Impala Style Guide

All languages:

- All source files should include the standard Apache license header.
- All lines should be limited to 90 characters in width
- Remove trailing whitespace. To automate this, install pre-commit.sh into .git/hooks/pre-commit and make it executable.

C++

See the C++ Style Guide

Java

- Currently undocumented - new code should follow existing style conventions. Note that you should use clang-format to automatically reformat Java code, as described in the C++ Style Guide

Thrift

We don't have a formal style guide, but we do adhere to a few conventions:

- Structs
  - Struct names begin with a "T", and use a capital letter for each new word, with no underscores.
  - All fields should be declared as either optional or required
  - Don't use semicolons after struct member declarations; they're optional, so we optimize for typing efficiency.
- Functions
  - Function names start with a capital letter and have a capital letter for each new word, with no underscores.
  - Each function should take exactly one parameter, named FunctionNameParameter, and should return either void or FunctionNameResponse. This convention allows incremental updates.
- Parameters
  - Parameter names use camel case (with no underscores).
- Services
  - Service names should end in the word "Service"

Here's an example:

```
service MetadataRepositoryService {
  TRegisterResponse Register(1: TRegisterRequest request);
}

struct TRegisterRequest {
  1: required list<string> applicationIds
  2: required TNetworkAddress address
}

struct TRegisterResponse {
  1: required set<TMetadatum> metadata
}
```

Python

When editing existing code, you should follow existing code style in the file you are editing. In general, our code tries to follow PEP8 style with some exceptions:

- We wrap lines at 90 characters (instead of 80)
- We use 2 spaces indent (instead of 4)

We automatically run flake8-diff on jobs pre-commit to report style violations on lines that patches modify. It is configured using setup.cfg in the Impala repository.

You can run flake8 in several ways. Note that a lot of existing Python code in Impala doesn't meet our current style standards. We fix such issues incrementally when we modify the code.

```
# Works in an Impala development environment. Lists all violations for a file or subdirectory.
impala-flake8 <path to file>

# Show newly introduced violations for your current HEAD commit in git.
# Requires you to install flake8-diff separately.
flake8-diff HEAD^ HEAD

# Generate the same violations as the Impala jenkins bot for your current HEAD commit in git.
./bin/jenkins/critique-gerrit-review.py --dryrun
```

The following tools have also been found useful by developers to scan and/or automatically fix formatting issues:

- pycodestyle (formerly known as pep8) - A checker for PEP8 conformance
- autopep8 - A tool that automatically formats code to conform with PEP8
- vim-syntastic - A plugin for vim that can highlight style violations

All tools above can be used with the same configuration file. As a starting point, consider using this one:

```
[pep8]
# E101 - Reindent all lines.
# E111 - indentation is not a multiple of four
# E114 - indentation is not a multiple of four (comment)
# E251 - Remove whitespace around parameter '=' sign.
# E301 - Add missing blank line.
ignore = E101,E111,E114,E251,E301
max-line-length = 90
```