AUTOMATE ALL THE THINGS!

Virtual Design Master Season 4

Challenge 3

Paul Woodward

@EXPLOREVM Paul.Woodward.Jr@Gmail.com

Contents

Executive Summary

Now that we have, assumedly, taken control back of the Earth datacenter, it's time to continue to repopulate Earth. As an aftereffect of the datacenter breach and subsequent discovery of a pro-zombie movement on earth, Mr. Billionaire has placed a temporary freeze on adding members to the Earth Repopulation Technology Heroes team. This hiring freeze is on hold until Mr. Billionaire can come up with a peaceful resolution of differences with the pro-zombie faction to prevent any disruption of service at CHIDC01. Due to the freeze, the Earth Repopulation Heroes Team will be required to keep up with the ever increasing workload with their limited numbers. Automating tasks and services is a must.

Scope

Utilizing two different automation platforms, deploy 4 new Linux servers. Patch and manage 2 of the servers, as well as deploy a NGINX web server using the automation programs. Finally, deploy "Hello World" to each server using the automation tools, pulling code stored on GitHub.

Requirements, Constraints, Assumptions, and Risks

Requirements

- Deploy two different orchestration platforms on any host platform
- Deploy two (2) CentOS and two (2) Ubuntu servers using a provisioning service (your choice) so that one (1) of each are managed by each orchestration platform
- Create processes to patch each server daily
- Deploy an NGINX web server using your orchestration platform
- Deploy a "Hello World" onto each server using your orchestration systems and the source stored on GitHub
- Adaptability of the infrastructure is required to meet future challenges

Constraints

- Limited manpower is available

Assumptions

- There is an assumption, within reason, of best practices being used in the initial datacenter configuration.

Risks

- The pro-zombie movement could attack again, security must be considered

Automation Platforms

The two automation platforms chosen by the ERTH team are Ansible and Puppet Labs. Outlined below are some of the highlights of each platform and the reasoning behind their selection.

Ansible

Ansible is an easy to implement, easy to learn automation platform. Ansible does not require agents or a client server relationship with the servers it manages; it connects via SSH. The ease of deployment is matched by the ease of use of Ansible with Playbooks. Using YAML, playbooks are easy to read, easy to configure automation jobs to be utilized by Ansible.

Puppet

Puppet has been selected for the automation project due to its relative ease to learn. With Puppet admins can create code by selecting desired states and configuration of nodes and the platform will automatically generate the code. An added benefit Puppet is the Puppet Enterprise Console. The consoled provides a dashboard where admins can monitor the success and failures of code changes, as well as view which and how many nodes have had changes made to them. This feature integrates well with the existing NOC operations and security changes implemented in response to the attack by the bad actor.

Design and Deployment of the Platforms

Ansible

The Ansible Control Machine runs on any Linux platform which supports Python 2.6 or 2.7. It can be installed on a Windows based system, but requires lots of workarounds and is not officially supported by Ansible. For this reason, an Ubuntu Server 16.04 LTS virtual machine was created. As Ansible and Ubuntu are both light on system requirements, and in the interest of conserving resources, the VM was configured with 1 vCPU, 2 GB of vRAM, and 40 GB hard disk. This installation of Ubuntu Server comes with Python 3.0 installed by default, so Python 2.7 will be installed post creation.

Ansible Directory Structure

The Ansible directory will be broken down into two sections for usage and management purposes. The two tiered structure is designed to keep the production and, although not specifically called out by Mr. Billionaire, test/dev environment variables separated.

- Production
    - Group_vars
        - All
        - App
        - DB
        - Web
    - Host_vars
        - hostnameXX
    - Prod_Inventory
    - tasks
- Test/Dev
    - Group_vars
        - All
        - App
        - DB
        - Web
    - Host_vars
        - hostnameXX
    - TestDev_Inventory
    - tasks

Inventory File Structure

```
[vCenter]
#vCenter Server Info
ansible_ssh_host=10.15.2.5 Ansible_ssh_user=root ansible_ssh_pass=

[app]
# list for application VMs
CHIDCAPP[13:17].newearth.local

[db]
# list for database VMs
CHIDCDB[07:11].newearth.local

[web]
# list for web VMs
CHIDCWEB[29:33].newearth.local

[CHIDC01:children]
# group for all VMs that will populate CHIDC01
app
db
web

[CHIDC01:vars]
# group for variables across all VMs in CHIDC01
ansible_ssh_user=AnsibleAdmin
ansible_ssh_pass=P@ssw0rd!
ntp_server=0.pool.ntp.org

```

Puppet

Given the size of the datacenters, Puppet will be deployed in a “Monolithic” architecture. This means that the Puppet master, Puppet Enterprise Console, and PuppetDB will be installed on one server. Just like Ansible, Puppet be deployed on Linux as well. The Puppet VM will be installed with Ubuntu Server 16.04 LTS, however, as it performs multiple tasks, it requires more resources:

- 2 vCPU
- 6 GB of vRAM
- 40 GB hard disk.
    - 20 GB of which for /opt/

The above listed Puppet VM will support up to 10 nodes per Puppet documentation, which is ideal for the initial deployment and management of 2 VMs as defined by the scope of the project. As additional nodes are added to Puppet management, the monolithic architecture can be scaled to support up to 2000 nodes before additional VMs are needed.

Code Design Process

Ansible Playbooks

Ansible playbooks will be utilized for the following tasks:

- Deploying a CentOS & a Ubuntu server

- Continued patching, management, and version control of the servers
- Deploying NGINX Web Server

To save space, examples of the code can be found in the Appendix.

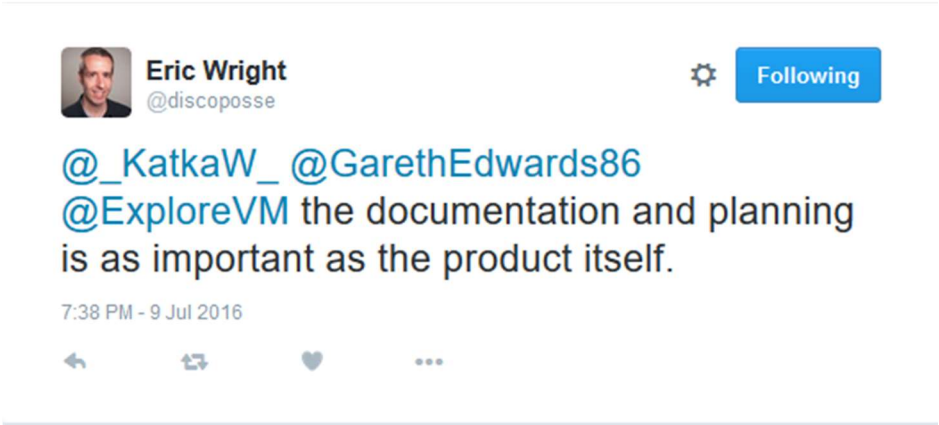
Full code can be found here: https://github.com/PWoodward83/ExploreVM

Lessons Learned

Lots. My exposure to automation up until the start of this challenge has been a little work with PowerShell/PowerCLI. I've known full well that this has been a skillset missing from my toolbox for a long time. Thankfully, this challenge has thrust me into learning, or attempting to learn, not just one, but two solutions.

The two programs I chose, Ansible and Puppet, were merely pulled from the ether as the only experience I've had with them was speaking with the vendor at VMUGs/VMworld, or reading blog posts about their use. I'm rather grateful for this challenge, as now I've seen how easy and useful the world of automation can be. Post Virtual Design Master, I am going to continue my learning with Ansible, as I found the product to be rather intuitive and easy to work with.

I had stared spending a lot of time early in challenge 3 rebuilding my home lab so that it could handle and perform the automations, and I was neglecting the documentation aspect. I decided that, since having a video demo of the solution was listed as a "bonus", and not a requirement, that I would focus on learning the infrastructure and coding that goes into each solution. To quote one of our illustrious leaders:



Appendix and References

Code samples

Installing NGINX using a playbook

```
#name install_nginx.playbook
---
- hosts: web
  sudo: yes

  tasks:
  - include: "tasks/install_nginx.yaml"
```

- The playbook points to the web host group in the inventory file, allows use of sudo rights, then includes the task to install NGINX which is stored in the tasks directory

Create VM from Template code example

```
#config file to create a VM from an existing template

- vsphere_guest:
    vcenter_hostname: CHIDCVC01.newearth.local
    username: root
    password:
    guest: CHIDCLINUX[02:03]
    from_template: yes
    template_src: UbuntuTemplate
    cluster: CHIDC
- vsphere_guest:
    vcenter_hostname: CHIDCVC01.newearth.local
    username: root
    password:
    guest: CHIDCLINUX[04:05]
    from_template: yes
    template_src: CentOSTemplate
    cluster: CHIDC
```

- This config file connects to a vCenter Server and creates a new VM from a template

Reference Links

GetHub
https://github.com/PWoodward83/ExploreVM
https://github.com/PWoodward83/Hello-World

http://frankhinek.com/create-ansible-playbook-on-github-to-build-mesos-clusters/

Ansible
https://www.ansible.com/
http://docs.ansible.com/ansible/intro_installation.html
http://www.jeffgeerling.com/blog/creating-custom-dynamic-inventories-ansible
http://everythingshouldbevirtual.com/creating-vsphere-vms-using-ansible
http://docs.ansible.com/ansible/vsphere_guest_module.html

Puppet
https://puppet.com/
https://docs.puppet.com/puppet/4.5/reference/architecture.html
https://docs.puppet.com/pe/latest/install_system_requirements.html

Ubuntu
https://wiki.ubuntu.com/XenialXerus/ReleaseNotes?_ga=1.78804500.1471387936.1468092416
http://www.ubuntu.com/download/server/install-ubuntu-server
https://help.ubuntu.com/lts/serverguide/preparing-to-install.html

CentOS
https://www.centos.org/download/

NGINX
https://www.nginx.com/blog/installing-nginx-nginx-plus-ansible/

Hello World
https://guides.github.com/activities/hello-world/
http://codingbee.net/tutorials/ansible/ansible-example-playbook/
http://docs.ansible.com/ansible/git_module.html