DevOps Monitoring Challenge

Submission by Chris Spannos

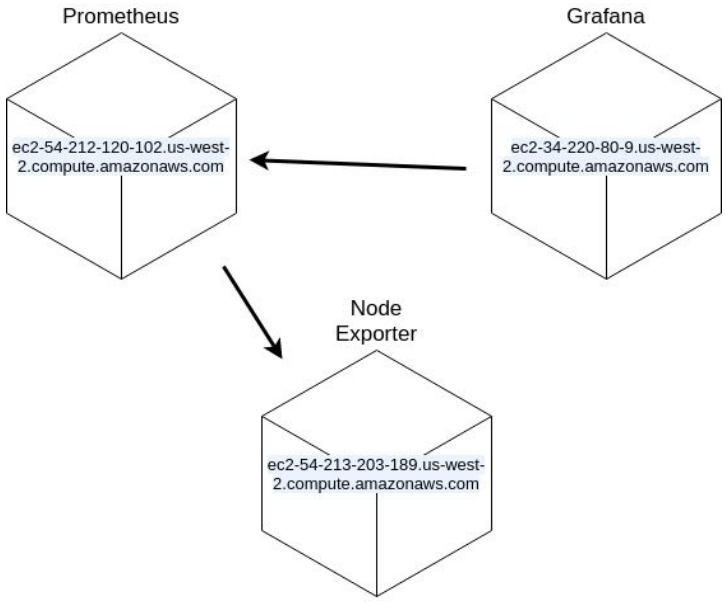
My submission includes solutions for:

- ➔ Setting up 3 t2.micro AWS EC2 instances
- ➔ Configuring Prometheus to monitor 2 EC2 instances
- ➔ Connecting Prometheus to Grafana to collect and display:
  - ◆ cpu
  - ◆ memory
  - ◆ disk space
  - ◆ high cpu
  - ◆ high memory
  - ◆ high disk usage
  - ◆ network traffic rate
  - ◆ upload rate
  - ◆ download rate

1. Set up 3 t2.micro AWS EC2 instances

First, I set up the three EC2 Linux instances: one each for Prometheus, Grafana and for a node exporter.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
Grafana	i-0172c5aed909b5807	t2.micro	us-west-2a	running	2/2 checks ...	None	ec2-34-220-80-9.us-west-2.compute.amazonaws.com	34.220.80.9
Prometheus	i-0b5c042a8d26e98ba	t2.micro	us-west-2a	running	2/2 checks ...	None	ec2-54-212-120-102.us-west-2.compute.amazonaws.com	54.212.120.102
Server (node exporter)	i-01344df8aea158521	t2.micro	us-west-2a	running	2/2 checks ...	None	ec2-54-213-203-189.us-west-2.compute.amazonaws.com	54.213.203.189



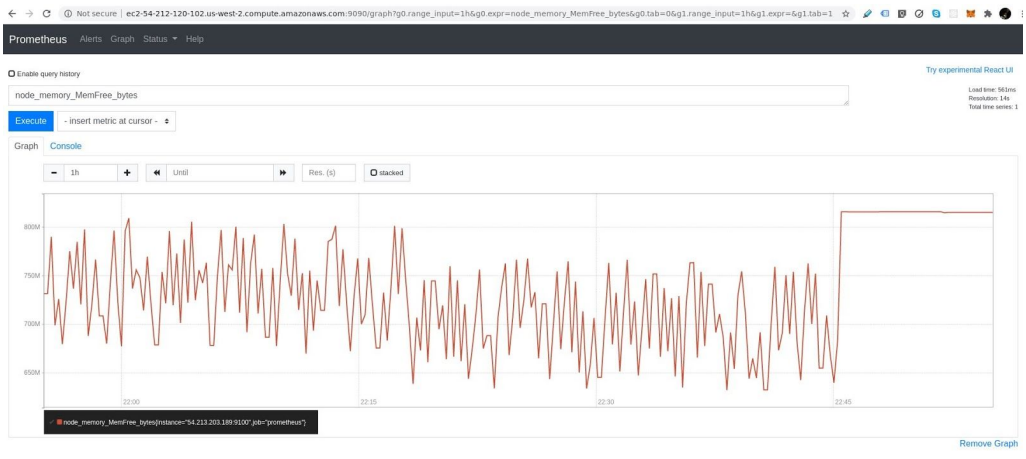
Then I set up the security group to configure ports for services on each instance:

Inbound rules

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	-
HTTP	TCP	80	::/0	-
SSH	TCP	22	0.0.0.0/0	-
Custom TCP	TCP	9090	0.0.0.0/0	prometheus
Custom TCP	TCP	3000	0.0.0.0/0	grafana
HTTPS	TCP	443	0.0.0.0/0	-
Custom TCP	TCP	9100	0.0.0.0/0	server / node exporter

2. Configure Prometheus to monitor 2 EC2 instances

I configured Prometheus to monitor 2 instances. The first instance was so Prometheus could monitor itself. The second instance was to monitor a node exporter configured on a separate instance:



3. Connect Prometheus to Grafana

I connected Prometheus to Grafana to collect and display cpu, memory, and disk space usage. I additionally simulated high cpu, high memory, and high disk usage on monitored instances (see screenshot). I ran the high cpu and memory simulations by installing stress-ng on the node exporter instance. I simulated high disk usage by using

the ‘fallocate’ command to create a file using half the disk space (4G of 8G). I also collected and displayed the network traffic rate, upload rate, and download rate: