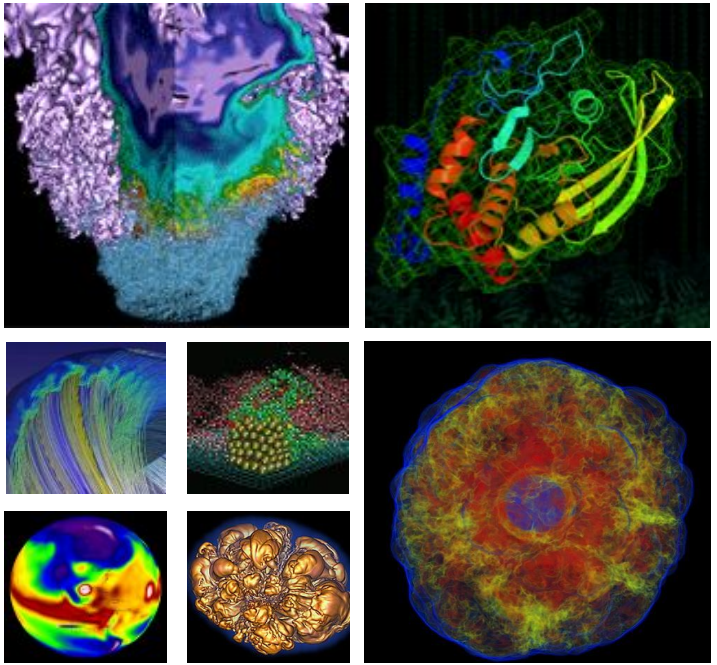
FaaS via fn the New Middleware



Cary ‘data junkie?’ Whitney

HEPiX Fall 2018

NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Outline

- Direction
- Why FaaS?
- Elastic DSL
- How to.
  - fn publisher
  - fn consumer

Why the change of direction?

- Data collects can be big and personal.
  - Different locations have different needs and desires.
  - Size and complexity is also variable.
  - Software changes fast, thus what was best yesterday may not be today.
- The data collect is only a tool.
  - Good to share tool experience and results including components.
- How to access the data collect is also only a tool. (FaaS)
- What is important is the data and the story the data tells.
  - Love to hear some good data stories. I should preset some.

Definitions

- **FaaS is Function as a Service (wikipedia)**

Function as a service (**FaaS**) is a category of cloud computing services that provides a platform allowing customers to develop, run, and manage application functionalities without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app.

- **fn Project**
  - **Software used to perform the FaaS duties.**
  - **https://fnproject.io**
  - **Low overhead, just stand up a simple server not a whole infrastructure.**

Why add FaaS setup?

- Allow easier/simpler access to data.
- We can define a data API not just an access API.
- Data source can change but the client code does not.

Elasticsearch DSL

- This the python bindings on top of Elasticsearch python bindings.
- Simpler definitions to access Elastic data sources.
- Python
  - import elasticsearch
  - import elasticsearch_dsl
- Tangent: These same libraries can be used as Direct Elastic Access or via their API.

Elasticsearch_dsl (Cont)

- https://elasticsearch-dsl.readthedocs.io/en/latest/

```
from elasticsearch import Elasticsearch
from elasticsearch_dsl import Search

indx = modbus*
subkey = meter
subvalue = pdu6

esdb = Elasticsearch([{'host': host,'port': port, }])
s = Search(using=esdb, index=indx)
s = s.query("term", **{subkey: subvalue})
s = s.query('range', **{'@timestamp':{'gte': 'now-30m', ‘lt':'now'}})
s = s.sort(‘-@timestamp')
s = s.aggs.metric('power_sum', 'sum', field=valueField)
s = s[0:1]
response = s.execute()
print 'Total %d hits found.' % response.hits.total
print ‘Power Sum:’ % response.aggregations.power_sum
```

Elasticsearch (Non dsl)

```
{'query':
   {'bool':
    {'must':
      [{'term': {'subkey': 'subvalue'}}, {'range': {'@timestamp': {'gte': 'now-30m', 'lt': 'now'}}}]
   }},
    'sort': [{'@timestamp': {'order': 'desc'}}], 'from': 0, 'size': 1}
```

fn setup

- fn init --runtime python get_agg
- This creates a directory containing:

func.py func.yaml requirements.txt test.json

requirements.txt
fdk
urllib3 <1.23
elasticsearch
elasticsearch_dsl

func.yaml
name: get_agg
version: 0.0.16
runtime: python
entrypoint: python3 func.py
format: json

fn function (func.py)

```
import fdk
import json
from elasticsearch import Elasticsearch
from elasticsearch_dsl import Search

def handler(ctx, data=None, loop=None):
  x = dict()
  if data and len(data) > 0:
    body = json.loads(data)
    index = body.get("index")
    elements = body.get("elements")
  indx = index + '*'
  (subkey, subvalue) = elements.split('|')
  esdb = Elasticsearch([{'host': host,'port': port, }])
  s = Search(using=esdb, index=indx)
  s = s.query("term", **{subkey: subvalue})
  s = s.query('range', **{'@timestamp':{'gte': 'now-30m', 'lt':'now'}})
  s = s.sort('-@timestamp')
  s = s.aggs.metric('power_sum', 'sum', field=valueField)
  s = s[0:1]
  response = s.execute()
  x['hits'] = response.hits.total
  x['power_sum'] = response.aggregations.power_sum
  return json.dumps(x)

if __name__ == "__main__":
  fdk.handle(handler)
```

- Inside this code one could access any and/or multiple data sources.

fn finish

- echo -n '{"index":"modbus", “elements”:"meter|pdu6"}' | fn run
- fn deploy --app get_agg

Client Code

```
#!/usr/bin/python

import json
import requests
import sys
import os
import stat

def main():
  url = “https://fn.nersc.gov:/r/get_agg/get_agg“
  headers = {"Content-Type": "application/json; charset=utf8”}
  data = '{"index":"modbus", “elements":"meter|pdu6"}'
  r = requests.post(url, headers=headers, data=data)

  y = json.loads(r.json())
  hits = y.get(“hits")
  power_sum = y.get(“power_sum”)

  print("hits: %s" % hits)
  print(“Power sum:” % power_sum)

if __name__ == "__main__":
  main()
```

- ISSUES:
  - Need to know the index
  - Need to know the data and data structure

Data API

> ./get_job.py **-s edison -j 10703089**

Gathering Cray nodes
data: **{"system":"edison", "jobid":10703089}**
Job: 10703089 ran on the following nodes: [u'c5-0c1s6n1', u'c3-3c0s2n2', u'c5-0c2s8n0', u'c5-0c2s7n1', u'c3-3c0s1n3', u'c5-0c2s7n3']

Getting job 10703089 start and end times
Job: 10703089 Start: 2018-09-15T00:15:15-0700 and Ended: 2018-09-16T07:52:40-0700

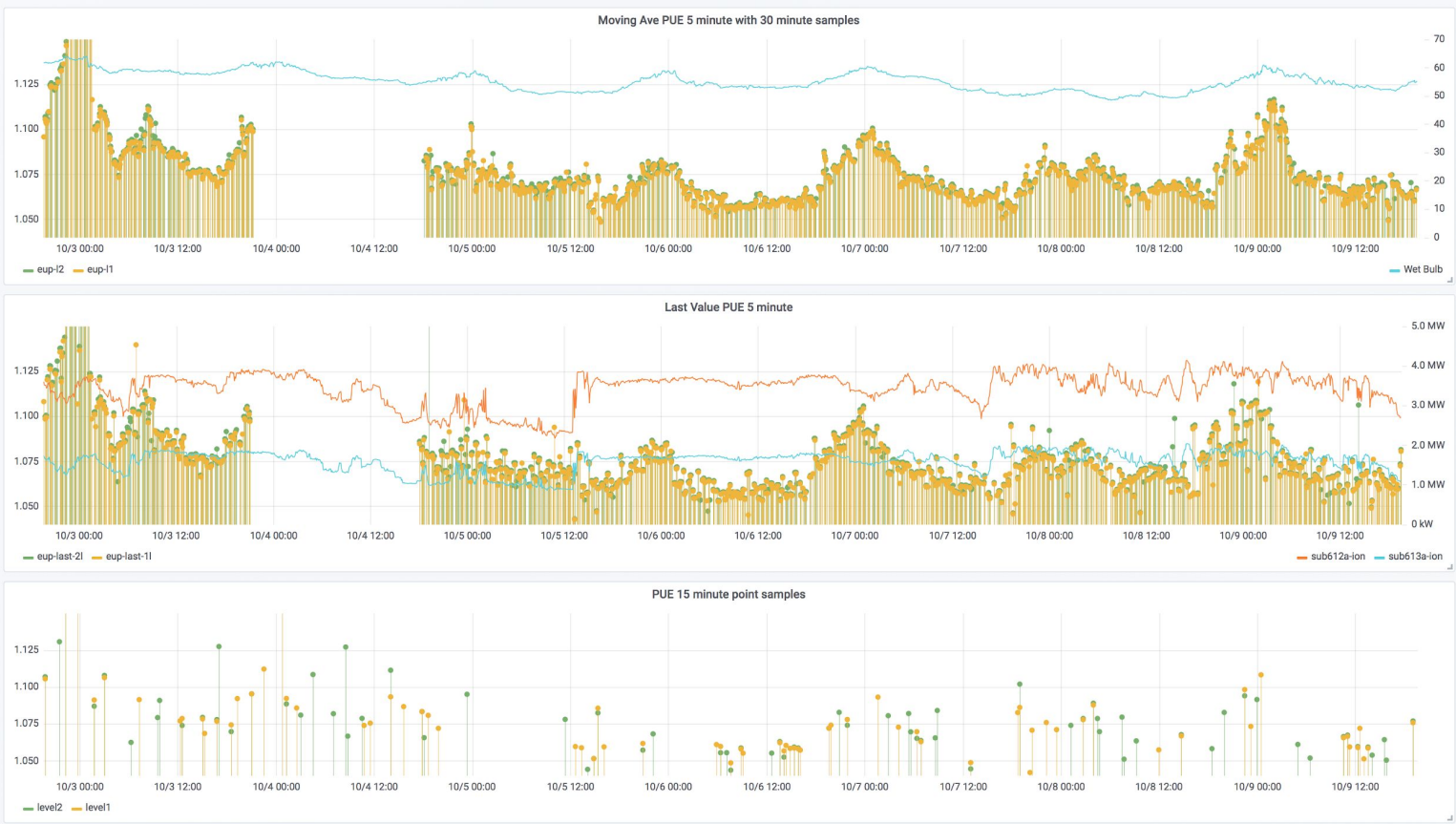
Gathering Cray information:
    Node c5-0c1s6n1 Energy Used: 38186472 Joules
    Node c5-0c1s6n1 CPU Temp: Average: 62.559223301C, High: 68.0C, Low: 51.0C
    Node c5-0c1s6n1 Power: Average: 335 W, High: 353 W, Low: 90 W
...
Checking to see what scratch filesystems where used by the job
Data: **{"index":"edison-lustre", "metric":"category:jobstats|jobid:10703089", "value":"mount", "start":"2018-09-15T00:15:15-0700", "end":"2018-09-16T07:52:40-0700"}**
Job: 10703089 ran on the following filesystems: [u'scratch1', u'scratch3', u'scratch2']

Gathering scratch1 Nodes
Data: **{"fs":"scratch1", "jobid":10703089}**
Job: 10703089 have 720432 hits. Targets for filesystem: scratch1 are: [u'MDT0000', u'OST0000', u'OST0005', u'OST000e', u'OST0014', u'OST0012', u'OST000c', u'OST0002', u'OST0004', u'OST0013', u'OST000f', u'OST0011', u'OST0008', u'OST0017', u'OST0003', u'OST000a', u'OST000b', u'OST0010', u'OST0007', u'OST0015', u'OST0006', u'OST0001', u'OST0009', u'OST0016', u'OST000d']

Conclusion

- With FaaS, which fn is one implementation, we can concentrate on a Data API and not have to worry about everyone storing the data in the same way/format.
- We can insulate users from change.

Quick story. PUE



Thank You