

# Cloud enabling educational platforms with corc

Rasmus Munk<sup>1</sup>, David Marchant<sup>1</sup> and Brian Vinter<sup>2</sup>

<sup>1</sup>Niels Bohr Institute, Blegdamsvej 17, Copenhagen, 2100, Denmark

<sup>3</sup>Aarhus University, Ny Munkegade 120, Aarhus C, 8000, Denmark

## Abstract

In this paper, it is shown how teaching platforms at educational institutions can utilize cloud platforms to scale a particular service, or gain access to compute instances with accelerator capability such as GPUs. Specifically at the University of Copenhagen (UCPH), it is demonstrated how the internal JupyterHub service, named Data Analysis Gateway (DAG), could utilize compute resources in the Oracle Cloud Infrastructure (OCI). This is achieved by utilizing the introduced Cloud Orchestrator (corc) framework, in conjunction with the novel JupyterHub spawner named MultipleSpawner. Through this combination, we are able to dynamically orchestrate, authenticate, configure, and access interactive Jupyter Notebooks in the OCI with user defined hardware capabilities. These capabilities include settings such as the minimum amount of CPU cores, memory and GPUs the particular orchestrated resources must have. This enables teachers and students at educational institutions such as UCPH to gain easy access to the required capabilities for a particular course. In addition, we lay out how this groundwork, will enable us to establish a Grid of Clouds between multiple trusted institutions. This enables the exchange of surplus computational resources that could be employed across their organisational boundaries.

## Keywords

teaching, cloud computing, grid of clouds, Jupyter Notebook

## 1. Introduction

The availability of required computational resources in organisations, such as scientific or educational institutions, is a crucial aspect of delivering the best scientific research and teaching. When teaching courses involving data analysis techniques it can be beneficial to have access to specialized platforms, such as GPU accelerated architectures.

At higher educational institutions, such as the University of Copenhagen (UCPH) or Lund University (LU), these centers are substantial investments, that are continuously maintained and upgraded. However, the usage of these resources often varies wildly between being fully utilized to sitting idly by.

We therefore propose, that these institutional resources be made available (with varying priority) across trusted educational and scientific organisations. Foremost, this is to enable the

---

*CTE 2020: 8th Workshop on Cloud Technologies in Education, December 18, 2020, Kryvyi Rih, Ukraine*

✉ rasmus.munk@nbi.ku.dk (R. Munk); d.marchant@ed-alumni.net (D. Marchant); vinter@au.dk (B. Vinter)

🌐 [https://research.ku.dk/search/result/?pure=en/persons/rasmus-munk\(a72145a7-0203-4791-bb8f-6073bc23fe2f\).html](https://research.ku.dk/search/result/?pure=en/persons/rasmus-munk(a72145a7-0203-4791-bb8f-6073bc23fe2f).html)


(R. Munk); [https://research.ku.dk/search/result/?pure=en/persons/david-gray-marchant\(ff6af890-33df-4414-9a9d-c3d33258ad1f\).html](https://research.ku.dk/search/result/?pure=en/persons/david-gray-marchant(ff6af890-33df-4414-9a9d-c3d33258ad1f).html) (D. Marchant);

[https://pure.au.dk/portal/en/persons/brian-vinter\(a4fe861f-5a04-4e93-a5b3-c633045eb82e\).html](https://pure.au.dk/portal/en/persons/brian-vinter(a4fe861f-5a04-4e93-a5b3-c633045eb82e).html) (B. Vinter)

ORCID 0000-0003-0333-4295 (R. Munk); 0000-0003-4262-7138 (D. Marchant); 0000-0002-3947-9878 (B. Vinter)

© 2020 Copyright for this paper by its authors.

Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

voluntary sharing of underused resources to other institutions, thereby potential establishing greater scalability than can be found within each individual institution.

### **1.1. Basic IT**

Within institutions such as UCPH, there is a mixture of services that each provides. At the very basic level, there are infrastructure services such as networking, account management, email, video conferencing, payroll management, license management, as well OS and software provisioning. In this paper, we define these as Basic IT services. At educational institutions, additional services can be added to this list, these include services for handling student enrollment, submissions, grading, course management, and forum discussions. As with the initial Basic IT services, these are typically off the shelf products that needs to be procured, installed, configured and maintained on a continuous basis.

A distinguishing trait of Basic IT services, in an education context, is that they are very predictable in terms of the load they will exhibit, both in times of high and low demand. For instance, there will be busy junctions, such as assignment hand in days, release of grades, student enrollment, and so on. In contrast, holiday and inter-semester periods will likely experience minor to no usage. Given this, these services are classic examples of what cloud computing was developed to provide. Efficient utilization of on-demand resources, with high availability and scalability to handle fluctuating usage in a cost effective manner.

### **1.2. Science IT**

Science IT services, in contrast, revolve around the institutions scientific activities whether by researchers or students. They include services such as management, sharing, transferring, archiving, publishing, and processing of data, in order to facilitate the scientific process. In addition, these facilities also enable lecturers to utilize their research material in courses, giving students access to the same platform and resources.

What distinguishes these services, is that they impose different constraints compared to Basic IT services. These typically involve areas such as, computational load, security, budgetary, scientific, and legal requirements, among others. For example, it is often too inefficient, or costly to utilize public cloud resources for the storing and processing of large scientific datasets at the petabyte scale. In this case, a more traditional approach such as institutional compute resources is required [1].

Research fields such as climate science [2], oceanography [3], and astronomy [4], often employ experimental simulations as a common scientific tool. These simulations produce output up to petabytes in size, that still need to be stored for subsequent postprocessing and analysis. Upon a scientific discovery from this process, the resulting datasets needs to be archived in accordance with regulatory requirements, which in the case of UCPH is 5 years [5] (only available in Danish).

### **1.3. Institutional resources**

High Performance Computing (HPC) and regular compute centers are often established at higher educational institutions to provide Science IT services. The UCPH [6], University of

Antwerp [7], and LU [8] compute centers are examples of this. In addition, institutions can also gain access to similar resources through joint facilities like the Vienna Scientific Cluster [9], which supports 19 institutions, 10 of which are higher educational institutions. Finally there are national and pan-national resources such as ARCHER2 (UK) [10] or the EuroHPC [11] that review applications before access is granted.

These established centers are very expensive to build and have a limited lifespan before they need to be replaced. Even smaller educational compute platforms follow a similar life-cycle. For instance, at the UCPH a typical machine has a lifetime of 5 years before it needs to be replaced. This is whether the machine has been heavily utilized or not. Therefore, it is important that these systems across institutions are utilized, not only efficiently, but at maximum capacity throughout their lifetime.

For organising the sharing of resources across trusted educational and scientific organisations, inspiration is drawn from the way traditional computational Grids have been established [12]. The difference is, that instead of establishing a Grid where individual resources are attached, this model will instead be based on each institution establishing a Cloud of resources that are shared via a Grid. This means that the Grid is responsible for interconnecting disjointed clouds, whether they be institutional or public cloud platforms. The result being an established model for sharing cloud resources across educational institutions in support of cloud services for bachelor and master courses, general workshops, seminars and scientific research.

In this paper, we present how an existing teaching and research service at UCPH could be enabled with access to a cloud framework, which is the first step towards a Grid of Clouds resources. We accomplish this by using the Cloud Orchestrator (corc) framework [13]. Through this, we are able to empower the DAG service with previously inaccessible compute resources across every course at UCPH. This was previously not feasible with internal resources alone. Since we do not have access to other institutional resources at this point in time, we utilized a public cloud provider to scale the service with external resources.

## 2. Background

At the Niels Bohr Institute (NBI), part of UCPH, we host a number of Science IT services that are part of providing a holistic educational platform for researchers, teachers, students, and general staff. A subset of these Science IT services have been especially beneficial across all levels of teaching. Namely, services such as the University Learning Management System (LMS), called Absalon, which is based on Canvas [14] for submissions and grading. The Electronic Research Data Archive (ERDA) [15] for data management and sharing tasks. In addition to the Data Analysis Gateway (DAG) [16], which is a JupyterHub powered platform for interactive programming and data processing in preconfigured environments.

### 2.1. Teaching platforms

The combination of these subset services, in particular the combination of ERDA and DAG, has been especially successful. Teachers have used these to distribute course material through ERDA, which made the materials available for students to work on at the outset of the course. This ensures that students can get on with the actual learning outcomes from the get go, and

not spend time on tedious tasks such as installing prerequisite software for a particular course. Due to budgetary limitations, we have only been able to host the DAG service with standard servers, that don't give access to any accelerated architectures.

Across education institutions, courses in general have varying requirements in terms of computing resources, environments, and data management, as defined by the learning outcomes of the course. The requirements from computer science, data analysis, and physics oriented courses are many, and often involve specialized compute platforms. For example, novel data analysis techniques, such as Machine Learning or Deep Learning have been employed across a wide range of scientific fields. What is distinct about these techniques is the importance of the underlying compute platform on which it is being executed. Parallel architectures such as GPUs in particular are beneficial in this regard, specifically since the amount of independent linear systems that typically needs to be calculated to give adequate and reliable answers are immense. The inherent independence of these calculations, makes them suitable for being performed in parallel, making it hugely beneficial to utilize GPUs [17].

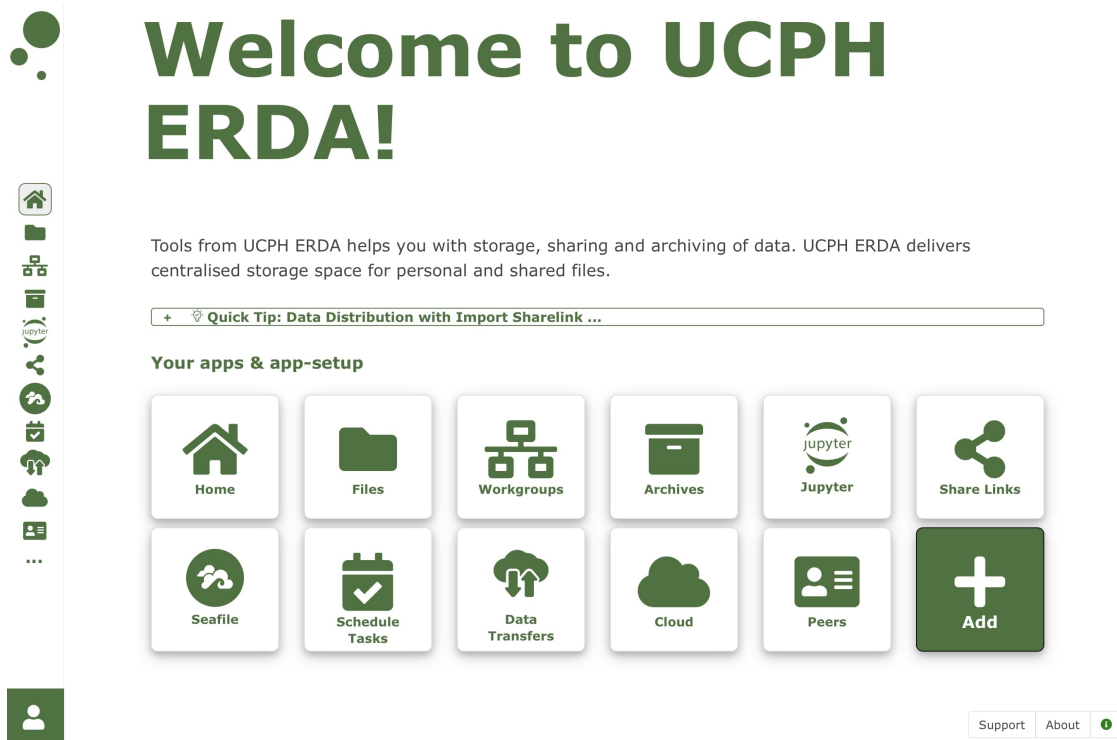
Given that the DAG service was an established service at UCPH for data analysing and programming in teaching bachelor and master students, it seemed the ideal candidate to enable with access to cloud resources with accelerator technology. For instance, courses such as Introduction to Computing for Physicists (abbreviated to DATF in Danish) [18], Applied Statistics: From Data to Results (APPSTAT) [19], and High Performance Parallel Computing (HPPC) [20], all would benefit from having access to GPU accelerators to solve several of the practical exercises and hand-in assignments.

## 2.2. ERDA

ERDA provides a web based data management platform across UCPH with a primary focus on the Faculty of Science. Its primary role is to be a data repository for all employees and students across UCPH. Through a simple web UI powered by a combination of an Apache webserver and a Python based backend, users are able to either interact with the different services through its navigation menu, or a user's individual files and folders via its file manager. An example of the interface can be seen in figure 1. The platform itself is a UCPH-specific version of the open source Minimum Intrusion Grid (MiG) [21], that provides multiple data management functionalities. These functionalities includes easy and secure upload of datasets, simple access mechanisms through a web file manager, and the ability to establish collaboration and data sharing between users through Workgroups.

## 2.3. Jupyter

Project Jupyter [22] develops a variety of open source tools. These tools aim at supporting interactive data science, and scientific computing in general. The foundation of these is the IPython Notebook (.ipynb) format (evolved out of the IPython Project [23]). This format is based on interpreting special segments of a JSON document as source code, which can be executed by a custom programming language runtime environment, also known as a kernel. The JupyterLab [24] interface (as shown in figure 2) is the standard web interface for interacting with the underlying notebooks. JupyterHub [25] is the de-facto standard to enable multiple



**Figure 1:** ERDA Interface

users to utilize the same compute resources for individual Jupyter Notebook/Lab sessions. It does this through its own web interface gateway and backend database, to segment and register individual users before allowing them to start a Jupyter session.

In addition, JupyterHub allows for the extension of both custom Spawners and Authenticators, enabling 3rd party implementations. The Authenticator is in charge of validating that a particular request is from an authentic user. The responsibility of the Spawner is how a Jupyter session is to be scheduled on a resource. Currently there exist only static Spawners that utilize either preconfigured resources that have been deployed via Batch, or Container Spawners, or at selective cloud providers such as AWS [26]. As an exception to this, the WrapSpawner [27] allows for dynamic user selections through predefined provides. However, these profiles cannot be changed after the JupyterHub service is launched, making it impossible to dynamically change the set of supported resources and providers. Therefore it would be of benefit if a Spawner extended the WrapSpawner’s existing capabilities with the ability to dynamically add or remove providers and resources.

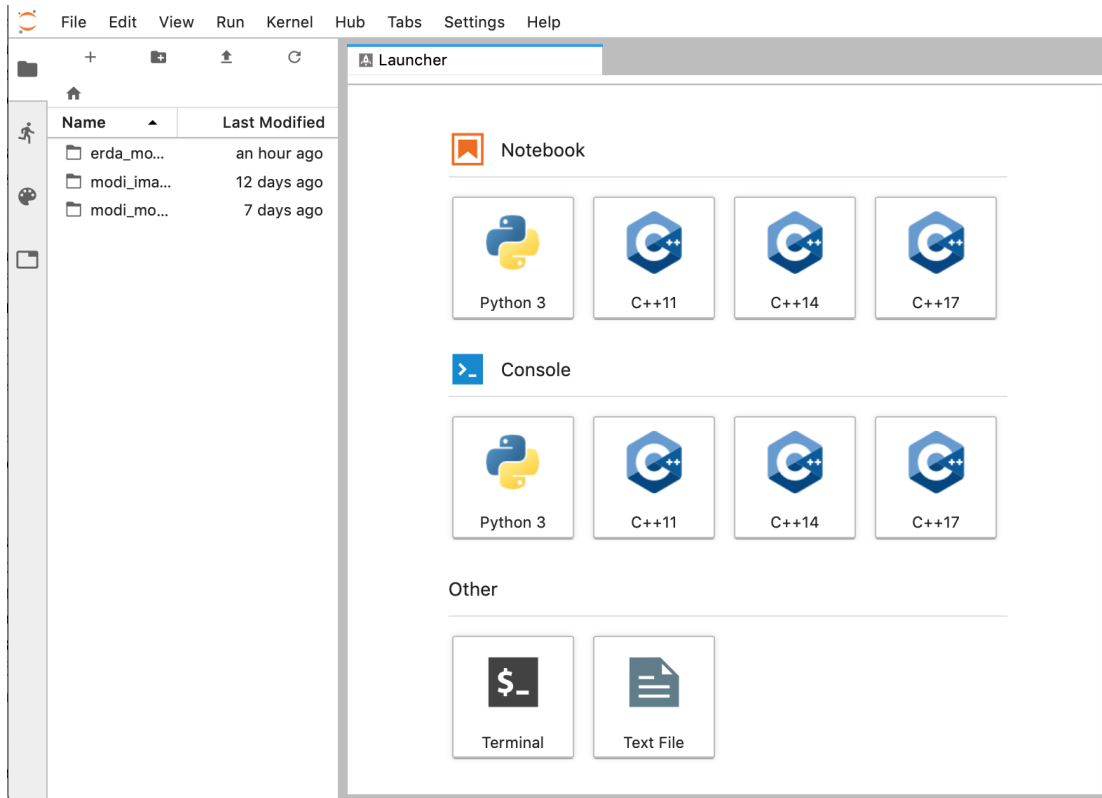


Figure 2: JupyterLab Interface

### 3. Related work

As presented in [28], Web-based learning by utilizing cloud services and platforms as part of the curriculum is not only feasible, but advisable. In particular, when it comes to courses with programming activities for students, educational institutions should enable access to innovative Web-based technologies that supports their learning. These include interactive programming, version control and automated programming assessments to ensure instant feedback.

#### 3.1. Interactive programming portals

Research in cloud computing for education typically revolves around using Web-enabled Software as a Service (SaaS) applications. Examples of such include platforms such as GitHub [29], Google Docs [30], Google Colaboratory [31], Kaggle [32], and Binder [33]. Each of these can fill a particular niche in a course at the teacher's or student's discretion. Nevertheless, the provided capability often does come with its own burdens, in that the administration of the service is often left to the teaching team responsible for the course. This responsibility typically includes establishing student access, course material distribution to the specific platform, guides on how

**Table 1**  
Subset of Jupyter Cloud Platforms Features

Provider	Native Persistence	Languages	Collaborate	MaxTime (inactive, max)
Binder[37]	None	User specified <sub>1</sub>	Git	10m, 12h <sup>2</sup>
Kaggle [38]	Kaggle Datasets	Python3, R	Yes	60m, 9h
Google Colab [39]	GDrive, GCloud Storage	Python3, R	Yes	60m, 12h* <sup>3</sup>
Azure Notebooks [40, 41]	Azure Libraries	Python{2,3}, R, F#	NA	60m, 8h* <sup>4</sup>
CoCalc [42]	CoCalc Project	Python{2,3}, R, Julia, etc	Yes*	30m, 24h
Datalore [43]	Per Workbook	Python3	Yes	60m, 120h <sup>5</sup>
DAG [16]	ERDA	Python2,3, R, C++, etc	Yes	2h, unlimited <sup>6</sup>

to get started with the service and solving eventual problems related to the service throughout the course. In addition, many of the external cloud services that offer free usage, often have certain limitations, such as how much instance utilisation a given user can consume in a given time span. Instead, providing such functionalities as Science IT services, could reduce these overheads and enable seamless integration into the courses. Furthermore, existing resources could be used to serve the service by scaling through an established Grid of Clouds.

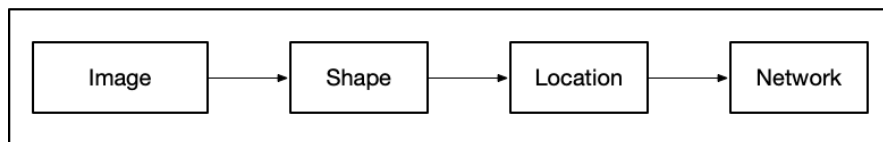
In terms of existing public cloud platforms that can provide Jupyter Notebook experiences, DAG is similar to Google Colaboratory, Binder, Kaggle, Azure Notebooks [34], CoCalc [35], and Datalore [36]. All of these online options, have the following in common. They all have free tier plans available with certain hardware and usage limitations. All are run entirely in the web browser and don't require anything to be installed locally. At most they require a valid account to get started. Each of them present a Jupyter Notebook or Notebook like interface, which allows for both export and import of Notebooks in the standard format. An overview of a subset of the supported features and usage limits across these platforms can be seen in Table 1, and their hardware capabilities in Table 2. From looking at the features, each provider is fairly similar in terms of enabling Languages, Collaborating, and Native Persistence (i.e. the ability to keep data after the session has ended). However, there is a noticeable difference, in the maximum time (MaxTime) that each provider allows a given session to be inactive before it is stopped. With CoCalc being the most generous, allowing 24 hours of activity before termination. In contrast, internal hosted services such as DAG allow for the institution to define this policy. At UCPH, we have defined this to be 2 hours of inactivity, and an unlimited amount of active time for an individual session. However, as Table 2 shows, we currently don't provide any GPU capability, which is something that could be changed through the utilisation of an external cloud with GPU powered compute resources.

Given this, the DAG service seemed as the ideal candidate to empower with external cloud resources. Both because it provides similar features as the public cloud providers in terms of Languages and Collaborate ability, but also since it is integrated directly with UCPHs data management service.



**Table 2**  
Hardware available on Jupyter Cloud Platforms

Provider	CPU	Memory (GB)	Disk Size (GB)	Accelerators
Binder	NA	1 Min, 2 MAX	No specified limit*	None
Kaggle1	4 cores	17	5	None
Kaggle2	2 cores	14	5	GPU <sup>7</sup> or TPU <sup>8</sup> [44]
Google Colab Free	NA	NA	GDrive 15	GPU or TPU (thresholded access)
Azure Notebooks (per project)	NA	4	1	GPU (Pay)
Cocalc (per project)	1 shared core	1 shared	3	None
Datalore	2 cores	4	10	None
DAG	8 cores	8	unlimited <sup>9</sup>	None



**Figure 3:** Workflow for orchestrating a compute node

### 3.2. Cloud Orchestration

Cloud resources are typically provided by the infrastructure service through some form of orchestration. Orchestration is a term for providing an automated method to configure, manage and coordinate computer systems [45]. Through orchestration, an organisation or individual is able to establish a complex infrastructure through a well defined workflow. For instance, the successful creation of a compute node involves the processing of a series of complex tasks that all must succeed. An example of such a workflow can be seen in figure 4. Here a valid Image, Shape, Location and Network has to be discovered, selected, and successfully utilized together in order for the cloud compute node to be established. An Image is the target operating system and distribution, for instance Ubuntu 20.04 LTS. A Shape is the physical configuration of the node, typically involving the amount of CPU cores, memory and potential accelerators. Location is typically the physical location of where the resource is to be created. Cloud providers often use the term Availability Zone instead but it generally defines which datacenter to utilize for the given task. Network encompasses the entirety of the underlying network configuration, including which Subnet, Gateway, and IP address the compute node should utilize. In the context of a federated network like a Grid, the orchestration would ideally involve the automated provisioning of the computational resource, the configuration of said resource, and ensure that the resource is correctly reachable through a network infrastructure.

Multiple projects have been developed that automate development and system administration



tasks such as maintenance, testing, upgrading, and configuration. These includes packages such as TerraForm [46], Puppet [47], Chef [48], and Ansible [49], all of which open source projects that can be utilized across a range of supported cloud providers. Nevertheless, in terms of enabling workflows that can provide orchestration capabilities, these tools are limited in that they typically only focuses on a subset of the orchestration functionalities such as provisioning and deployment or configuration and maintenance. For instance TerraFrom is a tool that focuses on infrastructure deployment whereas Puppet, Chef and Ansible are primarily concerned with configuration and maintenance of existing systems. In contrast commercial cloud providers typically also provide their own orchestration-like tools and Software Development Kits (SDKs), enabling the ability to interact with their respective cloud system. For instance, Oracle provides the Oracle Cloud Infrastructure CLI [50] tool that can interact with their infrastructure. The same applies to the Amazon AWS CLI [51], in addition to a vast complement of tool-kits [52] that provide many different AWS functionalities including orchestration. In contrast, commercial cloud provided tools are often limited to only support the publishing cloud vendor and do not offer cross-cloud compatibility, or the ability to utilize multiple cloud providers interchangeably.

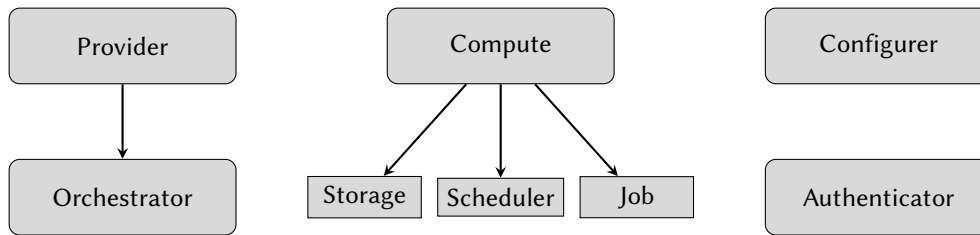
Cloud orchestration developments for the scientific community, especially those aiming to provide cross-cloud deployments, have mostly been based on utilizing on premise cloud IaaS platforms such as OpenStack [53] and OpenNebula [54]. Developments have focused on providing higher layers of abstraction to expose a common APIs that allow for the interchangeable usage of the underlying supported IaaS platforms. The infrastructure is typically defined in these frameworks through a Domain Specific Language (DSL) that describes how the infrastructure should look when orchestrated. Examples of this include cloud projects such as INDIGO-cloud [55] [56], AgroDAT [57] and Occopus [57]. These frameworks, nonetheless do not allow for the utilization of commercial or public cloud platforms, since they rely on the utilization of organisationally defined clouds that are traditionally deployed, managed, and hosted by the organisation itself. Although required, if as stated, we are to establish a Grid of Clouds which should allow for the inclusion of public and commercial cloud platforms. The corc framework was developed and designed to eventually support the scheduling of cloud resources across both organisations and public cloud providers.

## **4. The first cloud enabled service**

To establish a Grid of Cloud resources, we started with enabling the usage of a single public cloud provider to schedule DAG Notebooks on. Through this we created the foundations for the eventual Grid structure that would allow the resources to be scheduled across multiple clouds and organisations.

### **4.1. Corc**

The corc framework was implemented as a Python package. The package establishes the foundations for essential functions such as orchestration, computation, configuration, and authentication against supported cloud providers and cloud resources. Overall, corc is a combination of an Infrastructure as a Service (IaaS) management library, and a computation oriented



**Figure 4:** Cloud Orchestrator Framework Overview

scheduler. This enables the ability to schedule services on a given orchestrated resource. An overview of the architecture can be seen in figure 4.1.

The first provider to be integrated into the framework was the OCI IaaS. This was chosen, because the UCPH had a preexisting collaboration with Oracle, that enabled the usage of donated cloud resources for testing and development. As also highlighted, this does not limit the integration of other cloud providers into the framework, which the framework was designed for. Furthermore, as explored in section 2.3. A new Spawner, named MultipleSpawner was introduced, to provide the necessary dynamic selection of cloud providers.

As Figure 4.1 indicates, for each provider that corc supports, an orchestrator for that provider needs to be defined within corc. In addition, the framework defines three other top level components, namely Compute, Configurer, and Authenticator. All three are abstract definitions allowing for specific implementations to support the targeted resources which they apply to. A service can therefore be enabled with the ability to utilize cloud resources by integrating the corc components into the service itself. This method is limited to services that are developed in Python. In addition, corc also defines a Command Line Interface (CLI), that can be used to interact with the cloud provided resources directly. Details about how the framework and CLI can be used will not be presented in this paper, but can be found in [13].

```

{
  "virtual_machine": [
    {
      "name": "oracle_linux_7_8",
      "provider": "oci",
      "image": "Oracle Linux 7.8"
    }
  ]
}
  
```

Listing 1: Spawner Deployment configuration

## 4.2. MultipleSpawner

MultipleSpawner [58] is a Python package allowing for the selection of dynamic Spawners and resources. Structurally, it is inspired by the WrapSpawner [27], through the MultipleSpawner integrates corc into the Spawner itself. This enables the JupyterHub service to manage and utilize

cloud resources on a dynamic set of providers. In order to enable the MultipleSpawner to support these dynamic resources providers, two JSON configuration files needs to be defined. One of these is shown in listing 1, and defines the specific resource type that should be deployed on the provider. Currently the MultipleSpawner supports deploying, 'virtual\_machine', 'container', and 'bare\_metal' resources. The other configuration file is shown in listing 2. It defines the template configuration settings that specify which Spawner, Configurer, and Authenticator the MultipleSpawner should use to spawn, configure and connect to the deployed resource.

```
[
  {
    "name": "VirtualMachine Spawner",
    "resource_type": "virtual_machine",
    "providers": ["oci"],
    "spawner": {
      "class": "sshspawner.sshspawner.SSHSpawner",
      "kwargs": {
        "remote_hosts": [{"endpoint}],
        "remote_port": "22",
        "ssh_keyfile": "~/.corc/ssh/id_rsa",
        "remote_port_command": "/usr/bin/python3
        /usr/local/bin/get_port.py"
      }
    },
    "configurer": {
      "class": "corc.configurer.AnsibleConfigurer",
      "options": {
        "host_variables": {
          "ansible_user": "opc",
          "ansible_become": "yes",
          "ansible_become_method": "sudo",
          "new_username": "{JUPYTERHUB_USER}"
        },
        "host_settings": {
          "group": "compute",
          "port": "22"
        },
        "apply_kwargs": {
          "playbook_path": "setup_ssh_spawner.yml"
        }
      }
    },
    "authenticator": {
      "class": "corc.authenticator.SSHAuthenticator",
      "kwargs": {"create_certificate": "True"}
    }
  }
]
```

```

    }
  },
]

```

Listing 2: Spawner Template configuration

## 5. Results

By integrating corc into the MultipleSpawner, we enabled the architecture shown in figure 5, where the DAG service is able to dynamically schedule Jupyter Notebooks across the two resource providers. As is indicated by figure 5, the UCPH and OCI providers are defined to orchestrate resources, in this case cloud compute instances, in preparation for scheduling a requested Notebook. In order to validate that the architecture worked as expected, we setup a test environment on a separate machine. This machine was configured with a corc and JupyterHub environment, where OCI was defined as a corc provider and the MultipleSpawner as the designated JupyterHub Spawner. With this in order, the JupyterHub service was ready to be launched on the machine.

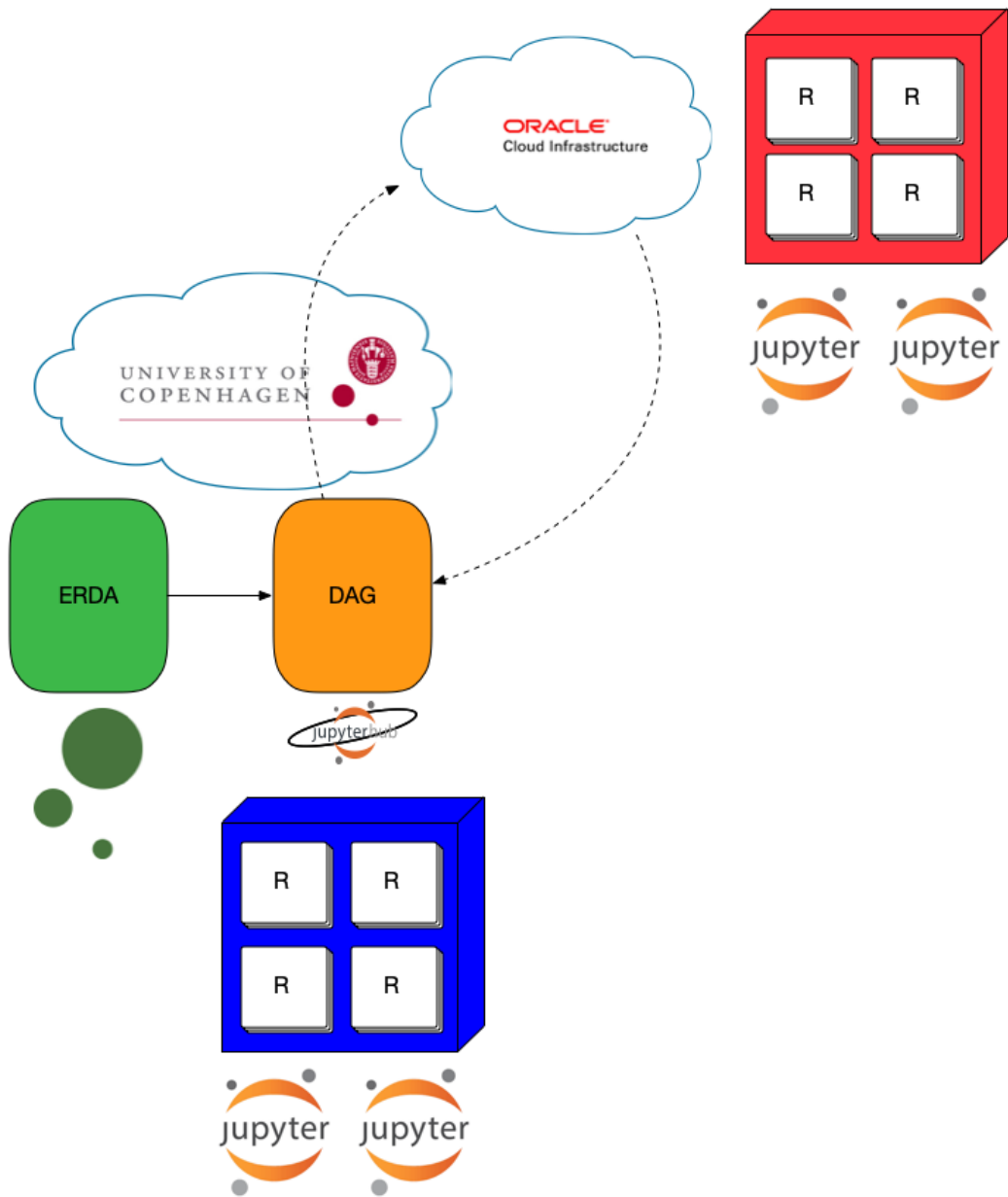
The MultipleSpawner was configured to use the template and deployment settings defined in listing 1 and 2. This enables the MultipleSpawner to create Virtual Machine cloud resources at the OCI. Subsequently, the MultipleSpawner uses the SSHSpawner [59] created by the National Energy Research Scientific Computing (NERSC) Center to connect and launch the Notebook on the orchestrated resource. Prior to this, it uses the corc defined SSHAuthenticator and AnsibleConfigurer to ensure that the MultipleSpawner can connect to a particular spawned resource and subsequently configure it with the necessary dependencies.

An example of a such a spawn with the specified requirements can be seen in figure 6. To validate that this resource had been correctly orchestrated, the corc CLI was utilized to fetch the current allocated resources on OCI. Listing 3 shows that an instance with 12 oracle CPUs, 72 GB of memory and one NVIDIA P100 GPU had been orchestrated. This reflects the minimum shape that could be found in the EU-FRANKFURT-1-AD-2 availability domain that met the GPU requirement.

```

rasmusmunk$ corc oci orchestration instance list
{
  "instances": [
    {
      ...
      "availability_domain": "lfc b :EU-FRANKFURT-1-AD-2",
      "display_name": "instance20201018103638",
      "image_id": "ocid1.image.oc1.eu-frankfurt....",
      "shape": "VM.GPU2.1",
      "shape_config": {
        ...
        "gpus": 1,
        "max_vnic_attachments": 12,

```

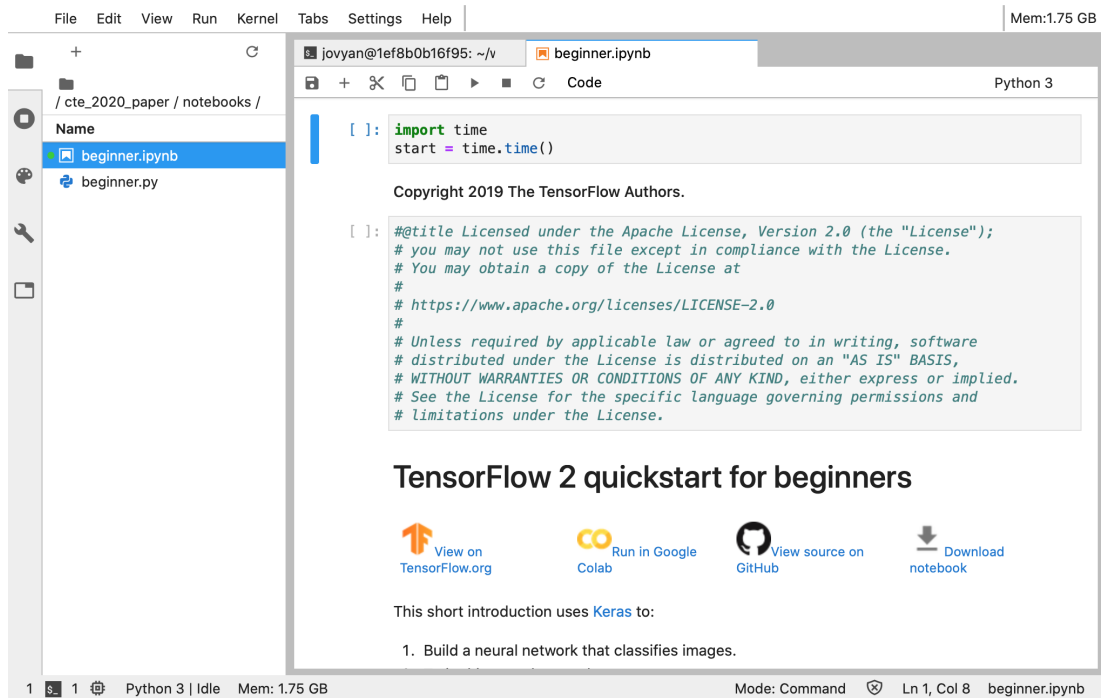


**Figure 5:** DAG MultipleSpawner Architecture, R = Resource

```

        "memory_in_gbs": 72.0,
        "ocpus": 12.0,
    },
}
],
"status": "success"

```



**Figure 6:** MultipleSpawner Interface

}

### Listing 3: Running OCI Notebook Instance

As shown in figure 7, the JupyterHub spawn action redirected the Web interface to the hosted Notebook on the cloud resources. Relating this to the mentioned courses at UCPH, this then enabled the students with access to an interactive programming environment via the JupyterLab interface.

Building upon this, a simple benchmark was made to evaluate the gain in getting access to a compute resource with a NVIDIA P100 GPU. A Notebook with the Tensorflow and Keras quick start application [60] was used to get a rough estimate of how much time would be saved in building a simple neural network that classifies images. Listing 5, shows the results of running the notebook on the GPU powered compute resource for ten times in a row, and listing 4 shows the results of running the same benchmark on an existing DAG resource. As this shows, the GPU version was on average 24,7 seconds faster or in other words gained on average a 2,8 speedup compared to the DAG resource without a GPU.

## Server Options

**Provider:**  
Oracle Cloud

**Resource Type:**  
Virtual Machine

**Resource Specification:**  
Amount of memory (GB):  
1  
Number of CPU cores:  
1  
Number of GPUs:  
1

**Session Configuration:**  
How many minutes should it run for?:  
120

Start

Figure 7: A Tensorflow + Keras Notebook on an OCI resource

```
(python3) jovyan@d203812f76e8:~/work/cte_2020_paper/notebooks$ \  
> python3 beginner.py  
Took: 38.107945919036865  
Took: 36.123350381851196  
Took: 37.37455701828003  
Took: 37.69051790237427  
Took: 41.16242790222168  
Took: 37.24052095413208  
Took: 38.685391902923584  
Took: 40.02782320976257  
Took: 38.40936994552612  
Took: 39.34704780578613  
Average: 38.41689529418945
```

Listing 4: DAG compute resource Tensorflow times



```

(python3) jovyan@56e3c30c2af6:~/work/cte_2020_paper/notebooks$ \
> python3 beginner.py
Took: 19.479900360107422
Took: 12.859123706817627
Took: 13.047293186187744
Took: 13.296776056289673
Took: 13.002363204956055
Took: 13.118329048156738
Took: 13.067508935928345
Took: 13.089284658432007
Took: 13.160099506378174
Took: 13.032178401947021
Average: 13.715285706520081

```

Listing 5: OCI GPU compute resource Tensorflow times

From this simple benchmarking example, we can see that by utilizing the `MultipleSpawner` in combination with `corc`, users are able to get access through a simple gateway to the expected performance gains of accelerators like a GPU. Expanding on this, the teachers and students at UCPH will now be able to request a compute resource with a GPU on demand, thereby gaining simple access to achieving similar faster runtimes in their exercises and assignments.

## 6. Conclusions and Future Work

In this paper, we presented our work towards establishing a Grid of Clouds that enables organisations, such as educational institutions to share computational resources amongst themselves and external collaborators. To accomplish this, we introduced `corc` as a basic building block enables the ability to orchestrate, authenticate, configure, and schedule computation on a set of resources by a supported provider.

OCI was the first provider we chose to support in `corc`, foremost because of the existing collaboration with UCPH and the associated credits that got donated to this project. This enabled us to utilize said provider to cloud enable part of the DAG service at UCPH. This was made possible through the introduction of the `MultipleSpawner` package that utilized `corc` to dynamically chose between supported cloud providers. We demonstrated that the `MultipleSpawner` was capable of scheduling and stopping orchestrated and configured resources at OCI via a local researcher’s machine.

In terms of future work, the next step involves the establishment of a Grid layer on top of the UCPH and OCI clouds. This Grid layer is planned to enable the establishment of a federated pool of participating organisations to share their resources. By doing so, we will be able to dynamically utilize cross organisation resources for services such as DAG, allowing us for instance to spawn Notebooks across multiple institutions such as other universities. Enabling the sharing of underused resources across the Grid participants. To accomplish this, `corc` also needs to be expanded to support additional providers, foremost through the integration of the Apache `libcloud` [61] library which natively supports more than 30 providers, we will allow `corc`

and subsequently the MultipleSpawner to be utilized across a wide range of cloud providers.

## Acknowledgments

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 765604. Furthermore, many thanks is given to Oracle for donating the cloud resources that made this project possible.

## References

- [1] A. Gupta, L. V. Kale, F. Gioachin, V. March, C. H. Suen, B. Lee, P. Faraboschi, R. Kaufmann, D. Milojevic, The who, what, why, and how of high performance computing in the cloud, in: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, volume 1, 2013, pp. 306–314. doi:10.1109/CloudCom.2013.47.
- [2] B. Vinter, J. Bardino, M. Rehr, K. Birkelund, M. O. Larsen, Imaging data management system, in: Proceedings of the 1st International Workshop on Next Generation of Cloud Architectures, CloudNG:17, Association for Computing Machinery, New York, NY, USA, 2017. URL: <https://doi.org/10.1145/3068126.3071061>. doi:10.1145/3068126.3071061.
- [3] D. Häfner, R. L. Jacobsen, C. Eden, M. R. B. Kristensen, M. Jochum, R. Nuterman, B. Vinter, Veros v0.1 – a fast and versatile ocean simulator in pure python, Geoscientific Model Development 11 (2018) 3299–3312. URL: <https://gmd.copernicus.org/articles/11/3299/2018/>. doi:10.5194/gmd-11-3299-2018.
- [4] P. Padoan, L. Pan, M. Juvela, T. Haugbølle, S. Nordlund, The origin of massive stars: The inertial-inflow model, Astrophysical Journal 900 (2020). doi:10.3847/1538-4357/abaa47.
- [5] University of Copenhagen policy for scientific data, Technical Report, University of Copenhagen, Copenhagen, 2014. URL: <https://kunet.ku.dk/arbejdsmraader/forskning/data/forskningsdata/Documents/Underskrevetogendeligversionafpolitikforopbevaringafforskningsdata.pdf>.
- [6] University of Copenhagen, SCIENCE AI Centre, 2020. URL: <https://ai.ku.dk/research/>.
- [7] University of Antwerp, High Performance Computing CalcUA, 2020. URL: <https://www.uantwerp.be/en/core-facilities/calcu/>.
- [8] Lund University, LUNARC: Lund University Computing Center, 2020. URL: <https://www.maxiv.lu.se/users/it-services/lunarc/>.
- [9] VSC - Vienna Scientific Cluster, VSC - Vienna Scientific Cluster, 2009. URL: <https://vsc.ac.at//access/>.
- [10] The University of Edinburgh, ARCHER2 on-demand, 2019. URL: <https://www.epcc.ed.ac.uk/facilities/demand-computing/archer2>.
- [11] Directorate-General for Communications Networks, Content and Technology (European Commission), State of the Union 2020. EuroHPC: The European Joint Undertaking on High-Performance Computing, Technical Report, 2020. URL: <https://op.europa.eu/en/publication-detail/-/publication/dff20041-f247-11ea-991b-01aa75ed71a1/language-en>. doi:10.2759/26995.

- [12] I. Foster, C. Kesselman, High Performance Computing: From Grids and Clouds to Exascale, volume 20 of *Advances in Parallel Computing*, IOS Press, 2011, pp. 3 – 30. doi:10.3233/978-1-60750-803-8-3.
- [13] R. Munk, corc: An open source tool for orchestrating Multi-Cloud resources and scheduling workloads, 2021. URL: <https://github.com/rasmunk/corc>.
- [14] Instructure, Canvas LMS, 2021. URL: <https://www.instructure.com/canvas/about>.
- [15] J. Bardino, M. Rehr, B. Vinter, R. Munk, ERDA, 2021. URL: <https://www.erda.dk>.
- [16] Rasmus Munk, jupyter\_service, 2020. URL: [https://github.com/ucphhpc/jupyter\\_service](https://github.com/ucphhpc/jupyter_service).
- [17] G. Zaccane, M. R. Karim, A. Menshawy, Deep Learning with TensorFlow, Packt, 2017, p. 320.
- [18] University of Copenhagen, Introduction to Computing for Physicists, 2021. URL: <https://kurser.ku.dk/course/nfya06018u/>.
- [19] University of Copenhagen, Applied Statistics: From Data to Results, 2021. URL: <https://kurser.ku.dk/course/nfyk13011u/>.
- [20] University of Copenhagen, High Performance Parallel Computing, 2021. URL: <https://kurser.ku.dk/course/nfyk18001u/>.
- [21] J. Berthold, J. Bardino, B. Vinter, A principled approach to grid middleware, in: Y. Xiang, A. Cuzzocrea, M. Hobbs, W. Zhou (Eds.), *Algorithms and Architectures for Parallel Processing*, volume 7016 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2011, pp. 409–418. doi:10.1007/978-3-642-24650-0{\\_}35.
- [22] Project Jupyter, About us, 2021. URL: <https://jupyter.org/about>.
- [23] F. Perez, B. E. Granger, IPython: A system for interactive scientific computing, *Computing in Science Engineering* 9 (2007) 21–29. doi:10.1109/MCSE.2007.53.
- [24] Project Jupyter, JupyterLab Documentation, 2018. URL: <http://jupyterlab.readthedocs.io/en/stable/>.
- [25] Project Jupyter, JupyterHub, 2020. URL: <https://pypi.org/project/jupyterhub/>.
- [26] J. Crist, Spawners, 2019. URL: <https://github.com/jupyterhub/jupyterhub/wiki/Spawners>.
- [27] wrapspawner for Jupyterhub, 2020. URL: <https://github.com/jupyterhub/wrapspawner>.
- [28] S. Proskura, S. Lytvynova, The approaches to web-based education of computer science bachelors in higher education institutions, volume 2643, CEUR-WS, 2020, pp. 609–625. URL: <http://ceur-ws.org/Vol-2643/paper36.pdf>, 7th Workshop on Cloud Technologies in Education, CTE 2019 ; Conference Date: 20 December 2019.
- [29] GitHub, Where the world builds software, 2021. URL: <https://www.github.com>.
- [30] Google, Google Docs: Free Online Documents for Personal Use, 2021. URL: <https://www.google.com/docs/about/>.
- [31] Google, Welcome to Colaboratory, 2021. URL: <https://colab.research.google.com/notebooks/intro.ipynb>.
- [32] Kaggle Inc., Kaggle: Your Machine Learning and Data Science Community, 2019. URL: <https://www.kaggle.com>.
- [33] Project Jupyter, Binder, 2017. URL: <https://mybinder.org/>.
- [34] Microsoft, Microsoft Azure Notebooks, 2021. URL: <https://notebooks.azure.com>.
- [35] Sagemath, Inc., CoCalc - Collaborative Calculation and Data Science, 2021. URL: <https://cocalc.com>.
- [36] JetBrains, Datalore – Online Data Science Notebook by JetBrains, 2020. URL: <https://>

- datalore.jetbrains.com.
- [37] The Binder Team, Frequently Asked Questions – Binder 0.1b documentation, 2017. URL: <https://mybinder.readthedocs.io/en/latest/faq.html>.
  - [38] Kaggle Inc., Kaggle Notebooks Documentation, 2021. URL: <https://www.kaggle.com/docs/notebooks>.
  - [39] Google, Colaboratory: Frequently Asked Questions , 2021. URL: <https://research.google.com/colaboratory/faq.html>.
  - [40] Microsoft, Azure Notebooks Overview, 2019. URL: <http://web.archive.org/web/20200818200412/https://docs.microsoft.com/en-us/azure/notebooks/azure-notebooks-overview>.
  - [41] Microsoft, Quickstart: Create a project with a custom environment, 2018. URL: <http://web.archive.org/web/20190607015705/https://docs.microsoft.com/en-us/azure/notebooks/quickstart-create-jupyter-notebook-project-environment>.
  - [42] Sagemath, Inc., What is CoCalc?, 2021. URL: <https://doc.cocalc.com/index.html>.
  - [43] JetBrains, Billing documentation, 2021. URL: <https://datalore.jetbrains.com/documentation>.
  - [44] Kaggle Inc., Efficient GPU Usage Tips and Tricks , 2020. URL: <https://www.kaggle.com/page/GPU-tips-and-tricks>.
  - [45] Red Hat Inc., What is orchestration?, 2021. URL: <https://www.redhat.com/en/topics/automation/what-is-orchestration>.
  - [46] Terraform, Terraform Documentation , 2021. URL: <https://www.terraform.io/docs/>.
  - [47] Puppet, Powerful infrastructure automation and delivery, 2021. URL: <https://puppet.com>.
  - [48] Chef, Chef Infra, 2021. URL: <https://www.chef.io/products/chef-infra>.
  - [49] Red Hat, Inc., Ansible is Simple IT Automation , 2021. URL: <https://www.ansible.com>.
  - [50] Oracle Corporation, Oracle Cloud Infrastructure CLI, 2019. URL: <https://github.com/oracle/oci-cli>.
  - [51] Amazon Web Services, Inc., AWS Command Line Interface, 2021. URL: <https://aws.amazon.com/cli/>.
  - [52] Amazon Web Services, Inc., Tools to build on AWS: Tools for developing and managing applications on AWS, 2021. URL: <https://aws.amazon.com/tools/>.
  - [53] OpenStack, Open Source Cloud Computing Infrastructure - OpenStack, 2021. URL: <https://www.openstack.org>.
  - [54] OpenNebula Systems, OpenNebula – Open Source Cloud & Edge Computing Platform, 2021. URL: <https://opennebula.io>.
  - [55] INDIGO - DataCloud, INDIGO DataCloud , 2020. URL: <http://web.archive.org/web/20200512041341/https://www.indigo-datacloud.eu/>.
  - [56] M. Caballer, S. Zala, a. L. García, G. Moltó, P. O. Fernández, M. Velten, Orchestrating complex application architectures in heterogeneous clouds, *Journal of Grid Computing* 16 (2018) 3–18. doi:10.1007/s10723-017-9418-y.
  - [57] J. Kovács, P. Kacsuk, Occopus: a multi-cloud orchestrator to deploy and manage complex scientific infrastructures, *Journal of Grid Computing* 16 (2018) 19–37. doi:10.1007/s10723-017-9421-3.
  - [58] R. Munk, multiplespawner, 2021. URL: <https://github.com/ucphhpc/multiplespawner>.
  - [59] NERSC, sshspawner, 2020. URL: <https://github.com/NERSC/sshspawner>.
  - [60] NVIDIA, TensorFlow 2 quickstart for beginners, 2021. URL: <https://www.tensorflow.org/>

tutorials/quickstart/beginner.

[61] The Apache Software Foundation, Apache Libcloud , 2021. URL: <https://libcloud.apache.org>.