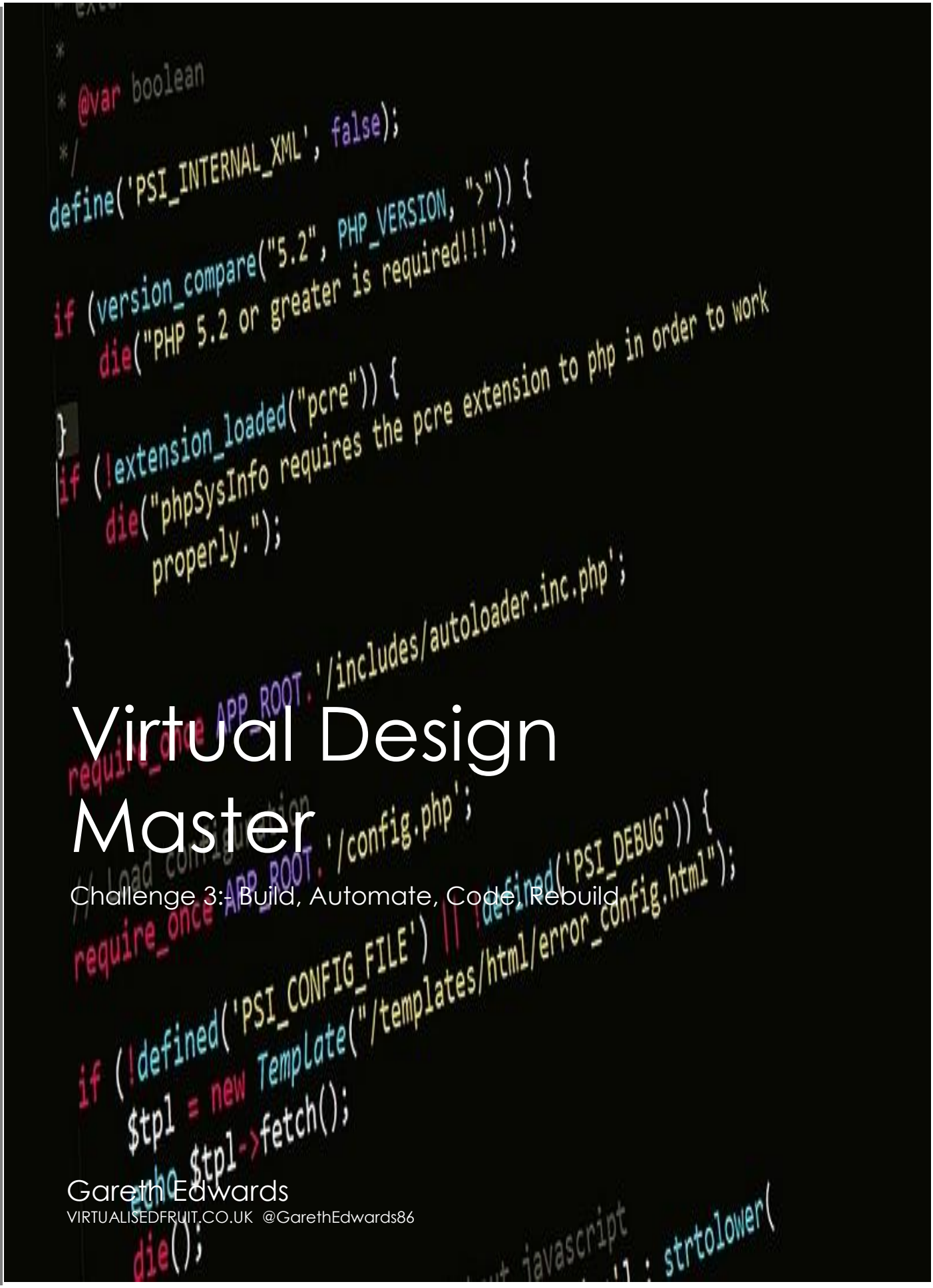


Virtual Design Master

Challenge 3:- Build, Automate, Code, Rebuild

Gareth Edwards
VIRTUALISEDFRUIT.CO.UK @GarethEdwards86

[Synopsis]

As we learned in our last challenge, not everything is as it seems. We are beginning to learn more about the group that hijacked the supply shipment. They believe the zombies are to be protected, and Earth belongs to them. We now have to worry about our re-colinization efforts being sabotaged by this rogue group. Our billionaire philanthropist friend has temporarily frozen new recruits to the re-colinization effort until he comes up with a plan. He thanks you for your ideas from the second challenge, and is beginning to implement them throughout his organization. Since our infrastructure is growing, but our ranks are not, orchestration has become a necessity. We can not rely on being at the keyboard in hostile environments or remote locations. To solve our problem for this design scenario, you need to design a configuration management framework that is able to deal with the launch and continuous configuration of your infrastructure. We don't know what to expect, so our infrastructure needs to be adaptable.

This is a practical and documentation challenge.

You will need to: • Deploy two different orchestration platforms on any host platform of your choice • 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 with the source stored on GitHub BONUS: Integrating anything else into your deployment workflow such as security checks, build triggers, and more will be nice add ons Share your code, and build instructions as well as a document detailing the infrastructure design. Having video walkthroughs of your working environment are also a bonus. Make sure to note any dependencies you have to be able to deploy your infrastructure.

Table of Contents

1) Executive Summary

a) Project Overview

If we didn't think the zombies were bad enough they seem to have a cult following them and willing to protect them. We now need to concentrate on making our infrastructure as automated as possible as it is key to our survival.

b) Intended Audience

This guide is intended for the vDM judges plus anyone within our team to build our systems.

c) Project Summary

In this project we automate as much of our systems as possible from system provisioning, updates and code deployment.

i) Project Requirements

PRQ001. Deploy two different orchestration platforms on any host platform of your choice
PRQ002. 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
PRQ003. Create processes to patch each server daily
PRQ004. Deploy an NGINX web server using your orchestration platform
PRQ005. Deploy a "Hello World" onto each server using your orchestration systems and with the source stored on GitHub
PRQ006. BONUS Requirement:- Integrating anything else into your deployment workflow such as security checks, build triggers, and more will be nice add ons

ii) Project Assumptions

A001. We can use any currently available software systems
A002. We would have access to any amount of hardware we require
A003. We have friendly developers to help us
A004. We would have more than a week!

iii) Project Constraints

C001. Current lab conditions are a bottle neck for testing
C002. Time!
C003. Using the specified OS versions

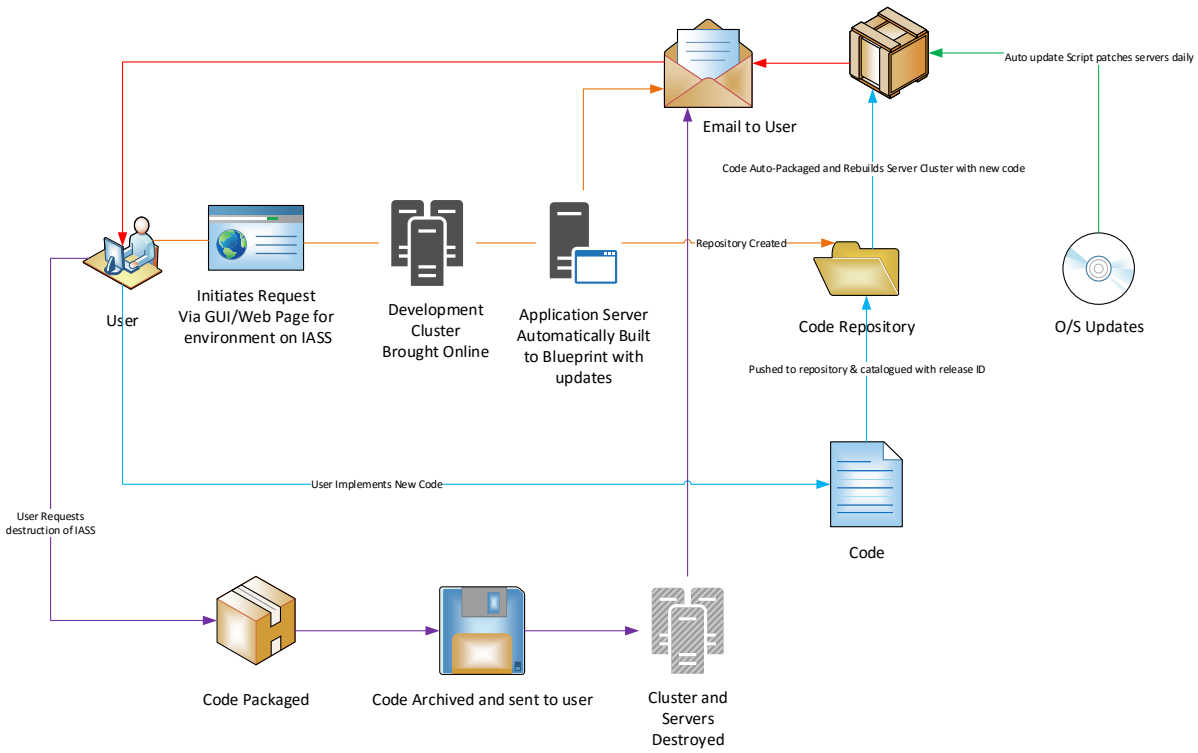
iv) Project Risks

PRI001. We may introduce code that will impact production services

2) Logical Design Summary

a) User Interaction

As this task is all about creating an automated workflow system we need to first outline the proposed system in order to do this. Where possible we are going to attempt to have it completely autonomous so that we can have a continuous delivery system so we can further prevent any security or bug based issues.



i) IASS

Somewhere within our design we will need some form of infrastructure and provide this as a service. This will need to be capable of being controlled with an API or scripts so we can build new infrastructure.

ii) Orchestration

The orchestration tool(s) must be able to pull in some form of predefined blue print that can hand off to IASS and also kick start the automated builds.

iii) Automation

(1) Operating Systems

Within our design we must be able to automate the update and control of the operating system chosen. Where possible this may require another technology layer of which we can interact with for scalability.

(2) Application Automation

Any application that we deploy must be able to be automated or containerised so that it can be consistent, code can be tracked and easily rebuilt or destroyed if required.

b) Systems Management Lifecycle

Where possible we need to try and involve a systems management lifecycle to the above so that everything can be audited. This may not just involve one systems but we should attempt where possible to provide feedback from each layer so the process of tracking is as simple as possible.

3) Let the coding begin..

a) Our first choice, Stay on Premise

I have explored two several on premise options and found that due to time constraints and feature sets available that I would utilise VMware's vRealize Orchestrator and vRealize Automation. This was also to enhance the end users experience when trying to build the infrastructure as a service. The major negative to this is I was unable to provide an automated build at present when code is pushed to the GitHub. The blueprint would need to be re-run via the GUI or via script at predefined intervals for now.

i) Infrastructure

Our infrastructure for this element will be based on our original hardware and using VMware. We have decided where possible each new virtual infrastructure request will be containerised in their own VLAN to ensure that no data is spread between segments unless requested to the infrastructure team.

ii) Orchestration

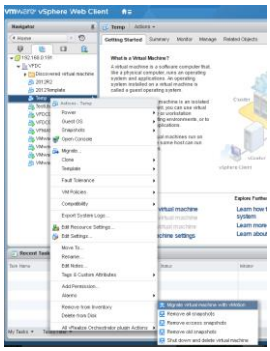
We will be using vCenter Orchestrator to perform the management and creation of VMs from predefined blueprints. Some of these will already for the code embedded for applications and updating services via automated tasks for the operating system.

iii) Automation

For the automation element and for an enhanced end user experience we will be using VMware vRealize Automation. We will be using this so that there are minimal touches on the back end infrastructure.

i) Conclusion

The above idea overall seems to work in practise but it is still missing the code release element. Unfortunately, due to the time rebuilding my lab I was not able to fully establish a run book or work flow. However, this has given me some exposure to vRO and I will continue to use this as I know this will be key to my home lab in the future. Some may think I could have used the great resource (VMWare Hands-on-Labs, n.d.)but it's not easy to implement your own code or ISOs so I didn't want to put much time into here. The screenshot below shows the example run books I got into vCenter.



b) A journey to the cloud

I was able to find some elements that I felt where missing from the on premise version but they were cloud based for now in parts. Unfortunately, I did not leave enough time to produce a workflow in this area and I was unable to access some of the resources so I am placing all the below in theory with a hope to try this soon. I took pretty much all of my inspiration from (Christner, n.d.) "How to Automate Docker Builds and Auto Deploy" but for me it is missing the ease of use for users. We would need to automate a GUI for this of which could ironically be deployed by the same system, maybe this could be seen and a single point of failure.

i) Infrastructure

For rapid deployment and to lessen the burden on our own infrastructure we are going to use Amazon web services and these can easily be called by an API. This of course probably isn't the most cost effective route but can be seen as easier to implement and we haven't been constrained by a budget so it seems to be the best choice.

ii) Orchestration

For orchestration we will be using Tutum for the code & rebuilds but Ansible to control the initials builds so that we can ensure we cover the requirements of CentOS and Ubuntu software stacks.

iii) Automation

Automation will take form in GitHub and DockerHub to control the release of code and then into Tutum to rebuild the infrastructure/containers.

i) Conclusion

This idea can pretty much cover all the projects requirements of the deployment but would require more thought and a higher amount of input from the user unless we could create a GUI interface to run books for Ansible.

c) 11th Hour Panic:- A combination of the two

I was able to find out since the release of the above articles and me starting to open accounts for some of the aspects, I soon realised what appeared to be cloud connectors at the time of writing are now possible to back to bring your own servers. This appears to have now got over my code issue with my first design and I can fully automate GitHub releasing code via Docker Hub to Docker Cloud (formally Tatum) all automatically. This also allows for my code to be tracked and can even send me an email every time the code is updated deployed. I feel if I could have got the run book to work in vRO and then deploy the curl url as an example below it could have auto created these into containers and deploy NGNIX and our required code with ease.

https://docs.docker.com/docker-cloud/infrastructure/byoh/

I have performed a video example here:-

https://www.youtube.com/watch?v=OaaRoKBDLvc

d) You missed patching?!?!

Yes I have but I was thinking if these containers are recreated on a regular basis I could have a hook to update on build but automate the base images to launch each night, update and then shutdown ready for deployment. I could also just use a cronjob to perform this on the linux operating systems every night but a new update may well break release code on the operating system of which the developers should be aware of. If this was the case it may be wise to hold 14 days' worth of images to ensure we could roll back to retest the code.

2) Final Thoughts

a) What would I have done different

This task soon became a fight for survival. Whilst trying to be clever and think about automating the updates one of my core components failed taking down my entire lab. After a few calls into support I managed to get the network back online but this caused a split in my VSAN and all my data was lost, along with my backup system of which didn't appear to be archiving as expected. We all learn to check things regularly even if it's a lab, but it was a good time to start afresh after migrating from many versions of hardware and breaking my domain schema several times. It also opened up my mind to looking at alternative hypervisors of which I started to look at Nutanix of which at the time of writing unfortunately doesn't support NVMe so that wrote both my NUC hosts off. It has also made me realise I don't get enough exposure to automation, orchestration or even containers anywhere near enough and I must continue to learn after all this is over. I also think that there isn't any one right way of doing this and maybe a combination of all the systems would be best such as cloud for test and dev and on premise for production or even the other way round depending on sprawl and rebuilds. I didn't even start to explore other cloud systems such as Azure, Digital Ocean or even IBM SoftLayer. I know I will be keen to get vRO up and going along with Nutanix even if its nested for now. They had a great document referencing easy (Docker deployments on AHV, 2016) which may of well suited the requirements set out.

References

Christner, B. (n.d.). *How to Automate Docker Builds and Auto Deploy*. Retrieved from Brian Christner: https://www.brianchristner.io/how-to-automate-docker-builds-end-to-end/

Docker deployments on AHV. (2016, April). *Docker Containers on AHV*. Retrieved from Nutanix.com: http://www.nutanix.com/go/docker-container-best-practices-guide-with-acropolis-hypervisor.html

VMWare Hands-on-Labs. (n.d.). *VMware HOL*. Retrieved from http://labs.hol.vmware.com/

Disclaimer

The view expressed in this document are my own and do not necessarily reflect the views of my current, previous or future employer(s). This is a fictional design and some elements may not work correctly within your infrastructure. All data and information provided on this this document is for informational purposes only. I make no representations as to accuracy, completeness, currentness, suitability, or validity of any information throughout the document & will not be liable for any errors, omissions, or delays in this information or any losses, injuries, or damages arising from its display or use. All information is provided on an as-is basis.

Revision History

Version	Performed By	Date / Time	Comment	Action
0.1	Gareth Edwards	08/07/2016	Template Reset	Initial Action
0.2	Gareth Edwards	09/07/2016	Initial body created	Initial Thoughts
0.3	Gareth Edwards	10/07/2016	Lab Build	
0.4	Gareth Edwards	11/07/2016	Updates	
0.5	Gareth Edwards	12/07/2016	Docker Cloud	
1.0	Gareth Edwards	13/07/2016	Release for submission	Release for submission

End of document