Arrikto Proposal for ensuring Kubeflow as a Platform

Overview

We propose the creation of a Control Plane WG, which will maintain the common control plane and services that will allow for the release of Kubeflow as a coherent platform with simple, e2e workflows. The goal is to include the whole set of Kubeflow's functionality in an easy to install way on Kubernetes, either on the cloud or on-prem, without depending on external cloud services or closed source solutions. This WG will recommend at least one combination of versions for Kubernetes and the common services, which will be tested to work per Kubeflow release.

The purpose is to allow for a meaningful next release of Kubeflow, with its core components working seamlessly together, and to provide a way for vendors to have a tested reference, to build their platforms/distros on top.

Why?

See this message on Kubeflow's discuss mailing list for the rationale behind this proposal and why we think it makes sense to be able to deploy Kubeflow as a whole.

Currently, Kubeflow cannot be easily released as a coherent platform, due to the following two reasons,:

1. A number of components that tie Kubeflow together as a platform are left outside of the existing WGs, and thus are becoming unmaintained
2. Kubeflow's official manifests repository is in a very bad state:
    a. Current manifests are not easily maintainable
    b. Current manifests require a very specific, rigid structure imposed by `kfctl` to perform var substitutions.
    c. We currently have 3 different versions of manifests:
        i. A version in the individual application repos (e.g., what KFServing or KFP publishes)
        ii. A version in the manifests repo, which has been rewritten to work with `kfctl`
        iii. A second version in the manifests repo, which is architected to work with `kustomize`, but at the same time depends on the manifests that are meant to work with `kfctl`. And the two are not even in sync.

Scope

The Control Plane WG will take care of maintaining:

1. A common set of manifests for the current five (5) official Kubeflow applications (Training Operators, KFP, Notebooks, KFServing, Katib)
2. Manifests for a set of specific common services (Istio, Knative, Dex, Cert-Manager)
3. Common, currently unmaintained, apps, which tie the major components together (Central Dashboard, Profile Controller, PodDefaults Controller)
4. Instructions for a complete Kubeflow installation

The WG will provide at least one combination of versions for Kubernetes, Istio, Knative, Dex, and Cert-Manager that will be tested to work with a Kubeflow release, and rely on platform owners to test and provide/support more combinations if they like.

The WG's scope is to gradually drive towards a complete GitOps approach for future Kubeflow deployments.

Out of scope

This WG is not going to maintain platform/deployment-specific tools, or corresponding manifests. Thus, it will not maintain `kfctl`, `KFAM`, or any other similar code base. This will be left to the platform owners, who will have to choose what tool they want to use to deploy for their platform, and maintain it accordingly.

Implementation Details

Manifests

1. Create manifests for all current KF applications. We'll use `kustomize` to leverage manifests created by app owners, essentially creating a pipeline from app repos to a new, control plane manifests repo. Results in:
    - Less friction of recreating the manifests to work with `kfctl`'s var substitution pattern, and leverage the work already done in upstream app repos
    - Maintainable reference manifests that can be then used and customized by either individual users or/and platform owners.
    - Easier way for users and platform owners to integrate this repo with their GitOps tools (e.g., Argo, kpt, kapp, kfctl), or perform GitOps with `kubectl apply` directly.
    - Smoother updates with kustomize's method of separating user changes into overlays.
2. Create manifests for the following 3rd-party applications:
   Istio, Knative, Dex, Cert-Manager
   The above manifests will form a reference deployment of Kubeflow. This deployment will have the following hard requirements:
    - It will **not** use or depend upon any external cloud service offering or specific hardware.

    - It will not cater to any special user requirement (e.g., custom Istio, OIDC integration with LDAP).
    - It will be tested with at least one combination of versions for these 3rd-party applications along with a Kubernetes version for every release.

  Results in:

    - An independent, base Kubeflow deployment, which will provide traction with users that do not want to be locked-in with any vendor, either on-prem or on the cloud.
    - An easy way for platform owners to be able to re-use much of this reference deployment and manifests, and repurpose it for their platforms (e.g. create overlays for `kfctl` and maintain them separately, or create a platform that includes part of the components or a subset of the common services).
3. Provide e2e instructions on kubeflow.org of how to perform an installation on an existing Kubernetes cluster, using these manifests for the recommended combination of versions.

Central Dashboard, PodDefaults Controller, Profile Controller

Maintain and further advance the Central Dashboard to serve its current purpose of being the UI that connects together all Kubeflow web apps. Maintain the PodDefaults Controller and the Profile Controller.