WEB DEVELOPING – Python

Interne na lokalnom PC

Obrázok, na ktorom je text, snímka obrazovky, diagram, písmo

Automaticky generovaný popis

WSGI – web server gateway interface

Teoreticky navrh produkčneho postredia

Obrázok, na ktorom je text, diagram, snímka obrazovky, rad

Automaticky generovaný popis

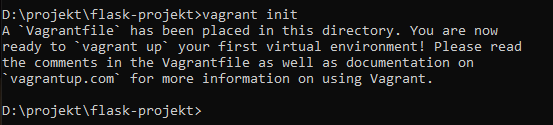
Nginx – rychle , flexibilny vie nativne komunikovat cez WSGI.(alternativa apache)

Je potrebne vediet aku verziu aplikacia použiva , ako ju skompilovat doprodukčneho porstredia a hlavne čo všetko musi byt v tom produkčnom prostredi aby sa ta apka spravala tak ako sa ma spravat.

-----------------------------------Buildovanie projektu---------------------------------------------

Použijeme Vagrant a VirtualBox

Pomocou vagrantu vieme spravit



V tom sa nachadza konfiguracia

V file sme pridali config –

...

# boxes at https://vagrantcloud.com/search.

config.vm.box = "ubuntu/xenial64"

config.vm.provider "virtualbox" do |v|

v.memory = 2048

v.cpus = 4

end

#public network = bridged pridelena ip adresa z routru

config.vm.network "public\_network"

# Disable automatic box update checking. If you disable this, then

...

Vagrant halt – zastavy machinu

Vagrant up naštartuje

Vagrant ssh pripoji sa na machinu

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

Použivame xenial čo je serverova verzia ubuntu všetko odladene a toto je vhodne pre produkčne servre ktore maju poskytnu stabilitu ale ked chceme vyvyjat v velkom pythone.

vagrant@ubuntu-xenial:~$ sudo apt-get install python3.6

ale aj tak to nenajde preto treba pridat iny repozitar ktory toto už obsahuje.

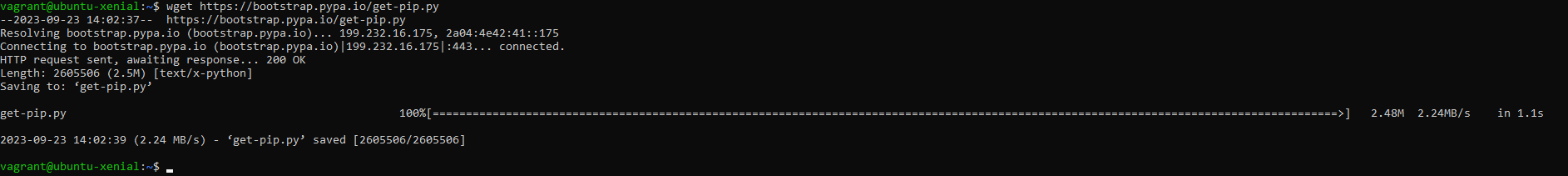
vagrant@ubuntu-xenial:~$ sudo add-apt-repository ppa:deadsnakes/ppa

potom dam

vagrant@ubuntu-xenial:~$ sudo apt-get update

aby sme načitali tieto baličky

nainstalovali sme pip –



Pomocou toho smse pripraivli vagrant na to aby nam naconfiguroval virtualnu machinu

Nasledne ak to mame pomocou „vagrant SSH“ sa vieme pripojit na stroj

Stiahneme pip pre 3.6.15python –

wget <https://bootstrap.pypa.io/pip/3.6/get-pip.py>

pootm to treba naistalovat –

sudo python3 get-pip.py



Toto sa nam nainsataluje do celho systemu –

sudo pip install virtualenv

**instalacia Gitu –**

vytvorenie gitu na serveri –

vagrant@ubuntu-xenial:~$ git config --global user.name "RichardPajzinka"

vagrant@ubuntu-xenial:~$ git config --global user.email [rpajzinka@gmail.com](mailto:rpajzinka@gmail.com)

potom pripojim a vytovim ssh pre git a nasledne ked vytvorim SSH a hodim to do nastaveniach tak cez cmd zadam toto čo mi tam hovori github a nasledne

Obrázok, na ktorom je text, snímka obrazovky, softvér, multimediálny softvér

Automaticky generovaný popis

Ako uploadovat file –

vagrant@ubuntu-xenial:/vagrant$ git add Vagrantfile

vagrant@ubuntu-xenial:/vagrant$ git commit -m "pridanie vagrant file"

[main 016969f] pridanie vagrant file

1 file changed, 85 insertions(+)

create mode 100644 Vagrantfile

vagrant@ubuntu-xenial:/vagrant$ git push origin master

ako skontrolvoat commit –

git checkout

-------------------vytvorenie prostredia pre ostatnych

Ak by sme chceli vytvorit nove prostredie alebo naopak by si niekto chcel pozriet projekt

Stači spustit bootstrap a pripravy to prostredie – „bootstrap.sh“ň

Prostredie – teoreticky toto može byt produkčne prostredie a server

Obrázok, na ktorom je text, snímka obrazovky, diagram, softvér

Automaticky generovaný popis

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

Vytvoritl som si virtualne prostredie

Ak toto maeme tak treba to aktivovat avšak pri kopirovani nastala chyba -

vagrant@ubuntu-xenial:/vagrant$ sudo virtualenv venv

OSError: [Errno 71] Protocol error: '/usr/local/bin/python3' -> '/vagrant/venv/bin/python'

Toto znamena to že v zdielanom foldri sa nemohol vytvorit eniroment avšak to nevadi kedže vieme to ohandlovat –

virtualenv --python=/usr/local/bin/python3.7 venv --always-copy

to ked sme dali tak nam to zbehlo -

vagrant@ubuntu-xenial:/vagrant$ virtualenv --python=/usr/local/bin/python3 venv --always-copy

created virtual environment CPython3.6.15.final.0-64 in 5421ms

creator CPython3Posix(dest=/vagrant/venv, clear=False, no\_vcs\_ignore=False, global=False)

seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app\_data\_dir=/home/vagrant/.local/share/virtualenv)

added seed packages: pip==21.3.1, setuptools==59.6.0, wheel==0.37.1

activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

Nasledne activujeme prostredie + instalujeme flask –

Obrázok, na ktorom je snímka obrazovky, text

Automaticky generovaný popis

vagrant@ubuntu-xenial:/vagrant$ . venv/bin/activate

(venv) vagrant@ubuntu-xenial:/vagrant$ pip install flask

Collecting flask

Using cached Flask-2.0.3-py3-none-any.whl (95 kB)

Collecting itsdangerous>=2.0

Using cached itsdangerous-2.0.1-py3-none-any.whl (18 kB)

Collecting click>=7.1.2

Using cached click-8.0.4-py3-none-any.whl (97 kB)

Collecting Werkzeug>=2.0

Using cached Werkzeug-2.0.3-py3-none-any.whl (289 kB)

Collecting Jinja2>=3.0

Using cached Jinja2-3.0.3-py3-none-any.whl (133 kB)

Collecting importlib-metadata

Using cached importlib\_metadata-4.8.3-py3-none-any.whl (17 kB)

Collecting MarkupSafe>=2.0

Using cached MarkupSafe-2.0.1-cp36-cp36m-manylinux\_2\_5\_x86\_64.manylinux1\_x86\_64.manylinux\_2\_12\_x86\_64.manylinux2010\_x86\_64.whl (30 kB)

Collecting dataclasses

Using cached dataclasses-0.8-py3-none-any.whl (19 kB)

Collecting zipp>=0.5

Using cached zipp-3.6.0-py3-none-any.whl (5.3 kB)

Collecting typing-extensions>=3.6.4

Using cached typing\_extensions-4.1.1-py3-none-any.whl (26 kB)

Installing collected packages: zipp, typing-extensions, MarkupSafe, importlib-metadata, dataclasses, Werkzeug, Jinja2, itsdangerous, click, flask

Successfully installed Jinja2-3.0.3 MarkupSafe-2.0.1 Werkzeug-2.0.3 click-8.0.4 dataclasses-0.8 flask-2.0.3 importlib-metadata-4.8.3 itsdangerous-2.0.1 typing-extensions-4.1.1 zipp-3.6.0

Idem vytvorit zakladnu flask aplikaciu –

from flask import flask

flask\_app = Flask(\_\_name\_\_)

@flask\_app.route("/")

def index():

return "hello world"

pred tym ako to spusitm v enviroemnt prostredi musim pzoriet kau mam IP adresu

tu je tato IP adresa na ktoru sa budem pripajat –

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

Exportnem si 

Pre ip 0.0.0.0 aby to počuvalo pre všetky IP nie len localne

A spustim to tam dostanem

Obrázok, na ktorom je text, snímka obrazovky, písmo, rad

Automaticky generovaný popis

A v console –

(venv) vagrant@ubuntu-xenial:/vagrant$ flask run --host 0.0.0.0

\* Serving Flask app 'app.py' (lazy loading)

\* Environment: production

WARNING: This is a development server. Do not use it in a production deployment.

Use a production WSGI server instead.

\* Debug mode: off

\* Running on all addresses.

WARNING: This is a development server. Do not use it in a production deployment.

\* Running on http://10.0.2.15:5000/ (Press CTRL+C to quit)

192.168.0.17 - - [01/Oct/2023 09:45:35] "GET / HTTP/1.1" 200 -

192.168.0.17 - - [01/Oct/2023 09:45:35] "GET /favicon.ico HTTP/1.1" 404 –

Taktiež na to možem vytvorit aj script cez python –

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

run.py –

from app import flask\_app

if \_\_name\_\_ == "\_\_main\_\_":

debug = False

host = "0.0.0.0"

flask\_app.run(host, debug=debug)

ktory funguje tiež cez tu apliakciu, taka mensia automatizacia aby som nemusel ručne nejak spuštat a tak. Je to iba hello world ale vidime komunikaciu cez share folder a taktiež pracu s localnou adresou.

Pushovanie na git –

(venv) vagrant@ubuntu-xenial:/vagrant$ git add .

(venv) vagrant@ubuntu-xenial:/vagrant$ git commit -m "flask\_hello\_world\_local\_connection"

[main 343056c] flask\_hello\_world\_local\_connection

19 files changed, 746 insertions(+)

create mode 100644 .vagrant/machines/default/virtualbox/action\_provision

create mode 100644 .vagrant/machines/default/virtualbox/action\_set\_name

create mode 100644 .vagrant/machines/default/virtualbox/box\_meta

create mode 100644 .vagrant/machines/default/virtualbox/creator\_uid

create mode 100644 .vagrant/machines/default/virtualbox/id

create mode 100644 .vagrant/machines/default/virtualbox/index\_uuid

create mode 100644 .vagrant/machines/default/virtualbox/private\_key

create mode 100644 .vagrant/machines/default/virtualbox/synced\_folders

create mode 100644 .vagrant/machines/default/virtualbox/vagrant\_cwd

create mode 100644 .vagrant/rgloader/loader.rb

create mode 100644 .vs/VSWorkspaceState.json

create mode 100644 .vs/flask-projekt/v16/.suo

create mode 100644 .vs/slnx.sqlite

create mode 100644 \_\_pycache\_\_/app.cpython-36.pyc

create mode 100644 app.py

create mode 100644 bootstrap.sh

create mode 100644 run.py

create mode 100644 ubuntu-xenial-16.04-cloudimg-console.log

(venv) vagrant@ubuntu-xenial:/vagrant$ git push

Git add . -  
**Add All Untracked Files and Modifications**:

If you want to add all untracked files and modifications to the staging area, you can use the following command:

Routing a flask –

Decorator – tento decorator meni nejakym sposobom funkcionalitu

@flask\_app.route("/")

Tejto funckie –

def index():

return "hello world"

ale vdaka tomu vieme pisat momentalne iba IPv4 adresu

pridal som

@flask\_app.route("/admin/")

def view\_admin():

return"hello z inej url"

do app.py a spustil som to cez python run.py

Obrázok, na ktorom je text, snímka obrazovky, písmo, rad

Automaticky generovaný popis

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

Aby sme po každej zmene v našej appke nemuseli vypnut a zapnut run.py tak možeme zmenit to takto

from app import flask\_app

if \_\_name\_\_ == "\_\_main\_\_":

debug = True

host = "0.0.0.0"

flask\_app.run(host, debug=debug)

a tym padom

(venv) vagrant@ubuntu-xenial:/vagrant$ python run.py

\* Serving Flask app 'app' (lazy loading)

\* Environment: production

WARNING: This is a development server. Do not use it in a production deployment.

Use a production WSGI server instead.

\* Debug mode: on

\* Running on all addresses.

WARNING: This is a development server. Do not use it in a production deployment.

\* Running on http://10.0.2.15:5000/ (Press CTRL+C to quit)

\* Restarting with stat

\* Debugger is active!

\* Debugger PIN: 238-082-952

\* Detected change in '/vagrant/app.py', reloading

\* Restarting with stat

\* Debugger is active!

\* Debugger PIN: 238-082-952

Aj po zmene nam to hned reštartovlaia vieme kde sme spravili zmen

Navrh aplikacie –

Obrázok, na ktorom je text, písmo, rad, snímka obrazovky

Automaticky generovaný popis

Ideme na HTML page –

Aktivujem si svoje venv -

vagrant@ubuntu-xenial:/vagrant$ . venv/bin/activate

(venv) vagrant@ubuntu-xenial:/vagrant$ ls -ltr

Pred každym zapnutim

(venv) vagrant@ubuntu-xenial:/vagrant$ python run.py

Je fajn si poziet ifcondig aku IP adresu aktulne použivame

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

V skripte welcome\_page.html

Sme vytvoili html welcome page pre našu webstranku

Obrázok, na ktorom je text, snímka obrazovky, písmo, softvér

Automaticky generovaný popis

Vieme spravit jinja template

Obrázok, na ktorom je text, snímka obrazovky, softvér, displej

Automaticky generovaný popis

Vieme použit block title bez toho aby sme nemuseli manualne vytvarat každu stranku

Jinja ak chceme aby vykonala nejaky kod vieme použit

Obrázok, na ktorom je text, písmo, snímka obrazovky

Automaticky generovaný popis

About sme prepisali pomocou jinja templatu aby sme dalšie stranky nemuseli vytvarat viac menej rovnake

about.jinja

{% extends "\_main.jinja" %}

{% block title %} Vitajte na blogu {% endblock %}

{% block main %}

Ahojte ja som Richard a vitajte v kategorii about.

{% endblock %}

Možeme vytvorit aju nejaky gitognore file avšak ja si to cele budem updatovat

Koli všetkemu čo potrbeujem mat tam kde mam

Muism všetky IF a FOR prikazy dat v jinja template aby mu chapal

{% extends "\_main.jinja" %}

{% block title %} Vitajte na Articles {% endblock %}

{% block main %}

{% if articles %}

<ul>

{% for id, article in articles %}

<li>

<h2> {{ article.title }} </h2>

<p> {{ article.content }} </p>

</li>

{% endfor %}

</ul>

{% else %}

<p> Ziadne article :( </p>

{% endif %}

{% endblock %}

Jinja takychto podporuje kopec.

Dokumentacia –

<https://jinja.palletsprojects.com/en/2.10.x/templates/#template-inheritance>

redirect vytvroti specialny response ktory dostaneme od uživatela –

http response status code 302 – presmerovanie na iny stranku

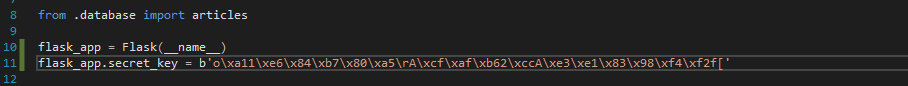
session managment aby sa nestalo to že niekto pred tym ako sa vobec prihlasi bude mat pristup na stranku

na použivanie tohto potrebujeme taktiež secret key – aby sme vedeli povedat či ide o rovnakeho použivatela

pomocou toho si vygenrujeme kluč ktory dodame do app.py

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis



A takto si flask vždy pamata mna ako uživatela a teraz sa možem preklikavat kludne bez toho aby osm sa musel opatovne prihlasovat na stranku

Ked toto mame a funguje ideme do tohto projektu vsunut databazy.

Obrázok, na ktorom je text, snímka obrazovky, diagram, písmo

Automaticky generovaný popis

Velmi podobne aj pri http reqestoch

Ale budme epoužit sql databasu , ideme vytvorit schemu

Použijeme flask s contextom , pozna request a app

Potrebujeme pristupovat k Database lepšie ako cez CMD použijeme sqlitebrowser.org

Tu vidim aj svoj blog.db

