Escuela Universitaria de Ingeniería Técnica de Telecomunicación La Salle

Trabajo Final de Máster

Máster Universitario en Ingeniería de Telecomunicaciones

Cloud Analytics and Security Automation with Ansible

Alumno	Profesor Ponente
Antón Elías Paz De Paz	Adrià Castany Mirats

ACTA DEL EXAMEN
DEL TRABAJO FINAL DE MASTER

Reunido el Tribunal calificador en el día de la data, el alumno

D. **Antón Elías Paz De Paz**

expuso su Trabajo de Final de Grado, el cual trató sobre el siguiente tema:

Cloud Analytics and Security Automation with Ansible

Acabada la exposición y contestadas por parte del alumno las objeciones formuladas por los Sres. Miembros del tribunal, este valorará el mencionado trabajo con la calificación de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENTE DEL TRIBUNAL

Resumen

La demanda de recursos instantáneos y temporales en los departamentos de IT de las empresas ha incrementado a lo largo de los años. Esta necesidad ha hecho que las empresas no inviertan en tener sus servidores, servicios y aplicaciones en lo que se denomina *"on premise"* y cada vez son más empresas que utilizan los "*cloud"* públicos para albergar y mantener su infraestructura.

Esta necesidad ha hecho que la forma en que se despliegan los servicios haya visto una evolución hacia la automatización del despliegue y configuración de servicios, todo desde un "*click*.

En este proyecto se explicará como desplegar, de manera automatizada, un sistema de monitorización para la empresa (ELK), la cual estará encargada de la parte de analítica, el despliegue de un gestor de accesos (AM) para los recursos y una aplicación/servicio (Wordpress/Apache).

Todo este despliegue automático se hará mediante el uso de Ansible como herramienta de automatización y el "*cloud"* público de AWS.

Resum

La demanda de recursos instantanis i temporals en els departaments de IT de les empreses ha incrementat al llarg dels anys. Aquesta necessitat ha fet que les empreses no inverteixin a tenir els seus servidors, serveis i aplicacions en el que es denomina “on premise” i cada vegada són més empreses que utilitzen els “cloud” públics per a albergar i mantenir la seva infraestructura.

Aquesta necessitat ha fet que la forma en què es despleguen els serveis hagi vist una evolució cap a l'automatització del desplegament i configuració de serveis, tot des d'un “clic.

En aquest projecte s'explicarà com desplegar, de manera automatitzada, un sistema de monitoratge per a l'empresa (ELK), la qual estarà encarregada de la part d'analítica, el desplegament d'un gestor d'accessos (AM) per als recursos i una aplicació/servei (Wordpress/Apatxe).

Tot aquest desplegament automàtic es farà mitjançant l'ús de Ansible com a eina d'automatització i el “cloud” públic de AWS.

Abstract

The demand for instant and temporary resources in the IT departments of companies, has seen an increased over the years. This need has meant that companies do not invest in having their servers, services and applications in what is called “on premise” and more and more companies are using public clouds to host and maintain their infrastructure.

This need has meant that the way services are deployed has seen an evolution towards the automation of the deployment and configuration of services, all from a click.

This project will explain how to automatically deploy a monitoring system for the company (ELK), which will oversee the analytics part, the deployment of an access manager (AM) to secure resources within a company and an application / service (Wordpress / Apache).

All of the above will be an automated deployment using Ansible as the automation tool and the AWS cloud as the public cloud.

INDEX

PALABRAS CLAVE

AWS - es una subsidiaria de Amazon que ofrece API y plataformas de computación en la nube bajo demanda a individuos, empresas y gobiernos, con un sistema de pago por uso medido.

ElasticSearch - es un motor de búsqueda RESTful de código abierto construido sobre Apache Lucene y publicado bajo una licencia de Apache. Está basado en Java y puede buscar e indexar archivos de documentos en diversos formatos.

Kibana - es una herramienta de exploración y visualización de datos de código abierto que se utiliza para el análisis de registros y series de tiempo, el monitoreo de aplicaciones y los casos de uso de inteligencia operativa. Ofrece funciones potentes y fáciles de usar, como histogramas, gráficos de líneas, gráficos circulares, mapas de calor y soporte geoespacial integrado.

Logstash - es una herramienta de código abierto para recopilar, analizar y almacenar registros para uso futuro.

ELK - es el acrónimo de tres proyectos de código abierto: Elasticsearch, Logstash y Kibana.

AM - la gestión de acceso se refiere a los procesos y tecnologías utilizados para controlar y monitorear el acceso a la red. Funciones de gestión de acceso, como autenticación, autorización, auditoría de confianza y seguridad,

Redis - es un almacén de estructura de datos en memoria de código abierto que se utiliza como base de datos, caché y agente de mensajes.

FileBeat - es un cargador ligero para reenviar y centralizar datos de registro.

Tokens - token de seguridad o token de hardware, token de autenticación o token criptográfico, un dispositivo físico para la autenticación de la computadora

LDAP - (Protocolo ligero de acceso a directorios) es un protocolo de Internet que se utiliza para buscar datos desde un servidor. Este protocolo abierto se utiliza para almacenar y recuperar información de una estructura de directorio jerárquica denominada árbol de información de directorio.

VPC - es una red virtual dedicada a su cuenta de AWS. Está lógicamente aislado de otras redes virtuales en la nube de AWS.

VPN - una red privada virtual (VPN) le brinda privacidad y anonimato en línea al crear una red privada desde una conexión pública a Internet.

IGW - una puerta de enlace de Internet (IGW) permite que los recursos de su VPC accedan a Internet,

SSH - es un software estándar para admitir la transferencia de datos cifrados entre dos computadoras. Se puede utilizar para admitir inicios de sesión seguros, transferencias de archivos o conexiones de propósito general

Tomcat - es un servidor de aplicaciones diseñado para ejecutar servlets Java y representar páginas web que utilizan la codificación de páginas del servidor Java.

Squid - es un proxy web HTTP de almacenamiento en caché y reenvío. Tiene una amplia variedad de usos, incluida la aceleración de un servidor web mediante el almacenamiento en caché de solicitudes repetidas, almacenamiento en caché web, DNS ...

Playbook - es una unidad organizada de scripts que define el trabajo para una configuración de servidor administrada por la herramienta de automatización Ansible.

PARAULES CLAU

AWS - és una subsidiària de Amazon que ofereix API i plataformes de computació en el núvol sota demanda a individus, empreses i governs, amb un sistema de pagament per ús mesurat.

ElasticSearch - és un motor de cerca RESTful de codi obert construït sobre Apatxe Lucene i publicat sota una llicència d'Apatxe. Està basat en Java i pot buscar i indexar arxius de documents en diversos formats.

Kibana - és una eina d'exploració i visualització de dades de codi obert que s'utilitza per a l'anàlisi de registres i sèries de temps, el monitoratge d'aplicacions i els casos d'ús d'intel·ligència operativa. Ofereix funcions potents i fàcils d'usar, com a histogrames, gràfics de línies, gràfics circulars, mapes de calor i suport geoespacial integrat.

Logstash - és una eina de codi obert per a recopilar, analitzar i emmagatzemar registres per a ús futur.

ELK - és l'acrònim de tres projectes de codi obert: Elasticsearch, Logstash i Kibana.

AM - la gestió d'accés es refereix als processos i tecnologies utilitzats per a controlar i monitorar l'accés a la xarxa. Funcions de gestió d'accés, com a autenticació, autorització, auditoria de confiança i seguretat,

Redis - és un magatzem d'estructura de dades en memòria de codi obert que s'utilitza com a base de dades, caixet i agent de missatges.

FileBeat - és un carregador lleuger per a reexpedir i centralitzar dades de registre.

Tokens - token de seguretat o token de maquinari, token d'autenticació o token criptogràfic, un dispositiu físic per a l'autenticació de la computadora

LDAP - (Protocol lleuger d'accés a directoris) és un protocol d'Internet que s'utilitza per a buscar dades des d'un servidor. Aquest protocol obert s'utilitza per a emmagatzemar i recuperar informació d'una estructura de directori jeràrquica denominada arbre d'informació de directori.

VPC - és una xarxa virtual dedicada al seu compte de AWS. Està lògicament aïllat d'altres xarxes virtuals en el núvol de AWS.

VPN - una xarxa privada virtual (VPN) li brinda privacitat i anonimat en línia en crear una xarxa privada des d'una connexió pública a Internet.

IGW - una porta d'enllaç d'Internet (IGW) permet que els recursos del seu VPC accedeixin a Internet,

SSH - és un programari estàndard per a admetre la transferència de dades xifrades entre dues computadores. Es pot utilitzar per a admetre inicis de sessió segurs, transferències d'arxius o connexions de propòsit general

Tomcat - és un servidor d'aplicacions dissenyat per a executar servlets Java i representar pàgines web que utilitzen la codificació de pàgines del servidor Java.

Squid - és un proxy web HTTP d'emmagatzematge en caixet i reexpedició. Té una àmplia varietat d'usos, inclosa l'acceleració d'un servidor web mitjançant l'emmagatzematge en caixet de sol·licituds repetides, emmagatzematge en caixet web, DNS ...

Playbook - és una unitat organitzada de scripts que defineix el treball per a una configuració de servidor administrada per l'eina d'automatització Ansible.

KEYWORDS

AWS – is a subsidiary of Amazon providing on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis.

ElasticSearch – is an open source, RESTful search engine built on top of Apache Lucene and released under an Apache license. It is Java-based and can search and index document files in diverse formats.

Kibana – is an open-source data visualization and exploration tool used for log and time-series analytics, application monitoring, and operational intelligence use cases. It offers powerful and easy-to-use features such as histograms, line graphs, pie charts, heat maps, and built-in geospatial support.

Logstash – is an open source tool for collecting, parsing, and storing logs for future use.

ELK – is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana.

AM – Access management refers to the processes and technologies used to control and monitor network access. Access management features, such as authentication, authorization, trust and security auditing,

Redis – is an open source, in-memory data structure store, used as a database, cache and message broker.

FileBeat – is a lightweight shipper for forwarding and centralizing log data.

Tokens – Security token or hardware token, authentication token or cryptographic token, a physical device for computer authentication

LDAP – (Lightweight Directory Access Protocol) is an internet protocol, which is used to look up data from a server. This open protocol is used to store as well as retrieve information from a hierarchical directory structure called as directory information tree.

VPC – is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS Cloud

VPN – A **virtual private network** (VPN) gives you online privacy and anonymity by creating a private network from a public internet connection.

IGW – An Internet Gateway (**IGW**) allows resources within your VPC to access the internet,

SSH – is a software standard to support encrypted data transfer between two computers. It can be used to support secure logins, file transfers or general purpose connects

Tomcat – is an application server designed to execute Java servlets and render web pages that use Java Server page coding.

Squid – is a caching and forwarding HTTP web proxy. It has a wide variety of uses, including speeding up a web server by caching repeated requests, caching web, DNS …

Playbook – is an organized unit of scripts that defines work for a server configuration managed by the automation tool Ansible.

1 INTRODUCTION

The demand for instant and temporary resources in the IT departments of companies, has seen an increased over the years. This need has meant that companies do not invest in having their servers, services and applications in what is called “on premise” and more and more companies are using public clouds to host and maintain their infrastructure.

Today, we hear the word of Start-up almost in every conversation and news regarding entrepreneurship or job scouting. Creating a company has become easier with the existence of clouds services or IaaS, where now any small company doesn’t need to rely solely on pure iron investment to build their IT infrastructure, now they can pay for what they are using and need at a certain moment, without the need to invest larges amount of money upfront. This fact coupled with the existence of new automation tools have created the perfect environment for automation cloud deployments and DevOps methodology.

When it comes to DevOps, automation provides the means to develop and deploy faster while maintaining or even improving quality. It does this not by replacing the need for human labor but by amplifying the performance of individuals by taking over the many dull, repetitive tasks that occupy their time. It also helps by reducing the friction that arises when key, interrelated tasks are not orchestrated properly.

During this project I will explain a possible architecture that could be used for a Start-up to even a full-fledged company since all the components can easily be scalable to meet the needs of the growing company. How is this going to be scalable? Well, the first thing of the bat is that we will be deploying all the services in AWS cloud, this will help us with the scalability issue regarding “iron”, then with the correct topology, we will be able to scale it logically.

The cloud architecture that I will be explaining could be a perfect valid architecture for a SaaS company, where they will offer a service to the public internet. This means that as a company we will want to secure this resource and also monitor it for future problems or statistics. We will be using some free open source applications and some proprietary applications, that for a commercial release, we should buy the needed license to be fully compliant.

This cloud architecture as mention before, will be composed of a service, a monitoring system, and a security tool to protect the resources that are open to the public internet. This doesn’t mean that we will be only installing three applications, since we will be needing many more in order to make our cloud infrastructure sealed to the outside world, with only a few points of entry.

All of this will be deployed and configured using an automation tool called Ansible. This is an open source tool that does not require any agent on the client side making it an easy tool that only needs an SSH connection to the instances.

2 MOTIVATION AND OBJECTIVES

I do not think much explanation is required for the answer to this question in this automation era. Anywhere we go, we see things that are being automated, either with minimal or no human intervention at all.

The objectives of this project or PoC is to deploy an infrastructure In AWS that will allow any service or resource to be closely monitored and secured. This will all be done with a DevOps methodology using Ansible as the deployment tool.

The deployment will consist of an ELK Stack and AM6.5 as the access manager. The ELK stack will have filebeat as the log shipping service and a Redis buffer before the Logstash for higher throughput deployments.

The outcome of this project is to have a fully working playbook that can be reused for future deployments and as starting point for other deployments.

Clearly, we know by now that automation removes manual errors, dependency on an individual, performs faster, and achieves accuracy thereby achieving consistency and reliability. Hence, automating everything enables the devops objective of high-quality delivery, enables frequent releases and faster releases. This is the main motivation gear of this project, to benefit from the time developing this playbook as an excuse to further improve my knowledge of architecture design, cloud architecture, AWS and many more applications and services that I never had the time needed to get myself into mud and figure everything out.

"Automation applied to an inefficient operation will magnify the inefficiency." – Bill Gates

3 PoC topology Overview

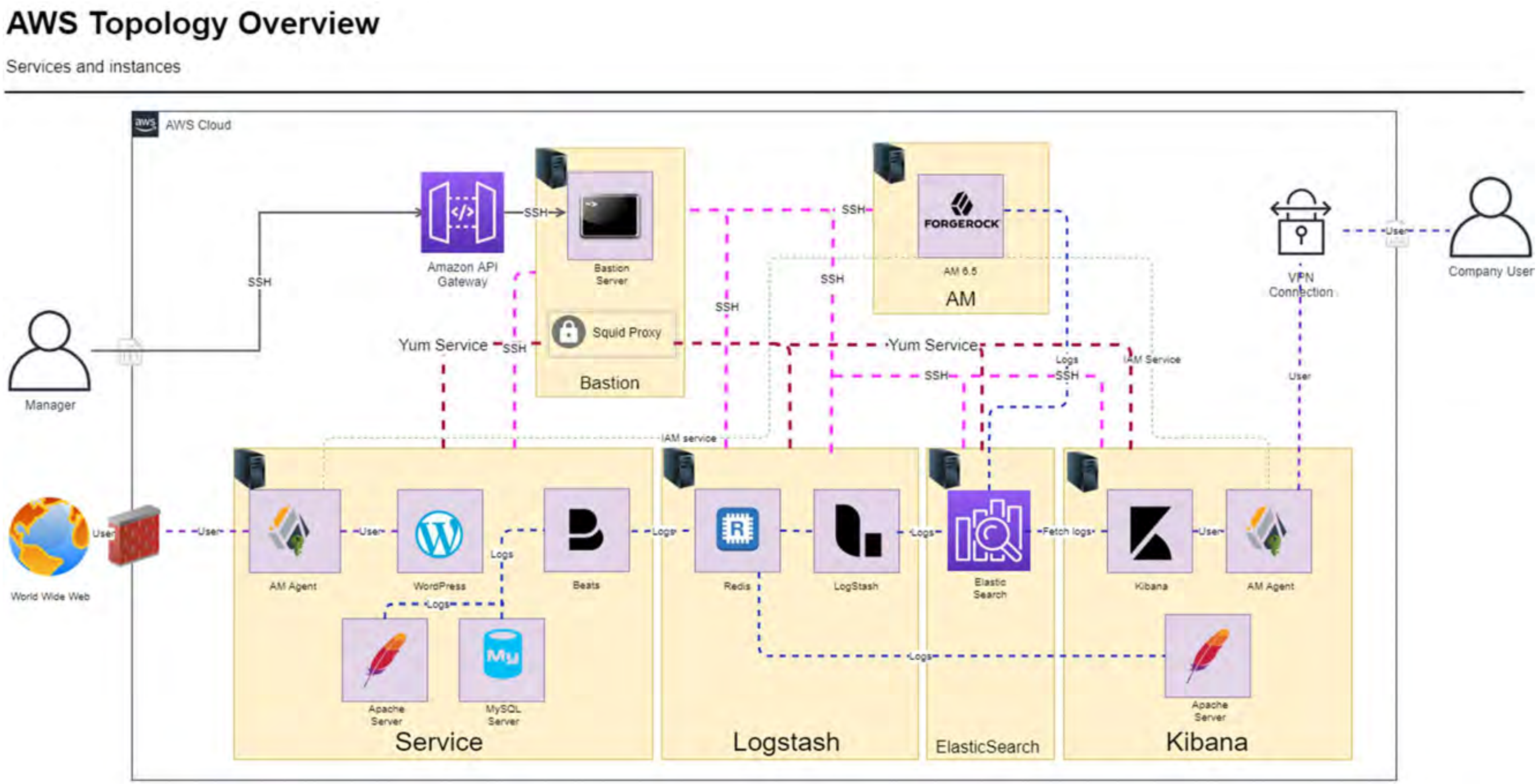


Figure 1 - AWS Topology Overvie

3.1 PoC Deployment Summary

As you can see in the previous figure, the topology is composed of the ELK stack (ElasticSearch, Logstash & Kibana), AM, a bastion, and a service.

All of these services will be deployed in AWS instances, each with different specifications, through an Ansible playbook that will automatize the deployment via a script written in Yaml language.

The first instance that must be deployed is the bastion. In this design, it will play 3 roles, roles that should be separated in different instances but for a PoC this will keep our AWS costs low. The bastion will work as a jump server for the manager, this means it will contain all the SSH keys to access the rest of the instances; Ansible's bastion where most of the code will be executed; and as a reverse proxy for the yum service so there is only one door for management traffic.

Regarding the service, a Wordpress server has been configured. This service needs an Apache and a MySQL server. All these services will log all the activities that are occurring in the server, those logs will be sent by the log shipping service to the Redis buffer in the Logstash instance.

Once the logs are collected by Filebeat, they are received by the Redis service that acts as a buffer for the Logstash input. Here the Logstash will ingest data from the Redis service, transforms it, and then sends it to the ElasticSearch instance.

The last step in the monitorization stack is the monitorization of the logs and this will be accessible through the dashboard provided by Kibana. This dashboard can be manually configured and customized.

Regarding security, an access manager will be deployed, this will protect the Wordpress admin webpage and the needed resources. This is done by deploying an agent in each of the instances that need to be protected. The agent will redirect the authentication to the AM server which will manage the user authentication and authorization. Once the user has authenticated with Forgerock's AM, the user will have a session cookie or token that will allow them to access the resource.

A VPN guide will also be explained to provide access to inside employees or costumers depending on the need of the architecture. This is the only step that will not be fully automatized since Ansible fully depends on a series of modules, that interact with the AWS api, that are yet to be developed.

During this PoC I will not be configuring nor explaining the firewall that needs to be in the perimeter of the network, this is due to that the Firewall automation and configuration its outside of the project scope.

4 AWS CLOUD INFRASTRUCTURE

4.1 NETWORKING

Until today the networking aspect of any design would depend on how much “iron” (routers, switches) you could have at your disposal. This would mean that if you needed to add more instances or create new subnets, you would have to configure by hand these networking devices. With cloud networking all of these problems and more are no longer an issue since cloud providers provide all of these configurations in an easy and accessible manner.

Amazon Web Services (AWS) provides the Networking tools and resources that enable you to securely connect to the cloud and control, isolate and distribute your applications.

4.1.1 VPCs [1]

A virtual private cloud (VPC) is a virtual network linked to your AWS account that is logically isolated from other VPCs. These VPC can be seen as a private network where you can have subnets, security groups, configure route tables and assign Ips.

As a company grows the need for greater segmentation of services becomes necessary to monitor costs, control access, and provide easier environmental management. A multi-VPC solution solves these issues by providing specific accounts for IT services and users within an organization.

In our case, Figure 2, we have defined two VPCs one for the Management network, where the bastion resides, and the other for the ELK stack and application. For future development or in case of applying this to a real environment, for further segmentation, it would be wise to separate the latter into two other VPCs and create a VPC for the company users that connect through a VPN. Refer to Figure 3.

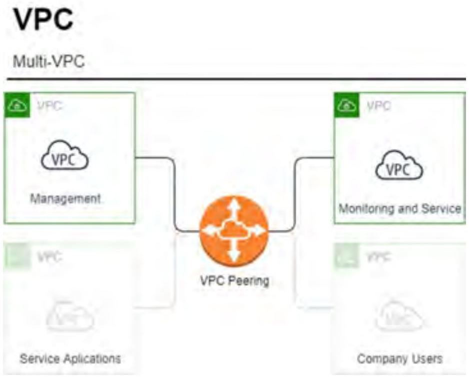


Figure 2 - Multi-VPC in our project

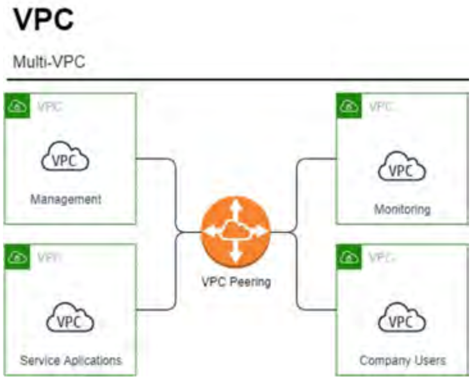


Figure 3 - Multi-VPC

4.1.1.1 VPC Peering

The simplest way to connect two VPCs is to use VPC Peering. With this setup, a connection enables full bidirectional connectivity between the VPCs which routes traffic between the VPCs.

A VPC peering is a point-to-point connectivity, but it has its drawbacks. For example, if you have a VPC peering connection between VPC A and VPC B and between VPC A and VPC C, an instance in VPC B cannot transit through VPC A to reach VPC C. To route packets between VPC B and VPC C, you are required to create another VPC peering connection between them, making this VPC peering not very scalable due to the mesh topology that it creates. In AWS the peering connections have been limited to 125.

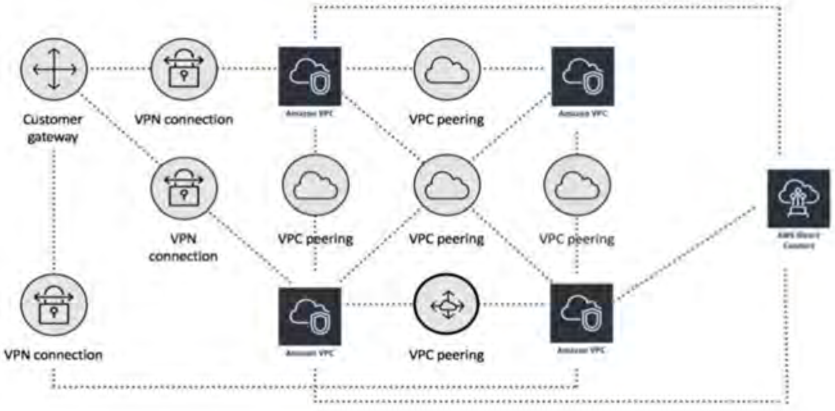


Figure 4 - VPC peering

When should we use VPC peering? VPC peering is best used when resources in one VPC must communicate with resources in another VPC, the environment of both VPCs is controlled and secured, and the number of VPCs to be connected is less than 10.

4.1.1.1.1 Transit Gateway

Transit Gateways fill the scalability gap that VPC peering creates, enabling you to route traffic through "peering connections" and it also provides the possibility to merge all traffic that is bound to the internet and only create one entry and exit point in your architecture. The Transit Gateway controls how traffic is routed among all the connected networks using route tables simplifying management and reduces operational costs.

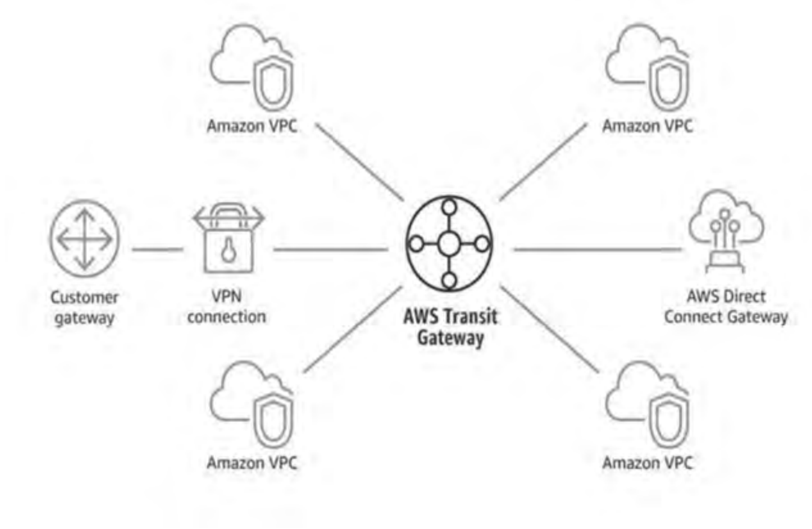
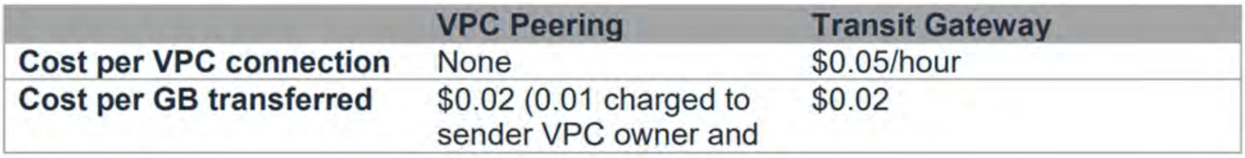


Figure 5 - Transit Gateways

The only downside to this technology is the cost associated to the technology and its limitations: *"Transit Gateway solves the complexity involved with creating and managing multiple VPC peering connections at scale. While this makes TGW a good default for most network architectures, VPC peering is still a valid choice due to the following advantages it has over TGW:"*



	VPC Peering	Transit Gateway
Cost per VPC connection	None	$0.05/hour
Cost per GB transferred	$0.02 (0.01 charged to sender VPC owner and	$0.02

Figure 6 - VPC Peering vs Transit Gateway

4.1.1.2 Subnets [2]

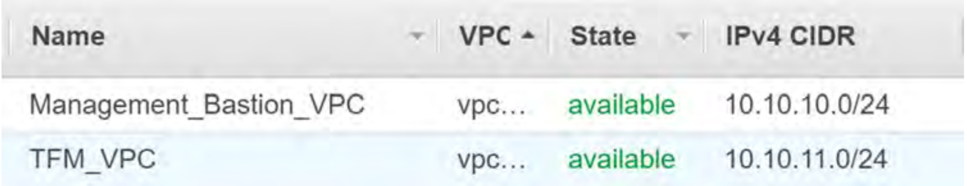
"A subnet is a range of IP addresses in your VPC. You can launch AWS resources into a specified subnet. Use a public subnet for resources that must be connected to the internet, and a private subnet for resources that won't be connected to the internet."

As any subnet, they work the same way but in AWS they have a few more reserved IPs than in what you would normally have in your home:

- `10.0.0.0`: *Network address.*
- `10.0.0.1`: *Reserved by AWS for the VPC router.*

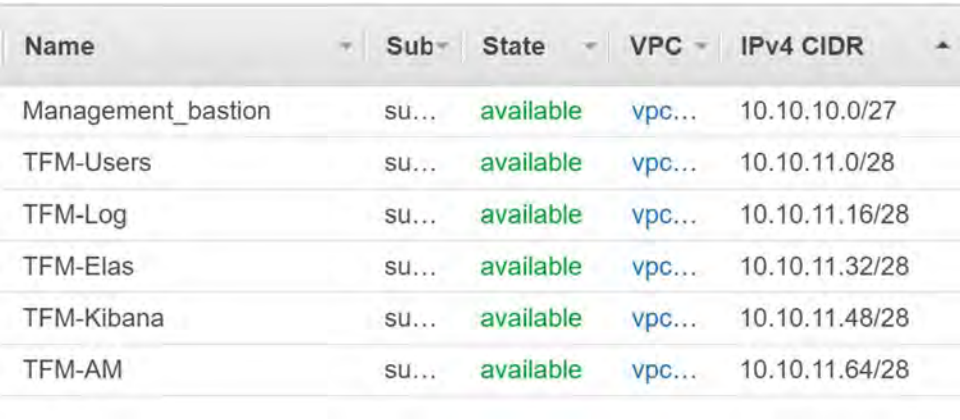
- `10.0.0.2`: *Reserved by AWS. The IP address of the DNS server is the base of the VPC network range plus two. For VPCs with multiple CIDR blocks, the IP address of the DNS server is located in the primary CIDR.*
- `10.0.0.3`: *Reserved by AWS for future use.*
- `10.0.0.255`: *Network broadcast address. We do not support broadcast in a VPC, therefore we reserve this address.*

In our scenario we have divided the subnets as follow:



Name	VPC	State	IPv4 CIDR
Management_Bastion_VPC	vpc…	available	10.10.10.0/24
TFM_VPC	vpc…	available	10.10.11.0/24

Figure 7 - VPCs in project



Name	Sub	State	VPC	IPv4 CIDR
Management_bastion	su…	available	vpc…	10.10.10.0/27
TFM-Users	su…	available	vpc…	10.10.11.0/28
TFM-Log	su…	available	vpc…	10.10.11.16/28
TFM-Elas	su…	available	vpc…	10.10.11.32/28
TFM-Kibana	su…	available	vpc…	10.10.11.48/28
TFM-AM	su…	available	vpc…	10.10.11.64/28

Figure 8 - Subnets in project

4.1.1.3 Internet Gateway IGW [3]

“An internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between your VPC and the internet.”

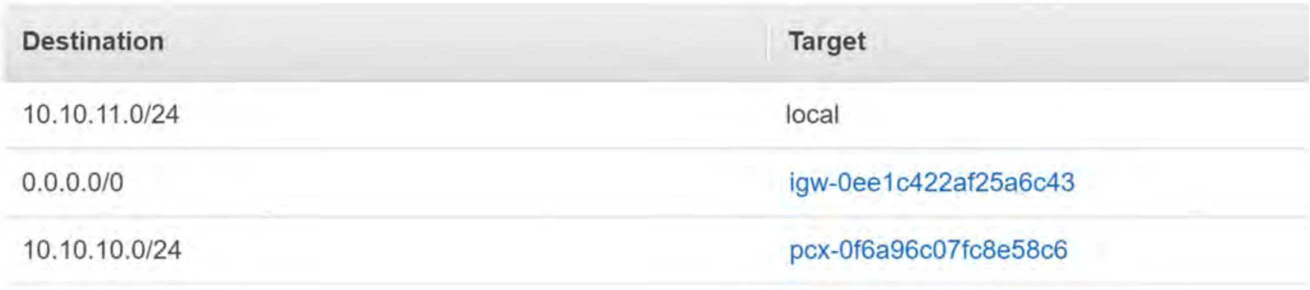
An internet gateway serves two purposes: to provide a target in your VPC route tables for internet-routable traffic, and to perform network address translation (NAT) for instances that have been assigned public IPv4 addresses.

In order for any possible communication with internet, each VPC has internet gateway in order to use it afterwards with the Elastic IP addresses.

4.1.1.4 Route Tables [4]

How does all the routing happen in the VPCs? AWS VPCs have implicit routers that use route tables to control where network traffic is directed. Each subnet in your VPC must be associated with a route table, which controls the routing for the subnet (subnet route table) but also you can explicitly associate a subnet with a particular route table. Otherwise, the subnet is implicitly associated with the main route table. A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same subnet route table.

The route tables inside the same VPC automatically provides connectivity between subnets from the same VPC, but if that VPC contains an IGW or a VPC peering, this should be added in the route table in order to provide the desired connectivity.



Destination	Target
10.10.11.0/24	local
0.0.0.0/0	igw-0ee1c422af25a6c43
10.10.10.0/24	pcx-0f6a96c07fc8e58c6

Figure 9 - Route table with IGW and VPC peering

4.1.2 Elastic Network Interfaces [5]

Elastic Network Interfaces are network interfaces that are created and associated to an instance, this means that you can remove this interface and attach it to another instance. This proofs really useful since network interfaces is where you attached the security groups desired. This means that all the rules associated to ports are attached to the interface and not the instance itself, allowing you to keep the configuration of the interface even if you terminate the instance.

A network interface can include the following attributes:

- A primary private IPv4 address from the IPv4 address range of your VPC
- One or more secondary private IPv4 addresses from the IPv4 address range of your VPC
- One Elastic IP address (IPv4) per private IPv4 address
- One public IPv4 address
- One or more IPv6 addresses
- One or more security groups
- A MAC address

- A source/destination check flag
- A description

4.1.3 Elastic IP Addresses [6]

In AWS in order to access the Internet, you will have to assign an Elastic IP address is a public IPv4 address. If your instance does not have a public IPv4 address, you can associate an Elastic IP address with your instance to enable communication with the internet. For example, this allows you to connect to your instance from anywhere inside the topology.

As mentioned above, this address will enable the instance to exit to the internet. This is only possible if the VPC has configured an Internet Gateway.

In order for the bastion and the service to operate, each of the instances will need an EIP with different ingress and egress rules. Luckily, the EIPs are assigned to an interface which you will already have configured for the instance making it possible to communicate through the desired ports.

4.2 SECURITY

Security is a process not a product, this means that there is no application that will provide you with all the security you desire and need by just installing it. Security comes from a series of processes and methods that will provide you security at different levels.

4.2.1 Security Groups

Security Groups works in a similar fashion to a firewall as it carries a set of rules that filter traffic entering and leaving the EC2 instances. As said earlier, security groups are associated with the EC2 instances and offer protection at the ports and protocol access level. Typically, the firewall possesses a ‘Deny rule,’ but the SG has a “Deny All” that allows data packets to be dropped if no rule is assigned to them from the source IP.

If we compare it to a Network Access Control List (NACL), security groups form the first layer of defence at the instance level in a cloud computing environment whereas NACLs provides a second layer of protection at the subnet level.

In our scenario we will have to assign different Security Groups for each instance since they use different ports for each application.

4.2.2 Network ACLs [7]

Network ACLs are the next step of security that you can easily and freely use in AWS at the VPC level. This ACLs work as any other ACL that you may have configured in the past but will only be applied to the VPC as a whole, you can't apply an rule for each subnet.

The following are the basic things that you need to know about network ACLs:

- Your VPC automatically comes with a modifiable default network ACL. By default, it allows all inbound and outbound IPv4 traffic and, if applicable, IPv6 traffic.
- You can create a custom network ACL and associate it with a subnet. By default, each custom network ACL denies all inbound and outbound traffic until you add rules.
- Each subnet in your VPC must be associated with a network ACL. If you don't explicitly associate a subnet with a network ACL, the subnet is automatically associated with the default network ACL.
- You can associate a network ACL with multiple subnets. However, a subnet can be associated with only one network ACL at a time. When you associate a network ACL with a subnet, the previous association is removed.
- A network ACL has separate inbound and outbound rules, and each rule can either allow or deny traffic.
- Network ACLs are stateless, which means that responses to allowed inbound traffic are subject to the rules for outbound traffic (and vice versa).

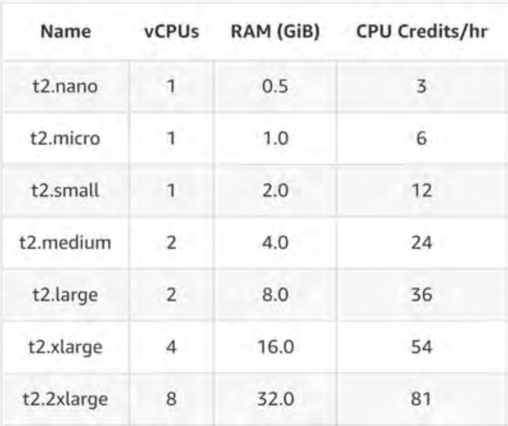
4.3 INSTANCES

Before creating any instance, you will have to decide which type of instance you need and which OS you will run.

4.3.1 Types [8] [9]

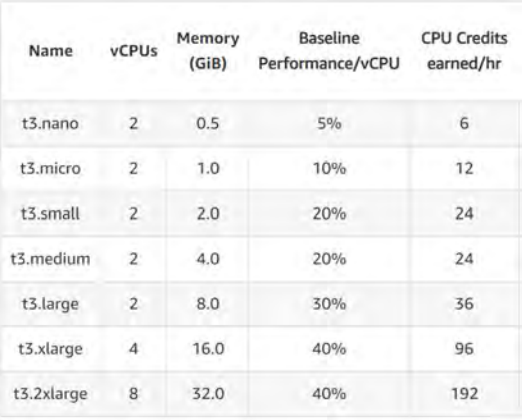
Amazon EC2 provides a wide selection of instance types optimized to fit different uses cases by varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications. Each instance type includes one or more instance sizes, allowing you to scale your resources to the requirements of your target workload.

In the following images you will see the different specifications for two different types of instances. Since these are the ones being used in this PoC.



Name	vCPUs	RAM (GiB)	CPU Credits/hr
t2.nano	1	0.5	3
t2.micro	1	1.0	6
t2.small	1	2.0	12
t2.medium	2	4.0	24
t2.large	2	8.0	36
t2.xlarge	4	16.0	54
t2.2xlarge	8	32.0	81

Figure 10 - EC2 T2 instances



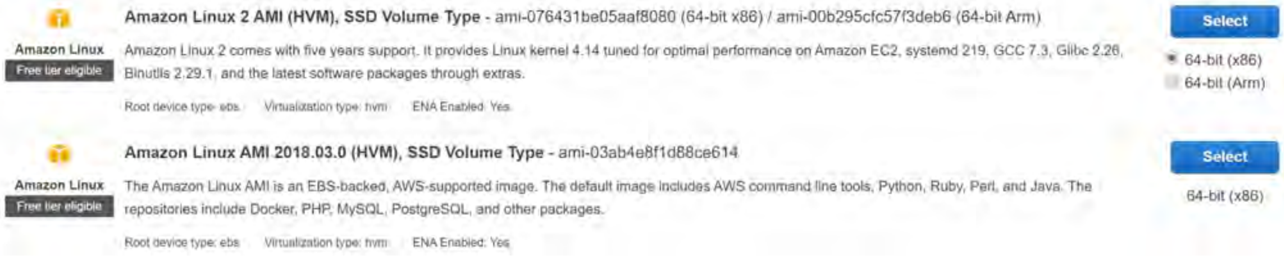
Name	vCPUs	Memory (GiB)	Baseline Performance/vCPU	CPU Credits earned/hr
t3.nano	2	0.5	5%	6
t3.micro	2	1.0	10%	12
t3.small	2	2.0	20%	24
t3.medium	2	4.0	20%	24
t3.large	2	8.0	30%	36
t3.xlarge	4	16.0	40%	96
t3.2xlarge	8	32.0	40%	192

Figure 11 - EC2 T3 Instances

During this PoC I have used the t2.micro for almost all of the instances and two t3.small for the service instance and AM instance due to that t2.micro instances only have a maximum of 2 network interfaces and a 1GB ram limitation.

4.3.2 OS

When launching an instance, you have a large variety free tier OS and proprietary OS to choose from. During this PoC I chose both Amazon Linux AMI for simplicity and resource efficiency.



Amazon Linux
Free tier eligible

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-076431be05aaf8080 (64-bit x86) / ami-00b295cfc57f3deb6 (64-bit Arm)

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

● 64-bit (x86)
○ 64-bit (Arm)

Amazon Linux
Free tier eligible

Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-03ab4e8f1d88ce614

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

64-bit (x86)

Figure 12 - Amazon Ami

There are two major differences regarding these two images. The first is that Amazon Linux 2 is the newer version of Linux from Amazon but it still is a bit behind repositor but it also has its advantages, which brings me to the second difference and that is that Amazon Linux 2 has systemctl installed.

Systemctl is a new tool to control the systemd system and service. This is the replacement of old SysV init system management. It may seem like it is not much, but you will be shocked on how many applications need systemctl as a requirement for off the shelve installation.

4.4 Instance Summary

Below is the table showing a more detailed information on the configuration done in this PoC.

Instance	Instance Type	OS	# Interfaces	VPC	Subnets	Security Group	Elastic IP
Bastion	T2.micro	Linux Ami	2	10.10.10.0/24	10.10.10.0/27	Bastion	Yes
Apache	T3a.small	Linux 2	3	10.10.11.0/24	10.10.11.0/28 10.10.11.16/28 10.10.11.64/28	Apache	Yes
Logstash	T2.micro	Linux Ami	2	10.10.11.0/24	10.10.11.16/28 10.10.11.32/28	Logstash	No
ElasticSearch	T2.micro	Linux Ami	2	10.10.11.0/24	10.10.11.32/28 10.10.11.48/28	ElasticSearch	No
Kibana	T2.micro	Linux Ami	2	10.10.11.0/24	10.10.11.48/28 10.10.11.128/25	Kibana	No
AM 6.5	T3a.small	Linux 2	2	10.10.11.0/24	10.10.11.32/28 10.10.11.64/28	AM	No

4.4.1 Logical Topology

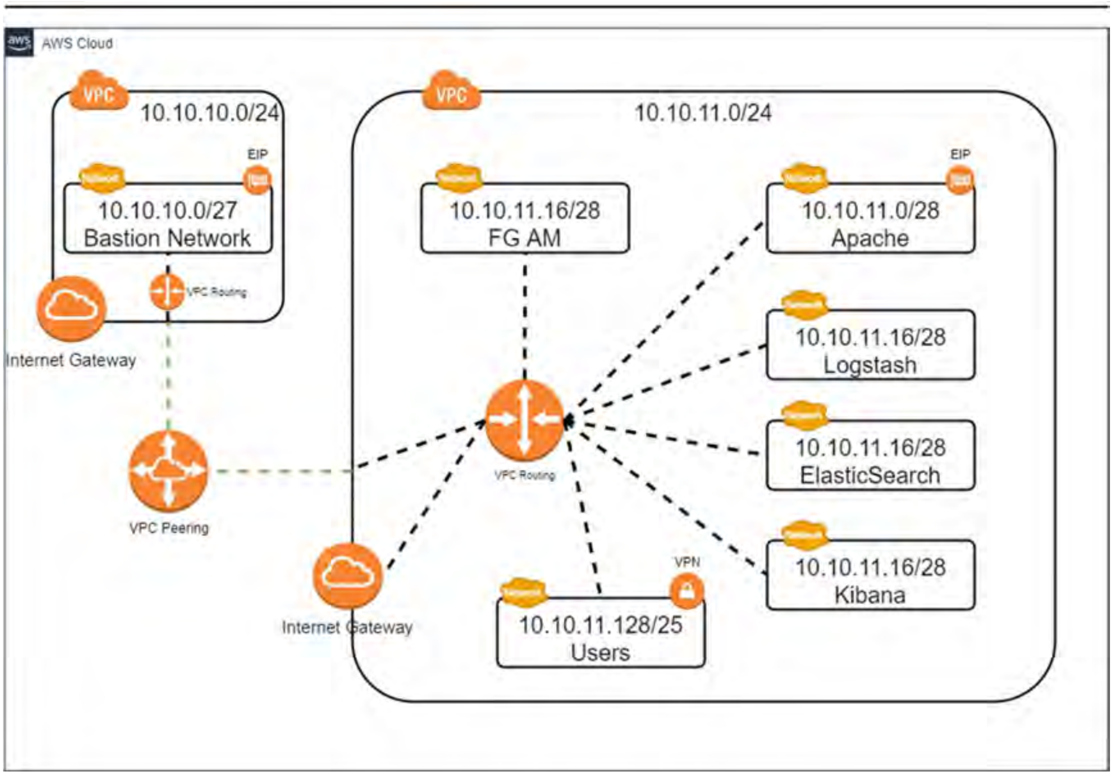


Figure 13 - Logical Topology

5 SERVICE ORIENTED ARCHITECTURE MIDDLEWARE

In this section I will be explaining the middleware employed during the deployment. Note that all the middleware used is free for testing purposes and open source. The only product that you will need to get a previous agreement of use with the OEM is with the AM 6.5 web agent, for which you will need to have a license.

5.1 CLIENT SERVICE

This is the service or services that will be exposed to the public internet and thus will be monitored with the ELK stack and protected with the AM6.5.

5.1.1 Wordpress Server

WordPress is a free and open-source content management system written in PHP and paired with a MySQL or MariaDB database. Features include a plugin architecture and a template system, referred to within WordPress as Themes.

I have decided to build a WordPress server since it was a web server that provides a lot of logs messages for debugging the ELK stack and also because it features a login interface which we will change for the AM login redirection.

This WordPress server requires an Apache server and a MySQL server running in the instance, which made the deployment with Ansible a bit trickier, but it provides a good example for complex installations for a web server.

5.2 MONITORING STACK

When talking about monitoring, there are three sub-domains of the problem: **metrics**, **logs** and **alerts**. Each of these is connected to a stream of data coming from the target system.

Metrics data stream contains real-time data about application or service (system components) performances. Metrics data can be seen as a real-time stream of scalar values with timestamps that represent the activities of different system parts.

Log data stream contains information about different activities and events within the system. Those data are usually in the form of free-form text records or, in some cases, JSON messages.

Alert data stream is what is called the information that helps tune this monitoring stack, depending on the values you receive and cross referencing the information, you will be able to proactively create an alerting system in your platform.

Because of the differences in their nature and different tools used to handle metrics and logs, in our case we will focus on the logging system instead of the metric and alert system, due to lack of time for this current PoC.

5.2.1 Filebeat [10]

*"**Filebeat** is a lightweight shipper for forwarding and centralizing log data. Installed as an agent on your servers, **Filebeat** monitors the log files or locations that you specify, collects log events, and forwards them either to Elasticsearch or Logstash for indexing."*

When you start Filebeat, it starts one or more inputs that monitor the logs that you have specified during the configuration. When it detects a change on the log, the harvester will send these lines to the libbeat which it will aggregate all the lines and send them to the next node, In our PoC this will be the Logstash.

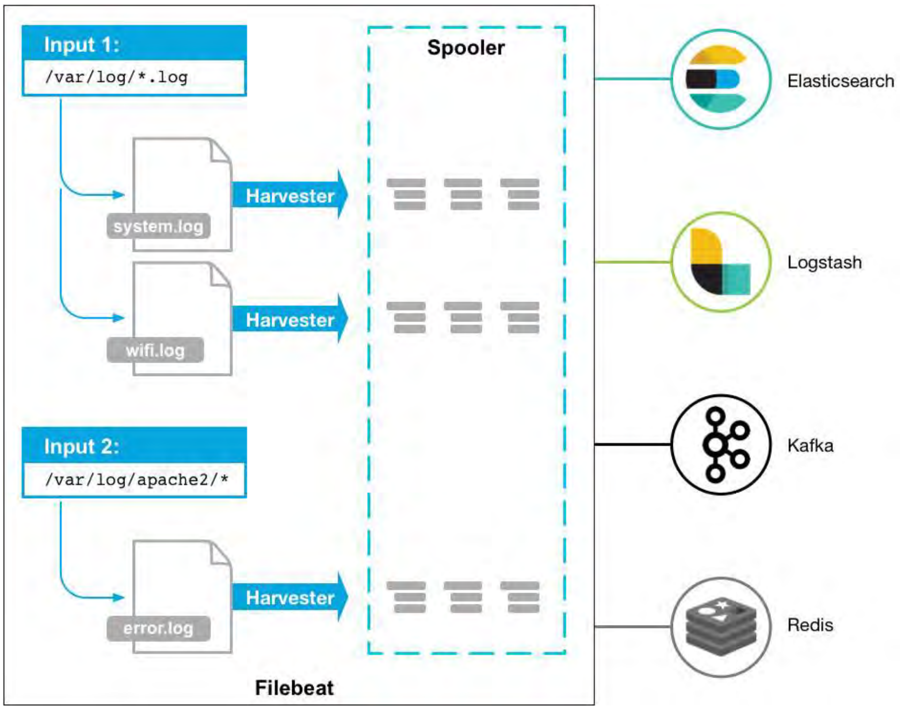


Figure 14 - Filebeat harvester

5.2.2 Redis [11]

"Redis (Remote Diccionary Service) is an in-memory data structure project implementing a distributed, in-memory key-value database with optional durability. Redis supports different kinds of abstract data structures, such as strings, lists, maps, sets, sorted sets, HyperLogLogs, bitmaps, streams, and spatial indexes. "

So, what does Redis do? Well, before the aggregated logs reach the Logstash service, they will be received by Redis which acts as a buffering layer, giving downstream components better chances of processing and indexing the data successfully.

This data will then be accessed by the Logstash as the latter is able to index all the information.

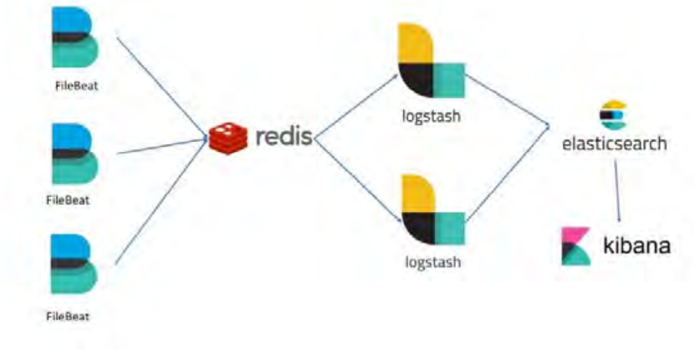


Figure 15 - Redis Service

5.2.3 Logstash [12]

"Logstash dynamically ingests, transforms, and ships your data regardless of format or complexity. Derive structure from unstructured data with grok, decipher geo coordinates from IP addresses, anonymize or exclude sensitive fields, and ease overall processing."

Logstash allows you to collect data from different systems and it also does something even more important: it normalises different schema from different applications. What does this mean? Simply put, Logstash allows you to create a single common format that will be applied to all the data that it gathers. This allows analytics engines like Elasticsearch and visualisation tools like Kibana to make the most of your data. If this wasn't the case, you can imagine what a complicated job it would be to cross reference information in Kibana and you wouldn't be able to compare data sets or see hay they impact on one another.

Scalablity

Logstash is fully scalable horizontally, this means that you can create clusters of Logstash, running the same pipeline, and that are fed from different Filebeats services. There are a few recommendations regarding scalability:

- Filebeats should **load balance** across a group of Logstash nodes.
- A minimum of two Logstash nodes are recommended for high availability.
- It's common to deploy just one Filebeat input per Logstash node, but multiple Beats inputs can also be deployed per Logstash node to expose independent endpoints for different data sources.

Regarding this PoC, there is only one Logstash in use due to low usage and for a lack of more services. This doesn't mean that the topology is incorrect, it only means that this PoC is not intended to simulate a real environment but if needed the only thing that should be done regarding the Logstash would be to create a cluster service.

5.2.4 ElasticSearch [13]

"Elasticsearch is a distributed, open source search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured. Elasticsearch is built on Apache Lucene and was first released in 2010 by Elasticsearch N.V. (now known as Elastic). Known for its simple REST APIs, distributed nature, speed, and

scalability, Elasticsearch is the central component of the Elastic Stack, a set of open source tools for data ingestion, enrichment, storage, analysis, and visualization. Commonly referred to as the ELK Stack (after Elasticsearch, Logstash, and Kibana), the Elastic Stack now includes a rich collection of lightweight shipping agents known as Beats for sending data to Elasticsearch."

ElasticSearch is a document-oriented database designed to store, retrieve, and manage data. When you use Elasticsearch, you store data in JSON document form.

Elasticsearch has its own query domain-specific language in which you specify the query in JSON format. You can also nest other queries where you require search on different fields by applying some conditions, different weights, recent documents, values of some predefined fields.

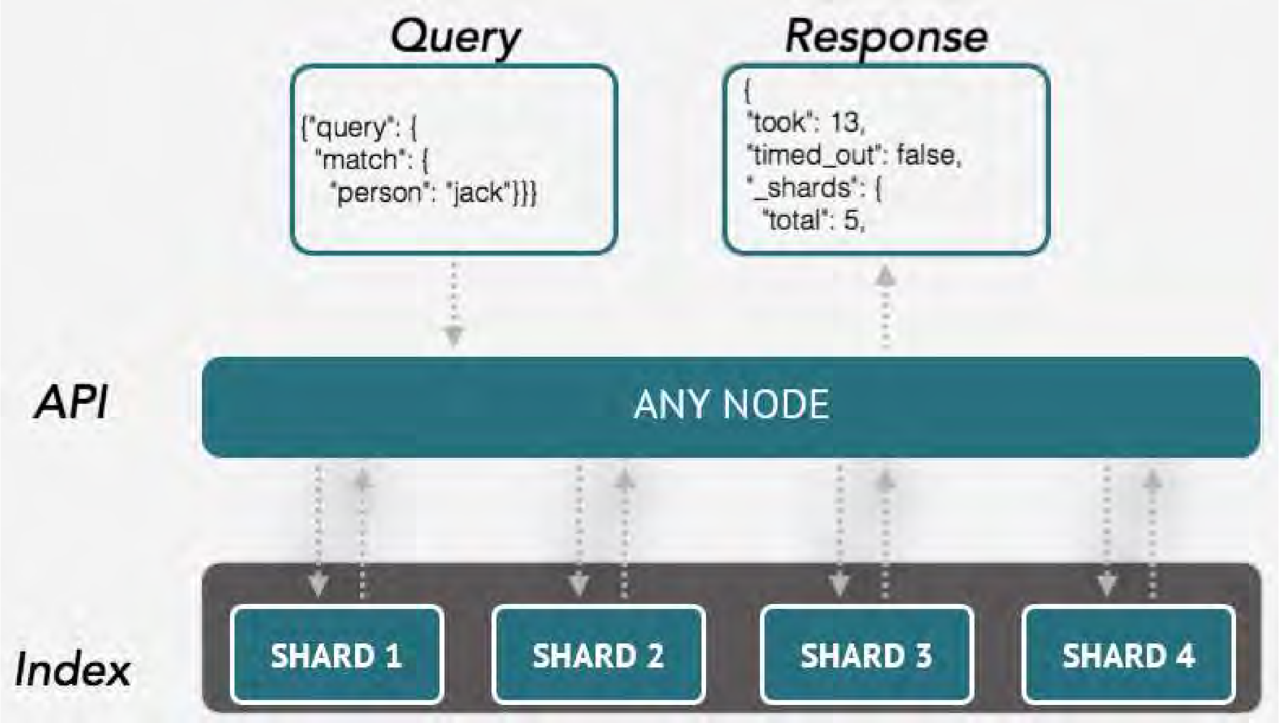


Figure 16 - ElastiSearch Diagram

The index is a collection of documents that have similar characteristics. An index is identified by a unique name that refers to the index when performing indexing search, update, and delete operations. In a single cluster, we can define as many indexes as we want. In our PoC we will have an index for every instance and service running.

Scalablity

ElasticSearch is fully capable of working with a single node, which is a single server that holds some data. In Elasticsearch, generally, one Elasticsearch instance runs per machine. Elasticsearch uses distributed computing, so having separate machines would help, as there would be more hardware resources.

The same as in LogStash, ElasticSearch can also be nested into clusters where this will add a total new level of redundancy and resiliency. Elastic clusters must contain a specific number of instances and come in different flavours.

- **(Eligible) master nodes**: controls the cluster.
- **Http nodes**: to run your queries to.
- **Data nodes**: the place data is stored, obviously.
- **Coordinating nodes**: see them as smart load balancers.

If you want to have a fault tolerant cluster the minimum requirement is:

- 3 locations to host your nodes. 2 locations to run half of your cluster, and one for the backup master node.
- 3 master nodes. You need an odd number of eligible master nodes to avoid split brains when you lose a whole data center. Put one master node in each location so you hopefully never lose the quorum.
- 2 http nodes, one in each primary data center.
- As many data nodes as you need, split evenly between both main locations.

Regarding this PoC, there is only one ElasticSearch node working for the same reasons as the Logstash. If you wish to use this in a production environment, I recommend creating a cluster for fault tolerance.

5.2.5 Kibana [14]

"Kibana is an open source frontend application that sits on top of the Elastic Stack, providing search and data visualization capabilities for data indexed in Elasticsearch."

Kibana is commonly known as the charting tool for the Elastic Stack and is used as the user interface for monitoring, managing, and securing an ELK stack cluster.

Kibana's core feature is data querying and analysis. Kibana is mainly used for visualizing data retrieved from logs. Using various methods, users can search the data indexed in Elasticsearch for specific events or strings within their data for root cause analysis and diagnostics. Based on these queries, users can use Kibana's visualization features which allow users to visualize data in a variety of different ways, using charts, tables, geographical maps and other types of visualizations.

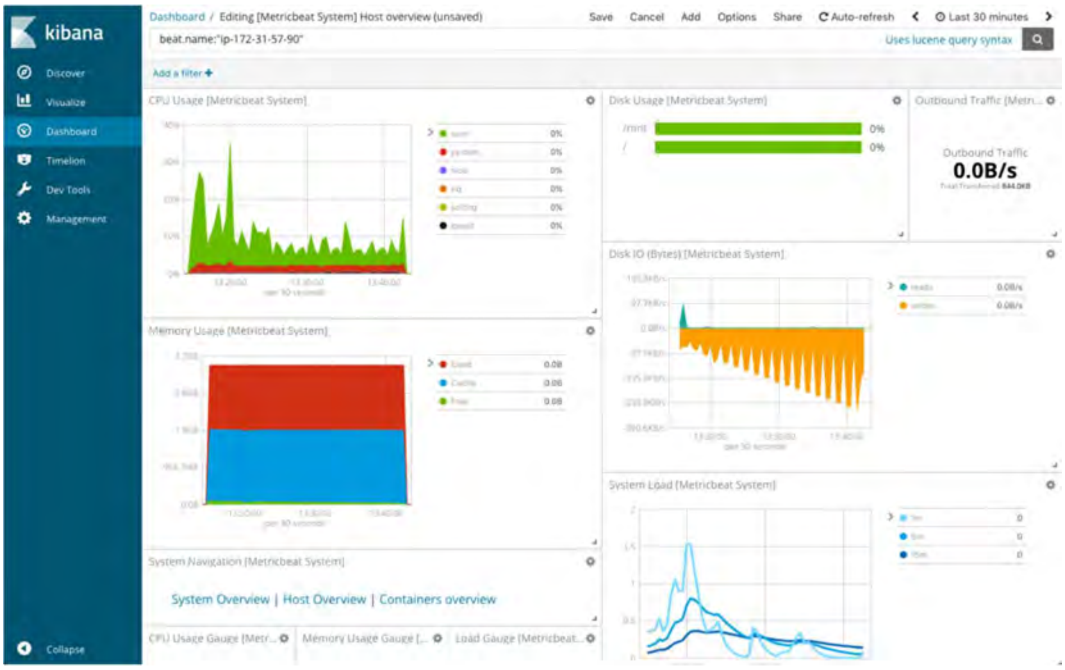

kibana
Discover
Visualize
Dashboard
Timelion
Dev Tools
Management
Collapse
Dashboard / Editing [Metricbeat System] Host overview (unsaved)
Save
Cancel
Add
Options
Share
Auto-refresh
Last 30 minutes
beat.name:"ip-172-31-57-90"
Uses lucene query syntax
Add a filter +
CPU Usage [Metricbeat System]
Disk Usage [Metricbeat System]
Outbound Traffic [Metr...
Outbound Traffic
0.0B/s
Disk IO (Bytes) [Metricbeat System]
Memory Usage [Metricbeat System]
System Load [Metricbeat System]
System Navigation [Metricbeat System]
System Overview | Host Overview | Containers overview
CPU Usage Gauge [Metr...
Memory Usage Gauge [...
Load Gauge [Metricbeat...

Figure 17 – Kibana example

In our PoC we have a Kibana deployed in one instance and is accessible for those users that have access to the VPN. Normally you should not let free access to this tool, so in this scenario we have protected this resource with the AM6.5 web agent which will authenticate and authorize users.

5.3 Security Middleware

For this PoC we will be introducing a layer of security that it oversees authorization and authenticating resources. AM is capable of handling SSO tokens, federation, and social media SSO. But for simplicity of this PoC we will be only using AM6.5 to protect the resources that are accessed by users and the internet.

5.3.1 Access Manager Concepts

In the following section I will be writing a small introduction on Access Managers where different concepts will arise, thus the need for this small section that I will explain those concepts beforehand.

Forgerock access manager, when deployed in a stack, the sessions will be stored in the CTS (core token service). This service will normally run a LDAP (Lightweight Directory Access Protocol) server, where the user information, session token, and cookies will be stored.

So, what is a session token? In computer science, session token is a piece of data that is used in network communications to identify a session, a series of related message exchanges. Session identifiers become necessary in cases where the communications infrastructure uses a stateless protocol such as HTTP.

How do you get a session token or session cookie? Sessions require the user or client to be able to hold on to cookies. Cookies provided by AM's Session Service may contain a JSON Web Token (JWT) with the session or just a reference to where the session is stored.

AM issues a cookie to the user or entity regardless of the session location for client-based and CTS-based sessions. By default, the cookie's name is iPlanetDirectoryPro in Forgerock products. For sessions stored on the client, the iPlanetDirectoryPro cookie contains all the information that would be held in the CTS token store.

Client-based session cookies are comprised of two parts. The first part of the cookie is identical to the cookie used by CTS-based sessions, which ensures the compatibility of the cookies regardless of the session location. The second part is a JSON Web Token (JWT), and it contains session information.

With this session token, the user will be able to log in the applications that nee

5.3.2 AM6.5 [15] From Forgerock

ForgeRock is a multinational identity and access management software company that develops commercial open source identity and access management products for internet of things, customer, cloud, mobile, and enterprise environments.

AM from ForgeRock provides a service called access management, which manages access to resources, such as web pages, application, web services available over the network. Once it is set up, AM provides an infrastructure for managing users, roles, and access to resources.

AM centralizes access control by handling both authentication and authorization. Authentication is the process of identifying an individual, for example, by confirming a successful login. Authorization is the process of granting access to resources to authenticated individuals.

How does AM manage authentications? AM centralization resides by using a variety of authentication modules that connect to identity repositories that store identities and provide authentication services. The identity repositories can be implemented as LDAP directories, relational databases, RADIUS, Windows authentication, one-time password services, and other standards-based access management systems.

AM lets you chain together a series of authentication chains that let you configure stronger authentication procedures for more sensitive resources. It also evaluates risk continuously by checking the device from where you are logging in and the location. This centralization permits AM to manage access policies separate from applications and resources. This can avoid issues that could arise when developers must embed policy decisions into their applications by only modifying the policy in AM without the need to redeploy the application.

In this PoC the tokens will be used when a user wants to have access to the protected resources.

5.3.2.1 AM Topology and Components

In order to correctly configure AM you will also need to configure what is called the CTS server and the config server. The first one is the server where all the tokens are saved and normally is an LDAP server, the latter is where all the AM configuration resides, this is due to easily escalate horizontally and in case of failure of the AM application you can just restart it or reinstall it without losing the configuration files.

ForgeRock recommends a specific topology where the CTS server and the config servers are deployed in a 2 by 2 configuration with replication for maximum H.

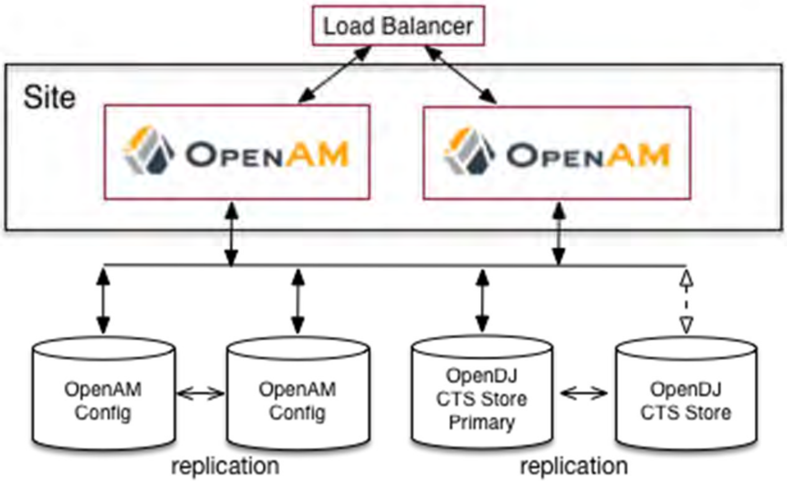


Figure 18 - AM topology

5.3.2.2 Web Agent Authorization [16]

How does a Web Agent work? Consider the case where AM protects a user profile web page. An AM web agent installed in the web server intercepts client requests to enforce policy. The policy says that only authenticated users can access the page to view and to update their profiles.

When a user browses to the profile page, the AM agent intercepts the request. The web agent notices that the request is to access a protected resource, but the request is coming from a user who has not yet logged in and consequently has no authorization to visit the page. The web agent therefore redirects the user's browser to AM to authenticate.

AM receives the redirected user, serving a login page that collects the user's email and password. With the email and password credentials, AM authenticates the user, and creates a session for the user. AM then redirects the user to the web agent, which gets the policy decision from AM for the page to access, and grants access to the page.

While the user has a valid session with AM, the user can go away to another page in the browser, come back to the profile page, and gain access without having to enter their email and password again.

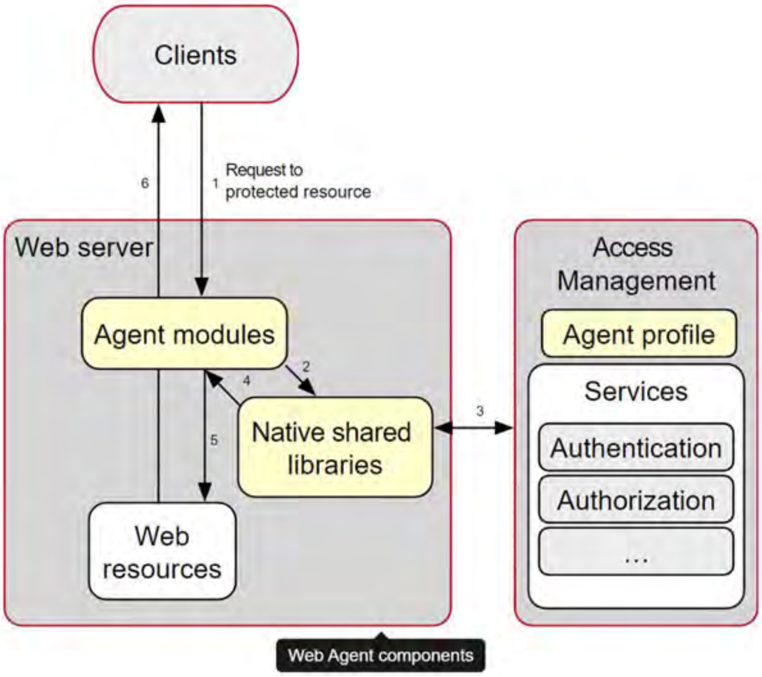


Figure 19 - Web Agent Diagram

With AM the web site developer can offer a profile page, but the web site developer never hast to manage who can access the page. As an AM administrator you can change authentication and authorization independently of the updates of the website whilst the agent is still active.

5.4 OTHER APPLICATIONS

5.4.1 Apache Web Server [17]

"Apache is an open-source and free web server software that powers around 40% of websites *around the world. The official name is* Apache HTTP Server*, and it's maintained and developed by the Apache Software Foundation.*

It allows website owners to serve content on the web — hence the name "web server." It's one of the oldest and most reliable web servers."

In this PoC the apache web server will serve as the backend server for the Wordpress application and also for the Kibana web application.

5.4.2 MySQL server [18]

"MySQL is a relational database management system based on SQL – Structured Query Language. The application is used for a wide range of purposes, including data warehousing, e-commerce, and logging applications. The most common use for mySQL however, is for the purpose of a web database."

In this PoC, the MySQL server is needed to run the Wordpress in the web server.

5.4.3 Tomcat 8 [19]

“Tomcat is an application server designed to execute Java servlets and render web pages that use Java Server page coding. Accessible as either a binary or a source code version, Tomcat's been used to power a wide range of applications and websites across the Internet.”

Tomcat is similar to the Apache server, but in this case, Tomcat will focus on executing java servlets and deploying them in the instance. Tomcat is indispensable to run AM6.5 which comes in a binary that must be deployed using tomcat.

5.4.4 SquidProxy [20]

“Squid is a caching and forwarding HTTP web proxy. It has a wide variety of uses, including speeding up a web server by caching repeated requests, caching web, DNS and other computer network lookups for a group of people sharing network resources, and aiding security by filtering traffic.”

The SquidProxy in the PoC is being used as a proxy for the yum command of all the instances, this will help with installation and maintenance whilst providing a secure connection through an instance instead of having a doorway to the internet.

Best practices state that this proxy server should have its own instance, but for simplicity we have configured this service inside the bastion server.

6 AUTOMATION

6.1 AUTOMATION TOOL

Automating infrastructure set up and configurations and the software deployment is the key highlight of DevOps practice. DevOps practice is heavily dependent on Automation in order to make deliveries over a period of few hours and make frequent deliveries across platforms.

How can DevOps teams accomplish such task? Automation tools are the answer.

6.1.1 Ansible [21]

"Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs.

Designed for multi-tier deployments since day one, Ansible models your IT infrastructure by describing how all of your systems inter-relate, rather than just managing one system at a time.

It uses no agents and no additional custom security infrastructure, so it's easy to deploy - and most importantly, it uses a very simple language (YAML, in the form of Ansible Playbooks) that allow you to describe your automation jobs in a way that approaches plain English."

6.2 METHODOLOGY

During the automatization process you will have to set a methodology or best practices before starting. Due to our current PoC and topology we must address the following questions:

- How many entry points will our infrastructure have in order to access the instances?
- Who will be launching the playbooks?
- From where these playbooks will be launched?
- How many endpoints will be open to the public?

All these questions and more will have to be answered before starting to code. My approach in this PoC will be by using a bastion or jump server, that will act as an intermediary gateway from the outside world to our newly deployed infrastructure.

6.2.1 Bastion Server

A bastion or jump server is an instance that runs inside our cloud infrastructure that has an the SSH port open to outside connections. This will be the only instance that will permit remote connections using crypto keys as the mean to authenticate.

During the execution of the playbook, some instructions of the code will run directly from our laptop and some others will execute inside the bastion instance. This means that the bastion must have installed the appropriate python libraries and ansible software to run our instructions.

In this PoC the bastion will serve as a jump server for the SSH connections, a remote ansible server and a software repository since it will always have access to the internet.

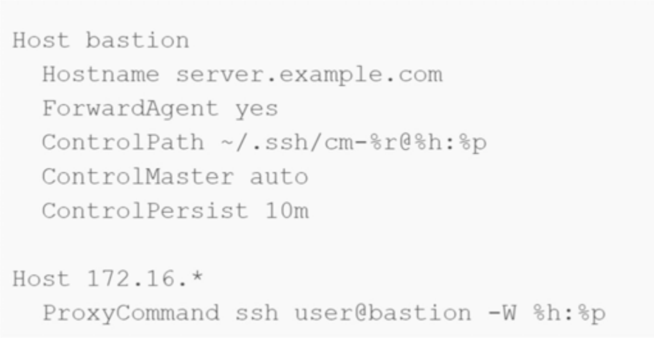
6.2.2 SSH Multiplexing [22]

One question arose during the development and that was, how can I connect to an instance that is two SSH jumps away from my laptop? And the answer is SSH multiplexing. With this solution I was able to connect through SSH to any instance inside the AWS infrastructure that had direct connection to my bastion.

“Generally speaking, multiplexing is the ability to carry multiple signals over a single connection. Similarly, SSH multiplexing is the ability to carry multiple SSH sessions over a single TCP connection.

One of the primary advantages of using SSH multiplexing is that it speeds up certain operations that rely on or occur over SSH. For example, let’s say that you’re using SSH to regularly execute a command on a remote host. Without multiplexing, every time that command is executed your SSH client must establish a new TCP connection and a new SSH session with the remote host. With multiplexing, you can configure SSH to establish a single TCP.”

How does it work? The idea of the multiplexing is by modifying the SSH config file that you see below.



```
Host bastion
  Hostname server.example.com
  ForwardAgent yes
  ControlPath ~/.ssh/cm-%r@%h:%p
  ControlMaster auto
  ControlPersist 10m

Host 172.16.*
  ProxyCommand ssh user@bastion -W %h:%p
```

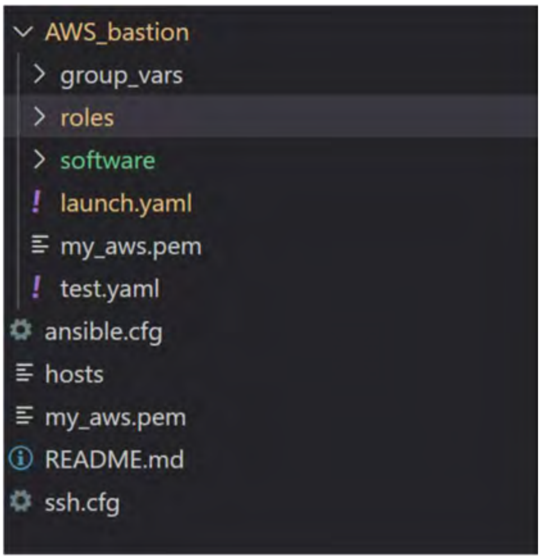
Figure 20 - SSH Multiplexing Configuration

Some of the configuration options are a bit complicated so, let’s break them down:

- The **ControlPath** entry specifies where to store the “control socket” for the multiplexed connections. In this case, %r refers to the remote login name, %h refers to the target host name, and %p refers to the destination port. Including this information in the control socket name helps SSH separate control sockets for connections to different hosts.
- The **ControlMaster** setting is what activates multiplexing. With the auto setting, SSH will try to use a master connection if one exists, but if one doesn’t exist it will create a new one.
- The **ControlPersist** setting keeps the master connection alive for the specified period of time after it has remained idle (no connections).

6.2.3 Code Structure in Ansible

In order to take on a project of this size, is imperative to have a code structure and if possible, a modular code structure. With this idea on mind you will be able to reuse the code in future projects and also it will be easier to take out modules that you no longer need.



```
∨ AWS_bastion
  > group_vars
  > roles
  > software
  ! launch.yaml
  ≡ my_aws.pem
  ! test.yaml
⚙ ansible.cfg
≡ hosts
≡ my_aws.pem
ⓘ README.md
⚙ ssh.cfg
```

Figure 21 - FIle Structure

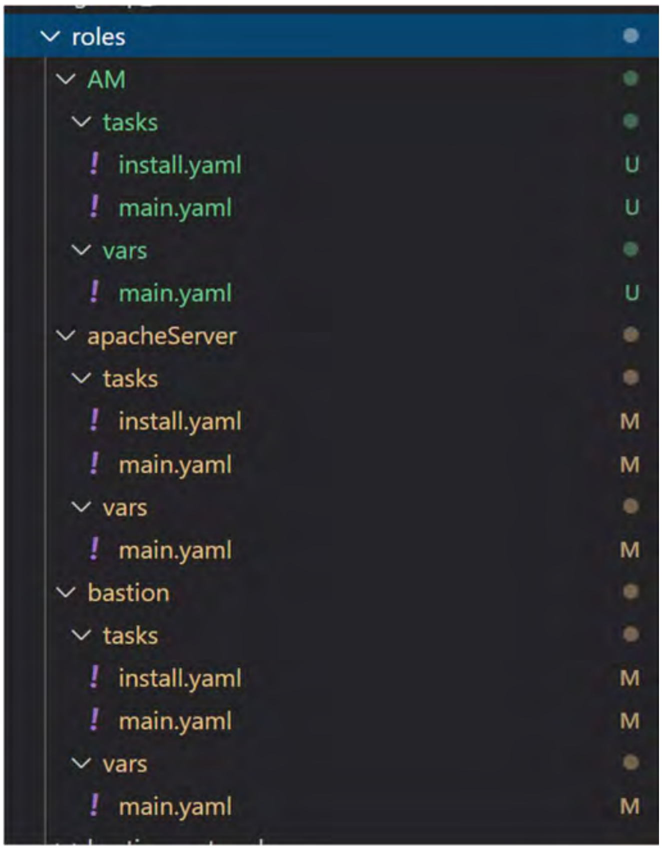
As you can see on the image above, the code is structured in several folders but it only has one main file, which in this case is "launch.yaml". Using ansible you will always will only be able to launch one playbook at a time. Does that mean that you have to write everything in one file? No, ansible has created the idea of Roles.

6.2.4 Roles

In Ansible, the role is the primary mechanism for breaking a playbook into multiple files. This simplifies writing complex playbooks, and it makes them easier to reuse. The breaking of playbook allows you to logically break the playbook into reusable components.

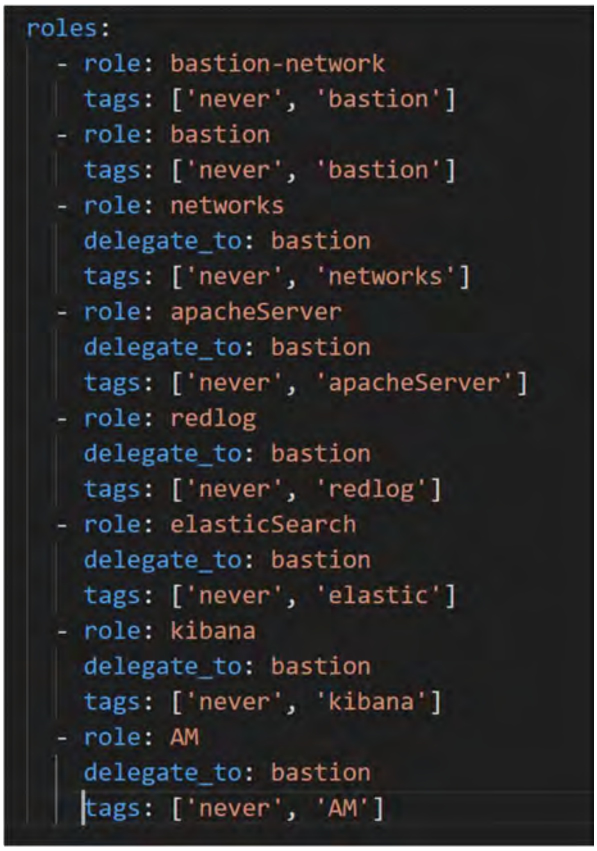
Each role is basically limited to a particular functionality or desired output, with all the necessary steps to provide that result either within that role itself or in other roles listed as dependencies.

Roles are not playbooks. Roles are small functionality which can be independently used but have to be used within playbooks. There is no way to directly execute a role. Roles have no explicit setting for which host the role will apply to.



```
∨ roles                      ●
  ∨ AM                       ●
    ∨ tasks                  ●
      ! install.yaml         U
      ! main.yaml            U
    ∨ vars                   ●
      ! main.yaml            U
  ∨ apacheServer             ●
    ∨ tasks                  ●
      ! install.yaml         M
      ! main.yaml            M
    ∨ vars                   ●
      ! main.yaml            M
  ∨ bastion                  ●
    ∨ tasks                  ●
      ! install.yaml         M
      ! main.yaml            M
    ∨ vars                   ●
      ! main.yaml            M
```

Figure 22- PoC Roles



```yaml
roles:
  - role: bastion-network
    tags: ['never', 'bastion']
  - role: bastion
    tags: ['never', 'bastion']
  - role: networks
    delegate_to: bastion
    tags: ['never', 'networks']
  - role: apacheServer
    delegate_to: bastion
    tags: ['never', 'apacheServer']
  - role: redlog
    delegate_to: bastion
    tags: ['never', 'redlog']
  - role: elasticSearch
    delegate_to: bastion
    tags: ['never', 'elastic']
  - role: kibana
    delegate_to: bastion
    tags: ['never', 'kibana']
  - role: AM
    delegate_to: bastion
    tags: ['never', 'AM']
```

Figure 23 - Roles in playbook

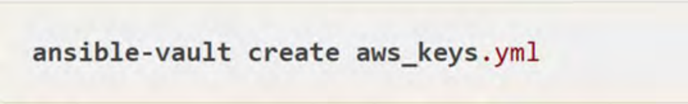
In this PoC there is one role for each module or instance created. Each role will have its main where the instance is created and configured and its installation file where all the instructions and configuration of services take place.

6.2.5 Using and configuring the Vault

Ansible Vault is a feature of ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles. These vault files can then be distributed or placed in source control.

To enable this feature, a command line tool - ansible-vault - is used to edit files, and a command line flag (--ask-vault-pass, --vault-password-file or --vault-id) is used.

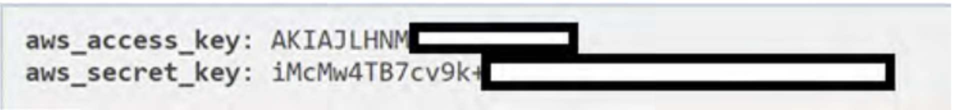
In our PoC, since we will have to use the AWS keys to access the cloud, it is recommended to store this keys with the vault option as follows:



```
ansible-vault create aws_keys.yml
```

Figure 24 - Vault editing

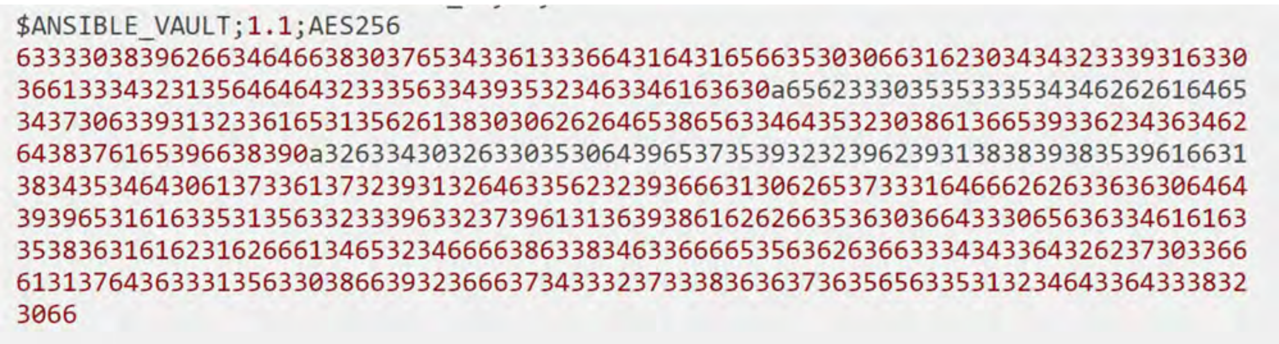
Once it opens add the IAM AWS keys:



```
aws_access_key: AKIAJLHNM[redacted]
aws_secret_key: iMcMw4TB7cv9k+[redacted]
```

Figure 25 - Vault editing keys

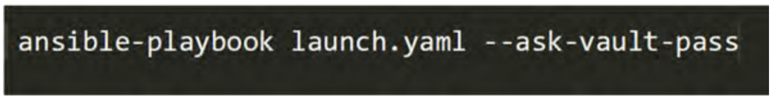
Once you save the file, all the content will be encrypted:



```
$ANSIBLE_VAULT;1.1;AES256
63333038396266346466383037653433613336643164316566353030663162303434323339316330
3661333432313564646432333563343935323463346163630a656233303535333534346262616465
34373063393132336165313562613830306262646538656334643532303861366539336234363462
6438376165396638390a326334303263303530643965373539323239623931383839383539616631
38343534643061373361373239313264633562323936663130626537333164666262633636306464
39396531616335313563323339633237396131363938616262663536303664333065636334616163
35383631616231626661346532346666386338346336666535636263663334343364326237303366
61313764363331356330386639323666373433323733383636373635656335313234643364333832
3066
```

Figure 26 - Vault Encrypted Keys

Now the only thing to do, is launch the ansible playbook like this:



```
ansible-playbook launch.yaml --ask-vault-pass
```

Figure 27 - Vault pass Playbook

6.3 Ansible Module Configurations

In this section I will be going through all the configurations that appear in the code, and all the modules that are needed to interact with AWS.

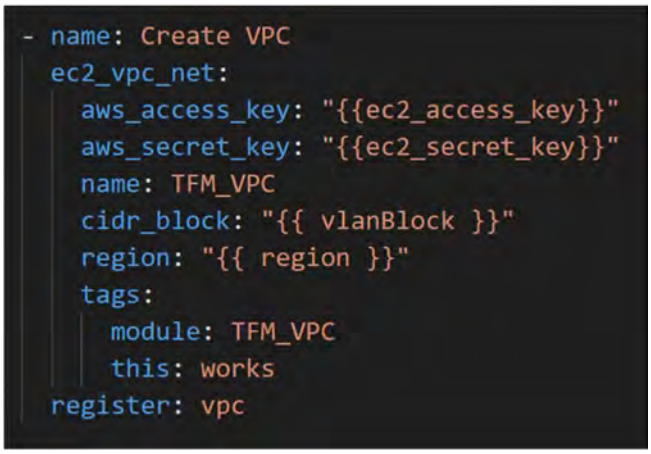
Note that the full code is attached in the annex section and check for updated versions of the code in:

https://bitbucket.org/nacsom/ansible-aws/src/master/

6.3.1 AWS Modules

Ansible has different modules to interact with the different components inside AWS. In this section I will be going over the ones used.

6.3.1.1 Create VPC



```
- name: Create VPC
  ec2_vpc_net:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    name: TFM_VPC
    cidr_block: "{{ vlanBlock }}"
    region: "{{ region }}"
    tags:
      module: TFM_VPC
      this: works
  register: vpc
```

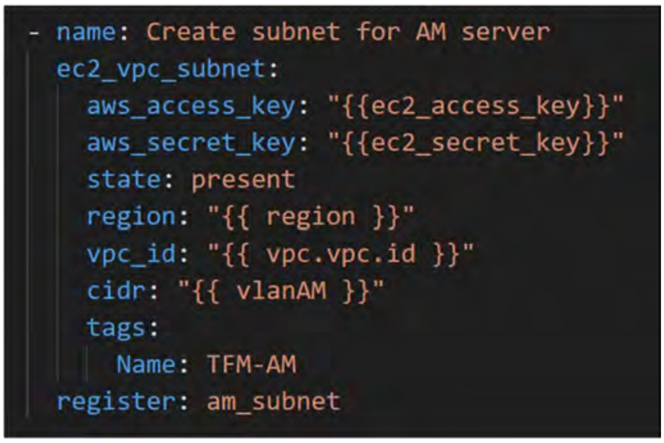
Figure 28 - Ansible VPC creation

In the image on the left you will see the module needed to create a VPC with Ansible.

There are more options for this module, but these are the necessary for it to work.

Note that cidr_block is the network assigned to the VPC.

6.3.1.2 Create Subnet



```
- name: Create subnet for AM server
  ec2_vpc_subnet:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    state: present
    region: "{{ region }}"
    vpc_id: "{{ vpc.vpc.id }}"
    cidr: "{{ vlanAM }}"
    tags:
      Name: TFM-AM
  register: am_subnet
```

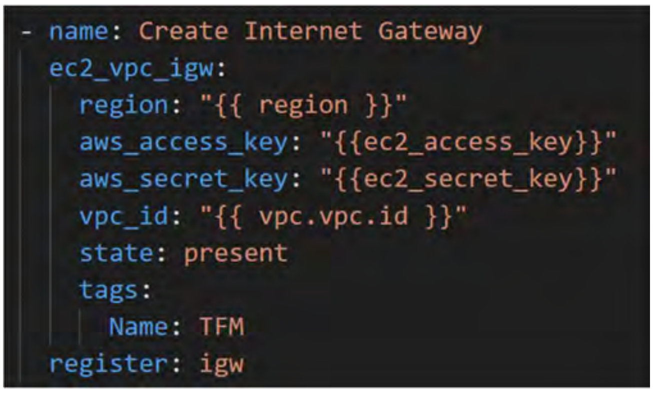
Figure 29 - Ansible Subnet creation

In the image on the left you will see the module needed to create a subnet with Ansible.

There are more options for this module, but these are the necessary for it to work.

Note that most of the variables needed will be provided by previous configuration modules or information modules.

6.3.1.3 Create Internet Gateway



```
- name: Create Internet Gateway
  ec2_vpc_igw:
    region: "{{ region }}"
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    vpc_id: "{{ vpc.vpc.id }}"
    state: present
    tags:
      Name: TFM
  register: igw
```

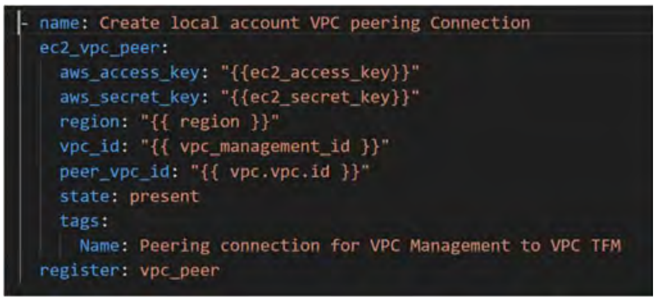
Figure 30 - Ansible IGW creation

In the image on the left you will see the module needed to create a IGW with Ansible.

There are more options for this module, but these are the necessary for it to work.

Note that most of the variables needed will be provided by previous configuration modules or information modules.

6.3.1.4 *Create VPC Peering connection*

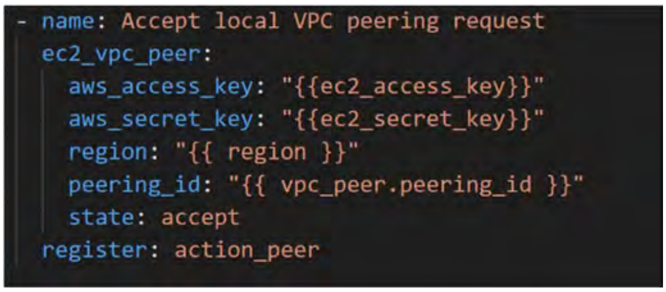


```yaml
- name: Create local account VPC peering Connection
  ec2_vpc_peer:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    region: "{{ region }}"
    vpc_id: "{{ vpc_management_id }}"
    peer_vpc_id: "{{ vpc.vpc.id }}"
    state: present
    tags:
      Name: Peering connection for VPC Management to VPC TFM
  register: vpc_peer
```

Figure 31 - Ansible VPC peering creation

In the image on the left you will see the module needed to create a VPC peerng with Ansible.

There are more options for this module, but these are the necessary for it to work.

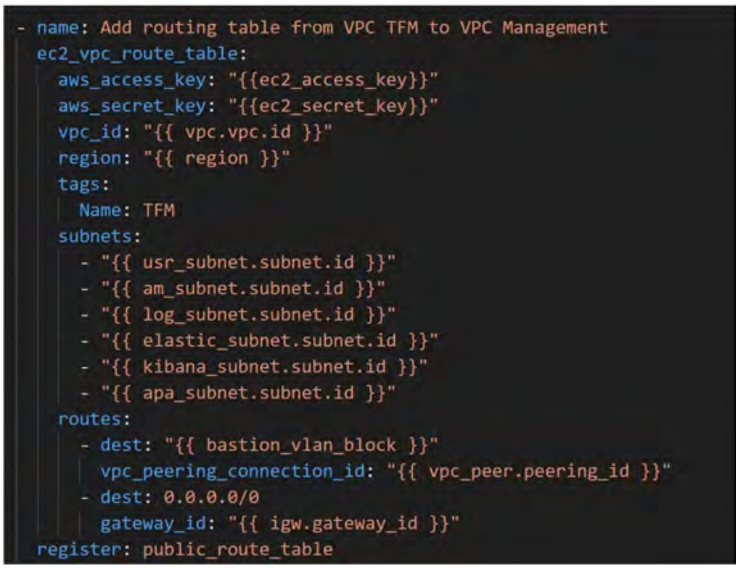


```yaml
- name: Accept local VPC peering request
  ec2_vpc_peer:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    region: "{{ region }}"
    peering_id: "{{ vpc_peer.peering_id }}"
    state: accept
  register: action_peer
```

Figure 32 - Ansible VPC peering Configuration

In order for the previous configuration to take effect, you have to also add this other module where you will specify the **peering id** and the **state: accept**.

6.3.1.5 *Create Routing Table*



```yaml
- name: Add routing table from VPC TFM to VPC Management
  ec2_vpc_route_table:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    vpc_id: "{{ vpc.vpc.id }}"
    region: "{{ region }}"
    tags:
      Name: TFM
    subnets:
      - "{{ usr_subnet.subnet.id }}"
      - "{{ am_subnet.subnet.id }}"
      - "{{ log_subnet.subnet.id }}"
      - "{{ elastic_subnet.subnet.id }}"
      - "{{ kibana_subnet.subnet.id }}"
      - "{{ apa_subnet.subnet.id }}"
    routes:
      - dest: "{{ bastion_vlan_block }}"
        vpc_peering_connection_id: "{{ vpc_peer.peering_id }}"
      - dest: 0.0.0.0/0
        gateway_id: "{{ igw.gateway_id }}"
  register: public_route_table
```

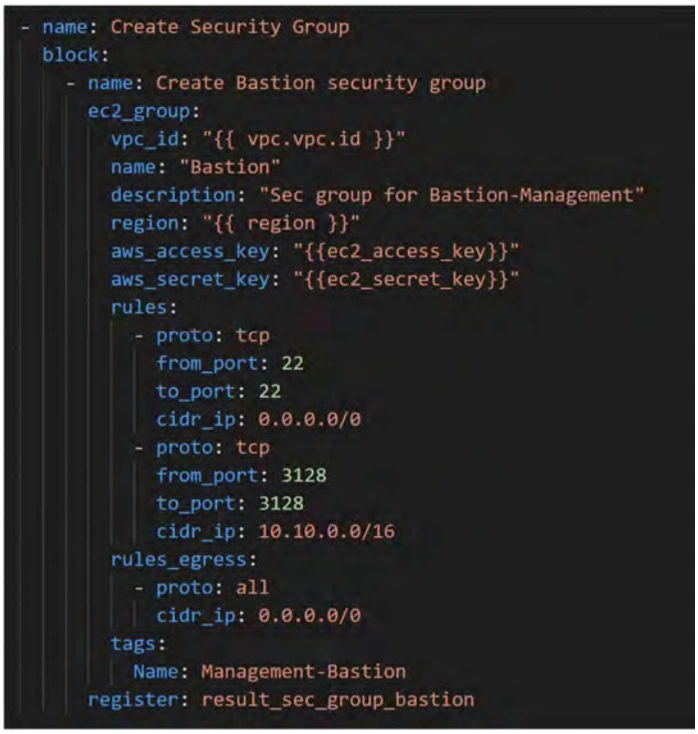
Figure 33 - Ansible Route Table Creation

In the image on the left you will see the module needed to create a Route table with Ansible.

There are more options for this module, but these are the necessary for it to work.

Regarding this configuration, if you look on the las part, you will see that in order for the VPC to have a connection with the outside world, you will have to specify the IGW as a destiny.

6.3.1.6 Create Security Group



```
- name: Create Security Group
  block:
    - name: Create Bastion security group
      ec2_group:
        vpc_id: "{{ vpc.vpc.id }}"
        name: "Bastion"
        description: "Sec group for Bastion-Management"
        region: "{{ region }}"
        aws_access_key: "{{ec2_access_key}}"
        aws_secret_key: "{{ec2_secret_key}}"
        rules:
          - proto: tcp
            from_port: 22
            to_port: 22
            cidr_ip: 0.0.0.0/0
          - proto: tcp
            from_port: 3128
            to_port: 3128
            cidr_ip: 10.10.0.0/16
        rules_egress:
          - proto: all
            cidr_ip: 0.0.0.0/0
        tags:
          Name: Management-Bastion
      register: result_sec_group_bastion
```

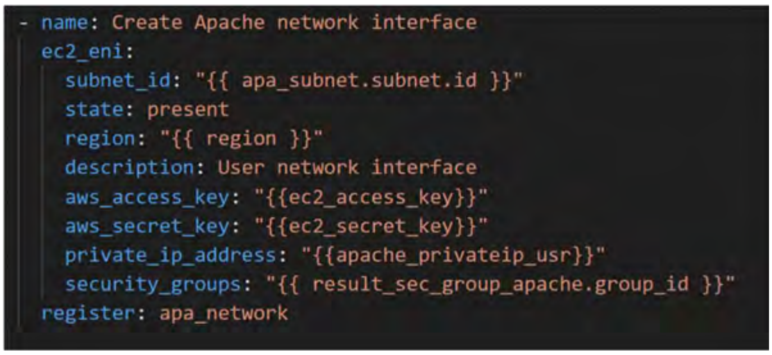
Figure 34 - Ansible Security Group Creation

In the image on the left you will see the module needed to create a Security Group with Ansible.

There are more options for this module, but these are the necessary for it to work.

As mentioned before, security groups act like the ACLs inside your instance. With a security group you will be able to specify which ports will be open in each instance.

6.3.1.7 Create Interface



```
- name: Create Apache network interface
  ec2_eni:
    subnet_id: "{{ apa_subnet.subnet.id }}"
    state: present
    region: "{{ region }}"
    description: User network interface
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    private_ip_address: "{{apache_privateip_usr}}"
    security_groups: "{{ result_sec_group_apache.group_id }}"
  register: apa_network
```

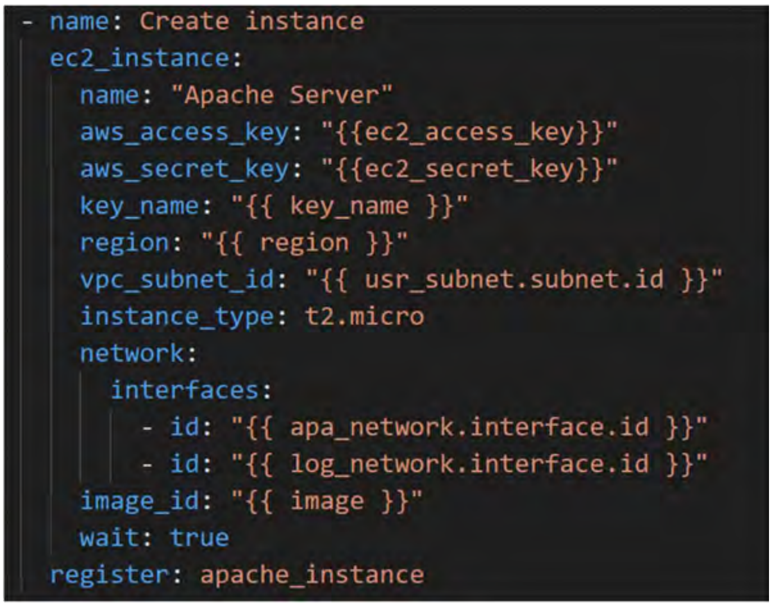
Figure 35 - Ansible AWS Interface

In the image on the left you will see the module needed to create a network interface with Ansible.

There are more options for this module, but these are the necessary for it to work.

During the network configuration is imperative to assign the desired security group for that interface.

6.3.1.8 *Create Instance*



```
- name: Create instance
  ec2_instance:
    name: "Apache Server"
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    key_name: "{{ key_name }}"
    region: "{{ region }}"
    vpc_subnet_id: "{{ usr_subnet.subnet.id }}"
    instance_type: t2.micro
    network:
      interfaces:
        - id: "{{ apa_network.interface.id }}"
        - id: "{{ log_network.interface.id }}"
    image_id: "{{ image }}"
    wait: true
  register: apache_instance
```

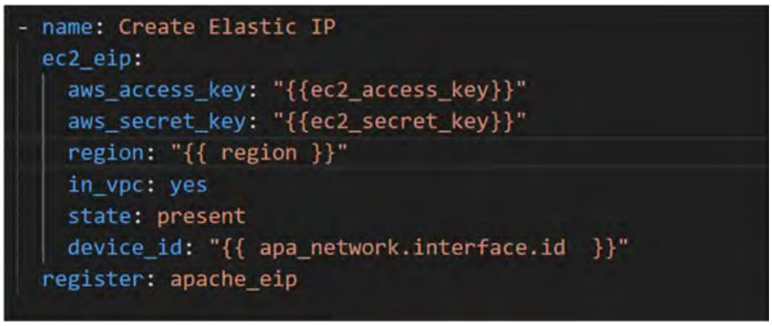
Figure 36 - Ansible AWS Instance

In the image on the left you will see the module needed to create an instance in AWS with Ansible.

There are more options for this module, but these are the necessary for it to work.

In this module you will have to specify the type of instance that you want and also the interfaces attached to it.

6.3.1.9 *Create Elastic IP*



```
- name: Create Elastic IP
  ec2_eip:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    region: "{{ region }}"
    in_vpc: yes
    state: present
    device_id: "{{ apa_network.interface.id  }}"
  register: apache_eip
```

Figure 37 - Ansible AWS EIP

In the image on the left you will see the module needed to create an Elastic IP in AWS with Ansible.

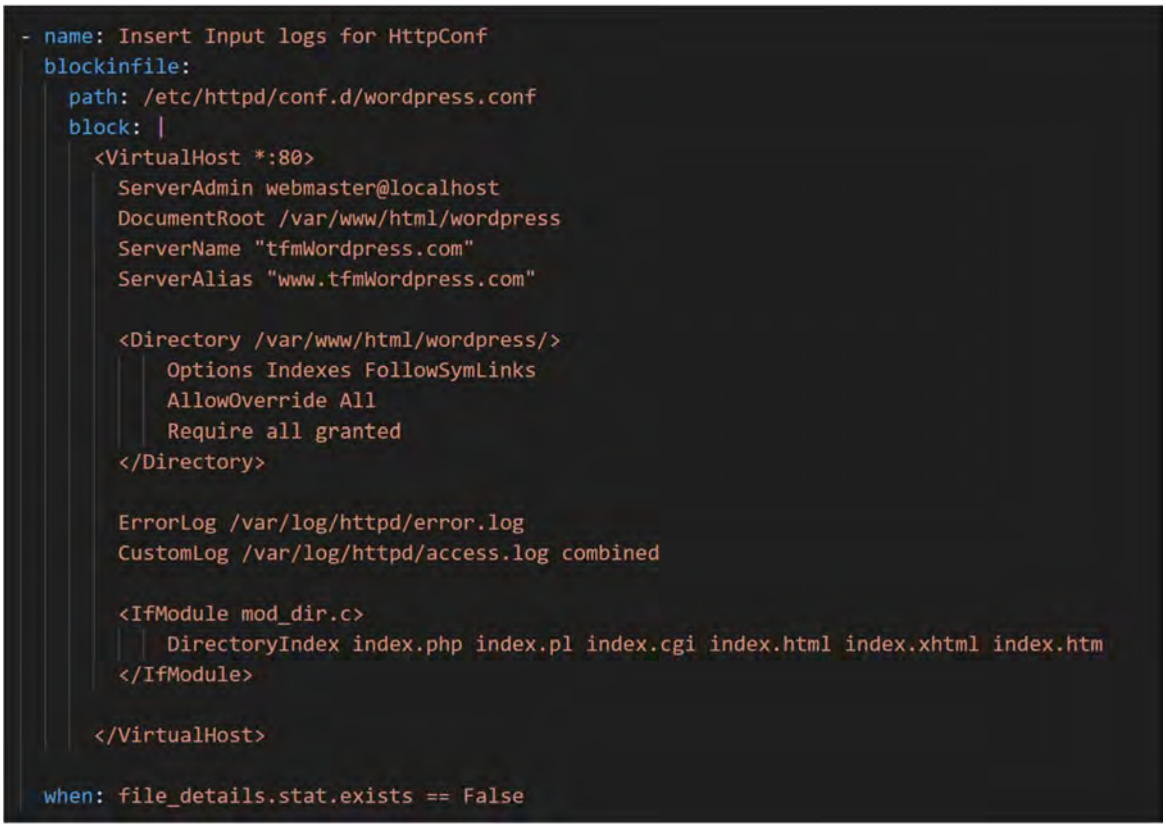
There are more options for this module, but these are the necessary for it to work.

In this module you can specify if this EIP could be reused or always stay assigned to this interface.

6.3.2 Apache server

The installation of the Apache web server is very straight forward, the only important aspect of this is that you will only need to create the "httpd.conf" file for the server you desire, in this case, WordPress. The httpd.conf file in apache is the main configuration file for the Apache web server.

In this file you will also be able to specify if there is a redirection to the secure web HTTPS. In this PoC this was not inside the scope due to restrictions regarding DNS services which I will explain in another section.



```
- name: Insert Input logs for HttpConf
  blockinfile:
    path: /etc/httpd/conf.d/wordpress.conf
    block: |
      <VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html/wordpress
        ServerName "tfmWordpress.com"
        ServerAlias "www.tfmWordpress.com"

        <Directory /var/www/html/wordpress/>
            Options Indexes FollowSymLinks
            AllowOverride All
            Require all granted
        </Directory>

        ErrorLog /var/log/httpd/error.log
        CustomLog /var/log/httpd/access.log combined

        <IfModule mod_dir.c>
            DirectoryIndex index.php index.pl index.cgi index.html index.xhtml index.htm
        </IfModule>

      </VirtualHost>

  when: file_details.stat.exists == False
```

Figure 38 - Ansible Apache httpd.conf file

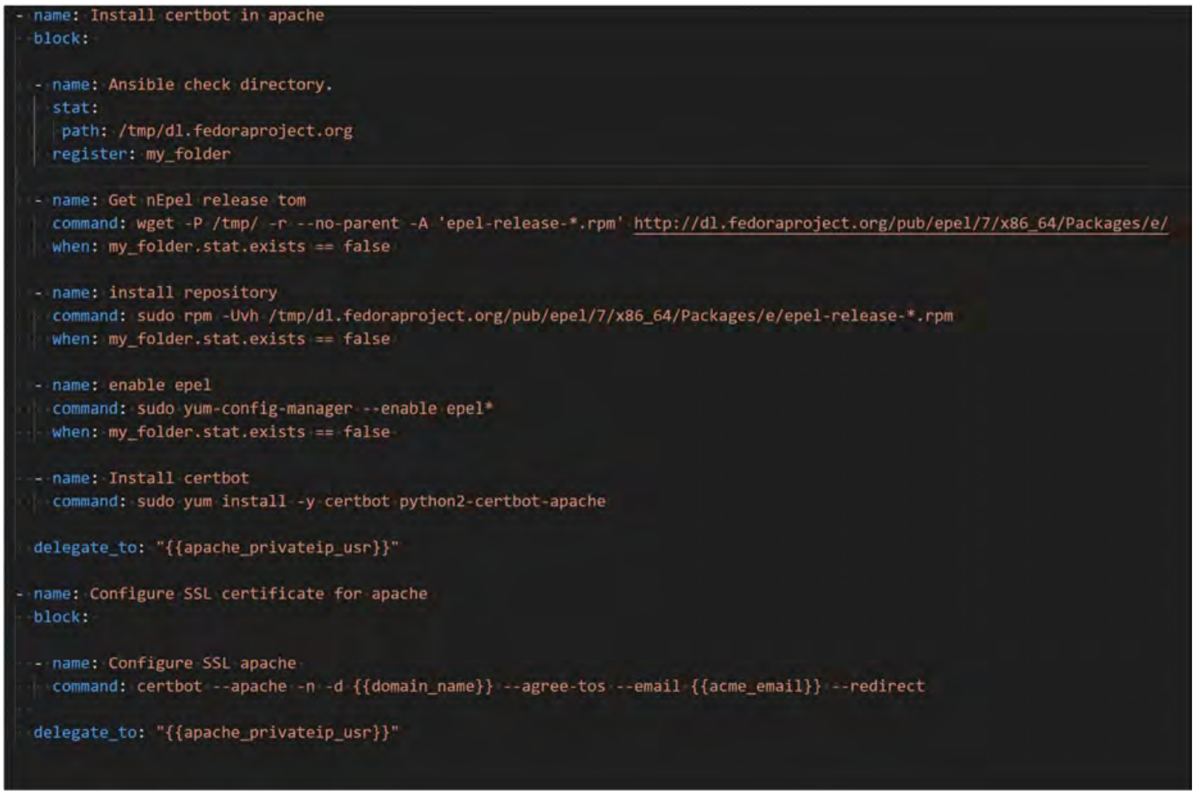
6.3.2.1 LetsEncrypt & Certbot configuration

One simple action to have a "secure" communication with the web server is to encrypt the http communication into https with SSL. This is done by obtaining a web certificate for your website.

How do you do it? First, you need to find a certificate authority, you can get access to these authorities by paying a small fee or using an opensource non-profit authority. I will be explaining the latter.

I will be using LetsEncrypt, that is a non-profit certificate authority run by Internet Security Research Group that provides X.509 certificates for Transport Layer Security encryption at no charge. How do I get a certificate? Certbot is your answer. Certbot is an easy-to-use client that fetches a certificate from Let's Encrypt—an open certificate authority launched by the EFF, Mozilla, and others—and deploys it to a web server. Certbot will need the registered domain name and the admin email, the rest is done internally by the client.

Below you will see the configuration needed to deploy your own https certificate.



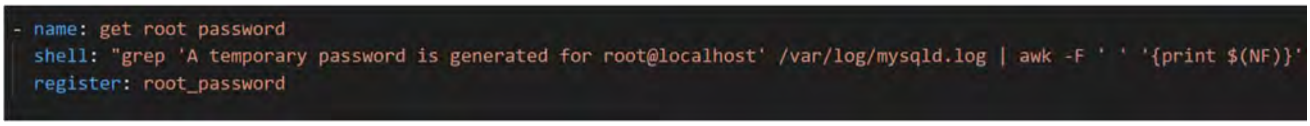
```
- name: Install certbot in apache
  block:

   - name: Ansible check directory.
     stat:
      path: /tmp/dl.fedoraproject.org
     register: my_folder

   - name: Get nEpel release tom
     command: wget -P /tmp/ -r --no-parent -A 'epel-release-*.rpm' http://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/
     when: my_folder.stat.exists == false

   - name: install repository
     command: sudo rpm -Uvh /tmp/dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/epel-release-*.rpm
     when: my_folder.stat.exists == false

   - name: enable epel
     command: sudo yum-config-manager --enable epel*
     when: my_folder.stat.exists == false

   - name: Install certbot
     command: sudo yum install -y certbot python2-certbot-apache

  delegate_to: "{{apache_privateip_usr}}"

- name: Configure SSL certificate for apache
  block:

   - name: Configure SSL apache
     command: certbot --apache -n -d {{domain_name}} --agree-tos --email {{acme_email}} --redirect

  delegate_to: "{{apache_privateip_usr}}"
```

Figure 39 - SSL Certbot configuration

6.3.3 MySQL server

During the MySQL server installation, you will need to automate a series of actions in order to be able to start the server correctly.

The first of the actions, to change the default root password created during the installation and replace it with a secure and known one. **NOTE** in newer versions of MySQL the default password and the password you establish as root will rotate every 'x' days. If you don't want this you will have to manually deactivate this function or change the number of days until rotation.

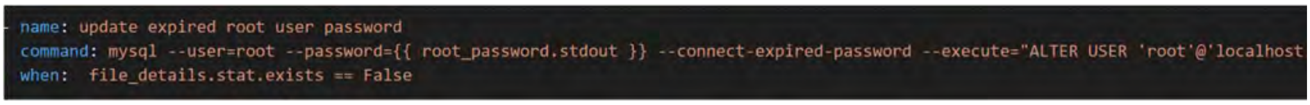


```
- name: get root password
  shell: "grep 'A temporary password is generated for root@localhost' /var/log/mysqld.log | awk -F ' ' '{print $(NF)}'
  register: root_password
```

Figure 40 - Ansible MySQL Password Conf

In the figure above you will see how we find the temporary root password assigned during the installation. This password is needed in order to change it.

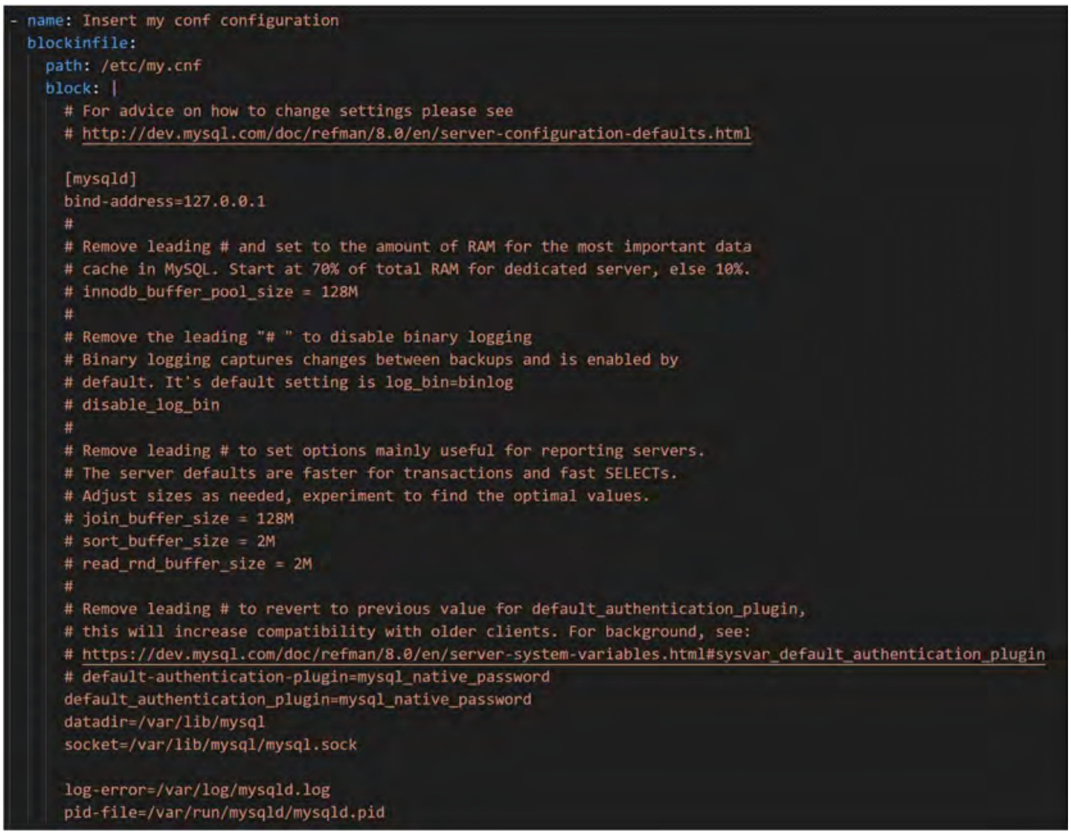
In the figure below you will be able to see how we change the password executing remotely the MySQL CLI.



```
- name: update expired root user password
  command: mysql --user=root --password={{ root_password.stdout }} --connect-expired-password --execute="ALTER USER 'root'@'localhost
  when:  file_details.stat.exists == False
```

Figure 41 - Ansible MySQL Password change

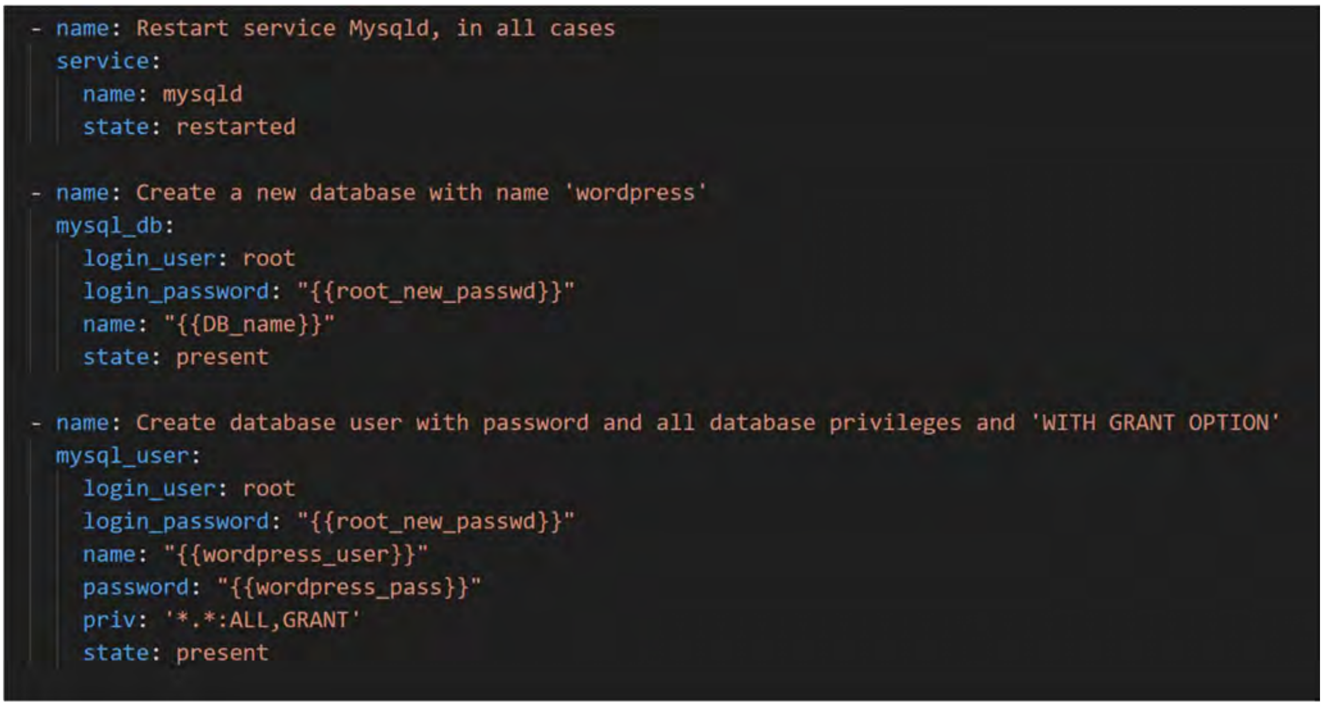
Once this is done, the only thing needed is to bind the server to the localhost and define the paths of the logs and where the socket will be created. This is done by configuring the “my.conf” file. The “my.conf” file in MySQL is the main configuration file for the MySQL server.



```
- name: Insert my conf configuration
  blockinfile:
    path: /etc/my.cnf
    block: |
      # For advice on how to change settings please see
      # http://dev.mysql.com/doc/refman/8.0/en/server-configuration-defaults.html

      [mysqld]
      bind-address=127.0.0.1
      #
      # Remove leading # and set to the amount of RAM for the most important data
      # cache in MySQL. Start at 70% of total RAM for dedicated server, else 10%.
      # innodb_buffer_pool_size = 128M
      #
      # Remove the leading "# " to disable binary logging
      # Binary logging captures changes between backups and is enabled by
      # default. It's default setting is log_bin=binlog
      # disable_log_bin
      #
      # Remove leading # to set options mainly useful for reporting servers.
      # The server defaults are faster for transactions and fast SELECTs.
      # Adjust sizes as needed, experiment to find the optimal values.
      # join_buffer_size = 128M
      # sort_buffer_size = 2M
      # read_rnd_buffer_size = 2M
      #
      # Remove leading # to revert to previous value for default_authentication_plugin,
      # this will increase compatibility with older clients. For background, see:
      # https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_default_authentication_plugin
      # default-authentication-plugin=mysql_native_password
      default_authentication_plugin=mysql_native_password
      datadir=/var/lib/mysql
      socket=/var/lib/mysql/mysql.sock

      log-error=/var/log/mysqld.log
      pid-file=/var/run/mysqld/mysqld.pid
```

Figure 42 - Ansible MySQL my.conf file

Last but not least, you will have to restart the service and create the database and the user needed, in this case, for WordPress.



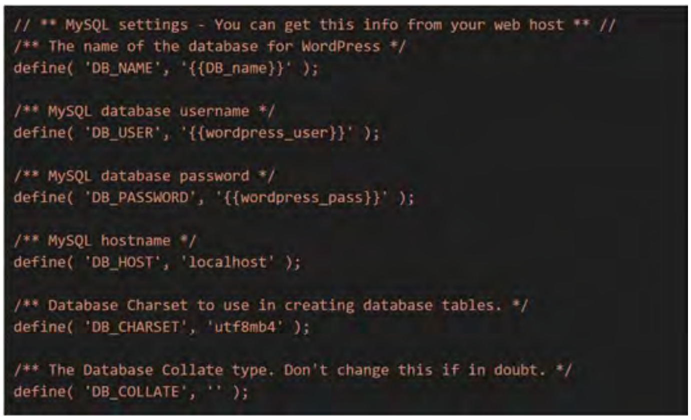
```
- name: Restart service Mysqld, in all cases
  service:
    name: mysqld
    state: restarted

- name: Create a new database with name 'wordpress'
  mysql_db:
    login_user: root
    login_password: "{{root_new_passwd}}"
    name: "{{DB_name}}"
    state: present

- name: Create database user with password and all database privileges and 'WITH GRANT OPTION'
  mysql_user:
    login_user: root
    login_password: "{{root_new_passwd}}"
    name: "{{wordpress_user}}"
    password: "{{wordpress_pass}}"
    priv: '*.*:ALL,GRANT'
    state: present
```

Figure 43 - Ansible MySQL database and user creation

6.3.4 WordPress

Wordpress configuration is very straight forward, during a manual installation process… but if you want to skip some of those steps, you will have to become creative and create the installation files and the htaccess file beforehand.

These configurations are in the annex, but below you will see some snippets of the configuration.



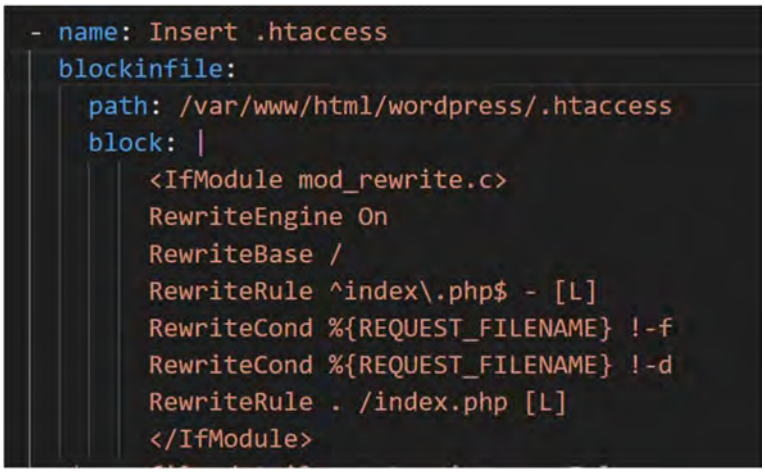
```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', '{{DB_name}}' );

/** MySQL database username */
define( 'DB_USER', '{{wordpress_user}}' );

/** MySQL database password */
define( 'DB_PASSWORD', '{{wordpress_pass}}' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8mb4' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );
```

Figure 44 - wp-config.php

This is part of the configuration you will have to create before running the server.

This will specify the Wordpress the MySQL database and user created for it.

Normally this file is created automatically if you load the web and configure it manually, in this way we are jumping that configuration.



```
- name: Insert .htaccess
  blockinfile:
    path: /var/www/html/wordpress/.htaccess
    block: |
        <IfModule mod_rewrite.c>
        RewriteEngine On
        RewriteBase /
        RewriteRule ^index\.php$ - [L]
        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteCond %{REQUEST_FILENAME} !-d
        RewriteRule . /index.php [L]
        </IfModule>
```

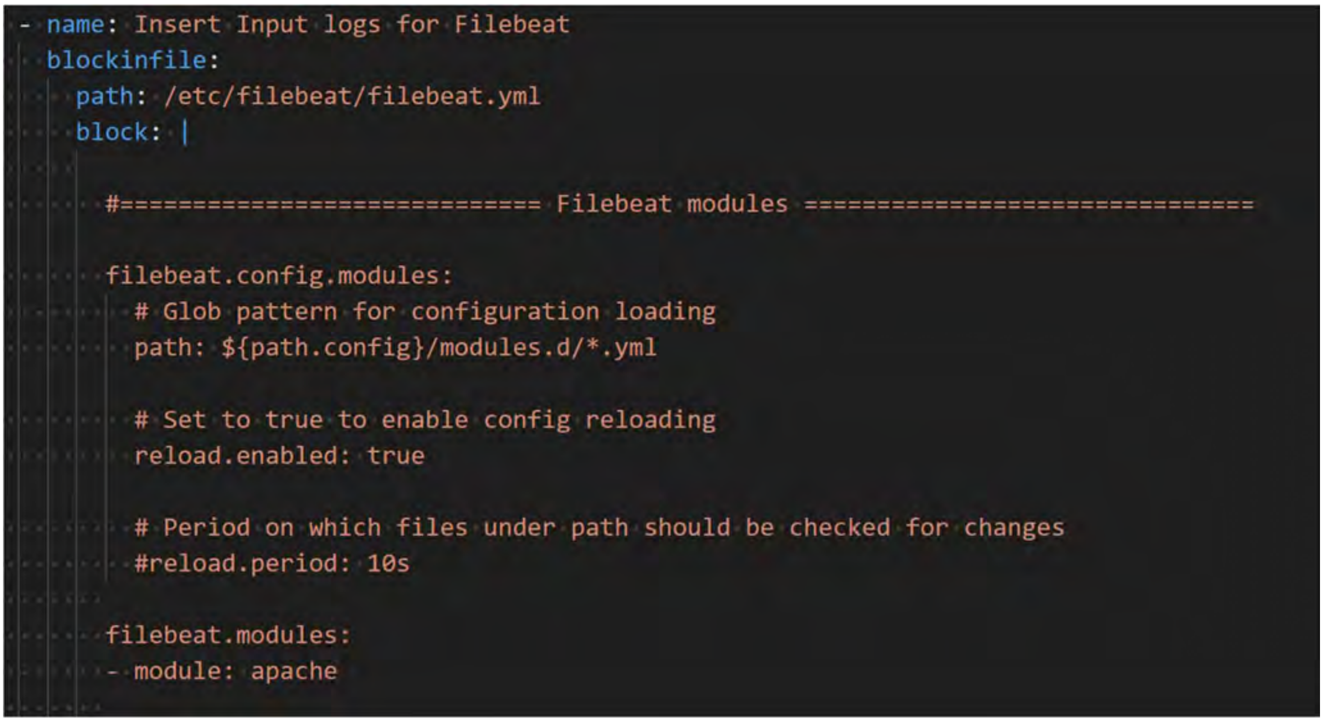
Figure 45 - htaccess configuration

The same as the wpconfig file, you will have to create this file beforehand in order to jump some configurations.

This example allows directory password protection; redirects visitors to a personalized error webpage if they input the wrong authentication information; and allows the use of SSI with '.html' files.

6.3.5 Filebeat

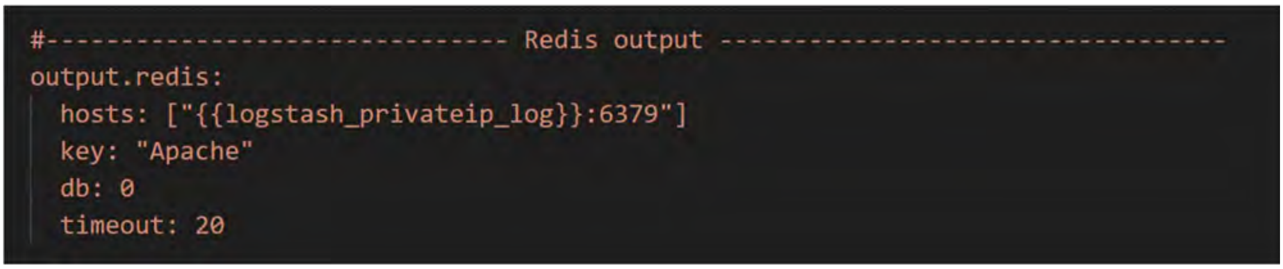
Filebeat configuration is relatively easy, you will only need to modify the filebeat.yml file in the Filebeat directory. There you will be able to modify all the inputs, logs, and outputs. In this scenario we will only enable the apache module.



```
- name: Insert Input logs for Filebeat
  blockinfile:
    path: /etc/filebeat/filebeat.yml
    block: |

      #============================= Filebeat modules ===============================

      filebeat.config.modules:
        # Glob pattern for configuration loading
        path: ${path.config}/modules.d/*.yml

        # Set to true to enable config reloading
        reload.enabled: true

        # Period on which files under path should be checked for changes
        #reload.period: 10s

      filebeat.modules:
      - module: apache
```

Figure 46 - Filebeat configuration

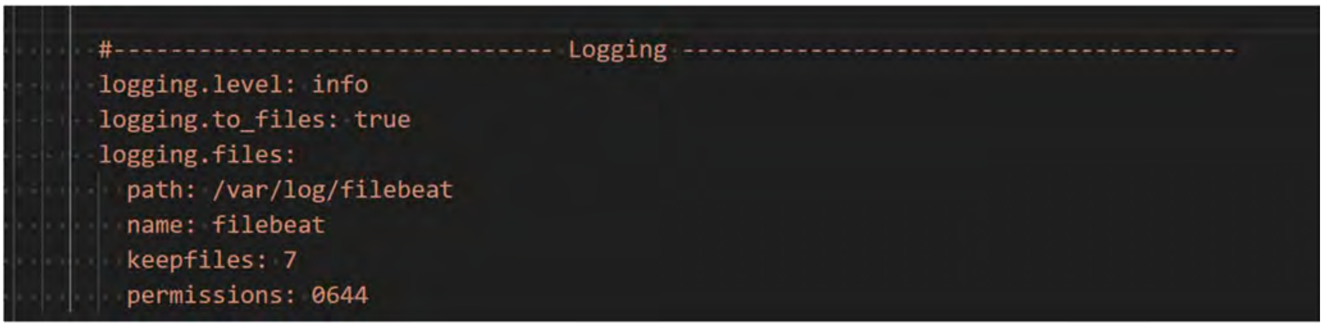
One important configuration that needs to be done is that in order to use Redis to receive and buffer the log data, you need to configure this output the following way:



```
#------------------------------- Redis output ---------------------------------
output.redis:
  hosts: ["{{logstash_privateip_log}}:6379"]
  key: "Apache"
  db: 0
  timeout: 20
```

Figure 47 – Filebeat Output configuration

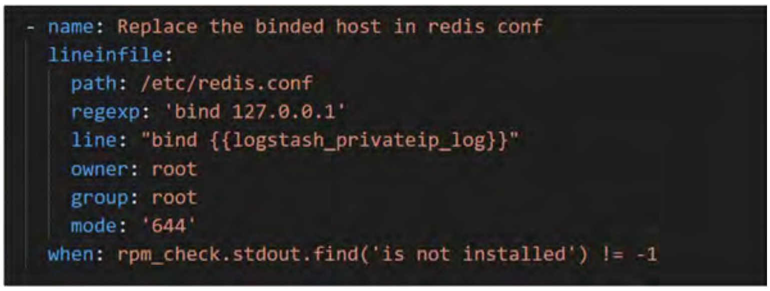
To tweak the logging of the filebeat you will need to modify the logging section in the Filebeat.yml file in the filebeat directory as shown below.



```
#-------------------------------- Logging ---------------------------------------
logging.level: info
logging.to_files: true
logging.files:
  path: /var/log/filebeat
  name: filebeat
  keepfiles: 7
  permissions: 0644
```

Figure 48 - Filebeat logging configuration

6.3.6 Redis



```
- name: Replace the binded host in redis conf
  lineinfile:
    path: /etc/redis.conf
    regexp: 'bind 127.0.0.1'
    line: "bind {{logstash_privateip_log}}"
    owner: root
    group: root
    mode: '644'
  when: rpm_check.stdout.find('is not installed') != -1
```

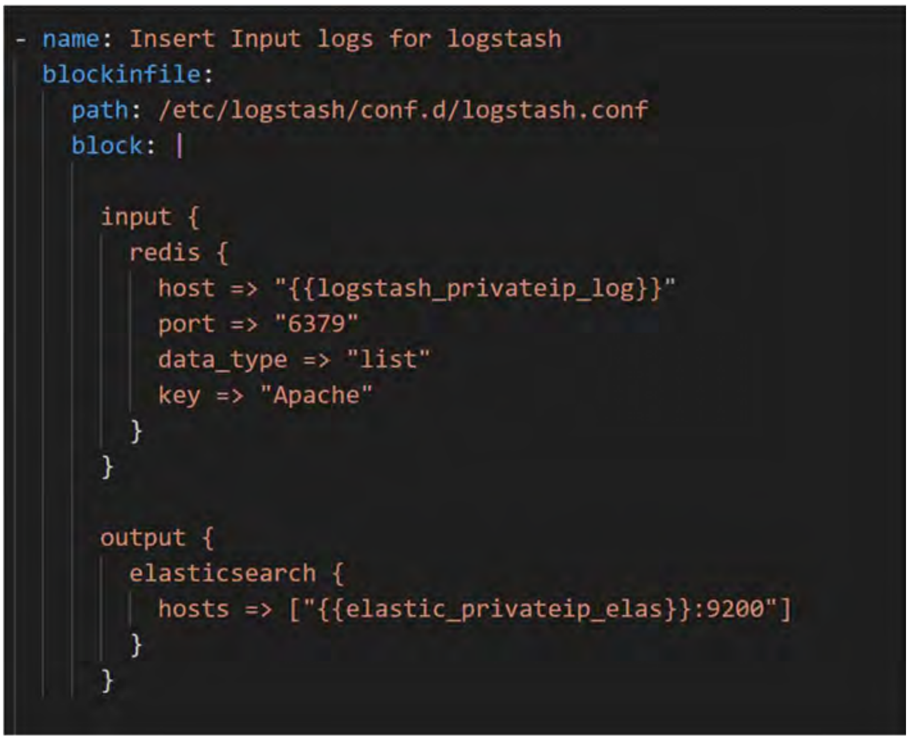
Figure 49 - Redis configuration

Redis configuration is really easy and straight forward, you only need to tell Redis service to listen through an interface by binding it.

6.3.7 Logstash

Logstash configuration can be as easily configured for it to work but here you can play around with Logstash creating custom configurations for indexing information, configure different inputs and outputs, modifying how the data structure is created and many more.

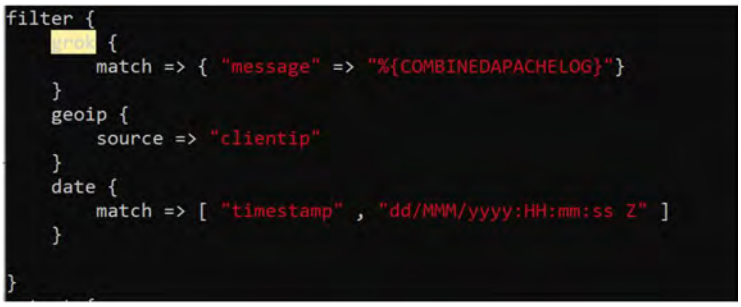
In this PoC we have kept it simple in order to show the full ELK stack but Logstash by itself is a whole big world, but when you have your configuration ready you only need to paste it in this file and you are done.



```
- name: Insert Input logs for logstash
  blockinfile:
    path: /etc/logstash/conf.d/logstash.conf
    block: |

      input {
        redis {
          host => "{{logstash_privateip_log}}"
          port => "6379"
          data_type => "list"
          key => "Apache"
        }
      }

      output {
        elasticsearch {
          hosts => ["{{elastic_privateip_elas}}:9200"]
        }
      }
```

Figure 50 - Logstash configuration

Logstash has an ample list of filters which performs intermediary processing on an event. Filters are often applied conditionally depending on the characteristics of the event. In our scenario we will not be applying such filters since the filebeat module will be applying them. But an example of these filters for apache could be the following:

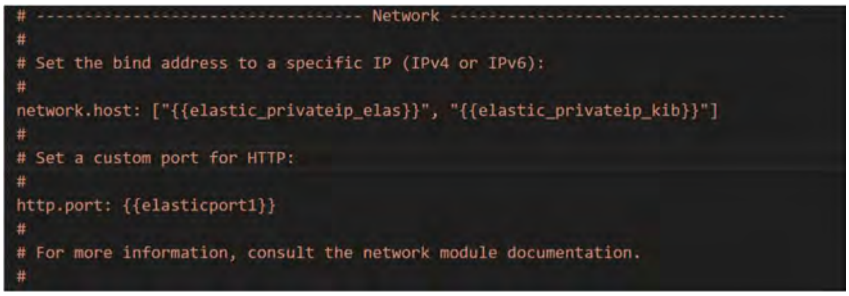


```
filter {
    grok {
        match => { "message" => "%{COMBINEDAPACHELOG}"}
    }
    geoip {
        source => "clientip"
    }
    date {
        match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
    }

}
```

Figure 51 - Logstash Filter

6.3.8 ElasticSearch

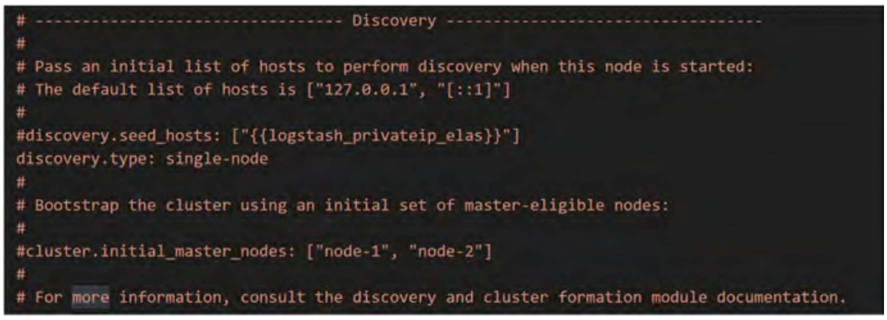
The ElasticSearch configuration as the configuration of Logstash is on another level, you can micromanage a lot of different configurations and ways of working but I will only be showing the basics and the ones used in this PoC, with only one Elastic cluster.



```
# ---------------------------------- Network -----------------------------------
#
# Set the bind address to a specific IP (IPv4 or IPv6):
#
network.host: ["{{elastic_privateip_elas}}", "{{elastic_privateip_kib}}"]
#
# Set a custom port for HTTP:
#
http.port: {{elasticport1}}
#
# For more information, consult the network module documentation.
#
```

Figure 52 - Elastic network configuration

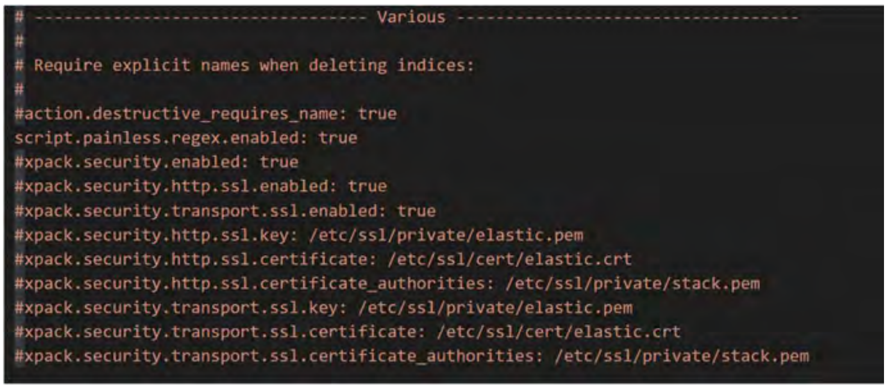
You will need to configure the network interfaces where the elastic will be listening and also de port.



```
# --------------------------------- Discovery ----------------------------------
#
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
#discovery.seed_hosts: ["{{logstash_privateip_elas}}"]
discovery.type: single-node
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
#cluster.initial_master_nodes: ["node-1", "node-2"]
#
# For more information, consult the discovery and cluster formation module documentation.
```

Figure 53 - Elastic Discovery Configuration

In this section is where you will need to configure the architecture of the elastic. In our case is only one node soy you will need to configure it this way. I order to form a cluster you will have to define the roles and the cluster in this file for each elasticsearch instance.



```
# ---------------------------------- Various -----------------------------------
#
# Require explicit names when deleting indices:
#
#action.destructive_requires_name: true
script.painless.regex.enabled: true
#xpack.security.enabled: true
#xpack.security.http.ssl.enabled: true
#xpack.security.transport.ssl.enabled: true
#xpack.security.http.ssl.key: /etc/ssl/private/elastic.pem
#xpack.security.http.ssl.certificate: /etc/ssl/cert/elastic.crt
#xpack.security.http.ssl.certificate_authorities: /etc/ssl/private/stack.pem
#xpack.security.transport.ssl.key: /etc/ssl/private/elastic.pem
#xpack.security.transport.ssl.certificate: /etc/ssl/cert/elastic.crt
#xpack.security.transport.ssl.certificate_authorities: /etc/ssl/private/stack.pem
```

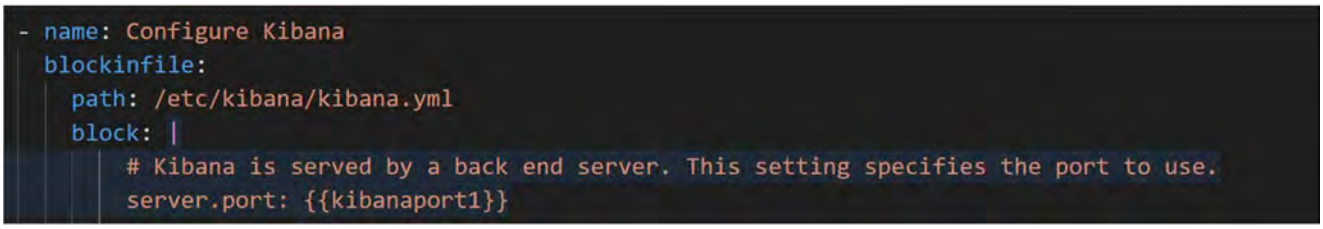
Figure 54 - Elastic SSL

This is the configuration you will need to add in order to establish secure connections with other instances.

In our PoC this has been configured but not used due to a limitation of the User and Key management that will be discussed further on.

6.3.9 Kibana

The Kibana configuration for it to start correctly it is very simple, you only need to configure the server port. This doesn't mean that you can't configure anything else, on the contrary, Kibana works the same way as ElasticSearch and Logstash, it has plenty of different configurations, but we are sticking with the basics.



```
- name: Configure Kibana
  blockinfile:
    path: /etc/kibana/kibana.yml
    block: |
        # Kibana is served by a back end server. This setting specifies the port to use.
        server.port: {{kibanaport1}}
```

Figure 55 - Kibana configuration

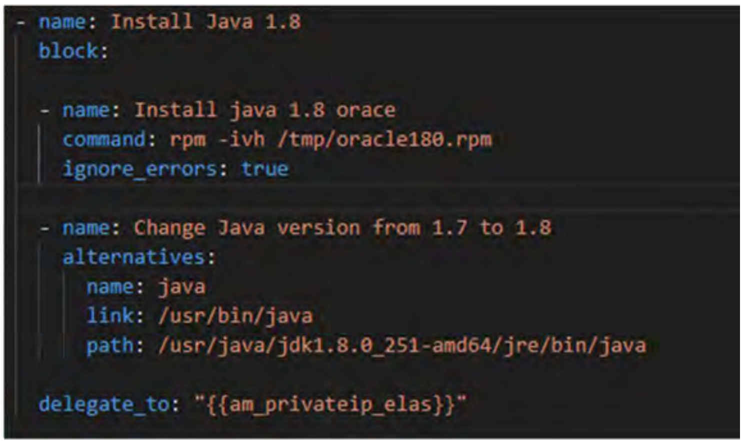
6.3.10 AM 6.5

AM6.5 is a web application that needs to run on tomcat 7/8.5/9 and needs Java 8/11. These are the basic requirements for AM6.5 to run in a machine. In order to deploy AM we need to configure and install these applications.

6.3.10.1 Installing Java Oracle or OpenJDK

Regarding the java distro I recommend using the Oracle java 1.8 version instead of the OpenJDK, this is due to the fact that Forgerock, AM's developer, has compiled the java application with the Oracle version and this has proven to be important during my experience with the product.

The disadvantage of using Oracle's java is that you have to pre download it and upload it to the instance.



```
- name: Install Java 1.8
  block:

  - name: Install java 1.8 orace
    command: rpm -ivh /tmp/oracle180.rpm
    ignore_errors: true

  - name: Change Java version from 1.7 to 1.8
    alternatives:
      name: java
      link: /usr/bin/java
      path: /usr/java/jdk1.8.0_251-amd64/jre/bin/java

  delegate_to: "{{am_privateip_elas}}"
```

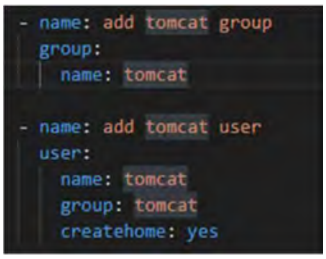
Figure 56 - Java Installation

One important thing to do after installing java is to configure it to be the main java version for the instance, you can do this by using the alternatives command in linux.

6.3.10.2 Installing Tomcat and configuring the environment

Before installing Tomcat you need to set up the user environment:

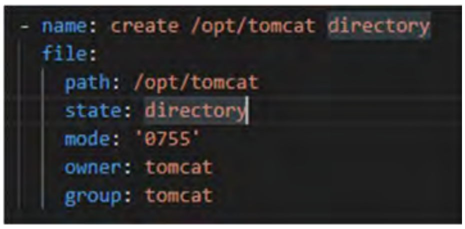
- Create tomcat user and group



```
- name: add tomcat group
  group:
    name: tomcat

- name: add tomcat user
  user:
    name: tomcat
    group: tomcat
    createhome: yes
```

Figure 57 - User and group creation

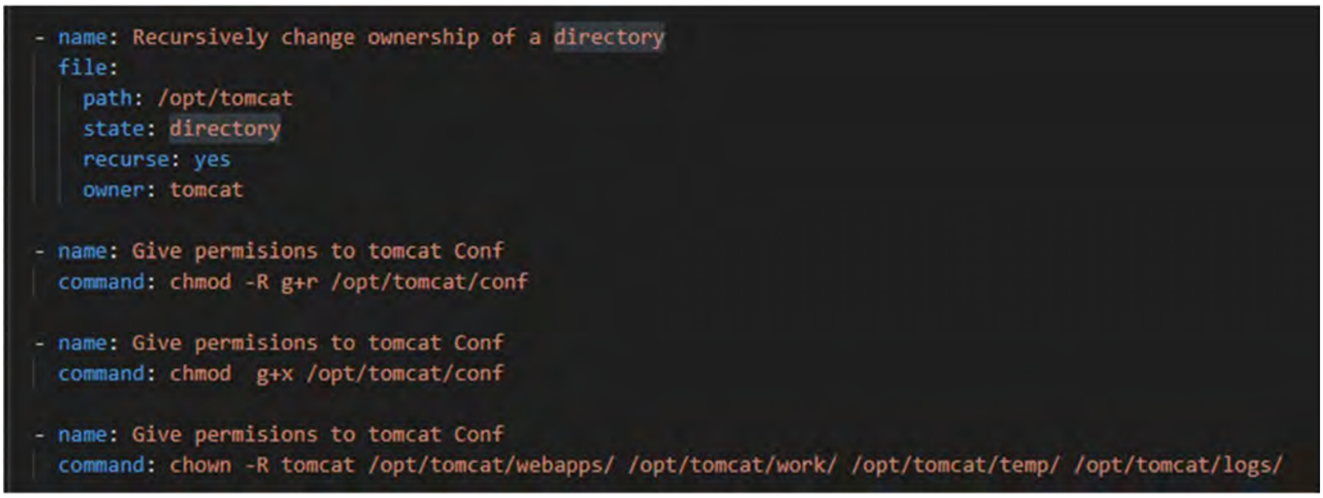
- Create opt directory



```
- name: create /opt/tomcat directory
  file:
    path: /opt/tomcat
    state: directory
    mode: '0755'
    owner: tomcat
    group: tomcat
```

Figure 58 - Directory creation

Once you have unzipped the application to /opt/tomcat, you will need to change some folder permissions:

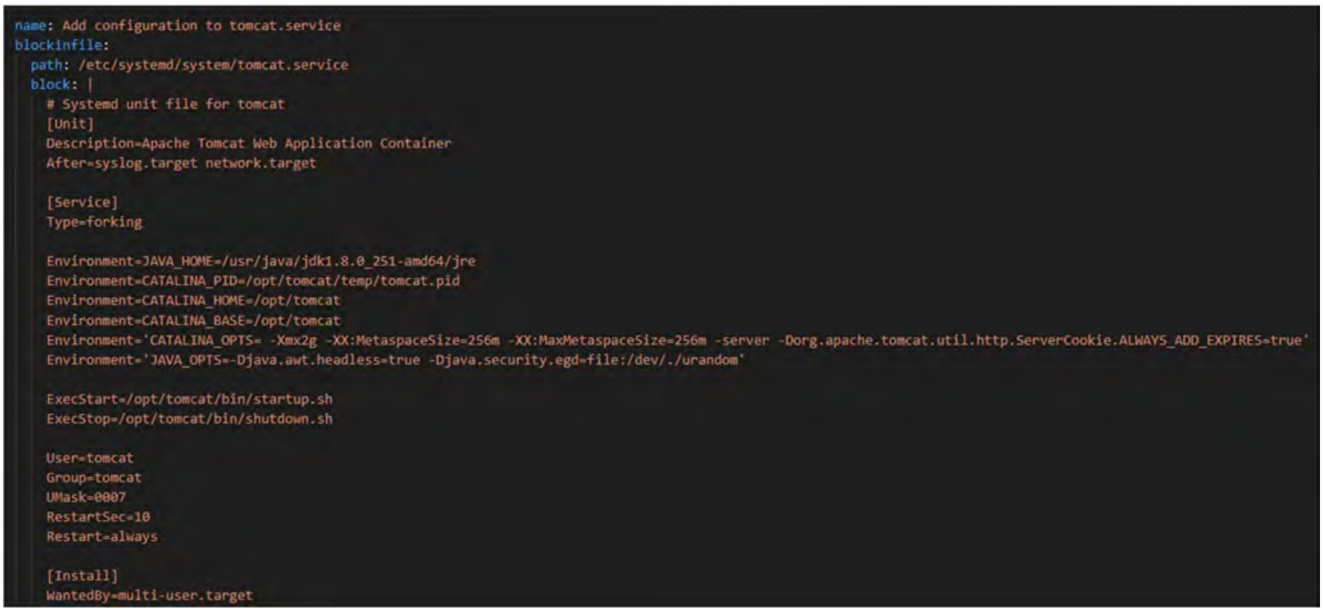


```
- name: Recursively change ownership of a directory
  file:
    path: /opt/tomcat
    state: directory
    recurse: yes
    owner: tomcat

- name: Give permisions to tomcat Conf
  command: chmod -R g+r /opt/tomcat/conf

- name: Give permisions to tomcat Conf
  command: chmod  g+x /opt/tomcat/conf

- name: Give permisions to tomcat Conf
  command: chown -R tomcat /opt/tomcat/webapps/ /opt/tomcat/work/ /opt/tomcat/temp/ /opt/tomcat/logs/
```

Figure 59 - Folder permissions

Last two steps are to create the service file and to modify the tomcat user file in order to create a user for the manager section of tomcat.

Let’s start with the service file that will be added to the systemd directory so each time tomcat is launched it will start with a defined configuration.

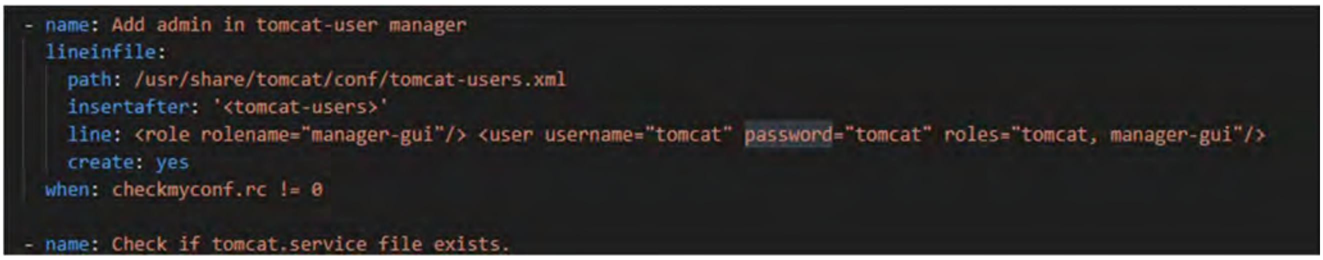


```
name: Add configuration to tomcat.service
blockinfile:
  path: /etc/systemd/system/tomcat.service
  block: |
    # Systemd unit file for tomcat
    [Unit]
    Description=Apache Tomcat Web Application Container
    After=syslog.target network.target

    [Service]
    Type=forking

    Environment=JAVA_HOME=/usr/java/jdk1.8.0_251-amd64/jre
    Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
    Environment=CATALINA_HOME=/opt/tomcat
    Environment=CATALINA_BASE=/opt/tomcat
    Environment='CATALINA_OPTS= -Xmx2g -XX:MetaspaceSize=256m -XX:MaxMetaspaceSize=256m -server -Dorg.apache.tomcat.util.http.ServerCookie.ALWAYS_ADD_EXPIRES=true'
    Environment='JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'

    ExecStart=/opt/tomcat/bin/startup.sh
    ExecStop=/opt/tomcat/bin/shutdown.sh

    User=tomcat
    Group=tomcat
    UMask=0007
    RestartSec=10
    Restart=always

    [Install]
    WantedBy=multi-user.target
```

Figure 60 -- Tomcat.Service file

In this file is where you can modify and tune the JVM parameters to best fit the AM app. In this case I have only added the configurations recommended by the provider.

And last but not least, the tomcat users file where you can create your user and password to enter the management area of Tomcat



```
- name: Add admin in tomcat-user manager
  lineinfile:
    path: /usr/share/tomcat/conf/tomcat-users.xml
    insertafter: '<tomcat-users>'
    line: <role rolename="manager-gui"/> <user username="tomcat" password="tomcat" roles="tomcat, manager-gui"/>
    create: yes
  when: checkmyconf.rc != 0

- name: Check if tomcat.service file exists.
```

Figure 61 -- Tomcat-user configuration

6.4 ANSIBLE LIMITATIONS

Ansible has limitations as any other automation software, it also has a lot of positive things, but I think you have already seen that if you have reached this section without jumping directly here.

One of the Cons, is that once a playbook is running, you can't add or remove hosts from inventory file, this can be crucial if your use case is such that you have a real time code which generates IP addresses where the same script is supposed to run.

The same thing with hosts, is with variables, you cannot add variables to global vars file at runtime which limits a lot your playbook and style of coding. You can save variables temporarily but those will disappear once the playbook is finished; you won't keep all the new variables generated in that run in the var file.

Ansible makes fresh connection to remote hosts for each module it executes. The disadvantage of this is that making so many connection attempts makes it prone to connection failures. And one connection failure can put the execution of the entire play in jeopardy.

Error reporting is not great to say the least. If your playbook fails because of some syntax issue, ansible points you to the line number and adds a generic syntax issue msg. You need to figure out where exactly the issue is, is it a semi colon or parenthesis or space. This can be frustrating.

Another tedious thing about Ansible is the fact that you cannot give tags to whole roles without blocking the rest of the code that its outside the roles, like pre-tasks before declaration of the roles.

Finally is that Ansible solely relies on the modules that are implemented, creating limitations for different configurations, in this case, using new AWS endpoints or configurations.

So far those are the most relevant problems that I have found playing around with Ansible, but I guess you will find many more since this is a new open source tool and still needs a lot of work to be done.

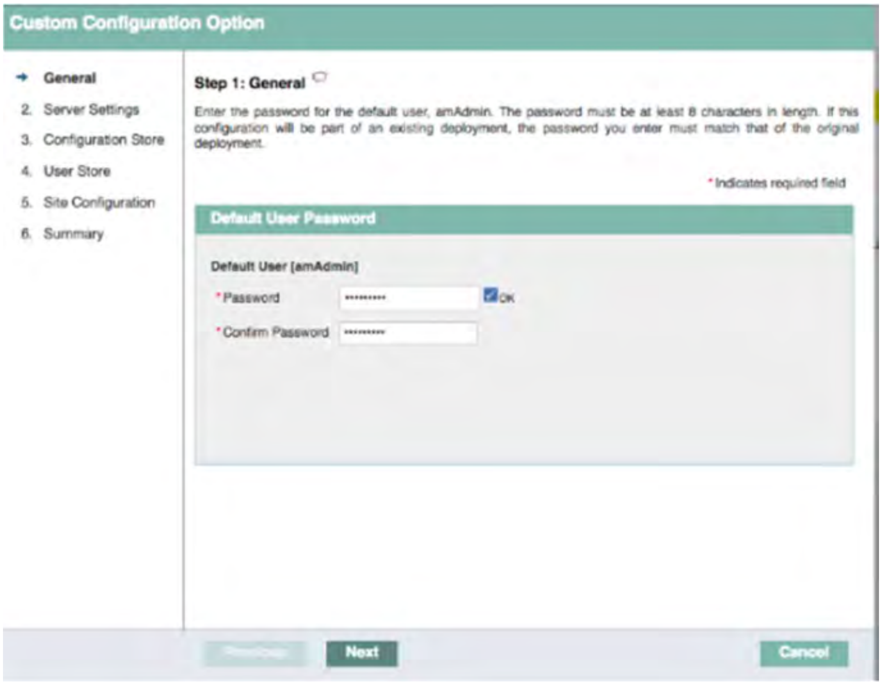
7 MANUAL CONFIGURATIONS

Due to ansible's limitations some configurations must be hand configured, in the case of the VPN and Route 53, ansible has some modules but not all those needed to configure them easily.

Regarding AM6.5 there are no modules but Forgerock has a automation tool called Amster, in this case all of the configurations below can be done with Amster. In my case I started automating the deployment with Amster but was unable to fix an internal bug.

7.1 CONFIGURING AM6.5 (CUSTOM CONFIGURE MANUALLY)

- In the initial configuration screen, click Create New Configuration under Custom Configuration.
- Read the license agreement. If you agree to the license, click "I agree to the license agreement", and then click Continue.
- On the Default User Password page, provide a password with at least eight characters for the AM Administrator, amadmin.



Step 1: General

Enter the password for the default user, amAdmin. The password must be at least 8 characters in length. If this configuration will be part of an existing deployment, the password you enter must match that of the original deployment.

*Indicates required field

Default User Password

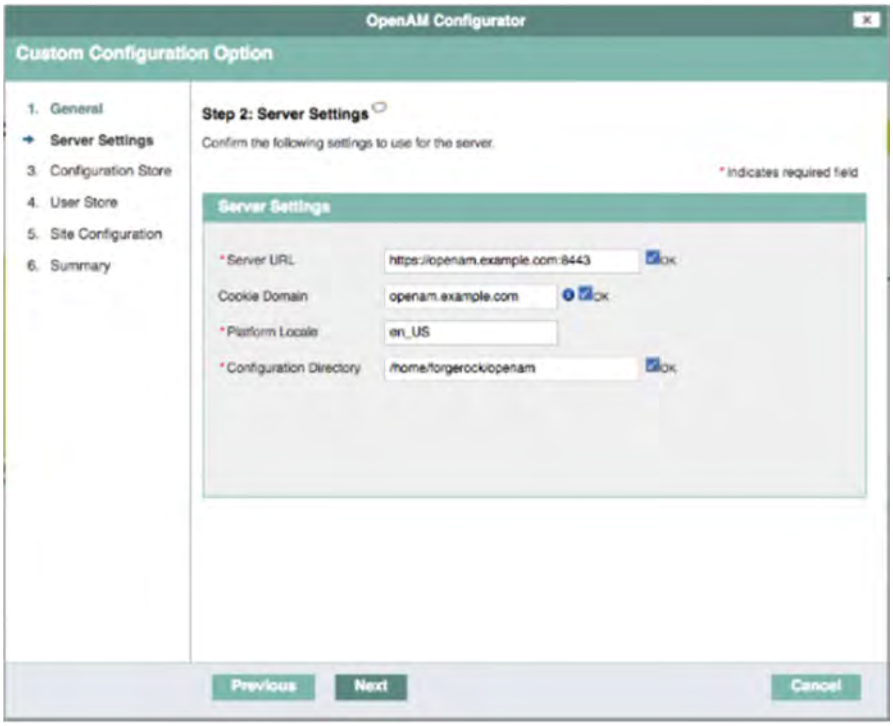
Default User [amAdmin]

*Password ********** ☑ OK

*Confirm Password **********

Figure 62 - AM configurator Step1

- Verify that the server settings are valid for your configuration.



Step 2: Server Settings

Confirm the following settings to use for the server.

* indicates required field

Server Settings

- * Server URL: https://openam.example.com:8443 ☑ OK
- Cookie Domain: openam.example.com ☑ OK
- * Platform Locale: en_US
- * Configuration Directory: /home/forgerock/openam ☑ OK

Figure 63 - Figure 59 - AM configurator Step2

Server URL

Provide a valid URL to the base of your AM web container, including a FQDN.

In a test environment, you can simulate the FQDN by adding it to your /etc/hosts as an alias. The following excerpt shows lines from the /etc/hosts file on a Linux system where AM is installed.

Cookie Domain

Domain that created cookies will be valid for, for example example.com.

- In the Configuration Store screen, you can accept the defaults to allow AM to store configuration data in an embedded directory.
- In the User Store screen, you configure where AM looks for user identities. In this case use Embedded.
- In the Site Configuration screen, you can set up AM as part of a site where the load is balanced across multiple AM servers. We will not need this option during this deployment.
- Finish.

Now that you have installed AM6.5 with the embedded option, you can configure the agent.

7.1.1 Web Agent configuration

7.1.1.1 Configuring the agent in AM

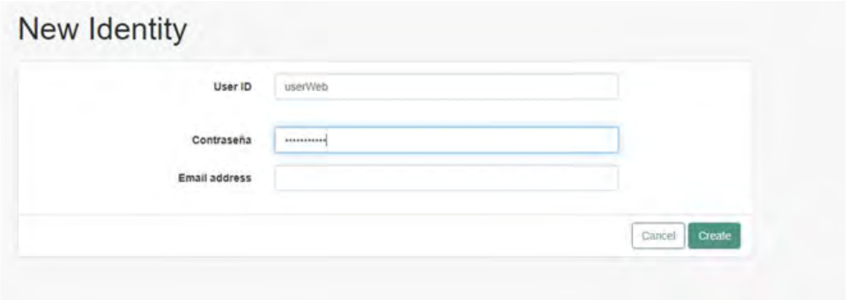
At first you have to log-in to the OpenAM console as amadmin user. Some configuration steps in this part are redundant to the previous section, however I rather keep it there to make sure, you don’t miss some important settings:

- User Name: amAdmin
- Password: forgerock
- Log In

Realms -> /TFM

Identities -> Add Group

Identities -> Add Identity



New Identity

User ID: userWeb

Contraseña: ••••••••••

Email address:

Figure 64 -- Creating new Identity

Identities -> User -> Add Group

Applications -> Agents -> Web -> Agent -> Web -> New…

- Name: webagent
- Password: webagent
- Re-Enter Password: webagent
- Server URL: https://openam.example.com:8080/openam
- Agent URL: https://url-servicio:port
- Create

Authorization -> Policies -> New Policy Set

- Name: Protected Resource
- Resource Types: Url

Authorization -> Policies -> Protected Resource -> Add Policy

- Name: Apache Web Server
- Resource Type: URL

- Resources: *//:*.*:*
- https://url:port/login

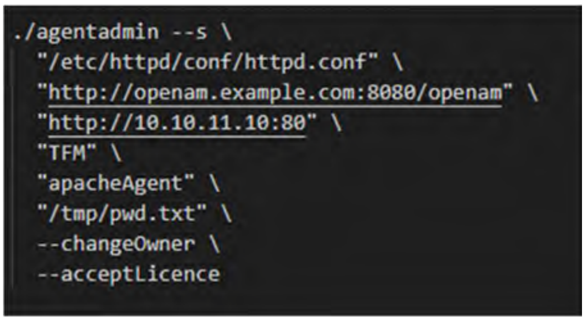
Authorization -> Policies -> Protected Resource -> Apache Web Server -> Subjects -> Add Subject Condition

- Users & Groups
- Authenticated

7.1.1.2 Deploying the agent

To deploy AM web agent, you need to deploy the webagent into the home directory of Apache and proceed to install the agent following the instructions that the installation menu provides.

Another possibility is to install the agent, once it is unzipped in the home directory, is to install it with all the options at once.



```
./agentadmin --s \
  "/etc/httpd/conf/httpd.conf" \
  "http://openam.example.com:8080/openam" \
  "http://10.10.11.10:80" \
  "TFM" \
  "apacheAgent" \
  "/tmp/pwd.txt" \
  --changeOwner \
  --acceptLicence
```

Figure 65 - Agent installation configuration

Once you have done all the steps above, you will be redirected to the AM login web page where you will authenticate before being able to access the webpage.

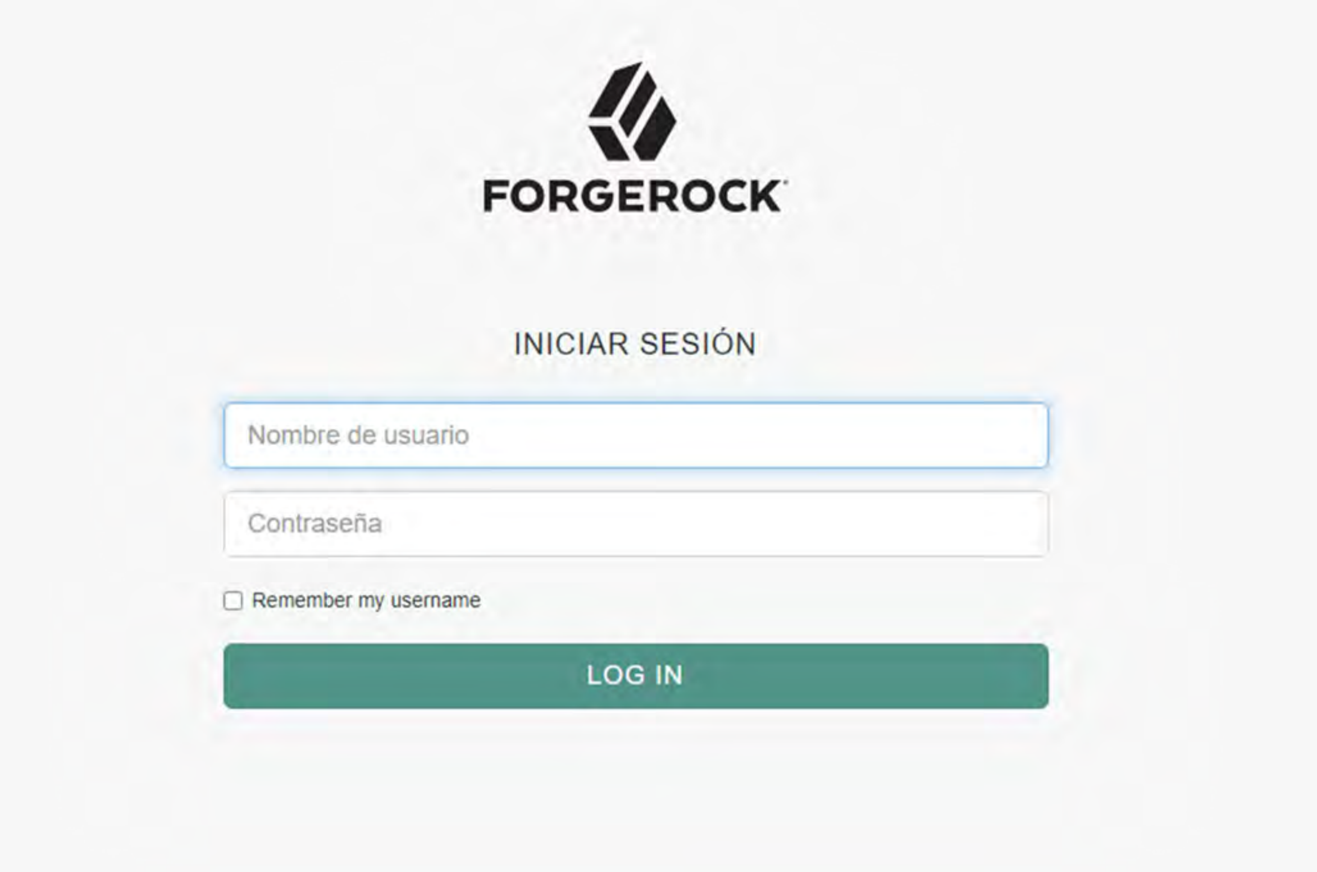


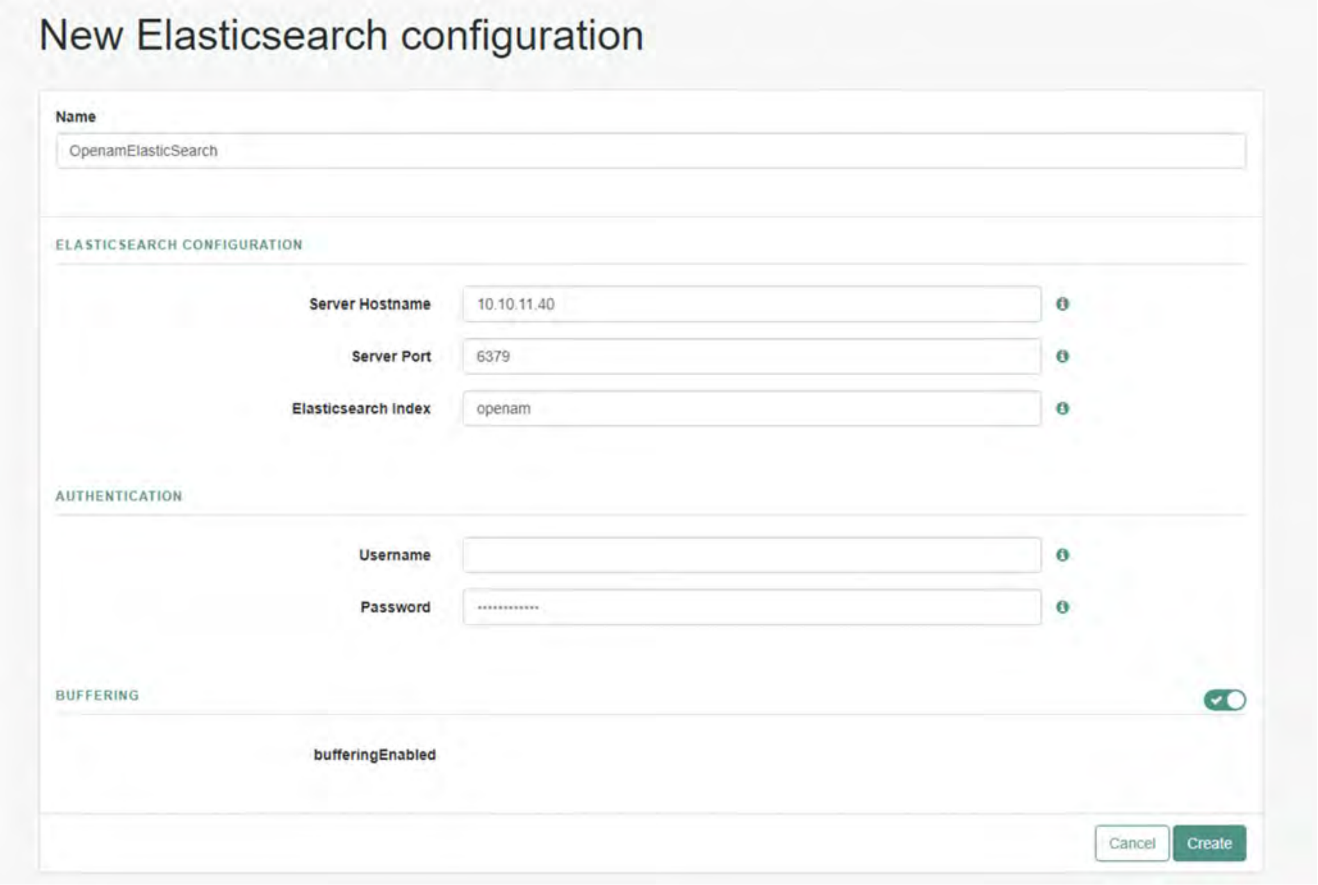
Figure 66 -- AM login

7.1.2 Configure Elasticsearch as output for Audit logs AM6.5

To configure Elasticsearch as an indexer for the logs of openam you need to go to:

Global Settings -> Audit Logging -> Secondary Configurations -> Add new configuration

Select Elastic Search in the dropdown menu and fill in with the values from your topology as shown below.



New Elasticsearch configuration

Name
OpenamElasticSearch

ELASTICSEARCH CONFIGURATION

Server Hostname: 10.10.11.40

Server Port: 6379

Elasticsearch Index: openam

AUTHENTICATION

Username:

Password: ••••••••••••

BUFFERING

bufferingEnabled

Figure 67 -- Elasticsearch configuration in AM6.5

7.2 CLIENT VPN CONFIGURATION

Using Ansible in AWS relies on modules that interoperate directly with AWS endpoints making this one of the drawbacks of Ansible, since the only way of interacting with the cloud is through these made modules. Creating your own modules for Ansible is not an easy job for the less experienced users making this an impossible task.

Regarding the creation of the VPN client configuration, we have reached this standpoint situation where we would have to write the python script or use another tool for configuring cloud infrastructures or do it manually.

7.2.1 VPC configuration

Firstly, we must create a VPC for the company's users and the VPC peering as I have shown previously. Why should you isolate the clients on its own VPC? For security reasons, by separating logically from the network you are able to have more control over the networks that they can access and if needed be, suspend the VPC peering in case of an intrusion, leaving the attacker in a network that has no access to no nodes.

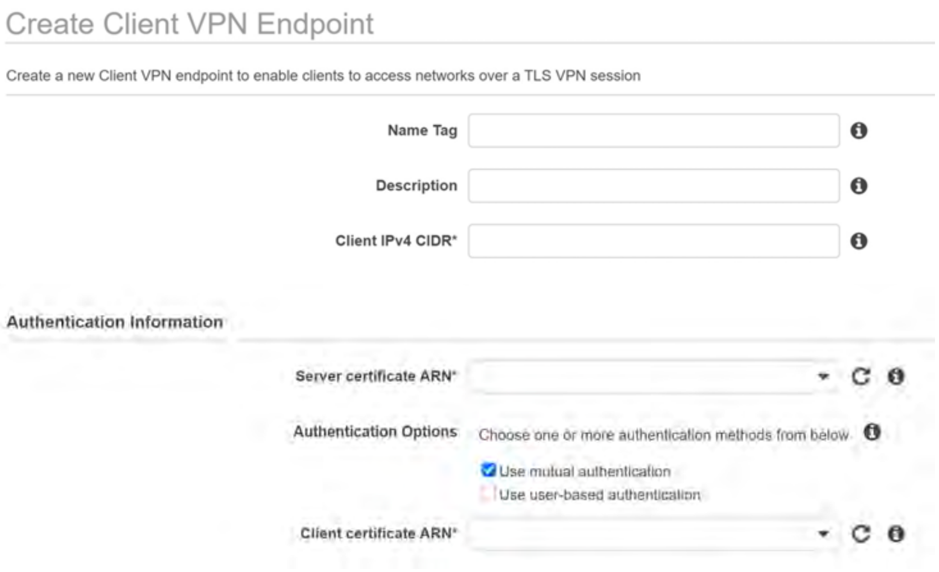
Once you have configured the VPC network and the VPC peering connection it is time to configure the route tables and if needed be the network ACLs. Since this VPN will be for the company's users, they will have access to the web servers and the kibana servers. This is something that must be limited in the network ACL.

7.2.2 Manual Client VPN configuration

For the manual configuration of the VPN, firstly you will have to generate the server & client certificates and keys. This is done by using openrsa or openssl application in Linux terminal. To do so follow these easy steps:

https://docs.aws.amazon.com/vpn/latest/clientvpn-admin/client-authentication.html#mutual

Next you will have to create a client VPN Endpoint where you will have to provide a Client IPv4 CIDR from which client IP addresses are allocated and the server and client certificates created previously.



Create Client VPN Endpoint

Create a new Client VPN endpoint to enable clients to access networks over a TLS VPN session

Name Tag

Description

Client IPv4 CIDR*

Authentication Information

Server certificate ARN*

Authentication Options: Choose one or more authentication methods from below

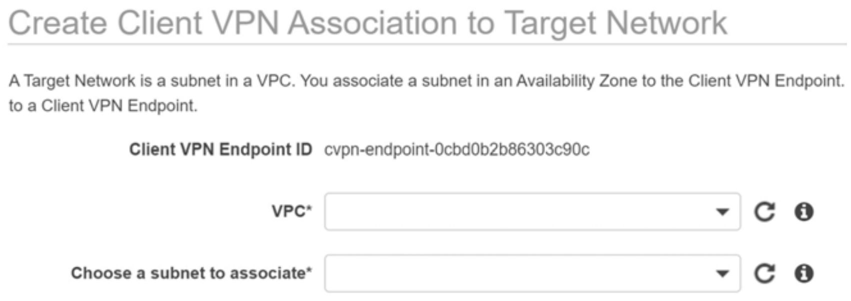
☑ Use mutual authentication

☐ Use user-based authentication

Client certificate ARN*

Figure 68 - Client VPN Endpoint Configuration

After creating the VPN configuration, you will have to create an association with a VPC and the subnet that will be associated with the endpoint.



Create Client VPN Association to Target Network

A Target Network is a subnet in a VPC. You associate a subnet in an Availability Zone to the Client VPN Endpoint. to a Client VPN Endpoint.

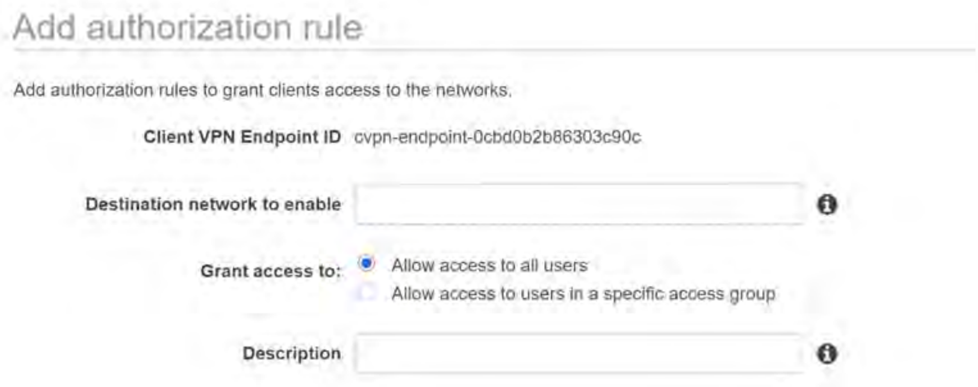
Client VPN Endpoint ID cvpn-endpoint-0cbd0b2b86303c90c

VPC*

Choose a subnet to associate*

Figure 69 - Client VPN Association Configuration

Finally you need to authorize users to connect to a specific network, this is done by adding the authorization rule to the network that you want to give access to the users that connect through a VPN.



Add authorization rule

Add authorization rules to grant clients access to the networks.

Client VPN Endpoint ID cvpn-endpoint-0cbd0b2b86303c90c

Destination network to enable

Grant access to: ● Allow access to all users ○ Allow access to users in a specific access group

Description

Figure 70 - Authorization rule configuration

For a more detailed explanation on how to create your own VPN, follow these instructions:

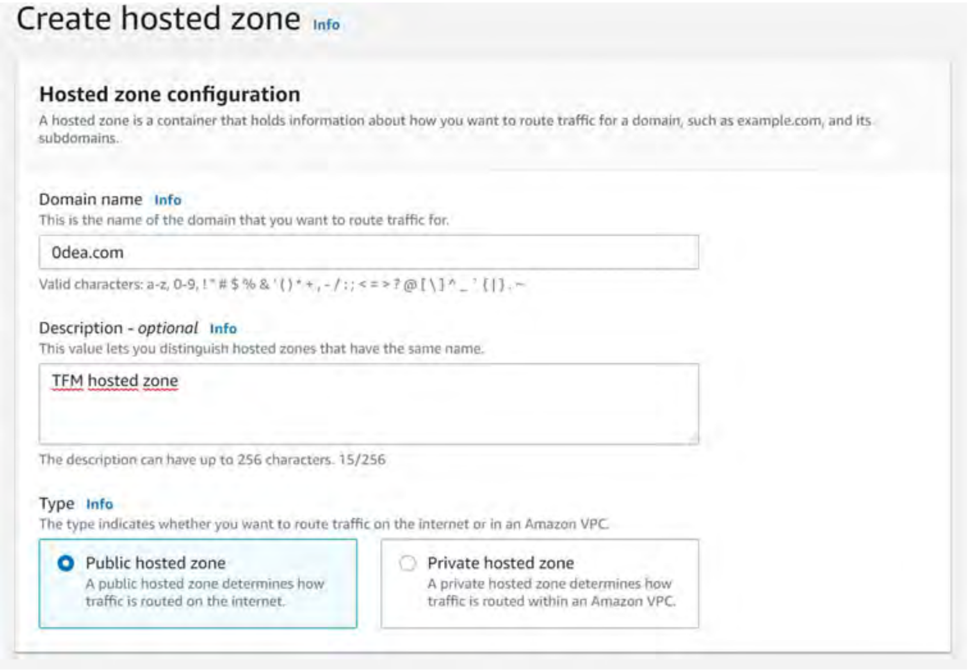
https://docs.aws.amazon.com/vpn/latest/clientvpn-admin/cvpn-getting-started.html#cvpn-getting-started-certs

7.3 Route 53 AWS DNS service

Route 53 is a scalable and highly available Domain Name System service for AWS. Here you can register new public domains and private domains. If you don't have a domain name you can create one and host it in AWS or if you have one you can import the zone file of the domain name.

In the latter case, you will have to create a hosted zone in AWS and import the zone file. What is a zone file? A zone file is a text file that describes a DNS zone. The zone file contains mappings between domain names and IP addresses and other resources, organized in the form of text representations of resource records.

First, we will create a hosted zone for the domain 0dea.com and we configure a public hosted zone:



Create hosted zone Info

Hosted zone configuration

A hosted zone is a container that holds information about how you want to route traffic for a domain, such as example.com, and its subdomains.

Domain name Info

This is the name of the domain that you want to route traffic for.

0dea.com

Valid characters: a-z, 0-9, ! " # $ % & ' ( ) * + , - / : ; < = > ? @ [\ ] ^ _ ` { | } . ~

Description - *optional* Info

This value lets you distinguish hosted zones that have the same name.

TFM hosted zone

The description can have up to 256 characters. 15/256

Type Info

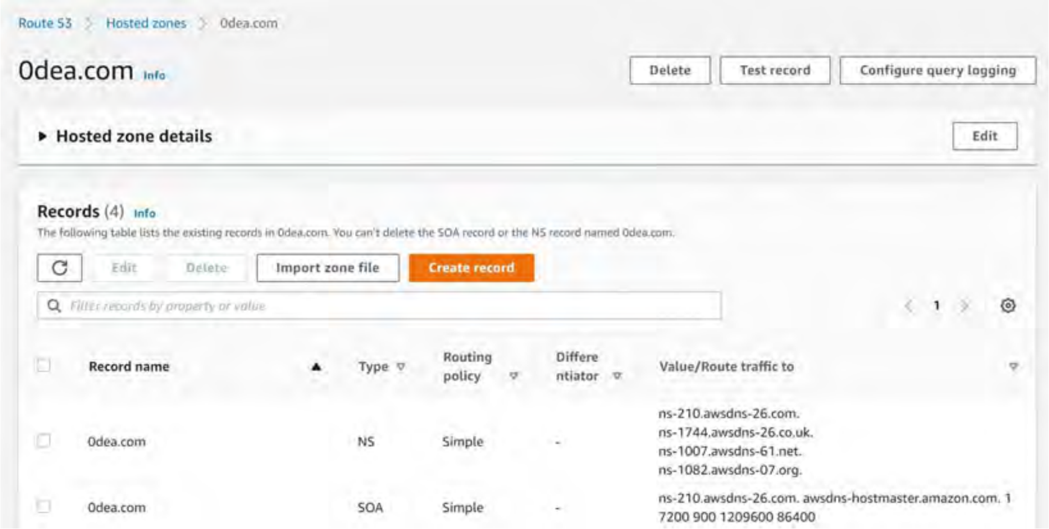
The type indicates whether you want to route traffic on the internet or in an Amazon VPC.

● Public hosted zone
A public hosted zone determines how traffic is routed on the internet.

○ Private hosted zone
A private hosted zone determines how traffic is routed within an Amazon VPC.

Figure 71 - Create Hosted Zone

Next, we will import the zone file, where we will copy the contents of the zone file.



Records (4) Info

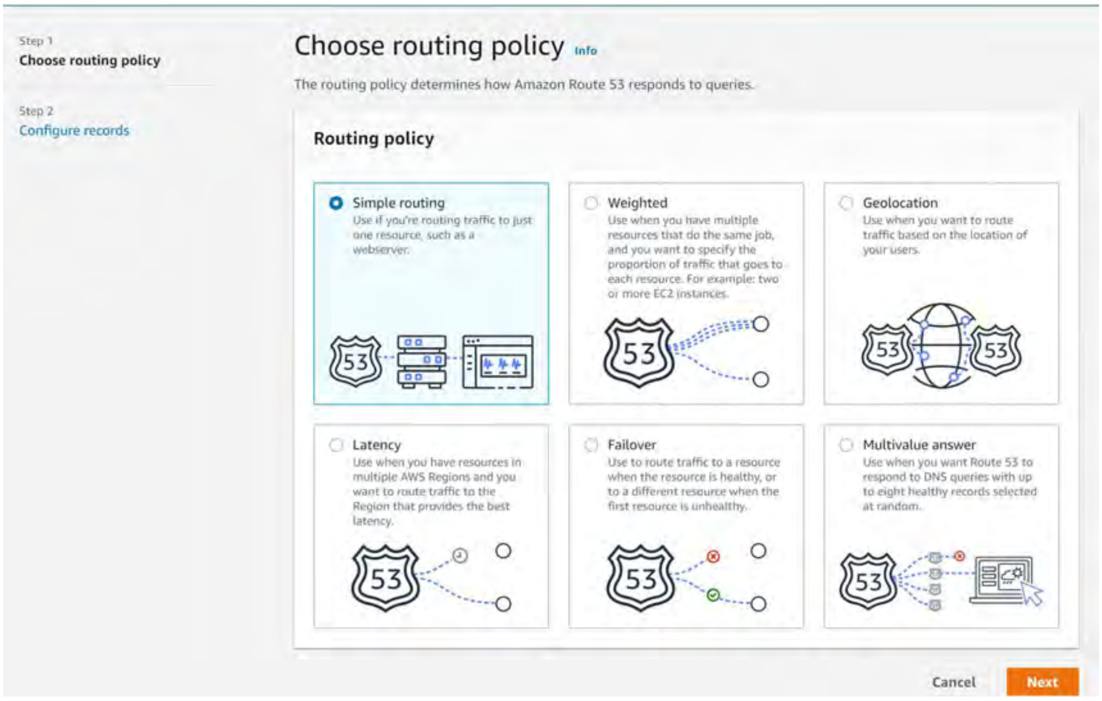
The following table lists the existing records in 0dea.com. You can't delete the SOA record or the NS record named 0dea.com.

□	Record name	Type	Routing policy	Differentiator	Value/Route traffic to
□	0dea.com	NS	Simple	-	ns-210.awsdns-26.com. ns-1744.awsdns-26.co.uk. ns-1007.awsdns-61.net. ns-1082.awsdns-07.org.
□	0dea.com	SOA	Simple	-	ns-210.awsdns-26.com. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400

Figure 72 - Importing Zone file

Once that is done, your domain will be routed to AWS but not to a service. To link the domain name to a service you must create a record, where you can specify a domain name or a subdomain. I will be creating the subdomain web.0dea.com which will be linked to the apache server.

In order to link to a singe machine without a load balancer you have to select a simple routing policy:



Choose routing policy Info

The routing policy determines how Amazon Route 53 responds to queries.

Routing policy

● Simple routing
Use if you're routing traffic to just one resource, such as a webserver.

○ Weighted
Use when you have multiple resources that do the same job, and you want to specify the proportion of traffic that goes to each resource. For example: two or more EC2 instances.

○ Geolocation
Use when you want to route traffic based on the location of your users.

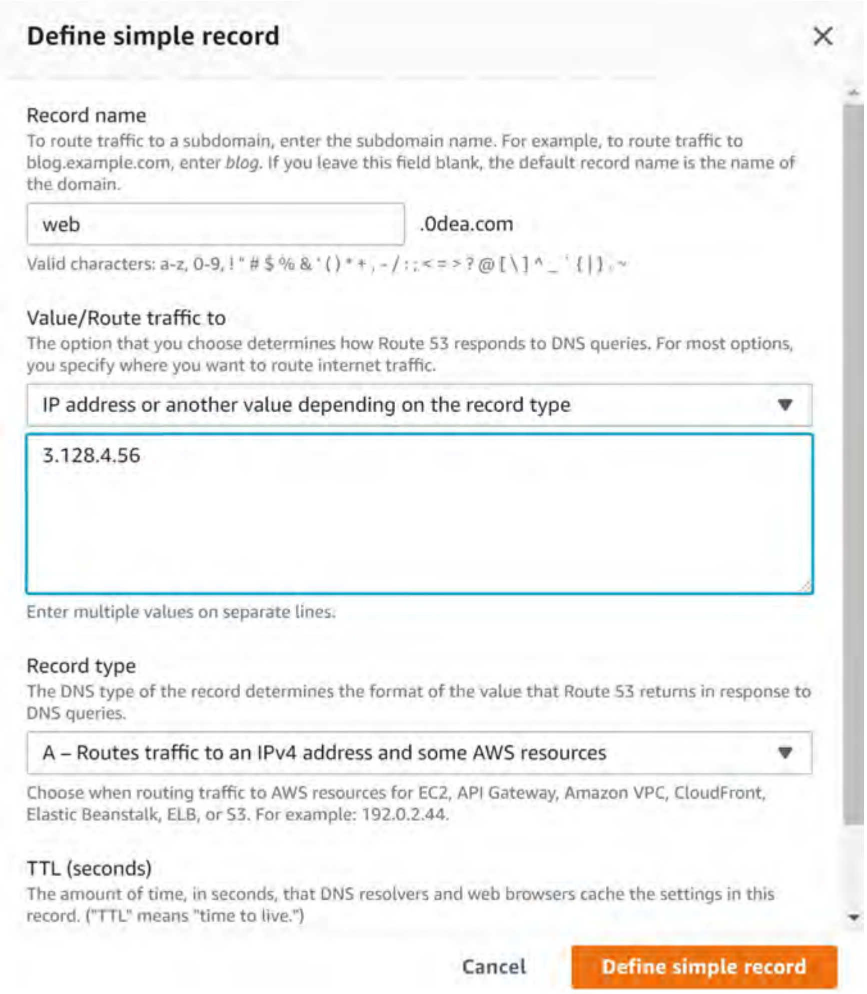
○ Latency
Use when you have resources in multiple AWS Regions and you want to route traffic to the Region that provides the best latency.

○ Failover
Use to route traffic to a resource when the resource is healthy, or to a different resource when the first resource is unhealthy.

○ Multivalue answer
Use when you want Route 53 to respond to DNS queries with up to eight healthy records selected at random.

Figure 73 - Routing Policy

Next, you need to configure the subdomain and the public IP address that will be linked to:



Define simple record

Record name

To route traffic to a subdomain, enter the subdomain name. For example, to route traffic to blog.example.com, enter *blog*. If you leave this field blank, the default record name is the name of the domain.

web .0dea.com

Valid characters: a-z, 0-9, ! " # $ % & ' ( ) * + , - / : ; < = > ? @ [\ ] ^ _ ` { | } . ~

Value/Route traffic to

The option that you choose determines how Route 53 responds to DNS queries. For most options, you specify where you want to route internet traffic.

IP address or another value depending on the record type

3.128.4.56

Enter multiple values on separate lines.

Record type

The DNS type of the record determines the format of the value that Route 53 returns in response to DNS queries.

A – Routes traffic to an IPv4 address and some AWS resources

Choose when routing traffic to AWS resources for EC2, API Gateway, Amazon VPC, CloudFront, Elastic Beanstalk, ELB, or S3. For example: 192.0.2.44.

TTL (seconds)

The amount of time, in seconds, that DNS resolvers and web browsers cache the settings in this record. ("TTL" means "time to live.")

Figure 74 – Creation of a simple recod

Once that is done, your subdomain will be redirected to the server or reverse proxy that you have configured. This change will take up to 24h in order for the DNS name to be routed to your instance.

8 TRICKS AND TIPS

8.1 ACCESSING TOMCAT MANAGER AND AM6.5 FROM OUTSIDE AWS

You may have come upon the impossibility of accessing the “Localhost” or tomcat webpage inside your AWS architecture since the instance is hidden from the internet. Panic not, because you can use our friend SSH and its tunnel abilities to connect directly to your instance through the localhost.

```
ssh -L 80:10.10.11.42:8080 ec2-user@18.195.9.19 -i my_aws.pem  -v
```

When you use this command in your laptop you will be accessing **10.10.11.42:8080** when you write in your browser **localhost:80**, this will forward all traffic to this instance ec2-user@18.195.9.19 that then will forward to this host **10.10.11.42.**

9 FUTURE DEVELOPMENTS

This is a PoC still in its early stages, which means that a lot of ground must be covered to meet a production environment requirement.

9.1 USER & KEY MANAGEMENT

During the development of this PoC we have made some cuts regarding user and password management. All of the instances and services were installed and raised by the user root. This is not a secure way of doing nor a recommended one. I have only taken that route for simplicity of management and creation of users and keys.

As a future development, this is one of the things that must be taken care of before using this for a production environment. This could be done manually or by automating some scripts for password rotation but if you want to orchestrate this you will need to use a password management tool.

9.1.1 CyberArk [23]

"The CyberArk Core Privileged Access Security Solution is the industry's most complete solution for protecting, controlling, and monitoring privileged access across cloud and hybrid environments. Designed from the ground up for security, the CyberArk solution helps organizations efficiently manage privileged account credentials and access rights, proactively monitor privileged account activity, intelligently identify suspicious activity, and quickly respond to threats."

"Automated rotation of privileged credentials (passwords and SSH keys) and/or just-in-time privileged access eliminates time-consuming and error-prone administrative tasks, safeguarding credentials used in on-premises, hybrid, and cloud environments."

9.2 ENCRYPT INTERNAL COMMUNICATIONS

This is partly implemented, but it is blocked to the previous point. To establish secure internal communications between, for example, ElasticSearch and Kibana, that will require a management of users and SSH keys (self-signed). This will bring a lot of micro-management to the PoC which will defeat the purpose of an automated infrastructure.

9.3 IMPROVED INTERNAL DNS SERVICE

For a production environment the architecture should have an internal DNS service which would help navigate through all the instances and subdomains.

9.4 KUBERNETES ARCHITECTURE

Architectures based on containers, automates the process of scaling, managing, updating and removing containers. In other words, it is a container orchestration platform. This could greatly improve the ELK stack and also the access manager stack for high loads.

9.5 AM6.5 DEPLOYMENT WITH AMSTER

Amster is a command-line interface built upon the ForgeRock Access Management REST interface. You can use Amster in DevOps processes, such as continuous integration, command-line installations, and scripted cloud deployments.

During this deployment I came upon few bugs with Amster when deploying an embedded environment instead of the AM stack. This issue delayed the automatization of the AM forcing me to do the configuration manually. This is why the Amster is placed at the top of the queue of future lines of development.

9.6 ARCHITECTURE

The architecture proposed in this PoC is a valid architecture, but it lacks resiliency, redundancy and an independency of the platform that could be solved by an orchestration system of containers. These are valid points to consider when you are designing your production environment but some of these changes would mean a total overhaul of this PoC and starting it from the beginning. This is intended to be done in a parallel line of development.

Regarding our current PoC architecture, there are a few tweaks that should be done; first, the bastion should be independent from the jump server instance and should be placed in the same VPC as the other instances. The same happens to the Squid proxy, this should be an independent instance in another VPC.

All of these changes would make our architecture more secure since we could just turn off the instances that are an open door to attacks or creating new ones, whilst still having a fully functional architecture. This line of development will be the first to be tackled in this PoC.

10 PROJECT SUMMARY

Devops has become the new SysAdmin in the world of IT, with the work done for this PoC, it proves that automation is indeed a powerful tool in IT and cloud management. The possibilities of use of this methodology seem, today, almost infinite in an overall overview but limited on the tools if presented standalone. This seems to force the DevOps engineer to use more than one tool for configuration, deployment, and management.

The way to approach the programming that requires the implementation of the proposed architecture on the AWS IaaS, has been presented with Ansible and some of the tools or options have been explained, which may allow interested readers to get started in this type of cloud automation in a quickly and easily manner.

The automation tool "Ansible" can be used in the field of teaching both in architectural design for autonomous deployment or administration tool, and in the field of system administration where it can be used to remotely update and install middleware. Activity structured and scheduled in successive stages, of increasing difficulty, is adaptable at various academic levels. The progressive advance in the study of the proposed situation has allowed to reach a more formal and synthesized knowledge, favouring the connection between concepts and procedures, as well as individual discovery.

Automation tools will always sound and look amazing once they are completed but as you may have seen, to automate a five-minute task it will take you three times that time to automate it. It may not seem much, but that its why automation is only practical to recursive tasks where you will only spend once a lot of time programming the code and then just one click for each iteration.

Regarding this PoC, there is still much work to be done and polished, as explained in the proposed lines of development, but this code could easily be used and reused for any production site for the deployment side. As a disclaimer, the code provided doesn’t provide the needed security measures nor redundancy and resiliency measures in its current release.

11GIT POC REPOSITORY

You will be able to see and download the full development of this PoC and the lines of development explained before. Juts download the repo from:

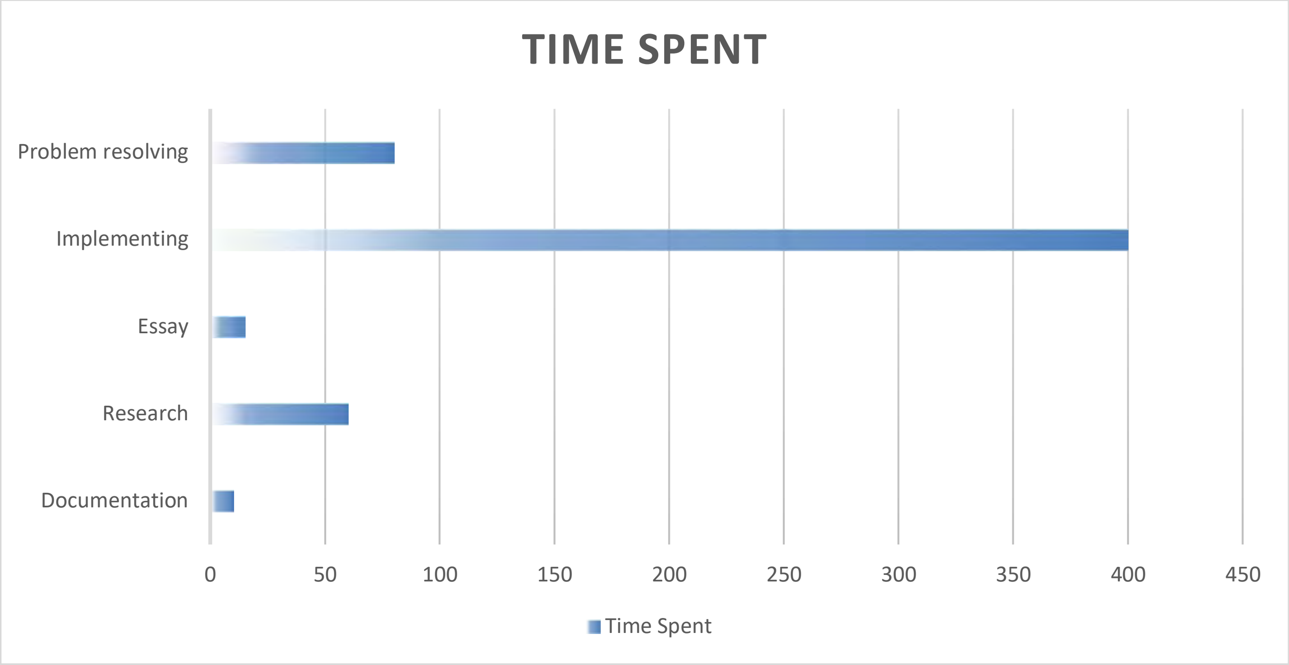
https://bitbucket.org/nacsom/ansible-aws/src/master/

12 TIME SPENT ON THE POC

Time spent is divided in 5 categories:

- Research – 60h
- Documentation – 10h
- Essay – 15h
- Implementation– 400h
- Problem solving – 80h

Total: **565h**



13 CITATIONS AND REFERENCES (IEE STANDARD)

[1] Amazon Web Services, Inc., [Online]. Available: https://d1.awsstatic.com/whitepapers/building-a-scalable-and-secure-multi-vpc-aws-network-infrastructure.pdf.

[2] Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Subnets.html.

[3] Amazon Web Services Documentation, [Online]. Available: https://docs.aws.amazon.com.

[4] Amazon Web Pages Documenation , [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Route_Tables.html.

[5] Amazon Web Services Documentation ENI, [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html.

[6] Amazon Web Service Documentation EIP, [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html.

[7] Amazon Web Servies ACLs, [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html.

[8] Amazon Web Services Documentation EC2 T3, [Online]. Available: https://aws.amazon.com/ec2/instance-types/t3/?nc1=h_ls.

[9] Amazon Web Services Documentation EC2 T3, [Online]. Available: https://aws.amazon.com/ec2/instance-types/t2/?nc1=h_ls.

[10] Elastic Guide for Filebeat, “elastic.co,” [Online]. Available: https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html.

[11] Wikipedia Redis, [Online]. Available: https://en.wikipedia.org/wiki/Redis.

[12] Elastic.co Logstash, [Online]. Available: https://www.elastic.co/logstash.

[13] ElasticSearch , [Online]. Available: https://www.elastic.co/es/what-is/elasticsearch.

[14] ElasticCo Kibana, [Online]. Available: https://www.elastic.co/what-is/kibana.

[15] Forgerock AM6.5, [Online]. Available: https://backstage.forgerock.com/docs/am/6.5/quick-start-guide/.

[16] ForgeRock , [Online]. Available: https://backstage.forgerock.com/docs/am/5.5/authorization-guide/.

[17] D. G.. [Online]. Available: https://www.hostinger.com/tutorials/what-is-apache.

[18] 123 Reg Support Center, [Online]. Available: https://www.123-reg.co.uk/support/servers/what-is-mysql-and-why-do-i-need-it/.

[19] M. Davis. [Online]. Available: https://www.futurehosting.com/blog/five-reasons-you-should-use-tomcat/.

[20] Wikipedia SquidProxy, [Online]. Available: https://en.wikipedia.org/wiki/Squid_(software).

[21] RedHat Ansible, [Online]. Available: https://www.ansible.com/overview/how-ansible-works.

[22] S. Lowe. [Online]. Available: https://blog.scottlowe.org/2015/12/11/using-ssh-multiplexing/.

[23] CyberArk Inc, [Online]. Available: https://lp.cyberark.com/rs/316-CZP-275/images/Core-Privileged-Access-Security-ds.pdf.

14 TABLE OF FIGURES

15 ANNEX

15.1LAUNCH

```
---
- hosts: localhost
  remote_user: ec2-user
  become: yes
  become_user: root
  connection: local

  vars:
    ansible_ssh_private_key_file: my_aws.pem


  pre_tasks:
    - name: Saving Global Variables
      set_fact:
        cacheable: yes
        key_name: my_aws
        region: eu-central-1
        vpc_management_id: vpc-0aa341c7735ba6ada
        subnet_management_id: subnet-0a86044e6d60248dd
        bastion_igw_id: igw-052d2bf8565871cd7
        bastion_vlan_block: 10.10.10.0/24

    - name: Create Security Group and upload SSH keys
      block:

      - name: Create an EC2 key
        ec2_key:
          aws_access_key: "{{ec2_access_key}}"
          aws_secret_key: "{{ec2_secret_key}}"
          name: "{{ key_name }}"
          region: "{{ region }}"
        register: ec2_key

      - name: Save private key
        copy:
          content: "{{ ec2_key.key.private_key }}"
          dest: "../{{ key_name }}.pem"
          mode: 0600
        when: ec2_key.changed

  roles:
    - role: bastion-network
      tags: ['never', 'bastion']
    - role: bastion
      tags: ['never', 'bastion']
    - role: networks
      delegate_to: bastion
      #tags: ['never', 'networks']
    - role: apacheServer
      delegate_to: bastion
      tags: ['never', 'apacheServer']
    - role: redlog
      delegate_to: bastion
      tags: ['never', 'redlog']
    - role: elasticSearch
      delegate_to: bastion
      #tags: ['never', 'elastic']
    - role: kibana
      delegate_to: bastion
      tags: ['never', 'kibana']
    - role: AM
      delegate_to: bastion
      tags: ['never', 'AM']
    - role: VPN
      delegate_to: bastion
      tags: ['never', 'VPN']


  post_tasks:
    - name: Bastion Public IP
      debug:
        msg: "{{ bastion_eip.public_ip }}.eu-central-1.compute.amazonaws.com"
      tags: ['never', 'bastion']
```

15.2 NETWORK

```
# AWS playbook that creates all the networks needed for the deployment of the ELS stack, AM  and server.
---

- name: Create VPCs, Subnets & Internet Gateway
  block:

  - name: Create VPC
    ec2_vpc_net:
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      name: TFM_VPC
      cidr_block: "{{ vlanBlock }}"
      region: "{{ region }}"
      tags:
        module: TFM_VPC
        this: works
    register: vpc

  - name: Saving variable VPC
    set_fact:
      cacheable: yes
      vpc: "{{ vpc }}"

  - name: Create subnet for AM server
    ec2_vpc_subnet:
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      state: present
      region: "{{ region }}"
      vpc_id: "{{ vpc.vpc.id }}"
      cidr: "{{ vlanAM }}"
      tags:
        Name: TFM-AM
    register: am_subnet

  - name: Saving variable AM subnet
    set_fact:
      cacheable: yes
      am_subnet: "{{ am_subnet }}"

  - name: Create subnet for Redis and Logstash server
    ec2_vpc_subnet:
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      state: present
    region: "{{ region }}"
    vpc_id: "{{ vpc.vpc.id }}"
    cidr: "{{ vlanLog }}"
    tags:
      Name: TFM-Log
  register: log_subnet

- name: Saving variable log subnet
  set_fact:
    cacheable: yes
    log_subnet: "{{ log_subnet }}"

- name: Create subnet for Elastic Search server
  ec2_vpc_subnet:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    state: present
    region: "{{ region }}"
    vpc_id: "{{ vpc.vpc.id }}"
    cidr: "{{ vlanElas }}"
    tags:
      Name: TFM-Elas
  register: elastic_subnet

- name: Saving variable Elastic subnet
  set_fact:
    cacheable: yes
    elastic_subnet: "{{ elastic_subnet }}"

- name: Create subnet for Kibana server
  ec2_vpc_subnet:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    state: present
    region: "{{ region }}"
    vpc_id: "{{ vpc.vpc.id }}"
    cidr: "{{ vlanKib }}"
    tags:
      Name: TFM-Kibana
  register: kibana_subnet

- name: Saving variable Kibana subnet
  set_fact:
    cacheable: yes
    kibana_subnet: "{{ kibana_subnet }}"

- name: Create subnet for users vlan
```

```
  ec2_vpc_subnet:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    state: present
    region: "{{ region }}"
    vpc_id: "{{ vpc.vpc.id }}"
    cidr: "{{ vlanUsr }}"
    tags:
      Name: TFM-Users
  register: usr_subnet

- name: Saving variable User subnet
  set_fact:
    cacheable: yes
    usr_subnet: "{{ usr_subnet }}"

- name: Create Internet Gateway
  ec2_vpc_igw:
    region: "{{ region }}"
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    vpc_id: "{{ vpc.vpc.id }}"
    state: present
    tags:
      Name: TFM
  register: igw

- name: Saving variable Internet Gateway
  set_fact:
    cacheable: yes
    igw: "{{ igw }}"

- name: Create local account VPC peering Connection
  ec2_vpc_peer:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    region: "{{ region }}"
    vpc_id: "{{ vpc_management_id }}"
    peer_vpc_id: "{{ vpc.vpc.id }}"
    state: present
    tags:
      Name: Peering connection for VPC Management to VPC TFM
  register: vpc_peer

- name: Accept local VPC peering request
  ec2_vpc_peer:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    region: "{{ region }}"
    peering_id: "{{ vpc_peer.peering_id }}"
    state: accept
  register: action_peer

- name: Add routing table from VPC TFM to VPC Management
  ec2_vpc_route_table:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    vpc_id: "{{ vpc.vpc.id }}"
    region: "{{ region }}"
    tags:
      Name: TFM
    subnets:
      - "{{ usr_subnet.subnet.id }}"
      - "{{ am_subnet.subnet.id }}"
      - "{{ log_subnet.subnet.id }}"
      - "{{ elastic_subnet.subnet.id }}"
      - "{{ kibana_subnet.subnet.id }}"
    routes:
      - dest: "{{ bastion_vlan_block }}"
        vpc_peering_connection_id: "{{ vpc_peer.peering_id }}"
      - dest: 0.0.0.0/0
        gateway_id: "{{ igw.gateway_id }}"
  register: public_route_table

- name: Add routing table from VPC Management to VPC TFM
  ec2_vpc_route_table:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    vpc_id: "{{ vpc_management_id }}"
    region: "{{ region }}"
    tags:
      Name: Management_Bastion
    subnets:
      - "{{ subnet_management_id }}"
    routes:
      - dest: "{{ vlanBlock }}"
        vpc_peering_connection_id: "{{ vpc_peer.peering_id }}"
      - dest: 0.0.0.0/0
        gateway_id: "{{ bastion_igw_id }}"
  register: public_route_table
```

15.3 BASTION

15.3.1 Network

```
# AWS playbook that creates the network mininal requiremtns to deploy a ssh bastion server
---
- name: Create VPCs, Subnet & Internet Gateway
  block:

  - name: Create VPC
    ec2_vpc_net:
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      name: Management_Bastion_VPC
      cidr_block: "{{ vlanBlock }}"
      region: "{{ region }}"
      tags:
        module: Management_Bastion_VPC
        this: works
    register: vpc

  - name: Saving variable VPC
    set_fact:
      cacheable: yes
      vpc_management_id: "{{ vpc }}"

  - name: Create subnet for Bastion server
    ec2_vpc_subnet:
      map_public: yes
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      state: present
      region: "{{ region }}"
      vpc_id: "{{ vpc.vpc.id }}"
      cidr: "{{ vlanManagement }}"
      tags:
        Name: Management_bastion
    register: management_bastion_subnet

  - name: Saving variable Bastion subnet
    set_fact:
      cacheable: yes
      am_subnet: "{{ management_bastion_subnet }}"

  - name: Create Internet Gateway
    ec2_vpc_igw:
      region: "{{ region }}"
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      vpc_id: "{{ vpc.vpc.id }}"
      state: present
      tags:
        Name: Bastion
    register: igw

  - name: Saving variable Internet Gateway
    set_fact:
      cacheable: yes
      igw: "{{ igw }}"

  - name: Set up public subnet route table
    ec2_vpc_route_table:
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      vpc_id: "{{ vpc.vpc.id }}"
      region: "{{ region }}"
      tags:
        Name: Management_Bastion
      subnets:
        - "{{ management_bastion_subnet.subnet.id }}"
      routes:
        - dest: 0.0.0.0/0
          gateway_id: "{{ igw.gateway_id }}"
    register: management_route_table

  - name: Updating Global Variables
    set_fact:
      cacheable: yes
      vpc_management_id: "{{ vpc.vpc.id }}"
      subnet_management_id: "{{ management_bastion_subnet.subnet.id }}"
      bastion_igw_id: "{{ igw.gateway_id }}"
      bastion_vlan_block: "{{ vlanBlock }}"
```

15.3.2 Main

```
# AWS playbook
---

- name: Create Security Group
  block:
    - name: Create Bastion security group
      ec2_group:
        vpc_id: "{{ vpc.vpc.id }}"
        name: "Bastion"
        description: "Sec group for Bastion-Management"
        region: "{{ region }}"
        aws_access_key: "{{ec2_access_key}}"
        aws_secret_key: "{{ec2_secret_key}}"
        rules:
          - proto: tcp
            from_port: 22
            to_port: 22
            cidr_ip: 0.0.0.0/0
          - proto: tcp
            from_port: 3128
            to_port: 3128
            cidr_ip: 10.10.0.0/16
        rules_egress:
          - proto: all
            cidr_ip: 0.0.0.0/0
        tags:
          Name: Management-Bastion
      register: result_sec_group_bastion

- name: Create Bastion Interface
  block:
  - name: Create Mngmnt network interface
    ec2_eni:
      subnet_id: "{{ management_bastion_subnet.subnet.id }}"
      state: present
      region: "{{ region }}"
      description: Management network interface
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      private_ip_address: 10.10.10.10
      security_groups: "{{ result_sec_group_bastion.group_id }}"
    register: management_network

  - name: Create instance
    ec2_instance:
      name: "Bastion Server"
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      key_name: "{{ key_name }}"
      region: "{{ region }}"
      vpc_subnet_id: "{{ management_bastion_subnet.subnet.id }}"
      instance_type: t2.micro
      network:
        interfaces:
          - id: "{{ management_network.interface.id }}"
        assign_public_ip: true
      image_id: "{{ image }}"
    wait: true
  register: bastion_instance

- name: Saving Bastion Instance info
  set_fact:
    cacheable: yes
    bastion_instance: "{{ bastion_instance }}"

- name: Debug result
  debug:
    msg: "{{ bastion_instance }}"
    verbosity: 2

- name: Create Elastic IP
  ec2_eip:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    region: "{{ region }}"
    in_vpc: yes
    state: present
    device_id: "{{ management_network.interface.id  }}"
  register: bastion_eip

- name: Debug result
  debug:
    msg: "{{ bastion_eip }}"
    verbosity: 2

- name: Saving Elastic Bstion ip
  set_fact:
    cacheable: yes
    bastion_eip: "{{ bastion_eip }}"

- name: Add Bastion to hosts
  add_host:
    hostname: "{{bastion_eip.public_ip }}.eu-central-1.compute.amazonaws.com"
    ansible_host: bastion
    ansible_port: 22

- name: Replace the bastion SSH.config entry with the new one
  lineinfile:
    path: ../ssh.cfg
    regexp: 'Hostname'
    line: "  Hostname {{ bastion_eip.public_ip }}"
    owner: root
    group: root
    mode: '645'
  - name: Replace the bastion SSH.config entry with the new one
    lineinfile:
      path: /etc/ssh/ssh_config
      regexp: 'Hostname'
      line: "  Hostname {{ bastion_eip.public_ip }}"
      owner: root
      group: root
      mode: '645'

  - name: Replace the bastion in hosts file with the new one
    lineinfile:
      path: ../hosts
      regexp: 'bastion'
      line: "bastion ansible_host={{ bastion_eip.public_ip }}"
      owner: root
      group: root
      mode: '644'

- name: Upload my_aws.pem to the instance
  command: scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i my_aws.pem my_aws.pem
ec2-user@{{ bastion_eip.public_ip }}:/tmp/my_aws.pem

- name: sleep for 60 seconds and continue with play
  wait_for:
    timeout: 60

- name: Change permissions to my_aws.pem
  file:
    path: /tmp/my_aws.pem
    mode: 400
    owner: root
  delegate_to: "{{ bastion_eip.public_ip }}"

- name: install Python packages
  include: install.yaml
  delegate_to: "{{ bastion_eip.public_ip }}"
```

15.4 APACHE

15.4.1 Main

```
# AWS playbook
---
- name: Create Security Group
  block:
    - name: Create Apache security group
      ec2_group:
        vpc_id: "{{ vpc.vpc.id }}"
        name: "Apache"
        description: "Sec group for TFM"
        region: "{{ region }}"
        aws_access_key: "{{ec2_access_key}}"
        aws_secret_key: "{{ec2_secret_key}}"
        rules:
          - proto: tcp
            from_port: 22
            to_port: 22
            cidr_ip: 10.10.10.0/27
          - proto: tcp
            from_port: 80
            to_port: 80
            cidr_ip: 0.0.0.0/0
          - proto: tcp
            from_port: 8081
            to_port: 8081
            cidr_ip: 0.0.0.0/0
          - proto: tcp
            from_port: 443
            to_port: 443
            cidr_ip: 0.0.0.0/0
          - proto: tcp
            from_port: 3306
            to_port: 3306
            cidr_ip: 10.10.11.0/24
        rules_egress:
          - proto: all
            cidr_ip: 0.0.0.0/0
        tags:
          Name: TFM-Apache
      register: result_sec_group_apache

- name: Create Apache Interfaces
  block:
  - name: Create User network interface
    ec2_eni:
```

```
    subnet_id: "{{ usr_subnet.subnet.id }}"
    state: present
    region: "{{ region }}"
    description: User network interface
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    private_ip_address: "{{apache_privateip_usr}}"
    security_groups: "{{ result_sec_group_apache.group_id }}"
  register: usr_network

- name: Create Redis/Logstash network interface
  ec2_eni:
    subnet_id: "{{ log_subnet.subnet.id }}"
    state: present
    region: "{{ region }}"
    description: log network interface
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    private_ip_address: "{{apache_privateip_log}}"
    security_groups: "{{ result_sec_group_apache.group_id }}"
  register: log_network

- name: Create AM network interface
  ec2_eni:
    subnet_id: "{{ am_subnet.subnet.id }}"
    state: present
    region: "{{ region }}"
    description: log network interface
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    private_ip_address: "{{apache_privateip_am}}"
    security_groups: "{{ result_sec_group_apache.group_id }}"
  register: am_network

- name: Create instance
  ec2_instance:
    name: "Apache Server"
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    key_name: "{{ key_name }}"
    region: "{{ region }}"
    vpc_subnet_id: "{{ usr_subnet.subnet.id }}"
    instance_type: t2.small
    network:
      interfaces:
        - id: "{{ usr_network.interface.id }}"
        - id: "{{ log_network.interface.id }}"
        - id: "{{ am_network.interface.id }}"
      image_id: "{{ image }}"
      wait: true
    register: apache_instance

  - name: Create Elastic IP
    ec2_eip:
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      region: "{{ region }}"
      in_vpc: yes
      state: present
      device_id: "{{ usr_network.interface.id  }}"
    register: apache_eip


- name: Test for line
  shell: grep "apache" /etc/hosts | wc -l
  register: test_grep

- name: add apache host to hostfile
  lineinfile:
    dest: /etc/hosts
    line: "{{apache_privateip_log}}  apache"
    owner: root
    group: root
    mode: '644'
  when: test_grep.stdout == 0

- name: Replace the apache in hosts file with the new one
  lineinfile:
    path: /etc/hosts
    regexp: 'apache'
    line: "{{apache_privateip_log}}  apache"
    owner: root
    group: root
    mode: '644'
  when: test_grep.stdout != 0

- name: install Apache packages
  include: install.yaml
```

15.4.2 Installation

```
---
- name: Prepare apache files in bastion
  block:

  - name: Check if Wordpress exists in /tmp
    stat: path=/tmp/latest.zip
    register: folder

  - name: Download Wordpress
    shell: cd /tmp/ && curl -L -O https://wordpress.org/latest.zip
    when: not folder.stat.exists

  - name: Send Wordpress to Apache server
    command: "sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem /tmp/latest.zip ec2-user@{{apache_privateip_usr}}:/tmp/wordpress-5.4.zip"

  - name: Send OpenamAgent to Apache server
    command: "sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem /tmp/webagents.zip ec2-user@{{apache_privateip_usr}}:/tmp/webagents.zip"

- name: Install apache and MySQL
  block:

  - name: Check if Mysqld repo exists in /tmp
    stat: path=/tmp/mysql80-community-release-el7-3.noarch.rpm
    register: folder

  - name: Download MYSQLD server repo
    command:  wget -P /tmp/ https://dev.mysql.com/get/mysql57-community-release-el7-9.noarch.rpm
    when: not folder.stat.exists

  - name: Enable amazon-linux-extras packages
    shell: "amazon-linux-extras enable php7.3"
    become: yes

  - name: Install Mysql repo
    yum:
      state: present
      name: /tmp/mysql57-community-release-el7-9.noarch.rpm
    when: not folder.stat.exists

  - name: Install HTTPd MYSQLd PHP Python Plugins
    yum:
      state: present
      name:
          - httpd.x86_64
```

```
        - mysql-community-common.x86_64
        - mysql-community-devel.x86_64
        - mysql-community-libs.x86_64
        - mysql-community-libs-compat.x86_64
        - mysql-connector-python.x86_64
        - mysql-community-server.x86_64
        - php-common.x86_64
        - php-json.x86_64
        - php-mysqlnd.x86_64
        - php-pdo.x86_64
        - php.x86_64
        - mysql-connector-python3.x86_64
        - mysql-connector-python.x86_64
        - MySQL-python.x86_64

      skip_broken: yes
    when: not folder.stat.exists

  - name: Ensure mysql is running and starts on boot
    service:
      name: mysqld
      state: started
      enabled: yes
    become: yes

  - name: get root password
    shell: "grep 'A temporary password is generated for root@localhost' /var/log/mysqld.log | awk -F ' ' '{print
$(NF)}'"
    register: root_password


  - name: Debug result
    debug:
      msg: "{{ root_password }}"
      verbosity: 2

  - name: Debug result
    debug:
      msg: "{{ root_password.stdout }}"
      verbosity: 2

  - name: Check if my conf file exists.
    stat:
      path:  /etc/my.cnf.bck
    register: file_details

  - name: update expired root user password
```

```
    command: mysql --user=root --password={{ root_password.stdout }} --connect-expired-password --
execute="ALTER USER 'root'@'localhost' IDENTIFIED BY '{{ root_new_passwd }}';"
    when:  file_details.stat.exists == False

  - name: Copy my conf
    command: mv  /etc/my.cnf  /etc/my.cnf.bck
    when: file_details.stat.exists == False

  - name: Touch the same file, but add/remove some permissions
    file:
      path: /etc/my.cnf
      state: touch
      mode: '0644'
    when: file_details.stat.exists == False

  - name: Insert my conf configuration
    blockinfile:
      path: /etc/my.cnf
      block: |
        # For advice on how to change settings please see
        # http://dev.mysql.com/doc/refman/8.0/en/server-configuration-defaults.html

        [mysqld]
        bind-address=127.0.0.1
        #
        # Remove leading # and set to the amount of RAM for the most important data
        # cache in MySQL. Start at 70% of total RAM for dedicated server, else 10%.
        # innodb_buffer_pool_size = 128M
        #
        # Remove the leading "# " to disable binary logging
        # Binary logging captures changes between backups and is enabled by
        # default. It's default setting is log_bin=binlog
        # disable_log_bin
        #
        # Remove leading # to set options mainly useful for reporting servers.
        # The server defaults are faster for transactions and fast SELECTs.
        # Adjust sizes as needed, experiment to find the optimal values.
        # join_buffer_size = 128M
        # sort_buffer_size = 2M
        # read_rnd_buffer_size = 2M
        #
        # Remove leading # to revert to previous value for default_authentication_plugin,
        # this will increase compatibility with older clients. For background, see:
        #                                        https://dev.mysql.com/doc/refman/8.0/en/server-system-
variables.html#sysvar_default_authentication_plugin
        # default-authentication-plugin=mysql_native_password
        default_authentication_plugin=mysql_native_password
        datadir=/var/lib/mysql
        socket=/var/lib/mysql/mysql.sock

        log-error=/var/log/mysqld.log
        pid-file=/var/run/mysqld/mysqld.pid

    when: file_details.stat.exists == False

  - name: Restart service Mysqld, in all cases
    service:
      name: mysqld
      state: restarted
      enabled: yes


  - name: Create a new database with name 'wordpress'
    mysql_db:
      login_user: root
      login_password: "{{root_new_passwd}}"
      name: "{{DB_name}}"
      state: present

  - name: Create database user with password and all database privileges and 'WITH GRANT OPTION'
    mysql_user:
      login_user: root
      login_password: "{{root_new_passwd}}"
      name: "{{wordpress_user}}"
      password: "{{wordpress_pass}}"
      priv: '*.*:ALL,GRANT'
      state: present

  delegate_to: "{{apache_privateip_usr}}"

- name: Configure apache server and PHP
  block:

  - name: Check if Wordpress.yml file exists.
    stat:
      path:  /etc/httpd/conf.d/welcome.conf.bck
    register: file_details

  - name: Copy welcome.conf.bck
    command: mv  /etc/httpd/conf.d/welcome.conf  /etc/httpd/conf.d/welcome.conf.bck
    when: file_details.stat.exists == False

  - name: Touch the same file, but add/remove some permissions
    file:
```

```
   path: /etc/httpd/conf.d/wordpress.conf
   state: touch
   mode: u+rw,g-wx,o-rwx

 - name: Insert Input logs for HttpConf
  blockinfile:
   path: /etc/httpd/conf.d/wordpress.conf
   block: |
    <VirtualHost *:80>
     ServerAdmin webmaster@localhost
     DocumentRoot /var/www/html/wordpress
     ServerName "{{ServerName}}"
     ServerAlias "{{ServerAlias}}"

     <Directory /var/www/html/wordpress/>
       Options Indexes FollowSymLinks
       AllowOverride All
       Require all granted
     </Directory>

     ErrorLog /var/log/httpd/error.log
     CustomLog /var/log/httpd/access.log combined

     <IfModule mod_dir.c>
       DirectoryIndex index.php index.pl index.cgi index.html index.xhtml index.htm
     </IfModule>

    </VirtualHost>

  when: file_details.stat.exists == False

 - name: Start service httpd, if not started
  service:
   name: httpd
   state: restarted
   enabled: yes
  become: yes

 delegate_to: "{{apache_privateip_usr}}"

- name: Install Wordpress
 block:

 - name: Check for apache exists in /tmp
  stat: path=/var/www/html/wordpress
  register: folder
```

```
- name: Unzip files
  shell: cd /tmp/ && sudo unzip -q wordpress-5.4.zip -d /var/www/html/
  when: not folder.stat.exists

- name: Change directory permisions in wordpress
  command: find /var/www/html/wordpress/ -type d -exec chmod 755 {} \;

- name: Change file permisions in wordpress
  command: find /var/www/html/wordpress/ -type f -exec chmod 644 {} \;

- name: Check if wp-config.php file exists.
  stat:
    path: /var/www/html/wordpress/wp-config.php
  register: file_details

- name: Create wp-config.php
  file:
    path: /var/www/html/wordpress/wp-config.php
    state: touch
    mode: '644'
  when: file_details.stat.exists == False

- name: Insert wp-config.php
  blockinfile:
    path: /var/www/html/wordpress/wp-config.php
    block: |
         <?php
         /**
         * The base configuration for WordPress
         *
         * The wp-config.php creation script uses this file during the
         * installation. You don't have to use the web site, you can
         * copy this file to "wp-config.php" and fill in the values.
         *
         * This file contains the following configurations:
         *
         * * MySQL settings
         * * Secret keys
         * * Database table prefix
         * * ABSPATH
         *
         * @link https://wordpress.org/support/article/editing-wp-config-php/
         *
         * @package WordPress
         */

         // ** MySQL settings - You can get this info from your web host ** //
```

```
        /** The name of the database for WordPress */
        define( 'DB_NAME', '{{DB_name}}' );

        /** MySQL database username */
        define( 'DB_USER', '{{wordpress_user}}' );

        /** MySQL database password */
        define( 'DB_PASSWORD', '{{wordpress_pass}}' );

        /** MySQL hostname */
        define( 'DB_HOST', 'localhost' );

        /** Database Charset to use in creating database tables. */
        define( 'DB_CHARSET', 'utf8mb4' );

        /** The Database Collate type. Don't change this if in doubt. */
        define( 'DB_COLLATE', '' );

        define( 'WP_HOME', 'https://{{domain_name}}' );
        define( 'WP_SITEURL', 'https://{{domain_name}}' );
        define('FORCE_SSL_ADMIN', true);

        /**#@+
        * Authentication Unique Keys and Salts.
        *
        * Change these to different unique phrases!
        * You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org
secret-key service}
        * You can change these at any point in time to invalidate all existing cookies. This will force all users to
have to log in again.
        *
        * @since 2.6.0
        */
        define(         'AUTH_KEY',                                                          '@!aMAQC`kC>J~~05nxeD
x{]6_W.x>A@N9T+V&V5$Xw%nm3e+he:Xxpr).oY%RIM' );
        define(                                                                               'SECURE_AUTH_KEY',
'Kyz&t85=79Bf!Q8K=2t|Sq%!s&CfhU$Z<ud!VmkZ+b:LU/,:Q7`JfV/vO2^H@|VN' );
        define( 'LOGGED_IN_KEY',    ',?;}Jr*yF:Q0ub!$zFHNL`!q7>v/h*YbHdIL6z?dF}RUKbcPwc3l+/s057!}{JhN' );
        define( 'NONCE_KEY',       '+LItbX1Qq:DI]KDGx^,uO&No[zLWNfzAm#+myTdH==p/o?j,g-FNOpV.P`3BdaD2'
);
        define( 'AUTH_SALT',       '[E%h#a6L]T?%/BGeKE:7p)pSMl]E2.^Xx~Ub9.rG8S{n}bybXggN}m~P&&(%Jq;o' );
        define(                                                                              'SECURE_AUTH_SALT',
'AD)0Lg5%AsByVsz|dGr0EB/13N:[<G{GL;s)N;T*vL$.wzW9LUdv~;]8pJQQw>kj' );
        define( 'LOGGED_IN_SALT',    'encc2<aek.,u$]V+!Q~$G<F0$_s(LYZ-_^w4CWjjLJ4+Fn[qN1?-*7lc@f5sWFj;'
);
        define( 'NONCE_SALT',       ';ol/^Mm@8!9[_eKJH4k{75J}1_Y!(6.wg({EPzi{jQmb ILk6o)ei/<UV!HK|-Yl' );
     /**#@-*/

     /**
     * WordPress Database Table prefix.
     *
     * You can have multiple installations in one database if you give each
     * a unique prefix. Only numbers, letters, and underscores please!
     */
     $table_prefix = 'wp_';

     /**
     * For developers: WordPress debugging mode.
     *
     * Change this to true to enable the display of notices during development.
     * It is strongly recommended that plugin and theme developers use WP_DEBUG
     * in their development environments.
     *
     * For information on other constants that can be used for debugging,
     * visit the documentation.
     *
     * @link https://wordpress.org/support/article/debugging-in-wordpress/
     */
     define( 'WP_DEBUG', false );

     /* That's all, stop editing! Happy publishing. */

     /** Absolute path to the WordPress directory. */
     if ( ! defined( 'ABSPATH' ) ) {
      define( 'ABSPATH', __DIR__ . '/' );
     }

     /** Sets up WordPress vars and included files. */
     require_once ABSPATH . 'wp-settings.php';

 when: file_details.stat.exists == False

- name: Check if .htaccess file exists.
 stat:
  path: /var/www/html/wordpress/.htaccess
 register: file_details

- name: Create .htaccess
 file:
  path: /var/www/html/wordpress/.htaccess
  state: touch
  mode: '644'
 when: file_details.stat.exists == False
```

```
  - name: Insert .htaccess
    blockinfile:
      path: /var/www/html/wordpress/.htaccess
      block: |
          <IfModule mod_rewrite.c>
          RewriteEngine On
          RewriteBase /
          RewriteRule ^index\.php$ - [L]
          RewriteCond %{REQUEST_FILENAME} !-f
          RewriteCond %{REQUEST_FILENAME} !-d
          RewriteRule . /index.php [L]
          </IfModule>
    when: file_details.stat.exists == False

  - name: Start service httpd, if not started
    service:
      name: httpd
      state: restarted
      enabled: yes


  delegate_to: "{{apache_privateip_usr}}"

- name: Prepare Filebeats in apache
  block:

  - name: Check if filebeat exists in /tmp
    stat: path=/tmp/filebeat-7.6.1-x86_64.rpm
    register: folder

  - name: Download Beats from ELKweb to bastion
    shell: cd /tmp/ && curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.6.1-x86_64.rpm
    when: not folder.stat.exists

  - name: Send Beats zip to Apache server
    command: "sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem /tmp/filebeat-7.6.1-x86_64.rpm ec2-user@{{apache_privateip_usr}}:/tmp/filebeat-7.6.1-x86_64.rpm"

- name: Configure File beat
  block:

  - name: Check if filebeat-7.6.1-x86_64.rpm is installed
    command: rpm -q filebeat-7.6.1-1.x86_64
    ignore_errors: true
    register: rpm_check
```

```
 - name: Install beats
   command: rpm -vi /tmp/filebeat-7.6.1-x86_64.rpm
   when: rpm_check.stdout.find('is not installed') != -1

 - name: Check if Filebeat.yml file exists.
   stat:
     path: /etc/filebeat/filebeat.yml.bck
   register: file_details

 - name: Copy Filebeat.yml.bck
   command: mv /etc/filebeat/filebeat.yml /etc/filebeat/filebeat.yml.bck
   when: file_details.stat.exists == False

 - name: Create Filebeat.yml
   file:
     path: /etc/filebeat/filebeat.yml
     state: touch
     mode: '644'
   when: file_details.stat.exists == False

# - name: Check if Input logs is configured
 #  command: grep -w "/etc/httpd/logs/access_log" /etc/filebeat/filebeat.yml
 #  check_mode: no
 #  ignore_errors: yes
 #  changed_when: no
 #  register: conf_check


 - name: Insert Input logs for Filebeat
   blockinfile:
     path: /etc/filebeat/filebeat.yml
     block: |

       #========================== Filebeat modules ==============================

       filebeat.config.modules:
         # Glob pattern for configuration loading
         path: ${path.config}/modules.d/*.yml

         # Set to true to enable config reloading
         reload.enabled: true

         # Period on which files under path should be checked for changes
         #reload.period: 10s

       filebeat.modules:
       - module: apache
```

```
    #------------------------------- Redis output ----------------------------------
    output.redis:
     hosts: ["{{logstash_privateip_log}}:6379"]
     key: "filebeat"
     db: 0
     timeout: 5
     data_type: "list"


    #------------------------------- Logging ---------------------------------------
    logging.level: info
    logging.to_files: true
    logging.files:
     path: /var/log/filebeat
     name: filebeat
     keepfiles: 7
     permissions: 0644

  when: file_details.stat.exists == False

 - name: Start service filebeat, if not started
  service:
   name: filebeat
   state: started
   enabled: yes
  become: yes

 delegate_to: "{{apache_privateip_usr}}"

- name: Configure File beat
 block:

 - name: add openam FQDN to hostfile
  lineinfile:
   dest: /etc/hosts
   line: "{{am_privateip_am}}  openam.example.com"
   owner: root
   group: root
   mode: '644'
  when: test_grep.stdout == 0

 delegate_to: "{{apache_privateip_usr}}"

- name: Install certbot in apache
 block:
  - name: Ansible check directory.
    stat:
     path: /tmp/dl.fedoraproject.org
    register: my_folder

  - name: Get nEpel release tom
    command: wget -P /tmp/ -r --no-parent -A 'epel-release-*.rpm'
http://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/
    when: my_folder.stat.exists == false

  - name: install repository
    command: sudo rpm -Uvh /tmp/dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/epel-release-*.rpm
    when: my_folder.stat.exists == false

  - name: enable epel
    command: sudo yum-config-manager --enable epel*
    when: my_folder.stat.exists == false

  - name: Install certbot
    command: sudo yum install -y certbot python2-certbot-apache

  delegate_to: "{{apache_privateip_usr}}"

- name: Configure SSL certificate for apache
  block:

  - name: Configure SSL apache
    command: certbot --apache -n -d {{domain_name}} --agree-tos --email {{acme_email}} --redirect

  delegate_to: "{{apache_privateip_usr}}"
```

15.5 LOGSTASH

15.5.1 Main

```
# AWS playbook
---

- name: Create Security Group
  block:
    - name: Create Redsis Logstash security group
      ec2_group:
        vpc_id: "{{ vpc.vpc.id }}"
        name: "Redsis Logstash"
        description: "Sec group for TFM"
        region: "{{ region }}"
        aws_access_key: "{{ec2_access_key}}"
        aws_secret_key: "{{ec2_secret_key}}"
        rules:
          - proto: tcp
            from_port: 22
            to_port: 22
            cidr_ip: 0.0.0.0/0
          - proto: tcp #logstash
            from_port: 5044
            to_port: 5044
            cidr_ip: "{{vlanBlock}}"
          - proto: tcp #redis
            from_port: 6379
            to_port: 6379
            cidr_ip: "{{vlanBlock}}"
          - proto: tcp
            from_port: 443
            to_port: 443
            cidr_ip: 0.0.0.0/0
        rules_egress:
          - proto: all
            cidr_ip: 0.0.0.0/0
        tags:
          Name: TFM-Log
      register: result_sec_group_log

- name: Create Elastic Interfaces
  block:

  - name: Create Elastic network interface
    ec2_eni:
      subnet_id: "{{ elastic_subnet.subnet.id }}"
      state: present
      region: "{{ region }}"
      description: elastic network interface
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      private_ip_address: "{{logstash_privateip_elas}}"
      security_groups: "{{ result_sec_group_log.group_id }}"
    register: elastic_network

  - name: Create Redis/Logstash network interface
    ec2_eni:
      subnet_id: "{{ log_subnet.subnet.id }}"
      state: present
      region: "{{ region }}"
      description: log network interface
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      private_ip_address: "{{logstash_privateip_log}}"
      security_groups: "{{ result_sec_group_log.group_id }}"
    register: log_network

  - name: Create instance
    ec2_instance:
      name: "Redis Logstash Server"
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      key_name: "{{ key_name }}"
      region: "{{ region }}"
      vpc_subnet_id: "{{ log_subnet.subnet.id }}"
      instance_type: t2.micro
      network:
        interfaces:
          - id: "{{ log_network.interface.id }}"
          - id: "{{elastic_network.interface.id}}"
      image_id: "{{ image }}"
      wait: true
    register: log_instance

- name: Debug result
  debug:
    msg: "{{ log_instance }}"
    verbosity: 2

- name: Test for line
  shell: grep "redis" /etc/hosts | wc -l
  register: test_grep

- name: add redis host to hostfile
```

```
  lineinfile:
    dest: /etc/hosts
    line: "{{logstash_privateip_log}}  redis"
    owner: root
    group: root
    mode: '644'
  when: test_grep.stdout == 0

- name: Replace the redis in hosts file with the new one
  lineinfile:
    path: /etc/hosts
    regexp: 'redis'
    line: "{{logstash_privateip_log}}  redis"
    owner: root
    group: root
    mode: '644'
  when: test_grep.stdout != 0

- name: Test for line
  shell: grep "logstash" /etc/hosts | wc -l
  register: test_grep

- name: add logstash host to hostfile
  lineinfile:
    dest: /etc/hosts
    line: "{{logstash_privateip_log}}  logstash"
    owner: root
    group: root
    mode: '644'
  when: test_grep.stdout == 0

- name: Replace the logstash in hosts file with the new one
  lineinfile:
    path: /etc/hosts
    regexp: 'logstash'
    line: "{{logstash_privateip_log}}  logstash"
    owner: root
    group: root
    mode: '644'
  when: test_grep.stdout != 0

- name: install Redis & logstash packages
  include: install.yaml
```

15.5.2 Installation

```
---
- name: Prepare logstash files & redis files
  block:

  - name: Check for java exists in /tmp
    stat: path=/tmp/java.zip
    register: folder

  - name: Upload Java software to the instance
    shell: cd /tmp/ && scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i my_aws.pem
oracle180.rpm ec2-user@{{ logstash_privateip_log }}:/tmp/oracle180.rpm

  - name: Check for redis exists in /tmp
    stat: path=/tmp/redis.zip
    register: folder

  - name: Download Redis from AWS to bastion
    command: sudo yum install redis-3.2.12-2.el6.x86_64 --downloadonly --downloaddir=/tmp/redis
    when: not folder.stat.exists

  - name: Zip redis installation files
    shell: cd /tmp/ && sudo zip -r redis.zip redis/
    when: not folder.stat.exists

  - name: Send Redis to Redis Logstash server
    command: sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem
/tmp/redis.zip ec2-user@{{logstash_privateip_log}}:/tmp/redis.zip

  - name: Check for logstash exists in /tmp
    stat: path=/tmp/logstash-7.6.1.rpm
    register: folder

  - name: Download Logstash from AWS to bastion
    shell: cd /tmp/ && curl -L -O https://artifacts.elastic.co/downloads/logstash/logstash-7.6.1.rpm
    when: not folder.stat.exists

  - name: Send Logstash to Redis Logstash server
    command: sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem
/tmp/logstash-7.6.1.rpm ec2-user@{{logstash_privateip_log}}:/tmp/logstash-7.6.1.rpm

#- name: Configure certificates
#  block:
#  - name: Check for stack cert exists
#    stat: path=/etc/ssl/private/stack.pem
#    register: folder
#  - name: Generate an OpenSSL private key with the default values (4096 bits, RSA)
#    openssl_privatekey:
#      path: /etc/ssl/private/stack.pem
#      cipher: aes256
#      passphrase: stack
#      type: DSA
#    when: not folder.stat.exists

#  - name: Generate an OpenSSL private key with the default values (4096 bits, RSA)
#    openssl_privatekey:
#      path: /etc/ssl/private/logstash.pem
#      cipher: aes256
#      type: DSA
#
#  - name: Generate an OpenSSL Certificate Signing Request
#    openssl_csr:
#      path: /etc/ssl/csr/logstash.csr
#      privatekey_path: /etc/ssl/private/logstash.pem
#      common_name: tfmStack
#      #key_usage: Signature
#
#  - name: Generate certificate
#    openssl_certificate:
#      path: /etc/ssl/crt/logstash.crt
#      privatekey_path: /etc/ssl/private/stack.pem
#      csr_path: /etc/ssl/csr/logstash.csr
#      provider: selfsigned
#
#  - name: Send Logstash.pem
#     command: sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem
/etc/ssl/private/logstash.pem ec2-user@{{logstash_privateip_log}}:/etc/ssl/private/logstash.pem
#
#  - name: Send Stack.pem
#     command: sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem
/etc/ssl/private/stack.pem ec2-user@{{logstash_privateip_log}}:/etc/ssl/private/stack.pem
#
#  - name: Send Logstash.crt
#     command: sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem
/etc/ssl/crt/logstash.crt ec2-user@{{logstash_privateip_log}}:/etc/ssl/crt/logstash.crt

- name: Install Java 1.8
  block:

  - name: Install java 1.8 orace
    command: rpm -ivh /tmp/oracle180.rpm
    ignore_errors: true
  - name: Change Java version from 1.7 to 1.8
    alternatives:
      name: java
      link: /usr/bin/java
      path: /usr/java/jdk1.8.0_251-amd64/jre/bin/java

  delegate_to: "{{logstash_privateip_log}}"

- name: Install redis
  block:

  - name: Check for redis exists in /tmp
    stat: path=/tmp/redis
    register: folder

  - name: Unzip files
    shell: cd /tmp/ && sudo unzip -q redis.zip
    when: not folder.stat.exists

  - name: Check if jemalloc-3.3.1-1.8.amzn1.x86_64.rpm is installed
    command: rpm -q jemalloc-3.3.1-1.8.amzn1.x86_64
    ignore_errors: true
    register: rpm_check

  - name: Install redis dependency
    command: rpm -U /tmp/redis/jemalloc-3.3.1-1.8.amzn1.x86_64.rpm
    when: rpm_check.stdout.find('is not installed') != -1

  - name: Check if redis-3.2.12-2.el6.x86_64 is installed
    command: rpm -q redis-3.2.12-2.el6.x86_64
    ignore_errors: true
    register: rpm_check

  - name: Install redis
    command: rpm -U /tmp/redis/redis-3.2.12-2.el6.x86_64.rpm
    when: rpm_check.stdout.find('is not installed') != -1

  - name: Replace the binded host in redis conf
    lineinfile:
      path: /etc/redis.conf
      regexp: 'bind 127.0.0.1'
      line: "bind {{logstash_privateip_log}}"
      owner: root
      group: root
      mode: '644'
    when: rpm_check.stdout.find('is not installed') != -1
  - name: Start service Redis, if not started
    service:
      name: redis
      state: started

  delegate_to: "{{logstash_privateip_log}}"


- name: Install logstash server
  block:

  - name: Check if logstash-7.6.1.rpm is installed
    command: rpm -q logstash-7.6.1-1.noarch
    ignore_errors: true
    register: rpm_check

  - name: Install amazon plugin
    command: rpm -U /tmp/logstash-7.6.1.rpm
    when: rpm_check.stdout.find('is not installed') != -1

  - name: Replace the Xms y Xmg jvm.options logstash
    lineinfile:
      path: /etc/logstash/jvm.options
      regexp: '-Xms1g'
      line: "-Xms512m"
      owner: root
      group: root
      mode: '644'
    when: rpm_check.stdout.find('is not installed') != -1

  - name: Replace the Xms y Xmg jvm.options logstash
    lineinfile:
      path: /etc/logstash/jvm.options
      regexp: '-Xmx1g'
      line: "-Xmx512m"
      owner: root
      group: root
      mode: '644'
    when: rpm_check.stdout.find('is not installed') != -1

  - name: Register Logstash in service
    command:  /usr/share/logstash/bin/system-install /etc/logstash/startup.options sysv
    when: rpm_check.stdout.find('is not installed') != -1

  - name: Configure logstash server
    block:
```

```
- name: Check if my-pipeline.conf file exists.
  stat:
    path: /etc/logstash/conf.d/my-pipeline.conf
  register: file_details

- name: Create my-pipeline.conf
  file:
    path: /etc/logstash/conf.d/my-pipeline.conf
    state: touch
    mode: '644'
  when: file_details.stat.exists == False

- name: Insert Input logs for logstash
  blockinfile:
    path: /etc/logstash/conf.d/my-pipeline.conf
    block: |
      input {
        redis {
          host => "{{logstash_privateip_log}}"
          port => "6379"
          data_type => "list"
          key => "filebeat"
        }
      }
      filter {
          grok {
              match => { "message" => "%{COMBINEDAPACHELOG}"}
          }
          geoip {
              source => "clientip"
          }
          date {
            match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
          }
      }
      output {
        elasticsearch {
          id => "logstash"
          index => "filebeat-"
          hosts => ["{{elastic_privateip_elas}}:9200"]
        }
        stdout { codec => rubydebug }
      }


  when: file_details.stat.exists == False
```

```
- name: Install systemctl filebeat service
  command: sudo /usr/share/logstash/bin/system-install /etc/logstash/startup.options systemd
  ignore_errors: true


- name: Start service Logstash, if not started
  service:
    name: logstash
    state: started

delegate_to: "{{logstash_privateip_log}}"
```

15.6 ElasticSearch

15.6.1 Main

```
# AWS playbook
---

- name: Create Security Group
 block:
  - name: Create Elastic security group
   ec2_group:
    vpc_id: "{{ vpc.vpc.id }}"
    name: "Elastic"
    description: "Sec group for TFM"
    region: "{{ region }}"
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    rules:
     - proto: tcp
      from_port: 22
      to_port: 22
      cidr_ip: 10.10.10.0/27
     - proto: tcp
      from_port: "{{elasticport1}}"
      to_port: "{{elasticport1}}"
      cidr_ip: 10.10.11.0/24
     - proto: tcp
      from_port: "{{elasticport2}}"
      to_port: "{{elasticport2}}"
      cidr_ip: 10.10.11.0/24
    rules_egress:
     - proto: all
      cidr_ip: 0.0.0.0/0
    tags:
     Name: TFM-Elastic
   register: result_sec_group_elastic

- name: Create Elastic Interfaces
 block:
 - name: Create Elastic network interface
  ec2_eni:
   subnet_id: "{{ elastic_subnet.subnet.id }}"
   state: present
   region: "{{ region }}"
   description: Elastic network interface
   aws_access_key: "{{ec2_access_key}}"
   aws_secret_key: "{{ec2_secret_key}}"
   private_ip_address: "{{elastic_privateip_elas}}"
```

```
      security_groups: "{{ result_sec_group_elastic.group_id }}"
    register: elastic_network

  - name: Create Kibana network interface
    ec2_eni:
      subnet_id: "{{ kibana_subnet.subnet.id }}"
      state: present
      region: "{{ region }}"
      description: Kibana network interface
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      private_ip_address: "{{elastic_privateip_kib}}"
      security_groups: "{{ result_sec_group_elastic.group_id }}"
    register: kibana_network

  - name: Create instance
    ec2_instance:
      name: "Elastic Server"
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      key_name: "{{ key_name }}"
      region: "{{ region }}"
      vpc_subnet_id: "{{ elastic_subnet.subnet.id }}"
      instance_type: t2.micro
      network:
        interfaces:
          - id: "{{ elastic_network.interface.id }}"
          - id: "{{ kibana_network.interface.id }}"
      image_id: "{{ image }}"
      wait: true
    register: elastic_instance

  - name: Debug result
    debug:
      msg: "{{ elastic_instance }}"
      verbosity: 2

- name: Test for line
  shell: grep "elasticSearch" /etc/hosts | wc -l
  register: test_grep

- name: add elasticSearch host to hostfile
  lineinfile:
    dest: /etc/hosts
    line: "{{elastic_privateip_elas}}  elasticSearch"
    owner: root
    group: root
    mode: '644'
  when: test_grep.stdout == 0

- name: Replace the elasticSearch in hosts file with the new one
  lineinfile:
    path: /etc/hosts
    regexp: 'elasticSearch'
    line: "{{elastic_privateip_elas}}  elasticSearch"
    owner: root
    group: root
    mode: '644'
  when: test_grep.stdout != 0

- name: install Elastic packages
  include: install.yaml
```

15.6.2 Installation

```
---
- name: Prepare Elastic files
  block:

  - name: Check for java exists in /tmp
    stat: path=/tmp/java.zip
    register: folder

  - name: Upload Java software to the instance
    shell: cd /tmp/ && scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i my_aws.pem
oracle180.rpm ec2-user@{{ elastic_privateip_elas }}:/tmp/oracle180.rpm

  - name: Ansible check directory.
    stat:
     path: /tmp/elastic
    register: my_folder

  - name: Ansible Create directory if not exists
    file:
     path: /tmp/elastic
     state: directory
     mode: 0755
     group: root
     owner: root
    when: my_folder.stat.exists == false
  - name: Check if elasticSearch exists in /tmp
    stat: path=/tmp/elastic/elasticsearch-7.6.1-x86_64.rpm
    register: folder

  - name: Download ElasticSeaarch from ELKweb to bastion
    shell: cd /tmp/elastic/ && curl -L -O https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.6.1-x86_64.rpm
    when: not folder.stat.exists

  - name: Check if elasticSearch SHA512 exists in /tmp
    stat: path=/tmp/elastic/elasticsearch-7.6.1-x86_64.rpm.sha512
    register: folder

  - name: Download ElasticSeaarch from ELKweb to bastion
    shell: cd /tmp/elastic/ && curl -L -O https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.6.1-x86_64.rpm.sha512
    when: not folder.stat.exists

  - name: Zip elastic installation files
    shell: cd /tmp/ && sudo zip -r elastic.zip elastic/
    when: not folder.stat.exists

  - name: Send elastic zip to ElasticSearch server
    command: "sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem /tmp/elastic.zip ec2-user@{{elastic_privateip_elas}}:/tmp/elastic.zip"

#- name: Configure certificates
#  block:

#  - name: Check for stack cert exists
#    stat: path=/etc/ssl/private/stack.pem
#    register: folder
#
#  - name: Generate an OpenSSL private key with the default values (4096 bits, RSA)
#    openssl_privatekey:
#      path: /etc/ssl/private/stack.pem
#      cipher: aes256
#      passphrase: stack
#      type: DSA
#    when: not folder.stat.exists
#
#  - name: Generate an OpenSSL private key with the default values (4096 bits, RSA)
#    openssl_privatekey:
#      path: /etc/ssl/private/elastic.pem
#      cipher: aes256
#      type: DSA
#    when: not folder.stat.exists
```

```
#
#  - name: Generate an OpenSSL Certificate Signing Request
#    openssl_csr:
#      path: /etc/ssl/csr/elastic.csr
#      privatekey_path: /etc/ssl/private/elastic.pem
#      common_name: tfmStack
#      #key_usage: Signature
#    when: not folder.stat.exists
#
#  - name: Generate certificate
#    openssl_certificate:
#      path: /etc/ssl/crt/elastic.crt
#      privatekey_path: /etc/ssl/private/stack.pem
#      csr_path: /etc/ssl/csr/elastic.csr
#      provider: selfsigned
#    when: not folder.stat.exists
#
#  - name: Send Logstash.pem
#     command: sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem
/etc/ssl/private/elastic.pem ec2-user@{{elastic_privateip_elas}}:/etc/ssl/private/elastic.pem
#
#  - name: Send Stack.pem
#     command: sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem
/etc/ssl/private/stack.pem ec2-user@{{elastic_privateip_elas}}:/etc/ssl/private/stack.pem
#
#  - name: Send Logstash.crt
#     command: sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem
/etc/ssl/crt/elastic.crt ec2-user@{{elastic_privateip_elas}}:/etc/ssl/crt/elastic.crt

- name: Install Java 1.8
  block:

  - name: Install java 1.8 orace
    command: rpm -ivh /tmp/oracle180.rpm
    ignore_errors: true

  - name: Change Java version from 1.7 to 1.8
    alternatives:
      name: java
      link: /usr/bin/java
      path: /usr/java/jdk1.8.0_251-amd64/jre/bin/java

  delegate_to: "{{elastic_privateip_elas}}"

- name: Install ElasticSearch server
  block:
```

```
- name: Check for elastic exists in /tmp
  stat: path=/tmp/elastic
  register: folder

- name: Unzip files
  shell: cd /tmp/ && sudo unzip -q elastic.zip
  when: not folder.stat.exists

- name: Check if elasticsearch-7.6.1-x86_64.rpmis installed
  command: rpm -q elasticsearch-7.6.1-1.x86_64
  ignore_errors: true
  register: rpm_check

#- name: Install elastic Search
#  command: shasum -a 512 -c /tmp/elastic/elasticsearch-7.6.1-x86_64.rpm.sha512
#  when: rpm_check.stdout.find('is not installed') != -1

- name: Install elastic Search
  command: rpm -U /tmp/elastic/elasticsearch-7.6.1-x86_64.rpm
  when: rpm_check.stdout.find('is not installed') != -1

- name: Replace the Xms y Xmg jvm.options elasticseach
  lineinfile:
    path: /etc/elasticsearch/jvm.options
    regexp: '-Xms1g'
    line: "-Xms512m"
    owner: root
    group: root
    mode: '644'
  when: rpm_check.stdout.find('is not installed') != -1

- name: Replace the Xms y Xmg jvm.options elasticseach
  lineinfile:
    path: /etc/elasticsearch/jvm.options
    regexp: '-Xmx1g'
    line: "-Xmx512m"
    owner: root
    group: root
    mode: '644'
  when: rpm_check.stdout.find('is not installed') != -1

- name: Check if elasticsearch.yml file exists.
  stat:
    path: /etc/elasticsearch/elasticsearch.yml.bck
  register: file_details

- name: Copy elasticsearch.yml.bck
```

```
  command: mv /etc/elasticsearch/elasticsearch.yml /etc/elasticsearch/elasticsearch.yml.bck
  when: file_details.stat.exists == False

- name: Create elastic.yml
  file:
    path: /etc/elasticsearch/elasticsearch.yml
    state: touch
    mode: '644'
  when: file_details.stat.exists == False

- name: Configure ElasticSearch
  blockinfile:
    path: /etc/elasticsearch/elasticsearch.yml
    block: |
      # ---------------------------------- Cluster -----------------------------------
      #
      # Use a descriptive name for your cluster:
      #
      cluster.name: TFM
      #
      # ------------------------------------ Node ------------------------------------
      #
      # Use a descriptive name for the node:
      #
      node.name: elastic
      #
      # Add custom attributes to the node:
      #
      #node.attr.rack: r1
      #
      # ----------------------------------- Paths ------------------------------------
      #
      # Path to directory where to store the data (separate multiple locations by comma):
      #
      path.data: /var/lib/elasticsearch
      #
      # Path to log files:
      #
      path.logs: /var/log/elasticsearch
      #
      # ----------------------------------- Memory -----------------------------------
      #
      # Lock the memory on startup:
      #
      #bootstrap.memory_lock: true
      #
      # Make sure that the heap size is set to about half the memory available
      # on the system and that the owner of the process is allowed to use this
      # limit.
      #
      # Elasticsearch performs poorly when the system is swapping the memory.
      #
      # ---------------------------------- Network -----------------------------------
      #
      # Set the bind address to a specific IP (IPv4 or IPv6):
      #
      network.host: ["{{elastic_privateip_elas}}", "{{elastic_privateip_kib}}"]
      #
      # Set a custom port for HTTP:
      #
      http.port: {{elasticport1}}
      #
      # For more information, consult the network module documentation.
      #
      # --------------------------------- Discovery ----------------------------------
      #
      # Pass an initial list of hosts to perform discovery when this node is started:
      # The default list of hosts is ["127.0.0.1", "[::1]"]
      #
      #discovery.seed_hosts: ["{{logstash_privateip_elas}}"]
      discovery.type: single-node
      #
      # Bootstrap the cluster using an initial set of master-eligible nodes:
      #
      #cluster.initial_master_nodes: ["node-1", "node-2"]
      #
      # For more information, consult the discovery and cluster formation module documentation.
      #
      # ---------------------------------- Gateway -----------------------------------
      #
      # Block initial recovery after a full cluster restart until N nodes are started:
      #
      #gateway.recover_after_nodes: 3
      #
      # For more information, consult the gateway module documentation.
      #
      # ---------------------------------- Various -----------------------------------
      #
      # Require explicit names when deleting indices:
      #
      #action.destructive_requires_name: true
      script.painless.regex.enabled: true
      #xpack.security.enabled: true
      #xpack.security.http.ssl.enabled: true
```

```
        #xpack.security.transport.ssl.enabled: true
        #xpack.security.http.ssl.key: /etc/ssl/private/elastic.pem
        #xpack.security.http.ssl.certificate: /etc/ssl/cert/elastic.crt
        #xpack.security.http.ssl.certificate_authorities: /etc/ssl/private/stack.pem
        #xpack.security.transport.ssl.key: /etc/ssl/private/elastic.pem
        #xpack.security.transport.ssl.certificate: /etc/ssl/cert/elastic.crt
        #xpack.security.transport.ssl.certificate_authorities: /etc/ssl/private/stack.pem

  when: file_details.stat.exists == False

- name: Start service elastic search, if not started
  service:
    name: elasticsearch
    state: started

delegate_to: "{{elastic_privateip_elas}}"
```

15.7 Kibana

15.7.1 Main

```
# AWS playbook
---

- name: Create Security Group
 block:
  - name: Create Kibana security group
   ec2_group:
    vpc_id: "{{ vpc.vpc.id }}"
    name: "Kibana"
    description: "Sec group for TFM"
    region: "{{ region }}"
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    rules:
     - proto: tcp
      from_port: 22
      to_port: 22
      cidr_ip: 10.10.10.0/27
     - proto: tcp
      from_port: "{{kibanaport1}}"
      to_port: "{{kibanaport1}}"
      cidr_ip: 0.0.0.0/0
     - proto: tcp
      from_port: "{{kibanaport2}}"
      to_port: "{{kibanaport2}}"
      cidr_ip: 0.0.0.0/0
    rules_egress:
     - proto: all
      cidr_ip: 0.0.0.0/0
    tags:
     Name: TFM-Kibana
   register: result_sec_group_kibana

- name: Create Kibana Interfaces
 block:
 - name: Create Kibana public network interface
  ec2_eni:
   subnet_id: "{{ usr_subnet.subnet.id }}"
   state: present
   region: "{{ region }}"
   description: Kibana network interface
   aws_access_key: "{{ec2_access_key}}"
   aws_secret_key: "{{ec2_secret_key}}"
   private_ip_address: "{{kibana_privateip_usr}}"
```

```
      security_groups: "{{ result_sec_group_kibana.group_id }}"
    register: user_network

  - name: Create Kibana network interface
    ec2_eni:
      subnet_id: "{{ kibana_subnet.subnet.id }}"
      state: present
      region: "{{ region }}"
      description: Kibana network interface
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      private_ip_address: "{{kibana_privateip_kib}}"
      security_groups: "{{ result_sec_group_kibana.group_id }}"
    register: kibana_network

  - name: Create instance
    ec2_instance:
      name: "Kibana Server"
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      key_name: "{{ key_name }}"
      region: "{{ region }}"
      vpc_subnet_id: "{{ usr_subnet.subnet.id }}"
      instance_type: t2.micro
      network:
        interfaces:
          - id: "{{ user_network.interface.id }}"
          - id: "{{ kibana_network.interface.id }}"
      image_id: "{{ image }}"
      wait: true
    register: kibana_instance


  - name: Create Elastic IP
    ec2_eip:
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      region: "{{ region }}"
      in_vpc: yes
      state: present
      device_id: "{{ user_network.interface.id  }}"
    register: kibana_eip


- name: Test for line
  shell: grep "kibana" /etc/hosts | wc -l
  register: test_grep
```

```
- name: add Kibana host to hostfile
  lineinfile:
    dest: /etc/hosts
    line: "{{kibana_privateip_kib}}  kibana"
    owner: root
    group: root
    mode: '644'
  when: test_grep.stdout == 0

- name: Replace the kibana in hosts file with the new one
  lineinfile:
    path: /etc/hosts
    regexp: 'kibana'
    line: "{{kibana_privateip_kib}}  kibana"
    owner: root
    group: root
    mode: '644'
  when: test_grep.stdout != 0

- name: install Kibana packages
  include: install.yaml
```

15.7.2 Installation

```
---
- name: Prepare Elastic files
  block:

  - name: Check for java exists in /tmp
    stat: path=/tmp/java.zip
    register: folder

  - name: Upload Java software to the instance
    shell: cd /tmp/ && scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i my_aws.pem oracle180.rpm ec2-user@{{ kibana_privateip_kib }}:/tmp/oracle180.rpm

  - name: Check if kibana exists in /tmp
    stat: path=/tmp/kibana-7.6.2-x86_64.rpm
    register: folder

  - name: Download kibana from ELKweb to bastion
    shell: cd /tmp/ && curl -L -O https://artifacts.elastic.co/downloads/kibana/kibana-7.6.2-x86_64.rpm
    when: not folder.stat.exists

  - name: Send kiabana  to Kiabana server
```

```
    command: "sudo scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem /tmp/kibana-7.6.2-x86_64.rpm ec2-user@{{kibana_privateip_kib}}:/tmp/kibana-7.6.2-x86_64.rpm"

- name: Install Java 1.8
  block:

  - name: Install java 1.8 orace
    command: rpm -ivh /tmp/oracle180.rpm
    ignore_errors: true

  - name: Change Java version from 1.7 to 1.8
    alternatives:
      name: java
      link: /usr/bin/java
      path: /usr/java/jdk1.8.0_251-amd64/jre/bin/java

  delegate_to: "{{kibana_privateip_kib}}"

- name: Install Kibana server
  block:

  - name: Check if kibana-7.6.2-x86_64.rpm is installed
    command: rpm -q kibana-7.6.2-1.x86_64
    ignore_errors: true
    register: rpm_check

  - name: Install kibana
    command: rpm -U /tmp/kibana-7.6.2-x86_64.rpm
    when: rpm_check.stdout.find('is not installed') != -1

  - name: Check if Kibana.yml file exists.
    stat:
      path: /etc/kibana/kibana.yml.bck
    register: file_details

  - name: Copy kibana.yml.bck
    command: mv /etc/kibana/kibana.yml /etc/kibana/kibana.yml.bck
    when: file_details.stat.exists == False

  - name: Create kibana.yml
    file:
      path: /etc/kibana/kibana.yml
      state: touch
      mode: '644'
    when: file_details.stat.exists == False

  - name: Configure Kibana
```

```
    blockinfile:
      path: /etc/kibana/kibana.yml
      block: |
        # Kibana is served by a back end server. This setting specifies the port to use.
        server.port: {{kibanaport1}}

        # Specifies the address to which the Kibana server will bind. IP addresses and host names are both valid
values.
        # The default is 'localhost', which usually means remote machines will not be able to connect.
        # To allow connections from remote users, set this parameter to a non-loopback address.
        server.host: "{{kibana_privateip_usr}}"

        # Enables you to specify a path to mount Kibana at if you are running behind a proxy.
        # Use the `server.rewriteBasePath` setting to tell Kibana if it should remove the basePath
        # from requests it receives, and to prevent a deprecation warning at startup.
        # This setting cannot end in a slash.
        #server.basePath: ""

        # Specifies whether Kibana should rewrite requests that are prefixed with
        # `server.basePath` or require that they are rewritten by your reverse proxy.
        # This setting was effectively always `false` before Kibana 6.3 and will
        # default to `true` starting in Kibana 7.0.
        #server.rewriteBasePath: false

        # The maximum payload size in bytes for incoming server requests.
        #server.maxPayloadBytes: 1048576

        # The Kibana server's name.  This is used for display purposes.
        #server.name: "your-hostname"

        # The URLs of the Elasticsearch instances to use for all your queries.
        #elasticsearch.hosts: ["http://localhost:9200"]
        elasticsearch.hosts: ["http://{{elastic_privateip_kib}}:{{elasticport1}}"]

        # When this setting's value is true Kibana uses the hostname specified in the server.host
        # setting. When the value of this setting is false, Kibana uses the hostname of the host
        # that connects to this Kibana instance.
        #elasticsearch.preserveHost: true

        # Kibana uses an index in Elasticsearch to store saved searches, visualizations and
        # dashboards. Kibana creates a new index if the index doesn't already exist.
        #kibana.index: ".kibana"

        # The default application to load.
        #kibana.defaultAppId: "home"

        # If your Elasticsearch is protected with basic authentication, these settings provide
# the username and password that the Kibana server uses to perform maintenance on the Kibana
# index at startup. Your Kibana users still need to authenticate with Elasticsearch, which
# is proxied through the Kibana server.
#elasticsearch.username: "kibana"
#elasticsearch.password: "pass"

# Enables SSL and paths to the PEM-format SSL certificate and SSL key files, respectively.
# These settings enable SSL for outgoing requests from the Kibana server to the browser.
#server.ssl.enabled: false
#server.ssl.certificate: /path/to/your/server.crt
#server.ssl.key: /path/to/your/server.key

# Optional settings that provide the paths to the PEM-format SSL certificate and key files.
# These files are used to verify the identity of Kibana to Elasticsearch and are required when
# xpack.security.http.ssl.client_authentication in Elasticsearch is set to required.
#elasticsearch.ssl.certificate: /path/to/your/client.crt
#elasticsearch.ssl.key: /path/to/your/client.key

# Optional setting that enables you to specify a path to the PEM file for the certificate
# authority for your Elasticsearch instance.
#elasticsearch.ssl.certificateAuthorities: [ "/path/to/your/CA.pem" ]

# To disregard the validity of SSL certificates, change this setting's value to 'none'.
#elasticsearch.ssl.verificationMode: full

# Time in milliseconds to wait for Elasticsearch to respond to pings. Defaults to the value of
# the elasticsearch.requestTimeout setting.
#elasticsearch.pingTimeout: 1500

# Time in milliseconds to wait for responses from the back end or Elasticsearch. This value
# must be a positive integer.
#elasticsearch.requestTimeout: 30000

# List of Kibana client-side headers to send to Elasticsearch. To send *no* client-side
# headers, set this value to [] (an empty list).
#elasticsearch.requestHeadersWhitelist: [ authorization ]

# Header names and values that are sent to Elasticsearch. Any custom headers cannot be overwritten
# by client-side headers, regardless of the elasticsearch.requestHeadersWhitelist configuration.
#elasticsearch.customHeaders: {}

# Time in milliseconds for Elasticsearch to wait for responses from shards. Set to 0 to disable.
#elasticsearch.shardTimeout: 30000

# Time in milliseconds to wait for Elasticsearch at Kibana startup before retrying.
#elasticsearch.startupTimeout: 5000
```

```
      # Logs queries sent to Elasticsearch. Requires logging.verbose set to true.
      #elasticsearch.logQueries: false

      # Specifies the path where Kibana creates the process ID file.
      #pid.file: /var/run/kibana.pid

      # Enables you specify a file where Kibana stores log output.
      #logging.dest: stdout

      # Set the value of this setting to true to suppress all logging output.
      #logging.silent: false

      # Set the value of this setting to true to suppress all logging output other than error messages.
      #logging.quiet: false

      # Set the value of this setting to true to log all events, including system usage information
      # and all requests.
      #logging.verbose: false

      # Set the interval in milliseconds to sample system and process performance
      # metrics. Minimum is 100ms. Defaults to 5000.
      #ops.interval: 5000

      # Specifies locale to be used for all localizable strings, dates and number formats.
      # Supported languages are the following: English - en , by default , Chinese - zh-CN .
      #i18n.locale: "en"

  when: file_details.stat.exists == False

- name: Start service kibana, if not started
  service:
    name: kibana
    state: started

delegate_to: "{{kibana_privateip_kib}}"
```

15.8AM

15.8.1 Main

```
# AWS playbook
---

- name: Create Security Group
  block:
    - name: Create AM security group
      ec2_group:
        vpc_id: "{{ vpc.vpc.id }}"
        name: "AM"
        description: "Sec group for TFM"
        region: "{{ region }}"
        aws_access_key: "{{ec2_access_key}}"
        aws_secret_key: "{{ec2_secret_key}}"
        rules:
          - proto: tcp
            from_port: 22
            to_port: 22
            cidr_ip: 10.10.10.0/16
          - proto: tcp
            from_port: "{{amport1}}"
            to_port: "{{amport1}}"
            cidr_ip: 10.10.11.0/16
          - proto: tcp
            from_port: "{{amport3}}"
            to_port: "{{amport3}}"
            cidr_ip: 10.10.11.0/16
          - proto: tcp
            from_port: "{{amport2}}"
            to_port: "{{amport2}}"
            cidr_ip: 10.10.11.0/16
          - proto: tcp
            from_port: "{{amport4}}"
            to_port: "{{amport4}}"
            cidr_ip: 10.10.11.0/16
        rules_egress:
          - proto: all
            cidr_ip: 0.0.0.0/0
        tags:
          Name: TFM-AM
      register: result_sec_group_am

- name: Create Elastic Interfaces
  block:
  - name: Create Elastic network interface
  ec2_eni:
    subnet_id: "{{ elastic_subnet.subnet.id }}"
    state: present
    region: "{{ region }}"
    description: Elastic network interface
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    private_ip_address: "{{am_privateip_elas}}"
    security_groups: "{{ result_sec_group_am.group_id }}"
  register: elastic_network

- name: Create AM network interface
  ec2_eni:
    subnet_id: "{{ am_subnet.subnet.id }}"
    state: present
    region: "{{ region }}"
    description: AM network interface
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    private_ip_address: "{{am_privateip_am}}"
    security_groups: "{{ result_sec_group_am.group_id }}"
  register: am_network

- name: Create Elastic IP
  ec2_eip:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    region: "{{ region }}"
    in_vpc: yes
    state: present
    device_id: "{{ am_network.interface.id  }}"
  register: apache_eip

- name: Create instance
  ec2_instance:
    name: "AM Server"
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    key_name: "{{ key_name }}"
    region: "{{ region }}"
    vpc_subnet_id: "{{ elastic_subnet.subnet.id }}"
    instance_type: t3a.small
    network:
      interfaces:
        - id: "{{ am_network.interface.id }}"
        - id: "{{ elastic_network.interface.id }}"
    image_id: "{{ image }}"
```

```
   wait: true
  register: elastic_instance


 - name: Debug result
  debug:
   msg: "{{ elastic_instance }}"
   verbosity: 2

- name: Test for line
 shell: grep "am" /etc/hosts | wc -l
 register: test_grep

- name: add am host to hostfile
 lineinfile:
  dest: /etc/hosts
  line: "{{am_privateip_elas}}  am"
  owner: root
  group: root
  mode: '644'
 when: test_grep.stdout == 0

- name: Replace the am in hosts file with the new one
 lineinfile:
  path: /etc/hosts
  regexp: 'am'
  line: "{{am_privateip_elas}}  am"
  owner: root
  group: root
  mode: '644'
 when: test_grep.stdout != 0

- name: install AM packages
 include: install.yaml
```

15.8.2 Installation

```
---
- name: Prepare files in bastion
 block:
 - name: Upload AM software to the instance
  shell: cd /tmp/ && scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i my_aws.pem amfiles.zip ec2-user@{{ am_privateip_elas }}:/tmp/amfiles.zip

 - name: Check for tomcat exists in /tmp
  stat: path=/tmp/tomcat.zip
```

```
    register: folder

  - name: Download Tomcat 8.5
    get_url:
      url: https://apache.brunneis.com/tomcat/tomcat-8/v8.5.57/bin/apache-tomcat-8.5.57.tar.gz
      dest: /tmp/apache-tomcat-8.5.57.tar.gz

  - name: Zip java installation files
    shell: cd /tmp/ && sudo zip -r tomcat.zip tomcat/
    when: not folder.stat.exists

  - name: Upload Tomcat software to the instance
    shell: cd /tmp/ && scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i my_aws.pem
apache-tomcat-8.5.57.tar.gz ec2-user@{{ am_privateip_elas }}:/tmp/apache-tomcat-8.5.57.tar.gz

  - name: Upload Java software to the instance
    shell: cd /tmp/ && scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i my_aws.pem
oracle180.rpm ec2-user@{{ am_privateip_elas }}:/tmp/oracle180.rpm

#- name: Check for key exists in /tmp
#  stat: path=/tmp/my_aws.pem
#  register: key
#  delegate_to: "{{am_privateip_elas}}"

#- name: Upload my_aws.pem to the instance
#    command: scp -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i /tmp/my_aws.pem
/tmp/my_aws.pem ec2-user@{{ am_privateip_elas }}:/tmp/my_aws.pem
 # when: not key.stat.exists

#- name: Change ownership of my_aws.pem
#  file:
#    path: /tmp/my_aws.pem
#    state: file
#    owner: root
#    group: root
#    mode: '0400'

#  delegate_to: "{{am_privateip_elas}}"

- name: Configure Yum proxy
  block:

  - name: install the latest version of Tomact
    yum:
      name:
        - tomcat
        - tomcat-webapps
    - tomcat-docs-webapp
    - tomcat-admin-webapps
   state: latest

 - name: Add proxy line in yum conf
  lineinfile:
   path: /etc/yum.conf
   insertafter: '\[main\]'
   line: proxy=http://10.10.10.10:3128
   create: yes


 delegate_to: "{{am_privateip_elas}}"

- name: Install Java 1.8
 block:

 - name: Install java 1.8 orace
  command: rpm -ivh /tmp/oracle180.rpm
  ignore_errors: true

 - name: Change Java version from 1.7 to 1.8
  alternatives:
   name: java
   link: /usr/bin/java
   path: /usr/java/jdk1.8.0_251-amd64/jre/bin/java

 delegate_to: "{{am_privateip_elas}}"

- name: Install Tomcat server
 block:

 - name: Check for tomcat exists in /tmp
  stat: path=/tmp/tomcat
  register: folder

 - name: Unzip files
  shell: cd /tmp/ && sudo unzip -q tomcat.zip
  when: not folder.stat.exists

 - name: add tomcat group
  group:
   name: tomcat

 - name: add tomcat user
  user:
   name: tomcat
```

```
      group: tomcat
      createhome: yes

  - name: create /opt/tomcat directory
    file:
      path: /opt/tomcat
      state: directory
      mode: '0755'
      owner: tomcat
      group: tomcat

  - name: Unzip Tomcat installation files
    command: sudo tar xvf /tmp/apache-tomcat-8*tar.gz -C /opt/tomcat --strip-components=1

  - name: Recursively change ownership of a directory
    file:
      path: /opt/tomcat
      state: directory
      recurse: yes
      owner: tomcat

  - name: Give permisions to tomcat Conf
    command: chmod -R g+r /opt/tomcat/conf

  - name: Give permisions to tomcat Conf
    command: chmod  g+x /opt/tomcat/conf

  - name: Give permisions to tomcat Conf
    command: chown -R tomcat /opt/tomcat/webapps/ /opt/tomcat/work/ /opt/tomcat/temp/
/opt/tomcat/logs/

  - name: Check whether tomcat-user configuration
    command: grep -Fq '<user username="tomcat" password="tomcat" roles="tomcat, manager-gui"/>'
/usr/share/tomcat/conf/tomcat-users.xml
    register: checkmyconf
    check_mode: no
    ignore_errors: yes
    changed_when: no

  - name: Add admin in tomcat-user manager
    lineinfile:
      path: /usr/share/tomcat/conf/tomcat-users.xml
      insertafter: '<tomcat-users>'
      line: <role rolename="manager-gui"/> <user username="tomcat" password="tomcat" roles="tomcat,
manager-gui"/>
      create: yes
    when: checkmyconf.rc != 0
```

```
- name: Check if tomcat.service file exists.
  stat:
    path: /etc/systemd/system/tomcat.service
  register: file_details

- name: Copy tomcat.service.bck
  command: mv /etc/systemd/system/tomcat.service /etc/systemd/system/tomcat.service.bck
  when: file_details.stat.exists == False

- name: Create tomcat.service
  file:
    path: /etc/systemd/system/tomcat.service
    state: touch
    owner: root
    group: tomcat
    mode: '644'
  when: file_details.stat.exists == False

- name: Add configuration to tomcat.service
  blockinfile:
    path: /etc/systemd/system/tomcat.service
    block: |
      # Systemd unit file for tomcat
      [Unit]
      Description=Apache Tomcat Web Application Container
      After=syslog.target network.target

      [Service]
      Type=forking

      Environment=JAVA_HOME=/usr/java/jdk1.8.0_251-amd64/jre
      Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
      Environment=CATALINA_HOME=/opt/tomcat
      Environment=CATALINA_BASE=/opt/tomcat
      Environment='CATALINA_OPTS= -Xmx2g -XX:MetaspaceSize=256m -XX:MaxMetaspaceSize=256m -server -
Dorg.apache.tomcat.util.http.ServerCookie.ALWAYS_ADD_EXPIRES=true'
      Environment='JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'

      ExecStart=/opt/tomcat/bin/startup.sh
      ExecStop=/opt/tomcat/bin/shutdown.sh

      User=tomcat
      Group=tomcat
      UMask=0007
      RestartSec=10
      Restart=always
```

```
        [Install]
        WantedBy=multi-user.target
    when: file_details.stat.exists == False

  - name: Configure java enviroment variables
    command: echo "JAVA_HOME=/usr/java/jdk1.8.0_251-amd64" >> /etc/profile && echo "export
JAVA_HOME" >> /etc/profile && JAVA_HOME=/usr/java/jdk1.8.0_251-amd64 && export JAVA_HOME

  - name: Check whether ~/.bashrc configuration
    command: grep -Fq "export JAVA_HOME=" ~/.bashrc
    register: checkmyconf
    check_mode: no
    ignore_errors: yes
    changed_when: no

  - name: Add java home environment
    blockinfile:
      path: ~/.bashrc
      block: |
        export JAVA_HOME=$(dirname $(dirname $(readlink $(readlink $(which javac)))))
        export PATH=$PATH:$JAVA_HOME/bin
        export CLASSPATH=.:$JAVA_HOME/jre/lib:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar
    when: checkmyconf.rc != 0
   # when: file_details.stat.exists == False

  - name: Enable environment variable
    shell: source ~/.bashrc
    when: checkmyconf.rc != 0
   # when: file_details.stat.exists == False

  - name: Restart Tomcat
    service:
      name: tomcat
      state: restarted
      enabled: yes
    become: yes


  delegate_to: "{{am_privateip_elas}}"

- name: Deploy AM war
  block:

  - name: Check for amfiles exists in /tmp
    stat: path=/tmp/amfiles
    register: folder
  - name: Unzip files
    shell: cd /tmp/ && sudo unzip -q amfiles.zip -d /tmp/amfiles
    when: not folder.stat.exists

  - name: Change file ownership, group and permissions
    file:
      path:  /tmp/amfiles/am6.5.war
      owner: tomcat
      group: tomcat
      mode: '2755'
    when: not folder.stat.exists

  - name: Check for AM war in webapps
    stat: path=/usr/share/tomcat/webapps/{{AM_War_Name}}.war
    register: folder

  - name: Move AM to tomcat webpps
    command: mv /tmp/amfiles/am6.5.war /usr/share/tomcat/webapps/{{AM_War_Name}}.war
    when: not folder.stat.exists

  - name: Check for amster files in usr share
    stat: path=/usr/share/amster
    register: folder

  - name: Unzip Amster
    command: sudo unzip -q /tmp/amfiles/Amster -d /usr/share/amster
    when: not folder.stat.exists

  - name: Ensure Tomcat is running and starts on boot
    service:
      name: tomcat
      state: started
      enabled: yes
    become: yes

  delegate_to: "{{am_privateip_elas}}"

#- name: Install AM6.5 with Amster
#  block:

  #- name: Check if Amster instalation file exists.
  #  stat:
  #    path: /usr/share/amster/installEmbedded.amster
  #  register: file_details
#
  #- name: Create Instalation file for Embedded AM
```

```
 #  file:
 #    path: /usr/share/amster/installEmbedded.amster
 #    state: touch
 #    mode: '644'
 #  when: file_details.stat.exists == False
#
 #- name: Create installation files
 #  blockinfile:
 #    path: /usr/share/amster/installEmbedded.amster
 #    block: |
 #      "install-openam --serverUrl http://{{amFQDN}}:8080/{{AM_War_Name}} \
 #      --adminPwd {{adminPasswd}} \
 #      --cookieDomain {{amCookieDomain}} \
 #      --cfgDir {{cfgDir}} \
 #      --acceptLicense"
 #  when: file_details.stat.exists == False

 #- name: Create installation files
 #  copy:
 #    dest:  /usr/share/amster/installEmbedded.amster
 #    force: no
 #    owner: root
 #    mode: 0644
 #    content: |
 #      install-openam --serverUrl http://{{amFQDN}}:8080/{{AM_War_Name}} \
 #      --adminPwd {{adminPasswd}} \
 #      --cookieDomain {{amCookieDomain}} \
 #      --cfgDir {{cfgDir}} \
 #      --acceptLicense \
 #      :exit



 #- name: Execute Amster with instalation file
 #  shell: /usr/share/amster/amster /usr/share/amster/installEmbedded.amster
   #when: file_details.stat.exists == False

# delegate_to: "{{am_privateip_elas}}"
```