Kubernetes Unleashed

Herve Meftah
admin@crunchydevops.com

CrunchyDevOps.com— March 31, 2021

Introduction

Info: Know which version of Kubernetes you are using.

command
```
kubectl get nodes
```

1 Install Prow

Prow is a Kubernetes based CI/CD system. Jobs can be triggered by various types of events and report their status to many different services. In addition to job execution, Prow provides GitHub automation in the form of policy enforcement, chat-ops via `/foo` style commands, and automatic PR merging.

1.1 Whitespace

In Python, whitespace is syntactically significant. Python programmers are especially sensitive to the effects of whitespace on code clarity. Follow these guidelines related to whitespace:

- Use spaces instead of tabs for indentation.
- Use four spaces for each level of syntactically significant indenting.
- Lines should be 79 characters in length or less.
- Continuations of long expressions onto additional lines should be indented by four extra spaces from their normal indentation level.
- In a file, functions and classes should be separated by two blank lines.
- In a class, methods should be separated by one blank line.
- In a dictionary, put no whitespace between each key and colon, and put a single space before the corresponding value if it fits on the same line.
- Put one—and only one—space before and after the = operator in a variable assignment.
- For type annotations, ensure that there is no separation between the variable name and the colon, and use a space before the type information.

1.2 Naming

PEP 8 suggests unique styles of naming for different parts in the language. These conventions make it easy to distinguish which type corresponds to each name when reading code. Follow these guidelines related to naming:

- Functions, variables, and attributes should be in lowercase_ underscore format.
- Protected instance attributes should be in _leading_underscore format.
- Private instance attributes should be in __double_leading_ underscore format.
- Classes (including exceptions) should be in CapitalizedWord format.
- Module-level constants should be in ALL_CAPS format.
- Instance methods in classes should use self, which refers to the object, as the name of the first parameter.
- Class methods should use cls, which refers to the class, as the name of the first parameter.

1.3 Expressions and Statement

The *Zen of Python* states: “There should be one—and preferably only one—obvious way to do it.” PEP 8 attempts to codify this style in its guidance for expressions and statements:

- Use inline negation (if a is not b) instead of negation of positive expressions (if not a is b).
- Don’t check for empty containers or sequences (like [] or ”) by comparing the length to zero (`if len(somelist) == 0`). Use `if not somelist` and assume that empty values will implicitly evaluate to False.
- The same thing goes for non-empty containers or sequences (like `[1] or 'hi'`). The statement `if somelist` is implicitly True for nonempty values.
- Avoid single-line if statements, for and while loops, and except compound statements. Spread these over multiple lines for clarity.
- If you can’t fit an expression on one line, surround it with parentheses and add line breaks and indentation to make it easier to read.
- Prefer surrounding multiline expressions with parentheses over using the line continuation character.

1.4 Imports

PEP 8 suggests some guidelines for how to import modules and use them in your code:

- Always put `import` statements (including `from x import y`) at the top of a file.
- Always use absolute names for modules when importing them, not names relative to the current module’s own path. For example, to import the `foo` module from within the `bar` package, you should use from `bar import foo`, not just `import foo`.
- If you must do relative imports, use the explicit syntax `from . import foo`
- Imports should be in sections in the following order: standard library modules, third-party modules, your own modules. Each subsection should have imports in alphabetical order.

Note: The Pylint tool (`https://www.pylint.org`) is a popular static analyzer for Python source code. Pylint provides automated enforcement of the PEP 8 style guide and detects many other types of common errors in Python programs. Many IDEs and editors also include linting tools or support similar plug-ins.

1.5 Things to Remember

- Always follow the Python Enhancement Proposal #8 (PEP 8) style guide when writing Python code.
- Sharing a common style with the larger Python community facilitates collaboration with others.
- Using a consistent style makes it easier to modify your own code later.

2 Know the differences between bytes and str

In Python, there are two types that represent sequences of character data: bytes and str. Instances of bytes contain raw, unsigned 8-bit values (often displayed in the ASCII encoding):