Adding a Custom Python Script to Your Workflow

You can run Python scripts from the TD Workflow using the Python operator (py>:). Create the workflow definition using TD Console or using TD Workflow from the command line.

In the workflow definition, specify a Docker image to use for running the Python script. When the workflow task starts, a new Docker container is created based on the specified docker image. Then, the Python script is executed in the container in an isolated environment.

Prerequisites

- Make sure this feature is enabled for your TD account.
- Basic Knowledge of Treasure Workflow's syntax.
- If you intend to use the CLI, you need to do the following:
  - Download and install the TD Toolbelt and the TD Toolbelt Workflow module. Learn more about TD Workflow Quickstart.
  - Set up Docker on your local machine

Python Examples

You might want to view examples, for basics such as:

- How to call functions
- How to pass parameters to functions
- How to use environment variables
- How to import functions

Add your Python Script to Treasure Workflow

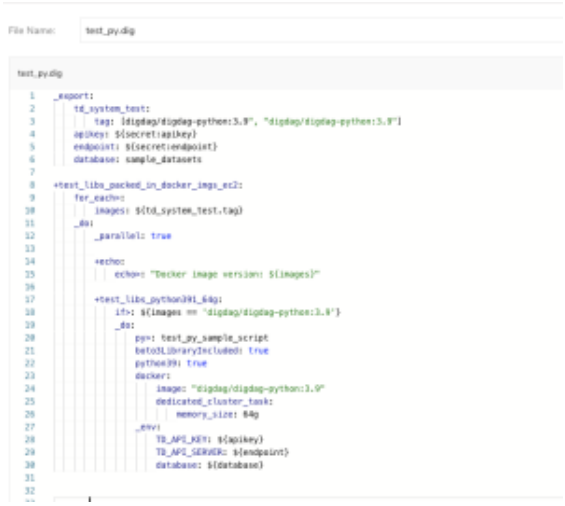
If you have more than a few scripts to add, using the command line method is recommended.

Using TD Console

1. Navigate to Data **Workbench** > **Workflows**.
2. Select the workflow to which you would like to add the Python scripts.
3. Select **Launch Project Editor**.
4. Select **Edit Files.**
5. Select **Add New File.**
6. Type in your dig filename.
7. Add the py> operator and specify a Docker image that you want to use. Your script might look like this sample:

```
+py_custom_code:
  py>: tasks.printMessage
  docker:
    image: "digdag/digdag-python:3.9"
```

8. You can add each script or copy-paste the text of each script into the new script editor window.



9. Select **Save & Commit.**

Using td CLI

You can add a python script to your existing workflow using the command line. New users may need to first create a workflow using the command line.

1. Add a workflow definition .dig file and Python script to the workflow directory.
2. Specify a Docker image that you want to use for the py>: operator in the .dig file.
3. Add syntax similar to the following to your workflow dig file to add the py> operator and specify the Docker image. Your script might look like the following sample:
4. Push the workflow to Treasure Data using td CLI command `td wf push <project_name>`

```
+<wf_task_name>:
  py>: <script_filename>.<function_name>
  docker:
    image: "digdag/digdag-python:3.9"
```

Docker Images

The Python scripts in TD Workflows are managed and run by Treasure Data in isolated Docker containers. Treasure Data provides a number of base Docker images to run in the container. You can pick the appropriate Docker image to run your Python script in, based on the Python version and libraries supported by the image.

Learn more about Docker image versioning for Custom Scripts.

View the below sample using the Python 3.9 Docker image.

```
+task_name:
  py>: <script_filename>.<function_name>
  docker:
    image: "digdag/digdag-python:3.9"
```

Install your own Python Libraries

In addition to the libraries provided by the Docker image, you can install additional 3rd-party libraries using the pip install command within the Python script.

```
import os
import sys
os.system(f"{sys.executable} -m pip install NumPy")
import NumPy
```

Using Docker Images on Your Local Laptop

Docker images are also published in Dockerhub and publicly available for use on your own laptop for evaluation or testing purposes.

Prerequisite: Docker runtime installed.

You can confirm the python version as follows on your laptop:

```
$ docker run -it --rm digdag/digdag-python:3.9 python --version
> Python 3.9.1
```

To run an interactive session, you can run as follows:

```
$ docker run -it --rm digdag/digdag-python:3.9 bash
$ whoami
> td-user
```

Python interactive shell is launched when running digdag/digdag-python:3.9 without arguments:

```
$ docker run -it --rm digdag/digdag-python:3.9
> Python 3.9.1 (default, Jan 12 2021, 16:56:42)
>>
```

You can get a complete list of library versions using pip freeze:

```
$ docker run -it --rm digdag/digdag-python:3.9 pip freeze
> alembic==1.4.3
> attrs==20.3.0
> boto3==1.15.18
> certifi==2020.12.
> …
$ docker run -it --rm digdag/digdag-python:3.9 pip freeze | grep scikit
> scikit-learn==0.24.0
$ docker run -it --rm digdag/digdag-python:3.9 pip freeze | grep pytd
> pytd==1.4.0
```