

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

Ryan T. Peterson

Western Governors University

Table of Contents

Summary	3
Review of Other Work	7
Changes to the Project Environment.....	12
Methodology	18
Project Goals and Objectives	21
Project Timeline.....	28
Unanticipated Requirements	31
Conclusions.....	32
Project Deliverables	34
References.....	36
Appendix A.....	39
Appendix B	40
Appendix C	41
Appendix D.....	45
Appendix E	46

Summary

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.

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. 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.

To achieve their goals, AstroMined Labs hired KubeOps to build a distributed machine learning pipeline based on KubeFlow. 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 the future mission, the ML Ops team wished to monitor every part of the KubeFlow infrastructure using Prometheus and Grafana to allow them to detect problems proactively before they worsen.

When the mission commences, 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, we have built a seamless hybrid computing environment that maximizes performance while minimizing costs. The control plane used was AWS Elastic Kubernetes Service (EKS) so that powerful GPU-enabled instances can do the computational heavy lifting. Prometheus and Grafana have been deployed in a distributed manner for ease of management and scalability using Kubernetes.

The centerpiece of this project was the KubeFlow machine learning pipeline, and all supporting steps were 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 was 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 brought this same experience to their corporate datacenters, and used their existing virtual machines and bare-metal servers. KubeOps created a single, unified data plane for the KubeFlow Machine Learning pipelines to operate on by connecting the on-premises and cloud environments in a seamless fashion. We accomplished this with a combination of technologies, including AWS Direct Connect, Istio Service Mesh, and Gloo Mesh.

The second goal of this project was 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 had the dual utility of both testing the platform and training the AstroMined Labs ML Ops team on its use. The KubeFlow Project provided convenient example pipelines (samples, 2021) to test the deployment. These examples came complete with full datasets for training machine learning models, and Kubernetes manifests to automate the deployments. The project's final phase was 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 consisted of the following phases, which correspond to the project objectives described later:

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

Review of Other Work

Work 1: Kubeflow on Amazon EKS

This blog was essentially a tutorial that explained how to deploy Kubeflow on Amazon EKS clusters using P3 worker instances. After that, it demonstrated how to use Kubeflow in conjunction with Kubernetes to efficiently carry out machine learning tasks such as training and model serving. They utilized a Jupyter notebook based on the TensorFlow framework during the training session. Creating and sharing machine learning documents in various programming languages is possible using an open-source web application called a Jupyter notebook. Examples of these languages include Python, Scala, and R. A Python notebook was utilized in this particular example.

The goal of the Kubeflow project was to make it easier to deploy machine learning projects on Kubernetes. Examples of such projects include TensorFlow. Additionally, there are plans to incorporate support for additional frameworks, such as MXNet, Pytorch, Chainer, and others. These frameworks can use the GPUs available in the Kubernetes cluster for machine learning tasks since AWS recently announced that Amazon EKS now supports GPU worker instances. Although it is possible to run machine learning workloads on CPU instances, GPU instances have thousands of CUDA cores, which significantly improves performance when it comes to training deep neural networks and processing large data sets. CPU instances are required to run machine learning workloads.

An environment within which Docker images can be constructed as required for this tutorial. Since Docker and the AWS Command Line Interface (CLI) are already installed on the AWS Cloud9 IDE, the authors recommended using it. Next, they explained how to create an

EKS cluster with GPU instances by following the instructions or using the `eksctl` command-line tool that Weaveworks provides.

In order to spawn Jupyter notebooks that have persistent volumes attached to them, Kubeflow requires a default storage class. In Kubernetes, a `StorageClass` is a way to describe the type of storage (for example, the types of EBS volume: `io1`, `gp2`, `sc1`, and `st1`) that an application can request for its persistent storage. This storage can be used to store data that cannot be deleted. The tutorial showed how to create a Kubernetes default storage class that supports dynamic provisioning of persistent volumes backed by the Amazon Elastic Block Store (EBS). The general-purpose SSD volume type was used.

Work 2: MLOps: Continuous delivery and automation pipelines in machine learning

Data science and machine learning (ML) are becoming fundamental skills for solving complex real-world issues. Numerous businesses invest in their data science teams and machine learning (ML) capabilities to develop predictive models that can provide users with business value. Given relevant training data for their use case, data scientists can implement and train an ML model with predictive performance on an offline holdout dataset. The challenge is to construct an integrated ML system and operate it continuously in production.

Because a machine learning system is a software system, the same best practices should be used while developing and running ML systems. On the other hand, machine learning systems are distinct from other types of software in that Data scientists or ML researchers are typically included in a team working on a machine learning project. These individuals strongly emphasize exploratory data analysis, model creation, and experimentation. The deployment process is not as

straightforward as installing an offline-trained machine learning model and using it as a prediction service.

In the context of continuous integration (CI), testing and validating code and components and data, data schemas, and models are essential concerns. A single software package or service is what a CD is about, but it also refers to a system that should automatically deploy another service. CT is a brand-new property that is exclusive to ML systems. Its primary focus is on automatically retraining and serving the models.

Determining the business use case for machine learning (ML) is followed by several steps involved in delivering an ML model to production. These steps can either be accomplished manually or through an automated pipeline. The degree of automation necessary to finish each step in the pipeline is one factor that determines how mature the machine learning process is.

A machine learning pipeline that is already in production continuously provides prediction services to new models trained on new data. In level 1, an engineer will deploy an entire training pipeline that will run automatically and repeatedly to serve the trained model as the prediction service. This pipeline will be used to serve the trained model. In this section, we will discuss the components of the architecture that need to be added to enable machine learning continuous training. Data values skews are substantial shifts in the statistical properties of the data, which cause retraining of the model to be carried out so that it can account for these shifts. A schema skew can occur when an unexpected feature is received, when not all of the expected features are received or when an expected feature is received with an unexpected value.

An optional additional component for level 1 ML pipeline automation is a feature store. Standardizing the definition, storage, and access to features for training and serving can be done with the help of a feature store. By using the feature store as the data source for experimentation,

continuous training, and online serving, ML engineers can prevent a skew that occurs between training and serving. This approach ensures that the components utilized during training will be the same ones utilized during actual service. To assist with data lineage, reproducibility, and comparisons, information regarding each execution of the machine learning pipeline is recorded.

ML engineers can also use it to debug errors and unusual occurrences. If the pipeline was halted because of a failed step, tracking these intermediate outputs allows ML engineers to start the pipeline back up from the most recent step and continue processing without redo the steps that have already been finished. A pointer to the previously trained model can be used if it is necessary to revert to an earlier version of the model or produce evaluation metrics for an earlier model when the pipeline is given new test data. By using these metrics, engineers will be able to compare the performance of a newly trained model to the recorded performance of the model that came before it.

Work 3: Run Amazon EKS Anywhere!

Amazon EKS has seen widespread adoption as the solution of choice for managing Kubernetes clusters on AWS. Amazon announced at re:Invent 2020 a new deployment option for Amazon EKS that enables users to operate the system within on-premise environments. The open-source EKS Distro, which was already available for use, is utilized by EKS Anywhere. Only the creation of production clusters on vSphere environments is supported by this release. Docker makes it possible even to use local computers as part of a development cluster.

When it comes to running Kubernetes in on-premise environments, EKS Anywhere is a fantastic choice. It is simple to install on local machines running Linux or macOS and virtual machines operating within an infrastructure powered by vSphere. The author was excited to learn

about any new features that come with this deployment option. This may be the best solution when subject to any regulatory restrictions or if users want to use this distribution in the environments they manage.

If you want to run EKS Anywhere on a machine, it needs to have the macOS or Ubuntu 20.04 LTS operating system installed, and it also needs to have the Docker 20 version installed. Unfortunately, ARM-based CPUs and the M1 chipsets used by Apple devices are not supported. A folder bearing the cluster's name will be produced during the installation, and the kubeconfig file will be located within this folder. It allows users to connect to their cluster to view their nodes, namespaces, and system workloads using the standard kubectl commands. Cilium is currently being utilized as the CNI Provider within EKS Anywhere.

This enables users to take advantage of the capabilities offered by sandboxed programs within an operating system kernel known as eBPF. The tutorial showed how to develop a basic Nginx deployment and then implement it on a Kubernetes cluster. Following the successful deployment of the workload, users can now access the workload through the EKS Console.

Changes to the Project Environment

Original

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.

Changes

After their on-premises datacenter was connected to the AWS cloud using Direct Connect, the ML Ops team saw an opportunity to refine their computing model. Maximizing the use of their own servers, which are equipped with highly powerful GPUs, would save the company on AWS usage costs for the high-performance GPU-powered EC2 instances. Instead, they decided to shift most of their traditional computing needs by using VMware Cloud for AWS to expand their DRS pool into the cloud. Since this work was not included in the original project goals, the ML Ops team handled all VMware Cloud configurations.

To further maximize the use of their on-premises servers, the ML Ops team also opted to install a second Nvidia Tesla v100 GPU in each of their servers. KubeOps then configured Kubernetes and KubeFlow to exhaust all on-premises nodes in EKS Anywhere before shifting any work toward AWS EKS. This method will also save on data transfer costs in both dollars and time when the microsatellite missions officially commence.

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 helped the ML Ops team become more efficient by following Agile methodology. The first involved managing infrastructure in the same way application developers managed code by using software to quickly and reliably configure environments using the Infrastructure as Code concept. Next, forming cross-functional teams helped break down the departmental silos built over the years. Another critical shift was 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, improved collaboration between infrastructure and development teams fostered a DevOps culture that paid 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 used Agile on infrastructure projects with careful planning and continuous iteration.

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 were regularly shown work in progress to receive early and ongoing feedback. Furthermore, because deliverables were released throughout the project rather than at the end, the ML Ops team delivered value in stages. While, in many cases, all infrastructure was required for a solution to go live, there were frequent opportunities 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 was most straightforward for the ML Ops team was not always 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 have been delayed due to the lack of development environments. Using Agile principles, the operations team first constructed all the development environments in their own Kubernetes namespace, allowing each project to begin. The ML Ops team then examined QA and production environments by iterating on the Kubernetes manifests from the development namespace. Later environments were scheduled dependent on development cycle time and project priorities.

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

Daily Scrum meetings assisted IT operations specialists in meeting 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.

Agile teams should avoid extensive documentation per the Agile Manifesto (Manifesto for Agile Software Development, 2001). Documentation was created as late in the process as possible without delaying the project to minimize waste documenting things that inevitably changed later. The KubeOps team focused on creating visual documentation, concise writing, and informal documentation formats such as hand-drawn sketches or photos.

Project Goals and Objectives

Goals, Objectives, and Deliverables Table

Goal	Supporting objectives	Met/Unmet
1. Created a unified control plane across the AWS and on-premises environment	1.a. Built the AWS EKS cloud environment	Met
	1.b. Configured EKS Anywhere for AstroMined Labs' on-premises datacenter	Met
	1.c. Connected the on-premises and cloud environments with AWS Direct Connect, Istio Service Mesh, and Gloo Mesh	Met
2. Created a managed machine learning pipeline on the unified control plane	2.a. Installed and configured KubeFlow on the newly created unified control plane	Met
	2.b. Installed and configured Prometheus, Grafana, and AlertManager on the unified control plane	Met
3. Tested the environment and prepared the ML Ops team for go-live	3.a. Conducted testing of the KubeFlow machine learning pipeline using sample datasets	Met
	3.b. Produced documentation and end-user training in preparation for the handover	Met

Goals and Objectives Descriptions

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

The first goal of the project was to create a unified control plane across the AWS and on-premises environment. Going forward, AstroMined Labs will be able to deploy, manage, and scale containerized applications running Kubernetes on AWS, thanks to the Amazon EKS control plane that KubeOps built. The new EKS Anywhere (EKS-A) offering brought this same experience to their corporate datacenters using virtual machines and bare-metal servers. To create a single, unified data plane for the KubeFlow Machine Learning pipelines to operate on,

KubeOps connected the on-premises and cloud environments in a seamless fashion. We accomplished 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

Going forward, AstroMined labs will be able to deploy, manage, and scale containerized applications running Kubernetes on AWS, thanks to Amazon EKS, thanks to the Amazon EKS control plane that KubeOps built. Using Amazon EKS, we distributed 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 made 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.

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. By utilizing compute environments across both environments, this projects allowed the ML Ops team 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 their on-premises datacenter.

Realizing the benefits of a hybrid cloud environment could only occur when developers could consistently deploy and run their applications on a consistent infrastructure. Whether the

infrastructure was in a physical or virtual location and any other specifics of the underlying infrastructure were abstracted away from developers and applications.

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

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 was necessary to connect the on-premises and cloud environments in a seamless fashion. We accomplished this with a combination of technologies, namely AWS Direct Connect, Istio Service Mesh, and Gloo Mesh.

The AWS Direct Connect cloud service connected AWS resources with on-premises resources by providing the shortest network path available. Because network traffic was routed through the AWS global network and never touched the public internet, there was 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 was Cologix COL2, located in Columbus, Ohio. It was possible to establish a direct connection with both the US-East-2 and GovCloud AWS regions from this location, which will 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.

With the two EKS control planes now physically connected, the main challenge became connecting the services hosted within those control planes to create a unified management layer.

The design of a service mesh solved these service connectivity challenges so that developers could focus on the business logic of their services rather than the underlying infrastructure.

KubeOps deployed an Istio Service Mesh, which was the most widely used and well-documented production service mesh. Gloo Mesh served as an abstraction layer above Istio and AWS App Mesh and prioritized security, simplified multicluster operation, and provided global failover routing for services across clusters regardless of where they run. Gloo Mesh supported the Kubernetes clusters on EKS-Anywhere, EKS, vSphere VMs, and bare metal servers.

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

The second goal of this project was 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 was 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. Since The ML Ops team at AstroMined Labs will be responsible for deploying their required toolsets on top of the KubeFlow platform, KubeOps only deployed the basic scaffolding.

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 was first installed and configured. The next step was to configure Prometheus to communicate with the AlertManager. As a final step, we created 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 utilized the Quick Start Reference Deployments provided by AWS. These documents provided AWS Cloud Formation templates and detailed guidance which allowed us to follow the best practices developed by the experts at AWS. Once the deployment was complete, the ML Ops team had complete observability of their entire infrastructure through Grafana dashboards.

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

This final goal of the project had a dual utility of both testing the platform and training the AstroMined Labs ML Ops team on its use. Since the ML Ops team decided to deploy more Kubernetes nodes with their on-premises infrastructure, we used that as an opportunity to train them on the process. Next, the KubeFlow Project provided convenient example pipelines (samples, 2021) to test the deployment. These examples came complete with full datasets for training machine learning models, and Kubernetes manifests to automate the deployments. The project's final phase was to compile all 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. Conducted Testing of the KubeFlow Machine Learning Pipeline

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

Objective 3.b. Compiled Documentation and End-user Training

The final phase of the project was 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.

Project Timeline

Since the Agile methodology was used for this project, all work was done in two-week sprints. The sprints were deliberately not scheduled full to allow for change orders, testing, and redeployment of failed components. All projected sprint start and end dates were adhered to. The project had an actual end date of 5/22/2022, and the sprint schedule that was followed is outlined in the following table:

Milestone or deliverable	Duration (hours or days)	Projected and actual start date	Projected and actual end date
1.a.i. To facilitate a unified virtual network and conform to AWS best practices, created a virtual private cloud (VPC) that was configured with public and private subnets.	4 hours	Sprint 1 02/14/2022	2/27/2022
1.a.ii. To enable highly available architecture that is less prone to failure, the VPC spanned three Availability Zones.	4 hours	Sprint 1 02/14/2022	2/27/2022
1.a.iii In the public subnets, created a demilitarized zone (DMZ) to allow for internet access to resources in the private subnets using managed NAT gateways.	4 hours	Sprint 1 02/14/2022	2/27/2022
1.a.iv In the private subnets, configured 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 02/14/2022	2/27/2022
1.a.v In the DMZ, Ubuntu bastion hosts were configured as an ASG of EC2 instances spanning all three AZs. This allowed inbound Secure Shell (SSH) access to the Kubernetes nodes without exposing them directly to the open internet.	4 hours	Sprint 1 02/14/2022	2/27/2022
1.a.vi The bastion hosts were configured with the Kubernetes command line interface (kubectl) command line interface for managing the Kubernetes cluster.	1 day	Sprint 1 02/14/2022	2/27/2022
1.a.vii The final step was to configure the ASG of Kubernetes nodes as an Amazon EKS cluster to provide the Kubernetes control plane.	3 days	Sprint 1 02/14/2022	2/27/2022

Milestone or deliverable	Duration (hours or days)	Projected and actual start date	Projected and actual end date
1.b.i. To fully realize the benefits of a highly available architecture, the first step was to build two vCenter Server Appliances (VCSA) and connect them using vCenter-HA.	1 day	Sprint 2 02/28/2022	3/13/2022
1.b.ii. To fully realize the benefits of resource pooling and distributed resource scheduling, provisioned two ESXi host servers for each VCSA.	1 day	Sprint 2 02/28/2022	3/13/2022
1.b.iii. Each ESXi host housed VMs for a minimum of one Kubernetes Control Node and two Kubernetes Worker Nodes each.	1 day	Sprint 2 02/28/2022	3/13/2022
1.b.iv. vSphere DRS was 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 02/28/2022	3/13/2022
1.b.v. The AWS EKS Anywhere control plane was installed and configured to use the vSphere DRS cluster as a managed node group.	3 days	Sprint 2 02/28/2022	3/13/2022
1.c.i. Connected the AWS Cloud VPC with the on-premises datacenter using AWS Direct Connect.	3 days	Sprint 3 03/14/2022	3/27/2022
1.c.ii. Created a control plane network in both the AWS cloud and the on-premises datacenter using Istio Service Mesh.	2 days	Sprint 3 03/14/2022	3/27/2022
1.c.iii. Connected the Istio Service Meshes into a unified control plane using Gloo Mesh.	2 days	Sprint 3 03/14/2022	3/27/2022
2.a.i. Validated the EKS Cluster created in the first 3 phases.	1 day	Sprint 4 03/28/2022	4/10/2022
2.a.ii. Created a storage class for the persistent data volumes used by the control plane.	1 day	Sprint 4 03/28/2022	4/10/2022
2.a.iii. Created a Kubernetes Namespace for the KubeFlow deployment.	1 day	Sprint 4 03/28/2022	4/10/2022
2.a.iv. Deployed the Kubernetes manifest YAML files provided by the KubeFlow project.	2 days	Sprint 4 03/28/2022	4/10/2022
2.a.v. Validated the installation succeeded and set administrative controls.	2 days	Sprint 4 03/28/2022	4/10/2022

Milestone or deliverable	Duration (hours or days)	Projected and actual start date	Projected and actual end date
2.b.i. Deployed Prometheus and AlertManager using the AWS Quick Start.	4 hours	Sprint 5 04/11/2022	4/24/2022
2.b.ii. Configured Prometheus to collect data from all parts of the infrastructure	3 days	Sprint 5 04/11/2022	4/24/2022
2.b.iii. Configured AlertManager to alert the ML Ops Team when infrastructure issues threaten the reliability of the platform	2 days	Sprint 5 04/11/2022	4/24/2022
2.b.iv. Deployed Grafana using the AWS Quick Start.	4 hours	Sprint 5 04/11/2022	4/24/2022
2.b.v. Configured Grafana Dashboards to use the Prometheus data sources to provide real-time observability into the performance of the platform.	2 days	Sprint 5 04/11/2022	4/24/2022
3.a.i. Deployed 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.	Planned: 1 day Actual: 5 days due to expansion of on-premises deployments	Sprint 6 04/25/2022	5/8/2022
3.a.ii. Deployed a Jupyter Hub notebook server that integrated into the workflow of the Data Scientists and Machine Learning Engineers at AstroMined Labs.	1 day	Sprint 6 04/25/2022	5/8/2022
3.a.iii. Worked with senior members of the Data Science and Machine Learning staff to demonstrate usage of the notebook server and determined the likely points of confusion that were documented for junior members of the staff.	4 days	Sprint 6 04/25/2022	5/8/2022
3.a.iv. Demonstrated the configuration of Prometheus, AlertManager, and Grafana to the ML Ops team.	Planned: 4 days Actual: 3 days and moved to Sprint 7	Sprint 7 05/09/2022	5/22/2022
3.b.i. Completed architecture diagrams of the infrastructure, including all diagrams included in this document.	4 hours	Sprint 7 05/09/2022	5/22/2022
3.b.ii. Linked all official documentation sources for the various components of this project	4 hours	Sprint 7 05/09/2022	5/22/2022
3.b.iii. Compiled a repository of example projects to be used for training purposes	1 day	Sprint 7 05/09/2022	5/22/2022
3.b.iv. Recorded videos training demonstrating how to deploy the example projects	6 days	Sprint 7 05/09/2022	5/22/2022

Unanticipated Requirements

The ML Ops team realized an opportunity to improve their computing model after their on-premises datacenter was successfully connected to the AWS cloud using Direct Connect. The business would reduce its AWS usage costs for high-performance GPU-powered EC2 instances if it made the most of its own servers' highly powerful graphics processing units (GPUs) and used them to their full potential. Instead, they concluded that most of their traditional computing requirements could be satisfied by expanding their DRS pool into the cloud by utilizing VMware Cloud for Amazon Web Services. The ML Ops team managed all VMware Cloud configurations because this particular piece of work was not part of the initial project goals.

The ML Ops team decided to install a second Nvidia Tesla v100 GPU in each of their on-premises servers. This was done to get the most out of their servers. After that, KubeOps configured Kubernetes and KubeFlow to use up all the on-premises nodes in the EKS Anywhere cluster before moving any work to the AWS EKS cluster. When the microsatellite missions are officially started, this method will save money on data transfer costs, both in terms of dollars and time.

Conclusions

Kubeflow has rapidly evolved into an indispensable tool for the ML Ops team. It helped organize their projects, took advantage of cloud computing, and enabled machine learning engineers to dig in and develop the finest models they could build. Kubeflow was designed to make it easier for 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 mattered for AstroMined Labs, which was the main driver behind this project to begin with. That metric was the time it 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 measured the time the process took with their old workflow, and then produced the same model using same dataset in the KubeFlow pipeline. By comparison, the KubeFlow process took less than 40% of the time than the original workflow, which exceeded even our bold predictions of 30%. AstroMined labs had previously deemed that even a 10% improvement is worth the money, time, and effort involved in this project, so they were thrilled with these results.

The ML Ops team also estimated that model accuracy will improve by as much as 5%, and we have found that to be a fair estimate that has been proven by model datasets. These accuracy gains came from the increased computing power, mainly from cloud GPUs, which allowed deeper and wider neural networks to be created. This meant that more features could be used as inputs to the models, and more layers of deep learning could take place, which generally increases accuracy of models with most datasets.

The final metric that was used to determine this project's success is the reliability of the platform and the amount of maintenance that is needed to be performed by the ML Ops team.

We have estimated 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 it was with their original infrastructure, as the platform is largely self-healing. These factors can only be measured on a long-term basis, so data to prove our predictions is not yet available.

Project Deliverables

Appendix A shows how AWS Direct Connect connects AWS and on-premises resources using the shortest network path. Because network data is routed through AWS and never the public internet, bottlenecks and latency spikes are less likely during transit. AstroMined Labs' closest Direct Connect location is Cologix COL2 in Columbus, Ohio. This facility will have direct connections to AWS's US-East-2 and GovCloud regions, facilitating future collaboration with NASA in GovCloud. Several Amazon Partner Network providers built 100 Gbps fiber backbone links between Pittsburgh and Columbus to ensure availability, speed, and latency.

Appendix B shows a small sample of the Prometheus metrics that were configured to be collected. These metrics focus directly on the Kubernetes deployments. The Kubernetes metrics server was enabled to provide Prometheus with the most detailed information possible about the cluster, with a single API endpoint that provides metrics for the whole control plane.

Appendix C first shows the configuration of an AlertManager rule in Prometheus, which is to alert on “High Pod Memory” in this example. The next screenshot shows how AlertManager was configured to integrate with Slack, so that AstroMined Labs could receive and respond to alerts with their chosen workflow. The third screenshot shows an example alert being triggered within Prometheus. Finally, the final screenshot of Appendix C shows the alert being received in the Slack workspace of AstroMined Labs.

Appendix D shows a Grafana dashboard that was built to display the metrics that are gathered by Prometheus. Observability into their infrastructure was a primary goal for AstroMined Labs, and this dashboard helps provide real-time visibility into the performance of their Kubernetes Clusters.

Appendix E shows a Kubernetes YAML manifest file that was created to simplify the deployment of KubeFlow. This YAML file allows the entire cluster to be deployed with a single command, which ensures repeatable and simple processes that the ML Ops team can use to automate these deployments in the future.

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/86df7dcfd896fc2674f757a2463eba-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/>

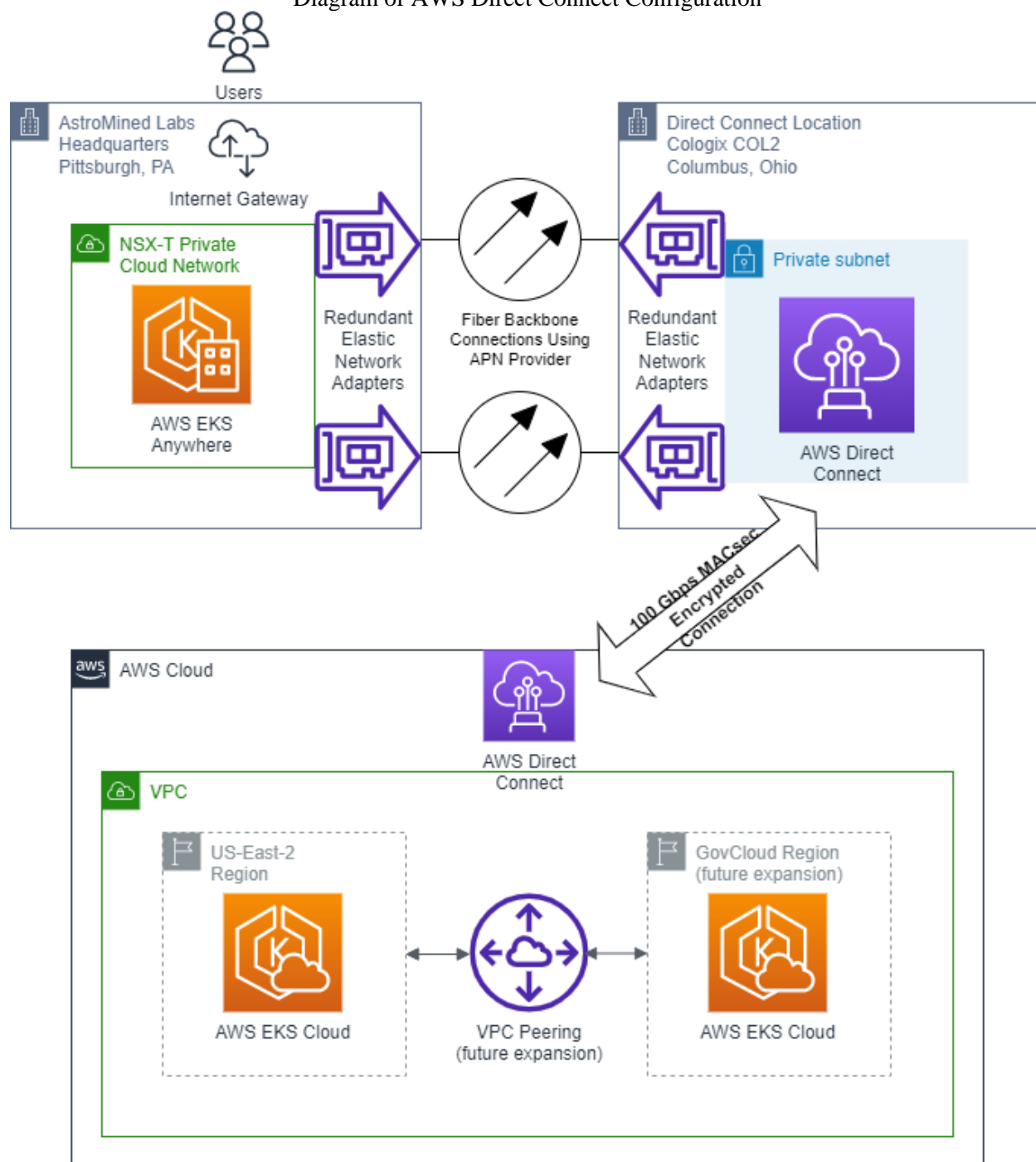
MLOps: Continuous delivery and automation pipelines in machine learning / Google Cloud. (n.d.). Google Cloud; cloud.google.com. Retrieved May 28, 2022, from <https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning>

Tan, E.-H., & Gupta, A. (2018, September 30). *Kubeflow on Amazon EKS* / Amazon Web Services. Amazon Web Services; aws.amazon.com. <https://aws.amazon.com/blogs/opensource/kubeflow-amazon-eks/>

Alemdar, E. (2021, September 14). *Run Amazon EKS Anywhere!* Run Amazon EKS Anywhere!; www.kloia.com. <https://www.kloia.com/blog/run-amazon-eks-anywhere>

Appendix A

Diagram of AWS Direct Connect Configuration



Appendix B

Screenshot of Prometheus Configured to Collect Kubernetes Metrics

AlertmanagerPrometheus Time Series CollectioCluster Monitoring for Kubernetes

←→↺🏠⚠️ Not secure | 192.168.1.200:30000/targets?search=🔗☆🟢🕒🚩🔧☰🗄️👤

Prometheus AlertsGraphStatus ▾ Help⚙️🌙🌑

Targets

AllUnhealthyCollapse All

🔍 Filter by endpoint or labels

kube-state-metrics (1/1 up)show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://kube-state-metrics.kube-system.svc.cluster.local:8080/metrics	UP	<div>instance="kube-state-metrics.kube-system.svc.cluster.local:8080"</div> <div>job="kube-state-metrics"</div>	13.581s ago	7.318ms	

kubernetes-apiservers (1/1 up)show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://192.168.1.200:6443/metrics	UP	<div>instance="192.168.1.200:6443"</div> <div>job="kubernetes-apiservers"</div>	14.774s ago	119.361ms	

kubernetes-cadvisor (3/3 up)show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://kubernetes.default.svc/api/v1/nodes/k8s-worker2/proxy/metrics/cadvisor	UP	<div>beta_kubernetes_io_arch="amd64"</div> <div>beta_kubernetes_io_os="linux"</div> <div>instance="k8s-worker2"</div> <div>job="kubernetes-cadvisor"</div> <div>kubernetes_io_arch="amd64"</div> <div>kubernetes_io_hostname="k8s-worker2"</div> <div>kubernetes_io_os="linux"</div>	13.418s ago	27.221ms	

kubernetes-nodes (3/3 up)show less

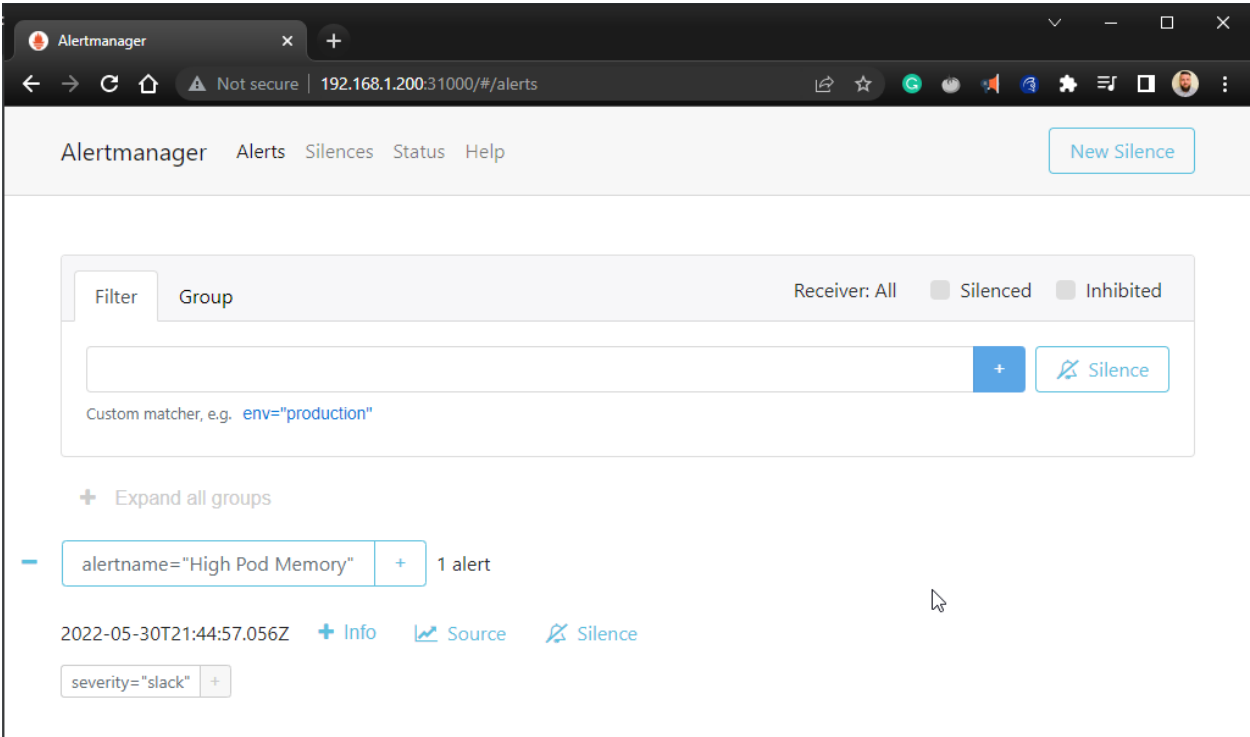
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://kubernetes.default.svc/api/v1/nodes/k8s-worker1/proxy/metrics	UP	<div>beta_kubernetes_io_arch="amd64"</div> <div>beta_kubernetes_io_os="linux"</div> <div>instance="k8s-worker1"</div> <div>job="kubernetes-nodes"</div> <div>kubernetes_io_arch="amd64"</div> <div>kubernetes_io_hostname="k8s-worker1"</div> <div>kubernetes_io_os="linux"</div>	10.691s ago	118.230ms	

kubernetes-service-endpoints (5/5 up)show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.175.7:9153/metrics	UP	<div>instance="192.168.175.7:9153"</div> <div>job="kubernetes-service-endpoints"</div> <div>k8s_app="kube-dns"</div> <div>kubernetes_io_cluster_service="true"</div> <div>kubernetes_io_name="CoreDNS"</div> <div>kubernetes_name="kube-dns"</div> <div>kubernetes_namespace="kube-system"</div>	12.776s ago	3.083ms	

Appendix C

Screenshot of AlertManager rule in Prometheus



AlertManager configured to integrate with Slack

slack app directory

Q Search App Directory

Browse


Manage

Build

AstroMined

▼

[← Manage Apps](#)



Prometheus Alert Manager

Open in Slack

i

This app was created by a member of your workspace, AstroMined.
The permissions this app requests have not been reviewed by Slack.

Description

Configuration

Authorizations

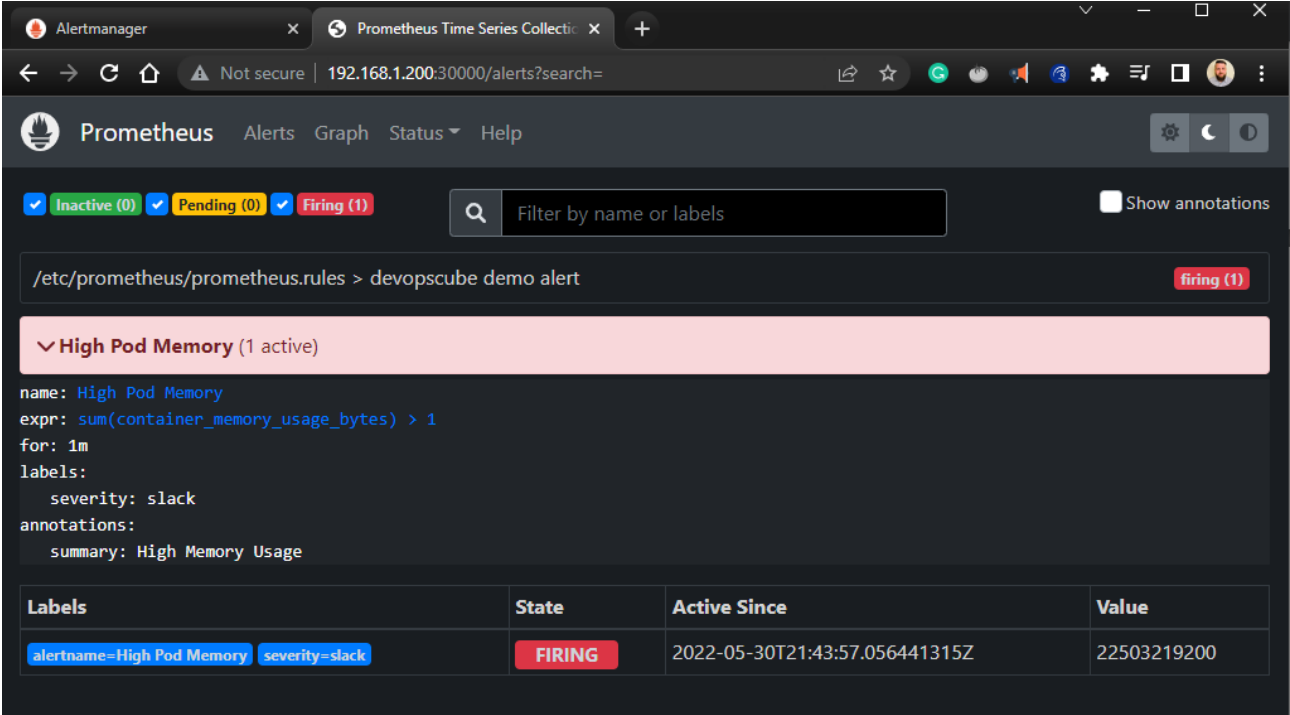
On behalf of the app, Prometheus Alert Manager can:

- Post messages to specific channels in Slack
- Post messages to channel

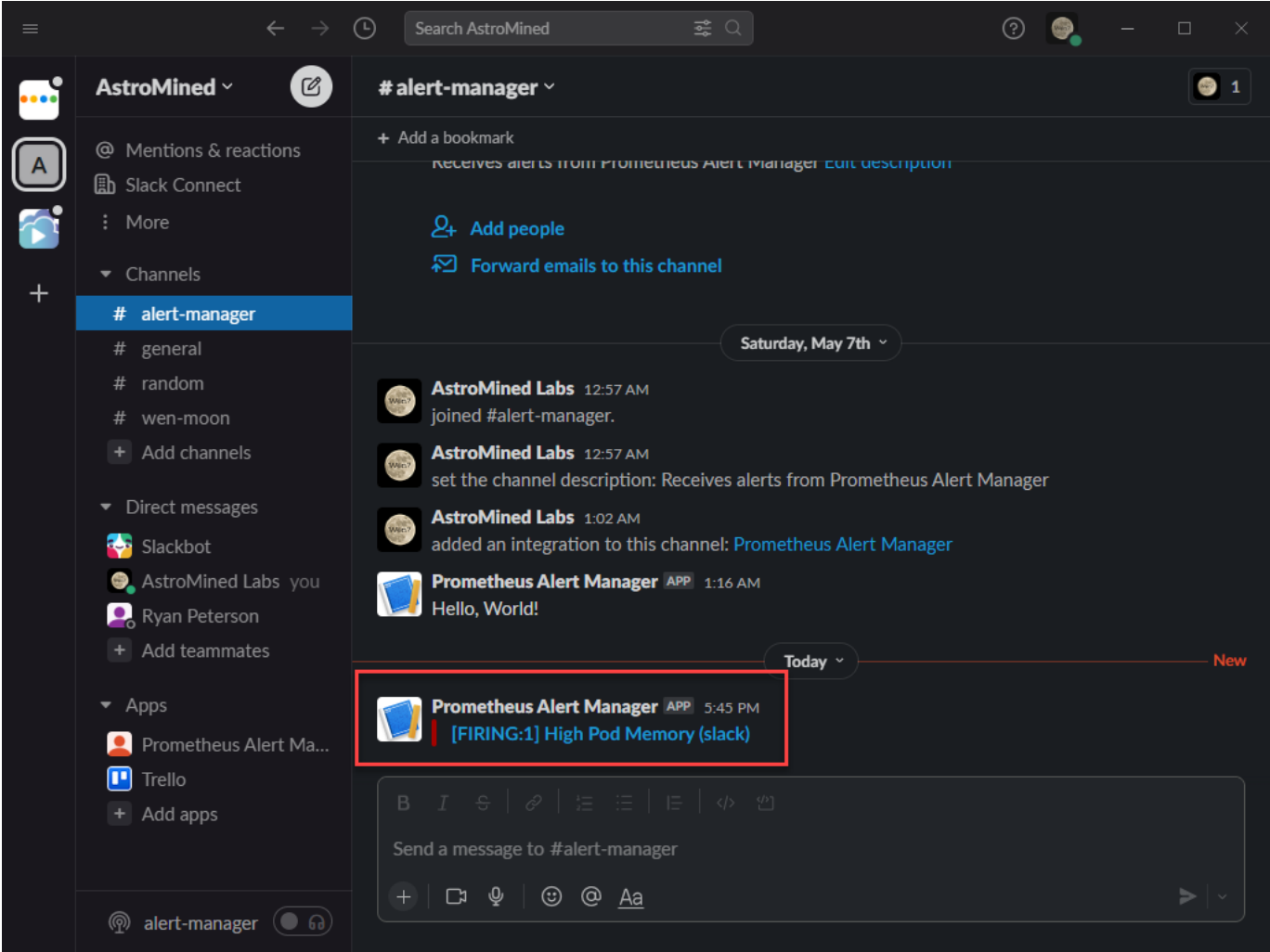
Your authorization

AstroMined Labs on May 7, 2022

example alert being triggered within Prometheus

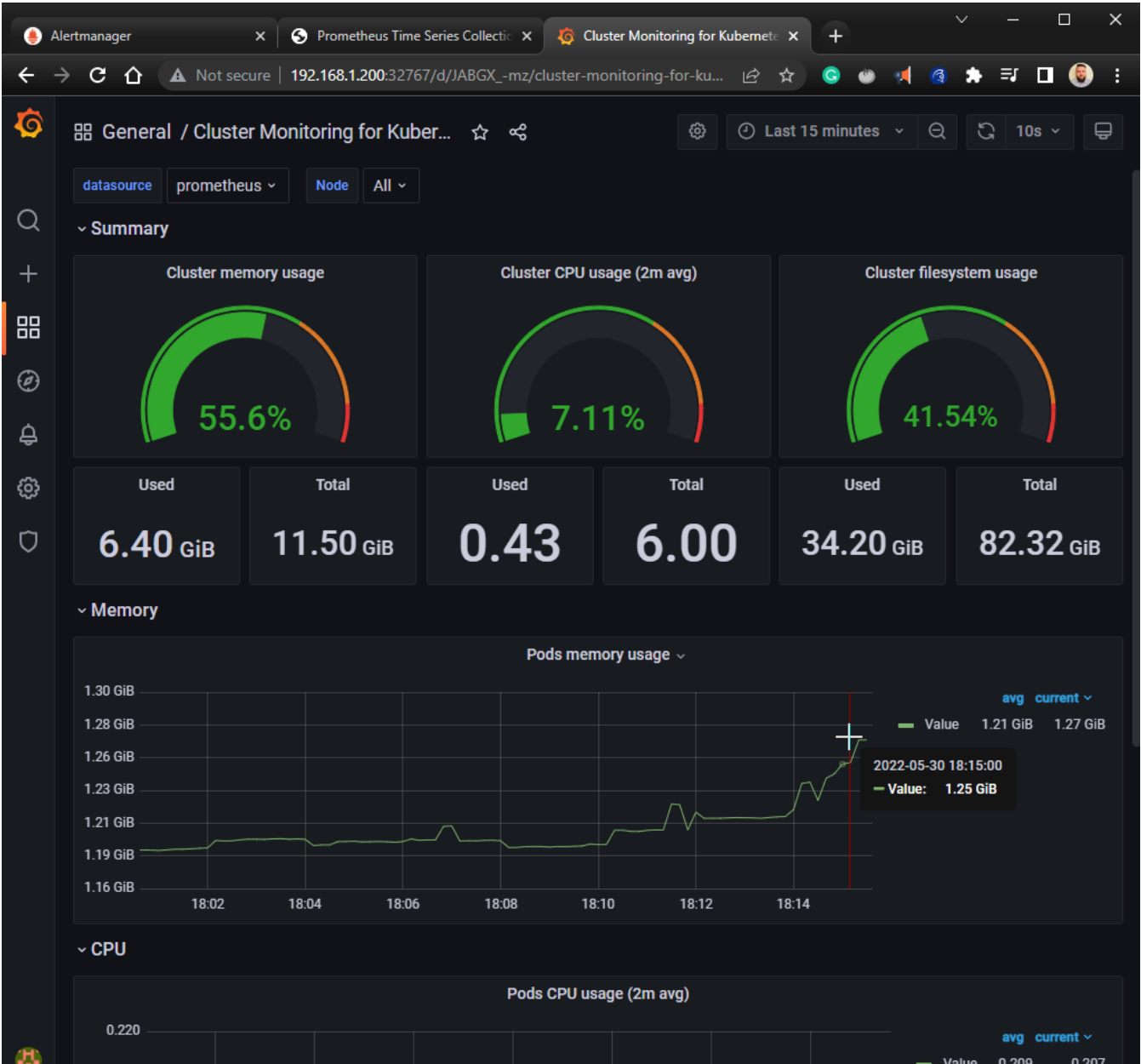


the alert being received in the Slack workspace of AstroMined Labs.



Appendix D

Grafana dashboard that was built to display the metrics that are gathered by Prometheus



Appendix E

Kubernetes YAML Manifest for Deploying KubeFlow

```

apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
# Cert-Manager
- ../common/cert-manager/cert-manager/base
- ../common/cert-manager/kubeflow-issuer/base
# Istio
- ../common/istio-1-11/istio-crds/base
- ../common/istio-1-11/istio-namespace/base
- ../common/istio-1-11/istio-install/base
# OIDC Authservice
- ../common/oidc-authservice/base
# Dex
- ../common/dex/overlays/istio
# KNative
- ../common/knative/knative-serving/overlays/gateways
- ../common/knative/knative-eventing/base
- ../common/istio-1-11/cluster-local-gateway/base
# KubeFlow namespace
- ../common/kubeflow-namespace/base
# KubeFlow Roles
- ../common/kubeflow-roles/base
# KubeFlow Istio Resources
- ../common/istio-1-11/kubeflow-istio-resources/base

# KubeFlow Pipelines
- ../apps/pipeline/upstream/env/platform-agnostic-multi-user
# KFServing
- ../apps/kfserving/upstream/overlays/kubeflow
# Katib
- ../apps/katib/upstream/installs/katib-with-kubeflow
# Central Dashboard
- ../apps/centraldashboard/upstream/overlays/kserve
# Admission Webhook
- ../apps/admission-webhook/upstream/overlays/cert-manager
# Notebook Controller
- ../apps/jupyter/jupyter-web-app/upstream/overlays/istio
# Jupyter Web App
- ../apps/jupyter/notebook-controller/upstream/overlays/kubeflow
# Profiles + KFAM
- ../apps/profiles/upstream/overlays/kubeflow
# Volumes Web App
- ../apps/volumes-web-app/upstream/overlays/istio
# Tensorboards Controller
- ../apps/tensorboard/tensorboard-controller/upstream/overlays/kubeflow
# Tensorboard Web App
- ../apps/tensorboard/tensorboards-web-app/upstream/overlays/istio
# Training Operator
- ../apps/training-operator/upstream/overlays/kubeflow
# User namespace
- ../common/user-namespace/base

# KServe
- ../contrib/kserve/kserve
- ../contrib/kserve/models-web-app/overlays/kubeflow

```