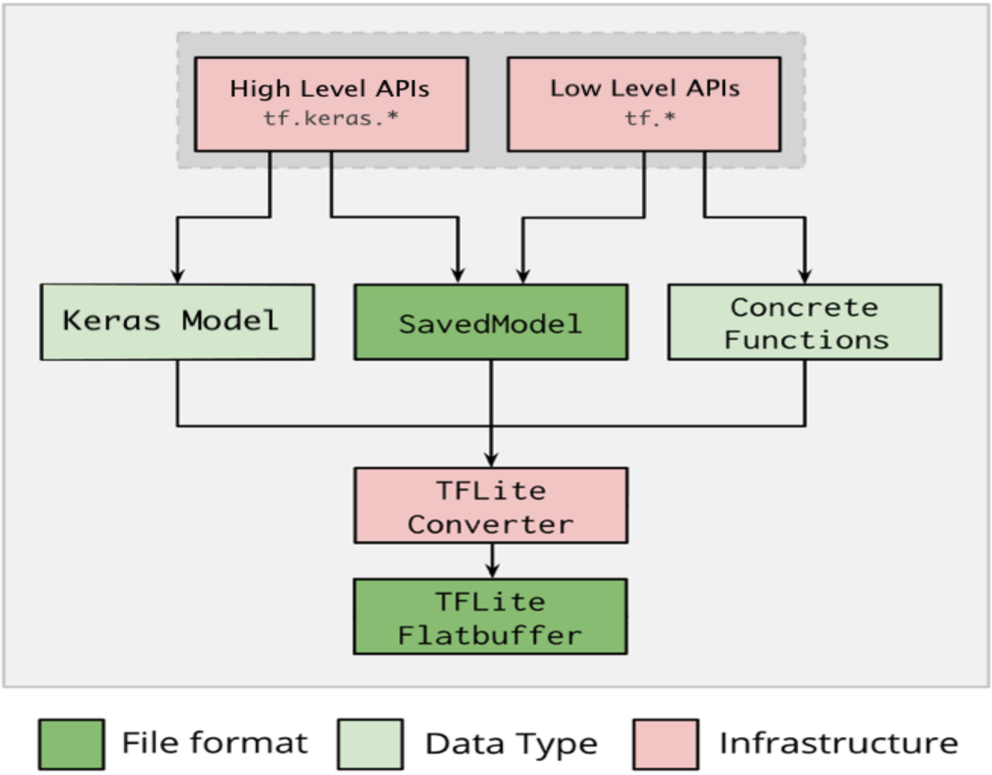
TensorFlow Lite converter:

The TensorFlow Lite converter takes a TensorFlow model and generates a TensorFlow Lite model (an optimized FlatBuffer format identified by the .tflite file extension). You have the following two options for using the converter:

Python API (recommended): This makes it easier to convert models as part of the model development pipeline, apply optimizations, add metadata and has many more features.
Command line: This only supports basic model conversion.



Python API

Helper code: To identify the installed TensorFlow version, run print(tf.__version__) and to learn more about the TensorFlow Lite converter API, run print(help(tf.lite.TFLiteConverter)).

If you've installed TensorFlow 2.x, you have the following two options: (*if you've installed TensorFlow 1.x, refer to Github*)

- Convert a TensorFlow 2.x model using tf.lite.TFLiteConverter. A TensorFlow 2.x model is stored using the SavedModel format and is generated either using the high-level tf.keras.* APIs (a Keras model) or the low-level tf.* APIs (from which you generate concrete functions). As a result, you have the following three options (examples are in the next few sections):
    - tf.lite.TFLiteConverter.from_saved_model() (**recommended**): Converts a SavedModel.
    - tf.lite.TFLiteConverter.from_keras_model(): Converts a Keras model.
    - tf.lite.TFLiteConverter.from_concrete_functions(): Converts concrete functions.
- Convert a TensorFlow 1.x model using tf.compat.v1.lite.TFLiteConverter (examples are on Github):
    - tf.compat.v1.lite.TFLiteConverter.from_saved_model(): Converts a SavedModel.
    - tf.compat.v1.lite.TFLiteConverter.from_keras_model_file(): Converts a Keras model.
    - tf.compat.v1.lite.TFLiteConverter.from_session(): Converts a GraphDef from a session.
    - tf.compat.v1.lite.TFLiteConverter.from_frozen_graph(): Converts a Frozen GraphDef from a file. If you have checkpoints, then first convert it to a Frozen GraphDef file and then use this API as shown here.
    -

Convert a SavedModel (recommended)

The following example shows how to convert a SavedModel into a TensorFlow Lite model.

```
import tensorflow as tf

# Convert the model
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir) # path to the SavedModel directory
tflite_model = converter.convert()
```

```
# Save the model.
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```

Convert a Keras model

The following example shows how to convert a Keras model into a TensorFlow Lite model.

```
import tensorflow as tf

# Create a model using high-level tf.keras.* APIs
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(units=1, input_shape=[1]),
    tf.keras.layers.Dense(units=16, activation='relu'),
    tf.keras.layers.Dense(units=1)
])
model.compile(optimizer='sgd', loss='mean_squared_error') # compile the model
model.fit(x=[-1, 0, 1], y=[-3, -1, 1], epochs=5) # train the model
# (to generate a SavedModel) tf.saved_model.save(model, "saved_model_keras_dir")

# Convert the model.
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the model.
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```

Convert concrete functions

The following example shows how to convert concrete functions into a TensorFlow Lite model.

```
import tensorflow as tf

# Create a model using low-level tf.* APIs
class Squared(tf.Module):
  @tf.function
  def __call__(self, x):
    return tf.square(x)
model = Squared()
```

```
# (ro run your model) result = Squared(5.0) # This prints "25.0"
# (to generate a SavedModel) tf.saved_model.save(model, "saved_model_tf_dir")
concrete_func = model.__call__.get_concrete_function()

# Convert the model
converter = tf.lite.TFLiteConverter.from_concrete_functions([concrete_func])
tflite_model = converter.convert()

# Save the model.
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```