FROM MONOLITH TO DOCKER DISTRIBUTED APPLICATIONS

Carlos Sanchez

@csanchez

ABOUT ME

Senior Software Engineer @ CloudBees

Author of Jenkins Kubernetes plugin

Long time OSS contributor at Apache Maven, Eclipse, Puppet,…

Linux containers

Filesystem

Users

Processes

Network

BUT IT IS NOT TRIVIAL

Kernel Sanders
@lstoll

The solution: Docker. The problem? You tell me.

OUR USE CASE

Scaling Jenkins

Your mileage may vary

ARCHITECTURE

Docker Docker Docker

Isolated Jenkins masters

Isolated slaves and jobs

Memory and CPU limits

OFFICIAL REPOSITORY

jenkins ★

Last pushed: 11 days ago

Repo Info Tags

Supported tags and respective `Dockerfile` links

- `latest`, `1.609.2` (*Dockerfile*)

For more information about this image and its history, please see the relevant manifest file (`library/jenkins`) in the `docker-library/official-images` GitHub repo.

Jenkins

The Jenkins Continuous Integration and Delivery server.

This is a fully functional Jenkins server, based on the Long Term Support release .

Jenkins

DOCKER PULL COMMAND

```
docker pull jenkins
```

DESCRIPTION

Official Jenkins Docker image

PUBLIC | AUTOMATED BUILD

jenkinsci/jnlp-slave ☆

Last pushed: 6 days ago

Repo Info | Tags | Dockerfile | Build Details

Jenkins JNLP slave Docker image

A Jenkins slave using JNLP to establish connection.
See Jenkins Distributed builds for more info.

Usage :

```
docker run jenkinsci/jnlp-slave -url http://jenkins-server:port <secret> <slave
```

optional environment variables:

- *JENKINS_URL*: url for the Jenkins server, can be used as a replacement to -url option, or to set alternate jenkins URL
- *JENKINS_TUNNEL*: (HOST:PORT) connect to this slave host and port instead of Jenkins server, assuming this one do route TCP traffic to Jenkins master. Useful when when Jenkins runs behind a load balancer, reverse proxy, etc.

How would you design your infrastructure if you couldn't login? Ever.

Kelsey Hightower

EMBRACE FAILURE!

CLUSTER SCHEDULING

Distribute tasks across a cluster of hosts

Running in public cloud, private cloud, VMs or bare metal

HA and fault tolerant

With Docker support of course

APACHE MESOS

A distributed systems kernel

ALTERNATIVES

Docker Swarm / Kubernetes

MESOSPHERE MARATHON

APACHE ZOOKEEPER

TERRAFORM

TERRAFORM

```
resource "aws_instance" "worker" {
    count = 1
    instance_type = "m3.large"
    ami = "ami-xxxxxx"
    key_name = "tiger-csanchez"
    security_groups = ["sg-61bc8c18"]
    subnet_id = "subnet-xxxxxx"
    associate_public_ip_address = true
    tags {
        Name = "tiger-csanchez-worker-1"
        "cloudbees:pse:cluster" = "tiger-csanchez"
        "cloudbees:pse:type" = "worker"
    }
    root_block_device {
        volume_size = 50
    }
```

TERRAFORM

- State is managed
- Runs are idempotent
  - `terraform apply`
- Sometimes it is too automatic
  - Changing image id will restart all instances

@DEVOPS_BORAT

DevOps Borat

To make error is human. To propagate error to all server in automatic way is #devops.

STORAGE

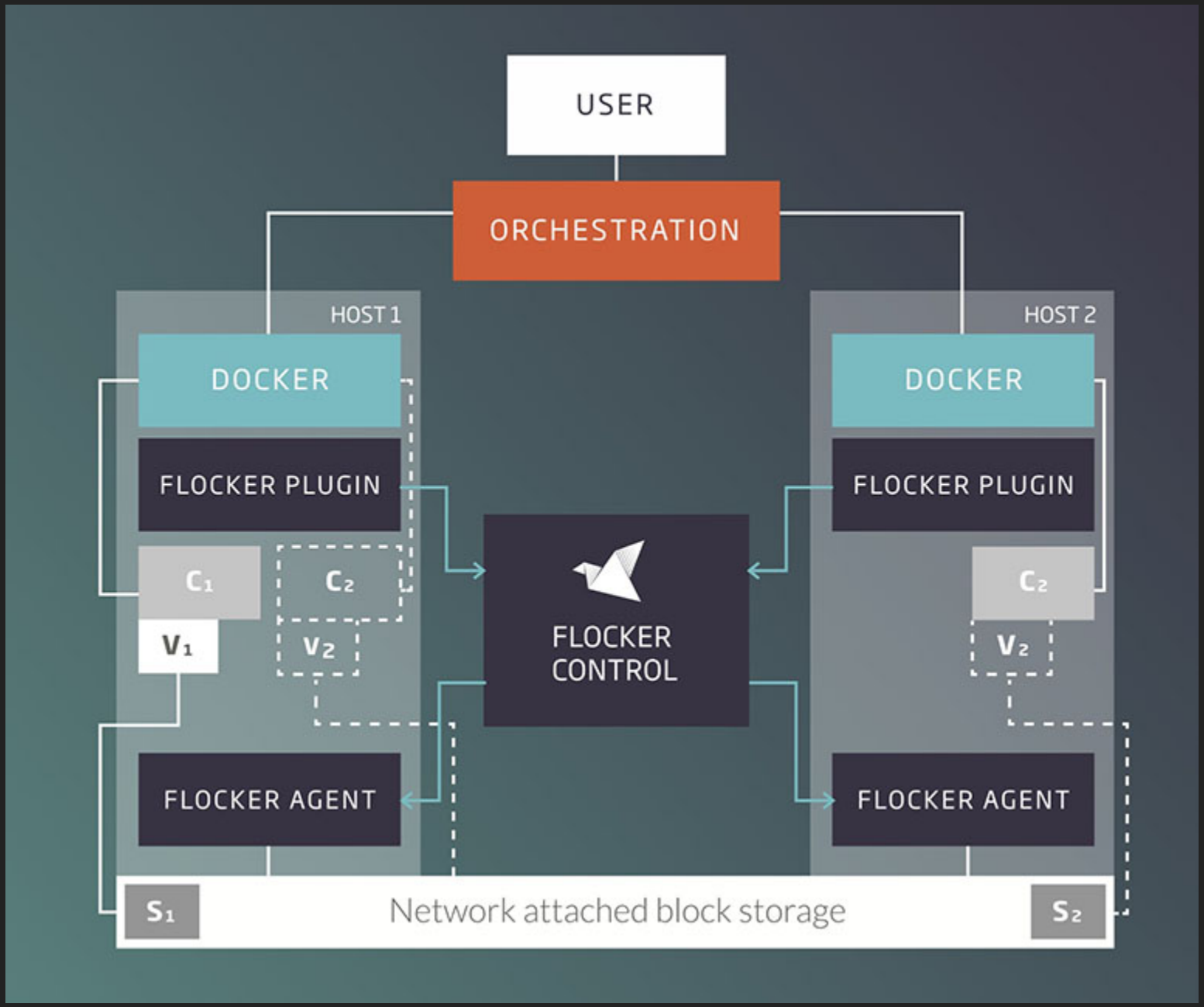
Handling distributed storage

Servers can start in any host of the cluster

And they can move when they are restarted

DOCKER VOLUME PLUGINS

- Flocker
- GlusterFS
- NFS
- EBS


USER
ORCHESTRATION
HOST 1
HOST 2
DOCKER
DOCKER
FLOCKER PLUGIN
FLOCKER PLUGIN
C1
C2
C2
V1
V2
V2
FLOCKER
CONTROL
FLOCKER AGENT
FLOCKER AGENT
S1
Network attached block storage
S2

KUBERNETES

- GCE disks
- Flocker
- GlusterFS
- NFS
- EBS

SIDEKICK CONTAINER

A privileged container that manages mounting for other containers

Can execute commands in the host and other containers

A lot of magic happening with `nsenter`

IN OUR CASE

Sidekick container (*castle* service)

Jenkins masters need persistent storage, slaves (*typically*) don't

Supporting EBS (AWS) and external NFS

CASTLE

- Jenkins master container requests data on startup using *entrypoint*
  - REST call to Castle
- Castle checks authentication
- Creates necessary storage in the backend
  - EBS volumes from snapshots
  - Directories in NFS backend

CASTLE

- Mounts storage in requesting container
  - EBS is mounted to host, then bind mounted into container
  - NFS is mounted directly in container
- Listens to Docker event stream for killed containers

CASTLE: BACKUPS AND CLEANUP

Periodically takes S3 snapshots from EBS volumes in AWS

Cleanups happening at different stages and periodically

EMBRACE FAILURE!

PERMISSIONS

Containers should not run as root

Container user id != host user id

i.e. `jenkins` user in container is always 1000 but matches `ubuntu` user in host

CAVEATS

Only a limited number of EBS volumes can be mounted

Docs say `/dev/sd[f-p]`, but `/dev/sd[q-z]` seem to work too

Sometimes the device gets corrupt and no more EBS volumes can be mounted there

NFS users must be centralized and match in cluster and NFS server

MEMORY

Scheduler needs to account for container memory requirements and host available memory

Prevent containers for using more memory than allowed

Memory constrains translate to Docker --memory

WHAT DO YOU THINK HAPPENS WHEN?

Your container goes over memory quota?


¡CACHONDEO.COM

WHAT ABOUT THE JVM?

WHAT ABOUT THE CHILD PROCESSES?

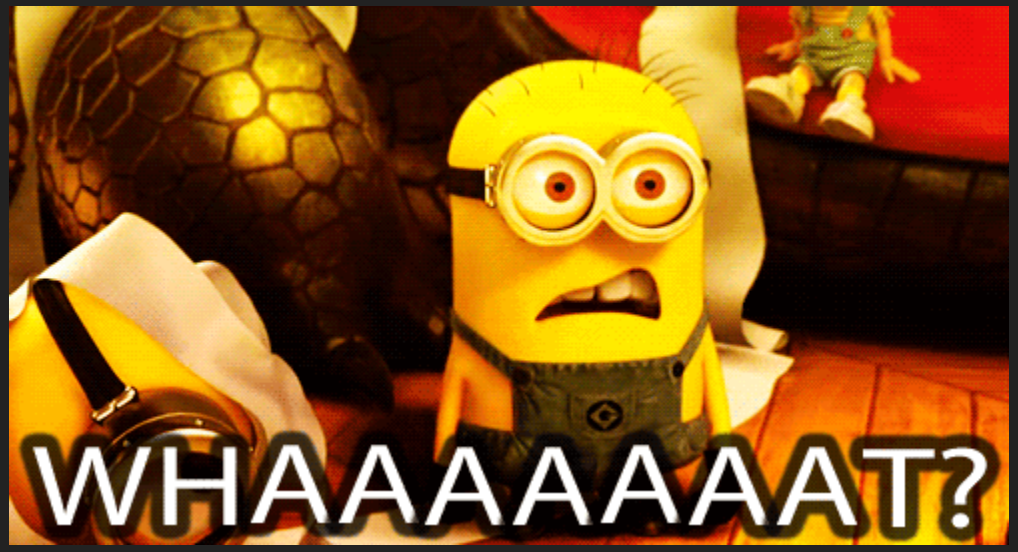
CPU

Scheduler needs to account for container CPU requirements and host available CPUs

WHAT DO YOU THINK HAPPENS WHEN?

Your container tries to access more than one CPU

Your container goes over CPU limits



Totally different from memory

Mesos/Kubernetes CPU translates into Docker `--cpu-shares`

OTHER CONSIDERATIONS

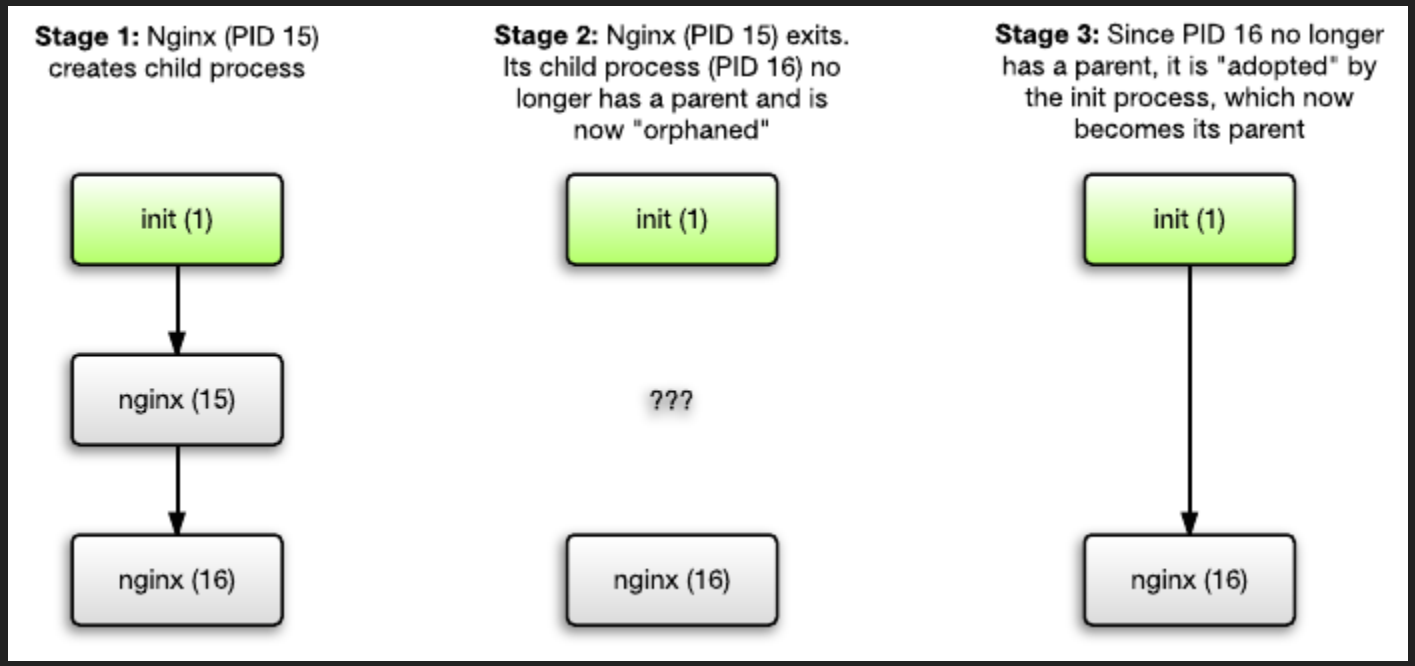
DOCKER AND THE PID 1 ZOMBIE REAPING PROBLEM

https://blog.phusion.nl/2015/01/20/docker-and-the-pid-1-zombie-reaping-problem/

Zombie processes are processes that have terminated but have not (yet) been waited for by their parent processes.

The init process -- PID 1 -- has a special task. Its task is to "adopt" orphaned child processes

PROCESS ADOPTION



THIS IS A PROBLEM IN DOCKER

Jenkins slaves run multiple processes

But Jenkins masters too, and they are long running

TINI

Systemd or SysV init is too heavyweight for containers

> *All Tini does is spawn a single child (Tini is meant to be run in a container), and wait for it to exit all the while reaping zombies and performing signal forwarding.*

PROCESS REAPING

Docker 1.9 gave us trouble at scale, rolled back to 1.8

Lots of *defunct* processes

NETWORKING

Multiple services running in the same ports

Must redirect from random ports in the host

Services running in one host need to access services in other hosts

NETWORKING: SERVICE DISCOVERY

DNS is not great, caching can happen at multiple levels

`marathon-lb` uses `haproxy` and Marathon API

A typical `nginx` reverse proxy is also easy to setup

There are more complete solutions like Consul

NETWORKING: SECURITY

Prevent/Allow

from	to
container	host
container	container
container	another host
container	container in another host

NETWORKING: SECURITY

Prevent/Allow

from	to	
container	host	`iptables`
container	container	`--icc=false` + `--link`, `docker0` bridge device tricks
container	another host	`--ip-forward=false`, `iptables`
container	container in another host	`iptables`

NETWORKING: SOFTWARE DEFINED NETWORKS

Create new custom networks on top of physical networks

Allow grouping containers in subnets

Not trivial to setup

NETWORKING: SOFTWARE DEFINED NETWORKS

Battlefield: Calico, Flannel, Weave and Docker Overlay Network

DOCKER OVERLAY

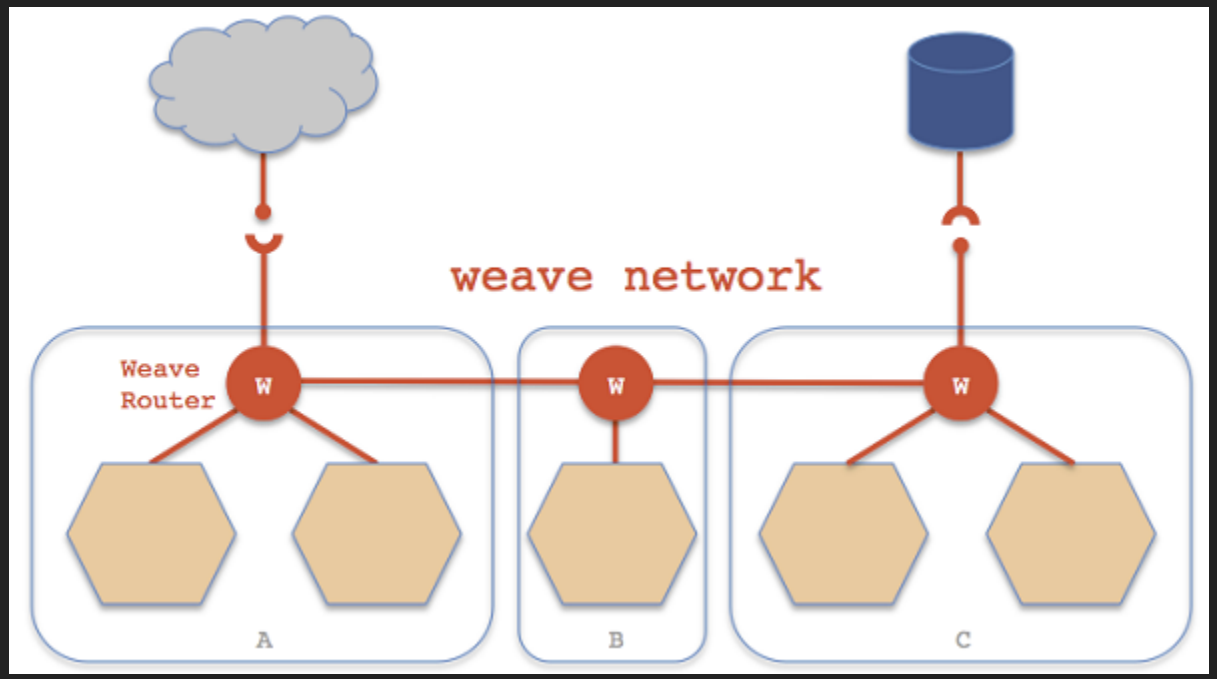
Docker networking with default `overlay` driver, using VxLAN

```
# On the Swarm master
docker network create --driver overlay --subnet=10.0.9.0/24 my-net
```

Uses Consul, etcd or ZooKeeper as key-value stores

WEAVE

UDP and VxLAN backends

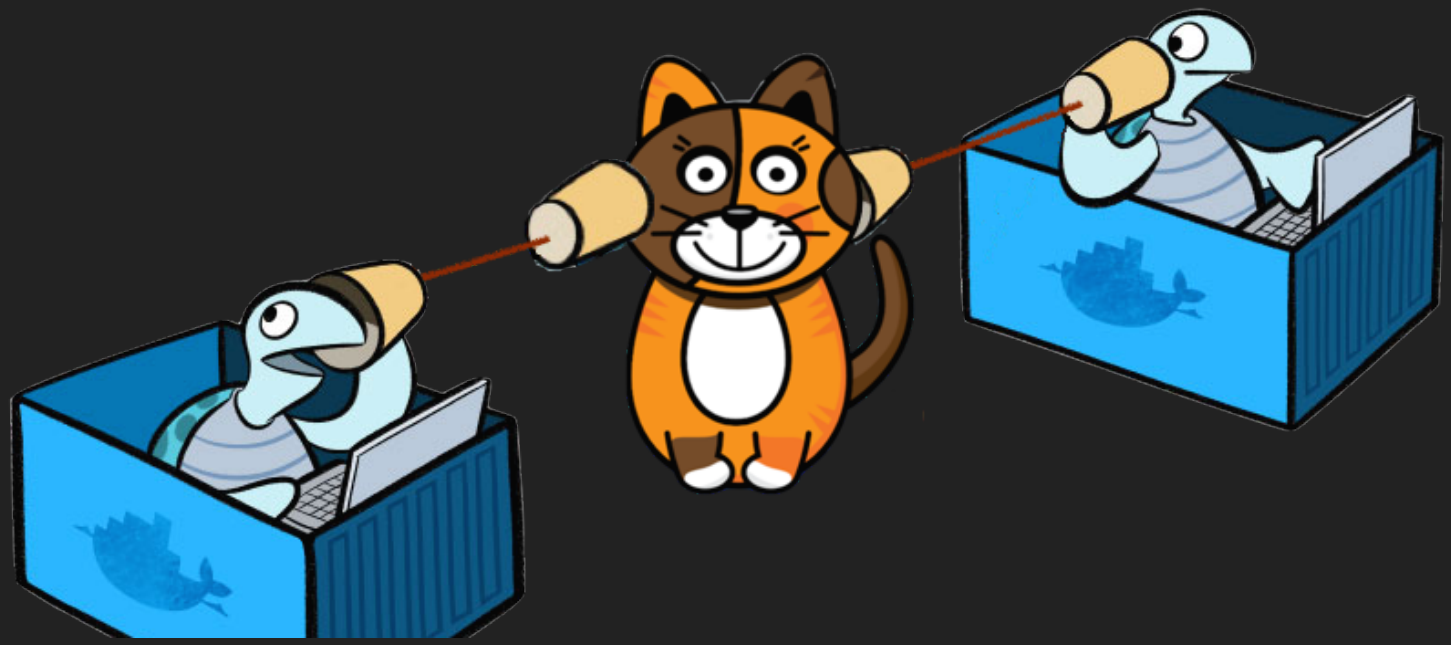


COREOS `FLANNEL`

UDP and VxLAN backends

Uses `etcd` for key-value store

PROJECT CALICO



A pure Layer 3 model

SCALING

New and interesting problems

TERRAFORM AWS

- Instances
- Keypairs
- Security Groups
- S3 buckets
- ELB
- VPCs

AWS

Resource limits: VPCs, S3 snapshots, some instance sizes

Rate limits: affect the whole account

Retrying is your friend, but with exponential backoff

TERRAFORM OPENSTACK

- Instances
- Keypairs
- Security Groups

OPENSTACK

Custom flavors

Custom images

Different CLI commands

There are not two OpenStack installations that are the same

UPGRADES / MAINTENANCE

Moving containers from hosts

Draining hosts

Rolling updates

Blue/Green deployment

Immutable infrastructure

THANKS

csanchez.org

 csanchez

 carlossg