EVOLUCIÓN Y GESTIÓN DE LA CONFIGURACIÓN

CURSO 2021/2022

Segunda convocatoria - Septiembre 2022

INSTRUCCIONES GENERALES

- **Ponga su nombre y apellidos de manera clara en la cabecera de todas las hojas que entregue.**
- **Debe responder a las preguntas en hojas distintas a las del enunciado.**
- **Puede usar tantas hojas como estime conveniente.**
- **Puede usar lápiz si lo estima conveniente así como cualquier otro tipo de bolígrafo.**
- **No puede usar Internet ni apuntes para la realización del examen.**
- **No puede preguntar dudas a otros estudiantes.**
- **El examen durará un máximo de 3 horas.**
- **Respondan a cada pregunta en hojas separadas.**
- **Durante todo el tiempo puede contestar todas las preguntas y organizarse el tiempo como lo considere mejor.**

EVOLUCIÓN Y GESTIÓN DE LA CONFIGURACIÓN

CURSO 2021/2022

PREGUNTA 1 (1,5 PUNTOS)

Va a iniciar un proyecto de desarrollo de software con un equipo de 5 personas y se están planteando que sea un proyecto FLOSS. Conteste a las siguientes preguntas razonando la respuesta:

- ¿Qué ventajas e inconvenientes tendría que fuera un proyecto FLOSS?
- ¿Qué licencia elegiría para el proyecto?
- ¿Qué modelo de gobernanza propondría?

PREGUNTA 2 (2 PUNTOS)

Conteste a las siguientes preguntas razonando la respuesta:

- ¿Qué objetivos persigue la ingeniería de líneas de producto software? Ponga un ejemplo de un dominio en el que se le ocurre que podrían usarse
- ¿Qué procesos existen en ingeniería de líneas de productos y para qué sirven?

PREGUNTA 3 (1,5 PUNTO)

Desarrolle las siguientes cuestiones razonando la respuesta:

- ¿Cuál es el objetivo de un buen banco de pruebas?
- Describa dos técnicas de diseño de casos de prueba poniendo un ejemplo de su uso.
- ¿Qué diferencia hay entre diseñar, automatizar, ejecutar y evaluar pruebas?

EVOLUCIÓN Y GESTIÓN DE LA CONFIGURACIÓN

CURSO 2021/2022

e.t.s. ingeniería informática

PREGUNTA 4 (1,5 PUNTOS)

Los ficheros que se muestran a continuación describen parte de la configuración con Ansible del proyecto Django de Decide (el sistema de votación online que hemos visto en la asignatura).

```
- hosts: all

tasks:
 - include: packages.yml
   tags: ["packages"]
 - include: user.yml
 - include: python.yml
   tags: ["app"]
 - include: files.yml
   tags: ["files"]
 - include: database.yml
   tags: ["database"]
 - include: django.yml
   tags: ["django"]
 - include: services.yml
   tags: ["services"]
```

playbook.yml

```
- name: Git clone
  become: yes
  become_user: decide
  git:
    repo: 'https://github.com/wadobo/decide.git'
    dest: /home/decide/decide
    version: master
- name: Python virtualenv
  become: yes
  become_user: decide
  pip:
    name: "gunicorn"
    virtualenv: /home/decide/venv
    virtualenv_python: python3
- name: Requirements
  become: yes
  become_user: decide
  pip:
    requirements: /home/decide/decide/requirements.txt
    virtualenv: /home/decide/venv
    virtualenv_python: python3
```

python.yml

```
- name: Collect static
  become: yes
  become_user: decide
  shell: ~/venv/bin/python manage.py collectstatic --noinput
  args:
    chdir: /home/decide/decide/decide

- name: Database migration
  become: yes
  become_user: decide
  shell: ~/venv/bin/python manage.py migrate --noinput
  args:
    chdir: /home/decide/decide/decide

- name: Admin superuser
  become: yes
  become_user: decide
    shell: ~/venv/bin/python manage.py shell -c "from django.contrib.auth.models import User; User.ob-
jects.filter(username='admin') or User.objects.create_superuser('admin', 'admin@example.com', 'admin')"
    args:
       chdir: /home/decide/decide/decide
```

django.yml

Responda a las siguientes preguntas:

1. ¿Para qué se usa Ansible dentro del proyecto Decide visto en prácticas? ¿Qué otras alternativas conocen, de las vistas en clase, para efectuar la misma tarea?
2. Indique qué cambio tendría que hacer en los ficheros de Ansible para que usase su propio proyecto de Decide.
3. Dentro del fichero python.yml, observamos que se usa gunicorn. ¿Para qué se usa? ¿Qué ventajas ofrece con respecto al software equivalente que se usa en un entorno de desarrollo?
4. Imaginemos que surgen nuevas dependencias Python a añadir dentro del proyecto ¿Cómo podría añadirlas modificando los ficheros mostrados arriba?
5. ¿En qué fichero añadiría las dependencias de paquetes de la distribución Linux usada? Escriba un ejemplo de cómo añadir el paquete Docker a cada una de las máquinas virtuales lanzadas con Vagrant.
6. Que hace el comando collectstatic del fichero django.yml.
7. Explique la instrucción de django.yml que se encarga de generar el usuario administrador en Django. ¿Qué estrategia se está usando para crearlo? ¿Se le ocurre alguna otra forma de hacerlo?

PREGUNTA 5 (1 PUNTOS)

1. Indique como realizaría una copia remota de un repositorio de decide bajo su cuenta de Github.
2. Indique la instrucción para crear una rama llamada *egc-septiembre sin cambiar a esa rama*.
3. Indique la instrucción para saltar a la rama recién creada *egc-septiembre.*
4. Se han realizado cambios en la rama *egc-septiembre en más de un commit* ¿Cómo integraría esos cambios en la rama *master* de tal forma que no se pierda el detalle de cada commit tras la integración?
5. Al parecer, hay algún conflicto que no le permite continuar. Describa como los corregiría e integraría, indicando las instrucciones por consola a realizar.
6. Se desea integrar el *commit* 7255015b de otra rama *egc-features* en la rama *master*. Indique qué instrucción podría usarse para integrar solo ese *commit*.
7. Al parecer el *commit* 7255015b da problemas para integrarse en la rama *master*. Indique la instrucción para abortar la integración.

EVOLUCIÓN Y GESTIÓN DE LA CONFIGURACIÓN

CURSO 2021/2022

e.t.s. ingeniería informática

PREGUNTA 6 (1 PUNTOS)

El siguiente fichero representa un extracto de un workflow de GitHub Actions para el proyecto Decide.

```
1   name: Decide
2   on: [push]
3   jobs:
4     build:
5       runs-on: ubuntu-latest
6       steps:
7       - uses: actions/checkout@v1
8       - uses: actions/setup-python@v1
9         with:
10          python-version: 3.8
11      - name: Install dependencies
12        run: |
13          python -m pip install --upgrade pip
14          pip install -r requirements.txt
15      - run: cd decide;python manage.py migrate
16      - run: cd decide; ./manage.py test;
17    deploy:
18      runs-on: ubuntu-latest
19      steps:
20      - uses: actions/checkout@v1
21      - name: Publish
22        run: echo "Publish in a web platform"
```

Responda a las siguientes preguntas:

1. ¿Para qué sirve Travis o GitHub Actions dentro del proyecto Decide visto en prácticas? Enumere 3 tareas dentro del ciclo de desarrollo de software que se vean beneficiadas por este tipo de herramientas.
2. Explique los conceptos de *step*, *workflow* y *job* dentro del contexto de GitHub Actions.
3. Explique para qué se usa la cláusula *uses* en el workflow anterior y por qué hay sitios donde no aparece dicha cláusula.
4. ¿Qué hay que modificar en el workflow anterior para que "deploy" se ejecute después de "build" y no en paralelo? Indique la(s) línea(s) a modificar/eliminar/añadir y el nuevo texto que añadiría.
5. Sabiendo que existe un *Action* de GitHub referenciable por "azure/webapss-deploy@v2" y que recibe tres parámetros "app-name", "package" y "publish-profile". Indique cómo lo usaría, como último paso de "deploy", indicando que la "app-name" es "decide", el "package" es "./decide", y el "publish-profile" está almacenado en los *secrets* de GitHub bajo el nombre "AZURE_PB".
6. ¿Qué hay que modificar en el workflow anterior para que solo se dispare cuando se haga push a la rama "master"? Indique la(s) línea(s) a modificar/eliminar/añadir y el nuevo texto que añadiría.

EVOLUCIÓN Y GESTIÓN DE LA CONFIGURACIÓN

CURSO 2021/2022

PREGUNTA 7 (1,5 PUNTOS)

1. ¿Para qué sirve Docker dentro del proyecto Decide visto en prácticas? Enumere 3 diferencias frente al uso de Travis o GitHub Actions.
2. Explique los conceptos de *contenedor*, *volumen* e *imagen* dentro del contexto de Docker.
3. Indique la serie de instrucciones que debería ejecutar para hacer lo siguiente: (1) descargar y correr la imagen "ubuntu" en su última versión, (2) actualizar los paquetes de Ubuntu dentro del contenedor (si lo necesita, suponga que el container id es "102030ae"), (3) descargar desde GitHub el código de Decide dentro del contenedor (utiliza una URL genérica si no la recuerda), (4) instalar las dependencias Python de Decide dentro del contenedor y (5) crear una imagen con la etiqueta "ubuntu/decide" a partir del contenedor actual.
4. ¿Qué instrucción permite correr un contenedor a partir de la imagen "ubuntu/decide" anterior, pero indicando que el puerto 80 del host redirija al puerto 8000 del contenedor y que la carpeta del host "~/settings" se accesible desde la carpeta "/app/decide/settings" del contenedor?
5. Rellene el siguiente docker-compose.yml para que tenga un solo servicio cuya imagen es "ubuntu/decide", donde el nombre del contenedor sea "decideApp" y que haga lo mismo que el apartado 6 anterior.

```
version: '3.4'
services:
  decideService:
    restart: always
    …
```

6. Suponiendo que ha levantado correctamente el servicio con "docker-compose up -d", ¿cómo visualiza el log de todos (sólo hay uno) los contenedores de dicho docker-compose?