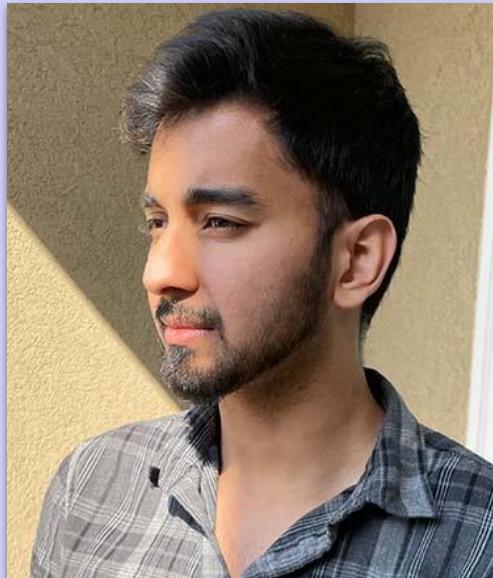




agnostiq

 covalent.xyz

Santosh K. Radha, PhD
Head of Product



Ara Ghukasyan, PhD
Research Software Engineer



Rethinking for the Compute Era.





A simple, scalable, and performant platform designed for high-compute

✓ **Unify all hardware & clouds**

Make all of your hardware, clouds, and on-premises resources accessible to your end-users with a single line of Python code

✓ **Heterogeneous hardware & software**

Covalent supports workflows that leverage a diverse range of hardware & software.

✓ **Billing, user management & administration**

Gain out-of-the-box cloud capabilities such as teams & organization-level management, billing & cost management, authentication, RBAC and more



Multi-cloud mobility.

with a single line of code

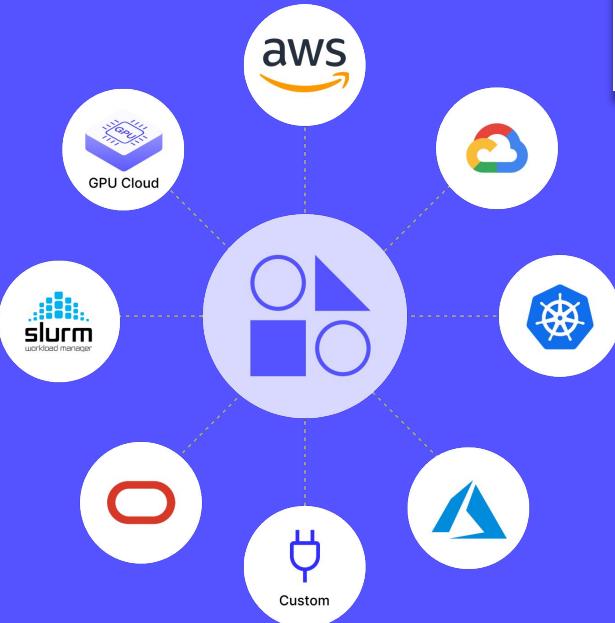


documentation



A screenshot of a Jupyter Notebook cell titled "Untitled-1". The code in the cell is:

```
def train_svm(data, C, gamma):
    X, y = data
    clf = svm.SVC(C=C, gamma=gamma)
    clf.fit(X,y)
    return clf
```



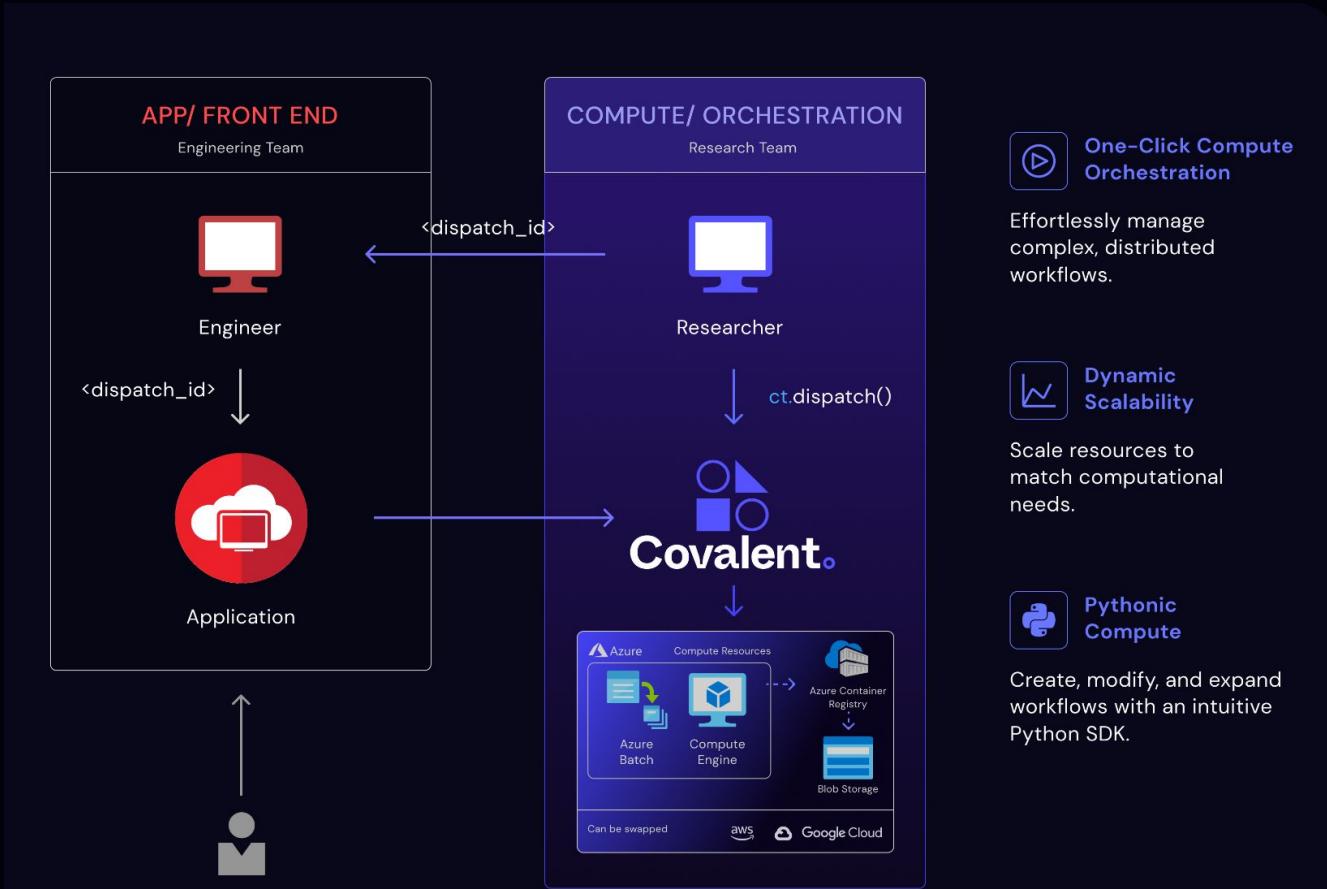
Customize executors to run tasks wherever

```
awsbatch_exec = AWSBatchExecutor(  
    vcpu=64,  
    num_gpus=4,  
    memory=16,  
    time_limit=7200,  
)  
  
qct.electron(awsbatch_exec, deps_pip=["qiskit==0.43.1"])  
def classical_compute(**data):  
    # your code here  
    ...
```

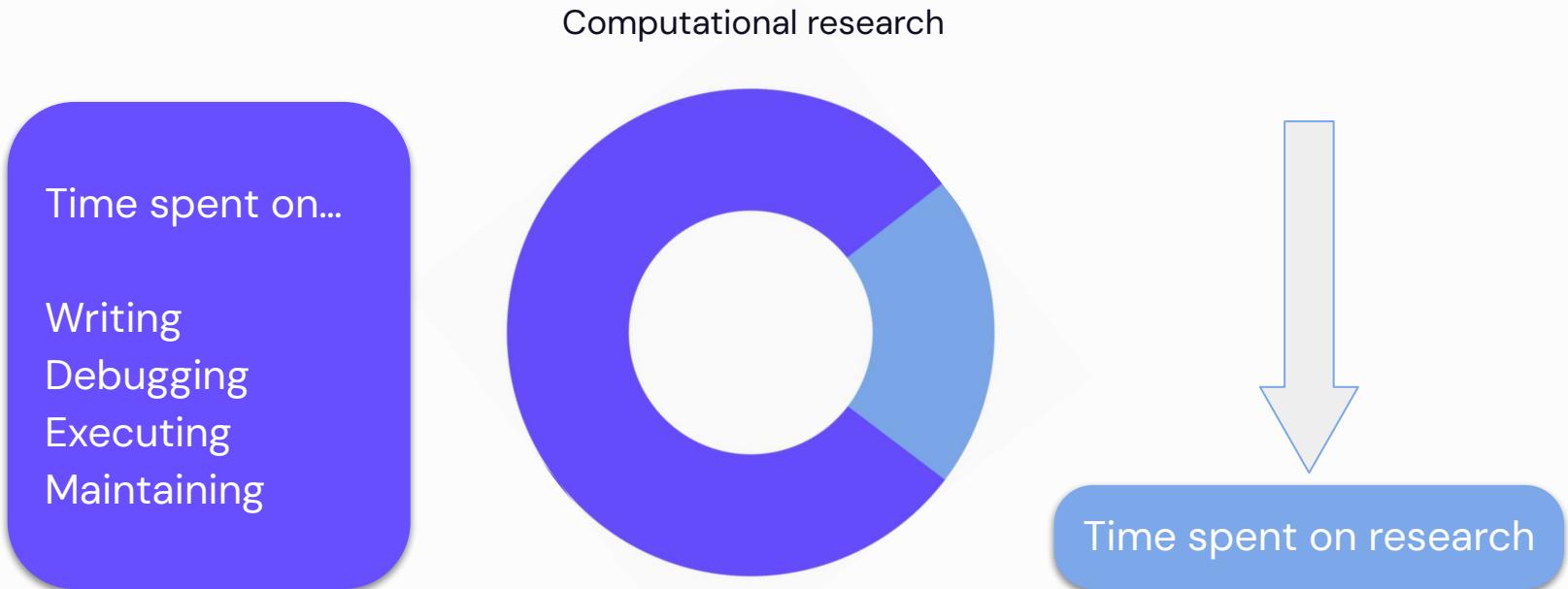
```
slurm_exec = SlurmExecutor(  
    username = "user",  
    address = "login.cluster.org",  
    ssh_key_file = "/home/user/.ssh/id_rsa",  
    options={  
        "qos": "regular",  
        "time": "01:30:00",  
        "nodes": 1,  
        "constraint": "gpu",  
    },  
    prerun_commands=[  
        "module load package/1.2.3"  
    ],  
    postrun_commands=[  
        "cp * ~/mount_dir/user_account -r"  
    ]  
)
```

Covalent as a backend for high-compute applications

- Orchestrate large scale computations
- Track, manage, and share computation results



Why does Covalent exist?



Where Covalent fits in the (Python) stack



Covalent supports a growing ecosystem of hardware and software

Classical Resources

- Slurm executor
- AWS Fargate executor
- AWS Batch executor
- Azure executor
- GCP executor
- Kubernetes executor



Software Packages

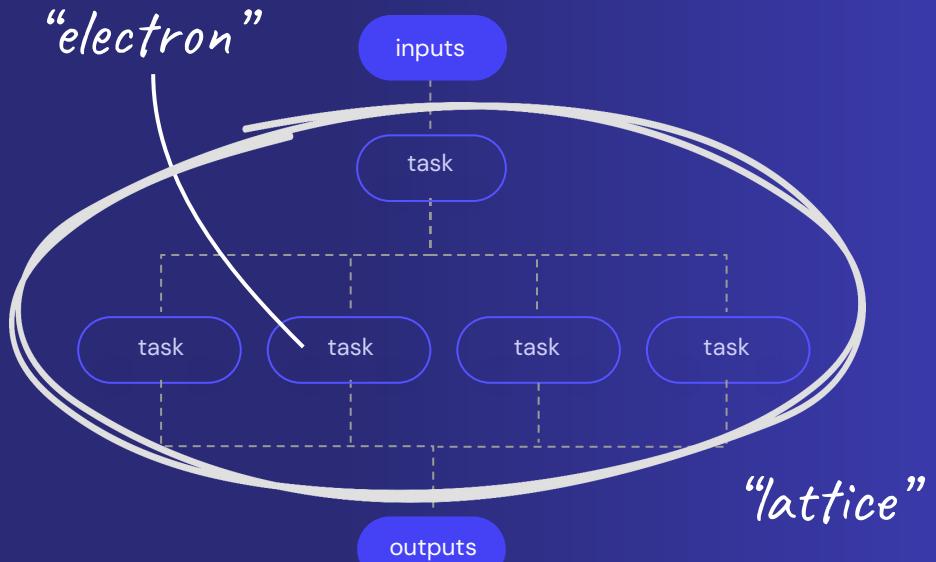
Any and all packages!



Languages



Concepts review



Electrons

Granular tasks that comprise a workflow

```
@ct.electron  
def task(x):  
    ...  
    # your code here
```



Lattice

workflow defined by task interactions

```
@ct.lattice  
def workflow(inputs):  
    outputs = []  
    for x in inputs:  
        outputs.append(task(x))  
    return outputs
```



Dispatch

Async deploy, track and reproduce workflows

```
dispatch_id = ct.dispatch(workflow)(inputs)
```



Tutorial Notebooks

🔗 https://github.com/AgnostiqHQ/tutorials_covalet_pydata_2023.git



Notebook 1: Covalent Quickstart

Covalent Quickstart

Using Covalent means interacting with four main elements.

Two decorators for *creating workflows*:

- `@ct.electron` - designates workflow tasks
- `@ct.lattice` - designates the main workflow function

and two functions for *running workflows*:

- `ct.dispatch()` - dispatches the workflow
- `ct.get_result()` - retrieve a workflow result

```
1 import covalent as ct
2
3 # Construct manageable tasks out of functions
4 # by adding the @ct.electron decorator
5 @ct.electron
6 def add(x, y):
7     return x + y
8
9 @ct.electron
10 def multiply(x, y):
11     return x*y
12
13 # Note that electrons can be shipped to variety of compute
14 # backends using executors, for example, "local" computer.
15 # See below for other common executors.
16 @ct.electron(executor="local")
17 def divide(x, y):
18     return x/y
19
20 # Construct the workflow by stitching together
21 # the electrons defined earlier in a function with
22 # the @ct.lattice decorator
23 @ct.lattice
24 def workflow(x, y):
25     r1 = add(x, y)
26     r2 = multiply(r1, y)
27     return r2
```



3. Using Covalent w/ a remote Dask Cluster

Notebook 2: Covalent + Dask + AWS

‐ TUTORIAL: Automatic Image Captioning

+ Code + Markdown

‐ Imports

```
1 import os
2 from dataclasses import dataclass
3 from datetime import datetime_
4 from typing import List
5
6 import covalent as ct
7 import torch
8 from PIL import Image
9 from transformers import BlipForConditionalGeneration, BlipProcessor
10
11 DASK_SCHEDULER_ADDRESS = "<paste-from-terraform-output>"
```

[1]

Constants

```
1 ROW_BLANK = "{datetime} | {image_name:<30} | {description}\n"
2
3 TRIGGER_DIR = os.path.abspath("./images-triggered")
4 CATALOG = os.path.join(TRIGGER_DIR, "catalog.txt")
5
6 if not os.path.exists(CATALOG):
7     with open(CATALOG, "w") as f:
8         f.write("")
```

[3]

Utilities

```
1 def get_model(type: object_id: str, label=None, **params):
```



Python

Python

4. LLMs with Covalent and AWS Batch

Notebook 3: Covalent w/ LLMs and Stable Diffusion

▼ TUTORIAL: AI Image Generator for Wikipedia Articles

+ Code + Markdown

Imports

```
1 import re
2 from collections import OrderedDict
3 from pathlib import Path
4 from typing import Any, List
5
6 import covalent as ct
7 import torch
8 from diffusers import StableDiffusionPipeline
9 from transformers import T5ForConditionalGeneration, T5Tokenizer
10 from wikipediaapi import Wikipedia
```

[1]

```
1 # Declare some constants.
2 MAX_LENGTH_PROMPT = 77
3 INPUT_BLANK = "summarize: {text}" # format prompt text into this
4 TOKENIZER_MODEL = "t5-small"
```

[2]

First, some useful helper code...

Post processor class that will render the outputted images in this Jupyter Notebook.



```
1 class PostProcessor:
2     """Post-process the generated images and summaries."""
3
4     # Regex pattern for splitting sentences.
```

5. LLMs with Covalent Cloud

Covalent Cloud

A fully-managed platform for AI & HPC



Covalent

app.covalent.xyz/settings/billing

Settings

Account | Hardware | Billing | Notifications

Billing

Compute Charges \$57.83

Billing and Payment Change how you pay your plan + Add a payment method

Invoices list

Date ↑	Invoice ID	Issue Type	Amount
2023-10-01	5bca01c9-09be-42a9-824d-ead553ae9a18	Subscription	\$43.45 U
2023-10-04	594e7463-af6c-4db3-b093-e059e25213fc	Subscription	\$32.26 U

Covalent

app.covalent.xyz

Welcome Back, Ara

Introduction

Covalent is a Pythonic workflow tool for computational scientists, AI/ML software engineers, and anyone who needs to run experiments on limited or expensive computing resources including quantum computers, HPC clusters, GPU arrays, and cloud services.

Get started

Overview

All time

- Experiments 0
- Dispatches 188
- Running 1

Recent

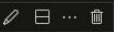
- 54b4fbf8-e659-449b-a... (Completed)
- 131e6291-c755-46de-9... (Completed)
- 2cf846d0-70cc-4cc9-9... (Completed)
- 669bf9d0-6684-4b45-a... (Completed)
- 39f1efcd-69c3-4b16-b... (Completed)
- 9b2db5b7-e480-4728-a... (Completed)

API Key

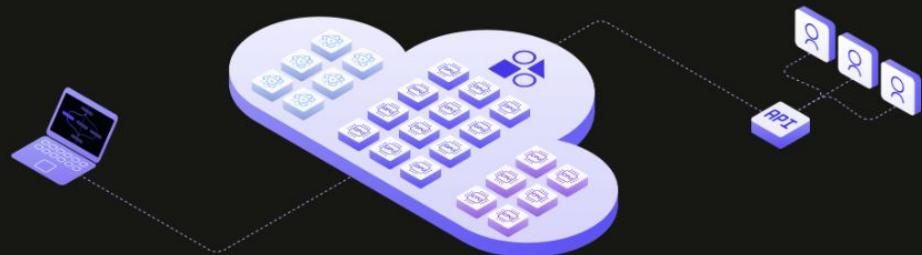
Activity List

- Dispatch 54b4fbf8-e659-449b-a... has completed ~1m
- Dispatch 131e6291-c755-46de-9... has completed ~1m
- Dispatch 2cf846d0-70cc-4cc9-9... has completed ~1m
- Dispatch 669bf9d0-6684-4b45-a... has completed ~1m
- Dispatch 39f1efcd-69c3-4b16-b... has completed ~1m
- Dispatch 9b2db5b7-e480-4728-a... is running ~4m
- Dispatch 1dc58... has completed ~4m
- Dispatch 54b4fbf8-e659-449b-a... is running ~4m
- Dispatch 8b374... is running ~4m
- Dispatch 1dc58... is running ~4m

Notebook 3: Let's try it on Covalent Cloud



- ▼ Fully Managed workflow orchestration with *Covalent Cloud*.



Usage

Covalent Cloud provides a very similar interface to open-source Covalent.

- 👉 Add one or more `@ct.electron` decorators to designate workflow tasks (i.e. *electrons*).
- 👉 Specify executors to choose electron compute resources, without worrying about provider-specific details.

Covalent Cloud Highlights

- ☁ Complete cloud abstraction. No infrastructure setup, no provider credentials required.
- 🌐 Support for multiple users and enterprises.
- 💰 Unified billing and usage tracking.



You can contribute too!

🔗 github.com/AgnostiqHQ/covalent/issues

Label issues and pull requests for new contributors Dismiss

Now, GitHub will help potential first-time contributors [discover issues](#) labeled with [good first issue](#)

[Filters](#) [Labels 50](#) [Milestones 1](#) [New issue](#)

[Clear current search query, filters, and sorts](#)

<input type="checkbox"/>	Open	Closed	Author	Label	Projects	Milestones	Assignee	Sort
<input type="checkbox"/>	7 Open	38 Closed						
<input type="checkbox"/>	Adding a <code>ct.get_all_results()</code> function	feature good first issue help wanted	#1701 opened on Jun 13 by Andrew-S-Rosen					2
<input type="checkbox"/>	@ct.lattice(<code>ct.electron</code>) seems to change function scope	good first issue unitaryhack-bounty	#1643 opened on May 12 by santoshkumarradha					3
<input type="checkbox"/>	Convert Qiskit Runtime/AWS Batch neural network tutorial to notebook format	documentation good first issue help wanted	#1572 opened on Apr 11 by araghukas					3
<input type="checkbox"/>	Switch from using <code>ct.wait</code> to <code>Electron.wait_for</code>	good first issue help wanted improvement	#1525 opened on Feb 15 by kessler-frost	 1 task				2
<input type="checkbox"/>	Fix MNIST tutorial image	good first issue help wanted	#1422 opened on Nov 22, 2022 by cjao					1



Covalent is the simplest way to orchestrate
high-compute workflows across
heterogeneous computing environments



[agnostiqHQ/covalent](#) ★



[covalent.xyz](#)



[@covalentxyz](#)

aQ

