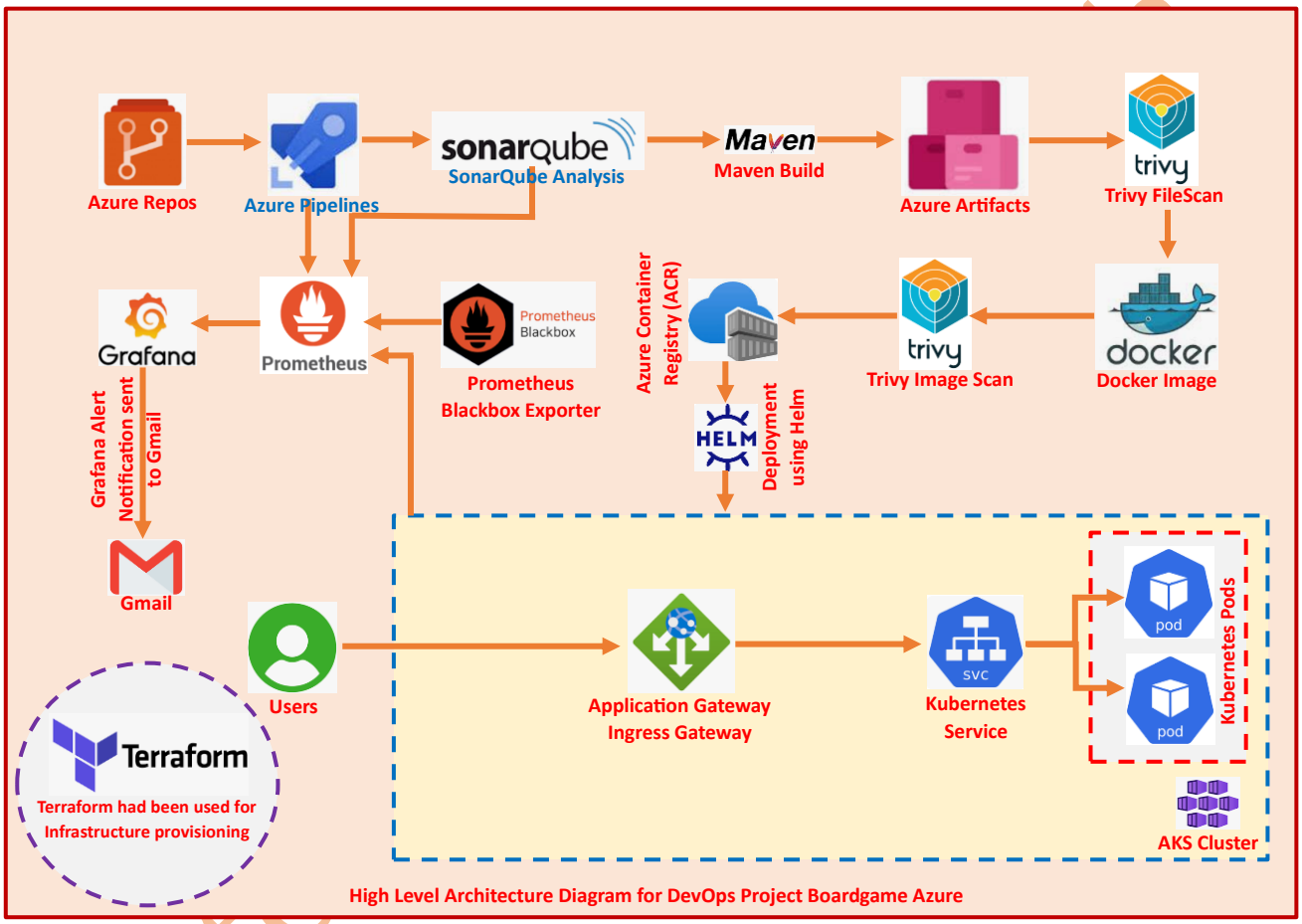
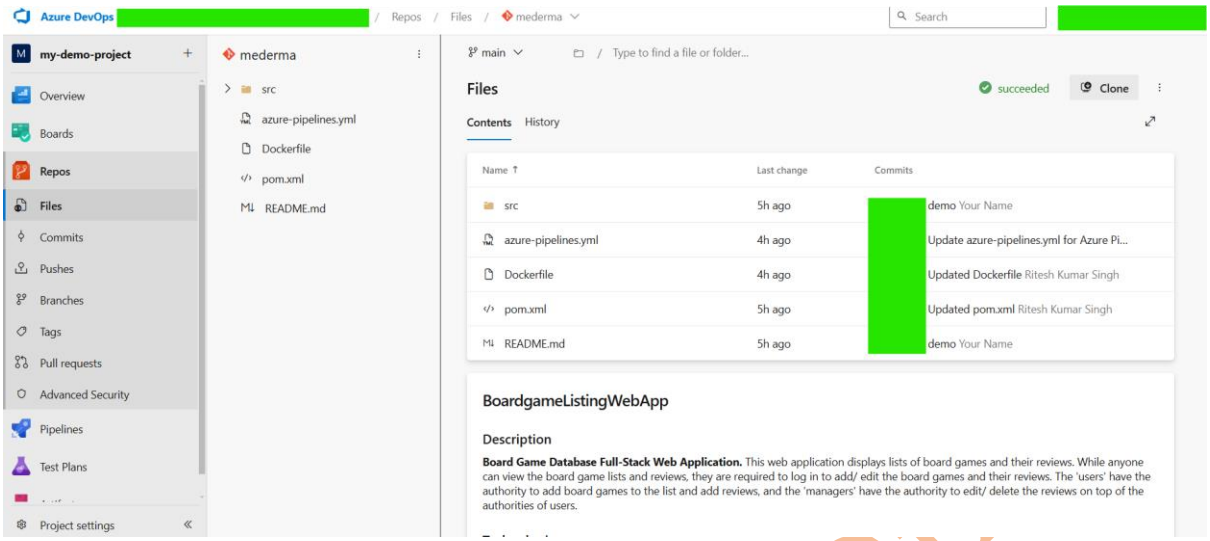
DevOps Project Boardgame Azure

This DevOps project aims to create the infrastructure, 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. Azure DevOps Pipeline was used as CI/CD Tool. For Code Quality check SonarQube, to keep the Artifacts Azure Artifacts Feed and Maven was used as a Build Tool. Trivy was used as a Security Scanning tool, Docker Image was stored in ACR (Azure Container Registries) and for deployment Helm has been used. The high-level architecture diagram for the project is as shown below.



The source code was present in the Azure Repos and Azure DevOps Pipeline 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. Azure Artifacts Feed was used to keep the Artifacts then Docker Image was created which was scanned using Trivy Image Scan as explained earlier. Azure Container Registries (ACR) was used to store the Docker Image. The Deployment had been done with the created Docker Image present in the ACR. The Application Gateway 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 Public IP Address of the Application Gateway Ingress Controller and started accessing the Application using the created URL.

The source code present in the Azure Repos as shown in the screenshot attached below.



I had provided all access to the deployment user demo limited to the namespace boardgame using service account, Role and RoleBinding as shown in the screenshot attached below.

```
[root@devopsagent-vm ~]# cat sa-role-rolebinding.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: demo
  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: demo
```

```
cat sa-role-rolebinding.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: demo
  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: demo
```

```
[root@devopsagent-vm ~]# kubectl apply -f sa-role-rolebinding.yaml
serviceaccount/demo created
role.rbac.authorization.k8s.io/user-role created
rolebinding.rbac.authorization.k8s.io/user-rolebinding created
[root@devopsagent-vm ~]# vim secret.yaml
[root@devopsagent-vm ~]# kubectl apply -f secret.yaml
secret/mysecretname created
[root@devopsagent-vm ~]# kubectl get secrets -n boardgame
NAME                           TYPE                                  DATA   AGE
boardgame-auth                 kubernetes.io/dockerconfigjson        1      [redacted]
ingress-secret                 kubernetes.io/tls                     2
mysecretname                   kubernetes.io/service-account-token   3
sh.helm.release.v1.boardgame.v1   helm.sh/release.v1                 1
sh.helm.release.v1.boardgame.v2   helm.sh/release.v1                 1
[root@devopsagent-vm ~]# kubectl describe secret mysecretname -n boardgame
Name:         mysecretname
Namespace:    boardgame
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: demo
              kubernetes.io/service-account.uid: [redacted]

Type:  kubernetes.io/service-account-token

Data
====
namespace:  9 bytes
token:      [redacted]
ca.crt:     1761 bytes
```

I had shared the kubeconfig file with the deployment user demo as shown in the screenshot attached below.

```
[root@[redacted] ~]# cat ~/.kube/config clear
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: [redacted]
    server: https://boardgame-cluster-dns-onwkr7zc.f[redacted]2.privatelink.eastus2.azmk8s.io:443
  name: boardgame-cluster
contexts:
- context:
    cluster: boardgame-cluster
    user: demo
  name: dexter
current-context: dexter
kind: Config
preferences: {}
users:
- name: demo
  user:
    token: [redacted]
```

apiVersion: v1

clusters:

- cluster:

    certificate-authority-data:
LXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXK

    server: https://boardgame-cluster-dns-
oXXXXXXc.fXXXXXXXXXXXXXXXXXX2.privatelink.eastus2.azmk8s.io:443

  name: boardgame-cluster

contexts:

- context:

    cluster: boardgame-cluster

    user: demo

  name: dexter

current-context: dexter

kind: Config

preferences: {}

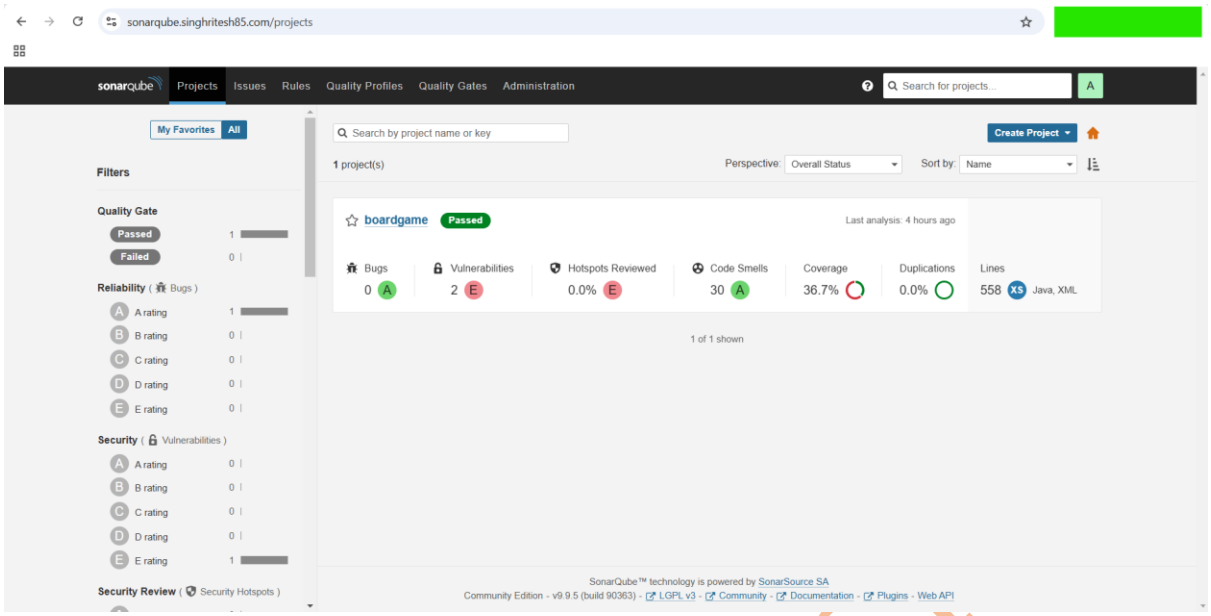
users:

- name: demo

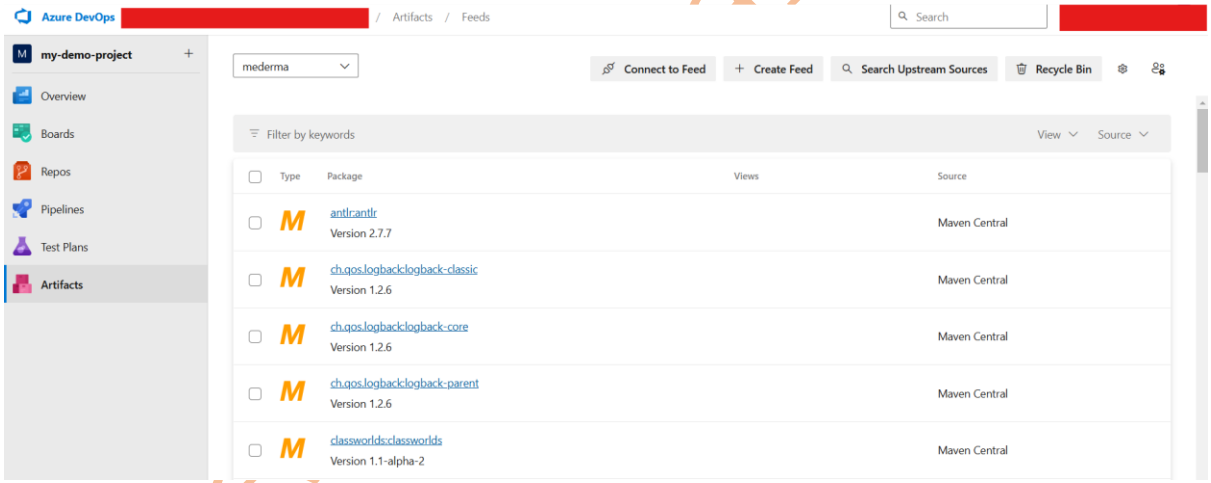
  user:

    token:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

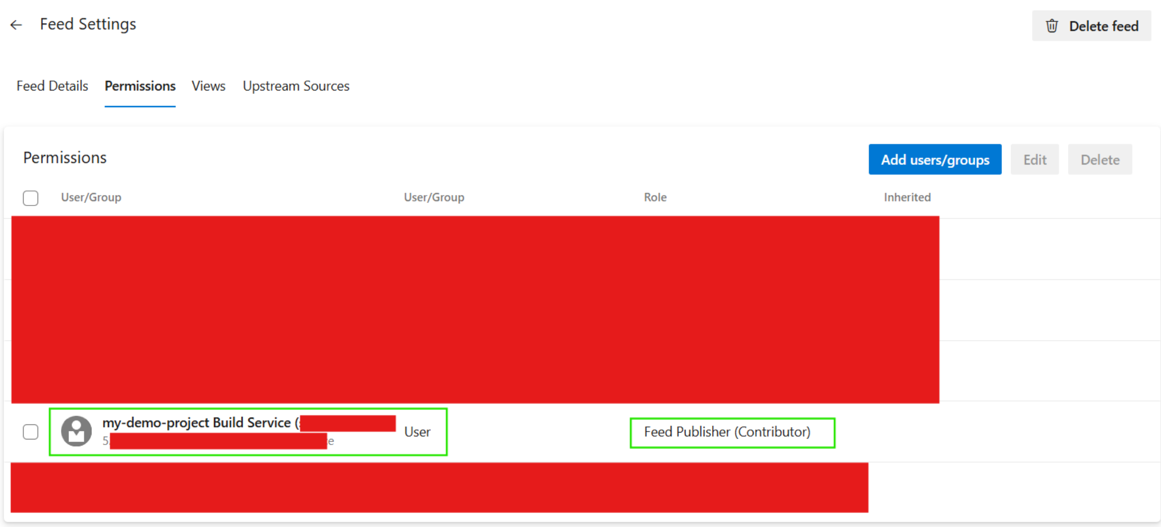
The screenshot for SonarQube after execution of Azure DevOps Pipeline is as shown in the screenshot attached below.



The screenshot for Azure Artifacts Feed after execution of Azure DevOps Pipeline is as shown in the screenshot attached below.



I had provided Contributor Access to the user for the Azure Artifacts Feed as shown in the screenshot attached below.



Installation of node-exporter in AKS Cluster had been done using the helm chart with the commands as shown in the screenshot attached below.

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

As explained above I had installed Node-Exporter in AKS Cluster using the helm chart, node-exporter pods were created as a part of daemonset. Whenever a new node will be created in this AKS Cluster then a pod of node-exporter will also be created on the newly created node as a part of daemonset.

Create kubernetes secret for Azure Container Registries (ACR) using the command as shown below

kubectl create ns boardgame && kubectl create secret docker-registry boardgame-auth --docker-server=https://boardgamecontainer24registry.azurecr.io --docker-username=boardgamecontainer24registry --docker-password=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX8x -n boardgame

Create kubernetes secret using certificate as shown in the screenshot attached below

kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n boardgame

```
[root@[redacted] ~]# kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n boardgame
```

The ingress rule to create the kubernetes ingress in AKS Cluster is as shown in the screenshot attached below.

```
[demo@devopsagent-vm ~]$ cat ingress-rule.yaml
# kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n boardgame
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: boardgame-ingress
  namespace: boardgame
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
  - secretName: ingress-secret
  rules:
  - host: boardgame.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: boardgame-folo
            port:
              number: 80
```

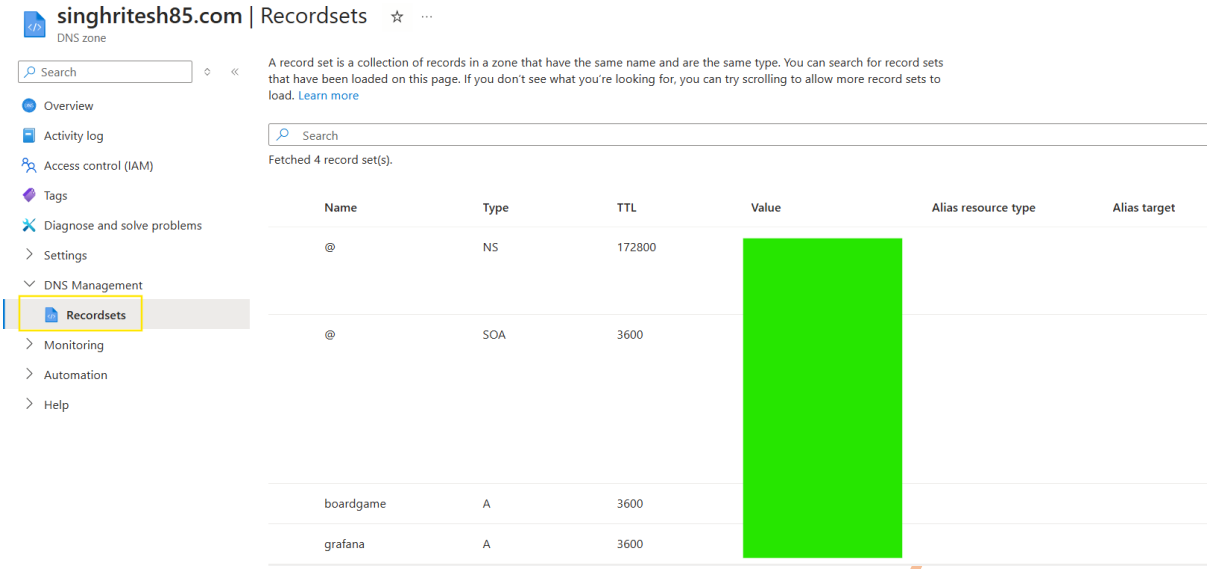
The kubernetes ingress in AKS Cluster is as shown in the screenshot attached below.

```
[demo@devopsagent-vm ~]$ kubectl get ing -n boardgame
NAME                CLASS                       HOSTS                        ADDRESS          PORTS     AGE
boardgame-ingress   azure-application-gateway   boardgame.singhritesh85.com  135.[redacted].241   80, 443   3h50m
```

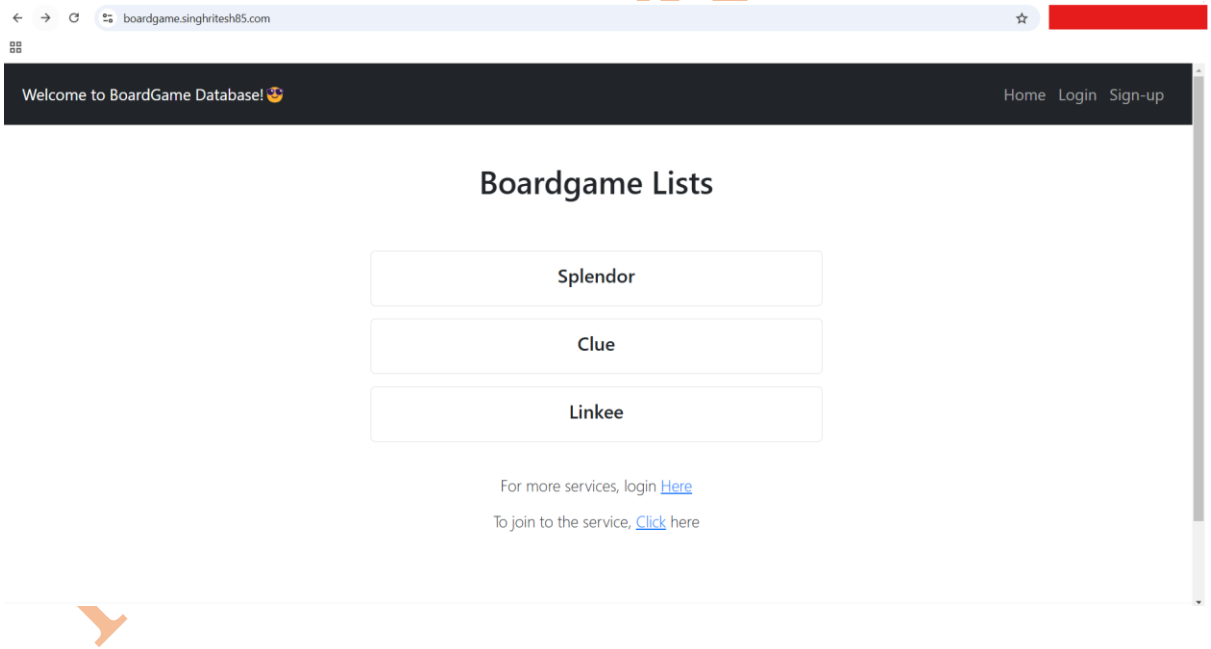
cat ingress-rule.yaml

```
# kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n boardgame
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: boardgame-ingress
  namespace: boardgame
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
  - secretName: ingress-secret
  rules:
  - host: boardgame.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: boardgame-folo
            port:
              number: 80
```

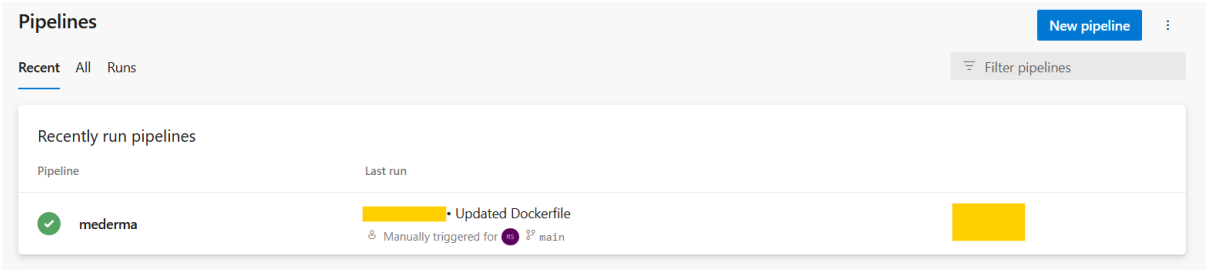
I created the record set in Azure DNS Zone for boardgame application using Public IP Address of the Application Gateway Ingress Controller as shown in the screenshot attached below.



Finally, I accessed the Application using the newly created URL as shown in the screenshot attached below.



The screenshot for Azure DevOps Pipeline after successful execution is as shown below.



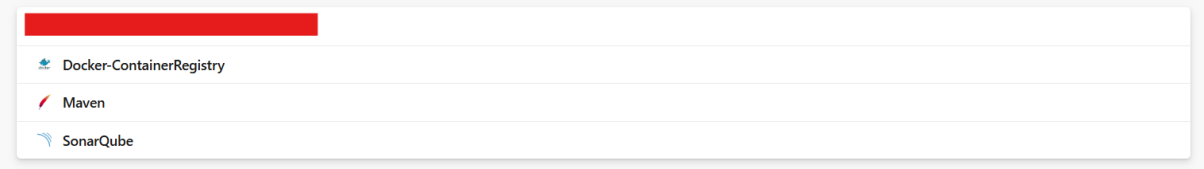
Kubernetes Pods, Service and Deployments for the Application had been created after successfully running the Azure DevOps Pipeline as shown in the screenshot attached below.

```
[demo@devopsagent-vm ~]$ kubectl get all -n boardgame
NAME                                    READY   STATUS    RESTARTS   AGE
pod/boardgame-folo-5[redacted]8   1/1     Running   0          92m
pod/boardgame-folo-5[redacted]x   1/1     Running   0          92m

NAME                       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
service/boardgame-folo     ClusterIP   10.[redacted].51   <none>        80/TCP    3h30m

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/boardgame-folo      2/2     2            2           3h30m

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/boardgame-folo-5[redacted]f   2         2         2       92m
```

The deployment user demo did not have overall access for the entire AKS cluster but had all the access in the namespace boardgame as shown in the screenshot attached below.

```
[demo@devopsagent-vm ~]$ kubectl get all -n boardgame
NAME                                    READY   STATUS    RESTARTS   AGE
pod/boardgame-folo-5[redacted]8   1/1     Running   0          151m
pod/boardgame-folo-5[redacted]x   1/1     Running   0          151m

NAME                       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
service/boardgame-folo     ClusterIP   10.[redacted].51   <none>        80/TCP    4h29m

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/boardgame-folo      2/2     2            2           4h29m

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/boardgame-folo-5[redacted]f   2         2         2       151m
[demo@devopsagent-vm ~]$ kubectl get nodes
Error from server (Forbidden): nodes is forbidden: User "system:serviceaccount:boardgame:demo" cannot list resource "nodes" in API group "" at the cluster scope
```

Service Connections in Azure DevOps had been created for Azure Container Registries (ACR), Azure Artifacts Feed and SonarQube 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 Azure DevOps Agent, SonarQube-Server, Prometheus-Server, Blackbox-Exporter Server, Grafana-Server, and AKS 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 http_2xx_example present in prometheus configuration file prometheus.yml must match with the module name of blackbox exporter configuration file** (monitor_website.yml) present on the blackbox exporter server at the path (/opt/blackbox_exporter_linux_amd64/monitor_website.yml).

Prometheus blackbox exporter 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.

For Prometheus the configuration file is as shown below.

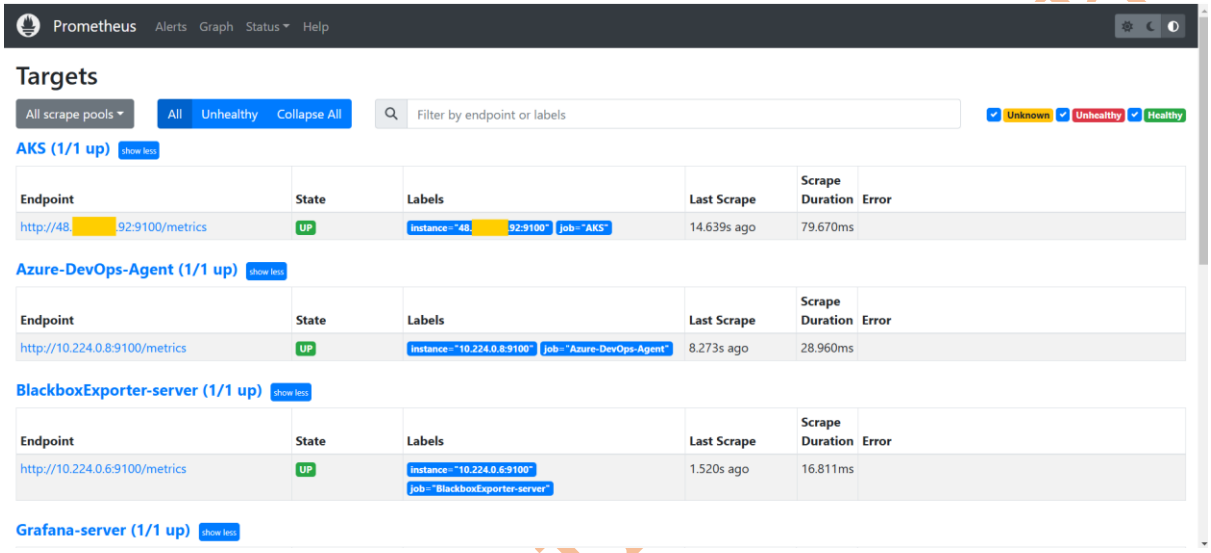
```
cat /etc/prometheus/prometheus.yml
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (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: ["10.224.0.4:9100"]
- job_name: "BlackboxExporter-server"
  static_configs:
    - targets: ["10.224.0.6:9100"]
- job_name: "Azure-DevOps-Agent"
  static_configs:
    - targets: ["10.224.0.8:9100"]
- job_name: "SonarQube-Server"
  static_configs:
    - targets: ["10.224.0.5:9100"]
- job_name: "AKS"
  static_configs:
    - targets: ["48.211.137.92:9100"]
- 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: 10.224.0.6:9115  # The blackbox exporter's real hostname:port.
```

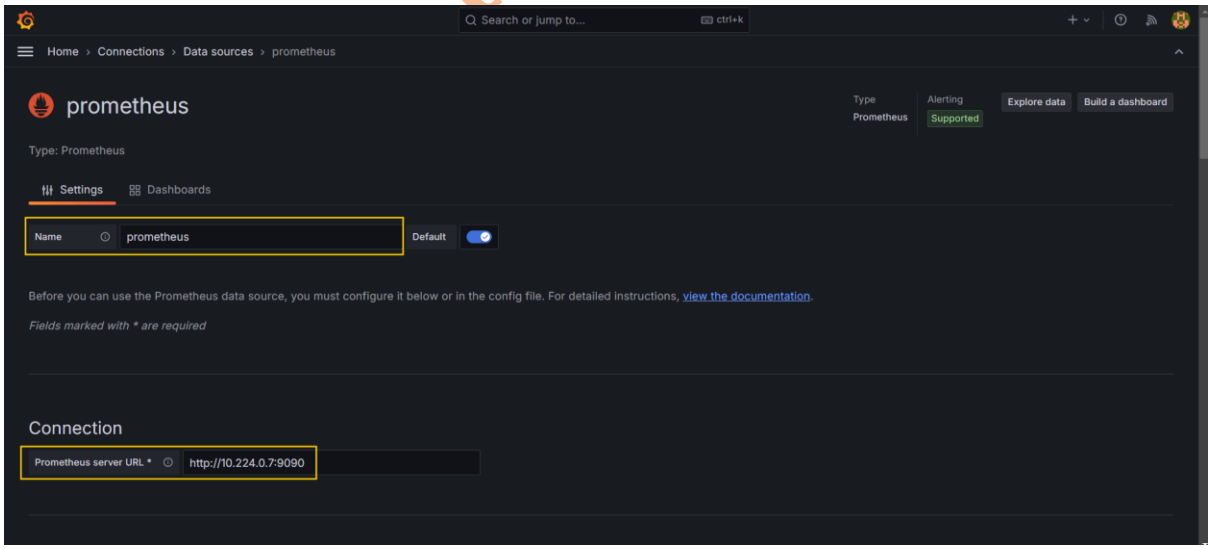
After change in configuration file for prometheus restarted and checked the prometheus service status as shown in the screenshot attached below.

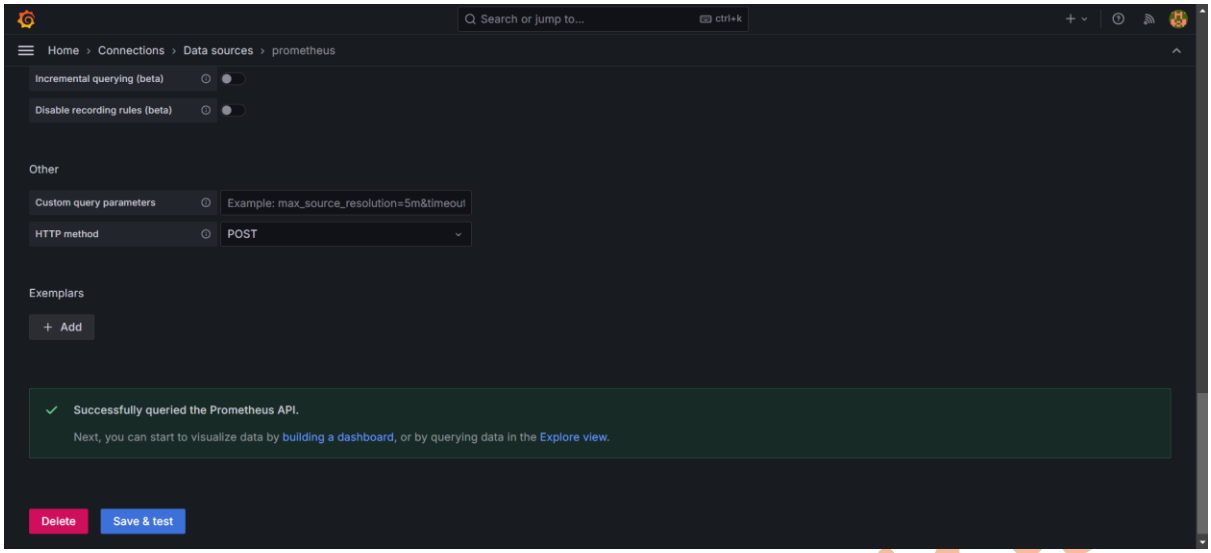
```
[root@prometheus-vm ~]# systemctl restart prometheus.service
[root@prometheus-vm ~]# systemctl status prometheus.service
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2024-11-04 13:14:41 UTC; 21s ago
 Main PID: 6932 (prometheus)
```

All the targets are up as can be seen from the prometheus dashboard, screenshot attached below.



Prometheus acted as the DataSource for Grafana and tested its connection, which was successful.





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

Checked the service for prometheus blackbox exporter. It was in Running state as shown in the screenshot attached below.

```
[root@blackboxexporter-vm ~]# systemctl status blackbox_exporter.service
● blackbox_exporter.service - Blackbox Exporter
   Loaded: loaded (/etc/systemd/system/blackbox_exporter.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2024-11-04 13:18:03 UTC; 1h 30min ago
```

I disabled the certificate validation in the configuration of blackbox exporter as shown in the screenshot attached below.

tls_config:

insecure_skip_verify: true

```
[root@blackboxexporter-vm ~]# cat /opt/blackbox_exporter_linux_amd64/monitor_website.yml
modules:
  http_2xx_example:
    prober: http
    timeout: 5s
    http:
      valid_http_versions: ["HTTP/1.1", "HTTP/2.0"]
      valid_status_codes: []  # Defaults to 2xx
      method: GET
      tls_config:
        insecure_skip_verify: true
```

cat /opt/blackbox_exporter_linux_amd64/monitor_website.yml

modules:

http_2xx_example:

prober: http

timeout: 5s

http:

valid_http_versions: ["HTTP/1.1", "HTTP/2.0"]

valid_status_codes: [] # Defaults to 2xx

method: GET

tls_config:

insecure_skip_verify: true

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

Configuration of Alerts in Grafana

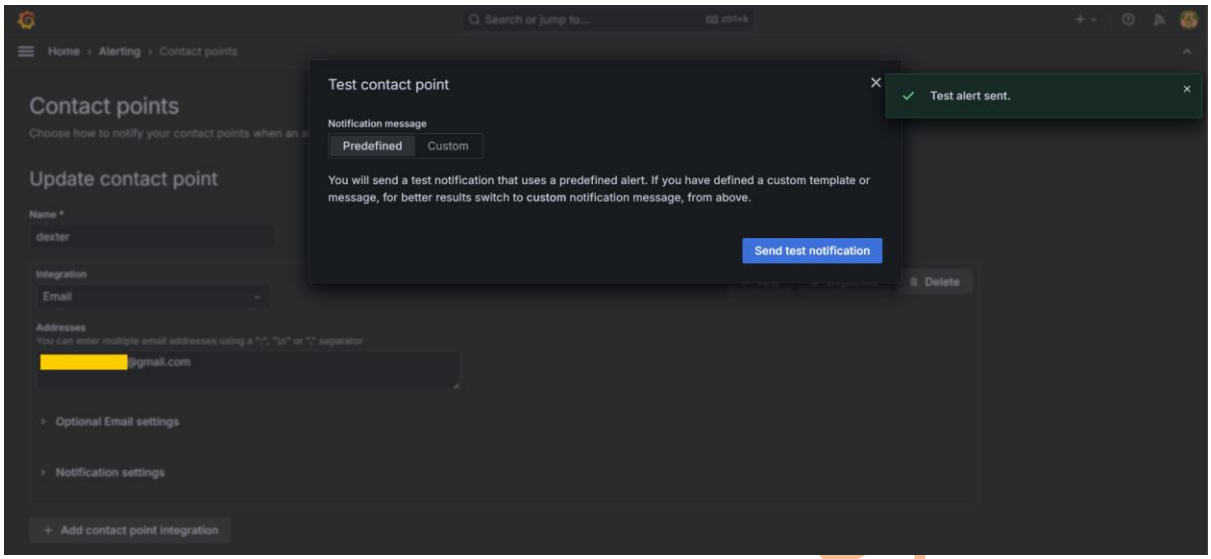
To configure Email Alerts in Grafana I changed the configuration for grafana as shown in the screenshot attached below and then restarted the service for grafana and checked its status as shown in the screenshot attached below.

```
[root@grafana-vm ~]# vim /etc/grafana/grafana.ini
```

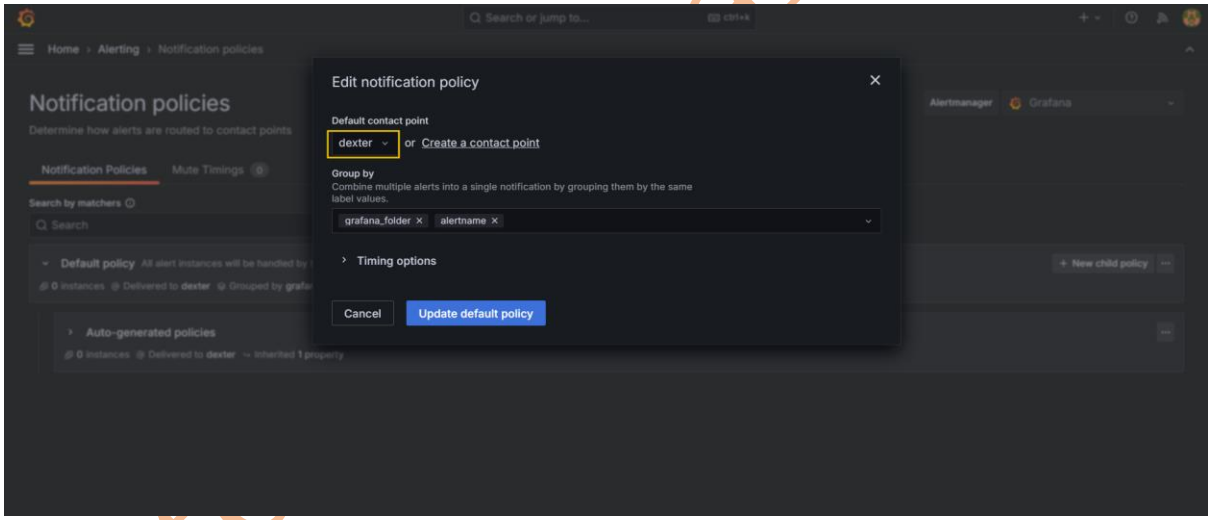
```
# Specifies whether Entra password auth can be used for the MSSQL data source
# Disabled by default, needs to be explicitly enabled
;azure_entra_password_credentials_enabled = false

#################################### Role-based Access Control ###########
[rbac]
;permission_cache = true

# Reset basic roles permissions on boot
# Warning left to true, basic roles permissions will be reset on every boot
#reset_basic_roles = false

# Validate permissions' action and scope on role creation and update
; permission_validation_enabled = true

#################################### 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
```

```
[root@grafana-vm ~]# systemctl restart grafana-server.service
[root@grafana-vm ~]# systemctl status grafana-server.service
● grafana-server.service - Grafana instance
   Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2024-11-04 15:40:09 UTC; 1min 5s ago
```

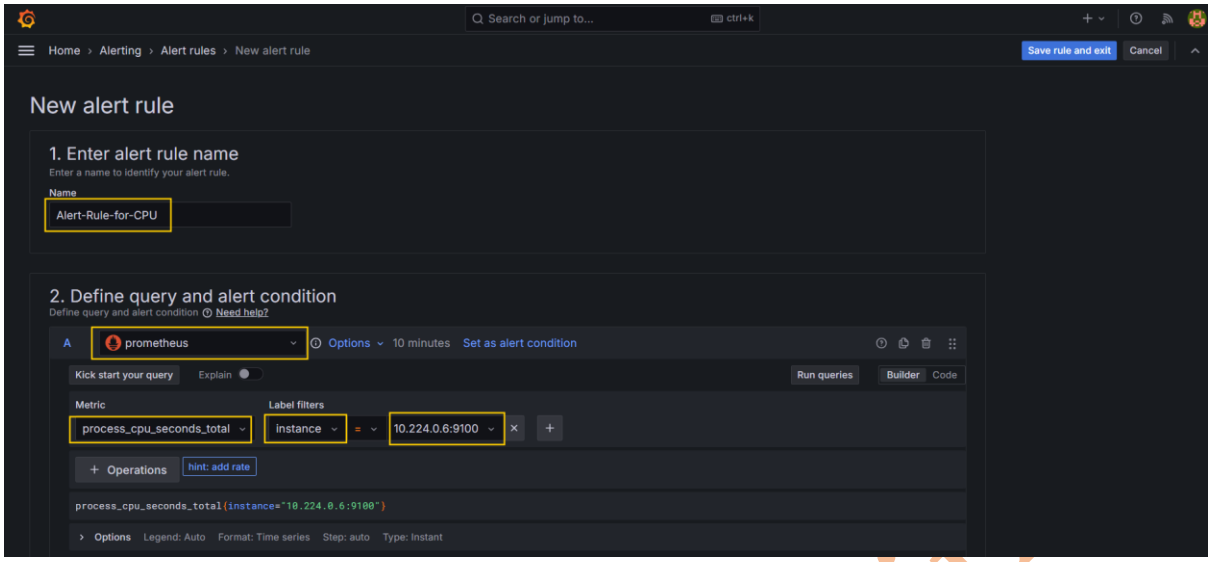
The **contact points** in Grafana Alerts had been created and tested as shown in the screenshot attached below.

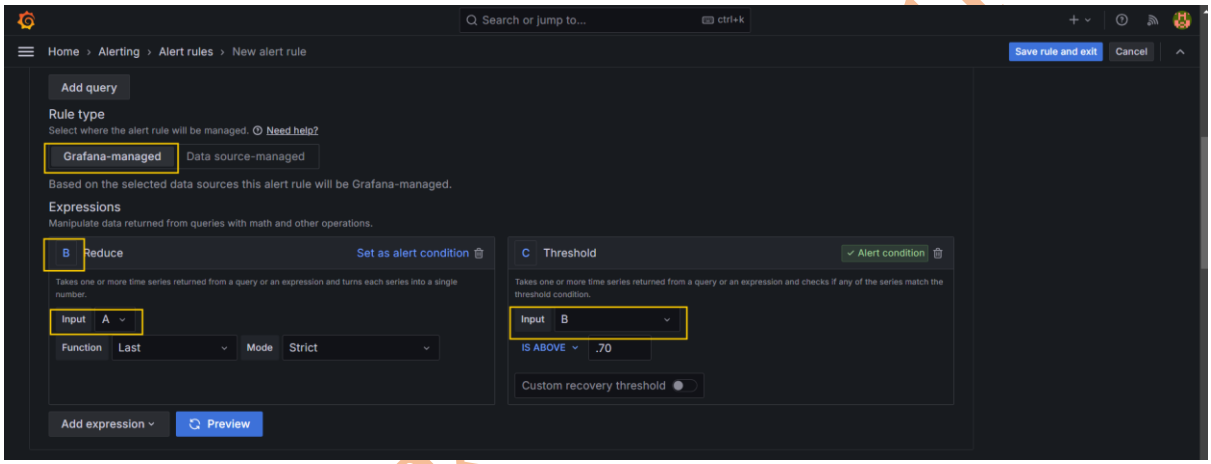


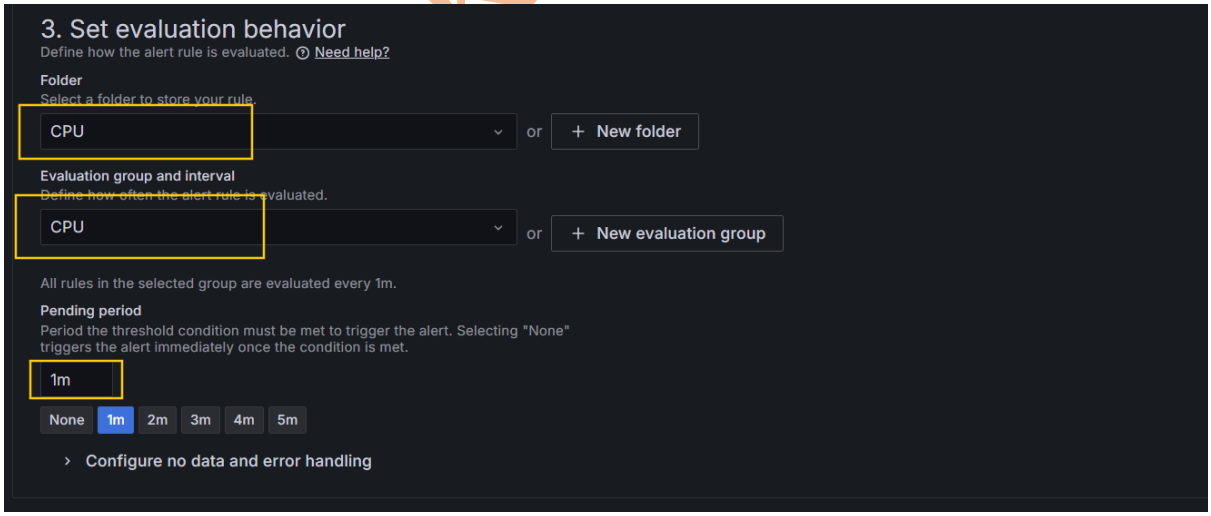
The default policy for the **Notification Policy** is as shown in the screenshot attached below.

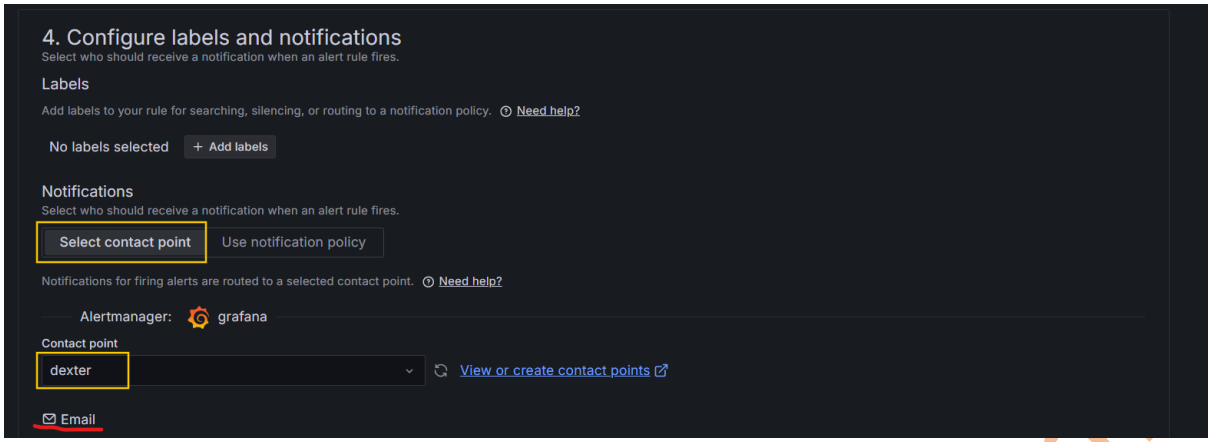


Finally create the **Alert Rule** in Grafana Alerts as shown in the screenshot attached below.

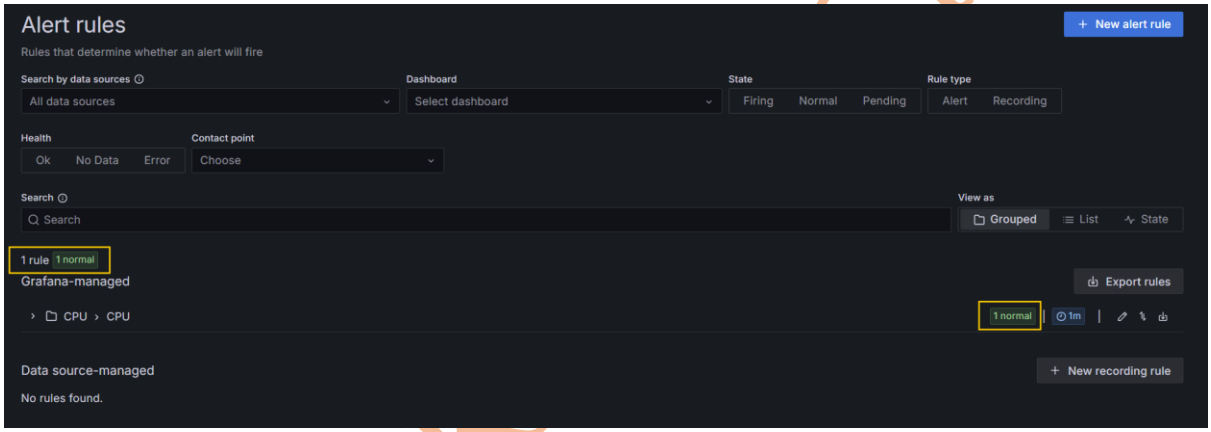

Search or jump to... ctrl+k
Home › Alerting › Alert rules › New alert rule
Save rule and exit
Cancel
New alert rule
1. Enter alert rule name
Enter a name to identify your alert rule.
Name
Alert-Rule-for-CPU
2. Define query and alert condition
Define query and alert condition Need help?
A
prometheus
Options 10 minutes
Set as alert condition
Kick start your query
Explain
Run queries
Builder
Code
Metric
Label filters
process_cpu_seconds_total
instance
=
10.224.0.6:9100
Operations
hint: add rate
process_cpu_seconds_total(instance="10.224.0.6:9100")
Options
Legend: Auto
Format: Time series
Step: auto
Type: Instant


Search or jump to... ctrl+k
Home › Alerting › Alert rules › New alert rule
Save rule and exit
Cancel
Add query
Rule type
Select where the alert rule will be managed. Need help?
Grafana-managed
Data source-managed
Based on the selected data sources this alert rule will be Grafana-managed.
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
A
Function
Last
Mode
Strict
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
B
IS ABOVE
.70
Custom recovery threshold
Add expression
Preview


3. Set evaluation behavior
Define how the alert rule is evaluated. Need help?
Folder
Select a folder to store your rule.
CPU
or
New folder
Evaluation group and interval
Define how often the alert rule is evaluated.
CPU
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.
1m
None
1m
2m
3m
4m
5m
Configure no data and error handling

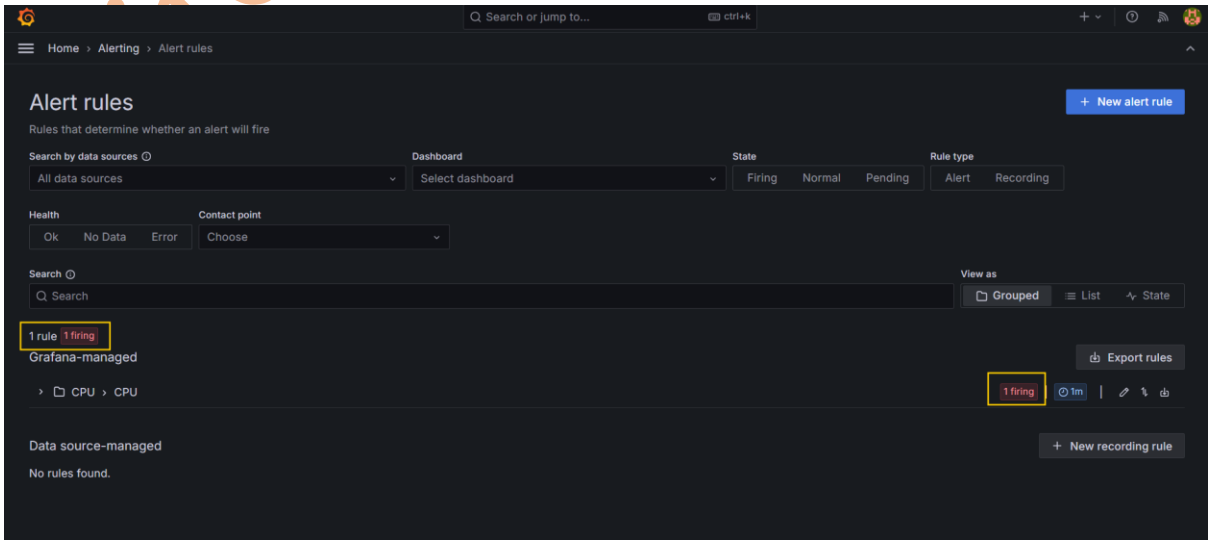


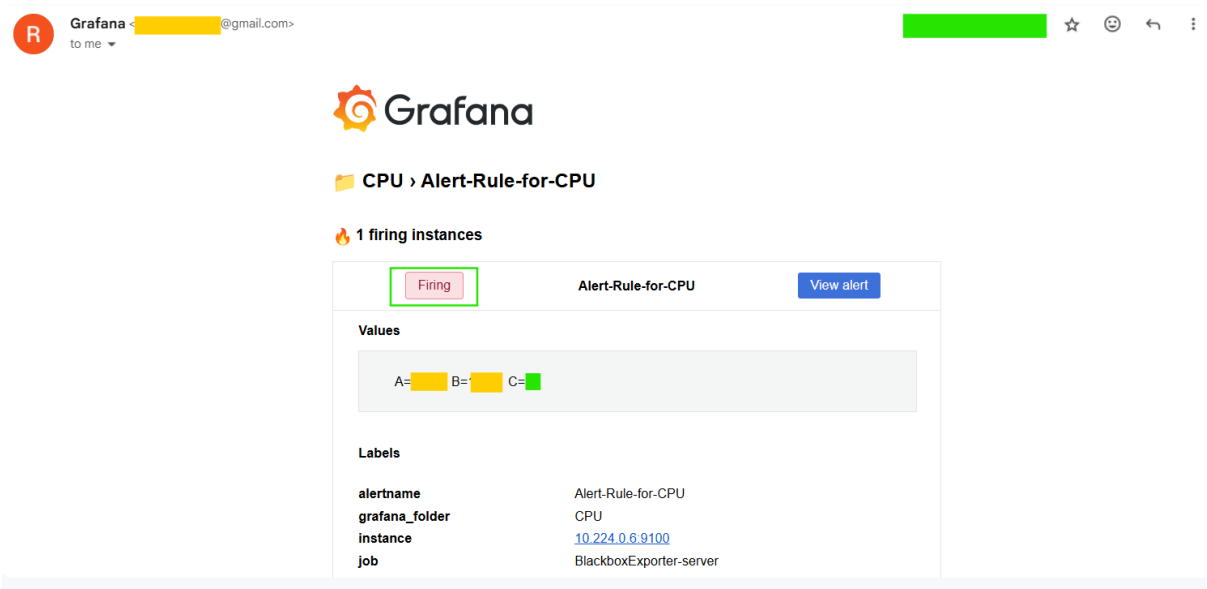
After creation of Alert Rule we can this Alert Rule in Normal Condition as shown in the screenshot attached below.



Whenever the Alert Rule crosses the Threshold, it will Trigger the Grafana Alert and will send the Email on Group Email Id of your team, so that the team will get notified on this issue.

After some times it crosses the threshold and this Alert will come into the Firing Condition as shown in the screenshot attached below and an email will be sent to the Group Email Id.





After the team received this Email on their Email Id, they will investigate its RCA (Root Cause Analysis) and will check the CPU usage using the command **htop**. Team will check whether any unnecessary crontab, any process is running which will utilize more CPU. If yes then edit the crontab and comment out that entry for crontab or kill that process. If necessary, team will check the Log file to investigate this issue or if needed upgrade the VM Size of Azure VM.