KubeCon | CloudNativeCon

Europe 2019

Do you really need on premises serverless ?

Igor Khapov

Who am I ?

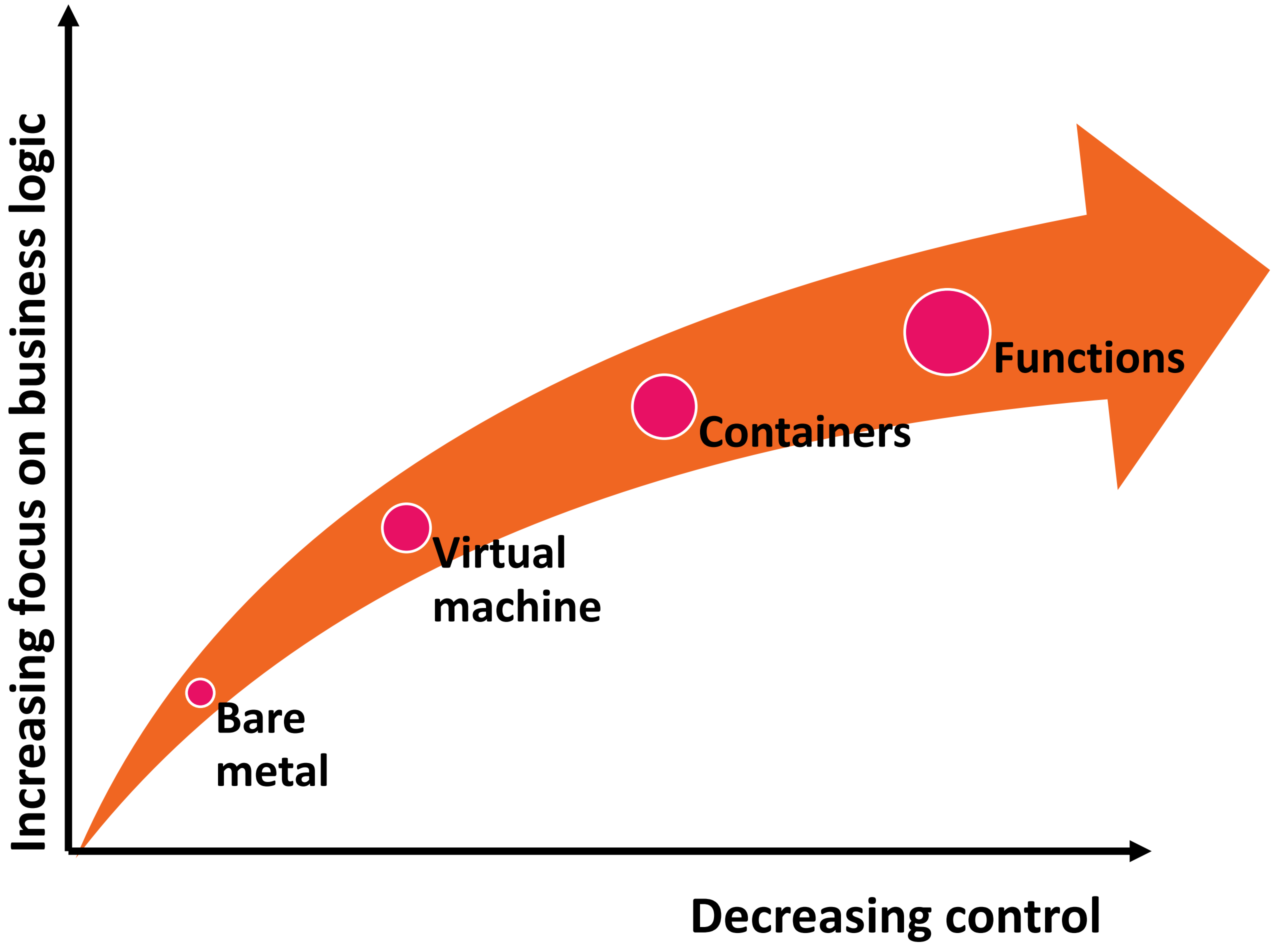
- **Igor Khapov**
  - #ibm #moscow_dev_lab #developer #manager #kubernetes #serverless
  - #x86-64_ppc64le #data_science_platform

What is serverless ?

KubeCon CloudNativeCon Europe 2019

Serverless architectures

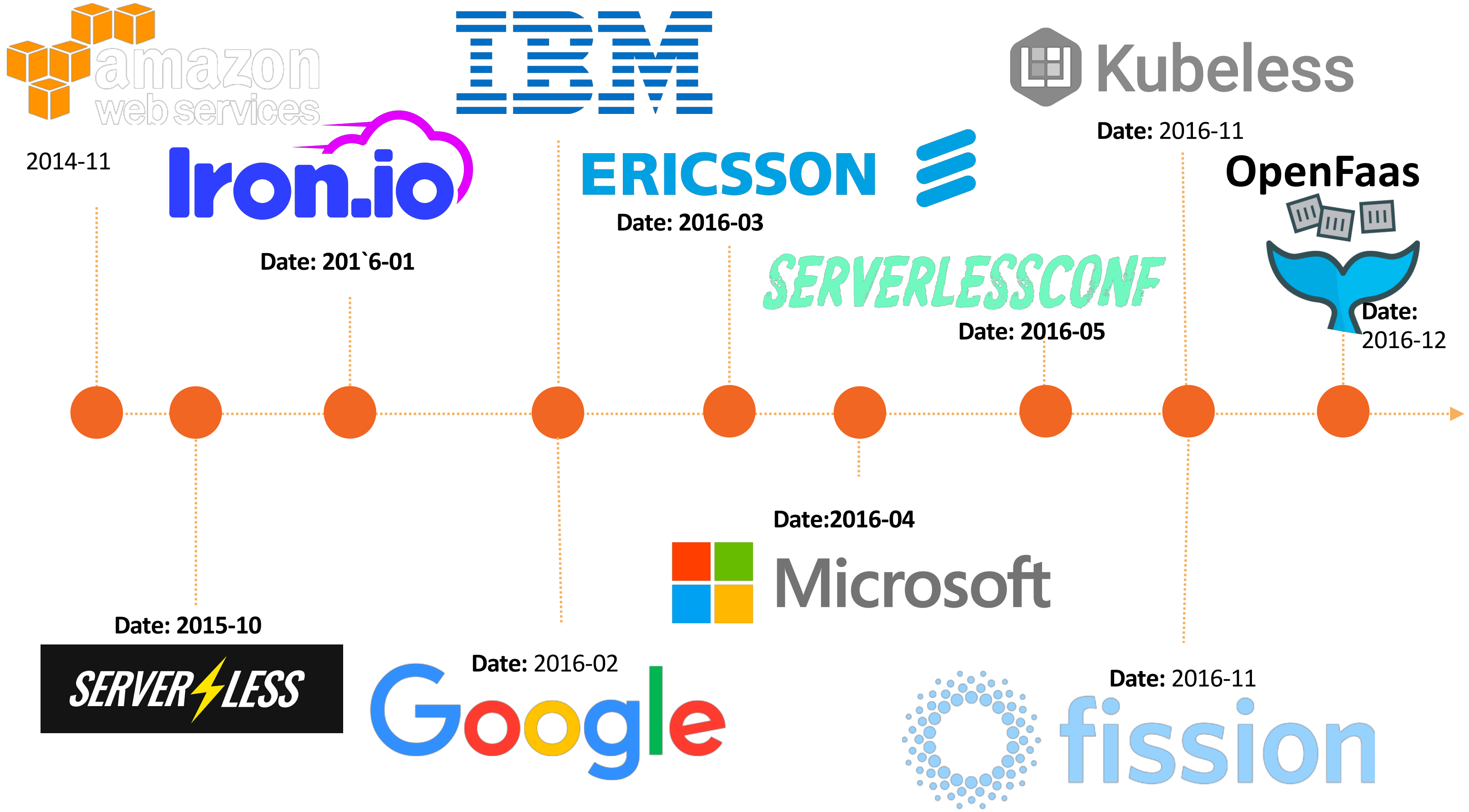
are application designs that incorporate third-party “Backend as a Service” services, and include custom code run in managed, ephemeral containers on a “Functions as a Service” platform. *



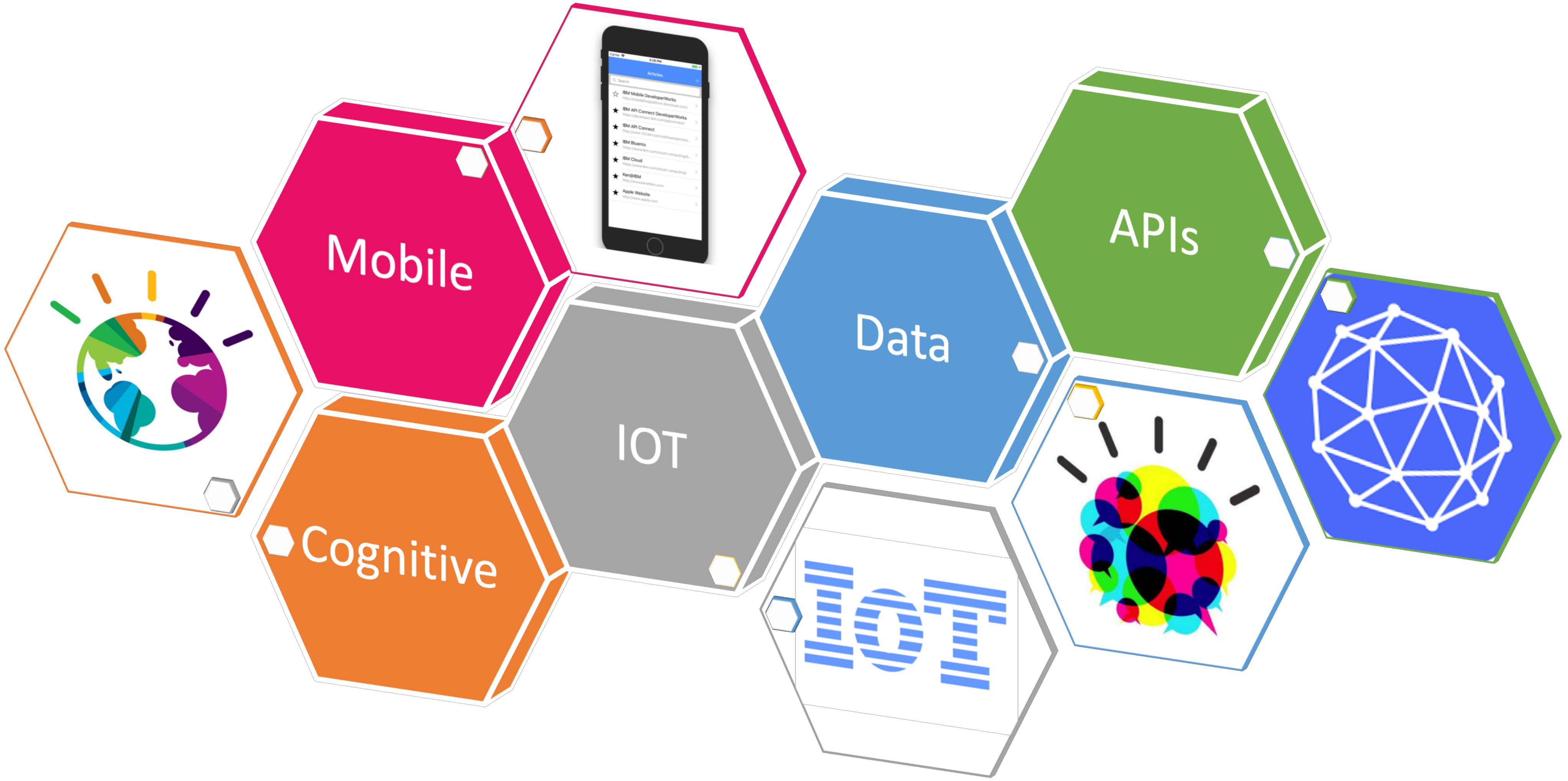
History

Launch Timeline

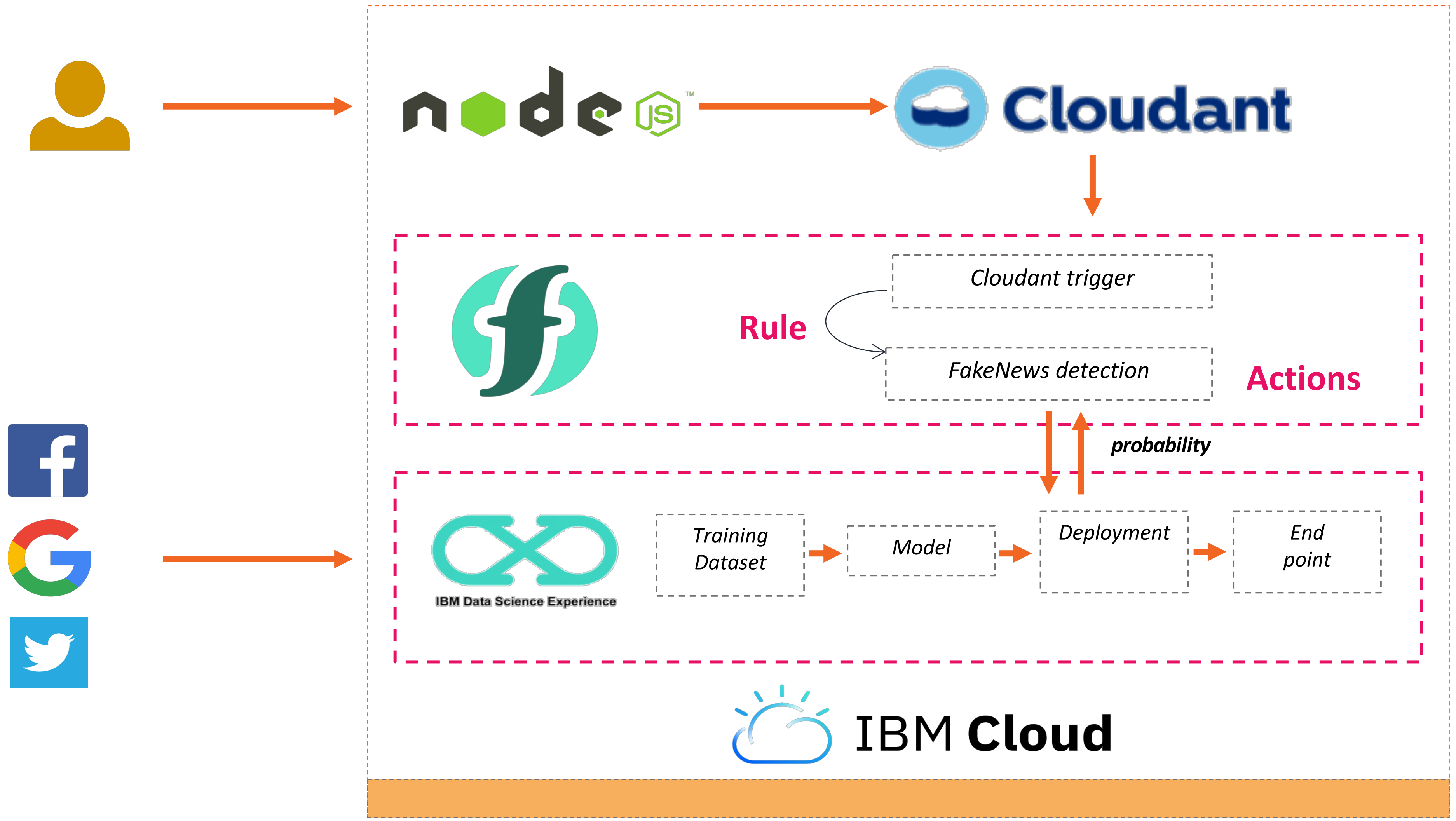
#open_source #serverless
#platforms #trend #history



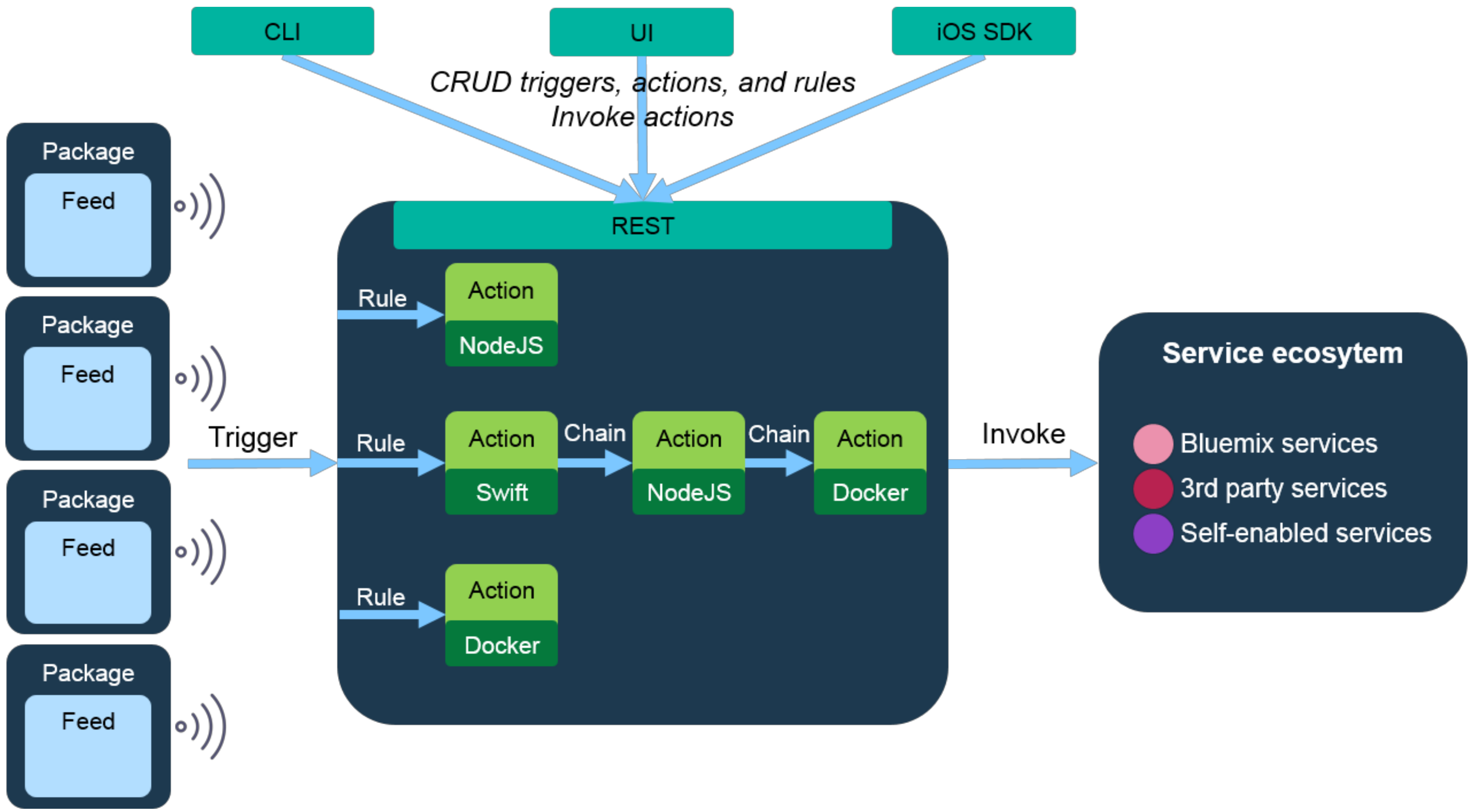
Main use cases



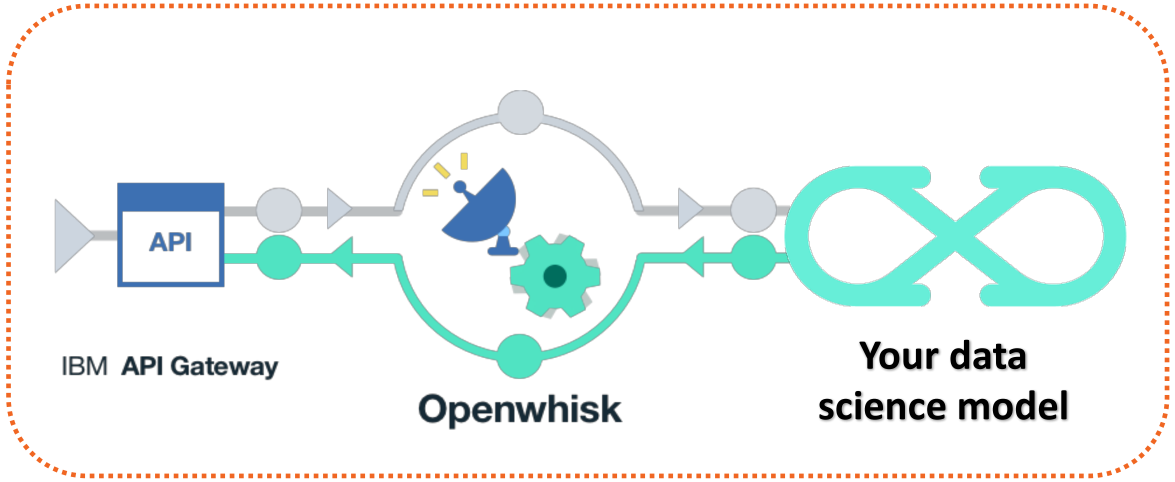
My first use case

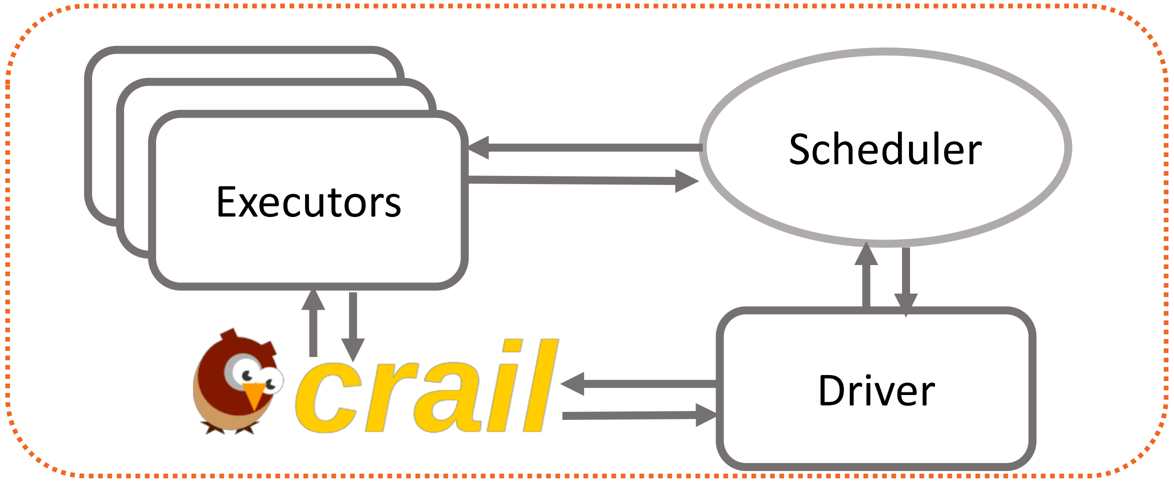


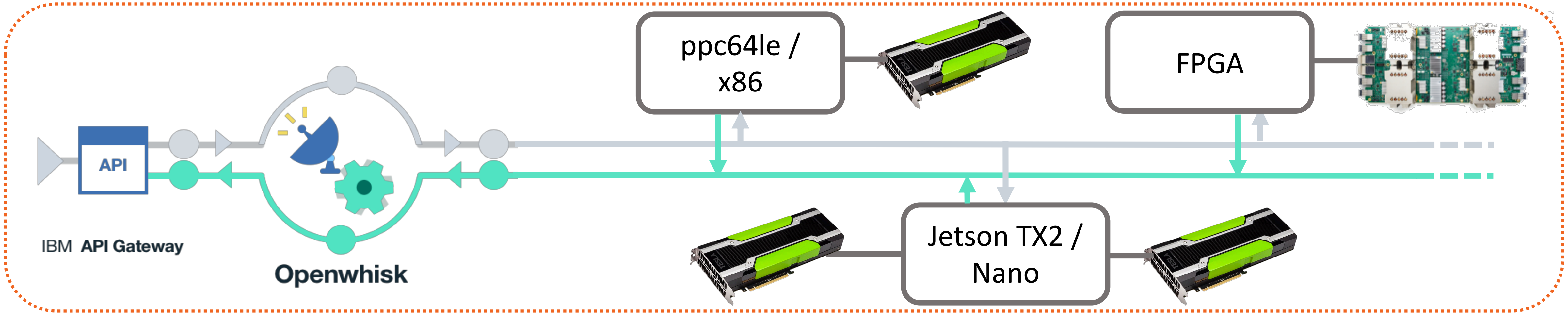
OpenWhisk architecture



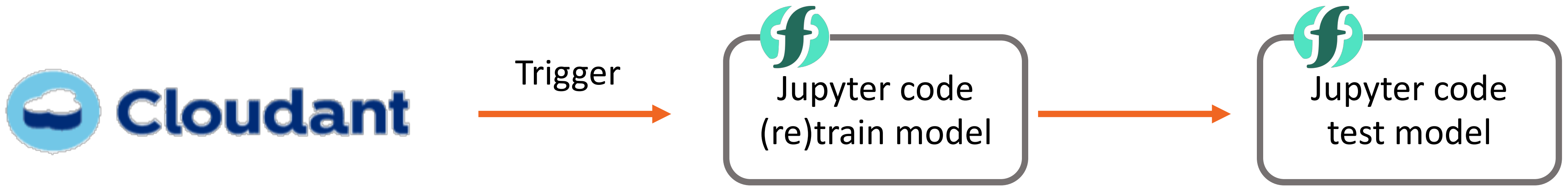
Serverless and data science





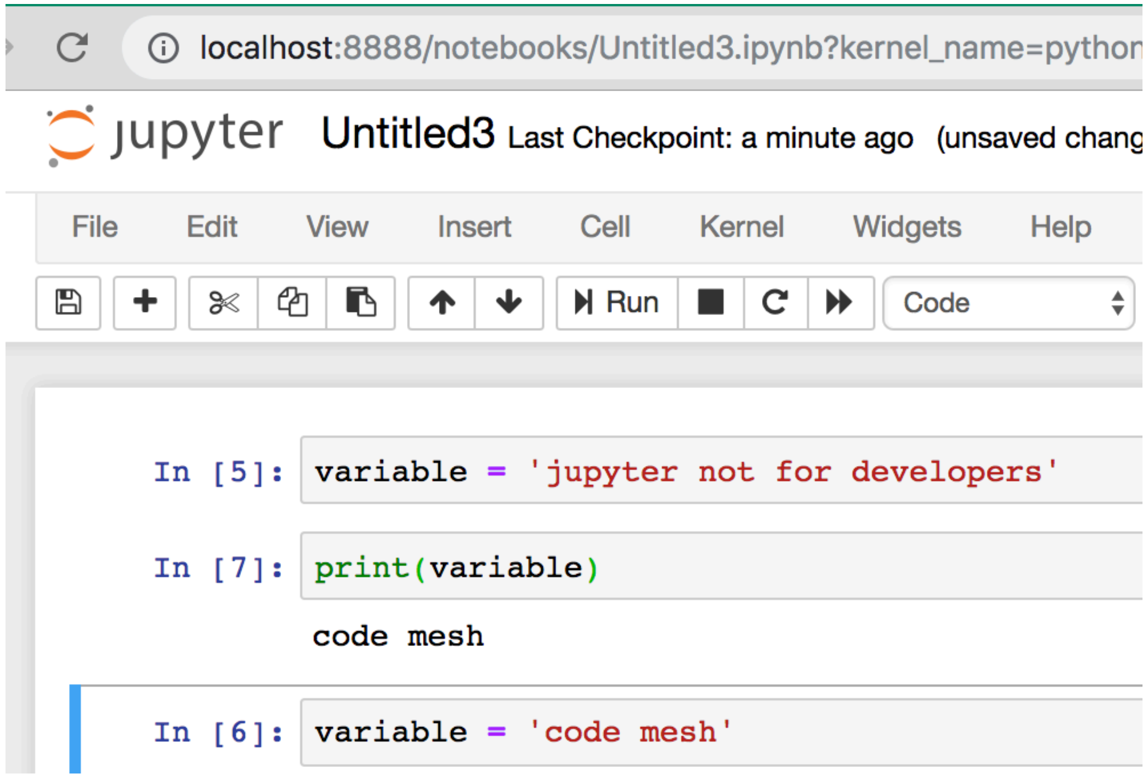


Function and data science

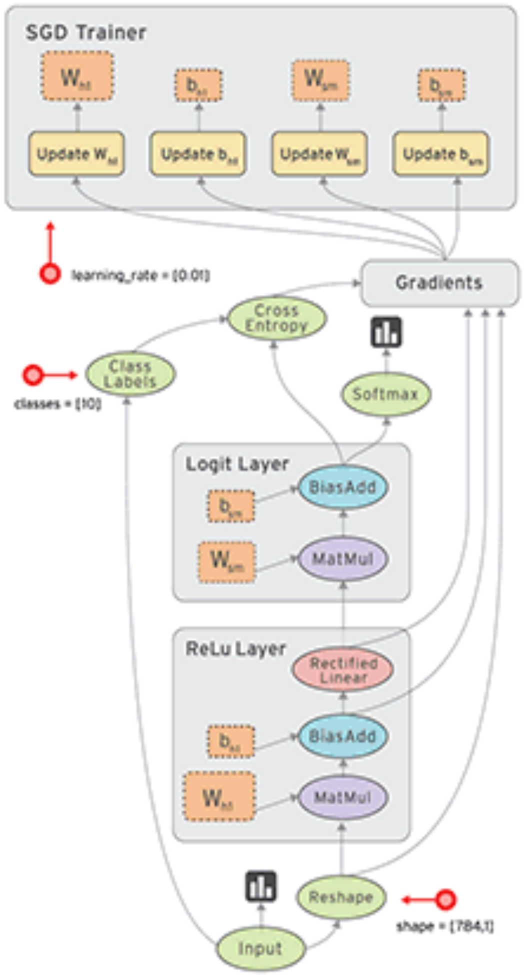


Jupyter nb flow process

KubeCon | CloudNativeCon Europe 2019

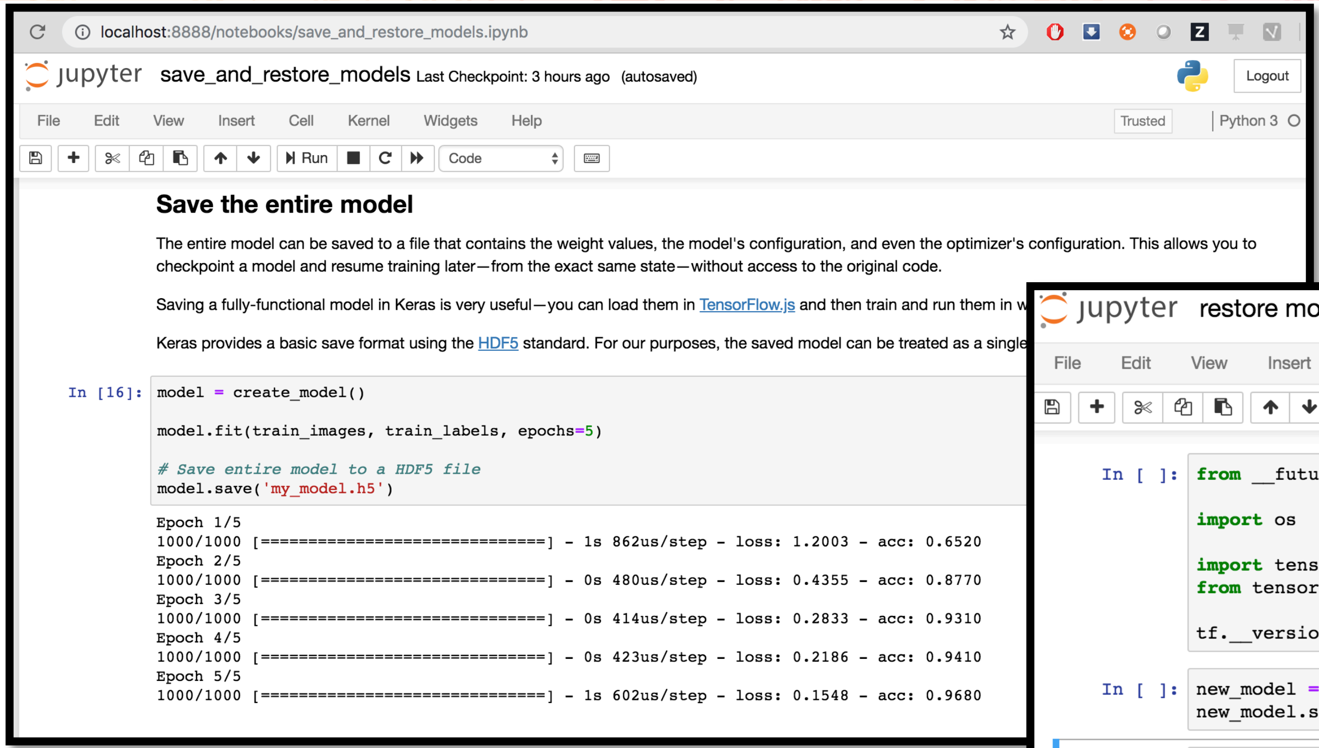

localhost:8888/notebooks/Untitled3.ipynb?kernel_name=python
jupyter Untitled3 Last Checkpoint: a minute ago (unsaved chang
File Edit View Insert Cell Kernel Widgets Help
Run
Code

```
In [5]: variable = 'jupyter not for developers'

In [7]: print(variable)

        code mesh

In [6]: variable = 'code mesh'
```



TF implementation
Save and restore a model

KubeCon CloudNativeCon Europe 2019



localhost:8888/notebooks/save_and_restore_models.ipynb

jupyter save_and_restore_models Last Checkpoint: 3 hours ago (autosaved)

Save the entire model

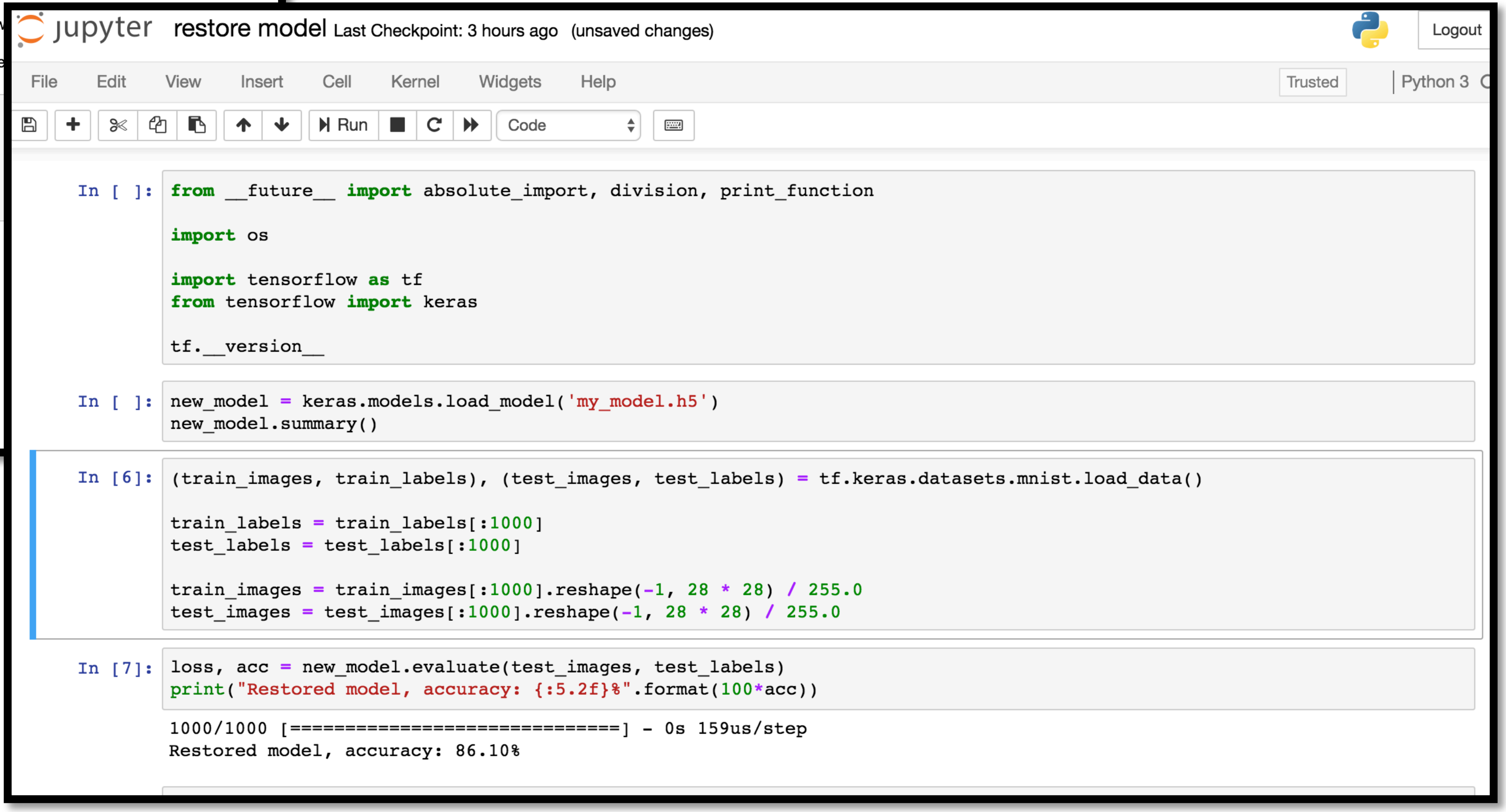
The entire model can be saved to a file that contains the weight values, the model's configuration, and even the optimizer's configuration. This allows you to checkpoint a model and resume training later—from the exact same state—without access to the original code.

Saving a fully-functional model in Keras is very useful—you can load them in TensorFlow.js and then train and run them in w

Keras provides a basic save format using the HDF5 standard. For our purposes, the saved model can be treated as a single

```
In [16]: model = create_model()

         model.fit(train_images, train_labels, epochs=5)

         # Save entire model to a HDF5 file
         model.save('my_model.h5')
```

```
Epoch 1/5
1000/1000 [==============================] - 1s 862us/step - loss: 1.2003 - acc: 0.6520
Epoch 2/5
1000/1000 [==============================] - 0s 480us/step - loss: 0.4355 - acc: 0.8770
Epoch 3/5
1000/1000 [==============================] - 0s 414us/step - loss: 0.2833 - acc: 0.9310
Epoch 4/5
1000/1000 [==============================] - 0s 423us/step - loss: 0.2186 - acc: 0.9410
Epoch 5/5
1000/1000 [==============================] - 1s 602us/step - loss: 0.1548 - acc: 0.9680
```



jupyter restore model Last Checkpoint: 3 hours ago (unsaved changes)

```
In [ ]: from __future__ import absolute_import, division, print_function

        import os

        import tensorflow as tf
        from tensorflow import keras

        tf.__version__
```

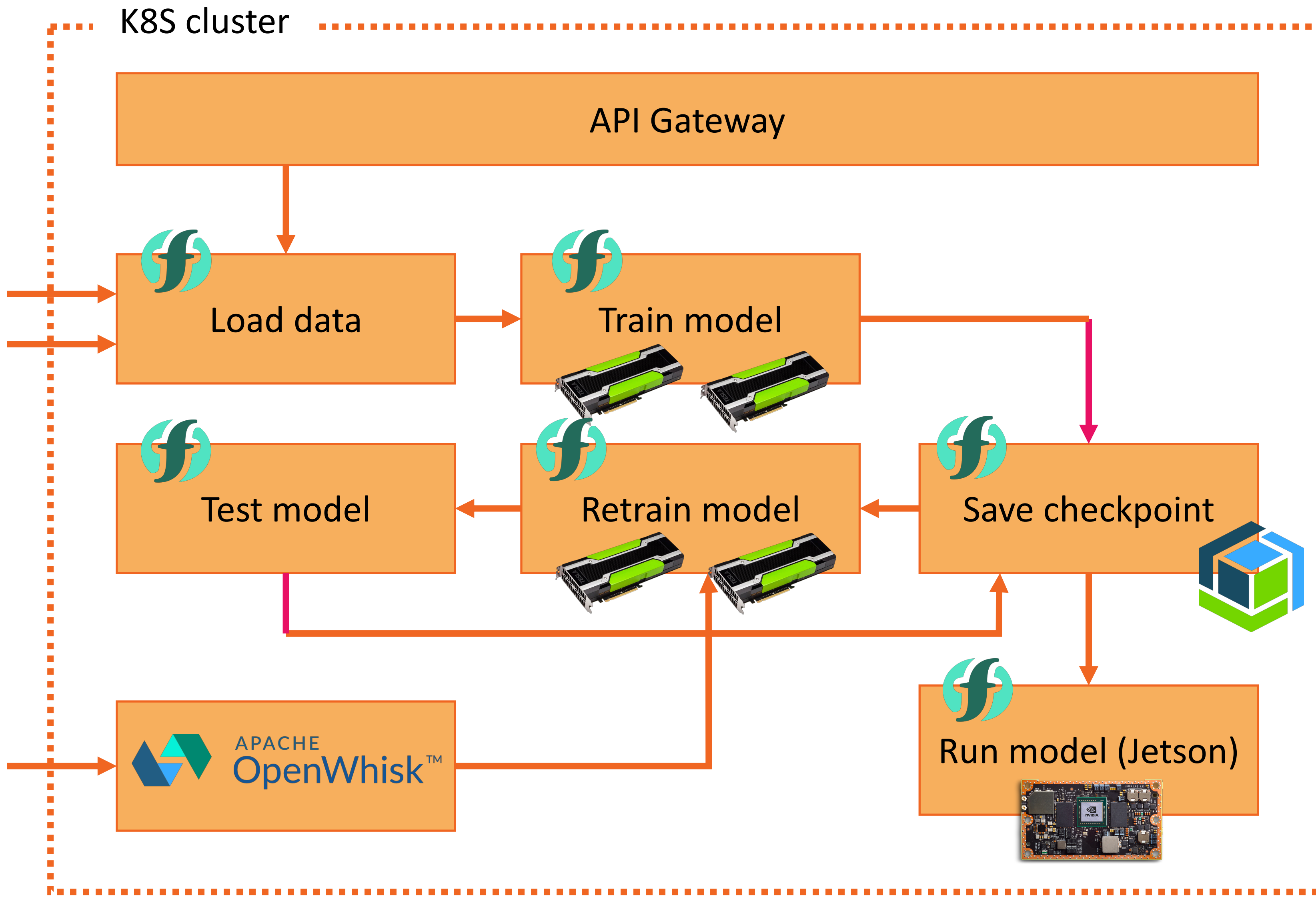
```
In [ ]: new_model = keras.models.load_model('my_model.h5')
        new_model.summary()
```

```
In [6]: (train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data()

        train_labels = train_labels[:1000]
        test_labels = test_labels[:1000]

        train_images = train_images[:1000].reshape(-1, 28 * 28) / 255.0
        test_images = test_images[:1000].reshape(-1, 28 * 28) / 255.0
```

```
In [7]: loss, acc = new_model.evaluate(test_images, test_labels)
        print("Restored model, accuracy: {:5.2f}%".format(100*acc))
```

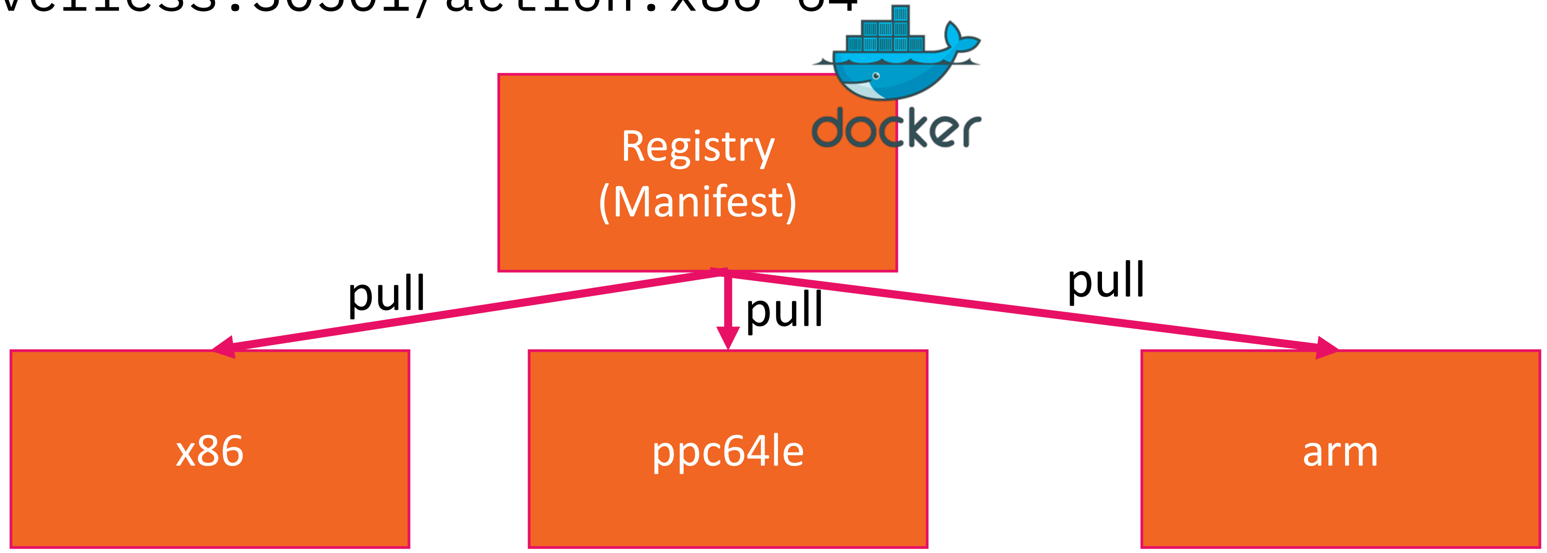
```
1000/1000 [==============================] - 0s 159us/step
Restored model, accuracy: 86.10%
```

Target architecture



Docker for multiple architectures

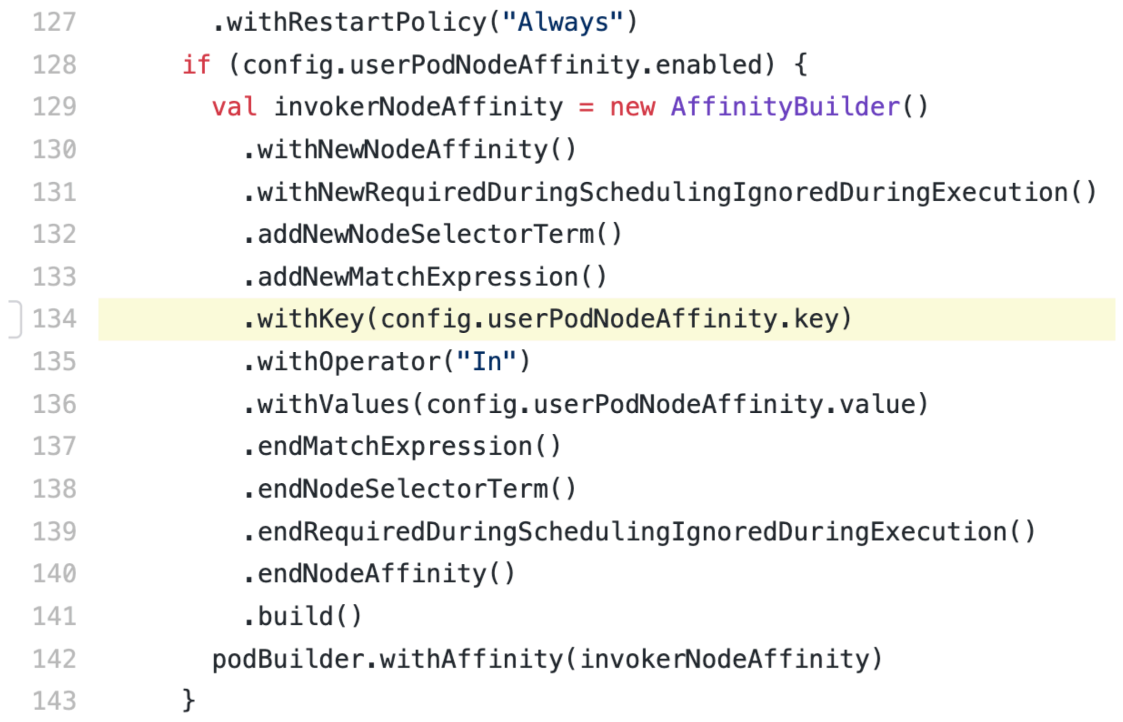
```
docker -D manifest create –insecure \
      serverless:30501/action:latest \
      serverless:30501/action:ppc64le \
      serverless:30501/action:x86-64
```



```
root@serverless:~# docker images |grep ac1|grep -v 18 |grep -v none
serverless:30501/ac1                    ppc64le                  1a9dd94f6deb       2 weeks ago           200MB
serverless:30501/ac1                    latest                   cb82052802de       5 weeks ago           172MB
serverless:30501/ac1                    x86-64                   cb82052802de       5 weeks ago           172MB
```

Scheduler customisation

KubernetesClient.scala



```scala
      .withRestartPolicy("Always")
    if (config.userPodNodeAffinity.enabled) {
      val invokerNodeAffinity = new AffinityBuilder()
        .withNewNodeAffinity()
        .withNewRequiredDuringSchedulingIgnoredDuringExecution()
        .addNewNodeSelectorTerm()
        .addNewMatchExpression()
        .withKey(config.userPodNodeAffinity.key)
        .withOperator("In")
        .withValues(config.userPodNodeAffinity.value)
        .endMatchExpression()
        .endNodeSelectorTerm()
        .endRequiredDuringSchedulingIgnoredDuringExecution()
        .endNodeAffinity()
        .build()
      podBuilder.withAffinity(invokerNodeAffinity)
    }
```

KubernetesContainerFactory.scala

KubernetesContainer.scala

KubernetesContainerFactory.scala

InvokerReactive.scala

KubernetesContainerFactoryProvider

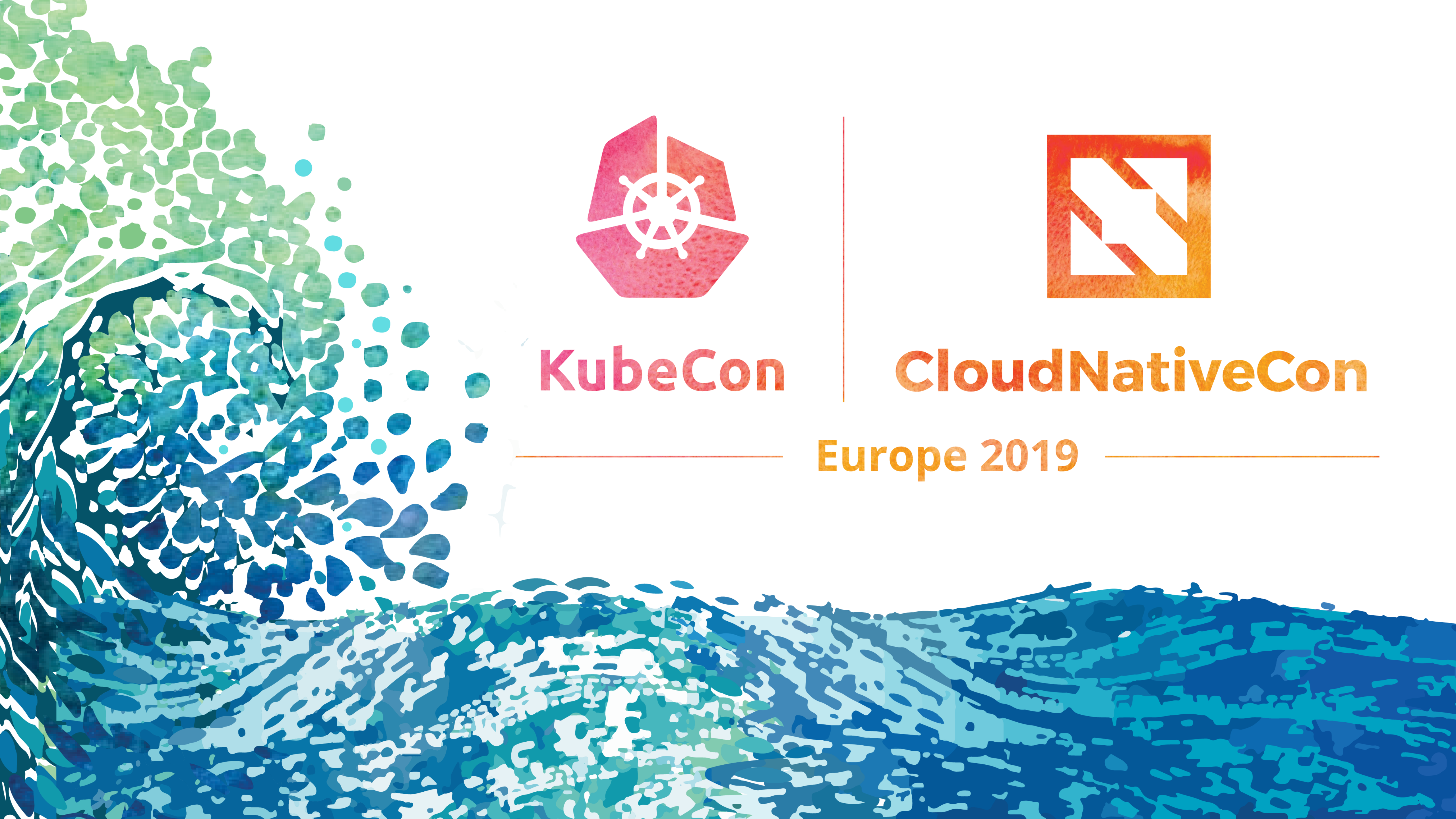
Demo

Is all actions should be hardware agnostic ?

- Collocation to the data warehouse
- Selectors for GPU / TPU resources
- Selectors for resources (RAM, cores …)

You need on premise serverless if

- You have a lot of in-company developer and you want to simplifier their job
- You have a range of functions whish in NOT hardware agnostic
- You want to increase utilization of your resources
- You want to split your workflow into small steps and store temporary results
- You have some time to implement or adopt that



KubeCon | CloudNativeCon

Europe 2019