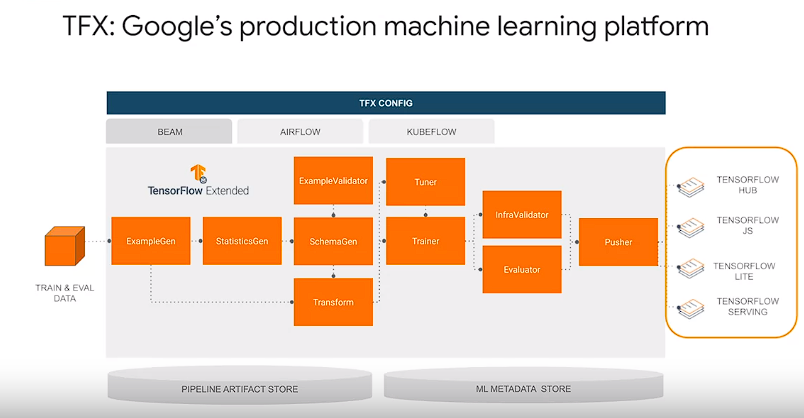
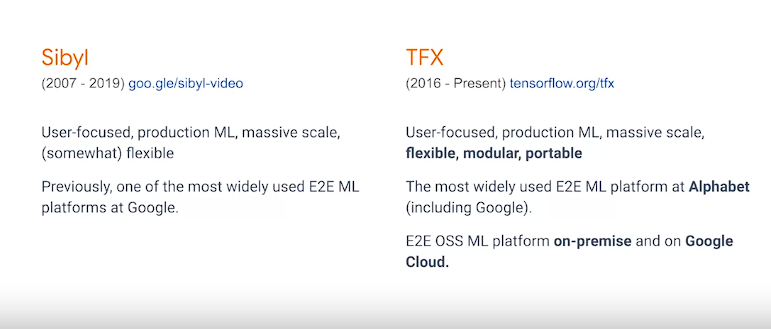
Certainly, here are the points extracted from the passage in a more detailed manner:

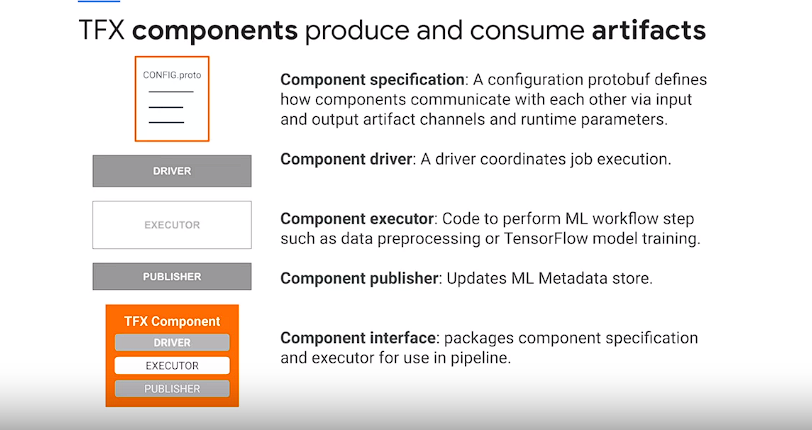
1. **Introduction of Doug Kelly**:
   * Doug Kelly is introduced as a Machine Learning Solutions Engineer and instructor at the Advanced Solution Lab within Google Cloud.
2. **Discussion Topic: TensorFlow Extended (TFX)**:
   * The passage focuses on discussing TensorFlow Extended (TFX), which is described as Google's production Machine Learning platform based on TensorFlow.
3. **Usage and Open Sourcing**:
   * TFX is noted to be widely used internally at Google and was fully open-sourced in 2019, allowing broader access and contributions from the developer community.



1. **Features of TFX**:
   * TFX provides a flexible configuration framework and shared libraries to facilitate the integration of common machine-learning tasks.
   * It assists in managing the entire machine learning project life cycle, from initial prototyping to deploying models into production.
2. **Orchestration and Portability**:
   * TFX is designed to orchestrate machine learning workflows, making them portable across different environments and orchestration frameworks.
   * It is capable of being deployed on various computing platforms, including on-premise setups and Cloud services such as Google Cloud.
3. **Supported Deployment Targets**:
   * TFX supports various deployment targets for TensorFlow models, including TF Serving, TF Lite, TensorFlow JS, and TFO (TensorFlow Object).
4. **Integration with Google Cloud Services**:
   * TFX runs on top of AI Platform pipelines, which seamlessly integrates with several managed Google Cloud services like AI Platform training, AI Platform prediction, and Dataflow for distributed data processing.
5. **Benefits of TFX**:
   * TFX simplifies distributed compute resources, increases development and experimentation velocity, and automates machine learning operational processes for both individual and multiple machine learning pipelines.
6. **Wide Adoption at Alphabet**:
   * TFX is described as the most widely used machine learning platform at Alphabet, supporting tens of thousands of user and programmatic machine learning pipelines across Alphabet subsidiary companies such as DeepMind, Verily, and Waymo.
7. **Open Source Community and Corporate Adoption**:
   * Since its open-sourcing in 2019, TFX has garnered an active open-source community and corporate partners who have extended its capabilities for their own machine learning production workflows.
8. **Examples of Usage**:
   * Examples of companies leveraging TFX include Twitter, Airbus, SAP, and Yahoo! Japan, each utilizing TFX for various machine learning applications tailored to their needs.
9. **Evolution and Best Practices**:
   * TFX is portrayed as the latest evolution in Google's machine learning and pipeline infrastructure, incorporating best practices learned from over a decade of industrial-scale machine learning at Google.
10. **Leveraging TFX for Project Success**:
    * Implementing machine learning workflows with TFX is suggested to leverage Google's best practices and infrastructure, thereby increasing the probability of project success.
11. **Support on Google Cloud**:
    * Furthermore, running TFX on Google Cloud is highlighted as supporting project success at scale, given Google's robust infrastructure and services

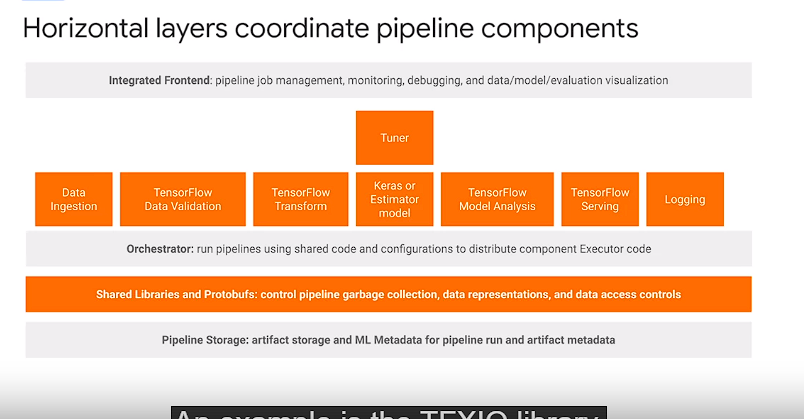


1. **TFX Components**:
   * TFX components are modular implementations of machine learning tasks within a pipeline. They incorporate Google's best practices on tasks like data partitioning, validation, and transformation.
   * Each component produces and consumes structured data representations called **artifacts**, allowing data transfer between components during pipeline execution.
2. **Elements of a Component**:



* + Components consist of five elements:
    - Component specification (or component specs) define inputs, outputs, and runtime parameters using protocol buffers.
    - A driver class coordinates job execution, artifact retrieval, and primary executor job.
    - An executor class performs the actual computation on input artifacts.
    - A component interface packages the specification, driver, and executor.
    - A publisher logs component runs and metadata, writing output artifacts to the storage.

1. **Execution of TFX Components**:
   * Components execute sequentially at runtime:
     + The driver reads runtime parameters and retrieves required artifacts.
     + The executor performs computation and generates output artifacts.
     + The publisher logs component runs and metadata, writing output artifacts to storage.
2. **TFX Pipelines**:
   * TFX pipelines are sequences of components linked by a directed acyclic graph (DAG) of artifact dependencies.
   * Components communicate through input and output channels, with channels serving as abstract connections between producers and consumers of data.
3. **Parameters in Pipelines**:
   * Parameters allow for changing pipeline behavior without altering code. They enable experimentation with different configurations, such as training steps or data splits.
4. **Metadata Store**:
   * TFX implements a metadata store using the ML metadata library to standardize metadata definition, storage, and querying for ML pipelines.
   * Metadata is stored in a relational backend, which could be a local SQL database or a managed database in production deployments.
5. **Orchestrators**:
   * Orchestrators coordinate pipeline runs, ensuring consistency in execution order, logging, retries, failure recovery, and parallelization of data processing.
   * They schedule component executions based on the DAG of artifact dependencies.
6. **Task and Data-Aware Pipelines**:
   * TFX pipelines are both task and data-aware, meaning they can be run manually as tasks and automatically check for necessary re-computation based on changes in artifact dependencies.
7. **TFX Horizontal Layers**:
   * TFX horizontal layers consist of shared libraries and protobufs for defining abstractions that simplify pipeline development across different environments.
   * There are four layers: integrated frontend, orchestrators, shared libraries, and pipeline storage.
8. **Pipeline Storage**:
   * Pipeline storage includes ML metadata for recording execution metadata and artifact locations, and pipeline artifacts storage for organizing artifacts on local or remote Cloud file systems.



1. **TFX Standard Components**:
   * TFX standard components are **pre-packaged** with TensorFlow and aim to **improve** pipeline development **velocity**.
   * They can be mixed with **custom** components to suit specific machine learning workflow needs.
   * Each standard component is designed around **common** machine learning tasks and encapsulates Google's ML best practices.
2. **ExampleGen Component**:
   * **ExampleGen** is the **entry point** to a TFX pipeline, responsible for **ingesting** data.
   * It supports **ingestion** of external data sources like CSV, TF Records, Avro, and Parquet.
   * It produces TF examples or TF sequence examples as outputs, along with **directory management** and **logging** for consistency.
3. **Benefits of ExampleGen**:
   * **ExampleGen** offers configurable and reproducible data **partitioning** and shuffling into TF Records.
   * It supports **ingestion** from various sources, including BigQuery, and leverages Apache Beam for **scalable** data **ingestion**.
   * ExampleGen is customizable for new input data formats and **ingestion** methods, saving time during pipeline development.
4. **Data Abstractions in TFX**:
   * TFX organizes data using abstractions such as **spans**, **versions**, and **splits**.
   * **Spans** group training examples, and each **span** can hold multiple **versions** of data.
   * **Splits** within **versions** allow for **partitioning** data, commonly into Train, Dev, and Test partitions.
5. **StatisticsGen Component**:
   * **StatisticsGen** computes summary statistics for features over Train, Dev, and Test **splits** using Apache Beam and TensorFlow Data Validation.
   * It produces a data set statistics artifact for downstream components, aiding in model evaluation and monitoring.
6. **SchemaGen Component**:
   * **SchemaGen** generates a schema description of input data characteristics, inferred from feature data distributions.
   * It produces a schema artifact that specifies data types, allowed ranges, and other properties, aiding in data validation and monitoring.
7. **ExampleValidator Component**:
   * **ExampleValidator** identifies anomalies in example data by comparing data statistics against a schema.
   * It can detect **validity** checks, feature-serving skew, and data drift, enhancing data quality monitoring during model training.
8. **TF Transform Component**:
   * **TF Transform** performs feature engineering on TF examples data using TensorFlow transformations defined in preprocessing functions.
   * It encapsulates feature engineering logic and produces a saved model artifact and transform data artifact for downstream model training.
9. **Benefits of TF Transform**:
   * **TF Transform** enables consistent feature engineering at training and serving time, reducing train-serving skew.
   * It supports common feature transformations like embedding, vocabulary generation, normalization, and bucketization, aiding in model performance and scalability.

n this module, we'll delve into TFX pipeline orchestration and workflows on Google Cloud. Let's explore the key points:

Motivation for Orchestration:

Orchestration brings standardization and software engineering best practices to ML workflows, allowing more focus on problem-solving.

Standardized pipelines facilitate code sharing, reuse, and portability across environments, fostering collaboration within ML teams.

Benefits of Standardization:

Standardized pipelines enable experienced team members to focus on problem-solving while managing and monitoring pipelines effectively.

New team members find it easier to ramp up on projects and make contributions when familiar with TFX pipelines and components.

TFX Pipeline Development Workflow:

Experimentation often starts in Jupyter Notebooks, where pipelines are built iteratively using an interactive context object.

Notebooks allow for interactive execution and artifact visualization, with automatic organization of pipeline artifacts in the local file system.

Pipelines developed in notebooks can be exported to production-ready orchestrators like Apache Beam with minimal code changes.

Choice of Orchestrators:

TFX pipelines are portable across orchestrators, supporting on-premise or cloud deployments.

Supported orchestrators include Apache Airflow, Kubeflow Pipelines, and Apache Beam, with each offering unique features and use cases.

Orchestrator Options:

Apache Beam Direct Runner: Suitable for local debugging and prototyping, simplifying system configuration and debugging.

Kubeflow Pipelines on Google Cloud: Offers a hosted and managed environment for scalable ML workflows, integrated with other Google Cloud services.

Apache Airflow: A fully managed version available on Google Cloud Composer, providing a flexible and mature orchestration solution.

TFX Command Line Interface (CLI):

The TFX CLI enables a range of pipeline actions across orchestrators, including creation, update, deletion, and monitoring of runs.

It facilitates manual pipeline orchestration tests and accelerates pipeline development by copying over template pipelines for modification.

Apache Beam for Data Processing:

Apache Beam is used for orchestrating and executing TFX pipelines, supporting scalable data processing in distributed environments.

It allows for packaging pipelines defined in notebooks into standalone files for local debugging and later scaling on distributed compute environments.

Understanding the choice of orchestrators for TFX pipelines is crucial for deploying scalable and robust ML workflows on Google Cloud. Each orchestrator offers distinct advantages, catering to different project requirements and environments.

User

Flexibility and Scalability:

Apache Beam provides a unified framework for running batch and streaming data processing jobs on various execution engines, including a direct runner for single-node execution and other distributed runners.

TFX leverages Beam for running tasks, ensuring a high degree of scalability across compute clusters, which is crucial for processing large datasets efficiently.

Abstraction Layer:

Beam offers an abstraction layer that enables TFX to run its supported data runners without requiring code modifications.

This abstraction simplifies pipeline development and execution, allowing developers to focus on improving model performance and delivering business impact.

Integration with TFX Libraries:

Many TFX libraries are built on the Beam programming model, enabling scalable data processing within TFX pipelines:

TensorFlow Data Validation computes dataset statistics using the Beam model for batch computation.

TensorFlow Transform performs batch feature engineering and transformations using the Beam programming model.

TensorFlow Model Analysis evaluates model metrics across data splits and slices using the Beam model for both batch and streaming processing.

BulkInference is powered by Beam for batch inference tasks within TFX pipelines.

Portability and Compatibility:

Beam's software development kit (SDK) can work with various runtimes, including full operating systems, Python, or Docker containers, translating them into distributed compute clusters via runners.

TFX components wrap up executable code in Beam pipelines, making them compatible with different runners such as Spark, Flink, and Google Cloud Dataflow.

Key Primitives:

Beam provides essential primitives for building data processing pipelines:

PCollection: Represents distributed multi-element datasets, serving as pipeline data.

PTransform: Processes elements in a collection to create new output collections without modifying the input.

ParDo: Core parallel processing transform, invoking user-specified functions on input elements independently and possibly in parallel.

Unified Framework for Batch and Streaming Data:

Beam balances correctness, latency, and costs for processing large, unbounded, out-of-order, and globally distributed datasets.

It offers a variety of IO transforms for reading and writing data to different storage types, ensuring portability and interoperability.

By learning Apache Beam, you gain a portable API for building sophisticated data-parallel processing pipelines within TFX. This unified framework streamlines pipeline development and enables seamless execution across various execution engines or runners, empowering you to focus on enhancing model performance and

User

Google Cloud is an excellent option for developing and running TFX pipelines. It has scalability in tight integration with a number of managed services for automation to improve your ml development velocity.

Play video starting at ::15 and follow transcript0:15

Let's examine how you can run TFX on Google Cloud to scale up your data processing and pipeline automation.

Play video starting at ::23 and follow transcript0:23

This diagram shows a high level overview of TFX on Google Cloud.

Play video starting at ::28 and follow transcript0:28

TFX sits on top of several layers of abstraction on Google Cloud to simplify running your tensor flow modeling code and production.

Play video starting at ::37 and follow transcript0:37

On Google Cloud, TFX runs on top of an AI platform Kubeflow pipelines instance. Which in turn runs on top of a Kubernete's cluster.

Play video starting at ::46 and follow transcript0:46

This instance simplifies the setup of a Kubernetes compute cluster, which coordinates the execution of your actual pipeline jobs. As well as hosts ml metadata in its persistent storage.

Play video starting at ::59 and follow transcript0:59

Pipeline artifacts storage is handled by cloud storage. As shown in your notebook lab, TFX simplifies standardized pipelined directory management for artifacts and metadata on local file system. TFX's capabilities can also be further extended to do so on Cloud storage, or remote globally distributed and scalable file system.

Play video starting at :1:21 and follow transcript1:21

You can also directly use BigQuery Google Clouds data warehousing solution as a data source.

Play video starting at :1:29 and follow transcript1:29

Further underpinning your TFX pipelines data processing is Apache beam jobs executed using the manage data flow product. Let's go one layer deeper to see how TFX runs on Google Cloud. First, the TFX domain specific language for pipeline authoring takes a user defined pipeline in a pipeline dot Pi file. That includes a mix of standard and custom components.

Play video starting at :1:54 and follow transcript1:54

Second, the TFX pipeline software development kit translates your pipeline description. Into a Kubeflow pipeline EMO file that can be executed with a Kubeflow dag runner.

Play video starting at :2:6 and follow transcript2:06

Third, your TFX pipeline is now a Kubeflow pipeline, that is run on a google Kubernetes engine cluster. The Argo workflow engine coordinates, your pipelines components. And your pipeline ml metadata is stored on a persistent disk on your Google Kubernetes Engine instance.

Play video starting at :2:25 and follow transcript2:25

Depending on your pipeline runtime setting, your pipeline also enjoy secure access to underlying Google Cloud services.

Play video starting at :2:33 and follow transcript2:33

The key takeaway for running pipelines on top of Google Cloud, is that the TFX DSL is converting your TFX pipeline code. Into a Kubeflow pipeline through the Kubeflow pipelines DSL. It is simplifying the operation of running on Google Cloud. Again, enabling you to keep focused on your machine learning problem and delivering business impact.

Play video starting at :2:57 and follow transcript2:57

TFX supports tight integrations with many Google Cloud services, and runs on top of cloud AI platform pipelines.

Play video starting at :3:6 and follow transcript3:06

For standard data and model components that use Apache Beam to implement data parallel pipelines. Google Cloud Dataflow manages the distribution of component executed data processing workflows. Under the TFX extensions API, there's also support for using many Google Cloud Managed Services. To scale your machine learning workflow with your data. These include, a custom ExampleGen for data stored in BigQuery, Google Cloud's data warehouse solution.

Play video starting at :3:38 and follow transcript3:38

The BigQuery example gen component can take SQL queries as input to generate TF record data splits to feed into your pipeline.

Play video starting at :3:47 and follow transcript3:47

The standard example Gen also supports data ingestion for data stored on the Google storage file system. Distributed using glob file patterns in a number of different file formats.

Play video starting at :4: and follow transcript4:00

A custom trainer component for running TensorFlow jobs on cloud AI platform training. A custom tuner component for distributing tuner jobs on AI platform training Using the familiar cares tuner API's.

Play video starting at :4:15 and follow transcript4:15

A custom pusher component to push TFX models to Cloud AI platform prediction for a managed, scalable production model server.

Play video starting at :4:24 and follow transcript4:24

Cloud AI platform runs on top of a Google Kubernetes engine cluster. So, your TFX pipeline has integrations to store pipeline artifacts on Cloud Storage. Operational logs on cloud logging, and ML metadata on cloud SQL.

Play video starting at :4:40 and follow transcript4:40

This simplification of pipeline artifact and metadata tracking allows you to accelerate your TFX development, and experimentation velocity. Furthermore, Google Cloud development tools like Cloud Functions, Container Registry, In cloud build streamline TFX pipeline coaching. Testing, deployment, and automatic triggering of continuous pipeline training. We will discuss using these tools for TFX pipeline continuous integration and deployment workflows on Google Cloud in the next module.

ChatGPT

Google Cloud provides an excellent environment for developing and running TFX pipelines, offering scalability and tight integration with managed services for automation, thereby enhancing your ML development velocity. Let's explore how you can leverage Google Cloud to scale up your data processing and automate your pipeline:

Abstraction Layers:

TFX sits on top of several layers of abstraction on Google Cloud, simplifying the execution of TensorFlow modeling code in production.

It runs on an AI Platform Kubeflow Pipelines instance, which in turn operates on a Kubernetes cluster, managing the execution of pipeline jobs and hosting ML metadata in persistent storage.

Artifacts Storage:

Cloud Storage handles pipeline artifacts storage, providing a scalable solution for managing artifacts and metadata.

TFX simplifies standardized pipeline directory management for artifacts and metadata, facilitating seamless integration with Cloud Storage or other remote scalable file systems.

Data Sources:

BigQuery, Google Cloud's data warehousing solution, can serve as a data source for TFX pipelines, enabling direct integration with large datasets stored in BigQuery tables.

Data Processing:

Apache Beam jobs are executed using Cloud Dataflow, which manages the distribution of component-executed data processing workflows.

TFX supports tight integrations with Google Cloud services, offering extensions API for leveraging various managed services to scale ML workflows with data.

Components and Services:

TFX provides standard and custom components tightly integrated with Google Cloud services:

Custom ExampleGen for data stored in BigQuery, allowing SQL queries as input to generate TFRecord data splits.

Custom Trainer component for running TensorFlow jobs on Cloud AI Platform Training.

Custom Tuner component for distributing tuner jobs on AI Platform Training using the Keras Tuner APIs.

Custom Pusher component to deploy TFX models to Cloud AI Platform Prediction for scalable production model serving.

Integration with Cloud Services:

Cloud AI Platform, running on a Google Kubernetes Engine cluster, integrates with TFX pipelines to store pipeline artifacts on Cloud Storage, operational logs on Cloud Logging, and ML metadata on Cloud SQL.

Simplified artifact and metadata tracking accelerates TFX development and experimentation velocity.

Development Tools:

Google Cloud development tools like Cloud Functions, Container Registry, and Cloud Build streamline TFX pipeline coaching, testing, deployment, and automatic triggering of continuous pipeline training.

By leveraging Google Cloud for running TFX pipelines, you can take advantage of its managed services, scalability, and tight integrations to accelerate your ML development and deployment processes. The abstraction layers provided by TFX and Google Cloud simplify pipeline execution, allowing you to focus on solving your machine learning problems and delivering business impact effectively.

Top of Form

Top of Form