Moving Woophy.com to Amazon Web Services

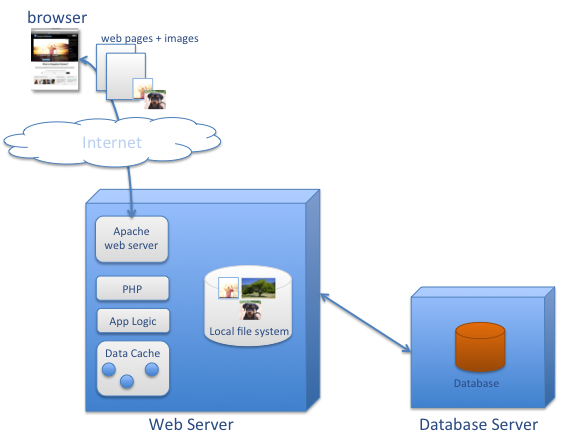
Motiviation

In order to reduce operational costs and increase end-user experience, we will move the woophy.com website’s servers from their current location (2 servers running in the Netherlands) onto the world-wide Amazon Web Services platform. A number of the benefits that we’ll gain from this move are:

* Images will be served from a world-wide CDN (content delivery network). This will ensure faster page loads and a better image viewing experience regardless of where on the planet you are accessing woophy.com. (*In the currently deployed architecture, all images sit on a single web server – which means that server is handling all image access load for the entire planet. In addition, this single server serves as both the single web server of the system and the single place images are stored – making it a SINGLE POINT OF FAILURE.*)
* The web server and database server will be deployed within and managed by this Amazon Web Services “cloud” infrastructure. This will allow us to dynamically scale the horsepower (CPU speeds, memory, disc size and speed, etc.) of the woofy.com servers up and down based on end-user demand.
* Once woophy.com is deployed to Amazon Web Services, its network/systems architecture will be very similar to that of the HSM “image social network” and SBM. This will allow woofy.com to leverage tools and techniques developed by HSM as it builds out the HSM “image social network” and SBM.

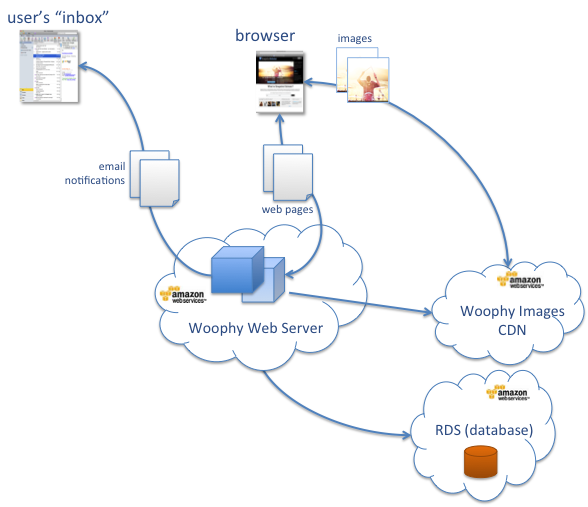
Current woofy.com Architecture

This diagram illustrates the current woofie.com architecture. Currently, all user traffic and all image storage for woofie.com are provided by 2 servers located in a web hosting facility in the Netherlands. This means that all requests for woofie images from anywhere in the world are satisfied by a single web server – that could go down any time and bring down the entire woofie.com product and community.



Proposed New woofy.com Architecture

This diagram illustrates the proposed architecture for woofie.com. This new architecture leverages the experience, robustness, and world-wide reach of the Amazon Web Services “cloud.” By deploying woofie.com in the “cloud,” we can eliminate single points of failure while improving the overall performance of the site.



Transitioning woofy.com to Amazon Web Services “cloud”

These are the high-level steps that must be followed in order to transition woofy.com from its current 2-servers-in-the-Netherlands deployment to the Amazon Web Services cloud.

1. Build set of regression test cases that we’ll run (likely by hand) to verify that any changes we make to woofy.com didn’t break anything!
2. Move all woofy.com source code into version control. (Currently the source code is not version controlled.)
3. Configure and deploy a “qa/dev/test” EC2 (elastic cloud) server in AWS (Amazon Web Services)
4. Configure Amazon CloudFront and S3 to act as the CDN for woophy.com images
5. Write and test changes to woophy.com source code that uploads images to the CDN (rather than to the local web server like woofy.com currently does).
6. Deploy changes to production server (in the Netherlands) so that from that time forward, all woofy.com images are served up by the CDN.
7. Create an Amazon RDS (database services) machine that will serve as the database server for woofy.com. Configure the database to handle both a test database and the production database.
8. Create a “production” EC2 server. Configure it for running woofy.com in production.
9. Modify woofy.com source code to use the RDS database. Test.
10. Migrate database data to new “production” RDS database. Migrate DNS to point woofy.com to our new Amazon Web Services deployment.

Transitioning steps Diagram



Transitioning details…

The sections below provide more detail regarding the large transition steps.

Step 3) Configure QA/Dev/Test EC2 Server

* Get a good starting point AMI (Amazon Machine Image) upon which we’ll base our server (it defines server size, OS, services, etc.)
* Create an EC2 instance
* Configure server with all the extra stuff: memcached, PHP, etc.
* Copy/rsync all the images from production Netherlands woofy.com web server to this server
* Create a local MySQL database. Populate it with a dump from the production woofy.com database.
* Get all production code onto this machine.
* Create required DNS entries (test.woofy.com, etc.)
* Test this using the test cases we have designed earlier.

Step 4) Configure Amazon CloudFront and S3 to act as the CDN for woophy.com images

* Create S3 buckets (for test and production)
* Attach CloudFront to S3 buckets
* Create appropriate DNS entries for test and production image CDNs

Step 5) Write and test “upload to S3” code (to replace how images are currently stored)

* Write rewrite rule that maps [www.woophy.com/images/bla](http://www.woophy.com/images/bla) onto a CDN URL with permanent redirect.
* Modify all places within the woophy.com code that generates screens that have URLs. Changes it so that it generates URLs like cdn.woophy.com/foo.jpg
* Change upload code in woophy.com so that it uploads to S3 rather than to the local disk.
* Design and write and test a script that uploads all images from the woophy.com web server into S3.
* Design and write and test a script that uploads all images that have been added to the woophy.com since the last time they were uploaded.
* Add feature to allow us to turn image upload functionality on/off.

Step 6) Deploy “upload to S3” changes to production server in the Netherlands

* Do big image upload from the real live woophy.com server to S3.
* Do this all in one transaction:

1. Turn off all upload capabilities (this will only be off for a few minutes.)
2. Run the copy all recent images upload to S3
3. Copy all the new “upload to S3” code changes to the woophy.com server in the Netherlands
4. Turn “uploads” back on. From this point forward, any uploads will upload to S3. And any web pages people see served up from woophy.com will have image urls like: cdn.woophy.com/mydog.jpg

Step 7) Create an Amazon RDS (database services) machine

* Provision a machine large enough for production + test
* Configure the database with same attributes as woophy.com database
* Load a dump of production database into “production” and “test” databases.
* Prepare to be able to have the production database a “replication slave” of the database on the production Netherlands woophy.com server.
* Modify source code/config on test EC2 server to use this RDS “test” database.
* Run complete regression test against test/qa EC2 server.

Step 8) Create a “production” EC2 server. Deploy. Test (in “alpha” mode)

* Get a good starting point AMI (Amazon Machine Image) upon which we’ll base our server (it defines server size, OS, services, etc.)
* Create an EC2 instance
* Configure server with all the extra stuff: memcached, PHP, etc.
* This machine is a “production sized” machine (as opposed to the smaller test/qa machine).
* Once the machine is properly configured, store it as a private AMI (so we can create clones of it later)
* Check out all woofy.com source code to this server.
* Point to “test” RDS database.
* Test thoroughly in “test” mode (using test database and test CDN)

Step 9) Modify and test woofy.com source code to use the RDS database.

* Point to “test” RDS database.
* Test thoroughly in “test” mode (using test database and test CDN)

Step 10) Migrate Database and woofy.com “production” over to AWS servers

* Shorten the DNS entries for woofy.com to TTL (time to live) of 1 hour. After a day or so, you can do the next steps.
* Set up replication between live Netherlands woophy.com database server and our “production” RDS instance. So for a while the database would feel a bit slower to users, but we’d be ensuring that our RDS database data is up to date.
* (To reiterate, at this point, any INSERT, UPDATE, DELETEs to the woophy.com database will also exist in our copy of the production database!)
* Now do all of these in a single “transaction” (not literally, but do the all nearly at the same time).
  + Configure the server to prevent any database updates for 1 hour
  + Turn off replication from the woophy.com Netherlands database
  + Turn off “slave replication” to the Amazon RDS “production” database
  + Change DNS to point woophy.com to our Amazon EC2 server (within 1 hour, all accesses of woophy.com will actually point the user’s browser to the Amazon EC2)
  + (NOTE: For only 1 hour a small subset of users will hit the old “Netherlands” woofy.com server and won’t be able to do database updates. Many people though, will be directed to the new Amazon version of woophy.com. And they WILL! be able to do database updates.
* 1+ hour after doing the previous task, no one will be connecting to the original Netherlands woophy.com server.
* Wait a day. Modify the woophy.com DNS entry to have a TTL of 24 hours (or whatever is reasonable).
* We are now ported over to Amazon Web Services!

My original notes regarding moving woofy.com (from me, Marcel, etc.)

My first thoughts about moving Woophy to another server:

1. Set up new server: LAMP, configure apache and PHP (Rewrite Rules, UTF-8 support). Make sure all necessary PHP extensions are installed (gd, mbstring, memcache, mysql, simplexml, exif, uploadprogress)

2. Copy application files to new server, without user input. So no photos and empty database. Make sure every functionality (register, upload, comment, rate, mail etc.) works. Remove input after testing.

3. Close down old website or, more friendly, only disable user input. This requires some extra work. Not disabling input and synchronizing data after moving requires a lot of extra work. I would go for disabling input.

4. Move photos and database.

5. Change DNS records.

/////////// THOUGHTS ////////

1) Put all their code into version control

2) Configure an AWS server and RDS server for deployment

3) Other stuff?...

Use "rsync" to keep files in two servers in sync

///////////// FROM MARCEL //////////

Consider "sync"???

http://www.123cloud.eu/123cloud-hosting-products/public-cloud

Marco van den Akker | vda.hosting 1:12 PM

123Cloud is also offering cloud solutions.

Scott Lane Williams 1:12 PM

as i'm trying to figure this all out, i'll PING you.

Marco van den Akker | vda.hosting 1:12 PM

you should check out Spotcloud.com

///////////// NEED ////////////////////

1) Monintoring via google analytics and/or other tools

2) Admin screens. Admin login, processes, etc.

3) List of cron jobs and other things that comprise the architecture.

4) Any special extra OS configurations or add-ons

5) Access to both the "web" application server and the "database" server

6) Database "root" access.

7) Any source control/versioning?

8) Any architecture documentation.

9) Any special setup for the AMFPHP software

10) Any PHP accelerators used?

11) Special configuration for sendmail? Did it use external mail sending service?

12) Which version of linux was/is it running?

13) What were the specs of the app server?: disk space, memory, processors, linux version, php version, etc?

14) What were the specs of the database server?: disk space, memory, processors, linux version, php version, etc?

15) Any special MySQL configurations?

16) Any database replication, redundancy, etc?