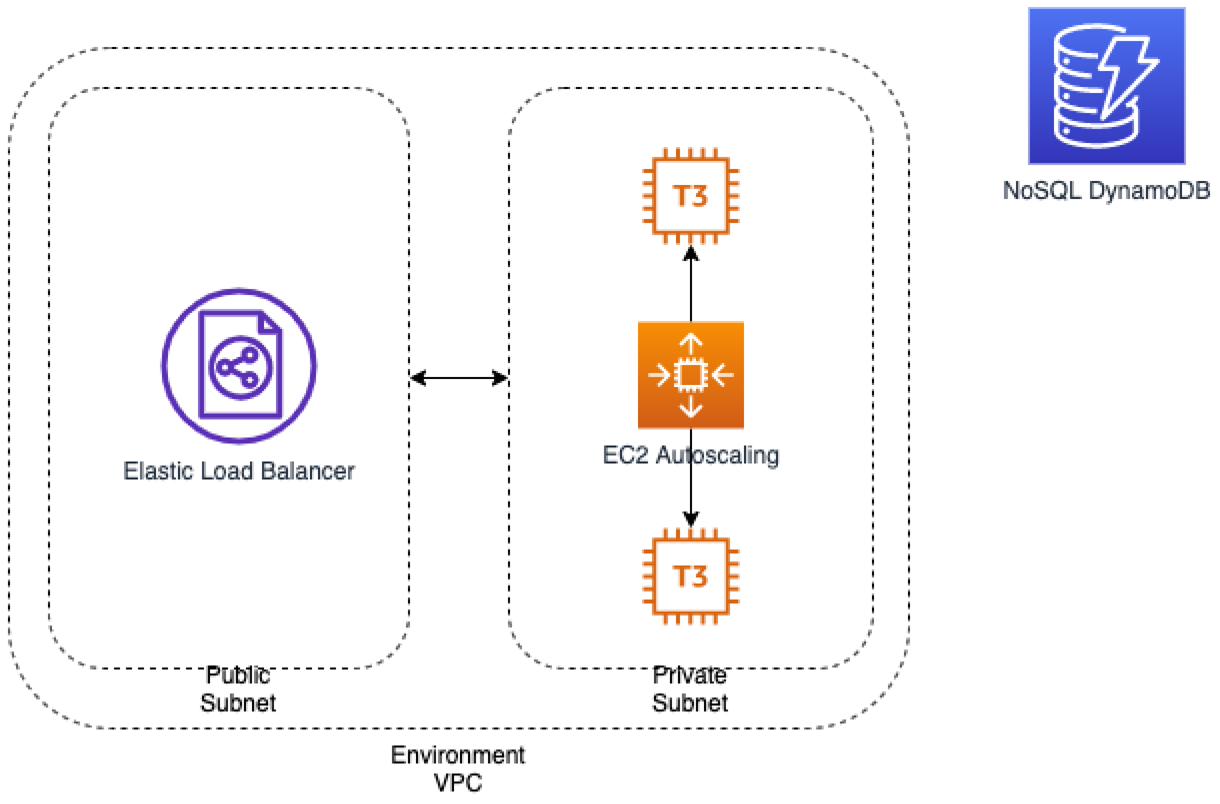
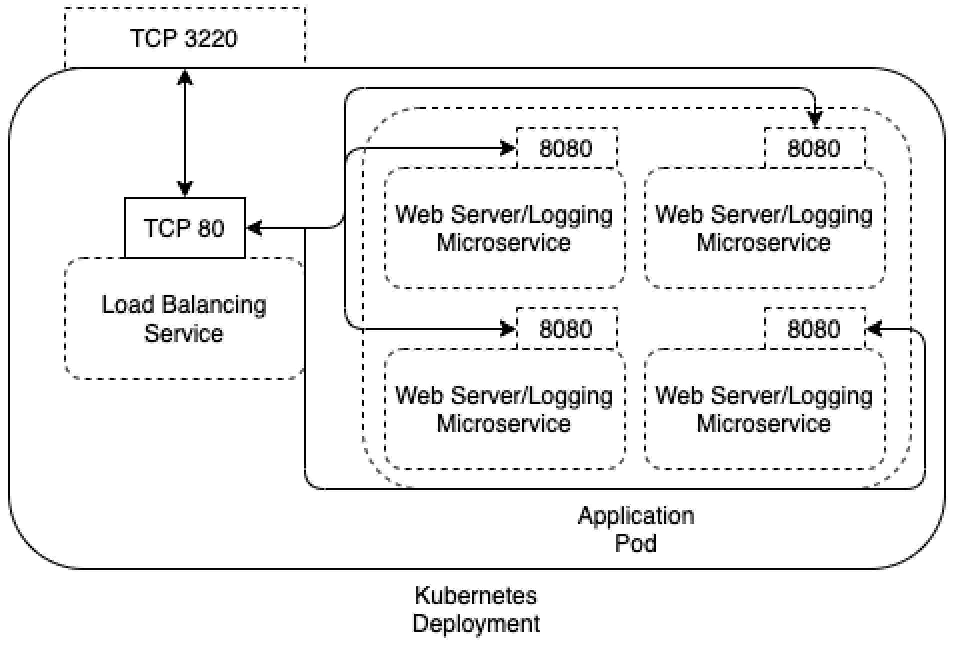
Pollinate Technical Assessment

The virtual architecture for this deployment can be described as below:



And the Kubernetes architecture can be described using the below:



TCP 3220
TCP 80
Load Balancing Service
8080
Web Server/Logging Microservice
8080
Web Server/Logging Microservice
8080
Web Server/Logging Microservice
8080
Web Server/Logging Microservice
Application Pod
Kubernetes Deployment

Questions and Tasks:

1. Explain with as much detail as you can of your high-level design and explain why that would meet the requirements outlined above.

The implemented k8s deployment has multiple pods that are scalable within each node on the cluster as well as the autoscaling group being configurable to scale to any required size.

2. Provide details of your web API that will be used to submit a curl command (curl -X POST http:// /app) that will insert the Date/Time stamp into your database. The API code should be well documented.

Submitting a POST request using curl or other web tools to the ELB public DNS address over HTTP will result in an entry being logged to the provisioned DynamoDB table. This is run as a microservice that covers all paths of the request URL.

3. In addition, provide any automation for your backup/restore process you would implement.

The image used is stored in Docker Hub and is publicly accessible. The cluster uses an AWS provided AMI and failed instances are automatically replaced by the EC2 Autoscaling Group.

In addition to the above, implementing point-in-time restores against the DynamoDB table will allow rollbacks should the table become corrupted. This can be done through the resource in Terraform.

4. Provide details of your Persistency Layer, with details of your cluster setup and configuration.

This is handled entirely by DynamoDB as a service, as the microservices themselves are stateless.

5. If you consider any Load Balancer or queuing service to be used, please be ready to explain the reasons and your suggested configurations for each.

Load balancing was used in order to facilitate the provisioning of multiple containers to process requests, as well as additional pods and hosts deployed to the EKS cluster. There are no concerns with concurrency in this specific case as there is no reading of data, only writing.

6. Please consider monitoring and maintainability.

In lieu of a tool such as Prometheus that has the ability to monitor k8s resources directly, the use of EKS as well as CloudWatch event triggers or alarms will allow us to monitor the solution.

The solution is saved and configured in Terraform state, except for the final step of running the Kubernetes deployment. The solution can thus be destroyed and rebuilt through the use of `terraform destroy` from within the workspace.

7. Please explain how elastic your service is, what would be the trigger points and how the scale up or down would work.

The service is, in theory, horizontally scalable up to 40 instances within the cluster (the maximum for an EC2 Autoscaling Group). These can be scaled vertically in addition and more instances of different can be added in Terraform as a way of ensuring minimal downtime during these alterations to the scalability.

After initial benchmarking, it would be suggested that this scaling be handled via either predictive scaling (during known busy periods) or through scaling based on requests issued to the load balancer, with a heuristic target of one host per X requests (varying of course by the size of the instance).