Dockerfile

```
FROM ubuntu:16.04

MAINTAINER Kolotovkin Maxim

RUN apt-get -y update
ENV PGVER 9.5
RUN apt-get install -y postgresql-$PGVER

# postgres
USER postgres
RUN /etc/init.d/postgresql start &&\
    psql --command "ALTER USER postgres WITH SUPERUSER PASSWORD '12345';" &&\
    createdb -E utf8 -T template0 -O postgres my_database &&\
    /etc/init.d/postgresql stop

RUN echo "synchronous_commit = off" >> /etc/postgresql/$PGVER/main/postgresql.conf
RUN echo "fsync = off" >> /etc/postgresql/$PGVER/main/postgresql.conf

EXPOSE 5432

VOLUME  ["/etc/postgresql", "/var/log/postgresql", "/var/lib/postgresql"]

# root
USER root
RUN apt-get -y update
RUN apt-get install -y nodejs
RUN apt-get install -y npm
RUN apt-get install -y nodejs-legacy

# copy app to container
ENV APP /root/app
ADD ./ $APP

# go to app
WORKDIR $APP

# node proj manager
RUN npm install forever -g

RUN npm install

EXPOSE 80

RUN apt-get update && apt-get install -y openssh-server
RUN mkdir /var/run/sshd
RUN echo 'root:hackathon' | chpasswd
RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/'
/etc/ssh/sshd_config

RUN sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g' -
i /etc/pam.d/sshd

ENV NOTVISIBLE "in users profile"
RUN echo "export VISIBLE=now" >> /etc/profile

EXPOSE 22

# go to app
WORKDIR $APP

# start postgres          forever start node    sshd
CMD service postgresql start && forever start index.js && /usr/sbin/sshd -D
```

Собираем и запускаем контейнер

```
sudo docker build -t hack_nptest .
sudo docker run -d -P --name hack_nptest hack_nptest
```

```
Определяем соответствие реальным портам
sudo docker port hack_nptest
      22/tcp -> 0.0.0.0:32776
      5432/tcp -> 0.0.0.0:32774
      80/tcp -> 0.0.0.0:32775

Подключаемся по ssh
ssh root@172.17.0.1 -p 32776

Тестируем ноду в браузере http://172.17.0.1:32775/
HELLO NODE API
```

Проверим работу node в мэнэджере forever

```
root@48958c2527b2:~# forever list
info:    Forever processes running
data:        uid  command         script   forever pid id logfile
uptime
data:    [0] iiut /usr/bin/nodejs index.js 48      54
/root/.forever/iiut.log 0:0:2:37.967
```

Проверим как заполнилась БД

```
Меняем юзера на постгре
root@a1eb5093b309:~/app# su - postgres

Входим в режим работы с psql
postgres@a1eb5093b309:~$ psql
psql (9.5.14)
Type "help" for help.

postgres=# \l
                                  List of databases
    Name     |  Owner   | Encoding  | Collate | Ctype |   Access privileges
-------------+----------+-----------+---------+-------+-----------------------
 my_database | postgres | UTF8      | C       | C     |
 postgres    | postgres | SQL_ASCII | C       | C     |
 template0   | postgres | SQL_ASCII | C       | C     | =c/postgres          +
             |          |           |         |       | postgres=CTc/postgres
 template1   | postgres | SQL_ASCII | C       | C     | =c/postgres          +
             |          |           |         |       | postgres=CTc/postgres
(4 rows)

postgres=# \c my_database
You are now connected to database "my_database" as user "postgres".

my_database=# \dt
         List of relations
 Schema |  Name  | Type  |  Owner
--------+--------+-------+----------
 public | people | table | postgres
(1 row)

my_database=# select * from people;
 man_id | man_nickname | man_age
--------+--------------+---------
      1 | Alice        |      21
      2 | Bob          |      21
(2 rows)
```

Для выхода в root

```
my_database=# \q
postgres@b639e9351a69:~$ su - root
```

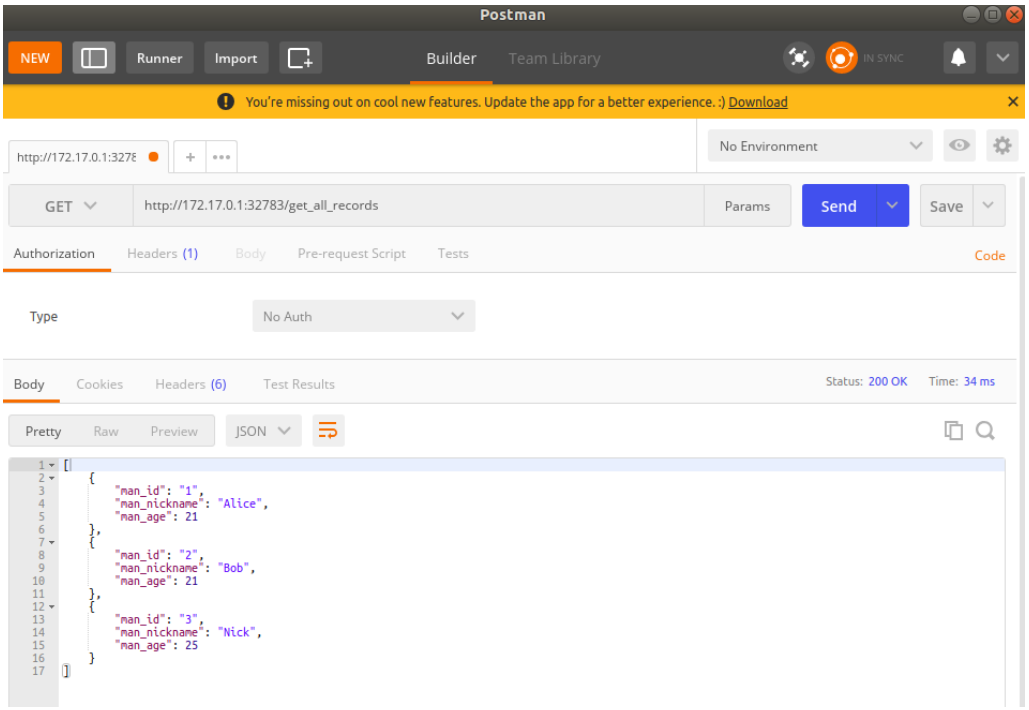
Тестируем результат

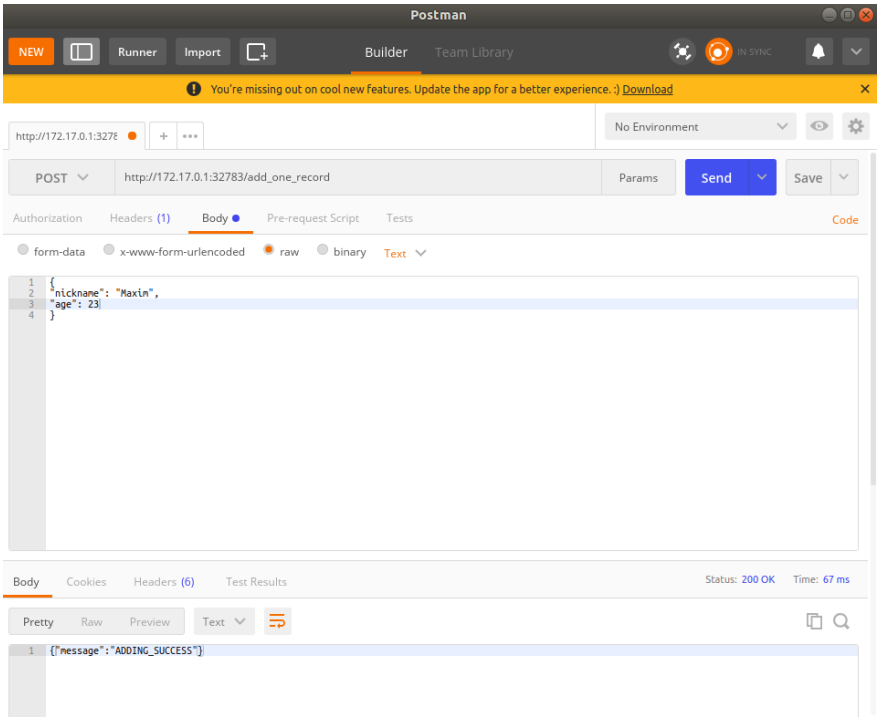
Проверка в браузере

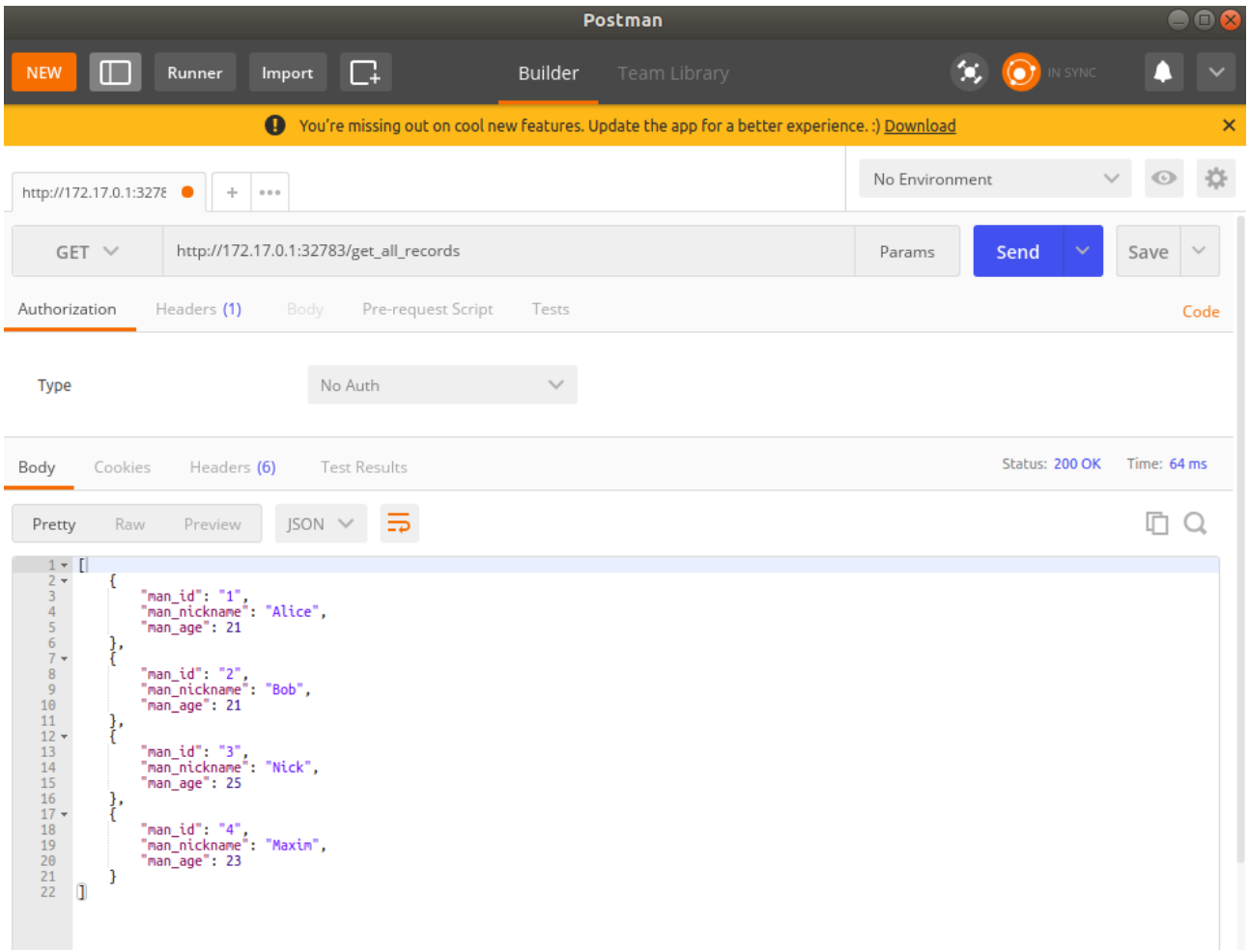
http://172.17.0.1:32786/

HELLO NODE API

Postman







Публикуем

```
$ sudo docker stop hack_nptest

$ ./connectICP.sh

$ cd hack_nptest/

$ sudo docker tag hack_nptest_3 icp.bmstu.ru:8500/team99/hack_node_api

$ sudo docker push icp.bmstu.ru:8500/team99/hack_node_api
```

Проверим, что образ появился на сервере

```
$ kubectl get images
```

Далее уточним активный namespace:
kubectl config current-context

Если не установлен нужный нам namespace=ваш логин, установим его:
kubectl config set-context my-context —namespace=team99

Теперь мы можем развернуть образ на кластере ICP:
$
kubectl run team99nodeapi --image=icp.bmstu.ru:8500/team99/hack_node_api

```
deployment "team99nodeapi" created
```
Проверим выполнение команды
```
$ kubectl get pods
NAME                              READY     STATUS    RESTARTS   AGE
team99nodeapi-747c88c578-tvtxh    1/1       Running   0          1m
```

Далее необходимо разрешить форвардинг портов к нашему Поду. Для этого используем команду expose.

Обратите внимание, нужно явно указывать уникальные имена портов:

```
$ kubectl expose deployment team99nodeapi --name=ssh99nodeapi --
type=LoadBalancer --port=22 –target-port=22

service "ssh99nodeapi" exposed

$ kubectl expose deployment team99nodeapi --name=http99nodeapi --
type=LoadBalancer --port=80 –target-port=80

service "http99nodeapi" exposed
```

Осталось узнать порты, которые использовал Kubernetes для форвардинга в порты 22 и 80 нашего Пода:

```
$ kubectl get svc
NAME            TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)

http            LoadBalancer   10.0.0.209    <pending>     80:31290/TCP

http99nodeapi   LoadBalancer   10.0.0.63     <pending>     80:30202/TCP

ssh             LoadBalancer   10.0.0.46     <pending>     22:32485/TCP

ssh99nodeapi    LoadBalancer   10.0.0.186    <pending>     22:30562/TCP

team99ubuntu    LoadBalancer   10.0.0.230    <pending>     22:32530/TCP
```

Нам был выделен порт 31290 под hack_nodetest http. Теперь мы можем обратиться к нашему контейнеру.

По http: http://195.19.40.201:30202/
По ssh: ssh root@195.19.40.201 -p 30562

В браузере
http://195.19.40.201:30202/