Read Me

Document Overview

The goal of this Read Me is to provide a fundamental guide to understanding the solution as it has been designed, developed and deployed. This documents intention is to provide a clear explanation into the intended design, choices made in development of the solution and constraints and restrictions within the solution. Provided alongside this information are detailed step by step instructions on how to deploy the solution. Also contained within this document are links and resources used throughout the solutions development, and constructive feedback on how the development of this project went.

Exercise Requirements

The overall goal of this exercise is to create and write a code based deployment of an ElasticSearch service cluster to Amazon Web Services (AWS). The solution submitted should be repeatable, reusable, and secure. At a base level, the provided solution should be able to launch an AWS instance, install ElasticSearch and configure the required security parameters. The ideal solution would also include functionality to scale the ElasticSearch service of at least 3 nodes clustered together.

The completed solution should also come with a README file (this document) that provides instructions to run the solution, provide a description of the solution detailing its components and the choices made for each, a list of resources consulted, and feedback on the exercise including time spent.

The overall solution can utilize any tools, frameworks, or API's, barring of the constraints listed below.

The following constraints have been applied to this exercise as well. They are as follows:

- Must use AWS.
  - Additional AWS services can be utilized but must be detailed in documentation.
- ElasticSearch access and communication must be secure.
  - A self-signed cert is an acceptable level of for the ElasticSearch cluster.
- Utilization of the Amazon ElasticSearch Service is prohibited.

Technical Overview

Now that we have an understanding of what is being requested we can start to dissect the requirements into more manageable pieces, to design what the initial solution could be. Based on the above, we know that the solution is going to need to contain, at the very least, a few AWS technologies. To provide compute resources to the solution we could leverage AWS EC2, for networking we will could utilize AWS VPC, for storage we can utilize AWS S3 or EC2 EBS volumes, and for security we can utilize AWS IAM users and security groups. These AWS Services can provide the foundation for an ElasticSearch application to run on-top of and are a good starting point for moving forward with the design.

Of course, the success of this exercise hinges upon actually getting an ElasticSearch node up and running. So to do so, were going to need to install the ElasticSearch software. ElasticSearch is provided as open source software from the company Elastic and as such we'll be downloading the software we need directly from their repos.

Design Overview

With the above technologies in mind we can begin to formulate a solution that allows us to encapsulate each of the aforementioned AWS services and provision the resources necessary to hosting the ElasticSearch application. Knowing this, the first technology that comes to mind that provides a way for us to automate the provisioning and configuration of the required AWS Services is AWS CloudFormation. AWS CloudFormation will allow for the creation of a stack based on the input of a template file that contains information as to how the resources within each AWS service should be provisioned. The templates used within CloudFormation are by nature, well structured, extensible and reusable, giving great control over what the infrastructure should be doing and for understanding what the expected outcome should be along the way. By using CloudFormation we can launch a stack that contains an EC2 instance to host ElasticSearch, allowing us to provision or utilize a pre-existing VPC and its subnets to provide networking capabilities, and ensure that the necessary security groups and their permissions are applied to the EC2 instance.

With a platform selected to handle the provisioning of the resources need for ElasticSearch we still need a way to install and configure the ElasticSearch software. Well, luckily for us CloudFormation can handle that workload as well, making it an easy choice for the basis of the solution. Within the CloudFormation stack template definitions for the installation, configuration and securing process for ElasticSearch are allowed. By launching the cfn-init utility, provided by AWS under the EC2 Instance userdata properties, we can create metadata information within the EC2 instance creation that contains all of the commands, scripts, files, users, and services needed to bootstrap the ElasticSearch installation process.

Now that we have a process for configuring the AWS resources needed to host ElasticSearch, a way to preform the ElasticSearch installation and configuration, we'll need to find a solution for securing the platform as requested in the requirements. This part seemed pretty straightforward, the ElasticSearch documentation for ElasticSearch version 6.x mentions that all ElasticSearch installations come with x-pack preinstalled. Using the x-pack utility we can easily generate a self-signed certificate to be used in a test environment, and add an administrative user to access the node using local file-based auth. All that we have to do is configure the elasticsearch.yml file to enable x-pack security and configure a file-based authentication realm, and update roles.yml to include an administrative role with complete control over the ElasticSearch node. This can easily be done as a part of the CloudFormation template to be processed during the bootstrap phase.

With all of that said a quick review of the items we've identified so far lead us to the following architecture:

- CloudFormation to provision the underlying AWS resources needed to host the ElasticSearch environment.
- cfn-init utility and appropriate metadata associated with the EC2 instance configuration to install and configure ElasticSearch.
- x-pack to provide ElasticSearch security, including SSL and user based authentication.

Design Details

- How did you choose to automate the provisioning and bootstrapping of the instance? Why?

As detailed above I decided to use CloudFormation to automate the provisioning of the AWS resources and to bootstrap the instance with the installation and configuration of ElasticSearch and x-pack. I choose CloudFormation based mostly on its ease of use and connectivity into pretty much every AWS service. I thought about using OpsWorks to handle the bootstrapping phase of this solution but after delving into things a bit more I came to the conclusion that it was simply just an extra layer of complexity on what is, relatively, a simple task. I wanted to be able to focus on answering the requirements as best as I could without getting to off track and down a rabbit hole with a technology i'm not completely familiar with.

- How did you choose to secure ElasticSearch? Why?

The requirements for this exercise asked for two things specific to security on the ElasticSearch deployment, a self-signed certification for encrypted communication to the node, and credential based authentication to the node. So starting with the self-signed certificate, I used elasticsearch-certutil to generate a self-signed test certificate to be used by the ElasticSearch application. Now, I've worked with ElasticSearch installation before specifically in demonstration and lab environments so I've configured ElasticSearch to use file-based auth before. I knew that we could easily configure the elasticsearch.yml file to require authentication and the roles.yml file to include a simple administrative role that has master permissions across ElasticSearch and all its indexes. Once I had these files created properly all I needed to do was run the elasticsearch-users command to generate an administrative user for use to access the node. I choose this direction for securing the node because one, it is built in functionality within ElasticSearch and two it felt like the most direct and easiest way for me to meet the requirements in a concise manner.

- How would you monitor this instance? What metrics would you monitor?

I think CloudWatch is a decent starting point for monitoring this as a stand-alone test environment. The problem with CloudWatch is it doesn't monitor all of the metrics that we would want to see when keeping an eye on an ElasticSearch installation. The basic CloudWatch metrics only allow us to see CPU utilization, Disk usage and disk I/O, and network usage and I/O. Obviously, with something like ElasticSearch we would also want to be able to view memory utilization and service status for ElasticSearch. If I were to add monitoring to this environment I'd either create a custom plugin to poll those metrics on the EC2 host or use a third-party service to integrate into this solution to capture the metrics and provide them into a centralized monitoring platform.

- Could you extend your solution to launch a secure cluster of ElasticSearch nodes? What would need to change to support this use case?

I'm fairly confident this solution could be scaled out to support a clustered installation of ElasticSearch. Now, I would not say that this solution would provide a truly "secured" cluster environment of ElasticSearch without some fairly drastic overhaul to the security requirements. To get this to work we'd need to do a couple things to the ElasticSearch installation and to the CloudFormation template. First, we'd need to configure the CloudFormation template to launch this ElasticSearch EC2 instance into an auto-scaling group with a maximum and minimum size set to the number of nodes needed within the cluster. We'd also need to include a couple of different items as part of the auto-scaling group template. We'd need to add a launch configuration based on the EC2 instance we originally deploy as a part of the CloudFormation

Stack and include the user-data used to configure elasticsearch on that EC2 instance. Now, I will be honest I have no idea how that works when we use cfn-init to bootstrap that instance? I would have to do more research into whether or not that is a possible solution. Either way, if that IS possible, we'd then need to ensure that the AWS EC2 cloud plugin is installed as a part of that bootstrap and then configure the ElasticSearch elasticsearch.yml file to include the AWS discovery settings so each node can see, and communicate with each other upon creation.

- Could you extend your solution to replace a running ElasticSearch instance with little or no downtime? How?

Currently, no. I don't see a way that you could launch this as a part of another ElasticSearch cluster and have everything go smoothly. The node *might* start indexing and sharding if it had the AWS EC2 cloud plugin installed and the EC2 instance was launched into the same subnet as the existing cluster with open security on the ports ElasticSearch uses to communicate with nodes but thats a lot of ifs and bad practices. A lot would need to be done to migrate indexes over to this host. For starters, at the very least, we would need to work on changing the ElasticSearch configuration to match that of the existing cluster. But to my understanding thats not even scratching the tip of a much larger iceberg that would need to be tackled. (Not to mention the security implications from launching something like this into even a staging environment)

- Was it a priority to make your code well structured, extensible, and reusable?

It was one of the main reasons why I decided upon using CloudFormation. Unfortunately, due to time constraints, limitations of the exercise and a drive to meet the complete set of requirements as best as I could, the finished result didn't turn out to be as well structured, extensible or reusable as I would of hoped. Quite a few shortcuts were made to allow this to fit within the constraints of the exercise that I don't think would be reliably reusable or allow this to be expanded upon easily.

- What sacrifices did you make due to time?

I took quite a few shortcuts in the configuration of ElasticSearch due to time constraints. For instance, the StartES configSet exists entirely because I need to initiate the trial version of x-pack to enable security in ElasticSearch and would not be something I would want to do in any sort of production or even a "real" development environment. I also wasn't able to meet the extra credit ask as well. I had some fairly "cheap" ideas with how to do it initially, like simply adding three EC2 instance creations as a part of the CloudFormation Templates, but they felt sort of counter productive to the idea of this exercise and I decided to scrap them for a more complete answer to the main requirements as they were asked. I also would have like to have done more documentation on *how* I did everything. I'm a fairly firm believer in that build guides should be a thing and I would like to have a detailed description that breaks down the Template and explains not just what I'm doing but *why* I'm doing it. I personally believe theres a lot of value add to be had there.

Build Guide

The easiest way to deploy this solution would be to utilize the AWS web GUI for CloudFormation as there are a few parameters that need to be passed to the CloudFormation Stack to be used within the template.

To do so:

1. Log into the AWS web console
2. Navigate to CloudFormation under MangementTools
3. Select Create Stack
4. Under “Choose a Template” select “Specify an Amazon S3 template URL”
5. Paste the following link:
    1. https://s3.us-east-2.amazonaws.com/alexsalsifycode/elasticsearch.yaml
6. Select Next
7. Enter a Stackname (can be whatever you desire it to be)
8. Select a pre-existing EC2 keypair to be used with the EC2 instance to SSH into it
9. Enter in a Password to be used throughout the script.
    1. Will need this later to authenticate to the ElasticSearch Node
10. Select a subnet to deploy the EC2 instance too
11. Select the VPC that will be used
12. Select Next
13. Select Next
    1. If you would like to add tags to the CloudFormation stack nows the time. We are not using any IAM roles for this or enabling monitoring/rollback functionality.
14. Review Settings and select create
15. Wait a few minutes for the Stack to finish creating the EC2 instance
    1. Because we used x-pack to secure the node there are some userdata commands that are run after cfn-init. So the Stack will come back in a ready state before we’ve finished applying the security settings and restarting the ElasticSearch Service. Just giver her a minute to do her thing.
16. Navigate to the Outputs tab of the Stack you just create and copy the InstancePublicIP value.
17. Open a new tab in your web browser and navigate to https://InstancePublicIP:9200
18. Accept the certificate
19. Log into ElasticSearch with:
    1. Username: admin
    2. Password: “Password Specified as a part of the CloudFormation Stack creation”
20. Viola, ElasticSearch on an EC2 instance.

Feedback

All together I would say this was a fun exercise for me personally. I felt like the overview of the requirements for what you’re looking for were a little bit vague but in reality I can understand how that vagueness can be an avenue to open up discussion and leave things up to interpretation. Regardless, I certainly learned a lot in my time spent working on this.

In total I spent about 12-16 hours working on this project give or take. I will note that a solid portion of that time was being frustrated with yaml formatting and having to read through manuals to ensure my syntax was correct.

Resources

https://aws.amazon.com/cloudformation/aws-cloudformation-templates/

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-init.html#aws-resource-cloudformation-init-syntax

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/deploying.applications.html https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-init.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-signal.html

https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html