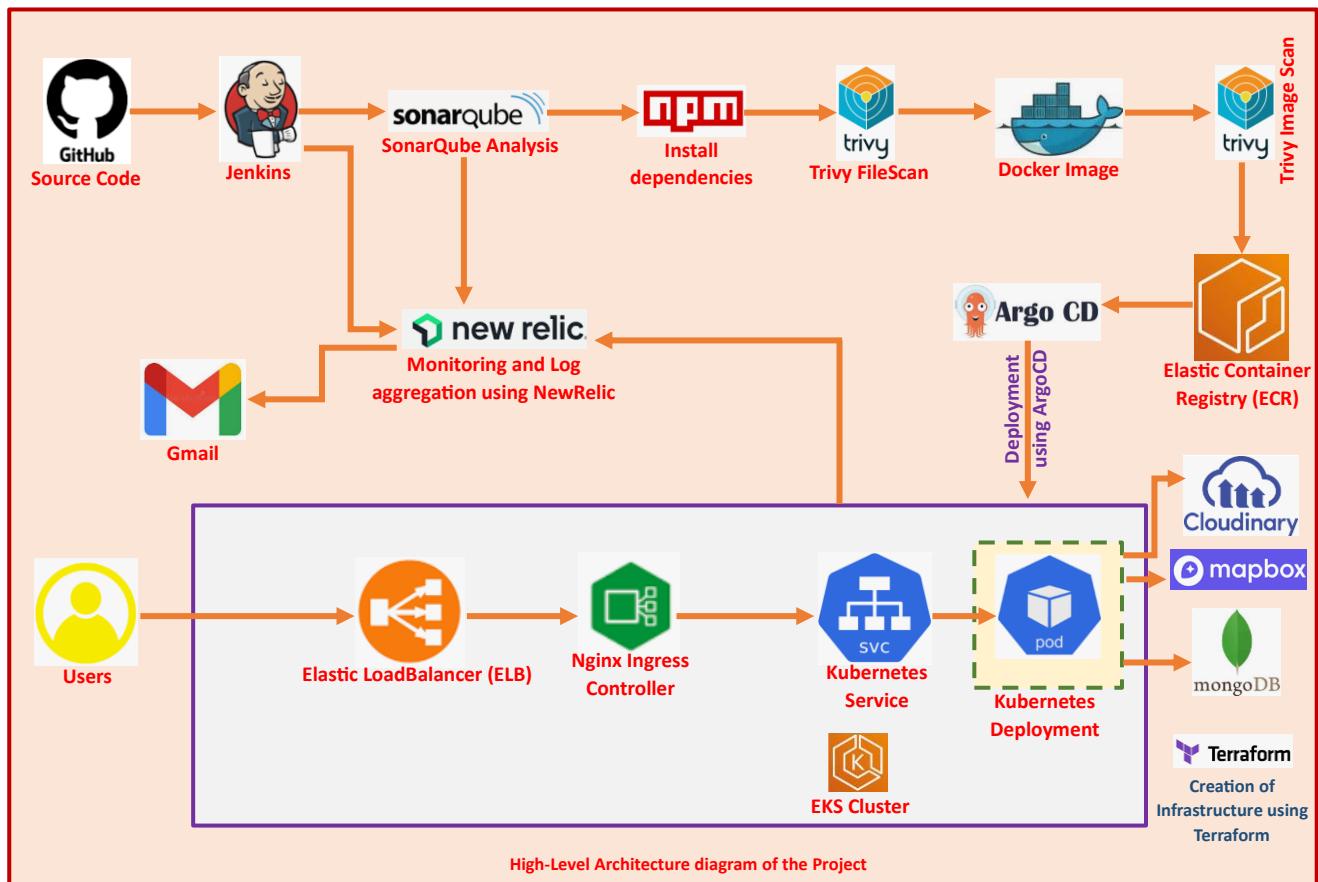


## DevOps Project Yelp-Camp



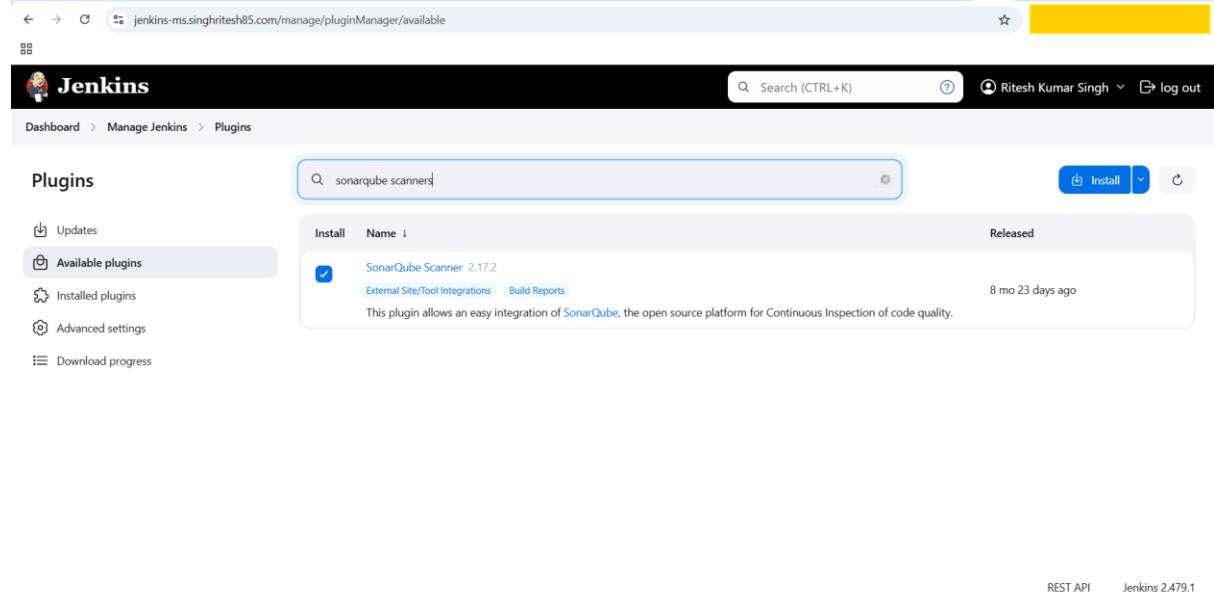
This DevOps Project aims to create the infrastructure of EKS Cluster, Elastic Container Registry (ECR), Jenkins Master, Jenkins Slave and SonarQube Server with the help of Terraform. Creation of CI/CD pipeline using Jenkins for the purpose of Deployment of the Application. For Monitoring and Log Aggregation 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.



The screenshot shows the Jenkins Plugin Manager interface. In the search bar, 'sonarqube scanners' is typed. A single result, 'SonarQube Scanner 2.17.2', is listed under the 'Available plugins' section. The plugin is described as allowing for easy integration of SonarQube, an open-source platform for continuous inspection of code quality. It was released 8 months and 23 days ago. There is an 'Install' button next to the listing.

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 c...v|base64  
ZGtiZmx5YmF2  
  
[root@... ~]# echo -n 4...1 | base64  
NDc3OTYxNzU4OTc3MjMx  
  
[root@... ~]# echo -n R...4 | base64  
U1pfCkDbmhqZDEysU51eVVHVhaSkNHWHU0  
[root@... ~]# echo -n "mongodb+srv://  
bw9uZ29kYitzcnY6Ly9tb25nbzpBZG1pbjEyM0BkZXh0ZXIudWUwcW4ubW9uZ29kYi5uZXQvP3J1  
dH05J3JpdGVzPXRydWUmdz1tYwpvcml0eSzhcHBOYW1lPWR1ehRlcge=  
[root@... ~]# echo -n e...s | base64  
ZXhwubGljYXR1ZGV2b3Bz
```

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.

The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with various services like Stream Processing, Triggers, Migration, Data Federation, Security, and Network Access. The main area is titled 'Clusters' and shows a single cluster named 'dexter'. It includes sections for 'Data Services', 'Clusters', and 'Enhance Your Exp...'. A modal window is open in the center, providing instructions for connecting to the cluster. It says '2. Install your driver' and 'Run the following on the command line': `npm install mongodb`. Below that, it says '3. Add your connection string into your application code' and provides a code snippet: `mongodb://mongo:<db_password>@dexter.ueqgn.mongodb.net/?retryWrites=true&w=majority&appName=dexter`. It also says to replace `<db_password>` with the password for the `mongo` database user. At the bottom of the modal are 'Go Back' and 'Done' buttons.

## 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

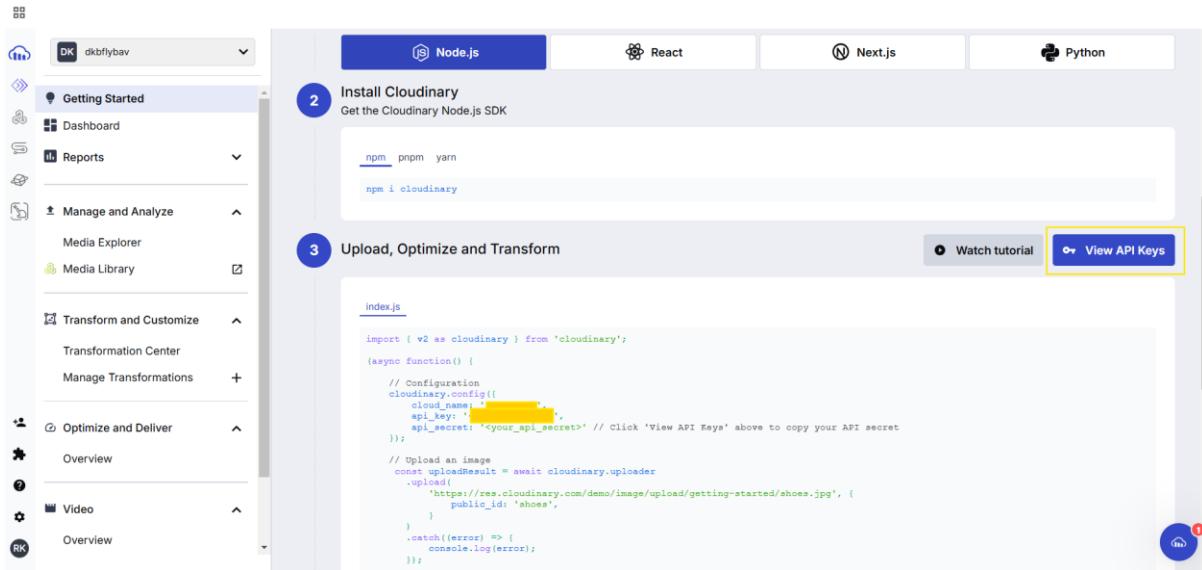
[⟳ Refresh](#)



**Last modified:** less than a minute ago

**URLs:** N/A

Ry



## 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

```

| NAME                               | AGE             | TYPE         | CLUSTER-IP | EXTERNAL-IP | PORT(S)                   |
|------------------------------------|-----------------|--------------|------------|-------------|---------------------------|
| ingress-nginx-controller           | 3: redacted/TCP | LoadBalancer | redacted   | redacted    | 80: redacted/TCP, 443/TCP |
| ingress-nginx-controller-admission |                 | ClusterIP    | redacted   | <none>      | 443/TCP                   |

```

[root@redacted ~]# kubectl edit svc ingress-nginx-controller -n ingress-nginx
service/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]: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<sup>®</sup>

```
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:XXXXXXXXX:certificate/XXXXXX-XXXXXX-XXXXXX-XXXXXX  
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
```

## Installation of ArgoCD

To install ArgoCD follow the procedure as mentioned below.

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

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

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

kubectl create namespace argocd

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

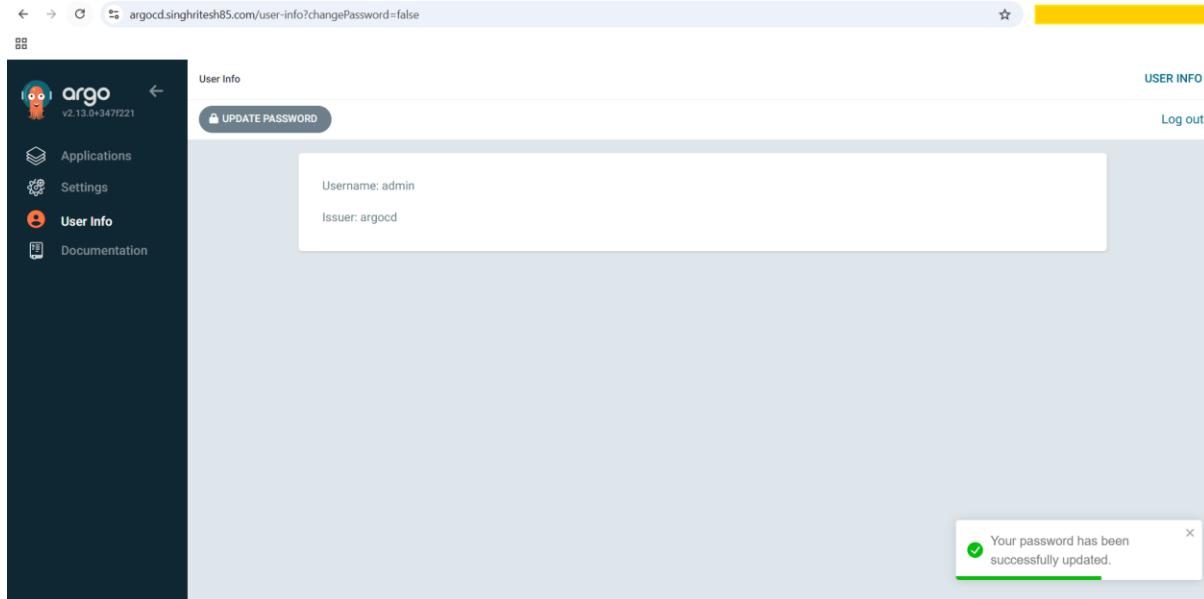
### Get password for ArgoCD

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

```
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.



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@ip-172-31-31-15 ~]# 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@... ~]# 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.

```
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.

```
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.

```
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:

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

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
```

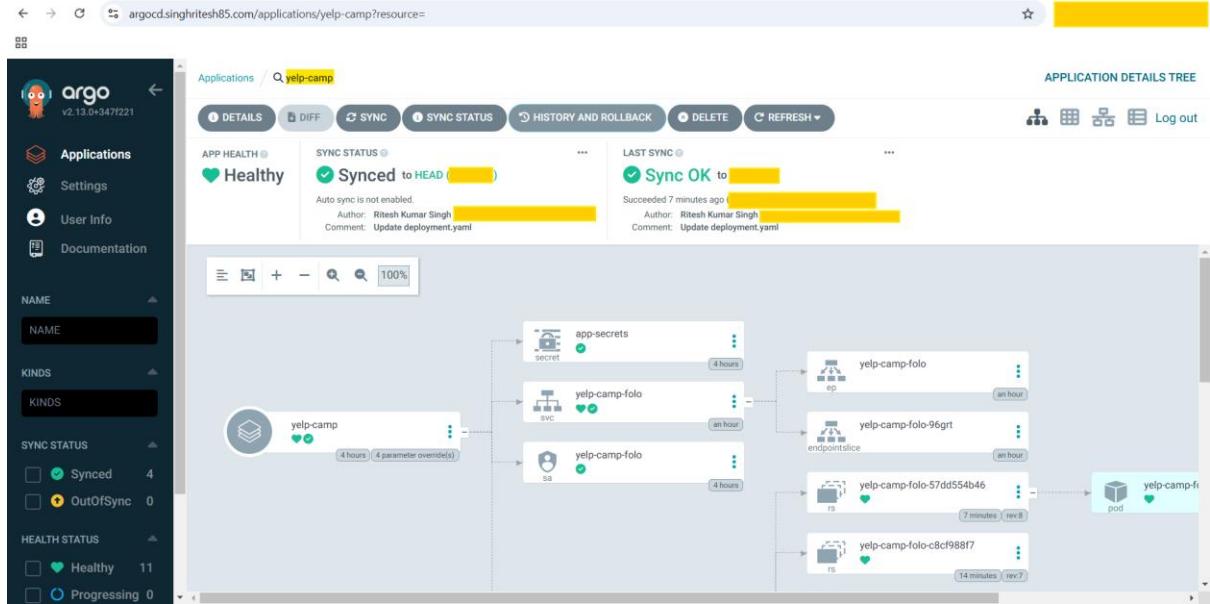
```
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
server: https://4XXXXXXXXXXXXXXXXXXXXX6.gr7.us-east-2.eks.amazonaws.com
name: arn:aws:eks:us-east-2:02XXXXXXXX6:cluster/eks-demo-cluster-dev
contexts:
- context:
  cluster: 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:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

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

The screenshot displays the SonarQube web interface. At the top, there is a navigation bar with tabs for 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. A search bar is located at the top right. Below the navigation bar, a table lists projects with columns for 'S' (Status), 'W' (Weight), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. One project, 'test', is listed with a green checkmark icon and a yellow warning icon. The main content area shows the analysis for the 'YelpCamp' project. It includes a summary card with metrics: Bugs (0 A), Vulnerabilities (0 A), Hotspots Reviewed (0.0% E), Code Smells (1 A), Coverage (0.0% O), Duplications (0.0% G), and Lines (9.9k \$ JavaScript). On the left, there are filters for Quality Gate (Passed 1, Failed 0), Reliability (A rating 1, B rating 0, C rating 0, D rating 0, E rating 0), Security (A rating 1, B rating 0, C rating 0, D rating 0, E rating 0), and Security Review (Security Hotspots 0). The bottom of the page includes footer links for SonarQube technology information and community documentation.

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

The screenshot shows the ArgoCD web interface. The left sidebar contains navigation links for 'Applications', 'Settings', 'User Info', 'Documentation', and filtering options for 'Favorites Only', 'Sync Status' (Unknown 0, Synced 1, OutOfSync 0), and 'Health Status' (Unknown 0, Progressing 0, Suspended 0, Healthy 1, Degraded 0). The main content area is titled 'APPLICATIONS SUMMARY' and features two large circular dashboards: 'Sync' (Synced 1) and 'Health' (Healthy 1). Below these are sections for 'APPLICATIONS' (1), 'SYNCED' (1), 'HEALTHY' (1), 'CLUSTERS' (1), and 'NAMESPACES' (1). A legend on the right defines colors for application status: Unknown (white), Synced (green), OutOfSync (yellow), Progressing (blue), Suspended (purple), Healthy (green), Degraded (red), and Missing (orange).



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
```

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

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

| Record name                  | Type  | Value/Route traffic to | TTL (s) |
|------------------------------|-------|------------------------|---------|
| singhritesh85.com            | NS    |                        | 172800  |
| singhritesh85.com            | SOA   |                        | 900     |
| [REDACTED]..                 | CNAME |                        | 300     |
| argocd.singhritesh85.com     | A     |                        | -       |
| jenkins-ms.singhritesh85.com | A     |                        | -       |
| sonarqube.singhritesh85.com  | A     |                        | -       |
| yelcamp.singhritesh85.com    | A     |                        | -       |

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

```
[jenkins@yellow ~]$ kubectl get all -n yelp-camp
NAME                                         READY   STATUS    RESTARTS   AGE
pod/yelp-camp-folo-5[REDACTED]6-1[REDACTED]d  1/1     Running   0          17m
NAME                                         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/yelp-camp-folo   ClusterIP   172.[REDACTED].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[REDACTED]6  1       1         1         17m
```

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

The image displays two screenshots of the YelpCamp application. The top screenshot shows the home page with a background image of a person camping at sunset. The page includes a 'View Campgrounds' button and a '© 2020' copyright notice. The bottom screenshot shows a map of South Asia with a blue marker indicating a specific location. The map is labeled 'All Campgrounds' and 'Mera Bharat'. A large orange 'RIV' watermark is overlaid on the bottom left of the map.

YelpCamp

Home Campgrounds Logout

Welcome to YelpCamp!

Jump right in and explore our many campgrounds.

Feel free to share some of your own and comment on others!

View Campgrounds

© 2020

YelpCamp Home Campgrounds New Campground Logout

All Campgrounds

Mera Bharat

RIV

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

The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'Clusters' selected, showing 'dexter' as the current cluster. Under 'DATABASES', there are two databases: 'sample\_mflix' and 'test'. The 'test' database is expanded, showing three collections: 'campgrounds', 'reviews', and 'sessions'. The 'campgrounds' collection is currently selected. At the top, it says 'VERSION 7.0.15 REGION [REDACTED]'. Below the collection name, it shows 'STORAGE SIZE: [REDACTED] LOGICAL DATA SIZE: [REDACTED] TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: [REDACTED]'. There are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. A search bar at the bottom says 'Type a query: { field: 'value' }'. The results section shows 'QUERY RESULTS: 1-2 OF 2' with one document listed:

```
_id: ObjectId('[REDACTED]')
geometry: Object
reviews: Array (empty)
title: "Mera Bharat"
location: "Delhi"
price: 11
```

## 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.

The screenshot shows the NewRelic interface. On the left, there's a sidebar with various monitoring categories like 'All Entities', 'APM & Services', 'Query Your Data', etc. The 'Kubernetes' category is selected and highlighted in blue. The main area has a heading 'Get set up and start seeing Kubernetes data in minutes' and a sub-instruction 'Visualize your cluster, from the control plane to nodes and pods. Check the health of each entity, explore logs, and see how your apps are performing.' Below this is a 'Deploy the integration' button and a 'See our docs' link. To the right, there's a preview window showing a dashboard with several charts and metrics related to the Kubernetes cluster.

**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

- 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

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.

**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

- 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.

Namespace for the integration (default: newrelic)

2 Configure the Kubernetes operation mode

Are you using a GKE Autopilot cluster ?

**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

- 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

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

Gather Log data  
Collect logs for all containers running in the cluster.  
 Forward all logs with full enrichment  
 Forward all logs with minimal enrichment (low data mode)  
Low data mode excludes labels and annotations from log records but retains a minimal set of Kubernetes metadata including cluster\_name, container\_name, namespace\_name and pod\_name.  
More detail about low data mode can be found in our docs [docs](#)

Continue

Install the Kubernetes integration  
Run this command on your host to install Kubernetes integration.

```
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=[REDACTED] --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  
logging.lowDataMode=true
```

Use a proxy

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@ip-172-31-11-46 ~]# 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=[REDACTED] --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 Helm-ing!  
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.

| NAME  | READY | STATUS  | RESTARTS | AGE        |
|---|-------|---------|----------|------------|
| newrelic-bundle-kube-state-metrics-[REDACTED]     | 1/1   | Running | 0        | [REDACTED] |
| newrelic-bundle-newrelic-logging-[REDACTED]       | 1/1   | Running | 0        | [REDACTED] |
| newrelic-bundle-newrelic-logging-[REDACTED]       | 1/1   | Running | 0        | [REDACTED] |
| newrelic-bundle-nri-kube-events-[REDACTED]        | 2/2   | Running | 0        | [REDACTED] |
| newrelic-bundle-nri-metadata-injection-[REDACTED] | 1/1   | Running | 0        | [REDACTED] |
| newrelic-bundle-nrk8s-ksm-[REDACTED]              | 2/2   | Running | 0        | [REDACTED] |
| newrelic-bundle-nrk8s-kubelet-[REDACTED]          | 2/2   | Running | 0        | [REDACTED] |
| newrelic-bundle-nrk8s-kubelet-[REDACTED]          | 2/2   | Running | 0        | [REDACTED] |

**new relic**

## Integrations & Agents

Installing on account: [Yellow Bar] 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.

**Test the connection**

| Connection type        | Status     | Details                  |
|------------------------|------------|--------------------------|
| Kubernetes integration | Successful | (Successfully installed) |

**Test connection** **See your data**

**new relic**

## Kubernetes

Save view

**Entities** **Dashboards**

| Name                 | CPU usage (%) | Memory usage (%) |
|----------------------|---------------|------------------|
| eks-demo-cluster-dev | 7.91%         | 30.73%           |

**+ Create a workload** **+ Add data**

**List** **Navigator** **Lookout**

**Kubernetes / Clusters**

**eks-demo-cluster-dev** **☆** **Tags** **Metadata** **Workloads**

**Infrastructure** **Good**

**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**

**QUICK GUIDE**

Click a node or pod on the visual to explore more details. For more filter options, click the entity types below.

| Entity Type | Icon   |
|-------------|--------|
| NODE        | [Icon] |
| POD         | [Icon] |
| Critical    | [Icon] |
| Warning     | [Icon] |
| CPU         | [Icon] |
| Memory      | [Icon] |
| Storage     | [Icon] |

**Manage Alerts**

**Since 30 minutes ago (UTC)**

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

The screenshots show the New Relic interface for installing agents on a Linux system. The top screenshot is titled 'Enter your user key' and shows a user key being entered. The bottom screenshot is titled 'Install agent' and shows a command to run on a Linux host to enable infrastructure metrics. Both screenshots include a 'Continue' button at the bottom.

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

```
[root@redacted ~]# curl -Ls https://download.newrelic.com/install/newrelic-cli/scripts/install.sh | bash && sudo NEW_RELIC_API_KEY=redacted NEW_RELIC_ACCOUNT_ID=3628931 /usr/local/bin/newrelic install
```

**Integrations & Agents**

Linux

Data source

With one command, you'll get infrastructure and log data flowing in so you can start observing your system. We'll also discover and recommend integrations for greater visibility into your stack.

Enter your credentials  
Install agent  
Test the connection

Test the connection

We'll test your connection once the installation has completed.

| Agent                  | Status    | Details  |
|------------------------|-----------|--|
| Logs Integration       | Installed | Agent installed successfully.  |
| Infrastructure Agent   | Installed | Agent installed successfully.  |
| Golden Signal Alerts   | Installed | Agent installed successfully.  |
| Integrate your AWS ... | Detected  | We found an uninstrumented data source. <a href="#">Install</a> <a href="#">Integrate your AWS account</a> |

See your data

## Hosts

|   | Name ↑   | Agent ver... | CPU usa... | Memory ... | Storage u... | Network ... | Network r... |     |
|---|--|--------------|------------|------------|--------------|-------------|--------------|-----|
| ☆ | ip- <span style="background-color: red; color: white;">REDACTED</span> .us-east-2.compute.inter... | 1.57.2       | 8.74%      | 27.71%     | 68.73%       | 873 B/s     | 10 kB/s      | ... |
| ☆ | ip- <span style="background-color: red; color: white;">REDACTED</span> .us-east-2.compute.inter... | 1.57.2       | 7.67%      | 27.51%     | 58.08%       | 822 B/s     | 9.05 kB/s    | ... |
| ☆ | ip- <span style="background-color: red; color: white;">REDACTED</span> .us-east-2.compute.internal | 1.57.2       | 2.21%      | 34.88%     | 17.83%       | 1.41 kB/s   | 2.66 kB/s    | ... |
| ☆ | ip- <span style="background-color: red; color: white;">REDACTED</span> .us-east-2.compute.inter... | 1.57.2       | 2.03%      | 18.93%     | 35.18%       | 699 B/s     | 259 B/s      | ... |
| ☆ | ip- <span style="background-color: red; color: white;">REDACTED</span> .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.

Synthetic Monitoring / Synthetic monitors

dexter • Good

Summary

MONITOR

100% Success rate last 30 min

0/105 Checks failed last 30 min

0/20 Locations failing now

Everything's A-OK! No failures detected within the last 30 min

Duration and availability of checks in the last 30 min

- Manama, BH: 0 failures / 7 checks, 895 ms max, 882 ms median, Success!
- Columbus, OH, USA: 0 failures / 7 checks, 106 ms max, 92.9 ms median, Success!
- Mumbai, IN: 0 failures / 6 checks, 1.06 s max, 970 ms median, Success!

Activity stream

Monitor created dexter 12:40pm

Created monitor 'dexter'

Related entities

Is this supposed to work alone? Most things don't operate in isolation. Add more entities and we'll detect connections. Add entities

User Experience

See full map Add to dashboard

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

**Integrations & Agents**

← Integrations & Agents / Node.js

Installing on account: [REDACTED] [Leave Feedback](#)

**Data source**  
**Node.js**

Pinpoint and solve issues down to the line of code with Node.js monitoring from New Relic. With features like service maps and error analytics, our Node.js agent helps you get the full picture of your app environment.

Select instrumentation method  
 Enter your credentials  
 Instrument your container  
 Test the connection

Follow these steps to start monitoring your app. Use environment variables to make changes to your agent configuration.

- 1 Add 'newrelic' as a dependency to your package.json file**

```
package.json 1 line 20.0 B
1 "newrelic": "latest"
```

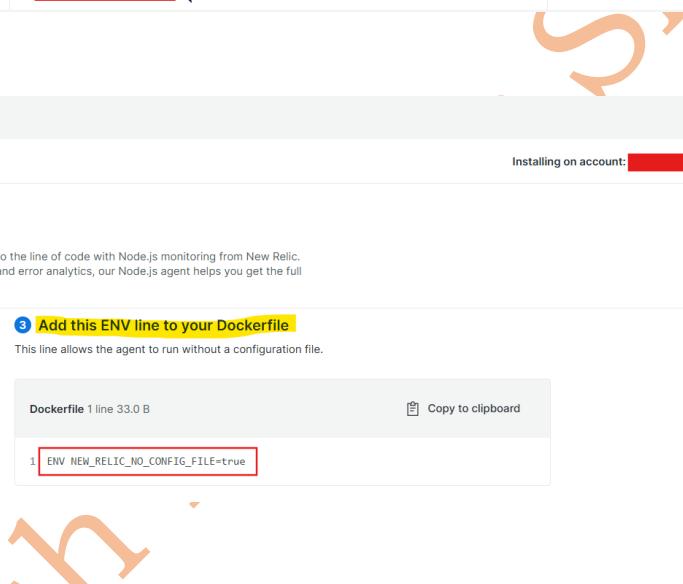
[Copy to clipboard](#)

**2 In the first line of your app's main module, add:**

```
1 line 20.0 B
1 require('newrelic');
```

[Copy to clipboard](#)

**Do this entry in first line of app.js file**



**Integrations & Agents**

← Integrations & Agents / Node.js

Installing on account: [REDACTED] [Leave Feedback](#)

**Data source**  
**Node.js**

Pinpoint and solve issues down to the line of code with Node.js monitoring from New Relic. With features like service maps and error analytics, our Node.js agent helps you get the full picture of your app environment.

Select instrumentation method  
 Enter your credentials  
 Instrument your container  
 Test the connection

- 3 Add this ENV line to your Dockerfile**

This line allows the agent to run without a configuration file.

```
Dockerfile 1 line 33.0 B
1 ENV NEW_RELIC_NO_CONFIG_FILE=true
```

[Copy to clipboard](#)



**Integrations & Agents**

← Integrations & Agents / Node.js

Installing on account: [REDACTED] [Leave Feedback](#)

**Data source**  
**Node.js**

Pinpoint and solve issues down to the line of code with Node.js monitoring from New Relic. With features like service maps and error analytics, our Node.js agent helps you get the full picture of your app environment.

Select instrumentation method  
 Enter your credentials  
 Instrument your container  
 Test the connection

- 4 Set the config options via ENV directives**

In addition to setting distributed tracing and logging, you can set other agent configuration options [here](#).

Analyze your AI application's responses  
 Turn this on to enable AI monitoring for your AI application.

**Protect your license key**  
 Do not include your license key in your Dockerfile or Docker image. For more information, see our documentation on license key security.  
[See our docs](#)

```
Dockerfile 2 lines 71.0 B
1 ENV NEW_RELIC_DISTRIBUTED_TRACING_ENABLED=true
2 ENV NEW_RELIC_LOG=stdout
```

[Copy to clipboard](#)

**Do this entry in your Dockerfile**



Do the below line entry in your Dockerfile.

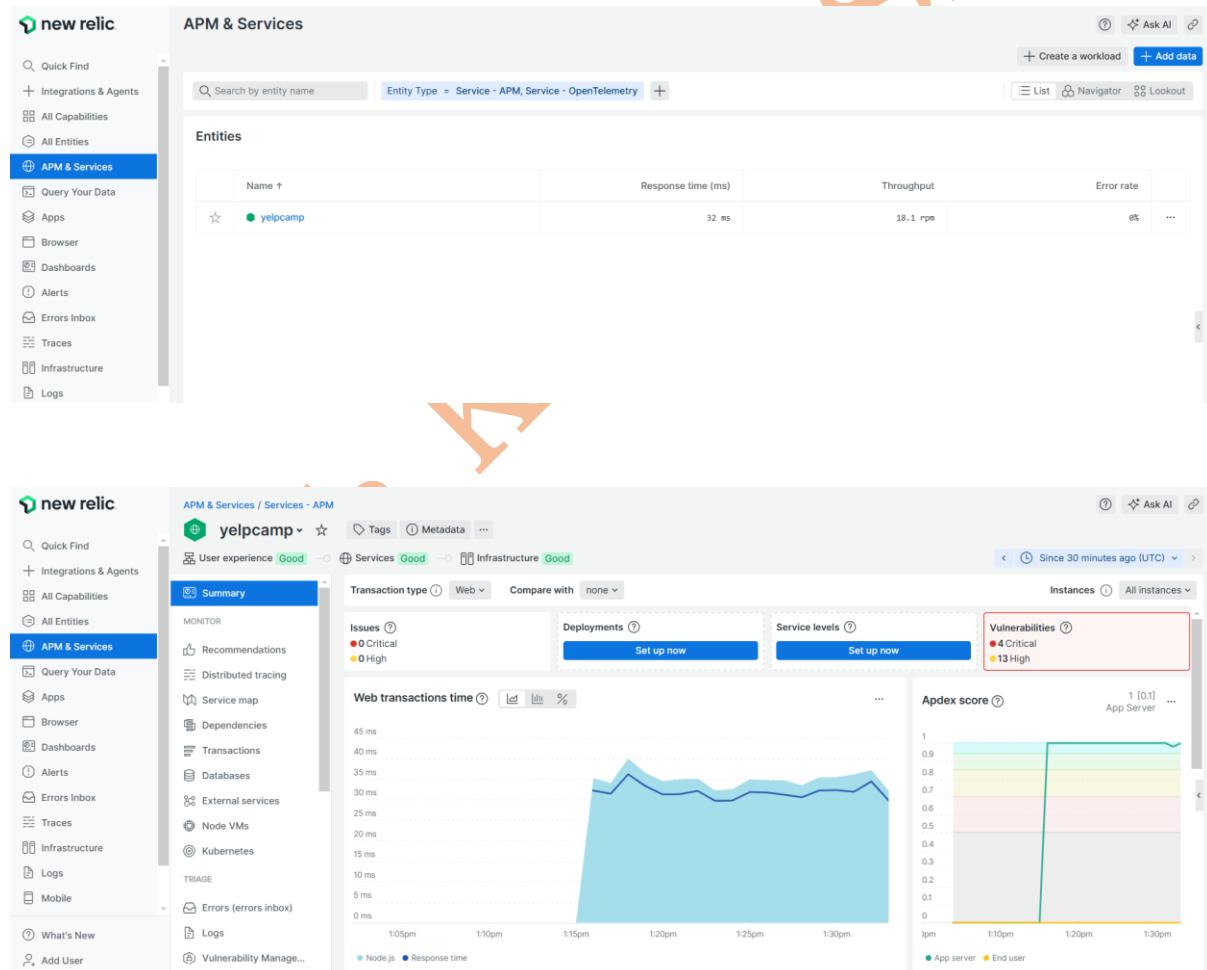
```
ENV NEW_RELIC_LICENSE_KEY=cXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
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=cXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
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.

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

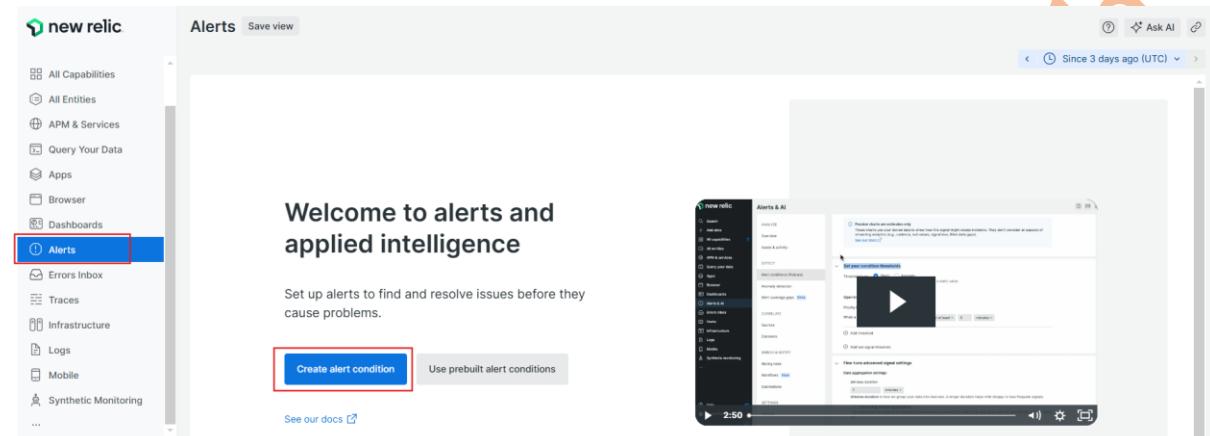
Total Request = Satisfied Request + Tolerated Request + Frustrated Request

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.

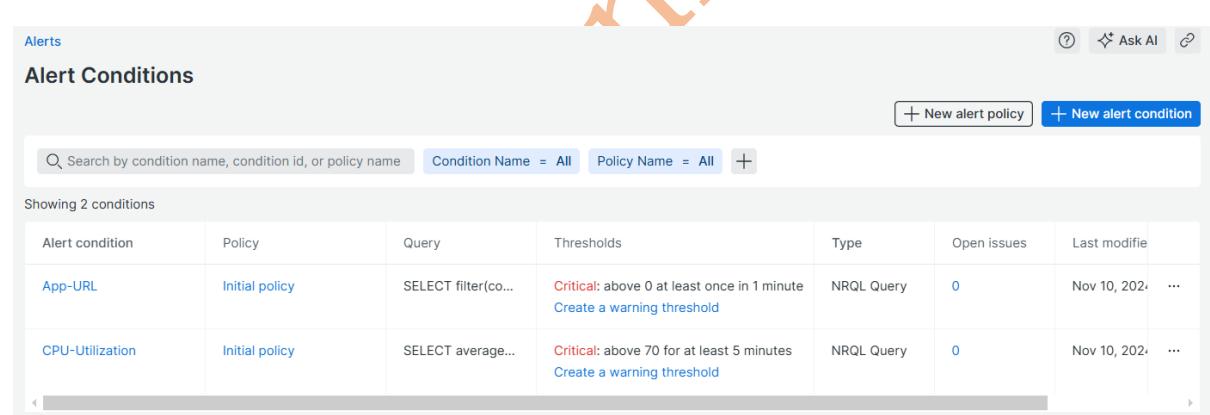
## Alert in NewRelic

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.



The screenshot shows the New Relic interface with the 'Alerts' menu item highlighted. The main area displays a 'Welcome to alerts and applied intelligence' message with a 'Create alert condition' button. A video player window is overlaid on the right side of the screen, showing a tutorial about alert conditions. The video player has a play button and a progress bar at 2:50.

The screenshot shows the 'Alert Conditions' page. At the top, there are buttons for '+ New alert policy' and '+ New alert condition'. Below is a search bar and filters for 'Condition Name' and 'Policy Name'. A table lists two existing alert conditions:

| Alert condition | Policy         | Query               | Thresholds  | Type       | Open issues | Last modified |
|-----------------|----------------|---------------------|---|------------|-------------|---------------|
| App-URL         | Initial policy | SELECT filter(co... | Critical: above 0 at least once in 1 minute<br>Create a warning threshold | NRQL Query | 0           | Nov 10, 2024  |
| CPU-Utilization | Initial policy | SELECT average...   | Critical: above 70 for at least 5 minutes<br>Create a warning threshold   | NRQL Query | 0           | Nov 10, 2024  |

The screenshot shows the AWS CloudWatch Metrics Insights interface for creating a new alert policy. The top navigation bar includes 'Alerts / Alert Policies', 'Initial policy', 'Metadata', and 'Workloads'. Below the navigation, there's an ID field with a redacted value, and tabs for 'Alert conditions', 'Notifications', and 'Settings' (which is selected). The 'Policy name' field contains 'Initial policy'. Under 'Correlation', there's a checkbox for 'Correlate and suppress noise' which is unchecked. A note says 'We'll analyze and group related incidents into issues. See our docs' with a link icon. In the 'Issue creation preference' section, three options are listed: 'One issue per policy' (selected, highlighted with a red border), 'One issue per condition', and 'One issue per condition and signal'. Each option has a descriptive note below it. At the bottom left is a 'Save' button with a checkmark icon. A large orange arrow points upwards from the bottom of the page towards this save button.

This is a modal dialog for configuring an email notification. It has a title 'Email' with an envelope icon. The 'Email destination' field contains 'Email' with a clear ('X') button. The 'Email subject' field contains '{{ issueTitle }}'. Under 'Custom Details (optional)', there's a note about Handlebars syntax and a text input field. At the bottom, a message says 'Test notification sent successfully.' with a green checkmark icon, and a 'Send test notification' button. A red box highlights the 'Send test notification' button. At the very bottom of the dialog are 'Cancel' and 'Save' buttons. A large orange arrow points upwards from the bottom of the page towards the 'Save' button on the main page.

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.

Synthetic Monitoring

**Monitors**

Search by entity name Entity Type = Synthetic monitor +

| Name ↑ | Monitor status | Success rate ... | Locations fail... | Period | Monitor type |
|--------|----------------|------------------|-------------------|--------|--------------|
| dexter | Enabled        | 99.4%            | 20 / 20           | 60 s   | Ping         |

Lookout view

dexter query result is > 0.0 on 'App-URL'

New Relic Incident Intelligence <noreply@notifications.newrelic.com>

Critical priority issue is active

dexter query result is > 0.0 on 'App-URL'

Acknowledge Close issue Go to issue

1 incidents

- dexter query result is > 0.0 on 'App-URL'

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.

Synthetic Monitoring / Synthetic monitors

dexter • Good

Summary

MONITOR

RESULTS

RESOURCES

TRIAGE

DIGNOSTIC

REPORTS

SLA

SERVICE LEVELS

SETTINGS

GENERAL

ALERT CONDITIONS

MORE VIEWS

Add app

Change tracking

Events explorer

Help

52.5% Success rate last 30 min

303/600 Checks failed last 30 min

0/20 Locations failing now

Network Error DNS resolution failed for ho... Last error detected 3 minutes ago

Duration and availability of checks in the last 30 min

Hong Kong, HK 22 failures / 30 checks ... November 10th 8:49 PM November 10th 9:17 PM Run check

Sydney, AU 19 failures / 30 checks ... November 10th 8:49 PM November 10th 9:17 PM Run check

Tokyo, JP 14 failures / 30 checks ... November 10th 8:49 PM November 10th 9:17 PM Run check

Activity stream

Since 30 minutes ago (UTC) Manage alerts

Critical Issue Closed dexter 3:45pm dexter query result is > 0.0 on 'App-URL'

Critical Issue Activated dexter 3:25pm dexter query result is > 0.0 on 'App-URL'

Related entities

You have gaps in your alert coverage Some parts of your stack don't have alerts watching them. Set up alerts

User Experience

Services 1

## Monitor Jenkins with NewRelic

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

The screenshot shows the Jenkins interface with the 'Manage Jenkins' and 'Plugins' sections selected. A search bar at the top contains 'OpenTelemetry'. Below it, a table lists the 'OpenTelemetry' plugin, which has been successfully installed. The table includes columns for 'Install', 'Name', and 'Released'. The 'Released' column shows '5 days 15 hr ago'. A large orange watermark reading 'Simplifying' is diagonally across the page.

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

The screenshot shows the New Relic interface with the 'Integrations & Agents' section selected. A search bar at the top contains 'jenkins'. Below it, a list shows the 'Jenkins' agent, which is highlighted with a green dot. Another screenshot below shows the 'Installation plan' for Jenkins, where the 'Jenkins' account is selected. The 'Begin installation' button is visible. A large orange watermark reading 'Simplifying' is diagonally across the page.

The screenshot shows the New Relic interface for integrating with Jenkins. On the left sidebar, under 'Integrations & Agents', 'Jenkins' is selected. The main content area is titled 'Installations' and shows a 'Data source' section for Jenkins. It lists three steps: 'Select an account' (done), 'Configure Jenkins OpenTelemetry plugin' (in progress), and 'Validate Jenkins Integration' (not started). The current step, 'Configure Jenkins OpenTelemetry plugin', has two sub-steps: 'Enter an OTLP Endpoint' (with a red box around the URL field containing 'https://otlp.nr-data.net:4317') and 'Enter your credentials'. A 'Copy your license key' button is located at the bottom of this section.

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.

The screenshot shows the 'Installation plan' page for Jenkins. It includes a 'Select an account' step (done), an 'Already instrumented Jenkins?' section (with 'Skip this step' and 'Begin installation' buttons), and a 'Deploy quickstart' step (not started). On the right side, there's a 'Looking for your license key?' section with a 'Copy license key' button highlighted by a red box. Below it, a note says: 'If you're using our guided install, you won't need this. We'll handle the details for you.'

Scope ?  
Global (Jenkins, nodes, items, all child items, etc)

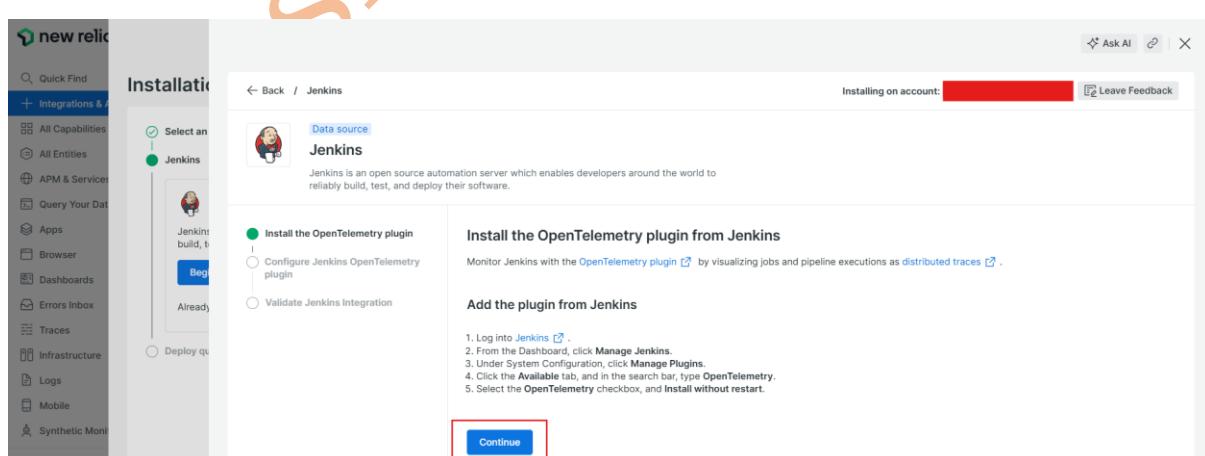
Secret  
[REDACTED]

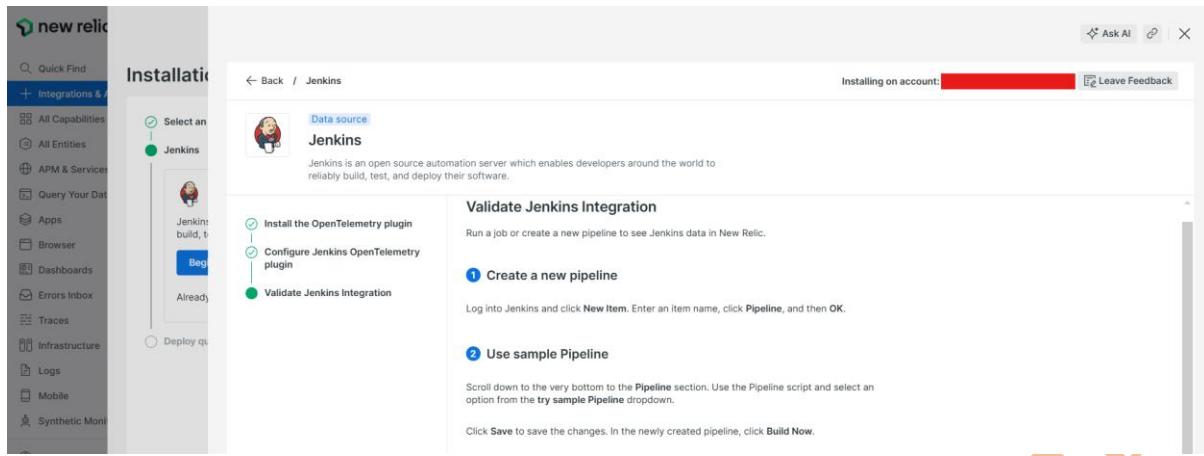
ID ?  
api-key

Description ?  
api-key

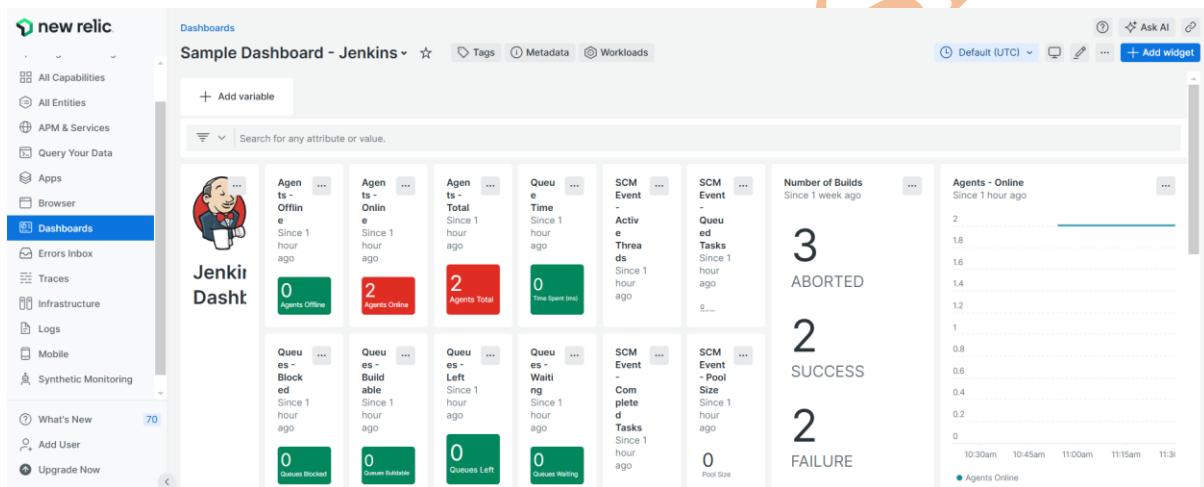
**Save**

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





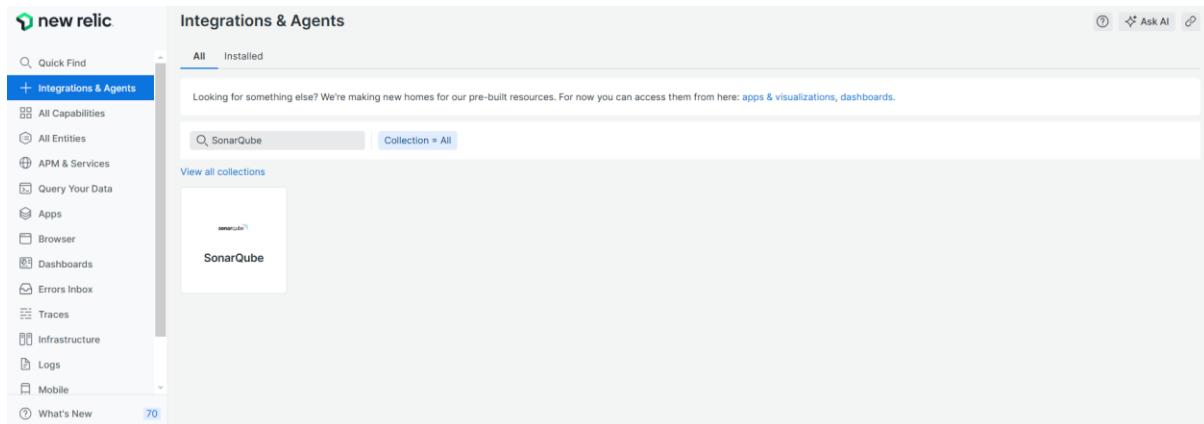
For the demonstration purpose I ran the Jenkins Job 7 times after providing the required parameters to run it and below is the screenshot of the NewRelic Jenkins Dashboard for Monitoring of Jenkins.



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.



**Installation plan**

Select an account: SonarQube

SonarQube

Identify and optimize SonarQube performance with New Relic SonarQube monitoring.

**Begin installation**

Already instrumented SonarQube? Skip this step

Deploy quickstart

Ingest credentials Support

Looking for your license key?

If your quickstart installation requires a license key, copy it from here:

Generate and copy license key

If you're using our guided install, you won't need this. We'll handle the details for you.

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

New Relic

Installation / SonarQube

Select an account: SonarQube

Data source: SonarQube

Identify and optimize SonarQube performance with New Relic SonarQube monitoring.

**Begin**

Already instrumented SonarQube? Skip this step

Deploy quickstart

Installing on account: [REDACTED] Leave Feedback

Enter your credentials

Install the infrastructure agent

To use the SonarQube integration, you need to first install the infrastructure agent on the same host. All our on-host integrations require the infrastructure agent, which helps expose and report metrics to New Relic.

Linux or macOS Windows

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

Copy to clipboard

Continue

To install infrastructure agent manually

If you'd rather install our infrastructure agent manually, you can follow a tutorial for manual installation.

Linux Windows

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

```
[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]:B@[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 (>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
```

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

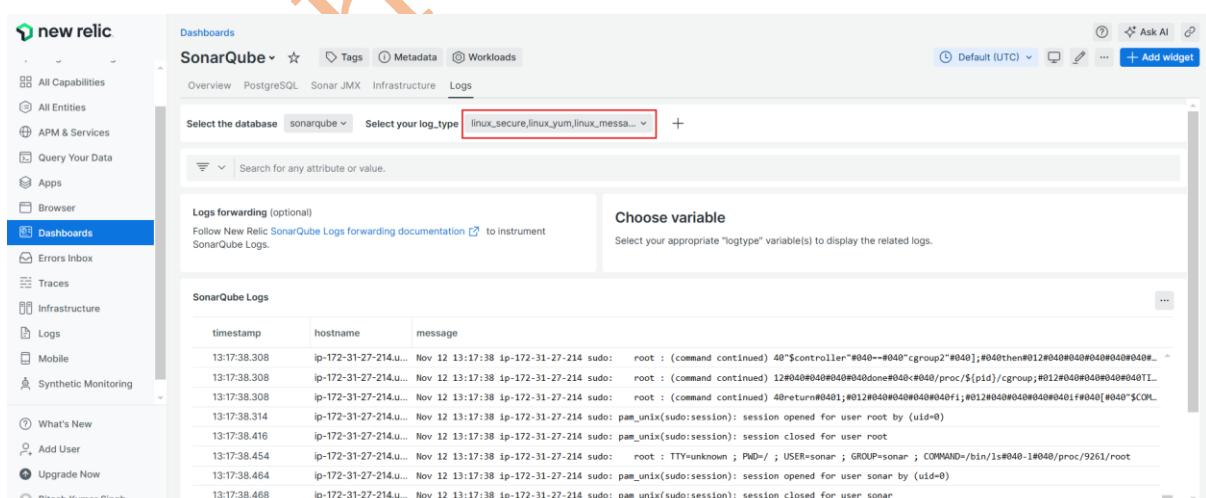
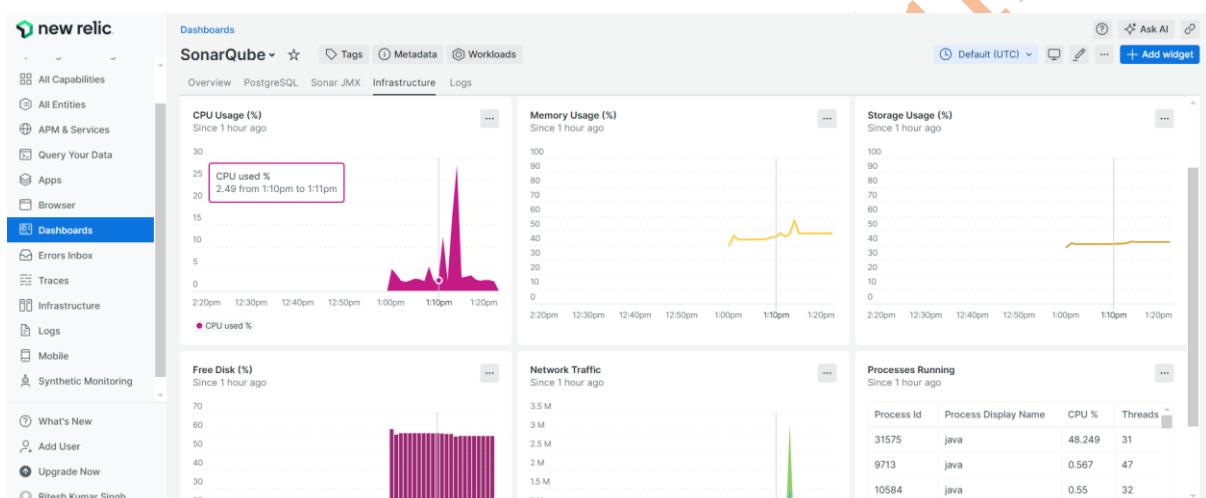
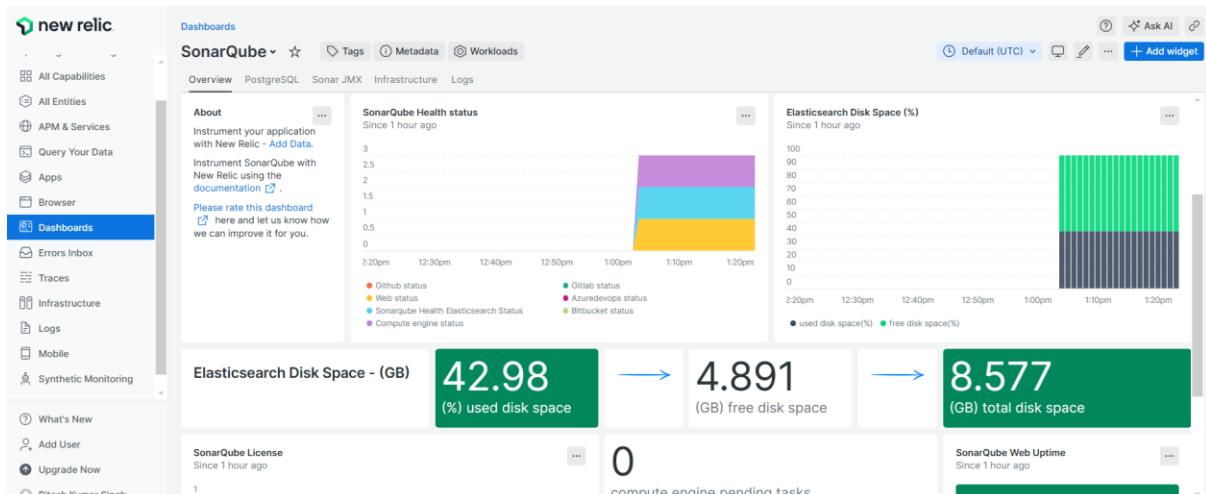
```
logs:
  - name: cloud-init.log
    file: /var/log/cloud-init.log
    attributes:
      logtype: linux_cloud-init
  - name: messages
    file: /var/log/messages
    attributes:
      logtype: linux_messages
  - name: secure
    file: /var/log/secure
    attributes:
      logtype: linux_secure
  - 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
```

```
[root@... 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
  - 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@... opt]# sudo systemctl restart newrelic-infra
[root@... 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.

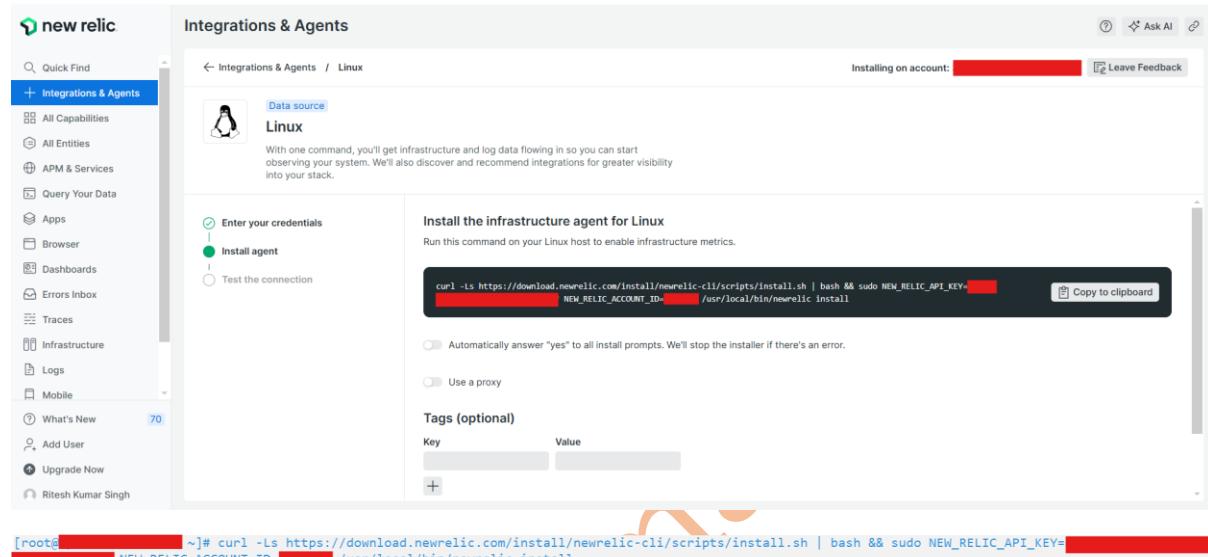


## Configure Log aggregation in Jenkins Master and Jenkins Slave

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.

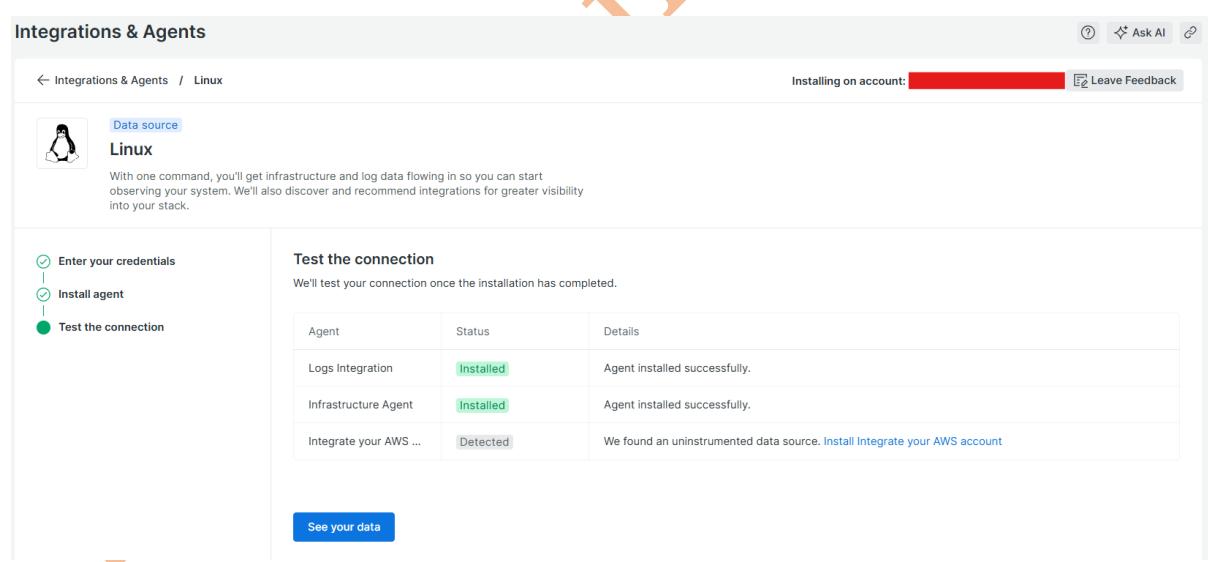


The screenshot shows the New Relic interface for integrating with a Linux host. On the left sidebar, under 'Integrations & Agents', the 'Linux' data source is selected. The main panel displays instructions to 'Install the infrastructure agent for Linux' with a command-line snippet:

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

Below the command are two optional checkboxes: 'Automatically answer "yes" to all install prompts. We'll stop the installer if there's an error.' and 'Use a proxy'. A 'Tags (optional)' section allows adding key-value pairs. A terminal window at the bottom shows the command being run on a root shell:

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

The second screenshot shows the continuation of the setup process. The 'Test the connection' step is displayed, indicating that the connection will be tested once the installation is complete. Below this, a table shows the status of various agents installed on the system:

| Agent                  | Status    | Details  |
|------------------------|-----------|--|
| Logs Integration       | Installed | Agent installed successfully.  |
| Infrastructure Agent   | Installed | Agent installed successfully.  |
| Integrate your AWS ... | Detected  | We found an uninstrumented data source. <a href="#">Install Integrate your AWS account</a> |

A blue 'See your data' button is located at the bottom of this section.

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

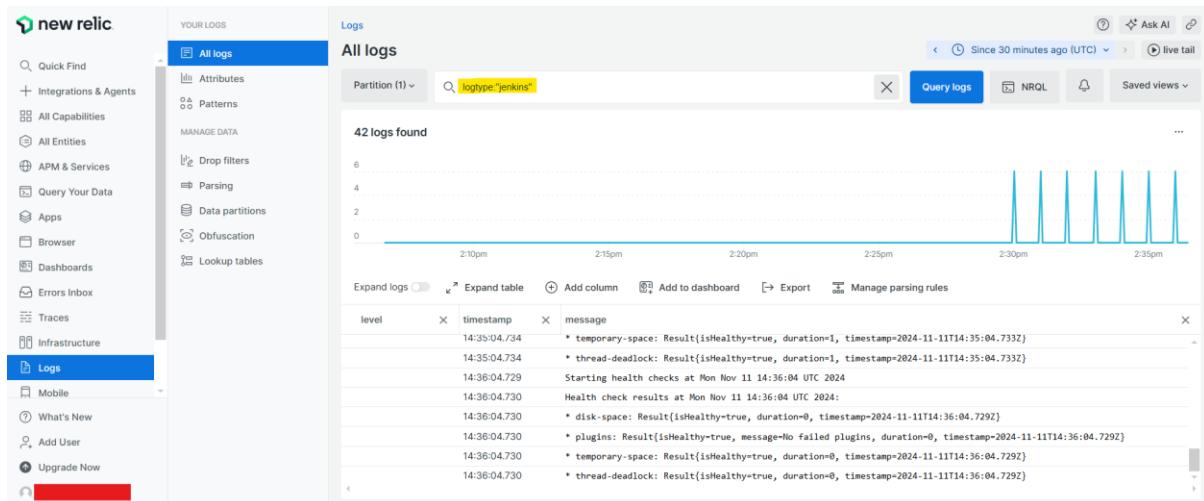
```
- name: jenkins-health-checker-log
  file: /var/lib/jenkins/logs/health-checker.log
  attributes:
    logtype: jenkins
```

```
[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.

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 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') {  
                    // SonarQube analysis steps  
                }  
            }  
        }  
    }  
}
```

```

        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}'
    }
}

```

```
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'

    }
}

}

}

}
```

**Reference:** - <https://github.com/Colt/YelpCamp.git>