Docker

an introduction

Markus van Dijk SURF SARA

Process isolation

Why

- Testing, multiple instances
- Vulnerable SW

History

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

But we have VMs

Containers

- Shared OS
- FS copy-on-write
- Lightweight start/stop
- *microservice*
- One container, one job
- Externally connected
- Prepare to fail

VMs

- Independent OSes
- Duplicated FS
- Full boot/shutdown
- *LAMP* etc
- Many jobs
- Configure firewalls etc
- 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
- User IDs
- Processes
- Network
- Some vulnerabilities
  Run in VM for complete protection for now

Docker toolbox

- **`docker`**
  Defines single container and manages its lifecycle
  View logs, execute 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, …

P3P3
Enabling Dynamic Services

- Dynamic: ✓
- Service: ✓

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