bonobo
Simple ETL in Python 3.5+
Romain Dorgueil
@rdorgueil

CTO/Hacker in Residence
Technical Co-founder
(Solo) Founder
Eng. Manager
Developer
Felt too young in a Linux Cauldron
Dismantler of Atari computers
Basic literacy using a Minitel
Guitars & accordions
Off by one baby

L’Atelier BNP Paribas
WeAreTheShops
RDC Dist. Agency
Sensio/SensioLabs
AffiliationWizard
STARTUP ACCELERATION PROGRAMS
NO HYPE, JUST BUSINESS

launchpad.atelier.net
bonobo
Simple ETL in Python 3.5+
Plan

• History of Extract Transform Load
  • Concept; Existing tools; Related tools; Ignition

• Practical Bonobo
  • Tutorial; Under the hood; Demo; Plugins & Extensions; More demos

• Wrap up
  • Present & future; Resources; Sprint; Feedback
Once upon a time...
Extract Transform Load

• **Not new.** Popular concept in the 1970s [1] [2]

• **Everywhere.** Commerce, websites, marketing, finance, …

Extract Transform Load
Extract Transform Load

Extract 

Transform more

Join

Transform

Load

HTTP POST

DB

log?
Data Integration Tools

- Pentaho Data Integration (IDE/Java)
- Talend Open Studio (IDE/Java)
- CloverETL (IDE/Java)
Data Integration Tools

- Java + IDE based, for most of them
- Data transformations are blocks
- IO flow managed by connections
- Execution

GUI first, eventually code :-(

In the Python world …

- Bubbles (https://github.com/stiivi/bubbles)
- PETL (https://github.com/alimanfoo/petl)
- (insert a few more here)
- and now… Bonobo (https://www.bonobo-project.org/)

You can also use amazing libraries including Joblib, Dask, Pandas, Toolz, but ETL is not their main focus.
Other scales...
Small Automation Tools

• Mostly aimed at simple recurring tasks.

• Cloud / SaaS only.
Big Data Tools

- Can do anything. And probably more. Fast.
- Either needs an infrastructure, or cloud based.
Story time
Partner 1 Data Integration
WE GOT DEALS !!!
Tiny bug there... Can you fix it?
My need

• A data integration / ETL tool using code as configuration.
• Preferably Python code.
• Something that can be tested (I mean, by a machine).
• Something that can use inheritance.
• Fast & cheap install on laptop, thought for servers too.
And that’s Bonobo
It is ...

• A framework to **write ETL jobs in Python 3** (3.5+)

• Using the same concepts as the old ETLs.

• You can use OOP!

Code first. Eventually a GUI will come.
It is NOT …

- Pandas / R Dataframes
- Dask (but will probably implement a dask.distributed strategy someday)
- Luigi / Airflow
- Hadoop / Big Data / Big Query / …
- A monkey (spoiler: it’s an ape, damnit french language…)
Let’s see...
Create a project

~ $ pip install bonobo

~ $ bonobo init europython/tutorial

~ $ bonobo run europython/tutorial
...demo

~/europython/tutorial $ bonobo run .

- range in=1 out=42
- Filter in=42 out=21
- print in=21

~/europython/tutorial $
import bonobo

def extract():
    yield 'euro'
    yield 'python'
    yield '2017'

def transform(s):
    return s.title()

def load(s):
    print(s)

graph = bonobo.Graph(
    extract,
    transform,
    load,
)
...demo

~ $ bonobo run .
...demo

~ $ bonobo run first.py
Under the hood...
graph = bonobo.Graph(...)

```python
CsvReader('clients.csv')

retrieve_orders

update_crm

InsertOrUpdate('db.site', 'clients', key='guid')

retrieve_orders
```
class Graph:

    def __init__(self, *chain):
        self.edges = {}
        self.nodes = []

        self.add_chain(*chain)

    def add_chain(self, *nodes, _input=None, _output=None):
        # ...
bonobo.run(graph)

or in a shell...
$ bonobo run main.py
CsvReader('clients.csv')

InsertOrUpdate('db.site', 'clients', key='guid')

retrieve_orders

update_crm

retrieve_orders

update_crm
CsvReader('clients.csv')

InsertOrUpdate('db.site', 'clients', key = 'guid')

update_crm

retrieve_orders

Context + Thread

Context + Thread

Context + Thread

Context + Thread

Thread

Thread

Thread

Thread
class GraphExecutionContext:
    def __init__(self, graph, plugins, services):
        self.graph = graph
        self.nodes = [
            NodeExecutionContext(node, parent=self)
            for node in self.graph
        ]
        self.plugins = [
            PluginExecutionContext(plugin, parent=self)
            for plugin in plugins
        ]
        self.services = services
class ThreadPoolExecutorStrategy(Strategy):
    def execute(self, graph, plugins, services):
        context = self.create_context(graph, plugins, services)
        executor = self.create_executor()

        for node_context in context.nodes:
            executor.submit(
                self.create_runner(node_context)
            )

        while context.alive:
            self.sleep()

        executor.shutdown()

        return context
</ implementation details >
Transformations

a.k.a

nodes in the graph
def get_more_infos(api, **row):
    more = api.query(row.get('id'))

    return {
        **row,
        **(more or {}),
    }
Generators

def join_orders(order_api, **row):
    for order in order_api.get(row.get('customer_id')):
        yield {
            **row,
            **order,
        }
Iterators

```python
extract = ('foo', 'bar', 'baz',
)

extract = range(0, 1001, 7)
```
Classes

class RiminizeThis:
    def __call__(self, **row):
        return {
            **row,
            'Rimini': 'Woo-hou-wo...',
        }

Anything, as long as it’s callable().
Configurable classes

```python
from bonobo.config import Configurable, Option, Service

class QueryDatabase(Configurable):
    table_name = Option(str, default='customers')
    database = Service('database.default')

    def call(self, database, **row):
        customer = database.query(self.table_name, customer_id=row['clientId'])
        return {
            **row,
            'is_customer': bool(customer),
        }
```
Configurable classes

```python
from bonobo.config import Configurable, Option, Service

class QueryDatabase(Configurable):
    table_name = Option(str, default='customers')
    database = Service('database.default')

    def call(self, database, **row):
        customer = database.query(self.table_name, customer_id=row['clientId'])
        return {
            **row,
            'is_customer': bool(customer),
        }
```
Configurable classes

```python
from bonobo.config import Configurable, Option, Service

class QueryDatabase(Configurable):
    table_name = Option(str, default='customers')

database = Service('database.default')

def call(self, database, **row):
    customer = database.query(self.table_name, customer_id=row['clientId'])
    return {
        **row,
        'is_customer': bool(customer),
    }
```
Configurable classes

from bonobo.config import Configurable, Option, Service

class QueryDatabase(Configurable):

table_name = Option(str, default='customers')
database = Service('database.default')

def call(self, database, **row):
    customer = database.query(self.table_name, customer_id=row['clientId'])
    return {
        **row,
        'is_customer': bool(customer),
    }
Configurable classes

```python
query_database = QueryDatabase(
    table_name='test_customers',
    database='database.testing',
)
```
Services
Define as names

def class QueryDatabase(Configurable):
    database = Service('database.default')

    def call(self, database, **row):
        return { ... }
import bonobo

graph = bonobo.Graph(...)  

def get_services():
    return {
        'database.default': MyDatabaseImpl()
    }
Bananas!
Library

bonobo.FileReader(...)
bonobo.CsvReader(...)
bonobo.JsonReader(...)
bonobo.PickleReader(...)
bonobo.ExcelReader(...)
bonobo.XMLReader(...)

... more to come

bonobo.FileWriter(...)
bonobo.CsvWriter(...)
bonobo.JsonWriter(...)
bonobo.PickleWriter(...)
bonobo.ExcelWriter(...)
bonobo.XMLWriter(...)

... more to come
Library

bonobo.Limit(limit)
bonobo.PrettyPrinter()
bonobo.Filter(...)

... more to come
Extensions & Plugins
Console Plugin
Jupyter Plugin

In [1]:
import bonobo
from bonobo.commands.run import get_default_services
from bonobo.examples.datasets import coffeeshops

In [2]:
bonobo.run(
    coffeeshops.graph,
    services=get_default_services(coffeeshops.__file__)
)

- OpenDataSoftAPI in=1 out=181, in=181 out=181
- FileWriter in=181 out=181

Out[2]: <bonobo.execution.graph.GraphExecutionContext at 0x1109fd6a0>
**SQLAlchemy Extension**

```python
bonobo_sqlalchemy.Select(
    query,
    *,
    pack_size=1000,
    limit=None
)

bonobo_sqlalchemy.InsertOrUpdate(
    table_name,
    *,
    fetch_columns,
    insert_only_fields,
    discriminant,
    ...
)
```
Docker Extension

$ pip install bonobo[docker]

$ bonobo runc myjob.py
Dev Kit

https://github.com/python-bonobo/bonobo-devkit
More examples
Use filesystem service.
Write to a CSV
Also write to JSON
Rimini open data
Europython attendees

featuring...

jupyter notebook
selenium & firefox

~/bdk/demos/europython2017
French companies registry

featuring...
docker
postgresql
sql alchemy

~/bdk/demos/sirene
Wrap up
Young

• First commit : December 2016

• 23 releases, ~420 commits, 4 contributors

• Current « stable » 0.4.3

• Target : 1.0 early 2018
Python 3.5+

- `{**}`
- async/await
- `(…, *, ...)`
- GIL :(
1.0

- 100% Open-Source.
- Light & Focused.
- Very few dependencies.
- Comprehensive standard library.
- The rest goes to plugins and extensions.
Small scale

• 1 minute to install
• Easy to deploy

• **NOT**: Big Data, Statistics, Analytics …
• **IS**: Lean manufacturing for data
Interwebs are crazy

spangry 50 days ago [‐]
I haven't tried this yet, but am praying that it delivers even half of what it promises.

Nydhal 50 days ago [‐]
You mean excel?

rkda 50 days ago [‐]
All those references to monkeys hurt my head. Bonobos are not monkeys.

nn3 50 days ago [‐]
The picture looks more like a Gorilla than a Bonobo too.

[‐] glebaron 14 points 1 month ago
Bonobos are apes, not monkeys. In some situations this distinction is very important.

[‐] alcalde 9 points 1 month ago
Not here. Python not only has duck typing; it has the little known primate typing feature.
Data Processing for Humans
www.bonobo-project.org

docs.bonobo-project.org

bonobo-slack.herokuapp.com

github.com/python-bonobo

Let me know what you think!
Sprint

- Sprints at Europython are amazing
- Nice place to learn about Bonobo, basics, etc.
- Nice place to contribute while learning.
- You’re amazing.
Thank you!

@monkcage

@rdorgueil

https://goo.gl/e25eoa