RDS Monitoring

We need to enable the Enhanced Monitoring on the RDS to enable AWS RDS metrics. Please follow the below-given steps to enable enhanced monitoring.

To use enhanced monitoring, we must create an IAM role and then enable Enhanced Monitoring.

To create an IAM role for Amazon RDS enhanced monitoring

1. Open the IAM console at **https://console.aws.amazon.com**.
2. In the navigation pane, choose Roles.
3. Choose **Create role**.
4. Choose the **AWS service** tab, and then choose **RDS** from the list of services.
5. Choose **RDS - Enhanced Monitoring**, and then choose **Next**.
6. Ensure that the **Permissions policies** shows **AmazonRDSEnhancedMonitoringRole**, and then choose **Next**.
7. For **Role name**, enter a name for your role. For example, enter **demo**.
8. The trusted entity for your role is the AWS service **monitoring.rds.amazonaws.com**.
9. Choose **Create role**.

To turn on Enhanced Monitoring

1. Scroll to **Additional configuration**.
2. In **Monitoring**, choose **Enable Enhanced Monitoring** for your DB instance or read replica.
3. Set the **Monitoring Role** property to the IAM role that you created in the previous step to permit Amazon RDS to communicate with Amazon CloudWatch Logs for you, or choose **Default** to have RDS create a role for you named rds-monitoring-role.
4. Set the **Granularity** property to the interval, in seconds, between points when metrics are collected for your DB instance or read replica. The **Granularity** property can be set to one of the following values: 1, 5, 10, 15, 30, or 60.

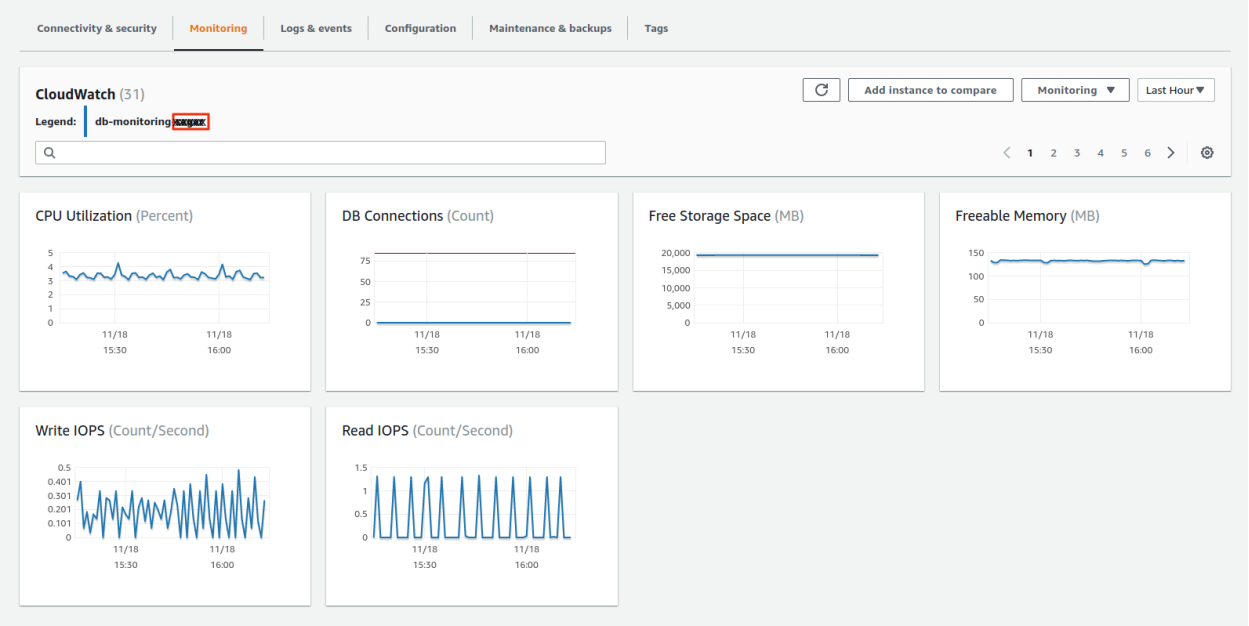
The fastest that the RDS console refreshes is every 5 seconds. If you set the granularity to 1 second in the RDS console, you still see updated metrics only every 5 seconds. You can retrieve 1-second metric updates by using CloudWatch Logs.

Cloudwatch to view metrics and for alerting

You can use cloudwatch to view metrics and implement alerts.

To view Enhanced metrics on cloudwatch

1. You can also view the metrics in the monitoring tab in the RDS console. This tab shows several metric graphs for each database.



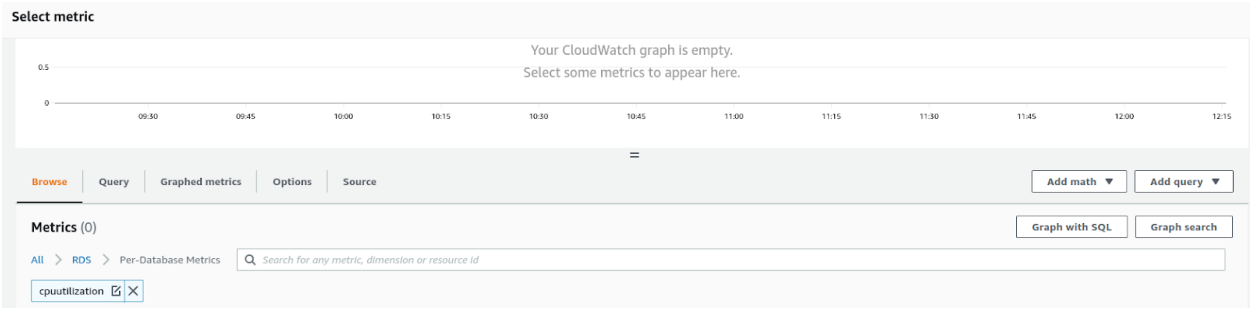
2. You can use the cloudwatch console also to view the metrics. For that, Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
3. Under **Metrics** click on **All Metrics.**
4. Select the AWS region.
5. Click on **Browse** and select **RDS,** now you can select the option that how you want to see the RDS metrics.

To implement alerts

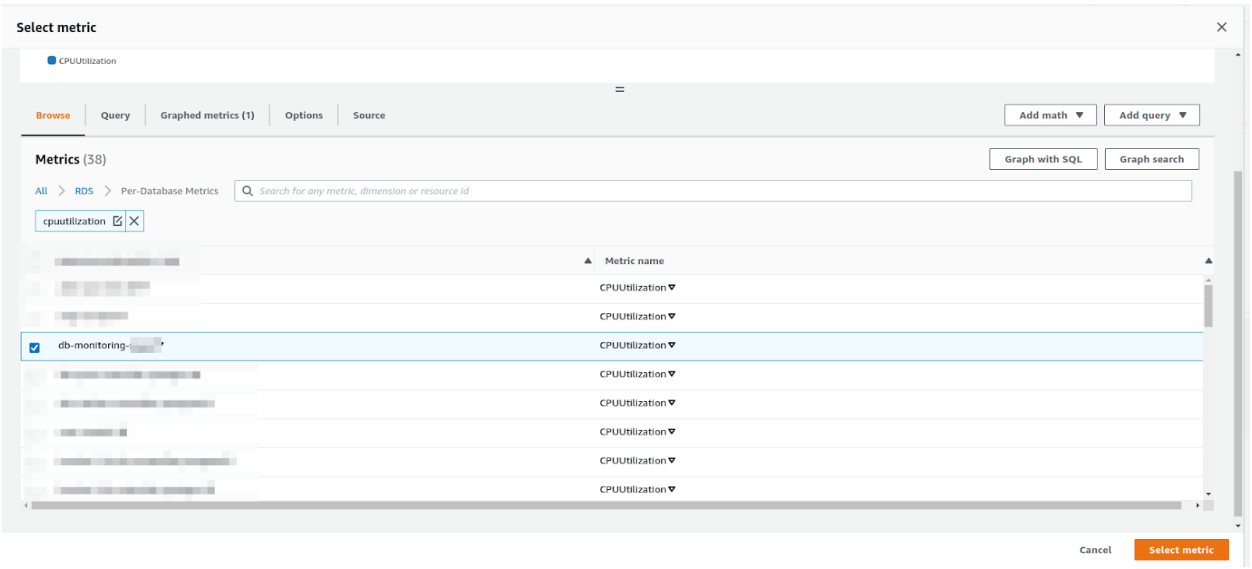
Navigate to the cloudwatch console because the Cloudwatch alarms are created from the cloudwatch console.

Click on **create alarm**.

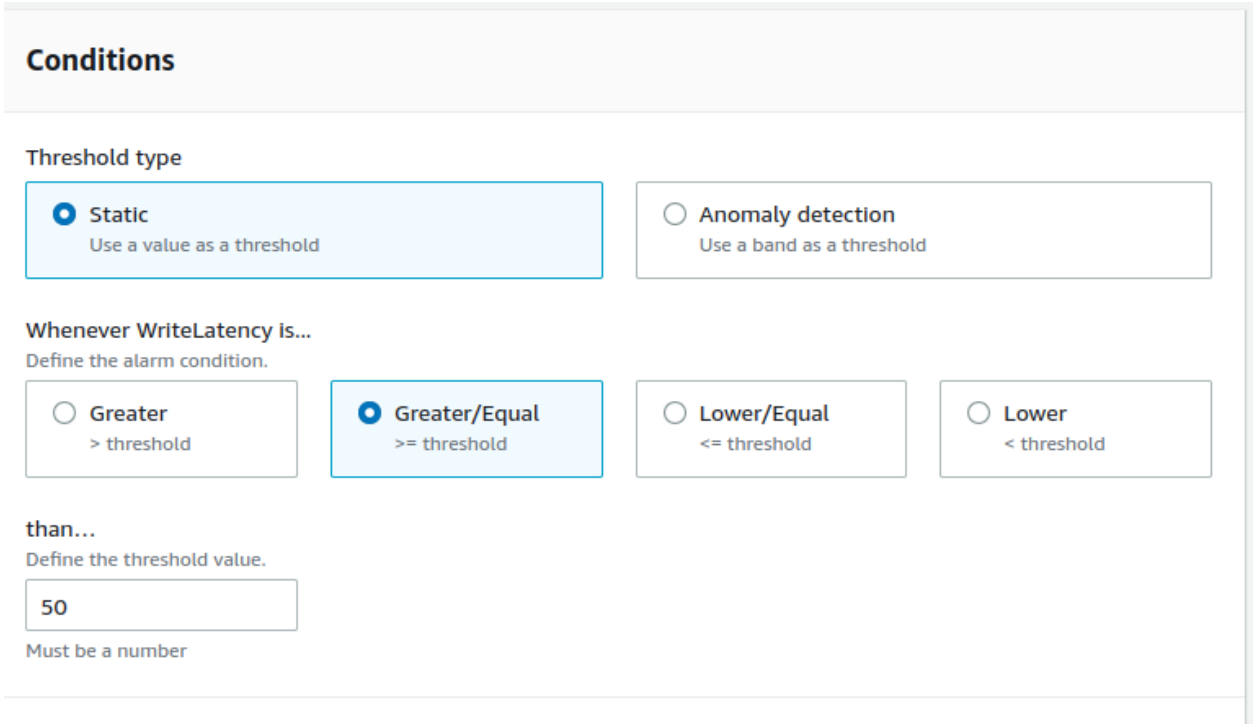
Click on **Select Metric** and type the name of the metric.



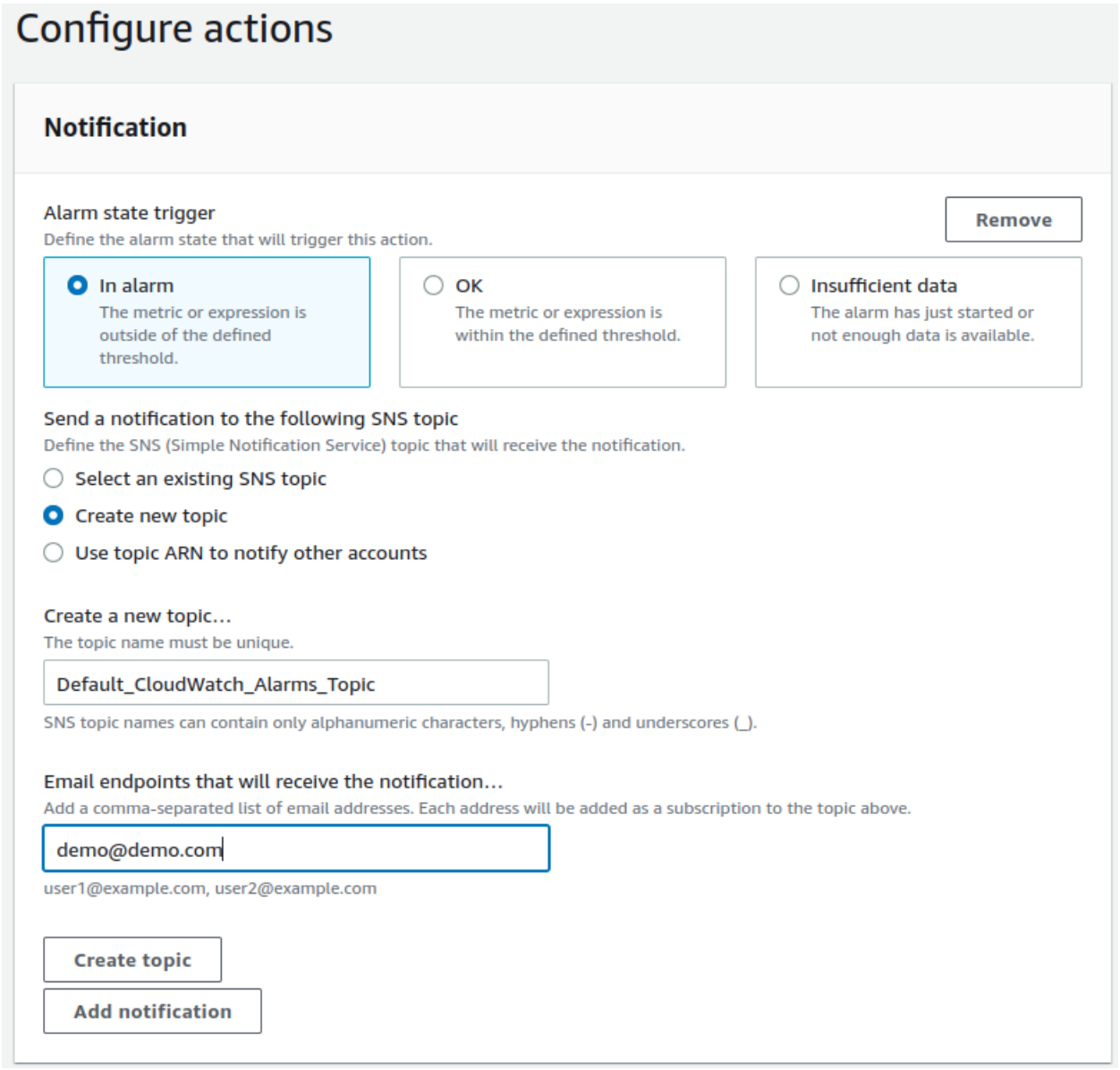
Choose the metric for the database you are going to monitor (which you can find in the tile labeled Per-Database Metrics). Click **Select Metric.**



Select the threshold for your alarm.

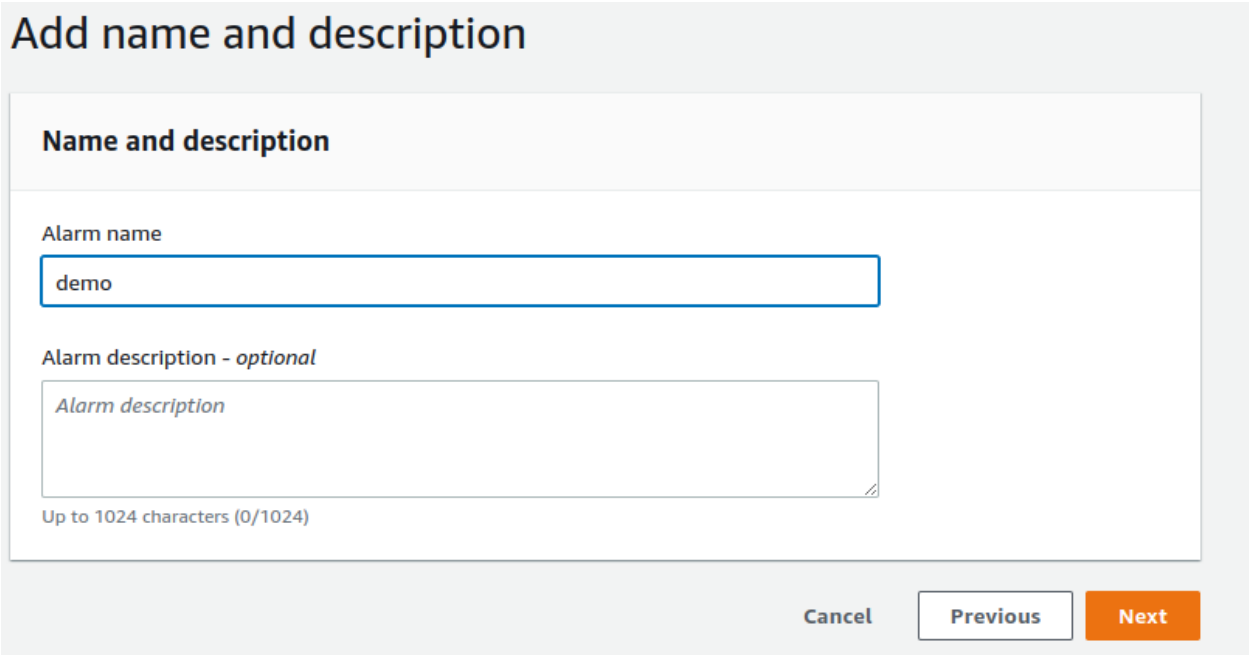


Click on **Next** and select the notification method for your alarm.

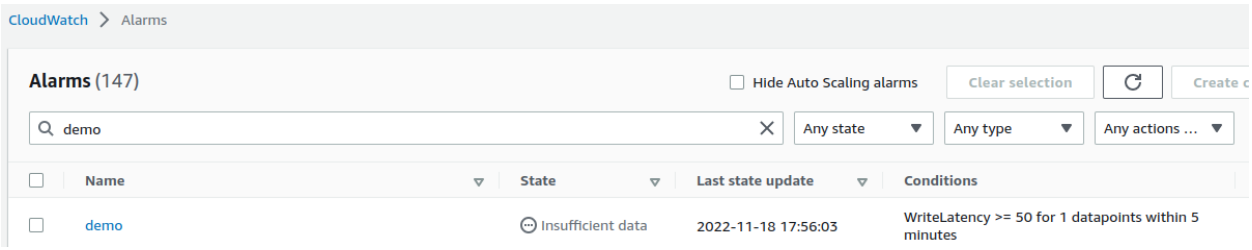


If your SNS topic is already created then you can select that and if you are creating a new topic then click on **Create topic.** Once done click on **Next**.

In the next window give your alarm a name.



Click on the **Next.** Now review your alarm and click on **Create alarm.**



Now your alarm is created you will receive a notification whenever this alarm is triggered.

Viewing metrics in grafana.

To view cloudwatch metrics in grafana we have 2 options: -

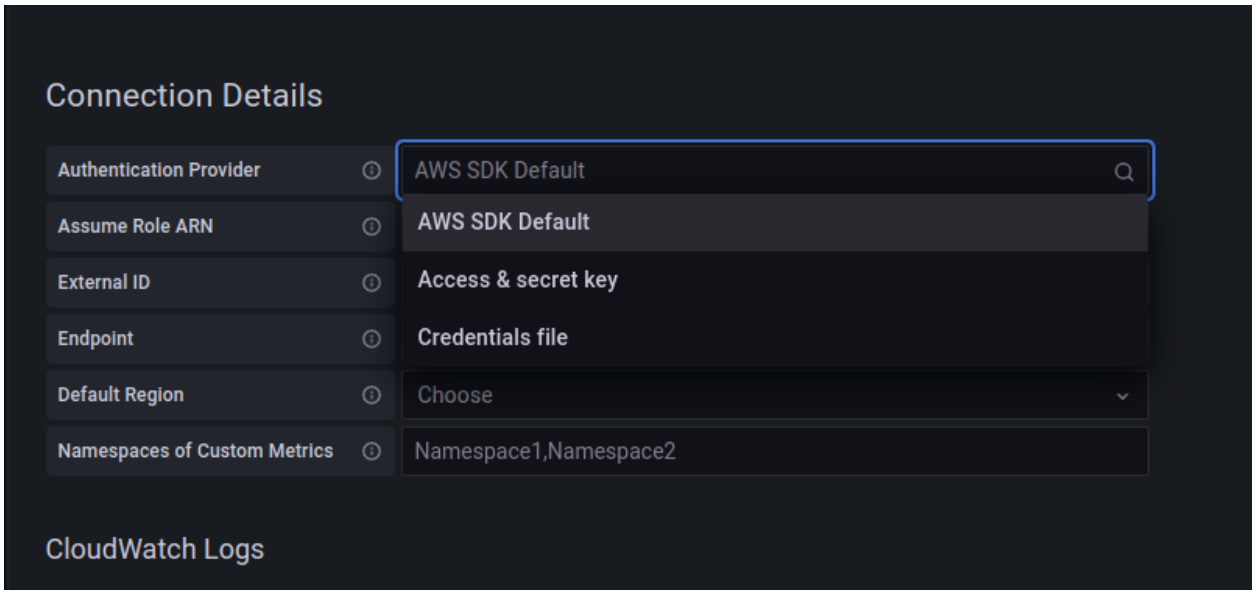
1. Adding cloudwatch source in grafana.
2. Adding metrics from Prometheus.

Before moving further we need to create an IAM role with read access to cloudwatch so that grafana/ prometheus can read data from cloudwatch. Follow this **link** to create IAM role

1. Adding cloudwatch source.

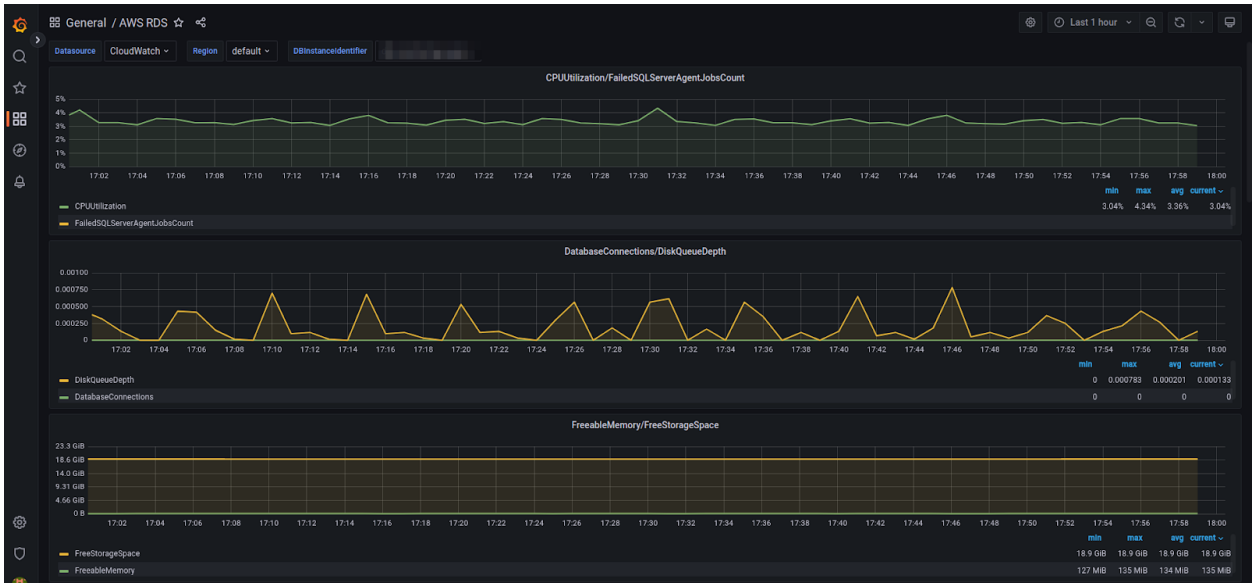
In this approach, we will use cloudwatch as a data source and in the setting of the data source, you will get the option to add the connection details. You can select the SDK Default in which

you have to pass the Role ARN that you have created in the previous step. If you want to use an AWS IAM User to access the cloudwatch, then you have to pass the access key and secret key in the grafana or you can create a credentials file with the access key and secret key and upload it onto the grafana.



Once you have entered your authentication provider details, select the **Default Region** and click on **Save & test**. Now your cloudwatch data source is added.

Import the dashboard to access the metrics. You can use the dashboard ID **707**, to access the metrics. Your dashboard will look like this.



2. Adding metrics from Prometheus.

To add metrics from cloudwatch we need to create a yace exporter in our EKS cluster. For that, we need an IAM user who has the privilege to read data from cloudwatch.

Create a credentials file with the access key and secret key of the user. Once done run the below-mentioned command.

```
cat ./credentials | base64
```

Change the path of the credentials file according to your scenario. Now create a deployment file with the following manifest.

```
apiVersion: v1
kind: Namespace
metadata:
 name: yace
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: yace-rds
 namespace: yace
spec:
 selector:
   matchLabels:
     app: yace-rds
 replicas: 1
 template:
   metadata:
     labels:
       app: yace-rds
     annotations:
       prometheus.io/scrape: "true"
       prometheus.io/port: "5000"
   spec:
     containers:
     - name: yace
       image: quay.io/invisionag/yet-another-cloudwatch-exporter:v0.21.0-alpha
       ports:
       - containerPort: 5000
       volumeMounts:
         - name: yace-rds-config
```

```
          mountPath: /tmp/config.yml
          subPath: config.yml
        - name: yace-rds-credentials
          mountPath: /exporter/.aws/credentials
          subPath: credentials
      resources:
        limits:
          memory: "128Mi"
          cpu: "500m"
    volumes:
      - configMap:
          defaultMode: 420
          name: yace-rds-config
        name: yace-rds-config
      - secret:
          defaultMode: 420
          secretName: yace-rds-credentials
        name: yace-rds-credentials
---
apiVersion: v1
kind: ConfigMap
metadata:
 name: yace-rds-config
 namespace: yace
data:
 config.yml: |
   discovery:
     jobs:
     - regions:
       - us-west-2
       type: rds
       enableMetricData: true
       metrics:
         - name: BinLogDiskUsage
           statistics:
           - Average
           period: 300
           length: 3600
         - name: BurstBalance
           statistics:
           - Average
           period: 300
           length: 3600
         - name: CPUUtilization
           statistics:
```

```
          - Average
          period: 300
          length: 3600
        - name: CPUCreditUsage
          statistics:
          - Average
          period: 300
          length: 3600
        - name: CPUCreditBalance
          statistics:
          - Average
          period: 300
          length: 3600
        - name: DatabaseConnections
          statistics:
          - Average
          period: 300
          length: 3600
        - name: DiskQueueDepth
          statistics:
          - Average
          - Maximum
          period: 300
          length: 3600
        - name: FailedSQLServerAgentJobsCount
          statistics:
          - Average
          period: 300
          length: 3600
        - name: FreeableMemory
          statistics:
          - Average
          period: 300
          length: 3600
        - name: FreeStorageSpace
          statistics:
          - Average
          period: 300
          length: 3600
        - name: MaximumUsedTransactionIDs
          statistics:
          - Average
          period: 300
          length: 3600
        - name: NetworkReceiveThroughput
```

```
      statistics:
      - Average
      period: 300
      length: 3600
    - name: NetworkTransmitThroughput
      statistics:
      - Average
      period: 300
      length: 3600
    - name: OldestReplicationSlotLag
      statistics:
      - Average
      period: 300
      length: 3600
    - name: ReadIOPS
      statistics:
      - Average
      period: 300
      length: 3600
    - name: ReadLatency
      statistics:
      - Maximum
      - Average
      period: 300
      length: 3600
    - name: ReadThroughput
      statistics:
      - Average
      period: 300
      length: 3600
    - name: ReplicaLag
      statistics:
      - Average
      period: 300
      length: 3600
    - name: ReplicationSlotDiskUsage
      statistics:
      - Average
      period: 300
      length: 3600
    - name: SwapUsage
      statistics:
      - Average
      period: 300
      length: 3600
```

```
        - name: TransactionLogsDiskUsage
          statistics:
          - Average
          period: 300
          length: 3600
        - name: TransactionLogsGeneration
          statistics:
          - Average
          period: 300
          length: 3600
        - name: WriteIOPS
          statistics:
          - Average
          period: 300
          length: 3600
        - name: WriteLatency
          statistics:
          - Maximum
          - Average
          period: 300
          length: 3600
        - name: WriteThroughput
          statistics:
          - Average
          period: 300
          length: 3600
---
apiVersion: v1
kind: Secret
metadata:
 name: yace-rds-credentials
 namespace: yace
data:
 # Add in credentials the result of:
 # cat ./credentials | base64
 credentials: |
   XXXX
--
apiVersion: v1
kind: Service
metadata:
  labels:
    app: yace-rds
  name: yace-svc
  namespace: yace
```

```
spec:
  ports:
  - port: 5000
    protocol: TCP
    targetPort: 5000
  selector:
    app: yace-rds
```

In the above manifest replace the value of region with your region in configmap and in secret replace XXX with the value you have received by running command **cat ./credentials | base64** in the previous step**.**

Before moving further deploy Prometheus on the cluster with the following configuration.

Add the job exporter in your Prometheus.

```
      - job_name: "yace-exporter"
        static_configs:
          - targets: ["yace-svc.yace.svc:5000"]
```

It will import the jobs from the yace pod and feed them to Prometheus. For alerting you need to add the below-mentioned alerting rule in Prometheus.

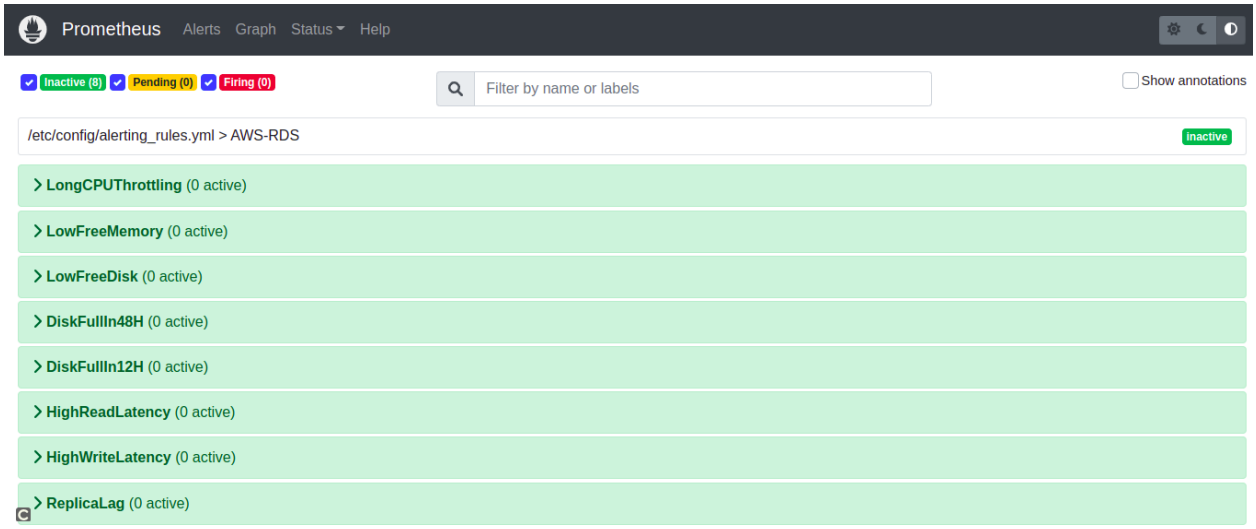
```
groups:
   - name: AWS-RDS
     rules:
     - alert: LongCPUThrottling
       expr: |
         aws_rds_cpuutilization_average > 95
       for: 10m
       labels:
         severity: page
       annotations:
         summary: CPU over 95% for 10 minutes in instance
{{$labels.dimension_DBInstanceIdentifier}}
     - alert: LowFreeMemory
       expr: |
         aws_rds_freeable_memory_average < 128*1024*1024
       for: 10m
       labels:
         severity: page
       annotations:
         summary: Free memory under 128MB in instance
```

```
{{$labels.dimension_DBInstanceIdentifier}}
     - alert: LowFreeDisk
       expr: |
         aws_rds_free_storage_space_average < 512*1024*1024
       for: 10m
       labels:
         severity: page
       annotations:
         summary: Free disk under 512MB
{{$labels.dimension_DBInstanceIdentifier}}
     - alert: DiskFullIn48H
       expr: |
         predict_linear(aws_rds_free_storage_space_average[48h], 48 * 3600) < 0
       for: 10m
       labels:
         severity: warning
       annotations:
         summary: Disk will be full within 48 hours in instance
{{$labels.dimension_DBInstanceIdentifier}}
     - alert: DiskFullIn12H
       expr: |
         predict_linear(aws_rds_free_storage_space_average[12h], 12 * 3600) < 0
       for: 10m
       labels:
         severity: page
       annotations:
         summary: Disk will be full within 12 hours in instance
{{$labels.dimension_DBInstanceIdentifier}}
     - alert: HighReadLatency
       expr: |
         aws_rds_read_latency_average > 0.250
       for: 10m
       labels:
         severity: warning
       annotations:
         summary: Average read latency over 250ms in instance
{{$labels.dimension_DBInstanceIdentifier}}
     - alert: HighWriteLatency
       expr: |
         aws_rds_write_latency_average > 0.250
       for: 10m
       labels:
         severity: warning
       annotations:
         summary: Average write latency over 250ms in instance
```

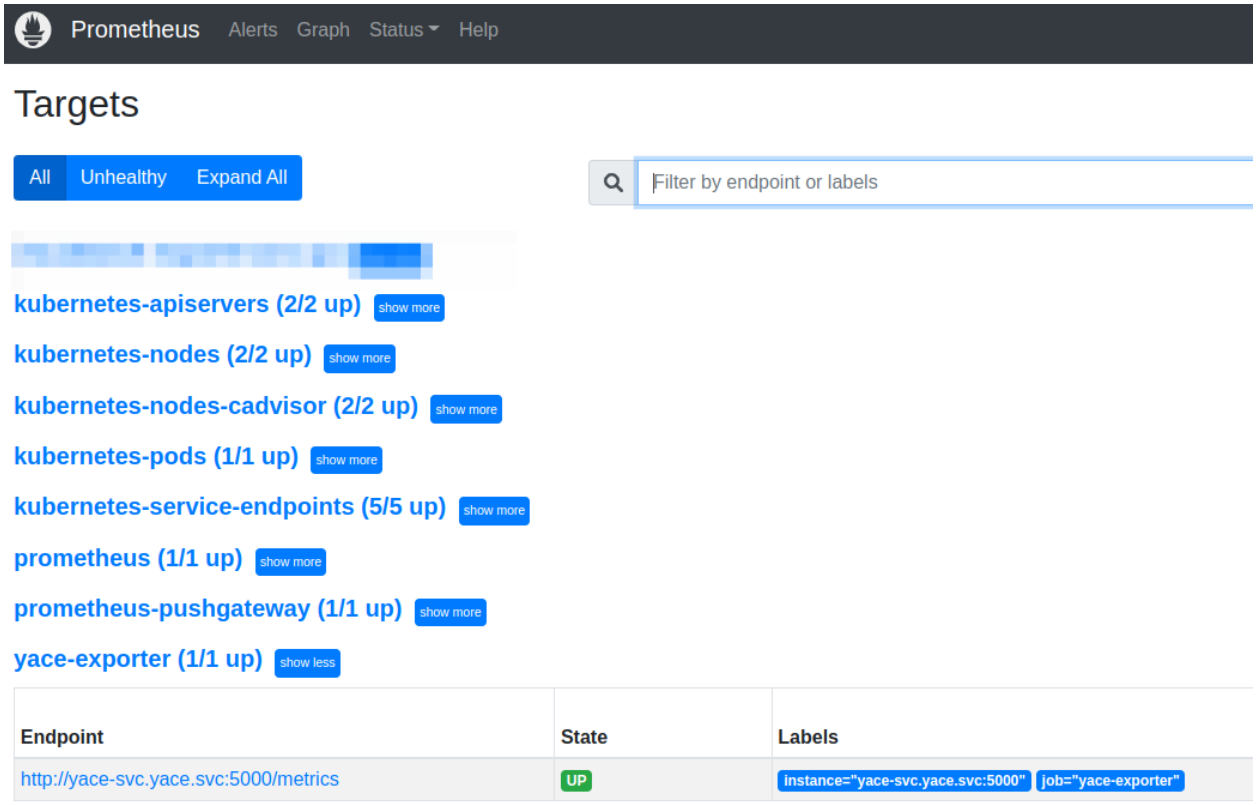
```
{{$labels.dimension_DBInstanceIdentifier}}
     - alert: HighDiskQueue
       expr: |
         aws_rds_disk_queue_depth_average > 25
       for: 10m
       labels:
         severity: warning
       annotations:
         summary: High disk queue depth in instance
{{$labels.dimension_DBInstanceIdentifier}}
          - alert: ReplicaLag
       expr: |
         aws_rds_oldest_replication_slot_lag_average > 10
       for: 10m
       labels:
        severity: warning
       annotations:
        summary: Average replication slot lag
{{$labels.dimension_DBInstanceIdentifier}}
```

This configuration will enable the alerts on cloudwatch metrics, you can change the values in the configuration according to your need.

Once all the configuration is done you can see the alerts on the Prometheus.



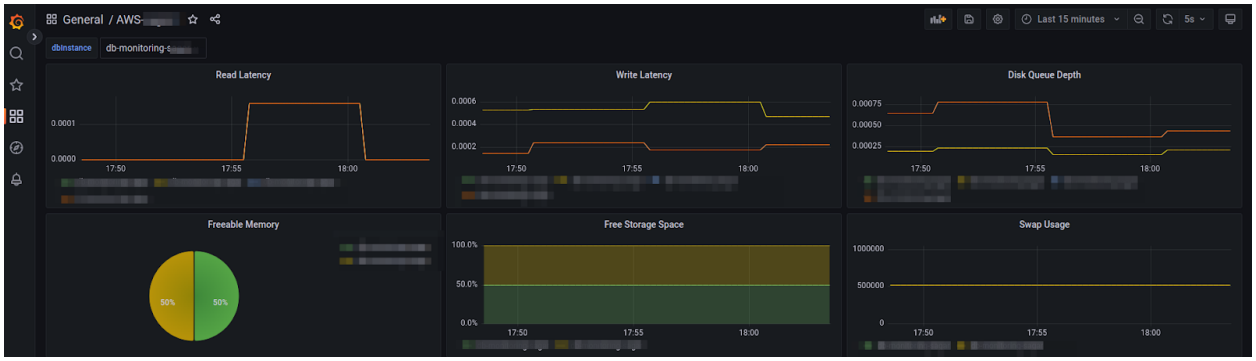
And under the targets, you can see the yace-exporter is up.



Now on grafana add Prometheus as a data source and import this dashboard. Use the wget command to download the dashboard.

```
wget
https://raw.githubusercontent.com/sagar-rafay/grafana-files/main/rds-promcat-dashboard.json
```

Once done import the dashboard on your grafana and explore the metrics. You can also implement alerts from the grafana dashboard.



References: -

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Monitoring.html
https://promcat.io/apps/aws-rds
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch-Grafana-support.html
https://www.bluematador.com/blog/how-to-monitor-amazon-rds-with-cloudwatch
https://medium.com/@_oleksii_/using-aws-cloudwatch-in-grafana-8294b7a2e7dd