Virtual Design Master

Season 5 Challenge 1

Version 1.0

Chris Porter

What Can Go Wrong With Terraforming Robots

Document History

Version	Date	Changes
1.0	05/07/2017	Promoted v0.3 to v1.0
0.3	04/07/2017	‘Completed’ Logical Design although it lacks sufficient links back to Conceptual Design and ‘Completed’ Physical Design. Added Terraform file to build environment. Added 1 diagram…., added very little humour.
0.2	03/07/2017	Expanded Logical Design, started Physical Design
0.1	02/07/2017	Document creation and first draft of Conceptual Design

Contents

1. Overview

1.1 Design Brief

We now must get ready to continue to deploy the newest version of the HumanityLink application. We know we want to spread our efforts out across the world as we begin the terraforming effort. Because most of our time have been spent preparing the earth for recolonization,

we are looking for your recommendations for building out the new version of the HumanityLink application. we have been so busy with this.

We are working on building an army of robots to carry out the terraforming efforts. These robots will be operated in shifts, to ensure there is a fleet running at all times. All robot maintenance is done while they are off shift. The new version of the HumanityLink application will add the features needed for the scheduling, operations, and maintenance of the robot fleet.

We need you to design and document a resilient 3-site architecture. We do not have limitations on hypervisor, hardware, networking stack, or applications. You will be working with many assumptions and will need to ensure your infrastructure design is able to handle the potential for unknown workloads (Hint: You should be able to scale your environment easily in any direction). Our primary concern is resiliency in as many layers as possible. The design and initial deployment will be on Earth and you can assume that Earth-like environmental conditions will be available wherever the design will be deployed.

Your design will need to clearly illustrate how you provide resiliency and be prepared to also defend how you made decisions on where your resiliency lies and what tradeoffs you have introduced when making those decisions.

1.2 Design Summary

In order to meet the requirements of the brief, this design provides an infrastructure platform using AWS. Using AWS provides a highly resilient, scalable, secure and reliable platform on which to build out the HumanityLink software, in multiple local sites and across global regions.

1.3 Scope

The design does not cover the application architecture of the HumanityLink software.

2 Conceptual Design

2.1 Requirements

ID	Requirement
R001	The infrastructure must be able to support applications running in a local N+2 site design i.e. across three sites
R002	The infrastructure must be deployed in multiple geographical regions spread across Earth
R003	The infrastructure must be able to support a large number of robots
R004	The infrastructure must be able to rapidly scale up, down, in and out. It is not required to do the hokey cokey however[i].
R005	The infrastructure must be available 24/7 to support the robot fleet, as even when off shift the robots are undergoing maintenance and shifts may overlap.
R006	To ensure the chance of disruption to the terraforming effort is reduced as much as possible, resiliency must be included in the design wherever possible.
R007	The infrastructure design must be flexible as at present the application stack is unknown
R008	Due to problems with rogue actors in the past, the design must be as secure as possible, particular in detecting and monitoring unauthorised access of our systems.
R009	Based upon Assumption A009, the environment must be able to handle the rapid deployment of updates at scale to the HumanityLink application stack.
R010	Based upon Assumption A010, the architecture must be resilient to external network attacks.

2.2 Constraints

ID	Constraint
C001	The infrastructure has to be built on Earth. Currently no other planets, space ships, space stations, space boats, space cars, space rockets, death stars, meteors, comets, asteroids or satellites are available.

2.3 Risks

ID	Risk	Mitigation
K001	Zombies may still be on earth and may attack datacentres and/or terraforming robots	Ensure datacentres use 2 factor biometric access controls which will prohibit zombie access. Use multiple data centres and regions to ensure availability if datacentres are compromised
K002	Terraforming as a process is still in its infancy and may result in local climate disruption, including extreme temperatures and flash flooding	Use multiple data centres and regions to ensure availability if datacentres are disrupted by climate changes.
K003	Robots may become sentient and turn on humans	Ensure robots programmed with Asimov's three laws. Ensure data

		centres have adequate physical security.
K004	Third party infrastructure providers may fail or go out of business	Use multiple data centres and regions to ensure availability if datacentres are disrupted by local failures. See A004 for business failure mitigation.

2.4 Assumptions

ID	Assumption
A001	There is a mobile network on earth, but it is unreliable so robots are not guaranteed to be online all the time
A002	All current AWS regions and services are available
A003	Terraforming effort is currently focused in Northern Europe and North America but will expand to other regions.
A004	Any infrastructure provider used by HumanityLink would be considered as a critical global infrastructure and would therefore be supported by the new government if the provider were to fail for financial reasons
A005	Sites are defined as being different datacentres that are far enough apart so as to avoid being on the same flood plain but close enough that they can be connected using a low latency high bandwidth network.
A006	There are no regulatory requirements which would require data protection or long term retention of data for audit purposes.
A007	The HumanityLink application stack will use both servers and containers.
A008	There is currently no requirement for IPv6.
A009	Terraforming with robots, indeed, terraforming at all is still a new and uncertain process[ii]
A010	A strange combination of people have survived who quite liked zombies, really hate terraforming and are mid level hackers.
A011	Github is available because they had really good backups and we've managed to restore it on AWS. Honest. And without it, downloading and using tools like terraform is really hard.
A012	Our billionaire philanthropist is still funding things so billing is not a concern.

A013	

3 Logical Design

The following design decisions have been made to meet the conceptual design;

Design Decision ID	DD001
Design Decision	Infrastructure will be built using a hyperscale cloud provider
Justification	Using a cloud provider such as AWS or Azure will allow the infrastructure to easily scale in any direction, will allow for additional regions to be added quickly and consistently and will ensure multiple levels of resiliency at data centre layer.
Conceptual Design Reference	R002, R004, R006

Design Decision ID	DD002
Design Decision	AWS will be selected as the cloud provider for the infrastructure
Justification	• AWS provides multiple regions across the global which support at least 3 availability zones. • AWS availability zones have multiple levels or resiliency • AWS have a wide range of instance types which will allow the infrastructure to support high performance requirements. • They have a rich set of services which could be leveraged by HumanityLink, such as AWS IoT.
Conceptual Design Reference	R001, R002

Design Decision ID	DD003
Design Decision	Infrastructure will be available from two AWS regions; us-east-1 and eu-west-1
Justification	Infrastructure availability from more than one region allows for failover in the event of issues occurring in one DC. These two regions have been selected specifically as they have 3 or more availability zones. They are on separate tectonic plates in case there are some *really* adverse side effects from the terraforming process
Conceptual Design Reference	R001, K001, K002, K004

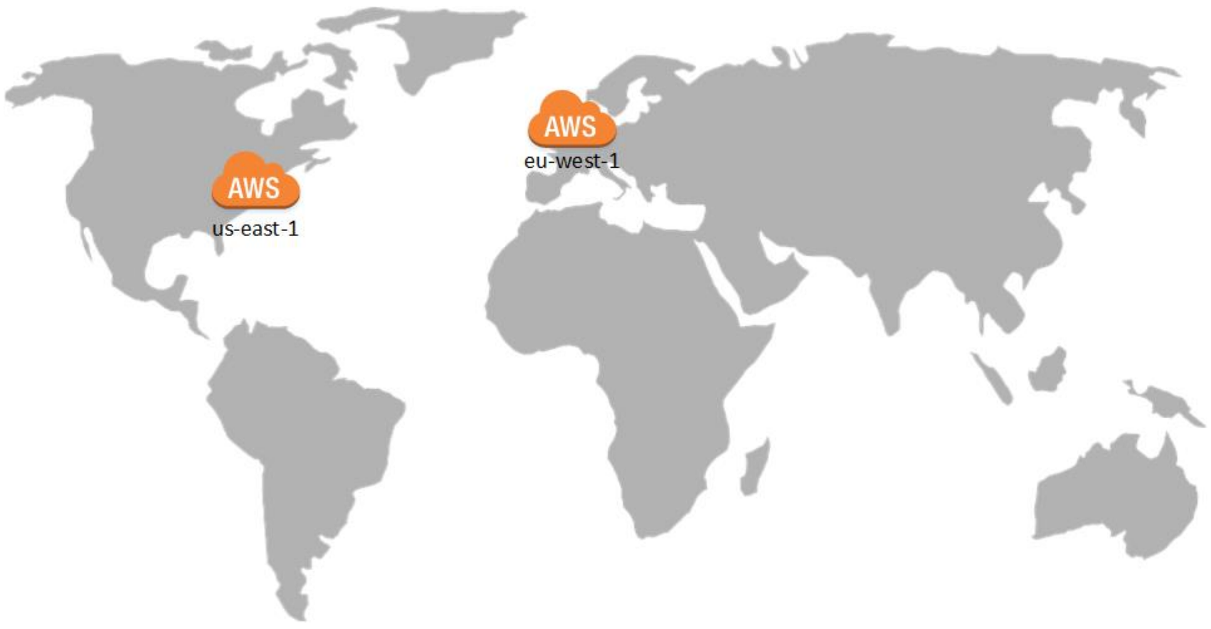


Figure 1; AWS Regions

3.1 AWS Account Overview

To help ensure separation between Production and Non Production systems, separate AWS accounts will be setup for each of these. A third account will be setup which will act as the Master account and will be where security logs are stored. Access to the master account will be very limited to ensure the integrity of security logs.

AWS Organisations will be used to manage the three accounts. AWS Organisations will allow us to set consistent account configuration across all accounts and to limit available services in each account to only those which are required[iii].

3.2 Networking

We will create a VPC in the Prod and Non Prod accounts. By creating our own VPCs we can control the CIDR allocation ensuring that there are no clashes from the CIDR range which AWS automatically creates for the Default VPC. A VPC will not be required in the Master account as it will not be used for workloads.

An additional VPC will be created in the Prod account for shared services. This VPC will then be connected to the other VPCs using VPC peering. This will allow for us to avoid duplicating shared services such as authentication, bastion hosts, monitoring etc for Prod and Non Prod. By using VPC peering, whilst the Shared Services VPC can talk to the other VPCs, the Prod and Non Prod VPCs are still isolated from one another as transitive routing via a VPC is not supported by AWS.

IPv6 will not be enabled on VPCs but this can be enabled later for existing VPCs[iv]. VPC Tenancy will be set to Default (i.e. shared tenancy), rather than Dedicated as this cannot be changed post creation and some AWS services and features do not support Dedicated Tenancy. If Dedicated Tenancy is required, this can still be set when launching an EC2 instance.

Each VPC will be split into two types of subnets, Public and Private. A subnet of each type will be allocated to each availability zone. The only difference between these subnets is that the public subnet is able to route directly to the internet, whereas the private subnet cannot. Workloads that are put in the public subnet will be limited as they will be directly accessible from the internet.

To enable replication between the two regions, software VPN appliances will be deployed in each VPC to create a VPN connection between VPCs in different regions.

For internal DNS, NTP and DHCP, we will use AWS' native services provided with a VPC. For external DNS we will use Route 53. This will allow resiliency across regions by using a routing policy which routes traffic to both regions.

For internet access, an internet gateway will be attached to each VPC. A NAT Gateway will be deployed in each availability zone to allow instances in the private subnets to connect to the internet. By deploying a NAT gateway in each AZ, we ensure connectivity is not disrupted for instances in other AZs.

3.3 Security

A zero trust model will be implemented using AWS Security Groups. These will be used to logically separate different groups of servers and only allow necessary communication between each group. Wherever possible we will use rules in Security Groups which reference other Security Groups, rather than specific IP addresses. This greatly helps the manageability of security groups as they require less updating when IP addresses change. VPC Flow Logging will be enabled to capture denied traffic for troubleshooting and to alert on abnormal behaviour.

AWS IAM will be used to apply a zero trust model to administrative access. This begins by the use of separate AWS accounts for Prod and Non Prod. Even if an administrator gains full access in the non prod account, they cannot make changes which will affect the Prod account. IAM policies will grant administrators and services only the rights they need to perform their job. By using tags, we can ensure very granular access controls, limiting access to IAM users down to specific EC2 instances. All access to the system will required the use of MFA. An IAM Password Policy will be applied which will be based upon the latest NIST Guidelines; specifically, password complexity will not be enforced and passwords will not expire. NIST research shows that password complexity and regular password expiry make users pick poor passwords. By combining a lengthy password which users are more likely to remember with two factor authentication we ensure strong security without frustrating users.

Several AWS logging services will be enabled to enhance the security of the environment. All logging services will use S3 buckets stored in the Main Account, to limit the potentially for tampering. AWS Cloudtrail will be enabled to provide a log of all changes made via the API and the console, which can be reviewed to understand what happened if unauthorised changes are made. AWS Config will be enabled to track the state resources in AWS, and with Config Rules we will alert on unauthorised changes being made. Some of the config rules we use are the following;

- Track changes to IAM groups and policies, specifically to alert on unauthorised privileged escalation
- Track security groups allowing unrestricted SSH traffic
- Track CloudTrail being enabled or disabled.

To collate all logging into a central place and provide search and analysis of those logs, Splunk will be deployed in the environment. Splunk has rich integration with AWS and supports the analysis of a wide range of AWS log types.

Encryption at rest is not required as there are no data protection regulations to meet and we consider AWS datacentres to be under our control. Also for the same reasons encryption in transit for communication with the VPC is not required. However, encryption in transit from the internet to instances in public subnets will be required to stop interception of application traffic, to meet requirement R010. AWS WAF will be used with ALBs to protect against common internet attacks such as SQL injection.

AWS Inspector will be used to help check EC2 instances for known vulnerabilities and poor configuration. The AWS Inspector agent will be installed on all EC2 Instances by being included in a custom AMI.

To access the environment remotely whilst limiting what is exposed to the internet, bastion hosts will be used. These will be placed in a public subnet and access restricted by security groups to specific IP addresses on ports 22 and 3389 only.

3.4 Instances

Scheduled snapshots will be taken of EC2 instances using the EBS Snapshot Scheduler[v]. The Snapshot Scheduler uses tags to determine which instances should be backed up via snapshot and how long to retain those snapshots for.

To ensure EC2 instances are kept up to date, AWS Patch Manager will be used.

Instances will be tagged with the following tags to allow for security policies to limit access, help identify the resource's purpose and enable backups;

- Name
- Environment (Production, Staging, Development, Test)
- Project
- Owner
- scheduler:ebs-snapshot

3.5 Configuration Management

Terraform will be used to build and maintain the environment. With Terraform we will use modules to define the infrastructure as a template which will then be applied to both Prod and Non Prod Accounts, to ensure consistency between all environments. Terraform allows us to adopt an 'infrastructure as code' approach, so that changes can be version controlled and repeated consistently.

3.6 Availability

To ensure availability between availability zones, all supporting infrastructure will be deployed across 3 AZs. Also, for the HumanityLink application, it should make use of Application Load Balancers and Auto Scaling Groups. The ALB would be configured to use EC2 instances from each availability zone.

Cloudwatch will be used for monitoring. Detailed monitoring will be enabled for all EC2 instances.

3.7 Application Delivery

Design Decision ID	DD004
Design Decision	To provide continuous delivery of updates to the HumanityLink application, a deployment pipeline should be used[vi]. This would consist of a source code

	repository, a continuous integration service, an artefact repository and/or container registry, a release management tool and a scheduler/orchestrator (for containers only). A suggested stack for this would be GitHub, Jenkins, Artifactory, Docker Registry, Spinnaker, and Kubernetes[vii].
Justification	As the HumanityLink software is continually enhanced to react to the rapid pace of scientific understanding in using terraforming, software updates must be deployed in a robust fashion without introducing barriers which may result in prolonged periods of sub optimal terraforming.
Conceptual Design Reference	R009

Time constraints and the (current) skill set of the infrastructure architect meant that this design does not include any further detail on the Application Delivery design decision above. The next steps would be to perform a PoC on these tools and their integration, then amend where necessary before performing a wider rollout.

4 Physical Design

4.1 AWS Organizations

The following configuration is required for AWS Organizations;

AWS Organisation Setup Type	Enable all Features

4.2 AWS Accounts

The following configuration is required for AWS Accounts in AWS Organisations;

Account Name	**Email Address**	**Master Account Access IAM Role**
Master	AWSAccount-Master@humanitylink.org	N/A
Production	AWSAccount-Production@humanitylink.org	ProductionOrganizationAccountAccess Role
Pre Production	AWSAccount-PreProduction@humanitylink.org	PreProductionOrganizationAccountAcc essRole

4.3 AWS IAM

4.3.1 Password Policy

The following IAM password policy will be applied in all accounts;

Minimum Password Length	16
Require at least one uppercase letter	No
Require at least one lowercase letter	No
Require at least one number	No
Require at least one non-alphanumeric character	No
Allow users to change their own password	Yes
Enable password expiration	No
Prevent password reuse	N/A
Password expiration requires administrator reset	N/A

4.3.2 Customer Managed IAM Policies

The following Customer Managed IAM Policy will be created and named ‘RequireMFA’. This will ensure IAM users must authenticate with an MFA token for all actions[viii].

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["*"],
    "Resource": ["*"],
    "Condition": {"NumericLessThan": {"aws:MultiFactorAuthAge": "3600"}}
  }]
}
```

4.3.3 Groups

The following IAM Groups will initially be created. For operators this will allow them to create and managed EC2 instances and use CloudWatch for diagnostics.

Group Name	AWS Account(s)	Description	Attached Policies
Administrators	Master	Access only to Master account	AdministratorAccess (AWS Managed) RequireMFA (Customer Managed)
Administrators	Production, Pre Production	Full access Production and Pre Production accounts but not to Master account	AdministratorAccess (AWS Managed) RequireMFA (Customer Managed)
Power Users	Production, Pre Production	Full access without ability to edit IAM configuration to both Production and Pre Production accounts but not to Master account	PowerUserAccess (AWS Managed) RequireMFA (Customer Managed)
Operators	Production, Pre Production	Full EC2 and CloudWatch access to both Production and Pre Production accounts but not to Master account	AmazonEC2FullAccess (AWS Managed) CloudWatchFullAccess (AWS Managed) RequireMFA (Customer Managed)
Developers	Pre Production	Full EC2 and CloudWatch access but only to the Pre Production account	AmazonEC2FullAccess (AWS Managed) CloudWatchFullAccess (AWS Managed) RequireMFA (Customer Managed)
Service Account(s)	Production, Pre Production	Access where required based on least privilege access	Account Specific

4.3.4 Users

Users will be setup by a member of the Administrators group so their MFA device can be configured. All users will be given passwords and/or API keys.

4.4 S3 Buckets

The following S3 buckets are required;

Bucket Name	AWS Account	Notes
humanitylink-cloudtrail- us-east-1	Master	CloudTrail from multiple accounts can be directed to the same S3 bucket[ix]
humanitylink-cloudtrail- eu-west-1	Master	CloudTrail from multiple accounts can be directed to the same S3 bucket
humanitylink-awsconfig-us-east-1	Master	AWS Config from multiple accounts can be directed to the same S3 bucket[x]
humanitylink-awsconfig- eu-west-1	Master	AWS Config from multiple accounts can be directed to the same S3 bucket

4.5 VPCs

Name Tag	AWS Account	Region	IPv4 CIDR Block	IPv6 CIDR Block	Tenancy

vpc_production	Production	us-east-1	10.1.0.0/16	No IPv6 CIDR Block	Default
vpc_preproduction	Pre Production	us-east-1	10.2.0.0/16	No IPv6 CIDR Block	Default
vpc_sharedservices	Production	us-east-1	10.3.0.0/16	No IPv6 CIDR Block	Default
vpc_production	Production	eu-west-1	10.4.0.0/16	No IPv6 CIDR Block	Default
vpc_preproduction	Pre Production	eu-west-1	10.5.0.0/16	No IPv6 CIDR Block	Default
vpc_sharedservices	Production	eu-west-1	10.6.0.0/16	No IPv6 CIDR Block	Default

4.6 Routing Tables

The following routing tables are required. Additional Routes are in addition to the default local route and any routes added via VPC Peering.

Name	**Additional routes**
rt_public	Destination; 0.0.0.0/0, Target; IGW
rt_private_a	Destination; 0.0.0.0/0, Target; NAT Gateway in AZ A
rt_private_b	Destination; 0.0.0.0/0, Target; NAT Gateway in AZ B
rt_private_c	Destination; 0.0.0.0/0, Target; NAT Gateway in AZ C

4.7 Subnets

The following is an example of the subnets that will be created in the Production VPC in the us-east-1 region. The same subnet setup would be created for all accounts

Name	**Availability Zone**	**Subnet**	**Route Table**
sn_public_a	us-east-1a	10.1.0.0/19	rt_public
sn_public_b	us-east-1b	10.1.32.0/19	rt_public
sn_public_c	us-east-1c	10.1.64.0/19	rt_public
sn_private_a	us-east-1a	10.1.128.0/19	rt_private_a
sn_private_b	us-east-1b	10.1.160.0/19	rt_private_b
sn_private_c	us-east-1c	10.1.192.0/19	rt_private_c

4.8 Further Physical Design

Due to time constraints, further physical design is not included.

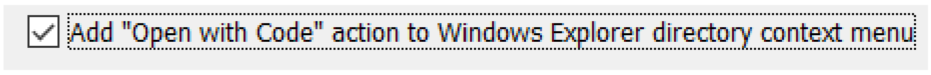
5 Implementation Guide

5.1 Installing Terraform

1. Download terraform from https://www.terraform.io/downloads.html
2. Extract to your computer
3. If in Windows, add the path to the extracted terraform executable to PATH variable
4. Terraform can now be run from the command line

5.2 Installing Visual Studio Code

1. Download Visual Studio Code (VSC) from https://code.visualstudio.com/download
2. Run the installer, tick to add an option to open folders with VSC

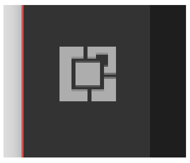


✓ Add "Open with Code" action to Windows Explorer directory context menu

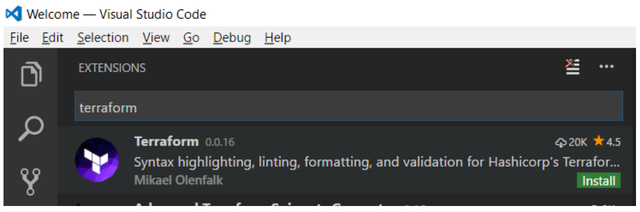
3. Installer will complete and VSC can now be opened from the Start Menu.

5.3 Install the Terraform extension for VSC

1. Open VSC
2. Click the extension button from the left hand menu



3. Type terraform in the search bar
4. Click the green 'Install' button next to the extension from Mikael Olenfalk



Welcome — Visual Studio Code

File Edit Selection View Go Debug Help

EXTENSIONS

terraform

Terraform 0.0.16 20K ★4.5
Syntax highlighting, linting, formatting, and validation for Hashicorp's Terrafor...
Mikael Olenfalk
Install

5. That's it, you'll now get syntax highlighting and automatic formatting in Terraform files

5.4 Basic Terraform File

The following Terraform file will create the production VPC and the first public and private subnets in the us-east-1 region. It also creates the internet gateway, NAT gateway and route tables that configure the subnets to use the correct gateways.

It should be run using the following command;

```
$ terraform apply -var aws_access_key=1234 -var aws_secret_key=5678
```

Main.tf

```
# Variables
variable "aws_access_key" {
  type = "string"
}

variable "aws_secret_key" {
  type = "string"
}

# Provider Setup
provider "aws" {
  access_key = "${var.aws_access_key}"
  secret_key = "${var.aws_secret_key}"
  region     = "us-east-1"
}

# VPC Creation

resource "aws_vpc" "vpc_production" {
  cidr_block       = "10.1.0.0/16"

  tags {
    Name = "production"
  }
}

# NAT Gateway EIP

resource "aws_eip" "eip_nat" {
  vpc      = true
}

# Internet Gateway

resource "aws_internet_gateway" "igw" {
  vpc_id = "${aws_vpc.vpc_production.id}"

  tags {
    Name = "igw"
  }
}

# NAT Gatway in AZ A

resource "aws_nat_gateway" "ngw_a" {
  allocation_id = "${aws_eip.eip_nat.id}"
  subnet_id     = "${aws_subnet.sn_public_a.id}"
}

# Route Table for all Public Subnets

resource "aws_route_table" "rt_public" {
  vpc_id = "${aws_vpc.vpc_production.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_internet_gateway.igw.id}"
  }

  tags {
    Name = "rt_public"
  }
}

# Route Table for all Private Subnet in AZ A

resource "aws_route_table" "rt_private_a" {
  vpc_id = "${aws_vpc.vpc_private.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_nat_gateway.ngw_a.id}"
  }

  tags {
    Name = "rt_private_a"
  }
}

# Public Subnet in AZ A

resource "aws_subnet" "sn_public_a" {
  vpc_id     = "${aws_vpc.vpc_production.id}"
  cidr_block = "10.1.0.0/19"
  availability_zone = "us-east-1a"

  tags {
    Name = "sn_public_a"
  }
}

# Public Subnet in AZ A Route Table Association
```

```
resource "aws_route_table_association" "rt_public_a" {
  subnet_id      = "${aws_subnet.sn_public_a.id}"
  route_table_id = "${aws_route_table.rt_public.id}"
}

# Private Subnet in AZ A

resource "aws_subnet" "sn_private_a" {
  vpc_id     = "${aws_vpc.vpc_production.id}"
  cidr_block = "10.1.128.0/19"
  availability_zone = "us-east-1a"

  tags {
    Name = "sn_private_a"
  }
}

# Private Subnet in AZ A Route Table Association

resource "aws_route_table_association" "rt_private_a" {
  subnet_id      = "${aws_subnet.sn_private_a.id}"
  route_table_id = "${aws_route_table.rt_private_a.id}"
}
```

6 Appendix A; References

[i] Hokey cokey
https://en.wikipedia.org/wiki/Hokey_cokey

[ii] Terraforming
https://en.wikipedia.org/wiki/Terraforming

[iii] Accessing and Administering the Member Accounts in Your Organization;
http://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_accounts_access.html

[iv] Migrating to IPv6;
http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-migrate-ipv6.html#vpc-migrate-ipv6-cidr

[v] EBS Snapshot Scheduler
https://aws.amazon.com/answers/infrastructure-management/ebs-snapshot-scheduler/

[vi] Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation (Addison-Wesley Signature) by Jez Humble and David Farley

[vii] Why (and How) Spinnaker and Kubernetes Work Together Seamlessly
http://blog.armory.io/why-spinnaker-and-kubernetes-work-together-seamlessly/

[viii] Example 1: Granting Access After Recent MFA Authentication (GetSessionToken)
http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_example-policies.html#ExampleMFAforIAMUserAge

[ix] Turning on CloudTrail in Additional Accounts
http://docs.aws.amazon.com/awscloudtrail/latest/userguide/turn-on-cloudtrail-in-additional-accounts.html

[x] Aggregating AWS Config log files from multiple AWS accounts
https://www.linkedin.com/pulse/aggregating-aws-config-log-files-from-multiple-nazim-akbarov