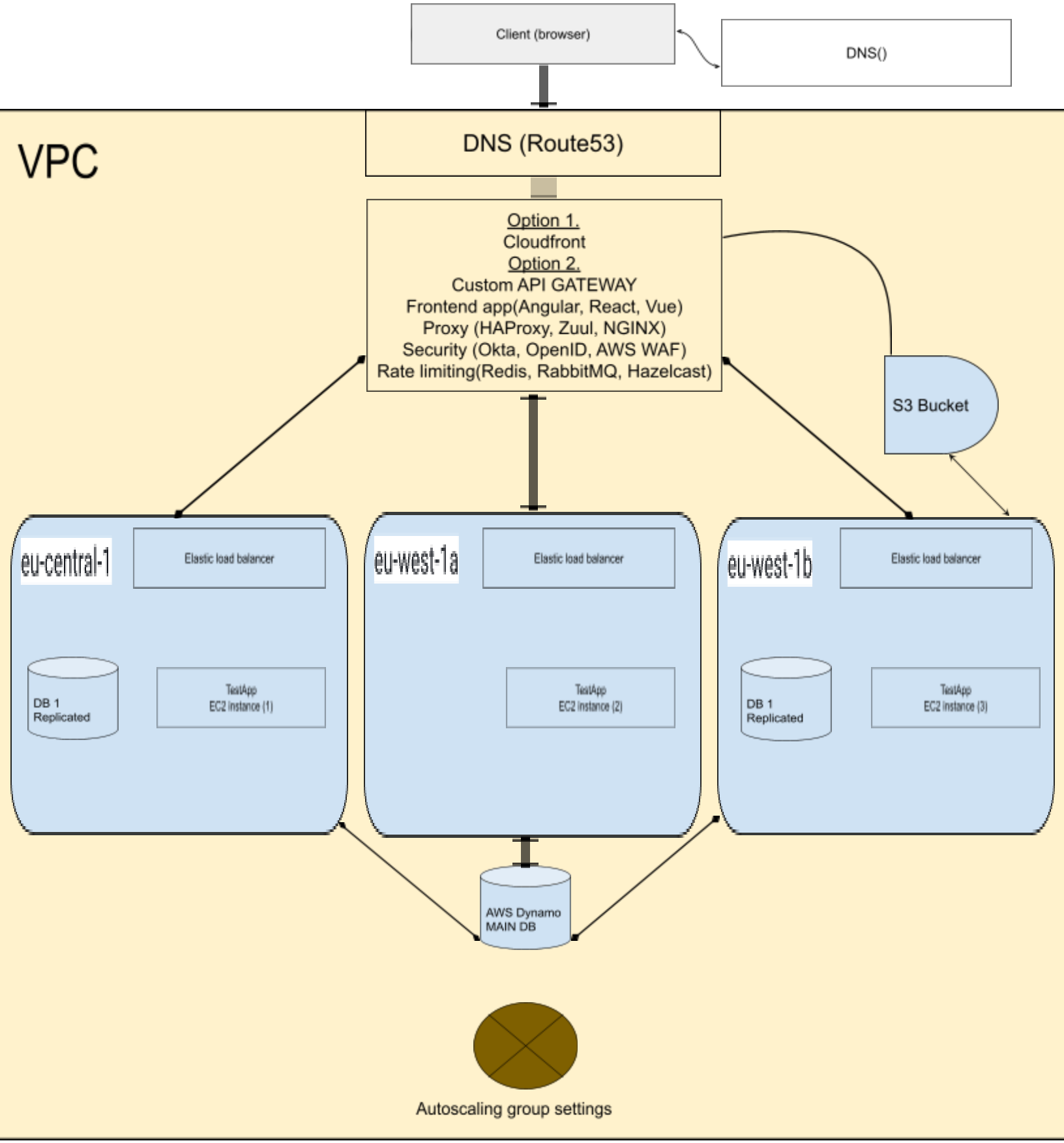
Client (browser)
DNS()
VPC
DNS (Route53)
Option 1.
Cloudfront
Option 2.
Custom API GATEWAY
Frontend app(Angular, React, Vue)
Proxy (HAProxy, Zuul, NGINX)
Security (Okta, OpenID, AWS WAF)
Rate limiting(Redis, RabbitMQ, Hazelcast)
S3 Bucket
eu-central-1
Elastic load balancer
DB 1
Replicated
TestApp
EC2 instance (1)
eu-west-1a
Elastic load balancer
TestApp
EC2 instance (2)
eu-west-1b
Elastic load balancer
DB 1
Replicated
TestApp
EC2 instance (3)
AWS Dynamo
MAIN DB
Autoscaling group settings

1.)

a.) High availability can be achieved with services such as;
Elastic load balancer(which sends metrics to Cloudwatch)
Route53(used for dns failover, service health checks, latency...)
Elastic IP addresses(resource() just switches while client has same ip)

b.) If we plan to develop an application with self healing ability, then we must do it by design(e.g using Circuit breaker pattern). On the other hand, if we plan to deploy, on AWS we can enable 'Auto healing' through OpsWorks layer creation. Also, we can enable it on our load balancers where we can ensure health checks which can be done by creating Auto-scaling Groups. It has the ability to recreate EC2 instances.

c.) For this purpose we use three load balancers, if one goes down another can take point of failure, we can enable it via autoscaling group settings.

d.) By design if we use AWS, there we have 'Shared Responsibility Model for EC2 Storage', so we are secured from HW point of view, but on the other hand we are responsible for managing/securing our instances by implementing backup, encrypt and other. Usage of Cognito or Okta...

e.) Cloudwatch can track latency changes and here we can utilize load balancer to handle sudden jumps in traffic.

f.) Performance can be achieved with Cloudfront CDN by fetching content from EC2, S3, load balancers or other resources. Adding DynamoDB and caching.
On the app side to improve performance, for frontend we can implement Server side rendering and for backend we utilize microservices or serverless architecture approach.

2.)

Would investigate, eventually fix and verify in following steps;
Examine error messages which give us information and direction...
DNS are properly set, check routing policies...
Rules on Firewall, if everything is properly set, to allow communication
WEB server settings(IIS>Windows/Apache or NGINX>Linux)
Database configuration for credentials, if docker image {is is in good state}, ip/port check etc.
Network settings on all entities, check IPtables….

3.)

List of troubleshooting steps;
DNS logs check is possible
Check if I can connect via raw IP by ICMP
Change the hostname of your Amazon Linux instance
IS in a different region and thus unable to connect
Check network settings on local instance
Verify ssh access if applicable

4.)

a/

Disk space (virtual or physical or snapshots or...) ?
Memory (shared by host...) ?
VM in healthy state ?
*cpu cores(upgrade plan) ?

b/

Would check which component is causing unresponsiveness;
resources, would enable cloudwatch, VM itself(logs/events, status checks, configurations, etc.{in host machine}), VPC or VPG configuration, if the network is properly configured, load balancer not sharing load properly, is security group rules properly configured, ACL's, if vm is in some other region, zombie processes...

c/

Depending on observed issue/s, I would follow that direction(google for resolution, take caution and try to fix by scientific method).
Would backup first or take a snapshot, create a new volume from that snapshot, run FileSystem check and attach that volume to another instance for testing.
If it's a Windows VM, use EC2Rescue to troubleshoot OS on that instance.
If it's a Linux VM, I would use 'top', 'lotop', 'vmstat' or other utilities to verify processes, disk I/O and their resource usage or network issues.

```
do {
   if (resources) {
      "would scale or configure autoscaling(if disk, I would redeploy)"
   }
   elseif (misconfiguration) {
      "would try to reconfigure using pattern from other VM"
   }
   else {
      "consult with colleagues(by describing state/issues) to find solution/fix"
   }
}
until (problem has been fixed or at least cause/issue has been recognized)
```