



Presentación bot Telegram @comuni_bot

Grado en Ingeniería Informática. 4º Curso

Asignatura: Infraestructura Virtual

Universidad de Granada



GitHub

Sergio Cáceres Pintor

<https://github.com/sergiocaceres>



Índice

- * 1. Proyecto elegido y breve descripción
- * 2. Servicios necesarios
- * 3. Integración continua
- * 4. Despliegue en un PaaS
- * 5. Entorno de Pruebas
- * 6. Despliegue en un IaaS



1. Proyecto elegido y breve descripción

- * Bot de Telegram. ¿Por qué?
- * Usuarios de Comunio.



2. Servicios necesarios

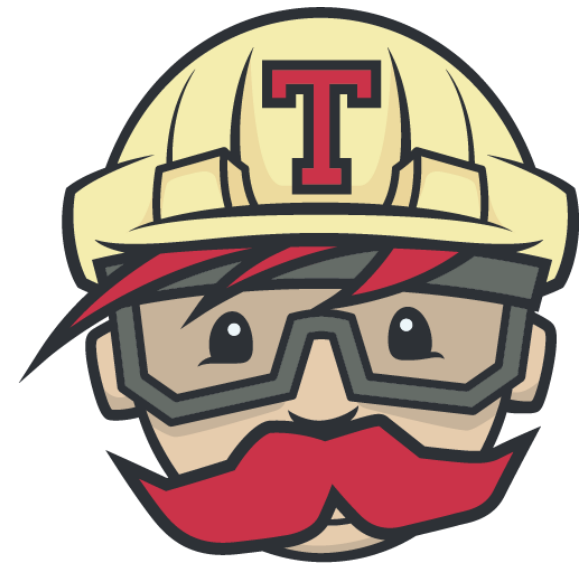
- * Base de datos
- * Herramienta de Scrapeo
- * Uso de Api
- * Lenguaje Python
- * TravisCI
- * Despliegue total - Azure



3. Integración continua

- * Usaremos Travis CI

- * ¿Para qué sirve Travis CI?





3. Integración continua

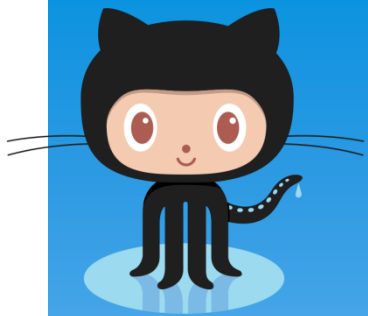
- * Sincronización con TravisCI
- * Fichero Makefile necesario

```
branches:  
  except:  
    - Documentacion
```

```
language: python  
python:  
  - "2.7"
```

```
# command to install dependencies  
install: make install
```

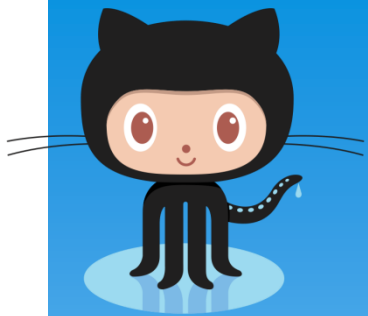
```
# command to run tests  
script: make test
```



3. Integración continua

¿Problemas con TravisCI?
Sí, variables de entorno





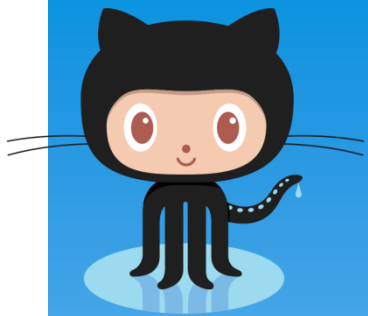
3. Integración continua

¿Solución?

Añadir dichas variables en TravisCI

TOKENBOT	<input type="password"/>
PASS_BD	<input type="password"/>
USR_BD	<input type="password"/>





3. Integración continua

Si todo está correcto
y pasamos los test...

sergiocaceres / IV  build passing

✓ sergiocaceres/IV

101

🕒 Duration: 31 sec

📅 Finished: about 6 hours ago



4. Despliegue en un PaaS

- * Realizado en Heroku



- * Base de datos PostgreSQL

- * Snap-Ci



4. Despliegue en un PaaS

- * Sincronización con Heroku -> Fichero Procfile

```
worker: cd bot_telegram && python bot.py
```

- * Heroku actúa por cada push en nuestro repositorio
- * Para ello debe pasar los test de Travis-Ci. ¿Cómo se sabe?



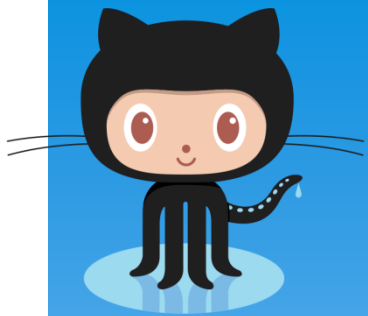
4. Despliegue en un PaaS

Tenemos marcada la casilla

☒ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

[Disable Automatic Deploys](#)



4. Despliegue en un PaaS

Base de datos -> PostgreSQL ¿Cómo se accede desde el bot?

```
psycopg2.connect(database, user, password, host)
```

Donde database, user, password y host lo podemos ver en nuestra cuenta de Heroku, asociada a dicha Base de Datos.



4. Despliegue en un PaaS

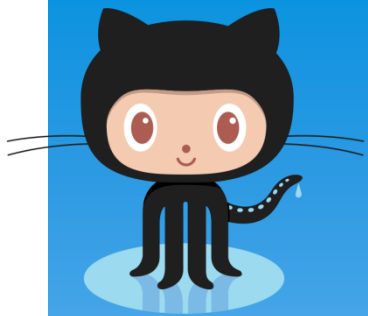
¿Travis-Ci?

Se desplegará si pasa los test

The screenshot shows the Travis CI interface for the repository `sergiocaceres/IV (master)`. The page title is "Pipelines". A green banner states: "This pipeline was automatically configured by Snap. [Customize it here.](#)". Below this, a build job is listed with the following details:

#1	Nov 9th 2016 20:17
7161991	Actualizando Readme Hito 3 cl...

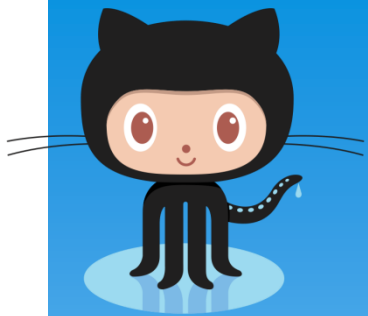
On the right side of the build job, the status is displayed as **PASSED** in green text, with a timestamp of "2 minutes ago" and a duration of "26s". Below the status, there is a link to "EditMe logs" and a circular refresh icon.



4. Despliegue en un PaaS

¿Problemas con Heroku?
Sí, variables de entorno y su
tiempo en ejecución





4. Despliegue en un PaaS



¿Solución?

Añadir dichas variables en Heroku y poner worker en el fichero Procfile

Config Vars

PASS_BD

[Redacted]



TOKENBOT

[Redacted]



USR_BD

[Redacted]



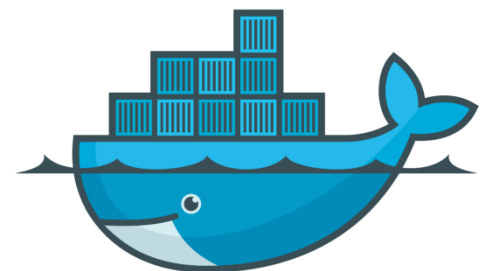
```
worker: cd bot_telegram && python bot.py
```




5. Entorno de pruebas

- * Docker

- * Fichero Dockerfile



docker



5. Entorno de pruebas

- * Debemos sincronizar con nuestro GitHub
- * Cada push, se recibirá y analizará

PUBLIC | AUTOMATED BUILD

sergiocaceres/iv ☆

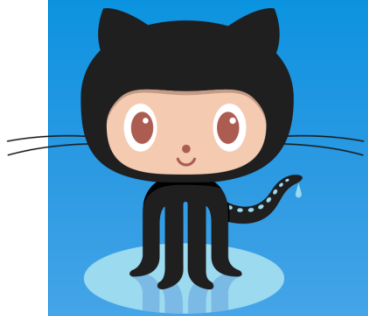
Last pushed: 16 minutes ago

[Repo Info](#) [Tags](#) [Dockerfile](#) [Build Details](#) [Build Settings](#) [Collaborators](#) [Webhooks](#) [Settings](#)

Status	Actions	Tag	Created	Last Updated
✓ Success		latest	23 minutes ago	16 minutes ago
✓ Success		latest	21 hours ago	21 hours ago
✓ Success		latest	21 hours ago	21 hours ago

Source Repository

 [sergiocaceres/IV](#)



5. Entorno de pruebas

Una vez enlazado...

```
sudo docker pull sergiocaceres/iv  
para descargar el contenedor
```



5. Entorno de pruebas

Fichero Dockerfile

Variables de entorno -> ARG, ENV
Instalar git, repositorios,
herramientas python y
requirements



5. Entorno de pruebas

Para lanzar el contenedor con nuestras variables de entorno

```
sergio@sergio-VirtualBox:~/Escritorio/InfraestructuraV/GIT/IV$ sudo docker run -e "TOKENBOT=[REDACTED]" -e "PASS_BD=[REDACTED]" -e "USR_BD=[REDACTED]" -i -t sergiocaceres/iv /bin/bash  
root@da13c3498a80:/# cd IV/ && make execute  
cd bot_telegram && python bot.py
```

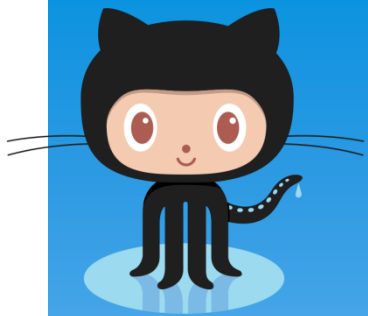


5. Entorno de pruebas

¿Problemas con Docker?

Sí, otra vez las variables de entorno





5. Entorno de pruebas



¿Solución?

Por supuesto, siempre hay una solución.

Fichero Dockerfile ->
Ejecución con
parámetros

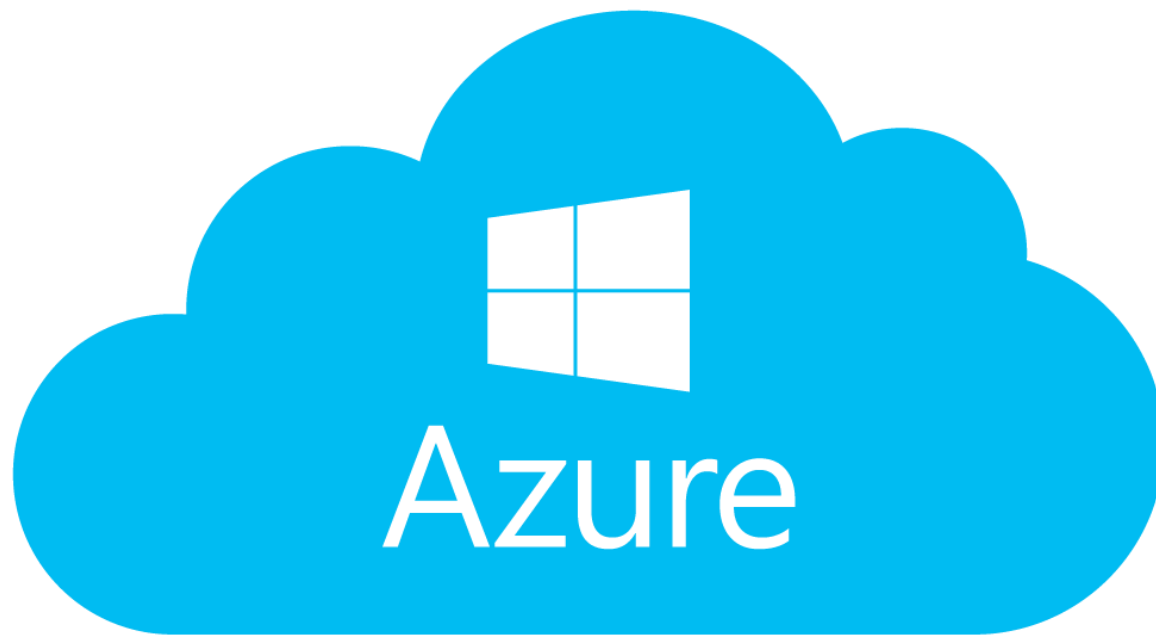
```
ARG TOKENBOT
ARG PASS_BD
ARG USR_BD
ENV TOKENBOT=$TOKENBOT
ENV PASS_BD=$PASS_BD
ENV USR_BD=$USR_BD
```

```
sergio@sergio-VirtualBox:~/Escritorio/InfraestructuraV/GIT/IV$ sudo docker run -e "TOKENBOT=[REDACTED]" -e "PASS_BD=[REDACTED]" -e "USR_BD=[REDACTED]" -i -t sergiocaceres/iv /bin/bash
root@da13c3498a80:/# cd IV/ && make execute
cd bot_telegram && python bot.py
```



6. Despliegue en un IaaS

- * Vagrant
- * Ansible
- * Azure
- * Fabric





6. Despliegue en un IaaS

- * Necesitaremos levantar una máquina virtual en Azure
- * Hacemos uso de Vagrant para crear la máquina virtual
- * Usaremos Ansible para el aprovisionamiento
- * Usaremos Fabric para lanzar la ejecución del bot





6. Despliegue en un IaaS

Nos creamos un certificado para enlazarlo con nuestra cuenta de Azure.

```
openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -keyout azure.pem -out azure.pem  
openssl x509 -inform pem -in azure.pem -outform der -out azure.cer  
chmod 600 azure.pem
```

La insertamos en la web Azure

[Visite el nuevo portal](#) [ESTADO DEL CRÉDITO](#)   @hotmail.com

configuración

[SUSCRIPCIONES](#) [CERTIFICADOS DE ADMINISTRACIÓN](#) [ADMINISTRADORES](#) [GRUPOS DE AFINIDAD](#) [USO](#)

NOMBRE	ESTADO	SUSCRIPCIÓN	ID. DE SUSCRIPCIÓN ↓	HUELLA DIGITAL	FECHA DE EXPIRACI...	🔍
Sergio	✓ Creado	Pase para Azure	██████████	██████████	24/01/2027	



6. Despliegue en un IaaS

Creamos fichero Vagrantfile ->
vagrant init

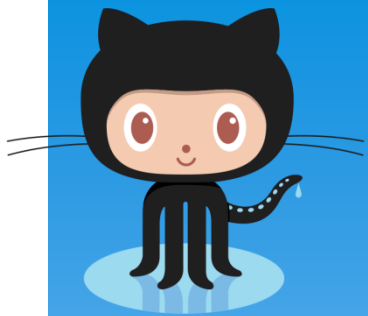
Una vez creado, especificamos
nuestra ID de Azure, nombre de la
máquina virtual, box utilizada,
llamada al playbook ansible, ...



6. Despliegue en un IaaS

Fichero Ansible -> Configuramos
todo lo que debe tener instalada la
máquina virtual

Incluimos supervisor. ¿Para qué?
Para que siempre esté en ejecución
nuestro bot



6. Despliegue en un IaaS

Desplegamos:
`vagrant up --provider=azure`

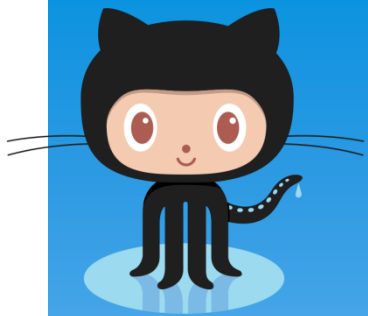




6. Despliegue en un IaaS

Supervisor -> fichero .conf
Le decimos lo que queremos que
ejecute junto con las variables de
entorno

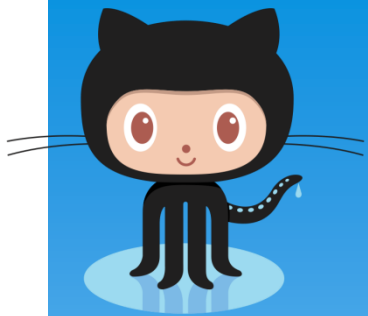
```
[program:comuni_bot]
autostart=false
command=python bot_telegram/bot.py
user=sergio
directory=/home/sergio/IV
environment=
    TOKENBOT="{{TOKENBOT}}",
    USR_BD="{{USR_BD}}",
    PASS_BD="{{PASS_BD}}",
redirect_stderr=true
stdout_logfile=/var/log/supervisor/comuni_bot.log
stderr_logfile=/var/log/supervisor/comuni_bot-error.log
```



6. Despliegue en un IaaS

Nohup

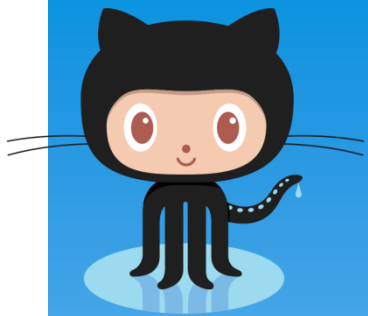
Lo mismo que supervisor pero sin
necesidad de fichero .conf



6. Despliegue en un IaaS

Fabric -> Fichero fabfile.py

Contiene una serie de funciones
para ejecutar con Fabric.



6. Despliegue en un IaaS

¿Cómo se ejecuta?

fab -p CONTRASEÑA -H
usuario@host ORDEN

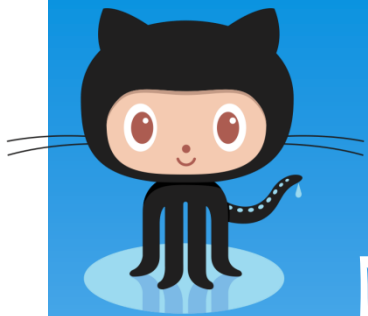
```
sergio@sergio-VirtualBox:~/Escritorio/InfraestructuraV$ fab -p Abecedario1234# -  
H sergio@comunibot.cloudapp.net iniciar  
[sergio@comunibot.cloudapp.net] Executing task 'iniciar'  
[sergio@comunibot.cloudapp.net] run: sudo supervisorctl start comuni_bot  
[sergio@comunibot.cloudapp.net] out: comuni_bot: started  
[sergio@comunibot.cloudapp.net] out:  
  
Done.  
Disconnecting from comunibot.cloudapp.net... done.  
sergio@sergio-VirtualBox:~/Escritorio/InfraestructuraV$
```



6. Despliegue en un IaaS

Con esto ya funcionaría perfectamente, pero... queremos más. Automatizaremos el proceso con un script

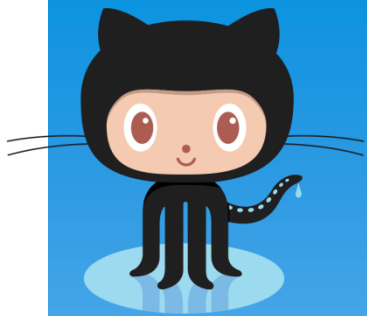




6. Despliegue en un IaaS

Nuestro script de despliegue automático ejecuta todos los pasos anteriores con tan solo con una orden:

```
./script.sh
```



6. Despliegue en un IaaS

```
#!/bin/bash

sudo apt-get update

# Instalamos vagrant
sudo wget https://releases.hashicorp.com/vagrant/1.8.6/vagrant_1.8.6_i686.deb
sudo dpkg -i vagrant_1.8.6_i686.deb

# Instalar plugin para azure
sudo vagrant plugin install vagrant-azure

# Instalación Ansible
sudo apt-get install ansible

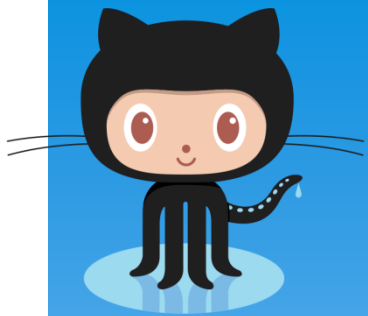
# Despliegue en Azure
sudo vagrant up --provider=azure

# Instalamos Fabric
sudo apt-get install fabric

# Despliegue de la aplicación con Fabric

# Actualiza el supervisor
fab -p Abecedario1234# -H sergio@comunibot.cloudapp.net recargar
#Inicia el supervisor
#fab -p Abecedario1234# -H sergio@comunibot.cloudapp.net iniciar

#Inicia con nohup
fab -p Abecedario1234# -H sergio@comunibot.cloudapp.net iniciar_hup
```



6. Despliegue en un IaaS

¿Problemas para desplegar?

Por supuesto, nuestras queridas
variables de entorno





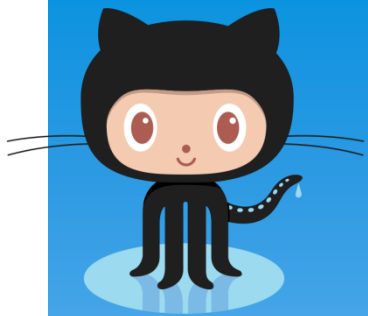
6. Despliegue en un IaaS



¿Solución?

Poner las variables de entorno en el fichero Ansible, en el del supervisor y en el de Fabric.

¿Dónde y cómo se pone?



6. Despliegue en un IaaS

Ansible

Supervisor

```
vars:
  TOKENBOT: "{{ lookup('env', 'TOKENBOT') }}"
  USR_BD: "{{ lookup('env', 'USR_BD') }}"
  PASS_BD: "{{ lookup('env', 'PASS_BD') }}"
tasks:
```

```
environment=
  TOKENBOT="{{TOKENBOT}}",
  USR_BD="{{USR_BD}}",
  PASS_BD="{{PASS_BD}}",
```

Fabric

```
def iniciar():
    with shell_env(TOKENBOT=os.environ['TOKENBOT'], USR_BD=os.environ['USR_BD'], PASS_BD=os.environ['PASS_BD']):
        run('sudo supervisorctl start comuni_bot')
```



Bot funcionando



Sergio Cáceres

20:17:51

/start



Comuniobot

20:17:53

Introduzca acción que desea realizar, usuario de comunio y contraseña

Acciones:

1. **/Alineacion**: Devuelve la alineación con la que se jugó la última jornada
2. **/Noticias**: Devuelve las dos últimas noticias
3. **/Mercado**: Devuelve el mercado de fichajes de la comunidad
4. **/Ofertas**: Devuelve las ofertas que me han hecho
5. **/Traspasos**: Devuelve las ofertas que yo he hecho

Ejemplo: **/Alineacion**,usuario,contraseña

Una vez que introduzca sus datos correctamente, solo debera introducir la acción deseada



|Escribe un mensaje...





Bot funcionando



Sergio Cáceres

20:17:25

/alineacion



Comuniobot

20:17:29

Sergio Asenjo, Nico Pareja, Rosales, Albentosa, Trigueros, Tana,
Hernán Pérez, Dani García, Oriol Riera, Cristiano Ronaldo, Neymar



Escribe un mensaje...





Infraestructura Virtual

¿Preguntas?

