

Cloud Hosting - Amazon Web Services

Thomas Floracks

When talking about hosting for web applications most companies think about renting servers or buying their own servers. The servers and the network that connects them to each other and to the Internet are the infrastructure to run a web application. With the introduction of *AWS (Amazon Web Services)*, Amazon started to provide this infrastructure of servers, network and mass storage as a service to run any kind of application in the cloud.

When trying to categorize the different offerings of Cloud Computing in the market the most common categorization might be the following:

- Infrastructure as a service
- Platform as a service
- Software as a service

The common denominator is “service” meaning that in any of the three cases the software, platform or infrastructure are not owned by the client but are rather a service that the client rents from a provider. Most Amazon services belong to the infrastructure category, although some service might better be classified as platform services.

Our company, VivaReal, has been successfully using Amazon Web Services since 2007. In this article you will learn how some Amazon Web Services

work and how we use them to host our web application.

Hosting on Amazon EC2 (AWS) - VivaReal Network

In 2006 Amazon added two services to their portfolio of web services called *S3 (Simple Storage Service)* and *EC2 (Elastic Computing Cloud)* [1]. When this infrastructure was made available to the public as a limited beta 2006 it was one of the first approaches to cloud computing and cloud hosting on the Internet.

The EC2 infrastructure is a virtualized data center and server farm. On top of this virtualization layer customers can launch server *instances* through simple API calls or a web interface. An instance is a virtual server with predefined hardware characteristics. When launching an instance there are different options for the hardware characteristics of the server - CPU, memory, disk space, 32bit/64bit and I/O performance. Instances range from a single CPU core, 1.7GB RAM, 400GB disk space to eight cores, 15.5 GB RAM and 1.2TB disk space. The assignment of all resources to an instance is static and guaranteed.

To launch a new server instance the user has to provide an *AMI (Amazon Machine Image)*. The AMI provides the pre-configured operating system of the new server including all necessary

software. The operating system and software included in the AMI will be available within less than a minute after launching a new instance with an AMI. An instance behaves the same way as a “normal”, standalone server as the virtualization layer is never visible to the operating system.

There are some pre-configured, official AMIs to choose from when launching a new instance, for example Windows Server, various Linux distributions or Open Solaris images. Users can also choose from AMIs that have been provided by companies like SUN - but most importantly Amazon allows the user to create his own AMI. An easy example would be the creation of a typical LAMP server (Linux, Apache, MySQL, PHP) AMI. On top of one of the existing Linux images, users would launch a server instance with a Linux AMI, connect through SSH and start installing and configuring all software needed for the LAMP server. Once the server and services are configured, users can create their own AMI by writing an image of the configured server to the *S3 (Simple Storage Service)*. Once stored on the S3 this new AMI can be used to launch a fully configured LAMP server, whenever needed. S3 is the complementary service to the EC2 - a persistent mass storage service with “100%” availability and data integrity.

This flexibility was one of the key factors why we chose to use EC2 for hosting. As a start-up we were often unsure about hardware requirements and we had no time or resources to do extensive load testing or scalability simulations. So we just had to jump in and hope that everything goes well. Amazon gave us the possibility to be totally flexible with the amount of servers and hardware configurations. In other classical hosting or collocation services, where we had to rent or buy the hardware, upgrades and changes to the hardware would have been very expensive.

For billing out EC2 and S3 use, Amazon uses the *utility computing* concept. Customers only pay the resources they actually use. EC2 resource use is billed on an hourly basis and the S3 use based on the amount of storage a client uses. For all Amazon services customers also pay all external (Internet) traffic per GB transferred.

The advantages over a classical rented or collocation hosting are:

- no upfront investment
- no long term contracts
- payments for hour of server (instance) use

Especially for startups this model is very attractive. There is no risk of buying / renting the wrong servers or too many servers.

Another advantage is the flexibility to simulate and test a system at any time. For stress testing a company can replicate the whole platform and only pay the 24 hours of server time they were actually running tests. Compared to the cost of renting or buying all the servers needed to deploy a 1:1 replica of a production system, cloud hosting offers a tremendous cost advantage over classic hosting. Many companies simply cannot afford to purchase twice the amount of server to replicate a test environment and often times have to use scaled down versions of their production system for testing - with all the risks involved. Cloud hosting and utility computing make this problem an issue of the past.

In the two years we have been using AWS new services have been added or existing services have been improved almost every month. In the following I will line out which services we are using today and how we benefit as a business.

Our experience after 2 years of hosting in the cloud

In 2007 our company “VivaReal” was a small Internet start-up that had just developed the first version of a system to host a number of real estate marketplaces in the Americas. With the launch date coming closer we had to make the decision on where to host our web servers, database servers and

application servers. We decided to use two Amazon EC2 instances for our initial hosting and since then have grown our infrastructure by adding more EC2 instances and AWS services. Today the VivaReal Network includes 6 marketplaces with individual domain names all sharing one central property database of around one million properties.

The VivaReal Network infrastructure consists of 3 layers: web servers, applications servers and backend (database) servers. In total we use 16 EC2 server instances to run the VivaReal platform. Each marketplace (domain) has its own instance running on Apache TomCat 6.x. For the TomCat web servers we use a medium instance with 1.7GB of memory and two processor cores running at 2.5GHz. Additional Amazon services allow us to dynamically scale the number of servers of each marketplace up and down depending on the load. These services are *Cloud Watch*, *Auto Scaling* and *Elastic Load Balancing*. Cloud Watch allows us to monitor the use of resources of each Web Server, e.g. free memory, CPU time or bandwidth used. When a web server has a high load and one of the monitored resources passes a critical limit the Cloud Watch service reports to the elastic load balancing service which will then use the auto scaling service to replicate the web server and divide the load between two TomCats.

As Amazon only charges for the resources we actually use this service helps cut cost as we do not have to buy the hardware for the highest load scenario. When needed the amount of servers scales up (during the day) and when not needed (at night) the additional servers will shut down and we don't pay for them anymore. The maximum and minimum amount of servers can be defined when using elastic load balancing. For example we can define that minimum two servers are used for a marketplace and that the maximum scale up are 4 servers.

The elastic load balancing additionally provides a basic fail-over mechanism. If one server stops responding on a port monitored by Cloudwatch the elastic load balancer will shut down the instance and replace it with a new, functional instance.

Unfortunately the load balancing provided by the Amazon services is low level TCP. It does not respect sticky user session nor does it offer any advanced, high level load balancing based on content rules.

Another AWS service that has proven to be very helpful in the VivaReal platform is *SQS (Simple Query Service)*. SQS provides a FiFo queue for plain text objects. The lifetime of an object in the queue is 4 days the maximum allowed size of an object 8kb. This

service is billed by the amount of put and get requests made to the queue. It has proven to be very useful whenever our application requires an asynchronous behavior. A good example in our system is the processing of XML feeds. We receive massive amounts of properties in XML format, sometimes over 100.000 property entries in one XML file. Our system starts processing the feeds and writes objects to the SQS queue. These objects are read from the queue and written to our database in a controlled way by another process. If we would write all information directly to the database we would risk blocking or overloading the database server. The asynchronous processing with the help of the SQS allows us to buffer the information in the queue and intelligently process it depending on the load on our database servers.

The last service I want to present is *Cloud Front* a content delivery network that runs on top of the S3 storage. S3 was designed to be a data storage, not fitted for delivering content to a large amount of users on the Web. But Amazon recently added a new service called “cloud front” to it’s line up to make objects stored on S3 deliverable to a large amount of users. Objects stored on S3 can be marked for use on Cloud Front which will replicate them on many data centers around the globe for quick delivery. We use cloud front

to deliver property photos and this way take load off our web servers. Cloud Front is billed for each GB transferred.

When keeping information and vital services in a cloud environment security is always an issue. Amazon uses a strict key-base approach for access to instances. A user gets a key pair to access an instance and to launch new instances. This is a secure approach compared to password based logins. The keys can be generated or changed from a master account at any time. In order to mitigate the risk of unwanted access to this master account Amazon offers optional multi factor authentication. If a company decides to use this extra security layer email, password and a 6 digit key generated by a physical device are required in order to login to the master account.

Standard firewall rules are applied to each instance. By default all ports but the ones required for SSH access are closed. The rules can be changed for each instance after or before launching it by defining a firewall rule set. The security of each instance in an account is sole responsibility of the user. Hacked instances that form part of a bot net or that send spam are considered a violation of Amazon’s terms of use. Repeated violation of these terms will result in a suspension of the user’s account.

Other crucial factors when choosing our hosting service provider are reliability and availability. Amazon offers three different zones (basically data centers) when launching an instance - two on the US east coast and one in Europe. If an entire data center fails the application can be launched in any of the two other data centers or, if cost is not an issue, a company could permanently mirror all data and applications between two zones. To date we have not experienced a massive failure of a data center.

The Amazon service level agreement aims to guarantee a 99.9% availability of all services. Basic email support is included and support contracts with 24*7 phone support are available at an added cost.

Limitations of AWS

Nothing is perfect and neither are the AWS services. There are certain limitations that make the use of EC2, SQS and Co difficult or impossible in certain cases.

First of all there is only a fixed set of possible hardware configurations available. Some parts of our application would ideally be hosted on a 4GB server. But there is no such configuration on Amazon, so we had to find a way to make it work on a 1.7 GB server and scale vertically or we had to accept the cost of an over dimensioned

7.8 GB server. Before starting with EC2 companies should evaluate if the available hardware configurations fit their needs.

Although the network between the EC2 servers, SQS and S3 is very fast and reliable, multi casting is not permitted. Many common cluster configurations, for example for Jboss, use multi casting and will therefore not work on the Amazon infrastructure. Most of the times alternative cluster implementations are available but it might be a time intensive task to find and configure a good solution.

The whole EC2 infrastructure is virtualized, so if one of your virtual server crashes due to failure of the underlying hardware all data written to virtual hard drives will be lost.

This problem can be solved by using a service called “*block storage*” which is a persistent storage that can be attached to a virtual server. Still, all crucial data should be written to the S3 storage all the time as this is the only 100% persistent and reliable storage option in the Amazon infrastructure. Additionally data can be copied to local servers at any time. If huge amounts of data have to be copied from or to the S3 Amazon gives companies the possibility to mail a hard drives or backup tapes. Amazon will transfer the data between the backup device and the S3

send the device back to the user. This service is called “Import/ Export” and may offer big time savings for companies that want to transfer terabytes of data to the cloud.

Security risks exist when using “unofficial” AMIs. Besides the AMIs provided by Amazon and partners, like SUN, Microsoft or IBM, there are other “unofficial” AMIs that are shared by other AWS users. Amazon does not guarantee the well functioning of these AMIs and only runs a basic security check. There are known cases where AMIs with preinstalled spyware have been shared by other users. We only use official AMIs for this reason.

Disc access on EC2 instances is generally slower than on a normal server. If a company plans to migrate an application to EC2 that relies on a fast HDD raid setup EC2 might not be the right choice.

We designed our application to keep all vital data in RAM in 95% of the cases. One of the main requirements for our websites are fast response times that we can only achieve by using “in memory” - databases and caches to avoid disc access.

Especially when an application makes heavy use of Amazon services like SQS or load balancing companies create a huge dependency on Ama-

zon as their provider. If a company rented or owned their servers they could easily migrate to another provider by transporting their servers or by renting an identical hardware configuration somewhere else. This is not possible anymore when using AWS services that are called through an API. Moving away from Amazon means changing the application in this case. The dependency on the provider is even more notorious with “platform as service” providers such as Google App Engine or “Software as a Service” providers such as Salesforce or Google Docs [2].

Other Cloud hosting providers

Of course Amazon is not the only provider of infrastructure as a service anymore. Many other companies are offering similar services some more specialized than others:

- Rackspacecloud, <http://www.rackspacecloud.com>
- Microsoft Azure Platform, <http://www.microsoft.com/azure/>
- Sun Cloud, <http://www.sun.com/solutions/cloudcomputing/>

This is just a selection of possible providers. Depending on the project one might be a better fit than the other.

For companies that want to go one step further there is also the alternative of using “platform as a service” - provi-

ders such as Google App Engine. The platform provides all services needed to develop your applications and scale them automatically so that a company does not have to worry about the infrastructure or server technology - the complete development platform is already there. The downside is that the services have to be used exactly the way they were planned. There is no freedom of technology, for example in terms of the database, programming languages or frameworks. The dependency on the

provider is very high as the applications will only work on the providers platform and nowhere else.

Referencias

[1] *Amazon Web Services Website*, <http://aws.amazon.com/>

[2] *Richard Stallmann, Guardian Newspaper Website, 2008* <http://www.guardian.co.uk/technology/2008/sep/29/cloud.computing.richard.stallman>

About the author. *Thomas Floracks is a German citizen and the COO (Chief Operation Officer) of the VivaReal Network, a real estate marketplace covering numerous markets in Latin America. VivaReal operates 6 real estate portals with operations focusing on Brazil, Colombia, the United States and Mexico. VivaReal's head quarter and main production office is located in Bogotá, Colombia. Since moving to Bogotá 5 years ago Thomas manages the development of the VivaReal project. His specialties include the definition of information architectures (IA) and search engine optimization (SEO).*