Data Atoms
Get your own UNICORE storage
# Use cases

## Peer-to-peer data exchange

- results/research on the edge of the network
- direct exchange e.g. for the sake of confidentiality
### Use cases

#### Peer-to-peer data exchange

- results/research on the edge of the network
- direct exchange e.g. for the sake of confidentiality

#### Moving data between infrastructures

- supercomputing facility to/from computation cloud
- often more efficient than through 3rd party
Requirements

1. quick deployment
2. short lived (one transfer?)
3. quick undeployment
4. single user
5. compatible with established tools
Requirements

1. quick deployment
2. short live times (one transfer?)
3. quick undeployment
4. single user
5. compatible with established tools

- one click deployment
- cert pining
- minimal UNICORE stack
Solution: Docker

1. light-weight virtualization solution
2. created for shipping applications from one machine to another
3. well-established technologies: namespaces, cgroups, union fs, lxc/libcontainer
4. separation between data and images
5. good ways for exchanging images
Docker Terminology

Images
- read only templates e.g. ubuntu+apache
- images are used to create docker containers
- images can be easily shared

Registers
- image repository
- private or public stores where you can upload or download images e.g. DockerHub

Containers
- similar to a directory
- everything that is needed for an application to run
- containers are created from images
- each container is “a secure & isolated application platform”
Creating images

Interactive

Run a basis image, install services, save the work. Push to repository (DockerHub).
Creating images

<table>
<thead>
<tr>
<th>Interactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run a basis image, install services, save the work. Push to repository (DockerHub).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Imperative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describe the installation and configuration steps. Build image and push.</td>
</tr>
</tbody>
</table>
FROM tomcat:7.0
MAINTAINER Jedrzej Rybicki
ADD supervisord.conf /etc/supervisord.conf
RUN DEBIAN_FRONTEND=noninteractive apt-get update && apt-get install supervisor -y && \
  apt-get clean autoclean && apt-get autoremove && \
  rm -rf /var/lib/{apt,dpkg,cache,log}
EXPOSE 8080
RUN wget -q -O '/tmp/voyant.zip' 'http://dev.voyant-tools.org/downloads/current/VoyantServer.zip' && \
  mkdir /tmp/voyant/ && cd /tmp/voyant/ && unzip -qq /tmp/voyant.zip && \
  rm -rf /usr/local/tomcat/webapps/ROOT/ && cp -r _app/ /usr/local/tomcat/webapps/ROOT/ && \
  cd /usr/local/tomcat/ && rm -rf /tmp/voyant.zip /tmp/voyant/
VOLUME /usr/local/tomcat/temp
CMD supervisord -c /etc/supervisord.conf
Creating images

<table>
<thead>
<tr>
<th>Interactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run a basis image, install services, save the work. Push to repository (DockerHub).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Imperative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describe the installation and configuration steps. Build image and push.</td>
</tr>
</tbody>
</table>
Creating images

Interactive

Run a basis image, install services, save the work. Push to repository (DockerHub).

Imperative

Describe the installation and configuration steps. Build image and push.

Automatic

Based on imperative description, building process “in the cloud” (from GitHub directly to DockerHub).
Building stand-alone Docker-based storage-alone UNICORE images
Building stand-alone Docker-based storage-alone UNICORE images

Five stages of grief:

1. denial
2. anger
3. bargaining
4. depression
5. acceptance
Building stand-alone Docker-based storage-alone UNICORE images

Five stages of grief learning:

1. denial
2. anger
3. bargaining
4. depression
5. acceptance
Software installation

Starting interactive docker session:

```
$ docker run --it --h unicore --p 7777:7777
   --v /home/jj/certs:/certs/ --v /some/path:/storage/
    ubuntu:latest /bin/bash
```

Deb packages are cool:

```
$ apt-get install --y curl
$ curl http://unicoresoft.nebula.grid.icm.edu.pl/gpg.public | apt-key add --
$ echo "deb http://unicoresoft.nebula.grid.icm.edu.pl/repository/
  dist/UNICORE/rc/debian wheezy main" >> /etc/apt/sources.list
$ apt-get install --y unicore-unicorex
```
Software installation: anger & depression

keytool ...
Software configuration:

/etc/unicore/unicorex/wsrf.xml

Certificates:

<property name="container.baseurl" value="https://unicore:7777/services" />
<property name="container.host" value="unicore" />
<property name="container.security.credential.path" value="/certs/server−keystore.jks" />
<property name="container.security.credential.password" value="password" />
<property name="container.security.credential.format" value="jks" />
Software configuration

/etc/unicore/unicorex/wsrf.xml

Services:
- StorageManagement
- StorageFactory
- Enumeration
- FileTransfer
- FileTransferBFT
- FileTransferRBYTEIO
- FileTransferSBYTEIO
- ServiceGroupEntry
- Registry

REST:
- de.fzj.unicore.uas.rest.CoreServices
Software configuration

/etc/unicore/unicorex/uas.config

# container.onstartup.1=de.fzj.unicore.cisprovider.impl.InitOnStartup
# container.onstartup.2=de.fzj.unicore.bes.util.BESOnStartup
coreServices.sms.factory.DEFAULT.path=/store/
container.security.gateway.waitForStartup=false
container.security.gateway.registration=false
container.security.gateway.checkSignature=false
container.externalregistry.use=false

$ docker run -it -h unicore -p 7777:7777
   -v /home/jj/certs/:/certs/ -v /some/path/:/storage/
   ubuntu:latest /bin/bash
Software configuration

Access rights: /etc/unicore/unicorex/simpleuudb

<fileAttributeSource>
<entry key="CN=Joe Doe">
  <attribute name="role">
    <value>user</value>
  </attribute>
  <attribute name="xlogin">
    <value>joe</value>
  </attribute>
  <attribute name="group">
    <value>users</value>
  </attribute>
</entry>
</fileAttributeSource>
Publishing and reusing

$ docker commit [container] unicore/dataatom
$ docker push
$ docker run -h unicore -p 7777:7777 -v /home/jj/certs:/certs/ -v /some/path:/storage/ -d unicore/dataatom
Summary

Data Atoms

Stand-alone easy-deployable storage elements

- experimental Docker container available
- works pretty well
Summary

Data Atoms

Stand-alone easy-deployable storage elements

- experimental Docker container available
- works pretty well
- ... but there are some problems
- after-effect ucc container

Further work:

- UFTP? ⇒ authorization (stand-alone)
- other UNICORE elements?
- containers as jobs/workflows?
Thanks

j.rybicki@fz-juelich.de