Virtual Design Master

Season 3 - Challenge 4

To the shipping docks!

Prepared by

Dennis George

Authors

The following authors contributed to the creation of this deliverable.

Dennis George
892 Bessy Trail, Milton, ON L9T 0A6 Canada (905) 699 – 3151 dennisgeorg@gmail.com

Revision History

Revision	Change Description	Updated By	Date
0.1	Document Created	Dennis George	07/28/2015

Table of Contents

SECTION 1: OVERVIEW

Executive Summary

Project Overview

Our millionaire philanthropist friend is seeking an infrastructure design for permanent human colonization on Mars. Virtual Design Master an online reality show that challenges virtualization professionals to come up with innovative virtualization designs has been tasked to select the next Virtual Design Master to design a permanent IT infrastructure on Mars.

In challenge 1, the team designed an “on-prem” infrastructure solution to support various critical control systems such as the Environmental system, Greenhouse control system, and productivity and collaboration systems.

As part of the challenge 2 initiatives, the team was tasked to design a solution to support the same requirements in a public cloud of choice and provide justification for the same.

Then for challenge 3, the team was asked to design a disaster recovery plan for some key applications on Mars, by deploying our own infrastructure or leveraging the cloud.

Now that we’ve established ourselves across the solar system, for Challenge 4 it is time to make plans for taking our home planet back from the zombies. Select teams of humans and robots will be sent back to Earth to attempt to secure different areas across the globe. We will of course need working infrastructures help us with this as we reclaim the planet. Chances of sending back IT people are pretty slim, so you will need to create an orchestrated system to rebuild the infrastructures. We know this is not an easy task so we are going to start small.

Our test will be a simple web application that displays “Welcome Back to Earth!”.

Since we are evaluating the best way to accomplish orchestrating our infrastructure, you will need to use two different orchestration tools, two different operating system, and two different web servers. Because we don’t know what type of infrastructure we’re going to end up with (we could end up with a mix of equipment and operating systems), the web servers must run inside of a container. Share your code, and build instructions so that a complete walk through is available. Make sure to include dependencies your application may have.

After careful planning, and thought process the team has designed the follow design to deploy the Web Applications in an automated and orchestrated fashion.

Project Goals

During the course of the project, the Virtual Design Master team and Dennis George identified a number of different project goals. The following summarizes those goals and illustrates how this infrastructure them.

Goal ID	Description
GO01	Automate and orchestrate infrastructure build procedures for rebuilding web servers on Earth.
GO02	Use container technology to accommodate varying hardware and OS availability.
GO03	Solution should accommodate at least two different Operating Systems and two different Web Servers.
GO04	Share the code used to build and associated instructions, including any dependencies the applications may have.

Constraints

During the course of the project, the team has identified different constraints to the Disaster Recovery Plan.

The following table summarizes the constraints:

Constraint ID	Description
CS01	Learn and design an automated deployment process for a technology that one has zero experience in four hours! Way to go Melissa, you made me learn Docker tech. and container tech in significantly short amount of time. This is why we love what we do!

Risks

During the course of the project, the team has identified risks to the Disaster Recovery Plan, and the following table summarizes the risks:

Risk ID	Description
RI01	My first attempt with Docker containers! As it was apparent from Challenge 3!
RI02	HTTP traffic over the “wild web”! Good thing it is just a welcome message else the security team would flip out!

Assumptions

The team has made some assumptions for the proposed automated infrastructure deployment plan and the following table summarizes the assumptions made:

Assumption ID	Description
AS01	We are most likely going to deploy this infrastructure on public cloud services that have somehow survived the apocalypse on Earth. Besides we are short of IT folks to send to Earth.
AS02	The firewall exceptions for the Web application to be exposed to the internet is expected to be configured for access on the Internet.

SECTION 2: UBUNTU AND APACHE2 WEB SERVER

Architecture

Overview

The primary driver for this design has been automation from the ground up. Meaning that the team has decided to leverage a vanilla Ubuntu Docker image and automate the layering of services required to build the web server and deliver the web page, along with automated deployment of the containers.

The design has emphasized on quick and hands-off deployment of the infrastructure across the different locations (whether public or private cloud). The end-goal being to deliver the message that we are back on Earth to reclaim what is rightfully ours!

Image build

Overview

For the Docker image creation process itself, the team has configured the following Dockerfile config. to leverage a publicly available Ubuntu image, and automatically layer it with Apache web server, necessary tools and also download the webpage payload from our Mars base station (**http://104.197.47.189:5000/**) which happens to be a 3-node Kubernetes cluster running on the Google Compute Engine fronted by loadbalancers on Mars!

As a backup, a finished image will be stored in a private repository, and managed with any updates to mitigate the risk of public repositories not being under our control.

Dockerfile

```
FROM ubuntu
MAINTAINER Dennis George <dennisgeorg@gmail.com>
RUN sudo apt-get update && apt-get install -y apache2
RUN sudo apt-get update && apt-get install -y curl
RUN sudo update-rc.d apache2 enable
RUN sudo service apache2 start
RUN curl http://104.197.47.189:5000 > /var/www/html/index.html
```

Dependencies

The following table outlines the dependencies for the automated image build process:

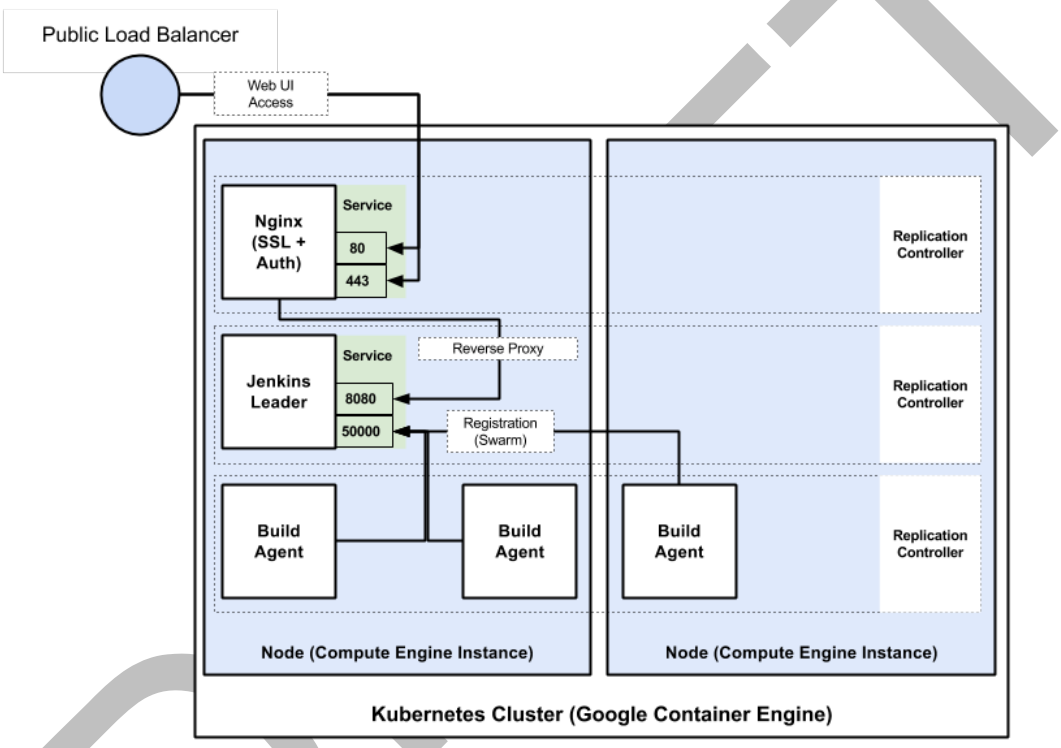
Dependency ID	Description
WEBSRV01DEP01	Availability of the vanilla Ubuntu Docker image on publicly available repositories.
WEBSRV01DEP02	Availability of Apache2 web server on publicly available Ubuntu repositories.
WEBSRV01DEP03	Availability of Curl on publicly available Ubuntu repositories.
WEBSRV01DEP04	Availability of our Mars base infrastructure for the web page payload.
WEBSRV01DEP05	Availability of public Google drive infrastructure for images embedded in the web page (for the aesthetics!).

Container deployment

Overview

For automation of the container deployment, the team has decided to leverage the Kubernetes controller along with the Jenkins and Swarm.

Kubernetes architecture



Dependencies

The following table outlines the dependencies for the automated image build process:

Dependency ID	Description
WEBSRV01DEP06	
WEBSRV01DEP07	
WEBSRV01DEP08	
WEBSRV01DEP09	
WEBSRV01DEP10	

SECTION 3: COREOS WITH NGINX WEB SERVER

Image build

Overview

Dependencies

The following table outlines the dependencies for the automated image build process:

Dependency ID	Description
WEBSRV02DEP01	Availability of the vanilla CoreOS Docker image on publicly available repositories.
WEBSRV02DEP02	Availability of NGINX web server on publicly available CoreOS repositories.
WEBSRV02DEP03	Availability of Curl on publicly available CoreOS repositories.
WEBSRV02DEP04	Availability of our Mars base infrastructure for the web page payload.
WEBSRV02DEP05	Availability of public Google drive infrastructure for images embedded in the web page (for the aesthetics!).

Container deployment

Overview

Dependencies

The following table outlines the dependencies for the automated image build process:

Dependency ID	Description
WEBSRV01DEP06	
WEBSRV01DEP07	
WEBSRV01DEP08	
WEBSRV01DEP09	
WEBSRV01DEP010	

SECTION 4: REFERENCES

Supplemental Information and Links

These links provide further information on the concepts and recommendations discussed during this document.

- Your First Custom Image in Docker – Thank you Melissa for planting the Docker bug in me!
- Docker user guide - Docker 101!
- ANNOUNCING DOCKER MACHINE, SWARM, AND COMPOSE FOR ORCHESTRATING DISTRIBUTED APPS – A glimpse into Docker Machine, Swarm and Compose.
- HOW TO AUTOMATE DOCKER BUILDS AND AUTO DEPLOY - This article describes how to create an Automated Docker Build that auto deploys Docker containers in combination with these services
- Automated Image Builds with Jenkins, Packer, and Kubernetes – GCE documentation outlining the process for automation of container deployment.