



Virtual Design Master

CHALLENGE 3

CENTRAL COMMAND

ADAM POST

ADAMJPOST@GMAIL.COM

@SEMI_TECHNICAL 

Table of Contents

| | |
|---|----|
| <i>Operations Briefing</i> | 3 |
| <i>Overview</i> | 3 |
| <i>Objective Summary</i> | 3 |
| <i>Scope and Requirements</i> | 4 |
| <i>Project Scope</i> | 4 |
| <i>Project Parameters</i> | 4 |
| <i>Requirements</i> | 4 |
| <i>Constraints</i> | 4 |
| <i>Assumptions</i> | 4 |
| <i>Risks</i> | 4 |
| <i>Technical Requirements</i> | 5 |
| <i>Availability</i> | 5 |
| <i>Manageability</i> | 5 |
| <i>Performance</i> | 5 |
| <i>Recoverability</i> | 5 |
| <i>Security</i> | 5 |
| <i>HumanityLink Central Command</i> | 6 |
| <i>Conceptual Design</i> | 6 |
| <i>Infrastructure Architecture</i> | 6 |
| <i>Infrastructure Management</i> | 6 |
| <i>Code Repository</i> | 6 |
| <i>Configuration Management</i> | 6 |
| <i>Content Distribution</i> | 6 |
| <i>Deployment Pipeline</i> | 6 |
| <i>Conceptual Diagram</i> | 7 |
| <i>Logical Design</i> | 8 |
| <i>Infrastructure Architecture</i> | 8 |
| <i>Management Architecture</i> | 10 |
| <i>Logical Diagram</i> | 12 |
| <i>Service Associations</i> | 12 |
| <i>Solution Stack</i> | 13 |
| <i>Physical Design</i> | 14 |

| | |
|--|----|
| Physical Diagram..... | 14 |
| Configuration Tables..... | 14 |
| Deployment and Management..... | 15 |
| HumanityLink Distributed Modeling | 15 |
| Summary | 15 |
| Procedure Overview | 15 |
| Logical Diagram | 16 |
| Physical Configuration Table..... | 17 |
| Infrastructure Deployment Plan..... | 18 |
| Operating System Configuration Plan..... | 35 |
| Application Deployment Pipeline | 36 |
| HumanityLink Fleet Management..... | 38 |
| Summary | 38 |
| Procedure Overview | 38 |
| Logical Diagram | 40 |
| Physical Configuration Table..... | 41 |
| Infrastructure Deployment Plan..... | 42 |
| Operating System Configuration Plan..... | 63 |
| Application Deployment Pipeline | 64 |
| Operational Procedures | 66 |
| Preparation Tasks | 66 |
| Managing Terraform..... | 69 |
| Enable Terraform Remote State..... | 69 |
| Manage and Modify Terraform Code | 70 |
| Managing Chef Cookbook Versions | 72 |
| Managing CodePipeline Deployment..... | 74 |
| References..... | 75 |

Operations Briefing

Overview

Having operationally recovered from the recent highly-impactful, publicly-visible outbreak of EatBrains, HumanityLink Corporation has once again focused its attention on plans for expansion. As part of this new strategy, leadership has decided to aggressively push forward beyond previously defined growth goals. In doing so, leadership believes that “a new precedent for Excavation industry excellence will be established, and all future efforts will be held to this standard”.

There are multiple benefits associated with this approach. First, HumanityLink will regain any progress lost as a result of downtime and quickly be on pace to achieve defined excavation objectives. Secondly, the gesture will serve as a much-needed public demonstration of company strength that will help repair perception of the organization and its leadership team.

To support this initiative, some modification to the existing infrastructure architecture will be required. Although the infrastructure hosting HumanityLink Distributed Modeling and HumanityLink Fleet Management was originally designed to be massively-scalable, a decision has been made to limit the impact whole-tier or whole-environment failure can have on the excavation initiative.

It is for this reason that additional, dedicated HLDM and HLFM environments will be needed to support excavation operations in new areas. These environments will align with the original architectures, but additional capabilities are needed to centrally provision and manage these new deployments **(R01)**. Creating this management infrastructure, planning for automated deployments and providing administrative guidance should be the focus of the architecture team throughout this initiative.

Because of limited human resources and existing strain on the technical team, the new management solution must be operationally efficient **(C02)**. In addition to technical optimizations, regular bowling nights will be scheduled to increase morale, and trust falls will be leveraged extensively to enhance team confidence and cohesion. To celebrate top performers, a prestigious parking space will be awarded. In this way, management will have fully addressed the issue of resource insufficiency.

Finally, reconnaissance performed by HumanityLink Scout shows that many of the regions targeted for terraforming have not yet been cleared of danger. While these threats are not relevant to the excavation and repair fleets, it is desired to keep all processing and control infrastructure within regions that have been secured militarily. Otherwise, management trusts the architecture team to make suitable decisions in support of stated objectives. **(A01, A02, A03)**

Objective Summary

- Provide a scalable central control environment that can both deploy and manage new HLDM and HLFM environments **(R01)**
- Account for addition of very distant or dangerous sites that cannot be actively manned **(A01-03)**
- Automate deployment of cloud infrastructure associated with HLDM / HLFM expansion **(R03)**
- Provide a mechanism for controlling operating system and application configuration **(R05)**
- Implement a continuous deployment model for HLDM and HLFM applications **(R02)**
- Ensure all modifications to code and configuration are tracked centrally **(R04)**
- Create procedure to assist operations staff with administering configuration and code versioning **(RM03-RM04)**

Scope and Requirements

Project Scope

This project focuses on constructing a centralized infrastructure from which new HumanityLink Distributed Modeling and HumanityLink Fleet Management environments can be deployed and managed (**R01**). Accommodations for very remote and/or dangerous Excavation sites are included (**R07**). Continuous deployment will also be instituted, increasing the overall agility of the HumanityLink operation (**R02, R03**). Items not listed in the objective summary section are out of scope for this project.

Project Parameters

Requirements

| ID | Description |
|-----|--|
| R01 | Create a centralized provisioning and management environment for HumanityLink |
| R02 | Institute a continuous deployment model for Humanitylink releases |
| R03 | Automate new site infrastructure deployment for HLDM/HLFM environments |
| R04 | Version control for infrastructure and application code is required |
| R05 | Provide configuration management to assist with instance configuration consistency |
| R06 | Utilize managed or platform-integrated solutions, wherever possible |
| R07 | A mechanism for improving performance for distant sites must be provided |

Constraints

| ID | Description |
|-----|--|
| C01 | Tools selected should be open source to align with existing HumanityLink stack |
| C02 | Sufficient manpower to operate a non-centralized solution does not exist |

Assumptions

| ID | Description |
|-----|---|
| A01 | HumanityLink may need to be deployed to a dangerous zone |
| A02 | HumanityLink may need to be deployed to an off-earth location |
| A03 | Production operations will be centralized in the US due to increased defense measures |
| A04 | CloudFront will be used to increase performance for remote or off-earth deployments |

Risks

| ID | Description | Mitigation |
|-----|--|--|
| r01 | Excavation sites are dependent on WAN connectivity | Use of Redundant WAN connections mitigates this risk. It is not possible to deploy control infrastructure in a dispersed manner and meet overall project requirements. |
| r02 | Excavation sites may still be dangerous to humans | Control infrastructure will be placed centrally and new environments will be deployed remotely using the control environment. These choices mitigate the risk. |
| r03 | Solution leverages a single region for production | Multiple availability zones are used, providing a 99.95% SLA. DR approach proposed in original solution design can be used to increase this figure further. |

Technical Requirements

Availability

| ID | Description |
|------|--|
| RA01 | System must make use of highly-available hosted solutions, wherever possible |
| RA02 | Multiple instances must be deployed per-service across multiple availability zones |
| RA03 | No code updates are permitted to completely impact the service being hosted |
| RA04 | Single instances must be protected by an automatic restart mechanism |

Manageability

| ID | Description |
|------|--|
| RM01 | Leverage automation tools to initially deploy a new HLDM environment |
| RM02 | Leverage automation tools to initially deploy a new HLFM environment |
| RM03 | Account for ongoing management and updates to HLDM environments and code |
| RM04 | Account for ongoing management and updates to HLFM environments and code |
| RM05 | Implement continuous configuration management |
| RM06 | Create operational procedures associated with managing version control |
| RM07 | Include every infrastructure layer within scope of automation and management |
| RM08 | Deployment of new code versions must be administratively approved |
| RM09 | Systems must have limited access to the internet to obtain system updates |

Performance

| ID | Description |
|------|--|
| RP01 | Sizing must accommodate management of several new HLDM/HLFM environments |
| RP02 | Uploads and downloads of data to and from the system must be accelerated |

Recoverability

| ID | Description |
|------|--|
| RR01 | All Central Command instances must be backed up once per-24H |
| RR02 | In a failure situation, all Central Command components must be recoverable in 1H |

Security

| ID | Description |
|------|--|
| RS01 | Management components must be restricted to only access required |
| RS02 | Network access inbound must be limited to required management and SSH services |

HumanityLink Central Command

Conceptual Design

Infrastructure Architecture

A centralized environment will be created dedicated to HumanityLink Central Command, which will perform critical provisioning and management operations for existing and new deployments. This environment will be separate from the application environments that are being managed, but sufficient connectivity will be enabled so required management operations can be performed. In support of projected growth goals, components will be sized to allow addition and management of multiple new HLDM and HLFM environments. **(R01, R03)**

Infrastructure Management

New environments will be provisioned using a declarative infrastructure-as-code tool, which will efficiently perform deployment operations as well as retain information regarding environment state. The ability to simply describe future-state relieves burden typically associated with procedural planning. **(C02)**

Code Repository

Versioning for all infrastructure and application code is a key requirement. To meet this requirement, code repositories will be provided to capture published code and keep track of all submitted versions. This will improve the ability to both troubleshoot and roll back to a more stable version, when required. **(R04)**

Configuration Management

A configuration management tool will be provided within this design to handle operating system and application-level configuration items. This tool will also ensure all necessary agents required for continuous deployment are present. **(R05)**

Once instances are launched, they will be automatically registered with this configuration management tool. One configuration management server will be dedicated to HumanityLink Distributed Modeling, and another will be dedicated to HumanityLink Fleet Management environments. This will allow for independent scaling of the configuration management servers, and will also segregate their operation.

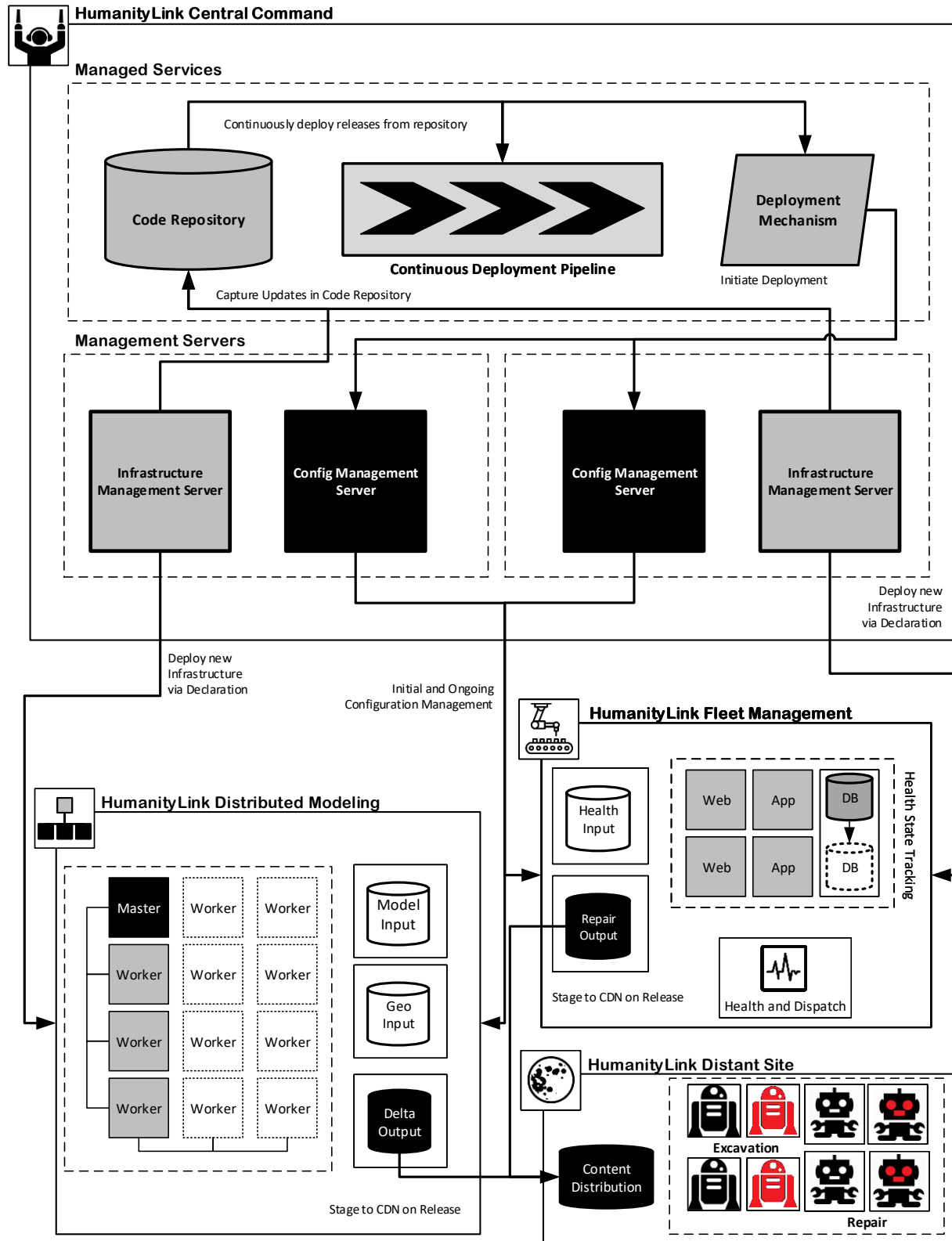
Content Distribution

Given the centralized nature of the HumanityLink application and projected plans for growth into very distant or off-earth areas, a mechanism for improving performance of these sites will be included. This will utilize content distribution, which will stage required content close to the consumer, where it can be accessed much more quickly than would be possible without the solution. **(R07, A01, A02, A04)**

Deployment Pipeline

In order to support rapid releases of HumanityLink required by the new initiative, a continuous deployment pipeline service will be leveraged. This will help orchestrate the build, test and deployment process, which will accomplish deployment goals while making best use of limited personnel resources. **(R02)**

Conceptual Diagram



Arrangement of primary components within the dedicated Central Command environment.

Logical Design

Infrastructure Architecture

A single AWS account dedicated to hosting HumanityLink Central Command components will be created, and within this account a single VPC will be provisioned. Two subnets associated with two separate availability zones will be created using address space not overlapping with any other deployment. Instances responsible for each service will be distributed across these availability zones for resiliency purposes. **(R01, RA02)**

Managed environments will be created in separate AWS accounts using a single VPC per application, in accordance with the original infrastructure design. These managed HumanityLink Distributed Modeling and Fleet Management environments will be created in the same region as HumanityLink Central Command, enabling these VPC's to leverage cross-account VPC peering.

VPC peering will allow for direct communication between managing and managed components without requiring complex configuration associated with third-party VPN solutions, or the risk of a publicly-exposed configuration. Each managed environment will require an independent peering relationship with the management VPC. Establishment of peering relationships with a central VPC does not enable transitive communication, so managed environments will have no means of communicating.

HLCC address space will be allocated from the top of an octet range, and managed application environments will be provisioned by incrementing that same octet from the bottom of the range. NAT gateways will be deployed in each of the private subnets to provide outbound access to the internet for use in retrieving patches and other relevant system updates. Virtual Private Gateways and NACL's will provide inbound administrative access for the technical team, and this access will be limited to SSH only.

Infrastructure and Configuration Management components have been sized to initially accommodate a 400-node global deployment. Considering the initial production and DR deployment of HLDM and HLFM consists of less than 50 nodes, this design will accommodate addition of several new environments without resizing **(RP01)**. When the time comes to expand this deployment, upward adjustment of OpsWorks for Chef Automate instance resources will be all that is required. Terraform management servers will likely not require adjustment, as their purpose is to periodically execute the tool and provide management tools to the administrator only. These duties are not expected to be computationally-intensive.

From a security perspective, instances will be permitted to access hosted services, including S3, CodeCommit, CodeDeploy and CodePipeline, as required. Instances will not be permitted to access one-another, in line with standard security requirements previously established. **(RS01, RS02)**

To support off-earth excavation sites, AWS CloudFront will be forward-deployed closest to the work site using AWS region EARTH-MOON-1 **(R07)**. Aside from this addition, the architecture of HLDM and HLFM requires no adjustment, as work is performed asynchronously and input/output is limited to files transmitted to and from S3. Reliable or low-latency connectivity is not required by the application architecture. However, use of CloudFront will allow application files to transit the more-reliable AWS infrastructure, which will both increase the reliability and performance of these remote excavation sites.

Design Decisions

| Area | Decision | Meets Requirement | Justification |
|-------------------------------------|-------------------------------|-----------------------------|---|
| Account Architecture | Parent/Child | R01 | Central Command will exist in a parent AWS account, and all HLDM/HLFM environments will exist in children. This will eliminate account maximum constraints, enhance segregation and increase flexibility. |
| Environment Location | US-EAST-1 | R01 | All control and expansion environments will be located in the US due to the increased military protection available. The original design leverages this location, and HLCC will expand on this. |
| Environment Segregation Method | Dedicated VPC per-Environment | R01 | Use of a separate VPC for each application environment segregates operations between HLFM/HLDM in alignment with the original design. |
| Environment Connectivity Method | Cross-Account VPC Peering | R01 | Utilizing separate accounts allows creation of more VPC's than the per-account limit of 5, while preserving connectivity within the same region using peering. |
| WAN Connectivity | NAT Gateway | RM09 | In transitioning to a centralized control model, NAT gateways will be added to private subnets in support of control component WAN connectivity. |
| Remote Site Performance Improvement | CloudFront | R07 A04 RP02 | CloudFront will be utilized to increase download and upload performance of packages from remote and off-earth sites |
| Instance Configuration | M4 Large + Auto Recover | RP01 RA04 | This specification provides support for up to 200 managed nodes per-instance using Chef Automate. Auto-recover monitor provides HA-like functionality for single instances. |
| Instance Backup | EBS Snapshot 1XD | RR01 RR02 | A once daily EBS snapshot supports the instance-level RPO and RTO. |
| Instance Security | Security Groups | RS01 | Security groups will be configured to restrict unneeded access, and access from remote managed nodes to management servers will be allowed. |
| Instance Scaling | Scale Up | RP01 | Configuration management is a scale-up workload. Resources will need to be increased once HLFM/HLDM >400 nodes |
| Network Security | NACL | RS02 | Network ACL's will be added to prevent non-administrative access from remote workstations (SSH). |

Management Architecture

Infrastructure Management

HashiCorp Terraform will provide cloud provisioning ability to the environment implementation team. This tool is declarative in nature, which requires that the administrator only describe the future state desired, leaving intelligence built into the tool to determine the implementation path.

As this relates to HumanityLink Central Command, this tool will construct new environments to house HumanityLink Distributed Modeling and HumanityLink Fleet Management. Components to be constructed include VPC's, subnets, access lists, security groups, S3 buckets, and EC2 instances, among others. Following initial deployment, a Terraform provisioner will associate instances with Chef Server, providing an accurate, uniform view of the inventory from that tool.

State files maintaining awareness of the environment current-state will be stored in S3 and code will be hosted in CodeCommit repositories. The speed and intelligence of the tool directly supports technical objectives and operational constraints associated with this project. **(R01, R03, C01, C02)**

Configuration Management

AWS OpsWorks for Chef Automate will provide initial and ongoing Operating System and Application configuration management. Separate servers will exist to service both HumanityLink Distributed Modeling and HumanityLink Fleet Management. Integration between OpsWorks for Chef Automate and AWS CodeCommit will be used to store cookbooks so changes to these items can be tracked over time. **(R05, C02)**

Content Distribution

AWS CloudFront will be utilized to support performance and reliability improvement for very distant or off-earth locations. This will allow S3 content originating from HLDM and HLFM to be transmitted over the AWS network to an endpoint (CDN) close to the consumer, where it can then be quickly accessed. Excavation sites can be provided with job information using this method without changing the overall systems architecture or location. **(R07, A04)**

Deployment Pipeline

To support continuous deployment of HumanityLink applications, AWS CodePipeline is being leveraged as part of this solution. This tool will orchestrate the build, testing, approval and deployment of new versions of HumanityLink without requiring excess administrative effort. All system images will be equipped with the AWS CodeDeploy agent used by this workflow, which will ensure uniform ability to apply application updates across the fleet. **(R02)**

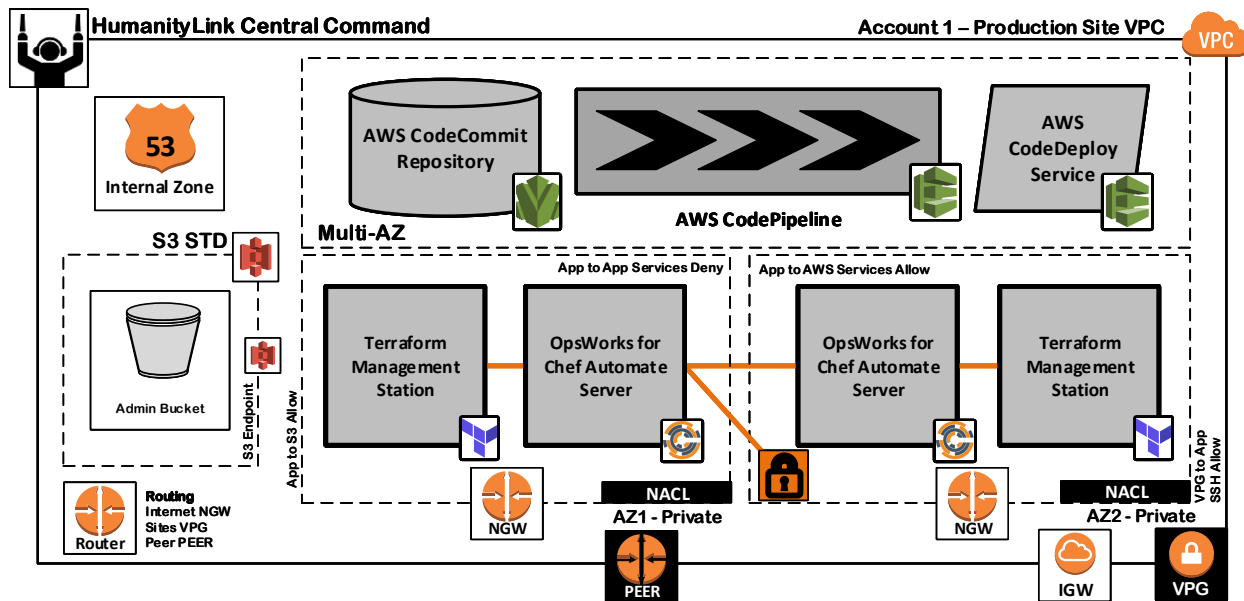
Code Repository

AWS CodeCommit will be employed universally as a back-end repository for Terraform, OpsWorks for Chef Automate and CodePipeline. This will ensure all changes to source materials are correctly tracked and versioned. Versioning of Chef cookbooks will ease troubleshooting and help improve operational efficiency. CodePipeline will be configured to monitor CodeCommit for updates to the application so these can be staged for widespread deployment. If this becomes required, CodeCommit can serve as a storage location for application binaries, as well. **(R04)**

Design Decisions

| Area | Decision | Meets Requirement | Justification |
|---------------------------------------|----------------------------|------------------------------------|---|
| Infrastructure Deployment Tool | Terraform | R01 R03 C01 C02 | A declarative tool with support for AWS has been chosen due to its awareness of deployment state and extensive feature set. |
| Infrastructure Tool Versioning | Code Commit + S3 | RA01 R04 R06 | Using Git for TF configuration files enables robust versioning. State will be stored remotely in S3 to enable versioning for that portion of the configuration. |
| Infrastructure Tool Location | Dedicated Instances | RA02 | Both HLDM and HLFM will use a dedicated Terraform server and state will only be accessed from there, preventing locking and duplicate deployment issues. |
| Configuration Management Tool | OpsWorks | R01 R03 R06 C01 | This choice takes advantage of AWS platform integration, and is widely supported and straightforward to use, supporting operational efficiency goals. |
| Configuration Management Version | OpsWorks for Chef Automate | R03 R05 R06 | OpsWorks for Chef Automate has been chosen for its AWS integration and ongoing management abilities. Chef is procedural and requires detailed plans |
| Configuration Management Registration | Terraform Provisioner | R05 RM05 | Following initial provisioning by Terraform, the Chef provider will run and associate the instance with Chef Server |
| Continuous Deployment Orchestrator | AWS CodePipeline | RA01 R02 R06 | CodePipeline will provide overall orchestration for the continuous deployment process, from final commit through testing and deployment |
| Continuous Deployment Mechanism | AWS CodeDeploy | RA01 R02 R06 | CodeDeploy will perform automated testing with deployment of software using an instance-deployed agent. |
| Continuous Deployment Source | Code Commit Repository | R04 R06 | This choice aligns with the use of Code Commit globally, allowing for a uniform code storage approach. |
| Continuous Deployment Trigger | Manual | RM08 | Upon release of a new version of code, administrative permission will have to be explicitly granted before deployment action takes place. |
| Continuous Deployment Mode | Rolling Upgrade | RA03 | Total downtime will be avoided by performing rolling upgrades of the application. For significant code revisions, another mode may be needed |

Logical Diagram

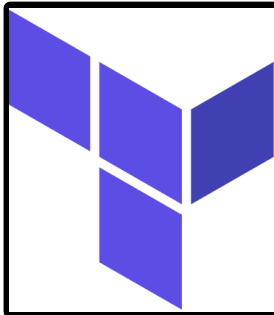


Logical configuration of HLCC servers and services within a production VPC.

Service Associations

| Service | Name | Purpose | Repository |
|--------------|-------------|----------------|------------|
| CodeCommit | INFRA | Terraform | - |
| CodeCommit | HLDM | HLDM Code | - |
| CodeCommit | HLFM | HLFM Code | - |
| CodeCommit | CONFIG | OpsWorks | - |
| CodePipeline | HLDM | HLDM Deploy | HLDM |
| CodePipeline | HLFM | HLFM Deploy | HLFM |
| OpsWorks | OPSWORKS01 | HLDM Config | CONFIG |
| OpsWorks | OPSWORKS02 | HLFM Config | CONFIG |
| Terraform | TERRAFORM01 | HLDM Terraform | INFRA |
| Terraform | TERRAFORM02 | HLFM Terraform | INFRA |
| S3 | ADMIN | Admin Storage | - |

Solution Stack



**Continuous Application
Deployment**

**Application Installation and
Configuration**

**Operating System and Role
Configuration**

Compute

Storage

Network

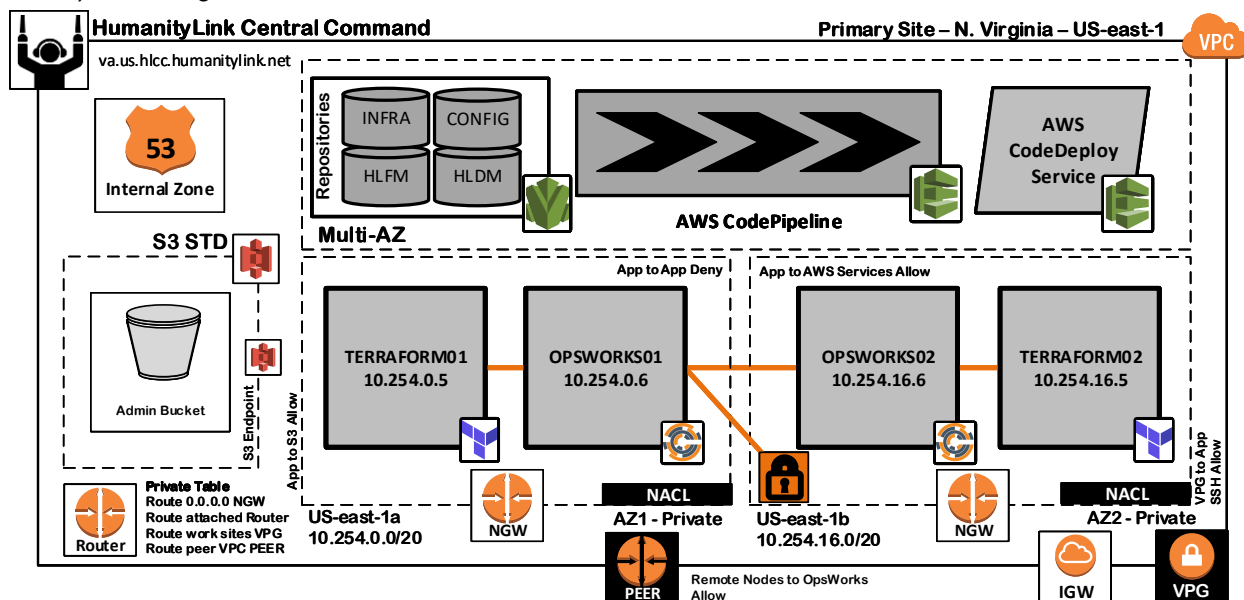
Environment

Code Repository

Code, infrastructure, configuration, and deployment management solutions used in HLCC

Physical Design

Physical Diagram



Physical configuration of HLCC servers and services within a production VPC.

Configuration Tables

Networks

| Private Network | Network | Usable IP's |
|-----------------|----------------|-------------|
| priv us-east-1a | 10.254.0.0/20 | 4091 |
| priv us-east-1b | 10.254.16.0/20 | 4091 |

Instances

| Name | IP | Type | CPU | Memory | Disk 1 | OS | Apps |
|-------------|-------------|------|-----|--------|--------|---------|---------------------|
| TERRAFORM01 | 10.254.0.5 | M4.L | 2 | 8 | 60 | CentOS7 | Terraform 0.9.11 |
| TERRAFORM02 | 10.254.16.5 | M4.L | 2 | 8 | 60 | CentOS7 | Terraform 0.9.11 |
| OPSWORKS01 | 10.254.0.6 | M4.L | 2 | 8 | 60 | Managed | Chef Server 12.15.8 |
| OPSWORKS02 | 10.254.16.6 | M4.L | 2 | 8 | 60 | Managed | Chef Server 12.15.8 |

Deployment and Management

HumanityLink Distributed Modeling

Summary

This section outlines primary tasks needed to create new HumanityLink Distributed Modeling environments using the specified approach and automation tools. The environment shown in this example is in existence, and these values cannot be used to create a new environment.

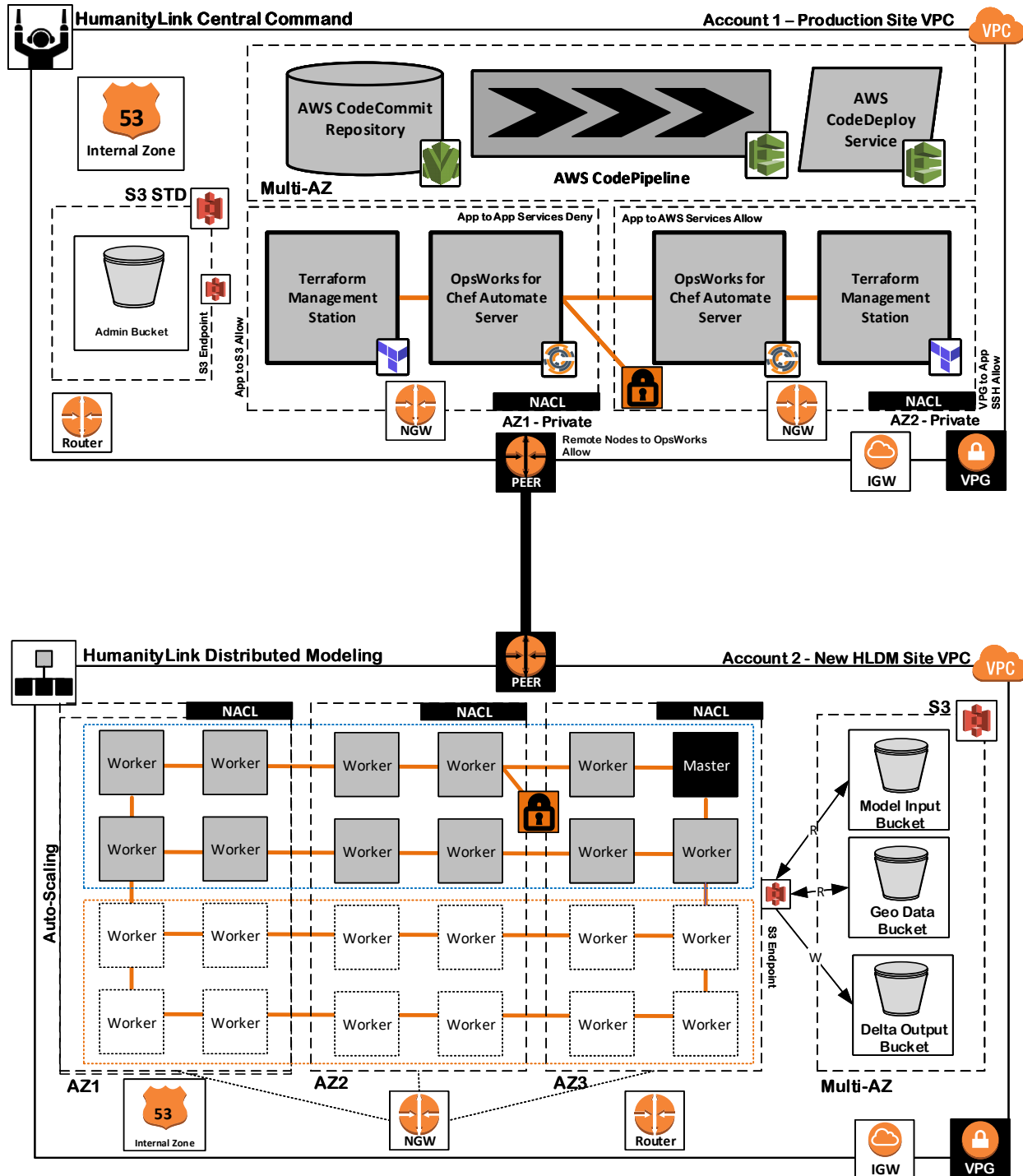
A structured address allocation and resource provisioning process must take place before new environment creation is initiated. This example is intended to show how deployment of the existing environment would be approached with these tools.

Procedure Overview

Terraform

- 1) Create Virtual Private Cloud and sub-resources
 - a. Subnets
 - i. 172.31.0.0/20
 - ii. 172.31.16.0/20
 - iii. 172.31.32.0/20
 - b. Access Lists
 - i. 10.1.0.0/24 SSH Allow
 - c. Internet Gateway
 - d. Virtual Private Gateway
 - e. NAT Gateway
 - f. Routing Tables
 - i. Route 0.0.0.0 NGW
 - ii. Route 10.1.0.0/24 VPG
 - iii. Route peer VPG PEER LINK
- 2) Create Object Storage resources
 - a. S3 Buckets
 - i. Model Input
 - ii. Geo Data
 - iii. Delta Output
- 3) Deploy Compute Instances
 - a. (12) HumanityLink Distributed Modeling nodes
 - i. Size R4.2XL
 - ii. OS CentOS7
 - iii. AMI ami-46c1b650
- 4) Register Compute Instances with OpsWorks for Chef Automate
 - a. <https://opsworks01-liznh147rllbmmk5.us-east-1.opsworks-cm.io/>

Logical Diagram



Central Command management components connected to a newly-deployed, managed HLDM VPC

Physical Configuration Table

The HLDM environment provisioning examples shown in this document will use the physical details below for illustration. These values mirror a production environment in operation. New networks, names and IP's must be allocated when planning to provision a new environment.

Networks

| Name | Network | Usable IP's |
|------------|----------------|-------------|
| us-east-1a | 172.31.0.0/20 | 4091 |
| us-east-1b | 172.31.16.0/20 | 4091 |
| us-east-1c | 172.31.32.0/20 | 4091 |

Instances

| Name | IP | Type | CPU | Memory | Disk | OS | Apps |
|--------|-------------|--------|-----|--------|------|---------|-----------|
| HLDM01 | 172.31.0.5 | R4.2XL | 8 | 61 | 60 | CentOS7 | HLDM v1.0 |
| HLDM02 | 172.31.16.5 | R4.2XL | 8 | 61 | 60 | CentOS7 | HLDM v1.0 |
| HLDM03 | 172.31.32.5 | R4.2XL | 8 | 61 | 60 | CentOS7 | HLDM v1.0 |
| HLDM04 | 172.31.0.6 | R4.2XL | 8 | 61 | 60 | CentOS7 | HLDM v1.0 |
| HLDM05 | 172.31.16.6 | R4.2XL | 8 | 61 | 60 | CentOS7 | HLDM v1.0 |
| HLDM06 | 172.31.32.6 | R4.2XL | 8 | 61 | 60 | CentOS7 | HLDM v1.0 |
| HLDM07 | 172.31.0.7 | R4.2XL | 8 | 61 | 60 | CentOS7 | HLDM v1.0 |
| HLDM08 | 172.31.16.7 | R4.2XL | 8 | 61 | 60 | CentOS7 | HLDM v1.0 |
| HLDM09 | 172.31.32.7 | R4.2XL | 8 | 61 | 60 | CentOS7 | HLDM v1.0 |
| HLDM10 | 172.31.0.8 | R4.2XL | 8 | 61 | 60 | CentOS7 | HLDM v1.0 |
| HLDM11 | 172.31.16.8 | R4.2XL | 8 | 61 | 60 | CentOS7 | HLDM v1.0 |
| HLDM12 | 172.31.32.8 | R4.2XL | 8 | 61 | 60 | CentOS7 | HLDM v1.0 |

Infrastructure Deployment Plan

This plan details deployment of all infrastructure layers required by the solution using Terraform. (**RM01, RM07**)

NOTE: Test run of this deployment was **SUCCESSFUL**. See “Terraform Test Run” section for full details.

Set Global Provider (aws.tf)

```
provider "aws" {
  access_key = "${var.aws_access_key}"
  secret_key = "${var.aws_secret_key}"
  region = "${var.aws_region}"
}
```

Set Global Variables (Terraform.tvars)

```
aws_access_key = "xxxxxx"
aws_secret_key = "xxxxxxx"
aws_key_path = "~/.ssh/aws.pem"
aws_key_name = "aws"
```

Set Deployment Variables (variables.tf)

This section sets global variables used in the deployment, including subnet allocations and source AMI.

```
variable "aws_access_key" {}
variable "aws_secret_key" {}
variable "aws_key_path" {}
variable "aws_key_name" {}

variable "aws_region" {
  description = "HumanityLink Production - US-EAST-1"
  default = "us-east-1"
}

variable "amis" {
  description = "CentOS7 AMI"
  default = {
    us-east-1 = "ami-46c1b650" #Validated CentOS7 Image for HumanityLink
  }
}

variable "vpc_cidr" {
  description = "HumanityLink Distributed Modeling VPC"
  default = "172.31.0.0/16" #HLDM Production VPC
}

variable "private_subnet_cidr1" {
  description = "HLDM AZ1 Private"
  default = "172.31.0.0/20" #HLDM Production VPC Private Subnet 1
}

variable "private_subnet_cidr2" {
  description = "HLDM AZ2 Private"
  default = "172.31.16.0/20" #HLDM Production VPC Private Subnet 2
}
```

```
variable "private_subnet_cidr3" {  
    description = "HLDM AZ3 Private"  
    default = "172.31.32.0/20" #HLDM Production VPC Private Subnet 3  
}  
  
variable "public_subnet_cidr1" {  
    description = "HLDM AZ1 Public"  
    default = "172.31.48.0/20" #HLDM Production VPC Public Subnet 1  
}  
  
variable "public_subnet_cidr2" {  
    description = "HLDM AZ2 Public"  
    default = "172.31.64.0/20" #HLDM Production VPC Public Subnet 2  
}  
variable "public_subnet_cidr3" {  
    description = "HLDM AZ3 Private"  
    default = "172.31.80.0/20" #HLDM Production VPC Public Subnet 3  
}
```

Create VPC, NGW, IGW, Routing and ACL's (vpc.tf)

```
resource "aws_vpc" "default" {
  cidr_block = "${var.vpc_cidr}"
  enable_dns_hostnames = true
  tags {
    Name = "HLDM"
  }
}

resource "aws_internet_gateway" "default" {
  vpc_id = "${aws_vpc.default.id}"
}

resource "aws_eip" "nat1a" {
  vpc = true
}
resource "aws_eip" "nat1b" {
  vpc = true
}
resource "aws_eip" "nat1c" {
  vpc = true
}

resource "aws_nat_gateway" "gw1a" {
  allocation_id = "${aws_eip.nat1a.id}"
  subnet_id     = "${aws_subnet.us-east-1a-public.id}"
}

resource "aws_nat_gateway" "gw1b" {
  allocation_id = "${aws_eip.nat1b.id}"
  subnet_id     = "${aws_subnet.us-east-1b-public.id}"
}

resource "aws_nat_gateway" "gw1c" {
  allocation_id = "${aws_eip.nat1c.id}"
  subnet_id     = "${aws_subnet.us-east-1c-public.id}"
}

/*
  Private Subnet 1
*/
resource "aws_subnet" "us-east-1a-private" {
  vpc_id = "${aws_vpc.default.id}"

  cidr_block = "${var.private_subnet_cidr1}"
  availability_zone = "us-east-1a"

  tags {
    Name = "Private Subnet 1"
  }
}
```

```

resource "aws_route_table" "us-east-1a-private" {
  vpc_id = "${aws_vpc.default.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_eip.nat1a.id}"
  }

  tags {
    Name = "Private 1 to Internet"
  }
}
resource "aws_route_table_association" "us-east-1a-private" {
  subnet_id = "${aws_subnet.us-east-1a-private.id}"
  route_table_id = "${aws_route_table.us-east-1a-private.id}"
}

/*
Private Subnet 2
*/
resource "aws_subnet" "us-east-1b-private" {
  vpc_id = "${aws_vpc.default.id}"

  cidr_block = "${var.private_subnet_cidr2}"
  availability_zone = "us-east-1b"

  tags {
    Name = "Private Subnet 2"
  }
}

resource "aws_route_table" "us-east-1b-private" {
  vpc_id = "${aws_vpc.default.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_eip.nat1b.id}"
  }

  tags {
    Name = "Private 2 to Internet"
  }
}
resource "aws_route_table_association" "us-east-1b-private" {
  subnet_id = "${aws_subnet.us-east-1b-private.id}"
  route_table_id = "${aws_route_table.us-east-1b-private.id}"
}

```

```

/*
  Private Subnet 3
*/
resource "aws_subnet" "us-east-1c-private" {
  vpc_id = "${aws_vpc.default.id}"

  cidr_block = "${var.private_subnet_cidr3}"
  availability_zone = "us-east-1c"

  tags {
    Name = "Private Subnet 3"
  }
}

resource "aws_route_table" "us-east-1c-private" {
  vpc_id = "${aws_vpc.default.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_eip.nat1c.id}"
  }

  tags {
    Name = "Private 3 to Internet"
  }
}

resource "aws_route_table_association" "us-east-1c-private" {
  subnet_id = "${aws_subnet.us-east-1c-private.id}"
  route_table_id = "${aws_route_table.us-east-1c-private.id}"
}

/*
  Public Subnet 1
*/
resource "aws_subnet" "us-east-1a-public" {
  vpc_id = "${aws_vpc.default.id}"

  cidr_block = "${var.public_subnet_cidr1}"
  availability_zone = "us-east-1a"

  tags {
    Name = "Public Subnet 1"
  }
}

resource "aws_route_table" "us-east-1a-public" {
  vpc_id = "${aws_vpc.default.id}"

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

```

```

    tags {
      Name = "Public 1 Route Table"
    }
  }
  resource "aws_route_table_association" "us-east-1a-public" {
    subnet_id = "${aws_subnet.us-east-1a-public.id}"
    route_table_id = "${aws_route_table.us-east-1a-public.id}"
  }

  /*
    Public Subnet 2
  */
  resource "aws_subnet" "us-east-1b-public" {
    vpc_id = "${aws_vpc.default.id}"

    cidr_block = "${var.public_subnet_cidr2}"
    availability_zone = "us-east-1b"

    tags {
      Name = "Public Subnet 2"
    }
  }

  resource "aws_route_table" "us-east-1b-public" {
    vpc_id = "${aws_vpc.default.id}"

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

    tags {
      Name = "Public 2 Route Table"
    }
  }
  resource "aws_route_table_association" "us-east-1b-public" {
    subnet_id = "${aws_subnet.us-east-1b-public.id}"
    route_table_id = "${aws_route_table.us-east-1b-public.id}"
  }

  /*
    Public Subnet 3
  */
  resource "aws_subnet" "us-east-1c-public" {
    vpc_id = "${aws_vpc.default.id}"

    cidr_block = "${var.public_subnet_cidr3}"
    availability_zone = "us-east-1c"

    tags {

```



```

        Name = "Public Subnet 3"
    }
}
resource "aws_route_table" "us-east-1c-public" {
    vpc_id = "${aws_vpc.default.id}"

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

    tags {
        Name = "Public 3 Route Table"
    }
}
resource "aws_route_table_association" "us-east-1c-public" {
    subnet_id = "${aws_subnet.us-east-1c-public.id}"
    route_table_id = "${aws_route_table.us-east-1c-public.id}"
}

#Administrative access ACL for remote sites via SSH, attached to private subnets
resource "aws_network_acl" "AdminToHLDM" {
    vpc_id = "${aws_vpc.default.id}"
    subnet_ids = ["us-east-1a-private", "us-east-1b-private", "us-east-1c-private"]
    egress {
        protocol    = "tcp"
        rule_no     = 2
        action      = "allow"
        cidr_block  = "10.1.0.0/24"
        from_port   = 22
        to_port     = 22
    }

    ingress {
        protocol    = "tcp"
        rule_no     = 1
        action      = "allow"
        cidr_block  = "10.1.0.0/24"
        from_port   = 22
        to_port     = 22
    }

    tags {
        Name = "AdminToHLDM"
    }
}

```

HLDM Instances (HLDM.tf)

This section creates security groups governing communication and deploys (12) HLDM nodes with EBS volumes. The deployment alternates subnets/availability zones for availability purposes. These nodes are then joined to OpsWorks for Chef Automate to enable ongoing configuration management at the operating system level.

```
/*
  HumanityLink Distributed Modeling Servers
*/
#Security groups allowing HLDM, SSH and ICMP inbound
resource "aws_security_group" "hldm" {
  name = "vpc_hldm"
  description = "Allow incoming HLDM connections."

  ingress { # HLDM Inbound
    from_port = 50075
    to_port = 50075
    protocol = "tcp"
    cidr_blocks = ["172.31.0.0/16"]
  }
  ingress { # SSH Inbound
    from_port = 22
    to_port = 22
    protocol = "tcp"
    cidr_blocks = ["10.0.1.0/24"]
  }
  ingress { # ICMP Inbound
    from_port = -1
    to_port = -1
    protocol = "icmp"
    cidr_blocks = ["10.0.1.0/24"]
  }
  vpc_id = "${aws_vpc.default.id}"
  tags {
    Name = "HLDM_SG"
  }
}

#Instances specified in sequence in order to alternate deployment across all (3)
Availability Zones
resource "aws_instance" "HLDM01" {
  ami = "${lookup(var.amis, var.aws_region)}"
  availability_zone = "us-east-1a"
  instance_type = "r4.2xlarge"
  key_name = "${var.aws_key_name}"
  vpc_security_group_ids = ["${aws_security_group.hldm.id}"]
  subnet_id = "${aws_subnet.us-east-1a-private.id}"
  source_dest_check = false
#GP2 EBS volume attachment
  root_block_device {
    volume_type = "gp2"
    volume_size = 60
    delete_on_termination = "true"
  }
}
```

```

    }
    #Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks01-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hldm"
        user_key        = "xxxxxxx"
        node_name       = "HLDM01"
        run_list        = [ "HLDM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type    = "ssh"
        user    = "svc_chef"
        password = "xxxxxxx"
    }
    #Tags to be used by CodeDeploy
    tags {
        Application = "HLDM"
    }
}

resource "aws_instance" "HLDM02" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1b"
    instance_type = "r4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hldm.id}"]
    subnet_id = "${aws_subnet.us-east-1b-private.id}"
    source_dest_check = false
    #GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
    #Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks01-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hldm"
        user_key        = "xxxxxxx"
        node_name       = "HLDM02"
        run_list        = [ "HLDM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type    = "ssh"
        user    = "svc_chef"
        password = "xxxxxxx"
    }
}

```

```

#Tags to be used by CodeDeploy
tags {
    Application = "HLDM"
}

resource "aws_instance" "HLDM03" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1c"
    instance_type = "r4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hldm.id}"]
    subnet_id = "${aws_subnet.us-east-1c-private.id}"
    source_dest_check = false
#GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
#Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks01-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hldm"
        user_key        = "xxxxxxx"
        node_name       = "HLDM03"
        run_list        = [ "HLDM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type      = "ssh"
        user      = "svc_chef"
        password  = "xxxxxxx"
    }
#Tags to be used by CodeDeploy
tags {
    Application = "HLDM"
}

resource "aws_instance" "HLDM04" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1a"
    instance_type = "r4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hldm.id}"]
    subnet_id = "${aws_subnet.us-east-1a-private.id}"
    source_dest_check = false
#GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
    }

```

```

        volume_size = 60
        delete_on_termination = "true"
    }
#Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks01-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hldm"
        user_key        = "xxxxxxx"
        node_name       = "HLDM04"
        run_list        = [ "HLDM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type    = "ssh"
        user    = "svc_chef"
        password = "xxxxxxx"
    }
#Tags to be used by CodeDeploy
    tags {
        Application = "HLDM"
    }
}

resource "aws_instance" "HLDM05" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1b"
    instance_type = "r4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hldm.id}"]
    subnet_id = "${aws_subnet.us-east-1b-private.id}"
    source_dest_check = false
#GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
#Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks01-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hldm"
        user_key        = "xxxxxxx"
        node_name       = "HLDM05"
        run_list        = [ "HLDM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type    = "ssh"
        user    = "svc_chef"
        password = "xxxxxxx"
    }
}

```

```

    }
    #Tags to be used by CodeDeploy
    tags {
        Application = "HLDM"
    }
}

resource "aws_instance" "HLDM06" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1c"
    instance_type = "r4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hldm.id}"]
    subnet_id = "${aws_subnet.us-east-1c-private.id}"
    source_dest_check = false
    #GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
    #Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks01-liznh147rllbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hldm"
        user_key        = "xxxxxxx"
        node_name       = "HLDM06"
        run_list        = [ "HLDM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type      = "ssh"
        user      = "svc_chef"
        password  = "xxxxxxx"
    }
    #Tags to be used by CodeDeploy
    tags {
        Application = "HLDM"
    }
}

resource "aws_instance" "HLDM07" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1a"
    instance_type = "r4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hldm.id}"]
    subnet_id = "${aws_subnet.us-east-1a-private.id}"
    source_dest_check = false

```

```

#GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
#Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks01-liznh147rllbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hldm"
        user_key        = "xxxxxxx"
        node_name       = "HLDM07"
        run_list        = [ "HLDM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type      = "ssh"
        user      = "svc_chef"
        password  = "xxxxxxx"
    }
#Tags to be used by CodeDeploy
    tags {
        Application = "HLDM"
    }
}

resource "aws_instance" "HLDM08" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1b"
    instance_type = "r4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hldm.id}"]
    subnet_id = "${aws_subnet.us-east-1b-private.id}"
    source_dest_check = false
#GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
#Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks01-liznh147rllbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hldm"
        user_key        = "xxxxxxx"
        node_name       = "HLDM08"
        run_list        = [ "HLDM_Role" ]
        version         = "12.15.8"
    }
    connection {

```

```

        type      = "ssh"
        user       = "svc_chef"
        password   = "xxxxxxx"
    }
    #Tags to be used by CodeDeploy
    tags {
        Application = "HLDM"
    }
}

resource "aws_instance" "HLDM09" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1c"
    instance_type = "r4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hldm.id}"]
    subnet_id = "${aws_subnet.us-east-1c-private.id}"
    source_dest_check = false
    #GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
    #Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks01-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hldm"
        user_key        = "xxxxxxx"
        node_name       = "HLDM09"
        run_list        = [ "HLDM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type      = "ssh"
        user       = "svc_chef"
        password   = "xxxxxxx"
    }
    #Tags to be used by CodeDeploy
    tags {
        Application = "HLDM"
    }
}

resource "aws_instance" "HLDM10" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1a"
    instance_type = "r4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hldm.id}"]
    subnet_id = "${aws_subnet.us-east-1a-private.id}"

```



```

    source_dest_check = false
#GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
#Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks01-liznh147rllbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hldm"
        user_key        = "xxxxxxx"
        node_name       = "HLDM10"
        run_list        = [ "HLDM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type      = "ssh"
        user      = "svc_chef"
        password  = "xxxxxxx"
    }
#Tags to be used by CodeDeploy
    tags {
        Application = "HLDM"
    }
}

resource "aws_instance" "HLDM11" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1b"
    instance_type = "r4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hldm.id}"]
    subnet_id = "${aws_subnet.us-east-1b-private.id}"
    source_dest_check = false
#GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
#Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks01-liznh147rllbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hldm"
        user_key        = "xxxxxxx"
        node_name       = "HLDM11"
        run_list        = [ "HLDM_Role" ]
        version         = "12.15.8"
    }
}

```

```

        connection {
            type      = "ssh"
            user       = "svc_chef"
            password   = "xxxxxxx"
        }
#Tags to be used by CodeDeploy
        tags {
            Application = "HLDM"
        }
    }

resource "aws_instance" "HLDM12" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1c"
    instance_type = "r4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hldm.id}"]
    subnet_id = "${aws_subnet.us-east-1c-private.id}"
    source_dest_check = false
#GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
#Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks01-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hldm"
        user_key        = "xxxxxxx"
        node_name       = "HLDM12"
        run_list        = [ "HLDM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type      = "ssh"
        user       = "svc_chef"
        password   = "xxxxxxx"
    }
#Tags to be used by CodeDeploy
        tags {
            Application = "HLDM"
        }
    }
}

```

This section creates S3 buckets to be used by HLDM, including lifecycle rules for data management.

#Creates base S3 buckets to be used by HLFM, including lifecycle rule for Infrequent Access

```
resource "aws_s3_bucket" "ModelInput" {
  bucket = "ModelInput"
  acl    = "private"

  versioning {
    enabled = true
  }
  lifecycle_rule {
    id      = "ModelsToIA"
    prefix  = "*"
    enabled = true

    transition {
      days          = 30
      storage_class = "STANDARD_IA"
    }
  }
}

resource "aws_s3_bucket" "GeoData" {
  bucket = "GeoData"
  acl    = "private"

  versioning {
    enabled = true
  }
  lifecycle_rule {
    id      = "GeoToIA"
    prefix  = "*"
    enabled = true
    transition {
      days          = 30
      storage_class = "STANDARD_IA"
    }
  }
}

resource "aws_s3_bucket" "DeltaOutput" {
  bucket = "DeltaOutput"
  acl    = "private"

  versioning {
    enabled = true
  }
  lifecycle_rule {
    id      = "DeltaToIA"
```

```

prefix = "*"
enabled = true

transition {
  days = 30
  storage_class = "STANDARD_IA"
}
}

```


Terraform Test Run

Results: Terraform plan **successfully** runs with the outlined environment build code. See embedded TXT file for results

```

C:\AWS\terraform>terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

[Full content omitted – see attached file]


hldm_terraform_vali
dation.txt

Plan: 43 to add, 0 to change, 0 to destroy.

```

Operating System Configuration Plan

Using OpsWorks for Chef Automate, perform basic configuration of the OS, install roles and updates (**RM03, RM07**). Primary objectives are being listed, as custom recipes and cookbooks require development.

HumanityLink Distributed Modeling Role Objectives (HLDM_Role)

- Assign HLDM role to registered nodes
- Set system hostname
- Install AWS CodeDeploy agent for continuous app deployment
- Check currently deployed version of HLDM against repository
- Deploy new version of HLDM from repository, if required
- Check current state of system updates against repository
- Deploy new system updates from repository, if required
- Test connectivity to other nodes in the farm
- Test connectivity to S3 buckets
- Check status of HLDM services and join farm

Application Deployment Pipeline

This section outlines tasks to be completed by Amazon Web Services code testing and release services. Production code repositories will be monitored for commits, and these actions will trigger a sequence of stages mediated by AWS CodePipeline.

Testing of this code will be conducted by CodeBuild, an automated, container-based solution. If testing is successful, results will be presented to administrators and deployment can be authorized.

Once approved, CodeDeploy will leverage an instance-deployed agent to update HumanityLink Distributed Modeling and reboot instances one at a time. Following this process, all instances will be running the latest version of HLDM. This solution results in continuous deployment for the application.

(R02)

Code Repository

- Add CodeCommit repository to development station using GIT and clone locally
- Perform development against HumanityLink Distributed Modeling code base
- Commit changes to CodeCommit repository

Code Pipeline

- Monitor CodeCommit repository HLDM for commits to the production branch
- Engage CodeBuild to perform automated testing of code before deployment

Code Build

- Utilize standard Ubuntu-based containers to build and test HLDM code
- Report results to CodePipeline workflow

Code Pipeline

- If tests conducted during CodeBuild are successful, send notification to Administrators

Approval

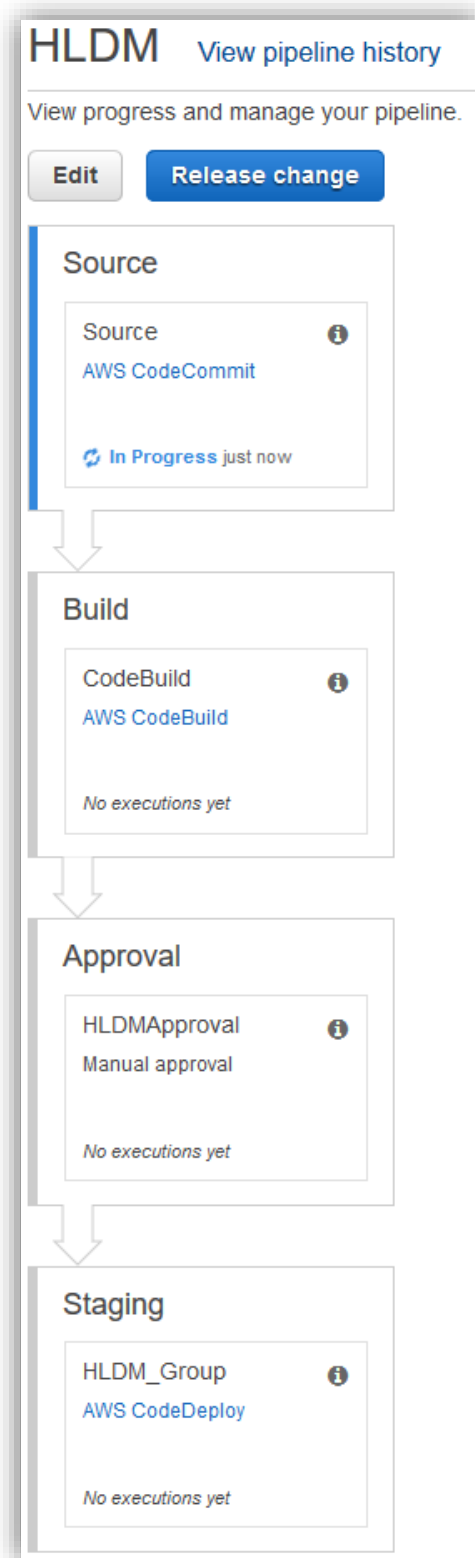
- Review results from automated CodeBuild testing process
- Provide CodePipeline permission to deploy HLDM across the instance fleet

Code Pipeline

- Engage CodeDeploy service to execute deployment of HLDM code across instance fleet

Code Deploy

- Update installations of HLDM across the instance fleet
 - Apply to instances with tag HLDM = True
 - Update one instance at a time
- Deployment of the latest version of HLDM is complete.



Orchestrated deployment phases resulting in continuous deployment for HLDM.

HumanityLink Fleet Management

Summary

This section outlines primary tasks needed to create new HumanityLink Fleet Management environments using the specified approach and automation tools. The environment shown in this example is in existence, and these values cannot be used to create a new environment.

A structured address allocation and resource provisioning process must take place before new environment creation is initiated. This example is intended to show how deployment of the existing environment would be approached with these tools.

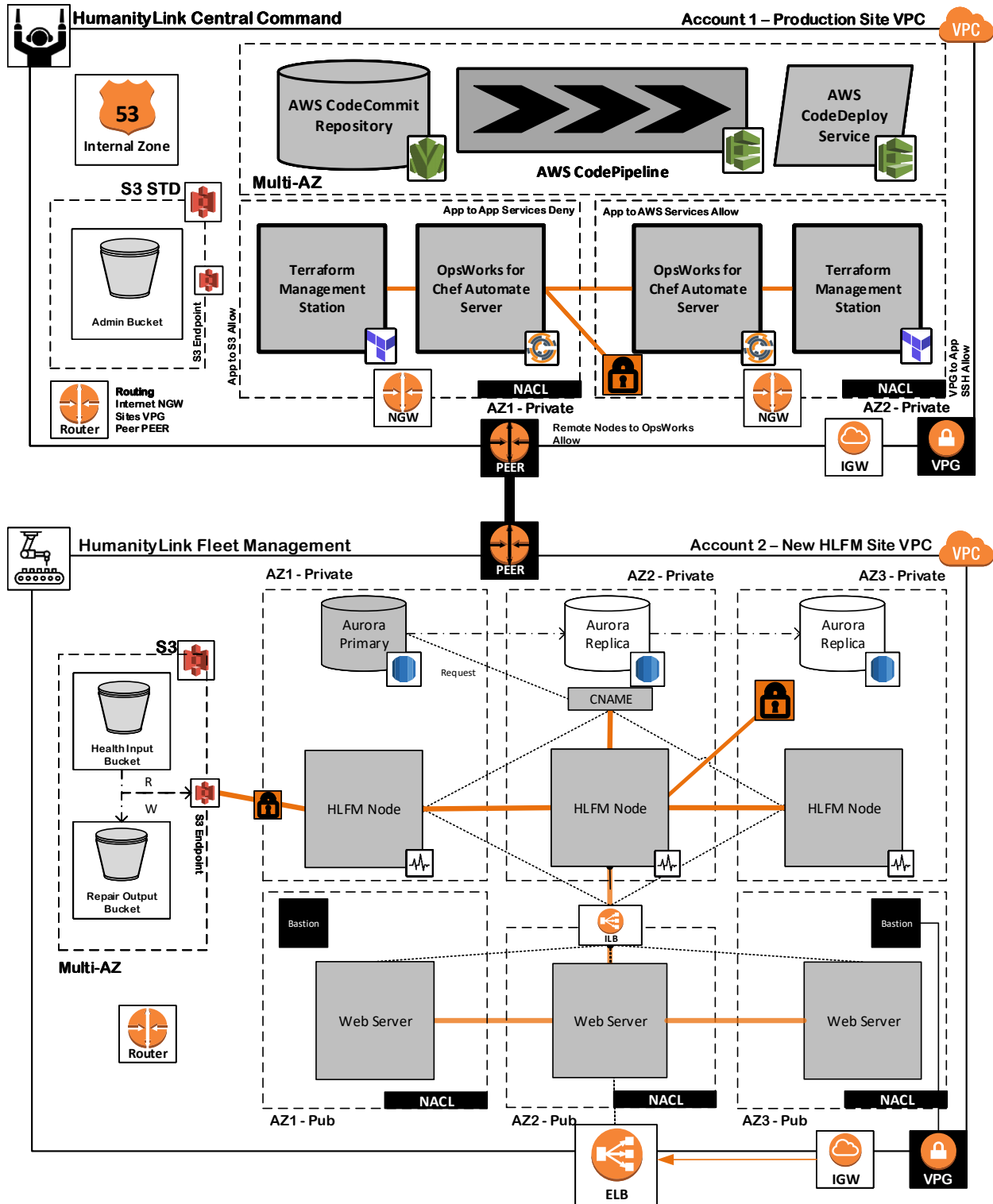
Procedure Overview

Terraform

- 1) Create Virtual Private Cloud and sub-resources
 - a. Public Subnets
 - i. 172.32.0.0/20
 - ii. 172.32.16.0/20
 - iii. 172.32.32.0/20
 - b. Private Subnets
 - i. 172.32.48.0/20
 - ii. 172.32.64.0/20
 - iii. 172.32.80.0/20
 - c. Access Lists
 - i. 10.1.0.0/24 SSH Allow Bastions
 1. 172.32.0.252
 2. 172.32.32.252
 - d. Internet Gateway
 - e. Virtual Private Gateway
 - f. NAT Gateway
 - g. Public Routing Tables
 - i. Route 0.0.0.0 IGW
 - ii. Route 10.1.0.0/24 VPG
 - iii. Route peer VPG PEER LINK
 - h. Private Routing Tables
 - i. Route 0.0.0.0 NGW
 - ii. Route peer VPG PEER LINK
 - i. External Load Balancer
 - i. External to Web Tier
 - j. Internal Load Balancer
 - i. Internal to App Tier

- 2) Create Object Storage resources
 - a. S3 Buckets
 - i. Health Input
 - ii. Repair Output
- 3) Deploy Compute Instances
 - a. (3) HumanityLink Fleet Management nodes
 - i. Size C4.2XL
 - ii. OS CentOS7
 - iii. AMI ami-46c1b650
 - b. (6) Web Server nodes
 - i. Size M4.L
 - ii. OS CentOS7
 - iii. AMI ami-46c1b650
 - c. (2) Bastion Hosts
 - i. Size T2.M
 - ii. OS CentOS7
 - iii. AMI ami-46c1b650
- 4) Deploy Aurora RDS instances and cluster
- 5) Register Compute Instances with OpsWorks for Chef Automate
 - a. <https://opsworks02-liznh147rllbmmk5.us-east-1.opsworks-cm.io/>

Logical Diagram



Central Command management components connected to a newly-deployed, managed HLFM VPC

Physical Configuration Table

The HLFM environment provisioning examples shown in this document will use the physical details below for illustration. These values mirror a production environment in operation. New networks, names and IP's must be allocated when planning to provision a new environment.

Networks

| Public Network | Network | Usable IP's |
|----------------|----------------|-------------|
| pub us-east-1a | 172.32.0.0/20 | 4091 |
| pub us-east-1b | 172.32.16.0/20 | 4091 |
| pub us-east-1c | 172.32.32.0/20 | 4091 |

| Private Network | Network | Usable IP's |
|-----------------|----------------|-------------|
| priv us-east-1a | 172.32.48.0/20 | 4091 |
| priv us-east-1b | 172.32.64.0/20 | 4091 |
| priv us-east-1c | 172.32.80.0/20 | 4091 |

Instances

| Name | IP | Type | CPU | Memory | Disk | OS | Apps |
|-------------|---------------|-----------|-----|--------|------|---------|------------|
| BAS01 | 172.32.0.252 | T2.M | 2 | 24 | 60 | CentOS7 | Lynx, SSH |
| BAS02 | 172.32.32.252 | T2.M | 2 | 24 | 60 | CentOS7 | Lynx, SSH |
| WEB01 | 172.32.0.5 | M4.L | 2 | 8 | 60 | CentOS7 | NGINX 1.13 |
| WEB02 | 172.32.16.5 | M4.L | 2 | 8 | 60 | CentOS7 | NGINX 1.13 |
| WEB03 | 172.32.32.5 | M4.L | 2 | 8 | 60 | CentOS7 | NGINX 1.13 |
| WEB04 | 172.32.0.6 | M4.L | 2 | 8 | 60 | CentOS7 | NGINX 1.13 |
| WEB05 | 172.32.16.6 | M4.L | 2 | 8 | 60 | CentOS7 | NGINX 1.13 |
| WEB06 | 172.32.32.6 | M4.L | 2 | 8 | 60 | CentOS7 | NGINX 1.13 |
| HLFM01 | 172.32.48.5 | C4.2XL | 8 | 15 | 60 | CentOS7 | HLFM 1.0 |
| HLFM02 | 172.32.64.5 | C4.2XL | 8 | 15 | 60 | CentOS7 | HLFM 1.0 |
| HLFM03 | 172.32.80.5 | C4.2XL | 8 | 15 | 60 | CentOS7 | HLFM 1.0 |
| AuroraPri | Auto | DB.R3.2XL | 8 | 61 | 1000 | Managed | Aurora RDS |
| AuroraRepl1 | Auto | DB.R3.2XL | 8 | 61 | 1000 | Managed | Aurora RDS |
| AuroraRepl2 | Auto | DB.R3.2XL | 8 | 61 | 1000 | Managed | Aurora RDS |

Infrastructure Deployment Plan

This plan details deployment of all infrastructure layers required by the solution using Terraform. (RM02, RM07)

NOTE: Test run of this deployment was **SUCCESSFUL**. See “Terraform Test Run” section for full details.

Set Global Provider (aws.tf)

```
provider "aws" {  
    access_key = "${var.aws_access_key}"  
    secret_key = "${var.aws_secret_key}"  
    region = "${var.aws_region}"  
}
```

Set Global Variables (Terraform.tvars)

```
aws_access_key = "xxxxxx"  
aws_secret_key = "xxxxxxx"  
aws_key_path = "~/.ssh/aws.pem"  
aws_key_name = "aws"
```

Set Deployment Variables (variables.tf)

This section sets global variables used in the deployment, including subnet allocations and source AMI.

```
variable "aws_access_key" {}  
variable "aws_secret_key" {}  
variable "aws_key_path" {}  
variable "aws_key_name" {}  
  
variable "aws_region" {  
    description = "HumanityLink Production - US-EAST-1"  
    default = "us-east-1"  
}  
  
variable "amis" {  
    description = "CentOS7 AMI"  
    default = {  
        us-east-1 = "ami-46c1b650" #Validated CentOS7 Image for HumanityLink  
    }  
}  
  
variable "vpc_cidr" {  
    description = "HumanityLink Fleet Management VPC"  
    default = "172.32.0.0/16" #HLFM Production VPC  
}  
  
variable "private_subnet_cidr1" {  
    description = "HLFM AZ1 Private"  
    default = "172.32.48.0/20" #HLFM Production VPC Private Subnet 1  
}  
  
variable "private_subnet_cidr2" {  
    description = "HLFM AZ2 Private"  
    default = "172.32.64.0/20" #HLFM Production VPC Private Subnet 2  
}
```

```

variable "private_subnet_cidr3" {
    description = "HLFM AZ3 Private"
    default = "172.32.80.0/20" #HLFM Production VPC Private Subnet 3
}

variable "public_subnet_cidr1" {
    description = "HLFM AZ1 Public"
    default = "172.32.0.0/20" #HLFM Production VPC Public Subnet 1
}

variable "public_subnet_cidr2" {
    description = "HLFM AZ2 Public"
    default = "172.32.16.0/20" #HLFM Production VPC Public Subnet 2
}

variable "public_subnet_cidr3" {
    description = "HLFM AZ3 Private"
    default = "172.32.32.0/20" #HLFM Production VPC Public Subnet 3
}

```

Create VPC, NGW, IGW, VPG, and Routing (vpc.tf)

This section deploys the base VPC, adds NAT, Internet and VPN gateways and deploys appropriate routing tables. Internal and External load balancers are also added to distribute requests across Web and Application servers.

```

resource "aws_vpc" "default" {
    cidr_block = "${var.vpc_cidr}"
    enable_dns_hostnames = true
    tags {
        Name = "HLFM"
    }
}

resource "aws_internet_gateway" "default" {
    vpc_id = "${aws_vpc.default.id}"
}

resource "aws_vpn_gateway" "vpg" {
    vpc_id = "${aws_vpc.default.id}"

    tags {
        Name = "virtual private gateway"
    }
}

resource "aws_eip" "nat1a" {
    vpc = true
}

resource "aws_eip" "nat1b" {
    vpc = true
}

resource "aws_eip" "nat1c" {

```

```

    vpc      = true
}

resource "aws_nat_gateway" "gw1a" {
    allocation_id = "${aws_eip.nat1a.id}"
    subnet_id     = "${aws_subnet.us-east-1a-public.id}"
}

resource "aws_nat_gateway" "gw1b" {
    allocation_id = "${aws_eip.nat1b.id}"
    subnet_id     = "${aws_subnet.us-east-1b-public.id}"
}

resource "aws_nat_gateway" "gw1c" {
    allocation_id = "${aws_eip.nat1c.id}"
    subnet_id     = "${aws_subnet.us-east-1c-public.id}"
}

/*
Private Subnet 1
*/
resource "aws_subnet" "us-east-1a-private" {
    vpc_id = "${aws_vpc.default.id}"

    cidr_block = "${var.private_subnet_cidr1}"
    availability_zone = "us-east-1a"

    tags {
        Name = "Private Subnet 1"
    }
}

resource "aws_route_table" "us-east-1a-private" {
    vpc_id = "${aws_vpc.default.id}"

    route {
        cidr_block = "0.0.0.0/0"
        gateway_id = "${aws_eip.nat1a.id}"
    }

    tags {
        Name = "Private 1 to Internet"
    }
}

resource "aws_route_table_association" "us-east-1a-private" {
    subnet_id = "${aws_subnet.us-east-1a-private.id}"
    route_table_id = "${aws_route_table.us-east-1a-private.id}"
}

```

```

/*
  Private Subnet 2
*/
resource "aws_subnet" "us-east-1b-private" {
  vpc_id = "${aws_vpc.default.id}"

  cidr_block = "${var.private_subnet_cidr2}"
  availability_zone = "us-east-1b"

  tags {
    Name = "Private Subnet 2"
  }
}

resource "aws_route_table" "us-east-1b-private" {
  vpc_id = "${aws_vpc.default.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_eip.nat1b.id}"
  }

  tags {
    Name = "Private 2 to Internet"
  }
}

resource "aws_route_table_association" "us-east-1b-private" {
  subnet_id = "${aws_subnet.us-east-1b-private.id}"
  route_table_id = "${aws_route_table.us-east-1b-private.id}"
}

/*
  Private Subnet 3
*/
resource "aws_subnet" "us-east-1c-private" {
  vpc_id = "${aws_vpc.default.id}"

  cidr_block = "${var.private_subnet_cidr3}"
  availability_zone = "us-east-1c"

  tags {
    Name = "Private Subnet 3"
  }
}

resource "aws_route_table" "us-east-1c-private" {
  vpc_id = "${aws_vpc.default.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_eip.nat1c.id}"
  }
}

```

```

    tags {
      Name = "Private 3 to Internet"
    }
  }
resource "aws_route_table_association" "us-east-1c-private" {
  subnet_id = "${aws_subnet.us-east-1c-private.id}"
  route_table_id = "${aws_route_table.us-east-1c-private.id}"
}

/*
  Public Subnet 1
*/
resource "aws_subnet" "us-east-1a-public" {
  vpc_id = "${aws_vpc.default.id}"

  cidr_block = "${var.public_subnet_cidr1}"
  availability_zone = "us-east-1a"

  tags {
    Name = "Public Subnet 1"
  }
}

resource "aws_route_table" "us-east-1a-public" {
  vpc_id = "${aws_vpc.default.id}"

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = "${aws_internet_gateway.default.id}"
  }
  route {
    cidr_block = "10.1.0.0/24"
    gateway_id = "${aws_vpn_gateway.vpg.id}"
  }

  tags {
    Name = "Public 1 Route Table"
  }
}
resource "aws_route_table_association" "us-east-1a-public" {
  subnet_id = "${aws_subnet.us-east-1a-public.id}"
  route_table_id = "${aws_route_table.us-east-1a-public.id}"
}

```

```

/*
Public Subnet 2
*/
resource "aws_subnet" "us-east-1b-public" {
    vpc_id = "${aws_vpc.default.id}"

    cidr_block = "${var.public_subnet_cidr2}"
    availability_zone = "us-east-1b"

    tags {
        Name = "Public Subnet 2"
    }
}

resource "aws_route_table" "us-east-1b-public" {
    vpc_id = "${aws_vpc.default.id}"

    route {
        cidr_block = "0.0.0.0/0"
        gateway_id = "${aws_internet_gateway.default.id}"
    }
    route {
        cidr_block = "10.1.0.0/24"
        gateway_id = "${aws_vpn_gateway.vpg.id}"
    }
    tags {
        Name = "Public 2 Route Table"
    }
}

resource "aws_route_table_association" "us-east-1b-public" {
    subnet_id = "${aws_subnet.us-east-1b-public.id}"
    route_table_id = "${aws_route_table.us-east-1b-public.id}"
}

/*
Public Subnet 3
*/
resource "aws_subnet" "us-east-1c-public" {
    vpc_id = "${aws_vpc.default.id}"

    cidr_block = "${var.public_subnet_cidr3}"
    availability_zone = "us-east-1c"

    tags {
        Name = "Public Subnet 3"
    }
}

resource "aws_route_table" "us-east-1c-public" {
    vpc_id = "${aws_vpc.default.id}"

    route {
        cidr_block = "0.0.0.0/0"
    }
}

```



```

        gateway_id = "${aws_internet_gateway.default.id}"
    }
    route {
        cidr_block = "10.1.0.0/24"
        gateway_id = "${aws_vpn_gateway.vpg.id}"
    }
    tags {
        Name = "Public 3 Route Table"
    }
}
resource "aws_route_table_association" "us-east-1c-public" {
    subnet_id = "${aws_subnet.us-east-1c-public.id}"
    route_table_id = "${aws_route_table.us-east-1c-public.id}"
}

#Administrative access ACL for remote sites via SSH, attached to private subnets
resource "aws_network_acl" "AdminToHLFM" {
    vpc_id = "${aws_vpc.default.id}"
    subnet_ids = ["us-east-1a-private", "us-east-1b-private", "us-east-1c-private"]
    egress {
        protocol    = "tcp"
        rule_no     = 2
        action      = "allow"
        cidr_block  = "10.1.0.0/24"
        from_port   = 22
        to_port     = 22
    }

    ingress {
        protocol    = "tcp"
        rule_no     = 1
        action      = "allow"
        cidr_block  = "10.1.0.0/24"
        from_port   = 22
        to_port     = 22
    }

    tags {
        Name = "AdminToHLFM"
    }
}

```

```

#Internal elastic load balancer between HLFM and RDS
resource "aws_elb" "hlfmelb" {
  name                = "hlfmelb"
  availability_zones  = ["us-east-1a", "us-east-1b", "us-east-1c"]
  security_groups     = ["hlfm"]
  access_logs {
    bucket      = "adminstorage"
    bucket_prefix = "hlfmelb"
    interval    = 60
  }

  listener {
    instance_port      = 8009
    instance_protocol  = "tcp"
    lb_port            = 8009
    lb_protocol        = "tcp"
  }

  health_check {
    healthy_threshold      = 2
    unhealthy_threshold    = 2
    timeout                = 3
    target                 = "TCP:8009"
    interval               = 30
  }

  instances              = ["HLFM01", "HLFM02", "HLFM03"]
  cross_zone_load_balancing = true
  idle_timeout           = 400
  connection_draining    = true
  connection_draining_timeout = 400

  tags {
    Name = "hlfm-internal-elb"
  }
}

#External elastic load balancer between Public and Web
resource "aws_elb" "hlfmwebelb" {
  name                = "hlfmwebelb"
  availability_zones  = ["us-east-1a", "us-east-1b", "us-east-1c"]
  security_groups     = ["web"]
  access_logs {
    bucket      = "adminstorage"
    bucket_prefix = "hlfmwebelb"
    interval    = 60
  }

  listener {
    instance_port      = 443
    instance_protocol  = "https"
    lb_port            = 443
  }
}

```

```

    lb_protocol      = "https"
    ssl_certificate_id = "arn:aws:iam::123456789012:server-certificate/HLFMCERT"
}

health_check {
    healthy_threshold      = 2
    unhealthy_threshold    = 2
    timeout                = 3
    target                 = "HTTPS:443/"
    interval               = 30
}

instances           = ["WEB01", "WEB02", "WEB03", "WEB04", "WEB05", "WEB06"]
cross_zone_load_balancing = true
idle_timeout        = 400
connection_draining = true
connection_draining_timeout = 400

tags {
    Name = "hlfm-web-external-elb"
}
}

```

HLFM Instances (HLFM, Web, Database) (HLFM.tf)

This section deploys (3) HLFM application servers, (6) Web servers, and a 3-node Aurora database cluster. All instance deployments alternate availability zones to increase solution resiliency. These nodes are then joined to OpsWorks for Chef Automate to enable ongoing configuration management at the operating system level.

```
/*
  HumanityLink Fleet Management Servers
*/
#Security group allowing HLFM, SSH and ICMP inbound
resource "aws_security_group" "hlfm" {
  name = "vpc_hlfm"
  description = "Allow incoming HLFM connections."

  ingress { # HLFM Inbound
    from_port = 8009
    to_port = 8009
    protocol = "tcp"
    cidr_blocks = ["172.32.0.0/16"]
  }
  ingress { # SSH Inbound
    from_port = 22
    to_port = 22
    protocol = "tcp"
    cidr_blocks = ["172.32.0.0/16"]
  }
  ingress { # ICMP Inbound
    from_port = -1
    to_port = -1
    protocol = "icmp"
    cidr_blocks = ["172.32.0.0/16"]
  }
  vpc_id = "${aws_vpc.default.id}"
  tags {
    Name = "HLDM_SG"
  }
}

#Security group allowing HTTPS, SSH and ICMP inbound for Web
resource "aws_security_group" "web" {
  name = "vpc_hlfm"
  description = "Allow incoming web connections."

  ingress { # HTTPS Inbound
    from_port = 443
    to_port = 443
    protocol = "tcp"
    cidr_blocks = ["172.32.0.0/16"]
  }
  ingress { # SSH Inbound
    from_port = 22
    to_port = 22
  }
}
```

```

        protocol = "tcp"
        cidr_blocks = ["172.32.0.0/16"]
    }
    ingress { # ICMP Inbound
        from_port = -1
        to_port = -1
        protocol = "icmp"
        cidr_blocks = ["172.32.0.0/16"]
    }
    vpc_id = "${aws_vpc.default.id}"
    tags {
        Name = "WEB_SG"
    }
}

```

#Security group allowing SSH and ICMP inbound for Bastions

```

resource "aws_security_group" "bastion" {
    name = "vpc_bastion"
    description = "Allow incoming web connections."

    ingress { # SSH Inbound
        from_port = 22
        to_port = 22
        protocol = "tcp"
        cidr_blocks = ["10.1.0.0/24"]
    }
    ingress { # ICMP Inbound
        from_port = -1
        to_port = -1
        protocol = "icmp"
        cidr_blocks = ["172.32.0.0/16"]
    }
    vpc_id = "${aws_vpc.default.id}"
    tags {
        Name = "BASTION_SG"
    }
}

```

#Instances specified in sequence in order to alternate deployment across all (3)

Availability Zones

#HLFM Application Tier Servers

```

resource "aws_instance" "HLFM01" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1a"
    instance_type = "c4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hlfm.id}"]
    subnet_id = "${aws_subnet.us-east-1a-private.id}"
    source_dest_check = false
}

```

#GP2 EBS volume attachment

```

root_block_device {
    volume_type = "gp2"
}

```

```

        volume_size = 100
        delete_on_termination = "true"
    }
    #Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks02-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hlfm"
        user_key        = "xxxxxxx"
        node_name       = "HLFM01"
        run_list        = [ "HLFM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type    = "ssh"
        user    = "svc_chef"
        password = "xxxxxxx"
    }
    #Tags to be used by CodeDeploy
    tags {
        Application = "HLFM"
    }
}
resource "aws_instance" "HLFM02" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1b"
    instance_type = "c4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hlfm.id}"]
    subnet_id = "${aws_subnet.us-east-1b-private.id}"
    source_dest_check = false
    #GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 100
        delete_on_termination = "true"
    }
    #Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks02-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hlfm"
        user_key        = "xxxxxxx"
        node_name       = "HLFM02"
        run_list        = [ "HLFM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type    = "ssh"
        user    = "svc_chef"
        password = "xxxxxxx"
    }
}

```

```

    #Tags to be used by CodeDeploy
    tags {
        Application = "HLFM"
    }
}

resource "aws_instance" "HLFM03" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1c"
    instance_type = "c4.2xlarge"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.hlfm.id}"]
    subnet_id = "${aws_subnet.us-east-1c-private.id}"
    source_dest_check = false
    #GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 100
        delete_on_termination = "true"
    }
    #Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks02-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hlfm"
        user_key        = "xxxxxxx"
        node_name       = "HLFM03"
        run_list        = [ "HLFM_Role" ]
        version         = "12.15.8"
    }
    connection {
        type      = "ssh"
        user      = "svc_chef"
        password  = "xxxxxxx"
    }
}

```

```

    #Tags to be used by CodeDeploy
    tags {
        Application = "HLFM"
    }
}

#HLFM Web Tier Servers

resource "aws_instance" "WEB01" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1a"
    instance_type = "m4.large"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.web.id}"]
    subnet_id = "${aws_subnet.us-east-1a-public.id}"
    source_dest_check = false
    associate_public_ip_address = true
    #GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
    #Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks02-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hlfm"
        user_key        = "xxxxxxx"
        node_name       = "WEB01"
        run_list        = [ "WEB_Role" ]
        version         = "12.15.8"
    }
    connection {
        type      = "ssh"
        user      = "svc_chef"
        password  = "xxxxxxx"
    }
    #Tags to be used by CodeDeploy
    tags {
        Application = "HLFM_Web"
    }
}

resource "aws_instance" "WEB02" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1b"
    instance_type = "m4.large"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.web.id}"]
    subnet_id = "${aws_subnet.us-east-1b-public.id}"
    source_dest_check = false

```



```

        associate_public_ip_address = true
#GP2 EBS volume attachment
        root_block_device {
            volume_type = "gp2"
            volume_size = 60
            delete_on_termination = "true"
        }
#Attachment to OpsWorks for Chef Automate
        provisioner "chef" {
            server_url      = "https://opsworks02-liznh147rllbmmk5.us-east-
1.opsworks-cm.io/"
            user_name      = "hlfm"
            user_key       = "xxxxxxx"
            node_name      = "WEB02"
            run_list       = [ "WEB_Role" ]
            version        = "12.15.8"
        }
        connection {
            type          = "ssh"
            user          = "svc_chef"
            password      = "xxxxxxx"
        }
#Tags to be used by CodeDeploy
        tags {
            Application = "HLFM_Web"
        }
    }

resource "aws_instance" "WEB03" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1c"
    instance_type = "m4.large"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.web.id}"]
    subnet_id = "${aws_subnet.us-east-1c-public.id}"
    source_dest_check = false
    associate_public_ip_address = true
#GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
#Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks02-liznh147rllbmmk5.us-east-
1.opsworks-cm.io/"
        user_name      = "hlfm"
        user_key       = "xxxxxxx"
        node_name      = "WEB03"
        run_list       = [ "WEB_Role" ]
        version        = "12.15.8"
    }

```

```

    }
    connection {
        type      = "ssh"
        user      = "svc_chef"
        password   = "xxxxxxx"
    }
    #Tags to be used by CodeDeploy
    tags {
        Application = "HLFM_Web"
    }
}

resource "aws_instance" "WEB04" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1a"
    instance_type = "m4.large"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.web.id}"]
    subnet_id = "${aws_subnet.us-east-1a-public.id}"
    source_dest_check = false
    associate_public_ip_address = true
    #GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
    #Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks02-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hlfm"
        user_key        = "xxxxxxx"
        node_name       = "WEB04"
        run_list        = [ "WEB_Role" ]
        version         = "12.15.8"
    }
    connection {
        type      = "ssh"
        user      = "svc_chef"
        password   = "xxxxxxx"
    }
    #Tags to be used by CodeDeploy
    tags {
        Application = "HLFM_Web"
    }
}

```

```

resource "aws_instance" "WEB05" {
  ami = "${lookup(var.amis, var.aws_region)}"
  availability_zone = "us-east-1b"
  instance_type = "m4.large"
  key_name = "${var.aws_key_name}"
  vpc_security_group_ids = ["${aws_security_group.web.id}"]
  subnet_id = "${aws_subnet.us-east-1b-public.id}"
  source_dest_check = false
  associate_public_ip_address = true
#GP2 EBS volume attachment
  root_block_device {
    volume_type = "gp2"
    volume_size = 60
    delete_on_termination = "true"
  }
#Attachment to OpsWorks for Chef Automate
  provisioner "chef" {
    server_url      = "https://opsworks02-liznh147rllbmmk5.us-east-
1.opsworks-cm.io/"
    user_name       = "hlfm"
    user_key        = "xxxxxxx"
    node_name       = "WEB05"
    run_list        = [ "WEB_Role" ]
    version         = "12.15.8"
  }
  connection {
    type      = "ssh"
    user      = "svc_chef"
    password  = "xxxxxxx"
  }
#Tags to be used by CodeDeploy
  tags {
    Application = "HLFM_Web"
  }
}

resource "aws_instance" "WEB06" {
  ami = "${lookup(var.amis, var.aws_region)}"
  availability_zone = "us-east-1c"
  instance_type = "m4.large"
  key_name = "${var.aws_key_name}"
  vpc_security_group_ids = ["${aws_security_group.web.id}"]
  subnet_id = "${aws_subnet.us-east-1c-public.id}"
  source_dest_check = false
  associate_public_ip_address = true
#GP2 EBS volume attachment
  root_block_device {
    volume_type = "gp2"
    volume_size = 60
    delete_on_termination = "true"
  }
}

```

```

#Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks02-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hlfm"
        user_key        = "xxxxxxx"
        node_name       = "WEB06"
        run_list        = [ "WEB_Role" ]
        version         = "12.15.8"
    }
    connection {
        type    = "ssh"
        user    = "svc_chef"
        password = "xxxxxxx"
    }
    #Tags to be used by CodeDeploy
    tags {
        Application = "HLFM_Web"
    }
}
#Bastion Host deployment
resource "aws_instance" "BAS01" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1a"
    instance_type = "t2.medium"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.bastion.id}"]
    subnet_id = "${aws_subnet.us-east-1a-public.id}"
    source_dest_check = false
    associate_public_ip_address = true
}
#GP2 EBS volume attachment
root_block_device {
    volume_type = "gp2"
    volume_size = 60
    delete_on_termination = "true"
}
#Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks02-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hlfm"
        user_key        = "xxxxxxx"
        node_name       = "BAS01"
        run_list        = [ "BASTION_Role" ]
        version         = "12.15.8"
    }
    connection {
        type    = "ssh"
        user    = "svc_chef"
        password = "xxxxxxx"
    }
}

```

```

#Tags to be used by CodeDeploy
    tags {
        Application = "HLFM_Bastion"
    }
}

resource "aws_instance" "BAS02" {
    ami = "${lookup(var.amis, var.aws_region)}"
    availability_zone = "us-east-1c"
    instance_type = "t2.medium"
    key_name = "${var.aws_key_name}"
    vpc_security_group_ids = ["${aws_security_group.bastion.id}"]
    subnet_id = "${aws_subnet.us-east-1c-public.id}"
    source_dest_check = false
    associate_public_ip_address = true
#GP2 EBS volume attachment
    root_block_device {
        volume_type = "gp2"
        volume_size = 60
        delete_on_termination = "true"
    }
#Attachment to OpsWorks for Chef Automate
    provisioner "chef" {
        server_url      = "https://opsworks02-liznh147r1lbmmk5.us-east-
1.opsworks-cm.io/"
        user_name       = "hlfm"
        user_key        = "xxxxxxx"
        node_name       = "BAS02"
        run_list        = [ "BASTION_Role" ]
        version         = "12.15.8"
    }
    connection {
        type      = "ssh"
        user      = "svc_chef"
        password  = "xxxxxxx"
    }
#Tags to be used by CodeDeploy
    tags {
        Application = "HLFM_Bastion"
    }
}

```

#Aurora Database Cluster and instance creation for HLFM

```
resource "aws_rds_cluster_instance" "cluster_instances" {
  count          = 3
  identifier     = "hlfm-db-${count.index}"
  cluster_identifier = "${aws_rds_cluster.default.id}"
  instance_class = "db.r3.2xlarge"
}

resource "aws_rds_cluster" "default" {
  cluster_identifier = "hlfm-db"
  availability_zones = ["us-east-1a", "us-east-1b", "us-east-1c"]
  database_name      = "HLFM"
  master_username    = "master"
  master_password    = "xxxxxxxx"
}
```

S3 Buckets with Lifecycle Rules

This section creates S3 buckets to be used by HLFM, including lifecycle rules for data management.

#Creates base S3 buckets to be used by HLFM, including lifecycle rule for Infrequent Access

```
resource "aws_s3_bucket" "HealthInput" {
  bucket = "HealthInput"
  acl    = "private"

  versioning {
    enabled = true
  }
  lifecycle_rule {
    id       = "HealthToIA"
    prefix   = "*"
    enabled  = true

    transition {
      days      = 30
      storage_class = "STANDARD_IA"
    }
  }
}

resource "aws_s3_bucket" "RepairOutput" {
  bucket = "RepairOutput"
  acl    = "private"

  versioning {
    enabled = true
  }
  lifecycle_rule {
    id       = "RepairToIA"
    prefix   = "*"
  }
}
```

```
enabled = true
transition {
  days      = 30
  storage_class = "STANDARD_IA"
}
}
```

Terraform Test Run

Results: Terraform plan **successfully** runs with the outlined environment build code. See embedded TXT file for results

```
C:\AWS\terraform>terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.
```

[Full content omitted – see attached file]



hlfm_terraform_vali
dation.txt

Plan: 63 to add, 0 to change, 0 to destroy.

Operating System Configuration Plan

Using OpsWorks for Chef Automate, perform basic configuration of the OS, install roles and updates. **(RM04, RM07)** Primary objectives are being listed, as custom recipes and cookbooks require development.

HumanityLink Fleet Management Role Objectives (HLFM_Role)

- Assign HLFM role to registered nodes
- Set system hostname
- Install AWS CodeDeploy agent for continuous app deployment
- Deploy latest ODBC/JDBC driver
- Connect to Aurora database and test connectivity
- Check currently deployed version of HLFM against repository
- Deploy new version of HLFM from repository, if required
- Check current state of system updates against repository
- Deploy new system updates from repository, if required
- Test connectivity to other nodes in the farm
- Test connectivity to S3 buckets
- Synchronize application configuration against S3
- Check status of HLFM services and join farm
- Query instance name, register instance with internal ELB

HumanityLink Fleet Management Web Role Objectives (WEB_Role)

- Assign Web role to registered nodes
- Set system hostname
- Install AWS CodeDeploy agent for continuous app deployment
- Check currently deployed version of NGINX and extensions against repository
- Deploy new version of NGINX and extensions from repository, if required
- Check current state of system updates against repository
- Deploy new system updates from repository, if required
- Synchronize static web content from S3
- Check status of NGINX services
- Test connectivity to HLFM application servers
- Query instance name, register instance name with external ELB

Application Deployment Pipeline

This section outlines tasks to be completed by Amazon Web Services code testing and release services. Production code repositories will be monitored for commits, and these actions will trigger a sequence of stages mediated by AWS CodePipeline.

Testing of this code will be conducted by CodeBuild, an automated, container-based solution. If testing is successful, results will be presented to administrators and deployment can be authorized.

Once approved, CodeDeploy will leverage an instance-deployed agent to update HumanityLink Fleet Management and reboot instances one at a time. Following this process, all instances will be running the latest version of HLFM. This solution results in continuous deployment for the application. **(R02)**

Code Repository

- Add CodeCommit repository to development station using GIT and clone locally
- Perform development against HumanityLink Fleet Management code base
- Commit changes to CodeCommit repository

Code Pipeline

- Monitor CodeCommit repository HLFM for commits to the production branch
- Engage CodeBuild to perform automated testing of code before deployment

Code Build

- Utilize standard Ubuntu-based containers to build and test HLFM code
- Report results to CodePipeline workflow

Code Pipeline

- If tests conducted during CodeBuild are successful, send notification to Administrators

Approval

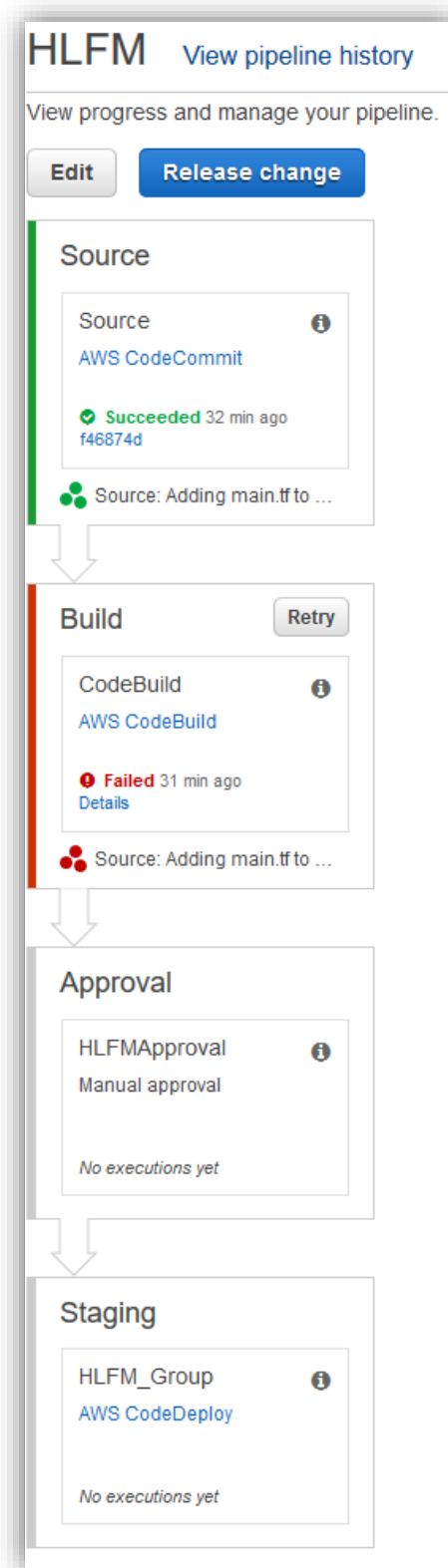
- Review results from automated CodeBuild testing process
- Provide CodePipeline permission to deploy HLFM across the instance fleet

Code Pipeline

- Engage CodeDeploy service to execute deployment of HLFM code across instance fleet

Code Deploy

- Update installations of HLFM across the instance fleet
 - Apply to instances with tag HLFM = True
 - Update one instance at a time
- Deployment of the latest version of HLFM is complete.



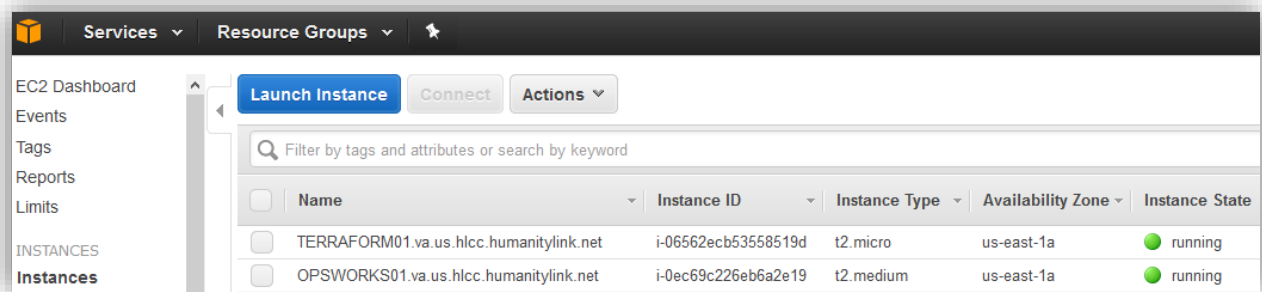
Orchestrated deployment phases resulting in continuous deployment for HLFM.

Operational Procedures

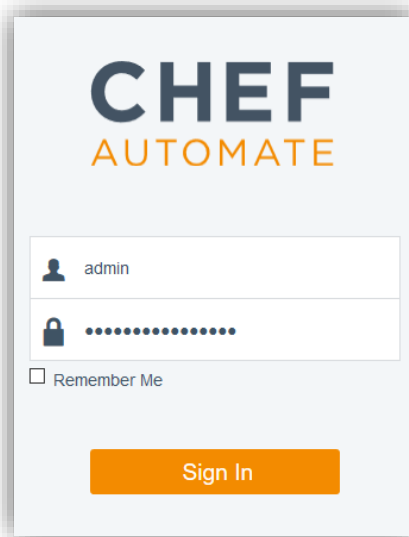
Provided Operational Procedures correspond to requirement **RM06**.

Preparation Tasks

- 1) Ensure OpsWorks and Terraform administrative servers are running
 - a. Log in to the AWS Console, select **Services > EC2**
 - b. Verify required instances are running
 - i. OPSWORKS01
 - ii. TERRAFORM01



- 2) Verify the OpsWorks for Chef Automate server is accessible
 - a. <https://opsworks01-xxxxxxxxxxxxxxxx.us-east-1.opsworks-cm.io/viz/#/>



- 3) Verify the ability to SSH to the Terraform management server

```
centos@ip-172-254-10-34/tmp/opsworks01-nbdxwczilpqzba
[centos@ip-172-254-10-34 opsworks01-nbdxwczilpqzba]$ hostname
TERRAFORM01.va.us.hlcc.humanitylink.net
[centos@ip-172-254-10-34 opsworks01-nbdxwczilpqzba]$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 9001
    inet 172.254.10.34 netmask 255.255.240.0 broadcast 172.254.15.255
```

- 4) Install Terraform on the Terraform management server

#Change to temporary directory

Cd /tmp

#Download zipped binary

Wget https://releases.hashicorp.com/terraform/0.9.11/terraform_0.9.11_linux_amd64.zip

#Unzip Terraform binary

unzip terraform_0.9.11_linux_amd64.zip

#Test run Terraform (no install required)

Terraform

```
[centos@ip-172-254-10-34 terraform]$ terraform
Usage: terraform [--version] [--help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.

Common commands:
  apply          Builds or changes infrastructure
  console        Interactive console for Terraform interpolations
  destroy        Destroy Terraform-managed infrastructure
  env            Environment management
  fmt            Rewrites config files to canonical format
  get            Download and install modules for the configuration
  graph          Create a visual graph of Terraform resources
  import         Import existing infrastructure into Terraform
  init           Initialize a new or existing Terraform configuration
  output         Read an output from a state file
  plan           Generate and show an execution plan
  push           Upload this Terraform module to Atlas to run
  refresh        Update local state file against real resources
  show           Inspect Terraform state or plan
  taint          Manually mark a resource for recreation
  untaint        Manually unmark a resource as tainted
  validate       Validates the Terraform files
  version        Prints the Terraform version

All other commands:
  debug          Debug output management (experimental)
  force-unlock   Manually unlock the terraform state
  state          Advanced state management
```

Successful download and execution of Terraform

5) Install GIT

#Use YUM to install GIT, then test run

`Sudo yum install git`

`git`

```
[centos@ip-172-254-10-34 terraform]$ git
usage: git [--version] [--help] [-c name=value]
       [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
       [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
       [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
       <command> [<args>]
```

Successful execution of GIT following installation

6) Install Text Editor

#Add Microsoft Visual Studio Code YUM repository

`sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc`

`sudo sh -c 'echo -e "[code]\nname=Visual Studio\nCode\nbaseurl=https://packages.microsoft.com/yumrepos/vscode\nenabled=1\nngpgcheck=1\nngpgkey=https://packages.microsoft.com/keys/microsoft.asc" > /etc/yum.repos.d/vscode.repo'`

#Install Visual Studio Code from YUM repository

`yum check-update`

`sudo yum install code`

7) Attach to Infrastructure Git Repository

#Set GIT Credentials

`git config --global credential.helper '!aws codecommit credential-helper $@'`

`git config --global credential.UseHttpPath true`

#Attach to GIT Repository

`git clone https://git-codecommit.us-east-1.amazonaws.com/v1/repos/infra`

```
[centos@ip-172-254-10-34 tmp]$ git clone https://git-codecommit.us-east-1.
amazonaws.com/v1/repos/infra
Cloning into 'infra'...
warning: You appear to have cloned an empty repository.
[centos@ip-172-254-10-34 tmp]$ cd infra/
[centos@ip-172-254-10-34 infra]$
```

Attachment and cloning of a remote CodeCommit repository.

Managing Terraform Code Versions

Enable Terraform Remote State

In order to meet infrastructure management requirements and enable version control for Terraform state, this deployment will utilize an S3 bucket with versioning enabled for back-end storage.

Should a serious issue occur, Terraform state can be restored from an earlier version. Follow the instructions below to perform this initial setup.

Procedure

- 1) Create a new file that will contain remote state configuration
 - a. This example will use "statefile.tf"
- 2) Insert appropriate configuration instructing Terraform to store state in S3

```
data "terraform_remote_state" "network" {  
  backend = "s3"  
  config {  
    bucket = "humanitylinkadmin"  
    region = "us-east-1"  
    access_key = "xxxxxxxxxxxxxxxxx"  
    secret_key = "xxxxxxxxxxxxxxxxx"  
  }  
}
```

- 3) Re-run Terraform Init to initialize the remote state file

```
C:\AWS\terraform>terraform init  
  
Terraform has been successfully initialized!  
  
You may now begin working with Terraform. Try running "terraform plan" to see  
any changes that are required for your infrastructure. All Terraform commands  
should now work.  
  
If you ever set or change modules or backend configuration for Terraform,  
rerun this command to reinitialize your environment. If you forget, other  
commands will detect it and remind you to do so if necessary.
```

- 4) Run Terraform State Push to sync changes to the remote file
- 5) Should a serious issue occur, restore a previous version of the state file using the S3 console.
- 6) Establishment of remote state for Terraform is complete.

Manage and Modify Terraform Code

- 1) Copy Terraform files into the GIT directory recently cloned

Example: `Cp /tmp/terraform /tmp/infra`

- 2) List directory content to verify Terraform TF files are present

```
[centos@ip-172-254-10-34 infra]$ ls -lah | grep tf
-rw-rw-r--. 1 centos centos 139 Jul 16 02:24 aws.tf
-rw-rw-r--. 1 centos centos 14K Jul 16 03:15 hldm.tf
-rw-rw-r--. 1 centos centos 15K Jul 16 04:59 hlfm.tf
-rw-rw-r--. 1 centos centos 735 Jul 16 04:14 s3.tf
-rw-rw-r--. 1 centos centos 155 Jul 16 02:24 terraform.tfvars
-rw-rw-r--. 1 centos centos 1.4K Jul 16 04:29 variables.tf
-rw-rw-r--. 1 centos centos 7.9K Jul 16 05:14 vpc.tf
```

- 3) Perform initial synchronization of these files to the repo to establish the master branch

#Add directory contents to tracked file list

Git add .

#Check status of tracked files

Git status

#Commit Terraform files to repository

Git commit -m "Initial Terraform commit"

```
8 files changed, 2799 insertions(+)
create mode 100644 aws.tf
create mode 100644 hldm.tf
create mode 100644 hlfm-test.txt
create mode 100644 hlfm.tf
create mode 100644 s3.tf
create mode 100644 terraform.tfvars
create mode 100644 variables.tf
create mode 100644 vpc.tf
```

Terraform files committed to Infrastructure repository.

#Push content of local repository to remote CodeCommit repository

Git push origin master

```
[centos@ip-172-254-10-34 infra]$ git push origin master
Counting objects: 13, done.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (13/13), 9.55 KiB | 0 bytes/s, done.
Total 13 (delta 4), reused 0 (delta 0)
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/infra
 * [new branch]      master -> master
```

- 4) Perform ongoing adjustment of code within Terraform files, as required.

```
centos@ip-172-254-10-34:/tmp/infra
[centos@ip-172-254-10-34 infra]$ nano -w vpc.tf
GNU nano 2.3.1 File: vpc.tf

resource "aws_vpc" "default" {
  cidr_block = "${var.vpc_cidr}"
  enable_dns_hostnames = true
  tags {
    Name = "HLFMVPC"
  }
}
```

- 5) Commit changed Terraform files to the CloudCommit repository

#Commit changes to Repository

Git commit -m "First Terraform Revision"

Git push origin master

```
[centos@ip-172-254-10-34 infra]$ git commit -m "First Revision"
[master 408bddf] First Revision
Committer: Cloud User <centos@TERRAFORM01.va.us.hlcc.humanitylink.net>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 2 insertions(+), 2 deletions(-)
```

Synchronization of Terraform code changes to Git repository.

- 6) Execute Terraform Plan to ensure changes to Terraform code are valid

#Execute Terraform plan

Terraform plan

```
Plan: 50 to add, 0 to change, 0 to destroy.
[centos@ip-172-254-10-34 infra]$
```

- 7) Adjustment of Terraform code and capture of changes within the code repository is complete.

Managing Chef Cookbook Versions

- 1) Launch the ChefDK from an administrative workstation
 - a. This example uses a Windows-based platform
- 2) Configure AWS CLI credentials

#Configure AWS Credentials

aws configure

```
PS C:\aws\config> aws configure
AWS Access Key ID [*****HCOA]:
AWS Secret Access Key [*****HzX1]:
Default region name [us-east-1]:
Default output format [json]:
```

- 3) Configure GIT to utilize AWS credentials

#Configure GIT Credential Manager

git config --global credential.helper "!aws codecommit credential-helper \$@"

*git config --global credential.UseHttpPath true*Locate

- 4) Navigate to a local folder suitable to serve as the Cookbook repository
- 5) Create a local clone of the CodeCommit repository intended to store Cookbooks

git clone <https://git-codecommit.us-east-1.amazonaws.com/v1/repos/config>

- 6) Download a copy of the Cookbooks intended to be used and unzip contents
 - a. knife cookbook site download nginx
 - b. knife cookbook site download java

```
PS C:\aws\config> knife cookbook site download nginx
WARNING: No knife configuration file found
Downloading nginx from Supermarket at version 2.7.6 to C:/aws/config/nginx-2.7.6.tar.gz
Cookbook saved: C:/aws/config/nginx-2.7.6.tar.gz

PS C:\aws\config> knife cookbook site download java
WARNING: No knife configuration file found
Downloading java from Supermarket at version 1.50.0 to C:/aws/config/java-1.50.0.tar.gz
Cookbook saved: C:/aws/config/java-1.50.0.tar.gz
```

- 7) Add Cookbook content to Git repository

Git add .

git commit -m "Initial Cookbook upload"

```
130 files changed, 8444 insertions(+)
create mode 100644 java/.foodcritic
create mode 100644 java/.gitignore
create mode 100644 java/.kitchen.macOS.yml
```

8) Push Cookbook content to the Git repository

Git push origin master

```
PS C:\aws\config> git push origin master
Counting objects: 153, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (139/139), done.
Writing objects: 100% (153/153), 109.49 KiB | 0 bytes/s, done.
Total 153 (delta 37), reused 0 (delta 0)
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/config
* [new branch]      master -> master
```

9) Make ongoing adjustments to Cookbook content, as required

```
# Author:: Bryan W. Berry (<bryan.berry@gmail.com>)
# Cookbook:: java
# Recipe:: oracle_1386
#
# Copyright:: 2010-2015, Chef Software, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

include_recipe 'java::notify'

unless node.recipe?('java::default')
  Chef::Log.warn('Using java::default instead is recommended.')

  # Even if this recipe is included by itself, a safety check is nice...
  if node['java']['java_home'].nil? || node['java']['java_home'].empty?
    include_recipe 'java::set_attributes_from_version'
  end
end
```

10) Commit changes to Git repository

Git commit -m "Revised Java recipe"

```
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   java/recipes/openjdk.rb
        modified:   nginx/recipes/headers_more_module.rb
```

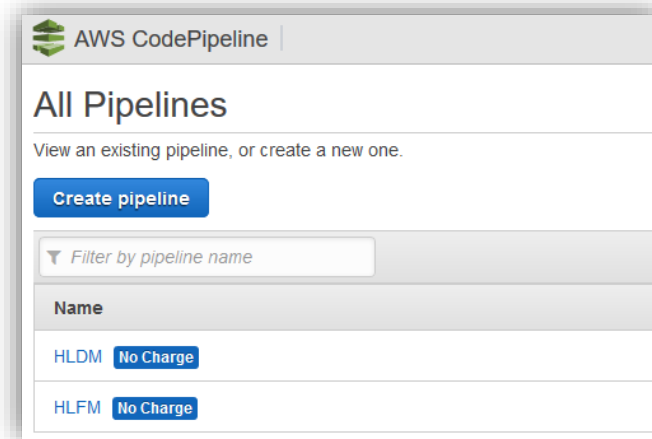
11) Push updated content to the repository to complete the versioning process.

```
PS C:\aws\config> git push origin master
```

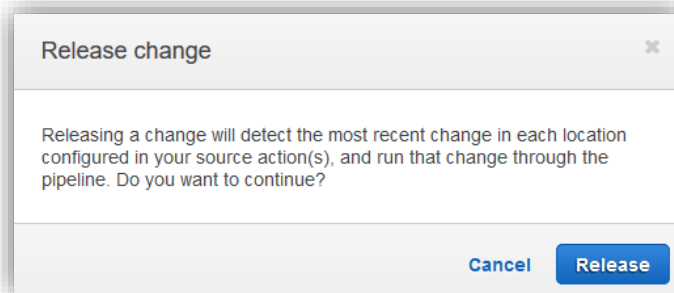
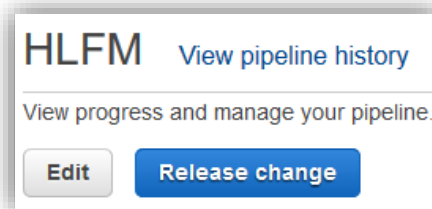
Managing Application Code and Deployment

Release Changes to the Pipeline

- 1) Commit and push code changes to the GIT repository associated with the application
 - a. HLDM - <https://git-codecommit.us-east-1.amazonaws.com/v1/repos/hldm>
 - b. HLFM - <https://git-codecommit.us-east-1.amazonaws.com/v1/repos/hlfm>
- 2) Log in to the AWS Console and select **Services > CodePipeline**



- 3) Select the pipeline associated with the application in question
- 4) Select Release Change and confirm release to force changed code through the pipeline



- 5) Monitor the History tab within CodePipeline to assess results of the code release.

| Execution ID | Status | Start time |
|--|--------|----------------------|
| f095c02a-7589-4d4a-8bfb-77d2df75b760 Source: Adding Java test | Failed | Jul 16, 2017 7:55 AM |

References

The following documentation was referenced in the creation of this solution design:

- [Adam Post – VDM Challenge 1 – HLDM / HLFM Architecture](#)
- [Adam Post – VDM Challenge 2 – Security Improvements](#)
- [Terraform – Beyond the Basics Blog](#)
- [Terraform – AWS VPC with Public/Private Subnets](#)
- [Terraform and Chef Together – How to Provision](#)
- [Srapbag of Useful Terraform Tips – Charity.wtf](#)
- [Get Set Up with AWS OpsWorks for Chef Automate](#)
- [HashiCorp Terraform Documentation](#)
- [AWS OpsWorks for Chef Automate Documentation](#)
- [AWS CodeCommit Documentation](#)
- [AWS CodePipeline Documentation](#)
- [AWS CodeDeploy Documentation](#)
- [AWS Command Line Interface Documentation](#)
- [AWS VPC Documentation](#)
- [AWS S3 Documentation](#)
- [AWS RDS Documentation](#)