



FROM MONOLITH TO DOCKER DISTRIBUTED APPLICATIONS

Carlos Sanchez

@csanchez

Watch online at carlossg.github.io/presentations

ABOUT ME

Senior Software Engineer @ CloudBees

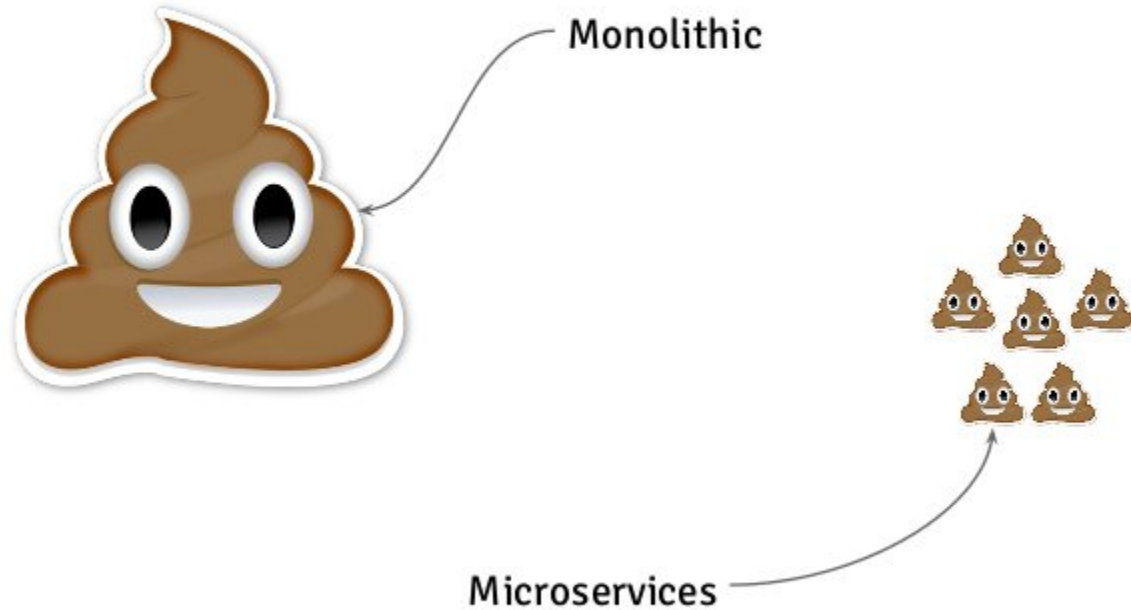
Author of Jenkins Kubernetes plugin

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

DOCKER DOCKER DOCKER



Monolithic vs Microservices



OUR USE CASE



Scaling Jenkins

Your mileage may vary

Tiger

Cluster Summary Virtual Machines Masters



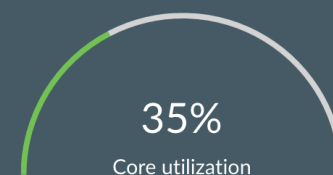
2000 healthy 0 warn 0 critical



317 healthy 0 warn 0 critical

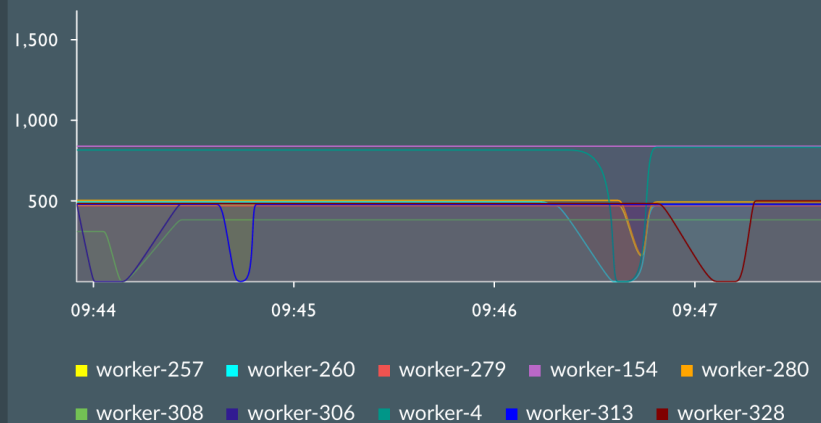


7 healthy 0 warn 0 critical

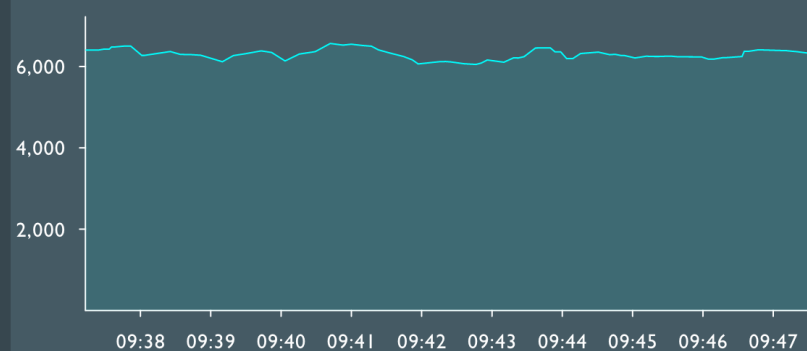


Used cores: 1300.5
Total cores: 3748

Workload



Executors



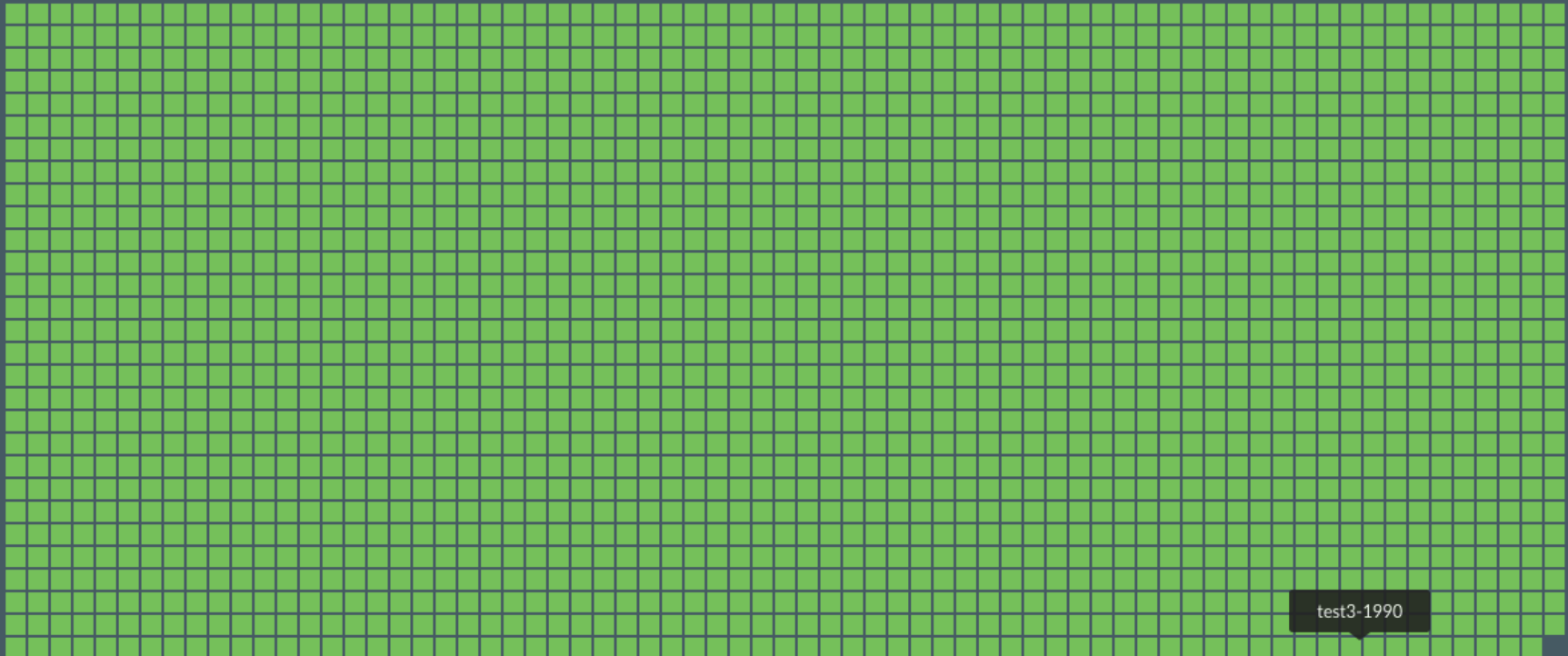
Administration

Cluster Summary

Virtual Machines

Masters

Masters



test3-1990

2000

Master 52d6f9f7-05de-4af8-88c3-30cccbf01883

Cluster: jwpse2
Server: 10.16.239.225:5050
Version: 0.28.2
Built: 3 months ago by root
Started: yesterday
Elected: yesterday

LOG

Slaves

Activated	313
Deactivated	0

Tasks

Staging	1,480
Starting	0
Running	11,095
Killing	0
Finished	0
Killed	2,145,109
Failed	41,123
Lost	294

Resources

	CPUs	Mem	Disk
Total	3732	12490.4 GB	32833.6 GB
Used	1500	9644.0 GB	0 B
Offered	1142.7	1965.9 GB	9537.5 GB
Idle	1089.3	880.4 GB	23296.1 GB

Active Tasks

ID	Name	State	Started ▼	Host	
test3-0942.3d13c1b2:18981c6c-61bd-456e-9bf5-995464be4327	test3-0942.3d13c1b2:18981c6c-61bd-456e-9bf5-995464be4327	STAGING		ec2-54-197-216-238.compute-1.amazonaws.com	Sandbox
test3-0129.2f37ba5c:20885af1-7f5e-4458-bb5d-8b5f8a3e7aaa	test3-0129.2f37ba5c:20885af1-7f5e-4458-bb5d-8b5f8a3e7aaa	STAGING		ec2-54-197-216-238.compute-1.amazonaws.com	Sandbox
test3-1835.5714308c:b80d3dbe-6d91-4181-a04b-5e3aa83fceaeb	test3-1835.5714308c:b80d3dbe-6d91-4181-a04b-5e3aa83fceaeb	STAGING		ec2-54-226-81-206.compute-1.amazonaws.com	Sandbox
test3-0702.54fa8694:b9a3cc9a-58f9-400d-b2ac-6c9dd151a963	test3-0702.54fa8694:b9a3cc9a-58f9-400d-b2ac-6c9dd151a963	STAGING		ec2-54-158-142-122.compute-1.amazonaws.com	Sandbox
test3-0131.efb771db:3dffd1a19-d39d-431a-ad3e-973a4a932398	test3-0131.efb771db:3dffd1a19-d39d-431a-ad3e-973a4a932398	STAGING		ec2-54-158-164-174.compute-1.amazonaws.com	Sandbox
test3-0845.f95b124b:c26c1e86-8cf1-4337-9d51-48b4b3e901f2	test3-0845.f95b124b:c26c1e86-8cf1-4337-9d51-48b4b3e901f2	STAGING		ec2-54-221-153-146.compute-1.amazonaws.com	Sandbox
test3-0241.23f69555:bb19e1b7-8011-409b-9629-190ed80eca92	test3-0241.23f69555:bb19e1b7-8011-409b-9629-190ed80eca92	STAGING		ec2-52-91-32-40.compute-1.amazonaws.com	Sandbox
test3-0069.3e2cd99c:6693f055-a1b6-42bb-a113-72a4c32c99ad	test3-0069.3e2cd99c:6693f055-a1b6-42bb-a113-72a4c32c99ad	STAGING		ec2-52-91-32-40.compute-1.amazonaws.com	Sandbox
test3-0437.ce767edb:0a3dea36-ecf9-497f-87f3-c84d9f43756e	test3-0437.ce767edb:0a3dea36-ecf9-497f-87f3-c84d9f43756e	STAGING		ec2-52-91-88-48.compute-1.amazonaws.com	Sandbox
test3-0045.5e5035ab:7e134c01-f459-443d-8dcd-2b755ae3bf84	test3-0045.5e5035ab:7e134c01-f459-443d-8dcd-2b755ae3bf84	STAGING		ec2-54-221-10-243.compute-1.amazonaws.com	Sandbox
test3-1919.d433af93:2d77536a-5eb4-4337-a0fd-b26b2a28bc84	test3-1919.d433af93:2d77536a-5eb4-4337-a0fd-b26b2a28bc84	STAGING		ec2-54-152-63-208.compute-1.amazonaws.com	Sandbox
test3-0107.0baadf18:4260eb52-99c1-4453-9e49-1a011a699f47	test3-0107.0baadf18:4260eb52-99c1-4453-9e49-1a011a699f47	STAGING		ec2-54-152-63-208.compute-1.amazonaws.com	Sandbox
test3-0906.d65513ff:c6f477e9-492b-4710-b1f1-c5fbbc36fa41	test3-0906.d65513ff:c6f477e9-492b-4710-b1f1-c5fbbc36fa41	STAGING		ec2-54-160-57-84.compute-1.amazonaws.com	Sandbox
test3-1495.f51d529d:b71ea06a-703f-4e12-acda-f504999f961	test3-1495.f51d529d:b71ea06a-703f-4e12-acda-f504999f961	STAGING		ec2-54-164-144-29.compute-1.amazonaws.com	Sandbox
test3-1418.8a9636b1:3923824f-d39f-4a5b-90eb-712c74e65d5c	test3-1418.8a9636b1:3923824f-d39f-4a5b-90eb-712c74e65d5c	STAGING		ec2-54-164-144-29.compute-1.amazonaws.com	Sandbox
test3-1793.700a3038:b18d3b3d-1480-4674-b4aa-ad2708a53f3c	test3-1793.700a3038:b18d3b3d-1480-4674-b4aa-ad2708a53f3c	STAGING		ec2-52-90-142-73.compute-1.amazonaws.com	Sandbox
test3-0789.868c8d8b:0421730a-e875-4ddd-938e-b17b2bbe5467	test3-0789.868c8d8b:0421730a-e875-4ddd-938e-b17b2bbe5467	STAGING		ec2-54-197-213-95.compute-1.amazonaws.com	Sandbox
test3-1616.f14a1f7d:bcc9bede-40f4-4244-acf2-380c3517515f	test3-1616.f14a1f7d:bcc9bede-40f4-4244-acf2-380c3517515f	STAGING		ec2-52-91-88-48.compute-1.amazonaws.com	Sandbox
test3-0799.acda253d:908732dc-10b2-4a40-8287-7b577a668f90	test3-0799.acda253d:908732dc-10b2-4a40-8287-7b577a668f90	STAGING		ec2-54-226-40-53.compute-1.amazonaws.com	Sandbox
test3-1486.cc2ccfaa:545e3b70-5fe7-41b7-bab1-31d964d1ed4e	test3-1486.cc2ccfaa:545e3b70-5fe7-41b7-bab1-31d964d1ed4e	STAGING		ec2-54-234-65-165.compute-1.amazonaws.com	Sandbox
test3-0230.03416eb5:459c8c8d-a1cd-4841-ae25-537da338fe96	test3-0230.03416eb5:459c8c8d-a1cd-4841-ae25-537da338fe96	STAGING		ec2-54-234-65-165.compute-1.amazonaws.com	Sandbox
test3-0324.078ea2f6:d411cb33-a481-4e7f-969c-a7a40a12818a	test3-0324.078ea2f6:d411cb33-a481-4e7f-969c-a7a40a12818a	STAGING		ec2-52-90-142-73.compute-1.amazonaws.com	Sandbox
test3-1796.88772499:d7d3da03-57ac-42df-8254-b990ed294bb8	test3-1796.88772499:d7d3da03-57ac-42df-8254-b990ed294bb8	STAGING		ec2-184-73-101-218.compute-1.amazonaws.com	Sandbox
test3-0488.475b680e:2c77aa74-8da3-4710-86f7-eeb04668f7a7	test3-0488.475b680e:2c77aa74-8da3-4710-86f7-eeb04668f7a7	STAGING		ec2-54-88-19-71.compute-1.amazonaws.com	Sandbox
test3-1201.f880d741:2f9b865f-d721-4b66-a4c5-a862fbe15d10	test3-1201.f880d741:2f9b865f-d721-4b66-a4c5-a862fbe15d10	STAGING		ec2-107-22-135-75.compute-1.amazonaws.com	Sandbox
test3-0739.22d61a92:a6c84e6e-9256-40c2-9882-fa7875b91520	test3-0739.22d61a92:a6c84e6e-9256-40c2-9882-fa7875b91520	STAGING		ec2-107-22-135-75.compute-1.amazonaws.com	Sandbox
test3-1088.58635506:20256687-36fd-4bbd-99b8-002db94601ee	test3-1088.58635506:20256687-36fd-4bbd-99b8-002db94601ee	STAGING		ec2-54-159-8-130.compute-1.amazonaws.com	Sandbox

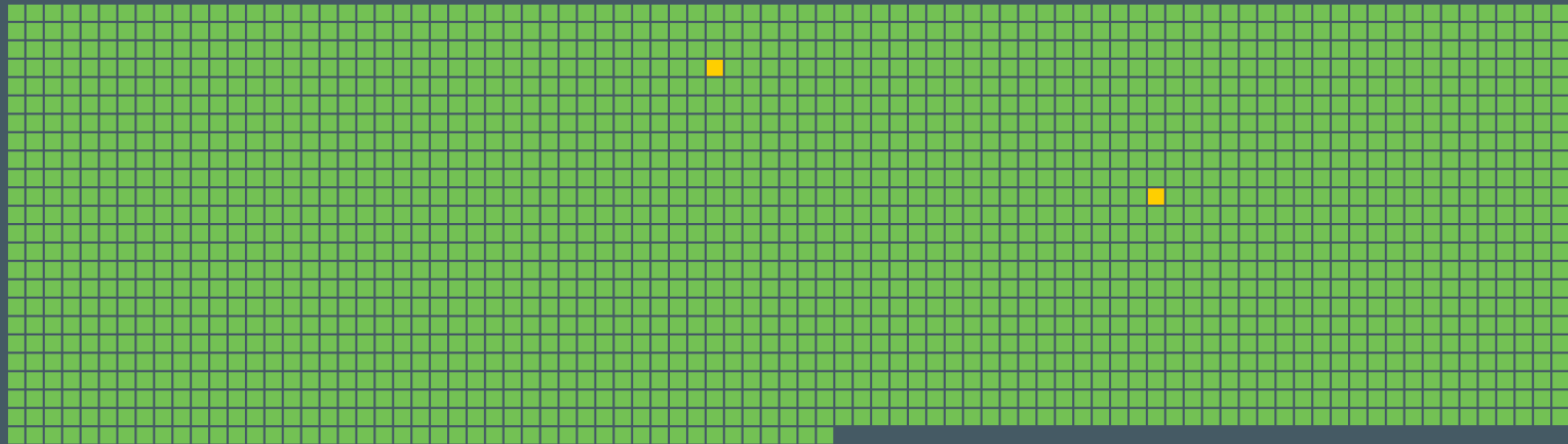
Tiger

Cluster Summary

Virtual Machines

Masters

Masters



2000

A 2000 JENKINS MASTERS CLUSTER

- 3 Mesos masters (m3.xlarge: 4 vCPU, 15GB, 2x40 SSD)
- 317 Mesos slaves (c3.2xlarge, m3.xlarge, m4.4xlarge)
- 7 Mesos slaves dedicated to ElasticSearch: (c3.8xlarge: 32 vCPU, 60GB)

12.5 TB - 3748 CPU

Running 2000 masters and ~8000 concurrent jobs

ARCHITECTURE



Kernel Sanders

@lstoll

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

Isolated Jenkins masters
Isolated build agents 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.

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



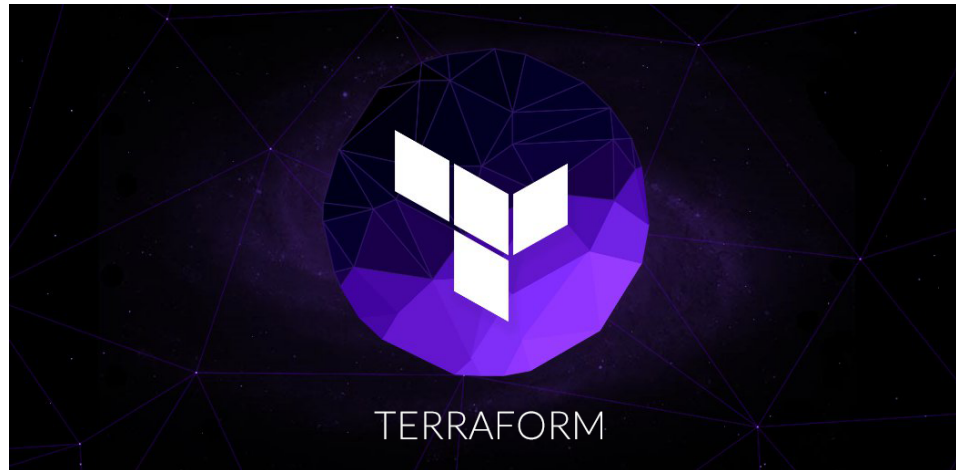
Docker Swarm / Kubernetes

MESOSPHERE MARATHON



MARATHON

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

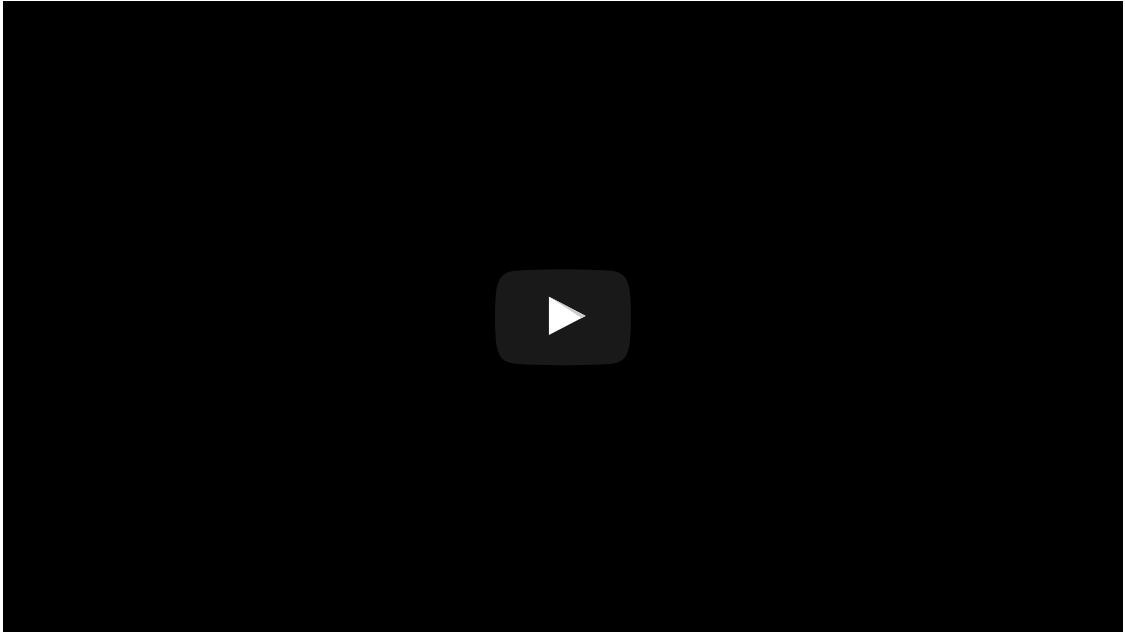



@DEVOPS_BORAT

DevOps Borat

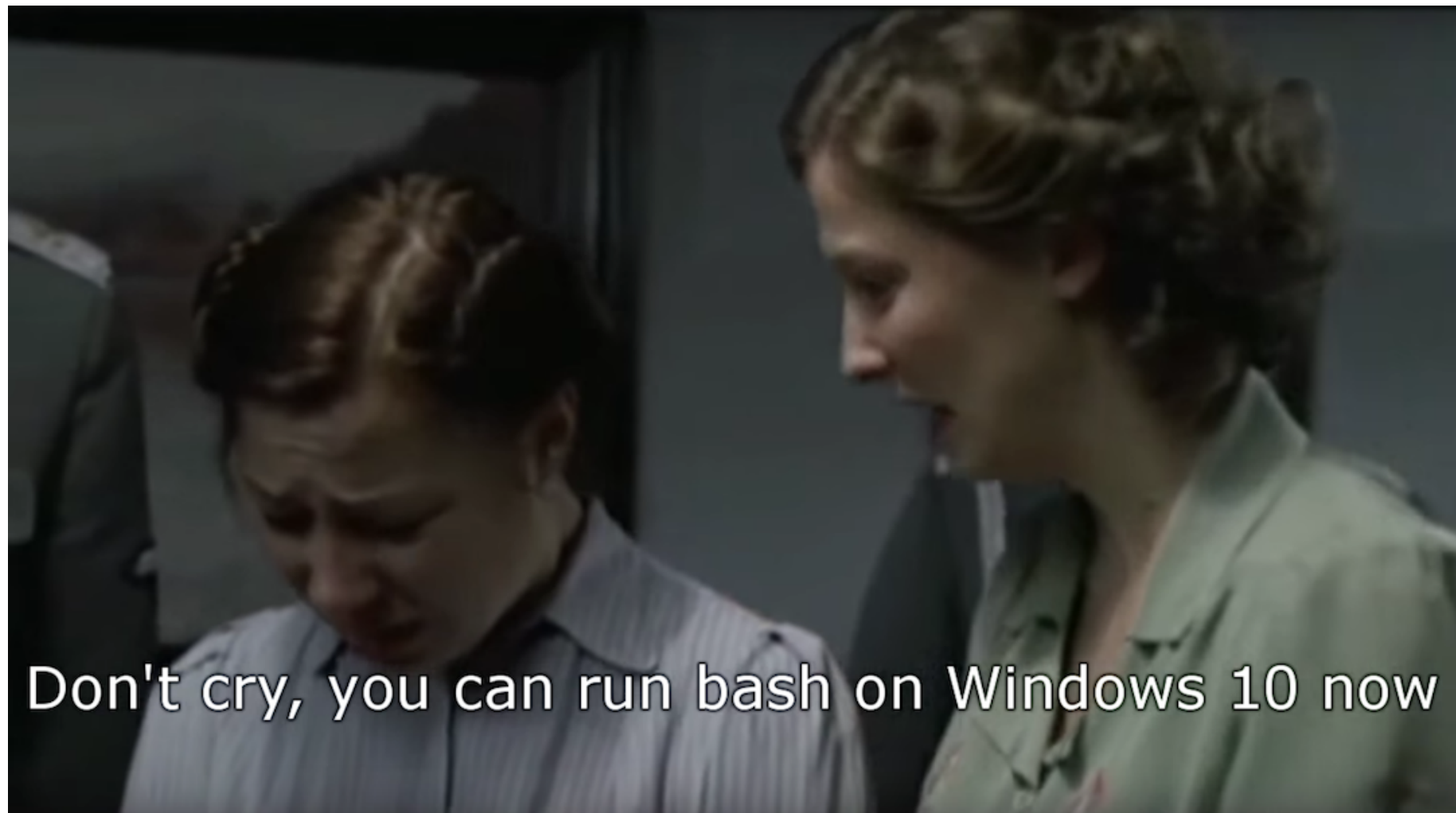
To make error is human. To propagate error to all server in automatic way is **#devops**.

**IF YOU HAVEN'T AUTOMATICALLY
DESTROYED SOMETHING BY
MISTAKE,
YOU ARE NOT AUTOMATING ENOUGH**



A close-up, low-angle shot of a man in a dark suit and white shirt, looking down. The lighting is dramatic, with strong highlights on his face and suit. The background is blurred, showing other people in a dimly lit room.

If you never used Docker in
production, leave the room now



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

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

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

Supporting EBS (AWS) and external NFS

PERMISSIONS

Containers should not run as root

Container user id \neq host user id

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

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?



ICACHONDEO.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`

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

NETWORKING: SOFTWARE DEFINED NETWORKS

Create new custom networks on top of physical networks

Allow grouping containers in subnets

NETWORKING: SOFTWARE DEFINED NETWORKS

Battlefield: Calico, Flannel, Weave and Docker Overlay Network

<http://chunqi.li/2015/11/15/Battlefield-Calico-Flannel-Weave-and-Docker-Overlay-Network/>

SCALING

New and interesting problems

LOGGING

Running ElasticSearch as a cluster service, and the ELK stack

Docker configured to log to syslog

Logstash redirecting syslog to ElasticSearch

Embedded Kibana dashboard in CloudBees Jenkins
Operations Center

AWS

Resource limits: VPCs, S3 snapshots, some instance sizes

Rate limits: affect the whole account

Retrying is your friend, but with exponential backoff

EMBRACE FAILURE!



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



cloudbees®