Virtual Design Master

Season 5 Challenge 2

Version 1.0

Chris Porter

A document in two halves that don’t make a whole

Document History

Version	Date	Changes
1.0	12/07/2017	Promoted combine document to version 1
0.3	12/07/2017	Combine security incident and response into this document from a separate document.
0.2	12/07/2017	Significant updates including a picture of a ship
0.1	08/07/2017	Document creation

Contents

1. Overview

Editor's Note	As my Challenge 1 submission made an incorrect assumption that an application architecture was not required, this document corrects that mistake by first including an application architecture. After that is outlined, it then goes on to address Challenge 2.

1.1 Application Design Prelude

The HumanityLink 2 development team left us with an initial brief that was ambiguous about the application which would run on top of the required 3 site architecture infrastructure. It was assumed that this was because the application requirements would be provided after the infrastructure had been designed, which meant the infrastructure had to be as flexible as possible to have the best chance of meeting an unknown application architecture. Therefore AWS was chosen for the infrastructure design for its flexibility.

The Development Team have now returned from their extended team building event and have now provided us with more details around the application. During their team building event they realised they were spending too much time administering their application stack and not enough time spent on coding. They also realised that the large-scale deployment of a little understood environmental platform such as terraforming would very rapidly accelerate scientific understanding of this process and they would therefore need to make changes to the application very quickly to respond. Even though large parts of the team building event are quite hazy (this could have been due to evening team building events or the fact that the event took place on Venus[i]), they also understood that if they are to make changes to the application rapidly these need to be tested so as not to make the application unstable and to also avoid introducing security vulnerabilities. In a final moment of enlightenment, they realised they should engage with the infrastructure architecture team to help them design a solution which would meet these requirements.

Scientists will need to analyse the data captured from robots. As the scientists didn't go on the team building exercise, they still think using Windows desktops and Excel is the best way to do this.

1.2 Application Design Summary

This design presents a Pivotal Cloud Foundry (PCF) infrastructure across two AWS regions, to support the application developer's requirements. AWS IoT with Greengrass is used to manage the army of robots. AWS Workspaces is used to meet the scientist's requirements.

1.3 Application Design Scope

This document covers the application architecture only, it does not cover the majority of the underlying infrastructure. The underlying infrastructure is covered in the following document; Chris Porter - VDMS5 Challenge 1 Submission - v1.0.pdf. However, any amendments required to that infrastructure that have been made now the application requirements are known will be included in this document.

1.4 Challenge 2 Design Brief

You thought everything was running smoothly after you implemented the HumanityLink 2.0 software across the earth. It was, for a little while at least.

Something has gone horribly wrong in the brand new infrastructure you have just implemented. The first site of your design from Challenge 1 has become infected with the EatBrains virus and ransomware which is now running rampant across your infrastructure. The EatBrains virus infects files and maps to SMB shares in order to spread itself further. There is also a variant that has the potential to store itself in memory on network devices and to emulate routers within the environment. EatBrains also has a phone home feature that allows for remote execution using the infected devices in the environment.

You must take your design from Challenge 1 and illustrate how you will accomplish the following:

1. *Define how you will secure your infrastructure*
2. *Describe how you will find where the intruder has already breached and has remote execution capability*
3. *Create a recovery plan for every layer of the stack*
4. *Create a walk-through of how you will recover the HumanityLink application system which has been compromised*

Be sure to include anything outside of this list that you think would thwart EatBrains in your environment.

2 Application Conceptual Design

2.1 Requirements

ID	Requirement
R001	Whilst resiliency will be built into the application wherever possible, the infrastructure supporting the application must also be highly resilient
R002	Wherever possible, limit the need for developers to be involved in configuring and maintaining the application stack
R003	Developers must be able to deploy code as rapidly as possible
R004	An army of robots will be built to carry out the terraforming effort
R005	The data from the robots is required to further enhance scientific understanding of the terraforming process
R006	A resilient 3-site architecture is required

2.2 Constraints

ID	Constraint
C001	AWS must be used, due to its selection in the previous challenge

2.3 Risks

ID	Risk	Mitigation
K001	A virus may encrypt SMB shares	See Security Incident and Response section
K002	Document Editor may run out of time and energy to properly document risks. Or the design in general.	None. Hope that by documenting this risk it manages to curry favour in some small way with the judges. Ummmm curry……

2.4 Assumptions

ID	Assumption
A001	The number of robots working on terraforming is 300,000 which is the mid point between the size of the British Army and US Army.[ii]
A002	GPS satellites are still operating

2.5 Amendments

The follow amendments are required to the infrastructure design in Challenge 1

ID	Amendment
M001	AWS Frankfurt (eu-central-1) would be used instead of AWS Dublin as it supports AWS Greengrass[iii]

3 Application Logical Design

The application developers want to focus on development rather than application stack management (**R002**), they need to make rapid, reliable and secure changes (**R003**), whilst manufacturing, tracking and communicating with a large number of devices (**R004**, **A001**), and they need to ingest a large volume of data from those devices (**R005, A001**). The infrastructure is also required to be deployed across three Availability Zones in AWS (**R006**) and as it must be highly resilient (**R001**), it will be available from at least two AWS regions.

HumanityLink will provide management of robot manufacturing, fleet management, and data analysis. Robot Design and Additional analysis will be performed using third party applications on end user desktops.

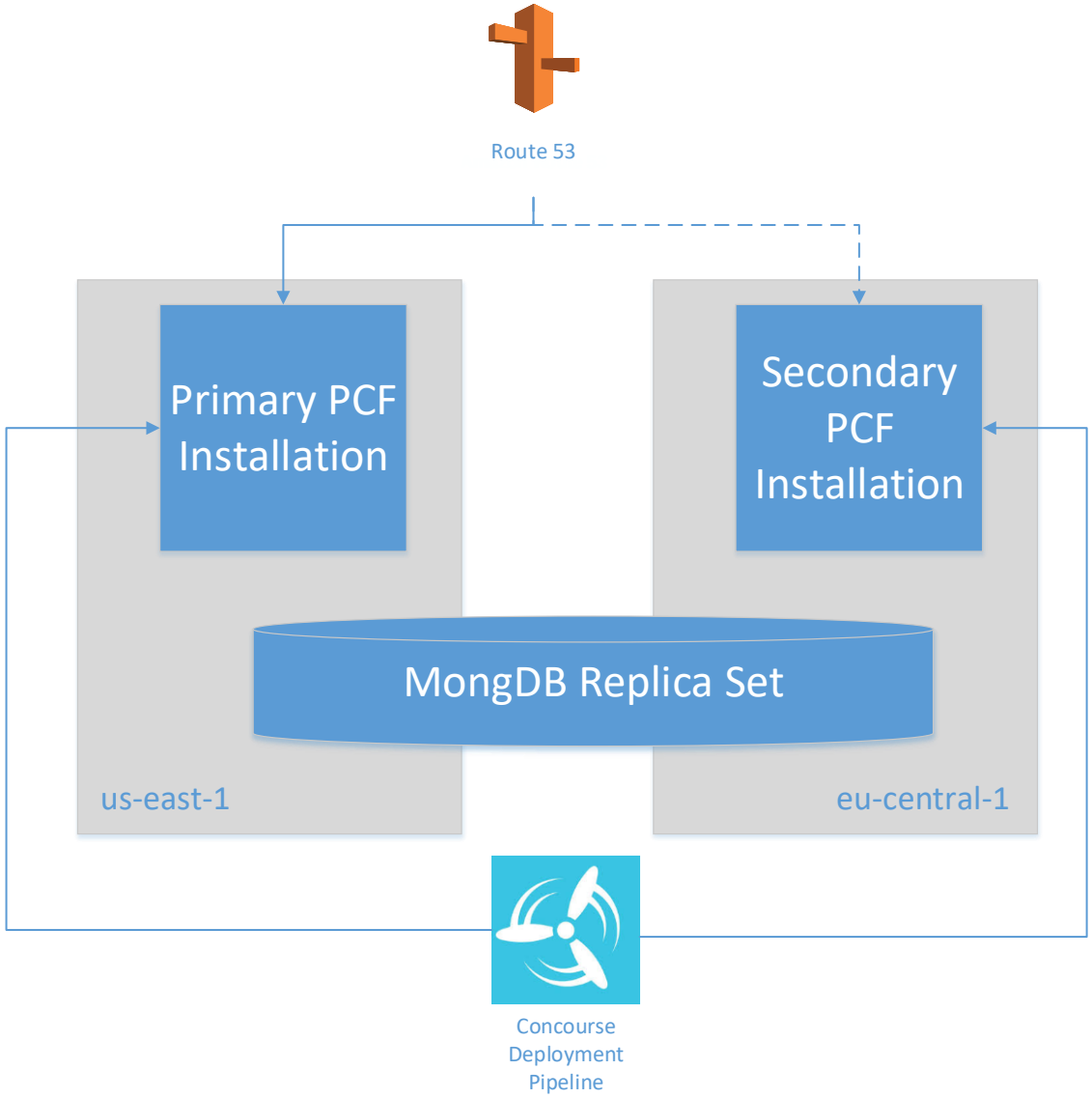

Figure 1: PCF High Level Diagram

To meet those requirements, Pivotal Cloud Foundry has been selected as Platform as a Service solution. The HumanityLink developers will be able to rapidly deploy to Cloud Foundry without having to think about scaling and securing the application stack. A separate Cloud Foundry installation will be setup in two regions, across multiple Availability Zones[iv][v], with the HumanityLink application deployed simultaneously to both regions[vi]. The applications in the CF installation in each

region will be fronted by AWS Elastic Load Balancers to disrupted load between Availability Zones for performance and resiliency. Failover between regions will be via DNS, using AWS Route53. As the databases in eu-central-1 will be read-only, all traffic will be routed to applications running out of the CF installation in us-east-1, using Failover Alias Resource Record Sets[vii] which point to the ELBs in each region. A health check will check that the ELBs are healthy, as well as a health check against the database in the region, and a failover will occur in Route53 if both are unhealthy.

As there will be a large volume of data (**R004**, **A001**), and the rapid learning from this data may require making changes to the data model, MongoDB has been selected as the data storage solution. MongoDB also has good integration with Cloud Foundry[viii]. MongoDB will be deployed in a 5 member replica set[ix], with 3 members in us-east-1 and 2 members in eu-central-1. The primary member will be located in us-east-1a. A replica set allows for automatic election and failover should an availability zone or entire region fail. The members in us-east-1 will have a priority of 1 and the members in eu-central-1 will have a priority of 0.5. This will prevent a failover to the members in eu-central-1 unless all members in us-east-1 are unavailable.

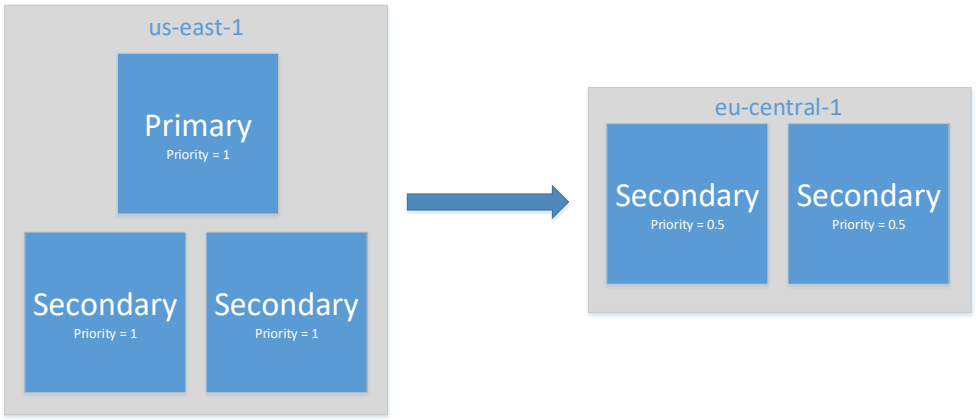

Figure 2: MongoDB Replica Sets

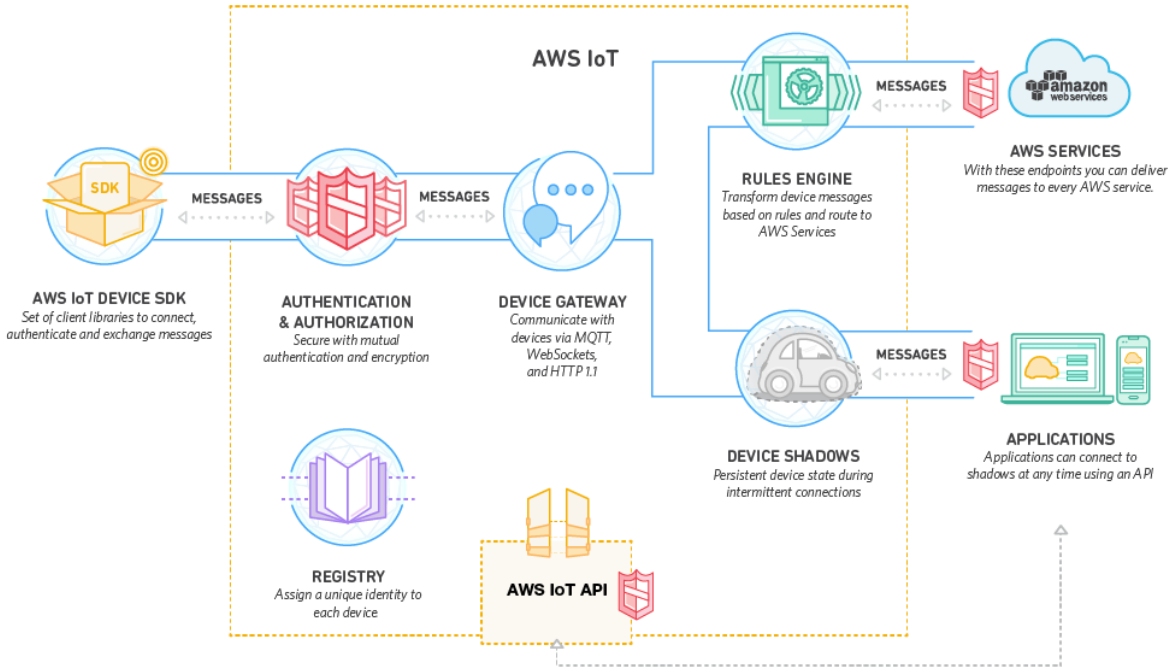

Figure 3; AWS IoT Overview[x]

AWS IoT will be used for managing the robot army, as it is custom built to track and command a large number of devices. The HumanityLink application will provide for analysis of data and control of robots via AWS IoT and its Rules Engine. Data will be captured such as temperature and GPS coordinates (**A002**), and commands triggered to robots depending on parameters such as temperature boundaries and/or geo fenced locations. Scientists will be able to view data in the HumanityLink application and export data for further analysis. They will then be able to make amendments in the HumanityLink application to adjust how robots carry out their action.

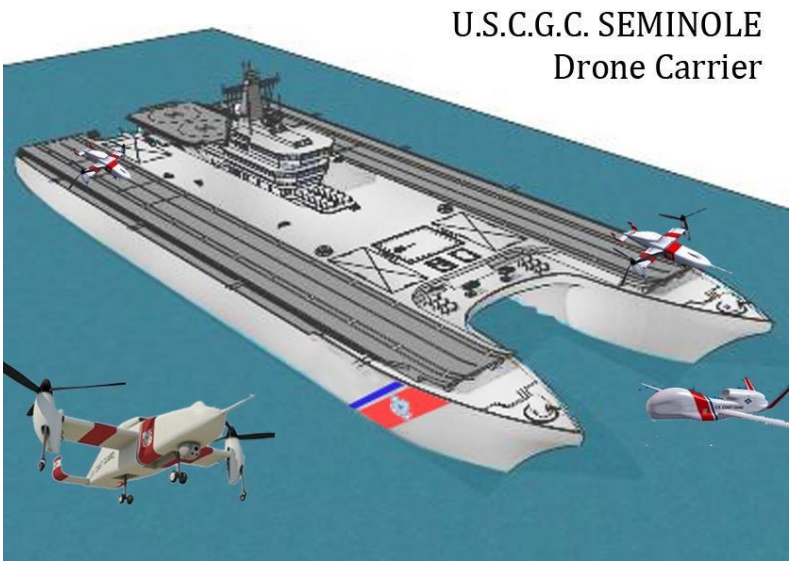

Figure 4; https://www.nextbigfuture.com/2015/01/drone-carrier-naval-ship-designs.html

Robots will operate from a number of re-purposed aircraft carriers, which will have a satellite uplink enabling connectivity to HumanityLink software.

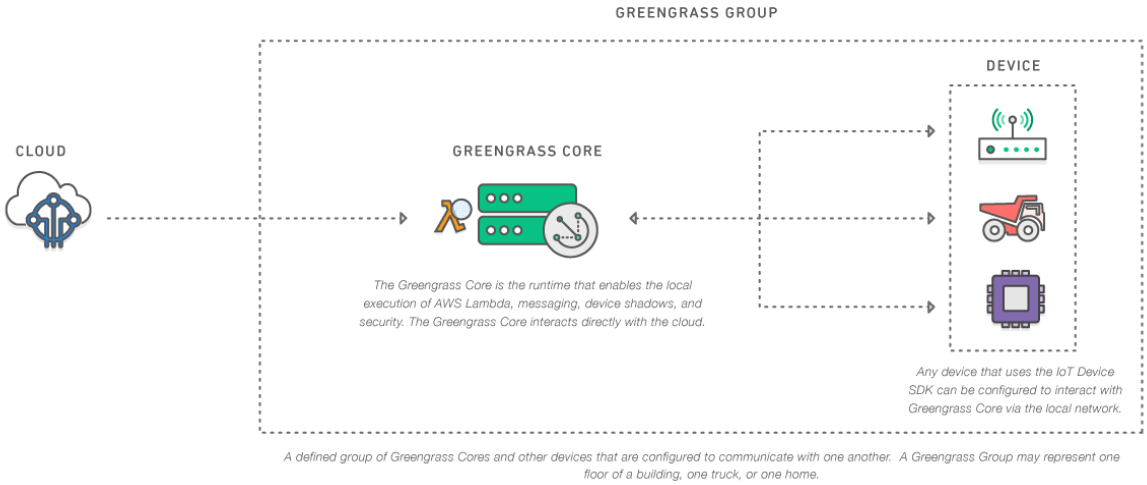

Figure 5: AWS Greengrass[xi]

AWS Greengrass[xii] will be used to support devices IoT whilst away from the carriers. Robots will operate in company[xiii] of robots up to 200 in size[xiv] with one of the robots being nominated as the Captain of the company. The Captain will carry the Greengrass device, deployed on a Raspberry Pi3[xv], communicating with the other robots in the Company using Wifi. Data would be cached whilst the robot company are away from the carriers and local Lambda functions would allow robots to make some decisions autonomously.

Concourse is recommended to the developers for the CI/CD tool, as it is developed by Pivotal and so therefore has very good integration with Cloud Foundry[xvi], allowing for rapid deployment (**R003).** A pipeline in Concourse will be created that will deploy to each of the Cloud Foundry installations at the two regions. Concourse also gets a vote as they publish live their own pipeline, which is very cool to look at and which can be found here; https://ci.concourse.ci/

Scientist Windows desktops will be provided by using AWS Workspaces. A Windows file server running on an EC2 instance will allow for the scientists to share data. Data will be replicated to the second AWS region using DFS-R. Applications would be managed and provided to the instances using AWS App Streams.

4 Application Physical Design and Implementation Guide

Due to time constraints and limited but slowly growing understanding of the technologies outlined in the Logical Design, a Physical Design and Implementation Guide for the application has been excluded from this Design Document.

5 Security Incident Response and Recovery

5.1 General Incident Description

The EatBrains virus has hit the infrastructure. It is understood to have been a zero day attack which was sent as an encrypted PDF within a Word document. One of the scientists opened this document on their Windows desktop and it has now infected the file server the scientists use for sharing data. The scientist are also sharing data between their workstations, so all workstations have now been compromised. EatBrains may also have infected devices in AWS' network.

5.2 Scenario 1; Single AZ Compromised

5.2.1 Incident Description

In this scenario, we assume the virus has only affected a single site. This single site is one of the Availability Zones in the us-east-1 region, AZ us-east-1a.

5.2.2 Incident Response

Alerts had been configured in Splunk to trigger when a large volume of denies are found in VPC flow logs, which is what initially brought the incident to our attention. Our zero trust model with security groups stopped the virus spreading past the file server and the scientists desktops. As AWS Security Groups are implemented in the AWS hypervisor, they are not susceptible to manipulation by the virus. The logs also showed us an entry being used for the phone home address. As we cannot add deny rules to Security Groups and we wish to still allow internet access, a Network ACL was created to block the IP address in use and hence stopping any remote control functionality. Using our data and their own logs, AWS could confidently say the virus had not infected network devices outside of the one AZ, but were not confident that the single AZ had not been compromised. They therefore shut off all network connectivity to the infected site without warning.

5.2.3 Recovery Plan, Part 1; Recover Service Availability

HumanityLink 2 is deployed using a CF manifest which specifies at least 3 instances for each application. CF is configured to map a CF Availability Zone to each AWS Availability Zone, and will therefore spread the instances across these AZs[xvii]. When one AZ fails, CF will restart an instance of the application in another AZ, whilst redirecting traffic to the remaining healthy instances. Therefore there should be no manual interaction required to recover service availability for the HumanityLink software in this scenario and minimal disruption to users.

As MongoDB has been deployed as multi member replica set with the primary member in us-east-1a, this will automatically failover to one of the other members in another AZ in us-east-1.

Terraforming operations would continue as AWS IoT is resilient across multiple AWS AZs.

As we assume all workspaces were compromised, they would be powered off.

We would use AWS Patch Manager to deploy a patch to all systems and update AWS Inspector to report on systems which remained unpatched to the vulnerability.

5.2.4 Recovery Plan, Part 2; Recover Compromised Systems

Once AWS had rectified the failed AZ and reconnected it to the network, CF would see the AZ as being back online and would start distributing instances as required. MongoDB would start replicating to the member in the recovered AZ.

The current AWS workspace instances would be powered off and rebuilt, with backups for the data drive restored from the previous backup. Applications would streamed back to the new workstations from App Stream.

5.2.5 Lessons Learned and Architecture Enhancements

To help prevent malware running in future we would deploy a application whitelisting application using a tool such as Microsoft AppLocker, coupled with a next generation anti malware solution such as Cylance. We would also remove admin rights from scientists to limit what they could run and stop them creating shares themselves. A firewall which can inspect traffic and proactively block malware phoning home such as Checkpoint or Palo Alto would also be deployed. All emails with encrypted attachments would be quarantined using a service such as Symantec's Email Security.cloud.

5.3 Scenario 2; Single AWS Region Compromised

5.3.1 Incident Description

In this scenario, we assume the virus has managed to affect an entire AWS region, us-east-1.

5.3.2 Incident Response

As with scenario 1, Splunk alerted us to the initial infection and the compromise to HumanityLink systems was limited by security groups. However, the virus managed to infect AWS network devices across multiple availability zones. To stop further spread of the virus to other AWS regions, AWS shut off all network connectivity to us-east-1 without warning.

5.3.3 Recovery Plan, Part 1; Recover Service Availability

The failover of the HumanityLink software should be almost seamless and happen in around a minute, based on a 30 second health check and a 30 second TTL. Health checks will fail for the load balancers and databases in us-east-1 so Route53 will start returning the failover record set which will redirect traffic to the load balancer in front of the CF applications in eu-central-1. MongoDB will elect one of the secondary members in eu-central-1 as a primary and start receiving write traffic. We would increase the priority of the members in eu-central 1 to 1.5 to stop the uncontrolled election of a member in us-east-1 once connectivity was restored.

Terraforming activity would not be immediately affected as they would operate independently using AWS Greengrass. However, when they returned to the aircraft carrier base, the Greengrass core devices would be unable to upload data and received any updated instructions. As it is assumed Greengrass groups are managed from an IoT Gateway in a single region, the Greengrass core would need to be reprogramed to use a Greengrass group in the eu-central-1 region. The terraforming effort would then recommence.

Using DNS we would repoint a cname to the DFS-R file server replica in eu-central-1 and restore this EC2 instance from a snapshot backup prior to the infect reaching the file server. and create new AWS Workspaces desktops in eu-central-1.

5.3.4 Recovery Plan, Part 2; Recover Compromised Systems

Once AWS re-enabled connectivity to us-east-1, MongoDB would start replicating to the members in this region. Once replication had caught up, and we were happy that the CF installation in us-east-1 was running fine, we would change the priority of the MongoDB members in eu-central-1 back to 0.5, which would result in one of the members in us-east-1 being elected as primary. This would

then make the health checks for the database layer in Route53 show as health and the primary record set would be returned, pointing to us-east-1, instead of the failover record set.

DFS-R replication would be reversed and new AWS workspaces instances built in us-east-1.

5.3.5 Lessons Learned and Architecture Enhancements

Everything from Scenario 1, plus a third party backup solution that would enable the replication to a separate region the backup of AWS Workspaces.

5.4 Scenario 3; Entire AWS Infrastructure Compromised

5.4.1 Incident Description

In this scenario, we assume the virus has managed to infect multiple AWS regions.

5.4.2 Incident Response

We were alerted and somewhat protected by our zero trust model, but the virus spread far quicker than anyone imagined. It infected AWS networking devices and managed to replicate between them, even across regions. We believe AWS took their entire infrastructure offline but it may have been the virus. AWS have sent carrier pigeons telling us their infrastructure is resilient to regions failing and have declared this an act of war. They are therefore not honouring any SLAs currently[xviii].

5.4.3 Recovery Plan, Part 1; Recover Service Availability

Service Availability would be lost for HumanityLink until AWS was restored. The robot army would act as in the last scenario.

5.4.4 Recovery Plan, Part 2; Recover Compromised Systems

Once AWS restored connectivity to its infrastructure, as we have made no changes, we would not need to make any changes to recover the HumanityLink application. We would need to rebuild the infected AWS Workspaces instances and restore the compromised file server from a snapshot backup.

5.4.5 Lessons Learned and Architecture Enhancements

Everything from scenarios 1 &2. So as to support HumanityLink availability even when AWS suffers a complete outage, it is suggested to make the following amendments;

- Use Cloudflare instead of Route53. This would enable DNS failover to environments outside of AWS without relying on an AWS service i.e. Route53
- Add a non AWS datacentre. This could be Pivotal Cloud Foundry running on VxRail in an alternate site in Frankfurt. VxRail is easy to deploy and scale, and there is a reference article for running PCF on top of VxRail[xix]. A MongoDB hidden member could be added at this site to the MongoDB replica set, enabling data replication.

6 Appendix A; References

[i] Atmosphere of Venus; Clouds
https://en.wikipedia.org/wiki/Atmosphere_of_Venus#Clouds

[ii] Compare Branches of the U.S. Military
http://us-military-branches.insidegov.com/
Total British Armed Forces
http://www.armedforces.co.uk/mod/listings/l0003.html

[iii] AWS Greengrass FAQ; Which AWS regions is AWS Greengrass service available in?
https://aws.amazon.com/greengrass/faqs/

[iv] High Availability in Cloud Foundry
https://docs.cloudfoundry.org/concepts/high-availability.html

[v] Reference Architecture for Pivotal Cloud Foundry on AWS
http://docs.pivotal.io/pivotalcf/1-8/refarch/aws/aws_ref_arch.html

[vi] UK Charity Raises Record Donations Powered by Cloud Foundry
https://content.pivotal.io/blog/uk-charity-raises-record-donations-powered-by-cloud-foundry

[vii] Configuring Active-Passive Failover by Using Amazon Route 53 Failover and Failover Alias Resource Record Sets
http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/dns-failover-configuring-options.html#dns-failover-failover-rrsets

[viii] MongoDB For Pivotal Cloud Foundry
https://pivotal.io/platform/services/data-management/mongodb

[ix] Replica Sets Distributed Across Two or More Data Centers
https://docs.mongodb.com/manual/core/replica-set-architecture-geographically-distributed/

[x] AWS IoT Features
https://aws.amazon.com/iot-platform/how-it-works/

[xi] What Is AWS Greengrass?
http://docs.aws.amazon.com/greengrass/latest/developerguide/what-is-gg.html

[xii] AWS re:Invent 2016: NEW LAUNCH! Introducing AWS Greengrass (IOT201)
https://www.youtube.com/watch?v=XQQjX8GTEko&feature=youtu.be&t=27m27s

[xiii] Company (military unit)
https://en.wikipedia.org/wiki/Company_(military_unit)

[xiv] AWS Greengrass Cloud API Limits
http://docs.aws.amazon.com/general/latest/gr/aws_service_limits.html#limits_greengrass

[xv] Preparing a Raspberry Pi for Greengrass
http://docs.aws.amazon.com/greengrass/latest/userguide/prepare-raspi.html

[xvi] Concourse vs.
http://concourse.ci/concourse-vs.html

[xvii] High Availability in Cloud Foundry
https://docs.cloudfoundry.org/concepts/high-availability.html

[xviii] AWS Customer Agreement; Section 13.3 Force Majeure
https://aws.amazon.com/agreement/

[xix] Pivotal Cloud Foundry on Dell EMC VxRail Appliance
https://storagehub.vmware.com/#!/vmware-vsan/pivotal-cloud-foundry-on-dell-emc-vxrail-appliance-1