Amazon Web Services
25/05/2016
Lucian Maly <lucian.maly@oracle.com>

Proposal for

Small startup company (interview)

I. Objective

Dear customer of <<*Small startup company*>>, thank you for considering Amazon Web Services as your future platform. This document contains our recommendations for manageable, secure, scalable, high performing, efficient, elastic, highly available, fault tolerant and recoverable cloud architecture that allows your organization to grow organically.

II. Problems

Your current situation does not allow for un-quantified growth and is lacking prevention of disaster. This is major risk for your operation and can potentially affect your business at any time.

III. Goals

Analyze the <<*Small startup company*>> requirements and provide solutions using various Amazon Web Services offerings, provide an architecture diagram and state all assumptions made during the design and explicitly state how Amazon Web Services will help overcome the current problems specific to the company. Reader is expected to have basic computer knowledge, but not necessarily Amazon Web Services exposure.

IV. Recommendations

When architecting for the cloud, one must assume that anything can fail at any time. In order to prevent that from happening, it is necessary to break down its components into smaller, decoupled pieces and work with each of them separately, that is – separate computing, storage, and database and apply principles to each.

Amazon Web Services brings lot of built-in features to address business continuity. I recommend using **Elastic Load Balancing (ELB)** and multiple **Availability Zones** (essentially separate data centers in the same region) for all of your servers. Not only that, it will effectively distribute load among your **Elastic Compute Cloud (EC2)** Linux servers, but also ensures that your services will be unaffected if one data center becomes unavailable for some reason.

We provide three different types of EC2 instances – **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. I recommend starting with smaller on-demand instances and once you understand the level of workload, choose the right size and combine on demand and reserved instances. The options are endless.

In certain scenarios, such as when flash traffic is expected, or in the case where a load test cannot be configured to gradually increase traffic, we recommend that you contact us to have your load balancer "**pre-warmed**". We also recommend using only secure protocols listeners and enabling **SSL termination** on the ELB to release load on the backend instances.

Your company would particularly benefit from replicating the entire environment to another AWS account using **CloudFormation** (using templates) or **Cloudformer** (snapshot of the whole infrastructure). This will give you the ability to leverage two exactly the same environments – either for testing purposes or in combination with the highly available DNS service **Route 53** and **Machine Images (AMI)** (snapshot of your server) in different regions in a stand-by or multi-site scenario.

Just make sure you **tag** (key/value pair to easily identify resources) your instances from day one – it will makes things easier later on when you have many more resources to manage. Both AWS accounts can be linked using **consolidated billing** which gives you the ability to see it all in one place.

Security of data in transit is taken with the utmost priority. All of the virtual servers are completely isolated by design, and on the top of that, one of the most important networking features we provide is resource isolation using **Virtual Private Cloud (VPC)**. **Security group** acts as a virtual firewall for your instances in to control inbound and outbound VPC traffic and **Network Access Lists (NACL)** is another, optional layer of security for your VPC that controls traffic between one or more subnets.

Scaling to meet the demand with uncertainty around when and how much this demand will be, is extremely easy with Amazon Web Services. Our EC2s come with elasticity out of the box. When configured to do so, **Scaling Groups** (template of EC2 instances) and **Auto Scaling** (rules for scaling) will automatically increase the amount of instances when the load is high (or at a specific time) and decreases the amount, when the load is low. You don't have to buy too much infrastructure too soon or not enough too late, it is all happening automatically.

I also highly recommend specifying **user data** while creating EC2 instances to run a configuration script during launch. This is a simple auto-healing/bootstrapping option that can for example fetch the default configuration from highly durable storage **Simple Storage Service (S3)** (online file storage) and will ensure failed/unhealthy instances are always replaced and restarted with the default configuration. Server side encryption is possible with S3, so you can be absolutely confident that your data is secure.

Later on, I definitely suggest registering existing instances to **OpsWorks** and automate operations in your stack. This gives you an even better self-healing infrastructure option that recovers from failed service instances using the native auto healing feature of OpsWorks.

It would be a great benefit for your company <<*Small startup company*>> if objects with infrequent access in S3 (with either **standard** or **reduced redundancy** type of storage) will be automatically archived by **lifecycle management rules** to **Glacier**. That is our leading lower cost archiving product with 3-5 hours retrieval policy.

The ability to configure your database and data access layer for high performance and throughput is a matter of a few clicks in the **AWS console**. Our managed **Relational Database Service (RDS)** provides MySQL engine and is fully scalable and has multi AZ failover using synchronous physical replication and automatic backups by default. If that is not enough, you can also very easily create read replica(s) for heavy database workloads or improve performance with **ElastiCache** (in memory caching using MemcacheD or Redis engines).

If a large portion of your user base is long distance, we can significantly help you improve latency and user experience in their browsers using **CloudFront** (content delivery network). This allows distributing objects from S3 to the edge locations/endpoints closer to your customers and they will always have a better chance of retrieving the file(s) from closer to them.

We take security very seriously here at Amazon Web Services and provide many different levels of protection. The first layer of security is **Identity & Access Management (IAM)** which enables you to manage users and user permissions and secures access to the environment. The rule of thumb is to grant minimum possible access that is necessary. Each user can have certain policies attached to his/her account, allowing him to use specific AWS resources. I recommend using user groups (at least three: administrators, power users and read-only) for easier user management especially if you expect team expansion. Always make sure you enable **multi-factor authentication (MFA)** on all of your accounts which adds another layer of protection. We also support SAML protocol which simplifies authentication and user migration if you already have users on your directory server.

V. Timetable

I recommend breaking the project into phases, and have a schedule for each phase. Times will vary based on the size of the current setup, but for standard architecture of this type, you should expect that:

Table 1

Phase	**Description of Work**	**Duration**
Phase *One*	Create resources in AWS	Immediately
Phase *Two*	Migrate your on-premise to AWS, both systems will run in parallel, perform tests	~1 month
Phase *Three*	Shut down on premise and only use AWS	~1 day

VI. Budget

One of the major benefits of our platform is that you only pay for what you use (pay-as-you-go model). It is fair to say that at the early stages, most of the cost will come from compute and storage. Throughout the time, it will become clearer what the cost is during busy and idle periods. To give you some idea, I included two of the most typical similar scenarios and corresponding costs:

Table 2

Type	Region	URL	Anticipated Costs
Simple marketing site	Sydney	https://calculator.s3.amazonaws.com/index.html#key=calc-19ED1A76-86E5-4E5B-BCFA-009D565A125A&r=SYD	~$230 (USD per month per region)
Large on-demand web site	Sydney	https://calculator.s3.amazonaws.com/index.html#key=calc-1B74B8A3-430B-42D9-A294-9B0CD82395E7&r=SYD	~$1280 (USD per month per region)

VII. Next step

Our solutions team is here for you 24/7 to discuss and evaluate progress throughout and at the end of the project. Our website http://aws.amazon.com/ is also great source of information and please keep in mind that we have four different types of extremely helpful post-sales technical support.

VIII. Conclusion

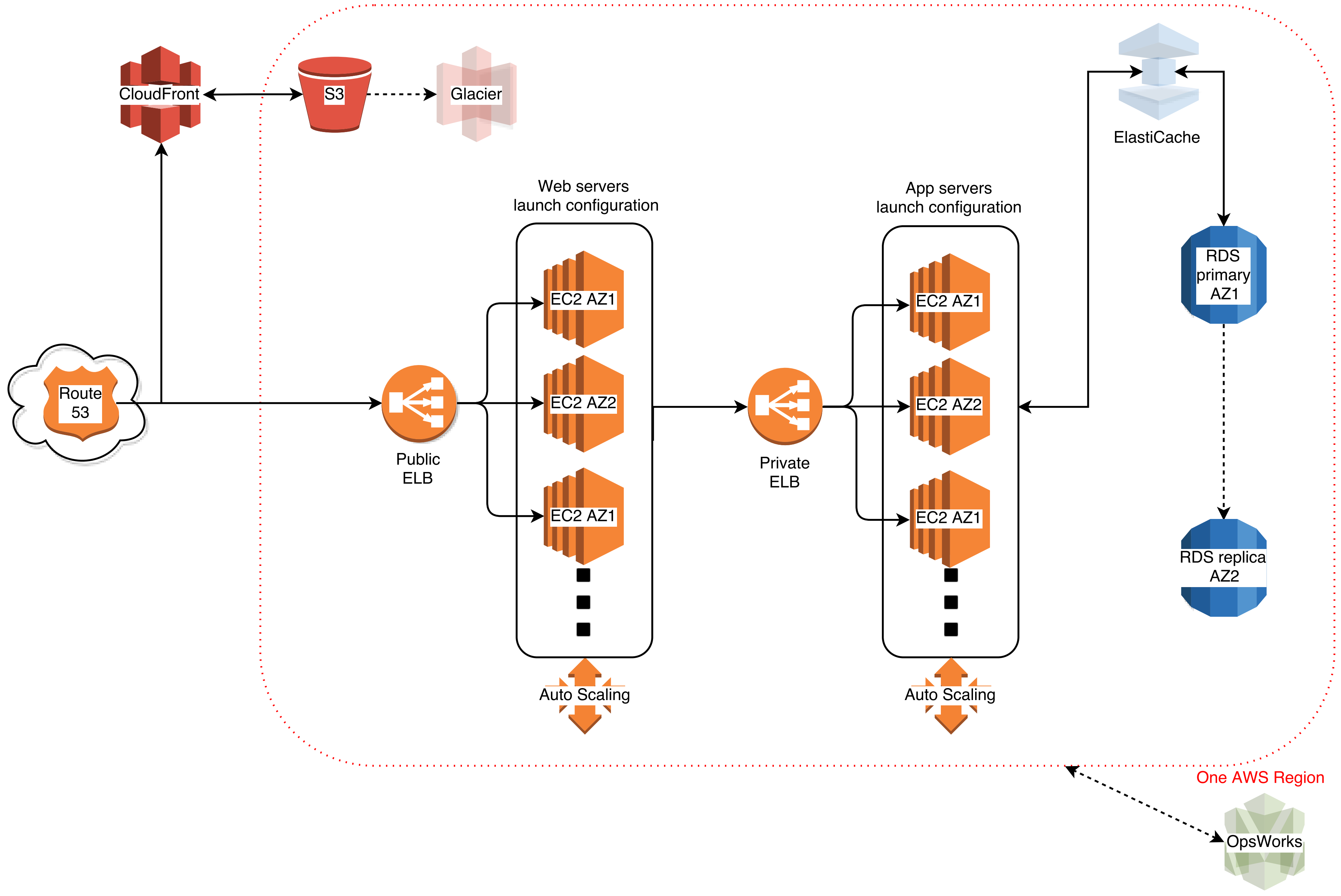
We focus on the key inputs for your business and deliver them with the right quality and in a timely fashion. The above solution provides zero upfront investment, efficient resource utilization, and usage-based costing and zero-day time to market. Do not hesitate to contact me anytime, should you have any further questions. On the behalf of Amazon Web Services I wish you a great success and I sincerely hope we will be your choice and companion in your cloud journey.

Appendix A

Attached is the schema of the proposed solution.

Legend:

- *ELB* = elastic load balancer
- *EC2* = elastic compute cloud
- *AZ1*, *AZ2* = different availability zones
- *Low opacity icons* (Glacier, ElastiCache, OpsWorks) = highly recommended, but not required



Lucian Maly 25/05/2016 - Draw.io