End-to-end: Set up Prometheus + Node Exporter + Alertmanager + Grafana (with alerts) on an Ubuntu EC2 instance (AWS)

Ensure your AWS Security Group allows SSH(22) from your IP and (for testing) your IP on 9090 (Prometheus), 9093 (Alertmanager), 9100 (node_exporter), and 3000 (Grafana). In production, restrict these ports to admin IPs only

1 — Update & install prerequisites

sudo apt update && sudo apt upgrade -y
sudo apt install -y wget tar vim curl jq

2 — Install Prometheus

wget
https://github.com/prometheus/prometheus/releases/download/v2.52.0/prometheus-2.52.0.linux-amd64.tar.gz
tar xvfz prometheus-2.52.0.linux-amd64.tar.gz
cd prometheus-2.52.0.linux-amd64

Create user & directories:
sudo useradd --no-create-home --shell /bin/false prometheus
sudo mkdir -p /etc/prometheus /var/lib/prometheus
sudo chown prometheus:prometheus /etc/prometheus /var/lib/prometheus

Copy binaries & consoles:
sudo cp prometheus promtool /usr/local/bin/
sudo chown prometheus:prometheus /usr/local/bin/prometheus /usr/local/bin/promtool
sudo cp -r consoles console_libraries /etc/prometheus/
sudo chown -R prometheus:prometheus /etc/prometheus

Create Prometheus config **`vi /etc/prometheus/prometheus.yml`**

```
global:
  scrape_interval: 15s

rule_files:
  - "/etc/prometheus/alert.rules.yml"

scrape_configs:
```

```
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']
```

Save file and chown: - changing ownership
sudo chown prometheus:prometheus /etc/prometheus/prometheus.yml

Create systemd service **vi /etc/systemd/system/prometheus.service**

```
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
ExecStart=/usr/local/bin/prometheus \
  --config.file=/etc/prometheus/prometheus.yml \
  --storage.tsdb.path=/var/lib/prometheus
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Start Prometheus:
sudo systemctl daemon-reload
sudo systemctl start prometheus
sudo systemctl enable prometheus
sudo systemctl status prometheus

Verify: open http://<EC2-IP>:9090 → Targets should show Prometheus UP. (If not, check journalctl -u prometheus -f)

3 — Install Node Exporter (collects Linux metrics)

```
wget
https://github.com/prometheus/node_exporter/releases/download/v1
.6.1/node_exporter-1.6.1.linux-amd64.tar.gz
tar xvf node_exporter-1.6.1.linux-amd64.tar.gz
sudo cp node_exporter-1.6.1.linux-amd64/node_exporter
/usr/local/bin/
```

Create user & systemd unit **vi /etc/systemd/system/node_exporter.service**:

```
[Unit]
Description=Prometheus Node Exporter
After=network.target

[Service]
User=root
ExecStart=/usr/local/bin/node_exporter
Restart=always

[Install]
WantedBy=default.target
```

```
Start it
sudo systemctl daemon-reload
sudo systemctl start node_exporter
sudo systemctl enable node_exporter
```

Verify: curl http://localhost:9100/metrics | head — you should see node_cpu_seconds_total{...} lines.

Prometheus will scrape node_exporter automatically per prometheus.yml.

4 — Create sample alert rules (CPU & Memory)

Create vi /etc/prometheus/alert.rules.yml with these rules:

```
groups:
- name: node_alerts
  rules:

  - alert: HighCPUUsage
    expr: 100 - (avg by(instance)
(irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100) > 80
    for: 2m
    labels:
      severity: warning
      team: infra
    annotations:
      summary: "High CPU usage on {{ $labels.instance }}"
      description: "CPU usage > 80% for 2 minutes on {{
$labels.instance }}."

  - alert: HighMemoryUsage
    expr: 100 * (1 - (node_memory_MemAvailable_bytes /
node_memory_MemTotal_bytes)) > 80
    for: 5m
    labels:
      severity: warning
      team: infra
    annotations:
      summary: "High Memory usage on {{ $labels.instance }}"
      description: "Memory usage > 80% for 5 minutes on {{
$labels.instance }}."
```

```
Reload Prometheus:
sudo systemctl restart prometheus
# confirm rules loaded:
curl -s http://localhost:9090/api/v1/rules | jq .
```

5 — Install Alertmanager (handles notifications)

```
wget
https://github.com/prometheus/alertmanager/releases/download/v0.
25.0/alertmanager-0.25.0.linux-amd64.tar.gz
tar xvf alertmanager-0.25.0.linux-amd64.tar.gz
sudo cp alertmanager-0.25.0.linux-amd64/alertmanager
/usr/local/bin/
sudo cp alertmanager-0.25.0.linux-amd64/amtool /usr/local/bin/
sudo mkdir -p /etc/alertmanager
sudo cp alertmanager-0.25.0.linux-amd64/alertmanager.yml
/etc/alertmanager/

Edit /etc/alertmanager/alertmanager.yml — Slack webhook example
(replace with your webhook):

global:
  resolve_timeout: 5m
  slack_api_url:
'https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXX
XXXXXX'

route:
  receiver: 'slack'
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 3h

receivers:
- name: 'slack'
```

```
  slack_configs:
  - channel: '#alerts'
    send_resolved: true
    text: |
      *Alert:* {{ .CommonAnnotations.summary }}
      *Instance:* {{ range .Alerts }}{{ .Labels.instance }}{{
end }}
      *Details:* {{ .CommonAnnotations.description }}

Check syntax (always do this before restart):

amtool check-config /etc/alertmanager/alertmanager.yml

If it says Checking ... SUCCESS, you’re good.

Create systemd unit /etc/systemd/system/alertmanager.service:

[Unit]
Description=Alertmanager
After=network.target

[Service]
User=prometheus
ExecStart=/usr/local/bin/alertmanager
--config.file=/etc/alertmanager/alertmanager.yml
Restart=always

[Install]
WantedBy=multi-user.target

sudo systemctl daemon-reload
sudo systemctl start alertmanager
sudo systemctl enable alertmanager
sudo systemctl status alertmanager
```

Verify Alertmanager UI: http://<EC2-IP>:9093

6 — Wire Prometheus → Alertmanager

Ensure /etc/prometheus/prometheus.yml contains:

```
alerting:
  alertmanagers:
    - static_configs:
        - targets: ['localhost:9093']

rule_files:
  - "/etc/prometheus/alert.rules.yml"
```

```
sudo systemctl restart prometheus
```

7 — Install Grafana (official repo, Ubuntu)

```
# add repo & key
sudo apt-get install -y software-properties-common wget
apt-transport-https
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key
add -
echo "deb https://packages.grafana.com/oss/deb stable main" |
sudo tee /etc/apt/sources.list.d/grafana.list
sudo apt update
sudo apt install -y grafana

# start grafana
sudo systemctl start grafana-server
sudo systemctl enable grafana-server
```

Open Grafana: http://<EC2-IP>:3000 (default admin/admin — change password).

8 — Add Prometheus as Grafana data source

In Grafana UI: Configuration → Data sources → Add data source → Prometheus

URL: http://localhost:9090 → Save & Test (should say Data source is working).

9 — Create dashboard panel & attach alert (Grafana Unified Alerting, step-by-step)

A. Create panel with CPU metric

Grafana: Create → Dashboard → Add new panel.

In Query (Metrics) tab here select code , select Prometheus datasource, enter:

```
100 - (avg by(instance)
(irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)
```

Verify graph shows data. Set panel title CPU Usage.

Create Alert Rule (from panel)

1. In panel editor: click Alert → Create Alert Rule.
2. Name: High CPU Usage.
3. Condition: choose Query A, set:
    - WHEN avg() OF query(A, 5m, now) IS ABOVE 80
    - Evaluate every: 1m

4. Pending/for: set 2m (means must be above threshold for 2 minutes to fire).

5. Labels: severity=warning team=infra.

6. No data/error behavior: choose Alerting or as desired.

7. Notifications: choose Contact point (Slack/email). If you used Alertmanager for notifications, you can skip Grafana notification and rely on Prometheus→Alertmanager; otherwise configure Grafana contact point (Grafana Alerting -> Contact points -> add Slack webhook).

Save panel & dashboard.

Memory alert

```
100 * (1 - (node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes))
```

Repeat the same above procedure

11 — Testing alerts

Cpu test

```
sudo apt install -y stress
# spin 2 CPU workers for 180s
stress --cpu 2 --timeout 180
```

Memory test

```
sudo apt install -y stress-ng
# allocate memory (use with caution)
stress-ng --vm 1 --vm-bytes 80% --vm-keep --timeout 180s
```

After the pending time, alerts should fire:

- Prometheus: curl -s http://localhost:9090/api/v1/alerts | jq .
- Alertmanager UI: http://<EC2-IP>:9093
- Grafana Alerts page: Alerting → Alert Rules (shows firing)
- Slack/email should receive notifications (if configured)

Trouble shooting

If the alert manager fails to start

Check for the errors with the below command

sudo journalctl -u alertmanager -b --no-pager -n 200

If the error is related to "Unable to create data directory" err="mkdir data/: permission denied" — Alertmanager is trying to create a data/ folder (its default storage path) but the service user does not have permission where it’s trying to create it.

Fix

sudo systemctl stop alertmanager

Create a dedicated data directory (standard location: /var/lib/alertmanager):

sudo mkdir -p /var/lib/alertmanager

Make sure the directory is owned by the user that runs alertmanager.

Check your systemd unit to see the User= value (likely prometheus from earlier steps). Run:

```
cat /etc/systemd/system/alertmanager.service
```

If the unit contains User=prometheus use this next line; otherwise replace prometheus with the user appearing in the unit (or root if no User is set).

```
sudo chown -R prometheus:prometheus /var/lib/alertmanager

sudo chmod 750 /var/lib/alertmanager
```

Edit the systemd unit to tell Alertmanager to use that directory (adds explicit storage path). Open the unit:

```
sudo vim /etc/systemd/system/alertmanager.service
```

Modify the ExecStart line so it includes --storage.path=/var/lib/alertmanager. Example full unit (paste/replace contents):

```
[Unit]

Description=Alertmanager

After=network.target


[Service]
```

```
User=prometheus

ExecStart=/usr/local/bin/alertmanager
--config.file=/etc/alertmanager/alertmanager.yml
--storage.path=/var/lib/alertmanager

Restart=on-failure


[Install]

WantedBy=multi-user.target


sudo systemctl daemon-reload

sudo systemctl start alertmanager

sudo systemctl enable alertmanager


sudo systemctl status alertmanager -l

sudo journalctl -u alertmanager -n 200 --no-pager


You should no longer see the permission denied error and
Alertmanager should enter active (running).
```

Slack integration

create a channel - select channel - add apps - apps - incoming webhooks - open app-configuration- add to slack - select slack channels - add incoming webhook integration - here u can see url - save settings - you can see notification