into downstream software products, while the DevOps team handles the infrastructure and configuration of the machine for model development. This monolithic style of working results in a machine learning process that is not reusable, difficult to scale and maintain, and even tougher to audit and perform model improvement, and it is easily fraught with errors and unnecessary complexities.

However, by incorporating the microservice design pattern to machine learning development, we can address a host of these concerns and really streamline the productionalization process.

Kubeflow

Kubeflow is a platform that is created to enhance and simplify the process of deploying machine learning workflows on Kubernetes. Using Kubeflow, it becomes easier to manage a distributed machine learning deployment by placing components in the deployment pipeline such as the training, serving, monitoring, and logging components into containers on the Kubernetes cluster.

The goal of Kubeflow is to abstract away the technicalities of managing a Kubernetes cluster so that a machine learning practitioner can quickly leverage the power of Kubernetes and the benefits of deploying products within a microservice framework. Kubeflow has its history as an internal Google framework for implementing machine learning pipelines on Kubernetes before being open sourced late 2017.

Table 46-1 is a sample of some of the components that run on Kubeflow.

Table 46-1. *Sample of Kubeflow Components*

Component	Description
 **Chainer**	**Chainer** is a define-by-run deep learning neural network framework. It also supports multi-node distributed deep learning and deep reinforcement algorithms.
 **Jupyter**	**Jupyter** provides a platform for the rapid prototyping and easy sharing of reproducible codes, equations, and visualizations.
 **ksonnet**	**ksonnet** provides a simple way to create and edit Kubernetes configuration files. Kubeflow makes use of ksonnet to help manage deployments.
 **Istio**	**Istio** eases microservice deployments by providing a uniform way to connect, secure, control, and observe services.
 **Katib**	**Katib** is a deep learning framework agnostic hyper-parameter tuning framework. It is inspired by Google Vizier.
 **MXNet**	**MXNet** is a portable and scalable deep learning library using multiple frontend languages such as Python, Julia, MATLAB, and JavaScript.

(*continued*)

Table 46-1. (*continued*)

Component	Description
 PyTorch	**PyTorch** is a Python deep learning library developed by Facebook based on the Torch library for Lua, a programming language.
 NVIDIA TensorRT	**TensorRT** is a platform for high-performance and scalable deployment of deep learning models for inference.
 Seldon	**Seldon** is an open source platform for deploying machine learning models on Kubernetes.
 TensorFlow	**TensorFlow** provides an ecosystem for the large-scale productionalization of deep learning models. This includes distributed training using TFJob, serving with TF Serving, and other Tensorflow Extended components such as TensorFlow Model Analysis (TFMA) and TensorFlow Transform (TFT).

Working with Kubeflow

1. **Set up a Kubernetes cluster on GKE.**

   ```
   # create a GKE cluster
   gcloud container clusters create ekaba-gke-cluster

   # view the nodes of the kubernetes cluster on GKE
   kubectl get nodes
   ```

2. **Create OAuth client ID to identify Cloud IAP:** Kubeflow uses Cloud Identity-Aware Proxy (Cloud IAP) to connect to Jupyter and other running web apps securely. Kubeflow uses email addresses for authentication. In this section, we'll create an OAuth client ID which will be used to identify Cloud IAP when requesting access to a user's email account: