**Application Monitoring with Prometheus, Node Exporter, Blackbox Exporter & Grafana**

ThisPOC demonstrates how to monitor an application server and external endpoints using Prometheus, Node Exporter, Blackbox Exporter, and Grafana dashboard

**Project Index:**

**Step 1: Provision Cloud Servers**: Setting up the two required cloud instances.  
**Step 2: Clone and Build the Application**: Getting the application ready on Server-1.  
**Step 3: Run the Application**: Starting the application on Server-1.  
**Step 4: Verify Application Access**: Confirming the application is running correctly.  
**Step 5: Install and Configure Node Exporter**: Installing and running the Node Exporter on Server-1 to collect system metrics.  
**Step 6: Install Prometheus on Monitoring Server**: Setting up Prometheus on Server-2.  
**Step 7: Run and Configure Prometheus**: Configuring Prometheus to scrape metrics from the application and Node Exporter.  
**Step 8: Configure Blackbox Exporter**: Understanding the purpose and function of Blackbox Exporter.  
**Step 9: Prometheus Configuration for Blackbox Exporter**: Integrating Blackbox Exporter with Prometheus for external monitoring.  
**Install Grafana**: Setting up the visualization tool.  
**Access Grafana UI**: Logging in to the Grafana dashboard.  
**Add Prometheus to Grafana as Data source**: Connecting Grafana to Prometheus.  
**Add Dashboards**: Importing pre-built dashboards for Node Exporter and Blackbox Exporter.

**Step-by-Step Guide: Setting Up Prometheus and Grafana**

**Step 1: Provision Cloud Servers**

Two cloud instances were provisioned   
 **Server-1** → **Application** + **Node Exporter**  
**Server-2** → **Prometheus + Grafana**

* Both servers were made accessible via SSH for configuration and deployment.

**A screenshot of a computer

AI-generated content may be incorrect.**

**These below rules should be added for security group**

**A screenshot of a computer

AI-generated content may be incorrect.**

**GITHUB URL : https://github.com/panthangiEshwary/application\_monitoring**

**Step 2: Clone and Build the Application**

On **Server-1**, the application repository was cloned from GitHub and built using Maven:

Navigate to the application folder:

sudo apt update  
git clone https://github.com/panthangiEshwary/application\_monitoring.git  
apt install openjdk-17-jre-headless  
apt install maven  
ls  
cd Task-Master-Pro  
mvn package

The Maven build generated a target folder containing the packaged application JAR file.

A screenshot of a computer program

AI-generated content may be incorrect.

We could see target folder now



Now go inside target folder and run below commands

**Step 3: Run the Application**

The application was executed using the following command:

java -jar target/todo-app-1.0-SNAPSHOT.jar

* For background execution:

java -jar target/todo-app-1.0-SNAPSHOT.jar &

A computer screen with text

AI-generated content may be incorrect.

**Step 4: Verify Application Access**

* Once the application was up and running, it was accessible on port **8080**.
* Application URL : [http://54.242.101.157:8080/login](http://54.242.101.157:8080/login?utm_source=chatgpt.com)
* The login page was successfully loaded, confirming that the application is running properly on the cloud server.

A screenshot of a computer login

AI-generated content may be incorrect.

**Step 5: Install and Configure Node Exporter**

To collect **server-level metrics** (CPU, memory, disk, and networking), we deployed **Node Exporter** on the cloud server.

**Step 5.1: Download Node Exporter**

* The Linux binary for Node Exporter v1.9.1 was downloaded from the official Prometheus GitHub releases
* DOC: <https://prometheus.io/download/>

We have to search for node expoter

A screenshot of a computer

AI-generated content may be incorrect.

Copy the linux Link: <https://github.com/prometheus/node_exporter/releases/download/v1.9.1/node_exporter-1.9.1.linux-amd64.tar.gz>

Now run the commad   
  
wget <https://github.com/prometheus/node_exporter/releases/download/v1.9.1/node_exporter-1.9.1.linux-amd64.tar.gz>

A screen shot of a computer

AI-generated content may be incorrect.

Now the file downloaded as tar shown below



**Step 5.2: Extract and Prepare the Binary**

* Extracted the downloaded tarball, removed the archive, and renamed the folder for simplicity:

tar -xvf node\_exporter-1.9.1.linux-amd64.tar.gz

rm node\_exporter-1.9.1.linux-amd64.tar.gz

mv node\_exporter-1.9.1.linux-amd64 node\_exporter

cd node\_exporter

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

**Step 5.3: Start Node Exporter**

* Node Exporter was started in the background:

./node\_exporter &

A black screen with white text

AI-generated content may be incorrect.

**Step 5.4: Verify Node Exporter**

* By default, Node Exporter listens on **port 9100**.
* Verified successful installation by accessing the metrics endpoint in a browser:

[http://54.242.101.157:9100/](http://54.242.101.157:9100/?utm_source=chatgpt.com)

* This confirmed that the server is now exposing metrics, ready to be scraped by **Prometheus**.A screenshot of a computer

  AI-generated content may be incorrect.

Now my Node exporter setup is running fine.

Here I could see metrics as well  
  
A screenshot of a computer

AI-generated content may be incorrect.

Now connect to second server.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 6: Install Prometheus on Monitoring Server**

To collect and store metrics, **Prometheus** was installed on the second cloud server (Monitoring server).

Doc used: <https://prometheus.io/download/>

**Step 6.1: Download Prometheus**

* The Prometheus binary was downloaded from the official releases page:

A screenshot of a computer

AI-generated content may be incorrect.

#wget <https://github.com/prometheus/prometheus/releases/download/v3.6.0-rc.0/prometheus-3.6.0-rc.0.linux-amd64.tar.gz>

A screenshot of a computer

AI-generated content may be incorrect.

Now it got downloaded as tar file  


**Step 6.2: Extract and Prepare Prometheus**

* Extracted the tarball, removed the archive, and renamed the folder for easier management:

A screenshot of a computer

AI-generated content may be incorrect.

Now I just renamed it as promotheus using below commnd

#mv prometheus-3.6.0-rc.0.linux-amd64/ Prometheus

Now file shows as below  


Lets go inside Prometheus and could see the executable file **prometheus.yml**

**Step 6.3: Verify Installation**

* The Prometheus directory now contains:
  + **prometheus** (main binary)
  + **promtool** (validation utility)
  + **prometheus.yml** (default config file)

At this stage, Prometheus is successfully installed on **Server-2**.



**Step 7: Run and Configure Prometheus**

**Step 7.1: Start Prometheus**

* On **Server-2 (Monitoring Server)**, Prometheus was started in the background:

./prometheus &

* By default, Prometheus runs on **port 9090**.
* Verified by accessing the web interface:

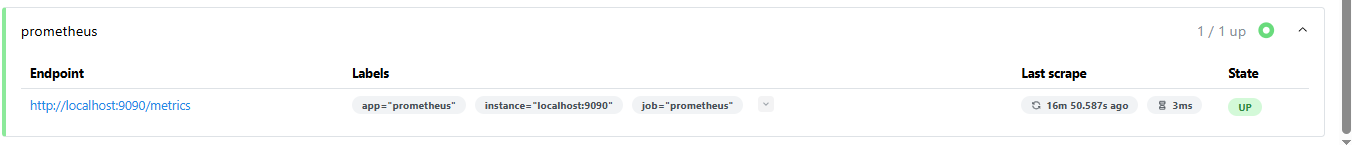
[http://98.84.141.121:9090](http://98.84.141.121:9090?utm_source=chatgpt.com)

A computer screen with green text

AI-generated content may be incorrect.

**Step 7.2: Verify Prometheus Targets**

* After starting Prometheus, we opened the **Targets page**:  
   [http://98.84.141.121:9090/targets](http://98.84.141.121:9090/targets?utm_source=chatgpt.com)
* At this stage, only the **default Prometheus self-monitoring endpoint** was visible.
* This means Prometheus was running correctly but not yet scraping any external services (like Node Exporter or the Application).



**Step 7.3: Add Application and Node Exporter as Targets**

Prometheus decides **what to monitor** through the scrape\_configs section inside prometheus.yml.

**Key Concepts:**

* **job\_name** → A label to identify a set of monitored endpoints.
* **targets** → The list of endpoints (IP:Port) that Prometheus scrapes for metrics.
* Prometheus scrapes all listed targets at the configured **scrape\_interval** (default: 15 seconds).

To scrape metrics from the **Application Server (Server-1)**, we edited the **prometheus.yml** configuration file:



A computer screen with blue text

AI-generated content may be incorrect.

We need to edit this scrape\_configs file section:

Example Scrape Job Format

Each scrape\_config looks like this:

scrape\_configs:  
 - job\_name: "node\_exporter"  
 static\_configs:  
 - targets: ["<server-ip>:9100"]

Here, <server-ip>:9100 is the address where Node Exporter is running.

**Adding Node Exporter Target**

In our setup, Node Exporter is running on **Server-1 (Application Server)** at IP 54.242.101.157 on port 9100.  
We added the following block to prometheus.yml:

- job\_name: "node\_exporter" # scraping Node Exporter on another server  
 static\_configs:  
 - targets: ["54.242.101.157:9100"]  
 labels:  
 app: "node\_exporter"

**A screenshot of a computer

AI-generated content may be incorrect.**

**Explanation:**

* **job\_name: "node\_exporter"** → Identifies this as the Node Exporter monitoring job.
* **targets: ["54.242.101.157:9100"]** → Prometheus will scrape metrics from Node Exporter running on Server-1.
* **labels: { app: "node\_exporter" }** → Adds a custom label to help filter/organize metrics in queries and dashboards.

**Apply Changes**After saving the configuration file, Prometheus was restarted to load the new settings:

pkill prometheus

./prometheus &

**Verification:**

* Navigated again to [http://98.84.141.121:9090/targets](http://98.84.141.121:9090/targets?utm_source=chatgpt.com)
* Verified that a new target appeared:

54.242.101.157:9100 (Node Exporter)

* Its health status showed **UP**, confirming successful scraping.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a chat

AI-generated content may be incorrect.

Prometheus is now **collecting system metrics** from the application server through Node Exporter.

**Step 8: Configure Blackbox Exporter**

The **Blackbox Exporter** allows Prometheus to monitor endpoints that don’t have a native exporter, such as websites, APIs, or external services. It works differently from Node Exporter because Prometheus does **not scrape the endpoint directly**. Instead, Prometheus scrapes the Blackbox Exporter, which performs the probe on the target.

The **Blackbox Exporter** works a bit differently from Node Exporter, so the scrape\_configs for it need some special handling. Let me break it down step by step.

**8.1. What Blackbox Exporter Does**

* Blackbox Exporter is used to **probe endpoints over HTTP, HTTPS, DNS, TCP, ICMP, etc.**
* Prometheus **does not scrape the target directly**; it scrapes the Blackbox Exporter, which then performs the probe.
* This is useful for monitoring **external websites, APIs, or any endpoint that doesn’t run an exporter**.

**8.2. How It Works**

1. Blackbox Exporter runs on its own port (default: 9115).
2. Wr configure modules in blackbox.yml to define **how to probe** (e.g., HTTP 2xx, TCP, ICMP).
3. Prometheus scrapes the Blackbox Exporter’s /probe endpoint and passes the target as a parameter

**Key Points to Remember**

1. **\_\_param\_target** → tells Blackbox Exporter **which URL to probe**.
2. **replacement in relabeling** → the **IP and port of Blackbox Exporter**, not the endpoint being probed.
3. **Modules** → defined in blackbox.yml, e.g., http\_2xx for checking HTTP status 200.

**Step 9: Prometheus Configuration for Blackbox Exporter**

The **Blackbox Exporter** follows the **multi-target exporter pattern**, which allows Prometheus to probe multiple endpoints through a single exporter instance.

* Prometheus does not scrape the target directly; instead, it scrapes the **Blackbox Exporter’s /probe endpoint**.
* The actual target URL is passed as a **parameter** (\_\_param\_target) via **relabeling**.
* Modules defined in blackbox.yml specify the type of probe (e.g., HTTP 2xx, TCP connect, ICMP ping).

**9.1 Scrape Configuration for Blackbox Exporter**

Add the following block to prometheus.yml under scrape\_configs:

- job\_name: blackbox

metrics\_path: /probe

params:

module:

- http\_2xx

static\_configs:

- targets:

- https://github.com/panthangiEshwary/application\_monitoring

- http://54.242.101.157:8080

relabel\_configs:

- source\_labels:

- \_\_address\_\_

target\_label: \_\_param\_target

- source\_labels:

- \_\_param\_target

target\_label: instance

- target\_label: \_\_address\_\_

replacement: 98.84.141.121:9115

A computer screen with green text

AI-generated content may be incorrect.



Added the ip in above yaml code where my appn is running & in replacement section we have to add the Ip (second server) URL for black box

Now we need to install black box

**Purpose of Blackbox Exporter**

Blackbox Exporter is specifically for **probing endpoints externally**, instead of relying on an exporter installed on the target machine.

You need it when:

1. **You cannot install an exporter on the target server**
   * Example: external websites like https://github.com/... or SaaS APIs.
   * Prometheus cannot scrape them directly because there’s no /metrics endpoint there.
2. **You want to monitor availability or correctness of services**
   * Blackbox Exporter can check:
     + HTTP status codes (200, 404, etc.)
     + HTTPS certificate validity
     + Response time / latency
     + TCP port connectivity
     + ICMP ping
3. **Centralized probing**
   * One Blackbox Exporter can probe multiple endpoints, even across different networks, and Prometheus scrapes it centrally.

**How It Works**

* Prometheus scrapes the **Blackbox Exporter** (e.g., http://<blackbox-ip>:9115/probe)
* Blackbox Exporter performs the **actual probe** to the target URL you specify via \_\_param\_target.

you have an application running at http://54.242.101.157:8080:

* If you tried scraping it directly with Prometheus, there’s no /metrics endpoint → **Prometheus fails**.
* With Blackbox Exporter, Prometheus scrapes the exporter, which probes the app’s HTTP endpoint and returns metrics about availability and response time.

**In short:**

**Blackbox Exporter is used for “external monitoring” of endpoints that don’t expose Prometheus metrics directly.**

* Node Exporter → for system metrics on your servers.
* Blackbox Exporter → for endpoint availability, latency, HTTP/S checks.

Doc : https://prometheus.io/download/

A screenshot of a computer

AI-generated content may be incorrect.

Wget https://github.com/prometheus/blackbox\_exporter/releases/download/v0.27.0/blackbox\_exporter-0.27.0.linux-amd64.tar.gz

A computer screen with green text

AI-generated content may be incorrect.



It got downloaded as tar file and we need to extract it

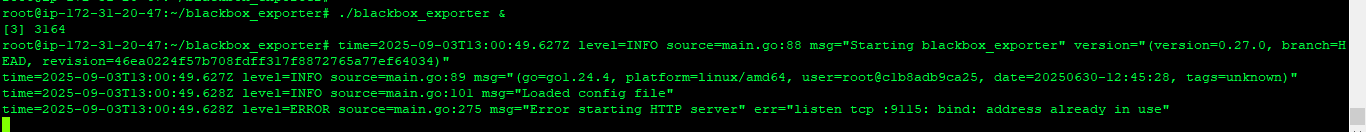
# tar -xvf blackbox\_exporter-0.27.0.linux-amd64.tar.gz

#mv blackbox\_exporter-0.27.0.linux-amd64/ blackbox\_exporter

Now it shows as below

A black screen with green and blue text

AI-generated content may be incorrect.

Now i am able to run blackbox\_exporter its eunning on port 9115  
  


<http://98.84.141.121:9115/>

now I could see my blackbox exporter

A screenshot of a computer

AI-generated content may be incorrect.

Now I run promethus as well

A screenshot of a computer

AI-generated content may be incorrect.

I could see below all **Endpoint’s are in UP**

A screenshot of a computer

AI-generated content may be incorrect.

Since Prometheus is now successfully scraping metrics via exporters, the next step is to **visualize and analyze those metrics using Grafana**. Here’s a clear, step-by-step approach:

DOC used : <https://grafana.com/grafana/download>

A screenshot of a computer

AI-generated content may be incorrect.

**Install Grafana**

sudo apt-get install -y adduser libfontconfig1 musl  
wget <https://dl.grafana.com/grafana-enterprise/release/12.1.1/grafana-enterprise_12.1.1_16903967602_linux_amd64.deb>  
sudo dpkg -i grafana-enterprise\_12.1.1\_16903967602\_linux\_amd64.deb

A computer screen with green text

AI-generated content may be incorrect.

A computer screen with green text

AI-generated content may be incorrect.

**Start and enable Grafana:**

sudo systemctl start grafana-server # Starts Grafana

sudo systemctl enable grafana-server # Makes it start on system boot

sudo systemctl status grafana-server # Check if it's running

**Access Grafana UI**

* Open a browser and go to:

http://<your-server-ip>:3000

* Default credentials:
  + **Username:** admin
  + **Password:** admin
* You will be prompted to **change the password** on first login

<http://98.84.141.121:3000/> now my Grafana is running on this port

A screenshot of a computer

AI-generated content may be incorrect.

Now Add Prometheus to Grafana as Datsource

Adding **Prometheus as a data source in Grafana** is essential because **Grafana itself does not collect metrics**—it only visualizes them. Prometheus collects metrics, and Grafana reads those metrics to display them in dashboards.

A screenshot of a computer

AI-generated content may be incorrect.

**Login to Grafana UI**

http://<your-server-ip>:3000

**Go to Configuration → Data Sources → Add data source**  
 **Select Prometheus**  
 **Set the Prometheus URL**

* If Prometheus runs on the same server:
* http://localhost:9090
* Or use the Prometheus server’s IP if remote:
* http://<prometheus-server-ip>:9090

**Click “Save & Test”**

* Grafana will confirm connection with Prometheus.

Add this promethus URL: <http://98.84.141.121:9090/> below

A screenshot of a computer

AI-generated content may be incorrect.

It shows successful

A screen shot of a computer

AI-generated content may be incorrect.

Now click on Dashboard

A screenshot of a computer

AI-generated content may be incorrect.

Here we have to set up dashboards

A screenshot of a computer

AI-generated content may be incorrect.

We can create our own dashboards

I am using here the following Doc : <https://grafana.com/grafana/dashboards/1860-node-exporter-full/>

A screenshot of a computer

AI-generated content may be incorrect.

I copied the Dashboard ID : 1860

Click on import

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Now add ID and select Prometheus as above and click import

imported the **Node Exporter Full Dashboard (ID: 1860)** into Grafana. Here's how to ensure it's set up correctly and start monitoring your system metrics:

A black line in a dark background

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Second Dashboard I am creating for black box

Doc used: <https://grafana.com/grafana/dashboards/7587-prometheus-blackbox-exporter/>

A screenshot of a computer

AI-generated content may be incorrect.

I just copied this ID: 7587and did the same as above and could see 2 dashboards

A screenshot of a computer

AI-generated content may be incorrect.

Now it shows UP (see screenshot below)

A screenshot of a computer

AI-generated content may be incorrect.

**Conclusion of Application Monitoring:**

The Proof of Concept (POC) successfully demonstrated a complete, end-to-end application monitoring solution using a combination of open-source tools: **Prometheus**, **Node Exporter**, **Blackbox Exporter**, and **Grafana**

This setup effectively showcased the core capabilities of each tool:

* **Prometheus** served as the central hub, reliably scraping and storing metrics from various sources.
* **Node Exporter** provided deep insight into the host server's health and performance by exposing crucial system-level metrics like CPU, memory, and disk usage.
* **Blackbox Exporter** proved essential for monitoring the availability and latency of external endpoints, such as the application's login page and a GitHub repository, without needing an agent on the target.
* **Grafana** transformed the raw data from Prometheus into actionable, easy-to-understand visualizations through pre-built dashboards, allowing for quick analysis of the application's performance and external dependencies.

By connecting these tools, the POC established a robust monitoring pipeline that covers both internal system metrics and external service availability. This project provides a scalable and cost-effective framework for real-time operational visibility and proactive issue detection in production environments.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*THANK YOU 😊\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*