Article

A Cloud-based Data Collaborative to Combat the COVID-19 Pandemic and to Solve Major Technology Challenges

Max Cappelari [1], John Belstner [2] Bryan Rodriguez [3], and Jeff Sedayao [4,*]

[1] XPRIZE Foundation; Max.Cappellari@xprize.org
[2] Intel Corporation; john.belstner@intel.com
[3] Intel Corporation; bryan.j.rodriguez@intel.com
[4] Intel Corporation; jeff.sedayao@intel.com
* Correspondence: jeff.sedayao@intel.com; Tel +1-408-765-2935

Abstract: The XPRIZE Foundation designs and operates multi-million-dollar, global competitions to incentivize the development of technological breakthroughs that accelerate humanity toward a better future. To combat the COVID-19 pandemic, the Foundation coordinated with several organizations to make available data sets about different facets of the disease and to provide the computational resources needed to analyze those data sets. This approach fits in with the XPRIZE Foundation's pre-pandemic plans to host contests that require analysis of data sets while providing the computation resources to those data sets to make it possible for a wide range of teams to be able to compete – democratizing data and its analysis. We describe the requirements, design, and implementation of the XPRIZE Data Collaborative, a cloud-based infrastructure that enables the XPRIZE to meet its COVID-19 mission and host future data-centric competitions. We offer our experiences as a case study of a Cloud Native application from design to implementation, one that used a surprising variety of Cloud services, technologies, and design patterns, from containers to VMs to serverless computing. We include our experiences of having users successfully exercise the Data Collaborative, detailing the challenges encountered and areas for possible improvement.

Keywords: containers; virtual machines**;** cloud; COVID-19; serverless; analytics**;** software defined infrastructure

1. Introduction

The XPRIZE Foundation [1] designs and operates multi-million-dollar, global competitions to incentivize the development of technological breakthroughs that accelerate humanity toward a better future. The Foundation's mission is to inspire and empower a global community of problem-solvers to positively impact our world. XPRIZE believes solutions to the world's problems can come from anyone, anywhere.

As part of that philosophy, XPRIZE looked to develop a Data Collaborative that will support the resolution of complex, global problems. It had three broad goals:

1. Comprising a collection of unique datasets and AI tools, this Data Collaborative would democratize data and the tools to analyze them, enabling virtually anyone, anywhere to use data to solve the world's grandest challenges.
2. It would provide access to the massive amounts of data that have been organized and cleaned so that it can be accessed in an accountable, transparent and responsible manner.
3. Data owners across industries and topic areas will feel safe in sharing their data while maintaining ownership, privacy and security. Data scientists, as well as Machine Learning and Artificial Intelligence systems will all have access to the Data

Collaborative resulting in enhanced problem-solving models and innovative approaches to solving Grand Challenges posed by XPRIZE competitions.

The COVID-19 pandemic inspired the XPRIZE Foundation to leverage existing Data Collaborative plans to make COVID-19 relevant data available on it. The Foundation sponsors the XPRIZE Pandemic Alliance [2], a cooperative of different organizations that would contribute data and make that available for analysis anywhere. The Data Collaborative would first be used for COVID-19 efforts and then be reused for XPRIZE competitions.

There are a number of existing environments and services similar to the Data Collaborative. Google's Kaggle [3] is a web-based environment that providers an analytics environment to users along with a large collection of publicly data sets. Like the Data Collaborative, is used for hosting competitions, focusing on machine learning oriented problems. Google also maintains Colab [4], which provides a Python oriented environment. Available commercial services for hosting analytics environments include CoCalc [5], Azure Notebook [6] (to be discontinued as a service in January 2021), and Nextjournal [7].

We present the design and implementation of the XPRIZE Data Collaborative as a case study of a Cloud Native Application (CNA) [8], from requirements to design to implementation to real operating experiences. Kratzke and Quist, in their study of the Cloud Native Applications, list the goals of CNA like elasticity and scaling, and emphasize isolating state with microservice architectures [9]. They also mention key features of a CNA to be "softwareization of infrastructure," also known as Software Defined Infrastructure (SDI) [10]. Gannon, Barga, and Sunaresan [11] agree that microservices and containers are a key component, but newer Cloud paradigms such as serverless computing needed to be considered. One of the surprises of our implementation is that we ended up utilizing many CNA features such as containers and SDI, but also serverless computing and older IaaS paradigms as virtual machines all in the same application. As we describe our implementation choices, we match those choices with common, well known design patterns.

In this first section, we have described critical background information such as the XPRIZE foundation's mission and goals for the Data Collaborative. We also described similar and related services that currently exist and provided context of Cloud Native Applications with which we will frame our case study. Section 2 focuses on detailed requirements of the Data Collaborative and the infrastructure design chosen to meet those needs. In Section 3, we look our experiences with this implementation, detailing the challenges we encountered and possible changes for future competitions.

2. Detailed Requirements and Architecture

In the introductory section of this paper, we broadly described the requirements and goals the Data Collaborative architecture. In this section, we will refine these into detailed needs, in order to motivate the design choices. This process will also enable us to measure the success of our design and implementation.

2.1. Contest lifecycle

The lifecycle of an XPRIZE Foundation contest needs to be comprehended in order to design an optimal infrastructure. In general, Figure 1 shows the flow of a contest:

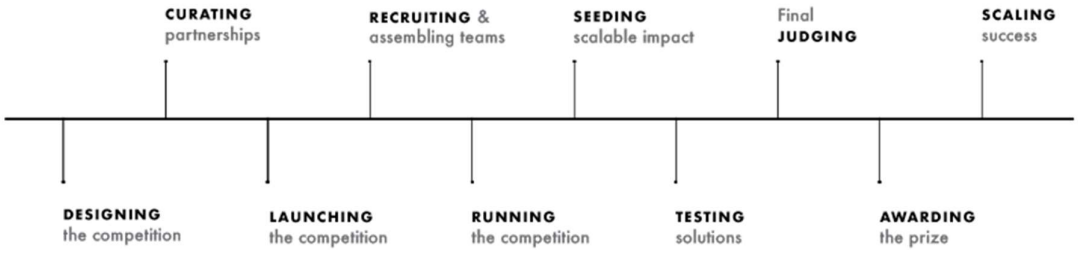

Figure 1: Typical lifecycle of an XPRIZE Competition

Contests have a definitive start and finish, so apart from the obvious need to assemble the infrastructure necessary for a contest, that infrastructure needs to be maintained and then taken down. The recruitment and team assemble phases require teams to register on an XPRIZE portal. During a competition, there are typically a number of rounds where progress is judged and the number of competitors is reduced. A contest could have as many as 5000 teams in the initial round, and there can be simultaneous contests of varying duration and in different stages of their lifecycle at any one time.

2.2. Data Collaborative Detailed Requirements

Given the broad requirements in section one and information about contest/challenge lifecycle, we generate more detailed requirements of what needs to be implemented:

1. **A highly scalable, accessible, and elastic infrastructure**: A single competition could involve as many as thousands of teams at one time, and as each competition milestones are reached, the number could be reduced by one or two orders of magnitude. Teams can be from almost anywhere in the world. The ability to rapidly scale up and down is required.
2. **A stable and highly usable analytics software platform**: Teams in XPRIZE competitions will need to be able to not just access data but analyze it. It should be familiar to many users and extendable.
3. **Isolated and secure analytics software platform**: Since awards in XPRIZE competitions can be multi-millions of dollars, teams and their work need to be effectively isolated from each. In addition, since we are providing the infrastructure for teams to run arbitrary code, we need to make sure that what we provide is not abused – not used as a base for attacks or noncompetition related work like crypto-mining.
4. **A scalable analytics software platform**: While the commons goal is to democratize data access and the ability to analyze that data, should a team wish to purchase more capacity, the platform should enable that.
5. **Control of data**: Contest sponsors and others want to be able to protect the data they provide and know who accesses it. The data should remain within the analytics platform, with it being difficult and time consuming to copy it outside of the analytics platform. Access to data needs to be recorded and attributable to a team.
6. **Manageability**: The commons infrastructure needs to be maintainable by a relatively small staff.
7. **Reasonably fast implementation**: The infrastructure needed to be stood up in a reasonably fast amount of time in order to make a difference in the Pandemic.
8. **Costs**: Costs should be minimized when possible.

We will refer to these requirements as we make design decisions.

With more detailed requirements, we enumerate what decisions need to be made. The following components and processes need to be implemented in order to make the Data Collaborative viable:

1. User analytics software platform
2. Team isolation
3. Naming Design and Infrastructure
4. User Authentication
5. Protecting data in transit (e.g. TLS)
6. Logging and Monitoring
7. Team Infrastructure Instantiation Process

We document the design options and final choices for each item above in the next section.

2.3. *Design Choices*

With a set of requirements and another set of components and processes that need to be selected, we outline choices and the thinking that goes behind them. We will step through each of the required items, discussing the possible options, choosing an option, and then justifying that decision in terms of the requirements defined above. We also point out when we use some of the more common cloud design patterns [12].

2.3.1. User analytics Software Platform

Requirement #2 (A stable and highly usable analytics software platform) drove our selection for an analytics software platform. Notebooks have become a crucial tool for data scientists [13], as a way of developing analytics code, visualizing analytics results, and sharing insights. NBView, a tool that estimates the number of publicly available notebooks, estimates that there are more than 9.5 million notebooks publicly available today on GitHub alone [14].

We looked at two choices for the standard Data Collaborative. The first choice was a Jupyter notebook [15], a web-based platform for analytics. The other choice was Zeppelin [16], another web-based platform for analytics and Apache project that runs on top of the Spark [17] analytics system. Jupyter notebooks are very widely used, stable, and has a larger community and eco-system around it. Although the community for the Zeppelin is growing, it isn't as large.

Another factor in analytics platform selection was requirement #4. Enabling teams to be able purchase more resources for their project made it necessary to isolate notebooks into units that could more resources. This was difficult to do quickly (requirement #7) in highly integrated multiuser implementations of Jupyter notebooks like JupyterHub [18]. Zeppelin notebooks also are tightly integrated with Spark. Jupyter notebooks can be implement on a standalone basis [18] (not part of JupyterHub). That property, along with the popularity of Jupyter notebooks, made the standalone Jupyter notebooks our platform of choice.

2.3.2. Team Isolation

As mentioned above, teams competing for large monetary prizes need to be effectively isolated from each other (requirement #3). The two most viable placement choices for each team's notebook are within VMs or within a container. Note that putting each team on an individual server would provide maximum isolation but would meet neither requirement #8 (prohibitively expensive) nor requirement #1 (not scalable). Using VMs would provide better isolation than using containers but would be more resource intensive and therefore more expensive. In the end, we deemed that containers have sufficient isolation at a better price point.

To implement requirement #4, each container would be in its own resource group. That allows teams to add resources to the resource group by working directly with the Cloud service provider.

2.3.2. Naming Design and Infrastructure

Our choice to use standalone Jupyter notebooks, rather than an integrated environment like JupyterHub, forced us to find an effective way to direct teams to their individual notebooks. We choose to hand each team a unique DNS name for their notebook, with the name being in the xprize.org domain. Rather than constantly make changes for each new team that was added or removed, we used a wild-card domain record in xprize.org that points all teams in a competition to an application load balancer in the Cloud. Web requests for a notebook arrive at the application load balancer (gateway) and are sent to a proxy server, which routes the request to the container containing the team's notebook based on the name of the container that we gave to that team. We show this flow and necessary infrastructure in Figure 2, which uses the gatekeeper design pattern [19].

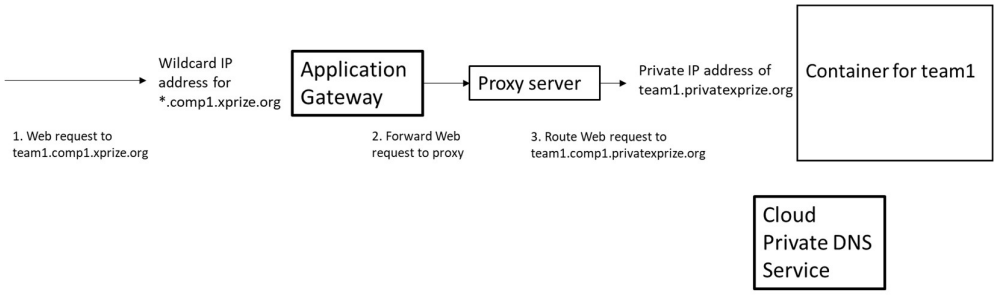

Figure 2: Routing of web requests to team notebook container

A team's container holding their notebook is in a private IP network space. In order to route traffic to that container from the application load balancer, we need a way to name the container and register that name into a private DNS space. Our cloud provider has a DNS service that can be manipulated programmatically, so we use a private DNS domain for our containers' names. This architecture also requires a proxy server which is used to route traffic.

2.3.3. User Authentication

Authentication was made somewhat more difficult since the official multiuser solution for Jupyter notebooks is JupyterHub, which we decided not to use. Since we used the standalone Jupyter notebook, we needed to use one of the available authentication methods available on it. Notebooks have two options for user authentication:

1. Embedded password
2. Authentication token

The embedded password would be difficult to manage (not meeting requirement #6), especially when a password would need to be changed. Some process would need to take the new password and replace it inside of the container. The second option was much more viable. XPRIZE already had portal for teams to sign up, and we could use that portal to generate the token for a new notebook and then handoff that token to the team. This also leverages a standard design pattern, federated identity [20].

2.3.4. Protecting Data in Transit

To enforce the isolation (requirement #3) and to help control access to data (requirement #5), we need to protect data in transit. In our case, this means encrypting data in transit – using Transport Layer Security (TLS) [21]. As this is a common practice, the chief design decision came in where to put the certificate and do the encryption/decryption – the TLS endpoint. We looked at the following options:

1. On each notebook container
2. On a sidecar container

3. Proxy
4. Application gateway

Option 1 is hard to manage, especially when certificates would need to change, as we would need a way to reach into every container to update the certificate. The sidecar container pattern [22] could be used – this puts up container next to each notebook container and handles being a TLS endpoint. This is easier to manage than the first option since all of those TLS sidecar containers would be very similar, but it drives up the number of containers and costs (requirement #8) and results in twice as many containers to manage. Putting the TLS endpoint on the proxies is a better option. This is a function that many proxies can perform, and there are far fewer proxies to manage for each competition. A better option is the last option – putting the endpoint on the Application Gateway. This leaves only one place to put the certificate – much easier to manage. It lessens the amount of processing done be containers and the proxies, reducing cost (requirement #8). We selected this option, which also is a standard cloud design pattern - gateway offloading [23].

2.3.5. Logging and monitoring

Putting each team in a notebook in a container enabled us to use some Cloud Native monitoring functions available from our cloud service provider. In particular, the provider has services that check for inappropriate uses of the compute facilities we provide, such as looking for cryptomining. The provider also has centralized logging repositories that we can send operational data, providing a single place to look for anomalies and investigate incidents. We chose to utilize both of these capabilities.

2.3.6. Instantiating team environment

When a new team receives a new notebook, there needs to some way of instantiating the environment for each team. This is involves running scripts that build the individual components for a team. Teams would be notified and received the authorization token after their infrastructure would be ready – this would an asynchronous process. The main design issue becomes where to run the scripts. The following options were available:

1. Run the instantiation scripts on XPRIZE portal.
2. Create a dedicate VM or container to run the scripts
3. Define a function to run the scripts and launch it in the cloud (serverless)

While option #1 is certainly possible and some processing has to be done on the portal (to provide the token), running the entire instantiation process could tie up resources there. Option #2 of creating a whole VM or container is viable, but creating a new one every time a notebook has to be instantiated will take time and cost money, while leaving one around ready to process notebook instantiation requests leaves dedicated resources idle. We choose option #3, creating a function that instantiates that notebook and informs the team when it is ready. The requirements fit in well with the serverless design pattern of Asynchronous background processing and messaging [24]. We took advantage of serverless computing options offered by our Cloud Service Provider to make this happen.

2.4. Architecture and Component Layout

Figure 3 shows the overall architecture of the components we described in the infrastructure for a single competition. Each team's notebook shared the same public IP address for, which is an address on an application gateway. The gateway has multiple functions, such as load balancing traffic to the proxy servers, running a Web Application Firewall, and being a TLS endpoint. We use reverse proxies on VMs to balance and filter traffic to the notebook containers, restrict the kind of operations permitted on data (requirement #5), and provide additional layers of isolation from the Internet. We mode the choice of VMs rather than containers as we ran into issues containerizing the proxy application. Also, as we were developing the application, we needed to log into the proxy periodically and make changes to configurations – this was much easier to do when they were in

VMs rather than containers. The proxy servers are on a segment that has access restrictions. The containers containing teams' notebooks are also on a separate segment. This segment has access lists that restrict where they can connect to on the Internet. Note that Teams are allowed to download software to their notebooks from some a limited number of software repositories. An additional firewall controls access form the containers to the data sets used as the basis for competition.

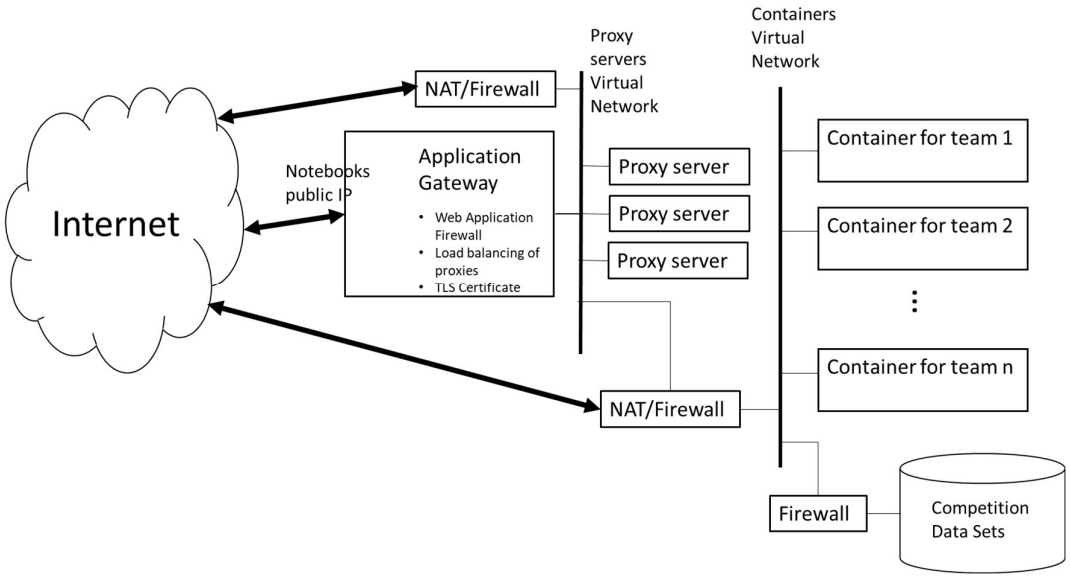

Figure 3: Component Architecture and Layout for each Competition

The infrastructure in Figure 3 is intended to be duplicated for each competition. We show this in Figure 4. This minimizes the opportunity for data sets from one competition leaking into another. It also reduces the effects of any single configuration mistake from affecting all competitions. This fits into the cloud design pattern called the Bulkhead pattern [25]

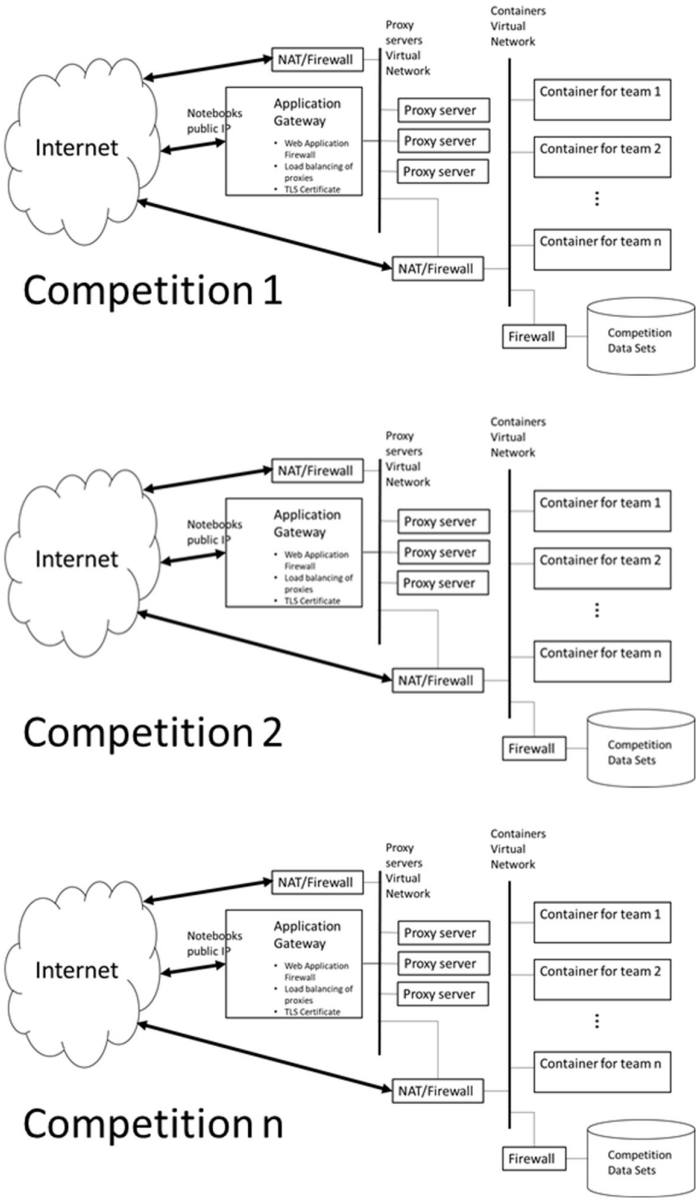

Figure 4: Separating infrastructure between competitions

One notable feature of our design is that we exercised a number of different cloud technologies. Table 1 lists components and types of cloud techniques used. While one would expect an almost total use of containers and microservices, we used a mix of Cloud technologies, including VMs and serverless.

Component	Cloud Technology
Team Notebooks	Container
Proxy Servers	VMs
Firewalls	Software Defined Infrastructure

TLS Endpoint	Software Defined Infrastructure
DNS	Software Defined Infrastructure
Notebook Instantiation	Serverless

Table 1: Infrastructure components and Cloud Technologies used

3. Results

The Data Collaborative is in active use, for both COVID-19 work [2] and for AI [26]. We ran a small competition of about 25 competitors. While the data collaborative works, but we did encounter issues. We had concentrated on setting up the environment, but we did not focus much what would take to it deprovision after a contest. Containers need to be deleted, and DNS entries should be removed. That scripting needs to take place – perhaps again run as a serverless function that will do all of the cleanup after a contest.

The notebook per container architecture had many benefits, such as improving notebook isolation and using native cloud container monitoring, but it had the drawback of being harder to maintain. If a software upgrade needed to be made to all of the notebooks in a competition, then we would have to go through each notebook, upgrade it in place if possible, and create new copies and then redeploy if not. We want to examine how much of requirement #4 is really needed, and whether we can use a more centralized approach using JupyterHub to simplify much of the architecture and operation.

Our proxy servers currently are in VMs. Work needs to be done to automate their configuration and sending log output in a standard format. Once that is completed, they should be moved to containers to simplify management and use fewer resources.

Another aspect that should be examined is how critical performance will be. We have concentrated on getting something working, but we have not focused on how important it will be to make analytics run as fast as possible.

4. Discussion

We hope this case study will be useful to those studying Cloud Computing implementation. Our work shows how application requirements can shape what cloud technologies and design patterns that are used. Those requirements can mandate using different cloud technologies, including older ones like VMs that seem to be increasingly dismissed as focus increases on microservices and containers and newer ones, like serverless computing.

Acknowledgments: We would like to thank Felix Labunsky of Microsoft for reviewing our architecture and providing some guidance. We would also like to thank Dan Gutwein of Intel for helping to drive this project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. XPRIZE Foundation. Available Online. https://www.xprize.org/ (accessed on 28 October 2020).
2. Accelerating Radical Solutions to COVID-19 and Future Pandemics. Available online. https://www.xprize.org/fight-covid19 (accessed on 28 October 2020).
3. Kaggle. Available Online. https://kaggle.com/ (accessed on 5 November 2020).
4. CoLab. Available Online. https://colab.research.google.com/notebooks/intro.ipynb (accessed on 5 November 2020).
5. CoCalc. Available Online. https://cocalc.com/ (accessed on 5 November 2020).
6. Azure Notebooks. Available Online. https://notebooks.azure.com/ (accessed on November 5, 2020).

7. Nextjournal. Available Online. https://nextjournal.com/ (accessed on November 5, 2020).
8. Andrikopoulos V, Fehling C, Leymann F. Designing for CAP-The Effect of Design Decisions on the CAP Properties of Cloud-native Applications. In the Proceedings of 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012), Porto, Portugal, 18-21 April 2012; pp. 365-374.
9. Kratzke N., Quint P.C. Understanding cloud-native applications after 10 years of cloud computing-a systematic mapping study. *Journal of Systems and Software* **2017**, *126*, 1-6.
10. Kang, J.M., Lin, T., Bannazadeh, H. and Leon-Garcia, A., 2014, May. Software-defined infrastructure and the SAVI testbed. International Conference on Testbeds and Research Infrastructures, Guangzhou, China 5-7 May 2014; Springer, 3-13.
11. Gannon D, Barga R, Sundaresan N. Cloud-native applications. *IEEE Cloud Computing*. **2017**, 4, 16-21.
12. Cloud Design Patterns. Available Online. https://docs.micrsoft.com/en-us/azure/architecture/patterns/.
13. Meyers, A. Data Science Notebooks – A Primer. Available Online. https://medium.com/memory-leak/data-science-notebooks-a-primer-4af256c8f5c6/ (accessed on 8 November 2020)
14. Esimate of Public Jupyter Notebooks on GitHuV. Available Online. (accessed on 8 November 2020).
15. Jupyter Notebook. Available Online. https://jupyter.org/ (accessed on 6 November 2020).
16. Apache Zeppelin. Available Online. https://zeppelin.apache.org/ (accessed on 7 November 2020).
17. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I. Spark: Cluster computing with working sets. *HotCloud* **2010**, 10, 95-101.
18. Jupyterhub. Available Online: https://jupyter.org/hub (accessed on 8 November 2020).
19. Gatekeeper Pattern. Available Online: https://docs.microsoft.com/en-us/azure/architecture/patterns/gatekeeper (accessed on 9 November 2020).
20. Federated Identity Pattern. Available Online: https://docs.microsoft.com/en-us/azure/architecture/patterns/federated-identity (accessed on 9 November 2020).
21. Rescorla E, Dierks T. RFC 8446: The transport layer security (TLS) protocol version 1.3. Available Online: https://tools.ietf.org/html/rfc8446 (accessed on 11 November 2020).
22. Sidecar pattern. Available Online: https://docs.microsoft.com/en-us/azure/architecture/patterns/sidecar (accessed on 9 November 2020).
23. Gateway Offloading pattern. Available Online: https://docs.microsoft.com/en-us/azure/architecture/patterns/gateway-offloading (accessed on 9 November 2020).
24. Asynchronous background processing and message. Available Online: https://docs.microsoft.com/en-us/dotnet/architecture/serverless/serverless-design-examples#asynchronous-background-processing-and-messaging (accessed 11 November 2020).
25. Bulkhead pattern. Available Online: https://docs.microsoft.com/en-us/azure/architecture/patterns/bulkhead (accessed on 9 November 2020).
26. AI and Data for the Benefit of Humanity. Available Online: https://xprize.org/aialliance (Accessed on November 9, 2020).