# Step 2 Documentation: Deploying and Configuring Prometheus Server Using Ansible and Helm Chart

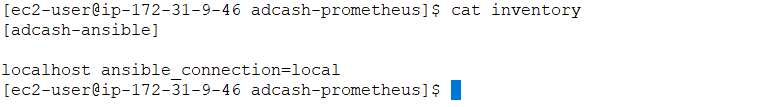
**Overview:**

This project demonstrates the deployment of a Prometheus monitoring server using the Ansible configuration management tool on an Amazon Linux EC2 instance within the AWS environment. The Prometheus server is configured to scrape metrics from an application hosted on a separate virtual machine (VM) within the same AWS infrastructure.

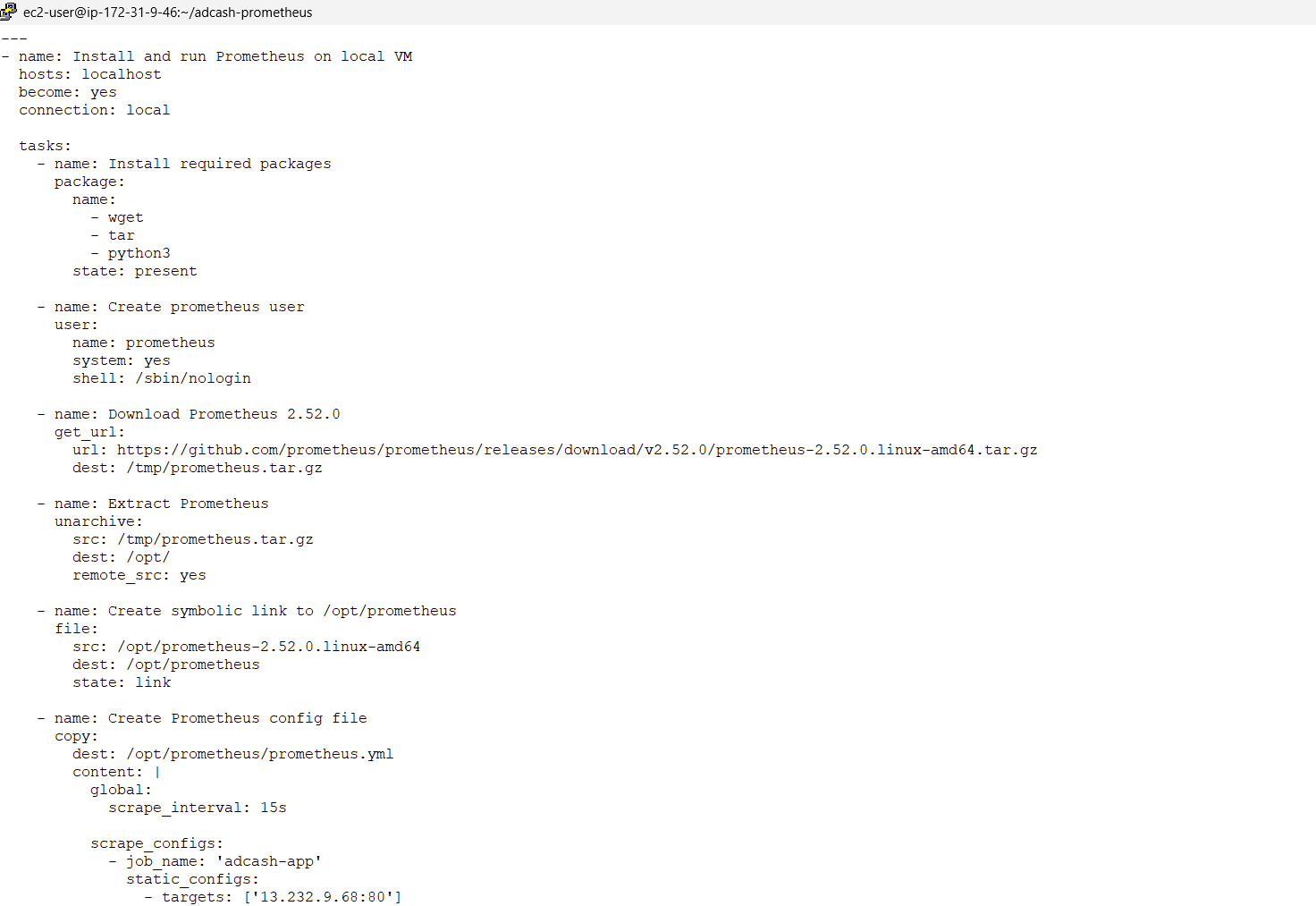
**Inventory Configuration:**

The Ansible inventory file defines the target host for playbook execution:

[adcash-ansible]

localhost ansible\_connection=local  
  


**Prometheus Server Playbook:**

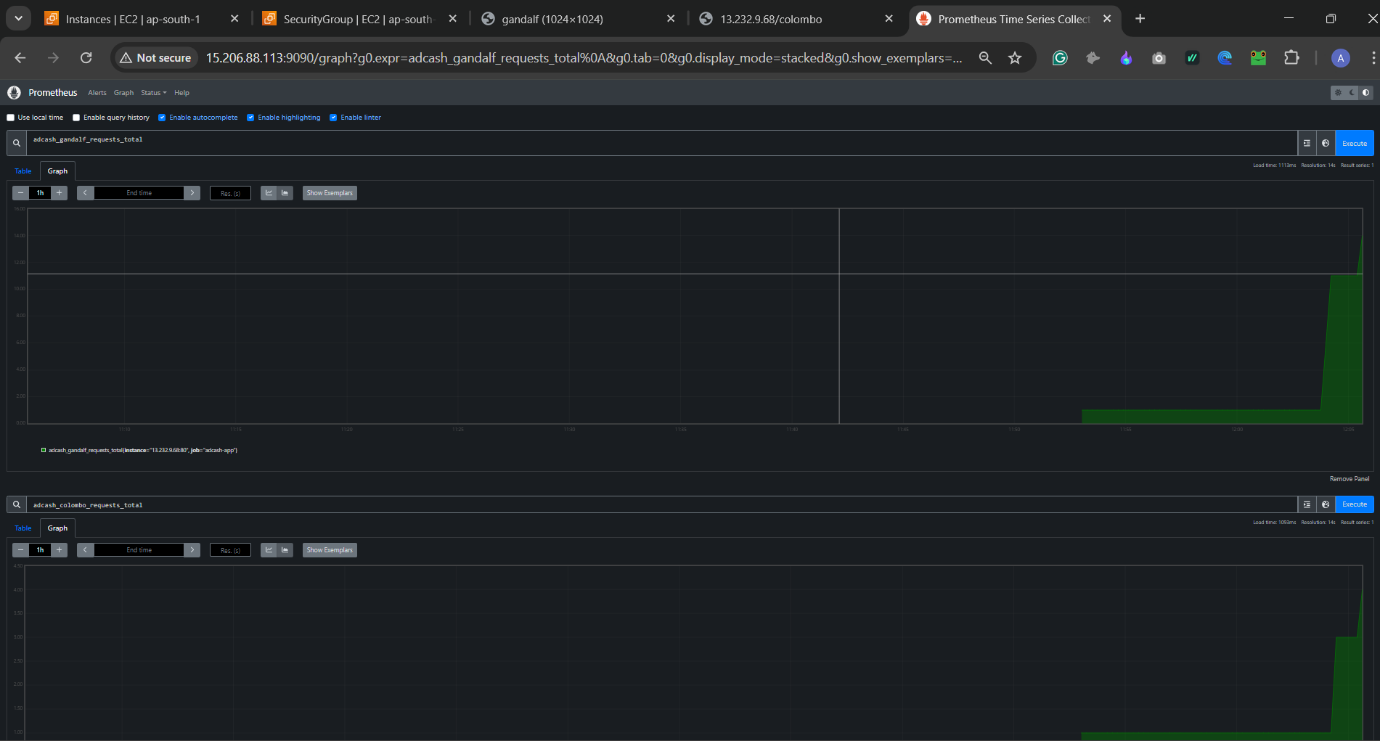
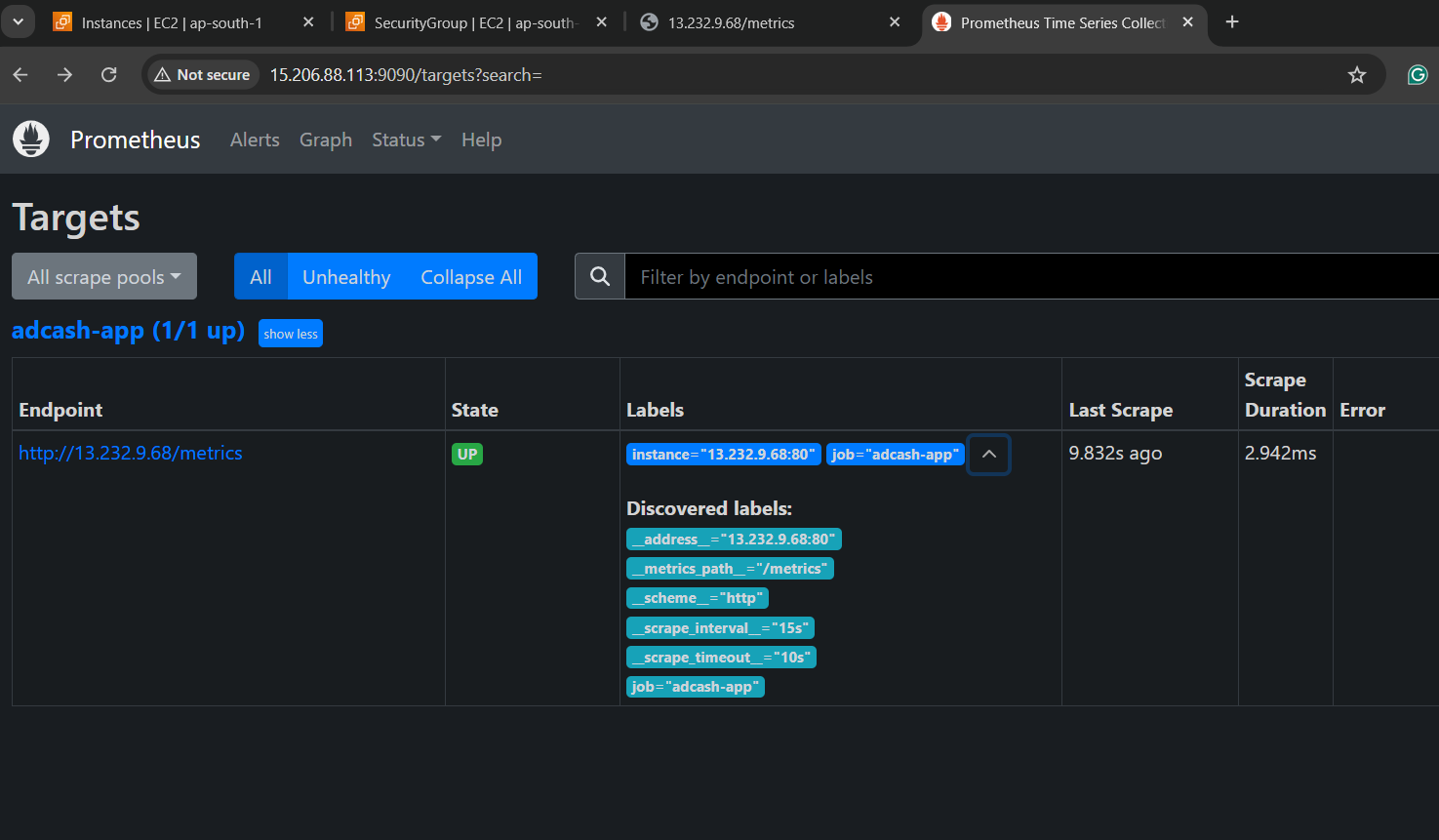
Ansible is used to configure and launch the Prometheus server on the EC2 instance. The server operates on port **9090** and is set to scrape metrics from an external application running   
on a separate VM. Metrics can be accessed via the following endpoints:  
  
  
  
<http://13.232.9.68/gandalf>

<http://13.232.9.68/colombo>

<http://13.232.9.68/metrics>

**Execution Command**

To run the playbook, execute the following command:

**ansible-playbook -i inventory adcash-prometheus-playbook.yml**

**Step 2 (optional): Deploying Prometheus Server via Helm on EKS Cluster**

This process outlines the steps taken to deploy a Prometheus server on our EKS Kubernetes cluster using a Helm chart. The cluster is also hosting our Node.js service.

**Create a Secret for Additional Scrape Configurations**

We begin by creating a Kubernetes secret to include additional scrape configurations for Prometheus:

**kubectl create secret generic prometheus-additional-scrape-configs \**

**--from-file=additional-scrape-configs.yaml**

**Contents of additional-scrape-configs.yaml:**

**- job\_name: 'adcash-eks'**

**metrics\_path: /metrics**

**static\_configs:**

**- targets: ['ae00b2b2077ee44b1a684c243853c3a6-813237557.ap-south-1.elb.amazonaws.com:80'**]

**Install Prometheus server using Helm:**

We use the following Helm command to install or upgrade Prometheus with a custom values file:

helm upgrade --install prometheus prometheus-community/prometheus -f main-values.yaml

**Contents of main-values.yaml:**

server:

persistentVolume:

enabled: false

service:

type: LoadBalancer

port: 80

extraScrapeConfigsSecret: prometheus-additional-scrape-configs

alertmanager:

enabled: false

pushgateway:

enabled: false

nodeExporter:

enabled: false

kubeStateMetrics:

enabled: false

**Commands:**

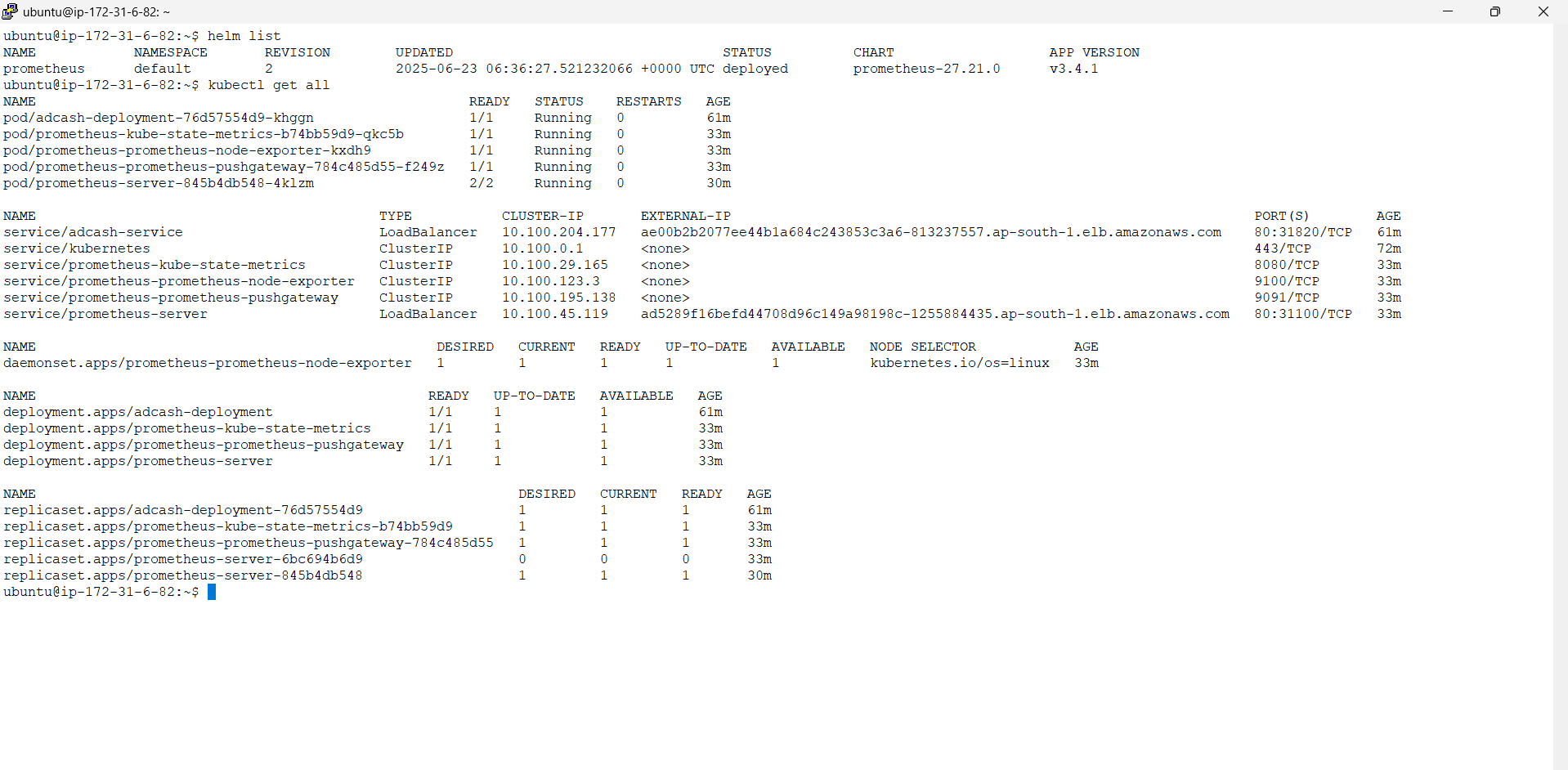
To verify the setup and check the deployed resources, we used the following commands:

**helm list**

**helm upgrade --install prometheus prometheus-community/prometheus -f main-values.yaml**

**helm list**

**kubectl get all**



**Accessing Prometheus UI**

Once the Prometheus server is deployed, it is exposed through a LoadBalancer service. The external IP or DNS name assigned by AWS can be used to access the Prometheus web UI.

**Prometheus URL Format:**

http://ae00b2b2077ee44b1a684c243853c3a6-813237557.ap-south-1.elb.amazonaws.com/

