## 在Azure Kubernetes Service上构建Kubeflow Machine Learning 平台

Kubernetes可以通过容器编排pipeline来应对许多计算挑战。从它诞生后，MachineLearning领域的数据科学家及工程师期待并尝试借助Kubernetes来搭建机器学习平台，从而为ML实践带来以下几个优势：

* 计算资源可扩展性
* GPU 支持
* 多租户
* 数据管理
* 基础设施抽象

直到Kubeflow的出现，在Kubernetes上部署机器学习平台带来了质的飞跃。

### Kubeflow背景

什么是Kubeflow？ 一开始是谷歌内部运行 TensorFlow 方式的开源，基于名为 TensorFlow Extended 的管道。 它最初只是一种在 Kubernetes 上运行 TensorFlow 作业的更简单方法，但后来扩展为一个多架构、多云框架，用于运行端到端机器学习工作流。

[Kubeflow](https://www.kubeflow.org/) 是 Kubernetes 的机器学习工具包。

要使用 Kubeflow，基本工作流程是：

下载并运行 Kubeflow 部署二进制文件。

自定义生成的配置文件。

运行指定的脚本以将您的容器部署到您的特定环境。

您可以调整配置以选择要用于 ML 工作流每个阶段的平台和服务：

* 数据准备
* 模型训练，
* 预测服务
* 服务管理

您可以选择在本地、本地或云环境中部署 Kubernetes 工作负载。

### Kubeflow 愿景

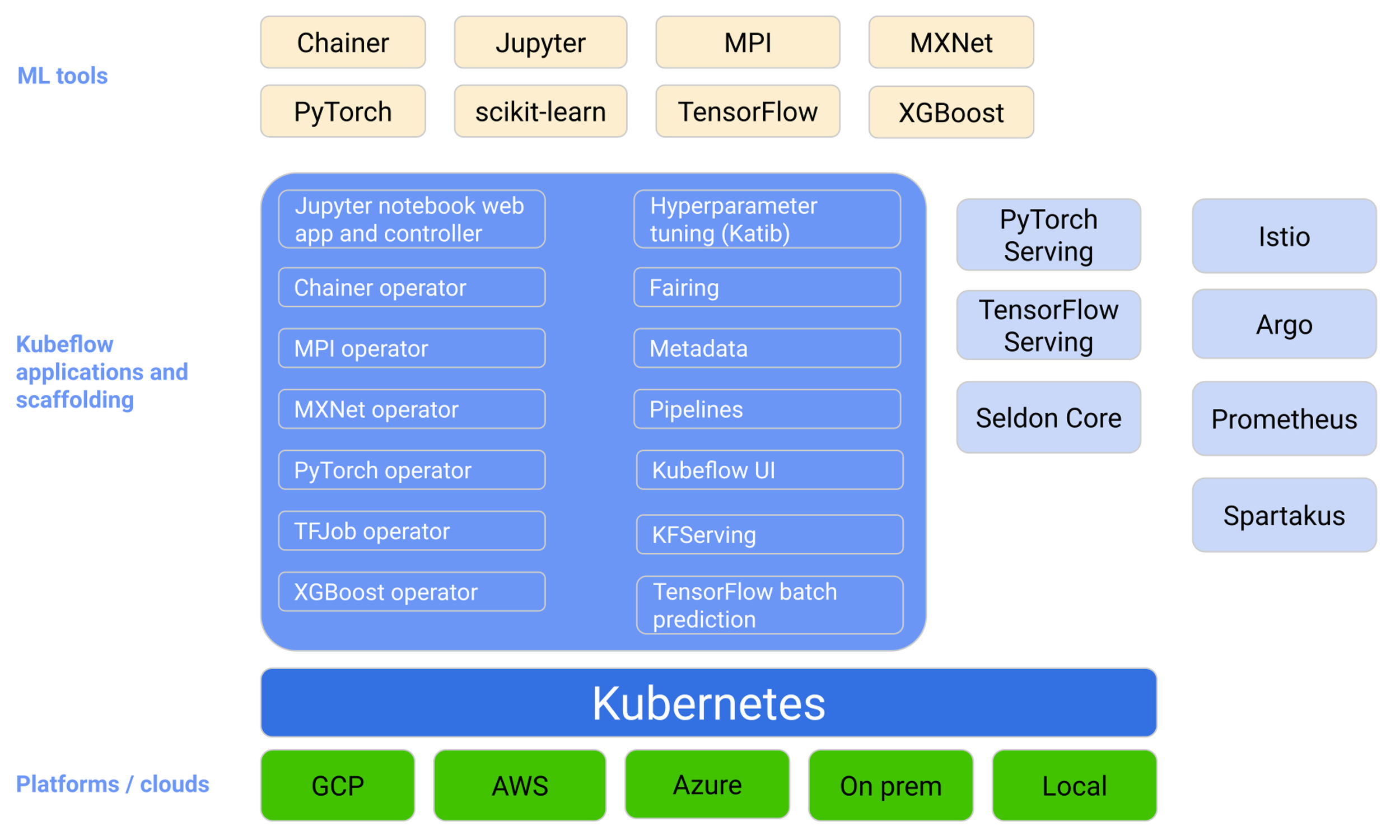
Kubeflow 项目致力于使机器学习 (ML) 工作流在 Kubernetes上的部署变得简单、便携和可扩展。我们的目标不是重新创建其他服务，而是提供一种直接的方法，将用于 ML的同类最佳开源系统部署到不同的基础设施。 无论您在何处运行 Kubernetes，都应该能够运行 Kubeflow。Kubeflow的目标是通过让 Kubernetes 做它擅长的事情，使机器学习 (ML) 模型的可扩展性和将它们部署到生产中尽可能简单：

* 在不同的基础设施上轻松、可重复、可移植的部署（例如，在笔记本电脑上进行试验，然后转移到本地集群或云）
* 部署和管理松散耦合的微服务
* 按需扩容

由于 ML从业者使用多种工具，其中一个关键目标是根据用户需求（在合理范围内）定制堆栈，让系统处理”无聊的东西”。虽然我们开始使用的是一组狭窄的技术，但我们正在与许多不同的项目合作以包含额外的工具。最终，我们希望拥有一组简单的清单，在 Kubernetes已经运行的任何地方为您提供易于使用的 ML堆栈，并且可以根据它部署到的集群进行自我配置。

### Kubeflow架构

下图显示了 Kubeflow 作为在 Kubernetes 之上安排 ML 系统组件的平台：

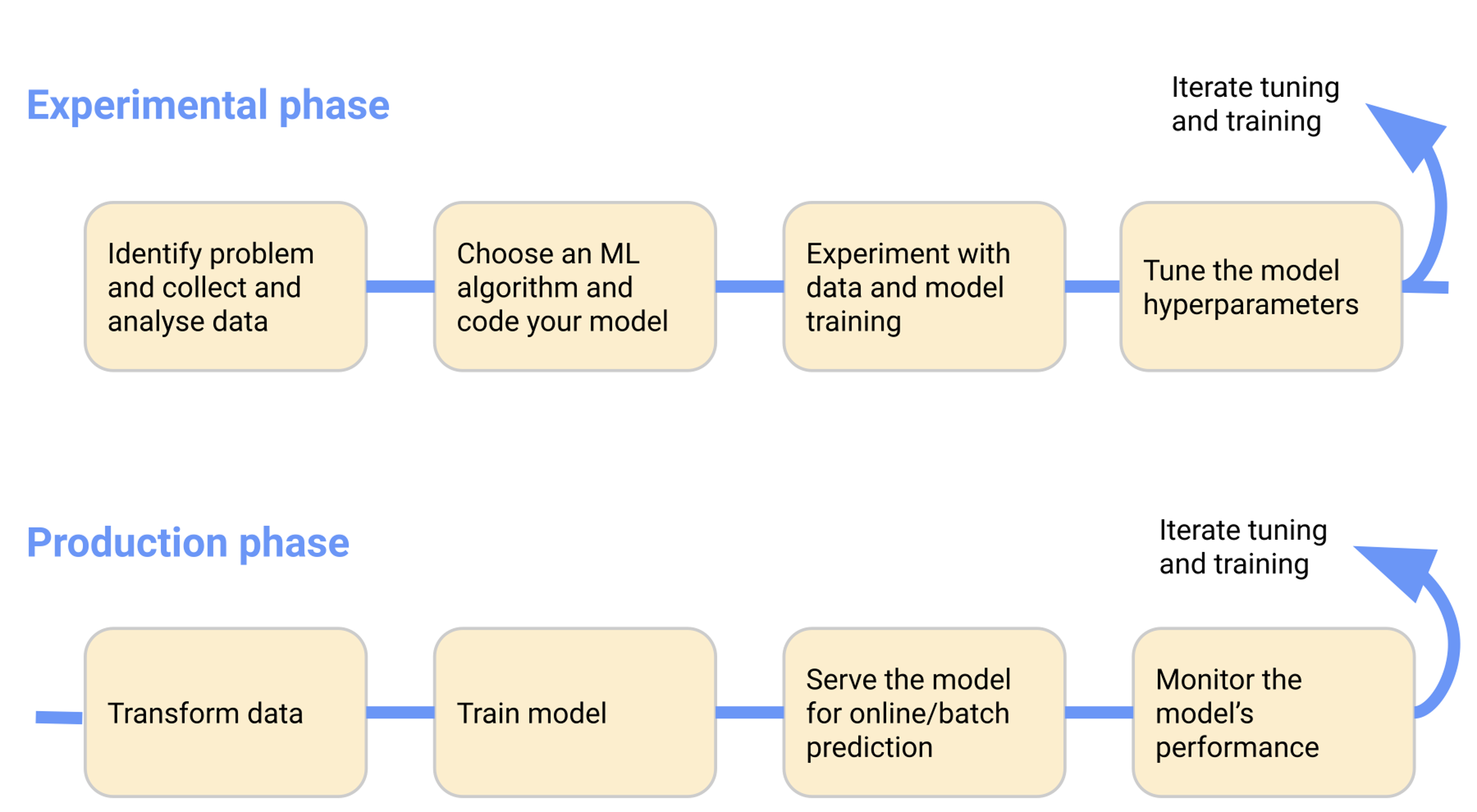


Kubeflow在[Kubernetes](https://kubernetes.io/)的基础上作为部署，缩放和管理复杂系统的ML系统。

使用 （[Kubeflow配置界面](https://www.kubeflow.org/docs/started/architecture/#interfaces)），您可以指定工作流所需的ML工具。然后，您可以将工作流部署到各种云、本地和本地平台，以进行实验和生产使用。

当您开发和部署 ML 系统时，ML 工作流通常由几个阶段组成。开发 ML系统是一个迭代过程。您需要评估 ML工作流各个阶段的输出，并在必要时对模型和参数应用更改，以确保模型不断产生您需要的结果。

为简单起见，下图按顺序显示了工作流阶段。工作流末尾的箭头指向流程以指示流程的迭代性质：



更详细地查看各个阶段：

* 在实验阶段，您根据初始假设开发模型，并迭代地测试和更新模型以产生您正在寻找的结果：
  + 确定您希望机器学习系统解决的问题。
  + 收集和分析训练 ML 模型所需的数据。
  + 选择 ML 框架和算法，并对模型的初始版本进行编码。
  + 试验数据并训练您的模型。
  + 调整模型超参数以确保最有效的处理和最准确的结果。
* 在生产阶段，您将部署一个执行以下过程的系统：
  + 将数据转换为训练系统所需的格式。为确保您的模型在训练和预测期间表现一致，转换过程在实验和生产阶段必须相同。
  + 训练机器学习模型。
  + 为模型提供在线预测或以批处理模式运行。
  + 监控模型的性能，并将结果提供给您的流程以调整或重新训练模型。

# 在AKS上构建Kubeflow Machine Learning 平台

Kubeflow 社区官方提供了如何使用 kfctl 二进制文件在 Azure 上部署 Kubeflow的指南。然而由于Azure Kubernetes Service对K8S版本的升级， 使用该指南安装后，并不能成功运行Kuberflow,我们可以先按如下标准步骤进行安装，在下一个章节（修改配置）进行Bug修补和配置修改，确保可以成功部署Kuberflow。

## 先决条件

* Kubeflow 部署在AKS上，允许数据科学家对 CPU 和 GPU 进行可扩展的访问，这些 CPU 和 GPU 在计算需要大量活动时自动增加，并在完成后缩减。**如在AKS上需要使用GPU节点，请先配置GPU node pool，按照Azure官方指南即可** 。如使用普通CPU计算节点，可以忽略此步骤。
* 安装[kubectl](https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl-on-linux)
* 安装和配置[Azure 命令行界面 (Az)](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest)
  + 登录 az login
* 安装 Docker
  + 对于 Windows 和 WSL：[指南](https://docs.docker.com/docker-for-windows/wsl/)
  + 对于其他操作系统：[Docker 桌面](https://docs.docker.com/docker-hub/)

你不需要拥有适用于 AKS（Azure Kubernetes 服务）的现有 Azure 资源组或群集。您可以在部署过程中创建集群。

## 应用布局

您的 Kubeflow 应用程序目录**${KF\_DIR}**包含以下文件和目录：

* **${CONFIG\_FILE}**是一个 YAML 文件，用于定义与您的 Kubeflow 部署相关的配置。
  + 此文件是您在部署 Kubeflow 时使用的基于 GitHub 的配置 YAML 文件的副本。例如，<https://raw.githubusercontent.com/kubeflow/manifests/v1.2-branch/kfdef/kfctl_azure.v1.2.0.yaml> 。
  + 当您运行kfctl apply或 时kfctl build，kfctl 会创建一个本地版本的配置文件 ，${CONFIG\_FILE}您可以在必要时进一步自定义。
* **kustomize**是一个包含 Kubeflow 应用程序的 kustomize 包的目录。
  + 该目录是在您运行kfctl build或时创建的kfctl apply。
  + 您可以自定义 Kubernetes 资源（修改清单并kfctl apply再次运行）。

如果您在运行这些脚本时遇到任何问题，请参阅[故障排除指南](https://www.kubeflow.org/docs/azure/troubleshooting-azure)以获取更多信息。

## Azure 设置

#### 要从命令行界面登录 Azure，请运行以下命令

az login  
  
az account set \--subscription \<NAME OR ID OF SUBSCRIPTION>

#### Azure Kubernetes Service集群的初始创建和设置

创建资源组：

az group create -n \<RESOURCE\_GROUP\_NAME> -l \<LOCATION>

示例变量：

* RESOURCE\_GROUP\_NAME=KubeTest
* LOCATION=westus

创建一个Azure Kubernetes Service的集群：

az aks create -g \<RESOURCE\_GROUP\_NAME> -n \<NAME> -s \<AGENT\_SIZE> -c  
<AGENT\_COUNT> -l \<LOCATION> \--generate-ssh-keys

示例变量：

* NAME=KubeTestCluster
* AGENT\_SIZE=Standard\_D4s\_v3
* AGENT\_COUNT=2
* RESOURCE\_GROUP\_NAME=KubeTest

**注意**：如果您使用基于 GPU 的 AKS 集群（例如：AGENT\_SIZE=Standard\_NC6），您还需要在集群节点上[安装 NVidia 驱动程序](https://docs.microsoft.com/azure/aks/gpu-cluster#install-nvidia-drivers)，然后才能将 GPU 与 Kubeflow 一起使用。

## Kubeflow 安装

**重要提示**：要使用多用户身份验证和命名空间分离在 Azure 上部署 Kubeflow，请使用[Azure 中使用 OICD](https://www.kubeflow.org/docs/azure/authentication-oidc)进行[身份验证](https://www.kubeflow.org/docs/azure/authentication-oidc)的说明。本指南中的说明仅适用于单用户 Kubeflow 部署。此类部署目前无法升级为多用户部署。

**注意**：kfctl 目前仅适用于 Linux 和 macOS 用户。如果您使用 Windows，则可以在 Windows Subsystem for Linux (WSL) 上安装 kfctl。WSL的设置请参考官方[说明](https://docs.microsoft.com/en-us/windows/wsl/install-win10)。

运行以下命令来设置和部署 Kubeflow。

1. 创建用户凭据。您只需要运行此命令一次。

2. az aks get-credentials -n \<NAME> -g \<RESOURCE\_GROUP\_NAME>

1. 从[Kubeflow 版本页面](https://github.com/kubeflow/kfctl/releases/tag/v1.2.0)下载 kfctl v1.2.0 版本 。
2. 解压kfctl：

tar -xvf kfctl\_v1.2.0\\_\<platform>.tar.gz

1. 运行以下命令来设置和部署Kubeflow。下面的代码包含一个可选命令，用于将二进制 kfctl 添加到您的路径中。如果不将二进制文件添加到路径中，则每次运行时都必须使用 kfctl 二进制文件的完整路径。
2. 在目录名称中仅使用字母数字字符或 -

export PATH=\$PATH:\"\<path-to-kfctl>\"

1. 将 KF\_NAME 设置为您的 Kubeflow 部署的名称。

export KF\_NAME=\<your choice of name for the Kubeflow deployment>

1. Kubeflow 目录设置。 例如，/opt/。 然后为此部署设置 Kubeflow 应用程序目录。

export BASE\_DIR=\<path to a base directory>  
export KF\_DIR=\${BASE\_DIR}/\${KF\_NAME}

1. 设置Kubeflow 部署所使用的Yaml模版

export CONFIG\_URI=\"https://raw.githubusercontent.com/kubeflow/manifests/v1.2-branch/kfdef/kfctl\_k8s\_istio.v1.2.0.yaml\"

1. 运行kfctl在AKS上部署kubeflow

mkdir -p \${KF\_DIR}  
  
cd \${KF\_DIR}  
  
kfctl apply -V -f \${CONFIG\_URI}

* **${KF\_NAME}** - 您的 Kubeflow 部署的名称。如果您需要自定义部署名称，请在此处指定该名称。例如，my-kubeflow或kf-test。KF\_NAME 的值必须由小写字母数字字符或”-“组成，并且必须以字母数字字符开头和结尾。此变量的值不能超过 25 个字符。它必须只包含一个名称，而不是目录路径。在创建存储 Kubeflow 配置的目录（即 Kubeflow 应用程序目录）时，您也可以使用此值作为目录名称。

- \*\*\${KF\_DIR}\*\* - Kubeflow 应用程序目录的完整路径。  
  
- \*\*\${CONFIG\_URI}\*\* - 要用于部署 Kubeflow 的配置 YAML 文件的  
 GitHub 地址。本指南中使用的 URI  
 是 <https://raw.githubusercontent.com/kubeflow/manifests/v1.2-branch/kfdef/kfctl\_k8s\_istio.v1.2.0.yaml>。当您运行kfctl  
 applyor kfctl build（参见下一步）时，kfctl 会创建配置 YAML  
 文件的本地版本，您可以在必要时进一步自定义。

1. 运行此命令以检查资源是否已在命名空间中正确部署kubeflow：

kubectl get all -n kubeflow

1. 打开 Kubeflow 仪表板

默认安装不会创建外部端点，但您可以使用端口转发来访问您的集群。运行以下命令：

kubectl port-forward svc/istio-ingressgateway -n istio-system 8080:80

接下来，http://localhost:8080在浏览器中打开。

要将仪表板打开到公共 IP 地址，您应该首先实施一个解决方案来防止未经授权的访问。您可以从[Azure 部署的访问控制中](https://www.kubeflow.org/docs/azure/authentication)阅读有关 Azure 身份验证选项的更多信息。

## 修改配置

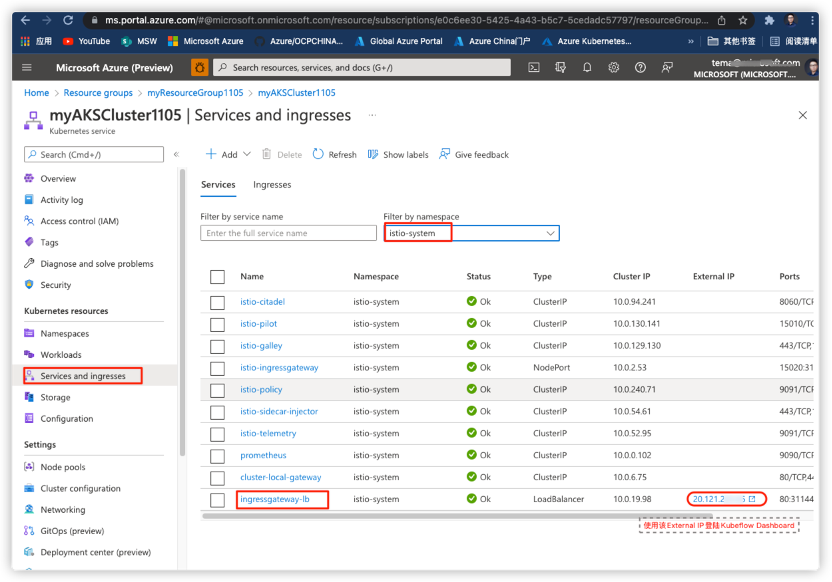
**由于AKS从1.19开始默使用containerd作为K8S集群认runtime（开始弃用 Docker Runtime），由于这种runtime变化及其他升级变化，致使kubeflow安装后产生了兼容性问题。**

1. 默认安装不会创建外部端点，想从外部长期访问Kubeflow Cental Dashboard需要给它配置一个[K8S **Service**](https://github.com/teo-ma/Build-a-kubeflow-machine-learning-platform-on-AKS/blob/main/yaml/serviceforistio-ingressgateway.yaml)**，**

// serviceforistio-ingressgateway.yaml  
  
apiVersion: v1  
  
kind: Service  
  
metadata:  
  
name: kubeflowdashboard-lb  
  
namespace: istio-system  
  
spec:  
  
type: LoadBalancer  
  
ports:  
  
- port: 80  
  
selector:  
  
app: istio-ingressgateway

运行**kubectl apply -f** serviceforistio-ingressgateway.yaml

在AKS console或kubectl可以查看生成的external IP，用于登陆Kubeflow Dashboard



1. **MPI Operator 可以轻松地在 Kubernetes 上运行 allreduce 风格的分布式训练。默认安装kubeflow后发现MPI operator 服务的Pod是失败的，解决方法是单独重新安装。**

git clone https://github.com/kubeflow/mpi-operator  
  
cd mpi-operator  
  
kubectl apply -f deploy/v2beta1/mpi-operator.yaml  
  
kubectl apply -k manifests/overlays/Kubeflow

1. **解决Kubeflow Dashboard无法显示Pipeline 功能tab**

kubectl edit destinationrule -n kubeflow ml-pipeline

**将 tls.mode（最后一行）从 ISTIO\_MUTUAL 修改为 DISABLE**

kubectl edit destinationrule -n kubeflow ml-pipeline-ui

**将 tls.mode（最后一行）从 ISTIO\_MUTUAL 修改为 DISABLE**

1. **解决无法运行Kubeflow Pipeline问题**

**由于AKS1.19之后版本将容器 Runtime从Docker换成了containerd，从而带来Kubeflow兼容问题，按如下方法修改workflow-controller-configmap的配置：**

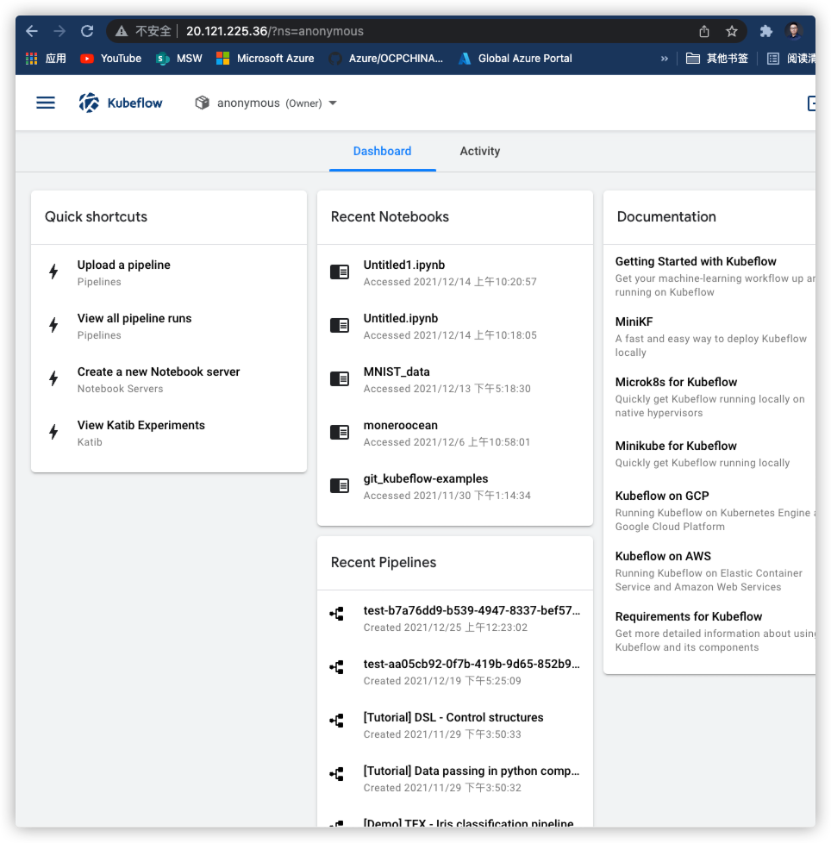
kubectl edit configmap workflow-controller-configmap -n kubeflow

**containerRuntimeExecutor: pns**

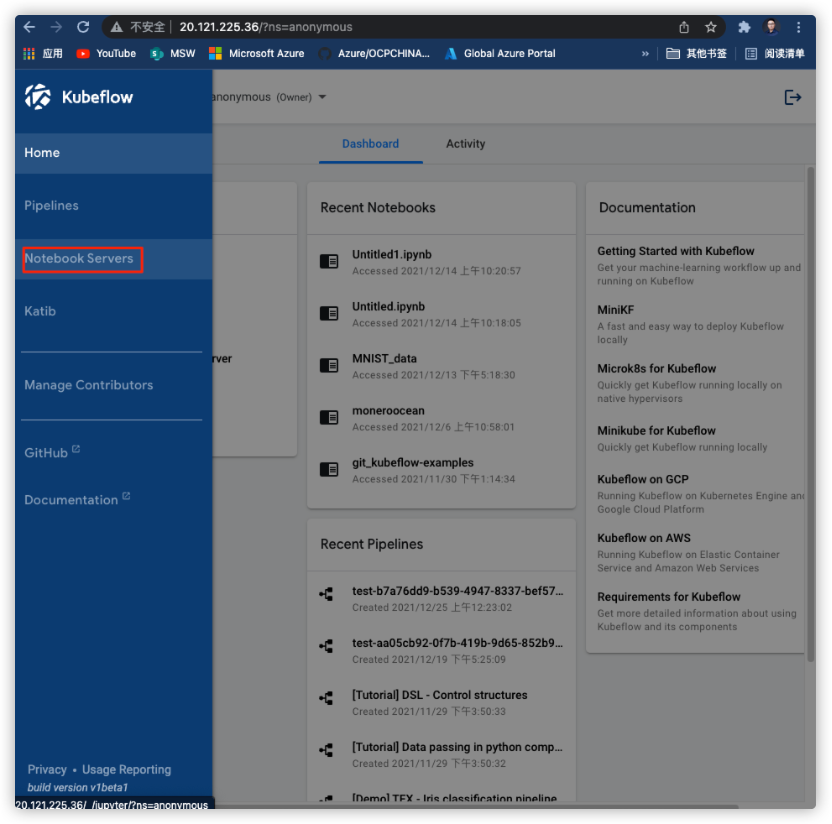
## 测试及演示

1. **在Kubeflow Dashboard上使用 Jupyter notebook运行机器学习训练**

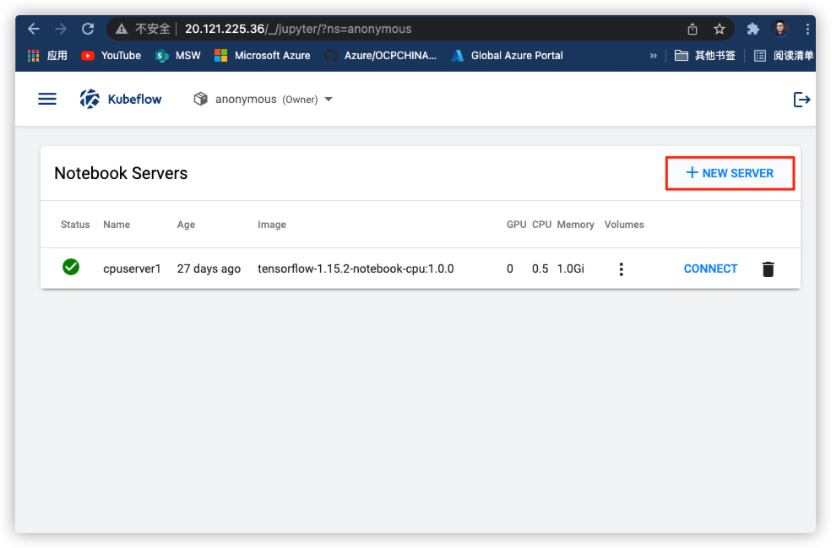
* **登陆Kubeflow Cental Dashboard**



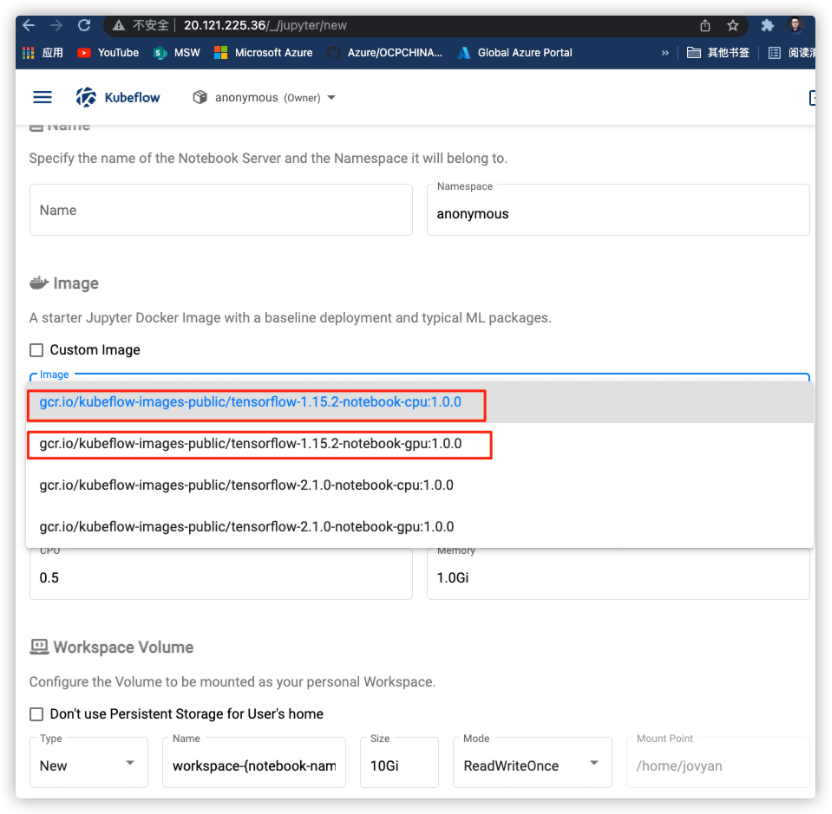
* **选择Notebook Server**



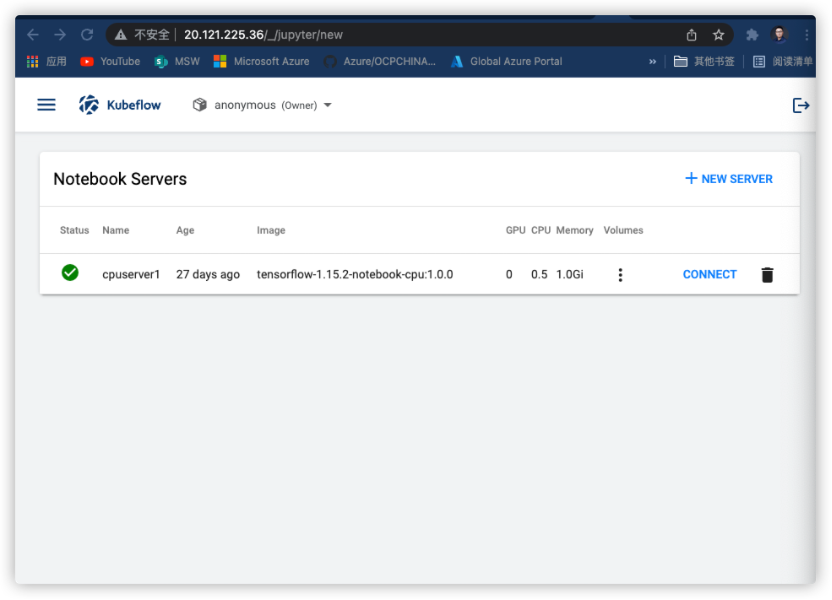
* **创建一个Notebook Server**

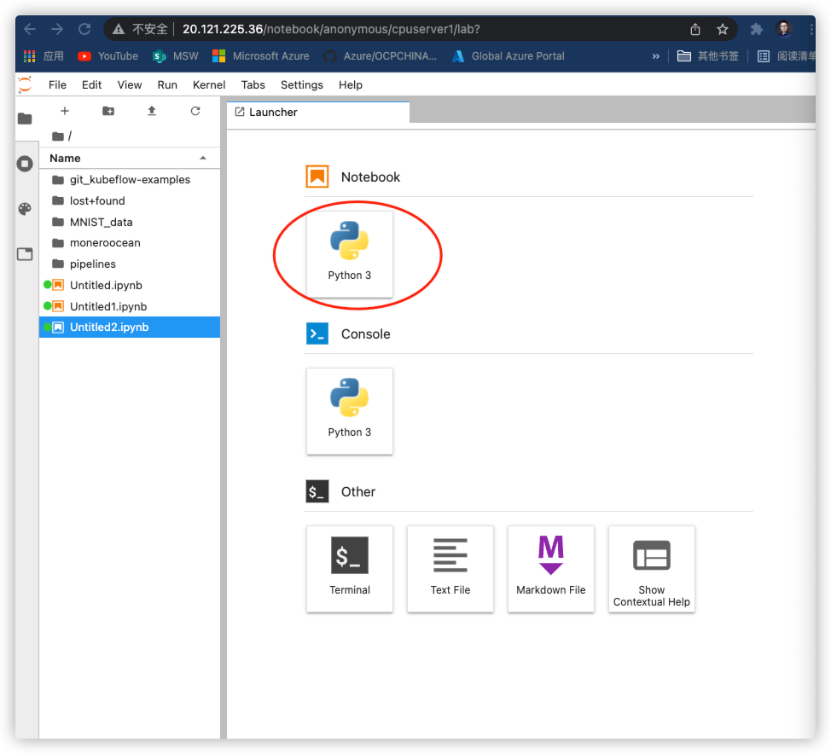


* **AKS默认支持tensorflow 1.15.2的CPU和GPU镜像，但允许使用客户化镜像，需要数据工程师或数据科学家自己准备镜像进行加载。我们例子里演示使用tensorflow 1.15.2的CPU或GPU镜像。**



* **连接Notebook Servers**

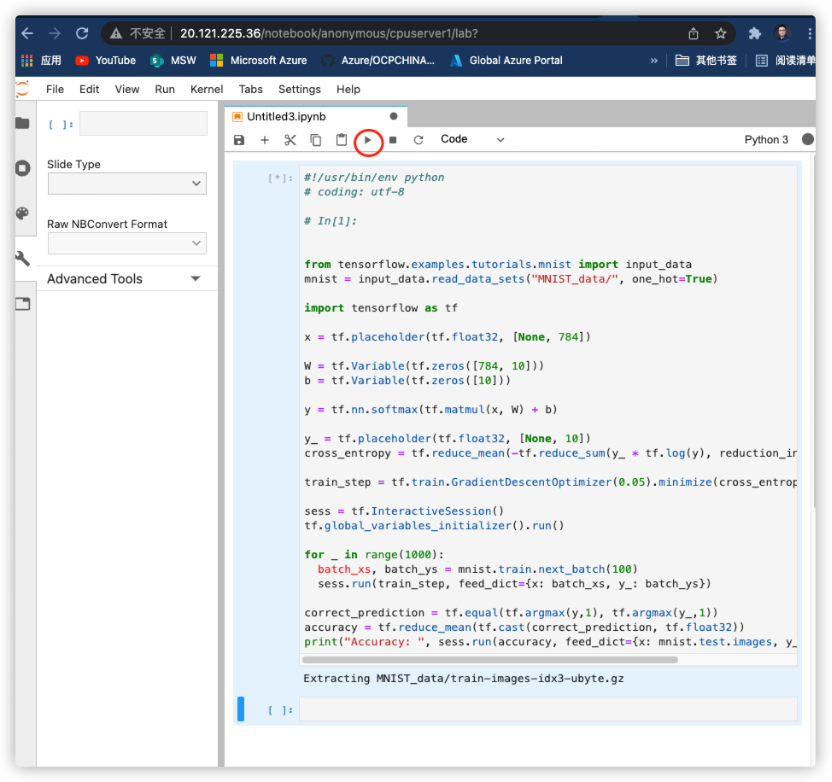
**将默认的http://20.121.225.36/notebook/anonymous/cpuserver1/tree?换成http://20.121.225.36/notebook/anonymous/cpuserver1/lab?，然后点击Python3图标，**



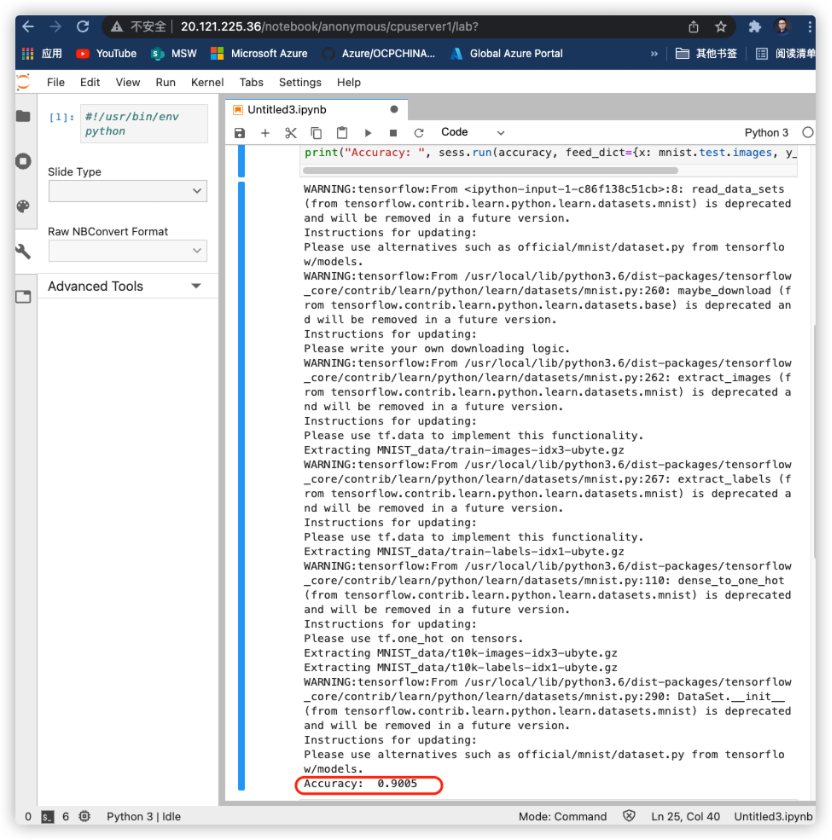
**运行一段mnist数据集训练集代码，Copy如下源代码到Jupyter notebook中，**

#!/usr/bin/env python  
# coding: utf-8  
  
# In[1]:  
  
  
from tensorflow.examples.tutorials.mnist import input\_data  
mnist = input\_data.read\_data\_sets("MNIST\_data/", one\_hot=True)  
  
import tensorflow as tf  
  
x = tf.placeholder(tf.float32, [None, 784])  
  
W = tf.Variable(tf.zeros([784, 10]))  
b = tf.Variable(tf.zeros([10]))  
  
y = tf.nn.softmax(tf.matmul(x, W) + b)  
  
y\_ = tf.placeholder(tf.float32, [None, 10])  
cross\_entropy = tf.reduce\_mean(-tf.reduce\_sum(y\_ \* tf.log(y), reduction\_indices=[1]))  
  
train\_step = tf.train.GradientDescentOptimizer(0.05).minimize(cross\_entropy)  
  
sess = tf.InteractiveSession()  
tf.global\_variables\_initializer().run()  
  
for \_ in range(1000):  
 batch\_xs, batch\_ys = mnist.train.next\_batch(100)  
 sess.run(train\_step, feed\_dict={x: batch\_xs, y\_: batch\_ys})  
  
correct\_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y\_,1))  
accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))  
print("Accuracy: ", sess.run(accuracy, feed\_dict={x: mnist.test.images, y\_: mnist.test.labels}))

**点击运行按钮，**

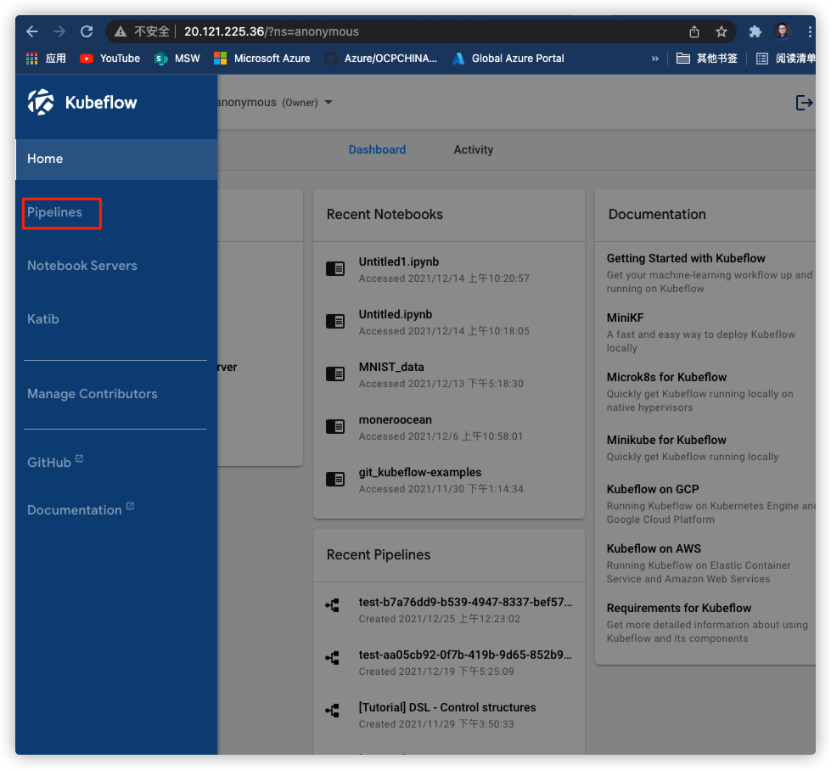


* **从运行结果可以看到准确率（中间会报出一些Warning，可以忽略）**

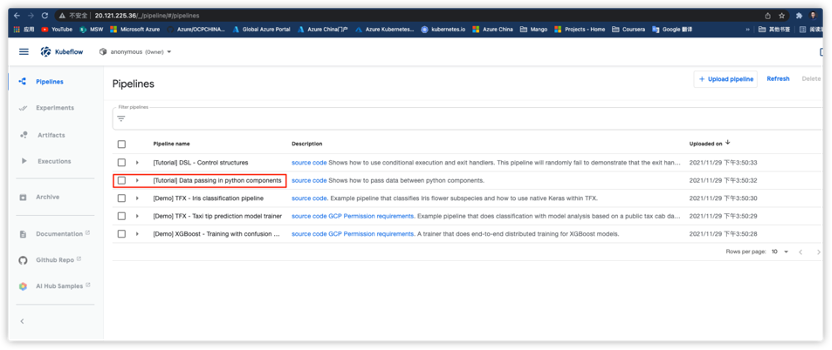


1. **运行Kubeflow Pipeline**

* **从Dashboard Home页面点击Pipelines**

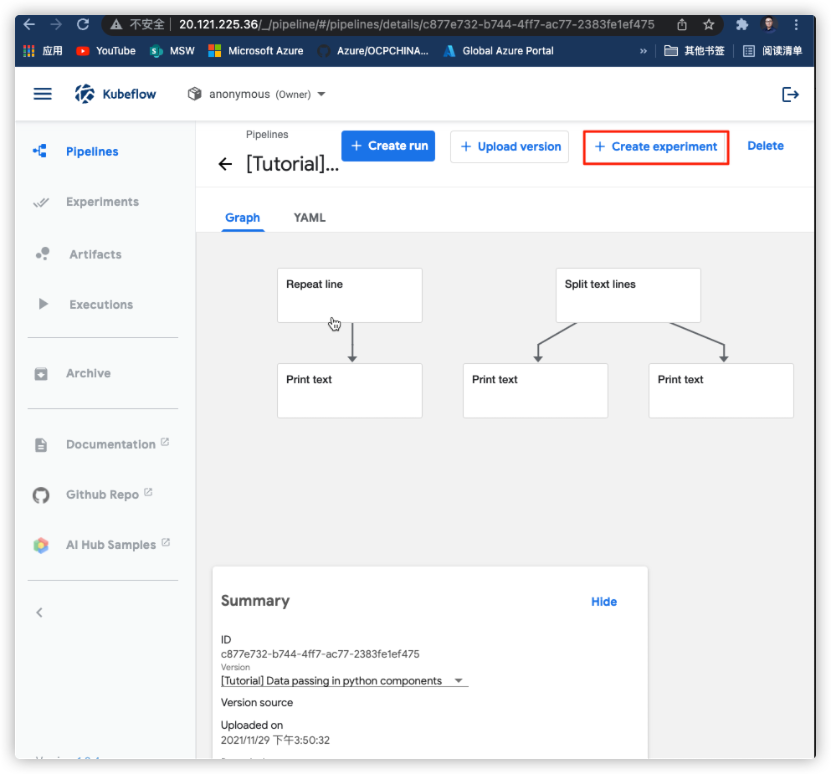


* **默认提供了5个pipeline用于例子体验，我们选择**[Tutorial] Data passing in python components 打开

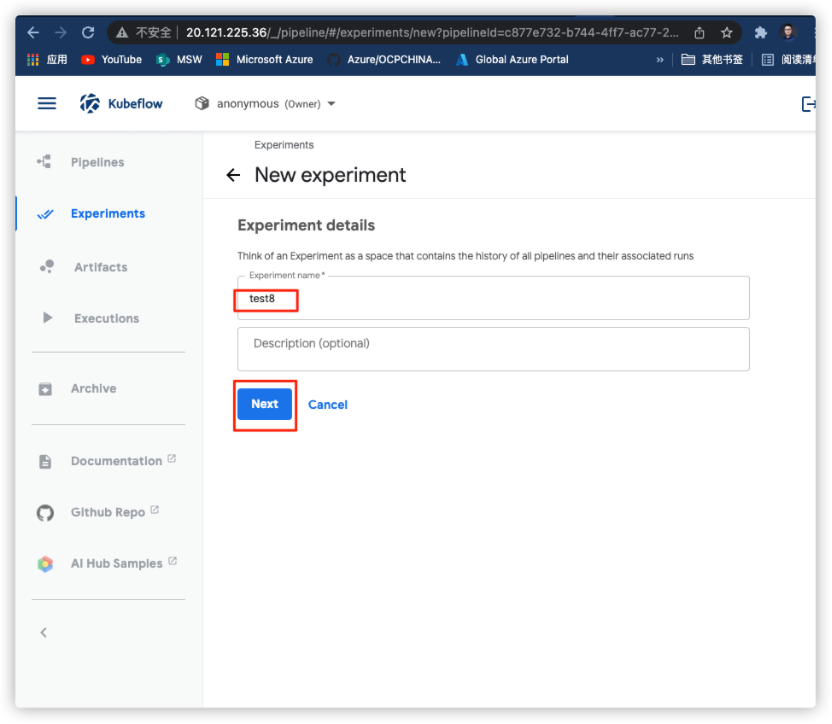


文本 描述已自动生成

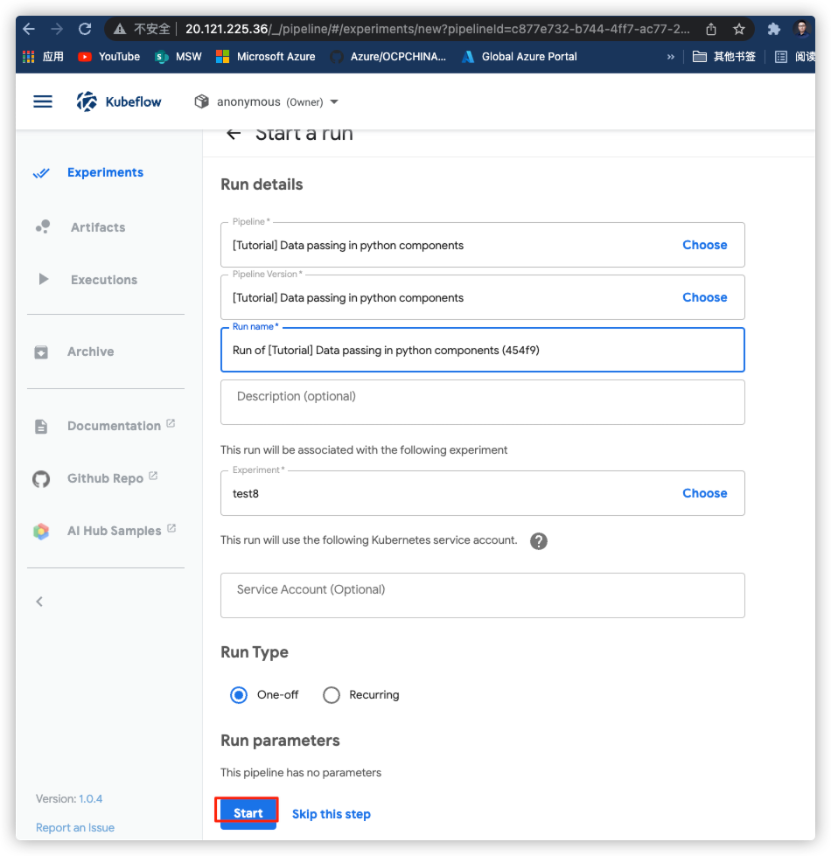
* **从打开的pipeline点击 Create experiment，创建数据pipeline实验环境，**



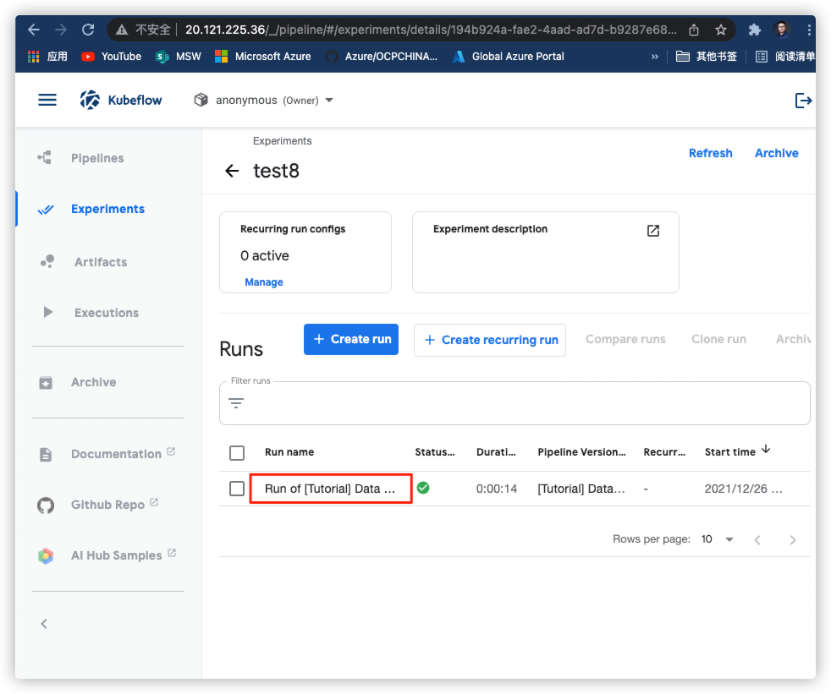
* **输入Experiment name，然后点击Next，**



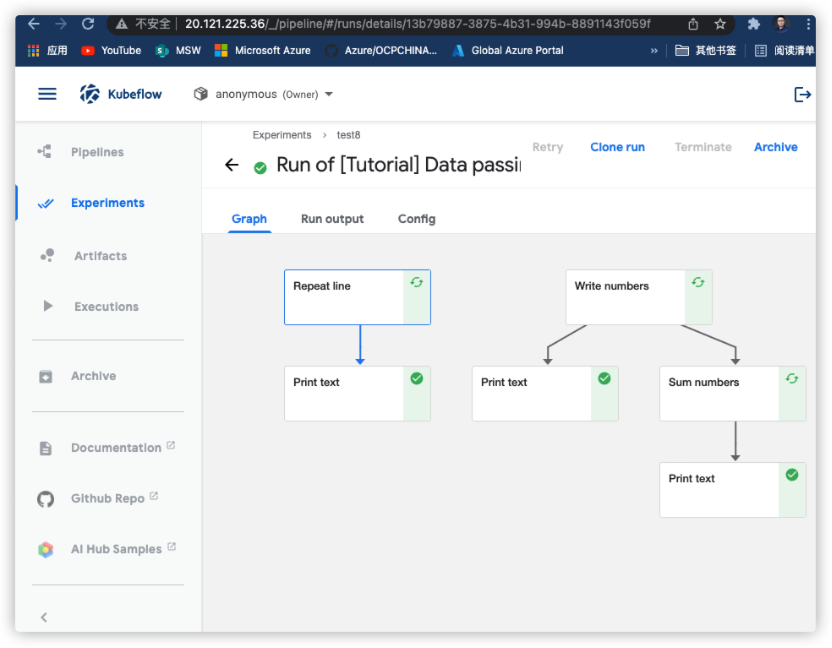
* **进入Start a run界面，点击Start 按钮运行该Experiment**



* **可以看到这个Experiment的运行记录，点击可以查看详细信息，**



* **以图形的方式直观的显示运行状态或结果，**



## 总结

Kubeflow 是一个由众多子项目组成的开源产品，它的愿景很简单：在 Kubernetes 上运行机器学习工作负载。目前，Kubeflow 主要的组件有 tf-operator，pytorch-operator，mpi-operator，pipelines，katib，kfserving 等。其中 tf-operator，pytorch-operator，mpi-operator 分别对应着对 TensorFlow，PyTorch 和 Horovod 的分布式训练支持；pipelines 是一个流水线项目，它基于 argo 实现了面向机器学习场景的流水线；katib 是基于各个 operator 实现的超参数搜索和简单的模型结构搜索的系统，支持并行的搜索和分布式的训练等；kfserving 是对模型服务的支持，它支持部署各个框架训练好的模型的在线推理服务。

Kubeflow 想解决的问题是如何基于 Kubernetes 去方便地维护 ML Infra。Kubeflow 中大多数组件的实现都是基于 Kubernetes Native 的解决方案，通过定义 CRD 来功能。这很大程度上减少了运维的工作。同时，利用 Kubernetes 提供的扩展性和调度能力，对大规模分布式训练和 AutoML 也有得天独厚的优势。

本文主要从实践角度在Azure Kubernetes Service 上搭建Kuberflow平台并进行基本测试体验，更多的功能大家可以参考[Kubeflow官方文档](https://www.kubeflow.org/)进行实践。