Automated deployments with SaltStack & Docker

Roberto Aguilar, roberto@baremetal.io
How many of us have:
Spent too much time deploying new software?
Spent too much time deploying new software?
Or have answered the question:
“Can we get an install of Cassandra?”
“Can we get an install of ____________Memcached_________?”
“Can we get an install of _____________ RabbitMQ___________?”
“Can we get an install of _____Redis_____?”
"Can we get an install of _________________?"
“maybe next week.”

— Your friendly devops / sysadmin

roberto@baremetal.io
The answer *should* be:
“on it!”

– Your friendly devops / sysadmin
“you can do it yourself!”

–Your friendly devops / sysadmin
How do we get there?
“How to build a dynamic compute environment?”
Dynamic Compute Environment

❖ Easily start and stop services
❖ Experimentation with a low barrier to entry
❖ Scale processes as needed
❖ Unique, isolated application environments
❖ Self-service
Separation of concerns
Separation of concerns

- Host systems are identical
- Host systems are application/service-unaware
- Services are self-contained
The Application Layer
The Twelve-Factor App
http://12factor.net
I. Codebase &
II. Dependencies
V. Build, Release & Run
IV. [Backing] Services
VII. Port-binding
III. Environment-based Config
Application Layer
The nuts and bolts
Compute Environment

- The way to interact with systems
- Server provisioning
- Base software stack
- System configuration
  - logging (syslog config)
  - networking (/etc/hosts, floating IPs, etc.)
  - metrics collection
Application Environment

- Image creation
- Image distribution
- Application runtime
Fill in the blanks
I. Codebase

II. Dependencies
I. Codebase

nginx service repo

```
[0][~/Projects/baremetal/containers/nginx(master)]
[berto@g6]$ find . -type f | grep -v .git
./Dockerfile
./files/etc/apt/nginx.pgp
./files/etc/apt/sources.list.d/nginx.list
./files/etc/nginx/nginx.conf
[…]
```
II. Dependencies

Dockerfile (http://docs.docker.io/en/latest/use/builder/)

- **FROM** - Defines the base image: OS, version, etc.
- **ADD** - Adds files to image
- **RUN** - Commands to configure image
- **EXPOSE** - Specifies exposed ports
- **ENV** - Defines environment variables
- **VOLUME** - Filesystem directories that are sharable
- **CMD** - Default command to run when launched
II. Dependencies


```
FROM ubuntu:quantal
MAINTAINER Roberto Aguilar roberto@baremetal.io

ADD files/etc/apt/nginx.pgp /etc/apt/nginx.pgp
ADD files/etc/apt/sources.list.d/nginx.list /etc/apt/sources.list.d/nginx.list

RUN apt-key add /etc/apt/nginx.pgp
RUN apt-get update
RUN apt-get install -y nginx

EXPOSE 80 443
CMD /usr/sbin/nginx -g 'daemon off;'
```
II. Dependencies

Dockerfile (http://docs.docker.io/en/latest/use/builder/)

FROM ubuntu:quantal
MAINTAINER Roberto Aguilar roberto@baremetal.io
II. Dependencies

Dockerfile (http://docs.docker.io/en/latest/use/builder/)

ADD files/etc/apt/nginx.pgp /etc/apt/nginx.pgp
ADD files/etc/apt/sources.list.d/nginx.list /etc/apt/sources.list.d/nginx.list
II. Dependencies


RUN apt-key add /etc/apt/nginx.pgp
RUN apt-get update
RUN apt-get install -y nginx
II. Dependencies

Dockerfile (http://docs.docker.io/en/latest/use/builder/)

EXPOSE 80 443
II. Dependencies

Dockerfile (http://docs.docker.io/en/latest/use/builder/)

CMD /usr/sbin/nginx -g 'daemon off;'
V. Build, Release & Run
Docker

Builds images

docker build -t <image_name> .

Container runtime

docker run -d <image_name> [command]
Docker Registry

github.com/dotcloud/docker-registry

Host images
docker push <image_name>

Distribute images
docker pull <image_name>
Check out the Docker Index

Ready-made, downloadable images

http://index.docker.io
Anatomy of an image name

registry.local/baremetal/postgresql
Anatomy of an image name

registry.local/baremetal/postgresql

service
Anatomy of an image name

registry.local/barematal/postgresql

user (optional)
Anatomy of an image name

registry.local/baremetal/postgresql

user/service (push to index)
Anatomy of an image name

registry.local/baremetal/postgresql

registry (optional)
Anatomy of an image name

registry.local/baremetal/postgresql

registry (optional)

user (optional)

service
VII. Port-binding
VII. Port-binding

$ docker run -d -p 80 -p 443 registry.local/baremetal/nginx
1052eb879f4e[…]

$ docker ps | chop
CONTAINER ID   IMAGE         PORTS
1052eb879f4e   …/nginx       0.0.0.0:49155->443/tcp, 0.0.0.0:49157->80/tcp

$ alias chop="sed -e 's/ *\|/\|/g' | cut -d'|' -f 1,2,6 | column -s '|' -t"
VII. Port-binding

$ docker run -d -p 80 -p 443 registry.local/baremetal/nginx
1052eb879f4e[…]

$ docker ps | chop
CONTAINER ID  IMAGE     PORTS
1052eb879f4e  […]nginx  0.0.0.0:49155->443/tcp, 0.0.0.0:49157->80/tcp

$ alias chop="sed -e 's/ *\|/\|/g' | cut -d'\|' -f 1,2,6 | column -s '|' -t"
$ docker run -d -p 80 -p 443 registry.local/baremetal/nginx
1052eb879f4e[...]

$ docker ps | chop
CONTAINER ID IMAGE PORTS
1052eb879f4e [...]nginx 0.0.0.0:49155->443/tcp, 0.0.0.0:49157->80/tcp

$ alias chop="sed -e 's/ */\|/g' | cut -d'\|' -f 1,2,6 | column -s ' ' -t"
$ docker run -d -p 80 -p 443 registry.local/baremetal/nginx 1052eb879f4e[...]

$ docker ps | chop
CONTAINER ID  IMAGE          PORTS
1052eb879f4e  [...]nginx    0.0.0.0:49155->443/tcp, 0.0.0.0:49157->80/tcp

$ alias chop="sed -e 's/ */|/g' | cut -d'|' -f 1,2,6 | column -s '|' -t"
VII. Port-binding

$ docker run -d -p 80 -p 443 registry.local/baremetal/nginx
1052eb879f4e[...]

$ docker ps | chop
CONTAINER ID   IMAGE                  PORTS
1052eb879f4e   [...]nginx  0.0.0.0:49155->443/tcp, 0.0.0.0:49157->80/tcp

$ alias chop="sed -e "s/ */|/g" | cut -d'|' -f 1,2,6 | column -s '|' -t"
VII. Port-binding

baremetal@baremetal:~$ docker port 1052eb879f4e 443
0.0.0.0:49155
baremetal@baremetal:~$ docker port 1052eb879f4e 80
0.0.0.0:49157
VII. Port-binding

All interfaces, dynamic host port
-\( p \ 80 \)

All interfaces, explicit host port
-\( p \ 80:80 \)

Explicit interface, dynamic host port
-\( p \ 192.168.42.147::80 \)

Explicit interface, explicit host port
-\( p \ 192.168.42.147:80:80 \)
III. Environment-based Config
III. Environment-based Config

- postgresql://u:p@1.2.3.4:49156/db
- amqp://u:p@1.2.3.35:49901
- memcache://1.2.3.11:49302
- django

pg

rmq

cache
III. Environment-based Config

$ docker run -d \
  -p 1.2.3.42::8000 \
  -e MEMCACHED_URL=memcache://1.2.3.11:49302 \
  -e AMQP_URL=amqp://u:p@1.2.3.35:49901 \
  -e POSTGRESQL_URL=postgresql://u:p@1.2.3.4:49156/db \
registry.local/app1/djangoapp
III. Environment-based Config

settings.py - memcached setup

```python
import os
from urlparse import urlparse

backend = 'django.core.cache.backends.memcached.MemcachedCache'
memcached_url = urlparse(os.environ['MEMCACHED_URL'])
CACHES = {
    'default': {
        'BACKEND': backend,
        'LOCATION': memcached_url.netloc,
    }
}
```
III. Environment-based Config

settings.py - RabbitMQ setup

```python
import os

BROKER_URL = os.environ['AMQP_URL']
```
III. Environment-based Config

settings.py - postgresql setup

```python
import dj_database_url

dj_db_config = dj_database_url.config()
if dj_db_config:
    DATABASES['default'] = dj_db_config
```
Baremetal Platform
Baremetal Platform

❖ Thin as possible
❖ Run the platform on the platform
  ❖ Run as many services as possible in containers
Baremetal Platform

❖ Two main components
  ❖ container manager
  ❖ orchestration
❖ Two major goals
  ❖ minimize downtime
  ❖ eliminate single point of failure
Container Manager

- Starts containers assigned to the host
- Stops containers no longer assigned
- Restarts containers that should be running and die
- Restarts containers when service config changes
Orchestration

- Aware of all hosts in cluster
- Reacts to hosts joining and leaving cluster
- Reacts to configuration changes
- Assigns containers to hosts
Orchestration Configuration

- Repositories
  - tracks Docker image, git repo and branch
- Services
  - repo, dependencies, failover
- Applications
  - services, # instances, configuration, memory allocation
The nuts and bolts
Additional Services

- etcd - fault-tolerant configuration store
- serf - host discovery and message passing
- zfs - next-gen copy-on-write filesystem
Containerize as much as possible
Containerize as much as possible

- Run the platform on the platform
- Services remain self-contained
- Portable - simple to relocate to another system
- Easily deploy new versions and rollback if needed
Thanks!

Let us know what you think:

http://baremetal.io/scale