Article

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

Max Cappellari [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. 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 developed during the pandemic, from motivations and design to implementation. In contrast to previous Cloud deployment studies that focus on implementations of containers and microservices or serverless architecture, we describe how and why we used both containers, serverless, and other Cloud technologies, even older ones such as Virtual Machines. We include our experiences of having users successfully exercise the Data Collaborative, detailing the challenges encountered and areas for improvement.

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

---

1. Introduction

The XPRIZE Foundation [1-3] 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. The Foundation sponsors the XPRIZE Pandemic Alliance [4,5], 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.

Other existing environments and services exist that are similar to the Data Collaborative. Google's Kaggle [6] is a web-based environment that providers an analytics environment to users at large along with a large collection of publicly data sets. Like the Data Collaborative, is used for hosting competitions [7], focusing on machine learning oriented problems. Google also maintains Colab [8,9], which provides a Python oriented environment. Available commercial services for hosting analytics environments include CoCalc [10,11] and Nextjournal [12]. The XPRIZE Foundation differs from these operations by not focusing on individual data scientists or consumers and instead focusing on fewer, higher impact technology challenges run by teams of people.

We present the design and implementation of the XPRIZE Data Collaborative as a case study of a Cloud Native Application (CNA) [13-16], from requirements, design, and implementation to operating experiences. Many case studies and surveys of Cloud implementations focus on particular categories of Cloud Technology, such as microservices and container technologies [15,17,18] and serverless computing [19-21], but our presentation differs as we show how, why, and where they are used together in one application. In addition, we show how and why other cloud technologies, such like Software Defined Infrastructure (SDI) [22] and older, seemingly deprecated technologies like Virtual Machines (VMs), are integrated into an application. As we describe our implementation choices, we match those choices with common, well-known design Cloud Design patterns [14,15,20,21,23]. In the end, we evaluate our implementation, outline where our operational experiences uncovered design mistakes, how we dealt with those problems in the short term, and where the Data Collaborative needs to go in the future.

Another feature of our paper is a description of how the COVID-19 pandemic affected the Data Collaborative design, development, and implementation. The pandemic affected not only what problems the Collaborative would initially deal with but with also requirement priorities. There are also a number of nonrequirements that were also driven from the Pandemic - these are concerns that are well known to Cloud Computing practitioners but were given low or no priority for this particular effort.

In this first section, we have described critical background information such as the XPRIZE foundation's mission and goals for the Data Collaborative. We list similar and related services that currently exist and provided context regarding Cloud Native Applications with which we will frame our case study. Section 2 briefly describes the materials and methods used to create the Data Collaborative. Section 3 focuses on detailed requirements of the Data Collaborative and the infrastructure design chosen to meet those needs. In Section 4, we discuss our experiences with this implementation, detailing the challenges we encountered, how we handled those challenges, and possible changes for future competitions.

2. Materials and Methods

We designed and implemented the Data Collaborative using openly available commercial Cloud services and open source software. We do not specifically use the names of Cloud Service Providers used and instead describe the general approaches, based on well-known Cloud Design patterns that can be implemented on any Cloud Service Providers.

3. Results

In the introductory section of this paper, we broadly described the requirements and goals the Data Collaborative architecture. In this section, we will examine in greater depth the requirements and also some notable non-requirements, in order to motivate the design choices. This process will also enable us to measure the success of our design and implementation. From the requirements, we generate a list of specific items that need to be designed and implemented. A discussion of each item is produced, which goes over design options and which option that we chose to implement.

3.1. 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 typical flow of an XPRIZE contest:

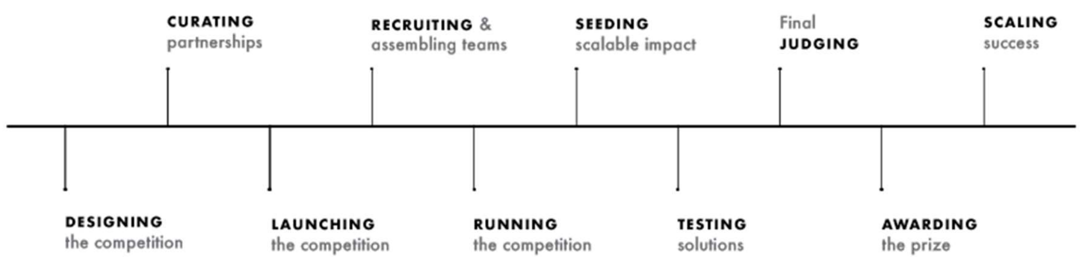

Figure 1: Typical lifecycle of an XPRIZE Competition

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. Once a competition is designed, and funded through partnerships, 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 5,000 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.

3.2. 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 compute 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
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.

3.3. Nonrequirements

The need to enable timely solutions to COVID-19 related problems created a number of what we consider **nonrequirements** - areas that we did not have as goals. Here is an itemized list:

1. **Portability between Clouds**: Extensive work has been done try to create technology that makes applications portable between Cloud Service Providers [24, 25] and avoiding vendor lock-in [26]. This goal was simply not a priority in our efforts to get the Data Collaborative up and running. We chose to use the Cloud Service Provider that we were most familiar with in an effort to get the service to production as soon as possible.
2. **Ultra-low cost:** While a reasonable cost is a key requirement (#8 above), having an extremely low cost, especially at the expense of having a usable and working solution, was not.
3. **High Performance**: While our requirement for a usable and scalable platform (requirements #1 and #2) demands enough performance to be usable, it did not demand high performance. We need the Data Collaborative in a relatively short time frame that worked and was usable. In addition, the contest time frames should allow for time for analyses to complete.

3.4. Design Choices

In the previous sections, we generated a set of requirements and a set of components and processes that need to be selected in order to create the Data Collaborative. In this section, we will step through each required items, discussing the possible options, choosing an option, and then justifying that decision in terms of the requirements defined above. These decisions will be framed, wherever possible, in terms of Cloud Design patterns.

3.4.1. 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 [27], 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 almost 10 million notebooks publicly available today on GitHub alone [28]. Platforms that offer similar functionality to our goals such Kaggle [6,7], CoCalc [8,9], and Nextjournal [10] all offer support for notebooks. Some sort of notebook seemed like an obvious choice for the Data Collaborative.

We looked at two choices for the standard Data Collaborative notebook. The first choice was a Jupyter notebook [29], a web-based platform for analytics. The other choice was Zeppelin [30], another web-based platform for analytics and Apache project that runs on top of the Spark [31] analytics system. Jupyter notebooks are very widely used, stable, and has a larger community and eco-system around it. Kaggle, CoCalc, and Nextjournal offer some varying levels of support for Jupyter notebooks. While the community for the Zeppelin is growing, it is not yet as popular.

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 [32]. Zeppelin notebooks also are tightly integrated with Spark. Jupyter notebooks can be implement on a standalone basis [32] (not part of JupyterHub). That property, along with the popularity of Jupyter notebooks, made the standalone Jupyter notebooks our platform of choice.

3.4.2. Team Isolation

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.

3.4.3. Naming Design and Infrastructure

Our choice to use standalone Jupyter notebooks instantiated in individual containers, rather than an integrated environment like JupyterHub, forced us to find an effective way to direct teams to their individual notebooks. If we had used JupyterHub, we could give all the teams a single domain name to connect to and then they would be directed to their notebook after they authenticated. Instead, we had to direct users on an individual team to the individual container for that team. We chose to do this through DNS host names in the URL for each notebook.

Figure 2 shows the infrastructure and process for how we connect teams to their notebooks. We present each team that joins a competition a unique DNS name for their notebook, with the name being in the xprize.org domain. In the example below, "team1" gets a URL such as "http://team1.comp1.xprize.org/" as step 1. Rather than create a new DNS [33] record for each new team that was added or removed, we used a wild-card domain record for *.comp1.xprize.org that points all teams in a competition to an application load balancer of our Cloud Service Provider.

An HTTPS request for a notebook arrives at the application load balancer (labeled Application gateway) and in step 2, are sent to a proxy server. In step 3, the proxy server routes the request to the container containing the team's notebook based on the name of the container that we gave to that team.

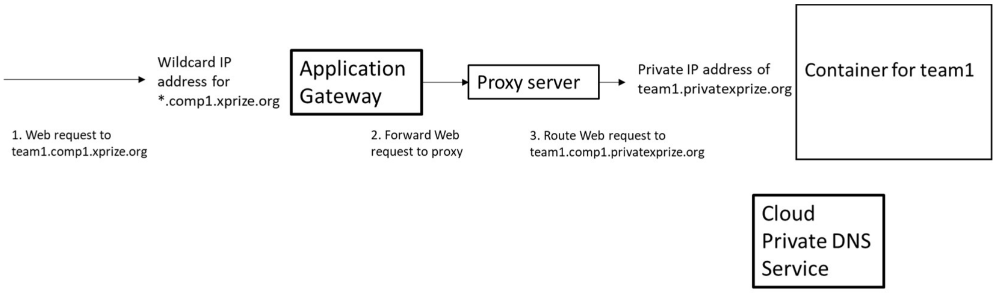

Figure 2: Routing of web requests to team notebook container

The container holding their notebook is in a private IP network space [34] which is not directly accessible to users on the Internet. In order for Step 3 to work, we need a way to name the container and register that name into a private DNS space. Our Cloud Service Provider has a DNS service that can be manipulated programmatically, so we use a private DNS domain for our containers' names. In Figure 2, this host name for the container with team1's notebook is team1.privatexprize.org.

While this level of detail may seem excessive, explaining the design of DNS namespace and architecture if important for our discussion. First, it shows how we applied the gatekeeper design pattern [35]. Second, it illustrates our use of SDI (Software Defined Infrastructure) [22], as the DNS infrastructure is allocated and loaded with domain information programmatically. Third, this infrastructure has some problematic features, and should be understood in order to put some of the problems we experienced (to be described later) into context.

3.4.4. User Authentication

Authentication of the users in a team became a design decision since the official multiuser solution for Jupyter notebooks is JupyterHub, which we decided not to use. Since we chose to give teams a standalone Jupyter notebook isolated into a container, we needed to use one of the available authentication methods available for individual notebooks. Individual Jupyter notebooks have two options for user authentication [36]:

1. Password
2. Authentication token

As a team would have to share a password, this would be insecure, violating requirement #3. Password management (e.g. forgotten passwords, managing password change and distribution software) would be additional work for the XPRIZE team to manage passwords, affecting requirement #6). XPRIZE already had a portal for teams to sign up that relied on third party authentication services from OAuth identify providers like Facebook and Google [37], and we decided use that portal to authenticate the users on a team, generate the token for a new notebook, and then distribute that token to the team members. Moreover, this matches a standard cloud design pattern, federated identity [38].

3.3.5. Protecting Data in Transit

To enforce isolation between teams (requirement #3) and to help control access to data (requirement #5), we need to protect data in transit to and from the Data Collaborative. In addition, we needed to make sure that the token that we decide to use for authentication would not traverse the Internet or Intranets in the clear. In our case, this means encrypting data in transit using Transport Layer Security (TLS) [39]. As this is a common practice with websites, 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 viable using services such as Let's Encrypt [40], but is potentially harder to manage and consumes more resources as adding the certificate process to each container increases the tasks and complexity to each container instantiation and rollout. The sidecar container pattern [41] could be used – this pattern would instantiate a new container next to each notebook container which would hold the certificate handles being a TLS endpoint, communicating the notebook container in the clear. This is easier to manage than the first option since all of those TLS sidecar containers could be identical, but it drives up the number of containers and costs (requirement #8) and results in twice as many containers to manage and monitor (affecting requirement #6). 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 – making container instantiation simpler and it only needs to be done once.. 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 [42].

3.3.6. 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. These are services not unique to this particular provider – other cloud service providers have these two capabilities. We chose to utilize both capabilities.

3.3.6. Instantiating team environment

Before a new team receives their notebook, there needs to some way of instantiating the environment for each team. Scripts need to be run that build the individual components for a team – the notebook container, the private DNS entries, and firewall rules. Teams would be notified and received the authorization token after their infrastructure is ready. Instantiating a new notebook for a new team happens sporadically. The main design question becomes where to run the scripts. The following options were available:

1. Run the instantiation scripts on XPRIZE portal.
2. Create a dedicated 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 must be done on the portal to provide the token, running the entire instantiation process ties up resources there and makes it a potential performance bottleneck and single point of failure. 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 having one 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 for Asychronous, Event-Driven processing [21,43]. We took advantage of serverless computing options offered by our Cloud Service Provider to implement this.

3.4. Architecture and Component Layout

Figure 3 shows the overall architecture of a single Data Collaborative competition, showing the placement of the components we for which we described the design. Each team's notebook shares the same public IP address, 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 made 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. Once we had the working configuration, we deployed using proxies in VMs.

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. Teams are allowed to download software to their notebooks from a limited number of software repositories. Another firewall controls access from the containers to the data sets used as the basis for competition. These data sets are also stored in the cloud on SDI.

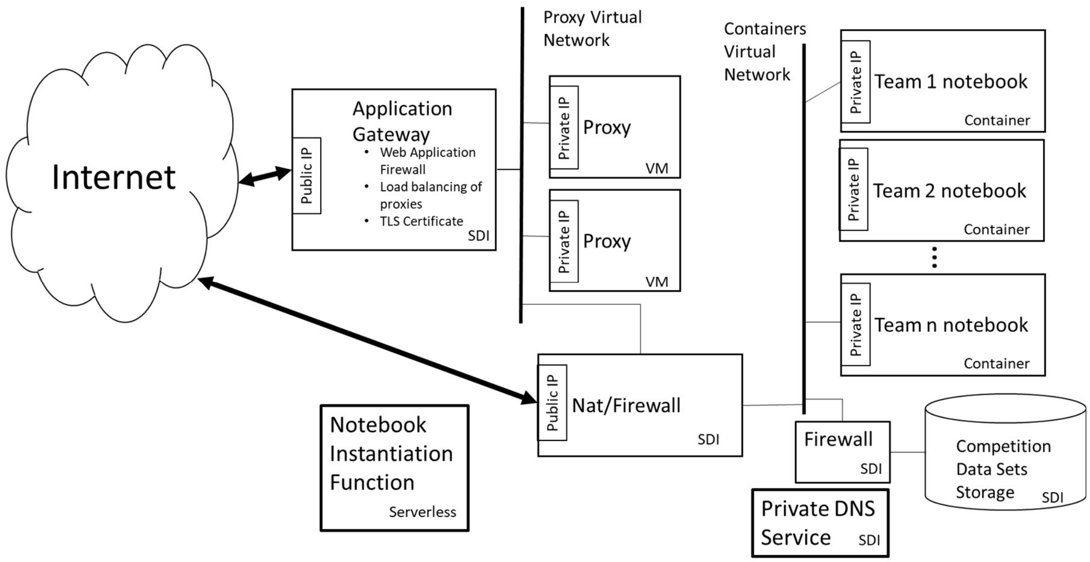

Figure 3: Architecture and Component Layout for a single Competition

The infrastructure in Figure 3 will be duplicated for each competition. 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 architectural decision matches the cloud design pattern called the Bulkhead pattern [25].

3.5. Other Implementation notes

Initial meetings with the members of the team (listed above as authors) on the Data Collaborative started in September 2019. A first workable prototype with deployment documentation was delivered in June of 2020. The four authors of this paper did not work on this project full time – there were other projects done during the time frame of this work.

4. Discussion

After completing the design and implementing the first version of the Data Collaborative, a number of key questions need to be discussed. Did the design and implementation meet our original goals? What were the areas that needed to be improved, in the short term and in the long term? How did the COVID-19 Pandemic affect the implementation? What are longer term implications for the development of Cloud Native Applications? In this section, we answer those questions.

4.1 *Evaluation of the Design and Implementation against Requirements*

Did our design and implementation of the Data Collaborative meet the goal of being a highly scalable analytics platform for competitions? It is in active use, for both COVID-19 work [4] and for general AI related challenges [45]. We first completed a small competition of about 25 teams, and are currently in the testing phase of a competition that has hosted some 300 teams [46], so it definitely is an operational utility for the XPRIZE foundation. Table 1 shows a list of the requirements in section 3.2 and whether the implementation met them. In general, requirements were fulfilled. Some requirements are hard to prove, like being "secure."

Table 1: Data Collaborative high level requirements and whether they were met

Goal	Goal Met?	Comments
1. Scalable Infrastructure	Yes	Scaled to 300 teams so far
2. Usable Analytics Platform	Yes	Used for 1 complete competition and one in progress
3. Isolated Secure platform	Possibly	Secure as far as we know – no incidents so far
4. Scalable Analytics Compute Platform (self-service capacity adds)	Probably	We have put in this capability but not yet exercised
5. Control of data	Yes	Access to data sets is controlled
6. Manageability	Yes	Manageable by a small XPRIZE team at current scale
7. Fast implementation	Yes	Started in last 2019 and first competitions in Mid 2020
8. Cost controlled	Yes	No evidence of cost issues at this point

While the Data Collaborative works and proved scaled up to a certain point, we did encounter issues. We had concentrated on setting up a competition environment under schedule pressure, but we did not focus much what would take to deprovision it after a contest. After our first competition, some configured resources, such as containers and DNS records had to be removed manually. This kind of problem is solvable through automation and scripting.

We also encountered problems regarding application state and containers. Since teams could add software packages to their notebook, any state changes such as this would be reflected of course the current state of the container. But if the container crashed for some reason and a replacement one was instantiated, any additions would be lost. This is because we did not separate out the stateful components of the notebook – additional software packages for instance. We fixed this by making sure that state changes like this would be saved and restored if the container went down and had to be restarted.

A more subtle problem with container state happened because of the DNS architecture shown in Figure 2. A container's name in our private DNS space has a particular Time To Live (TTL). As an example, let's say record for the container named team1.privatexprize.org and IP address associated of 192.168.1.1 has a TTL of two hours. If the container goes down and comes back up again with IP address 192.168.1.2, an infrastructure component such as our proxy could attempt to connect to team1.privatexprize.org at 192.168.1.1 for up to two hours (until the TTL expires). Container state exists not only in the container but in infrastructure in things like DNS. We reduced the impact of this problem by reducing the DNS TTL in our container private DNS.

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 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.

4.2. Future work

We had put our proxies into VMs to expedite getting the Data Collaborative running, but having to log into them to configure them and to look at logs make VMs unwieldy to work with. The proxy configuration and logging need to be automated and the proxies put into a container for easier deployment into contests and for maintenance. In the longer run, we are looking to see if the functionality of the proxies can be absorbed into cloud infrastructure. We have observed that even in the short time span of this project, native cloud infrastructure capabilities have started catching up to proxy functionality. In Section 3, we talked about the non-requirements of performance, low cost, and avoiding service provider lock-in. In the long run, those areas will need to be revisited.

4.3. Influences of the COVID-19 Pandemic

As mentioned above, the COVID-19 Pandemic pushed XPRIZE to focus on new challenges and to accelerate development. Like many other IT development projects, developing the Data Collaborative had to be done remotely. There were face to face meetings and work sessions during late 2019, but the bulk of development was done remotely at home by the team in locations from ranging from Northern California, Southern California, and Arizona. We would occasionally encounter problems in working with our Cloud Service Provider, but it was impossible to know whether these were from Cloud Service provider capacity problems or just network congestion from stay at home orders [47].

4.5. Implications for Cloud Native Application Development

Much of the cloud design pattern literature focuses on certain areas of Cloud technology, such as microservices/containers [15,17,18], serverless computing [19-21], or Software Defined Infrastructure [2]. Our experiences show that a Cloud Native Application can use any or all of these technologies, and even older Cloud technologies like Virtual machines. Design Patterns need to be more inclusive and prescriptive about which ones to use, as multiple technologies can apply in a single application implementation.

It is unfortunate that for our project, dealing with service provider lock-in [24-26] was not a priority. Tools for reducing vendor lock-in needs to be simple to use and standardized enough to make it easy to make this a priority as an effort that would have a good return on investment.

5. Conclusion

We have described the requirements, design, and implementation of the XPRIZE Data Collaborative, a cloud-based infrastructure that enables the XPRIZE Foundation to combat the COVID-19 pandemic and host future data-centric competitions. Our case study differs from others by including how we used not just one Cloud Technology like containers or serverless but others as well, even older technologies like VMs. Our implementation has proven successful to this point, but we did encounter issues, some simple and others more subtle, which we have shared so others could avoid them.

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. Hossain, M. and Kauranen, I., Competition-based innovation: the case of the X prize foundation. *Journal of Organization Design*. **2014**, *3*, pp.46-52.
3. Haller, J.B., Bullinger, A.C. and Möslein, K.M., 2011. Innovation contests. *Business & Information Systems Engineering*, **2011,** 3, pp.103-106.
4. Accelerating Radical Solutions to COVID-19 and Future Pandemics. Available online. https://www.xprize.org/fight-covid19 (accessed on 28 October 2020).
5. MacKay, M.J., Hooker, A.C., Afshinnekoo, E., Salit, M., Kelly, J., Feldstein, J.V., Haft, N., Schenkel, D., Nambi, S., Cai, Y. and Zhang, F. The COVID-19 XPRIZE and the need for scalable, fast, and widespread testing. *Nature biotechnology*, 2020, *38*, pp. 1021-1024.
6. Kaggle. Available Online. https://kaggle.com/ (accessed on 5 November 2020).
7. Yang, X., Zeng, Z., Teo, S.G., Wang, L., Chandrasekhar, V. and Hoi, S., July. Deep learning for practical image recognition: Case study on kaggle competitions. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, United Kingdom, August 2008; ACM: New York, NY, United States, 2018; pp. 923-931.
8. CoLab. Available Online. https://colab.research.google.com/notebooks/intro.ipynb (accessed on 5 November 2020).
9. Bisong, E., Google Colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform* (pp. 59-64). Apress, Berkeley, CA.
10. CoCalc. Available Online. https://cocalc.com/ (accessed on 5 November 2020).
11. Bouvin, N.O. From notecards to notebooks: there and back again. Proceedings of the 30th ACM Conference on Hypertext and Social Media, Hof, Germany, 2019; ACM: New York, NY, United States, 2019; pp. 19-28.
12. Nextjournal. Available Online. https://nextjournal.com/ (accessed on November 5, 2020).
13. 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.
14. Kratzke, N., 2018. A brief history of cloud application architectures. *Applied Sciences*, ***2018***, *8*, p.1368.
15. 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.
16. Gannon D, Barga R, Sundaresan N. Cloud-native applications. *IEEE Cloud Computing*. **2017**, 4, 16-21.
17. Balalaie, A., Heydarnoori, A. and Jamshidi, P. Microservices architecture enables devops: Migration to a cloud-native architecture. *IEEE Software*, **2016**, 33, pp. 42-52.
18. Burns, B. and Oppenheimer, D., Design patterns for container-based distributed systems. Proceedings of the 8th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 16).
19. Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A. and Suter, P. Serverless computing: Current trends and open problems. *Research Advances in Cloud Computing* **2017**, (pp. 1-20). Springer, Singapore.
20. Eismann, S., Scheuner, J., van Eyk, E., Schwinger, M., Grohmann, J., Herbst, N., Abad, C.L. and Iosup, A., 2020. Serverless Applications: Why, When, and How? *IEEE Software*, **2020**, 38, pp.32-39.

21. Hong, S., Srivastava, A., Shambrook, W. and Dumitraș, T. Go serverless: securing cloud via serverless design patterns. Proceedings of the 10th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 18).
22. Kang, J.M., Lin, T., Bannazadeh, H. and Leon-Garcia, A. Software-defined infrastructure and the SAVI testbed. International Conference on Testbeds and Research Infrastructures, Guangzhou, China 5-7 May 2014; Springer, 3-13.
23. Homer, A., Sharp, J., Brader, L., Narumoto, M. and Swanson, T. *Cloud design patterns: Prescriptive architecture guidance for cloud applications*. Microsoft patterns &Practices: 2014.
24. Kratzke, N., Quint, P.C., Palme, D. and Reimers, D., 2017. Project Cloud TRANSIT—Or to Simplify Cloud-native Application Provisioning for SMEs by Integrating Already Available Container Technologies. Proceedings of the European Project Space on Smart Systems, Big Data, Future Internet—Towards Serving the Grand Societal Challenges, Rome, Italy, 2016; V.Kantere and B. Koch eds.; SciTePress: Setubal, Portugal; pp. 2-26.
25. Bergmayr, A., Breitenbücher, U., Ferry, N., Rossini, A., Solberg, A., Wimmer, M., Kappel, G. and Leymann, F. A systematic review of cloud modeling languages. *ACM Computing Surveys (CSUR)* **2018**, *51*, pp.1-38.
26. Opara-Martins, J., Sahandi, R. and Tian, F. Critical review of vendor lock-in and its impact on adoption of cloud computing. Proceedings of the International Conference on Information Society (i-Society 2014), London, United Kingdom, November 2014; Infonomics Society: London, United Kingdom; pp. 94-99.
27. 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)
28. Estimate of Public Jupyter Notebooks on GitHub. Available Online. https://nbviewer.jupyter.org/github/parente/nbestimate/blob/master/estimate.ipynb (accessed on 10 January 2021).
29. Jupyter Notebook. Available Online. https://jupyter.org/ (accessed on 6 November 2020).
30. Apache Zeppelin. Available Online. https://zeppelin.apache.org/ (accessed on 7 November 2020).
31. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I. Spark: Cluster computing with working sets. *HotCloud* **2010**, 10, 95-101.
32. Jupyterhub. Available Online: https://jupyter.org/hub (accessed on 8 November 2020).
33. Mockpetris, P. *RFC 1035: Domain Names – Implementation and Specification*. RFC Editor: November 1987.
34. Rekhter, Y., Moskowitz, B., Karrenberg, D., Groot, G.D. and Lear, E., 1996. *RFC 1918: Address allocation for private internets*. RFC Editor: February 1996.
35. Homer, A., Sharp, J., Brader, L., Narumoto, M. and Swanson, T. *Cloud design patterns: Prescriptive architecture guidance for cloud applications*. Microsoft patterns &Practices: 2014. pp 72-74.
36. Security in the Jupyter notebook server. Available Online: https://jupyter-notebook.readthedocs.io/en/stable/security.html (accessed 10 January 2021).
37. Sun, S.T. and Beznosov, K. The devil is in the (implementation) details: an empirical analysis of OAuth SSO systems. Proceedings of the 2012 ACM Conference on Computer and Communications security, Raleigh, NC, United Sates, October 2012; ACM: New York, NY, United States, 2012; pp. 378-390.
38. Homer, A., Sharp, J., Brader, L., Narumoto, M. and Swanson, T. *Cloud design patterns: Prescriptive architecture guidance for cloud applications*. Microsoft patterns &Practices: 2014. pp 67-71.
39. 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).
40. Aas, J., Barnes, R., Case, B., Durumeric, Z., Eckersley, P., Flores-López, A., Halderman, J.A., Hoffman-Andrews, J., Kasten, J., Rescorla, E. and Schoen, S.. Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, United Kingdom, November 2019; ACM: New York, NY, United States, 2019; pp. 2473-2487.
41. Sidecar pattern. Available Online: https://docs.microsoft.com/en-us/azure/architecture/patterns/sidecar (accessed on 9 November 2020).
42. Gateway Offloading pattern. Available Online: https://docs.microsoft.com/en-us/azure/architecture/patterns/gateway-offloading (accessed on 9 November 2020).
43. 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).

44. Bulkhead pattern. Available Online: https://docs.microsoft.com/en-us/azure/architecture/patterns/bulkhead (accessed on 9 November 2020).
45. AI and Data for the Benefit of Humanity. Available Online: https://xprize.org/aialliance (Accessed on November 9, 2020).
46. Pandemic response Challenge. Available Online: https://www.xprize.org/challenge/pandemicresponse (Accessed on January 10, 2021).
47. Liu, S., Schmitt, P., Bronzino, F. and Feamster, N. Characterizing Service Provider Response to the COVID-19 Pandemic in the United States. arXiv preprint arXiv:2011.00419. 2020