Setting up a new project [WIP]

Creating Services and Containers

1. Create a new project in geostreams. Instructions are here.
2. Configure the docker-compose file for the services that need to be added. The docker-compose file linked below The docker-compose file for this project is based on the Default Geostreams file and traefik v2. This way minimal configuration is required on the machine.
3. a. **docker-compose.yml**

```
version: "3.3"

services:
  traefik:
    image: traefik:latest
    networks:
      - geostreams
    volumes:
      - ./traefik:/traefik/
      - /var/run/docker.sock:/var/run/docker.sock:ro
    ports:
      - 80:80
      - 443:443
      - 8001:8001

    command:
      - --entrypoints.private.address=:8001
      - --entrypoints.web.address=:80
      - --entrypoints.websecure.address=:443
      - --entrypoints.web.http.redirections.entryPoint.to=websecure
      - --entrypoints.web.http.redirections.entryPoint.scheme=https
      - "--providers.docker=true"
      - "--api=true"
      - --providers.docker.exposedbydefault=false

          # SSL Certificate Generation using Lets Encrypt
      # - "--certificatesresolvers.le.acme.caserver=https://acme-staging-v02.api.letsencrypt.
org/directory"
      - --certificatesResolvers.le.acme.email=dev@mbclab.ncsa.illinois.edu
      - --certificatesResolvers.le.acme.storage=/traefik/acme.json
      - --certificatesResolvers.le.acme.httpChallenge=true
      - --certificatesResolvers.le.acme.httpChallenge.entryPoint=web

    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.dashboard.service=api@internal"

      # Route setting
      - "traefik.http.routers.dashboard.rule=Host(`${TRAEFIK_HOST:-localhost}`)"

      # set TLS (https)
      - "traefik.http.routers.dashboard.tls=true"
      - "traefik.http.routers.dashboard.entrypoints=private"
      - "traefik.http.routers.dashboard.tls.certresolver=le"


  # ------------------------------------------------------------------------
  # GEOSTREAMS STACK
  # ------------------------------------------------------------------------
  geodashboard:
    image: hub.ncsa.illinois.edu/geostreams/gd-smartfarm:develop
    networks:
      - geostreams
    labels:
      - traefik.enable=true
      - traefik.http.services.geodashboard.loadbalancer.server.port=80
      - traefik.http.routers.geodashboard.rule=Host(`${TRAEFIK_HOST:-localhost}`) &&
(PathPrefix(`${GD_PREFIX_PATH:-/}`))
      - traefik.http.routers.geodashboard.entrypoints=websecure
      - traefik.http.routers.geodashboard.tls=true
```

```
      - traefik.http.routers.geodashboard.tls.certresolver=le
    restart: unless-stopped

  geostreams:
    image: geostreams/geostreams
    env_file:
      - ./geostreams.env
    networks:
      - geostreams
    labels:
      - traefik.enable=true
      - traefik.http.services.geostreams.loadbalancer.server.port=9000
      - traefik.http.routers.geostreams.rule=Host(`${TRAEFIK_HOST:-localhost}`) && (PathPrefix
(`${GEOSTREAMS_PREFIX_PATH:-/}`))
      - traefik.http.routers.geostreams.entrypoints=websecure
      - traefik.http.routers.geostreams.tls=true
      - traefik.http.routers.geostreams.tls.certresolver=le

    volumes:
      - ./application.conf:/home/geostreams/conf/application.conf
      - ./messages.en:/home/geostreams/conf/messages.en
    healthcheck:
       test: ["CMD", "curl", "-s", "--fail", "http://localhost:9000/geostreams/api/status"]
    restart: unless-stopped
  postgres:
    image: mdillon/postgis:9.5
    networks:
      - geostreams
    ports:
      - 5432:5432
    volumes:
      - ./postgres:/var/lib/postgresql/data
    restart: unless-stopped



networks:
  geostreams:
```

b. The following .env file sets up the routes:

```
TRAEFIK_HOST=localhost
GEOSTREAMS_PREFIX_PATH=/geostreams
GD_PREFIX_PATH=/
```

4. For geostreams, use the example application.conf https://github.com/geostreams/geostreams/blob/master/conf/reference.conf .
5. If using clowder, follow the link to initialize clowder https://github.com/clowder-framework/clowder#initializing-clowder.

Setting up the VM

- **Create a VM**
  - Install python-openstackclient
  - Create an rc file using the example below used for gltg nebula

```
export OS_AUTH_URL=http://nebula.ncsa.illinois.edu:5000/v2.0
export OS_TENANT_ID=c4121a001a8240d4a8b701d664ef4bf0    # Project ID in nebula
export OS_TENANT_NAME="GLTG"
export OS_PROJECT_NAME="GLTG"
export OS_USERNAME=<nebula_username>
export OS_PASSWORD=<nebula_password>
export OS_REGION_NAME="RegionOne"

export PS1="(openstack)[\e]0;\a]\h:\W \u [${OS_PROJECT_NAME}] $ "
```

- Run `bash --rcfile <rc_file_name>` to create a virtual environment
- Download and run the makevm-script (updated version in the comments) to create the vm.
- Use one of the available Floating IP Addresses on Nebula and provide it to the script.
- Email help+neteng@ncsa.illinois.edu to setup domain name for machine.
- The docker-compose file is setup to use Lets Encrypt for SSL certificate generation and renewal.

- **Setup Docker**
  - Install docker and docker-compose on the machine. Additionally, install pass and gnupg2 without which logging into docker will give an error.
    - Docker Installation Instructions
    - Docker-compose installation Instructions
    - ```
      sudo apt-get install pass gnupg2
      ```
  - Login to hub.ncsa.illinois.edu using the robot account login. The account information is stored in lastpass folder.
    - ```
      docker login -u `robot$githubgeostreams` -p `<TOKEN>`  hub.ncsa.illinois.edu
      ```
- **Set up Backup**
- **Initialize Geostreams**
  - Move the docker-compose file and other relevant files to the machine.
  - Run docker-compose up -d.
  - Initialize the database by running the geostreams container with command `initialize` once.

Configuring Build and Deployment Pipeline for Project

- Add the following Github Actions public key to the machine.

  `cat PATH_TO_KEY | ssh -i PATH_TO_ID_KEY user@hostname "cat >> ~/.ssh/authorized_keys"`

GitHub_Repo_Public_key
`ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDCL /mqWBK1BTwZxKnZ546F9IbFlYY6qgFzh9xTM6eY+DxagKVV1BAr4kbqamnLYDzrOLC6zY+8k3xGS1HSxp8UAWXLnPzDkb13uXj+neG ty7DwMIVWRVSc0JNa0cRaEKI1wC9AK1utKEU7aaGu6fZsmExmXNzIzxLIYVUFdW8G2GVoK9wNSba3OT2rneutgOUrb5PR6ADpfBEO8 h48CcP6edw5A2HoJ0ZXySeadvnInOhp3yisO3khaZ7t4ZPRtRVRM+M+V9H+1JpOmsulfAZUyEdntzU1MkduFAz+X5T /h9IhHYlplqJ00GEjc/zIPS9y39Be5XqgvMadapupmeGpZWU6K /xvluATcP1xGqy7ytytAr6ZIbsCyKWGJXeUYsR8K1MdNC8zAVB+2Cnu3Df0TUf7xIV6sSx66MtehAADIGxtik4KGl5DZiWENgf0aCS eJHAjRkxGx4mxb/Cp9iercoKs+6uIAKd7fU/pYK9kw3z8WlkOsTcQcII9hzY6bQs= Smartfarm Key for Github Actions`

- Update github deployment action and add the details for the new project under strategy->matrix.

```
name: <project_name>
gd-name: <lerna_project_name>
prod-host: <production_host_name>
dev-host: <dev_host_name>
file-path: <docker-compose file path on VM>
description:
```

- For development machines, an image with a develop tag will be created in hub.ncsa.illinois.edu and for production machines the latest tag will be used. The docker-compose files should be updated accordingly.