Current Infrastructure and Reasoning

Due to time shortage and cost control, the current infrastructure has a single subnet in a single availability zone with a single EC2 instance running. The project is done for speed rather than best practices or optimisations. It currently uses just IaC to deploy and host an AMI to display the header message with a reasonable security group and nginx configuration. It utilises input variables to provide the access key, secret key and the dynamic string to be deployed to the variable. The dynamic string is deployed to the EC2 using user_data attribute on the EC2 instance.

Available Options

To improve this, we can do many things, however, as the business requirements are not clear, I would just mention some of the things that can be done to make this a more optimal solution whilst keeping the costs in mind.

Improvements with Infrastructure

As mentioned above, the current infrastructure does not consider the high availability or reliability of the resources. To do this, we can use multiple availability zones with multiple subnets. We also need a Load Balancer in front of the EC2 instances.

As this is a dynamic website, we need a CI/CD pipeline to improve our continuous integration and continuous deployment process. This can be done using any tools but something small and simple could be to utilise GitHub actions. We can build, test, and deploy our changes every time before we make any changes. This wasn't done in the current solution due to time constraints.

Another conscious decision to make is where to store the website and the strings. Something simple like Amazon S3 would do a great job here. We can store the website along with the dynamic strings in the S3 bucket, host our website from there and using version control we can also keep track of the changes.

Another important change we can make is to use a configuration management tool to provide us with more control over the script that is run on the EC2 instance. Currently, we are using user_data and this isn't the optimal way as we have no clue whether the script has successfully run or not. With a CM tool, we can also set the variable with more ease.

Depending on the need, we can also use Amazon CloudFront for low-latency content delivery to all our users.

Improvements with Terraform

There are many improvements that can be done with the terraform code. Some notable changes are, to use environment variables for the Access Key and Secret Key rather than providing them every time the script is run. Split the resources and use terraform modules to write more reusable code. We can utilise terraform functions for things like CIDR subnets.