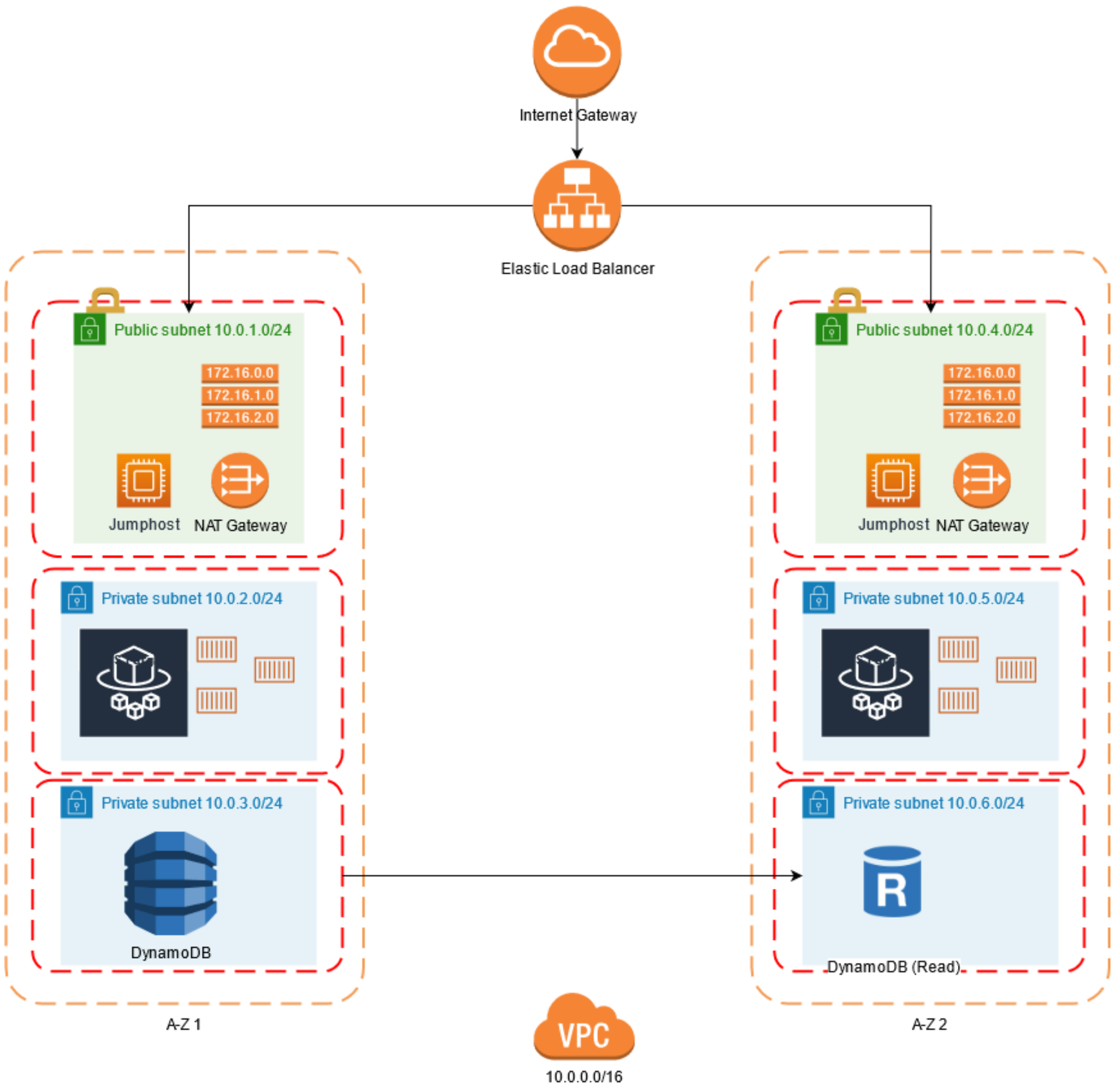

My proposed system design is shown at the above diagram

The setup comprises of 2 AZs to ensure high availability as such if 1 AZ is down the system will still be available.

Each AZ comprises of 1 public subnet and 2 private subnets (App, Database layer). The public subnet will contain 1 NAT Gateway and 1 Jumphost.

App layer contains the backend server that will be containerised and will be configured using Fargate to handle the auto-scaling of the containers. Using Fargate is recommended as it eliminates the effort of manually configuring each EC2 and it can orchestrate the containers based on the server load.

Database layer contains the DynamoDB with read replica set up to synchronized data to the other DynamoDB set up in the other AZ.

The NAT Gateway is set up to allow services located in the private subnet to allow access the internet, in this case to allow the cron job to query the spaceout.gov.sg site to poll for data, as well as to connect to other services to update software version etc.

The Jumphost (optional) is setup to allow incoming SSH connection for accessing the AWS services (For authorised users only).

Each subnet network traffic is secured using security group (red dotted lines) configured in each layer to only allow authorised traffic to flow into individual subnets.

An elastic load balancer (ELB) is configured to load balance the traffic to evenly spread traffic across 2 AZs, as well as to divert traffic in case 1 AZ went down. There are 2 options to use NLB and ALB and if cost is a concern NLB is recommended in this case.

When a request reaches the internet gateway, it will be directed to the ELB and from there, direct the traffic into one of the AZs public subnet. The request will subsequently be directed into the App layer subnet where one of the containers locating in the app layer will process the request and queries the data from the database in the DB layer subnet. The results will be returned to the App layer, and back to the client.