# Hands-on Lab: Static Code Analysis

Skills Network

Estimated time needed: **30** minutes

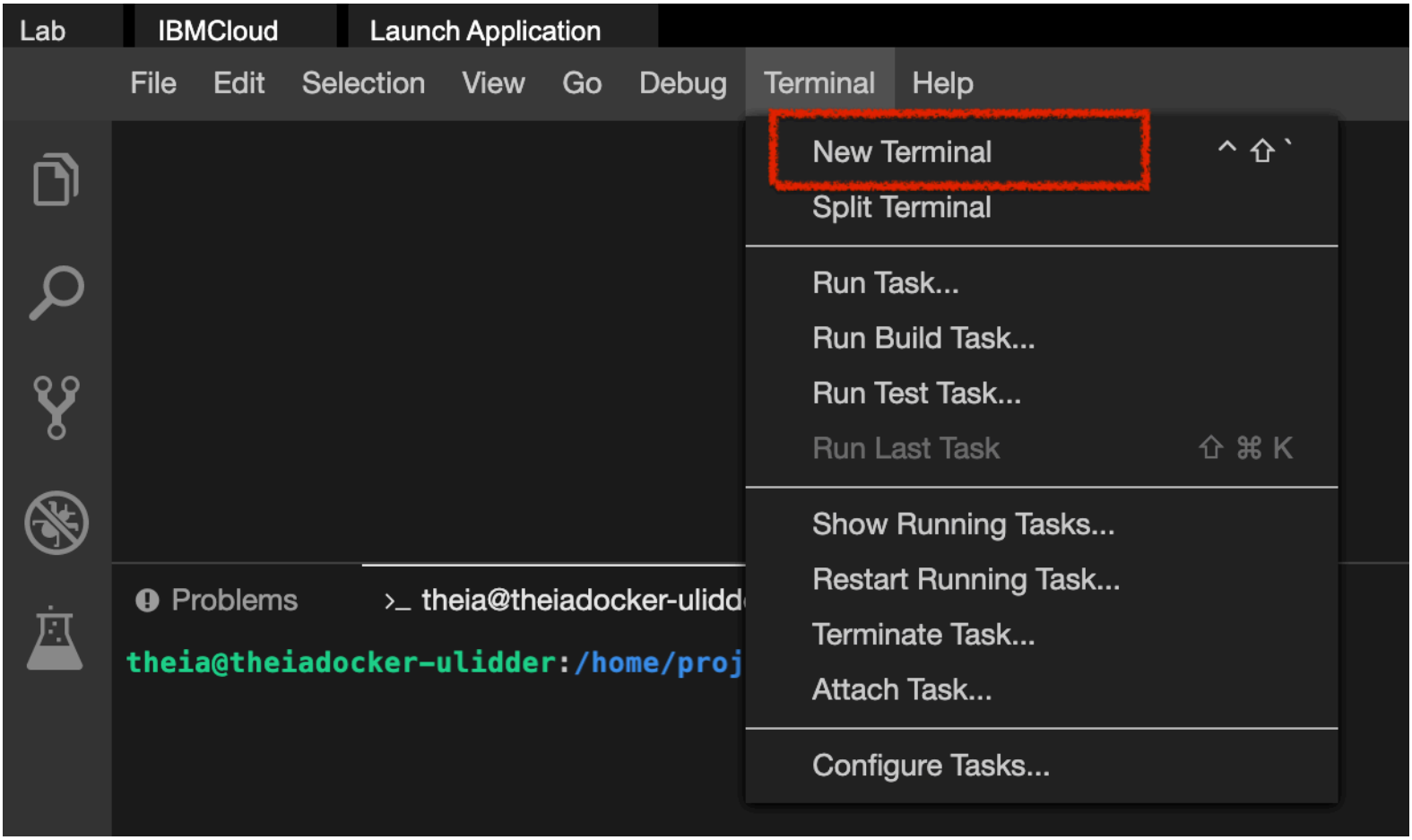
## Objectives

After completing this lab you will be able to:

- Install pylint package
- Run Static Code Analysis on a python program
- Check the compliance score of a python program.
- Fix common mistakes and improve the compliance score.

# Install the pylint package

1. Open a new terminal.




```
pip3 install pylint==2.11.1
```

3. This should install the pylint package.

```
theia@thei:/home/project$ pip3 install pylint
/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:15: CryptographyDeprecationWarning
m_bytes instead
  from cryptography.utils import int_from_bytes
/usr/lib/python3/dist-packages/secretstorage/util.py:19: CryptographyDeprecationWarning: in
tes instead
  from cryptography.utils import int_from_bytes
Defaulting to user installation because normal site-packages is not writeable
Collecting pylint
  Downloading pylint-2.9.3-py3-none-any.whl (372 kB)
     |████████████████████████████████| 372 kB 17.4 MB/s
Requirement already satisfied: mccabe<0.7,>=0.6 in /usr/local/lib/python3.6/dist-packages (
Collecting isort<6,>=4.2.5
  Downloading isort-5.9.2-py3-none-any.whl (105 kB)
     |████████████████████████████████| 105 kB 43.2 MB/s
Collecting astroid<2.7,>=2.6.2
  Downloading astroid-2.6.2-py3-none-any.whl (228 kB)
     |████████████████████████████████| 228 kB 39.7 MB/s
Requirement already satisfied: toml>=0.7.1 in /usr/local/lib/python3.6/dist-packages (from
Collecting wrapt<1.13,>=1.11
  Downloading wrapt-1.12.1.tar.gz (27 kB)
Requirement already satisfied: typing-extensions>=3.7.4 in /home/theia/.local/lib/python3.6
ylint) (3.7.4.3)
Collecting typed-ast<1.5,>=1.4.0
  Downloading typed_ast-1.4.3-cp36-cp36m-manylinux1_x86_64.whl (743 kB)
     |████████████████████████████████| 743 kB 36.6 MB/s
Requirement already satisfied: lazy-object-proxy>=1.4.0 in /home/theia/.local/lib/python3.6
ylint) (1.4.3)
Building wheels for collected packages: wrapt
  Building wheel for wrapt (setup.py) ... done
  Created wheel for wrapt: filename=wrapt-1.12.1-cp36-cp36m-linux_x86_64.whl size=69407 sha
cd14154e5b6a0171266f60f85
  Stored in directory: /home/theia/.cache/pip/wheels/32/42/7f/23cae9ff6ef66798d00dc5d659088
Successfully built wrapt
Installing collected packages: wrapt, typed-ast, isort, astroid, pylint
Successfully installed astroid-2.6.2 isort-5.9.2 pylint-2.9.3 typed-ast-1.4.3 wrapt-1.12.1
```

# Create a sample python file for static code analysis

Create a new file named **sample1.py**

Copy and paste the below code into **sample1.py**

```
# Define a function named 'add' that takes two arguments, 'number1' and 'number2'.
def add(number1, number2):
    # The function returns the sum of 'number1' and 'number2'.
    return number1 + number2
# Initialize the variable 'num1' with the value 4.
num1 = 4
# Initialize the variable 'num2' with the value 5.
num2 = 5
# Call the 'add' function with 'num1' and 'num2' as arguments and store the result in 'total'.
total = add(num1, num2)
# Print the result of adding 'num1' and 'num2' using the 'format' method to insert the values into the string.
print("The sum of {} and {} is {}".format(num1, num2, total))
```

Save the file **sample1.py**

# Run pylint

- Open a terminal

- Run the below command

```
pylint sample1.py
```

- Pylint goes through every line of code and gives you a list all the non-compliant lines.
- Pylint gives you a compliance score (10 being maximum).

# Correct the mistakes identified by pylint.

- Based on the report given by pylint changes were made to this code to address the following issues.
  - Exactly one space required after comma
  - Exactly one space required around assignment
- Create a new file named **sample2.py**
- Copy and paste the below code into **sample2.py**

```
# Define a function named 'add' that takes two arguments, 'number1' and 'number2'.
# The purpose of this function is to add the two numbers and return the result.
def add(number1, number2):
    # Return the sum of 'number1' and 'number2'.
    # This line computes the addition of the two input numbers and outputs the result.
    return number1 + number2
# Initialize the constant variable 'NUM1' with the value 4.
# Constants are usually written in uppercase letters to indicate that they should not be changed.
NUM1 = 4
# Initialize the variable 'num2' with the value 5.
# This variable will be used as the second input to the 'add' function.
num2 = 5
# Call the 'add' function with 'NUM1' and 'num2' as arguments.
# The result of this addition operation is stored in the variable 'total'.
total = add(NUM1, num2)
# Print a formatted string that displays the sum of 'NUM1' and 'num2'.
# The 'format' method is used to insert the values of 'NUM1', 'num2', and 'total' into the string.
print("The sum of {} and {} is {}".format(NUM1, num2, total))
```

Save the file **sample2.py**

# Run pylint

- Open a terminal
- Run the below command

```
pylint sample2.py
```

- This will give you the compliance score.
- This time you should see the score improve.

# Your task

Improve the score in sample2.py to a perfect 10 by correcting all the issues pointed out by pylint. If cant figure out how to solve some issues it is helpful to google the pylint message.

# Congratulations!

You now know how to perform static code analysis.

## Author(s)

Ramesh Sannareddy

## Other Contributors

Rav Ahuja

© IBM Corporation. All rights reserved.