Docker

an introduction

Markus van Dijk SURF SARA

Process isolation

Why

- Testing: reproducible environment
- Multiple instances
- Library versions
- Vulnerable SW

Process isolation

History

- Unix `chroot` (1982)
- BSD `jail` (2000)
- Linux containers LXC (2008)
- Docker (2013)
- Open Container Initiative (2015)

But we have VMs

Containers	**VMs**
Shared OS	Independent OSes
FS copy-on-write	Duplicated FS
Lightweight start/stop	Full boot/shutdown
microservice	*LAMP* etc
One container, one job	Many jobs
External plumbing	Configure firewalls etc
Prepare to fail	Robust

Typical use

- Develop and deploy in same environment
  Grid jobs: test on laptop!

- Task-driven service activation
  Front-end starts container for each job

- Easy migration iff prepare to fail
  Kill containers, restart jobs in new location

- Connected containers, need not run on same host
  DB servers, web front, job processors

Isolation

- File system
- Processes
- Network
- User IDs (coming soon)

- Some vulnerabilities
  Run in VM for complete protection for now

Docker toolbox

- **`docker`**
  Defines a single container and manages its lifecycle
  View logs, execute a process in running container

- **`docker-machine`**
  Transparently runs docker commands on remote host

- **`docker-compose`**
  Defines a set of connected containers, manages dependencies in lifecycle

- Various tools for orchestration
  fleet, swarm, mesos, …

Example: Cloud9

https://c9.io

- Cloud-based development workspace
- 1 docker workspace per project
- Ssh root access to your workspace
- Web browser IDE
- Container down after idle period
- Container up in seconds
- Project footprint: only your files

12
1
2
3
4
5
6
7
8
9
10
11