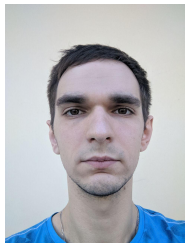
EPAM University Program 2020

Python Web App Hosted with AWS ECS



ZHUK ROMAN

Agenda

1. Create application.
2. Dockerize my app.
3. Use Jenkins for creating docker image on every push.
4. Host app with Amazon Web Services (ECS, RDS, S3, ELB, Auto Scaling).

Links

zhuk-roman/tabs

http://app.mit.pp.ua/

Building Application

It will be a simple bookmark application.

App will allow users to:

- Register and login to account
- Create new bookmarks and modify existing ones
- View list of bookmarks

App will use AWS RDS with MYSQL 8 database, AWS S3 bucket to store downloaded favicons of websites and AWS ECS to host app inside containers.

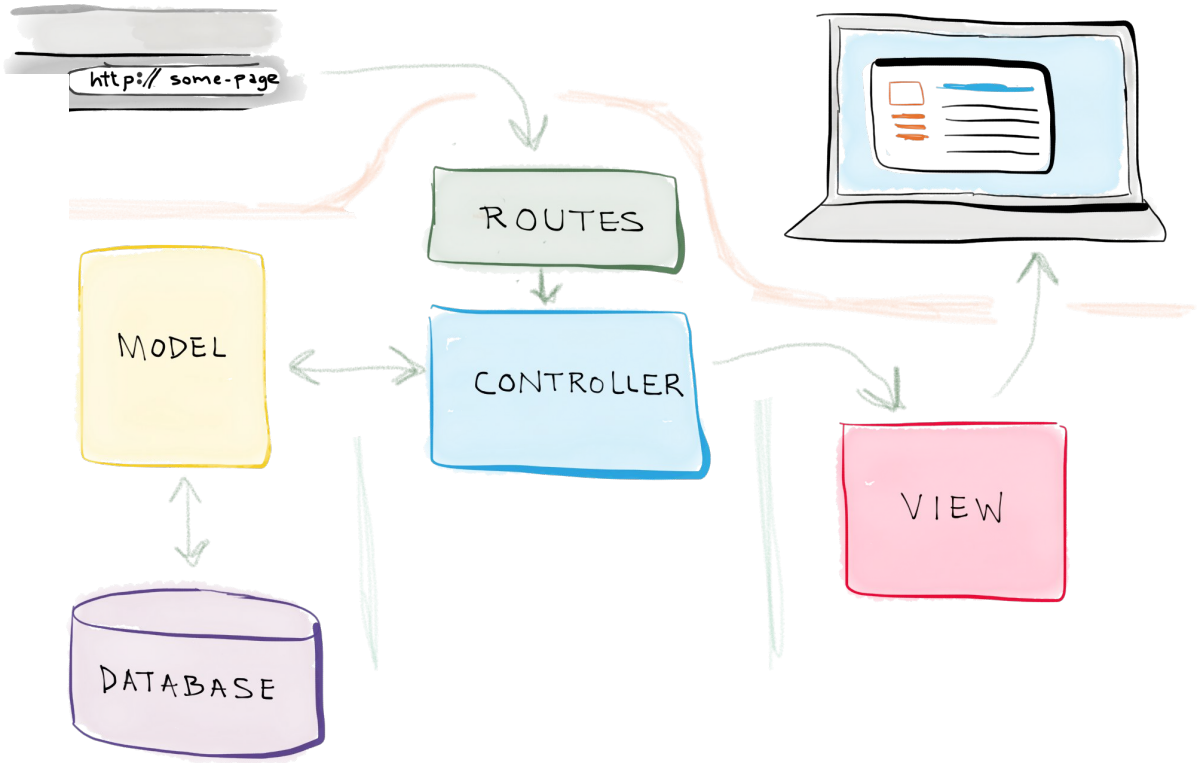
Python Packages

- **Flask** is a micro web framework written in Python. It is classified as a microframework. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.
- **Boto3** is the Amazon Web Services (AWS) SDK for Python. It enables Python developers to create, configure, and manage AWS services, such as EC2 and S3.
- **Flask-SQLAlchemy** is an extension for Flask that adds support for SQLAlchemy.
- **Flask-Login** provides user session management for Flask. It handles the common tasks of logging in, logging out, and remembering users sessions.
- **Flask-Bcrypt** is a Flask extension that provides bcrypt hashing utilities.
- **Flask-WTF** is a simple integration of WTForms for Flask.

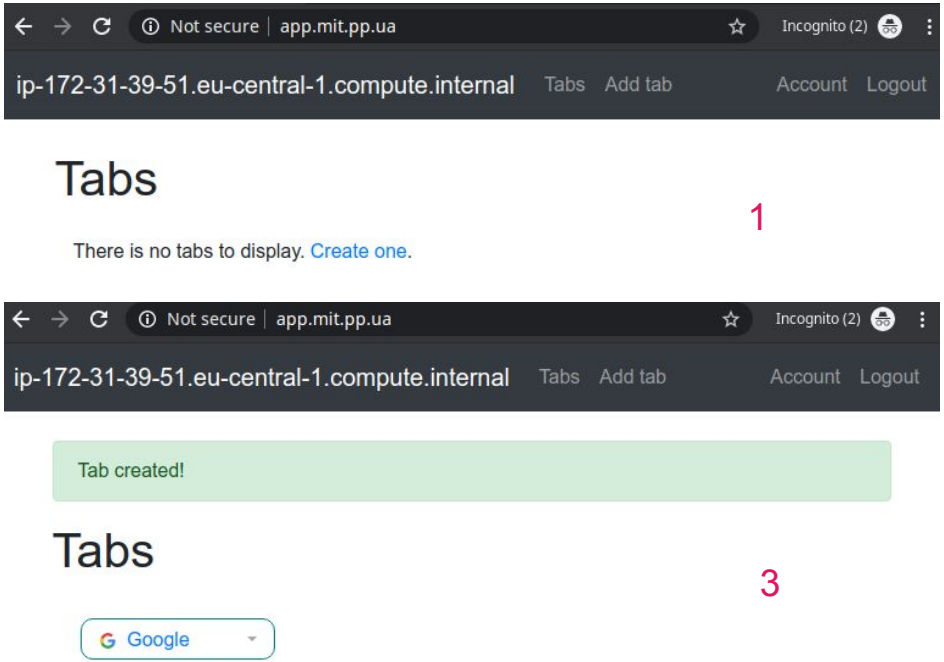
Also following packages were used: requests, os, favicon, random, datetime, bs4.

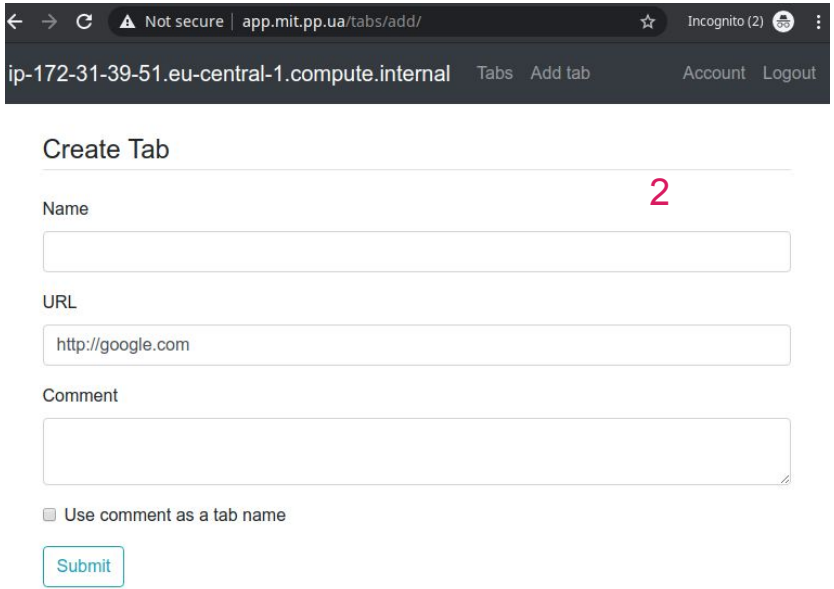
MVC

```
.
├── forms.py
├── __init__.py
├── models.py
├── routes.py
├── s3.py
└── templates
    ├── account.html
    ├── add_tab.html
    ├── edit_tab.html
    ├── home.html
    ├── layout.html
    ├── login.html
    ├── register.html
    └── tabs.html
```



Working App


Not secure | app.mit.pp.ua
Incognito (2)
ip-172-31-39-51.eu-central-1.compute.internal
Tabs
Add tab
Account
Logout
Tabs
There is no tabs to display. Create one.
1
Not secure | app.mit.pp.ua
Incognito (2)
ip-172-31-39-51.eu-central-1.compute.internal
Tabs
Add tab
Account
Logout
Tab created!
Tabs
3
Google


Not secure | app.mit.pp.ua/tabs/add/
Incognito (2)
ip-172-31-39-51.eu-central-1.compute.internal
Tabs
Add tab
Account
Logout
Create Tab
2
Name
URL
http://google.com
Comment
Use comment as a tab name
Submit

Dockerizing

Initial docker image is tiangolo/uwsgi-nginx-flask-docker

It is an image with uWSGI and Nginx for Flask applications in Python running in a single container.

```
root@28ebc0537c44:/# cat /etc/supervisor/conf.d/supervisord.conf
[supervisord]
nodaemon=true

[program:uwsgi]
command=/usr/local/bin/uwsgi --ini /etc/uwsgi/uwsgi.ini
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0

[program:nginx]
command=/usr/sbin/nginx
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0
# Graceful stop, see http://nginx.org/en/docs/control.html
stopsignal=QUIT
root@28ebc0537c44:/#
```

The combination of uWSGI with Nginx is a common way to deploy Python Flask web applications.
It is widely used in the industry and would provide decent performance.

Dockerfile

```
FROM tiangolo/uwsgi-nginx-flask:python3.7
WORKDIR /root
RUN curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip" && \
    unzip awscliv2.zip && \
    ./aws/install && \
    rm -rf awscliv2.zip aws/

COPY ./.aws /root/.aws
COPY ./aws_credentials /root/.aws/credentials

WORKDIR /app
COPY requirements.txt uwsgi.ini /app/
COPY tabs /app/tabs
COPY static /app/static

RUN pip install --upgrade pip && \
    pip install -r requirements.txt

EXPOSE 80
```

AWS Infrastructure

Terraform scenario was written for creating Aws infrastructure. Following components are managed with terraform:

- ECS cluster
- ECS service
- ECS task definition
- ECS auto scaling policies
- ECS auto scaling target
- CloudWatch alarms for scale in and scale out

RDS, S3, ELB, IAM users and policies were created manually.

Jenkins

Jenkins pipeline will build a docker image and push it to private repository on dockerhub.

AWS credentials with permissions to upload and delete S3 files are stored in Jenkins *secretfile* credentials. It will be added to our docker image.

Last step in pipeline is force ECS to pull latest image and deploy to all instances in ECS service.

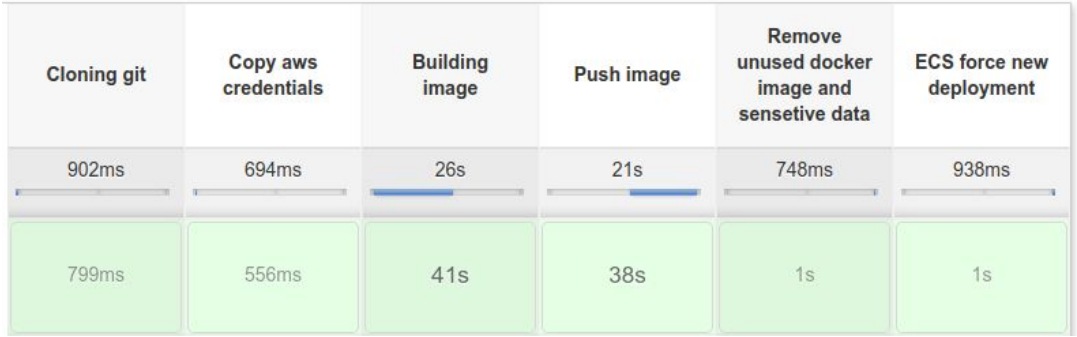
Jenkins Pipeline

```
environment {...}
 stages {
   stage('Cloning git') {...}
   stage('Copy aws credentials') {
     steps{
       script {
         withCredentials([file(credentialsId: 'aws_credentials', variable: 'aws_credentials')]) {
                   sh "cp \$aws_credentials aws_credentials"}
       }
     }
   }
   stage('Building image') {...}
   stage('Push image'){...}
   stage('Remove unused docker image and sensitive data') {...}
   stage('ECS force new deployment') {
     steps{
       sh "aws ecs update-service --cluster white-hart --service tabs --force-new-deployment"
     }
   }
 }
}
```

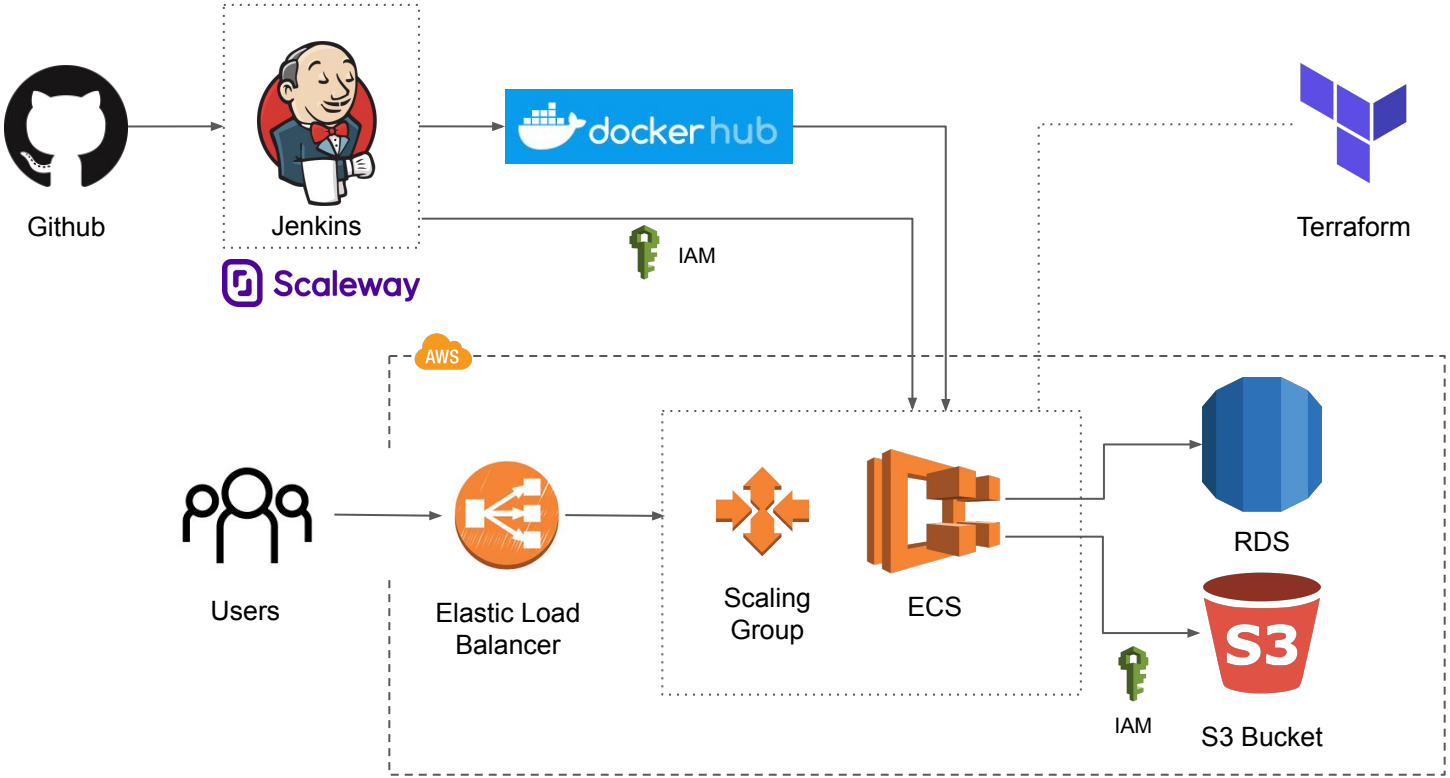
Jenkins

In order to get this pipeline work we will have to install **docker** and **aws-cli** to the machine where Jenkins is installed.

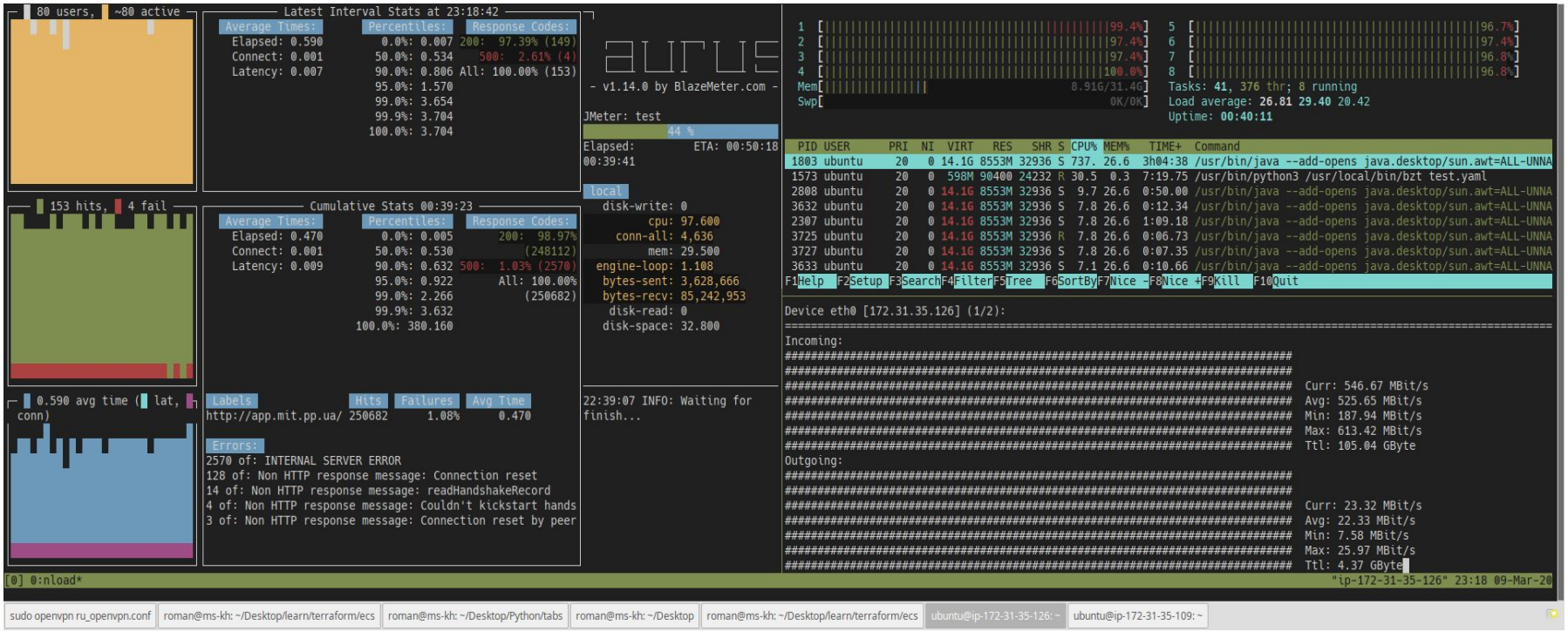
Jenkins user should be added to docker group. Aws user should have permission to perform UpdateService call for ECS service.



Architecture Design



Load Test


80 users, ~80 active
Latest Interval Stats at 23:18:42
Average Times:
Elapsed: 0.590
Connect: 0.001
Latency: 0.007
Percentiles:
0.0%: 0.007
50.0%: 0.534
90.0%: 0.806
95.0%: 1.570
99.0%: 3.654
99.9%: 3.704
100.0%: 3.704
Response Codes:
200: 97.39% (149)
500: 2.61% (4)
All: 100.00% (153)
- v1.14.0 by BlazeMeter.com -
JMeter: test
44 %
Elapsed: 00:39:41
ETA: 00:50:18
local
disk-write: 0
cpu: 97.600
conn-all: 4,636
mem: 29.500
engine-loop: 1.108
bytes-sent: 3,628,666
bytes-recv: 85,242,953
disk-read: 0
disk-space: 32.800
153 hits, 4 fail
Cumulative Stats 00:39:23
Average Times:
Elapsed: 0.470
Connect: 0.001
Latency: 0.009
Percentiles:
0.0%: 0.005
50.0%: 0.530
90.0%: 0.632
95.0%: 0.922
99.0%: 2.266
99.9%: 3.632
100.0%: 380.160
Response Codes:
200: 98.97% (248112)
500: 1.03% (2570)
All: 100.00% (250682)
0.590 avg time (lat, conn)
Labels
Hits
Failures
Avg Time
http://app.mit.pp.ua/ 250682 1.08% 0.470
Errors:
2570 of: INTERNAL SERVER ERROR
128 of: Non HTTP response message: Connection reset
14 of: Non HTTP response message: readHandshakeRecord
4 of: Non HTTP response message: Couldn't kickstart hands
3 of: Non HTTP response message: Connection reset by peer
22:39:07 INFO: Waiting for finish...
1 [99.4%]
2 [97.4%]
3 [97.4%]
4 [100.0%]
5 [96.7%]
6 [97.4%]
7 [96.8%]
8 [96.8%]
Mem[8.91G/31.4G]
Swp[0K/0K]
Tasks: 41, 376 thr; 8 running
Load average: 26.81 29.40 20.42
Uptime: 00:40:11
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
1803 ubuntu 20 0 14.1G 8553M 32936 S 737. 26.6 3h04:38 /usr/bin/java --add-opens java.desktop/sun.awt=ALL-UNNA
1573 ubuntu 20 0 598M 90400 24232 R 30.5 0.3 7:19.75 /usr/bin/python3 /usr/local/bin/bzt test.yaml
2808 ubuntu 20 0 14.1G 8553M 32936 S 9.7 26.6 0:50.00 /usr/bin/java --add-opens java.desktop/sun.awt=ALL-UNNA
3632 ubuntu 20 0 14.1G 8553M 32936 S 7.8 26.6 0:12.34 /usr/bin/java --add-opens java.desktop/sun.awt=ALL-UNNA
2307 ubuntu 20 0 14.1G 8553M 32936 S 7.8 26.6 1:09.18 /usr/bin/java --add-opens java.desktop/sun.awt=ALL-UNNA
3725 ubuntu 20 0 14.1G 8553M 32936 R 7.8 26.6 0:06.73 /usr/bin/java --add-opens java.desktop/sun.awt=ALL-UNNA
3727 ubuntu 20 0 14.1G 8553M 32936 S 7.8 26.6 0:07.35 /usr/bin/java --add-opens java.desktop/sun.awt=ALL-UNNA
3633 ubuntu 20 0 14.1G 8553M 32936 S 7.1 26.6 0:10.66 /usr/bin/java --add-opens java.desktop/sun.awt=ALL-UNNA
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit
Device eth0 [172.31.35.126] (1/2):
Incoming:
Curr: 546.67 MBit/s
Avg: 525.65 MBit/s
Min: 187.94 MBit/s
Max: 613.42 MBit/s
Ttl: 105.04 GByte
Outgoing:
Curr: 23.32 MBit/s
Avg: 22.33 MBit/s
Min: 7.58 MBit/s
Max: 25.97 MBit/s
Ttl: 4.37 GByte
[0] 0:nload*
"ip-172-31-35-126" 23:18 09-Mar-20
sudo openvpn ru_openvpn.conf
roman@ms-kh: ~/Desktop/learn/terraform/ecs
roman@ms-kh: ~/Desktop/Python/tabs
roman@ms-kh: ~/Desktop
roman@ms-kh: ~/Desktop/learn/terraform/ecs
ubuntu@ip-172-31-35-126: ~
ubuntu@ip-172-31-35-109: ~

Taurus Script

```
execution:
  - scenario: test
    concurrency: 80
    ramp-up: 30m
    hold-for: 60m
scenarios:
  test:
    requests:
    - http://app.mit.pp.ua/
    cookies:
      - name: remember_token
        value: 1|a3b8cf9671d622.......
        domain: app.mit.pp.ua
      - name: session
        value: .eJwlj0tqAzEQBe-itRfqlrpb8mUG....
        domain: app.mit.pp.ua
```

Load Test Metrics

Autoscale Event Messages

In event log of the ECS service we can see notifications when CloudWatch alarm triggers autoscale policy.

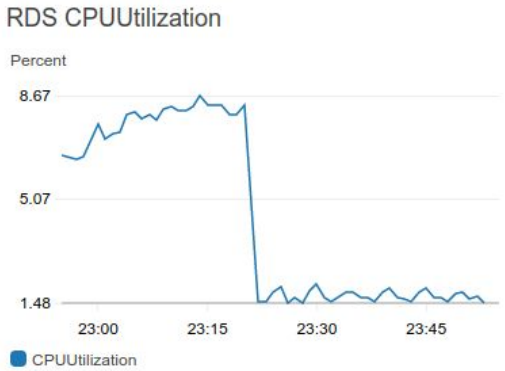
Message: Successfully set desired count to 5. Change successfully fulfilled by ecs. Cause: monitor alarm ServiceCPUScaleUp in state ALARM triggered policy scale-up

Message: Successfully set desired count to 3. Change successfully fulfilled by ecs. Cause: monitor alarm ServiceCPUScaleDown in state ALARM triggered policy scale-down

Load Test Metrics

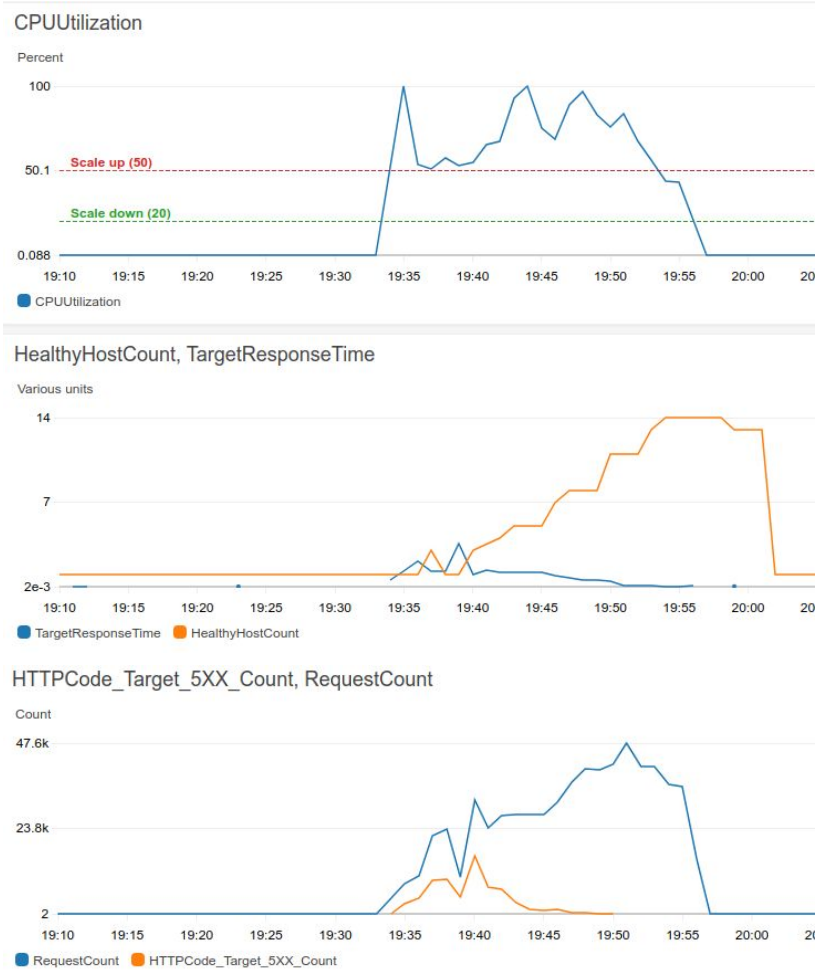
During the test RDS metrics was ok.

The highest RDS metric was CPU utilization.



It is clear from test that app can work stable
with about 10k visit/minute (2 request per visit)
using 9 ECS tasks(0.25 vcpu, 512 mb RAM each).

More Aggressive Test



Thank You!

My contacts

t.me/zhuk_roman

zhuk.roman.v@gmail.com