Подготовка доступа и работа ansible скриптов.

Для быстрой установки я подготовил скрипты автоматического развёртывания. Что бы ими было удобнее пользоваться готовлю config ssh:
~/.ssh/config

```
host *
        ForwardX11 yes
        Compression yes
        ServerAliveInterval 3
        ServerAliveCountMax 3
        ForwardAgent yes
# not ask permanent addede
        StrictHostKeyChecking no
        UserKnownHostsFile=/dev/null


host vega
        Hostname 116.202.113.188
        User root

host fobos
        Hostname 116.202.113.190
        User root

host zabbix
        Hostname 78.47.186.252
        User root

host b-nginx1
        Hostname 188.120.229.195
        User root

host b-nginx2
        Hostname 188.120.228.220
        user root

host zeus
        Hostname 116.202.131.197
        User root


Host old-prod
        Hostname burjauto.com
        Port 2022
        User root
```

Теперь, что бы получить доступ к любому из серверов достаточно ввести ssh zeus/vega/fobos etc.

Файл серверов ansible: inventory.yaml
Сервера группы web-servers - Nginx точки входа. Сервера проксируют запросы от пользователей, могут отдавать им картинки. На них нет никакой базы данных, соответственно и нагрузки тоже.
Сервера группы web-servers - основные рабочие сервера. Ни них работают базы данных, web-сервера и остальные, сопутствующие программы.

Сервера b-nginx1 и b-nginx2
Подготавливаются с помощью ansible. После подготовки нужно обратить внимание на следующие файлы:
/etc/nginx/nginx.conf, секция upstream backend_host {} В этом месте указываются ip адреса проксируемых серверов (в нашем случае vega и fobos)

Для добавления дополнительных обслуживаемых доменных имён:
- копируем файл /etc/nginx/sites-avaible/burjauto.conf
- меняем содержимое секций server
- получаем сертификат https (об этом ниже)
- делаем симлинк этого файла в папку /etc/nginx/sites-avaible/[file-name].conf
- проверяем правильность конфигурации nginx -t
- готовим файл на проксируемых серверах (см. описание в другой части)
- перезапускаем nginx

Работа с сертификатами, certbot
После установки certbot необходимо инициализировать:
certbot register --email ваш@адрес-для-напоминаний

Проверяем возможность получения сертификата

certbot certonly --dry-run -d example.com

Если выводит The dry run was successful - то всё норм, можно получать сертификат

certbot certonly --dry-run -d example.com

Проверка всех доступных сертификатов:
cat /etc/letsencrypt/live/*/cert.pem | openssl x509 -text | grep -o 'DNS:[^,]*' | cut -f2 -d:
так понятнее:
find /etc/letsencrypt/live/ -type l

полученные сертификаты необходимо добавить в nginx файл, в секцию с сертификатами.

После этого добавить в cron: 0 1 0 */1 * crontab renew - обновление раз в месяц.
ВАЖНО! Можно поставить обновление раз в неделю, для того, что certbot может получить ip адрес другого сервера и отказаться продлевать сертификат, за 1 месяц таких попытки будет 4 + предупреждение на электронную почту о истекающем сертификате для ручного вмешательства.

Сервер zeus
Это основной рабочий сервер, базовая ansible настройка этого сервера не отличается от vega или fobos. После его подготовки необходимо провести следующие работы:

База данных:
файл конфигурации (/etc/my.cnf имеет большое количество комментариев):
server-id = 11 - это уникальное число для реплики, в кластере это число не должно пересекаться
binlog_format = ROW - позволяет полностью убрать нагрузку на slave сервера, так как отдаёт в binlog файл переработанную информацию, которые slave просто записывают в таблицы без расчётов
slave_skip_errors - иногда из за этих ошибок может разваливаться реплика. С чем связана каждая конкретная ошибка нужно смотреть в документации по базе. В конфиге есть список, который я использовал в других проектах. В основном реплика разваливалась при изменениях в базе данных связки master-master.
Для создания реплики необходимо провести следующие действия
на мастере нужно добавить пользователей для репликации:

GRANT replication slave ON *.* TO "replica_db_vega"@"116.202.113.188" IDENTIFIED BY "so03solijf-09djs";

GRANT replication slave ON *.* TO "replica_db_fobos"@"116.202.113.190" IDENTIFIED BY "so03solijf-09djs";

FLUSH PRIVILEGES;

блокируем базу данных на запись:

FLUSH TABLES WITH READ LOCK; SET GLOBAL read_only = ON;

Выводим состояние базы данных:

show master status\G

Результат будет примерно таким:

File: mysql-bin.000005
Position: 404
Binlog_Do_DB:

Binlog_Ignore_DB: information_schema,mysql
Нас будут интересовать параметры file и position, из необходимо запомнить или записать. После этого из mysql нужно выйти и сделать копию баз(ы) для slave. После того, как копия будет готова базу(ы) данных необходимо разлочить и разрешить запись:

SET GLOBAL read_only = OFF;

Затем базу данных необходимо перенести на slave сервера (в нашем случае vega и fobos) и развернуть там. (настройка и включение slave в секции про slave)

Разумеется для прямого доступа к базе необходим свой пользователь, процедура известная, поэтому приводить её буду только командами:
CREATE USER 'local_db_user'@'localhost' IDENTIFIED BY 'Od02jflskfw0-3jfasl';
GRANT USAGE ON * . * TO 'local_db_user'@'localhost' IDENTIFIED BY 'Od02jflskfw0-3jfasl' ;
GRANT ALL PRIVILEGES ON * . * TO 'local_db_user'@'localhost' IDENTIFIED BY 'Od02jflskfw0-3jfasl' WITH GRANT OPTION;
FLUSH PRIVILEGES;

Все команды можно посмотреть в history mysql

Настройка lsync
Программа lsync предназначена для скоростной синхронизации файлов между серверами. Она работает на rsync, но не так грузит процессор и отслеживает только состояние файлов в указанных дирректориях.
Файл конфига расположен /etc/lsync.conf. Он достаточно прост и не требует больших пояснений
Для синхронизации так же используется файл /root/.ssh/config, ключи без парольной защиты, для серверной работы это допустимо.
Синхронизация происходит при старте демона *systemctl start lsyncd* в течении некоторого времени. После синхронизация происходит мгновенно.
Благодаря lsync отпадает необходимость работать на серверах vega или fobos, за ними просто нужно следить. Все что вы сделаете на zeus в папке проекта - синхронизируется автоматически.

Сервера fobos и vega
Это основные сервера для работы пользователей. Они не занимаются никакими действиями с базой данных, а только читают данные с локальной базы. База в таком режиме не блокируется, таким образом достигается высокая доступность и скорость отклика сайта. Их настройка производится с помощью ansible, настройка базы данных (/etc/my.cnf):
 server-id = 21 - это уникальное число для реплики, в кластере это число не должно пересекаться.

Для создания slave необходимо:
- развернуть базу, полученную с мастера
- подготовить пользователей для чтения и для реплики:

change master to master_host = "116.202.131.197", master_user = "replica_db_vega", master_password = "so03solijf-09djs", master_log_file = "mysql-bin.000001", master_log_pos = 775;
(Данные master_log_file и master_log_pos берём с сохранённых данных с мастера!)

start slave;

SHOW SLAVE STATUS\G

в выводе этой команды необходимо смотреть на строчку
Seconds_Behind_Master: 0

Это отставание слейва от мастера в секундах, 0 - говорит об отсутствии отставания

Создания пользователя в режиме чтения:

CREATE USER 'loc_db_vega'@'localhost' IDENTIFIED BY 'Kdo9w4usfdljs09';
GRANT SELECT * . * TO 'loc_db_vega'@'localhost' IDENTIFIED BY 'Kdo9w4usfdljs09';
GRANT SELECT ON TO 'loc_db_vega'@'localhost' IDENTIFIED BY 'Kdo9w4usfdljs09';
FLUSH PRIVILEGES;

Этот пользователь прописывается в конфиге сайта для чтения, для записи используется пользователь с мастера и на мастере

Настройка nginx на проксируемых серверах
 # В идеале лучше бы закрыть доступ на 80 и 443 порты на эти сервера, для того чтобы нельзя было достучаться на эти сервера ниоткуда, кроме как с nginx проксируемых серверов, но это вам решать, у меня для этого всё готово. Разве что можно добавить офис, для оперативного доступа используя свой DNS

ВАЖНО! Этот сервер работает и настраивается только на 80 порту, сертификат на него не предустанавливается и не используется. Это так же снимает нагрузку на процессор, так как ssl сессии сильно нагружают процессор. Делать это дважды (первый раз на входящем nginx сервере) смысла нет.

Конфиг nginx я использовал ваш, так что он вам максимально знаком, разве что я его облегчил и перенёс часть дублирующих настроек в основной /etc/nginx/nginx.conf

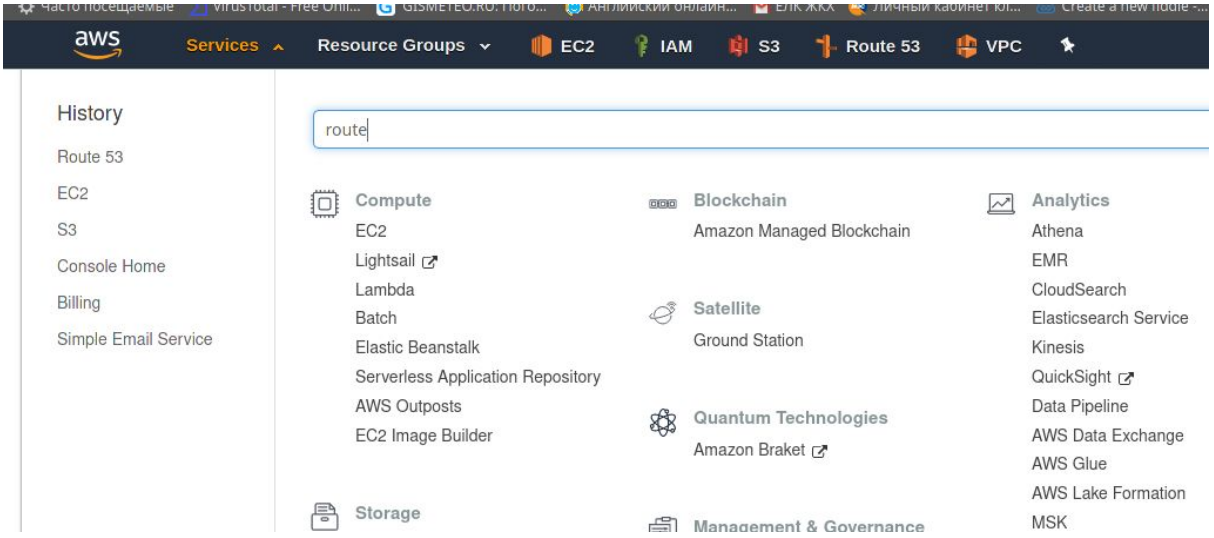
Ещё раз хочу подчеркнуть - настройка делается с помощью ansible, все шаги, пакеты, конфиги указаны там, всё подписано и проверено на серверах centos и

*redhat от Amazon. При использовании собственных серверов (например вы захотите посмотреть как это работает на виртуальных машинах Centos не забудьте отключить **Selinux**, иначе много чего не будет работать, я его не использую в повседневной работе, поэтому и отключаю, потому что не знаю)*

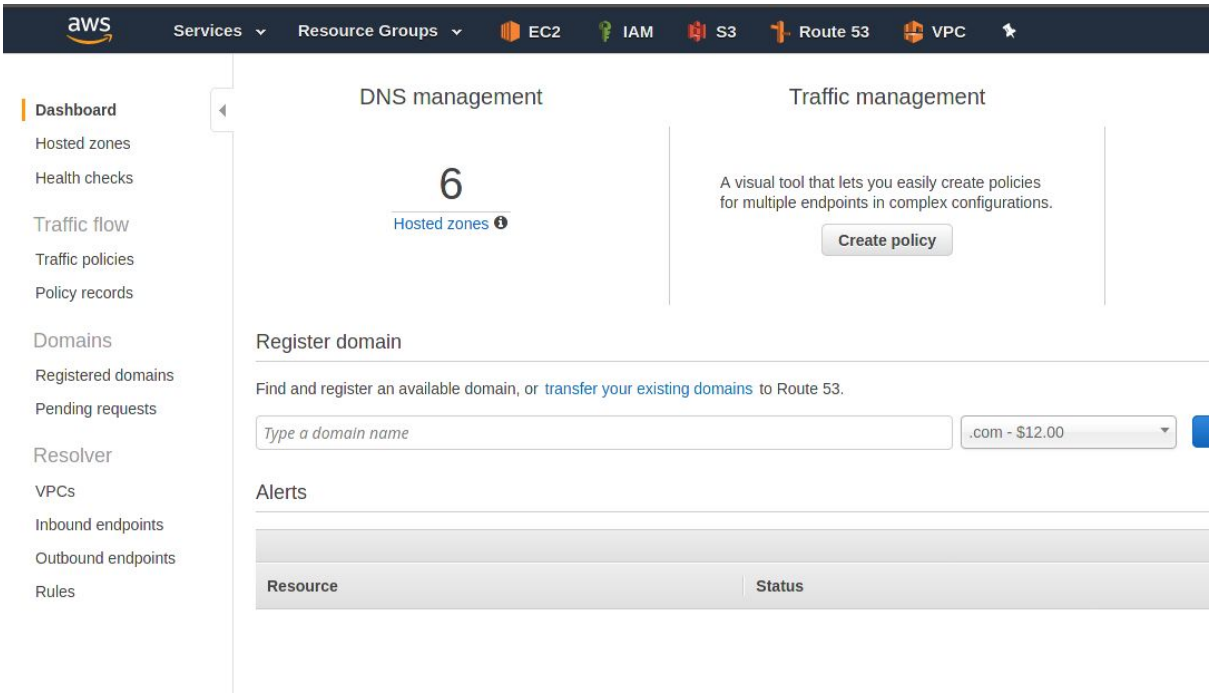
AWS ROUTE 53

Для достижения высокой стабильности и увеличения доступности серверов в качестве DNS используется Route53 от Amazon

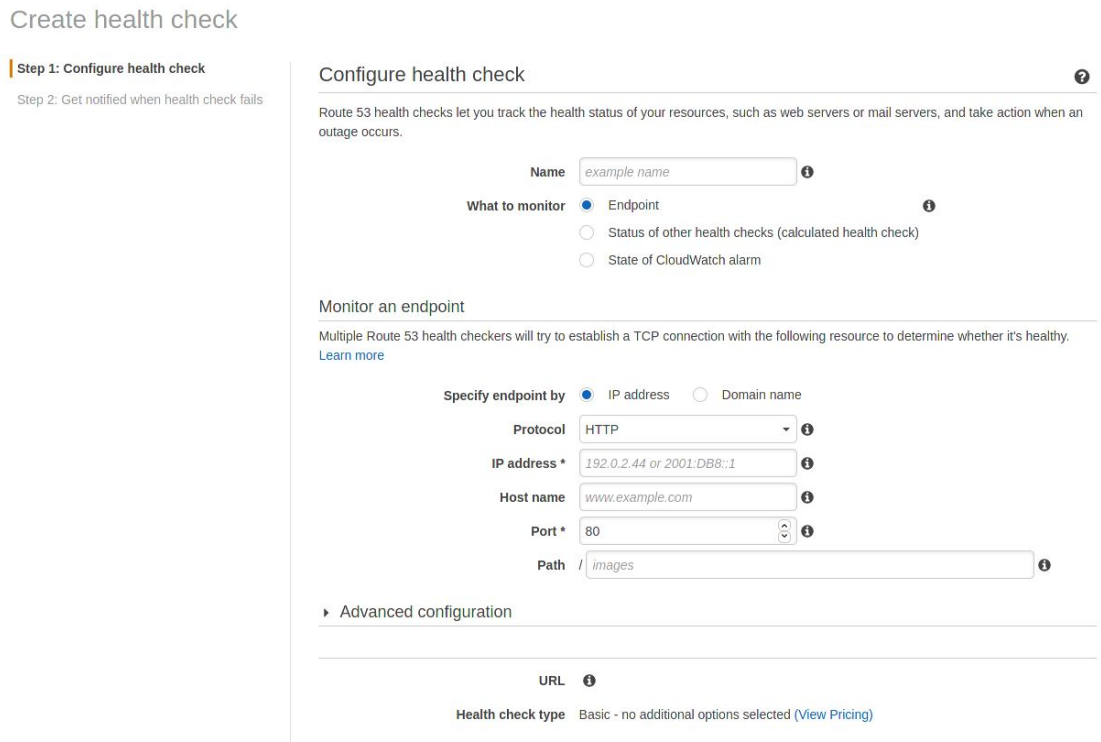
Настройка его производится следующим образом:
в Панели Console нажимаем на Service и вводим Route53



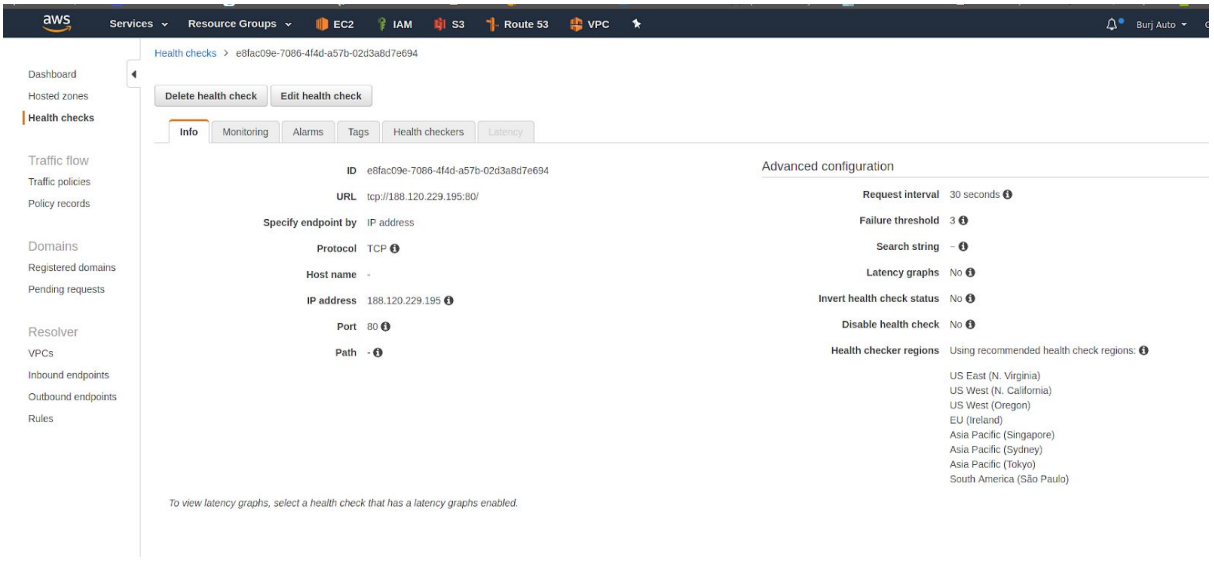
- Выбираем Health check



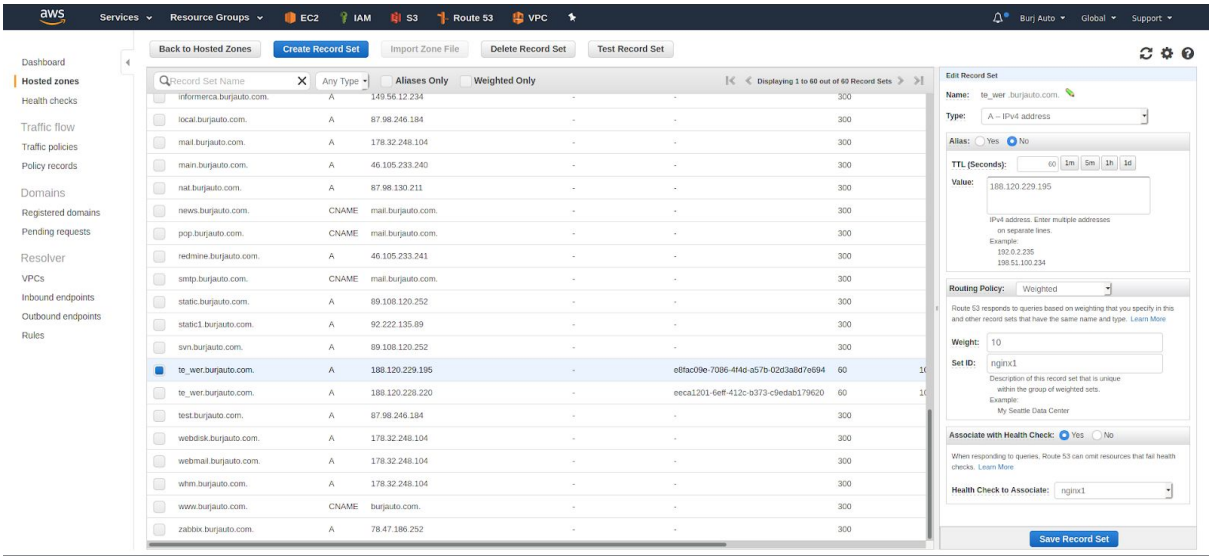
Тут можно создать новый health check



Указываем ip, port, возможно проверять наличие какой то картинки, файла. Я проверяю просто порт на ip - доступен 80 или нет. Если сервер отключится, то амазон автоматически исключит его из раздачи dns в течении 30 секунд



После настройки заходим в нужную зону, где необходимо добавить новый адрес (использую com для примера)
Создаём новую или редактируем старую запись:



Указываем тип записи A, ip адрес, выбираем параметр Weighted, указываем произвольное, понятное для нас имя и вес (вес это абстрактная величина, максимально близкое понятие - % выдачи ip этого сервера при запросе, так как сервера у нас равнозначные я указал на каждом число 10, чтобы каждый сервер получал равномерную нагрузку)
И после этого привязываем запись в Health check, для корректной работы сервиса.

Вопрос касательно сессий пользователей и ботов:
Обычно для этого используют redis и сессии хранят в них, но это уже решается на уровне проекта.