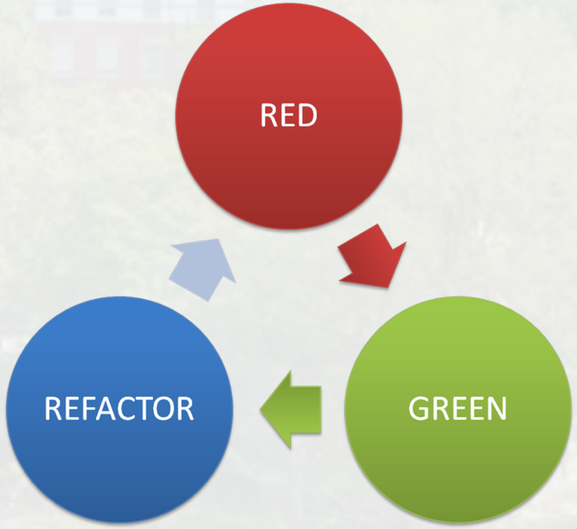
NYU COURANT INSTITUTE OF MATHEMATICAL SCIENCES

Behavioral Driven Development



Fall 2020, CSCI-GA 2820, Graduate Division, Computer Science

Instructor:

John J Rofrano

Senior Technical Staff Member | DevOps Champion
IBM T.J. Watson Research Center
rofrano@cs.nyu.edu @JohnRofrano

Create the Vagrant VM

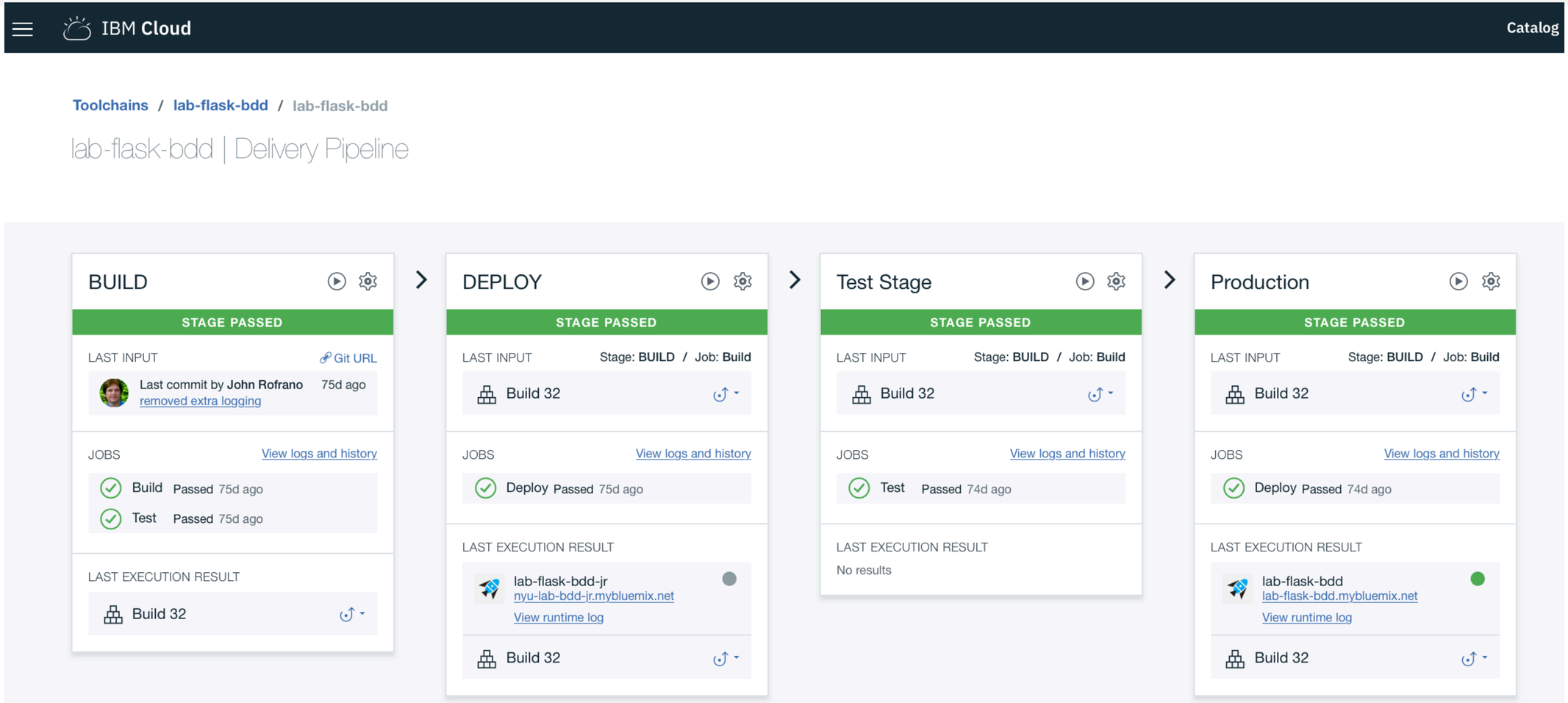
- Clone the project and start Vagrant provisioning the VM

```
$ git clone https://github.com/nyu-devops/lab-flask-bdd.git

$ cd lab-flask-bdd

$ vagrant up
```

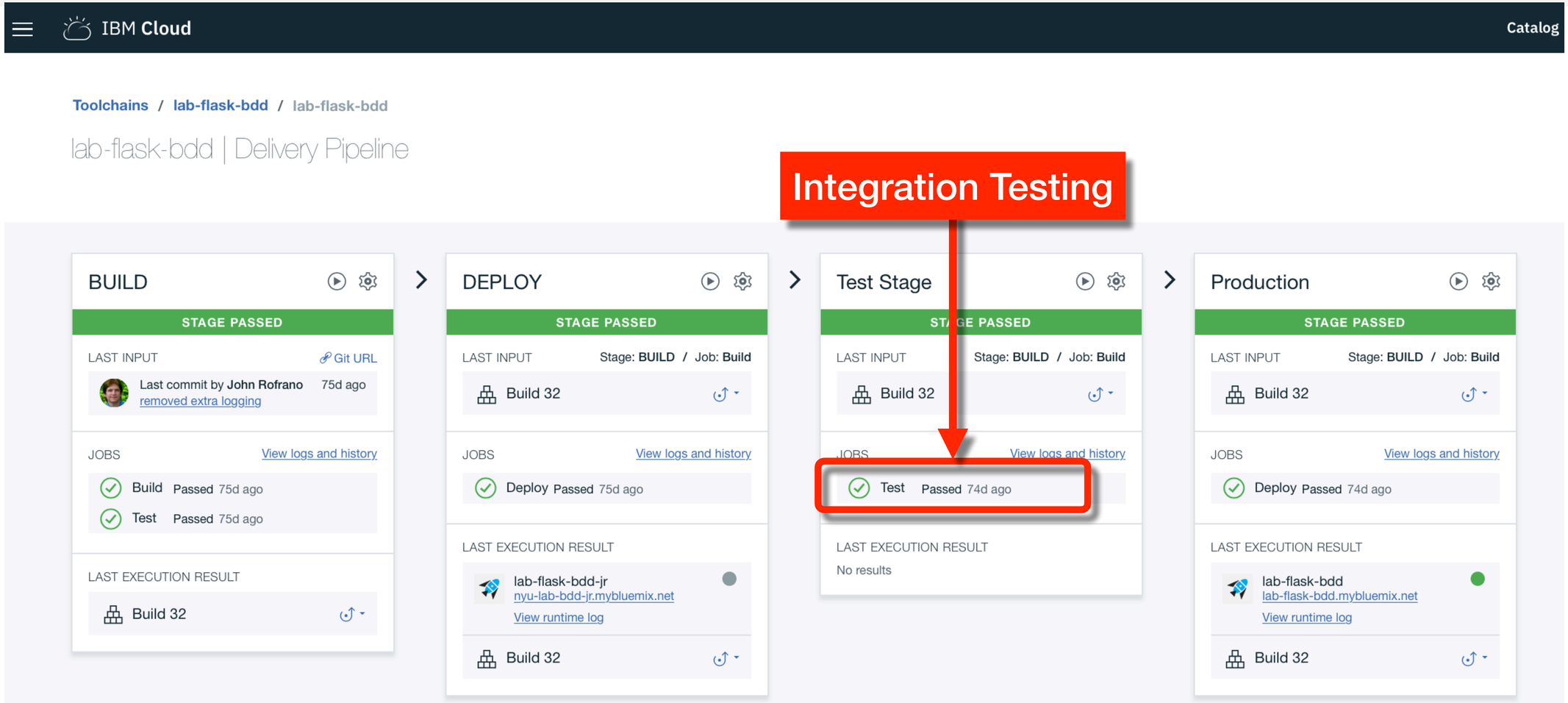
What Will You Learn?

- What is Behavioral Driven Development
- How Behavior-Driven Development can drive customer expectations
- How you can test your Acceptance Criteria and thus the definition of "Done"

What is the Goal?



What is the Goal?

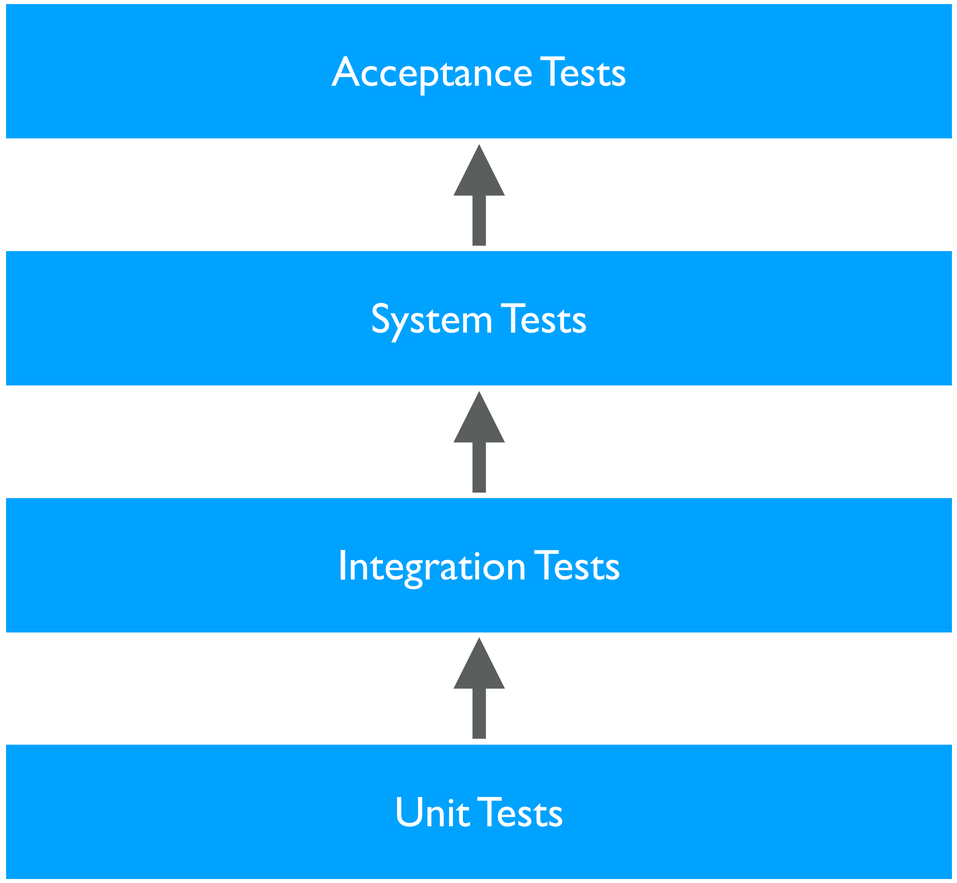


“If it's worth building, it's worth testing.
If it's not worth testing, why are you wasting your time working on it?”

–*agiledata.org*

Software Testing Levels

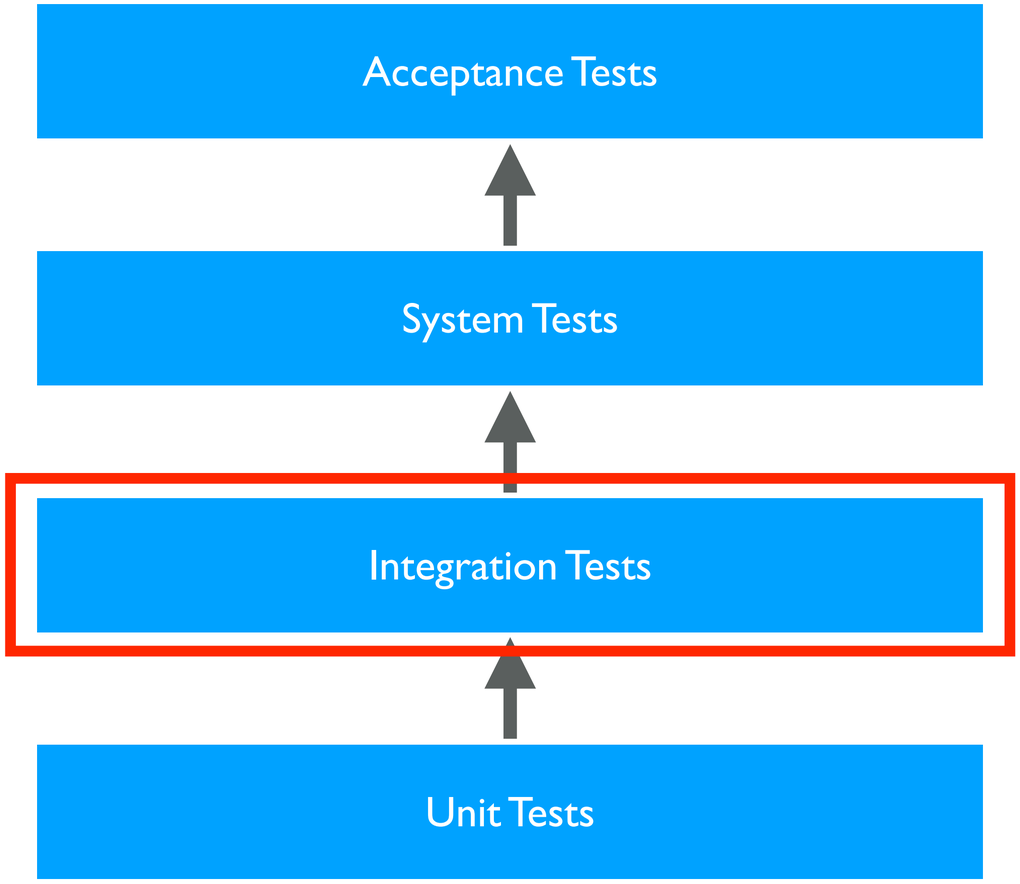
Acceptance Testing	A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.
System Testing	A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.
Integration Testing	A level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.
Unit Testing	A level of the software testing process where individual units/ components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.



http://softwaretestingfundamentals.com/software-testing-levels/

Software Testing Levels

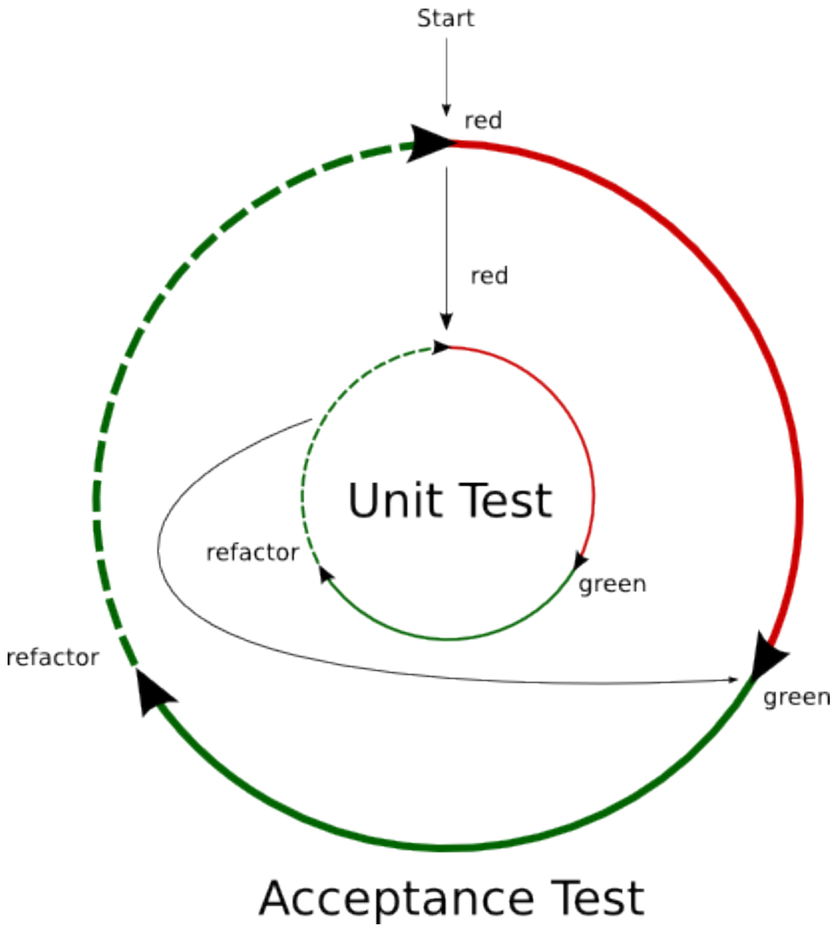
Acceptance Testing	A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.
System Testing	A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.
Integration Testing	A level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.
Unit Testing	A level of the software testing process where individual units/ components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.



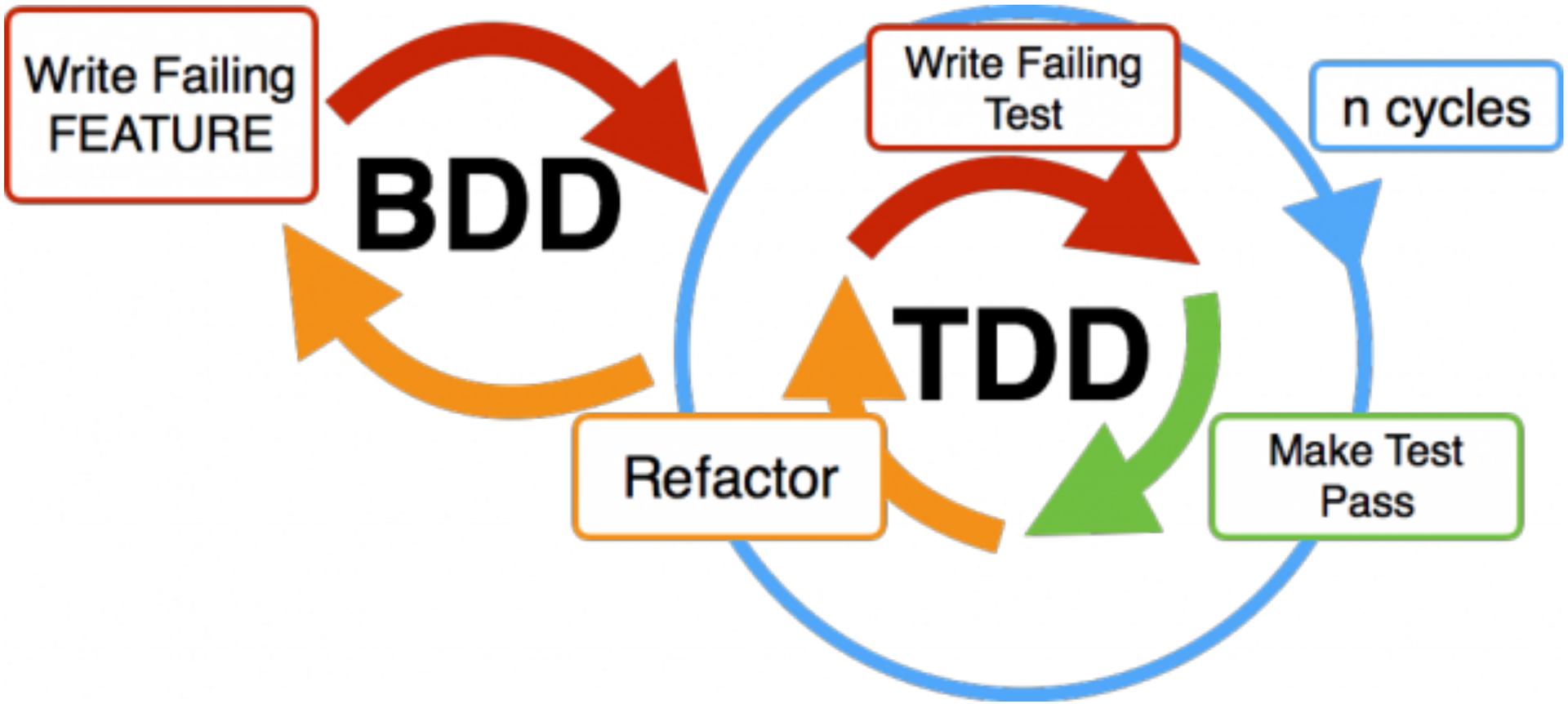
http://softwaretestingfundamentals.com/software-testing-levels/

What is BDD & TDD

- Behavior-Driven Development (BDD)
  - Describes the behavior of the system from the outside in
  - Used for Integration / Acceptance Testing
- Test Driven Development (TDD)
  - Tests the functions of the system from the inside out
  - Used for unit testing



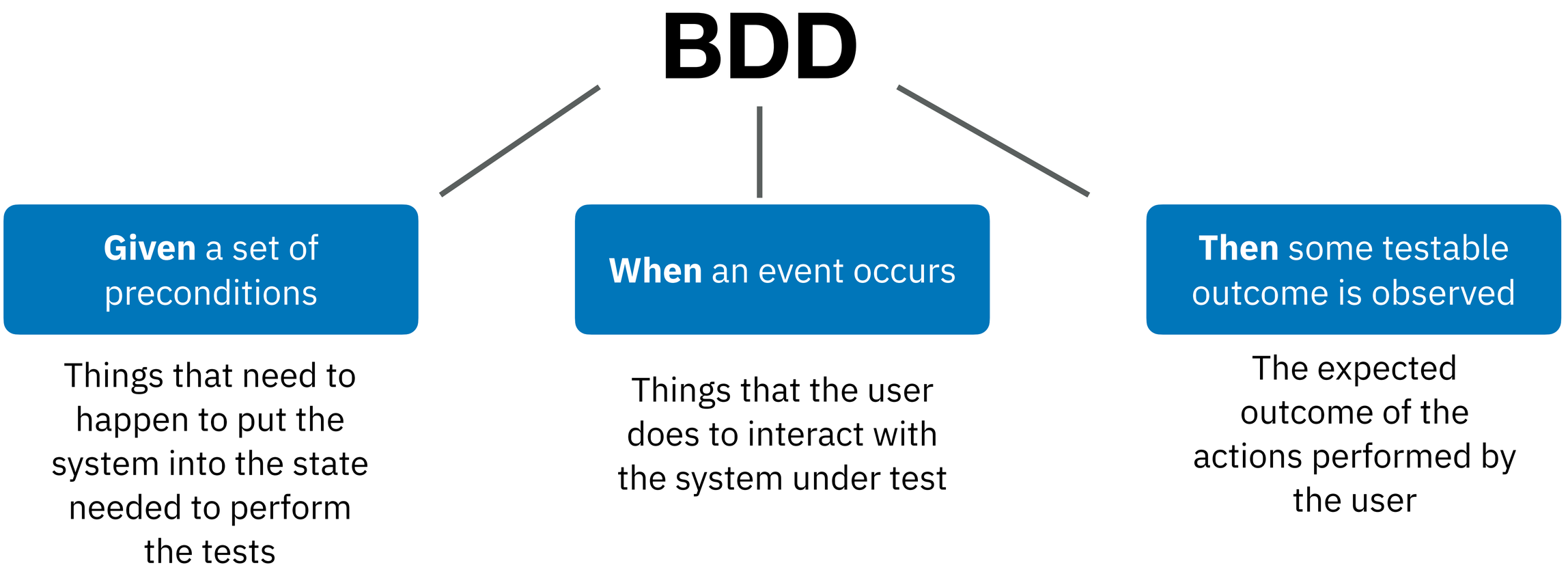
BDD & TDD


Write Failing FEATURE
BDD
Write Failing Test
n cycles
TDD
Refactor
Make Test Pass

“BDD is building the *right thing*, and TDD is building the *thing right*.”

–Anonymous

Behavior-Driven Development



BDD tactics:

- Apply the "Five Why's" principle to each proposed User Story, so that its purpose is clearly related to business outcomes
- Thinking "from the outside in", in other words implement only those behaviors which contribute most directly to these business outcomes, so as to minimize waste
- Describe behaviors in a single notation which is directly accessible to domain experts, testers and developers, so as to improve communication
- Apply these techniques all the way down to the lowest levels of abstraction of the software, paying particular attention to the distribution of behavior, so that evolution remains cheap

The 5 Why's

- The "5 why's" are a way of continuously asking "why?" over an over until you get to the core of why something happened.

?????

Example of The 5 Why's

Example of The 5 Why's

Q. "Why did the system fail in production?"

Example of The 5 Why's

Q. "Why did the system fail in production?"
A. "we found an error condition that wasn't caught"

Example of The 5 Why's

Q. "Why did the system fail in production?"
A. "we found an error condition that wasn't caught"

Q. "Why didn't we catch this condition?"

Example of The 5 Why's

Q. "Why did the system fail in production?"
A. "we found an error condition that wasn't caught"

Q. "Why didn't we catch this condition?"
A. "Because we didn't write a test case for it"

Example of The 5 Why's

Q. "Why did the system fail in production?"
A. "we found an error condition that wasn't caught"

Q. "Why didn't we catch this condition?"
A. "Because we didn't write a test case for it"

Q. "Why didn't we write a test case"

Example of The 5 Why's

Q. "Why did the system fail in production?"
A. "we found an error condition that wasn't caught"

Q. "Why didn't we catch this condition?"
A. "Because we didn't write a test case for it"

Q. "Why didn't we write a test case"
A. "Because the developer wasn't trained in TDD"

Example of The 5 Why's

Q. "Why did the system fail in production?"
A. "we found an error condition that wasn't caught"

Q. "Why didn't we catch this condition?"
A. "Because we didn't write a test case for it"

Q. "Why didn't we write a test case"
A. "Because the developer wasn't trained in TDD"

Q. "Why wasn't the developer trained in TDD?

Example of The 5 Why's

Q. "Why did the system fail in production?"
A. "we found an error condition that wasn't caught"

Q. "Why didn't we catch this condition?"
A. "Because we didn't write a test case for it"

Q. "Why didn't we write a test case"
A. "Because the developer wasn't trained in TDD"

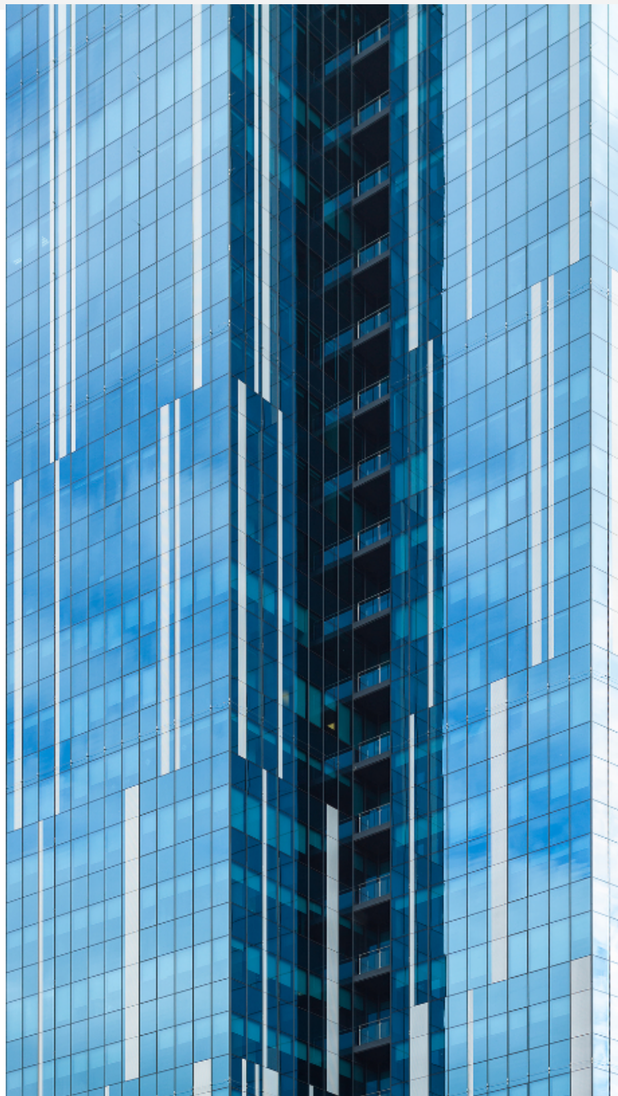
Q. "Why wasn't the developer trained in TDD?
A. "Because we cut the training budget"

Conclusion of The 5 Why's

- Why did the system fail in production?
- Because we cut the training budget and didn't properly train our new developers.

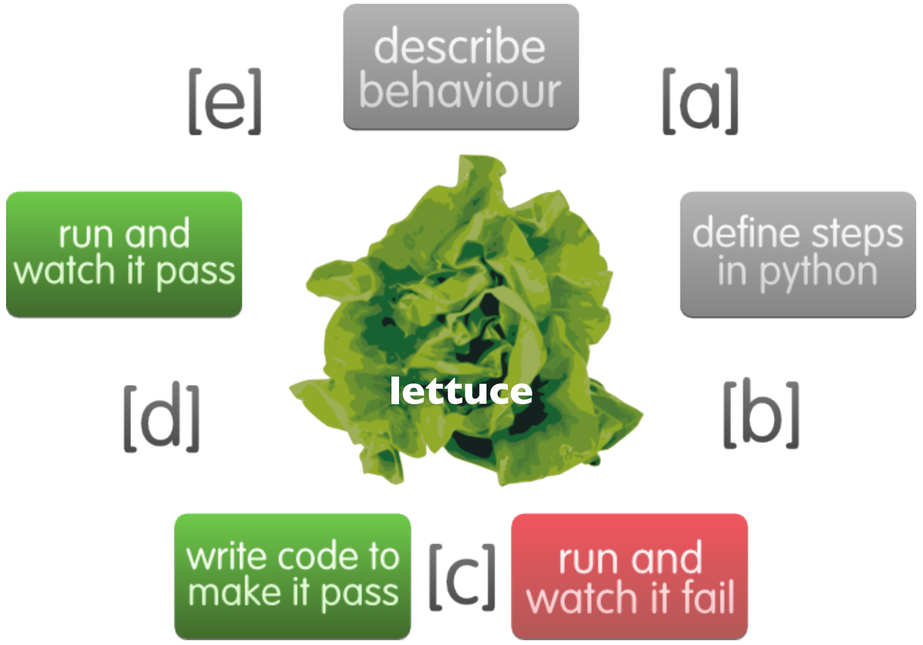
Expected Benefits of BDD

- BDD offers more precise guidance on organizing the conversation between developers, testers and domain experts
- Notations originating in the BDD approach, in particular the given-when-then canvas, are closer to everyday language and have a shallower learning curve compared to TDD tools
- Tools targeting a BDD approach generally afford the automatic generation of technical and end user documentation from BDD "specifications"



BDD Tools

Cucumber



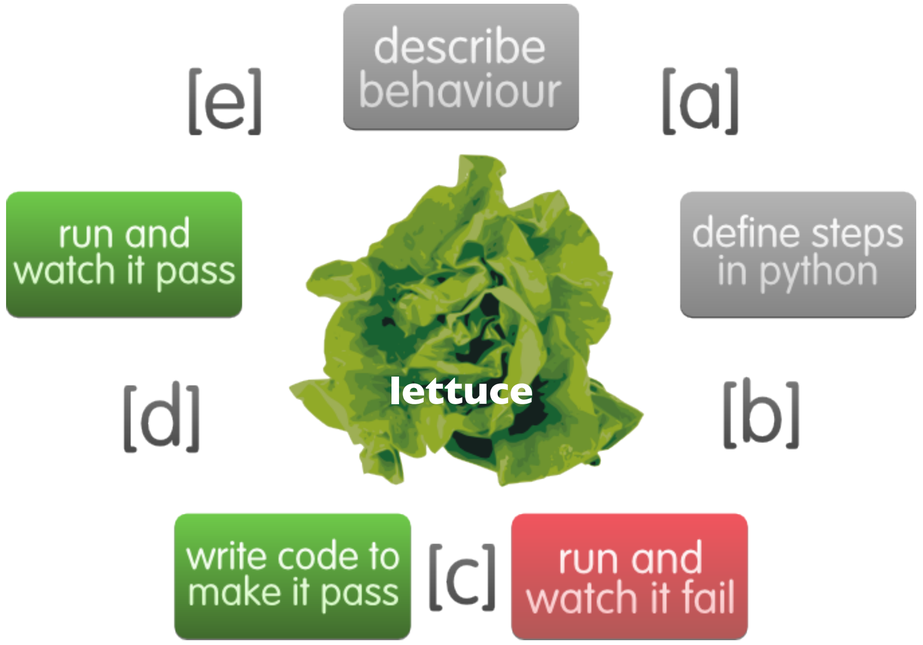
Concordion

Jasmine

jbehave

BDD Tools

Cucumber



Concordion

Jasmine

jbehave

BDD Workflow

BDD Workflow

- First, the developers, testers and business folks explore the problem domain, and collaborate to produce concrete examples that describe the behavior they want.

BDD Workflow

- First, the developers, testers and business folks explore the problem domain, and collaborate to produce concrete examples that describe the behavior they want.
- Next, the team use Behave to run those examples as automated acceptance tests.

BDD Workflow

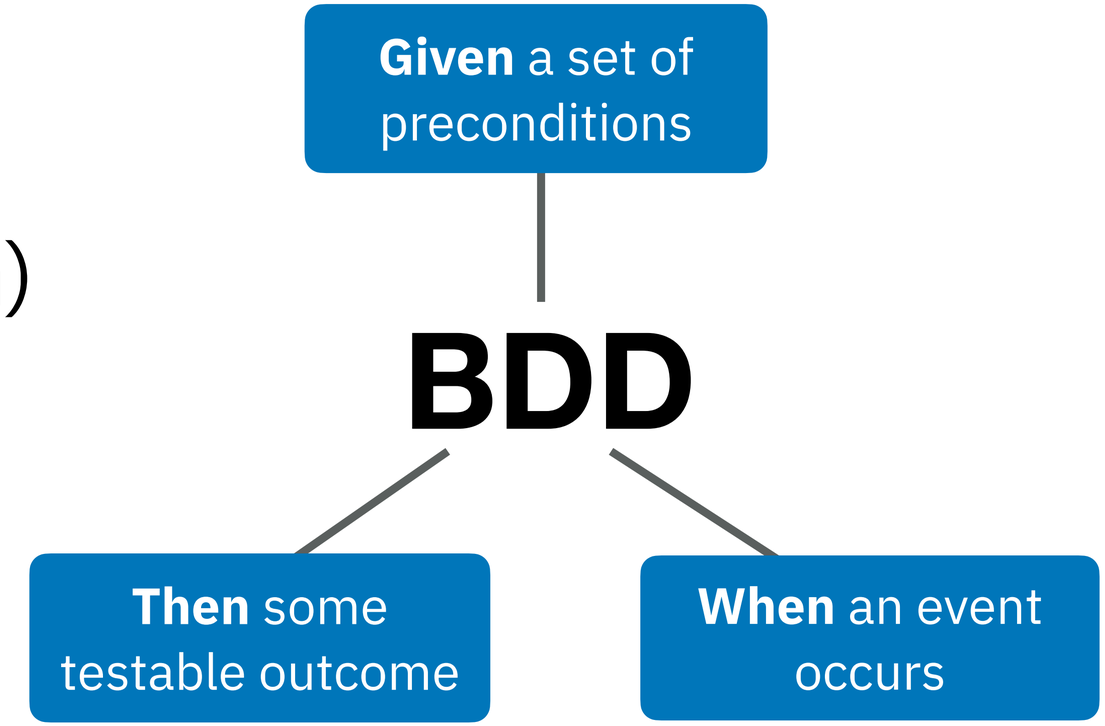
- First, the developers, testers and business folks explore the problem domain, and collaborate to produce concrete examples that describe the behavior they want.
- Next, the team use Behave to run those examples as automated acceptance tests.
- As the team work on the solution, Behave tells you which examples are implemented and working, and warns you about the ones that aren’t.

BDD Workflow

- First, the developers, testers and business folks explore the problem domain, and collaborate to produce concrete examples that describe the behavior they want.
- Next, the team use Behave to run those examples as automated acceptance tests.
- As the team work on the solution, Behave tells you which examples are implemented and working, and warns you about the ones that aren’t.
- Before you know it, you have one document that’s both the specification and the tests for your software.

Basic Template

- Given (some context)
- When (something happens)
- Then (some behavioral validation)



BDD Uses Gherkin Syntax

- **Given** — the purpose of givens is to put the system in a known state before the user (or external system) starts interacting with the system (in the When steps)
- **When** — each of these steps should describe the key action the user (or external system) performs
- **Then** — is used to observe outcomes. The observations should be related to the business value/benefit in your feature description
- **And** — is used for continuations. Given this And that... Then this And that... etc.

Retail BDD Example

```
Feature: Returns go to stock

As a store owner
In order to keep track of stock
I want to add items back to stock when they're returned.

Scenario 1: Refunded items should be returned to stock
  Given that a customer previously bought a black sweater from me
  And I have three black sweaters in stock.
  When he returns the black sweater for a refund
  Then I should have four black sweaters in stock.

Scenario 2: Replaced items should be returned to stock
  Given that a customer previously bought a blue garment from me
  And I have two blue garments in stock
  And three black garments in stock.
  When he returns the blue garment for a replacement in black
  Then I should have three blue garments in stock
  And two black garments in stock.
```

How it works

- Behave looks for a folder named `features` with files that have an extension of `.feature`
- It then looks for a folder under that called `steps` that contains the Python code to parse the Gherkin sentences in the feature files.

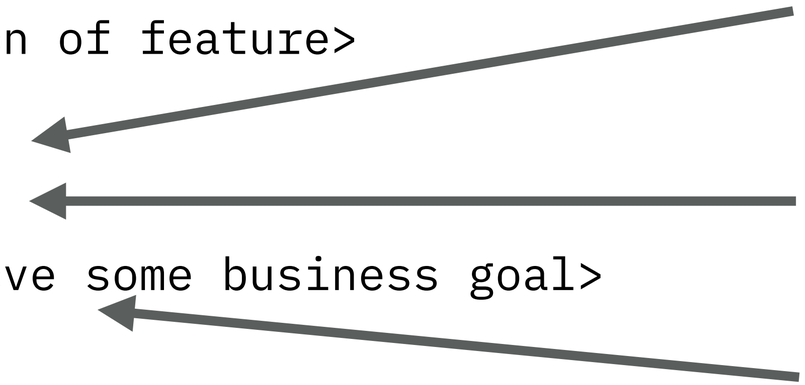
```
.
└── features
    ├── some.feature
    └── steps
        └── steps.py
```

Note: There is no relationship between feature files and step files. Behave loads all of the steps regardless of how many files they are contained in

Feature Definition

feature file:

```
Feature: <description of feature>

As a <steakholder>
I want <something>
So that <I can achieve some business goal>
```



Who will benefit from this feature; who wants is?

What does the feature do?

What business value will the stakeholder get out of this feature?

Feature Definition

feature file:

```
Feature: <description of feature>

As a <steakholder>
I want <something>
So that <I can achieve some business goal>
```

Who will benefit from this feature; who wants is?

What does the feature do?

What business value will the stakeholder get out of this feature?

Ask the "Five Why's" here

Step Definition

feature file:

```
Feature: <description of feature>

As a <steakholder>
I want <something>
So that <I can achieve some business goal>


Scenario: test something
  Given some known state
    And some other known state
   When some action is taken
   Then some outcome is observed
    And some other outcome is not observed.
```

Step Definition

feature file:

```
Feature: <description of feature>

As a <steakholder>
I want <something>
So that <I can achieve some business goal>


Scenario: test something
  Given some known state
    And some other known state
   When some action is taken
   Then some outcome is observed
    And some other outcome is not observed.
```

step file:

```
from behave import *

@given('some known state')
def step_impl(context):
    set_up(some, state)

@when('some action is taken')
def step_impl(context):
    perform(some, action)
```

Pet Store Example

pets.feature

```
Feature: The pet store service back-end
    As a Pet Store Owner
    I need a RESTful catalog service
    So that I can keep track of all my pets

Scenario: The server is running
    When I visit the "Home Page"
    Then I should see "Pet Demo RESTful Service"
    And I should not see "404 Not Found"
```

pet_steps.py

```
from behave import *

@when('I visit the "Home Page"')
def step_impl(context):
    """ Make a call to the base URL """
    context.driver.get(context.base_url)

@then('I should see "{message}"')
def step_impl(context, message):
    """ Check the document title for a message
"""
    assert message in context.driver.title

@then('I should not see "{message}"')
def step_impl(context, message):
    assert message not in context.resp.text
```

Pet Store Example

pets.feature

```
Feature: The pet store service back-end
    As a Pet Store Owner
    I need a RESTful catalog service
    So that I can keep track of all my pets

Scenario: The server is running
    When I visit the "Home Page"
    Then I should see "Pet Demo RESTful Service"
    And I should not see "404 Not Found"
```

pet_steps.py

```
from behave import *

@when('I visit the "Home Page"')
def step_impl(context):
    """ Make a call to the base URL """
    context.driver.get(context.base_url)

@then('I should see "{message}"')
def step_impl(context, message):
    """ Check the document title for a message
"""
    assert message in context.driver.title

@then('I should not see "{message}"')
def step_impl(context, message):
    assert message not in context.resp.text
```

Loading Test Data

- Sometimes you want to provide test data for your scenarios:

```
Background:
  Given a set of specific users
      | name       | department |
      | Barry      | Shipping   |
      | Pudey      | Receiving  |
      | Two-Lumps  | Receiving  |

Scenario: How many people in departments
  When we count the number of people in each department
  Then we will find two people in "Receiving"
  But we will find one person in "Shipping"
```

Steps to Load Sample data

- Sample data does not load itself
- You must write code to load it from the `context.table` variable

```
@given('a set of specific users')
def step_impl(context):
    for row in context.table:
        users.append({
                        'name': row['name'],
                        'department': row['department']
                       })
```

Environment Setup

- Behave has a way to setup all of the tests in one place: `environment.py`

```
import os
from behave import *
from selenium import webdriver

BASE_URL = os.getenv('BASE_URL', 'http://localhost:5000')

def before_all(context):
    """ Executed once before all tests """
    context.driver = webdriver.PhantomJS()
    context.driver.set_window_size(1120, 550)
    context.base_url = BASE_URL
```

More Environment Functions

- `before_step(context, step)`, `after_step(context, step)`
  - These run before and after every step. The step passed in is an instance of Step.
- `before_scenario(context, scenario)`, `after_scenario(context, scenario)`
  - These run before and after each scenario is run. The scenario passed in is an instance of Scenario.
- `before_feature(context, feature)`, `after_feature(context, feature)`
  - These run before and after each feature file is exercised. The feature passed in is an instance of Feature.

More Environment Functions

- `before_tag(context, tag)`, `after_tag(context, tag)`
  - These run before and after a section tagged with the given name.
  - They are invoked for each tag encountered in the order they're found in the feature file.
- `before_all(context)`, `after_all(context)`
  - These run before and after the whole shooting match.

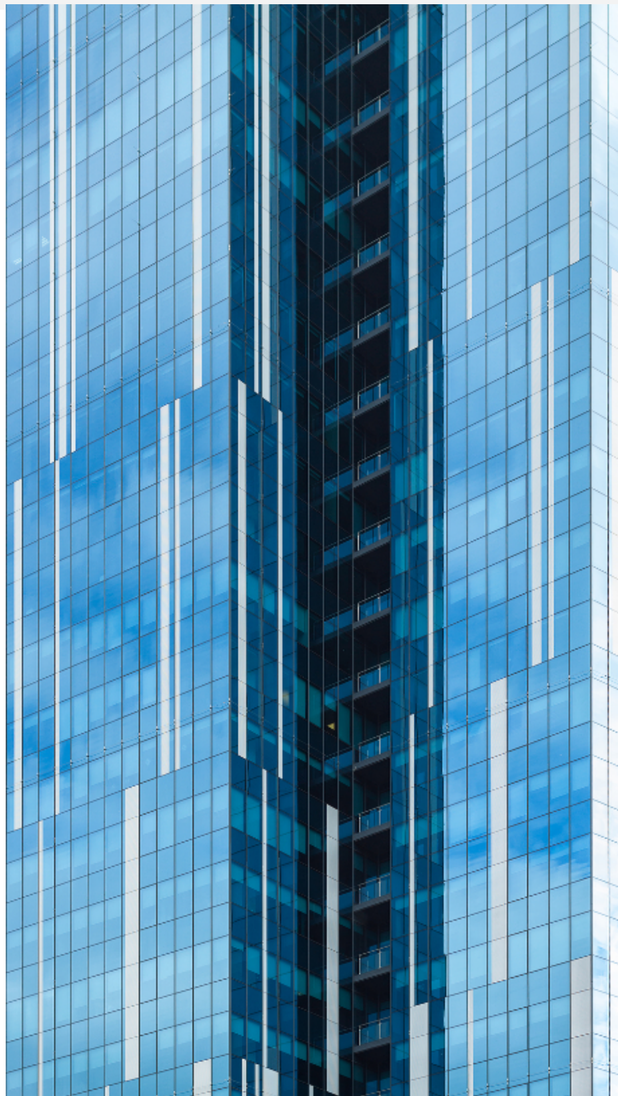
Logging Setup

- The following recipe works in all cases (log-capture on or off). If you want to use/configure logging, you should use the following snippet:

```
# -- FILE:features/environment.py
def before_all(context):
    # -- SET LOG LEVEL: behave --logging-level=ERROR ...
    # on behave command-line or in "behave.ini".
    context.config.setup_logging()

    # -- ALTERNATIVE: Setup logging with a configuration file.
    # context.config.setup_logging(configfile="behave_logging.ini")
```

Selenium

- Selenium automates browsers. That's it!
- It is a web browser driver to test web sites by interacting with the user interface just like a human would
- It supports Firefox, Chrome, Safari, IE, PhantonJS
- This makes it perfect for testing the integration of multiple microservices that share a common user interface



Initialize Selenium

- You can initialize Selenium by requesting a web drive and saving it in the behave context
- You must have that browser installed in order to use it

```
def before_all(context):
    """ Executed once before all tests """
    context.driver = webdriver.PhantomJS()
    context.driver.set_window_size(1120, 550)
    context.base_url = BASE_URL
```

Get the Value of a Text Field

- To get the value of a text field and assert that it contains some string

```
def step_impl(context, text_string):
    """ Get the value if a text field """
    element = context.driver.find_element_by_id(element_id)
    assert text_string in element.get_attribute('value')
```

Wait for Value

- To get the value of a text field and assert that it contains some string

```
def step_impl(context, text_string):
    """ Get the value if a text field """
    found = WebDriverWait(context.driver, WAIT_SECONDS).until(
        expected_conditions.text_to_be_present_in_element_value(
            (By.ID, element_id),
            text_string
        )
    )
    expect(found).to_be(True)
```

Type String into a Text Field

- Type a string into a text field

```
def step_impl(context, text_string):
    """ Type a string into a text field """
    element = context.driver.find_element_by_id(element_id)
    element.clear()
    element.send_keys(text_string)
```

Secret 🕵 Behave Sauce

Use this in IBM Cloud Pipeline Testing Stage

```
#!/bin/bash
#Invoke tests here
python --version

echo "**** Installing Chrome Headless..."
apt-get update
apt-get -qq install -y chromium-driver python3-selenium > /dev/null
echo "Chrome driver located at:"
which chromedriver
chromedriver --version

echo "**** Installing Python library requirements..."
pip install -qr requirements.txt

echo "*****************************************************"
echo "  R U N N I N G   T H E   T E S T S "
echo "*****************************************************"
echo "BASE_URL=" $BASE_URL
behave
```

Hands-On

“live session”

Some Assembly Required

- Tools you will need to complete this lab:
  - Computer running macOS, Linux, or Windows*
  - Text Editor (i will use atom.io)
  - Internet Access to clone GitHub repo
  - GitHub Account

Vagrant Time!

- Behave with Selenium needs a running app to test against
- We will use the ampersand (&) to run our app in the background

```
$ vagrant ssh
$ cd /vagrant
$ honcho start 2>&1 > /dev/null &
$ behave
```

- Alternately you can set the log level to critical

```
$ gunicorn --bind 0.0.0.0 --log-level=critical service:app &
```

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                              # features/pets.feature:13
    Given the following pets                                   # features/steps/pet_steps.py:15 0.033s
      | id | name  | category | available |
      | 1  | fido  | dog      | True      |
      | 2  | kitty | cat      | True      |
      | 3  | leo   | lion     | True      |
    When I visit the "Home Page"                               # features/steps/pet_steps.py:32 0.572s
    Then I should see "Pet Demo RESTful Service" in the title # features/steps/pet_steps.py:37 0.002s
    And I should not see "404 Not Found"                       # features/steps/pet_steps.py:42 0.024s

  Scenario: Create a Pet                    # features/pets.feature:18
    Given the following pets                # features/steps/pet_steps.py:15 0.027s
      | id | name  | category | available |
      | 1  | fido  | dog      | True      |
      | 2  | kitty | cat      | True      |
      | 3  | leo   | lion     | True      |
    When I visit the "Home Page"            # features/steps/pet_steps.py:32 0.022s
    And I set the "Name" to "Happy"         # features/steps/pet_steps.py:46 0.056s
    And I set the "Category" to "Hippo"     # features/steps/pet_steps.py:46 0.046s
    And I press the "Create" button         # features/steps/pet_steps.py:61 0.046s
    Then I should see the message "Success" # features/steps/pet_steps.py:76 0.019s

  Scenario: List all pets                   # features/pets.feature:25
    Given the following pets                # features/steps/pet_steps.py:15 0.029s
      | id | name  | category | available |
      | 1  | fido  | dog      | True      |
      | 2  | kitty | cat      | True      |
      | 3  | leo   | lion     | True      |
    When I visit the "Home Page"            # features/steps/pet_steps.py:32 0.024s
    And I press the "Search" button         # features/steps/pet_steps.py:61 0.044s
    Then I should see "fido" in the results # features/steps/pet_steps.py:66 0.032s
    And I should see "kitty" in the results # features/steps/pet_steps.py:66 0.016s
    And I should see "leo" in the results   # features/steps/pet_steps.py:66 0.017s

  Scenario: List all dogs                      # features/pets.feature:32
    Given the following pets                   # features/steps/pet_steps.py:15 0.033s
      | id | name  | category | available |
      | 1  | fido  | dog      | True      |
      | 2  | kitty | cat      | True      |
      | 3  | leo   | lion     | True      |
    When I visit the "Home Page"               # features/steps/pet_steps.py:32 0.021s
    And I set the "Category" to "dog"          # features/steps/pet_steps.py:46 0.047s
    And I press the "Search" button            # features/steps/pet_steps.py:61 0.035s
    Then I should see "fido" in the results    # features/steps/pet_steps.py:66 0.019s
    And I should not see "kitty" in the results # features/steps/pet_steps.py:71 0.015s
    And I should not see "leo" in the results  # features/steps/pet_steps.py:71 0.013s


1 feature passed, 0 failed, 0 skipped
4 scenarios passed, 0 failed, 0 skipped
23 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m1.633s
```

Stop the Background Server

- Since we started the server in the background with '&'
- We need to bring the server to the foreground to stop it

```
$ fg
$ <ctrl+c>
```

Let’s Reset

- Renamed the `features` folder to start fresh

```
$ cd /vagrant
$ mv features/ _features/
$
```

- Then run `behave`

```
$ behave
ConfigError: No steps directory in "/vagrant/features"
$
```

- The error is because there is no features folder so let’s correct that by creating one

Create the Behave File structure

- Create the behave file structure:
  - Folders: We need a `features` folder and a `features/steps` folder
  - Files: We need `pets.feature` and `pet_steps.py`

```
$ mkdir -p features/steps
$ touch features/pets.feature
$ touch features/steps/pet_steps.py
$
```

```
features
├── pets.feature
└── steps
    └── pet_steps.py
```

Run Behave

- Behave will no longer give an error, but there are still no tests

```
$ mkdir -p features/steps
$ touch features/pets.feature
$ touch features/steps/pet_steps.py
$ behave
0 features passed, 0 failed, 0 skipped
0 scenarios passed, 0 failed, 0 skipped
0 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.000s
$
```

Create a Feature file

https://gist.github.com/rofrano/617d5c1c5d619f4dfa80944370ecd0ef

- Edit the `pets.feature` file to have the following contents:

```
Feature: The pet store service back-end
    As a Pet Store Owner
    I need a RESTful catalog service
    So that I can keep track of all my pets

Background:
    Given the server is started

Scenario: The server is running
    When I visit the "home page"
    Then I should see "Pet Demo REST API Service"
    And  I should not see "404 Not Found"
```

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                 # features/pets.feature:9
    Given the server is started                   # None
    When I visit the "home page"                  # None
    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"          # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
0 steps passed, 0 failed, 0 skipped, 4 undefined
Took 0m0.000s

You can implement step definitions for undefined steps with these snippets:

@given(u'the server is started')
def step_impl(context):
    raise NotImplementedError(u'STEP: Given the server is started')

@when(u'I visit the "home page"')
def step_impl(context):
    raise NotImplementedError(u'STEP: When I visit the "home page"')

@then(u'I should see "Pet Demo REST API Service"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should see "Pet Demo REST API Service"')

@then(u'I should not see "404 Not Found"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should not see "404 Not Found"')
```

Run behave

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                  # features/pets.feature:9
    Given the server is started                    # None
    When I visit the "home page"                   # None
    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"           # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
0 steps passed, 0 failed, 0 skipped, 4 undefined
Took 0m0.000s

You can implement step definitions for undefined steps with these snippets:

@given(u'the server is started')
def step_impl(context):
    raise NotImplementedError(u'STEP: Given the server is started')

@when(u'I visit the "home page"')
def step_impl(context):
    raise NotImplementedError(u'STEP: When I visit the "home page"')

@then(u'I should see "Pet Demo REST API Service"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should see "Pet Demo REST API Service"')

@then(u'I should not see "404 Not Found"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should not see "404 Not Found"')
```

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:    # features/pets.feature:6

  Scenario: The server is running                 # features/pets.feature
    Given the server is started                   # None
    When I visit the "home page"                  # None
    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"          # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
0 steps passed, 0 failed, 0 skipped, 4 undefined
Took 0m0.000s

You can implement step definitions for undefined steps with these snippets:

@given(u'the server is started')
def step_impl(context):
    raise NotImplementedError(u'STEP: Given the server is started')

@when(u'I visit the "home page"')
def step_impl(context):
    raise NotImplementedError(u'STEP: When I visit the "home page"')

@then(u'I should see "Pet Demo REST API Service"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should see "Pet Demo REST API Service"')

@then(u'I should not see "404 Not Found"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should not see "404 Not Found"')
```

Run `behave`

This is where it failed
Yellow means the implementation is missing

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:    # features/pets.feature:6

  Scenario: The server is running                    # features/pets.feature
    Given the server is started                      # None
    When I visit the "home page"                     # None
    Then I should see "Pet Demo REST API Service"    # None
    And I should not see "404 Not Found"             # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
0 steps passed, 0 failed, 0 skipped, 4 undefined
Took 0m0.000s

You can implement step definitions for undefined steps with these snippets:

@given(u'the server is started')
def step_impl(context):
    raise NotImplementedError(u'STEP: Given the server is started')

@when(u'I visit the "home page"')
def step_impl(context):
    raise NotImplementedError(u'STEP: When I visit the "home page"')

@then(u'I should see "Pet Demo REST API Service"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should see "Pet Demo REST API Service"')

@then(u'I should not see "404 Not Found"')
def step_impl(context):
    raise NotImplementedError(u'STEP: Then I should not see "404 Not Found"')
```

Run `behave`

This is where it failed
Yellow means the implementation is missing

These are the steps we need to process the feature file

Let's *cut-n-paste* them into our `pet_steps.py` file

Needed Steps

- Add the following to the `pet_steps.py` file:

```
from behave import *
from service import app

@given(u'the server is started')
def step_impl(context):
    raise NotImplementedError('STEP: Given the server is started')

@when(u'I visit the "home page"')
def step_impl(context):
    raise NotImplementedError('STEP: When I visit the "home page"')

@then(u'I should see "Pet Demo REST API Service"')
def step_impl(context):
    raise NotImplementedError('STEP: Then I should see "Pet Demo REST API Service"')

@then(u'I should not see "404 Not Found"')
def step_impl(context):
    raise NotImplementedError('STEP: Then I not should see "404 Not Found"')
```

Needed Steps

- Add the following to the `pet_steps.py` file:

```
from behave import *
from service import app
```

These are critical to making it work but not suggested by Behave

```
@given(u'the server is started')
def step_impl(context):
    raise NotImplementedError('STEP: Given the server is started')

@when(u'I visit the "home page"')
def step_impl(context):
    raise NotImplementedError('STEP: When I visit the "home page"')

@then(u'I should see "Pet Demo REST API Service"')
def step_impl(context):
    raise NotImplementedError('STEP: Then I should see "Pet Demo REST API Service"')

@then(u'I should not see "404 Not Found"')
def step_impl(context):
    raise NotImplementedError('STEP: Then I not should see "404 Not Found"')
```

Run Behave Again

- Run the `behave` command again to see what we get
- This time the results are different because the steps exist
- But right now, all they do is raise exceptions
- We will need to replace this with the real test code

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                # features/pets.feature:9
    Given the server is started                  # features/steps/pet_steps.py:4 0.001s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 6, in step_impl
          raise NotImplementedError(u'STEP: Given the server is started')
      NotImplementedError: STEP: Given the server is started

    When I visit the "home page"                 # None
    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"         # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
0 steps passed, 1 failed, 3 skipped, 0 undefined
Took 0m0.001s
```

Run behave

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                  # features/pets.feature:9
    Given the server is started                    # features/steps/pet_steps.py:4 0.001s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 6, in step_impl
          raise NotImplementedError(u'STEP: Given the server is started')
      NotImplementedError: STEP: Given the server is started

    When I visit the "home page"                   # None
    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"           # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
0 steps passed, 1 failed, 3 skipped, 0 undefined
Took 0m0.001s
```

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:    # features/pets.feature:6

  Scenario: The server is running                 # features/
    Given the server is started                   # features/
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 6, in step_impl
          raise NotImplementedError(u'STEP: Given the server is started')
      NotImplementedError: STEP: Given the server is started

    When I visit the "home page"                  # None
    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"          # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
0 steps passed, 1 failed, 3 skipped, 0 undefined
Took 0m0.001s
```

Run behave

This is the RED in the RED/GREEN/REFACTOR loop

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:    # features/pets.feature:6

  Scenario: The server is running                 # features/
    Given the server is started                   # features/
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 6, in step_impl
          raise NotImplementedError(u'STEP: Given the server is started')
      NotImplementedError: STEP: Given the server is started

    When I visit the "home page"                  # None
    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"          # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
0 steps passed, 1 failed, 3 skipped, 0 undefined
Took 0m0.001s
```

Run behave

This is the RED in the RED/GREEN/REFACTOR loop

No big surprise because we are the ones that threw the exception

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:    # features/pets.feature:6

  Scenario: The server is running                 # features/
    Given the server is started                   # features/
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 6, in step_impl
          raise NotImplementedError(u'STEP: Given the server is started')
      NotImplementedError: STEP: Given the server is started

    When I visit the "home page"                  # None
    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"          # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
0 steps passed, 1 failed, 3 skipped, 0 undefined
Took 0m0.001s
```

Run behave

This is the RED in the RED/GREEN/REFACTOR loop

No big surprise because we are the ones that threw the exception

These were skipped

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:    # features/pets.feature:6

  Scenario: The server is running                    # features/
    Given the server is started                      # features/
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 6, in step_impl
          raise NotImplementedError(u'STEP: Given the server is started')
      NotImplementedError: STEP: Given the server is started

    When I visit the "home page"                     # None
    Then I should see "Pet Demo REST API Service"    # None
    And I should not see "404 Not Found"             # None

Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
0 steps passed, 1 failed, 3 skipped, 0 undefined
Took 0m0.001s
```

Run behave

This is the RED in the RED/GREEN/REFACTOR loop

No big surprise because we are the ones that threw the exception

These were skipped

This is the scenario in the features file that failed (line 9)

Add code to make it work

- Let's add code to `pet_steps.py` to make sure that the server is running
- This code starts the Flask test client to test our app and saves it to the `context` variable

```
@given('the server is started')
def step_impl(context):
    context.app = app.test_client()
```

What is Context?

- The `context` variable is passed into every step definition
- It survives the entire feature file and all of the steps
- That makes it a great place to pass information from one step to the next
- This is why we are storing `app` and `server` in the `context`

```
@given('the server is started')
def step_impl(context):
    context.app = app.test_client()


@when('I visit the "home page"')
def step_impl(context):
    context.resp = context.app.get('/')

@then('I should see "{message}"')
def step_impl(context, message):
    assert message in str(context.resp.data)
```

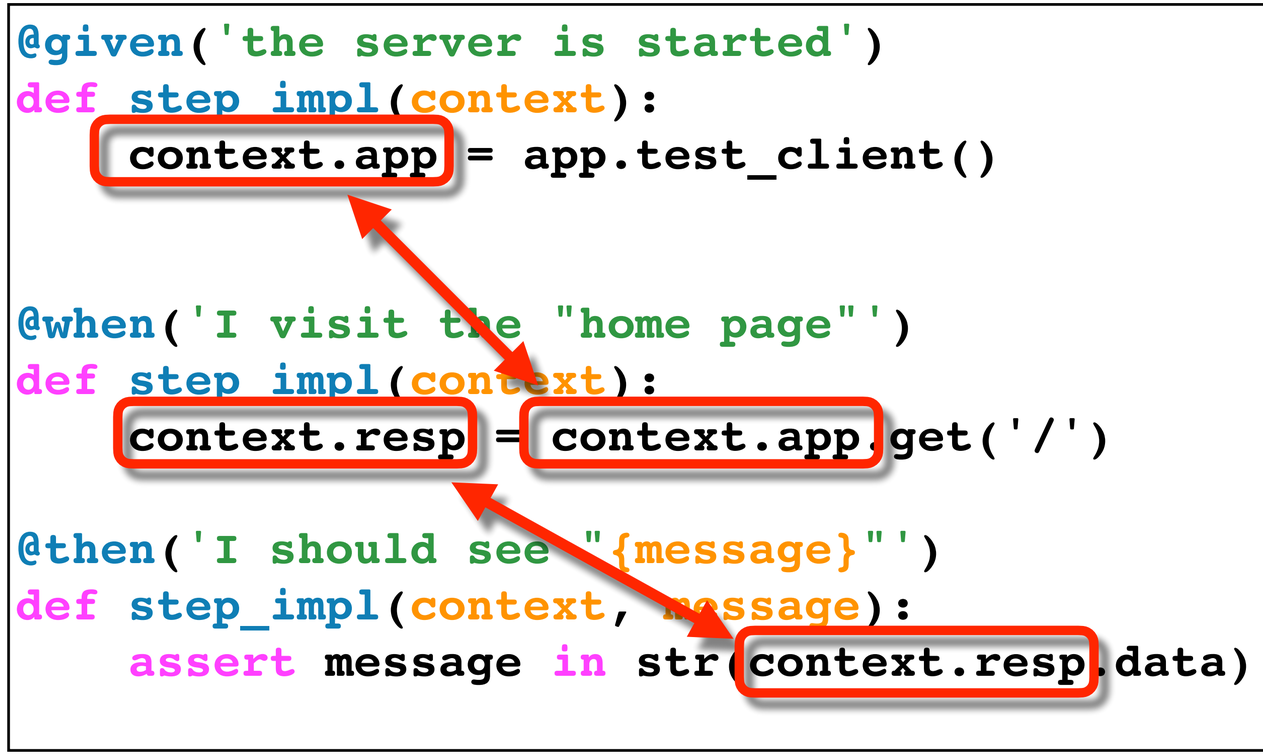
What is Context?

- The `context` variable is passed into every step definition
- It survives the entire feature file and all of the steps
- That makes it a great place to pass information from one step to the next
- This is why we are storing `app` and `server` in the `context`

```
@given('the server is started')
def step_impl(context):
    context.app = app.test_client()


@when('I visit the "home page"')
def step_impl(context):
    context.resp = context.app.get('/')

@then('I should see "{message}"')
def step_impl(context, message):
    assert message in str(context.resp.data)
```

What is Context?

- The `context` variable is passed into every step definition
- It survives the entire feature file and all of the steps
- That makes it a great place to pass information from one step to the next
- This is why we are storing `app` and `server` in the `context`



```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                # features/pets.feature:9
    Given the server is started                  # features/steps/pet_steps.py:4 0.011s
    When I visit the "home page"                 # features/steps/pet_steps.py:9 0.000s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 11, in step_impl
          raise NotImplementedError(u'STEP: When I visit the "home page"')
      NotImplementedError: STEP: When I visit the "home page"

    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"          # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
1 step passed, 1 failed, 2 skipped, 0 undefined
Took 0m0.011s
```

Run behave

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                  # features/pets.feature:9
    Given the server is started                    # features/steps/pet_steps.py:4 0.011s
    When I visit the "home page"                   # features/steps/pet_steps.py:9 0.000s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 11, in step_impl
          raise NotImplementedError(u'STEP: When I visit the "home page"')
      NotImplementedError: STEP: When I visit the "home page"

    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"          # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
1 step passed, 1 failed, 2 skipped, 0 undefined
Took 0m0.011s
```

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:    # features/pets.feature:6

  Scenario: The server is running                 # features/pets.
    Given the server is started                   # features/steps/pet_steps.py:4 0.011s
    When I visit the "home page"                  # features/steps/pet_steps.py:9 0.000s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 11, in step_impl
          raise NotImplementedError(u'STEP: When I visit the "home page"')
      NotImplementedError: STEP: When I visit the "home page"

    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"          # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
1 step passed, 1 failed, 2 skipped, 0 undefined
Took 0m0.011s
```

Run behave

This is now GREEN so we passed this step

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                    # features/pets.feature:9
    Given the server is started                      # features/steps/pet_steps.py:4 0.011s
    When I visit the "home page"                     # features/steps/pet_steps.py:9 0.000s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 11, in step_impl
          raise NotImplementedError(u'STEP: When I visit the "home page"')
      NotImplementedError: STEP: When I visit the "home page"

    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"          # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
1 step passed, 1 failed, 2 skipped, 0 undefined
Took 0m0.011s
```

Run behave

It's now time to fix the next RED step

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:    # features/pets.feature:6

  Scenario: The server is running                    # features/pets.feature:9
    Given the server is started                      # features/steps/pet_steps.py:4 0.011s
    When I visit the "home page"                     # features/steps/pet_steps.py:9 0.000s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 11, in step_impl
          raise NotImplementedError(u'STEP: When I visit the "home page"')
      NotImplementedError: STEP: When I visit the "home page"

    Then I should see "Pet Demo REST API Service" # None
    And I should not see "404 Not Found"          # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
1 step passed, 1 failed, 2 skipped, 0 undefined
Took 0m0.011s
```

Run `behave`

It's now time to fix the next RED step

Let's add the code to implement this test starting on line `11` of `pet_steps.py`

Add code to make it work

- Let’s add code to `pet_steps.py` to make the "home page" step work
- This code actually calls the `GET` method of our app for the URL `'/'`

```
@when('I visit the "home page"')
def step_impl(context):
    context.resp = context.app.get('/')
```

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                 # features/pets.feature:9
    Given the server is started                   # features/steps/pet_steps.py:4 0.009s
    When I visit the "home page"                  # features/steps/pet_steps.py:9 0.012s
    Then I should see "Pet Demo REST API Service" # features/steps/pet_steps.py:13 0.001s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 15, in step_impl
          raise NotImplementedError(u'STEP: Then I should see "Pet Demo REST API Service"')
      NotImplementedError: STEP: Then I should see "Pet Demo REST API Service"

    And I should not see "404 Not Found"          # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
2 steps passed, 1 failed, 1 skipped, 0 undefined
Took 0m0.022s
```

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                  # features/pets.feature:9
    Given the server is started                    # features/steps/pet_steps.py:4 0.009s
    When I visit the "home page"                   # features/steps/pet_steps.py:9 0.012s
    Then I should see "Pet Demo REST API Service" # features/steps/pet_steps.py:13 0.001s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 15, in step_impl
          raise NotImplementedError(u'STEP: Then I should see "Pet Demo REST API Service"')
      NotImplementedError: STEP: Then I should see "Pet Demo REST API Service"

    And I should not see "404 Not Found"           # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
2 steps passed, 1 failed, 1 skipped, 0 undefined
Took 0m0.022s
```

Run behave

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:    # features/pets.feature:6

  Scenario: The server is running                    # features/
    Given the server is started                      # features/
    When I visit the "home page"                     # features/steps/pet_steps.py:9 0.012s
    Then I should see "Pet Demo REST API Service" # features/steps/pet_steps.py:13 0.001s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 15, in step_impl
          raise NotImplementedError(u'STEP: Then I should see "Pet Demo REST API Service"')
      NotImplementedError: STEP: Then I should see "Pet Demo REST API Service"

    And I should not see "404 Not Found"             # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
2 steps passed, 1 failed, 1 skipped, 0 undefined
Took 0m0.022s
```

Run behave

The next one is now GREEN so we passed this step

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                    # features/pets.feature:9
    Given the server is started                      # features/steps/pet_steps.py:4 0.009s
    When I visit the "home page"                     # features/steps/pet_steps.py:9 0.012s
    Then I should see "Pet Demo REST API Service"    # features/steps/pet_steps.py:13 0.001s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 15, in step_impl
          raise NotImplementedError(u'STEP: Then I should see "Pet Demo REST API Service"')
      NotImplementedError: STEP: Then I should see "Pet Demo REST API Service"

    And I should not see "404 Not Found"             # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
2 steps passed, 1 failed, 1 skipped, 0 undefined
Took 0m0.022s
```

Run behave

Hopefully you are seeing the pattern?

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:    # features/pets.feature:6

  Scenario: The server is running                    # features/pets.feature:9
    Given the server is started                      # features/steps/pet_steps.py:4 0.009s
    When I visit the "home page"                     # features/steps/pet_steps.py:9 0.012s
    Then I should see "Pet Demo REST API Service"    # features/steps/pet_steps.py:13 0.001s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 15, in step_impl
          raise NotImplementedError(u'STEP: Then I should see "Pet Demo REST API Service"')
      NotImplementedError: STEP: Then I should see "Pet Demo REST API Service"

    And I should not see "404 Not Found"             # None


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
2 steps passed, 1 failed, 1 skipped, 0 undefined
Took 0m0.022s
```

Run behave

Hopefully you are seeing the pattern?

Wash… Rinse… Repeat… ;-)

Add code to make it work

- Let’s add code to `pet_steps.py` to make the "I should see" step work
- This code checks that the message is in the response data

```
@then('I should see "{message}"')
def step_impl(context, message):
    assert message in str(context.resp.data)
```

Add code to make it work

- Let’s add code to `pet_steps.py` to make the "I should see" step work
- This code checks that the message is in the response data

```
@then('I should see "{message}"')
def step_impl(context, message):
    assert message in str(context.resp.data)
```

We use variable substitution so that this step can be reused to check any message

Variable Substitution

- You can use variable substitution to make the steps more generic
- This step can now be used to check for any message in a response
- It’s a good idea to make steps as generic as possible for maximum reuse

Before:

```
@then('I should see "Pet Demo REST API Service"')
def step_impl(context):
    assert message in str(context.resp.data)
```

After:

```
@then('I should see "{message}"')
def step_impl(context, message):
    assert message in str(context.resp.data)
```

Variable Substitution

- You can use variable substitution to make the steps more generic
- This step can now be used to check for any message in a response
- It’s a good idea to make steps as generic as possible for maximum reuse

We replace this string with a variable

Before:

```
@then('I should see "Pet Demo REST API Service"')
def step_impl(context):
    assert message in str(context.resp.data)
```

After:

```
@then('I should see "{message}"')
def step_impl(context, message):
    assert message in str(context.resp.data)
```

Variable Substitution

- You can use variable substitution to make the steps more generic
- This step can now be used to check for any message in a response
- It's a good idea to make steps as generic as possible for max

We replace this string with a variable

Add it to the string using {}

Before:

```
@then('I should see "Pet Demo REST API Service"')
def step_impl(context):
    assert message in str(context.resp.data)
```

```
@then('I should see "{message}"')
def step_impl(context, message):
    assert message in str(context.resp.data)
```

Variable Substitution

- You can use variable substitution to make the steps more generic
- This step can now be used to check for any message in a response
- It’s a good idea to make steps as generic as possible for maximum reuse

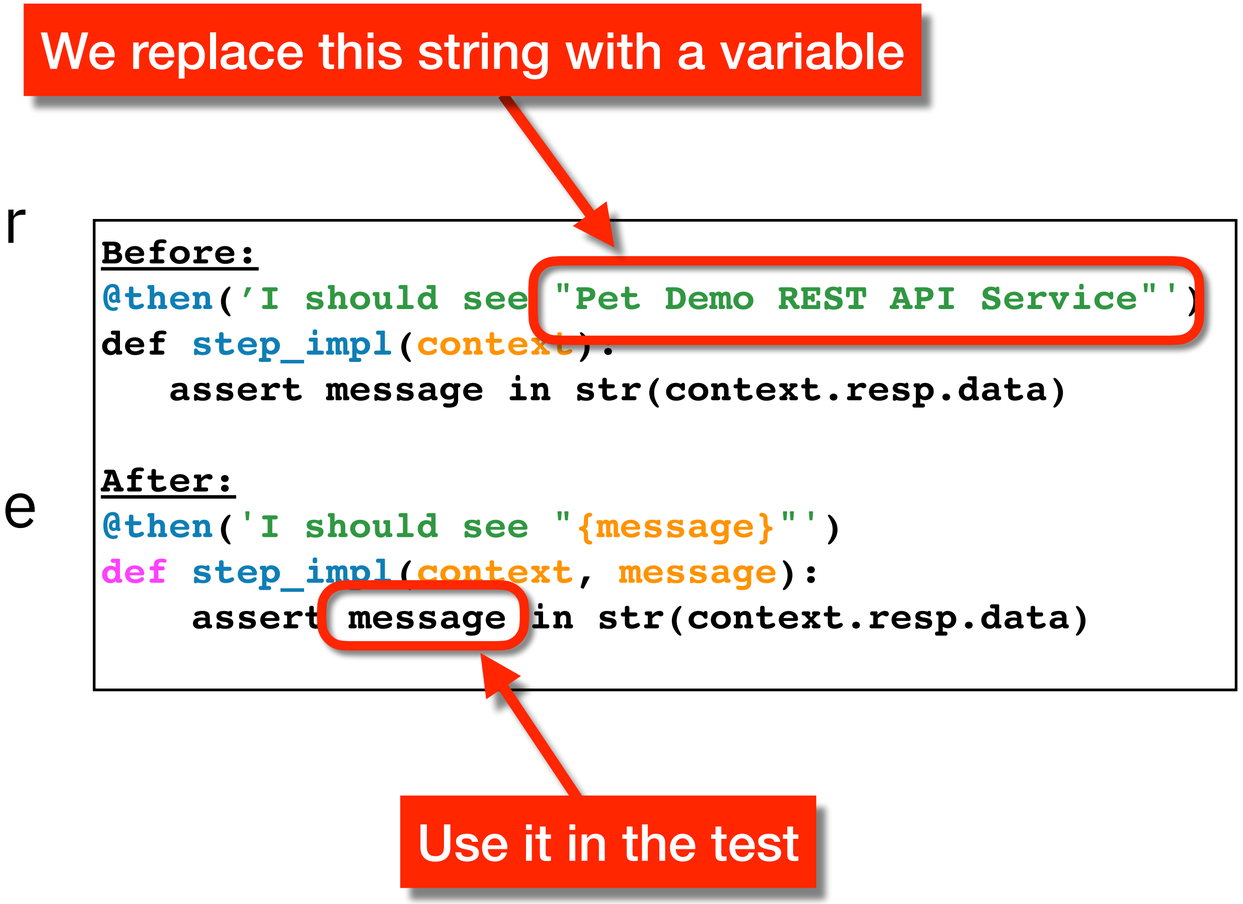
We replace this string with a variable

```
Before:
@then('I should see "Pet Demo REST API Service"')
def step_impl(context):
    assert message in str(context.resp.data)

After:
@then('I should see "{message}"')
def step_impl(context, message):
    assert message in str(context.resp.data)
```

And pass it into the function

Variable Substitution

- You can use variable substitution to make the steps more generic
- This step can now be used to check for any message in a response
- It’s a good idea to make steps as generic as possible for maximum reuse



```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                 # features/pets.feature:9
    Given the server is started                   # features/steps/pet_steps.py:4 0.008s
    When I visit the "home page"                  # features/steps/pet_steps.py:9 0.007s
    Then I should see "Pet Demo REST API Service" # features/steps/pet_steps.py:13 0.000s
    And I not should see "404 Not Found"          # features/steps/pet_steps.py:17 0.000s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 19, in step_impl
          raise NotImplementedError(u'STEP: Then I not should see "404 Not Found"')
      NotImplementedError: STEP: Then I not should see "404 Not Found"



Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
3 steps passed, 1 failed, 0 skipped, 0 undefined
Took 0m0.015s
```

Run behave

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                  # features/pets.feature:9
    Given the server is started                    # features/steps/pet_steps.py:4 0.008s
    When I visit the "home page"                   # features/steps/pet_steps.py:9 0.007s
    Then I should see "Pet Demo REST API Service" # features/steps/pet_steps.py:13 0.000s
    And I not should see "404 Not Found"           # features/steps/pet_steps.py:17 0.000s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 19, in step_impl
          raise NotImplementedError(u'STEP: Then I not should see "404 Not Found"')
      NotImplementedError: STEP: Then I not should see "404 Not Found"



Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
3 steps passed, 1 failed, 0 skipped, 0 undefined
Took 0m0.015s
```

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:    # features/pets.feature:6

  Scenario: The server is running                    # features/pets.feature:9
    Given the server is started                      # features/steps/pet_steps.py:5 0.000s
    When I visit the "home page"                     # features/steps/pet_steps.py:9 0.000s
    Then I should see "Pet Demo REST API Service"    # features/steps/pet_steps.py:13 0.000s
    And I not should see "404 Not Found"             # features/steps/pet_steps.py:17 0.000s
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 19, in step_impl
          raise NotImplementedError(u'STEP: Then I not should see "404 Not Found"')
      NotImplementedError: STEP: Then I not should see "404 Not Found"



Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
3 steps passed, 1 failed, 0 skipped, 0 undefined
Took 0m0.015s
```

Run behave

The next one is now GREEN so we passed this step

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                    # features/pets.feature:9
    Given the server is started                      # features/steps/pet_steps.py:4 0.008s
    When I visit the "home page"                     # features/steps/pet_steps.py:9 0.007s
    Then I should see "Pet Demo REST API Service"    # features/steps/pet_steps.py:13 0.000s
    And I not should see "404 Not Found"             # features/ste
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 19, in step_impl
          raise NotImplementedError(u'STEP: Then I not should see "404 Not Found"')
      NotImplementedError: STEP: Then I not should see "404 Not Found"



Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
3 steps passed, 1 failed, 0 skipped, 0 undefined
Took 0m0.015s
```

Run behave

One last statement needs a step that passes

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                    # features/pets.feature:9
    Given the server is started                      # features/steps/pet_steps.py:4 0.008s
    When I visit the "home page"                     # features/steps/pet_steps.py:9 0.007s
    Then I should see "Pet Demo REST API Service"    # features/steps/pet_steps.py:13 0.000s
    And I not should see "404 Not Found"             # features/ste
      Traceback (most recent call last):
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1456, in run
          match.run(runner.context)
        File "/Library/Python/2.7/site-packages/behave/model.py", line 1903, in run
          self.func(context, *args, **kwargs)
        File "features/steps/pet_steps.py", line 19, in step_impl
          raise NotImplementedError(u'STEP: Then I not should see "404 Not Found"')
      NotImplementedError: STEP: Then I not should see "404 Not Found"


Failing scenarios:
  features/pets.feature:9  The server is running

0 features passed, 1 failed, 0 skipped
0 scenarios passed, 1 failed, 0 skipped
3 steps passed, 1 failed, 0 skipped, 0 undefined
Took 0m0.015s
```

Run behave

One last statement needs a step that passes

Let's implement the last one

Add code to make it work

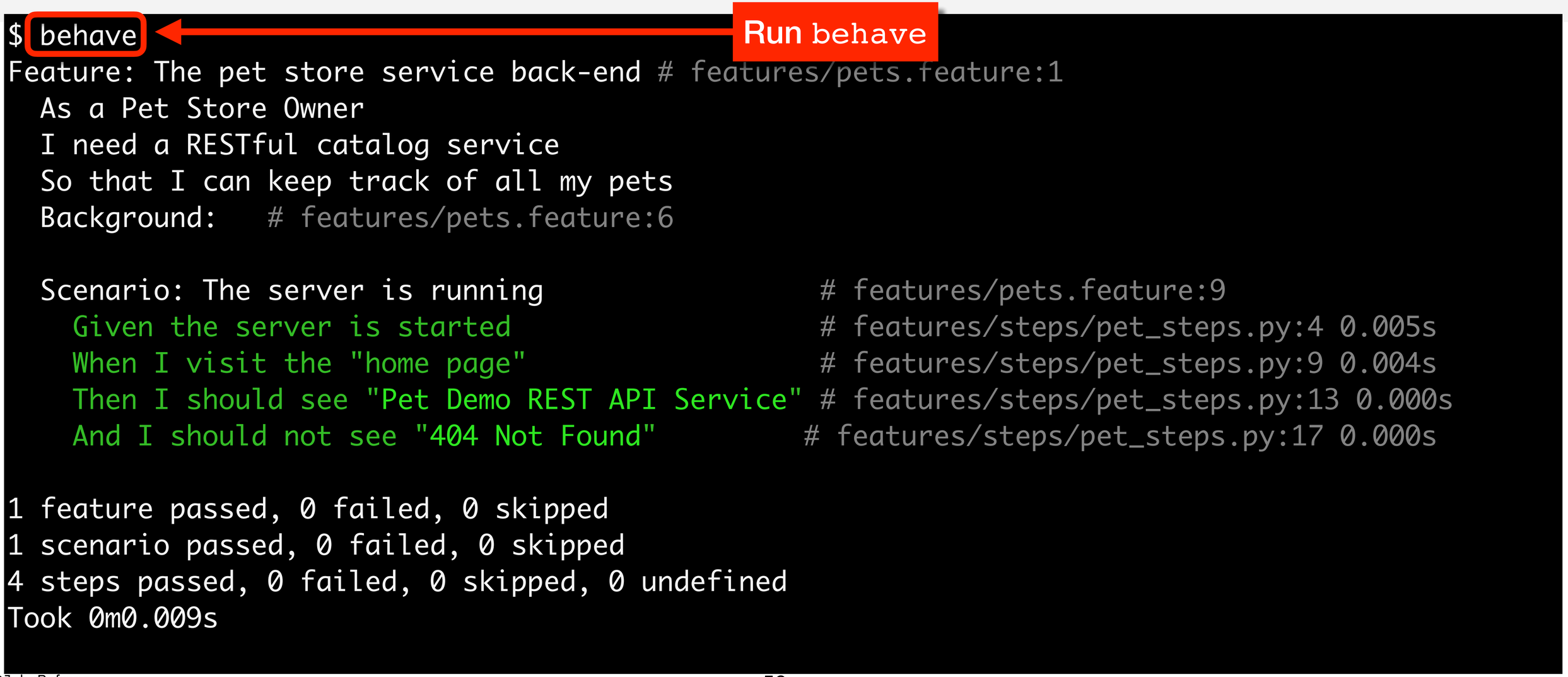
- Let’s add code to `pet_steps.py` to make the "I should not see" step work
- This code is the opposite of the last step. It checks that the message is not in the response data

```
@then('I should not see "{message}"')
def step_impl(context, message):
    assert message not in str(context.resp.data)
```

Success !!!

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                    # features/pets.feature:9
    Given the server is started                      # features/steps/pet_steps.py:4 0.005s
    When I visit the "home page"                     # features/steps/pet_steps.py:9 0.004s
    Then I should see "Pet Demo REST API Service" # features/steps/pet_steps.py:13 0.000s
    And I should not see "404 Not Found"           # features/steps/pet_steps.py:17 0.000s

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
4 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.009s
```

Success !!!




```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                    # features/pets.feature:9
    Given the server is started                      # features/steps/pet_steps.py:4 0.005s
    When I visit the "home page"                     # features/steps/pet_steps.py:9 0.004s
    Then I should see "Pet Demo REST API Service" # features/steps/pet_steps.py:13 0.000s
    And I should not see "404 Not Found"           # features/steps/pet_steps.py:17 0.000s

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
4 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.009s
```

Success !!!

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:   # features/pets.feature:6

  Scenario: The server is running                    # features/pets.feature:9
    Given the server is started                      # features/steps/pet_steps.py:4 0.005s
    When I visit the "home page"                     # features/steps/pet_steps.py:9 0.004s
    Then I should see "Pet Demo REST API Service"    # features/steps/pet_steps.py:13 0.000s
    And I should not see "404 Not Found"             # features/steps/pet_steps.py:17 0.000s

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
4 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.009s
```

Run behave

Everything is GREEN !!!

Success !!!

```
$ behave
Feature: The pet store service back-end # features/pets.feature:1
  As a Pet Store Owner
  I need a RESTful catalog service
  So that I can keep track of all my pets
  Background:    # features/pets.feature:6

  Scenario: The server is running                    # features/pets.feature:9
    Given the server is started                      # features/steps/pet_steps.py:4 0.005s
    When I visit the "home page"                     # features/steps/pet_steps.py:9 0.004s
    Then I should see "Pet Demo REST API Service"    # features/steps/pet_steps.py:13 0.000s
    And I should not see "404 Not Found"             # features/steps/pet_steps.py:17 0.000s

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
4 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.009s
```

Run behave

Everything is GREEN !!!

Everything is PASSING !!!

🙌

That’s All There Is To It

- RED, GREEN, REFACTOR
- Continue to write Scenarios that have no implementation
- Write steps to provide the implementation of the test
- Write the code to make the steps pass
- Refactor the code as needed knowing that the tests will tell you if you broke something

Let’s Reset Again

- Renamed the `_features` folder back to `features and exit`

```
$ rm -fr features
$ mv _features/  features/

$ exit
$ vagrant halt
```

Summary

- You just created your first BDD features and tested them
- Unlike TDD which is for a single microservice, BDD can be used to test cross-microservice functionality
- Hopefully this will help you write better Stories that can be turned into BDD features and tested

Further Reading

- Testing Flask: http://flask.pocoo.org/docs/0.11/testing/
- Behave: https://pythonhosted.org/behave/tutorial.html
- Gherkin: https://cucumber.io/docs/reference