Geonode Cloud

Versión: 1.0.0

Contents:

Capítulo 1

Introduction

GeoNode Cloud is an advanced implementation of the GeoNode[1] platform in the cloud, focused on maximizing the use of native or adapted technologies for cloud environments. This solution is designed to be deployed on Kubernetes, which facilitates its scalability, management and resilience.

GeoNode Cloud incorporates the GeoServer Cloud[2] project, which provides robust support for the publication, editing and management of geospatial data, thus reinforcing its purpose of offering a modern and efficient infrastructure for the management of geospatial information in the cloud.

With GeoNode Cloud, organizations can benefit from greater flexibility, reduced operational costs, and seamless integration with other cloud-native tools and services.

[1] https://github.com/GeoNode/geonode
[2] https://github.com/geoserver/geoserver-cloud

Capítulo 2

Tabla de contenidos

2.1 Project Structure

The project structure for deploying GeoNode Cloud and GeoServer Cloud on Kubernetes is organized into key directories that contain the manifests required to configure and operate the applications. Within the following repository is the project that contains all the manifests that will be used to perform the deployment.

2.1.1 Main Directories

- gn-cloud
- gs-cloud
- configs
- database

2.2 Architecture & Technology

The solution architecture is divided into the following components:

- Geonode Cloud Core[3]
- GeoNode Cloud Mapstore Client[4]
- Rabbitmq[5]

[3] https://github.com/Kan-T-IT/geonode-cloud-core
[4] https://github.com/Kan-T-IT/geonode-cloud-mapstore-client
[5] https://github.com/rabbitmq

- GeoServer Cloud[6]
- Postgres[7] with PostGis extension
- Nginx[8]
- Flower[9]

Specifically **Geonode Cloud Core** contains the following main technological components for its operation:

- Django Framework
- Memcached
- Geonode Import
- pyCSW[10]
- Celery
- Geoserver App Django - ACL Capability[11]

The architecture is based on the use of microservices, where it is planned to incorporate new microservices that today are in the monolithic component of Django.

2.2.1 Distribution and deployment

Docker images for all the services are available on DockerHub, under the KAN Territory & IT organization[12].

You can find production-suitable deployment files for docker-compose and podman under the docs/deploy folder.

[6] https://github.com/geoserver/geoserver-cloud
[7] https://github.com/postgres
[8] https://github.com/nginx/nginx
[9] https://github.com/mher/flower
[10] https://github.com/geopython/pycsw
[11] https://github.com/Kan-T-IT/geonode-cloud-core/tree/main/geonode/geoserver/acl
[12] https://hub.docker.com/u/kantit

2.2.2 Contributing

Please read the contribution guidelines before contributing pull requests to the Geonode Cloud project.

Follow the developer's guide to know more about the project's technical details.

2.2.3 Status

Read the changelog[13] for more information.

2.2.4 Bugs

GeoNode Cloud's issue tracking is at this Issues GitHub[14] repository.

2.2.5 Roadmap

TDB

2.3 Deployment

For the deployment of Geonode Cloud we can deploy it on different Kubernete platforms, here are the details of the deployment on MickoK8S

2.3.1 Deployment on MicroK8S

Requisites

- MicroK8S.
  - Ingress module.
  - DNS module.
  - Cert-manager module.

Use snap to install microk8s

[13] https://github.com/Kan-T-IT/geonode-cloud/releases
[14] https://github.com/Kan-T-IT/geonode-cloud/issues

```
sudo snap install microk8s --classic
```

Enable necesary micro8s modules

Create certmanager config to enable letsencrypt using your own email

```
microk8s kubectl apply -f - <<EOF
---
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
name: letsencrypt
spec:
acme:
    email: YOUREMAIL@DOMAIN.com
    server: https://acme-v02.api.letsencrypt.org/directory
    privateKeySecretRef:
    # Secret resource that will be used to store the account's private␣
↪key.
    name: letsencrypt-account-key
    # Add a single challenge solver, HTTP01 using nginx
    solvers:
    - http01:
        ingress:
        class: public
EOF
```

2.3.2 Deployment

Clone this repository

Edit all fields in .env file with the necesary information.

```
KUBERNETES_SITE_URL=GEONODE_CLOUD_FINAL_URL    # i.e.: cloud.mygeonode.
↪com
KUBERNETES_NODE_NAME=YOUR_CLUSTER_NAME_NAME    # usually host machine␣
↪name
KUBERNETES_VOL_DIR=YOUR_DESIRED_LOCATION       # this path shold exist
CLUSTER_ISSUER_NAME=YOUR_CLUSTER_ISSUER_NAME   # created earlier in␣
↪this guide
SERVER_PUBLIC_IP=YOU.RPU.BLI.CIP               # the public ipv4 of␣
↪the server
GEONODE_PASSWORD=admin                         # password for geonode␣
↪admin user
GEOSERVER_PASSWORD=geoserver                   # password for␣
↪geoserver admin user
```

Run

```
./install.sh`
```

and enjoy.