THE CHANGING ROLE OF THE ENTERPRISE IT ARCHITECT IN THE AGE OF CLOUD

Stratoscale
March 2017
<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>INTRODUCTION</td>
</tr>
<tr>
<td>5</td>
<td>THE BIG PICTURE</td>
</tr>
<tr>
<td>6</td>
<td>METHODOLOGY</td>
</tr>
<tr>
<td>7</td>
<td>RESEARCH</td>
</tr>
<tr>
<td></td>
<td>Gathering Information</td>
</tr>
<tr>
<td></td>
<td>Discovery</td>
</tr>
<tr>
<td></td>
<td>Site Survey</td>
</tr>
<tr>
<td>8</td>
<td>ANALYSIS</td>
</tr>
<tr>
<td>9</td>
<td>IMPLEMENTATION</td>
</tr>
<tr>
<td></td>
<td>Choosing a Cloud Provider</td>
</tr>
<tr>
<td></td>
<td>Location</td>
</tr>
<tr>
<td></td>
<td>Data Transfer</td>
</tr>
<tr>
<td>10</td>
<td>IMPROVING PROCESSES</td>
</tr>
<tr>
<td>11</td>
<td>AUTOMATION</td>
</tr>
<tr>
<td>12</td>
<td>CLOUD (AWS) VPC NETWORK DESIGN</td>
</tr>
<tr>
<td>12</td>
<td>DISCOVERY</td>
</tr>
<tr>
<td>12</td>
<td>ANALYSIS</td>
</tr>
<tr>
<td></td>
<td>Connectivity and Latency</td>
</tr>
<tr>
<td></td>
<td>Deployment and Orchestration</td>
</tr>
<tr>
<td>13</td>
<td>IMPLEMENTATION</td>
</tr>
<tr>
<td>13</td>
<td>Toolset</td>
</tr>
<tr>
<td>14</td>
<td>Migration Process</td>
</tr>
<tr>
<td>14</td>
<td>FUTURE CONSIDERATIONS</td>
</tr>
<tr>
<td>14</td>
<td>Scaling to Multiple VPCs</td>
</tr>
<tr>
<td>15</td>
<td>Multi-Region Deployments</td>
</tr>
<tr>
<td>15</td>
<td>GETTING CARRIED AWAY</td>
</tr>
</tbody>
</table>
Everything in the cloud seems so simple today. Just order the resources you want on AWS and bring up any application you want. Or, so it would seem. However, there are many factors that developers or engineers do not typically take into account when moving to the cloud; most of the time, they do not even know these factors exist.

People often wonder how the role of a solutions/enterprise architect fits into the day-to-day work of an organization. Today, an infrastructure architect has so much more to do than just designing an application — the architect is viewed as the individual who has an overall knowledge of all aspects of the system. It is important to note that the architect does not need to have extreme in-depth knowledge of every line of code and communication path between every subsystem, but rather a good overall understanding of the entire system — with experience and expertise regarding how the system should run and be optimized for a production environment.

The architect should be the one with his finger on the pulse and be up to date on industry trends, as well as ground-breaking technologies both in the software realm and the underlying infrastructure. He also needs to understand where it makes sense to incorporate a new technology into the solution. The architect will also know when it makes sense to refactor some parts of the solution to reduce costs or optimize operational aspects.

More important, architects should know how much of a disruption this new technology will have on the current process culture. He needs to understand that not every new and shiny object is worth pursuing — sometimes the resulting overhead and churn are just too great. In addition, understanding of compliance and regulatory issues is also expected of an architect.
THE BIG PICTURE

Keeping up with technology is not a simple task. The market is moving at such a pace that the time within which solutions become no longer relevant is decreasing year by year. If at one point you would budget for your hardware to be refreshed once every five years, you now need to do so every three years. With virtualization, however, you can worry less about hardware and start concentrating on the applications running on your hardware. The continuous cycle of new technologies repeats itself every few years, and the cycle is getting shorter each and every time.

We are at a point in time where information is really easy to come by. You can basically find anything you want: from how to create a homemade rocket to how to build a multi-site NoSQL database to serve as the backend store for your high traffic applications. All you need is to pop a few words into Google (or the search engine of your choice) and find your answer.

Because it’s so easy, the architect has to rise to the challenge and not only know how to implement the selected solution, but to implement it properly, the right way for the business. He also needs to be prepared to take a stand when choosing a solution or a path, expressing why it will or will not work for the specific needs of the organization.

The architect will have integral knowledge of the applications used by you today, with a good overall big-picture view of how the business operates on a day-to-day basis. He will be in the position to understand how a technology shift in the market will affect the organization from a pure technology perspective, and no less significant, from a human resource and organizational process perspective as well.
Most architects follow a simple pattern of steps to help lead the organization through the stages of any transition, be it new software or a new technology. Here is an example of such a flow.
RESEARCH

The constant need for knowledge is an integral part of an architect’s day-to-day tasks. As an architect, you are expected to learn new things on a regular basis, including different technologies emerging in your field, or in a totally unrelated field in which the technology might be applied to a product. For example, a compression algorithm that was built for big data analysis might be applied to a totally different application in the video streaming world. The technology is out there. You need to be the one to understand where and how it can be applied to your solution.

GATHERING INFORMATION

Information is everywhere, and the architect should gather the necessary ideas and tips online from multiple locations. For example, you can spend some time scouting blog posts on a subject of interest, be they vendor-specific or vendor-agnostic. (Twitter, LinkedIn and Facebook can be good for finding relevant posts.)

DISCOVERY

Important decisions are not made on a whim. A great deal of thought goes into considering the pros, cons and implications of any important decision. Without actually having the proper insight into resources each component actually uses (CPU, RAM, I/O, etc.), it will be very difficult to draw up a full picture. It’s also critical to understand the characteristics of the usage, with regard to network behavior, I/O patterns and throughput and resilience. All of this should be constantly documented throughout the lifecycle of all your components and applications. But, as we all know too well, in the Agile world and in accordance with the Agile manifesto, documentation usually takes a back seat: “Working software over comprehensive documentation.”

Therefore, before embarking on a major change, due diligence needs to be performed.

SITE SURVEY

The metrics of all your applications should be collected over a period of time using the tools you currently have in-house. Most enterprise applications will have a historical record of their performance, their usage, and behavior over time. This can be leveraged to collect all the data needed to prepare a complete picture of your application’s behavior.
ANALYSIS

Data analysis is an art. It can be very daunting, but also extremely useful when looking for valuable information. It is important to look not only at averages, but also at the data at peak and low times. When going to the cloud, you will want to make use of the elasticity available in most cloud platforms, which will allow you to grow over time and not have to provision for peak from Day One. It is important to understand that even if you are going to start low, you need to take into account resources to handle both load and redundancy.

Let’s look at a quick example.

<table>
<thead>
<tr>
<th></th>
<th>Low/High</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU (MHz)</td>
<td>1,500 / 10,000</td>
<td>4,500</td>
</tr>
<tr>
<td>RAM (GB)</td>
<td>0.5 / 128</td>
<td>16</td>
</tr>
<tr>
<td>Network (Mb/s)</td>
<td>1 / 10,000</td>
<td>300</td>
</tr>
<tr>
<td>Disk (IOPS)</td>
<td>20 / 2000</td>
<td>150</td>
</tr>
</tbody>
</table>

The smallest unit of EC2 available today from AWS, a t2.nano instance with 0.5GB RAM and 1 vCPU, should suffice, but only for the bare minimum. An m1.large instance with 7.5GB RAM and 2 vCPUs should be more than enough to handle some of the load. However, there could be two problems. First, the RAM might be a bit low. And second, it would present an Achilles heel in your design in that a single instance can (and will) fail, which could render your service useless and cause an outage.

Planning for multiple scenarios is the key. Although you cannot plan for every edge case, you should expect the unexpected. A solution design should be like an onion. It should comprise layers upon layers upon layers, with multiple paths available to provide the functionality you need, and allowing for failures along the way, but never crippling the system outright.
IMPLEMENTATION

After collecting the requirements, the data, it is time to plan for actual “getting-things-done.” The architect should help plan each and every phase of the migration. Planning for implementation might include, among other things, the following considerations.

CHOOSING A CLOUD PROVIDER

Depending on the application needs, some cloud providers might be a better fit for the project than others. For example, for a Windows-centric application, Azure might be more suitable. For a Linux-based application, maybe Google Compute Engine or AWS would be a better fit. The requirements will stem from the analysis of the data collected in the previous steps.

LOCATION

Does the application need to be in close proximity to a geographical location? If so, not all cloud providers have datacenters and sites across all regions. Therefore, latency could become an issue.

DATA TRANSFER

When migrating a solution from one platform to another, a significant amount of data might need to be moved. The considerations here could include: When should it be moved — close to the cutover point, or as early as possible and then to ship smaller increments? Do you want continuous replication between the old and the new? Of course, there are cost considerations. An optimal strategy should be chosen to minimize the cost. Does cost take the back seat because you need a certain level of performance? Do you choose synchronous or asynchronous replication to ensure that data is transferred in a timely fashion? What about data corruption and validation? How can you make sure that all of the data is valid? These are all questions that need to be answered.
IMPROVING PROCESSES

After collecting the requirements, the data, it is time to plan for actual “getting-things-done.” The architect should help plan each and every phase of the migration. Planning for implementation might include, among other things, the following considerations.

AUTOMATION

Standards and procedures are something that every enterprise needs to scale, function, and stay compliant. Some of the procedures can be manual, some in the form of a checklist, and others are simply knowledge passed down throughout the ranks during many years of work and collaboration.

When an organization makes a move towards the cloud, the points above are even more relevant; but there is one very important aspect that should be a guiding factor when you move to the cloud. The sheer scale with which you can and will operate in the cloud requires you to change your mindset and how you work. You will need to strive to automate anything and everything you do. Manual processes are not sustainable when operating at scale.

As an architect, you should take the time to understand the tools available to you today. Do these tools fit in with your current practices? If not, is there such a tool or perhaps the practices themselves need to be changed and to evolve. It does not really matter which tool you choose, be it Chef, Ansible, Puppet, Capistrano, Salt or perhaps a mix these combined with homegrown scripts and tools your company has collected over the years.

Now that we have gone through the steps and the process, let’s see a few practical examples of how these can be implemented.
Deploying your applications in a VPC requires planning. You are designing a whole new network topology in the cloud. This requires a substantial amount of advance planning, primarily because when you deploy a VPC, you define the CIDR block — and this is something that cannot change.

Here we have a basic design of an AWS VPC with one public subnet and one private subnet. This is a great start, however it has a number of fundamental flaws. The most elementary principle of design is to avoid having a single point of failure, using only a single availability zone for your deployments exposes you to a huge risk.

We will consider the VPC network design scenario based on the stages described above.
DISCOVERY

As part of discovery, you need to go through your applications and understand what is being deployed in an active-active or active-standby mode. The deliverable for such a task would be a list of servers in your product, and how they are deployed.

ANALYSIS

Now that you have a list of all the pieces of the puzzle, you need to understand how a change in the networking topology might affect your applications. First, it is important to understand the potential implications of splitting things over multiple networks. The most obvious are network connectivity and latency.

CONNECTIVITY AND LATENCY

In AWS, latency should not be much of an issue; it is actually quite simple. Each Availability Zone is connected to all the others in the same region by a high-speed network, which could be considered almost like a LAN link between subnets.

Regarding connectivity, as long as your applications are not using any esoteric protocols or communication mechanisms, and you can route over a network, you will not have any problems here either.

DEPLOYMENT AND ORCHESTRATION

Ensure that your deployment tools can cope with the fact that your applications are going to be on different networks. Ideally, this would be provided as a variable to your deployment scripts, allowing you to iterate over a list of networks to deploy your instances.

Another point that you should consider is the fact that if you’re using a custom solution for redundancy and using floating IP addresses between instances, this mechanism will work across different networks and availability zones. If not, you might have to invest some time researching a more common and standard solution that will work across multiple availability zones. This could perhaps be a problem with applications that work in a master-slave fashion, but your mileage will vary.

A design to provide a solution to the problem of a single Availability Zone would look something like this:
IMPLEMENTATION

Some pieces of the implementation will be carried out manually. There are others that will be beneficial to automate from start to finish.

For example, automating the VPC network creation might be something you will do manually, depending on the scale at which you operate. If this is something you will do only once, then investing a good number of hours on getting the automation correct from start to finish does not make that much sense. On the other hand, if you are going to repeat this process a number of times – across different regions, clouds or providers – the investment in automation will definitely pay off in the long run.

TOOLSET

Are your current tools adequate to do the job? If not, then it might be time to invest in some new tools, or at least improve your current ones: improve your automation, improve your code quality, or optimize the way you working. This migration could be a catalyst and opportunity for not only achieving a higher degree nor resilience, but also for improving processes and the way you work.
MIGRATION PROCESS

You will need to come up with a plan for how you will migrate your current applications without losing functionality and causing downtime.

There are a number of different methodologies that enable you to do this, including parallel deployments, staggered upgrades, and lift and shift deployments, to name just a few. Your migration plan will take into account any and all considerations that might impact on your users. There is always a balance that needs to be achieved between cost, time, and resources (human and infrastructure). It’s not always possible to duplicate your current infrastructure, due to any number of constraints, such as keeping databases in sync between different deployments.

It is not only the methodology that matters, but also the strategic thinking that should happen (which ties very well into the next point). An architect should not only focus on the task at hand – the immediate problem – but also understand how this could affect your solutions and strategy in the future. For the example above, it could take into consideration what will happen if you were to mirror this change at scale, or how to scale the solution to multiple cloud providers.

FUTURE CONSIDERATIONS

When making an architecture change, it makes perfect sense to not only plan for the near term, but also for the foreseeable future. It’s difficult to quantify how far down the road that might be because technology is moving at an unbelievable pace. It’s the architect’s job to determine how far down ahead to look at any given time, how to “future-proof” the solution, understanding how far ahead you should plan.

Here are a number of factors to consider as part of the design phase.

SCALING TO MULTIPLE VPCS

You might want to deploy in different VPCs for a number of reasons. It could be that you want to keep different environments – such as test, staging, and production – separate for compliance reasons, or simply because it is a best practice from an architecture perspective.

This raises the question of how to manage multiple VPCs with respect to pairing multiple VPCs, managing multiple IP address spaces and routes, and sharing resources between all the VPCs. You might choose to adopt the model of a common VPC for shared resources, duplicate some resources in each VPC, or adapt some hybrid solution.
MULTI-REGION DEPLOYMENTS

The whole trigger for this architectural change was the goal of eliminating a single point of failure, and each Availability Zone is in a different physical location in a region. Nonetheless, as an architect you want to plan for the worst-case scenario. Experience has shown that there have been failures in cloud providers that have rendered an entire region useless. For this reason, you might want to start planning for how you will expand beyond a single region. A significant number of additional implications will need to be taken into account. For example, latency and connectivity will be an issue with completely different geographical regions. How will these regions be connected? With a VPN to secure traffic between environments? Will the cloud provider’s VPN solution be sufficient or will a third-party VPN be required to meet your needs?

Resources such as security groups, AMI images, and S3 buckets will need to be replicated, as well — and synchronized. Some of them can be solved by the cloud provider with their built-in solutions, but others will need to be solved with either processes, automation or additional tooling.

GETTING CARRIED AWAY

As it goes with projects such as this, there is a chance you might go into too much detail — pile on a whole different list of technologies, improvements or nice-to-haves into the same process and project. An architect should keep the endgame in mind – what is actually needed and how to get there – without losing sight of the actual goal of providing additional resilience to the system and removing a single point of failure (in the form of a single Availability Zone).

There are certain elements that can be assimilated as part of this migration with minimal to no additional overhead or cost; when this is the case, the benefit to the organization is obvious. But certain elements can and should be shelved for a future version or iteration of the project, or split into new work items of their own. As an architect, keeping both yourself and the organization honest is your responsibility. By keeping the teams and the organization on track and true to your goal, the architect becomes a trusted advisor and core player in determining the technological direction of the company. This should lead to a goal of involving the architect early in the project to provide guidance both in the design phase and throughout the implementation.

Let’s consider another scenario for which the architect can be a driver of change.
Many organizations assume that moving to the cloud will drastically reduce their IT expenditure. This might seem to be the case while ramping up their development and growth on the cloud. However, there might come a time when you receive a call from the CFO asking why the cloud provider’s bill is so extravagantly high. Of course, this is a question of perspective, and whether the revenue generated from the use of the cloud dwarfs the costs incurred as a result.

But you will still be called on to reduce the costs for the company, and you should be prepared to offer a strategic plan for doing so.
When using a cloud provider, one of the first things you should have set up is a way to track what you spend on each instance. Most cloud provider solutions today have built-in tools that enable you to measure exactly how much you have been spending. They will even go so far as to provide tools (such as AWS Cost Optimization) that go into the details regarding ways for you to reduce some of your costs. You should use these tools to get a full picture of what is driving your cloud bill up, and if you find that they are not sufficient, use other tools, including the ones that you have developed in-house. These tools could be third-party solutions or perhaps home-made scripts that you have written over the years.

Considering the information above, you should be able to make some educated decisions about how to start saving money. Here are some factors that might be behind your high bill.

**CHANGING INSTANCE TYPES**

High cost might simply be the result of oversizing your deployment; you are wasting money because the resources are greater than needed.

**SCALING**

When you designed your solution, you designed it for peak usage. Implementing a simple scaling mechanism (assuming your application will support it) could drastically reduce your costs, and you will only use exactly what you need.

**XAAS**

You could possibly save money by making use of service that is given natively by your cloud provider instead of rolling your own. Available services include DBaaS, MQaaS, Object Storage, Monitoring, and LBaaS, to name a few.

**SERVERLESS**

Some of your applications are sitting idle 22 hours a day. They perform a job once a day, for a short period of time. You could be wasting resources throughout the day. Of course, not everything can be pushed out to a function that is run on demand, and significant amount of refactoring would be required before you can make use of a service such as Lambda.
IMPLEMENTATION

After having done your homework and research, you will decide on your plan of action, including timelines, milestones, and deliverables. Some solutions will be simple, and would entail a simple reprovisioning of the application — with the right sized resources. Other solutions could be more complicated. For example, if you have identified that using DBaaS will significantly reduce your costs, but the way you currently use your self-provisioned solution is not compatible with the cloud provider’s service. This will require some code changes and redesign of the product.

Perhaps, moving over to the provider’s LBaaS product will require you to change the logic in the way you route traffic within your applications, but once complete will significantly reduce the cost of running and operating your own solution.

IMPROVING THE PROCESS

Here the architect needs to look into tools (either available directly from the cloud provider, a third party or something that you would develop in house) that would enable you to not only make this a one-time occurrence, but an ongoing process that happens on a regular basis; to help you continuously reduce cost and optimize your environment on a regular basis.
There is a new wonder child in the technology world today. Containers as a technology have been around for a good number of years; however, their adoption became mainstream only in the past 18-24 months, with Docker driving this trend.

The architect should be the one to drive the organization’s journey to a containerized world. Such a journey would comprise a number of standard stages and might resemble the one described herein. Today an architect has so much more to do than just designing an application: the architect is viewed as the individual who has an overall knowledge of all aspects of the system. Again, it’s not necessary for the architect to know every single line of code, but rather a good overall understanding of the entire system — with the experience and expertise regarding how the system should run in a production environment.

One last note: An invaluable resource for you to gather and collect information for any and all of the above stages is technology conferences that focus intensely on a specific technology you would like to dive into. Each of the major technologies or vendors has at least one conference per year, such as DockerCon, KubeCon, AWS re:Invent and HashiConf. The natural continuation to the connections made and lessons learned from each of these in-person events would be the chat channels that each community uses. These differ from one group of developers and from one project to another: IRC, a Slack channel, and Gitter are just a few where you can ask questions, listen in, or scroll through a history of the logs where someone might already have asked your exact question.
Staying true to the Agile methodology — this is how you might describe a (fictional) user story for your architect for the next sprint.

"So the question is, what impact does this have on the position of DBA, and what do DBAs need to do to stay need to stay relevant in this rapidly changing industry?"

The modern infrastructure architect has to wear many different hats or juggle multiple roles within the organization. He is the one with his finger on the pulse of the technology used within the company, the direction in which business is going (in the short and long term), and in which direction the market (outside of the company) is headed. The insight and knowledge that an architect provides can be essential in taking the organization to a whole new level.
Stratoscale is the cloud infrastructure company, providing comprehensive cloud infrastructure software solutions for service providers, enterprise IT and development teams. The company’s comprehensive cloud data center software, Stratoscale Symphony, can be deployed in minutes on commodity x86 servers, providing an Amazon Web Services (AWS) experience with the ability to augment aging VMware infrastructure. Stratoscale was named a "Cool Vendor in Servers and Virtualization" by Gartner and is backed by over $70M from leading investors including: Battery Ventures, Bessemer Venture Partners, Cisco, Intel, Qualcomm Ventures, SanDisk and Leslie Ventures.

For more information, visit:
http://www.stratoscale.com

US Phone: +1 877 420-3244 | Email: sales@stratoscale.com