Task 0, create a cli tool

Let's create tool to parse a `nova/nova.conf` file in order to get expected values, like `my_ip` , if what `virt_type` etc..

What needs to be done?

- A function to handle the actual input.
- make it handle arguments from command line
- Let's make it professional
- Use `docopt` to simplify it!

reference: http://docopt.org

> deal with excel/csv etc...

Input processing

reference: https://docs.openstack.org/ocata/config-reference/compute/nova-conf-samples.html

The input is something like:

```
[DEFAULT]

# LOGS/STATE
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova
rootwrap_config=/etc/nova/rootwrap.conf

# SCHEDULER
compute_scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler

# VOLUMES
# configured in cinder.conf

# COMPUTE
compute_driver=libvirt.LibvirtDriver
instance_name_template=instance-%08x
api_paste_config=/etc/nova/api-paste.ini

# COMPUTE/APIS: if you have separate configs for separate services
# this flag is required for both nova-api and nova-compute
allow_resize_to_same_host=True

# APIS
osapi_compute_extension=nova.api.openstack.compute.contrib.standard_extensions
ec2_dmz_host=192.168.206.130
s3_host=192.168.206.130

# RABBITMQ
rabbit_host=192.168.206.130

# GLANCE
image_service=nova.image.glance.GlanceImageService

# NETWORK
network_manager=nova.network.manager.FlatDHCPManager
force_dhcp_release=True
dhcpbridge_flagfile=/etc/nova/nova.conf
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
# Change my_ip to match each host
my_ip=192.168.206.130
public_interface=eth0
vlan_interface=eth0
flat_network_bridge=br100
flat_interface=eth0

# NOVNC CONSOLE
novncproxy_base_url=http://192.168.206.130:6080/vnc_auto.html
# Change vncserver_proxyclient_address and vncserver_listen to match each compute host
vncserver_proxyclient_address=192.168.206.130
vncserver_listen=192.168.206.130


# AUTHENTICATION
```

```
auth_strategy=keystone
[keystone_authtoken]
auth_host = 127.0.0.1
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = nova
signing_dirname = /tmp/keystone-signing-nova

# GLANCE
[glance]
api_servers=192.168.206.130:9292

# DATABASE
[database]
connection=mysql+pymysql://nova:yourpassword@192.168.206.130/nova

# LIBVIRT
[libvirt]
virt_type=qemu
```

Let's simplify it for showing in one screen:

```
[DEFAULT]

# NETWORK
network_manager=nova.network.manager.FlatDHCPManager
force_dhcp_release=True
dhcpbridge_flagfile=/etc/nova/nova.conf
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
# Change my_ip to match each host
my_ip=192.168.206.130
public_interface=eth0
vlan_interface=eth0
flat_network_bridge=br100
flat_interface=eth0
# LIBVIRT
[libvirt]
virt_type=qemu
```

Load it as a string:

`"""` is used for multi-line string

```
In [1]: input = """
   ...: [DEFAULT]
   ...:
   ...: # NETWORK
   ...: network_manager=nova.network.manager.FlatDHCPManager
   ...: force_dhcp_release=True
   ...: dhcpbridge_flagfile=/etc/nova/nova.conf
   ...: firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
   ...: # Change my_ip to match each host
   ...: my_ip=192.168.206.130
   ...: public_interface=eth0
   ...: vlan_interface=eth0
   ...: flat_network_bridge=br100
   ...: flat_interface=eth0
   ...: # LIBVIRT
   ...: [libvirt]
   ...: virt_type=qemu"""
In [4]: print (input)

[DEFAULT]

# NETWORK
network_manager =nova.network.manager.FlatDHCPManager
force_dhcp_release =True
dhcpbridge_flagfile =/etc/nova/nova.conf
firewall_driver =nova.virt.libvirt.firewall.IptablesFirewallDriver
# Change my_ip to match each host
my_ip=192.168.206.130
public_interface =eth0
vlan_interface =eth0
flat_network_bridge =br100
flat_interface =eth0
# LIBVIRT
[libvirt]
virt_type=qemu
```

Let's do some pre-processing on it.

```
In [5]: inputList = input.split("\n")

In [6]: print (inputList)
['', '[DEFAULT]', '', '# NETWORK',
'network_manager=nova.network.manager.FlatDHCPManager' , 'force_dhcp_release=True' ,
'dhcpbridge_flagfile=/etc/nova/nova.conf' ,
'firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver' , '# Change my_ip
to match each host' , 'my_ip=192.168.206.130' , 'public_interface=eth0' ,
'vlan_interface=eth0' , 'flat_network_bridge=br100' , 'flat_interface=eth0' , '#
LIBVIRT', '[libvirt]', 'virt_type=qemu' ]

In [7]: type(inputList)
Out[7]: list
```

Now we have a list, where all items come from the lines of `nova.conf`, let's do something for it!

Step 1 the main function

We could build a function to parse specific parameter name and return the value ;-).

```
def parse(par,inputList):
  #magic things
  return valueForThePar
```

Then let's do the `magic`:

```
def parse(par,inputList):
  valueForThePar = "oops: there is no " + par + " found."
  for line in inputList:
    line = line.strip().split("=")
    if par == line[0]:
      valueForThePar = line[1]
  return valueForThePar
```

Let's run it:

```
In [12]: def parse(par,inputList):
    ...:   valueForThePar = "oops: there is no " + par + " found."
    ...:   for line in inputList:
    ...:     line = line.strip().split("=")
    ...:     if par == line[0]:
    ...:       valueForThePar = line[1]
    ...:   return valueForThePar
    ...:

In [13]: parse("neutron",inputList)
Out[13]: '#Err: there is no neutron found.'

In [14]: parse("virt_type",inputList)
Out[14]: 'qemu'
```

Seems working yet easy, correct?

- But we need a real program to handle a file as input instead of a string coming from copy paste, how could we do that?
- How could we let the program know which parameter need to be parsed?

Step 2 Input

Parse arguments

Let's use `sys.argv`, which is a list of the argument variables passed during program calling.

For example, let's write this test-argv.py:

```
import sys
print (sys.argv[0])
```

run it:

```
$ python3 test-argv.py
['test-argv.py' ]
```

Confused? How about this?

```
import sys
print (sys.argv[0:])
```

run it with different arguments:

```
$ python3 test-argv.py
['test-argv.py' ]
$ python3 test-argv.py a b c
['test-argv.py' , 'a', 'b', 'c']
```

Now with this power we could design our tool like this:

`python3 our-cool-tool-file-name.py --input <path-to-input-file> --par <par>`

Then it comes to the hardest part, naming the tool. Let's call it `novaConfParser.py` ;-).

Let's write `novaConfParser.py`, it's something like:

```
def parseArgVars():
    pass
# parse input in list type
def parse(par,inputList):
  valueForThePar = "oops: there is no " + par + " found."
  for line in inputList:
    line = line.strip().split("=")
    line = [item.strip() for item in line]
    if par == line[0]:
      valueForThePar = line[1]
  return valueForThePar

# ... something in between

# main process
def main():
    parseArgVars()
    print (parse(par,inputList))
main()
```

For the part of `parseArgVars()`, let's make it as below, which just parsed the arguments of `filePath`, `par`, and validate the arguments, if it's not validate, stop everything but provide the help info.

```python
import sys
argVars = sys.argv[1:]
argFormatErrorFlag  = True
helpInfo = """Usage:
  novaConfParser.py [--input <path-to-input-file>] [--par <par>]
Options:
  --help       Show this help screen.
Examples:
  python3 novaConfParser.py --input /nova/nova.conf --par my_ip

"""

# parse parseArgVars
def parseArgVars(argVars, flag):
  # init for filePath, par
  filePath, par = "", ""
  if len(argVars) == 4:
    if "--input" and "--par" in [argVars[0], argVars[2]]:
      # if in order: --input x --par y
      filePath, par = argVars[1], argVars[3]
      # argFormatErrorFlag is valid now
      flag = False
      # switch order if needed, in order: --par x --input y
      if argVars[0] != "--input":
        filePath, par = par, filePath
  return flag, filePath , par

def parse(par,inputList):
  valueForThePar  = "oops: there is no "  + par + " found."
  for line in inputList:
    line = line.strip().split("=")
    if par == line[0]:
      valueForThePar  = line[1]
  return valueForThePar



# main process
def main(argVars, argFormatErrorFlag ):
    # parse arguments and validate them
    argFormatErrorFlag , filePath, par = parseArgVars(argVars, argFormatErrorFlag )
    # in case argFormatErrorFlag is True, end and print the help info
    if argFormatErrorFlag :
        print (helpInfo)
        # End function without raise errors
        return None

    # ... something in between <---------

    print (parse(par,inputList))

# run main function

main(argVars, argFormatErrorFlag )
```

Read file

Till now we have:

- arguments parsed in the begining by `parseArgVars`
- `#... something in between < ---------`
- `nova.conf` 's handler `parse()` to deal with a `list()` input file

What the only something left here is to read the `nova.conf` file and return it as list:

```
def readInputFile(path):
    # do things
    return aListSplitedByLines
```

And it's easy:

```
def readInputFile(path):
    with open(path) as file:
        fileStr = file.read()
    return fileStr.split("\n")
```

A working tool was done!

And now we have the tool done:

```python
import sys
argVars = sys.argv[1:]
argFormatErrorFlag = True

helpInfo = """Usage:
  novaConfParser.py [--input <path-to-input-file>] [--par <par>]
Options:
  --help       Show this help screen.
Examples:
  python3 novaConfParser.py --input /nova/nova.conf --par my_ip

"""

# parse parseArgVars
def parseArgVars(argVars, flag):
  # init for filePath, par
  filePath, par = str(), str()
  if len(argVars) == 4:
    if "--input" and "--par" in [argVars[0], argVars[2]]:
      # if in order: --input x --par y
      filePath, par = argVars[1], argVars[3]
      # argFormatErrorFlag is valid now
      flag = False
      # switch order if needed, in order: --par x --input y
      if argVars[0] != "--input":
        filePath, par = par, filePath
  return flag, filePath , par

def parse(par,inputList):
  valueForThePar = "oops: there is no "  + par + " found."
  for line in inputList:
    line = line.strip().split("=")
    if par == line[0]:
      valueForThePar = line[1]
  return valueForThePar

def readInputFile(path):
  with open(path) as file:
    fileStr = file.read()
  return fileStr.split("\n")

# main process
def main(argVars, argFormatErrorFlag):
  # parse arguments and validate them
  argFormatErrorFlag, filePath, par = parseArgVars(argVars, argFormatErrorFlag)
  # in case argFormatErrorFlag is True, end and print the help info
  if argFormatErrorFlag:
    print (helpInfo)
    # End function without raise errors
    return None
  # Read file as a list
  inputList = readInputFile(filePath)

  # Build final output with parse()
  output = parse(par,inputList)
  # do the output
  print (output)

# run main function
main(argVars, argFormatErrorFlag)
```

Can we make life easier?

Basically our tool is like:

- handling arguments parsed in the beginning by `parseArgVars`
- Read file
- The function do handle main logic

With the tool to enable more complex things possible, the parseArgVars part could be much crazy below are options you could use:

- `argparse` (https://docs.python.org/3/library/argparse.html)
- `docopt` (http://docopt.org)

Let's take `docopt` as an example

docopt example

ref: https://github.com/docopt/docopt/blob/master/examples/

`novaConfParser_V2.py`:

```
"""Usage:
  novaConfParser_V2.py [--input <path-to-input-file>] [--par <par>]

Options:
  --help       Show this help screen.

Examples:
  python3 novaConfParser.py --input /nova/nova.conf --par my_ip
"""
from docopt import docopt


if __name__ == '__main__':
    arguments = docopt(__doc__, version='0.1.1rc')
    print(arguments)
```

Let's verify it:

It's doing thins magically good, we could see in this single example, `arguments` is a `dict`, which is actually covering the argument parse and validation.

```
$ python3 novaConfParser_V2.py
{'--input': False,
 '--par': False,
 '<par>': None,
 '<path-to-input-file>' : None}

$ python3 novaConfParser_V2.py  --input /nova/nova.conf  --par my_ip
{'--input': True,
 '--par': True,
 '<par>': 'my_ip',
 '<path-to-input-file>' : '/nova/nova.conf' }

$ python3 novaConfParser_V2.py  --help
Usage:
  novaConfParser_V2.py [--input <path-to-input-file>] [--par <par>]

Options:
  -h --help       Show this help screen.

Examples:
  python3 novaConfParser_V2.py  --input /nova/nova.conf  --par my_ip

$ python3 novaConfParser_V2.py  --version
0.1.1rc
```

We could directly use it as below:

```
In [7]: arguments["--input"]
Out[7]: True

In [8]: arguments = {'--input': True,
   ...:  '--par': True,
   ...:  '<par>': 'my_ip',
   ...:  '<path-to-input-file>' : '/nova/nova.conf' }
   ...:

In [9]: arguments["--input"]
Out[9]: True

In [10]: arguments['<path-to-input-file>' ]
Out[10]: '/nova/nova.conf'
```

What else?

- Shebang ref: https://en.wikipedia.org/wiki/Shebang_%28Unix%29
- `virtualenv`
- parse from or write to json, excel, xml, csv etc ?
    - built-in library: https://docs.python.org/3/library/index.html
    - Awesome Python: https://github.com/vinta/awesome-python

- Doing things for servers? paramiko / ansible
- exceptions: http://www.runoob.com/python/python-exceptions.html