构建无服务器架构的机器学习流水线

——训练自动驾驶的玩具车

张江山， AWS 解决方案架构师

AWS Robocar Rally 机器人车拉力赛

AWS Robocar Rally 机器人车拉力赛

AWS Robocar Rally 机器人车拉力赛

AWS Machine Learning Blog

Build an Autonomous Vehicle on AWS and Race It at the re:Invent Robocar Rally

by Justin De Castri and Sunil Mallya | on 28 SEP 2017 | in Apache MXNet On AWS, AWS Deep Learning AMIs | Permalink | Comments | Share

Autonomous vehicles are poised to take to our roads in massive numbers in the coming years. This has been made possible due to advances in deep learning and its application to autonomous driving. In this post, we take you through a tutorial that shows you how to build a remote control (RC) vehicle that uses Amazon AI services.

---

1) **Build an Autonomous Vehicle on AWS and Race It at the re:Invent Robocar Rally**
2) Build an Autonomous Vehicle Part 2: Driving Your Vehicle
3) Building an Autonomous Vehicle Part 3: Connecting Your Autonomous Vehicle
4) Building an Autonomous Vehicle Part 4: Using Behavioral Cloning with Apache MXNet for Your Self-Driving Car

---

Typically each autonomous vehicle is stacked with a lot of sensors that provide rich telemetry. This telemetry can be used to improve the driving of the individual vehicle but also the user experience. Some examples of those improvements are time saved by smart drive routing, increased vehicle range and efficiency, and increased safety and crash reporting. On AWS, customers like TuSimple have built a sophisticated autonomous platform using Apache MXNet. Recently TuSimple completed a 200-mile driverless ride.

To drive awareness of deep learning, AWS IoT, and the role of artificial intelligence (AI) in autonomous driving, AWS will host a workshop-style hackathon—Robocar Rally at re:Invent 2017. This is the first in a series of blog posts and Twitch videos for developers to learn autonomous AI technologies and to prepare for the hackathon. For more details on the hackathon, see Robocar Rally 2017.

机器人车组装过程

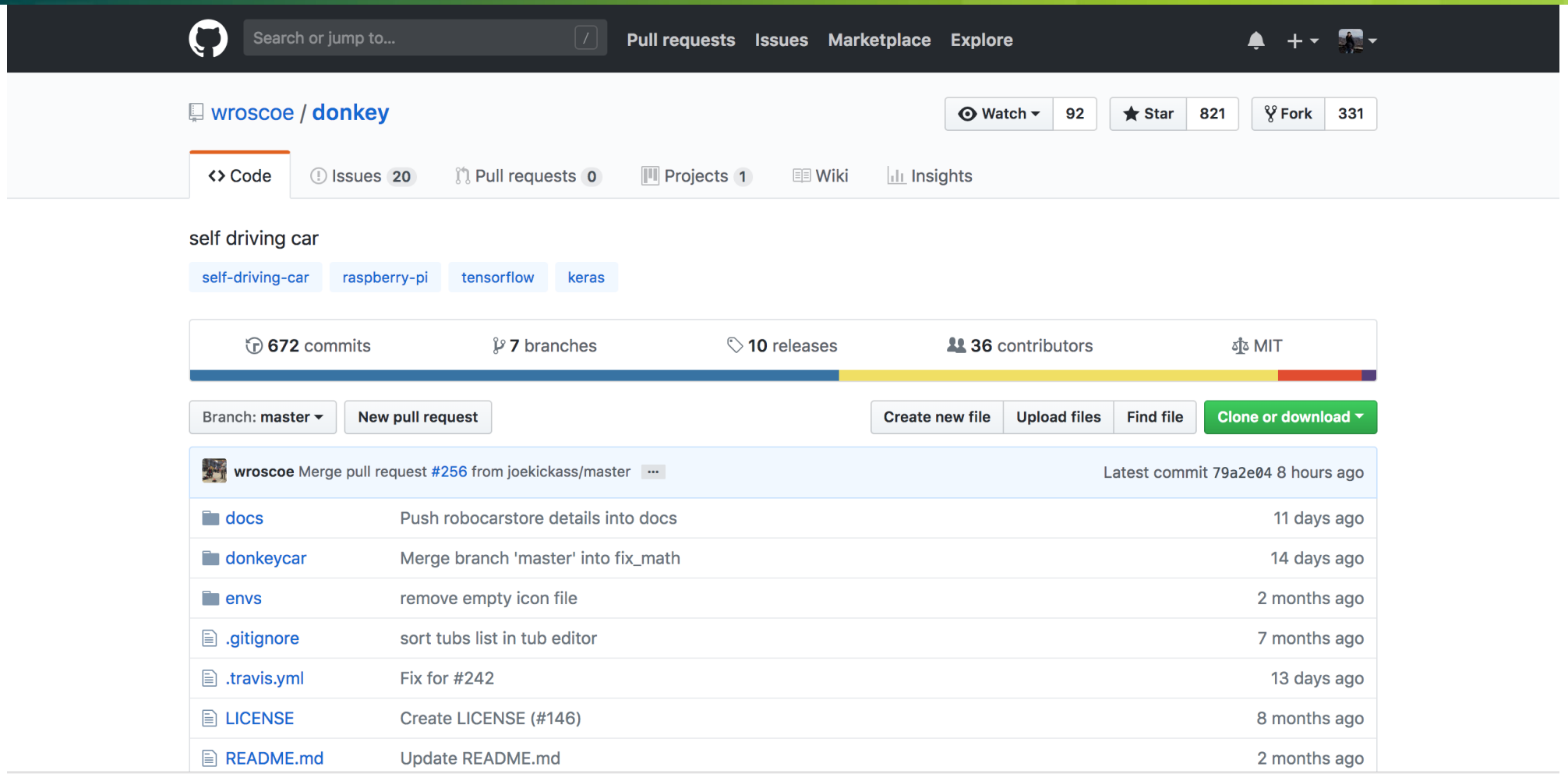
1. 硬件组装

2. Raspberry Pi 树莓派配置

3. 车辆转速和转向的控制参数校准

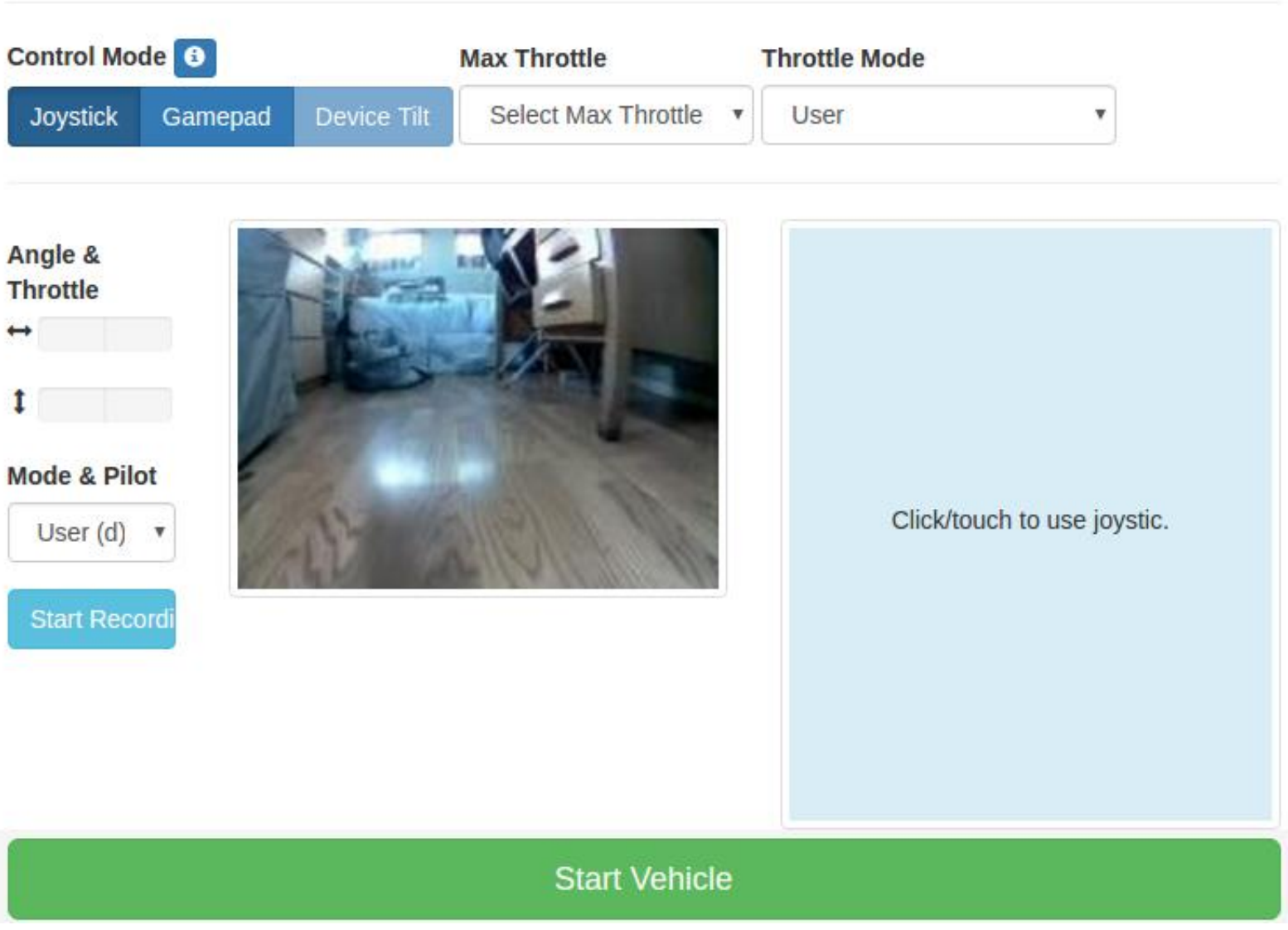
4. 手工驾驶对车辆的控制进行测试

Donkey Server



https://github.com/wroscoe/donkey

Donkey Server



数据采集


```
{"user/mode": "user", "cam/image_array": "9870_cam-image_array_.jpg", "user/throttle": 0.40, "user/angle": 0.9},
{"user/mode": "user", "cam/image_array": "9871_cam-image_array_.jpg", "user/throttle": 0.40, "user/angle": 0.9},
{"user/mode": "user", "cam/image_array": "9872_cam-image_array_.jpg", "user/throttle": 0.35, "user/angle": 0.8},
{"user/mode": "user", "cam/image_array": "9873_cam-image_array_.jpg", "user/throttle": 0.35, "user/angle": 0.8},
{"user/mode": "user", "cam/image_array": "9874_cam-image_array_.jpg", "user/throttle": 0.35, "user/angle": 0.9},
{"user/mode": "user", "cam/image_array": "9875_cam-image_array_.jpg", "user/throttle": 0.35, "user/angle": 0.9},
{"user/mode": "user", "cam/image_array": "9876_cam-image_array_.jpg", "user/throttle": 0.35, "user/angle": 0.8},
{"user/mode": "user", "cam/image_array": "9877_cam-image_array_.jpg", "user/throttle": 0.35, "user/angle": 0.8},
{"user/mode": "user", "cam/image_array": "9878_cam-image_array_.jpg", "user/throttle": 0.35, "user/angle": 0.8},
{"user/mode": "user", "cam/image_array": "9879_cam-image_array_.jpg", "user/throttle": 0.30, "user/angle": 0.7},
```

训练过程

1. 采集的数据存储在机器人车的树莓派

2. 将数据传送到 Amazon EC2

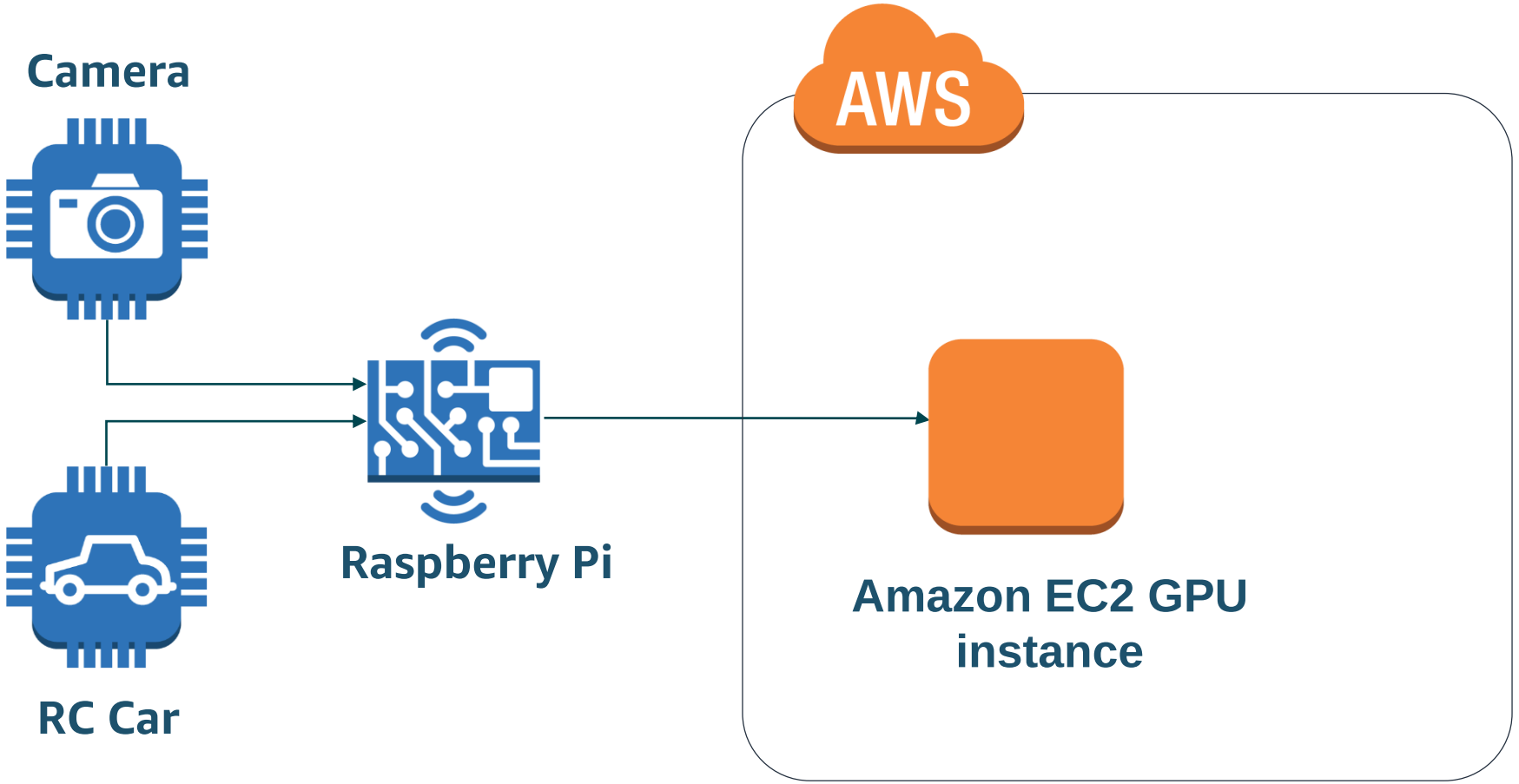
3. 训练模型
   python ~/d2/manage.py train --model ~/d2/models/mypilot

4. 将模型拷贝到树莓派

训练模型—CNN 的网络定义

```
def default_categorical():
    img_in = Input(shape=(120, 160, 3), name='img_in')                        # First layer, input layer, Shape comes from camera.py resolution, RGB
    x = img_in
    x = Convolution2D(24, (5,5), strides=(2,2), activation='relu')(x)       # 24 features, 5x5 kernel(convolution, feauture)window, 2wx2h stride, relu
…
    x = Convolution2D(64, (3,3), strides=(1,1), activation='relu')(x)       # 64 features, 3px3p kernal window, 1wx1h stride, relu
   # Possibly add MaxPooling (will make it less sensitive to position in image).  Camera angle fixed, so may not to be needed
   x = Flatten(name='flattened')(x)                                       # Flatten to 1D (Fully connected)
    x = Dense(100, activation='relu')(x)                                    # Classify the data into 100 features, make all negatives 0
    x = Dropout(.1)(x)                                                      # Randomly drop out (turn off) 10% of the neurons (Prevent overfitting)
    x = Dense(50, activation='relu')(x)                                     # Classify the data into 50 features, make all negatives 0
    x = Dropout(.1)(x)                                                      # Randomly drop out 10% of the neurons (Prevent overfitting)
    #categorical output of the angle
    angle_out = Dense(15, activation='softmax', name='angle_out')(x)        # Connect every input with every output and output 15 hidden units. Use Softmax
to give percentage. 15 categories and find best one based off percentage 0.0-1.0
    #continous output of throttle
    throttle_out = Dense(1, activation='relu', name='throttle_out')(x)      # Reduce to 1 number, Positive number only
    model = Model(inputs=[img_in], outputs=[angle_out, throttle_out])
    model.compile(optimizer='adam',  loss={'angle_out': 'categorical_crossentropy','throttle_out': 'mean_absolute_error'},loss_weights={'angle_out': 0.9,
'throttle_out': .001})
return model
```

架构



架构优化?

数据采集

机器学习流水线

...

训练过程回顾

1. 采集的数据存储在机器人车的树莓派

2. 将数据传送到 Amazon EC2

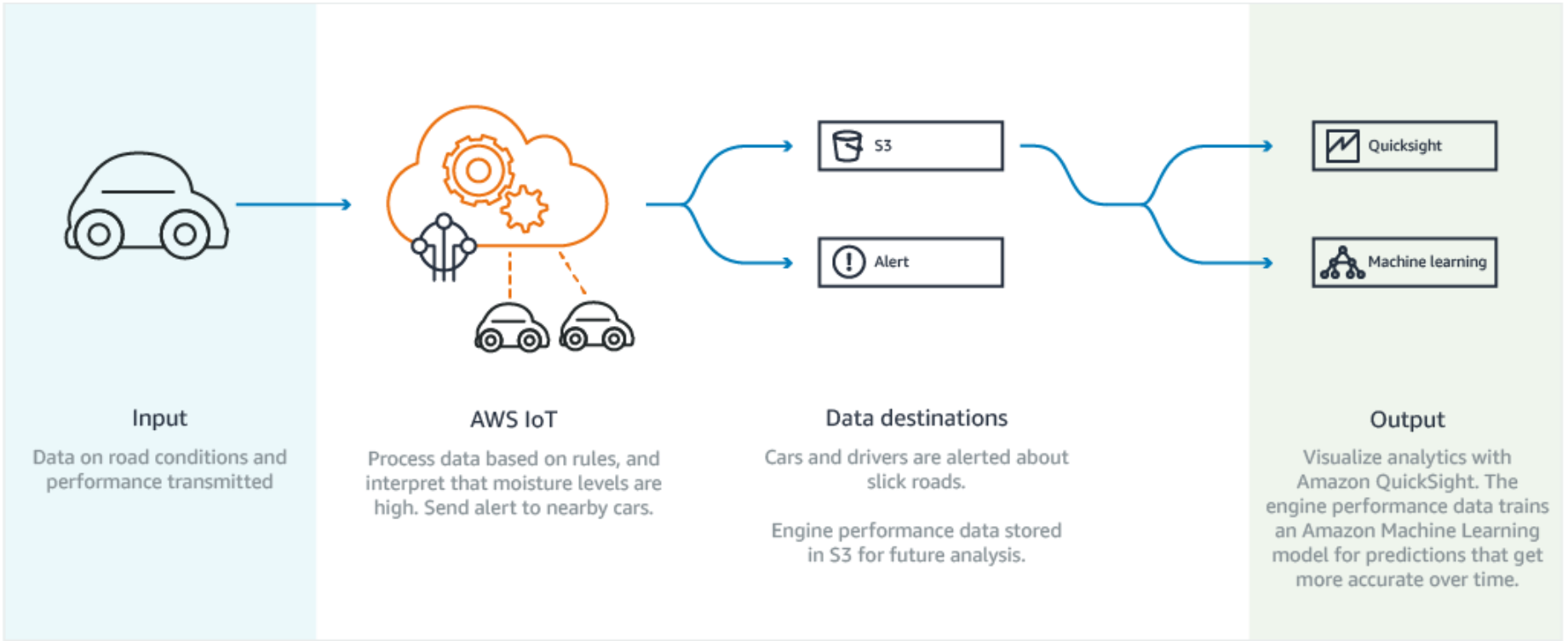
3. 训练模型

   python ~/d2/manage.py train --model ~/d2/models/mypilot

4. 将模型拷贝到树莓派

自动化的数据采集?

利用 AWS 物联网平台采集数据



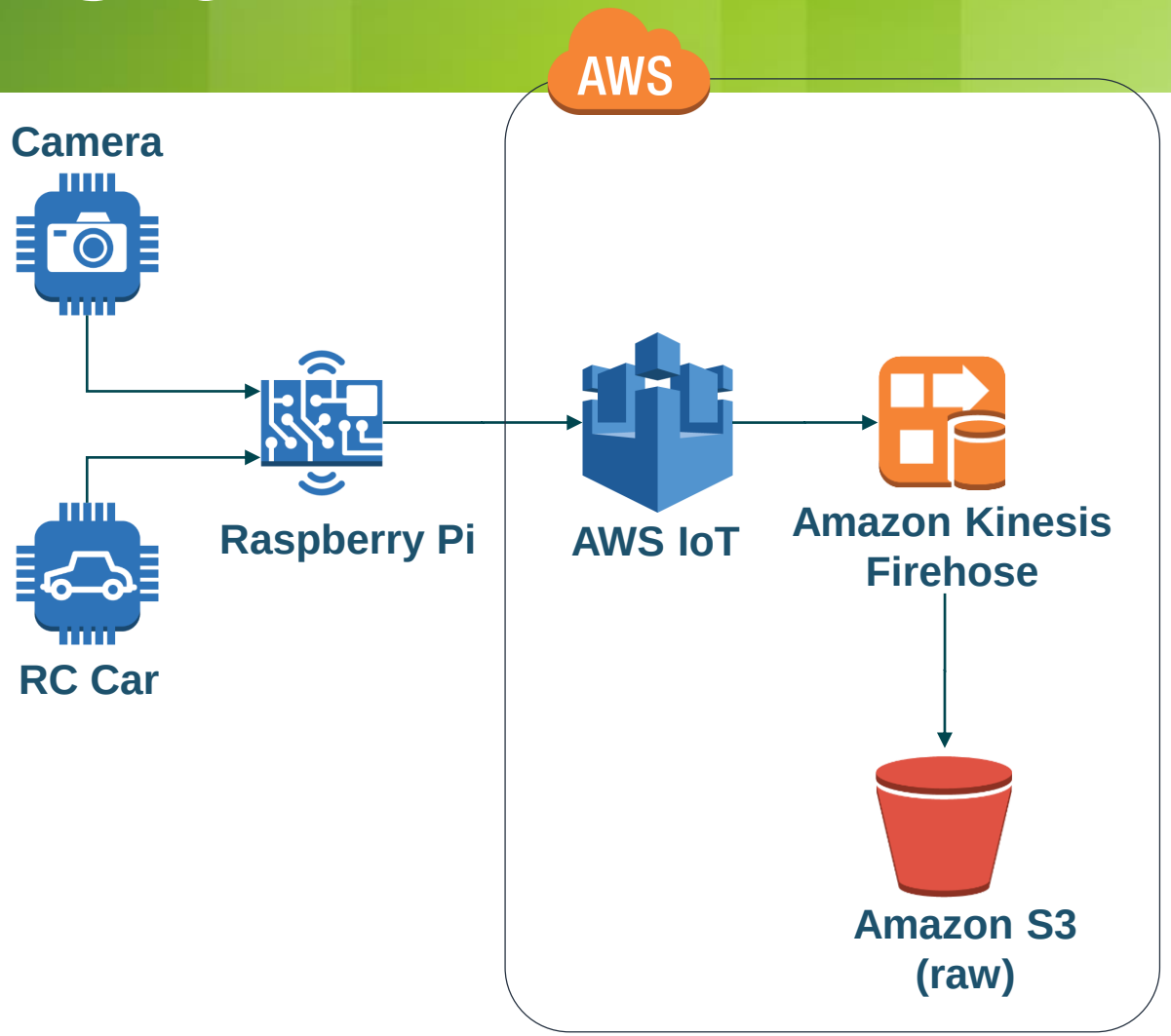
式例代码

```
def run(self, *args):
    '''
    API function needed to use as a Donkey part.

    Accepts values, pairs them with their inputs keys and saves them
    to disk.
    '''
    assert len(self.inputs) == len(args)

    self.record_time = int(datetime.now().timestamp() - self.start_time)
    record = dict(zip(self.inputs, args))
    self.put_record(record)
```

https://github.com/chankh/donkey/blob/master/donkeycar/parts/iot.py

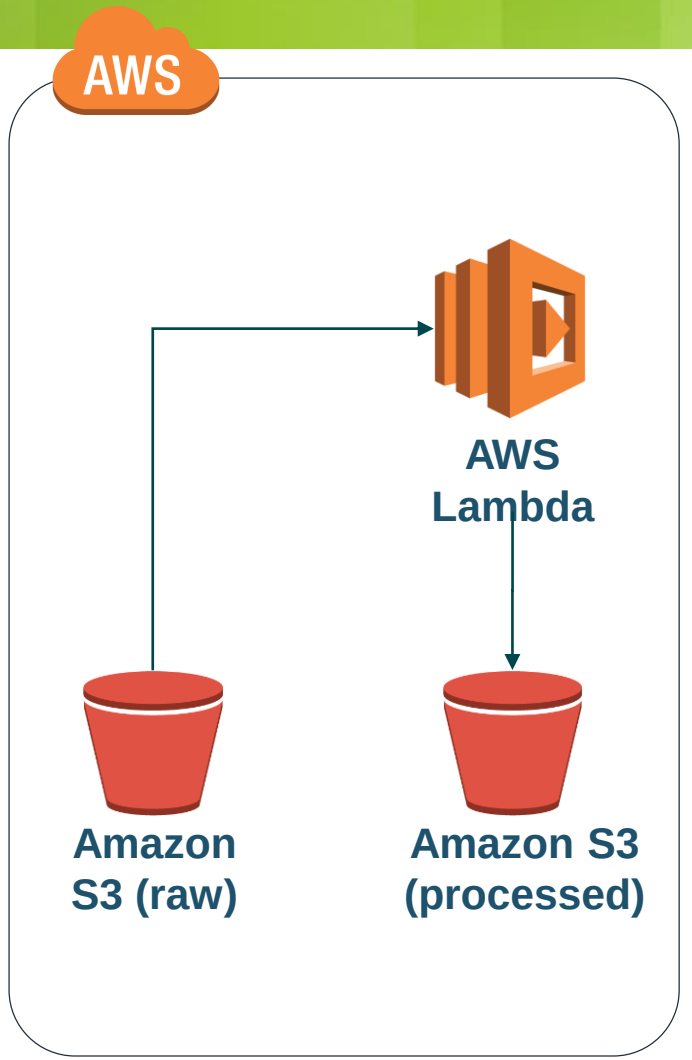
Stream record data using AWS IoT


```
{
  "current_ix":1525,
  "cam/image_array":"1525_cam-image_array_.jpg",
  "user/angle":0.7,
  "vehicleID":"demo",
  "time":"2018-05-10T09:15:38.338987",
  "user/mode":"user",
  "user/throttle":0.35,
  "image":<base64 encoded>
}
```

Process data using AWS Lambda


```
{
    "current_ix":1525,
    "cam/image_array":"1525_cam-image_array_.jpg",
    "user/angle":0.7,
    "vehicleID":"demo",
    "time":"2018-05-10T09:15:38.338987",
    "user/mode":"user",
    "user/throttle":0.35,
    "image":<base64 encoded>
}
```

训练过程回顾

1. ~~采集的数据存储在机器人车的树莓派~~

2. ~~将数据传送到 Amazon EC2~~

3. 训练模型

   python ~/d2/manage.py train --model ~/d2/models/mypilot

4. 将模型拷贝到树莓派

Amazon SageMaker

A **fully managed service** that enables **data scientists** and **developers** to quickly and easily **build** machine-learning based models **into production** smart applications.

什么是 Amazon SageMaker?

预配置的 notebook 实例

Build

优化的机器学习算法

One-click training for ML, DL, and custom algorithms

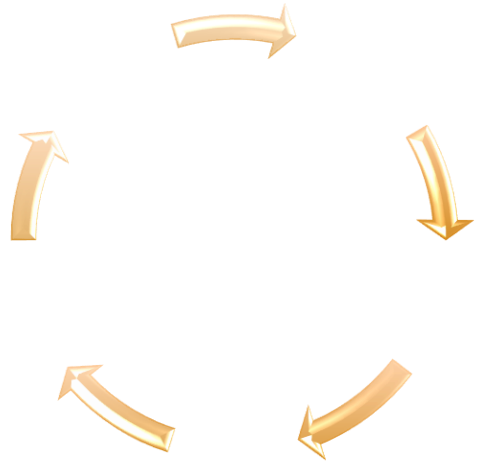
Train

Easier training with hyperparameter optimization

简便的部署方式

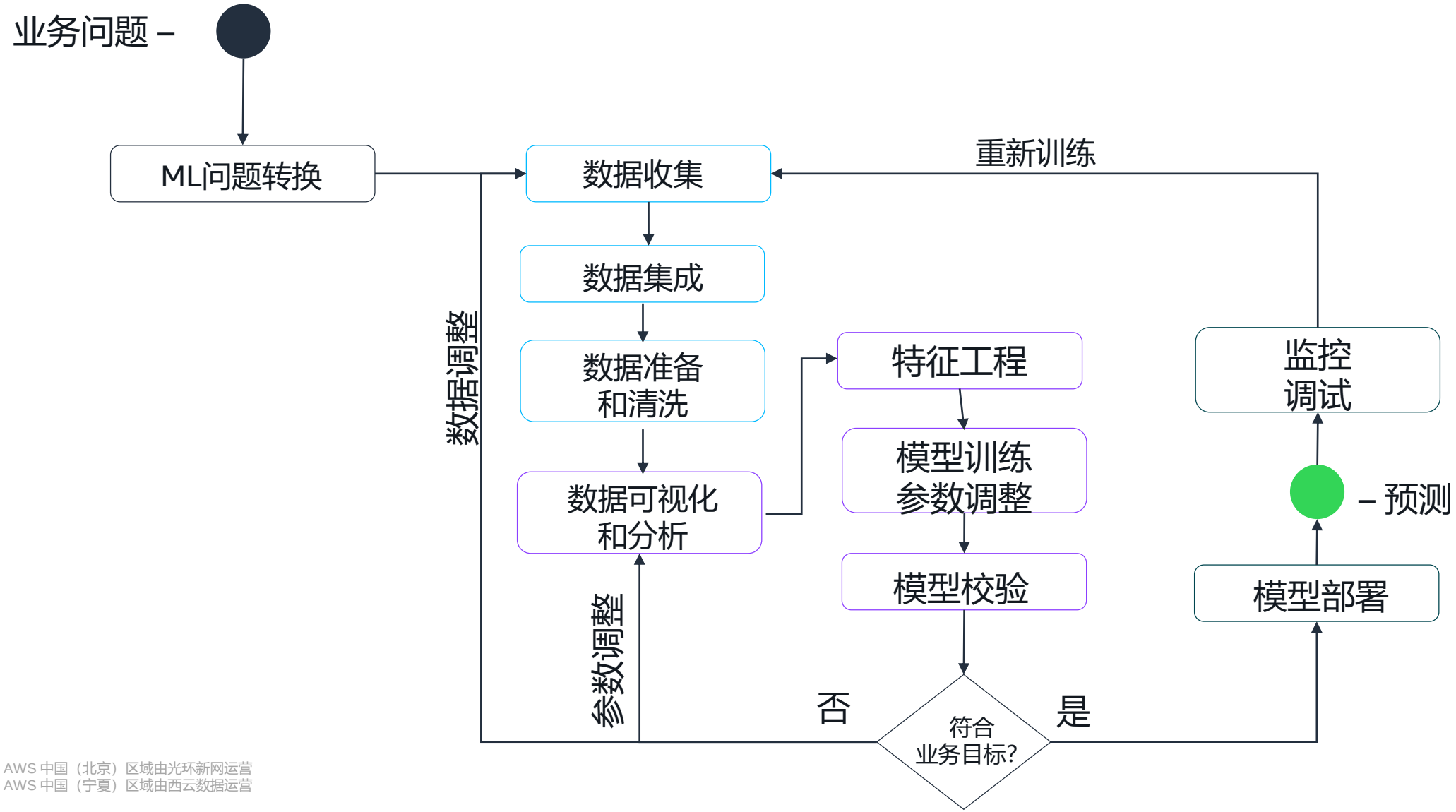
Deploy

完全托管的可伸缩部署



mxnet GLUON TensorFlow PYTORCH aws

机器学习流程



使用 Amazon SageMaker 训练

SageMaker 自带算法

K-means Clustering
PCA
Neural Topic Modelling
Factorisation Machines
Linear Learner – Regression
XGBoost
Latent Dirichlet Allocation
Image Classification
Seq2Seq
Linear Learner – Classification
DeepAR Forecasting

Bring Your Own Algorithms

ML Algorithms
R
MXNet
TensorFlow
Caffe
PyTorch
Keras
CNTK
…

MXNet & TensorFlow SDK

TensorFlow SDK
MXNet (Gluon) SDK

Apache Spark Estimator

Apache Spark Python library
Apache Spark Scala library

Amazon EMR

使用自带的算法来构建模型

- Amazon SageMaker 的算法程序打包封装在 Docker 镜像
- 在模型训练时，Amazon SageMaker 执行下面的命令:

```
docker run image train
```

- 在训练的时候，/opt/ml 和下面的子目录都被 Amazon SageMaker 占用

使用自带的算法来构建模型

- Training Data
  - /opt/ml/input/data/training
  - /opt/ml/input/data/validation
  - /opt/ml/input/data/testing

- Training Output
  - /opt/ml/output/failure
  - /opt/ml/model

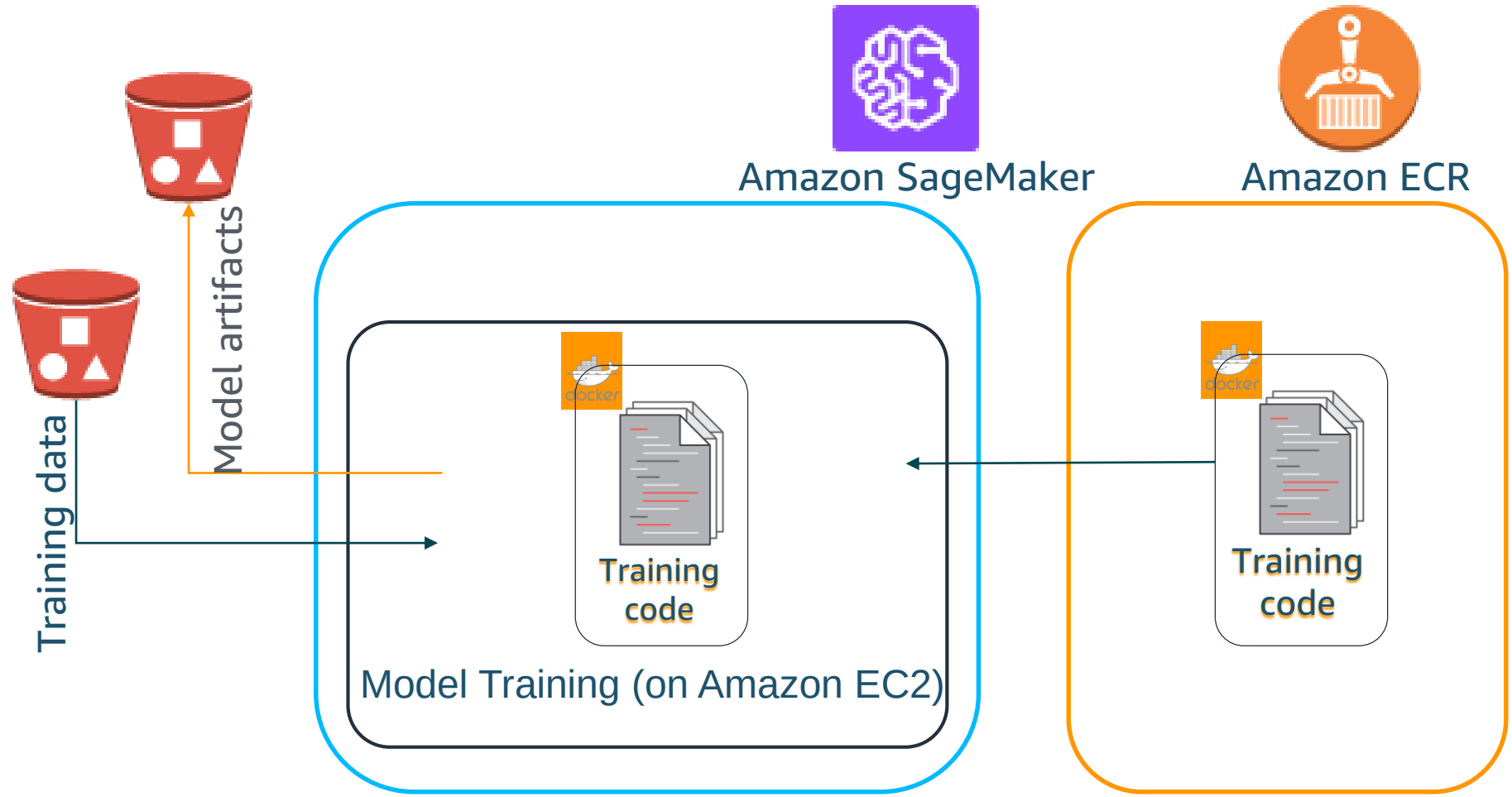
创建自定义的 Docker 镜像

```
FROM ubuntu:16.04
MAINTAINER KH Chan <khchan@amazon.com>
RUN apt-get -y update && apt-get install -y --no-install-recommends wget python virtualenv build-essential python3-dev
python3-pip gfortran libhdf5-dev ca-certificates \
    && rm -rf /var/lib/apt/lists/*

ENV PATH="/opt/program:${PATH}"
COPY donkeycar /opt/program
COPY train /opt/program
WORKDIR /opt/program
RUN virtualenv env -p python3 \
    && . env/bin/activate \
    && pip3 install keras==2.0.8 \
    && pip3 install tensorflow==1.3.0 \
    && pip3 install AWSIoTPythonSDK \
    && pip3 install -e . \
    && donkey createcar --path /opt/program/d2 \
    && chmod +x /opt/program/train
```

创建自定义的 Docker 镜像

```
#!/bin/bash
source /opt/program/env/bin/activate
python /opt/program/d2/manage.py train \
  --tub=/opt/ml/input/data/training/* \
  --model=/opt/ml/model/robocar
```

创建自定义的 Docker 镜像



在 Notebook 中测试

Training DonkeyCar model with SageMaker

The **SageMaker Python SDK** makes it easy to train and deploy ML models. In this notebook we train a model from data collected from the robocar.

Setup the environment

First we need to define a few variables that will be needed later. Here we specify a bucket to use and the role that will be used for working with SageMaker.

In [5]:
```
from sagemaker import get_execution_role

#Bucket with input data
data_location = 's3://autonomous-vehicles-data'

#IAM execution role that gives SageMaker access to resources in your AWS account.
#We can use the SageMaker Python SDK to get the role from our notebook environment.
role = get_execution_role()
```

Create the session

The session remembers our connection parameters to SageMaker. We'll use it to perform all of our SageMaker operations.

In [6]:
```
import sagemaker as sage
from time import gmtime, strftime

sess = sage.Session()
```

在 Notebook 中测试

Create an estimator and fit the model

In order to use SageMaker to fit our algorithm, we'll create an Estimator that defines how to use the container to train. This includes the configuration we need to invoke SageMaker training:

- The container name. This is constructed as in the shell commands above.
- The role. As defined above.
- The instance count which is the number of machines to use for training.
- The instance type which is the type of machine to use for training.
- The output path determines where the model artifact will be written.
- The session is the SageMaker session object that we defined above. Then we use fit() on the estimator to train against the data that we uploaded above.

In [13]:

```
account = sess.boto_session.client('sts').get_caller_identity()['Account']
region = sess.boto_session.region_name
image = '{}.dkr.ecr.{}.amazonaws.com/donkey:latest'.format(account, region)

tree = sage.estimator.Estimator(image,
                       role, 1, 'ml.c5.2xlarge',
                       output_path="s3://{}/output".format(sess.default_bucket()),
                       sagemaker_session=sess)

tree.fit(data_location)
```

```
#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#
010#010#010#010#015412/455 [==========================>...] - ETA: 31s - loss: 0.1404 - angle_out_loss: 0.1559 - thro
ttle_out_loss: 0.0383#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010
#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#
010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#0
10#010#010#010#010#010
/opt/program/env/lib/python3.5/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of
 issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(flo
at).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#
010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#0
10#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#01
0#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010#010
#010#015455/455 [==============================] - 364s - loss: 0.1405 - angle_out_loss: 0.1561 - throttle_out_loss:
 0.0373 - val_loss: 0.0332 - val_angle_out_loss: 0.0369 - val_throttle_out_loss: 0.0250
Epoch 00013: early stopping
===== Job Complete =====
Billable seconds: 5981
```


aws

在 Notebook 中测试

Download the model and deploy to robocar

Run the code below to get the S3 location for model data. Download this onto the robocar and test running with it.

In [14]:
```
print(tree.model_data)
```

```
s3://sagemaker-us-west-2-831212071815/output/donkey-2018-05-03-09-44-51-204/output/model.tar.gz
```

创建训练任务的 API-CreateTrainingJob

调用 CreateTrainingJob 接口 API，来启动一个训练任务。当任务结束后，Amazon SageMaker 将模型文件保存在指定的 Amazon S3上。

AlgorithmSpecification - 指定算法

HyperParameters - 配置算法相关的超参数从而优化模型的质量

InputDataConfig - 描述训练数据集，提供 Amazon S3 上数据集的存储位置

OutputDataConfig - 提供 Amazon S3 上模型文件存储的位置

ResourceConfig - 指定所需的计算资源，包括 ML compute instances 和 ML storage volumes，以及分布式训练时机器的数量

RoleARN - 提供 Amazon SageMaker 在训练过程所需要的访问其他服务的权限 The Amazon Resource Number (ARN)

StoppingCondition - 设定任务执行的超时时长

创建训练任务的 API-CreateTrainingJob

```
import boto3, os, datetime

def main(event, context):
    sm = boto3.client('sagemaker')
    training_job_name = 'donkey'+str(datetime.datetime.today()).replace(' ', '-').replace(':', '-').rsplit('.')[0]
    role_arn = "arn:aws:iam::831212071815:role/service-role/AmazonSageMaker-ExecutionRole-20171204T163957”
    algorithm_spec = {
        "TrainingImage": "831212071815.dkr.ecr.us-west-2.amazonaws.com/donkey:latest",
        "TrainingInputMode": "File"
    }
    resource_config = {
        "InstanceType": "ml.c5.2xlarge",
        "InstanceCount": 1,
        "VolumeSizeInGB": 30
    }
    stopping_condition = { "MaxRuntimeInSeconds": 86400 }
```

创建训练任务的 API-CreateTrainingJob

```
input_data = [{
        "ChannelName": "training",
        "DataSource": {
            "S3DataSource": {
                "S3DataType": "S3Prefix",
                "S3Uri": "s3://autonomous-vehicles-data",
                "S3DataDistributionType": "FullyReplicated"
            }
        },
        "CompressionType": "None",
        "RecordWrapperType": "None"
}]
output_data = {
    "KmsKeyId": "",
    "S3OutputPath": "s3://sagemaker-us-west-2-831212071815/output"
}
```

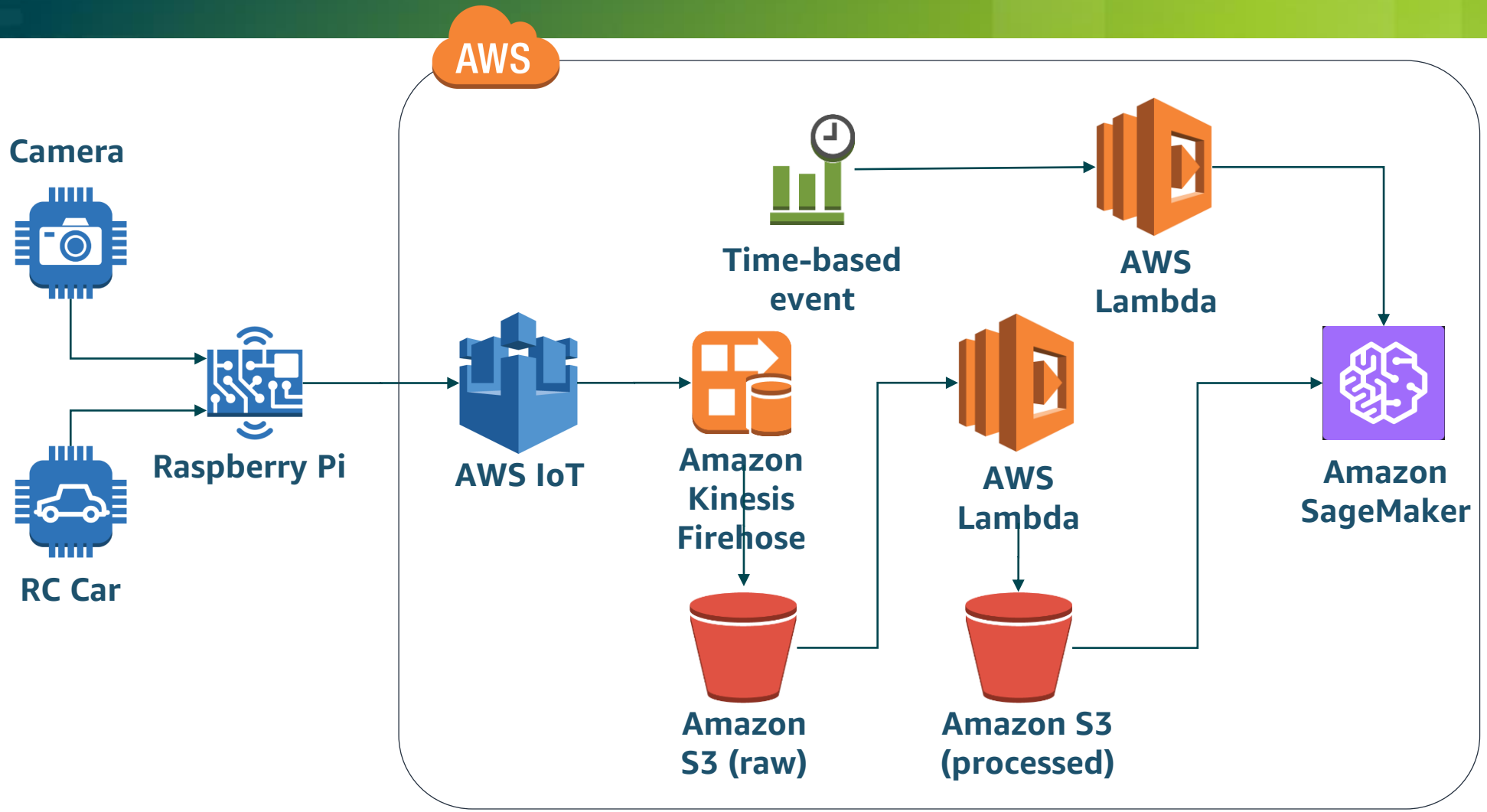
创建训练任务的 API-CreateTrainingJob

```
print("Starting training job %s" % training_job_name)
resp = sm.create_training_job(
    TrainingJobName=training_job_name,
    AlgorithmSpecification=algorithm_spec,
    RoleArn=role_arn,
    InputDataConfig=input_data,
    OutputDataConfig=output_data,
    ResourceConfig=resource_config,
    StoppingCondition=stopping_condition,
    HyperParameters={},
    Tags=[])

print(resp)
```

创建训练任务的 API-CreateTrainingJob


```
{
  "TrainingJobArn": "arn:aws:sagemaker:us-west-2:831212071815:training-job/donkey2018-05-29-08-09-54",
  "ResponseMetadata": {
    "RequestId": "6db2646c-55e6-4eb9-9d21-42809d8bcd04",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "content-type": "application/x-amz-json-1.1",
      "date": "Tue, 29 May 2018 08:09:54 GMT",
      "x-amzn-requestid": "6db2646c-55e6-4eb9-9d21-42809d8bcd04",
      "content-length": "100",
      "connection": "keep-alive”
    },
    "RetryAttempts": 0
  }
}
```

最终架构


AWS
Camera
RC Car
Raspberry Pi
AWS IoT
Time-based event
Amazon Kinesis Firehose
Amazon S3 (raw)
AWS Lambda
Amazon S3 (processed)
AWS Lambda
Amazon SageMaker

架构优化?

数据采集

机器学习流水线

模型发布（AWS Greengrass）

...

感谢参加 AWS INNOVATE 2018 在线技术大会

我们希望您在这里找到感兴趣的内容！
也请帮助我们完成**投票打分**和**反馈问卷**。
欲获取关于 AWS 的更多信息和技术内容，可以通过以下方式找到我们：

- 微信公众号：AWSChina
- 新浪微博：https://www.weibo.com/amazonaws/
- 领英：https://www.linkedin.com/company/aws-china/
- 知乎：https://www.zhihu.com/org/aws-54/activities/
- 视频中心：http://aws.amazon.bokecc.com/
- 更多线上活动：https://aws.amazon.com/cn/about-aws/events/webinar/