pycsw Workshop

version 1.4

Jeff McKenna, Tom Kralidis, Angelos Tzotsos

June 12, 2013

Contents

pycsw Workshop

Welcome to the pycsw workshop. This workshop is a hands-on workshop that will give you an introduction to the popular pycsw metadata publishing software.

Note

The workshop instructions are also available as a single PDF document 

Background

Home

All workshop materials are maintained openly through a GitHub repository: https://github.com/geopython/pycsw-workshop. Contributions are always welcome.

The canonical location of the live workshop is always at http://geopython.github.io/pycsw-workshop/.

History

Initial workshop structure was created by Jeff McKenna of Gateway Geomatics.

Acknowledgements

The initial pycsw workshop materials were created through funding provided by the Oregon Coastal Management Program, through an FGDC CAP grant, in 2013.

Assumptions

As this workshop is designed for use with the OSGeo-Live virtual machine, basic knowledge of Unix commands is assumed.

License

Workshop Requirements

The pycsw workshop requires the following installed locally:

- 7-Zip (ability to extract *.7z* files)

- VirtualBox (virtual machine software, ability to load virtual disk *.vmdk files)
- OSGeo-Live Virtual Machine (which contains pycsw)

> ***Note***
>
> We recommend using the OSGeo-Live Virtual Machine method, although OSGeo-Live is available also through a bootable DVD or USB drive.

1. Install VirtualBox

- download the VirtualBox platform package for your local machine
- run the installer, and select the default setup options (approve any device security questions)

2. Download OSGeo-Live

> ***Caution!***
>
> You'll need a minimum of 10GB of free hard disk space, as well as a machine with 2GB of RAM.

Download

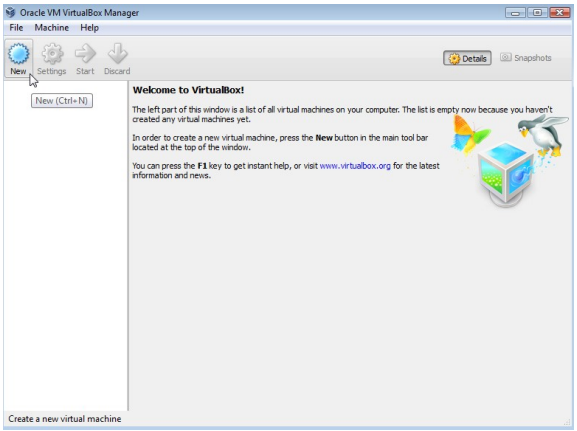
- download the OSGeo-Live Virtual Machine (*.7z) file. It will likely take you ~1 hour to download the 3GB file. There are several sites you can download this from:
  - official site
  - UC Davis mirror
  - National Technical University of Athens mirror

Extract

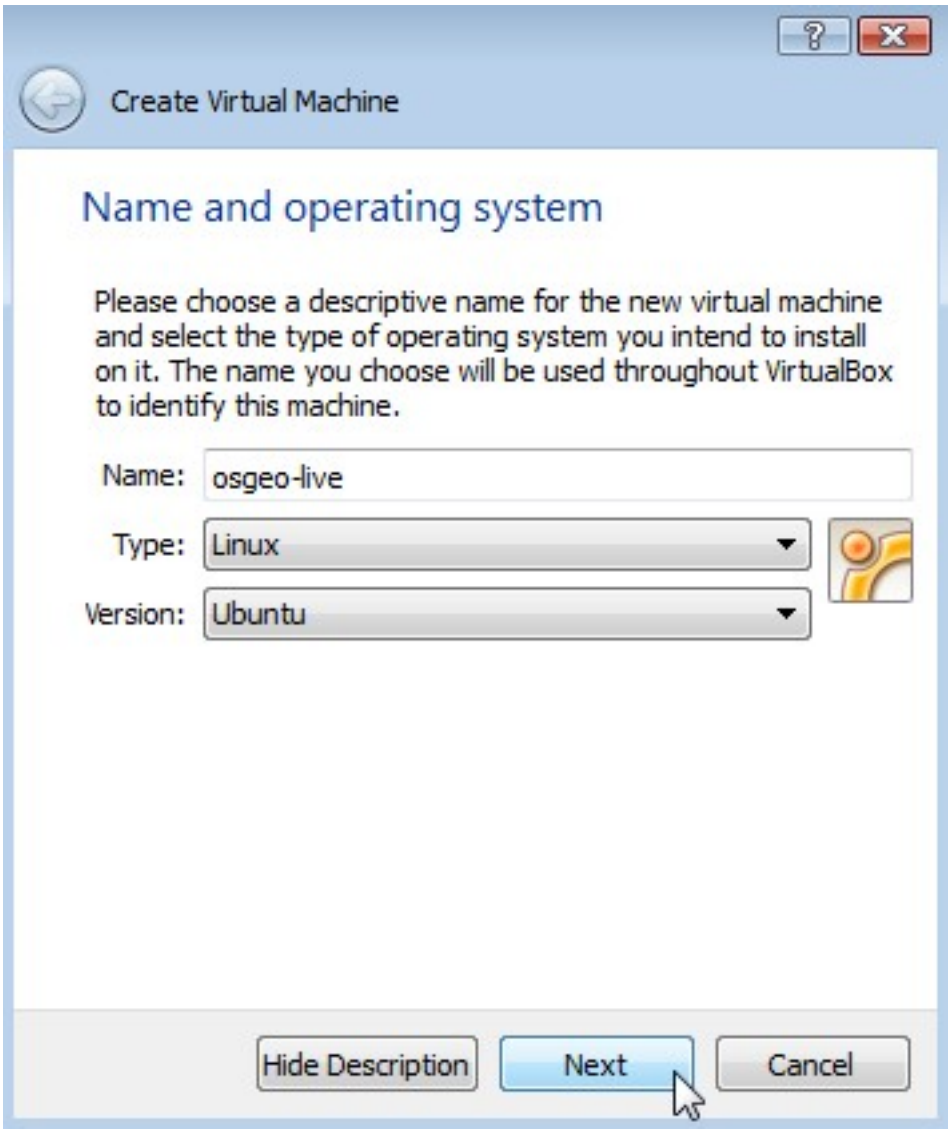
- using 7-Zip, open the *.7z* archive and extract the *.vmdk* file onto your hard disk (the extracted file is ~10GB in size)

3. Create Virtual Machine

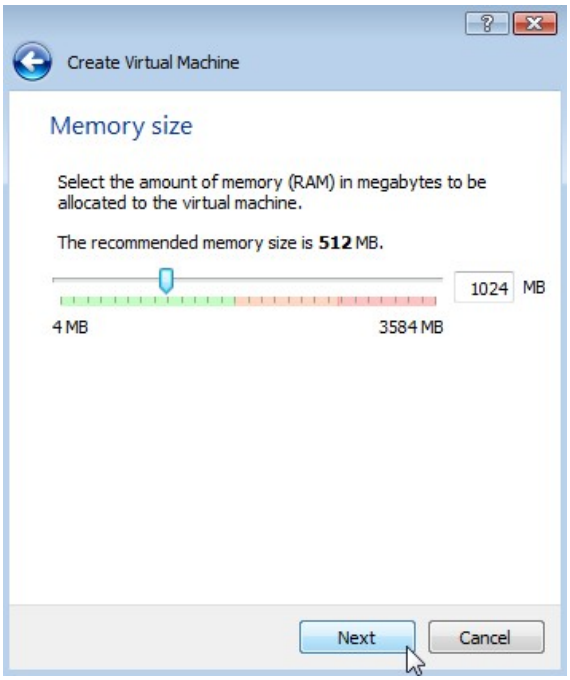
- start VirtualBox ("Oracle VM VirtualBox)
- click on the *New* button to create a VM



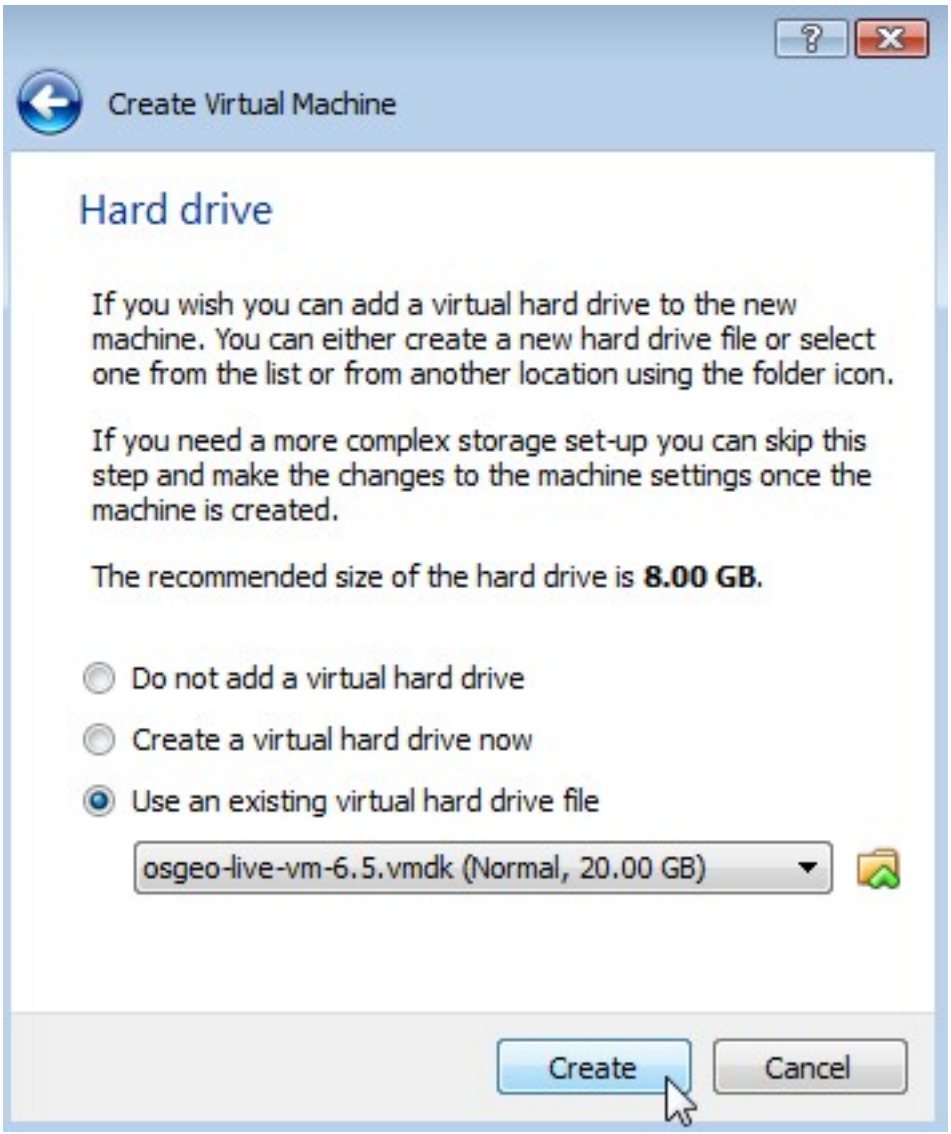
- enter "osgeo-live" for the name, and select *Type: Linux* and *Version: Ubuntu*



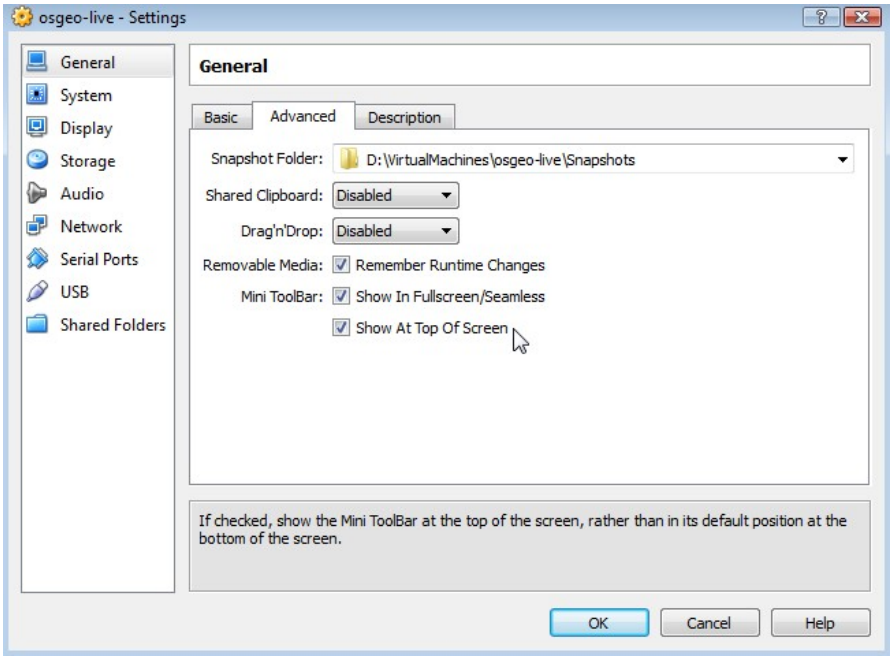
- In the next screen set the memory to 1024 MB (or more if your host computer has more than 4GB).



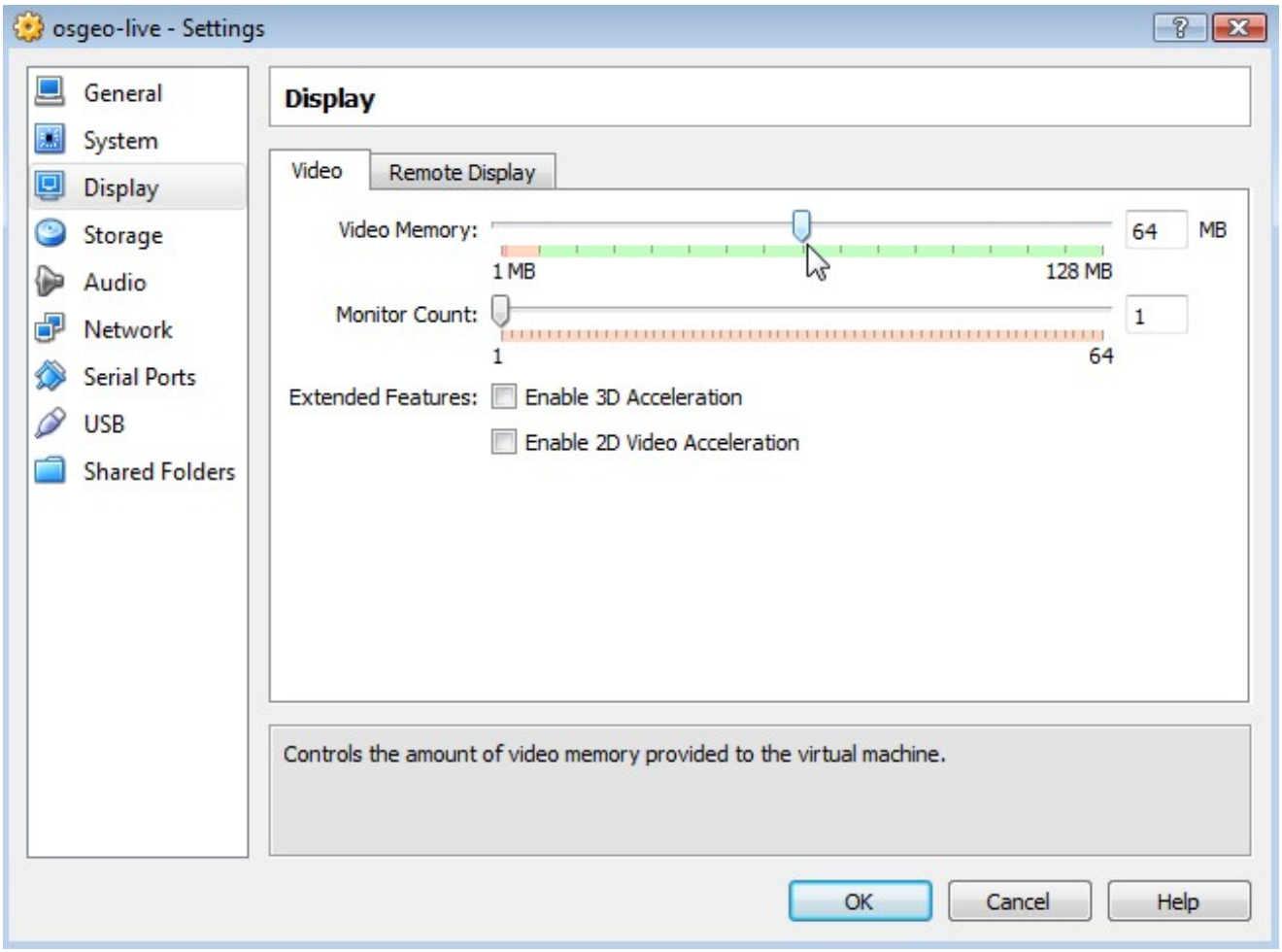
- Continue to the next screen and choose "Use existing hard disk". Then click on the button (a folder icon) to browse to where you saved the *.vmdk file. Select this file, press *Next* and *Create*.



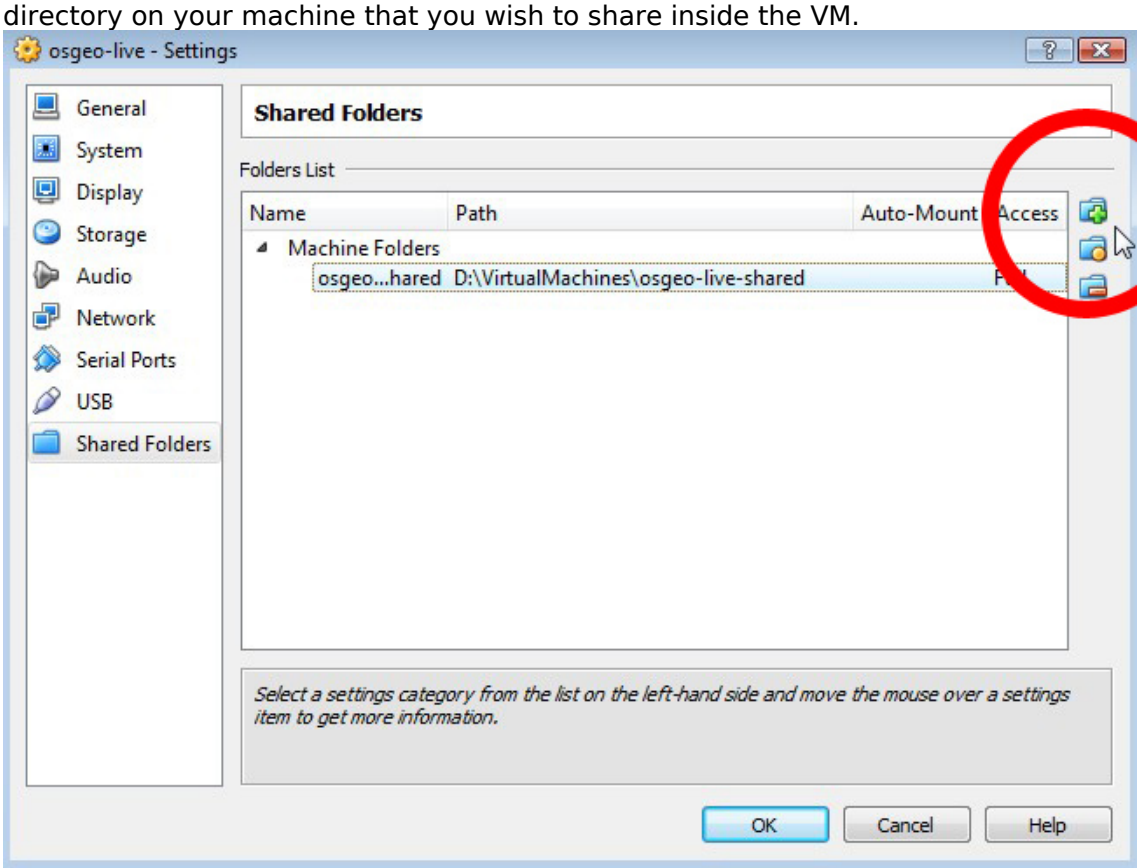
- Once the VM is created, click on the Settings button. In the "General" section, go to the Advanced tab, and click to select "Show at top of screen" for the Mini toolbar.



- In the "Display" section and increase video memory to 32 or 64 MB.



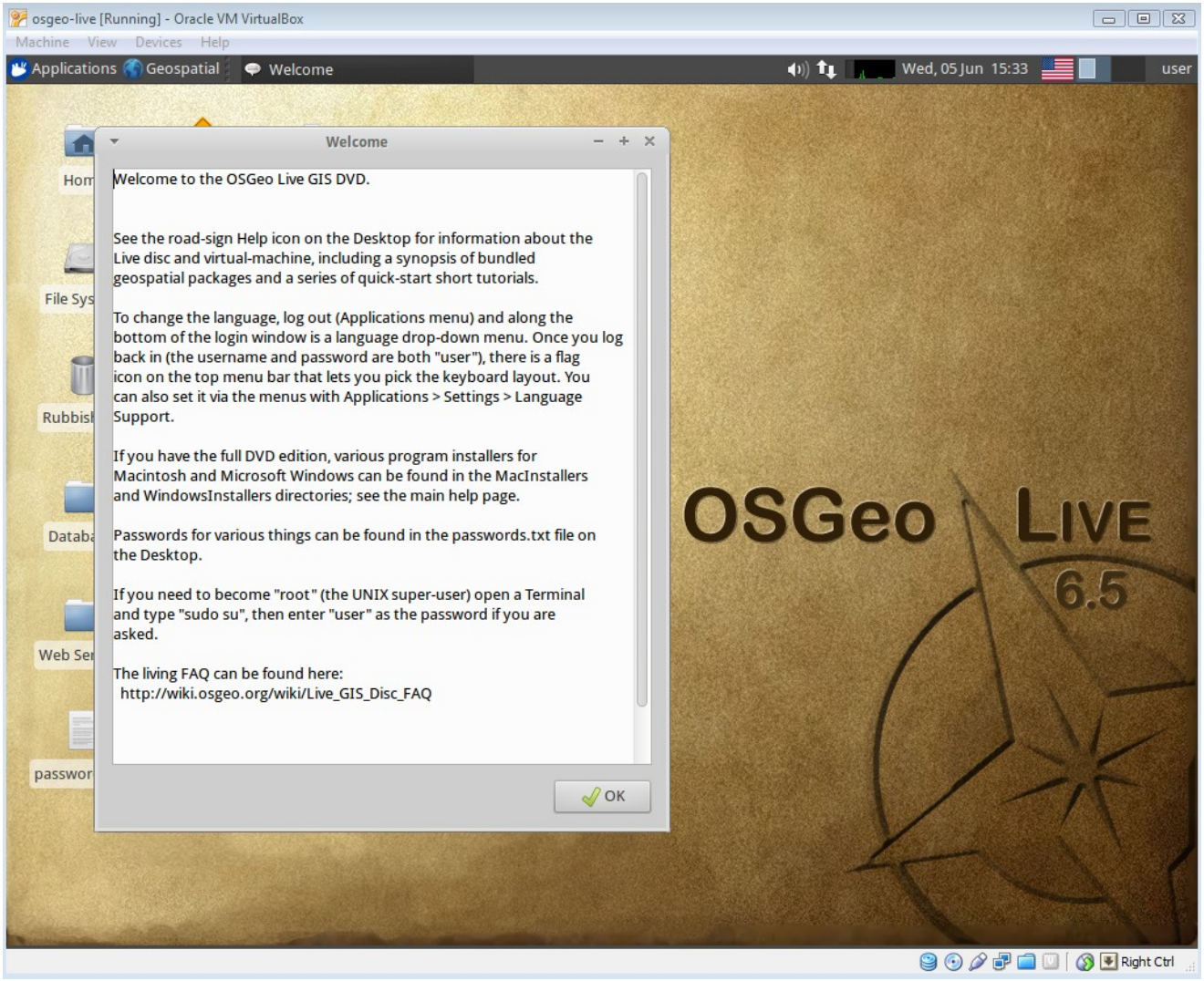
- In the "Shared Folders" section, click the "Add folder" (green + icon on the right) to find a directory on your machine that you wish to share inside the VM.



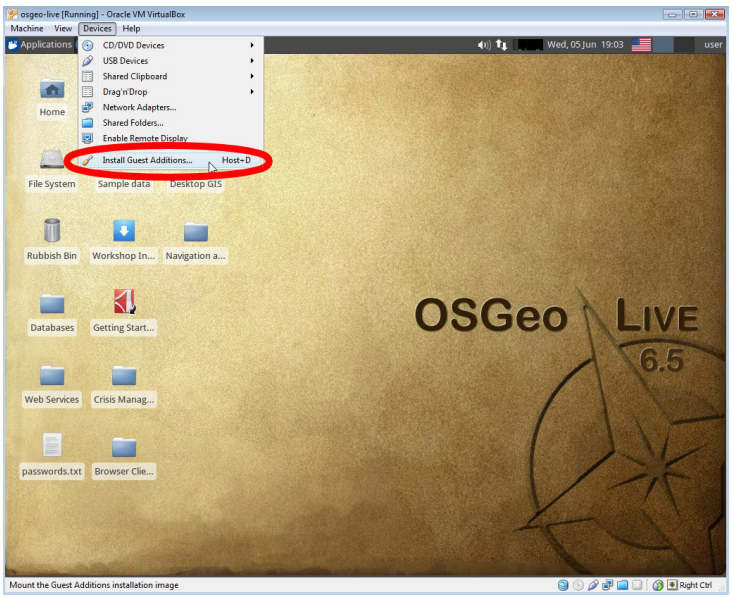
Once the "Folder path" and "Folder name" are defined, click OK, and close the settings window.

4. Run the Virtual Machine

- Now bootup the VM by clicking the Start (green arrow) button. OK any warning messages.



- To improve video performance and enable the shared folders, open the Devices menu and click "Install Guest Additions".



- Next, on the desktop you will see an icon named "VBOXADDITIONS_4.2.12_84980", click it (this mounts the drive). You can then close this window.
- Open a Terminal window (in top left click "Applications" / "Accessories" / "Terminal Emulator"
- In the Terminal, execute the following:

```
user@osgeolive:~$ sudo apt-get install linux-headers-`uname -r`

   password: user

user@osgeolive:~$ cd /media/VBOXADDITIONS_4.2.12_84980

user@osgeolive:/media/VBOXADDITIONS_4.2.12_84980$ sudo ./VBoxLinuxAdditions.run
```

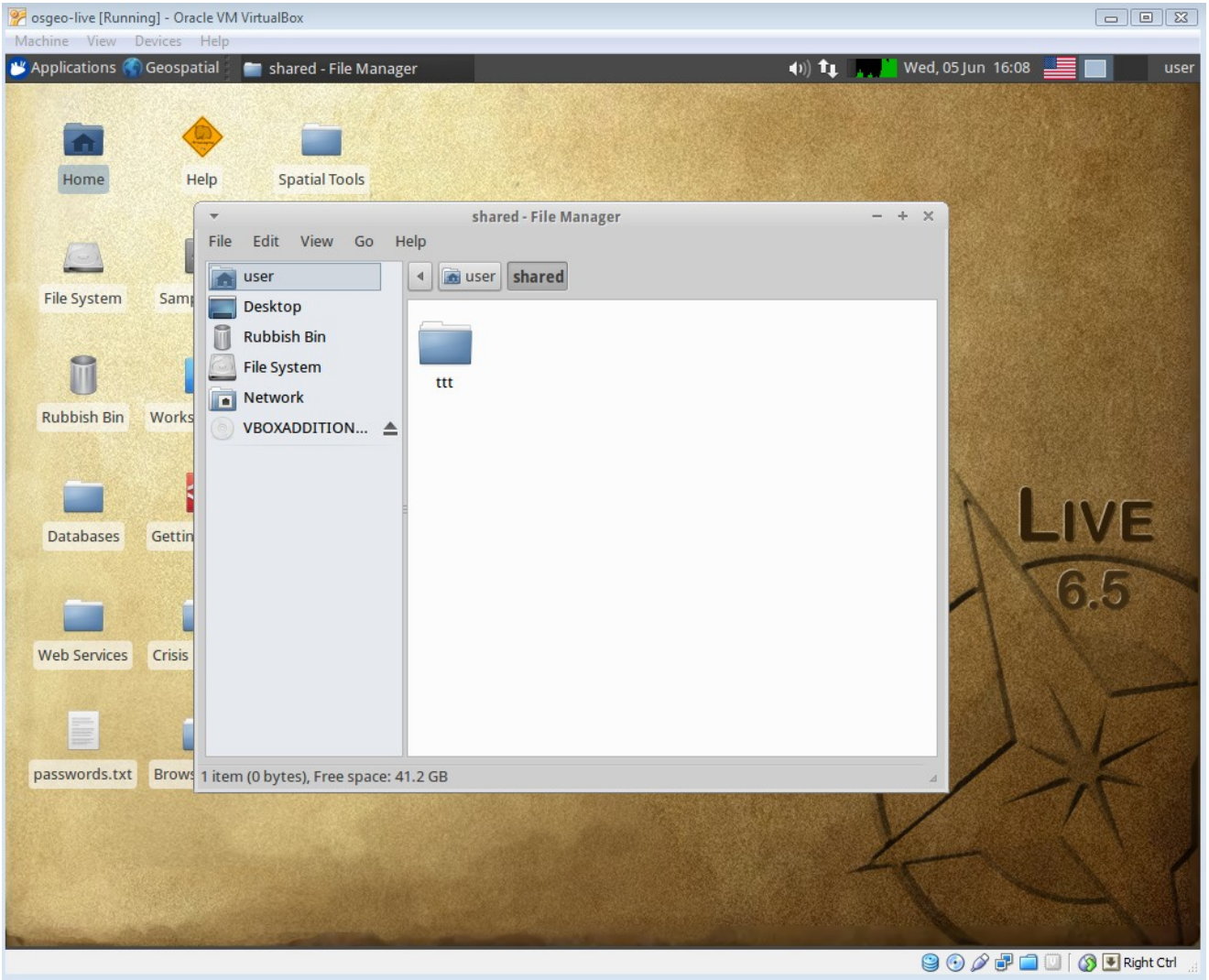
- reboot the machine (click on "user" in top-right of desktop, and select "Reboot")

- Open a Terminal window again, and execute the following (where "osgeo-live-shared" is the name you entered earlier in the Settings for the shared folder):

```
user@osgeolive:~$ mkdir shared

user@osgeolive:~$ sudo mount -t vboxsf -o uid=user,rw osgeo-live-shared /home/user/shared
```

You can now create a test folder on your local machine (in my case "ttt") and then view it within the virtual machine.



Metadata Background

Metadata is often described as "data about data", or the *who*, *what*, *where*, and *when*. In the spatial world, for each dataset we maintain, we should record information about the data such as:

- general description
- location
- usage restrictions
- projection
- technical contact
- date created
- date modified
- version

Benefits

Maintaining metadata for your datasets is important for several reasons:

1. Internal: local management
    - tracking dataset management
    - scheduling data updates
2. External: discovery
    - allowing your dataset to be used outside your organization

Standardized Metadata

With the growth of geographic information systems (GIS) in the 80's and 90's, geographic datasets became a requirement for decision makers across the world. The expansion of the Internet to share information through the late 90's and 2000's has now brought 'discovery' of geographic data into the hands of the average citizen.

Note

Metadata standards have been introduced since the mid-90's with the goals of:

- outlining specific required parameters
- common terminology
- consistency
- interoperability

Dublin Core

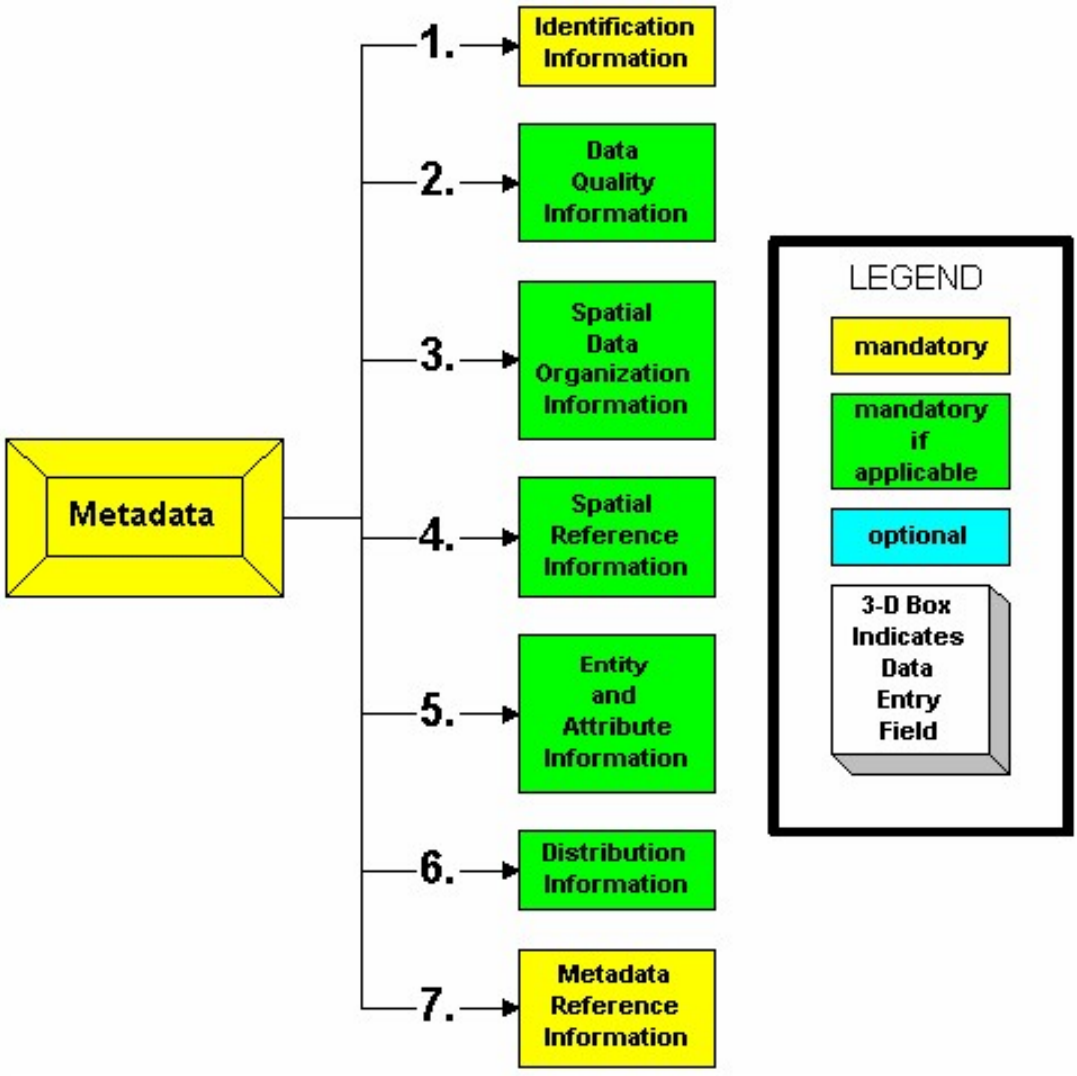
- named from workshop in Dublin, Ohio in 1995
- established a core/common group of 15 metadata elements

Example:

```
<head profile="http://dublincore.org">
..
<meta name="DC.Identifier" schema="DCterms:URI"
content="http://tutorialsonline.info/Common/DublinCore.html" />
<meta name="DC.Format" schema="DCterms:IMT" content="text/html" /> <meta name="DC.Title" xml
<meta name="DC.Creator" content="Alan Kelsey" />
<meta name="DC.Subject" xml:lang="EN" content="Dublin Core Meta Tags" />
<meta name="DC.Publisher" content="Alan Kelsey, Ltd." />
<meta name="DC.Publisher.Address" content="alan@tutorialsonline.info" />
<meta name="DC.Contributor" content="Alan Kelsey" />
<meta name="DC.Date" scheme="ISO8601" content="2007-01-06" />
<meta name="DC.Type" content="text/html" />
<meta name="DC.Description" xml:lang="EN"
  content="Learning Advanced Web Design can be fun and easy! Look at a site designed specifi
<meta name="DC.Identifier" content="http://tutorialsonline.info/Common/DublinCore.html" />
<meta name="DC.Relation" content="TutorialOnline.info" scheme="IsPartOf" />
<meta name="DC.Coverage" content="Hennepin Technical College" />
<meta name="DC.Rights" content="Copyright 2011, Alan Kelsey, Ltd.  All rights reserved." />
<meta name="DC.Date.X-MetadataLastModified" scheme="ISO8601" content="2007-01-06" />
<meta name="DC.Language" scheme="dcterms:RFC1766" content="EN" />
```

FGDC Content Standard for Digital Geospatial Metadata (CSDGM)

- approved by the U.S. Federal Geographic Data Committee originally in 1994
- composed of Sections, Compound Elements, Data Elements



ISO 19115

- International Standards Organization's TC211 committee created this in 2003
- consisting of more than 400 "Core", "Mandatory", and "Optional" elements

Table 3 — Core metadata for geographic datasets

Dataset title (M) (MD_Metadata > MD_DataIdentification.citation > CI_Citation.title)	**Spatial representation type** (O) (MD_Metadata > MD_DataIdentification.spatialRepresentationType)
Dataset reference date (M) (MD_Metadata > MD_DataIdentification.citation > CI_Citation.date)	**Reference system** (O) (MD_Metadata > MD_ReferenceSystem)
Dataset responsible party (O) (MD_Metadata > MD_DataIdentification.pointOfContact > CI_ResponsibleParty)	**Lineage** (O) (MD_Metadata > DQ_DataQuality.lineage > LI_Lineage)
Geographic location of the dataset (by four coordinates or by geographic identifier) (C) (MD_Metadata > MD_DataIdentification.extent > EX_Extent > EX_GeographicExtent > EX_GeographicBoundingBox or EX_GeographicDescription)	**On-line resource** (O) (MD_Metadata > MD_Distribution > MD_DigitalTransferOption.onLine > CI_OnlineResource)
Dataset language (M) (MD_Metadata > MD_DataIdentification.language)	**Metadata file identifier** (O) (MD_Metadata.fileIdentifier)
Dataset character set (C) (MD_Metadata > MD_DataIdentification.characterSet)	**Metadata standard name** (O) (MD_Metadata.metadataStandardName)
Dataset topic category (M) (MD_Metadata > MD_DataIdentification.topicCategory)	**Metadata standard version** (O) (MD_Metadata.metadataStandardVersion)
Spatial resolution of the dataset (O) (MD_Metadata > MD_DataIdentification.spatialResolution > MD_Resolution.equivalentScale or MD_Resolution.distance)	**Metadata language** (C) (MD_Metadata.language)
Abstract describing the dataset (M) (MD_Metadata > MD_DataIdentification.abstract)	**Metadata character set** (C) (MD_Metadata.characterSet)
Distribution format (O) (MD_Metadata > MD_Distribution > MD_Format.name and MD_Format.version)	**Metadata point of contact** (M) (MD_Metadata.contact > CI_ResponsibleParty)
Additional extent information for the dataset (vertical and temporal) (O) (MD_Metadata > MD_DataIdentification.extent > EX_Extent > EX_TemporalExtent or EX_VerticalExtent)	**Metadata date stamp** (M) (MD_Metadata.dateStamp)

ISO 19139

- the XML implementation schema for ISO 19115 specifying the metadata record format
- may be used to describe, validate, and exchange geospatial metadata prepared in XML

OGC CSW Specification

The Open Geospatial Consortium (OGC) OpenGIS Catalog Service Implementation Specification, currently at version 2.0.2, is a standard for discovering and retrieving spatial data and metadata. Catalog Services for the Web (CSW) is a profile/part of the Catalog Service Implementation Specificaton that allows for publishing and searching of metadata.

Operations

```
−<ows:OperationsMetadata>
  +<ows:Operation name="GetCapabilities"></ows:Operation>
  +<ows:Operation name="GetRepositoryItem"></ows:Operation>
  +<ows:Operation name="DescribeRecord"></ows:Operation>
  +<ows:Operation name="GetDomain"></ows:Operation>
  +<ows:Operation name="GetRecordById"></ows:Operation>
  +<ows:Operation name="GetRecords"></ows:Operation>
  +<ows:Parameter name="version"></ows:Parameter>
  +<ows:Parameter name="service"></ows:Parameter>
  +<ows:Constraint name="XPathQueryables"></ows:Constraint>
  +<ows:Constraint name="PostEncoding"></ows:Constraint>
  +<inspire_ds:ExtendedCapabilities xsi:schemaLocation="http://inspire.ec.europa.eu/
    </inspire_ds:ExtendedCapabilities>
</ows:OperationsMetadata>
```

CSW defines several possible operations to discover and retrieve metadata, and groups these operations into 3 "classes":

Service Class

- GetCapabilities (mandatory) - allow clients to retrieve information describing the service instance

Discovery Class

- DescribeRecord (mandatory) - allows a client to discover elements of the information model supported by the target catalog service
- GetRecords (mandatory) - get metadata records
- GetRecordById (optional) - get metadata records by ID
- GetDomain (optional) - obtain runtime information about the range of values of a metadata record element or request parameter.

Management Class

- Harvest (optional) - references the data to be inserted or updated in the catalog
- Transaction (optional) - defines an interface for creating, modifying and deleting catalog records.

Example Live Requests

- GetCapabilities
- DescribeRecord
- GetRecords
- GetRecordById
- GetDomain
- Harvest
- Transaction

Introduction to pycsw

pycsw is a lightweight metadata publisher, written in Python. It is easily configured, and can plug into your architecture.

The following sections will introduce your to the powers of pycsw.

Goals

Initially conceived in 2010, the overal vision of the development team was to:

1. **Create an Open Source standalone metadata publisher in Python**

Many other metadata publishing options exist, but mostly in Java. Python is an Open Source scripting language, that is supported on all major platforms, and is very popular in the geospatial world today.

2. **Make it lightweight and easy to configure: focused on one task, publishing**

   Keep the goals of the project to simple metadata publishing; don't get into other tasks such as metadata editing and acquisition. Rather than a bloated "kitchen sink" concept. limit the software to easy metadata publishing.

3. **Design so additional metadata formats can be easily supported.**

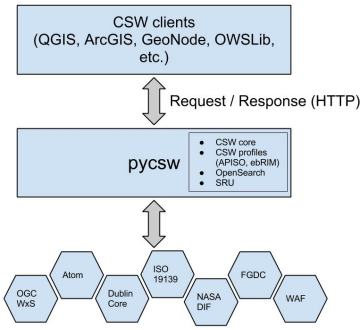
   Although initially the core metadata model was Dublic Core, design the sofware so that many other metadata formats can be plugged in.

Note

The initial pycsw 'manifesto' is still available here. It is a nice way to understand the project's vision.

Features

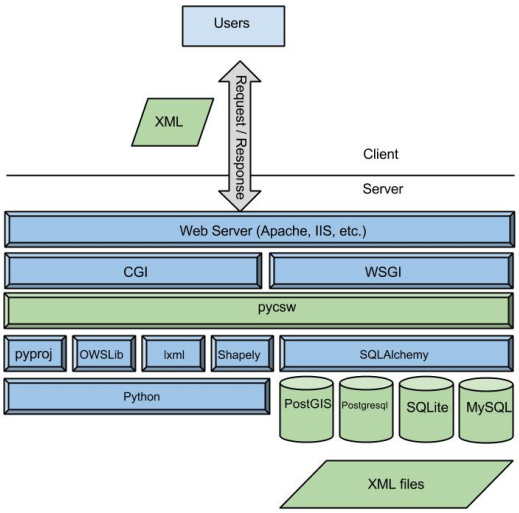
Component Architecture



Use Cases

Installing

Software Architecture



Configuring

Exercises

Advanced pycsw

Contents:

CSW-T

- CSW class for management functions
- write access to repository
- insert, update, delete
- CSW has no authentication mechanism; left up to provider
- push (`Transaction`) or pull (`Harvest`) operations

Warning

Enabling CSW-T opens up for write access of your data via HTTP

Enabling CSW-T in pycsw

- set `manager.transactions` to `true`
- pycsw uses IP authentication
- limit read/write access to list of allowed IP addresses
    - set in configuration (`manager.allowed_ips`)

- IP address like `192.168.0.11`
- wildcards like `192.168.0.*`
- CIDR like `192.168.100.0/24`

Transactions

- 'push' metadata into repository
- pycsw does no application level backup/versioning

Note

Always optimize your database accordingly (`pycsw-admin.py -c optimize_db`, `VACUUM ANALYZE`, etc.) when making changes to the repository

Insert

- inserts a record into the repository
  - like `insert into table values (...);`
- pycsw honours the identifier in the metadata
  - if absent, pycsw sets identifier
- example: https://raw.github.com/geopython/pycsw/master/tests/suites/manager/post/Transaction-dc-01-insert.xml

Note

Always ensure your metadata has an identifier

Update (full)

- full update of metadata record
- will update based on identifier found in XML
  - if no identifier, pycsw will insert as a new record
- example: https://raw.github.com/geopython/pycsw/master/tests/suites/manager/post/Transaction-dc-02-update-full.xml

Update (property)

- partial update of metadata record(s)
  - like `update table set foo="bar"`
- you can apply an OGC filter to make updates on specific records
  - like `update table set foo="bar" where identifier=12345`
- example: https://raw.github.com/geopython/pycsw/master/tests/suites/manager/post/Transaction-iso-03-update-recprop.xml

Delete

- deletes record(s) from the repository
  - like `delete from table where identifier=12345`
- example: https://raw.github.com/geopython/pycsw/master/tests/suites/manager/post/Transaction-iso-05-delete.xml

Harvesting

- 'pull' metadata into repository from remote URL
- pycsw supports many formats for harvesting
  - WMS, WFS, WCS, WPS, WAF, SOS
  - Dublin Core, FGDC, ISO, RDF
  - even other CSW servers

Note

Always optimize your database accordingly (`pycsw-admin.py -c optimize_db`, `VACUUM ANALYZE`, etc.) when making changes to the repository

Exporting

- dump all records in pycsw repository
- use `pycsw-admin.py` to export all records to XML files on disk

```
$ pycsw-admin.py -c export_records -f /path/to/default.cfg -p /tmp/metadata
```

- creates files in /tmp/metadata
- files are named by metadata record identifier
- want to import them back into another repository?

```
$ pycsw-admin.py -c setup_db -f /path/to/default.cfg
$ pycsw-admin.py -c load_records -f /path/to/default.cfg -p /tmp/metadata
```

Tips and Tricks

Importing Metadata Recursively

- use the `pycsw-admin.py -r` switch

```
$ pycsw-admin.py -c load_records -f path/to/default.cfg -p /path/to/metadata -r
```

Making CSW XML POST requests

- different from traditional HTTP POST
  - no form key/value pairs
  - client opens HTTP connection and send XML directly

Using `pycsw-admin.py`:

```
$ pycsw-admin.py -c post_xml -u http://labs.gatewaygeomatics.com/csw -x /path/to/request.xml
```

Using curl:

```
$ curl -H "Content-Type: text/xml" -X POST -d @request_file.xml http://labs.gatewaygeomatics
```

JSON Output

- for DescribeRecord, GetRecordById, GetRecords
- set `outputformat` to `application/json` as part of request

Get Raw Metadata

- use GetRepositoryItem (based on ebRIM profile)

```
$ GET "http://labs.gatewaygeomatics.com/csw?service=CSW&version=2.0.2&request=GetRepository
```

Optimizing the Repository

```
$ pycsw-admin.py -c optimize_db
```

Dependency Tracing

- use `pycsw-admin.py -c get_sysprof`
- valuable when multiple versions of pycsw and / or supporting libraries are on the same system

Debugging Issues

- turn on logging (set `server.loglevel` to `DEBUG` and `server.logfile` to a writable file)
- set `server.pretty_print` to `true`
- monitor logfile when testing (i.e. `tailf /path/to/pycsw-log.txt`)
- report issues / bugs to pycsw issue tracker / mailing list
- specify environment and supporting libraries (i.e. `pycsw-admin.py -c get_sysprof`)

Community

Exercises

pycsw and Open Data

- pycsw is embedded in various Open Data portal software
- metadata editing / management done with portal
- portal exposes CSW service automagically
- easy integration into existing apps/workflows

Contents:

GeoNode

- Open Source Geospatial Content Management System
  - geospatial data / metadata management
  - interactive mapping
  - collaboration

- pycsw enabled out of the box
    - embedded CSW

Open Data Catalog

- Code for America app
- open data publishing
- pycsw enabled out of the box
    - CSW embdedded

pycsw Future Development

1.6.0 (June 2013)

- extended harvesting (WAF, SOS, RDF)
- ISO 19115-2 (gmi) support
- spatial relevance ranking
- enhanced OGC filter support
- flexible administation enhancements

1.8 / 2.0

- CSW 3.0
- OPeNDAP integration via pydap
- THREDDS catalog harvesting
- CKAN integration
- native spatial databases
- backends (GeoCouch)
- PostgreSQL full text search (FTS)
- enhanced harvesting / additional formats / APIs
- search engine libraries
- Open Data metadata JSON summary format