

DevOps Monitoring Challenge

Edison Chukwuemeka

Setup EC2 Instances

- Log into AWS console: aws.amazon.com
- Select the EC2 service as shown below.

The screenshot shows the AWS Management Console homepage. In the 'Recently Selected services' section, the 'EC2' icon is highlighted with a red circle. Other services like Billing and S3 are also listed.

- Select the Amazon Linux 2 AMI SSD Volume machine.

The screenshot shows the 'Step 1: Choose an Amazon Machine Image (AMI)' screen. It lists two AMIs: 'Amazon Linux 2 AMI (HVM, SSD Volume Type)' and 'Red Hat Enterprise Linux 8 (HVM, SSD Volume Type)'. The first option is selected and highlighted with a red circle. A modal window shows the details of the selected AMI, including its ID and architecture options (64-bit x86 and 64-bit Arm). The '64-bit (x86)' option is selected and highlighted with a red circle.

- Select t2.micro instance and click next.

The screenshot shows the 'Step 2: Choose an Instance Type' screen. The 't2.micro' instance is selected and highlighted with a red circle. At the bottom right, the 'Next: Configure Instance Details' button is circled in red.

- Configure the EC2 Instance.

The screenshot shows the 'Step 3: Configure Instance Details' screen. It includes fields for 'Number of instances' (set to 1), 'Purchasing option' (Request Spot instances), 'Network' (vpc-5003b03b), 'Placement group' (No instance placement group), 'Domain join directory' (None), and 'CPU options' (Specify CPU options). The 'IAM role' field is highlighted with a red circle. At the bottom right, the 'Next: Add Storage' button is circled in red.

Setup EC2 Instances

➤ Add Storage

Step 4: Add Storage

You can attach up to 20 volumes using the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options on Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (Mbps)	Delete on Termination	Encryption
Root	/dev/xvda	snap-05741302b4fa30b4d	8	General Purpose SSD (gp2)	100 / 3000	N/A	Yes	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 20 GiB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

Cancel Previous Review and Launch Next: Add Tags

➤ Add Name tag to the EC2 Instance

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes. Learn more about tagging your Amazon EC2 resources.

Key	Value
Name	i-01234567890abcdef0

Add another tag (optional)

Cancel Previous Review and Launch Next: Configure Security Group

➤ Configure the Security group

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: Create a new security group Select an existing security group

Security group name: client-2-group

Description: launch-wizard-2 created 2020-11-02T17:46:56.387-00:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Anywhere	SSH Access for Admin
Custom TCP	TCP	9090	Anywhere	HTTP Access for Prometheus
Custom TCP	TCP	9100	Anywhere	HTTP Access for Node_Exporter
Custom TCP	TCP	3000	Anywhere	HTTP Access for Grafana

Add Rule

Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Previous Review and Launch

Specifies which port connects to EC2 instance

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click Launch to assign a key pair to your instance and complete the launch process.

► Improve your instances' security: Your security group, client-2-group, is open to the world.
Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only. You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups

AMIs Details

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-03657b056516ab7912
Free tier eligible
Root Device Type: sda1 Virtualization type: hvm

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GiB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

Security Groups

Security group name: client-2-group

Cancel Previous Launch

AMI Details

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-03657b056516ab7912
Free tier eligible
Root Device Type: sda1 Virtualization type: hvm

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GiB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

Security Groups

Security group name: client-2-group

Cancel Previous Launch

Setup EC2 Instances

The screenshot shows the AWS EC2 Instances page. At the top, there is a green banner indicating "Successfully stopped i-00c29f807254305b3". The main table displays three instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm Status	Availability zone	Public IPv4 DNS
prometheus	i-00fd61efe18bc8a2c	Stopped	t2.micro	-	No alarms +	us-east-2c	-
client-1	i-0fb9b77348985f1d6	Stopped	t2.micro	-	No alarms +	us-east-2a	-
Client-2	i-00c29f807254305b3	Stopped	t2.micro	-	No alarms +	us-east-2c	-

The instance "Client-2" is selected, indicated by a checked checkbox in the first column. Below the table, the details for "Instance: i-00c29f807254305b3 (Client-2)" are shown, with the "Details" tab selected. The page includes a sidebar with navigation links for EC2 services like Instances, Instance Types, and Launch Templates.

Configure Prometheus to monitor 2 EC2 instances

Configure Prometheus to monitor client-2:

- ssh to client-2
- Update the server:

```
$ sudo yum update -y
```

```
MobaXterm Personal Edition v20.3
(X server, SSH client and network tools)

Your computer drives are accessible through the /drives path
Your DISPLAY is set to 167.98.111.240:0.0
When using SSH, your remote DISPLAY is automatically forwarded
Each command status is specified by a special symbol (✓ or ✘)

Important:
This is MobaXterm Personal Edition. The Professional edition
allows you to customize MobaXterm for your company: you can add
your own logo, your parameters, your welcome message and generate
either an MSI installation package or a portable executable.
We can also modify MobaXterm or develop the plugins you need.
For more information: https://mobaxterm.mobatek.net/download.html

02/11/2028 0 12:23:47 ➤ /home/mobaxterm  ssh ec2-user@134.104.165 -i /drives/c/Users/Wunmi/Downloads/AWS/EC2-InstanceKey.pem
Warning: Permanently added '134.104.165' (RSA) to the list of known hosts.
X11 forwarding request failed on channel 0

[ec2-user@ip-172-31-40-127 ~]$ sudo yum update -y

[ec2-user@ip-172-31-40-127 ~]$ curl -LO
https://github.com/prometheus/prometheus/releases/download/v2.22.0/prometheus-2.22.0.linux-amd64.tar.gz
[ec2-user@ip-172-31-40-127 ~]$ sha256sum prometheus-2.22.0.linux-amd64.tar.gz
[ec2-user@ip-172-31-40-127 ~]$ tar xvf prometheus-2.22.0.linux-amd64.tar.gz
[ec2-user@ip-172-31-40-127 ~]$ sudo cp prometheus-files/prometheus /usr/local/bin/
[ec2-user@ip-172-31-40-127 ~]$ sudo cp prometheus-files/promtool usr/local/bin/
[ec2-user@ip-172-31-40-127 ~]$ curl -LO
https://github.com/prometheus/node_exporter/releases/download/v1.0.1/node_exporter-1.0.1.linux-amd64.tar.gz
[ec2-user@ip-172-31-40-127 ~]$ sha256sum node_exporter-1.0.1.linux-amd64.tar.gz
[ec2-user@ip-172-31-40-127 ~]$ tar xvf node_exporter-1.0.1.linux-amd64.tar.gz
[ec2-user@ip-172-31-40-127 ~]$ sudo cp node_exporter-1.0.1.linux-amd64/node_exporter /usr/local/bin/
[ec2-user@ip-172-31-40-127 ~]$ sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter
```

- Create a user - node_exporter:

```
$ sudo useradd --no-create-home --shell /bin/false node_exporter
```

- Create a user - prometheus:

```
$ sudo useradd --no-create-home --shell /bin/false Prometheus
$ sudo mkdir /etc/Prometheus
$ sudo mkdir /var/lib/Prometheus
$ sudo chown prometheus:prometheus /etc/Prometheus
$ sudo chown prometheus:prometheus /var/lib/Prometheus
```

- Download prometheus:

```
$ curl -LO
```

```
https://github.com/prometheus/prometheus/releases/download/v2.22.0/prometheus-2.22.0.linux-amd64.tar.gz
```

- Validate the downloaded file:

```
$ sha256sum prometheus-2.22.0.linux-amd64.tar.gz
```

- Unpack the downloaded file:

```
$ tar xvf prometheus-2.22.0.linux-amd64.tar.gz
```

- Copy the binary file to /usr/local/bin and set the user and group ownership to Prometheus user:

```
$ sudo cp prometheus-files/prometheus /usr/local/bin/
```

```
$ sudo cp prometheus-files/promtool usr/local/bin/
```

- Download node_exporter:

```
$ curl -LO
```

```
https://github.com/prometheus/node\_exporter/releases/download/v1.0.1/node\_exporter-1.0.1.linux-amd64.tar.gz
```

- Validate the downloaded file:

```
$ sha256sum node_exporter-1.0.1.linux-amd64.tar.gz
```

- Unpack the downloaded file:

```
$ tar xvf node_exporter-1.0.1.linux-amd64.tar.gz
```

- Copy the binary file to /usr/local/bin and set the user and group ownership to node_exporter user:

```
$ sudo cp node_exporter-1.0.1.linux-amd64/node_exporter /usr/local/bin/
```

```
$ sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter
```

Configure Prometheus to monitor 2 EC2 instances

➤ Create prometheus.service file:

```
$ sudo nano /etc/systemd/system/prometheus.service
```

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

```
[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
--config.file /etc/prometheus/prometheus.yml \
--storage.tsdb.path /var/lib/prometheus/ \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries
```

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

➤ Create node_exporter.service file:

```
$ sudo nano /etc/systemd/system/node_exporter.service
```

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

```
[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter
```

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

Configure Prometheus to monitor 2 EC2 instances

- Configure Prometheus to scrape data from node_exporter on the EC2 instances.

```
global:  
  scrape_interval: 15s  
  
scrape_configs:  
  - job_name: 'prometheus'  
    scrape_interval: 5s  
    static_configs:  
      - targets: ['localhost:9100']  
  - job_name: 'client-1'  
    scrape_interval: 5s  
    static_configs:  
      - targets: ['172.31.15.4:9100']  
  - job_name: 'client-2'  
    scrape_interval: 5s  
    static_configs:  
      - targets: ['172.31.40.127:9100']
```

- Reload systemd:

```
$ sudo systemctl daemon-reload
```

- Start Prometheus service

```
$ sudo systemctl start Prometheus
```

- Check the status of Prometheus service.

```
$ sudo systemctl status Prometheus
```

- Enable Prometheus service.

```
$ sudo systemctl enable prometheus
```

- Start the node_exporter service:

```
$ sudo systemctl start node_exporter
```

- Check the status of the service:

```
$ sudo systemctl status node_exporter
```

- Enable node_exporter:

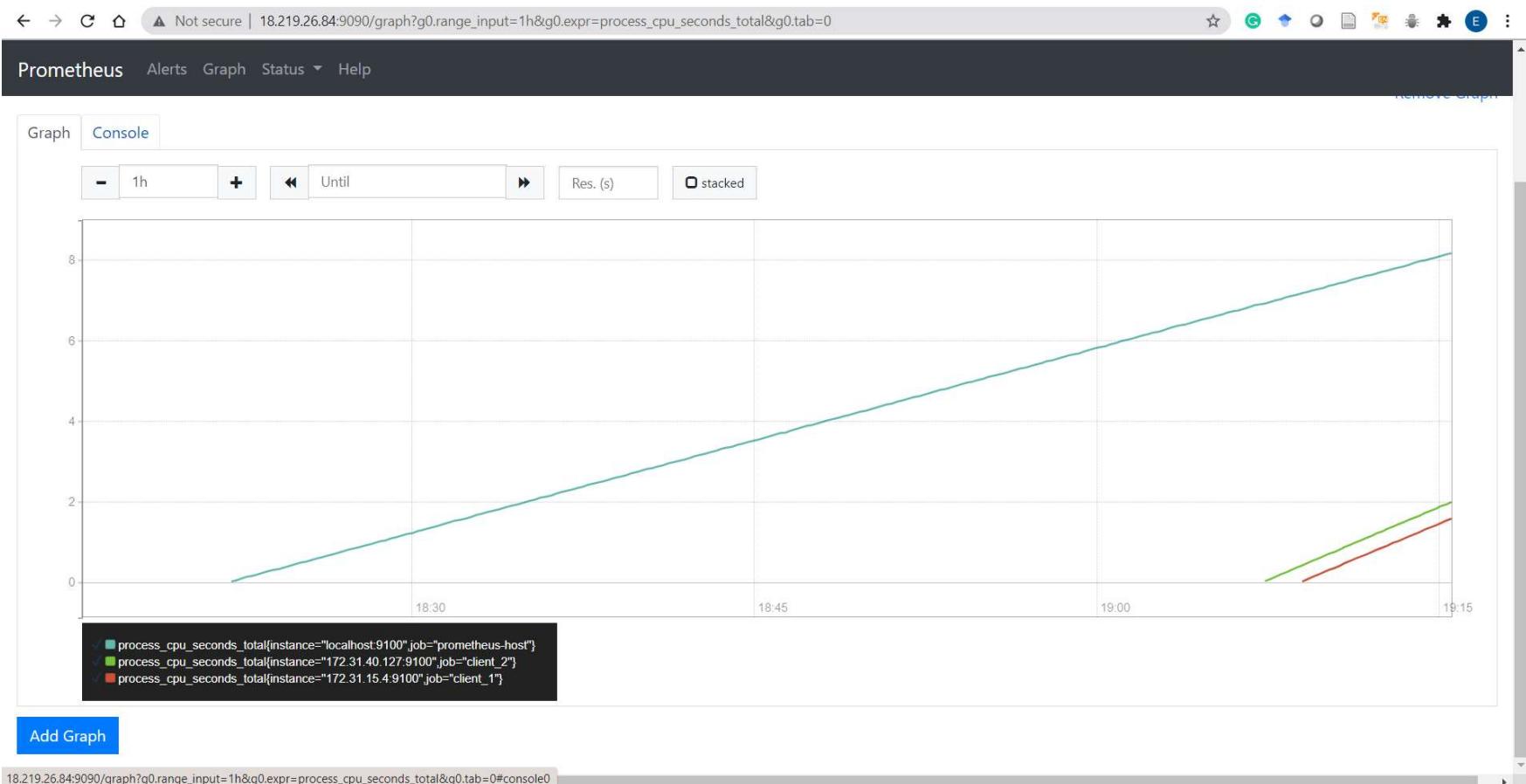
```
$ sudo systemctl enable node_exporter
```

Monitor 2 EC2 Instances through Prometheus

The screenshot shows the Prometheus Targets page with three healthy targets listed:

- client_1 (1/1 up)**: Endpoint `http://172.31.15.4:9100/metrics`, State UP, Labels `instance="172.31.15.4:9100" job="client_1"`, Last Scrape 1.594s ago, Scrape Duration 13.2ms.
- client_2 (1/1 up)**: Endpoint `http://172.31.40.127:9100/metrics`, State UP, Labels `instance="172.31.40.127:9100" job="client_2"`, Last Scrape 1.579s ago, Scrape Duration 12.5ms.
- prometheus-host (1/1 up)**: Endpoint `http://localhost:9100/metrics`, State UP, Labels `instance="localhost:9100" job="prometheus-host"`, Last Scrape 4.488s ago, Scrape Duration 12.54ms.

Monitor 2 EC2 Instances through Prometheus



Connect Prometheus to Grafana

Install Grafana on the control machine – Prometheus-host

➤ SSH to the EC2 Instance – Prometheus-host:

```
$ ssh ec2-user@18.219.26.84 -i  
/drives/c/Users/Wunmi/Downloads/AWS/EC2-InstanceKey.pem
```

➤ Update installed packages

```
$ sudo yum update -y
```

➤ Add a new yum repo for Grafana called grafana.repo

```
$ sudo nano /etc/yum.repos.d/grafana.repo  
  
[grafana]  
name=grafana  
  
baseurl=https://packages.grafana.com/oss/rpm  
repo_gpgcheck=1  
enabled=1  
gpgcheck=1  
gpgkey=https://packages.grafana.com/gpg.key  
sslverify=1  
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

➤ Install Grafana:

```
$ sudo yum install Grafana -y
```

➤ Reload system:

```
$ sudo systemctl daemon-reload
```

➤ Start Grafana Server:

```
$ sudo systemctl start grafana-server
```

➤ Check the status of Grafana Server:

```
$ sudo systemctl status grafana-server
```

➤ Enable Grafana start upon booting the EC2 instance

```
$ sudo systemctl enable grafana-server.service
```

Connect Prometheus to Grafana

Connect to Grafana Server:

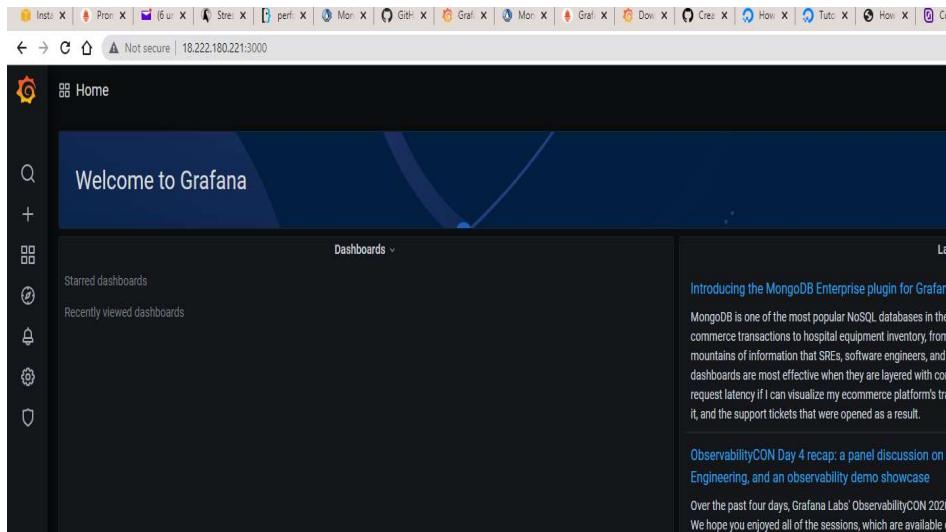
➤ Launch the EC2 instances:

The screenshot shows the AWS EC2 Instances page with the following details:

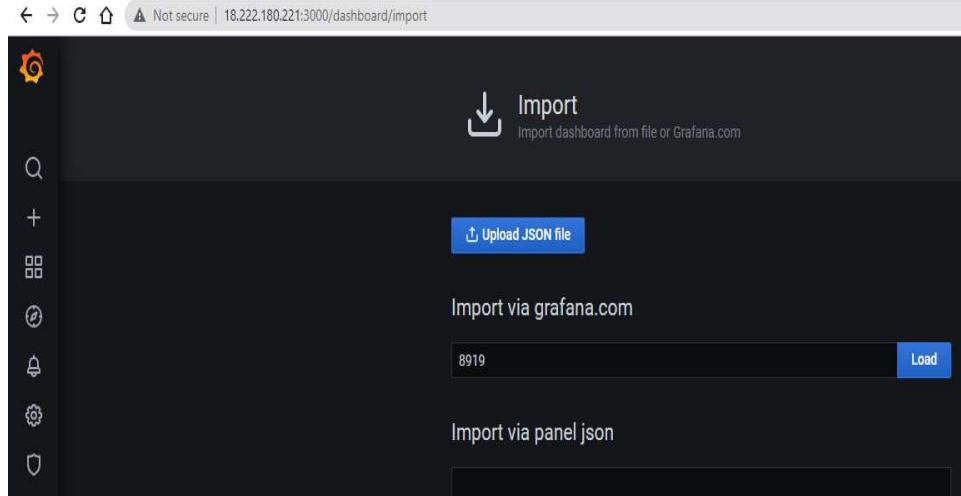
- Instances (3/3) Info**: Shows three instances:
 - prometheus**: Instance ID i-00fd61efe18bc8a2c, Running, t2.micro, No alarms, us-east-2c, Public IPv4 18.222.180.221, Elastic IP -.
 - client-1**: Instance ID i-0fb9b77348985f1d6, Running, t2.micro, No alarms, us-east-2a, Public IPv4 18.218.19.223, Elastic IP -.
 - Client-2**: Instance ID i-00c29f807254305b3, Running, t2.micro, No alarms, us-east-2c, Public IPv4 ec2-3-19-61-28.us-east-2, Elastic IP -.
- Monitoring**: Displays various metrics for the instances:
 - CPU utilization (%)
 - Status check failed (any) (count)
 - Status check failed (instance) (count)
 - Status check failed (system) (count)
 - Network in (bytes)
 - Network out (bytes)
 - Network packets in (count)
 - Network packets out (count)Each metric chart shows data for the three instances, with orange dots for client-1 and green dots for Client-2.

Connect Prometheus to Grafana

- Connect to Grafana using the public ip of the host machine with port 3000
\$ 18.222.180.221:3000

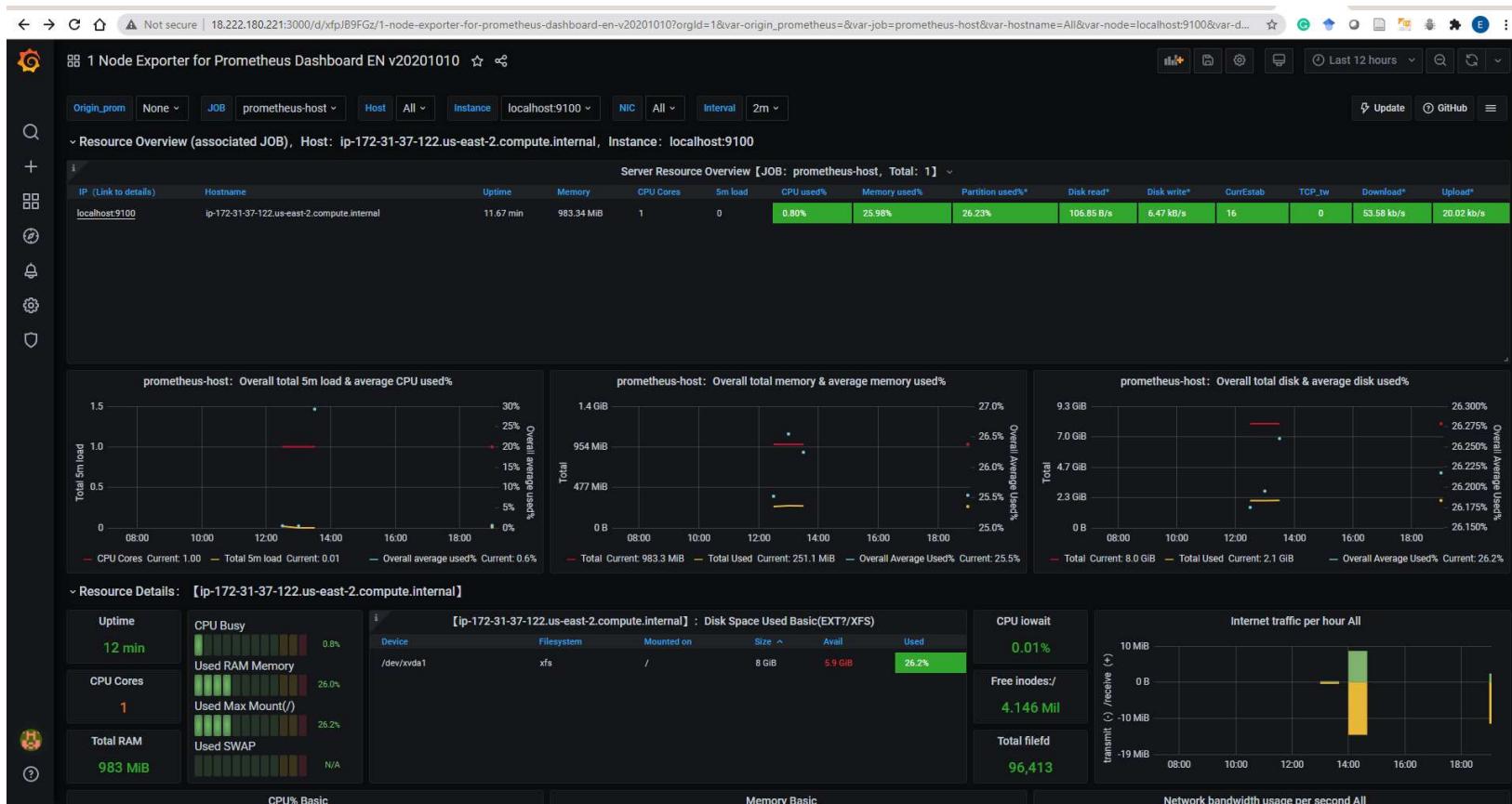


➤ Load a Dashboard



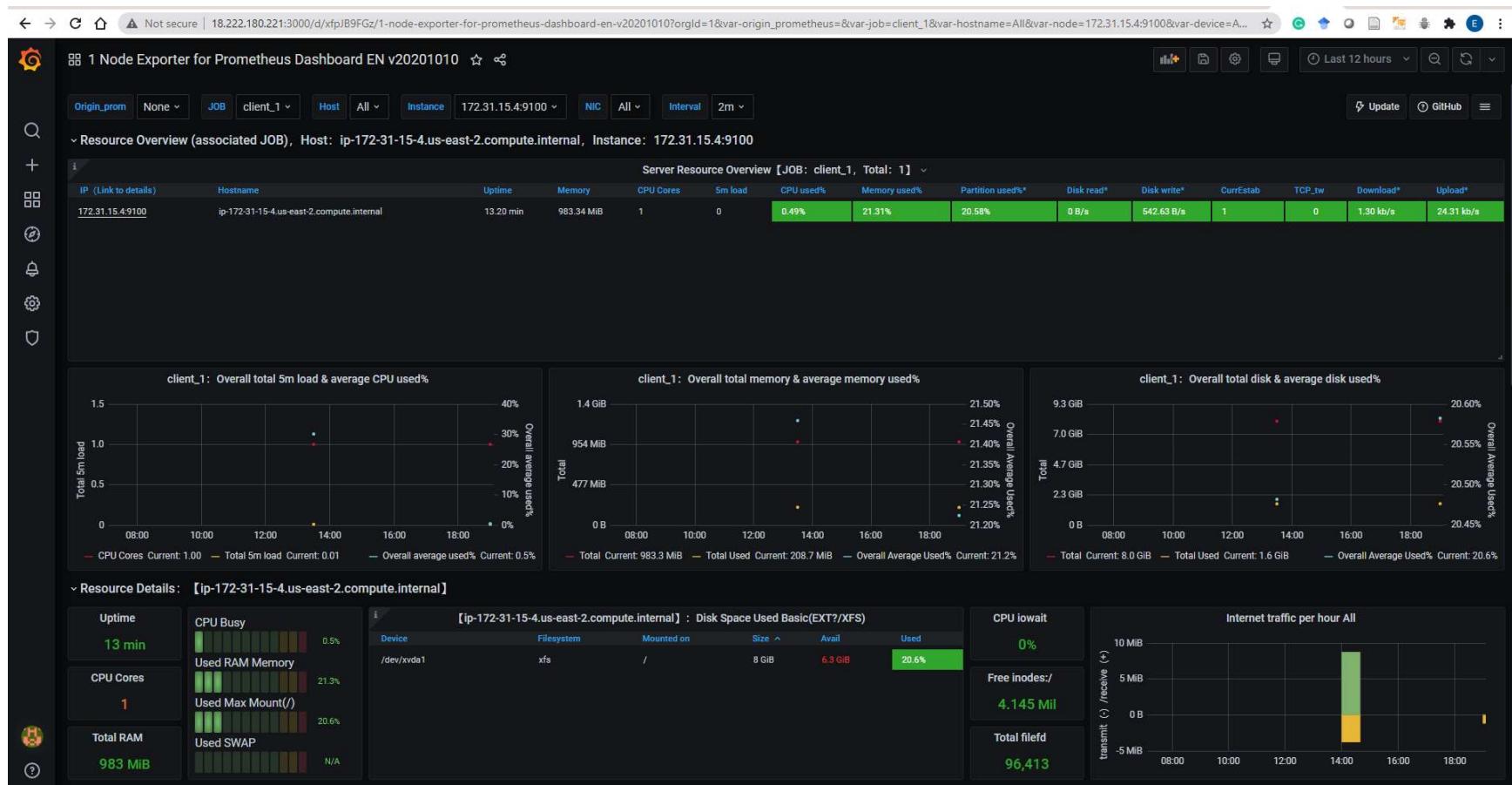
Collect and Display cpu, memory and disk usage

Prometheus-host server



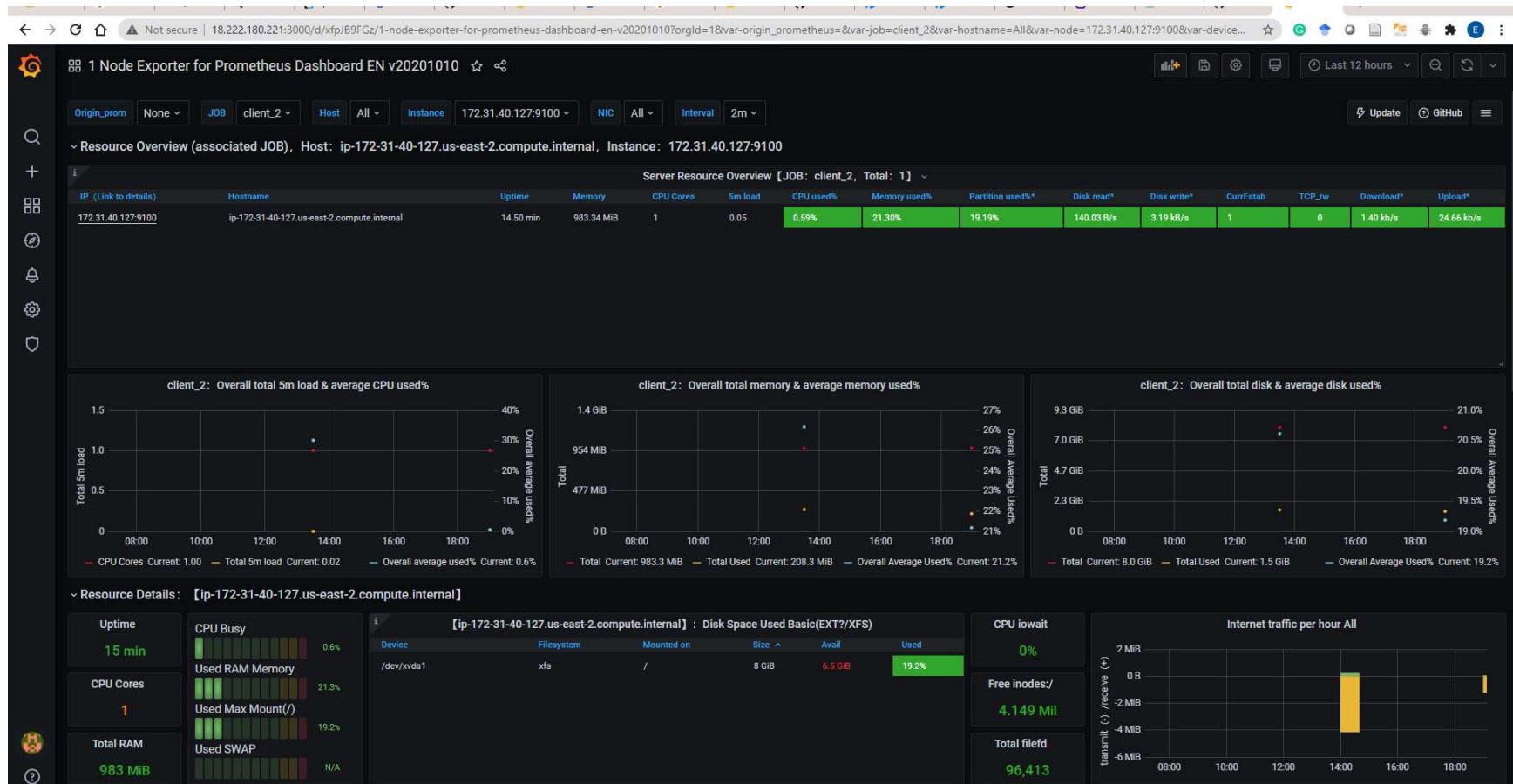
Collect and Display cpu, memory and disk usage

Client-1 server



Collect and Display cpu, memory and disk usage

Client-2 server



Simulate high cpu, memory, and disk usage on monitored instances

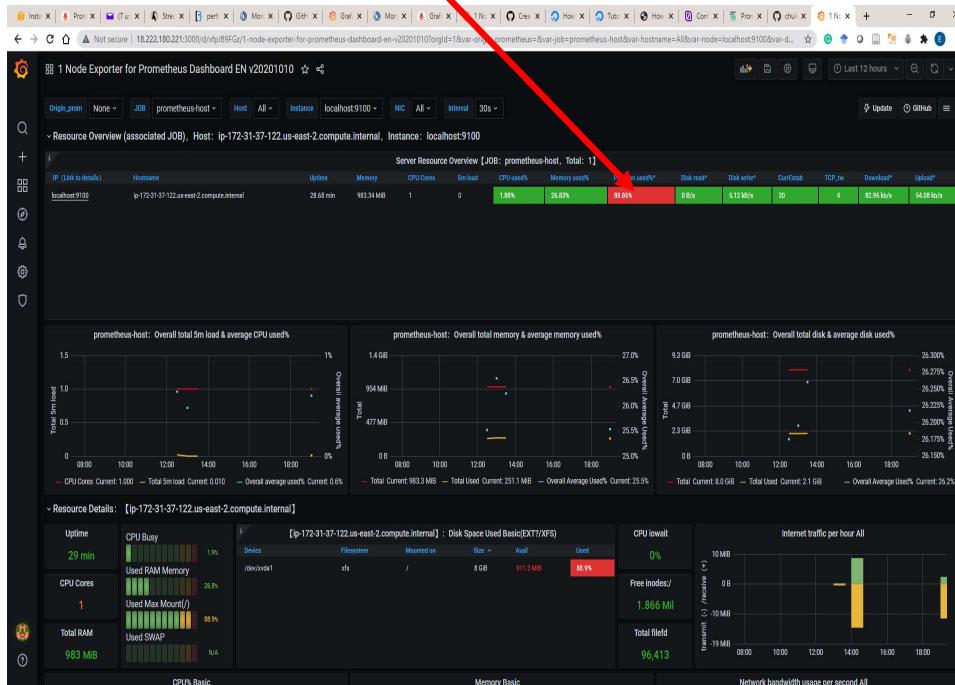
Prometheus-Host Server – Simulate Disk usage

```
[ec2-user@ip-172-31-37-122 ~]$ df -h
Filesystem  Size Used Avail Use% Mounted on
devtmpfs    474M  0 474M  0% /dev
tmpfs       492M  0 492M  0% /dev/shm
tmpfs       492M 448K 492M  1% /run
tmpfs       492M  0 492M  0% /sys/fs/cgroup
/dev/xvda1   8.0G 2.1G 5.9G 27% /      ======> Before running the test
tmpfs       99M  0 99M  0% /run/user/1000
[ec2-user@ip-172-31-37-122 ~]$ fallocate -l 5G test-disk
[ec2-user@ip-172-31-37-122 ~]$ df -h
Filesystem  Size Used Avail Use% Mounted on
devtmpfs    474M  0 474M  0% /dev
tmpfs       492M  0 492M  0% /dev/shm
tmpfs       492M 448K 492M  1% /run
tmpfs       492M  0 492M  0% /sys/fs/cgroup
/dev/xvda1   8.0G 7.1G 912M 89% /      ======> After Running the test
tmpfs       99M  0 99M  0% /run/user/1000
[ec2-user@ip-172-31-37-122 ~]$ rm test-disk
[ec2-user@ip-172-31-37-122 ~]$ df -h
Filesystem  Size Used Avail Use% Mounted on
devtmpfs    474M  0 474M  0% /dev
tmpfs       492M  0 492M  0% /dev/shm
tmpfs       492M 448K 492M  1% /run
tmpfs       492M  0 492M  0% /sys/fs/cgroup
/dev/xvda1   8.0G 2.1G 5.9G 27% /      ======> After deleting the file.
tmpfs       99M  0 99M  0% /run/user/1000
[ec2-user@ip-172-31-37-122 ~]$
```

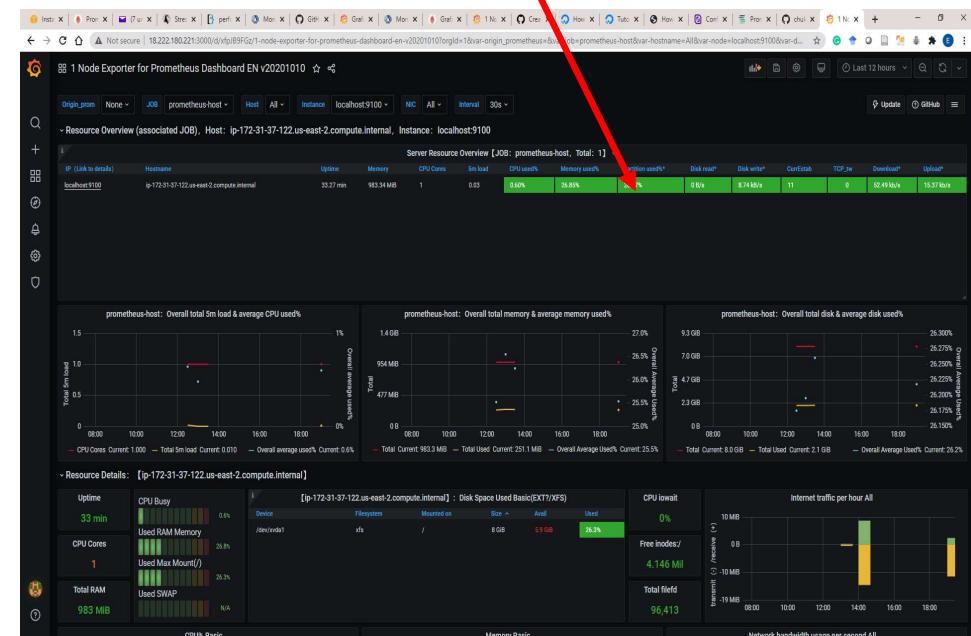
Simulate high cpu, memory, and disk usage on monitored instances

Prometheus-Host Server – Simulate Disk usage

After Allocating the memory.



After Deleting the file.

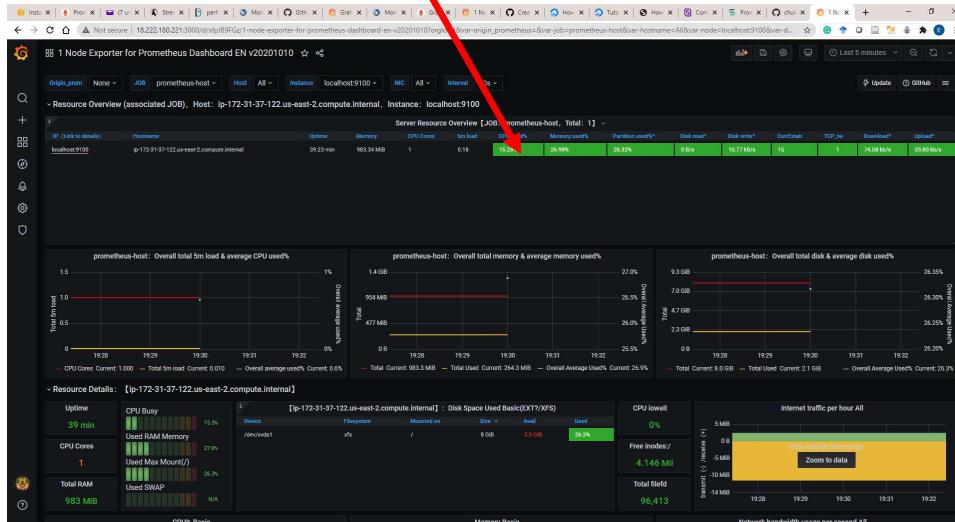


Simulate high cpu, memory, and disk usage on monitored instances

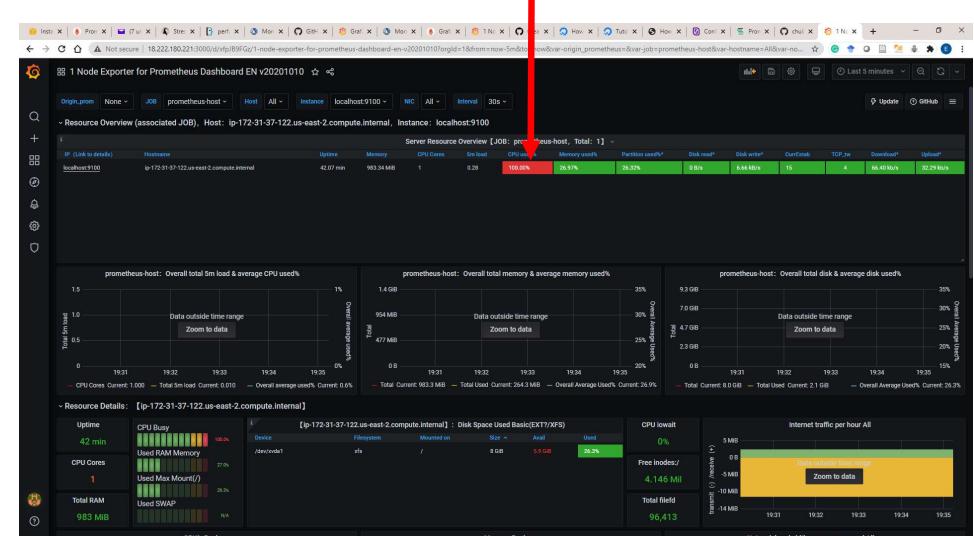
Prometheus-Host Server – Simulate cpu usage

```
[ec2-user@ip-172-31-37-122 ~]$ nproc | xargs seq | xargs -n1 -P4 md5sum /dev/zero
```

Before running the process.



Running the test.



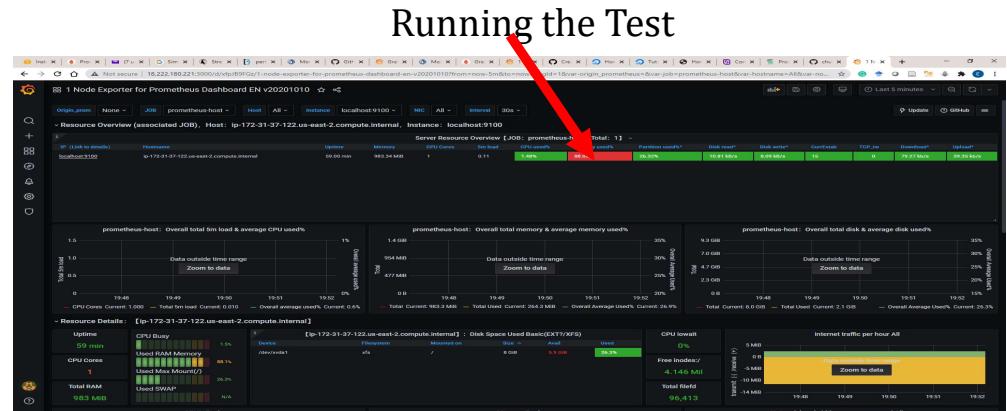
Simulate high cpu, memory, and disk usage on monitored instances

Prometheus-Host Server – Simulate Memory usage:
[ec2-user@ip-172-31-37-122 ~]\$ python
memoryLoadRunner.py 600

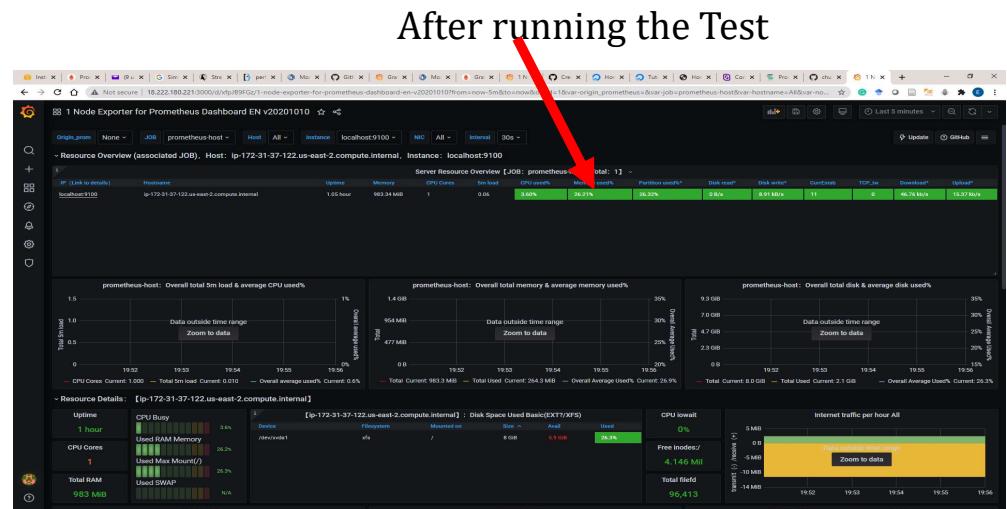
```
#!/usr/bin/env python
import sys
import time

if len(sys.argv) != 2:
    print "usage: fillmem <number-of-megabytes>"
    sys.exit()
```

```
count = int(sys.argv[1])
megabyte = (0,) * (1024 * 1024 / 8)
data = megabyte * count
while True:
    time.sleep(1)
```



Running the Test



After running the Test

Simulate high cpu, memory, and disk usage on monitored instances

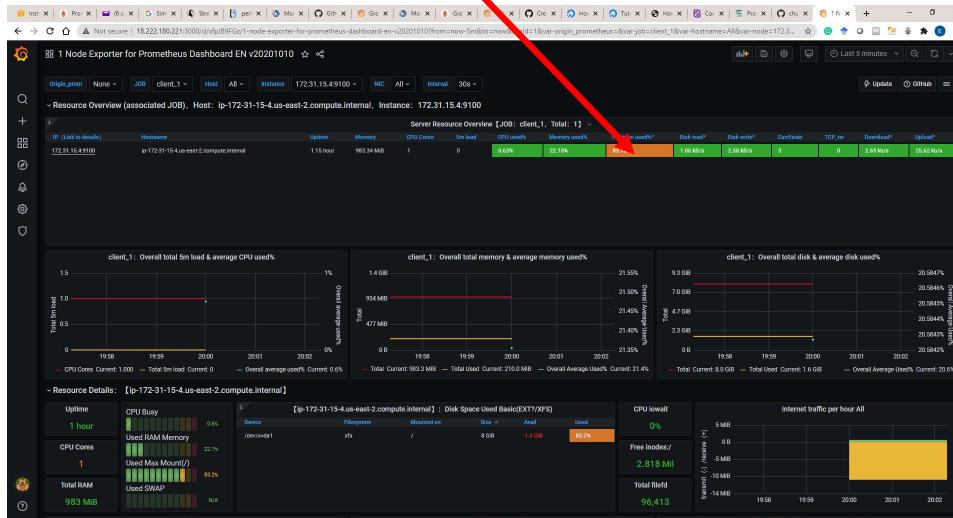
Client-1 Server – Simulate Disk usage

```
[ec2-user@ip-172-31-15-4 ~]$ df -h
Filesystem  Size Used Avail Use% Mounted on
devtmpfs    474M  0 474M  0% /dev
tmpfs       492M  0 492M  0% /dev/shm
tmpfs       492M 436K 492M  1% /run
tmpfs       492M  0 492M  0% /sys/fs/cgroup
/dev/xvda1   8.0G 1.7G 6.4G 21% /  ======> Before allocating disk space.
tmpfs       99M  0 99M  0% /run/user/1000
[ec2-user@ip-172-31-15-4 ~]$ fallocate -l 5G test-client-1_disk
[ec2-user@ip-172-31-15-4 ~]$ df -h
Filesystem  Size Used Avail Use% Mounted on
devtmpfs    474M  0 474M  0% /dev
tmpfs       492M  0 492M  0% /dev/shm
tmpfs       492M 436K 492M  1% /run
tmpfs       492M  0 492M  0% /sys/fs/cgroup
/dev/xvda1   8.0G 6.7G 1.4G 84% /  ======> After Allocating disk space
tmpfs       99M  0 99M  0% /run/user/1000
[ec2-user@ip-172-31-15-4 ~]$ rm test-client-1_disk
[ec2-user@ip-172-31-15-4 ~]$ df -h
Filesystem  Size Used Avail Use% Mounted on
devtmpfs    474M  0 474M  0% /dev
tmpfs       492M  0 492M  0% /dev/shm
tmpfs       492M 436K 492M  1% /run
tmpfs       492M  0 492M  0% /sys/fs/cgroup
/dev/xvda1   8.0G 1.7G 6.4G 21% /  ======> After deleting the allocated disk space
tmpfs       99M  0 99M  0% /run/user/1000
[ec2-user@ip-172-31-15-4 ~]$
```

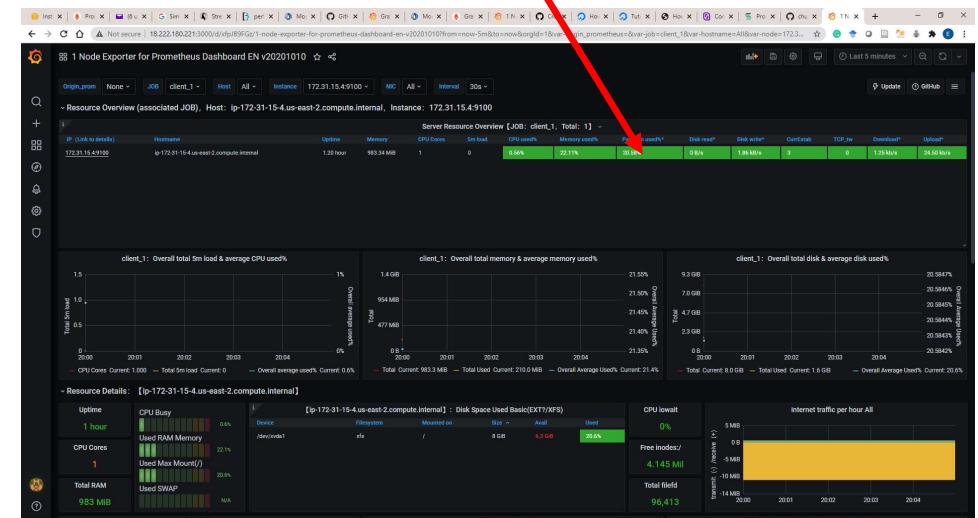
Simulate high cpu, memory, and disk usage on monitored instances

Client-1 Server – Simulate Disk usage

After Allocating the disk space.



After Deleting the allocated disk space.

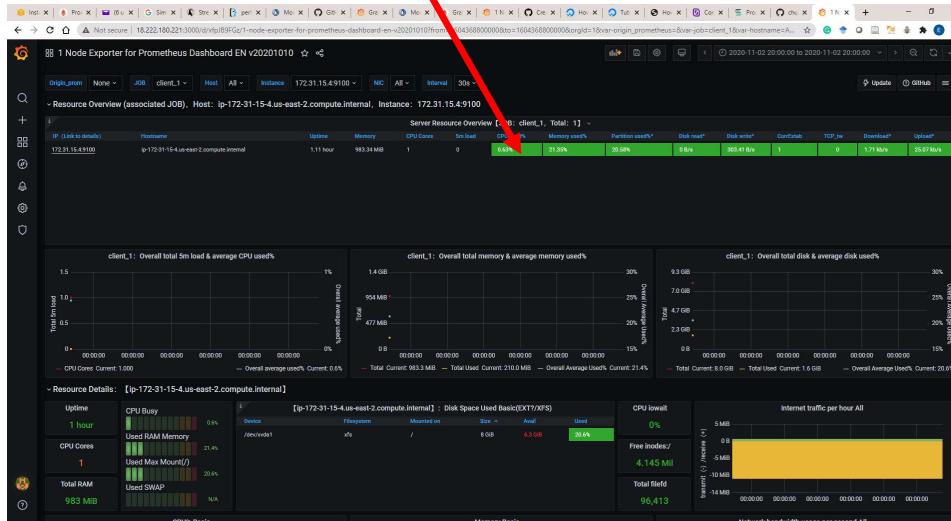


Simulate high cpu, memory, and disk usage on monitored instances

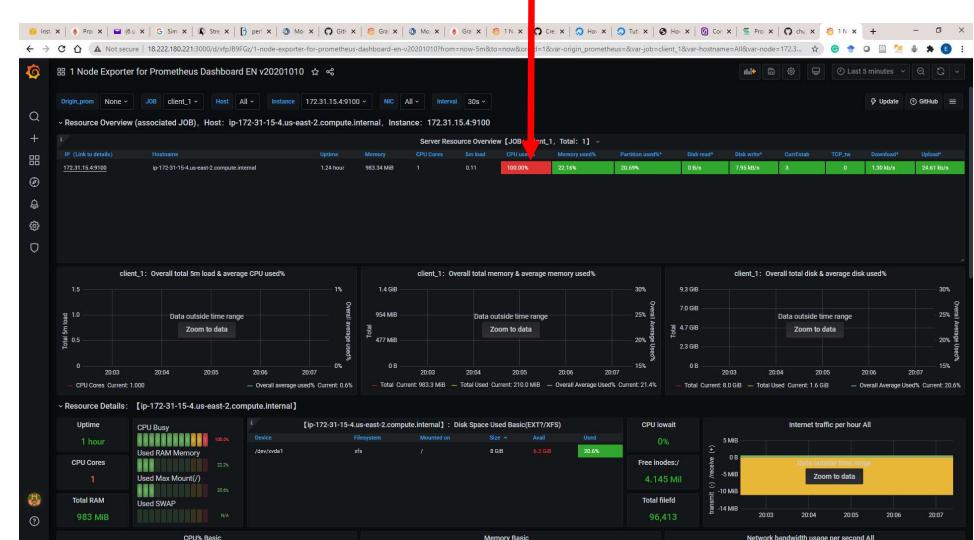
Prometheus-Host Server – Simulate cpu usage

```
[ec2-user@ip-172-31-15-4 ~]$ nproc | xargs seq | xargs -n1 -P4 md5sum /dev/zero
```

Before running the process.



Running the test.



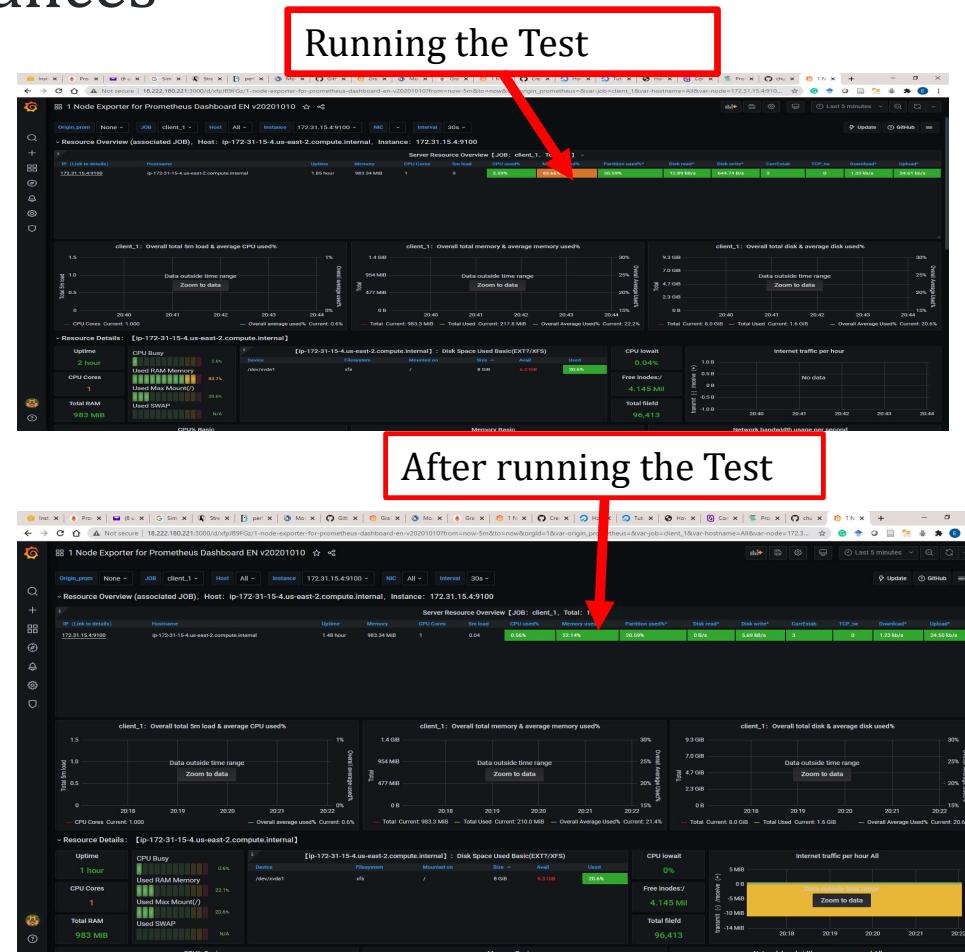
Simulate high cpu, memory, and disk usage on monitored instances

Prometheus-Host Server – Simulate Memory usage:
[ec2-user@ip-172-31-15-4 ~]\$ python
memoryLoadRunner.py 600

```
#!/usr/bin/env python
import sys
import time

if len(sys.argv) != 2:
    print "usage: fillmem <number-of-megabytes>"
    sys.exit()

count = int(sys.argv[1])
megabyte = (0,) * (1024 * 1024 / 8)
data = megabyte * count
while True:
    time.sleep(1)
```



Simulate high cpu, memory, and disk usage on monitored instances

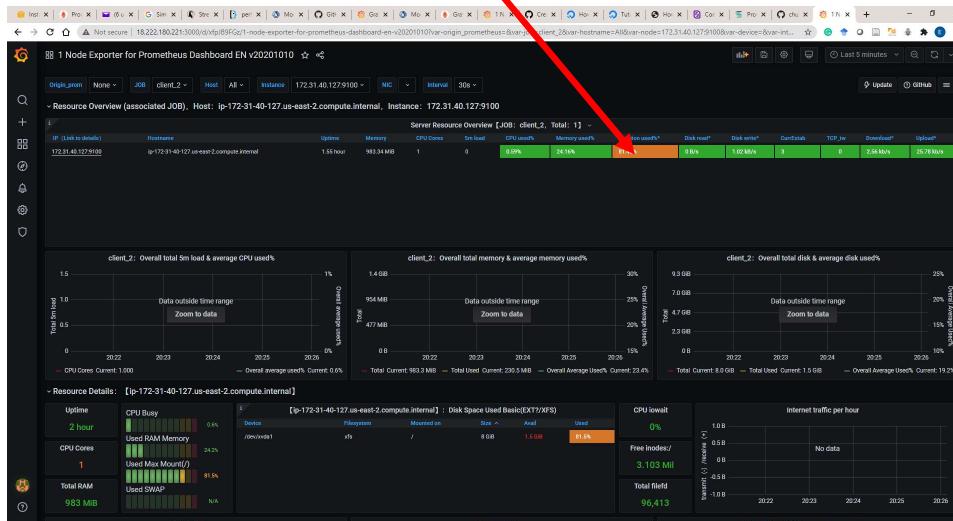
Client-2 Server: Simulate Disk usage

```
[ec2-user@ip-172-31-40-127 ~]$ df -h
Filesystem  Size Used Avail Use% Mounted on
devtmpfs    474M  0 474M  0% /dev
tmpfs       492M  0 492M  0% /dev/shm
tmpfs       492M 436K 492M  1% /run
tmpfs       492M  0 492M  0% /sys/fs/cgroup
/dev/xvda1   8.0G 1.6G 6.5G 19% / ======> before allocating disk space
tmpfs       99M  0 99M  0% /run/user/1000
[ec2-user@ip-172-31-40-127 ~]$ fallocate -l 5G test-client-2
[ec2-user@ip-172-31-40-127 ~]$ df -h
Filesystem  Size Used Avail Use% Mounted on
devtmpfs    474M  0 474M  0% /dev
tmpfs       492M  0 492M  0% /dev/shm
tmpfs       492M 436K 492M  1% /run
tmpfs       492M  0 492M  0% /sys/fs/cgroup
/dev/xvda1   8.0G 6.6G 1.5G 82% / ======> After allocating disk space
tmpfs       99M  0 99M  0% /run/user/1000
[ec2-user@ip-172-31-40-127 ~]$ rm test-client-2
[ec2-user@ip-172-31-40-127 ~]$ df -h
Filesystem  Size Used Avail Use% Mounted on
devtmpfs    474M  0 474M  0% /dev
tmpfs       492M  0 492M  0% /dev/shm
tmpfs       492M 436K 492M  1% /run
tmpfs       492M  0 492M  0% /sys/fs/cgroup
/dev/xvda1   8.0G 1.6G 6.5G 19% / ======> After deallocating the disk space
tmpfs       99M  0 99M  0% /run/user/1000
[ec2-user@ip-172-31-40-127 ~]$
```

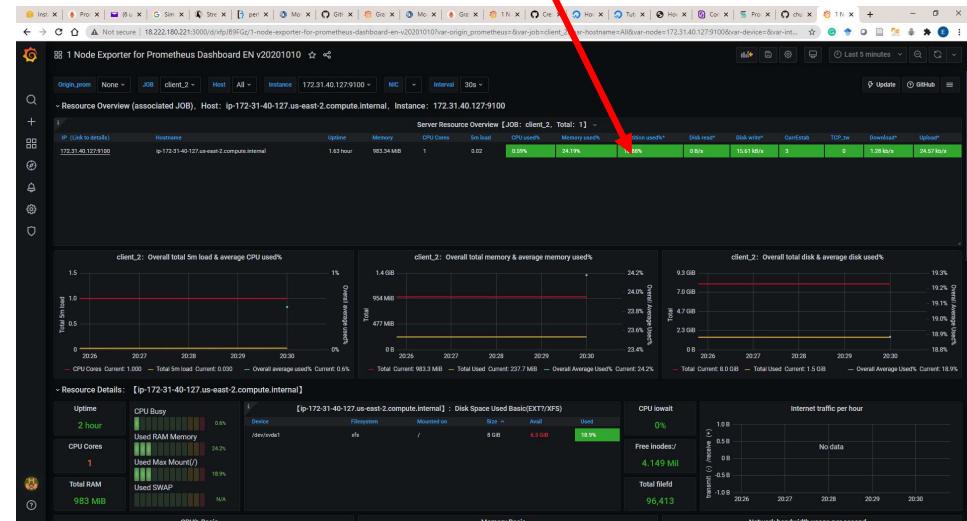
Simulate high cpu, memory, and disk usage on monitored instances

Prometheus-Host Server – Simulate Disk usage

After Allocating the 5G disk space.



After Deleting the file.

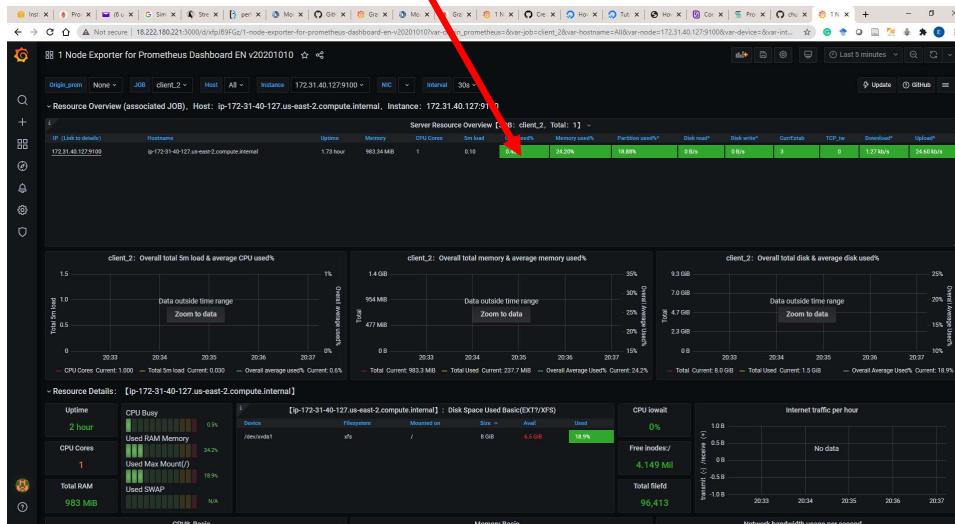


Simulate high cpu, memory, and disk usage on monitored instances

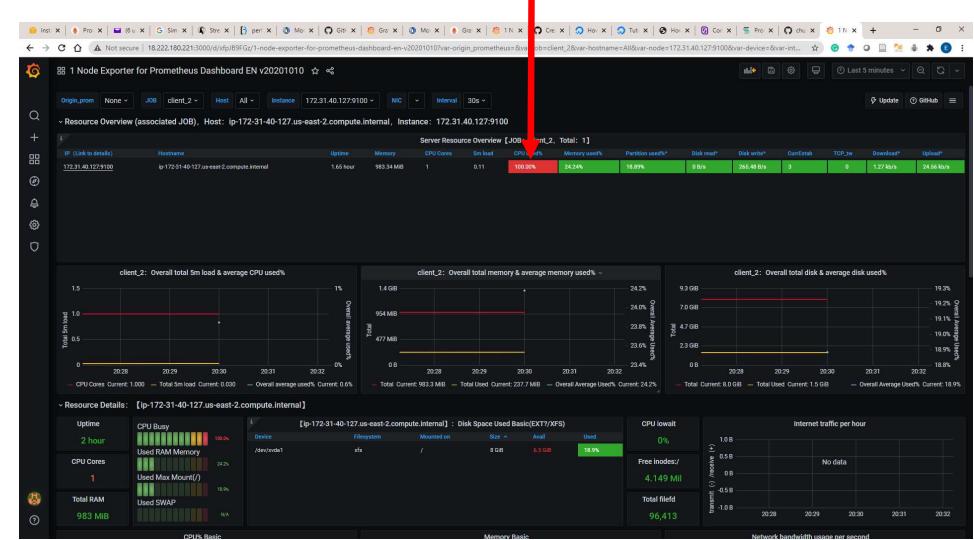
Prometheus-Host Server – Simulate cpu usage

```
[ec2-user@ip-172-31-40-127 ~]$ nproc | xargs seq | xargs -n1 -P4 md5sum /dev/zero
```

Before running the process.



Running the test.



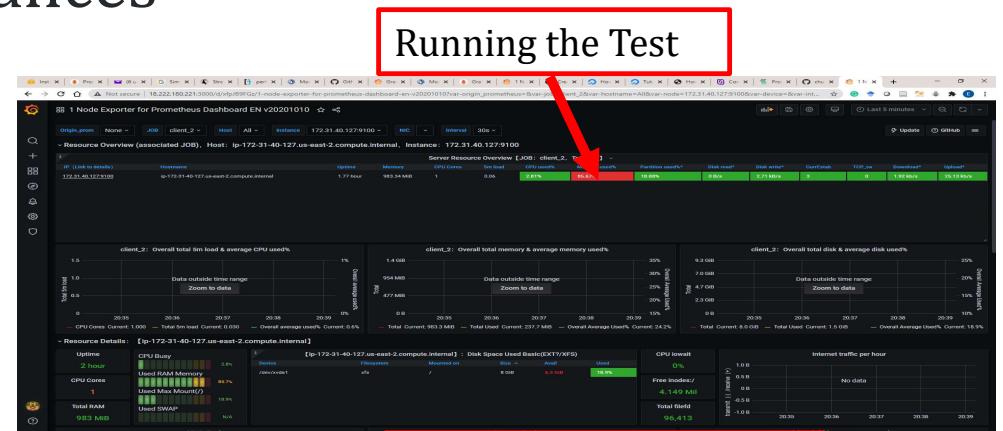
Simulate high cpu, memory, and disk usage on monitored instances

Prometheus-Host Server – Simulate Memory usage:
[ec2-user@ip-172-31-40-127 ~]\$ python
memoryLoadRunner.py 600

```
#!/usr/bin/env python
import sys
import time

if len(sys.argv) != 2:
    print "usage: fillmem <number-of-megabytes>"
    sys.exit()

count = int(sys.argv[1])
megabyte = (0,) * (1024 * 1024 / 8)
data = megabyte * count
while True:
    time.sleep(1)
```



After running the Test

