Deploying UNICORE
With One Click

7. September 2015  |  Sander Apweiler and Benedikt von St. Vieth
Motivation

Use software to define your infrastructure and services
Motivation

Use software to define your infrastructure and services

Reasons

- reproducability
- easier deployments
- automated updates
- faster failovers
Motivation

Use software to define your infrastructure and services

Reasons

- reproducability
- easier deployments
- automated updates
- faster failovers

Technologies

- Ansible, Chef, puppet, ...
- docker
puppet
about puppet

- configuration management tool
- declarative definition of resources
  - packages
  - services
  - configuration files
- operating system agnostic
- public-private-key infrastructure for communication
about **puppet** – **client-server**

- node asks puppetmaster about his resources
- puppetmaster sends catalog to node
- node enforces the configuration, does necessary corrections and reports to puppetmaster
about puppet – declarative definition

```ruby
node 'myhost.example.com' {
  class { 'mysql::server':
    root_password => 'password'
  }
  mysql::db { 'mydb':
    user => 'myuser',
    password => 'mypass',
    host => 'localhost',
    grant => ['SELECT', 'UPDATE'],
  }
}
```
deploying \texttt{UNICORE}

- install required Java
- create necessary directories
- load .tar.gz-file to server
- extract .tar.gz-file
- modify configure.properties
- run configuration script
- run install script
- start server components
deploying **UNICORE – with puppet**

Steps required when using puppet:
- add **UNICORE** class to server definition

```
node myhost.example.com {
  include ::unicore
}
```

Minimal set of variables:
- `unicore_hostname`: `mydns.example.com ($::fqdn)`
- `unicore_servers_version`: 7.4.0
puppet resources used

- user
- package
  - install Java for current operating system
- file
  - copy archive to server
  - create configuration from template
- exec
  - extract archive
  - run configure step
  - run installation
  - start service
puppet

**pro puppet:**

- controls operating system and services
- uses same code for \( N \) deployments

**con puppet:**

- you need a server or VM
- operating system already is a overhead
- no atomic updates
docker
about Docker

Lightweight virtualization using technologies like cgroups, namespaces, libcontainer and AuFS

build

... a image containing your application

ship

... your image using a registry or the central docker Hub

run

... the application by starting a container from the image
build – Dockerfile

FROM centos:latest
MAINTAINER Benedikt von St. Vieth

# install dependencies and clean-up afterwards
RUN yum -y update && yum install -y wget java-1.8.0-openjdk-headless && yum clean all && rm -rf /var/lib/{rpm,yum}

RUN wget -q -O '/unicore.tgz' '<...>/...tgz' && mkdir '/opt/unicore' && 
tar -xzf /unicore.tgz -C /opt/unicore && 
rm -rf /unicore.tgz /opt/unicore/docs && 
cd /opt/unicore && 
  sed -i ... configure.properties

ADD entrypoint.sh /entrypoint.sh
ENTRYPOINT /entrypoint.sh
EXPOSE 8080
deploying UNICORE – docker run

docker run -d -p 8082:8082
   -e WSRFLITEURL=hostname
   -e GATEWAYPORT=8082 benedicere/unicore
docker

**pro docker:**
- run UNICORE deployment with one command
- use same images for $N$ deployments
- start on every Linux, virtualized on Windows/OSX

**con docker:**
- increase complexity when using volumes and links
conclusion and outlook
conclusion

**puppet**
- well suited for reproducible operations

**docker**
- more complex integration testing
- good alternative for Live-CD
- ease operations
## Conclusion

<table>
<thead>
<tr>
<th>Puppet</th>
<th>Docker</th>
</tr>
</thead>
<tbody>
<tr>
<td>well suited for reproducible operations</td>
<td>more complex integration testing</td>
</tr>
<tr>
<td></td>
<td>good alternative for Live-CD</td>
</tr>
<tr>
<td></td>
<td>ease operations</td>
</tr>
<tr>
<td></td>
<td>x509 authentication makes it very complex</td>
</tr>
<tr>
<td></td>
<td>too many and to complex configuration files</td>
</tr>
</tbody>
</table>
outlook

**puppet**
- introduce *puppet* class per service
- implement update procedure

**docker**
- build images per service
- use `docker-compose`
- move persistence to external volumes
- use UNICORE packages