

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 November 6, 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, November 6, 2023



Jon Eckerth

Abstract:

TODO

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	3
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	10
3.2.1 Loosely Coupled Problems	10
3.2.2 Tightly Coupled Problems	11
4 Creation of the Artifact	12
4.1 Initial Goals	12
4.2 Overall Structure	12
4.3 Selection of Workflow Management Tools	13
4.4 Implementation of the Artifact	17
4.4.1 Infrastructure	18
4.4.2 Tightly Coupled HPC Workloads	21
4.4.3 Supplementary Services	25
4.5 Evaluation of the Artifact	29
5 Summary and Outlook	31
5.1 Summary	31
5.2 Outlook	31
Appendix	32
List of literature	82

List of abbreviations

AI	Artificial Intelligence
CaC	Configuration as Code
CC	Cloud Computing
CI/CD	Continuous Integration/Continuous Delivery
CLASP	Cloud Application Services Platform
CNCF	Cloud Native Computing Foundation
CNI	Container Network Interface
CWL	Common Workflow Language
FAM	Fabric Attached Memory
FOSS	Free and Open Source Software
GASNet	Global Address Space Networking
HPC	High Performance Computing
HPE	Hewlett Packard Enterprise
IaC	Infrastructure as Code
IP	Intellectual Property
k8s	Kubernetes
LCP	Loosely Coupled Problems
LXC	Linux Containers
MAUT	Multi-attribute Utility Theory
ML	Machine Learning
NPO	Non-Governmental Organization
PFS	Pachyderm File System
PoC	Proof of Concept
PV	Persistent Volume
RAD	Rapid Application Development
RBAC	Role Based Access Control
RDMA	Remote Direct Memory Access
ROC	Rank order centroid
RR	Rank Reciprocal

RS	Rank Sum
SMART-ER	SMART Exploiting Ranks
SMART	Simple Multiattribute Rating Technique
SME	Subject-Matter Expert
SSO	Singele Sign On
TCP	Tightly Coupled Problems
UDP	User Datagram Protocol
VM	Virtual Machine
WSM	Weighted Sum Model

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	13
7	Pachykouda datum distribution amongst workers	21
8	Arkouda workers on Kubernetes (k8s)	23
9	Pachyderm High-Level Architecture	25
10	Stars and Stack overflow Questions Comparison	33
11	ROC weights	37
12	RR weights	37
13	RS weights	38
14	Swim lane Diagram of the communication between the user and Pachyderm	39

List of Tables

1	Weighting of the criteria	16
2	Evaluation of the suggested tools	17
3	Evaluation of the additional tools	17
4	Comparison of Project Popularity	33
5	Composite scores of Workflow managers, sorted by final score	34

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, analyze, 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 use cases, 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 Arkouda⁹ 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?*

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:

⁹Merrill/Reus/Neumann 2019

¹⁰Home Page / Pachyderm 2023

- **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' running on the Heydar Cluster*
- **C3:** *Further integrations of tools from both sides of the spectrum, addressing many of aforementioned pain-points*

2 Methodology

2.1 Prototyping

The methodology of Prototyping in the context of software development has been a well established practice for many years and has been discussed in literature for at least the last 40 years¹¹. It is especially useful in complex situations where the requirements might not be fully known or understood at the beginning of the project and need to be discovered in an iterative and exploratory process, as it enables the project to be partitioned into more manageable chunks which can be tackled individually¹².

At their core prototyping is a state transition model, where a final state is approached in a series of non-ideal intermediate steps, each of which revealing new information about the problem domain and the solution space and applying this new information to the next iteration¹³.

It is described as the methodology of with the highest constructive plasticity, as it supports many different flavors and variations of the core concept, but as long as it follows the core principles of this iterative approximation of an ideal version, and specifically evaluates the results of the iterations it should be considered a methodologically complete scientific method¹⁴.

For this project we will be using a concatenated version of the so called Rapid Application Development (RAD) model, which has been quite established in software development, as it is very well suited for projects with a high degree of complexity and uncertainty, while reducing the administrative overhead of the project¹⁵. Reducing the administrative overhead is especially important for this project, as the time frame is limited to 3 months, and the project is being done by a single person, therefore more complex tools like the spiral model¹⁶, which is a popular model emphasizing a risk analysis with each step of the iteration, would be too time-consuming to be feasible for this project.

First there will be a central initial goal which will be the goal of what we are trying to reach, as the requirements will change with the knowledge acquired and the scope shifted with each iteration, we do not need to plan as meticulously as we would for a waterfall approach.

We then enter the section with the prototyping iterations, where we will alternate between the implementation of what we planned in the previous section and the evaluation of the results of the previous iteration which serve as the basis for the next iteration.

¹¹Gomaa 1983

¹²Naumann/Jenkins 1982

¹³Kraushaar/Shirland 1985

¹⁴Wilde/Hess w.y.

¹⁵Martin 1991

¹⁶Boehm 1988

Then a final evaluation of the whole project will be done, where the final results will be evaluated and the project will be concluded, and advice for the pickup of the project will be given.

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 multidimensional.

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 framework¹⁷. 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.

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

Abb. 1: Formula for calculating the WSM score¹⁸

Where:

- w_j : This represents the weight assigned to the j -th criterion. Weights are determined by the decision-makers based on the relative importance of each criterion. They should be normalized (i.e., the sum of all weights should be 1 or 100%) to maintain a consistent scale.
- a_{ij} : This represents the score or rating of the i -th alternative concerning the j -th criterion. This score is an assessment of how well the alternative meets or satisfies the specific criterion.

¹⁷Triantaphyllou 2000, p. 1

¹⁸Weighted Sum Model 2022

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.

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 score¹⁹

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 .

¹⁹Taken from Fülöp 2005, p. 6

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 nature²⁰.

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 weights²¹. 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.

²⁰Fülöp 2005, p. 6

²⁰Multi-Criteria Decision Analysis for Use in Transport Decision Making 2014, p. 26

²¹Roberts/Goodwin 2002, p. 296

3 State of the Art

While this project assumes that the reader is already familiar with the basic concepts of High Performance Computation as well as having a basic understanding of the cloud computing tech stack, especially containerization and software defined infrastructure, this chapter will give a brief overview of the current state of the art in the field of High Performance Computing and Cloud Computing.

3.1 Containerization

Containerization is the process of isolating a process from the rest of the user space of the operating system. It essentially creates a virtual environment for the process to run in, which can be customized to the needs of the process, without affecting the rest of the system. This makes it possible to run multiple processes on the same machine, without having to worry about them interfering with each other or the rest of the system²². This is especially useful for dependency management, as it allows running multiple versions of the same library on the same machine, and only the process that needs a specific version of the library will use it. In comparison to the more traditional approach of using virtual machines, containerization is much more lightweight, as it does not need to emulate a whole operating system, but builds on top of the hosts kernel and only isolates the user space²³.

Container Solutions

Currently the containerization paradigm is spreading rapidly with multiple solutions available, the most popular being Docker²⁴. But within the HPC community Singularity²⁵ has gained more and more traction, while the previous options have historically been more popular in CC. The main differentiating factor between the two is that Docker is reliant on the Docker daemon, which needs to be run as root, while Singularity does not need a daemon and can be run as unprivileged user. This generally makes Singularity generally more appealing to HPC users, as it is more in line with the security policies of most HPC clusters, and simplifies the process of running containers on HPC clusters. But Docker is still the more popular option in CC, as it is more flexible and has a larger ecosystem of tools and services built around it which makes rapid prototyping and development easier.

²²*IEEE Xplore Full-Text PDF*: 2023

²³*What Is a Container?* 2023

²⁴*Stack Overflow Developer Survey 2022* 2023

²⁵‘Introduction to Software Containers with Singularity.’ w.y.

These tools make use of two features of the Linux kernel, namely cgroups and namespaces²⁶. Cgroups are used to limit the resources a process can use, while namespaces are used to isolate the user space of the process.

Software defined Infrastructure

Through the advent of large scale cloud endeavors and their offering of dynamically scaling services, the need for a way to automatically manage and partition the underlying infrastructure arose. This led to the development of software defined infrastructure, which is the process of abstracting the underlying infrastructure and managing it through software. The paradigm of software defined is especially pronounced in the field of networking²⁷, where software defined networking is used to manage the underlying network infrastructure, where the different underlying network devices are abstracted to a homogeneous network, which then gets managed through virtualization and software²⁸. Large scale cloud providers like Amazon Web Services, Microsoft Azure and Google Cloud Platform all make use of software defined infrastructure to manage their underlying infrastructure.

Large Scale Container Orchestration

These large scale software defined infrastructures are used in tandem and managed by large scale orchestrators like OpenShift²⁹, Openstack³⁰ and Kubernetes³¹. The orchestrators interface with the underlying infrastructure and manage the services running on top of it and provide a unified interface for the user to interact with. This makes it possible to largely run on any, even heterogeneous, infrastructure, while providing a highly flexible and scalable interface for the user to interact with. Making it especially useful for large scale cloud providers, as it allows them to provide a unified interface for their users, while still being able to use different underlying infrastructure.

3.2 High Performance Computing Frameworks

3.2.1 Loosely Coupled Problems

Loosely Coupled Problems (LCP) also known in the industry as "embarrassingly parallel"³² problems are problems that can be broken up into smaller independent tasks that can be executed in parallel.

²⁶ *What Is a Container?* 2023

²⁷ Xia et al. 2015

²⁸ Baur et al. 2015

²⁹ *Red Hat OpenShift Enterprise Kubernetes Container Platform* 2023

³⁰ *Open Source Cloud Computing Infrastructure* 2023

³¹ *Production-Grade Container Orchestration* 2023

³² **smtm**

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.

Message Passing Interface (MPI) vs Shared Memory (OpenMP) or Partioned Global Address Space (PGAS)³³

³³**smtn**

4 Creation of the Artifact

4.1 Initial Goals

As this project was first and foremost a project, designed to interactively explore the problem space 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 an easily graspable example of its capabilities.

While we are approaching the problem from the perspective of the HPC community, the intended end user of this tool are the data scientists and Subject-Matter Experts (SMEs) that are working with the HPC systems, and therefore the tool needs to be designed and selected with the fact in mind that the end user 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

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 end user (right).

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

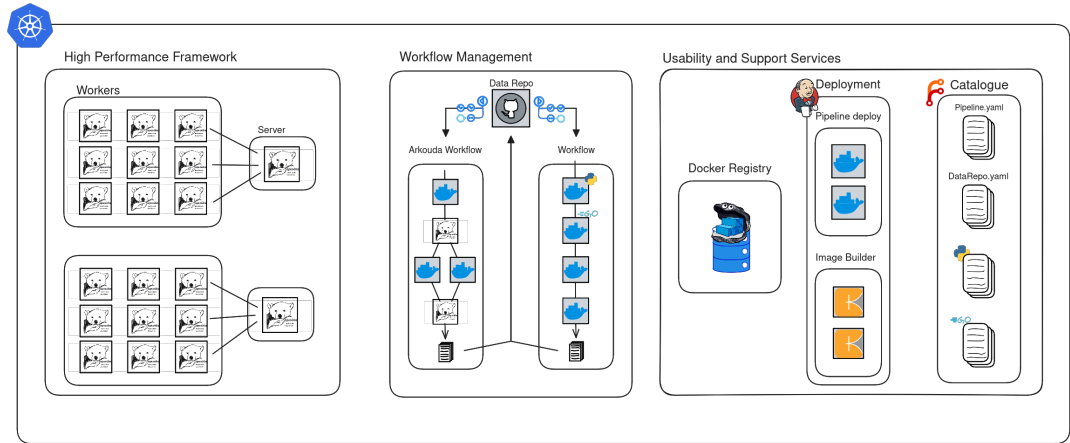


Abb. 6: Pachykouda high level infrastructure diagram

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) project³⁴.
- **Cloud Application Services Platform (CLASP):** An in-house developed workflow manager, written in Java, utilizing Servlet to execute workflows³⁵.
- **Snaplogic:** A commercial low-code/no-code workflow manager with a focus on data integration and data engineering³⁶.

³⁴ Argoproj/Argo-Workflows 2023

³⁵ Sayers, C. et al. 2015

³⁶ iPaaS Solution for the Enterprise 2023

But given that it was possible to select projects outside 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 extensibility³⁷.
- **Kubeflow:** A k8s-native platform for deploying, monitoring, and running ML workflows and experiments, also a CNCF project, streamlining ML operations alongside other Kubernetes resources³⁸.
- **Knative:** An open-source k8s-based platform to build, deploy, and manage modern serverless workloads, simplifying the process of building cloud-native applications³⁹.
- **Luigi:** An open-source Python module created by Spotify to build complex pipelines of batch jobs, handling dependency resolution, workflow management, and visualization seamlessly⁴⁰.
- **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 hinted end users of the tool are not primarily HPC experts, the tool needs to be easy to use and understand, and should not require the end user 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 end users 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 end users, allowing them to easily extend the developed tool further to their needs.

³⁷Haines 2022

³⁸Kubeflow 2023

³⁹Home - Knative 2023

⁴⁰Spotify/Luigi 2023

- **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 end users 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 years⁴¹, 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 intersects with many 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.

Weighing 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.

⁴¹ 24 Top AI Statistics & Trends In 2023 – Forbes Advisor 2023

3. **License** This criterion has to be weighted carefully, as a highly restrictive license might be a deal-breaker, 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 considered as well.
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 14.

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 as well as their weighing, we can begin 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

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

Criteria	Pachyderm	Argo	CLASP	Snaplogic
Ease of use	TBD	TBD	TBD	TBD
Extensibility	TBD	TBD	TBD	TBD
Community, Support & Docs	10	2.32	2.5	5.03
Maturity	TBD	TBD	TBD	TBD
Strategic alignment	TBD	TBD	TBD	TBD
License	10	7.5	10	0
Cost	TBD	TBD	TBD	TBD

Tab. 2: Evaluation of the suggested tools

Criteria	Airflow	Kubeflow	Knative	Luigi	CWL
Ease of use	TBD	TBD	TBD	TBD	TBD
Extensibility	TBD	TBD	TBD	TBD	TBD
Community, Support & Docs	10	2.25	0.74	2.29	0.22
Maturity	TBD	TBD	TBD	TBD	TBD
Strategic alignment	TBD	TBD	TBD	TBD	TBD
License	7.5	7.5	7.5	7.5	7.5
Cost	TBD	TBD	TBD	TBD	TBD

Tab. 3: Evaluation of the additional tools

Conclusion of the Selection Process

TODO: Write conclusion 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 were happening concurrently, they each address a different aspect of the project and therefore mostly 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 tool⁴². The Pachyderm documentation gave two recommendations for setting up an initial development environment, preferably Docker Desktop or alternatively Minikube⁴³. Due to the exclusive license of Docker-Desktop⁴⁴, which prevents large companies free usage of the product⁴⁵ 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 storage⁴⁶, 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 hard drive. Using the Helm packagemanager⁴⁷ for k8s the at that point the newest version 2.6.4 was installed from the official Artifacthub repository⁴⁸.

The host 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 repository⁴⁹, 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 tooling 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

⁴²*Pachyderm Docs - On-Prem Deploy* 2023

⁴³*Pachyderm Docs - Local Deploy* 2023

⁴⁴*Docker Terms of Service* / Docker 2022

⁴⁵*Docker FAQs* / Docker 2021

⁴⁶Inc 2023

⁴⁷*Helm Docs Home* 2023

⁴⁸*Artifacthub Pachyderm 2.6.4* 2023

⁴⁹Eckert 2023

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 infrastructure⁵⁰.

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 cluster⁵¹, 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 5/1.

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 space⁵².

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 infrastructure⁵³.

Here we are 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 CNIs⁵⁴, which all fulfill this essential task in different ways. As the needs regarding the network plugin were 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 clusters⁵⁵, provided security and enterprise support as well having a wide range of features⁵⁶. But Calico proved to be more difficult to set up than expected, after consulting with a colleague who set up a different cluster with Calico, it was decided to use Flannel as a CNI instead. Flannel turned out

⁵⁰ *Creating a Cluster with Kubeadm* 2023

⁵¹ *Ansible* 2023

⁵² *Installing Kubeadm* 2023

⁵³ *Cluster Networking* 2023

⁵⁴ *Kubernetes CNI Plugins* 2023

⁵⁵ *Explore Network Plugins for Kubernetes* 2023

⁵⁶ *Mehndiratta* 2023

to be much easier to set up and configure, as it is a very lightweight CNI which is designed for bare metal clusters⁵⁷, and foregoes the more advanced security features of Calico.

The Flannel configuration used for the cluster can be found at Appendix 5/2 it is closely based on the example configuration provided by the Flannel documentation⁵⁸.

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 hard drive 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 Ceph⁵⁹ or GlusterFS⁶⁰ in combination with the Rook project⁶¹. Which will need to be explored in future iterations.

As described in section 4.4.2 a service hosting Fabric Attached Memory (FAM) will be needed in future iterations as well.

⁵⁷ *Flannel* 2023

⁵⁸ *Flannel Install Config* 2023

⁵⁹ *Ceph.Io — Home* 2023

⁶⁰ *Gluster* 2023

⁶¹ *Rook* 2023

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.

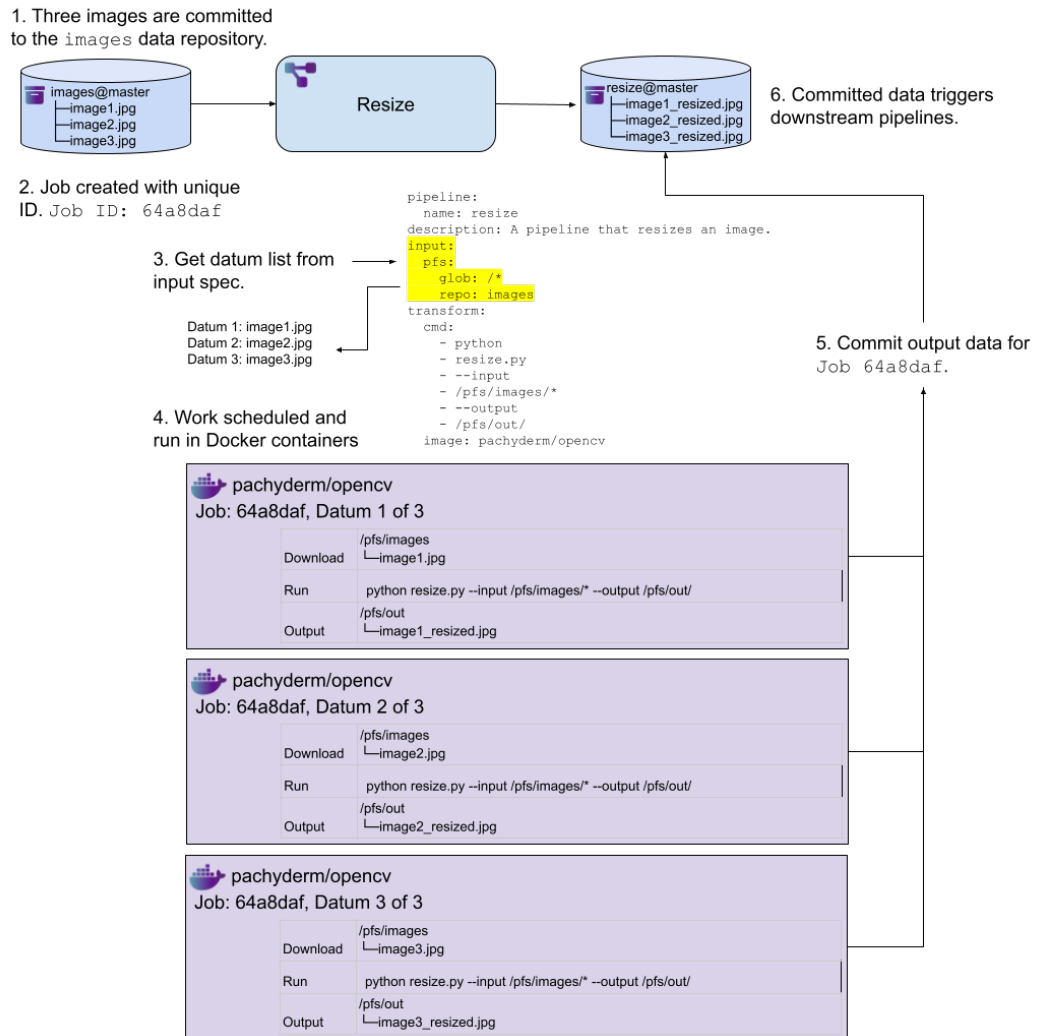


Abb. 7: Pachykouda datum distribution amongst workers ⁶²

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

⁶²Taken from: *Intro to Pipelines* 2023

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 14

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 worker node 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 framework⁶³, which itself is a python binding for the Chapel programming language⁶⁴.

For that step an Arkouda worker was installed bare metal on the head node 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 team⁶⁵. The container was then modified to be able to communicate with the Arkouda worker on the head node of the cluster, it can now send data to the worker and receive the results.

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,

⁶³ Arkouda Github Repository 2023

⁶⁴ Chapel-Lang 2023

⁶⁵ Arkouda-Contrib/Arkouda-Docker at Main · Bears-R-Us/Arkouda-Contrib 2023

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

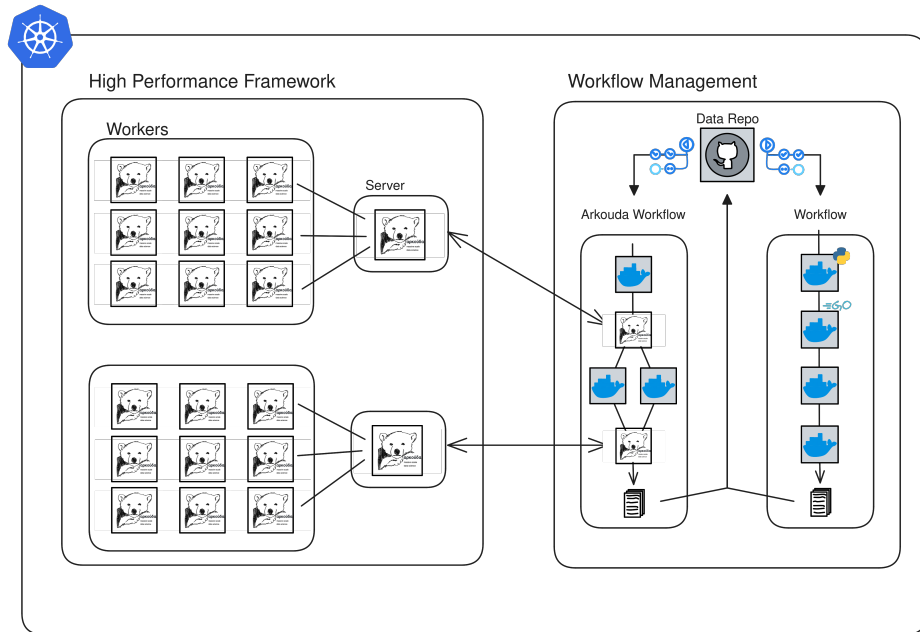


Abb. 8: Arkouda workers on the Heydar cluster

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 repo⁶⁶,

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 5/4 which in turn is based on the official Arkouda documentation⁶⁷.

⁶⁶ *Bears-R-Us/Arkouda-Contrib/Arkouda-Helm-Charts* 2023

⁶⁷ *Arkouda-Contrib/Arkouda-Docker at Main · Bears-R-Us/Arkouda-Contrib* 2023

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. Instead, they would need to spawn their own worker swarm.

Another issue is that there are currently going through the standard Pod to pod communication configuration of flannel, which means that the entire traffic between the client container and the Arkouda server as well as the traffic between the workers is all happening over emulated overlay network which enables the containers on the different nodes to communicate with each other as if they were on the same network, no matter of the actual infrastructure below it. The communication protocol of the Arkouda servers is UDP based Global Address Space Networking (GASNet), which provides the Remote Direct Memory Access (RDMA) needed for the Arkouda framework to work, but this incurs a significant overhead in the form of the encapsulation of the UDP packets into TCP packets.

Also, the containers are currently not compatible with the OpenFAM project⁶⁸, which is being developed as an integration to Arkouda and Chapel by the Hewlett Packard Systems Architecture Lab⁶⁹, it extends the Arkouda framework with the ability to use FAM as banks of RDMA enabled memory, which can be accessed by the Arkouda workers. This would prove to be a significant improvement as it has the potential to reduce the overall overhead of the communication⁷⁰ amongst the workers as well as to the server, by cutting down the overall amount of network traffic.

The pachyderm platform itself might also benefit from the integration of FAM, as it could be used to store the datums in the PFS, providing the running pipeline processes with a much faster access to the data.

Third iteration - FAM

While significant efforts have already been made to successfully integrate Arkouda and FAM, these have so far been focussing on bare metal installations, for that reason, in order to integrate the FAM enabled Arkouda working from within a containerized environment the tools would need to be custom recompiled matching the new environment. Therefore, we needed to:

1. Compile OpenFAM in the Container
2. Compile custom Chapel in the Container with OpenFAM
3. Compile custom Arkouda in the Container with the OpenFAM enabled Chapel

⁶⁸Keeton/Singhal/Raymond 2019

⁶⁹Byrne et al. 2023

⁷⁰Chou et al. 2019

4. Rebuild the Arkouda container with the new Arkouda binary
5. Rewrite the k8s deployment to make use of OpenFAM

This section was quite challenging as it required a deep understanding of the PoC implementations of the OpenFAM, Arkouda and Chapel projects and was cut short by the time constraints of the project and was therefore not brought to a successful conclusion. **The current state of the project can be found in the PoC repository⁷¹ and in the appendix at ??.** But this showed us that there is a lot of potential in the integration of FAM into the container based HPC world, as it could provide a significant performance boost to the overall system and should be explored further in future iterations.

4.4.3 Supplementary Services

As the other branches of the prototyping were happening, the need for additional services and infrastructure arose to support the development of the prototype as well as to increase the general usability of the prototype. This section will especially describe the services which help to make this prototype a more complete solution.

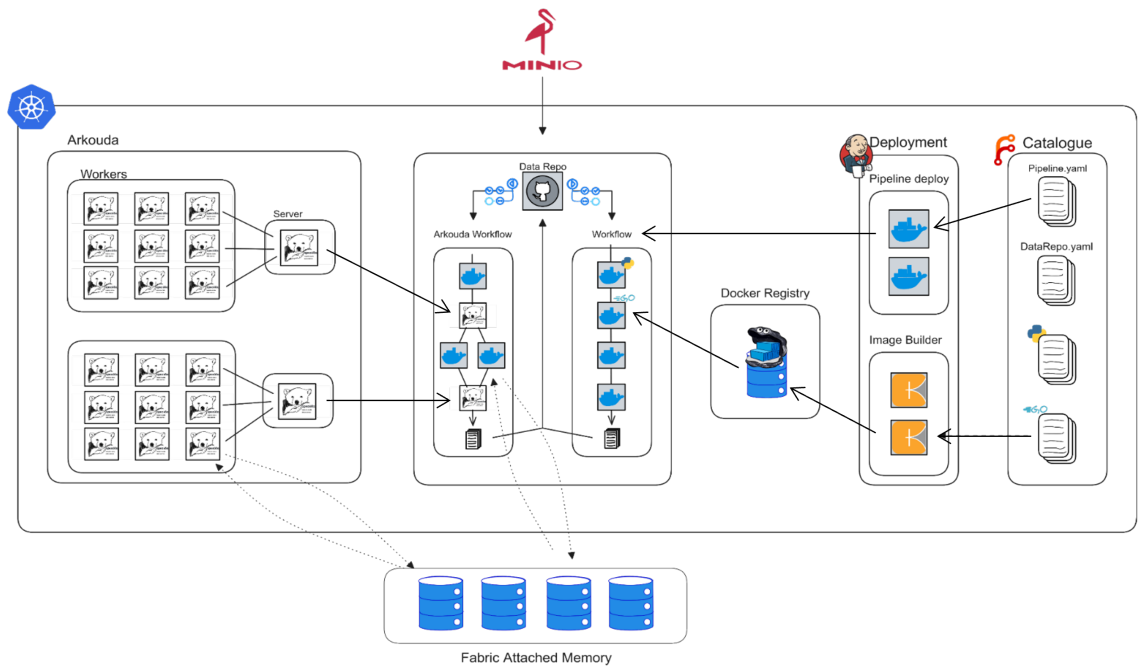


Abb. 9: Pachyderm High-Level Architecture

⁷¹eckerthPoCRepository2023

Docker Registry

One thing that was quite apparent from the get go, was the need for a central docker registry. As Pachyderm does not manage the docker images itself, but relies on the user to provide them somehow externally.

During the first iterations when the development was being done on Minikube as described in 4.4.1, the internal Registry of the node was enough. But as soon as we moved over to the Heydar system keeping the Hosts internal registries in sync was of course not feasible, Therefore we added a private docker registry to the cluster⁷². The deployment config is based on the official docker registry helm chart⁷³ and can be found at Appendix 5/5.

Frogejo Catalogue

As this should be the PoC of an end user tool, we should also look into usability features and based on previous experience of the team, the need for a catalogue of previously developed pipelines and processing code was identified. The idea was to create something similar to the CLASP catalogue⁷⁴ but for the Pachyderm ecosystem. Meaning that users can share, search and deploy workflows from a central catalog, without having to worry about the underlying infrastructure.

Having HPC software in a completely contained and version system directly addresses many of the original problem statements, described in 1.2, especially the problem of reproducibility, environment management and the lack of portability.

To start of we, installed a simple github-like interface for the catalogue, called Frogejo⁷⁵ which is a fork of the Free and Open Source Software (FOSS) project Gitea⁷⁶, being maintained by the Non-Governmental Organization (NPO) Codeberg e.V.⁷⁷. The installation was done using their official helm chart⁷⁸ and can be found at Appendix 5/6.

Jenkins

Now that we have a place to hold and version the code files and a place to hold the resulting docker images, we need a way to build and deploy them automatically. For that purpose an installation of Jenkins was added to the Cluster. Jenkins is a FOSS CI/CD tool, which is widely used in industry⁷⁹. It enables us to execute arbitrary code in an controlled environment based

⁷²Kumar 2020

⁷³*Docker-Registry 1.10.0 · Phntom/Phntom* 2023

⁷⁴Sayers, Craig et al. 2015

⁷⁵forgejo 2023

⁷⁶*Gitea - Git with a Cup of Tea* 2023

⁷⁷Codeberg 2023

⁷⁸*Forgejo 0.13.0 · Forgejo/Forgejo-Helm* 2023

⁷⁹*Jenkins - Market Share, Competitor Insights in Continuous Integration And Delivery* 2023

on triggers or schedules. We will be using it to automatically build and deploy docker images whenever code is pushed to the Forgejo catalogue, as well as automatically deploying pipeline scripts to Pachyderm.

By using this we ensure that the code and the docker images are always in sync and that the user does not have to worry about building and deploying the images manually or interacting with the underlying Cluster. We also extend the factor of provenance which was so far limited to the data itself, to the containers code and pipeline spec as well, now having total oversight into which input begets what output. This was achieved in multiple steps as this turned out to be quite an involved process.

First of was of course the general installation of the project into the cluster based on the image provided by Jenkins⁸⁰. Then we have to integrate Jenkins into the the Kubernetes cluster as well, as we want it to be able to spawn its own worker pods within the cluster to handle the building of the containers and pushing of the pipelines. Also an integration with the Forgejo catalogue was needed, so that Jenkins would be informed about new commits to the catalogue and could start the building process.

The installation instructions, RBAC and the configuration of the Jenkins installation can be found at 4.

CI/CD Pipeline

Now that we have a Jenkins installation which is able to spawn its own worker pods on the cluster and have set up System wide Webhooks for the Forgejo catalogue, which will inform Jenkins about every new commit to any repository on the catalogue we will need to develop a pipeline which will tell Jenkins what to do if a new commit is detected.

As the Pachyderm team say themselves, "*.../ users with limited experience with containerization, cloud computing, and distributed systems may find it challenging to use Pachyderm effectively.*"⁸¹ Unfortunate many SMEs are more focused on their domain specific knowledge and usually only have a limited understanding of the underlying infrastructure, an effect which has already been noticed in classical HPC⁸², which is likely more pronounced in a field which has only recently started to gain traction in the wider scientific community.

Therefore providing an way to deploy code and pipelines while minimizing the interaction with the underlying infrastructure is a key feature of this part of the prototype. The goal is to create a pipeline which can take a regular software project form the Forgejo catalogue, make reasonable assumptions about the project structure and build the required docker images and deploy the pipeline to Pachyderm.

⁸⁰ Jenkins 4.8.2 · Jenkins/Jenkinsci 2023

⁸¹ Pachyderm Target Audience 2023

⁸² Shenoi/Shah/Joshi 2019

As completely functional Pachyderm project, consisting of a N processing steps requires the following components:

- N code files, one for each processing step
- N Dockerfiles, one for each processing step
- N pipeline specifications, one for each processing step
- At least one description file for the data repository(s)
- If needed supporting files, which should be included in one ore more image, but should not get their own.

But as we want to minimize the interaction with the underlying infrastructure, we want to minimize the amount of input the user has to provide, without restricting the knowing users. In order to achieve this the pipeline first detects the existence of the above components. We then try to make a reasonable assumption about the structure of the project and the relation between the components, considering: directory structure, naming conventions and static code analysis. The pipeline will then try to fill in gaps in the project structure, like missing Dockerfile by using default values and insights gained form the previous steps. The pipeline itself is written in a mix of Groovy, Python and bash, the Code and a flowchart describing the process can be found in the appendix at ??.

There was also a custom container image created, extending official Jenkins worker image⁸³, enabling us to build docker images from within a running container in the cluster by using a dedicated kaniko sidecar container⁸⁴ which enables us to build docker images without having to run Docker in Docker, which is not recommended⁸⁵. The image also contains the necessary tools to interact with the Pahcyderm API and to run the python scripts which are used to analyze the project structure and do the code generation, it can be found at Appendix 5/7.

While this pipeline is currently working as intended, it is still missing many edgcases and does need further development to be able to handle more than the most basic projects. Even though its development was cut short it does show the potential of this approach and how it could be used to make the Pachyderm ecosystem more accessible to users which are not as familiar with the underlying infrastructure, while still allowing more experienced users to interact with the system directly.

⁸³ *Jenkins/Jenkins - Docker Image | Docker Hub* 2023

⁸⁴ *Kaniko - Build Images In Kubernetes* 2023

⁸⁵ *Using Docker-in-Docker for Your CI or Testing Environment? Think Twice.* 2023

4.5 Evaluation of the Artifact

The original problems described in 1.2 were seven-fold, and were addressed in the following way:

- **Workload Resilience and Fault Tolerance in HPC:** A problem which is typically addressed by the HPC community by using a combination of checkpointing and job scheduling⁸⁶ is now being directly addressed via the inclusion of Pachyderm. Because Pachyderm isolates each of the processes into their own container, and tracks each of the steps individually it can easily restart a failed step, or even a failed job, without having to restart the entire workflow. While this only works for the standard LCP workloads, its provenance features reduce the data loss, should a TCP workload fail.
- **Environment/Package Management in HPC:** Replacing the classical HPC package management solutions with a containerized approach, simplifies the deployment of code from the users' perspective massively, as they have almost complete control of the environment their code is going to run in, all the while giving the administrators the ease of mind that the code is not going to interfere with the rest of the system.
- **Probability issues with HPC:** Same goes for the probability issues, as the containerized approach allows for a much more fine-grained control of the environment, users can rapidly iterate and test their code on their local machines, before deploying it to the HPC system.
- **Scalability issues with HPC:** Scalability is one of the main features of k8s, and therefore of Pachyderm a cluster can easily be scaled up or down, depending on the current workload. Many cloud providers even offer hosted k8s clusters, which can be scaled up or down on demand, and therefore allow for a very flexible approach to the problem. While in classical HPC systems, the cluster is usually fixed in size, and therefore the user has to wait for the next available slot.
- **Interconnected Problem-Solving in CC** This one was one of the problems which was not directly addressed by Pachyderm or Kubernetes directly, in order to solve this problem, Arkouda was containerized and made usable for Pachyderm workloads. As of right now, the layers of network abstractions and the lack of OpenFAM support have a negative impact on the performance of the TCP workloads, but successfully proves the concept of interconnected problem-solving on a CC system.
- **Provenance and Versioning:** Combining the advantages the Pachyderm File System with the completely CI/CD based approach to the deployment of the workflows, allows a tracking of each and every part that goes into each and every step of the workflows.

While this project does not present a complete solution to all the problems, it does present a viable path forward for a more modern approach to HPC. The combination of HPC and

⁸⁶Jin et al. 2010

Pachyderm allows for a much more flexible approach to the problem and with future work and low level driver support and usability features like the Jenkins Pipeline, as well as a well maintained ecosystem of pipeline steps which can be used to build more complex workflows and reutilize existing code developed by fellow researchers, this approach could be a significant improvement over the current state of the art.

Unfortunately the time was cut short before the project could be fully completed and therefore some goals like the integration of OpenFAM, the switch to a low abstraction CNI and especially multi parameter performance testing could not be completed in time, it is apparent that the integration of these would bring the project much closer to the performance of the classical HPC systems, while still maintaining the flexibility and ease of use of the containerized approach, and therefore should be considered for future work.

5 Summary and Outlook

5.1 Summary

By way of this project paper we have assessed and described the state of the art for current technologies in the areas of containerized software, container orchestration and how those tie in with Software defined infrastructure. We have done the same for the state of the art in the solution of complex problems found in the field of and solved with HPC, namely the two larger classes of problems, LCP and TCP problems, and what strategies are usually employed to solve those problems.

We have then addressed the problems identified by the problem statement with the statement of an initial goal, and structured an intervention to solve those problems by way of the creation of a prototype. Before the iterative process of prototyping could begin we had to make a non-trivial decision on the choice of a container orchestrator/ workflow manager. For which we defined, discussed and weighted the selection criteria, and then evaluated the available options against those criteria, and We found our best fit through the employment of the SMART-ER method, which was the k8s-based orchestrator named Pachyderm.

The iterative process of prototyping was split into three main areas of focus, the infrastructure, the solution to TCPs and the integration of a complete CI/CD pipeline. Each of these areas was given at least two iterations and the results of each iteration were documented, evaluated and recommendations for future iterations were made.

The final results of the prototype were then evaluated against the initial goal and the problems identified in the problem statement, and the resulting artifact was found to be a valid form of intervention to solve the problems identified in the problem statement, while still being limited in scope and applicability, as was to be expected from a non production prototype.

5.2 Outlook

As already discussed, the limiting factor of this project was the available time, which left many avenues to still be explored both in respect to the prototype itself, and how it can be further extended to explore avenues not yet addressed by this project.

Of most importance in this case would be a further exploration of an efficient solution to TCP problems, as the current solution does technically work, but is lacking the efficiencies to be a viable solution for large scale problems. An initial starting point for this could be the network stack in between the containers and the underlying network infrastructure.

Appendix

Appendix Index

Appendix 1 Discussion of Tool Evaluation and Weighing	33
Appendix 1/1 Extensibility	33
Appendix 1/2 Community, Support & Docs	33
Appendix 1/3 License	34
Appendix 1/4 Strategic alignment	36
Appendix 1/5 Ease of Use	36
Appendix 1/6 Maturity	36
Appendix 1/7 Cost	36
Appendix 2 Diagrams	37
Appendix 2/1 Lookup table weighing functions	37
Appendix 2/2 Pipeline Communication Swim Lane Diagram	37
Appendix 2/3 Pipeline Communication Swim Lane Diagram	37
Appendix 3 Minikube installation instructions	38
Appendix 4 CI/CD installation instructions	45
Appendix 5 Kubernetes setup scripts	50
Appendix 5/1 Ansible setup script	50
Appendix 5/2 Flannel configuration	52
Appendix 5/3 Bash verification script	57
Appendix 5/4 Arkouda Setup	59
Appendix 5/5 Docker Registry Deployment	66
Appendix 5/6 Frogejo Deployment	69
Appendix 5/7 Jenkins Dockerfile	81

Appendix 1: Discussion of Tool Evaluation and Weighing

Appendix 1/1: Extensibility

Appendix 1/2: Community, Support & Docs

This section assesses the level of external support provided for each project. To evaluate this support, we will focus on three distinct aspects and combine them into a single score. Firstly, we will examine the size of the community, as a substantial community often indicates project maturity and the availability of extensive support. As proxies for community size, we will consider two central metrics: the number of stars on GitHub and the quantity of questions on Stack Overflow.

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

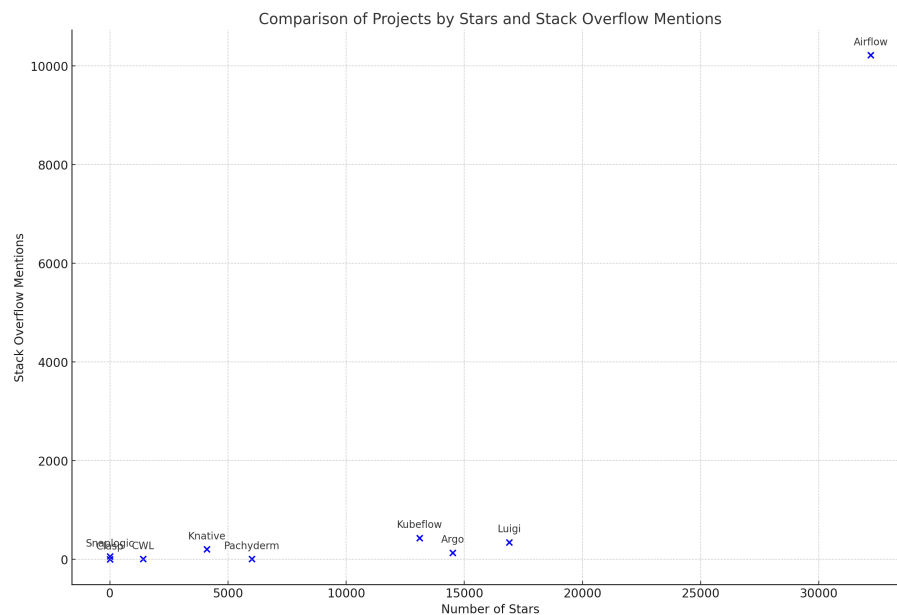


Abb. 10: Stars and Stack Overflow Questions Comparison

To gauge the level of support and community engagement surrounding these projects, we have

devised a composite score that normalizes and combines the GitHub stars and Stack Overflow questions metrics. The calculation of this score involves 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 scale of 0-10 and then taking their average.

Additionally, we acknowledge that some commercial tools, as well as certain open-source projects, offer enterprise support, reducing the reliance on the community for assistance. Similarly, projects developed in-house often have access to the original development team for support. Therefore, we will apply a flat bonus of 5 points to the scores of projects offering enterprise support and a flat bonus of 2.5 points to projects developed in-house.

$$S_i = \frac{1}{2} \left(\frac{x_i - \min(x)}{\max(x) - \min(x)} \times 10 + \frac{y_i - \min(y)}{\max(y) - \min(y)} \times 10 \right) + B_i$$

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 , along with the respective bonuses B_i , provide a comprehensive metric for comparing project popularity, community engagement, and the availability of additional support options, all on the same scale.

Project	Composite Score	Enterprise Bonus	Inhouse Bonus	Final Score
Airflow	10.00	0	0	10.00
Pachyderm	0.93	5	2.5	8.43
Snaplogic	0.03	5	0	5.03
Luigi	2.79	0	0	2.79
Clasp	0.00	0	2.5	2.5
Argo	2.32	0	0	2.32
Kubeflow	2.25	0	0	2.25
Knative	0.74	0	0	0.74
CWL	0.22	0	0	0.22

Tab. 5: Composite scores of Workflow managers, sorted by final score

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:

- **Ideal Situation (Score: 10):** This refers to cases where either the tool is in the public domain (and therefore not subject to copyright restrictions) or where our organization

possesses a direct ownership or significant influence over the licensing terms. This situation provides the most flexibility, allowing for extensive modification, redistribution, and proprietary use without concern for licensing infringements.

- **Permissive License (Score: 7.5):** Tools under licenses like MIT, BSD, or Apache 2.0 fall into this category. These licenses are highly permissive and generally allow for broad freedom, including modification, distribution, and private use, with minimal restrictions, often limited to liability and warranty.
- **Restrictive or Reciprocal Licenses (Score: 2.5):** Licenses such as the GPL or AGPL are more restrictive, requiring any changes to be open-sourced or contributions to be made back to the community. These “copyleft” licenses can be problematic in proprietary settings where modifications or integrations need to remain confidential.
- **Unacceptable Licenses (Score: 0):** This includes licenses that impose burdensome conditions or high costs, proprietary software where the source code is unavailable, or situations where the licensing terms make it impractical to use within our projects. For instance, licenses that mandate the purchase of additional software, restrict certain types of use, or pose potential legal risks would fall into this category.

Now we will evaluate the licenses of the tools in question, and assign them a score based on the above criteria.

- **Pachyderm** The licensing model of Pachyderm follows a model which has similarities with the "Open Core model"⁸⁷. Which means that while the core functionalities are published as the "COMMUNITY EDITION" with a permissive source-available License (Apache License 2.0)⁸⁸. 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** Argo’s adoption of the Apache License 2.0⁸⁹ aligns with common practices for open-source projects, affording users considerable freedom. This permissive license simplifies the use, modification, and redistribution of the software, an aspect that’s particularly beneficial for collaborative development or integration into proprietary software. Given our requirements and operational context, this offers us the flexibility needed for adaptation and potential enhancements without stringent restrictions, streamlining any developmental efforts we undertake with Argo.

⁸⁷ *Pachyderm -Pricing 2022*

⁸⁸ *Pachyderm/LICENSE at Master · Pachyderm/Pachyderm 2023*

⁸⁹ *Argo-Cd/LICENSE at Master · ArgoProj/Argo-Cd 2023*

- **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
- **Snaplogic** is an entirely commercial product which does not provide insight into nor the right to modify their Software⁹⁰. But as they might agree this is not a total knockout criterion for this entire project, but in regard to the licensing it will be weighted with 0.
- **Airflow** is licensed under the Apache License 2.0.⁹¹
- **Kubeflow** is licensed under the Apache License 2.0.⁹²
- **Knative** is licensed under the Apache License 2.0.⁹³
- **Luigi** is licensed under the Apache License 2.0.⁹⁴
- **CWL** is licensed under the Apache License 2.0.⁹⁵

Appendix 1/4: Strategic alignment

Appendix 1/5: Ease of Use

Appendix 1/6: Maturity

Appendix 1/7: Cost

This section aims to compare the relative cost of the products in relation to each other. We previously factored in the enterprise features, so when enterprise support is available and applicable we will take this into consideration. Here we have three categories of products first those which are completely free and without any enterprise support, secondly those which are free but offer enterprise support and lastly those which operate on a subscription basis.

⁹⁰ *SnapLogic – Master Subscription Agreement 2023*

⁹¹ *License — Airflow Documentation 2023*

⁹² *Kubeflow/LICENSE at Master · Kubeflow/Kubeflow 2023*

⁹³ *Knative Docs/LICENSE at Main · Knative/Docs 2023*

⁹⁴ *Luigi/LICENSE at Master · Spotify/Luigi 2023*

⁹⁵ *Cwl-Utils/LICENSE at Main · Common-Workflow-Language/Cwl-Utils 2023*

Appendix 2: Diagrams

Appendix 2/1: Lookup table weighing functions

Appendix 2/2: Pipeline Communication Swim Lane Diagram

Rank	Attributes								
	2	3	4	5	6	7	8	9	10
1	0.7500	0.6111	0.5208	0.4567	0.4083	0.3704	0.3397	0.3143	0.2929
2	0.2500	0.2778	0.2708	0.2567	0.2417	0.2276	0.2147	0.2032	0.1929
3		0.1111	0.1458	0.1567	0.1583	0.1561	0.1522	0.1477	0.1429
4			0.0625	0.0900	0.1028	0.1085	0.1106	0.1106	0.1096
5				0.0400	0.0611	0.0728	0.0793	0.0828	0.0846
6					0.0278	0.0442	0.0543	0.0606	0.0646
7						0.0204	0.0334	0.0421	0.0479
8							0.0156	0.0262	0.0336
9								0.0123	0.0211
10									0.0100

Abb. 11: ROC weights ⁹⁶

Rank	Attributes								
	2	3	4	5	6	7	8	9	10
1	0.6667	0.5455	0.4800	0.4379	0.4082	0.3857	0.3679	0.3535	0.3414
2	0.3333	0.2727	0.2400	0.2190	0.2041	0.1928	0.1840	0.1767	0.1707
3		0.1818	0.1600	0.1460	0.1361	0.1286	0.1226	0.1178	0.1138
4			0.1200	0.1095	0.1020	0.0964	0.0920	0.0884	0.0854
5				0.0876	0.0816	0.0771	0.0736	0.0707	0.0682
6					0.0680	0.0643	0.0613	0.0589	0.0569
7						0.0551	0.0525	0.0505	0.0488
8							0.0460	0.0442	0.0427
9								0.0393	0.0379
10									0.0341

Abb. 12: RR weights ⁹⁷

Appendix 2/3: Pipeline Communication Swim Lane Diagram

⁹⁶Taken from: Roberts/Goodwin 2002

⁹⁷Taken from: Roberts/Goodwin 2002

⁹⁸Taken from: Roberts/Goodwin 2002

⁹⁹Taken from: *Intro to Pipelines* 2023

Rank	Attributes								
	2	3	4	5	6	7	8	9	10
1	0.6667	0.5000	0.4000	0.3333	0.2857	0.2500	0.2222	0.2000	0.1818
2	0.3333	0.3333	0.3000	0.2667	0.2381	0.2143	0.1944	0.1778	0.1636
3		0.1667	0.2000	0.2000	0.1905	0.1786	0.1667	0.1556	0.1455
4			0.1000	0.1333	0.1429	0.1429	0.1389	0.1333	0.1273
5				0.0667	0.0952	0.1071	0.1111	0.1111	0.1091
6					0.0476	0.0714	0.0833	0.0889	0.0909
7						0.0357	0.0556	0.0667	0.0727
8							0.0278	0.0444	0.0545
9								0.0222	0.0364
10									0.0182

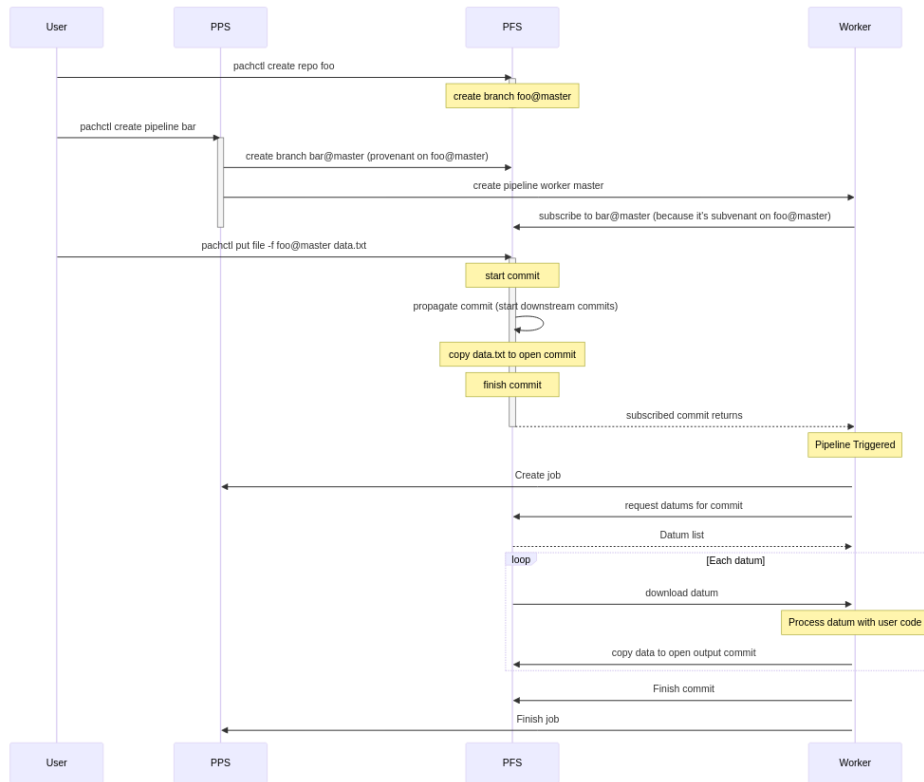
Abb. 13: RS weights ⁹⁸

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
20   ↪ equivalent file.
21
22
23 ### kubectl
24
25 Simply following the instructions on the [kubernetes
26   ↪ website](https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/)
27   ↪ should be sufficient.
28
29 But for the sake of completeness, here is what I did:
30
31 ```bash
32 curl -LO "https://dl.k8s.io/release/$(curl -L -s
33   ↪ https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
34 sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
35 ```
36
37 If the proxy is giving you grief one can simply download the binary elsewhere
38   ↪ and copy it to the target machine. (not recommended)

```

Abb. 14: Swim lane Diagram of the communication between the user and Pachyderm⁹⁹

```

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
    ↳ values file later. \
150 In this case it is 'manual'.
151
152 ### Installing [MinIO](https://min.io/docs/minio/linux/index.html)
153
154 We now install an S3 compatible storage system. Which one does not really
    ↳ 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:
242     ↪ https://docs.microsoft.com/en-us/azure/aks/concepts-storage#storage-classes
243     storageClass: manual
244 # storageSize specifies the size of the volume to use for postgresql
245 # Recommended Minimum Disk size for Microsoft/Azure: 256Gi - 1,100 IOPS
246     ↪ https://azure.microsoft.com/en-us/pricing/details/managed-disks/
247 ...
248 ```
249
250 and for the etcd at around 'L144'
251
252 ```yaml
253 ...
254 # GCP: https://cloud.google.com/compute/docs/disks/performance#disk_types
255 # Azure:
256     ↪ https://docs.microsoft.com/en-us/azure/aks/concepts-storage#storage-classes
257 #storageClass: manual
258 storageClassName: manual
259
260 # storageSize specifies the size of the volume to use for etcd.
261 # Recommended Minimum Disk size for Microsoft/Azure: 256Gi - 1,100 IOPS
262     ↪ https://azure.microsoft.com/en-us/pricing/details/managed-disks/
263 ...
264 ```
265
266 #### SSL Certificates
267
268 My setup refuses to work without SSL certificates, so I had to generate some.
269
270 ```bash
271 openssl genrsa -out <CertName>.key 2048
272 openssl req -new -x509 -sha256 -key <CertName>.key -out <CertName>.crt
273
274 kubectl create secret tls <SecretName> --cert=<CertName>.crt
275     ↪ --key=<CertName>.key
276 ```
277
278 We then edit the 'values.yaml' file at around 'L683' to use the certificates.
279
280 ```yaml
281 ...
282     tls:
283         enabled: true
284         secretName: "<SecretName>"
285         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
293     ↪ https://github.com/pachyderm/pachyderm/releases/download/v2.6.5/pachctl_2.6.5_amd64.deb
294     ↪ && sudo dpkg -i /tmp/pachctl.deb
295 '''
296
297 ### Deploy
298
299 Now that the values file is ready we can install Pachyderm.
300
301 '''bash
302 helm install pachyderm pachyderm/pachyderm \
303     -f ./values.yml pachyderm/pachyderm \
304     --set postgresql.volumePermissions.enabled=true \
305     --set deployTarget=LOCAL \
306     --set proxy.enabled=true \
307     --set proxy.service.type=NodePort \
308     --set proxy.host=localhost \
309     --set proxy.service.httpPort=8080
310 '''
311
312 Now you might want to connect to the dashboard. This can be done by
313     ↪ port-forwarding the service.
314
315 '''bash
316 pachctl port-forward
317 '''
318
319 :tada: Now we should be able to access the dashboard at 'http://localhost:4000'
320     ↪ :tada:
```

Appendix 4: CI/CD installation instructions

```
1 # Setting up the GitOps CI/CD
2
3 This part is concerned with setting up both the Version Control System Forgejo
4     ↪ and the CI/CD system Jenkins. \
5
6 While Forgejo is a fork of Gitea, it is still sparsely documented and thus we
7     ↪ will not setup the runner system of Forgejo, but instead use Jenkins for
8     ↪ CI/CD.
9
10
```

```
6
7 ## Namespaces
8
9 As always we create namespaces to keep things clean:
10
11 ```bash
12 kubectl create namespace forgejo
13 kubectl create namespace jenkins
14 ```
15
16 ## Persistent Volumes
17
18 We need to create a location for the persistent volume:
19
20 ```bash
21 mkdir -p /mnt/forgejo/postgres
22 mkdir -p /mnt/forgejo/zero
23 mkdir -p /mnt/jenkins
24 sudo chown -R eckerth:users /mnt/forgejo/ /mnt/jenkins
25 ```
26
27 We then create the persistent volumes for Forgejo ....:
28
29 ``` yml
30 apiVersion: storage.k8s.io/v1
31 kind: StorageClass
32 metadata:
33   name: forgejo
34 provisioner: kubernetes.io/no-provisioner
35 volumeBindingMode: WaitForFirstConsumer
36 ---
37 apiVersion: v1
38 kind: PersistentVolume
39 metadata:
40   name: forgejo-postgres
41   labels:
42     type: local
43 spec:
44   capacity:
45     storage: 10Gi
46   accessModes:
47     - ReadWriteOnce
48   storageClassName: forgejo
49   local:
50     path: /mnt/forgejo/postgres
51   nodeAffinity:
52     required:
53       nodeSelectorTerms:
54         - matchExpressions:
55           - key: kubernetes.io/hostname
56             operator: In
```

```
57         values:
58         - heydar20.labs.hpecorp.net
59 ---
60 apiVersion: v1
61 kind: PersistentVolume
62 metadata:
63   name: forgejo-0
64   labels:
65     type: local
66 spec:
67   capacity:
68     storage: 10Gi
69   accessModes:
70     - ReadWriteOnce
71   storageClassName: forgejo
72   local:
73     path: /mnt/forgejo/zero
74   nodeAffinity:
75     required:
76       nodeSelectorTerms:
77       - matchExpressions:
78         - key: kubernetes.io/hostname
79           operator: In
80           values:
81             - heydar20.labs.hpecorp.net
82
83   '''
84
85   ... and for Jenkins:
86
87   '''yaml
88 apiVersion: storage.k8s.io/v1
89 kind: StorageClass
90 metadata:
91   name: jenkins
92 provisioner: kubernetes.io/no-provisioner
93 volumeBindingMode: WaitForFirstConsumer
94
95 ---
96 apiVersion: v1
97 kind: PersistentVolume
98 metadata:
99   name: jenkins
100  labels:
101    type: local
102 spec:
103   capacity:
104     storage: 10Gi
105   accessModes:
106     - ReadWriteOnce
107   storageClassName: jenkins
```

```
108   local:
109     path: /mnt/jenkins
110   nodeAffinity:
111     required:
112       nodeSelectorTerms:
113       - matchExpressions:
114         - key: kubernetes.io/hostname
115           operator: In
116           values:
117         - heydar20.labs.hpecorp.net
118   ```
119
120   ```bash
121   kubectl -n forgejo apply -f ./forgejo/volumes.yaml
122   kubectl -n jenkins apply -f ./jenkins/volumes.yaml
123   ```
124
125   ## Installation
126
127   After these are applied we can simply install the helm chart:
128
129   ```bash
130   helm repo add jenkins https://charts.jenkins.io
131   helm repo update
132   helm install -n jenkins jenkins jenkins/jenkins -f ./jenkins/values.yaml
133   helm install -n forgejo forgejo oci://codeberg.org/forgejo-contrib/forgejo -f
134     ↪ ./forgejo/values.yaml
135   ```
136
137   ## Configuring
138
139   In order to connect both Jenkins and Forgejo we will have to adjust some
140     ↪ configurations.
141
142   1. As we want Jenkins to be able to be able spawn pods on the cluster, we will
143     ↪ need to give it the needed permissions.
144   For this one can use the service_account.yaml file in the jenkins folder.
145
146   ```bash
147   kubectl -n jenkins apply -f ./jenkins/service_account.yaml
148   ```
149
150   2. We add the following config map to Forgejo in order to allow it to send
151     ↪ webhooks to out jenkins host.
152
153   ```yaml
154   additionalConfigSources:
155     - configMap:
156       name: gitea-app-ini
157   ```
```



```
155 ```yaml
156 apiVersion: v1
157 kind: ConfigMap
158 metadata:
159   name: gitea-app-ini
160 data:
161   webhook: |
162     ALLOWED_HOST_LIST=<jenkins server>
163 ```
164
165 ```bash
166 kubectl -n forgejo apply -f ./forgejo/configmap.yaml
167 ```
168
169 2. We then have to go into the Forgejo admin pannel and enable the system wide
    ↪ Webhooks,
```

Appendix 5: Kubernetes setup scripts

Appendix 5/1: 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 '/\s/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 5/2: 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 5/3: 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 5/4: 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  '''

```

Appendix 5/5: Docker Registry Deployment

```
1
2 apiVersion: storage.k8s.io/v1
3 kind: StorageClass
4 metadata:
5   name: pachyderm
6 provisioner: kubernetes.io/no-provisioner
7 volumeBindingMode: WaitForFirstConsumer
8 ---
9 apiVersion: v1
10 kind: PersistentVolume
11 metadata:
12   name: registry-pv
13   labels:
14     type: local
15 spec:
16   capacity:
17     storage: 10Gi
18   accessModes:
19     - ReadWriteOnce
20   storageClassName: registry
21   local:
22     path: /mnt/registry/
23   nodeAffinity:
24     required:
25       nodeSelectorTerms:
26         - matchExpressions:
27           - key: kubernetes.io/hostname
28             operator: In
29             values:
30               - heydar20.labs.hpecorp.net
31 ---
32 apiVersion: v1
33 kind: PersistentVolumeClaim
34 metadata:
35   name: registry-pvc
36 spec:
37   accessModes:
38     - ReadWriteOnce
39   resources:
40     requests:
41       storage: 10Gi
42   storageClassName: registry
43   selector:
44     matchLabels:
45       type: local
46 ---
47 apiVersion: apps/v1
48 kind: Deployment
49 metadata:
```

```
50   name: cluster-registry
51   labels:
52     app: cluster-registry
53 spec:
54   replicas: 1
55   selector:
56     matchLabels:
57       app: cluster-registry
58   template:
59     metadata:
60       labels:
61         app: cluster-registry
62     spec:
63       volumes:
64         - name: certs-vol
65           hostPath:
66             path: /mnt/registry/certs
67             type: Directory
68         - name: registry-vol
69           persistentVolumeClaim:
70             claimName: registry-pvc
71
72     containers:
73       - image: registry:2
74         name: cluster-registry
75         imagePullPolicy: IfNotPresent
76         env:
77           - name: REGISTRY_HTTP_TLS_CERTIFICATE
78             value: "/certs/registry.crt"
79           - name: REGISTRY_HTTP_TLS_KEY
80             value: "/certs/registry.key"
81         ports:
82           - containerPort: 5000
83         volumeMounts:
84           - name: certs-vol
85             mountPath: /certs
86           - name: registry-vol
87             mountPath: /var/lib/registry
88 ---
89 apiVersion: v1
90 kind: Service
91 metadata:
92   labels:
93     app: cluster-registry
94   name: cluster-registry
95 spec:
96   ports:
97     - port: 5000
98       nodePort: 31320
99       protocol: TCP
100     targetPort: 5000
```

```
101 selector:
102     app: cluster-registry
103     type: NodePort
```

Appendix 5/6: Forgejo Deployment

```
1 # Default values for gitea.
2 # This is a YAML-formatted file.
3 # Declare variables to be passed into your templates.
4 ## @section Global
5 #
6 ## @param global.imageRegistry global image registry override
7 ## @param global.imagePullSecrets global image pull secrets override; can be
8   ↪ extended by 'imagePullSecrets'
9 ## @param global.storageClass global storage class override
10 ## @param global.hostAliases global hostAliases which will be added to the
11   ↪ pod's hosts files
12 global:
13   imageRegistry: ""
14   ## E.g.
15   ## imagePullSecrets:
16   ##   - myRegistryKeySecretName
17   ##
18   imagePullSecrets: []
19   storageClass: "forgejo"
20   hostAliases: []
21   # - ip: 192.168.137.2
22   #   hostnames:
23   #     - example.com
24
25 ## @param replicaCount number of replicas for the statefulset
26 replicaCount: 1
27
28 ## @param clusterDomain cluster domain
29 clusterDomain: cluster.local
30
31 ## @section Image
32 ## @param image.registry image registry, e.g. gcr.io,docker.io
33 ## @param image.repository Image to start for this pod
34 ## @param image.tag Visit: [Image
35   ↪ tag](https://codeberg.org/forgejo/-/packages/container/forgejo/versions).
36   ↪ Defaults to 'appVersion' within Chart.yaml.
37 ## @param image.pullPolicy Image pull policy
38 ## @param image.rootless Whether or not to pull the rootless version of Forgejo,
39   ↪ only works on Forgejo 1.14.x or higher
40 image:
41   registry: "codeberg.org"
42   repository: forgejo/forgejo
43   # Overrides the image tag whose default is the chart appVersion.
44   tag: ""
45   pullPolicy: Always
46   rootless: false # only possible when running 1.14 or later
47
48 ## @param imagePullSecrets Secret to use for pulling the image
49 imagePullSecrets: []
```

```
45
46 ## @section Security
47 # Security context is only usable with rootless image due to image design
48 ## @param podSecurityContext.fsGroup Set the shared file system group for all
    ↪ containers in the pod.
49 podSecurityContext:
50   fsGroup: 1000
51
52 ## @param containerSecurityContext Security context
53 containerSecurityContext: {}
54 #   allowPrivilegeEscalation: false
55 #   capabilities:
56 #     drop:
57 #       - ALL
58 #   # Add the SYS_CHROOT capability for root and rootless images if you intend
    ↪ to
59 #   # run pods on nodes that use the container runtime cri-o. Otherwise, you
    ↪ will
60 #   # get an error message from the SSH server that it is not possible to read
    ↪ from
61 #   # the repository.
62 #   # https://gitea.com/gitea/helm-chart/issues/161
63 #   add:
64 #     - SYS_CHROOT
65 #   privileged: false
66 #   readOnlyRootFilesystem: true
67 #   runAsGroup: 1000
68 #   runAsNonRoot: true
69 #   runAsUser: 1000
70
71 ## @deprecated The securityContext variable has been split two:
72 ## - containerSecurityContext
73 ## - podSecurityContext.
74 ## @param securityContext Run init and Forgejo containers as a specific
    ↪ securityContext
75 securityContext: {}
76
77 ## @section Service
78 service:
79   ## @param service.http.type Kubernetes service type for web traffic
80   ## @param service.http.port Port number for web traffic
81   ## @param service.http.clusterIP ClusterIP setting for http autosetup for
    ↪ statefulset is None
82   ## @param service.http.loadBalancerIP LoadBalancer IP setting
83   ## @param service.http.nodePort NodePort for http service
84   ## @param service.http.externalTrafficPolicy If 'service.http.type' is
    ↪ 'NodePort' or 'LoadBalancer', set this to 'Local' to enable source IP
    ↪ preservation
85   ## @param service.http.externalIPs External IPs for service
86   ## @param service.http.ipFamilyPolicy HTTP service dual-stack policy
87   ## @param service.http.ipFamilies HTTP service dual-stack familiy
```

```
    ↪ selection,for dual-stack parameters see official kubernetes
    ↪ [dual-stack concept
    ↪ documentation](https://kubernetes.io/docs/concepts/services-networking/dual-stack)
88 ## @param service.http.loadBalancerSourceRanges Source range filter for http
    ↪ loadbalancer
89 ## @param service.http.annotations HTTP service annotations
90 http:
91   type: NodePort
92   port: 3000
93   clusterIP: None
94   loadBalancerIP:
95   nodePort: 30070
96   externalTrafficPolicy: Local
97   externalIPs:
98   ipFamilyPolicy:
99   ipFamilies:
100  loadBalancerSourceRanges: []
101  annotations: {}
102 ## @param service.ssh.type Kubernetes service type for ssh traffic
103 ## @param service.ssh.port Port number for ssh traffic
104 ## @param service.ssh.clusterIP ClusterIP setting for ssh autosetup for
    ↪ statefulset is None
105 ## @param service.ssh.loadBalancerIP LoadBalancer IP setting
106 ## @param service.ssh.nodePort NodePort for ssh service
107 ## @param service.ssh.externalTrafficPolicy If 'service.ssh.type' is
    ↪ 'NodePort' or 'LoadBalancer', set this to 'Local' to enable source IP
    ↪ preservation
108 ## @param service.ssh.externalIPs External IPs for service
109 ## @param service.ssh.ipFamilyPolicy SSH service dual-stack policy
110 ## @param service.ssh.ipFamilies SSH service dual-stack familiy selection,for
    ↪ dual-stack parameters see official kubernetes [dual-stack concept
    ↪ documentation](https://kubernetes.io/docs/concepts/services-networking/dual-stack)
111 ## @param service.ssh.hostPort HostPort for ssh service
112 ## @param service.ssh.loadBalancerSourceRanges Source range filter for ssh
    ↪ loadbalancer
113 ## @param service.ssh.annotations SSH service annotations
114 ssh:
115   type: NodePort
116   port: 22
117   clusterIP: None
118   loadBalancerIP:
119   nodePort: 30071
120   externalTrafficPolicy: Local
121   externalIPs:
122   ipFamilyPolicy:
123   ipFamilies:
124   hostPort:
125   loadBalancerSourceRanges: []
126   annotations: {}
127
128 ## @section Ingress
```

```
129 ## @param ingress.enabled Enable ingress
130 ## @param ingress.className Ingress class name
131 ## @param ingress.annotations Ingress annotations
132 ## @param ingress.hosts[0].host Default Ingress host
133 ## @param ingress.hosts[0].paths[0].path Default Ingress path
134 ## @param ingress.hosts[0].paths[0].pathType Ingress path type
135 ## @param ingress.tls Ingress tls settings
136 ## @extra ingress.apiVersion Specify APIVersion of ingress object. Mostly would
    ↪ only be used for argocd.
137 ingress:
138   enabled: false
139   # className: nginx
140   className:
141   annotations:
142     {}
143   # kubernetes.io/ingress.class: nginx
144   # kubernetes.io/tls-acme: "true"
145   hosts:
146     - host: heydar20.labs.hpecorp.net:30070
147       paths:
148         - path: /
149           pathType: Prefix
150   tls: []
151   # - secretName: chart-example-tls
152   #   hosts:
153   #     - git.example.com
154   # Mostly for argocd or any other CI that uses 'helm template | kubectl apply'
    ↪ or similar
155   # If helm doesn't correctly detect your ingress API version you can set it
    ↪ here.
156   # apiVersion: networking.k8s.io/v1
157
158 ## @section StatefulSet
159 #
160 ## @param resources Kubernetes resources
161 resources:
162   {}
163   # We usually recommend not to specify default resources and to leave this as
    ↪ a conscious
164   # choice for the user. This also increases chances charts run on environments
    ↪ with little
165   # resources, such as Minikube. If you do want to specify resources, uncomment
    ↪ the following
166   # lines, adjust them as necessary, and remove the curly braces after
    ↪ 'resources:'.
167   # limits:
168   #   cpu: 100m
169   #   memory: 128Mi
170   # requests:
171   #   cpu: 100m
172   #   memory: 128Mi
```



```
173
174 ## Use an alternate scheduler, e.g. "stork".
175 ## ref:
176     ↪ https://kubernetes.io/docs/tasks/administer-cluster/configure-multiple-schedulers/
177 ##
178 ## @param schedulerName Use an alternate scheduler, e.g. "stork"
179 schedulerName: ""
180
181 ## @param nodeSelector NodeSelector for the statefulset
182 nodeSelector: {}
183
184 ## @param tolerations Tolerations for the statefulset
185 tolerations: []
186
187 ## @param affinity Affinity for the statefulset
188 affinity: {}
189
190 ## @param dnsConfig dnsConfig for the statefulset
191 dnsConfig: {}
192
193 ## @param priorityClassName priorityClassName for the statefulset
194 priorityClassName: ""
195
196 ## @param statefulset.env Additional environment variables to pass to
197     ↪ containers
198 ## @param statefulset.terminationGracePeriodSeconds How long to wait until
199     ↪ forcefully kill the pod
200 ## @param statefulset.labels Labels for the statefulset
201 ## @param statefulset.annotations Annotations for the Forgejo StatefulSet to be
202     ↪ created
203 statefulset:
204   env:
205     []
206     # - name: VARIABLE
207     #   value: my-value
208   terminationGracePeriodSeconds: 60
209   labels: {}
210   annotations: {}
211
212 ## @section Persistence
213 #
214 ## @param persistence.enabled Enable persistent storage
215 ## @param persistence.existingClaim Use an existing claim to store repository
216     ↪ information
217 ## @param persistence.size Size for persistence to store repo information
218 ## @param persistence.accessModes AccessMode for persistence
219 ## @param persistence.labels Labels for the persistence volume claim to be
220     ↪ created
221 ## @param persistence.annotations Annotations for the persistence volume claim
222     ↪ to be created
223 ## @param persistence.storageClass Name of the storage class to use
```

```
217 ## @param persistence.subPath Subdirectory of the volume to mount at
218 persistence:
219   enabled: true
220   existingClaim:
221   size: 10Gi
222   accessModes:
223     - ReadWriteOnce
224   labels: {}
225   annotations: {}
226   storageClass: "forgejo"
227   subPath:
228
229 ## @param extraVolumes Additional volumes to mount to the Forgejo statefulset
230 extraVolumes: []
231 # - name: postgres-ssl-vol
232 #   secret:
233 #     secretName: gitea-postgres-ssl
234
235 ## @param extraContainerVolumeMounts Mounts that are only mapped into the
236   ↪ Forgejo runtime/main container, to e.g. override custom templates.
237 extraContainerVolumeMounts: []
238
239 ## @param extraInitVolumeMounts Mounts that are only mapped into the
240   ↪ init-containers. Can be used for additional preconfiguration.
241 extraInitVolumeMounts: []
242
243 ## @deprecated The extraVolumeMounts variable has been split two:
244 ## - extraContainerVolumeMounts
245 ## - extraInitVolumeMounts
246 ## As an example, can be used to mount a client cert when connecting to an
247   ↪ external Postgres server.
248
249 ## @param extraVolumeMounts **DEPRECATED** Additional volume mounts for init
250   ↪ containers and the Forgejo main container
251 extraVolumeMounts: []
252 # - name: postgres-ssl-vol
253 #   readOnly: true
254 #   mountPath: "/pg-ssl"
255
256 ## @section Init
257 ## @param initPreScript Bash shell script copied verbatim to the start of the
258   ↪ init-container.
259 initPreScript: ""
260 #
261 # initPreScript: |
262 #   mkdir -p /data/git/.postgresql
263 #   cp /pg-ssl/* /data/git/.postgresql/
264 #   chown -R git:git /data/git/.postgresql/
265 #   chmod 400 /data/git/.postgresql/postgresql.key
266
267 ## @param initContainers.resources.limits initContainers.limits Kubernetes
268   ↪ resource limits for init containers
```

```
262 ## @param initContainers.resources.requests.cpu initContainers.requests.cpu
    ↳ Kubernetes cpu resource limits for init containers
263 ## @param initContainers.resources.requests.memory
    ↳ initContainers.requests.memory Kubernetes memory resource limits for
    ↳ init containers
264 initContainers:
265   resources:
266     limits: {}
267     requests:
268       cpu: 100m
269       memory: 128Mi
270
271 # Configure commit/action signing prerequisites
272 ## @section Signing
273 #
274 ## @param signing.enabled Enable commit/action signing
275 ## @param signing.gpgHome GPG home directory
276 ## @param signing.privateKey Inline private gpg key for signed Forgejo actions
277 ## @param signing.existingSecret Use an existing secret to store the value of
    ↳ 'signing.privateKey'
278 signing:
279   enabled: false
280   gpgHome: /data/git/.gnupg
281   privateKey: ""
282   # privateKey: |-
283   #   -----BEGIN PGP PRIVATE KEY BLOCK-----
284   #   ...
285   #   -----END PGP PRIVATE KEY BLOCK-----
286   existingSecret: ""
287
288 ## @section Gitea
289 #
290 gitea:
291   ## @param gitea.admin.username Username for the Forgejo admin user
292   ## @param gitea.admin.existingSecret Use an existing secret to store admin
    ↳ user credentials
293   ## @param gitea.admin.password Password for the Forgejo admin user
294   ## @param gitea.admin.email Email for the Forgejo admin user
295   admin:
296     # existingSecret: gitea-admin-secret
297     existingSecret:
298     username: NoName
299     password: NoPassword
300     email: "gitea@local.domain"
301
302   ## @param gitea.metrics.enabled Enable Forgejo metrics
303   ## @param gitea.metrics.serviceMonitor.enabled Enable Forgejo metrics service
    ↳ monitor
304   metrics:
305     enabled: false
306     serviceMonitor:
```

```
307     enabled: false
308     #   additionalLabels:
309     #     prometheus-release: prom1
310
311 ## @param gitea.ldap LDAP configuration
312 ldap:
313     []
314     # - name: "LDAP 1"
315     #   existingSecret:
316     #     securityProtocol:
317     #   host:
318     #   port:
319     #   userSearchBase:
320     #   userFilter:
321     #   adminFilter:
322     #   emailAttribute:
323     #   bindDn:
324     #   bindPassword:
325     #   usernameAttribute:
326     #   publicSSHKeyAttribute:
327
328 # Either specify inline 'key' and 'secret' or refer to them via
329 ↪ 'existingSecret'
330 ## @param gitea.oauth OAuth configuration
331 oauth:
332     []
333     # - name: 'OAuth 1'
334     #   provider:
335     #   key:
336     #   secret:
337     #   existingSecret:
338     #   autoDiscoverUrl:
339     #   useCustomUrls:
340     #   customAuthUrl:
341     #   customTokenUrl:
342     #   customProfileUrl:
343     #   customEmailUrl:
344
345 ## @param gitea.config Configuration for the Forgejo server, ref:
346 ↪ [config-cheat-sheet](https://docs.gitea.io/en-us/config-cheat-sheet/)
347 config: {}
348 # APP_NAME: "Forgejo: Git with a cup of tea"
349 # RUN_MODE: dev
350 #
351 # server:
352 #   SSH_PORT: 22
353 #
354 # security:
355 #   PASSWORD_COMPLEXITY: spec
356
357 ## @param gitea.additionalConfigSources Additional configuration from secret
```

```
    ↪ or configmap
356 additionalConfigSources:
357     - configMap:
358         name: gitea-app-ini
359     # - secret:
360     #     secretName: gitea-app-ini-oauth
361
362
363
364 ## @param gitea.additionalConfigFromEnvs Additional configuration sources
    ↪ from environment variables
365 additionalConfigFromEnvs:
366
367 ## @param gitea.podAnnotations Annotations for the Forgejo pod
368 podAnnotations: {}
369
370 ## @param gitea.ssh.logLevel Configure OpenSSH's log level. Only available
    ↪ for root-based Forgejo image.
371 ssh:
372     logLevel: "INFO"
373
374 ## @section LivenessProbe
375 #
376 ## @param gitea.livenessProbe.enabled Enable liveness probe
377 ## @param gitea.livenessProbe.tcpSocket.port Port to probe for liveness
378 ## @param gitea.livenessProbe.initialDelaySeconds Initial delay before
    ↪ liveness probe is initiated
379 ## @param gitea.livenessProbe.timeoutSeconds Timeout for liveness probe
380 ## @param gitea.livenessProbe.periodSeconds Period for liveness probe
381 ## @param gitea.livenessProbe.successThreshold Success threshold for liveness
    ↪ probe
382 ## @param gitea.livenessProbe.failureThreshold Failure threshold for liveness
    ↪ probe
383 # Modify the liveness probe for your needs or completely disable it by
    ↪ commenting out.
384 livenessProbe:
385     enabled: true
386     tcpSocket:
387         port: http
388     initialDelaySeconds: 200
389     timeoutSeconds: 1
390     periodSeconds: 10
391     successThreshold: 1
392     failureThreshold: 10
393
394 ## @section ReadinessProbe
395 #
396 ## @param gitea.readinessProbe.enabled Enable readiness probe
397 ## @param gitea.readinessProbe.tcpSocket.port Port to probe for readiness
398 ## @param gitea.readinessProbe.initialDelaySeconds Initial delay before
    ↪ readiness probe is initiated
```

```
399  ## @param gitea.readinessProbe.timeoutSeconds Timeout for readiness probe
400  ## @param gitea.readinessProbe.periodSeconds Period for readiness probe
401  ## @param gitea.readinessProbe.successThreshold Success threshold for
    ↪ readiness probe
402  ## @param gitea.readinessProbe.failureThreshold Failure threshold for
    ↪ readiness probe
403  # Modify the readiness probe for your needs or completely disable it by
    ↪ commenting out.
404  readinessProbe:
405    enabled: true
406    tcpSocket:
407      port: http
408    initialDelaySeconds: 5
409    timeoutSeconds: 1
410    periodSeconds: 10
411    successThreshold: 1
412    failureThreshold: 3
413
414  # # Uncomment the startup probe to enable and modify it for your needs.
415  ## @section StartupProbe
416  #
417  ## @param gitea.startupProbe.enabled Enable startup probe
418  ## @param gitea.startupProbe.tcpSocket.port Port to probe for startup
419  ## @param gitea.startupProbe.initialDelaySeconds Initial delay before startup
    ↪ probe is initiated
420  ## @param gitea.startupProbe.timeoutSeconds Timeout for startup probe
421  ## @param gitea.startupProbe.periodSeconds Period for startup probe
422  ## @param gitea.startupProbe.successThreshold Success threshold for startup
    ↪ probe
423  ## @param gitea.startupProbe.failureThreshold Failure threshold for startup
    ↪ probe
424  startupProbe:
425    enabled: false
426    tcpSocket:
427      port: http
428    initialDelaySeconds: 60
429    timeoutSeconds: 1
430    periodSeconds: 10
431    successThreshold: 1
432    failureThreshold: 10
433
434  ## @section Memcached
435  ## @descriptionStart
436  ## Memcached is loaded as a dependency from
    ↪ [Bitnami](https://github.com/bitnami/charts/tree/master/bitnami/memcached)
    ↪ if enabled in the values. Complete Configuration can be taken from their
    ↪ website.
437  ## @descriptionEnd
438  #
439  ## @param memcached.enabled Memcached is loaded as a dependency from
    ↪ [Bitnami](https://github.com/bitnami/charts/tree/master/bitnami/memcached)
```

```
    ↪ if enabled in the values. Complete Configuration can be taken from their
    ↪ website.
440 ## ref: https://hub.docker.com/r/bitnami/memcached/tags/
441 ## @param memcached.service.ports.memcached Port for Memcached
442 memcached:
443   enabled: true
444   # image:
445   #   registry: docker.io
446   #   repository: bitnami/memcached
447   #   tag: ""
448   #   digest: ""
449   #   pullPolicy: IfNotPresent
450   #   pullSecrets: []
451   service:
452     ports:
453       memcached: 11211
454
455 ## @section PostgreSQL
456 ## @descriptionStart
457 ## PostgreSQL is loaded as a dependency from
458   ↪ [Bitnami](https://github.com/bitnami/charts/tree/master/bitnami/postgresql)
459   ↪ if enabled in the values. Complete Configuration can be taken from their
460   ↪ website.
461 ## @descriptionEnd
462 #
463 ## @param postgresql.enabled Enable PostgreSQL
464 ## @param postgresql.global.postgresql.auth.password Password for the 'gitea'
465   ↪ user (overrides 'auth.password')
466 ## @param postgresql.global.postgresql.auth.database Name for a custom database
467   ↪ to create (overrides 'auth.database')
468 ## @param postgresql.global.postgresql.auth.username Name for a custom user to
469   ↪ create (overrides 'auth.username')
470 ## @param postgresql.global.postgresql.service.ports.postgresql PostgreSQL
471   ↪ service port (overrides 'service.ports.postgresql')
472 ## @param postgresql.primary.persistence.size PVC Storage Request for
473   ↪ PostgreSQL volume
474 postgresql:
475   enabled: true
476   global:
477     postgresql:
478       auth:
479         password: NoPassword
480         database: NoDB
481         username: NoUser
482       service:
483         ports:
484           postgresql: 5432
485   primary:
486     persistence:
487       size: 10Gi
488
```

```
481 # By default, removed or moved settings that still remain in a user defined
    ↪ values.yaml will cause Helm to fail running the install/update.
482 # Set it to false to skip this basic validation check.
483 ## @section Advanced
484 ## @param checkDeprecation Set it to false to skip this basic validation check.
485 ## @param test.enabled Set it to false to disable test-connection Pod.
486 ## @param test.image.name Image name for the wget container used in the
    ↪ test-connection Pod.
487 ## @param test.image.tag Image tag for the wget container used in the
    ↪ test-connection Pod.
488 checkDeprecation: true
489 test:
490   enabled: true
491   image:
492     name: busybox
493     tag: latest
494
495 ## @param extraDeploy Array of extra objects to deploy with the release
496 ##
497 extraDeploy: []

1 apiVersion: v1
2 kind: ConfigMap
3 metadata:
4   name: gitea-app-ini
5 data:
6   webhook: |
7     ALLOWED_HOST_LIST=*
```


Appendix 5/7: Jenkins Dockerfile

```
1 FROM jenkins/jenkins:lts-jdk11
2 USER root
3
4 ARG HTTP_PROXY
5 # HTTP_PROXY="http://proxy.its.hpecorp.net:80" \
6 ARG NO_PROXY
7 #
   ↪ NO_PROXY="localhost,cluster.local,.cluster.local,.labs.hpecorp.net,127.0.0.1,192.1
   ↪ .
8
9
10 ENV HTTP_PROXY=$HTTP_PROXY
11 ENV HTTPS_PROXY=$HTTP_PROXY
12 ENV http_proxy=$HTTP_PROXY
13 ENV https_proxy=$HTTP_PROXY
14 ENV NO_PROXY=$NO_PROXY
15 ENV DEBUG_BUILD_OPTION=$DEBUG
16
17
18 RUN apt-get update && apt-get install -y \
19     apt-transport-https \
20     ca-certificates \
21     curl \
22     tree \
23     gnupg \
24     software-properties-common \
25     python3-pip \
26     && rm -rf /var/lib/apt/lists/*
27
28
29 RUN curl -o /tmp/pachctl.deb -L
   ↪ https://github.com/pachyderm/pachyderm/releases/download/v2.6.5/pachctl_2.6.5_amd6
   ↪ && dpkg -i /tmp/pachctl.deb
30
31 COPY ./requirements.txt ./requirements.txt
32
33 # gitdb==4.0.10
34 # GitPython==3.1.32
35 # Pygments==2.16.1
36 # smmap==5.0.0
37 # gitignore-parser==0.1.6
38 # python-pachyderm==7.6.0
39 # pyyaml==5.4.1
40
41 RUN pip install -r ./requirements.txt
42 RUN rm ./requirements.txt
43
44 USER jenkins
```

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 Github 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).
- Baur, D./Seybold, D./Griesinger, F./Tsitsipas, A./Hauser, C. B./Domaschka, J. (2015):** Cloud Orchestration Features: Are Tools Fit for Purpose? In: *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*. 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), pp. 95–101. DOI: 10.1109/UCC.2015.25. URL: <https://ieeexplore.ieee.org/abstract/document/7431400> (retrieval: 11/05/2023).
- 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).
- Byrne, J./Kuno, H./Ghosh, C./Shome, P. (2023):** Coupling Chapel-Powered HPC Workflows for Python. In.
- 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).
- Chou, C. C./Chen, Y./Milojicic, D./Reddy, N./Gratz, P. (2019): Optimizing Post-Copy Live Migration with System-Level Checkpoint Using Fabric-Attached Memory. In: *2019 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC)*. 2019 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC). Denver, CO, USA: IEEE, pp. 16–24. ISBN: 978-1-72816-007-8. DOI: 10.1109/MCHPC49590.2019.00010. URL: <https://ieeexplore.ieee.org/document/8946189/> (retrieval: 10/29/2023).
- Cluster Networking (2023). Kubernetes. URL: <https://kubernetes.io/docs/concepts/cluster-administration/networking/> (retrieval: 10/18/2023).
- Codeberg (2023): Codeberg.Org. Codeberg.org. URL: <https://codeberg.org/> (retrieval: 10/31/2023).
- 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).
- Docker-Registry 1.10.0 · Phntom/Phntom (2023). URL: <https://artifacthub.io/packages/helm/phntom/docker-registry> (retrieval: 10/30/2023).
- 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/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).
- forgejo (2023)**: Forgejo. Codeberg.org. URL: <https://codeberg.org/forgejo/forgejo> (retrieval: 10/30/2023).
- Forgejo 0.13.0 · Forgejo/Forgejo-Helm (2023). URL: <https://artifacthub.io/packages/helm/forgejo-helm/forgejo> (retrieval: 10/31/2023).
- 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).
- Gitea - Git with a Cup of Tea* (2023). Gitea. URL: <https://github.com/go-gitea/gitea> (retrieval: 10/31/2023).
- Gluster (2023). URL: <https://www.gluster.org/> (retrieval: 10/22/2023).
- Gomaa, H. (1983)**: The Impact of Rapid Prototyping on Specifying User Requirements. In: *SIGSOFT Softw. Eng. Notes* 8.2, pp. 17–27. ISSN: 0163-5948. DOI: 10.1145/1005959.1005964. URL: <https://doi.org/10.1145/1005959.1005964> (retrieval: 11/03/2023).
- 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).
- Home Page | Pachyderm (2023). URL: <https://www.pachyderm.com/> (retrieval: 10/27/2023).
- Hoste, K./Timmerman, J./Georges, A./De Weirtdt, 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).
- IEEE Xplore Full-Text PDF: (2023). URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5599253> (retrieval: 11/03/2023).
- 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).

- @lasthash ‘Introduction to Software Containers with Singularity.’ (w.y.) (RRZE). URL: <https://hpc.fau.de/files/2020/05/2020-05-12-singularity-containers.pdf>.
- iPaaS Solution for the Enterprise (2023). SnapLogic. URL: <https://www.snaplogic.com/> (retrieval: 10/07/2023).
- Jenkins - Market Share, Competitor Insights in Continuous Integration And Delivery (2023). 6sense. URL: <https://www.6sense.com/tech/continuous-integration/jenkins-market-share> (retrieval: 10/31/2023).
- Jenkins 4.8.2 · Jenkins/Jenkinsci (2023). URL: <https://artifacthub.io/packages/helm/jenkinsci/jenkins> (retrieval: 10/31/2023).
- Jenkins/Jenkins - Docker Image | Docker Hub (2023). URL: <https://hub.docker.com/r/jenkins/jenkins> (retrieval: 11/01/2023).
- Jin, H./Chen, Y./Zhu, H./Sun, X.-H. (2010):** Optimizing HPC Fault-Tolerant Environment: An Analytical Approach. In: *2010 39th International Conference on Parallel Processing*. 2010 39th International Conference on Parallel Processing (ICPP). San Diego, CA, USA: IEEE, pp. 525–534. ISBN: 978-1-4244-7913-9. DOI: 10.1109/ICPP.2010.80. URL: <http://ieeexplore.ieee.org/document/5599253/> (retrieval: 11/03/2023).
- Kaniko - Build Images In Kubernetes* (2023). GoogleContainerTools. URL: <https://github.com/GoogleContainerTools/kaniko> (retrieval: 11/01/2023).
- Keeton, K./Singhal, S./Raymond, M. (2019):** The OpenFAM API: A Programming Model for Disaggregated Persistent Memory. In: *OpenSHMEM and Related Technologies. OpenSHMEM in the Era of Extreme Heterogeneity*. Ed. by Swaroop Pophale/Neena Imam/Ferrol Aderholdt/Manjunath Gorentla Venkata. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 70–89. ISBN: 978-3-030-04918-8. DOI: 10.1007/978-3-030-04918-8_5.
- Knative Docs/LICENSE at Main · Knative/Docs (2023). GitHub. URL: <https://github.com/knative/docs/blob/main/LICENSE> (retrieval: 10/21/2023).
- Kraushaar, J. M./Shirland, L. E. (1985):** A Prototyping Method for Applications Development by End Users and Information Systems Specialists. In: *MIS Quarterly* 9.3, pp. 189–197. ISSN: 0276-7783. DOI: 10.2307/248948. JSTOR: 248948. URL: <https://www.jstor.org/stable/248948> (retrieval: 11/03/2023).
- 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).
- Kumar, P. (2020):** How to Setup Private Docker Registry in Kubernetes (K8s). URL: <https://www.linuxtechi.com/setup-private-docker-registry-kubernetes/> (retrieval: 10/29/2023).
- 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).

- Martin, J. (1991):** Rapid Application Development. In collab. with Internet Archive. New York : Macmillan Pub. Co. ; Toronto : Collier Macmillan Canada ; New York : Maxwell Macmillan International. 826 pp. ISBN: 978-0-02-376775-3. URL: <http://archive.org/details/rapidapplication00mart> (retrieval: 11/03/2023).
- 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.
- Naumann, J. D./Jenkins, A. M. (1982):** Prototyping: The New Paradigm for Systems Development. In: *MIS Quarterly* 6.3, pp. 29–44. ISSN: 0276-7783. DOI: 10.2307/248654. JSTOR: 248654. URL: <https://www.jstor.org/stable/248654> (retrieval: 11/03/2023).
- Open Source Cloud Computing Infrastructure (2023). OpenStack. URL: <https://www.openstack.org/> (retrieval: 11/05/2023).
- 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 Target Audience (2023). URL: <https://docs.pachyderm.com/latest/learn/target-audience/> (retrieval: 11/01/2023).
- Pachyderm/LICENSE at Master · Pachyderm/Pachyderm (2023). URL: <https://github.com/pachyderm/pachyderm/blob/master/LICENSE> (retrieval: 10/21/2023).
- Pachyderm -Pricing (2022). Pachyderm. URL: <https://www.pachyderm.com/pricing/> (retrieval: 10/21/2023).
- Production-Grade Container Orchestration (2023). Kubernetes. URL: <https://kubernetes.io/> (retrieval: 11/05/2023).
- Red Hat OpenShift Enterprise Kubernetes Container Platform (2023). URL: <https://www.redhat.com/en/technologies/cloud-computing/openshift> (retrieval: 11/05/2023).
- 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.
- Sayers, Craig/Laffitte, H./Reddy, P./Ozonat, K./Sayal, M./Simitsis, A./Singhal, S./Koutrika, G./Das, M./Aji, A./Bosamiya, H. A./Riss, M./Wilkinson, K. (2015):** CCloud Application Services Platform. In.
- Shenoi, V. V./Shah, V./Joshi, S. K. (2019):** HPC Education for Domain Scientists: An Indian Experience and Perspective. In: *2019 26th International Conference on High Performance Computing, Data and Analytics Workshop (HiPCW)*. 2019 26th International Conference on High Performance Computing, Data and Analytics Workshop (HiPCW). Hyderabad, India: IEEE, pp. 64–70. ISBN: 978-1-72814-894-6. DOI: 10.1109/HiPCW.2019.00023. URL: <https://ieeexplore.ieee.org/document/9001797/> (retrieval: 11/01/2023).
- 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).
- Stack Overflow Developer Survey 2022 (2023). Stack Overflow. URL: <https://survey.stackoverflow.co/2022/#section-most-popular-technologies-other-tools> (retrieval: 11/05/2023).
- 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).
- Using Docker-in-Docker for Your CI or Testing Environment? Think Twice. (2023). URL: <https://jpetazzo.github.io/2015/09/03/do-not-use-docker-in-docker-for-ci/> (retrieval: 11/01/2023).
- Weighted Sum Model* (2022). In: *Wikipedia*. URL: https://en.wikipedia.org/w/index.php?title=Weighted_sum_model&oldid=1114224941 (retrieval: 10/16/2023).
- What Is a Container? (2023): What Is a Container? | Docker. URL: <https://www.docker.com/resources/what-container/> (retrieval: 11/05/2023).
- Wilde, T./Hess, T. (w.y.):** Methodenspektrum der Wirtschaftsinformatik: Überblick und Portfoliobildung. In.
- Xia, W./Wen, Y./Foh, C. H./Niyato, D./Xie, H. (2015):** A Survey on Software-Defined Networking. In: *IEEE Communications Surveys & Tutorials* 17.1, pp. 27–51. ISSN: 1553-877X. DOI: 10.1109/COMST.2014.2330903. URL: <https://ieeexplore.ieee.org/abstract/document/6834762> (retrieval: 11/05/2023).