Table of Contents

[Σύνδεσμος github 1](#_Toc70591939)

[Περιγραφή playbook 1](#_Toc70591940)

[Configure Database Server 1](#_Toc70591941)

[Configure Application Server 2](#_Toc70591942)

[Configure Web Server 4](#_Toc70591943)

[Έλεγχος Λειτουργίας 6](#_Toc70591944)

# Σύνδεσμος github

<https://github.com/theohitman/pms18.git>

# Περιγραφή playbook

## Configure Database Server

Εγκαθιστούμε την PostgreSQL χρησιμοποιώντας το ansible role geerlingguy.postgresql. Με την παρακάτω εντολή κατεβάζουμε τον ρόλο από το Ansible Galaxy ώστε να είναι διαθέσιμος στην ansible.

|  |
| --- |
| ansible-galaxy install geerlingguy.postgresql |

Σε αυτό το κομμάτι του playbook εγκαθιστούμε την PostgreSQL στον db01 και με το task “ change listen\_addresses to '\*'” ορίζουμε στην βάση να ακούει σε όλες τις IP διευθύνσεις του VM.

|  |
| --- |
| - hosts: database  roles:  - role: geerlingguy.postgresql  become: yes  tasks:  - name: change listen\_addresses to '\*'  lineinfile:  dest: /etc/postgresql/10/main/postgresql.conf  regexp: "^#listen\_addresses"  line: "listen\_addresses = '\*'"  state: present  backup: yes  become: yes  notify:  - restart postgresql  tags:  - postgres  vars:  postgresql\_databases:  - name: demodb  postgresql\_users:  - name: demouser  password: pass123  postgresql\_locales:  - 'en\_US.UTF-8'  postgresql\_hba\_entries:  - { type: local, database: all, user: postgres, auth\_method: peer }  - { type: local, database: all, user: all, auth\_method: peer }  - { type: host, database: all, user: all, address: '127.0.0.1/32', auth\_method: md5 }  - { type: host, database: all, user: all, address: '0.0.0.0/0', auth\_method: md5 }  - { type: host, database: all, user: all, address: '::1/128', auth\_method: md5 } |

## 

## Configure Application Server

Ορίζουμε κάποιες μεταβλητές που έχουν ισχύ μέσα στο playbook και θα χρειαστούν στα tasks που ακολουθούν.

|  |
| --- |
| vars:  user\_dir: "/home/vagrant"  app\_dir: "{{user\_dir}}/flask-project"  git\_repo\_url: "https://github.com/tsadimas/flask-example-project.git"  git\_repo\_branch: "main" |

Στα επόμενα 2 tasks αντιγράφουμε το public key του github στα known hosts ώστε να μην μας ρωτάει αν το εμπιστευόμαστε και κάνουμε clone το git repo αφού το διαβάσει από την παραπάνω μεταβλητή (git\_repo\_url).

|  |
| --- |
| - name: ensure github.com is a known host  lineinfile:  dest: "{{user\_dir}}/.ssh/known\_hosts"  create: yes  state: present  line: "{{lookup('pipe', 'ssh-keyscan -t rsa github.com')}}"  regexp: "^github\\.com"  - name: clone flask project  git:  repo: "{{git\_repo\_url}}"  version: "{{git\_repo\_branch}}"  clone: yes  dest: "{{app\_dir}}"  changed\_when: true |

Στο πρώτο task γίνεται αντιγραφή του αρχείου .env.example σε .env και στο δεύτερο task γεμίζει αυτό το αρχείο από τις μεταβλητές που έχουμε δώσει εμείς στην ansible και συγκεκριμένα από τα yml αρχεία που βρίσκονται μέσα στους φακέλους group\_vars και host\_vars.

|  |
| --- |
| - name: copy env file  shell : "cp {{app\_dir}}/.env.example {{app\_dir}}/.env"  - name: populate ~/.env  lineinfile:  dest: "{{app\_dir}}/.env"  state: present  regexp: "^{{item.key}}="  line: "{{item.key}}='{{item.value}}'"  with\_items:  - "{{app.env | dict2items}}" |

Στο επόμενο κομμάτι του playbook έχουμε ένα block όπου εγκαθιστά και ενεργοποιεί το Virtual Environment που θα τρέξει η εφαρμογή. Στο πρώτο task εγκαθιστά τα requirements της εφαρμογής και ενεργοποιεί το Virtual Environment. Έπειτα στο rescue σε περίπτωση που αποτύχει το πρώτο task εγκαθιστά όλα τα απαραίτητα dependencies που χρειάζεται για να τρέξει ένα Virtual Environment αφού ελέγξει ότι πρόκειται για Ubuntu VM. Τέλος στο always εγκαθιστά και πάλι όλα τα requirements της εφαρμογής.

|  |
| --- |
| - name: install and activate virtual environment  block:  - name: setup virtual environment  pip:  requirements: "{{app\_dir}}/requirements.txt"  virtualenv: "{{app\_dir}}/fvenv"  virtualenv\_python: python3  state: forcereinstall  rescue:  - name: "APT - install misc packages"  apt:  name: [python3, python3-pip, python3-virtualenv, virtualenv, libmysqlclient-dev, mycli, pgcli, libpq-dev]  update\_cache: yes  become: yes  become\_user: root  when: hostvars[inventory\_hostname].ansible\_distribution == 'Ubuntu'  always:  - name: setup virtual environment  pip:  requirements: "{{app\_dir}}/requirements.txt"  virtualenv: "{{app\_dir}}/fvenv"  virtualenv\_python: python3  state: forcereinstall |

Σε αυτό το κομμάτι αντιγράφουμε το αρχείο flaskgunicorn.service.j2 που είναι οι ρυθμίσεις του service του gunicorn application server στο path "/etc/systemd/system/" που αποθηκεύονται τα services. Στο δεύτερο task ορίζουμε το service αυτό να ξεκινάει κατά την εκκίνηση του server και τέλος με τον hanlder κάνει restart το flaskgunicorn service μετά την αντιγραφή του αρχείου.

|  |
| --- |
| - name: copy flask service file  template:  src: ../files/flask/flaskgunicorn.service.j2  dest: "/etc/systemd/system/flaskgunicorn.service"  become: yes  become\_user: root  notify: restart flaskgunicorn  - name: ensure flaskgunicorn service started  service:  name: flaskgunicorn  state: started  enabled: yes  become: yes  handlers:  - name: restart flaskgunicorn  service:  name: flaskgunicorn  state: restarted  become: yes |

## Configure Web Server

Εγκατάσταση του nginx web server και αντιγραφή των αρχείων παραμετροποίησης για http και https σύνδεση.

|  |
| --- |
| - name: "APT - install nginx"  apt:  name: nginx  update\_cache: yes  - name: copy nginx conf file  template:  src: ../files/nginx/project.nginx.j2  dest: "/etc/nginx/sites-available/flask"  - name: copy nginx https conf file  template:  src: ../files/nginx/project.https.nginx.j2  dest: "/etc/nginx/sites-available/flaskssl" |

Αντιγράφουμε τα certificates που δημιουργήσαμε για τον web server στο path /etc/ssl/private/

|  |
| --- |
| - name: copy certificates  copy:  src: "{{item}}"  dest: /etc/ssl/private/  with\_items:  - ../files/certs/server.crt  - ../files/certs/server.key |

Ενεργοποιούμε το site για http και https σύνδεση δημιουργώντας ένα soft link στο path "/etc/nginx/sites-enabled/ ". Στο τρίτο task απενεργοποιούμε το default site διαγράφοντας το soft link. Τέλος χρησιμοποιώντας έναν handler κάνουμε restart το service του nginx server για να πάρει τις αλλαγές.

|  |
| --- |
| - name: enable flask site in nginx  file:  src: "/etc/nginx/sites-available/flask"  dest: "/etc/nginx/sites-enabled/flask"  state: link  notify: restart nginx  - name: enable flaskssl site in nginx  file:  src: "/etc/nginx/sites-available/flaskssl"  dest: "/etc/nginx/sites-enabled/flaskssl"  state: link  notify: restart nginx  - name: de-activate default nginx site  file:  path: /etc/nginx/sites-enabled/default  state: absent  notify: restart nginx    become: yes  become\_user: root  handlers:  - name: restart nginx  service:  name: nginx  state: restarted  become: yes |

## Έλεγχος Λειτουργίας

<http://192.168.135.101>

<https://192.168.135.101>