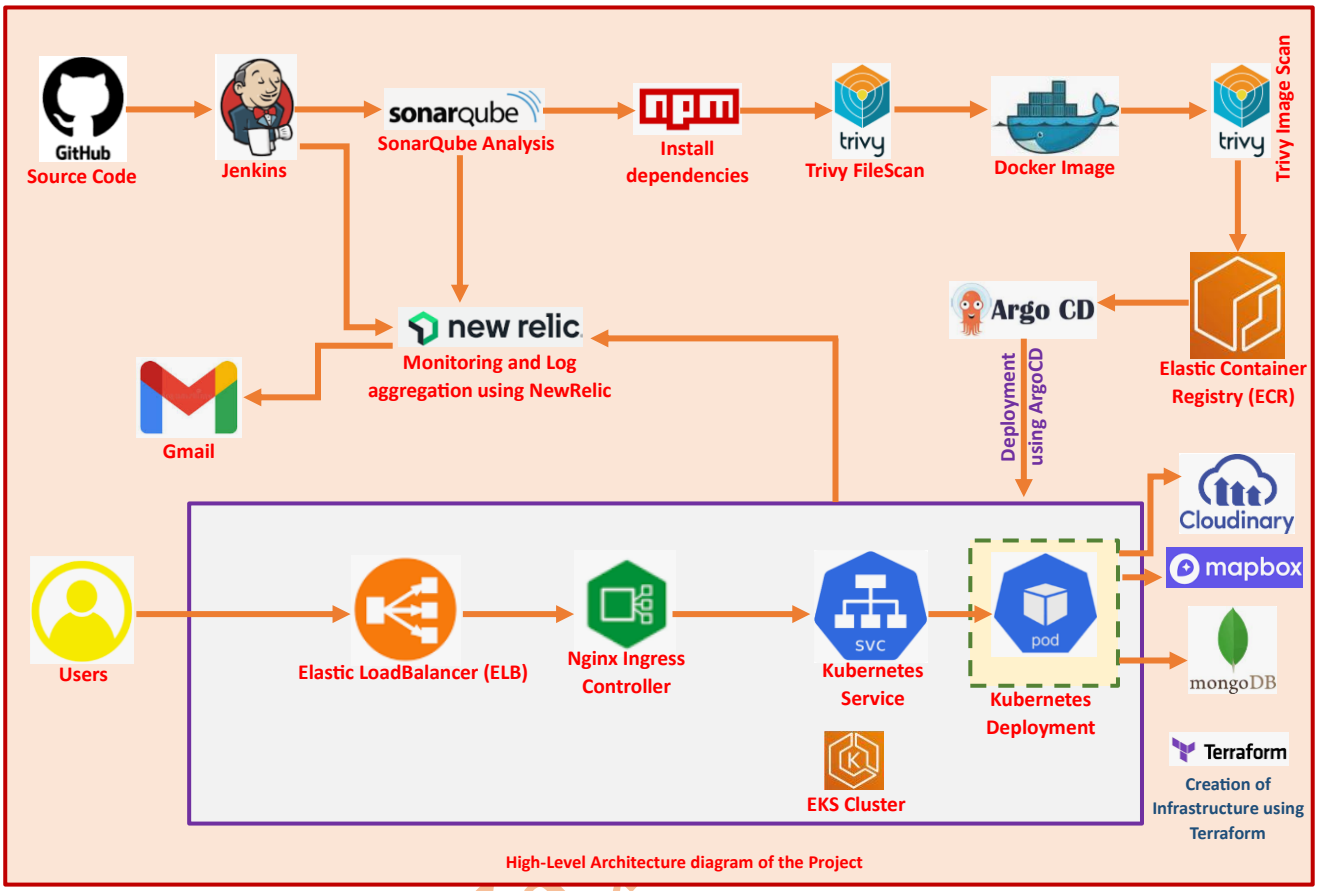
DevOps Project Yelp-Camp



High-Level Architecture diagram of the Project

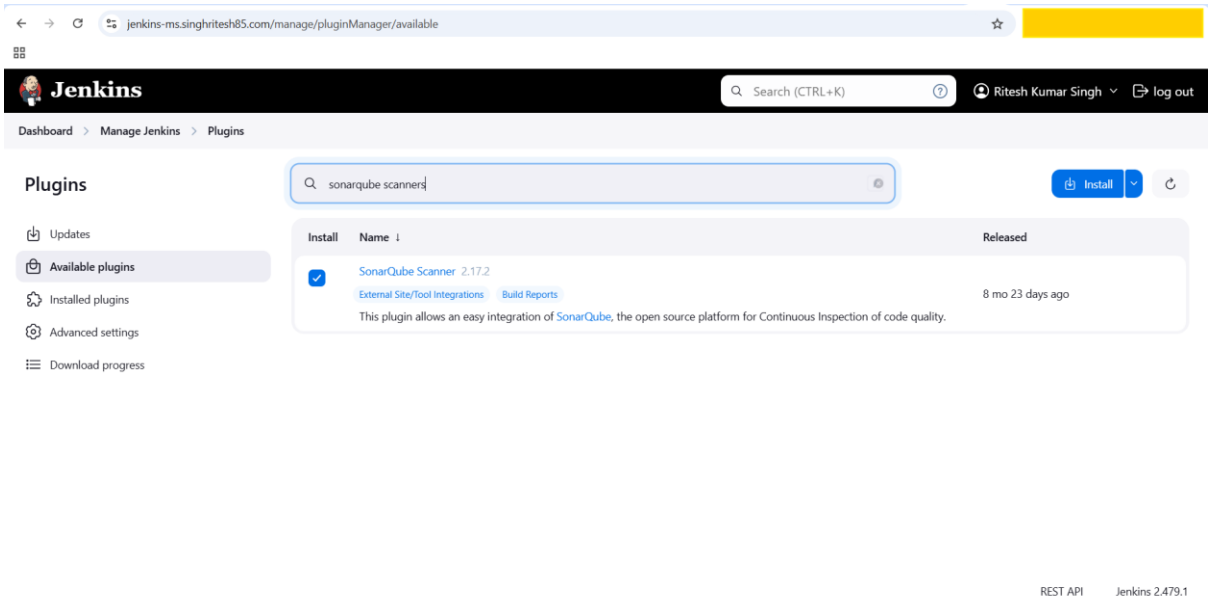
This DevOps Project aims to create the infrastructure of EKS Cluster, Elastic Container Registry (ECR), Jenkins Master, Jenkins Slave and SonarQube Server with the help of Terraform. Creation of CI/CD pipeline using Jenkins for the purpose of Deployment of the Application. For Monitoring and Log Aggregation NewRelic had been used which is shown in the high-level architecture diagram above. Alerts had been created in NewRelic and will send the Email on the Group Email Id if the Alert condition crosses the threshold. In this project Mapbox, Cloudinary and MongoDB had been used. The credentials for Mapbox, Cloudinary and MongoDB had been encrypted using base64 and provided through the kubernetes secrets using helm chart present in the GitHub Repository https://github.com/singhritesh85/helm-chart-yelp-camp.git. Credentials used had in this project had been changed later.

Agent for NewRelic had been installed in Jenkins Master, Jenkins Slave, SonarQube Server and EKS Cluster. In EKS cluster this Agent was installed using the helm chart which is explained at the later stage in this project.

To monitor Jenkins Job using NewRelic you need to install OpenTelemetry Plugin in Jenkins and do the configuration from Jenkins Systems and for monitoring SonarQube the procedure to install the NewRelic Agent has been discussed later in this project.

The Source code for this project is present in the GitHub Repository https://github.com/singhritesh85/YelpCamp.git**.**

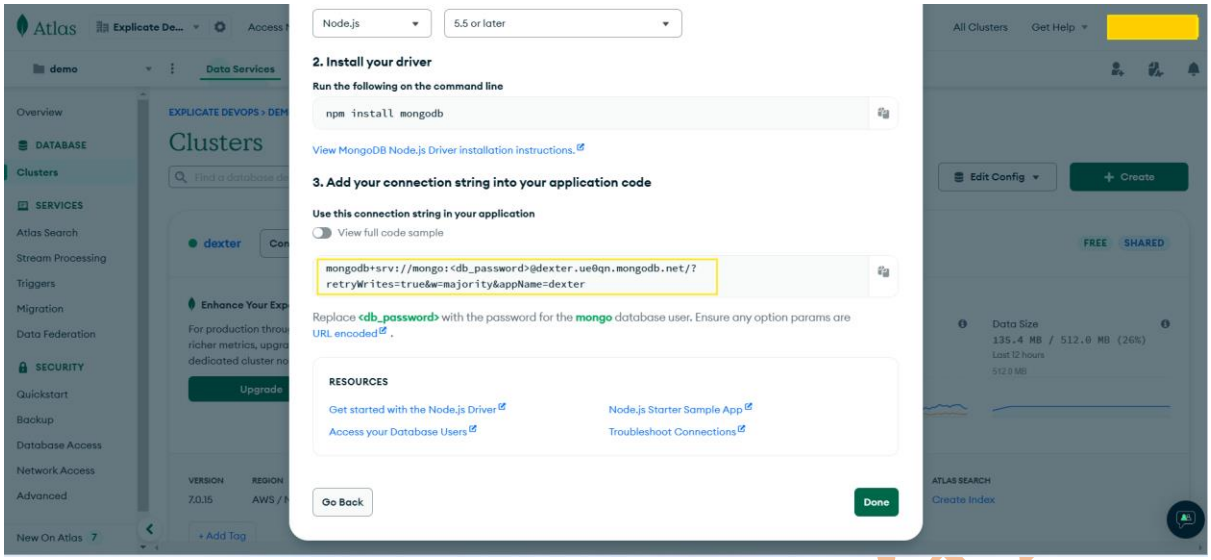
To configure SonarQube Integration with Jenkins install SonarQube Scanner Plugin in Jenkins and do the configuration from Jenkins System. For Configuration of SonarQube from Jenkins System you need to provide the Name, SonarQube Server URL, and the Credentials. For Credentials you can provide credential for SonarQube in Jenkins as a secret text in the form of secret token of SonarQube.

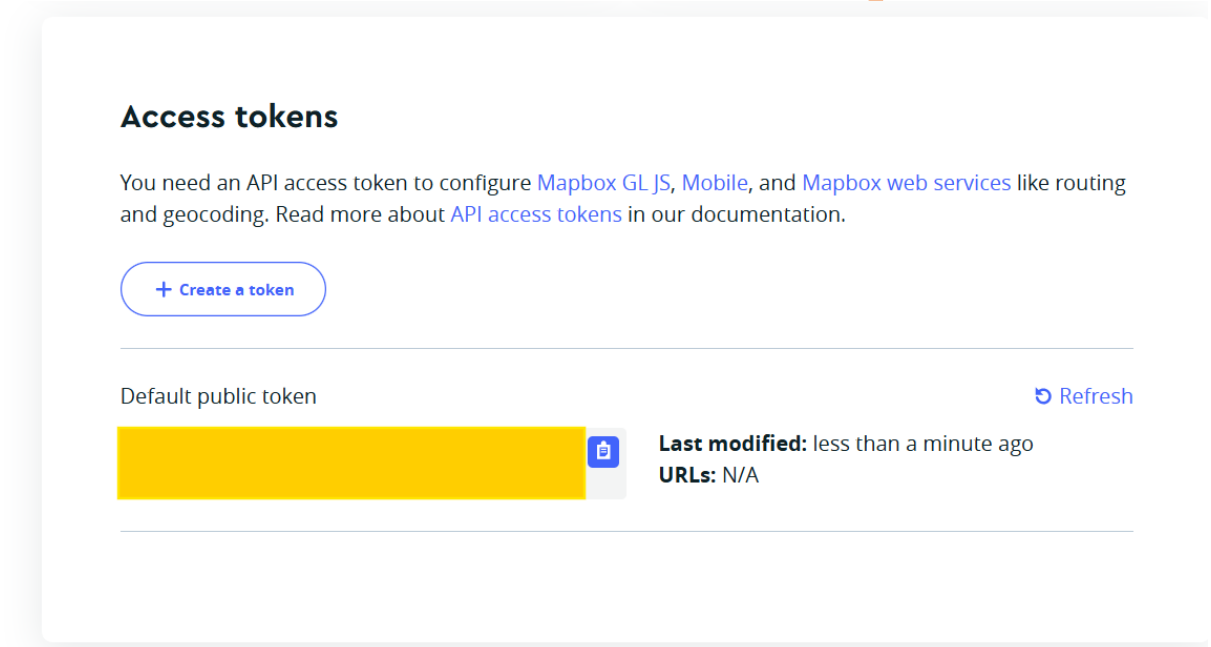


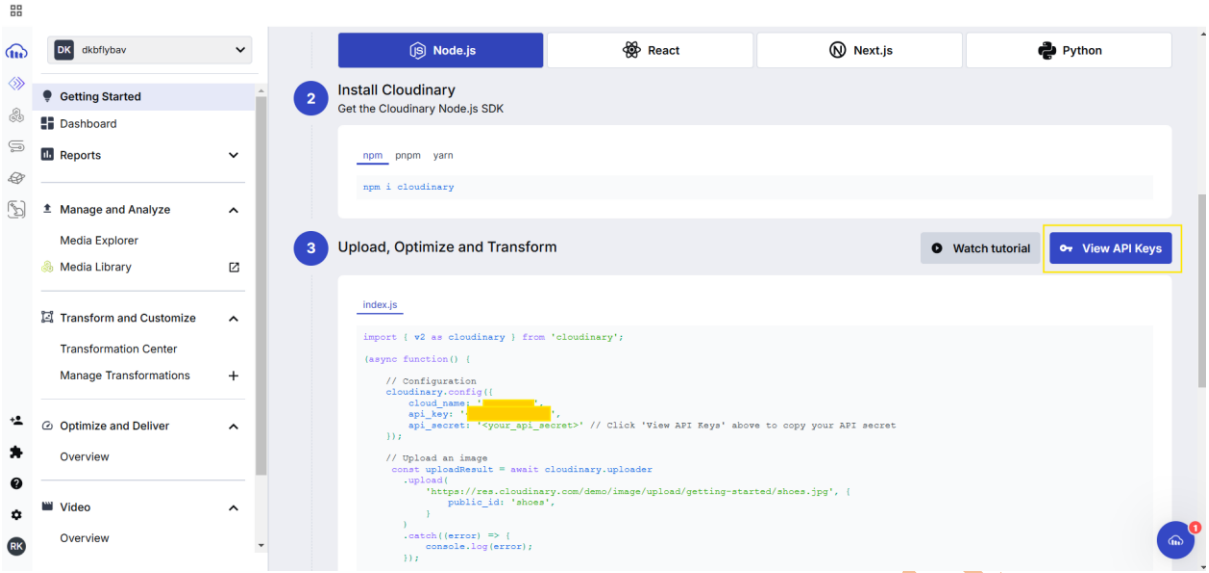
As discussed above in this project I had used MapBox, Cloudinary and MongoDB which credentials I had provided in the **.env** file and these credentials I had encrypted with base64 and provided to kubernetes pod as a secret which I provided in the helm chart present in the GitHub Repository https://github.com/singhritesh85/helm-chart-yelp-camp.git. I had changed all the credentials for MapBox, Cloudinary and MongoDB used in this project. Below command shows how I have encrypted these credentials using base64.

```
[root@                    ~]# echo -n d          v|base64
ZGtiZmx5YmF2
[root@                    ~]# echo -n 4                1 | base64
NDc3OTYxNzU4OTc3MjMx
[root@                    ~]# echo -n R                4 | base64
UlpfckVDbmhqZDEySU51eVVHVVhaSkNHWHU0
[root@                    ~]# echo -n "mongodb+srv://                                                                " | base64
bW9uZ29kYitzcnY6Ly9tb25nbzpBZG1pbjEyM0BkZXh0ZXIudWUwcW4ubW9uZ29kYi5uZXQvP3Jl
dHJ5V3JpdGVzPXRydWUmdz1tYWpvcml0eSZhcHBOYW11PWRleHRlcg==
[root@                    ~]# echo -n e           s | base64
ZXhwbGljYXRlZGV2b3Bz
```

Below screenshot shows the credentials which I used throughout this project and these credentials I have changed later after completion of the project from the MapBox, Cloudinary and MongoDB Atlas account.


Atlas
Explicate De...
demo
Data Services
Overview
DATABASE
Clusters
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced
New On Atlas
Clusters
dexter
Upgrade
Node.js
5.5 or later
2. Install your driver
Run the following on the command line
npm install mongodb
View MongoDB Node.js Driver installation instructions.
3. Add your connection string into your application code
Use this connection string in your application
View full code sample
mongodb+srv://mongo:<db_password>@dexter.ue0qn.mongodb.net/?retryWrites=true&w=majority&appName=dexter
Replace <db_password> with the password for the mongo database user. Ensure any option params are URL encoded.
RESOURCES
Get started with the Node.js Driver
Node.js Starter Sample App
Access your Database Users
Troubleshoot Connections
Go Back
Done
All Clusters
Get Help
Edit Config
Create
FREE
SHARED
Data Size
135.4 MB / 512.0 MB (26%)
Last 12 hours
ATLAS SEARCH
Create Index


Access tokens
You need an API access token to configure Mapbox GL JS, Mobile, and Mapbox web services like routing and geocoding. Read more about API access tokens in our documentation.
+ Create a token
Default public token
Refresh
Last modified: less than a minute ago
URLs: N/A



Installation of Nginx Ingress Controller

To create the Nginx Ingress Controller, follow the procedure as shown in the screenshot attached below.

```
[root@[redacted] ~]# kubectl create ns ingress-nginx
namespace/ingress-nginx created
[root@[redacted] ~]# helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
"ingress-nginx" has been added to your repositories
[root@[redacted] ~]# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
Update Complete. ⎈Happy Helming!⎈
[root@[redacted] ~]# helm install ingress-nginx ingress-nginx/ingress-nginx -n ingress-nginx
NAME: ingress-nginx
LAST DEPLOYED: Mon Apr 29 13:25:55 2024
NAMESPACE: ingress-nginx
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The ingress-nginx controller has been installed.
It may take a few minutes for the load balancer IP to be available.
You can watch the status by running 'kubectl get service --namespace ingress-nginx ingress-nginx-controller --output wide --watch'

An example Ingress that makes use of the controller:
  apiVersion: networking.k8s.io/v1
  kind: Ingress
  metadata:
    name: example
    namespace: foo
```

```
[root@[redacted] ~]# kubectl get svc -n ingress-nginx
NAME                                 TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)
                AGE
ingress-nginx-controller             LoadBalancer   [redacted]      [redacted]                                                               80:[redacted]/TCP,44
3:[redacted] TCP [redacted]
ingress-nginx-controller-admission   ClusterIP      [redacted]      <none>                                                                   443/TCP
[redacted]
```

```
root@[redacted] ~]# kubectl edit svc ingress-nginx-controller  -n ingress-nginx
ervice/ingress-nginx-controller edited
```

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  annotations:
    meta.helm.sh/release-name: ingress-nginx
    meta.helm.sh/release-namespace: ingress-nginx
    service.beta.kubernetes.io/aws-load-balancer-backend-protocol: http
    service.beta.kubernetes.io/aws-load-balancer-connection-idle-timeout: "60"
    service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled: "true"
    service.beta.kubernetes.io/aws-load-balancer-ssl-cert: arn:aws:acm:us-east-2:0[redacted]6:certificate/[redacted]
    service.beta.kubernetes.io/aws-load-balancer-ssl-ports: https
    service.beta.kubernetes.io/aws-load-balancer-type: elb
  creationTimestamp: "[redacted]"
  finalizers:
  - service.kubernetes.io/load-balancer-cleanup
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
    app.kubernetes.io/version: 1.10.1
    helm.sh/chart: ingress-nginx-4.10.1
  name: ingress-nginx-controller
  namespace: ingress-nginx
  resourceVersion: "[redacted]"
  uid: [redacted]
spec:
  allocateLoadBalancerNodePorts: true
  clusterIP: [redacted]
  clusterIPs:
  - [redacted]
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
```

```
    helm.sh/chart: ingress-nginx-4.10.1
  name: ingress-nginx-controller
  namespace: ingress-nginx
  resourceVersion: "[redacted]"
  uid: [redacted]
spec:
  allocateLoadBalancerNodePorts: true
  clusterIP: [redacted]
  clusterIPs:
  - [redacted]
  externalTrafficPolicy: Cluster
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - appProtocol: http
    name: http
    nodePort: [redacted]
    port: 80
    protocol: TCP
    targetPort: http
  - appProtocol: https
    name: https
    nodePort: [redacted]
    port: 443
    protocol: TCP
    targetPort: http
  selector:
    app.kubernetes.io/component: controller
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/name: ingress-nginx
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - hostname: [redacted].us-east-2.elb.amazonaws.com
```

Rite

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

kubectl get svc -n ingress-nginx

kubectl edit svc ingress-nginx-controller -n ingress-nginx

Add the below annotations and change the targetPort from https to http

Annotations to be added

==========

service.beta.kubernetes.io/aws-load-balancer-backend-protocol: http

service.beta.kubernetes.io/aws-load-balancer-connection-idle-timeout: "60"

service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled: "true"

service.beta.kubernetes.io/aws-load-balancer-ssl-cert: arn:aws:acm:us-east-2:XXXXXXXX:certificate/XXXXXX-XXXXXXX-XXXXXXX-XXXXXXXX

service.beta.kubernetes.io/aws-load-balancer-ssl-ports: https

service.beta.kubernetes.io/aws-load-balancer-type: elb

You need to change the targetPort for https to http in nginx ingress controller service as written below:-

==========

Before:

```
 ports:
   - name: http
     port: 80
     protocol: TCP
     targetPort: http
   - name: https
     port: 443
     protocol: TCP
     targetPort: https
```

After:

```
 ports:
   - name: http
     port: 80
     protocol: TCP
     targetPort: http
   - name: https
     port: 443
     protocol: TCP
     targetPort: 80
```

Ritesh Kumar Singh

Installation of ArgoCD

To install ArgoCD follow the procedure as mentioned below.

```
[root@ ~]# kubectl create namespace argocd
namespace/argocd created
[root@ ~]# kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
serviceaccount/argocd-dex-server created
serviceaccount/argocd-notifications-controller created
serviceaccount/argocd-redis created
serviceaccount/argocd-repo-server created
serviceaccount/argocd-server created
role.rbac.authorization.k8s.io/argocd-application-controller created
role.rbac.authorization.k8s.io/argocd-applicationset-controller created
```

```
[root@ ~]# kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath='{.data.password}' | base64 -d
[root@jenkins-slave ~]#
```

```
[root@ ~]# 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
```

```
[root@ ~]# kubectl apply -f argocd-ingress-rule.yaml
```

```
[root@ ~]# kubectl get ing -A
NAMESPACE   NAME              CLASS    HOSTS                    ADDRESS                                                                  PORTS   AGE
argocd      minimal-ingress   <none>   argocd.singhritesh85.com .us-east-2.elb.amazonaws.com   80      69s
```

kubectl create namespace argocd

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

Get password for ArgoCD

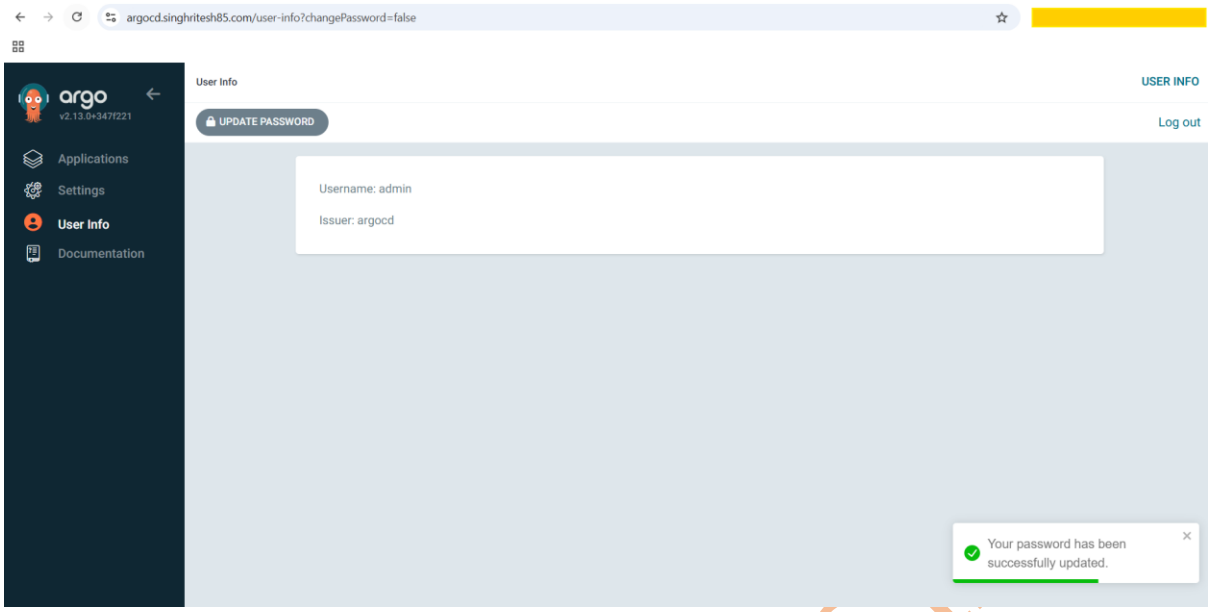
kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath='{.data.password}' | base64 -d

Ritesh Kumar Singh

```
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 example you can also choose 443


kubectl apply -f argocd-ingress-rule.yaml
```

I have updated the default password of ArgoCD as shown in the screenshot attached below. To do so you need to provide the old password, New Password and again confirm with the New Password then your password will be changed.

Ritesh Kumar Singh



To customize the DNS setting on Jenkins Slave Server I did changes in /etc/resolv.conf file as shown in the screenshot attached below. Whenever you will reboot the Jenkins Slave Server this change will no longer exist.

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

To make these changes as specified above in the /etc/resolv.conf file permanent either you create a dhcp-option set and **edit vpc settings** of that VPC and attach the newly created dhcp-option set using the changed domain-name and domain-name-server or add the entry of line **supersede domain-name "singhritesh85.com";** and **supersede domain-name-servers 8.8.8.8;** in file **/etc/dhcp/dhclient.conf** and **reboot** the Jenkins Slave Server to make these changes permanently. After reboot below screenshot shows the entry of /etc/resolv.conf file.

```
[root@ip-172-31-31-15 ~]# cat /etc/resolv.conf
options timeout:2 attempts:5
; generated by /usr/sbin/dhclient-script
search singhritesh85.com us-east-2.compute.internal
nameserver 8.8.8.8
```

If you do the entry of **supersede domain-name-servers 8.8.8.8** line only**;** in file **/etc/dhcp/dhclient.conf** and reboot your Jenkins Slave Server then the entry of /etc/resolv.conf file will be as shown in the screenshot attached below and this will also work as I have only updated the nameserver and not the domain-name.

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

Provide restricted access to the deployment user jenkins for the Kubernetes Cluster using service account, Kubernetes Role and Kubernetes RoleBinding as shown below. The deployment user jenkins has all the access within the namespace yelp-camp but does not have overall access for the entire Kubernetes Cluster.

Ritesh Kumar Singh

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

To create token for service account jenkins, create the kubernetes secrets as shown below.

Ritesh Kumar Singh

```
cat secret.yaml
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: mysecretname
  namespace: yelp-camp
  annotations:
    kubernetes.io/service-account.name: Jenkins
```

To see the token describe the kubernetes secrets as shown below.

Ritesh Kumar Singh

kubectl describe secret mysecret -n yelp-camp

Name: mysecretname

Namespace: yelp-camp

Labels: kubernetes.io/legacy-token-last-used=20XX-XX-XX

Annotations: kubernetes.io/service-account.name: jenkins

kubernetes.io/service-account.uid: XXXXXXXX-XXXX-XXXX-aXX1-XXXXXXXXXXXX

Type: kubernetes.io/service-account-token

Data

====

ca.crt: 1107 bytes

namespace: 9 bytes

token:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Ritesh Kumar Singh

Finally share the below kubeconfig file with the deployment user jenkins as shown below.

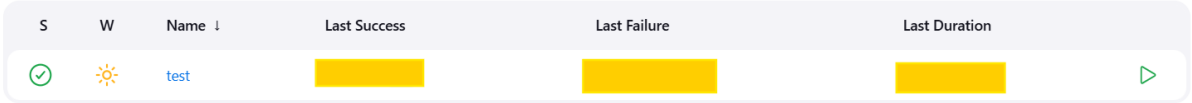
On Jenkins Slave Server the kubeconfig file is as shown below.

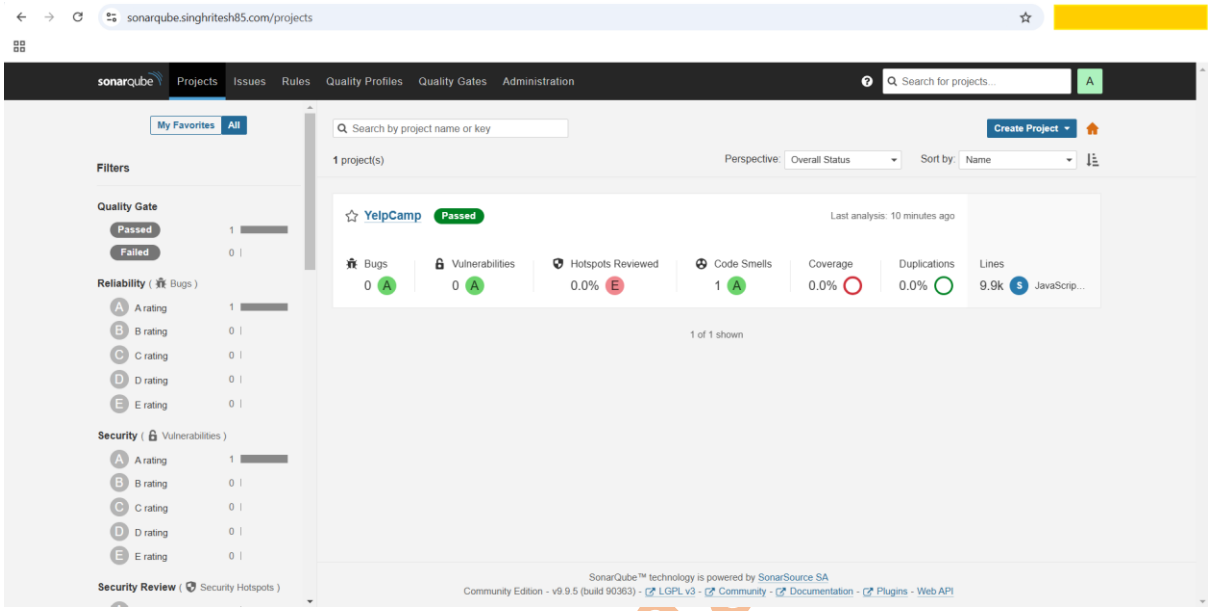
cat ~/.kube/config

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXx
    server: https://4XXXXXXXXXXXXXXXXXXXXXXXXXXXX6.gr7.us-east-2.eks.amazonaws.com
  name: arn:aws:eks:us-east-2:02XXXXXXXX6:cluster/eks-demo-cluster-dev
contexts:
- context:
    cluster: arn:aws:eks:us-east-2:02XXXXXXXX6:cluster/eks-demo-cluster-dev
    user: jenkins
  name: dexter
current-context: dexter
kind: Config
preferences: {}
users:
- name: jenkins
  user:
    token: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

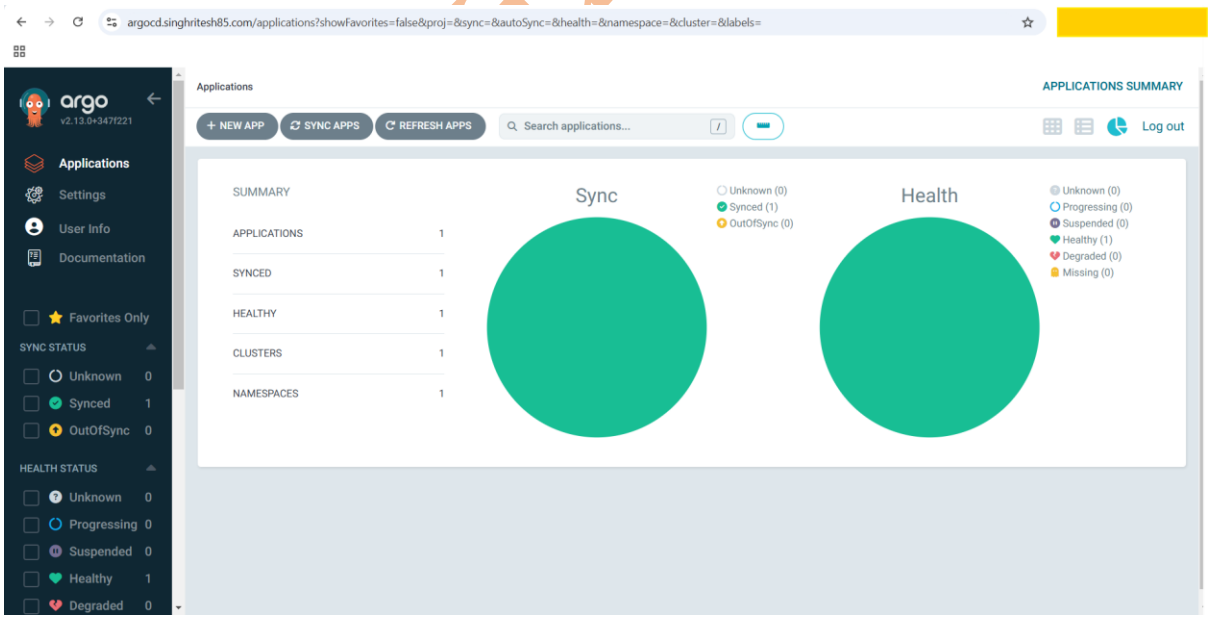
Ritesh Kumar Singh

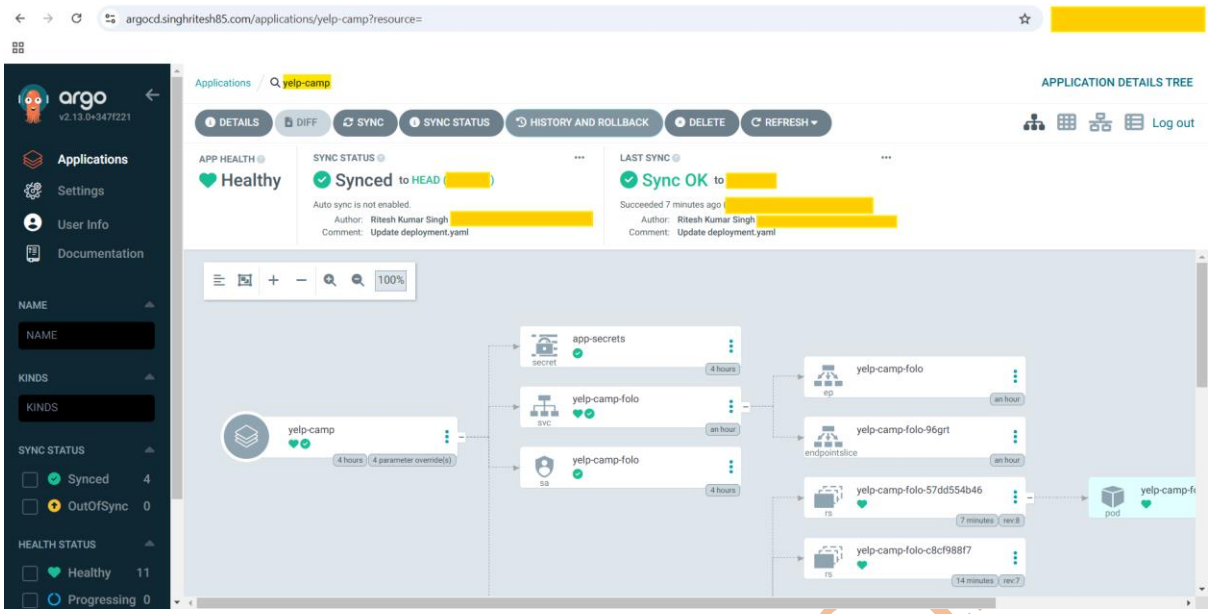
Below screenshot shows the Jenkins Job after its successful execution and SonarQube.





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





Create the Kubernetes ingress using the ingress rule as shown in the screenshot attached below.

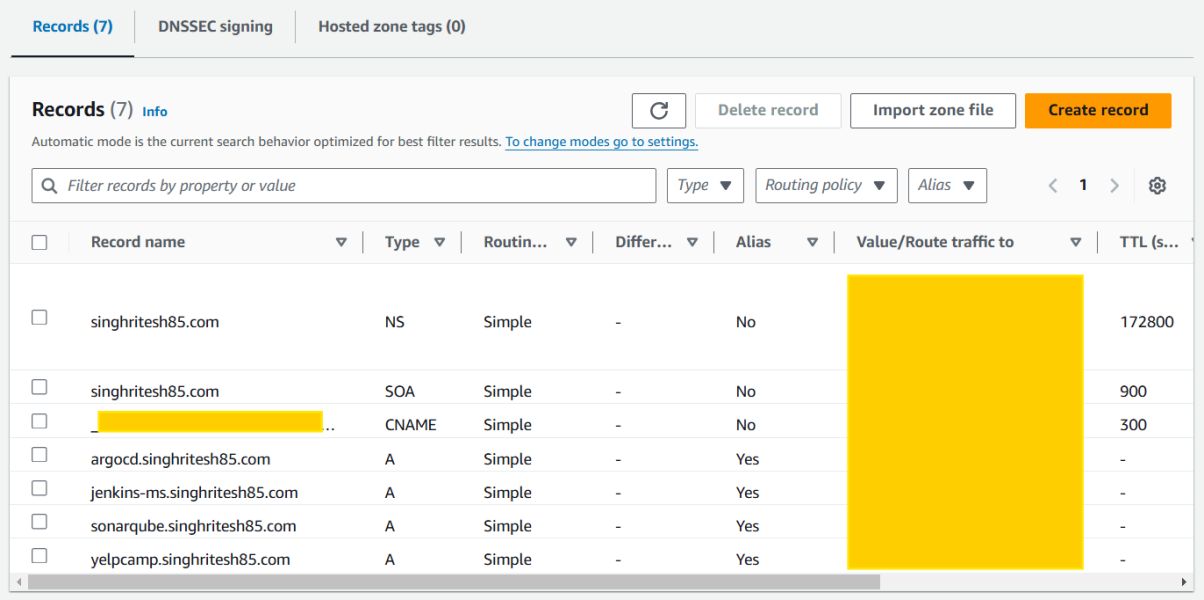
cat ingress-rule.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
 name: yelpcamp-ingress
 namespace: yelp-camp
 annotations:
  kubernetes.io/ingress.class: nginx
spec:
 ingressClassName: nginx
 rules:
 - host: yelpcamp.singhritesh85.com
  http:
   paths:
   - path: /
     pathType: Prefix
     backend:
      service:
        name: yelp-camp-folo   ### Provide your service Name
        port:
          number: 80
```

Ritesh Kumar Singh

```
[jenkins@ ~]$ kubectl get ing -A
NAMESPACE   NAME              CLASS   HOSTS                     ADDRESS                                                  PORTS   AGE
argocd      minimal-ingress   nginx   argocd.singhritesh85.com  a                                9.us-east-2.elb.amazonaws.com   80      
yelp-camp   yelpcamp-ingress  nginx   yelpcamp.singhritesh85.com a                               9.us-east-2.elb.amazonaws.com   80      
```

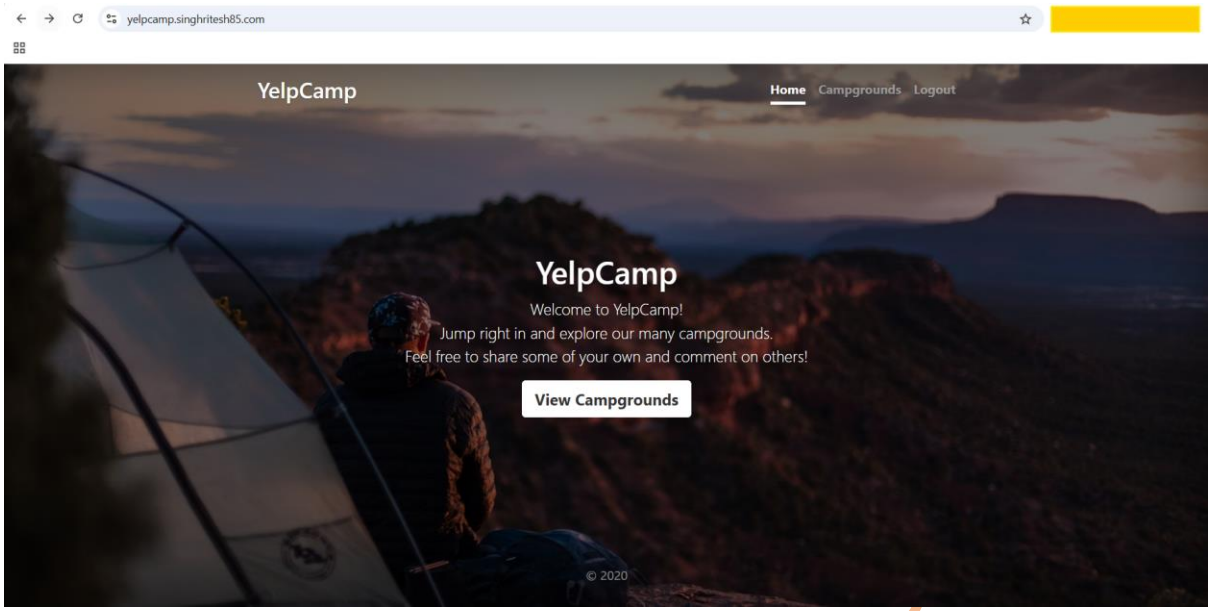
Using this ingress do the entry for DNS name in Route53 and create the record set as show in the screenshot attached below.

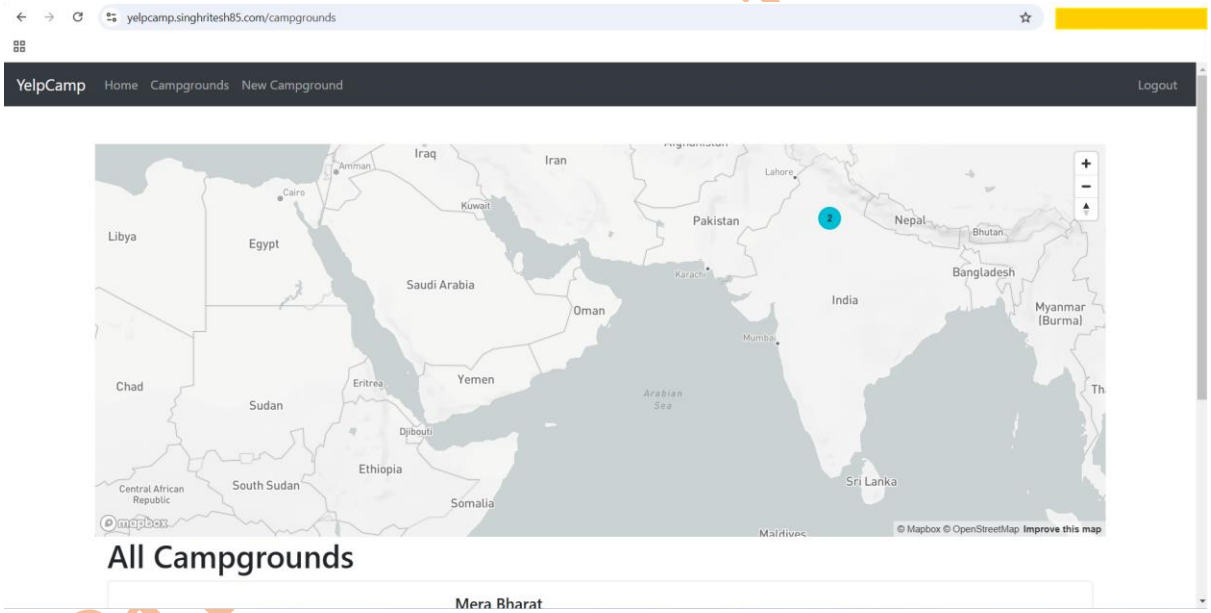


After successful execution of Jenkins Job kubernetes pods, service and deployment will be created as shown in the screenshot attached below.

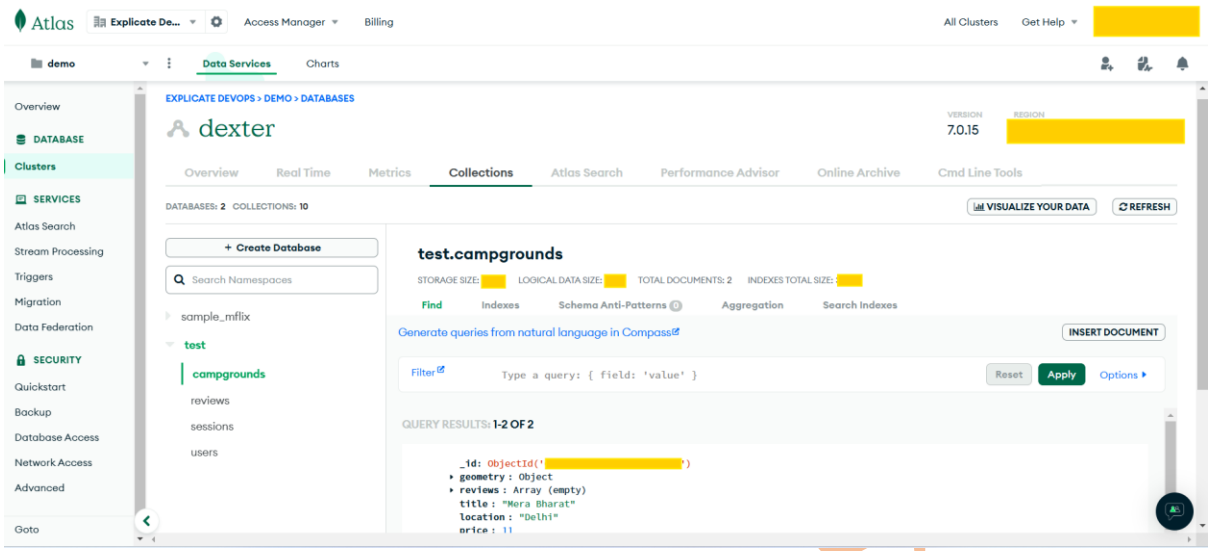
```
[jenkins@ ~]$ kubectl get all -n yelp-camp
NAME                                READY   STATUS    RESTARTS   AGE
pod/yelp-camp-folo-5 6-1 d   1/1     Running   0          17m

NAME                       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/yelp-camp-folo     ClusterIP   172. .29   <none>        80/TCP    71m

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/yelp-camp-folo     1/1     1            1           71m

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/yelp-camp-folo-5 6   1         1         1       17m
```

Finally, access the application Yelp-Camp as shown in the screenshot attached below.





I had created two new campgrounds which entries are also reflected in the MongoDB as shown in the screenshot attached below.

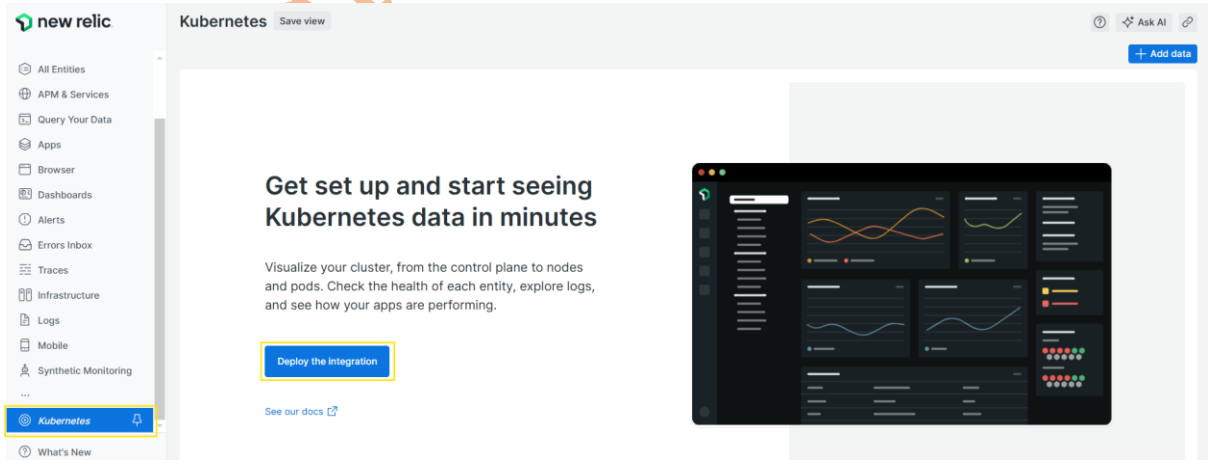


Monitoring using NewRelic

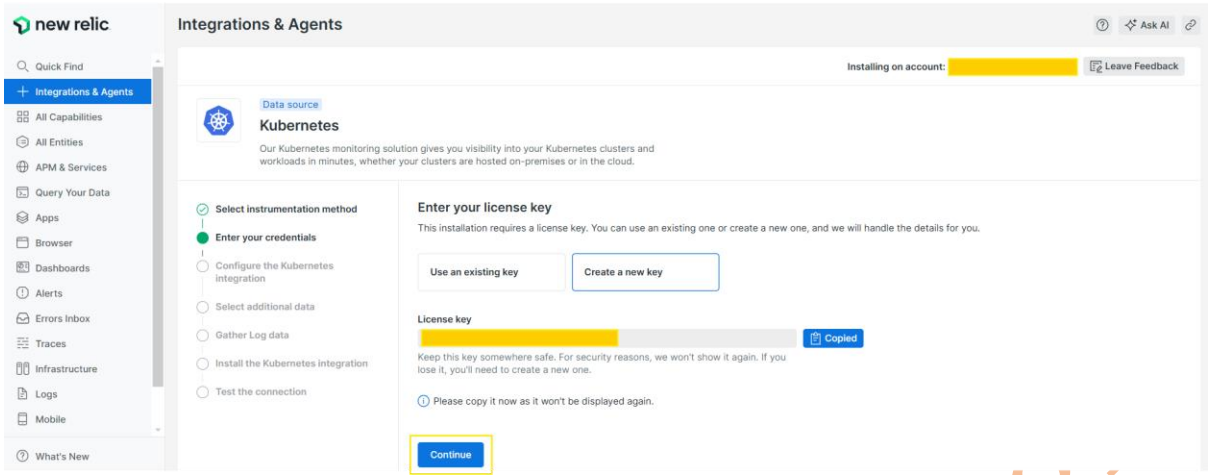
To monitor using NewRelic you need to install the NewRelic Agent on EKS Cluster, Jenkins Master and Jenkins Slave as shown in the screenshot attached below. This Agent will extract the metrics and logs from the EKS Cluster, Jenkins Master and Jenkins Slave and send to the NewRelic.

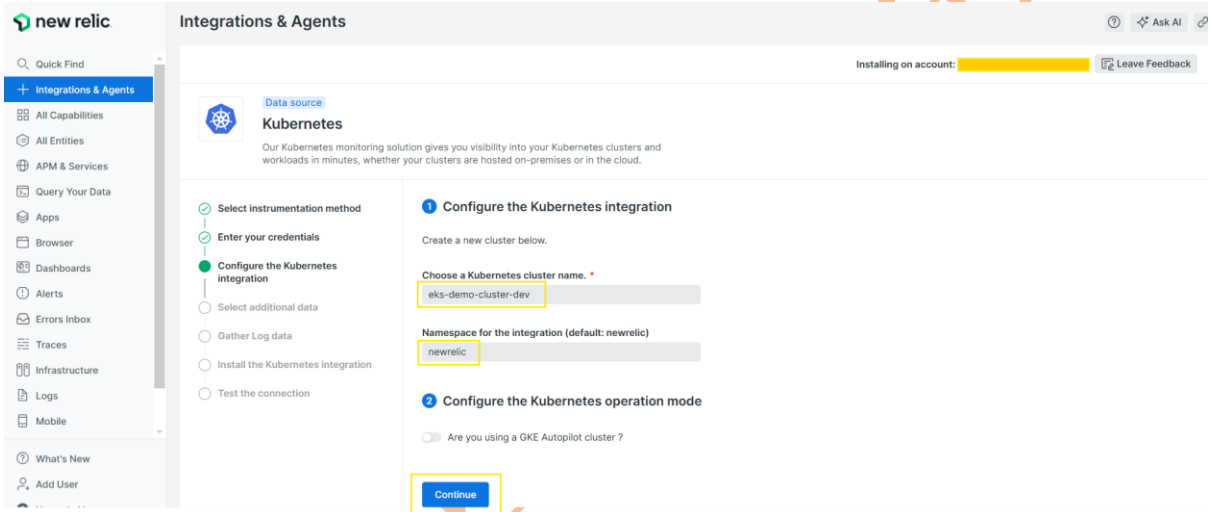
Procedure to be followed to install NewRelic Agent on SonarQube Server will be explained at later stage in this project.

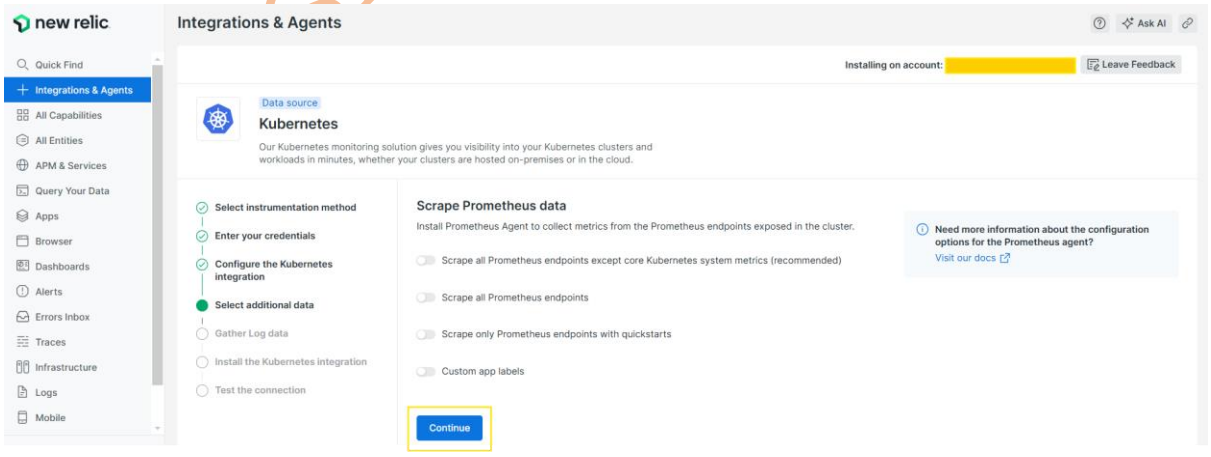
I had installed NewRelic Agent on EKS Cluster with the helm as shown in the screenshot attached below.

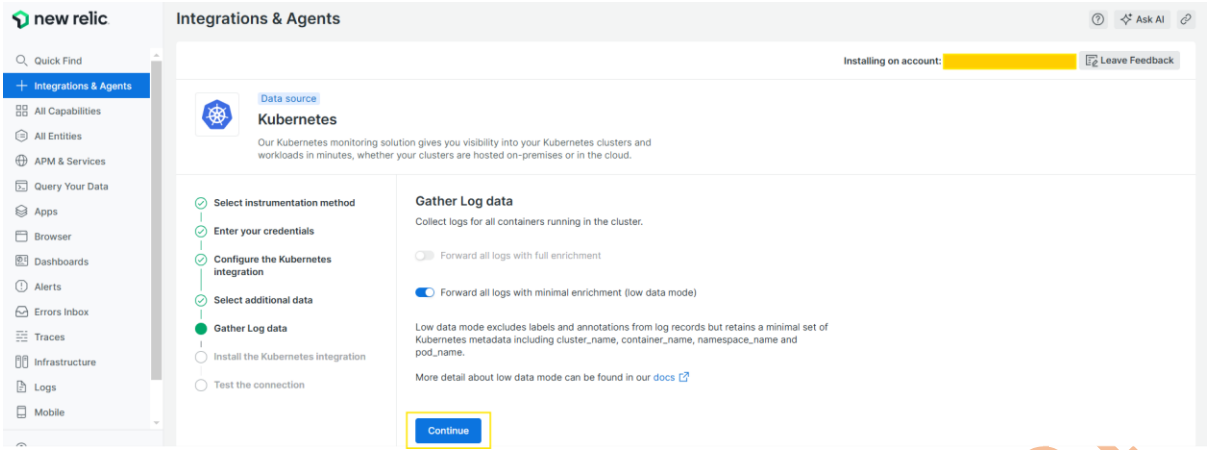


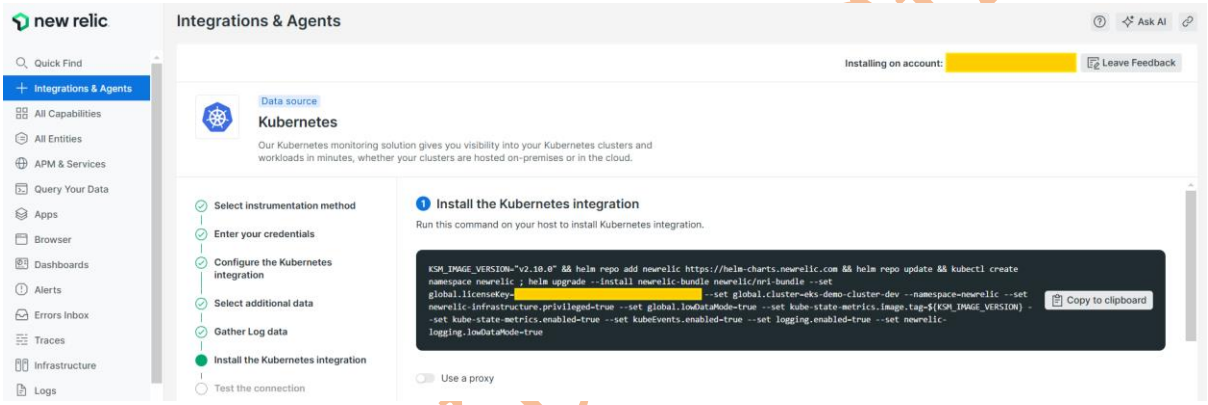
Ramesh Kumar S


new relic
Integrations & Agents
Ask AI
Quick Find
Integrations & Agents
All Capabilities
All Entities
APM & Services
Query Your Data
Apps
Browser
Dashboards
Alerts
Errors Inbox
Traces
Infrastructure
Logs
Mobile
What's New
Installing on account:
Leave Feedback
Data source
Kubernetes
Our Kubernetes monitoring solution gives you visibility into your Kubernetes clusters and workloads in minutes, whether your clusters are hosted on-premises or in the cloud.
Select instrumentation method
Enter your credentials
Configure the Kubernetes integration
Select additional data
Gather Log data
Install the Kubernetes integration
Test the connection
Enter your license key
This installation requires a license key. You can use an existing one or create a new one, and we will handle the details for you.
Use an existing key
Create a new key
License key
Copied
Keep this key somewhere safe. For security reasons, we won't show it again. If you lose it, you'll need to create a new one.
Please copy it now as it won't be displayed again.
Continue


new relic
Integrations & Agents
Ask AI
Quick Find
Integrations & Agents
All Capabilities
All Entities
APM & Services
Query Your Data
Apps
Browser
Dashboards
Alerts
Errors Inbox
Traces
Infrastructure
Logs
Mobile
What's New
Add User
Installing on account:
Leave Feedback
Data source
Kubernetes
Our Kubernetes monitoring solution gives you visibility into your Kubernetes clusters and workloads in minutes, whether your clusters are hosted on-premises or in the cloud.
Select instrumentation method
Enter your credentials
Configure the Kubernetes integration
Select additional data
Gather Log data
Install the Kubernetes integration
Test the connection
1 Configure the Kubernetes integration
Create a new cluster below.
Choose a Kubernetes cluster name. *
eks-demo-cluster-dev
Namespace for the integration (default: newrelic)
newrelic
2 Configure the Kubernetes operation mode
Are you using a GKE Autopilot cluster ?
Continue


new relic
Integrations & Agents
Ask AI
Quick Find
Integrations & Agents
All Capabilities
All Entities
APM & Services
Query Your Data
Apps
Browser
Dashboards
Alerts
Errors Inbox
Traces
Infrastructure
Logs
Mobile
Installing on account:
Leave Feedback
Data source
Kubernetes
Our Kubernetes monitoring solution gives you visibility into your Kubernetes clusters and workloads in minutes, whether your clusters are hosted on-premises or in the cloud.
Select instrumentation method
Enter your credentials
Configure the Kubernetes integration
Select additional data
Gather Log data
Install the Kubernetes integration
Test the connection
Scrape Prometheus data
Install Prometheus Agent to collect metrics from the Prometheus endpoints exposed in the cluster.
Scrape all Prometheus endpoints except core Kubernetes system metrics (recommended)
Scrape all Prometheus endpoints
Scrape only Prometheus endpoints with quickstarts
Custom app labels
Need more information about the configuration options for the Prometheus agent?
Visit our docs
Continue



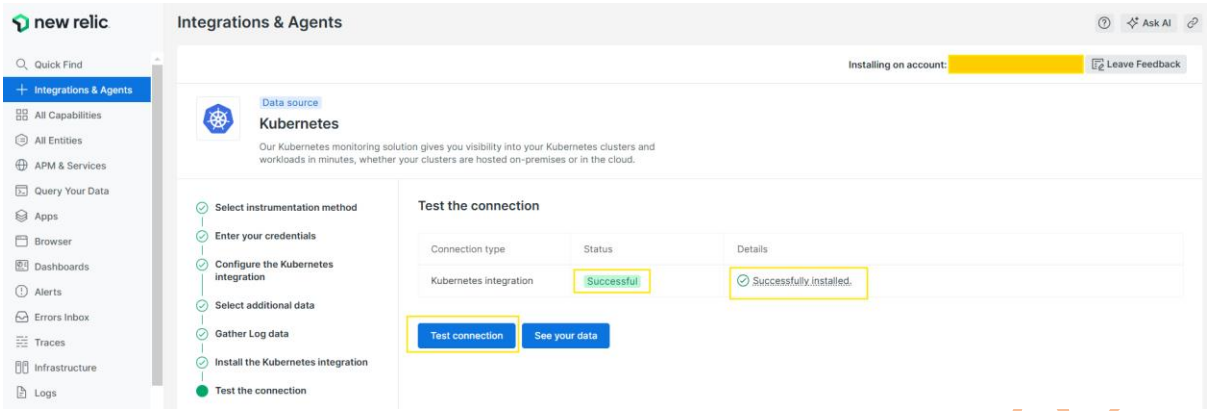


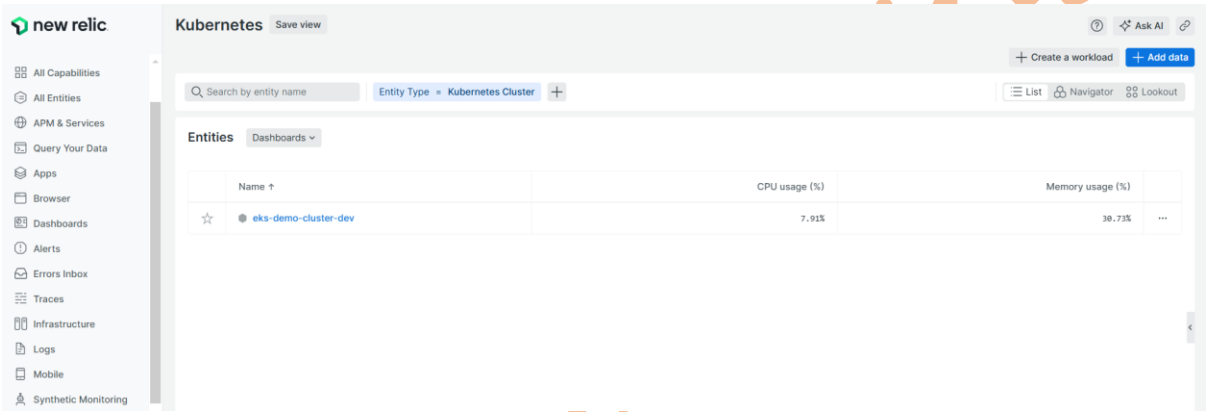
copy the above command to install the NewRelic Agent in EKS Cluster and run on a machine where kubeconfig file is present as shown in the screenshot attached below.

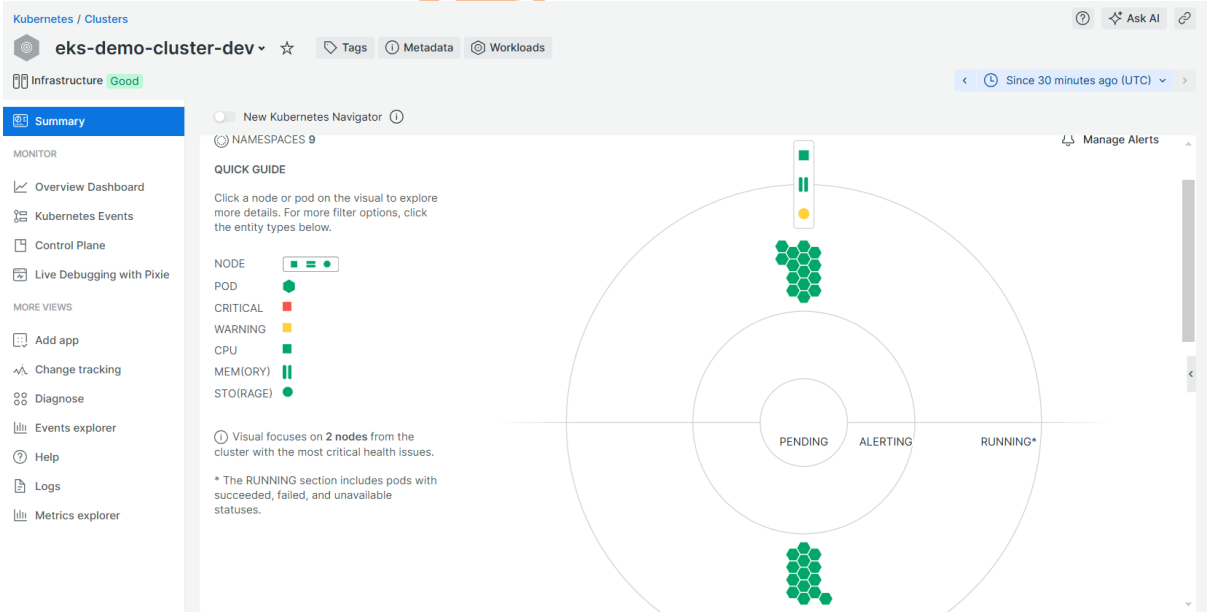
```
[root@ ~]# KSM_IMAGE_VERSION="v2.10.0" && helm repo add newrelic https://helm-charts.newrelic.com && helm repo update && kubectl create namespace newrelic ; helm upgrade --install newrelic-bundle newrelic/nri-bundle --set global.licenseKey= --set global.cluster=eks-demo-cluster-dev --namespace=newrelic --set newrelic-infrastructure.privileged=true --set global.lowDataMode=true --set kube-state-metrics.image.tag=${KSM_IMAGE_VERSION} --set kube-state-metrics.enabled=true --set kubeEvents.enabled=true --set logging.enabled=true --set newrelic-logging.lowDataMode=true
"newrelic" already exists with the same configuration, skipping
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
...Successfully got an update from the "newrelic" chart repository
Update Complete. ⎈Happy Helming!⎈
namespace/newrelic created
Release "newrelic-bundle" does not exist. Installing it now.
```

Finally, following pods will be created in the namespace newrelic as shown in the screenshot attached below.

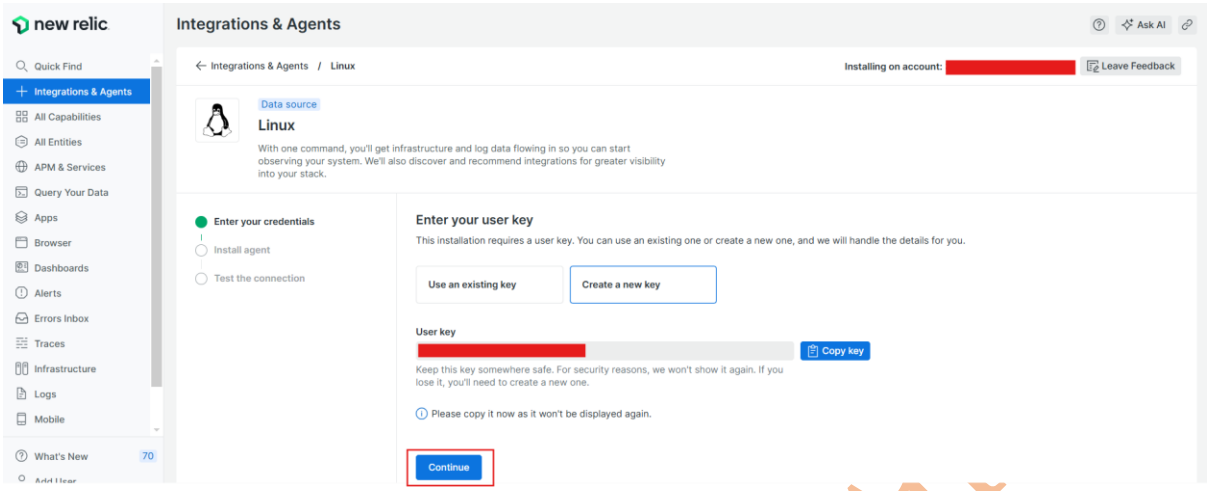
```
[root@ip-172-31-11-46 ~]# kubectl get pods -n newrelic
NAME                                                  READY   STATUS    RESTARTS   AGE
newrelic-bundle-kube-state-metrics-                   1/1     Running   0
newrelic-bundle-newrelic-logging-                     1/1     Running   0
newrelic-bundle-newrelic-logging-                     1/1     Running   0
newrelic-bundle-nri-kube-events-                      2/2     Running   0
newrelic-bundle-nri-metadata-injection-               1/1     Running   0
newrelic-bundle-nrk8s-ksm-                            2/2     Running   0
newrelic-bundle-nrk8s-kubelet-                        2/2     Running   0
newrelic-bundle-nrk8s-kubelet-                        2/2     Running   0
```

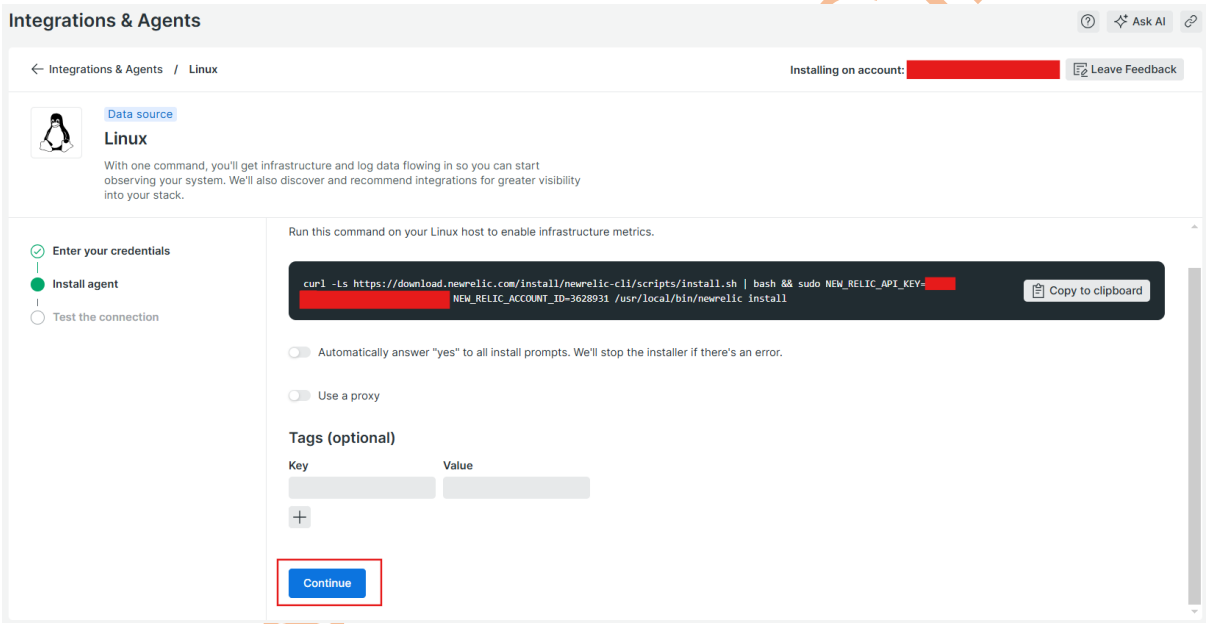

new relic
Integrations & Agents
Ask AI
Quick Find
Integrations & Agents
All Capabilities
All Entities
APM & Services
Query Your Data
Apps
Browser
Dashboards
Alerts
Errors Inbox
Traces
Infrastructure
Logs
Installing on account:
Leave Feedback
Data source
Kubernetes
Our Kubernetes monitoring solution gives you visibility into your Kubernetes clusters and workloads in minutes, whether your clusters are hosted on-premises or in the cloud.
Select instrumentation method
Enter your credentials
Configure the Kubernetes integration
Select additional data
Gather Log data
Install the Kubernetes integration
Test the connection
Test the connection
Connection type
Status
Details
Kubernetes integration
Successful
Successfully installed.
Test connection
See your data


new relic
Kubernetes
Save view
Ask AI
All Capabilities
All Entities
APM & Services
Query Your Data
Apps
Browser
Dashboards
Alerts
Errors Inbox
Traces
Infrastructure
Logs
Mobile
Synthetic Monitoring
Create a workload
Add data
Search by entity name
Entity Type = Kubernetes Cluster
List
Navigator
Lookout
Entities
Dashboards
Name
CPU usage (%)
Memory usage (%)
eks-demo-cluster-dev
7.91%
30.73%


Kubernetes / Clusters
Ask AI
eks-demo-cluster-dev
Tags
Metadata
Workloads
Infrastructure
Good
Since 30 minutes ago (UTC)
Summary
MONITOR
Overview Dashboard
Kubernetes Events
Control Plane
Live Debugging with Pixie
MORE VIEWS
Add app
Change tracking
Diagnose
Events explorer
Help
Logs
Metrics explorer
New Kubernetes Navigator
NAMESPACES 9
Manage Alerts
QUICK GUIDE
Click a node or pod on the visual to explore more details. For more filter options, click the entity types below.
NODE
POD
CRITICAL
WARNING
CPU
MEM(ORY)
STO(RAGE)
Visual focuses on 2 nodes from the cluster with the most critical health issues.
* The RUNNING section includes pods with succeeded, failed, and unavailable statuses.
PENDING
ALERTING
RUNNING*

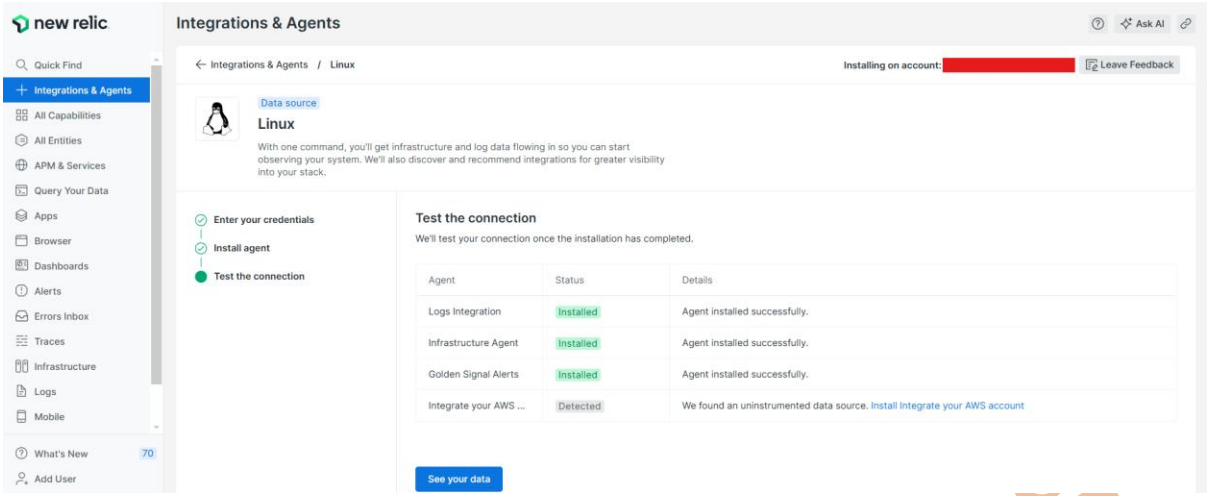
Installing NewRelic Agents on Jenkins Master and Jenkins Slave as shown in the screenshot attached below.





copy the above command to install the NewRelic Agent and run on Jenkins Master and Jenkins Slave.

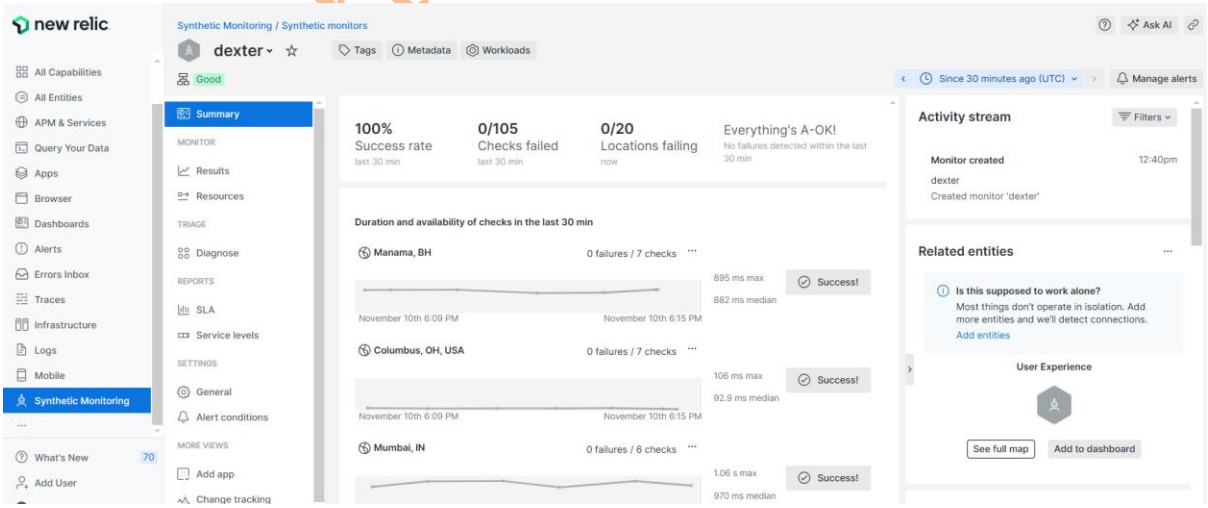
```
[root@ ~]# curl -Ls https://download.newrelic.com/install/newrelic-cli/scripts/install.sh | bash && sudo NEW_RELIC_API_KEY=
 NEW_RELIC_ACCOUNT_ID= /usr/local/bin/newrelic install
```



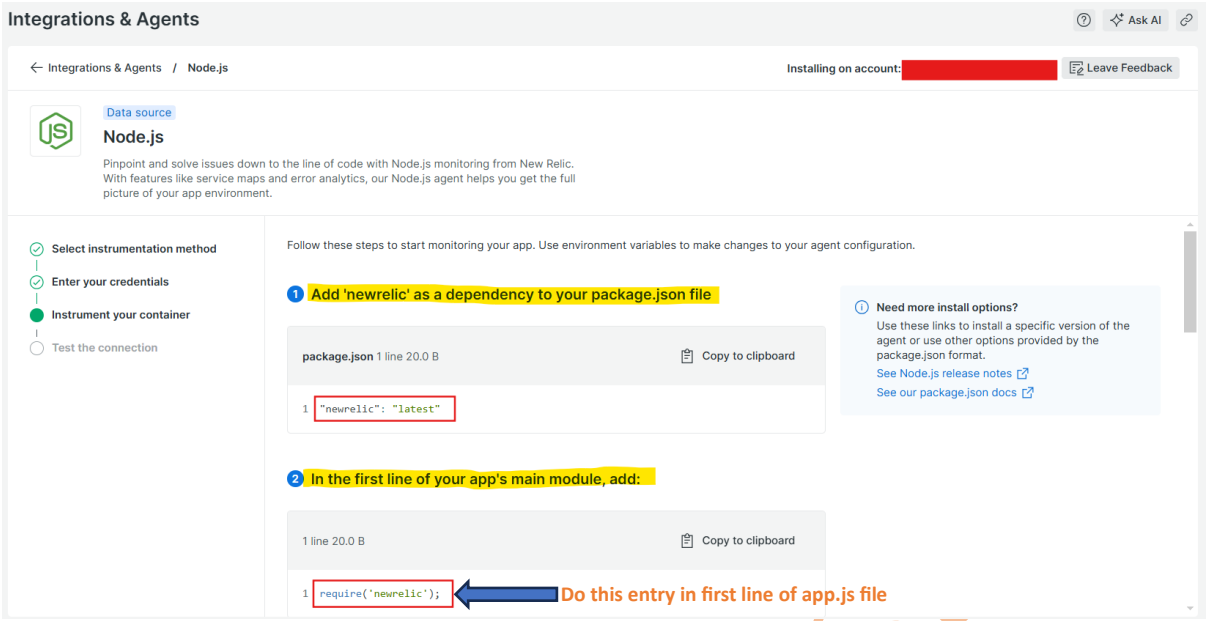
Hosts [redacted]

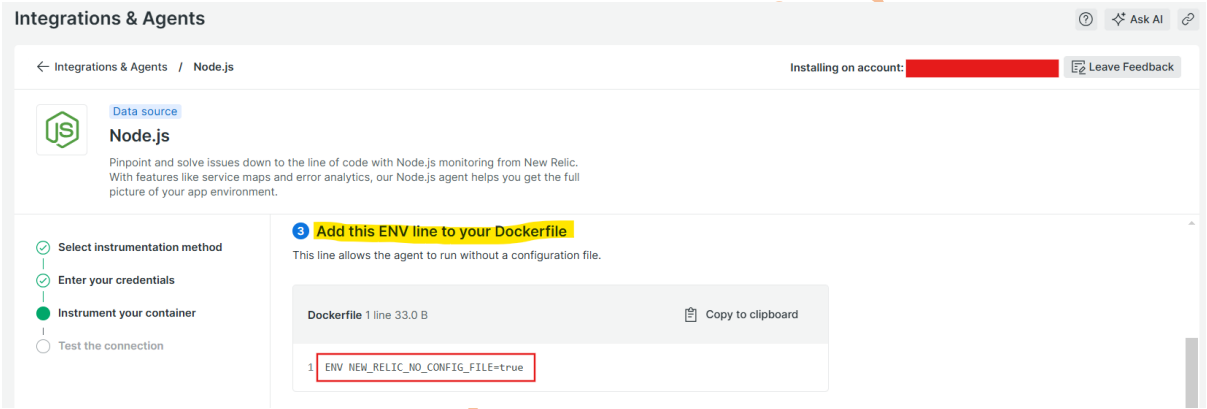
	Name ↑	Agent ver...	CPU usa...	Memory ...	Storage u...	Network ...	Network r...	
☆	ip-[redacted]us-east-2.compute.inter...	1.57.2	8.74%	27.71%	68.73%	873 B/s	10 kB/s	···
☆	ip-[redacted]us-east-2.compute.inter...	1.57.2	7.67%	27.51%	58.08%	822 B/s	9.05 kB/s	···
☆	ip-[redacted]us-east-2.compute.internal	1.57.2	2.21%	34.88%	17.83%	1.41 kB/s	2.66 kB/s	···
☆	ip-[redacted].us-east-2.compute.inter...	1.57.2	2.03%	18.93%	35.18%	699 B/s	259 B/s	···
☆	ip-[redacted]us-east-2.compute.inter...	1.57.2	3.62%	56.57%	19.36%	3.66 kB/s	52.3 kB/s	···

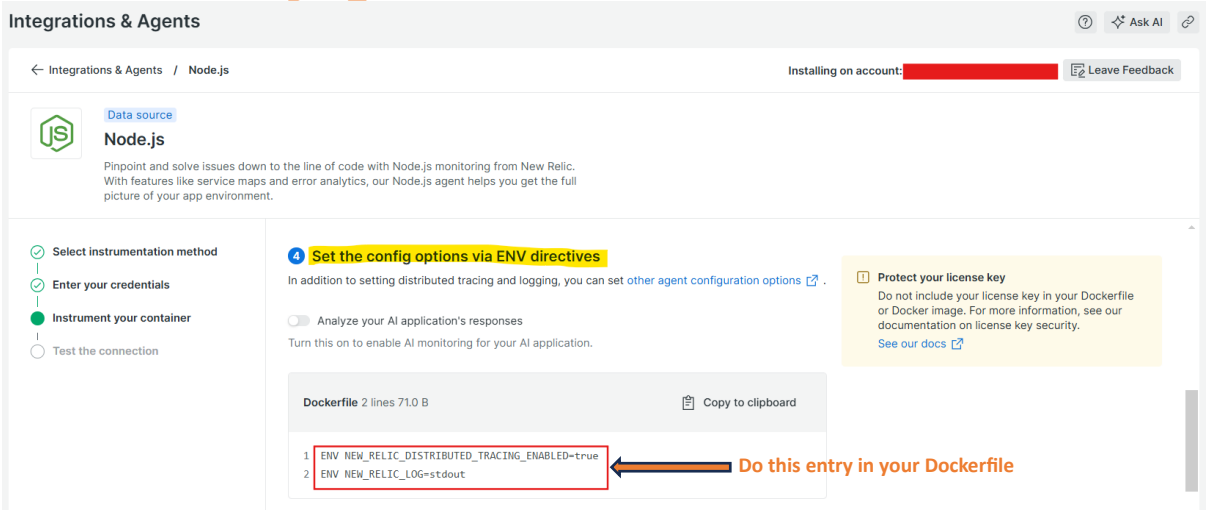
I am performing synthetic monitoring on Application URL https://yelpcamp.singhritesh85.com. On a periodic Interval of every 1 min, it will perform the Synthetic Monitoring from All the public Locations. The below screenshot shows the result of Synthetic Monitoring.



For Application Performance Monitoring (APM), follow the procedure as written below.







Do the below line entry in your Dockerfile.

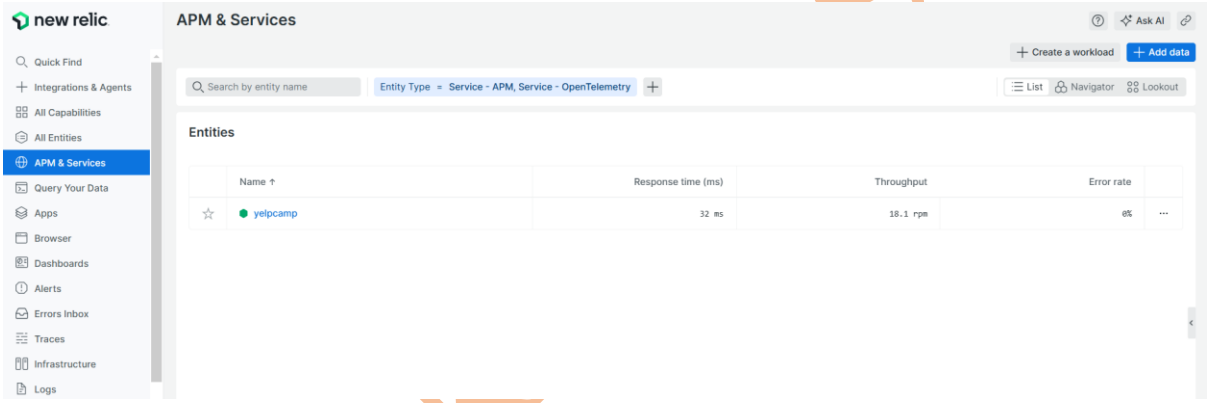
ENV NEW_RELIC_LICENSE_KEY=cXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXL

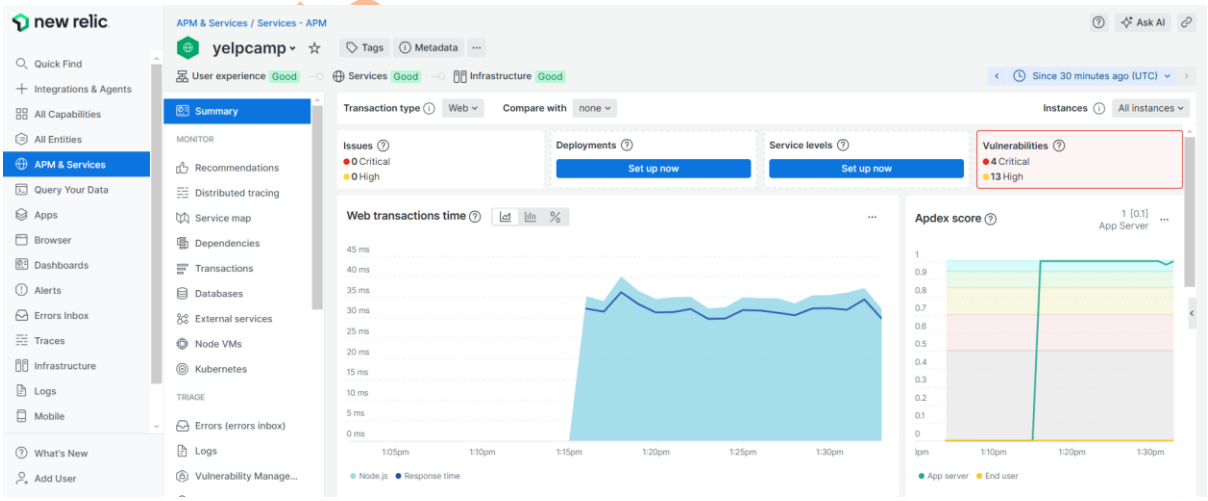
ENV NEW_RELIC_APP_NAME="yelpcamp"

Final entry in the Dockerfile is as shown below.

```
ENV NEW_RELIC_NO_CONFIG_FILE=true
ENV NEW_RELIC_DISTRIBUTED_TRACING_ENABLED=true
ENV NEW_RELIC_LOG=stdout
ENV NEW_RELIC_LICENSE_KEY=c[redacted]L
ENV NEW_RELIC_APP_NAME="yelpcamp"
```

I ran the Jenkins Job to get the changes deployed in the Application and provided the necessary parameters to ran it and check the console for NewRelic APM and found as shown in the screenshot attached below.





Web transaction time is time taken by web application to respond to a request and send back the response.

$$\text{Appdex Score} = \frac{\text{Satisfied Request} + (\text{Tolerated Request} / 2)}{\text{Total Number of Request}}$$

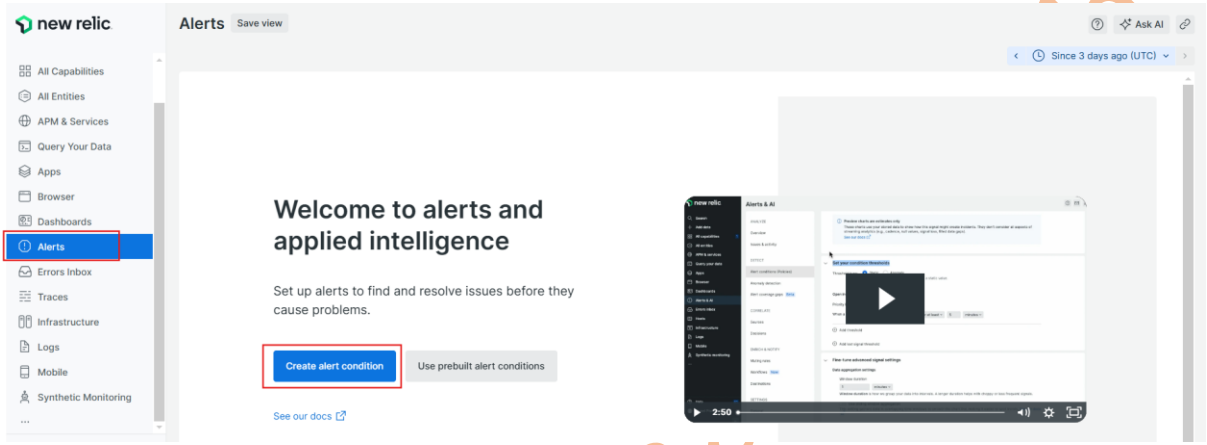
Total Request = Satisfied Request + Tolerated Request + Frustrated Request

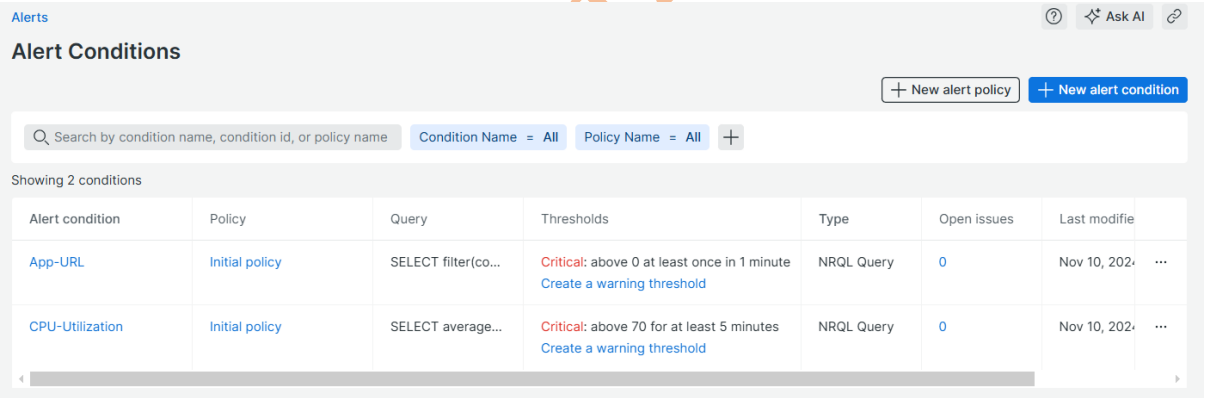
It is calculated to measure the user's satisfaction with the web application and its value varies between 0 to 1. 0 is the worst score and 1 is the best score.

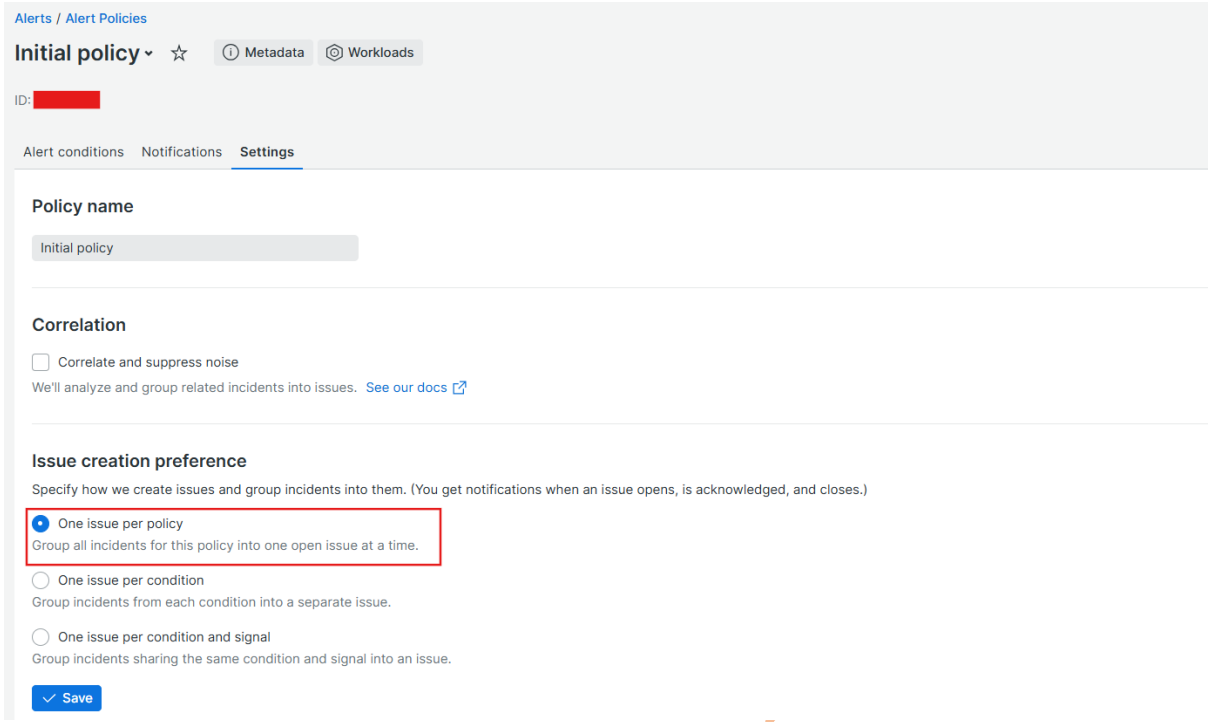
Alert in NewRelic

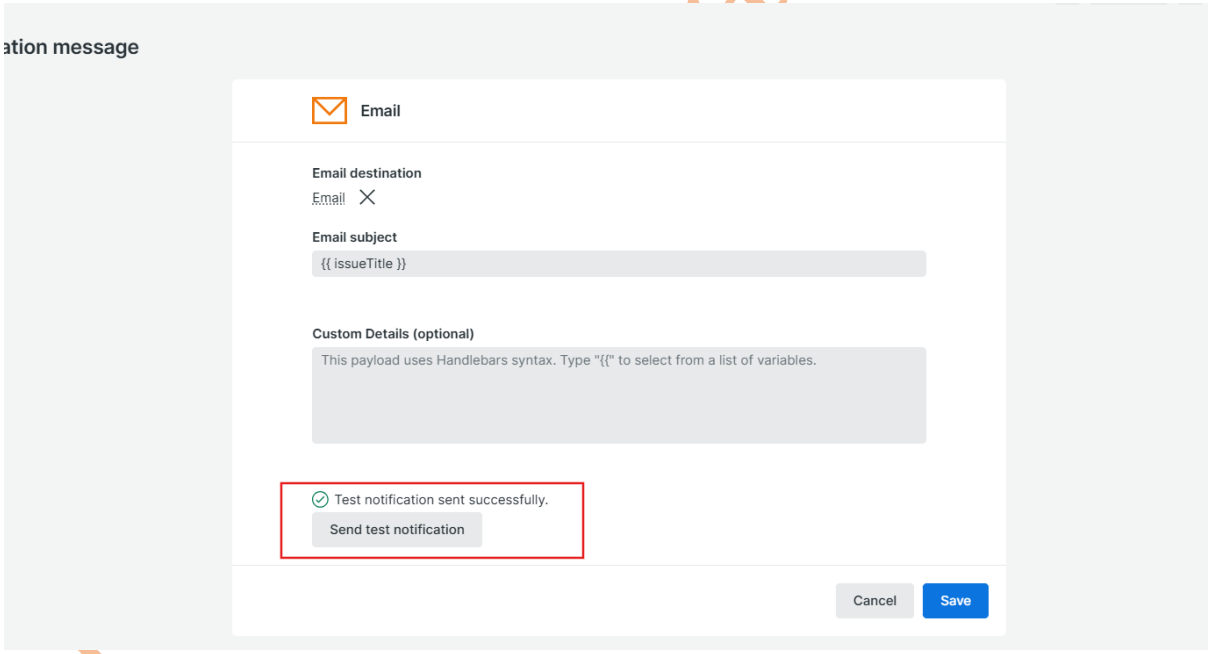
To configure Alerts in NewRelic, I need to configure Alert Conditions, Alert Policies, Workflows and Destination.

To configure Alert Condition, do the configuration as written below.

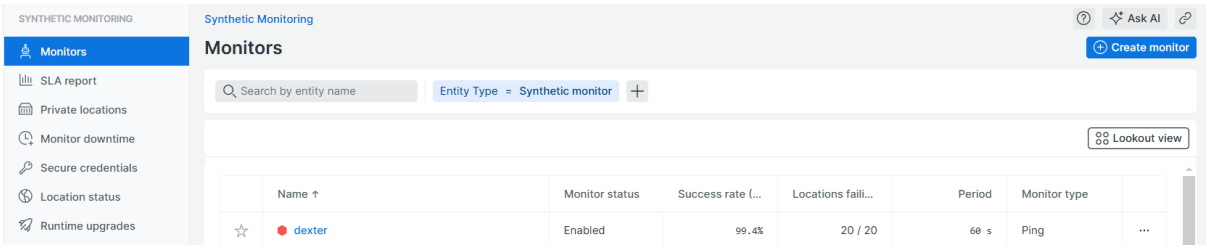


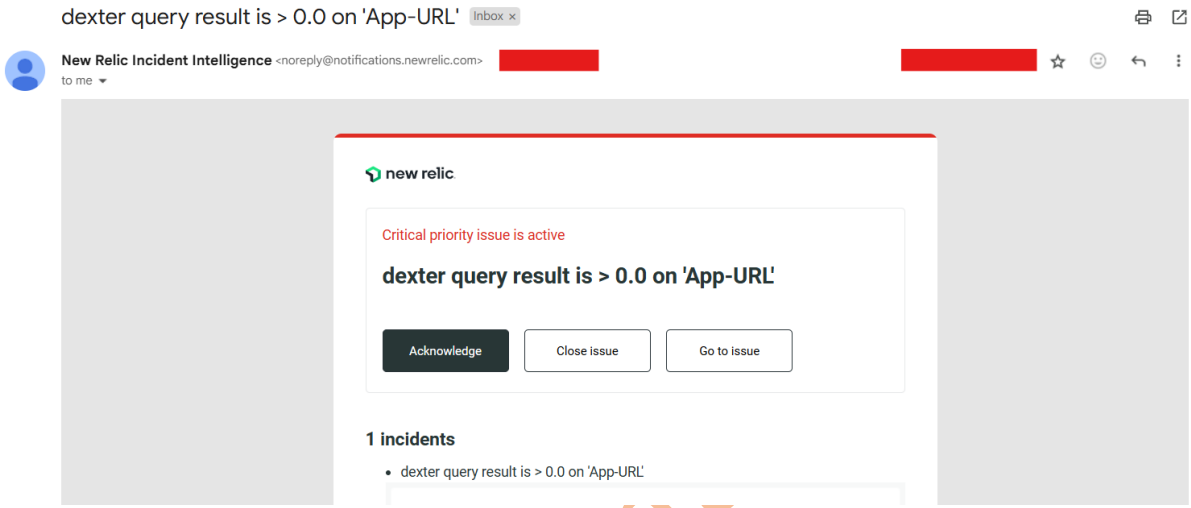




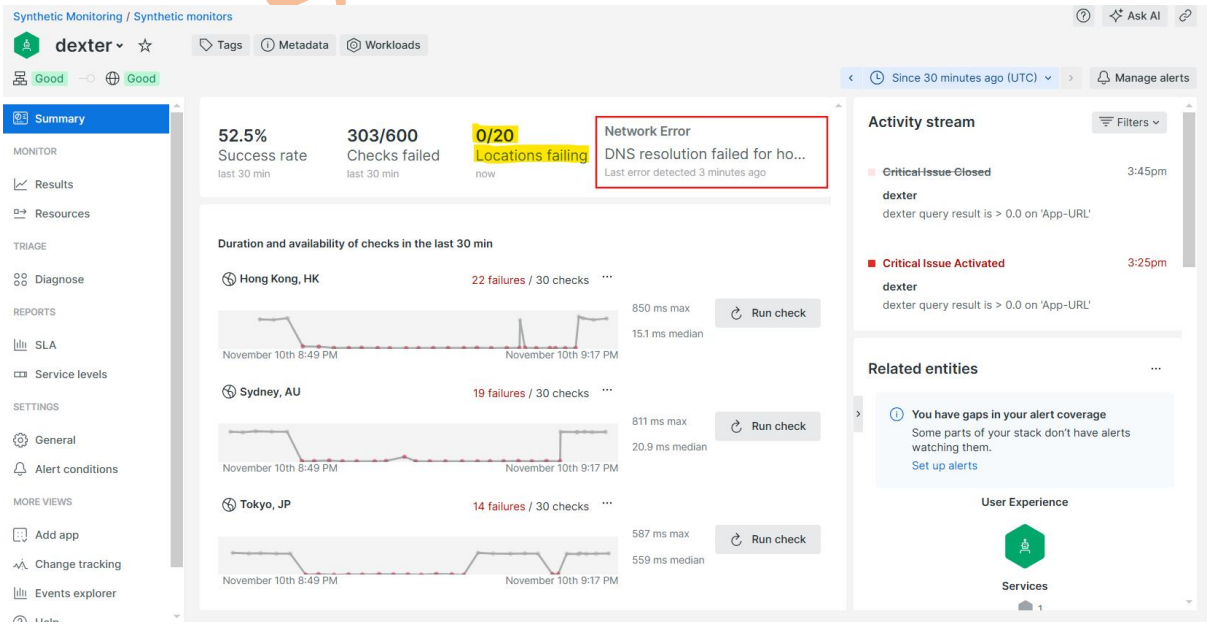


For demonstration purpose I removed the entry of record set of yelp-camp Application URL from Route53 hosted zone and checked that Alert condition comes into critical as it crosses the threshold. And hence an Email will be triggered to destination Email ID as shown in the screenshot attached below.



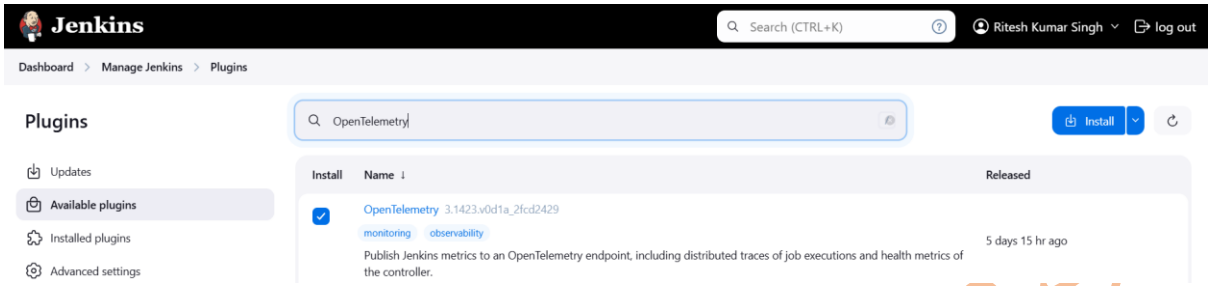


After the concerned team will get notification on your group Email Id, they will Acknowledge the issue and do RCA (Root Cause Analysis). As a part of RCA, they are supposed to check the entry in Route53 hosted zone for record set if it is not present then they will make an entry for the same and this issue will be resolved. It is very important that concerned team should resolve the issue as per the SLA (between the organisation and its client). For demonstration purpose I had deleted the record set form Aws Route53 but in your organization if any one deleted an Aws Resource and you want to find out the person who deleted it then you need to check the CloudTrail > Event history. Then filter out the Event name, finally you will be able to find out the User who deleted that Aws Resource.

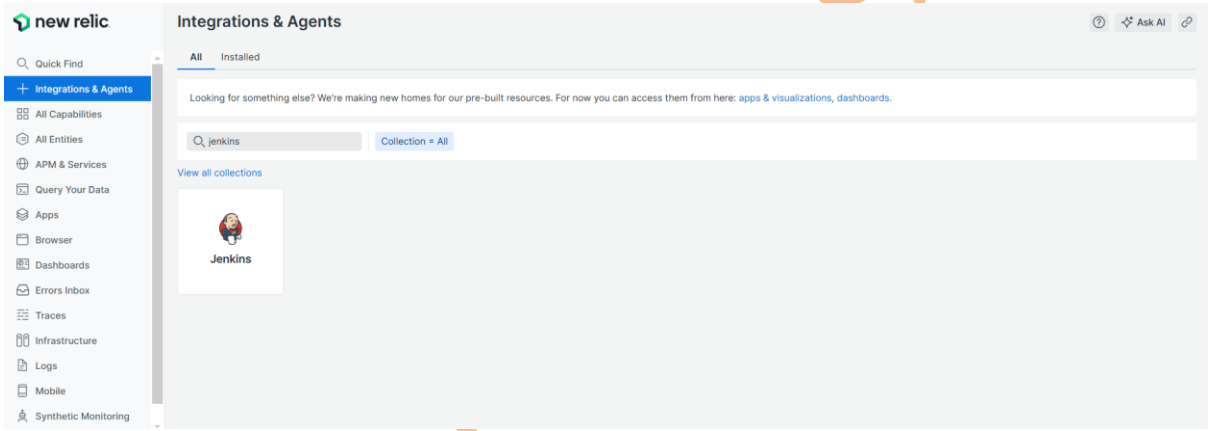


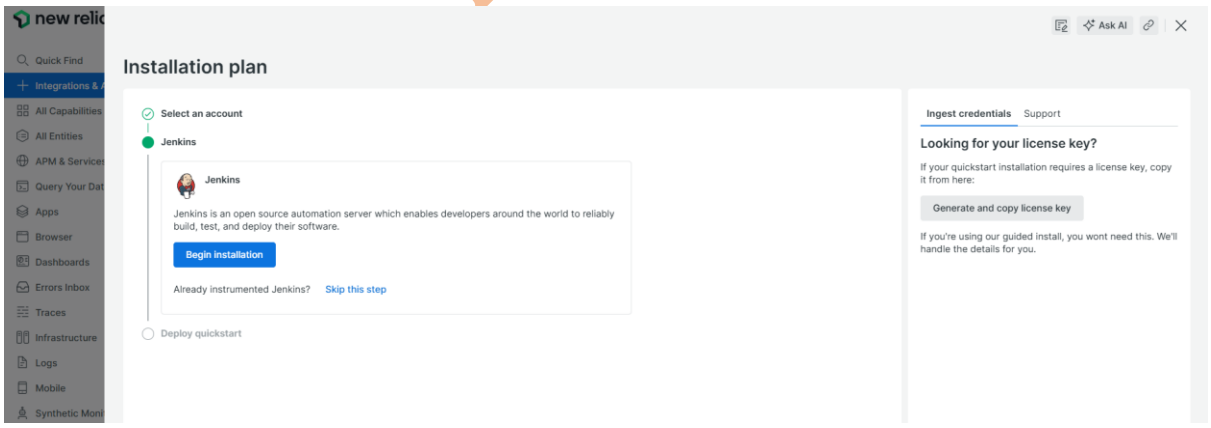
Monitor Jenkins with NewRelic

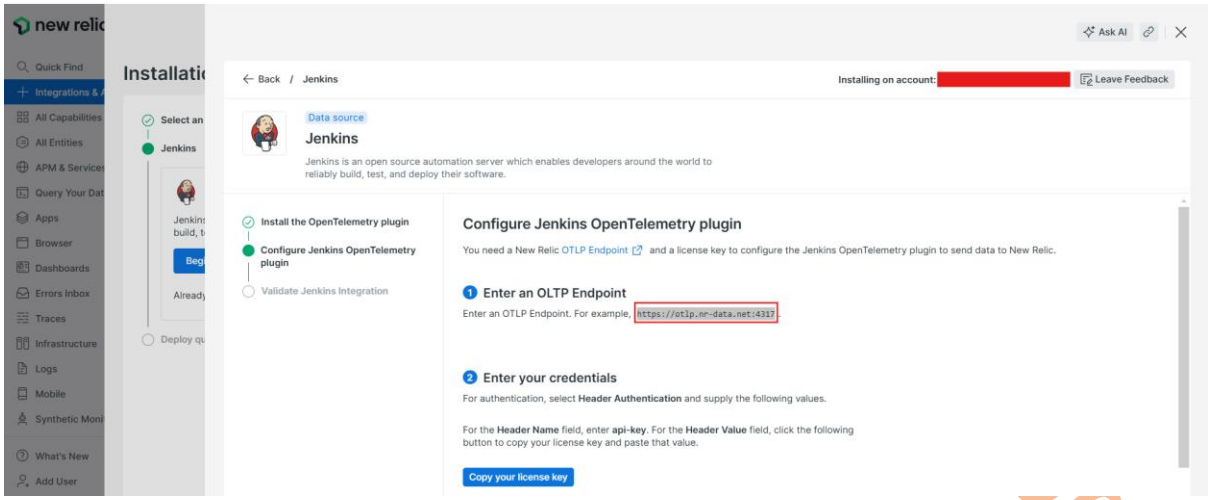
To Monitor Jenkins Job using NewRelic first install the plugin OpenTelemetry in Jenkins as shown in the screenshot attached below.



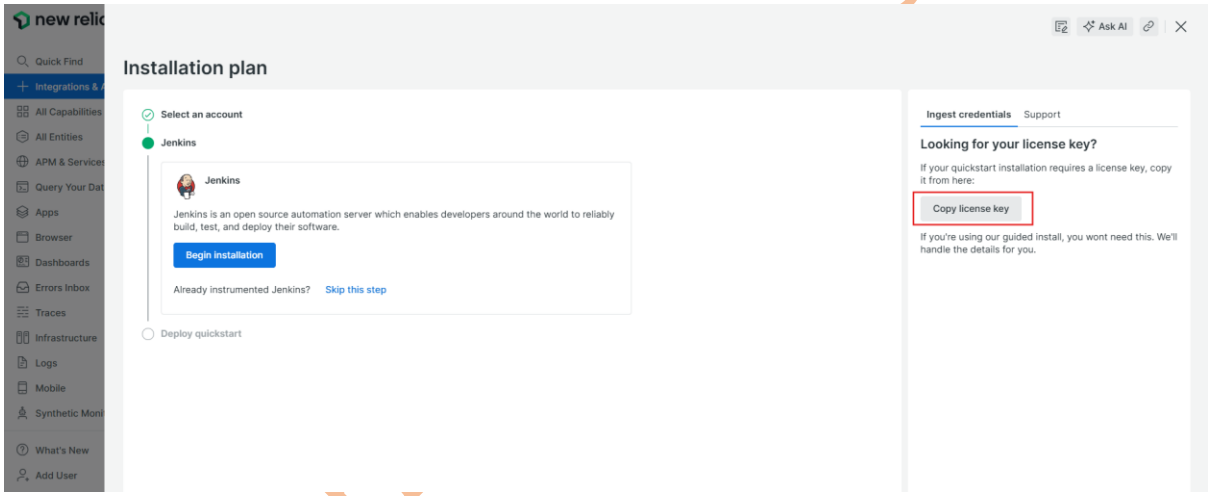
Check the Agent in NewRelic as shown in the screenshot attached below.



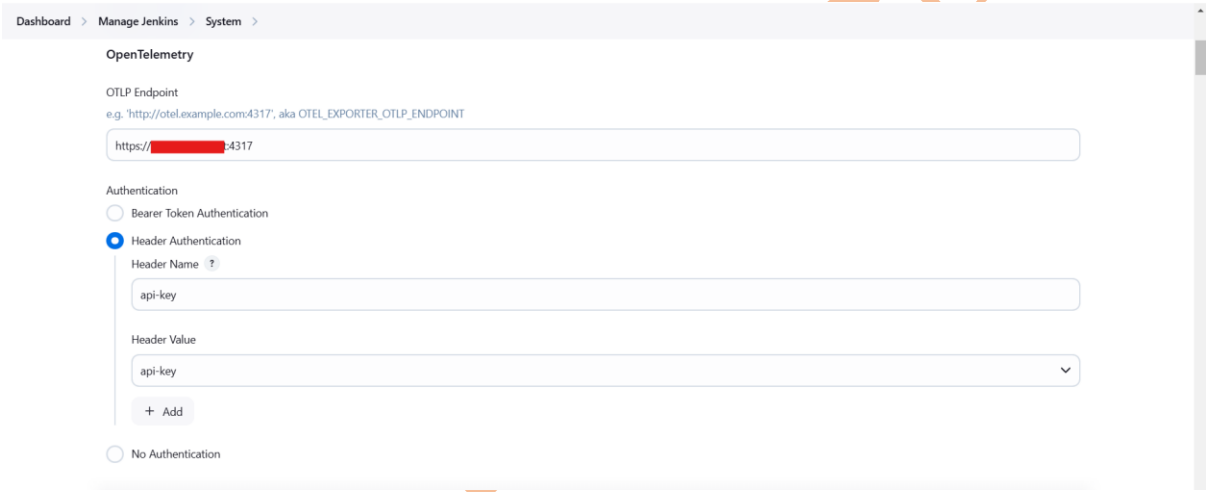




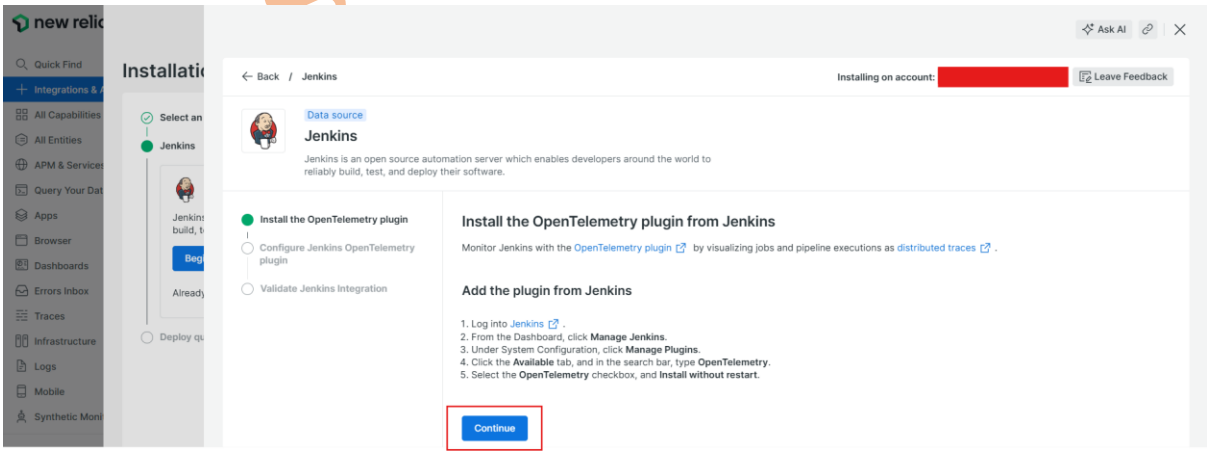
Provide this OLTP endpoint in Jenkins while configuring the Open Telemetry as shown in the screenshot attached below and the API-Key (License Key) copied from the step as shown below.

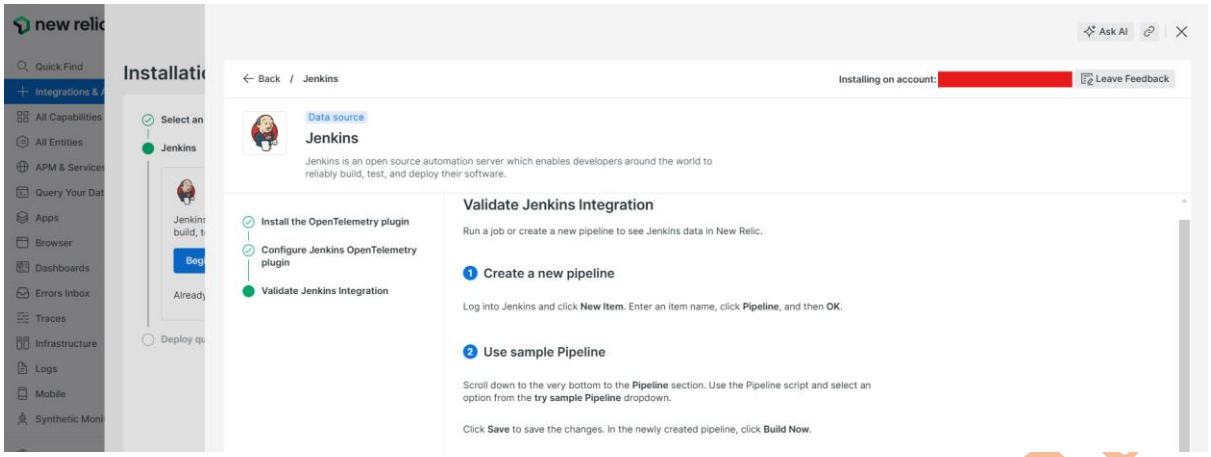


Ritesh Singh

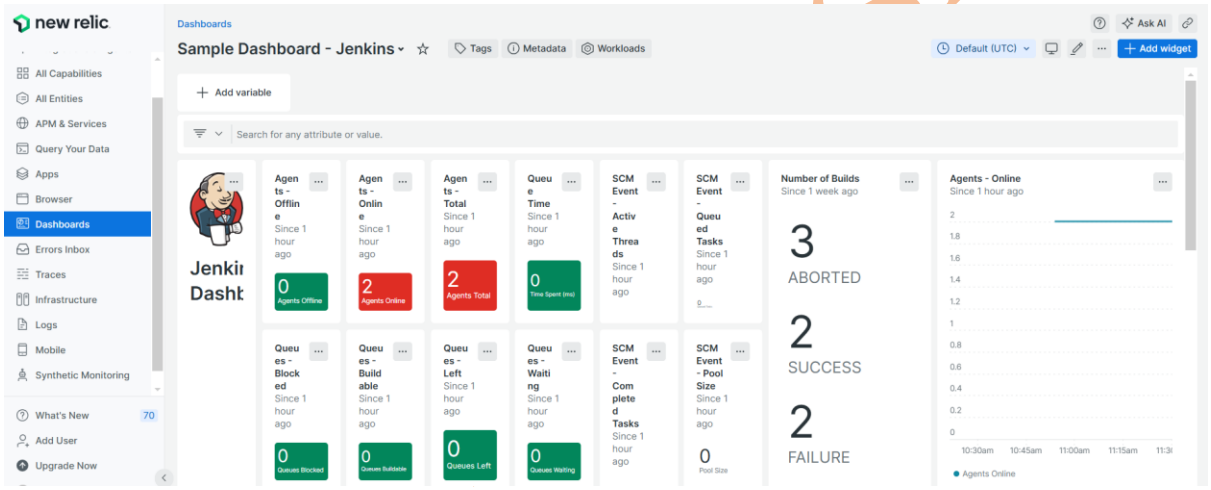


Now proceed for Jenkins Configuration in NewRelic as shown in the screenshot attached below.





For the demonstration purpose I ran the Jenkins Job 7 times after providing the required parameters to run it and below is the screenshot of the NewRelic Jenkins Dashboard for Monitoring of Jenkins.

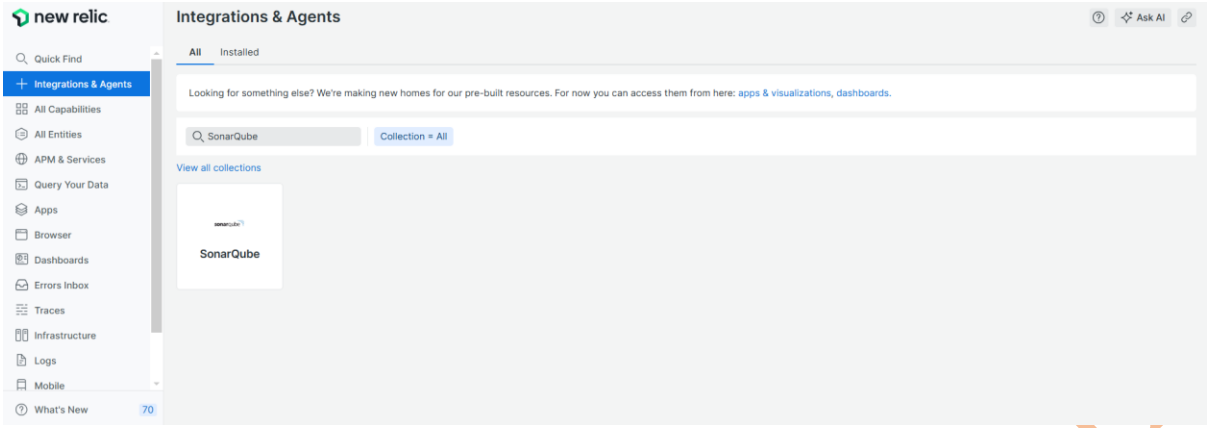


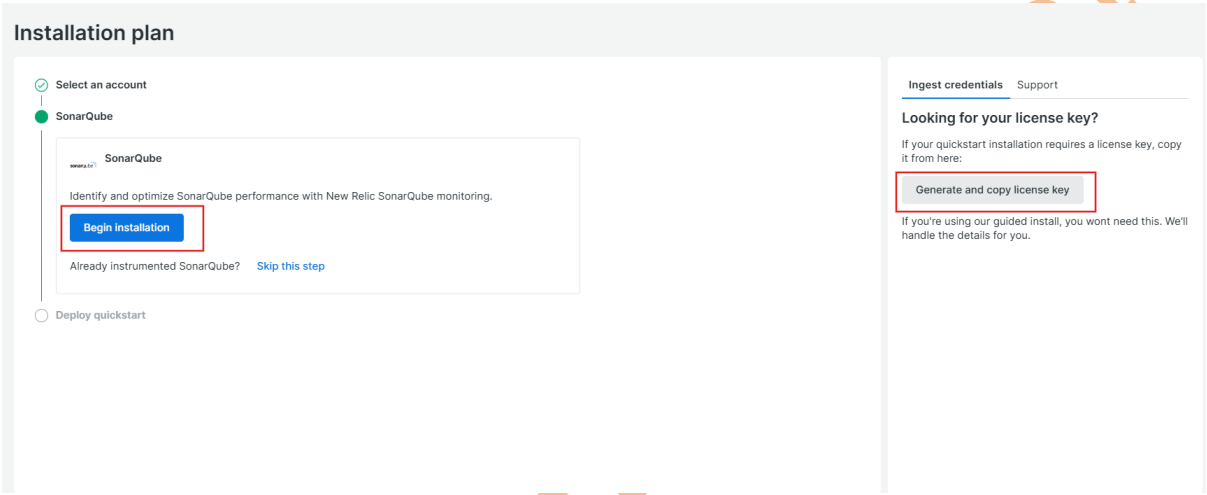
It shows two Agents are Online line (One Jenkins Master + One Jenkins Slave), I ran the Jenkins Job and made it failed and two times, ran successfully two times and aborted it three times which is shown in the screenshot above.

Monitoring SonarQube with NewRelic

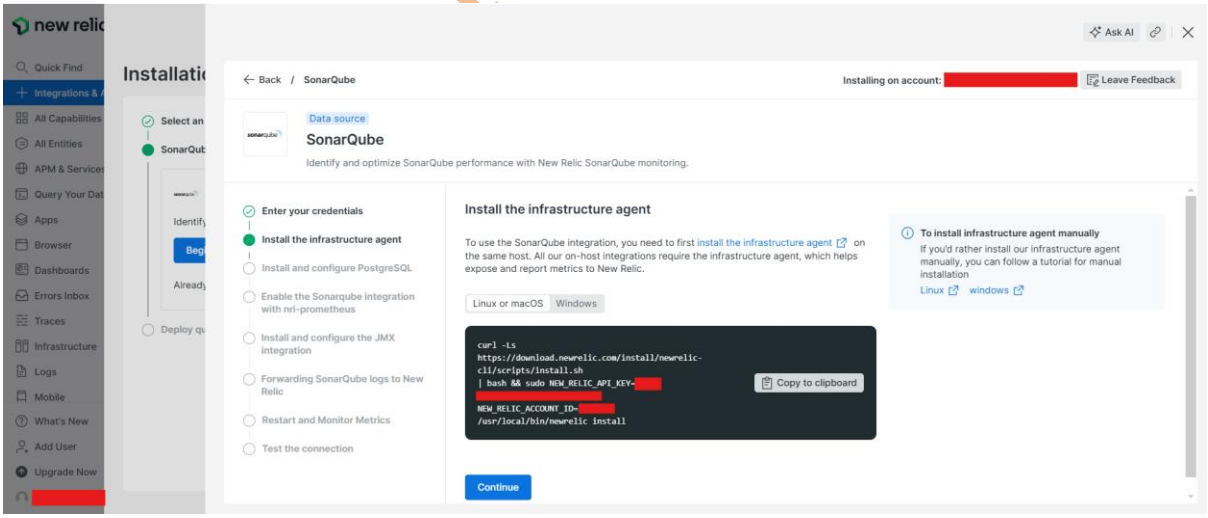
To Monitor SonarQube with NewRelic install the Agent of SonarQube for NewRelic as shown in the screenshot attached below.

Pitesh Singh





Install the NewRelic Agent on SonarQube as shown in the screenshot attached below.




```
[root@ opt]# curl -Ls https://download.newrelic.com/install/newrelic-cli/scripts/install.sh | bash && sudo NEW_RELIC_API_KEY=
NEW_RELIC_ACCOUNT_ID= /usr/local/bin/newrelic install
```

Provide the necessary details as shown in the screenshot attached below and create a file at the path /etc/newrelic-infra/integrations.d/nri-prometheus-config.yml

cat /etc/newrelic-infra/integrations.d/nri-prometheus-config.yml

```
integrations:
  - name: nri-prometheus
    config:
      # When standalone is set to false nri-prometheus requires an infrastructure agent to work and send data. Defaults to true
      standalone: false
      # When running with infrastructure agent emitters will have to include infra-sdk
      emitters: infra-sdk
      # The name of your cluster. It's important to match other New Relic products to relate the data.
      cluster_name: "sonarqube"
      targets:
        - description: Sonarqube metrics list
          urls: ["http://aXXXn:AXXXXXX3@X.XXX.XXX.XXX:9000/api/monitoring/metrics"]
      #     tls_config:
      #     ca_file_path: "/etc/etcd/etcd-client-ca.crt"
      #     cert_file_path: "/etc/etcd/etcd-client.crt"
      #      key_file_path: "/etc/etcd/etcd-client.key"
      # Whether the integration should run in verbose mode or not. Defaults to false
      verbose: false
      # Whether the integration should run in audit mode or not. Defaults to false.
      # Audit mode logs the uncompressed data sent to New Relic. Use this to log all data sent.
      # It does not include verbose mode. This can lead to a high log volume, use with care
      audit: false
      # The HTTP client timeout when fetching data from endpoints. Defaults to 30s.
      # scrape_timeout: "30s"
      # Length in time to distribute the scraping from the endpoints
      scrape_duration: "5s"
      # Number of worker threads used for scraping targets.
      # For large clusters with many (&gt;400) endpoints, slowly increase until scrape
      # time falls between the desired `scrape_duration`.
      # Increasing this value too much will result in huge memory consumption if too
      # many metrics are being scraped.
      # Default: 4
      # worker_threads: 4
      # Whether the integration should skip TLS verification or not. Defaults to false
      insecure_skip_verify: true
    timeout: 10s
```

Ritesh Kumar Singh

```
[root@ip-172-31-18-254 opt]# cat /etc/newrelic-infra/integrations.d/nri-prometheus-config.yml
integrations:
  - name: nri-prometheus
    config:
      # When standalone is set to false nri-prometheus requires an infrastructure agent to work and send data. Defaults to true
      standalone: false
      # When running with infrastructure agent emitters will have to include infra-sdk
      emitters: infra-sdk
      # The name of your cluster. It's important to match other New Relic products to relate the data.
      cluster_name: "sonarqube"
      targets:
        - description: Sonarqube metrics list
          urls: ["http://a[redacted]h:A[redacted]3@[redacted]:9000/api/monitoring/metrics"]
      #     tls_config:
      #     ca_file_path: "/etc/etcd/etcd-client-ca.crt"
      #     cert_file_path: "/etc/etcd/etcd-client.crt"
      #      key_file_path: "/etc/etcd/etcd-client.key"
      # Whether the integration should run in verbose mode or not. Defaults to false
      verbose: false
      # Whether the integration should run in audit mode or not. Defaults to false.
      # Audit mode logs the uncompressed data sent to New Relic. Use this to log all data sent.
      # It does not include verbose mode. This can lead to a high log volume, use with care
      audit: false
      # The HTTP client timeout when fetching data from endpoints. Defaults to 30s.
      # scrape_timeout: "30s"
      # Length in time to distribute the scraping from the endpoints
      scrape_duration: "5s"
      # Number of worker threads used for scraping targets.
      # For large clusters with many (&gt;400) endpoints, slowly increase until scrape
      # time falls between the desired `scrape_duration`.
      # Increasing this value too much will result in huge memory consumption if too
      # many metrics are being scraped.
      # Default: 4
      # worker_threads: 4
      # Whether the integration should skip TLS verification or not. Defaults to false
      insecure_skip_verify: true
    timeout: 10s
```

Ritesh Kumar

To configure logging edit the file as shown in the screenshot attached below present at the path /etc/newrelic-infra/logging.d/logging.yml

```
logs:
  - name: cloud-init.log
    file: /var/log/cloud-init.log
    attributes:
      logtype: linux_cloud-init
  - name: messages
    file: /var/log/messages
    attributes:
      logtype: linux_messages
  - name: secure
    file: /var/log/secure
    attributes:
      logtype: linux_secure
  - name: yum.log
    file: /var/log/yum.log
    attributes:
      logtype: linux_yum
  - name: newrelic-cli.log
    file: /root/.newrelic/newrelic-cli.log
    attributes:
      newrelic-cli: true
      logtype: newrelic-cli
  - name: sonar_logs
    file: /opt/sonarqube/logs/sonar.log
    attributes:
      logtype: sonar_logs
  - name: ce_logs
    file: /opt/sonarqube/logs/ce.log
    attributes:
      logtype: sonar_ce_logs
  - name: es_logs
    file: /opt/sonarqube/logs/es.log
    attributes:
      logtype: sonar_es_logs
  - name: web_logs
    file: /opt/sonarqube/logs/web.log
    attributes:
      logtype: sonar_web_logs
```

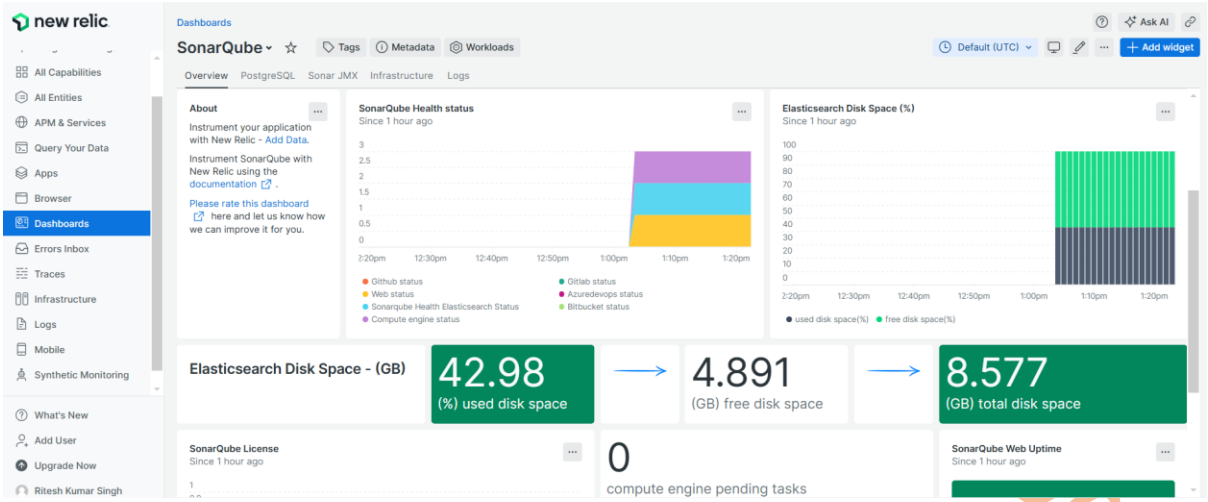
Ritesh Kumar Singh

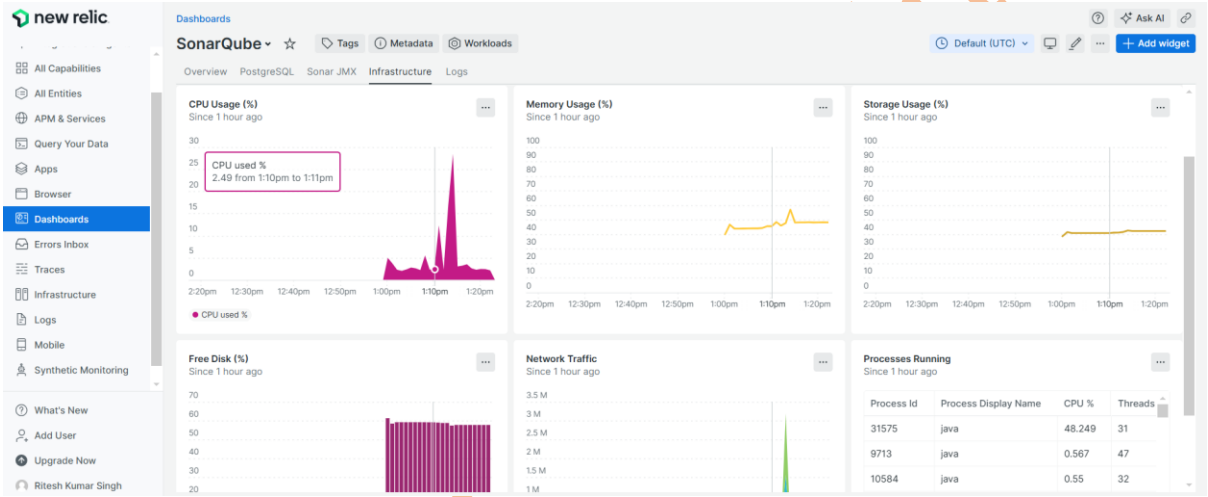
```
[root@[redacted]pt]# cat /etc/newrelic-infra/logging.d/logging.yml
logs:
  - name: cloud-init.log
    file: /var/log/cloud-init.log
    attributes:
      logtype: linux_cloud-init
  - name: messages
    file: /var/log/messages
    attributes:
      logtype: linux_messages
  - name: secure
    file: /var/log/secure
    attributes:
      logtype: linux_secure
  - name: yum.log
    file: /var/log/yum.log
    attributes:
      logtype: linux_yum
  - name: newrelic-cli.log
    file: /root/.newrelic/newrelic-cli.log
    attributes:
      newrelic-cli: true
      logtype: newrelic-cli
  - name: sonar_logs
    file: /opt/sonarqube/logs/sonar.log
    attributes:
      logtype: sonar_logs
  - name: ce_logs
    file: /opt/sonarqube/logs/ce.log
    attributes:
      logtype: sonar_ce_logs
  - name: es_logs
    file: /opt/sonarqube/logs/es.log
    attributes:
      logtype: sonar_es_logs
  - name: web_logs
    file: /opt/sonarqube/logs/web.log
    attributes:
      logtype: sonar_web_logs
```

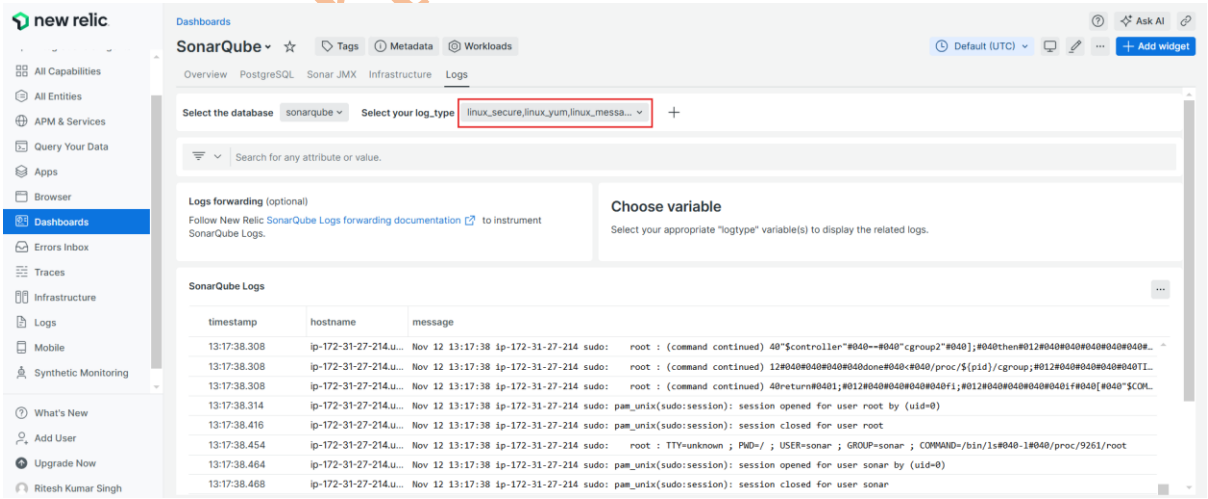
After the above-mentioned configuration restarted the NewRelic service and checked its status as shown in the screenshot attached below.

```
[root@[redacted] opt]# sudo systemctl restart newrelic-infra
[root@[redacted] opt]# sudo systemctl status newrelic-infra
● newrelic-infra.service - New Relic Infrastructure Agent
   Loaded: loaded (/etc/systemd/system/newrelic-infra.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2024-11-11 12:20:34 UTC; 5s ago
```

Finally, you will be able to monitor SonarQube as shown in the screenshot attached below.


new relic
Dashboards
SonarQube
Tags
Metadata
Workloads
Ask AI
Default (UTC)
Add widget
Overview
PostgreSQL
Sonar JMX
Infrastructure
Logs
All Capabilities
All Entities
APM & Services
Query Your Data
Apps
Browser
Dashboards
Errors Inbox
Traces
Infrastructure
Logs
Mobile
Synthetic Monitoring
What's New
Add User
Upgrade Now
Ritesh Kumar Singh
About
Instrument your application with New Relic - Add Data.
Instrument SonarQube with New Relic using the documentation .
Please rate this dashboard here and let us know how we can improve it for you.
SonarQube Health status
Since 1 hour ago
Github status
Web status
Sonarqube Health Elasticsearch Status
Compute engine status
Gitlab status
Azuredevops status
Bitbucket status
Elasticsearch Disk Space (%)
Since 1 hour ago
used disk space(%)
free disk space(%)
Elasticsearch Disk Space - (GB)
42.98
(%) used disk space
4.891
(GB) free disk space
8.577
(GB) total disk space
SonarQube License
Since 1 hour ago
0
compute engine pending tasks
SonarQube Web Uptime
Since 1 hour ago


new relic
Dashboards
SonarQube
Tags
Metadata
Workloads
Ask AI
Default (UTC)
Add widget
Overview
PostgreSQL
Sonar JMX
Infrastructure
Logs
CPU Usage (%)
Since 1 hour ago
CPU used %
2.49 from 1:10pm to 1:11pm
Memory Usage (%)
Since 1 hour ago
Storage Usage (%)
Since 1 hour ago
Free Disk (%)
Since 1 hour ago
Network Traffic
Since 1 hour ago
Processes Running
Since 1 hour ago
Process Id
Process Display Name
CPU %
Threads
31575
java
48.249
31
9713
java
0.567
47
10584
java
0.55
32

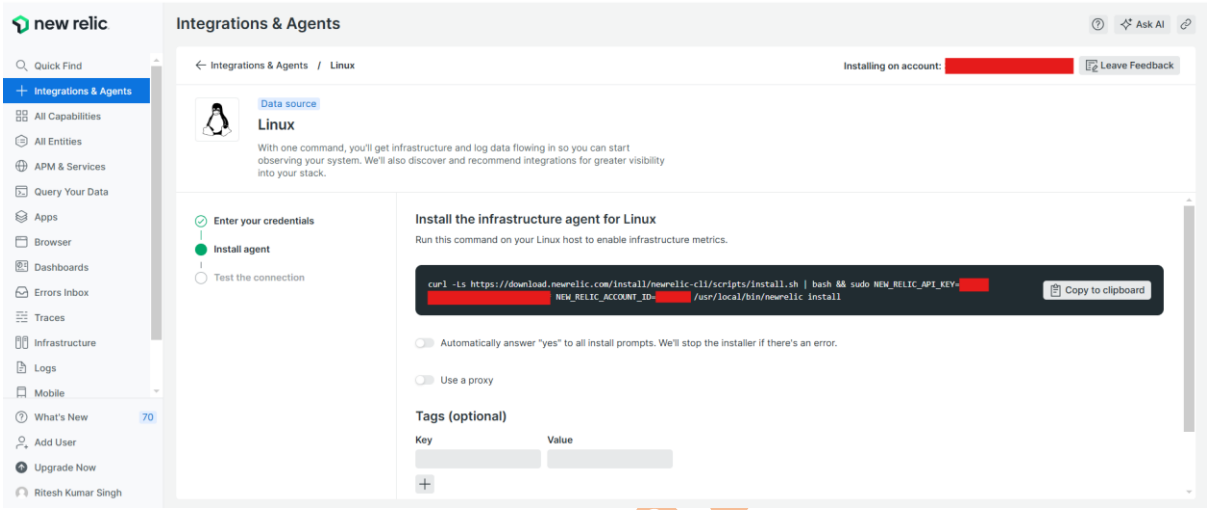

new relic
Dashboards
SonarQube
Tags
Metadata
Workloads
Ask AI
Default (UTC)
Add widget
Overview
PostgreSQL
Sonar JMX
Infrastructure
Logs
Select the database
sonarqube
Select your log_type
linux_secure,linux_yum,linux_messa...
Search for any attribute or value.
Logs forwarding (optional)
Follow New Relic SonarQube Logs forwarding documentation to instrument SonarQube Logs.
Choose variable
Select your appropriate "logtype" variable(s) to display the related logs.
SonarQube Logs
timestamp
hostname
message

Configure Log aggregation in Jenkins Master and Jenkins Slave

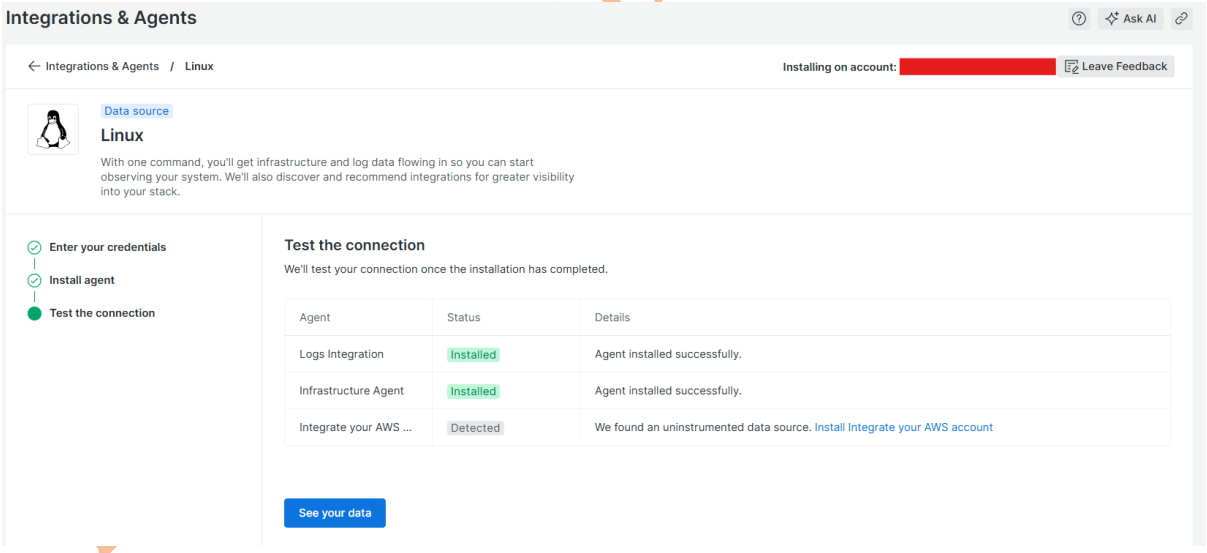
To configure Log aggregation in Jenkins Master and Jenkins Slave follow the procedure as written below.

Jenkins Master

Install NewRelic Agent on Jenkins Master as discussed above and it is also shown in the screenshot attached below.




```
[root@                ~]# curl -Ls https://download.newrelic.com/install/newrelic-cli/scripts/install.sh | bash && sudo NEW_RELIC_API_KEY=
                NEW_RELIC_ACCOUNT_ID=          /usr/local/bin/newrelic install
```



Open the file /etc/newrelic-infra/logging.d/logging.yml and updated with the configuration as shown in the screenshot attached below.

- name: jenkins-health-checker-log

  file: /var/lib/jenkins/logs/health-checker.log

  attributes:

   logtype: jenkins

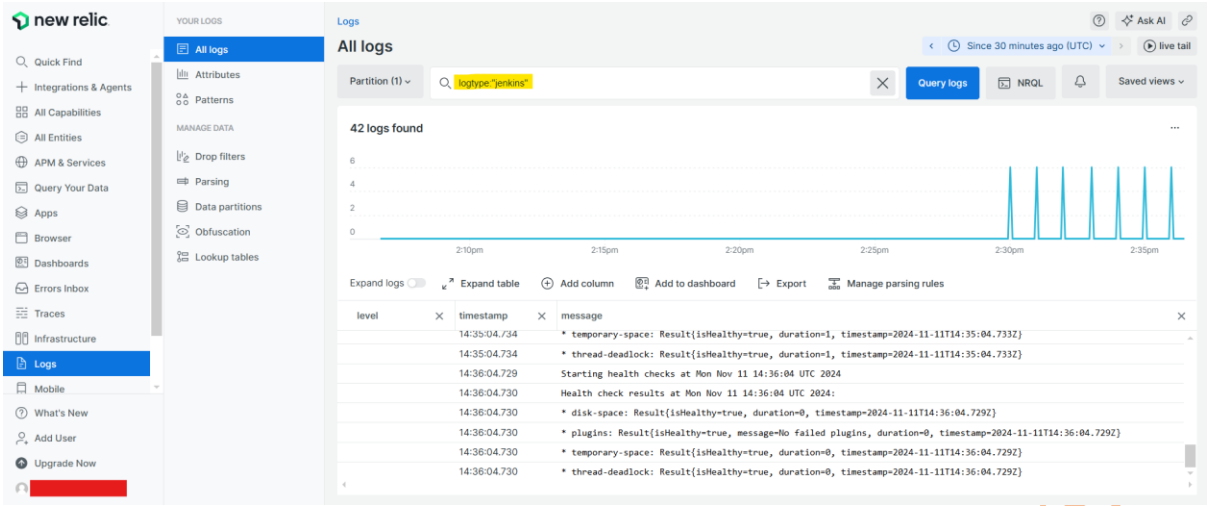
```
[root@[redacted] ~]# cat /etc/newrelic-infra/logging.d/logging.yml
logs:
  - name: cloud-init.log
    file: /var/log/cloud-init.log
    attributes:
      logtype: linux_cloud-init
  - name: messages
    file: /var/log/messages
    attributes:
      logtype: linux_messages
  - name: secure
    file: /var/log/secure
    attributes:
      logtype: linux_secure
  - name: yum.log
    file: /var/log/yum.log
    attributes:
      logtype: linux_yum
  - name: newrelic-cli.log
    file: /root/.newrelic/newrelic-cli.log
    attributes:
      newrelic-cli: true
      logtype: newrelic-cli
  - name: jenkins-health-checker-log
    file: /var/lib/jenkins/logs/health-checker.log
    attributes:
      logtype: jenkins
```

You can also capture the Slaves and Jenkins Job related logs from the path /var/lib/jenkins/logs but here I am only capturing the health-checker logs as shown in the screenshot attached above.

Ritesh

Then restart the newrelic service as shown in the screenshot attached below finally, check the logs in newrelic console and filter logtype as jenkins.

```
[root@[redacted] ~]# systemctl restart newrelic-infra.service
[root@[redacted] ~]# systemctl status newrelic-infra.service
● newrelic-infra.service - New Relic Infrastructure Agent
   Loaded: loaded (/etc/systemd/system/newrelic-infra.service; enabled; vendor preset: disabled
   Active: active (running) since Mon 2024-11-11 14:33:45 UTC; 5s ago
```



Jenkins Slave

Install NewRelic Agent on Jenkins Slave as discussed above and it is also shown in the screenshot attached below.

```
[root@[redacted] opt]# curl -Ls https://download.newrelic.com/install/newrelic-cli/scripts/install.sh | bash && sudo NEW_RELIC_API_KEY=[redacted]
[redacted] NEW_RELIC_ACCOUNT_ID=[redacted] /usr/local/bin/newrelic install
```

```
[root@[redacted] opt]# cat /etc/newrelic-infra/logging.d/logging.yml
logs:
  - name: cloud-init.log
    file: /var/log/cloud-init.log
    attributes:
      logtype: linux_cloud-init
  - name: messages
    file: /var/log/messages
    attributes:
      logtype: linux_messages
  - name: secure
    file: /var/log/secure
    attributes:
      logtype: linux_secure
  - name: yum.log
    file: /var/log/yum.log
    attributes:
      logtype: linux_yum
  - name: newrelic-cli.log
    file: /root/.newrelic/newrelic-cli.log
    attributes:
      newrelic-cli: true
      logtype: newrelic-cli
```

If you need to aggregate logs from any specific file to NewRelic then provide its path as mentioned below.

- name: <provide a suitable name>

  file: <provide the path of the log file>

  attributes:

    logtype: <provide a log type here>

Finally restart and check the status the NewRelic Service using the command systemctl restart newrelic-infra and systemctl status newrelic-infra respectively.

Ritesh Kumar Singh

Jenkinsfile to create jenkins declarative pipeline is as shown below.

Jenkinsfile

```
pipeline{
   agent{
      node{
         label "Slave-1"
         customWorkspace "/home/jenkins/demo"
      }
   }
   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/YelpCamp.git']])
         }
      }
      stage("SonarAnalysis"){
         steps {
            withSonarQubeEnv('SonarQube-Server') {
```

Ritesh kumar Singh

```
            sh 'sonar-scanner -Dsonar.projectKey=YelpCamp -Dsonar.projectName=YelpCamp'
          }
        }
      }
      stage("Quality Gate") {
        steps {
          timeout(time: 1, unit: 'HOURS') {
            waitForQualityGate abortPipeline: true
          }
        }
      }
      stage("Install Dependencies"){
        steps {
          sh 'npm install'
        }
      }
      stage("Trivy Scan files"){
        steps{
          sh 'trivy fs . > /home/jenkins/trivy-filescan.txt'
        }
      }
      stage("Docker-Image"){
        steps{
          sh 'docker build -t ${REPO_NAME}:${TAG_NAME} . --no-cache'
          sh 'trivy image --exit-code 0 --severity MEDIUM,HIGH ${REPO_NAME}:${TAG_NAME}'
          //sh 'trivy image  --exit-code 1 --severity CRITICAL ${REPO_NAME}:${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}'
        }
      }
```

Ritesh Kumar Singh

```
        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 yelp-camp --project default --repo
https://github.com/singhritesh85/helm-chart-yelp-camp.git --path ./folo --dest-namespace yelp-
camp --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 yelp-camp'
            }
        }
    }
}
```

Ritesh Kumar Singh

Reference: - https://github.com/Colt/YelpCamp.git