# Submarine

# 项目概况

* **概述**

Submarine 通过 notebook 支持使用 spark & python 进行数据处理和算法开发(数据预处理).

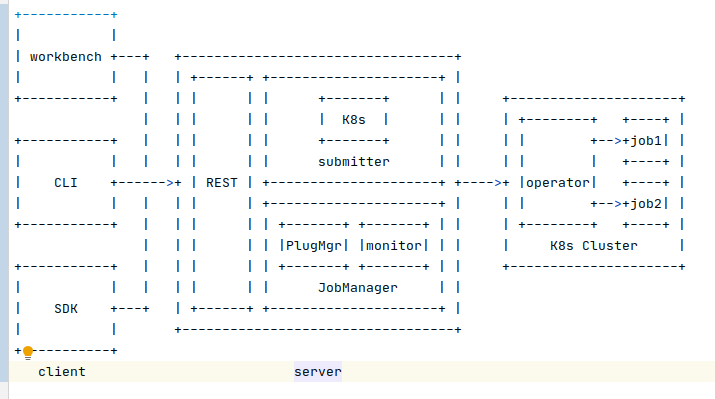
Submarine 支持多种机器学习框架进行模型训练。比如PyTorch,TensorFlow(机器学习)

Submarine 支持 Yarn、Kubernetes、Docker和资源调度。(基础设施)

Submarine 服务主要由九个组件组成，您可以使用以下命令查看它们：

kubectl get pods -n ${your\_namespace}

* tf-operator：使用户能够分布式运行 TensorFlow 任务
* pytorch-operator：使用户能够分布式运行 PyTorch 任务
* notebook-controller : Jupyter notebook控制器
* submarine-traefik: Kubernetes 入口控制器,反向代理,该组件非必需,可以不用
* submarine-database:存储元数据的 MySQL 数据库
* submarine-minio:机器学习工件的对象存储(可以保存模型),An object store for machine learning artifacts
* submarine-mlflow: 模型管理平台
* submarine-tensorboard: 分布式训练实验的可视化工具
* submarine-server: 处理API请求，并将分布式训练实验提交给 Kubernetes。
* **三种方式**



Workbench:通过submarine管理UI进行操作

CLI:安装submarine SDK的时候就会附带Submarine CLI ,然后通过命令行可以操作管理submarine



SDK:在submarine代码submarine/submarine-sdk/pysubmarine目录下,运行

pip install .

安装submarine SDK,然后可以通过sdk操作管理

# 代码结构

Submarine支持三种方式运行，CLI/SDK/UI

* submarine-client：为用户提供CLI接口(目前仅支持 YARN 服务)，通过CLI接口运行submarine
* submarine-cloud-v2:Submarine程序的操作员（操作员为用户提供了一个新选项，可以将Submarine服务部署到他们的Kubernetes集群。）
* submarine-commons：定义多个包中使用的公用方法，主要与hadoop相关。
* submarine-dist：存储预发布文件
* submarine-sdk：为Submarine用户提供python SDK，通过sdk运行submarine
* submarine-server：包含核心服务器，restful风格接口，k8s提交者
* submarine-test：提供submarine端对端和k8s提交者测试
* submarine-workbench:分两部分，workbench-server和workbench-web

workbench-server: 是一个基于Jetty的web服务器服务。Workbench server提供RESTful和Websocket接口。RESTful界面为workbench web提供了项目、部门、用户和角色等数据库的管理功能。

 workbench-web：基于Angular.js框架的web前端服务。Workbench-web用户可以通过浏览器管理submarine项目、部门、用户、角色。您还可以使用notebook开发机器学习算法、模型发布和其他生命周期管理。

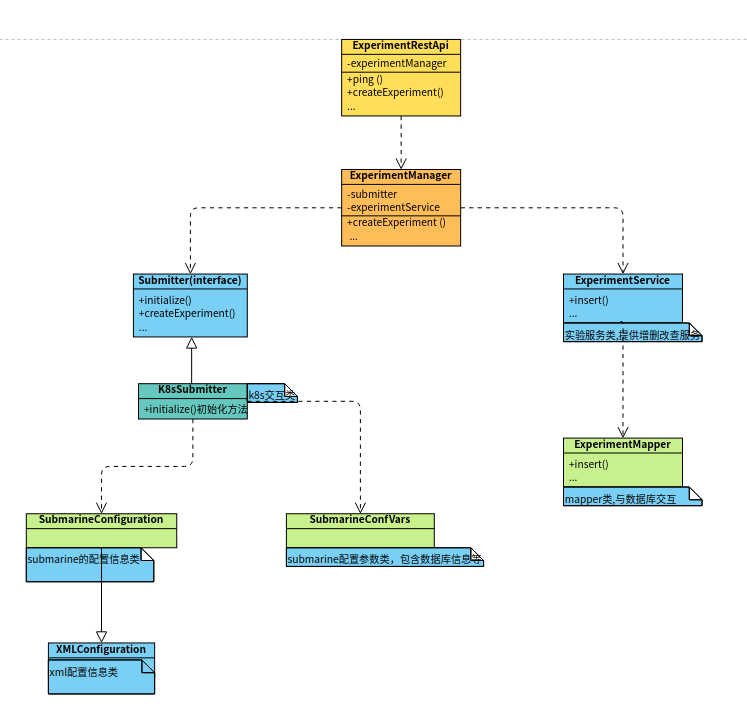
* dev-support：分两部分，mini-submarine和submarine-installer

mini-submarine：通过Submarine提供的docker镜像，您可以在单个docker环境中体验Submarine的所有功能，而mini-submarine还为开发人员提供了一个开发和测试环境，避免了安装和部署运行时环境的麻烦。

submarine-installer：subside-installer是我们针对yarn-3.1+及以上版本的submarine运行时环境安装工具。通过使用submarine-installer，可以轻松安装和部署像docker，nvidia-docker, nvidia driver, ETCD, Calico network等系统服务。

**类图:**

首先按Experiment接口举例,其他接口可以参考如下类图关系



**流程:**

参考如上类图,如下描述请求流程

1. 客户端发送Experiment接口请求
2. ExperimentRestApi接收到客户端请求,实例化ExperimentManager类,,调用ExperimentManager对象处理客户端请求
3. ExperimentManager在初始化的时候,会加载SubmitterManager,SubmitterManager初始化时加载k8s相关的jar到类加载器,通过反射实例化Submitter接口的实现类K8sSubmitter,他的主要作用是调用K8sSubmitter与k8s交互,其次实例化ExperimentService服务类调用ExperimentService与数据库交互,以及处理一些简单的逻辑
4. K8sSubmitter首先通过XMLConfiguration的实现类SubmarineConfiguration加载配置信息(解析submarine-site.xml文件和类SubmarineConfVars,SubmarineConfVars类是submarine的配置类,其中包含端口,数据库,账户,密码等配置信息)

其次调用initialize()方法进行初始化操作(加载KubeConfig配置信息,初始化ApiClient客户端),ApiClient用于调用k8s接口,但是submarine在使用的时候,又会对ApiClient进行封装,针对不同请求封装成不同的client,比如获取pod信息,则封装使用podClient.

K8sSubmitter在进行上面操作后,会根据ExperimentManager调用的方法调用对应的接口与k8s交互,并拿到k8s的处理结果,对结果进行解析处理后返回ExperimentManager

1. ExperimentService调用MyBatisUtil工具类,得到SqlSession,SqlSession用于和数据库交互

通过SqlSession,可以得到Mapper代理类,比如Experiment的代理类是ExperimentMapper,得到ExperimentMapper,就可以操作数据表experiment,对表进行增删改查操作

**关键类：**

* submarine-server/server-core/目录下的SubmitterManager.java类：k8s配置和环境初始化相关的类
* K8sSubmitter:k8s交互类
* ApiClient:调用k8s接口的客户端
* SubmarineConfiguration:submarine配置信息管理类,加载配置信息(解析submarine-site.xml文件和类SubmarineConfVars,
* SubmarineConfVars:submarine配置信息类,其中包含端口,数据库,账户,密码等配置信息
* MyBatisUtil:mybatis工具类

# 依赖环境

* **版本要求**

Java 8(1.8)

Kubernetes v1.21.2

Maven 3.6.3

Docker latest

Helm v3.8.1

NodeJS 14(latest)

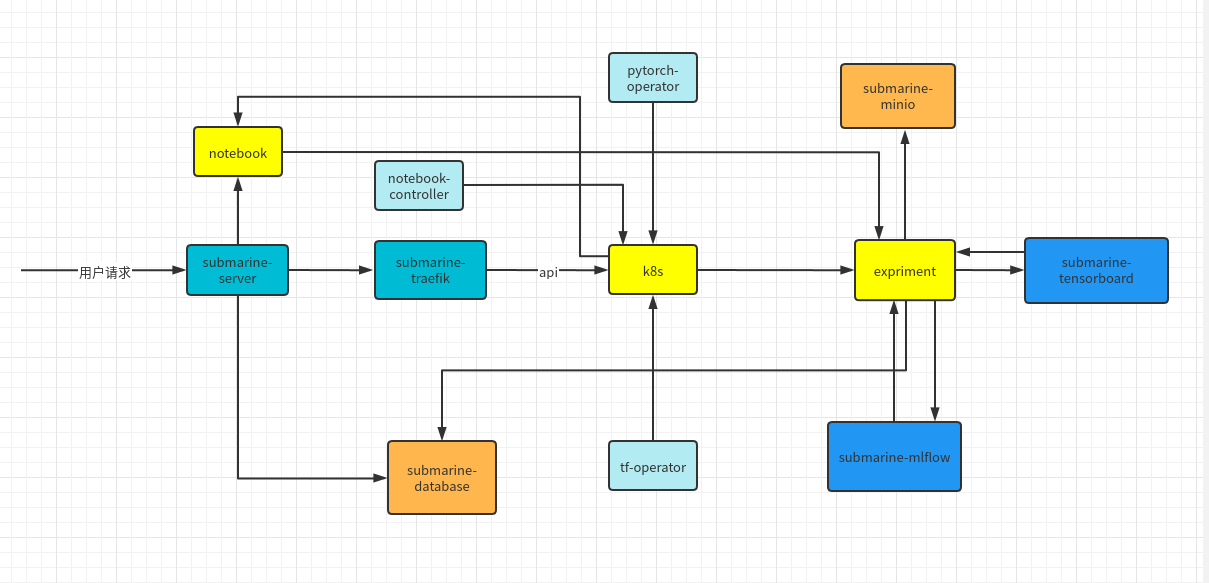
Go 1.16

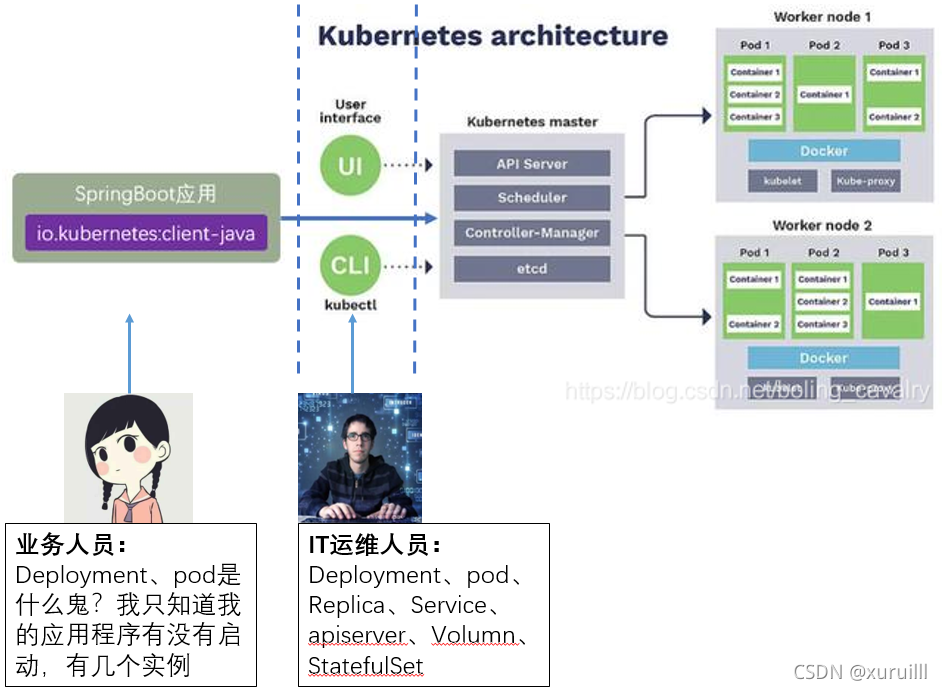
Python 3.7

* Kubernetes：k8s，一个可移植的、可扩展的开源平台，用于管理容器化的工作负载和服务。Kubernetes 为你提供服务发现和负载均衡、存储编排、自动部署和回滚、自动完成装箱计算、自我修复、密钥与配置管理。
* KinD：kind是另一个Kubernetes SIG项目，但它与minikube有很大区别。它可以将集群迁移到Docker容器中，这与生成虚拟机相比，启动速度大大加快。简而言之，kind是一个使用Docker容器节点运行本地Kubernetes集群的工具（CLI）。
* Java：Submarine开发语言
* Maven：Java三方依赖的管理工具
* Docker：容器引擎
* Helm：Helm是k8s的包管理工具，类似Linux系统常用的 apt。使用helm可以简化k8s应用部署。一个 Helm 包，其中包含了运行一个应用所需要的镜像、依赖和资源定义等，还可能包含 Kubernetes 集群中的服务定义。
* NodeJS：Node.js是一个事件驱动I/O服务端JavaScript环境，基于Google的V8引擎，V8引擎执行Javascript的速度非常快，性能非常好。
* Go：被设计成一门应用于搭载 Web 服务器，存储集群或类似用途的巨型中央服务器的系统编程语言。
* Python：模型使用的开发语言

# **架构**

**架构图**





Submarince是基于client-java完成与K8S的交互

# **快速体验**

minikube ：Minikube是一种轻量化的Kubernetes集群，是Kubernetes社区为了帮助开发者和学习者能够更好学习和体验k8s功能而推出的，借助个人PC的虚拟化环境就可以实现Kubernetes的快速构建启动。（Minikube其实就是一个在本地运行的 Kubernetes）

* **环境搭建**

Kubernetes v1.21.2

Docker latest

Helm v3.8.1

minikube v1.25.2

kubectl v1.23.5

注：minikube 和kubectl 版本关系不能差距太大

* **运行命令**

minikube start --vm-driver=docker --cpus 4 --memory 4096 --image-mirror-country='cn' --image-repository='registry.cn-hangzhou.aliyuncs.com/google\_containers' --kubernetes-version v1.21.2

--vm-driver：指定启动驱动

--cpus:为minikube虚拟机分配CPU核数

--memory:为minikube虚拟机分配内存数

--image-mirror-country:需要使用的镜像镜像的国家/地区代码

--image-repository：镜像加速地址

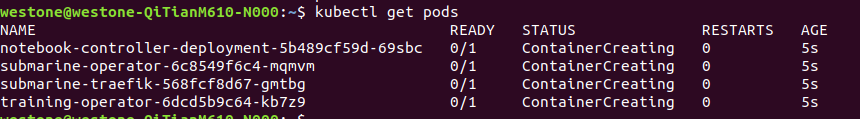
--kubernetes-version：minikube 虚拟机将使用的 kubernetes 版本

其次运行如下命令

git clone <https://github.com/apache/submarine.git>

cd submarine

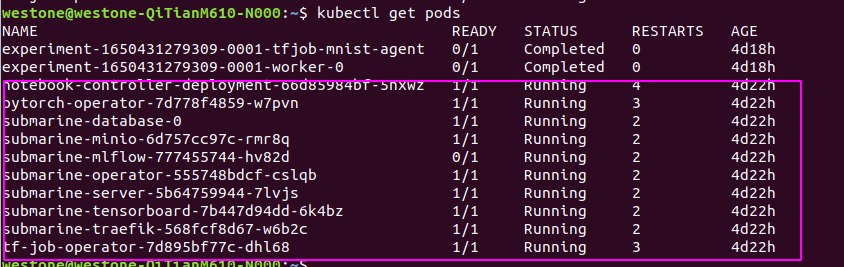
helm install submarine ./helm-charts/submarine



kubectl apply -f submarine-cloud-v2/artifacts/examples/example-submarine.yaml



kubectl get pods查看服务状态（需要等很长时间）



查看系统

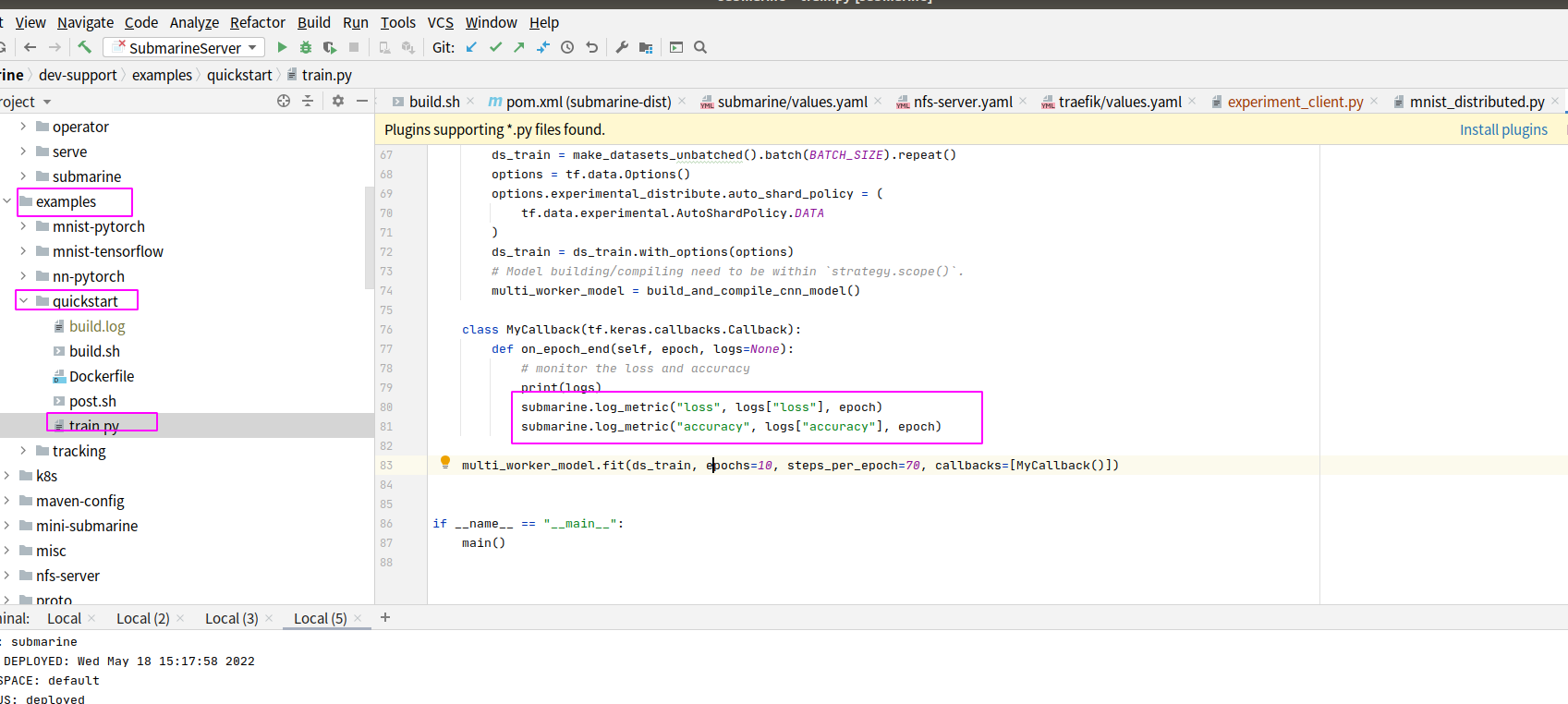
minikube ip：查看服务ip

http://{minikube ip}:32080

账户密码默认：admin/admin



上面Charts指标写入数据库位置如下:

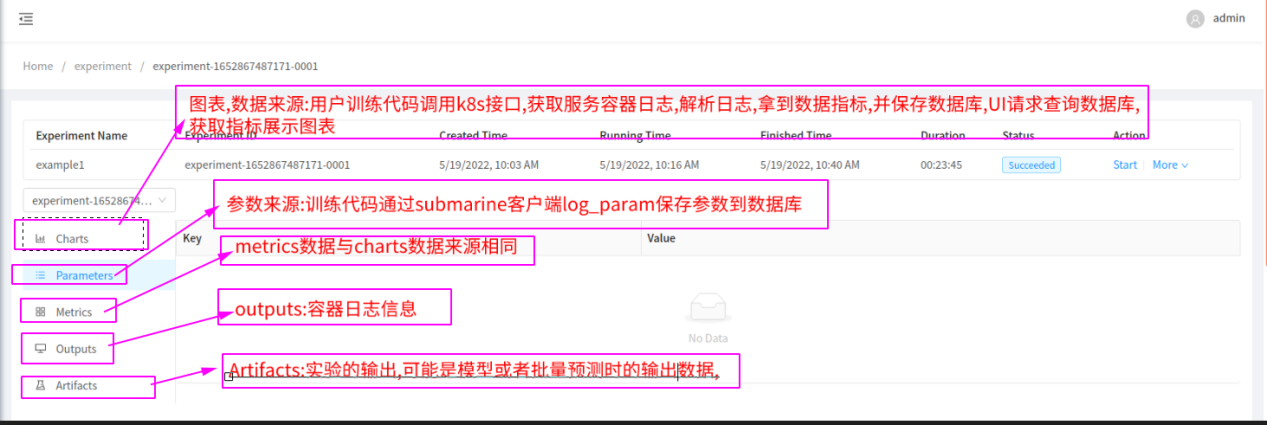


服务启动后，可以运行如下命令查看服务日志状态等

minikube dashboard

* **指标说明**

实验指标:



/api/v1/experiment/logs/experiment-1650431279309-0001查看运行日志

192.168.49.2：32080/api/metric/selective获取运行指标数据

# **本地开发**

* **运行前端**

步骤1：安装npm

步骤2：cd 到workbench-web目录下，运行命令：npm install

如果报如下错误：



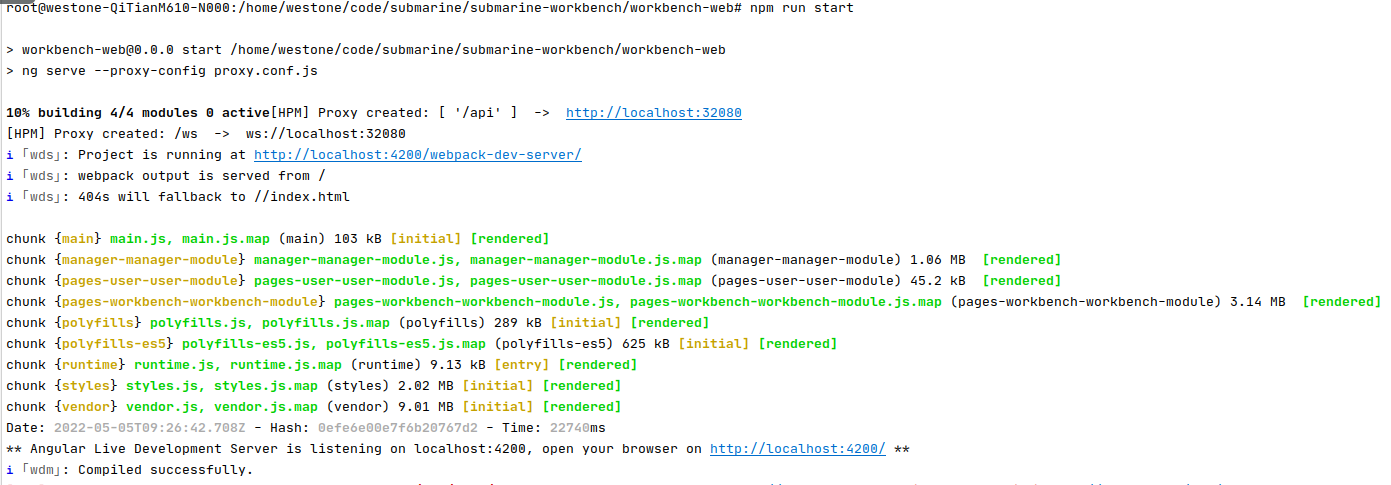
解决方法：

执行命令：vim /etc/sysctl.conf，添加如下内容：

fs.inotify.max\_user\_watches=524288

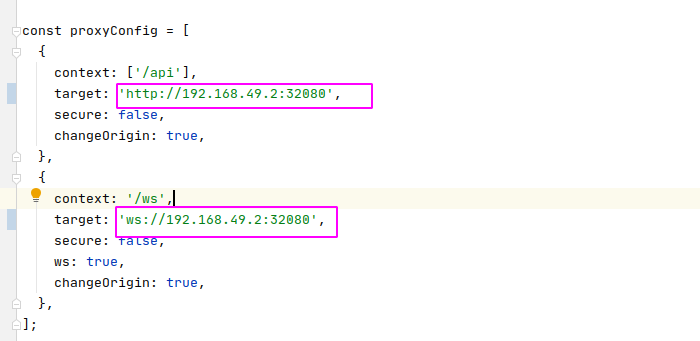
然后再执行命令：sysctl -p

步骤3：执行命令 npm run start



修改连接后端地址

在目录workbench-web下的文件proxy.conf.js，修改后端api地址和ws地址



* **运行后端**

进入submarine源代码目录下，运行命令：

mvn clean

mvn install -DskipTests

在 minikube 中构建新的服务器 docker 镜像

|  |
| --- |
| *# switch to minikube docker daemon to build image directly in minikube* eval $(minikube docker-env)  *# run docker build* ./dev-support/docker-images/submarine/build.sh  *# exit minikube docker daemon* eval $(minikube docker-env -u) |

删除服务器部署，k8s的operator将使用新映像创建一个新服务

kubectl delete deployment submarine-server

注意事项

如果mvn install -DskipTests报错,可以将submarine-server/server-api下的pom.xml文件如下图中的代码注释掉,submarine-server/server-core下的pom.xml同样的代码也要注释掉



然后再执行mvn install -DskipTests命令

* **查看数据库**

运行命令 docker ps -a | grep database



docker exec -it 55dcaf86f552 bash执行命令进入容器

mysql -usubmarine -p 密码：password

show database;

use submarine;

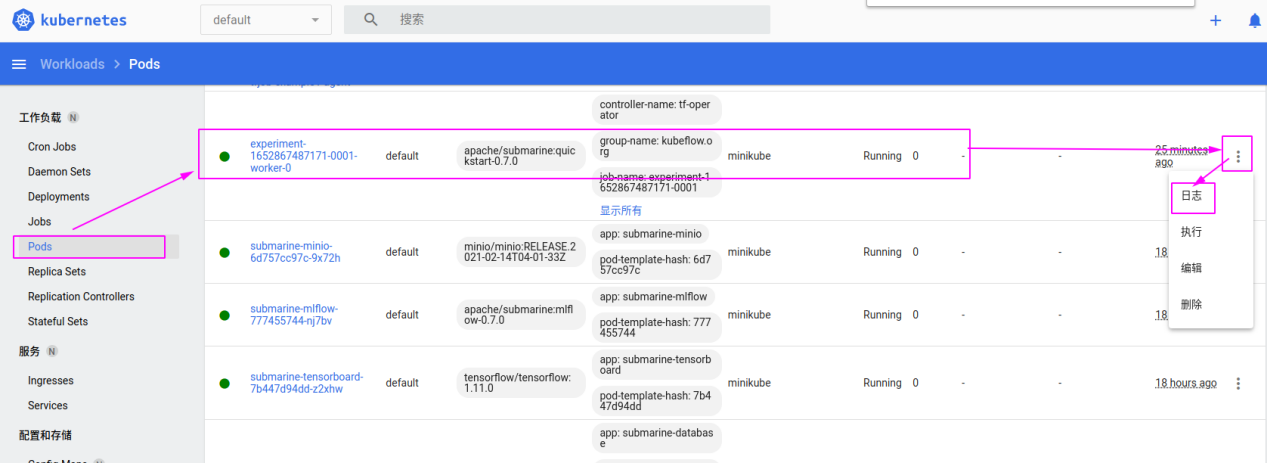
show tables;

select \* from experiment;

* **查看容器状态或日志**

输入命令:minikube dashboard ,打开k8s的dashboard,可以看到容器相关的状态,日志,资源使用率,创建时间等等.

如下截图步骤就可以看到日志





# **Helm**

* **概述**

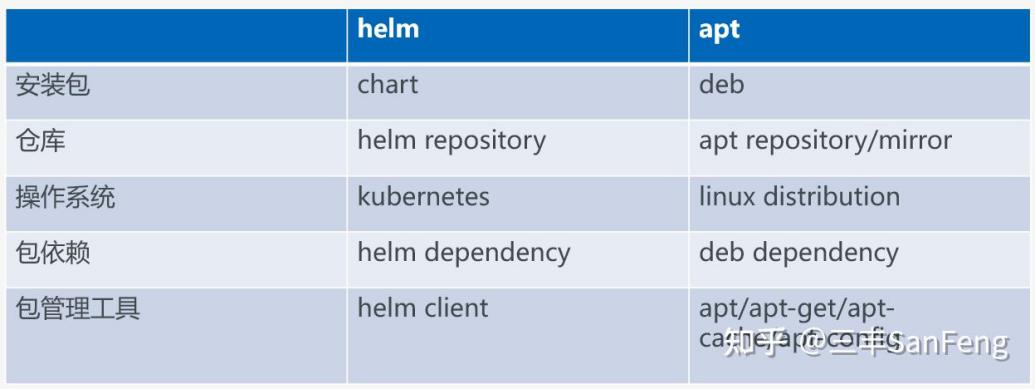
https://helm.sh/docs/topics/charts/

Helm 是 Kubernetes 的包管理器

Chart是一个Helm 包。它包含在 Kubernetes 集群内运行应用程序、工具或服务所需的所有资源定义。可以把它想象成 Kubernetes 的 Homebrew 公式、Apt dpkg 或 Yum RPM 文件。

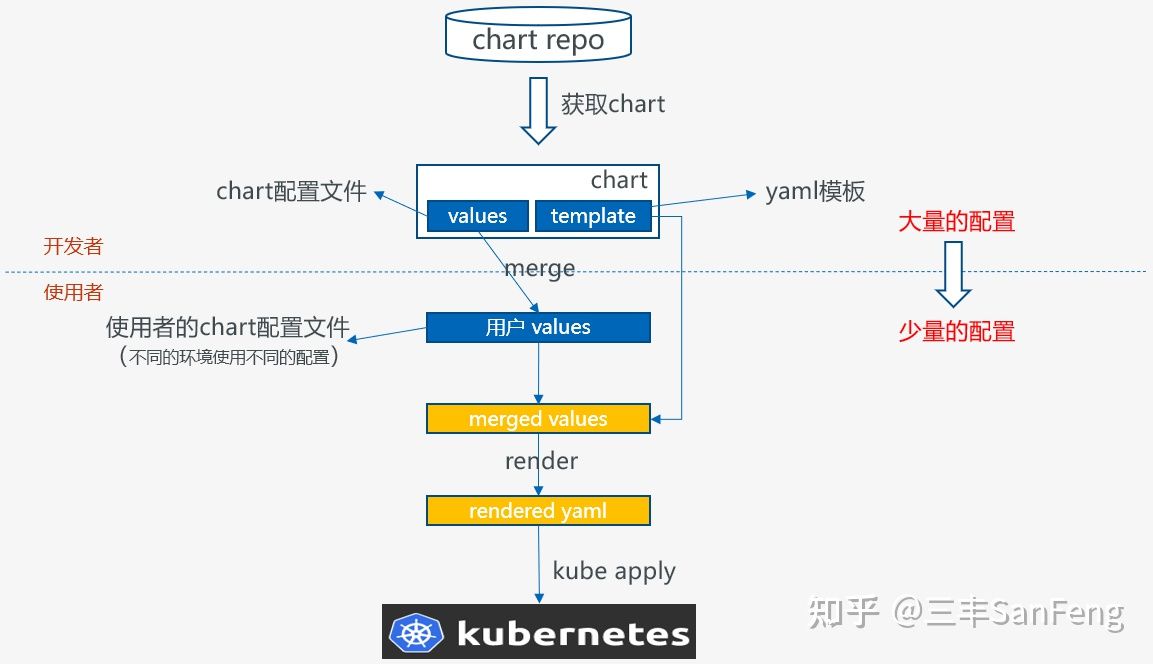
存储库是可以收集和共享图表的地方。它类似于 Perl 的 CPAN 存档或 Fedora 包数据库，但用于 Kubernetes 包。

Release是在 Kubernetes 集群中运行的图表的实例。一个图表通常可以多次安装到同一个集群中。每次安装时，都会创建一个新版本。考虑一个 MySQL 图表。如果您希望在集群中运行两个数据库，则可以将该图表安装两次。每个都有自己的发行版，而发行版又会有自己的发行版名称。



helm的核心运行流程:

* 从chart仓库中获取chart；
* 使用者配置自己的values文件，根据自己的运行环境对values进行修改；
* 默认values文件和使用者values文件会进行一个merge，形成最终的values文件；
* 使用最终的values文件，渲染chart的template，形成可以被kubernetes执行的yaml；
* 调用kube apply提交yaml到kubernetes



* **常用命令**

helm repo add bitnami https://charts.bitnami.com/bitnami

添加chart存储库bitnami

helm search repo bitnami

列出可以安装的chart

安装chart,前提要先启动kubenates

helm repo update

helm install bitnami/mysql --generate-name

helm show chart bitnami/mysql 或

helm show all bitnami/mysql

获取有关图表的所有信息

helm list 或helm ls

显示所有已部署版本的列表

helm list --all

helm uninstall mysql-1612624192

卸载版本

helm get -h 或 helm help helm -h

阅读帮助文本

helm repo list

查看配置了哪些存储库

helm repo add xx

添加新的存储库

helm repo update

更改图表存储库

helm repo remove

删除存储库



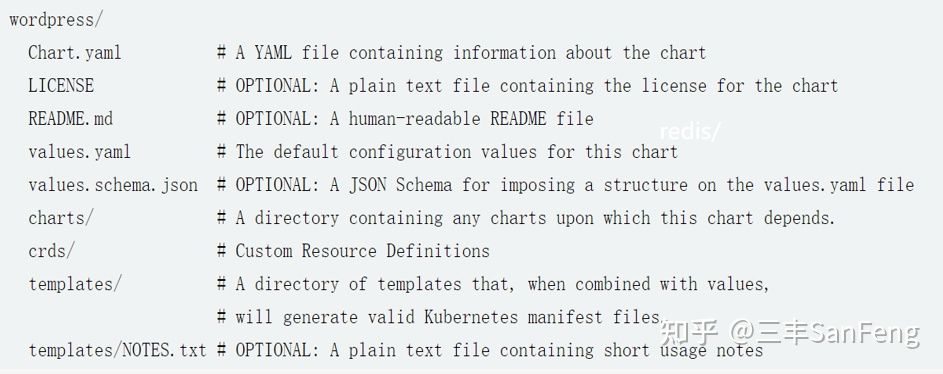
资料:

<https://zhuanlan.zhihu.com/p/350328164>

https://blog.csdn.net/yujia\_666/article/details/121139896

* **chart详解**

下图是wordpress的helm chart目录结构，每个元素的解释如下：



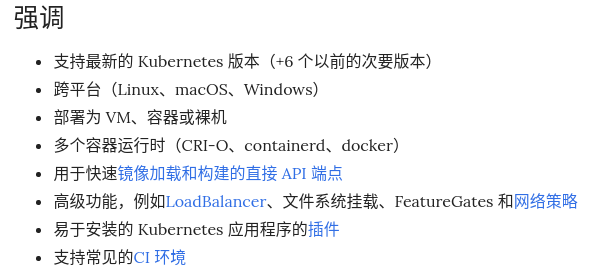
* Chart.yaml: chart的信息，包括chart的版本信息、描述信息、依赖关系，等
* LICENSE: (可选)chart的LICENSE信息
* README.md: (可选)chart的说明文件
* values.yaml: chart的默认配置信息
* values.schema.json: (可选) values配置信息的元信息（字段类型、字段描述、字段之间的依赖等），格式json
* charts: 依赖的其他chart
* crds: Custom Resource Definitions
* templates: 部署模板，结合values.yaml会渲染出kubernetes yaml文件
* templates/NOTES.txt: (可选)安装说明

# **Minikube**

* **概述**

<https://minikube.sigs.k8s.io/docs/>

minikube 在 macOS、Linux 和 Windows 上快速搭建本地 Kubernetes 集群



* **常用命令**

minikube start

启动集群

kubectl get po -A 或 minikube kubectl -- get po -A

访问集群

minikube dashboard

查看Kubernetes Dashboard

创建一个示例部署并在端口 8080 上公开它

kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.4

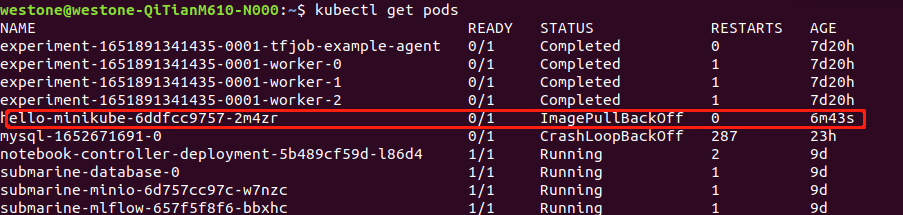
kubectl expose deployment hello-minikube --type=NodePort --port=8080

查看部署服务

kubectl get services hello-minikube

查看服务是否正常

kubectl get pods



访问此服务的最简单方法是让 minikube 为您启动 Web 浏览器

minikube service hello-minikube

在不影响已部署应用程序的情况下暂停 Kubernetes：

minikube pause

取消暂停暂停的实例：

minikube unpause

停止集群：

minikube stop

删除所有 minikube 集群：

minikube delete --all

# **Kubectl**

* **概述**

kubectl 是 Kubernetes 集群的命令行工具,通过 kubectl 能够对集群本身进行管理,并能够在集群上进行容器化应用的安装部署。

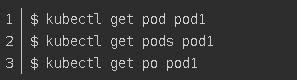
* **常用命令**

kubectl 命令的语法



(1)comand:指定要对资源执行的操作,例如 create、get、describe 和 delete

(2)TYPE:指定资源类型,资源类型是大小写敏感的,开发者能够以单数、复数和缩略的形式。例如:



1. NAME:指定资源的名称,名称也大小写敏感的。如果省略名称,则会显示所有的资源,例如:

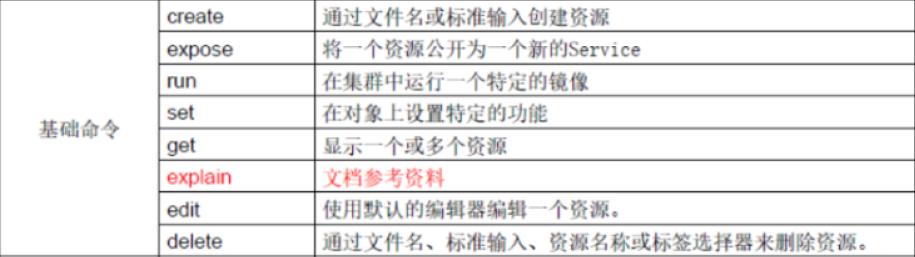


1. flags:指定可选的参数。例如,可用-s 或者–server 参数指定 Kubernetes API server 的地址和端口。

kubectl help 获取更多信息



1. 基础命令



1. 部署和集群管理命令



1. 故障和调试命令



1. 其他命令



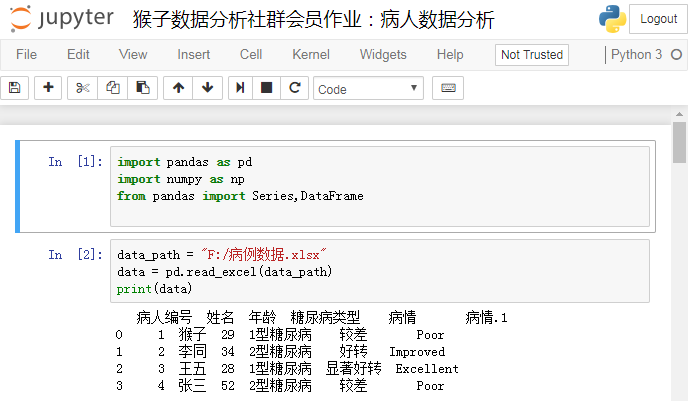
# **Jupyter Notebook**

* **概述**

Jupyter项目是一个非盈利的开源项目，源于2014年的ipython项目，因为它逐渐发展为支持跨所有编程语言的交互式数据科学和科学计算

Jupyter notebook（http://jupyter.org/） 是一种 Web 应用，能让用户将说明文本、数学方程、代码和可视化内容全部组合到一个易于共享的文档中。

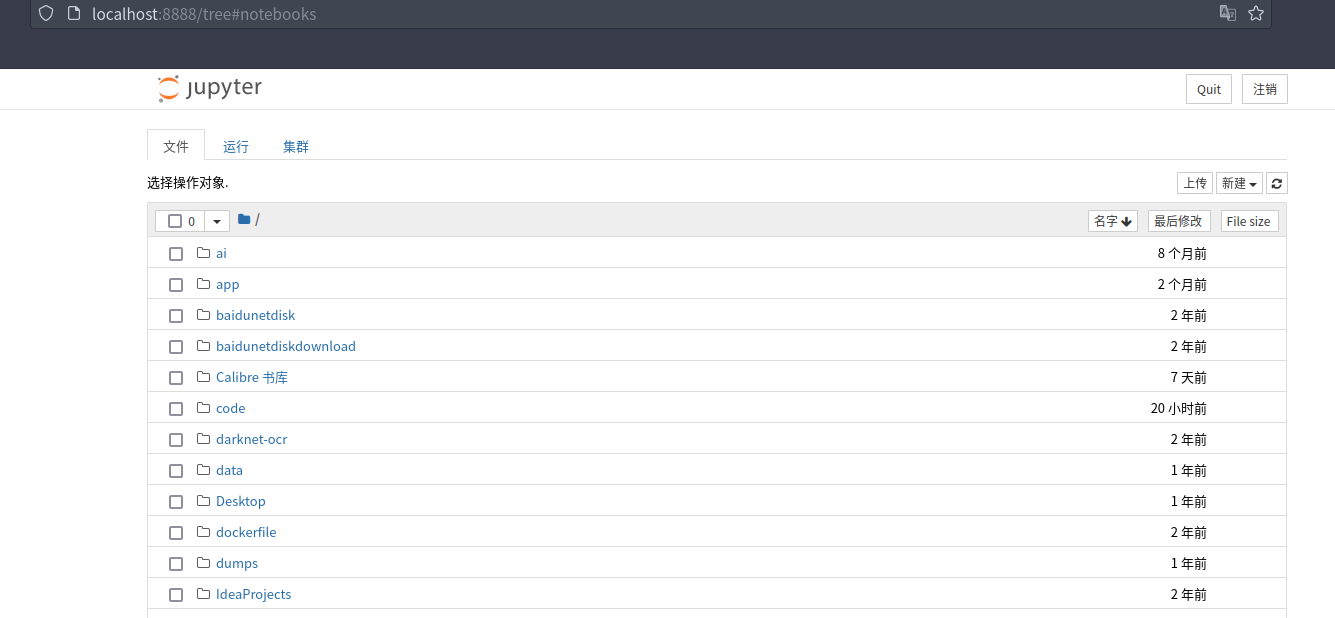
在没有notebook之前，在IT领域工作的我都是这样工作的：在普通的 Python shell 或者在IDE（集成开发环境）如Pycharm中写代码，然后在word中写文档来说明你的项目。这个过程很繁琐，通常是写完代码，再写文档的时候我还得重头回顾一遍代码。最蛋疼的地方在于，有些数据分析的中间结果，我还的重新跑代码，然后把结果弄到文档里给客户看。有了notebook之后，我的世界突然美好了许多，因为notebook 可以直接在代码旁写出叙述性文档，而不是另外编写单独的文档。也就是它可以能将代码、文档等这一切集中到一处，让用户一目了然。



* **如何启动**

本机输入命令:jupyter notebook,就可以启动





资料:

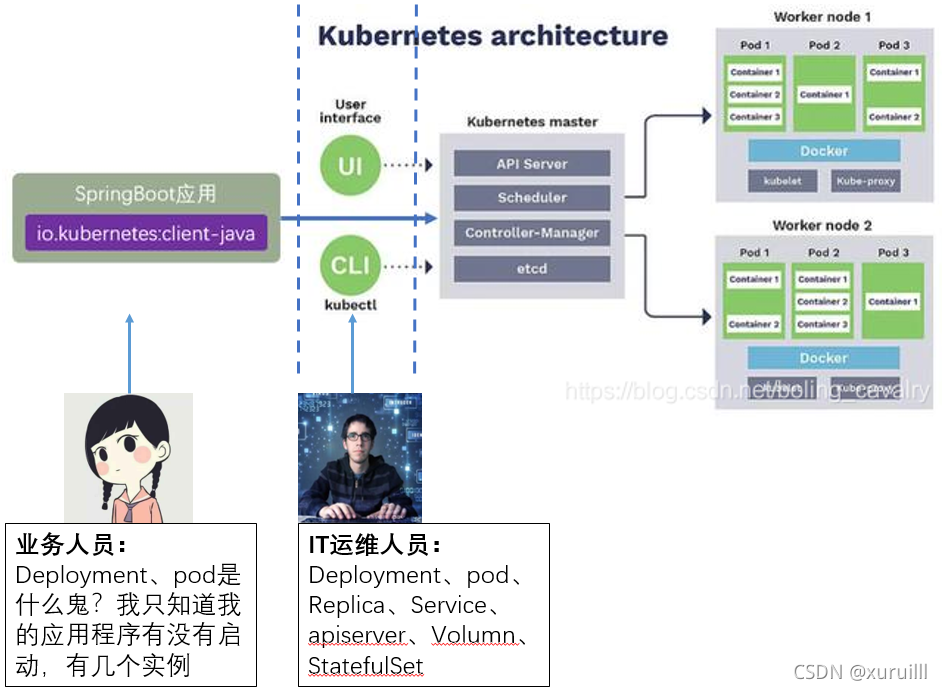
<https://www.zhihu.com/question/46309360/answer/254638807>

https://blog.csdn.net/mengxianglong123/article/details/124751871

# **client-java k8s**

* **概述**

与K8S交互的程序,对Deploymen、pod、service等对象的查询、创建、删除、伸缩等操作。



资料:

https://blog.csdn.net/xuruilll/article/details/121292675

# **Tensorboard**

* **概述**

Tensorboard原本是Google TensorFlow的可视化工具，可以用于记录训练数据、评估数据、网络结构、图像等，并且可以在web上展示，对于观察神经网络的过程非常有帮助。

对大部分人而言，深度神经网络就像一个黑盒子，其内部的组织、结构、以及其训练过程很难理清楚，这给深度神经网络原理的理解和工程化带来了很大的挑战。为了解决这个问题，tensorboard应运而生。Tensorboard是tensorflow内置的一个可视化工具，它通过将tensorflow程序输出的日志文件的信息可视化使得tensorflow程序的理解、调试和优化更加简单高效。Tensorboard的可视化依赖于tensorflow程序运行输出的日志文件，因而tensorboard和tensorflow程序在不同的进程中运行。

* **快速开始**

编写文件test.py

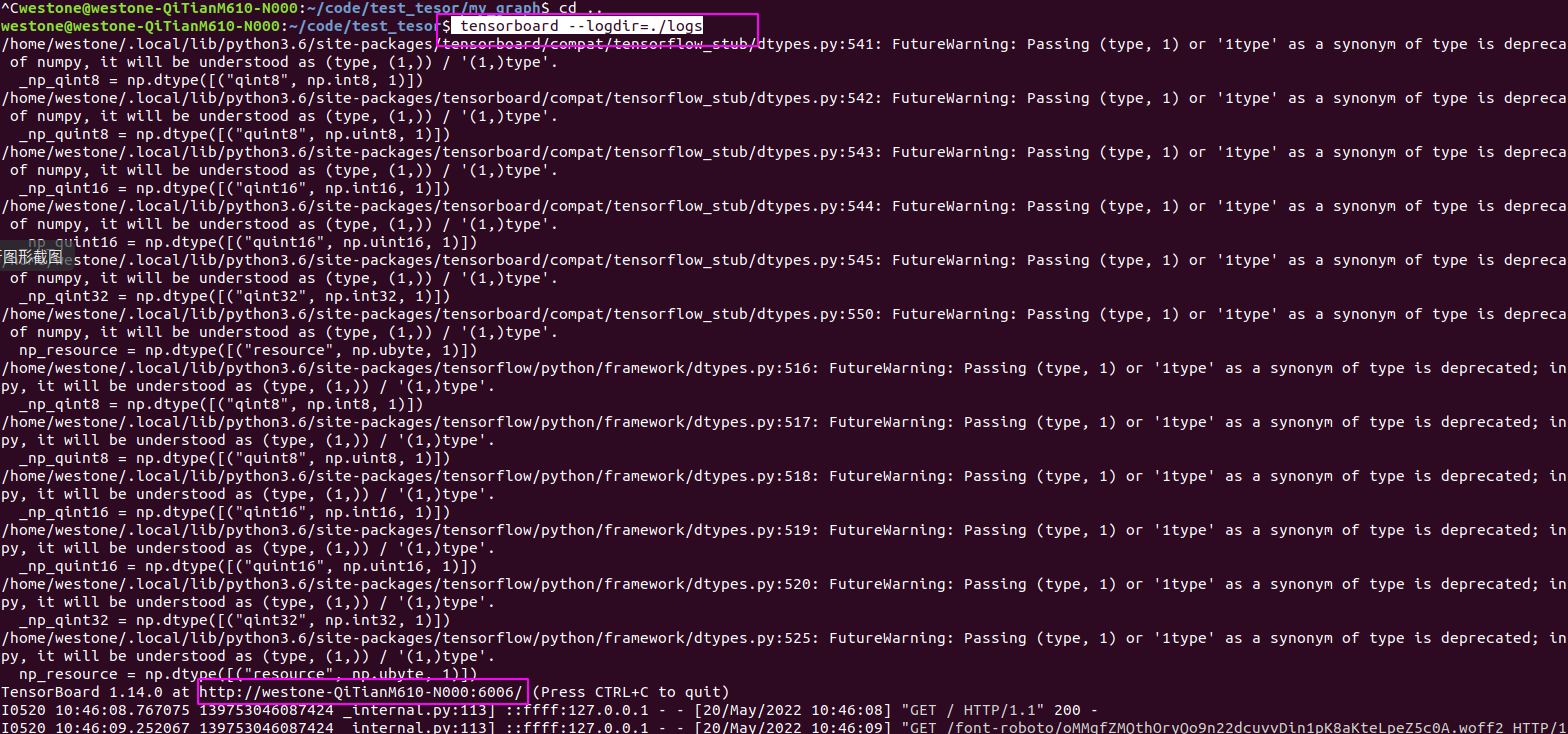
|  |
| --- |
| import tensorflow as tf    # 定义一个计算图，实现两个向量的减法操作  # 定义两个输入，a为常量，b为变量    a=tf.constant([10.0, 20.0, 40.0], name='a')    b=tf.Variable(tf.random\_uniform([3]), name='b')    output=tf.add\_n([a,b], name='add')    # 生成一个具有写权限的日志文件操作对象，将当前命名空间的计算图写进日志中    writer=tf.summary.FileWriter('./logs', tf.get\_default\_graph())    writer.close() |

执行python test.py命令

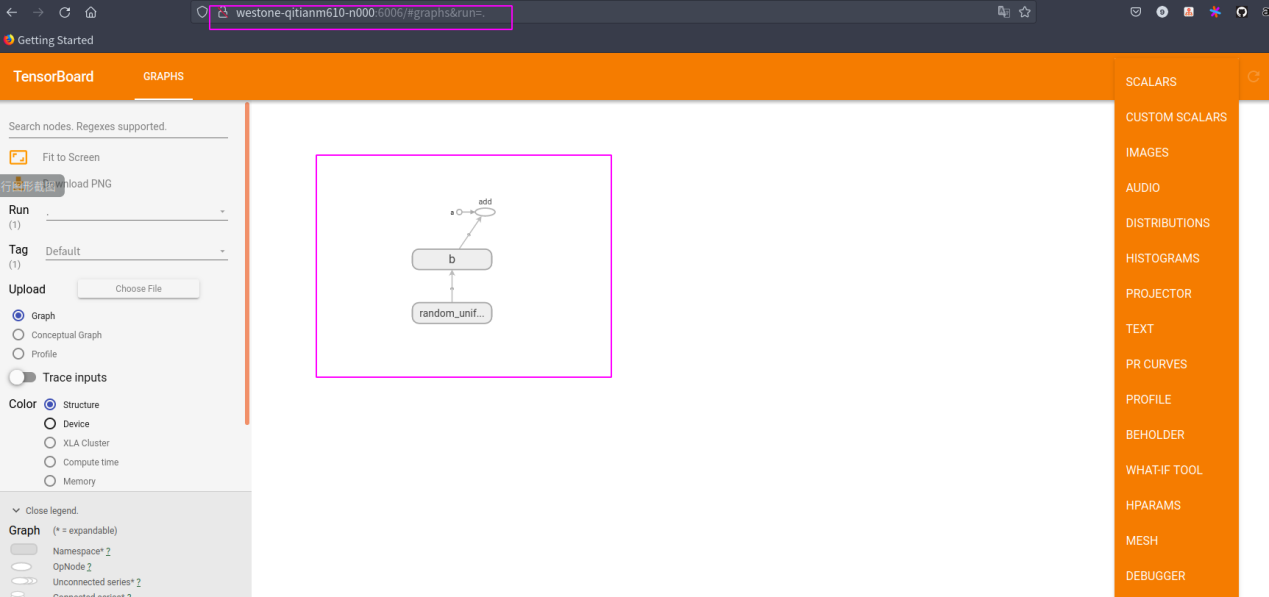
会在test.py文件当前目录下生成文件夹logs

然后在test.py目录下执行如下命令

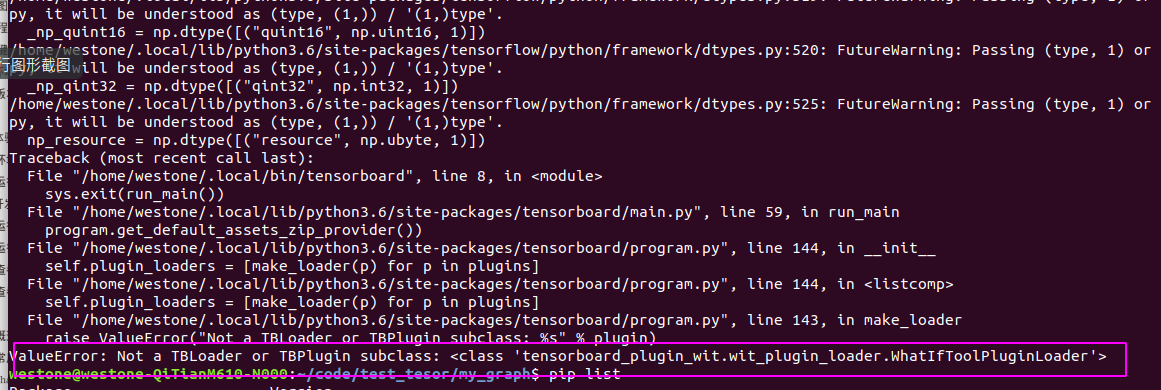
tensorboard --logdir=./logs



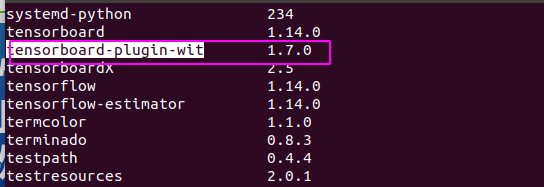
访问如图链接地址



如果报错:



可能原因是因为tensorboard-plugin-wit和tensorboard版本不匹配,此时只需要卸载tensorboard-plugin-wit就行



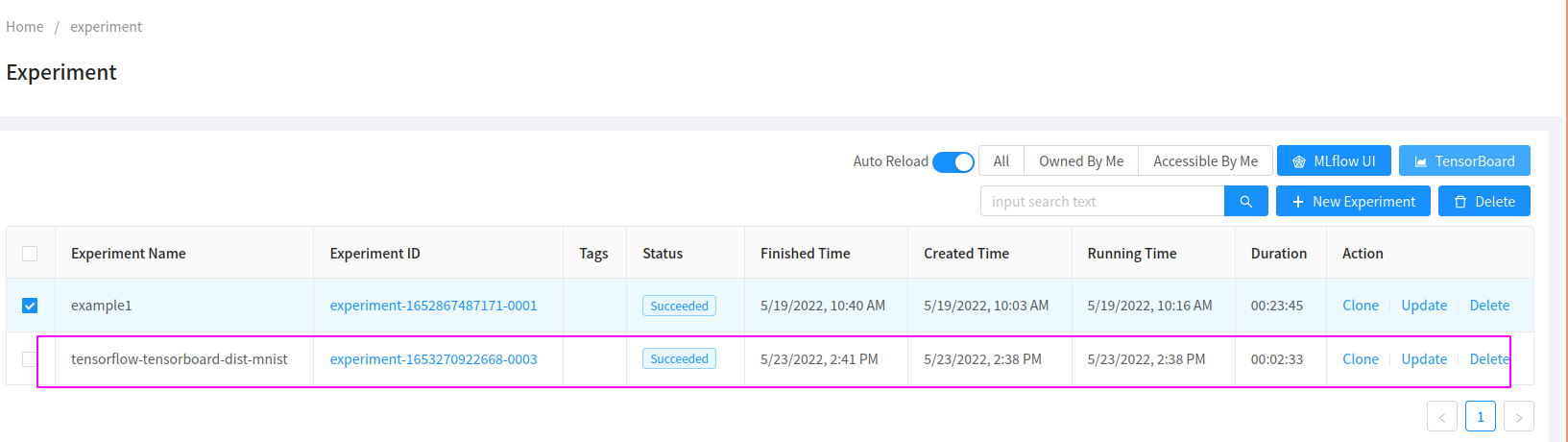
卸载命令: pip uninstall tensorboard-plugin-wit

* **Submarine结合**

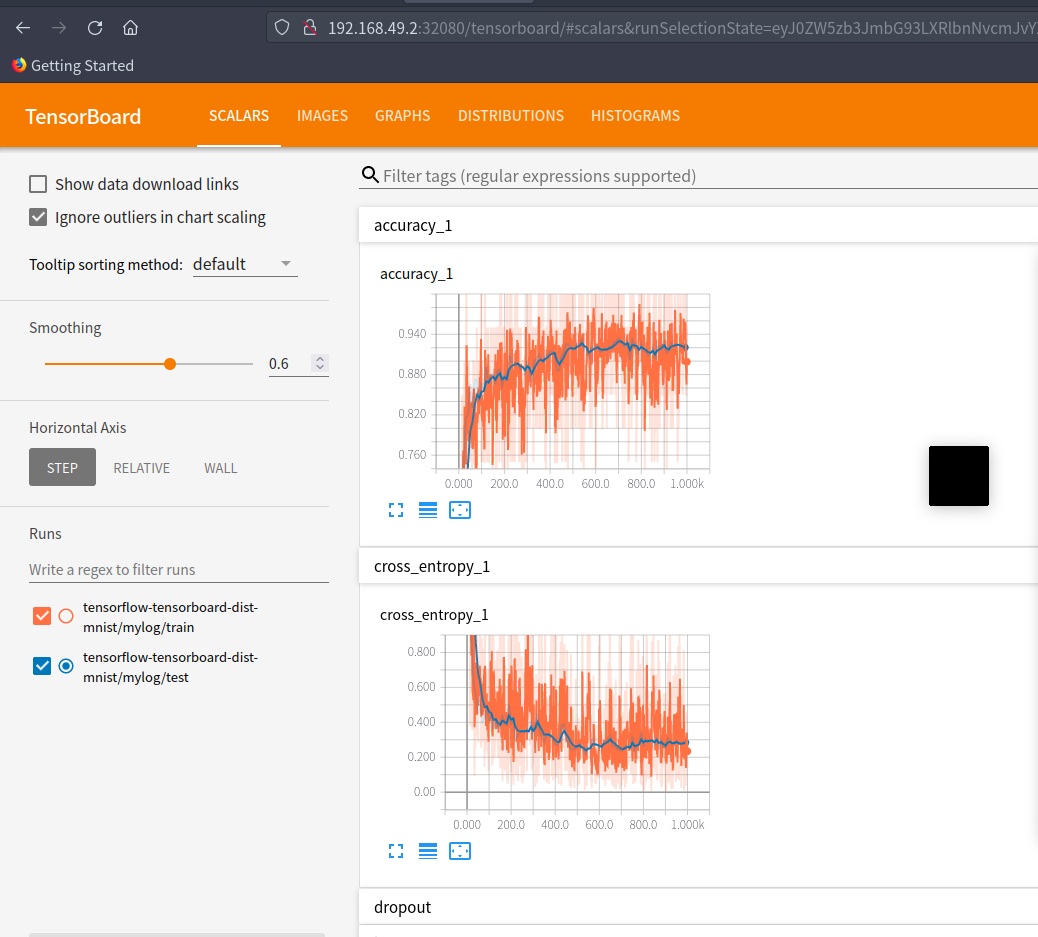
执行如下脚本,新增试验:

|  |
| --- |
| curl -X POST -H "Content-Type: application/json" -d ' {  "meta": {  "name": "tensorflow-tensorboard-dist-mnist",  "namespace": "default",  "framework": "TensorFlow",  "cmd": "python /var/tf\_mnist/mnist\_with\_summaries.py --log\_dir=$(SUBMARINE\_TENSORBOARD\_LOG\_DIR) --learning\_rate=0.01 --batch\_size=20",  "envVars": {  "ENV\_1": "ENV1"  }  },  "environment": {  "image": "apache/submarine:tf-mnist-with-summaries-1.0"  },  "spec": {  "Worker": {  "replicas": 1,  "resources": "cpu=1,memory=512M"  }  } }  ' http://192.168.49.2:32080/api/v1/experiment |

在submarine界面可以看到如下:



等该试验执行完成后,点击tensorboard按钮,得到如下界面



代码参考文件:

submarine/dev-support/examples/quickstart/mnist\_with\_summaries.py

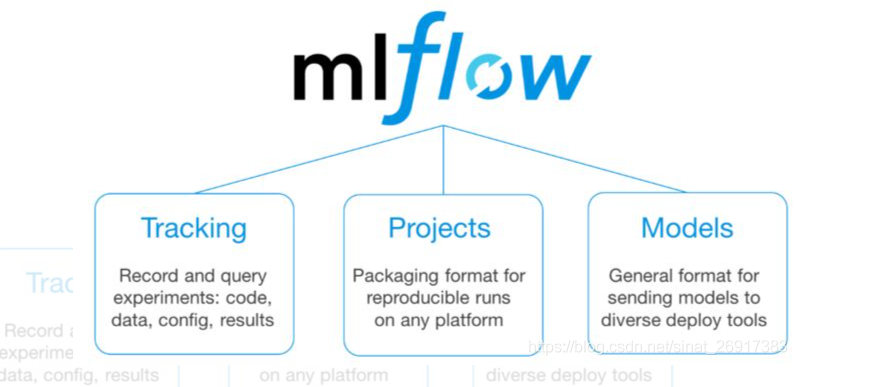
# **MLflow**

* **概述**

开源机器学习平台,一个能够覆盖机器学习全流程（从数据准备到模型训练到最终部署）的新平台。

MLFlow是一款管理机器学习工作流程的工具，核心由以下4个模块组成：

1. MLflow Tracking：如何通过API的形式管理实验的参数、代码、结果，并且通过UI的形式做对比。
2. MLflow Projects：代码打包的一套方案
3. MLflow Models：一套模型部署的方案
4. MLflow Model Registry：一套管理模型和注册模型的方案



Tracking：Tracking就是记录模型运行过程中产生的各项数据，主要有参数、模型指标、持久化的模型等。

Project:项目管理主要解决依赖包及代码运行问题。其实现方式就是通过一些元信息进行项目描述，如下图MLproject文件记录的项目名称，运行环境、参数和运行命令。

Models:模型管理和项目管理类似，会将模型文件（model.pkl）和模型描述信息（MLmodel）打包在同一文件夹下。描述信息会含有模型调用的方式和持久化的模型文件名。

模型的部署:实现的方式就是启动Flask（web server）监听某个端口，接收JSON格式的请求后调用训练好的模型进行预测输出结果。在使用过程中有个细节需要注意，当Flask接收到JSON格式的数据后会使用pandas中的read\_json将其转换为dataframe，但此dataframe的列顺序是按照列名的字典序排列的。

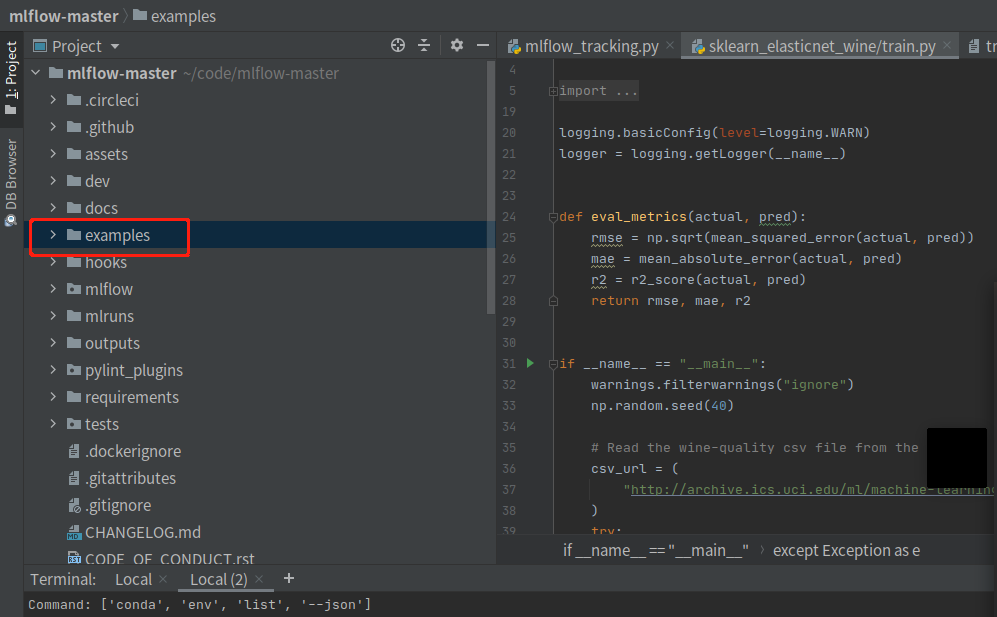
资料:

<https://github.com/mlflow/mlflow>

<https://mlflow.org/docs/latest/index.html>

* **快速开始**

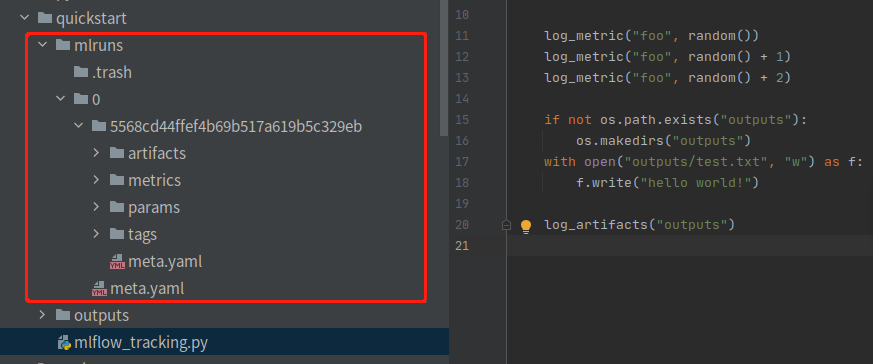
1. 前提条件:安装python(最好3.7及以上)
2. 运行命令:pip install mlflow
3. 运行命令:pip install mlflow-skinny
4. 下载mlflow源码:git clone <https://github.com/mlflow/mlflow.git>
5. 导入pycharm,如下图:



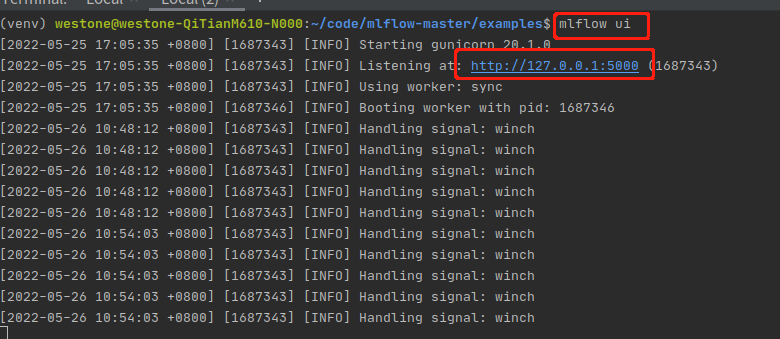
1. 运行例子,example/quickstart目录下的mlflow\_tracking.py

python ./example/quickstart/mlflow\_tracking.py

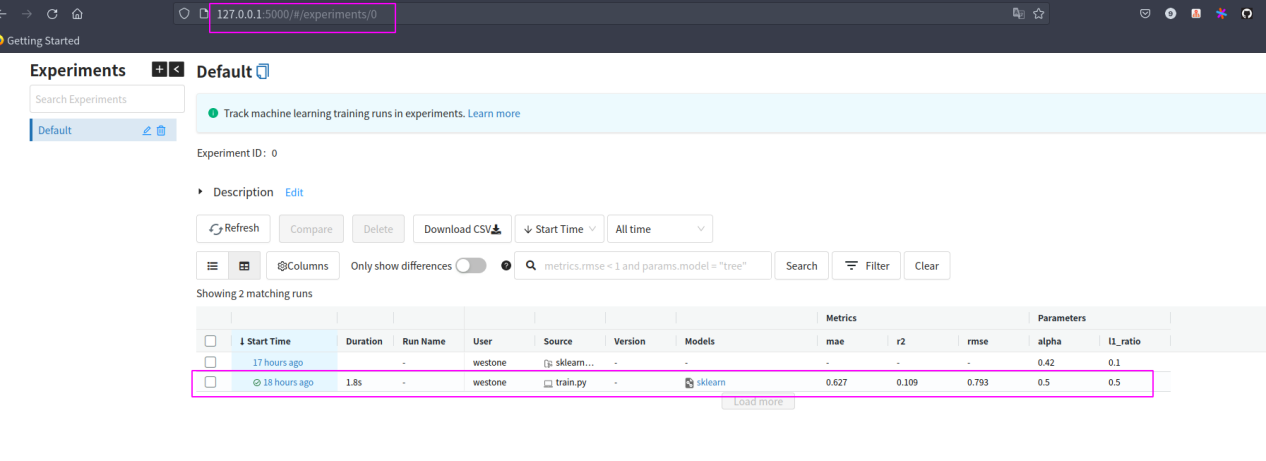
运行该命令后,可以在当前目录下看见mlruns的目录

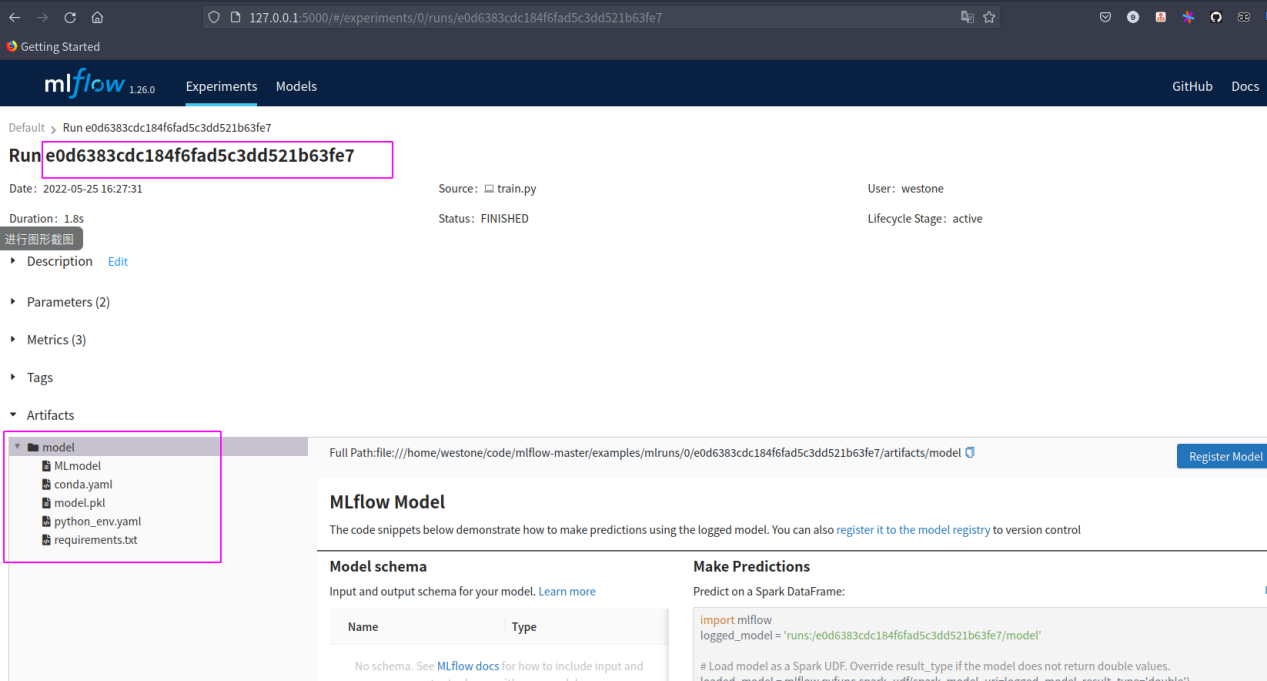


1. 当前目录下运行命令:mlflow ui



访问如上链接



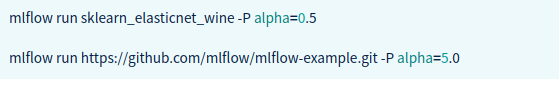


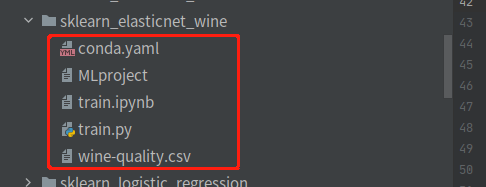
* **功能**

1. 打包工程

除了上面演示的,mlflow允许您将代码及其依赖项打包为一个工程,该工程可以在其他数据上执行相同的过程.每个工程都包含其代码和一个MLproject定义其依赖项（例如 Python 环境）的文件，以及可以在项目中运行哪些命令以及它们采用哪些参数。

例如:





1. 保存模型和服务模型

MLflow 包含一种通用MLmodel格式，用于保存来自不同风格的各种工具的model.许多模型可以用作 Python 函数，因此MLmodel文件可以声明每个模型应如何解释为 Python 函数，以便让各种工具为其提供服务。

例子:

运行如下命令,会返回MLflow ID,并且在查看mlflow UI的时候,能看见程序运行保存下来的model,里面包含MLmodel描述文件等.

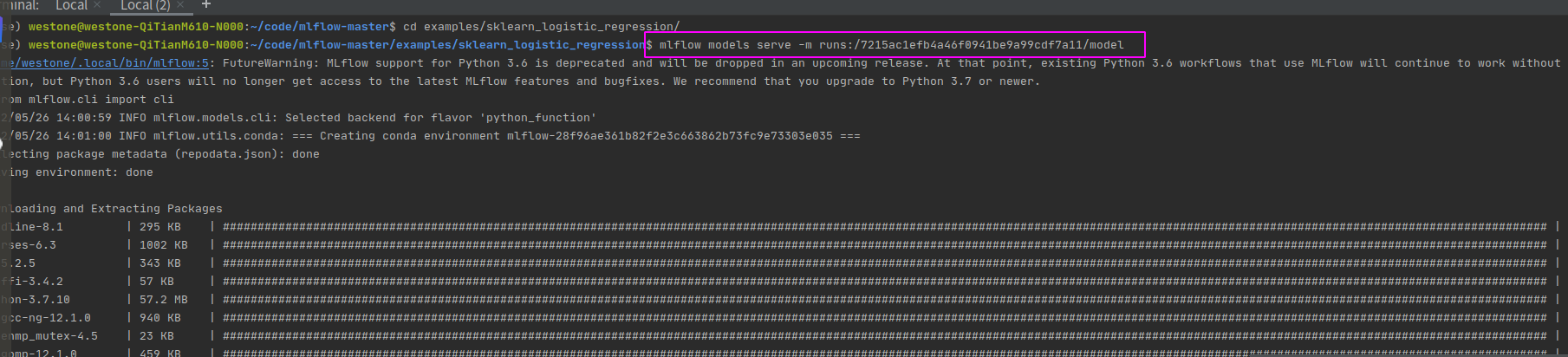
|  |
| --- |
| python sklearn\_logistic\_regression/train.py |

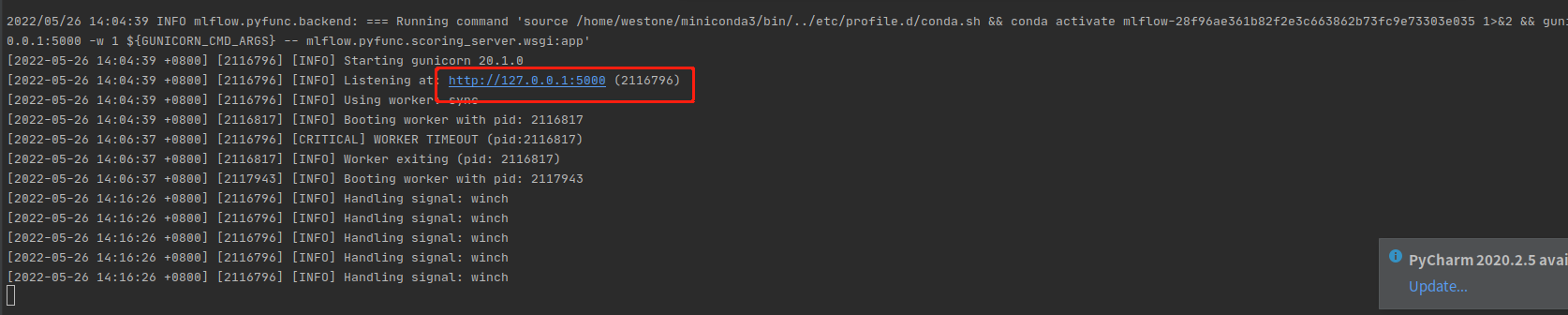


得到该文件后,我们可以运行model,并将一些示例数据传递给它并查看预测结果。

运行命令如下:

mlflow models serve -m runs:/<RUN\_ID>/model

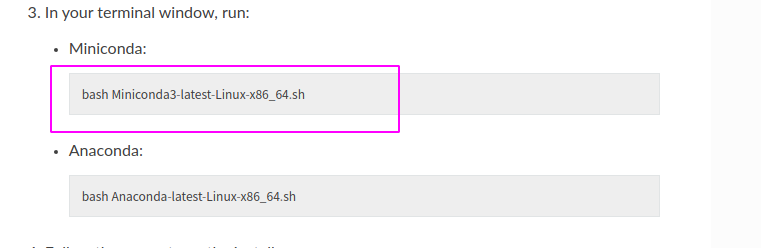




如上图表示运行成功

如果提示conda不兼容,需求去conda官网重新安装conda,不能通过pip安装

<https://conda.io/projects/conda/en/latest/user-guide/install/linux.html>



如上一切顺利后,可以调用模型服务

curl -d '{"columns":["x"], "data":[[1], [-1]]}' -H 'Content-Type: application/json; format=pandas-split' -X POST localhost:5000/invocations

返回结果:

[1, 0]

1. MLflow Tracking

MLflow 跟踪组件是一个 API 和 UI，用于在运行机器学习代码时记录参数、代码版本、指标和输出文件，并在以后可视化结果。MLflow Tracking 允许您使用Python、REST、R API和Java API API 记录和查询实验。

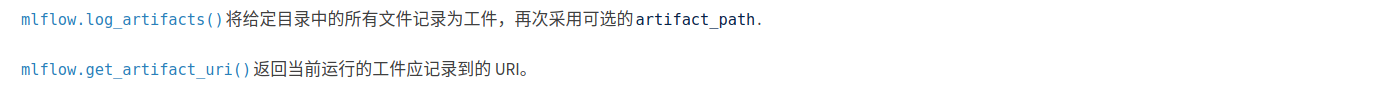
MLflow runs可以记录到本地文件、SQLAlchemy 兼容数据库或远程跟踪服务器。默认情况下，MLflow Python API 日志在本地运行到mlruns您运行程序的目录中的文件。然后，您可以运行mlflow ui以查看记录。

要远程记录运行，请将MLFLOW\_TRACKING\_URI环境变量设置为跟踪服务器的 URI 或调用mlflow.set\_tracking\_uri().

MLflow runs和artifacts如何被记录?

MLflow runs可以记录到本地文件、SQLAlchemy 兼容数据库或远程跟踪服务器。MLflow artifacts可以持久化到本地文件和各种远程文件存储解决方案。为了存储runs和artifacts，MLflow 使用两个组件进行存储：后端存储和artifacts存储。后端存储保存 MLflow 实体（运行、参数、指标、标签、注释、元数据等），而artifacts存储保存工件（文件、模型、图像、内存对象或模型摘要等）。



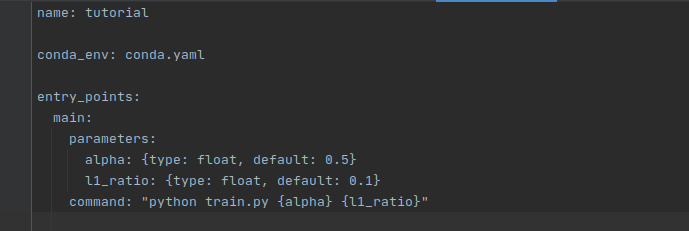


1. Mlflow project

MLflow project是一种以可重用和可重现的方式打包数据科学代码的格式，主要基于约定。此外，Projects 组件包括用于运行项目的 API 和命令行工具，从而可以将project链接到工作流中。

从本质上讲，MLflow 项目只是组织和描述代码以让其他数据科学家（或自动化工具）运行它的约定。每个项目只是一个文件目录或一个 Git 存储库，其中包含您的代码。MLflow 可以根据在此目录中放置文件的约定运行一些项目，但您可以通过添加MLproject文件来更详细地描述您的项目，该文件是YAML格式的文本文件.

MLproject文件实例:



# **Operator**

* **概念**

Operator 概念是由 CoreOS 的工程师于2016年提出的，它让工程师可以根据应用独有的领域逻辑为应用编写自定义的控制器。这句话说的有点虚空，我们通过一个简单的例子理解一下 Operator 。

假设有一个连接数据库的 Java Web程序。你想将其部署到您的k8s集群。理想情况下，你会希望用 Deployment 部署应用然后暴露给 Service，对于应用服务的后端则是使用 StatuflSet 部署数据库。所以需要完成两部分的部署才能把整个应用服务部署完成：

* 无状态部分：Java Web 应用。
* 有状态部分：数据库。

在上面的例子中，我们可以应用我们对应用程序与数据库之间的关系的了解，创建一个控制器，该控制器将以某种特定方式运行时执行某些操作。比如备份、更新、数据还原这些任务该如何完成取决于应用程序本身和业务限制（领域知识）。这些与应用强相关的操作就是 Kubernetes Operator要做的：代替原本需要由SRE（Site Reliability Engineers）和运维工程师来完成操作的执行。

Operator 模型基于 Kubernetes 中的两个概念结合而成：自定义资源和自定义控制器。

**自定义资源:**

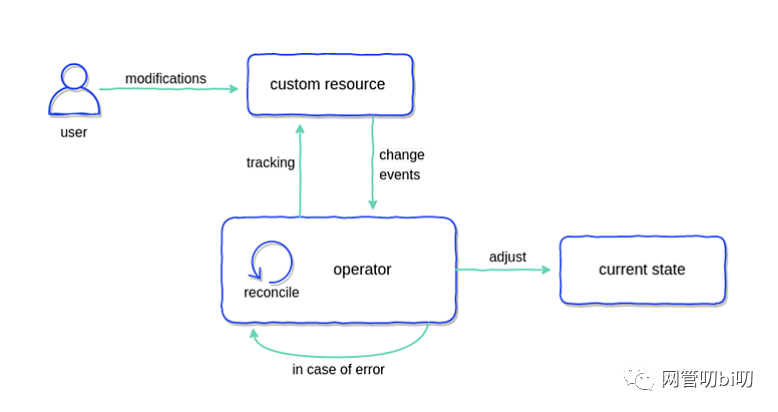
在 Kubernetes中，资源是 Kubernetes API中的一个端点，用于存储一堆特定类型的API对象。它允许我们通过向集群添加更多种类的对象来扩展Kubernetes。添加新种类的对象之后，我们可以像其他任何内置对象一样，使用 kubectl 来访问我们自定义的 API 对象。

以 Pod 或 Deployment 为例。编写清单时，必须在 YAML 文件中指定一种类型（是 Pod 还是 Deployment）。自定义资源就是不由 Kubernetes 原生提供的资源对象。

**自定义控制器:**

Kubernetes 的所有控制器，都有一个控制循环，负责监控集群中特定资源的更改，并确保特定资源在集群里的当前状态与控制器自身定义的期望状态保持一致。

举例来说有一个 Deployment 控制器管控值集群里的一组 Pod ，当你 Kill 掉一个 Pod 。控制器发现定义中期望的Pod数量与当前的数量不匹配，它就会马上创建一个 Pod 让当前状态与期望状态匹配。控制器这种让关联资源的当前状态向期望状态迈进的过程叫做调谐（reconcile）。



# **要求**

1. experiment任务要随时能查看服务状态,可以启动,暂停,恢复,删除
2. experiment可以查看日志
3. 查看实时指标:可以,需要改前端