REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
01/01/2020	Technical Report	

4. TITLE AND SUBTITLE
AWS GovCloud Resource and Cost Analysis

5a. CONTRACT NUMBER
W56KGU-18-D-0004/001

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Mahakian, Joseph; Holmdahl, Scott; Bada, Quadri; Silva, Steffani; Tretler, Zach;

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102

8. PERFORMING ORGANIZATION REPORT NUMBER
PRS-19-3881

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Background Investigation Services (NBIS) Program Management Office

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This technical report outlines many of the most impactful principles and techniques for reducing the cost of hosting a program on Amazon Web Services (AWS) GovCloud. To validate the cost saving techniques outlined in this technical report, a proof of concept was performed using a prototypical AWS DevSecOps account. The scope of this paper is limited to data and applications hosted in GovCloud and does not address cost optimization for systems hosted outside of GovCloud, including external data transferring into or out of GovCloud.

15. SUBJECT TERMS
Information Architectures; Network Architectures; Amazon Web Services; maintenance; GovCloud; cost savings; cloud-hosted IT systems; operation;

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Susan Carpenito
				37	**19b. TELEPHONE NUMBER** *(Include area code)* 781-271-7646

AWS GovCloud Resource and Cost Analysis

The views, opinions and/or findings contained in this report are those of The MITRE Corporation and should not be construed as an official government position, policy, or decision, unless designated by other documentation.

Author(s):
Joseph Mahakian
Scott Holmdahl
Quadri Bada
Steffani Silva
Zach Tretler

McLean, VA
January 2020

MITRE

This page intentionally left blank.

Executive Summary

This technical report outlines many of the most impactful principles and techniques for reducing the cost of hosting a program on Amazon Web Services (AWS) GovCloud. In full production, cloud-hosted IT systems have the potential to incur significant annual costs associated with operation and maintenance, so in order to optimize resource allocation and achieve significant cost savings, the Program Management Office (PMO) should leverage cost-saving techniques and services which are designed to complement AWS' dynamic cloud environment.

PMOs can realize cost savings and institute effective cost management practices by focusing on three areas: deployment strategies, cost management tools, and programmatic best practices. The scope of these considerations includes both architecture design decisions as well as the integration of services to improve the PMO's insight into billing.

To validate the cost saving techniques outlined in this technical report, a proof of concept was performed using a prototypical AWS DevSecOps account. The account currently costs an estimated $7,059 per month, whereas by applying the principles, the estimated bill was reduced by 43% to $4,001 per month while maintaining identical functionality. Because EC2 and RDS instances account for 97% of the bill for the example DevSecOps account, the majority of the implemented cost saving techniques were related to EC2 and RDS instances. Not all of this report's cost saving techniques could be applied to a typical DevSecOps account, but should be considered nonetheless.

Focusing on the compute cost component, one of the most practical and effective techniques for the PMO to reduce annual costs is the utilization of Reserved Instances, which offer significant discounts based on time commitment and payment method, instead of On-Demand Instances. To understand the potential cost savings of using a Reserved Instance, a calculation was performed for a representative instance of a typical program (a Linux c5.xlarge EC2 instance in the GovCloud US-EAST region).

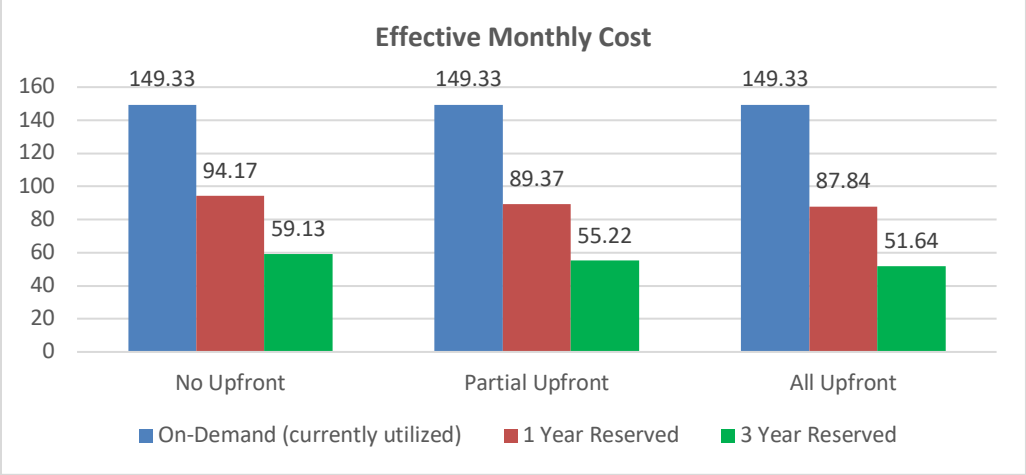

Figure 1 Cost Comparison of On-Demand and Reserved Instances

As demonstrated in Figure 1, the PMO can immediately realize cost savings of 37% by switching to one-year Reserved Instances, or 60% by switching to three-year Reserved Instances. Even further cost savings can be achieved with other tools such as auto-scaling groups.

Table of Contents

List of Figures

List of Tables

This page intentionally left blank.

1 Introduction

Many government programs interact with one or more IT systems of systems that is developed, operated, and maintained by several government stakeholder groups. In a cloud-hosted IT stack, many of the technical components of the program's system may be hosted in Amazon Web Services (AWS) GovCloud. In full production, there are often significant costs incurred by utilizing GovCloud services. With the large volume of services being used to create the program's IT system, it is imperative to evaluate their use in a dynamic and scalable cloud environment to ensure maximum cost effectiveness while maintaining full operational capability.

The scope of this paper is limited to data and applications hosted in GovCloud and does not address cost optimization for systems hosted outside of GovCloud, including external data transferring into or out of GovCloud. Although many government programs run solely in GovCloud regions, the paper may include tools and services which have not yet been released in GovCloud regions at this time. There is usually a delay before AWS services for publicly available regions are accredited and become available for use in GovCloud.

This technical report is broken into three themes of cost optimization: cost deployment strategies, cost management tools, and programmatic best practices. The cost deployment strategies sections discusses nine AWS recommended management strategies to limit the costs associated with running instances in the cloud environment. The cost management tools section defines the AWS services available to implement these cost-management strategies. Finally, the programmatic best practices defines recommended actions to introduce and maintain a cost-effective cloud environment to an organization. The report culminates in an evaluation of an sample GovCloud DevSecOps environment and how the utilization of these management strategies can significantly reduce program costs.

2 Optimize for the Cloud

In a traditional datacenter environment, teams were limited to either design architectures within the constraints of existing hardware or request additional resources, which is often an arduous task. Datacenter procurements require the purchase of enough servers to support an application during peak hours, which results in a misallocation of resources while equipment sits idle during non-peak hours.

The cloud is fundamentally different from a traditional datacenter in that it offers virtually unlimited scalable capacity with tremendous opportunity for increased design flexibility. In the cloud cost model, one only pays for the resources one uses. In order to maximize the benefits of hosting systems in a cloud environment, cloud-based applications should be architected in a method that is appropriate and economical for the environment it is hosted in. This operating model requires an IT strategy which is cloud conscious.

2.1 AWS GovCloud

AWS GovGloud regions are isolated cloud regions of Amazon Web Services for workloads with direct or indirect ties to U.S. Government functions or services. All customers who use GovCloud must either be Government organizations or other approved private entities in Government-related industries such as Aerospace, Law Enforcement, and Healthcare.

Currently there are two GovCloud regions, US-West and US-East. The regions are physically located in the United States and are designed to allow U.S. government agencies and customers to move sensitive workloads into the cloud by addressing their specific regulatory and compliance requirements, including Federal Risk and Management Program (FedRAMP), High Department of Defense Security Requirements Guide (DoD SRG), Impact Level 5, and Criminal Justice Services (CJIS).

3 Cost Optimization – AWS Deployment Strategies

This section of the technical report describes cost-saving strategies and practices for resources deployed in an AWS cloud environment. While many of the concepts described in this section are cloud-provider agnostic, the exact tools discussed are AWS-specific.

Nine strategies discussed in this section include:

- Reserved Instances: instances set in advance at a discounted rate
- Spot Instances: instances that can be bid on and will run until cost exceeds bid price
- Auto Scaling and Automating Elasticity: scaling instances based on volume demand
- S3 Object Lifecycle Management: storage type changes by object lifecycle and access frequency
- Instance States and Stopping Instances: billing only occurs during running states
- Tagging: labels applied to resources to maintain visibility for cost management
- AWS Config: change management of AWS resource configurations
- AWS Organizations: managing policies and monitoring of all AWS accounts within an organization
- EC2 Right Sizing: appropriately sizing EC2 instance type and size based on usage demand

3.1 Reserved Instances

Many program's workloads on GovCloud have a predictable usage pattern, with consistent downtimes and long term utility. For these stable applications, the PMO can achieve significant cost savings by converting these instances from On-Demand Instances to Reserved Instances.

Reserved Instances enable organizations to commit to usage parameters for a specified amount of time (either one year or three years) in exchange for significant cost savings compared to On-Demand Instances. Functionally, Reserved Instances are identical to On-Demand Instances and are available for services including EC2, RDS, Redshift, and more. The discount rate is a function of time commitment (one year or three years) and payment method (no upfront, partially upfront, or entirely upfront), with maximum savings occurring for three year commitments paid entirely upfront.

To understand the potential cost savings of using a Reserved Instance, a proof of concept has been calculated for a Linux c5.xlarge EC2 instance on the GovCloud (US-EAST) region. This instance features 4 vCPU and 8GB Memory. The graph below shows the monthly cost associated with running this EC2 instance given the different pricing models.

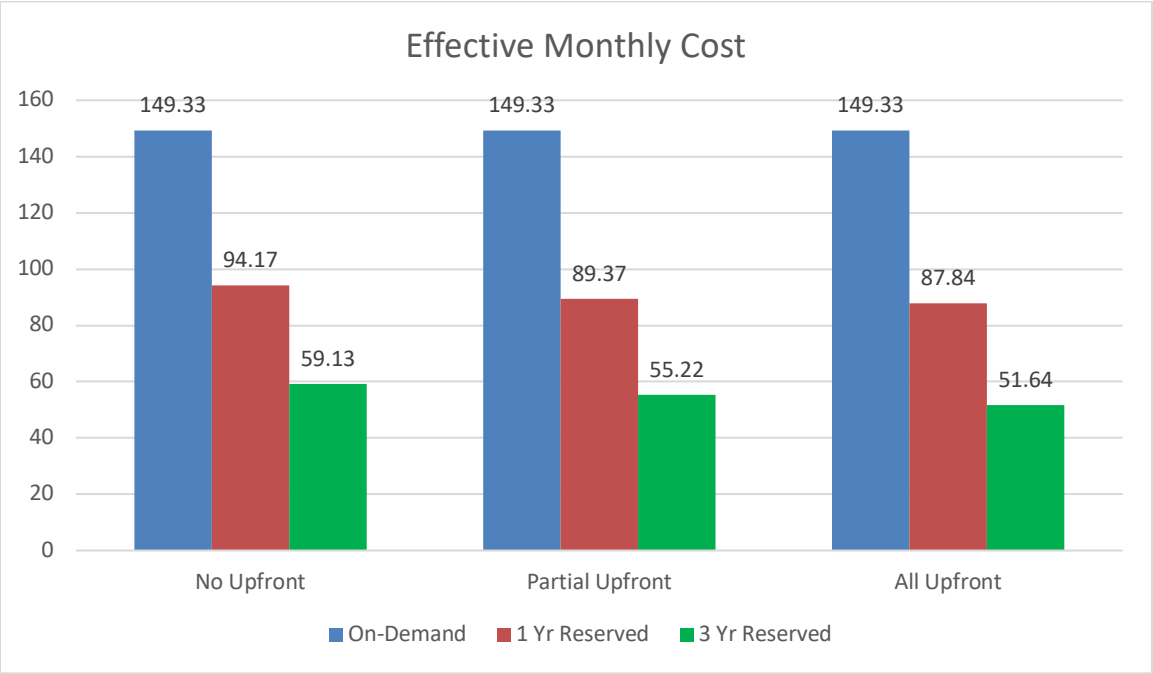

Figure 2 Cost Comparison of On-Demand and Reserved Instances

Currently, a c5.xlarge On-Demand Instance costs $149.33 per month in the cloud. If that exact instance were transitioned to the Reserved Instance price model for a three year agreement, with no upfront payment, the price is reduced to $59.13 per month, which represents more than 60% savings. Even further savings could be realized with a partially or entirely upfront payment.

3.2 Spot Instances

In addition to On-Demand and Reserved Instances, AWS also offers EC2 Spot instances, which allow one to bid on spare EC2 computing capacity at a significantly discounted rate compared to On-Demand and Reserved Instances. Figure 3 illustrates the significant discount rate Spot Instances offer when compared to the other EC2 types. The figure displays monthly costs for deploying a c5.xlarge instance in the GovCloud (US-EAST) region for all variations of On-Demand, Reserved, and Spot instances. The c5.xlarge and the collective c-series EC2 family are the most commonly deployed EC2 instances in the example program's DevSecOps architecture (see Section 6).

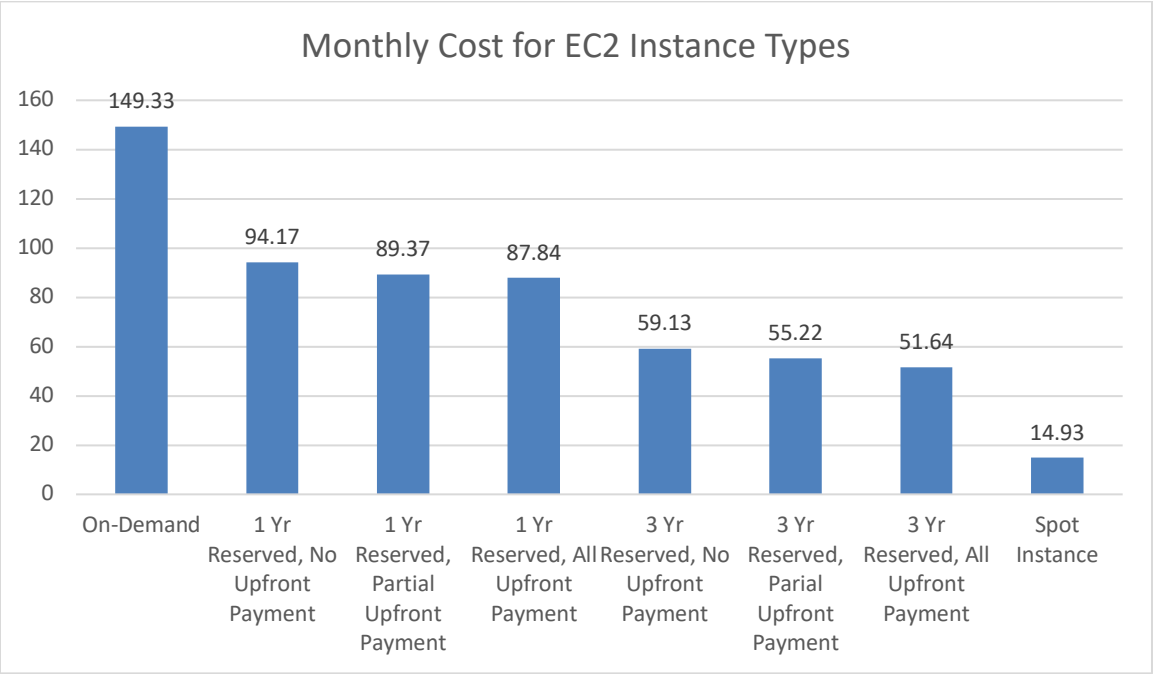

Figure 3 EC2 Instance Prices

Unlike the discounted Reserved Instances, Spot Instances do not require an upfront commitment or upfront payment. However, because Spot Instances can be terminated if the Spot price exceeds your maximum bid price or if no capacity is available for the specified instance type, they are best for flexible workloads. By properly architecting specific workloads, you can take advantage of Spot pricing for a wide range of needs.

3.3 Auto Scaling and Automating Elasticity

In cloud environments such as AWS, resources are elastic and can immediately scale up or down to match application requirements. This elasticity allows cloud administrators to match the supply of resources, to demand, which optimizes cost in the process. There are two primary types of elasticity, time-based and volume-based. Time-based elasticity implies turning off instances when they are not being used (see Section **Error! Reference source not found.**), while volume-b ased elasticity involves matching the scale of resources to the intensity of demand. Amazon provides tools to automate both volume and time-based elasticity.

Section **Error! Reference source not found.** discusses strategies for automating time-based e lasticity. The best tool to automating volume-based elasticity is Auto Scaling, which automatically increases or decreases the number of EC2 instances during demand spikes and lulls. Auto Scaling monitors your applications and automatically adjusts capacity to maintain performance based on thresholds set by the cloud administrator. Applications that have stable demand patterns such as predicable workloads on a hourly, daily, or weekly basis, are excellent candidates to be configured with an Auto Scaling configuration.

To enable Auto Scaling, an Auto-Scaling Group (ASG) can be configured. An ASG is AWS managed service which provides a set of EC2 instances with automatic scaling logic to launch or

terminate instances according to computing requirements. ASGs enable AWS users to instantiate business rules which, for example, launch additional EC2 instances when the current set of EC2 instances exceeds 80% capacity for more than 30 seconds.

The cost savings associated with ASGs are realized when they are implemented on applications that have variable usage patterns. For example, Gitlab Runners require significant computing capacity when running jobs, but sit idle for long durations between jobs. With an ASG in place, the Gitlab Runners can utilize just one EC2 instance when idle between jobs, then scale up to multiple, appropriately-sized EC2 instances when jobs arrive.

3.4 S3 Object Lifecycle Management

Amazon Simple Storage Service (S3) is an object storage service with virtually unlimited storage capacity. There are a number of S3 storage classes that offer different degrees of durability and availability at different price points. It is common for many applications on AWS to store logs and other information, which are referred to as objects, in an S3 bucket. Since S3 has numerous storage classes which are each optimized for different use cases, it is important to manage objects with lifecycle configurations so they are stored in appropriate S3 classes to minimize storage costs. By enabling lifecycle configurations, one can define rules to transition objects from one storage class to another cheaper storage class to save on storage costs.

The different storage classes, their use cases, and their associated cost for GovCloud US-East (calculated for 50 TB of storage or less) are listed in Table 1**.** The different mobility options for transferring objects between storage classes are illustrated in Figure 4**.**

Table 1 S3 Storage Classes

Storage Class	Designed For	Durability (Designed For)	Availability (Designed For)	Cost (Month)
STANDARD	Frequently accessed data	99.999999999%	99.99%	$0.039 per GB
STANDARD_IA	Long-lived, infrequently accessed data	99.999999999%	99.9%	$0.02 per GB
INTELLIGENT_TIERING	Long-lived data with changing or unknown access patterns	99.999999999%	99.9%	$0.039 per GB
ONEZONE_IA	Long-lived, infrequently accessed, non-critical data	99.999999999%	99.5%	$0.016 per GB
GLACIER	Long-term data archiving with retrieval times ranging from minutes to hours	99.999999999%	99.99% (after you restore objects)	$0.006 per GB
DEEP_ARCHIVE	Archiving rarely accessed data with a default retrieval time of 12 hours	99.999999999%	99.99% (after you restore objects)	$0.0024 per GB

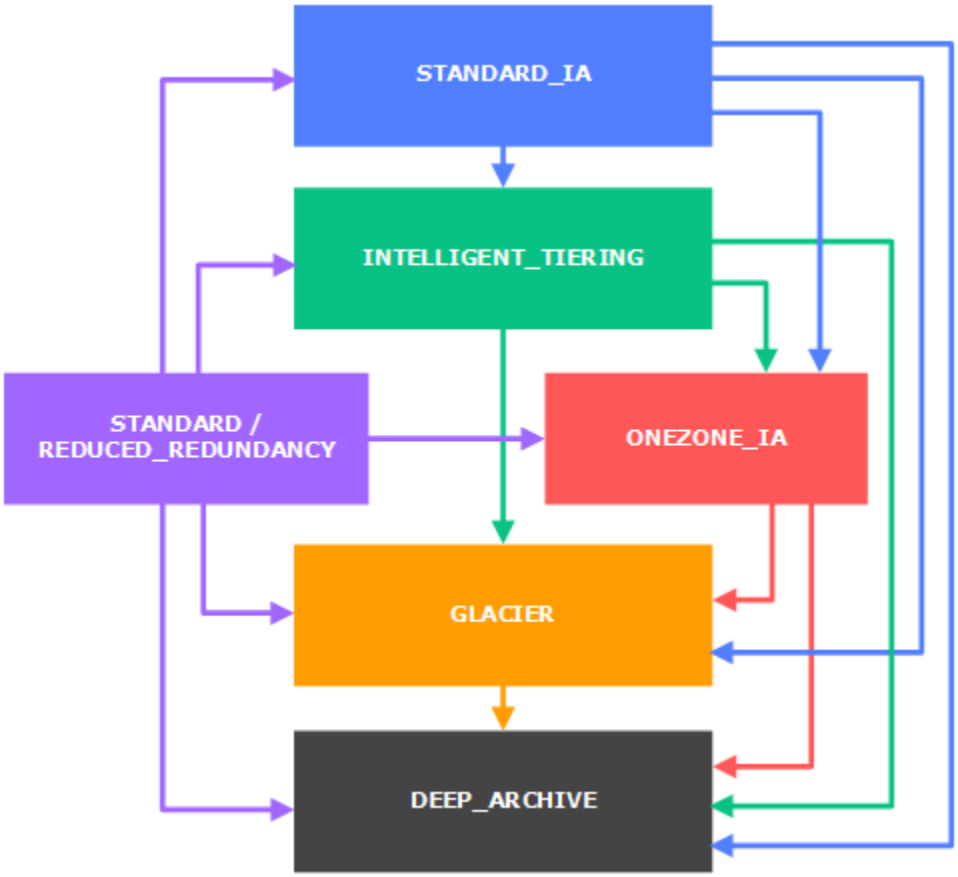

Figure 4 S3 Object Mobility

To manage object storage in a cost effective manner, S3 lifecycle configurations should be configured which define actions (e.g., transfer, expire) that S3 applies to a group of objects. Log files are generally good candidates for S3 lifecycle management since logs are often required for a specified amount of time, after which they can be either archived for infrequent use and regulatory compliance, or expired (i.e., deleted). An example lifecycle for application logs can been seen in Figure 5**.**

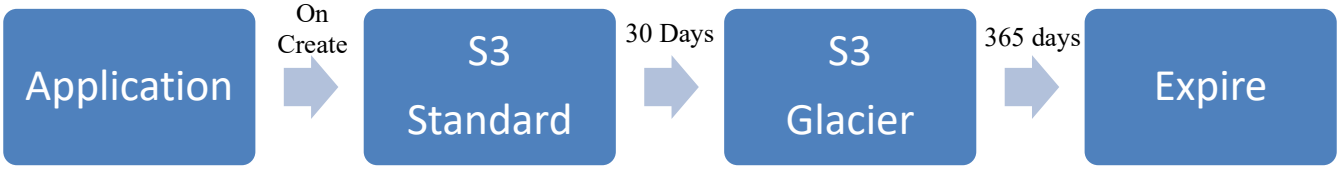

Figure 5 Sample Lifecycle Management for Application Logging on S3

Hypothetically, if 50TB of logs and artifacts were produced in an enterprise environment with lifecycle management not configured, the cost to store 50TB in a *Standard* storage class would cost $1,950 per month. Alternatively, if the logs were transferred to *Glacier* after 30 days (i.e., only the most recent 30 days of logs were stored in a *Standard* storage class), the cost would be reduced to $437.50 per month. This represents a 78% reduction of S3 storage costs.

3.5 Instance States and Stopping Instances

Amazon EC2 and RDS services have a number of possible states in their lifecycle from Launch to Termination. These states and the procedures that associated with each state are illustrated in **Error! Reference source not found.**.

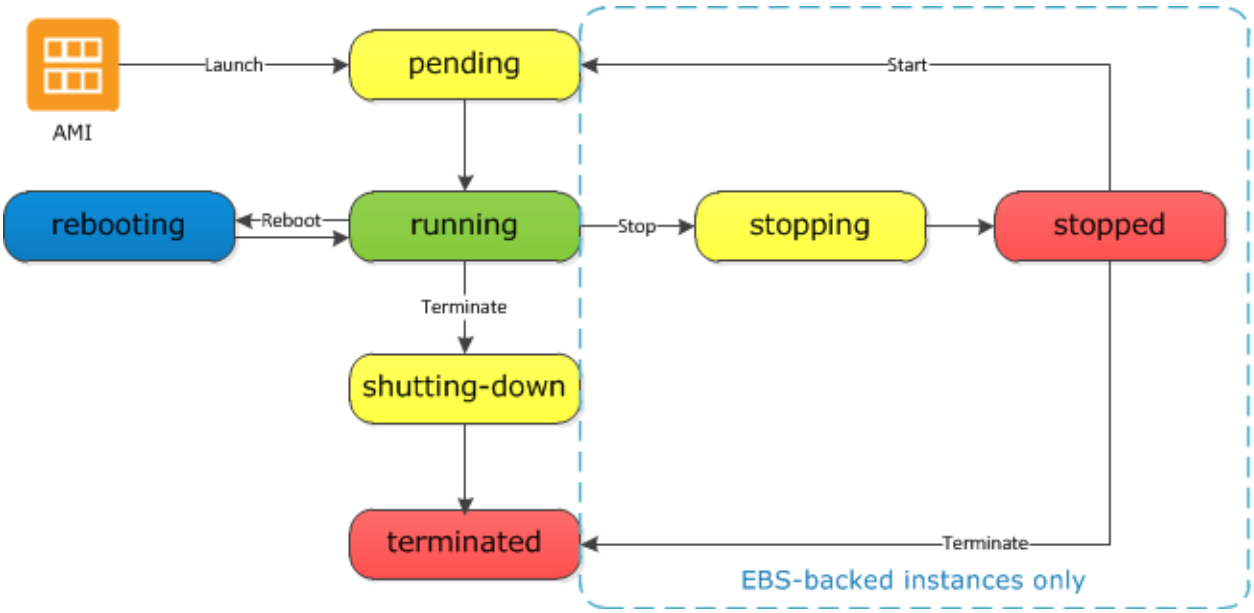

Figure 6 Transitions Between Instance States

EC2 and RDS instances are billed on either a per-second or per-hour basis, and billing is applied only during certain states. An important observation is that instances are not billed when they are stopped. A complete table of billing policies per instance state can be seen in **Error! Reference s ource not found.**. When an instance is stopped by an administrator, it enters the "stopping" state, and then the "stopped" state. Amazon does not charge usage or data transfer fees for the instance after it has been stopped, however Amazon does charge for the storage of any Amazon EBS volumes attached to the instance since they are not destroyed and remain operational when an instance is stopped.

The billing structure of instance states and stopped instances provides an opportunity for drastic cost savings for certain types of workflows. EC2 or RDS systems, which are used infrequently or for intermittent jobs, such as batch processing, are ideal candidates for this type of procedure. Dev and Test environments which are only used during working hours are also prime candidates to be stopped while they are not being used. The instances can be stopped while they are not in use to pause their billing accrual, then restarted when needed. Instances can be stopped programmatically through an API call, which allows this process to be scripted. Likewise AWS offers tools such as an EC2 scheduler which automates the start/stop procedure through CloudFormation templates. Furthermore, instances retain their private IPv4 address, which means that an Elastic IP address associated with the private IPv4 address or network interface will still be associated with your instance after a stop/start procedure.

Table 2 Instance States and Billing

Instance State	Description	Instance usage billing
pending	The instance is preparing to enter the running state. An instance enters the pending state when it launches for the first time, or when it is restarted after being in the stopped state.	Not billed
running	The instance is running and ready for use.	Billed
stopping	The instance is preparing to be stopped or stop-hibernated.	Not billed if preparing to stop Billed if preparing to hibernate
stopped	The instance is shut down and cannot be used. The instance can be restarted at any time.	Not billed
shutting-down	The instance is preparing to be terminated.	Not billed
terminated	The instance has been permanently deleted and cannot be restarted.	Not billed **Note:** Reserved Instances that applied to terminated instances are billed until the end of their term according to their payment option.

See Section 6.2.3 for a discussion of how stopping and re-starting instances would be implemented on the example DevSevOps account.

3.6 Tagging

A tag is a label which contains custom metadata that either the user or AWS assigns to an AWS resource. Each tag consists of a key and value pair, where each key can have one or multiple values. Each resource can have up to 50 user-applied tags, such as *[createdBy = employee90440]*. Tagging resources is an essential component of effective cost management in the cloud since it enables administrators to streamline resource management, access management, and cost allocation by allowing for detailed cost tracking. Tagging gives direct accountability and visibility into IT costs by team and application.

Cost allocation tags are an optional form of tagging that can be activated in the Billing & Cost Management Dashboard. After cost allocation tags have been activated, AWS uses the tags to organize resources in a cost allocation report so one can understand the cause and quantity of charges associated with each resource. For example, if you tag resources with an application name, you can track the total cost of a single application. This would allow the system administrators to directly track costs of different applications and use that information to drive future decisions. You can also use tags to filter views in Cost Explorer for further insight into system costs.

3.7 AWS Config

AWS Config is a service which provides a detailed view of the configuration of AWS resources in your AWS account. When enabled, Config discovers resources that exist in your account, records their current configuration, and captures any changes to these configurations. This includes how the resources are related to one another and how they were configured in the past, which enables administrators to understand how configurations and relationships change over time. This ability provides numerous benefits including continuous assessment, change management, compliance monitoring, operational troubleshooting, and cost understanding.

To effectively manage costs in the cloud, cloud administrators must understand exactly what is running in the cloud and for what purpose. Although this concept may appear obvious, it is often overlooked in cloud deployments as it is fundamentally different from on premise datacenter deployments where infrastructure costs are fixed, making resource evaluation less critical. Monitoring resources with methods such as Tagging and AWS Config reduces the likelihood of misallocated resources and inefficient architectures.

3.8 AWS Organizations

Organizations or applications that use multiple AWS accounts to fulfill their IT needs may choose to enroll in AWS Organizations at no additional cost. AWS Organizations provides central governance and management across multiple AWS accounts. It allows you to create groups of accounts, apply policies for governance, automate account creation, and consolidate billing.

Amazon offers volume discounts for services purchased in large quantities. Most of the common services such as EC2 and S3 have volume pricing tiers across certain usage dimensions that provides lower prices the more one uses the service. Below is an example of the volume discounts offered for S3.

S3 Standard Storage	Pricing
First 50 TB / Month	$0.023 per GB
Next 250 TB / Month	$0.022 per GB
Over 500 TB / Month	$0.021 per GB

Enabling AWS Organizations will allow organizations with multiple accounts to take advantage of the discount volume pricing. For billing purposes, AWS treats all the accounts in the organization as if they were one account, combing the usage from all accounts to determine which volume pricing tiers apply. In addition to volume discounts, AWS Organizations also allows for consolidated billing with a unified billing dashboard which shows a view of charges incurred by all accounts.

Enrolling multiple GovCloud accounts in AWS Organizations would reduce current and future costs as the system expands in production, as it has the potential to reduce programmatic costs through volume discounts. Additionally, it will provide administrators more insight into incurred costs through the consolidated billing dashboards and reports. AWS Organizations was deployed

on GovCloud in April of 2019 and should be incorporated into most programs' GovCloud strategy.

3.9 EC2 Right Sizing

Right sizing is a key mechanism for optimizing costs on AWS, but is often ignored when migrating or developing systems in the cloud. Right sizing is the process of matching instance types and size to workload demands and capacity requirements in order to minimize costs. Right sizing also includes the process of monitoring existing instances and identifying opportunities to eliminate or downsize without compromising reliability and other requirements.

Historically, IT systems were provisioned to be able to accommodate traffic during peak hours. This approach is not optimized for the cloud and should not be used when deploying instances in GovCloud. To minimize costs, instances should take advantage of the elasticity offered by cloud environments and should be provisioned based on average usage rather than peak usage with supplemental tools such as auto-scaling to ensure appropriate performance during peak hours.

The first step to right sizing is to monitor and analyze the current use of existing services and infrastructure to gain insight into performance and usage patterns. When monitoring instances, the most important hardware considerations to monitor are CPU, Memory, Network, and Storage including disk read/write operations. These metrics should be collected and analyzed to determine not only the most appropriate EC2 size, but also instance family (General Purpose, Compute Optimized, Memory Optimized, Storage Optimized, Accelerated Compute).

Amazon recommends that instances with a maximum CPU usage and memory usage of less than 40% over a four-week period are candidates for downsizing to smaller instances. As a general rule of thumb, they recommend EC2 instances which meet this threshold can safely be cut in half. For example, a c4.8xlarge which uses less than 40% of available CPU over a 4 week period, can be moved to a c4.4xlarge which would save approximately $570 every month.

4 Cost Optimization – Cost Management Tools

AWS services have a complex costing model resulting due to the cloud's abilities to scale based on resource utilization. This allows users to quickly adjust services and save money based on utilization. By consistently monitoring system infrastructure and reacting to changes as they occur, an IT system can work efficiently while maintaining low costs. To help the PMO take advantage of this structure, Amazon offers tools that monitor utilization, set thresholds, and provide detailed cost breakdowns. By understanding cloud cost management and utilizing Amazon's cost management tools, the government program can effectively scale resources while maintaining system security and performance, ultimately resulting in significant cost savings.

4.1 AWS Billing and Cost Management

For many government programs, different AWS accounts will be used with different root users, users, access, and AWS services. AWS Consolidated Billing can be used to aggregate the billing of services across multiple accounts for the purposes of cost savings and centralized cost management. To utilize this feature, AWS Organizations should be used as an account management service so that one master account can pay the charges across member accounts. Each GovCloud account must map to a commercial account, and those commercial accounts must be part of one single master commercial organization through the Organizations service. The master commercial account can view resource usage and activity information for all the AWS GovCloud accounts in the AWS Management Console. This master commercial account is also the account to which all charges will be billed for both commercial and GovCloud, which enables the PMO to reap the cost saving and management benefits of consolidated billing.

4.2 AWS Cost Explorer

AWS Cost Explorer helps one track spending via default or custom reports which are accessible via console or API. For example, one can see which services, accounts, or regions are most expensive and quickly analyze on one's resources, such as the utilization of EC2 instances. One can also get more in-depth guidance such as forecasting, resource optimization, and Reserved Instance purchase recommendations. The cost per request is $0.01, which can be generated hourly and/or daily.

4.3 AWS Trusted Advisor

AWS Trusted Advisor scans one's AWS environment and recommends improvements specific to one's architecture. Without signing up or paying for the service, all AWS customers are entitled to seven core Trusted Adviser checks which focus on permissions, EBS and RDS public snapshots, and service limits. These checks are accessible through dashboards in the console or via API and can come with a weekly notification email with updates on Trusted Advisor check statuses and potential savings. In the console, cost optimization can be selected to view suggested cost-reducing system changes. By tracking recent changes and checks such as low utilization Amazon EC2 Instances, Trusted Advisor helps reduce costs in one's AWS environment.

4.4 AWS Budgets

AWS Budgets can be used to automate the continuous monitoring of costs. Rather than constantly checking costs or usage, the Budgets tool can be utilized to alert the team when thresholds are reached that may lead to unexpected spending. The team can choose to monitor based on estimated costs and predicted costs, but can also set up alerts for monitoring the utilization of resources, such as a Reserved Instance. Using this in conjunction with AWS Organizations and CloudWatch alarms, the PMO can allow users to create account-specific budgets which can then be viewed through the budget dashboard and monitored via alerts from email, Slack, or Amazon Chime. Utilizing AWS Budgets costs $0.02 per day per account.

4.5 AWS CloudWatch

While Budgets focuses on monitoring cost and utilization thresholds, AWS CloudWatch monitors and automatically reacts to system health and performance issues. CloudWatch consolidates logs and metrics generated by each AWS service to provide quickly-digestible information to the team in the dashboard and through alerts. Beyond simply alerting the team to take action, CloudWatch allows users to set automated actions in which workflows are triggered by either user-specified thresholds or anomalies detected by Amazon's machine learning algorithms. For instance, changes in resource demand could trigger CloudWatch to enable auto-scaling for an EC2 instance. CloudWatch can provide extreme granularity, collecting metrics as frequently as every second and publishing them every minute. The information CloudWatch publishes in this process is available via the CloudWatch dashboard and API.

4.6 AWS CloudTrail

AWS CloudTrail is necessary to help the team manage security and compliance across accounts by monitoring account activity and mitigating risk. Through this service, account activity is tracked and workflows can be triggered to maintain high security with minimal oversight. For example, if an API call makes an S3 bucket public, CloudTrail will log this event and can respond by automatically adding specific security policies to the S3 bucket. In the white paper "Security at Scale: Logging in AWS," Amazon demonstrates how easy it is to track and retrieve activity data for auditing and enforcing compliance standards. CloudTrail comes free of charge, though S3 storage is needed to store log data.

4.7 Cost Optimization Monitor

The AWS Cost Optimization Monitor consolidates billing reports into an S3 bucket and presses the data for viewing and retrieval in an Elasticsearch service. With this feature enabled, users can utilize Kibana dashboards to search and visualize aggregated cost data as seen in Figure 5. For example, the team's Kibana dashboard may display cost by EC2 hours per dollar invested, instances running per hour, or cost by instance type. The Cost Optimization Monitor tool is available to the account that makes payments through the consolidated billing feature (see

Section 4.1). To use this feature, the team launches the Cost Optimization Monitor stack and then configures the payment account to save billing reports to a designated S3 bucket. Charges for this service are billed by Amazon Elasticsearch instance hours and storage, Amazon EC2 usage and licensing, and Elastic Load Balancing base pricing. The cost per hour will also fluctuate depending on deployment size, master node count and type, and data node count and type. The lowest cost per hour is estimated to be $0.40 and the highest cost per hour is estimated to be $3.83.

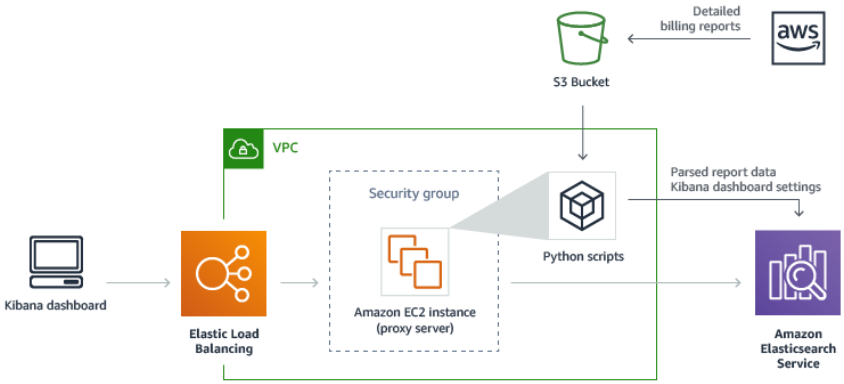

Figure 7 Cost Optimization Monitor architecture

4.8 AWS Cost and Usage Report

AWS Cost and Usage Reports are another method for reviewing each account's spending. This report type focuses on views by product, usage type, operations, more detailed information by Reserved Instance type, and the unique combinations of each of these attributes. By using the Consolidated Billing feature from AWS Organizations, the master account will have access to cost and usage reports for each linked account. These reports can be downloaded from the S3 console, accessed through the AWS Billing and Cost Management API, or queried through an Amazon service such as Redshift, QuickSight, or Athena. While there is no cost to access these reports, storing reports in S3 will incur normal S3 storage costs. Up to ten cost reports can be created per account which make cumulative estimated costs available throughout the month with information aggregated by the hour or the day. One finalized report will be provided by Amazon each month containing final calculations for all blended (affected by instance type and rate) and unblended rates, usage, and cost changes (e.g., due to refunds or credits).

5 Cost Optimization – Programmatic Best Practices

Before deploying services and cost optimization tools, it is better to first examine the program's current practices and work to craft a cost-conscious system. A Cloud Center of Excellence (CCoE) team should be appointed to monitor and enact the following practices and services within the program's environment. The following best practices will help guide the project toward being more cost-aware and, ultimately, minimize cloud-related expenses.

5.1 Architecture

Every government program needs to be deliberate about where it houses data and how it transfers data. Begin with a consumption model to determine current computing requirements based on the program's historical usage, then modify these requirements using cost optimization tools such as AWS Auto Scaling and Cost Explorer. Once these cost optimization tools have been introduced, there should be periodic reviews of the architecture to determine potential areas of improvement.

5.2 DevOps

The fundamental principle of any DevOps system is the ability to frequently release updates in order to shorten delivery time to the customer. In addition to time savings, an effective DevOps pipeline should reduce program costs over the entirety of its lifecycle, including the operation and maintenance phases. Sharing resources in the cloud environment during the integration phase can save on costs as well, while limiting to only necessary resources and keeping unused resources shut down. DevOps allows for continuous testing and monitoring which can find software bugs and apply fixes quicker. This can also be used for tracking cost optimization models to ensure they are running the correct instances and deploying resources only when needed to maximize cost reduction.

5.3 Lean Culture

Lean culture focuses on reducing waste by defining value, mapping value streams, creating flow, establishing pull-based systems, and pursuing perfection. Government agencies should implement a lean culture which clearly delineates its customer value and continuously tweaks its business process to maximize that value. Any processes that are deemed unnecessary for the customer should be reduced or eliminated. The process should be regularly evaluated to ensure a steady flow of progress is maintained across teams, enabling employees to adapt to changing customer needs. A pull-based system is defined as a system where the customer's product is delivered when the products are needed and limiting the amount of "in-progress" items. Finally, always test and improve upon any products delivered to the customer. In an agile system, it is important to get a functioning product released, then re-evaluate and perfect the system by reviewing customer feedback.

5.4 AWS-Oriented Business Process Re-Engineering

In order to facilitate cost savings, the government program can collect usage statistics for its AWS services and consider re-engineering its business process to minimize expenditures. For example, if certain data is housed for longer than necessary, the business process could be re-engineered to reduce storage duration (and therefore overall storage volume). In accordance with lean principles, when re-engineering the business process is important to minimize task duration by eliminating low-value tasks and appropriately dividing or combining tasks based on the size and time to completion. On top of this, tasks should be ordered based on required delivery date, available work capacity, and size. Tasks can either be done sequentially or in parallel depending on if it relies on information from external sources to complete the task. Employees should be given more control in the decision making process for their assigned tasks. It will save time and reduce overhead costs if employees are comfortable making decisions without requiring constant authorization and oversight.

5.5 Policy-Driven Automation

Utilize business policies to track automated processes, enable easily-adjustable automation, and understand the holistic effects of automation. Additionally, usage metrics can inform business policies and enable “smarter” policies. Incorporate methods to monitor GovCloud usage and determine which processes can (and cannot) be automated. Monitoring can be done using tags to track instance usage and filter by both application type and deployment zone (i.e., region or availability zone). There should be one centralized system or tool to monitor the GovCloud environment.

6 DevSecOps GovCloud Evaluation

This section provides an evaluation of the state of a standard DevSecOps GovCloud environment with the information that was available at the time of this technical report. Some assumptions were made in order to fill in missing information for certain items, such as storage size. Each assumption was made based off estimates of existing systems.

After the baseline environment was captured, the relevant principles described in this paper were conceptually applied to demonstrate the potential cost savings that can be achieved. All of the cost saving measures that are applied in this section preserve the full operational capabilities of the account and would not negatively affect performance.

6.1 Current Cost Estimate

A total of 61 individual AWS services are used in the operation of the example DevSecOps account. A complete table of all services, their descriptions, and associated costs can be found in Table 3. Costs were calculated for each service using the AWS Simple Monthly Calculator, and all costs were calculated for the GovCloud US-EAST region. By summing the cost for each individual service, an estimated monthly cost of $7,059.25 was calculated for a DevSecOps account.

Table 3 AWS Services in a DevSecOps Account

Service	ID	Description	Instance Type	Storage	Instance Cost	Storage Cost
EC2	1	SCCA SRX	c4.2xlarge	180GB	298.66	23.76
EC2	2	Bastion Host	c5.large	180GB	149.33	23.76
EC2	3	Bastion Host (Windows)	c5.large	150GB	142.01	19.80
EC2	4	Nginx Reverse Proxy	N/A	N/A	N/A	N/A
EC2	5	GitLab A	c5.xlarge	180GB	149.33	23.76
EC2	6	GitLab B	c5.xlarge	180GB	149.33	23.76
EC2	7	Artifactory A	c5.large	180GB	74.67	23.76
EC2	8	Artifactory B	c5.large	180GB	74.67	23.76
EC2	9	XRAY A	c5.2xlarge	180GB	298.66	23.76
EC2	10	XRAY B	c5.2xlarge	180GB	298.66	23.76
EC2	11	AD Admin	c5.large	150GB	74.67	19.80
EC2	12	AD CS	c5.large	150GB	74.67	19.80
EC2	13	BIND A	c5.large	180GB	74.67	23.76
EC2	14	BIND B	c5.large	180GB	74.67	23.76
EC2	15	WSUS	c5.large	150GB + 500GB	74.67	85.80
EC2	16	Jira A	c5.xlarge	180GB	149.33	23.76
EC2	17	Jira B	c5.xlarge	180GB	149.33	23.76
EC2	18	SonarQube	c5.large	180GB	74.67	23.76
EC2	19	CheckMarx	c5.large	150GB	74.67	23.76
EC2	20	Confluence A	c5.xlarge	180GB	149.33	23.76
EC2	21	Confluence B	c5.xlarge	180GB	149.33	23.76
EC2	22	Chef	c5.xlarge	180GB	149.33	23.76
EC2	23	Chef Automate	c5.2xlarge	180GB	298.66	23.76
EC2	24	RunnerA01	c5.large	180GB	74.67	23.76
EC2	25	RunnerA02	c5.large	180GB	74.67	23.76
EC2	26	RunnerA03	c5.large	180GB	74.67	23.76
EC2	27	RunnerA04	c5.large	180GB	74.67	23.76
EC2	28	RunnerA05	c5.large	180GB	74.67	23.76

EC2	29	RunnerA06	c5.large	180GB	74.67	23.76
EC2	30	RunnerA07	c5.large	180GB	74.67	23.76
EC2	31	RunnerA08	c5.large	180GB	74.67	23.76
EC2	32	RunnerA09	c5.large	180GB	74.67	23.76
EC2	33	RunnerA10	c5.large	180GB	74.67	23.76
EC2	34	RunnerB01	c5.large	180GB	74.67	23.76
EC2	35	RunnerB02	c5.large	180GB	74.67	23.76
EC2	36	RunnerB03	c5.large	180GB	74.67	23.76
EC2	37	RunnerB04	c5.large	180GB	74.67	23.76
EC2	38	RunnerB05	c5.large	180GB	74.67	23.76
EC2	39	RunnerB06	c5.large	180GB	74.67	23.76
EC2	40	RunnerB07	c5.large	180GB	74.67	23.76
EC2	41	RunnerB08	c5.large	180GB	74.67	23.76
EC2	42	RunnerB09	c5.large	180GB	74.67	23.76
EC2	43	RunnerB10	c5.large	180GB	74.67	23.76
ENI	1	Public Interface	N/A	N/A	N/A	N/A
ENI	2	Management Interface	N/A	N/A	N/A	N/A
ENI	3	Private Interface	N/A	N/A	N/A	N/A
EFS	1	GitLab App A	N/A	N/A	N/A	N/A
EFS	2	GitLab App B	N/A	N/A	N/A	N/A
RDS	1	GitLab	db.m4.large	200GB	175.07	27.60
RDS	2	Artifactory	db.m4.xlarge	200GB	175. 07	27.60
RDS	3	XRAY	db.m4.large	200GB	175. 07	27.60
RDS	4	SonarQube	db.m4.large	200GB	175. 07	27.60
RDS	5	Jira	db.m4.large	200GB	175. 07	27.60
RDS	6	Confluence	db.m4.large	200GB	175. 07	27.60
ELB	1	GitLab	N/A	N/A	N/A	N/A
ELB	2	Artifactory	N/A	N/A	N/A	N/A
ELB	3	Jira	N/A	N/A	N/A	N/A
ELB	4	SonarQube	N/A	N/A	N/A	N/A
ELB	5	Confluence	N/A	N/A	N/A	N/A
ElastiCache	1	GitLab	N/A	N/A	N/A	N/A
Directory Service	1	AWS Directory Service	N/A	180GB	74.67	23.76
Total:					**5821.81**	**1237.44**
					Total:	**7059.25**

The most common and costliest services in the example DevSecOps account are the use of EC2 and RDS instances. Collectively, the EC2 and RDS instances account for $6,862.39 of the total $7,059.25, or 97% of the account's monthly cost. Therefore the majority of the conscious cost improvements are related to EC2 and RDS services, but the other cost saving techniques outlined in this paper are still viable for environments using a broader set of managed services across multiple AWS accounts.

6.2 Cost Conscious Improvements

6.2.1 Reserved Instances

The single most effective recommendation for cost optimization in the prototypical DevSecOps account, which can also be applied to any other accounts in the government program, is the use of Reserved Instances in place of On-Demand Instances. A more detailed description of Reserved Instances can be found in Section 3.1.

Table 4 lists all EC2 and RDS instance types that are used in the modeled DevSecOps accounts, along with their monthly costs for On-Demand and Reserved types, respectively. The table also lists the difference in cost and the percent discount of using a Reserved Instance rather than the On-Demand Instances, which are currently utilized for all of the example program's EC2 and RDS instances. Figure 8 shows a visual comparison of the pricing difference.

Table 4 Cost Comparison of DevSecOps Instance Types

Instance Type	On Demand	3 Year Reserved (No Upfront)	Cost Reduction	Percent of Costs Saved
c5.large	$74.67	$29.93	$44.74	60.0%
c5.xlarge	$149.33	$59.13	$90.20	60.4%
c5.2xlarge	$298.66	$118.99	$179.67	60.2%
db.m5.large	$152.82	$134.23	$18.59	12.2%
db.m5.xlarge	$302.88	$165.62	$137.26	45.3%

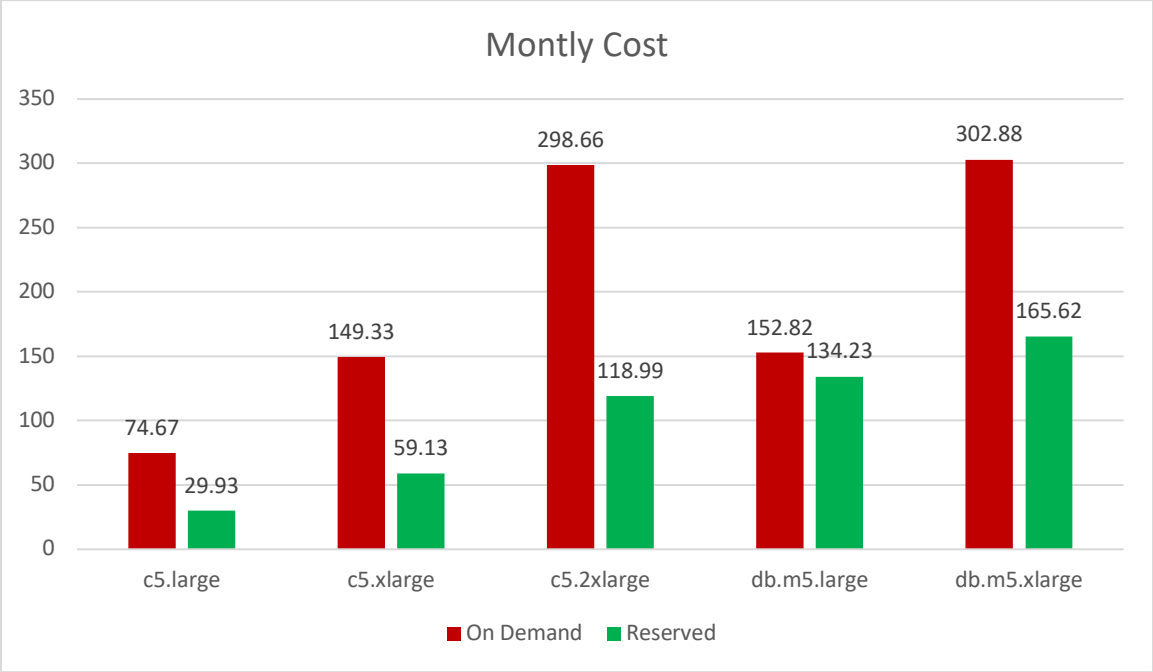

Figure 8 On-Demand Compared to Reserved Pricing

Table 5 shows the application of switching all EC2 and RDS instances from On-Demand to Reserved Instance types. The price difference is calculated for each type of instance existing in the account then multiplied by the quantity deployed. Costs for 3-year Reserved Instances with no upfront payment were used to calculate the cost savings. By using 3-year, no upfront payment instances, there is no additional immediate capital required to initiate this strategy.

Table 5 Application of Reserved Instances to DevSecOps Account

Instance Type	Quantity Deployed	Reserved Savings per Instance	Total Savings
c5.large	31	$44.74	$1386.94
c5.xlarge	8	$90.20	$721.6
c5.2xlarge	4	$179.67	$718.68
db.m5.large	5	$18.59	$92.95
db.m5.xlarge	1	$137.26	$137.26
Total	**49**		**$3057.43**

By utilizing Reserved Instances in a DevSecOps account, the total monthly bill was reduced from $7,059.25 to $4,001.82, which represents $3057.43 in cost savings. Reserved Instances were able to achieve a 43% savings with no upfront cost or performance reduction.

6.2.2 Auto-Scaling Groups

Another powerful technique for saving operational costs in a cloud environment is the implementation of Auto-Scaling Groups (ASG), which is an AWS managed service that utilizes automated scaling logic to launch or terminate instances according to computing requirements. ASGs enable AWS users to instantiate business rules which, for example, launch additional EC2 instances when the current set of EC2 instances exceeds 80% capacity for more than a specified amount of time. A detailed description of ASGs can be found in Section 3.3 Auto Scaling and Automating Elasticity.

The cost savings associated with ASGs are realized when they are implemented on applications that have variable usage patterns. For example, Gitlab Runners (or equivalent tools) require significant computing capacity when running jobs, but sit idle for long durations between jobs. With an ASG in place, the Gitlab Runners can utilize just one EC2 instance (instead of twenty) when idle between jobs, then scale up to multiple, appropriately-sized EC2 instances when Gitlab Runner jobs arrive.

In practice, the GitLab runners are often used for an hour or less per day, but they are required to be left running due to security scans. In this cost calculation exercise, we will assume the runners are only used for 1 hour a day (a liberal estimate), and the remaining 19 instances are set behind an auto-scaling group. In practice, the remaining instances could scale to be more or less than 19 instances, and their instance type could vary to accommodate jobs, but we will hold this assumption to show equivalent compute capacity of the current state at a reduced cost. To reconfigure the runners behind an auto-scaling group, one runner is running 24x7 so there is never any downtime when starting a job. When a job is submitted, the single runner will have a demand that exceeds capacity, and the auto-scaling group will be initiated. The remaining 19 instances will be deployed in a matter of minutes, and the workload can be distributed between them. This generates a total of $Projected\ Billable\ Hours = (1\ instance\ x\ 24\ hours) + (19\ isntances\ x\ 1\ hour) = 43\ instance\ hours$, when compared to the current state of all 20 instances running 24 hours a day, or $Current\ Billable\ Hours = (20\ instance\ x\ 24\ hours) = 480\ instance\ hours$. The use of auto-scaling groups reduces the billable time for the runners by 91 percent, reducing the monthly bill for the runners from $1,493.28 to $133.78.

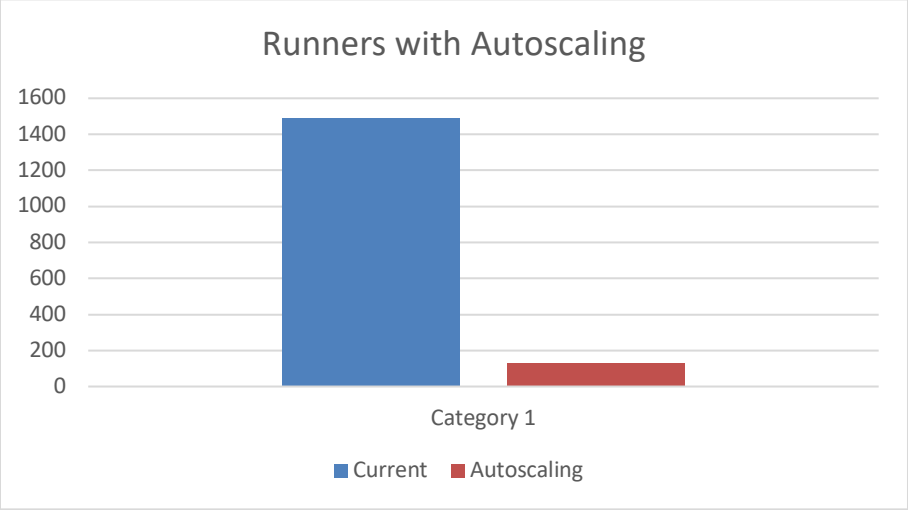

Figure 9 Cost Reduction using Auto-scaling

6.2.3 Stopped Instances

If Reserved Instances are not adopted, an alternative strategy would be stopping and starting instances based on utilization, particularly for the GitLab Runner instances. If Reserved Instances are used, stopping instances has no useful application since Reserved Instances are billed regardless of their usage. In contrast, On-Demand Instances can be stopped and started with no additional charges incurred. See **Error! Reference source not found.** for a complete e valuation of when instances are billed.

Stopping instances can be done programmatically through an API, and likewise can be started again through an API which allows for the entire stop-and-restart procedure to be automated. When an instance is stopped and restarted, it has the ability to retain its private IPv4 address, which means an Elastic IP address associated with the private IPv4 address will remain preserved for each instance after a stop-and-restart procedure.

If appropriate, workloads could be batched to run at scheduled times during the day when the Gitlab Runners are active. Alternatively the Gitlab Runners could be started when the script detects and incoming workload and automatically turned off when the workload is complete.

Table 6 Cost of a Single GitLab Runner

Instance Type	Compute Cost	Storage Cost	Total
c5.large	$74.67	$23.76	$98.43

The cost of a single Gitlab Runner is an estimated $98.43 per month, with the majority of the cost coming from the compute cost (i.e., when the instance is running). If the instances are only used for 2 hours a day, stopping the instances could reduce the compute cost of each Gitlab Runner by 92%, or $68.71 per month. When applied to each of the 20 Gitlab Runners, this strategy could result in a total savings of $1,374.20 per month.

6.2.4 AWS Organizations and Consolidated Billing

The use of AWS Organizations allows organizations with multiple accounts, such as government organizations, to consolidate billing to a single dashboard, while also taking advantage of volume discounts for services including EC2 and S3. For billing purposes, AWS treats all the accounts in the organization as if they were one account.

It is recommended to consider the adoption of AWS Organizations to take advantage of volume discounts and potentially simplify the billing process. At a minimum, the Consolidated Billing features will provide greater insight to organizational costs of running on GovCloud.

6.3 Cost Management

By using Cost Explorer and Cost and Usage Reports in conjunction with organizations, the PMO can easily view and track detailed account-specific spending reports. By utilizing the Cost Optimization Monitor with Kibana dashboards, the team can search and visualize this cost data. To reduce time spent manually monitoring these reports and ensure the team is alerted when thresholds are reached that may lead to unexpected spending, AWS Budgets should be implemented with AWS Organizations so the team can push account-specific alerts to team members through email, Slack, or Amazon Chime. To automatically respond to changes in spending, CloudWatch should be used to trigger workflows such as auto-scaling EC2 instances based on changes in resource demand. Beyond implementing these measures for cost management, the team should also opt into the free Trusted Advisor notifications feature to receive weekly notification emails with updates on potential savings specific to the program architecture for continuous improvement in the future.

7 Conclusion and Next Steps

With an increasing number of emerging Government IT systems being deployed in the cloud, and legacy systems migrating to the cloud, it is vital to architect these systems in a fashion that leverages the scalable nature of the cloud and capitalizes on the opportunities offered by cloud services. There are a number of important design considerations, cost management tools, and programmatic best practices that should be considered when architecting a program in GovCloud or other cloud platforms. The successful implementation of the tools and practices outlined in this report can significantly reduce the cost of running workloads in the cloud, which was conceptually demonstrated in an evaluation of a standard DevSecOps GovCloud account.

A cloud centric deployment of both new systems, and legacy applications being migrated to the cloud, often requires the support and guidance of the PMO to enable greater integration of cloud native services. Often times when deploying both new and legacy systems to the cloud, precedence is given to ensuring the systems are operational, with other considerations such as resource and cost optimization occurring after. When deploying new applications to the cloud, it is recommended that the PMO considers applying optimization strategies such as the ones outlined in this report. Legacy systems which are migrated to the cloud should also be reviewed for potential cloud efficiencies as appropriate.

The most impactful efficiencies and cost strategies discussed in the report include utilizing AWS Organizations to consolidate billing and volume discounts, identifying long-term workloads which can benefit from Reserved Instances, identifying fluctuating workloads which can benefit from Auto-Scaling groups, applying lifecycle management configurations for S3 storage, and monitoring environments with AWS native cost management tools.

Appendix A References

- AWS GovCloud
    - https://aws.amazon.com/govcloud-us/
    - https://docs.aws.amazon.com/govcloud-us/latest/UserGuide/welcome.html
- Figure 4 S3 Object Mobility
    - https://docs.aws.amazon.com/AmazonS3/latest/dev/lifecycle-transition-general-considerations.html
- **Error! Reference source not found.**
    - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-lifecycle.html
- Figure 7 Cost Optimization Monitor architecture
    - https://aws.amazon.com/solutions/cost-optimization-monitor/
- DevOps
    - https://www.business2community.com/cloud-computing/how-to-use-9-cloud-devops-best-practices-for-cost-control-02163854
- Lean Culture
    - https://theleanway.net/The-Five-Principles-of-Lean
- Business Process Re-engineering
    - https://www.ijcaonline.org/archives/volume44/number23/6424-8653
    - https://www.inteqgroup.com/blog/6-key-business-process-reengineering-steps
- Figure 10 Relationship Between Regions and Availability Zones
    - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html
- Security at Scale: Logging in AWS
    - https://d0.awsstatic.com/whitepapers/compliance/AWS_Security_at_Scale_Logging_in_AWS_Whitepaper.pdf
- Laying the Foundation (1/7)
    - https://docs.aws.amazon.com/whitepapers/latest/cost-optimization-laying-the-foundation/introduction.html
- Cost Management in the AWS Cloud (2/7)
    - https://docs.aws.amazon.com/whitepapers/latest/cost-management/introduction.html
- Amazon EC2 Reserved Instances and Other AWS Service Reservation Models (3/7)
    - https://docs.aws.amazon.com/whitepapers/latest/cost-optimization-reservation-models/introduction.html

- Leveraging Amazon EC2 Spot Instances at Scale (4/7)
    - https://docs.aws.amazon.com/whitepapers/latest/cost-optimization-leveraging-ec2-spot-instances/introduction.html
- Creating a Culture of Cost Transparency and Accountability (5/7)
    - https://docs.aws.amazon.com/whitepapers/latest/cost-optimization-transparency-accountability/introduction.html?did=wp_card&trk=wp_card
- Automating Elasticity (6/7)
    - https://docs.aws.amazon.com/whitepapers/latest/cost-optimization-automating-elasticity/introduction.html?did=wp_card&trk=wp_card
- Right Sizing (7/7)
    - https://docs.aws.amazon.com/whitepapers/latest/cost-optimization-right-sizing/introduction.html?did=wp_card&trk=wp_card

Appendix B Cloud Computing

Cloud computing is providing developers and IT departments with the ability to focus on what matters most and avoid undifferentiated work like procurement, maintenance, and capacity planning. Specific user needs require different models and deployment strategies, with each type of cloud service and deployment method providing you with different levels of control, flexibility, and management. Understanding the differences between Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS), as well as what deployment strategies you can use, can help you decide what set of services is right for your needs.

B.1 Compute Models

There are three main models for cloud computing. Each model represents a different part of the cloud computing stack.

B.1.1 IaaS

Infrastructure as a Service contains the basic building blocks for cloud IT and typically provide access to networking features, computers (virtual or on dedicated hardware), and data storage space. Infrastructure as a Service provides you with the highest level of flexibility and management control over your IT resources and is most similar to existing IT resources.

B.1.2 SaaS

Software as a Service provides you with a completed product that is run and managed by AWS. In most cases, people referring to Software as a Service are referring to end-user applications. With Software as a Service, you only need to think about how you will use that particular piece software, not how it is maintained or how the infrastructure is managed.

B.1.3 PaaS

Platforms as a Service remove the need to manage the underlying infrastructure (e.g. hardware and operating systems) and instead focus on the deployment and management of your applications. This increases efficiency as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

B.2 Deployment Models

B.2.1 Cloud

A cloud-based application is fully deployed and run in the cloud, with applications in the cloud either having been created in the cloud or migrated to the cloud from an existing infrastructure. Cloud-based applications can be built on low-level infrastructure pieces or can use higher level services that provide abstraction from the management, architecting, and scaling requirements of core infrastructure.

B.2.2 On Premise

On-premises deployment is used for its ability to provide dedicated resources, but does not provide as many benefits as cloud computing. In most cases this deployment model is the same as legacy IT infrastructure, while using application management and virtualization technologies to maximize utilization of resources. Using virtualization and resource management tools, this process of on-premise deployment is sometimes called the “private cloud”.

B.2.3 Hybrid

A hybrid deployment is a way to connect infrastructure and applications between cloud-based resources and local resources. This hybrid deployment between the cloud and existing on-premises infrastructure extends an organization's infrastructure into the cloud while connecting resources from the cloud to its internal system.

Appendix C AWS Overview

The AWS cloud is hosted worldwide in AWS Regions and Availability Zones (AZ). Each AWS Region is separate consisting of multiple Availability Zones (typically 3). The example program operates within the GovCloud (US-East) region. Each Availability Zone is a fully isolated partition of the AWS infrastructure that consists of discrete data centers which are fully redundant.

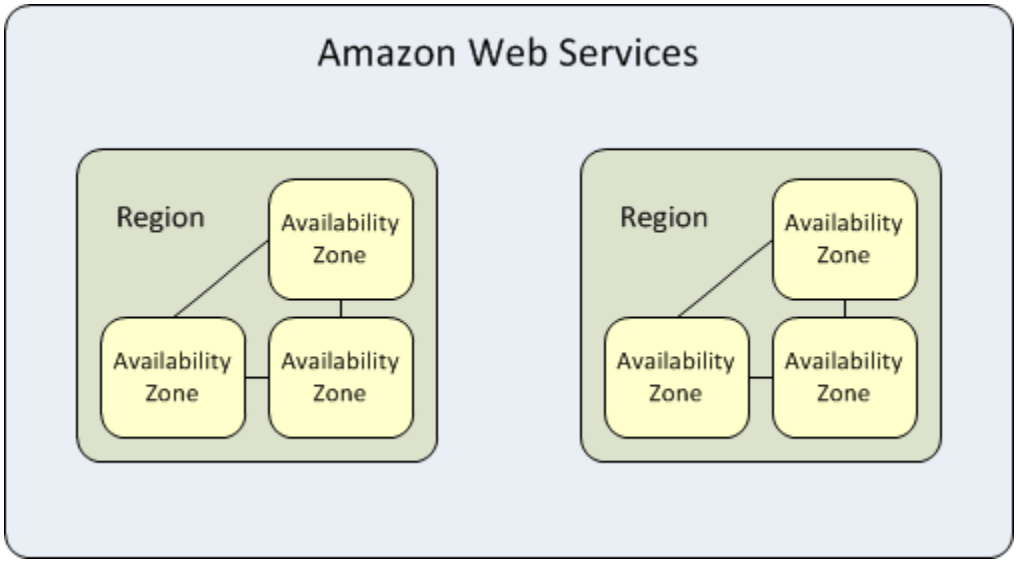

Figure 10 Relationship Between Regions and Availability Zones

Cloud administrators have the option to choose which geographic region and availability zone will host their service (although some resources are region agnostic). Each region has its own unique pricing model per service, and not all regions offer the same services. A cloud administrator chooses which region to deploy services to based on considerations including latency, redundancy, data requirements, service availability, and price.

Appendix D Abbreviations and Acronyms

API	Application Programming Interface
AWS	Amazon Web Services
AZ	Availability Zone
CJIS	Criminal Justice Information Services
COTS	Commercial Off-the-Shelf
CSP	Cloud Service Provider
CUI	Controlled Unclassified Information
DevOps	Development and Operations
DevSecOps	Development Security and Operations
EBS	Elastic Block Store
EC2	Elastic Cloud Compute
FedRAMP	Federal Risk and Authorization Management Program
IaaS	Infrastructure as a Service
IT	Information Technology
PaaS	Platform as a Service
RDS	Relational Database Service
RI	Reserved Instance
S3	Simple Storage Service
SaaS	Software as a Service
SRG	Security Requirements Guide

MITRE