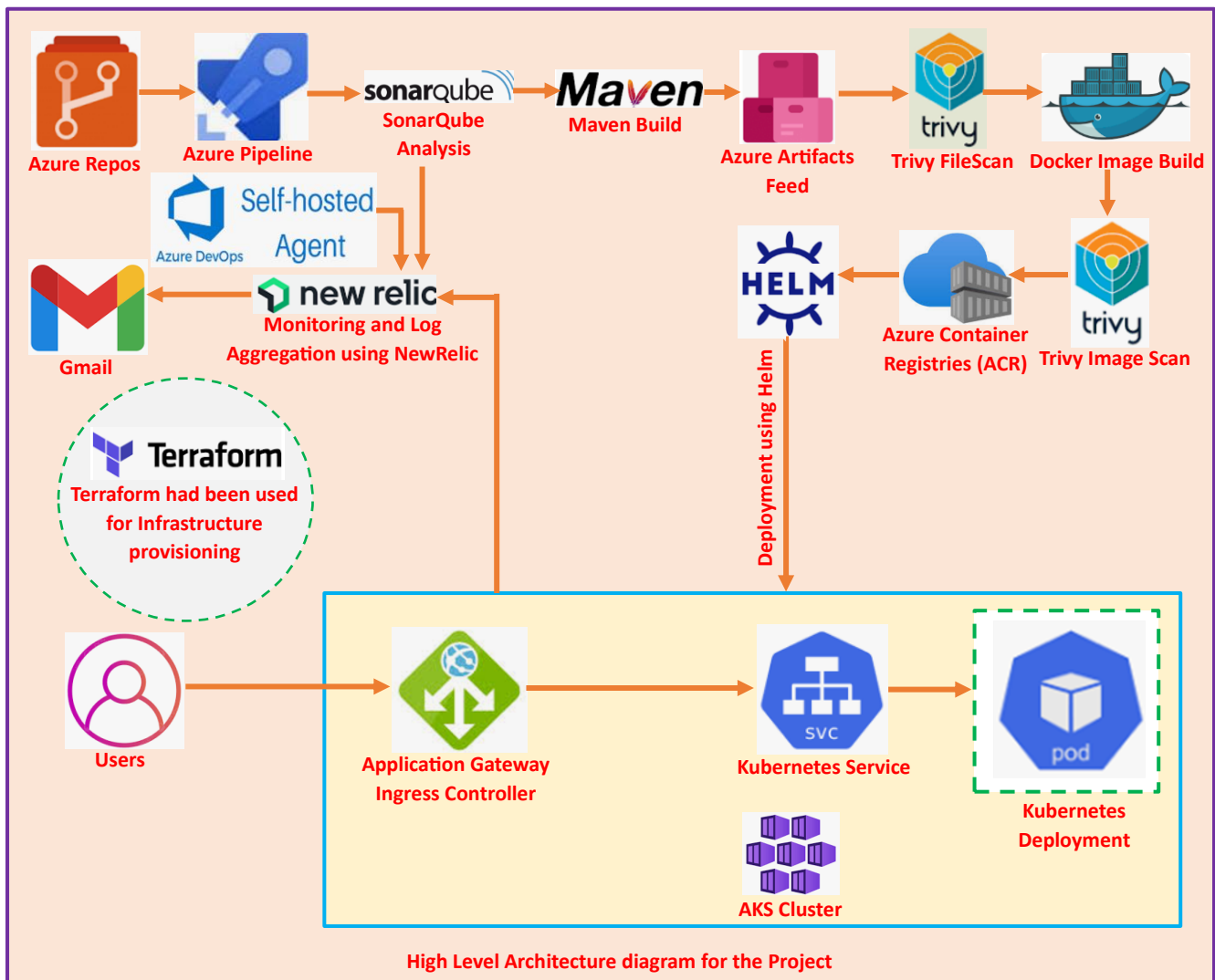


DevOps Project Blogging App Deployment Monitoring and Log Aggregation using NewRelic Azure



This DevOps Project aims to create the infrastructure using Terraform and established the CI/CD set-up using Azure DevOps. For monitoring and Log Aggregation NewRelic had been used which was also shown in the screenshot attached above. The source code was present in the Azure Repos, which was deployed through the Azure DevOps Pipeline. SonarQube had been used for Code Analysis and Maven was used as the Build Tool, artifacts for this project were kept in the Azure Artifacts Feed. Trivy was used as the File Scan and Docker Image Scan which was also shown in the Architecture diagram shown above. The created Docker Image was kept in Azure Container Registries (ACR) which was deployed to Azure Kubernetes Services (AKS Cluster) using the Helm as shown in the Architecture diagram shown above.

For Azure DevOps Pipeline I had used Self-hosted-Agent and followed the below procedure to install it.

```
[root@devopsagent-vm ~]# cd /opt && mkdir myagent && cd myagent

[root@devopsagent-vm myagent]# wget https://vstsagentpackage.azureedge.net/agent/[REDACTED]/vsts-agent-linux-x64-[REDACTED].tar.gz

[root@devopsagent-vm myagent]# tar -xvf vsts-agent-linux-x64-[REDACTED].tar.gz

[root@devopsagent-vm myagent]# rm -f vsts-agent-linux-x64-[REDACTED].tar.gz

[root@devopsagent-vm myagent]# ./bin/installdependencies.sh

[demo@devopsagent-vm myagent]$ ./config.sh

Azure Pipelines
agent v[REDACTED]

[demo@devopsagent-vm myagent]$ ./env.sh
[demo@devopsagent-vm myagent]$
[demo@devopsagent-vm myagent]$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/opt/sonar-scanner/bin:/opt/apache-maven/bin:/opt/node-v16.0.0/bin:/opt/dependency-check/bin:/usr/local/bin

[demo@devopsagent-vm myagent]$ sudo ./svc.sh install

[demo@devopsagent-vm myagent]$ sudo ./svc.sh start
```

Below screenshot shows the source code which was present in the Azure Repos.

Blogging-App

- src
- Dockerfile
- pom.xml
- README.md

Files

Contents History

Name ↑	Last change	Commits
src	12m ago	[REDACTED] demo root
Dockerfile	12m ago	[REDACTED] demo root
pom.xml	12m ago	[REDACTED] demo root
README.md	15m ago	[REDACTED] Added README.md Ritesh Kumar Singh

Introduction

TODO: Give a short introduction of your project. Let this section explain the objectives or the motivation behind this project.

Getting Started

TODO: Guide users through getting your code up and running on their own system. In this section you can talk about:

1. Installation process
2. Software dependencies
3. Latest releases
4. API references

I had created Azure Artifacts Feed named as Maven and provided contributor access for the deployment user from Azure Artifacts Feed settings which was shown in the screenshot attached below.

Maven

Connect to Feed

+ Create Feed

Search Upstream Sources

Recycle Bin

Settings



Click here to provide permission



Connect to the feed to get started

Connect to Feed

Or find packages upstream

Search Upstream Sources

[Learn more about Azure Artifacts](#)

Feed Details Permissions Views Upstream Sources

Permissions

Add users/groups

Edit

Delete

User/Group

User/Group

Role

Inherited



[Redacted] User

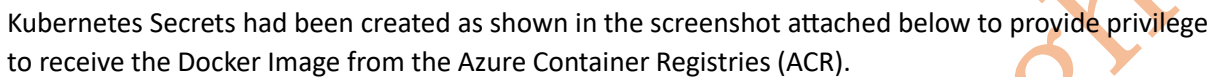
Feed Publisher (Contributor)

I had updated the pom.xml for Feed connection as shown in the screenshot attached below.

```
Bloggng-App  : main  pom.xml
> src
  Dockerfile
  pom.xml
  README.md

pom.xml
102 <repositories>
103   <repository>
104     <id>Maven</id>
105     <url>https://[Redacted]</url>
106   </repository>
107   <repository>
108     <id>Maven</id>
109     <url>https://[Redacted]</url>
110   </repository>
111 </repositories>
112 <distributionManagement>
113   <repository>
114     <id>Maven</id>
115     <url>https://[Redacted]</url>
116   </repository>
117 </distributionManagement>
118 </project>
```

Service connection for Azure DevOps had been created for **SonarQube**, **Azure Artifacts Feed** and **Docker Registry** as shown in the screenshot attached below.



Kubernetes Secrets had been created as shown in the screenshot attached below for TLS of the kubernetes ingress.

[illegible]

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

Kubernetes Secrets in the namespace `blogapp` had been listed out using the command `kubectl get secrets -n blogapp` as shown in the screenshot attached below.

```
[root@yellow01 ~]# kubectl get secrets -n blogapp
```

NAME	TYPE	DATA	AGE
bloggingapp-auth	kubernetes.io/dockerconfigjson	1	
ingress-secret	kubernetes.io/tls	2	

I had provided restricted access to the deployment user **demo** in the AKS Cluster using service account, Role and Role Binding as shown in the screenshot attached below. The deployment user **demo** had all the access in the namespace blogapp but did not have entire access over the AKS Cluster. That means the deployment user demo, access was restricted to the namespace blogapp.

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

Rite

cat sa-role-rolebinding.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: demo

namespace: blogapp

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

name: user-role

namespace: blogapp

rules:

- apiGroups: ["*"]

resources: ["*"]

verbs: ["*"]

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: user-rolebinding

namespace: blogapp

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: Role

name: user-role

subjects:

- namespace: blogapp

kind: ServiceAccount

name: demo

Finally, Kubernetes Secrets with the name of **mysecret** had been created and its token was used in the kubeconfig file as shown in the screenshot attached below.

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

```
[root@redhat ~]# kubectl describe secret mysecret -n blogapp
Name:         mysecret
Namespace:    blogapp
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: demo
              kubernetes.io/service-account.uid: 11111111-1111-1111-1111-111111111111
Type:         kubernetes.io/service-account-token

Data
====
namespace: 7 bytes
token:      [REDACTED]
ca.crt:     1765 bytes
```

```
cat secret.yaml
```

```
---
```

```
apiVersion: v1
```

```
kind: Secret
```

```
type: kubernetes.io/service-account-token
```

```
metadata:
```

```
  name: mysecret
```

```
  namespace: blogapp
```

```
  annotations:
```

```
    kubernetes.io/service-account.name: demo
```

```
kubectl describe secret mysecret -n blogapp
```

Name: mysecret

Namespace: blogapp

Labels: <none>

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

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

Type: kubernetes.io/service-account-token

Data

====

ca.crt: 1765 bytes

namespace: 7 bytes

token:

XX
XX
XX
XX
XX

Below screenshot shows the kubeconfig file which was shared with the deployment user demo.


```
cat ~/.kube/config

apiVersion: v1

clusters:
- cluster:
    certificate-authority-data:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

    server: https://blogapp-cluster-dns-
XXXXXXXXX.XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.privatelink.eastus2.azmk8s.io:443

    name: blogapp-cluster

contexts:
- context:
    cluster: blogapp-cluster

    user: demo

    name: dexter

current-context: dexter

kind: Config

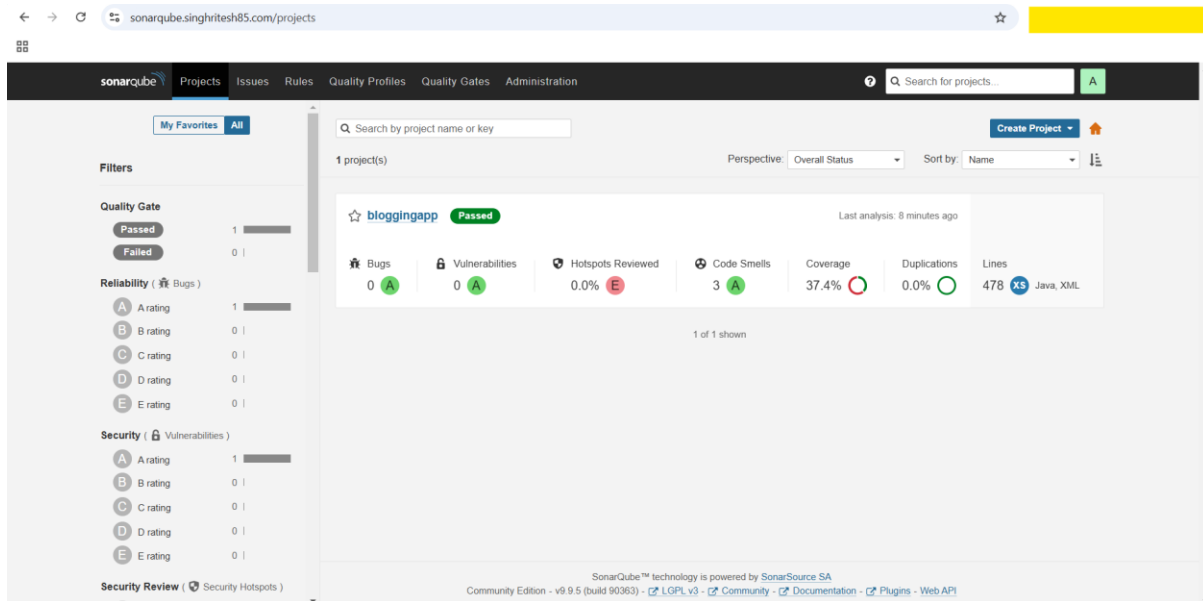
preferences: {}

users:
- name: demo

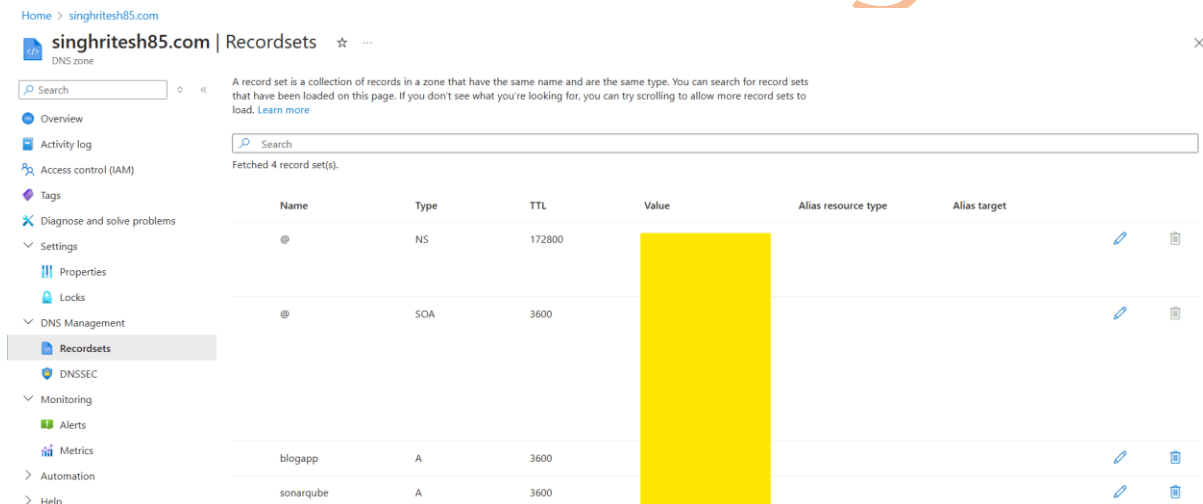
  user:

    token:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

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



Below screenshot shows the entry for Azure DNS Zone for creation of Record Set.



After running Azure DevOps Pipeline Kubernetes Pods, Kubernetes Service and Kubernetes Deployment had been created as shown in the screenshot attached below.

```
[demo@devopsagent-vm ~]$ kubectl get all -n blogapp
NAME                                READY   STATUS    RESTARTS   AGE
pod/blogapp-folo-                   1/1     Running   0           53m
pod/blogapp-folo-                   1/1     Running   0           53m

NAME                                TYPE           CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/blogapp-folo                ClusterIP      10.10.10.10   <none>        80/TCP     121m

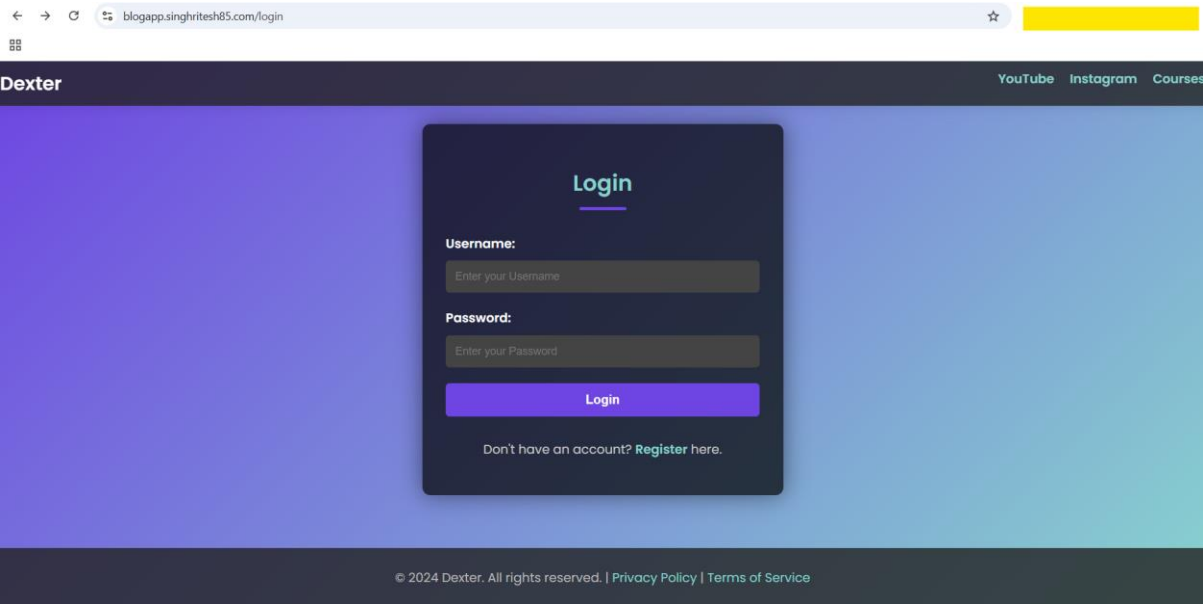
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/blogapp-folo        2/2     2             2           121m

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/blogapp-folo-       0         0         0       66m
replicaset.apps/blogapp-folo-       2         2         2       53m

[demo@devopsagent-vm ~]$ kubectl get ing -n blogapp
NAME                CLASS          HOSTS                                ADDRESS          PORTS    AGE
blogapp-ingress     azure-application-gateway  blogapp.singhritesh85.com  135.10.10.10    80, 443  5h30m

[demo@devopsagent-vm ~]$ kubectl get get nodes
error: the server doesn't have a resource type "get"
```

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




For Monitoring AKS Cluster using NewRelic I installed the NewRelic Agent 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=blogapp-cluster --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 newrelic-prometheus-agent.enabled=true --set newrelic-prometheus-agent.lowDataMode=true --set newrelic-prometheus-agent.config.kubernetes.integrations_filter.enabled=false --set k8s-agents-operator.enabled=true --set logging.enabled=true --set newrelic-logging.lowDataMode=true
"newrelic" has been added to your repositories
Hang tight while we grab the latest from your chart repositories...
...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.
NAME: newrelic-bundle
LAST DEPLOYED: Mon Nov
NAMESPACE: newrelic
STATUS: deployed
REVISION: 1
TEST SUITE: None
[root@~]# kubectl get pods -n blogapp
NAME READY STATUS RESTARTS AGE
blogapp-folo- 1/1 Running 0
blogapp-folo- 1/1 Running 0
[root@~]# kubectl get pods -n newrelic
NAME READY STATUS RESTARTS AGE
newrelic-bundle-k8s-agents-operator- 2/2 Running 0
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-newrelic-prometheus-agent-0 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
[root@~]#
```

← Integrations & Agents / Kubernetes

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



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](#)

- Select instrumentation method
- Enter your credentials
- Configure the Kubernetes integration
- Select additional data
- Enable APM auto-instrumentation
- Gather Log data
- Install the Kubernetes integration
- Configure APM auto-instrumentation
- Test the connection**

After successful installation of NewRelic Agent on AKS Cluster we saw the AKS Cluster with the name of blogapp-cluster in NewRelic Console as shown in the screenshot attached below.

▼ **Kubernetes Clusters** [View all \(1\)](#) [Dashboards](#)

Name ↑	CPU usage (%)	Memory usage (%)
● blogapp-cluster	9.94%	39.57%

Kubernetes / Clusters

blogapp-cluster ☆ [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

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) 


① 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*

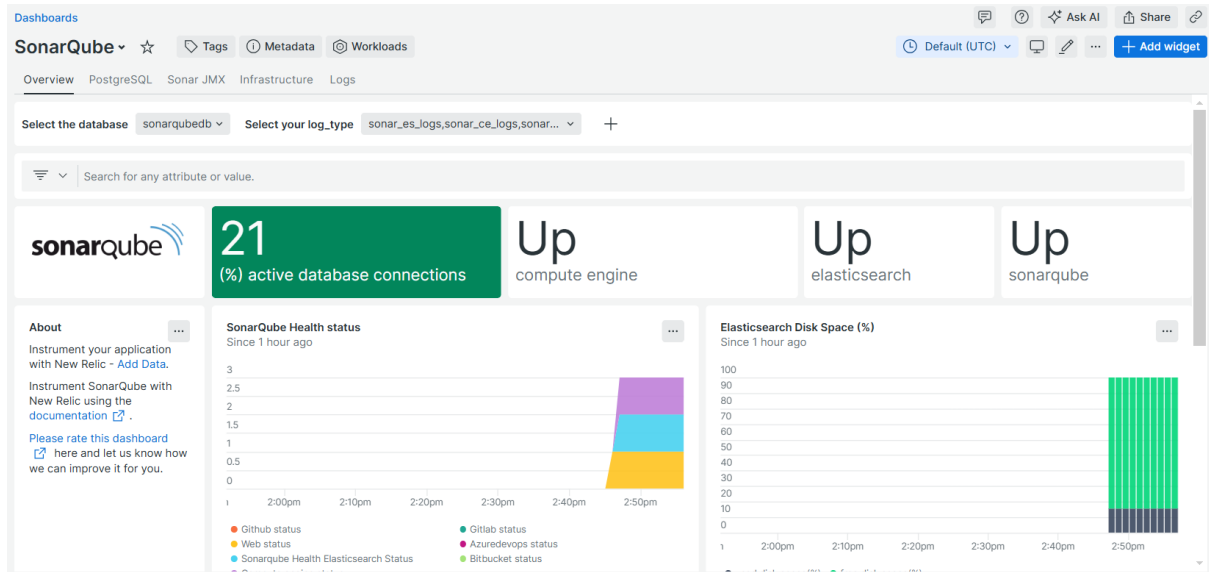
For Monitoring SonarQube using NewRelic I installed the NewRelic Agent for SonarQube on SonarQube Server as shown in the screenshot attached below.

```
[root@SonarQube-Server ~]# 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
Installing New Relic CLI v0.97.1
Installing to /usr/local/bin
```



Welcome to New Relic. Let's set up full stack observability for your environment.
Our Data Privacy Notice: <https://newrelic.com/termsandconditions/services-notices>

After successful installation of NewRelic Agent on SonarQube Server I was able to see the sonarqube dashboard for monitoring in NewRelic as shown in the screenshot attached below.



As shown in the screenshot attached below I was able to see the SonarQube logs in the NewRelic console.

The screenshot shows the 'SonarQube Logs' page. It includes a 'Logs forwarding (optional)' section with a link to 'Follow New Relic SonarQube Logs forwarding documentation'. A 'Choose variable' section allows selecting logtype variables. The main part of the page is a table of log entries.


timestamp	hostname	message
14:46:26.448	SonarQube-Server	2024.11.25 14:46:26 INFO es[[o.e.p.PluginsService] loaded module [transport-netty4]
14:46:26.448	SonarQube-Server	2024.11.25 14:46:26 INFO es[[o.e.p.PluginsService] no plugins loaded
14:46:26.480	SonarQube-Server	2024.11.25 14:46:26 INFO es[[o.e.e.NodeEnvironment] using [1] data paths, mounts [[/dev/sda2]], net usable_space [25.9gb], net total_spa...
14:46:26.490	SonarQube-Server	2024.11.25 14:46:26 INFO es[[o.e.e.NodeEnvironment] heap size [512mb], compressed ordinary object pointers [true]
14:46:26.587	SonarQube-Server	2024.11.25 14:46:26 INFO es[[o.e.n.Node] node name [sonarqube], node ID [zhboQShrRpaTagHpubHF3Q], cluster name [sonarqube], roles [data_froz...
14:46:29.499	SonarQube-Server	2024.11.25 14:46:29 INFO es[[o.e.t.NettyAllocator] creating NettyAllocator with the following configs: [name=unpooled, suggested_max_allocat...
14:46:29.538	SonarQube-Server	2024.11.25 14:46:29 INFO es[[o.e.i.r.RecoverySettings] using rate limit [40mb] with [default=40mb, read=0b, write=0b, max=0b]
14:46:29.587	SonarQube-Server	2024.11.25 14:46:29 INFO es[[o.e.d.DiscoveryModule] using discovery type [zen] and seed hosts providers [settings]
14:46:29.833	SonarQube-Server	2024.11.25 14:46:29 INFO es[[o.e.g.DanglingIndicesState] gateway.auto_import_dangling_indices is disabled, dangling indices will not be auto...
14:46:30.017	SonarQube-Server	2024.11.25 14:46:30 INFO es[[o.e.n.Node] initialized
14:46:30.017	SonarQube-Server	2024.11.25 14:46:30 INFO es[[o.e.n.Node] starting ...

Alerts in NewRelic

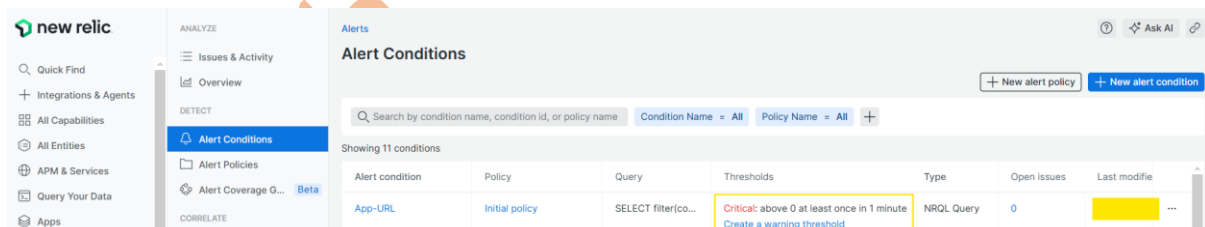
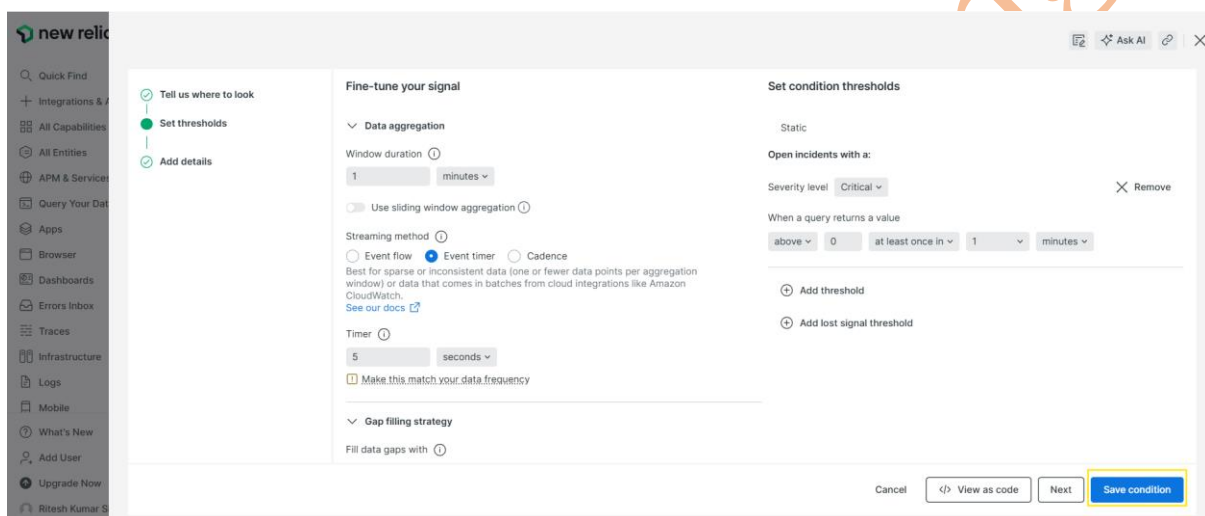
For Alerts in NewRelic we need to discuss about **Destination**, **Alert Condition**, **Alert Policy**, and **Workflows**.

I had created an Email destination with the Group Email ID of the concerned team as shown in the screenshot attached below.

Destinations (2) [See notifications log](#)

Type	Name ↑	URL/Details	Last Updated	Updated By	Used By	Enabled
	Email			Ritesh Kumar Singh	2 workflows	<input checked="" type="checkbox"/>

An alert condition had been created for Synthetic Monitoring as shown in the screenshot attached below.



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 Create a warning threshold	NRQL Query	0	

An Alert condition for CPU Utilization had been created as shown in the screenshot attached below.

new relic

Quick Find
Integrations & ...
All Capabilities
All Entities
APM & Services
Query Your Data
Apps
Browser
Dashboards
Errors Inbox
Traces
Infrastructure
Logs
Mobile
What's New
Add User
Upgrade Now
Ritesh Kumar S

Tell us where to look
Set thresholds
Add details

Fine-tune your signal

▼ Data aggregation

Window duration ⓘ
1 minutes ▼

☐ Use sliding window aggregation ⓘ

Streaming method ⓘ
☒ Event flow ☐ Event timer ☐ Cadence
 Best for steady or frequently reporting data (at least one data point per aggregation window).
[See our docs](#)

Delay ⓘ
2 minutes ▼

▼ Gap filling strategy

Fill data gaps with ⓘ
None ▼

Set condition thresholds

Static

Open incidents with a:

Severity level Critical ▼ ✕ Remove

When a query returns a value
 above ▼ 70 for at least ▼ 5 minutes ▼

Cancel

CPU-Utilization	Initial policy	SELECT average...	Critical: above 70 for at least 5 minutes Create a warning threshold	NRQL Query	0		...
-----------------	----------------	-------------------	---	------------	---	--	-----

An Alert Policy had been created as shown in the screenshot attached below.

Alerts / Alert Policies

Initial policy ☆ ⓘ Metadata ⚙ Workloads

Alert conditions Notifications Settings

Policy name
Initial policy

Correlation
☐ Correlate and suppress noise
 We'll analyze and group related incidents into issues. [See our docs](#)

Issue creation preference
 Specify how we create issues and group incidents into them. (You get notifications when an issue opens, is acknowledged, and closes.)

☒ One issue per policy
 Group all incidents for this policy into one open issue at a time.

☐ One issue per condition
 Group incidents from each condition into a separate issue.

☐ One issue per condition and signal
 Group incidents sharing the same condition and signal into an issue.

With this Alert Policy two Alert Conditions had been associated as shown in the screenshot attached below.

Alerts / Alert Policies

Initial policy ☆ Metadata Workloads

ID: [REDACTED]

Alert conditions Notifications Settings

+ New alert condition

Search by condition name or id Condition Name = All +

Showing 2 conditions

Alert condition	Query	Thresholds	Type	Open issues	Last modified
App-URL	SELECT filter(count(*), WH...	Critical: above 0 at least once in 1 minute Create a warning threshold	NRQL Query	0	[REDACTED]
CPU-Utilization	SELECT average(cpuPerce...	Critical: above 70 for at least 5 minutes Create a warning threshold	NRQL Query	0	[REDACTED]

The Workflow had been shown in the screenshot attached below which showed with this Alert Workflow an Alert Policy, and an Email Destination had been attached.

new relic

Quick Find Integrations & ... All Capabilities All Entities APM & Services Query Your Data Apps Browser Dashboards Errors Inbox Traces Infrastructure Logs Mobile What's New Add User Upgrade Now Ritesh Kumar S

Edit workflow

Workflow-App-URL

Give it a unique, descriptive name you'll recognize later

View as co... Manage tags ... Delete workfl... Enable workflow

Filter data

Select the kinds of issues you want to send. Use the basic filter for the most common attributes or the advanced filter for all attributes.

Basic Advanced

Tag Policy Priority

Initial policy [57] [0]

Filter is valid! Click here to see the full query.

Additional settings

Notify

Choose one or more destinations and add an optional message.

Activated Acknowledged Closed

Need help?

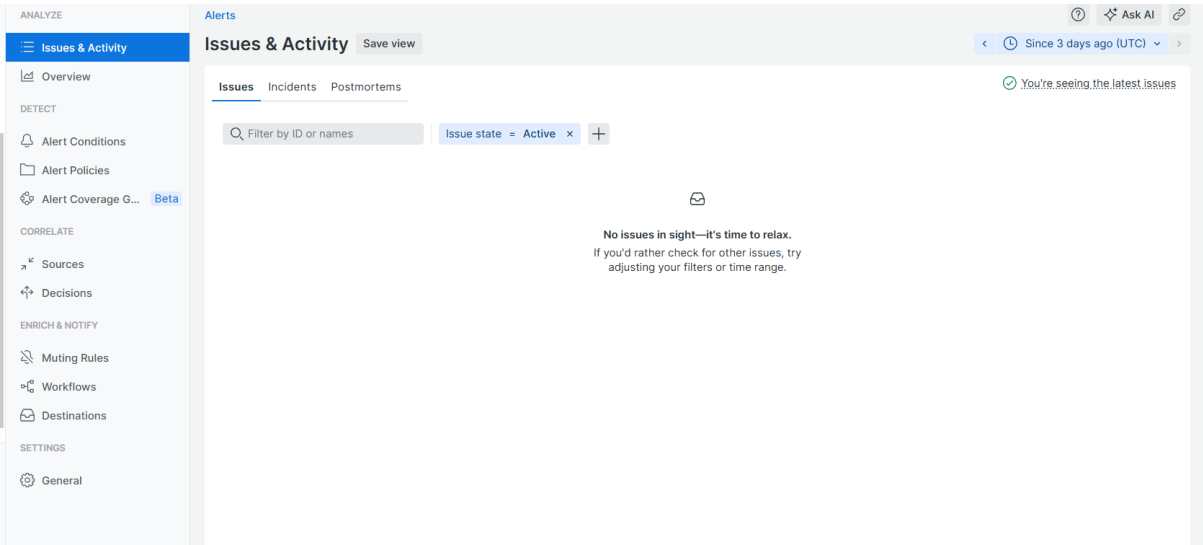
Workflow docs

Destination Docs Manage destinations Workflow triggers

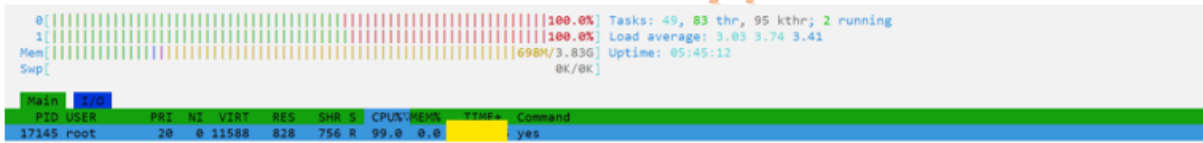
Cancel Update workflow

At present there is no Alert, but for the demonstration purpose I had created Alert by increasing the CPU Utilization more than 70% and I had deleted the Application URL <https://blogapp.singhritesh85.com> entry from the Azure DNS Zone Record set for which I had created Synthetic Monitoring to monitor this Application URL.

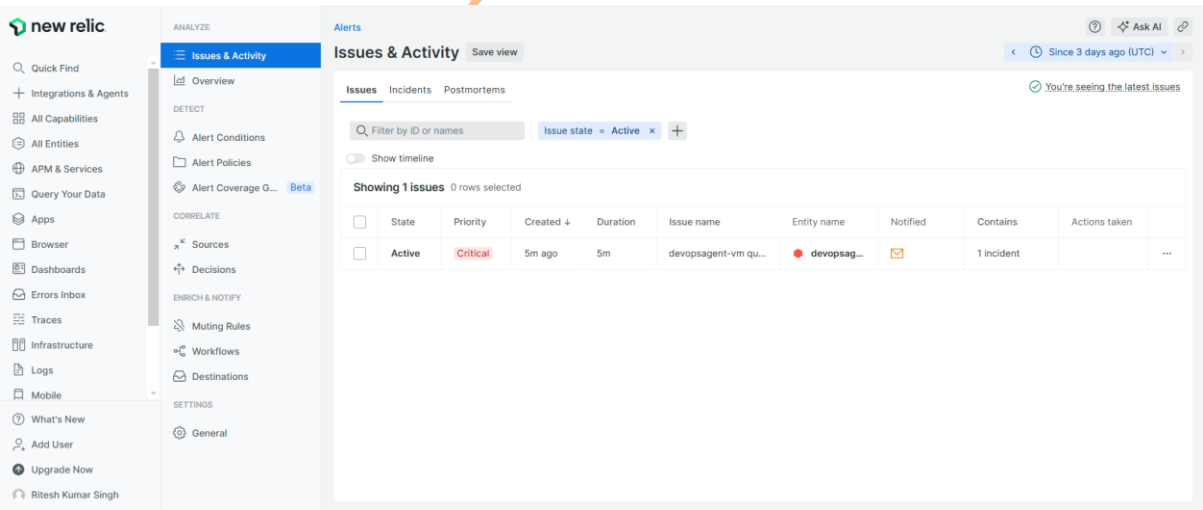
Below screenshot showed no Active alerts now.



For demonstration purpose I increased the CPU Utilization with the help of the command **yes > /dev/null &** and checked the same using the **htop** command. The screenshot for the same is as shown below.



Now the Alert is in Active condition as shown in the screenshot attached below.



An Email had been triggered to the Group Email ID as shown in the screenshot attached below. After the team received this Email on their Email Id, they will investigate its RCA (Root Cause Analysis) and will check the CPU usage using the command **htop**. Team will check whether any unnecessary crontab, any process is running which will utilize more CPU. If yes then edit the crontab and

comment out that entry for crontab or kill that process. If necessary, team will check the Log file to investigate this issue or if needed upgrade the VM Size of Azure VM.

devopsagent-vm query result is > 70.0 for 5 minutes on 'CPU-Utilization' Inbox x



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

[Unsubscribe](#)

new relic

Critical priority issue is active

devopsagent-vm query result is > 70.0 for 5 minutes on 'CPU-Utilization'

[Acknowledge](#) [Close issue](#) [Go to issue](#)

1 incidents

- devopsagent-vm query result is > 70.0 for 5 minutes on 'CPU-Utilization'

The Alert for Synthetic Monitoring is also in Active condition as shown in the screenshot attached below.

new relic

ANALYZE

Issues & Activity [Save view](#)

Issues Incidents Postmortems

Filter by ID or names Issue state = Active x

Show timeline

Showing 2 issues 0 rows selected

	State	Priority	Created	Duration	Issue name	Entity name	Notified	Contains
<input type="checkbox"/>	Active	Critical	1m ago	1m	dexter query result I...	dexter	<input checked="" type="checkbox"/>	1 incident
<input type="checkbox"/>	Active	Critical	5m ago	5m	devopsagent-vm qu...	devopsag...	<input checked="" type="checkbox"/>	1 incident

After the concerned team will get notification on their group Email Id as shown in the screenshot attached below, they will Acknowledge the issue and do RCA (Root Cause Analysis). As a part of RCA, they are supposed to check the entry in Azure DNS 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 Azure DNS Zone but, in your organization, if any one deleted an Azure Resource and you want to find out the person who deleted it then you need to check the Azure Monitor > Activity Log. Then filter out the Subscription into which the Azure Resource was existed and Timespan. You will be able to see the person who had initiated that Event.

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'

1 impacted entities

- dexter

Alert Policy

Policy Name	Initial policy
Condition	App-URL

Ritesh Kumar