DTI Guide: Python on C3 AI Suite

This guide is meant to shed more clarity on how python methods are defined and used on the C3 AI Suite. For now, this is primarily a discussion about ActionRuntimes and how they can affect the execution of your Python methods, as well as how to define them.

Python Action Runtimes

Whenever the C3 AI Suite runs a python method, it first loads an associated '`ActionRuntime`' and runs python within that. Essentially, an ActionRuntime is a conda virtual environment.

Traditional Direct Definition and Provisioning of ActionRuntimes

To define a new ActionRuntime, you must provide an action runtime definition in the seed data of your package. An ActionRuntime definition is a .json file with the following format:

```
{
  "id": "py-deepLearning" ,
  "name": "py-deepLearning",
  "connector": "remote",
  "language": "Python",
  "runtime": "CPython",
  "runtimeVersion": "3.6",
  "modules": {
    "conda.numpy": "=1.15.2",
    "conda.cython": "=0.29.12",
    "conda.scikit-learn": "=0.21.2",
    "conda.scipy": "=1.2.1",
    "conda.pip": "=10.0.1",
    "conda.ply": "=3.11",
    "conda.pandas": "=0.22.0",
    "conda.openblas": "=0.3.6",
    "conda.matplotlib": "=3.1.0",
    "conda.dill": "=0.2.8.2",
    "pip.tensorflow": "==1.9.0",
    "pip.slimit": "==0.8.1",
    "pip.jsonpickle": "==1.2",
    "pip.js2py": "==0.66",
    "pip.treeinterpreter": "==0.2.2",
    "pip.eli5": "==0.9.0",
    "pip.nltk": "==3.2.4",
    "conda.libgfortran": "=3.0",
    "pip.textblob": "==0.12.0"
  },
  "repositories": [
    "https://repo.continuum.io/pkgs/main",
    "https://artifacts.c3-e.com/v1/anaconda",
    "conda-forge"
  ]
}
```

Here, we are using the .json format to define a new instance of the ActionRuntime Type which will be created when you provision your package.

Some important fields are:

- **runtimeVersion**: This field specifies the version of python to use. Currently, the latest supported python is 3.6.
- **modules**: This is a dictionary defining packages and their requested versions. Any conda package should be specified like '`conda.<package_name>`'. Then the version is specified with "`=<version>`". Any pip package should be specified like '`pip.<package_name>`', then the version is specified with "`==<version>`". The difference in the version specification is down to the difference between how conda and pip expect version specifications on the command line.
- **repositories**: This field contains a list of conda repository names or URLs.

Once you've defined your ActionRuntime .json file, you need to place it in the seed data of your C3 AI Suite package. It should go in the file `<repository_dir>/<package_dir>/seed/ActionRuntime/<runtime_id>.json` where `<repository_dir>` and `<package_dir>` are the repository and package directories of your package, and `<runtime_id>` is a name matching the 'id' field of the .json file. For instance, in the example above, this would be `py-deeplearning.json`.

Once this definition is in place, re-provision your package and reload your jupyter notebook. You should be able to find and install a python kernel with a name matching the environment name as specified in the .json file.

See the DTI Guide: Data Integration on C3 AI Suite for more about the .json seed data format.

Defining and Managing ActionRuntimes Via IDS

While runtime environments can be created and managed using IDS, we do not currently recommend this approach for the DTI environment.

Checking Your Defined ActionRuntimes

You can check what ActionRuntimes are already defined on your C3 Cluster. The Type `CondaActionRuntime` defines the method '`requirementsFileForLanguage`' which allows you to get a dictionary linking ActionRuntime names with their requirements files.

In the Static Console, run:

```
var res = CondaActionRuntime.requirementsFilesForLanguage('Python');
for (k in res) {
  console.log(k);
}
```

Or via a python connection:

```
res = c3.CondaActionRuntime.requirementsFilesForLanguage('Python')
for k in res:
  print(k)
```

Additionally, the C3DTI provide the '`provision-action-runtime.py`' helper script. This is available in the **c3-helper-scripts github repository**. Once you download this repository, the `` `provision-action-runtime.py` `` script can used to list action runtimes as follows:

```
python provision-action-runtime.py --server <vanity_url> --tenant <tenant> --tag <tag> --list
```

Inspect Installed Packages for an ActionRuntime

We can also inspect the installed packages for a given action runtime by looking at the '`value`' of the appropriate key. For example, in JavaScript:

```
var res = CondaActionRuntime.requirementsFilesForLanguage('Python')
console.log(res['py-mlutils_1_0_0'])
```

Which gives us:

```
#conda env create --file requirements.yaml
name: py-mlutils_1_0_0
channels:
- https://repo.continuum.io/pkgs/main
dependencies:
- dill=0.2.8.2
- numpy=1.15.2
- pandas=0.23.4
- python-dateutil
- python=3.6
```

Or in python:

```
res = c3.CondaActionRuntime.requirementsFilesForLanguage('Python')
print(res['py-mlutils_1_0_0'])
```

Specifying ActionRuntime Method

When defining a new python method on a Type, we specify the ActionRuntime environment with the `py` annotation. For example, consider the method 'getFileSourceSpec' in the IDXFile Type in the **mnistExample**:

```
@py(env='idxfile')
getFileSourceSpecPreprocess: member function(serializedPreprocessor: string, preprocessFuncName: string,
enableLocalClientStorage: boolean = true): !FileSourceSpec py server
```

Here, we see the 'py' annotation being used with the parameter 'env'. This parameter contains the string 'idxfile'. This means the 'getFileSourceSpecPreprocess' function will be run with the 'py-idxfile' ActionRuntime environment.

Inline Python Methods

Methods can be implemented as 'inline' (see 'Inline Methods' **here**). In the context of Python methods, this means if you're currently executing the function from a Python context, the method will be executed in your current python context.

This means if you define an inline python method which requires specific packages not normally available, the method will fail if your context doesn't have the necessary packages.

Additional Resources

- Developer Documentation
  - https://developer.c3.ai/docs/7.12.25/guide/guide-c3aisuite-basic/ds-python-apis
- Jupyter notebooks
  - https://<vanity_url>/jupyter/notebooks/tutorials/TutorialIntroJupyterNotebook.ipynb
- C3.ai Academy Videos
  - **Data Science Module**: 'Python APIs and Runtimes'
    - Python Runtimes