Elastic Git Documentation

Release 0.1.2

Simon de Haan

March 07, 2016

Contents

Elastic Git is a library for modelling data, storing it in git and querying it via elastic search.

```
>>> from elasticgit import EG
>>> from elasticgit.models import Model, IntegerField, TextField
>>>
>>> workspace = EG.workspace('.test_repo')
>>> workspace.setup('Simon de Haan', 'simon@praekeltfoundation.org')
>>>
>>> # Models can be defined like
>>> class Person(Model):
...     age = IntegerField('The Age')
...     name = TextField('The Name')
...
>>> # But for doctests we're going to import an existing one
>>> from elasticgit.tests.base import TestPerson as Person
>>> person1 = Person({'age': 10, 'name': 'Foo'})
>>> workspace.save(person1, 'Saving Person 1')
>>>
>>> person2 = Person({'age': 20, 'name': 'Bar'})
>>> workspace.save(person2, 'Saving Person 2')
>>>
>>> person3 = Person({'age': 30, 'name': 'Baz'})
>>> workspace.save(person3, 'Saving Person 3')
>>>
>>> # Elasticsearch does this automatically every few seconds
>>> # but not fast enough for unit tests.
>>> workspace.refresh_index()
>>>
>>> # Accessing the data ES knows about
>>> es_person1, es_person2 = workspace.S(
...     Person).filter(age__gte=20).order_by('-name')
>>> es_person1.name
u'Baz'
>>> es_person2.name
u'Bar'
>>>
>>> # Accessing the actual Person object stored in Git
>>> git_person1 = es_person1.get_object()
>>> git_person1.name
u'Baz'
>>> git_person1.age
30
>>>
>>> sorted(dict(git_person1).keys())
['_version', 'age', 'name', 'uuid']
>>>
```

CHAPTER 1

Using a Remote workspace

When paired with unicore.distribute it is possible to connect to a Git repository hosted on a network somewhere instead of needing file system access. This is done via the RemoteWorkspace.

In a distributed hosting environment this can help eliminate issues where applications may run on different servers than where the Git content repositories live.

```
>>> from elasticgit.workspace import RemoteWorkspace
>>> from unicore.content.models import Page
>>> rws = RemoteWorkspace('http://localhost:6543/repos/unicore-sample-content.json')
>>> rws.sync(Page)
({u'c63477768fe745809b411878ac9c0023'}, set())
>>> rws.S(Page).count()
1
>>> [page] = rws.S(Page)
>>> page.title
u'page title'
>>> page.uuid
u'c63477768fe745809b411878ac9c0023'
```

Note: Please note that the RemoteWorkspace is currently read only.

1.1 Workspace

class elasticgit.workspace.**EG**

A helper function for things in ElasticGit.

classmethod workspace(*workdir*, *es={}*, *index_prefix=None*)

Create a workspace

Parameters

- **workdir** (*str*) – The path to the directory where a git repository can be found or needs to be created when *Workspace.setup()* is called.
- **es** (*dict*) – The parameters to pass along to elasticutils.get_es()
- **index_prefix** (*str*) – The index_prefix use when generating index names for Elasticsearch

Returns *Workspace*

class elasticgit.workspace.**RemoteWorkspace**(*url*, *es=None*, *index_prefix=None*)
A workspace that connects to a unicore.distribute server hosted somewhere on the network.

This is a read only version of the *Workspace*

class elasticgit.workspace.**Workspace**(*repo*, *es*, *index_prefix*)
The main API exposing a model interface to both a Git repository and an Elasticsearch index.

Parameters

- **repo** (*git.Repo*) – A git.Repo instance.
- **es** (*dit*) – A dictionary of values one would pass to elasticutils.get_es to get an Elasticsearch connection
- **index_prefix** (*str*) – The prefix to use when generating index names for Elasticsearch

S(*model_class*)
Get a elasticutils.S object for the given model class. Under the hood this dynamically generates a elasticutils.MappingType and elasticutils.Indexable subclass which maps the Elasticsearch results to *elasticgit.models.Model* instances on the UUIDs.

Parameters **model_class** (elasticgit.models.Model) – The class to provide a search interface for.

delete(*model*, *message*, *author=None*, *committer=None*)
Delete a :py:class‘elasticgit.models.Model‘ instance from Git and the Elasticsearch index.

Parameters

- **model** (elasticgit.models.Model) – The model instance
- **message** (*str*) – The commit message to remove the model from Git with.
- **author** (*tuple*) – The author information (name, email address) Defaults repo default if unspecified.
- **committer** (*tuple*) – The committer information (name, email address). Defaults to the author if unspecified.

destroy()
Removes an ES index and a Git repository completely. Guaranteed to remove things completely, use with caution.

exists()
Check if the Git repository or the ES index exists. Returns True if either of them exist.

Returns bool

get_mapping(*model_class*)
Get a mapping from Elasticsearch for a model_class :param elasticgit.models.Model model_class: :returns: dict

index_ready()
Check if the index is ready

Returns bool

pull(*branch_name='master'*, *remote_name='origin'*)
Fetch & Merge in an upstream's commits.

Parameters

- **branch_name** (*str*) – The name of the branch to fast forward & merge in
- **remote_name** (*str*) – The name of the remote to fetch from.

refresh_index()
Manually refresh the Elasticsearch index. In production this is not necessary but it is useful when running tests.

reindex(*model_class*, *refresh_index=True*)
Same as *reindex_iter()* but returns a list instead of a generator.

reindex_iter(*model_class*, *refresh_index=True*)
Reindex everything that Git knows about in an iterator

Parameters

- **model_class** (elasticgit.models.Model) –
- **refresh_index** (*bool*) – Whether or not to refresh the index after everything has been indexed. Defaults to True

save(*model*, *message*, *author=None*, *committer=None*)
Save a *elasticgit.models.Model* instance in Git and add it to the Elasticsearch index.

Parameters

- **model** (elasticgit.models.Model) – The model instance
- **message** (*str*) – The commit message to write the model to Git with.
- **author** (*tuple*) – The author information (name, email address) Defaults repo default if unspecified.
- **committer** (*tuple*) – The committer information (name, email address). Defaults to the author if unspecified.

setup(*name*, *email*)
Setup a Git repository & ES index if they do not yet exist. This is safe to run if already existing.

Parameters

- **name** (*str*) – The name of the committer in this repository.
- **email** (*str*) – The email address of the committer in this repository.

setup_custom_mapping(*model_class*, *mapping*)
Add a custom mapping for a model class instead of accepting what the model_class defines.

Parameters

- **model_class** (elasticgit.models.Model) –
- **dict** – the Elastisearch mapping definition

Returns dict, the decoded dictionary from Elasticsearch

setup_mapping(*model_class*)
Add a custom mapping for a model_class

Parameters **model_class** (elasticgit.models.Model) –

Returns dict, the decoded dictionary from Elasticsearch

sync(*model_class*, *refresh_index=True*)
Resync a workspace, it assumes the Git repository is the source of truth and Elasticsearch is made to match. This involves two passes, first to index everything that Git knows about and unindexing everything that's in Elastisearch that Git does not know about.

Parameters

- **model_class** (elasticgit.models.Model) – The model to resync

- **refresh_index** (*bool*) – Whether or not to refresh the index after indexing everything from Git

1.2 Models

class elasticgit.models.**BooleanField**(*doc*, *required=False*, *default=None*, *static=False*, *fallbacks=()*, *mapping={}*, *name=None*)

A boolean field

default_mapping = {'type': 'boolean'}
Mapping for Elasticsearch

class elasticgit.models.**DictField**(*doc*, *fields*, *default=None*, *static=False*, *fallbacks=()*, *mapping=()*)

A dictionary field

class elasticgit.models.**FloatField**(*doc*, *required=False*, *default=None*, *static=False*, *fallbacks=()*, *mapping={}*, *name=None*)

A float field

default_mapping = {'type': 'float'}
Mapping for Elasticsearch

class elasticgit.models.**IntegerField**(*doc*, *required=False*, *default=None*, *static=False*, *fallbacks=()*, *mapping={}*, *name=None*)

An integer field

default_mapping = {'type': 'integer'}
Mapping for Elasticsearch

class elasticgit.models.**ListField**(*doc*, *fields*, *default=[]*, *static=False*, *fallbacks=()*, *mapping={}*)

A list field

default_mapping = {'type': 'string'}
Mapping for Elasticsearch

class elasticgit.models.**Model**(*config_data*, *static=False*, *es_meta=None*)

Base model for all things stored in Git and Elasticsearch. A very thin wrapper around confmodel.Config.

Subclass this model and add more field as needed.

Parameters **config_data** (*dict*) – A dictionary with keys & values to populate this Model instance with.

Configuration options:

Parameters

- **_version** (*dict*) – Model Version Identifier
- **uuid** (*str*) – Unique Identifier

set_read_only()

Mark this model instance as being read only. Returns self to allow it to be chainable.

Returns self

class elasticgit.models.**TextField**(*doc*, *required=False*, *default=None*, *static=False*, *fallbacks=()*, *mapping={}*, *name=None*)

A text field

class elasticgit.models.**URLField**(*doc*, *required=False*, *default=None*, *static=False*, *fallbacks=()*, *mapping={}*, *name=None*)

A url field

mapping = {'type': 'string'}

Mapping for Elasticsearch

class elasticgit.models.**UnicodeTextField**(*doc*, *required=False*, *default=None*, *static=False*, *fallbacks=()*, *mapping={}*, *name=None*)

A text field

1.3 Storage Manager

class elasticgit.storage.**StorageManager**(*repo*)

An interface to *elasticgit.models.Model* instances stored in Git.

Parameters **repo** (*git.Repo*) – The repository to operate on.

create_storage(*bare=False*)

Creates a new git.Repo

Parameters **bare** (*bool*) – Whether or not to create a bare repository. Defaults to False.

delete(*model*, *message*, *author=None*, *committer=None*)

Delete a model instance from Git.

Parameters

- **model** (elasticgit.models.Model) – The model instance
- **message** (*str*) – The commit message.
- **author** (*tuple*) – The author information (name, email address) Defaults repo default if unspecified.
- **committer** (*tuple*) – The committer information (name, email address). Defaults to the author if unspecified.

Returns The commit.

delete_data(*repo_path*, *message*, *author=None*, *committer=None*)

Delete a file that's not necessarily a model file.

Parameters

- **repo_path** (*str*) – Which file to delete.
- **message** (*str*) – The commit message.
- **author** (*tuple*) – The author information (name, email address) Defaults repo default if unspecified.
- **committer** (*tuple*) – The committer information (name, email address). Defaults to the author if unspecified.

Returns The commit

destroy_storage()

Destroy the repository's working dir.

get(*model_class*, *uuid*)

Get a model instance by loading the data from git and constructing the model_class

Parameters

- **model_class** (*elasticgit.models.Model*) – The model class of which an instance to return
- **uuid** (*str*) – The uuid for the object to retrieve

Returns :py:class:elasticgit.models.Model

get_data(*repo_path*)
Get the data for a file stored in git

Parameters **repo_path** (*str*) – The path to the file in the Git repository

Returns str

git_name(*model*)
Return the file path to where the data for a *elasticgit.models.Model* lives.

Parameters **model** (*elasticgit.models.Model*) – The model instance

Returns str

```
>>> from git import Repo
>>> from elasticgit.tests.base import TestPerson
>>> from elasticgit.storage import StorageManager
>>> person = TestPerson({'age': 1, 'name': 'Foo', 'uuid': 'the-uuid'})
>>> sm = StorageManager(Repo('.'))
>>> sm.git_name(person)
'elasticgit.tests.base/TestPerson/the-uuid.json'
>>>
```

git_path(*model_class*, **args*)
Return the path of a model_class when layed out in the git repository.

Parameters

- **model_class** (*class*) – The class to map to a path
- **args** (*tuple*) – Optional bits to join together after the path.

Returns str

```
>>> from git import Repo
>>> from elasticgit.tests.base import TestPerson
>>> from elasticgit.storage import StorageManager
>>> sm = StorageManager(Repo('.'))
>>> sm.git_path(TestPerson)
'elasticgit.tests.base/TestPerson'
>>> sm.git_path(TestPerson, 'some-uuid.json')
'elasticgit.tests.base/TestPerson/some-uuid.json'
>>>
```

iterate(*model_class*)
This loads all known instances of this model from Git because we need to know how to re-populate Elasticsearch.

Parameters **model_class** (*elasticgit.models.Model*) – The class to look for instances of.

Returns generator

load(*file_path*)
Load a file from the repository and return it as a Model instance.

Parameters **file_path** (*str*) – The path of the object we want a model instance for.

Returns *elasticgit.models.Model*

path_info(*file_path*)
Analyze a file path and return the object's class and the uuid.

Parameters **file_path** (*str*) – The path of the object we want a model instance for.

Returns (model_class, uuid) tuple or `None` if not a model file path.

pull(*branch_name='master'*, *remote_name=None*)
Fetch & Merge in an upstream's commits.

Parameters

- **branch_name** (*str*) – The name of the branch to fast forward & merge in
- **remote_name** (*str*) – The name of the remote to fetch from.

read_config(*section*)
Read a config block for a git repository.

Parameters **section** (*str*) – The section to read.

Returns dict

storage_exists()
Check if the storage exists. Returns `True` if the directory exists, it does not check if it is an actual `git.Repo`.

Returns bool

store(*model*, *message*, *author=None*, *committer=None*)
Store an instance's data in Git.

Parameters

- **model** (*elasticgit.models.Model*) – The model instance
- **message** (*str*) – The commit message.
- **author** (*tuple*) – The author information (name, email address) Defaults repo default if unspecified.
- **committer** (*tuple*) – The committer information (name, email address). Defaults to the author if unspecified.

Returns The commit.

store_data(*repo_path*, *data*, *message*, *author=None*, *committer=None*)
Store some data in a file

Parameters

- **repo_path** (*str*) – Where to store the file.
- **data** (*obj*) – The data to write in the file.
- **message** (*str*) – The commit message.
- **author** (*tuple*) – The author information (name, email address) Defaults repo default if unspecified.
- **committer** (*tuple*) – The committer information (name, email address). Defaults to the author if unspecified.

Returns The commit

write_config(*section*, *data*)
Write a config block for a git repository.

Parameters

- **section** (*str*) – The section to write the data for.
- **data** (*dict*) – The keys & values of data to write

1.4 Search Manager

class elasticgit.search.**ESManager**(*storage_manager*, *es*, *index_prefix*)
An interface to *elasticgit.models.Model* instances stored in Git.

Parameters

- **workspace** (elasticgit.workspace.Workspace) – The workspace to operate on.
- **es** (*elasticsearch.Elasticsearch*) – An Elasticsearch client instance.

create_index(*name*)
Creates the index in Elasticsearch

Parameters **name** (*str*) –

destroy_index(*name*)
Destroys the index in Elasticsearch

Parameters **name** (*str*) –

get_mapping(*name*, *model_class*)
Retrieve a mapping for a model class in a specific index

Parameters

- **name** (*str*) –
- **model_class** (elasticgit.models.Model) –

Returns dict

index(*model*, *refresh_index=False*)
Index a *elasticgit.models.Model* instance in Elasticsearch

Parameters

- **model** (elasticgit.models.Model) – The model instance
- **refresh_index** (*bool*) – Whether or not to manually refresh the Elasticsearch index. Useful in testing.

Returns *elasticgit.models.Model*

index_exists(*name*)
Check if the index already exists in Elasticsearch

Parameters **name** (*str*) –

Returns bool

index_name(*name*)
Generate an Elasticsearch index name using given name and prefixing it with the index_prefix. The resulting generated index name is URL quoted.

Parameters **name** (*str*) – The name to use for the index.

index_ready(*name*)
Check if an index is ready for use.

Parameters **name** (*str*) –

Returns bool

index_status(*name*)
Get an index status

Parameters **name** (*str*) –

raw_unindex(*model_class*, *uuid*, *refresh_index=False*)
Remove an entry from the Elasticsearch index. This differs from `unindex()` because it does not require an instance of *elasticgit.models.Model* because you're likely in a position where you don't have it if you're trying to unindex it.

Parameters

- **model_class** (elasticgit.models.Model) – The model class
- **uuid** (*str*) – The model's UUID
- **refresh_index** (*bool*) – Whether or not to manually refresh the Elasticsearch index. Useful in testing.

refresh_indices(*name*)
Manually refresh the Elasticsearch index. In production this is not necessary but it is useful when running tests.

Parameters **name** (*str*) –

setup_custom_mapping(*name*, *model_class*, *mapping*)
Specify a mapping for a model class in a specific index

Parameters

- **name** (*str*) –
- **model_class** (elasticgit.models.Model) –
- **mapping** (*dict*) – The Elasticsearch mapping definition

Returns dict

setup_mapping(*name*, *model_class*)
Specify a mapping for a model class in a specific index

Parameters

- **name** (*str*) –
- **model_class** (elasticgit.models.Model) –

Returns dict

unindex(*model*, *refresh_index=False*)
Remove a *elasticgit.models.Model* instance from the Elasticsearch index.

Parameters

- **model** (elasticgit.models.Model) – The model instance
- **refresh_index** (*bool*) – Whether or not to manually refresh the Elasticsearch index. Useful in testing.

Returns *elasticgit.models.Model*

class elasticgit.search.**SM**(*model_class*, *in_*, *index_prefixes=None*)
A search interface similar to elasticutils.S to retrieve elasticgit.search.ReadOnlyModelMappingType instances stored in Elasticsearch. These can be converted to elasticgit.model.Model instances using ReadOnlyModelMappingType.to_object().

Parameters

- **model_class** (*type*) – A subclass of *elasticgit.models.Model* for generating a mapping type.
- **in** (*list*) – A list of git.Repo instances, or a list of repo working dirs.
- **index_prefixes** (*list*) – An optional list of index prefixes corresponding to the repos in *in_*.

get_repo_indexes()
Generate the indexes corresponding to the repos.

Returns list

elasticgit.search.**index_name**(*prefix*, *name*)
Generate an Elasticsearch index name using given name and prefixing it with the given index_prefix. The resulting generated index name is URL quoted.

Parameters

- **prefix** (*str*) – The prefix to use for the index.
- **name** (*str*) – The name to use for the index.

Returns str

1.5 Utilities

elasticgit.utils.**fqcn**(*klass*)
Given a class give it's fully qualified class name in dotted notation. The inverse of *load_class*

Parameters **klass** (*class*) –

```
>>> from elasticgit.utils import fqcn
>>> from elasticgit.tests.base import TestPerson
>>> fqcn(TestPerson)
'elasticgit.tests.base.TestPerson'
>>>
```

elasticgit.utils.**introspect_properties**(*model_class*)
Introspect a *elasticgit.models.Model* and retrieve a suitable mapping to use when indexing instances of the model in Elasticsearch.

```
>>> from elasticgit.models import Model, TextField
>>>
>>> class TestModel(Model):
...     field = TextField('A text field')
...
>>> from elasticgit.utils import introspect_properties
>>>
>>> sorted(introspect_properties(TestModel).keys())
['_version', 'field', 'uuid']
>>>
```

elasticgit.utils.**load_class**(*class_path*)
Load a class by it's class path

Parameters **class_path** (*str*) – The dotted.path.to.TheClass

```
>>> from elasticgit.utils import load_class
>>> load_class('elasticgit.tests.base.TestPerson')
<class 'elasticgit.tests.base.TestPerson'>
>>>
```

1.6 Tools

Elastic Git provides various command line tools via the `elasticgit.tools` module.

```
$ python -m elasticgit.tools --help
```

Elasticgit command line tools.

```
usage: python -m elasticgit.tools [-h]
                                  {dump-schema,load-schema,migrate-gitmodel-repo,shell,version,resync
                                  ...
```

Sub-commands:

dump-schema Dump model information as an Avro schema.

```
usage: python -m elasticgit.tools dump-schema [-h] class_path
```

Positional arguments:

class_path	python path to Class.

load-schema Dump an Avro schema as an Elasticgit model.

```
usage: python -m elasticgit.tools load-schema [-h] [-m key=FieldType]
                                               [-r OldModelName=NewShiny]
                                               schema_file [schema_file ...]
```

Positional arguments:

schema_files	path to Avro schema file.

Options:

-m, --map-field	Manually map specific field names to Field classes. Formatted as ''field=IntegerField''
-r, --rename-model	Manually rename a model.Formatted as ''OldModel-Name=NewShiny''

migrate-gitmodel-repo Migrate a GitModel based repository layout to anElastic-Git repository layout

```
usage: python -m elasticgit.tools migrate-gitmodel-repo [-h]
                                                         working_dir
                                                         module_name
```

Positional arguments:

working_dir	The directory of git model repository to migrate.
module_name	The module to put the migrated data in.

shell Load a repo and make an EG workspace available for debugging

```
usage: python -m elasticgit.tools shell [-h] [-m MODELS] [-n] workdir
```

Positional arguments:

workdir Path to the repository's working directory.

Options:

-m, --models The models module to load.

-n=True, --no-introspect-models=True Do not find & load models automatically.

version Tools for versioning & version checking a content repository

```
usage: python -m elasticgit.tools version [-h] -n NAME -l LICENSE -a AUTHOR
                                          [-au AUTHOR_URL] [-f FILE_NAME]
```

Options:

-n, --name The name to give this repository

-l, --license The license the publish this content under.

-a, --author The author

-au, --author-url The url where to find more information about the author

-f=.unicore.json, --file=.unicore.json The file to write to. Set to '-' for stdout.

resync Tools for resyncing data in a git repository with what is in the search index.

```
usage: python -m elasticgit.tools resync [-h] [-c CONFIG_FILE] -m MODEL_CLASS
                                         [-s SECTION_NAME] [-i INDEX_PREFIX]
                                         [-u ES_HOST] [-p GIT_PATH]
                                         [-f MAPPING_FILE] [-r RECREATE_INDEX]
```

Options:

-c, --config Python paste config file.

-m, --model The model class to load.

-s=app:cmsfrontend, --section-name=app:cmsfrontend The section from where to read the config keys.

-i, --index-prefix The index prefix to use

-u, --es-host The elasticsearch url to use

-p, --git-path The path to the repository.

-f, --mapping-file The path to a custom mapping file.

-r=False, --recreate-index=False Whether or not to recreate the index from scratch.

1.6.1 Avro

class elasticgit.commands.avro.**FieldMapType**(*mapping*)

A custom type for providing mappings on the command line for the *SchemaLoader* tool.

Parameters **mapping** (*str*) – A mapping of a key to a field type

```
>>> from elasticgit.commands.avro import FieldMapType
>>> mt = FieldMapType('uuid=elasticgit.models.UUIDField')
>>> mt.key
'uuid'
>>> mt.field_class
<class 'elasticgit.models.UUIDField'>
>>>
```

class elasticgit.commands.avro.**RenameType**(*mapping*)
A custom type for renaming things.

Parameters **mapping** (*str*) – A mapping of an old name to a new name

```
>>> from elasticgit.commands.avro import RenameType
>>> rt = RenameType('OldName=NewName')
>>> rt.old
'OldName'
>>> rt.new
'NewName'
>>>
```

class elasticgit.commands.avro.**SchemaDumper**
Dump an Avro JSON schema for an Elasticgit Model.

```
python -m elasticgit.tools dump-schema elasticgit.tests.base.TestPerson
```

dump_schema(*model_class*)
Return the JSON schema for an *elasticgit.models.Model*.

Parameters **model_class** (elasticgit.models.Model) –

Returns str

get_field_info(*name*, *field*)
Return the Avro field object for an *elasticgit.models.Model* field.

Parameters

- **name** (*str*) – The name of the field
- **field** (*confmodel.fields.ConfigField*) – The field

Returns dict

run(*class_path*)
Introspect the given class path and print the schema to *self.stdout*

Parameters **class_path** (*str*) – The path to the model file to introspect

class elasticgit.commands.avro.**SchemaLoader**
Load an Avro JSON schema and generate Elasticgit Model python code.

```
python -m elasticgit.tools load-schema avro.json
```

generate_model(*schema*, *field_mapping={}*, *model_renames={}*, *include_header=True*)
Generate Python code for the given Avro schema

Parameters

- **schema** (*dict*) – The Avro schema
- **field_mapping** (*dict*) – An optional mapping of keys to field types that can be used to override the default mapping.

- **model_renames** (*dict*) – An optional mapping of model names that can be used to rename a model.

Parak bool include_header Whether or not to generate the header in the source code, this is useful of you're generating a list of model schema but don't want the header and import statements printed every time.

Returns str

generate_models(*schemas*, *field_mapping={}*, *model_renames={}*)
Generate Python code for the given Avro schemas

Parameters

- **schemas** (*list*) – A list of Avro schema's
- **field_mapping** (*dict*) – An optional mapping of keys to field types that can be used to override the default mapping.

Returns str

run(*schema_files*, *field_mappings=None*, *model_renames=None*)
Inspect an Avro schema file and write the generated Python code to `self.stdout`

Parameters

- **schema_files** (*list*) – The list of file pointers to load.
- **field_mappings** (*list*) – A list of *FieldMapType* types that allow overriding of field mappings.
- **model_renames** (*list*) – A list of *RenameType* types that allow renaming of model names

elasticgit.commands.avro.**deserialize**(*data*, *field_mapping={}*, *module_name=None*)
Deserialize an Avro schema and define it within a module (if specified)

Parameters

- **data** (*dict*) – The Avro schema
- **field_mapping** (*dict*) – Optional mapping to override the default mapping.
- **module_name** (*str*) – The name of the module to put this in. This module is dynamically generated with `imp.new_module()` and only available during code generation for setting the class' `__module__`.

Returns *elasticgit.models.Model*

```
>>> from elasticgit.commands.avro import deserialize
>>> schema = {
... 'name': 'Foo',
... 'type': 'record',
... 'fields': [{
...         'name': 'some_field',
...         'type': 'int',
...     }]
... }
>>> deserialize(schema)
<class 'Foo'>
>>>
```

elasticgit.commands.avro.**serialize**(*model_class*)
Serialize a *elasticgit.models.Model* to an Avro JSON schema

Parameters **model_class** (elasticgit.models.Model) –

Returns str

```
>>> from elasticgit.commands.avro import serialize
>>> from elasticgit.tests.base import TestPerson
>>> json_data = serialize(TestPerson)
>>> import json
>>> schema = json.loads(json_data)
>>> sorted(schema.keys())
[u'fields', u'name', u'namespace', u'type']
>>>
```

1.6.2 Git Model

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

e

B

C

D

E

F

G

I

L

M

P

R

S

T

U

W