Proposed AWS Architecture For Startup - Logan Zhao

Table of Contents

1. Assumption of this case

Your current architecture doesn't meet the requirement for rapidly growing up business, such as unable to handle un-quantified growth requests and lacking the ability for disaster recovery and high possible performance in the coming future.

This document will provide recommendations for small startups to migrate your current web application to cloud and leverage AWS Cloud platform to build reliable, scalable, secure, and highly performing infrastructure for the most demanding web applications.

2. Executive Summary

In this section we will analyze your challenges your company are facing currently and requirements in the future. Provide abstract solutions using AWS cloud products and how it can articulate your current business needs.

2.1 Requirement Analysis

Existing IT infrastructure: Currently you are using LAMP stack with MySQL, Apache and PHP all running on **one desktop PC within a small office**.

With current infrastructure, the issues that you can foresee are:

- Current compute scale cannot handle the growing workload, but you are uncertain about how much workload it will be and will waste over-provisioned resources costs.
- Your current model is lack of disaster recovery plan with only one desktop PC within a small office.
- Your database and data access tier are lacking performance and throughput.
- High latency for remote location's user browsing experience.
- Unable to distribute the workload effectively to resources and unable to auto-recover failed instance.
- Limited access control for different team and security from data panel.
- Lack of archival strategy for "cold-tier" objects.

To solve those issues above, the new IT architecture requirements includes:

- The capability to that helps web applications scale to match regular traffic spikes can also handle an unexpected load, also reduce cost as much as possible.
- Able to provide disaster recovery for web application when one datacenter meet with unexpected faults such as PHU failure or nature disasters.
- Provision high performance and seamless scalability DB service.
- Provide low latency for users to access website content even from different global locations.
- Effective distribution of load and service healing capability for failed service instance.
- Security of data at rest and in transit and access control for role-based teams.
- An archival strategy for inactive objects greater than 6 months.
- The capability to easy manage and replicate to multiple environments for long-term development plan.

2.2 Solution Abstracts

We recommend you use AWS cloud to host your web applications to fulfill your business requirement above.

Solution Design Principles:

a. Elasticity

AWS can build elastic cloud service which is a perfect ecosystem for startup company. As a startup company, the business growth is unpredictable and traditional web hosting are generally provisioning instances ahead of projected traffic. In AWS, instances (using Auto Scaling Group) can be provisioned on the fly according to a set of triggers for scaling the fleet out and back in.

b. Cost-saving

As for all AWS service, you will only pay for resources that you use, no upfront investments needed, when given workload grows up, more EC2 instances will be added into the auto scaling group to provide more computing capability. Conversely, when given workload drops, EC2 instances will be decommissioned to save cost while providing enough computing capability.

Besides, AWS provide three different types of EC2 instances with different cost – on demand, reserved (for a specific period of time) and spot instances (bid on unused instances) and allow for flexibility in choice of size also.

c. High availability and disaster recovery

One of the considerations for the solution design is avoid single point of failure causing the entire service down. So loose-decoupling of services and reliability will be very essential. In web-hosting, AWS allows you to provision web-tier, application tier and DB service in different logical components. And each tier can spread their instances into multiple availability zones to provide redundancy.

d. Performance Efficiency

AWS leverages the following services to lower latency from global user experience and DB performance tuning. Elastic Load Balancer will distribute inbound workload to different EC2 instances, and use CloudFront to reduce the website response time, it is a CDN service and will cache the static content of website in nearest edge server when users are visiting the website. EC2 instance can scale up/scale out depending on your performance requirement, as well as AWS DB service. Elastic Cache is in-memory application caches can reduce load on services and improve performance and scalability on the database tier by caching frequently used information.

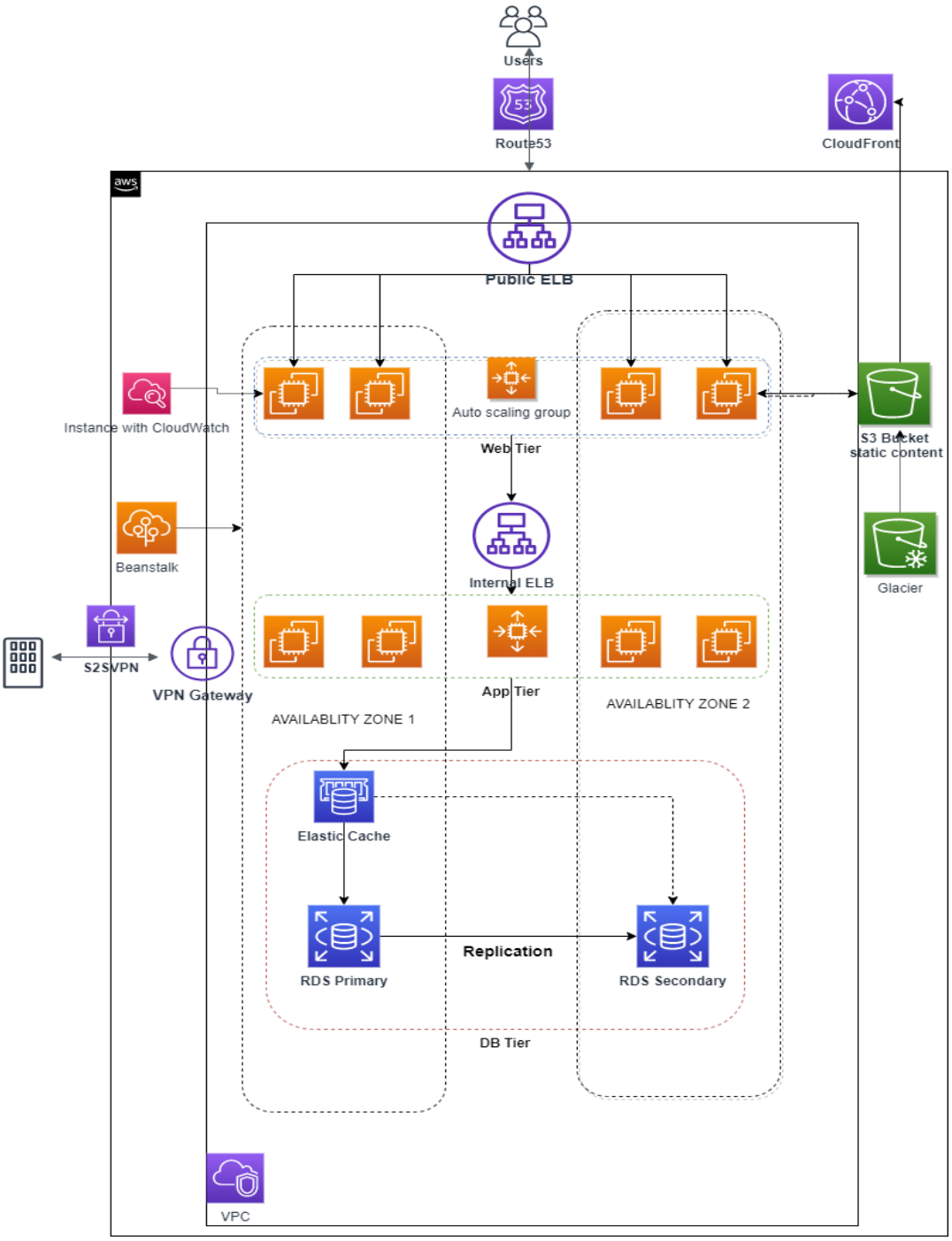
e. Security and compliance

Security and compliance is the highest priority in AWS. As an AWS customer, you will benefit from a data center and network architecture built to meet the requirements of the most security-sensitive organizations. From administrative perspective, Identity & Access Management (IAM) which enables you to manage users and user permissions and secures access to the environment. From data perspective, Virtual Private Cloud (VPC) and security group provide a logical isolated virtual network, and virtual firewall to limit access.

3. Solution Design

3.1 Architecture Design

Overall architecure design are as below:



Here are the key points:

Web and app tier: Beanstalk, CloudWatch, Auto Scaling Group, Availability Zone, EC2.

Network tier: Elastic Load Balancer, CloudFront, VPC, Security Group, Amazon Route 53, Elastic IP, VPN gateway

Data tier: S3 storage service, S3 Glacier, RDS, ElasticCache.

3.2 Design Details

1. Firstly, users access the website via Public DNS name of an Amazon Route53. **Amazon Route53** here is to resolve DNS name to a good-working site Elastic IP. In most case, it will route traffic to primary site. However, if there is any disruption against primary site, the traffic will be re-directed to DR site for business reliability.

2. Because we configure **CloudFront** to deliver static content of website for better performance, the user request will go to CloudFront next. CloudFront here is configured to work with web servers and S3 storages provisioned along with Elastic Beanstalk service. CloudFront will cache the static contents in edge servers. User requests to static content will be responded from edge servers which is in nearest locations to users for lower latency.

3. For dynamic content or content that is not cached by CloudFront, user requests will go to **Public Elastic Load balancer** provisioned together with the Elastic Beanstalk. ELB distribute http(s) traffic to web tier EC2 instances in an auto-scaling group. Also EC2 instances in an auto-scaling group can be configured into different **availability zones** to achieve high availability of business.

4. Regarding network tier design, Elastic Load Balancer will route traffic to backend EC2 instances evenly. Different tier services are deployed in different subnets aligned to different availability zones. All these subnets will belong to a regional **VPC**. Subnet will be associated with **Security Group** configuration to restrict inbound and outbound traffic for access control. Also, you can set up **Site to Site VPN** for corp users with secure connection from on-premises to AWS VPC.

5. For data tier, we have three kinds of storage service in this architecture.

a) **S3 service** is to store static content of websites. S3 service provides a highly scalable, reliable, fast, inexpensive data storage infrastructure and easy to manage. All the static content will be stored in S3 services. Additionally, S3 service will provide encryption feature to protect data at rest.

b) **S3 Glacier service** is for archive purpose to store out-of-date data retired from S3 service. Like S3 service, Glacier could enable with encryption feature to protect data at rest.

c) **Relational Database Service** is a managed database service similar with MySQL, PostgreSQL etc. Rational database will create a standby instance in another availability zone. Data will be synchronized from primary to standby DB. If there is any outage with primary DB, standby DB in another availability zone will take over to serve data requests.

d) **ElasticCache** is an optional component here to enhance the RDS performance by creating read replica for heavy DB workloads (in memory caching using MemcacheD or Redis engines).

3.3 Migration Plan:

I suggest you can divide into three phases for the migration:

Phase	Description of plan	Duration

1	Start to create AWS resource and build infrastructure	2 weeks for implementation of resources
2	Running workloads from on-premises and AWS in parallel and perform load test	2 weeks for run benchmarking and troubleshoot any break-fix issue
3	Dismission on-prem servers and run workloads on AWS	1 week to monitor if the migration goes smoothly

3.4 Summary

We focus on the key inputs for your business and deliver them with the right quality and in a timely fashion. The above solution provides zero upfront investment, efficient resource utilization, and usage-based costing and zero-day time to market. Customer will achieve a cost-effective, auto-scaling, fault-tolerant, operability-easy and elastic cloud infrastructure with even better performance.

Appendix

https://aws.amazon.com/security/
https://aws.amazon.com/getting-started/projects/launch-lamp-web-app/
https://aws.amazon.com/disaster-recovery/?nc1=h_ls
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/php-hatutorial.html?icmpid=docs_tutorial_projects
https://aws.amazon.com/cloudfront/
https://aws.amazon.com/blogs/aws/new-cross-region-replication-for-amazon-s3/
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html