Web Application Development

Alexander Menshchikov

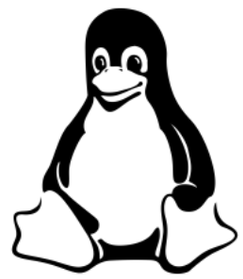
WAD 2020 Lecture Notes – #5 «Production»

Production

WAD 2020 Lecture Notes – #5 «Production»

!?

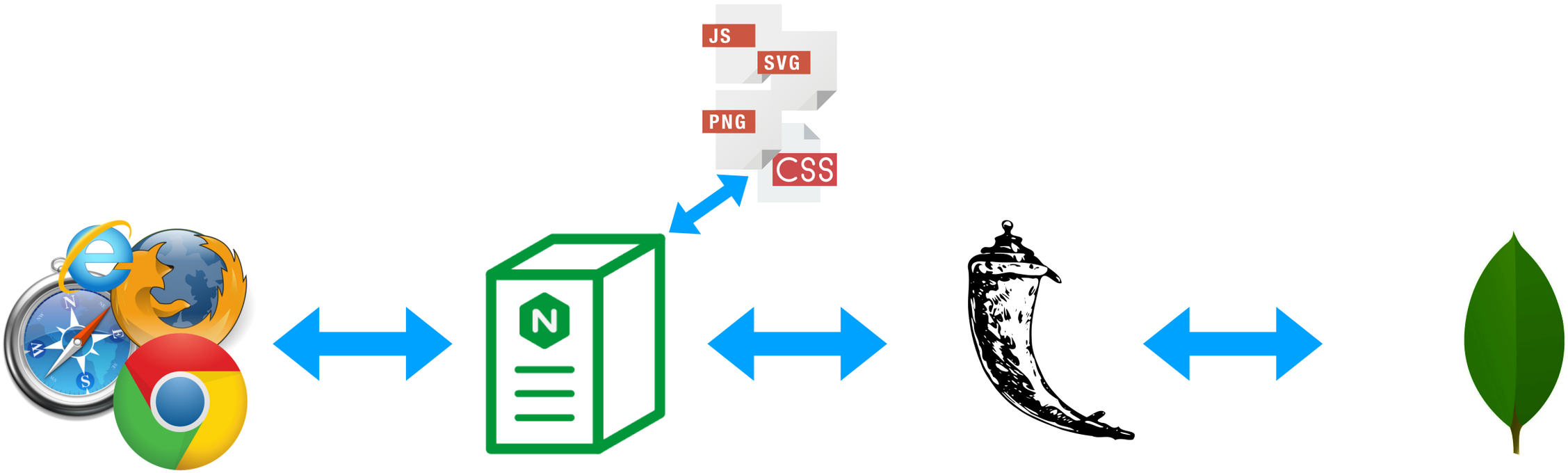
Quiz time



https://quiz.itmo.xyz/

Production

Web Application Architecture



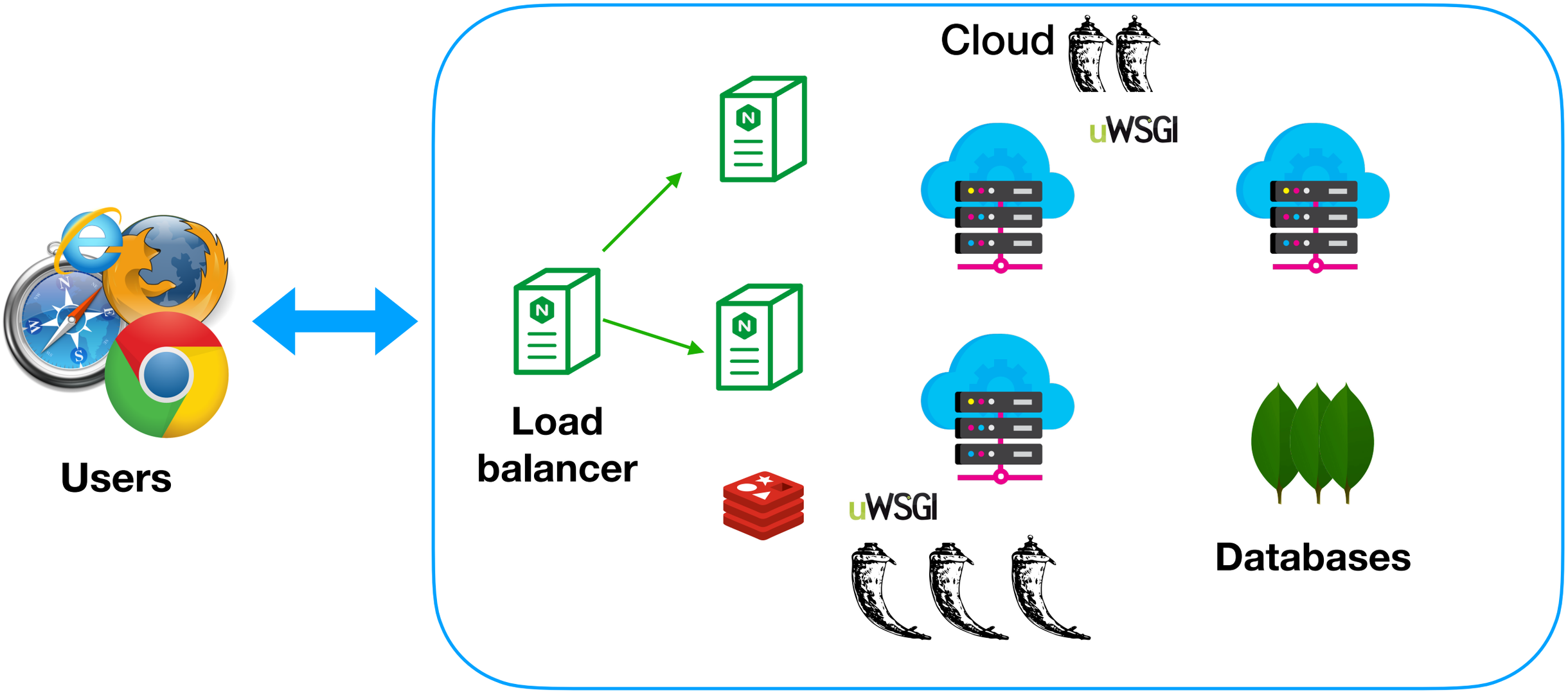
Web browser

Nginx

Flask

MongoDB Database

Web Application Architecture

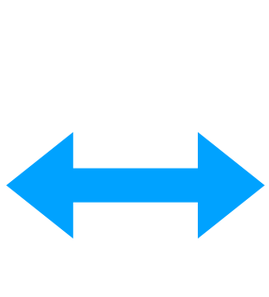


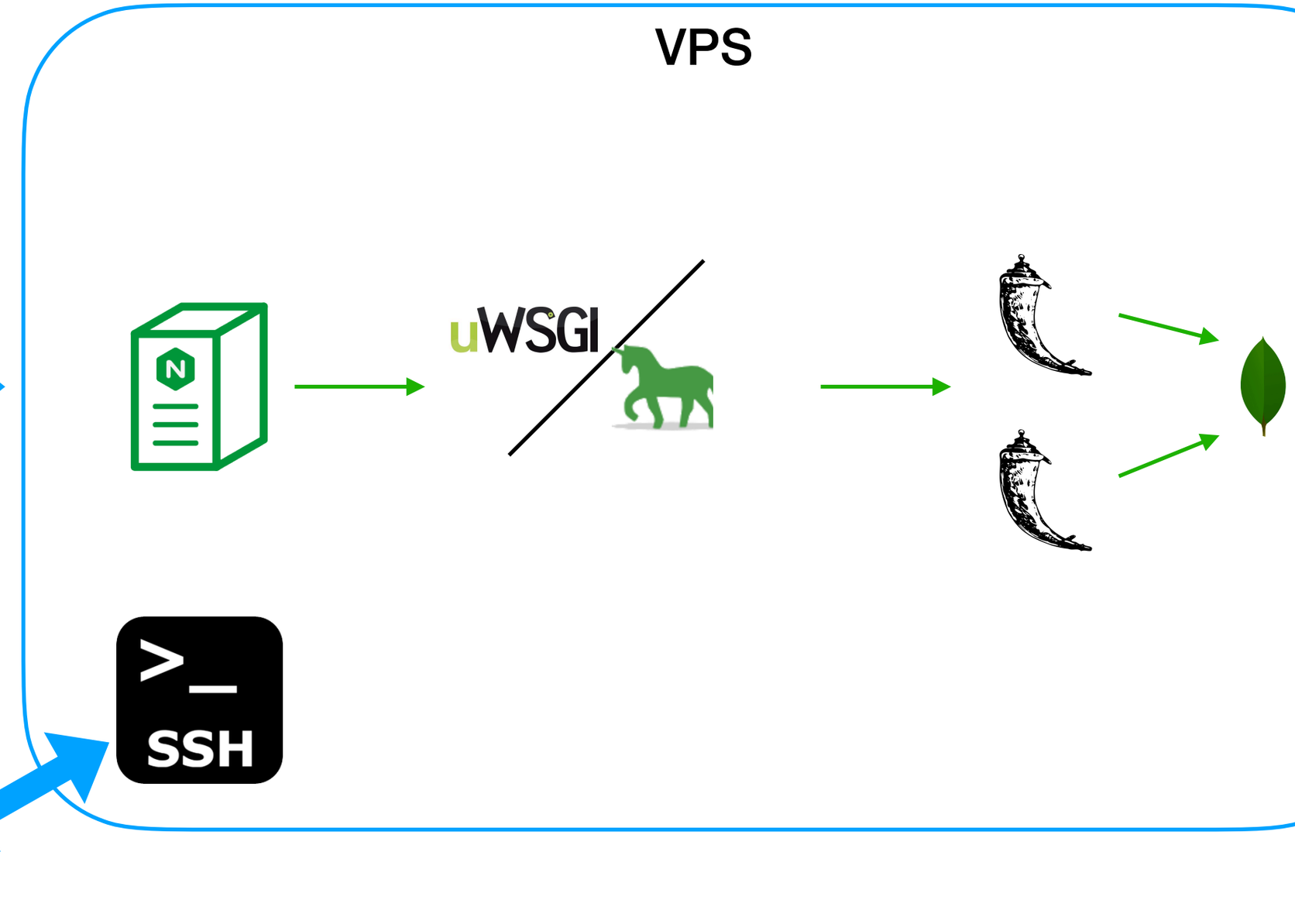
Infrastructure providers

Simple infrastructure

Users

Admin





Algorithm

- Generate SSH keys
- Buy VPS
- Connect via SSH
- Basic OS configuration
- Setup environment

SSH keygen

> ssh-keygen

```
Generating public/private rsa key pair.

Enter file in which to save the key (/<_your_path_>/.ssh/id_rsa):
```

> [enter]

```
OK
```

> cat /<_your_path_>/.ssh/id_rsa.pub

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDy21Sz18T67eA6O/
Romm30cyvlLnTP3OlwpyLE4r97CUGy6AxspjGoYLIwU4NpeGZouD2/
wHkPmXC40QzLY7mxqKzmvZJtfe2KCrHR4z+eopeGqU5+jHyBtASci+4rOXvhm8yrm1VQzm1ST9fcF4
gQNkbnXrrl5q3hoHgjPGe5enOjorvHam1LDXF6EaLmUuKrx5qfNzkOq6eMVE5Ctc/oOy/
uWehpklc+cFiAPWiR6EVwMv5eDJUrEBWvkYCR5jYFCHipHgG5BnzUvqf0Uhf0NP2cjChZ9xPmZQ3sZ
UDOZ2pRBhNlBxMUSuKXiwXlmaPMkvtm74BzKJit8fXowVHD
```

Buy VPS

- vultr.com

- digitalocean.com

- vscale.io

- I will provide VPS for each group project

Buy VPS

production

SPB

Online

Ubuntu 18.04 64bit

512 MB RAM 20 GB SSD 1 CPU 200 ₽

Server settings Usage Graphs History Backups

Hostname and tags

Hostname cs793178

Tags Add tags

Public network settings

IP address	95.213.251.120	Subnet mask	255.255.255.0
PTR-record	Edit record	Gateway	95.213.251.1

SSH keys for root user

n0str

Connect via SSH

> ssh root@95.213.251.120

The authenticity of host '95.213.251.120 (95.213.251.120)' can't be established.

ECDSA key fingerprint is SHA256:…/…

Are you sure you want to continue connecting (yes/no/[fingerprint])?

> yes

Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-76-generic x86_64)

root@cs793178:~#

Basic os configuration

- Update packages
- Create users
- Check SSH settings
- Setup firewall
- Add SSH keys of the team members

Setup environment

- Install GIT, Python
- Install Nginx web server
- Install MongoDB
- Install other software
- Configure Nginx and MongoDB
- Clone and run the web application

Vim

`:w` - write (save) the file, but don't exit

`:wq` - write (save) and quit

`:q!` - quit and throw away unsaved changes

`/pattern` - search for pattern

`i` - insert before the cursor

`o` - append (open) a new line below the current line

`Escape` - exit insert mode

.vimrc

```
syntax on
colorscheme desert
set tabstop=4
set shiftwidth=4
set expandtab
```

Deploy

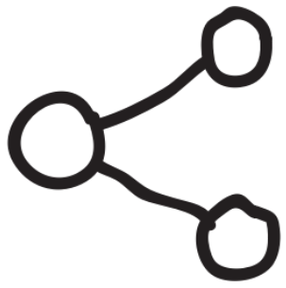
Local	On your computer
Development	Global, for testing
Staging	Pre-release
Production	Live

Deploy

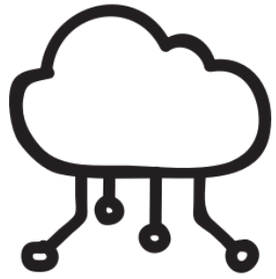
Deploy in our world

Configure

Setup access

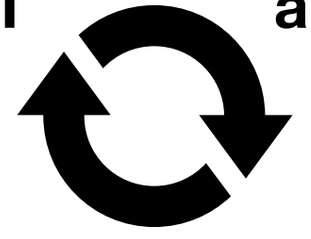


Get sources from GIT



Connect all parts

Monitoring



DevOps

Roles

- Architect
- DevOps engineer
- Release manager
- SRE

Literature

- Vim cheat sheet: https://vim.rtorr.com/
- Putty: https://www.ssh.com/ssh/putty/windows/
- Google SRE workbook: https://landing.google.com/sre/workbook/toc/

Demo

Flask

Logging

```python
from flask import Flask, render_template, request
from logging.handlers import RotatingFileHandler
import logging


app = Flask(__name__)

handler = logging.handlers.RotatingFileHandler('logs\\app.log', maxBytes=32, backupCount=2)
handler.setLevel(logging.DEBUG)
handler.setFormatter(logging.Formatter('%(asctime)s [in %(pathname)s:%(lineno)d]: %(message)s '))

app.logger.addHandler(handler)
app.logger.setLevel(logging.DEBUG)
app.logger.info('This message goes to stderr and app.log!')
```

```
root@cs793178:~/demo/logs# ls -la
total 20
drwxr-xr-x 2 root root 4096 Apr 24 17:55 .
drwxr-xr-x 4 root root 4096 Apr 24 17:55 ..
-rw-r--r-- 1 root root   84 Apr 24 17:55 app.log
-rw-r--r-- 1 root root   76 Apr 24 17:55 app.log.1
-rw-r--r-- 1 root root  100 Apr 24 17:55 app.log.2
```

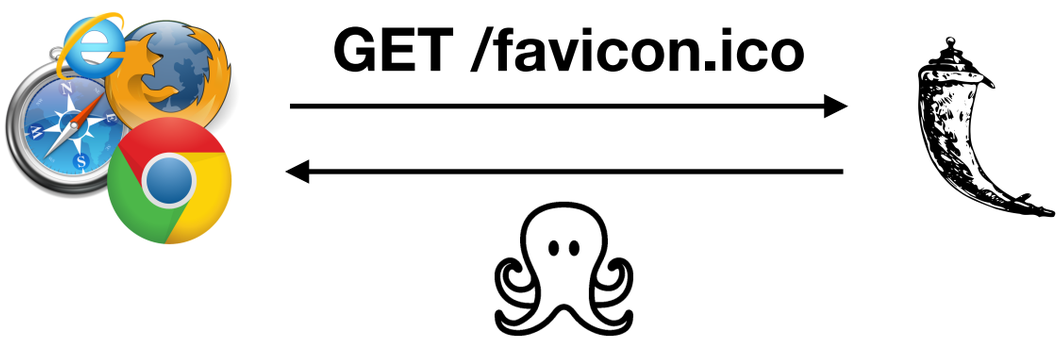
Url for

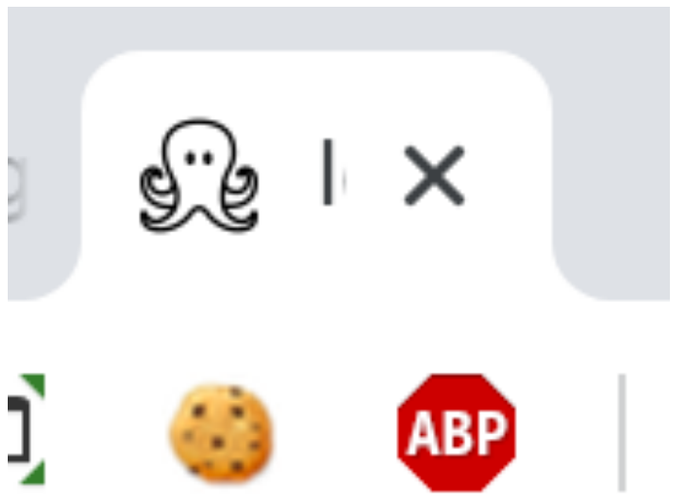
```python
@app.route('/')
def index():
    return f"<a href='{ url_for('index') }'>Home</a> | \
    <a href='{ url_for('login') }'>Login</a> | \
    <a href='{ url_for('register') }'>Register</a> | \
    <a href='{ url_for('profile', username='Alexander') }'>Profile</a>"


@app.route('/login')
def login():
    return 'login'

@app.route('/register/')
def register():
    return 'register'

@app.route('/user/<username>')
def profile(username):
    return f"{username}\'s profile"
```

Favicon

```python
from flask import Flask, send_from_directory


app = Flask(__name__)

@app.route('/favicon.ico')
def favicon():
    return send_from_directory("static", "favicon.ico", mimetype="image/vnd.microsoft.icon")
```

```python
@app.route('/')
def index():
    return "Hello!"


if __name__ == "__main__":
    app.run(host='localhost', port=5000, debug=True)
```





Flash with categories

```python
if name == "alex":
    flash(f"Goodbye, {name}", "success")
else:
    flash(f"No such user: {name}", "danger")
```

```jinja
{% with messages = get_flashed_messages(with_categories=true) %}
  {% if messages %}
    <ul class=flashes>
    {% for category, message in messages %}
      <li class="{{ category }}">{{ message }}</li>
    {% endfor %}
    </ul>
  {% endif %}
{% endwith %}
```

File upload

GET /

```
<form method=POST enctype="multipart/form-data>"
      <input type=file name=file>
      <input type=submit value=Upload>
</form>
```

POST /

- Check if file was sent
- Check that filename is not empty
- Check that file extension is allowed
- Save to upload folder

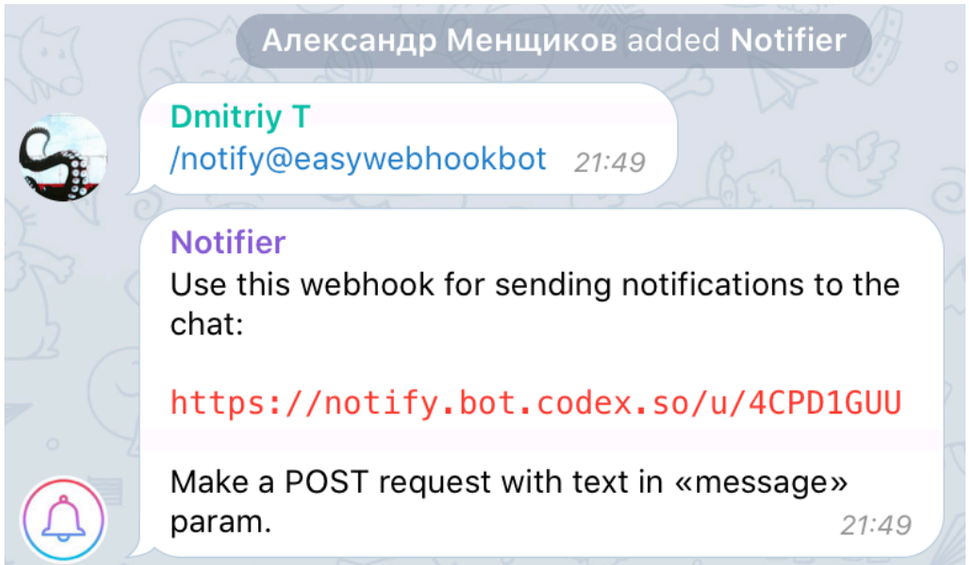
GET /uploads/<filename>

File upload

```python
from flask import Flask, render_template, flash, send_from_directory, request
import os

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'upload'
app.config['SECRET_KEY'] = 'the random string'

@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == "POST":
        ff = request.files['file']
        ff.save(os.path.join(app.config['UPLOAD_FOLDER'], ff.filename))
        flash('Successfully saved', 'success')
    return render_template("form.html")


@app.route('/uploads/<filename>')
def uploaded_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)


if __name__ == "__main__":
    app.run(host='localhost', port=5000, debug=True)
```

Requests

@easywebhookbot




```python
@app.errorhandler(InternalServerError)
def handle_500(e):
    requests.post("https://notify.bot.codex.so/u/4CPD1GUU", {
        "message": f"*Exception* on the server: `{str(e)}`",
        "parse_mode": "Markdown"
    })
    return str(e), 500
```

Literature

- Logging: https://flask.palletsprojects.com/en/1.1.x/logging/
- Url-building: https://flask.palletsprojects.com/en/1.1.x/quickstart/#url-building
- Favicon: https://flask.palletsprojects.com/en/1.1.x/patterns/favicon/
- Flashing: https://flask.palletsprojects.com/en/1.1.x/patterns/flashing/#flashing-with-categories
- Fileupload: https://flask.palletsprojects.com/en/1.1.x/patterns/fileuploads/
- Python Requests: https://github.com/psf/requests

Tutorials

- Flask on Ubuntu: https://realpython.com/kickstarting-flask-on-ubuntu-setup-and-deployment/

- Gunicorn on Ubuntu: https://www.digitalocean.com/community/tutorials/how-to-serve-flask-applications-with-gunicorn-and-nginx-on-ubuntu-18-04

Demo

Production

Nginx

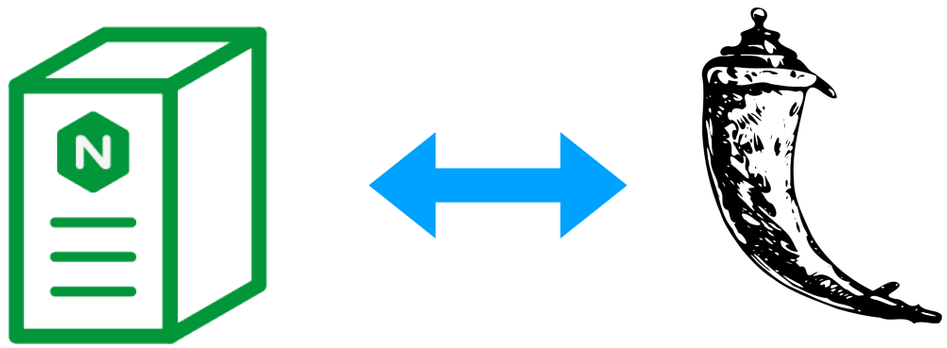
```
server {
    listen 80 default_server;

    access_log access.log;
    access_log error.log;

    location / {
        proxy_pass http://localhost:5000/;
    }
}
```

/etc/nginx/sites-available/default

Nginx

0.0.0.0:80

localhost:5000



Nginx

Flask

Nginx basic auth

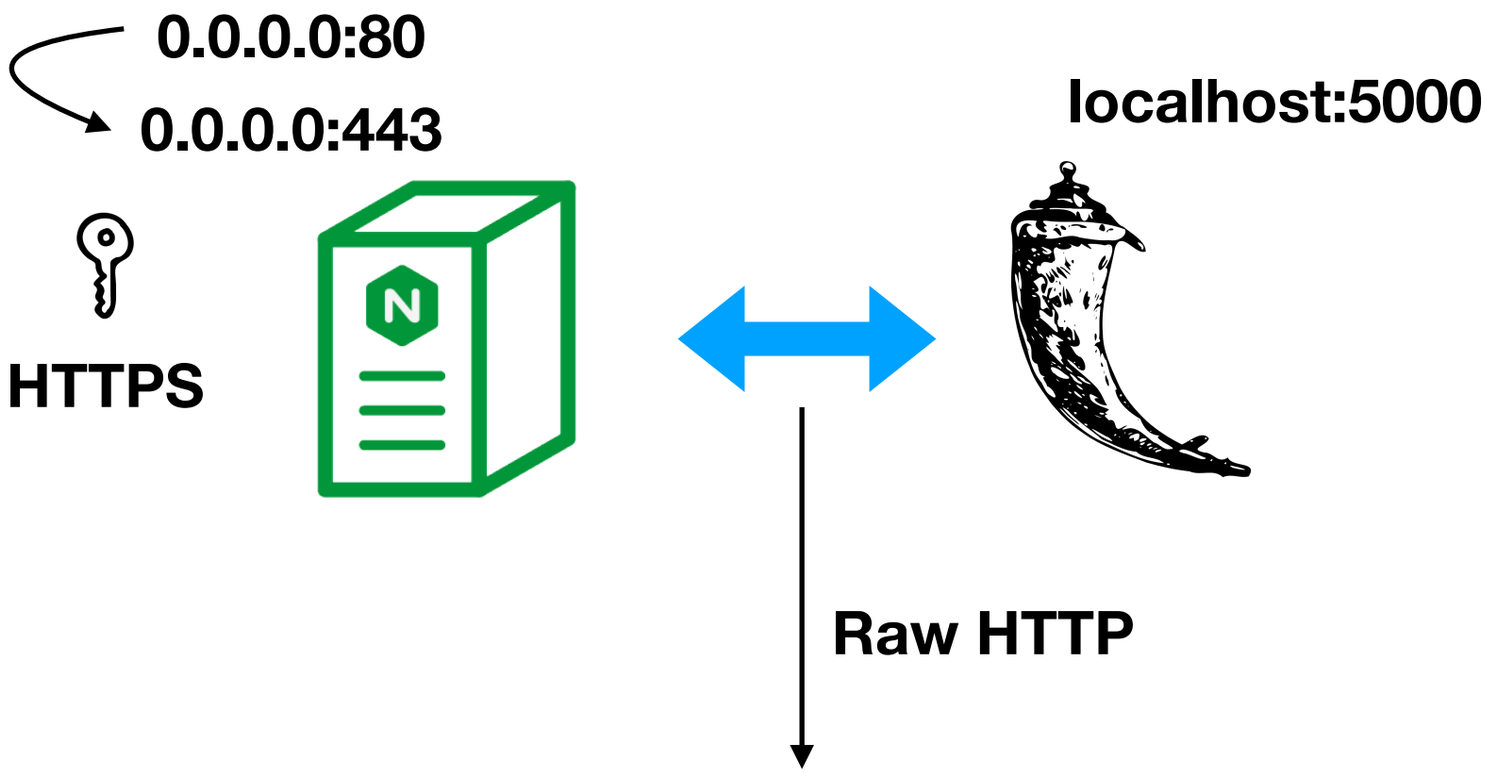
```
> apt install -y apache2-utils

> htpasswd -c /etc/nginx/.htpasswd user
New password:
Re-type new password:
Adding password for user user
```

… add to /etc/nginx/sites-available/…

```
        auth_basic            "You shall not pass";
        auth_basic_user_file /etc/nginx/.htpasswd;
```

```
> service nginx reload
```

Nginx




```
20:47:42.411987 IP localhost.5000 > localhost.34592: Flags [P.], seq 18:155, ack 561, win 512, options [nop,nop,TS val 1843124502 ecr 1843124502], length 137
E...'.@.@..+.......... ..)...8S..........
m...m...Content-Type: text/html; charset=utf-8
Content-Length: 774
Server: Werkzeug/1.0.1 Python/3.6.9
Date: Fri, 24 Apr 2020 17:47:42 GMT


20:47:42.411992 IP localhost.34592 > localhost.5000: Flags [.], ack 155, win 511, options [nop,nop,TS val 1843124502 ecr 1843124502], length 0
E..4x.@.@.......... ....8S..).....(.....
m...m...
20:47:42.412039 IP localhost.5000 > localhost.34592: Flags [P.], seq 155:929, ack 561, win 512, options [nop,nop,TS val 1843124502 ecr 1843124502], length 774
E..:'.@.@.............. ..)...8S...../.....
m...m...<!DOCTYPE html>
<html>
  <head>
    <title>This is an example page</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Certbot

```
apt-get update
apt-get install software-properties-common
add-apt-repository universe
add-apt-repository ppa:certbot/certbot
apt-get update
apt-get install certbot python-certbot-nginx
certbot --nginx
```

→ **Nginx config**

Nginx with SSL

```
server {
    if ($host = l5.itmo.xyz) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 ;
    server_name l5.itmo.xyz;
    return 404; # managed by Certbot
}
```

```
server {
    access_log access.log;
    access_log error.log;

    location / {
        proxy_pass http://localhost:5000/;
    }

    server_name l5.itmo.xyz; # managed by Certbot

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/l5.itmo.xyz/fullchain.pem; #
    ssl_certificate_key /etc/letsencrypt/live/l5.itmo.xyz/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Cert
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbo
}
```

Screen

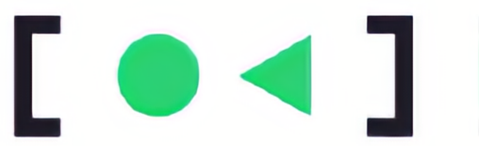
```
apt install screen
screen
screen -r
screen -ls
Ctrl+a Ctrl+d
exit
```

Systemd

```
[Unit]
Description=Example WAD application
After=network.target

[Service]
User=root
WorkingDirectory=/root/demo/
ExecStart=/usr/bin/python3 01_logging.py
Restart=always

[Install]
WantedBy=multi-user.target
```

/etc/systemd/system/wad.service

```
systemctl enable wad.service
systemctl daemon-reload

service wad status
service wad start
service wad stop
service wad restart

journalctl -u wad.service
journalctl -u wad.service --
since "15 minutes ago»
```



Literature

- Certbot: https://certbot.eff.org/lets-encrypt/ubuntubionic-nginx
- Gitignore: https://git-scm.com/docs/gitignore
- Nginx tutorial: https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-18-04
- Flask on systemd: https://blog.miguelgrinberg.com/post/running-a-flask-application-as-a-service-with-systemd
- Screen: https://linuxize.com/post/how-to-use-linux-screen/
- Basic auth: https://docs.nginx.com/nginx/admin-guide/security-controls/configuring-http-basic-authentication/

Demo

Assignment #5

Task #5 - Production

Coding → Deploy → Code-review

Coding

Preparations

1. Install Python programming language **v3.8.2**: https://www.python.org/downloads/
2. Install Flask framework with **pip**: https://docs.python.org/3/installing/index.html#basic-usage
3. Install PyMongo and Flask PyMongo with **pip**
4. Install MongoDB: https://docs.mongodb.com/manual/installation/

What you already have after task #4

1. Flask web application, which can authenticate user with password:
    - Listen on `localhost:5000`
    - Render authentication form on `http://localhost:5000/`
    - Return static images and files on `http://localhost:5000/static/<image_name>`
    - Has secret page for authenticated users on `http://localhost:5000/cabinet`
2. Valid usernames and passwords are stored in MongoDB database

Basic part

1. Add image upload function in cabinet `http://localhost:5000/cabinet/`
2. Image should be saved to `upload` folder

Optimal part

1. Add file extension checks
2. Add function that returns uploaded image
   `http://localhost:5000/upload/<image_name>.png`
3. Show user's avatar in `cabinet` (you can store link to the file in cookie or database)

Challenging part

1. Store the whole file in MongoDB

Deploy

1. Register on GitHub: https://github.com/
2. Join our organization: https://github.com/itmo-wad/
3. You can create new personal repository or use repository for the previous tasks (but don't remove old files, just append new directories)
4. Commit and push your sources to GitHub. And don't forget to describe shortly what have you done in `README.md` file. Use Markdown format: https://guides.github.com/features/mastering-markdown/

Code-review

1. Communicate in Telegram chat
2. Help others to complete the assignment

Cabinet

Store your passwords here

Upload avatar

Выберите файл Файл н...выбран

Upload

Logout

Cabinet

Store your passwords here

Upload avatar

Выберите файл Файл н...выбран

Upload

Successfully saved

Logout

Teams

Team 1	Time picker	https://github.com/itmo-wad/time-picker
Team 2	📆🙋‍♂️ Who when can	https://github.com/itmo-wad/Who_when_can
Team 3	🔄🍕 Ur-opinion	https://github.com/itmo-wad/Ur-opinion
Team 4	CatchMe	https://github.com/itmo-wad/-CatchMe
Team 5	🧸❓ Cozy quiz	
Team 6	🇨🇭🔪 Swiss knife	https://github.com/itmo-wad/Swiss-knife
Team 7	😄⛏️ Emoji picker	https://github.com/itmo-wad/Emoji-picker

Practice time