Virtual Design Master

Challenge 1 – Design and Orchestration

Timothy J. Patterson

ABSTRACT

In order to save all of Mankind, a highly available and portable batch processing data manufacturing system is required. This application has been built on top of the only remaining AWS region in the world and has the power and flexibility to save us from a world of brain eating zombies.

Table of Contents

1. Executive Summary

The world is once again in disarray. Society, as we know it has fallen, and the zombie outbreak is back with a vengeance. Our billionaire friend has planned for this and his space tourism company has built a large base on the moon that is designed to support what is left of the human race. We have been tasked with building a highly scalable and easily deployable application and its corresponding infrastructure for manufacturing purposes as quickly as possible.

1.1. Scope

Currently, there is only one space ship depot on Earth, located in Cape Canaveral, Florida in the U.S.A. Three more are currently being build as fast as possible with more to follow. Each depot is capable of producing a launch ready ship in 72 hours. Larger ships are currently being designed.

The manufacturing application and its corresponding infrastructure must be completely reliable. Downtime is NOT acceptable, as ANY malfunction of the manufacturing systems is catastrophic, and will cost precious human lives.

Lucky for us, Amazon's entire us-east-1 region has thus far survived the new zombie apocalypse due to their high security and shroud of secrecy. Amazon has also been ramping up a new region for their services on the moon itself, and is estimated to be open for production workloads in a month's time.

A batch processing manufacturing application will be built on top of the AWS infrastructure. We will take advantage of many of AWS' built-in services to bolster our application's functionality, decrease our development time, and ensure high availability of core components. This application will be written in Java and will take advantage of the AWS Java SDK for interaction with the AWS managed services that will feed information into our application. We will use AWS SQS (simple queue service), AWS RDS (relational database service), AWS S3 (simple storage service), and Auto Scaling groups as supporting services.

The application will reside on the AWS EC2 (elastic cloud compute) service, and take advantage of the VPC (virtual private cloud) segregation features.

2. Current State Analysis

2.1. On-Premises Network

- LAN: We have a standard, flat LAN. We have access to the Internet via an off-the-shelf router via NAT and our WAN connection.

- WAN: We have a 100 Mbps Internet connection at our manufacturing facility. This link is only used to facilitate Internet access for manufacturing employees.

2.2. Physical Servers

There are no datacenter or servers running on premises.

2.3. Amazon Web Services

- We have maintained a great working relationship with AWS in the past. As such we have an account with no spending or provisioning limits attached to it.
- The entire us-east-1 region is available for use. We have access to three availability zones within this region: us-east-1a, us-east-1b, us-east-1c.
- Each availability zone is a separate physical datacenter, with high bandwidth and low latency interconnects.
- AWS is currently wrapping up construction on their new moon region, and should be available for use within a month's time.

3. Requirements, Assumptions, Constraints, and Risks

3.1. Requirements

- R01: The application must have a client facing web layer.
- RO2: The application must have a message queuing middle layer.
- R03: The application must have a database backend.
- R04: The application must maintain an uptime SLA of at least 99.9999%.
- R05: The application must be easily deployable.
- R06: The application must be able to scale under load.

3.2. Assumptions

- A01: Since this application is being built from scratch, we do not know what the workload characteristics and demands will look like in production.
- A02: There should be some kind of protection mechanism in place to preserve the output of the manufactured data.
- A03: The 100 Mbps link from our manufacturing facility will be enough bandwidth to communicate with the application's web layer and upload / download manufacturing jobs and data in an appropriate time frame.

3.3. Constraints

- C01: AWS only has one region active at this time. The rest have fallen to the zombies.

- C02: Since this is a new environment, there will be additional up-front work to develop the necessary provisioning scripts. Once created, the entire environment can be re-deployed within minutes.

3.4. Risks

- X01: If we lose our Internet connectivity to AWS we will be unable to communicate with the job manager web application and cannot upload the raw data needed for the processing jobs.
- X02: If the zombies breach all three datacenters our entire infrastructure will be in jeopardy.

4. Application Overview

The application being developed will take advantage of the following managed services provided by AWS:

- Amazon EC2: Elastic Compute Cloud. EC2 offers virtual machines that are booted from an AMI (Amazon Machine Image). Additionally, EC2 offers services that enhance the availability and scalability of EC2 instances. Examples of add-ons that we will use include Auto Scaling Groups, Elastic Load Balancers, and Elastic Block Storage. ELBs offer 99.9999% availability.

- Amazon S3: Simple Storage Service. S3 is a highly scalable and reliable object storage service. All data stored in S3 is guaranteed to be available at least 99.9999% of the time. Data stored in S3 is replicated automatically between availability zones within the given region.

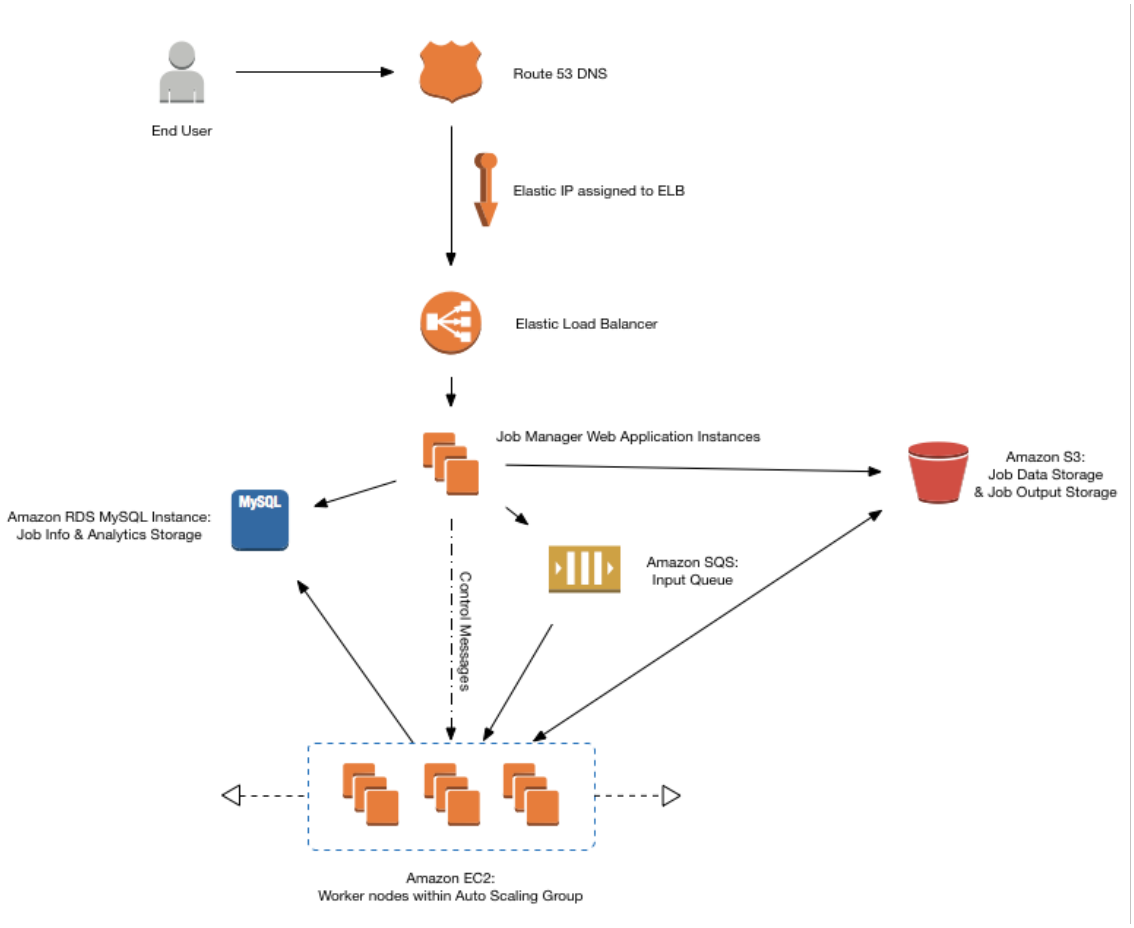
- Amazon SQS: Simple Queuing Service. SQS is a message queuing and delivery system for applications that require a messaging bus. SQS *guarantees* every message will be delivered *at least once* without data loss.

- Amazon RDS: Relational Database Service. RDS provides access to a managed database as a service. RDS simply hands you an endpoint and database credentials that you can use within your application. It is scalable and configurable. RDS can be configured to maintain multiple replicas in different availability zones. Availability can be guaranteed with use of replicas. AWS handles all database backups and upgrades automatically. RDS instances live in a subnet within your VPC.

- Amazon Route 53: Highly available, globally distributed hosted DNS service.

- Amazon Glacier: Long-term, resilient, and secure storage for archiving.

The application will be comprised of a client facing web tier residing on EC2, a message queuing middle-tier residing within both EC2 and SQS, and a database tier residing within RDS.
Please reference the following diagram (full resolution diagram found in appendix):



4.1. Application Workflow

Our application has been designed to function using this workflow:

1. A user accesses the Job Manager web application using a secured (SSL) web site.
2. The user then submits the manufacturing job details and uploads the raw data that is to be processed.
3. The Job Manager application performs three consecutive actions:
    a. Writes job metadata to a MySQL database residing within RDS. The job metadata includes: job ID, location of the raw data within the S3 bucket, and fields for job progress.
    b. Uploads the raw data that is to be processed into a S3 bucket.
    c. Inserts a message into SQS containing the job ID.

4. As seen in the above application diagram, there is a group of worker nodes running. These nodes are running a Java application that continuously polls the SQS queue looking for jobs to process. Once a job is found, a single worker node pulls the job ID out of the queue and queries the MySQL database for the job details.
5. Once a worker node has the appropriate job data it begins its processing routine.
6. During the processing cycle, the worker node writes statistical data back to the MySQL database for the given job ID (progress, ETA, etc.)
7. Once finished processing, the worker node uploads its output to a new S3 bucket labeled with the job ID.
8. Additionally, the worker node updates the MySQL database indicating the job is complete and removes the job ID from the SQS queue.

4.2. Application Details

The application consists of two custom components, the job manager web application and the processing application.

The web application is a simple PHP web site that collects and reports information to/from the end user. The web application is also responsible for ensuring the raw job data was properly uploaded into the S3 bucket and that a SQS job is submitted. The application uses the AWS PHP SDK to accomplish these tasks.

The processing application is a Java program that performs manufacturing algorithms against the raw job data. The algorithms are top secret and therefore cannot be divulged here. The processing application uses the AWS Java SDK to communicate with S3 to gather the raw job data, and SQS to retrieve jobs that require processing. Since the worker nodes run as a group, it is important for the application to also query the MySQL database after pulling a job ID. This ensures that a job is only executed once. It is equally important that the processing application also removes the SQS message from the queue once processing is complete. If a SQS message is not removed it will be re-inserted into the queue after a timeout period. These measures ensure that every message on the bus will always be processed once and only once.

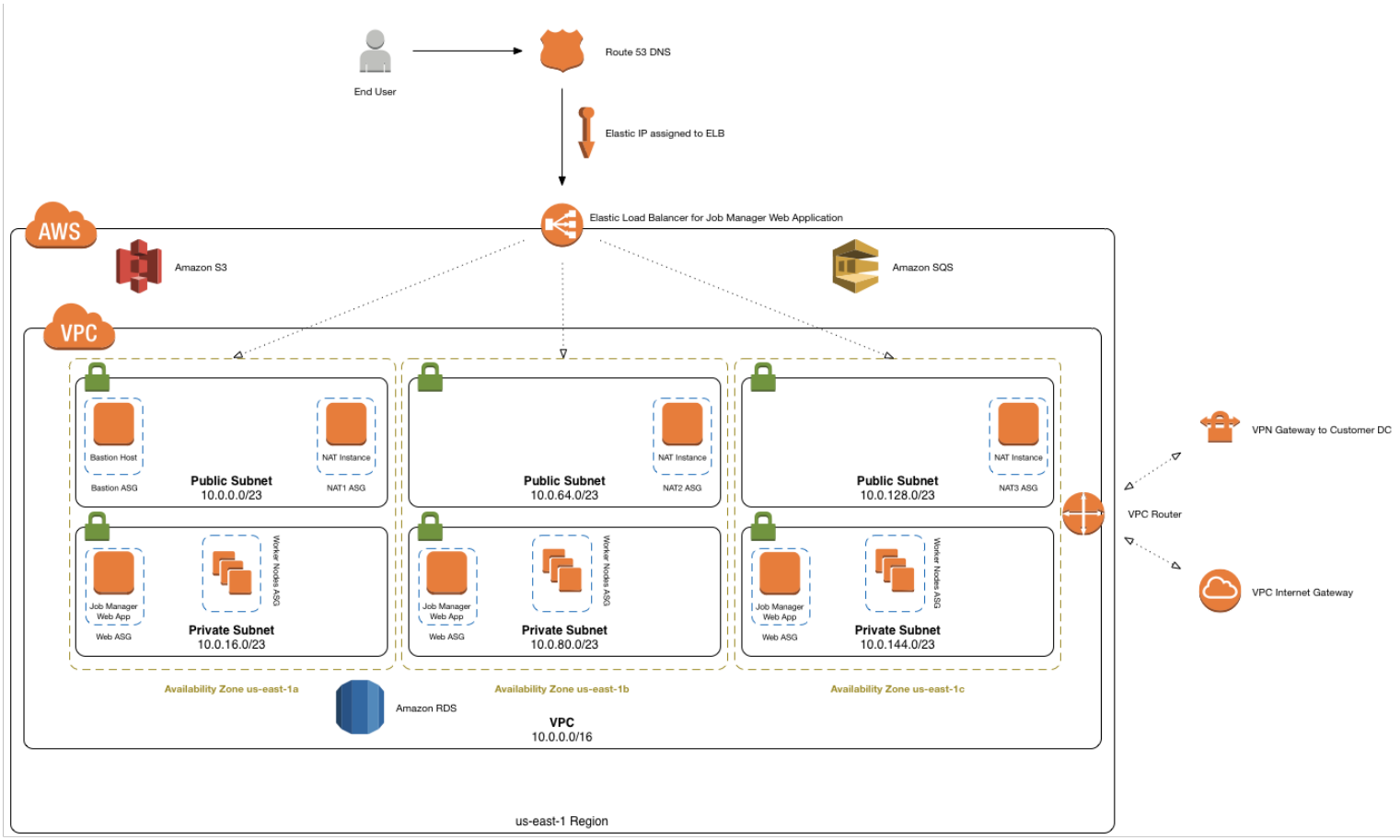
Both applications use native language tools to communicate with the MySQL database residing within RDS.

Our S3 buckets utilize lifecycle rules. These rules are configured to archive their data to AWS Glacier on a daily basis.

5. Infrastructure Overview

The application will be hosted on the AWS EC2 service, contained within its own VPC (virtual private cloud). A VPC offers complete isolation from other AWS customers and allows you to connect to AWS as an extension to your own datacenter via a VPN customer gateway.

Please reference the following diagram:



5.1. Infrastructure Design

The infrastructure is contained within its own VPC, which spans three availability zones. Each availability zone contains two subnets, a public subnet for Internet facing instances, and a private subnet for backend systems.

The VPC provides Internet access via the VPC router construct and an AWS provided VPC Internet Gateway. Note: The VPC can be optionally connected to our local manufacturing facility via a VPN gateway; however, this is not in use for us at this time and could come in handy in the future…

The VPC provides a bastion host with an Elastic IP address (EIP). The EIP allows us to gain access to the internal subnets via SSH from the Internet within the VPC

infrastructure for management purposes. The bastion host is also protected with an auto-scaling group. The auto-scaling group is configured with a maximum of one and a minimum of one, and is allowed to float between availability zones. This means that if the bastion host dies it will be re-provisioned in about 5 minutes time.

Each availability zone has its own NAT instance. These instances provide outbound Internet access to EC2 instances residing in the private subnets. This is useful for downloading OS updates, accessing provisioning scripts residing on code repositories, and interfacing with public AWS services such as S3, SQS, and AWS API endpoints. The NAT instances are all protected with their own auto-scaling groups and utilize boot scripts for their configuration. The auto-scaling groups are configured with a maximum of one and a minimum of one, and are pinned to their specific availability zones. This means that if a NAT instance dies, it will be re-provisioned within about 5 minutes time. All back end servers can still process their data, just not access the Internet or post their results until the connection is restored.

5.2. Infrastructure Details – Job Manager Web Application

All users interfacing with the application connect via the job manager web interface. In order for a user to connect to the web interface, they enter the URL into their browser. The DNS entries for the application are stored within AWS' Route 53 service. Route 53 returns a host record for an Elastic Load Balancer (ELB).

The web application resides on a set of EC2 instances, configured as follows:

- Instance type of m3.medium (1 CPU, 3.75GB RAM).
- 10GB OS boot device, EBS backed.
- 64-bit Amazon Linux (RHEL derivative).
- Apache / PHP (installed on boot).

These instances are launched and protected with an Auto Scaling Group. The Auto Scaling Group itself is comprised of two components: a launch configuration and an auto-scaling configuration. A launch configuration contains details on the type of instances you want to be launched, any user-data scripts that should be launched upon boot, and any other instance level details you wish to include (tags, etc.) The auto-scaling configuration contains the tunable values for: max instances, min instances, which ELB to attach to, and when and how to scale.

The web application has been configured to launch with the m3.medium instance type with the Amazon Linux AMI, and has a boot up script configured as follows:

```
#! /bin/bash
yum install apache php mysql git
cd /var/www/html; git clone http://<URL to code
repository>/jobmanager.git
```

This launch configuration will bootstrap our instances for us and automatically configure Apache/PHP, and download our application's code into the appropriate

directory. By launching instances in this fashion they are decoupled from the applications that they are supporting and vice versa. The instance comes pre-bundled with the AWS API utilities.

The auto-scaling configuration for the job manager instances is configured with a minimum of 2 and a maximum of 5. This ensures that there is always at least two job manager instances up and running at all times. The auto-scaling configuration is also set up to ensure that any instances it launches are connected to the ELB. Additionally, we have configured the auto-scaling group to watch the CPU load metric on our instances. If the provisioned instances become too overloaded, the auto-scaling group will provision more instances and join them to the ELB until a maximum of 5 instances have been created. The auto-scaling group will also scale down the instances if they are no longer being used to a minimum of two. The auto-scaling group is configured to always spread its instances across multiple availability zones.

The ELB technically resides within the public VPC subnets and has an EIP associated with it. Route 53 is configured with a special type of DNS record that always points at the ELB object itself. SSL certificates are installed on the ELB for security for data in flight. The ELB only allows access to the backend instance via SSL.

The job manager web application instances themselves all live within the private subnets of the VPC and are not exposed directly to the Internet.

The web application instances should be treated as throwaway components. They can be terminated and re-provisioned in very short order with no impact to the application's functionality. This enables a continuous integration model, as to push out a new code update all you have to do is terminate one instance at a time and let the auto-scaling groups re-provision your instances.

5.3. Infrastructure Details – Worker Nodes

The worker nodes watch SQS for their processing tasks via a polling process. The worker nodes are responsible for the heavy data processing tasks.

The worker nodes are comprised of a set of EC2 instances, configured as follows:

- Instance type of c3.4xlarge (16 CPUs, 30GB RAM).
- 10GB OS boot device, EBS backed.
- 2 x 160GB SSD ephemeral (local) drives.
- 64-bit Amazon Linux (RHEL derivative).
- Our custom Java processing application (installed on boot).

These instances are launched and protected with an Auto Scaling Group. The Auto Scaling Group itself is comprised of two components: a launch configuration and an auto-scaling configuration. A launch configuration contains details on the type of instances you want to be launched, any user-data scripts that should be launched upon boot, and any other instance level details you wish to include (tags, etc.) The auto-scaling configuration contains the tunable values for: max instances, min instances, what metric(s) to watch, and when and how to scale.

The worker nodes been configured to launch with the c3.4xlarge instance type with the Amazon Linux AMI, and has a boot up script configured as follows:

```
#! /bin/bash
yum install java git
mkdir /app; cd /app; git clone https://<URL to code repository>/worker.git
```

This launch configuration will bootstrap our instances for us and automatically configure Java, and download our application's code into the appropriate directory. By launching instances in this fashion they are decoupled from the applications that they are supporting and vice versa. The instance comes pre-bundled with the AWS API utilities.

The auto-scaling configuration for the worker node instances is configured with a minimum of 3 and a maximum of 75. This ensures that there are always at least three worker node instances up and running at all times. The auto-scaling group has been configured to watch the CPU load metric on the instances. If the provisioned instances become too overloaded, the auto-scaling group will provision more instances until a maximum of 75 instances have been created. The auto-scaling group will also scale down the instances if they are no longer being used to a minimum of 3. The auto-scaling group is configured to always spread its instances across all three availability zones.

The worker nodes should be treated as throwaway components. They can be terminated and re-provisioned in very short order with no impact to the application's functionality. This enables a continuous integration model, as to push out a new code update all you have to do is terminate one instance at a time and let the auto-scaling groups re-provision your instances.

6. Provisioning

All of the components for both the application its supporting infrastructure can be automatically provisioned and controlled via code in the form of CloudFormation scripts.

CloudFormation is a provisioning system created by AWS that allows an administrator to define objects using JSON notation. The scripts describe all of the components that are to be provisioned and assign properties to them. It is best practice to layer multiple CloudFormation scripts together into a stack.

The infrastructure and application in this use case will make use of CloudFormation for re-usable provisioning. By taking advantage of a code based infrastructure deployment, we can simply re-deploy the entire infrastructure and the application in one step in the future. This can even be to other regions… Remember the moon?

The CloudFormation scripts are to be layered as follows:

- Infrastructure / VPC (includes ELBs, NAT instances and bastion host)
- Managed Services (S3, SQS, RDS)
- Application Requirements (auto-scaling groups)

Since CloudFormation scripts are a form of code, it is easy to push the code for the entire infrastructure into a continuous integration (CI) model. The scripts should be version controlled and stored in a code repository (git) with multiple branches for development, pre-production, and production. Version releases should also be tagged for potential rollback within each branch.

Please refer to the appendix for an example of an infrastructure CloudFormation script.

7. Security and data retention

The entire infrastructure and application stack has been built with data security and retention in mind from the top down.

This stack employs the following security and data retention techniques:

- SSL encryption for all in-flight data from the Internet into the ELB.
- VPC segregation for our EC2 instances. This keeps them separate from all other public AWS traffic.
- The VPCs utilize security groups. Security groups act like stateful firewall devices and allow you to configure rules. Rules are applied to security groups. Instances are members of one or more security groups.
- The VPCs utilize network ACLs. Network ACLs act like router ACLs and allow you to configure rules that apply to entire subnets.
- S3 bucket lifecycle rules automatically archive already processed data and stores it into Glacier for long-term storage.

8. The future

Since this is a brand new application and we are not aware of its real-world workload requirements, this design has been created with the future in mind.

8.1. Scaling up / down

All of the instance types used in this design can be replaced with a much larger and more powerful instance type. This provides the contingency for more CPU and RAM horsepower if/when needed. Likewise, the instance types can be scaled down for periods of non-use. These concepts apply to both the web application nodes and the worker nodes. To apply a scale up/down change, simply terminate one instance at a time and let the auto-scaling groups re-provision the instances.

8.2. Scaling out/in

Just as the instances in this design can scale up/down, they can also scale horizontally either out or in. The auto-scaling groups can be configured with different values for maximum and minimum number of instances to be running. This allows for massive scalability in times of peak loads or a lean deployment during valleys. To apply a scale out/in change, simply update the parameters of your auto-scaling group. The new values take effect immediately.

8.3. Disaster recovery / Expansion into new regions

Since the entire application AND its infrastructure are fully encapsulated in code the whole stack can be easily re-provisioned in place for disaster recovery or stood up in an entirely new region! This is accomplished by simply executing the CloudFormation scripts to form the new stack.

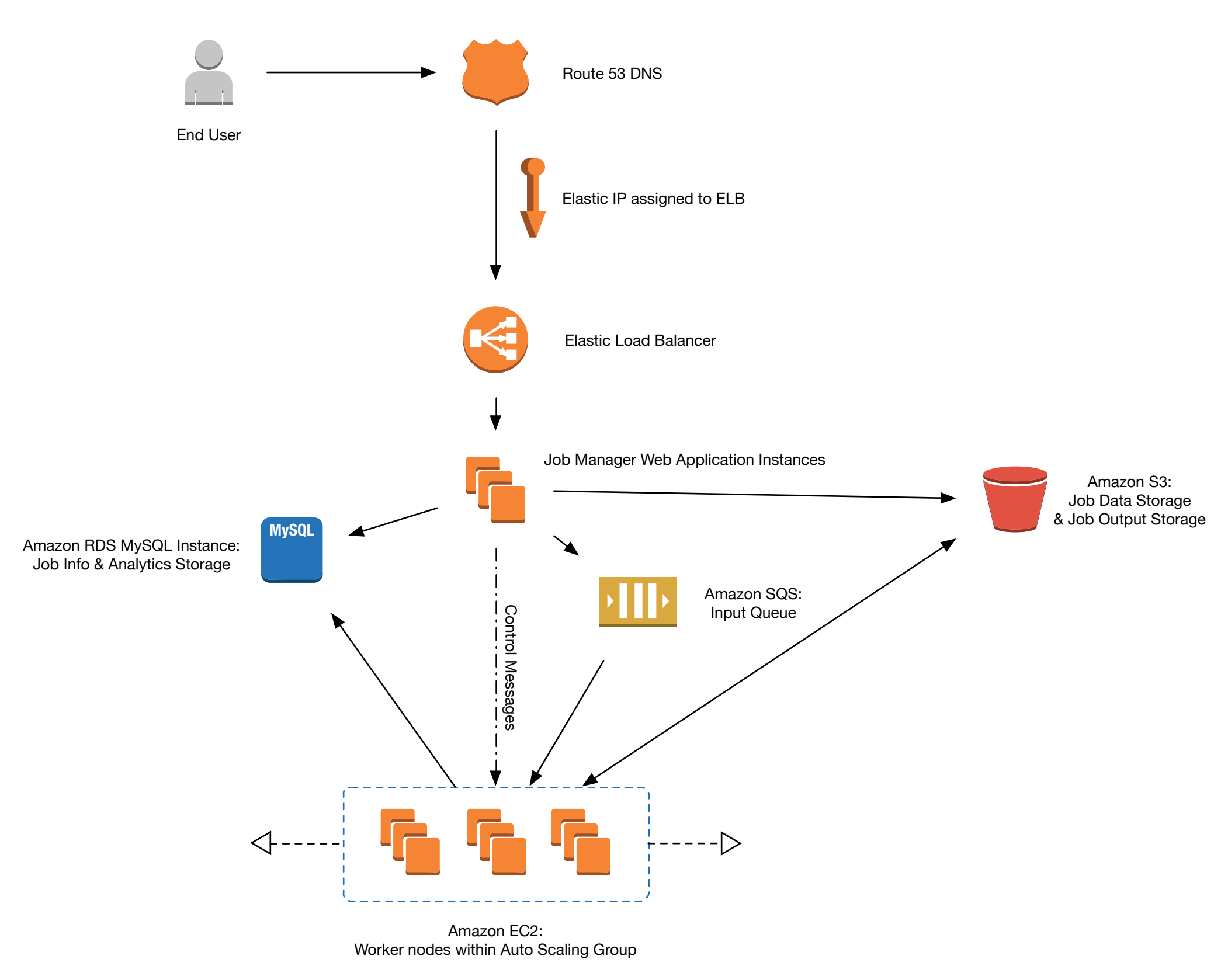
9. References

- AWS Documentation:
  https://aws.amazon.com/documentation/

- AWS SDK Kits:
  https://aws.amazon.com/tools/

- AWS Instance Types:
  http://aws.amazon.com/ec2/instance-types/

10. Appendix


End User
Route 53 DNS
Elastic IP assigned to ELB
Elastic Load Balancer
Job Manager Web Application Instances
Amazon S3:
Job Data Storage
& Job Output Storage
MySQL
Amazon RDS MySQL Instance:
Job Info & Analytics Storage
Amazon SQS:
Input Queue
Control Messages
Amazon EC2:
Worker nodes within Auto Scaling Group

End User

Route 53 DNS

Elastic IP assigned to ELB

AWS

Elastic Load Balancer for Job Manager Web Application

Amazon S3

Amazon SQS

VPC

Bastion Host

Bastion ASG

NAT Instance

NAT1 ASG

Public Subnet
10.0.0.0/23

NAT Instance

NAT2 ASG

Public Subnet
10.0.64.0/23

NAT Instance

NAT3 ASG

Public Subnet
10.0.128.0/23

VPN Gateway to Customer DC

VPC Router

VPC Internet Gateway

Job Manager Web App

Web ASG

Worker Nodes ASG

Private Subnet
10.0.16.0/23

Job Manager Web App

Web ASG

Worker Nodes ASG

Private Subnet
10.0.80.0/23

Job Manager Web App

Web ASG

Worker Nodes ASG

Private Subnet
10.0.144.0/23

Availability Zone us-east-1a

Availability Zone us-east-1b

Availability Zone us-east-1c

Amazon RDS

VPC
10.0.0.0/16

us-east-1 Region

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "Created by: Timothy Patterson.  AWS VPC CloudFormation template.
Creates a multi-tier VPC with two subnets (one public, one private) each in three
availability zones. The public subnet contains a NAT device for Internet access from
the private subnet and a bastion host to allow SSH access to the hosts in the private
subnet.",

  "Parameters" : {

    "KeyName": {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the
bastion host and NAT instances",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "255",
      "AllowedPattern" : "[\\x20-\\x7E]*",
      "ConstraintDescription" : "can contain only ASCII characters."
    },

    "SSHFrom" : {
      "Description" : "Lockdown SSH access to the bastion host (default can be
accessed from anywhere)",
      "Type" : "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default" : "0.0.0.0/0",
      "AllowedPattern" :
"(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription" : "must be a valid CIDR range of the form x.x.x.x/x."
    },

    "BastionInstanceType" : {
      "Description" : "Bastion Host EC2 instance type",
      "Type" : "String",
      "Default" : "m1.small",
      "AllowedValues" : [
"t1.micro","m1.small","m1.medium","m1.large","m1.xlarge","m2.xlarge","m2.2xlarge","m2.
4xlarge","m3.xlarge","m3.2xlarge","c1.medium","c1.xlarge","cc1.4xlarge","cc2.8xlarge",
"cg1.4xlarge"],
      "ConstraintDescription" : "must be a valid EC2 instance type."
    },

    "NATInstanceType" : {
      "Description" : "NET Device EC2 instance type",
      "Type" : "String",
      "Default" : "m1.small",
      "AllowedValues" : [
"t1.micro","m1.small","m1.medium","m1.large","m1.xlarge","m2.xlarge","m2.2xlarge","m2.
```

```
4xlarge","m3.xlarge","m3.2xlarge","c1.medium","c1.xlarge","cc1.4xlarge","cc2.8xlarge",
"cg1.4xlarge"],
      "ConstraintDescription" : "must be a valid EC2 instance type."
    },

    "VPCCIDR" : {
      "Description" : "Subnet CIDR range for the entire VPC.",
      "Type" : "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default" : "0.0.0.0/0",
      "AllowedPattern" :
"(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription" : "must be a valid CIDR range of the form x.x.x.x/x."
    },

    "AZ1Name": {
      "Description" : "AWS name of the availability zone. Ex: us-east-1a",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "20",
      "AllowedPattern" : "[\\x20-\\x7E]*",
      "ConstraintDescription" : "can contain only ASCII characters."
    },

    "AZ1publicCIDR" : {
      "Description" : "Subnet CIDR range for the first AZ's public network",
      "Type" : "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default" : "0.0.0.0/0",
      "AllowedPattern" :
"(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription" : "must be a valid CIDR range of the form x.x.x.x/x."
    },

    "AZ1privateCIDR" : {
      "Description" : "Subnet CIDR range for the first AZ's private network",
      "Type" : "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default" : "0.0.0.0/0",
      "AllowedPattern" :
"(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription" : "must be a valid CIDR range of the form x.x.x.x/x."
    },

    "AZ2Name": {
      "Description" : "AWS name of the availability zone. Ex: us-east-1a",
      "Type": "String",
      "MinLength": "1",
```

```
      "MaxLength": "20",
      "AllowedPattern" : "[\\x20-\\x7E]*",
      "ConstraintDescription" : "can contain only ASCII characters."
    },

    "AZ2publicCIDR" : {
      "Description" : "Subnet CIDR range for the second AZ's public network",
      "Type" : "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default" : "0.0.0.0/0",
      "AllowedPattern" :
"(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription" : "must be a valid CIDR range of the form x.x.x.x/x."
    },

    "AZ2privateCIDR" : {
      "Description" : "Subnet CIDR range for the second AZ's private network",
      "Type" : "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default" : "0.0.0.0/0",
      "AllowedPattern" :
"(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription" : "must be a valid CIDR range of the form x.x.x.x/x."
    },

    "AZ3Name": {
      "Description" : "AWS name of the availability zone. Ex: us-east-1a",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "20",
      "AllowedPattern" : "[\\x20-\\x7E]*",
      "ConstraintDescription" : "can contain only ASCII characters."
    },

    "AZ3publicCIDR" : {
      "Description" : "Subnet CIDR range for the third AZ's public network",
      "Type" : "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default" : "0.0.0.0/0",
      "AllowedPattern" :
"(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription" : "must be a valid CIDR range of the form x.x.x.x/x."
    },

    "AZ3privateCIDR" : {
      "Description" : "Subnet CIDR range for the third AZ's private network",
      "Type" : "String",
      "MinLength": "9",
```

```
      "MaxLength": "18",
      "Default" : "0.0.0.0/0",
      "AllowedPattern" :
"(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription" : "must be a valid CIDR range of the form x.x.x.x/x."
    },

    "VPCName": {
      "Description" : "Name of the VPC. Ex: PreProdVPC",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "20",
      "AllowedPattern" : "[\\x20-\\x7E]*",
      "ConstraintDescription" : "can contain only ASCII characters."
    },

    "VPCEnvironment": {
      "Description" : "Value for the environment tag for objects in the VPC. Ex:
dev/preprod/prod",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "20",
      "AllowedPattern" : "[\\x20-\\x7E]*",
      "ConstraintDescription" : "can contain only ASCII characters."
    }
  },

  "Mappings" : {
    "AWSInstanceType2Arch" : {
      "t1.micro"    : { "Arch" : "64" },
      "m1.small"    : { "Arch" : "64" },
      "m1.medium"   : { "Arch" : "64" },
      "m1.large"    : { "Arch" : "64" },
      "m1.xlarge"   : { "Arch" : "64" },
      "m2.xlarge"   : { "Arch" : "64" },
      "m2.2xlarge"  : { "Arch" : "64" },
      "m2.4xlarge"  : { "Arch" : "64" },
      "m3.xlarge"   : { "Arch" : "64" },
      "m3.2xlarge"  : { "Arch" : "64" },
      "c1.medium"   : { "Arch" : "64" },
      "c1.xlarge"   : { "Arch" : "64" },
      "cc1.4xlarge" : { "Arch" : "64Cluster" },
      "cc2.8xlarge" : { "Arch" : "64Cluster" },
      "cg1.4xlarge" : { "Arch" : "64GPU" }
    },

    "AWSRegionArch2AMI" : {
      "us-east-1"      : { "32" : "ami-a0cd60c9", "64" : "ami-aecd60c7", "64Cluster"
: "ami-a8cd60c1",      "64GPU" : "ami-eccf6285" },
      "us-west-2"      : { "32" : "ami-46da5576", "64" : "ami-48da5578", "64Cluster"
: "NOT_YET_SUPPORTED", "64GPU" : "NOT_YET_SUPPORTED" },
```

```
      "us-west-1"      : { "32" : "ami-7d4c6938", "64" : "ami-734c6936", "64Cluster"
: "NOT_YET_SUPPORTED", "64GPU" : "NOT_YET_SUPPORTED" },
      "eu-west-1"      : { "32" : "ami-61555115", "64" : "ami-6d555119", "64Cluster"
: "ami-67555113",      "64GPU" : "NOT_YET_SUPPORTED" },
      "ap-southeast-1" : { "32" : "ami-220b4a70", "64" : "ami-3c0b4a6e", "64Cluster"
: "NOT_YET_SUPPORTED", "64GPU" : "NOT_YET_SUPPORTED" },
      "ap-southeast-2" : { "32" : "ami-b3990e89", "64" : "ami-bd990e87", "64Cluster"
: "NOT_YET_SUPPORTED", "64GPU" : "NOT_YET_SUPPORTED" },
      "ap-northeast-1" : { "32" : "ami-2a19aa2b", "64" : "ami-2819aa29", "64Cluster"
: "NOT_YET_SUPPORTED", "64GPU" : "NOT_YET_SUPPORTED" },
      "sa-east-1"      : { "32" : "ami-f836e8e5", "64" : "ami-fe36e8e3", "64Cluster"
: "NOT_YET_SUPPORTED", "64GPU" : "NOT_YET_SUPPORTED" }
    }
  },

  "Resources" : {

    "VPC" : {
      "Type" : "AWS::EC2::VPC",
      "Properties" : {
        "CidrBlock" : { "Ref" : "VPCCIDR" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "Name", "Value" : { "Ref" : "VPCName" } },
          { "Key" : "environment", "Value" : { "Ref" : "VPCEnvironment" } },
          { "Key" : "group", "Value" : "sso" },
          { "Key" : "system", "Value" : "gio.services" }
        ]
      }
    },

   "NATHARole" : {
     "Type" : "AWS::IAM::Role",
     "Properties" : {
       "AssumeRolePolicyDocument" : {
        "Version" : "2012-10-17",
        "Statement": [ {
          "Effect": "Allow",
          "Principal": {
            "Service": [ "ec2.amazonaws.com" ]
          },
          "Action": [ "sts:AssumeRole" ]
         } ]
       },
      "Path" : "/",
      "Policies" : [ {
        "PolicyName" : "NATUpdateRouteTables",
        "PolicyDocument" :
        {
          "Version": "2012-10-17",
          "Statement": [ {
```

```
            "Effect": "Allow",
            "Action": [
              "ec2:DescribeInstances",
              "ec2:ModifyInstanceAttribute",
             "ec2:DescribeSubnets",
                "ec2:DescribeRouteTables",
                "ec2:CreateRoute",
                "ec2:ReplaceRoute"
            ],
            "Resource": "*"
            } ]
        }
      } ]
     }
   },

    "NATInstanceProfile" : {
      "Type" : "AWS::IAM::InstanceProfile",
      "Properties" : {
        "Path" : "/",
       "Roles" : [ { "Ref" : "NATHARole" }]
      }
    },

    "AZ1publicSubnet" : {
      "Type" : "AWS::EC2::Subnet",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "CidrBlock" : { "Ref" : "AZ1publicCIDR" },
        "AvailabilityZone" : { "Ref" : "AZ1Name" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "public" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-", { "Ref" : "AZ1Name" }, "-public" ] ] } }
        ]
      }
    },

    "AZ2publicSubnet" : {
      "Type" : "AWS::EC2::Subnet",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "CidrBlock" : { "Ref" : "AZ2publicCIDR" },
        "AvailabilityZone" : { "Ref" : "AZ2Name" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "public" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-", { "Ref" : "AZ2Name" }, "-public" ] ] } }
        ]
```

```
      }
    },

    "AZ3publicSubnet" : {
      "Type" : "AWS::EC2::Subnet",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "CidrBlock" : { "Ref" : "AZ3publicCIDR" },
        "AvailabilityZone" : { "Ref" : "AZ3Name" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "public" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-", { "Ref" : "AZ3Name" }, "-public" ] ] } }
        ]
      }
    },

    "InternetGateway" : {
      "Type" : "AWS::EC2::InternetGateway",
      "Properties" : {
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "public" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-IGW" ] ] } }
        ]
      }
    },

    "GatewayToInternet" : {
       "Type" : "AWS::EC2::VPCGatewayAttachment",
       "Properties" : {
         "VpcId" : { "Ref" : "VPC" },
         "InternetGatewayId" : { "Ref" : "InternetGateway" }
       }
    },

    "publicRouteTable" : {
      "Type" : "AWS::EC2::RouteTable",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "public" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-PublicRouteTable" ] ] } }
        ]
      }
    },
```

```
    "publicRoute" : {
      "Type" : "AWS::EC2::Route",
      "DependsOn" : "GatewayToInternet",
      "Properties" : {
        "RouteTableId" : { "Ref" : "publicRouteTable" },
        "DestinationCidrBlock" : "0.0.0.0/0",
        "GatewayId" : { "Ref" : "InternetGateway" }
      }
    },

    "AZ1publicSubnetRouteTableAssociation" : {
      "Type" : "AWS::EC2::SubnetRouteTableAssociation",
      "Properties" : {
        "SubnetId" : { "Ref" : "AZ1publicSubnet" },
        "RouteTableId" : { "Ref" : "publicRouteTable" }
      }
    },

    "AZ2publicSubnetRouteTableAssociation" : {
      "Type" : "AWS::EC2::SubnetRouteTableAssociation",
      "Properties" : {
        "SubnetId" : { "Ref" : "AZ2publicSubnet" },
        "RouteTableId" : { "Ref" : "publicRouteTable" }
      }
    },

    "AZ3publicSubnetRouteTableAssociation" : {
      "Type" : "AWS::EC2::SubnetRouteTableAssociation",
      "Properties" : {
        "SubnetId" : { "Ref" : "AZ3publicSubnet" },
        "RouteTableId" : { "Ref" : "publicRouteTable" }
      }
    },

    "publicNetworkAcl" : {
      "Type" : "AWS::EC2::NetworkAcl",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "public" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-PublicNetworkACL" ] ] } }
        ]
      }
    },

    "InboundHTTPpublicNetworkAclEntry" : {
      "Type" : "AWS::EC2::NetworkAclEntry",
      "Properties" : {
        "NetworkAclId" : { "Ref" : "publicNetworkAcl" },
```

```
      "RuleNumber" : "100",
      "Protocol" : "6",
      "RuleAction" : "allow",
      "Egress" : "false",
      "CidrBlock" : "0.0.0.0/0",
      "PortRange" : { "From" : "80", "To" : "80" }
    }
  },

  "InboundHTTPSpublicNetworkAclEntry" : {
    "Type" : "AWS::EC2::NetworkAclEntry",
    "Properties" : {
      "NetworkAclId" : { "Ref" : "publicNetworkAcl" },
      "RuleNumber" : "101",
      "Protocol" : "6",
      "RuleAction" : "allow",
      "Egress" : "false",
      "CidrBlock" : "0.0.0.0/0",
      "PortRange" : { "From" : "443", "To" : "443" }
    }
  },

  "InboundSSHpublicNetworkAclEntry" : {
    "Type" : "AWS::EC2::NetworkAclEntry",
    "Properties" : {
      "NetworkAclId" : { "Ref" : "publicNetworkAcl" },
      "RuleNumber" : "102",
      "Protocol" : "6",
      "RuleAction" : "allow",
      "Egress" : "false",
      "CidrBlock" : { "Ref" : "SSHFrom" },
      "PortRange" : { "From" : "22", "To" : "22" }
    }
  },

  "InboundEmphemeralpublicNetworkAclEntry" : {
    "Type" : "AWS::EC2::NetworkAclEntry",
    "Properties" : {
      "NetworkAclId" : { "Ref" : "publicNetworkAcl" },
      "RuleNumber" : "103",
      "Protocol" : "6",
      "RuleAction" : "allow",
      "Egress" : "false",
      "CidrBlock" : "0.0.0.0/0",
      "PortRange" : { "From" : "1024", "To" : "65535" }
    }
  },

  "OutboundpublicNetworkAclEntry" : {
    "Type" : "AWS::EC2::NetworkAclEntry",
    "Properties" : {
```

```
        "NetworkAclId" : { "Ref" : "publicNetworkAcl" },
        "RuleNumber" : "100",
        "Protocol" : "6",
        "RuleAction" : "allow",
        "Egress" : "true",
        "CidrBlock" : "0.0.0.0/0",
        "PortRange" : { "From" : "0", "To" : "65535" }
      }
    },

    "AZ1publicSubnetNetworkAclAssociation" : {
      "Type" : "AWS::EC2::SubnetNetworkAclAssociation",
      "Properties" : {
        "SubnetId" : { "Ref" : "AZ1publicSubnet" },
        "NetworkAclId" : { "Ref" : "publicNetworkAcl" }
      }
    },

    "AZ2publicSubnetNetworkAclAssociation" : {
      "Type" : "AWS::EC2::SubnetNetworkAclAssociation",
      "Properties" : {
        "SubnetId" : { "Ref" : "AZ2publicSubnet" },
        "NetworkAclId" : { "Ref" : "publicNetworkAcl" }
      }
    },

    "AZ3publicSubnetNetworkAclAssociation" : {
      "Type" : "AWS::EC2::SubnetNetworkAclAssociation",
      "Properties" : {
        "SubnetId" : { "Ref" : "AZ3publicSubnet" },
        "NetworkAclId" : { "Ref" : "publicNetworkAcl" }
      }
    },

    "AZ1privateSubnet" : {
      "Type" : "AWS::EC2::Subnet",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "CidrBlock" : { "Ref" : "AZ1privateCIDR" },
        "AvailabilityZone" : { "Ref" : "AZ1Name" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "private" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-", { "Ref" : "AZ1Name" }, "-private" ] ] } }
        ]
      }
    },

    "AZ2privateSubnet" : {
      "Type" : "AWS::EC2::Subnet",
```

```
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "CidrBlock" : { "Ref" : "AZ2privateCIDR" },
        "AvailabilityZone" : { "Ref" : "AZ2Name" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "private" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-", { "Ref" : "AZ2Name" }, "-private" ] ] } }
        ]
      }
    },

    "AZ3privateSubnet" : {
      "Type" : "AWS::EC2::Subnet",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "CidrBlock" : { "Ref" : "AZ3privateCIDR" },
        "AvailabilityZone" : { "Ref" : "AZ3Name" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "private" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-", { "Ref" : "AZ3Name" }, "-private" ] ] } }
        ]
      }
    },

    "AZ1privateRouteTable" : {
      "Type" : "AWS::EC2::RouteTable",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "private" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-", { "Ref" : "AZ1Name" }, "-PrivateRouteTable" ] ] } }
        ]
      }
    },

    "AZ2privateRouteTable" : {
      "Type" : "AWS::EC2::RouteTable",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "private" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-", { "Ref" : "AZ2Name" }, "-PrivateRouteTable" ] ] } }
        ]
```

```
      }
    },

    "AZ3privateRouteTable" : {
      "Type" : "AWS::EC2::RouteTable",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "private" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-", { "Ref" : "AZ3Name" }, "-PrivateRouteTable" ] ] } }
        ]
      }
    },

    "AZ1privateSubnetRouteTableAssociation" : {
      "Type" : "AWS::EC2::SubnetRouteTableAssociation",
      "Properties" : {
        "SubnetId" : { "Ref" : "AZ1privateSubnet" },
        "RouteTableId" : { "Ref" : "AZ1privateRouteTable" }
      }
    },

    "AZ2privateSubnetRouteTableAssociation" : {
      "Type" : "AWS::EC2::SubnetRouteTableAssociation",
      "Properties" : {
        "SubnetId" : { "Ref" : "AZ2privateSubnet" },
        "RouteTableId" : { "Ref" : "AZ2privateRouteTable" }
      }
    },

    "AZ3privateSubnetRouteTableAssociation" : {
      "Type" : "AWS::EC2::SubnetRouteTableAssociation",
      "Properties" : {
        "SubnetId" : { "Ref" : "AZ3privateSubnet" },
        "RouteTableId" : { "Ref" : "AZ3privateRouteTable" }
      }
    },

    "privateNetworkAcl" : {
      "Type" : "AWS::EC2::NetworkAcl",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "Tags" : [
          { "Key" : "Application", "Value" : { "Ref" : "AWS::StackId" } },
          { "Key" : "network", "Value" : "private" },
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-PrivateNetworkACL" ] ] } }
        ]
      }
```

```
},

"InboundprivateNetworkAclEntry" : {
  "Type" : "AWS::EC2::NetworkAclEntry",
  "Properties" : {
    "NetworkAclId" : { "Ref" : "privateNetworkAcl" },
    "RuleNumber" : "100",
    "Protocol" : "6",
    "RuleAction" : "allow",
    "Egress" : "false",
    "Icmp" : { "Code" : "-1", "Type" : "-1" },
    "CidrBlock" : "0.0.0.0/0",
    "PortRange" : { "From" : "0", "To" : "65535" }
  }
},

"OutBoundprivateNetworkAclEntry" : {
  "Type" : "AWS::EC2::NetworkAclEntry",
  "Properties" : {
    "NetworkAclId" : { "Ref" : "privateNetworkAcl" },
    "RuleNumber" : "100",
    "Protocol" : "6",
    "RuleAction" : "allow",
    "Egress" : "true",
    "Icmp" : { "Code" : "-1", "Type" : "-1" },
    "CidrBlock" : "0.0.0.0/0",
    "PortRange" : { "From" : "0", "To" : "65535" }
  }
},

"AZ1privateSubnetNetworkAclAssociation" : {
  "Type" : "AWS::EC2::SubnetNetworkAclAssociation",
  "Properties" : {
    "SubnetId" : { "Ref" : "AZ1privateSubnet" },
    "NetworkAclId" : { "Ref" : "privateNetworkAcl" }
  }
},

"AZ2privateSubnetNetworkAclAssociation" : {
  "Type" : "AWS::EC2::SubnetNetworkAclAssociation",
  "Properties" : {
    "SubnetId" : { "Ref" : "AZ2privateSubnet" },
    "NetworkAclId" : { "Ref" : "privateNetworkAcl" }
  }
},

"AZ3privateSubnetNetworkAclAssociation" : {
  "Type" : "AWS::EC2::SubnetNetworkAclAssociation",
  "Properties" : {
    "SubnetId" : { "Ref" : "AZ3privateSubnet" },
    "NetworkAclId" : { "Ref" : "privateNetworkAcl" }
```

```
      }
    },

   "AZ1NATASG" : {
     "Type" : "AWS::AutoScaling::AutoScalingGroup",
     "Properties" : {
      "AvailabilityZones" : [ { "Ref" : "AZ1Name" } ],
      "VPCZoneIdentifier" : [ { "Ref" : "AZ1publicSubnet" } ],
      "LaunchConfigurationName" : { "Ref" : "NATLaunchConfig" },
      "MinSize" : "1",
      "MaxSize" : "1",
      "DesiredCapacity" : "1",
        "Tags" : [
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", ["NAT-", { "Ref" :
"AZ1Name" }] ]}, "PropagateAtLaunch" : "true" },
          { "Key" : "environment", "Value" : { "Ref" : "VPCEnvironment" },
"PropagateAtLaunch" : "true" },
          { "Key" : "group", "Value" : "sso", "PropagateAtLaunch" : "true" },
          { "Key" : "system", "Value" : "gio.services", "PropagateAtLaunch" : "true" }
        ]
     }
   },

   "AZ2NATASG" : {
     "Type" : "AWS::AutoScaling::AutoScalingGroup",
     "Properties" : {
      "AvailabilityZones" : [ { "Ref" : "AZ2Name" } ],
      "VPCZoneIdentifier" : [ { "Ref" : "AZ2publicSubnet" } ],
      "LaunchConfigurationName" : { "Ref" : "NATLaunchConfig" },
      "MinSize" : "1",
      "MaxSize" : "1",
      "DesiredCapacity" : "1",
        "Tags" : [
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", ["NAT-", { "Ref" :
"AZ2Name" }] ]}, "PropagateAtLaunch" : "true" },
          { "Key" : "environment", "Value" : { "Ref" : "VPCEnvironment" },
"PropagateAtLaunch" : "true" },
          { "Key" : "group", "Value" : "sso", "PropagateAtLaunch" : "true" },
          { "Key" : "system", "Value" : "gio.services", "PropagateAtLaunch" : "true" }
        ]
     }
   },

   "AZ3NATASG" : {
     "Type" : "AWS::AutoScaling::AutoScalingGroup",
     "Properties" : {
      "AvailabilityZones" : [ { "Ref" : "AZ3Name" } ],
      "VPCZoneIdentifier" : [ { "Ref" : "AZ3publicSubnet" } ],
      "LaunchConfigurationName" : { "Ref" : "NATLaunchConfig" },
      "MinSize" : "1",
      "MaxSize" : "1",
```

```
      "DesiredCapacity" : "1",
        "Tags" : [
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", ["NAT-", { "Ref" :
"AZ3Name" }] ]}, "PropagateAtLaunch" : "true" },
          { "Key" : "environment", "Value" : { "Ref" : "VPCEnvironment" },
"PropagateAtLaunch" : "true" },
          { "Key" : "group", "Value" : "sso", "PropagateAtLaunch" : "true" },
          { "Key" : "system", "Value" : "gio.services", "PropagateAtLaunch" : "true" }
        ]
      }
   },

   "NATLaunchConfig" : {
     "Type" : "AWS::AutoScaling::LaunchConfiguration",
     "DependsOn" : "GatewayToInternet",
     "Properties" : {
      "ImageId"  : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" : "AWS::Region"
}, { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "NATInstanceType" },
"Arch" ] } ] },
      "SecurityGroups" : [{ "Ref" : "NATSecurityGroup" }],
        "InstanceType" : { "Ref" : "NATInstanceType" },
        "KeyName"  : { "Ref" : "KeyName" },
      "AssociatePublicIpAddress" : "true",
      "IamInstanceProfile" : { "Ref" : "NATInstanceProfile" },
      "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
        "#!/bin/bash\n\n",
        "cd /usr/local/bin; /usr/bin/wget
https://raw.githubusercontent.com/ralex-aws/vpc/master/ha-nat.sh\n",
        "chmod 755 /usr/local/bin/ha-nat.sh\n",
        "logger -t \"vpc\" -- \"Starting HA NAT setup.\"\n",
        "/usr/local/bin/ha-nat.sh\n"
      ]]}}
     }
   },

    "NATSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable internal access to the NAT device",
        "VpcId" : { "Ref" : "VPC" },
        "SecurityGroupIngress" : [
           { "IpProtocol" : "tcp", "FromPort" : "80",  "ToPort" : "80",  "CidrIp" :
"0.0.0.0/0" } ,
           { "IpProtocol" : "tcp", "FromPort" : "443", "ToPort" : "443", "CidrIp" :
"0.0.0.0/0" } ],
        "SecurityGroupEgress" : [
           { "IpProtocol" : "tcp", "FromPort" : "80",  "ToPort" : "80",  "CidrIp" :
"0.0.0.0/0" } ,
           { "IpProtocol" : "tcp", "FromPort" : "443", "ToPort" : "443", "CidrIp" :
"0.0.0.0/0" } ],
        "Tags" : [
```

```
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [{ "Ref" : "VPCName" },
"-NATSecurityGroup"] ]} }
        ]
      }
    },

    "BastionSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable access to the Bastion host",
        "VpcId" : { "Ref" : "VPC" },
        "SecurityGroupIngress" : [ { "IpProtocol" : "tcp", "FromPort" : "22",
"ToPort" : "22",  "CidrIp" : { "Ref" : "SSHFrom" }} ],
        "SecurityGroupEgress"  : [ { "IpProtocol" : "tcp", "FromPort" : "22",
"ToPort" : "22",  "CidrIp" : { "Ref" : "VPCCIDR" }}],
        "Tags" : [
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [{ "Ref" : "VPCName" },
"-BastionSecurityGroup"] ]} }
        ]
      }
    },

   "BastionASG" : {
     "Type" : "AWS::AutoScaling::AutoScalingGroup",
     "Properties" : {
      "AvailabilityZones" : [ { "Ref" : "AZ1Name" }, { "Ref" : "AZ2Name" }, { "Ref" :
"AZ3Name" } ],
      "VPCZoneIdentifier" : [ { "Ref" : "AZ1publicSubnet" }, { "Ref" :
"AZ2publicSubnet" }, { "Ref" : "AZ3publicSubnet" } ],
      "LaunchConfigurationName" : { "Ref" : "BastionLaunchConfig" },
      "MinSize" : "1",
      "MaxSize" : "1",
      "DesiredCapacity" : "1",
        "Tags" : [
          { "Key" : "Name", "Value" : { "Fn::Join" : ["", [ { "Ref" : "VPCName" },
"-Bastion" ] ]}, "PropagateAtLaunch" : "true" },
          { "Key" : "environment", "Value" : { "Ref" : "VPCEnvironment" },
"PropagateAtLaunch" : "true" },
          { "Key" : "group", "Value" : "sso", "PropagateAtLaunch" : "true" },
          { "Key" : "system", "Value" : "gio.services", "PropagateAtLaunch" : "true" }
        ]
     }
   },

   "BastionLaunchConfig" : {
     "Type" : "AWS::AutoScaling::LaunchConfiguration",
     "DependsOn" : "GatewayToInternet",
     "Properties" : {
      "ImageId"  : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" : "AWS::Region"
}, { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "BastionInstanceType" },
"Arch" ] } ] },
```

```
      "SecurityGroups" : [{ "Ref" : "BastionSecurityGroup" }],
        "InstanceType" : { "Ref" : "BastionInstanceType" },
        "KeyName"  : { "Ref" : "KeyName" },
      "AssociatePublicIpAddress" : "true"
     }
   }
  },

  "Outputs" : {
    "VPCId" : {
      "Description" : "VPCId of the newly created VPC",
      "Value" :  { "Ref" : "VPC" }
    },
    "AZ1publicSubnet" : {
      "Description" : "SubnetId of the public subnet",
      "Value" :  { "Ref" : "AZ1publicSubnet" }
    },
    "AZ2publicSubnet" : {
      "Description" : "SubnetId of the public subnet",
      "Value" :  { "Ref" : "AZ2publicSubnet" }
    },
    "AZ3publicSubnet" : {
      "Description" : "SubnetId of the public subnet",
      "Value" :  { "Ref" : "AZ3publicSubnet" }
    },
    "AZ1privateSubnet" : {
      "Description" : "SubnetId of the private subnet",
      "Value" :  { "Ref" : "AZ1privateSubnet" }
    },
    "AZ2privateSubnet" : {
      "Description" : "SubnetId of the private subnet",
      "Value" :  { "Ref" : "AZ2privateSubnet" }
    },
    "AZ3privateSubnet" : {
      "Description" : "SubnetId of the private subnet",
      "Value" :  { "Ref" : "AZ3privateSubnet" }
    },
    "BastionSecurityGroup" : {
      "Description" : "Security group of the Bastion Host",
      "Value" :  { "Ref" : "BastionSecurityGroup" }
    }
  }
}
```