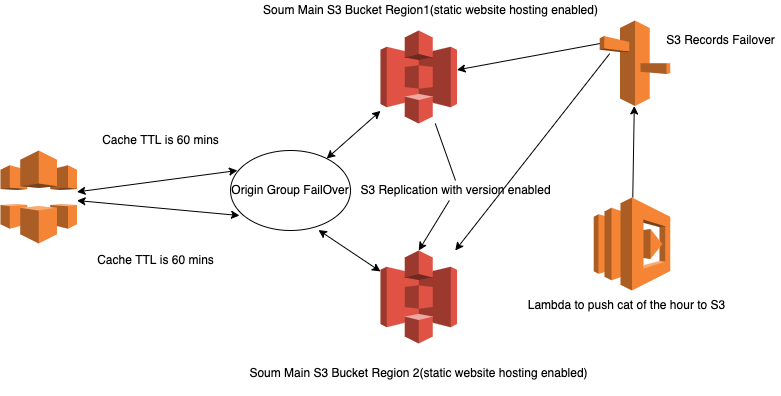
Soum Assessment Documentation

**Cat of the Hour**

In order to make cat of the hour APP highly available, secured and cost optimized, I have proposed the below design:



* CloudFront:Cloudfront provides highly available cached objects. It servs objects among multiple regions with low latency. I choosed to deploy cloudfront on only North America, Europe, Asia, Middle East, and Africa instead all edge locations. Also I have configured Cloudfront to redirect HTTP to HTTPS requests to establish a secure connection.
* S3 Bucketss: to store the static website. Configuration for S3 buckets are the below
  + Public Access Block Configuration : block all kind of public access configuration by enabling block public ACL, policy, Ignore public ACL and Restrict public buckets configuration
  + Enable S3 Replication between S3 buckets with version enabled
  + Enable lifecycle configuration to move objects that infrequently accessed to a cheaper storage class, I choose Intelligent-Tiering storage class for automatic transition to cheap storage classes
* Route 53: for Domain name translation and failover purposes
* Lambda Function: to generate cat of the hour image and push it to S3

**Putting services together:**

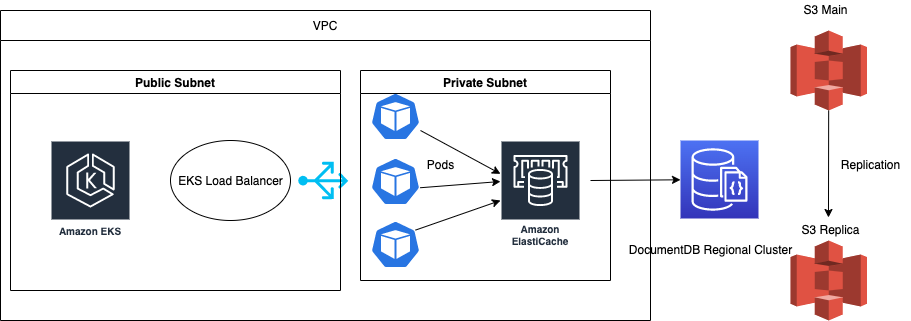
I put CloudFront in the front of customer requests instead of S3 to secure S3 buckets following security best practices AND to cache objects on CloudFront to reduce requests coming to S3. S3 is fast and can serve a large number of requests, but with Cloudfront, we get less latency and more availability since Cloudfront is a regional service.

For S3 Buckets, I suggest to create two S3 buckets in a different regions with lifecycle policy enabled and block public access enabled (as mentioned above), then configured Cloudfront to have two S3 origins configured inside Origin Group Failover to make sure that Cloudfront can always get the new objects from the available S3 bucket

Lambda Function: this lambda will generate the image every hour and send it to S3 bucket using Route53 health check, this will ensure that S3 bucket always available to be used by Lambda

**Image of the day**

This design is for calling image of the day from Kubernetes cluster



* Vpc: I proposed to use VPC to keep services in both private and public networks
* EKS: to manage public requests from customers who request the image
* Elasticache: for caching requests to the database we put Elasticach
* DocuementDB Regional cluster: datastore for the application, regional means in case of the current region becomes unavailable, we can use Database on another region
* S3 Buckets: Same as design of Cat of the Hour( but without cloudfront
  + Note: my suggestion here to add cloudfront also and store the link of the cloudfront image path instead of S3

**Putting All together:**

To ensure that infrastructure is secured I suggest moving EKS and Elasticache inside VPC. EKS has the ability to use private and public subnets. Therefore, we can use public subnet for load balancer to serve public requests and load balancer communicates with pods and other resources inside private subnets. With this setup, worker nodes(pods) and other resources will be secured from public requests, which is the best practice for building a good infrastructure

Pods will talk first to elasticach to get the image of the day path on S3(using DNS record concept that we used in the Cat of the hour design), if there are no cache keys, it will talk to mongodb.

To achieve high availability, we will deploy EKS and elasticache into another region. S3 already deployed into two regions and DocumentDB is regional, therefore, in case of disaster, we can rely on them without the need to deploy into another region.