# DevOps Task - AWS Task

All code repositories used to store the terraform configuration and lambda functions should be accessible for inspection. If not explicitly stated, apps or websites used by different tasks are at candidates discretion. Websites can be a simple index.html with minimum HTML code just to show that it works, and service being deployed by CodeDeploy can be a simple nginx setup with index.html that can be changed by pushing to the code repository.

1. Create an IAM read-only user that has:
    a. read-only S3 access to buckets for tasks below
    b. read-only access to CloudFront distributions for tasks below
    c. read-only access to lambda functions for tasks below
    d. read-only access for EC2 (VPC, subnets…)
2. Create a Cloudfront distribution with a single S3 bucket origin (can be any static page). Create a Lambda@Edge function that will, based on a list of request URIs, load content from different origin (can be a S3 bucket or custom origin). For example:
    a. example.com would load content from first bucket
    b. example.com/login would load content from the second
3. Create a Lambda@Edge function that will send a response to a client containing OWASP HTTP security headers.
4. Create a script (python, bash, ruby, whatever your preference is) that will upload and publish new versions of ViewerRequest, ViewerResponse, OriginRequest and OriginResponse functions to Lambda@Edge. Each function code should be in separate directory named by function attachment

(ViewerRequest,ViewerResponse…). The same script should update a Cloudfront distribution with new versions of the Lambda@Edge functions (Cloudfront ID passed as a parameter to the script).

5. Create, using Terraform, two new VPCs in different regions of your choice:
    a. 3 public
    b. 3 private
    c. 3 DB private subnets
    d. managed NAT gateway
    e. private DNS zone
6. Peer the two new VPCs using Terraform
7. Write a lambda function that will react to Instance reachability events and notify using Email or Slack API.
8. Create a EC2 launch template attached to ASG. The EC2 instance should be setup with Nginx serving a static content that can be updated using AWS CodeDeploy from a git repository (either github or AWS codecommit). The instances in ASG should be autoscalable and deployable using AWS CodeDeploy.