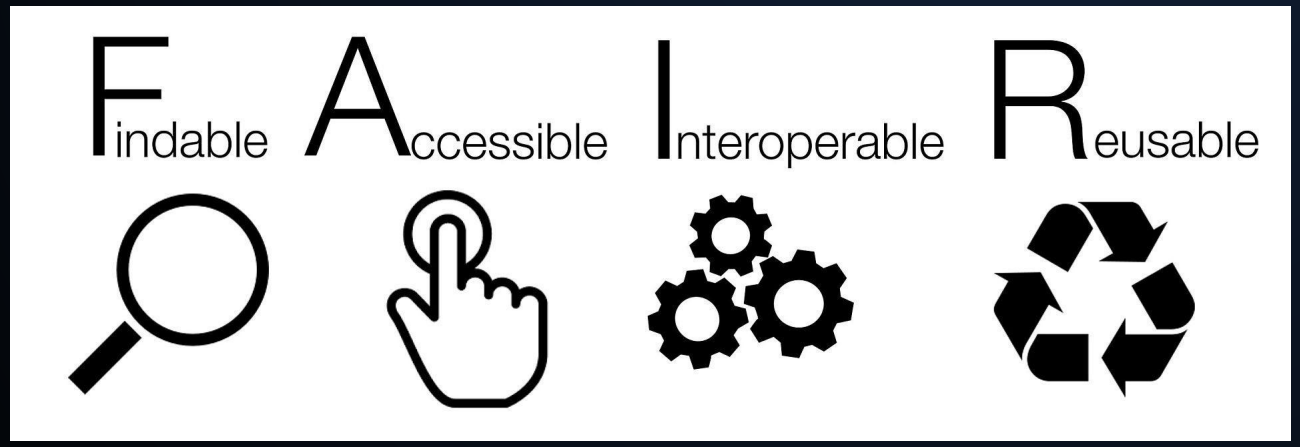
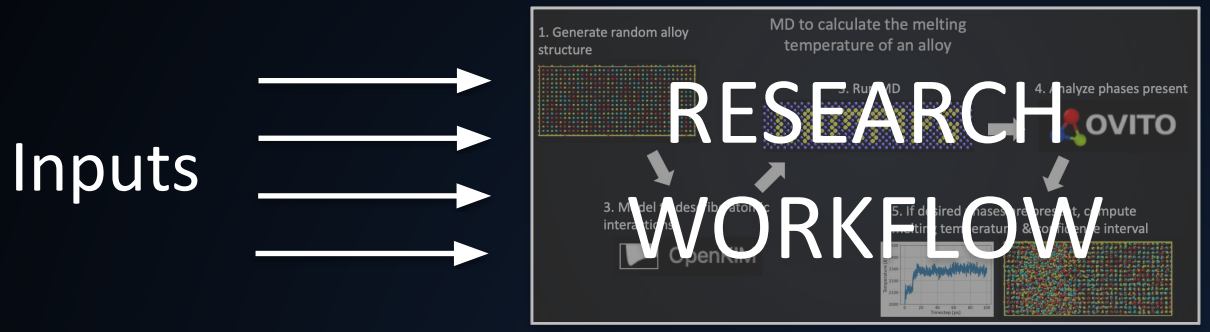
Summary – Sim2l features

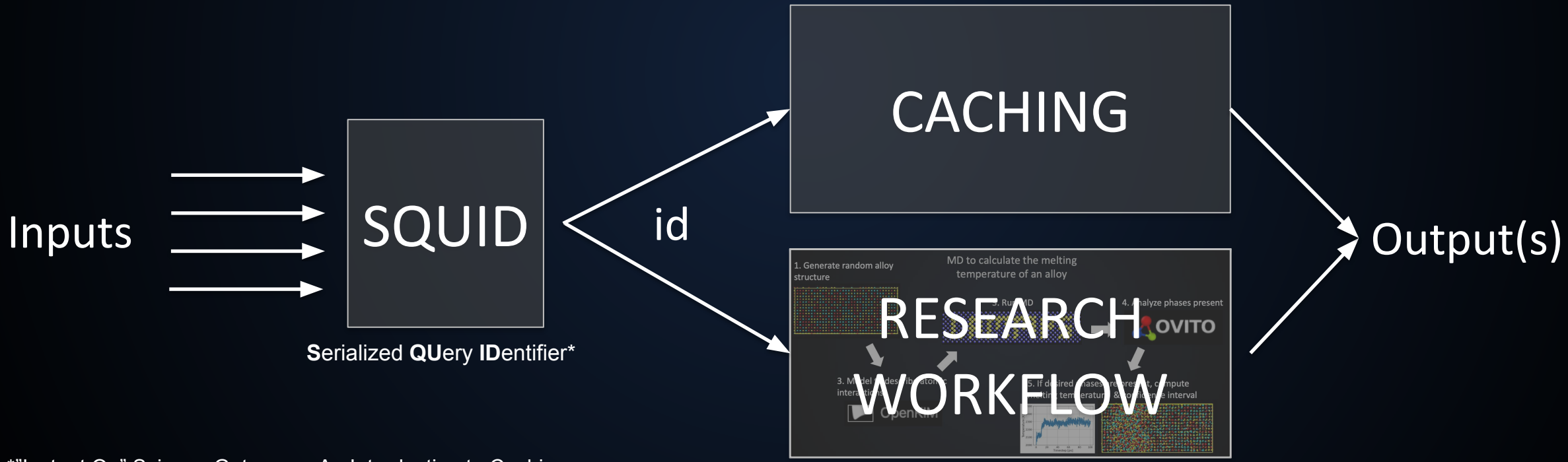
- End to end computational workflow (Repro)
- Published SimTools:
  - Are containerized (Repro)
  - Have DOIs and are indexed by Web of Science & google scholar (F,A)
- Declared and validated inputs and outputs (R, I)
- Services, including metadata, are queryable (F, A, I)
- Automatic result caching (A, R, I)



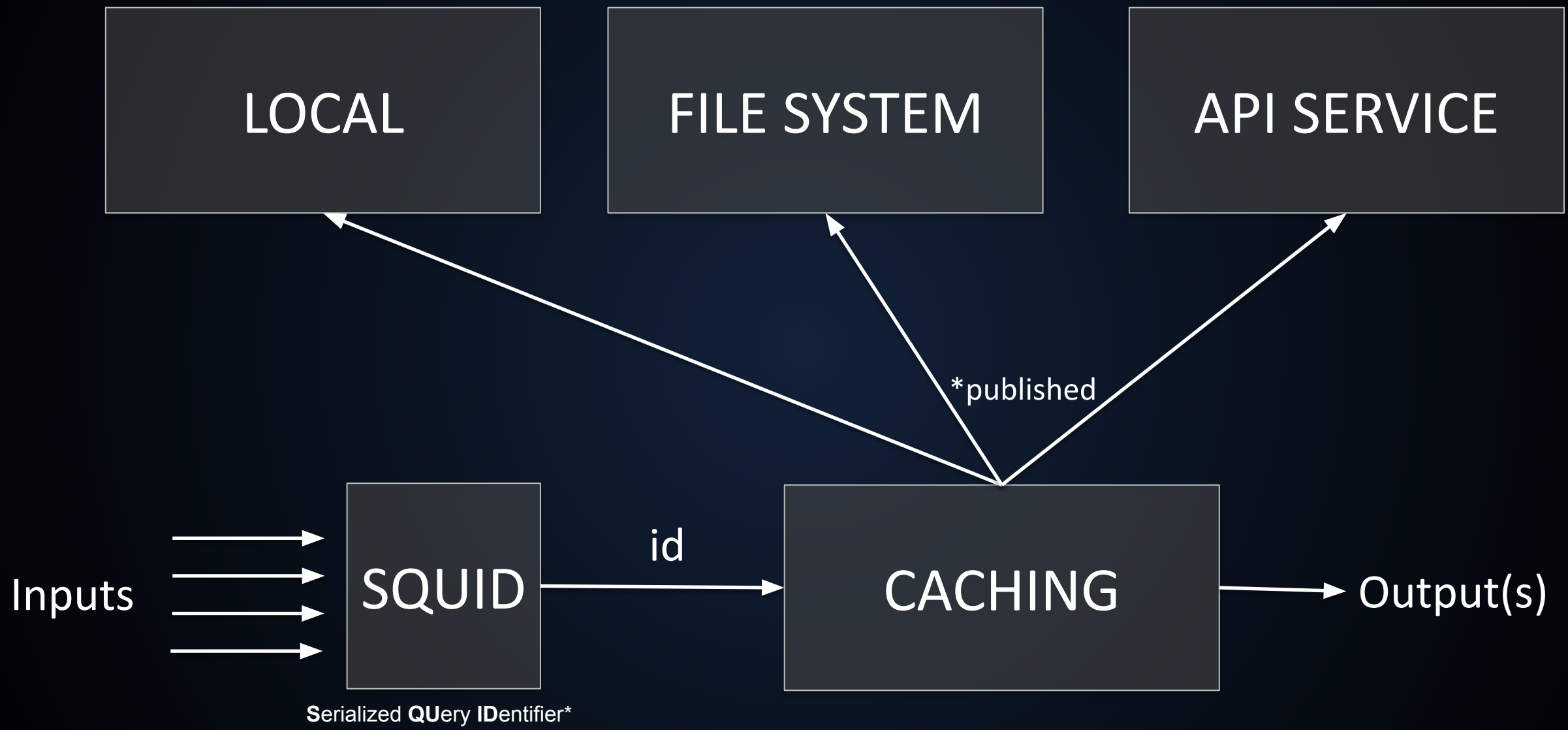
+ Reproducible

Caching





*"Instant On" Science Gateways: An Introduction to Caching Simulation Results and a Path Towards Data Exploration


LOCAL
FILE SYSTEM
API SERVICE
*published
Inputs
SQUID
id
CACHING
Output(s)
Serialized QUery IDentifier*

Configuration

```
sessionid 1891257
results_directory /home/nanohub/denphi/data/results/1891257
filexfer_port 9018
filexfer_cookie [redacted]
filexfer_decoration

nanovis_server render14.nanohub.org:2001
molvis_server render14.nanohub.org:2020
vtkvis_server render14.nanohub.org:2010
vmdmds_server render14.nanohub.org:2018
geovis_server render14.nanohub.org:2015
fury_server fury-server.hubzero.org
fury_data_prefix /srv/nanohub

application_name "Quantum Dot Lab"
version test
session_token [redacted]
hub_name nanoHUB
hub_url https://nanohub.org
hub_template NaN
job_protocol submit
cache_hosts instanton2.nanohub.org:8088
cache_user ionhelper
cache_write_host instanton2.nanohub.org:8086
squiddb http://instanton2.nanohub.org:5000/api/v1
```

SimToolCache.py

```
if squiddb.startswith('https:') or  squiddb.startswith('http:'):
    self.ds_handler = WSDataStore
else:
    self.ds_handler = FileDataStore
```

FILE SYSTEM

API SERVICE

Caching

nanoHUB jupyter Untitled42 (unsaved changes) Submit a ticket Terminate Session

File Edit View Insert Cell Kernel Widgets Help Snippets Trusted Python 3

Run Code Appmode Validate

In [4]:

```
1 import simtool as st
2 simToolName = "st4pnjunction"
3 nb = st.utils.searchForSimTool(simToolName)
4 nb
```

Out[4]:

```
{'notebookPath': '/apps/st4pnjunction/r10/simtool/st4pnjunction.ipynb',
 'simToolName': 'st4pnjunction',
 'simToolRevision': 'r10',
 'published': True}
```

In [6]:

```
1 inputs = st.utils.getSimToolInputs(nb)
2 #inputs
```

In [*]:

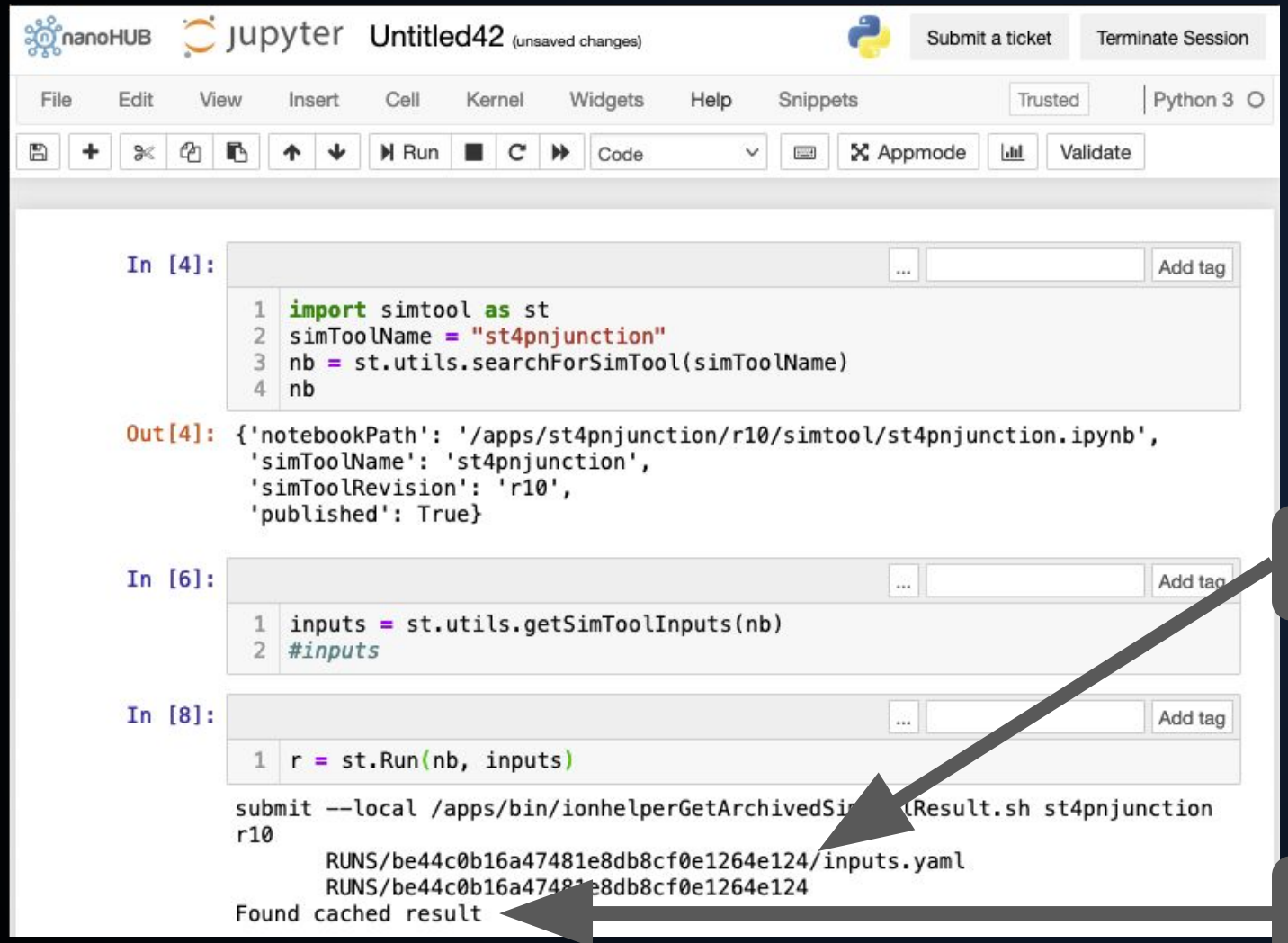
```
1 r = st.Run(nb, inputs)
```

```
submit --local /apps/bin/ionhelperGetArchivedSimToolResult.sh st4pnjunction
r10
       RUNS/c137e54cdb6e47a7a5be42b223884953/inputs.yaml
       RUNS/c137e54cdb6e47a7a5be42b223884953
submit --local /apps/bin/ionhelperRunSimTool.sh st4pnjunction r10
       RUNS/c137e54cdb6e47a7a5be42b223884953/inputs.yaml
```

```
Input Notebook:  /apps/st4pnjunction/r10/simtool/st4pnjunction.ipynb
Output Notebook: st4pnjunction.ipynb
Executing:  63%|???????   | 29/46 [00:30<00:16,  1.01cell/s]
```

Running new Sim2L

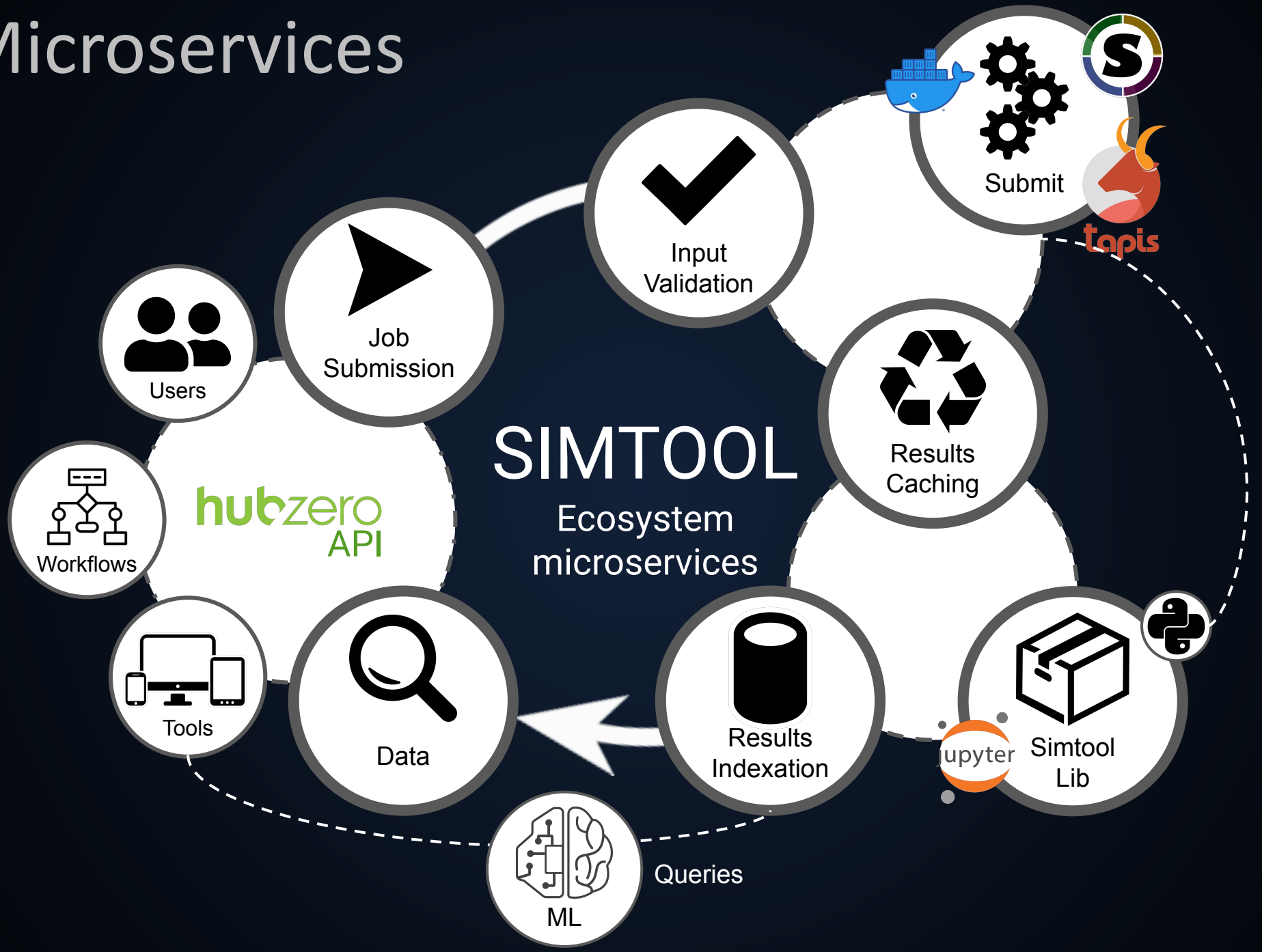
Caching



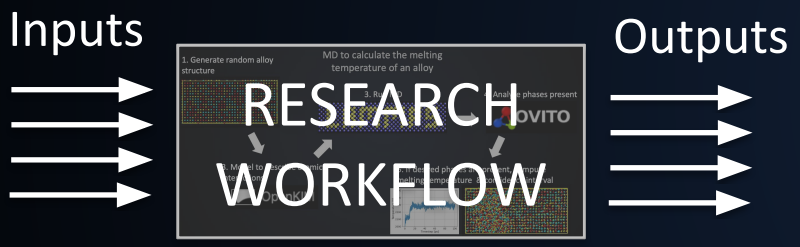
Local cache

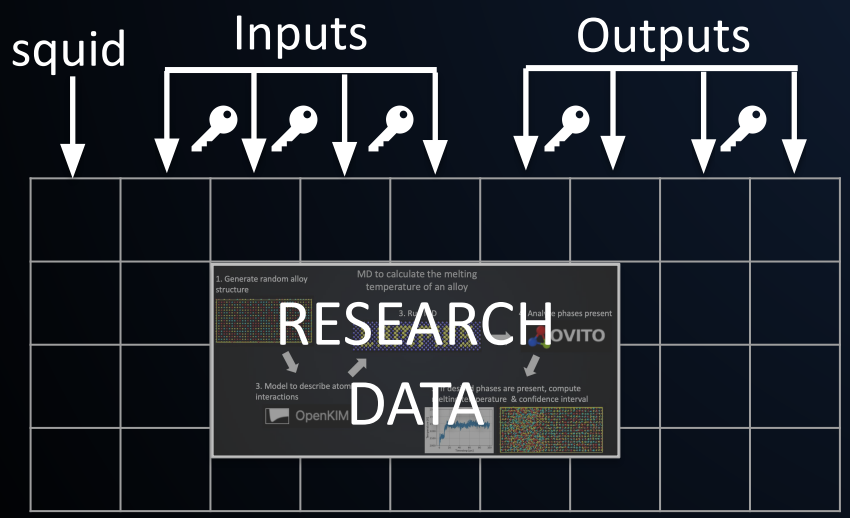
Cached results

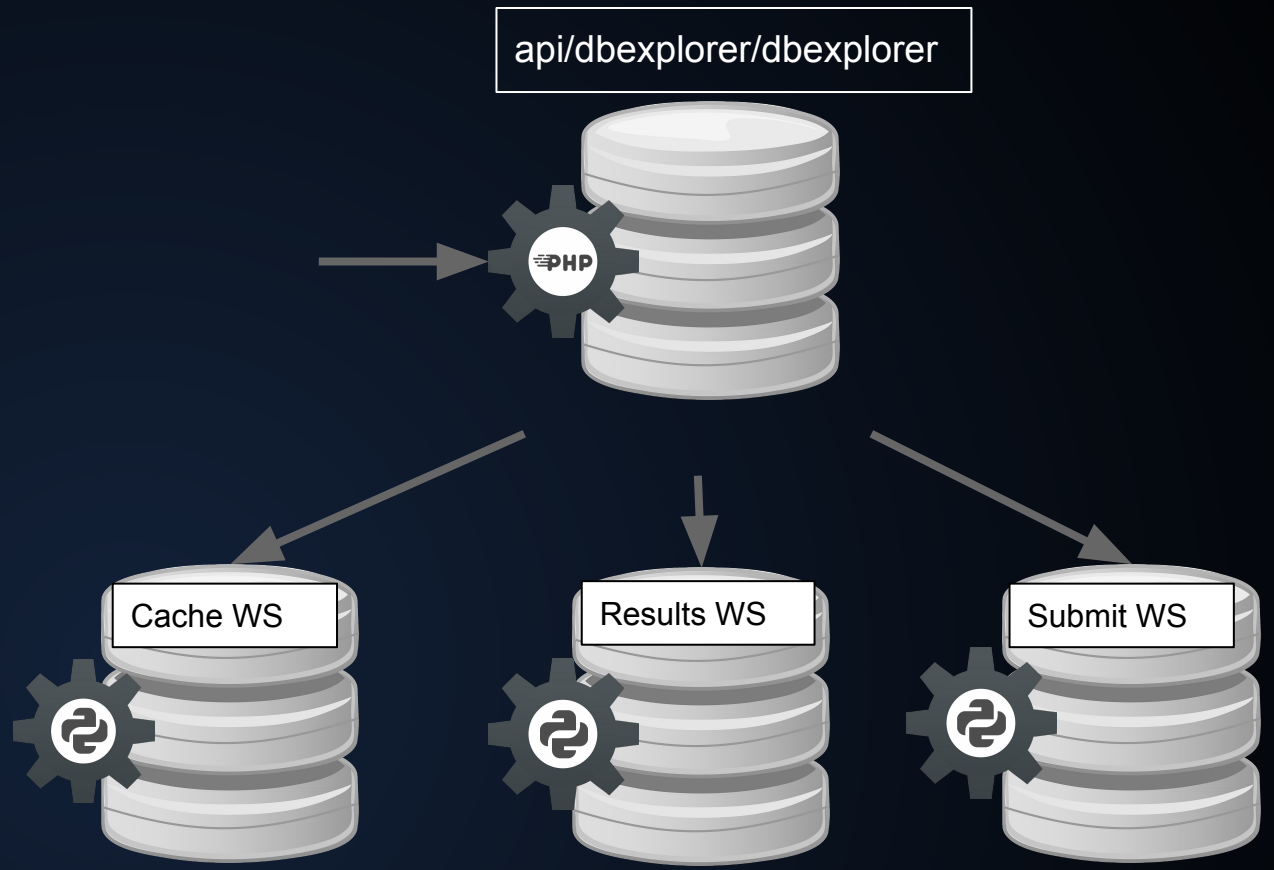
Microservices



Microservices







https://nanohub.org/api/dbexplorer/dbexplorer/
https://nanohub.org/developer/api/endpoint/dbexplorer

nanoHUB REMOTE

https://pypi.org/project/nanohub-remote/

NanoHUB remote - setup

Import Libraries

- nanohub.remote is required to access nanohub web services
- pandas, plotly and numpy are used to visualize data

In [1]:
```
import nanohub.remote as nr
import pandas as pd
import json
from plotly.graph_objects import FigureWidget

```

Authentication Data

If the services are going to be accessed outside of nanohub, a proper authentication is required

- to get client id and secret, create a web application (https://nanohub.org/developer/api/applications/new), use "https://127.0.0.1" as Redirect URL
- to get username and password, register on nanohub.org (https://nanohub.org/register/)

In [2]:
```
auth_data = {
  'client_id': 'XXXXXXXX',
  'client_secret': 'XXXXXXXX',
  'grant_type': 'password',
  'username': 'XXXXXXXX',
  'password': 'XXXXXXXX'
}
```

Is posible to reuse current nanoHUB session by reading its information from the resources file

In [3]:
```
import os
auth_data = {
    'grant_type' : 'tool',
}
with open(os.environ["SESSIONDIR"]+"/resources") as file:
    lines = [line.split(" ", 1) for line in file.readlines()]
    properties = {line[0].strip(): line[1].strip() for line in lines if len(lin
    auth_data["sessiontoken"] = properties["session_token"]
    auth_data["sessionnum"] = properties["sessionid"]
```

Import nanohub-remote

Regular Authentication

Reuse jupyter session auth

Querying Tools and parameters

Web Services Session

Using the authentication data nanohub-remote creates a Generic session (nr.Tools(auth_data))

In [4]:
```
session = nr.Session(auth_data)
```

Create a session
Session(auth_data, url=HZ_API_ENDPOINT)

To view all available tools on the results database use the tools end point

In [5]:
```
req_json = session.requestPost('dbexplorer/dbexplorer/tools?simtool=true')
req_json = req_json.json()
pd.DataFrame([p for p in req_json["results"]])
```

List all published simtools

Out[5]:

	last_version	tool_id	total_versions
0	r9	introtosimtools	2
1	r13	meltrccas	1
2	r34	meltingkim	3
3	r21	mdsandbox	1
4	r39	meltheas	3
5	r9	st4pnjunction	2

Query tool tool inputs and outpus

Given an specific tool is posible to query their schema inputs

In [6]:
```
SIM2L = "st4pnjunction"
req_json = session.requestPost('dbexplorer/dbexplorer/tool_detail', data={'tool':SIM2L, 'simtool':True})
req_json = req_json.json()
```

Print its inputs:

In [7]:
```
pd.DataFrame(req_json['results'][0][SIM2L]["input"]).transpose()
```

View parameters

As well as its outputs:

In [9]:
```
pd.DataFrame(req_json['results'][0][SIM2L]["output"]).transpose()
```

Out[9]:

	description	label	type
Parameters	Parameters	Parameters	string
Efn	Efn	Efn	dict
Efp	Efp	Efp	dict
Hole Density	Hole Density	Hole Density	dict
Electron Density	Electron Density	Electron Density	dict
Equilibrium Hole Density	Equilibrium Hole Density	Equilibrium Hole Density	dict
Equilibrium Electron Density	Equilibrium Electron Density	Equilibrium Electron Density	dict
Charge Density	Charge Density	Charge Density	dict
Equilibrium Net Charge Density	Equilibrium Net Charge Density	Equilibrium Net Charge Density	dict
Equilibrium Ei	Equilibrium Ei	Equilibrium Ei	dict
Equilibrium Potential	Equilibrium Potential	Equilibrium Potential	dict
Equilibrium Ec	Equilibrium Ec	Equilibrium Ec	dict
Ec	Ec	Ec	dict
Equilibrium Ev	Equilibrium Ev	Equilibrium Ev	dict
Ev	Ev	Ev	dict
Equilibrium Electric Field	Equilibrium Electric Field	Equilibrium Electric Field	dict
Electric Field	Electric Field	Electric Field	dict
Equilibrium Recombination Rate	Equilibrium Recombination Rate	Equilibrium Recombination Rate	dict
Recombination Rate	Recombination Rate	Recombination Rate	dict
CV Characteristic	CV Characteristic	CV Characteristic	dict

Querying - squid

Results Queries,

To request data it is necesary to define filters based on the inputs, and specify the results (outputs) that are going to be returned

- The following query search for data for the tool 'polymod' that contain a number of monomer types (num_zmatrices) equal to 2, and return the Bond scale factor additionally, all queries return the unique identifier (squid) of that result

In [11]:

```
search = {
    'tool':SIM2L,
    'simtool':True,
    'filters':json.dumps([
        {'field':'squid','operation':'=','value':'st4pnjunction/r9/a0fe446ed8b54a50a24b6ff8151a5b1151520ce3'},
    ]),
    'results':json.dumps([
        'input.p_len',
        'input.n_len',
        'input.i_len',
        'output.Equilibrium Potential',
        'output.Equilibrium Ei'
    ]),
}
req_json = session.requestPost('dbexplorer/dbexplorer/search', data=search)
req_json = req_json.json()
results = req_json['results']
pd.DataFrame(results)
```

Out[11]:

	input.i_len	input.n_len	input.p_len	output.Equilibrium Ei	output.Equilibrium Potential	squid
0	0	6	3.1	{'function': [0.315636008547, 0.315636008533, ...	{'delta': [0, 1.4000023362826e-11, 2.080000061...	st4pnjunction/r9/a0fe446ed8b54a50a24b6ff8151a5...

Squid as filter

inputs

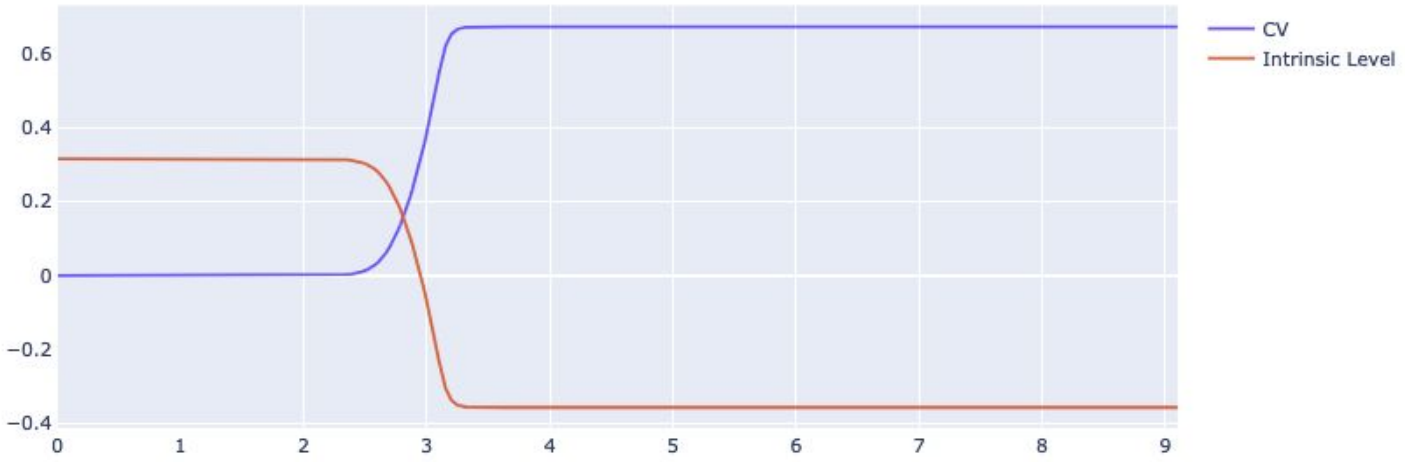
outputs

Querying - Plotting Single result

outputs

In [12]:

```
Potential = results[0]["output.Equilibrium Potential"]
Intrinsic = results[0]["output.Equilibrium Ei"]
FigureWidget(data =[
    dict(
       type="scatter",
        x = Potential["position"],
        y = Potential["delta"],
        name="CV"
    ),
    dict(
       type="scatter",
        x = Intrinsic["position"],
        y = Intrinsic["function"],
        name="Intrinsic Level"
    ),
])
```



Querying - Filters

- The following query request the IV charateristics and doping from pntoy devices based on multiple filters

In [13]:

```
N_LEN = results[0]["input.n_len"]
I_LEN = results[0]["input.i_len"]
VOLTAGE = [0,0.6]
TEMPERATURE = [300,300]
MATERIALP = "Si"
IMPURITY = "false"

search = {
    'tool':SIM2L,
    'simtool':True,
    'filters':json.dumps([
        {'field':'input.n_len','value':N_LEN,'operation':'=='},
        {'field':'input.temperature','value':TEMPERATURE[0],'operation':'>='},
        {'field':'input.temperature','value':TEMPERATURE[1],'operation':'<='},
        {'field':'input.i_len','value':I_LEN,'operation':'=='},
        {'field':'input.materialp','value':MATERIALP,'operation':'=='},
        {'field':'input.impurity','value':IMPURITY,'operation':'=='},
        {'field':'input.voltage','value':0,'operation':'=='},
    ]),
    'results':json.dumps([
        'input.Na',
        'input.Nd',
        'input.n_len',
        'input.i_len',
        'input.p_len',
        'output.Parameters',
        'output.IV Characteristic',
    ])
}
req_json = session.requestPost('dbexplorer/dbexplorer/search', data=search)
req_json = req_json.json()
pd.DataFrame(req_json['results'])
```

Multiple filters only indexable parameters

inputs

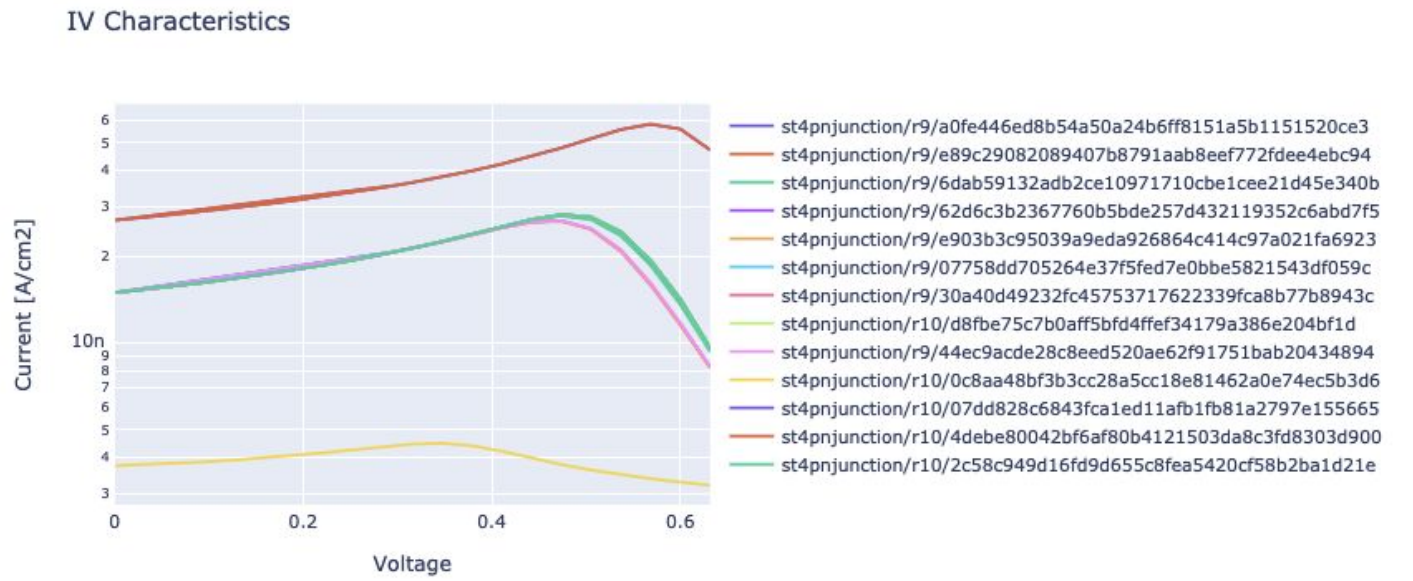
outputs

Querying - Plotting Multiple results

In [14]:

```
layout1 = {
    'title' : "IV Characteristics",
    'xaxis':{
        'title' : 'Voltage',
    },
    'yaxis':{
        'title' : 'Current [A/cm2]',
        'type' : 'log',
    },
}
traces = []
for res in req_json['results']:
    traces.append({
        'type': 'scatter',
        'name': res['squid'],
        'x':res['output.IV Characteristic']['voltage'],
        'y':res['output.IV Characteristic']['function'],
        'text': "N:" + str(res['input.n_len']) + ", P:" + str(res['input.p_len']) + "<BR>" + "Na:" + "{:.2e}".
    })
fig = FigureWidget(traces, layout1)
fig
```

outputs



Querying - Interactive Apps