



**Hewlett Packard
Labs**



DHBW
Duale Hochschule
Baden-Württemberg

**Development and demonstration of a proof-of-concept for the integration of
programming frameworks for high performance computing into a container-based
workflow orchestrator.**

2. Project Paper

submitted on October 25, 2023

Faculty for Business and Health

Degree program in Business Informatics

International Management for Business and Information Technology

by

JON ECKERTH

Practice supervisor :

⟨ Hewlett Packard Labs ⟩
⟨ Harumi Kuno, PhD ⟩
⟨ Principal Research Scientist ⟩

DHBW Stuttgart:

⟨ Dominic Viola ⟩

Signature of the practice supervisor:

Statement of Integrity

Hereby I solemnly declare:

1. that this 2. Project Paper titled "*Development and demonstration of a proof-of-concept for the integration of programming frameworks for high performance computing into a container-based workflow orchestrator.*" is entirely the product of my own scholarly work, unless otherwise indicated in the text or references, or acknowledged below;
2. I have indicated the thoughts adopted directly or indirectly from other sources at the appropriate places within the document;
3. this 2. Project Paper has not been submitted either in whole or part, for a degree at this or any other university or institution;
4. I have not published this 2. Project Paper in the past;
5. the printed version is equivalent to the submitted electronic one.

I am aware that a dishonest declaration will entail legal consequences.

Dublin, October 25, 2023



Jon Eckerth

Abstract:

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

| | |
|---|------------|
| List of abbreviations | IV |
| List of figures | VI |
| List of tables | VII |
| 1 Introduction | 1 |
| 1.1 Motivation | 1 |
| 1.2 Problem Statement | 2 |
| 1.3 Research Questions | 3 |
| 1.4 Contributions | 4 |
| 2 Methodology | 5 |
| 2.1 Prototyping | 5 |
| 2.2 Decision Making | 6 |
| 2.2.1 Weighted Sum Model | 6 |
| 2.2.2 Simple Multi-Attribute Rating Technique | 7 |
| 2.2.3 SMART Exploiting Ranks | 8 |
| 3 State of the Art | 9 |
| 3.1 Containerization | 9 |
| 3.2 High Performance Computing Frameworks | 9 |
| 3.2.1 Loosely Coupled Problems | 9 |
| 3.2.2 Tightly Coupled Problems | 9 |
| 4 Creation of the Artifact | 10 |
| 4.1 Initial Goals | 10 |
| 4.2 Overall Structure | 11 |
| 4.3 Selection of Workflow Management Tools | 12 |
| 4.4 Implementation of the Artifact | 16 |
| 4.4.1 Infrastructure | 16 |
| 4.4.2 Tightly Coupled HPC Workloads | 19 |
| 4.5 Evaluation of the Artifact | 22 |
| 5 Conclusion | 23 |
| 6 Summary and Outlook | 24 |
| 6.1 Summary | 24 |
| 6.2 Outlook | 24 |
| Appendix | 25 |
| List of literature | 54 |

List of abbreviations

| | |
|-----------------|--|
| AI | Artificial Intelligence |
| k8s | Kubernetes |
| ML | Machine Learning |
| HPC | High Performance Computing |
| CC | Cloud Computing |
| CI/CD | Continuous Integration/Continuous Delivery |
| IaC | Infrastructure as Code |
| PoC | Proof of Concept |
| HPE | Hewlett Packard Enterprise |
| CNCF | Cloud Native Computing Foundation |
| CLASP | Cloud Application Services Platform |
| CWL | Common Workflow Language |
| SME | Subject Matter Expert |
| TCP | Tightly Coupled Problems |
| LCP | Loosely Coupled Problems |
| SMART | Simple Multiattribute Rating Technique |
| WSM | Weighted Sum Model |
| MAUT | Multi-attribute Utility Theory |
| SMART-ER | SMART Exploiting Ranks |
| ROC | Rank order centroid |
| RS | Rank Sum |
| RR | Rank Reciprocal |
| PFS | Pachyderm File System |
| PV | Persistent Volume |
| VM | Virtual Machine |
| CNI | Container Network Interface |
| FAM | Fabric Attached Memory |
| SSO | Single Sign On |
| IP | Intellectual Property |

UDP User Datagram Protocol

RBAC Role Based Access Control

List of Figures

| | | |
|----|---|----|
| 1 | Formula for calculating the Weighted Sum Model (WSM) score ¹ | 6 |
| 2 | Formula for calculating the Simple Multiattribute Rating Technique (SMART) score ² | 7 |
| 3 | Formula for the Rank order centroid (ROC) weights | 8 |
| 4 | Formula for the Rank Sum (RS) weights | 8 |
| 5 | Formula for the Rank Reciprocal (RR) weights | 8 |
| 6 | Pachykouda high level diagram showing three main aspects | 11 |
| 7 | Pachykouda datum distribution amongst workers | 19 |
| 8 | Arkouda workers on Kubernetes (k8s) | 21 |
| 9 | Stars and Stackoverflow Questions Comparison | 26 |
| 10 | Swimlane Diagram of the communication between the user and Pachyderm | 30 |

List of Tables

| | | |
|---|--|----|
| 1 | Weighting of the criteria | 15 |
| 2 | Evaluation of the suggested tools | 15 |
| 3 | Evaluation of the additional tools | 15 |
| 4 | Comparison of project popularity | 26 |
| 5 | Evaluation of the suggested tools | 28 |
| 6 | Evaluation of the additional tools | 29 |

1 Introduction

In this section, the underlying motivation of this project is explained. Furthermore, the problems which will be addressed by this project are described, which serve as the basis for the research questions which will guide this project and ultimately result in solutions and further questions which are listed in the contributions section and discussed in the conclusion.

1.1 Motivation

The proliferation of "Big Data" has led to the need to compute, analyse, and visualize ever-increasing amounts of datasets, which themselves are getting more and more complex, has led to an ever increasing demand for more efficient and quicker ways to process data.

Both the High Performance Computing (HPC) and the Cloud Computing (CC) community have been working on solutions to distribute and parallelize computations for decades, both with their own approaches and solutions to their respective problems.

While the HPC community has been putting a lot of effort into developing new and extremely efficient ways to parallelize computations, the CC community has been focusing on improving the flexibility, scalability and resilience of their solutions as well as improving the ease of use for their developers and users.

Both used to be very distinct and separate communities due to their very different usecases, while the HPC community was mostly concerned with scientific computing and simulations of physical phenomena, the CC community is mostly concerned with providing a reliable and easily up and down scalable infrastructure for the industry and businesses.

Now with the advent of Machine Learning (ML) and Artificial Intelligence (AI) the two communities are starting to converge, as the ML and AI community is adopting the tools and techniques of both communities to solve their problems as they see fit.

But this convergence of the two is not without its problems, being developed in two coexisting and separate communities, the tools and techniques of both communities are not always compatible with each other, the goal of this project is to find a way to bridge this gap and to find a way to combine the best of both worlds.

1.2 Problem Statement

The following key problems have emerged from the convergence of High Performance Computing (HPC) and Cloud Computing (CC) communities, especially in the context of Machine Learning (ML) and Artificial Intelligence (AI) research:

- **Workload Resilience and Fault Tolerance in HPC:** HPC systems often lack mechanisms to recover from task failures within larger jobs, running for an extended time. When a component task fails, it can invalidate the entire computation, requiring a restart from scratch. This need for resilient failover and verification strategies as well as the need to avoid computational wastage is a key challenge for HPC systems, especially with every increasing system sizes and complexity.³
- **Environment/Package Management in HPC:** HPC systems are notorious for their complex package management systems. As having a shared infrastructure between many users each with their own specific needs and requirements of different versions of packages, libraries and software, all the while sharing a common environment. Many solutions to this problem have been developed, each with their own advantages and disadvantages.⁴⁵⁶⁷
- **Portability Issues with HPC:** Tying in with the previous point, HPC systems are often designed to be optimized for specific hardware as well as having a very specific software stack. This makes the portability of applications between different HPC systems very difficult and often infeasible.⁸ This lack of portability contrasts sharply with the more platform-agnostic nature of CC environments, where the containerization of applications has become the norm for ensuring portability.
- **Scalability and Flexibility in HPC:** Due to its direct access to the hardware and very specific hardware needs, HPC systems are often hard to dynamically scale and inflexible. while CC systems are designed to be easily scalable and flexible and are often designed to be hardware agnostic and abstract away the underlying hardware. This becomes especially relevant in the context of heterogeneous hardware, where the hardware is not uniform and consists of different types of hardware, which is becoming more and more common in the context of ML and AI research.
- **Lack of Interconnected Problem Solving in CC:** The workloads traditionally deployed on CC systems are often independent of each other, like load balancing, web hosting, etc. This is in stark contrast to the interconnected nature of HPC workloads, where each part of the input data is dependent on the other parts of the input data, such that all nodes of the system need to be able to communicate with each other.

³Egwutuoha et al. 2013

⁴Dubois/Epperly/Kumfert 2003

⁵Bzeznik et al. 2017

⁶Gamblin et al. 2015

⁷Hoste et al. 2012

⁸Canon/Younge 2019, p. 50

- **Provenance and Reproducibility:** Another need that is becoming more and more important in the context of ML and AI research is the need for provenance and reproducibility of results. Being able to tell which data was used to train the model, is of ever increasing importance as the influence the resulting models have on our lives increases as well as the data used to train the model. This is especially important since it is crucial to ensure that the data is not biased, outdated, or otherwise flawed, which could lead to incorrect predictions, decisions, or recommendations. In addition various data sources, from images to text, may have copyright restrictions that, when overlooked, can lead to significant legal complications.
- **Versioning Limitations:** The dynamic nature of ML and AI research necessitates robust versioning solutions for data, configurations and code. While CC has developed many solutions to this problem over the years, making them their own subsection of the ecosystem, namely Continuous Integration/Continuous Delivery (CI/CD) tools for the testing and deployment of applications as well as Infrastructure as Code (IaC) tools for the deployment of infrastructure. While many solutions have been developed for the one-off deployment of HPC systems, the dynamic nature of CC systems necessitates a more robust solution to this problem, from which the HPC community could benefit as well.

1.3 Research Questions

To address the aforementioned problems, to bridge the gap between the two paradigms and to combine the best of both worlds, an integration of the two paradigms is needed. This was accomplished by integrating a HPC framework called 'Arkouda'⁹ into a container based CC workflow management tool called 'Pachyderm'¹⁰ and integrating both with the supporting infrastructure the CC system enables us to use. This process of integration and prototyping as well as the explanation of the underlying concepts and technologies will be the focus of this project.

- **RQ1:** *How can a high-performance computing framework be effectively integrated into a container-based workflow management tool?*
- **RQ2:** *What are the opportunities for improving the integration of high-performance computing frameworks with container-based workflow management tools?*

⁹Merrill/Reus/Neumann 2019

¹⁰**pachydermPachyderm**

1.4 Contributions

In order to address the problems stated above, find answers to the research questions and to bridge the gap between the two paradigms, the following contributions were made:

- **C1:** *An analysis of the problem space and existing solution, within the constraints of time, resources and businesses needs.*
- **C2:** *A prototype implementation combining the 'Arkouda' framework with the k8s based workflow orchestrator 'Pachyderm'*
- **C3:** *Further integrations of tools from both sides of the spectrum, addressing many of aforementioned pain-points*

2 Methodology

2.1 Prototyping

Needs to have a methodology from the Spectrum of Methodologies for Business information systems¹¹

Argumentation why this project is centrally a Prototyping project:

- The research questions are directly inspired by the needs of the customer
- The limitations and the scope are both defined by the available resources of the business unit as well as the time constraints of the project and the available know-how
- Based¹² can be classified as a presentation prototype in which we do a vertical integration of many different systems, according to Budde this can be described as a vertical interface, as it reaches through the entire stack of technological abstraction¹³
- to create this prototype we will be using Which will be using spiral model¹⁴

¹¹Wilde/Hess w.y.

¹²Budde et al. 1992, p. 91

¹³Budde et al. 1992, p. 94

¹⁴Boehm 1988

2.2 Decision Making

As previously described, the methodology of Prototyping benefits from a very tight loop of iterations between the different phases of the project. While this is highly effective in producing a good end result, it can also take many iterations and a lot of experimentation until an adequate tool or solution has been found. Given the constraints of a limited time frame for this project, it becomes crucial to use this time as efficiently as possible. Sometimes, when the time does not permit a thorough exploration of

To ensure that the decisions made are the most optimal within the constraints of the available information, adopting a systematic, replicable, and transparent decision-making process becomes essential. Over the years, various frameworks have been crafted to guide decision-making, particularly when information is complex and multi-dimensional.

2.2.1 Weighted Sum Model

Evangelos Triantaphyllou suggests that the Weighted Sum Model (WSM) is in practice the most used and most relevant decision-making framework¹⁵. The WSM method, by design, mandates the assignment of specific weights to each criterion based on its relevance. Subsequent to this, every alternative is evaluated based on these weighted criteria, resulting in a cumulative score. The alternative with the highest score is therefore the optimal choice.

$$A_i^{WSM-score} = \sum_{j=1}^n w_j a_{ij} \quad for \ i = 1, 2, 3, \dots, m.$$

Abb. 1: Formula for calculating the WSM score¹⁶

Where:

- !TODO! add explanations of variables

This method, despite its simplicity and direct approach, isn't without limitations. One notable drawback is its dependence on dimensionless scales. For the weights to properly reflect the criteria's importance, the scores need to be on a common, dimensionless scale, a detail not always feasible or convenient in practice.

¹⁵Triantaphyllou 2000, p. 1

¹⁶*Weighted Sum Model* 2022

2.2.2 Simple Multi-Attribute Rating Technique

In contrast to the WSM, which predominantly utilizes a direct mathematical approach to rank alternatives based on their weighted sum scores, the Simple Multiattribute Rating Technique (SMART) methodology offers a more comprehensive approach to multi-criteria decision-making. While WSM is primarily concerned with simple weighted arithmetic sums, the SMART method dives deeper, ensuring that diverse performance values—both quantitative and qualitative are harmonized and placed on a common scale.

The SMART method, grounded in Multi-attribute Utility Theory (MAUT), provides a structured framework that encompasses more than just the weighting of criteria. It involves:

1. Discernment of vital criteria pertinent to the decision in focus.
2. Weight allocation to each criterion in accordance to its significance.
3. Evaluation of each potential alternative against the identified criteria, culminating in a score.
4. Aggregation of these individual scores via their associated weights, yielding a total score for every alternative.

By adhering to the SMART framework, alternatives can be sequenced based on their aggregated weighted scores. This systematic approach equips decision-makers to choose solutions that align closely with their objectives. The computational formula integral to the SMART method is:

$$x_j = \frac{\sum_{i=1}^m w_i a_{ij}}{\sum_{i=1}^m w_i}, \quad j = 1, \dots, n.$$

Abb. 2: Formula for calculating the SMART score¹⁷

Where:

- x_j Is the overall utility score for alternative j . The higher the score, the better the alternative, in comparison to the other alternatives.
- a_{ij} Is the utility score for alternative j for the criterion i .
- w_i Is the weight of criterion i .

This method's emphasis on utility functions ensures a more nuanced and adaptable approach to decision-making compared to models like WSM, making it suitable for complex scenarios where criteria and alternatives are diverse in nature¹⁸.

¹⁷Taken from Fülöp 2005, p. 6

¹⁸Fülöp 2005, p. 6

2.2.3 SMART Exploiting Ranks

The SMART Exploiting Ranks (SMART-ER) method is a variant of the SMART method that attempts to alleviate the largest issue of the original SMART method, namely the problem of a somewhat arbitrary ranking of the options if no numerical values can be derived.

This method addresses the issue by letting the decision maker simply ranking the different criteria in relation to each other and then normalizing the weights¹⁹. They propose the different weighting curves.

$$w_i(ROC) = \frac{1}{n} \sum_{j=1}^n \frac{1}{j}, \quad i = 1, \dots, n.$$

Abb. 3: Formula for the ROC weights

The ROC takes the centroid of the rank order and uses the reciprocal of the rank as the weight.

$$w_i(RS) = \frac{n+1-i}{n(n+1)/2}, \quad i = 1, \dots, n.$$

Abb. 4: Formula for the RS weights

The RS uses linear curve where weights are normalized by dividing them by the sum of all weights.

$$w_i(RR) = \left(\frac{\frac{1}{i}}{\sum_{j=1}^n \frac{1}{j}} \right), \quad \text{rank } i = 1, \dots, n, \quad \text{option } j = 1, \dots, n$$

Abb. 5: Formula for the RR weights

The RR emphasizes the most important criteria by using the reciprocal of the rank as the weight, then normalizing each weight by the sum of all reciprocals.

¹⁸Multi-Criteria Decision Analysis for Use in Transport Decision Making 2014, p. 26

¹⁹Roberts/Goodwin 2002, p. 296

3 State of the Art

3.1 Containerization

Container Solutions

Software defined Infrastructure

Large Scale Container Orchestration

3.2 High Performance Computing Frameworks

3.2.1 Loosely Coupled Problems

Loosely Coupled Problems (LCP) also known in the industry as "embarrassingly parallel"²⁰ problems are problems that can be broken up into smaller independent tasks that can be executed in parallel.

tools like Mapreduce and Spark

3.2.2 Tightly Coupled Problems

In contrast to LCP problems, Tightly Coupled Problems (TCP) problems are problems that can not be broken up into smaller independent tasks that can be executed in parallel, instead of working independently, each atomic task needs to communicate at least with one other task. A good example of a TCP problem are the n-body problems, where the position of each body is dependent on the position of all other bodies.

tools like MPI, OpenMP and Chapel

²⁰smtn

4 Creation of the Artifact

4.1 Initial Goals

As this project was first and foremost a project, designed to interactively explore the problemspace from the perspective of the HPC community, all the while being contained by business requirements and time constraints, the initial goals of this project were very broad and open ended. At first the initial goal was simply to create a Proof of Concept (PoC) of a realistic workflow engine using the "Arkouda" project, in order to present the Customer with a easily graspable example of its capabilities.

While we are approaching the problem from the perspective of the HPC community, the intended enduser of this tool are the data scientists and Subject Matter Expert (SME)s that are working with the HPC systems, and therefore the tool needs to be designed and selected with the fact in mind that the enduser will most likely not be knowledgeable in the field of HPC or the underlying infrastructure.

In the first iteration of the project a preselection of possible Workflow management tools was given from the business side, with the option to increase the scope if the presented tools were not sufficient.

Therefore the goals of the first iteration of this project was twofold, first to determine which, if any, of the presented tools were suitable for the task at hand, and to determine what would make an adequate PoC for the customer.

The following iterations are split into the tree main aspects of the project and will be discussed in their own subsections. While these steps where happening concurrently, they each address a different aspect of the project and therefore underwent their own iterative processes.

4.2 Overall Structure

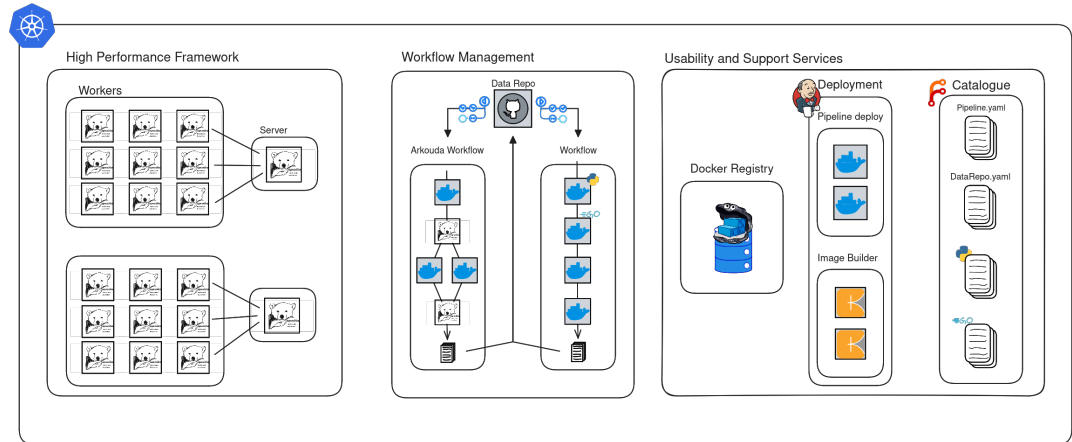


Abb. 6: Pachykouda high level infrastructure diagramm

As can be seen in figure 6, the artifact is composed of 3 main components, the **Central Workflow Engine** which is responsible for the orchestration of the workflows (center) and interfaces directly with the underlying infrastructure, the **HPC Framework** which is responsible for the execution of TCP workloads (left) and the **Supplementary Services** which aim at improving the usability and accessibility for the enduser (right).

All this is build ontop of a hardware agnostic k8s cluster, which is responsible for the orchestration of the different components and the underlying infrastructure.

4.3 Selection of Workflow Management Tools

As described in section 4.1, the first iteration of this project was to determine which, if any, of the presented tools were suitable for the task at hand. The following section will describe the process of selecting the tools and the criteria that were used to evaluate them. Because the time frame does not allow for a full integration and testing of all the presented tools in depth we will be using a decision making framework to evaluate the tools, as described in the Methodologies 2.2 to determine which tools will be most suitable for an initial PoC and will serve as a good starting point for the project and future iterations.

- **Pachyderm:** A k8s based Workflow manager, written in go which was recently acquired by Hewlett Packard Enterprise (HPE).
- **Argo:** A k8s based Workflow manager , written in go, which is a Cloud Native Computing Foundation (CNCF) project²¹.
- **Cloud Application Services Platform (CLASP):** An in-house developed workflow manager, written in Java, utilizing Servlet to execute workflows²².
- **Snaplogic:** A commercial low-code/no-code workflow manager with a focus on data integration and data engineering²³.

But given that it was possible to select projects outside of the initial selection, the following projects also need to be considered:

- **Airflow:** A Python-based workflow manager under the CNCF umbrella, known for its easy-to-use interface and extensibility²⁴.
- **Kubeflow:** A k8s-native platform for deploying, monitoring, and running ML workflows and experiments, also a CNCF project, streamlining ML operations alongside other Kubernetes resources²⁵.
- **Knative:** An open-source k8s-based platform to build, deploy, and manage modern serverless workloads, simplifying the process of building cloud-native applications²⁶.
- **Luigi:** An open-source Python module created by Spotify to build complex pipelines of batch jobs, handling dependency resolution, workflow management, and visualization seamlessly²⁷.

²¹Argoproj/Argo-Workflows 2023

²²Sayers et al. 2015

²³iPaaS Solution for the Enterprise 2023

²⁴Haines 2022

²⁵Kubeflow 2023

²⁶Home - Knative 2023

²⁷Spotify/Luigi 2023

- **Common Workflow Language (CWL):** An open-standard for describing analysis workflows and tools in a way that makes them portable and scalable across a variety of software and hardware environments, from workstations to cluster, cloud, and high-performance computing environments.

Selection Criteria

Due to this extensive list of diverse tools, a set of criteria was established to determine which tool would be the most suitable for the task at hand. The following list of criteria was established to evaluate the tools:

- **Ease of use:** As the intended endusers of the tool are not primarily HPC experts, the tool needs to be easy to use and understand, and should not require the enduser to have a deep understanding of the underlying infrastructure. While we can expect that the administration of the infrastructure will be done by adequately trained personnel, the enduser should be spared having to adapt to the underlying infrastructure as much as possible.
- **Extensibility:** One significant constraint of the project is the restricted number of available work-hours. Given that the project's environment predominantly centers around HPC (High Performance Computing) workloads, it's essential for the tool to be easily expandable without requiring extensive modifications to the underlying system. Ideally this property would be transferred to the enduser, allowing them to easily extend the developed tool further to their needs.
- **Community, Support and Documentation:** It is not enough that the software technically permits extensibility, the software also needs to be adequately documented and a support framework needs to be in place. Be it a community of users or a dedicated support team, the enduser and the developers need to be able to rely on the software being maintained and updated as well as being able to find expert help in case of problems.
- **Maturity:** With the boom of AI and ML in recent years²⁸, the number of tools and frameworks has exploded, and while this is a good thing it also means that a lot of these tools are still paving their way and are developing rapidly. While this is not necessarily a bad thing, it does mean that the tool might not be ready for production use and might not be able to provide the stability and reliability that is required for a production environment or are lacking in documentation and support.
- **Strategic alignment with HPE:** As this project is being developed within the context of HPE, it is important to consider the strategic alignment of the tool with HPE. HPE has is a large company with a diverse portfolio of products and services, and this project

²⁸ *24 Top AI Statistics & Trends In 2023 – Forbes Advisor 2023*

intersects with many different parts of the company. Therefore it is important to consider the strategic alignment of the tool with HPE and its products and services.

- **License:** While this PoC is not a commercial product in itself but rather an exploration of the problem space and a demonstration of what a final commercial product might be like, it is important to consider the licenses of the tools that are being used. Having to strip out a tool later on because of licensing issues would be a significant setback and therefore needs to be considered.
- **Cost:** Time is not the only constraint of this project, as the project is being developed within the context of HPE it is important to consider the cost of the tools that are being used.

Weighting of the Criteria

An integral part of the SMART methodology is the weighting of the criteria, as described in section 2.2. In order to rank the criteria themselves, as they are quite hard to quantify, We will be using the weighing methodology as described in the SMART-ER methodology 2.2.3.

The first step of which is the ranking of the criteria from most important to least important.

1. **Extensibility** As this is first and foremost a prototyping project, the actual development it at least for the first couple steps of the highest importance.
2. **Community, Support & Docs** This also applies for the external support available to the development team as if they are stuck, no developed can proceed, no matter the other factors.
3. **License** This criterion has to weighted carefully, as a highly restrictive license might be a dealbreaker, but a license that is too permissive might conflict with the strategic alignment with HPE.
4. **Strategic alignment with HPE** As this is developed by and for HPE their requirements need to be consider aswell.
5. **Ease of Use** While the ease of use is important as this should eventually become a product, for now the central aspect is to create a PoC therefore the usability is a priority, but not the highest.
6. **Cost** As this is a PoC and not a commercial product, the cost is not the highest priority as this will be of small scale and therefore the cost will be negligible in most cases.
7. **Maturity** While the maturity of the tool is important, as this is a PoC and not a commercial product, if the maturity of the tool does not impact the extensibility of the tool or the development process, it is not the highest priority.

As all these criteria are quite important, the weighting function selected for the criteria is the RS function, as described in section 2.2.3, as it does not rank the criteria too harshly. The lookup tables for the weighting function can be found in the appendix ??.

| Criteria | Weight |
|--------------------------------------|--------|
| Extensibility | 0.2500 |
| Community, Support and Documentation | 0.2143 |
| License | 0.1786 |
| Strategic alignment with HPE | 0.1429 |
| Ease of use | 0.1071 |
| Maturity | 0.0714 |
| Cost | 0.0357 |

Tab. 1: Weighting of the criteria

Evaluation of the Tools

Now that we have established the criteria aswell as their weighing, we can beginn to evaluate the tools based on the criteria. Here we will be using a mix of Methodologies, as some of these criteria can simply be indexed via analogous values, while others are of a more non specific nature. The discussion of which values will be used on which weighing scale for the tools comparison can be found in the appendix under

| Criteria | Pachyderm | Argo | CLASP | Snaplogic |
|---------------------------|-----------|------|-------|-----------|
| Ease of use | TBD | TBD | TBD | TBD |
| Extensibility | TBD | TBD | TBD | TBD |
| Community, Support & Docs | TBD | TBD | TBD | TBD |
| Maturity | TBD | TBD | TBD | TBD |
| Strategic alignment | TBD | TBD | TBD | TBD |
| License | TBD | TBD | TBD | TBD |
| Cost | TBD | TBD | TBD | TBD |

Tab. 2: Evaluation of the suggested tools

The following table shows the evaluation of the tools which where chosen for their relevance to the problem space, based on the criteria and the weighting of the criteria:

| Criteria | Airflow | Kubeflow | Knative | Luigi | CWL |
|---------------------------|---------|----------|---------|-------|-----|
| Ease of use | TBD | TBD | TBD | TBD | TBD |
| Extensibility | TBD | TBD | TBD | TBD | TBD |
| Community, Support & Docs | TBD | TBD | TBD | TBD | TBD |
| Maturity | TBD | TBD | TBD | TBD | TBD |
| Strategic alignment | TBD | TBD | TBD | TBD | TBD |
| License | TBD | TBD | TBD | TBD | TBD |
| Cost | TBD | TBD | TBD | TBD | TBD |

Tab. 3: Evaluation of the additional tools

Conculsion of the Selection Process

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

4.4 Implementation of the Artifact

This section will describe the iterative process of implementing the larger artifact and is broken up into 3 subsections. While these steps where happening concurrently, they each address a different aspect of the project and therefore underwent their own iterative processes.

4.4.1 Infrastructure

First iteration - Minikube

As the decision of the Workflow management tool was made, it was obvious that a dedicated k8s infrastructure was needed to run the tool²⁹. The Pachyderm documentation gave two recommendations for setting up an initial development environment, preferably Docker Desktop or alternatively Minikube³⁰. Due to the exclusive license of Docker-Desktop³¹, which prevents large companies free usage of the product³² the choice fell on Minikube for an initial test setup.

In addition to the underlying k8s Pachyderm also needs an external S3 Storage Bucket for its Pachyderm File System (PFS) for which we used MinIO, a self hostable S3 compliant object storage³³, which was also based on recommendations by the Pachyderm documentation.

The persistent storage requirements for the Pachyderm itself was fulfilled by manually creating two Persistent Volume (PV)’s on the hosts local harddrive. Using the Helm packagemanager³⁴ for k8s the at that point newest version 2.6.4 was installed from the official Artifactory repository³⁵.

²⁹*Pachyderm Docs - On-Prem Deploy* 2023

³⁰*Pachyderm Docs - Local Deploy* 2023

³¹*Docker Terms of Service / Docker* 2022

³²*Docker FAQs / Docker* 2021

³³*Inc* 2023

³⁴*Helm Docs Home* 2023

³⁵*Artifactory Pachyderm 2.6.4* 2023

The hosts system of this iteration was a single ProLiant DL385 Gen10 Plus running Ubuntu 22.04.3 LTS x86_64. During the setup every step was diligently noted and put into a repository³⁶, alongside the needed scripts. The instructions can be found in the appendix at 3.

Learnings from the first iteration

The shortcomings of this naive first iteration became apparent very quickly, which was to be expected, as the goal of this iteration was to create a minimal working example to get a better understanding of the toolings and the underlying infrastructure.

The first and foremost issue where the limitations imposed by Minikubes reliance on an Internal Virtual Machine (VM), during testing the inability to on the fly increase the resources of the VM became a significant bottleneck. At some point during the testing of 4.4.2 the VM was so overloaded that the installation was irreparably damaged which was seen as a sign to move on to the next iteration.

Another more subtle issue was the discrepancy between the experience a small scale k8s installation within Minikube and a large scale k8s cluster like the one that would be used in later steps of the project. Therefore it was decided that a more realistic k8s cluster would be needed for the next iteration, which became the Heydar cluster.

Second iteration - Heydar Cluster

Improving upon the shortcomings of the first iteration, the second iteration was based in the attempt to create a more realistic k8s cluster. To achieve this 20 ProLiant DL360 Gen9 Servers, running Ubuntu 22.04.3 LTS x86_64 were used to create a bare metal k8s cluster, using kubeadm as it provides deep integration with the underlying infrastructure³⁷.

But a bare metal cluster also comes with its own set of challenges, as the cluster needs to be provisioned and configured manually. In order to automate this process, the Ansible automation tool was used to set up all the nodes in parallel and to ensure that all the nodes are in the same state. Ansible is a declarative tool which allows for the automation of the provisioning and configuration of the cluster³⁸, by specifying the desired state of the cluster in a playbook and then applying it to the cluster. The Ansible playbook used for the setup of the cluster can be found at Appendix 4/1.

Which unknowingly caused conflict between the Ansible playbook and the maintenance scripts of the cluster as the Heydar machines. As k8s needs very specific configurations on the underlying infrastructure like the deactivation of swap space³⁹.

³⁶Eckerth 2023

³⁷*Creating a Cluster with Kubeadm* 2023

³⁸*Ansible* 2023

³⁹*Installing Kubeadm* 2023

This was resolved by consulting with the maintainer of the cluster and adjusting the Ansible playbook as well as the maintenance config for the cluster nodes accordingly, after we had identified the issue.

One important aspect of a production like cluster is the networking, as k8s does not natively manage communication on a cluster level, but instead relies on so called Container Network Interface (CNI)s to manage and abstract the underlying network infrastructure⁴⁰.

Here we where spoiled for choice once again, as there are a multitude of different CNIs available, each with their own advantages and disadvantages. The Kubernetes documentation provides a non exhaustive list of 17 different CNIs⁴¹, which all fulfill this essential task in different ways. As the needs regarding the network plugin where not very specific at this point, the choice fell on Calico, as surface level research showed that it was a popular choice for bare metal clusters⁴², provided security and enterprise support as well having a wide range of features⁴³. But Calico proved to be more difficult to setup than expected, after consulting with a college who set up a different cluster with Calico, it was decided to use Flannel as a CNI instead. Flannel turned out to be much easier to setup and configure, as it is a very lightweight CNI which is designed for bare metal clusters⁴⁴, and foregoes the more advanced security features of Calico.

The Flannel configuration used for the cluster can be found at Appendix 4/2 it is closely based on the example configuration provided by the Flannel documentation⁴⁵.

Learnings from the second iteration

The second iteration was a significant improvement over the first iteration, as it provided a much more realistic environment for the development of the artifact. But it also came with its own set of challenges, as the bare metal cluster needed to be provisioned and configured manually, which was a significant time investment.

What became apparent very quickly was that the solution for the provisioning of the PV was nowhere near scalable, as it relies on the local harddrive of the host machine and therefore must host the container on the same machine as the PV which defeats the purpose of a multi node cluster in the first place. Therefore a more scalable solution needs to be implemented for the next iteration. A possible solution could be the use of distributed storage solutions like Ceph⁴⁶ or GlusterFS⁴⁷ in combination with the Rook project⁴⁸. which will need to be explored in future iterations.

⁴⁰ *Cluster Networking* 2023

⁴¹ *Kubernetes CNI Plugins* 2023

⁴² *Explore Network Plugins for Kubernetes* 2023

⁴³ Mehndiratta 2023

⁴⁴ *Flannel* 2023

⁴⁵ *Flannel Install Config* 2023

⁴⁶ *Ceph.Io — Home* 2023

⁴⁷ *Gluster* 2023

⁴⁸ *Rook* 2023

As described in section XXX a service hosting Fabric Attached Memory (FAM) will be needed in future iterations aswell.

4.4.2 Tightly Coupled HPC Workloads

As described in section 3.2.2 TCP problems are a large part of the HPC world, but seem to lack native support in Pachyderm. Pachyderm as it exists as of writing this thesis, is centralized around LCP problems, as it is designed to work with large amounts of data but with each so called "datum" being independent of each other. This is a very good fit for LCP problems, and ties into their concepts of data lineage, versioning and providence.

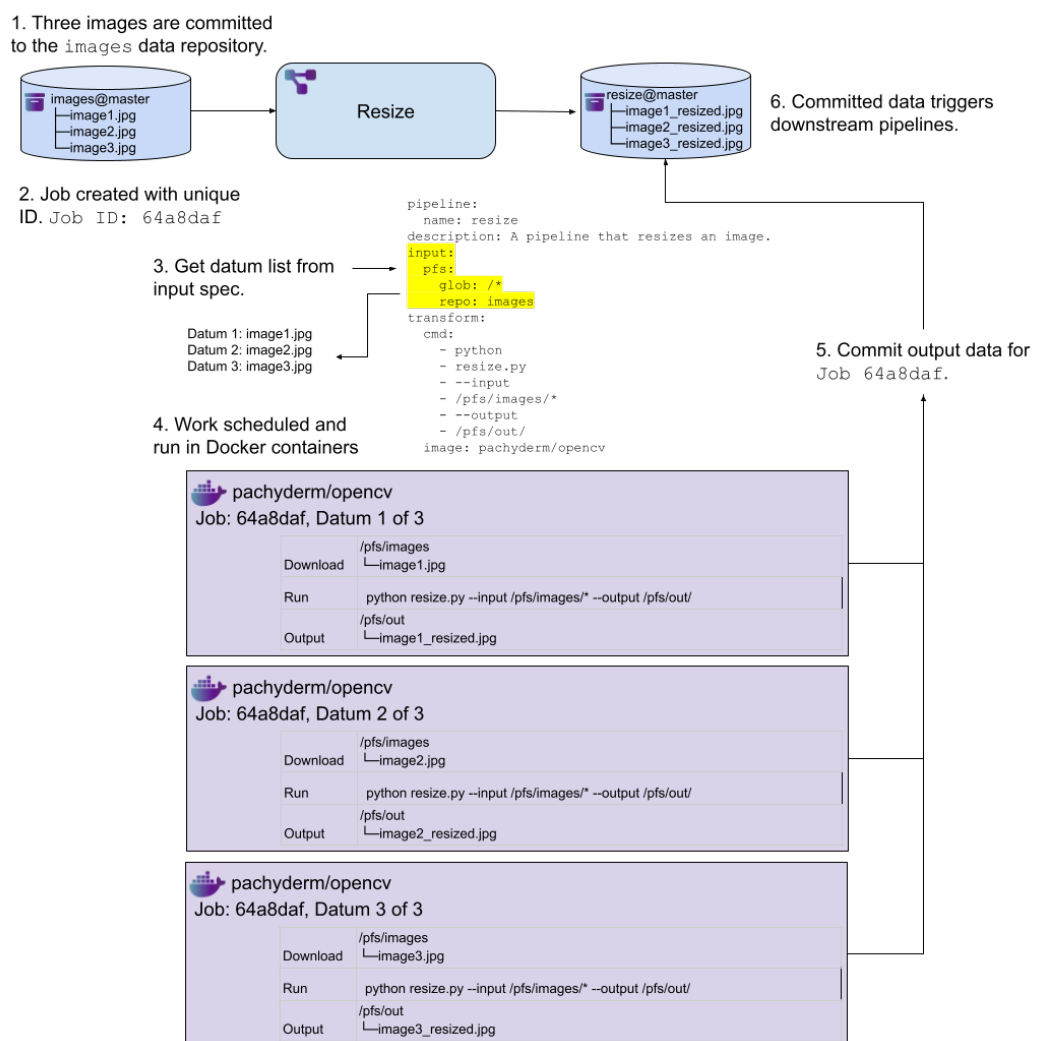


Abb. 7: Pachykouda datum distribution amongst workers ⁴⁹

Diagram 7 shows Pachyderms approach to distribute their datums amongst workers, given an already defined pipeline. Once Data files are added to the input repository, Pachyderm will

⁴⁹Taken from: *Intro to Pipelines* 2023

determine Based on a glob pattern whether the files are relevant datums for the pipeline. If the newly added data fits the pattern each of the files will be supplied to its own instantiation of a worker, all originating from the same image, which will then process the data concurrently and independently of each other. After the worker has finished its task, the resulting datums are then collected in their own repository of data. A more detailed swim lane diagram of this process can be found in the appendix at Appendix 2/1.

This approach is very well suited for LCP problems, as the datums are independent of each other and can be processed in parallel without any issues. But it is not well suited for Large TCP problems, if the computation of the data can not be split into distinct independent datum files, or the computation is reliant on the intercommunication of the datums. If the datasets are small enough, this does not really present a problem as one can simply take all the data into a single workernode and process it there. But as a single worker node can only utilize the resources of a single physical compute node, this does not scale well with the size of the dataset and defeats the purpose of a distributed system in the first place.

So our goal for this section is a way to find a way to enable pachyderm to pool the entire resources of the cluster, in order to solve a TCP problem.

First iteration - PachyKouda

As a first attempt to address this issue, it was decided that the integration of a TCP framework into Pachyderm on the container level would be the best approach. So the first iteration is based on the idea of a Pachyderm conforming client container, which is able to interface with an external TCP framework, which can handle the reception of the data, the distribution of the data amongst the workers and the collection of the results to reintegrate them into the PFS.

The first iteration of this idea was called PachyKouda, as it was based on the Arkouda TCP framework⁵⁰, which itself is a python binding for the Chapel programming language⁵¹.

For that step an Arkouda worker was installed bare metal on the headnode of the Heydar cluster, in order to verify the feasibility of the idea, with the goal of moving the worker into the cluster in the next iteration.

The client container was based on the official User Datagram Protocol (UDP)-based build by the Arkouda team⁵². The container was then modified to be able to communicate with the Arkouda worker on the headnode of the cluster, it can now send data to the worker and receive the results.

⁵⁰ *Arkouda Github Repository* 2023

⁵¹ *Chapel-Lang* 2023

⁵² *Arkouda-Contrib/Arkouda-Docker at Main · Bears-R-Us/Arkouda-Contrib* 2023

Learnings from the first iteration

The first iteration was a total success, as it proved the feasibility of being able to use a client container to forward the data processing to an external Arkouda worker. As described earlier, the goal of the next iteration is to move the Arkouda worker into the cluster, in order to be able to utilize the full resources of the cluster.

Second iteration - Kymera

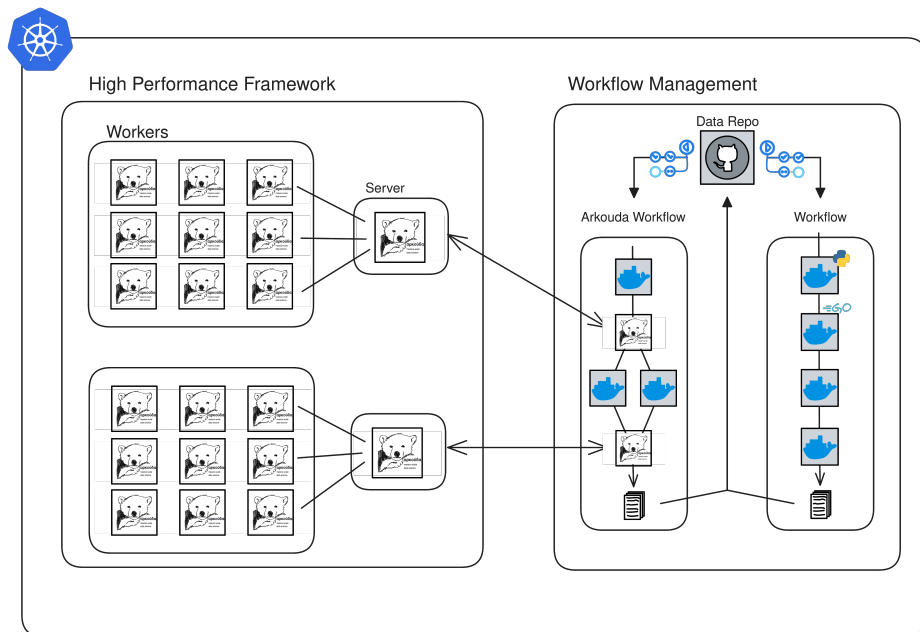


Abb. 8: Arkouda workers on the Heydar cluster

Diagram 8 above shows a high level overview of how the workers interface with the client container in the workflow. The Arkouda container which is part of the workflow is still the same as in the first iteration, but now instead of interfacing with an external worker it is interfacing with a worker swarm hosted across the cluster.

The Swarm is split into two parts, one central Arkouda server, facilitating the communication between the client container and the workers and the workers also called locales themselves. The locales and the server are based on the helm charts provided by the Arkouda-Contrib repo⁵³,

⁵³ *Bears-R-Us/Arkouda-Contrib/Arkouda-Helm-Charts* 2023

A detailed walk through the setup of the Role Based Access Control (RBAC), Secrets and deployments for the Heydar Cluster can be found in the appendix at Appendix 4/4 which in turn is based on the official Arkouda documentation⁵⁴.

Learnings from the second iteration

As Arkouda does not currently provide multi tenancy of their Server, meaning that they can only be connected a single client at a time, so if multiple pipelines need to solve a TCP at the same time, they would not be able to share the same worker swarm.

Third iteration - FAM

4.5 Evaluation of the Artifact

⁵⁴*Arkouda-Contrib/Arkouda-Docker at Main · Bears-R-Us/Arkouda-Contrib* 2023

5 Conclusion

6 Summary and Outlook

6.1 Summary

6.2 Outlook

Appendix

Appendix Index

| | |
|---|----|
| Appendix 1 Discussion of Tool Evaluation and Weighing | 26 |
| Appendix 1/1 Extensibility | 26 |
| Appendix 1/2 Community, Support & Docs | 26 |
| Appendix 1/3 License | 27 |
| Appendix 1/4 Strategic alignment | 28 |
| Appendix 1/5 Ease of Use | 28 |
| Appendix 1/6 Maturity | 28 |
| Appendix 1/7 Cost | 28 |
| Appendix 2 Diagrams | 29 |
| Appendix 2/1 Pipeline Communication Swim Lane Diagram | 29 |
| Appendix 3 Minikube installation instructions | 30 |
| Appendix 4 Kubernetes setup scripts | 38 |
| Appendix 4/1 Ansible setup script | 38 |
| Appendix 4/2 Flannel configuration | 40 |
| Appendix 4/3 Bash verification script | 45 |
| Appendix 4/4 Arkouda Setup | 47 |

Appendix 1: Discussion of Tool Evaluation and Weighing

Appendix 1/1: Extensibility

Appendix 1/2: Community, Support & Docs

This section describes how much external support is given, as this support can come in many different shapes and forms we will be especially focusing on three different aspects and combine them into one score. First, we are looking at the community size, as a large community is a good indicator for the maturity of the project and the amount of support available. As a proxy for this, we will look at two central metrics, the amount of stars on Github and the amount of questions on Stackoverflow.

Tab. 4: Comparison of project popularity

| Project | GitHub Stars | Stack Overflow Questions |
|-----------|--------------|--------------------------|
| Pachyderm | 6,000 | 6 |
| Argo | 14,500 | 136 |
| Clasp | 0 | 0 |
| Snaplogic | 0 | 57 |
| Airflow | 32,200 | 10,218 |
| Kubeflow | 13,100 | 434 |
| Knative | 4,100 | 204 |
| Luigi | 16,900 | 346 |
| CWL | 1,400 | 6 |

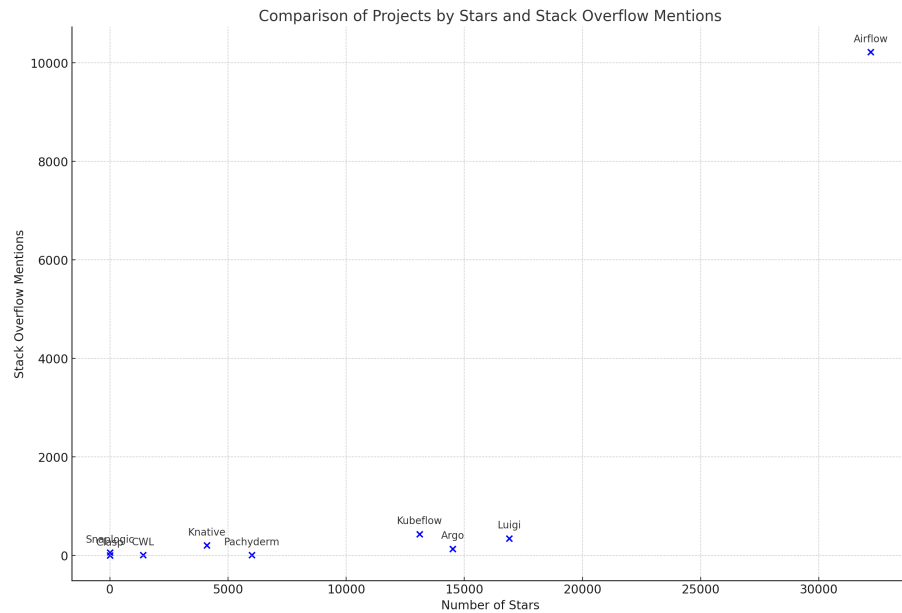


Abb. 9: Stars and Stackoverflow Questions Comparison

To quantify the support and community engagement around these projects, we introduce a

composite score that normalizes and combines the GitHub stars and Stack Overflow questions metrics. This score is calculated using the following methodology:

Each project is represented as a point $P_i = (x_i, y_i)$ in a two-dimensional space, with x_i and y_i being the number of GitHub stars and Stack Overflow questions, respectively, for the i -th project. The composite score S_i for each project is computed by normalizing these values to a 0-10 scale and then averaging them:

$$S_i = \frac{1}{2} \left(\frac{x_i - \min(x)}{\max(x) - \min(x)} \times 10 + \frac{y_i - \min(y)}{\max(y) - \min(y)} \times 10 \right) \quad \text{for } i = 1, 2, \dots, n$$

Here, $\min(x)$, $\max(x)$, $\min(y)$, and $\max(y)$ represent the minimum and maximum values of GitHub stars and Stack Overflow questions across all projects, respectively. The final scores S_i provide a balanced metric for comparing the projects' popularity and community engagement, taking both aspects into account on the same scale.

Appendix 1/3: License

As discussed in section 4.3 the tools in consideration should not be too restrictive. To evaluate the criteria we will employ a 4 bucket system, where tools with a permissive license get a value of 7.5, tools where the licensing is ideal get a 10, and tools which have a not acceptable license get weighted with 0.

- **Pachyderm** The licensing model of Pachyderm follows a model which has similarities with the "Open Core model"⁵⁵. Which means that while the core functionalities are published as the "COMMUNITY EDITION" with a permissive source-available License (Apache License 2.0)⁵⁶. Functionality like Single Sign On (SSO) or the ability to create more than 16 pipelines are part of a different distribution under a Commercial License.

But in our case this is of no concern, as the startup behind the Pachyderm software, including its Intellectual Property (IP) was acquired by HPE. Giving us a free hand to modify without needing to worry.

- **Argo** is licensed under the Apache License 2.0.⁵⁷ Apache 2.0 is quite a standard license permissive license for this kind of product, as it enables [TO DO]
- **CLASP** is not a published software and therefore not under any specific license. But similar considerations as the ones of Pachyderm apply here as well, as it is an internal project the IP also completely belongs to HPE

⁵⁵ *Pachyderm -Pricing 2022*

⁵⁶ *Pachyderm/LICENSE at Master · Pachyderm/Pachyderm 2023*

⁵⁷ *Argo-Cd/LICENSE at Master · ArgoProj/Argo-Cd 2023*

- **Snaplogic** is an entirely commercial product which does not provide insight into nor the right to modify their Software⁵⁸. But as they might agree this is not a total knockout criterion for this entire project, but in regards to the licensing it will be weighted with 0.
- **Airflow** is licensed under the Apache License 2.0.⁵⁹
- **Kubeflow** is licensed under the Apache License 2.0.⁶⁰
- **Knative** is licensed under the Apache License 2.0.⁶¹
- **Luigi** is licensed under the Apache License 2.0.⁶²
- **CWL** is licensed under the Apache License 2.0.⁶³

Appendix 1/4: Strategic alignment

Appendix 1/5: Ease of Use

Appendix 1/6: Maturity

Appendix 1/7: Cost

| Criteria | Pachyderm | Argo | CLASP | Snaplogic |
|---------------------------|-----------|------|-------|-----------|
| Ease of use | TBD | TBD | TBD | TBD |
| Extensibility | TBD | TBD | TBD | TBD |
| Community, Support & Docs | TBD | TBD | TBD | TBD |
| Maturity | TBD | TBD | TBD | TBD |
| Strategic alignment | TBD | TBD | TBD | TBD |
| License | 10 | 5 | 10 | 0 |
| Cost | TBD | TBD | TBD | TBD |

Tab. 5: Evaluation of the suggested tools

⁵⁸ *SnapLogic – Master Subscription Agreement* 2023

⁵⁹ *License — Airflow Documentation* 2023

⁶⁰ *Kubeflow/LICENSE at Master · Kubeflow/Kubeflow* 2023

⁶¹ *Knative Docs/LICENSE at Main · Knative/Docs* 2023

⁶² *Luigi/LICENSE at Master · Spotify/Luigi* 2023

⁶³ *Cwl-Utils/LICENSE at Main · Common-Workflow-Language/Cwl-Utils* 2023

| Criteria | Airflow | Kubeflow | Knative | Luigi | CWL |
|---------------------------|---------|----------|---------|-------|-----|
| Ease of use | TBD | TBD | TBD | TBD | TBD |
| Extensibility | TBD | TBD | TBD | TBD | TBD |
| Community, Support & Docs | TBD | TBD | TBD | TBD | TBD |
| Maturity | TBD | TBD | TBD | TBD | TBD |
| Strategic alignment | TBD | TBD | TBD | TBD | TBD |
| License | 5 | 5 | 5 | 5 | 5 |
| Cost | TBD | TBD | TBD | TBD | TBD |

Tab. 6: Evaluation of the additional tools

Appendix 2: Diagrams

Appendix 2/1: Pipeline Communication Swim Lane Diagram

⁶³Taken from: *Intro to Pipelines* 2023

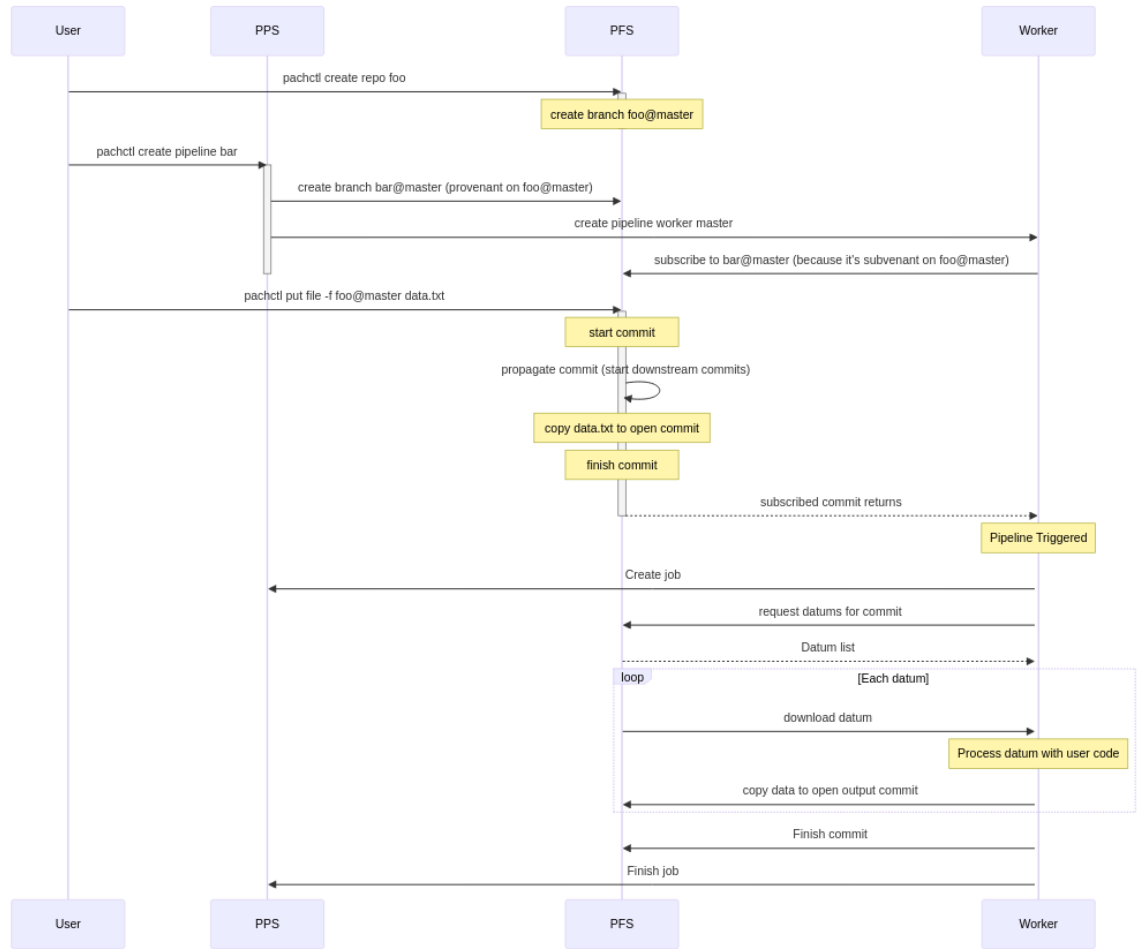


Abb. 10: Swimlane Diagram of the communication between the user and Pachyderm

Appendix 3: Minikube installation instructions

```

1 # Pachyderm
2
3 ## Installation
4
5 These instructions are based upon the excellent guide by
6   ↪ [Pachyderm](https://docs.pachyderm.com/latest/set-up/on-prem/)
7
8
9 ### Proxy
10
11 If you are in the HPE internal network, you will need to set up the proxy.
12 Simply execute the following command:
13
14 ```bash
15 export HTTP_PROXY=http://web-proxy.corp.hpecorp.net:8080
16 export HTTPS_PROXY=http://web-proxy.corp.hpecorp.net:8080
17 ```
18
19 If you want to make this permanent, add these lines to the '~/.bashrc' or
  
```

```

    ↪ equivalent file.
18
19 ### kubectl
20
21 Simply following the instructions on the [kubernetes
    ↪ website](https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/)
    ↪ should be sufficient.
22 But for the sake of completeness, here is what I did:
23
24 ```bash
25 curl -LO "https://dl.k8s.io/release/$(curl -L -s
    ↪ https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
26 sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
27 ```
28
29 If the proxy is giving you grief one can simply download the binary elsewhere
    ↪ and copy it to the target machine. (not recommended)
30
31 ### Installing minikube
32
33 The same things apply for minikube as for kubectl.
34 The proper instructions can be found on the [minikube
    ↪ website](https://minikube.sigs.k8s.io/docs/start/)
35 But here is what I did anyway:
36
37 ```bash
38 curl -LO
    ↪ https://storage.googleapis.com/minikube/releases/latest/minikube_latest_amd64.deb
39 sudo dpkg -i minikube_latest_amd64.deb
40 ```
41
42 We can then test the installation by running:
43
44 ```bash
45 minikube start
46 kubectl cluster-info
47 ```
48
49 If you are getting an error stating that it is not able to connect to the
    ↪ cluster you might need to set the following environment variable:
50
51 ```bash
52 export
    ↪ NO_PROXY=localhost,127.0.0.1,10.96.0.0/12,192.168.59.0/24,192.168.49.0/24,192.168.
53 ```
54
55 ### Installing [helm](https://helm.sh/docs/intro/install/)
56
57 Same procedure as every year...
58
59 ```bash

```

```

60 curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee
    ↪ /usr/share/keyrings/helm.gpg > /dev/null
61 sudo apt-get install apt-transport-https --yes
62 echo "deb [arch=$(dpkg --print-architecture)
    ↪ signed-by=/usr/share/keyrings/helm.gpg]
    ↪ https://baltocdn.com/helm/stable/debian/ all main" | sudo tee
    ↪ /etc/apt/sources.list.d/helm-stable-debian.list
63 sudo apt-get update
64 sudo apt-get install helm
65 '''
66
67 ### [Persistent
    ↪ Storage](https://kubernetes.io/docs/tasks/configure-pod-container/configure-persis
68
69 We need to create a persistent volume for etcd and the postgres database.
70 Therefore we need to create a directory for each of them.
71
72 '''bash
73 mkdir -p /mnt/pachyderm/etcd
74 mkdir -p /mnt/pachyderm/postgres
75 '''
76
77 We then create the configuration files for the persistent volumes.
78
79 '''yaml
80 apiVersion: v1
81 kind: PersistentVolume
82 metadata:
83   name: etcd-pv
84 labels:
85   type: local
86 spec:
87   capacity:
88     storage: 10Gi
89   accessModes:
90     - ReadWriteOnce
91   storageClassName: manual
92   local:
93     path: /mnt/pachyderm/etcd
94
95 ---
96
97 apiVersion: v1
98 kind: PersistentVolume
99 metadata:
100   name: postgres-pv
101 labels:
102   type: local
103 spec:
104   capacity:
105     storage: 10Gi

```



```
106     accessModes:
107         - ReadWriteOnce
108     storageClassName: manual
109     local:
110         path: /mnt/pachyderm/postgres
111 '''
112
113 And then the corresponding persistent volume claims.
114
115 '''yaml
116 apiVersion: v1
117 kind: PersistentVolumeClaim
118 metadata:
119     name: etcd-pvc
120 spec:
121     storageClassName: manual
122     accessModes:
123         - ReadWriteOnce
124     resources:
125         requests:
126             storage: 10Gi
127
128 ---
129
130 apiVersion: v1
131 kind: PersistentVolumeClaim
132 metadata:
133     name: postgres-pvc
134 spec:
135     storageClassName: manual
136     accessModes:
137         - ReadWriteOnce
138     resources:
139         requests:
140             storage: 10Gi
141 '''
142
143 Then we add the storage class to the cluster.
144
145 '''bash
146 kubectl apply -f filename.yaml
147 '''
148
149 We then take note of the storage class name because we will add it to the helm
150 ↪ values file later. \
151 In this case it is 'manual'.
152
153 ### Installing [MinIO](https://min.io/docs/minio/linux/index.html)
154
155 We now install an S3 compatible storage system. Which one does not really
156 ↪ matter, but I chose MinIO because it is easy to install and configure.
```

```

155
156 ```bash
157 wget
    ↪ https://dl.min.io/server/minio/release/linux-amd64/archive/minio_20230619195250.0.
    ↪ -O minio.deb
158 sudo dpkg -i minio.deb
159
160 mkdir -p /mnt/pachyderm/minio
161
162 # to manually start the server
163 minio server /mnt/pachyderm/minio --console-address :9001
164 ```
165
166 The standard password is 'minioadmin:minioadmin'
167
168 Then you can access the web interface at 'http://localhost:9001' where you
    ↪ should login, change the password and create a bucket. \
169 The access credentials for the bucket will be added to the helm values file
    ↪ later, so take note of them.
170
171 ### Installing [Pachyderm](https://docs.pachyderm.com/latest/set-up/on-prem/)
172
173 First we need to add the Pachyderm helm repository:
174
175 ```bash
176 helm repo add pachyderm https://helm.pachyderm.com
177 helm repo update
178 ```
179
180 We then get the values file from the repository and edit it to our liking.\
181 My setup is based on the version '2.6.4-1', so it might be different for future
    ↪ versions.
182
183 ```bash
184 wget
    ↪ https://raw.githubusercontent.com/pachyderm/pachyderm/2.6.x/etc/helm/pachyderm/val
185 ```
186
187 #### MinIO
188
189 First we change the deploy target at line 'L7'
190
191 ```yaml
192 # Deploy Target configures the storage backend to use and cloud provider
193 # settings (storage classes, etc). It must be one of GOOGLE, AMAZON,
194 # MINIO, MICROSOFT, CUSTOM or LOCAL.
195 deployTarget: "MINIO"
196 ...
197 ```
198
199 This does not need to be set when using something else but with MinIO we also

```

```
    ↪ have to set 'L544' to "MINIO"
200
201 ```yaml
202 ...
203 storage:
204     # backend configures the storage backend to use. It must be one
205     # of GOOGLE, AMAZON, MINIO, MICROSOFT or LOCAL. This is set automatically
206     # if deployTarget is GOOGLE, AMAZON, MICROSOFT, or LOCAL
207     backend: "MINIO"
208     ...
209 ```
210
211 A little further down ('L635') we find the MinIO configuration. We need to set
    ↪ the endpoint, access key and secret key.
212
213 This point was a little tricky as I had MinIO installed on the same machine as
    ↪ Pachyderm, but it would take no other value than the outward facing IP
    ↪ address of the machine.
214
215 ```yaml
216 ...
217     minio:
218         # minio bucket name
219         bucket: "<bucket name>"
220         # the minio endpoint. Should only be the hostname:port, no http/https.
221         endpoint: "10.X.X.X:9000"
222         # the username/id with readwrite access to the bucket.
223         id: "<id>"
224         # the secret/password of the user with readwrite access to the bucket.
225         secret: "<secret>"
226         # enable https for minio with "true" defaults to "false"
227         secure: "false"
228         # Enable S3v2 support by setting signature to "1". This feature is being
            ↪ deprecated
229         signature: ""
230     ...
231 ```
232
233 #### Storage classes
234
235 Now we add the storage classes we created earlier to the Postgres at 'L784'
236
237 ```yaml
238 ...
239     # AWS: https://docs.aws.amazon.com/eks/latest/userguide/storage-classes.html
240     # GCP: https://cloud.google.com/compute/docs/disks/performance#disk_types
241     # Azure:
        ↪ https://docs.microsoft.com/en-us/azure/aks/concepts-storage#storage-classes
242     storageClass: manual
243     # storageSize specifies the size of the volume to use for postgresql
244     # Recommended Minimum Disk size for Microsoft/Azure: 256Gi - 1,100 IOPS
```

```

    ↪ https://azure.microsoft.com/en-us/pricing/details/managed-disks/
245 ...
246 '''
247
248 and for the etcd at around 'L144'
249
250 '''yaml
251 ...
252
253 # GCP: https://cloud.google.com/compute/docs/disks/performance#disk_types
254 # Azure:
    ↪ https://docs.microsoft.com/en-us/azure/aks/concepts-storage#storage-classes
255 #storageClass: manual
256 storageClassName: manual
257
258 # storageSize specifies the size of the volume to use for etcd.
259 # Recommended Minimum Disk size for Microsoft/Azure: 256Gi - 1,100 IOPS
    ↪ https://azure.microsoft.com/en-us/pricing/details/managed-disks/
260
261 ...
262 '''
263
264 #### SSL Certificates
265
266 My setup refuses to work without SSL certificates, so I had to generate some.
267
268 '''bash
269 openssl genrsa -out <CertName>.key 2048
270 openssl req -new -x509 -sha256 -key <CertName>.key -out <CertName>.crt
271
272 kubectl create secret tls <SecretName> --cert=<CertName>.crt
    ↪ --key=<CertName>.key
273 '''
274
275 We then edit the 'values.yaml' file at around 'L683' to use the certificates.
276
277 '''yaml
278 ...
279   tls:
280     enabled: true
281     secretName: "<SecretName>"
282     newSecret:
283       create: false
284     ...
285 '''
286
287 ### CLI
288
289 To directly interact with the cluster we need to install the Pachyderm CLI.
290
291 '''bash
```

```
292 curl -o /tmp/pachctl.deb -L
    ↪ https://github.com/pachyderm/pachyderm/releases/download/v2.6.5/pachctl_2.6.5_amd64.deb
    ↪ && sudo dpkg -i /tmp/pachctl.deb
293 ```
294
295 ### Deploy
296
297 Now that the values file is ready we can install Pachyderm.
298
299 ```bash
300 helm install pachyderm pachyderm/pachyderm \
301   -f ./values.yml pachyderm/pachyderm \
302   --set postgresql.volumePermissions.enabled=true \
303   --set deployTarget=LOCAL \
304   --set proxy.enabled=true \
305   --set proxy.service.type=NodePort \
306   --set proxy.host=localhost \
307   --set proxy.service.httpPort=8080
308
309 ```
310
311 Now you might want to connect to the dashboard. This can be done by
    ↪ port-forwarding the service.
312
313 ```bash
314 pachctl port-forward
315 ```
316
317 :tada: Now we should be able to access the dashboard at 'http://localhost:4000'
    ↪ :tada:
```

Appendix 4: Kubernetes setup scripts

Appendix 4/1: Ansible setup script

```

1 ---
2 - hosts: heydar_nodes
3   become: yes
4   tasks:
5     - name: Setting up environment variables
6       lineinfile:
7         path: /etc/environment
8         line: "{{ item }}"
9       with_items:
10        - "https_proxy=http://proxy.its.hpecorp.net:80"
11        - "HTTP_PROXY=http://proxy.its.hpecorp.net:80"
12        - "http_proxy=http://proxy.its.hpecorp.net:80"
13        -
14          ↪ "NO_PROXY=localhost,127.0.0.1,10.0.0.0/8,172.16.0.0/16,10.93.246.68/28"
15
16    - name: Update and install necessary packages
17      apt:
18        name: "{{ packages }}"
19        update_cache: yes
20      vars:
21        packages:
22          - apt-transport-https
23          - ca-certificates
24          - curl
25
26    - name: Add Kubernetes apt-key
27      shell: |
28        curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | gpg
29          ↪ --yes --dearmor -o
30          ↪ /etc/apt/keyrings/kubernetes-archive-keyring.gpg
31        echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg]
32          ↪ https://apt.kubernetes.io/ kubernetes-xenial main" | tee
33          ↪ /etc/apt/sources.list.d/kubernetes.list
34        apt-get update -y
35        apt-get install -y kubelet kubeadm kubectl containerd
36        apt-mark hold kubelet kubeadm kubectl
37
38    - name: Enable necessary kernel modules and sysctl parameters
39      shell: |
40        modprobe br_netfilter
41        echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
42        echo '1' > /proc/sys/net/ipv4/ip_forward
43        sysctl -p
44
45    - name: Disable swap
46      shell: |

```

```
42      swapoff -a
43      sed -i -e '/ swap / s/^/#/' -e '/\swap.img/ s/^/#/' /etc/fstab #
      ↪ comments out swap in fstab
44
45      - name: Join the Kubernetes cluster
46      shell: |
47      kubeadm join 10.93.246.87:6443 --token 0v7aoq.65ib2v0g70a6em49
      ↪ --discovery-token-ca-cert-hash
      ↪ sha256:9cb5e62dd86cd7e94718c866575cd023c98cc89f2849dad3d25dfd75b13d1b72
```

Appendix 4/2: Flannel configuration

```
1 apiVersion: v1
2 kind: Namespace
3 metadata:
4   labels:
5     k8s-app: flannel
6     pod-security.kubernetes.io/enforce: privileged
7   name: kube-flannel
8 ---
9 apiVersion: v1
10 kind: ServiceAccount
11 metadata:
12   labels:
13     k8s-app: flannel
14   name: flannel
15   namespace: kube-flannel
16 ---
17 apiVersion: rbac.authorization.k8s.io/v1
18 kind: ClusterRole
19 metadata:
20   labels:
21     k8s-app: flannel
22   name: flannel
23 rules:
24 - apiGroups:
25   - ""
26   resources:
27     - pods
28   verbs:
29     - get
30 - apiGroups:
31   - ""
32   resources:
33     - nodes
34   verbs:
35     - get
36     - list
37     - watch
38 - apiGroups:
39   - ""
40   resources:
41     - nodes/status
42   verbs:
43     - patch
44 - apiGroups:
45   - networking.k8s.io
46   resources:
47     - clustercidrs
48   verbs:
49     - list
```



```
50   - watch
51 ---
52 apiVersion: rbac.authorization.k8s.io/v1
53 kind: ClusterRoleBinding
54 metadata:
55   labels:
56     k8s-app: flannel
57   name: flannel
58 roleRef:
59   apiGroup: rbac.authorization.k8s.io
60   kind: ClusterRole
61   name: flannel
62 subjects:
63 - kind: ServiceAccount
64   name: flannel
65   namespace: kube-flannel
66 ---
67 apiVersion: v1
68 data:
69   cni-conf.json: |
70     {
71       "name": "cbr0",
72       "cniVersion": "0.3.1",
73       "plugins": [
74         {
75           "type": "flannel",
76           "delegate": {
77             "hairpinMode": true,
78             "isDefaultGateway": true
79           }
80         },
81         {
82           "type": "portmap",
83           "capabilities": {
84             "portMappings": true
85           }
86         }
87       ]
88     }
89   net-conf.json: |
90     {
91       "Network": "172.16.0.0/16",
92       "Backend": {
93         "Type": "vxlan"
94       }
95     }
96 kind: ConfigMap
97 metadata:
98   labels:
99     app: flannel
100    k8s-app: flannel
```

```
101     tier: node
102   name: kube-flannel-cfg
103   namespace: kube-flannel
104 ---
105 apiVersion: apps/v1
106 kind: DaemonSet
107 metadata:
108   labels:
109     app: flannel
110     k8s-app: flannel
111     tier: node
112   name: kube-flannel-ds
113   namespace: kube-flannel
114 spec:
115   selector:
116     matchLabels:
117       app: flannel
118       k8s-app: flannel
119   template:
120     metadata:
121       labels:
122         app: flannel
123         k8s-app: flannel
124         tier: node
125     spec:
126       affinity:
127         nodeAffinity:
128           requiredDuringSchedulingIgnoredDuringExecution:
129             nodeSelectorTerms:
130               - matchExpressions:
131                 - key: kubernetes.io/os
132                   operator: In
133                   values:
134                     - linux
135       containers:
136       - args:
137         - --ip-masq
138         - --kube-subnet-mgr
139         command:
140         - /opt/bin/flanneld
141         env:
142         - name: POD_NAME
143           valueFrom:
144             fieldRef:
145               fieldPath: metadata.name
146         - name: POD_NAMESPACE
147           valueFrom:
148             fieldRef:
149               fieldPath: metadata.namespace
150         - name: EVENT_QUEUE_DEPTH
151           value: "5000"
```

```
152     image: docker.io/flannel/flannel:v0.22.0
153     name: kube-flannel
154     resources:
155         requests:
156             cpu: 100m
157             memory: 50Mi
158     securityContext:
159         capabilities:
160             add:
161                 - NET_ADMIN
162                 - NET_RAW
163             privileged: false
164     volumeMounts:
165         - mountPath: /run/flannel
166           name: run
167         - mountPath: /etc/kube-flannel/
168           name: flannel-cfg
169         - mountPath: /run/xtables.lock
170           name: xtables-lock
171     hostNetwork: true
172     initContainers:
173     - args:
174         - -f
175         - /flannel
176         - /opt/cni/bin/flannel
177       command:
178         - cp
179       image: docker.io/flannel/flannel-cni-plugin:v1.1.2
180       name: install-cni-plugin
181       volumeMounts:
182         - mountPath: /opt/cni/bin
183           name: cni-plugin
184     - args:
185         - -f
186         - /etc/kube-flannel/cni-conf.json
187         - /etc/cni/net.d/10-flannel.conflist
188       command:
189         - cp
190       image: docker.io/flannel/flannel:v0.22.0
191       name: install-cni
192       volumeMounts:
193         - mountPath: /etc/cni/net.d
194           name: cni
195         - mountPath: /etc/kube-flannel/
196           name: flannel-cfg
197     priorityClassName: system-node-critical
198     serviceAccountName: flannel
199     tolerations:
200     - effect: NoSchedule
201       operator: Exists
202     volumes:
```

```
203     - hostPath:
204         path: /run/flannel
205         name: run
206     - hostPath:
207         path: /opt/cni/bin
208         name: cni-plugin
209     - hostPath:
210         path: /etc/cni/net.d
211         name: cni
212     - configMap:
213         name: kube-flannel-cfg
214         name: flannel-cfg
215     - hostPath:
216         path: /run/xtables.lock
217         type: FileOrCreate
218         name: xtables-lock
```

Appendix 4/3: Bash verification script

```
1 #!/bin/bash
2
3 # Define color codes
4 RED='\033[0;31m'
5 GREEN='\033[0;32m'
6 NC='\033[0m' # No Color
7
8 # Initialize error flag
9 error_flag=0
10
11 # Function to print info messages
12 info() {
13     echo -e "${GREEN}[INFO] $1${NC}"
14 }
15
16 # Function to print error messages
17 fail() {
18     echo -e "${RED}[ERROR] $1${NC}"
19     error_flag=1
20 }
21
22 # Checking installation of necessary packages
23 dpkg -l | grep -qw apt-transport-https || fail "apt-transport-https is not
    ↪ installed"
24 dpkg -l | grep -qw ca-certificates || fail "ca-certificates is not installed"
25 dpkg -l | grep -qw curl || fail "curl is not installed"
26 dpkg -l | grep -qw kubelet || fail "kubelet is not installed"
27 dpkg -l | grep -qw kubeadm || fail "kubeadm is not installed"
28 dpkg -l | grep -qw kubect1 || fail "kubect1 is not installed"
29 dpkg -l | grep -qw containerd || fail "containerd is not installed"
30
31 # Check Kubernetes APT source list
32 grep -q "https://apt.kubernetes.io/ kubernetes-xenial main"
    ↪ /etc/apt/sources.list.d/kubernetes.list || fail "Kubernetes APT source
    ↪ list is not configured correctly"
33
34 # Check if swap is disabled
35 swapon --summary | grep -q swap && fail "Swap is not disabled"
36
37 # Check containerd configuration
38 grep -q 'SystemdCgroup = true' /etc/containerd/config.toml || fail
    ↪ "SystemdCgroup is not enabled in containerd configuration"
39
40 # Check sysctl parameters
41 [ "$(cat /proc/sys/net/bridge/bridge-nf-call-iptables)" == "1" ] || fail
    ↪ "bridge-nf-call-iptables is not enabled"
42 [ "$(cat /proc/sys/net/ipv4/ip_forward)" == "1" ] || fail "ip_forward is not
    ↪ enabled"
43
```

```
44 # Check proxy settings for services
45 [ -f /etc/systemd/system/containerd.service.d/http-proxy.conf ] || fail "Proxy
    ↪ settings for containerd service is not configured"
46 [ -f /etc/systemd/system/kubelet.service.d/http-proxy.conf ] || fail "Proxy
    ↪ settings for kubelet service is not configured"
47
48 # Check Kubernetes node status
49 if command -v kubectl &> /dev/null; then
50     kubectl get nodes || fail "Failed to get Kubernetes nodes. Check if the
        ↪ node has joined the cluster successfully"
51 else
52     info "kubectl command not found. Skipping Kubernetes node check"
53 fi
54
55 # Check status of services
56 if systemctl --all --type=service --state=active | grep -qw containerd; then
57     systemctl is-active --quiet containerd || fail "containerd service is not
        ↪ running"
58 else
59     info "containerd service not found. Skipping service status check"
60 fi
61
62 if systemctl --all --type=service --state=active | grep -qw kubelet; then
63     systemctl is-active --quiet kubelet || fail "kubelet service is not running"
64 else
65     info "kubelet service not found. Skipping service status check"
66 fi
67
68 # Print summary
69 if [ $error_flag -eq 0 ]; then
70     info "All checks passed successfully."
71 else
72     echo -e "${RED}Some checks failed. Please check the error messages
        ↪ above.${NC}"
73 fi
```

Appendix 4/4: Arkouda Setup

```
1
2 # Arkouda
3
4 Based on the helm charts in the [Arkouda Contrib
    ↪ repository](https://github.com/Bears-R-Us/arkouda-contrib/tree/main/arkouda-helm-c
5 we can now start to deploy Arkouda in our kubernetes Kluster.
6 These installation instructions are based on the readme of the same repo.
7
8 ```bash
9 git clone git@github.com:Bears-R-Us/arkouda-contrib.git
10 ```
11
12 ## Namespace
13
14 For this we create its own namespace.
15
16 ```bash
17 kubectl create namespace arkouda
18 ```
19
20 If you want to make your live a little bit easier and work with many differnt
    ↪ namespaces, you can add the following alias to your '.bashrc' or
    ↪ '.zshrc' file.
21
22 ```bash
23 alias kark='kubectl --namespace arkouda'
24 ```
25
26 This keeps you from having to type '--namespace arkouda' or '-n arkouda' every
    ↪ time you want to interact with the arkouda namespace.
27
28 ## Secrets
29
30 To get the containers to talk to each other and to interface with the
    ↪ kubernetes api we need to create some secrets.
31
32 ### SSH
33
34 The first secret we create is the ssh secret. This is used to connect to the
    ↪ pods and to the kubernetes api. \
35 As requested by the
    ↪ [dokumentation](https://github.com/Bears-R-Us/arkouda-contrib/tree/3e4050bfef2bf2a
    ↪ this ssh key needs to be created while impersonating a user with the
    ↪ 'ubuntu' username.
36
37 ```bash
38 adduser ubuntu --disabled-password --gecos ""
39 su ubuntu -c "ssh-keygen -t rsa -b 4096 -C \"ubuntu@arkouda\" -f ~/id_rsa -q -N
    ↪ \"\""
```

```
40
41 # then we create the secret
42 kark create secret generic arkouda-ssh --from-file=id_rsa=./id_rsa
    ↪ --from-file=id_rsa.pub=./id_rsa.pub
43 '''
44
45 ### SSL
46
47 The second secret we need is a ssl secret. This is used to connect to the
    ↪ Kubernetes API. \
48 This secret is created by generating a self signed certificate.
49
50 '''bash
51
52 # we start by generating the certificate
53 # note do not change the name of the certificate, as it is hardcoded in the
    ↪ yaml file
54 openssl genrsa -out tls.key 2048
55
56 # creating the certificate signing request
57 openssl req -new -key tls.key -out tls.csr -subj "/CN=arkouda/O=group1"
58
59
60 # now we create a CSR object in the kubernetes api
61
62 cat <<EOF | kark apply -f -
63 apiVersion: certificates.k8s.io/v1
64 kind: CertificateSigningRequest
65 metadata:
66   name: arkouda
67 spec:
68   request: $(cat tls.csr | base64 | tr -d '\n')
69   signerName: kubernetes.io/kube-apiserver-client
70   usages:
71     - digital signature
72     - key encipherment
73     - client auth
74 EOF
75
76 # and get it approved by an admin
77 kark certificate approve arkouda
78
79
80 # from this we get the certificate
81 kark get csr arkouda -o jsonpath='{.status.certificate}' | base64 --decode >
    ↪ tls.crt
82
83 # now we can verify whether the certificate is valid (this is specific to
    ↪ minikube)
84 curl --cacert /home/<your username>/.minikube/ca.crt --cert ./tls.crt --key
    ↪ ./tls.key https://$(minikube ip):8443/api/
```



```

85
86
87 # and create the secret
88 kark create secret generic arkouda-tls --from-file=tls.crt=./tls.crt
    ↪ --from-file=tls.key=./tls.key
89 ```
90
91 ### Cluster Role
92
93 The following section is an excerpt of the [Arkouda UDP Server
    ↪ documentation](https://github.com/Bears-R-Us/arkouda-contrib/tree/3e4050bfef2bf2a2
94
95 ## ClusterRoles
96
97 The Kubernetes API permissions are in the form of a ClusterRole (scoped to all
    ↪ namespaces). For the purposes of this demonstration, the ClusterRoles
    ↪ are as follows. Corresponding Role definitions only differ in that that
    ↪ the Kind field is Role and metadata has a namespace element.
98
99 ### GASNET udp Integration
100
101 The arkouda-udp-server deployment discovers all arkouda-udp-locale pods on
    ↪ startup to create the GASNET udp connections between all Arkouda
    ↪ locales. Accordingly, Arkouda requires Kubernetes pod list and get
    ↪ permissions. The corresponding ClusterRole is as follows:
102
103 ```yaml
104 apiVersion: rbac.authorization.k8s.io/v1
105 kind: ClusterRole
106 metadata:
107   name: arkouda-pod-reader
108 rules:
109 - apiGroups: [""]
110   resources: ["pods"]
111   verbs: ["get", "watch", "list"]
112 ```
113
114 This ClusterRole is bound to the arkouda Kubernetes user as follows:
115
116 ```yaml
117 kind: ClusterRoleBinding
118 apiVersion: rbac.authorization.k8s.io/v1
119 metadata:
120   name: arkouda-pod-reader-binding
121 subjects:
122 - kind: User
123   name: arkouda
124   apiGroup: rbac.authorization.k8s.io
125 roleRef:
126   kind: ClusterRole
127   name: pod-reader

```

```
128   apiGroup: rbac.authorization.k8s.io
129   '''
130
131   ### Service Integration
132
133   Arkouda-on-Kubernetes integrates with Kubernetes service discovery by creating
134       ↪ a Kubernetes service upon arkouda-udp-server startup and deleting the
135       ↪ Kubernetes service upon teardown. Consequently, Arkouda-on-Kubernetes
136       ↪ requires full Kubernetes service CRUD permissions to enable service
137       ↪ discovery. The corresponding ClusterRole is as follows:
138
139   '''yaml
140   apiVersion: rbac.authorization.k8s.io/v1
141   kind: ClusterRole
142   metadata:
143     name: service-endpoints-crud
144   rules:
145   - apiGroups: [""]
146     resources: ["services","endpoints"]
147     verbs: ["get","watch","list","create","delete","update"]
148   '''
149
150   This ClusterRole is bound to the arkouda Kubernetes user as follows:
151
152   '''yaml
153   kind: ClusterRoleBinding
154   apiVersion: rbac.authorization.k8s.io/v1
155   metadata:
156     name: arkouda-service-endpoints-crud
157   subjects:
158   - kind: User
159     name: arkouda
160     apiGroup: rbac.authorization.k8s.io
161   roleRef:
162     kind: ClusterRole
163     name: service-endpoints-crud
164     apiGroup: rbac.authorization.k8s.io
165   '''
166
167   ## Locale-Pods
168
169   Now we can edit the 'arkouda-udp-locale.yaml' file to match our needs. \
170   For reference, the following is the configuration on my test setup.
171
172   '''yaml
173   ##### Pod Settings #####
174
175   imageRepository: bearsrus
176   releaseVersion: v2023.05.05
177   imagePullPolicy: IfNotPresent
```

```

175 resources:
176   limits:
177     cpu: 1000m
178     memory: 1024Mi
179   requests:
180     cpu: 1000m
181     memory: 1024Mi
182
183 ##### Arkouda Locale Configuration #####
184
185 server:
186   port: # Arkouda port, defaults to 5555
187   memTrack: true
188   numLocales: 4
189   threadsPerLocale: 4
190 external:
191   persistence:
192     enabled: false
193     path: /arkouda-files # pod directory path, must match arkouda-udp-server
194     hostPath: /mnt/arkouda # host directory path, must match arkouda-udp-server
195 secrets:
196   tls: arkouda-tls # name of tls secret used to access Kubernetes API
197   ssh: arkouda-ssh # name of ssh secret used to launch Arkouda locales
198   '''
199
200 These can be deployed by moving into the 'arkouda-helm-charts' dir and running
201   ↪ the following command:
202
203   '''bash
204   helm install -n arkouda arkouda-locale arkouda-udp-locale/
205   '''
206
207   ### Arkouda-Server
208
209   Same goes for the 'arkouda-udp-server.yaml' file. \
210   For reference, the following is the configuration on my test setup.
211   (to find out what the 'k8sHost' is, run 'kubectl cluster-info')
212
213   '''yaml
214   resources:
215     limits:
216       cpu: 1000m
217       memory: 1024Mi
218     requests:
219       cpu: 1000m
220       memory: 1024Mi
221
222   ##### Pod Settings #####
223
224   imageRepository: bearsrus
225   releaseVersion: v2023.05.05

```

```
225 imagePullPolicy: IfNotPresent
226
227 ##### Arkouda Driver Configuration #####
228
229 server:
230   numLocales: 1 # total number of Arkouda locales = number of
231     ↳ arkouda-udp-locale pods + 1
232   authenticate: false # whether to require token authentication
233   verbose: true # enable verbose logging
234   threadsPerLocale: 5 # number of cpu cores to be used per locale
235   memMax: 2000 # maximum bytes of RAM to be used per locale
236   memTrack: true
237   logLevel: LogLevel.DEBUG
238   service:
239     type: ClusterIP # k8s service type, usually ClusterIP, NodePort, or
240       ↳ LoadBalancer
241     port: # k8s service port Arkouda is listening on, defaults to 5555
242     nodeport: # if service type is Nodeport
243     name: # k8s service name
244   metrics:
245     collectMetrics: false # whether to collect metrics and make them available
246       ↳ via k8s service
247     service:
248       name: # k8s service name for the Arkouda metrics service endpoint
249       port: # k8s service port for the Arkouda metrics service endpoint,
250         ↳ defaults to 5556
251       targetPort: # k8s targetPort mapping to the Arkouda metrics port,
252         ↳ defaults to 5556
253   locale:
254     appName: arkouda-locale
255     podMethod: GET_POD_IPS
256   external:
257     persistence:
258       enabled: true
259       path: /opt/locale # pod directory path, must match arkouda-udp-locale
260       hostPath: /mnt/arkouda # host machine path, must match arkouda-udp-locale
261     k8sHost: https://192.168.49.2:8443
262     namespace: arkouda # namespace Arkouda will register service
263     service:
264       name: arkoudaserver # k8s service name Arkouda will register
265       port: # k8s service port Arkouda will register, defaults to 5555
266   metricsExporter:
267     imageRepository: bearsrus
268     releaseVersion: v2023.05.05 # prometheus-arkouda-exporter release version
269     imagePullPolicy: IfNotPresent
270     service:
271       name: # prometheus-arkouda-exporter service name
272       port: 5080 # prometheus-arkouda-exporter service port, defaults to 5080
273     pollingIntervalSeconds: 5
274   secrets:
275     tls: arkouda-tls # name of tls secret used to access Kubernetes API
```

```

271  ssh: arkouda-ssh # name of ssh secret used to launch Arkouda locales
272  '''
273
274  Which can be deployed by moving into the 'arkouda-helm-charts' dir and running
    ↪ the following command:
275
276  '''bash
277  helm install -n arkouda arkouda-server arkouda-udp-server/
278  '''
279
280  Horray! We now have a working Arkouda cluster running in our kubernetes cluster.
281
282  # Pachykouda - Client
283
284  Now we have to create an image which enables pachyderm to send messages to the
    ↪ arkouda cluster.
285  To accomplish this we need to create a docker image which contains the arkouda
    ↪ client, takes the arkouda server ip and arbitrary arkouda commands as
    ↪ arguments and then executes the commands on the server.
286
287  ## Local Registry
288
289  To be able to develop and deploy this image locally, we need to set up a local
    ↪ docker registry within the kubernetes cluster.
290
291  '''bash
292  sudo mkdir -p /mnt/registry/certs
293
294  # create the certificate
295
296  sudo openssl req -newkey rsa:4096 -nodes -sha256 -keyout
    ↪ /mnt/registry/certs/registry.key -addext "subjectAltName =
    ↪ DNS:master-node-k8" -x509 -days 365 -out /mnt/registry/certs/registry.crt
297
298  sudo chown -R nobody:nogroup /mnt/registry
299  '''
300
301  Now if you want to push or pull from this repository you need to add the
    ↪ certificate to your trusted certificates.
302
303  '''bash
304  sudo -S bash -c 'openssl s_client -showcerts -connect
    ↪ heydar20.labs.hpecorp.net:31320 </dev/null 2>/dev/null | openssl x509
    ↪ -outform PEM > /tmp/heydar20.labs.hpecorp.net.pem && mkdir -p
    ↪ /etc/docker/certs.d/heydar20.labs.hpecorp.net:31320 && cp
    ↪ /tmp/heydar20.labs.hpecorp.net.pem
    ↪ /etc/docker/certs.d/heydar20.labs.hpecorp.net:31320/ca.crt && systemctl
    ↪ restart docker'
305
306  '''

```

List of literature

- 24 Top AI Statistics & Trends In 2023 – Forbes Advisor (2023). URL: <https://www.forbes.com/advisor/business/ai-statistics/> (retrieval: 10/07/2023).
- Ansible (2023). Ansible. URL: <https://github.com/ansible/ansible> (retrieval: 10/22/2023).
- Argo-Cd/LICENSE at Master · Argoproj/Argo-Cd (2023). GitHub. URL: <https://github.com/argoproj/argo-cd/blob/master/LICENSE> (retrieval: 10/21/2023).
- Argoproj/Argo-Workflows (2023). Argo Project. URL: <https://github.com/argoproj/argo-workflows> (retrieval: 10/07/2023).
- Arkouda Gituhb Repository (2023). Bears-R-Us. URL: <https://github.com/Bears-R-Us/arkouda> (retrieval: 10/23/2023).
- Arkouda-Contrib/Arkouda-Docker at Main · Bears-R-Us/Arkouda-Contrib (2023). GitHub. URL: <https://github.com/Bears-R-Us/arkouda-contrib/tree/main/arkouda-docker> (retrieval: 10/23/2023).
- Artifacthub Pachyderm 2.6.4 (2023). URL: <https://artifacthub.io/packages/helm/pachyderm/pachyderm/2.6.4> (retrieval: 10/18/2023).
- Bears-R-Us/Arkouda-Contrib/Arkouda-Helm-Charts (2023). GitHub. URL: <https://github.com/Bears-R-Us/arkouda-contrib/tree/main/arkouda-helm-charts/arkouda-udp-server> (retrieval: 10/25/2023).
- Boehm, B. W. (1988): A Spiral Model of Software Development and Enhancement. In: *Computer* 21.5, pp. 61–72. ISSN: 0018-9162. DOI: 10.1109/2.59. URL: <http://ieeexplore.ieee.org/document/59/> (retrieval: 10/18/2023).
- Budde, R./Kautz, K./Kuhlenkamp, K./Züllighoven, H. (1992): What Is Prototyping? In: *Information Technology & People* 6.2/3, pp. 89–95. ISSN: 0959-3845. DOI: 10.1108/EUM0000000003546. URL: <https://doi.org/10.1108/EUM0000000003546> (retrieval: 10/02/2023).
- Bzeznik, B./Henriot, O./Reis, V./Richard, O./Tavard, L. (2017): Nix as HPC Package Management System. In: *Proceedings of the Fourth International Workshop on HPC User Support Tools*. HUST’17. New York, NY, USA: Association for Computing Machinery, pp. 1–6. ISBN: 978-1-4503-5130-0. DOI: 10.1145/3152493.3152556. URL: <https://dl.acm.org/doi/10.1145/3152493.3152556> (retrieval: 10/03/2023).
- Canon, R. S./Younge, A. (2019): A Case for Portability and Reproducibility of HPC Containers. In: *2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC)*. 2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC). Denver, CO, USA: IEEE, pp. 49–54. ISBN: 978-1-72816-028-3. DOI: 10.1109/CANOPIE-HPC49598.2019.00012. URL: <https://ieeexplore.ieee.org/document/8950982/> (retrieval: 10/04/2023).
- Ceph.io — Home (2023). URL: <https://ceph.io/en/> (retrieval: 10/22/2023).
- Chapel-Lang (2023): Chapel-Lang/Chapel: A Productive Parallel Programming Language. URL: <https://github.com/chapel-lang/chapel> (retrieval: 10/23/2023).
- Cluster Networking (2023). Kubernetes. URL: <https://kubernetes.io/docs/concepts/cluster-administration/networking/> (retrieval: 10/18/2023).

- Creating a Cluster with Kubeadm (2023). Kubernetes. URL: <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/> (retrieval: 10/18/2023).
- Cwl-Utils/LICENSE at Main · Common-Workflow-Language/Cwl-Utils (2023). GitHub. URL: <https://github.com/common-workflow-language/cwl-utils/blob/main/LICENSE> (retrieval: 10/21/2023).
- Docker FAQs | Docker (2021). URL: <https://www.docker.com/pricing/faq/> (retrieval: 10/18/2023).
- Docker Terms of Service | Docker (2022). URL: <https://www.docker.com/legal/docker-terms-service/> (retrieval: 10/18/2023).
- Dubois, P./Epperly, T./Kumfert, G. (2003):** Why Johnny Can't Build [Portable Scientific Software]. In: *Computing in Science & Engineering* 5.5, pp. 83–88. ISSN: 1558-366X. DOI: 10.1109/MCISE.2003.1225867. URL: <https://ieeexplore.ieee.org/abstract/document/1225867> (retrieval: 10/03/2023).
- Eckerth, J. (2023):** Installation Instructions Minikube Commit 69a05a755facc8d387e031ff5991c20d46db URL: [https://github.com/Jo-Eck/ProjectPaper-2/tree/69a05a755facc8d387e031ff5991c20d46db](https://github.com/Jo-Eck/ProjectPaper-2/tree/69a05a755facc8d387e031ff5991c20d46db/project/Pachyderm) project/Pachyderm (retrieval: 10/18/2023).
- Egwutuoha, I. P./Levy, D./Selic, B./Chen, S. (2013):** A Survey of Fault Tolerance Mechanisms and Checkpoint/Restart Implementations for High Performance Computing Systems. In: *J Supercomput* 65.3, pp. 1302–1326. ISSN: 1573-0484. DOI: 10.1007/s11227-013-0884-0. URL: <https://doi.org/10.1007/s11227-013-0884-0> (retrieval: 10/03/2023).
- Explore Network Plugins for Kubernetes (2023): Explore Network Plugins for Kubernetes: CNI Explained | TechTarget. IT Operations. URL: <https://www.techtarget.com/searchitoperations/tip/Explore-network-plugins-for-Kubernetes-CNI-explained> (retrieval: 10/19/2023).
- Flannel (2023). flannel-io. URL: <https://github.com/flannel-io/flannel> (retrieval: 10/19/2023).
- Flannel Install Config (2023). URL: <https://github.com/flannel-io/flannel/blob/master/Documentation/kube-flannel.yml> (retrieval: 10/19/2023).
- Fülöp, J. (2005):** Introduction to Decision Making Methods. In: *BDEI-3 Workshop, Washington*, pp. 1–15. URL: <https://www.academia.edu/download/43447287/decisionmakingmethods.pdf> (retrieval: 10/12/2023).
- Gamblin, T./LeGendre, M./Collette, M. R./Lee, G. L./Moody, A./de Supinski, B. R./Futral, S. (2015):** The Spack Package Manager: Bringing Order to HPC Software Chaos. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '15. New York, NY, USA: Association for Computing Machinery, pp. 1–12. ISBN: 978-1-4503-3723-6. DOI: 10.1145/2807591.2807623. URL: <https://dl.acm.org/doi/10.1145/2807591.2807623> (retrieval: 10/03/2023).
- Gluster (2023). URL: <https://www.gluster.org/> (retrieval: 10/22/2023).
- Haines, S. (2022):** 'Workflow Orchestration with Apache Airflow'. In: *Modern Data Engineering with Apache Spark: A Hands-On Guide for Building Mission-Critical Streaming Applications*. Springer, pp. 255–295.
- Helm Docs Home (2023). URL: <https://helm.sh/docs/> (retrieval: 10/18/2023).

- Home - Knative (2023). URL: <https://knative.dev/docs/> (retrieval: 10/07/2023).
- Hoste, K./Timmerman, J./Georges, A./De Weirdt, S. (2012):** EasyBuild: Building Software with Ease. In: *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*. 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, pp. 572–582. DOI: 10.1109/SC.Companion.2012.81. URL: <https://ieeexplore.ieee.org/abstract/document/6495863> (retrieval: 10/03/2023).
- Inc, M. (2023):** MinIO | MinIO for Kubernetes. MinIO. URL: <https://min.io> (retrieval: 10/18/2023).
- Installing Kubeadm (2023). Kubernetes. URL: <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/> (retrieval: 10/22/2023).
- Intro to Pipelines (2023). URL: <https://docs.pachyderm.com/latest/learn/intro-pipelines/> (retrieval: 10/23/2023).
- iPaaS Solution for the Enterprise (2023). SnapLogic. URL: <https://www.snaplogic.com/> (retrieval: 10/07/2023).
- Knative Docs/LICENSE at Main · Knative/Docs (2023). GitHub. URL: <https://github.com/knative/docs/blob/main/LICENSE> (retrieval: 10/21/2023).
- Kubeflow (2023). Kubeflow. URL: <https://www.kubeflow.org/> (retrieval: 10/07/2023).
- Kubeflow/LICENSE at Master · Kubeflow/Kubeflow (2023). GitHub. URL: <https://github.com/kubeflow/kubeflow/blob/master/LICENSE> (retrieval: 10/21/2023).
- Kubernetes CNI Plugins (2023). Kubernetes. URL: <https://kubernetes.io/docs/concepts/cluster-administration/addons/> (retrieval: 10/19/2023).
- License — Airflow Documentation (2023). URL: <https://airflow.apache.org/docs/apache-airflow/stable/license.html> (retrieval: 10/21/2023).
- Luigi/LICENSE at Master · Spotify/Luigi (2023). GitHub. URL: <https://github.com/spotify/luigi/blob/master/LICENSE> (retrieval: 10/21/2023).
- Mehndiratta, H. (2023):** Comparing Kubernetes Container Network Interface (CNI) Providers | Kubevious.Io. URL: <https://kubevious.io/blog/post/comparing-kubernetes-container-network-interface-cni-providers> (retrieval: 10/19/2023).
- Merrill, M./Reus, W./Neumann, T. (2019):** Arkouda: Interactive Data Exploration Backed by Chapel. In: *Proceedings of the ACM SIGPLAN 6th on Chapel Implementers and Users Workshop*. CHIUIW 2019. New York, NY, USA: Association for Computing Machinery, p. 28. ISBN: 978-1-4503-6800-1. DOI: 10.1145/3329722.3330148. URL: <https://dl.acm.org/doi/10.1145/3329722.3330148> (retrieval: 10/04/2023).
- Multi-Criteria Decision Analysis for Use in Transport Decision Making (2014). Red. by Michael Bruhn Barfod/Steen Leleur. DTU Lyngby: DTU Transport.
- Pachyderm Docs - Local Deploy (2023). Pachyderm Docs - Local Deploy. URL: <https://docs.pachyderm.com/2.6.x/set-up/local-deploy/> (retrieval: 10/18/2023).
- Pachyderm Docs - On-Prem Deploy (2023). Pachyderm Docs - On-Prem Deploy. URL: <https://docs.pachyderm.com/latest/set-up/on-prem/> (retrieval: 10/18/2023).
- Pachyderm/LICENSE at Master · Pachyderm/Pachyderm (2023). URL: <https://github.com/pachyderm/pachyderm/blob/master/LICENSE> (retrieval: 10/21/2023).

- Pahcyderm -Pricing (2022). Pachyderm. URL: <https://www.pachyderm.com/pricing/> (retrieval: 10/21/2023).
- Roberts, R./Goodwin, P. (2002):** Weight Approximations in Multi-Attribute Decision Models. In: *Journal of Multi-Criteria Decision Analysis* 11.6, pp. 291–303. ISSN: 1099-1360. DOI: 10.1002/mcda.320. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/mcda.320> (retrieval: 10/16/2023).
- Rook (2023). URL: <https://rook.io/> (retrieval: 10/22/2023).
- Sayers, C./Laffitte, H./Reddy, P./Ozonat, K./Sayal, M./Simitsis, A./Singhal, S./Koutrika, G./Das, M./Aji, A./Bosamiya, H./Riss, M./Wilkinson, K./Lucio, J./Cantal, A./Carvalho, C. (2015):** Cloud Application Services Platform (CLASP): User Guide, Introduction, and Operation. In: pp. 1–60.
- SnapLogic – Master Subscription Agreement (2023). SnapLogic. URL: <https://www.snaplogic.com/master-subscription-agreement> (retrieval: 10/21/2023).
- Spotify/Luigi* (2023). Spotify. URL: <https://github.com/spotify/luigi> (retrieval: 10/12/2023).
- Triantaphyllou, E. (2000):** ‘Introduction to Multi-Criteria Decision Making’. In: *Multi-Criteria Decision Making Methods: A Comparative Study*. Ed. by Evangelos Triantaphyllou. Applied Optimization. Boston, MA: Springer US, pp. 1–4. ISBN: 978-1-4757-3157-6. DOI: 10.1007/978-1-4757-3157-6_1. URL: https://doi.org/10.1007/978-1-4757-3157-6_1 (retrieval: 10/12/2023).
- Weighted Sum Model* (2022). In: *Wikipedia*. URL: https://en.wikipedia.org/w/index.php?title=Weighted_sum_model&oldid=1114224941 (retrieval: 10/16/2023).
- Wilde, T./Hess, T. (w.y.):** Methodenspektrum der Wirtschaftsinformatik: Überblick und Portfoliobildung. In.