

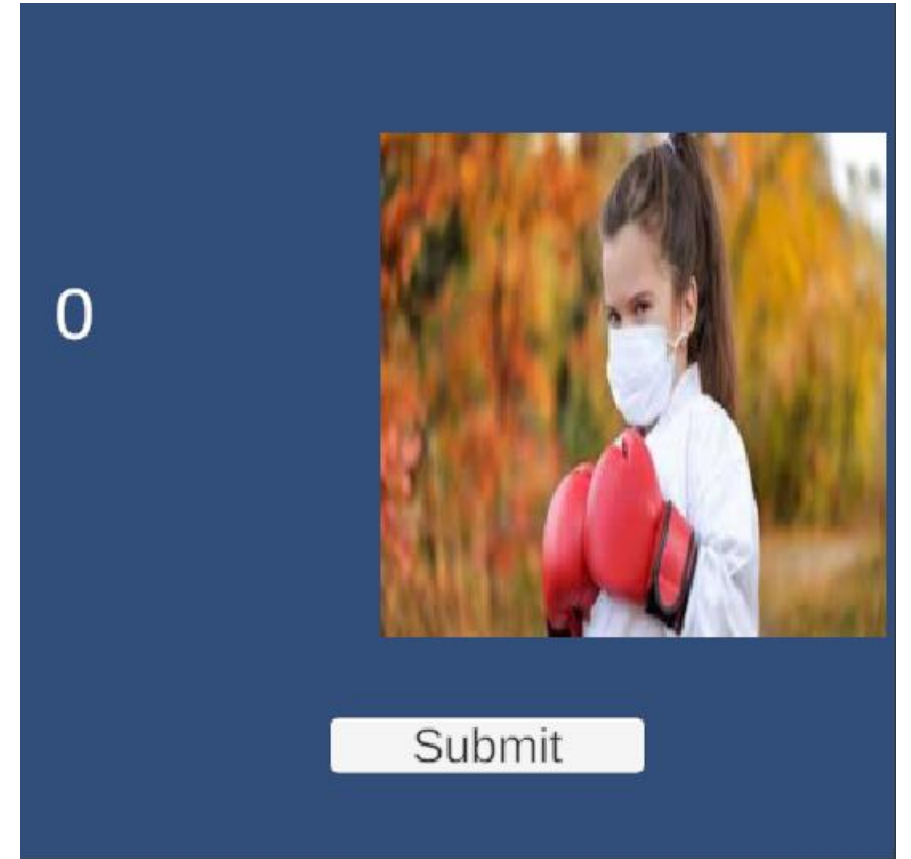
# NEW TECHNOLOGY IN IT APPLICATION DEVELOPMENT

**MQTT in Unity**

# Create Interface

---

- UI > Text
- UI > Button
- UI > RawImage
- Insert an image into RawImage.Texture



# Count Event

---

```
using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
using UnityEngine.UI;  
using TMPro;
```



```
public RawImage imgFace;  
public TMP_Text txtCount;
```

# Count Event

---

```
void Start()  
{  
    count = 0;  
}  
public void OnbtCountClick()  
{  
    count = count + 1;  
    txtCount.text = count.ToString();  
}  
void UpdateUI()  
{  
    txtCount.text = count.ToString();  
    UpdateUI();  
}
```

# MQTT

---

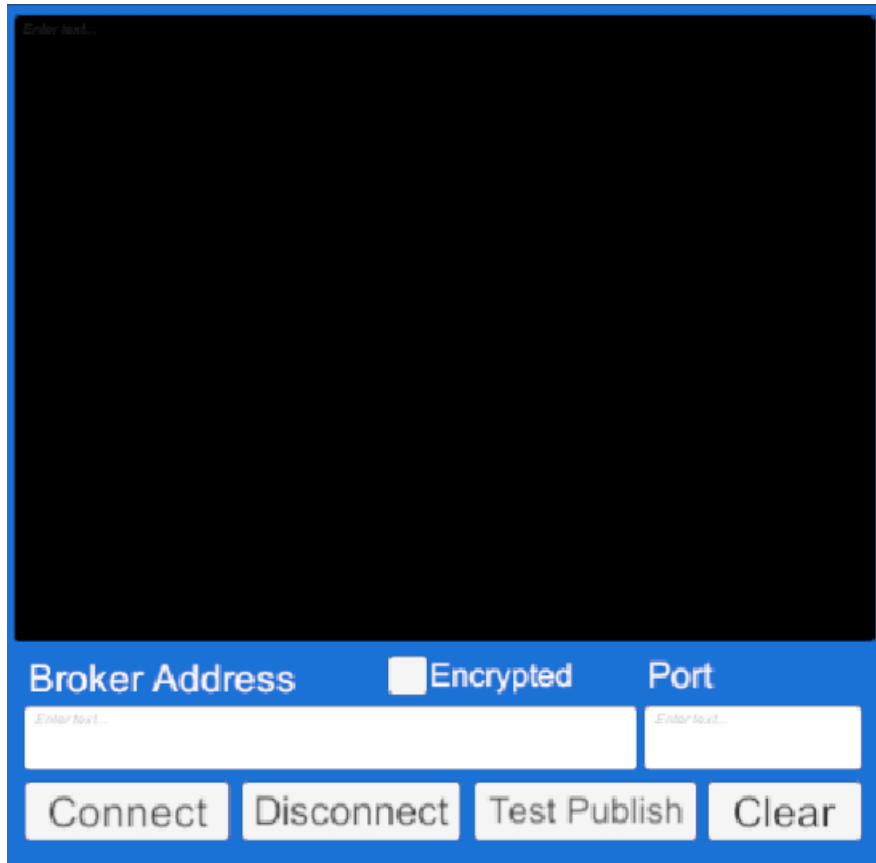
```
using uPLibrary.Networking.M2Mqtt;  
using uPLibrary.Networking.M2Mqtt.Messages;  
using M2MqttUnity;
```

```
public class MqttImage : M2MqttUnityClient  
{  
  
}
```

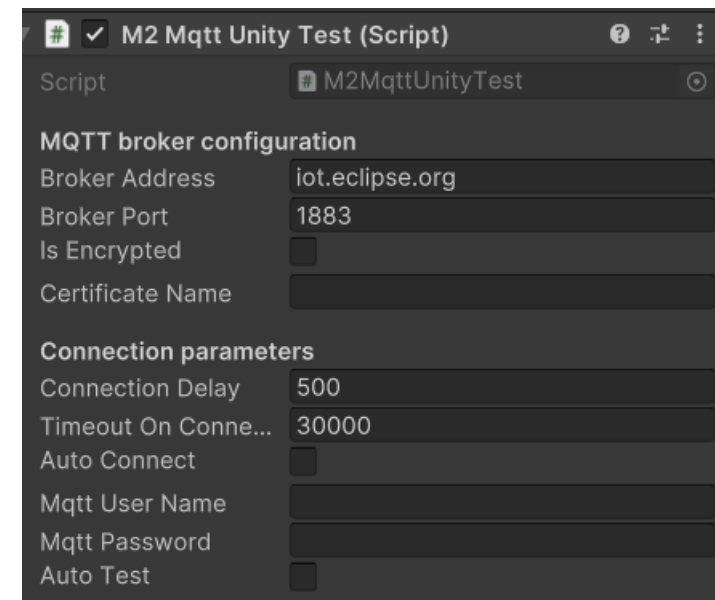


# M2MqttUnity Library lib

---



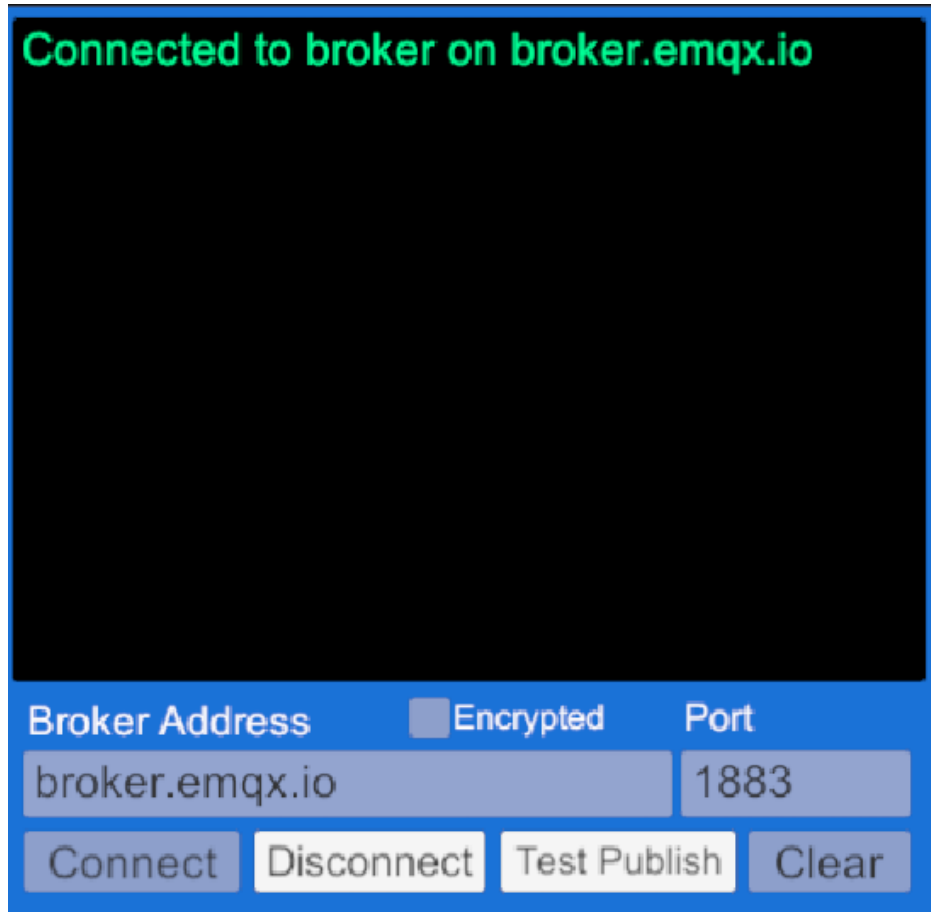
The screenshot shows a graphical user interface for the M2MqttUnity library. It features a large black rectangular area at the top, likely for displaying logs or messages. Below this area, there are input fields for 'Broker Address' and 'Port', each with a placeholder text 'Enter text...'. A checkbox labeled 'Encrypted' is positioned between the two input fields. At the bottom of the interface, there are four buttons: 'Connect', 'Disconnect', 'Test Publish', and 'Clear'.



The screenshot displays the configuration window for the 'M2 Mqtt Unity Test (Script)'. The window title is 'M2 Mqtt Unity Test (Script)' and it contains a section for 'MQTT broker configuration' and 'Connection parameters'. The 'MQTT broker configuration' section includes fields for 'Broker Address' (set to 'iot.eclipse.org'), 'Broker Port' (set to '1883'), 'Is Encrypted' (checkbox), and 'Certificate Name'. The 'Connection parameters' section includes fields for 'Connection Delay' (set to '500'), 'Timeout On Conne...' (set to '30000'), 'Auto Connect' (checkbox), 'Mqtt User Name', 'Mqtt Password', and 'Auto Test' (checkbox).

# M2MqttUnity Library lib

---


















broker.emqx.io

MQTT broker configuration	
Broker Address	broker.emqx.io
Broker Port	1883
Is Encrypted	<input type="checkbox"/>
Certificate Name	

# M2MqttUnity Library lib

---

User Interface		
Console Input Field	 ConsoleInputField (Input Field)	
Encrypted Toggle	 ToggleEncrypted (Toggle)	
Address Input Field	 AddressInputField (Input Field)	
Port Input Field	 PortInputField (Input Field)	
Connect Button	 ConnectButton (Button)	
Disconnect Button	 DisconnectButton (Button)	
Test Publish Button	 TestPublishButton (Button)	
Clear Button	 ClearButton (Button)	

```
[Tooltip("Set this to true to perform a testing cycle  
automatically on startup")]  
public bool autoTest = false;  
[Header("User Interface")]  
public InputField consoleInputField;  
public Toggle encryptedToggle;  
public InputField addressInputField;  
public InputField portInputField;  
public Button connectButton;  
public Button disconnectButton;  
public Button testPublishButton;  
public Button clearButton;
```



# M2MqttUnity's Event

---

- `base.OnConnecting();`
- `base.OnConnected();`
- `base.Start();`
- `base.Update();`

# Method

---

- protected override void OnConnecting()
- protected override void OnConnected()
- protected override void Start()
- protected override void Update();
- protected override void OnDisconnected()
- protected override void OnConnectionLost()
- protected override void DecodeMessage(string topic, byte[] message)
- protected override void SubscribeTopics()
- protected override void UnsubscribeTopics()

# Events

---

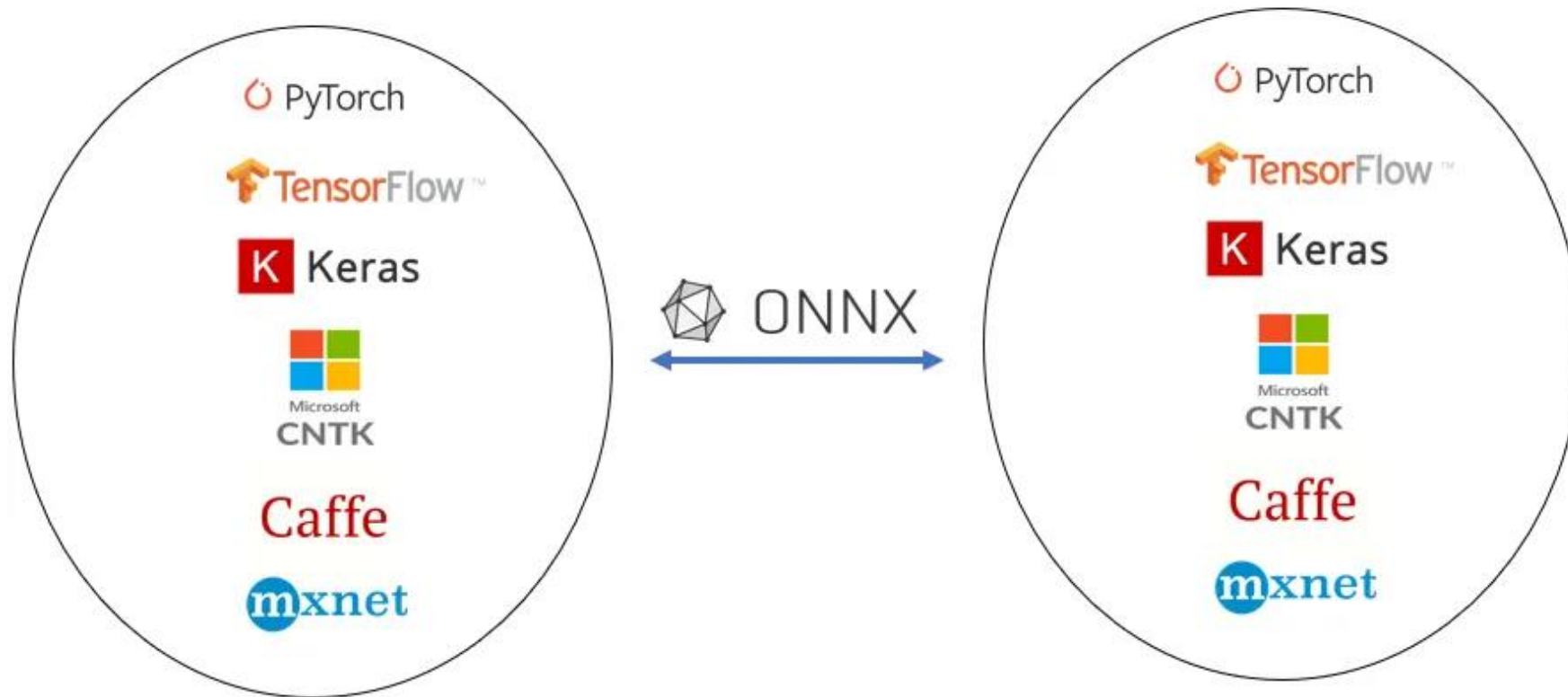
```
protected override void OnConnectionFailed(string errorMessage)
{
    Debug.Log("CONNECTION FAILED! " + errorMessage);
}
```

```
protected override void OnConnectionLost()
{
    Debug.Log("CONNECTION LOST!");
}
```

```
protected override void OnDisconnected()
{
    Debug.Log("Disconnected.");
}
```

# Open Neural Network Exchange

---



# Open Neural Network Exchange

---

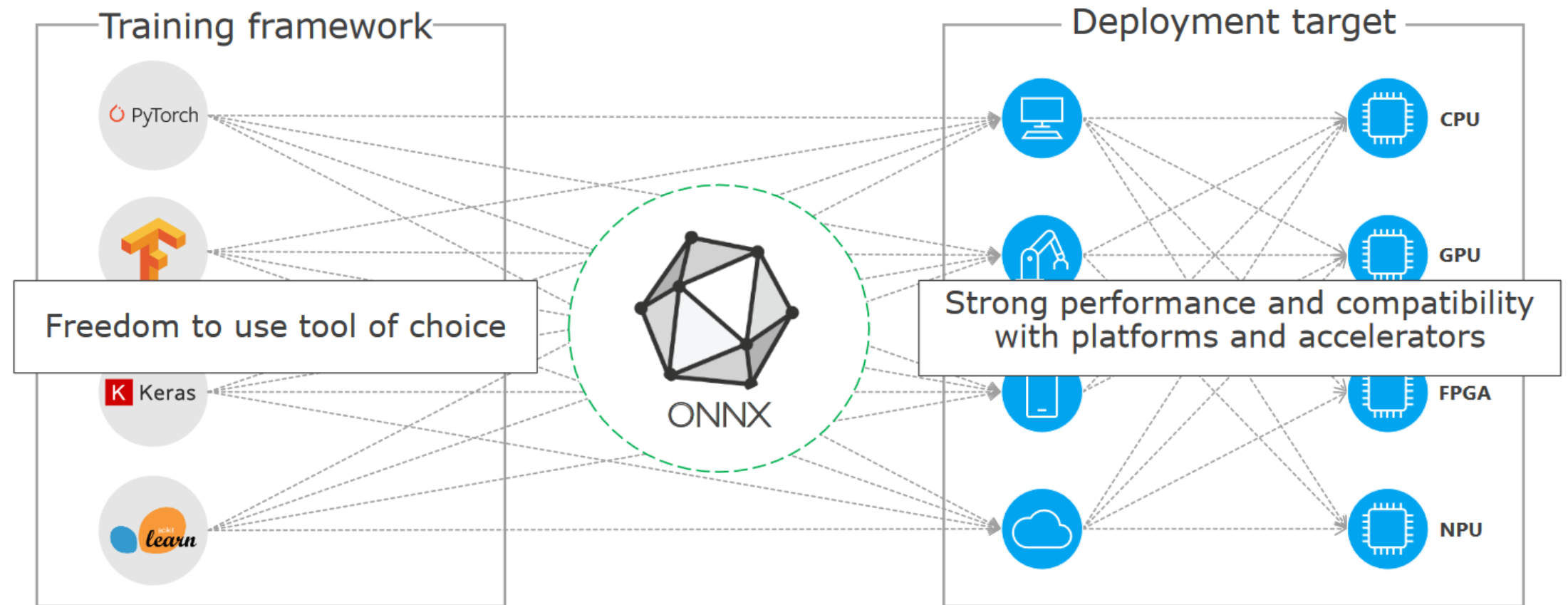


ONNX

The three main tasks of ONNX can be listed as follows:

1. Convert the model from any framework to ONNX format
2. Convert ONNX format to any desired framework
3. Faster inference with ONNX model on supported runtime engines

# Bridge



# Tensorflow to Onnx

---

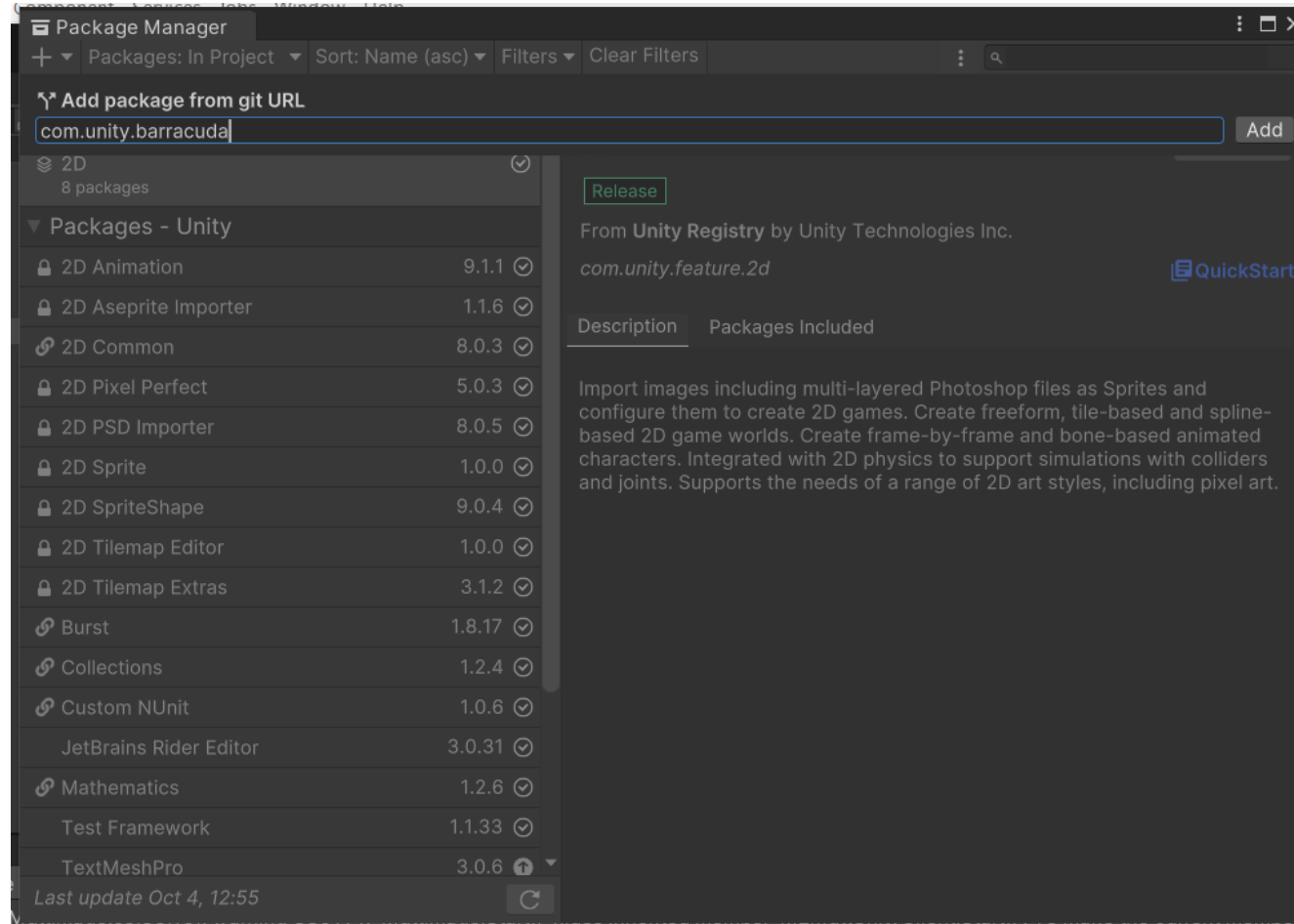
```
import numpy as np
import tensorflow as tf
import tf2onnx
# print(tf.version.VERSION)
mask_model = tf.keras.models.load_model('kerasModel.h5')

spec = (tf.TensorSpec((None, 224, 224, 3), tf.float32, name="input"),)
output_path = "mask" + ".onnx"

model_proto, _ = tf2onnx.convert.from_keras(mask_model,
input_signature=spec, opset=13, output_path=output_path)
output_names = [n.name for n in model_proto.graph.output]
```

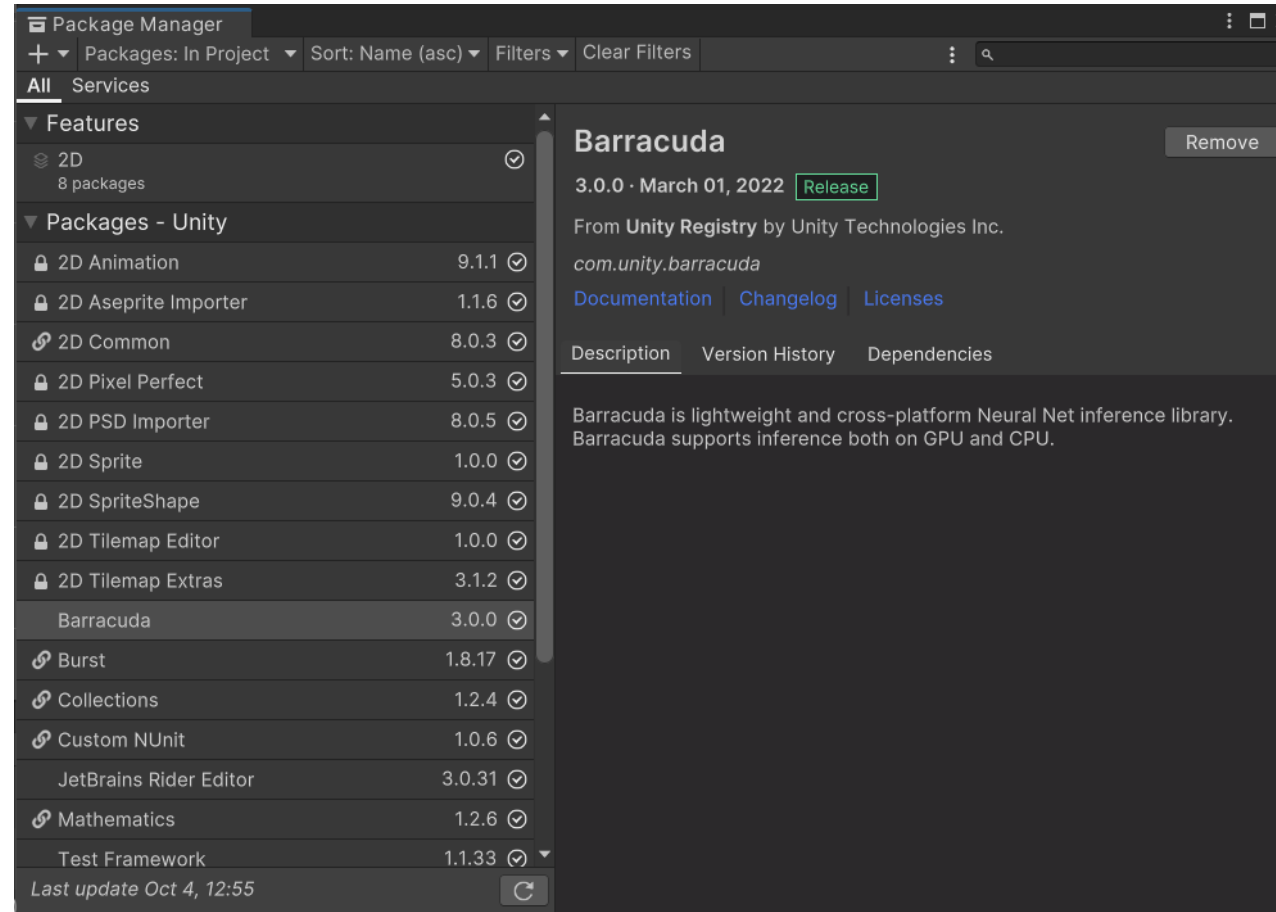
# Install Barracuda

com.unity.barracuda





# Install Barracuda



# Barracuda Lib

---

- `using Unity.Barracuda;`

```
public NNModel modelAsset;  
private Model m_RuntimeModel;  
private IWorker m_Worker;
```

```
void Awake()  
{  
    m_RuntimeModel = ModelLoader.Load(modelAsset);  
    m_Worker = WorkerFactory.CreateWorker(WorkerFactory.Type.CSharp, m_RuntimeModel);  
}
```

# Feeds

---

```
private int count, channels;  
private string[] feeds = {"pump", "masked"};  
private string[] classnames = {"MASKED FACE", "UNMASKED FACE", "NOBODY"};
```

# Message

---

```
protected override void DecodeMessage(string topic, byte[] message)
{
    string msg = System.Text.Encoding.UTF8.GetString(message);
    Debug.Log("Received: " + msg);

    if("pump" == topic){
        count = Int32.Parse(msg);
    }
}
```

# Prediction

---

```
string PredictImage()
{
    Tensor input = new Tensor(imgFace.texture, channels);
    m_Worker.Execute(input);

    Tensor output = m_Worker.PeekOutput();

    int[] maxIndex = output.ArgMax();

    return label[maxIndex[0]];
}
```

# Design and Implement

---

