4/15/19

Hey guys, good morning! These are some small questions and/or topics for Wayne and Eric dealing with the transition over to AWS. I’m not looking for a huge or big answer on some of these topics, but I’d like for us to talk and chat about them (and others), just to help us all be on the same page. If we want to meet, I’m happy to fill in and/or scribe and write down some of the conversation and/or discussion notes. Just trying to get the ball rolling. Thanks guys and enjoy!

* Back-ups (daily, nightly, etc.)
  + Two different pieces. 1=files, 2=databases
  + Code back-up will be in git (code repository)
  + The database is somewhat spread across a cluster of database servers (Aurora cluster)
    - The cluster is backed up daily (daily snapshot)
    - We get to determine how long to keep those snapshots
    - We could go between S3 and glacier as well
    - We could get do a snapshot at any time
    - There is also a roll-back option of up to 24 hours – entire back-track
  + Images and other stored files
    - We could setup a 2nd bucket but currently we don’t really have a plan here
    - Maybe put in file policies – Amazon glacier (cold storage on magnetic drives)
      * Automatically copy from S3 (S3 = amazon’s simple storage technology) to glacier
      * As a note, it may be more money to pull from glacier storage
* Mirrors/Rollovers/Failovers
  + Built-in redundancy – part of the deal. Everything is built on a cluster type model.
  + The servers will build and then either change the load balance going both forwards or backwards
  + We will still have some slow queries, but the load balancer should really help speed things up and keep the traffic going
* Storing back-ups (say a two-week rolling back-up)
  + We will try to stick with a two-week rolling back-up on database files
  + We could also setup a back-up to glacier every month, every quarter, whatever
  + We could sell client specific services based on needs – as detailed as needed based on pricing
* CF (Lucee) resets
  + We are hoping that this will take care of itself due to load balancing
  + AWS will be cycling things in and out as they die and/or are needed
  + We can’t directly login to these boxes
  + The old way we did things is not possible
  + These will be docker images not actual boxes
  + Ideally, we should be able to take a docker image and test things locally
  + Lucee does have an administrator page – We do need to use the Bastion server to get to this.
  + In order to view the log files, we will need to go to cloudWatch – normal Amazon login vs a direct Lucee login
* DB resets
  + Table locks – need to check
  + Read only – need to check
  + Conflict – need to check
  + We can’t directly login to these boxes
* Logins
  + We will be using normal login to begin with (as few changes as possible)
    - We want to move over to AWS as we are and then change from there
  + Eventually, we will be doing a more global user id login
* Training
  + Ongoing
  + Still some unknowns – need to check and make a plan here
  + Multiple areas to this training…
    - We eventually want to change the whole thing around
    - Writing code
    - Automating tests
    - Object oriented
      * Alan was asking Wayne where we are going and how we may end up using some of the new Redis server stuff (in memory caching and storage). Lots of options on the horizon.
    - Model, View, Controller (MVC stuff)
    - Local testing environments
    - Docker images
    - AWS stuff
    - Lots of levels
  + This will be a paradigm shift for our entire company
* Server monitors
  + Used to use Newtek and Nagios – this will be replaced by CloudWatch
  + Who is going to be watching and monitoring these notifications?
    - Subscription based (we can switch who is doing what)
* Slow queries
  + CloudWatch logs and alerts
  + Because of load balancing, we should have less and less slow queries that are problems
  + These will still exists (code problems, indexes, etc.)
* Storing and processing images
  + Wayne is working on this
    - Uploads to a generic S3 bucket and then we call a Lambda function that moves and resizes the images to the correct corp-specific bucket
    - We may need to get some new corp-specific settings dealing with resizes
    - Wayne can tweak the Lambda functions as needed
      * Full high quality rescale and resample of the original
    - We may end up storing a certain size and then resizing things on the fly
    - Check the width >> thinking it is 760 pixels wide (width)
* Storing and processing media/content
  + Exact same as before for now
  + Eventually, we can even store these in the same corp-specific S3 buckets
  + No resizing of the data… what they upload, we hold and store for them
* Virus protection
  + There isn’t any – no physical machine to infect
  + The only possible place we may need it is on physical storage locations – most everything is an image and/or virtual cloud of sorts
  + Most of the pieces run fully independent and there is not a shared environment out in the cloud – a shared environment is similar to a home or laptop computer where everything is shared and physical (drives and memory)
  + Instances get created and die dynamically
  + As we build new instances, we will be running automated tests
* Shared DB’s or single DB’s per corp
  + Right now, bring things over just as it is (shared DB’s).
  + Once there, on AWS, we will be looking into breaking things into single DB’s
  + We will also be figuring out high level database tables (adilas admin) vs corp-specific tables and data.
  + There will be future projects in the data modeling and data area
  + We will end up building our own database management tools
* Any VPN (virtual private network) or VPC (virtual private cloud – private space at Amazon) needs
  + Bastion server – basically a limited VPC connection into some of the boxes
  + Nobody can get directly into the VPC (locked space)
* Static IP addresses
  + Each time we spin up a new VPC, we get a static IP address for that load balancer (new stack)
  + The domain names will map automatically
* FTP
  + No can do – this will be a big change for us
  + Everything needs to go through the code repository and code build process
    - Everything is an image or instance of what should be there
* Git/branches/bit bucket stuff
  + We currently use bit bucket for code storage and code repository stuff
  + Eventually, we may end up switching over to AWS stuff for the code repository stuff
  + Branches – we don’t know how this is going to look yet, but we will have some separate branches. Talking about features and versions. Still unknown and may take some time to get the full plan here.
* Running DB updates
  + We will need to do this manually at this point.
  + There may be future plans to automate some of this
  + We would love to have the application help and watch for hooks, events, and needs
  + On the code pipelines… we can even have it require manual events and/or switches (if needed)
* Global user Id’s
* Adilas Café or common area
* Nightly server resets
* Adding new corporation (process)
* What about corporations that are on data storage
  + We may want to look at a migration between S3 and glacier storage
* Turning sites off (inactive corporations)
* Timing of back-ups, server updates, and virus scans
* Versioning
* Auto processing
* Scheduled tasks
* Testing vs Live (production)
  + Thinking of a blue and green target group (AKA a color coded bucket set). Basically, one set of code (single branch) but then migrate between the two different target groups.
  + The URL won’t change, the code in the background will just switch to which set of Lucee servers are going to be used, meaning pointing to the blue target or pointing to the green target.
  + The terms blue and green don’t mean anything as far as one being better or worst than the other… it is just a color change. Today it may be pointed to blue and then later, in the same day, maybe we point it to green. Just a simple color based toggle switch.
* Local instances
* Tech support
* Documentation for our clients (what we have in place and what we can market – tell people)
* Hotfixes and emergency uploads
* Server access
* Disaster recovery
* RDP – remote desktop type access
* Changes, requests, and tech support tickets
* SSL’s
* Domain names
* Sub domain names
* Clusters (groups)
* Session storage
  + Redis servers
  + There are multiple options and the system will manage the traffic and load
* Linux boxes
* Windows boxes
* Load balancing
  + Taking people off certain servers and getting them setup on their own servers may end up being a revenue stream. This is somewhat part of load balancing but doesn’t really play into capacity.
  + One of our goals is to be able to charge clients based off of real usage and/or load (data processing, storage, images, files, media/content, etc.)
* What about dedicated servers (other stacks)?
  + Sell it as a VPC’s – we could have up to 100 of these (current limit – we could increase if needed)
  + Basically build a private adilas instance with their own parameters
    - Totally configurable on stats and options
      * More of x or less of y or whatever
    - We could even do this based on industry types (our shared environments)
  + Shared stacks (basic adilas stacks) vs dedicated stacks (dedicated cloud model for our clients)
* Firewalls and hardening servers
* Updates and maintenance
* Cross training (what happens if Wayne gets hurt and/or sick?)
* How do we report a problem and/or issue
* Email options and email servers
* Text services or other communications (future)
* Newtek and what stays there
* Future
* Testing plan
  + We will still test and develop locally
  + All local boxes will be running a docker image
  + They, Wayne and Eric and Alan, will be helping to write testing code to run on Jenkins servers.
  + They, the cool coders, are able to write tests that virtually simulate real-life scenarios with clicks, decisions, events, reports, output, etc. Pretty deep.
  + There will also be unit tests, integration tests, etc.
* Migration plan
  + We will be creating new stacks for each existing server
    - Example: data0.adilas.biz, data1.adilas.biz, etc.
  + We will be bringing in all of the old code, warts and all, and then going from there
  + Minimal number of changes in order to get it up to speed
* Duplication, pairing, extending (looking forward)
  + We can take quick snapshots of the DB at any time
* Billing for clients
* Custom code (black boxes)
* API sockets and access levels
* Timing plans and/or issues
* Name ideas for new clients (light marketing terms) – dynamic, dedicated, server cluster, environment, web services, universe, galaxy type terms, failover, rollover, whatever…
* Drawing and/or graphic of the layout (marketing plan) – big dumb animal type picture
* Security flyer, news and updates, video, etc.
* Local testing and development environment
  + The plan is to use a Docker image and keeping everything standardized as much as possible.
  + Wayne will help us get setup and going on that process
* Build as if for years
* Definitions:
  + Stack – complete instance of adilas running (load balancer, db, Lucee, S3 buckets, etc – full recipe – all pieces below here are part of a single stack)
  + Application Load balancer (cluster)
    - Receives all of the web requests (managing all traffic)
    - Allows for filtering at the global level
      * Frontend firewall
      * Any outside access – management
    - Redirects different URL’s to different instances and/or clusters
  + Aurora database (cluster of MySQL databases)
  + Redis (in memory database - cluster for session storage and also caching of queries, etc.)
    - It might be cool if Wayne could show us how to write and retrieve directly from this in memory server stuff
    - It has a management interface to help look-up things and what not
    - The old session stuff used to use lots of memory… now we can potentially add even more without feeling a blotting affect.
    - This in memory storage can also survive server reboots, if needed
  + Individual Lucee boxes (CF engine – cluster)
  + Lambda (standalone code not tied to a specific computer or server – functions in the cloud)
    - We could use all kinds of listeners, triggers, and events
    - We can also use any other language that we want
  + S3 buckets (think of separate hard drives – clusters – S3=amazon simple storage services)
  + ------ magic line ----- (all above deal with the stack – below are universal)
  + The Lambda and certain S3 buckets may straddle these lines as well but generally speaking, the stack stuff is above this line
  + Cognito (global or universal login)
  + Code commit (Amazon’s code repository plus some other options – run all automated tests, update our stuff when we make changes)
    - Somewhat of a backend and management tool
    - Code pipelines and pushing and testing code
      * Doing some testing using a Jenkins server
  + IAM – security piece (create users and system admin management – roles, users, groups)
    - Our access into the AWS console. It also runs all of the security between the pieces and parts.
  + CloudWatch – logs, notifiers, exceptions, etc.
    - You are able to subscribe and unsubscribe to these messages
    - Universal logging service
    - This could be event driven (future options)
    - You can even query the log files using a special Amazon tool for the log files
  + Docker Image – this is what builds the image (the recipe).
  + Pipeline – a list of rules to go through
    - If needed, we could add a manual setup and/or trigger of sorts
* Other…