Automation Demystified

Darlene Wong & Yiyu Yang

Automation in General

- Regression Oriented
- Purpose: catch defects
- Client: regression engineers
- Process:
  - Collect what to test against
  - Setup environment
  - Run test
  - Collect report
  - Analyze report

Automation Design Gist

- Data and Code Separation
  - Code
  - Data:
    - Testbed
    - Static Names for Feature Test
    - Test Cases for Regression
- Pass/Fail Criteria
  - clear criteria for pass: actual value vs expected value
  - exception: performance & scaling
- Configuration and Verification
  - Verify Often and Fail Early

Automation Design Gist Continued

- Log Messages
  - Clear Message in Failure: actual value vs expected value
  - Logging Levels
    - critical/error/warning/info/debug/notset
  - Lacking of Logs
  - Inaccurate Information

Log Message Example 1

The received byte number is not expected.

Log Message Example 2

The received byte number, 12283, is not expected.

```
show counter interface ethernet1/1

Interface: ethernet1/1
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Physical port counters read from MAC:
--------------------------------------------------------------------------------
rx-broadcast                  5
rx-bytes                      12283
rx-multicast                  20
rx-unicast                    49
tx-broadcast                  8
tx-bytes                      36795
tx-multicast                  380
tx-unicast                    49
--------------------------------------------------------------------------------
```

Log Message Example 3

The received byte number is not expected. (intf=”ethernet1/1”, rx-bytes=12283, expected-rx-bytes=12284)

```
show counter interface ethernet1/1

Interface: ethernet1/1
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Physical port counters read from MAC:
--------------------------------------------------------------------------------
rx-broadcast                  5
rx-bytes                      12283
rx-multicast                  20
rx-unicast                    49
tx-broadcast                  8
tx-bytes                      36795
tx-multicast                  380
tx-unicast                    49
--------------------------------------------------------------------------------
```

The Zen of Python, by Tim Peters

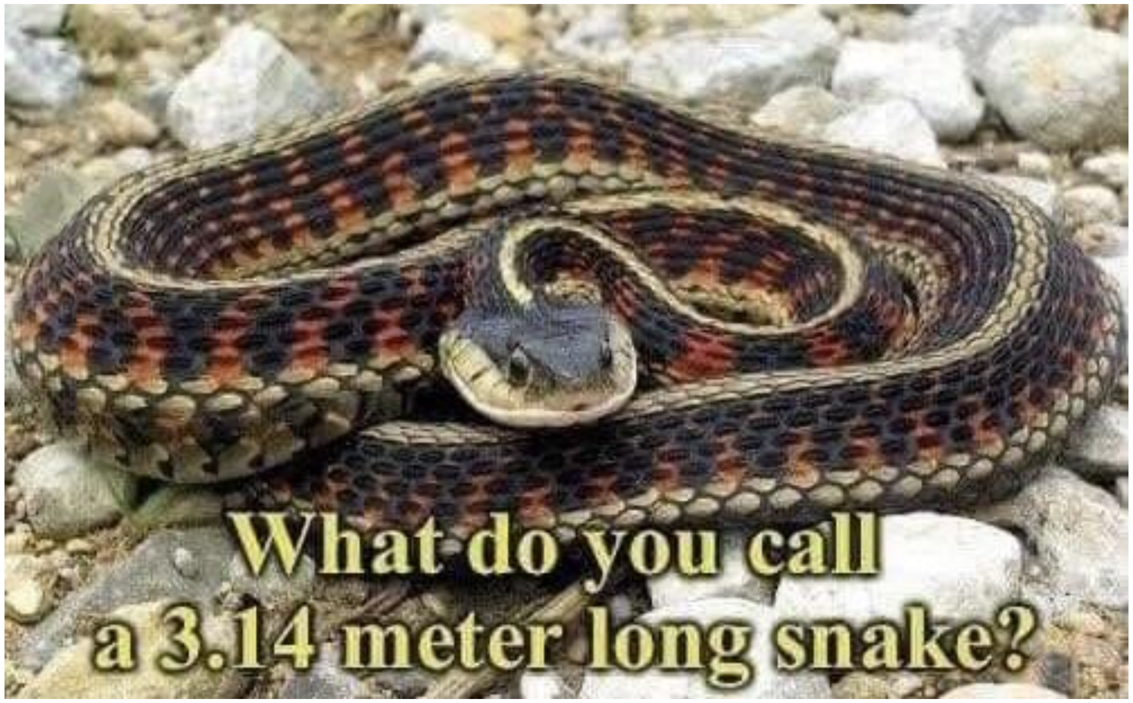
- Explicit is better than implicit.
- Simple is better than complex.
- Readability counts.
- Errors should never pass silently.
- In the face of ambiguity, refuse the temptation to guess.
- If the implementation is hard to explain, it's a bad idea.

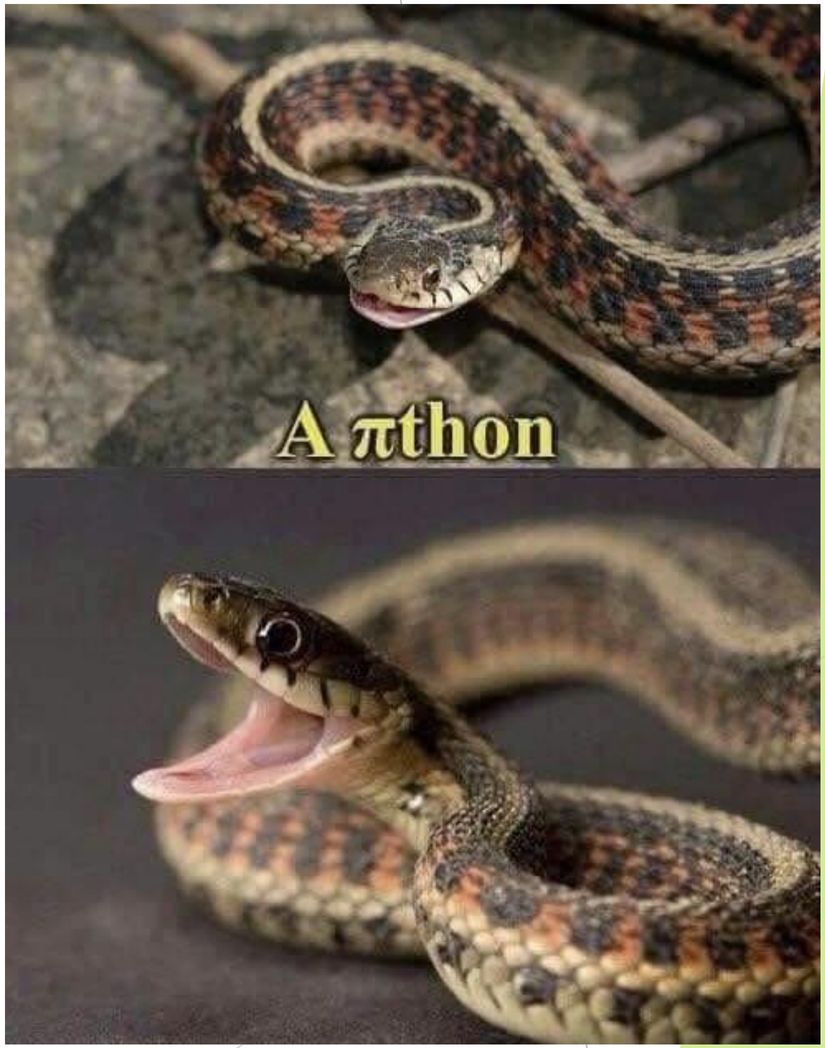
Python Virtual Environment - Setup

- Python Ecosystem (Python 3.6)
  - $ sudo apt-get install python3.6-venv
  - $ python3 -m venv py36
  - $ source py36/bin/activate
  - $ which python
    - /home/test/penvs/py36/bin/python
  - $ python --version
    - Python 3.6.7
  - $ deactivate

Virtual Environment - Packages

- In the virtual environment
  - $ pip install beautifulsoup4
  - $ pip list | grep beautifulsoup4
    - **beautifulsoup4 (4.7.1)**
  - >>> from bs4 import BeautifulSoup
  - >>>
- Not in the virtual environment
  - >>> from bs4 import BeautifulSoup
  - Traceback (most recent call last):
  -   File "<stdin>", line 1, in <module>
  - ImportError: No module named bs4


What do you call
a 3.14 meter long snake?


A πthon

Automation and the CI/CD Pipeline

- CI/CD Pipeline
  1. Checkin
  2. Build
  3. Automated Tests
  4. Deploy
- Benefits of CI/CD
  - Faster Release Cycles
  - Reduced Risk
  - Higher Quality

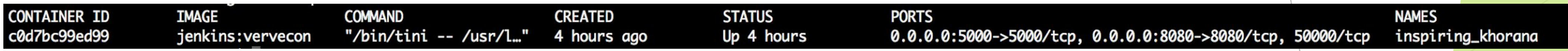
CI/CD Pipeline Walkthrough

1. Jenkins Installation
2. Jenkins Setup
3. GCP Setup
4. Create Application
5. Test and Deploy Application

Jenkins Installation (1/1)

- Build and run Jenkins Docker image
  - `$ docker build -t jenkins:vervecon .`

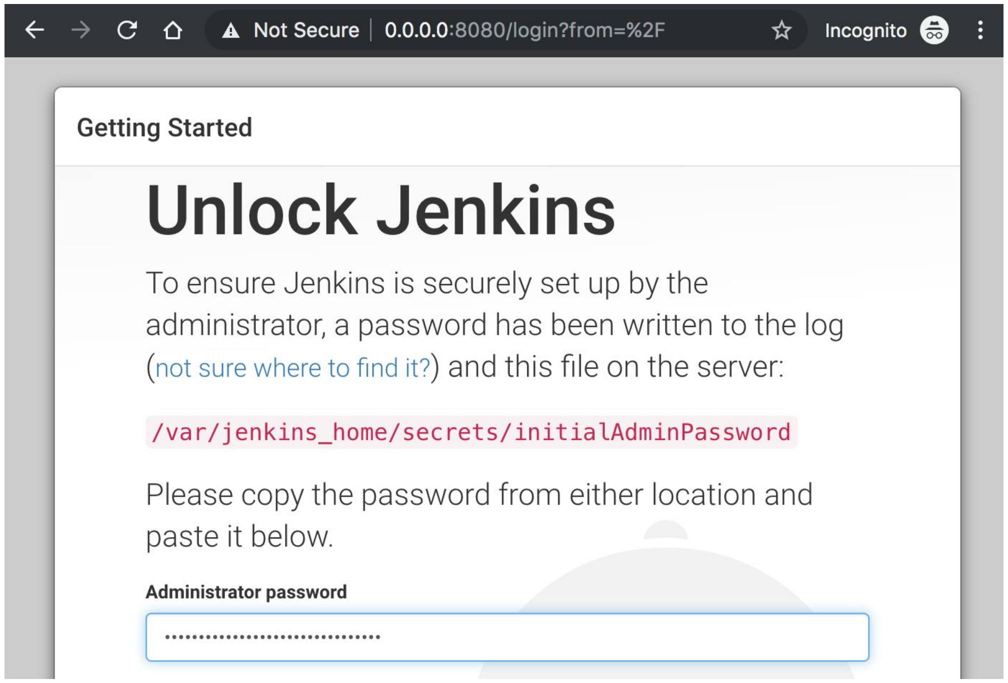
  -
    ```
    $ docker run -d -p 8080:8080 -p 5000:5000 \
    -v /var/run/docker.sock:/var/run/docker.sock -u root jenkins:vervecon
    ```
- Get the running container ID
  - `$ docker ps`



```
CONTAINER ID   IMAGE              COMMAND                  CREATED       STATUS       PORTS                                                       NAMES
c0d7bc99ed99   jenkins:vervecon   "/bin/tini -- /usr/l…"   4 hours ago   Up 4 hours   0.0.0.0:5000->5000/tcp, 0.0.0.0:8080->8080/tcp, 50000/tcp   inspiring_khorana
```

- Run bash in container to obtain password for unlocking Jenkins
  - `$ docker exec -it c0d7bc99ed99 bash`
  - `bash-4.4# cat /var/jenkins_home/secrets/initialAdminPassword`

Jenkins Setup (1/7)



Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

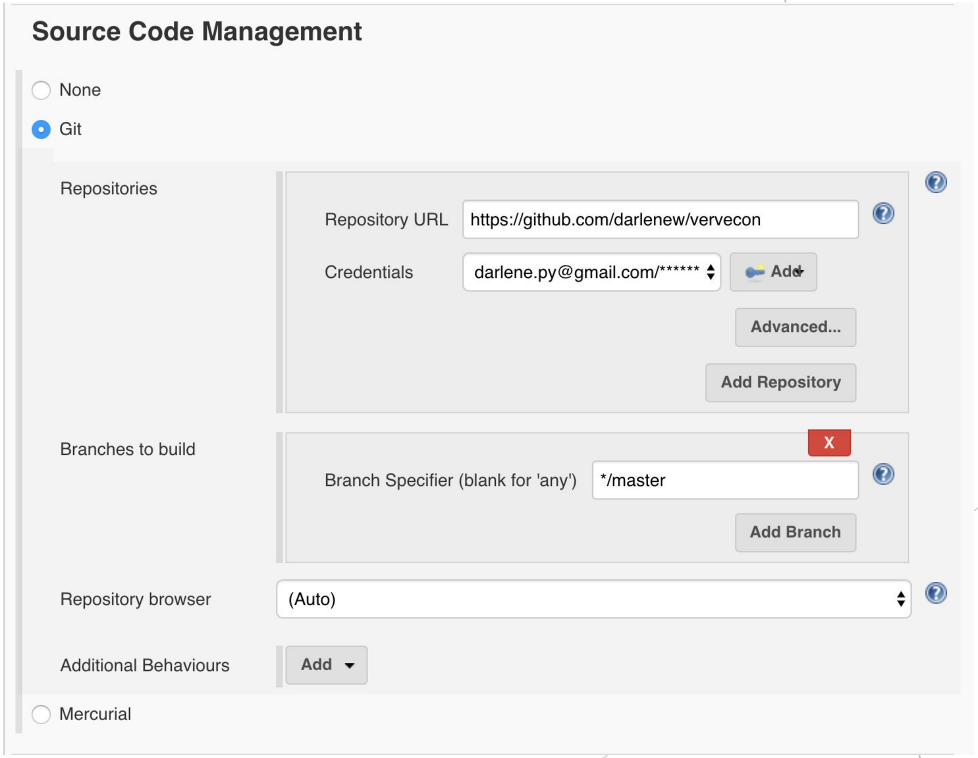
Administrator password

································

- Browse to `0.0.0.0:8080`
- Unlock Jenkins with initial Administrator password
- Go to **admin -> Configure** to change password
- Install plugin GitHub Integration Plugin

Jenkins Setup (2/7) – SCM Credentials

- GitHub Repository URL
- Credentials
  - username/password
  - SSH key
- Specify Branch to build/test



Source Code Management

None
[✓ Git]

Repositories
Repository URL: https://github.com/darlenew/vervecon
Credentials: darlene.py@gmail.com/******

Branches to build
Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

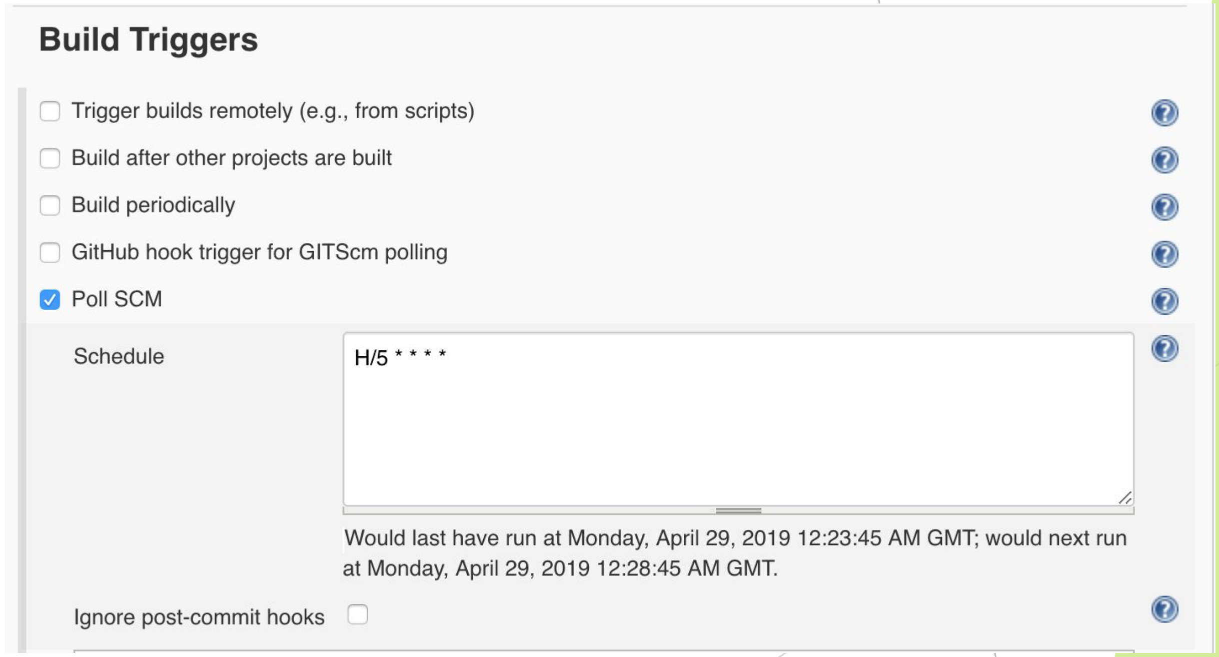
Mercurial

Jenkins Setup (3/7) - Job Trigger

- Want job to run when code is pushed to repository
- Multiple trigger options:
  - **Poll** the repository for changes
  - **Push notification** received when changes occur

Jenkins Setup (4/7) – Trigger Job by Polling

- Select **Poll SCM**
- Schedule polling frequency



Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub hook trigger for GITScm polling
- [✓] Poll SCM

Schedule

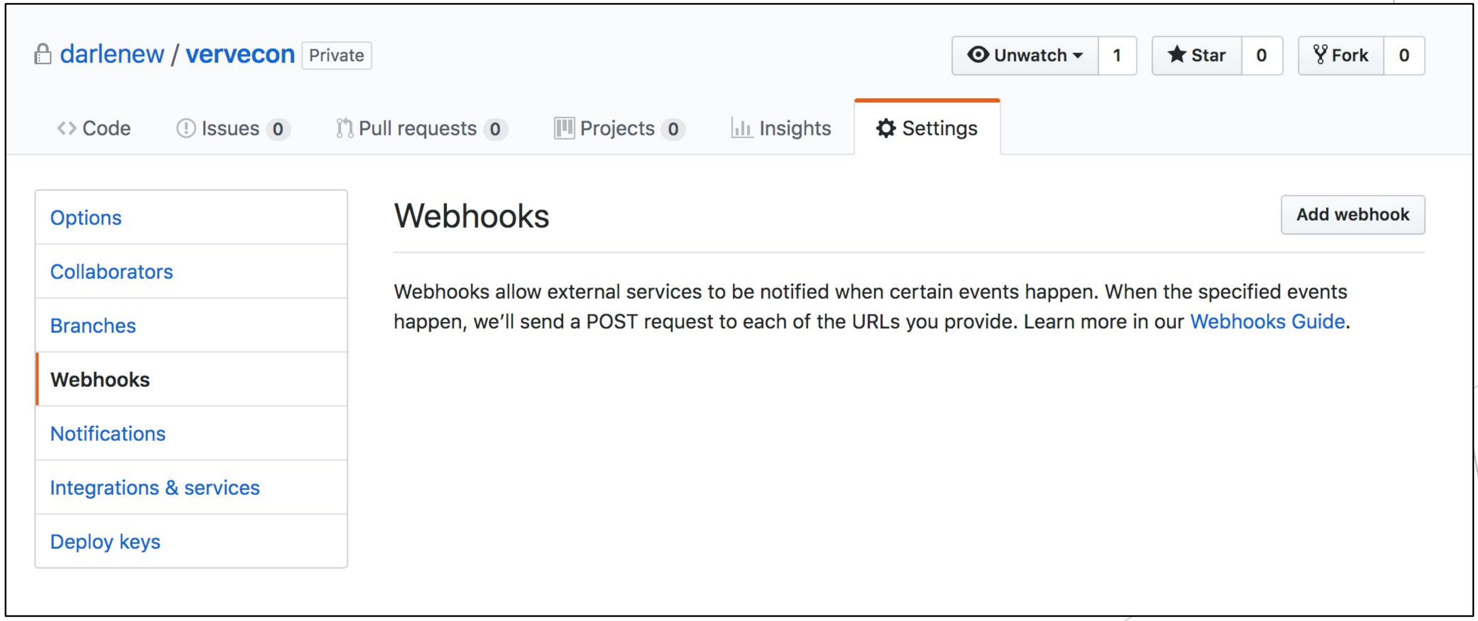
```
H/5 * * * *
```

Would last have run at Monday, April 29, 2019 12:23:45 AM GMT; would next run at Monday, April 29, 2019 12:28:45 AM GMT.

Ignore post-commit hooks [ ]

Jenkins Setup (5/7) – Trigger Job by Webhook

- On github.com, browse to your source repository’s **Settings** tab
- Click “Add webhook”

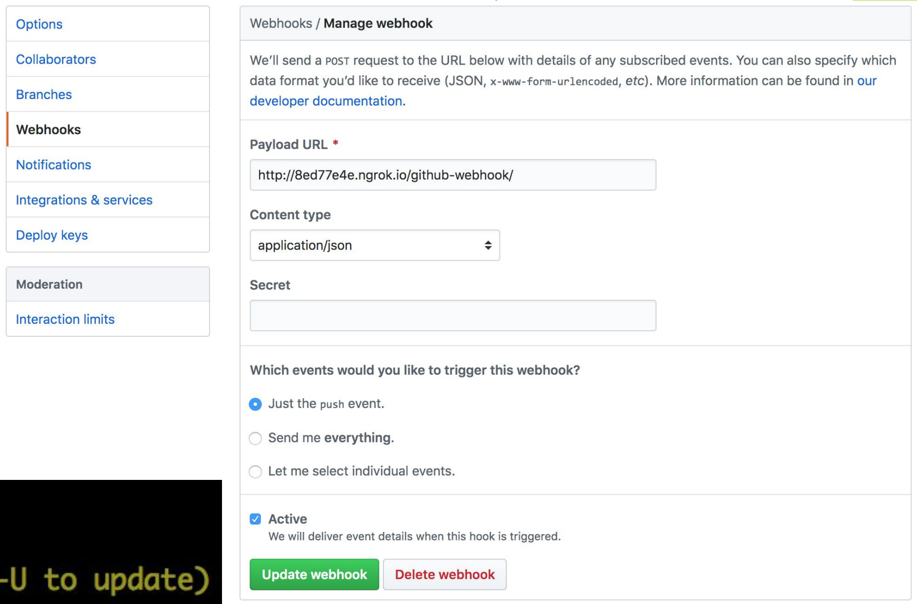


Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we’ll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

Jenkins Setup (6/7) – Configure Webhook

- Payload URL
  - $JENKINS_BASE_URL/github-webhook/
  - ngrok
    - Expose local web server on temporary public URL
    - `$ ngrok http 8080`
    - *e.g.* **http://1a765bb4.ngrok.io/github-webhook/**



Webhooks / **Manage webhook**

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, *etc*). More information can be found in our developer documentation.

Payload URL *
http://8ed77e4e.ngrok.io/github-webhook/

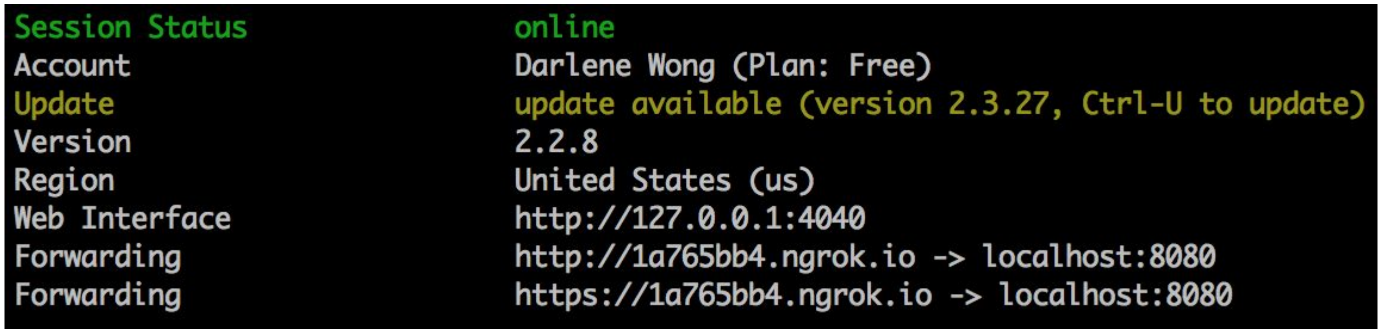
Content type
application/json

Secret

Which events would you like to trigger this webhook?

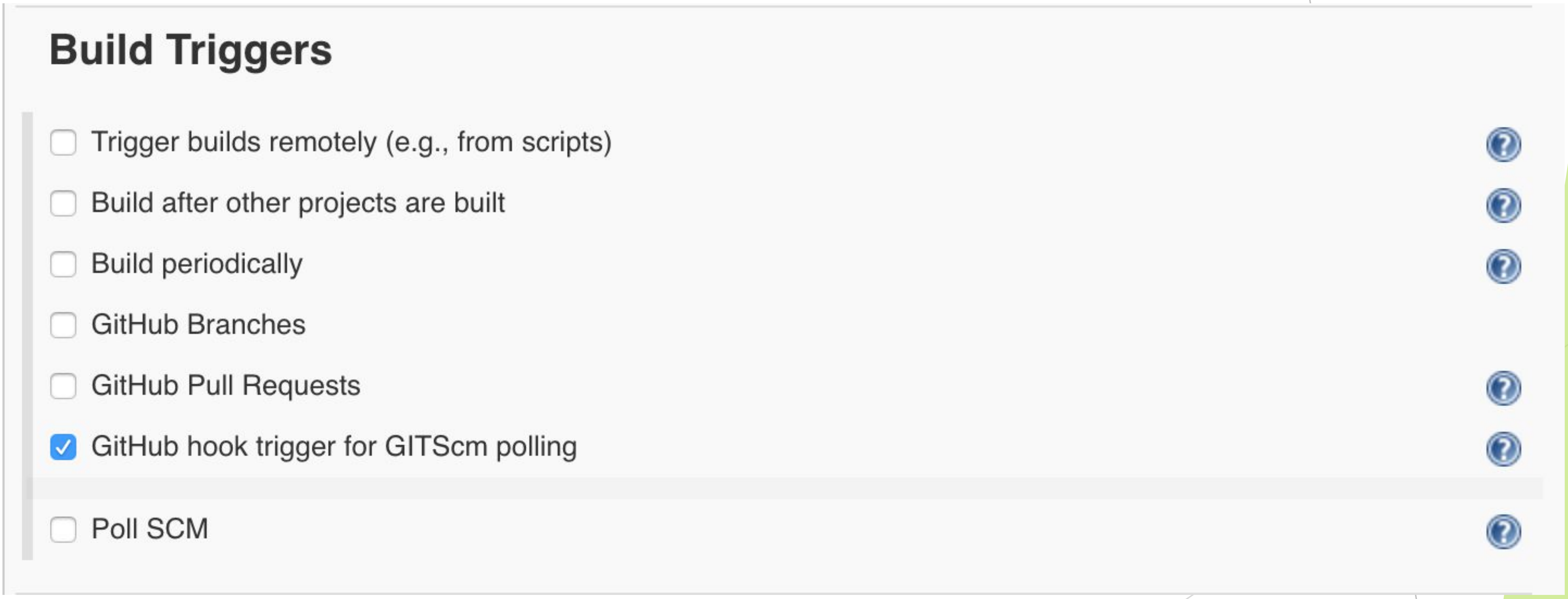
[✓ Just the push event.]
Send me **everything**.
Let me select individual events.

[✓ Active]
We will deliver event details when this hook is triggered.



```
Session Status                online
Account                       Darlene Wong (Plan: Free)
Update                        update available (version 2.3.27, Ctrl-U to update)
Version                       2.2.8
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://1a765bb4.ngrok.io -> localhost:8080
Forwarding                    https://1a765bb4.ngrok.io -> localhost:8080
```

Jenkins (7/7) – Configure Webhook build trigger

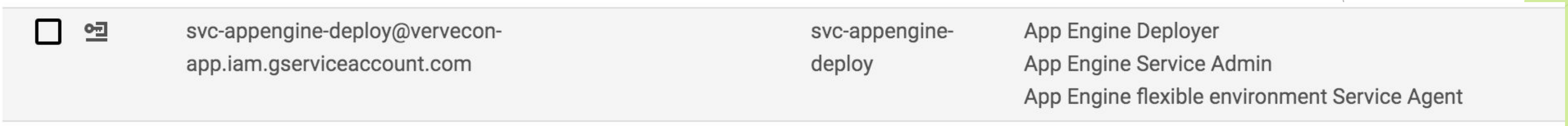


Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub Branches
- [ ] GitHub Pull Requests
- [✓] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

GCP Configuration (1/1)

- Create GCP project: $ `gcloud projects create vervecon-app`
- Initialize app engine app
  - `$ gcloud app create --project=vervecon-app --region=us-central`
- Enable billing on the project in GCP console
- Create service account with deployment permissions

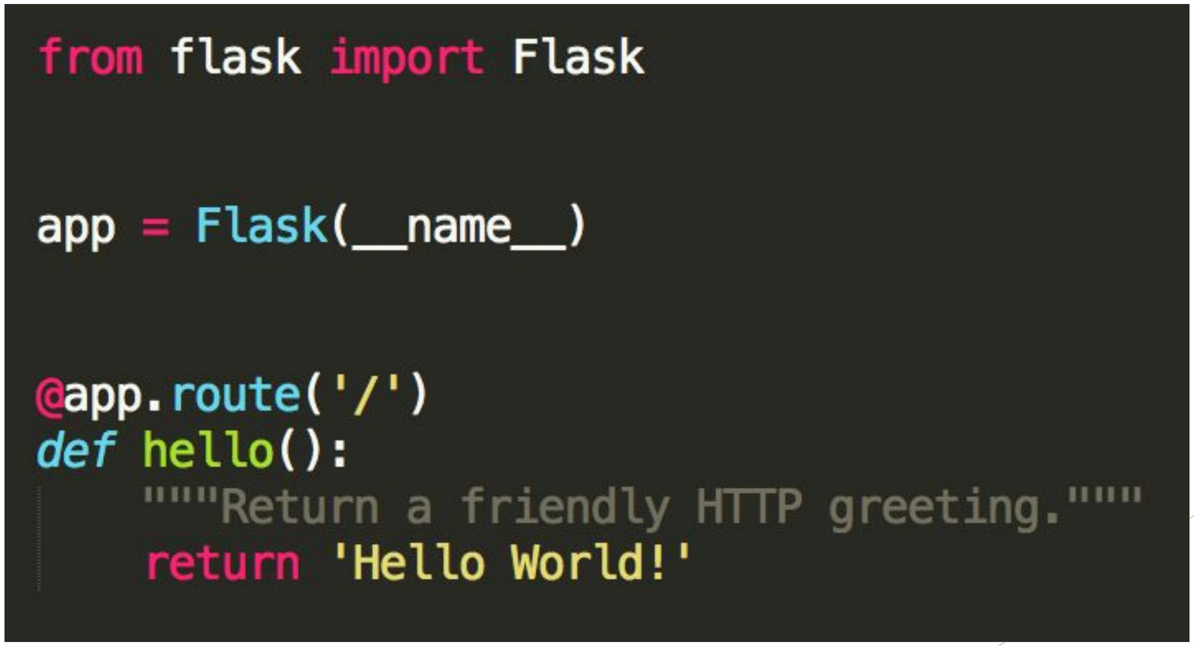


☐	svc-appengine-deploy@vervecon-app.iam.gserviceaccount.com	svc-appengine-deploy	App Engine Deployer App Engine Service Admin App Engine flexible environment Service Agent

- Copy service account key to Docker container
  - `$ docker cp ~/vervecon-app-d1f448f0a2b0.json dd1467d86cb7:/service-key.json`

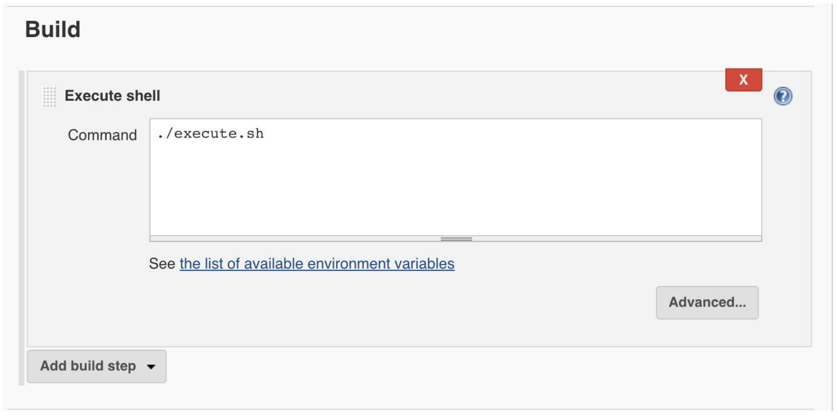
Create Application (1/1)

- Create **Hello World** Flask App Engine application



```
from flask import Flask


app = Flask(__name__)


@app.route('/')
def hello():
    """Return a friendly HTTP greeting."""
    return 'Hello World!'
```

Test and Deploy (1/2)

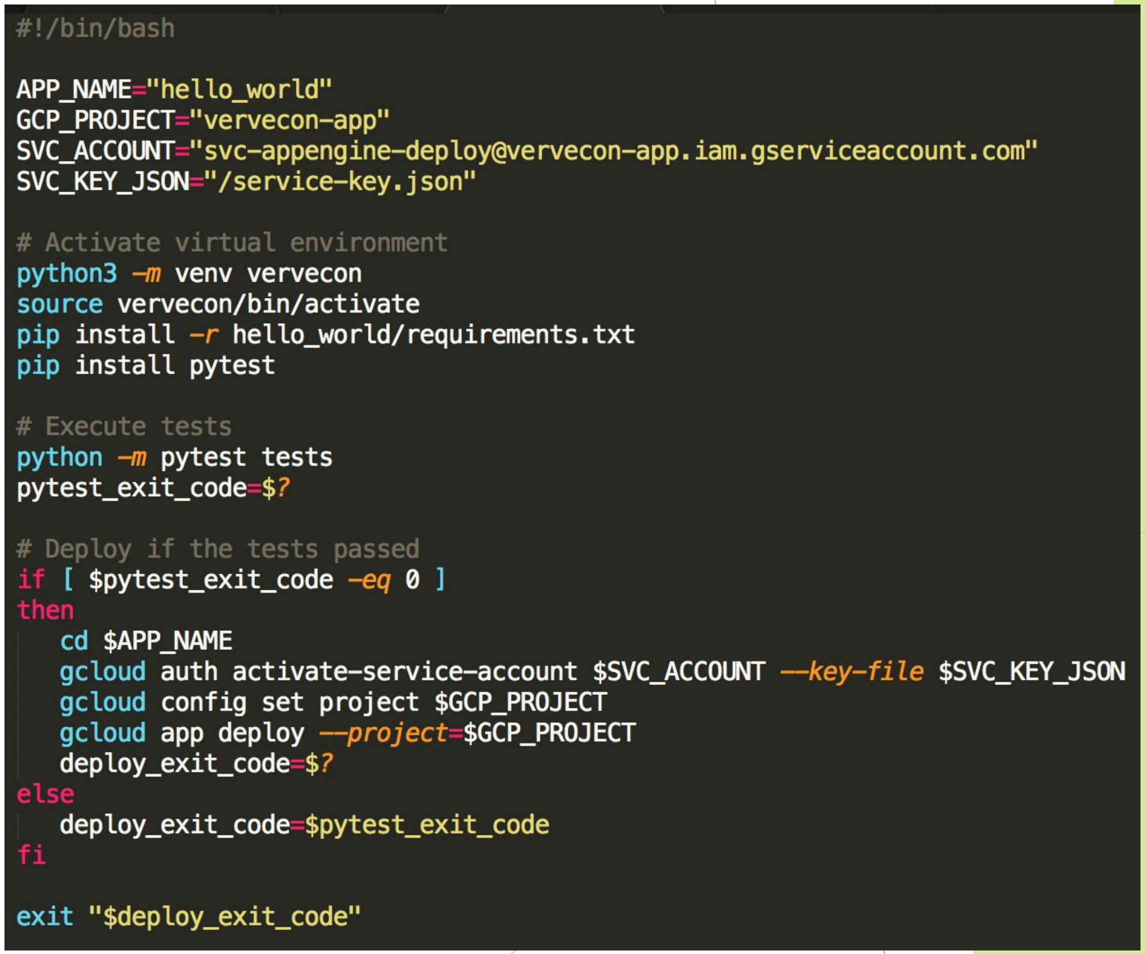


Build

Execute shell

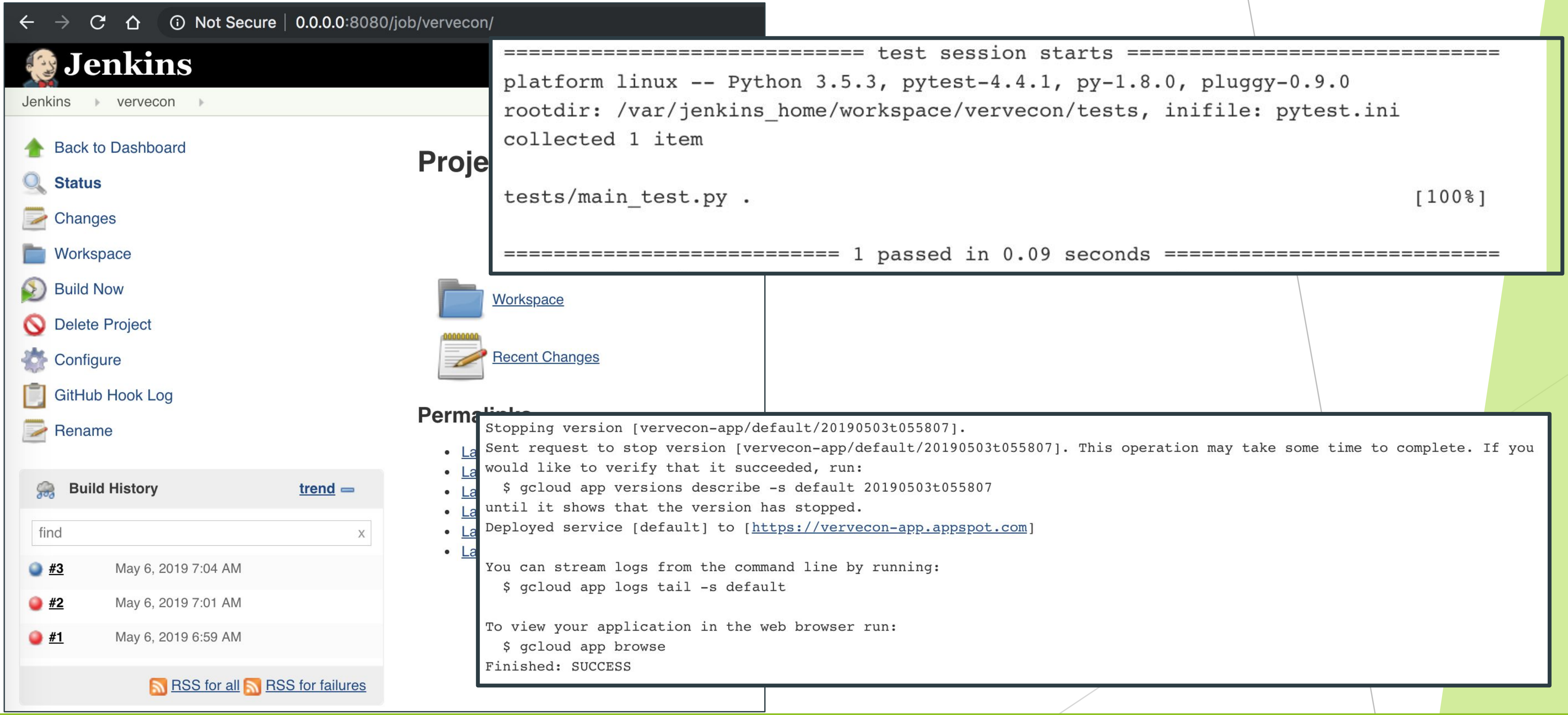
Command
```
./execute.sh
```

See the list of available environment variables



```bash
#!/bin/bash

APP_NAME="hello_world"
GCP_PROJECT="vervecon-app"
SVC_ACCOUNT="svc-appengine-deploy@vervecon-app.iam.gserviceaccount.com"
SVC_KEY_JSON="/service-key.json"

# Activate virtual environment
python3 -m venv vervecon
source vervecon/bin/activate
pip install -r hello_world/requirements.txt
pip install pytest

# Execute tests
python -m pytest tests
pytest_exit_code=$?

# Deploy if the tests passed
if [ $pytest_exit_code -eq 0 ]
then
   cd $APP_NAME
   gcloud auth activate-service-account $SVC_ACCOUNT --key-file $SVC_KEY_JSON
   gcloud config set project $GCP_PROJECT
   gcloud app deploy --project=$GCP_PROJECT
   deploy_exit_code=$?
else
   deploy_exit_code=$pytest_exit_code
fi

exit "$deploy_exit_code"
```

Test and Deploy (2/2)

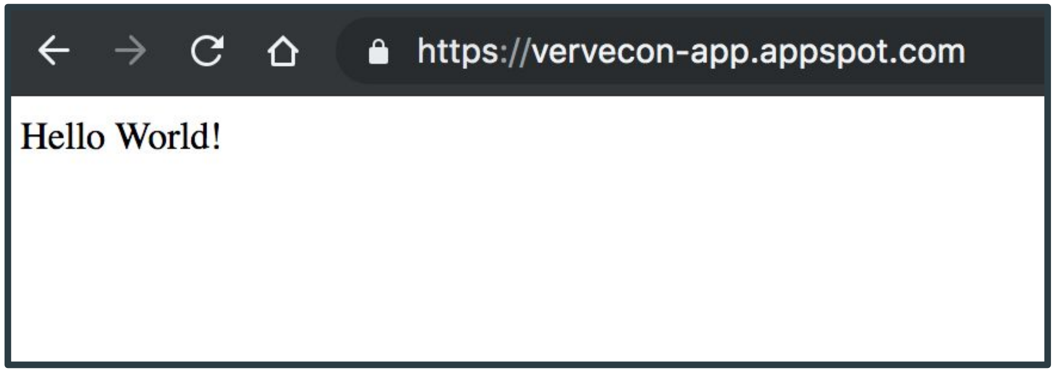


```
============================= test session starts ==============================
platform linux -- Python 3.5.3, pytest-4.4.1, py-1.8.0, pluggy-0.9.0
rootdir: /var/jenkins_home/workspace/vervecon/tests, inifile: pytest.ini
collected 1 item

tests/main_test.py .                                                     [100%]

=========================== 1 passed in 0.09 seconds ===========================
```

```
Stopping version [vervecon-app/default/20190503t055807].
Sent request to stop version [vervecon-app/default/20190503t055807]. This operation may take some time to complete. If you
would like to verify that it succeeded, run:
  $ gcloud app versions describe -s default 20190503t055807
until it shows that the version has stopped.
Deployed service [default] to [https://vervecon-app.appspot.com]

You can stream logs from the command line by running:
  $ gcloud app logs tail -s default

To view your application in the web browser run:
  $ gcloud app browse
Finished: SUCCESS
```

Success!

- Further changes to application will trigger tests
- Successful tests trigger deployment

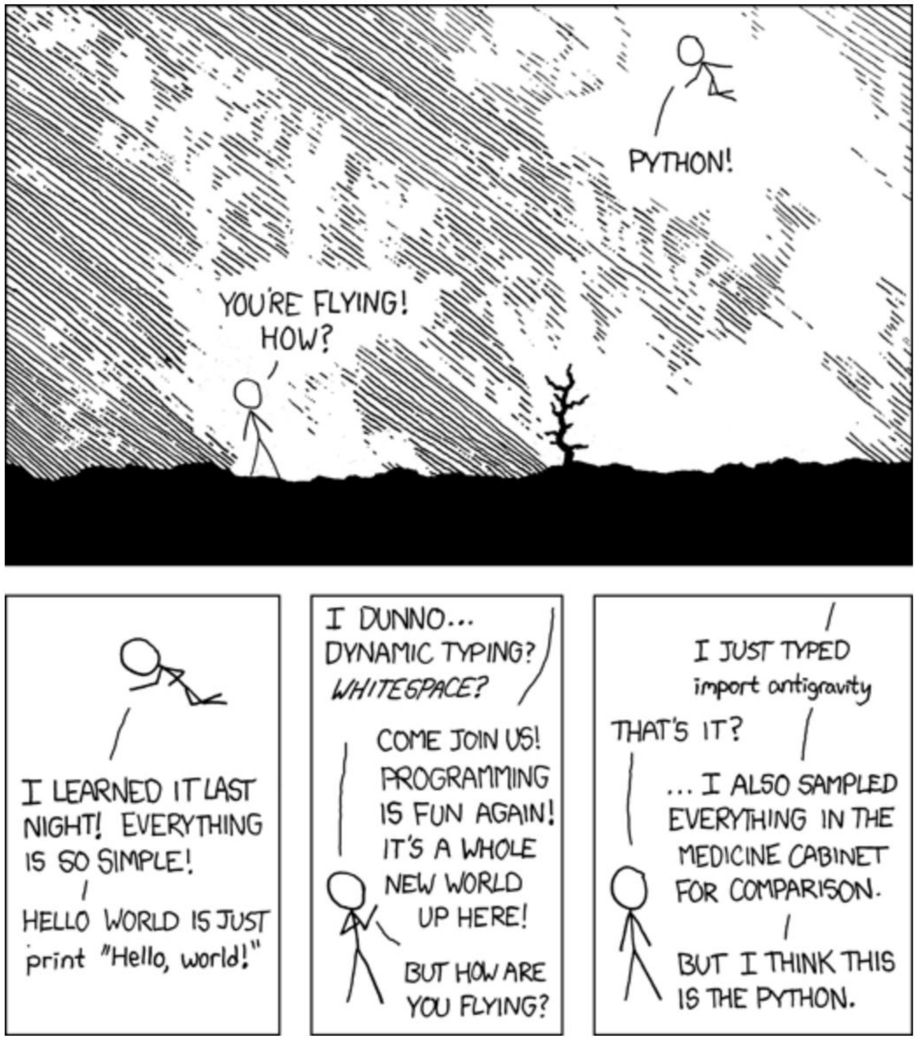

https://vervecon-app.appspot.com

Hello World!

Thank You!

https://github.com/darlenew/vervecon

yyang@paloaltonetworks.com
darlene.py@gmail.com



YOU'RE FLYING! HOW?
PYTHON!
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

References

- https://jenkins.io/doc/book/installing/
- https://ngrok.com/download
- https://developer.github.com/webhooks/
- https://wiki.jenkins.io/display/JENKINS/Github+Plugin