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](#_Toc441471473)

[Review of Other Work 3](#_Toc441471474)

[Changes to the Project Environment 3](#_Toc441471475)

[Methodology 3](#_Toc441471476)

[Project Goals and Objectives 3](#_Toc441471477)

[Project Timeline 4](#_Toc441471478)

[Unanticipated Requirements 4](#_Toc441471479)

[Conclusions 4](#_Toc441471480)

[Project Deliverables 4](#_Toc441471481)

[References 6](#_Toc441471482)

[Appendix A: 7](#_Toc441471483)

[Appendix B: 8](#_Toc441471484)

[Appendix C: 9](#_Toc441471485)

[Appendix D: 10](#_Toc441471486)

Note: On the pages that follow, keep the bolded main heading but delete the prompt material once you’ve reviewed and made the necessary additions and adjustments. Also, update the running head and title page. Remove this note as well

# 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 exisitng 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). It was created as late in the process as possible without delaying the project to minimize waste documenting things that inevitably changed later. The ML Ops team focused on creating visual documentation, concise writing, and informal documentation formats such as hand-drawn sketches or photos.

# Project Goals and Objectives

In this section, provide a detailed explanation on how the project goals and objectives were met. If goals and/or objectives were not accomplished, that’s fine as long as you provide a reason as to why.

**From the Proposal:** Copy your Goals, Objectives, and Deliverables section. You may also copy the Goals table as it helps the evaluator more clearly understand the hierarchy.

**What to Adjust:** Make sure this reads in past tense. You may remove the deliverables. You must include a clear, obvious statement with each goal and objective about how it was successfully completed. If you decide to leave the deliverable, connecting them with your objective’s success statement is an effective approach.

# Project Timeline

In this section, create a timeline that deals with planned and actual durations and the final project’s start and end dates. Include a paragraph after the timeline that compares the durations and provides information about where times did or did not work out as planned.

**From the Proposal**: Copy the timeline table.

**What to Adjust**: Create new headings that include: Planned Duration, Actual Duration, Actual Start Date, and Actual End Date. Keep the Deliverables column. Remove or rename the proposal’s date columns.

Note: Because the project is now completed, all the dates must be in the recent past. The evaluator will not compare them to the proposal dates.

# 

# Unanticipated Requirements

In this section, describe the requirements that emerged during implementation. These are not timing related (those go in the timeline) but relate to hardware, software, or personnel-type issues that arose during the project’s implementation. Additionally, be sure to explain how they were resolved or why they were not solved. This is a new section.

# Conclusions

In this section, explain the actual project accomplishments and immediate and potential future effects of the completed project. Include at least one measurable metric that was used to prove the project’s success.

**From the Proposal:** Copy the Outcome section.

**What to Adjust:** Change your Outcome to past tense. Include information about immediate and potential effects from the completed project. Provide specific examples where relevant. Update the measurable metric information from the proposal. If there is none, add new material that specifies what was measured and what measurement indicated success.

# Project Deliverables

In this section, describe at least three project artifacts (examples) that appear in the appendices. The artifacts should provide a logical display to substantiate the described successes and benefits of the completed project. Refer to each appendix item (included after the References section as Appendix A, Appendix B, etc.) and describe how it demonstrates evidence of the project’s completion. Artifacts may include such things as: code samples, screen shots, photos, flowcharts, process diagrams, tables, graphs, network diagrams (before and after), training materials or related documents (e.g., policies), etc. This is a new Section.

# References

List all the outside sources that the narrative refers to in-text. For in-text and reference list citations, please refer to the web link in the Course of Study or visit the WGU Writing Center.

Note: Ensure that you have an in-text citation for each full citation and vice versa. Those citation pairs must match up according to APA formatting. For example:

**In-text found in the body of the document:**

The article also states design elements shouldn’t be confusing and that mobile apps should be designed with disruption in mind as the user will frequently change tasks while using the mobile device. (Osborne, 2018)

**Full citation appearing in this section:**

Osborne, B. (2018). 6 Mobile App Development Best Practices Worth a Glance. Retrieved from https://www.topcoder.com/blog/6-mobile-app-development-best-practices

# Appendix A

# Title of Appendix

Put any supporting material in these appendices. Add additional or delete superfluous appendices as needed.

# Appendix B

# Title of Appendix

Put any supporting material in these appendices. Add additional or delete superfluous appendices as needed.

# Appendix C

# Title of Appendix

Put any supporting material in these appendices. Add additional or delete superfluous appendices as needed.