Assignment documentation

The following aims to document each step of the assignment based on the requirements. Ansible playbooks were used to achieve most of the configuration on the nodes.

“Use Terraform to implement infrastructure as code, creating a 3-virtual-machine cluster within a public cloud provider of your choice. The essential resources that Terraform must provision include networking components such as VPC, Internet Gateway, Route Table, Subnets, three virtual machine instances, one storage volume attached to each instance, and DNS records.”

I have chosen AWS as the cloud provider. The Terraform code has three components, a **provider.tf** file that states we are using AWS, **backend.tf** to store the .tfstate file in an AWS S3 bucket remotely, to not expose sensitive info in the git repo, and the **main.tf** that holds all the infrastructure configuration, including the network. Multiple ports are open to cover the various apps and services that will be running on the nodes.

“use Ansible playbooks to carry out the following tasks: update and upgrade system packages, configure a unique hostname for each VM, and install Docker on all instances. ”

This is achieved via two playbooks within the Ansible directory, **hostnames.yml** which sets unique hostnames, updates all system packages, and performs some dhcp changes to the network interfaces which can not be covered by terraform, and **docker.yml** which installs docker and other system prerequisites, initializes the swarm and adds the other two nodes as workers to the swarm. Output from the docker nodes ls command, where indeed the naming convention is based on a k8s cluster:

```
[root@k8s-master ~]# docker node ls
ID                            HOSTNAME       STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
nsnvqz8vjpfy4fyf1o37spr9r *   k8s-master     Ready     Active         Leader           25.0.6
m1mc2o9aqouj44af30h78sf5y     k8s-worker-1   Ready     Active                          25.0.6
lsr7a8xg11230yf1g8dlcpzsi     k8s-worker-2   Ready     Active                          25.0.6
[root@k8s-master ~]#
```

“disable password authentication on all VMs, ensuring that login is only possible via SSH keys”

This is achieved with **disable_password_login.yml**, it changes the sshd config and restarts the sshd service. A key pair was manually generated and uploaded to AWS which is specified to be added to the EC2 instances via Terraform.

“Implement the following firewall rules using iptables on all the instances:”

This is already implemented via Terraform, butI’ve created **iptables.yml** for demonstration purposes.

“set up a VPN network connecting all the servers [...] add a firewall rule that restricts traffic on the VPN interface to ICMP only, ensuring that each VM can ping the other two”

Achieved via **vpn.yml**, which installs and configures WireGuard on the three nodes in a server-client topology, creating a dedicated interface on all the machines along with assigning IPs and registering the clients to the server. Everything runs successfully, but to be honest this is not an area I'm an expert in, took an awful long time to get done, and I'm still not quite sure it's worked 🤷.

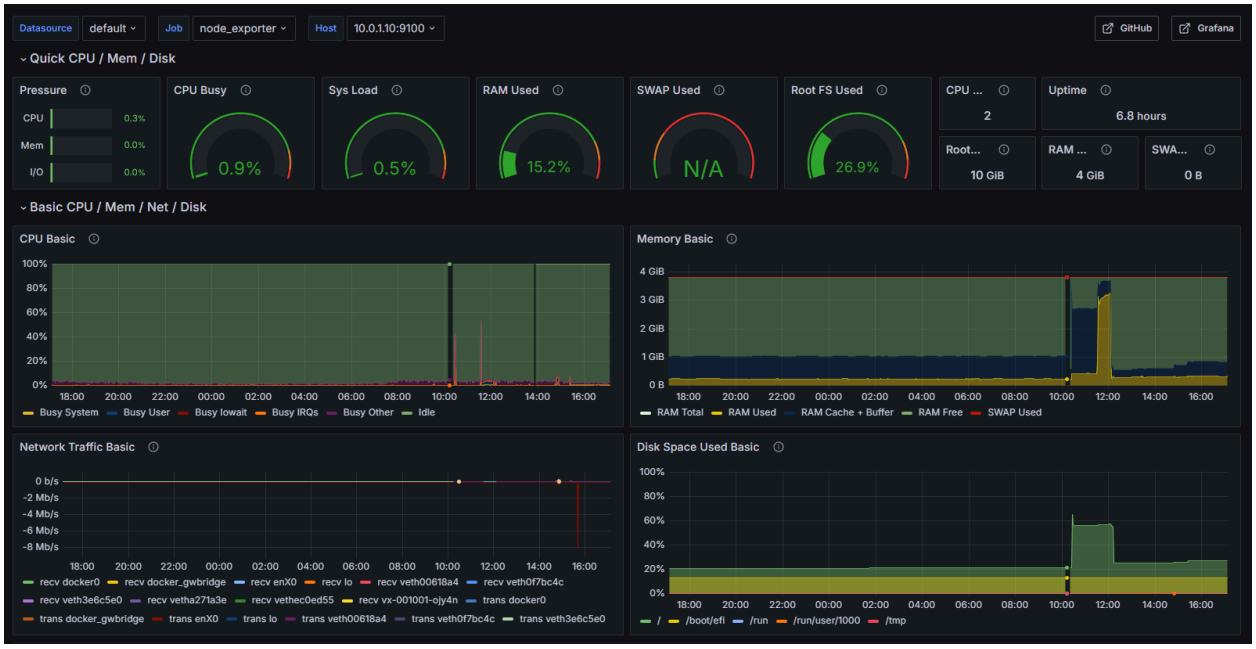
"Deploy a monitoring stack that includes Prometheus, Alertmanager, Blackbox Exporter, Pushgateway, and Grafana."

Here we have multiple playbooks to achieve the full stack, split up as follows:
Prometheus.yml handles the installation and configuration of prometheus and alertmanager on the main node, along with copying the alert rules which have been defined for CPU, Memory, HTTP response codes and State. It relies on simple bash scripts to fetch the packages and perform the installations and copies over unit files so both solutions may be managed via systemd.

Exporters.yml installs and configures node_exporter, blackbox_exporter(both as systemd services) and a Docker Swarm exporter as a Docker deployment to obtain metrics on system performance, HTTP response codes and Docker containers.

Grafana.yml does the installation and configuration of Grafana while copying over a ready made config, while the three dashboards have been added to grafana manually, via community made templates to save time.

Smtp.yml handles the installation of postfix and configures it to use the Google smtp server for outgoing messages, Alertmanager then connects to it to send out alerts based on the configured events and also Resolved status notifications. By stopping the node_exporter service on one of the nodes we can simulate it being down, confirming that alerting and notifications work.



Targets

All scrape pools | All | Unhealthy | Expand All | Filter by endpoint or labels

blackbox_exporter (5/5 up) show less

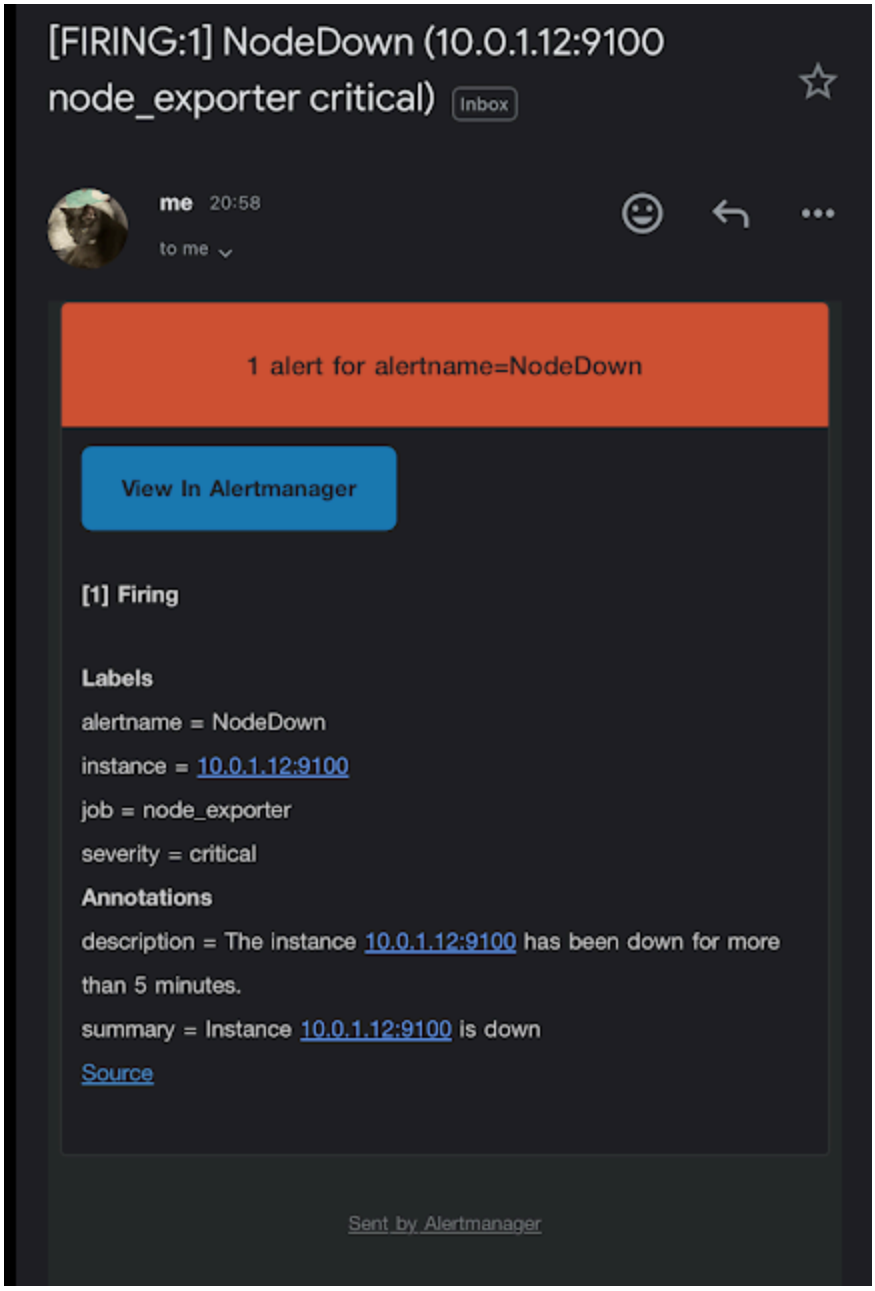
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.0.1.12:9102/probe	UP	instance="http://10.0.1.12:9106" job="blackbox_exporter"	13m 0s ago	2.136ms	
module="http_2xx" target="http://10.0.1.12:9106"					
http://10.0.1.12:9102/probe	UP	instance="http://10.0.1.11:9091" job="blackbox_exporter"	13m 9s ago	5.420ms	
module="http_2xx" target="http://10.0.1.11:9091"					
http://10.0.1.12:9102/probe	UP	instance="http://10.0.1.11:8100" job="blackbox_exporter"	13m 10s ago	13.539ms	
module="http_2xx" target="http://10.0.1.11:8100"					
http://10.0.1.12:9102/probe	UP	instance="http://10.0.1.10:9106" job="blackbox_exporter"	13m 9s ago	3.525ms	
module="http_2xx" target="http://10.0.1.10:9106"					
http://10.0.1.12:9102/probe	UP	instance="http://10.0.1.11:9106" job="blackbox_exporter"	13m 3s ago	2.570ms	
module="http_2xx" target="http://10.0.1.11:9106"					

cadvisor (3/3 up) show more

docker (1/1 up) show more

node_exporter (3/3 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.0.1.10:9100/metrics	UP	instance="10.0.1.10:9100" job="node_exporter"	13m 14s ago	14.787ms	
http://10.0.1.11:9100/metrics	UP	instance="10.0.1.11:9100" job="node_exporter"	13m 5s ago	16.727ms	
http://10.0.1.12:9100/metrics	UP	instance="10.0.1.12:9100" job="node_exporter"	13m 2s ago	11.771ms	



CI/CD

This component is handled by **ci-cd.yml**. It is unfortunately not tested to its full capacity, as even though the playbook handles the deployment of a gitlab stack successfully, it cripples the EC2 instance. The playbook also does a rough pass at adding a gitlab runner to run the CI/CD config, but the deployment of a simple hello world app is ultimately achieved manually.

Infrastructure cost report

I am able to produce an estimate of the total and granular costs of running this infrastructure as is, taking into account that network costs should be null given Amazon's 1GB free per month, s3 costs also free as only a small file is being stored.

Service	Cost per month	Units	Total per service/month
EC2 t2.medium	33.872	3	101.616
EBS Storage	0.10	30	3
Total			104.616