ASSIGNMENT FOR THE TECHNICAL INTERVIEW
CLOUD ARCHITECT

Mahmoud Ali

C2 General

Table of Contents

C2 General

1 Introduction

The scope of this document is to explain the developed solution that fulfill the customer requirement and help him to re-factor and move the Webserver Ghost Blog into the Cloud.

The solution is hosted inside Microsoft Azure Cloud and is leveraging on Cloud native services such as Web App Service for the frontend and on Storage Account (File share), MySQL database for the backend, and on Frontdoor and Web Application Firewall to load balance the incoming traffic and protect the web application against any exploits.

The App Service is running a customized ghost image which is built and pushed to Azure Container Registry (ACR).

The infrastructure will be deployed across two regions in active/standby mode. So, it provides a Geo-Redundancy in case of a region failure. If a regional outage affects the primary region, so the Front Door will fail over and route the incoming traffic to the secondary region without affecting the user requests.

2 Solution Design

The Ghost blog will be hosted inside Azure using Web App service with Linux OS based and deployed on two instances for high availability. It loads the customized image placed inside Azure Container Registry (ACR). The detailed design is illustrated in Figure 1.

The App Service is considering two deployment slots (Production, Testing). The “Testing” environment is a full replica of the production environment where DevOps teams can customize the application and release new versions before going into production without any downtime.

It also adapts the traffic spike and cover up to 4 times the typical traffic load by autoscaling the app instances based on CPU utilization or number of incoming requests.

Since the application is running inside a container, the data will not be saved when the container is restarted. To overcome that, a Storage account (File share) and MySQL database are used to keep the data persistent.

For Azure Storage, read-access geo-redundant storage (RA-GRS) is used. The data is replicated to a secondary region. If there is a regional outage or disaster, the Azure team performs a geo-failover to the secondary region. There is no customer action required for this failover.

For MySQL database, the Geo-Replication is used to create a readable secondary replica in a different region and fail over to a secondary database if the primary one fails or needs to be taken offline.

The Frontdoor will be used to load balance and route the incoming requests to the primary region and fail over to the secondary region in case of any regional outage. Moreover, Azure Web Application Firewall (WAF) is added on top of Azure Front Door that provides centralized protection for the web applications. It defends the web service against common exploits and vulnerabilities and keeps the service highly available for the users.

C2 General

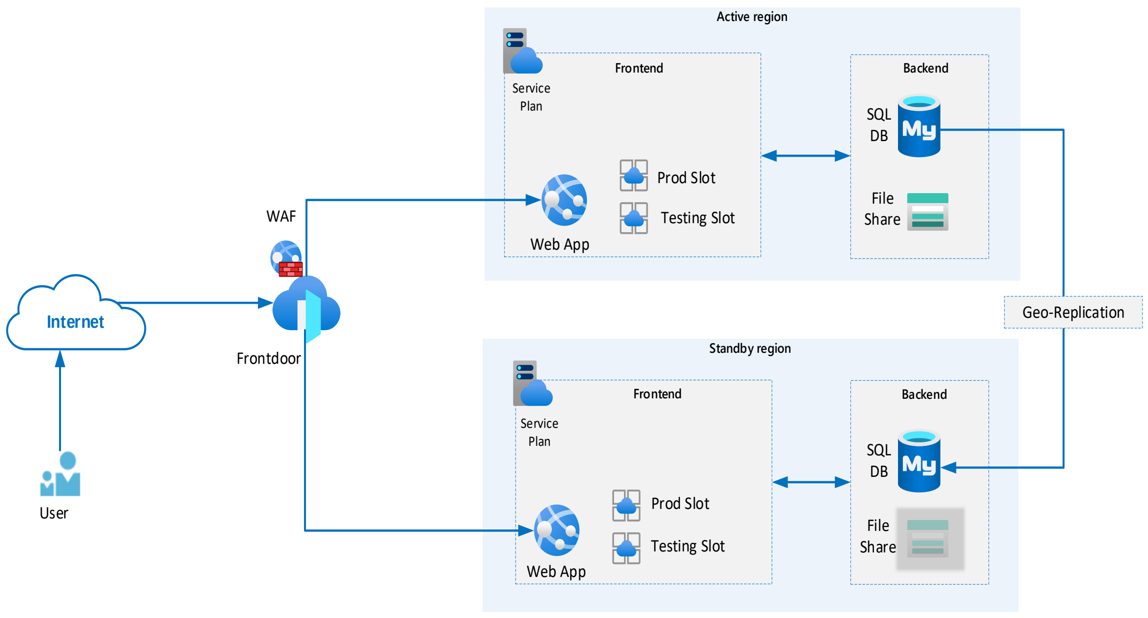


Figure 1: Design Topology to deploy Ghost blog on Azure

Moreover, the direct access to the Application is blocked and allowed only for the traffic originates from Azure Frontdoor. In this way, the incoming traffic is inspected by the WAF before arriving the application.

Additionally, the solution is also providing a full observability by accessing the container via SSH from Specific External IP Address, monitoring the App by enabling Application Insight, and exporting the diagnostic logs into Azure monitor.

At the end, if the customer wants to delete all the posts at once, he can trigger and run a script that access the Ghost MySQL database and remove the data.

Note: For this BC, I used the domains released by Azure (<app-name>.azurewebsites.net and <afd-hostname>.afdazure.net) in order to design and validate the solution. But the ideal scenario is to add the customer´s domain and upload their own certificate (either directly to the app or using Azure KeyVault).

3 Infra Configuration

- **Frontdoor** configured with the following option settings:
  - Session affinity Enabled
  - Health probe
  - Forwarding rule to HTTPS only
  - Redirecting rule from HTTP to HTTPS
  - WAF policy applied
  - App in Region 1 has higher priority than App in Region 2

- **WAF**
  - Deployed in Prevention mode.
  - Created the WAF policy with default Managed rules for the moment. It can be customized

- **App Service Plan**
  - S1 Sku that supports deployment slots and autoscaling features.
  - Type: Linux
  - Autoscaling rule based on CPU Utilization and incoming requests
    - Default instance 2

C2 General

        - Min 2
        - Max 8
    - Notify the administrator or any additional emails in case of any scale out/in.

- **Web App**
    - Deploy a customized ghost image from the ACR.
    - Two Deployment Slots for environments:
        - Production
        - Testing
    - Set HTTPS only and TLS 1.2 Enabled
    - Set FTP state to FTPS Only
    - Block direct access to the application, allow access only for the traffic originates from the Frontdoor
    - Block direct access to the application via webssh, allow access only from specific subnet
    - Monitory and Logging
        - Open SSH to Linux container
        - Enable App Health check
        - Enable Application Insight
        - Export diagnostic logs to a Log Analytics workspace
        - Enable docker container logging
        - Enable Application Logging
    - Mount an External Storage (Fileshare) to keep persistent data where ghost is saving the data (/var/lib/ghost/content)
    - Set the application settings for Azure MySQL database connection:
        - database__client
        - database__connection__host
        - database__connection__user
        - database__connection__password
        - database__connection__database

- **Storage Account**
    - Deployed in read-access geo-redundant storage (RA-GRS) replication mode
    - Create a File share

- **Azure Database for MySQL**
    - Create MySQL Single server with **General Purpose tier**
    - Create the firewall rules to permit the incoming connections.
    - Create the database
    - Enable Geo-replication and Add a replica database

4 Deployment Plan

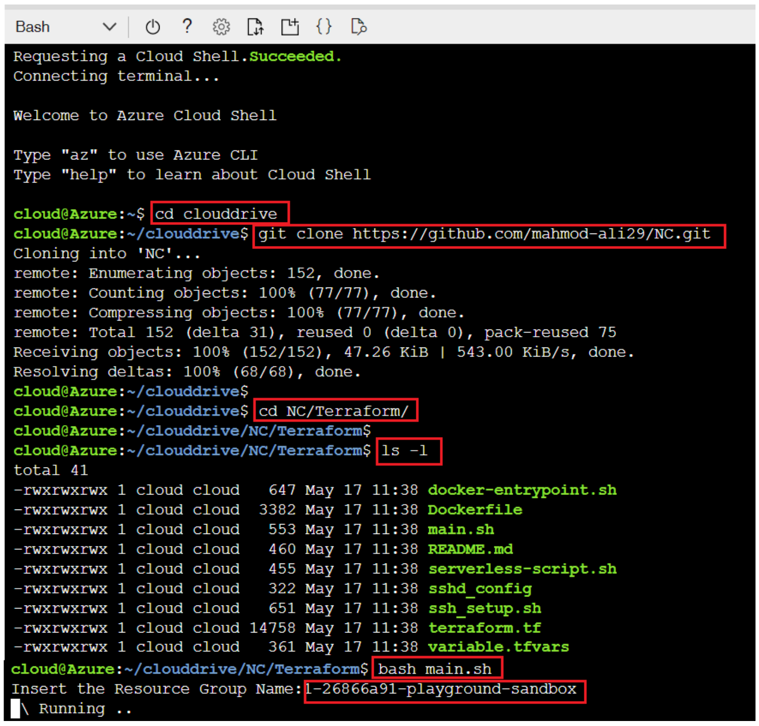
The infrastructure will be deployed by using Infrastructure as Code (IaC): Azure CLI/ARM Template or Terraform.

There are two ways to run the deploy the infrastructure, assuming the RG is already created. One of the following ways can be used.

1. To run the script of AzureCLI, open the Cloudshell in Bash mode
    - Go into clouddrive
        - **cd clouddrive**
    - Clone the repository:
        - **git clone https://github.com/mahmod-ali29/NC.git**
    - Go into repository
        - **cd NC/AzureCLI**
    - Run the script
        - **bash main.sh**
    - To run the serverless script
        - **bash serverless-script.sh**

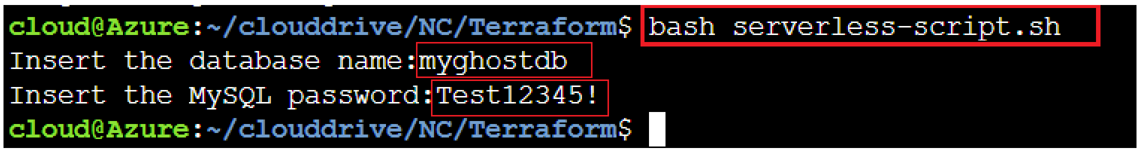
C2 General

2. To run the terraform script, Open the cloudshell in Bash mode
   - Go into clouddrive
     - **cd clouddrive**
   - clone the repository
     - **git clone https://github.com/mahmod-ali29/NC.git**
   - Go into git repository
     - **cd NC/Terraform**
   - Run the script
     - **bash main.sh**



```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

cloud@Azure:~$ cd clouddrive
cloud@Azure:~/clouddrive$ git clone https://github.com/mahmod-ali29/NC.git
Cloning into 'NC'...
remote: Enumerating objects: 152, done.
remote: Counting objects: 100% (77/77), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 152 (delta 31), reused 0 (delta 0), pack-reused 75
Receiving objects: 100% (152/152), 47.26 KiB | 543.00 KiB/s, done.
Resolving deltas: 100% (68/68), done.
cloud@Azure:~/clouddrive$
cloud@Azure:~/clouddrive$ cd NC/Terraform/
cloud@Azure:~/clouddrive/NC/Terraform$
cloud@Azure:~/clouddrive/NC/Terraform$ ls -l
total 41
-rwxrwxrwx 1 cloud cloud   647 May 17 11:38 docker-entrypoint.sh
-rwxrwxrwx 1 cloud cloud  3382 May 17 11:38 Dockerfile
-rwxrwxrwx 1 cloud cloud   553 May 17 11:38 main.sh
-rwxrwxrwx 1 cloud cloud   460 May 17 11:38 README.md
-rwxrwxrwx 1 cloud cloud   455 May 17 11:38 serverless-script.sh
-rwxrwxrwx 1 cloud cloud   322 May 17 11:38 sshd_config
-rwxrwxrwx 1 cloud cloud   651 May 17 11:38 ssh_setup.sh
-rwxrwxrwx 1 cloud cloud 14758 May 17 11:38 terraform.tf
-rwxrwxrwx 1 cloud cloud   361 May 17 11:38 variable.tfvars
cloud@Azure:~/clouddrive/NC/Terraform$ bash main.sh
Insert the Resource Group Name:1-26866a91-playground-sandbox
 \ Running ..
```

- To run the serverless script
  - **bash serverless-script.sh**



```
cloud@Azure:~/clouddrive/NC/Terraform$ bash serverless-script.sh
Insert the database name:myghostdb
Insert the MySQL password:Test12345!
cloud@Azure:~/clouddrive/NC/Terraform$
```

It may take up to 15-20 minutes to be ready.

These scripts already provide some parameters and others are to be filled:

Name	Description	Default
storageAccountName	Insert nameof the Storageaccount (should be globally unique)	Already filled
region1	Insert the active region (e.g.eastus, centralus)	Already filled
region2	Insert the standby region (e.g.eastus, centralus)	Already filled
External IP	Insert Public IP of your machine	Already filled
RG	Insert Resource Group name	to be inserted

Note:

C2 General

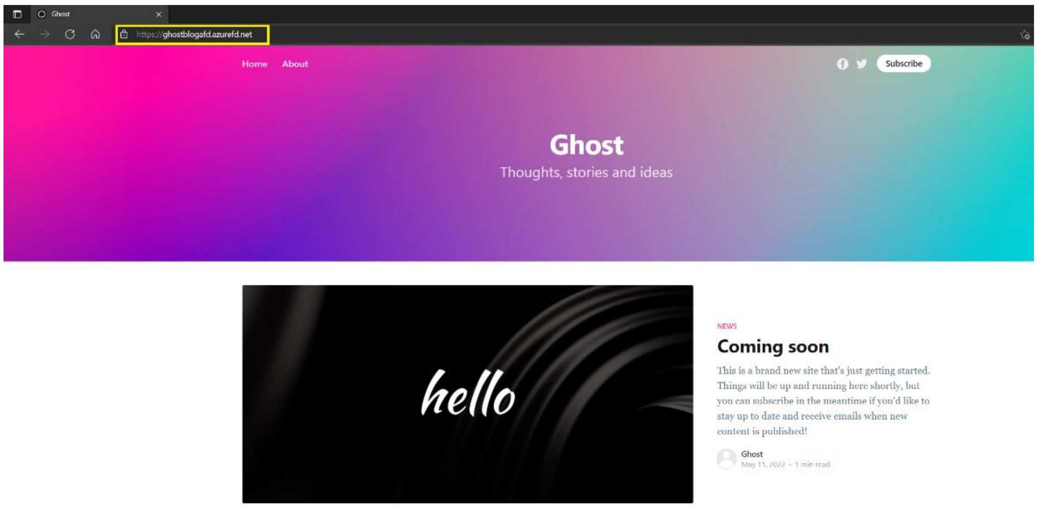
Due to some limitations that I have on my sandbox, I'm not able to deploy a MySQL as Single Server, I deployed it as a flexible server in order to validate and deploy the solution.

The script is deploying the components in region1 and region2. In specific, it is deploying the following resources:

- Azure container Registry (ACR)
- Frontdoor
- Web Application Firewall (WAF) Policy
- 2 x App Service Plan
- 2 x App Service
- 2 x App Service (Slot)
- Storage Account and File share
- Azure Database for MySQL flexible server
- Log Analytics workspace
- Application Insights

5 Output

Access the blog using the Frontdoor link

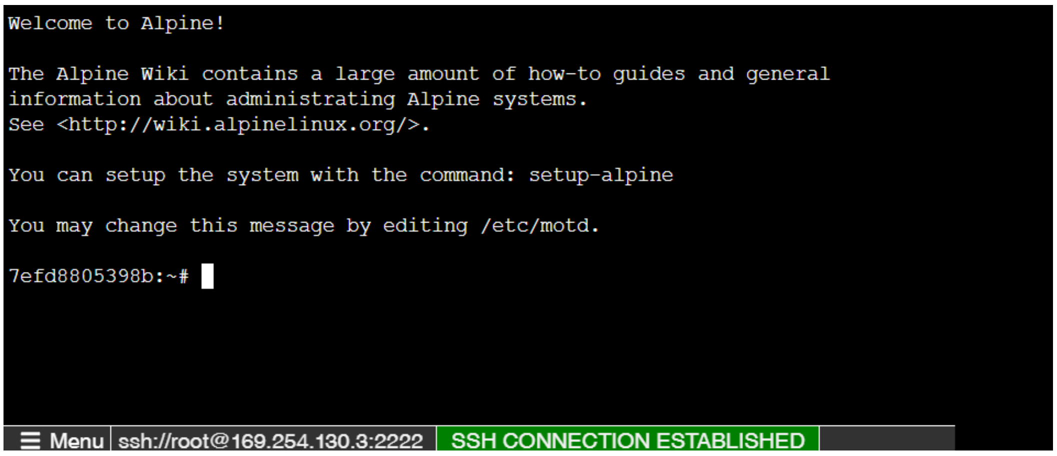


Coming soon

This is a brand new site that's just getting started. Things will be up and running here shortly, but you can subscribe in the meantime if you'd like to stay up to date and receive emails when new content is published!

Ghost
May 11, 2022 • 1 min read

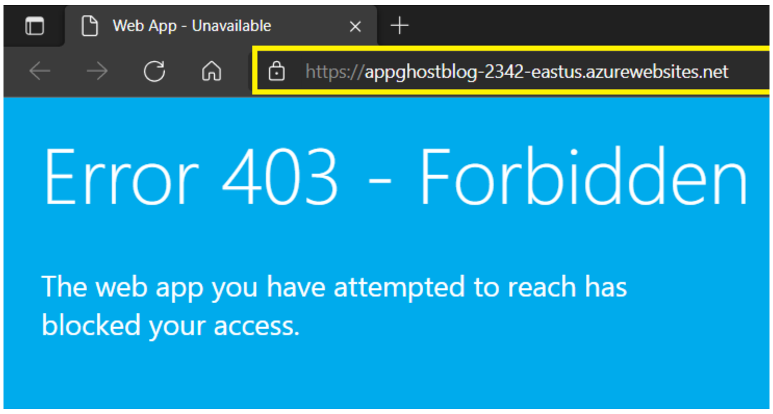
Access the Linux container via web SSH. Connection established



```
Welcome to Alpine!

The Alpine Wiki contains a large amount of how-to guides and general
information about administrating Alpine systems.
See <http://wiki.alpinelinux.org/>.

You can setup the system with the command: setup-alpine

You may change this message by editing /etc/motd.

7efd8805398b:~#
```

Access App1 directly using Webapp link Fails as intended

C2 General



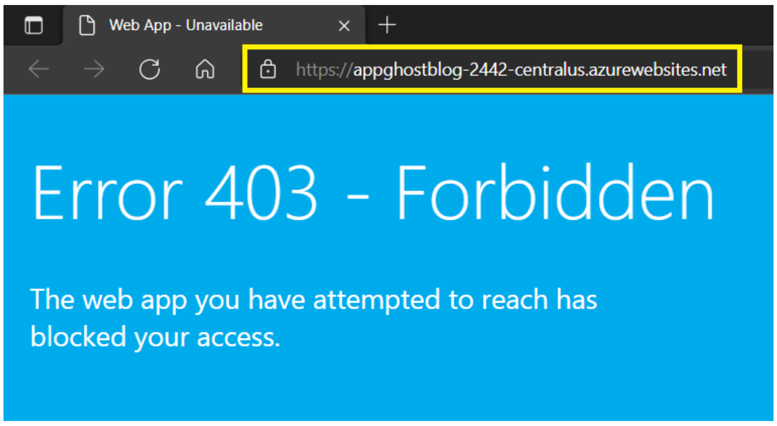
Web App - Unavailable

https://appghostblog-2342-eastus.azurewebsites.net

Error 403 - Forbidden

The web app you have attempted to reach has blocked your access.

Access App2 directly using Webapp link Fails as intended



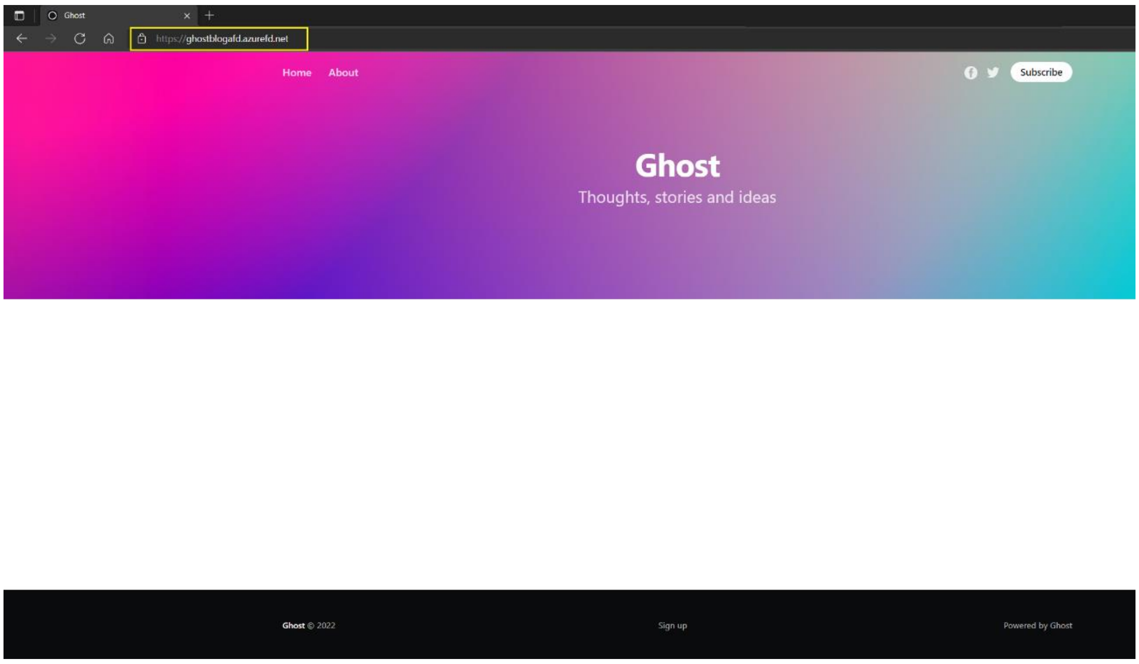
Web App - Unavailable

https://appghostblog-2442-centralus.azurewebsites.net

Error 403 - Forbidden

The web app you have attempted to reach has blocked your access.

Run the serverless script to remove all posts. All posts have been removed



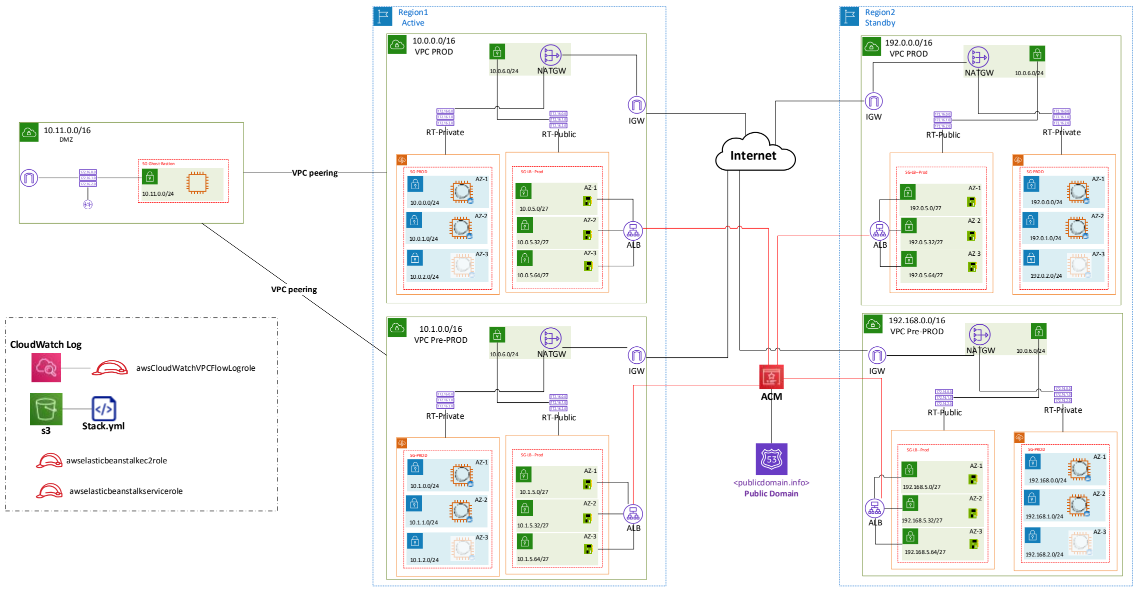
6 Ghost Blog on AWS

Alternatively, the above solution can also be deployed on AWS cloud but using different cloud components.

Elastic beanstalk will be used to host the Ghost blog.

The infrastructure looks like:

- VPC for each environment in each region
- Subnet/Route table/Internet Gateway/NAT Gateway for each environment in each region
- Elastic Beanstalk Application in each region
- Elastic Beanstalk Environments
    - PROD
    - PRE-PROD
- Customized Security Group
- Customized ACL
- Application Load balancer (ALB) with sticky Session
- Full stack observability
- AutoScaling Group
- AWS Certificate Manager
- Route 53 to provide Geo-redundancy
- RDS Database with Geo-replica to keep data persistent
- A dedicate VPC which is DMZ to access the EC2 instances of Ghost App.



C2 General

7 Reference Documents

Drawing.pdf

C2 General