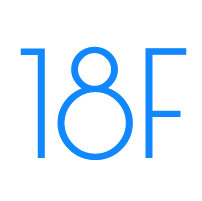
**Deliver To:**

****

Request for Quotation (RFQ)

4QTFHS150004

SIN 132-51

General Services Administration (GSA) Federal Acquisition Service, Integrated Technology Service National IT Commodity Program

401 West Peachtree Street NW, Suite 820 Atlanta, GA 30308

**Created by:**



1900 Campus Commons Drive, Suite 150

Reston, Virginia 20191

703.581.6500

Rich Zimmerman

Contracts Manager

Defender System

Continuous Monitoring

2015

Table of Contents

1.0 Introduction 1

2.0 Continuous Monitoring of Security and Vulnerability Threats 1

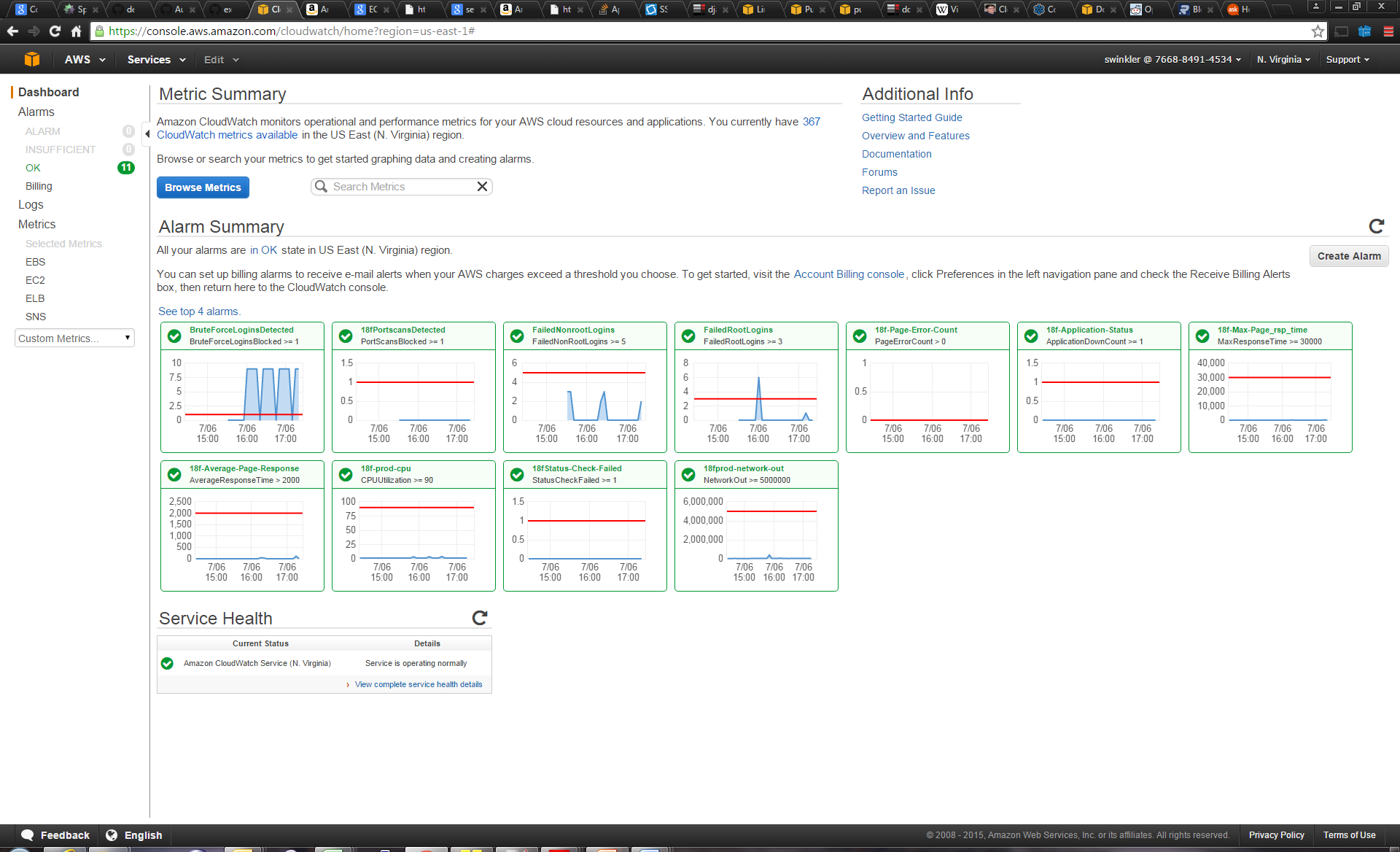
3.0 Performance and Availability Monitoring 2

4.0 Setup and Configuration 3

# Introduction

IntegrityOnePartners’ (IOP) Continuous Monitoring solution was implemented to demonstrate our ability to use automated processes to assess the security risk posture of the prototype environment. Our solution includes features to provide constant visibility into the security, availability, and performance characteristics of the 18F prototype. We used a combination of Linux features and custom scripts to create an automated data feed of security and performance metrics, and used the Amazon CloudWatch API to automatically feed up-to-date metrics to the Amazon CloudWatch console at 5 minute intervals (this can be configured to run at smaller or larger intervals as required). CloudWatch can be used to define alerts and alarms for any reported metric, and to create automated notifications or perform corrective action if an alarm threshold is exceeded. The CloudWatch console is used to display and review metrics, as well as to review alarm status for exceptional conditions or security risks and the efficacy of security controls. For the purposes of Defender, automatic email alerts are generated and sent to relevant DevOps and developer email addresses as alerts are triggered**.**

The high level CloudWatch console with custom metrics implemented appears as follows:



# Continuous Monitoring of Security and Vulnerability Threats

In order to demonstrate our ability to continuously monitor the security risk posture of our prototype, we enabled several features in the prototype and the corresponding metrics and alarms in CloudWatch:

* Brute Force Login Detection - We enabled the ability to detect and report on repetitive automated logins to the target environment, usually generated by scripts or bots attempting random penetration based on detection of an active IP address. In addition, security rules were established to block IPs attempting this type of attack.
* Port Scan Detection - We enabled the ability to detect and report on attempted port scans and block port scanners when detected.
* Failed Root Logins - We enabled the ability to detect and report on failed root logins as recorded in system authorization logs. IPs with repeated failed login attempts are temporarily blocked.
* Failed Non Root Logins - We enabled the ability to detect and report on failed logins to non-administrative accounts as recorded in system authorization logs. IPs with repeated failed login attempts are temporarily blocked.
* Sudo Activity - We enabled the ability to detect and report on use of the linux sudo facility in order to detect and respond to attempts to gain administrative access to resources from non-administrative accounts.
* Page Hit Count - We enabled the ability to detect and report statistics on the number of pages served. Large or unexpected volumes of page hits can be observed and responded to as potential denial of service attack.
* Production Network Throughput - We enabled the ability to detect and report statistics on the amount of outgoing network traffic served by the production server. Large or unexpected spikes in network bandwidth utilization can be observed and responded to as potential denial of service attack.

# Performance and Availability Monitoring

Additionally, we established corresponding metrics and alarms in CloudWatch to monitor the application’s availability and health:

* Application Status - We enabled the ability to detect and report on the operating status of the Defender Application. Should the application stop operating, down operating status metrics are communicated to the CloudWatch console and alarms notifications are sent.
* Server Operating Status - We enabled the ability to detect and report on server problems for the host virtual machine on which the Docker image is deployed. If the host machine goes down or is unreachable, the corresponding operating status metrics are communicated to the CloudWatch console and alarms notifications are sent.
* Page Error Counts - We enabled the ability to detect and report on application errors which result in the return of error pages to users. This information is used to continuously inform support teams if application problems result in substandard user experience and allow support teams to continuously fix and deploy updates to the application.
* Maximum and Average Page Response Times - We enabled the ability to detect and report on average and maximum page response times, allowing support teams to be informed of potentially unacceptable user response times and substandard user experience.
* CPU Utilization - We enabled the ability to detect and report on CPU utilization metrics, allowing support teams to be informed of potentially unacceptable user response times or other server issues which affect the operating status of the hosting platform.

# Setup and Configuration

The following describes how continuous monitoring was setup and configured inside AWS:

1. Make a scripts directory in your home directory:

mkdir ~/scripts

1. Copy the AWS CloudWatch log parsing scripts from the project repository scripts directory to the scripts directory in the home directory:

cp ~/projects/defender/scripts/\*.sh ~/scripts

1. Change directories into “defender”:

cd defender

1. Establish root access:

sudo bash

1. Install bower and grunt-cli node packages as root:

npm install –g bower grunt-cli

1. Exit root user session and change back to default user:

exit

1. Change directories into “src”:

cd src

1. Use bower to download and package client dependencies:

bower install

Type “n” and press “Enter” when asked if bower may use usage statistics

1. Install python-pip as root:

sudo apt-get install python-pip

Type “Y” and press “Enter” when prompted to continue

1. Use pip to install AWS CLI:

sudo pip install awscli

1. On top right of AWS Console, select user profile drop down and click “Security Credentials”.
2. Click “Groups” under “Details” on the left navigation menu.
3. Click the blue button on top labeled “Create New Group”
4. Type “DefenderAdmin” next to Group Name and click the blue button on the bottom right labeled “Next Step”.
5. Select the checkbox to the left of “AdministratorAccess” and click the blue button on the bottom right labeled “Next Step”.
6. Click the blue button on the bottom right labeled “Create Group”
7. Click “Users” under “Details” on the left navigation menu.
8. Click the blue button on top labeled “Create New Users”
9. Type “DefenderUser” next to the first entry under “Enter User Names:” and ensure that the check box is selected to “Generate an access key for each user”.
10. Click the blue button on the bottom right labeled “Create”.
11. Click the drop down blue text labeled “Show User Security Credentials” if you would like to copy the DefenderUser access key ID and Secret Access Key which will be used in upcoming steps. Otherwise, download the credentials by clicking the blue button on the bottom right labeled “Download Credentials” and refer to them from the credentials.csv file that will be downloaded locally through your browser.
12. On top right of AWS Console, select user profile drop down and click “Security Credentials”.
13. Click “Users” under “Details” on the left navigation menu.
14. Highlight the “DefenderUser” by selecting the check box to the left of the user name.
15. Click the white button on top labeled “User Actions” and select “Add User to Groups”.
16. Select the “DefenderAdmin” group by selecting the checkbox to the left of the user name and clicking the blue button on the bottom right labeled “Add to Groups”.
17. Another “User Action” is “Manage Password” where you can set up an optional password to associate with the DefenderUser account.
18. Click “Encryption Keys” on the bottom of the left side navigation menu.
19. Click the blue button on the top of the page labeled “Create Key”.
20. Type “DefenderKey” next to “Alias (required)” and click “Next Step”.
21. Select the “DefenderUser” user name under “Define Key Administrators” and click the blue button on the bottom right labeled “Next Step”.
22. Select the “DefenderUser” user name under “Define Key Usage Permissions” and click the blue button on the bottom right labeled “Next Step”.
23. Click “Finish”
24. Change directories to your home directory:

cd ~

1. Modify the .profile file using vi:

vi .profile

1. Add the following AWS CLI Environment variables according to your security credentials to the end of the .profile file:

export AWS\_ACCESS\_KEY\_ID=<refer to credentials.csv>

export AWS\_SECRET\_ACCESS\_KEY=<refer to credentials.csv>