Create ci pipeline that will deploy and configure these VMs using terraform and ansible according following requirements

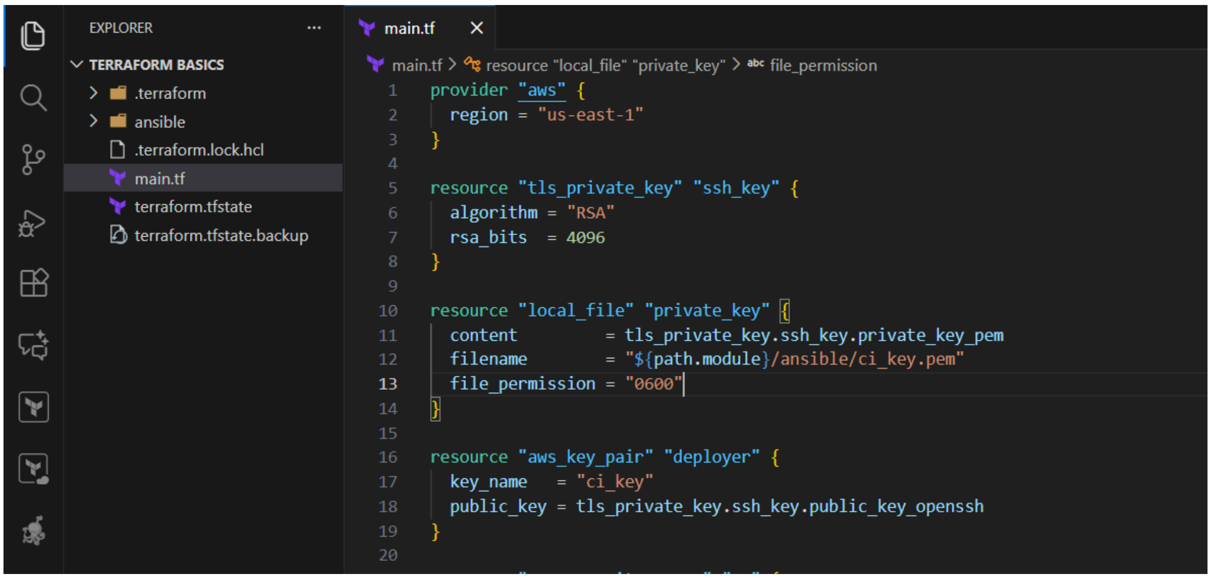
1. **Deploy 2 virtual machines using terraform.**
    - **first vm on Amazon linux, hostname: c8.local**
    - **second vm on ubuntu 21.04, hostname: u21.local**
2. **As a result of terraform execution, dynamically create inventory for ansible**
    - **c8.local should be in the frontend group**
    - **u21.local should be in the backend group**
3. **Create ansible playbook for c8.local and u21.local**
4. **for linux OS playbook should apply the following changes**
    - **selinux: disable**
    - **firewalld: disable**
5. **for frontend playbook group should install and configure nginx**
    - **nginx configuration should do proxying from port 80 on port 19999 to the**
    - **backend group**
6. **for the backend group, the playbook must install the Netdata application from the**

- **official repositories and run it on port 19999.**

Open vs code and write the script to create infrastuctures frontend as linux and backend as ubuntu.

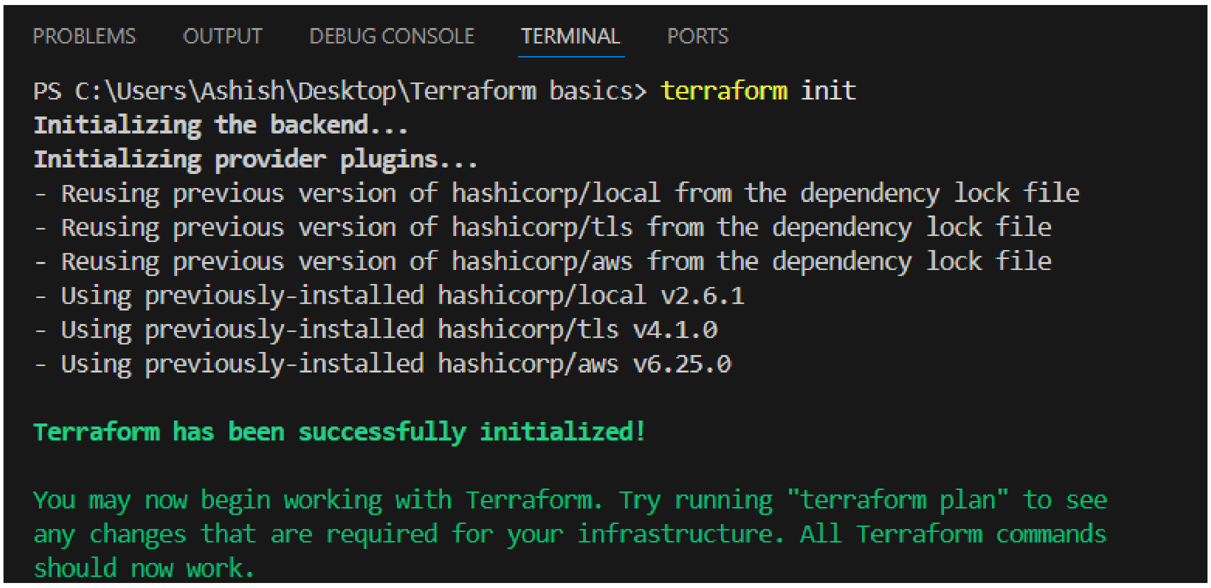
```
provider "aws" {
  region = "us-east-1"
}

resource "tls_private_key" "ssh_key" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

resource "local_file" "private_key" {
  content         = tls_private_key.ssh_key.private_key_pem
  filename        = "${path.module}/ansible/ci_key.pem"
  file_permission = "0600"
}

resource "aws_key_pair" "deployer" {
  key_name   = "ci_key"
  public_key = tls_private_key.ssh_key.public_key_openssh
}

resource "aws_security_group" "sg" {
  name = "ci-sg"

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 19999
    to_port     = 19999
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

```
data "aws_ami" "amazon_linux" {
  most_recent = true
  owners      = ["amazon"]

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*-x86_64-gp2"]
  }
}

data "aws_ami" "ubuntu_2104" {
  most_recent = true
  owners      = ["099720109477"] # Canonical

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*"]
  }
}

resource "aws_instance" "c8" {
  ami                    = data.aws_ami.amazon_linux.id
  instance_type          = "t3.micro"
  key_name               = aws_key_pair.deployer.key_name
  vpc_security_group_ids = [aws_security_group.sg.id]
  subnet_id = "subnet-0a192382de0e2bf6a"

  tags = {
    Name = "c8.local"
  }
}

resource "aws_instance" "u21" {
  ami                    = data.aws_ami.ubuntu_2104.id
  instance_type          = "t3.micro"
  key_name               = aws_key_pair.deployer.key_name
  vpc_security_group_ids = [aws_security_group.sg.id]
  subnet_id = "subnet-0a192382de0e2bf6a"

  tags = {
    Name = "u21.local"
  }
}

resource "local_file" "ansible_inventory" {
  filename = "${path.module}/ansible/inventory.ini"

  content = <<EOT
[frontend]
c8.local ansible_host=${aws_instance.c8.public_ip} ansible_user=ec2-user
ansible_ssh_private_key_file=./ci_key.pem

[backend]
u21.local ansible_host=${aws_instance.u21.public_ip} ansible_user=ubuntu
ansible_ssh_private_key_file=./ci_key.pem
EOT
}
```

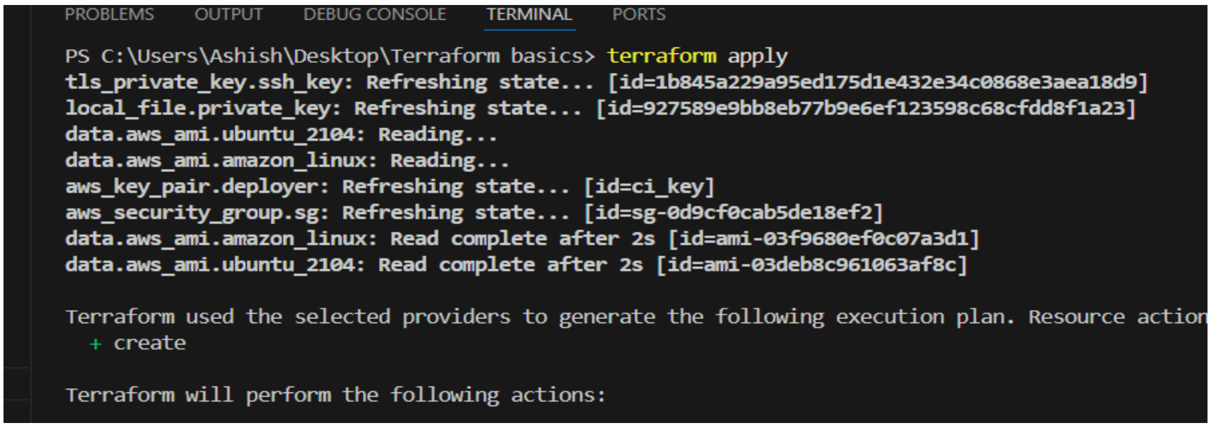


```
provider "aws" {
  region = "us-east-1"
}

resource "tls_private_key" "ssh_key" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

resource "local_file" "private_key" {
  content         = tls_private_key.ssh_key.private_key_pem
  filename        = "${path.module}/ansible/ci_key.pem"
  file_permission = "0600"
}

resource "aws_key_pair" "deployer" {
  key_name   = "ci_key"
  public_key = tls_private_key.ssh_key.public_key_openssh
}
```

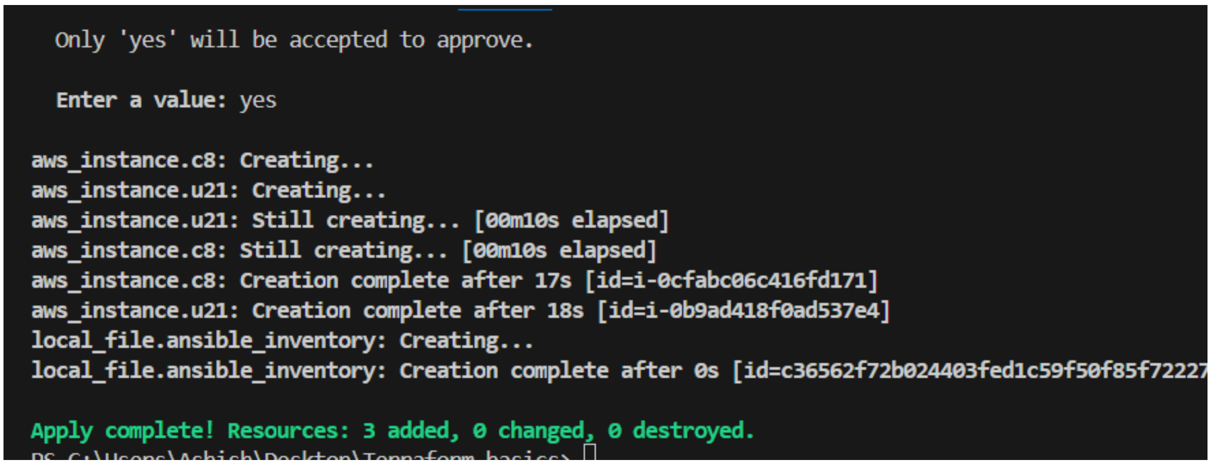
- **terraform init**
- **terraform apply**



```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/local from the dependency lock file
- Reusing previous version of hashicorp/tls from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/local v2.6.1
- Using previously-installed hashicorp/tls v4.1.0
- Using previously-installed hashicorp/aws v6.25.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
```

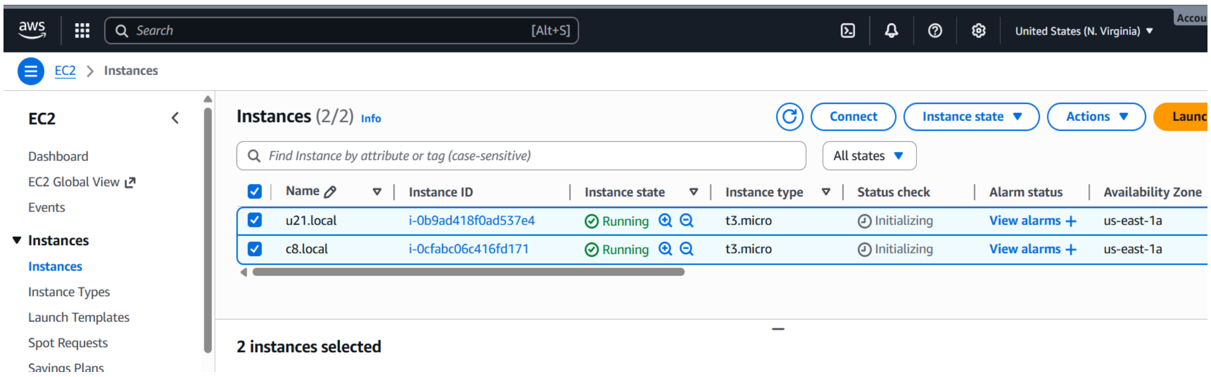


```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
tls_private_key.ssh_key: Refreshing state... [id=1b845a229a95ed175d1e432e34c0868e3aea18d9]
local_file.private_key: Refreshing state... [id=927589e9bb8eb77b9e6ef123598c68cfdd8f1a23]
data.aws_ami.ubuntu_2104: Reading...
data.aws_ami.amazon_linux: Reading...
aws_key_pair.deployer: Refreshing state... [id=ci_key]
aws_security_group.sg: Refreshing state... [id=sg-0d9cf0cab5de18ef2]
data.aws_ami.amazon_linux: Read complete after 2s [id=ami-03f9680ef0c07a3d1]
data.aws_ami.ubuntu_2104: Read complete after 2s [id=ami-03deb8c961063af8c]

Terraform used the selected providers to generate the following execution plan. Resource action
  + create

Terraform will perform the following actions:
```



```
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.c8: Creating...
aws_instance.u21: Creating...
aws_instance.u21: Still creating... [00m10s elapsed]
aws_instance.c8: Still creating... [00m10s elapsed]
aws_instance.c8: Creation complete after 17s [id=i-0cfabc06c416fd171]
aws_instance.u21: Creation complete after 18s [id=i-0b9ad418f0ad537e4]
local_file.ansible_inventory: Creating...
local_file.ansible_inventory: Creation complete after 0s [id=c36562f72b024403fed1c59f50f85f72227

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

Two instances has been created with this.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
u21.local	i-0b9ad418f0ad537e4	Running	t3.micro	Initializing	View alarms +	us-east-1a
c8.local	i-0cfabc06c416fd171	Running	t3.micro	Initializing	View alarms +	us-east-1a

2 instances selected

Add this files as playbooks

- **vi nginx.conf.j2**

```
user nginx;
worker_processes auto;


events {
    worker_connections 1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    sendfile        on;
    keepalive_timeout 65;

    upstream backend_app {
        server {{ hostvars['backend']['ansible_host'] |
default('127.0.0.1') }}:8080;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://backend_app;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
}
```

- ubuntu-playbook.yml

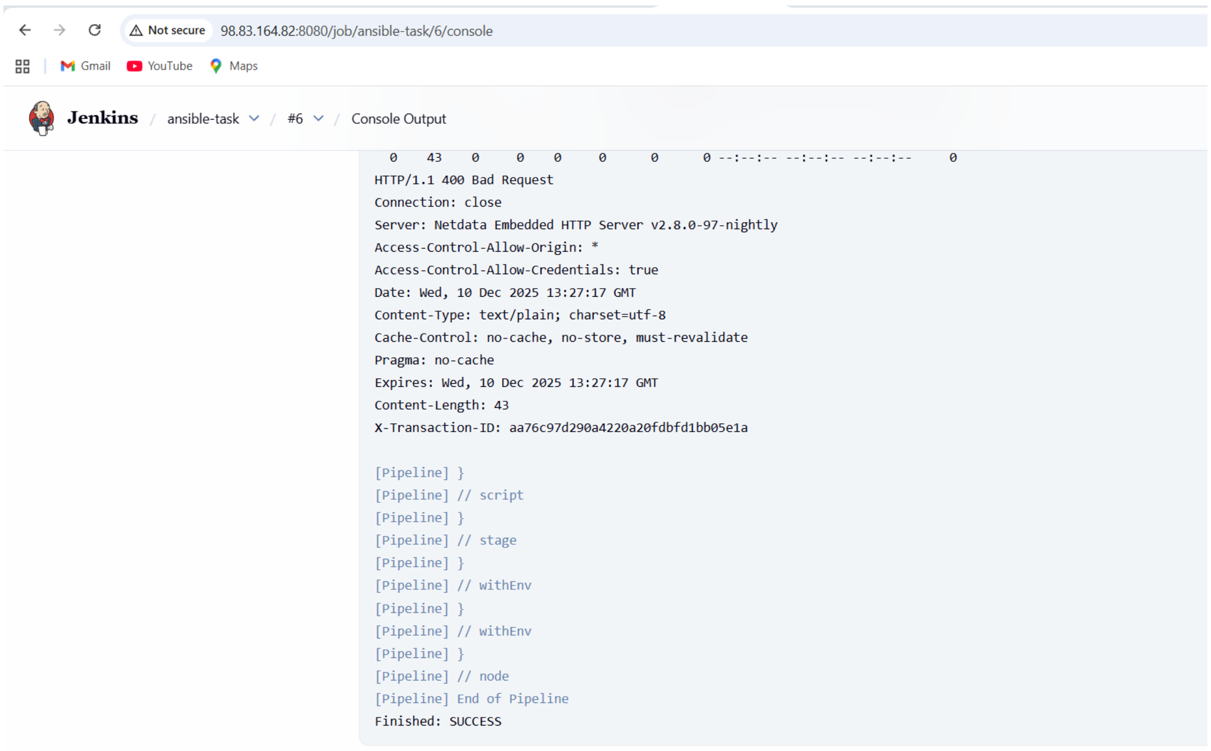
```
---
- hosts: backend
  become: yes

  tasks:
    - name: Install required packages
      apt:
        name:
          - curl
          - wget
        state: present
        update_cache: yes

    - name: Download Netdata installation script
      get_url:
        url: "https://my-netdata.io/kickstart.sh"
        dest: "/tmp/netdata-install.sh"
        mode: '0755'

    - name: Run Netdata installer (non-interactive)
      command: "bash /tmp/netdata-install.sh --non-
  interactive"
      args:
        creates: /etc/netdata/netdata.conf

    - name: Ensure Netdata service is running
      service:
        name: netdata
        state: started
        enabled: yes

    - name: Update netdata.conf to listen on 0.0.0.0
      lineinfile:
        path: /etc/netdata/netdata.conf
        regexp: '^#?bind socket to IP ='
        line: 'bind socket to IP = 0.0.0.0'
      notify: restart netdata
```

```
handlers:
  - name: restart netdata
    service:
      name: netdata
      state: restarted
```

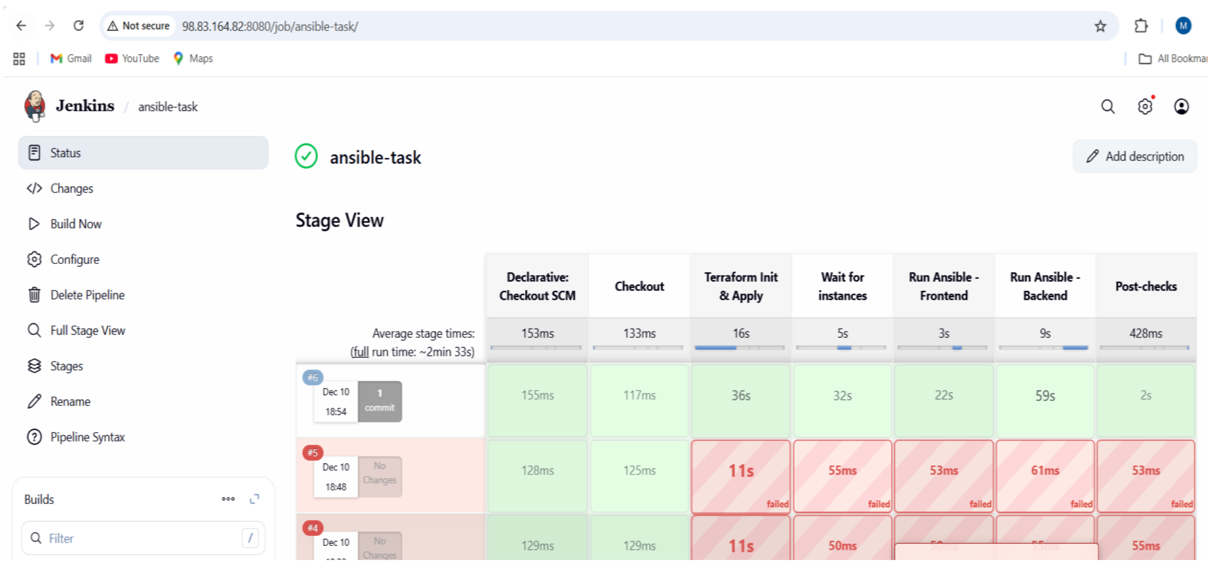
go to Jenkins server and build the pipeline

select go to git scm and give this url

- https://github.com/mujaheed00/ansible-task.git



```
  0    43    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
HTTP/1.1 400 Bad Request
Connection: close
Server: Netdata Embedded HTTP Server v2.8.0-97-nightly
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Date: Wed, 10 Dec 2025 13:27:17 GMT
Content-Type: text/plain; charset=utf-8
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: Wed, 10 Dec 2025 13:27:17 GMT
Content-Length: 43
X-Transaction-ID: aa76c97d290a4220a20fdbfd1bb05e1a

[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

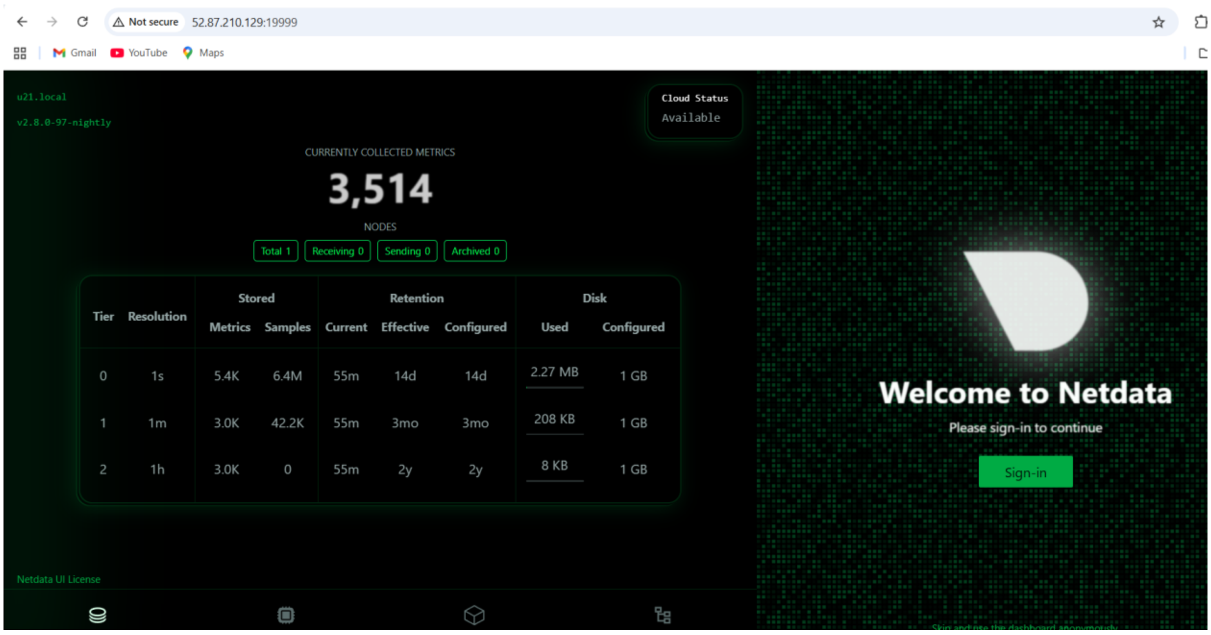


ansible-task

Stage View

	Declarative: Checkout SCM	Checkout	Terraform Init & Apply	Wait for instances	Run Ansible - Frontend	Run Ansible - Backend	Post-checks
Average stage times: (full run time: ~2min 33s)	153ms	133ms	16s	5s	3s	9s	428ms
#6 Dec 10 18:54 1 commit	155ms	117ms	36s	32s	22s	59s	2s
#5 Dec 10 18:48 No Changes	128ms	125ms	11s failed	55ms failed	53ms failed	61ms failed	53ms failed
#4 Dec 10 No Changes	129ms	129ms	11s	50ms			55ms

Search with with backend instance with the ip:19999



CURRENTLY COLLECTED METRICS

3,514

NODES

Total 1 Receiving 0 Sending 0 Archived 0

Tier	Resolution	Stored		Retention			Disk	
		Metrics	Samples	Current	Effective	Configured	Used	Configured
0	1s	5.4K	6.4M	55m	14d	14d	2.27 MB	1 GB
1	1m	3.0K	42.2K	55m	3mo	3mo	208 KB	1 GB
2	1h	3.0K	0	55m	2y	2y	8 KB	1 GB