How to Optimize Your AWS Resource Allocation

By Mick England, DevOps professional in the Boston area

Controlling AWS costs is a pain point for many businesses. The process can be tricky and is often complicated. To help users out, Amazon published a paper in October of last year titled AWS Well-Architected Framework containing best practices for systems architecture in AWS cloud. This paper contains a wealth of information for anyone involved in or planning a cloud migration. It addresses the “four pillars of a well-architected framework”; Security, Reliability, Performance Efficiency, and Cost Optimization. The last of these is often overlooked until spiraling costs get out of control.

The topic of cost optimization is the subject of another AWS paper published in February 2016. Titled Cost Optimization with AWS, that paper states that “A cost-optimized system allows you to pay the best price possible while still achieving your business objectives and meeting or exceeding requirements for other pillars in the framework.”

Along with covering best practices and architecture, the Cost Optimization with AWS details a number of tools available to make managing costs easier. First, Amazon’s own service, CloudWatch, can monitor your estimated AWS spend and send alerts based on criteria you provide. Amazon also has an AWS Price List API through which customers can query the prices of services and receive notifications when they change and Detailed Billing Reports include hourly line items. Lastly, Amazon provides the AWS Trusted Advisor which will analyze your account and identify areas where savings can be made (such as identifying unattached EBS volumes).

While these tools help, managing AWS spend is still a significantly complex task. So much so that it has given rise to a whole new class of third party software companies with the sole purpose of managing and monitoring AWS billing. Additionally, modern monitoring tools, like LogicMonitor, have built product capabilities to hook into AWS APIs and present cost information alongside resource utilization.

These tools can be a great help in bringing cloud costs under control, and by following a few best practices it is possible to avoid unexpected costs in the first place.

Amazon suggests users think about these four areas for cost optimization:

- Matched supply and demand
- Cost-effective resources
- Expenditure awareness
- Optimizing over time

I won’t address all of the areas Amazon suggests in this post, but I will focus on some aspects that are not only best practices from a spend management perspective, but will prove essential for a successful cloud migration.
By Mick England, DevOps professional in the Boston area

IaC, Agility, and DevOps

So what’s the relationship between IaC and keeping AWS costs under control? Well it relates to one of the main driving forces for businesses moving to the cloud – agility. One of the key factors in success, especially for startups, is the ability to deliver new features to customers quickly and before the competition. Since as early as 2001 the desire of businesses to grow quickly has been a key driving force for agile software development practices. This same desire has led to the phenomenal growth in the adoption of DevOps culture and practices. Agile DevOps practices can certainly increase the speed of software development, but prior to the emergence of public clouds companies still faced the physical limitations of data centers and hardware. No matter how fast code was developed, anything requiring new resources would be blocked by this limitation.

AWS solves the resource problem, but in doing so it poses new challenges. When migrating from data centers to the cloud it makes perfect sense to extend DevOps culture to the provisioning of infrastructure. If your operations and development teams are already working closely together on things such as configuration management, why not allow developers to manage their own resources? Many companies begin their cloud journey by creating AWS console accounts for developers to “let them play” with AWS services. In my experience this is a big mistake and can lead to spend management nightmares further down the road.

Many other tools exist for orchestrating AWS resources as code. Puppet and Chef, which started out as configuration management tools are now commonly used for provisioning. Ansible’s push model made it an obvious choice for cloud provisioning and it includes an extensive list of cloud modules. In 2014 Mitchel Hashimoto, the founder of Hashicorp and the creator of a number of tools integral to DevOps such as Vagrant and Packer, released Terraform, which is described as “a tool for building, changing, and versioning infrastructure safely and efficiently.”

IaC, Agility, and DevOps

Infrastructure as Code (IaC), as the name implies, is the practice of defining your infrastructure as code. The concept gained popularity with the emergence of public cloud offerings from Amazon and others, as well as with the emergence of Infrastructure as a Service or IaaS. But IaaS is not the same thing as IaC. IaS refers to online services that abstract the details of infrastructure, (such as physical computing resources, servers, networks etc., for the user). Amazon’s EC2 Console is an IaaS tool for provisioning cloud resources, but this provisioning through the portal can still be an entirely manual process expressed in mouse clicks rather than code.

Amazon’s IaC tool is called CloudFormation. This is a JSON based system that gives developers and systems administrators a way to create and manage collections of AWS resources. The use of templates can allow for repeatable processes. Additionally, CloudFormation will automatically take care of dependencies such as an ensuring that an Elastic Load Balancer exists before attempting to add an EC2 instance to it. CloudFormation has an API, so it can be extended and used with any language you choose.

Automation – Infrastructure as Code

Infrastructure as a Service or IaaS. But IaaS is not the same thing as IaC. Infrastructure as Code (IaC), as the name implies, is the practice of defining your infrastructure as code. The concept gained popularity with the emergence of public cloud offerings from Amazon and others, as well as with the emergence of Infrastructure as a Service or IaaS. But IaaS is not the same thing as IaC. IaS refers to online services that abstract the details of infrastructure, (such as physical computing resources, servers, networks etc., for the user). Amazon’s EC2 Console is an IaaS tool for provisioning cloud resources, but this provisioning through the portal can still be an entirely manual process expressed in mouse clicks rather than code.

Amazon’s IaC tool is called CloudFormation. This is a JSON based system that gives developers and systems administrators a way to create and manage collections of AWS resources. The use of templates can allow for repeatable processes. Additionally, CloudFormation will automatically take care of dependencies such as an ensuring that an Elastic Load Balancer exists before attempting to add an EC2 instance to it. CloudFormation has an API, so it can be extended and used with any language you choose.

Many other tools exist for orchestrating AWS resources as code. Puppet and Chef, which started out as configuration management tools are now commonly used for provisioning. Ansible’s push model made it an obvious choice for cloud provisioning and it includes an extensive list of cloud modules. In 2014 Mitchel Hashimoto, the founder of Hashicorp and the creator of a number of tools integral to DevOps such as Vagrant and Packer, released Terraform, which is described as “a tool for building, changing, and versioning infrastructure safely and efficiently.”

LogicMonitor.com
How to Optimize Your AWS Resource Allocation

By Mick England, DevOps professional in the Boston area

A far better approach is to spend the time developing or implementing provisioning tools that will allow developers the freedom they want, while maintaining a controlled environment in the sense of when and how resources are created and just as important, how and when they are stopped or destroyed.

**Matched Supply and Demand**

Important decisions, such as what type of EC2 instance or storage to use, cannot be ad hoc. As the documents cited above stress, one of the most important factors in keeping costs under control is to match supply and demand. You need to ensure you have enough infrastructure to run services without wasting money. This is what Amazon refers to as a cost-optimized infrastructure. Determining just what is “enough” however is a significant challenge that will need to go through many iterations before we have it right.

**Cost Effective Resources**

The second area cited by Amazon is “Cost-effective resources”. Developing cost effective resources requires a great deal of familiarity with AWS offerings, and keeping up with these can be a daunting task. If you do not have AWS experts in-house then partnering with one of the many AWS consulting services can make sense. Its essential to learn to identify when to use a service such as Dynamo DB rather than installing your own Mongo cluster. The knowledge behind these decisions can have a big impact on spend.

Even with adequate knowledge of AWS services, not every decision you make on day one will be a good one from the standpoint of cost-effective resources. The agility AWS provides makes it relatively painless to switch out architectures as new services become available or if your systems are not behaving the way you expected. However, if you correctly implemented your infrastructure as code, then changing it is even more painless.

If you follow best practices for scaling and take advantage of Elastic Load Balancers and AutoScaling groups, then you don’t need to worry too much about under sizing your infrastructure. Scaling out is simple and can be automatic. Of course you should still try to be as accurate as possible in the initial decisions regarding your infrastructure.

**Testing and Monitoring**

Unlike in a data center, it is relatively easy to test out different scenarios in AWS. If you have been following IT Operations best practices, you should have a good idea of how your applications perform on the hardware in your data centers and where potential resource bottlenecks may occur. You will know if a particular application is memory or CPU bound for example. In contrast, mapping AWS EC2 instances to physical hardware can be difficult. Amazon’s TOC Calculator can assist with this but how a particular instance size performs will differ from one application to another. The only way to really know what works for you is testing and monitoring.

Begin by ensuring you have accurate benchmarks of your production infrastructure. Pay particular attention to key metrics such as CPU usage per core, memory usage, and network saturation. Test with different instance sizes and different numbers of instances. A given workload may perform better on lots of smaller instances versus fewer large ones.
How to Optimize Your AWS Resource Allocation

By Mick England, DevOps professional in the Boston area

If you are planning to utilize a service, such as Dynamo DB to replace Mongo, make sure you have an accurate picture of how Mongo is performing and benchmark Dynamo against this for the same workloads.

Once you have defined your new infrastructure, be sure to implement it as code so it can be easily repeated and changed as necessary.

Monitor real production traffic, and use tools like LogicMonitor to find wasted resources such as unattached EBS volumes and idle EC2 instances. Use your monitoring solution to detect performance problems and gather metrics on system usage. Time frames are important, you should be monitoring the percentage of resources used on an hourly basis as well as system usage for given times of day.

For example, if your development environment is rarely used between 5pm and 8am, shut it down. Automate infrastructure management and communicate when systems will be unavailable. Exceptions can always be made and there should be an easy way to do this in your infrastructure code and management tools.

Expenditure Awareness

We can not keep costs under control if we are not aware of what resources we are spending on. Use AWS detailed billing or a monitoring tool like LogicMonitor to analyze your infrastructure from a cost perspective. Compare costs both of entire Autoscaling groups and the individual instances within them. Ideally, your monitoring solution will allow you to compare costs alongside resource usage graphs. With LogicMonitor, you can create a dashboard to display this information:
How to Optimize Your AWS Resource Allocation

By Mick England, DevOps professional in the Boston area

AWS expenditure awareness needs to be part of your culture. Everyone should clearly understand that the availability of effectively unlimited resources comes at a price (this is particularly helpful if you are doing things like allowing developers to manage their own resources). You can also help by setting controls and using conservative default limits that come with AWS accounts to prevent accidental spend. For EC2 instances, this varies by instance type. The t2.small instance type for example, has a default limit of 20, whereas a m4.4xlarge has a default of 10. In almost all cloud deployments you will have reason to increase these limits, so be conservative in the changes you make. Don’t request an increase to 1000 just in case you need it.

When it comes to expenditure awareness, one of the most useful AWS features is tagging. Tags are defined as key value pairs that can contain letters, numbers, white space and certain special characters. Detailed billing reports can be customized to include tags that you define. AWS assigned tags have the prefix “aws”, while user defined tags are prefixed with “user” in the cost allocation report, providing a good initial filter.

Tags should be used to identify resources in ways that correlate with your business and teams working with AWS. For example, if an engineering team is working on multiple products, it makes sense to define tags that can be used to identify all the resources associated with that product and establish costs. Another good use of tags is to identify how much you are spending in development versus production. Monitoring tools such as LogicMonitor will collect and display AWS costs per tag, service or region, enabling you to setup automated cost reports that get sent to the right people per team.

In the AWS world, costs should not just be a matter for the owner of the IT budget. If you have a science team responsible for big data processing for example, make sure that team is aware of the costs associated with how much data is kept or processed. This can help in defining realistic data retention policies and using the right AWS storage system in association with that.

**Optimizing Over Time**

The final area AWS suggests focusing on is optimizing over time. It’s pretty much impossible to get it correct out of the gate, so it’s necessary to consistently monitor resource usage against performance. Going back to the concept of tags, in addition to being indispensable for cost management, tags can also be very useful to your infrastructure management tools and code. Say for example, Amazon releases a new instance type that looks like it will be perfect for your application servers for a given product. With good tagging you can easily automate switching out the instance type for that group of servers within your infrastructure code.

Depending how you implemented your cloud migration, there maybe big cost optimizations that can be achieved through addressing technical debt. Almost always, a so-called “lift and shift” approach will require more resources than an application designed and built, to run on AWS. That being said, not many businesses have the luxury of completely rewriting an application prior to a cloud migration. There maybe some low hanging fruit that can be easily moved over to an AWS service. RabbitMQ can easily be replaced with SQS for example. But there will certainly be many cases in which a monolithic application can not be broken apart and will be deployed to AWS in a less than optimal way. For those cases, addressing technical debt and architecting for the cloud can lead to significant cost savings.
How to Optimize Your AWS Resource Allocation

By Mick England, DevOps professional in the Boston area

Conclusion

The ease of use and scalability of AWS makes it incredibly easy to build out new infrastructure or experiment with new services but also causes difficulties in managing spend. If you are using AWS it’s imperative to have good controls and cost tracking in place. AWS provides guidelines for how to bring costs under control, but you should follow the best practices listed above to ensure that spend doesn’t get out of control in the first place.

Remember to match supply and demand when it comes to AWS resources by ensuring you have enough infrastructure to run services without wasting money. Expenditure awareness needs to be built into your team culture and optimizing AWS spend needs is the responsibility of all teams using AWS and whoever manages the IT budget also police AWS spend.

Use the agility provided by the cloud to automate the spinning up and down of devices but make sure you are also removing the devices you are not actively using. Make sure you have a performance monitoring solution like LogicMonitor that will help you find wasted resources and test to see how your infrastructure performs with certain resources. It’s also important to stay organized by using tagging features in billing reports. Don’t worry if you don’t have spend optimized out of the gate, but do make a conscious effort to optimize spend over time.

This content is brought to you by LogicMonitor, the automated SaaS performance monitoring platform that provides IT Ops teams with end-to-end visibility and actionable metrics to manage today’s sophisticated on-premise, hybrid, and cloud infrastructures. Find out more information about LogicMonitor’s AWS monitoring here or sign up for a free 14 day trial today.