TangeloHub Documentation

Release None

Kitware, Inc.

September 21, 2015

Contents

Please visit the TangeloHub homepage or the GitHub repository for more information.

CHAPTER 1

User's Guide

1.1 Managing Data

1.2 Running an Analysis

1.3 Visualization

1.4 Creating an Analysis

For an overview of this process, see the demonstration video.

To create an analysis, start with the following steps:

- Log in to TangeloHub (or first register for an account).
- Navigate to the Data Management tab.
- Click the pencil icon on the left for the collection you want to add an analysis to. If you do not have edit permissions on any collection, contact Kitware to request access with your user name, organization, and desired use case.
- Navigate to the Analysis tab.
- Under "Create new analysis", enter an analysis name and click the "New analysis" button. A new empty analysis will now be saved and selected in the Analysis drop-down menu.
- Click the "Show script" and "Edit" buttons to show an editable script view in the main window.
- Select the language (Python or R) that you desire for your script.

At this point you will have a place to begin writing your Arbor script. You must now decide what input and output types you require, and what formats you expect them to be in. Inputs will be imported into the namespace of the script as variables before the script executes. Outputs are variable names expected to be present after the script runs at which point it will be exported from the script environment.

To add an input, click the plus button next to "Inputs" and enter the name (which must match the script variable name), type, description, and default value. The type represents both the high-level data type (table, tree, string, number, etc.) and format (CSV string, array of dictionaries, R dataframe, etc.) of the input variable in the script. Romanesco, the script execution engine used by Arbor, supports many types and formats described in this documented list. There are two additional properties specific to string inputs. Enter a value in "Comma-separated list of values" if your string input has a fixed set of named values. The interface for such a parameter will be presented as a drop-down list selection. Use "Input for column names" if a string input should represent a column name from another tabular input. In that

case, enter the name of the table input. The input will be presented to the user as a drop-down list of column names from that table.

To add an output, click on the plus button next to "Outputs" and enter the name (which must match the script variable name), type, and description. Similar to inputs, the type must specify the type and format of the variable upon completion of the script.

The final step is to enter the script itself. Start with any required library imports, and enter your script as a simple set of commands as would be entered from the language shell. Use input variables as if they have already been assigned a value, and be sure to create output variables whose names match each of your output names.

Note: After any changes are made, be sure to click the "Save" button to ensure your latest version is saved. Analysis runs will always be performed against the saved version, so be sure to click "Save" before running a new version of your analysis.

1.5 Creating a Workflow

A workflow is an analysis made up of other analyses chained together. Similar to script-based analyses, you must first do the following:

- Log in to TangeloHub (or first register for an account).
- Navigate to the Data Management tab.
- Click the pencil icon on the left for the collection you want to add an analysis to. If you do not have edit permissions on any collection, contact Kitware to request access with your user name, organization, and desired use case.
- Navigate to the Analysis tab.
- Under "Create new analysis", enter an analysis name and click the "New workflow" button (not the "New analysis" button, which is for creating scripts). A new empty workflow will now be saved and selected in the Analysis drop-down menu.
- Click the "Show script" and "Edit" buttons to show an editable workflow view in the main window.

At this point you will have a blank canvas for building your workflow. To add an item to the workflow, select an analysis from the *lower* "Select analysis" drop-down menu. (The upper "Select analysis" drop-down menu will switch over to editing a completely different analysis.) Once a desired analysis is selected, click "Add to workflow," which will add an analysis step box to your workflow diagram. Inputs appear as knobs on the left, while outputs appear as knobs on the right.

To connect analyses in the workflow, drag from an output knob of one analysis to an input knob of another. To expose an analysis input as a workflow input, click on an input knob and an input box will appear. To expose an analysis output as a workflow output, click on an output knob and an output box will appear.

Note: Upon completing your workflow, you must ensure that all input knobs have an incoming connection for the workflow to function properly.

Note: After any changes are made, be sure to click the "Save" button to ensure your latest version is saved. Analysis runs will always be performed against the saved version, so be sure to click "Save" before running a new version of your analysis.

Warning: It is undefined behavior to expose a string input that was defined to be a column from another table, unless that table is also an input. It is undefined because when setting up the analysis, the system does not know what the table's columns will be.

CHAPTER 2

Administrator Documentation

2.1 Prerequisites

2.2 Installation

2.2.1 Vagrant Install

A Vagrant install is an easy way to get everything running in a local virtual machine. This includes an install of MongoDB, Girder, Romanesco, Apache, and the TangeloHub application. To get started, install Vagrant, VirtualBox, and Ansible. It's then a matter of cloning this repository and running `vagrant up`:

```
git clone https://github.com/Kitware/tangelohub.git
cd tangelohub
vagrant up
```

When that completes (it will take some time - get a coffee), visit http://localhost:9080/ to visit the interface.

To see the Girder interface, visit http://localhost:9080/girder.

To create new analyses and save data, login to TangeloHub or Girder with username *girder* and password *girder*.

To log in to your virtual machine, run:

```
vagrant ssh
```

From that environment, you can restart Romanesco:

```
sudo stop romanesco
sudo start romanesco
```

To view the Romanesco log for analysis debugging:

```
sudo cat /var/log/upstart/romanesco.log
```

To restart Girder:

```
sudo stop girder
sudo start girder
```

To enter a local MongoDB shell:

```
mongo
```

The default Vagrant install will also install several R libraries needed for Arbor analyses, which takes a significant amount of time. To turn off this step, edit `devops/ansible/playbook.yml` and set the `arbor` variable to `false`.

2.2.2 Apache Installation

The TangeloHub application requires several components, as well as the main TangeloHub source.

Check out the TangeloHub repository. As a team developer:

```
git clone git@github.com:Kitware/tangelohub.git
```

Or as a contributor:

```
git clone https://github.com/Kitware/tangelohub.git
```

Next, install Girder. Follow the link for install instructions. After following the Girder install, you will also have a MongoDB instance running on your machine. The simplest Girder install consists of a *pip* install. Ensure you have an updated *pip* then install Girder:

```
sudo pip install -U pip
sudo pip install girder
```

Install R. R is needed in order to install and use Romanesco.

We need the Romanesco Girder plugin and web interface. To install them, use the `girder-install` command:

```
sudo girder-install -f web
git clone https://github.com/Kitware/romanesco.git
sudo girder-install -f plugin -s ./romanesco
```

Install an appropriate Girder config file:

```
sudo cp tangelohub/devops/ansible/girder.local.cfg path/to/site-packages/girder/conf/
```

You can find your `path/to/site-packages` with:

```
python
>>> import site
>>> site.getsitepackages()
```

Start Girder:

```
python -m girder &
```

Start a Romanesco worker with:

```
cd romanesco
sudo pip install -r requirements.txt
python -m romanesco &
```

TangeloHub requires npm and Grunt, which should already have been installed as part of the Girder installation:

```
curl -sL https://deb.nodesource.com/setup | sudo bash -
sudo apt-get install nodejs
sudo npm install -g grunt
```

Enter the source folder and build out all the npm dependencies:

```
cd tangelohub
npm install
```

If you modify the TangeloHub source, you can rebuild the app:

```
grunt init
grunt
```

If you are using Apache, use something like the following file in the `sites-available` folder (you can simply replace the *default* file there):

```
Listen 9080

<VirtualHost *:9080>
    DocumentRoot /path/to/tangelohub/app
    ProxyPass /girder http://localhost:9000
    ProxyPassReverse /girder http://localhost:9000
</VirtualHost>
```

You will need the proxy and proxy_html Apache modules:

```
sudo a2enmod proxy
sudo a2enmod proxy_http
```

After restarting Apache (`sudo apache2ctl restart`), visit your Girder web interface at http://localhost:9080/girder to enable the Romanesco plugin from the admin console. A restart of Girder is required to fully enable the Romanesco plugin.

Now you should be able to visit the running TangeloHub instance at http://localhost:9080.

Note: If you hit any `npm` errors in the install process, execute the following and retry the steps:

```
sudo rm -rf ~/tmp
sudo rm -rf ~/.npm
```

Note: If you hit any `pip` errors in the installation of Girder, execute the following and retry:

```
sudo pip install -U pip
```

2.3 Configuration

CHAPTER 3

Developer Documentation

3.1 Contributing

To contribute features and fixes, create a Git branch for your work:

```
git checkout -b my-topic-name
```

Commit your changes to that branch, then push the topic to GitHub:

```
git push -u origin my-topic-name
```

Due to setting upstream with *-u*, after new commits, you may simply:

```
git push
```

Use the GitHub pull request feature to suggest a topic for inclusion. Use GitHub tools to check for a clean Travis CI build and merge to the `master` branch.

3.2 Testing

Testing is performed with PhantomJS and uses Python to drive the process and setup a sandboxed Girder environment. Tests are performed automatically by Travis with resulting test data pushed to a CDash dashboard.

To initialize the testing environment, create a build folder and run CMake:

```
mkdir build
cd build
cmake ..
```

Run the tests with CTest. Use `-V` for verbose output.

```
ctest -V
```

3.3 API Reference

CHAPTER 4

Indices and tables

- genindex
- search