AWS Architecture Design – Assignment #2

BRIEF

Imagine that you meet with a small startup company in the early stages of their operations. Currently their architecture uses a LAMP stack with MySQL, Apache and PHP all running on one desktop PC within their small office. Like many small start-ups they are confident that they will be the next big thing and expect significant, rapid, yet un-quantified growth in the next few months. With this in mind, they are concerned about:

- scaling to meet the demand, but with uncertainty around when and how much this demand will be they are very concerned about buying too much infrastructure too soon or not enough too late!
- their lack of provision for Disaster Recovery
- their ability to configure their database and data access layer for high performance and throughput
- making the user experience in the browser very low latency even though a large portion of their user base will be from far away
- effective distribution of load
- a self-healing infrastructure that recovers from failed service instances
- security of data at rest and in transit
- securing access to the environment as the delivery team expands
- an archival strategy for inactive objects greater than 6 months
- ability to easily manage and replicate multiple environments based on their blueprint architecture.

OBJECTIVE

Recommend a manageable, secure, scalable, high performance, efficient, elastic, highly available, fault tolerant and recoverable architecture that allows the startup to organically grow. The architecture should specifically address the requirements/concerns as described above.

DELIVERABLES

A PDF document no greater than three or four pages in length that clearly and succinctly present an analysis of the startups requirements and the proposed architecture diagram. Clearly state all assumptions made during the design and explicitly state the referenced Amazon Web Services. Once complete, send the document to your recruiting coordinator and recruiter.

AWS Architecture Design – Assignment #2

-by Sivakumar P, Software Trainee at Tringapps.

Given:

Application type : LAMP stack.

Solution:

1. Scaling to meet the demand : (Service : AWS Autoscaling)

AWS Auto Scaling monitors our applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.

2. Disaster Recovery : (Service : Hot Standby)

- At Disaster Recovery in AWS, Hot Standby method can be followed because here one system runs simultaneously with another primary system.
- So when failure happens on the primary node, the hot standby immediately takes over replacing the primary system.

3. Database and data access layer for high performance and throughput : (Service : AWS Aurora for database and AWS ElastiCache for Redis)

- AWS Aurora is MySQL and PostgreSQL compatible cloud service and it serverless which scales upto 64Tb
- 5x performance improvement over MySQL and self healing with peer to peer replication which exactly suits the constraint

- Since LAMP uses MySQL as its database and it is not a key value pair based(NoSQL) database, Redis ElastiCache is used in the data access layer.

4. UX with very low latency though user will be far away :(Service : AWS Cloudfront CDN)

- CloudFront delivers your content through a worldwide network of data centers called edge locations.
- When a user requests content that we are serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.

5. Effective distribution of load : (Service : AWS Application Load Balancer (ALB))

- Since it is an LAMP stack, Application Load Balancer makes routing decisions at the application layer (HTTP/HTTPS), supports path-based routing, and can route requests to one or more ports on each container instance in our cluster.
- The dynamic mapping allows you to have multiple tasks from a single service on the same container instance.

6. Self-healing infrastructure : (Service : AWS Autohealing)

An instance can be a member of multiple layers. If any of those layers has auto healing disabled, AWS OpsWorks Stacks does not heal the instance if it fails.

7. Security of data at rest and in transit : (Service : AWS KMS, HSM & ACM)

- Key Management Service used to encrypt data on the customer’s behalf

- Hardware security module usd for security at Hardware Level.
- Also in this proposed architecture we have used AWS Security groups. We can also use Network ACL, and VPN for security reasons.

8. Securing access to the environment as the delivery team expands :
(Service : Identity Access Module(IAM))

- With IAM, we can specify who can access which services and resources, and under which conditions.
- With IAM policies, we can manage permissions to our workforce and systems to ensure least-privilege permissions.

9. An archival strategy for inactive objects greater than 6 months : (Service : AWS Glacier)

- The Amazon S3 Glacier storage classes are purpose-built for data archiving, providing us with the highest performance, most retrieval flexibility, and the lowest cost archive storage in the cloud.
- This can be used for the archival objects greater than 6 months.

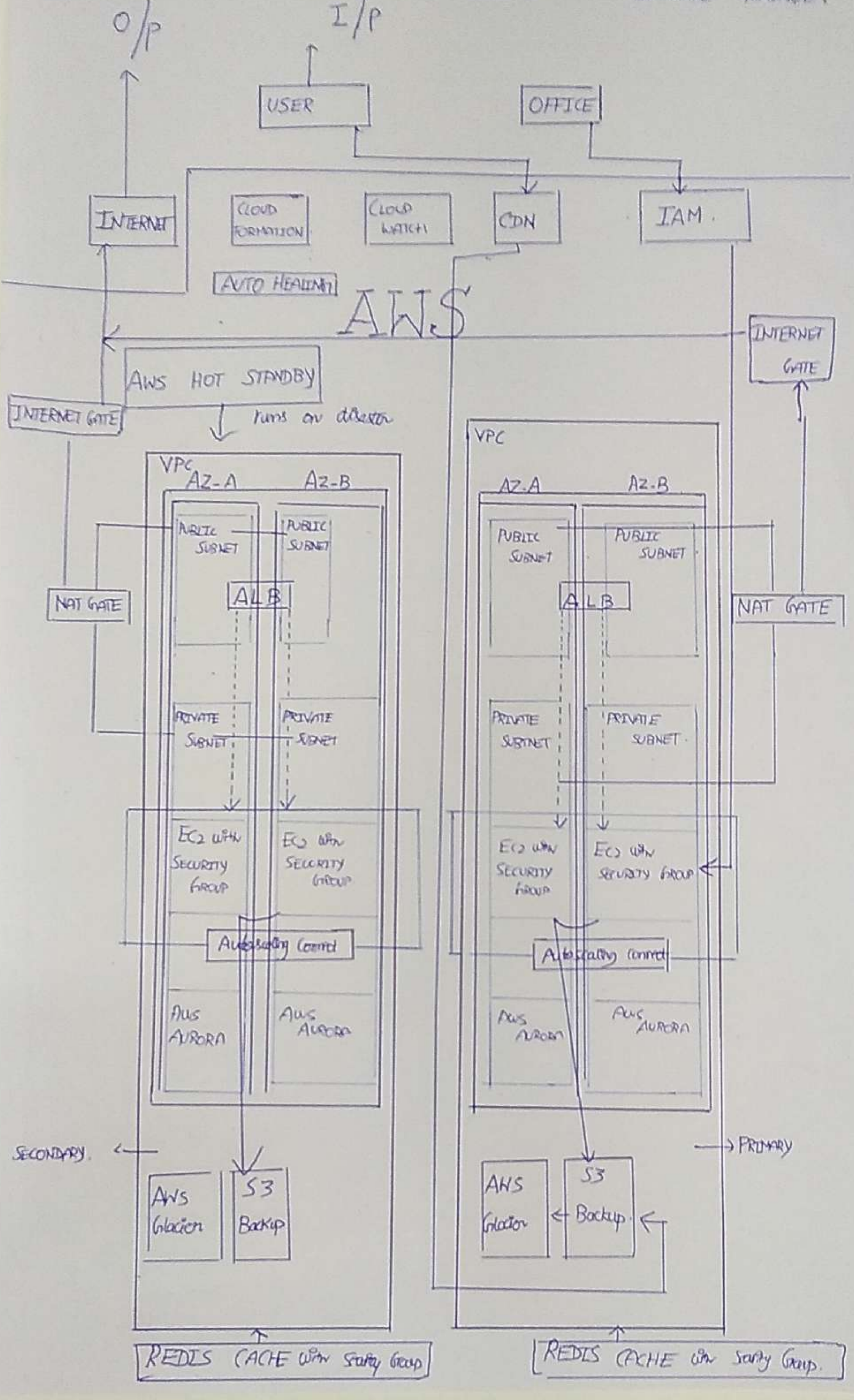
10. Ability to easily manage and replicate multiple environments based on their blueprint architecture. (Service : AWS CloudWatch, AWS CloudFormation)

- Use CloudWatch alarms to send repeated notifications.
- This architecture with AWS services can be converted into scripts/code in the form of CloudFormation templates and these templates can then be easily used to replicate environments.

Architecture flow can be expressed as below,

Architectural diagram

SIVAKUMAR P
SOFTWARE TRAINEE.



O/P
I/P
USER
OFFICE
INTERNET
CLOUD FORMATION
CLOUD WATCH
CDN
IAM.
AUTO HEALING
AWS
INTERNET GATE
AWS HOT STANDBY
runs on disaster
INTERNET GATE
VPC
AZ-A
AZ-B
PUBLIC SUBNET
PUBLIC SUBNET
ALB
NAT GATE
PRIVATE SUBNET
PRIVATE SUBNET
EC2 with SECURITY GROUP
EC2 with SECURITY GROUP
Autoscaling Control
AWS AURORA
AWS AURORA
VPC
AZ-A
AZ-B
PUBLIC SUBNET
PUBLIC SUBNET
ALB
NAT GATE
PRIVATE SUBNET
PRIVATE SUBNET
EC2 with SECURITY GROUP
EC2 with SECURITY GROUP
Autoscaling control
AWS AURORA
AWS AURORA
SECONDARY.
PRIMARY
AWS Glacier
S3 Backup
AWS Glacier
S3 Backup
REDIS CACHE with Security Group
REDIS CACHE with Security Group.