|  |  |
| --- | --- |
| **Project Title** | **Prometheus Monitoring and Alerting for Multiple EC2 Instances in Multiple Accounts with Slack Integration** |
| **Technologies** | **EC2, Prometheus, Alert, Alert Manager, Slack** |

**Problem Statement:**

Our organization has a number of Amazon EC2 instances running in multiple accounts, and we need a way to monitor their status and receive alerts when any of them go down. We also want to be able to easily collaborate with our team members when issues arise. Currently, we do not have a reliable system in place to do this, and as a result, we are at risk of experiencing prolonged downtime and reduced visibility into the performance and health of our instances. To address this, we want to set up a Prometheus-based monitoring and alerting system that integrates with Slack, so that we can receive notifications when any of the instances go down and collaborate on resolving any issues that arise.

**Approach:**

To create a Prometheus monitoring system for multiple Amazon Elastic Compute Cloud (EC2) instances in multiple accounts and send alerts to a Slack community when any of the servers are down, you will need to perform the following steps:

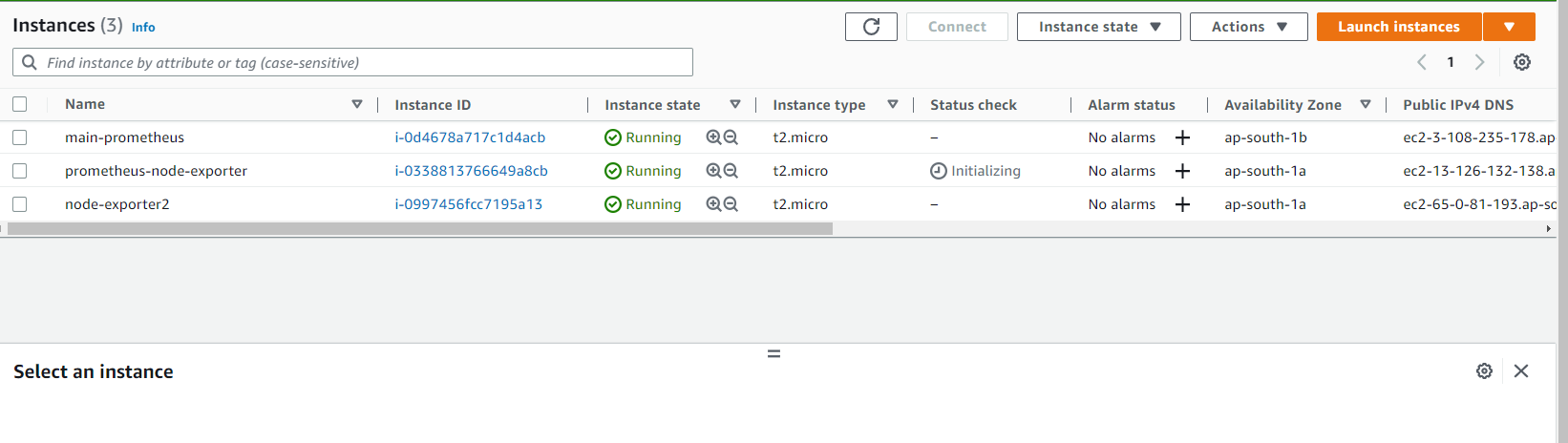
1. Set up a Prometheus server on one of the EC2 instances. This can be done by following the instructions in the Prometheus documentation.
2. Set up one or more Prometheus exporters on each of the other EC2 instances that you want to monitor. An exporter is a piece of software that runs on a host and exposes metrics in a format that Prometheus can scrape. There are many exporters available for different types of systems, such as the Node Exporter for Linux servers and the MySQL Exporter for databases.
3. Configure the Prometheus server to scrape metrics from the exporters. This can be done by modifying the configuration file for the server and adding a job configuration for each exporter.
4. Set up an alerting rule in Prometheus to send a notification to a Slack channel when a server goes down. This can be done by modifying the configuration file for the server and adding an alerting rule that specifies the condition under which the alert should be triggered and the Slack webhook to which the alert should be sent.
5. Test the system to ensure that it is working as expected. This can be done by simulating a server failure and verifying that the alert is sent to the Slack channel.

Some potential benefits of this system may include:

* Improved uptime: By monitoring the status of the EC2 instances and being alerted when any of them go down, you can take action to restore service as quickly as possible.
* Increased visibility: The Prometheus server and exporters will collect a wide range of metrics about the performance and health of the EC2 instances, which can help you identify trends and issues that may not be immediately apparent.
* Enhanced collaboration: By sending alerts to a Slack channel, you can make it easier for team members to stay informed about the status of the EC2 instances and collaborate on resolving any issues that arise.
* Overall, the result of this system should be a more reliable, transparent, and collaborative way to manage the EC2 instances in your environment.

**SOLUTION:**

**1.Create 3 Ec2 Instances, 1 being Prometheus server, rest being node exporters.**



**2. Install the Prometheus service on one of EC2 instance.**

Download the Prometheus monitoring system from the below URL for Linux instances.

Go <https://github.com/prometheus/prometheus/releases/download/v2.42.0/prometheus-2.42.0.linux-amd64.tar.gz>

1.cd /opt

2.sudo wget <https://github.com/prometheus/prometheus/releases/download/v2.42.0/prometheus-2.42.0.linux-amd64.tar.gz>

3.ls

aws Prometheus-2.42.0.linux-amd64.tar.gz rh

4.sudo tar xvf Prometheus-2.42.0.linux-amd64.tar.gz

5.ls

aws Prometheus-2.42.0.linux-amd64 Prometheus-2.42.0.linux-amd64.tar.gz rh

6 rename using mv

Sudo mv Prometheus-2.42.0.linux-amd64.tar.gz Prometheus

7.cd Prometheus

8.ls

Console\_libraries consoles LICENSE NOTICE Prometheus Prometheus.yml promtool

9. sudo vi Prometheus.yml

Add the prometheuse service IP:port number and exporter in this yml file of static-configs .

i.e prometehuserverip:9100, so that it will listen to node exporter and pull the metrics from node exporter server.

[ec2-user@ip-172-31-45-160 prometheus]$ cat prometheus.yml

=====================================================================================================================

Text

Description automatically generated

10. sudo ./Prometheus 🡪 to start the prometheus server

11. You can go to web browser and paste the public IP of server:9090

Graphical user interface, text, application, email

Description automatically generated

12.prometheus->status->targets

Graphical user interface, text, application

Description automatically generated

**3. Setup node exporter on one of the Ec2 instance:**

We use node exporter to collect: stats like cpu utilizations, servers status, disk space etc..

Node exporter configuration steps are here

[root@ip-172-31-45-214 opt]# history

1 cd /opt

2 sudo wget https://github.com/prometheus/node\_exporter/releases/download/v1.5.0/node\_exporter-1.5.0.linux-amd64.tar.gz

3 ls

4 sudo tar xf node\_exporter-1.5.0.linux-amd64.tar.gz

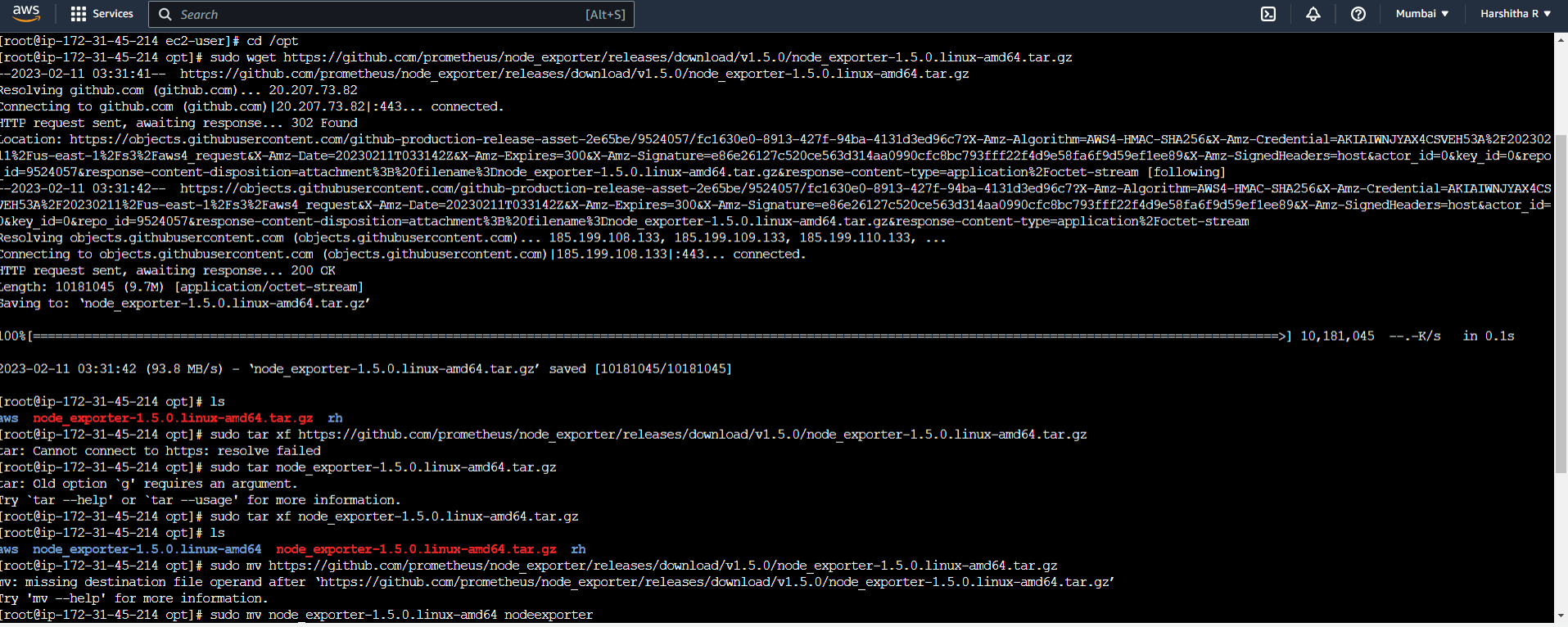
5 ls

6 Rename a new folder called nodeexporter using mv command

7 sudo mv node\_exporter-1.5.0.linux-amd64 nodeexporter

8 ls

aws nodeexporter node\_exporter-1.5.0.linux-amd64.tar.gz rh



**To start the node exporter:**

[root@ip-172-31-45-214 opt]# cd nodeexporter/

[root@ip-172-31-45-214 nodeexporter]# ls

LICENSE node\_exporter NOTICE

[root@ip-172-31-45-214 nodeexporter]# sudo ./node\_exporter

A picture containing graphical user interface

Description automatically generated

Add the nodexporter IP and port number in promethus.yml file , in order to Prometheus to collect the stats from node exporter

**4. REPEAT the same install the node exporter in one of other EC2 instance**

**5. ALERT MANAGER**

Install the alertmanager in one of EC2 where the Prometheus has been installed.

Download and install Prometheus alert manager

Wget <https://github.com/prometheus/alertmanager/releases/download/v0.25.0/alertmanager-0.25.0.linux-amd64.tar.gz>

alertmanager.yml file - adding slack webhooks

Text

Description automatically generated

Then start alertmanager

./alertmanager

Add the alert manager port number to prometehus.yml file

And the alert rule.

Alert\_rule.yml

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

The final prometehus.yml file after adding the alertmanager,node exporters,alert\_rules.

Text

Description automatically generated

Restart the Prometheus service after editing the Prometheus.yml

<https://linuxhint.com/install_prometheus_ubuntu/>

Add the alert.rules.yml in the same path as Prometheus.yml

5. Create an slack account and a slack channel-> app-monitoring

Graphical user interface, text, application, email

Description automatically generated

6.Create the App -> api.slack.com/apps

Eg: Monitoring

Graphical user interface, text, application, email

Description automatically generated

7.click on incoming webhook and activate incoming webhooks.

8.Specify the channel in which we want to trigger the alert, the webhook facilitates alert to trigger in that channel.

Graphical user interface, application

Description automatically generated

1. Copy the webhook and configure that webhook in alert manager which acts as receiver.

Graphical user interface, text, application, email

Description automatically generated

9.Now configure the alertmanager.yml

Cd /opt/alertmanager

Vi alertmanager.yml

Add the slack webhooks in alertmanager.yml

Text

Description automatically generated

Stop one of the services:

Example : sudo systemctl stop node\_exporter

The alert gets triggered and fires via alert manager and notifies in slack

Prometheus alert-> firing

Graphical user interface, text, application, email, website

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**OPTIONAL:**

3. Grafana installation:

Login to a ec2 insatnce

1. sudo su

2.cd /opt

3. sudo wget <https://dl.grafana.com/enterprise/release/grafana-enterprise-9.3.6.linux-amd64.tar.gz>

4 ls

5.sudo tar xf <https://dl.grafana.com/enterprise/release/grafana-enterprise-9.3.6.linux-amd64.tar.gz>

6. follow this doc for graphana installation:

<https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/>

<https://grafana.com/docs/grafana/latest/setup-grafana/installation/rpm/>

root@ip-172-31-12-17 ~]# history

1 cd ~

2 sudo vi /etc/yum.repos.d/grafana.repo

Paste this in Grafana.repo file::

[grafana]

name=grafana

baseurl=https://rpm.grafana.com

repo\_gpgcheck=1

enabled=1

gpgcheck=1

gpgkey=https://rpm.grafana.com/gpg.key

sslverify=1

sslcacert=/etc/pki/tls/certs/ca-bundle.crt

3 sudo yum install grafana

4 sudo systemctl start grafana-server

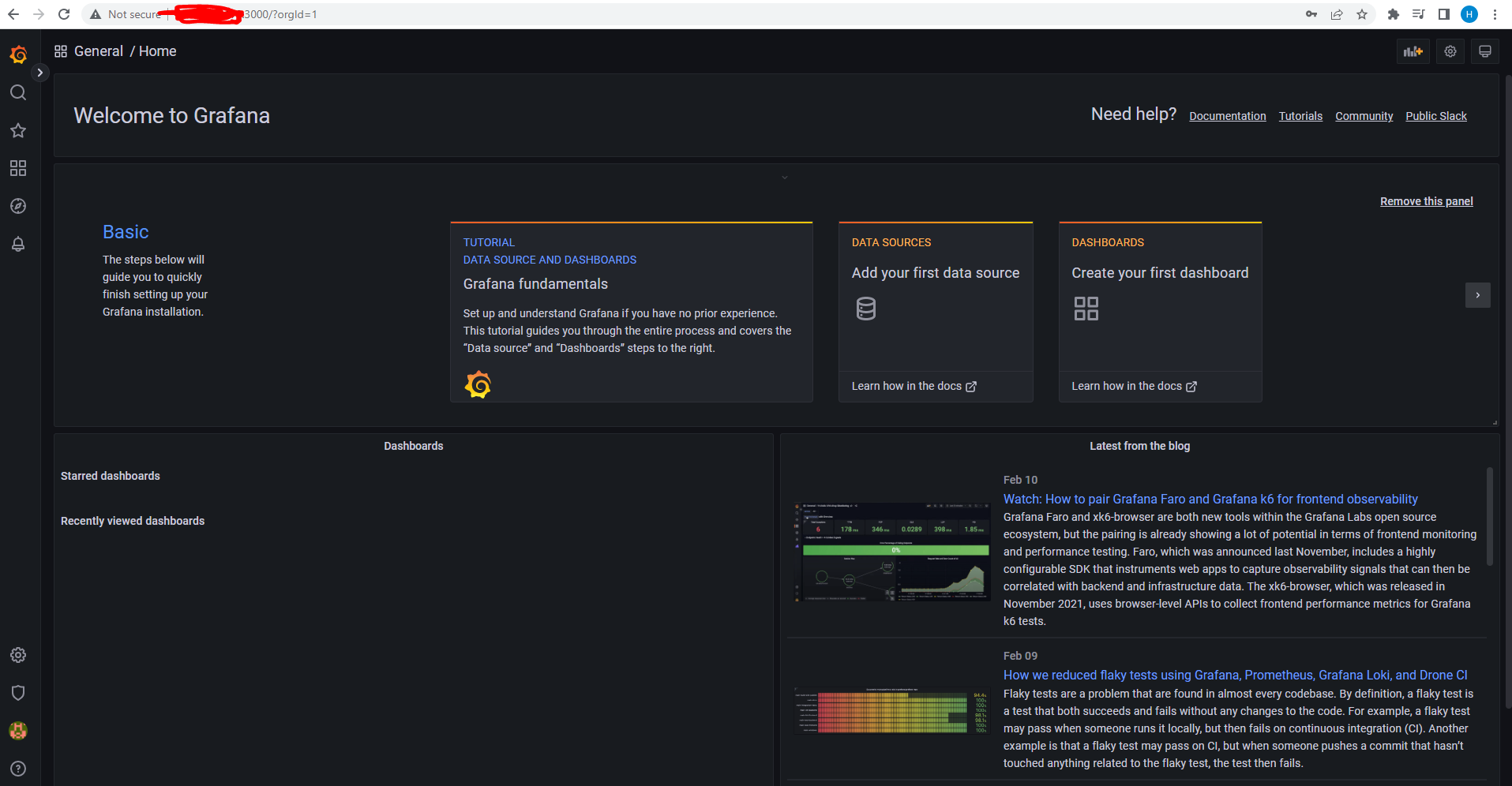
5 sudo systemctl status grafana-server

1. To enable Grafana automatically upon system boot

[root@ip-172-31-12-17 ~]# sudo systemctl enable grafana-server

Created symlink from /etc/systemd/system/multi-user.target.wants/grafana-server.service to /usr/lib/systemd/system/grafana-server.service.

8.port number is 3000



9.Add the datasource-prometheusserverip:9090

Graphical user interface, application, Teams

Description automatically generated

10. go to dashboard->import

Nodexporter -1869

Graphical user interface, application, Teams

Description automatically generated

