheme for Final Project

REFERENCES

[1] BOYD, E. M., AND FALES, A. W. Reflective learning: Key to learning from experience. *Journal of Humanistic Psychology 23*, 2 (1983), 99–117.

[2] HASHICORP. Vagrant. `https://developer.hashicorp.com/vagrant`. Accessed: 2024-11-21.

[3] JUDIT FULLANA, MARIA PALLISERA, J. C. R. F. P., AND PÉREZ-BURRIEL, M. Reflective learning in higher education: a qualitative study on students perceptions. *Studies in Higher Education 41*, 6 (2016), 1008–1022.