DevOps Project Netflix AWSandAzure

By Ritesh Kumar Singh

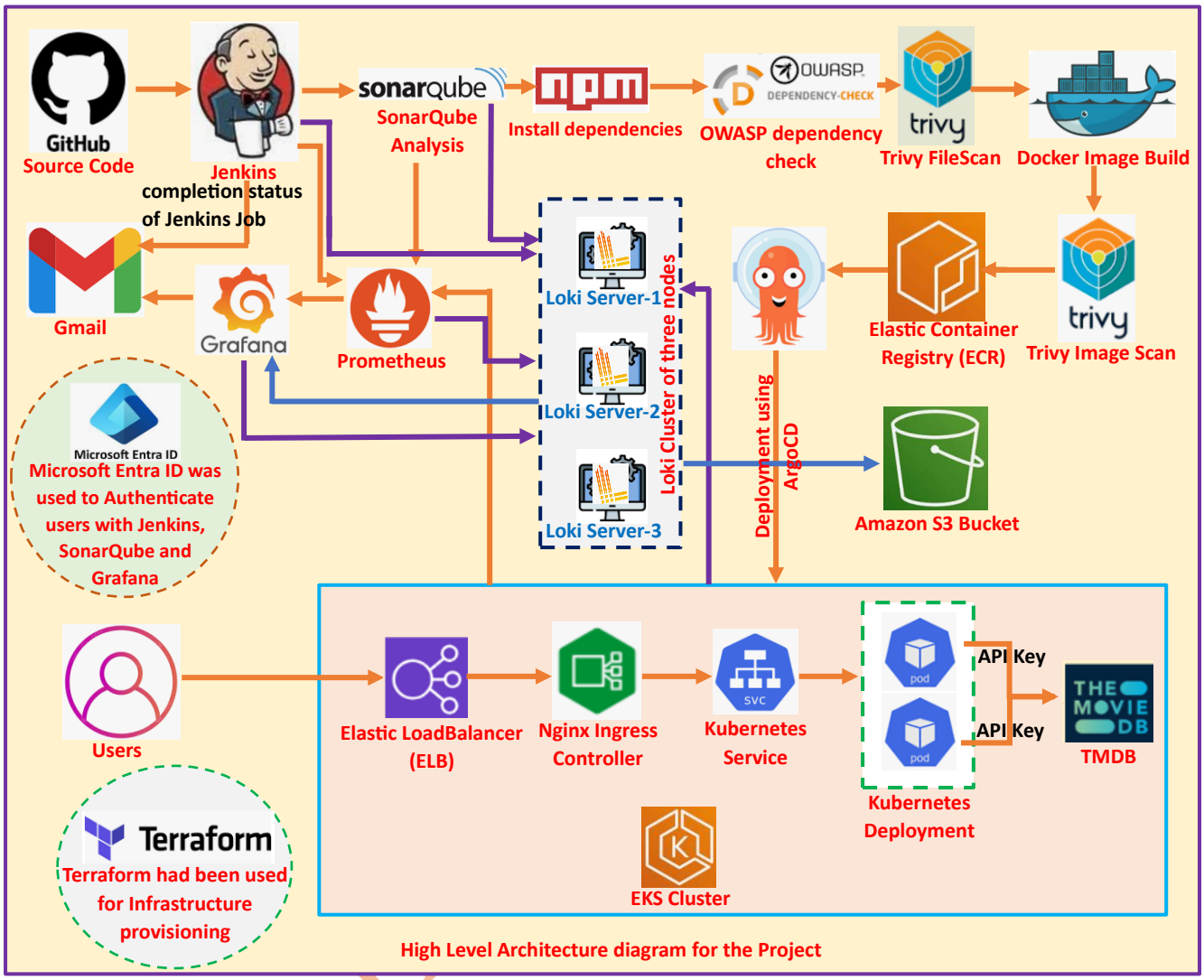
Email Address: - riteshkumarsingh9559@gmail.com

LinkedIn: - https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/

GitHub: - https://github.com/singhritesh85

या कुन्देन्दुतुषारहारधवला या शुभ्रवस्त्रावृता
या वीणावरदण्डमण्डितकरा या श्वेतपद्मासना।
या ब्रह्माच्युत शंकरप्रभृतिभिर्देवैः सदा वन्दिता
सा मां पातु सरस्वती भगवती निःशेषजाड्यापहा ॥

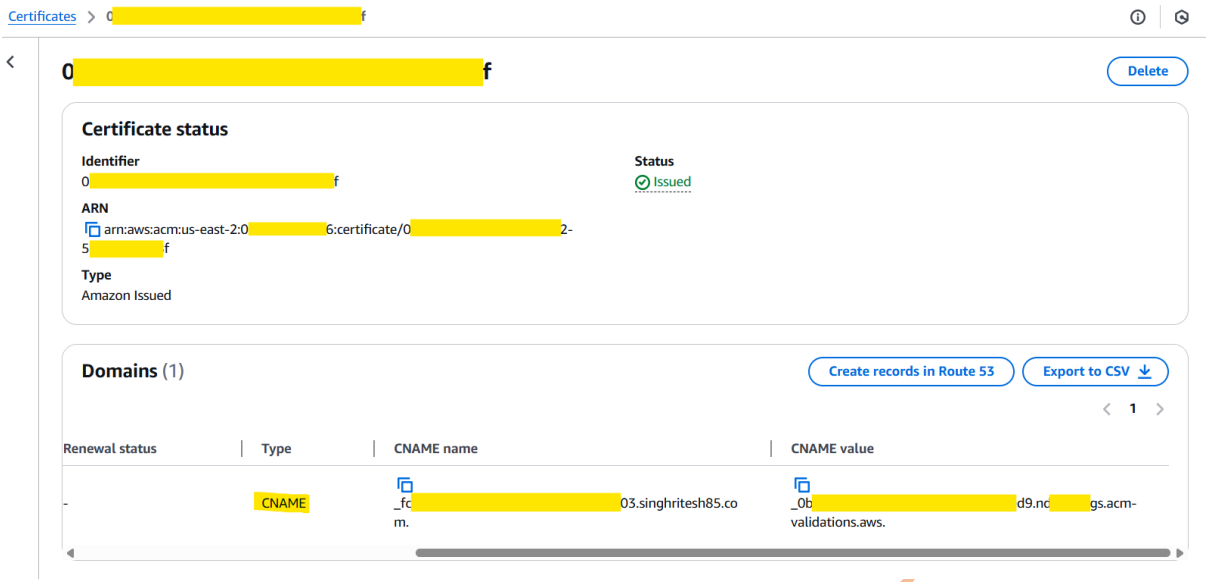
DevOps Project Netflix AWSandAzure

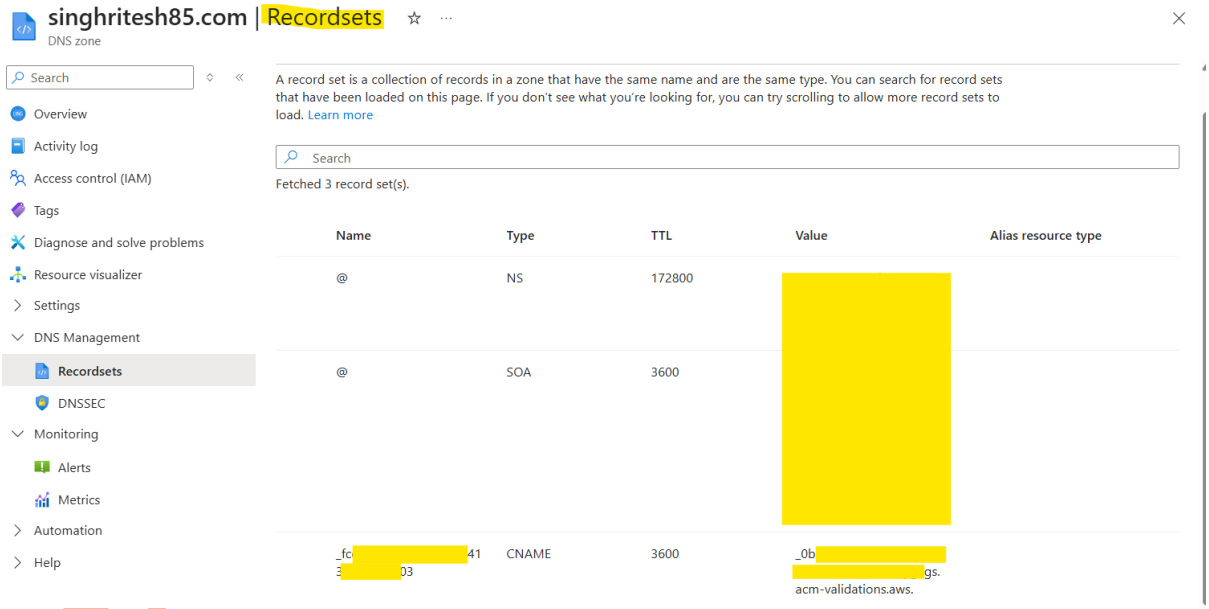


High Level Architecture diagram for the Project

This DevOps Project deals with creation of Infrastructure using Terraform and setup of CICD Pipeline using Jenkins, Monitoring using Prometheus and Grafana and Log Aggregation using Loki, Promtail and Grafana. SonarQube was used for Code-Analysis and NPM was used as the Build Tool as shown in the Architecture diagram above. OWASP dependency check had been performed to scan publicly known vulnerability and Trivy was used for FileScan and Docker Image Scan to identify and address known vulnerability hence enhancing the overall security. The Docker Image was kept in the Elastic Container Registry (ECR) and which was deployed to EKS Cluster using the ArgoCD as shown in the high-level architecture diagram above. User was able to access the Application through the Ingress and hence the Kubernetes Service. For this project the source code was present in the GitHub Repository https://github.com/singhritesh85/DevSecOps-Project.git. The Kubernetes Pod will connect with TMDB (The Movie Database) using the API Key (the API Key will be used for Authentication). The successful completion or failure of Jenkins Job will be notified through the notification on Group Email ID. To Authenticate users in Jenkins, SonarQube and Grafana I used Azure Entra ID (formerly known as Azure Active Directory). Grafana will send notification to Group Email ID if an Alert encountered.

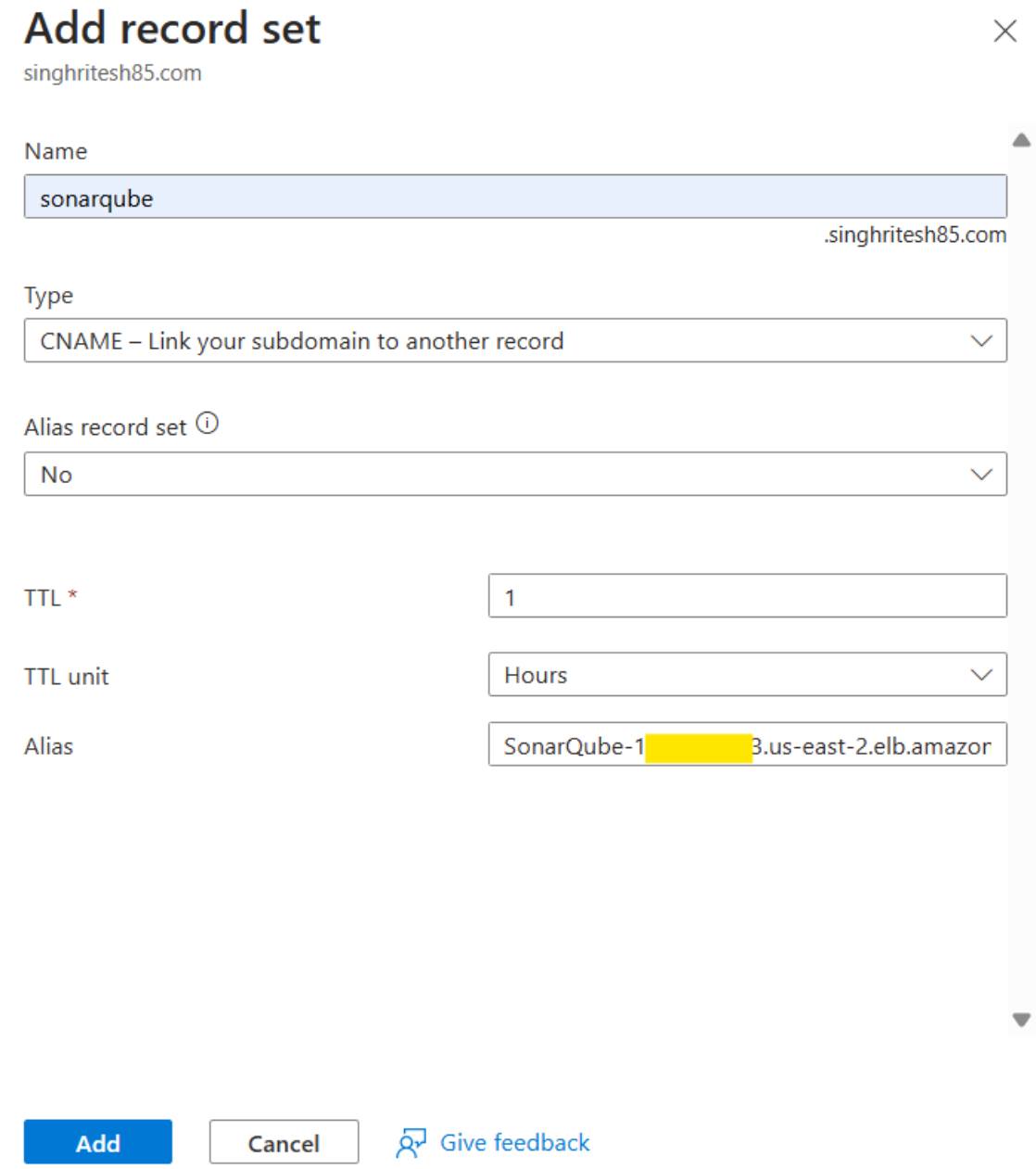
To validate the SSL Certificate generated from certificate manager I used DNS validation and did the entry for Azure DNS Zone record set of type CNAME as shown in the screenshot attached below.



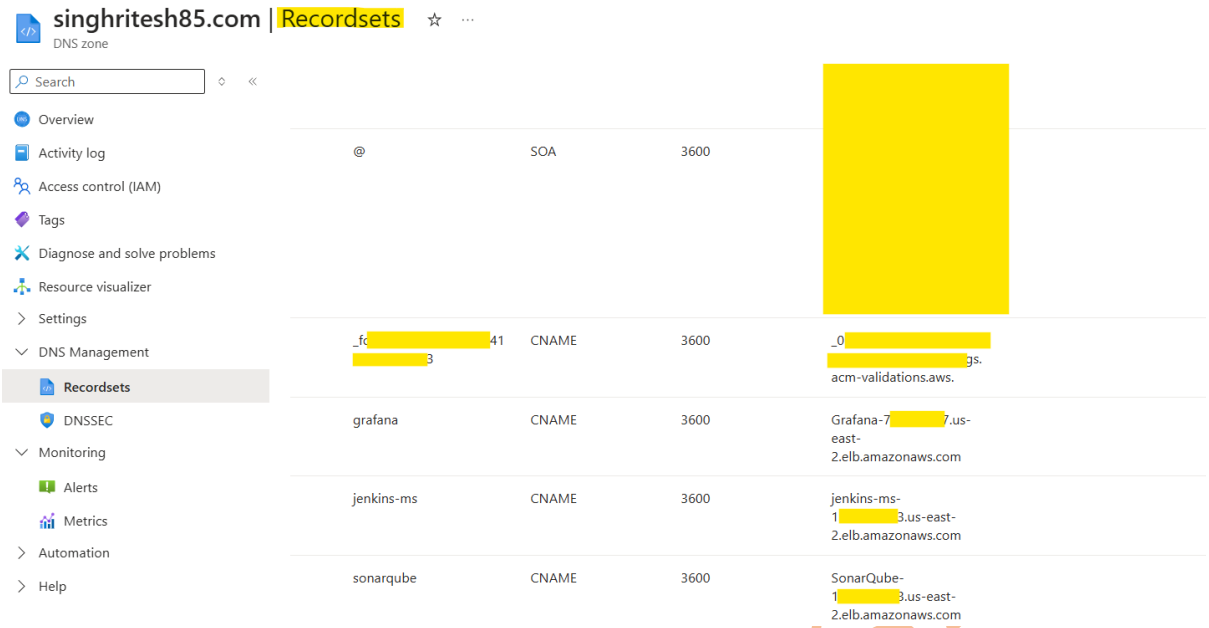


Then I wait for around 20 seconds and after that SSL Certificate was issued as shown in the first screenshot attached above.

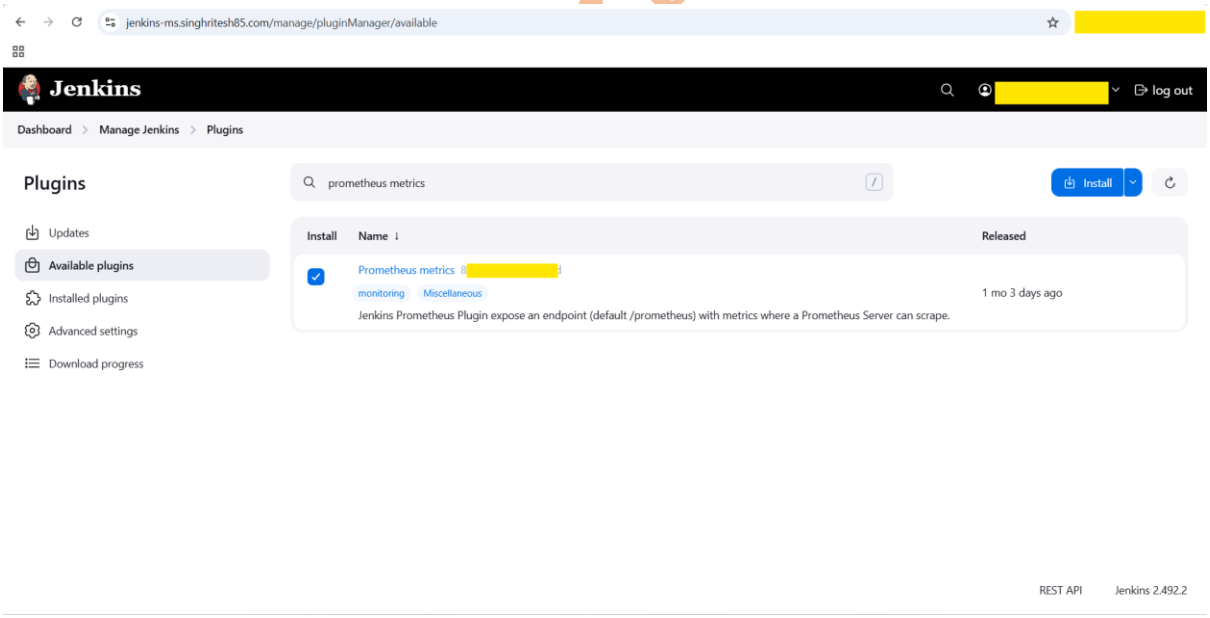
After running the terraform script LoadBalancers for SonarQube, Jenkins and Grafana had been created as shown in the screenshot attached below. I created the record set of type CNAME in Azure DNS Zone using the DNS Name of the LoadBalancers for example I had shown below the creation of record set for SonarQube.

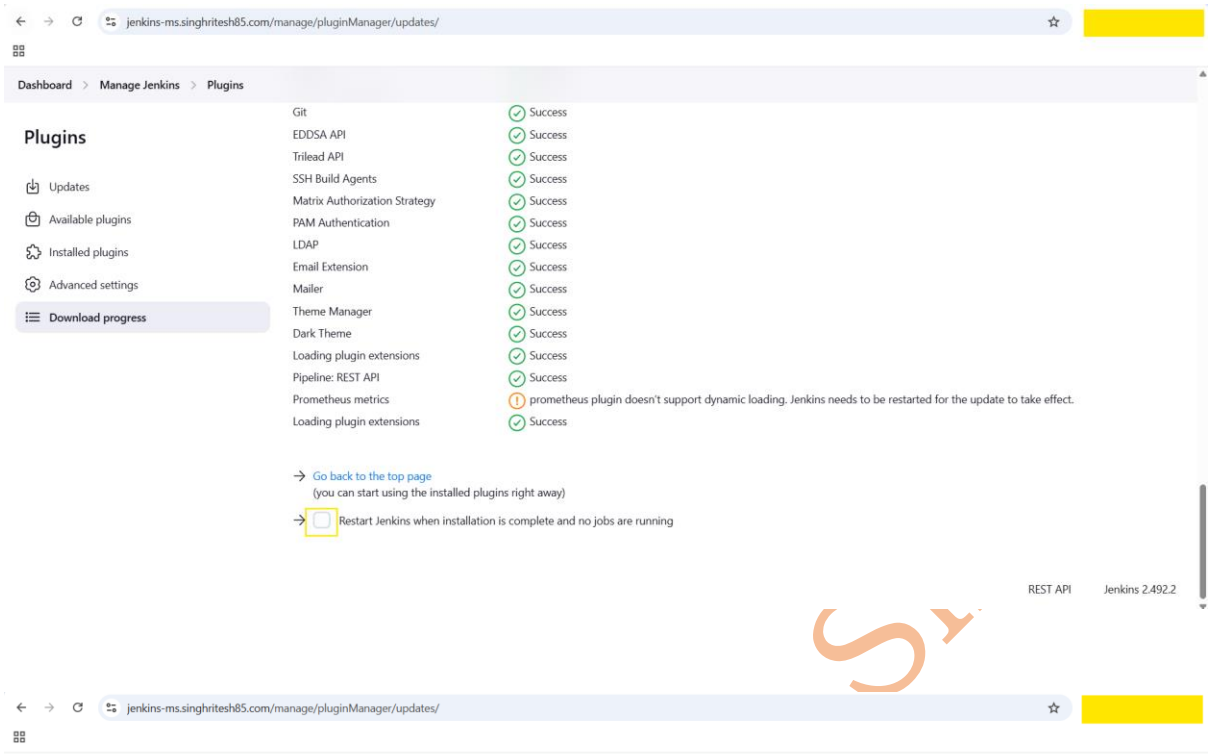


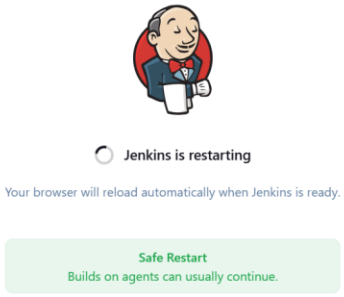
After doing the entry for all the DNS Names (of all the LoadBalancers) as shown in the screenshot attached below to create the record set. The final screenshot is as shown in the screenshot attached below.



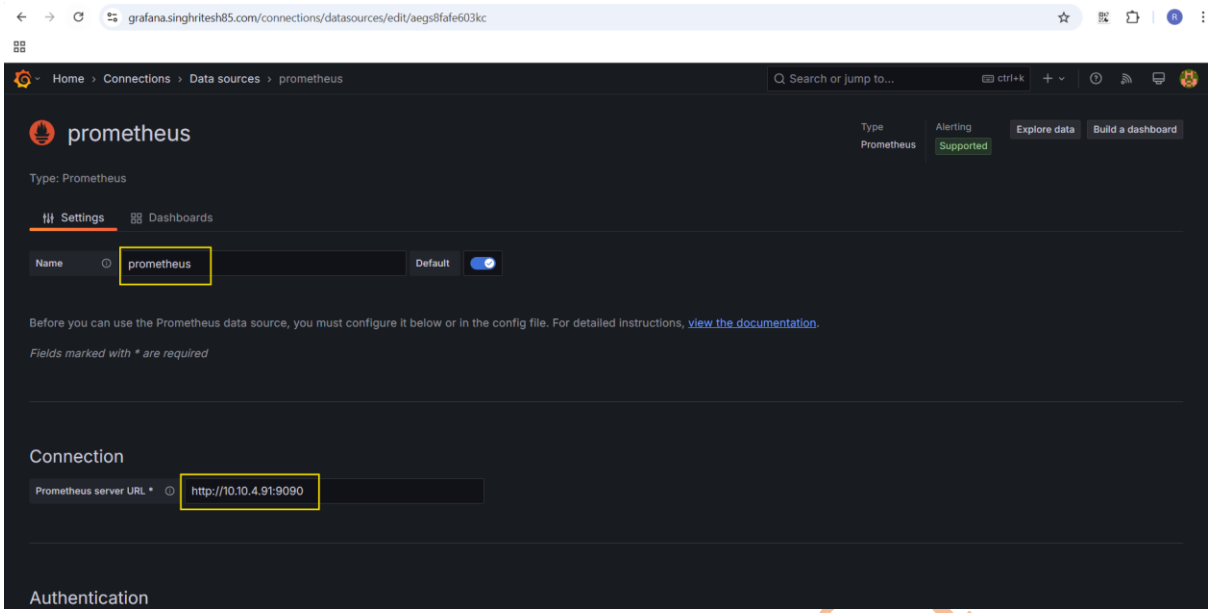
It is possible to monitor Jenkins Job using Prometheus and Grafana and for that I installed **Prometheus Metrics** plugin in Jenkins, after its installation I restarted the Jenkins as shown in the screenshot attached below.

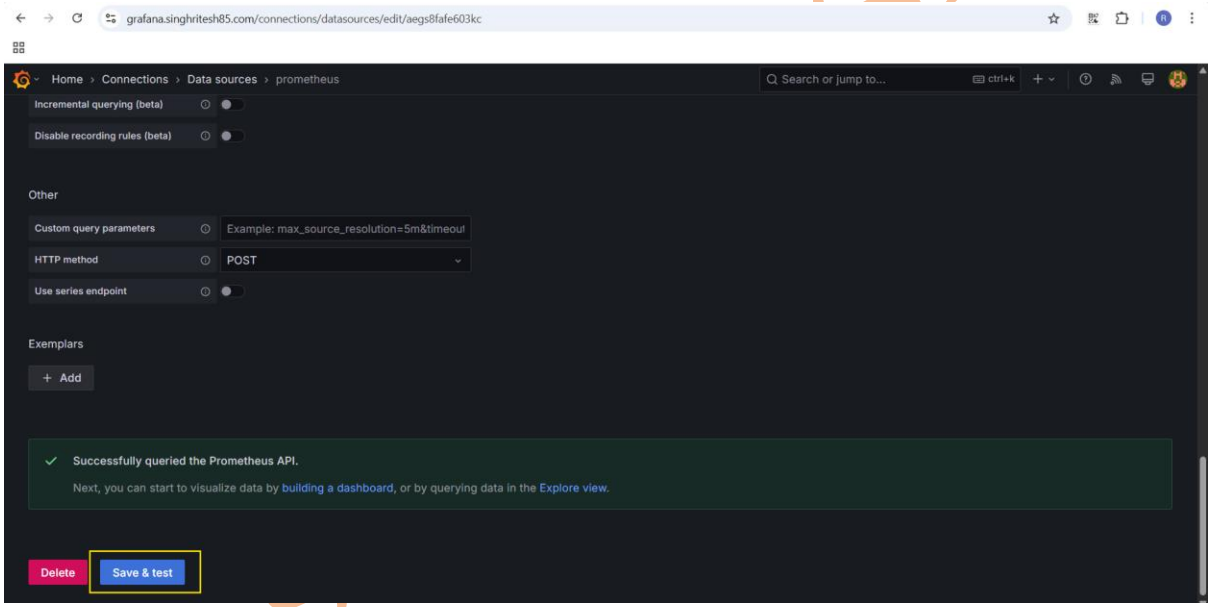


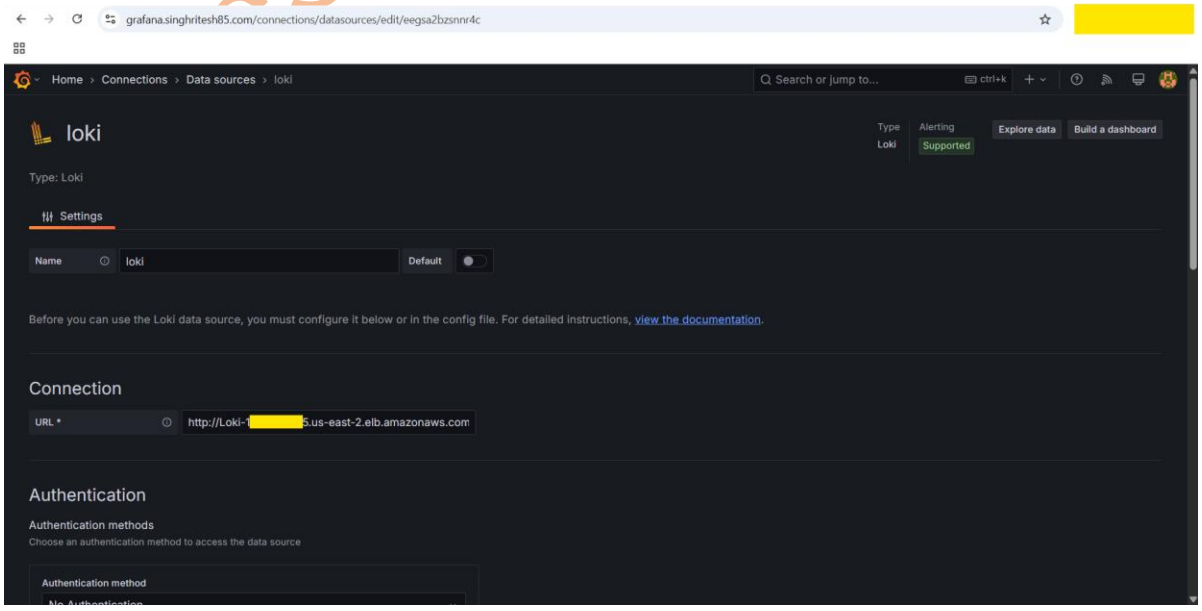


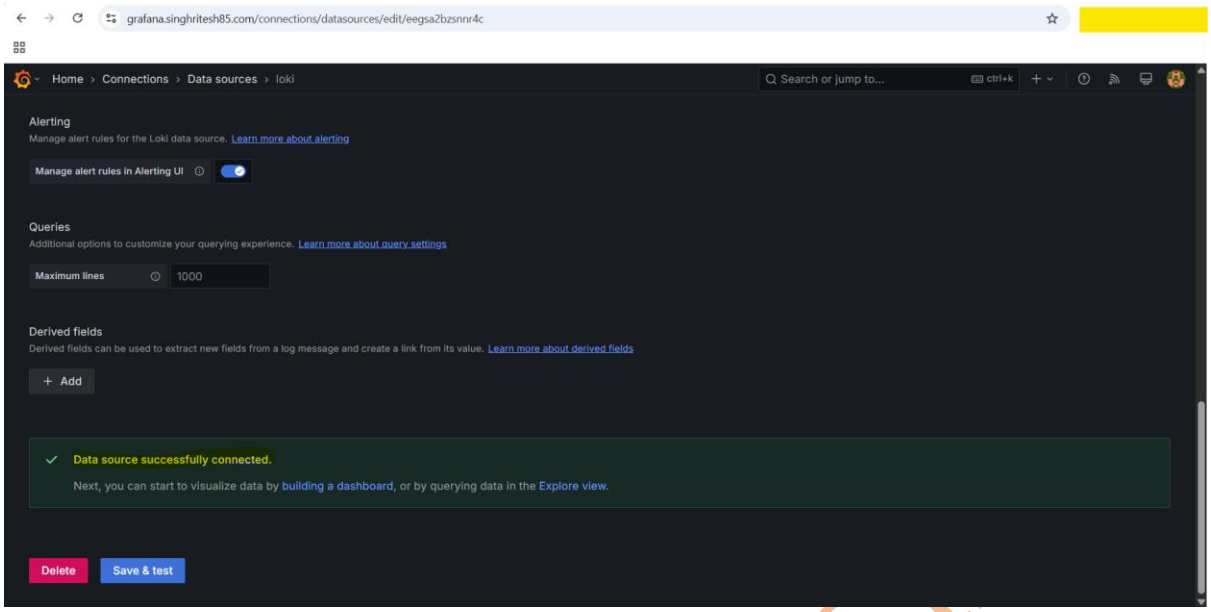


In Grafana I created two data sources each for prometheus and loki as shown in the screenshot attached below.


grafana.singhritesh85.com/connections/datasources/edit/aegs8fafe603kc
Home › Connections › Data sources › prometheus
Search or jump to... ctrl+k
prometheus
Type: Prometheus
Type
Prometheus
Alerting
Supported
Explore data
Build a dashboard
Settings
Dashboards
Name
prometheus
Default
Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, view the documentation.
Fields marked with * are required
Connection
Prometheus server URL *
http://10.10.4.91:9090
Authentication


grafana.singhritesh85.com/connections/datasources/edit/aegs8fafe603kc
Home › Connections › Data sources › prometheus
Search or jump to... ctrl+k
Incremental querying (beta)
Disable recording rules (beta)
Other
Custom query parameters
Example: max_source_resolution=5m&timeout
HTTP method
POST
Use series endpoint
Exemplars
Add
Successfully queried the Prometheus API.
Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view.
Delete
Save & test


grafana.singhritesh85.com/connections/datasources/edit/eegsa2bzsnnr4c
Home › Connections › Data sources › loki
Search or jump to... ctrl+k
loki
Type: Loki
Type
Loki
Alerting
Supported
Explore data
Build a dashboard
Settings
Name
loki
Default
Before you can use the Loki data source, you must configure it below or in the config file. For detailed instructions, view the documentation.
Connection
URL *
http://Loki-1
5.us-east-2.elb.amazonaws.com
Authentication
Authentication methods
Choose an authentication method to access the data source
Authentication method

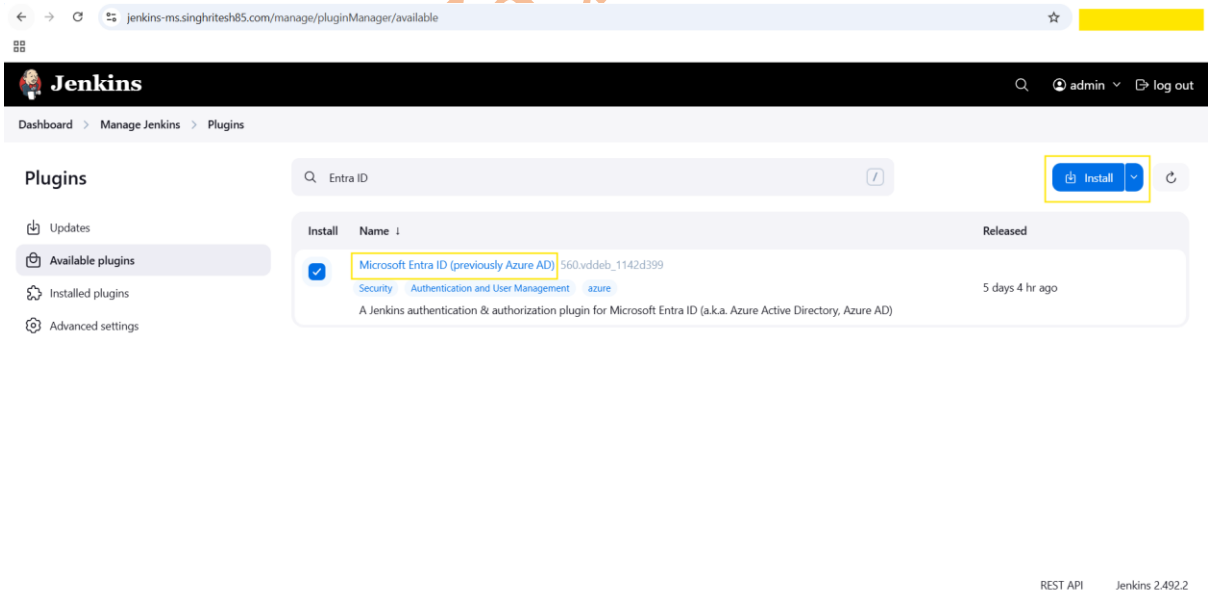


The data source for Grafana (Prometheus and Loki) had been configured as shown in the screenshot attached above.

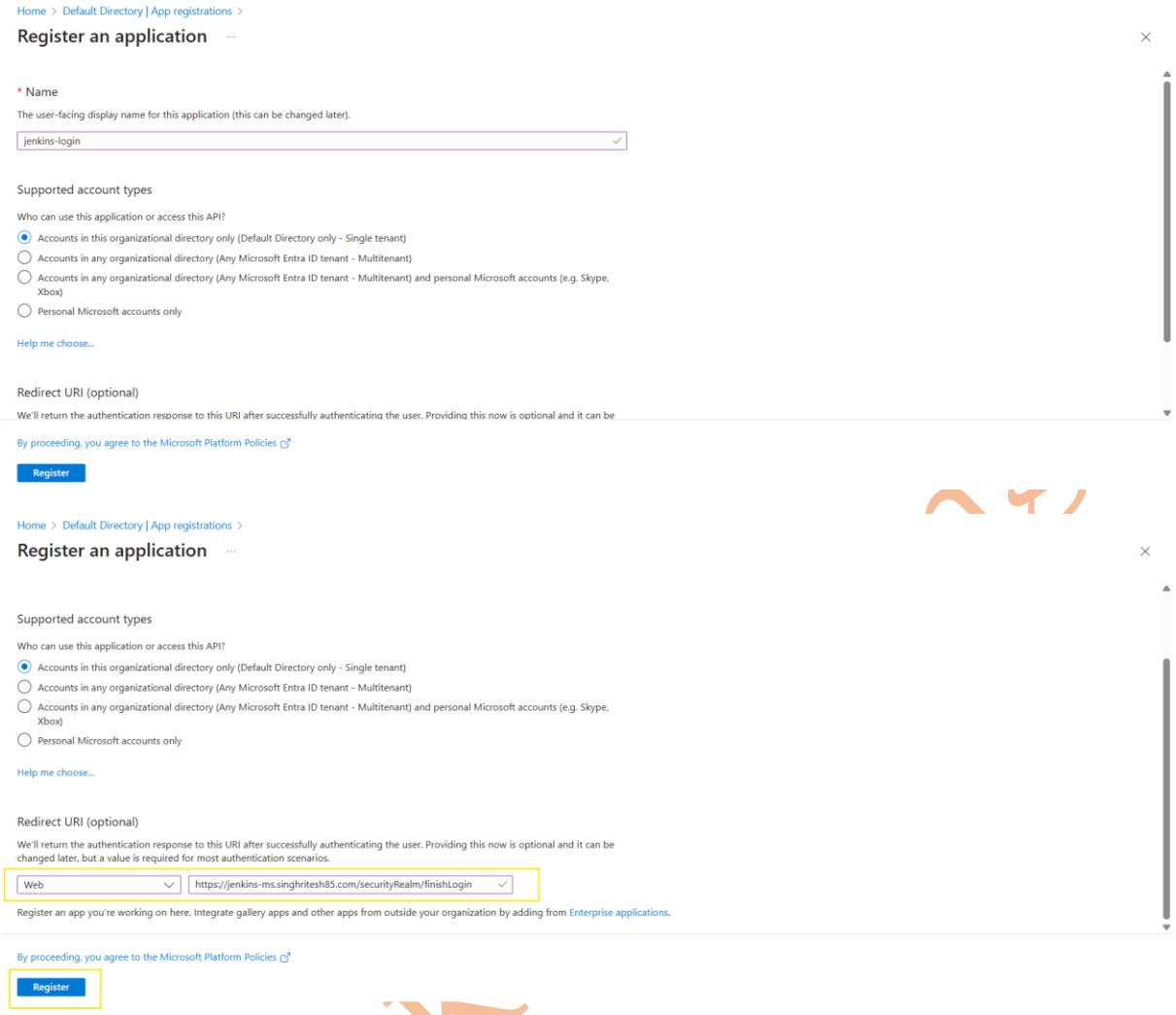
Now I will configure the Integration of Azure Entra ID with Jenkins, SonarQube and Grafana for Authentication

Configuration of Integration of Azure Entra ID with Jenkins

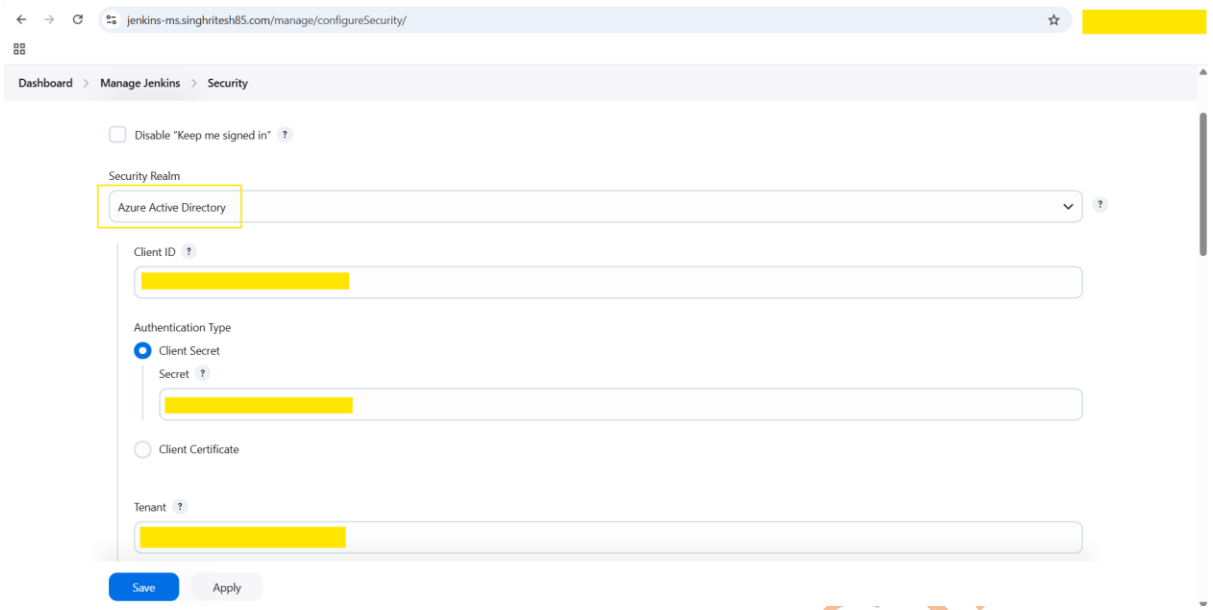
Search from the list of Available plugins for **Microsoft Entra ID** Plugin in Jenkins and install it as shown in the screenshot attached below.



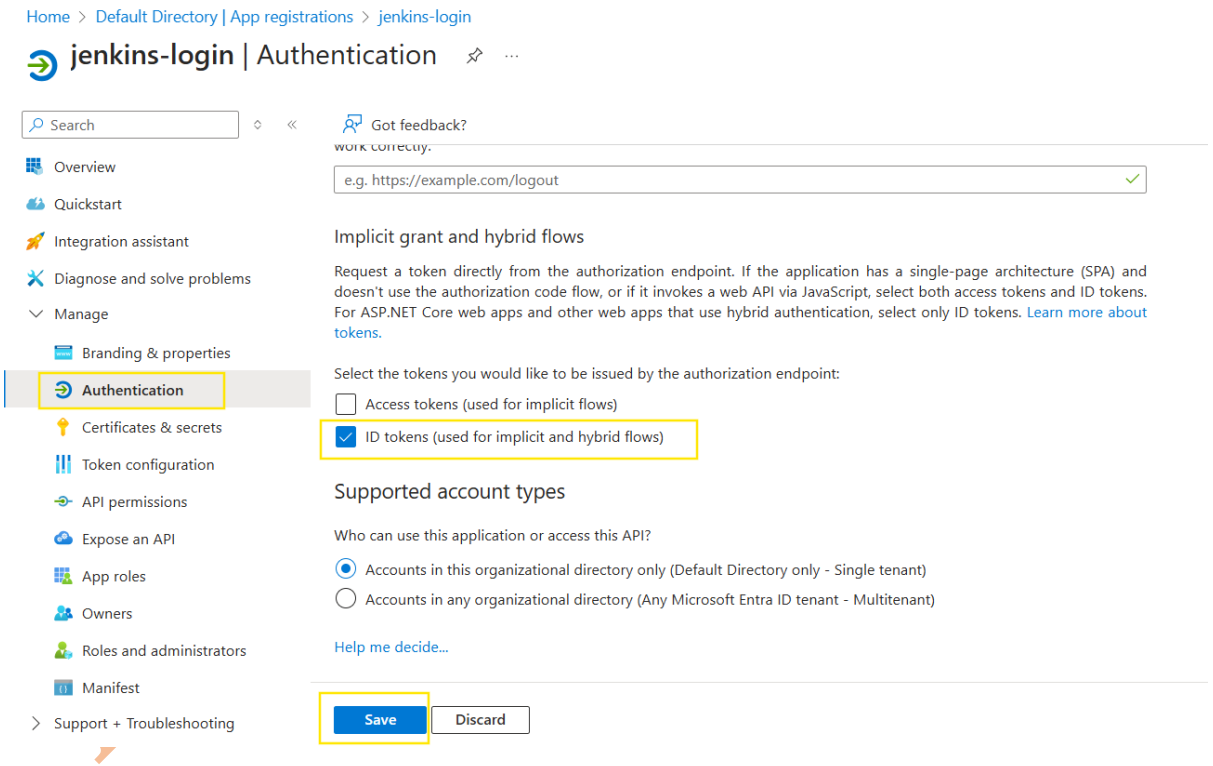
After installation of Azure Entra ID Plugin in Jenkins restart Jenkins and go to the Azure Portal and open Entra ID and create a new App Registration as shown in the screenshot attached below.



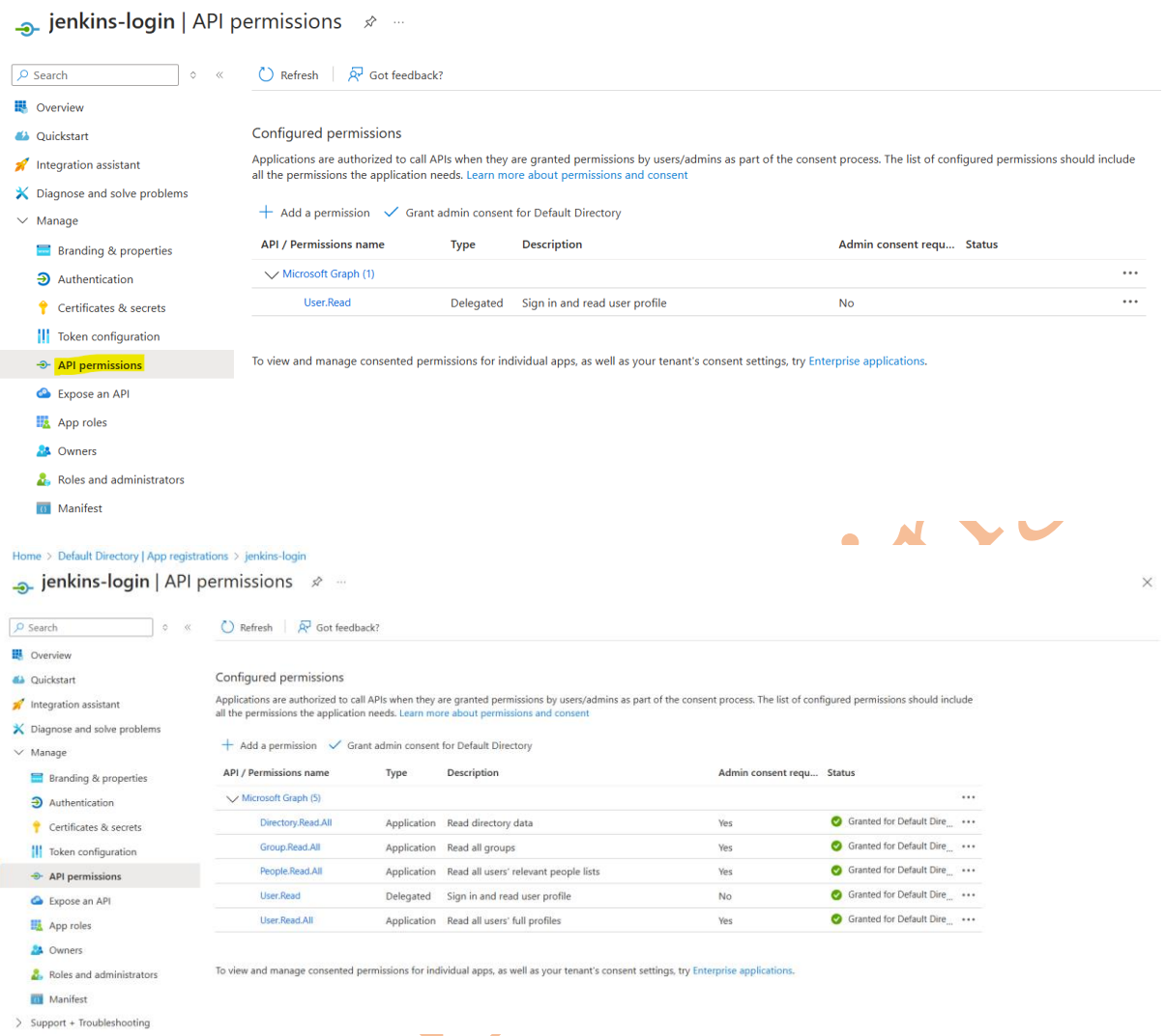
After App Registration in Azure go to Jenkins and **Manage Jenkins > Security > Security Realm and Select the Azure Active Directory** and provide the Client ID, Client Secret and Tenant ID then Save it as shown in screenshot attached below.



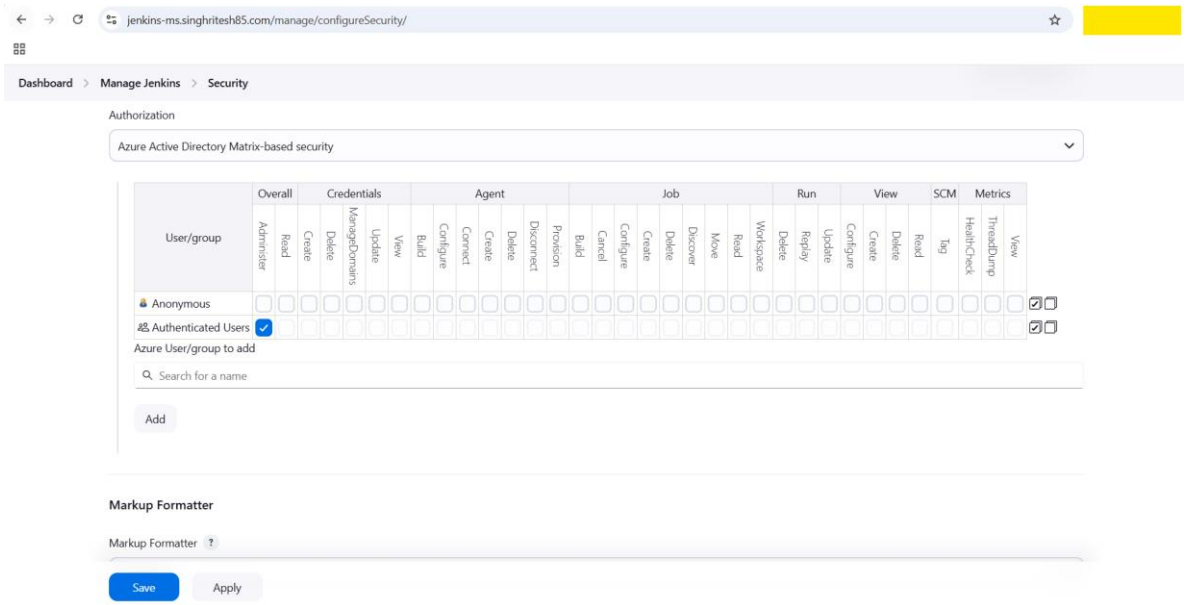
In the Registered Application go to Application and under **Implicit grant and hybrid flows** and select **ID Tokens** as shown in the screenshot attached below.



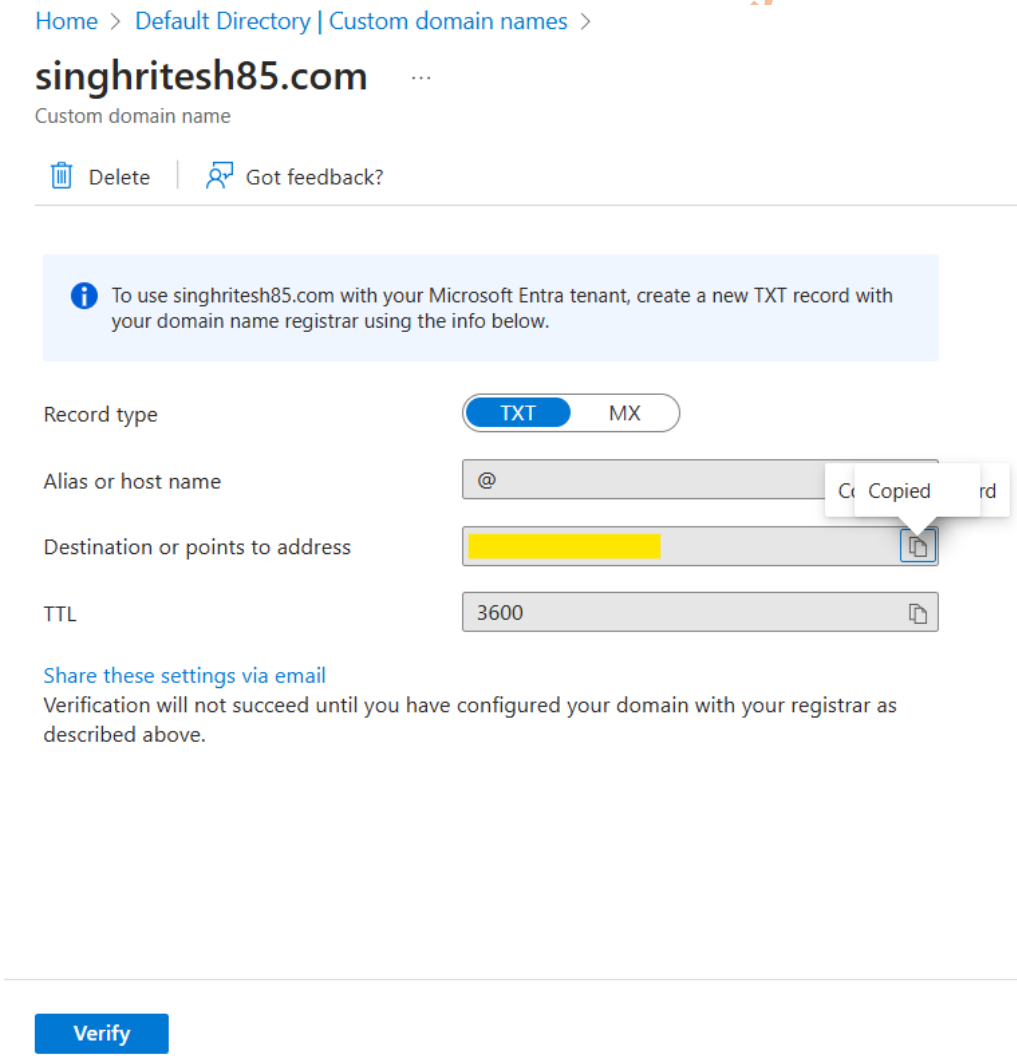
Now select API Permission > Add a permission > Microsoft Graph > Application Permissions People.Read.All, Group.Read.All, User.Read.All and Directory.Read.All in the created Registered App. Then Click on Grant admin consent as shown in the screenshot attached below



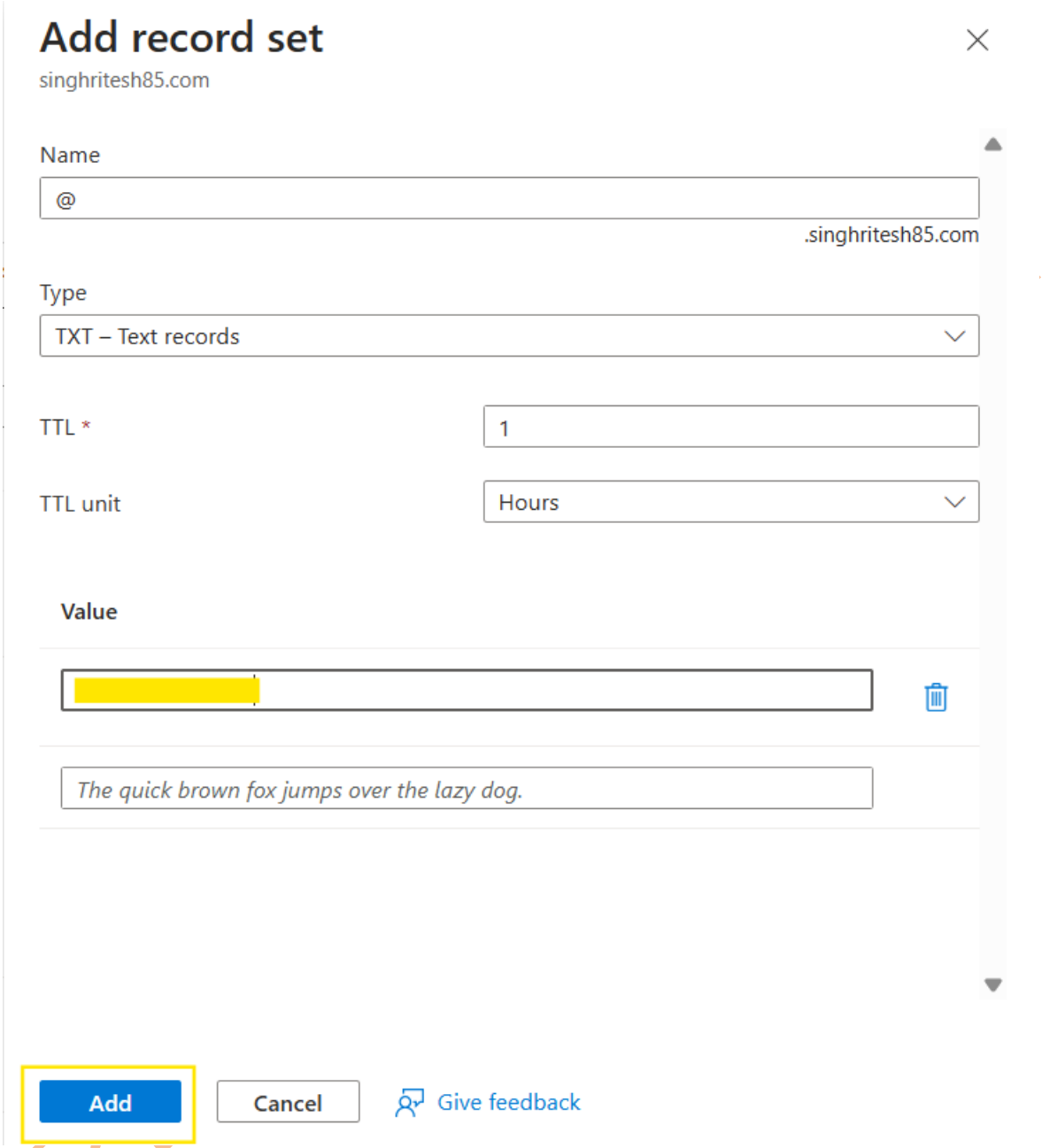
For Authorization select Azure Active Directory Matrix based Authorization as shown in the screenshot attached below, to login for the very first time provide Authenticated users as overall access and the login with a user and add other users.



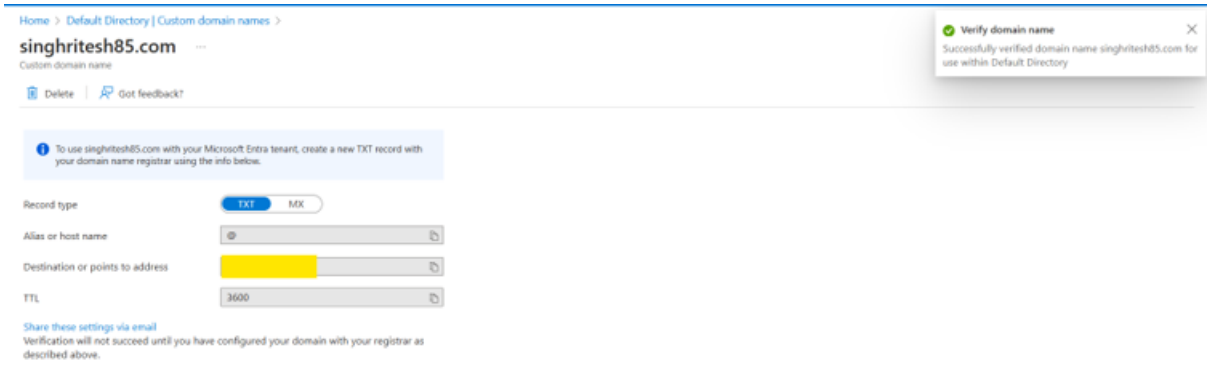
In my Azure Active Directory, I created a custom domain **singhritesh85.com** then made this custom domain as primary domain as shown in the screenshot attached below.



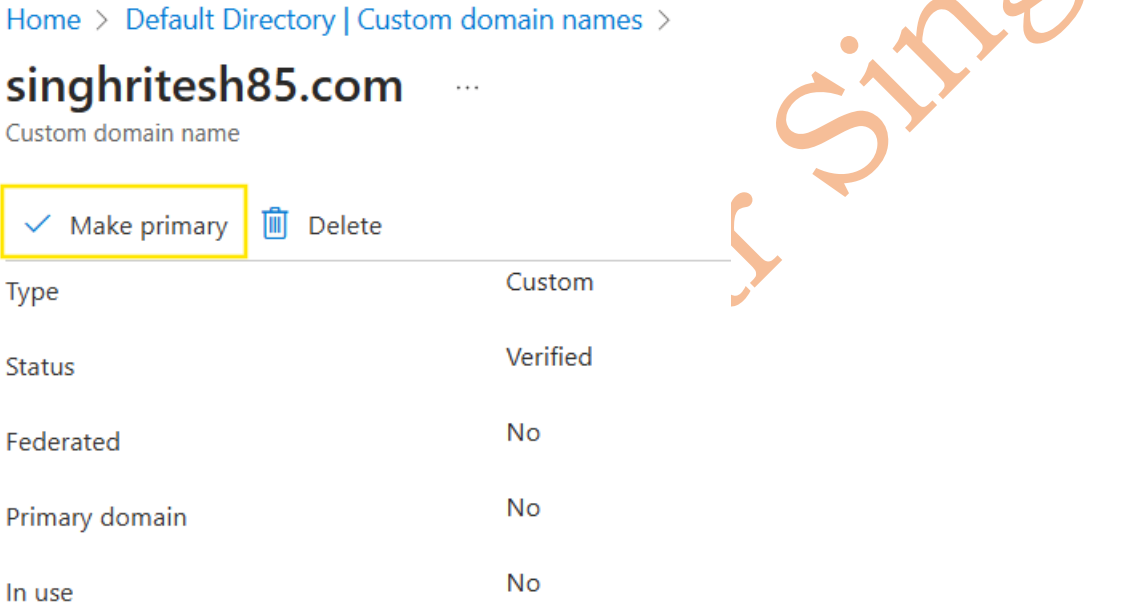
Do the entry for Alias or hostname with Destination in Azure DNS Zone to create a new record set as shown in the screenshot attached below.

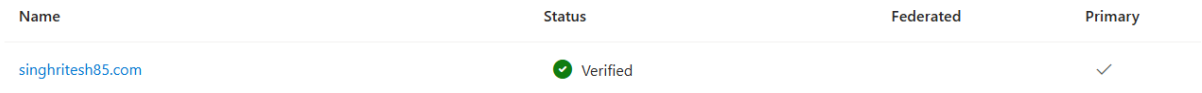


Then verified the it and found it was verified successfully as shown in the screenshot attached below.

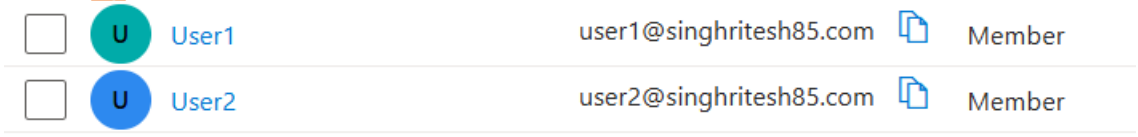


Then made this as a primary domain as shown in the screenshot attached below.

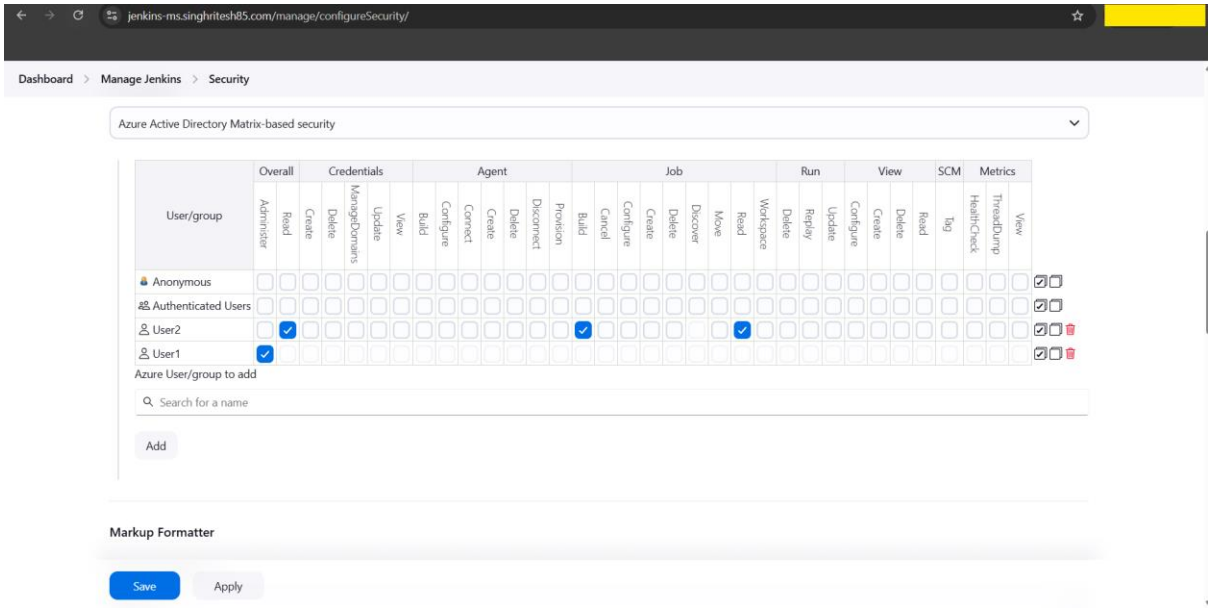




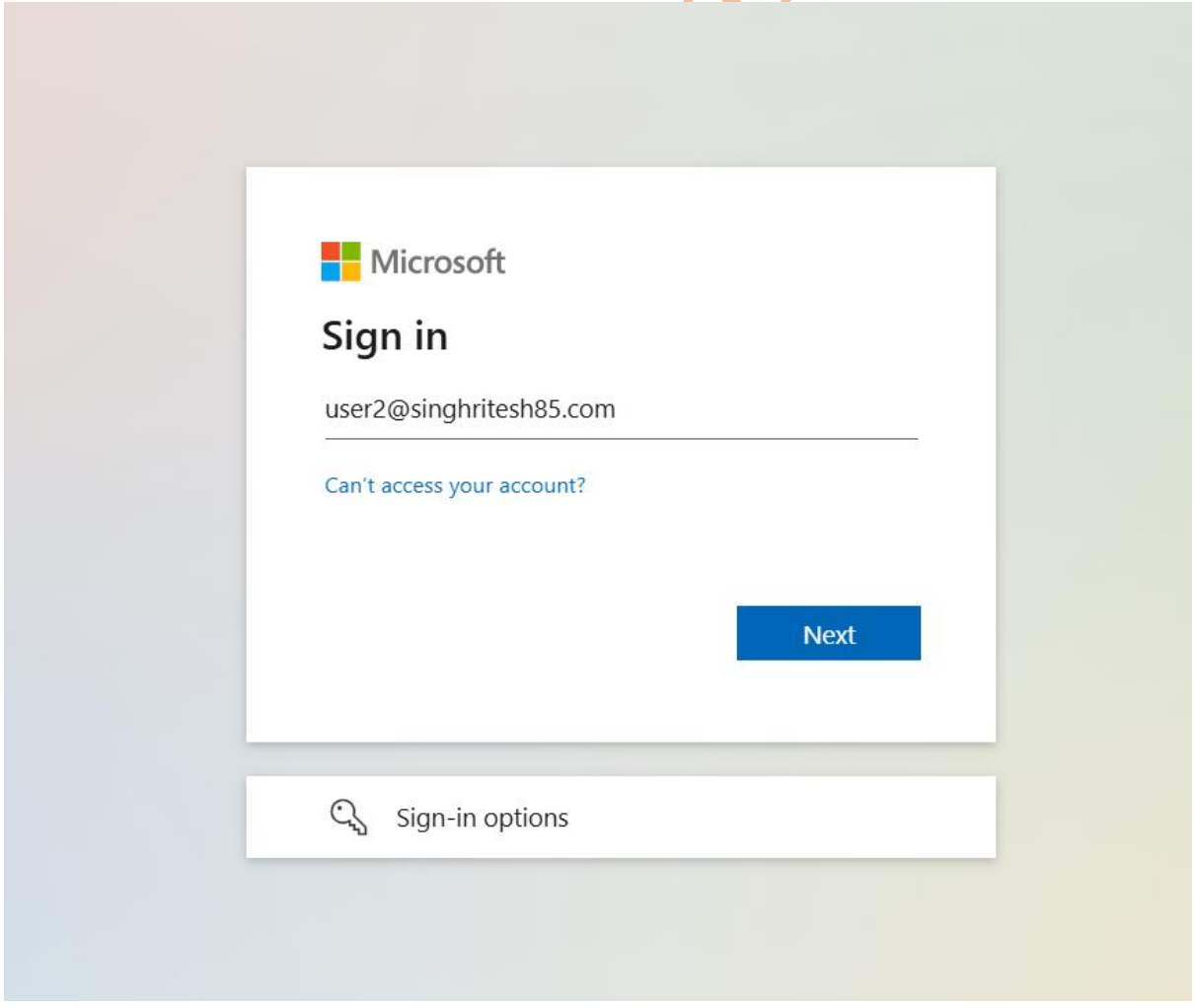
Then created two member users user1 and user2 as shown in the screenshot attached below.

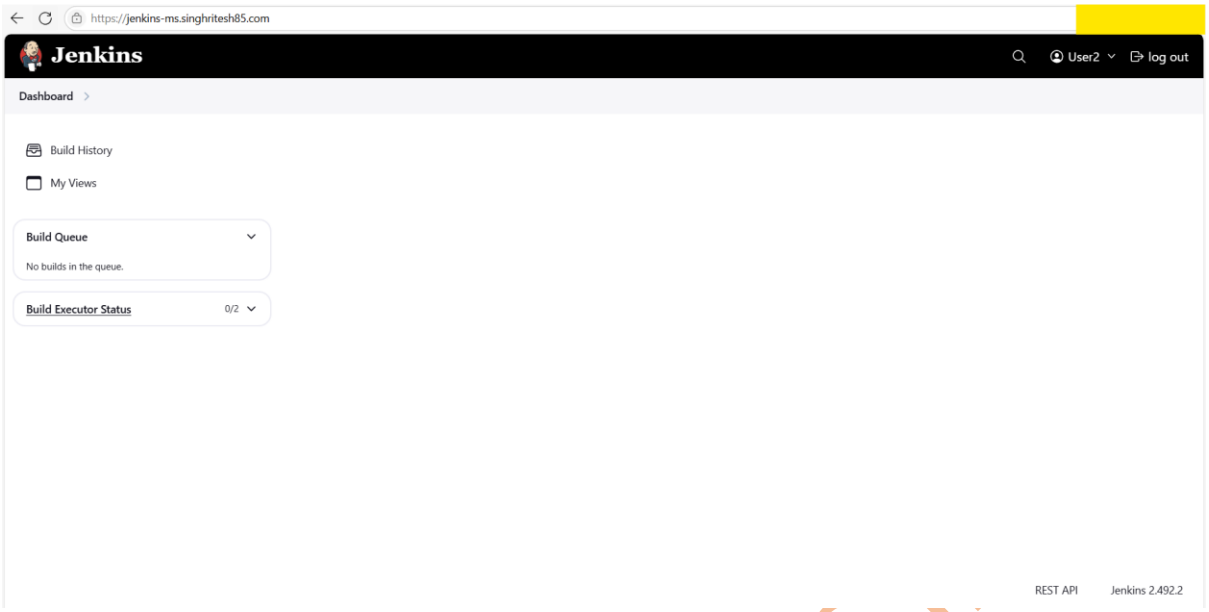


For the very first time I logged-in with a user then I provided user1 as administrator privileges and user2 as restricted access and remove the Authenticated user overall administrator access which I provided before first time logged-in as shown in the screenshot attached below.

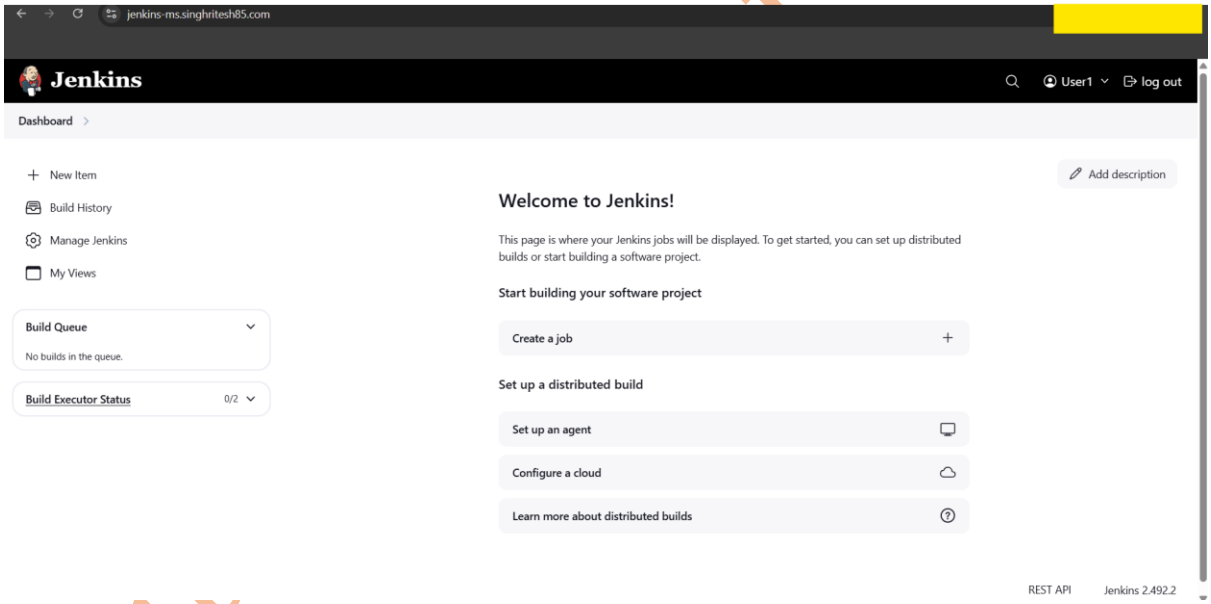


Then I logged-in with user2 and checked the access which user2 got as shown in the screenshot attached below.



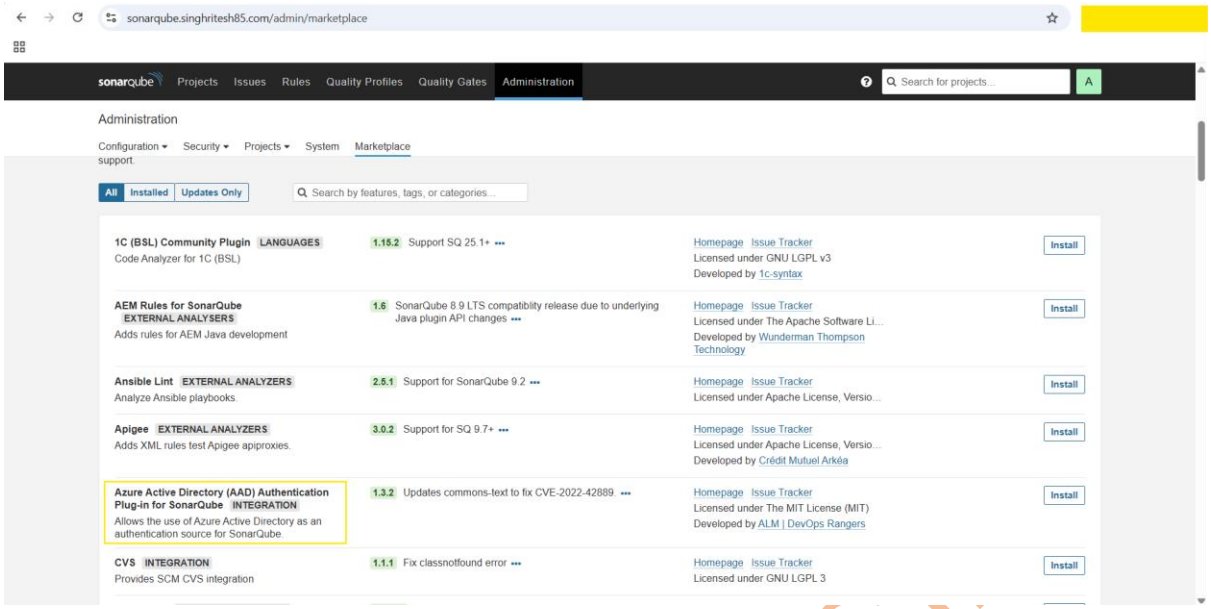


Below screenshot shows the access which I had when I logged-in with user1.

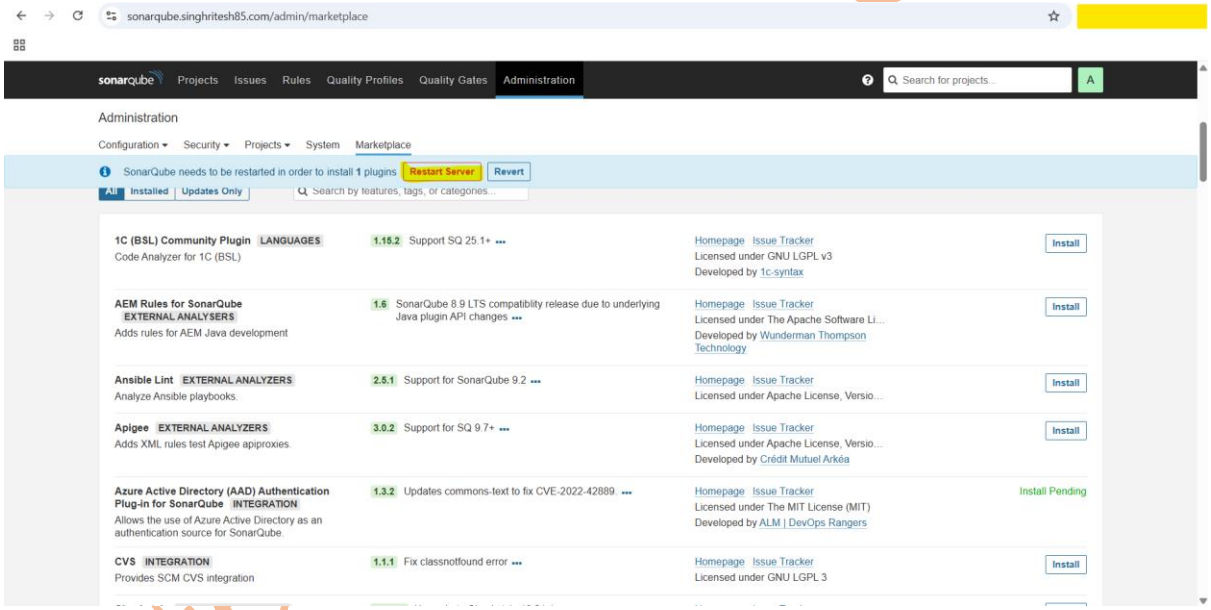


Integration of Azure Entra ID with SonarQube

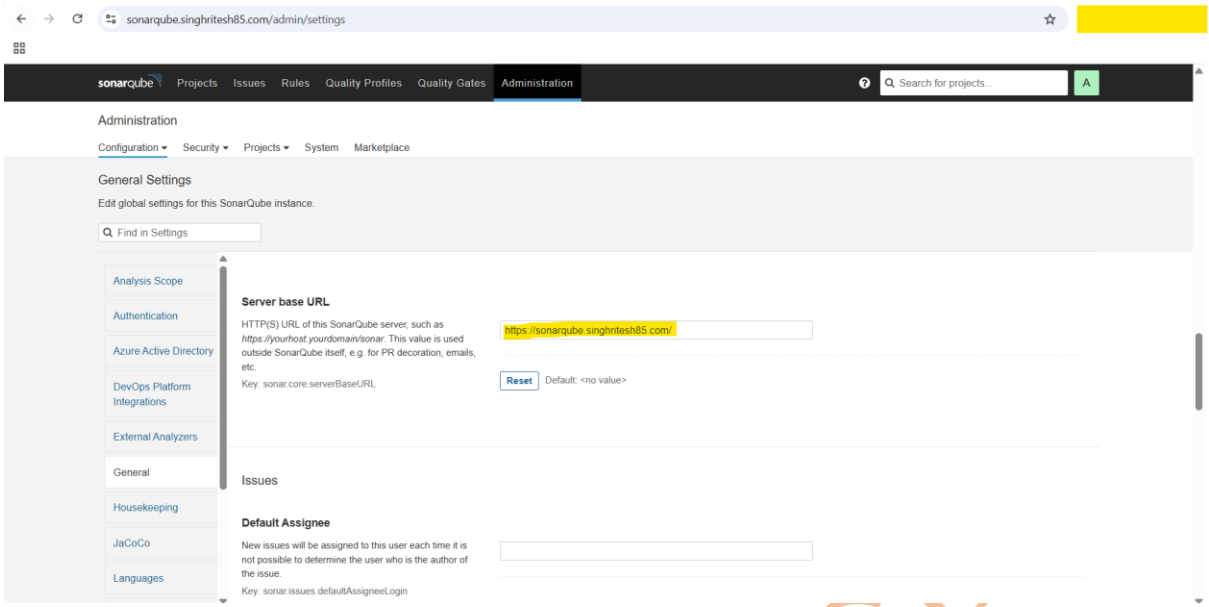
To Integrate Azure Active Directory with SonarQube I installed Azure Active Directory (AAD) Authentication Plug-in for SonarQube as shown in the screenshot attached below.



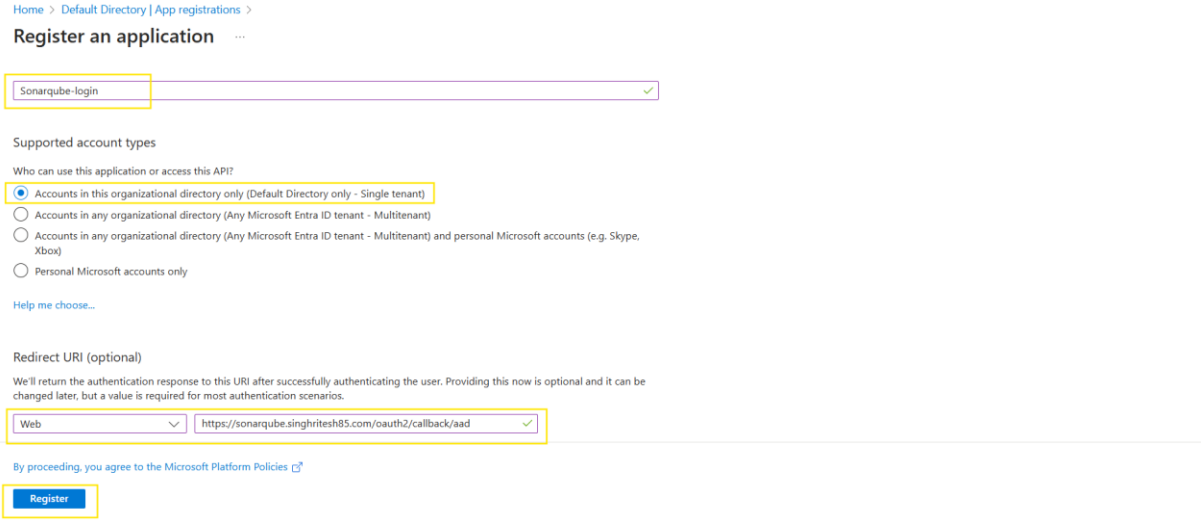
Then restart the sonarqube server as shown in the screenshot attached below.



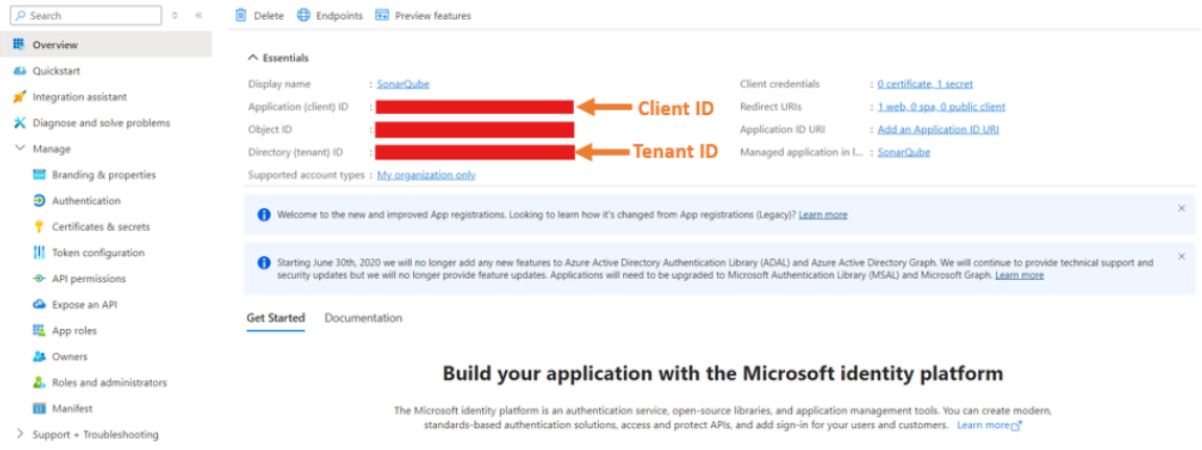
Go to SonarQube UI the to Administration > Configuration > General Settings > General and Add the **Server Base URL** as shown in the screenshot attached below.



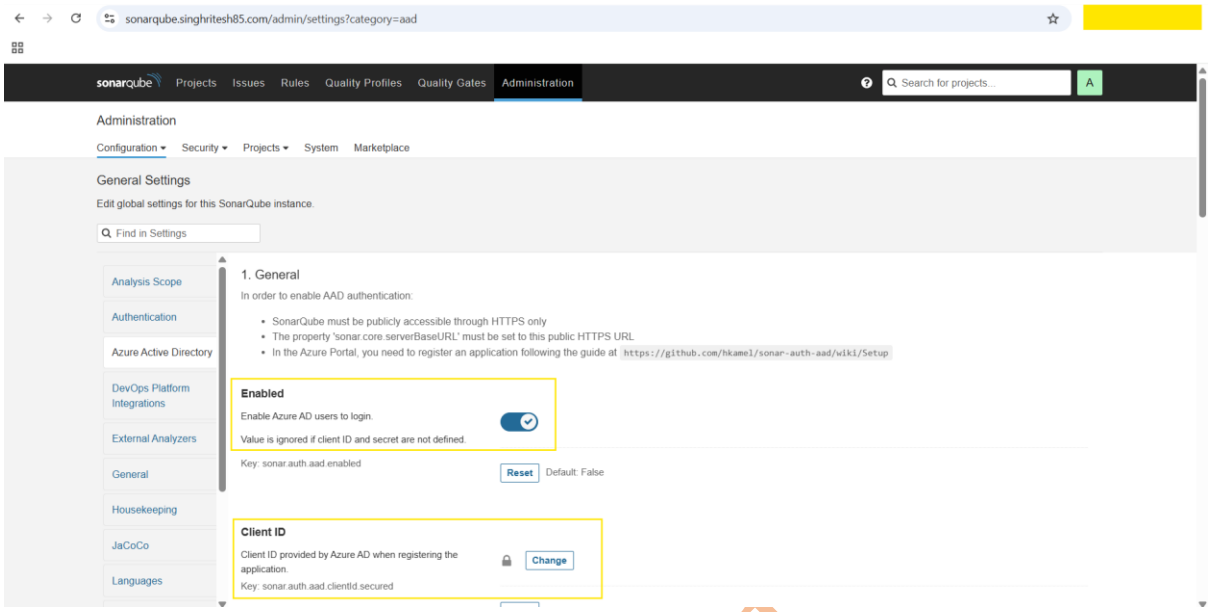
Create the App Registration in Azure Entra ID as shown in the screenshot attached below.

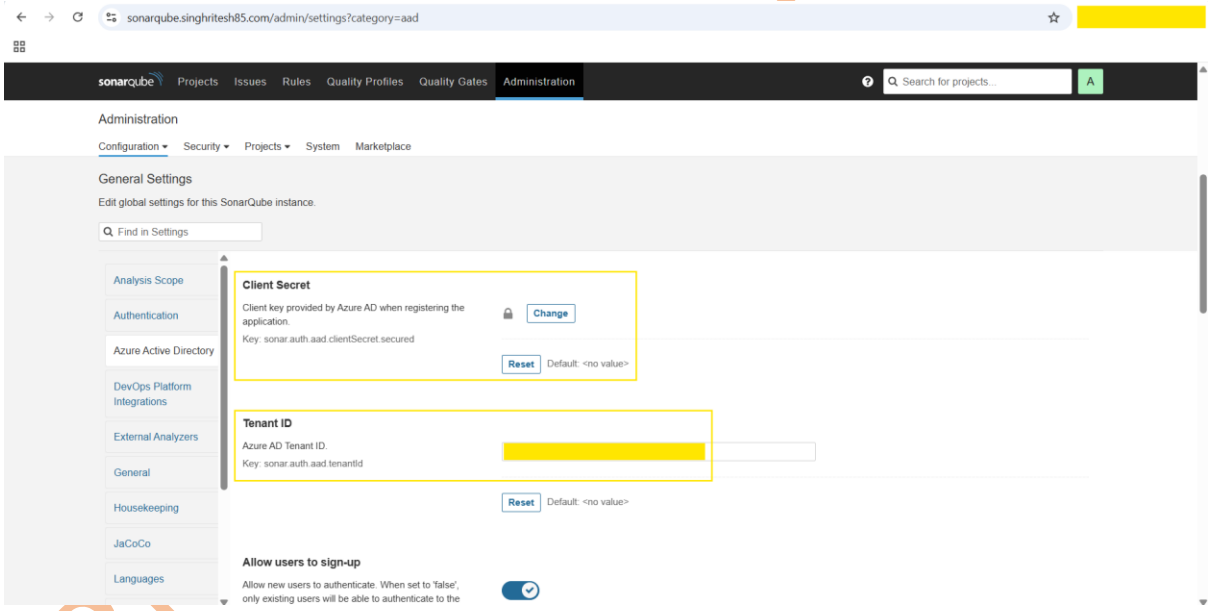


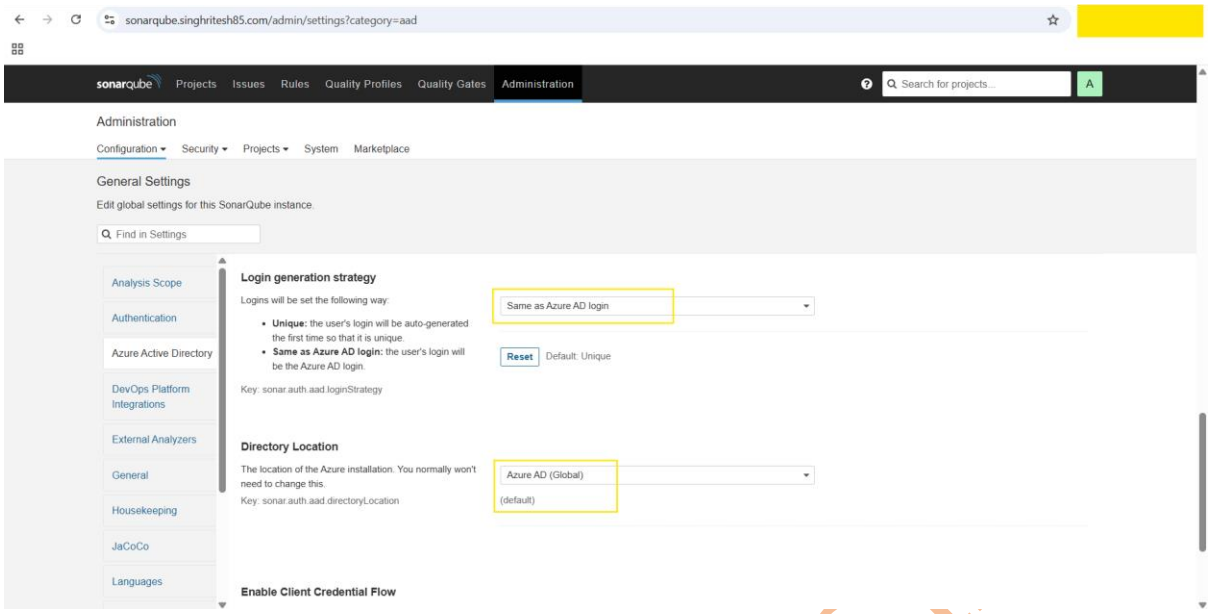
Create the client secret as shown in the screenshot attached below.



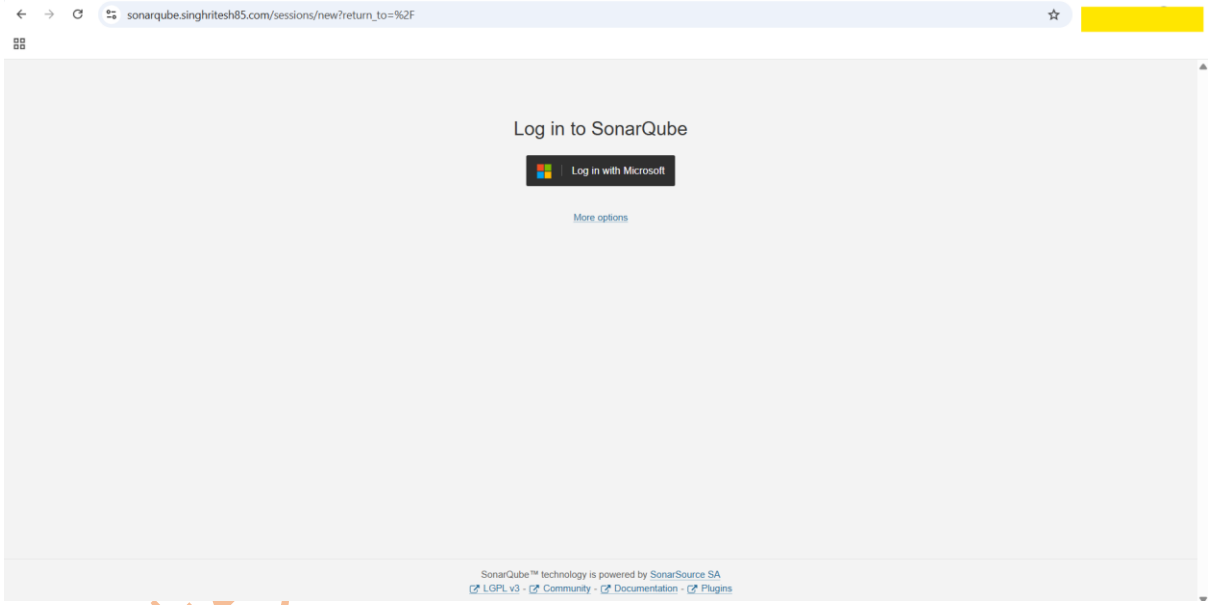
Now go to Sonarqube UI and then to Administration > Configuration > General Settings > Azure Active Directory as shown in the screenshot attached below and enable Azure AD users to login then provide the **client id, client secret** and **tenant id** as shown in the screenshot attached below.



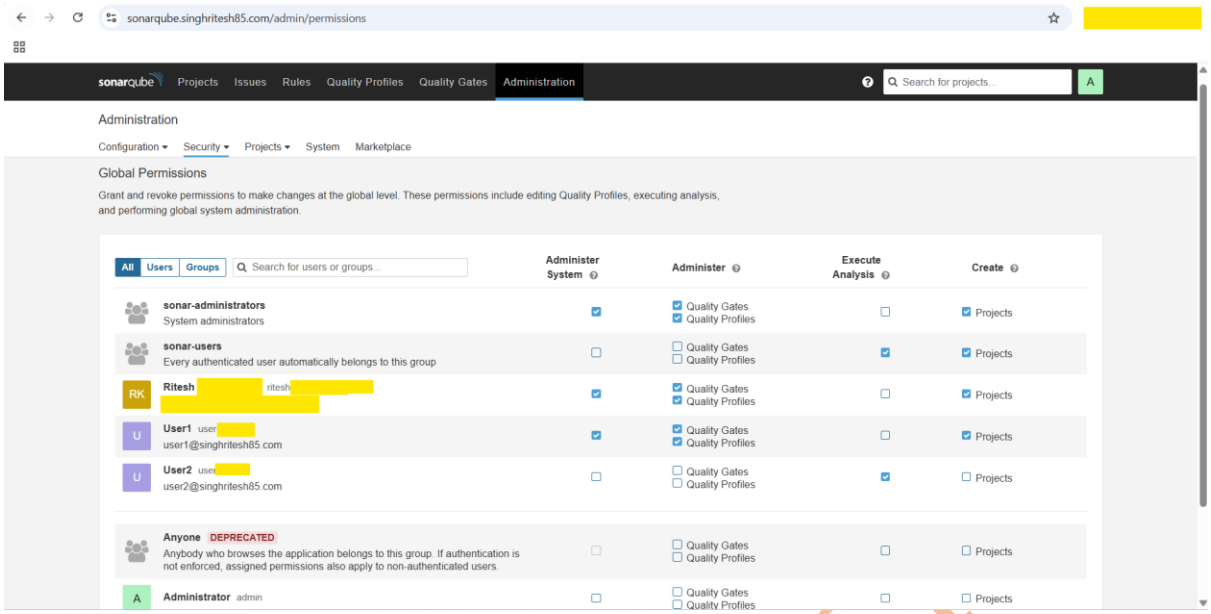




Then logout from SonarQube UI and login again as shown in the screenshot attached below.



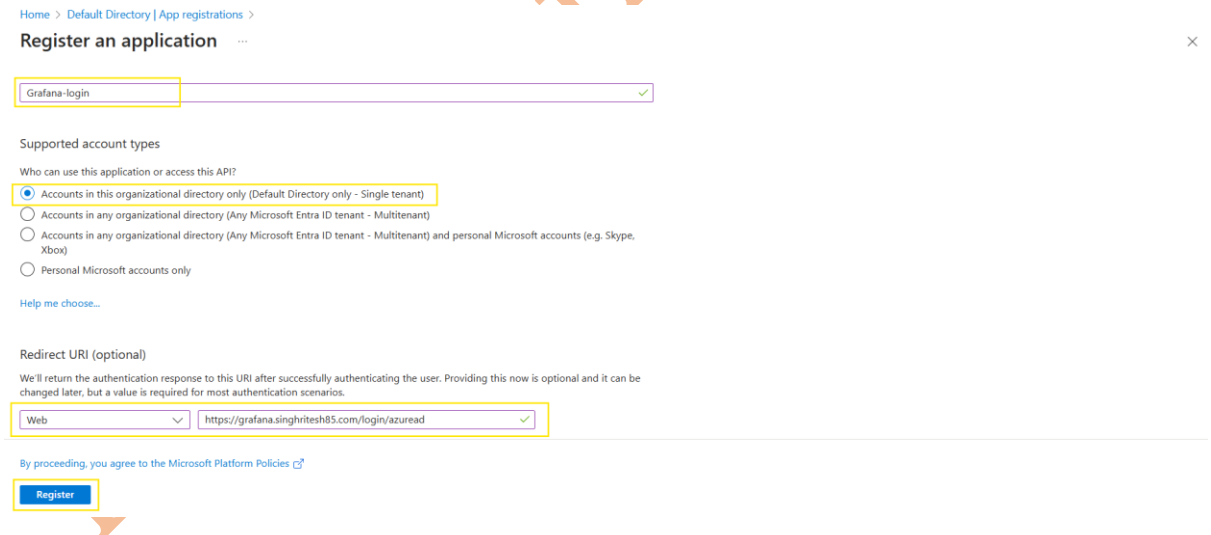
After the user will login once you can change their permission as shown in screenshot attached below.



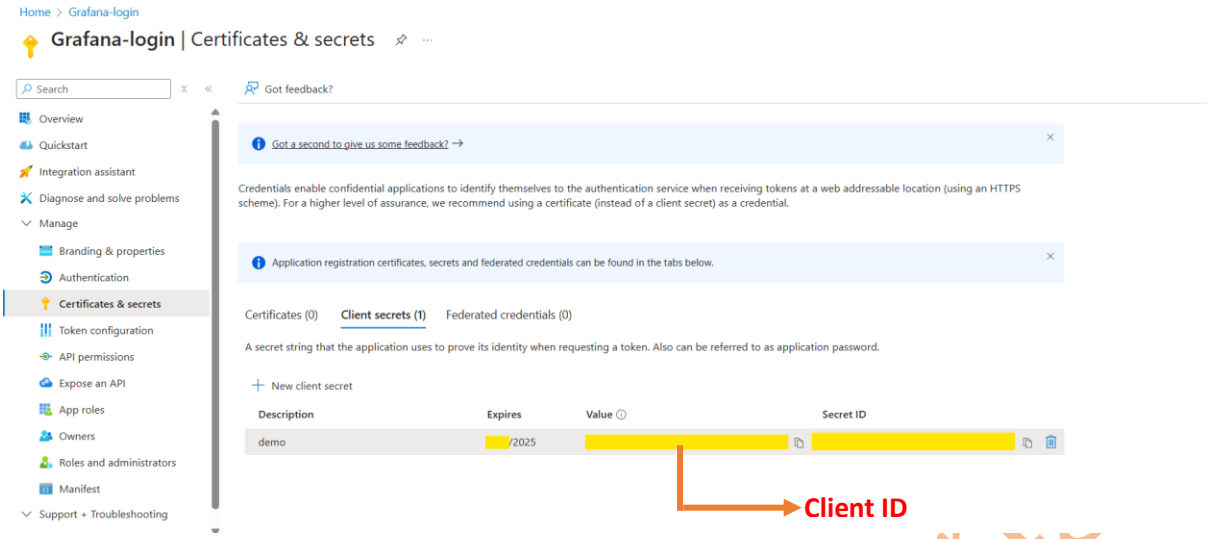
Integration of Azure Entra ID with Grafana

To integrate Azure Entra ID with Grafana I created an Azure Entra ID Service Principal (using APP Registration) as shown in the screenshot attached below.

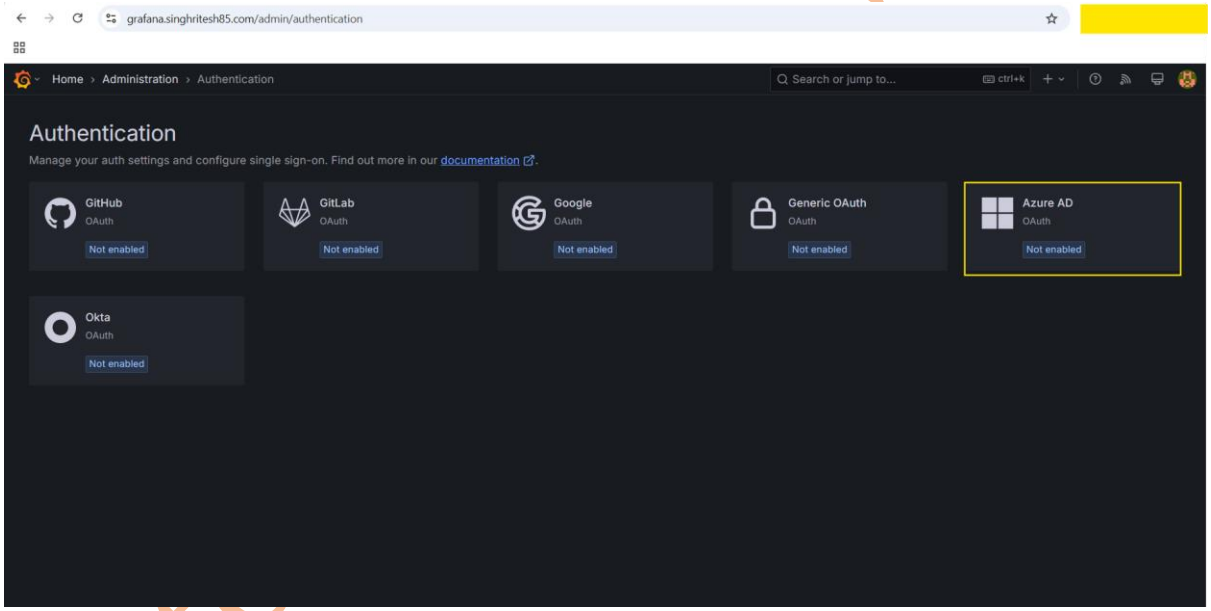
Go to App Registration > New Registration and configure a new Application as shown in the screenshot attached below.



Created the Secrets for Registered App **Grafana-login** in Azure Entra ID as shown in the screenshot attached below.

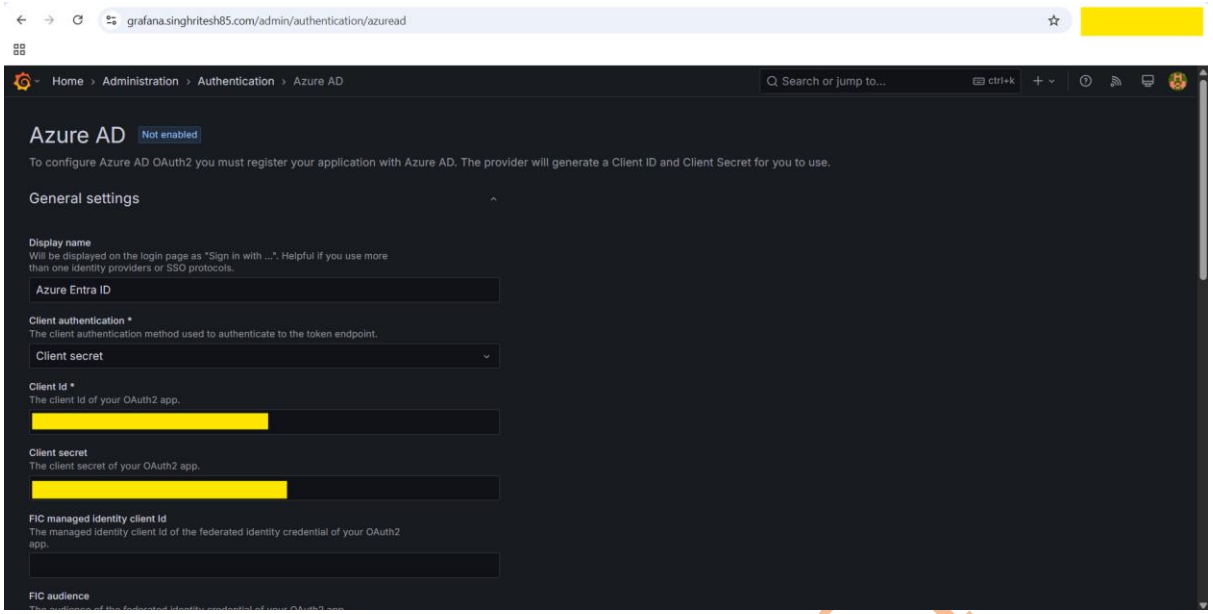


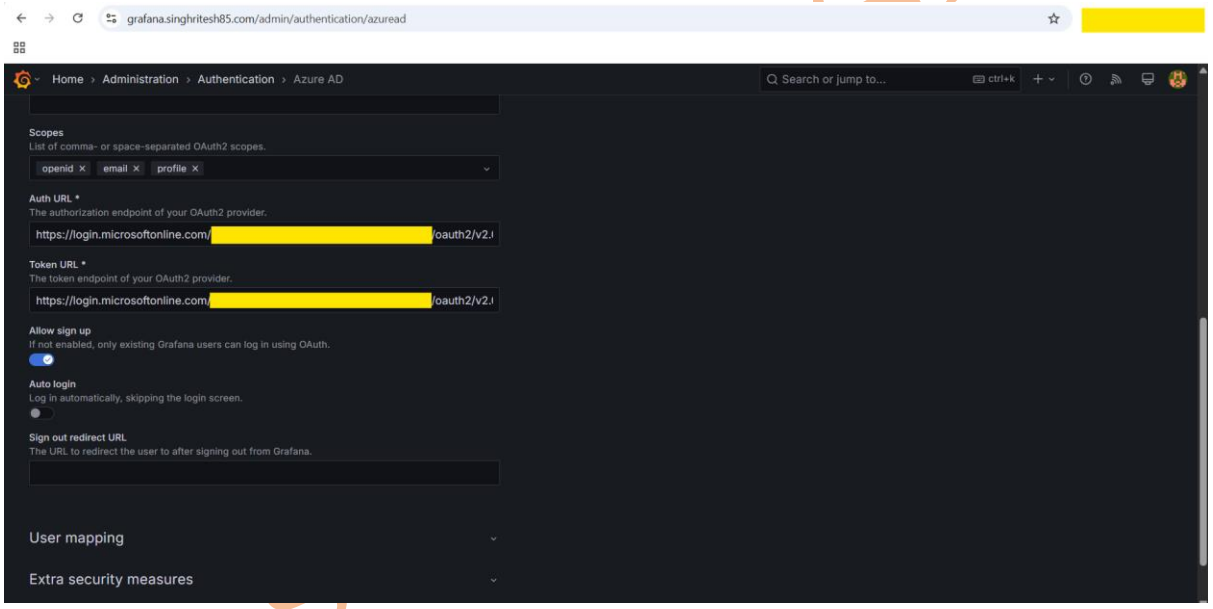
Now, login into the Grafana for the first time and update admin password then login into Grafana and Go to Grafana **Home > Administration > Authentication** and **enable the Azure AD OAuth** as shown in the screenshot attached below.

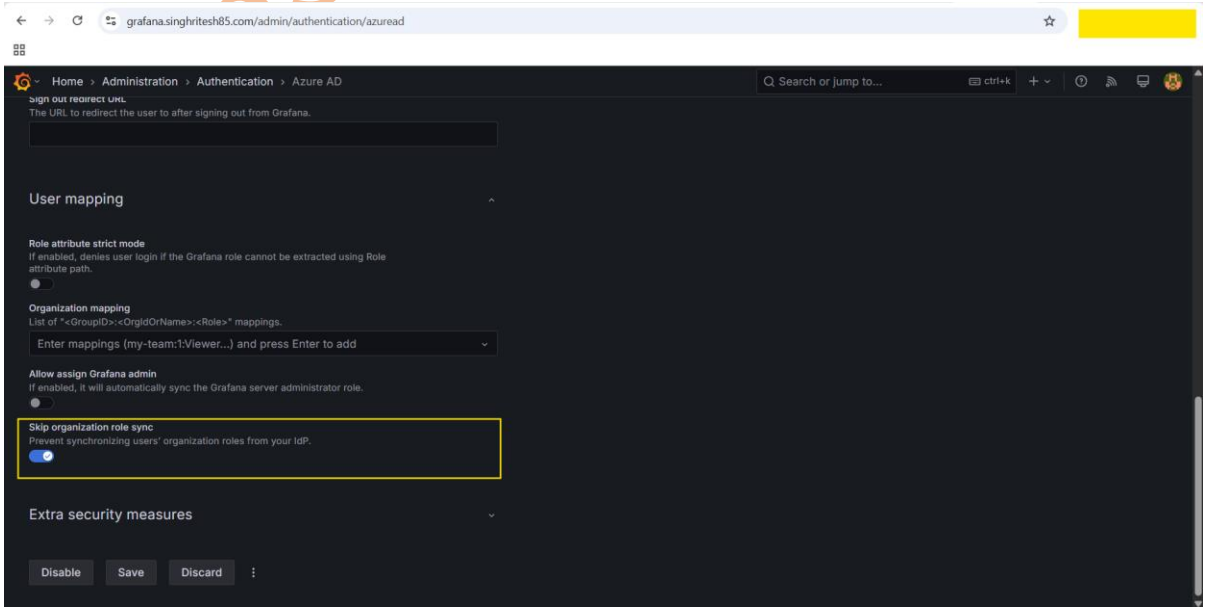


Provide the Client ID, Client Secret, Tenant ID and enable Allow Sign up and enable Skip organization role sync as shown in the screenshot attached below.

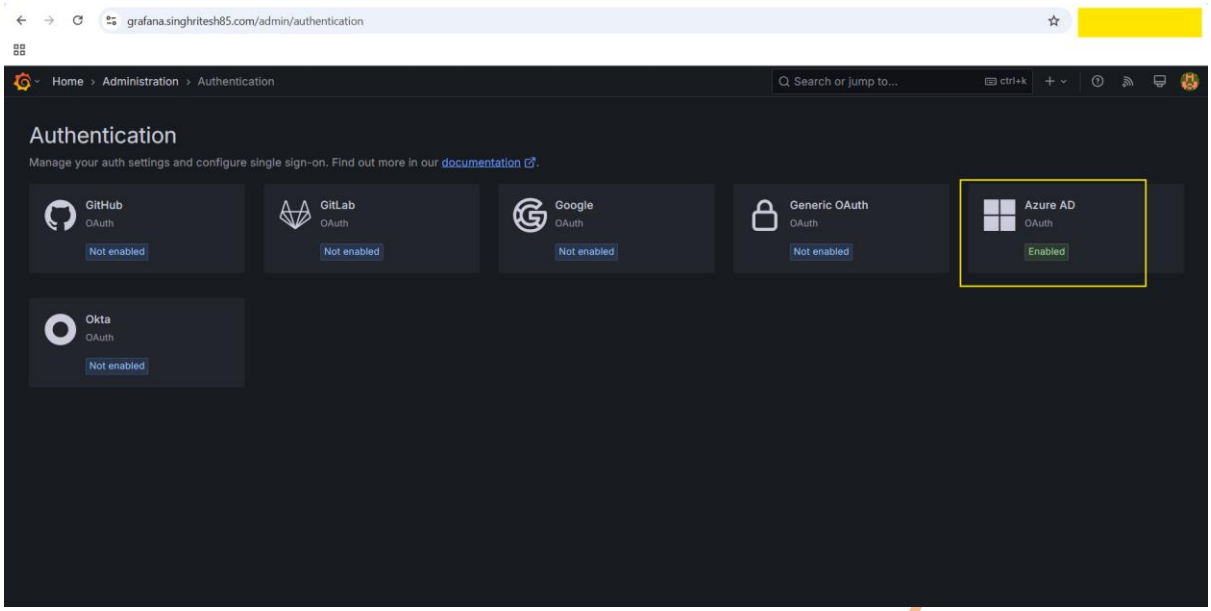
I enabled the **Skip organization role sync** otherwise Grafana will Sync the Azure Entra ID users with Main Org. Role as **Viewer** and Grafana Administrator cannot change it further but if I enabled **Skip organization role sync** then Grafana Administrator can change the viewer Role and can assign another Role as Editor or Admin. In this demonstration I created two users in Azure Entra ID user1 and user2. User1, I had provided as Administrator Role and user2 as default viewer access in Main Organisation (**Main Organisation always have Organisation ID 1**).


grafana.singhritesh85.com/admin/authentication/azuread
Home › Administration › Authentication › Azure AD
Search or jump to... ctrl+k
Azure AD
Not enabled
To configure Azure AD OAuth2 you must register your application with Azure AD. The provider will generate a Client ID and Client Secret for you to use.
General settings
Display name
Will be displayed on the login page as "Sign in with ...". Helpful if you use more than one identity providers or SSO protocols.
Azure Entra ID
Client authentication *
The client authentication method used to authenticate to the token endpoint.
Client secret
Client Id *
The client Id of your OAuth2 app.
Client secret
The client secret of your OAuth2 app.
FIC managed identity client Id
The managed identity client Id of the federated identity credential of your OAuth2 app.
FIC audience

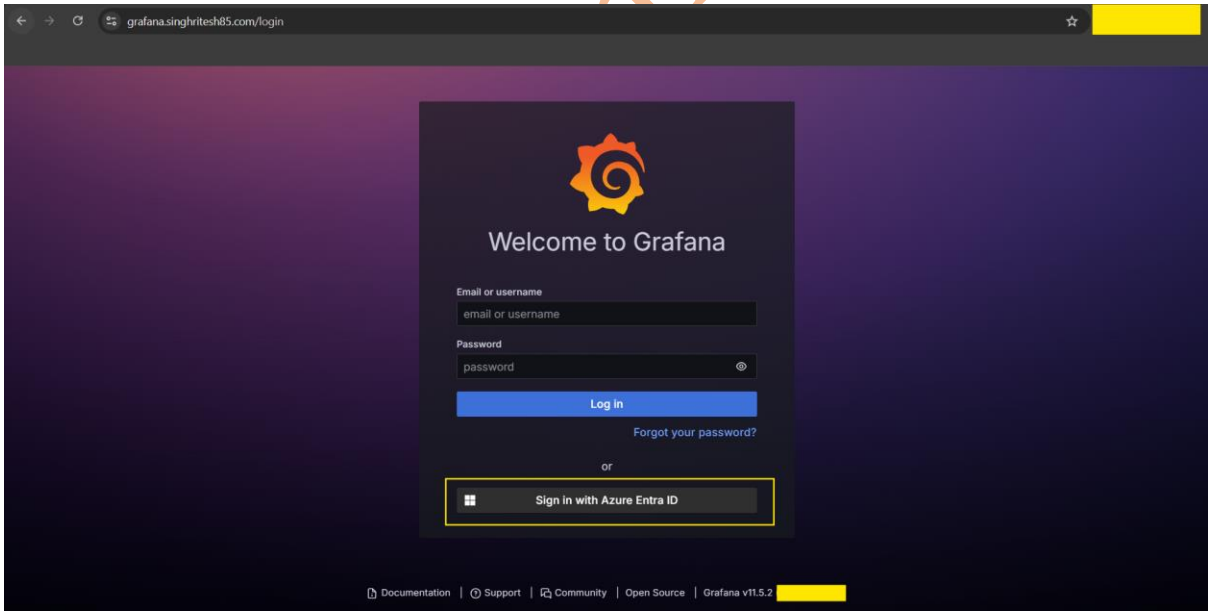

grafana.singhritesh85.com/admin/authentication/azuread
Home › Administration › Authentication › Azure AD
Search or jump to... ctrl+k
Scopes
List of comma- or space-separated OAuth2 scopes.
openid × email × profile ×
Auth URL *
The authorization endpoint of your OAuth2 provider.
https://login.microsoftonline.com/ /oauth2/v2.
Token URL *
The token endpoint of your OAuth2 provider.
https://login.microsoftonline.com/ /oauth2/v2.
Allow sign up
If not enabled, only existing Grafana users can log in using OAuth.
Auto login
Log in automatically, skipping the login screen.
Sign out redirect URL
The URL to redirect the user to after signing out from Grafana.
User mapping
Extra security measures


grafana.singhritesh85.com/admin/authentication/azuread
Home › Administration › Authentication › Azure AD
Search or jump to... ctrl+k
The URL to redirect the user to after signing out from Grafana.
User mapping
Role attribute strict mode
If enabled, denies user login if the Grafana role cannot be extracted using Role attribute path.
Organization mapping
List of "<GroupID>:<OrgIdOrName>:<Role>" mappings.
Enter mappings (my-team:1:Viewer...) and press Enter to add
Allow assign Grafana admin
If enabled, it will automatically sync the Grafana server administrator role.
Skip organization role sync
Prevent synchronizing users' organization roles from your IdP.
Extra security measures
Disable
Save
Discard

Now the Azure AD OAuth is enable as shown in the screenshot attached below.



Now, Grafana login dashboard will show the option of sign-in with Azure Entra ID as shown in the screenshot attached below. I logged-in to the Grafana dashboard with the Azure Entra ID user as shown in the screenshot attached below.



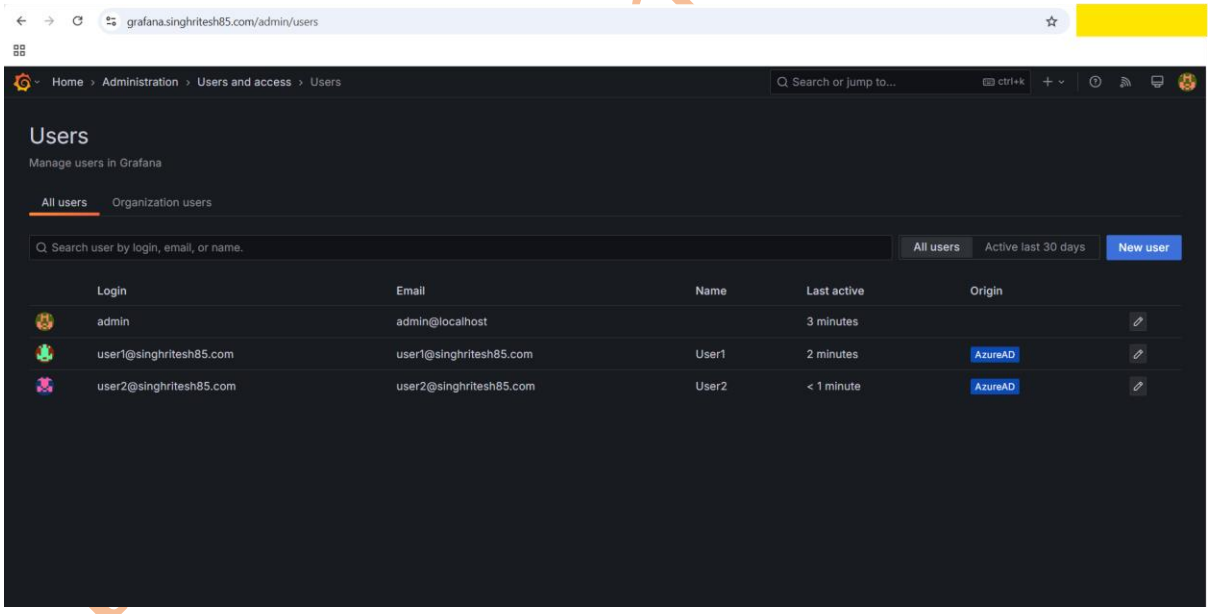
Now go to the Grafana Server and open the file **/etc/grafana/grafana.ini** and edit **root_url** under the [server] then restart the **grafana-server** service as shown in the screenshot attached below.

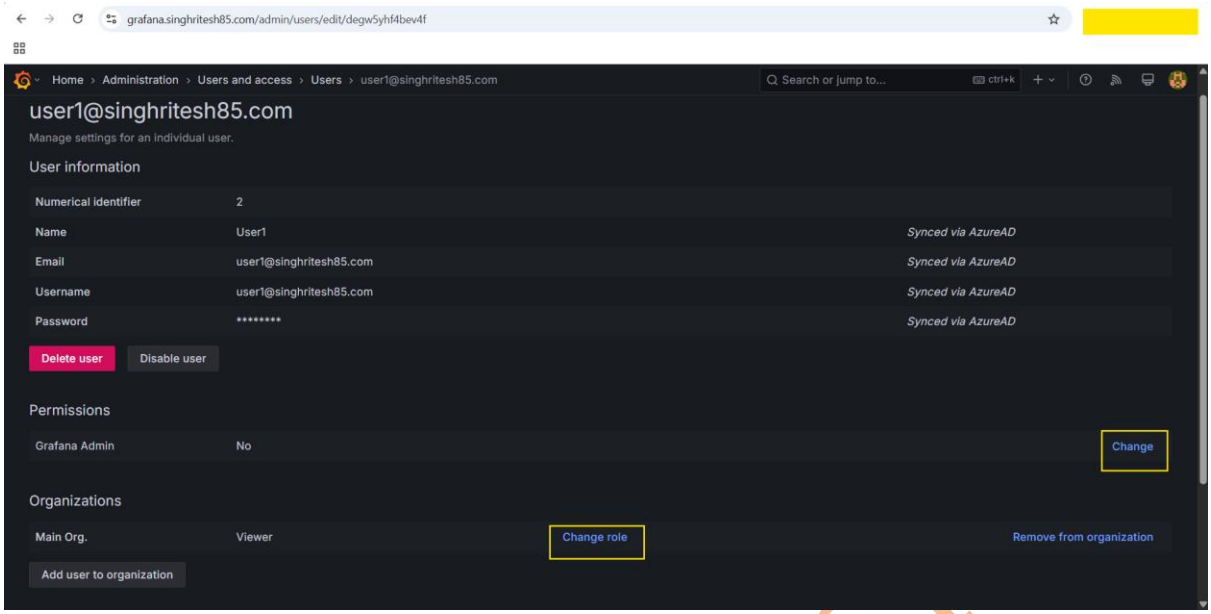
```
[root@                    ~]# vim /etc/grafana/grafana.ini
```

```
#################################### Server ####################################
[server]
# Protocol (http, https, h2, socket)
;protocol = http

# Minimum TLS version allowed. By default, this value is empty. Accepted values are: TLS1.2, TLS1.3. If nothing is set
;min_tls_version = ""

# The ip address to bind to, empty will bind to all interfaces
;http_addr =

# The http port to use
;http_port = 3000

# The public facing domain name used to access grafana from a browser
;domain = localhost

# Redirect to correct domain if host header does not match domain
# Prevents DNS rebinding attacks
;enforce_domain = false

# The full public facing url you use in browser, used for redirects and emails
# If you use reverse proxy and sub path specify full url (with sub path)
;root_url = %(protocol)s://%(domain)s:%(http_port)s/

root_url = "https://grafana.singhritesh85.com"
# Serve Grafana from subpath specified in `root_url` setting. By default it is set to `false` for compatibility reasons
```

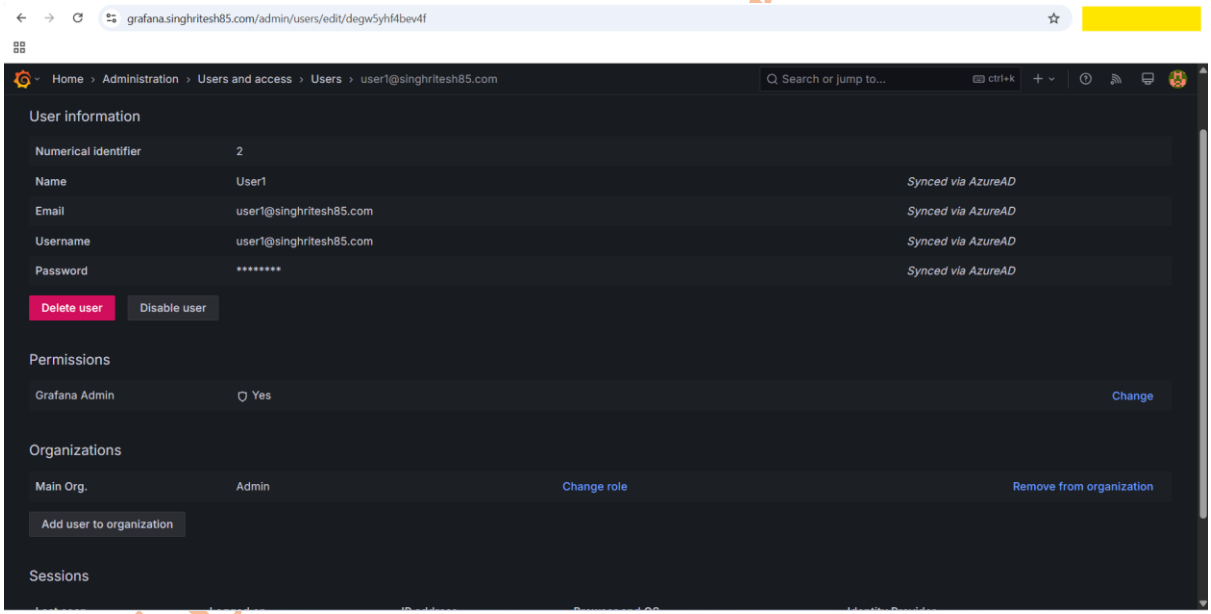
```
[root@                ~]# systemctl restart grafana-server.service
[root@                ~]# systemctl status grafana-server.service
● grafana-server.service - Grafana instance
   Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2025-
```

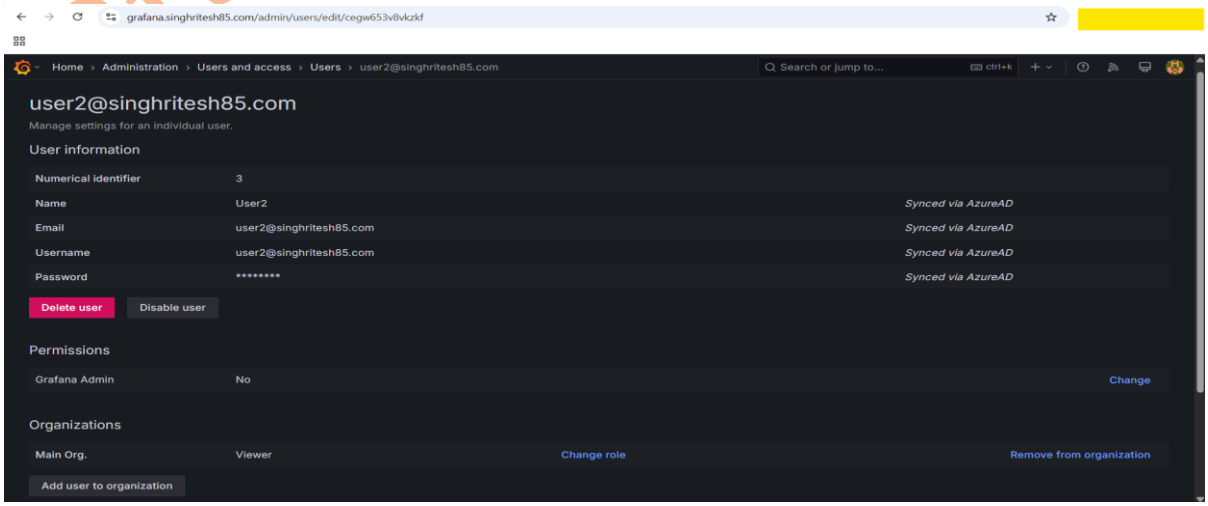
For the first time when a user logged-in, they had viewer access which Grafana Administrator can change as per the requirement as shown in the screenshot attached below.





As stated earlier I will provide Admin access to user1 and viewer access to user2 as shown in the screenshot attached below.





After that user1 will refresh Grafana UI and their Access will be reflected on the UI as well.

Installation of Nginx Ingress Controller in EKS Cluster

kubectl create ns ingress-nginx
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update

helm install ingress-nginx ingress-nginx/ingress-nginx -n ingress-nginx --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-ssl-cert"=arn:aws:acm:us-east-2:02XXXXXXXX6:certificate/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-connection-idle-timeout"="60" --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-cross-zone-load-balancing-enabled"="true" --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-type"="elb" --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-backend-protocol"=http --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-ssl-ports"=https --set controller.service.targetPorts.https=http --set-string controller.config.use-forwarded-headers="true"

```
[jenkins@                    ~]$ kubectl create ns ingress-nginx
```

```
[jenkins@                    ~]$ helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

```
[jenkins@                    ~]$ helm repo update
```

```
[jenkins@ip-10-10-4-202 ~]$ helm install ingress-nginx ingress-nginx/ingress-nginx -n ingress-nginx --set controller.service.annotations."service\.beta\.kuber
netes\.io/aws-load-balancer-ssl-cert"=arn:aws:acm:us-east-2:02        6:certificate/                                --set controller.service.annotations
."service\.beta\.kubernetes\.io/aws-load-balancer-connection-idle-timeout"="60" --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-b
alancer-cross-zone-load-balancing-enabled"="true" --set controller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-type"="elb" --set cont
roller.service.annotations."service\.beta\.kubernetes\.io/aws-load-balancer-backend-protocol"=http --set controller.service.annotations."service\.beta\.kubern
etes\.io/aws-load-balancer-ssl-ports"=https --set controller.service.targetPorts.https=http --set-string controller.config.use-forwarded-headers="true"
```

```
[jenkins@                    ~]$ kubectl get pods -n ingress-nginx
NAME                                        READY   STATUS    RESTARTS   AGE
ingress-nginx-controller-6              b   1/1     Running   0          2m31s
[jenkins@                    ~]$ kubectl get svc -n ingress-nginx
NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP                                                              PORT(S)
            AGE
ingress-nginx-controller             LoadBalancer   172.      .165   a                                      1.us-east-2.elb.amazonaws.com   80:3    7/TCP,44
3:3    8/TCP   2m38s
ingress-nginx-controller-admission   ClusterIP      172.      190    <none>                                                                   443/TCP
            2m38s
```

Installation of ArgoCD in EKS Cluster

In this project for EKS Cluster deployment I used ArgoCD and the ArgoCD CLI, I installed the ArgoCD as shown in the screenshot attached below.

kubectl create namespace argocd

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

```
[jenkins@                    ~]$ kubectl get nodes
NAME                                        STATUS   ROLES    AGE   VERSION
ip-10-      -121.us-east-2.compute.internal   Ready    <none>   87m   v1.30.4-eks-a      9
ip-10-      -153.us-east-2.compute.internal   Ready    <none>   87m   v1.30.4-eks-a      9
```

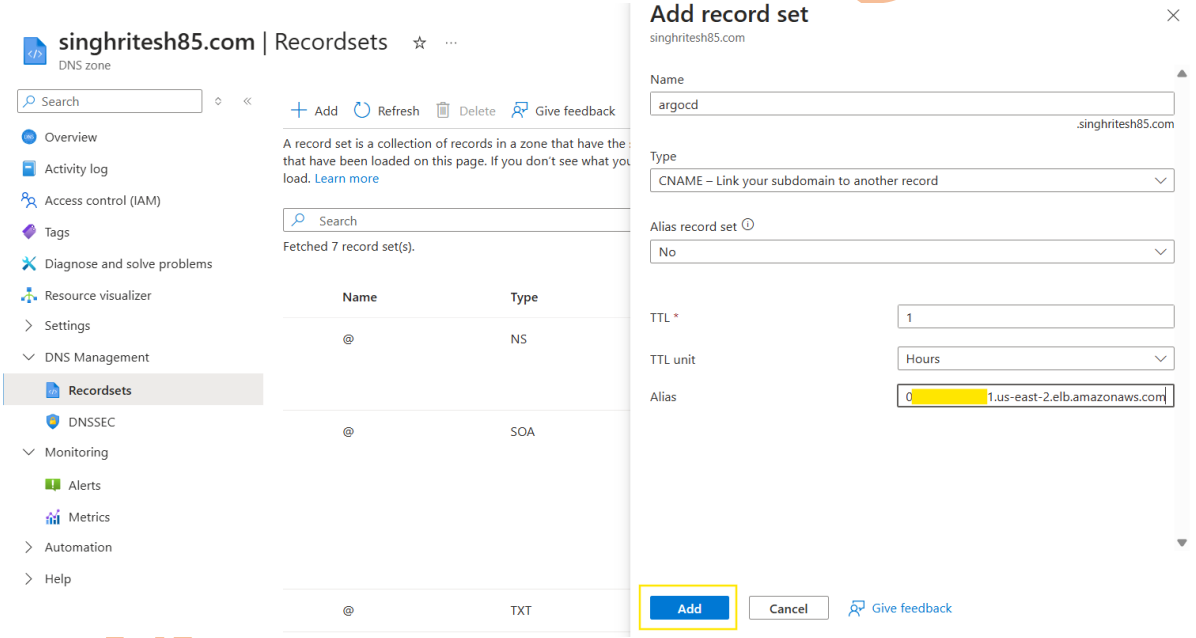
```
[jenkins@                    ~]$ kubectl create namespace argocd
namespace/argocd created
[jenkins@                    ~]$ kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```

The ingress rule for ArgoCD is as shown in the screenshot attached below.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  namespace: argocd
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"    ### You can use this option for this particular case for ArgoCD but not for all
#    nginx.ingress.kubernetes.io/ssl-redirect: "false"
spec:
  ingressClassName: nginx
  rules:
  - host: argocd.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: argocd-server   ### Provide your service Name
            port:
              number: 80     #### Provide your service port for this particular example you can also choose 443
```

```
[jenkins@                ~]$ cat argocd-ingress-rule.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  namespace: argocd
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"    ###  You can use this option for this particular case for ArgoCD but not for all
#    nginx.ingress.kubernetes.io/ssl-redirect: "false"
spec:
  ingressClassName: nginx
  rules:
  - host: argocd.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: argocd-server   ### Provide your service Name
            port:
              number: 80     #### Provide your service port for this particular example you can also choose 443
[jenkins@                ~]$ kubectl apply -f argocd-ingress-rule.yaml
kubecingress.networking.k8s.io/minimal-ingress created
[jenkins@                ~]$ kubectl get ing -n argocd
NAME              CLASS   HOSTS                     ADDRESS   PORTS   AGE
minimal-ingress   nginx   argocd.singhritesh85.com             80      11s
[jenkins@                ~]$ kubectl get ing -n argocd --watch
NAME              CLASS   HOSTS                     ADDRESS   PORTS   AGE
minimal-ingress   nginx   argocd.singhritesh85.com             80      16s
minimal-ingress   nginx   argocd.singhritesh85.com   a                                                1.us-east-2.elb.amazonaws.com   80      33s
```

The entry for DNS Name corresponding to HOST **argocd.singhritesh85.com** in Azure DNS Zone to create the record set is as shown in the screenshot attached below.

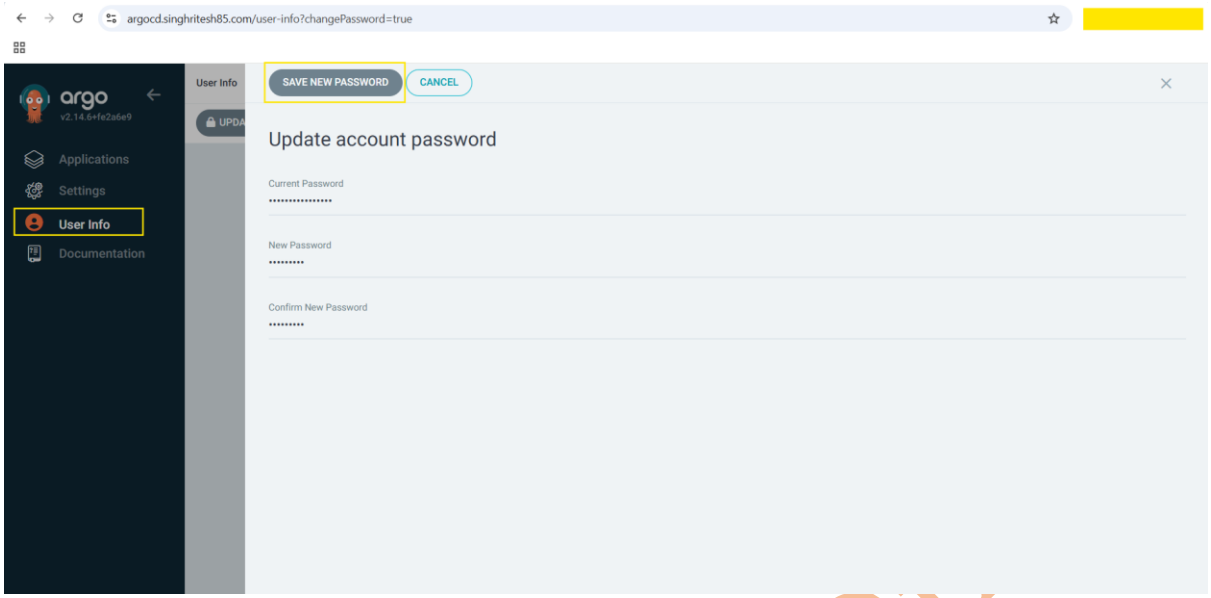


I generated the password for ArgoCD is as shown in the screenshot attached below.

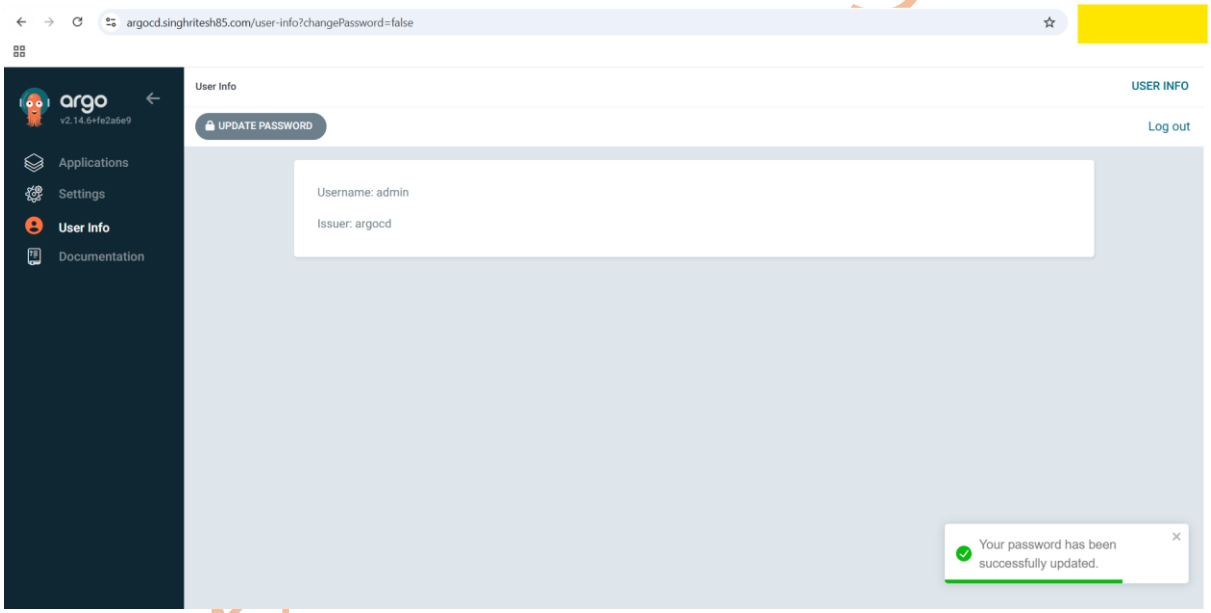
kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath='{.data.password}' | base64 -d

```
[jenkins@                ~]$ kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath='{.data.password}' | base64 -d
U              d[jenkins@ip-10-10-4-202 ~]$
```

After login into the ArgoCD for the first time I updated the ArgoCD Password as shown in the screenshot attached below.



Now the Password was successfully updated as verified by the screenshot attached below.



Installation of ArgoCD CLI is as shown in the screenshot attached below.

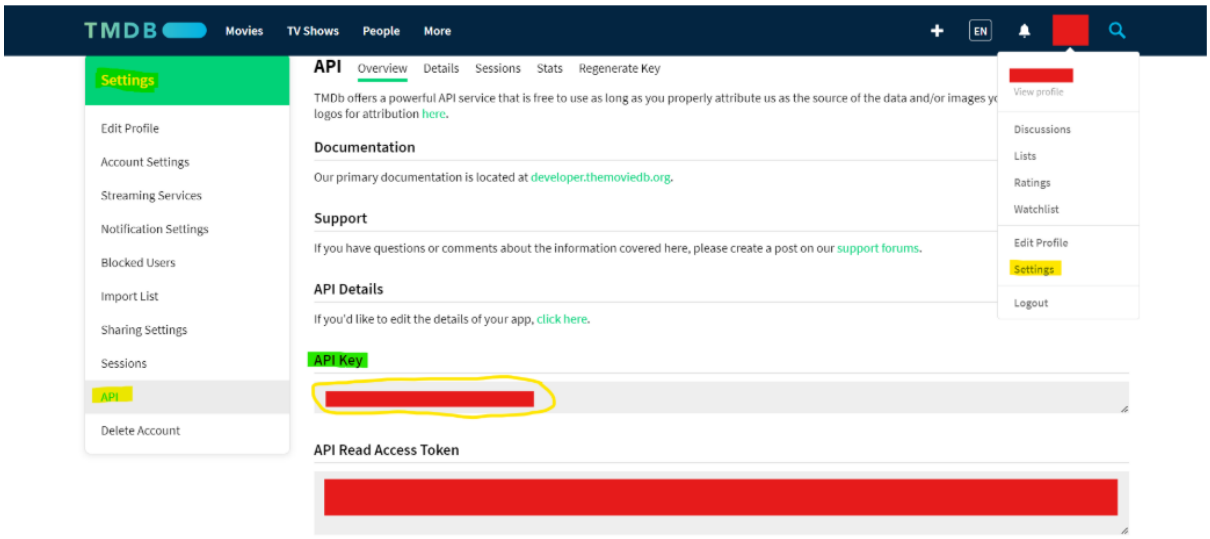
curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64

sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd

rm argocd-linux-amd64

```
[jenkins@                ~]$ curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64
[jenkins@                ~]$ sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd
[jenkins@                ~]$ rm argocd-linux-amd64
```

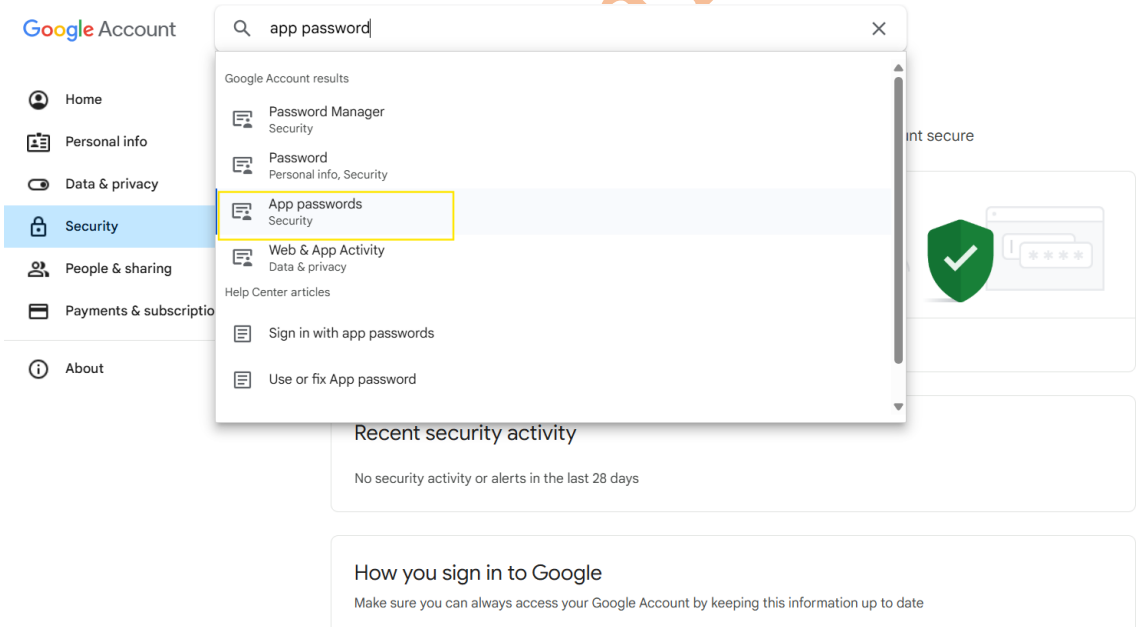
TMDB API Key can used as shown below while building the Docker Image.

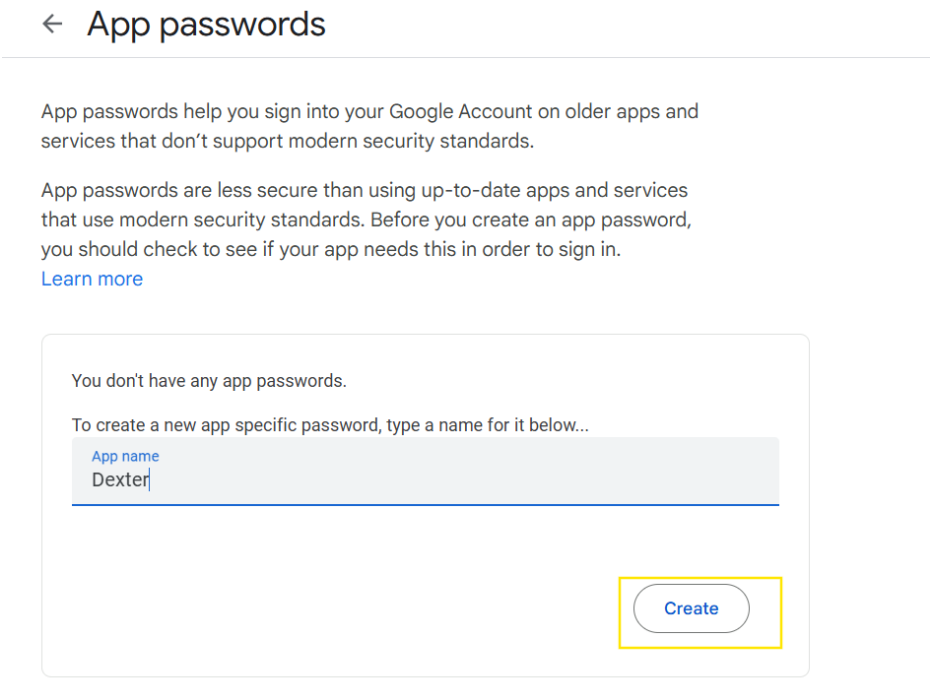


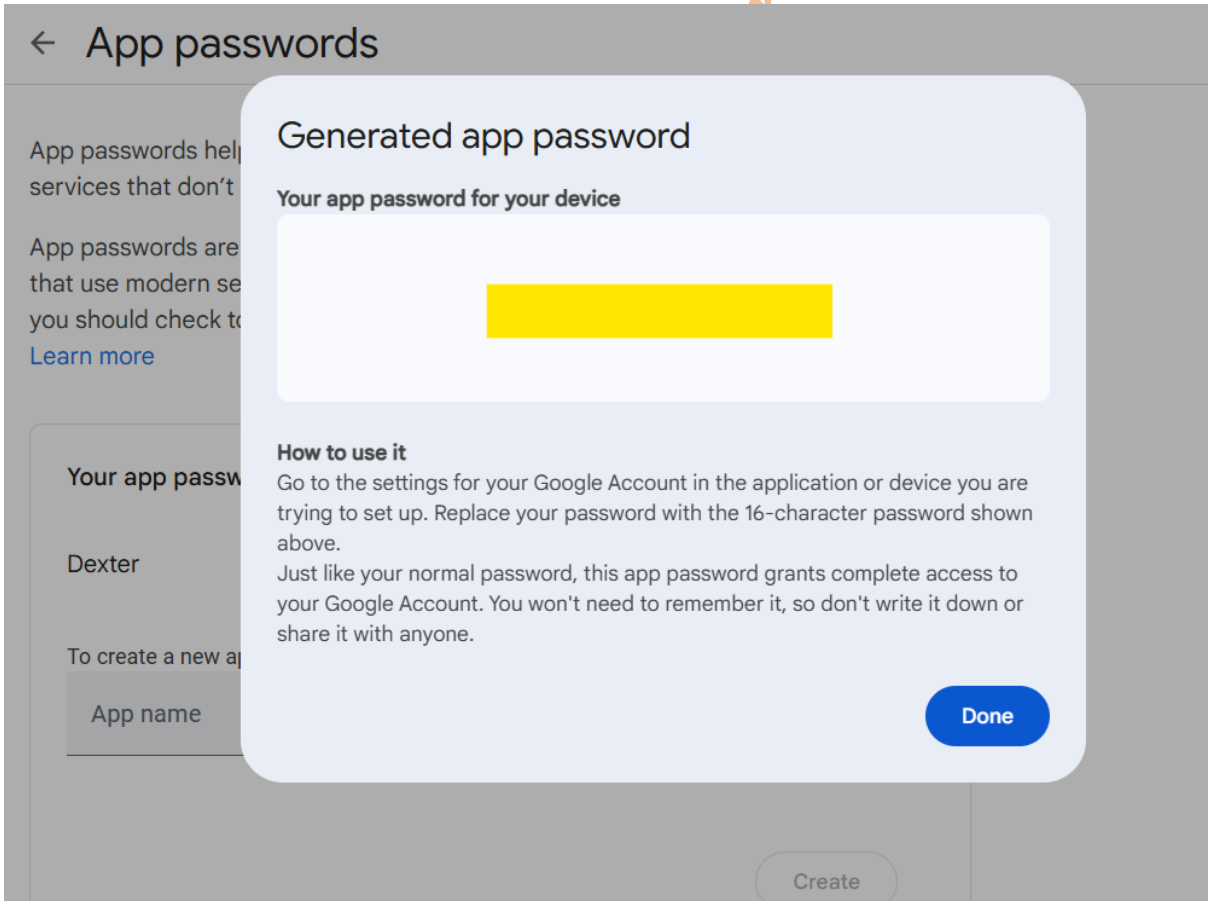
Configuration of Email to Send notification on Group Email-ID using Jenkins and Grafana

To configure Gmail to send notification to group Email ID I should have App Password for my Gmail account as shown in the screenshot attached below.

Go to your Gmail Account > Manage your Google Account > Security and then search for app password and click on App Passwords as shown in the screenshot attached below.

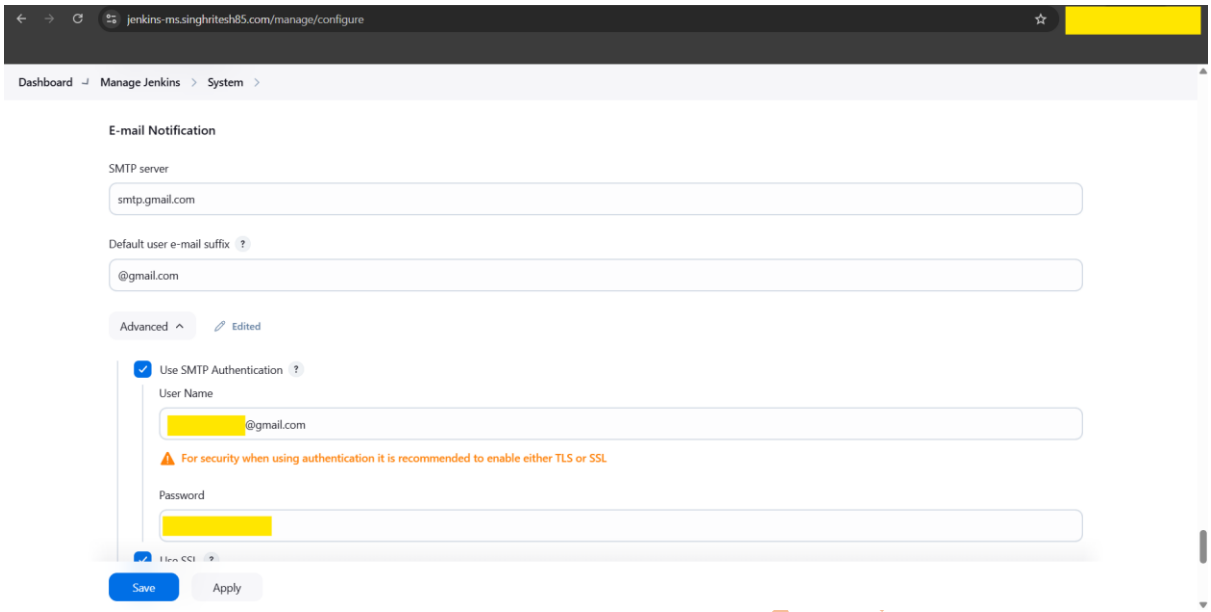


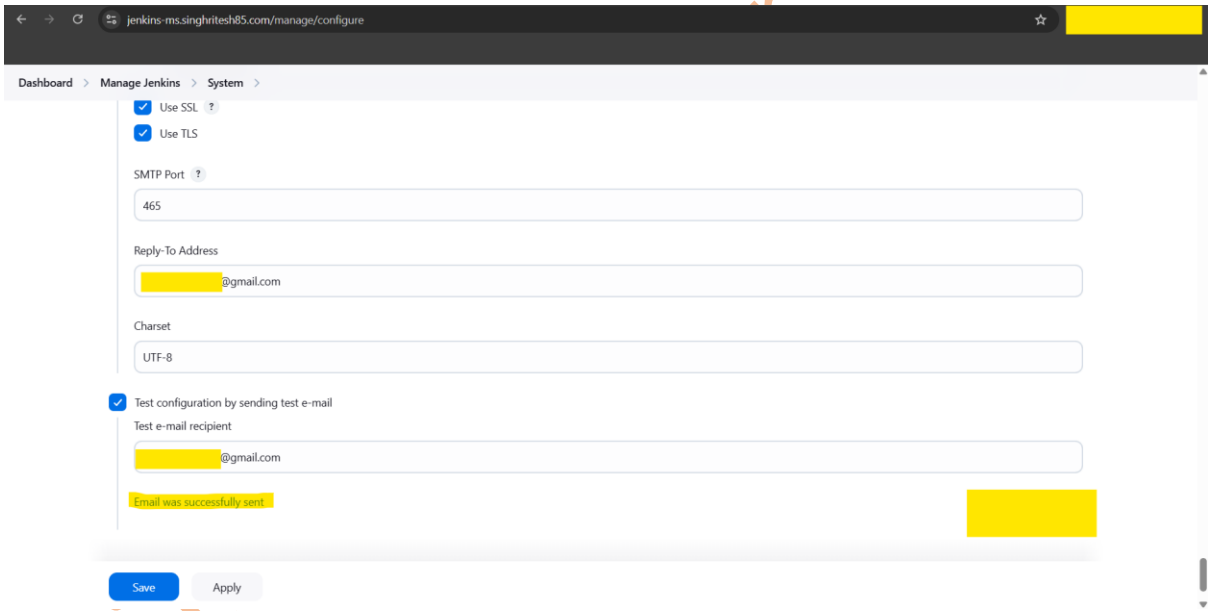




I had deleted this App Password after completion of this project, this App password does not exist anymore.

Now, configuration of email to Send notification on Group Email-ID using Jenkins and Grafana is as discussed below.





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@                    ~]# vim /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 = Netflix Clone
# 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
```

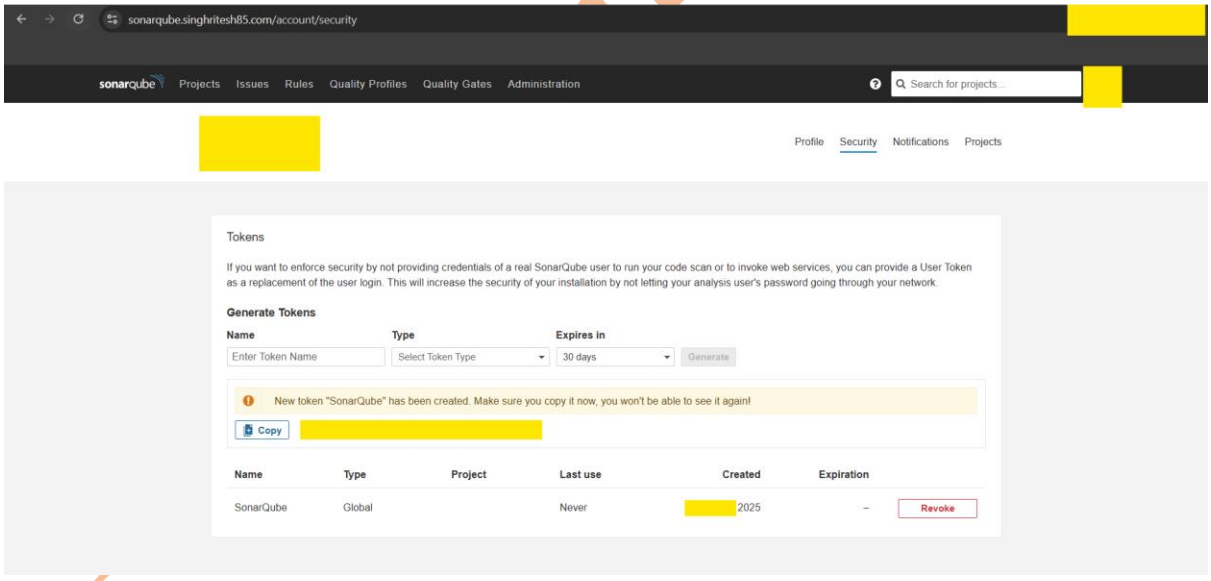
The restart the **grafana-server** service as shown in the screenshot attached below.

```
[root@[redacted] ~]# systemctl restart grafana-server.service
[root@[redacted] ~]# 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 Tue 2025-[redacted]
```

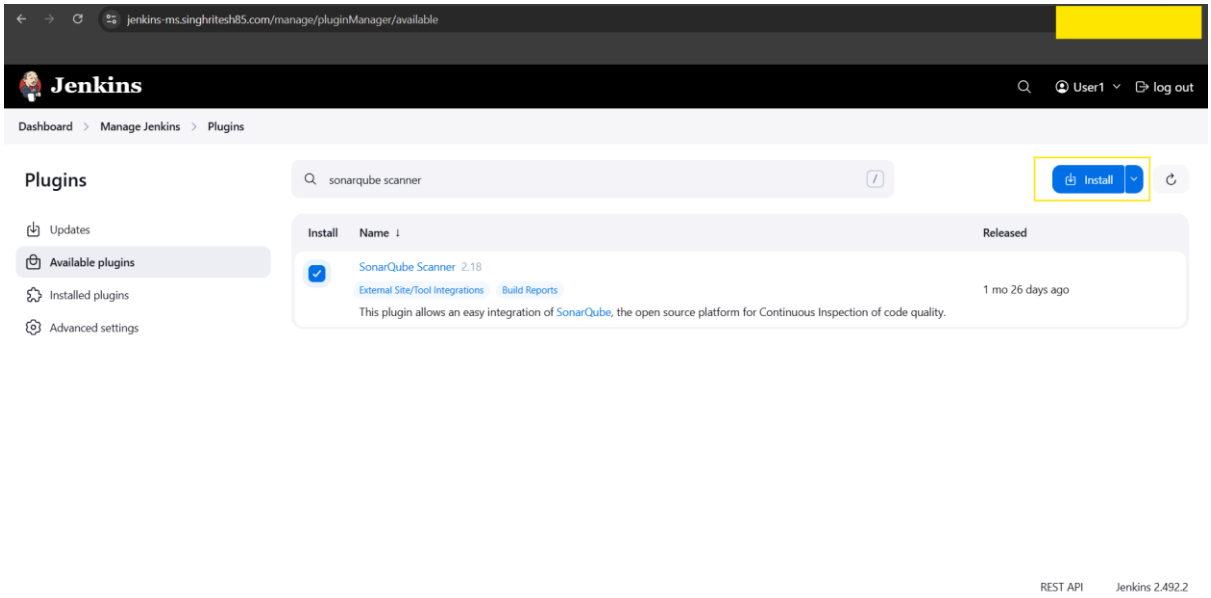
Creation of Alerts in Grafana I will discuss later here I will discuss first integration of Jenkins with SonarQube then will create the Jenkins Job and deploy the Netflix Application. To do so I used the Jenkinsfile as shown in the screenshot attached below. This Jenkinsfile is also available in my GitHub Repo https://github.com/singhritesh85/DevOps-Project-Netflix-Clone-Aws.git.

Integration of Jenkins with SonarQube

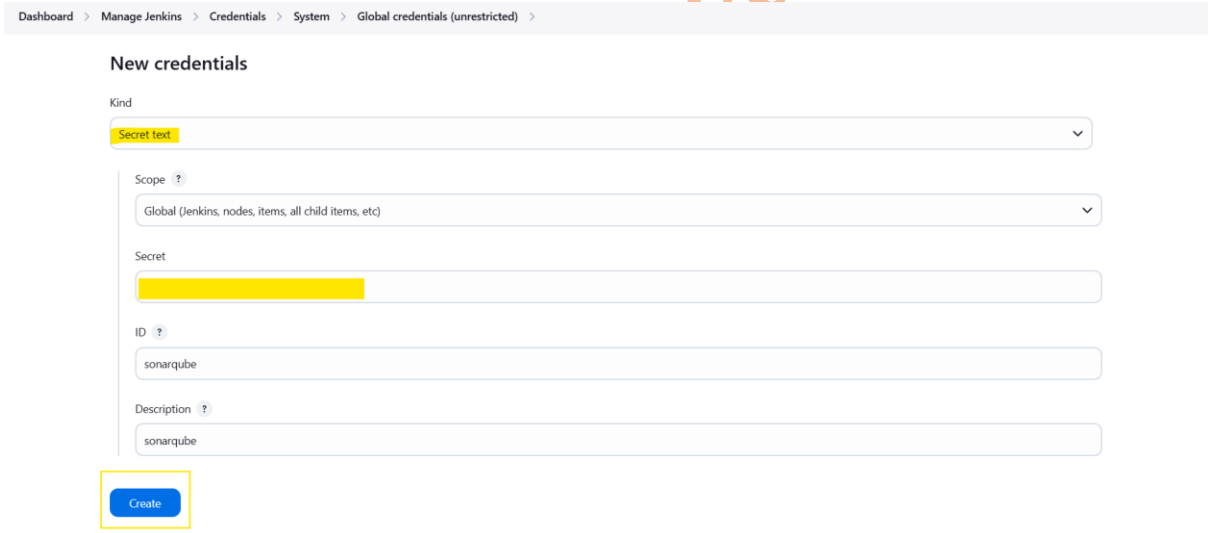
To Integrate Jenkins with SonarQube I need a Security Token in SonarQube which I created as shown in the screenshot attached below.



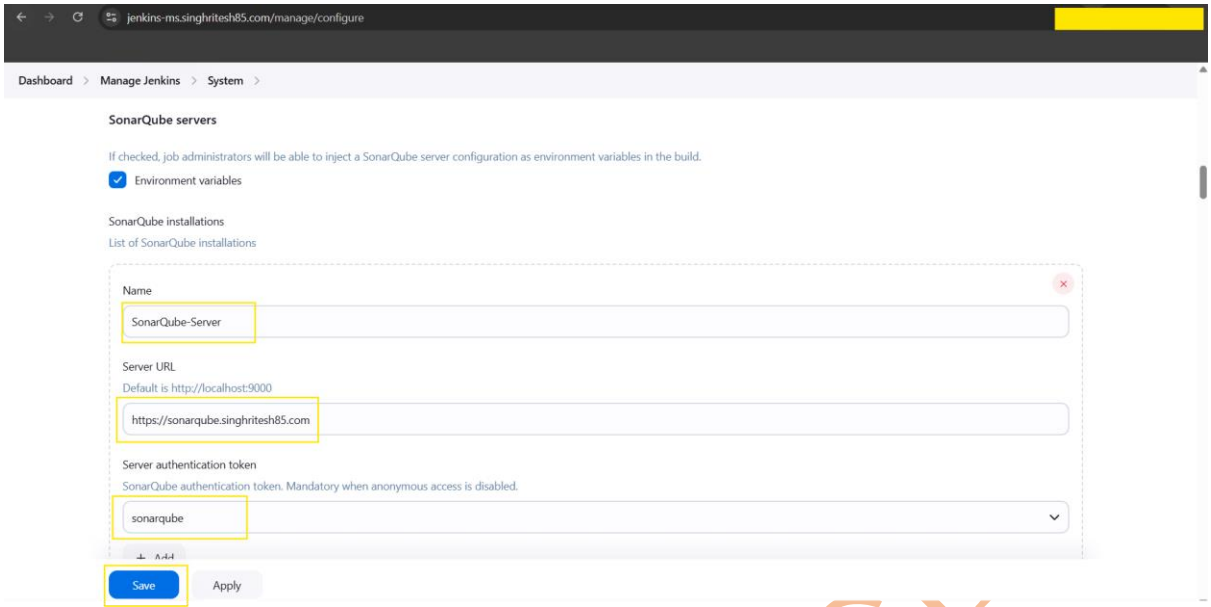
Now go to Jenkins Manage Jenkins > System > Plugins > Available Plugins and install the **SonarQube-Scanner** Plugin as shown in the screenshot attached below.



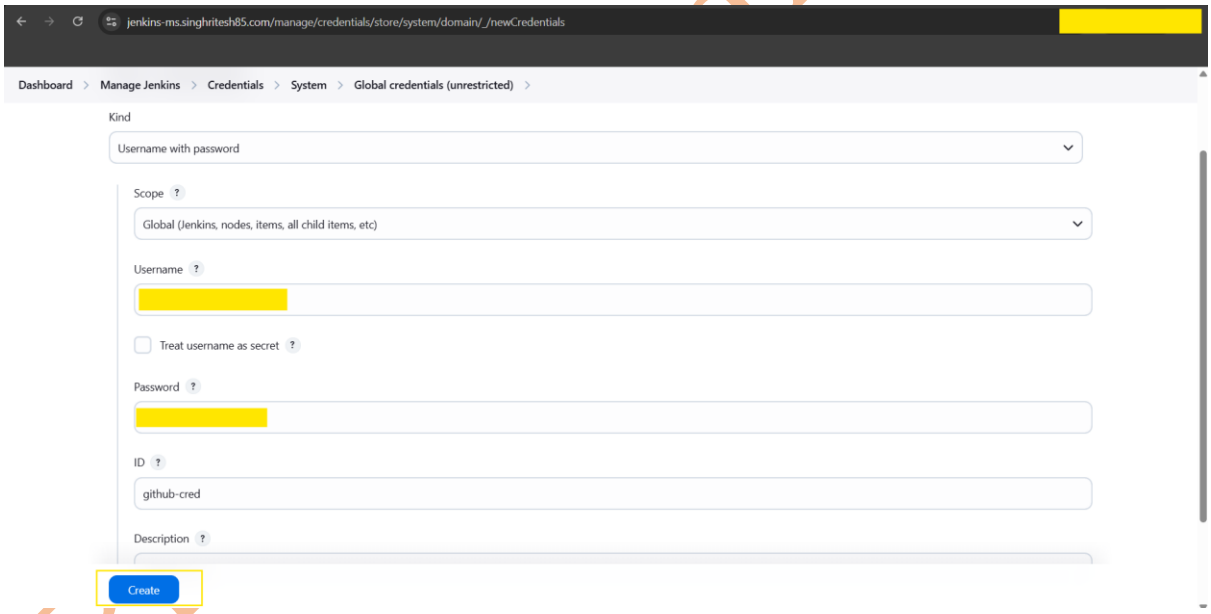
Do not restart the Jenkins after installation of **SonarQube-Scanner** Plugin. Then create a Jenkins Credential of kind Secret text with the SonarQube Secret Token which I created earlier as shown in the screenshot attached below.



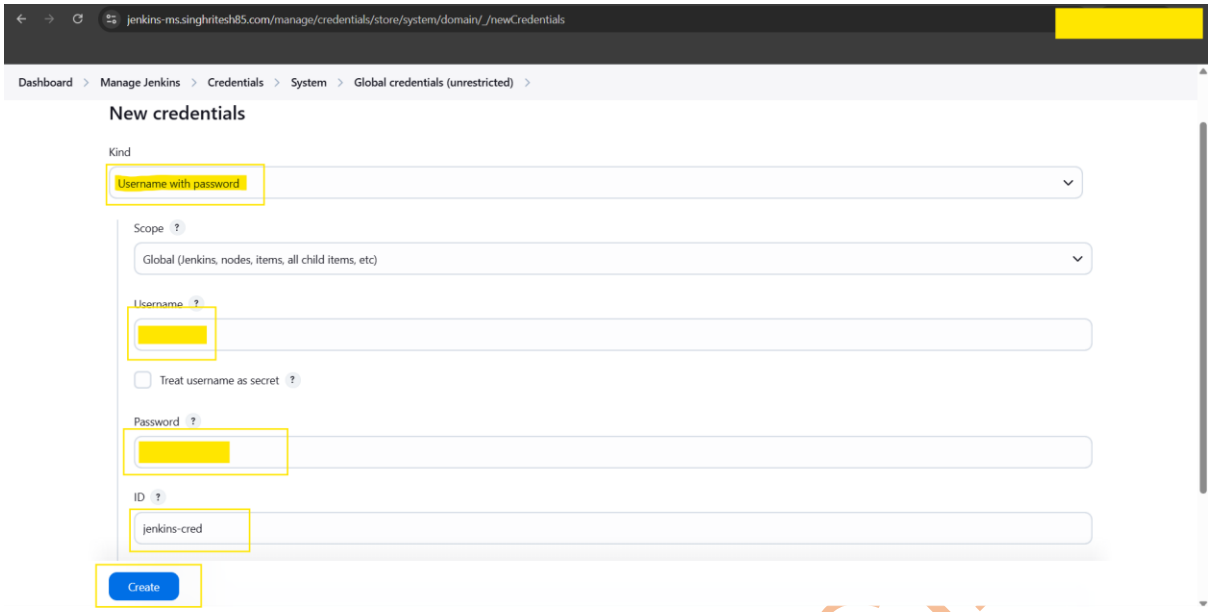
No go to Jenkins Manage Jenkins > System and search for SonarQube and do the configuration as shown in the screenshot attached below.



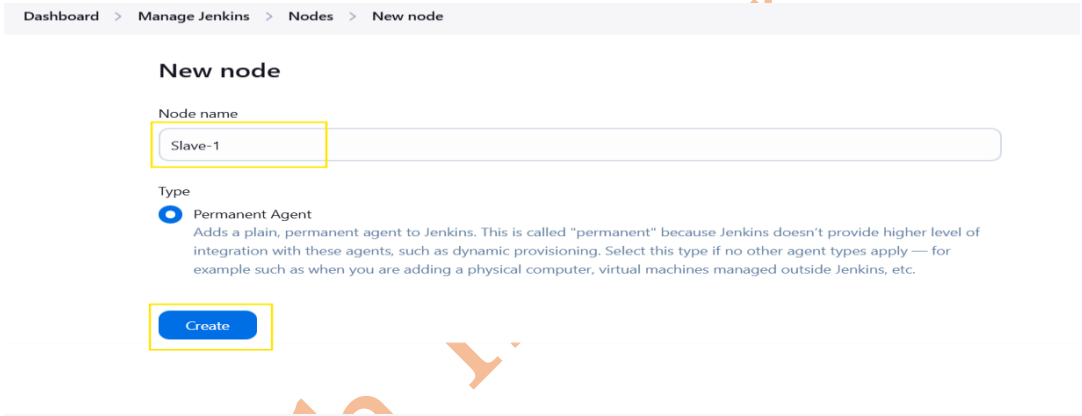
I had created the credential named as github-cred in Jenkins as shown in the screenshot attached below.

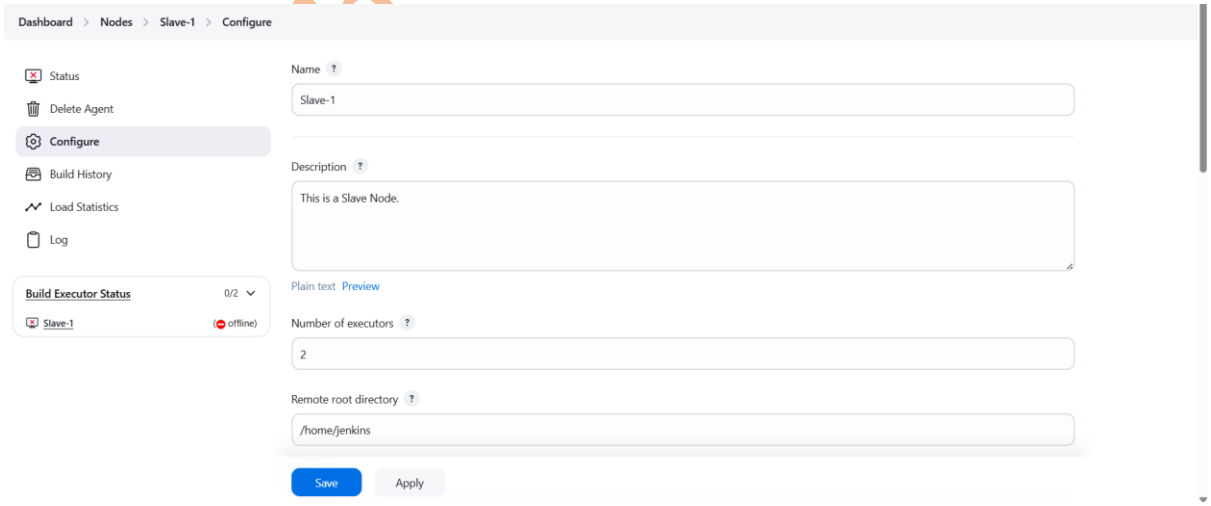


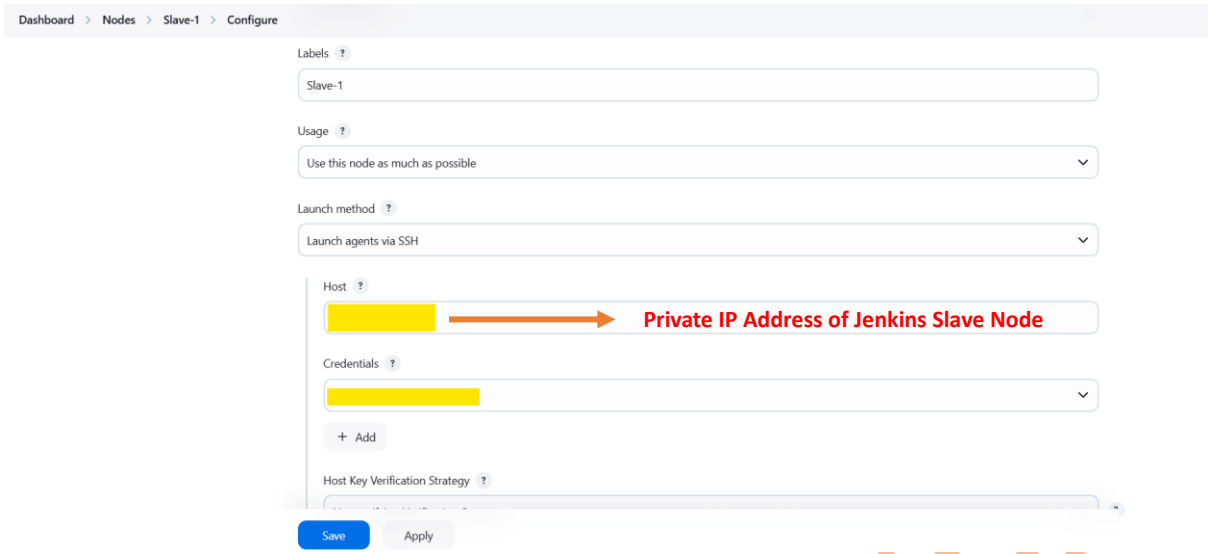
Create a Jenkins Secret to integrate Jenkins Slave Node with Jenkins Master as shown in the screenshot attached below.

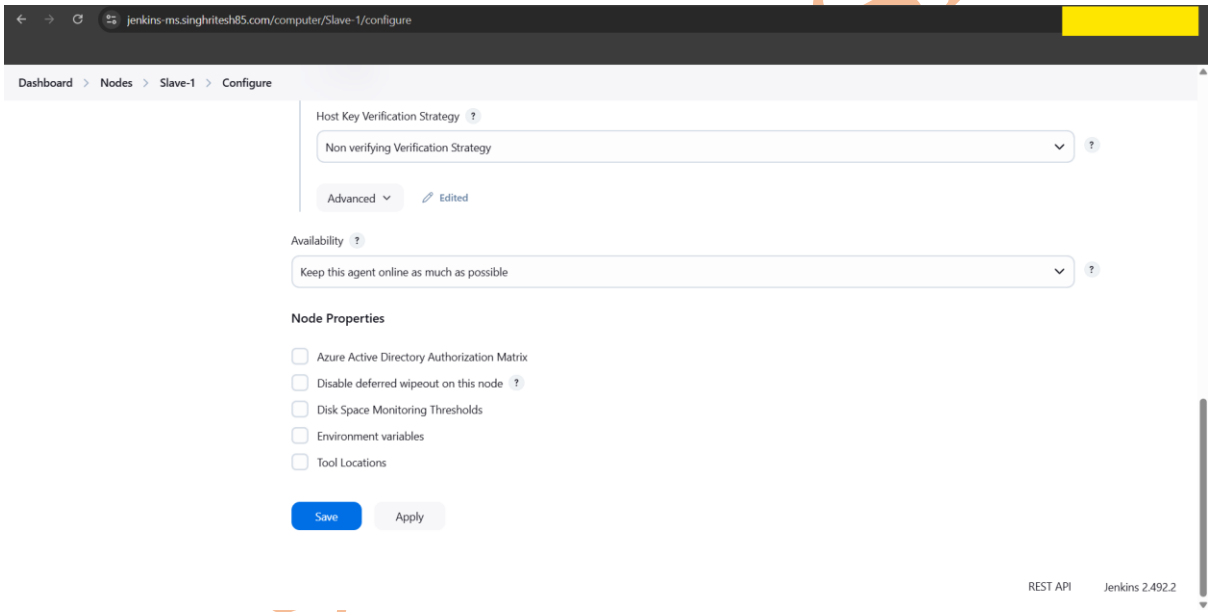


I had integrated a Slave node in Jenkins with the name of **Slave-1**. To do so go to Manage Jenkins > Nodes > New Node as shown in the screenshot attached below.









Now the Slave-1 is in Online mode and I had provided restricted access to the deployment user **jenkins** using Service Account, Role and Role Binding as shown in the screenshot attached below. The deployment user had all the accesses in the namespace **netflix** but does not had access for the entire EKS cluster. That means deployment user jenkins access was restricted to the namespace **netflix** in the EKS Cluster.

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

```
cat sa-role-rolebinding.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: jenkins
  namespace: netflix
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: user-role
  namespace: netflix
rules:
  - apiGroups: ["*"]
    resources: ["*"]
    verbs: ["*"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: user-rolebinding
  namespace: netflix
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: user-role
subjects:
- namespace: netflix
  kind: ServiceAccount
  name: jenkins
```

Created Kubernetes Secrets Which Token was utilized in the kubeconfig file (which was shared with the deployment user jenkins) as shown in the screenshot attached below.

```
[root@ ~]# cat secrets.yaml
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: mysecretname
  namespace: netflix
  annotations:
    kubernetes.io/service-account.name: jenkins
[root@ ~]# kubectl apply -f secrets.yaml
secret/mysecretname created
[root@ ~]# kubectl get secrets -n netflix
NAME           TYPE                                  DATA   AGE
mysecretname   kubernetes.io/service-account-token   3      9s
```

cat secrets.yaml

apiVersion: v1

kind: Secret

type: kubernetes.io/service-account-token

metadata:

name: mysecretname

namespace: netflix

annotations:

kubernetes.io/service-account.name: jenkins

```
[root@ ~]# kubectl describe secret mysecretname -n netflix
Name:         mysecretname
Namespace:    netflix
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: jenkins
              kubernetes.io/service-account.uid: 6                                    b

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1107 bytes
namespace:  7 bytes
token:
```

Below is the screenshot of kubeconfig file which I shared with the deployment user jenkins. The deployment user jenkins will create a directory named as **.kube** and will keep the kubeconfig file at this path as shown in the screenshot attached. The 600 permissions had been provided to the file .kube/config using the command **chmod 600 ~/.kube/config.**

```
[jenkins@                ~]$ cat ~/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:
    server: https://7                              D.y   .us-east-2.eks.amazonaws.com
  name: arn:aws:eks:us-east-2:02          6:cluster/eks-demo-cluster-dev
contexts:
- context:
    cluster: arn:aws:eks:us-east-2:02          5:cluster/eks-demo-cluster-dev
    user: jenkins
  name: dexter
current-context: dexter
kind: Config
preferences: {}
users:
- name: jenkins
  user:
    token:
```

After getting the kubeconfig file the deployment user jenkins checked its access in EKS Cluster and found that they have only the access inside the netflix namespace and not for the entire EKS Cluster as shown in the screenshot attached below.

```
[jenkins@                ~]$ kubectl get nodes
Error from server (Forbidden): nodes is forbidden: User "system:serviceaccount:netflix:jenkins" cannot list resource "nodes" in API group "" at the cluster sc
ope
[jenkins@                ~]$ kubectl get pods -n netflix
No resources found in netflix namespace.
[jenkins@                ~]$
```

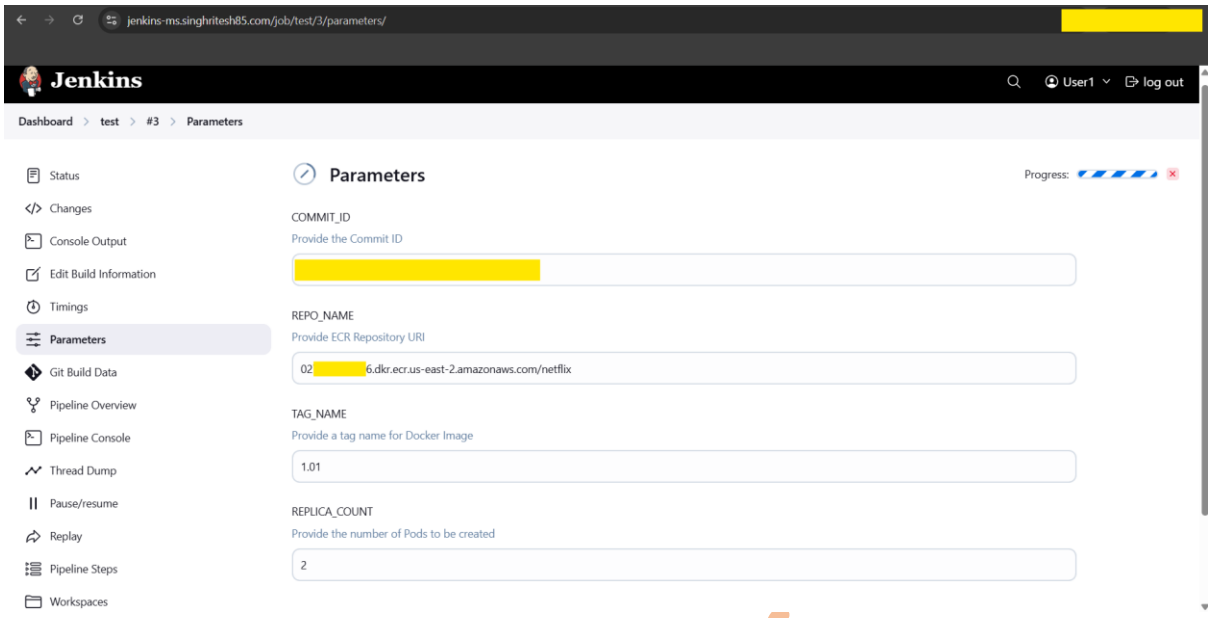
I had executed the Jenkins Job as shown in the screenshot attached below.

```
pipeline{
    agent{
        node{
            label "Slave-1"
            customWorkspace "/home/jenkins/mydemo"
        }
    }
    environment{
        JAVA_HOME="/usr/lib/jvm/java-17-amazon-corretto.x86_64"
        PATH="$PATH:$JAVA_HOME/bin:/opt/sonar-scanner/bin:/opt/dependency-check/bin:/opt/node-v16.0.0/bin"
    }
    parameters {
        string(name: 'COMMIT_ID', defaultValue: '', description: 'Provide the Commit ID')
        string(name: 'REPO_NAME', defaultValue: '', description: 'Provide ECR Repository URI')
        string(name: 'TAG_NAME', defaultValue: '', description: 'Provide a tag name for Docker Image')
        string(name: 'REPLICA_COUNT', defaultValue: '', description: 'Provide the number of Pods to be created')
    }
    stages{
        stage("clone-code"){
            steps{
                cleanWs()
                checkout scmGit(branches: [[name: "${COMMIT_ID}"]], extensions: [], userRemoteConfigs: [[credentialsId: 'github-cred', url: 'https://github.com/singhritesh85/DevSecOps-Project.git']])
            }
        }
        stage("SonarAnalysis"){
            steps {
                withSonarQubeEnv('SonarQube-Server') {
                    sh 'sonar-scanner -Dsonar.projectKey=netflix-clone -Dsonar.projectName=netflix-clone'
                }
```

```
        }
    }
    stage("Quality Gate") {
        steps {
         timeout(time: 1, unit: 'HOURS') {
            waitForQualityGate abortPipeline: true
         }
        }
    }
    stage("Install Dependencies"){
        steps {
            sh 'npm install'
        }
    }
    stage("OWASP Dependency Check"){
        steps{
            sh 'dependency-check.sh --disableYarnAudit --disableNodeAudit --scan . --out .'
        }
    }
    stage("Trivy Scan files"){
        steps{
            sh 'trivy fs . > /home/jenkins/trivy-filescan.txt'
        }
    }
    stage("Docker-Image"){
        steps{
            sh 'docker build --build-arg TMDB_V3_API_KEY=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX -t myimage:1.06 . --no-cache'
            sh 'docker tag myimage:1.06 ${REPO_NAME}:${TAG_NAME}'
            sh 'trivy image --exit-code 0 --severity MEDIUM,HIGH ${REPO_NAME}:${TAG_NAME}'
            sh 'aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 027330342406.dkr.ecr.us-east-2.amazonaws.com'
```

```
            sh 'docker push ${REPO_NAME}:${TAG_NAME}'
        }
      }
      stage("Deployment"){
        steps{
          //sh 'yes|argocd login argocd.singhritesh85.com --username admin --password Admin@123'
          sh 'argocd login argocd.singhritesh85.com --username admin --password Admin@123 --skip-test-tls --grpc-web'
          sh 'argocd app create netflix-clone --project default --repo https://github.com/singhritesh85/helm-repo-for-netflix-clone.git --path ./folo --namespace netflix --sync-option CreateNamespace=true --dest-server https://kubernetes.default.svc --helm-set service.port=80 --helm-set image.repository=${REPO_NAME} --helm-set image.tag=${TAG_NAME} --helm-set replicaCount=${REPLICA_COUNT} --upsert'
          sh 'argocd app sync netflix-clone'
        }
      }
    }
    post {
      always {
        mail bcc: '', body: "A Jenkins Job with Job Name ${env.JOB_NAME} has been executed", cc: '', from: '', replyTo: '', subject: "Jenkins Job ${env.JOB_NAME} has been executed", to: 'abc@gmail.com'
      }
      success {
        mail bcc: '', body: "A Jenkins Job with Job Name ${env.JOB_NAME} and Build Number=${env.BUILD_NUMBER} has been executed Successfully, Please Open the URL ${env.BUILD_URL} and click on Console Output to see the Log. The Result of execution is ${currentBuild.currentResult}", cc: '', from: '', replyTo: '', subject: "Jenkins Job ${env.JOB_NAME} has been Sucessfully Executed", to: 'abc@gmail.com'
      }
      failure {
        mail bcc: '', body: "A Jenkins Job with Job Name ${env.JOB_NAME} and Build Number=${env.BUILD_NUMBER} has been Failed, Please Open the URL ${env.BUILD_URL} and click on Console Output to see the Log. The Result of execution is ${currentBuild.currentResult}", cc: '', from: '', replyTo: '', subject: "Jenkins Job ${env.JOB_NAME} has been Failed", to: 'abc@gmail.com'
      }
    }
}
```

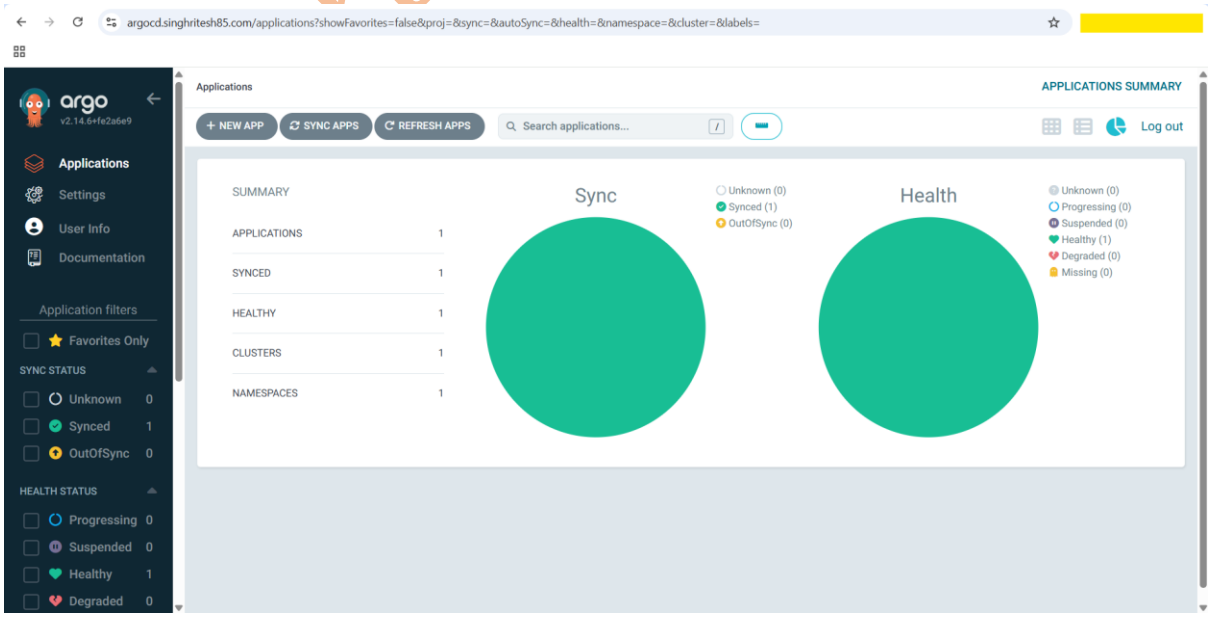
To run the Jenkins Job, we need to provide some parameters (String Parameters in Jenkins) as shown in the screenshot attached below.



Before Running the Jenkins Job change the Nameserver on Jenkins Slave node as shown in the screenshot attached below. Because for this project the DHCP Option which I am using is the default DHCP Option Set.

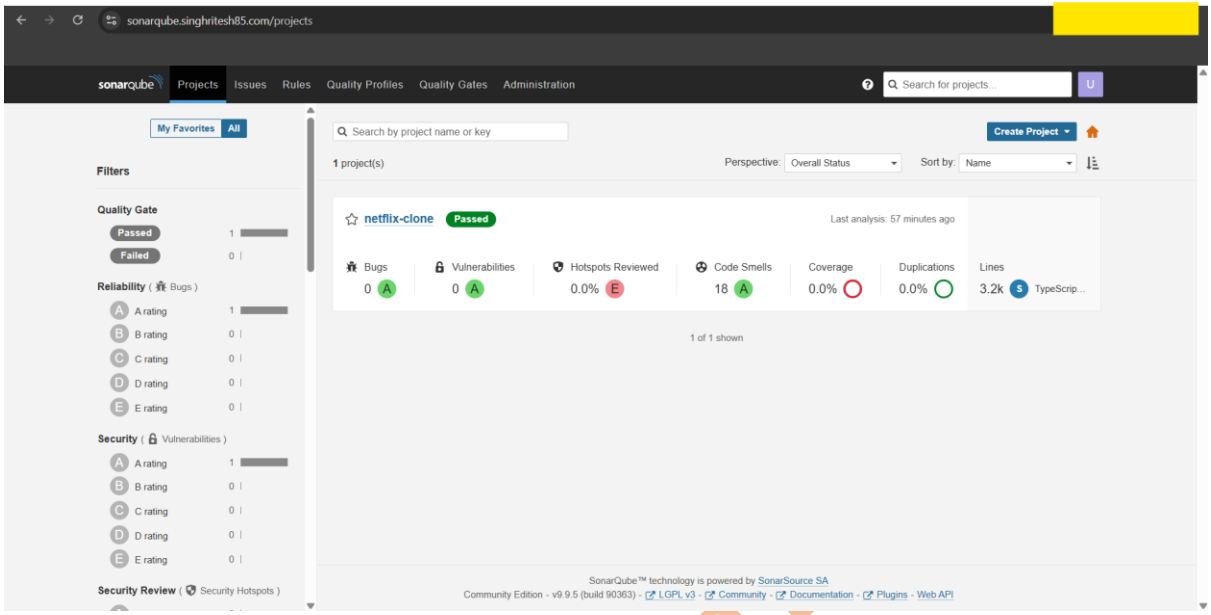
```
[root@                ~]# cat /etc/resolv.conf
; generated by /usr/sbin/dhclient-script
search us-east-2.compute.internal
options timeout:2 attempts:5
nameserver 8.8.8.8      #10.10.0.2
```

After successful execution of Jenkins Job the screenshot for ArgoCD is as shown below.



After successfully running the Jenkins Job, Pods had been created and the SonarQube UI are as shown in the screenshot attached below.

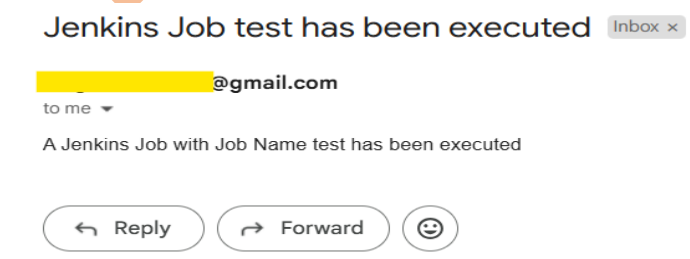
```
[jenkins@                    ~]$ kubectl get pods -n netflix
NAME                                  READY   STATUS    RESTARTS   AGE
netflix-clone-folo-5                  1/1     Running   0          32s
netflix-clone-folo-5                  1/1     Running   0          32s
```

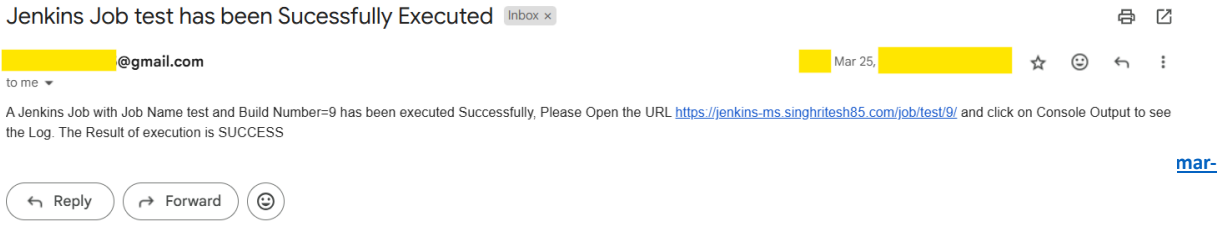


To access the Netflix Clone Application, I created the ingress rule as discussed below.

```
[jenkins@                    ~]$ cat ingress-rule.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: netflix-ingress
  namespace: netflix
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  rules:
  - host: netflix-clone.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: netflix-clone-folo
            port:
              number: 80
[jenkins@                    ~]$ kubectl apply -f ingress-rule.yaml
ingress.networking.k8s.io/netflix-ingress created
[jenkins@                    ~]$ kubectl get ing -n netflix
NAME              CLASS   HOSTS                          ADDRESS                                                    PORTS   AGE
netflix-ingress   nginx   netflix-clone.singhritesh85.com   a                                1.us-east-2.elb.amazonaws.com   80      12s
```

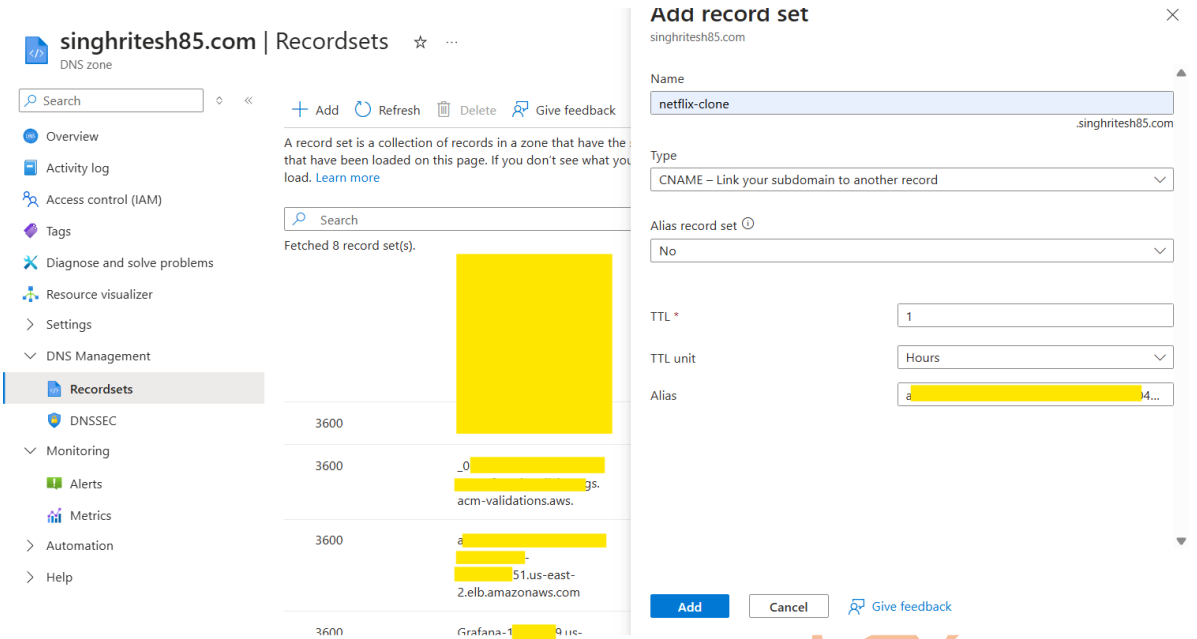
After Successful execution of Jenkins Job, I received two email one regarding a Jenkins Job had been executed and another Jenkins job had been executed successfully as shown in the screenshot attached below.



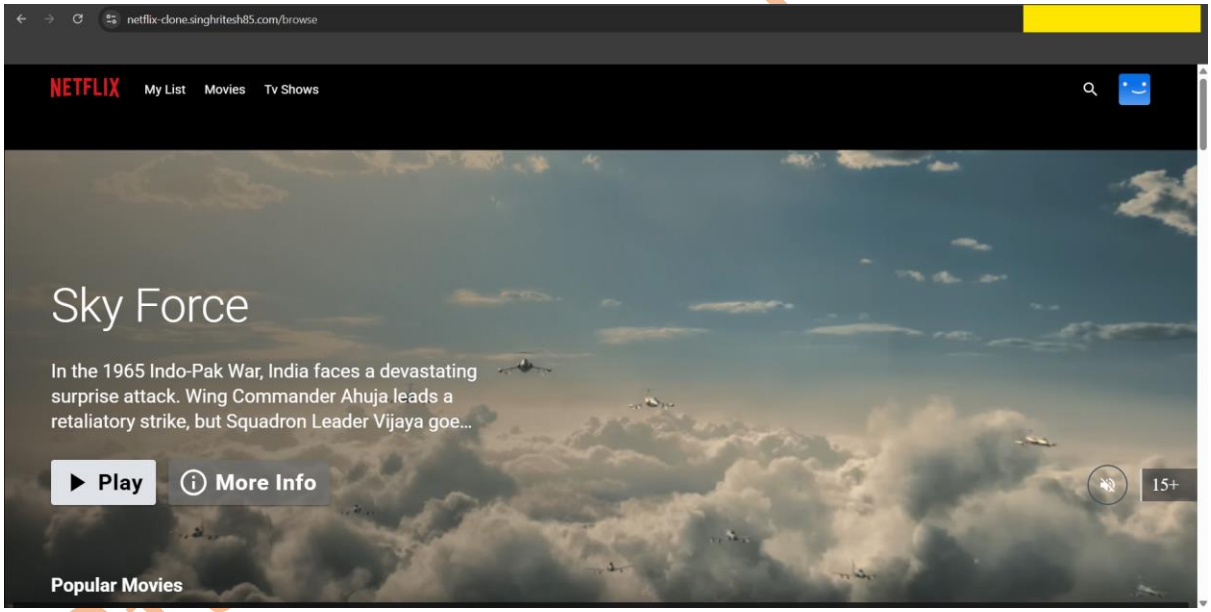


```
cat ingress-rule.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: netflix-ingress
  namespace: netflix
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  rules:
  - host: netflix-clone.singhritesh85.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: netflix-clone-folo
            port:
              number: 80
```

To access the Netflix clone Application through URL I did the entry of HOST of ingress corresponding to the DNS Name in Azure DNS Zone to create the record set of Type CNAME is as shown in the screenshot attached below.



Finally, I accessed the Netflix Clone application through the URL as shown in the screenshot attached below.



Installation of node-exporter and promtail had been done using the helm chart in the EKS Cluster as written 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

```
[root@                ~]# helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

```
[root@                    ~]# kubectl create ns node-exporter
```

```
[root@                ~]# helm install my-prometheus-node-exporter prometheus-community/prometheus-node-exporter --version 4.37.1 --set service.type=LoadBalanc
er -n node-exporter
```

Below screenshot shows the Kubernetes Service which was created for node-exporter using the helm chart.

```
[root@                ~]# kubectl get svc -n node-exporter
NAME                          TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)          AGE
my-prometheus-node-exporter   LoadBalancer   172.    .142    a                                      7.us-east-2.elb.amazonaws.com   9100:30804/TCP   2m19s
```

The updated prometheus configuration file **/etc/prometheus/prometheus.yml** is as shown in the screenshot attached below.

```
- job_name: "EKS"
  static_configs:
    - targets: ["a                                      3.us-east-2.elb.amazonaws.com:9100"]
```

Then restarted the prometheus service as 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 2025-
```

I installed the promtail using the helm chart as written below. First, I cloned the helm chart present in GitHub Repo.

git clone https://github.com/singhritesh85/helm-chart-promtail.git

After cloning helm chart from GitHub, I updated the values.yaml file of promtail helm chart with Loki Servers Private IP Addresses as shown in the screenshot attached below.

kubectl create ns promtail && helm upgrade --install promtail ./helm-chart-promtail -f ./helm-chart-promtail/values.yaml -n promtail

kubectl get pods -n promtail --watch

```
# -- The log level of the Promtail server
# Must be reference in `config.file` to configure `server.log_level`
# See default config in `values.yaml`
logLevel: info
# -- The log format of the Promtail server
# Must be reference in `config.file` to configure `server.log_format`
# Valid formats: `logfmt, json`
# See default config in `values.yaml`
logFormat: logfmt
# -- The port of the Promtail server
# Must be reference in `config.file` to configure `server.http_listen_port`
# See default config in `values.yaml`
serverPort: 3101
# -- The config of clients of the Promtail server
# Must be reference in `config.file` to configure `clients`
# @default -- See `values.yaml`
clients:
  - url: http://10.        :3100/loki/api/v1/push
  - url: http://10.        :3100/loki/api/v1/push
  - url: http://10.        :3100/loki/api/v1/push

# -- Configures where Promtail will save it's positions file, to resume reading after restarts.
# Must be referenced in `config.file` to configure `positions`
positions:
  filename: /run/promtail/positions.yaml
# -- The config to enable tracing
enableTracing: false
# -- A section of reusable snippets that can be reference in `config.file`.
# Custom snippets may be added in order to reduce redundancy.
```

```
[root@                ~]# git clone https://github.com/singhritesh85/helm-chart-promtail.git
```

```
[root@i           ~]# kubectl create ns promtail && helm upgrade --install promtail ./helm-chart-promtail -f ./helm-chart-promtail/values.yaml -n promtail
```

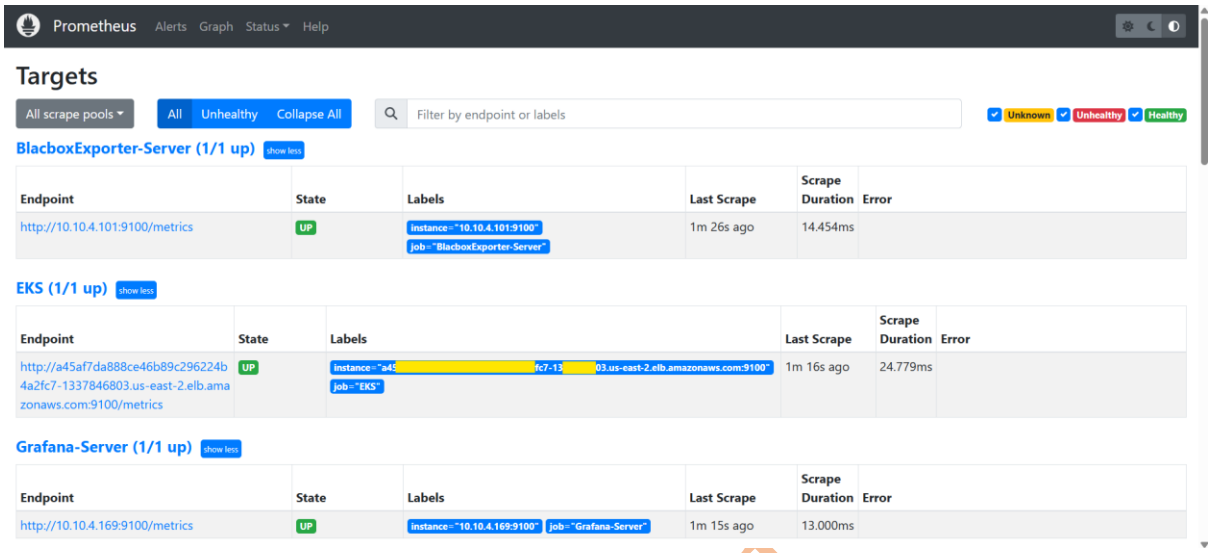
```
[root@                ~]# kubectl get pods -n promtail --watch
NAME                 READY   STATUS    RESTARTS   AGE
promtail-l       f   1/1     Running   0          2m11s
promtail-q       9   1/1     Running   0          2m11s
```

Monitoring Using Prometheus and Grafana and Log Aggregation using Loki

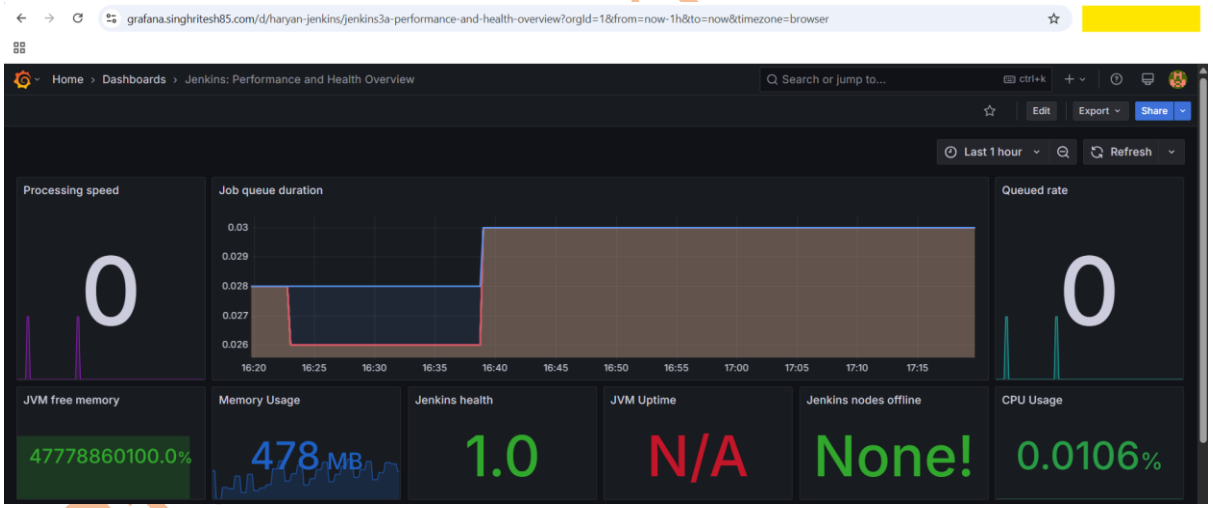
For Monitoring Tool I had used Prometheus and Grafana. To monitor SonarQube I had used SonarQube-Prometheus-Exporter which was installed using terraform at the path **/opt/sonarqube/extensions/plugins**. It was downloaded from the link https://github.com/dmeiners88/sonarqube-prometheus-exporter/releases/download/v1.0.0-SNAPSHOT-2018-07-04/sonar-prometheus-exporter-1.0.0-SNAPSHOT.jar. These steps had been covered in the terraform **user_data_sonarqube.sh**. It is basically a bootstrap script for SonarQube Server. For Monitoring Jenkins, you need to install the plugin Prometheus metrics and then restart Jenkins, these steps already been discussed at the starting. The configuration for prometheus had already been done in the terraform. I had taken sonarqube username and password as **admin** and **Admin123** respectively, you can choose as of your own choice and update the terraform script accordingly (Prometheus needs username and password to extract the metrics from SonarQube). I

had provided the terraform script with this GitHub Repository. I had already integrated Prometheus and Loki as a data source for Grafana which was already discussed above.

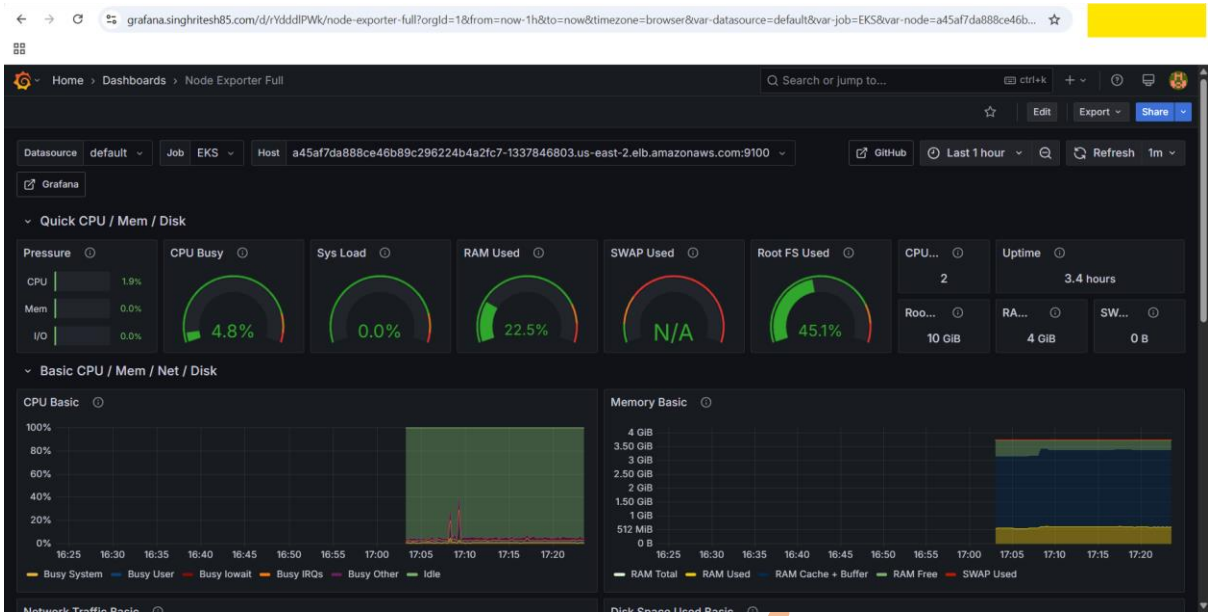
Here I checked the prometheus console and I found all the Targets was UP as shown in the screenshot attached below.



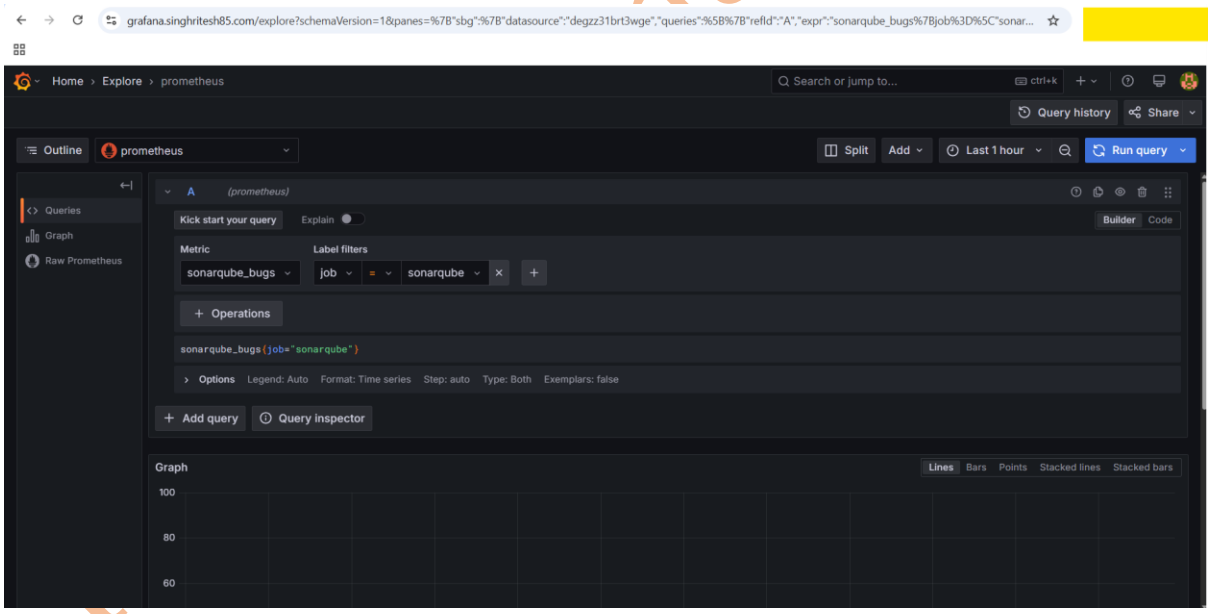
For Monitoring Jenkins Job using Prometheus I created the Grafana Dashboard using the Grafana ID **9964**.

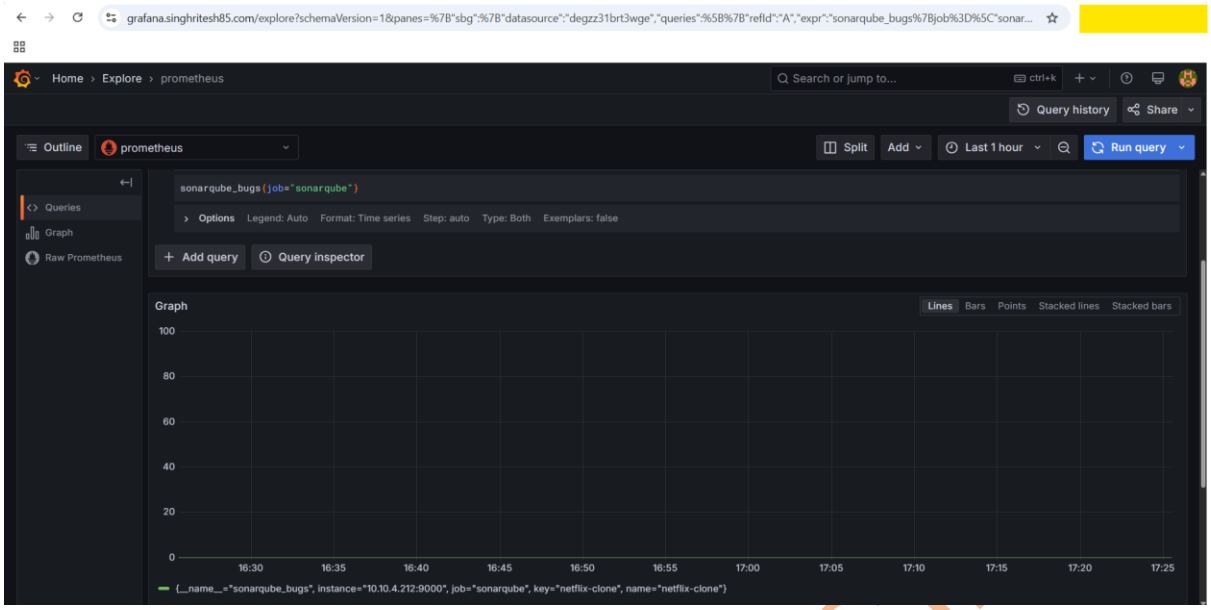


For Monitoring all the Servers and EKS Cluster health using the Node Exporter I used Grafana ID **1860**.

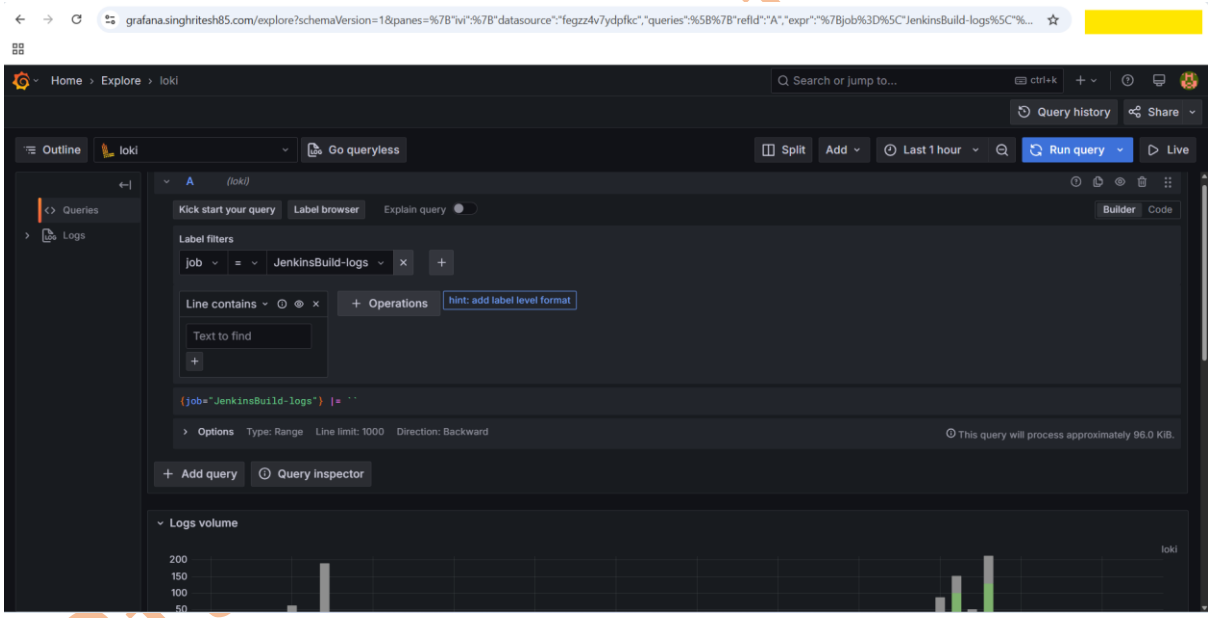


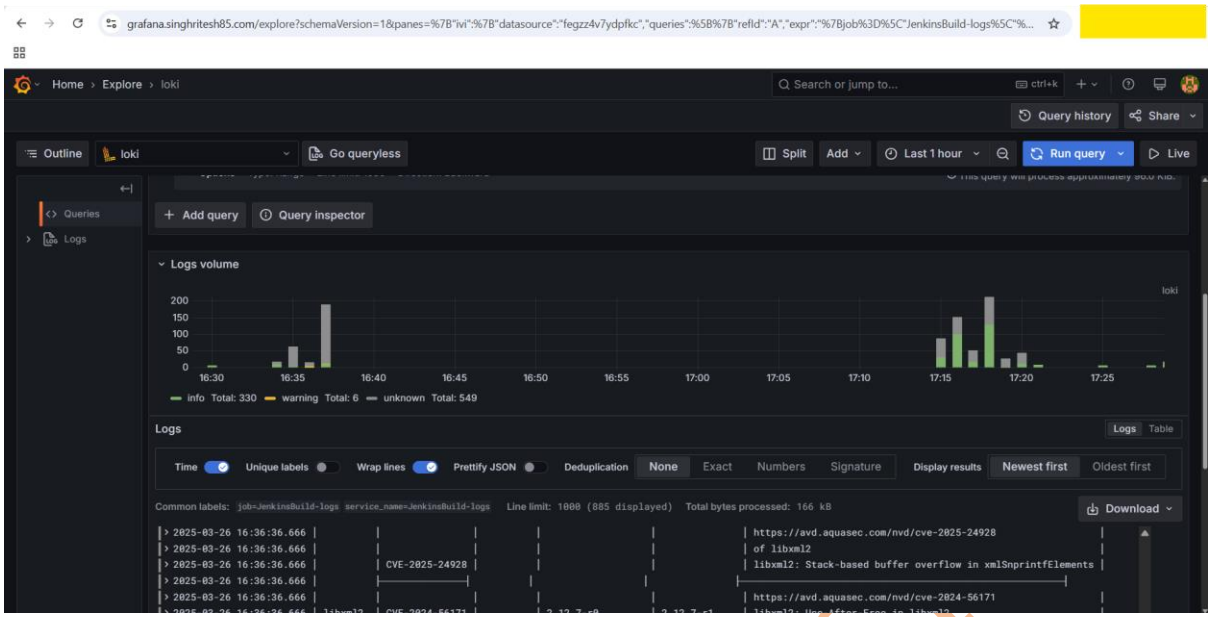
Grafana Metrics I started exploring as shown in the screenshot attached below.





Logs using Loki through Grafana I started exploring as shown in the screenshot attached below.





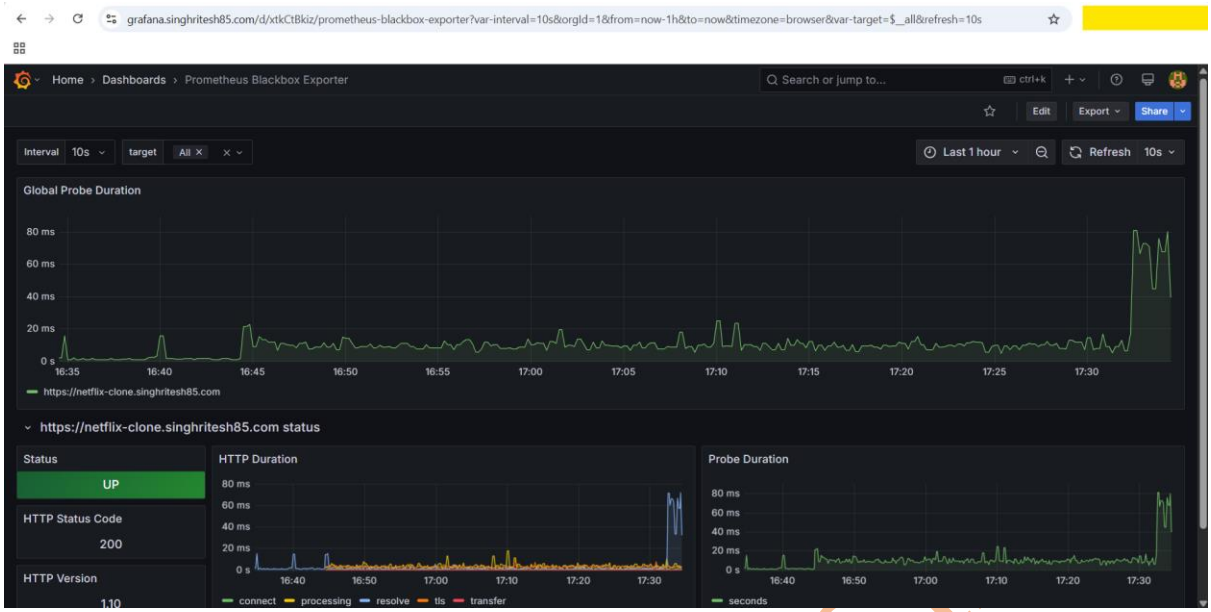
To achieve synthetic monitoring using Prometheus Blackbox Exporter I updated the **/etc/resolv.conf** file for Blackbox Exporter Server as shown in the screenshot attached below. I had used Google's Public DNS Server which is shown in the attached screenshot below.

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

Finally, I was able to perform the synthetics monitoring on the Netflix Clone Application URL as shown in the screenshot attached below. Application URL **https://netflix-clone.singhritesh85.com** had been monitored using blackbox exporter.

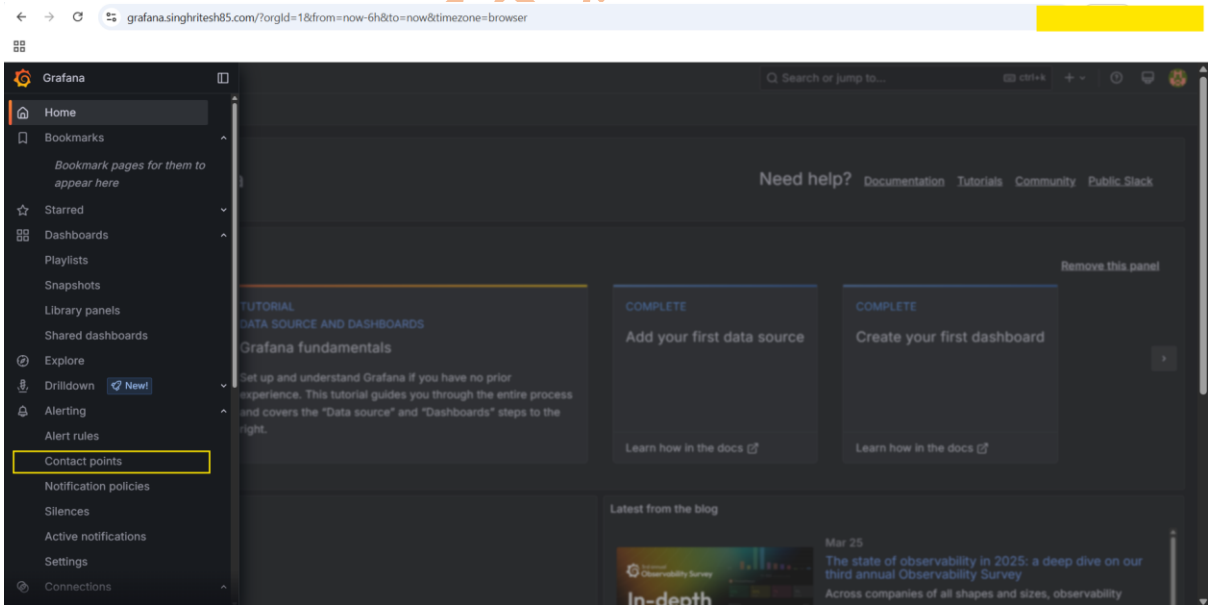
I had installed Blackbox Exporter on a different server and not on the Prometheus Server. **The module name** is monitor_website.yml present of the blackbox exporter server at the path (/opt/blackbox_exporter_linux_amd64/monitor_website.yml). Prometheus blackbox operator is used for endpoint monitoring (Synthetic Monitoring) across the protocol http, https, TCP and ICMP. In this project I am monitoring the Application URL **https://netflix-clone.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.

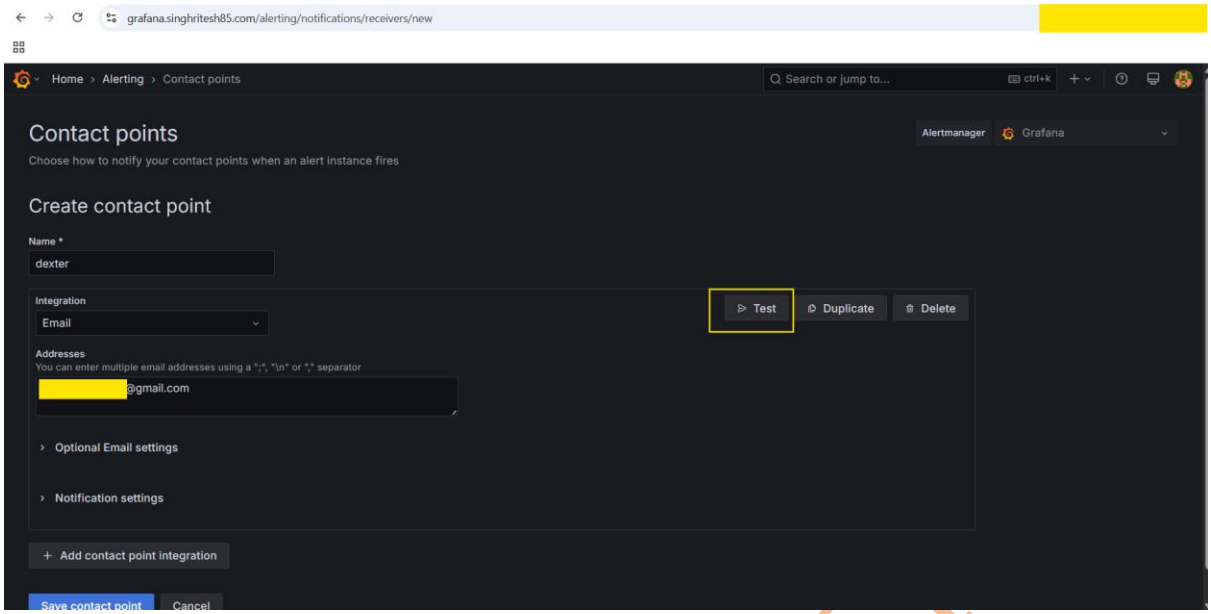
To create the Grafana Dashboard for Application URL Monitoring using blackbox exporter I had used the Grafana ID **7587** and below is the created Dashboard.

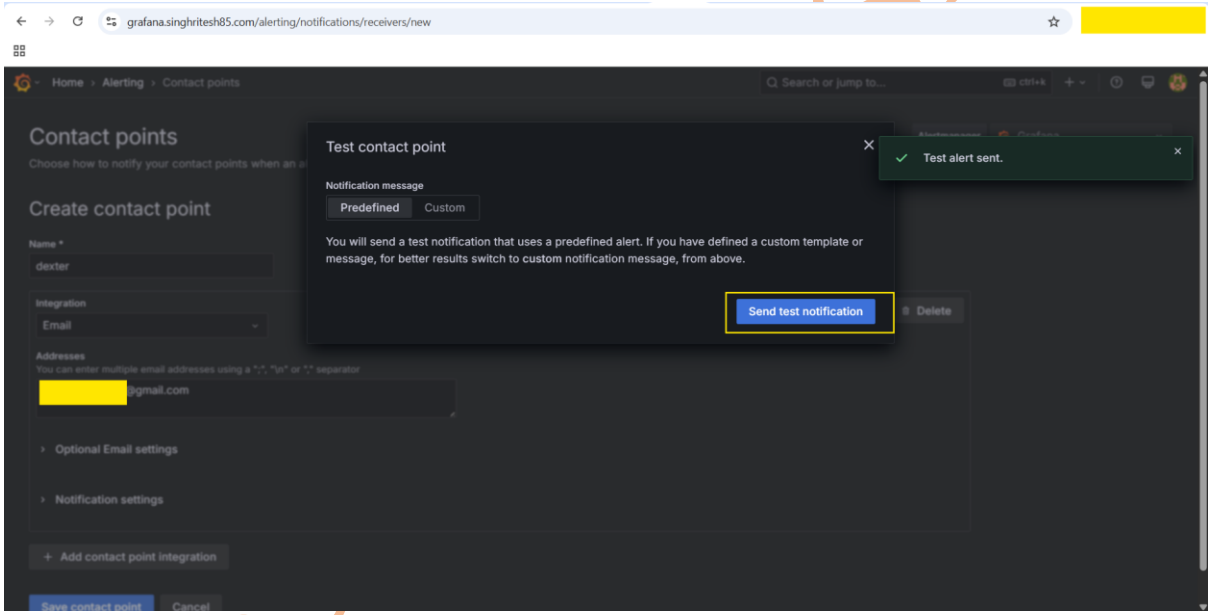


Configuration of Alerts in Grafana

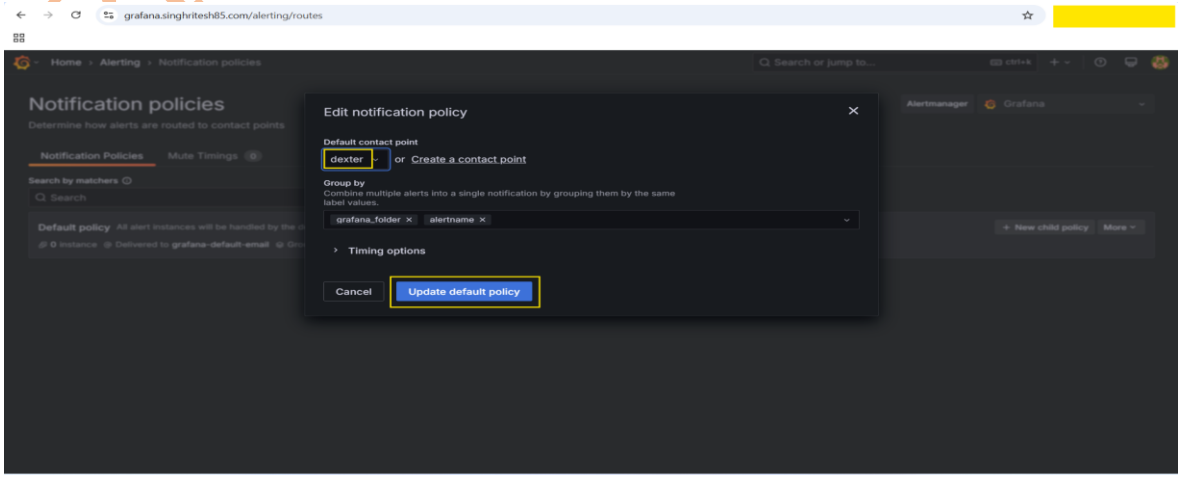
To configure Alerts in Grafana, first I created **contact points** with the Email ID and changed smtp settings in the configuration file /etc/grafana/grafana.ini of Grafana which I already discussed above. Here I had configured the contact points in Grafana UI as shown in the screenshot attached below.



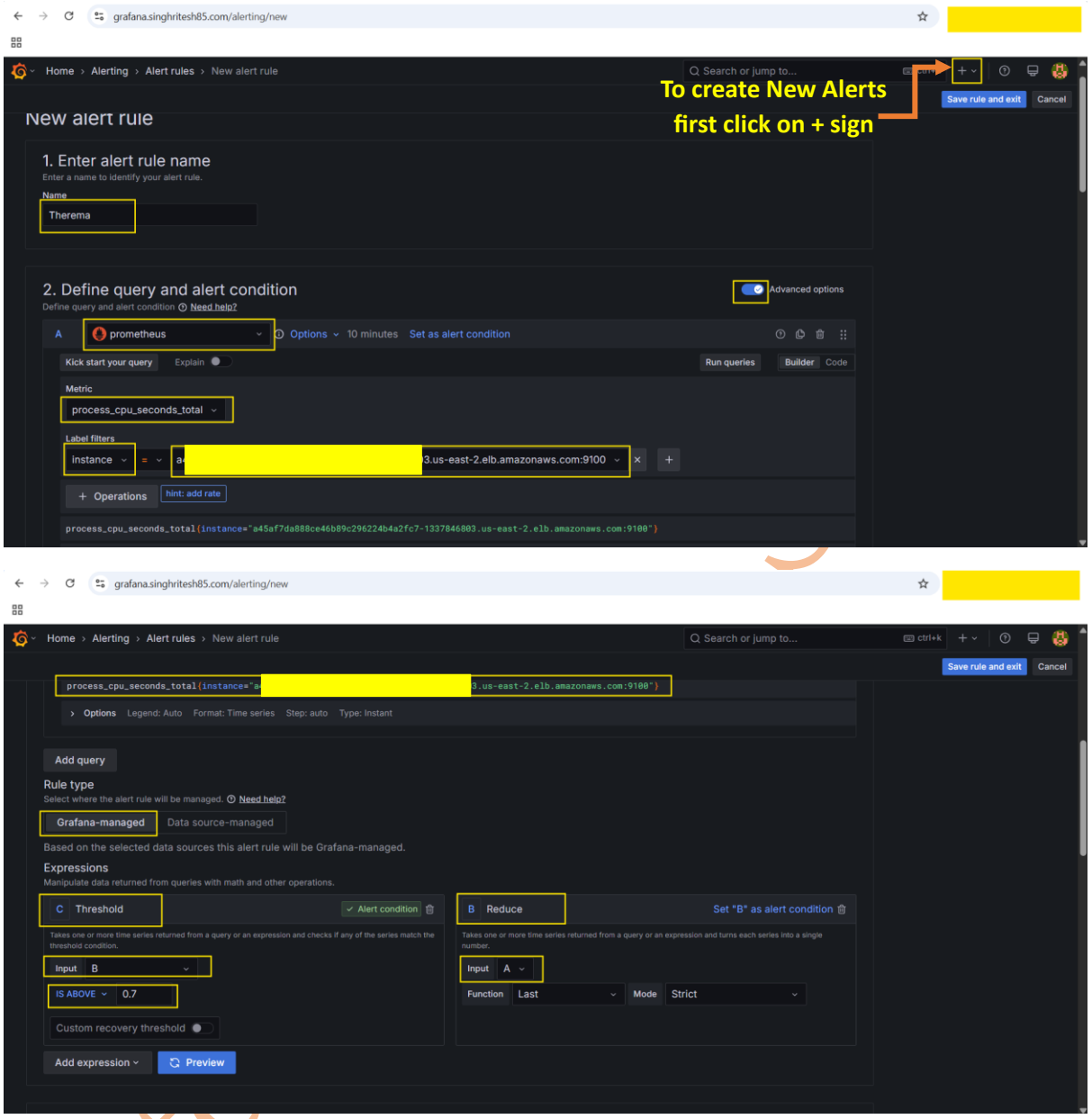


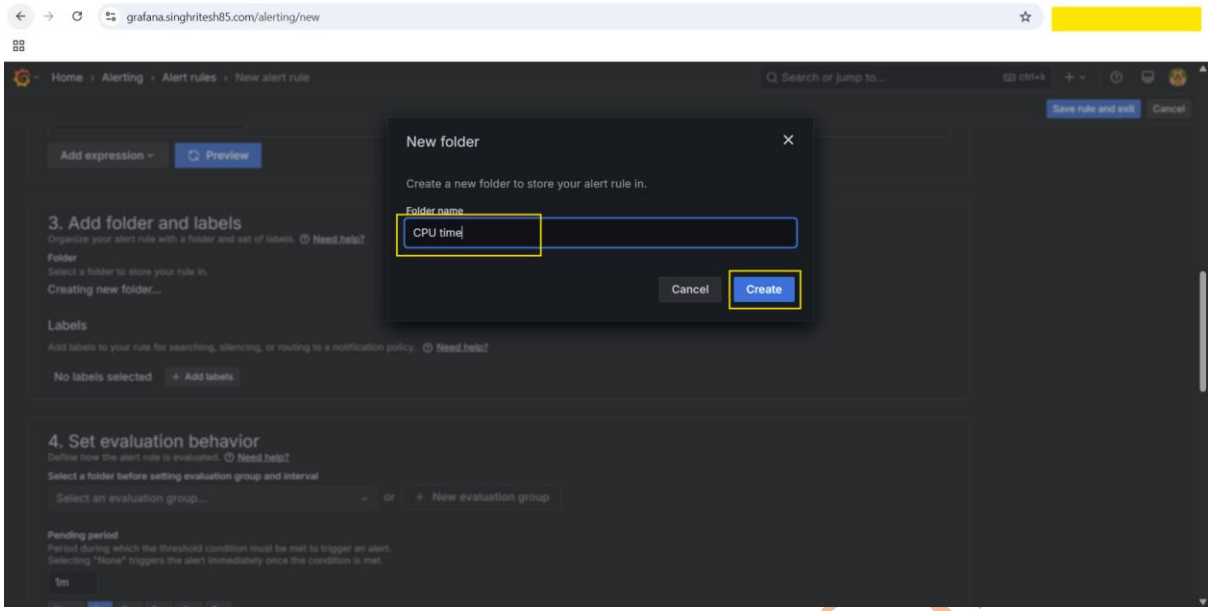


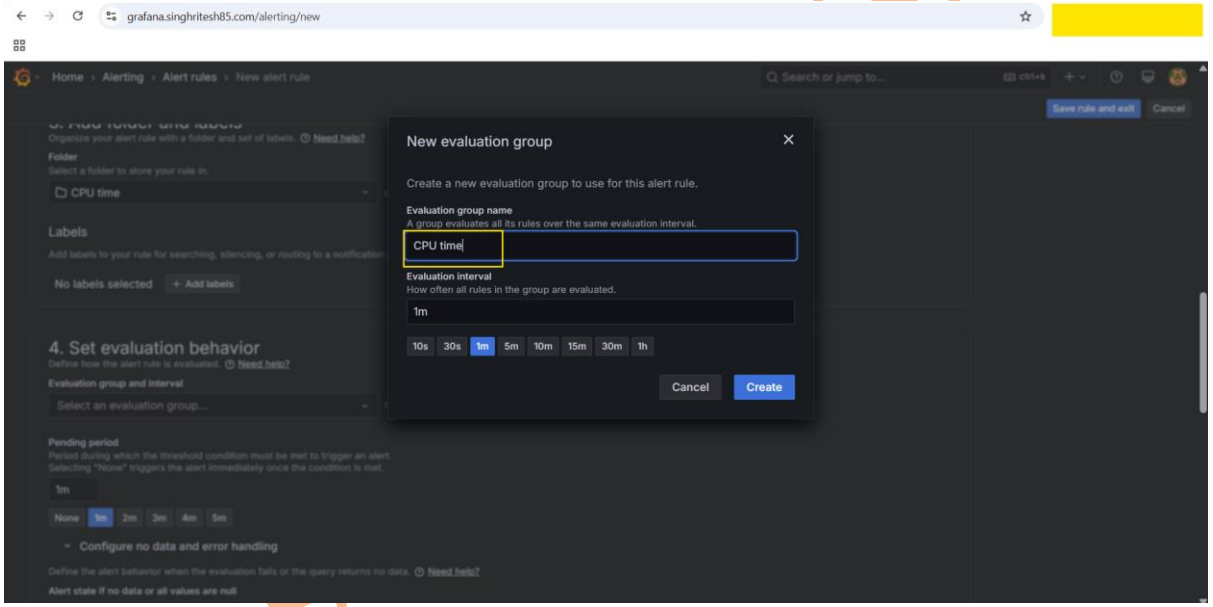
The Default **Notification Policy** had been changed as shown in the screenshot attached below.

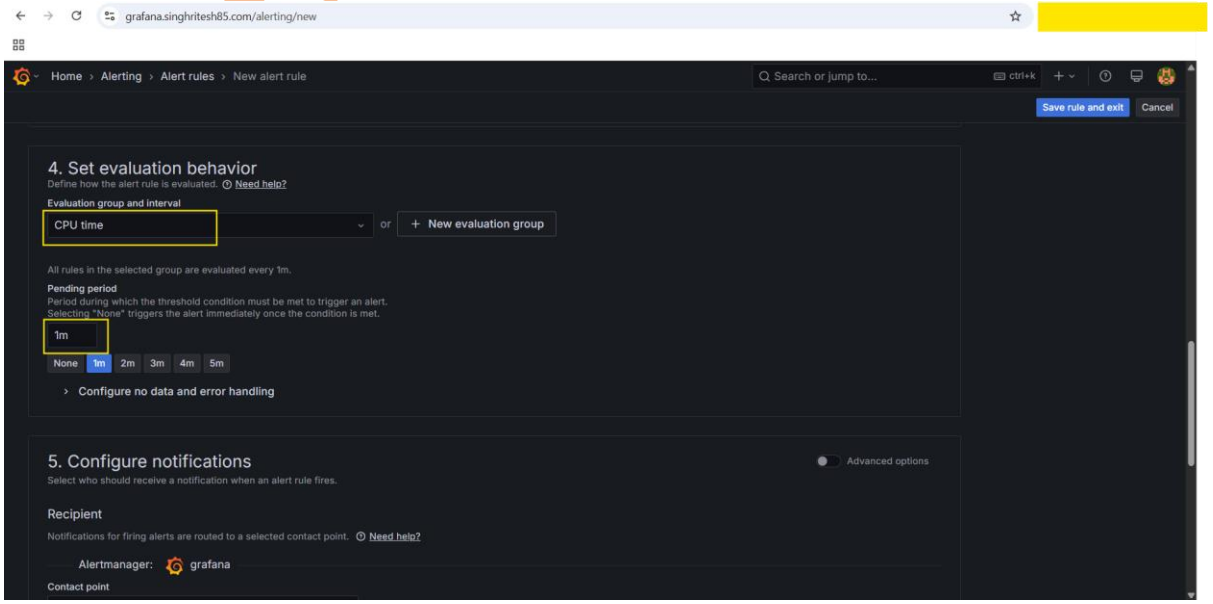


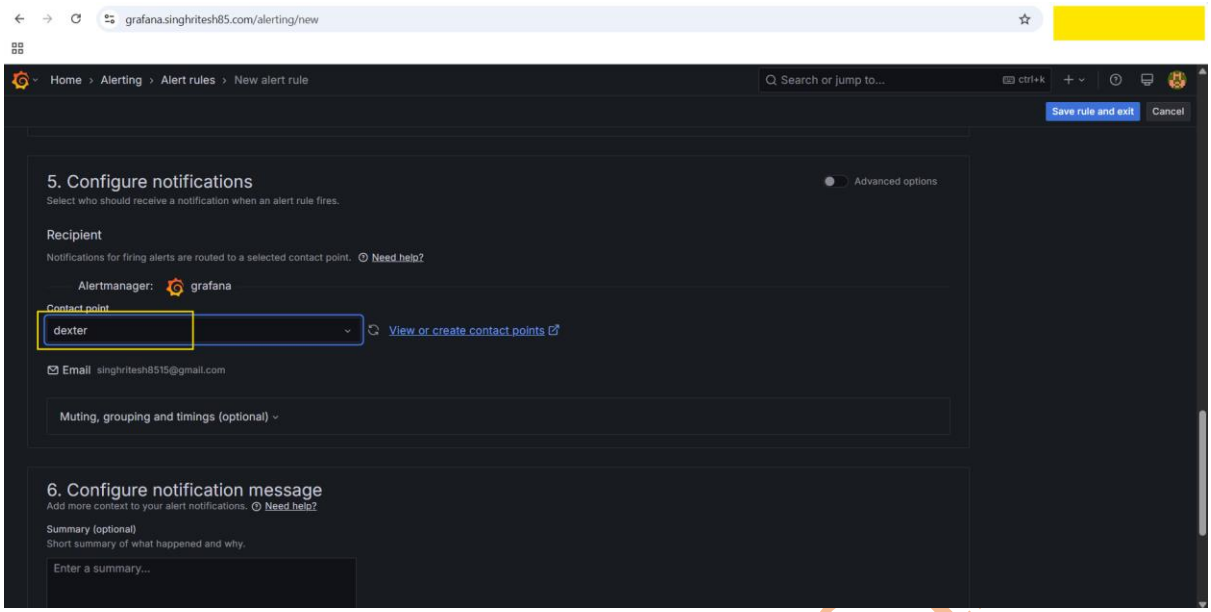
Configure **Alert Rule** as shown in the screenshot attached below.



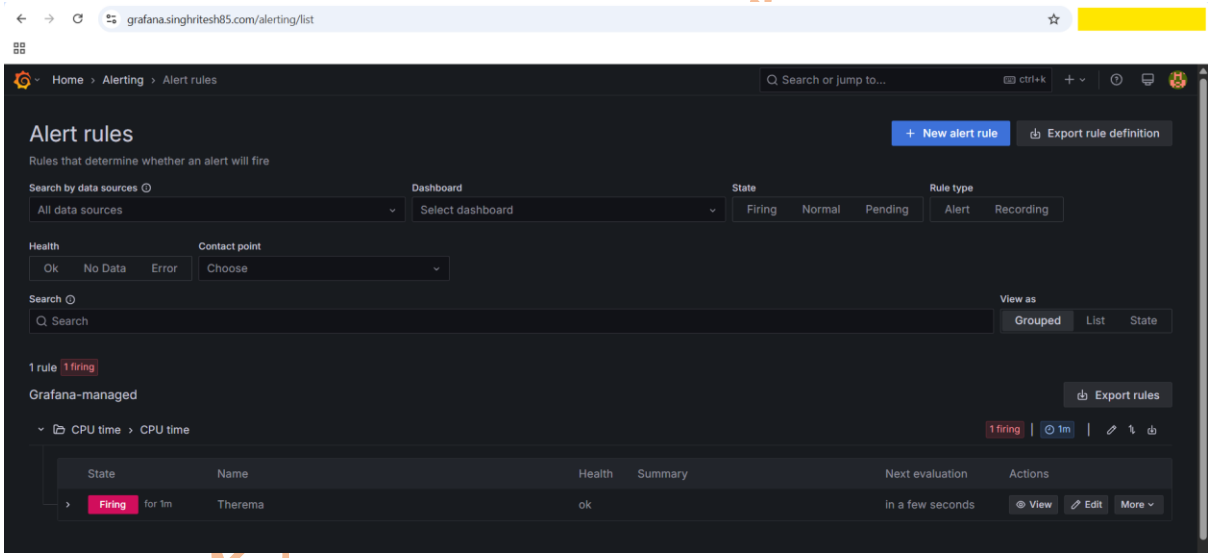

grafana.singhritesh85.com/alerting/new
Home › Alerting › Alert rules › New alert rule
Add expression
Preview
New folder
Create a new folder to store your alert rule in.
Folder name
CPU time
Cancel
Create
3. Add folder and labels
Folder
Creating new folder...
Labels
No labels selected
+ Add labels
4. Set evaluation behavior
Select a folder before setting evaluation group and interval
Select an evaluation group...
or
+ New evaluation group
Pending period
1m


grafana.singhritesh85.com/alerting/new
Home › Alerting › Alert rules › New alert rule
New evaluation group
Create a new evaluation group to use for this alert rule.
Evaluation group name
A group evaluates all its rules over the same evaluation interval.
CPU time
Evaluation interval
How often all rules in the group are evaluated.
1m
10s 30s 1m 5m 10m 15m 30m 1h
Cancel
Create
Folder
CPU time
Labels
No labels selected
+ Add labels
4. Set evaluation behavior
Evaluation group and interval
Select an evaluation group...
Pending period
1m
None 1m 2m 3m 4m 5m
Configure no data and error handling
Alert state if no data or all values are null

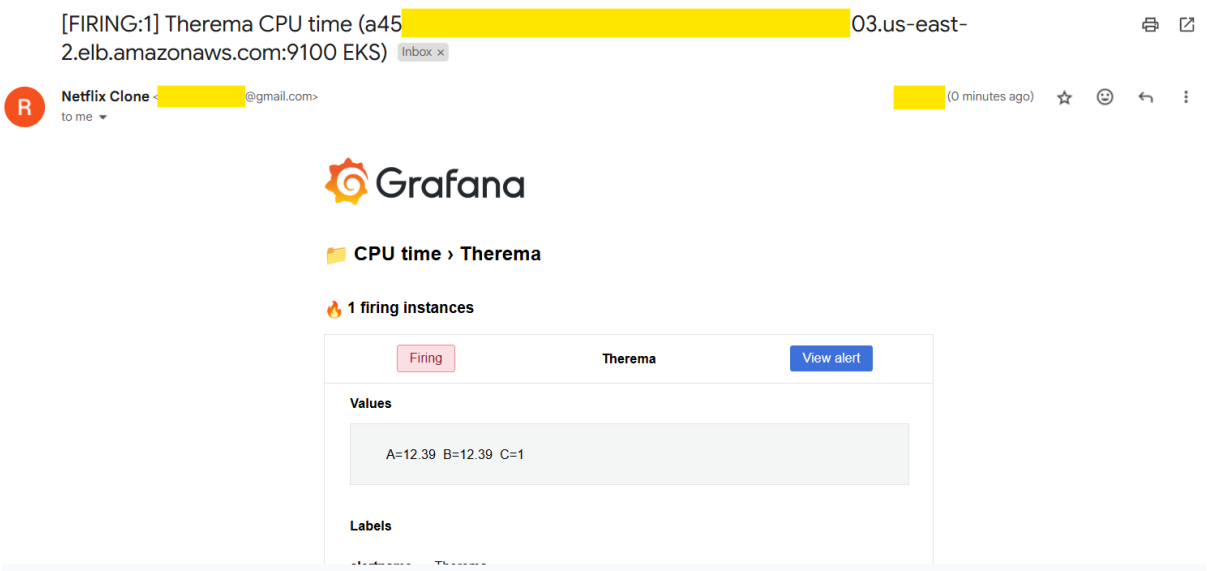

grafana.singhritesh85.com/alerting/new
Home › Alerting › Alert rules › New alert rule
Save rule and exit
Cancel
4. Set evaluation behavior
Define how the alert rule is evaluated.
Need help?
Evaluation group and interval
CPU time
or
+ New evaluation group
All rules in the selected group are evaluated every 1m.
Pending period
Period during which the threshold condition must be met to trigger an 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
5. Configure notifications
Select who should receive a notification when an alert rule fires.
Advanced options
Recipient
Notifications for firing alerts are routed to a selected contact point.
Need help?
Alertmanager:
grafana
Contact point



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 was sent to the Email ID as shown in the screenshot attached below.



Grafana

CPU time › Therema

1 firing instances

Firing | Therema | View alert

Values

A=12.39 B=12.39 C=1

Labels

alertname	Therema
grafana_folder	CPU time
instance	a45 03.us-east-2.elb.amazonaws.com:9100
job	EKS

Source Code: - https://github.com/singhritesh85/DevSecOps-Project.git

GitHub Repo: - https://github.com/singhritesh85/DevOps-Project-Netflix-Clone-Aws.git

Helm Chart: - https://github.com/singhritesh85/helm-repo-for-netflix-clone.git

Terraform Script: - https://github.com/singhritesh85/DevOps-Project-Netflix-Clone-Aws.git

Reference: - https://muditmathur121.medium.com/devsecops-netflix-clone-ci-cd-with-monitoring-email-990fbd115102