

INFRAESTRUCTURA




BAJO TU CONTROL

OSCAR DAVID ARBELÁEZ

SOBRE MI

- Desarrollador full stack @ AdRoll via BairesDev
- github.com/odarbelaeze

ME APASIONA:










- Los contenedores 
- La integración continua
- La entrega continua 

- La física (MSc. en Física)

NO ME APASIONA:

- Desplegar
manualmente
- Lidar con
inestabilidades

TEMAS

TÉCNICAMENTE HABLANDO

- Docker 
- Travis CI 
- docker-compose 

- docker swarm  

- docker-machine 




FILOSÓFICAMENTE HABLANDO

- Continuous integration
- Continuous delivery
- DevOps

POR QUÉ CONTINUOUS DELIVERY?

*CI/CD Es fundamental para la
productividad de un equipo.*

MI EQUIPO

- Sprints de 1 semana
- Usualmente no desplegamos los viernes
- Una tarea hecha =  
- Cada tarea pasa por:
 - Desarrollo
 - Peer review
 - Despliegue a Staging
 - Despliegue a Producción

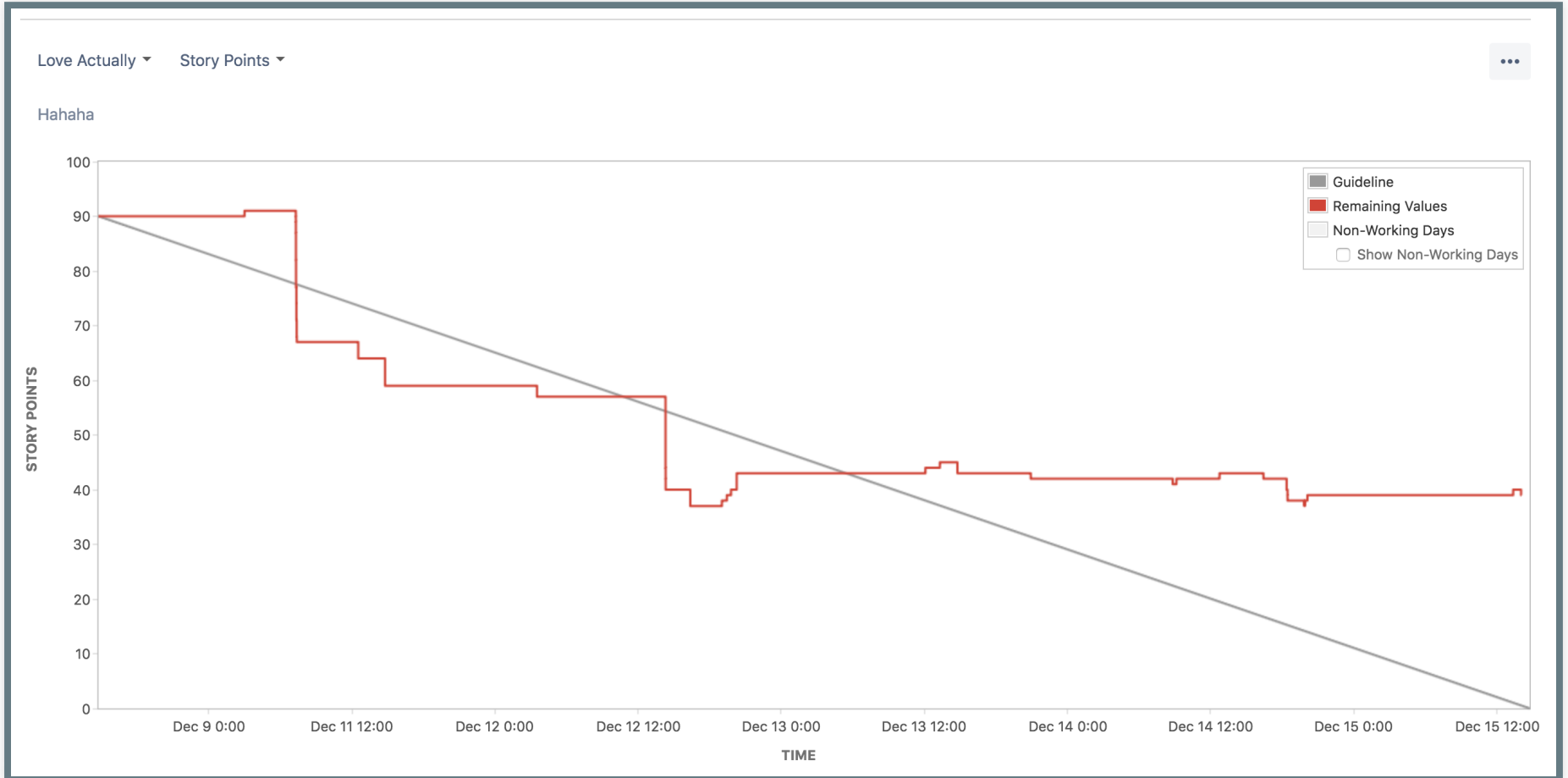
ANTES DE ADOPTAR CD



1 SEMANA DESPUES



2 SEMANAS DESPUES



QUÉ QUEREMOS?

MICRO SERVICIOS



DESPLIEGUES CON ZERO DOWNTIME





ROLLBACKS CON POCO ESRUERZO

BALANCEO DE CARGA





ESCALAMIENTO INDEPENDIENTE...

...EN VARIOS SERVIDORES



ARCHIVOS DE CONFIGURACION ENTENDIBLES 🙏

EJEMPLO

- 2 Micro servicios
 - 1 UI (React )
 - 1 API (Python )
- 1 Balanceador de carga (docker flow proxy)
- 1 Repositorio para el despliegue
- Desplegados en DigitalOcean

PASO 1: CONSTRUIR CONTENEDORES

API

github.com/odarbelaetze/infra-quotes

```
FROM python:alpine
ENV GUNICORN_WORKERS 2
ENV GUNICORN_BACKLOG=4096
RUN pip install gunicorn json-logging-py
COPY ./ /src/app
RUN pip install /src/app
EXPOSE 8000
CMD [ \
    "/usr/local/bin/gunicorn", \
    "--config", "/src/app/config/gunicorn.py", \
    "--log-config", "/src/app/config/logger.conf", \
    "-b", ":8000", \
    "quotes.app:APP" \
]
```



github.com/odarbelaetze/infra-quotes-ui

```
FROM node:carbon-alpine as builder
ADD . /src
WORKDIR /src
ENV NODE_ENV=production
RUN npm install && npm run build

FROM nginx:alpine
COPY config/nginx.conf /etc/nginx/conf.d/default.conf
COPY --from=builder /src/build /code/
EXPOSE 80
```

PASO 2: ENVIAR LOS CONTENEDORES A UN REGISTRO



hub.docker.com/r/odarbelaetze/quotes/

```
echo "$DOCKER_PASSWORD" | docker login \  
-u "$DOCKER_USERNAME" --password-stdin  
  
docker tag \  
  odarbelaetze/quotes:$TRAVIS_COMMIT \  
  odarbelaetze/quotes:latest  
  
docker push odarbelaetze/quotes:$TRAVIS_COMMIT  
docker push odarbelaetze/quotes:latest
```



hub.docker.com/r/odarbelaetze/quotes-ui/

```
echo "$DOCKER_PASSWORD" | docker login \  
-u "$DOCKER_USERNAME" --password-stdin  
  
docker tag \  
  odarbelaetze/quotes-ui:$TRAVIS_COMMIT \  
  odarbelaetze/quotes-ui:latest  
  
docker push odarbelaetze/quotes-ui:$TRAVIS_COMMIT  
docker push odarbelaetze/quotes-ui:latest
```

PASO 3: DOCKER COMPOSE

DESCRIBE TUS SERVICIOS

github.com/odarbelaetze/infra-production

```
version: '3'
services:
  quotes:
    image: odarbelaetze/quotes:6a5c3019debc0b2c1bbeaabb7df8ba8f53
    environment:
      - QUOTES_PREFIX=/api
    networks:
      - proxy
  deploy:
    replicas: 3
    labels:
      - com.df.notify=true
      - com.df.servicePath=/api
      - com.df.port=8000
```

DALE UNA PROBADA

#ItWorksOnMyMachine

```
docker swarm init
docker network create --driver overlay proxy
docker stack deploy -c balancer/docker-compose.yml balancer
docker stack deploy -c quotes/docker-compose.yml quotes
```


PASO 4: CONSTRUYE TO INFRAESTRUCTURA

docker-machine

```
docker-machine create \  
  --driver digitalocean \  
  --digitalocean-image ubuntu-16-04-x64 \  
  --digitalocean-ssh-key-fingerprint <...> \  
  --digitalocean-ssh-key-path ci/keys/id_rsa \  
  --digitalocean-access-token $DOTOKEN \  
manager
```

```
docker-machine ssh manager \  
  "docker swarm init ..."
```

```
docker-machine ssh manager \  
  "docker network create --driver overlay proxy"
```

NOTA: WORKER NODES

- Se pueden crear otras máquinas
- Se pueden conectar con `docker swarm join`
- Docker automáticamente correrá servicios en ellas
- Mantén el balanceador de carga en el manager

```
docker-machine create \  
  --driver digitalocean \  
  --digitalocean-image ubuntu-16-04-x64 \  
  --digitalocean-ssh-key-fingerprint <...> \  
  --digitalocean-ssh-key-path ci/keys/id_rsa \  
  --digitalocean-access-token $DOTOKEN \  
slave
```

```
docker-machine ssh slave \  
  "docker swarm join ..."
```

PASO 5: DESPLIEGE



CONSERVASTE LA IP DEL SERVIDOR 🐱💧

```
scp \
  -o "StrictHostKeyChecking no" \
  -i ci/keys/id_rsa \
  -r \
  balancer/ quotes/ \
  root@$SERVER_IP:
ssh \
  -o "StrictHostKeyChecking no" \
  -i ci/keys/id_rsa root@$SERVER_IP \
  'docker stack deploy -c balancer/docker-compose.yml balancer'
ssh \
  -o "StrictHostKeyChecking no" \
  -i ci/keys/id_rsa root@$SERVER_IP \
  'docker stack deploy -c quotes/docker-compose.yml quotes'
```

DISFRUTA

<http://167.99.57.14/>

NOTAS



Todos estos pasos deben ser ejecutados desde un sistema de integración continua.



Los secretos como tokens, claves SSH y otros que se requieren para el despliegue deben ser manipulados de forma segura.



Manejar secretos es un dolor de cabeza.

OPERACIONES

SI EL TIEMPO LO PERMITE



GRACIAS

Preguntas?

 odarbelaeze@gmail.com