The Future of In-Container Configuration Management

Diego Zamboni, Mark Burgess
CFEngine AS
diego.zamboni@cfengine.com
zzamboni
A grossly oversimplified history of computing

Pets vs cattle
Pets vs cattle

Service Model

- Pets are given names like pussinboots.cern.ch
- They are unique, lovingly hand raised and cared for
- When they get ill, you nurse them back to health

- Cattle are given numbers like vm0042.cern.ch
- They are almost identical to other cattle
- When they get ill, you get another one

- Future application architectures should use Cattle but Pets with strong configuration management are viable and still needed

Gavin McCance, CERN
Pets vs cattle

- Custom-built server (physical or VM)
  - Carefully configured (by hand or CM)
  - Hard to replicate

- Automatically-built server (VM or container)
  - Statically-configured (by hand or CM)
  - Easy to replicate
Pets vs cattle

- Custom-built server (physical or VM)
- Carefully configured (by hand or CM)
- Hard to replicate

- Automatically-built server (VM or container)
- Statically-configured (by hand or CM)
- Easy to replicate

Traditional domain of configuration management
Configuration management in the age of containers?
Configuration management in the age of containers?
Configuration management in the age of containers?
Configuration management in the age of containers?
Configuration management in the age of containers!
Configuration management in the age of containers!

Application deployment

Container management

Image management

Front-end

App Logic

DB

Application deployment
The pet-cattle scale
The pet-cattle scale
Balancing the scale

```json
{
    "cfengine/cfubuntu": {
        "base_image": "ubuntu",
        "install_cfengine": true,
        "packages": [ "openssh-server" ],
        "ports": [ 22 ]
    },
    "cfengine/cfdbservr": {
        "base_image": "cfengine/cfubuntu",
        "packages": [ "mysql-server", "mysql-client" ],
        "ports": [ 3306 ]
    },
    "cfengine/cfwebserver": {
        "base_image": "cfengine/cfubuntu",
        "packages": [ "apache2", "libapache2-mod-php5",
                      "php5-mysql" ],
        "ports": [ 80, 443 ]
    },
    "cfengine/cfwordpress": {
        "base_image": "cfengine/cfwebserver",
        "extralines": [
            "RUN rm -rf /var/www/html",
            "ADD http://wordpress.org/latest.tar.gz /wordpress.tar.gz",
            "RUN tar xvzf /wordpress.tar.gz",
            "RUN mv /wordpress /var/www/html",
            "RUN chown -R www-data:www-data /var/www/"
        ],
    },
    "cfengine/cfhaserver": {
        "base_image": "cfengine/cfubuntu",
        "packages": [ "haproxy" ],
        "ports": [ 80, 443 ]
    }
}
```
Balancing the scale

```json
{
"cfengine/cfubuntu": {
  "base_image": "ubuntu",
  "install_cfengine": true,
  "packages": [ "openssh-server" ],
  "ports": [ 22 ]
},
"cfengine/cfdbserver": {
  "base_image": "cfengine/cfubuntu",
  "packages": [ "mysql-server", "mysql-client" ],
  "ports": [ 3306 ]
},
"cfengine/cfwebserver": {
  "base_image": "cfengine/cfubuntu",
  "packages": [ "apache2", "libapache2-mod-php5", "php5-mysql" ],
  "ports": [ 80, 443 ]
},
"cfengine/cfwordpress": {
  "base_image": "cfengine/cfwebserver",
  ],
},
"cfengine/cfhaserver": {
  "base_image": "cfengine/cfubuntu",
  "packages": [ "haproxy" ],
  "ports": [ 80, 443 ]
}
}
```
Balancing the scale

```json
{
    "cfengine/cfubuntu": {
        "base_image": "ubuntu",
        "install_cfengine": true,
        "packages": [ "openssh-server" ],
        "ports": [ 22 ]
    },
    "cfengine/cfdbuserver": {
        "base_image": "cfengine/cfubuntu",
        "packages": [ "mysql-server", "mysql-client" ],
        "ports": [ 3306 ]
    },
    "cfengine/cfwebserver": {
        "base_image": "cfengine/cfubuntu",
        "packages": [ "apache2", "libapache2-mod-php5",
                      "php5-mysql" ],
        "ports": [ 80, 443 ]
    },
    "cfengine/cfwordpress": {
        "base_image": "cfengine/cfwebserver",
        "extralines": [ "RUN rm -rf /var/www/html",
                        "ADD http://wordpress.org/latest.tar.gz /var/www",
                        "RUN tar xzvf /wordpress.tar.gz",
                        "RUN mv /wordpress /var/www/html",
                        "RUN chown -R www-data:www-data /var/www/"
                    ],
    },
    "cfengine/cfhaserver": {
        "base_image": "cfengine/cfubuntu",
        "packages": [ "haproxy" ],
        "ports": [ 80, 443 ]
    }
}
```
Balancing the scale

```
bundle agent service_catalogue
{
  methods:

  any::
    "docker" usebundle => docker_discovery,
    comment => "Discover if we are in a docker container";

  "ssh" usebundle => setup_ssh,
    comment => "Start ssh and ensure correct keys are in";

  am_policy_hub::
    "setup_docker" usebundle => docker_infrastructure,
    comment => "Set up docker containers";

  web_server::
    "wordpress" usebundle => config_wordpress,
    comment => "Configure wordpress on the local host";

  db_server::
    "mysql" usebundle => config_mysql,
    comment => "Configure MySQL for wordpress";

  load_balancer::
    "haproxy" usebundle => config_haproxy,
    comment => "Configure haproxy for load balancer";
}
```
<table>
<thead>
<tr>
<th>Feature</th>
<th>In-container</th>
<th>Off-container</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image size</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>Runtime size</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>Image/Dockerfile proliferation</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>Startup time</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>Runtime changes</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>Immutable infrastructure</td>
<td>Green</td>
<td>Red</td>
</tr>
<tr>
<td>Repeatability across infrastructures</td>
<td>Green</td>
<td>Red</td>
</tr>
</tbody>
</table>
Orchestration

Parent

- child 1
- child 2
- child 3
Orchestration

```hcl
bundle agent main
{
  methods:
    parent:::
      "plant continuous delivery seeds" comment => "The files where people make changes";
      "pick the flowers" comment => "Reap the harvest";
      "package for delivery" comment => "Tie it with a bow";
    bud_containers:::
      "feed from the parent" comment => "pick up files and build something";
  services:
    parent:::
      "budding children" comment => "Spawn some fledgings";
}
```
Orchestration

```javascript
bundle agent service_budding_children(name, state)
{
    guest_environments:

        parent::

            "plant continuous delivery seeds"
            "pick the flowers"
            "package for delivery"

        bud_containers::

            "feed from the parent" comment =>

    services:
     parent::

        "budding children" comment => "Sp

bundle agent main
{
   methods:

           parent::

           bud_containers::

    services:
     parent::

        "budding children" comment => "Sp

body guest_details stem_cell
{
    guest_type => "docker";
    guest_image_name => "cf-stem-cell";
}```
What do we need from in-container CM?

- Lightweight
- Distributed
- Resilient
Thank you

Diego Zamboni
diego.zamboni@cfengine.com