АКЦИОНЕРНОЕ ОБЩЕСТВО "ГЕОИНФОРМАЦИОННЫЕ СИСТЕМЫ"

ОГРН 1229100010259
ИНН 9102283756
+7(978) 862-06-18, mail@mycrg.ru

295014, Республика Крым, г Симферополь, Евпаторийское ш, д. 8, офис 313/1

Инструкция по развертыванию Геоинформационной системы «GIS-Мастерская» (ГИС «GIS-Мастерская»)

Развертывание (Deployment) системы GIS-Мастерская

1. **Создать новую виртуальную машину (VM) с операционной системой (ОС) Линукс Ubuntu**
2. **Установить ПО (Docker, Java, docker-compose)**

- **Java**
  Установка JRE/JDK по умолчанию
  Самый простой вариант установки Java — использовать версию, входящую в пакет Ubuntu. По умолчанию в пакет Ubuntu 20.04 входит Open JDK 11 (версия JRE и JDK с открытым исходным кодом).
  Для установки этой версии нужно вначале обновить указатель пакетов:
  `sudo apt update`
  Затем нужно проверить, выполнялась ли установка Java ранее:
  `java -version`
  Если установка Java не выполнялась, вы увидите следующие результаты:

```
Output
Command 'java' not found, but can be installed with:

sudo apt install default-jre              # version 2:1.11-72, or
sudo apt install openjdk-11-jre-headless  # version 11.0.7+10-3ubuntu1
sudo apt install openjdk-13-jre-headless  # version 13.0.3+3-1ubuntu2
sudo apt install openjdk-14-jre-headless  # version 14.0.1+7-1ubuntu1
sudo apt install openjdk-8-jre-headless   # version 8u252-b09-1ubuntu1
```

Выполните следующую команду, чтобы установить по умолчанию среду Java Runtime Environment (JRE), которая установит JRE из OpenJDK 11:

`Sudo apt install default-jre`

JRE позволит вам запускать практически любое программное обеспечение Java. Проверьте установку с помощью следующей команды:

`java-version`

Вывод должен выглядеть следующим образом:

```
Output
openjdk version "11.0.7" 2020-04-14
OpenJDK Runtime Environment (build 11.0.7+10-post-Ubuntu-3ubuntu1)
OpenJDK 64-Bit Server VM (build 11.0.7+10-post-Ubuntu-3ubuntu1, mixed mode, sharing)
```

- **Doker**

Шаг 1 — Установка Docker

Первым делом обновите существующий список пакетов:

```
Sudo apt update
```

Затем установите несколько необходимых пакетов, которые позволяют apt использовать пакеты через HTTPS:

```
Sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Добавьте ключ GPG для официального репозитория Docker в вашу систему:

```
Curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
```

Добавьте репозиторий Docker в источники APT:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

Потом обновите базу данных пакетов и добавьте в нее пакеты Docker из недавно добавленного репозитория:

```
sudo apt update
```

Убедитесь, что установка будет выполняться из репозитория Docker, а не из репозитория Ubuntu по умолчанию:

```
apt-cache policy docker-ce
```

Вы должны получить следующий вывод, хотя номер версии Docker может отличаться:

Output of apt-cache policy docker-ce

```
docker-ce:
  Installed: (none)
  Candidate: 5:19.03.9~3-0~ubuntu-focal
  Version table:
     5:19.03.9~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
```

Обратите внимание, что docker-ce не установлен, но является кандидатом на установку из репозитория Docker для Ubuntu 20.04 (версия focal).

Установите Docker:

```
Sudo apt install docker-ce
```

Docker должен быть установлен, демон-процесс запущен, а для процесса активирован запуск при загрузке. Проверьте, что он запущен:

```
Sudo systemctl status docker
```

Вывод должен выглядеть примерно следующим образом, указывая, что служба активна и запущена:

```
Output
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2020-05-19 17:00:41 UTC; 17s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 24321 (dockerd)
      Tasks: 8
     Memory: 46.4M
     CGroup: /system.slice/docker.service
             └─24321 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

После установки Docker у вас будет доступ не только к службе Docker (демон-процесс), но и к утилите командной строки docker или клиенту Docker. Мы узнаем, как использовать команду docker позже в этом обучающем руководстве.

Шаг 2 — Настройка команды Docker без sudo

По умолчанию команда docker может быть запущена только пользователем root или пользователем из группы docker, которая автоматически создается при установке Docker. Если вы попытаетесь запустить команду docker без префикса sudo или с помощью пользователя, который не находится в группе docker, то получите следующий вывод:

```
Output
docker: Cannot connect to the Docker daemon. Is the docker daemon running on this host?.
See 'docker run --help'.
```

Если вы не хотите каждый раз вводить sudo при запуске команды docker, добавьте свое имя пользователя в группу docker:

```
Sudo usermod -aG docker ${USER}
```

Чтобы применить добавление нового члена группы, выйдите и войдите на сервер или введите следующее:

```
su - ${USER}
```

Вы должны будете ввести пароль вашего пользователя, чтобы продолжить. Проверьте, что ваш пользователь добавлен в группу docker, введя следующее:

```
id -nG
```

```
Output
sammy sudo docker
```

Если вам нужно добавить пользователя в группу docker, для которой вы не выполнили вход, объявите имя пользователя явно, используя следующую команду:

```
sudo usermod -aG docker username
```

- Docker-Compose
  Шаг 1 — Установка Docker Compose
  Чтобы получить самую последнюю стабильную версию Docker Compose, мы загрузим это программное обеспечение из официального репозитория Github .
  Для начала проверьте, какая последняя версия доступна на странице релизов. На момент написания настоящего документа наиболее актуальной стабильной версией является версия 1.28.6.
  Следующая команда загружает версию 1.28.6 и сохраняет исполняемый файл в каталоге `/usr/local/bin/docker-compose`, в результате чего данное программное обеспечение будет глобально доступно под именем docker-compose:

`sudo curl -L "https://github.com/docker/compose/releases/download/1.28.6/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose`

Затем необходимо задать правильные разрешения, чтобы сделать команду docker-compose исполняемой:

`sudo chmod +x /usr/local/bin/docker-compose`

Чтобы проверить успешность установки, запустите следующую команду:

`docker-compose --version`

Вывод будет выглядеть следующим образом:

```
Output

docker-compose version 1.26.0, build 8a1c60f6
```

Установка Docker Compose успешно выполнена.

3. **Развернуть (Deployment) doker-контейнеры с системой Gis-Porta**

1. Создать файл-переменных с названием "`init_variables.sh`" и положить рядом с `docker-compose.yml` содержимое которого:

```
#!/usr/bin/env bash
export JAVA_XMS_SIZE=4G
export JAVA_XMX_SIZE=6G
export APM_URL=http://${ELASTICSEARCH_HOST:-10.10.10.186}:8200
export ELK_IMAGE_TAG=${ELK_IMAGE_TAG:-7.4.2}
export ELASTICSEARCH_PASSWORD=${ELASTICSEARCH_PASSWORD:-changeFiz}
export ELASTICSEARCH_USERNAME=${ELASTICSEARCH_USERNAME:-elastic}
export ELASTICSEARCH_HOST=${ELASTICSEARCH_HOST:-10.10.10.186}
export KIBANA_HOST=${ELASTICSEARCH_HOST:-10.10.10.186}
export LOGSTASH_HOST=${ELASTICSEARCH_HOST:-10.10.10.186}
export DOCKER_CRG_HOST=${DOCKER_CRG_HOST:-cr.yandex/crp7o80lcrqlf17up9fq}
export CRG_USER=${CRG_USER:-fiz}
export RABBIT_PASS=${RABBIT_PASS:-314}
export DB_PASS=${DB_PASS:-314}
export GEOSERVER_USER=${GEOSERVER_USER:-admin@mail.ru}
export GEOSERVER_PASSWORD=${GEOSERVER_PASSWORD:-geoserver}
export RABBIT_TAG=${RABBIT_TAG:-management}
export POSTGRES_TAG=${POSTGRES_TAG:-11.2}
export POSTGIS_TAG=${POSTGIS_TAG:-11.7-2.5}
export IS_DATA_EXIST=${IS_DATA_EXIST:-}
export GEOSERVER_TAG=${GEOSERVER_TAG:-2.16.0}
export CRG_INTEGRATION_SERVICE_TAG=3130
export CRG_DATA_SERVICE_TAG=3130
export CRG_REGISTRY_TAG=3130
export CRG_AUTH_TAG=3130
export CRG_GIS_SERVICE_TAG=3130
export CRG_GATEWAY_TAG=3130
export CRG_API_TAG=3130
export CRG_WRAPPER_TAG=3130
export CRG_UI_TAG=3130
export UI_PLATFORM=simf
export UI_PROD=false
export UI_SERVER_HOST=
export UI_SERVER_PORT=
export UI_SWN=scratch_database
export UI_WS_PORT=
export UI_LOGO=/assets/logo/name/logo.png
export UI_FAVICON=/assets/logo/name/logo.png
export UI_HTTPS=1
export IS_DATA_EXIST=true
export UI_SEND_ERRORS_TO_TG_HTTP=1
export UI_SEND_ERRORS_TO_TG_HTTPS=1
export UI_SUPRESS_TOAST_ERRORS_HTTP=1
export UI_SUPRESS_TOAST_ERRORS_HTTPS=1
echo Init migrations
export GEOSERVER_DATA_DIR=${GEOSERVER_DATA_DIR:-/opt/data/geoserver}
```

```
export DB_DATA_DIR=${DB_DATA_DIR:-/opt/data/postgres}
```

где ***3130*** - актуальная версия релиза

2. Создать файл запуска с названием "`install.sh`" и положить рядом с `docker-compose.yml` содержимое которого:

```
#!/bin/bash
echo Copy compose file
ls -la
scp -P 23 docker-compose.yml user@212.110.158.218:/tmp
scp -P 23 docker-compose-monitoring.yml user@212.110.158.213:/tmp
ssh -p 23 user@212.110.158.218 'docker login --username oauth --password
AQAAAAAHmwGEAATuwbGb5Su7gEmCqL3jgP2qngA cr.yandex'
ssh -p 23 user@212.110.158.218 'docker image prune --all --force --filter "dangling=true"'
chmod +x init_variables.sh
echo pack this file 'serviceman do not have permission to copy .sh files to server'
tar -cf init_vars.tar init_variables.sh
echo copy archive
scp -P 23 init_vars.tar user@212.110.158.218:/tmp
ssh -p 23 user@212.110.158.218 'cd /tmp; tar -xf init_vars.tar; . ./init_variables.sh'
ssh -p 23 user@212.110.158.218 'cd /tmp; . ./init_variables.sh; docker-compose -f docker-
compose.yml down'
echo Clean assets
ssh -p 23 user@212.110.158.218 'cd /tmp; rm -rf assets;'
echo Create new assets
tar -cf configs.tar initialConfig
tar -cf scripts.tar migration-scripts
scp -P 23 configs.tar scripts.tar user@212.110.158.213:/tmp
ssh -p 23 user@212.110.158.218 'cd /tmp; mkdir assets; tar -xf configs.tar -C assets/'
ssh -p 23 user@212.110.158.21 'cd /tmp; tar -xf scripts.tar -C assets/'
echo Run migrations
ssh -p 23 user@212.110.158.218 'find /tmp/assets/migration-scripts/ -name '*.sh' -exec chmod
+x '{}' \;'
ssh -t -p 23 user@212.110.158.218 'cd /tmp; . ./init_variables.sh; cd /tmp/assets/;
./migration-scripts/run.sh'
echo UP docker-compose
ssh -p 23 user@212.110.158.218 'cd /tmp; . ./init_variables.sh; docker-compose -f docker-
compose.yml up -d'
```

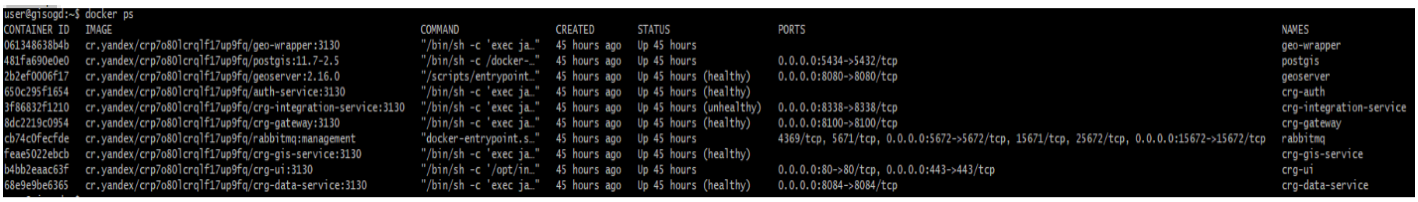
где ***212.110.158.218*** - ip-адрес VM на которой разворачивается система

a) Положить файлы сертификатов SSL ***(cert.crt, cert.key)*** в папку на сервере `/opt/ssl`

b) Запустить ***`install.sh`*** с локального компьютера через командную строку

c) Проверить что контейнеры поднялись и запущены

```
docker ps
```



user@gisogd:~$ docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
061348638b4b	cr.yandex/crp7o80lcrqlf17up9fq/geo-wrapper:3130	"/bin/sh -c 'exec ja…"	45 hours ago	Up 45 hours		geo-wrapper
481fa690e0e0	cr.yandex/crp7o80lcrqlf17up9fq/postgis:11.7-2.5	"/bin/sh -c /docker-…"	45 hours ago	Up 45 hours	0.0.0.0:5434->5432/tcp	postgis
2b2ef0006f17	cr.yandex/crp7o80lcrqlf17up9fq/geoserver:2.16.0	"/scripts/entrypoint…"	45 hours ago	Up 45 hours (healthy)	0.0.0.0:8080->8080/tcp	geoserver
650c295f1654	cr.yandex/crp7o80lcrqlf17up9fq/auth-service:3130	"/bin/sh -c 'exec ja…"	45 hours ago	Up 45 hours (healthy)		crg-auth
3f86832f1210	cr.yandex/crp7o80lcrqlf17up9fq/crg-integration-service:3130	"/bin/sh -c 'exec ja…"	45 hours ago	Up 45 hours (unhealthy)	0.0.0.0:8338->8338/tcp	crg-integration-service
8dc2219c0954	cr.yandex/crp7o80lcrqlf17up9fq/crg-gateway:3130	"/bin/sh -c 'exec ja…"	45 hours ago	Up 45 hours (healthy)	0.0.0.0:8100->8100/tcp	crg-gateway
cb74c0fecfde	cr.yandex/crp7o80lcrqlf17up9fq/rabbitmq:management	"docker-entrypoint.s…"	45 hours ago	Up 45 hours	4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, 15671/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp	rabbitmq
feae5022ebcb	cr.yandex/crp7o80lcrqlf17up9fq/crg-gis-service:3130	"/bin/sh -c 'exec ja…"	45 hours ago	Up 45 hours (healthy)		crg-gis-service
b4bb2eaac63f	cr.yandex/crp7o80lcrqlf17up9fq/crg-ui:3130	"/bin/sh -c '/opt/in…"	45 hours ago	Up 45 hours	0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp	crg-ui
68e9e9be6365	cr.yandex/crp7o80lcrqlf17up9fq/crg-data-service:3130	"/bin/sh -c 'exec ja…"	45 hours ago	Up 45 hours (healthy)	0.0.0.0:8084->8084/tcp	crg-data-service

где ***healthy/unhealthy*** - статус контейнера.

3. Восстановить данные из full backup

- БД

a) Скопировать и распаковать нужный файл бекапа в примонтированную контейнером папку, в нашем случае в `/opt/data/postgis` командой типа

```
cp /mnt/data/backup/DataBase_backup/pg_dumpall_Name_${datetime}.sql.gz -C /opt/data/postgis/
gunzip /opt/data/postgis/pg_dumpall_Name_${datetime}.sql.gz
```

b) Через pgAdmin выполнить команды в БД "postgrsql":

```
UPDATE pg_database SET datallowconn = 'false' WHERE datname = 'database_1;
```

```
SELECT pg_terminate_backend(pg_stat_activity.pid) FROM pg_stat_activity WHERE pg_stat_activity.datname = 'database_1' AND pid <> pg_backend_pid();
```

где database_1 – выбранная БД или выполнить bash-команду:

```
docker exec -it postgis psql --dbname=postgresql://fiz:314@192.168.0.102:5434/postgres -c \"UPDATE pg_database SET datallowconn = 'false' WHERE datname = 'database_1; SELECT pg_terminate_backend(pg_stat_activity.pid) FROM pg_stat_activity WHERE pg_stat_activity.datname = 'database_1' AND pid <> pg_backend_pid();
```

где: kpt – выбранная БД.

Это требуется выполнить со всеми БД, кроме "postgres"

c) Через pgAdmin выполнить команду (удаление БД) в БД "postgres":

```
dropdb database_1;
```

где database_1 – выбранная БД или выполнить bash-команду:

```
sudo docker exec -u postgres postgis /usr/bin/dropdb database_1
```

где database_1 - выбранная БД

Это требуется выполнить со всеми БД, кроме "postgres"

По итогу у вас останется только одна БД "postgres"

d) Выполнить bash-команду восстановления backup-файла:

```
sudo docker exec -u postgres postgis /usr/bin/pg_restore -C -d postgres /var/lib/postgresql/pg_dumpall_name_${datetime}.sql
```

e) Geoserver

Распаковать файлы из архива tar backup-файла в папку `/opt/data/geoserver`

4. Монтирование папок: подложки, прикрепленные файлы, регламенты, растры

В файле **`/etc/fstab`** прописать точки монтирования

```
nano /etc/fstab
```

Содержимое (добавить):

```
/dev/sdb1 /mnt/data ext4 defaults 0 0
/mnt/data/SpatialDataGeoserver /opt/spatial_data none bind 0 0
/mnt/data/file_storage /opt/file_storage none bind 0 0
/mnt/data/GeoWebCache /opt/gwc_storage none bind 0 0
/mnt/data/SpatialDataGeoserver/R_OFP_80cm_Pulkovo_1963_z6_Crimea
/opt/data/geoserver/spatial_data_u4 none bind 0 0
/mnt/data/SpatialDataGeoserver/R_OFP_80cm_Pulkovo_1963_z4_5_Crimea
/opt/data/geoserver/uploads none bind 0 0
/mnt/data/reglaments /opt/reglaments none bind 0 0
```

5. Перезапустить контейнер geoserver

```
docker restart geoserver
```

6. Настроить (БД, geoserver, прикрепленные файлы, растры)

Задаем в crontab выполнение скриптов backup

```
sudo crontab -e
```

Содержимое (добавить):

```
0 0 * * * /mnt/data/scripts/backup_Name_DB.sh >/dev/null 2>&1 #Каждый день в полночь 00:00
0 0 * * * find /mnt/data/backup/DataBase_backup -type f -mtime +31 -print0 | xargs -0 rm -f Проверка и удаление файлов старше 31 дня
0 0 * * * /mnt/data/scripts/backup_Name_geoserver.sh >/dev/null 2>&1 #Каждый день в полночь 00:00
0 0 * * * find /mnt/data/backup/geoserver_backup -type f -mtime +31 -print0 | xargs -0 rm -f Проверка и удаление файлов старше 31 дня
0 0 * * * /mnt/data/scripts/backup_Name_FS.sh >/dev/null 2>&1 #Каждый день в полночь 00:00
0 0 * * * find /mnt/data/backup/file_storage_backup -type d -mtime +31 -print0 | xargs -0 rm -f Проверка и удаление папок старше 31 дня
0 0 * * * /mnt/data/scripts/backup_Name_spatialData.sh >/dev/null 2>&1 #Каждый день в полночь 00:00
```

```
0 0 * * * find /mnt/data/backup/spatial_data_backup/rasters -type d -mtime +31 -print0 | xargs -0 rm -f Проверка и удаление папок старше 31 дня
```

7. Данная инструкция принимает факт монтирования раздела с данными в `/mnt/data`

Содержимое раздела:

`-GeoWebCache -` кэш созданный geoserver

`-SpatialDataGeoserver` - пространственные данные

`-backup -` хранилище бекапов

`-file_storage` - файловой хранилище (прикрепленные документы к слоям)

`-reglaments` - файлы регламентов

`-scripts` - скрипты автоматизации