Automating Elasticity

AWS Whitepaper

Automating Elasticity: AWS Whitepaper

Table of Contents

Automating Elasticity

Publication date: **March 2018** (*Document Details* (p. 8))

Abstract

This is the sixth in a series of whitepapers designed to support your cloud journey. This paper seeks to empower you to maximize value from your investments, improve forecasting accuracy and cost predictability, create a culture of ownership and cost transparency, and continuously measure your optimization status.

This paper discusses how you can automate elasticity to get the most value out of your AWS resources and optimize costs.

Introduction

In the traditional data center-based model of IT, once infrastructure is deployed, it typically runs whether it is needed or not, and all the capacity is paid for, regardless of how much it gets used. In the cloud, resources are elastic, meaning they can instantly grow or shrink to match the requirements of a specific application.

Elasticity allows you to match the supply of resources—which cost money—to demand. Because cloud resources are paid for based on usage, matching needs to utilization is critical for cost optimization. Demand includes both external usage, such as the number of customers who visit a website over a given period, and internal usage, such as an application team using development and test environments.

There are two basic types of elasticity: time-based and volume-based. Time-based elasticity means turning off resources when they are not being used, such as a development environment that is needed only during business hours. Volume-based elasticity means matching scale to the intensity of demand, whether that's compute cores, storage sizes, or throughput.

By combining monitoring, tagging, and automation, you can get the most value out of your AWS resources and optimize costs.

Monitoring AWS Service Usage and Costs

There are a couple of tools that you can use to monitor your service usage and costs to identify opportunities to use elasticity.

The Cost Optimization Monitor can help you generate reports that provide insight into service usage and costs as you deploy and operate cloud architecture. They include detailed billing reports, which you can access in the AWS Billing and Cost Management console. These reports provide estimated costs that you can break down in different ways (by period, account, resource, or custom resource tags) to help monitor and forecast monthly charges. You can analyze this information to optimize your infrastructure and maximize your return on investment using elasticity.

Cost Explorer is another free tool that you can use to view your costs and find ways to take advantage of elasticity. You can view data up to the last 13 months, forecast how much you are likely to spend for the next 3 months, and get recommendations on what Reserved Instances to purchase. You can also use Cost Explorer to see patterns in how much you spend on AWS resources over time, identify areas that need further inquiry, and see trends that can help you understand your costs. In addition, you can specify time ranges for the data as well as view time data by day or by month.

Tagging Resources

Tagging resources gives you visibility and control over cloud IT costs down to seconds and pennies, by team and application. Tagging lets you assign custom metadata to instances, images, and other resources. For example, you can categorize resources by owner, purpose, or environment, which helps you organize them and assign cost accountability.

When resources are accurately tagged, automation tools can identify key characteristics of those resources needed to manage elasticity. For example, many customers run automated start/stop scripts that turn off development environments during non-business hours to reduce costs. In this scenario, Amazon Elastic Compute Cloud (Amazon EC2) instance tags provide a simple way to identify development instances that should keep running.

Automating Elasticity

With AWS, you can automate both volume-based and time-based elasticity, which can provide significant savings. For example, companies that shut down EC2 instances outside of a 10-hour workday can save 70% compared to running those instances 24 hours a day. Automation becomes increasingly important as environments grow larger and become more complex, in which manually searching for elasticity savings becomes impractical.

Automation is powerful, but you need to use it carefully. It is important to minimize risk by giving people and systems only the minimum level of access required to perform necessary tasks. Additionally, you should anticipate exceptions to automation plans and consider different schedules and usage scenarios. A one-size-fits-all approach is seldom realistic, even within the same department. Choose a flexible and customizable approach to accommodate your needs.

Automating Time-Based Elasticity

Most non-production instances can and should be stopped when they are not being used. Although it is possible to manually shut down unused instances, this is impractical at larger scales. Let's consider a few ways to automate time-based elasticity.

AWS Instance Scheduler

The AWS Instance Scheduler is a simple solution that allows you to create automatic start and stop schedules for your EC2 instances. The solution is deployed using an AWS CloudFormation template, which launches and configures the components necessary to automatically start and stop EC2 instances in all AWS Regions of your account.

During initial deployment, you simply define the AWS Instance Scheduler default start and stop parameters and the interval you want it to run. These values are stored in Amazon DynamoDB and can be overridden or modified as necessary. A custom resource tag identifies instances that should receive AWS Instance Scheduler actions. The solution's recurring AWS Lambda function automatically starts and stops appropriately tagged EC2 instances. You can review the solution's custom Amazon CloudWatch metric to see a history of AWS Instance Scheduler actions.

Amazon EC2 API tools

You can terminate instances programmatically using Amazon EC2 APIs, specifically the StopInstances and TerminateInstances actions. These APIs let you build your own schedules and automation tools. When you stop an instance, the root device and any other devices attached to the instance persist. When you terminate an instance, the root device and any other devices attached during the instance launch are automatically deleted. For more information about the differences between rebooting, stopping, and terminating instances, see Instance Lifecycle in the *Amazon EC2 User Guide*.

AWS Lambda

AWS Lambda serverless functions are another tool that you can use to shut down instances when they are not being used. You can configure a Lambda function to start and stop instances when triggered by Amazon CloudWatch Events, such as a specific time or utilization threshold. For more information, read this Knowledge Center topic.

AWS Data Pipeline

AWS Data Pipeline is a web service that helps you reliably process and move data between different AWS compute and storage services, as well as on-premises data sources, at specified intervals. It can be used to stop and start Amazon EC2 instances by running AWS Command Line Interface (CLI) file commands on a set schedule. AWS Data Pipeline runs as an AWS Identity and Access Management (IAM) role, which eliminates key management requirements. For instructions on using this approach, watch this tutorial.

Amazon CloudWatch

Amazon CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS. You can use Amazon CloudWatch to collect and track metrics and log files, set alarms, and automatically react to changes in your AWS resources. You can use Amazon CloudWatch alarms to automatically stop or terminate EC2 instances that have gone unused or underutilized for too long. You can stop your instance if it has an Amazon Elastic Block Store (Amazon EBS) volume as its root device. A stopped instance retains its instance ID and can be restarted. A terminated instance is deleted. For more information on the difference between stopping and terminating instances, see the Stop and Start Your Instance in the *Amazon EC2 User Guide*.

For example, you can create a group of alarms that first sends an email notification to developers whose instance has been underutilized for 8 hours, and then terminates that instance if its utilization has not improved after 24 hours. For instructions on using this method, see the Amazon CloudWatch User Guide.

Automating Volume-Based Elasticity

By taking advantage of volume-based elasticity, you can scale resources to match capacity. The best tool for accomplishing this task is Auto Scaling, which you can use to optimize performance by automatically increasing the number of EC2 instances during demand spikes and decreasing capacity during lulls to reduce costs. Auto Scaling is well-suited for applications that have stable demand patterns and for ones that experience hourly, daily, or weekly variability in usage.

Beyond Auto Scaling for Amazon EC2, you can use Application Auto Scaling to automatically scale resources for other AWS services, including:

- Amazon Elastic Container Service (Amazon ECS) – You can configure Amazon ECS to use Service Auto Scaling to adjust its desired count up or down in response to CloudWatch alarms. For more information, read the documentation.
- Amazon EC2 Spot Fleets – A Spot Fleet can either launch instances (scale out) or terminate instances (scale in), within the range that you choose, in response to one or more scaling policies. For more information, read the documentation.
- Amazon EMR clusters – Auto Scaling in Amazon EMR allows you to programmatically scale out and scale in core and task nodes in a cluster based on rules that you specify in a scaling policy. For more information, read the documentation.
- Amazon AppStream 2.0 stacks and fleets – You can define scaling policies that adjust the size of your fleet automatically based on a variety of utilization metrics and optimize the number of running instances to match user demand. You can also choose to turn off automatic scaling and make the fleet run at a fixed size. For more information, read the documentation.
- Amazon DynamoDB – You can dynamically adjust provisioned throughput capacity in response to actual traffic patterns. This enables a table or a global secondary index to increase its provisioned read and write capacity to handle sudden increases in traffic without throttling. When the workload decreases, Application Auto Scaling decreases the throughput so that you don't pay for unused provisioned capacity. For more information, read to the documentation. You can also read our blog post Auto Scaling for Amazon DynamoDB.

Conclusion

The elasticity of cloud services is a powerful way to optimize costs. By combining tagging, monitoring, and automation, your organization can match its spending to its needs and put resources where they provide the most value. For more information about elasticity and other cost management topics, see the AWS Billing and Cost Management documentation.

Automation tools can help minimize some of the management and administrative tasks associated with an IT deployment. Similar to the benefits from application services, an automated or DevOps approach to your AWS infrastructure will provide scalability and elasticity with minimal manual intervention.

This also provides a level of control over your AWS environment and the associated spending. For example, when engineers or developers are allowed to provision AWS resources only through an established process and use tools that can be managed and audited (for example, a provisioning portal such as AWS Service Catalog), you can avoid the expense and waste that results from simply turning on (and most often leaving on) standalone resources.

Resources

- AWS Architecture Center
- AWS Whitepapers
- AWS Architecture Monthly
- AWS Architecture Blog
- This Is My Architecture videos
- AWS Answers
- AWS Documentation

Document Details

Contributors

The following individuals and organizations contributed to this document:

- Amilcar Alfaro, Sr. Product Marketing Manager, AWS
- Erin Carlson, Marketing Manager, AWS
- Keith Jarrett, WW BD Lead - Cost Optimization, AWS Business Development

Date	Description
March 2018	First publication

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.