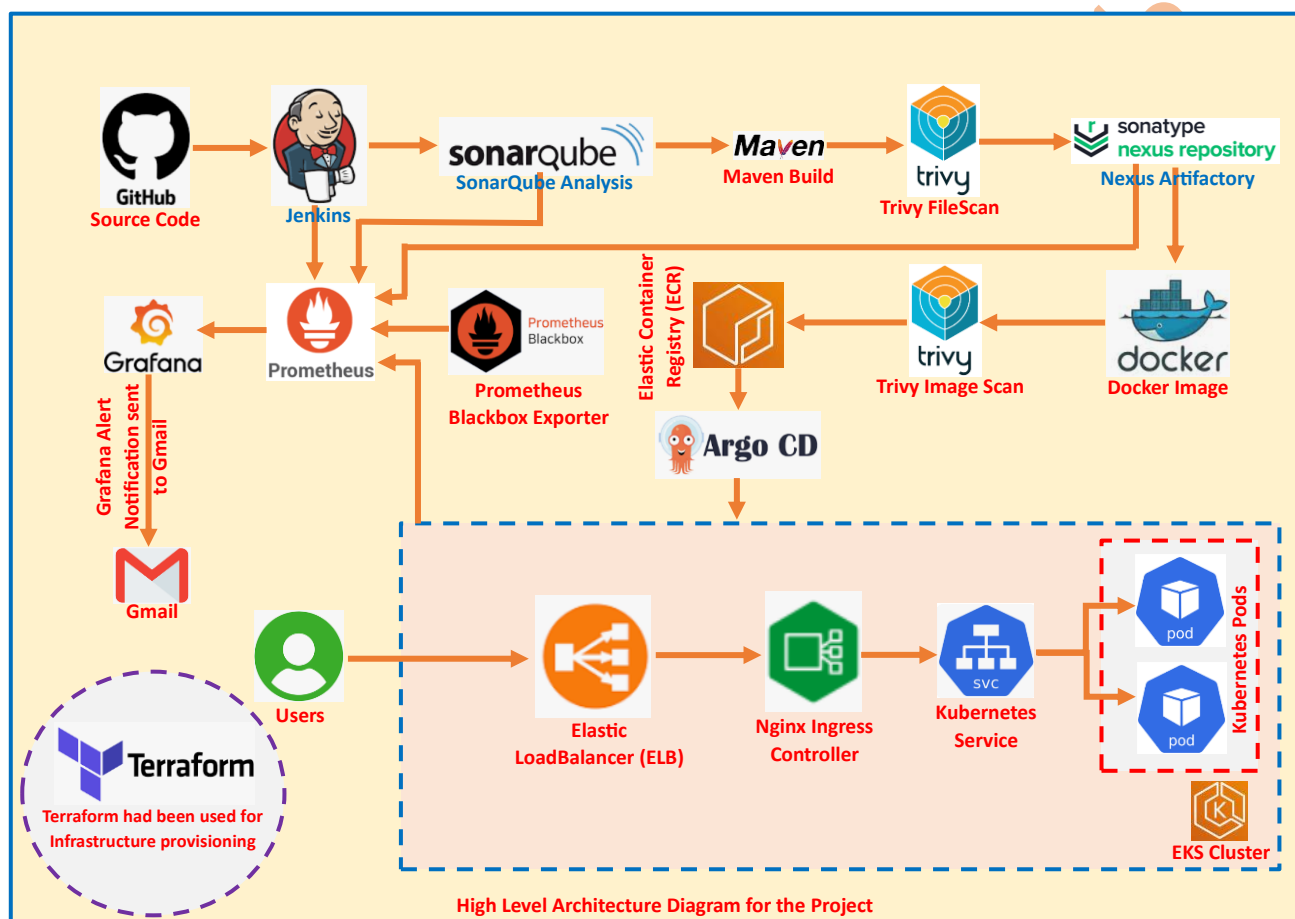


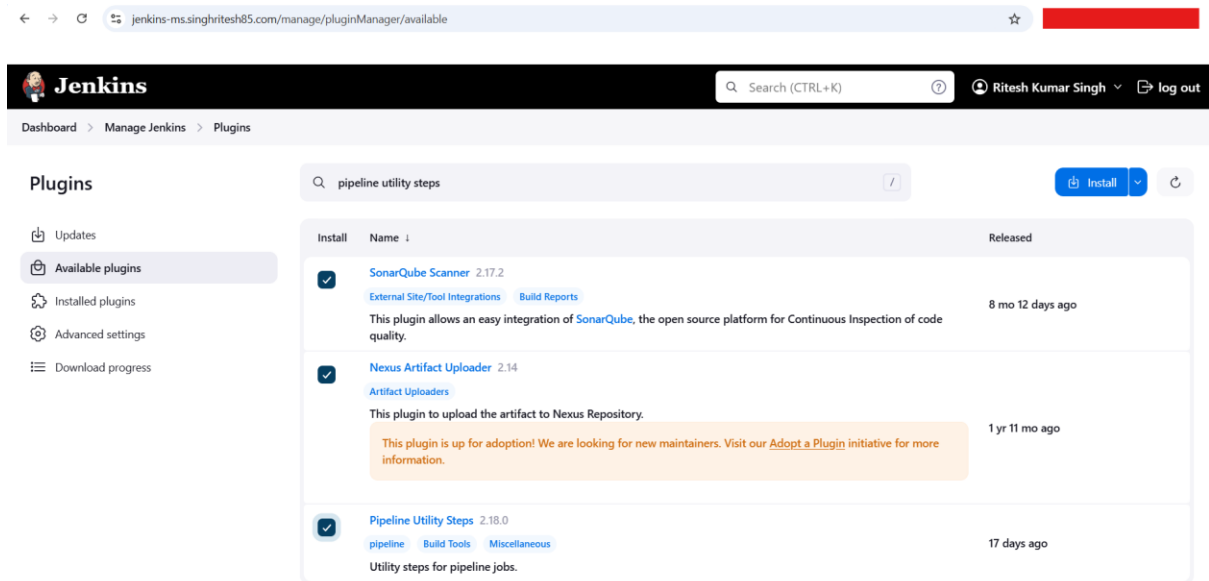
DevOps Project Boardgame

This DevOps project aims to create the infrastructure for the project, establish the end-to-end CI/CD setup and its monitoring using Prometheus and Grafana as Monitoring Tool. Terraform had been used for Infrastructure provisioning. Jenkins was used as CI/CD Tool. For Code Quality check SonarQube, to keep the Artifacts Nexus and Maven was used as a Build Tool. Trivy was used as a Security Scanning tool, Docker Image was stored in ECR (Elastic Container Registry) and for deployment ArgoCD had been used. The high-level architecture diagram for the project is as shown below.



The source code was present in the GitHub Repository and Jenkins (with Master-Slave Architecture) was used as the CI/CD Tool. SonarQube and Maven was used as Code Analysis and Build Tool respectively as shown in the high-level architecture diagram above. Trivy was used as a security Scanning Tool for File Scan and Docker Image Scan in the later stage after creation of Docker Image. Sonatype Nexus was used to keep the Artifacts then Docker Image was created which was scanned using Trivy Image Scan as explained earlier. Elastic Container Registry (ECR) was used to store the Docker Image. The Deployment had been done with the created Docker Image present in the ECR. The Nginx Ingress Controller was created proceeded by creation of ingress with ingress rule to route the incoming traffic to the service and hence the Application Pod. Finally, the URL will be created with the DNS Name of the Elastic LoadBalancer of Nginx Ingress Controller and started accessing the Application using the created URL.

For setting up CI/CD Set-up using Jenkins installed SonarQube Scanner, Nexus Artifact Uploader and Pipeline Utility Steps plugins in Jenkins as shown in the screenshot attached below.



Created **node-exporter** pod using **daemonset** on each worker nodes with the help of helm chart using the command as shown in the screenshot attached below. Whenever a new worker node will be created a pod of node-exporter will be created as a part of **daemonset**.

helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>

kubectl create ns node-exporter

helm install my-prometheus-node-exporter prometheus-community/prometheus-node-exporter --version 4.37.1 --set service.type=LoadBalancer -n node-exporter

The Kubernetes Pods and Service for Node-Exporter is as shown in the screenshot attached below. **Node-Exporter will extract the metrics from the EKS Cluster and send them to the Prometheus Server.** Here I had created centralized Prometheus and Grafana set-up on EC2 Instances. It is also possible to create the Prometheus and Grafana Pods using the helm chart with persistent volume but for this project I had used centralized Prometheus and Grafana set-up on EC2 Instances.

```
[root@~]# kubectl get all -n node-exporter
```

NAME	READY	STATUS	RESTARTS	AGE
pod/my-prometheus-node-exporter-	1/1	Running	0	76s
pod/my-prometheus-node-exporter-	1/1	Running	0	76s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/my-prometheus-node-exporter	LoadBalancer	172.20.204.18	5.us-east-2.elb.amazonaws.com	9100:30460/TCP

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE
daemonset.apps/my-prometheus-node-exporter	2	2	2	2	2	kubernetes.io/os=linux	76s

To customize the DNS setting On Jenkins Slave, /etc/resolv.conf file had been updated as shown in the screenshot attached below.

```
[root@██████████ ~]# cat /etc/resolv.conf
; generated by /usr/sbin/dhclient-script
search singhritesh85.com #us-east-2.compute.internal
options timeout:2 attempts:5
nameserver 8.8.8.8 #10.10.0.2
```

I had provided all the access to the deployment **user Jenkins**, specific to the **namespace boardgame** as shown in the screenshot attached below. To do so I had used service account with the name as Jenkins (same as the username). Created Role with Admin privileges in the Namespace boardgame and bonded this role with service account with the help of role binding as shown in the screenshot attached below.

```
[root@██████████ ~]# kubectl apply -f sa-role-relebinding.yaml
serviceaccount/jenkins created
role.rbac.authorization.k8s.io/user-role created
rolebinding.rbac.authorization.k8s.io/user-rolebinding created
[root@██████████ ~]# cat sa-role-relebinding.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: jenkins
  namespace: boardgame
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: user-role
  namespace: boardgame
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: user-rolebinding
  namespace: boardgame
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: user-role
subjects:
- namespace: boardgame
  kind: ServiceAccount
  name: jenkins
```

cat sa-role-relebinding.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: jenkins

namespace: boardgame

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

name: user-role

namespace: boardgame

rules:

- apiGroups: ["*"]

resources: ["*"]

verbs: ["*"]

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: user-rolebinding

namespace: boardgame

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: Role

name: user-role

subjects:

- namespace: boardgame

kind: ServiceAccount

name: jenkins

Created Kubernetes secret to provide the Token based Access for the deployment user Jenkins.

```
[root@██████████ ~]# kubectl apply -f secret.yaml
secret/mysecretname created
[root@██████████ ~]# cat secret.yaml
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: mysecretname
  namespace: boardgame
  annotations:
    kubernetes.io/service-account.name: jenkins
```

cat secret.yaml

apiVersion: v1

kind: Secret

type: kubernetes.io/service-account-token

metadata:

name: mysecretname

namespace: boardgame

annotations:

kubernetes.io/service-account.name: jenkins

```
[root@██████████ ~]# kubectl get secrets -n boardgame
NAME                TYPE                DATA      AGE
mysecretname        kubernetes.io/service-account-token    3         3m11s
[root@██████████ ~]# kubectl describe secret mysecretname -n boardgame
Name:         mysecretname
Namespace:    boardgame
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: jenkins
              kubernetes.io/service-account.uid: 3██████████
Type:         kubernetes.io/service-account-token

Data
====
token:        e██████████3
ca.crt:       1107 bytes
namespace:    9 bytes
```

Finally, I had shared the below kubeconfig file with the deployment user Jenkins as shown in the screenshot attached below. So that the user jenkins can access only the namespace boardgame with the admin privileges.

```

[jenkins@ip-10-10-4-200 ~]$ cat ~/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: [REDACTED]
    server: https://[REDACTED]EF.sk1.us-east-2.eks.amazonaws.com
    name: arn:aws:eks:us-east-2:[REDACTED]:cluster/eks-demo-cluster-dev
contexts:
- context:
    cluster: arn:aws:eks:us-east-2:[REDACTED]:cluster/eks-demo-cluster-dev
    user: jenkins
    name: dexter
current-context: dexter
kind: Config
preferences: {}
users:
- name: jenkins
  user:
    token: [REDACTED]

[jenkins@ip-10-10-4-200 ~]$ kubectl get nodes
Error from server (Forbidden): nodes is forbidden: User "system:serviceaccount:boardgame:jenkins" cannot list resource "nodes" in API group "" at the cluster scope
[jenkins@ip-10-10-4-200 ~]$ kubectl get secrets -n boardgame
NAME      TYPE      DATA  AGE
mysecretname  kubernetes.io/service-account-token  3      9m10s

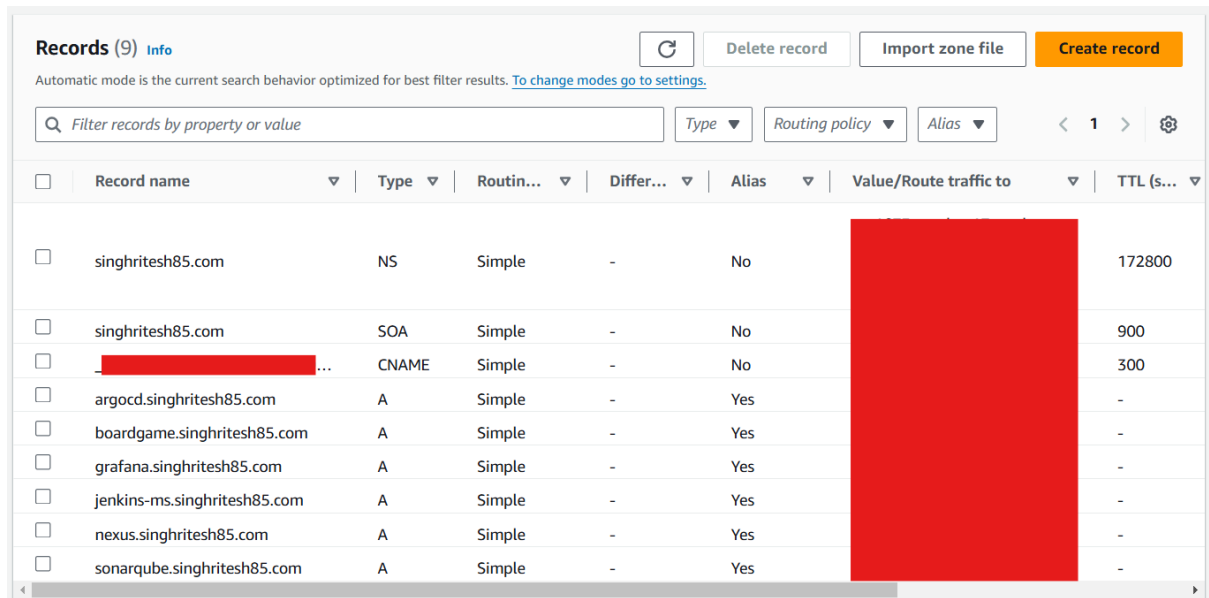
```

Ritesh Kumar S

The ingress had been created using which I accessed the Kubernetes Service and hence the Application running in the Kubernetes Pods.

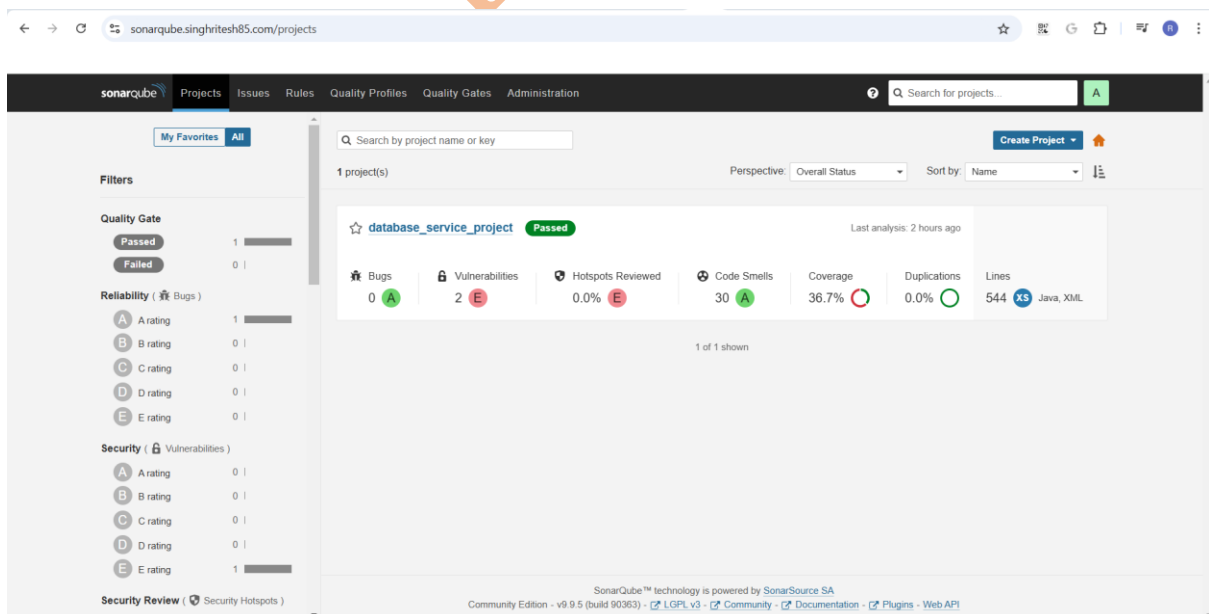
```
[root@ip-10-10-4-200 ~]# kubectl get ing -A
NAMESPACE   NAME           CLASS   HOSTS                                ADDRESS                                PORTS   AGE
argocd       minimal-ingress  nginx   argocd.singhritesh85.com             [REDACTED] 7-1 [REDACTED] 4.us-east-2.elb.amazonaws.com 80      1m
boardgame    boardgame-ingress  nginx   boardgame.singhritesh85.com           [REDACTED] 7-1 [REDACTED] 4.us-east-2.elb.amazonaws.com 80      1m
```

The entry for Route53 to create the record set is as shown in the screenshot attached below.



Record name	Type	Routin...	Differ...	Alias	Value/Route traffic to	TTL (s...)
singhritesh85.com	NS	Simple	-	No	[REDACTED]	172800
singhritesh85.com	SOA	Simple	-	No	[REDACTED]	900
[REDACTED]...	CNAME	Simple	-	No	[REDACTED]	300
argocd.singhritesh85.com	A	Simple	-	Yes	[REDACTED]	-
boardgame.singhritesh85.com	A	Simple	-	Yes	[REDACTED]	-
grafana.singhritesh85.com	A	Simple	-	Yes	[REDACTED]	-
jenkins-ms.singhritesh85.com	A	Simple	-	Yes	[REDACTED]	-
nexus.singhritesh85.com	A	Simple	-	Yes	[REDACTED]	-
sonarqube.singhritesh85.com	A	Simple	-	Yes	[REDACTED]	-

The screenshot for SonarQube, Nexus and ArgoCD after running the Jenkins Job is as shown in the screenshot attached below.

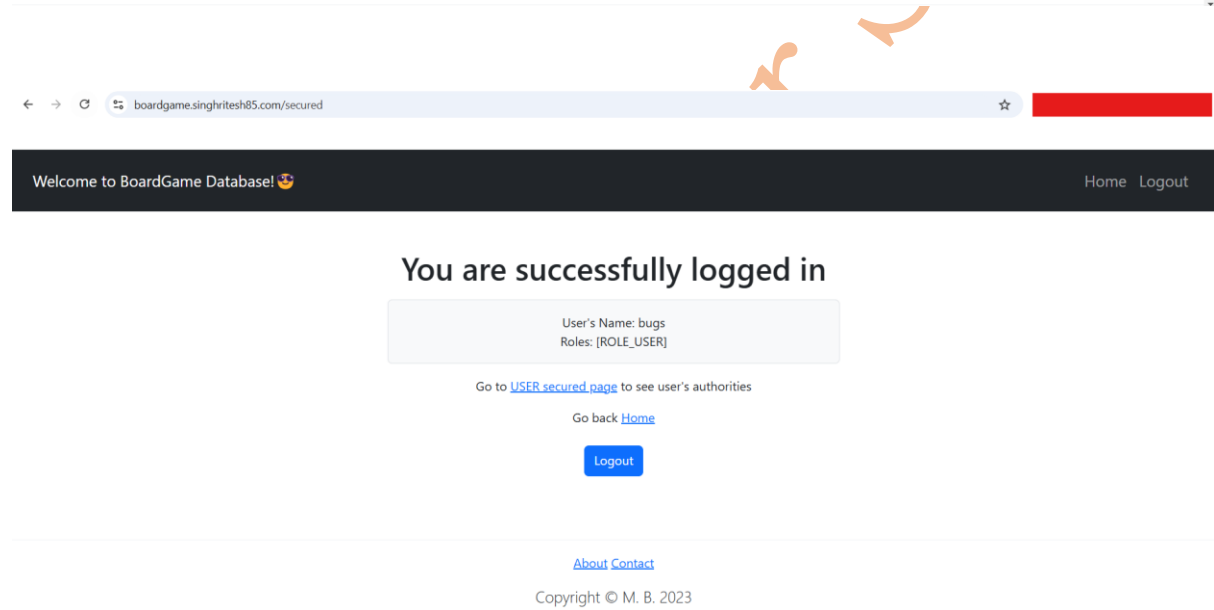
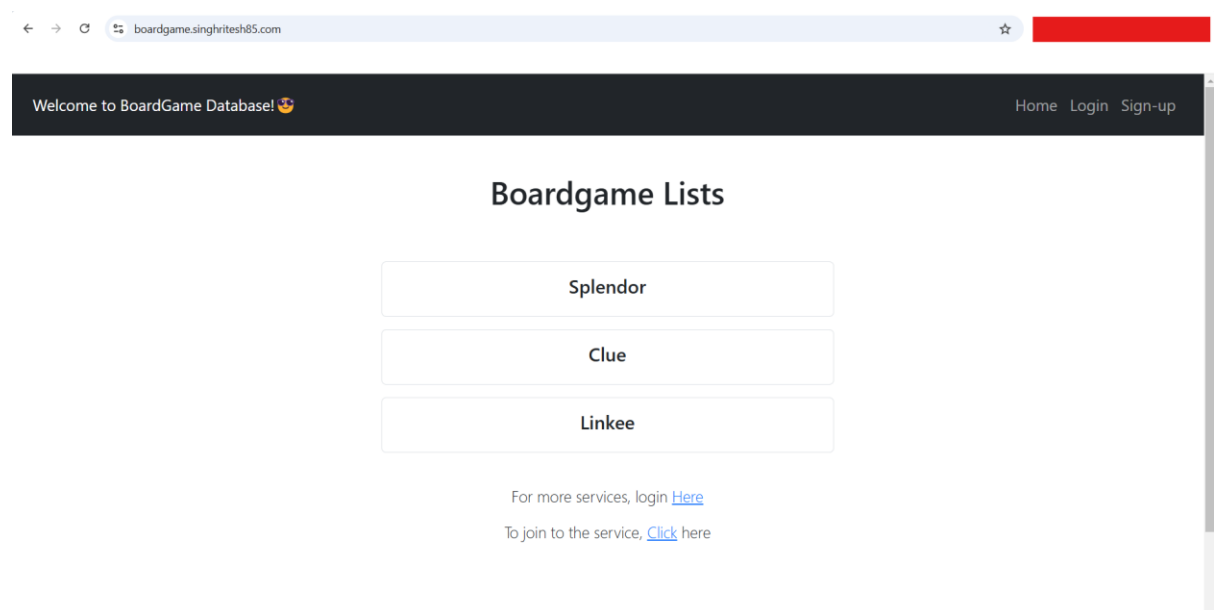


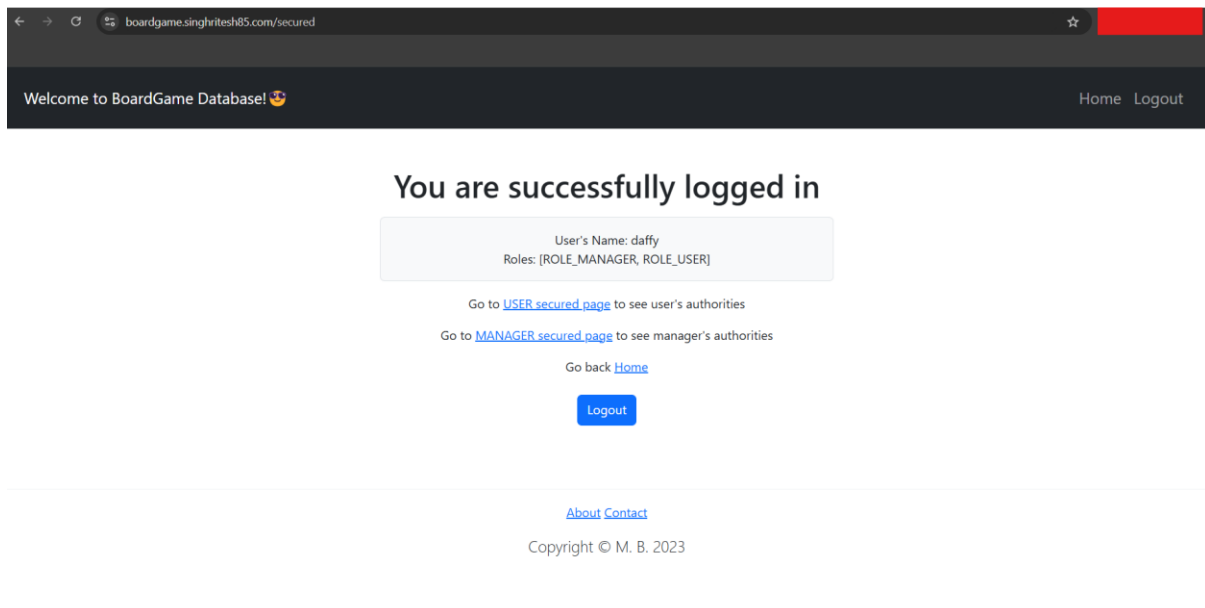
The screenshot shows the SonarQube web interface for the project 'database_service_project'. The project status is 'Passed'. The interface displays various quality metrics:

- Bugs: 0 (A)
- Vulnerabilities: 2 (E)
- Hotspots Reviewed: 0.0% (E)
- Code Smells: 30 (A)
- Coverage: 36.7% (C)
- Duplications: 0.0% (C)
- Lines: 544 (XS)

The interface also shows a 'Quality Gate' section with 'Passed' and 'Failed' counts, and a 'Security Review' section with 'Security Hotspots'.

Finally Access the Application using the created URL as shown in the screenshot attached below.





Monitoring using Prometheus and Grafana

For Monitoring I had used Prometheus and Grafana as monitoring tool. Node-Exporter will extract the metrics from the Jenkins-Master, Jenkins-Slave, SonarQube-Server, Nexus-Server, Prometheus-Server, Blackbox-Exporter Server, Grafana-Server and EKS Cluster and send to the Prometheus Server. The scrap_config section in the configuration file of Prometheus is as shown in the screenshot attached below. I had installed Blackbox Exporter on a different server and not on the Prometheus Server. **The module name which is shown in the yellow colour highlight for blackbox exporter must match with the module name of blackbox exporter configuration file (monitor_website.yml)** present of the blackbox exporter server at the path (/opt/blackbox_exporter_linux_amd64/monitor_website.yml). Prometheus blackbox operator is used for endpoint monitoring (Synthetic Monitoring) across the protocol http, https, TCP and ICMP. In this project I am monitoring the Application URL <https://boardgame.singhritesh85.com> with the help of Prometheus Blackbox-Exporter. Prometheus blackbox exporter will send the metrics to Prometheus. For this project Prometheus acts as a DataSource for Grafana and send metrics to Grafana which we can see with the help of Charts and Graphs.

```

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
  - job_name: "prometheus-server"
    static_configs:
      - targets: ["localhost:9100"]
  - job_name: "Grafana-server"
    static_configs:
      - targets: ["10.10.4.130:9100"]
  - job_name: "BlackboxExporter-server"
    static_configs:
      - targets: ["10.10.4.208:9100"]
  - job_name: "Jenkins-Master"
    static_configs:
      - targets: ["10.10.4.34:9100"]
  - job_name: "Jenkins-Slave"
    static_configs:
      - targets: ["10.10.4.200:9100"]
  - job_name: "SonarQube-Server"
    static_configs:
      - targets: ["10.10.4.88:9100"]
  - job_name: "Nexus-Server"
    static_configs:
      - targets: ["10.10.4.225:9100"]
  - job_name: "EKS"
    static_configs:
      - targets: ["a35be51fea2884edc99c32a74655676b-1950273345.us-east-2.elb.amazonaws.com:9100"]
  - job_name: 'blackbox'
    metrics_path: /probe
    params:
      module: [http_2xx_example] # Look for a HTTP 200 response.
    static_configs:
      - targets:
        - https://boardgame.singhritesh85.com
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: 10.10.4.208:9115 # The blackbox exporter's real hostname:port.

```

Ritesh K

```
cat /etc/prometheus/prometheus.yml

# my global config

global:

  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration

alerting:

  alertmanagers:
    - static_configs:
        - targets:
            # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label 'job=<job_name>' to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
  - job_name: "prometheus-server"

    static_configs:
      - targets: ["localhost:9100"]
  - job_name: "Grafana-server"

    static_configs:
      - targets: ["10.10.4.130:9100"]
  - job_name: "BlackboxExporter-server"

    static_configs:
```

```

- targets: ["10.10.4.208:9100"]
- job_name: "Jenkins-Master"
static_configs:
- targets: ["10.10.4.34:9100"]
- job_name: "Jenkins-Slave"
static_configs:
- targets: ["10.10.4.200:9100"]
- job_name: "SonarQube-Server"
static_configs:
- targets: ["10.10.4.88:9100"]
- job_name: "Nexus-Server"
static_configs:
- targets: ["10.10.4.225:9100"]
- job_name: "EKS"
static_configs:
- targets: ["a35be51fea2884edc99c32a74655676b-1950273345.us-east-2.elb.amazonaws.com:9100"]
- job_name: 'blackbox'
metrics_path: /probe
params:
  module: [http_2xx_example] # Look for a HTTP 200 response.
static_configs:
- targets:
- https://boardgame.singhritesh85.com
relabel_configs:
- source_labels: [__address__]
  target_label: __param_target
- source_labels: [__param_target]
  target_label: instance
- target_label: __address__
replacement: 10.10.4.208:9115 # The blackbox exporter's real hostname:port.

```

For prometheus blackbox exporter I had used the ID **7587** to create the Grafana Dashboard as shown in the screenshot attached below.



To monitor Jenkins Job using Prometheus and Grafana I installed the plugin Prometheus metrics on Jenkins and do the changes in configuration file for Prometheus as shown in the screenshots attached below.

The screenshot shows the Jenkins web interface at `jenkins-ms.singhritesh85.com/manage/pluginManager/available`. The user 'Ritesh Kumar Singh' is logged in. The 'Prometheus metrics' plugin is listed under 'Available plugins' with an 'Install' button. The plugin details show it was released '1 mo 7 days ago'.

REST API Jenkins 2.462.3

```
# metrics_path defaults to '/metrics'
# scheme defaults to 'http'.

static_configs:
  - targets: ["localhost:9090"]
- job_name: "prometheus-server"
  static_configs:
    - targets: ["localhost:9100"]
- job_name: "Grafana-server"
  static_configs:
    - targets: ["10.10.4.130:9100"]
- job_name: "BlackboxExporter-server"
  static_configs:
    - targets: ["10.10.4.208:9100"]
- job_name: "Jenkins-Master"
  static_configs:
    - targets: ["10.10.4.34:9100"]
- job_name: "Jenkins-Slave"
  static_configs:
    - targets: ["10.10.4.200:9100"]
- job_name: "SonarQube-Server"
  static_configs:
    - targets: ["10.10.4.88:9100"]
- job_name: "Nexus-Server"
  static_configs:
    - targets: ["10.10.4.225:9100"]
- job_name: "EKS"
  static_configs:
    - targets: ["a35be51fea2884edc99c32a74655676b-1950273345.us-east-2.elb.amazonaws.com:9100"]
- job_name: "Jenkins-Job"
  metrics_path: '/prometheus'
  static_configs:
    - targets: ["10.10.4.34:8080"]
- job_name: "blackbox"
  metrics_path: /probe
  params:
    module: [http_2xx_example] # Look for a HTTP 200 response.
  static_configs:
```

```
- job_name: "Jenkins-Job"

  metrics_path: '/prometheus'

  static_configs:

    - targets: ["10.10.4.34:8080"]
```

As I did changes in configuration file of Prometheus and hence I restarted the Prometheus service which is shown in the screenshot attached below.

```
[root@ ~]# systemctl restart prometheus.service
[root@ ~]# systemctl status prometheus.service
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2024-10-29 15:07:31 UTC; 10s ago
     Main PID: 3984 (prometheus)
    CGroup: /system.slice/prometheus.service
            └─3984 /usr/local/bin/prometheus --config.file=/etc/prometheus/prometheus.yml --storage.tsdb.path=/data --web.console.templates=/etc/prometheus/...

Oct 29 15:07:32 ip-10-10-4-223.us-east-2.compute.internal prometheus[3984]: ts=2024-10-29T15:07:32.158Z caller=head.go:760 level=info component=tsdb ...ent=10
Oct 29 15:07:32 ip-10-10-4-223.us-east-2.compute.internal prometheus[3984]: ts=2024-10-29T15:07:32.203Z caller=head.go:760 level=info component=tsdb ...ent=10
Oct 29 15:07:32 ip-10-10-4-223.us-east-2.compute.internal prometheus[3984]: ts=2024-10-29T15:07:32.204Z caller=head.go:760 level=info component=tsdb ...ent=10
Oct 29 15:07:32 ip-10-10-4-223.us-east-2.compute.internal prometheus[3984]: ts=2024-10-29T15:07:32.204Z caller=head.go:797 level=info component=tsdb ...8346ms
Oct 29 15:07:32 ip-10-10-4-223.us-east-2.compute.internal prometheus[3984]: ts=2024-10-29T15:07:32.215Z caller=main.go:1045 level=info fs_type=XFS_SUPER_MAGIC
Oct 29 15:07:32 ip-10-10-4-223.us-east-2.compute.internal prometheus[3984]: ts=2024-10-29T15:07:32.215Z caller=main.go:1048 level=info msg="TSDB started"
Oct 29 15:07:32 ip-10-10-4-223.us-east-2.compute.internal prometheus[3984]: ts=2024-10-29T15:07:32.216Z caller=main.go:1229 level=info msg="Loading c...us.yml
Oct 29 15:07:32 ip-10-10-4-223.us-east-2.compute.internal prometheus[3984]: ts=2024-10-29T15:07:32.220Z caller=main.go:1266 level=info msg="Completed loadi...us
Oct 29 15:07:32 ip-10-10-4-223.us-east-2.compute.internal prometheus[3984]: ts=2024-10-29T15:07:32.220Z caller=main.go:1009 level=info msg="Server is...ests."
Oct 29 15:07:32 ip-10-10-4-223.us-east-2.compute.internal prometheus[3984]: ts=2024-10-29T15:07:32.220Z caller=manager.go:1009 level=info component="...er..."
Hint: Some lines were ellipsized, use -l to show in full.
```


← → ↻ Not secure 3.147.29.195:9090/targets?search= ☆ 🗖 📄 📄 📄 📄 📄 📄

Prometheus Alerts Graph Status ▾ Help ⚙️ 🌙 ⓘ

Grafana-server (1/1 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.10.4.130:9100/metrics	UP	instance="10.10.4.130:9100" job="Grafana-server"	20.197s ago	15.021ms	

Jenkins-Job (1/1 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.10.4.34:8080/prometheus	UP	instance="10.10.4.34:8080" job="Jenkins-Job"	19.740s ago	10.683ms	

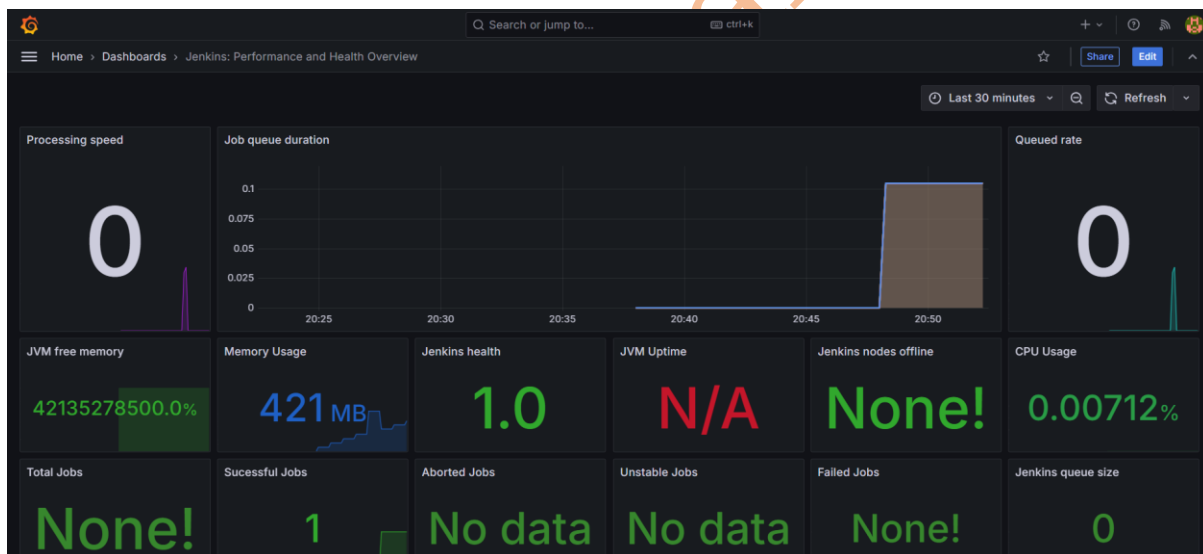
Jenkins-Master (1/1 up) [show less](#)

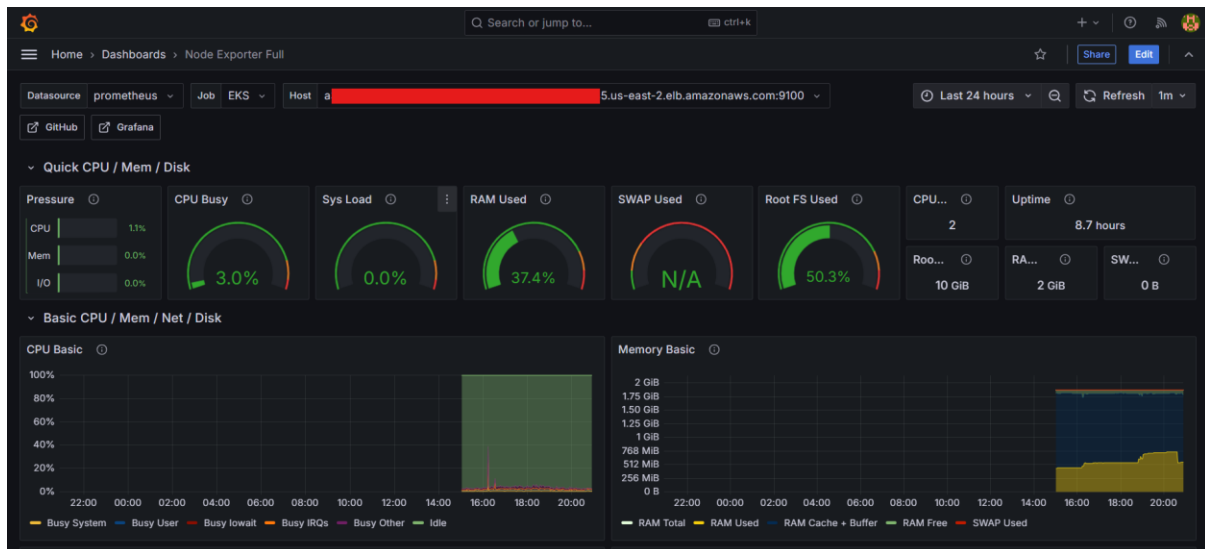
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.10.4.34:9100/metrics	UP	instance="10.10.4.34:9100" job="Jenkins-Master"	15.649s ago	48.328ms	

Jenkins-Slave (1/1 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.10.4.200:9100/metrics	UP	instance="10.10.4.200:9100" job="Jenkins-Slave"	7.463s ago	21.339ms	

To create the Grafana Dashboard for Jenkins Job and for Node Exporter I used the ID **9664** and **1860** respectively. Finally, the created Grafana Dashboard is as shown in the screenshots shown below.





Configuration of Alerts in Grafana

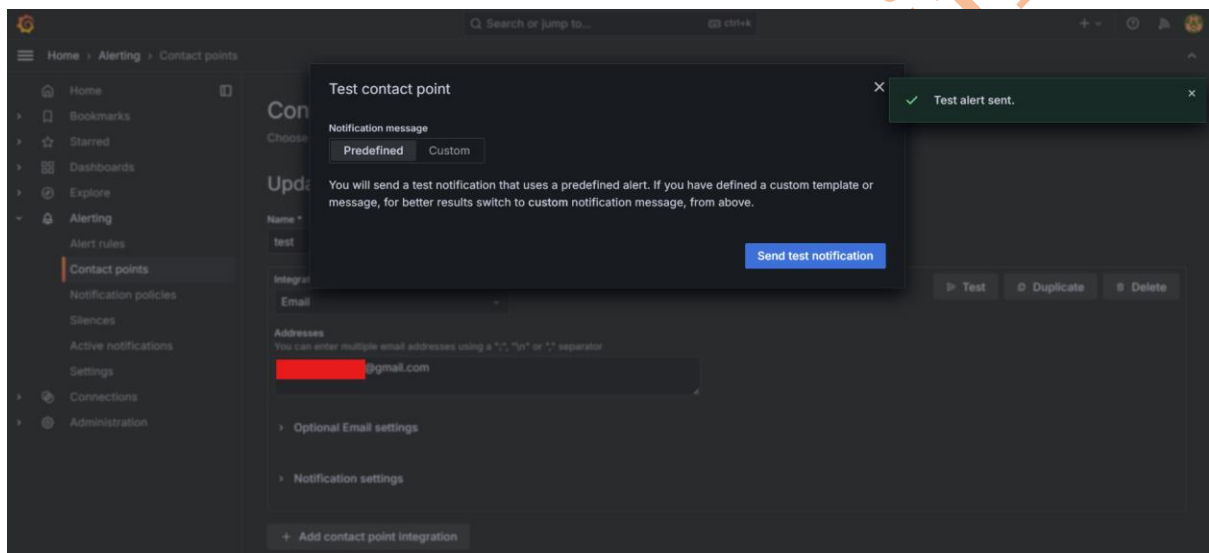
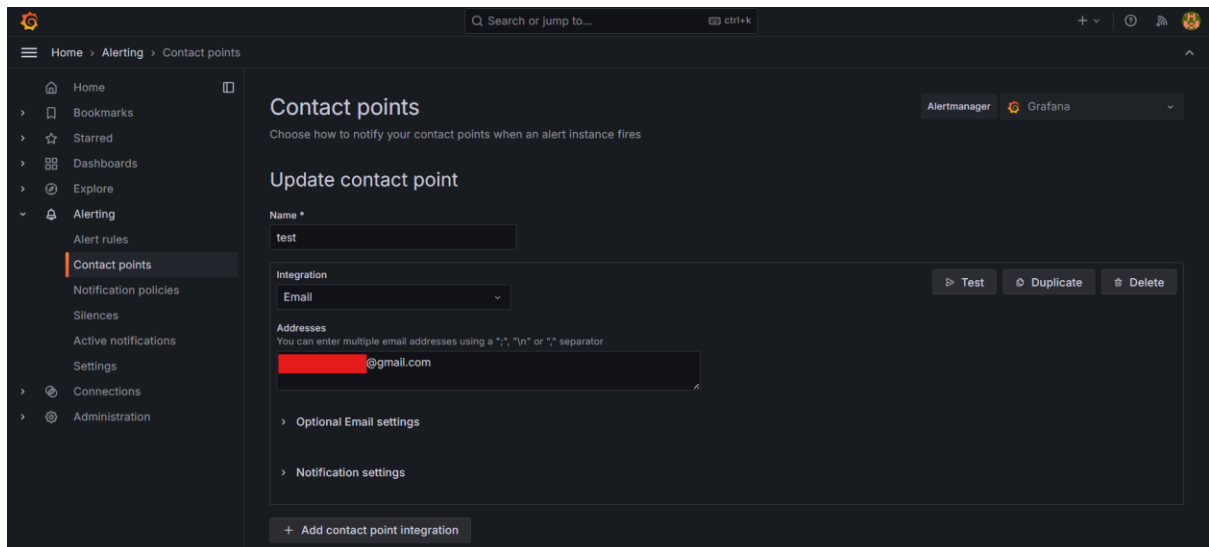
To configure Alerts in Grafana, first I created **contact points** with the Email ID and changed smtp settings in the configuration file `/etc/grafana/grafana.ini` of Grafana as shown in the screenshot attached below.

```
[root@ ~]# cat /etc/grafana/grafana.ini
```

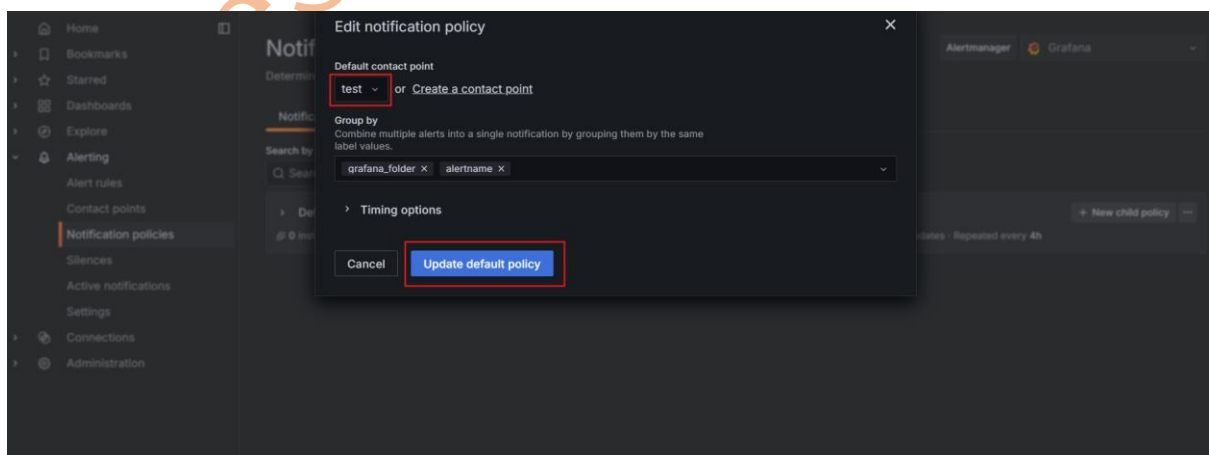
```
##### SMTP / Emailing #####
[smtp]
enabled = true
host = smtp.gmail.com:587
user = [REDACTED]@gmail.com
# If the password contains # or ; you have to wrap it with triple quotes. Ex ""#password;""
password = [REDACTED]
;cert_file =
;key_file =
skip_verify = true
from_address = [REDACTED]@gmail.com
from_name = Grafana
# EHLO identity in SMTP dialog (defaults to instance_name)
;ehlo_identity = dashboard.example.com
# SMTP startTLS policy (defaults to 'OpportunisticStartTLS')
;starttls_policy = NoStartTLS
# Enable trace propagation in e-mail headers, using the 'traceparent', 'tracestate' and (optionally) 'baggage' fields (defaults to false)
;enable_tracing = false

[smtp.static_headers]
# Include custom static headers in all outgoing emails
;Foo-Header = bar
;Foo = bar

[emails]
;welcome_email_on_sign_up = false
;templates_pattern = emails/*.html, emails/*.txt
;content_types = text/html
```



The Default **Notification Policy** had been changed as shown in the screenshot attached below.



Configure **Alert Rule** as shown in the screenshot attached below.

The screenshot shows the 'Edit rule' page in Grafana. The left sidebar has 'Alerting' > 'Alert rules' selected. The main content area has two steps:

- 1. Enter alert rule name**
Enter a name to identify your alert rule.
Name:
- 2. Define query and alert condition**
Define query and alert condition. [Need help?](#)
A Options 10 minutes, MD = 43200, Min. Interval = 1s Set as alert condition
Kick start your query Explain ☐ Run queries Builder Code
Metric:
Label filters: instance =
+ Operations hint: add rate

The screenshot shows the 'Edit rule' page in Grafana, continuing from the previous step. The 'Add query' section shows the query: `process_cpu_seconds_total(instance="i-90.us-east-2.elb.amazonaws.com:9100")`. The 'Expressions' section shows the rule type set to 'Grafana-managed' and the expression set to 'Threshold' with a value of '0.7'.

Add query

Rule type: ☐ Grafana-managed ☐ Data source-managed
The alert rule type cannot be changed for an existing rule.

Expressions
Manipulate data returned from queries with math and other operations.

B Reduce Set as alert condition ☐
Takes one or more time series returned from a query or an expression and turns each series into a single number.
Input: Function: Last Mode: Strict
Add expression Preview

C Threshold ☒ Alert condition ☐
Takes one or more time series returned from a query or an expression and checks if any of the series match the threshold condition.
Input: IS ABOVE Custom recovery threshold ☐

The screenshot shows the 'Edit rule' page in Grafana, continuing from the previous step. The third step is '3. Set evaluation behavior'.

3. Set evaluation behavior
Define how the alert rule is evaluated. [Need help?](#)

Folder
Select a folder to store your rule.
 or + New folder

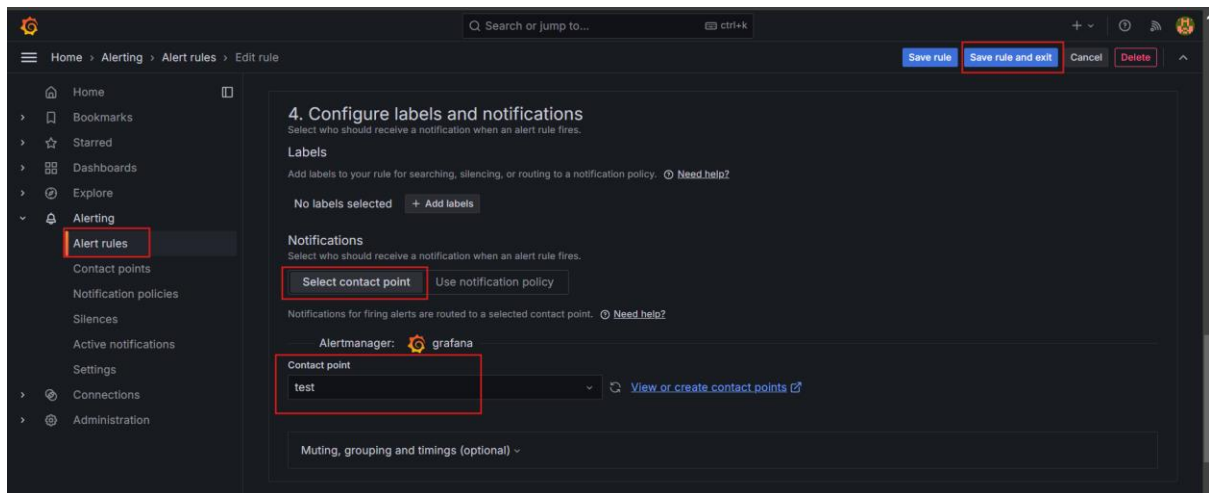
Evaluation group and interval
Define how often the alert rule is evaluated.
 or + New evaluation group

All rules in the selected group are evaluated every 1m.

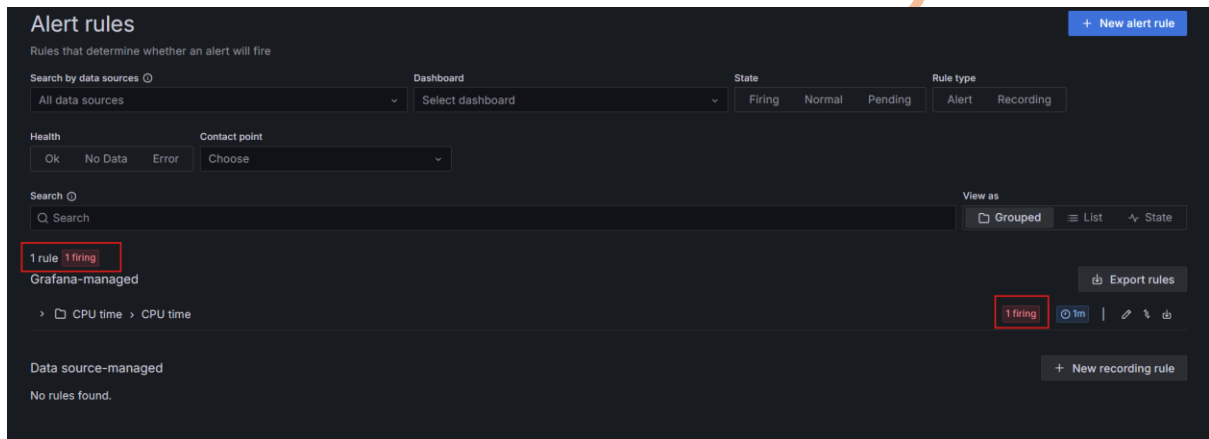
Pending period
Period the threshold condition must be met to trigger the alert. Selecting "None" triggers the alert immediately once the condition is met.

None 1m 2m 3m 4m 5m
☐ Pause evaluation ☐

> Configure no data and error handling



If the Alert Rule is in firing state after condition crosses the threshold condition, then Grafana console screenshot will be showing the same as shown in the screenshot attached below.



An Email will be sent to the Email ID as shown in the screenshot attached below.



📁 CPU time › Mederma

🔥 1 firing instances

Firing

Mederma

View alert

Values

A= B= C=

Labels

alertname

Mederma

grafana_folder

CPU time

instance

[ac90.us-east-2.elb.amazonaws.com:9100](#)

job

EKS

Ritesh Kumar