CYBERINSIGHT Total Cost of Operation Tutorial

by Julia D. Harrison, Associate Director, Center for High Performance Computing, University of Utah

10/11/2021

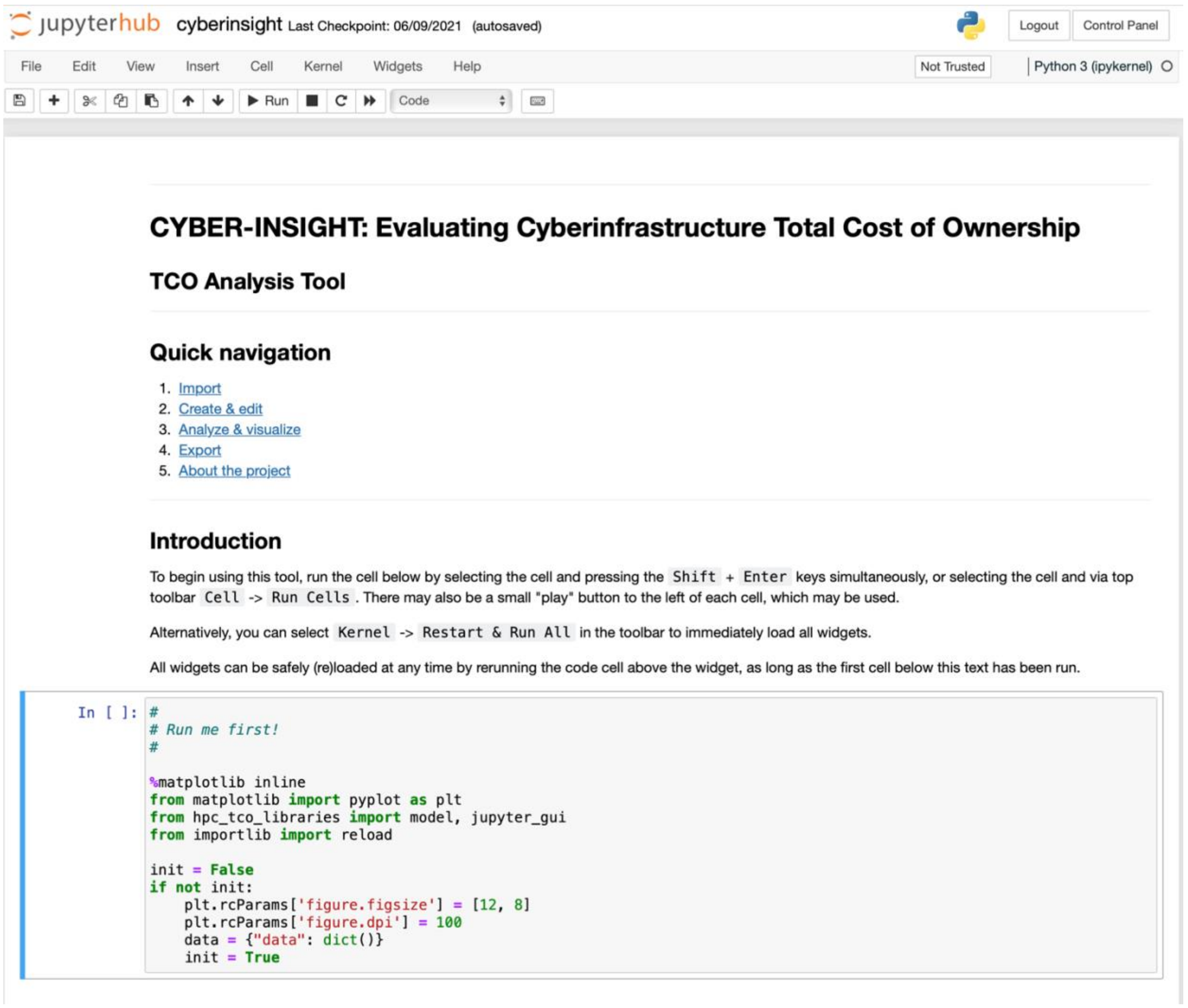
For questions, feedback or bug reports, please email cyberinsight-feedback@lists.utah.edu

Index:

Part 1. Start the tool up.

a. Navigate to the URL: https://cyberinsight.chpc.utah.edu/
b. Then click on Run the tool
c. Click on Sign in with CILogon
d. Select your Identity Provider (Your Institution, EDUCAUSE, National Science Foundation, etc.) Click on “Log On”.
e. Authenticate with your credentials associated with the Identity Provider you selected above.
f. This will take you the the cyberinsight jupyter hub.
g. Click on the “hpc-tco-tool”
h. Click on “cyberinsight.ipnb” to start the tool.

For questions, feedback or bug reports, please email cyberinsight-feedback@lists.utah.edu


jupyterhub cyberinsight Last Checkpoint: 06/09/2021 (autosaved)
Logout
Control Panel
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

CYBER-INSIGHT: Evaluating Cyberinfrastructure Total Cost of Ownership

TCO Analysis Tool

Quick navigation

1. Import
2. Create & edit
3. Analyze & visualize
4. Export
5. About the project

Introduction

To begin using this tool, run the cell below by selecting the cell and pressing the `Shift` + `Enter` keys simultaneously, or selecting the cell and via top toolbar `Cell` -> `Run Cells`. There may also be a small "play" button to the left of each cell, which may be used.

Alternatively, you can select `Kernel` -> `Restart & Run All` in the toolbar to immediately load all widgets.

All widgets can be safely (re)loaded at any time by rerunning the code cell above the widget, as long as the first cell below this text has been run.

```
In [ ]: #
        # Run me first!
        #

        %matplotlib inline
        from matplotlib import pyplot as plt
        from hpc_tco_libraries import model, jupyter_gui
        from importlib import reload

        init = False
        if not init:
            plt.rcParams['figure.figsize'] = [12, 8]
            plt.rcParams['figure.dpi'] = 100
            data = {"data": dict()}
            init = True
```

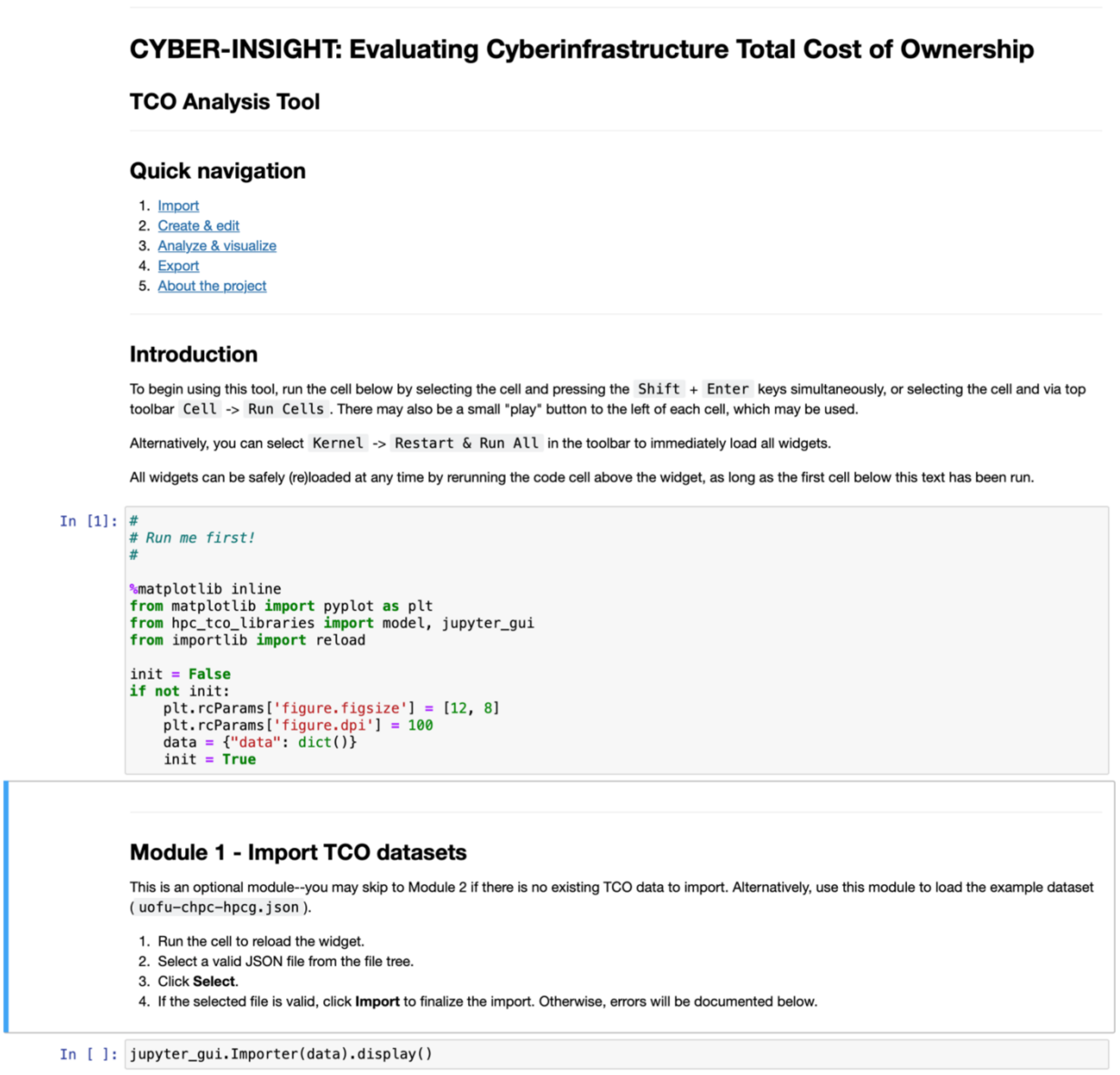
Part 2. Run an example provided by the University of Utah

1. Run the first cell to set things up. This cell has a “Run me first!” at the top as a comment. Select the cell and press the Shift + Enter keys simultaneously, or by selecting the cell and via top toolbar Cell -> Run Cells.

   Alternatively, you can select Kernel -> Restart & Run All in the toolbar to immediately load all widgets.

   All widgets can be safely (re)loaded at any time by rerunning the code cell above the widget, as long as the first cell below this text has been run.

   When you run the first cell, Module 1 – is highlighted.



CYBER-INSIGHT: Evaluating Cyberinfrastructure Total Cost of Ownership

TCO Analysis Tool

Quick navigation

1. Import
2. Create & edit
3. Analyze & visualize
4. Export
5. About the project

Introduction

To begin using this tool, run the cell below by selecting the cell and pressing the `Shift` + `Enter` keys simultaneously, or selecting the cell and via top toolbar `Cell` -> `Run Cells`. There may also be a small "play" button to the left of each cell, which may be used.

Alternatively, you can select `Kernel` -> `Restart & Run All` in the toolbar to immediately load all widgets.

All widgets can be safely (re)loaded at any time by rerunning the code cell above the widget, as long as the first cell below this text has been run.

```
In [1]: #
        # Run me first!
        #

        %matplotlib inline
        from matplotlib import pyplot as plt
        from hpc_tco_libraries import model, jupyter_gui
        from importlib import reload

        init = False
        if not init:
            plt.rcParams['figure.figsize'] = [12, 8]
            plt.rcParams['figure.dpi'] = 100
            data = {"data": dict()}
            init = True
```

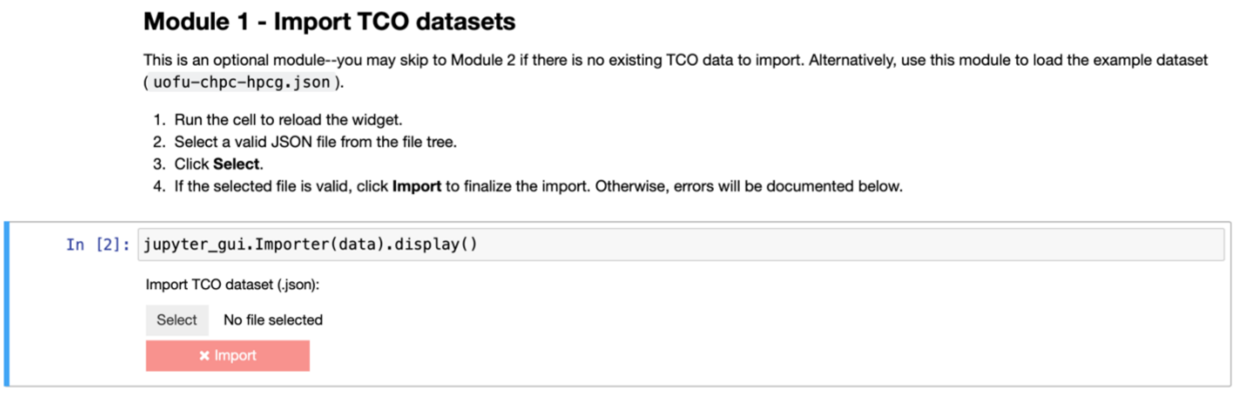
Module 1 - Import TCO datasets

This is an optional module--you may skip to Module 2 if there is no existing TCO data to import. Alternatively, use this module to load the example dataset (`uofu-chpc-hpcg.json`).

1. Run the cell to reload the widget.
2. Select a valid JSON file from the file tree.
3. Click **Select**.
4. If the selected file is valid, click **Import** to finalize the import. Otherwise, errors will be documented below.

```
In [ ]: jupyter_gui.Importer(data).display()
```

2. MODULE 1. For this section of the tutorial, we will have you import an example data set.
   2.1. Run the cell in the Module 1 section to reload the widget. Click on **Select**.



Module 1 - Import TCO datasets

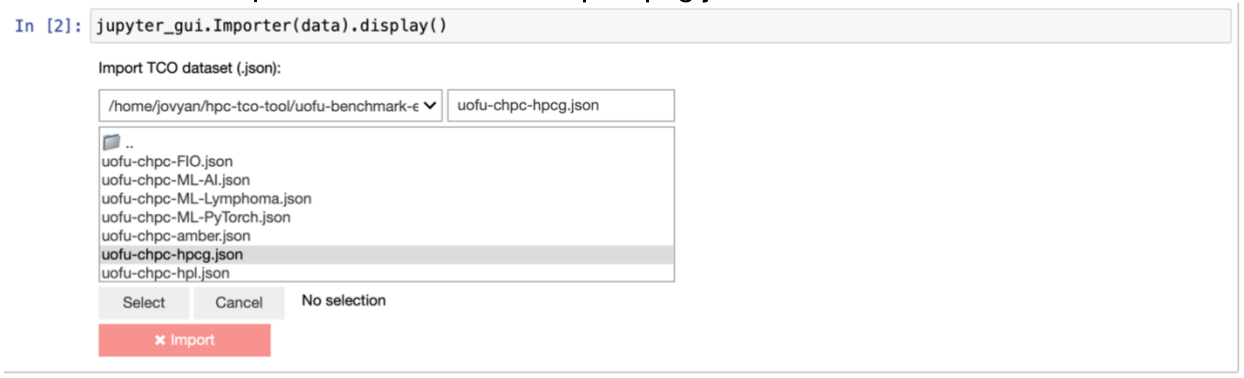
This is an optional module--you may skip to Module 2 if there is no existing TCO data to import. Alternatively, use this module to load the example dataset (`uofu-chpc-hpcg.json`).

1. Run the cell to reload the widget.
2. Select a valid JSON file from the file tree.
3. Click **Select**.
4. If the selected file is valid, click **Import** to finalize the import. Otherwise, errors will be documented below.

```
In [2]: jupyter_gui.Importer(data).display()
```

Import TCO dataset (.json):

2.2. Select a valid JSON file from the file tree. To run our example, choose “uofu-benchmark-examples” and then “uofu-chpc-hpcg.json”.



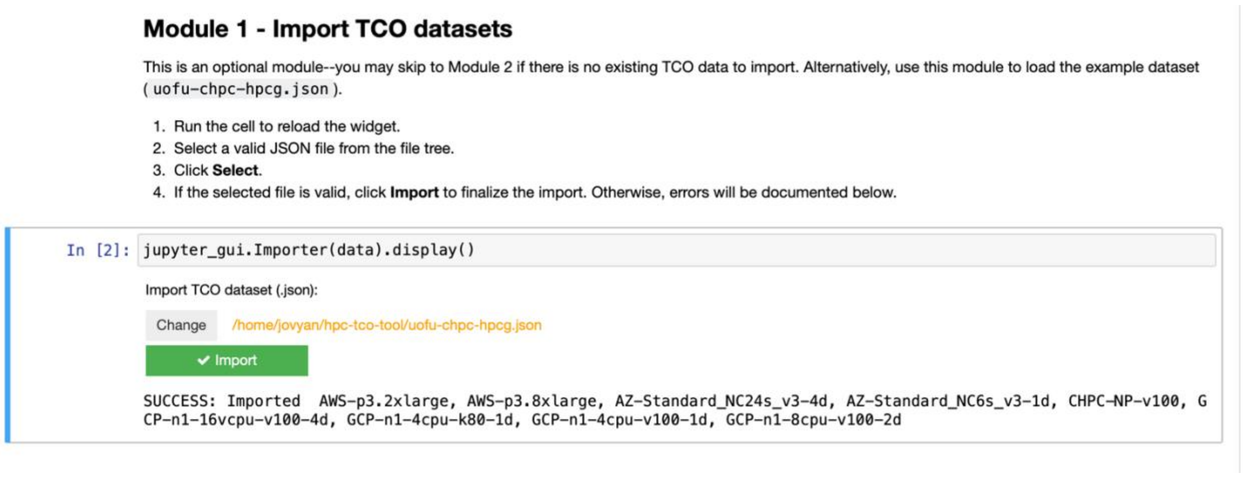
```
In [2]: jupyter_gui.Importer(data).display()
```

Import TCO dataset (.json):

/home/jovyan/hpc-tco-tool/uofu-benchmark-e | uofu-chpc-hpcg.json

..
uofu-chpc-FIO.json
uofu-chpc-ML-AI.json
uofu-chpc-ML-Lymphoma.json
uofu-chpc-ML-PyTorch.json
uofu-chpc-amber.json
uofu-chpc-hpcg.json
uofu-chpc-hpl.json

2.3. Click **Select** again. The full path to the file name is displayed.
If the selected file is valid, click **Import** to finalize the import. Otherwise, errors will be displayed in the cell. If you selected the example file above, you should see the following:



Module 1 - Import TCO datasets

This is an optional module--you may skip to Module 2 if there is no existing TCO data to import. Alternatively, use this module to load the example dataset (`uofu-chpc-hpcg.json`).

1. Run the cell to reload the widget.
2. Select a valid JSON file from the file tree.
3. Click **Select**.
4. If the selected file is valid, click **Import** to finalize the import. Otherwise, errors will be documented below.

```
In [2]: jupyter_gui.Importer(data).display()
```

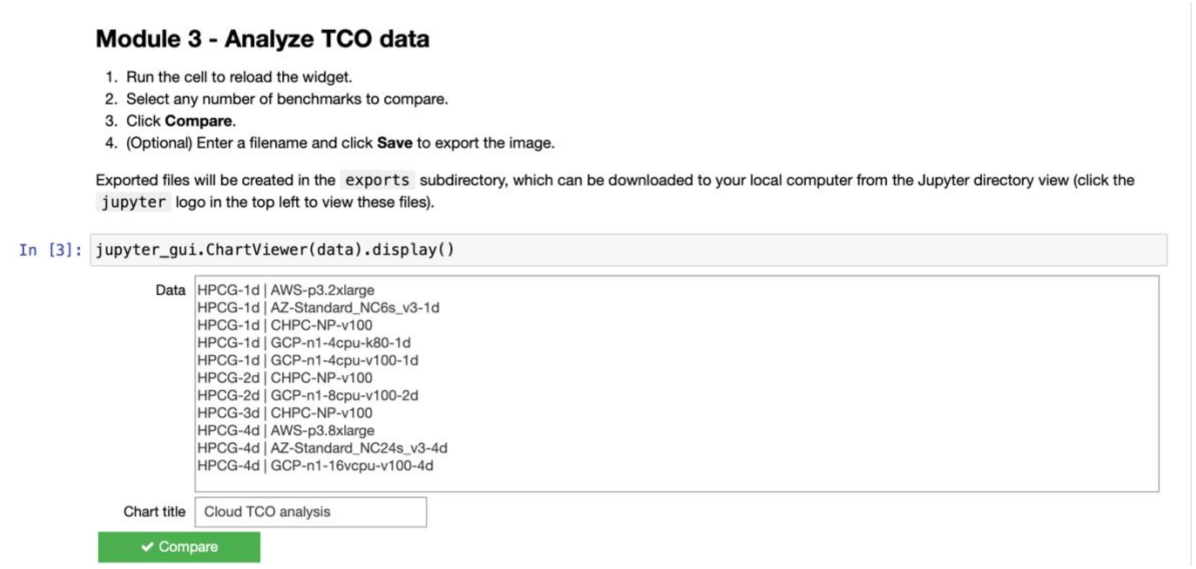
Import TCO dataset (.json):

/home/jovyan/hpc-tco-tool/uofu-chpc-hpcg.json

```
SUCCESS: Imported  AWS-p3.2xlarge, AWS-p3.8xlarge, AZ-Standard_NC24s_v3-4d, AZ-Standard_NC6s_v3-1d, CHPC-NP-v100, G
CP-n1-16vcpu-v100-4d, GCP-n1-4cpu-k80-1d, GCP-n1-4cpu-v100-1d, GCP-n1-8cpu-v100-2d
```

3. MODULE 2. (Not needed for running an example) See Part 3. Creating your own datasets. Skip running this module and go to MODULE 3.

4. MODULE 3 – Analyze TCO data
   4.1. Run the cell to reload the widget.



Module 3 - Analyze TCO data

1. Run the cell to reload the widget.
2. Select any number of benchmarks to compare.
3. Click **Compare**.
4. (Optional) Enter a filename and click **Save** to export the image.

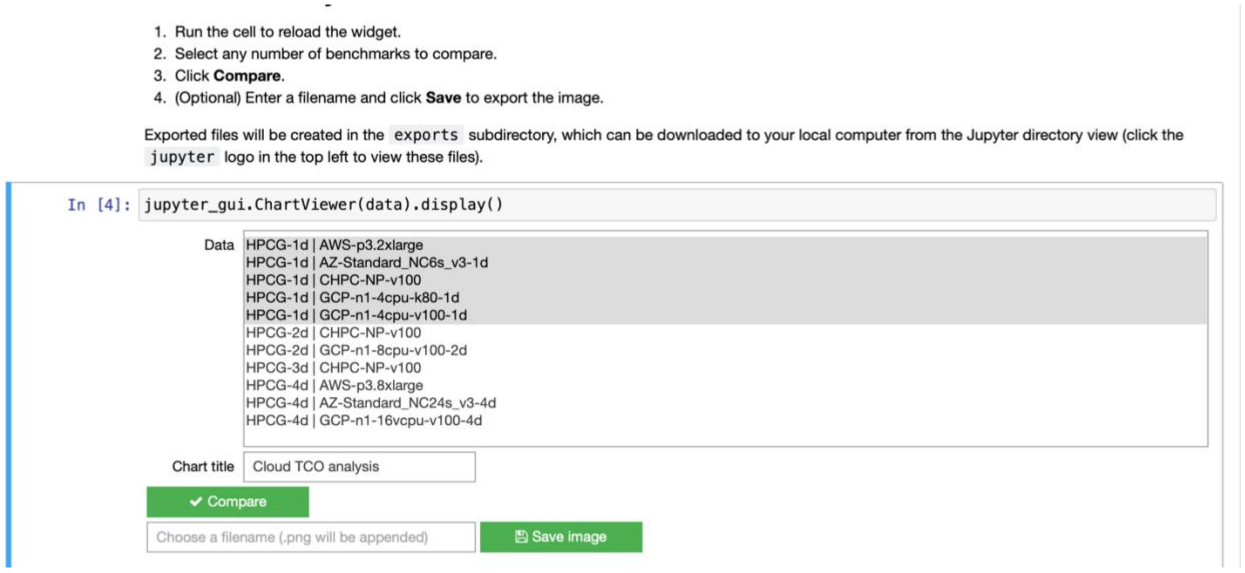
Exported files will be created in the `exports` subdirectory, which can be downloaded to your local computer from the Jupyter directory view (click the `jupyter` logo in the top left to view these files).

```
In [3]: jupyter_gui.ChartViewer(data).display()
```

Data
HPCG-1d | AWS-p3.2xlarge
HPCG-1d | AZ-Standard_NC6s_v3-1d
HPCG-1d | CHPC-NP-v100
HPCG-1d | GCP-n1-4cpu-k80-1d
HPCG-1d | GCP-n1-4cpu-v100-1d
HPCG-2d | CHPC-NP-v100
HPCG-2d | GCP-n1-8cpu-v100-2d
HPCG-3d | CHPC-NP-v100
HPCG-4d | AWS-p3.8xlarge
HPCG-4d | AZ-Standard_NC24s_v3-4d
HPCG-4d | GCP-n1-16vcpu-v100-4d

Chart title: Cloud TCO analysis

4.2. Select any number of benchmarks to compare in the Data window. For this tutorial, the benchmarks are named for the number of GPU devices used in the benchmark. Pick all of the 1-d benchmarks.



1. Run the cell to reload the widget.
2. Select any number of benchmarks to compare.
3. Click **Compare**.
4. (Optional) Enter a filename and click **Save** to export the image.

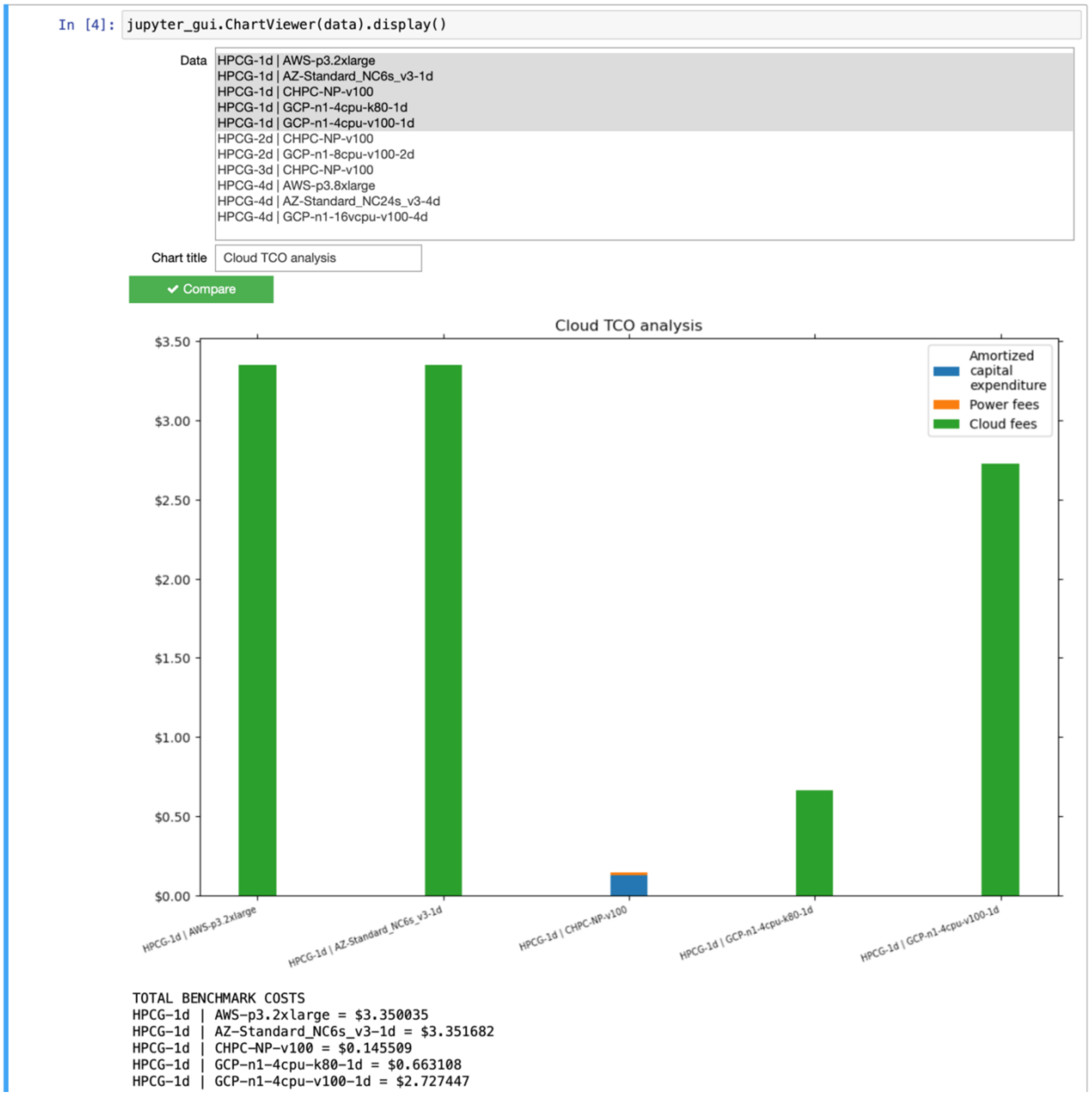
Exported files will be created in the `exports` subdirectory, which can be downloaded to your local computer from the Jupyter directory view (click the `jupyter` logo in the top left to view these files).

```
In [4]: jupyter_gui.ChartViewer(data).display()
```

Data
HPCG-1d | AWS-p3.2xlarge
HPCG-1d | AZ-Standard_NC6s_v3-1d
HPCG-1d | CHPC-NP-v100
HPCG-1d | GCP-n1-4cpu-k80-1d
HPCG-1d | GCP-n1-4cpu-v100-1d
HPCG-2d | CHPC-NP-v100
HPCG-2d | GCP-n1-8cpu-v100-2d
HPCG-3d | CHPC-NP-v100
HPCG-4d | AWS-p3.8xlarge
HPCG-4d | AZ-Standard_NC24s_v3-4d
HPCG-4d | GCP-n1-16vcpu-v100-4d

Chart title: Cloud TCO analysis

4.3. Click “Compare”.
A graph of the costs of the selected benchmarks are displayed.



```
In [4]: jupyter_gui.ChartViewer(data).display()
```

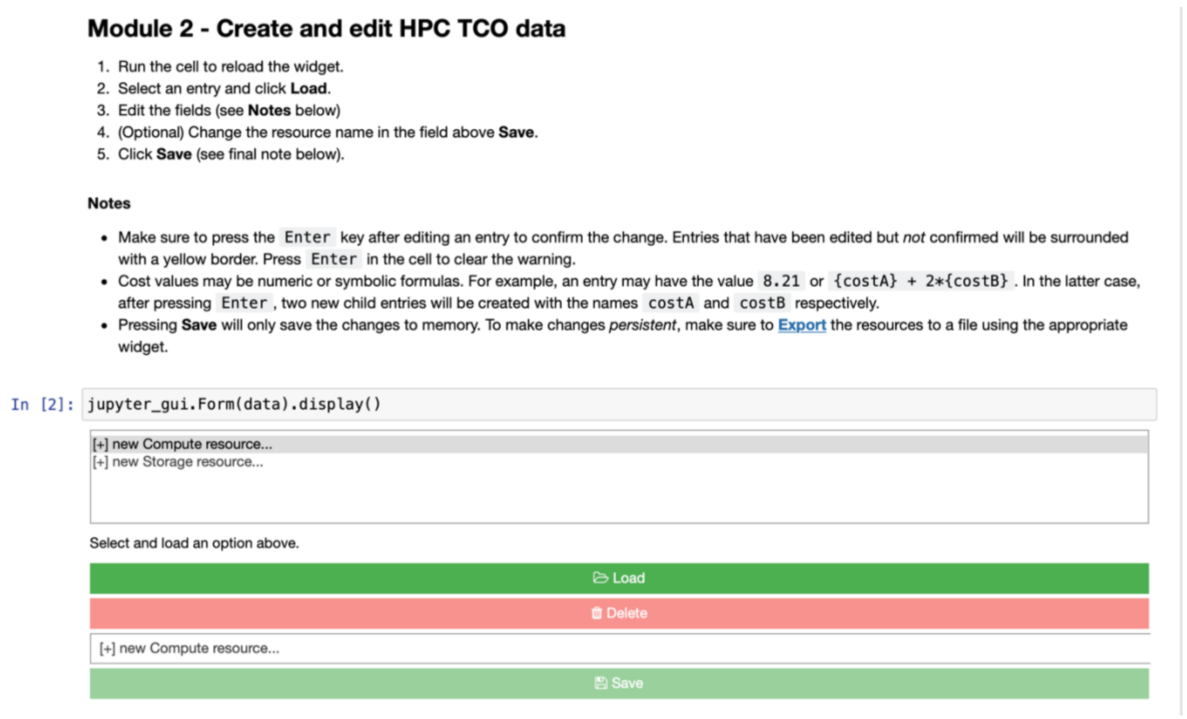
Data
HPCG-1d | AWS-p3.2xlarge
HPCG-1d | AZ-Standard_NC6s_v3-1d
HPCG-1d | CHPC-NP-v100
HPCG-1d | GCP-n1-4cpu-k80-1d
HPCG-1d | GCP-n1-4cpu-v100-1d
HPCG-2d | CHPC-NP-v100
HPCG-2d | GCP-n1-8cpu-v100-2d
HPCG-3d | CHPC-NP-v100
HPCG-4d | AWS-p3.8xlarge
HPCG-4d | AZ-Standard_NC24s_v3-4d
HPCG-4d | GCP-n1-16vcpu-v100-4d
Chart title
Cloud TCO analysis
Compare
Cloud TCO analysis
$3.50
$3.00
$2.50
$2.00
$1.50
$1.00
$0.50
$0.00
Amortized capital expenditure
Power fees
Cloud fees
HPCG-1d | AWS-p3.2xlarge
HPCG-1d | AZ-Standard_NC6s_v3-1d
HPCG-1d | CHPC-NP-v100
HPCG-1d | GCP-n1-4cpu-k80-1d
HPCG-1d | GCP-n1-4cpu-v100-1d

```
TOTAL BENCHMARK COSTS
HPCG-1d | AWS-p3.2xlarge = $3.350035
HPCG-1d | AZ-Standard_NC6s_v3-1d = $3.351682
HPCG-1d | CHPC-NP-v100 = $0.145509
HPCG-1d | GCP-n1-4cpu-k80-1d = $0.663108
HPCG-1d | GCP-n1-4cpu-v100-1d = $2.727447
```

Part 3. Create your own data sets – MODULE 2

1. Start the tool (see Part 1)
2. Run the first cell as described. (see Part 2)
3. Skip Module 1 and run the cell in **Module 2 Create and edit HPC TCO** data to reload the widget. For each resource you will be creating a separate data set which will include the costs and the benchmark data run on that resource

***NOTE: be sure to hit the enter key as you enter or change the data in each field. If the field needs to be "entered" it will be highlighted in yellow. If you do not hit the enter key in that field it will not be saved in step 3.4 (Save the dataset).*



Module 2 - Create and edit HPC TCO data

1. Run the cell to reload the widget.
2. Select an entry and click **Load**.
3. Edit the fields (see **Notes** below)
4. (Optional) Change the resource name in the field above **Save**.
5. Click **Save** (see final note below).

Notes

- Make sure to press the `Enter` key after editing an entry to confirm the change. Entries that have been edited but *not* confirmed will be surrounded with a yellow border. Press `Enter` in the cell to clear the warning.
- Cost values may be numeric or symbolic formulas. For example, an entry may have the value `8.21` or `{costA} + 2*{costB}`. In the latter case, after pressing `Enter`, two new child entries will be created with the names `costA` and `costB` respectively.
- Pressing **Save** will only save the changes to memory. To make changes *persistent*, make sure to Export the resources to a file using the appropriate widget.

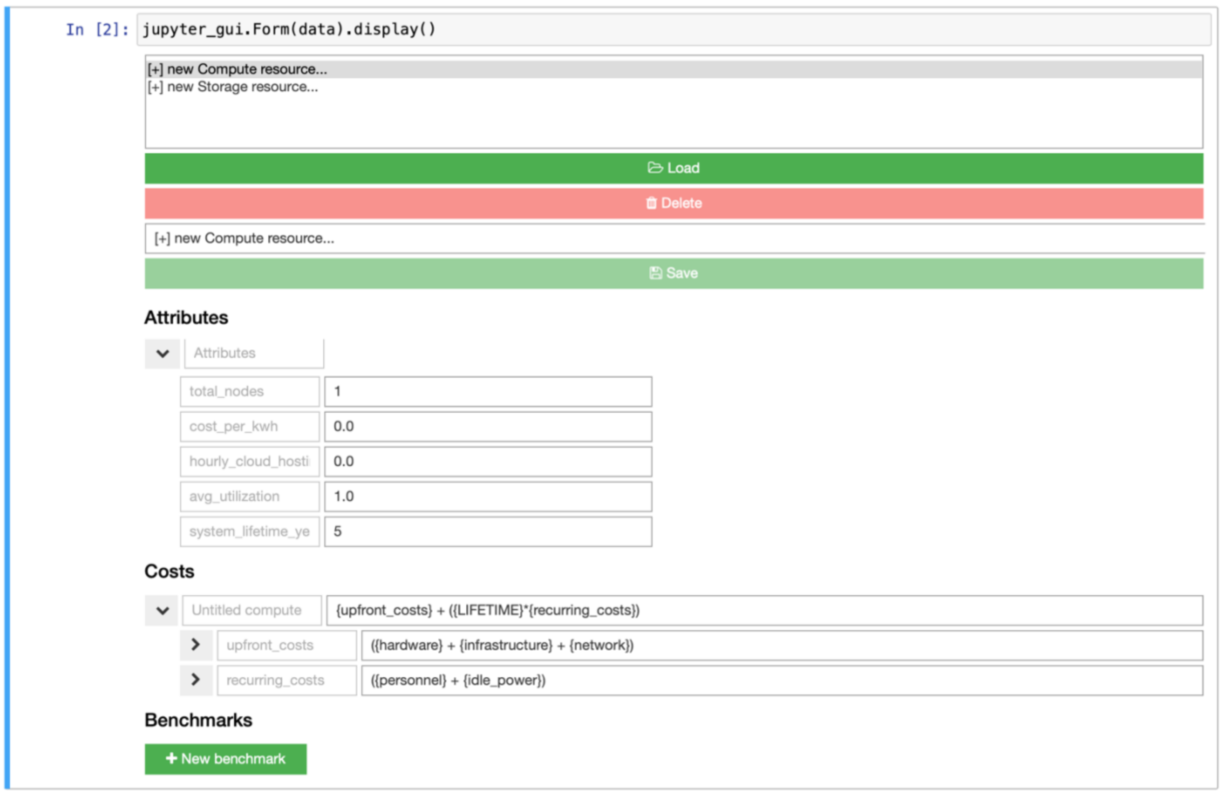
```
In [2]: jupyter_gui.Form(data).display()
```

[+] new Compute resource...
[+] new Storage resource...

Select and load an option above.

[+] new Compute resource...

4. Select "[+] new Compute Resource…" and then click "Load"
5. You will be presented with 2 sections where you input your data. The data entered will look very different between on-prem and cloud resources. We will go through creating an on-prem example and a cloud resource example.



```
In [2]: jupyter_gui.Form(data).display()
```

[+] new Compute resource...
[+] new Storage resource...

[+] new Compute resource...

Attributes

Attributes

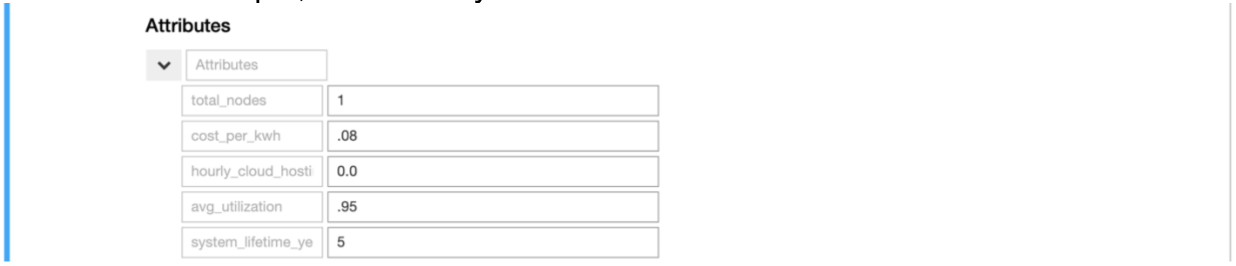
total_nodes	1
cost_per_kwh	0.0
hourly_cloud_hosti	0.0
avg_utilization	1.0
system_lifetime_ye	5

Costs

Untitled compute	{upfront_costs} + ({LIFETIME}*{recurring_costs})
upfront_costs	({hardware} + {infrastructure} + {network})
recurring_costs	({personnel} + {idle_power})

Benchmarks

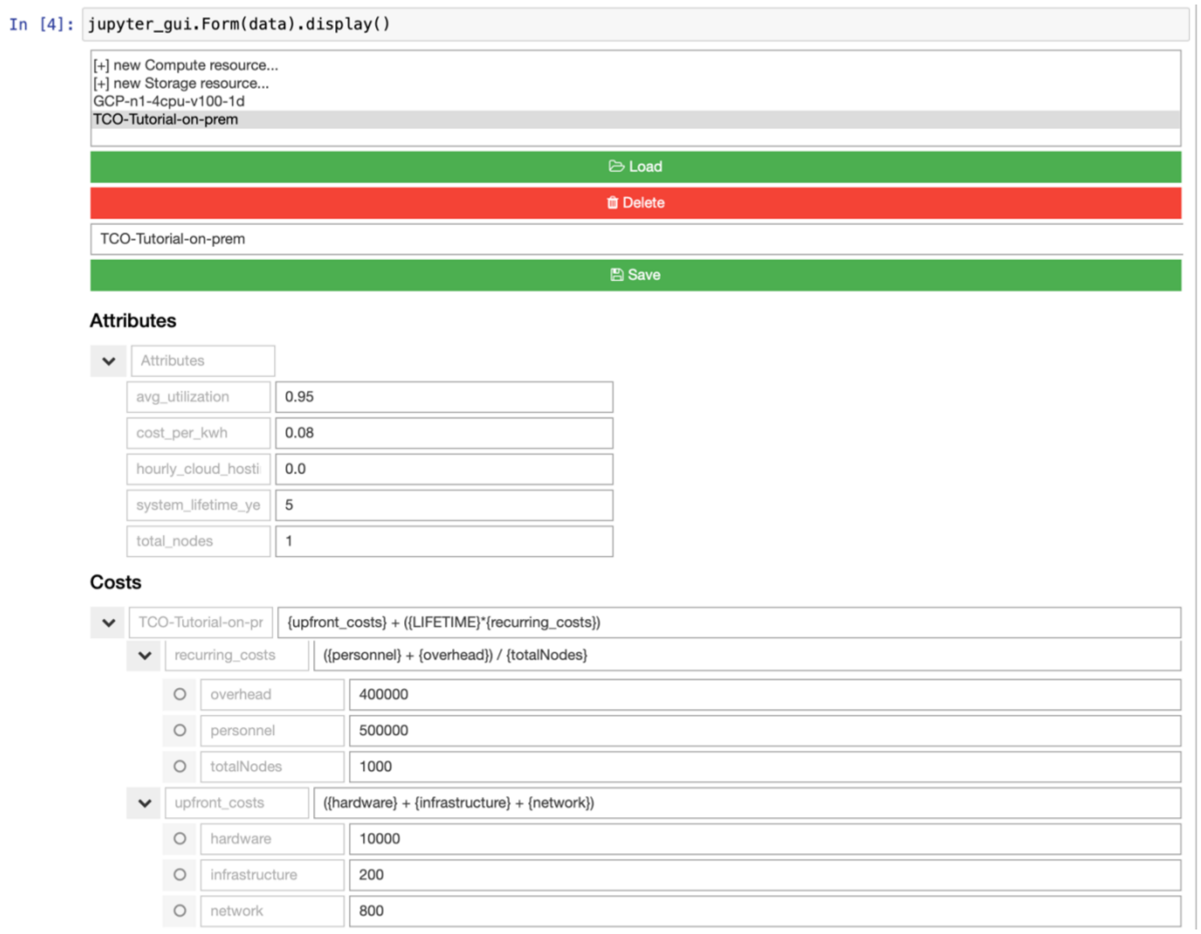
6. On-prem example
   6.1. **Attributes** – these are general factors used in determining the cost of the resources
      6.1.1. avg_utilization – scale depending upon how heavily loaded your clusters. For this example, let's say your on-prem cluster is 95% utilized. Enter .95 in this field.
      6.1.2. cost_per_kwh – would be your current rate for power. For this example, we'll use 8 cents. Enter .08 in this field.
      6.1.3. hourly_cloud_hosting – *leave zero for on prem.*
      6.1.4. system_lifetime_years – how long you expect to run the node For this example, we'll use 5 years. Leave the default.



Attributes

Attributes	
total_nodes	1
cost_per_kwh	.08
hourly_cloud_hosti	0.0
avg_utilization	.95
system_lifetime_ye	5

   6.2. **Costs** Expressing the formula with fields in {} creates the breakdown, so you can change and add your cost structure accordingly. Total cost is expressed by default as {upfront_costs} + {LIFETIME} * {recurring_costs}. The {LIFETIME} value is preset in the attributes section in the system_lifetime_years. You can minimize (click on V) and expand (click on >) the breakdown. Maximize both upfront and recurring costs to enter the data.
      6.2.1. Upfront costs – This is where you would put the one-time expenses such as the cost of hardware, switches and accessories. You could also add any additional staff expenses to setup the new node(s). By default we have set it to {hardware} + {infrastructure} + {network}. In this example we use the costs associated with a single node, the node cost is $10,000, the infrastructure cost is $200, and the network cost is $800.
      6.2.2. Recurring costs – these would be your regular ongoing costs of operating the cluster or node(s). Here you would break down in as much detail as you wish, the recurring costs of operating the clusters, including Personnel, Travel, Consulting, Datacenter rent, idle power, etc. These should be reflected in an annual total (which then utilizes the {LIFETIME} to calculate costs. (If you measure your cost of operation on a monthly basis, you can just multiply the total at the top by 12.) By default the formula is ({personnel} + {overhead}) / {totalNodes}. In this example, our annual costs for Personnel is $500,000 and Overhead is $400,000. This is the cost to run 1,000 nodes for one year. The result of the calculation is the cost to operate 1 node for 1 year. Enter these figures.



```
In [4]: jupyter_gui.Form(data).display()
```

[+] new Compute resource...
[+] new Storage resource...
GCP-n1-4cpu-v100-1d
TCO-Tutorial-on-prem

TCO-Tutorial-on-prem

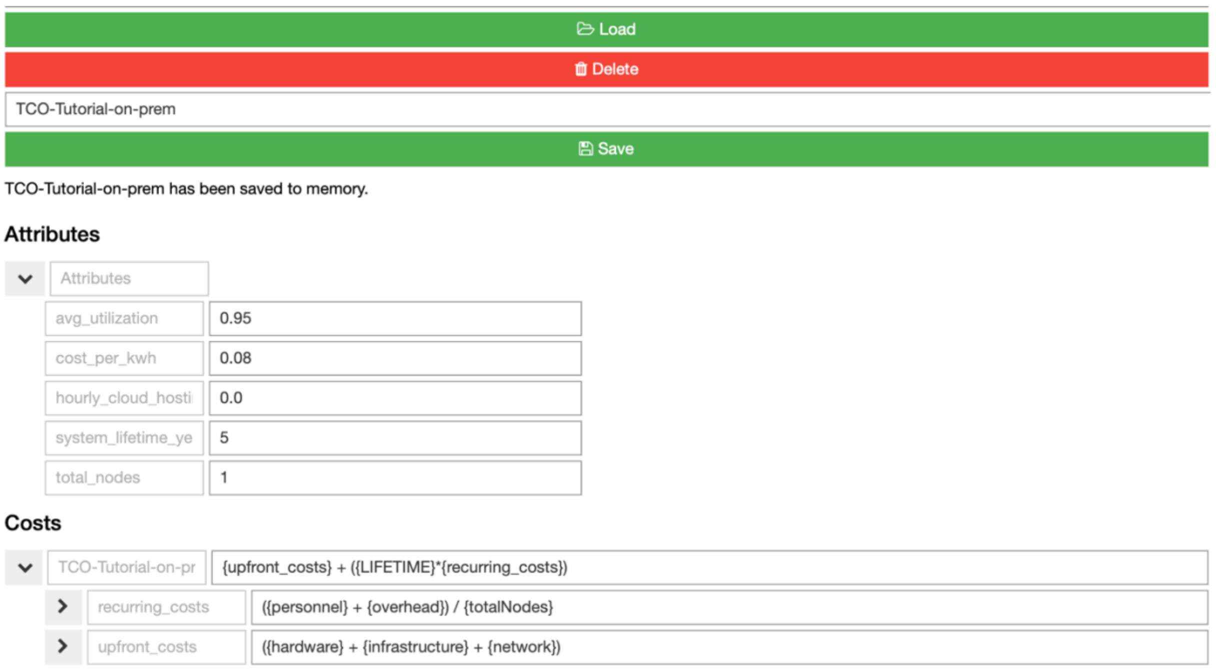
Attributes

Attributes	
avg_utilization	0.95
cost_per_kwh	0.08
hourly_cloud_hosti	0.0
system_lifetime_ye	5
total_nodes	1

Costs

TCO-Tutorial-on-pr	{upfront_costs} + ({LIFETIME}*{recurring_costs})
recurring_costs	({personnel} + {overhead}) / {totalNodes}
overhead	400000
personnel	500000
totalNodes	1000
upfront_costs	({hardware} + {infrastructure} + {network})
hardware	10000
infrastructure	200
network	800

6.3. Give your resource data set a name and save it to memory. It won't be saved to a file until later. Rename the resource by replacing the current name (just above the "Save" bar) with what you want to call this resource. Let's call it TCO-Tutorial-on-prem. Then click on the "Save" bar. You will see a message "TCO-Tutorial-on-prem has been saved to memory." To confirm it was saved correctly, you can run the widget again and the dataset should appear in the list.



Load

Delete

TCO-Tutorial-on-prem

Save

TCO-Tutorial-on-prem has been saved to memory.

Attributes

Attributes

avg_utilization	0.95
cost_per_kwh	0.08
hourly_cloud_hosti	0.0
system_lifetime_ye	5
total_nodes	1

Costs

TCO-Tutorial-on-pr	{upfront_costs} + ({LIFETIME}*{recurring_costs})
recurring_costs	({personnel} + {overhead}) / {totalNodes}
upfront_costs	({hardware} + {infrastructure} + {network})

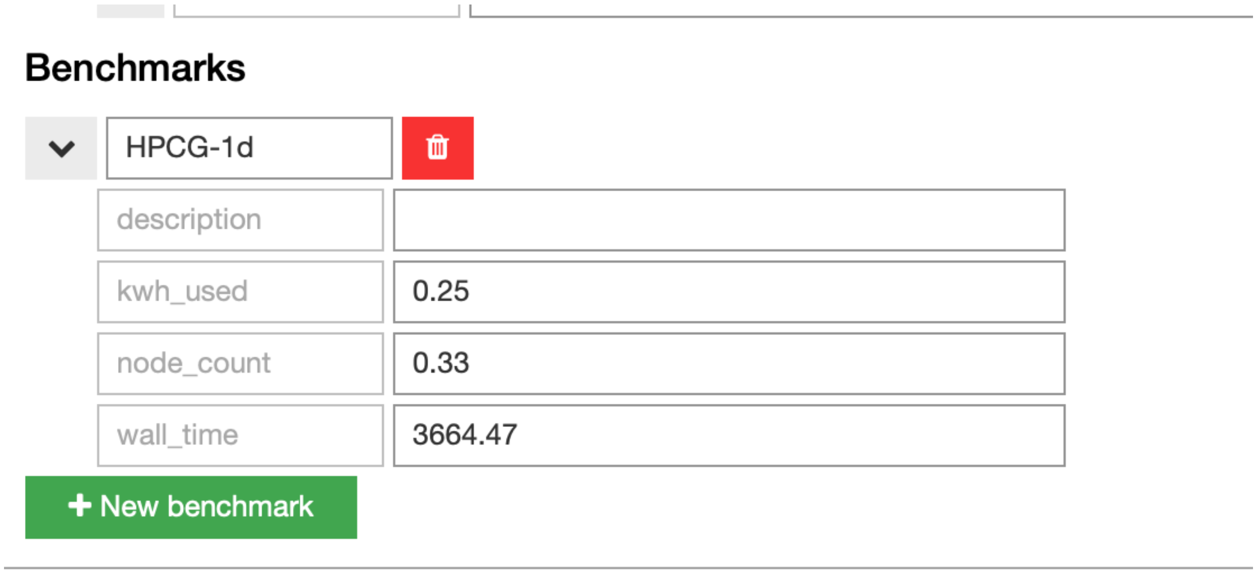
6.4. **Benchmarks** Click on the green "+ New benchmark" button.

6.4.1. Add a description. In this case, we will call it HPCG-1d (for hpcg benchmark run on 1 device (GPU).

6.4.2. Add the kwh_used for this particular benchmark. We get this information from XDMod. For this example, enter .25.

6.4.3. Add the node count. For this example, we say .33, since the node has 3 GPUs, and we only used 1. Also, sharing of nodes is enabled. If you do not allow sharing, you may want to put 1 node, as the rest of the resource is not usable while this benchmark was run.

6.4.4. Add the walltime for the benchmark in seconds. In this example, the run took 3664.47 seconds.



Benchmarks

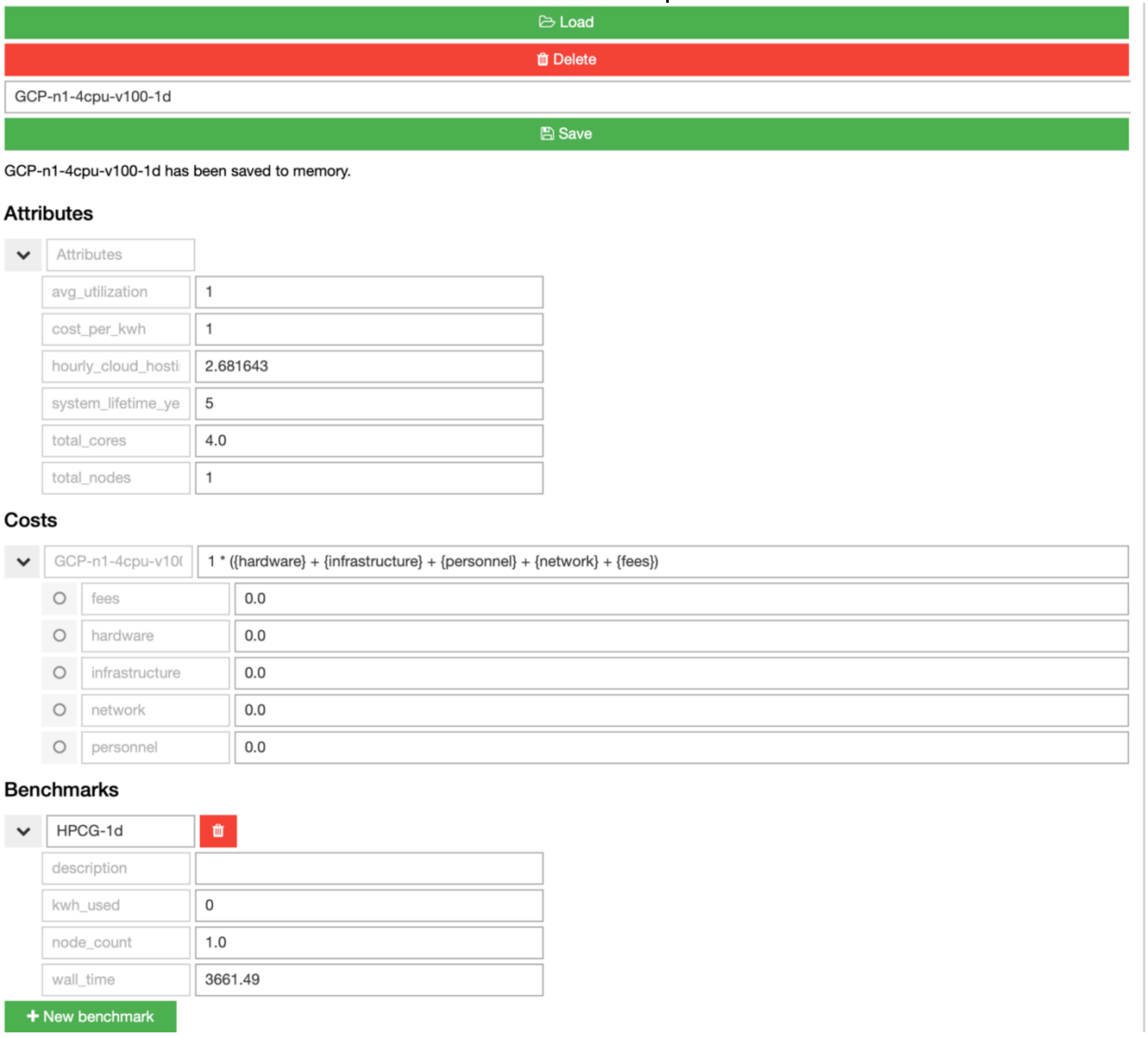
HPCG-1d

description	
kwh_used	0.25
node_count	0.33
wall_time	3664.47

+ New benchmark

6.5. IMPORTANT – click the Save bar again.

7. **Cloud Example** Select “[+] new Compute Resource…” and then click “Load”
   7.1. Attributes
      7.1.1. avg_utilization – *leave 1 for cloud*
      7.1.2. cost_per_kwh – *leave 0 for cloud*
      7.1.3. hourly_cloud_hosting – put in the cost for the resource you are creating. For this example, the hourly cost was $2.6816 for a single node on GCP with 4 v100 GPUs.
      7.1.4. system_lifetime_years – leave alone *not used for cloud resource*
   7.2. Costs – the cost breakdown does not apply to cloud resources. No change required.
   7.3. Benchmarks
      7.3.1. Again we ran the HPCG on 1 device on a google cloud node. So we will name this benchmark HPCG-1d again.
      7.3.2. Leave description, kwh_used blank.
      7.3.3. Node count =1, and the wall time for this benchmark is 3661.49
      7.3.4. Save this resource data and name it “GCP-n1-4cp-v100-1d”.



GCP-n1-4cpu-v100-1d

GCP-n1-4cpu-v100-1d has been saved to memory.

Attributes

Attributes	
avg_utilization	1
cost_per_kwh	1
hourly_cloud_hosti	2.681643
system_lifetime_ye	5
total_cores	4.0
total_nodes	1

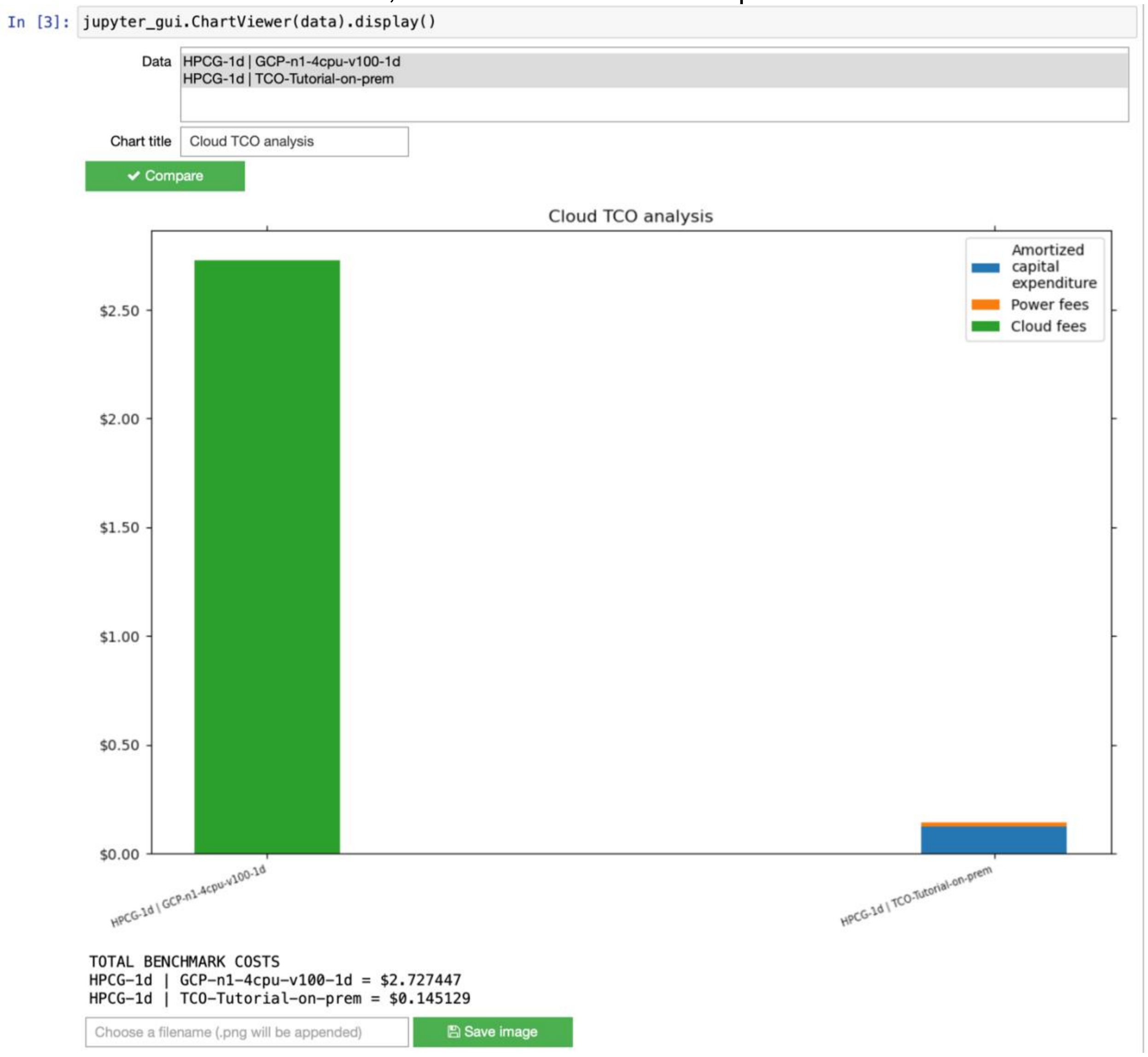
Costs

GCP-n1-4cpu-v10(	1 * ({hardware} + {infrastructure} + {personnel} + {network} + {fees})
fees	0.0
hardware	0.0
infrastructure	0.0
network	0.0
personnel	0.0

Benchmarks

HPCG-1d	
description	
kwh_used	0
node_count	1.0
wall_time	3661.49

8. Compare the cost of these two resources.
    8.1. Run Module 3. You should see the two benchmarks we enter on the two resources we created.
    8.2. Select both of these data sets, and the click on the "Compare" button.



```
In [3]: jupyter_gui.ChartViewer(data).display()
```

Data: HPCG-1d | GCP-n1-4cpu-v100-1d
HPCG-1d | TCO-Tutorial-on-prem

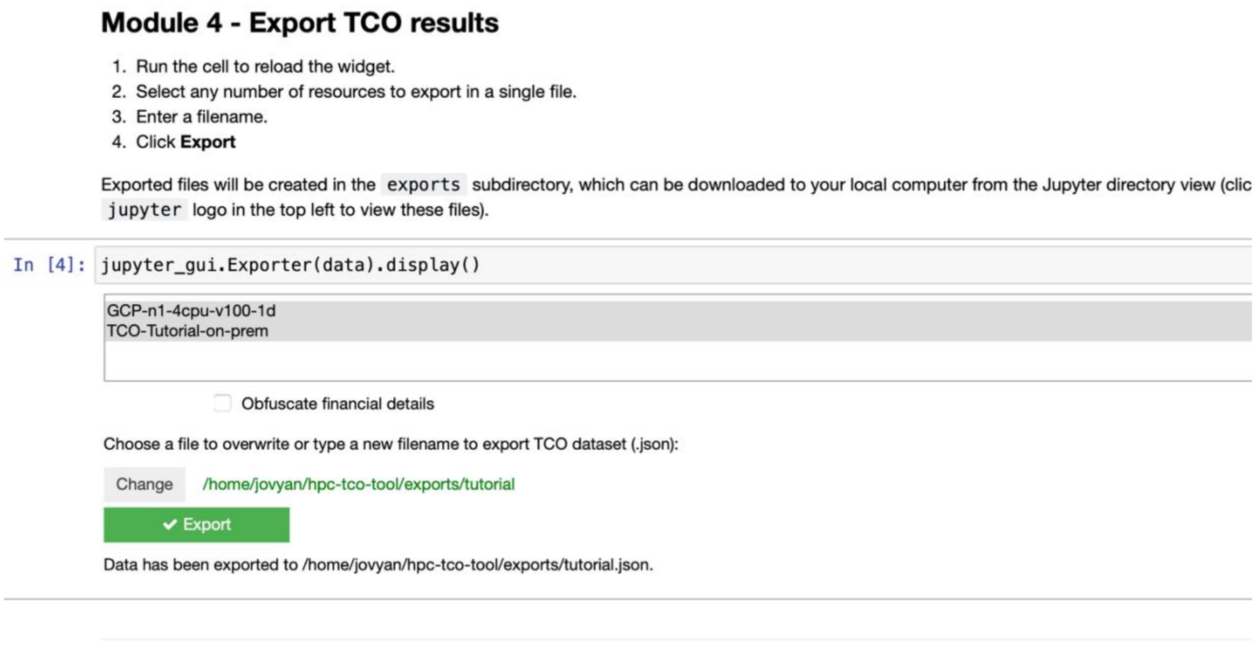
Chart title: Cloud TCO analysis

✓ Compare
Cloud TCO analysis
$2.50
$2.00
$1.50
$1.00
$0.50
$0.00
Amortized capital expenditure
Power fees
Cloud fees
HPCG-1d | GCP-n1-4cpu-v100-1d
HPCG-1d | TCO-Tutorial-on-prem
Choose a filename (.png will be appended)
Save image

```
TOTAL BENCHMARK COSTS
HPCG-1d | GCP-n1-4cpu-v100-1d = $2.727447
HPCG-1d | TCO-Tutorial-on-prem = $0.145129
```

9. **Export TCO results** IMPORTANT – *Save the dataset*. (MODULE 4)
    9.1. Click in and run the Module 4 cell.
    9.2. A list of the resource data sets you created are presented.
    9.3. Select the data sets you wish to save to a file which can later be imported using MODULE 1 in the future. In this case select both files.
    9.4. Once the desired data are highlighted. Click on the "Select" button.
    9.5. This will show you the path on your server where the file will be saved. (By default it is /home/jovyan/hpc-tec-tool/exports). Fill out the name you wish to give this set of data. Let's call it "tutorial". The .json extension will be added so do not add this to the file name.
    9.6. Click the "Select" button again.
    9.7. It will display the full path where the data set is to be exported.

9.8. Click on “Export”. It will display a message: “Data has been exported to /home/jovyan/hpc-tco-tool/exports/tutorial.json.”
9.9. Exported folder can be found in the notebook dashboard where you first clicked to open the tool.



Module 4 - Export TCO results

1. Run the cell to reload the widget.
2. Select any number of resources to export in a single file.
3. Enter a filename.
4. Click **Export**

Exported files will be created in the `exports` subdirectory, which can be downloaded to your local computer from the Jupyter directory view (clic `jupyter` logo in the top left to view these files).

```
In [4]: jupyter_gui.Exporter(data).display()
```

GCP-n1-4cpu-v100-1d
TCO-Tutorial-on-prem

☐ Obfuscate financial details

Choose a file to overwrite or type a new filename to export TCO dataset (.json):

Change /home/jovyan/hpc-tco-tool/exports/tutorial

✓ Export

Data has been exported to /home/jovyan/hpc-tco-tool/exports/tutorial.json.

Acknowledgement

I gratefully acknowledge the original University of Iowa team (Steve Fleagle, Kang-Pyo Lee, Ben Rogers, and Spencer Kuhl) who developed a preliminary prototype of the CYBER-INSIGHT tool.

I would like to express my gratitude to the team members of the TCO project: Wim Cardoen, Thomas E. Cheatham, III, David Chen (Oracle), Paul Fischer, David Richardson, Dao White, Chonghuan Xia.

This material is based upon work supported in part by the National Science Foundation under grant no. OAC-1812786.