Software Engineering for Real-Time NoSQL Systems-centric Big Data Analytics

by

William F. Clark

December, 2020

Director of Thesis: Dr. Venkat Gudivada

Major Department: Computer Science

Recent advances in Big Data Analytics (BDA) have stimulated widespread interest to integrate BDA capabilities into all aspects of a business. Before these advances, companies have spent time optimizing the software development process and best practices associated with application development. These processes include project management structures and how to deliver new features of an application to its customers efficiently. While these processes are significant for application development, they cannot be utilized effectively for the software development of Big Data Analytics. Instead, some practices and technologies enable automation and monitoring across the full lifecycle of productivity from design to deployment and operations of Analytics. This paper builds on those practices and technologies and introduces a highly scalable framework for Big Data Analytics development operations. This framework builds on top of the best-known processes associated with DevOps. These best practices are then shown using a NoSQL cloud-based platform that consumes and processes structured and unstructured real-time data. As a result, the framework produces scalable, timely, and accurate analytics in real-time, which can be easily adjusted or enhanced to meet the needs of a business and its customers.

Software Engineering for Real-Time NoSQL Systems-centric Big Data Analytics

A Thesis

Presented to the Faculty of the Department of Computer Science

East Carolina University

In Partial Fulfillment of the Requirements for the Degree

Master of Science in Computer Science

by

William F. Clark

December, 2020

Software Engineering for Real-Time NoSQL Systems-centric Big Data Analytics

by

William F. Clark

APPROVED BY:

DIRECTOR OF THESIS: ______________________________
Venkat Gudivada, PhD

COMMITTEE MEMBER: ______________________________
Mark Hills, PhD

COMMITTEE MEMBER: ______________________________
Nic Herndon, PhD

CHAIR OF THE DEPARTMENT
OF COMPUTER SCIENCE: ______________________________
Venkat Gudivada, PhD

DEAN OF
THE GRADUATE SCHOOL: ______________________________
Paul J. Gemperline, PhD

ACKNOWLEDGEMENTS

Nobody has been more important to me in the pursuit of my education than the members of my family. Your love and guidance will always be with me in whatever I pursue.

I would also like to thank ECU’s Department of Computer Science for providing me a world-class education. While simultaneously allowing me the flexibility to pursue a thriving and successful career.

Contents

List of Figures

Chapter 1: Introduction

Software development involves various steps and stages that need to be done effectively and accurately to produce a working, correct, and efficient software product. Researchers and practitioners in software engineering all agree that adopting the right software development process and using it effectively could make the difference between the success and failure of the development efforts. Several software process models like the waterfall, V, rapid prototyping, extreme programming, lean and agile models have emerged and have been used for various software development projects [1]. These processes offer guidelines that help improve and control the development activities. For significant data analytics development, some of the current software process models fit the mold better than others.

To choose the best model, you have to understand the current state of Big Data Analytics. Big Data Analytics overwhelms the existing software and hardware resources available due to its volume, velocity, and variety. Recently Big Data infrastructures and platforms have been developed that offer ways to handle and use big data. These different tools, if leveraged correctly, can facilitate scalability and provide environments that support the rapid growth of Big Data Analytics software. These types of tools are included in different services that are embedded within a cloud computing platform. These tools, combined with a middleware framework, significantly increases the reusability and transparency of big data analytic development. [2] This creates opportunities to share data, software, and infrastructures that facilitate better utilization of big data within an organization.

From the software engineering perspective, it is necessary to develop a software development process that can incorporate both the development infrastructure and the actual development of the software. The infrastructure and development must be considered while simultaneously dealing with all of the issues and challenges of big data. This requires a process that allows for progressing through the various stages of the development process from communication, modeling, design, construction, and deployment [1] in a significant way that is also suitable for the specific demands of BDA.

As several software development models are available, we investigate some of these models and then try to match the domain of BDA to one or more software processes that can be adopted for them. After a suitable model is chosen, a background on BDA enabling technologies like cloud computing and containerization is done. The different BDA characteristics, constraints, and requirements and defined. Once the process and infrastructure are described, an analysis of different approaches to BDA frameworks is shown. Finally, with the provided background defined, a novel BDA framework is introduced and implemented to show how the development processes and infrastructure components can be integrated. The implementation of the framework is done on a real-world problem of stock prediction. This problem was chosen because the frequency of the data is real-time and the variety of datasets available.

Research Contribution:

Introducing a suitable software development process that accommodates the unique needs of BDA is essential. Such a process should offer a systematic and well-defined model to follow throughout the development process. Thus, we attempt to understand the general themes associated with BDA development activities and propose a possible framework that will solve the current challenges with developing Reusable Big Data Analytics at Scale.

Chapter 2: State of Software Development in the Modern-Day

Software engineering practices offer the guidelines needed to analyze, design, and implement successful software projects. BDA development projects are no exception. They need to be organized and managed through a suitable software development model. Many software development models exist, including the following:

The Waterfall Model (aka. Sequential Development Model)

Each step is frozen before the next step. That is, the requirements, for example, must be completed and frozen before the design starts. As its linear model is easy to implement and control, and the number of resources required is minimal. Are simplicity and clear milestones have ensured that it remains one of the most popular models used [3]. However, its rigid format limits the flexibility and ability to incorporate change at different stages and also makes it hard to catch problems early in the project.

The Spiral Model

An iterative approach is used in the Spiral model but with a much greater emphasis throughout the process placed on risk analysis. It offers a fair amount of flexibility as it allows the team to adapt to accompanying risks and uncertainties, such as a rapid project schedule and changing team composition. This model is seen as being useful when dealing with rapidly moving technology with the need for rapid completion for the project to be successful [5].

The Unified Process (UP)

It is an iterative process in which the different phases of the project are divided into a series of time-boxed iterations. Each iteration results in an increment, a release of the system that contains added or improved functionality compared with the previous release. This process focuses on addressing the most critical risks early in the project life cycle. It supports multiple architectural models and views, making it suitable for use for a wide variety of projects [5].

The Aspect-Oriented Software Development (AOSD) Model

Improves the way software is developed by providing better ways for modularization and separating cross-cutting concerns. AOSD allows multiple concerns to be expressed separately and automatically unified into working systems. This can be helpful in the long-run as it supports software reuse. Separating the concerns results in problems in testing the different modules as knowledge about the implementation of many modules in the system will be needed. This reduces this model's overall flexibility and usefulness and requires very experienced teams [6].

Rapid Application Development (RAD)

They are emerged as a quick method to build small to medium-sized software applications using iterative prototyping steps. The model relies on joint application design (JAD), where the client is heavily involved in the whole development process and active in reviewing prototypes and generating accurate requirements. It is also characterized to be mainly suitable for Highly interactive, low complexity projects [7].

The Agile Model

Utilizes an adaptive team that can respond to the changing requirements of the customer and increases their satisfaction by rapid delivery of useful software. The most beneficial aspect is its incredibly high flexibility, which allows it to respond to the changing requirements effectively and quickly. However, when working on a large project, it becomes difficult to judge the efforts and the time required to complete the project. As such, it should only be utilized for smaller-scale projects or well-defined larger projects [8].

As we discuss the different software development models, it becomes clear that we need to identify and define the nature of BDA to be able to choose the right model for their development. The current attitude in big data analytics is that the requirements are common and obvious, and application designers will assess and produce the required software based on common knowledge. However, the complexity and specific nature of these applications introduce many challenges leading to a greater need to spend more time and effort identifying the exact requirements and managing the development project effectively.

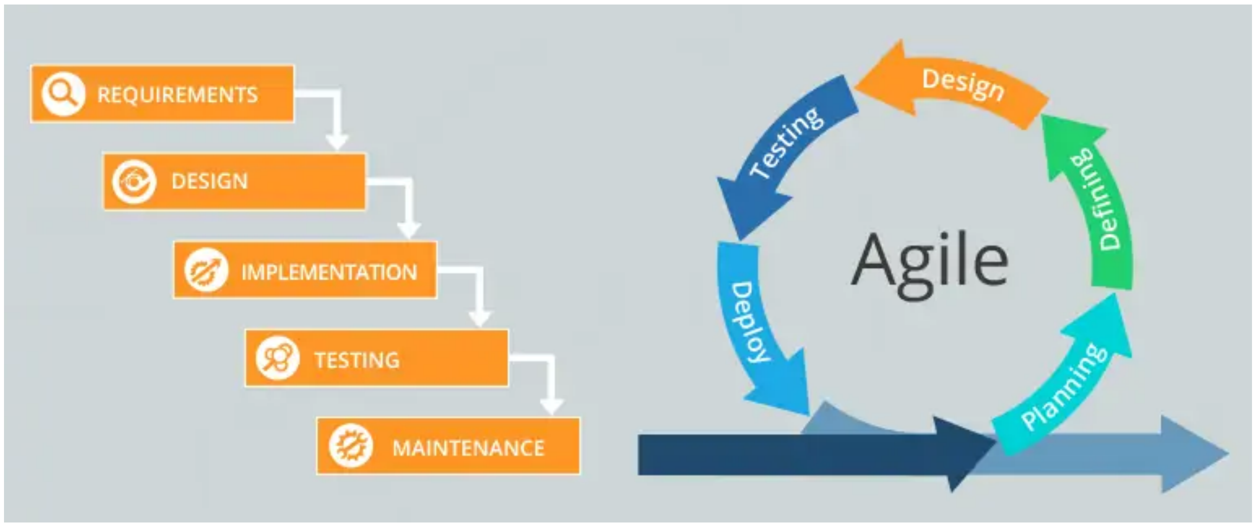


Figure 1: A comparison between the Waterfall and Agile methods [4]

2.1 Big Data Analytics Software Development

Examining the more recent software development models such as the Agile and AOSD, we recognize a more substantial potential to match them with these applications. AOSD relies on modularizing the application and supporting reuse. As we have seen so far, most applications in this domain share several standard requirements and features. Many of these are also cross-cutting through the different functionalities of the applications. As a result, these can be considered as generic concerns that can be modularized and reused for different applications.

However, this will introduce the added difficulty of accurately separating these concerns and providing the right information to ensure proper testing and integration. The unified process (UP) model also offers some potential as it offers an incremental approach to the application development and a common standardized language (such as UML: Unified Modeling Language) to define and work with the applications' components. The user and development team can create a plan identifying and prioritizing the requirements such that the development team can focus on one subset of these requirements at a time. However, the added risk analysis and quality assurance measures in this model impose additional efforts and costs on the project.

Agile processes rely on an initial review of the requirements that generate a prioritized backlog. This backlog is used to start work on one or more of its components to build a single deliverable feature of the software quickly. Although the backlog of requirements is prioritized early in the process, it is possible to revisit it after every sprint and reorganize it based on earlier sprints. Initial requirements gathering is a critical component of Big Data Analytics, as is iteration. The Agile model addresses both the original hardening of requirements and the flexibility needed to iterate on any Big Data Analytics process. The Agile process fits well with the nature of BDA, where most BDA are modular by nature, and specific requirements can be addressed, designed, implemented, and released incrementally. [8]

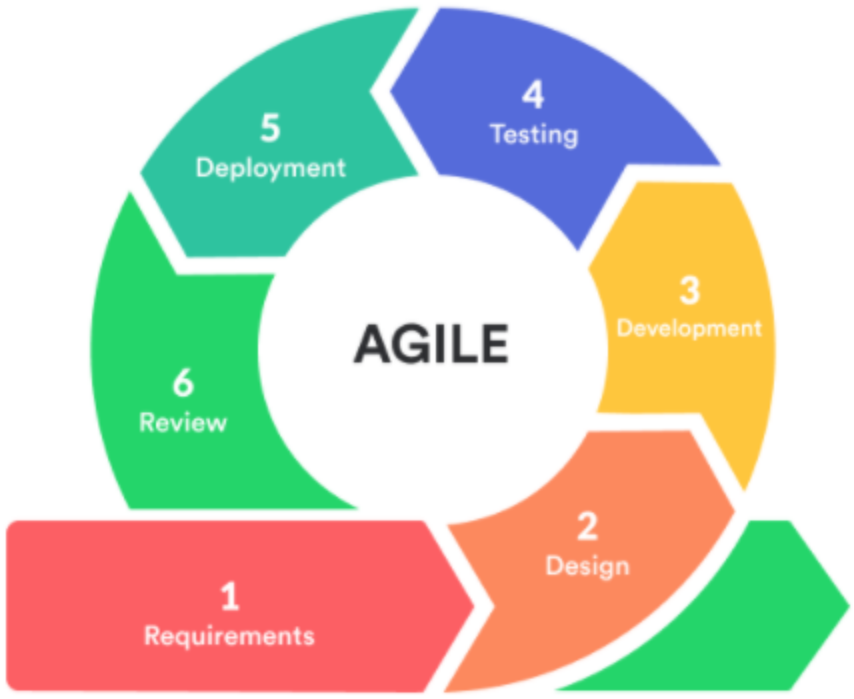


Figure 2: The continuous Agile lifecycle [4]

One of the essential features of the Agile model is the freedom the development team has in developing each component in a sprint and the intense focus on the rapid design and implementation of these components. In Big Data Analytic development, several features can be designed to use various standard components. These components could be methods for organizing the data, specific mathematical and statistical models to apply on the data, standard data transfer, and validation techniques. Such components can be selected early in the project for development and can be used later to build other components.

One of the main challenges that many see as the most important is how to engineer big data with agility [8][9] and in time to be useful.   To further enable, teams can adopt continuous integration and delivery (CI/CD) practices. These practices allow for the automation and constant transparency of code across a team, the validation of all aspects of a Big Data Project (data, model, code), and, finally, the delivery of the asset to stakeholders. This automation allows the BDA development team to be more focused on the requirements and also use the iterative nature of the Agile process to solicit and refine requirements throughout the project's lifecycle. To implement this automation, some current cloud-based services and tools provide the needed infrastructure to do this rapidly and at scale.

In [9], a study conducted regarding big data analytics shows that one of the main reasons for not using BDA is the difficulty of architecting these applications. Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing has rapidly become an emerging computing paradigm enabling various services to be provided to interested consumers at lower costs and skill entry points [10]. Designing effective cloud services is possible when the main requirements for cloud-based applications are identified and addressed.

Chapter 3: Architecting BDA in the Cloud

At the beginning of a BDA development, technical architecture needs to be established. Creating a flexible, scalable architecture is essential to supporting Agile Big Data Analytics. Envisioning the architecture is the first step in an Agile BDA framework. Architecture for fast data engineering and data science primarily focuses on technical architecture. The technical architecture trends are typically around the service models (IaaS, PaaS, and SaaS) available on the cloud. Especially since newer service models emerge, such as the Data-as-a-Service (DaaS) model, which supports explicitly data-intensive applications, better support for Big Data applications is being offered on the cloud [11]. Discovery is made to define the general background of these services and what they mean for Big Data Analytics in the cloud.

3.1 Cloud Platform Integration and Execution

In Agile Big Data, it makes sense to utilize the easiest-to-use, most approachable systems. Cloud computing and other platforms as a service help minimize infrastructure costs and maximize productivity. The simplicity of the platform stack helps a development team enable a return to agility. We'll use this stack to compose scalable systems in as few steps as possible. The stack lets us move fast and consume all available data without running into scalability problems that cause us to discard data or remake our application in flight.

Cloud platform integration aims to mask the heterogeneity among different cloud platforms offered by various cloud providers to provide a uniform way to access clouds of various vendors and provision, consume and manage resources from different clouds. There are two following generic approaches to resolve the incompatibilities between different cloud platforms. [12]

Cloud Broker-Based Approach

Many of the cloud platforms that currently exist can either migrate work from one to another or be able to call services from one to another. This is deemed the Cloud broker-based approach, which provides the ability to run workflow applications in intercloud environments. It acts as a mediator among users of workflow and providers of cloud systems, helping in the selection of the target cloud, accessing that cloud, and achieving user-defined requirements [13].

Container-Based Approach

This approach exploits containerization technology and orchestration like Docker and Kubernetes. Containerization creates compute resources that are singular and separate from the environment that they are run on [14]. Meaning they provide the ability to quickly and efficiently build and deploy workflows across cloud computing systems by encapsulating compute resources and delivering a user-defined execution environment [15]. A container packs only the libraries and packages needed by sub-workflow or workflow activity [15]. By doing that, the workflow portability and migration are improved, allowing seamless and agentless migration of workflows across diverse cloud infrastructures.

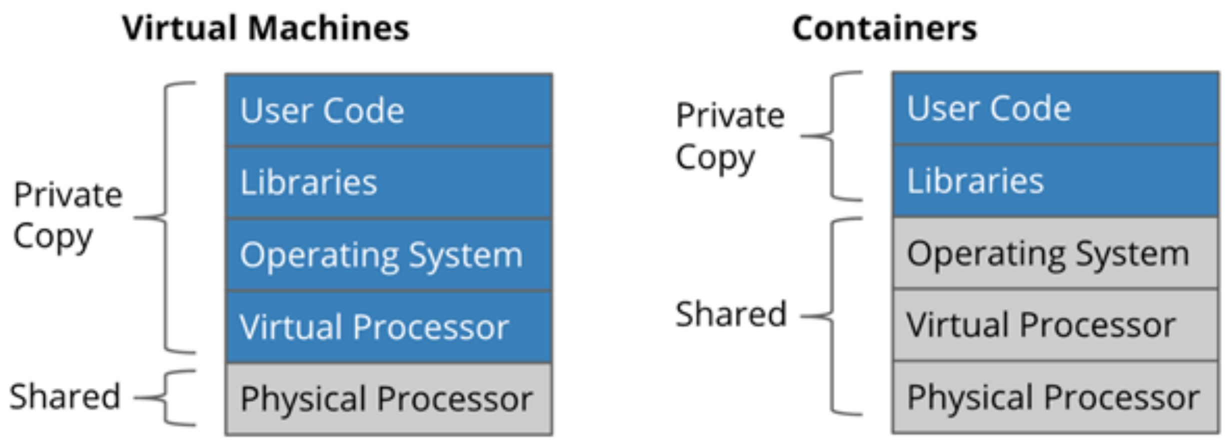


Figure 3: Differences between Virtual Machines versus Containers [16]

3.2 Resource Provisioning

Resource provisioning aims to select and provision the cloud resources that will be used to execute Big Data Analytic tasks such as consumption pipelines. There are two following approaches to resource provisioning.

Static Resource Provisioning Approach

This approach decides to provision virtual resources that are required to run workflow fragments or tasks before the execution of the workflow. It is not able to dynamically scale resources in or out [17]. The provisioned resources are fixed, and they are the only resources available during the whole period of workflow execution. Thus, such an approach is suitable to be used in cases where the demand for the workflow is predicted and fixed in terms of resources.

Dynamic Resource Provisioning Approach

In contrast, this approach takes the decision of provisioning resources during the execution of workflow or at runtime. It decides which resource types and configurations are most suitable, and when to add or remove resources according to the demands of the workflow. In other words, this approach is taking all decisions or refining initial ones at runtime and determining which virtual resources need to keep running and active, which resources should be provisioned and which resources from the provisioned resources should be deprovisioned as the workflow execution progresses. This approach aims to avoid provisioning because of its implication on performance and over-provisioning because of its implication on cost and system utilization.

As described, a BDA project's architecture is incremental and iterative. BDA project requirements and associated data can change week over week. Thus, the stack must also be flexible enough to account for this. Thus, the framework from consumption to data visualization utilizes a containerized dynamic provisioning workflow. That way, execution, and environment can be decoupled if the requirements of the project change during development. For example, if a new cloud provider is used or a data pipeline's volume increases or wholly decommissioned. Now that end to end execution in the framework has been defined, an understanding of the data processing must be done. As with the way execution is handled in BDA workflows, a modern solution exists for Data Storage and Data Movement

3.3 Big Data Storage and Movement

Big Data consists of extensive datasets primarily in the characteristics of volume, variety, velocity. The variability in data attributes requires a scalable architecture for efficient storage, manipulation, and analysis [18]. The combination overwhelms the current architectures by imposing very high demands on storage, communication, and processing capabilities. For example, relational databases have successfully managed to organize and use large volumes of data that have uniform and simple data types. However, the rapid increase in the amount of unstructured data and complex data structures makes it almost impossible for these traditional databases to work virtually [19]. To overcome these challenges, we will go into the types of storage used and data movement techniques needed to create agile analytics.

Data Storage in the Cloud

Big data workflow comprises of a set of data-intensive tasks, which communicate with large datasets. These large datasets should be stored in the cloud since the execution of big data workflows will be carried out in the cloud and be passed among workflow tasks using data movement techniques or shared storage systems [20]

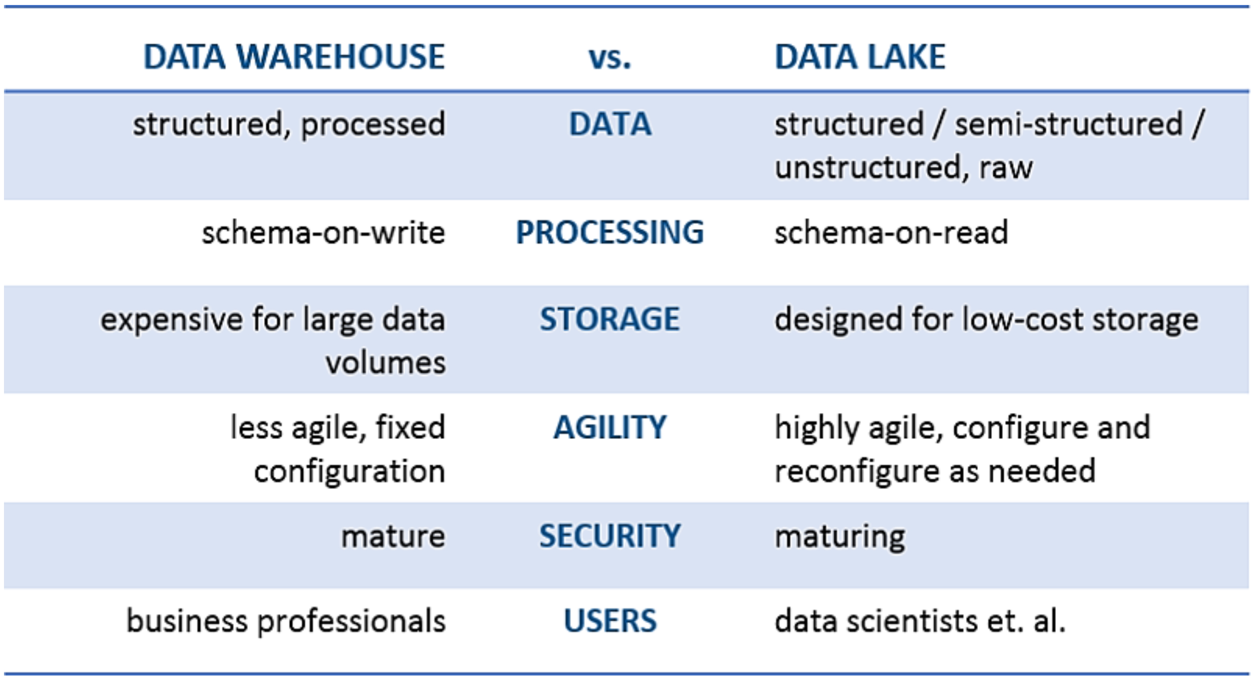
Cloud Storage and Database as a Service

This is a storage service offered by cloud providers. This approach requires the workflow management system to manage data on the cloud storage service [20]. Many cloud providers offer different kinds of the database as service models that can be pointed to with the right permissions. This storage service is typically a data warehouse and can be any type of database system that the development team wants. Given the nature of modern-day data, the data warehouse should be a NoSQL system so it can handle the different variety of data that is coming in. However, the reliability of data stored in the cloud storage system could be an issue with this approach [21].

Shared Cloud-backed File System

It intends to deploy shared file systems in the cloud [22], where the backend can be a single cloud or cloud-of-clouds. This is the type of storage system used in Data Lakes. Typically, the data is directly accessed via an endpoint and pushed into a data warehouse after it has been curated. The direct access model means that the single point of failure is no longer an issue. That said, the governance around these types of shared file systems can be difficult if not organized and cataloged correctly. The file system following this model can be either:

- Custom Cloud File System: The aim here is to build a custom shared file system for workflow tasks without the interposition of a proxy.
- Pre-developed Cloud File System: The aim here is to select and use an existing shared file system.



DATA WAREHOUSE	vs.	DATA LAKE
structured, processed	DATA	structured / semi-structured / unstructured, raw
schema-on-write	PROCESSING	schema-on-read
expensive for large data volumes	STORAGE	designed for low-cost storage
less agile, fixed configuration	AGILITY	highly agile, configure and reconfigure as needed
mature	SECURITY	maturing
business professionals	USERS	data scientists et. al.

Figure 4: Differences between data lake versus data warehouses [23]

Data storage for big data analytics has to be a combination of the above. Where the incoming data can be raw and unstructured. Typically, raw unstructured data is not what should be presented back to the end-user, whether it be a consumer of the data or an internal data scientist. Thus, a layer on top of the unstructured data can be provided that consumes the raw unstructured data and lands it into a data warehouse.

Data Movement

By moving the execution of big data analytics to the cloud, to stay agile, the working datasets should also be moved to the cloud. These datasets are large, and moving or transferring them is an expensive task. In the literature, several research works have been proposed various approaches to tackle the problem of data movement for data-intensive workflows. This intends to transfer data with minimal data transfer time. The following are three different techniques to achieve the lowest data transfer time [21]:

- Data Parallelism: The ability of a service to process data chunks in parallel with minimal performance loss. Such ability includes the processing of independent data on various compute resources.
- Data Streaming: This technique intends to enable data transport among workflow fragments/tasks through the use of data streams. That allows support for high-throughput and low latency.
- Data Batching: This technique intends to determine and control the arrival time and the rate of data transfer as opposed to the movement of data from one place to another as quickly as possible. As an alternative to transferring data to a task, this technique can be used to

delay data transfer or transfer data using lower capacity links in order to allocate resources to serve other crucial tasks

Big Data characteristics are voluminous, context-specific, heterogeneous, differ in frequency, and sometimes too complex to describe. Big Data Analytics workflows are also highly non-linear and contain several feedback loops. Setting up a framework that can quickly solve the software stack that is needed to implement big data analytics is key. This can be down by making sure that the execution of our end to end workflows are decoupled to its environment, and the cost of running the execution is based on the type of workflow that is running. That, combined with modern-day data architecture, allows for a team of Big Data Analytics developers to quickly go from idea to product. While at the same time being agile enough to make changes sprint over sprint if the insights created seem to be inaccurate or not what the team was looking for. Now that the specifics around the data pipeline execution infrastructure and storage for the framework have been addressed. The resulting analytics piece of the framework can be defined.

Chapter 4: Defining the Big Data Analytics Framework

4.1 Analytics Life Cycle

Big Data Analytics at scale, combined with an Agile software development framework, is particularly influential in the rapid creation and testing of hypotheses. One commonly used Big Data Analytics design has been depicted in various industry and research [24], [25]. It has harmonies with prior workflows defined in the context of data science and data mining, such as TDSP [26] and CRISP-DM [27][29]. Despite the minor differences, these representations have in common the data-centered essence of the process and the multiple feedback loops among the different stages [28]. Each of these processes fit within an agile lifecycle.

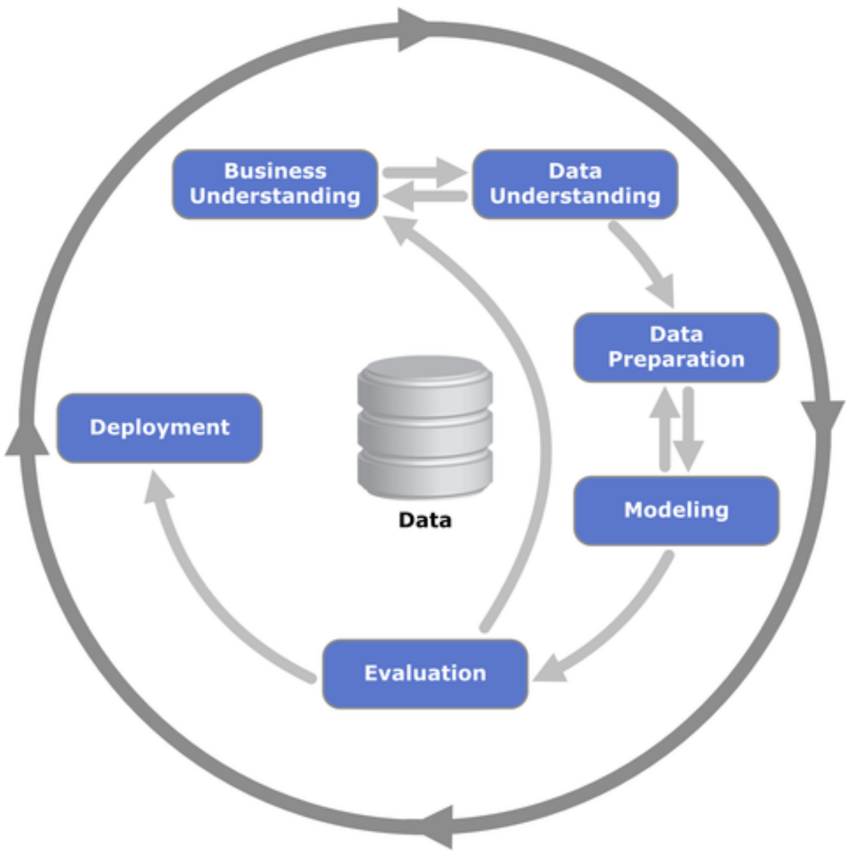


Figure 5: The CRISP-DM Lifecycle [30]

The above CRISP-DM model in Figure 5 is the basis for the analytics portion of the BDA Framework. These stages are broken down into the following to underline the specifics development processes associated with the Big Data Analytics framework.

Data Collection and Cleaning

During data collection, teams look for and integrate available datasets or collect their own. Often, they might train a partial model using available generic datasets and then use transfer learning together with more specialized data to train a more specific model. Data cleaning involves removing inaccurate or noisy records from the dataset, everyday activity to all forms of data science. Also, when a validation error is detected, data cleansing must be done to fix the error.

Data Labeling

Data labeling assigns ground truth labels to each record. For example, an engineer might have a set of images on hand which has not yet been labeled with the objects present in the image. Most of the supervised learning techniques require labels to be able to induce a model. Other techniques like reinforcement learning, use demonstration data, or environment rewards to adjust their policies. Labels can be provided either by engineers themselves, domain experts, or crowd workers in online crowdsourcing platforms.

Feature Engineering

Feature engineering refers to all activities that are performed to extract and select informative features for Big Data Analytics models. For some models, this stage is less explicit and often blended with the next stage, model training. Another common form of feature engineering is to join in a new data source to augment the existing features with new signals. This is especially true for convolutional neural networks.

Model Requirements

In the model requirements stage, designers decide which features are feasible to implement with Big Data Analytics and useful for a given existing product or a new one. Most importantly, in this stage, they also decide what types of models are most appropriate for the given problem.

Model Training

During model training, the chosen models are trained and tuned on the clean, collected data and their respective labels. Typically, models are trained and then retrained if the quality of the model is subpar. For some models, this step is integrated with feature engineering, which refers to the augmentation of the training and serving data with new features to improve the quality of the generated model.

Model Evaluation

This is when engineers evaluate the output model on tested or safeguard datasets using pre-defined metrics. For critical domains, this stage might also involve extensive human evaluation. The inference code of the model is then deployed on the targeted device and continuously monitored for possible errors during real-world execution. The information gained from the evaluation helps a team decide whether to invest resources in implementing the changes in production.

Indeed, the day-to-day work of an engineer doing Big Data Analytics involves frequent iterations over the selected model, hyper-parameters, and dataset refinement. Similar experimental properties have been observed in the past in scientific software [30] and hardware/software co-design [31]. After the results have an analytics algorithm that has been created, a test on each component associated with that result must be done. This is because not only is testing good practice, but it is also a part of the agile framework.

4.2 Validation and Testing

All areas within high-tech domains, such as finance, healthcare, and manufacturing, can leverage Agile to create rapid real-time validation. These validation activities for components that learn from data not only focus on programming bugs but also focus on inherent issues that arise from model errors and uncertainty. Understanding when and how models fail to make accurate predictions is an active research area [32], which is attracting more attention as BDA algorithms and optimization techniques become more complex. There are different types of testing that can be introduced in the BDA workflow.

Validating Pipeline Code

The first step in validating the pipeline is making sure that the code that is being used to generate the components is satisfactory in both performance and clarity. There are many different utilities that can be used to perform this type of analysis.

Validating Data

We can add tests to validate input data against the expected schema or to validate our assumptions about its valid values they fall within expected ranges or are not null. For engineered features, we can write unit tests to check they are calculated correctly. For example, numeric features are scaled or normalized, one-hot encoded vectors contain all zeroes and a single 1, or missing values are replaced appropriately. For instance, you might have an inherent bias in the training data where there are many more data points for a given value of a feature compared to the actual distribution in the real world, so it's important to check performance across different slices of the data.

Validating the Model Quality

While BDA model performance is non-deterministic, Data Scientists usually collect and monitor several metrics to evaluate a model's performance, such as error rates, accuracy, receiver operating characteristic (ROC), confusion matrix, precision, and recall. They are also useful during parameter and hyper-parameter optimization. As a simple quality gate, we can use these metrics to introduce Threshold and Regression tests in our pipeline, to ensure that new models don't degrade against a known performance baseline.

While some aspects are inherently non-deterministic and hard to automate, there are many types of automated tests that can add value and improve the overall quality of your BDA system. If the hypothesis is wrong, the developers can quickly test and see why and any discrepancies are occurring. This allows developers to quickly go back to the drawing board. It allows developers to test different models and data scientists to have more accurate information. The framework that has been described so far will need to be wrapped in an automation process for this agile, rapid testing and validation of the Big Data Analytics hypothesis to happen.

4.3 Modularization and Automation

To successfully adhere to Agile BDA DevOps, a custom pipeline is created to operationalize the big data pipeline workflow. The customized pipeline is created by partitioning the pipeline into separate tasks needed to produce BDA. These portioned tasks are also known as

Microservices. Building the pipeline in terms of these microservices allows for Agile practices to be implemented because it modularizes the pipeline into a set of tasks that can be accomplished iteratively. Due to the modularization of these tasks, the pipeline can now be operationalized and managed.

While feedback loops are typical in Agile software processes, the peculiarity of the Big Data Analytics workflow is related to the amount of experimentation needed to converge to a good model for the problem. Fast-paced model iterations require frequent deployment. To ensure that system deployment goes smoothly, several engineers recommend not only to automate the training and deployment pipeline but also to integrate the model building with the rest of the software, use standard versioning repositories for both BDA and non-BDA codebases, and tightly couple the BDA and non-BDA development sprints and standups. Just as software engineering is primarily about the code that forms shipping software, BDA is all about the data that powers analytic models. So, to effectively build out our microservices pipeline, we must utilize container technology as well as continuous integration and continuous deployment processes.

Containerization

Containers offer a logical packaging mechanism in which applications can be abstracted from the environment in which they run. This decoupling allows container-based applications to be deployed quickly and consistently, regardless of whether the target environment is a private data center, the public cloud, or even a developer's laptop. Containerization provides a clean separation of concerns, as developers focus on their application logic and dependencies, while IT operations teams can focus on deployment and management without bothering with application details such as specific software versions and configurations specific to the app

As can be seen from the above statement, containers can help accelerate the agile software development process. They do so by enabling the creation of application components using self-contained, efficient docker images. These images package their dependencies and help increase the velocity of development through a full CI/CD process. Containers also provide a reduction in both size as well as processing overhead when compared with VM based monolithic applications. These images also support incremental development as well as “roll forward or rollback based on versions. DevOps is a relatively new concept for most data scientists who are always developed and deployed in the same monolithic environment.

Continuous Integration

That is where automated tests come into the picture, so whenever any developer pushes his/her code to the repository, automated integration tests run on the server to check and validate seamless integration into the existing project. Hence, after every push, you will have a stable project in the repository. This is Continuous Integration (CI). The developers can write code following some standard guidelines that the team has decided. Once the work is done, the developer can run unit testing to check for any bugs. If developers don’t do that, then it has to be added in some future steps in the pipeline to avoid sending bugs into production. Hence, you have to decide when you want to do Unit Testing in the pipeline.

All of this is based on the notion that the codebase is maintained. Github is the most popular VCS, but there are other options as well, like Bitbucket. As the code is pushed into the VCS, the build pipeline has to come into action to compile the whole project to generate the final build file, which will be used for production. It's highly recommended that before pushing the code into production. Unit testing is applied here to avoid pushing bugs into production. These unit tests are in the form of data and model validation. Also, during this process, static code analyzers, can check on the code quality so that you push clean into production.

Continuous Delivery

With CI, your project is stable and is ready to be deployed into production. In most cases, you won't be immediately deployed into production. If your client wants to see the build, you can deploy on a mock server to get feedback from your client. You are always capable of going into production. This is what Continuous Delivery looks like.

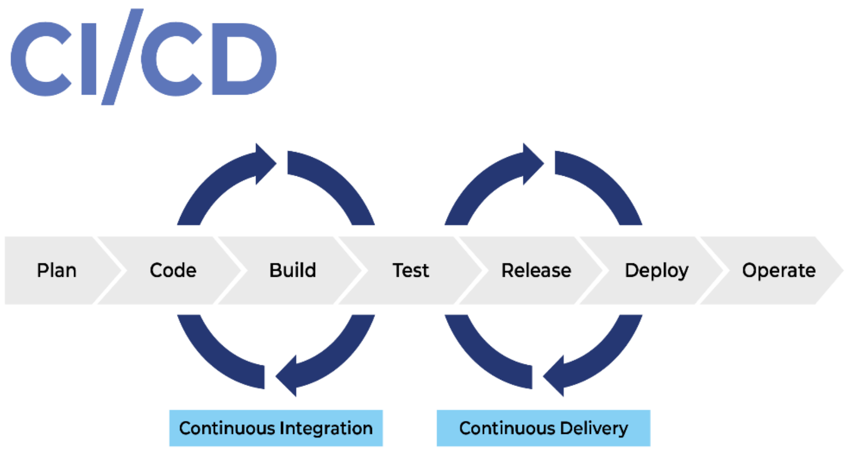


Figure 6: Continuous Integration and Delivery Workflow [33]

GitOps is a way to do Continuous Delivery. It works by using Git as a source of truth for declarative infrastructure and applications. The automated delivery pipeline automatically rolls out changes to your infrastructure when changes are made to Git. But the idea goes further - using tools to look at the actual production state, and tell you when what's source code doesn't match the real world, giving you the ability to spot the differences and fix problems accordingly.

In other words, GitOps extends pipelines with a feedback loop for observing and controlling the system. GitOps empowers developers to do operations.

To do this, GitOps aims to make developers more productive by applying familiar tools to hard things: operations management and monitoring. Every developer can use Git and make pull requests; now, they can use Git to accelerate and simplify operational tasks for Kubernetes. The benefits are far-reaching for a model for cloud-native CICD pipelines, faster mean time to deployment and mean time to recovery, actionable alerting, stable rollbacks and an overall coherent approach to understanding, observing, and managing apps.

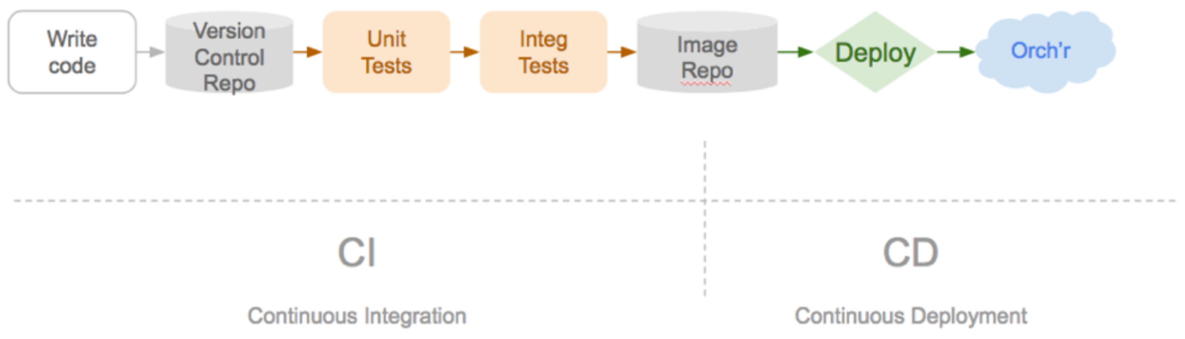


Figure 7: Continuous Integration and Delivery Workflow for GitOps [33]

Version Control and CI/CD tools can be used to automate the monitoring, tracking, and changes across the significant data analytics pipeline. A successful implementing of the CI/CD processes would increase the efficiencies and transparency across a release process. This process would include a version control system, a continuous integration server that can automate the build process, testing within that build process, and a deployment to a server that can run a container.

Chapter 5: Comparison of Similar Frameworks

To provide some scope of the novelty in the methodology of the defined framework in this paper. Research on current methodologies and frameworks was done. This researched showed that while there are methodologies that current utilize the tools and infrastructure that are mentioned above none of them account for the analytics portion of the framework. In general, there has been very little down to show how to integrate current development operations into a Big Data Analytics lifecycle from a development stand point.

In the paper Managing Life Cycles of Big Data Applications [34]. The paper focused predominately on the ability from one to integrate Big Data Infrastructure to account for the individual components of the Big Data Analytics Lifecycle. For example, the paper does an extensive review of the current pieces of Big Data Infrastructure that are used to enable Big Data Analytics. The authors walk through the life cycle that they are proposing by separating it into Development, Packaging, Composition, Enhancement, Deployment, Monitoring**,** Figure 8.

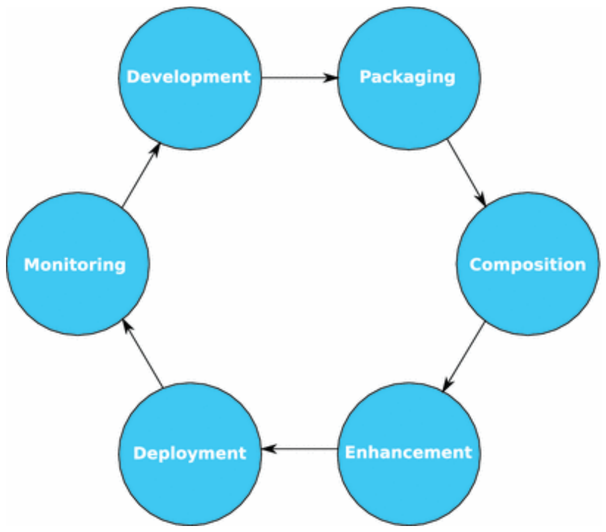


Figure 8: BDI Life Cycle Stack [34]

This lifecycle is specifically built around the Big Data Infrastructure. Meaning that each of these steps are for how to manage the life cycle of the infrastructure. This paper shows that it is helpful to consider an iterative lifecycle even when architecting the Big Data Analytics infrastructure framework piece out.

In a different paper [35] the same iterative life cycle to infrastructure is defined. In this paper the focus was around the containerization of workflows in a distributed embedded system environment. The paper shows the step by step process on how to create a distributed system using a cluster of Raspberry Pis' as individual nodes. The paper does a great job at explaining the technology stack that is used to create this cluster. This stack is relevant to the type of framework that is proposed in this paper because it uses containers with persistent storage. The architecture of this paper can be shown in Figure 9.

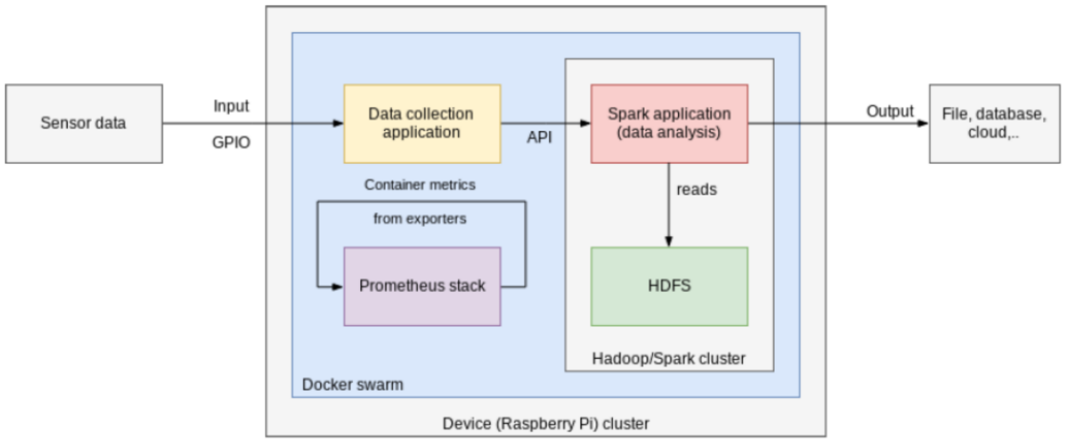


Figure 9: System Architecture of Distributed System [35]

The system architecture as shown in Figure 9**.** Reads in data in real time via IoT devices, then passes that data through a series of Raspberry Pis' the cluster and stores that data in persistent storage. This paper explains the benefits of developing how real-time Big Data Analytics systems in a containerized environment. Which is why the framework introduced in this paper utilizes the same concepts of containerization of processes attached to a persistent storage component.

In the paper From DevOps to DevDataOps [36] utilizes many of the same concept that are addressed in the proposed framework of this paper. These concepts are termed DataOps which is a term that represents development operations for data analytics. In this paper DataOps is described as a process that is intertwined with a CI/CD pipeline. The process is described as Plan, Code, Build, Test, Release, and Deploy. Which are the steps that are needed to have an Agile process. It then goes into all the different kinds of tools that are utilized to implement the CI/CD pipeline. This paper does a great job at explaining the benefits of DataOps and its uses. This paper does provide the challenges and proposes a summary solution on how to implement DataOps in an organization. The way that the CI/CD pipeline is configured is also shown in [39] when specifically, the infrastructure and the CI/CD pipeline is configured to run microservices. This microservice design pattern allows for developed to create a codebase that is not seen as a monolith [39]. Which is why in this paper the CI/CD and DevOps processes associated with the framework are in fact microservices. That described in this paper However, this paper does not provide a specific of a framework that could be useful for Big Data Analytics developers.

An actual use-case down by the The Volkswagen software development team. Showed that they developed a testing suite that was a part of their CI/CD framework [37]. Their suite provides a framework that utilizes some of the same technologies that are mentioned in this paper. Which is why it is included in this analysis. In their CI/CD framework they specific focus around the testing piece of an application. Which is important because it is the crossroads for not only Big Data Analytics but also the DevOps lifecycle. The Volkswagen team developed a framework that does Testing as a Service (TaaS). As stated in [37] the TaaS product enables developers to enjoy the benefits of rapid development velocity in the cloud native age. TaaS supports different kinds of tests. Which is essentially what the Big Data Analytics framework that this paper is proposing is also trying to achieve. Volkswagen's TaaS by automating the way that applications are deployed, tested and released allows for the software developers, to focus on planning and developing which is closer to developing business values [37]. In order to ensure all the benefits and the ongoing agility, the TaaS team developed the service in a more feedback driven way rather than a specification driven way.

The frameworks above show the benefits and provide use-cases for why the Big Data Analytics framework introduces in this framework makes sense. That said while some of components mentioned above are utilized. They do not account for every part needed for end to end development of Big Data Analytics. This paper explains the justification for the best type of software development project management style. It provides the technology stack used and shows how an analytics framework, such as CRISP-DM, can be implemented. All while using cutting edge infrastructure and processes that adheres to an Agile workflow methodology. The framework in this paper breaks down the methodology to checks all the boxes that are needed to implement BDA not just one aspect or another. From the way that the codebase is structured all the way to how releases are made to each component of the analytics lifecycle.

Chapter 6: Methodology and Implementation

As Big Data Analytics components have become more mature and integrated into larger software systems, our participants recognized the importance of integrating BDA development support into the traditional software development processes. However, achieving this level of integration can be challenging because of the different characteristics of BDA modules compared with standard software components. Nevertheless, due to the experimental and even more iterative nature of BDA development, unifying and automating the day-to-day pipeline of software engineers reduces overhead and facilitate progress in the field. The specific tools that we are used and how they should be structured to help with this overhead.

6.1 Constructing the Framework

NoSQL and InfluxDB

First and foremost, when it comes to Big Data Analytics, the way that the data is modeled and stored is most important. While through time, relational databases have been the go-to due to ease of use and ability to query. Current data structures are not as structured as they once were. This is where NoSQL databases have the upper hand in regard to modern Big Data Analytics. NoSQL databases are nontabular and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale effortlessly with large amounts of data and high user loads. NoSQL data models allow related data to be nested within a single data structure. NoSQL databases are better at allowing users to test new ideas and update data structures.

InfluxDB is a time-series database designed to handle high write and query loads. It is an integral component of the TICK stack. InfluxDB is meant to be used as a backing store for any use case involving large amounts of timestamped data, including DevOps monitoring, application metrics, IoT sensor data, and real-time analytics. InfluxDB is developed by InfluxData. It is an open-source, big data, NoSQL database that allows for massive scalability, high availability, fast write, and fast read. As a NoSQL, InfluxDB stores time-series data, which has a series of data points over time. These data points can be regular or irregular type based on the type of data resource. Some regular data measurements are based on a fixed interval time, such as system heartbeat monitoring data. Other data measurements could be based on a discrete event, such as trading transaction data and sensor data. InfluxDB is written on the go. This makes it easy to compile and deploy without external dependencies. It offers an SQL-like query language. The plug-in architecture design makes it very flexible to integrate other third-party products. Like other NoSQL databases, it supports different clients such as Go, Java, Python, and Node.js to interact with the database. Now that the storage aspect has been defined for the framework we need a solution that will maintain our codebase that can easily be utilized by Big Data Analytics Development Teams.

Version Control and Code Structure:

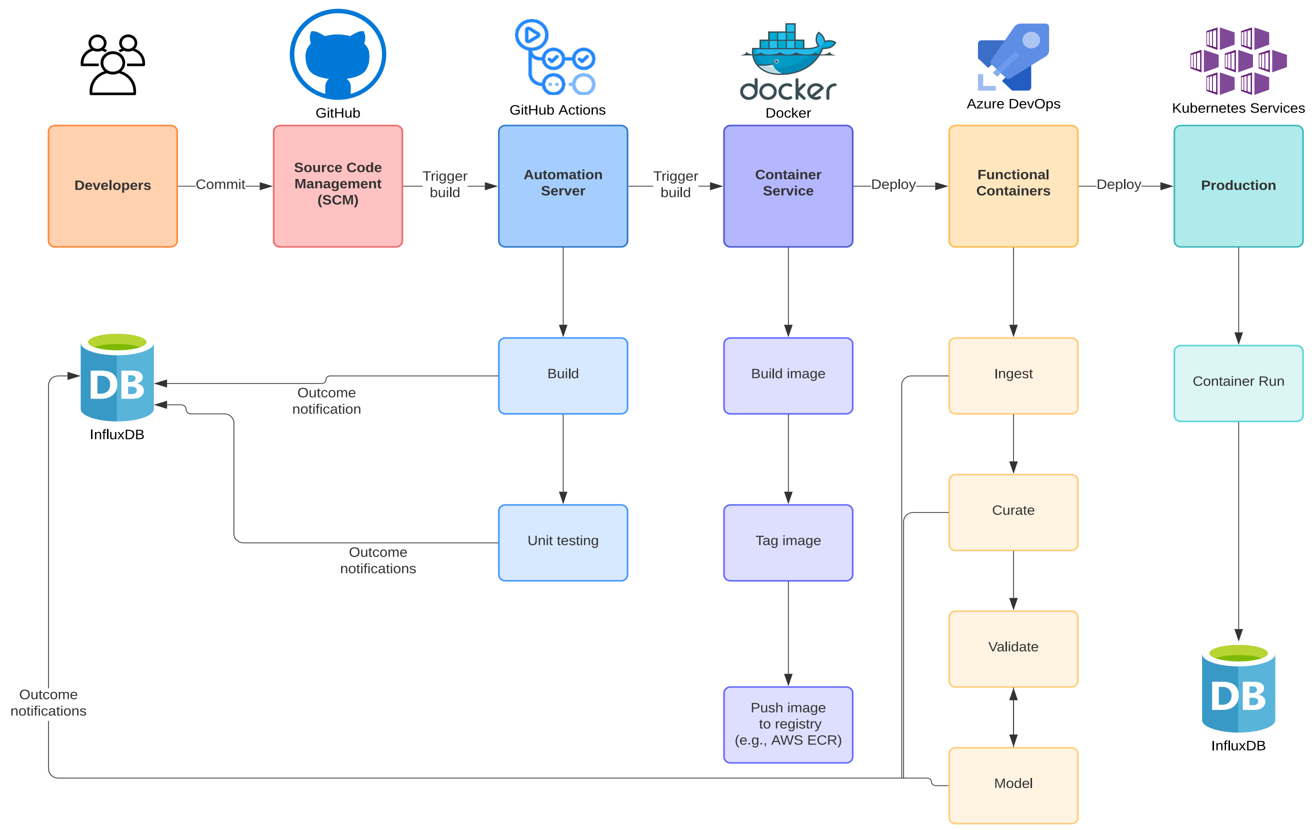
Our framework utilizes GitHub as the version control system. Github tracks the history of changes as people and teams collaborate on projects together. As the project evolves, teams can run tests, fix bugs, and contribute new code with the confidence that any version can be recovered. To effectively use GitHub, it is best to be consistent with how a project is set up. Below a customized code structure has been provided that will meet the needed requirements of a big data analytics project.

- docs
    - Documents folder used with GitHub pages for testing web-based visualizations and or custom web-based documentation.
- exploratory
    - Any ad-hoc or exploratory analytics codebase that is not considered to be in production and won't be actively maintained.
- source
    - The source directory is the primary working directory that developers operate out of.
- tests
    - Directory for all unit tests associated with the source code per deliverable. The files in this folder should mirror the generative files used to build a stream in the source folder.
- Dockerfile
    - The file that contains the build template of the container to be run
- Manifests
    - Contains the Kubernetes files that are used in the Azure DevOps pipeline to
- .workflows
    - The folder in which the GitHub Actions CI/CD build steps are defined

GitHub is the most widely used version control system and arguably the most powerful version control system on the market. In recent years GitHub has released new developer tools that enable more effective delivery of software products for development teams. For example, In 2019 GitHub Actions which is a service that allows development teams to implement CI/CD in a GitHub Repository. GitHub Actions makes it easier to automate how you build, test, and deploy your projects on any platform, including Linux, macOS, and Windows. As stated above the Big Data Analytics framework that we have defined needs the ability to deploy your workflows in a container dynamically. GitHub Actions enables us to keep our codebase and CI/CD implementation tightly coupled. The proposed framework thus utilizes the GitHub CI/CD service as a middleware to Microsoft Azure the cloud services. [36]

Microsoft Azure Services

Microsoft Azure, commonly referred to as Azure, is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through Microsoft-managed data centers. It provides software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS) and supports many different programming languages, tools, and frameworks, including both Microsoft-specific and third-party software and systems. Within the Azure Services the Big Data Analytics framework uses two of those



services, Azure Kubernetes Services, and Azure DevOps Pipeline both of which are handled by the GitHub Actions seen in Figure 10**.**

Figure 10: Infrastructure Framework Workflow from GitHub to AKS

Since our framework utilizes containers and dynamic resource provisioning, we need a container orchestration tool. Kubernetes is an open-source container management system used in large-scale enterprises in several vertical industries to perform a mission-critical task. Some of its abilities include Managing clusters of containers, providing tools for deploying applications, Scaling applications as and when needed, managing changes to the existing containerized applications, helping to optimize the use of underlying hardware beneath your container. Azure Kubernetes Service (AKS) is a managed containerized application more efficiently with a fully managed Kubernetes service. Azure Kubernetes Service (AKS) offers serverless Kubernetes, an integrated continuous integration and continuous delivery (CI/CD) experience, and enterprise-grade security and governance.

The Azure Kubernetes Services is coupled with the Azure DevOps pipeline this is operated by GitHub Actions. Azure DevOps Pipelines allows a developer to configure and track the build, test, and deploy everything created on GitHub. It does this by connecting the GitHub repository to a deployment server like AKS. The Azure DevOps pipeline operates off of GitHub Actions' continuous integration workflow. All of the build processes that are created can be tracked within the Azure DevOps Pipelines platform.

Now that the repository structure and the tools that we will use to execute our containers are defined. The containers that are deployed and executed need to be done organized functionally for this framework to stay agile. Meaning when new features or model builds are requested each container will not be impacting one another. This needs to be done across every aspect of the Big Data Analytics Framework pipeline. Since each aspect of the data pipeline is different. For example, the ingestion portion of the framework is different then the modeling portion. These differences result in challenging problems when BDA models are integrated into software systems at scale. Thus, it is essential to develop a rock-solid data pipeline, capable of continuously loading and massaging data, enabling engineers to try out many permutations of BDA algorithms with different hyper-parameters and different configurations without hassle. These pipelines created by Agile teams are automated, supporting training, deployment, and integration of models with the product they are a part of. To do this our pipelines execution and deployment need to be separated into the functional zones associated with each piece of work that is associated with Big Data Analytics.

Container Functional Zones

For effective continuous integration of the container repository, we will want to have a single repository for each part of the significant data analytics pipeline. The Repository structure that has been designed is a Multi-Repo structure. Thus, each repository can represent a different container. So, for an end to end pipeline, each of the containers allows for different kinds of developers to solve problems within their domain. We group each of the containers by the functional zone for each part of the framework. These functional zones are listed below:

- Ingest: Include any containers that load data into a database from an external resource
- Curate: Reads data from the database modifies it and then insert it back into tables
- Model: Any modeling that is done including machine learning, data mining, or analytics
- Validate: Model and data validations that are done on the incoming and outgoing data and any trained machine learning model
- Results: Graphs or Tables that show the results that are created from the trained model

Each of the functional zones above can have additional containers added on to it and are entirely independent of one another. At which point, if they need to make changes to the containers, they can without impacting the other containers. Once this is completed, the entire pipeline goes through promotion and moves into production. This framework is highly agile as stories can be created the focus on each aspect of the container. Given that the containers are independent of one another, they will not be blockers of one another. Now that the framework and its technology stack has been described. It is applied to a stock performance prediction problem.

6.2 Applying the Framework to a Stock Performance Prediction Problem

In the world of finance, there have been many attempts to use sophisticated algorithms to predict stock prices. Some progress has been made in predicting near-term [37] and long-term price changes [38]. The long-term prediction can predict with over 70 percent accuracy when only considering a limited number of stocks or sticks in the industry [39]. These working models often rely on the information of the company [40]. The main reason why stock price performance has been attempted so often is that if it is predicted correctly, then it is lucrative. This has created many novel ways of predicting stock price movement [41]. For example, stock price prediction can come from sentiment on breaking news, technical indicators derived via a company's balance sheet, or the moving average of the stock price through time. Data associated with each of the novel methods above are variable not only in frequency but also in the structure at which they are stored. Like how news data will be in an unstructured format while moving average data is structured. This makes stock price prediction an excellent problem to integrate into the framework. The framework can handle these variable data and can be shown to meet the needs of a significant data analytics pipeline effectively. To show the effectiveness of the framework, we will break down the stock prediction problem into its functional zones, Figure 11.

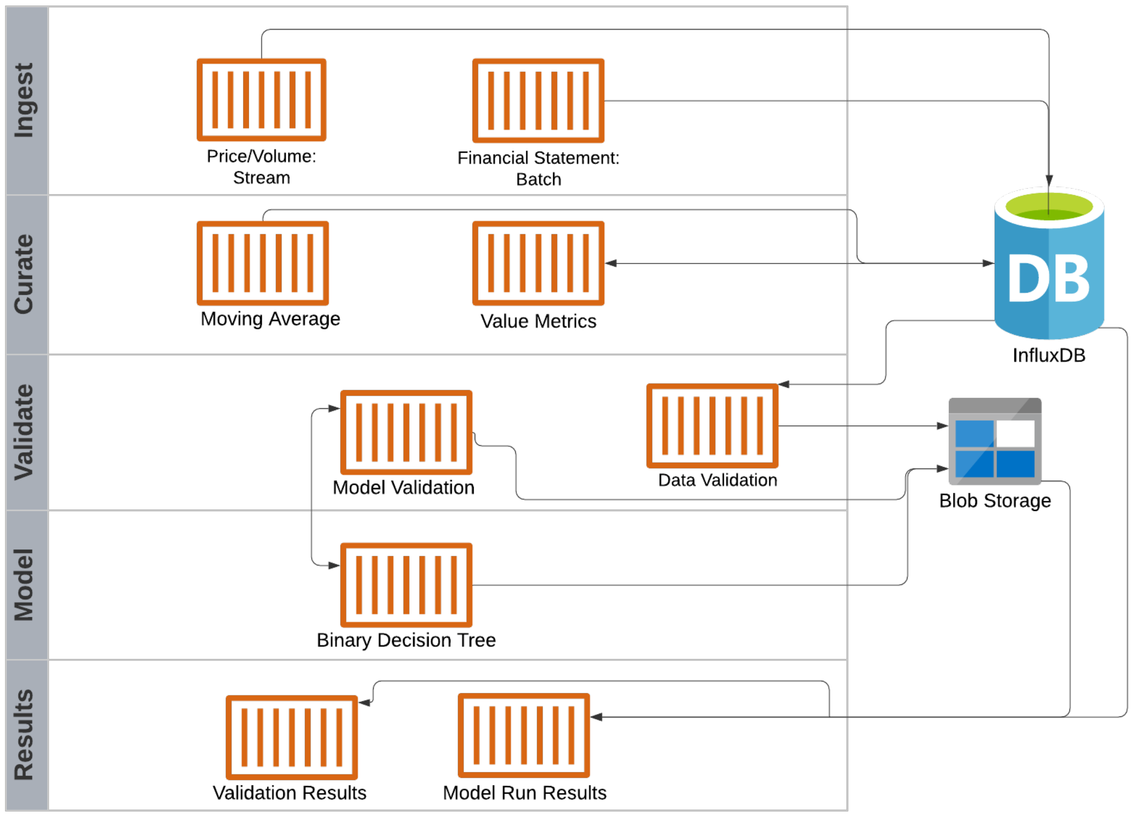


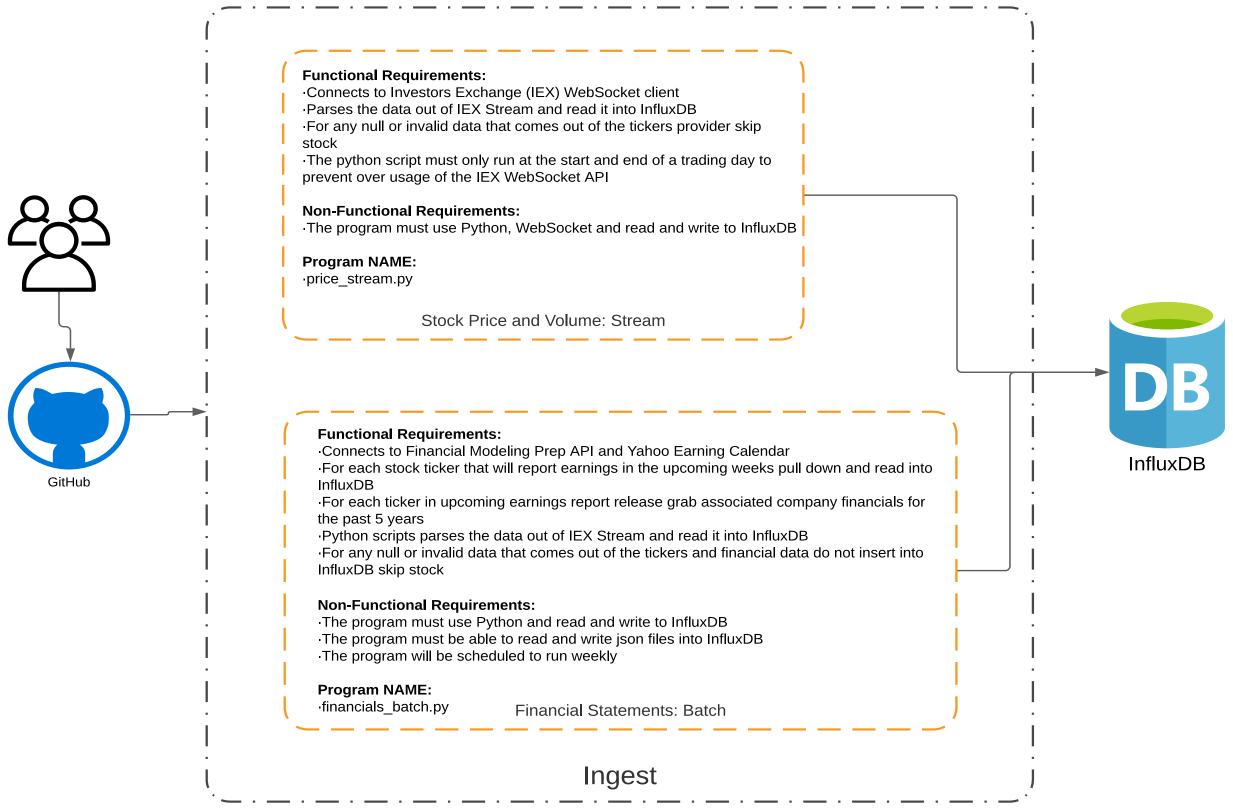
Figure 11: Container Functional Zones in the context of the Stock Prediction Problem

Ingest

The purpose of the Ingest Functional Zone, Figure 12, is to contain any codebase and their associated containers that will consume and store data into database environment. For the stock prediction the Ingest Functional Zone there are two containers. One of the containers is for spinning up a WebSocket consumer. The WebSocket consumer will stream in live data in real time from the IEX Exchange [42]. The data that is streamed is the current price and volume of all stock ticker changes for the stocks that are reported on the earnings calendar in the coming week. These data re then stored into a current price table. The earnings calendar data is pulled within the second batch container.

The batch container obtains a company's stock ticker in the upcoming week. The weekly batch build container will look at the companies that will report earnings in the coming week and pull down and insert those data into InfluxDB. The datasets that are used come from a few sources. The first is the earnings report data scraped from the Yahoo Earnings Calendar API [42]. The data that came from the IEX exchanged were real-time stock prices and the current PE ratio. The batch container quarterly financial statement dates and the daily open, close, high, and low prices associated with those dates came from the financial data API Financial Modeling Prep [43]. The Financial Modeling Prep API provides access to intraday time series, daily time series, weekly time series, monthly time-series data, and past US financial statements. These data are stored into a current financials table in InfluxDB.

The importance of the Ingest Functional Zone is if new sources of data need to be added. For example, if news data were to add for a sentiment analysis it can easily be done. This combined with the infrastructure to automatically deploy new containers into an associated zone without any thought of infrastructure is crucial for a highly agile and highly iterative work.



Functional Requirements:
·Connects to Investors Exchange (IEX) WebSocket client
·Parses the data out of IEX Stream and read it into InfluxDB
·For any null or invalid data that comes out of the tickers provider skip stock
·The python script must only run at the start and end of a trading day to prevent over usage of the IEX WebSocket API

Non-Functional Requirements:
·The program must use Python, WebSocket and read and write to InfluxDB

Program NAME:
·price_stream.py

Functional Requirements:
·Connects to Financial Modeling Prep API and Yahoo Earning Calendar
·For each stock ticker that will report earnings in the upcoming weeks pull down and read into InfluxDB
·For each ticker in upcoming earnings report release grab associated company financials for the past 5 years
·Python scripts parses the data out of IEX Stream and read it into InfluxDB
·For any null or invalid data that comes out of the tickers and financial data do not insert into InfluxDB skip stock

Non-Functional Requirements:
·The program must use Python and read and write to InfluxDB
·The program must be able to read and write json files into InfluxDB
·The program will be scheduled to run weekly

Program NAME:
·financials_batch.py

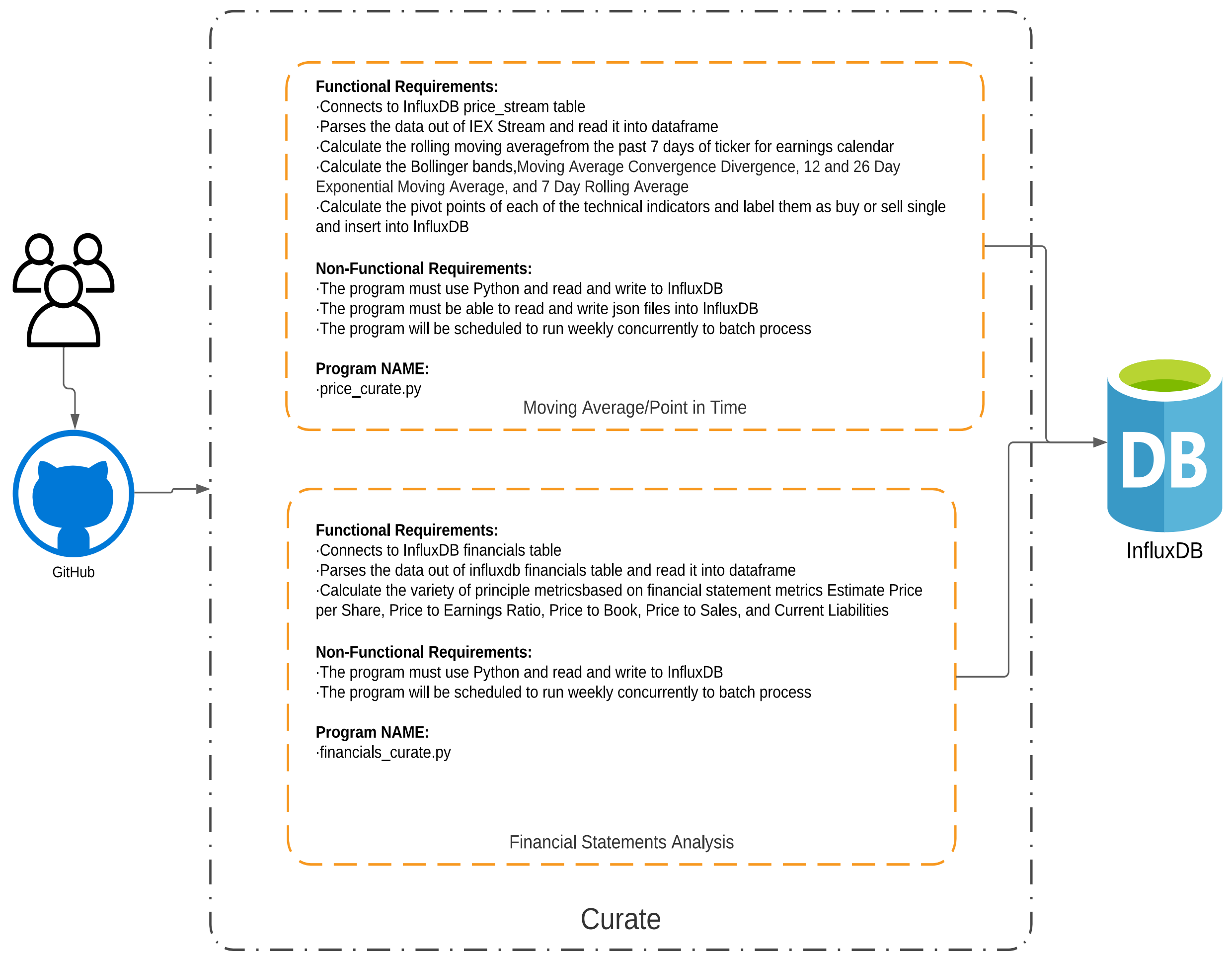
Figure 12: Ingest Functional Zone showing the Stream and Batch Containers

Curate

The Curate Functional Zone, Figure 13, houses any containers that involve feature engineering and data curation. In this Functional Zone the stock price prediction problem contains two containers. One container to handle the price movement data and another to handle the earnings and financial statement data. The price movement curation container reads in the current prices that are being streamed into InfluxDB. It then calculated a variety of metrics that are consider BUY or SELL signals. These metrics include the Exponential Moving Average, Moving Average Convergence and Divergence, Bollinger Bands, and Moving Average. Each of the moving averages from the current run with a lookback period of 14 days. For each of the stock tickers that are in the upcoming earnings calendar, these metrics are calculated and given a BUY and SELL Signal. Which can be translated into a True False Boolean binary operator.

The second container calculates the metrics associated with a company's financials that are released during an earnings report. Using the upcoming earnings calendar stock tickers value investing metrics are calculated per stock ticker. These metrics are derived via the reported P/E ratio, Price to Book, Price to Sales, Liabilities, and Earnings Per Share that is released on the companies balance sheet. These metrics are then aggregated up to the BUY and SELL per stock ticker. Both containers insert into two tables via a join on the stock ticker. One table is the historical metrics table that contains the calculated metrics over the past 5 Years' worth of quarterly financial data. The other table contains the current financials based on the quarter in which the job was run.

The Curation Functional Zone can be appended upon if a new metric or source of data becomes available. Which does not impact the upstream containers that are in the Ingest Zone but will allow for quicker analytics iterations downstream in the Model Functional Zone. Which is why it is important to have an intermediate step between the Ingest and Model zone.



Functional Requirements:
·Connects to InfluxDB price_stream table
·Parses the data out of IEX Stream and read it into dataframe
·Calculate the rolling moving averagefrom the past 7 days of ticker for earnings calendar
·Calculate the Bollinger bands,Moving Average Convergence Divergence, 12 and 26 Day Exponential Moving Average, and 7 Day Rolling Average
·Calculate the pivot points of each of the technical indicators and label them as buy or sell single and insert into InfluxDB

Non-Functional Requirements:
·The program must use Python and read and write to InfluxDB
·The program must be able to read and write json files into InfluxDB
·The program will be scheduled to run weekly concurrently to batch process

Program NAME:
·price_curate.py

Functional Requirements:
·Connects to InfluxDB financials table
·Parses the data out of influxdb financials table and read it into dataframe
·Calculate the variety of principle metricsbased on financial statement metrics Estimate Price per Share, Price to Earnings Ratio, Price to Book, Price to Sales, and Current Liabilities

Non-Functional Requirements:
·The program must use Python and read and write to InfluxDB
·The program will be scheduled to run weekly concurrently to batch process

Program NAME:
·financials_curate.py

Figure 13: Curate Functional Zone showing the requirements processing of incoming data

<u>Validate</u>

The Validation Functional Zone, Figure 14, contains two containers one for validating the trained model and the other for validating the incoming and curated data. The model validations consume an azure blob stored trained model. We then take a sample feature set and split it to include a training set. We then validate the model using a K-Fold Cross Validation, R2, Mean Square Error, and Build time. The other container in the validation zone will check incoming and curated datasets. The incoming data is checked for bad data types from the streaming set.

Meaning that if a stock price is not an integer an error is thrown. This is like the checks done to Relation Databases for schema consistency. The curated datasets that contain the features that will be fed into the model will be checked for nulls. If nulls exist or the data types are bad then the record is dropped. The count of records that are dropped due to nulls and data type errors are then stored in a log. These logs are then exported out as CSV and imported into azure blob storage with a time stamp. If the model is widely inaccurate then the developer can rework any of the zones to create a more accurate model. This allows for transparency across the development time and independent iterations on each aspect of the BDA lifecycle.

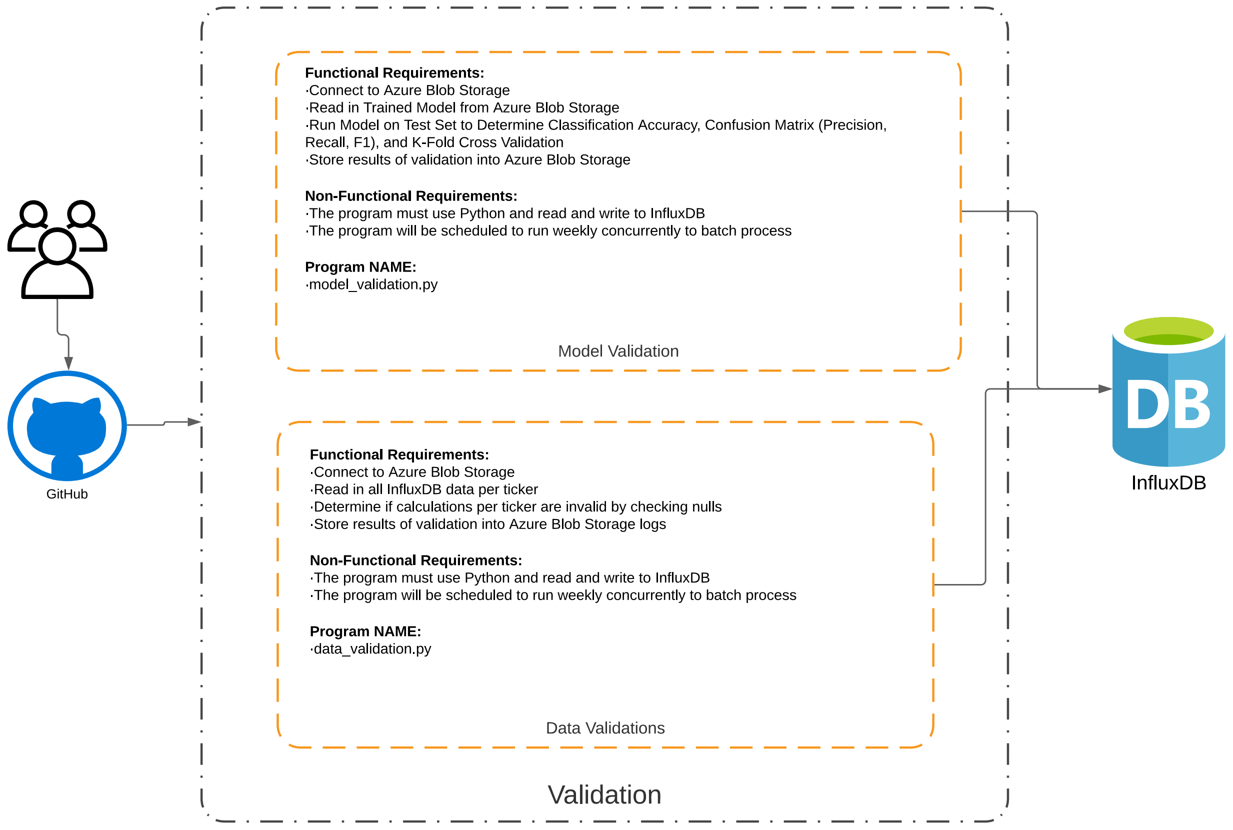

Functional Requirements:
·Connect to Azure Blob Storage
·Read in Trained Model from Azure Blob Storage
·Run Model on Test Set to Determine Classification Accuracy, Confusion Matrix (Precision, Recall, F1), and K-Fold Cross Validation
·Store results of validation into Azure Blob Storage
Non-Functional Requirements:
·The program must use Python and read and write to InfluxDB
·The program will be scheduled to run weekly concurrently to batch process
Program NAME:
·model_validation.py
Model Validation
Functional Requirements:
·Connect to Azure Blob Storage
·Read in all InfluxDB data per ticker
·Determine if calculations per ticker are invalid by checking nulls
·Store results of validation into Azure Blob Storage logs
Non-Functional Requirements:
·The program must use Python and read and write to InfluxDB
·The program will be scheduled to run weekly concurrently to batch process
Program NAME:
·data_validation.py
Data Validations
Validation
GitHub
DB
InfluxDB

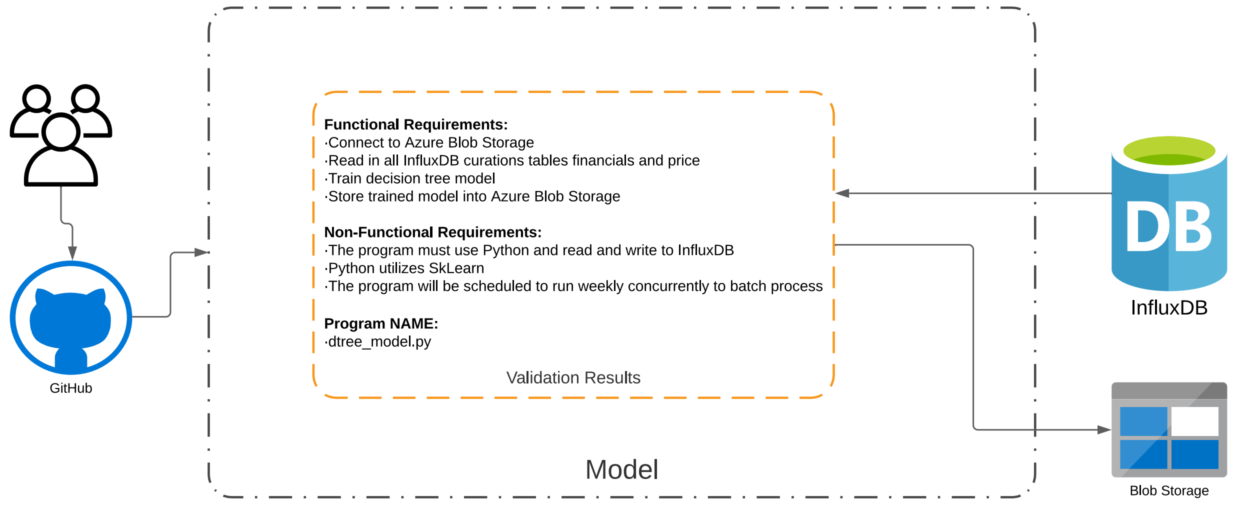
Figure 14: Validation Functional Zone for data and model validations

Model

Now to create the analytics a Model Functional Zone, Figure 15, exists to contain any quantitative or machine learning models. Stock price prediction is a problem that is highly researched as it can be lucrative if done accurately. For this example, we have created a Binary Decision Tree that is Trained on the BUY and SELL signals created in the Curate Functional Zone.

A Decision Tree Container is created to read in the Historical Metrics Table from InfluxDB. It then builds and trains the model based off these historical metrics data. This model is then exported out to the Azure Blob Storage. The Results Functional Zone will then pull down the trained model and visualize the prediction outputs.

The Model Functional Zone like all other zones can easily be added upon. Also, since the models are being stored into Blob Storage a Big Data Analytics Developer can easily pull down and compare different model outputs if need be. This continues to show that with this framework the development of Big Data Analytics can become highly iterative.



Functional Requirements:
·Connect to Azure Blob Storage
·Read in all InfluxDB curations tables financials and price
·Train decision tree model
·Store trained model into Azure Blob Storage

Non-Functional Requirements:
·The program must use Python and read and write to InfluxDB
·Python utilizes SkLearn
·The program will be scheduled to run weekly concurrently to batch process

Program NAME:
·dtree_model.py

Figure 15: Model Functional Zone showing the Decision Tree requirements

Results

After the model creation is done the results need to visualize for the end user. The Results Functional Zone, Figure 16, serves this purpose. This Functional Zone contains any codebase that will create reports or visualizations. This includes web application that are created to display interactive charts.

Big Data Analytics developers want to be able to view both the good and the bad. Two containers have been created to account for that. There is a model results container that host a Python Flask based web application that displays charts and tables of the stocks that should be bought. The results showed that there was some sort of medium to high accuracy within the model using the calculated the features. Although this was a promising outcome there are still a lot of things that could be improved upon to make this outcome better.

The second container is for validation results. This container exports out the any error reports and the model statistics through time. The validation results showed that the Decision Tree model was able to predict with 75% accuracy whether the stock was going to before the earnings statement was released.

Given the results for both the model and validations these can be used to go back to other functional zones to iterate over. For example, the accuracy of the model seems to be high so some overfitting might be happening. Within the Decision Tree container in the Model Functional Zone pruning the decision tree could be applied

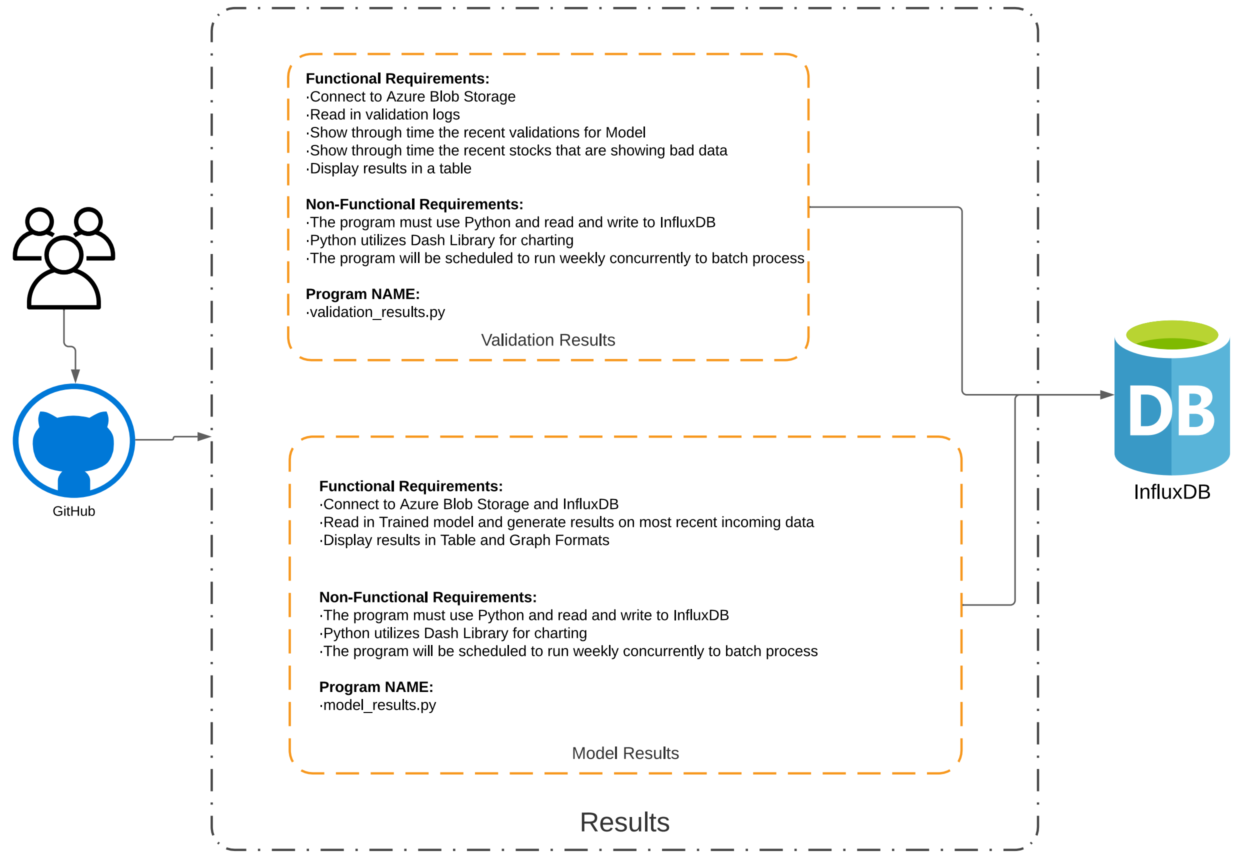

GitHub
Functional Requirements:
·Connect to Azure Blob Storage
·Read in validation logs
·Show through time the recent validations for Model
·Show through time the recent stocks that are showing bad data
·Display results in a table
Non-Functional Requirements:
·The program must use Python and read and write to InfluxDB
·Python utilizes Dash Library for charting
·The program will be scheduled to run weekly concurrently to batch process
Program NAME:
·validation_results.py
Validation Results
Functional Requirements:
·Connect to Azure Blob Storage and InfluxDB
·Read in Trained model and generate results on most recent incoming data
·Display results in Table and Graph Formats
Non-Functional Requirements:
·The program must use Python and read and write to InfluxDB
·Python utilizes Dash Library for charting
·The program will be scheduled to run weekly concurrently to batch process
Program NAME:
·model_results.py
Model Results
Results
DB
InfluxDB

Figure 16: Results Functional Zone showing the model and validations results

Chapter 7: Discussion

The entire framework end to end from infrastructure to building of the codebase to deploying via CI/CD to Azure Kubernetes is visually shown using the stock prediction problem. The stock prediction problem is broken down into appropriate analytics functional zones. Each of the containers are defined showing the type of requirements that are used within those functional zones. With the end results being that stock price and financial statements are being ingested into a NoSQL time series database, curated to create model features, a machine learning model being trained, both model and data being validated and finally results shown for each important aspect of the analytics lifecycle. This framework can easily be repeatable in a agile based Big Data Analytics development team. For example, say a Big Data Analytics team is given a new model to create. They could easily use the defined zones and insert each aspect of the life cycle easily into the functional zones.

The infrastructure that is provided in this framework solved for the main components of Big Data Analytics. The structuring, execution, integration, data movement and storage of Big Data Analytics. The cloud infrastructure is done in Microsoft Azure because of the seamless connection to GitHub and its associated CI/CD server Actions. Within GitHub the codebase is templated to structure each BDA component in a way that allows for any data science team to organize a codebase for a Big Data Analytics project. The way that the codebase is structured includes a generic configuration file that allows for the framework to handle the execution and integration of jobs that are being run in the Stock prediction problem. The execution of the job themselves are handled by the best-known practices of dynamically provisioning a containerized workflow in the cloud. Where dynamic provisioning is handled by the variable amount of resources depending on the size of the job you are trying to run. In terms types of Big Data movement in the cloud. The framework shows how the stock prediction problem was able to handle unstructured highly variable low latency data as well as structured in batch data.

The framework showed how the execution infrastructure is done in an Agile way that adheres to the analytics life cycle CRISP-DM. It does this by breaking down the high-level components of the CRISP-DM process into executable components called functional zones. These functional zones account for all the necessary components of a Big Data Analytics project. They include ingestion of data from a source, the curation of that source to meet the needs of the model, the validation of both data and model and then of course the results. The stock prediction problem shows how these zones are utilized and the associated functional and non-functional requirements of each piece of the stock prediction problem.

In the comparative analysis it of other frameworks while they are robust and fit into some aspects of framework defined in this paper. They do not account for every component needed for end to end development of Big Data Analytics. This paper explains the justification around the best type of software development project management style, provides the technology stack that is used and shows how an analytics framework such as CRISP-DM can be implemented. All while using cutting edge infrastructure and processes that adheres to an Agile workflow methodology. The framework in this paper breaks down the methodology to check all of the boxes that are needed to implement BDA not just one aspect or another. From the way that the codebase is structured all the way to how releases are made to each component of the analytics lifecycle. For example, the stock prediction problem shows that the ingest portion does not impact the modeling portion which allows for highly iterative development. Which means that this framework can be used to stand up different models or ingestion steps without impacting something upstream or downstream. By creating this functional zone and showing an example of the types of work that goes into them it gives a clear picture of the purpose and importance of this framework. These zones combined with the decoupling of environment to infrastructure as shown allows for even fast iterations.

Chapter 8: Conclusion

For many years it has been a best practice for a software development team to adhere to some sort of standards and development operations. The advent of new tools has made it easier for teams to automate their best practices and standards. The tools and associated processes have currently been for application development and releases. These processes can and should be utilized for teams using Big Data Analytics. Since the nature of Big Data Analytics development is different, the purpose of this research showed how to adopt current software development frameworks and provided how to use said framework in a modern architecture.

The researched showed that given the iterative nature of the Big Data Analytics lifecycle, that an Agile project workflow style aligns best for Big Data Analytics. This is because an Agile workflow is also iterative and recursive in nature. An agile workflow has a planning, development, testing, and deploying stages.

After the workflow process was defined it was necessary to understand the infrastructure associated with Big Data Analytics. In most modern solutions they include a cloud platform and the utilization of the services within that cloud platform. Given any cloud platform there are different ways of structuring the infrastructure in the cloud to be useful in Big Data Analytics. Researched showed that when creating the infrastructure for Big Data Analytics in the cloud there are best practices. These best practices include being able to dynamically provision resources given the volume of data. Using a container based approached so we can keep our environments interdependent of one another. Also, storing our data in a database that can account for the variety of data.

Given the best infrastructure for Big Data Analytics we want to be able to automate our workflows for any changes that are made. To do so a structure was defined to store our code using in GitHub. Now that the code can be versioned and tracked an of automation layer was added. This automation framework allowed for Continuous Integration and Continuous Delivery of our pipelines builds. Since our environment is containerized and dynamically resourced and on a CI/CD pipeline. Any build codebase can spin up and it won't impact another build. This allows us to create functional zones that house our containers. The functional zones are defined and structured to adhere to a Big Data Analytics Project. These functional zones are defined like the way that the CRSIP-DM and modern analytics life cycles. The functional zones include the ingest, curate, validate, model, and results zones. Within each of these zones contain codebases that are apart of the analytics lifecycle. Given the adherence to an Agile workflow developer can make changes to a container that is in one functional zone without impacting another.

This paper showed a useable and repeatable framework that can be used across development teams that are doing Big Data Analytics. With this framework a Big Data Analytics development team will create fast iterations of the models given an ever-changing data context. Considering applications to be built for big data analytics, it is essential to keep in mind the different conditions and challenges they impose. For one, they must handle the volume and variety of the data sets in use. Moreover, special considerations must be made to handle the velocity of big data. Also, as data sets sizes keep growing, these applications must have mechanisms to handle this growth during the different phases of data handling from generation and collection, organization and management, and performing the analysis and generating useful results.

References

1. Pressman, R. and B. Maxi. 2014, Software Engineering – A Practitioner's Approach, 8th edition, McGraw Hill Education, ISBN-13: 978-0078022128.

2. Al-Jaroodi, J. and N. Mohamed. 2012, "Middleware is STILL Everywhere!!!," in Concurrency and Computation: Practice and Experience, Wiley, Vol. 24, No. 16, pp. 1919-1926

3. Balaji, S., and M. S. Murugaiyan. 2012 , "Waterfall vs. V-Model vs. Agile: A comparative study on SDLC," in International Journal of Information Technology and Business Management, 2(1), pp. 26-30

4. Singh, M. (2020, October 29). Moving From Waterfall To Agile With Kanban. Retrieved June 03, 2020, from https://www.digite.com/blog/waterfall-to-agile-with-kanban/

5. Jacobson, I., G. Booch, J. Rumbaugh, J. Rumbaugh, and G. Booch, The unified software development process, Vol. 1, Reading: Addison-wesley, 1999.

6. Filman, R., T. Elrad, S. Clarke, and M. Akşit, Aspect-oriented software development, Addison-Wesley Professional, 2004.

7. Beynon-Davies, Paul, et al. "Rapid application development (RAD): an empirical review." European Journal of Information Systems 8.3 (1999): pp. 211-223.

8. Al-Jaroodi, J. and N. Mohamed. 2016, "Characteristics and Requirements of Big Data Analytics Applications," in Proc. 2nd IEEE International Conference on Collaboration and Internet Computing, Nov 2016, Pittsburgh, PA, USA

9. N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich. 2017, "Data management challenges in production Big Data Analytics," in Proc. of the 2017 ACM SIGMOD, pp. 1723–1726.

10. Russom, Philip. "Big data analytics." TDWI Best Practices Report, Fourth Quarter (2011): pp. 1-35.

11. Buyya, R., C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic. 2009, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation computer systems, Vol. 25, No. 6, pp. 599-616

12. D. Bohus, S. Andrist, and M. Jalobeanu, "Rapid development of multimodal interactive systems: a demonstration of platform for situated intelligence," in Proc. of the 19th ACM International Conference on Multimodal Interaction. ACM, 2017, pp. 493–494.

13. Gonidis et al. 2013. Cloud application portability: an initial view. In 6th Balkan Conference in Informatics. ACM.

14. Jrad et al. 2013. A broker-based framework for multi-cloud workflows. In intern. workshop on Multi-cloud applications and federated clouds.

15. Kurtzer et al. 2017. Singularity: Scientific containers for mobility of compute. PloS pp. 1-12, 5.

16. Pelk, H. (2017, June 14). Cloud container technology: The old and the new. Retrieved November 03, 2020, from https://itnext.io/cloud-container-technology-the-old-and-the-new-2a65ce6bd04d

17. Gerlach et al. 2014. Skyport: container-based execution environment management for multi-cloud scientific workflows. In 5th International Workshop on Data-Intensive Computing in the Clouds. IEEE Press, pp. 25–32.

18. Maria Alejandra Rodriguez and Rajkumar Buyya. 2017. A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. Concurrency and Computation: Practice and Experience 29, 8 (2017).

19. NIST. 2015, "NIST Big Data Interoperability Framework: Volume 1, Definitions," NIST Special Publication 1500-1, NIST Big Data Public Working Group (NBD-PWG) Definitions and Taxonomies Subgroup, doi: 10.6028/NIST.SP.1500-1.

20. Bizer, C., P. Boncz, M.L. Brodie, and O. Erling. 2012, "The meaningful use of big data: four perspectives--four challenges," ACM SIGMOD Record, pp. 56-60.

21. Massimo Cafaro and Giovanni Aloisio. 2011. Grids, clouds, and virtualization. In Grids, Clouds and Virtualization. Springer, pp.1–21.

22. Nachiappan et al. 2017. Cloud storage reliability for Big Data applications: A state of the art survey. Journal of Network and Computer Applications 97 (2017), pp. 35–47.

23. Data Lake vs. Data Warehouse - Working Together in the Cloud. (n.d.). Retrieved Augst 12, 2020, from https://panoply.io/data-warehouse-guide/data-warehouse-vs-data-lake/

24. Tudoran et al. 2016. Overflow: Multi-site aware big data management for scientific workflows on clouds. IEEE TCC 4, 1 (2016), pp. 76–89.

25. Suraj Pandey and Rajkumar Buyya. 2012. A survey of scheduling and management techniques for data-intensive application workflows. In Data Intensive Distributed Computing: Challenges and Solutions for Large-scale Information Management. IGI Global, pp. 156–176.

26. M. Salvaris, D. Dean, and W. H. Tok. 2018, “Microsoft AI Platform,” in Deep Learning with Azure. Springer, pp. 79–98.

27. C. Hill, R. Bellamy, T. Erickson, and M. Burnett. 2016, “Trials and tribulations of developers of intelligent systems: A field study,” in Visual Languages and Human-Centric Computing (VL/HCC), 2016 IEEE Symposium on. IEEE, pp. 162–170.

28. Microsoft (2019, December 10), The Team Data Science Process, Retrieved June 02, 2020 from https://docs.microsoft.com/enus/azure/machine-learning/team-data-science-process

29. J. Sillito and A. Begel. 2013, “App-directed learning: An exploratory study,” in 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), pp. 81–84.

30. J. E. Hannay, C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, and G. Wilson. 2009, “How do scientists develop and use scientific software?” in Proc. of the 2009 ICSE workshop on Software Engineering for Computational Science and Engineering. IEEE Computer Society, pp. 1–8.

31. G. De Michell and R. K. Gupta. 1997, “Hardware/software co-design,” Proc. of the IEEE, vol. 85, no. 3, pp. 349–365.

32. T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf. 2015, “Principles of explanatory debugging to personalize interactive Big Data Analytics,” in Proc. of the 20th International Conference on Intelligent User Interfaces, pp. 126–137.

33. Eneh, T. 2019. Most popular CI/CD pipelines and tools. Retrieved November 03, 2020, from https://medium.com/faun/most-popular-ci-cd-pipelines-and-tools-ccfdce429867

34. Ermilov, I., Ngomo, A. N., Versteden, A., Jabeen, H., Sejdiu, G., Argyriou, G., . . . Lehmann, J. 2017. Managing Lifecycle of Big Data Applications. *Communications in Computer and Information Science Knowledge Engineering and Semantic Web,* 263-276. doi:10.1007/978-3-319-69548-8_18

35. Scolati, R., Fronza, I., Ioini, N. E., Samir, A., & Pahl, C. 2019, A Containerized Big Data Streaming Architecture for Edge Cloud Computing on Clustered Single-board Devices. *Proceedings of the 9th International Conference on Cloud Computing and Services Science*. doi:10.5220/0007695000680080

36. Capizzi, A. 2020, From DevOps to DevDataOps: Data Management in DevOps Processes. Retrieved November 3, 2020, from https://www.researchgate.net/publication/338672377_From_DevOps_to_DevDataOps_Data_Management_in_DevOps_Processes

37. Poth, A., Werner, M., & Lei, X. 2018, How to Deliver Faster with CI/CD Integrated Testing Services *Communications in Computer and Information Science Systems, Software and Services Process Improvement,* 401-409. doi:10.1007/978-3-319-97925-0_33

38. Singh, C., Gaba, N. S., Kaur, M., & Kaur, B. (2019). Comparison of Different CI/CD Tools Integrated with Cloud Platform. *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. doi:10.1109/confluence.2019.8776985

39. Friedman, N. 2019, GitHub Actions now supports CI/CD, free for public repositories. Retrieved November 03, 2020, from https://github.blog/2019-08-08-github-actions-now-supports-ci-cd/

40. Rechenthin, Michael David. Machine-learning classification techniques for the analysis and prediction of high frequency stock direction. The University of Iowa, 2014.

41. Milosevic, Nikola. 2016, "Equity forecast: Predicting long term stock price movement using machine learning." arXiv preprint arXiv:1603.00751.

42. Shen, Shunrong, Haomiao Jiang, and Tongda Zhang, 2012. "Stock market forecasting using machine learning algorithms." Department of Electrical Engineering, Stanford University, Stanford, CA, pp. 1-5.

43. Bonde, Ganesh, and Rasheed Khaled. 2012, "Extracting the best features for predicting stock prices using machine learning." Proceedings on the International Conference on Artificial Intelligence (ICAI). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)

44. IEX Exchange – API Based Stock Market Data, iextrading.com N.p., 2020. Web. 13 Nov. 2019.

45. Yahoo Finance– Worldwide Financial News and Reports, yahoo.com N.p., 2020. Web. 13 Nov. 2019.

46. Technical Indicator Reference. Fmlabs.com. N.p., 2020. Web. 13 Nov. 2018.