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.
	+ URL from Wayne - <http://reinh.com/blog/2009/03/02/a-git-workflow-for-agile-teams.html>
	+ See the URL above for some good practices – looking for somewhat of a committee type decision
* 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
	+ What is done … the storage and/or database stuff is already done and is stored in the Amazon Cognito database – The Cognito service is an open id implementation that handles all of the single sign-on or single login
	+ Capable of emails, text messages, and phone calls to verify – we are leaning towards the email side of things
	+ The goal is to move the users outside of the main adilas environment
	+ URL from Wayne - <https://universe.adilas.biz/login?response_type=code&client_id=cpbp6o38rb8gd27phe4m65o7s&redirect_uri=https://new.adilas.biz/top_secret/oauth_login.cfm&scope=openid+email>
	+ The URL above has a main URL and a redirect URL mixed together
	+ There were some questions on what is the process to do this and how are we going to allow them to update things and even get linked to other corps? That might be multiple questions, all in one bigger question. Basically, we need to document the process and who gets what, how does someone invite, allow, deny, re-establish a link, etc. It needs more work.
	+ We also need a way to limit and/or figure out what role the existing payee table will play and how we manage that.
	+ We may need to setup a set of pages to help with the transition process – allow an old login and help to transfer them to the new system.
	+ We need to be able to copy payee info from a master list, to a sub or slave payee table.
	+ There was some initial talks about a cluster payee table. That would be somewhat of a master type list.
	+ On the old logins… we allowed users to skip the data 0 server and go directly to their own servers. We are hoping that the new AWS stuff helps us establish more of a direct path through a common doorway for all users.
	+ There are some security questions about how deep do we allow the open id being linked to someone else’s login credentials. Is there a time limit, do we record a history of who allowed what, etc.
	+ What about an auto adilas tech support person per corp (auto bridge)
	+ Questions about temporarily being able to use others permissions (logging in and being an alias user of sorts)
	+ Accountability of the user/payee id numbers and histories
	+ What about a generic guest account that may be used – This could an easy setup and/or copy button.
	+ Recording histories and full audit trail is huge
	+ Some of our clients may even want a page view history report based on who is looking at what
* Adilas Café or common area
	+ Login (super simple), choose which corps, news and updates, work wall (what corps – logo and name of company), play wall (what corps – logo and name of company), developer’s notebook, adilas user community, community funded projects, adilas market (who has skills and who wants to fill a role), training, adilas university, etc.
	+ Consolidated stats wall… wow, look at what I have done and/or can do
		- Possibly set certification levels based on number of successful events
		- Be able to sell their skills
	+ A job posting wall
	+ As a side note… work wall, play wall, stats wall, job posting wall, training wall, market wall (other services), community wall – multiple walls (hexagon, octagon, more…)
		- Alan would like to have some drag and drop options to help organize some of these pieces – I always do this, I really like to see this, I rarely do this, etc.
		- It would be cool if it were almost a 3D type room (just being silly)
		- It could also be cool if it were virtual – like they do on Marvel Avengers (open virtual touch)
	+ Being able to request access to someone else’s corporation
	+ Possible user forum
	+ Single sign-on with Facebook or some other major player login
	+ Allowing all and/or certain demo sites to general users – There may be a difference between a demo site and a play site
	+ Show how we can solve some of their issues
	+ We want everybody to pass through here (a quick pause) and then decide where to go from there – use the data sphere type analogy
	+ Keep this pretty fresh and fun
	+ Part of the adilas market could be a giant ecommerce type interface where all kinds of different adilas clients could sell their products and services (kinda like Amazon or ebay type environments).
* Nightly server resets
	+ We don’t think that these are going to be necessary
	+ We have redistributed a number of the in-memory needs to specific servers
	+ Hopefully the load balancer will help there as well
* Adding new corporation (process)
	+ Currently, we add one at data 0 and then go to the new server and add it there
	+ Wayne is thinking that we will create a Lambda function to create and/or virtually setup the new S3 buckets, new database instance, etc. Use an API socket call and automate the process
	+ Moving more to JSON scripts and automating the process
	+ We need to figure out exactly what pieces are needed per corp and then coding according to that needs list
* What about corporations that are on data storage
	+ We may want to look at a migration between S3 and glacier storage
	+ There are at least two pieces… stored files and the database. We could take a complete snapshot of the database and then send everything over to glacier until needed.
	+ There is a cost to pull things out of glacier
	+ Maybe leave it on the S3 buckets until they say shut it all the way down. At that point, we could move it to glacier.
* Turning sites off (inactive corporations)
	+ See notes above for data storage – the main difference is how likely is it to be pulled back up
* Timing of back-ups, server updates, and virus scans
	+ This will be way less than it used to be
	+ We will be using static Docker images. We will keep a back-up of those pieces.
	+ Server updates deal with changing the Docker images. This will keep rolling forward and we’ll add the updates as needed.
	+ Lots of virtual redundancy and built-in back-ups
* Versioning
	+ This is still being worked on and we have some ideas here – some of process will deal with how we use and deploy things from the Git repo (code repository).
	+ We will need a master branch (generic or neutral branch) – all other work is off of other branches
	+ Using feature branches, bug branches, etc.
		- We may want to completely redo our Git repository
		- Doing branches off of other branches
	+ We will standardize this process. At some point, we will need to recreate the entire Git repo structure.
	+ There is a URL link up higher that talks about some good Git best practices and such
	+ One of our huge key pieces that we are missing is our high-level project management. At some point, there may be some other pieces that come in as we grow and mature in our internal project management.
	+ We are somewhat using this transition over to AWS stuff as part of our transition into some of the places where we want to be. Things will keep changing and morphing as it rolls out.
	+ Anything stored in an S3 bucket has its own file versioning – what updated what at the file level
* Auto processing
	+ We still have a need to help update and sync data between servers for state compliance type records
* Scheduled tasks
	+ It depends on the task and what needs to be done
	+ Is this a ColdFusion or Lucee task or a database related task?
	+ We could use Lambda functions and/or tasks – even use different languages as needed
	+ We could have some of these outside Lambda functions run on different boxes that don’t interfere with normal user traffic
	+ We could schedule times, events, or on call API socket calls
	+ There was a question about cleaning up and clearing queue tasks and queue tables – some of the tables are getting huge and may need some clean-up and/or maintenance. Wayne was saying that we are virtually searching millions and millions of records every couple of minutes.
	+ Maybe rework what data is kept in the main table. We may create our own migration process between a temp table and a live table.
* 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.
	+ There will be two URL’s – one for testing and one for live – there may even be a third branch that allows for destructive type changes to the database. The third branch is still somewhat in question… not sure if we really need it.
	+ Wayne is thinking of doing a database dump and copy every night… that way we could do and run tests on real data. Basically, being able to test on current data (relatively).
* Local instances
	+ Moving from standalone ColdFusion servers and standalone MySQL database into a Docker image – full replica of the production environment
	+ The same unit tests that we use in live will be available for testing if needed
	+ You have to be able to run Docker for Windows, Max, Linux – at least 6 GB of RAM
* Tech support
	+ We don’t want a single pivot point… spread the load
		- We’ll do some cross-training and even some practice sessions
	+ Different levels
		- System level
		- Code level
		- Server level
	+ We will need someone to get up to speed on the AWS console
		- Be able to spin up stacks
		- Be able to get to the Amazon tech support console
		- Be able to be on call and even switch who is virtually on call
	+ We pay Amazon $100 a month and are able to create a case at any time
		- We can setup communication options
			* Phone call, chat, email, etc.
	+ We have Wayne, Eric, Alan, Brandon, Steve
		- We have other developers who could play in but we will start with the list above
		- There will be a number of alarms, alerts, logs, etc.
			* Those values will be logged and queued
			* We can setup subscriptions between services and figure out times when certain people
	+ We shouldn’t be talking individual servers… We need to focus on Amazon services
		- We can’t login to a specific server
		- Everything is built on a cluster or image type interface
	+ Two main issues…
		- Something is wrong in our code
			* This is one of our main focus points. We want to be fast and efficient with this.
			* Git, deployment, testing servers, live production servers, etc.
			* We need a roll-back plan as well
		- A service at Amazon isn’t working
* Documentation for our clients (what we have in place and what we can market – tell people)
	+ These are the x number of things we have in place… we then point them to the AWS site that talks about those features – let them (AWS) carry some of the weight there.
	+ We will build a pretty and simple page or flyer and then point them to AWS and their services per section.
* Hotfixes and emergency uploads
	+ The normal process is going through the code pipeline >> branches, commits, testing, live/production, etc.
		- The normal flow will roll forward
		- There are places where we could do manual interactions
	+ The theory is… because we are using the same scenarios in our local environment, we should be able to test and debug things before it gets all the way through the process
	+ This will still happen, but it should become less and less
	+ There was some discussions on testing and having the unit tests in place
	+ We are used to an instant FTP fix… the new process may take a minimum of 10-15 minutes. It takes time to rotate and rebuild those things
	+ Maybe we need to think about when we release new code and new features – timing of the releases
* Server access
	+ We have no direct access to these servers
	+ We can get to the Bastion server that lives in the VPC (virtual private cloud) – that is it
		- This server is not required for production
		- It is somewhat like an outpost to get us to a GUI
		- Putty connections to this server
	+ How secure is this access point?
		- You can only login with an SSH key/certificate
		- The servers have http, https, and SSH ports open – that is it
		- Once you get into that machine… it doesn’t have access to all other servers – everything is built somewhat on an island of sorts
* Disaster recovery
	+ All of the code to rebuild and recreate the current instance is backed up in multiple areas
	+ There is a way to copy things out to the Amazon Glacier
	+ All of our developers will have a Docker image with the full stack info
	+ For those clients that want their own back-up… We could setup an outside and/or additional service that creates full back-ups or data dumps and then stores those files somewhere or allow for user download – this could be its own revenue stream
		- We could do things on a schedule
		- Provide tools within the system to pull down their own data
		- Let the users get at their own data – exports, downloads, etc.
	+ EMP – electro magnetic pulse – Brief talks about options and reality of having that big of an event
	+ We may still do our own visual/digital (DVD and CD’s) back-ups
* RDP – remote desktop type access
	+ This doesn’t really exist
	+ Putty connection into the Bastion server
* Changes, requests, and tech support tickets
	+ There are two options…
		- Support tickets to adilas
		- Support tickets to AWS
	+ Most of the AWS stuff is all just params that get set to vary the number of options
	+ We have been super nimble… and able to make quick fixes… that is going to go away
	+ There are some fears in this transition – education and time are helping us to relieve some of those fears
	+ Some of the answers to this is writing good code and running some good tests to help us gain more confidence and do our due diligence
	+ All emails go to Wayne – company mandate… ☺
	+ On the adilas side, we could use that little gear on the side to allow open chat and/or contact support
		- Live chat could be recorded in elements of time and then even show date/time stamps and threads.
		- Maybe look at harvesting some of the existing tech support and making it available to the adilas community. This is an untapped reference. Tons of emails and outside docs that have been used to help teach others about the same issue and/or tech support question
		- Add videos, training options, link back to other threads and tech support entries (past entries).
	+ There were talks about needing an outside service where we could put updates that could be accessed outside of the adilas environment
* SSL’s
	+ Amazon has a Route 53 service that deals with SSL’s – they are free for those who run things over their services.
	+ We could save some money per year on the certificates (expenses)
	+ It auto renews
	+ Try to run everything through https
* Domain names
	+ Primary is adilas.biz
	+ Currently, we use Dotster.com to manage our domains. Amazon does have a service that helps to manage domain names. It is the Route 53 services.
	+ Super easy to point multiple domain names to the normal site
	+ We can also use programable functions to manage a few of these domain specific tasks.
	+ We will still allow for the customers to have their own domain name. This could be pointed to a normal shared stack and/or a full on private stack.
	+ There may be some tech support to help clients make record changes for their domain names.
* Sub domain names
	+ www – basic
	+ new –
	+ data0, data1, data2, etc.
		- stick with this as far as different stacks…
		- Almost using the sub domain as a pointer to different stacks
	+ Test, staging, etc. There will still be some forthcoming items here.
* Clusters (groups)
	+ We are calling these “stacks” or stacks of servers
	+ We will create an internal adilas control panel of sorts
		- Steve has a bunch of ideas of what he wants
		- Eric may be able to help with data warehouse stuff and collecting different stats
		- Alan could help with AJAX graphs, charts, etc.
* Session storage
	+ Redis servers
		- There are multiple options for session (in-memory storage) and the system will manage the traffic and load
		- We could expand this as needed. No longer tied to just plain RAM memory like we were
		- We could store JSON objects to help define some of the main system players and their properties
		- Wayne was talking about a virtual profile – Basically, a JSON object that has each client’s profile.
			* This could be look and feel, setting, naming conventions, all kinds of other system or world level settings
			* We could even do page settings, etc.
		- We could also cache certain pages (news and updates) and/or specific queries
		- We could use the Redis servers as an active cache of sorts
	+ No major code changes to start using the Redis servers.
* Linux boxes
	+ All of the Lucee boxes are Linux boxes
	+ All of the S3 buckets are Linux boxes
	+ The Alpine Linux boxes are case sensitive – we may run into some problems in our code
	+ Linux is more bare bones vs a Windows box that has so many things under one hood
		- Linux boxes are very minimal – enough to run the kernel and that’s about it
		- Docker images
		- Just enough code to get the task at hand done
	+ We can’t really login to these boxes – it all images and virtual Docker files
	+ From Brandon – he was scared of command line type interfaces, case sensitivity, and unknowns – only real computer nerds use Linux.
	+ If we keep the Linux boxes small, there is very little real estate for virus and hacking… it becomes less of a target.
* Windows boxes
	+ Temporary need for some of Calvin’s pieces – GPS services, PDF label services, other Windows based services
	+ We are planning on using some Amazon Lambda functions to run the Windows specific things and pieces.
	+ These are targeted (hack attempts)
	+ We are moving towards Linux boxes
	+ Local development environments will still be on Windows and Mac environments. They will all be running Docker images and thus simulating the Linux workflow
* Load balancing
	+ The main goal is evening out the load on the different boxes
	+ It could also do super high level validation on XSS (cross site scripting - hacking), SQL injections, etc. This is the top level resource allocator.
	+ This will force all confections to https
	+ Lot of other potential there as well
	+ 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.
	+ The load balancer does really do this, but 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.)
		- We will be able to monitor databases and S3 buckets to get some of these stats
		- Also, we will use Amazon CloudWatch to get stats, logs, 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
	+ Our first line of defense if all part of a VPC (virtual private cloud)
		- Almost all access is blocked to direct Internet traffic
		- It all gets processed though the VPC
	+ Security groups and kinda like our own firewall
	+ Wayne had and does have some hacking type tools that he can run against our servers. This process may be ongoing.
	+ Because of how things are setup… we are somewhat relying on Amazon to help support these pieces
	+ The actual attack surface (where someone could attack) are very minimal
	+ This may be a process that evolves over time
* Updates and maintenance
	+ Two main pieces
		- Updates to AWS stuff
			* We could toggle things on or off, we setup a maintenance window
		- Updates to our code
			* Going to be relying on the Git repo and code flow
	+ New Docker images – this will help us manage our upgrades
	+ Currently, we can modify each server as needed. Going forward, we can still keep the different stacks on different levels but ideally we get them all the same.
* Cross training (what happens if Wayne gets hurt and/or sick?)
	+ AWS has great tech support – good turn around time
	+ Getting users (our developers) working in their own local environments, that will help us know what is going on out in the real AWS land.
	+ Logs and subscribing to different channels
	+ Try to eliminate pinch spots, squeeze spots, and pivotal pieces
		- Build it so that we could step away
		- Map it out and make it doable for others
* How do we report a problem and/or issue
	+ For now, we will reach out and contact Wayne
	+ There may be some future options for official bug reporting processes and what not
	+ Keep using existing channels until we come up with a better and/or more complete plan
		- Emails, texts, phone calls, voicemail, GoToMeetings, Zoom, chat, Slack, etc.
* Email options and email servers
	+ Currently, we are using a shared email server that is located out at Newtek. We don’t have any admin rights out there, it is their shared server.
	+ Currently, we have to do an API call to send and/or generate any email communications
	+ We can’t send bulk emails, we can only do single emails
	+ We have minimal error handling
	+ We do have unlimited accounts, but we have a limited bandwidth/throughput
	+ At some point, we would love to add custom emails per corporation in order to get better results. Currently, we are using three or four main email addresses such as do\_not\_reply, sales\_receipt, message, etc.
	+ AWS has a simple messaging service.
		- We would be billed based off of usage
		- $2 per 100,000 email messages – possible mark-ups
		- This could be per corp
		- AWS has a service for mobile app push notifications as well – future options
* Text services or other communications (future)
	+ The AWS has an existing texting service
	+ $0.00645 per text – for the U.S.
	+ This may be built out later on
* Newtek and what stays there
	+ To start with… all pieces will be there until full transition takes place
	+ Our plan is to leave the content server and email servers there until we are ready
	+ All data servers will be migrated on a per server basis
		- As they come over, we may end up leaving them in place for two weeks to a month before getting rid of the older boxes.
	+ Our goal eventual goal is to have nothing critical over there
	+ We are not mad at them in any way. We are just upgrading to a more complete package. They have been very great to help and deal with. Good stuff.
* Future
	+ Coming and going to be exciting…
	+ There will be a lot to learn and lots of ways to grow…
* Testing plan
	+ Our goal is to have a 100% test coverage (that goal may be unattainable but still the goal)
	+ 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.
	+ Are built-in tests required for code sign-off?
		- We may determine this based off the different developers
		- Pie pieces and making sure that the whole still works and functions
	+ There may be certain tests that are customer mandated
		- We really want to work with the customers to help them get what they want
		- Some of this customer interaction may go all the way back to good project management
		- Test early and often
		- Getting customer sign-off
		- Test driven design
* 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
		- We may do some light clean-up – most of the old code will be untouched
		- Light Application setup, vars, https, sub domains, images, etc.
	+ Minimal number of changes in order to get it up to speed
	+ Testing the AWS stuff
	+ At some point, we kinda need an initial test group to just get in there and start pushing on things (live testing on AWS stuff)
* Duplication, pairing, extending (looking forward)
	+ We can take quick snapshots of the DB at any time
	+ Long term inactive storage – Amazon Glacier
	+ As far as pairing and joining… that is still out in front of us and somewhat of a new frontier
	+ Ongoing development – not sure how this will all look and/or play out – yet to be determined
* Billing for clients
	+ Currently we do a fixed price, we may have to grandfather some of the older clients in to the new model.
	+ All future clients, when ready, will be based off of storage, capacity, connections, usage, etc. We are headed in this direction.
	+ We may also be able to break things into smaller packages and have multiple add-ons that may be purchased and/or accessed separately.
		- Certain options, features, and components
		- Sizes, tiers, and what is being exposed (perceptions – mountain vs ice berg analogy)
		- This could be as simple as a single page that shows what permissions and/or options are available.
		- CRM, inventory, expense/receipts, payroll, eCommerce, File management, etc.
	+ Provide reports and stats
* Custom code (black boxes)
	+ We still really need this.
	+ Build custom code into the main or master branch
	+ Black box code gets really hard to update vs just simple core changes
	+ Build more towards settings, permissions, and configuration logic
	+ Try to make all of the servers as close to duplicated/cloned as possible
	+ We would like to re-look at and/or re-work how we do black box code – we still need it, we just need to look a little deeper into this.
	+ Go beyond corp-specific custom code… maybe theme, industry-specific, global settings (toggle switches), etc. Look at ways to be more efficient.
	+ Database type storage where we store certain lines of code and just feed it in to certain pages.
	+ There is a cost to being able to provide so much custom code being implemented
	+ Building out core functions (reusage code with settings) vs full on custom development
	+ Steve was asking questions about MVC (model view controller) type options
	+ Eric was talking about custom one-off’s and building more of a feature type interface where we could add things to a design framework.
	+ It would be super cool if we could keep building core pieces and then let them be consumed through API socket connections. Changing and making a paradigm shift on how things are built.
	+ Wayne brought in controlled testing and automated testing to help our confidence and shorten the development cycle.
	+ Be on the proactive side of this development (prepare for the future)
	+ Rewriting, refactoring, and building the system to be more resilient to change. Building with a long-term vision. Taking logic out of the pages and building them into the method. Plan on change.
		- Object oriented type approach
		- MVC – keep heading to the model, view, controller
		- Short-term costs vs long-term costs
		- Alan said that 80% of bugs happen from copy and paste
		- Automated testing will help us some of that
* API sockets and access levels
	+ Wayne sees the whole thing going to a full API socket level application. This could be internal API calls and/or external API calls.
	+ Almost a server less (not to a specific computer) – things live by themselves, but are still part of the bigger whole.
* 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…