Running Django apps on Docker Cloud

Felix Kerekes - REBS

About me - Felix Kerekes

- Coding in Python for ~8 years
- Started using Docker around 1 year ago

Google

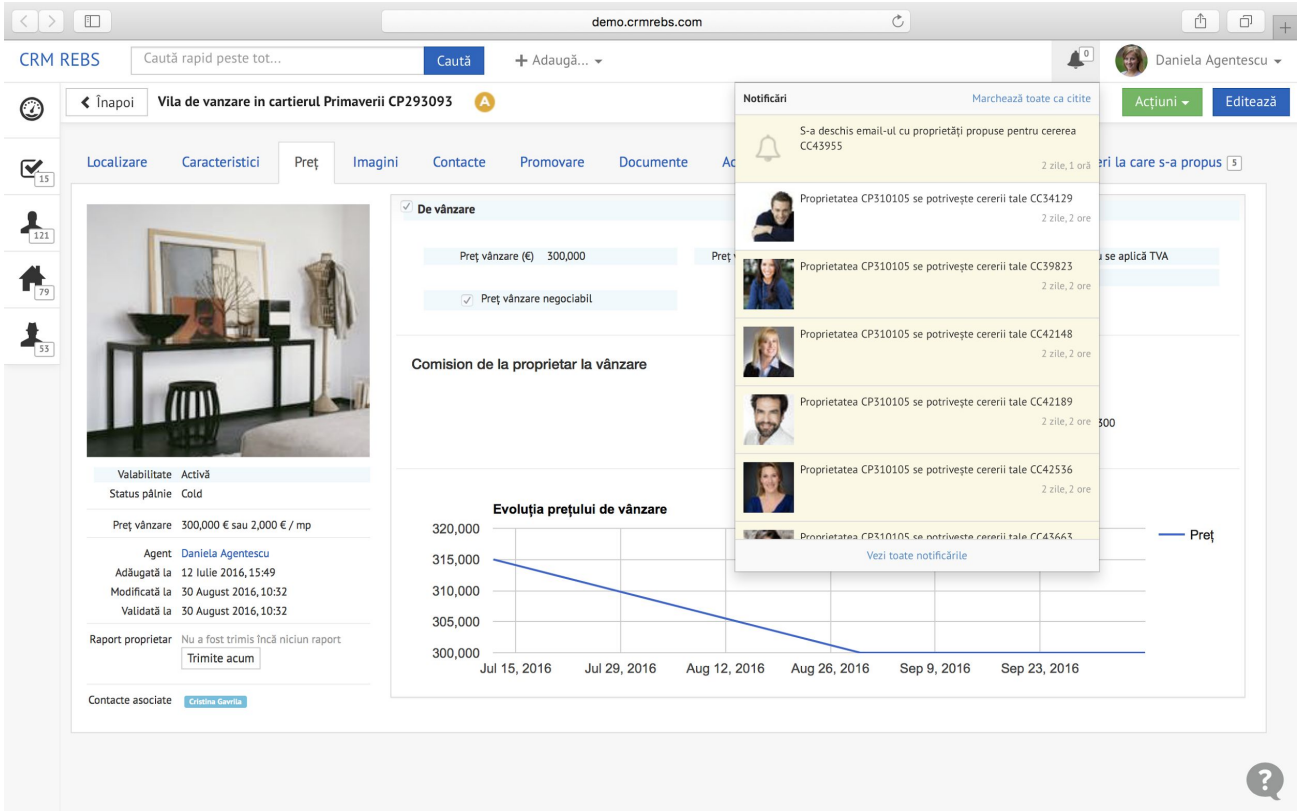
usetogether

rently.ro

REBS

Real Estate Business Solutions

The product - CRM REBS



Early days - Fabric for a single host

```
from fabric.api import *
from fabric.contrib import django

django.project('crmrebs')

env.hosts = ["YOUR_HOST_HERE"]

def deploy(name, tag='master'):
    set_env(name)

    with cd(env.path):
        run('git pull')
        run('git checkout %s' % tag)

    run('%s/bin/pip install -r %s' % (env.venv_path, env.requirements))
```

```
with cd(env.path):
    run_in_virtualenv('manage.py collectstatic --noinput')
    run_in_virtualenv('manage.py compilemessages')
    run_in_virtualenv('manage.py compress -e html -e js --force')
    run_in_virtualenv('manage.py syncdb')
    run_in_virtualenv('manage.py migrate --merge')

sudo('service crm-%s restart' % name)
sudo('service crm-queue restart')
sudo('service crm-queue-priority restart')
```

Going multi-host


```
crmrebs-app:
  git.latest:
    - name: {{ pillar['git']['url'] }}
    - user: {{ pillar['uwsgi']['user'] }}
    - target: {{ pillar['django']['path'] }}
    - identity: {{ pillar['git']['deployment_key'] }}
    - rev: {{ pillar['git']['rev'] }}
    - force_reset: True
    - require:
      - pkg: version-control-pkgs
      - file: deployment-key
```

```
crmrebs-app-virtualenv:
  virtualenv.managed:
    - name: {{ pillar['django']['virtualenv'] }}
    - user: {{ pillar['uwsgi']['user'] }}
    - requirements: {{ pillar['django']['requirements'] }}
    - user: {{ pillar['uwsgi']['user'] }}
    - pip_download_cache: {{ pillar['django']['virtualenv'] }}.cache
    - cwd: {{ pillar['django']['path'] }}
    - watch:
      - git: crmrebs-app
    - require:
      - pkg: crmrebs-app-virtualenv-dependencies
      - git: crmrebs-app
      - sls: python
```

Going multi-host

- First time we could actually deploy configuration & infrastructure changes
- ~ 4000 lines of configuration
  - Managing the whole stack (PostgreSQL, Memcached, Redis, RabbitMQ, uWSGi, nginx)
  - Prod and staging environment
  - Other self-hosted tools (Phabricator, Jenkins, Sentry etc.)
- Harder to scale
- Personally, it’s ugly
  - A lot of templated configuration files

The problems we faced

- Scaling
- Redundancy
- Flexibility

The solution - Docker

The solution - Docker

```
FROM buildpack-deps:xenial

MAINTAINER Felix Kerekes <felix@crmrebs.com>

# Install dependencies
RUN apt-get update && apt-get install -y --no-install-recommends \
  ...
  python-pip \
  sudo \
  && apt-get clean \
  && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

# Upgrade pip and install wheel
RUN pip install -U pip
RUN pip install wheel
```

```
# Install pip requirements
COPY requirements.txt /app/
RUN pip install --no-cache-dir -r requirements.txt

# Add application code
COPY . /app/

# Compile translation files
RUN ./manage.py compilemessages

EXPOSE 8000

CMD ./docker-entrypoint.sh
```

The solution - Docker Cloud

- Formerly Tutum, acquired by Docker
- CaaS - Containers as a Service
- Orchestration - multiple deployment strategies (emptiest node, high availability, every node)
- Multi-host networking
- Volumes
- Terminology
  - Node - tags
  - Stack - Stackfile (fig / docker-compose like)
  - Service
  - Container

Docker Cloud - Stackfiles

```
crmrebs-prod-memcached:
  command: memcached -m 8192
  image: 'memcached:1.4.25'
  restart: always
  tags:
    - high-memory

crmrebs-prod-postgres:
  environment:
    ...
  image: 'rebs/postgresql:9.5-postgis'
  restart: always
  tags:
    - crmrebs-prod
    - ssd
```

```
crmrebs-prod-rabbitmq:
  environment:
    ...
  hostname: rabbitmq
  image: 'rabbitmq:3.6.0-management'
  restart: always
  tags:
    - crmrebs-prod
    - ssd

crmrebs-prod-redis:
  image: 'redis:3.0.7'
  restart: always
  tags:
    - crmrebs-prod
    - ssd
```

```
crmrebs-prod-web:
  deployment_strategy: high_availability
  environment:
    ...
  image: 'rebs/crmrebs:master'
  links:
    - 'crmrebs-prod-memcached:memcached'
    - 'crmrebs-prod-postgres:postgres'
    - 'crmrebs-prod-rabbitmq:rabbitmq'
    - 'crmrebs-prod-redis:redis'
    - 'telegraf.monitoring:telegraf'
  mem_limit: 4096m
  restart: always
  sequential_deployment: true
  target_num_containers: 5
```

Docker Cloud - Stackfiles

```
crmrebs-prod-web-api:
  command: ./docker-entrypoint-api.sh
  deployment_strategy: high_availability
  environment:
    ...
  image: 'rebs/crmrebs:master'
  links:
    - 'crmrebs-prod-memcached:memcached'
    - 'crmrebs-prod-postgres:postgres'
    - 'crmrebs-prod-rabbitmq:rabbitmq'
    - 'crmrebs-prod-redis:redis'
    - 'telegraf.monitoring:telegraf'
  mem_limit: 2096m
  restart: always
  sequential_deployment: true
  target_num_containers: 3
```

```
crmrebs-prod-worker-batch:
  command: celery worker -A crmrebs -Q batch
  deployment_strategy: high_availability
    ...
  image: 'rebs/crmrebs:master'
  links:
    - 'crmrebs-prod-memcached:memcached'
    - 'crmrebs-prod-postgres:postgres'
    - 'crmrebs-prod-rabbitmq:rabbitmq'
    - 'crmrebs-prod-redis:redis'
    - 'telegraf.monitoring:telegraf'
  mem_limit: 4096m
  restart: always
  tags:
    - worker
  target_num_containers: 2
```

```
crmrebs-prod-worker-priority:
  command: celery worker -A crmrebs -Q priority
  deployment_strategy: high_availability
    ...
  image: 'rebs/crmrebs:master'
  links:
    - 'crmrebs-prod-memcached:memcached'
    - 'crmrebs-prod-postgres:postgres'
    - 'crmrebs-prod-rabbitmq:rabbitmq'
    - 'crmrebs-prod-redis:redis'
    - 'telegraf.monitoring:telegraf'
  mem_limit: 4096m
  restart: always
  tags:
    - worker
  target_num_containers: 2
```

Docker Cloud - Multi-host networking

- Overlay networking is now builtin in current Docker versions
  - `overlay` network driver
- Docker Cloud uses Weave
  - Before Docker implemented overlay networks
- All containers connected via a virtual network across all nodes
  - Zero configuration
  - All done seamlessly by Docker Cloud

Docker Cloud - Service discovery

- DNS
  - Service discovery - using service name
  - Load balancing

- dockercloud/haproxy
  - Custom image tailored for Docker Cloud, uses the Docker Cloud API
  - Linked to your frontend Docker services
  - Uses linked service’s environment variables for configuration

Docker Cloud - Service discovery

```
master-haproxy:
  image: 'dockercloud/haproxy:1.0.1'
  deployment_strategy: high_availability
  links:
    - 'crmrebs-prod-web.crmrebs-prod:crmrebs-prod-web'
    - 'crmrebs-prod-web-api.crmrebs-prod:crmrebs-prod-web-api'
    - 'jenkins.jenkins:jenkins'
    ...
  ports:
    - '80:80'
    - '443:443'
  restart: always
  roles:
    - global
  tags:
    - frontend
  target_num_containers: 2
```

```
crmrebs-prod-web:
  image: 'rebs/crmrebs:master'
  ...
  environment:
    - 'HEALTH_CHECK=check inter 2000 rise 3 fall 2'
    - 'HTTP_CHECK=OPTIONS /login/ HTTP/1.1\r\nHost:\ demo.crmrebs.com'
    - 'VIRTUAL_HOST=*crmrebs.com,www.crmrebs.ro'
```

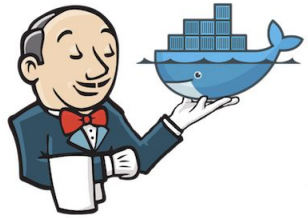
```
crmrebs-prod-web-api:
  image: 'rebs/crmrebs:master'
  ...
  environment:
    - VIRTUAL_HOST=api.crmrebs.com
```

Docker Cloud - Deploying

```
crmrebs-prod-deploy:
  command: ./docker-entrypoint-deploy.sh
  environment:
    ...
  image: 'rebs/crmrebs:master'
  links:
    - 'crmrebs-prod-memcached:memcached'
    - 'crmrebs-prod-postgres:postgres'
    - 'crmrebs-prod-rabbitmq:rabbitmq'
    - 'crmrebs-prod-redis:redis'
    - 'telegraf.monitoring:telegraf'

crmrebs-prod-web:
  deployment_strategy: high_availability
  sequential_deployment: true
  target_num_containers: 5
  ...
```

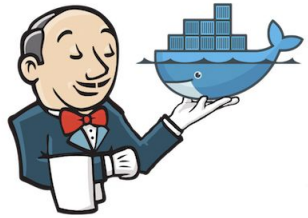
```
# ./docker-entrypoint-deploy.sh

#!/bin/bash -e

echo "Running migrations..."
./manage.py migrate --noinput

echo "Uploading static files and compressing/minimizing static assets..."
./manage.py collectstatic --noinput && ./manage.py compress -e html -e js
--force --engine jinja2 --engine django

echo "Loading permissions..."
./manage.py loaddata crmrebs/fixtures/permissions.json

echo "Loading tags..."
./manage.py loaddata crmrebs/fixtures/tags.json
```

CI & CD with Jenkins & Docker



- Testing jobs
  - Runs tests inside a Docker container, same image used in prod environment
- Build job
  - git pull
  - docker build
  - docker push
- Deploy job - uses Docker Cloud API
  - Update Docker Cloud stack
  - Run one-off deploy container
  - Restart web & worker containers

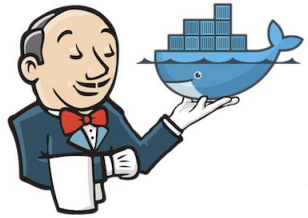
CI & CD with Jenkins & Docker



- Running Jenkins as a Docker service
  - Able to scale Jenkins slaves using Docker containers
  - Master and Slave Docker images

- Using Docker inside Docker containers - Dockerception
  - We do NOT use Docker-in-Docker
  - Link Docker socket and binary to the host's Docker daemon
  - Makes use of the host's Docker cache
  - Problem: mounting a directory as a data volume from within Jenkins will actually mount it on the host

CI & CD with Jenkins & Docker



```
jenkins:
  image: 'rebs/jenkins:latest'
  expose:
    - '8080'
    - '50000'
  links:
    - jenkins-slave
  restart: always
  volumes:
    - '/usr/bin/docker:/usr/bin/docker:ro'
    - '/var/run/docker.sock:/var/run/docker.sock:ro'
```

```
jenkins-slave:
  image: 'rebs/jenkins-slave:latest'
  deployment_strategy: high_availability
  target_num_containers: 2
  volumes:
    - '/usr/bin/docker:/usr/bin/docker:ro'
    - '/var/run/docker.sock:/var/run/docker.sock:ro'
```

Running multiple stacks

- Duplicate a Stackfile
  - Easily create as many isolated environments
- Stack configuration as code - custom deployer using Docker Cloud API

```
crmrebs-staging:
  base: crmrebs
  tag: develop
  base_domain: staging-crmrebs.com

crmrebs-myfeature1:
  base: crmrebs
  tag: myfeature1
  base_domain: myfeature1-crmrebs.com
```

```
crmrebs-myfeature2:
  base: crmrebs
  tag: myfeature2
  base_domain: myfeature2-crmrebs.com
```

Monitoring

- Using Docker to monitor Docker
- InfluxDB, Telegraf, Grafana stack
- Master Telegraf service which aggregates everything and writes to InfluxDB
- Run a Host Telegraf service on every host
  - Monitors host health
  - Monitors information about all running Docker containers

- For extra monitoring, point services to InfluxDB / Master Telegraf container
  - statsd style monitoring

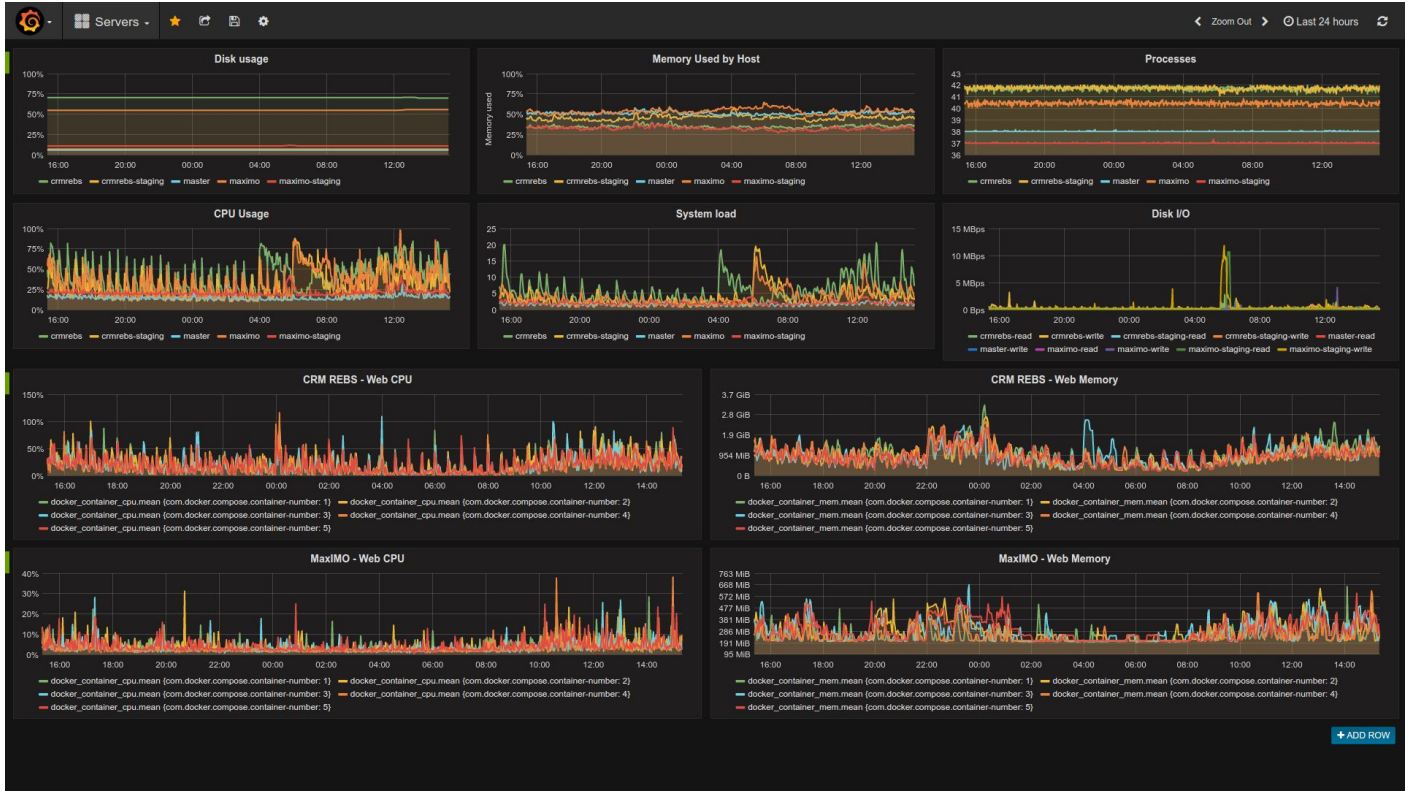
Monitoring

```
telegraf:
  image: 'rebs/telegraf:latest'
  links:
    - 'crmrebs-prod-memcached.crmrebs-prod:crmrebs-prod-memcached'
    - 'crmrebs-prod-postgres.crmrebs-prod:crmrebs-prod-postgres'
    - 'crmrebs-prod-rabbitmq.crmrebs-prod:crmrebs-prod-rabbitmq'
    - 'crmrebs-prod-redis.crmrebs-prod:crmrebs-prod-redis'
    - influxdb
    ...
  restart: always
```

```
telegraf-host:
  image: 'rebs/telegraf-host:latest'
  deployment_strategy: every_node
  privileged: true
  restart: always
  volumes:
    - '/:/rootfs:ro'
    - '/proc:/mnt/host_proc:ro'
    - '/sys:/sys:ro'
    - '/var/lib/docker:/var/lib/docker:ro'
    - '/etc/hostname:/mnt/hostname:ro'
    - '/var/run/docker.sock:/var/run/docker.sock:ro'
```

```
crmrebs-prod-web:
  ...
  links:
    - 'telegraf.monitoring:telegraf'
    ...
```

Monitoring


Servers
Zoom Out
Last 24 hours
Disk usage
Memory Used by Host
Memory used
Processes
CPU Usage
System load
Disk I/O
CRM REBS - Web CPU
CRM REBS - Web Memory
MaxIMO - Web CPU
MaxIMO - Web Memory
crmrebs
crmrebs-staging
master
maximo
maximo-staging
+ ADD ROW

Plans for the future

- Docker Swarm
  - Introduced in Docker v1.12
  - Orchestration
  - Built-in overlay network
  - Plugins
    - Network drivers - overlay
    - Volume plugins - Flocker, convoy

Questions?

Email: felix@crmrebs.com

Slides: http://bit.ly/rebs-docker

REBS

Real Estate Business Solutions