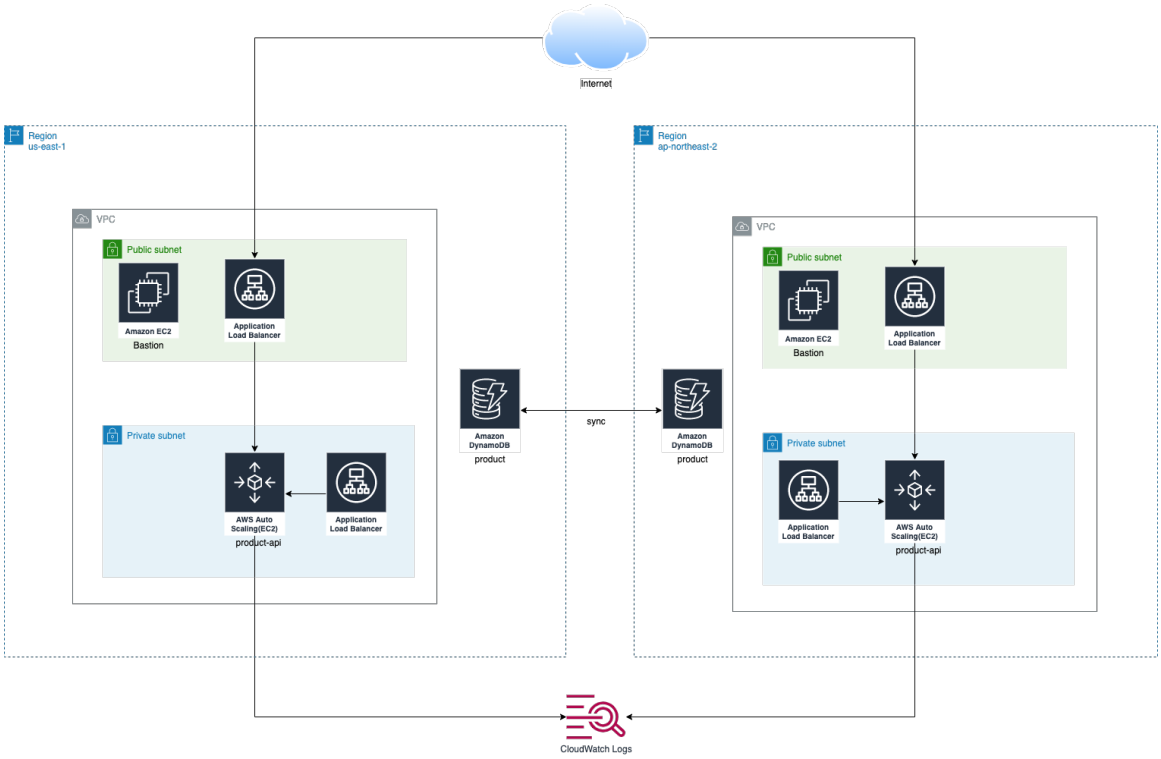
Architecture

This is an overall architecture of the system. Please refer the following diagram. You need to consider High Availability, Scalability and Cost when you design and setup the cloud infrastructure.



Application details

This application is implemented by Golang. It's running on Linux amd64 environment. When a user sends REST API, It saves the data to the DynamoDB table. The name of DynamoDB table is "product". It's hard-corded. You can't change it. The application generates a GUID automatically when it processes POST /v1/product API and saves to DynamoDB.

Binary Information

- Listen port : 8080
- Parameters
  You need to input a region information as a parameter when you start the application. (sample command : ./main us-east-1)
- Logging
  This application doesn't save logs to any files. It outputs logs to STDOUT and STDERR. You need to consider how to handle logs.

API Spec

- GET /health : Health check API.
    - Request sample
      curl {{ server address }}/health
    - Response
      HTTP code 200

- POST /v1/product : Add a product data to the DynamoDB table
    - Request sample
      curl -XPOST -H "Content-Type: application/json" -d "{"name":"apple", "category": "fruit"}" {{ server address }}/v1/product
    - Response
      You can see the GUID in response text if the record is stored in DynamoDB.

- GET /v1/product : The data exist or not in the DynamoDB table.
    - Request sample
      curl -XGET -H "Content-Type: application/json" -d "{"guid": "abcd1010"}" {{ server address }}/v1/product
    - Response
      You can check whether the data exists in the DynamoDB table or not. You can see “The data exists” or “Not exist data” as a response text.

- DELETE /v1/product : Try to delete the data stored in the DynamoDB table.
    - Request sample
      curl -XDELETE -H "Content-Type: application/json" -d "{"guid": "abcd1010"}" {{ server address }}/v1/product
    - Response
      You can see “{”msg”: “completed”}” as a response text when server works fine. This API tries to delete the data. Even if the data doesn’t exist in the DynamoDB table, It tries to delete it and return the complete message as a response. You might need to check in the AWS DynamoDB console.

- GET /region : Return the region information of the server.
    - Request sample
      curl {{ server address }}/region
    - Response
      “{“region”: “us-east-1”}” or “{“region”: “ap-northeast-2”}”

VPC

Basically, This system is multi-region system. Create the VPC with the below information. For traffic to go to the Internet, Public subnet is associated with Internet G/W and Private subnet is associated with NAT G/W. In case of communication of subnets located in different regions, It should be able to communicate using the AWS internal network.

- us-east-1
  - VPC name(tag) : USDEV
  - VPC CIDR : 10.10.0.0/16
  - subnet information
    - Name(tag) : usdev-pub-a (CIDR : 10.10.1.0/24)
    - Name(tag) : usdev-pub-b (CIDR : 10.10.2.0/24)
    - Name(tag) : usdev-priv-a (CIDR : 10.10.11.0/24)
    - Name(tag) : usdev-priv-b (CIDR : 10.10.12.0/24)
- ap-northeast-2
  - VPC name(tag) : APDEV
  - VPC CIDR : 10.20.0.0/16
  - subnet information
    - Name(tag) : apdev-pub-a (CIDR : 10.20.1.0/24)
    - Name(tag) : apdev-pub-b (CIDR : 10.20.2.0/24)
    - Name(tag) : apdev-priv-a (CIDR : 10.20.11.0/24)
    - Name(tag) : apdev-priv-b (CIDR : 10.20.12.0/24)

Bastion host

Setup a bastion EC2 to access AWS resources from your laptop. It's located in public subnet. It has public IP address and the IP should not be changed even if it's restarted. You can access product-api EC2 after accessing the Bastion EC2.

- SSH port : 37722
- AMI : Amazon Linux2
- Type : t2.small or t3.small
- Packages : curl, awscli and jq
- EC2 Tag : Name=bastion

Product API host

Setup a product-api EC2 using Auto Scaling Group. If CPU usage is getting increased or decreased, EC2 should be created or terminated automatically. If new EC2 is launched, the application should be deployed and attached to the load balancers automatically.

- AMI : Amazon Linux2
- Type : c5.large
- Load Balancers : product-ext and product-int
- EC2 Tag : Name=product-api
- Scale-out policy : Over CPU utilization 50%

Load Balancers

Setup Application Load Balancers. You need to create 2 load balancers for each region, one for external use and one for internal use. All load balancers should block the APIs not provided by the product-api server. For example, the load balancer forwards /health, /v1/product and /region API, but It blocks /manage. Blocked APIs by load balancer return 403 response code.

External LB

User who is from the Internet can access to the external LB.

- Listen port : 80
- Name : product-ext

Internal LB

Load balancer for VPC internal networking. User who is from the Internet cannot access to the internal LB.

- Listen port : 80
- Name : product-int

DynamoDB

Each region has the DynamoDB table named product. They MUST be synchronized with each other. For example, when a data is added to the DynamoDB table in AP region, the data should be available in read-time in the US region. When EC2 accesses DynamoDB table, It should use AWS internal network instead of the Internet.

Data column

The Golang application for this test project has the below code. Please design the DynamoDB column. GUID is unique value. Name and category are not unique value. You don't need to configure any index options.

```
type Product struct {
    guid string `dynamodbav:"guid"`
    name string `dynamodbav:"name"`
    category string `dynamodbav:"category"`
}
```

Table information

- us-east-1
    - name : product
    - read capacity : 10(provisioned)
    - write capacity : 10(provisioned)
- ap-northeast-2
    - name : product
    - read capacity : 10(provisioned)
    - write capacity : 10(provisioned)

Logs

Logs output from the product-api server should be uploaded to CloudWatch Logs. It should be sent in real time(within 3 mins) and be queryable in CloudWatch Log Insight. C.W logs group name is "/aws/ec2/product-api"