Building and Monitoring a KubeFlow Machine Learning Pipeline Using AWS EKS, Prometheus, and Grafana

Ryan T. Peterson

Western Governors University

Table of Contents

Proposal Overview

AstroMined Labs is a stealth startup developing a plan to launch what many believe could be the most disruptive and challenging mission in human history. The company will be the first to conduct a detailed survey of the asteroid belt between Mars and Jupiter to find the most suitable candidates to mine for natural resources. Although the scale of this ambition is undoubtedly massive, AstroMined Labs plans to leverage machine learning and automation to maximize their chance of success while minimizing mission costs.

Problem Summary

Current scientific belief is the asteroid belt contains between 1.1 and 1.9 million asteroids that are more significant than one kilometer (0.6 miles) in diameter and millions of more minor asteroids. (In Depth | Asteroids – NASA Solar System Exploration, n.d.) To minimize the time and money spent on the survey, AstroMined Labs will launch 500 microsatellites in an arcing pattern on each mission, with each microsatellite traveling towards a pre-determined sector of the belt. They will use their robust long-range sensors to gather data about the asteroids in their designated sector. Once they have covered half the distance between the Earth and the belt, the microsatellites will come to a consensus about the most promising sector, where they will converge to perform a detailed survey.

Since the microsatellites will be operating autonomously, they need to use as much of their energy and computing power for data collection and navigation. Therefore, Earth-based servers will conduct all computationally intensive machine learning model-building processes. The vast distance between Earth and the belt compounds the challenges presented by the sheer number of asteroids there, as communication delays between Earth and the belt can exceed 20

minutes. These factors combine to necessitate an extremely performant machine learning pipeline on the Earth-based servers.

IT Solution

With petabytes of data streaming back to Earth daily, it will be essential to train machine learning models using a distributed solution to minimize the time between when new data is received, and updated models are sent back to the microsatellites. KubeFlow is a purpose-built project to enable large machine learning pipelines to operate in a distributed Kubernetes-based computing environment. As 100% uptime is essential to the success of this project, the ML Ops team will monitor every part of the KubeFlow infrastructure using Prometheus and Grafana to allow them to detect problems proactively before they worsen.

AstroMined Labs will use their on-premises server farm as the primary storage pool to minimize the cost of storing such a massive amount of data. Using EKS Anywhere combined with an Istio Service Mesh, they can achieve a seamless hybrid computing environment that maximizes performance while minimizing costs. The control plane used will be AWS Elastic Kubernetes Service (EKS) so that powerful GPU-enabled instances can do the computational heavy lifting. Prometheus and Grafana will also be deployed in a distributed manner for ease of management and scalability using Kubernetes.

Implementation Plan

The centerpiece of this project is the KubeFlow machine learning pipeline, and all supporting steps are taken with the ultimate goal of providing a seamless user experience for the end users at AstroMined Labs. Because machine learning engineers work with a very diversified toolbox, one of the most important goals of KubeFlow is to customize the stack based on user

requirements while delegating time-consuming tasks to the system. The KubeFlow platform uses a set of manifests that will provide the end-user with a simple and easy-to-use machine learning stack that will self-configure anywhere a Kubernetes cluster is already running. This idea is described best by the mission statement for the KubeFlow project:

> Our goal is to make scaling machine learning (ML) models and deploying them to production as simple as possible, by letting Kubernetes do what it's great at:
>
> - Easy, repeatable, portable deployments on a diverse infrastructure (for example, experimenting on a laptop, then moving to an on-premises cluster or to the cloud)
> - Deploying and managing loosely-coupled microservices
> - Scaling based on demand (Introduction | Kubeflow, 2021)

The first goal of the project is to create a unified control plane across the AWS and on-premises environment. AstroMined labs will be able to deploy, manage, and scale containerized applications running Kubernetes on AWS, thanks to Amazon EKS. The new EKS Anywhere (EKS-A) offering brings this same experience to corporate datacenters using virtual machines and bare-metal servers. To ultimately create a single, unified data plane for the KubeFlow Machine Learning pipelines to operate on, connecting the on-premises and cloud environments in a seamless fashion will be necessary. We can accomplish this with a combination of technologies, including AWS Direct Connect, Istio Service Mesh, and Gloo Mesh.

The second goal of this project is to install the main workflow tools that will be used by AstroMined Labs, including KubeFlow, Prometheus, AlertManager, and Grafana. Kubeflow is a Kubernetes-centric Machine Learning (ML) platform. It includes elements for each stage of the machine learning lifecycle, from data exploration to model training and deployment. Prometheus

is a free and open-source system monitoring and alerting toolkit. The AlertManager is responsible for managing such alerts, which includes muting, inhibiting, aggregating, and sending notifications via various channels such as email, on-call notification systems, and chat platforms. Grafana connects to many data sources, including Graphite, Prometheus, Influx DB, ElasticSearch, and Postgres.

This final goal of the project will have a dual utility of both testing the platform and training the AstroMined Labs ML Ops team on its use. The KubeFlow Project provides convenient example pipelines (samples, 2021) to test the deployment. These examples come complete with full datasets for training machine learning models, and Kubernetes manifests to automate the deployments. The project's final phase will be to compile all of the documentation for the various components of the architecture into a uniform repository to facilitate training and troubleshooting for the ML Ops team.

To put the KubeFlow platform into production use, the overall Implementation Plan will consist of the following phases, which correspond to the project objectives described later:

1. Build the AWS EKS cloud environment
2. Configure EKS Anywhere for AstroMined Labs' on-premises datacenter
3. Connect the on-premises and cloud environments with AWS Direct Connect, Istio Service Mesh, and Gloo Mesh
4. Install and configure KubeFlow on the newly created unified control plane
5. Install and configure Prometheus, Grafana, and AlertManager on the unified control plane
6. Conduct testing of the KubeFlow machine learning pipeline using sample datasets
7. Produce documentation and end-user training in preparation for the handover

Review of Other Work

Work 1: Hidden Technical Debt in Machine Learning Systems

While developing and deploying ML systems is relatively quick and inexpensive, maintaining them over time is difficult and costly. This paper argues that expecting these quick wins for free is dangerous. Instead, the authors discover that massive ongoing maintenance costs are typical in real-world ML systems. The authors argue that ML systems have an exceptional capacity for incurring technical debt in this paper. This debt may be challenging to detect because it exists at the system rather than the code level. Because data influences ML system behavior, traditional abstractions, and boundaries may be subtly corrupted or invalidated.

ML systems mix signals together, which makes it hard to spot improvements. If the remaining errors are strongly linked to specific inputs, improving a single model may lower the accuracy of the whole system. When you change one thing, it affects everything else. Everything applies to the input, hyper-parameters, learning settings, sampling methods, convergence thresholds, data selection, and any other possible tweak. Undeclared consumers are expensive at best and dangerous at worst. These dangers arise because they create a tight coupling between models and other parts of the stack that is hidden. Without access controls, some of these consumers might be feeding the results of one model into another model without anyone knowing.

In software engineering, dependency debt is a big reason code is hard to understand and technical debt is high. In ML systems, it may be harder to find data dependencies than it is to find code debt. Unstable data signals can make ML systems too open to change, sometimes in a way that could be disastrous.

Several things can cause a model to have unused data dependencies. One common thing is that a feature to be added to a model early on becomes unnecessary as the model gets better. Another problem is all features are added to the model simultaneously, even if some don't add much value.

Relation of Artifact to Project Development

This paper truly describes the largest motivating factor behind the creation of the KubeFlow project, along with the desire of AstroMined Labs to implement this workflow. ML systems exacerbate the traditional potentials for technical debt in any software system's design because of the many layers of complexity and abstraction that are added. Machine Learning Engineers spend so much of their time cleansing data and validating models that they don't need the added burden of worrying about the underlying infrastructure powering their workflow. KubeFlow's mission is to simplify the infrastructure component so that ML teams can focus on their core work.

Work 2: How Shell Made Machine Learning on Kubernetes Easy With Arrikto Enterprise Kubeflow

Jimmy Guerrero and Masoud Mirmomeni, Lead Data Scientist at Shell, co-presented a keynote at KubeCon, presenting the idea that Kubernetes and machine learning are a match made in heaven. Shell's data scientists can now quickly deploy machine learning models into production by utilizing containers. Many machine learning models fail to make it into production, which is an open secret in the industry. Lack of skills, software, methodology, and collaboration ability are all factors that contribute to this.

There is often an expectation for Data Scientists to be Kubernetes experts, and these organizations are experiencing friction since that expectation doesn't match reality. Kubeflow makes it easier for cloud-native architecture and the machine learning community to collaborate. Google first introduced Kubeflow in 2017 as a comprehensive set of components that enables data scientists and operators to manage data, train models, tune and serve them, and monitor them.

Royal Dutch Shell has committed up to $2 billion per year to this effort and plans to invest even more in the coming years. Shell wanted to ensure that they could use all of the data when training their models without bankrupting their IT department. It is unrealistic to expect data scientists to be experts in Containers, Kubernetes, storage, horizontal scaling, GPUs, and various other topics.

Masoud's first assignment at Shell was to create a predictive model for a time series application. It took him two months to develop a proper model in Jupyter Notebook format. He and his team began developing a machine learning discipline with Arrikto and Kubeflow to reduce the time to put a model into production. Shell significantly reduced the complexity of

pushing machine learning code that works equally well on a laptop or in a production environment using Kale. Shell's data scientists can easily create production-ready pipelines with Kubeflow and share them with the operations team with a few clicks.\

Relation of Artifact to Project Development

Royal Dutch Shell has an almost immeasurable amount of experience taking a data-driven approach to finding and extracting natural resources. Although they are terrestrial-based, AstroMined Labs can learn a lot from their approach to solving many of the same challenges they will face with in-situ resource utilization in the asteroid belt. Shell has developed a process that AstroMined Labs strives to emulate, and this talk at KubeCon was a major inspiration for the desire of AstroMined Labs to use KubeFlow in production.

Work 3: Kubeflow: An MLOps Perspective

Kubernetes has established itself as the default execution engine for any cloud-native application. The prevalence of Kubernetes has resulted in significant changes in traditional software development. Wrapping your micro-service in a container and deploying it in a pod ensures that it will perform similarly on any Kubernetes cluster with sufficient resources. ML Ops should aid in ensuring that production quality is present and maintained in ML projects. What we're looking for in Kubeflow is how 'Cloud Native' it is to borrow a cliche that's overused or misapplied. The execution is scheduled in a Kubernetes cluster, either an Edge Cluster or a Data Center.

For microservices, ML Ops is similar to DevOps. However, this has more ML-related aspects than just the algorithms, such as data and model management. Cloud-Native implies that developers should build a system from the ground up for container and pod-based execution rather than being retrofitted to work with Kubernetes. An excellent data analysis tool or OLAP database such as Apache Presto, for example, can ingest data from multiple data stores or traditional databases.

A data analytics system that can scale itself autonomously is critical for algorithm selection and future ML work. With a good Model Development Environment and related languages and frameworks, KubeFlow aims to make machine learning model deployment as straightforward as possible. Instead of creating a single piece of Python code or a notebook, we break it down into discrete components. We're building machine learning microservices and orchestrating them with a workflow scheduler.

We're building machine learning microservices and orchestrating them with a workflow scheduler. KubeFlow must store data to persist the In/Out data between components as artifacts. Data between components is serialized and de-serialized to disk using the Python pickle

mechanism. The most important thing is to keep the model separate from the application. In use-cases such as media analytics, ML pipelines must pass a large amount of data to the model.

The model and the business microservice should reside on the same Kubernetes node. Since Kubeflow is built on top of Kubernetes, everything must run as Containers internally. Exposing an ML Scientist or ML Engineer to the complexities of creating a container image via Docker files and K8s deployment files, on the other hand, is not ideal.

Relation of Artifact to Project Development

This article further demonstrates the need for simplified toolsets to allow Data Scientists and Machine Learning Engineers to focus on the core of their work, which is extracting meaningful insights from data. AstroMined Labs will benefit from this simplified process immensely because of the obscene amounts of data they will be processing when their microsatellites are launched. The team can’t be bothered to deal with workflow issues when they have millions of dollars of investment literally hurtling through space waiting for instructions.

Work 4: Kubernetes Monitoring with Prometheus, the Ultimate Guide

This guide shows how to set up Kubernetes monitoring with Prometheus. It describes how to set up kube-state-metrics, pull and collect metrics, configure alerts and dashboards with a Prometheus server and metrics exporters, and the configuration of additional Prometheus stack components inside Kubernetes. Prometheus metrics are human-readable, self-explanatory, and delivered via a standard HTTP transport. The Prometheus server oversees scraping the targets on a regular basis, so that applications and services don't have to worry about emitting data because metrics are pulled rather than pushed. These Prometheus servers have several methods for automatically detecting scrape targets.

Ops teams must overcome Kubernetes-specific challenges to deploy reliable monitoring, alerting, and graphing. Prometheus is a good fit for microservices because you only need to expose a metrics port. Prometheus automatically generates Kubernetes label-based monitoring target configurations. Grafana retrieves metrics from multiple Prometheus servers and displays panels and dashboards.

Relation of Artifact to Project Development

The ML Ops team was sent this blog by SysDig, who was also developing a proposal for monitoring the current infrastructure at AstroMined Labs. Their proposal was to simply containerize their ML workloads and develop a simple monitoring system to provide observability. This proposal ultimately falls short of the needs of the ML Ops team, as simply containerizing the current workflow only adds complexity and more abstraction rather than the simplicity of using a purpose-built platform such as KubeFlow.

Project Rationale

Microsatellites must use as much energy and computational resources as possible for data collecting and navigation. Because of this, Earth-based servers will perform all machine learning model-building. Communication delays between Earth and the belt can approach 20 minutes because of their great distance. These elements require a fast machine learning pipeline on Earth-based computers.

With petabytes of data coming back to Earth every day, it will be vital to train machine learning models utilizing a distributed system to decrease the time between receiving new data and sending updated models to microsatellites. KubeFlow enables massive machine learning pipelines to run on Kubernetes. As 100% uptime is crucial to the success of this project, the ML Ops team will monitor the KubeFlow infrastructure using Prometheus and Grafana to discover problems before they worsen.

AstroMined Labs will use their on-premises server farm as the primary storage pool to reduce costs. An Istio Service Mesh and EKS Anywhere create a seamless hybrid computing environment that maximizes performance while saving expenses. AstroMined Labs will utilize AWS Elastic Kubernetes Service (EKS) to control powerful GPU-enabled instances. The ML Ops team will use Kubernetes to deploy Prometheus and Grafana for easier management and scalability.

AstroMined Labs is in desperate need of simplified toolkits that will allow Data Scientists and Machine Learning Engineers to focus on what they do best: extracting meaningful insights from data. AstroMined Labs will benefit greatly from this streamlined process due to the massive amounts of data that will be processed when their microsatellites are launched. When

millions of dollars of investment are literally hurtling through space waiting for instructions, the team can't be bothered to deal with workflow issues.

AstroMined Labs' entire business model is based on using a data-driven approach to discovering and extracting natural resources from the asteroid belt. Because of the many layers of complexity and abstraction that are added, ML systems exacerbate the traditional potentials for technical debt in any software system's design. Machine Learning Engineers spend so much time cleaning data and validating models that they don't need to worry about the underlying infrastructure that powers their workflow. The mission of KubeFlow is to simplify the infrastructure component so that machine learning teams can focus on their core work.

Current Project Environment

AstroMined Labs has a robust vSphere cluster setup for various workloads based on replicated environments at three physical data centers connected by high-speed networking. To allow designated reserved overcapacity to be used for extra workloads alongside the business-critical workloads that support AstroMined Labs' business, a shared capacity model is used, implemented using the vSphere resource reservation and limit features. Each host server contains NVIDIA GPU cards, which can be dedicated to a single VM or shared by multiple VMs via Passthrough or NVIDIA Grid mechanisms. The ML Ops Team management team has realized significant business benefits and ease of use by supporting this infrastructure with virtualization technology.

The ML Ops Team supports the business-critical workloads that support AstroMined Labs' day-to-day operations. The ML Ops Team wanted to take advantage of the opportunity to host new types of workloads as well as business-critical workloads on their existing infrastructure, using any spare compute capacity that was available.

The ML Ops Team recognized an opportunity to host high-performance computing (HPC) and machine learning (ML) workloads alongside AstroMined Labs' mission-critical workloads. These workloads are used by user communities that are separate from AstroMined Labs' core business users. These new workload opportunities are referred to as "resource computing" by the ML Ops Team. Applications that fit into this "resource computing" model include the TensorFlow and Caffe Machine Learning toolkits.

AstroMined Labs previously hosted these workloads on dedicated clusters of bare-metal servers. However, Users will share the ML Ops Team's virtualized compute resources across

different workloads and user groups. SQL Server clusters, web servers, Windows servers, Linux servers were among the first to be virtualized.

To support the portfolio of business-critical workloads, the ML Ops Team manages three physical datacenters on their property. Each of these three datacenters has an 80 Gbit/s networking bandwidth. The ML Ops Team group supports 40 Gbit networking across the three physical data centers.

Each physical datacenter houses nine Intel X86-based servers that contribute to the cluster. Although the hardware servers are physically distributed across three sites, the setup is configured as a single vCenter cluster. This provides numerous advantages, and the ML Ops Team can take advantage of VMware cluster functionality such as DRS, HA, and EVC.

The cluster consists of 27 servers, each with an Intel Skylake 6250 CPU, 640 GB memory, a 40Gbit SFP+ card, and one Nvidia Tesla V100 GPU, with room in the servers for a second GPU card. There are 27 servers and 27 GPUs in total.

Each high-performance virtual machine has a boot disk in the VMware storage environment. It consists of 64 datastores served by 16 storage devices. There are three types of storage: replicated SSD storage, replicated bulk storage, and bulk storage. The superior performance Data and scratch disks for virtual machines are network-attached to a centralized high-performance Lustre storage system, a parallel file system designed specifically for high-performance computing clusters.

On each server host, VMware vSphere v6.7 update one is installed. The ML Ops Team ensures that all server nodes are configured in the same way by scripting the installation steps. A Kickstart script completely automates the installation and configuration of VMware vSphere, including the addition of VMkernel adapters, datastores, and VLANs, as well as the installation

of VIBs (VMware Installable Bundles) into the ESXi hypervisor, port group settings, and the activation of the log server, NTP server, SSH keys, and access to the ESXi shell.

There are two main core routers on Cisco infrastructure, and every network connection is at least 40 Gbit or greater. The vCenters use more than 60 VLANs in their VMware configuration. A dual 40 Gbit link connects each ESXi server to a redundant set of core switches.

The storage network is responsible for connecting the storage to the ESXi servers. The solution is based on two redundant Ethernet fabrics that link the three datacenters. When one of the paths fails, there is always a backup path. One fabric consists of three Cisco MDS 9300 Series Multilayer Fabric Switches (one in each datacenter) linked by dark fiber and two 40Gbit switches. The ML Ops Team chose the modular core switch to be future-proof and to support new standards such as 25/50/100Gbit.

Mission-critical workloads are hosted in virtual machines on 27 vSphere host servers spread across three physical locations. Business-critical workloads consume up to 50% of total CPU power and memory. Some additional capacity is available within the 50% capacity that is not currently being used.

The cluster configures 50% of its total compute and memory power as reserve capacity. This is done primarily to allow business-critical workload virtual machines (VMs) to fail from one location to another if one of the locations fails. Business-critical workloads are not designed to take advantage of the additional 50%. Because of this overcapacity, the remaining 50% of the power can be used by workloads other than business-critical workloads.

The "reserved overcapacity" of 50% of physical host capacity is achieved by making a vSphere reservation on a resource pool within the available host CPU and memory power. A resource pool is assigned to the new "resource computing" workload VMs. The ML Ops Team

uses a vSphere limit to ensure that high-performance workloads never consume more compute power than business-critical workloads. The team now thinks about "resources" rather than "dedicated clusters" for any application type.

GPUs are an essential component of AstroMined Labs' infrastructure for HPC and machine learning workloads (GPUs). Eventually, the goal is to provide one or more GPUs to a single virtual machine. Currently, two approaches are used to connect GPUs to virtual machines. The vSphere Passthrough or DirectPath I/O method in the vSphere hypervisor, and the "NVIDIA Grid" software product, also known as Quadro DataCenter Virtual Workstation. GPUs can be configured to use DirectPath I/O or NVIDIA Grid. For advanced machine learning users, the DirectPath I/O approach, for example, allows one or more GPUs to be assigned to a single virtual machine. This method is best suited for higher-demand workloads that can fully utilize one or more GPUs.

On the other hand, the NVIDIA Grid method uses the concept of a "virtual GPU," or vGPU, to allow different virtual machines on a server to share access to a single physical GPU. Any virtual machine running on NVIDIA Grid could use either a single virtual GPU (corresponding to a single physical GPU) or a portion of a physical GPU. Using vGPUs in this way allows the ML Ops Team to keep the benefits of vSphere functionality like vMotion and Suspend and Resume while keeping the cluster maintainable.

Each researcher or group can take their own cluster section and use it whenever they want. Rather than having a dedicated cluster for each community, the ML Ops Team can save money by sharing spare capacity. Different application toolkits and platforms can run on additional guest operating systems within other virtual machines on the same hardware. These can be changed by operators as needed to meet the users' needs.

In terms of performance and security, workloads are separated from one another. When sensitive data is present in a project, vSphere can use virtualization mechanisms to isolate it from non-privileged users. Faults in one VM have no effect on the other machines in the cluster.

Using the vSphere platform, AstroMined Labs' ML Ops Team was able to accomplish the following:

- Costs are reduced by sharing resources and reusing the reserved over-capacity of the design. Because of the combination of different workloads on servers, compute resources can be used more efficiently.
- Physical servers that are not dependent on their resident workloads can be easily managed and maintained by the ML Ops Team.
- Redundancy is built into the cluster by design for mission-critical workloads. A server failure is not a problem because vSphere High Availability (HA) ensures that affected VMs are quickly restored on other servers.
- Failure of an entire physical datacenter has been planned for and can be tolerated by the system due to the multi-site design.
- Hardware is replaced incrementally rather than on a large scale.
- GPU-enabled virtual machines with varying workloads can be temporarily suspended in mid-operation if another workload requires the GPU. Users can then return to their original workload at a later time. This is a vSphere 6.7 feature that works in conjunction with NVIDIA Grid software.

Methodology

Traditional IT infrastructure waterfall-style management methods can stymie rapid innovation in the delivery of digital solutions. Four fundamental shifts in organizational thinking at AstroMined Labs can help the ML Ops team become more efficient following Agile methodology. The first involves managing infrastructure in the same way application developers manage code, using software to quickly and reliably configure environments using the Infrastructure as Code concept. Next, forming cross-functional teams can help break down the departmental silos built over the years in traditional settings. Another critical shift is simplifying processes for delivering infrastructure service offerings by developing a self-serve catalog of services that end-users can create in a push-button manner. Finally, improving collaboration between infrastructure and development teams by fostering a DevOps culture can pay immediate dividends through increased job satisfaction and productivity.

The tech world understands how Agile works in software development projects, but they are often unsure how to apply it in other situations such as infrastructure projects. The ML Ops team can use Agile on infrastructure projects with careful planning and continuous iteration to significant effect.

Agile methods are iterative and incremental, separating the project into pieces that are delivered in priority order and in a way that grows the solution over time. Stakeholders are regularly shown work in progress to receive early and ongoing feedback. Furthermore, because deliverables are released throughout the project rather than at the end, the ML Ops team can deliver value in stages. While, in many cases, all infrastructure is required for a solution to go live, there is frequently an opportunity to build it in pieces to manage project risk, reduce complexity, and even accelerate the project.

The ML Ops team typically receives a single request for new infrastructure environments (development, QA/staging, production, and failover). They have traditionally preferred to obtain the complete request at once to handle equipment orders and build out servers more efficiently in their on-premises datacenters. However, what's most straightforward for the ML Ops team may not be best for the business.

Developing environments in phases can improve the whole project in many agile initiatives. If numerous projects wait for their environments to be designed and activated, the last projects may be delayed due to the lack of development environments. Using Agile principles, the operations team may first construct all the development environments in their own Kubernetes namespace, allowing each project to begin. The ML Ops team can then examine QA and production environments by iterating on the Kubernetes manifests from the development namespace. Later environments would be scheduled dependent on development cycle time and project priorities.

This project demands complex networking, including bridging networks while preserving security. It gets more problematic when different entities control the networks, like AWS Direct Connect Partners and AWS itself. Using an Agile methodology, the ML Ops team may first demonstrate basic connectivity in an early iteration by showing that a data request can go from one network through all security layers to the destination network and back. This transaction validates the network design, reducing project risk through network prototyping.

Daily Scrum meetings can assist IT operations specialists meet with the networking team and representatives from the network operations center (NOC) to monitor the health and status of the solution components and act on issues. This cross-functional team's participation in an infrastructure scrum meeting may speed up IT infrastructure challenges. In addition to

infrastructure scrums, a member of this group may periodically attend application development or systems integration scrums to learn about their progress and difficulties and offer solutions.

Per the Agile Manifesto (Manifesto for Agile Software Development, 2001), Agile teams should avoid extensive documentation. It should be created as late in the process as possible without delaying the project to minimize waste documenting things that will change later. The ML Ops team should focus on creating visual documentation, concise writing, and informal documentation formats such as hand-drawn sketches or photos.

Project Goals, Objectives, and Deliverables

Goals, Objectives, and Deliverables Table

Goal	Supporting objectives	Deliverables enabling the project objectives
1. Create a unified control plane across the AWS and on-premises environment	**1.a.** Build the AWS EKS cloud environment	**1.a.i.** To facilitate a unified virtual network and conform to AWS best practices, create a virtual private cloud (VPC) that is configured with public and private subnets.
		1.a.ii. To enable highly available architecture that is less prone to failure, the VPC will span three Availability Zones.
		1.a.iii In the public subnets, create a demilitarized zone (DMZ) to allow for internet access to resources in the private subnets using managed NAT gateways.
		1.a.iv In the private subnets, configure an Auto Scaling Group (ASG) of Amazon Elastic Compute Cloud (EC2) instances spanning all three AZs to act as the Kubernetes nodes.
		1.a.v In the DMZ, Ubuntu bastion hosts are configured as an ASG of EC2 instances spanning all three AZs. This will allow inbound Secure Shell (SSH) access to the Kubernetes nodes without exposing them directly to the open internet.
		1.a.vi The bastion hosts are configured with the Kubernetes command line interface (kubectl) command line interface for managing the Kubernetes cluster.
		1.a.vii The final step is to configure the ASG of Kubernetes nodes as an Amazon EKS cluster to provide the Kubernetes control plane.
	1.b. Configure EKS Anywhere for AstroMined Labs' on-premises datacenter	**1.b.i.** To fully realize the benefits of a highly available architecture, the first step is to build two vCenter Server Appliances (VCSA) and connect them using vCenter-HA.
		1.b.ii. To fully realize the benefits of resource pooling and distributed resource scheduling, provision a minimum of two ESXi host servers for each VCSA.
		1.b.iii. Each ESXi host will house VMs for a minimum of one Kubernetes Control Node and two Kubernetes Worker Nodes each.
		1.b.iv. vSphere DRS will be deployed in the environment to enable all Kubernetes nodes to efficiently use the resources available to them by acting in a manner similar to an AWS Auto Scaling Group.
		1.b.v. The AWS EKS Anywhere control plane will be installed and configured to use the vSphere DRS cluster as a managed node group.
	1.c. Connect the on-premises and cloud environments with AWS Direct Connect, Istio Service Mesh, and Gloo Mesh	**1.c.i.** Connect the AWS Cloud VPC with the on-premises datacenter using AWS Direct Connect.
		1.c.ii. Create a control plane network in both the AWS cloud and the on-premises datacenter using Istio Service Mesh.
		1.c.iii. Connect the Istio Service Meshes into a unified control plane using Gloo Mesh.

Goal	Supporting objectives	Deliverables enabling the project objectives
2. Create a managed machine learning pipeline on the unified control plane	**2.a.** Install and configure KubeFlow on the newly created unified control plane	**2.a.i.** Validate the EKS Cluster created in the first 3 phases.
		2.a.ii. Create a storage class for the persistent data volumes used by the control plane.
		2.a.iii. Create a Kubernetes Namespace for the KubeFlow deployment.
		2.a.iv. Deploy the Kubernetes manifest YAML files provided by the KubeFlow project.
		2.a.v. Validate the installation has succeeded and set administrative controls.
	2.b. Install and configure Prometheus, Grafana, and AlertManager on the unified control plane	**2.b.i.** Deploy Prometheus and AlertManager using the AWS Quick Start.
		2.b.ii. Configure Prometheus to collect data from all parts of the infrastructure
		2.b.iii. Configure AlertManager to alert the ML Ops Team when infrastructure issues threaten the reliability of the platform
		2.b.iv. Deploy Grafana using the AWS Quick Start.
		2.b.v. Configure Grafana Dashboards to use the Prometheus data sources to provide real-time observability into the performance of the platform.
3. Test the environment and prepare the ML Ops team for go-live	**3.a.** Conduct testing of the KubeFlow machine learning pipeline using sample datasets	**3.a.i.** Deploy the example pipelines multiple times with members of the ML Ops team to ensure their full understanding of the process along with the reliability of the deployment.
		3.a.ii. Deploy a Jupyter Hub notebook server that will integrate into the workflow of the Data Scientists and Machine Learning Engineers at AstroMined Labs.
		3.a.iii. Work with senior members of the Data Science and Machine Learning staff to demonstrate usage of the notebook server and determine the likely points of confusion that will need to be documented for junior members of the staff.
		3.a.iv. Demonstrate the configuration of Prometheus, AlertManager, and Grafana to the ML Ops team.
	3.b. Produce documentation and end-user training in preparation for the handover	**3.b.i.** Complete architecture diagrams of the infrastructure, including all diagrams included in this document.
		3.b.ii. Links to all official documentation sources for the various components of this project
		3.b.iii. Compile a repository of example projects to be used for training purposes
		3.b.iv. Record videos training demonstrating how to deploy the example projects

Goals, Objectives, and Deliverables Descriptions

Goal 1. Create a unified control plane across the AWS and on-premises environment

The first goal of the project is to create a unified control plane across the AWS and on-premises environment. AstroMined labs will be able to deploy, manage, and scale containerized applications running Kubernetes on AWS, thanks to Amazon EKS. The new EKS Anywhere (EKS-A) offering brings this same experience to corporate datacenters using virtual machines and bare-metal servers. To ultimately create a single, unified data plane for the KubeFlow Machine Learning pipelines to operate on, connecting the on-premises and cloud environments in a seamless fashion will be necessary. We can accomplish this with a combination of technologies, including AWS Direct Connect, Istio Service Mesh, and Gloo Mesh.

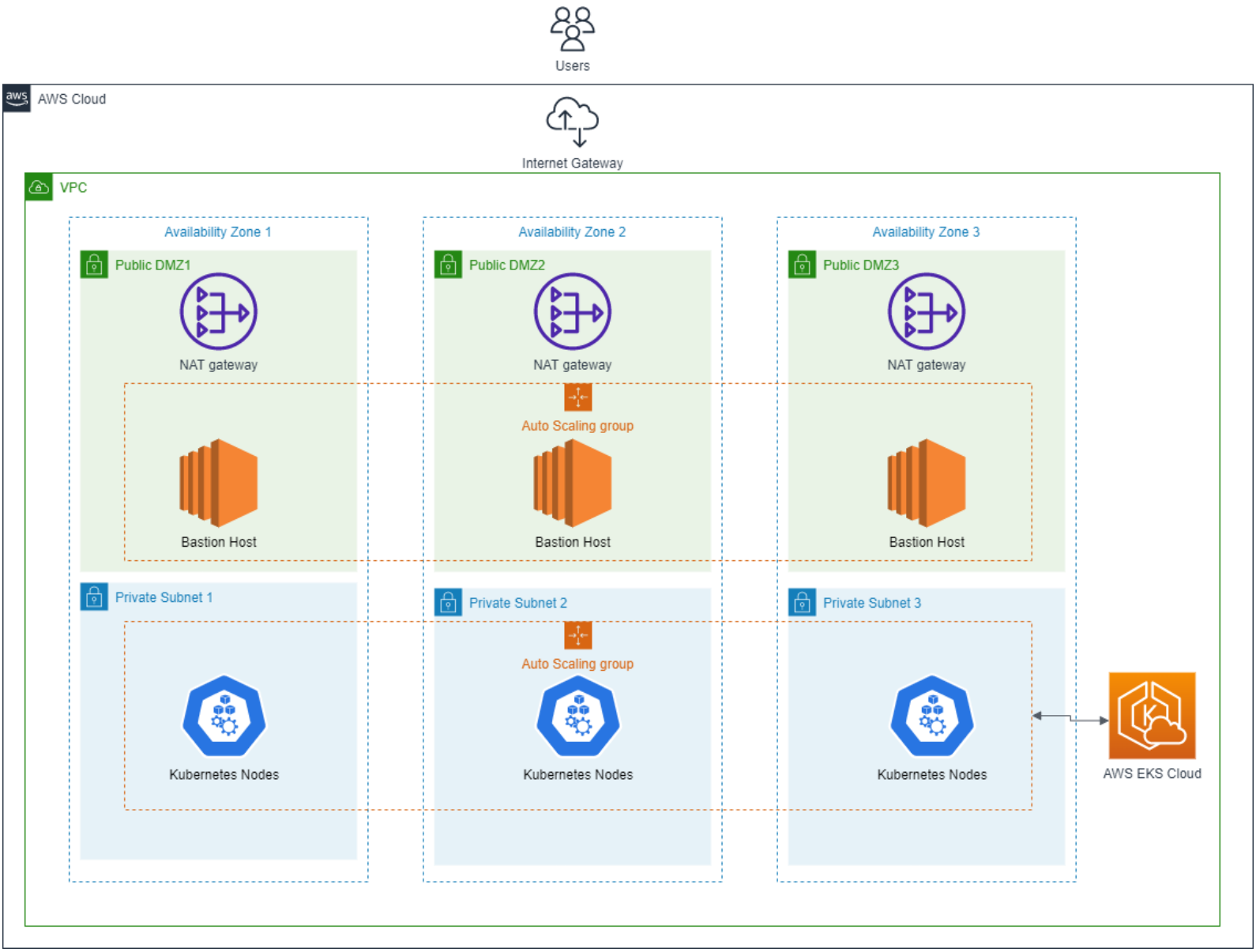
Objective 1.a. Build the AWS EKS Cloud Environment

AstroMined labs will be able to deploy, manage, and scale containerized applications running Kubernetes on AWS, thanks to Amazon EKS. Amazon EKS distributes the Kubernetes management infrastructure across multiple Availability Zones to avoid having a single point of failure. To ensure the use of existing tools and plugins from partners and the Kubernetes community, Amazon EKS has been certified to conform to the Kubernetes specification. Amazon EKS makes it simple for the ML Ops team to migrate applications currently running in a standard Kubernetes environment to Amazon EKS because EKS provides full compatibility options.

The deliverables needed to create the Amazon EKS cloud environment are as follows:

i. To facilitate a unified virtual network and conform to AWS best practices, create a virtual private cloud (VPC) that is configured with public and private subnets.
ii. To enable highly available architecture that is less prone to failure, the VPC will span three Availability Zones.
iii. In the public subnets, using managed NAT gateways creates a demilitarized zone (DMZ) to allow for internet access to resources in the private subnets.
iv. In the private subnets, configure an Auto Scaling Group (ASG) of Amazon Elastic Compute Cloud (EC2) instances spanning all three AZs to act as the Kubernetes nodes.
v. In the DMZ, Ubuntu bastion hosts are configured as an ASG of EC2 instances spanning all three AZs. This will allow inbound Secure Shell (SSH) access to the Kubernetes nodes without exposing them directly to the open internet.
vi. The bastion hosts are configured with the Kubernetes command line interface (kubectl) command line interface for managing the Kubernetes cluster.
vii. The final step is to configure the ASG of Kubernetes nodes as an Amazon EKS cluster to provide the Kubernetes control plane.

When fully deployed, the architecture of the AWS EKS implementation will conform to this diagram:



Objective 1.b. Configure EKS Anywhere in AstroMined Labs' On-premises Datacenter

The "hybrid cloud" buzzword refers to the combination of public cloud computing and on-premises computing infrastructure. Utilizing compute environments across both environments allows engineers to take advantage of the cost-effective scalability of public cloud services such as AWS while reaping the benefits of dedicated resources provided by private cloud computing in an on-premises datacenter.

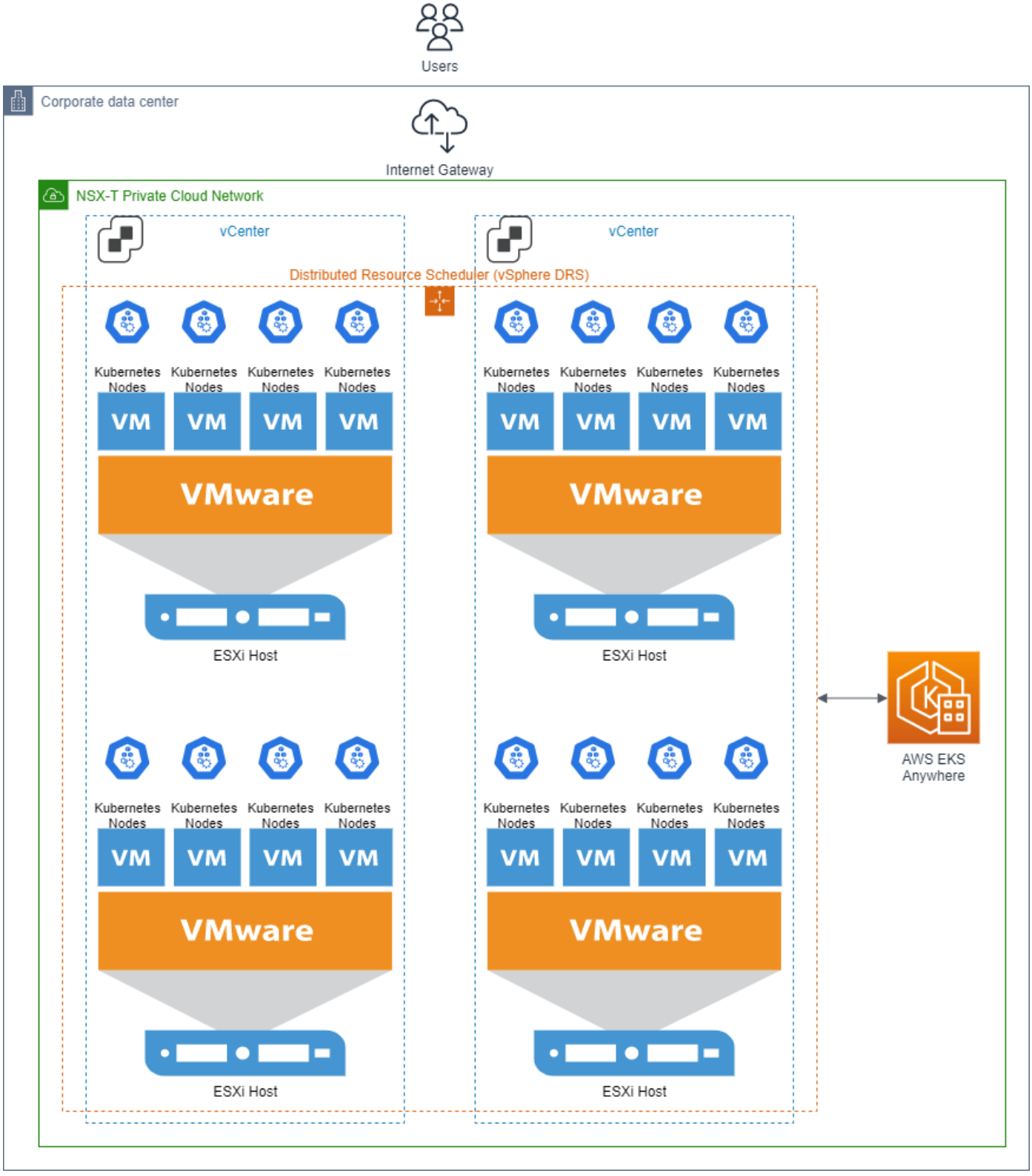
Realizing the benefits of a hybrid cloud environment can only occur when developers can consistently deploy and run their applications on a consistent infrastructure. Whether the infrastructure is in a physical or virtual location and any other specifics of the underlying infrastructure should be abstracted away from developers and applications.

The Amazon Elastic Kubernetes Service (EKS) enables operations teams to deploy Kubernetes clusters in an AWS public cloud quickly. The new EKS Anywhere (EKS-A) offering brings this same experience to corporate datacenters by facilitating the creation and operation of Kubernetes clusters on-premises using an organization's own virtual machines and bare-metal servers. EKS and EKS-A can work together to ensure that developers have the tools and resources they need to deploy their applications seamlessly and consistently from any location.

The deliverables needed to create the Amazon EKS Anywhere private cloud environment are as follows:

i. To fully realize the benefits of a highly available architecture, the first step is to build two vCenter Server Appliances (VCSA) and connect them using vCenter-HA.
ii. To fully realize the benefits of resource pooling and distributed resource scheduling, provision a minimum of two ESXi host servers for each VCSA.
iii. Each ESXi host will house VMs for a minimum of one Kubernetes Control Node and two Kubernetes Worker Nodes each.
iv. vSphere DRS will be deployed in the environment to enable all Kubernetes nodes to efficiently use the resources available to them by acting in a manner similar to an AWS Auto Scaling Group.
v. The AWS EKS Anywhere control plane will be installed and configured to use the vSphere DRS cluster as a managed node group.

When fully deployed, the architecture of the AWS EKS Anywhere implementation will conform to this diagram:



Objective 1.c. Connect the On-Premises VMware Datacenter with the AWS Cloud Environment

To ultimately create a single, unified data plane for the KubeFlow Machine Learning pipelines to operate on, it will be necessary to connect the on-premises and cloud environments in a seamless fashion. We can accomplish this with a combination of technologies, namely AWS Direct Connect, Istio Service Mesh, and Gloo Mesh.

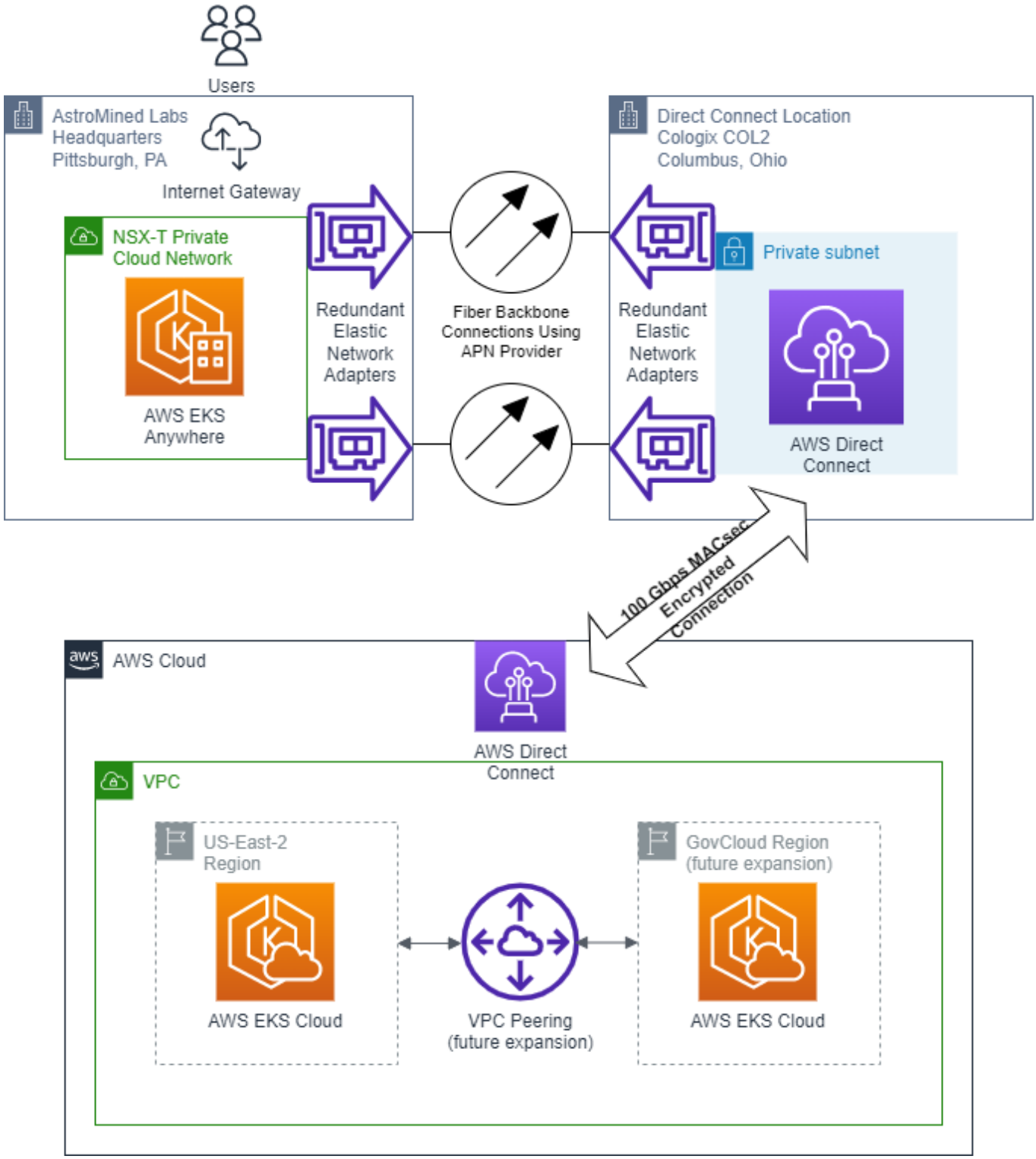
The deliverables needed to create the Amazon EKS Anywhere private cloud environment are as follows:

i. Connect the AWS Cloud VPC with the on-premises datacenter using AWS Direct.
ii. Create a control plane network in both the AWS cloud and the on-premises datacenter using Istio Service Mesh.
iii. Connect the Istio Service Meshes into a unified control plane using Gloo Mesh.

The AWS Direct Connect cloud service connects AWS resources with on-premises resources by providing the shortest network path available. Because network traffic is routed through the AWS global network and never touches the public internet, there is less chance of encountering bottlenecks or experiencing unexpected increases in latency during the transit phase.

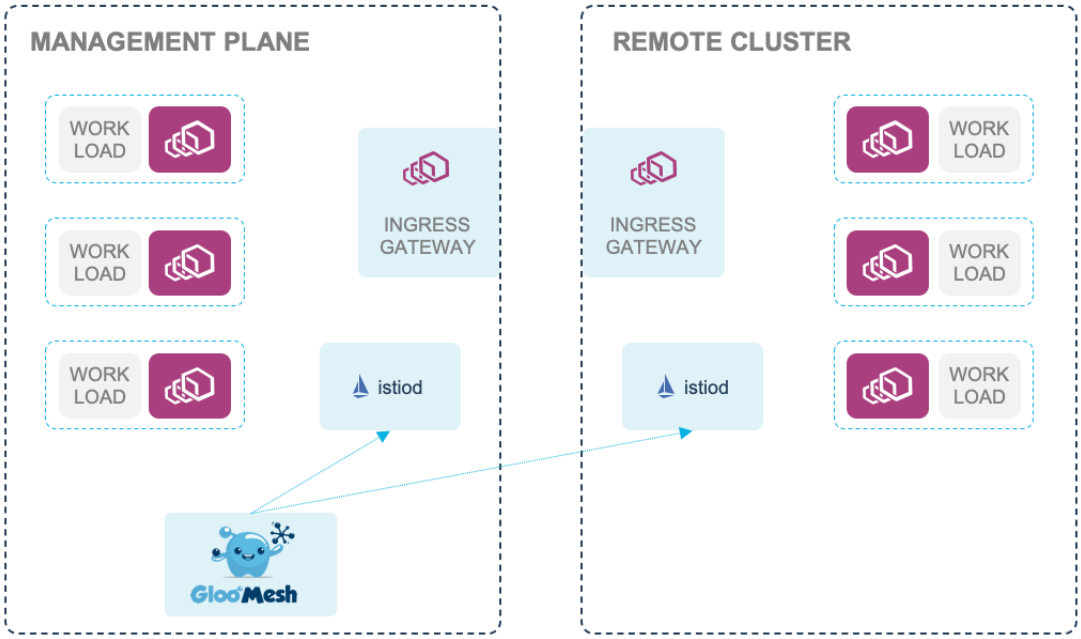
Given that AstroMined Labs is headquartered in Pittsburgh, Pennsylvania, the closest Direct Connect location is Cologix COL2, located in Columbus, Ohio. It will be possible to establish a direct connection with both the US-East-2 and GovCloud AWS regions from this location, which will also have the additional benefit of enabling future collaboration with NASA in the GovCloud region. The growing tech industries in Pittsburgh and Columbus have prompted several Amazon Partner Network providers to build 100 Gbps fiber backbone connections between the two cities, ensuring that availability, speed, and latency will never be a concern.

When completed, the network will conform to this diagram:



With the two EKS control planes now physically connected, the main challenge becomes connecting the services hosted within those control planes to create a unified management layer. The design of a service mesh solves these service connectivity challenges so that developers can focus on the business logic of their services rather than the underlying infrastructure. Istio is the most widely used and well-documented production service mesh, and it uses Envoy as a sidecar proxy.

Gloo Mesh is an abstraction layer above Istio and AWS App Mesh. Gloo Mesh prioritizes security, simplifies multicluster operation, and provides global failover routing for services across clusters regardless of where they are running. Gloo Mesh supports clusters on EKS-Anywhere, EKS, vSphere VMs, and bare metal servers. The diagram below depicts one possible architecture for running Istio, Envoy, and Gloo Mesh on EKS Anywhere and AWS cloud infrastructure.



Source: https://github.com/solo-io/gloo-mesh/blob/main/docs/content/img/gloomesh-2clusters.png

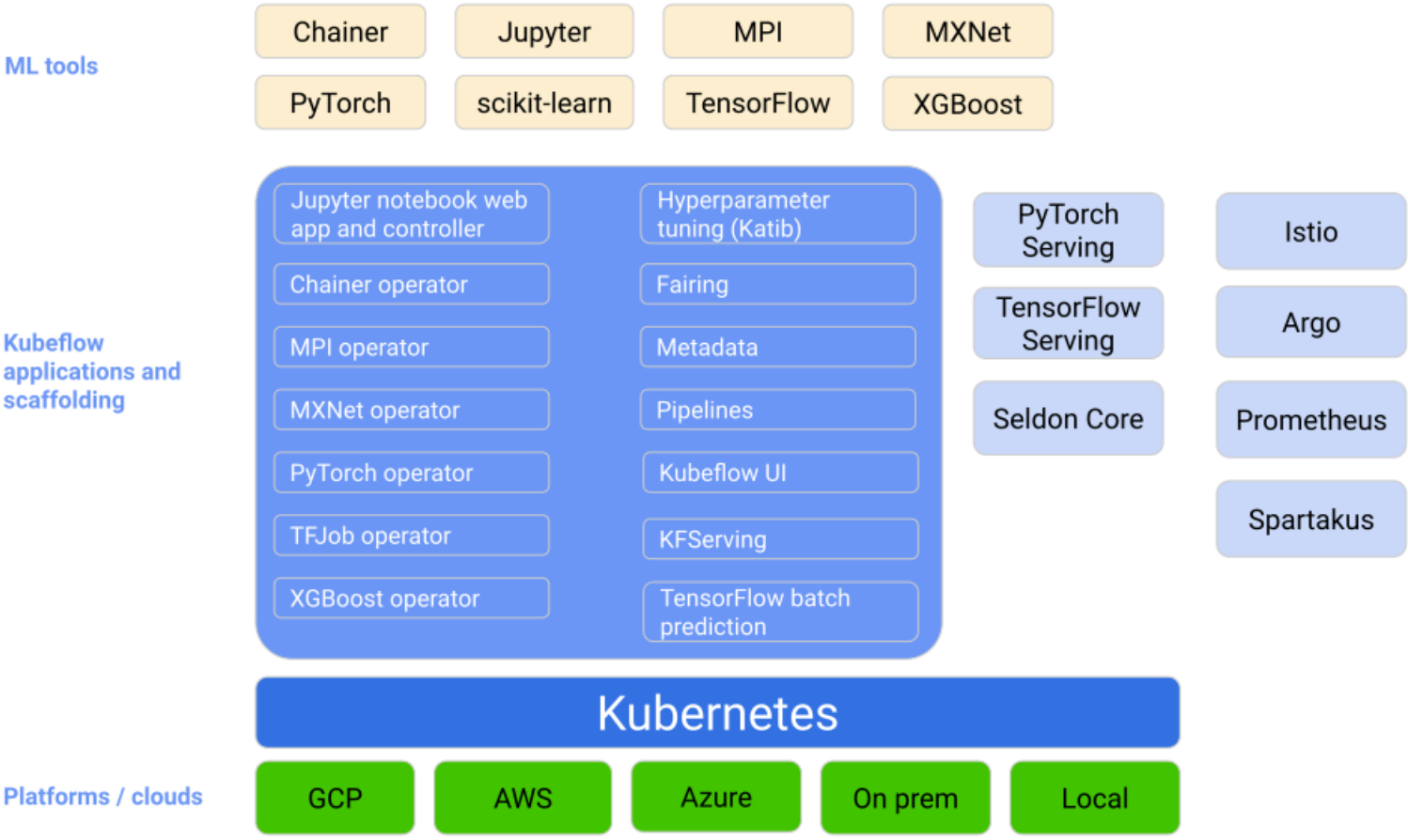
Goal 2. Create a managed machine learning pipeline on the unified control plane

The second goal of this project is to install the main workflow tools that will be used by AstroMined Labs, including KubeFlow, Prometheus, AlertManager, and Grafana. Kubeflow is a Kubernetes-centric Machine Learning (ML) platform. It includes elements for each stage of the machine learning lifecycle, from data exploration to model training and deployment. Prometheus is a free and open-source system monitoring and alerting toolkit. The AlertManager is responsible for managing such alerts, which includes muting, inhibiting, aggregating, and sending notifications via various channels such as email, on-call notification systems, and chat platforms. Grafana connects to many data sources, including Graphite, Prometheus, Influx DB, ElasticSearch, and Postgres.

Objective 2.a. Install and Configure KubeFlow on the Newly Created Unified Control Plane

Kubeflow is a Kubernetes-centric Machine Learning (ML) platform. It includes elements for each stage of the machine learning lifecycle, from data exploration to model training and deployment. There is no requirement to implement all components, and the ML Ops team is free to implement only those that they believe will benefit their users the most.

The diagram below shows a small sample of what is possible with the Kubeflow platform:



Source: (Architecture | Kubeflow, 2022)

Since The ML Ops team at AstroMined Labs will be responsible for deploying their required toolsets on top of the KubeFlow platform, we will only deploy the basic scaffolding.

The deliverables for the deployment process will follow these steps:

i. Validate the EKS Cluster created in the first 3 phases

ii. Create a storage class for the persistent data volumes used by the control plane

iii. Create a Kubernetes Namespace for the KubeFlow deployment

iv. Deploy the Kubernetes manifest YAML files provided by the KubeFlow project

v. Validate the installation has succeeded and set administrative controls

Objective 2.b Install and Configure Prometheus, AlertManager , and Grafana

Prometheus is a free and open-source system monitoring and alerting toolkit developed by SoundCloud. It is now a stand-alone open-source project, independent of any company. After Kubernetes, Prometheus became the second hosted project at the Cloud Native Computing Foundation (CNCF) in 2016. The Prometheus ecosystem comprises many different components, many of which are optional. It contains the central Prometheus server, which scrapes and stores time-series data and client libraries for instrumenting application code and various support tools.

Prometheus servers generate alerts delivered to an AlertManager via their respective alerting rules. The AlertManager is then responsible for managing such alerts, which includes muting, inhibiting, aggregating, and sending notifications via various channels such as email, on-call notification systems, and chat platforms. To begin the process of configuring alerting and notifications, the AlertManager must first be installed and configured. The next step is to configure Prometheus to communicate with the AlertManager. As a final step, create alerting rules in Prometheus.

Grafana is an open-source tool for executing data analytics, retrieving metrics, and monitoring apps with beautiful dashboards. Grafana connects to many data sources, including Graphite, Prometheus, Influx DB, ElasticSearch, MySQL, and PostgreSQL. Grafana's open-source nature lets us design bespoke plugins for various data sources. Time series analytics is helped by the tool. It helps us track user activity, application behavior, error frequency, error kind, and contextual events by providing comparative statistics. Organizations that don't want their data transmitted to a vendor cloud can deploy the project on-premises.

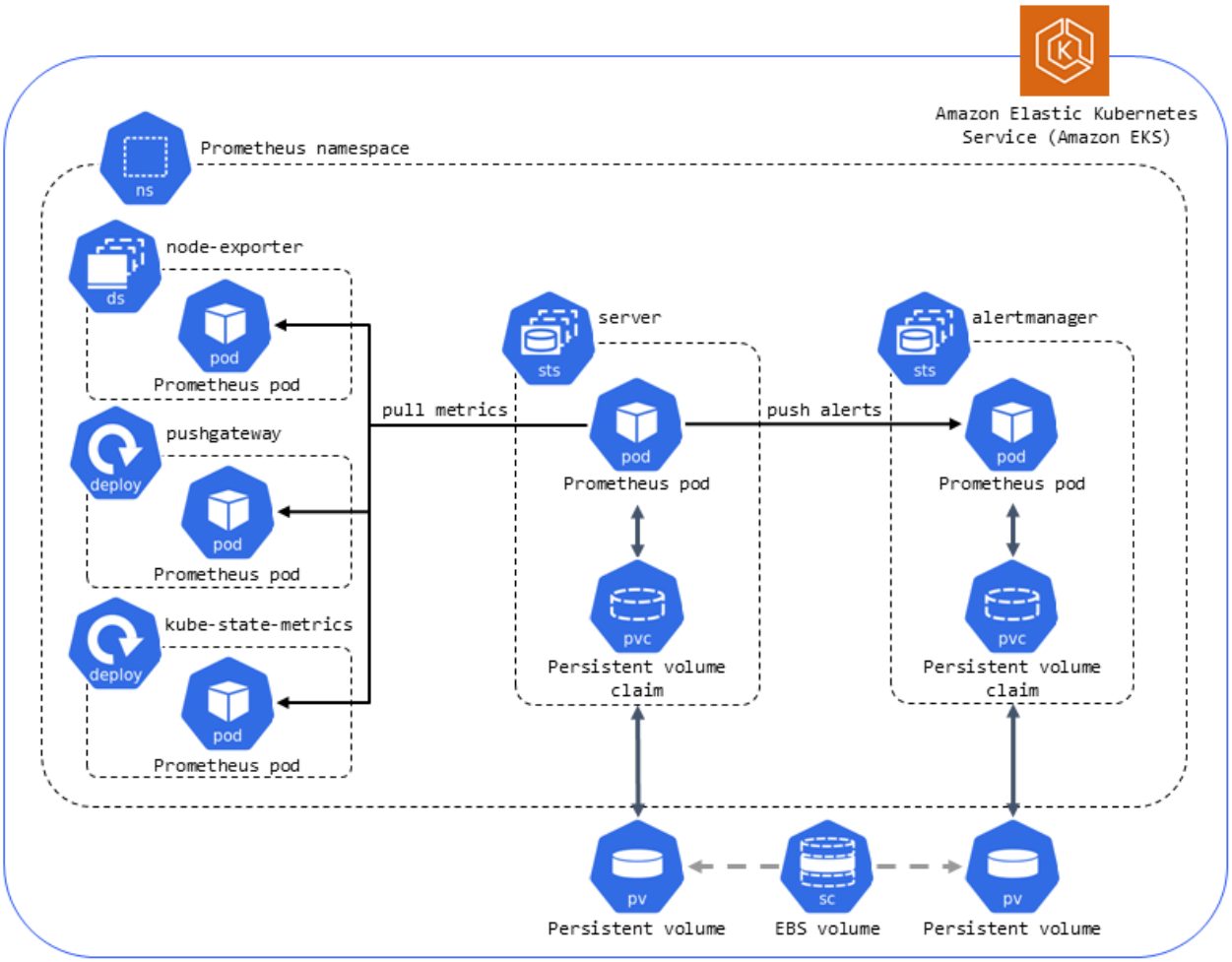
To ensure a smooth deployment of Prometheus, AlertManager, and Grafana, we will utilize the Quick Start Reference Deployments provided by AWS. These documents provide

AWS Cloud Formation templates and detailed guidance to follow the best practices developed by the experts at AWS.

The deliverables for this objective will be as follows:

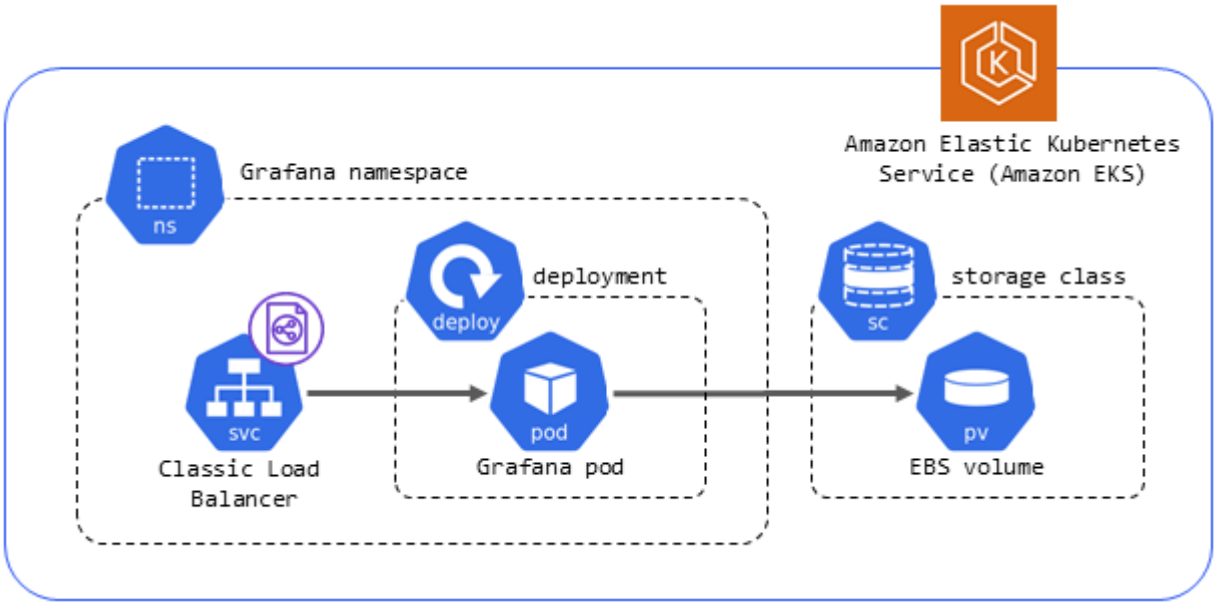
i. Deploy using the AWS Quick Start.
ii. Configure Prometheus to collect data from all parts of the infrastructure
iii. Configure AlertManager to alert the ML Ops Team when infrastructure issues threaten the reliability of the platform
iv. Deploy Grafana using the AWS Quick Start.
v. Configure Grafana Dashboards to use the Prometheus data sources to provide real-time observability into the performance of the platform.

The Prometheus for Amazon EKS Quick Start Reference Deployment (Joshi & McConnell, 2021) will produce the following architecture for the Prometheus and AlertManager components:



Source: https://aws-quickstart.github.io/quickstart-eks-prometheus/images/architecture_diagram.png

The Grafana for Amazon EKS Quick Start Reference Deployment (Joshi & McConnell, 2021) will produce the following architecture for the Grafana component, which will provide visualization and dashboarding capabilities for the data gathered by Prometheus:



Source: https://aws-quickstart.github.io/quickstart-eks-grafana/images/architecture_diagram.png

Once complete, the ML Ops team will have complete observability of their entire infrastructure through Grafana dashboards such as this example of the basic Kubernetes Control Plane:

Goal 3. Test the environment and prepare the ML Ops team for go-live

This final goal of the project will have a dual utility of both testing the platform and training the AstroMined Labs ML Ops team on its use. The KubeFlow Project provides convenient example pipelines (samples, 2021) to test the deployment. These examples come complete with full datasets for training machine learning models, and Kubernetes manifests to automate the deployments. The project's final phase will be to compile all of the documentation for the various components of the architecture into a uniform repository to facilitate training and troubleshooting for the ML Ops team.

Objective 3.a. Conduct Testing of the KubeFlow Machine Learning Pipeline Using Sample Datasets

This phase will have a dual utility of both testing the platform and training the AstroMined Labs ML Ops team on its use. The KubeFlow Project provides convenient example pipelines (samples, 2021) to test the deployment. These examples come complete with full datasets for training machine learning models along with Kubernetes manifests to automate the deployments.

During this phase, the following deliverables will be completed:

i. Deploy the example pipelines multiple times with members of the ML Ops team to ensure their full understanding of the process along with the reliability of the deployment.
ii. Deploy a Jupyter Hub notebook server that will integrate into the workflow of the Data Scientists and Machine Learning Engineers at AstroMined Labs.
iii. Work with senior members of the Data Science and Machine Learning staff to demonstrate usage of the notebook server and determine the likely points of confusion that will need to be documented for junior members of the staff.
iv. Demonstrate the configuration of Prometheus, AlertManager, and Grafana to the ML Ops team.

Objective 3.b. Compile Documentation and End-user Training in Preparation for the Handover

The final phase of the project will be to compile all of the documentation for the various components of the architecture into a uniform repository to facilitate training and troubleshooting for the ML Ops team.

The following deliverables will be included:

i. Complete architecture diagrams of the infrastructure, including all diagrams included in this document.

ii. Links to all official documentation sources for the various components of this project:

    1. Kubernetes: https://kubernetes.io/docs/home/
    2. Docker: https://docs.docker.com/
    3. Istio Service Mesh: https://istio.io/latest/docs/
    4. Gloo Service Mesh: https://docs.solo.io/gloo-mesh-enterprise
    5. AWS EKS: https://docs.aws.amazon.com/eks/index.html
    6. EKS Anywhere: https://anywhere.eks.amazonaws.com/docs/overview/
    7. AWS Direct Connect: https://docs.aws.amazon.com/directconnect/index.html
    8. KubeFlow: https://www.kubeflow.org/docs/
    9. JupyterHub: https://jupyterhub.readthedocs.io/en/stable/
    10. Prometheus: https://prometheus.io/docs/introduction/overview/
    11. AlertManager: https://prometheus.io/docs/alerting/latest/alertmanager/
    12. Grafana: https://grafana.com/docs/grafana/latest/

iii. Compile a repository of example projects to be used for training purposes

iv. Record videos training demonstrating how to deploy the example projects

Project Timeline with Milestones

Since the Agile methodology will be used for this project, all work will be done in two week sprints. The sprints are deliberately not scheduled full to allow for change orders, testing, and redeployment of failed components. The project has a projected end date of 9/11/2022, and the sprint schedule is outlined in the following table:

Milestone or deliverable	Duration (hours or days)	Projected start date	Anticipated end date
1.a.i. To facilitate a unified virtual network and conform to AWS best practices, create a virtual private cloud (VPC) that is configured with public and private subnets.	4 hours	**Sprint 1** 6/6/2022	6/19/2022
1.a.ii. To enable highly available architecture that is less prone to failure, the VPC will span three Availability Zones.	4 hours	**Sprint 1** 6/6/2022	6/19/2022
1.a.iii In the public subnets, create a demilitarized zone (DMZ) to allow for internet access to resources in the private subnets using managed NAT gateways.	4 hours	**Sprint 1** 6/6/2022	6/19/2022
1.a.iv In the private subnets, configure an Auto Scaling Group (ASG) of Amazon Elastic Compute Cloud (EC2) instances spanning all three AZs to act as the Kubernetes nodes.	2 days	**Sprint 1** 6/6/2022	6/19/2022
1.a.v In the DMZ, Ubuntu bastion hosts are configured as an ASG of EC2 instances spanning all three AZs. This will allow inbound Secure Shell (SSH) access to the Kubernetes nodes without exposing them directly to the open internet.	4 hours	**Sprint 1** 6/6/2022	6/19/2022
1.a.vi The bastion hosts are configured with the Kubernetes command line interface (kubectl) command line interface for managing the Kubernetes cluster.	1 day	**Sprint 1** 6/6/2022	6/19/2022
1.a.vii The final step is to configure the ASG of Kubernetes nodes as an Amazon EKS cluster to provide the Kubernetes control plane.	3 days	**Sprint 1** 6/6/2022	6/19/2022

Milestone or deliverable	Duration (hours or days)	Projected start date	Anticipated end date
1.b.i. To fully realize the benefits of a highly available architecture, the first step is to build two vCenter Server Appliances (VCSA) and connect them using vCenter-HA.	1 day	**Sprint 2** 6/20/2022	7/3/2022
1.b.ii. To fully realize the benefits of resource pooling and distributed resource scheduling, provision a minimum of two ESXi host servers for each VCSA.	1 day	**Sprint 2** 6/20/2022	7/3/2022
1.b.iii. Each ESXi host will house VMs for a minimum of one Kubernetes Control Node and two Kubernetes Worker Nodes each.	1 day	**Sprint 2** 6/20/2022	7/3/2022
1.b.iv. vSphere DRS will be deployed in the environment to enable all Kubernetes nodes to efficiently use the resources available to them by acting in a manner similar to an AWS Auto Scaling Group.	2 days	**Sprint 2** 6/20/2022	7/3/2022
1.b.v. The AWS EKS Anywhere control plane will be installed and configured to use the vSphere DRS cluster as a managed node group.	3 days	**Sprint 2** 6/20/2022	7/3/2022
1.c.i. Connect the AWS Cloud VPC with the on-premises datacenter using AWS Direct Connect.	3 days	**Sprint 3** 7/4/2022	7/17/2022
1.c.ii. Create a control plane network in both the AWS cloud and the on-premises datacenter using Istio Service Mesh.	2 days	**Sprint 3** 7/4/2022	7/17/2022
1.c.iii. Connect the Istio Service Meshes into a unified control plane using Gloo Mesh.	2 days	**Sprint 3** 7/4/2022	7/17/2022
2.a.i. Validate the EKS Cluster created in the first 3 phases.	1 day	**Sprint 4** 7/18/2022	7/31/2022
2.a.ii. Create a storage class for the persistent data volumes used by the control plane.	1 day	**Sprint 4** 7/18/2022	7/31/2022
2.a.iii. Create a Kubernetes Namespace for the KubeFlow deployment.	1 day	**Sprint 4** 7/18/2022	7/31/2022
2.a.iv. Deploy the Kubernetes manifest YAML files provided by the KubeFlow project.	2 days	**Sprint 4** 7/18/2022	7/31/2022
2.a.v. Validate the installation has succeeded and set administrative controls.	2 days	**Sprint 4** 7/18/2022	7/31/2022

Milestone or deliverable	Duration (hours or days)	Projected start date	Anticipated end date
2.b.i. Deploy Prometheus and AlertManager using the AWS Quick Start.	4 hours	**Sprint 5** 8/1/2022	8/14/2022
2.b.ii. Configure Prometheus to collect data from all parts of the infrastructure	3 days	**Sprint 5** 8/1/2022	8/14/2022
2.b.iii. Configure AlertManager to alert the ML Ops Team when infrastructure issues threaten the reliability of the platform	2 days	**Sprint 5** 8/1/2022	8/14/2022
2.b.iv. Deploy Grafana using the AWS Quick Start.	4 hours	**Sprint 5** 8/1/2022	8/14/2022
2.b.v. Configure Grafana Dashboards to use the Prometheus data sources to provide real-time observability into the performance of the platform.	2 days	**Sprint 5** 8/1/2022	8/14/2022
3.a.i. Deploy the example pipelines multiple times with members of the ML Ops team to ensure their full understanding of the process along with the reliability of the deployment.	1 day	**Sprint 6** 8/15/2022	8/28/2022
3.a.ii. Deploy a Jupyter Hub notebook server that will integrate into the workflow of the Data Scientists and Machine Learning Engineers at AstroMined Labs.	1 day	**Sprint 6** 8/15/2022	8/28/2022
3.a.iii. Work with senior members of the Data Science and Machine Learning staff to demonstrate usage of the notebook server and determine the likely points of confusion that will need to be documented for junior members of the staff.	4 days	**Sprint 6** 8/15/2022	8/28/2022
3.a.iv. Demonstrate the configuration of Prometheus, AlertManager, and Grafana to the ML Ops team.	4 days	**Sprint 6** 8/15/2022	8/28/2022
3.b.i. Complete architecture diagrams of the infrastructure, including all diagrams included in this document.	4 hours	**Sprint 7** 8/29/2022	9/11/2022
3.b.ii. Links to all official documentation sources for the various components of this project	4 hours	**Sprint 7** 8/29/2022	9/11/2022
3.b.iii. Compile a repository of example projects to be used for training purposes	1 day	**Sprint 7** 8/29/2022	9/11/2022
3.b.iv. Record videos training demonstrating how to deploy the example projects	6 days	**Sprint 7** 8/29/2022	9/11/2022

Outcome

Understanding statistics and neural networks are just the tip of the iceberg for what is required to build a machine learning model. Writing code in ML is not the only thing that needs to be done. All of this is handled by a single piece of software when you use Kubeflow.

> Photographers have Photoshop, architects have AutoCAD, and machine learning engineers have Kubeflow. Every artist benefits greatly by having software made specific to their discipline, and ML engineers are no different. (Johnson, 2020)

Kubeflow may be run locally or in the cloud on Kubernetes clusters. It makes it simple to use the capability of training machine learning models on numerous computers, which in turn reduces the amount of time needed for training. Kubernetes and its scalability are at the heart of the primary benefits derived from utilizing Kubeflow. After you have everything set up, running your program at a larger scale is a piece of cake.

Kubeflow is rapidly evolving into an indispensable tool for machine learning operations teams. It helps organize the project, takes advantage of cloud computing, and enables a machine learning engineer to dig in and develop the finest models that ML Engineers can build. Kubeflow was designed to make it easier for machine learning (ML) engineers and data scientists to leverage cloud assets, whether they are hosted in the public cloud or on-premises.

All of this points to the only metric that really matters for AstroMined Labs, which was the main driver behind this project to begin with. That metric is the time is takes an ML Engineer to develop a new machine learning model from start to finish. The time delay in communicating with their microsatellites is the biggest driver behind the value of this metric. We will measure the time the process takes with their old workflow, and then produce the same model using same dataset in the KubeFlow pipeline. We estimate that the KubeFlow process will take less than

30% of the time than the original workflow, but to AstroMined labs, even a 10% improvement is worth the money, time, and effort involved in this project.

The ML Ops team also estimates that model accuracy will improve by as much as 5%, and we find that to be a fair estimate. These accuracy gains will stem from the increased computing power, mainly from cloud GPUs, which will allow deeper and wider neural networks to be created. This means more features can be used as inputs to the models, and more layers of deep learning can take place, which generally increases accuracy of models with most datasets.

The final metric that will be used to determine this projects success is the reliability of the platform and the amount of maintenance that is needed to be performed by the ML Ops team. We estimate the Mean Time to Failure (MTTF) for this infrastructure will be 4 to 5 times higher than their original infrastructure, primarily through the use of AWS resources which have failover capabilities built in. Also, with the automation capabilities of Kubernetes and the KubeFlow platform, Mean Time to Repair will be less than 10% of what is was with their original infrastructure, as the platform is largely self-healing.

References

In Depth | Asteroids – NASA Solar System Exploration. (n.d.). NASA Solar System Exploration; solarsystem.nasa.gov. Retrieved April 23, 2022, from https://solarsystem.nasa.gov/asteroids-comets-and-meteors/asteroids/in-depth/#

Introduction | Kubeflow. (2021, November 29). Kubeflow; www.kubeflow.org. https://www.kubeflow.org/docs/started/introduction/#the-kubeflow-mission

solo-io. (2020, November 11). *gloo-mesh/gloomesh-2clusters.png at main · solo-io/gloo-mesh*. GitHub; github.com. https://github.com/solo-io/gloo-mesh/blob/main/docs/content/img/gloomesh-2clusters.png

Architecture | Kubeflow. (2022, March 1). Kubeflow; www.kubeflow.org. https://www.kubeflow.org/docs/started/architecture/

Joshi, S., & McConnell, Jay. (2021, April 0). *Prometheus for Amazon EKS on the AWS Cloud*. Prometheus for Amazon EKS on the AWS Cloud; aws-quickstart.github.io. https://aws-quickstart.github.io/quickstart-eks-prometheus/

Joshi, S., & McConnell, J. (2021, April 0). *Grafana for Amazon EKS on the AWS Cloud*. Grafana for Amazon EKS on the AWS Cloud; aws-quickstart.github.io. https://aws-quickstart.github.io/quickstart-eks-grafana/

samples, kubeflow. (2021, October 11). *pipelines/samples at sdk/release-1.8 · kubeflow/pipelines*. GitHub; github.com. https://github.com/kubeflow/pipelines/tree/sdk/release-1.8/samples

Documentation | Kubeflow. (n.d.). Kubeflow; www.kubeflow.org. Retrieved May 14, 2022, from https://www.kubeflow.org/docs/

Kubernetes Documentation | Kubernetes. (n.d.). Kubernetes; kubernetes.io. Retrieved May 14, 2022, from https://kubernetes.io/docs/home/

Docker Documentation. (2022, May 14). Docker Documentation; docs.docker.com. https://docs.docker.com/

Istio / Documentation. (n.d.). Istio; istio.io. Retrieved May 14, 2022, from https://istio.io/latest/docs/

Gloo Mesh Enterprise :: Gloo Mesh Enterprise Docs. (n.d.). Gloo Mesh Enterprise :: Gloo Mesh Enterprise Docs; docs.solo.io. Retrieved May 14, 2022, from https://docs.solo.io/gloo-mesh-enterprise

Amazon Elastic Kubernetes Service Documentation. (n.d.). Amazon Elastic Kubernetes Service Documentation; docs.aws.amazon.com. Retrieved May 14, 2022, from https://docs.aws.amazon.com/eks/index.html

Overview | EKS Anywhere. (n.d.). EKS Anywhere; anywhere.eks.amazonaws.com. Retrieved May 14, 2022, from https://anywhere.eks.amazonaws.com/docs/overview/

AWS Direct Connect Documentation. (n.d.). AWS Direct Connect Documentation; docs.aws.amazon.com. Retrieved May 14, 2022, from https://docs.aws.amazon.com/directconnect/index.html

JupyterHub — JupyterHub 2.3.0 documentation. (n.d.). JupyterHub — JupyterHub 2.3.0 Documentation; jupyterhub.readthedocs.io. Retrieved May 14, 2022, from https://jupyterhub.readthedocs.io/en/stable/

Documentation | Kubeflow. (n.d.). Kubeflow; www.kubeflow.org. Retrieved May 14, 2022, from https://www.kubeflow.org/docs/

Prometheus. (n.d.). *Overview | Prometheus*. Overview | Prometheus; prometheus.io. Retrieved May 14, 2022, from https://prometheus.io/docs/introduction/overview/

Prometheus. (n.d.). *Alertmanager | Prometheus*. Alertmanager | Prometheus; prometheus.io. Retrieved May 14, 2022, from https://prometheus.io/docs/alerting/latest/alertmanager/

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., & Dennison, D. (2015, 0 0). *Hidden Technical Debt in Machine Learning Systems*. Advances in Neural Information Processing Systems 28 (NIPS 2015). https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf

Guerrero, J. (2021, November 18). *Machine Learning on Kubernetes with Enterprise Kubeflow | Arrikto*. Accelerate Models to Market with Arrikto; www.arrikto.com. https://www.arrikto.com/blog/how-shell-made-machine-learning-on-kubernetes-easy-with-arrikto-enterprise-kubeflow/

Punnen, A. (2021, September 7). *Kubeflow: An MLOps Perspective. ML Pipelines and ML Components | by Alex Punnen | Towards Data Science*. Medium; towardsdatascience.com. https://towardsdatascience.com/kubeflow-an-mlops-perspective-17d33ac57c08

Burillo, M. (2021, February 16). *Kubernetes Monitoring with Prometheus, Ultimate Guide | Sysdig*. Sysdig; sysdig.com. https://sysdig.com/blog/kubernetes-monitoring-prometheus/

Manifesto for Agile Software Development. (2001, 0 0). Manifesto for Agile Software Development; agilemanifesto.org. https://agilemanifesto.org/

Johnson, J. (2020, July 13). *What is Kubeflow? Machine Learning Basics with Kubeflow – BMC Software | Blogs*. BMC Blogs; www.bmc.com. https://www.bmc.com/blogs/kubeflow/