VIRTUAL DESIGN MASTER 3

PUPAPHOBIA DESIGN DOCUMENT V1.8

Challenge 4

Steven Viljoen
7-28-2015

Contents

Revision History

Date	Revision number	Author	Comments
23 July 2015	V1.0	S.Viljoen	Initial draft
24 July 2015	v1.1	S.Viljoen	Added design choices
25 July 2015	V1.2	S.Viljoen	My first Docker container is born on vacation in Croatia
25 July 2015	V1.3	S.Viljoen	Ansible is awesome…Think I am getting this.
26 July 2015	V1.4	S.Viljoen	Nope…Really hate Ansible.
26 July 2015	V1.5	S.Viljoen	Ansible done…Probably quicker to just teach them how to deploy a web server rather than waiting for me to figure this out.
27 July 2015	v1.6	S.Viljoen	Puppet started…Ansible doesn't seem so bad anymore in comparison.
28 July 2015	v1.7	S.Viljoen	uuughh…Feels like I reading ancient Greek documents.
28 July 2015	v1.8	S.Viljoen	Out of time and grey matter

Executive summary

We're heading back to Earth.

We will be sending a tough (but not so bright) group of humans called the Epistemophobics to secure certain areas across the globe. To achieve this they will need working infrastructure.

As IT personnel are in short demand all infrastructure deployment will need to be handled by the technically challenged Epistemophobics. For this reason an orchestrated system needs to be developed to rebuild the infrastructure with little to no input from the !geeks.

Before leaving we will test out the approach by putting in 2 proof of concept models that will automate the deployment of a simple web application that displays "Welcome Back to Earth!"

To properly evaluate and compare the deployment possibilities we will be using 2 orchestrating tools, 2 different OS and 2 different web servers.

Given the unknown state and availability of the infrastructure on Earth the web servers will need to run within a container.

Disclaimer

As of 23 July 2015 the lead design engineer (name withheld) had no experience with any orchestration tool or Docker and his Linux skills fluctuated between a Stevie Wonder impression on his keyboard to '**ls**'ing the hell out of a RHEL file system. While all efforts were made to ensure that all documented code works and build steps are accurate he does profusely apologize for any preschool technical terms\descriptions.

The Laptop

Given the fact that no IT personnel will be sent back to Earth and that all deployments will be done by the Epistemophobics it has been decided that each group will be given a preinstalled laptop running the following:

1. Orchestration tools
    a. Ansible
    b. Puppet master
2. Common infrastructure services
    a. DNS server
    b. NTP server

Requirements

Reference	Description
RQ001	Needs to be able to be deployed by one of the Epistemophobics.
RQ002	Should require as little human interaction with the target infrastructure as possible given the skill level of the one deploying it.
RQ003	Should include a step-by-step walk through.
RQ004	Should be supported on the most common Operating Systems on Earth. (Windows, RHEL, CoreOS, Ubuntu, OS X)
RQ005	Should be scalable to extend past POC without full redesign.

Constraints

Reference	Description
CS001	Unknown state of the shared infrastructure on Earth (hubs).
CS002	Severe lack of knowledge with anything related to Docker, Orchestration tools and Linux.
CS003	

Risks

Reference	Description
RI001	Deployment team are technically illiterate.
RI002	Complex IT infrastructure (clouds) might not be available.
RI003	

Assumptions

Reference	Description
AS001	Electricity will still be working.
AS002	Basic NW connectivity will still be functional.
AS003	

Design Choices

Container

As the state and type of infrastructure that remains on Earth is unknown it has been decided that the applications will run in containers.

The container software of choice must fulfil the following requirements:

1. Must be able to use local image repositories. [CS001]
2. Needs to support at least 2 of the major Earth Operating systems.[RQ004]
3. Must be supported to run on major Earth based public cloud infrastructure.[RQ005]
4. Must be easy to extend functionality without impacting applications [RQ005]

Based on these requirements Docker has been chosen.

- Supports local image repositories ~ (Pustina)
- Runs on Ubuntu and CentOS~ (Docker.com)
- Supported on AWS and Azure ~ (Docker.com)
- Uses extension points to allow 3rd parties to extend the base functionality with specific modules. ~ (Marsden)
- Who can resist that little whale logo.

Web servers

Given that the web server will run in a container there are no real Operating system requirement however the following requirements have been identified to increase interoperability and future scalability.

1. Must support IPv6 [RQ005]
    a. Old IPv4 IPs could still be recorded in Earth based public DNS servers.
2. Must support current Mars based programming skills (mostly Python)

Based on these requirements the following 2 web servers have been chosen: ~ (Wikipedia.com)

1. Apache (HTTP Server)
    a. Supports WSGI
    b. Supports IPv6
2. nginx
    a. Supports WSGI
    b. Supports IPv6

Orchestration tools

To provide a better evaluation platform for which deployment method works best it is needed to have:

- Push deployment and Pull deployment.
- Agent and Agentless
- Needs to support at least 2 of the major Earth Operating systems.[RQ004]

The following Orchestration tools have been chosen

1. Ansible
    a. Push deployments
    b. No agents needed
    c. Runs on Ubuntu
2. Puppet
    a. Pull deployments
    b. Master server and node agents need to be installed
    c. Runs on CoreOS

ANSIBLE-NGINX BUILD

For the first proof of concept I am using 2 Ubuntu 14.04 machines running on VMware workstation on my laptop (it was a bit difficult to take my lab with me to Croatia on vacation).

This POC will be using:

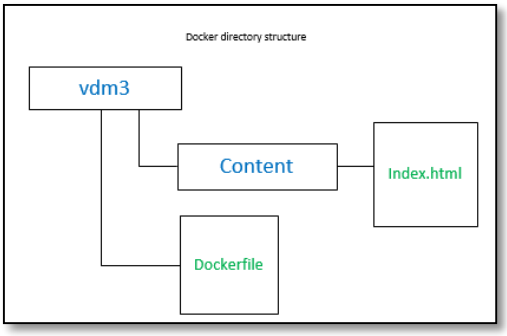
Container	Web Server	Operating System	Orchestration Tool
Docker	**Nginx**	**Ubuntu**	**Ansible**

The deployment process is divided into 4 sections

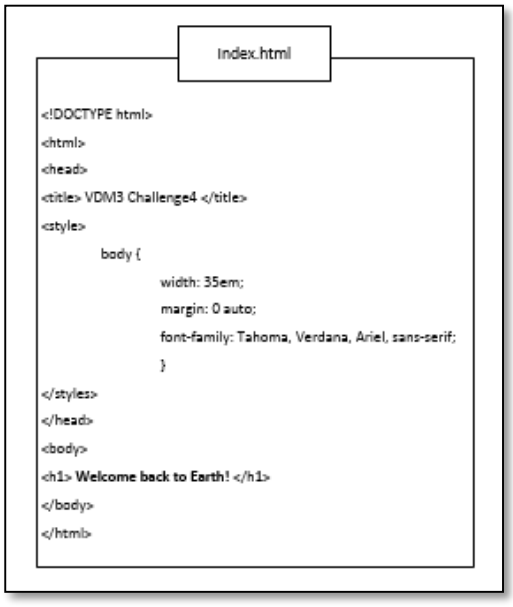
- Environment: Setting up the management environment
- Docker: Installing Docker and creating the Nginx image
- Ansible: Installing Ansible and creating the playbook
- Deployment: Running the Ansible playbook.

Environment

1. Install Openssh-server if needed.
2. Open port 22 on the UFW
3. Create the working directory structure
    - vdm3 directory
        - vdm3/Dockerfile
        - vdm3/content directory
            - vdm3/content/index.html



4. Modify the index.html file to display the required welcome message



Docker

1. Install Docker on the management server

```
$ sudo apt-get update

$ sudo apt-get install wget

$ wget -qO- https://get.docker.com/ | sh
```

2. Modify the vdm3/Dockerfile created in the beginning.

```
FROM nginx

COPY content/ /usr/share/nginx/html
```

3. Build the nginx image using the Dockerfile.

```
$ sudo Docker build –t earth_i1
```

4. Check the new image

```
$ Docker images
```

```
REPOSITORY          TAG                 IMAGE ID            CREATED
VIRTUAL SIZE
earth_i1            latest              8c532d16df5b        7 hours ago
132.9 MB
```

Ansible

1. Install Ansible on the management server

```
$ sudo apt-add-repository -y ppa:ansible/ansible
$ sudo apt-get update
$ sudo apt-get install -y ansible
```

2. Disabled StrictHostKeyChecking in ~/.ssh/config

```
Host *
  StrictHostkeyChecking no
```

*Don’t think we have to worry about Zombie hackers

3. Uncomment #ask_sudo_pass=True in /etc/ansible/ansible.cfg

```
ask_sudo_pass=True
```

4. Create Ansible hosts file in /etc/ansible

```
[earthwebs]

192.168.241.130
```

*This will need to be updated by the Epistemophobics to point to the target server IPs.

5. Create the Ansible playbook ***ubuntunginx.yml***

```
- name: Deploy nginx
  hosts: earthwebs
  sudo: True
  tasks:
   - name: install curl
     apt: name={{ item }} update_cache=yes
     with_items:
       - git
       - curl
       - libpq-dev
       - python-dev
       - python-psycopg2
   - name: install docker
     shell: curl -sSL https://get.docker.com/ | sh
     args:
       creates: /usr/bin/docker

   - name: install pip
     shell: curl https://bootstrap.pypa.io/get-pip.py | python -

   - name: Install docker-py
     pip: name=docker-py

   - name: Deploy container
     docker:
       name: earth_c1
       image: earth_i1
       state: started
```

6. Run the playbook

```
$ ansible-playbook –s ubuntunginx.yml -k
```

Deployment

```
PLAY [Configure development machine] ***************************************

GATHERING FACTS ************************************************************
ok: [192.168.241.130]

TASK: [install curl] *******************************************************
ok: [192.168.241.130] => (item=git,curl,libpq-dev,python-dev,python-psycopg2)

TASK: [install docker] *****************************************************
ok: [192.168.241.130]

TASK: [install pip] ********************************************************
changed: [192.168.241.130]

TASK: [Install docker-py] **************************************************
ok: [192.168.241.130]

TASK: [Deploy container] ***************************************************
failed: [192.168.241.130] => {"changes": ["{\"status\":\"Pulling repository eart
h_i1\"}\r\n", "{\"errorDetail\":{\"message\":\"Error: image library/earth_i1:lat
est not found\"},\"error\":\"Error: image library/earth_i1:latest not found\"}\r
\n"], "failed": true, "status": ""}
msg: Unrecognized status from pull.

FATAL: all hosts have already failed -- aborting

PLAY RECAP ********************************************************************
           to retry, use: --limit @/home/steven/ubuntunginx.retry

192.168.241.130            : ok=6    changed=1    unreachable=0    failed=1
```

Well that's dissapointing….even more so when you have spent the last 2 days trying to get to this point.

However, from what I have read the issue is caused due to the fact that the docker module looks for the referenced image in the public repository Docker Hub.

This could be solved by :

1. Uploading the image to the Docker Hub.
2. 'Somehow' pointing it to a local private repository
    a. Numerous attempts to do this but still haven't got it to work.

Epistemophobics deployment

The team will be directed to a predetermined site chosen for its infrastructure and suitable hosts.

Upon landing on Earth they will be expected to:

- Plug the laptop into an existing network
- Edit the ansible hosts file with the IPs of the appropriate target servers
- Run the playbook.
- Connect via browser to ensure that the web site is available and that the message is displayed.

PUPPET-APACHE BUILD

Container	Web Server	Operating System	Orchestration Tool
Docker	**Apache (HTTP Server)**	**Centos 7.0**	**Puppet**

The deployment process is divided into 5 sections

- Environment: Setting up the management environment
- Docker: Creating the Apache image
- Puppet Master: Installing and configuring the Puppet Master
- Puppet Node: Installing and configuring the Puppet Node
- Deployment: Setting up and deploying the Apache image

Environment Setup

Common infrastructure services

Deploy the DNS server

- Name:NS1.earth.com
- Install and configure Bind as a DNS server.

Docker

1. Reusing the previous Docker installation (to save time) so no installation is needed.

2. Create a new Dockerfile in vdm3/apachebuild/

```
FROM httpd

COPY content/ /usr/local/apache2/htdocs/
```

The content directory contains the same modified index.html file detailed in the nginx build.

3. Build the apache image using the Dockerfile.

```
$ sudo Docker build –t earth_i2
```

4. Check the new image

```
$ Docker images
```

```
REPOSITORY          TAG                 IMAGE ID            CREATED
VIRTUAL SIZE
earth_i2            latest              25f6b833467b        4 seconds ago
162 MB
```

Puppet Master

Deploy the Puppet Master server

- Name: puppet.earth.com
- open port 8140 in UFW
- Install and configure NTP – needed for the certificates

Install Puppet Master

- Install
  - *$ sudo apt-get update*
  - *$ apt-get install puppetmaster*

Configure certificates

- Add the puppetmaster names to the [Main] section in /etc/puppet/puppet.conf
  - certname = puppet
  - dns_alt_names = puppet, puppet.earth.com
- Generate a new Certificate
  - *$ sudo puppet master --verbose --no-daemonize*

Create the site.pp manifest file

- *$ sudo touch /etc/puppet/manifests/site.pp*

Start puppetmaster

- *$ sudo service puppetmaster start*

Puppet Node

On the node server

- Name: Node1.earth.com

Install the Puppet agent

- *$ sudo apt-get update*
- *$ sudo apt-get install puppet*

Configure agent to start at boot

- Change START to yes [Start=yes] in /etc/default/puppet

Configure /etc/puppet/puppet.conf

- Delete [master] section
- Add
  - [agent]
  - server = puppet.earth.com

Start the Puppet agent

- *$ sudo service puppet start*

Sign the agent node's certificate

- On master
  - *$ sudo puppet cert list* to identify all pending requests (no + prefix)

- *$ sudo puppet cert sign <agent fqdn>* or *sudo puppet cert sign --all*

Deployment

I will be using the puppet Docker module created by Gareth R (https://github.com/garethr/garethr-docker).

Install requirements

- *$ puppet module install puppetlabs/stdlib*
- *$ puppet module install --force stahnma-epel*

Install Docker module

- *$ puppet module install garethr/docker*

Configure Docker on PuppetAgents

- Modify site.pp
  node default {
  include 'docker'
- *$ puppet agent –t*

Setup the Apache image

- docker::image {'earth_i2'
  }

.....Time's up….

What still remains to be done:

Time and vMiss33 wait for no man… or something like that.

The following tasks would need to be accomplished but due to severe time restrictions are not documented in this manual.

- Define a private repository
- Deploy the images to the Puppet agents

Epilogue

As of 28 July 2015 the lead design engineer (name forgotten at this stage) has volunteered to join the Epistemophobics and lead the charge back on Earth.

References

Docker, Acodemy and. *Docker: Learn Docker In A DAY!* 2015.

Docker.com. *Installing docker on Ubuntu*. n.d. <https://docs.docker.com/installation/ubuntulinux/#installing-docker-on-ubuntu>.

—. *Supported installation*. n.d. <https://docs.docker.com/installation/>.

Hochstein, Lorin. *Ansible: Up and Runing*. 2014.

Marsden, Luke. *Extending Docker with Plugins*. 22 6 2015. <http://blog.docker.com/2015/06/extending-docker-with-plugins/>.

Pustina, Lukas. *Docker Registry or How to Run your own Private Docker Image Repository*. 18 02 2014. <https://blog.codecentric.de/en/2014/02/docker-registry-run-private-docker-image-repository/>.

R, Gareth. *garethr-docker*. n.d. <https://github.com/garethr/garethr-docker>.

Ubuntu.com. *Using the terminal*. n.d. <https://help.ubuntu.com/community/UsingTheTerminal>.

Wikipedia.com. *Comparison of web server software*. n.d. <https://en.wikipedia.org/wiki/Comparison_of_web_server_software>.

Wittig, Elizabeth. 2014. <https://puppetlabs.com/blog/starting-puppet-basics-from-puppet-labs-employee>.