XSEDE[16]

Tutorial – An Introduction to Brown Dog

An Elastic Data Cyberinfrastructure for Autocuration
and Digital Preservation

Monday, July 18, 2016

1:00 pm – 1:30 pm	Introduction to Brown Dog	Kenton McHenry

How to Use Brown Dog Services

1:30 pm – 1:45 pm	Hands-on Outline Environment Set-up	Smruti Padhy
1:45 pm – 2:05 pm	Demonstration of BD Fiddle	Jong Sung Lee
2:05 pm – 2:55 pm	Create Applications Using BD Services - Conversion Example - Extraction/Indexing/Retrieval Example - Combined Conversion and Extraction Example	 Jong Sung Lee Marcus Slavenas Marcus Slavenas
2:55 pm – 3:10 pm	Break	

How to Add Your Tool to Brown Dog Services

3:10 pm – 3:50 pm	How to Write a Converter	Kenton McHenry
3:50 pm – 4:30 pm	How to Write an Extractor	Smruti Padhy
4:30 pm – 5:00 pm	Wrap Up and Feedback	Team

Linux Cheat Sheet

Basic Commands

Basic Terminal Shortcuts

CTRL L = Clear the terminal
CTRL D = Logout
SHIFT Page Up/Down = Go up/down the terminal
CTRL A = Cursor to start of line
CTRL E = Cursor the end of line
CTRL U = Delete left of the cursor
CTRL K = Delete right of the cursor
CTRL W = Delete word on the left
CTRL Y = Paste (after CTRL U,K or W)
TAB = auto completion of file or command
CTRL R = reverse search history
!! = repeat last command
CTRL Z = stops the current command (resume with fg in foreground or bg in background)

Basic Terminal Navigation

ls -a = list all files and folders
ls <folderName> = list files in folder
ls -lh = Detailed list, Human readable
ls -l *.jpg = list jpeg files only
ls -lh <fileName> = Result for file only

cd <folderName> = change directory
    if folder name has spaces use " "
cd / = go to root
cd .. = go up one folder, tip: ../../../

du -h: Disk usage of folders, human readable
du -ah: " " " files & folders, Human readable
du -sh: only show disc usage of folders

pwd = print working directory

man <command> = shows manual (RTFM)

File Permissions

chown = change the owner of a file
    ex --> chown bob hello.txt
chown user:bob report.txt = changes the user owning report.txt to 'user' and the group owning it to 'bob'
-R = recursively affect all the sub folders
    ex --> chown -R bob:bob /home/Daniel

chmod = modify user access/permission – simple way
    u = user
    g = group
    o = other

    d = directory (if element is a directory)
    l = link (if element is a file link)
    r = read (read permissions)
    w = write (write permissions)
    x = eXecute (only useful for scripts and programs)

Basic file manipulation

cat <fileName> = show content of file
        (less, more)
head = from the top
    -n <#oflines> <fileName>

tail = from the bottom
    -n <#oflines> <fileName>

mkdir = create new folder
mkdir myStuff ..
mkdir myStuff/pictures/ ..

cp image.jpg newimage.jpg = copy and rename a file
cp image.jpg <folderName>/ = copy to folder
cp image.jpg folder/sameImageNewName.jpg
cp -R stuff otherStuff = copy and rename a folder
cp *.txt stuff/ = copy all of *<file type> to folder

mv file.txt Documents/ = move file to a folder
mv <folderName> <folderName2> = move folder in folder
mv filename.txt filename2.txt = rename file
mv <fileName> stuff/newfileName
mv <folderName>/ .. = move folder up in hierarchy

rm <fileName> .. = delete file (s)
rm -i <fileName> .. = ask for confirmation each file
rm -f <fileName> = force deletion of a file
rm -r <foldername>/ = delete folder

touch <fileName> = create or update a file

ln file1 file2 = physical link
ln -s file1 file2 = symbolic link

File Permissions (continued)

'+' means add a right
'-' means delete a right
'=' means affect a right

ex --> chmod g+w someFile.txt
    (add to current group the right to modify someFile.txt)

more info: man chmod

Accessing BD Services locally installed in the VM

- To generate tokens in locally installed bdfiddle

  username: **bdtut.user@xsede.org**
  password: **bdtutXy71816de**

- While using key_token.py to obtain tokens to access local bdfiddle, you need to type in the following values
  - Server [bd-api.ncsa.illinois.edu]: localhost:8080
  - Protocol [https]: http
  - Username: bdtut.user@xsede.org
  - Password: bdtutXy71816de

Index of /tutorials

Name	Last modified	Size	Description
Parent Directory		-	
01-BrownDog.pptx	2016-07-08 12:27	78M	
02-Setup.pptx	2016-07-08 12:27	6.6M	
03-BDFiddle.pptx	2016-07-08 12:27	19M	
04-Conversions.pptx	2016-07-08 12:27	3.3M	
05-Extractions.pptx	2016-07-08 11:35	5.0M	
06-Combined.pptx	2016-07-08 12:27	3.7M	
07-AddingConverters.pptx	2016-07-13 14:43	5.3M	
08-AddingExtractors.pptx	2016-07-08 12:27	6.1M	
09-ToolsCatalog-updated.pptx	2016-07-08 12:27	4.1M	
Template.pptx	2016-07-08 12:27	3.2M	
handouts.zip	2016-07-14 16:21	1.5M	
old/	2016-07-14 16:22	-	
samplefiles.zip	2016-07-14 16:21	4.8M	

Apache/2.4.7 (Ubuntu) Server at browndog.ncsa.illinois.edu Port 80

http://browndog.ncsa.illinois.edu/tutorials/

◉ File: ○ URL:

Browse...

Get Transformations

Token (Generate Token)

BD API Token

Select Options:

Convert:

Extract:

Submit

Output:

Code Snippets:

Terminal

Copy To Clipboard Download Code

Python
bd.py

Copy To Clipboard Download Code

Python
Native

Copy To Clipboard Download Code

bd.r

Copy To Clipboard Download Code

Matlab
bd.m

Copy To Clipboard Download Code

JavaScript
bd.js

Copy To Clipboard Download Code

```python
#!/usr/bin/python -u
import bd
import time
import os
from os.path import isfile, join, splitext, basename

#parameters to access BD services
bds = 'https://bd-api.ncsa.illinois.edu'

# key and token from the Brown Dog API Gateway service
api_key = ""
token = ""

# path to the folder to process
input_path = ""

# path to the folder to store output
# add time.time() to make the output folder unique for each run
output_path = ""+str(int(time.time()))

# output format
output_format = "pdf"

# Create a output folder
os.makedirs(output_path)

# create a list of files that is not a folder
onlyfiles = [join(input_path, f) for f in os.listdir(input_path) if
isfile(join(input_path, f))]

for (input_file) in onlyfiles:
    filename, file_extension = splitext(basename(input_file))
    input_format = file_extension[1:]

    print 'File: ' + basename(input_file) + ', ' + input_format + ' to '+ output_format

    # getting possible output format by the input format
    outputs = bd.outputs(bds, input_format, token)

    # if the output format is not supported, skip the conversion
    if not (output_format in outputs):
        print output_format, 'is not supported for', input_format
        continue

    # craete a output file name with path
    output_file = output_path + '/' + filename + "." + output_format

    # do the conversion
    bd.convert(bds, input_file, output_format, output_file, token, 60, True)
```

```
#!/usr/bin/python -u
import bd
import os
from os.path import isfile, join, splitext, basename

# this is the method to search dictionary by key
def findItem(obj, key):
    if key in obj: return obj[key]
    for k, v in obj.items():
        if isinstance(v, dict):
            item = findItem(v, key)
            if item is not None:
                return item

# this is the method to search key with given string
def search(values, searchFor):
   found = []
   for k in values:
      for v in values[k]:
         if searchFor in v.lower():
            found.append(k)
   return found

#parameters to access BD services
bds = 'https://bd-api.ncsa.illinois.edu'

# key and token from the Brown Dog API Gateway service
token = ""

# path to the folder to process
input_path = ""

# create a list of files that is not a folder
onlyfiles = [join(input_path, f) for f in os.listdir(input_path) if
isfile(join(input_path, f))]

text_store = {}

# processing the files in the given folder
for (input_file) in onlyfiles:
    filename, file_extension = splitext(basename(input_file))
    input_format = file_extension[1:]

    print 'Processing file: ' + basename(input_file)

    # do the extraction
    metadata = bd.extract(bds, input_file, token, 120)

    # process the metadata
    # find the "OCR" results and store
    for m in metadata['metadata.jsonld']:
        txt = findItem(m, 'ocr_simple')
        if not (txt is None):
            text_store[basename(input_file)] = txt

# find the keyword from the ocr results
print search(text_store, 'information')
```

```python
#!/usr/bin/python -u
import bd
import time
import os
from os.path import isfile, join, splitext, basename

#parameters to access BD services
bds = 'https://bd-api.ncsa.illinois.edu'

# token from the Brown Dog API Gateway service
token = ""

# path to the folder to process
input_file = ""

# path to the folder to store output
output_path = ""+str(int(time.time()))

# output format
output_format = "png"

def main():
    # create a output folder
    os.makedirs(output_path)

    # convert the image file to png
    output = convert(input_file, output_format, output_path, bds, token)

    # extract metadata from the converted file
    metadata = extract(output, bds, token)

    # print the "tag" section for to see the face detected
    print metadata['tags']

def convert(input_file, output_format, output_path, bds, token):
    filename, file_extension = splitext(basename(input_file))
    input_format = file_extension[1:]

    print 'File: ' + basename(input_file) + ', ' + input_format + ' to '+ output_format

    # getting possible output format by the input format
    outputs = bd.outputs(bds, input_format, token)

    # craete a output file name with path
    output_file = output_path + '/' + filename + "." + output_format

    # do the conversion
    return bd.convert(bds, input_file, output_format, output_file, token, 60, True)

def extract(extract_file, bds, token):
    print 'Processing File: ' + extract_file
    extract_output = bd.extract(bds, extract_file, token, 120)
    return extract_output

if __name__ == "__main__":
    main()

```

Template – ImageMagick_convert.sh

```
#!/bin/sh
#ImageMagick (v6.5.2)
#image
#bmp, dib, eps, fig, gif, ico, jpg, jpeg
#bmp, dib, eps, gif, jpg, jpeg, jp2, pcd, pdf, pgm

# NOTE: First four lines of comments above MUST follow pattern:
#Full software name (Version)
#Data types dealt with
#Comma separated list of inputs accepted
#Comma separated list of outputs accepted

output_filename=$(basename "$2")
output_format="${output_filename##*.}"

#Output PGM files as ASCII
if [ "$output_format" = "pgm" ]; then
  convert "$1" -compress none "$2"
else
  convert "$1" "$2"
fi
```

Exercise (/home/ubuntu/git/bd-tutorial-problems/07-AddingConverters/)

1. Write a wrapper script

   ```
   nano sciencify_convert.sh
   ```

2. Modify the SoftwareServer configuration (i.e. copy the script and edit alises)

   ```
   cp sciencify_convert.sh /home/ubuntu/git/polyglot/scripts/sh
   ```

   ```
   nano /home/ubuntu/git/polyglot/scripts/sh/.aliases.txt
   ```

3. Start the SoftwareServer (you can do this part in the bd-tmux pane)

   ```
   cd /home/ubuntu/git/polyglot/
   ```

   ```
   bin/SoftwareServerRestlet.sh
   ```

4. Test it with the BD CLI

   ```
   cd /home/ubuntu/git/bdcli
   ```

   ```
   bd –o science.txt examples/sonnet80.txt
   ```

```python
#!/usr/bin/env python
import logging
import numpy
import csv
from config import *
import pyclowder.extractors as extractors

def main():
    global extractorName, messageType, rabbitmqExchange, rabbitmqURL, logger, registrationEndpoints

    # Set logging
    logging.basicConfig(format='%(asctime)-15s %(levelname)-7s : %(name)s - %(message)s', level=logging.WARN)
    logging.getLogger('pyclowder.extractors').setLevel(logging.INFO)
    logger = logging.getLogger('extractor')
    logger.setLevel(logging.DEBUG)

    extractors.setup(extractorName=extractorName, messageType=messageType, rabbitmqURL=rabbitmqURL,
                     rabbitmqExchange=rabbitmqExchange)

    # Register extractor
    extractors.register_extractor(registrationEndpoints)

    # Connect to extractor and wait for files
    extractors.connect_message_bus(extractorName=extractorName, messageType=messageType,
                                   rabbitmqURL=rabbitmqURL,
                                   rabbitmqExchange=rabbitmqExchange,
                                   processFileFunction=process_file,
                                   checkMessageFunction=None)

def process_file(parameters):
    global field_name

    inputfile = parameters['inputfile']

    mean_desc = 'Mean total monthly precipitation'
    median_desc = 'Median total monthly precipitation'
    standard_dev_desc = 'Standard deviation of total monthly precipitation'

    # Initialize a records list and field name
    records = list()
    field_name = 'TPCP'

    # Read csv file contents into a variable
    csv_file = open(inputfile, 'rb')
    data_table = csv.DictReader(csv_file)

    # Read each row of the table and append the data into records array
    for data_row in data_table:
        records.append(float(data_row[field_name]))

    # Calculate mean, median and standard deviation and upload it back as metadata to Clowder
    mean = numpy.mean(records)
    median = numpy.median(records)
    std = numpy.std(records)

    # Context url
    context_url = "https://clowder.ncsa.illinois.edu/contexts/metadata.jsonld"

    # Store results as metadata
    metadata = {
        "@context": [context_url,
```

```
60                          {
61                              "mean": "http://clowder.ncsa.illinois.edu/"+ extractorName +
                                "#mean",
62                              "median": "http://clowder.ncsa.illinois.edu/" + extractorName
                                + "#median",
63                              "std": "http://clowder.ncsa.illinois.edu/" + extractorName +
                                "#std",
64                              "value": "http://clowder.ncsa.illinois.edu/" + extractorName +
                                "#value",
65                              "description": "http://clowder.ncsa.illinois.edu/"+
                                extractorName + "#description"
66                          }],
67          "attachedTo": {"resourceType": "file", "id": parameters["fileid"]},
68          "agent": {
69              "@type": "cat:extractor",
70              "extractor_id":
                "https://clowder.ncsa.illinois.edu/clowder/api/extractors/"+ extractorName
71          },
72          "content": {
73              "mean": {
74                  "value": round(mean, 3),
75                  "description": mean_desc
76              },
77              "median": {
78                  "value": round(median, 3),
79                  "description": median_desc
80              },
81              "std": {
82                  "value": round(std, 3),
83                  "description": standard_dev_desc
84              }
85          }
86      }
87
88      # upload metadata
89      extractors.upload_file_metadata_jsonld(mdata=metadata, parameters=parameters)
90      logger.info("Uploaded metadata %s", metadata)
91
92
93  if __name__ == '__main__':
94      main()
95
```

```
# ============================================================================
#
# In order for this extractor to run according to your preferences,
# the following parameters need to be set.
#
# Some parameters can be left with the default values provided here - in that
# case it is important to verify that the default value is appropriate to
# your system. It is especially important to verify that paths to files and
# software applications are valid in your system.
#
# ============================================================================

import os

# Name to show in rabbitmq queue list
extractorName = os.getenv('RABBITMQ_QUEUE', "ncsa.csv_stats")

# URL to be used for connecting to rabbitmq
rabbitmqURL = os.getenv('RABBITMQ_URI', "amqp://guest:guest@localhost/%2f")

# Name of rabbitmq exchange
rabbitmqExchange = os.getenv('RABBITMQ_EXCHANGE', "clowder")

# Type of files to process
messageType = "*.file.text.csv", "extractors." + extractorName

# Trust certificates, set this to false for self signed certificates
sslVerify = os.getenv('RABBITMQ_SSLVERIFY', False)

# Endpoints and keys for registering extractor information in CSV format.
registrationEndpoints = os.getenv('REGISTRATION_ENDPOINTS',

                                  "http://localhost:9000/clowder/api/extractors?key=key1,
                                  "
                                  "http://host2:9000/api/extractors?key=key2")

```

```
{
    "@context": "http://clowder.ncsa.illinois.edu/contexts/extractors.jsonld",
    "name": "ncsa.csv_stats",
    "version": "1.0",
    "description": "A simple JSON-LD extractor that calculates mean, median, and
    standard deviation of a column in a CSV file.",
    "author": "Sandeep Satheesan <sandeeps@illinois.edu>",
    "contributors": [],
    "contexts": [],
    "repository": {
        "repType": "git",
        "repUrl":
        "https://opensource.ncsa.illinois.edu/stash/scm/cats/extractors-tutorial.git"
    },
    "external_services": [],
    "dependencies": [],
    "bibtex": [],
    "dataset": ["gov.noaa.ncdc:C00841"]
}
```

XSEDE Tutorial Session: July 18, 2016
Introduction to Brown Dog
Tutorial Feedback

Thank you for attending our tutorial session. As a part of our on-going effort to improve the utility and usability of the tools we are developing we are asking attendees to answer the following questions. Note that all answers are anonymous. Comments are welcome anywhere!

---

1. What is your current job role/title?

_____

2. What level would you rate yourself in terms of writing code (in any language)?

1	2	3	4	5
expert		proficient		none

What programming languages are you familiar with?

_____

_____

3. How accessible was the content of presentations?

1	2	3	4	5
very		neutral		difficult to follow

Any suggestions as to what could be improved?

_____

_____

4. Please rate the perceived ease or difficulty with regards to calling/using brown dog via the API.

1	2	3	4	5
easy		neutral		difficult

5. Please rate the perceived ease or difficulty with regards to calling/using brown dog via any one of the sample clients.

1	2	3	4	5
easy		neutral		difficult

6. Please rate the perceived ease or difficulty with regards to adding a new converter or extractor.

1	2	3	4	5
easy		neutral		difficult

7. Was there anything in particular that you found confusing or counterintuitive? If so, what?

_____

XSEDE Tutorial Session: July 18, 2016
Introduction to Brown Dog
Tutorial Feedback

8. Do you have any suggestions for improving any of the interfaces or additional functionality you would like to see? Please describe:

_____
_____
_____
_____

9. Were there other material/topics you would have liked to have had presented? If so, what?

_____
_____

10. Is there material you think should be deleted? If so, what?

_____
_____

11. Are there any other relevant comments you want to share?

_____
_____
_____
_____

(Optional)

Name: _____Email:_____

Thank you for your feedback!
Please feel free to write additional comments below.