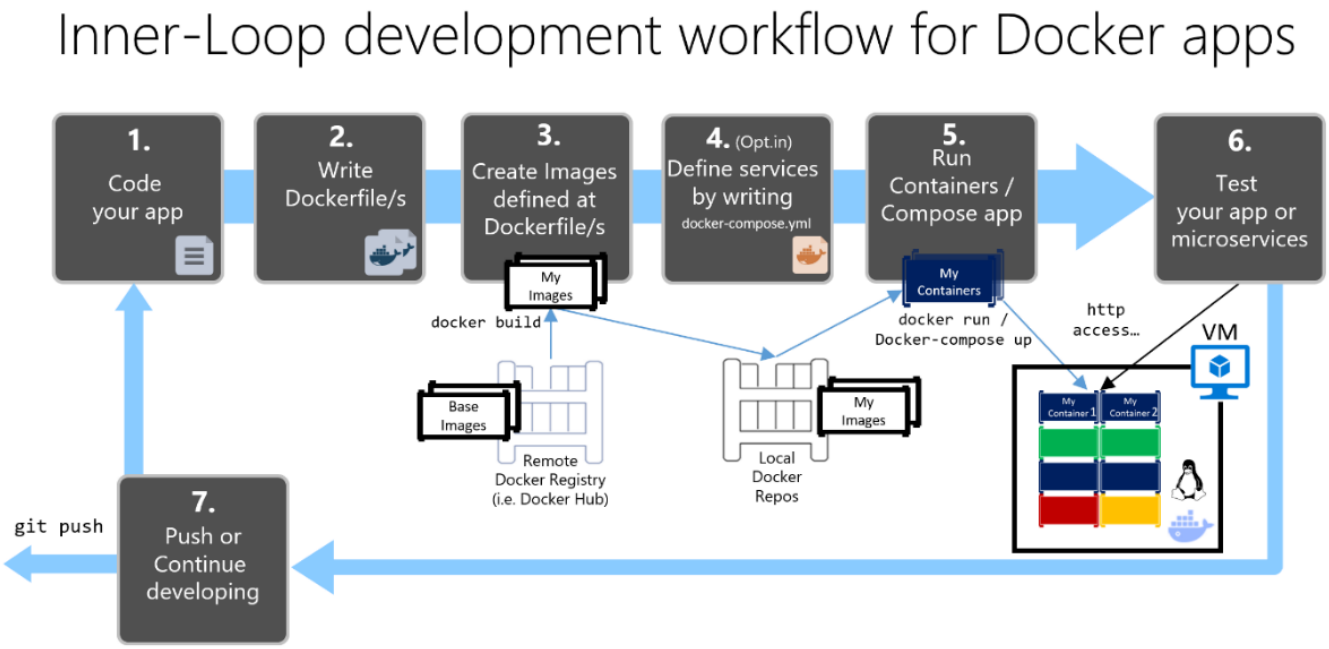
Curso Python, Django, Docker, Kubernetes

TODO: reconvertir ejemplo GCE a AWS, quitar kompose.

Objetivos

- Acostumbrarse al ciclo de integración continua que propone Docker y Kubernetes
- Obtener soltura en Linux, MySQL y Python
- Portar fácilmente aplicaciones hacia la nube.

Flujo de trabajo



Clase 1 - Introducción

Entorno de trabajo

Se propone con el alumno trabajar lo más posible con su propio equipo.

A partir de la segunda clase, se trabajará con Docker. Los usuarios de Linux y de Mac no tendrán mas complicación que correr unos pocos comandos. Pero la instalación de Docker en Windows puede ser complicada debido a versiones, caracteristicas del procesador y otras situaciones. Se proveen instrucciones, y se espera que la segunda clase el alumno concurra con esta herramienta instalada. Si no llega a tiempo, debe solicitar una clase extra de apoyo.

En todos los casos, para todos los sistemas operativos, tambien hará falta que el alumno habilite las opciones de Virtualización en la BIOS de su computadora.

Demostración - breve introducción

Docker puede trabajar en forma imperativa, y complementariamente, en forma declarativa

Fuente: https://docker-curriculum.com/#webapps-with-docker

Ejemplo de comandos imperativos

Bajamos una imagen desde Dockerhub, la instanciamos en memoria, y la ponemos a correr en el puerto 8888

```
docker run -p 8888:5000 --name contenedor_michos prakhar1989/catnip
```

Para verlo corriendo, lanzamos un navegador en http://localhost:8888

> Nota: si en lugar de Docker para Linux o de Docker para Windows se trabaja con Docker Toolbox (ver mas adelante), se debe obtener la ip interna donde está corriendo el contenedor, mediante el programa lazydocker, o con el comando

```
docker inspect contenedor_michos | grep IPAddress

Ejemplo: "IPAddress": "172.17.0.3"
```

Ahora si, lanzar un navegador en http://172.17.0.3:5000

Ejemplo de personalización en forma declarativa

Supongamos que queremos modificar algo de este imagen que obtuvimos en Dockerhub. Primero debemos obtener su Dockerfile:

```
cd /tmp
git clone --depth=1 https://github.com/prakhar1989/docker-curriculum
cd docker-curriculum/flask-app
```

- Como desarrolladores, modificamos el app.py de acuerdo a nuevas necesidades, si las hubiera, requirements.txt, templates/index.html, etc
- Como devops, modificamos el Dockefile a antojo.

Creamos una nueva, y nuestra, imagen local

```
docker build . -f Dockerfile -t mi-imagen-catnip
```

Notar la diferencia entre la imagen que bajamos de Dockerhub, y la que hicimos localmente

```
$ docker images | grep catnip

mi-imagen-catnip
```

```
latest              e984937991cb        29 minutes ago      926MB
prakhar1989/catnip
latest              34ca2beec464        2 months ago        699MB
```

Borramos el contenedor en memoria "contenedor_michos"

```
docker stop contenedor_michos
docker rm   contenedor_michos
```

Instanciamos nuestra propia imagen

```
docker run -p 8888:5000 --name contenedor_michos mi-imagen-catnip
```

Por curiosidad, intervenimos el contenedor corriendo, desde otra terminal:

Primero listamos los contenedores corriendo:

```
$ docker ps | grep catnip
d3918dfe2e32        mi-imagen-catnip                  "python ./app.py"
58 seconds ago      Up 56 seconds       0.0.0.0:8888->5000/tcp
contenedor_michos
```

Ahora entramos con

```
$ docker exec -it contenedor_michos sh

# ls -l
total 20
-rw-r--r-- 1 root root  299 Feb 18 15:34 Dockerfile
-rw-r--r-- 1 root root  199 Feb 18 15:34 Dockerrun.aws.json
-rw-r--r-- 1 root root 1996 Feb 18 15:34 app.py
-rw-r--r-- 1 root root   11 Feb 18 15:34 requirements.txt
drwxr-xr-x 2 root root 4096 Feb 18 15:36 templates

# pwd
/usr/src/app
```

Docker en Windows

Los usuarios de Windows Home no podrán instalar Docker for Windows Desktop edition, pues este requiere Hyper-V virtualization, como Windows Professional y Enterprise editions. Usuarios de otros Windows no

oficiales de tipo MiniOS, u "All in One", probablemente tampoco terminarán correctamente la instalación, o Docker se negará a arrancar.

Los usuarios de Windows Home users deben instalar Docker Toolbox, que usa VirtualBox en su lugar: https://docs.docker.com/toolbox/toolbox_install_windows/ - release downloads available here: https://github.com/docker/toolbox/releases - Toolbox instalará todo lo necesario, incluyendo VirtualBox.

Una vez que Docker for Windows o Docker Toolbox terminen de instalarse, el alumno debe probar lanzar un pequeño contenedor desde la línea de comandos:

```
docker run hello-world
```

Esto debería arrojar, tras terminar, un mensaje **hello-world**

> Una importante diferencia entre Docker Desktop for Windows vs. Docker Toolbox es que ya no podrá usar localhost. En su lugar, deberá acceder a su maquina mediante la dirección 192.168.99.100

Docker en el aula: sesión de Linux

En caso que la instalación en Windows se complique mucho, se ofrece incluida en el curso, una sesión en Linux en server propio, con el cual trabajar. Esta sesión conviene personalizarla para hacer mas amena la instrucción.

Los DevOps trabajan la mayor parte del tiempo en la terminal. Por lo tanto conviene setear ésta acordemente. E incluso si el alumno ha logrado setear correctamente su entorno en Windows, se explicarán los pasos necesarios: los devops pasan una gran parte del día conectados remotos, y conviene de hacerse de un entorno amigable para poder trabajar rápidamente y sin errores. Por el camino, hay una gran cantidad de meta conocimiento o de "residuo cognitivo" necesario para desempeñarse adentro de los contenedores con Linux.

Manejador de sesiones en consola

Se trata de gestores de sesiones, que conviene instalar con apt-get, yum, pacman, etc, y aprender a usarlos en sus páginas oficiales: Screen, Tmux o Byobu.

Para los ejemplos en clase, se trata de usar byobu.