Tensorflow Serving

손준영

Covered...

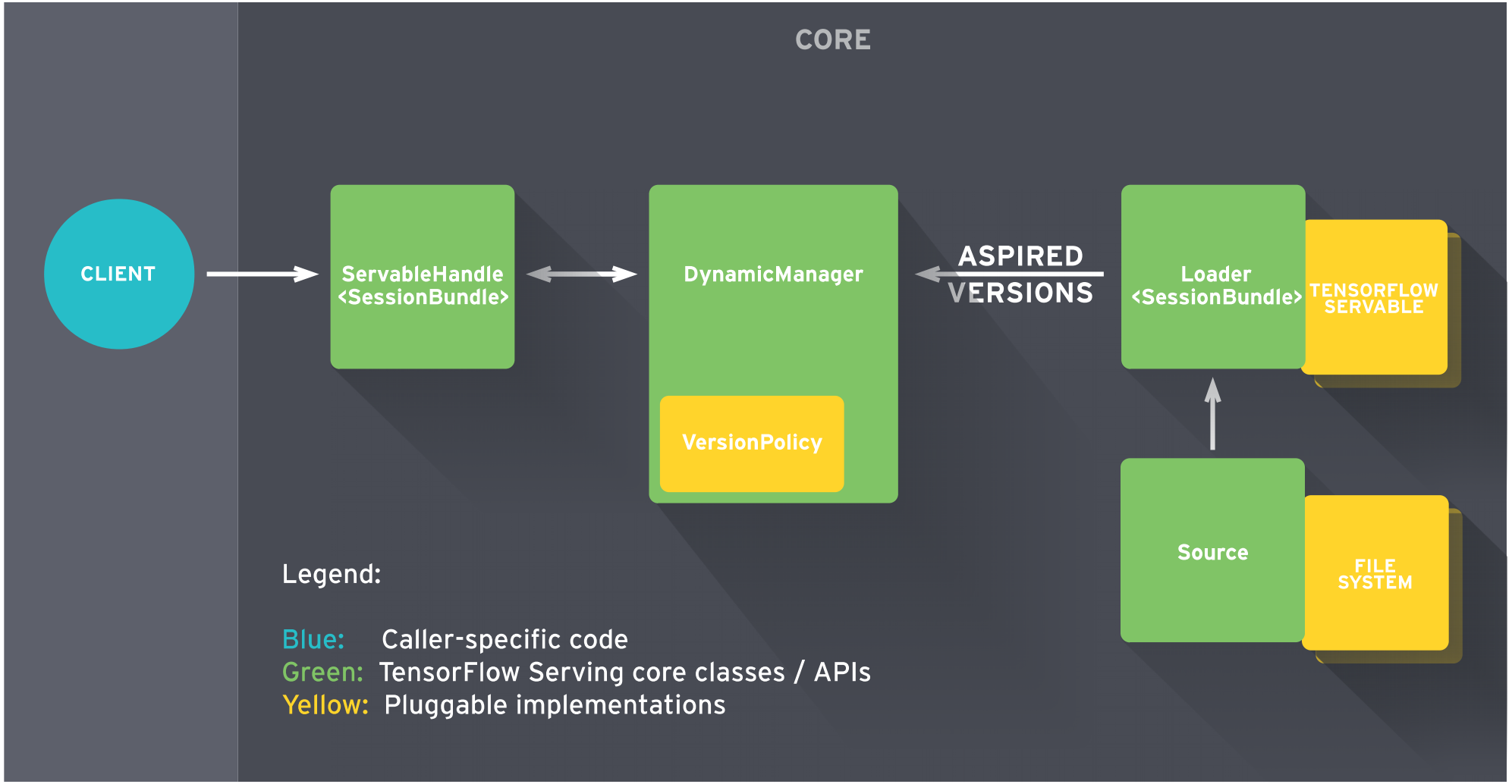
Get Started

- Architecture Overview
- Installation

Tutorials

- TensorFlow Serving Basics
- TensorFlow Serving Advanced

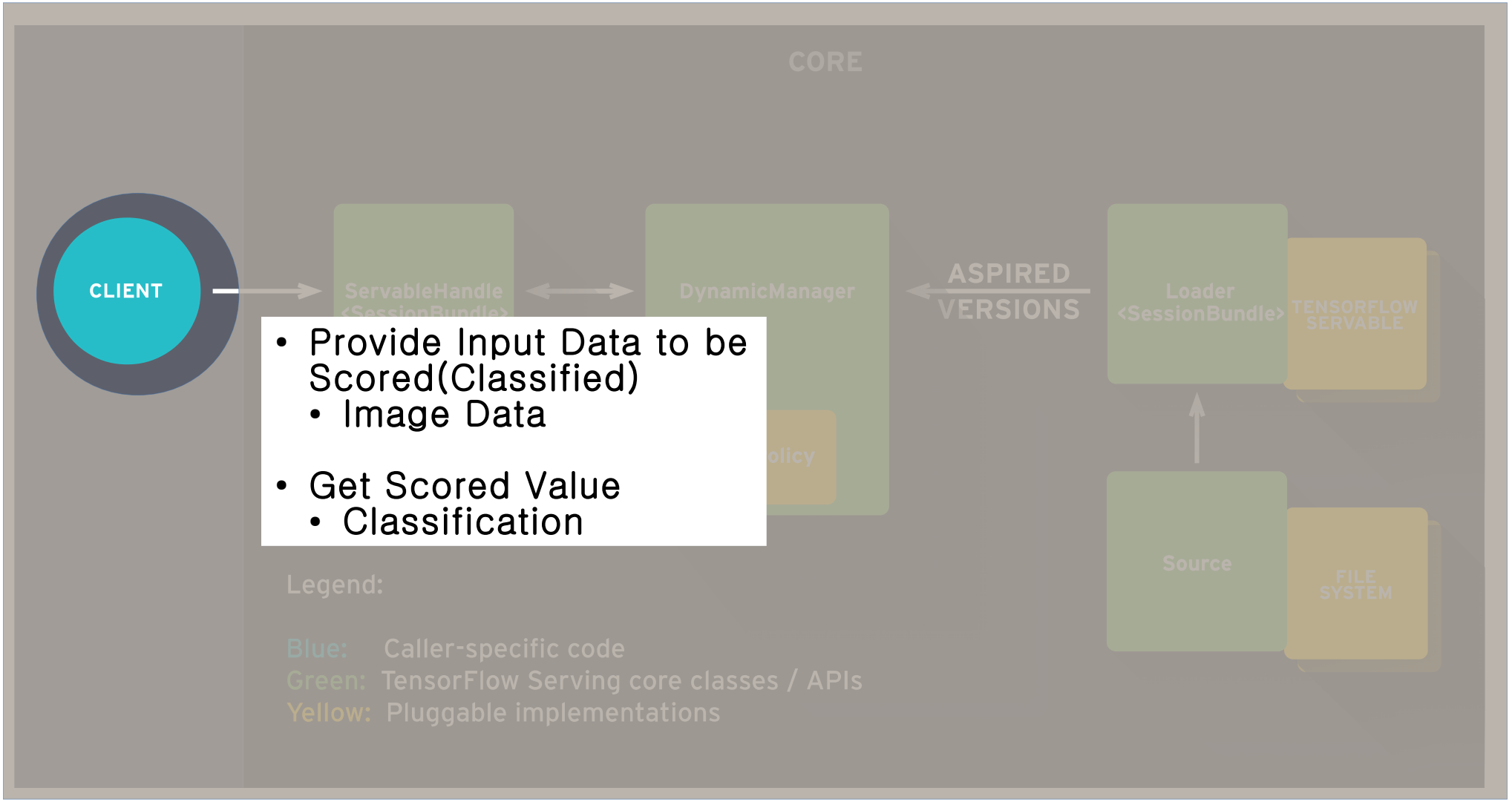
Architecture Overview



https://tensorflow.github.io/serving/architecture_overview

Get Started

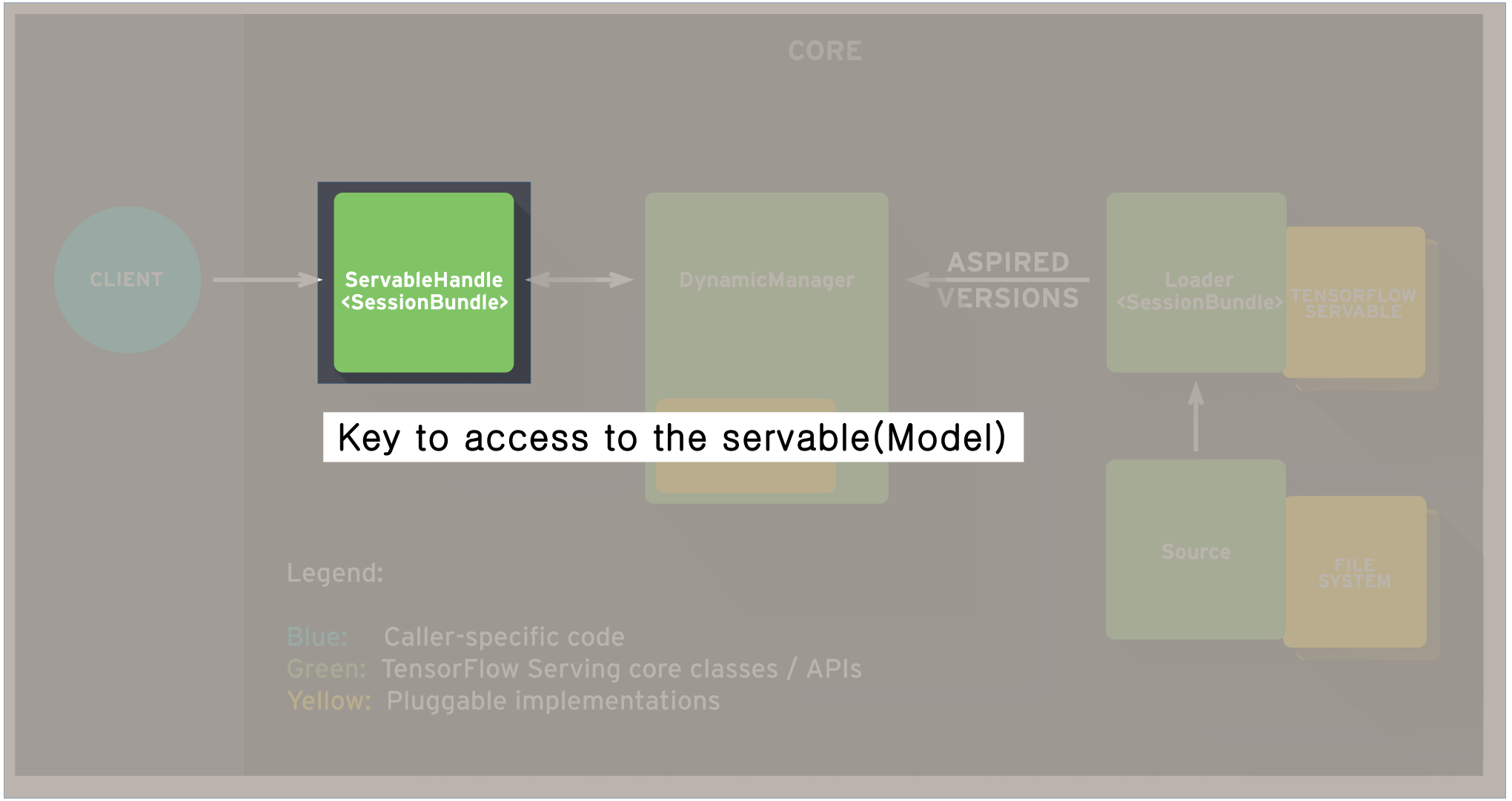
Architecture Overview



https://tensorflow.github.io/serving/architecture_overview

Get Started

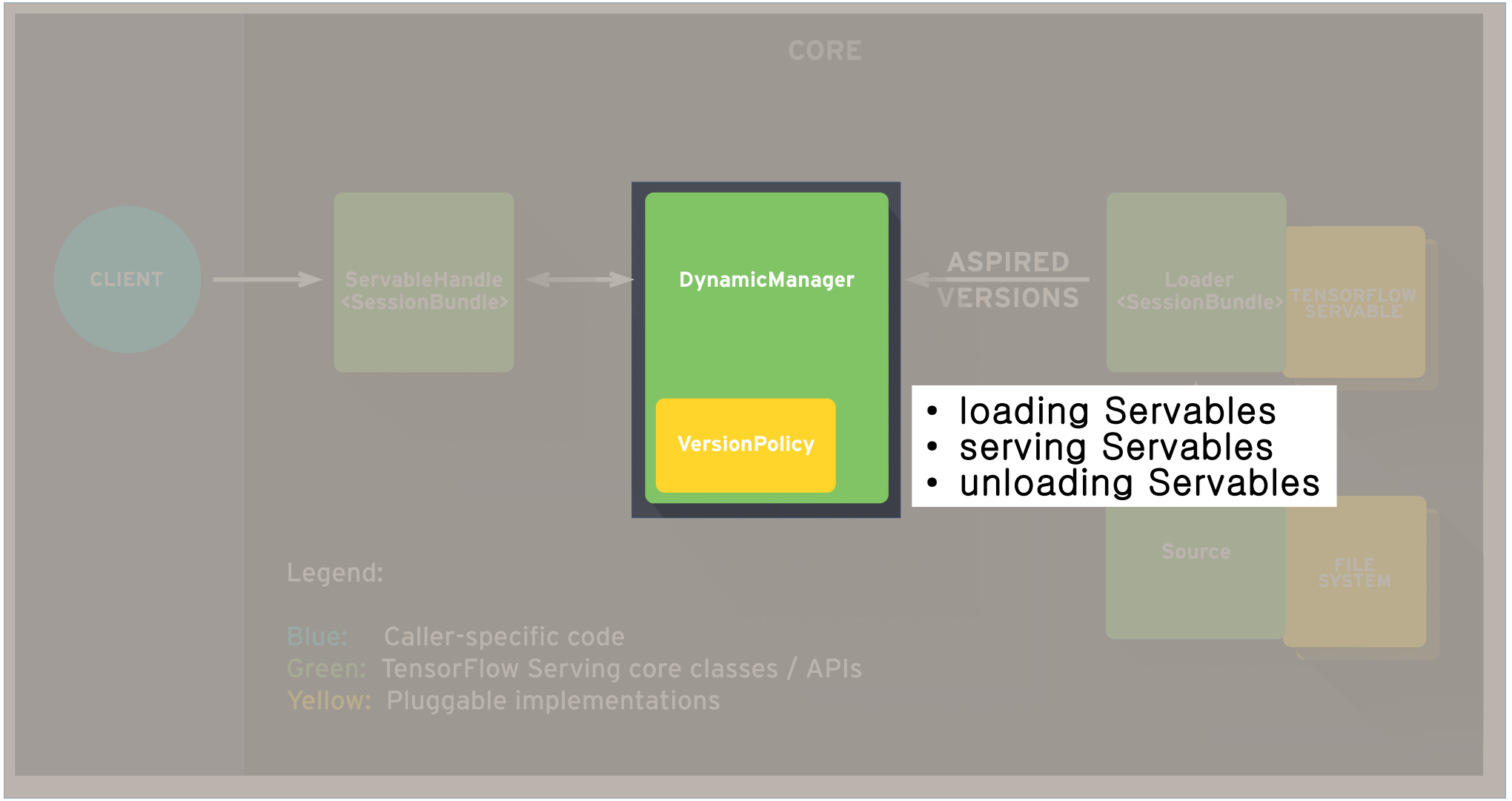
Architecture Overview



https://tensorflow.github.io/serving/architecture_overview

Get Started

Architecture Overview

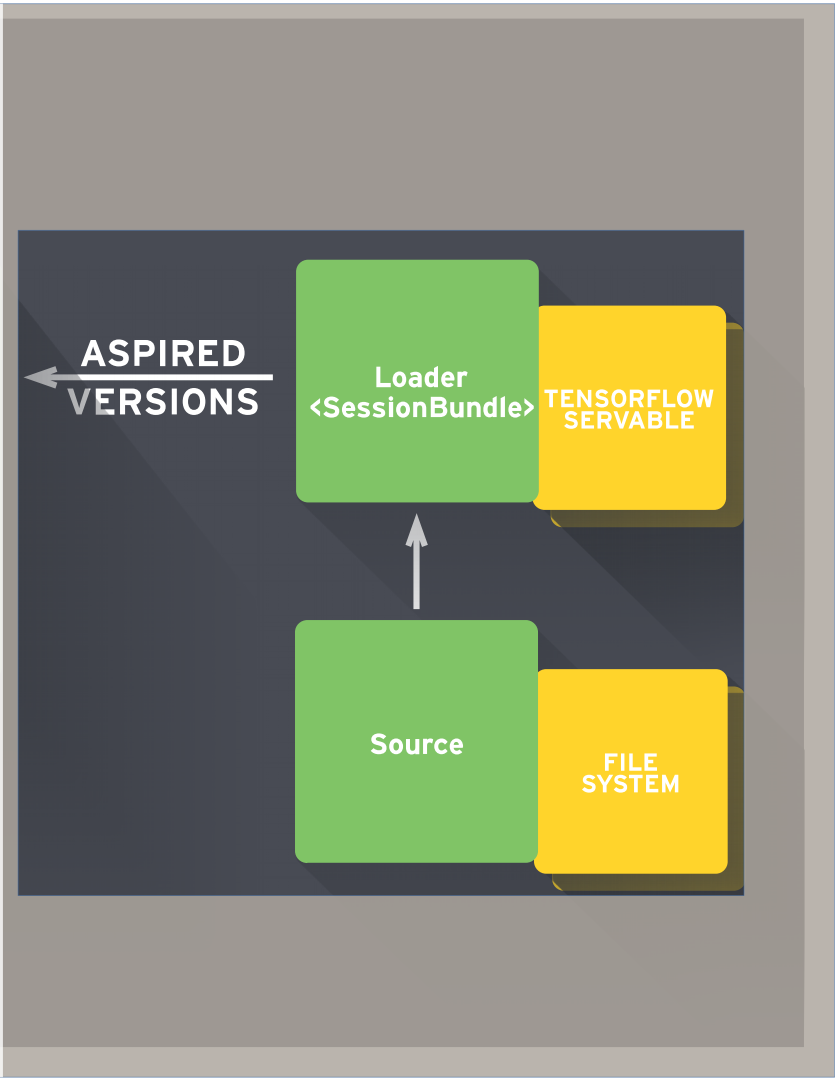


https://tensorflow.github.io/serving/architecture_overview

Get Started

Architecture Overview

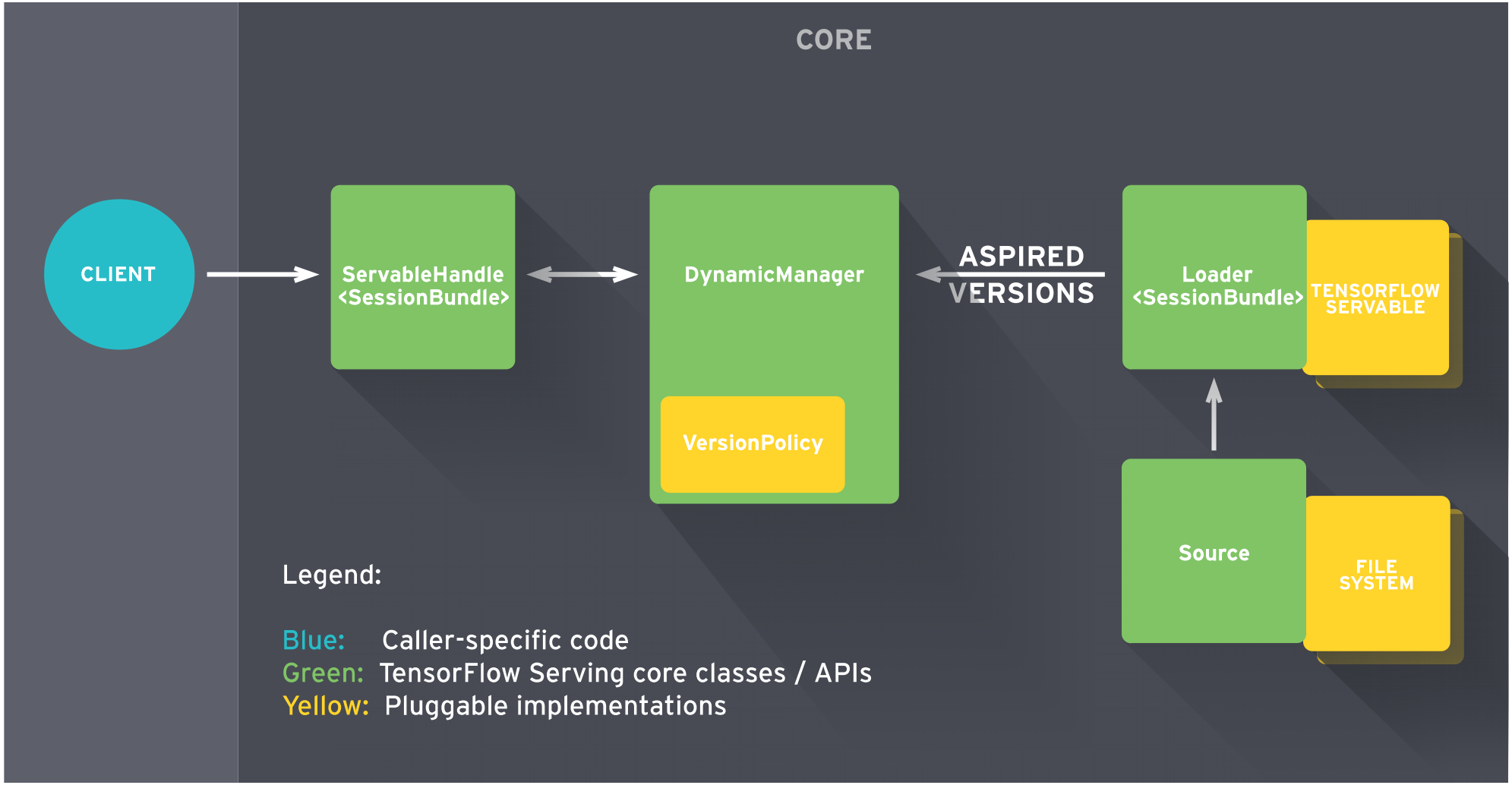
1. Servables
   1. Versions
   2. Model + Parameter(Weight, bias..)
   3. One Model can be sharded into several servable instances
2. Loaders
   1. Manage servable's lifecycle
   2. enable common infrastructure independent from specific learning algorithms
3. Sources
   1. originate servables.(stream of versions)
   2. one loader for each each version
   3. Aspired Versions – set of versions that should be loaded and ready



https://tensorflow.github.io/serving/architecture_overview

Get Started

Architecture Overview



Installation

- **Prerequisites**
  - **Bazel**
    https://github.com/bazelbuild/bazel/releases

    Compiling Package

    Java8 Recommended – (Set JAVA_HOME)
  - **Python 2.7(gRPC dependent)**
  - **GRPC(Google Remote Procedure Call)**
    - http://www.grpc.io/
    - Cloud api service protocol by google.
    - Use Protobuff for Data serialization of structured data
    - (sudo) pip install grpcio
  - **Annaconda User**

    ~ $ conda create -n python2 python=2.7 anaconda

    ~ $ source activate python2

Installation – Other Dep.

- sudo apt-get update && sudo apt-get install -y \
  build-essential \
  curl \
  git \
  libfreetype6-dev \
  libpng12-dev \
  libzmq3-dev \
  pkg-config \
  python-dev \
  python-numpy \
  python-pip \
  software-properties-common \
  swig \
  zip \
  zlib1g-dev

Tutorial - Basics

1. Train and Export TensorFlow Model

$>bazel build //tensorflow_serving/example:mnist_export

$>bazel-bin/tensorflow_serving/example/mnist_export /tmp/mnist_model

2. Bring Up Inference Service

$>bazel build //tensorflow_serving/example:mnist_inference

$>bazel-bin/tensorflow_serving/example/mnist_inference --port=9000 /tmp/mnist_model/00000001

(Load Exported Tensor Flow Model)

3. Test the Server

$>bazel build //tensorflow_serving/example:mnist_client

$>bazel-bin/tensorflow_serving/example/mnist_client --num_tests=1000 --server=localhost:9000

Train and Export TensorFlow Model

```
76  # Export model
77  # WARNING(break-tutorial-inline-code): The following code snippet is
78  # in-lined in tutorials, please update tutorial documents accordingly
79  # whenever code changes.
80  export_path = sys.argv[-1]
81  print 'Exporting trained model to', export_path
82  saver = tf.train.Saver(sharded=True)
83  model_exporter = exporter.Exporter(saver)
84  signature = exporter.classification_signature(input_tensor=x, scores_tensor=y)
85  model_exporter.init(sess.graph.as_graph_def(),
86              default_graph_signature=signature)
87  model_exporter.export(export_path, tf.constant(FLAGS.export_version), sess)
88  print 'Done exporting!'
```

Saver : Serialize graph variable values to the model export
Sess.graph.as_graph_def() : protobuf of the graph
default_graph_signature : model export signature
    exporter.classification_signature; Classification model
    Tensor binding
        input_tensor = x
        scores_tensor = y

Check Exported model

$>ls /tmp/mnist_model

00000001

$>ls /tmp/mnist_model/00000001

checkpoint export-00000-of-00001 export.meta

Export.meta : seriallized tensorflow::MetaGraphDef of the model.

Graph Def. + metadata(Signatures)

Export-????-of-???? : serialized variables of the graph

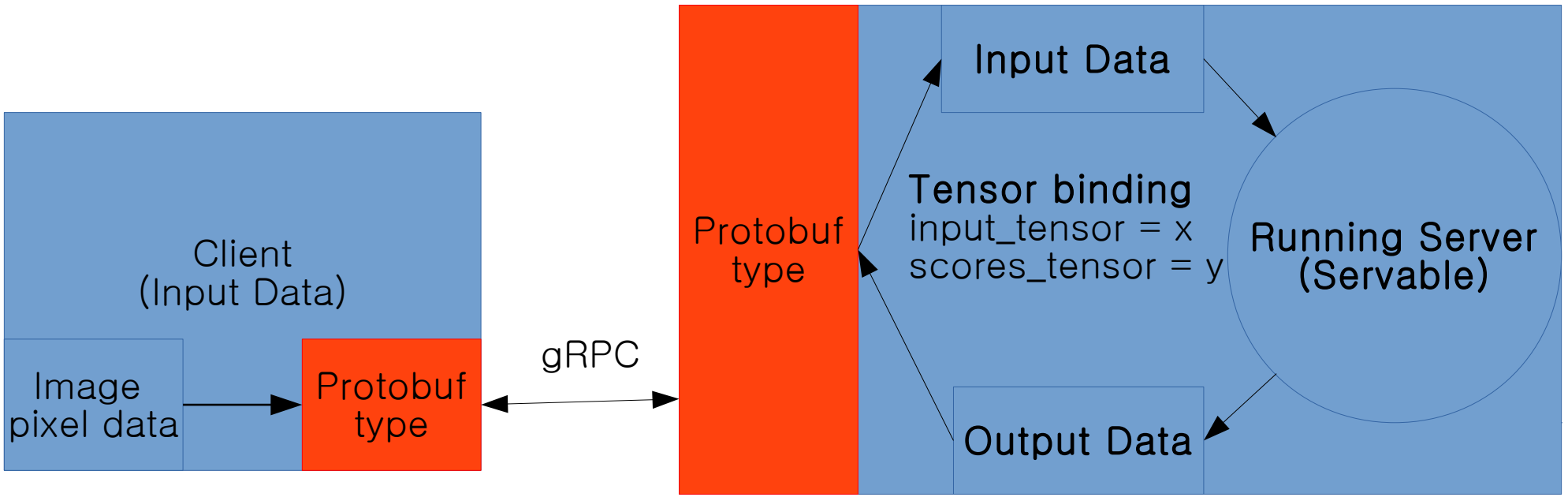
Bring Up Inference Service

Load Exported Tensorflow Model

```
184   tensorflow::SessionOptions session_options;
185   std::unique_ptr<SessionBundle> bundle(new SessionBundle);
186   const tensorflow::Status status =
187     tensorflow::serving::LoadSessionBundleFromPath(session_options,
188                                 bundle_path, bundle.get());
189   if (!status.ok()) {
190    LOG(ERROR) << "Fail to load tensorflow export: " << status.error_message();
191    return -1;
192   }
193
194   RunServer(FLAGS_port, std::move(bundle));
195
196   return 0;
197 }
198
```

LoadSessionBundleFromPath : load exported tensorflow model
& Create SessionBundle

Bring Up Inference Service



Tutorial - Advanced

1. Version Management

Dynamic Manager.

Unload Older versions and Load a new version models

2. Batching Scheduler

The next task would cause the batch to exceed the size target.

Waiting for more tasks to be added would exceed the timeout.

Tutorial - Advanced

```
413  UniquePtrWithDeps<tensorflow::serving::Manager> manager;
414  tensorflow::Status status = tensorflow::serving::simple_servers::
415    CreateSingleTFModelManagerFromBasePath(export_base_path, &manager);
416
417  TF_CHECK_OK(status) << "Error creating manager";
418
419  // Wait until at least one model is loaded.
420  std::vector<tensorflow::serving::ServableId> ready_ids;
421  // TODO(b/25545573): Create a more streamlined startup mechanism than polling.
422  do {
423   LOG(INFO) << "Waiting for models to be loaded...";
424   tensorflow::Env::Default()->SleepForMicroseconds(1 * 1000 * 1000 /*1 sec*/);
425   ready_ids = manager->ListAvailableServableIds();
426  } while (ready_ids.empty());
427
428  // Run the service.
429  RunServer(FLAGS_port, ready_ids[0].name, std::move(manager));
```

Version Management

CreateSingleTFModelManagerFromBasePath

SessionBundleSourceAdapter – SessionBundle Creater for each found Model in filesystem.

Dynamic Manager.

Tutorial - Advanced

Batch Scheduler

```
void MnistServiceImpl::Classify(CallData* calldata) {
  ...
  std::unique_ptr<Task> task(new Task(calldata));
  tensorflow::Status status = batch_scheduler_->Schedule(std::move(task));
}
```

Task(calldata)

Batch Scheduler

Test and Run the Server

Export Trained Models

Clear the export directory if it already exists:

```
$>rm -rf /tmp/mnist_model
```

Train (with 100 iterations) and export the first version of model:

```
$>bazel build //tensorflow_serving/example:mnist_export
$>bazel-bin/tensorflow_serving/example/mnist_export
--training_iteration=100 --export_version=1 /tmp/mnist_model
```

Train (with 2000 iterations) and export the second version of model:

```
$>bazel-bin/tensorflow_serving/example/mnist_export
--training_iteration=2000 --export_version=2 /tmp/mnist_model
```

Test and Run the Server

Run Server

```
$>mkdir /tmp/monitored
$>cp -r /tmp/mnist_model/00000001 /tmp/monitored
$>bazel build //tensorflow_serving/example:mnist_inference_2
$>bazel-bin/tensorflow_serving/example/mnist_inference_2 --port=9000 /tmp/monitored
```

Model 1 (100 iteration)

```
$>bazel build //tensorflow_serving/example:mnist_client
$>bazel-bin/tensorflow_serving/example/mnist_client --num_tests=1000 --server=localhost:9000
--concurrency=10
...
Inference error rate: 13.1%
```

Model 2 (1000 iteration)

```
$>cp -r /tmp/mnist_model/00000002 /tmp/monitored
$>bazel-bin/tensorflow_serving/example/mnist_client --num_tests=1000 --server=localhost:9000
--concurrency=10
...
Inference error rate: 9.5%
```