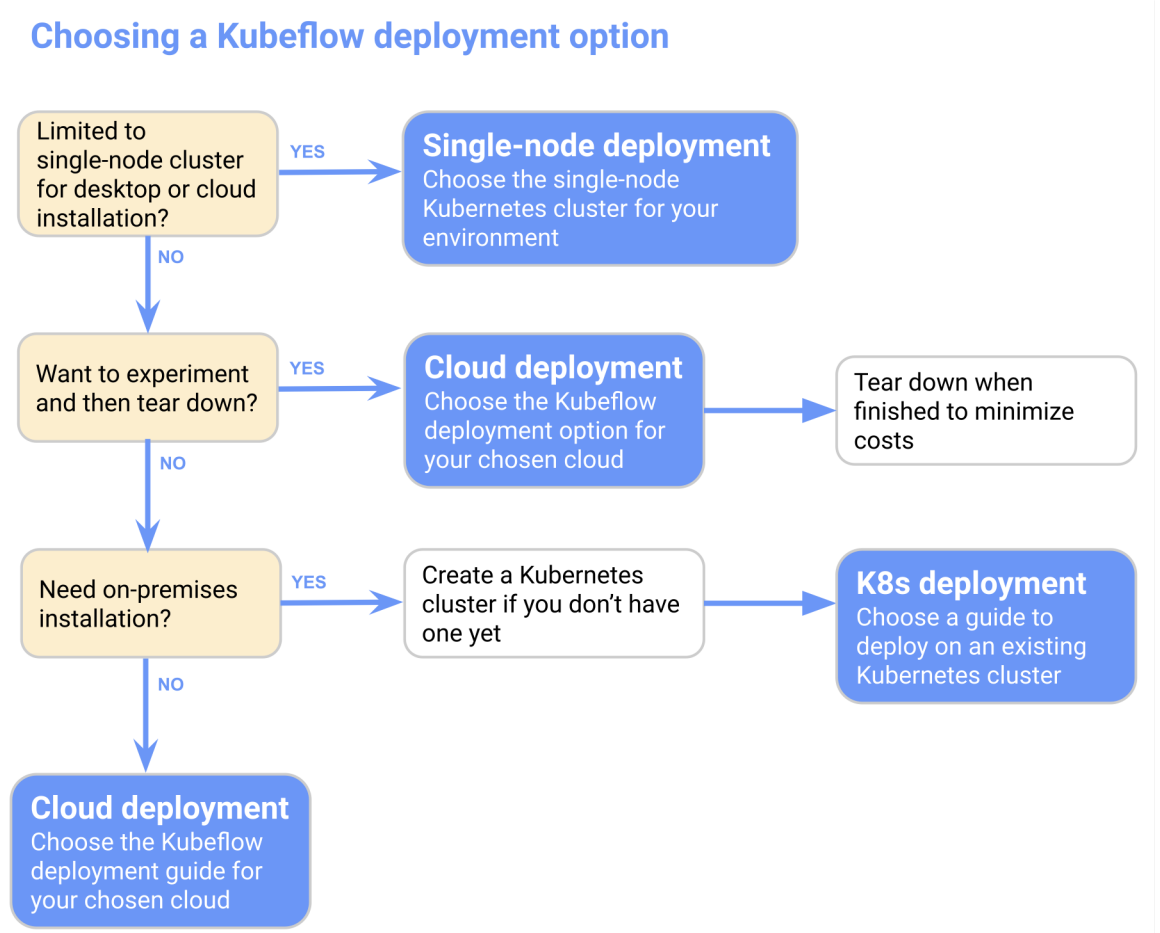
Kubeflow 控制器（Control Plane）

Weiqiang Zhuang
wzhuang@us.ibm.com
IBM CODAIT
Github id: adrian555

Kubeflow control plane

- 安装，管理和监视（Deploy, manage and monitor）Kubeflow
  - 文档（Document）
    - https://www.kubeflow.org/docs/started/getting-started/
  - 多样环境运行（On various environments）
    - GCP/AWS/IKS/OpenShift
    - Other K8S
    - On-prem Linux/MacOS/Windows
    - minikube/miniKF
  - 命令行或Operator安装（Deployed through command line or operator）
  - 部件和应用可配置（Configuration for the collection of components/applications）
    - Use one from manifests repo, or
    - Create your own
  - 源码仓库（Two repos）
    - kfctl https://github.com/kubeflow/kfctl
    - manifests https://github.com/kubeflow/manifests



https://www.kubeflow.org/docs/images/kubeflow-getting-started-diagram.svg

kfctl

- *kfctl* 控制器（the control plane for deploying and managing Kubeflow）
  - Run *kfctl* as a CLI with **KfDef** configurations for different Kubernetes flavors
  - *kubeflow/kfctl* also incubates an operator to deploy and monitor Kubeflow
- **KfDef** 配置文件（configurations）
  - are manifests specifying a set of applications to be deployed by their kustomization and resources by kustomize
  - resources of each application are organized in the layout for kustomize to process
  - *kubeflow/manifests* is the repo for the collection of KfDef configurations
- kustomize 资源配置生成器
  - customizes raw, template-free YAML files. It patches Kubernetes resources files with a kustomization file and various overlays.
  - kustomization is also a Kubernetes resource (kind: Kustomization). It contains the `generators` and `transformers` to be applied on the `resources`.

KfDef

- KfDef
  - yaml格式配置文件（Configuration through yaml）
  - 源码（Code）https://github.com/kubeflow/kfctl/blob/master/pkg/apis/apps/kfdef/v1/application_types.go
  - *应用（applications）* are in kustomize form
    - starting from v1.1 supports kustomize v3 in *stacks* form (a *kubeflow-apps* application is required)
  - Also support plugins for certain platforms (ie. Aws, Gcp)
  - 支持远程或本地资源配置文件仓库（Manifest repo can be either remote archive or local directory）
    - The directory structure for manifests follows kustomize requirement
    - Eg. Argo

```yaml
apiVersion: kfdef.apps.kubeflow.org/v1
kind: KfDef
metadata:
  name: kfdef-example
  namespace: kubeflow
spec:
  applications:
  - kustomizeConfig:
    name: application-1
    parameters:
    - name: param1
      value: value1
    repoRef:
      name: manifests
      path: application-1
  - kustomizeConfig:
    name: application-2
    overlays:
    - application
    repoRef:
      name: manifests
      path: application-2
  repos:
  - name: manifests
    url: https://example.com/manifests/v1.0.0.tar.gz
  version: v1.0.0
```

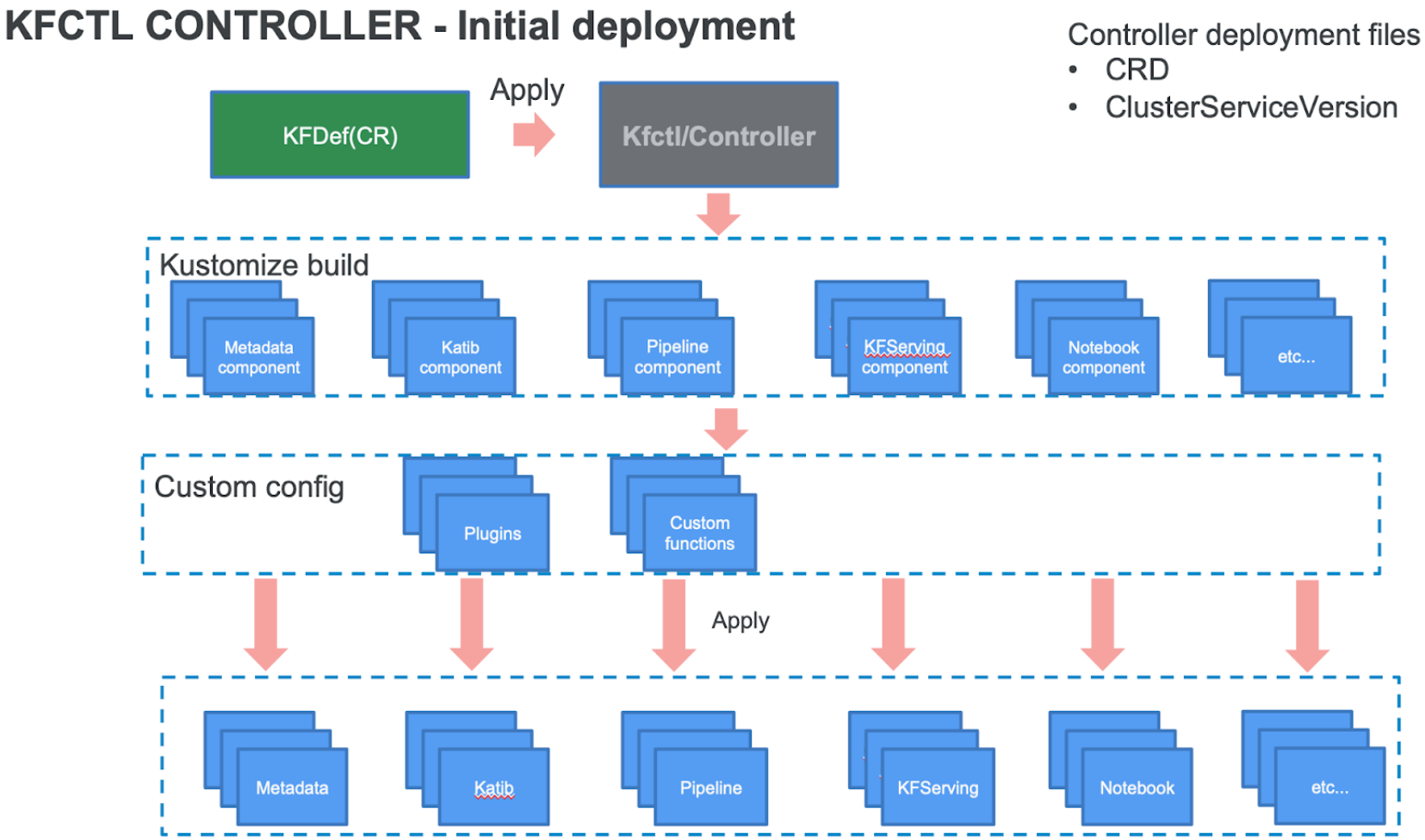
Configuration in yaml

```
.
├── application-1
│   └── base
│       ├── deployment.yaml
│       ├── kustomization.yaml
│       ├── params.env
│       ├── params.yaml
│       └── service.yaml
└── application-2
    ├── base
    │   ├── deployment.yaml
    │   ├── kustomization.yaml
    │   └── service.yaml
    └── overlays
        └── application
            ├── application.yaml
            └── kustomization.yaml
```

Directory structure

kfctl deployment flow

- kfctl



Controller deployment files
- CRD
- ClusterServiceVersion

Kubeflow control plane

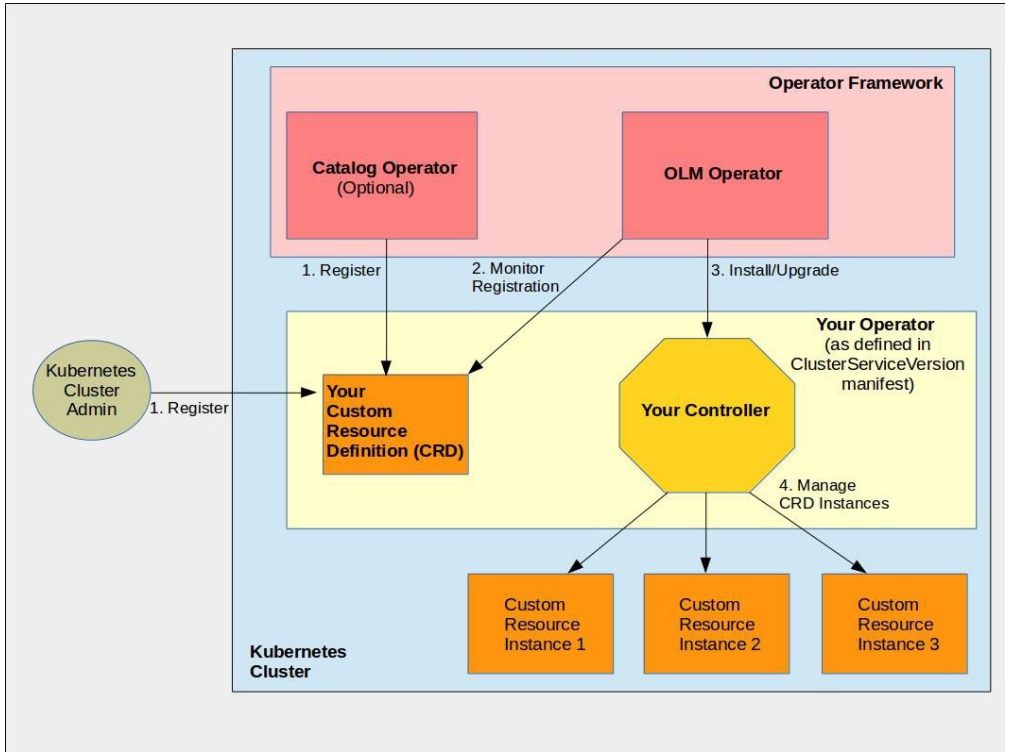
- kfctl
  - 显示所有命令（List all commands）
    - `$> kfctl help`
  - 安装和删除命令（Command line to install/uninstall Kubeflow）
    - `$> kfctl build –V –f <config_uri>`
    - `$> kfctl apply –V –f <config_uri>`
    - `$> kfctl delete –V –f <config_uri>`
  - <config_uri> can be remote or local
  - 基本工作流程（High-level flow）
    - Downloads the manifests for applications (if remote) from the *repo:uri* defined in the configuration file, and caches in the local disk
    - Loops through all applications' kustomization configuration and build/apply
    - Runs platform special handling if the configuration contains *plugins* section

Kubeflow control plane

- Kubeflow 应用资源配置文件仓库（manifests repo）
  - Maintains the manifests for Kubeflow's common applications
    - Argo, centraldashboard, admission-webhook, basic-auth, metadata, profiles and more
  - Other applications
  - Each application can be built with `kustomize` tool
    - `$> kustomize build`
    - `$> kubectl apply -k`

Kubeflow control plane

- Kubeflow Operator
  - 定制资源定义+API（CRD+API）
  - Configuration file is the custom resource (CR)
  - 管理Kubeflow及各应用生命周期（Operator helps deploy, monitor and manage the lifecycle of applications deployed on Kubernetes and OpenShift clusters）
    - Built with operator-sdk
    - Learn more about operators - link
  - 共享apply程序源码（Shares the same *apply* function with *kfctl* command）
  - *delete* function diffs from *kfctl* command
  - 文档（Document）
    - https://github.com/kubeflow/kfctl/blob/master/operator.md



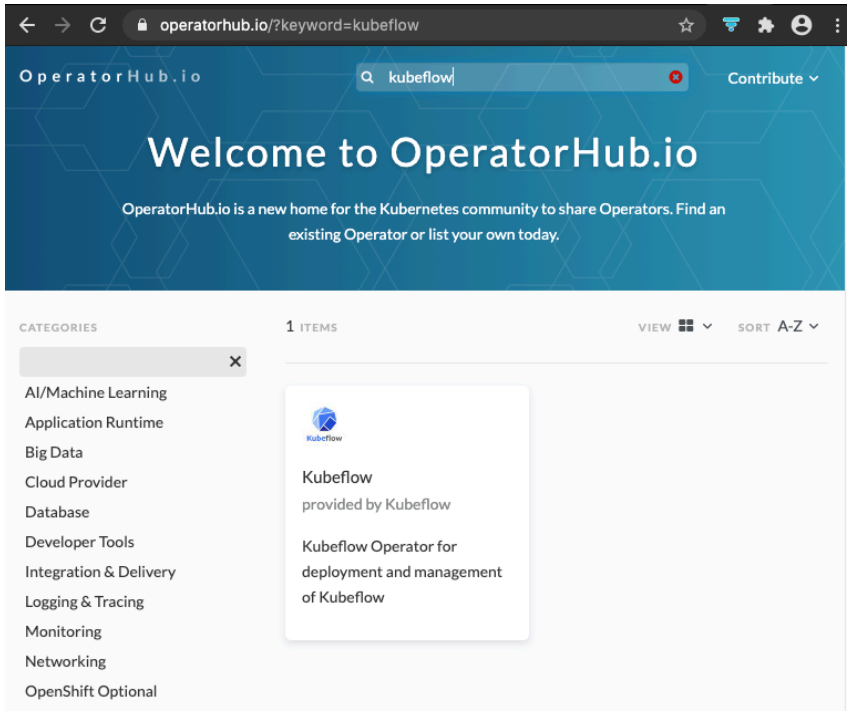
https://miro.medium.com/max/2116/1*GYLAUB7KGCysjPgwek-pPA.jpeg

Kubeflow control plane

- Kubeflow Operator
  - 源码结构（Code structure）
    - /deploy: Contains all the k8s resources for deploying the operator image and crd
    - /build: Operator image build script
    - /pkg/controller: main package for operator controller logic
    - /cmd/manager: main.go file for the operator go program
  - 监视相关资源（Kubeflow operator watches the KfDef and other related resources）
  - 两步安装Kubeflow（Two steps to install Kubeflow）
    - Deploy the Kubeflow operator, then
    - Install the Kubeflow by creating the KfDef CR
  - 监视和管理功能（Kubeflow operator continues to monitor and manage any KfDef CR created）

Kubeflow control plane

- Kubeflow operator
  - 安装 Kubeflow Operator
    - Operator can be deployed by command line
      - `$> export OPERATOR_NAMESPACE=operators`
      - `$> kubectl create ns ${OPERATOR_NAMESPACE}`
      - `$> cd deploy/`
      - `$> kustomize edit set namespace ${OPERATOR_NAMESPACE}`
      - `$> kustomize build | kubectl apply -f -`
    - Operator is registered on operatorhub.io, can be installed through OLM console
      - OLM discovers the Kubeflow operator from its catalog source
  - 安装Kubeflow（installed either by command lines or by subscription）
    - creating a KfDef CR from command line
      - `download the KfDef configuration file from kubeflow/manifests`
      - `add metadata.name`
      - `$> kubectl apply –f <kfdef_configuration.yaml>`
    - creating a *subscription* to the operator from the OLM console



Thank you!

谢 谢

Backup

- More topics
  - KfUpgrade
  - other kfctl sub-commands
  - `kpt fn` commands
  - kustomize v3 support in the coming release 1.1
  - code walkthrough

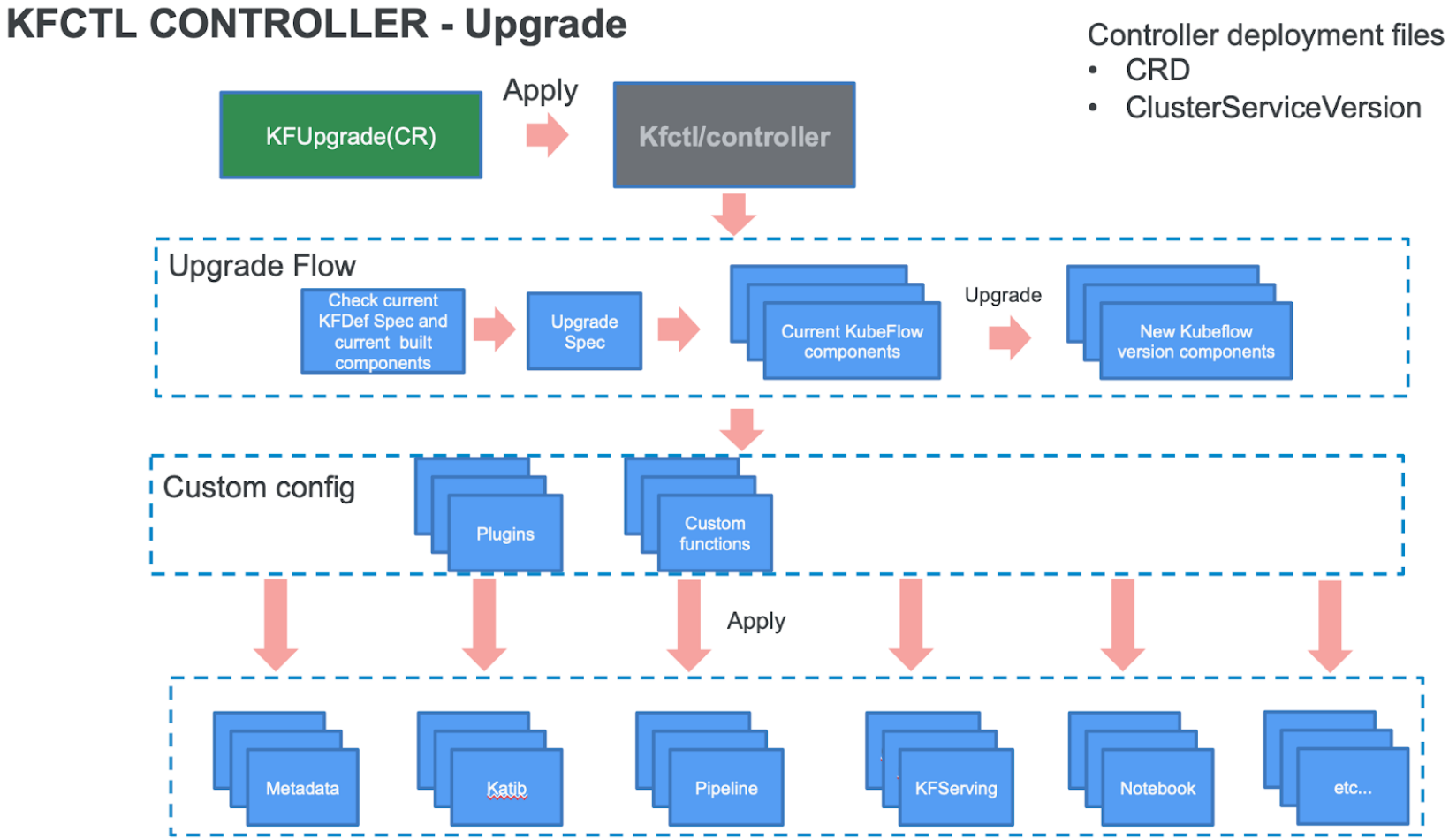
Backup

- KfUpgrade
  - in alpha

```yaml
apiVersion: kfupgrade.apps.kubeflow.org/v1alpha1
kind: KfUpgrade
metadata:
  name: kf-upgrade-v0.7.1
spec:
  currentKfDef:
    # Replace with the name of your Kubeflow app
    name: kubeflow-app
    version: v0.7.0
  newKfDef:
    # Replace with the name of your kubeflow app
    name: kubeflow-app
    version: v0.7.1
  # Replace this with the path to the KfDef that you are upgrading to
  baseConfigPath: https://example.com/manifests/v0.7.1.yaml
```

Backup

- kfctl



Code walkthrough

- Directories

```
kfctl
├── build
│   └── bin
├── cmd
│   ├── kfctl
│   │   └── cmd
│   ├── manager
│   └── plugins
│       └── dockerfordesktop
├── config
├── deploy
│   ├── crds
│   └── olm-catalog
│       └── kubeflow
├── hack
├── pkg
│   ├── apis
│   │   └── apps
│   ├── controller
│   │   └── kfdef
│   ├── kfapp
│   │   ├── aws
│   │   ├── coordinator
│   │   ├── dockerfordesktop
│   │   ├── existing_arrikto
│   │   ├── gcp
│   │   ├── kustomize
│   │   └── minikube
│   ├── kfconfig
│   │   ├── awsplugin
│   │   ├── gcpplugin
│   │   ├── loaders
│   │   └── testdata
│   ├── kfupgrade
│   ├── mirror
│   │   └── testdata
│   └── utils
├── py
│   └── kubeflow
│       └── kfctl
├── testing
│   └── workflows
│       ├── components
│       ├── environments
│       └── lib
└── third_party
```

```
manifests
├── admission-webhook
├── application
├── argo
├── aws
├── cert-manager
├── common
├── default-install
├── dex-auth
├── docs
├── experimental
├── gatekeeper
├── gcp
├── hack
├── istio
├── istio-1-3-1
├── jupyter
├── katib
├── kfdef
├── kfserving
├── knative
├── kubebench
├── kubeflow-roles
├── metacontroller
├── metadata
├── modeldb
├── mpi-job
├── mxnet-job
├── pipeline
├── plugins
├── profiles
├── pytorch-job
├── seldon
├── spark
├── tektoncd
├── tensorboard
├── tests
└── tf-training
```

```
kfdef
├── OWNERS
├── README.md
├── generic
│   ├── OWNERS
│   ├── README.md
│   ├── auth_oidc
│   │   ├── authservice.tmpl
│   │   ├── dex.tmpl
│   │   ├── envoy-filter.yaml
│   │   └── gateway.yaml
│   └── istio
│       ├── crds.yaml
│       └── istio-noauth.yaml
├── kfctl_anthos.v1.0.0.yaml
├── kfctl_anthos.v1.0.1.yaml
├── kfctl_anthos.yaml
├── kfctl_aws.v1.0.0.yaml
├── kfctl_aws.v1.0.1.yaml
├── kfctl_aws.yaml
├── kfctl_aws_cognito.v1.0.0.yaml
├── kfctl_aws_cognito.v1.0.1.yaml
├── kfctl_aws_cognito.yaml
├── kfctl_gcp_asm_exp.yaml
├── kfctl_gcp_basic_auth.v1.0.0.yaml
├── kfctl_gcp_basic_auth.v1.0.1.yaml
├── kfctl_gcp_basic_auth.yaml
├── kfctl_gcp_iap.v1.0.0.yaml
├── kfctl_gcp_iap.v1.0.1.yaml
├── kfctl_gcp_iap.yaml
├── kfctl_ibm.v1.0.0.yaml
├── kfctl_ibm.v1.0.1.yaml
├── kfctl_ibm.yaml
├── kfctl_istio_dex.v1.0.0.yaml
├── kfctl_istio_dex.v1.0.1.yaml
├── kfctl_istio_dex.yaml
├── kfctl_k8s_istio.v1.0.0.yaml
├── kfctl_k8s_istio.v1.0.1.yaml
├── kfctl_k8s_istio.yaml
└── kfctl_upgrade_gcp_iap_1.0.0.yaml
```

```
kfserving
├── kfserving-crds
│   ├── base
│   │   ├── crd.yaml
│   │   └── kustomization.yaml
│   └── overlays
│       └── application
│           ├── application.yaml
│           └── kustomization.yaml
└── kfserving-install
    ├── base
    │   ├── cluster-role-binding.yaml
    │   ├── cluster-role.yaml
    │   ├── config-map.yaml
    │   ├── kustomization.yaml
    │   ├── params.env
    │   ├── params.yaml
    │   ├── secret.yaml
    │   ├── service.yaml
    │   └── statefulset.yaml
    └── overlays
        └── application
            ├── application.yaml
            └── kustomization.yaml
```