



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile

Q&A WITH SPEAKER [SPEAKER BIO](#)

Yi-Hong Wang  
Software Developer  
IBM

YiHong is a software developer with the Cognitive OpenTech Group at the IBM Silicon Valley Lab. He has been an active contributor in the TensorFlow community for the past 2 years. He is also a committer in the node.JS community.



Va Barbosa  
Software Developer  
IBM

Va is a software developer with the Developer Advocacy Group at IBM. He has been an active contributor in the TensorFlow community for the past 2 years.

VIDEO

# IoT Challenges

Variety of edge devices

Quantity of edge devices

Different scale

Edge devices restrictions





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile

Q&A WITH SPEAKER [SPEAKER BIO](#)

Yi-Hong Wang  
Software Developer  
IBM

YiHong is a software developer with the Cognitive OpenTech Group at the IBM Silicon Valley Lab. He has been an active contributor in the TensorFlow community for the past 2 years. He is also a committer in the node.JS community.



Va Barbosa  
Software Developer  
IBM

Va is a software developer with the Developer Advocacy Group at IBM. He has been an active contributor in the TensorFlow community for the past 2 years.

VIDEO

Node-RED is an open source [flow-based programming tool](#) [wiring together IoT devices](#)

<https://nodered.org>





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

Q&A WITH SPEAKER [SPEAKER BIO](#)

**Yi-Hong Wang**  
Software Developer  
IBM

YiHong is a software developer with the Cognitive OpenTech Group at the IBM Silicon Valley Lab. He has been an active contributor in the TensorFlow community for the past 2 years. He is also a committer in the node.JS community.



**Va Barbosa**  
Software Developer  
IBM

Va is a software developer with the Developer Advocacy Group at IBM. He has been an active contributor in the TensorFlow community for the past 2 years.

VIDEO

The screenshot shows the Node-RED interface with a flow diagram titled 'Flow 1'. The flow consists of three parallel streams. Stream 1 starts with an 'HTTP in' node, followed by a 'JSON parse' node, and ends with a 'Log' node. Stream 2 starts with an 'HTTP in' node, followed by a 'JSON parse' node, and ends with a 'Log' node. Stream 3 starts with an 'HTTP in' node, followed by a 'JSON parse' node, and ends with a 'Log' node. The interface includes a sidebar with categories like 'Variables', 'Functions', and 'Messages'. A right-hand panel displays 'Information' about a selected node, 'Description', 'Mode Help', and 'Details'.

[https://www.youtube.com/watch?v=\\_ST7fiBLIw](https://www.youtube.com/watch?v=_ST7fiBLIw)



Be Sure to Visit the Sponsor Showcase!

powered by **Intrado**



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

Q&A WITH SPEAKER [SPEAKER BIO](#)

**Yi-Hong Wang**  
Software Developer  
IBM

YiHong is a software developer with the Cognitive OpenTech Group at the IBM Silicon Valley Lab. He has been an active contributor in the TensorFlow community for the past 2 years. He is also a committer in the node.js community.



**Va Barbosa**  
Software Developer  
IBM

Va is a software developer with the Developer Advocacy Group at IBM. He has been an active contributor in the TensorFlow community for the past 2 years.

VIDEO

The screenshot shows a Node-RED interface with a flow titled "Flow 1". The flow consists of two parallel sequences. Each sequence starts with an "input" node, followed by a "function" node, and ends with a "log" node. The two parallel flows converge at a "join" node, which then connects to another "log" node. The left sidebar contains categories for "functions" (with "input" selected) and "nodes" (including "function", "script", "debug", "change", and "join"). The right sidebar provides details for the selected "input" node, including its ID ("00000000-0000-0000-0000-000000000000"), type ("input"), and mode ("manual"). It also includes sections for "Description", "Outputs", "Properties", and "Details".



[https://www.youtube.com/watch?v=\\_ST7fiBLLtw](https://www.youtube.com/watch?v=_ST7fiBLLtw)



Be Sure to Visit the Sponsor Showcase!

powered by Intrado



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

Q&A WITH SPEAKER [SPEAKER BIO](#)

**Yi-Hong Wang**  
Software Developer  
IBM

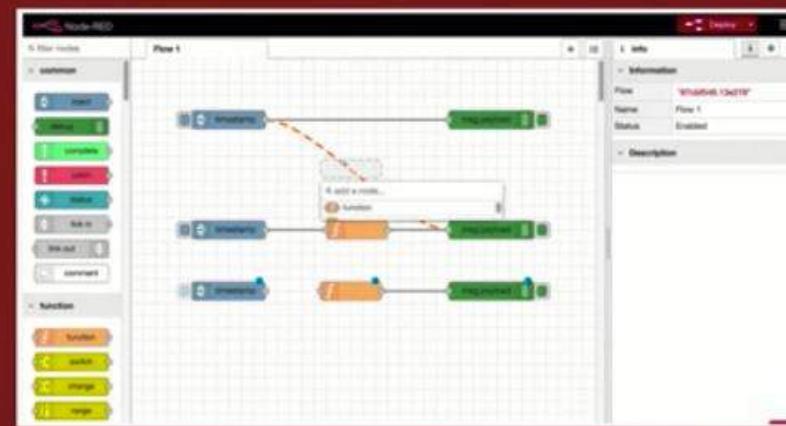
YiHong is a software developer with the Cognitive OpenTech Group at the IBM Silicon Valley Lab. He has been an active contributor in the TensorFlow community for the past 2 years. He is also a committer in the node.JS community.



**Va Barbosa**  
Software Developer  
IBM

Va is a software developer with the Developer Advocacy Group at IBM. He has been an active contributor in the TensorFlow community for the past 2 years.

VIDEO



[https://www.youtube.com/watch?v=\\_ST7fiBLJlw](https://www.youtube.com/watch?v=_ST7fiBLJlw)





# THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

Q&A WITH SPEAKER SPEAKER BIO

VIDEO

The screenshot shows a web browser displaying a Node-RED dashboard titled "weather". The dashboard features a grid of cards, each representing a different Node-RED node or flow. The cards include:

- node-red-contrib-aws-awsiot: Node-RED 773 with Device and Amazon PI IoT as MQTT and AWS IoT core input and output. This card shows how to connect and interact with AWS IoT Core.
- node-red-contrib-openweathermap: A Node-RED node that gets the weather report from openweathermap.org.
- OpenWeatherMap Weather Dashboard with Icons - Portrait: This flow pulls weather data from OpenWeatherMap (OWM) then presents it in a portrait orientation.
- node-red-contrib-dutch-weather: Dutch weather info & news automation for use with Node-RED.
- last Dow weather: last to learn simple weather flow.
- node-red-contrib-two-cards2-local: Node-RED nodes for The Weather Company (OWM) & Dutch Weather Station API.
- node-red-contrib-local-weather: Node-RED node for retrieve local weather forecast from TWC (Commerce).
- Get weather forecast from aglmetr.net (AKA yrcast): Please change "test geolocation" mode according your location.
- Weather Services and Devices: A collection of Node-RED nodes and flows that use Weather services.
- NordVPN Weather Apple HomeKit: Shows how to get weather data from NordVPN and send it to Apple HomeKit.
- Service Weather Alert Map: A Node-RED Service Weather Alert Map.

At the bottom of the dashboard, there is a footer with links to GitHub, API, Flow Library, About, Documentation, Code of Conduct, Community, Blog, Twitter, Forum, and Slack.

Enter question here...

NEW QUESTION



Share Your Experience on social - tag posts with #OSSummit #LFELC

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows the Node-RED interface running in a browser window. On the left, there is a sidebar with categories like 'common', 'function', and 'network'. A flow diagram titled 'Flow 1' is visible, consisting of a 'timestamp' node connected to a 'msg.payload' node. A success message box says 'Successfully deployed' and 'You have some unused configuration nodes. Click here to see them.' The URL in the address bar is 127.0.0.1:1880?flow=4d277aa5048704.

[Connect with attendees on the OSS+ELC Slack Channel!](#)

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

VIDEO

The screenshot shows the Node-RED interface running in a browser window. The left sidebar contains a library of nodes categorized into common, function, and network types. A flow titled 'Flow 1' is visible, starting with an 'inject' node followed by a 'Timestamp' node and a 'msg.payload' node. The right panel displays a 'Debug' tab showing the payload of the last message sent, which is a number: 1582529942572.



Take a break! Enjoy our Experiences + Fun &amp; Games!

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

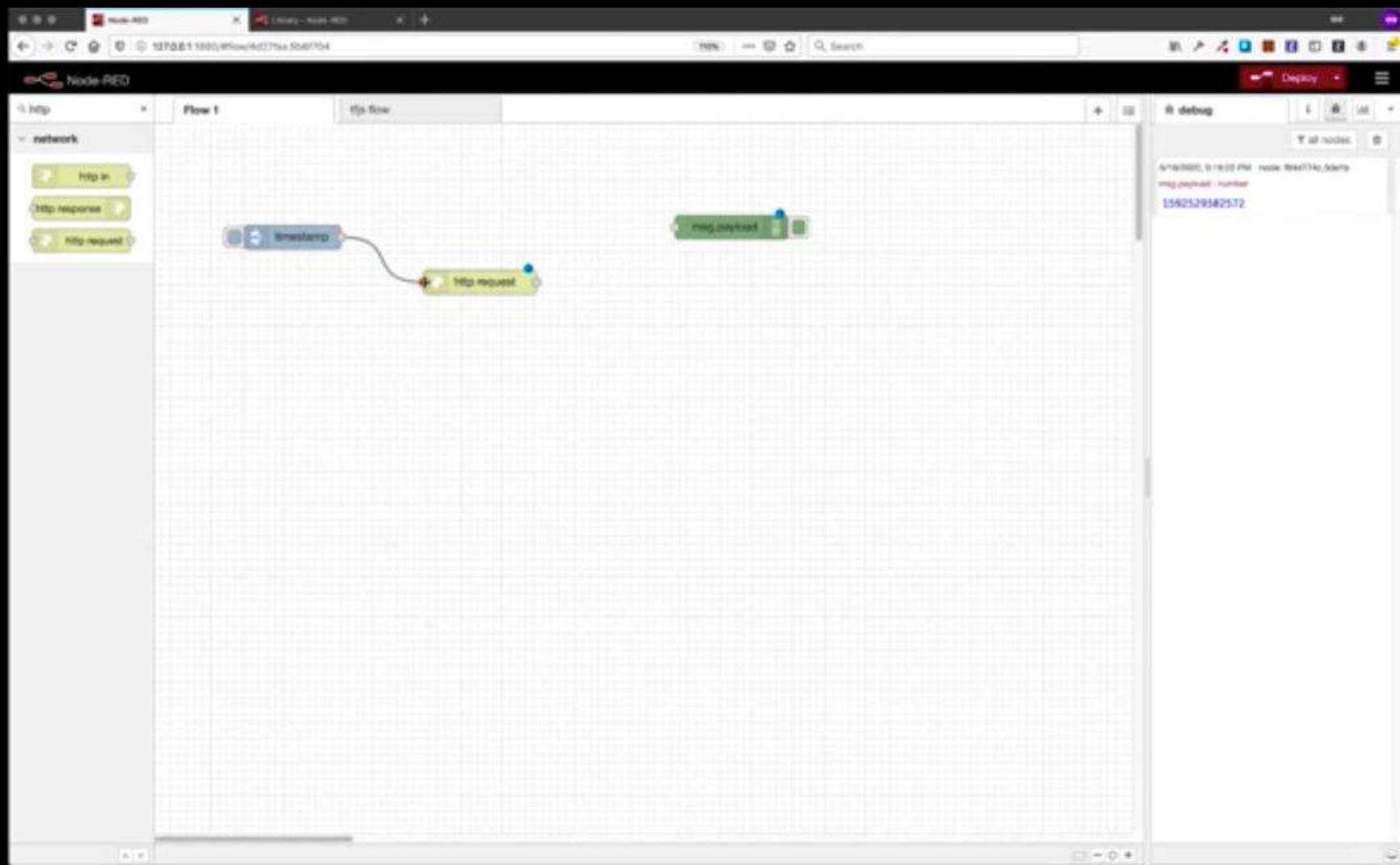
Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO





# THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED interface with a flow editor on the left and a configuration dialog on the right.

**Flow Editor:** The flow consists of three nodes: "Http in", "Browsing", and "Http request". The "Http in" node has a single output wire connecting to the "Browsing" node. The "Browsing" node has two output wires: one connecting to the "Http request" node, and another wire that loops back to the "Http in" node. The "Http request" node has a single output wire.

**Edit inject node Dialog:**

- Properties:** The payload is set to "Timestamp".
- Topic:** An empty text input field.
- Inject once after:** A dropdown menu showing "0.1 seconds, then".
- Repeat:** A dropdown menu showing "none".
- Name:** An empty text input field.

**Note:** "Interval between times" and "at a specific time" will use cron. "Interval" should be 596 hours or less. See info box for details.





# THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

VIDEO

The screenshot shows the Node-RED interface running in a browser window. The URL in the address bar is `127.0.0.1:1880/node/4d227fba5048704`. The main canvas displays a flow titled "Flow 1" with the following nodes connected sequentially:

- HTTP in
- HTTP response
- HTTP request
- Timestamp
- HTTP request

A modal dialog titled "Edit inject node" is open on the right side of the screen. It contains the following configuration options:

- Properties**:
  - Payload: `% 10`
  - Topic:
  - Inject once after `0.1` seconds, then
  - Repeat: `none`
  - Name: `Name`
- Note: "Interval between times" and "at a specific time" will use cron. "Interval" should be 596 hours or less. See info box for details.

On the far right, there is a "debug" panel showing the message `2019-05-09T16:00:00Z | r=0.00 Plat: node/4d227fba5048704 msg.payload -> number 1582529342532`.





# THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED interface with a flow titled "Flow 1". The flow consists of three nodes: an "Http in" node, a "Topic" node, and an "Http request" node. The "Http in" node has a single output wire connecting to the "Topic" node. The "Topic" node has a single output wire connecting to the "Http request" node. The "Http request" node is currently selected, opening a modal dialog titled "Edit http request node". The modal contains the following configuration:

- Properties**:
  - Method: GET
  - URL: `http://numbersapi.com/{payload[]}`
  - Append msg.payload as query string parameters
  - Enable secure (SSL/TLS) connection
  - Use authentication
  - Enable connection keep-alive
  - Use proxy
- Return**: a UTF-8 string
- Name**: Name

On the right side of the interface, there is a "debug" pane showing the message `msg.payload - number` and the value `1582529542572`.





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

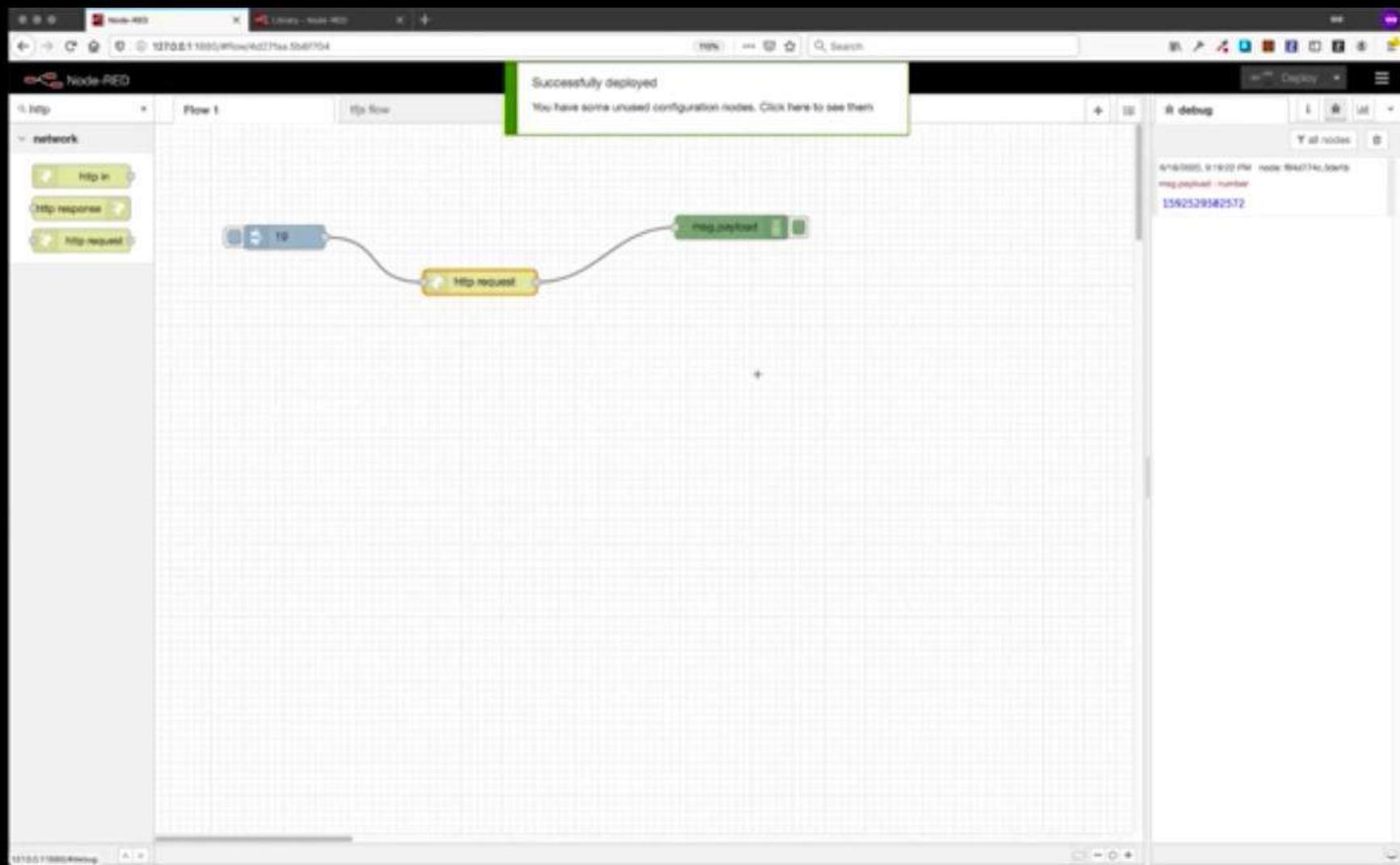
Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO



Questions? Ask the Events team on the #1-helpdesk Slack channel

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

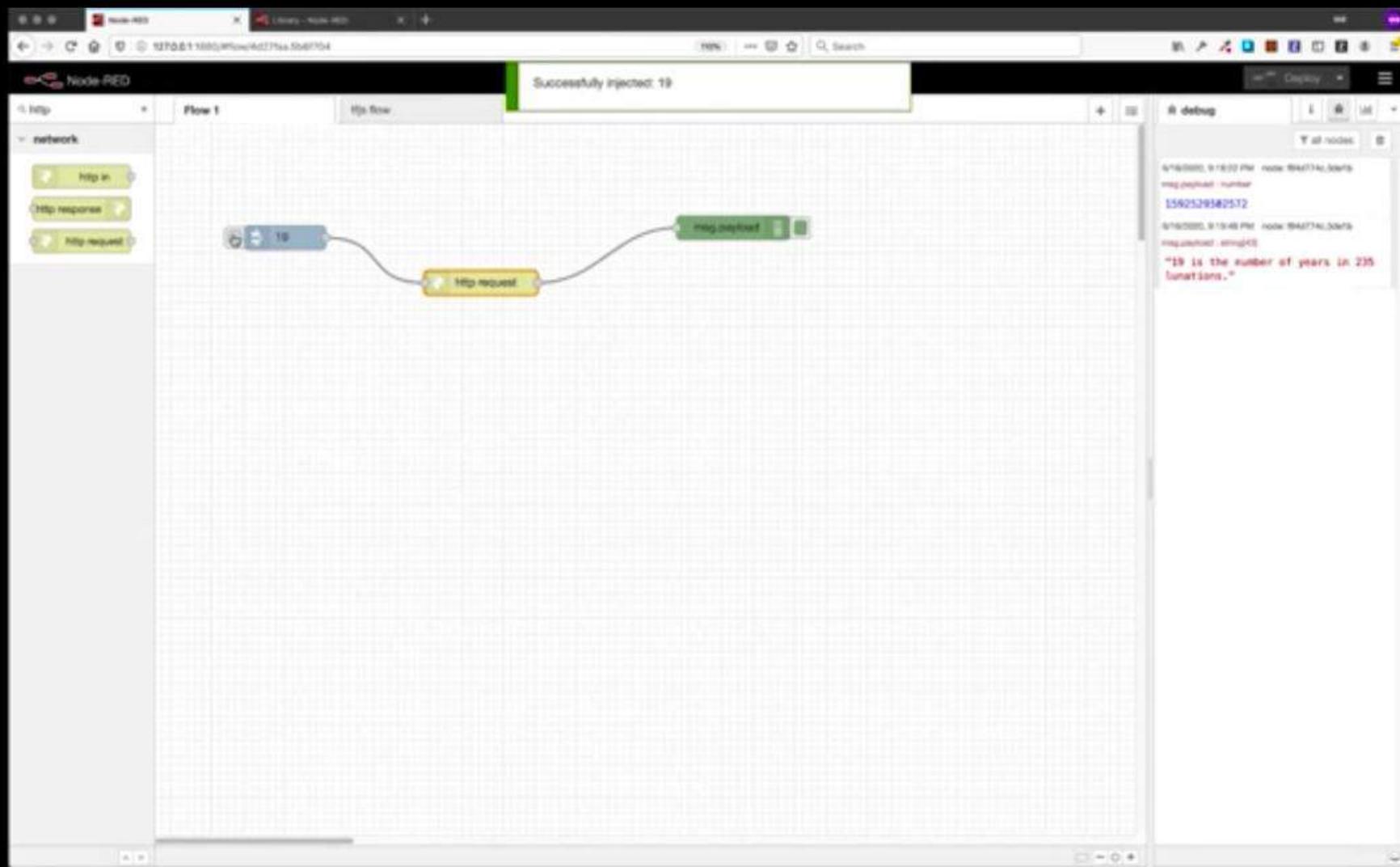
Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED interface with a flow titled "Flow 1". The flow consists of three main nodes: an "Http in" node, an "Http request" node, and a "msg.payload" output node. The "Http in" node has a single output wire that connects to the "Http request" node. The "Http request" node has a single output wire that connects to the "msg.payload" node. The "msg.payload" node has a green terminal where the output is displayed. On the right side of the interface, there is a "R debug" panel showing a log of messages. The log contains several entries, each with a timestamp, node ID, and message payload. The payloads are mostly identical, showing calculations related to leap years and the number 19.

```
0/18/2020, 9:19:00 PM node: 9b4d774c3aef9  
msg.payload: number  
1582529342572  
0/18/2020, 9:19:48 PM node: 9b4d774c3aef9  
requested: strng[4]  
"19 is the number of years in 235 lunations."  
0/18/2020, 9:19:57 PM node: 9b4d774c3aef9  
requested: strng[4]  
"19 is the number of years in 235 lunations."  
0/18/2020, 9:19:58 PM node: 9b4d774c3aef9  
requested: strng[4]  
"19 is the final year a person is a teenager."  
0/18/2020, 9:19:59 PM node: 9b4d774c3aef9  
requested: strng[4]
```

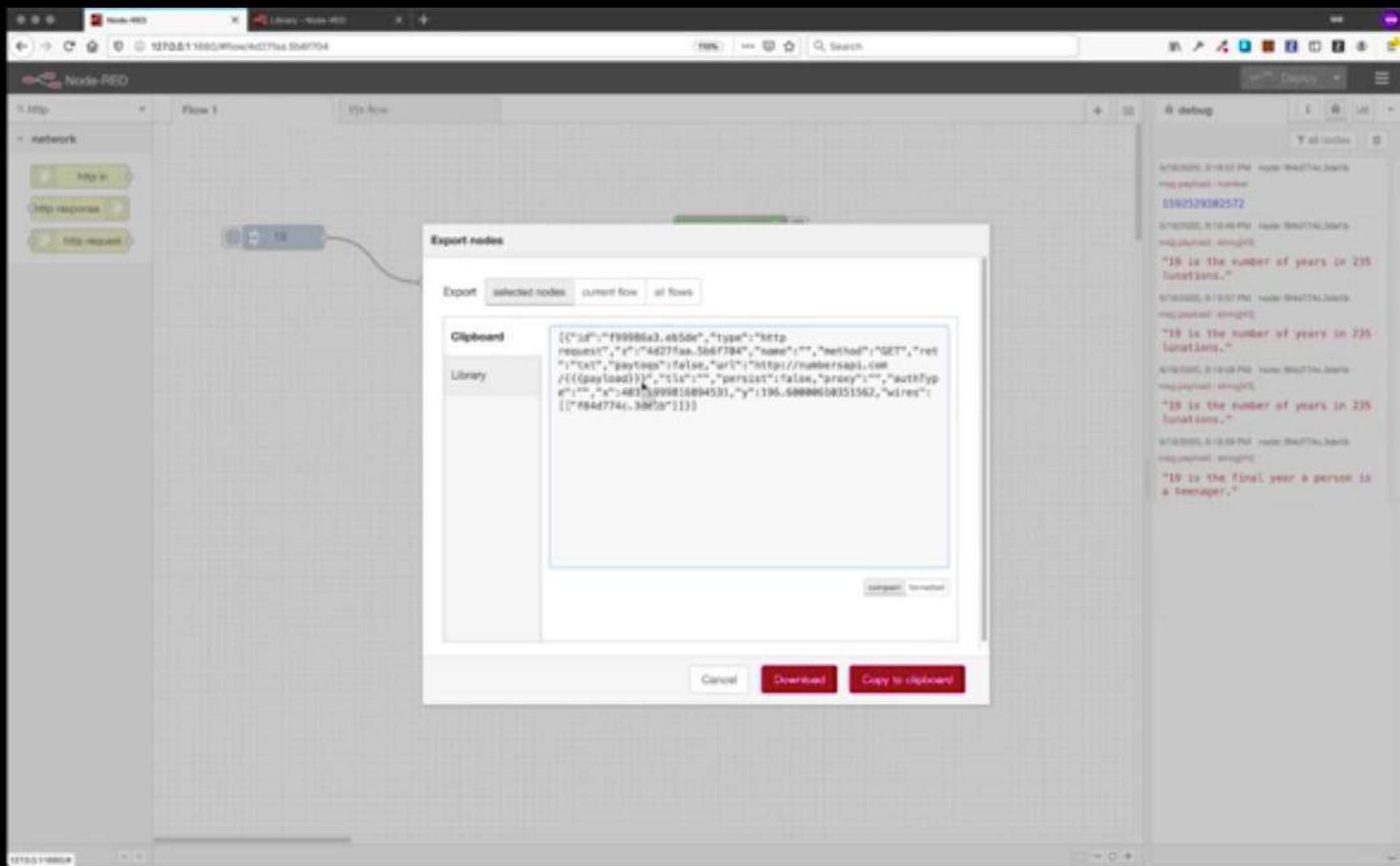


Be Sure to Visit the Sponsor Showcase!

powered by



VIDEO





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED interface running in a browser. The main area displays a flow titled "Flow 1" with several nodes connected by wires. One node is highlighted with a yellow border. A modal dialog box titled "Export nodes" is open in the foreground, containing a JSON representation of a selected node. The JSON is as follows:

```
{
  "id": "f99998e3-eb5de",
  "type": "http request",
  "z": "4d21faa5b6f784",
  "name": "",
  "method": "GET",
  "ret": "txt",
  "payto": false,
  "url": "https://membersapi.com/{{payload}}",
  "tls": "",
  "persist": false,
  "proxy": "",
  "authType": "",
  "x": 483,
  "y": 196,
  "wires": []
}
```

Below the JSON, there are three buttons: "Cancel", "Download", and "Copy to clipboard".

In the background, a terminal window shows several log entries from Node.js applications. Some of the logs include:

- 14:50:00, 01/10/20 10:40 PM : node:4d21faa5b6f784:info: payload - number  
1380529582572
- 14:50:00, 01/10/20 10:40 PM : node:4d21faa5b6f784:info: requested: string  
"19 is the number of years in 235 functions."
- 14:50:00, 01/10/20 10:40 PM : node:4d21faa5b6f784:info: payload - string  
"19 is the number of years in 235 functions."
- 14:50:00, 01/10/20 10:40 PM : node:4d21faa5b6f784:info: requested: string  
"19 is the number of years in 235 functions."
- 14:50:00, 01/10/20 10:40 PM : node:4d21faa5b6f784:info: payload - string  
"19 is the final year a person is a teenager."





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



VIDEO

TensorFlow.js is an open source library to **train** and **deploy** machine learning **models** with **JavaScript**

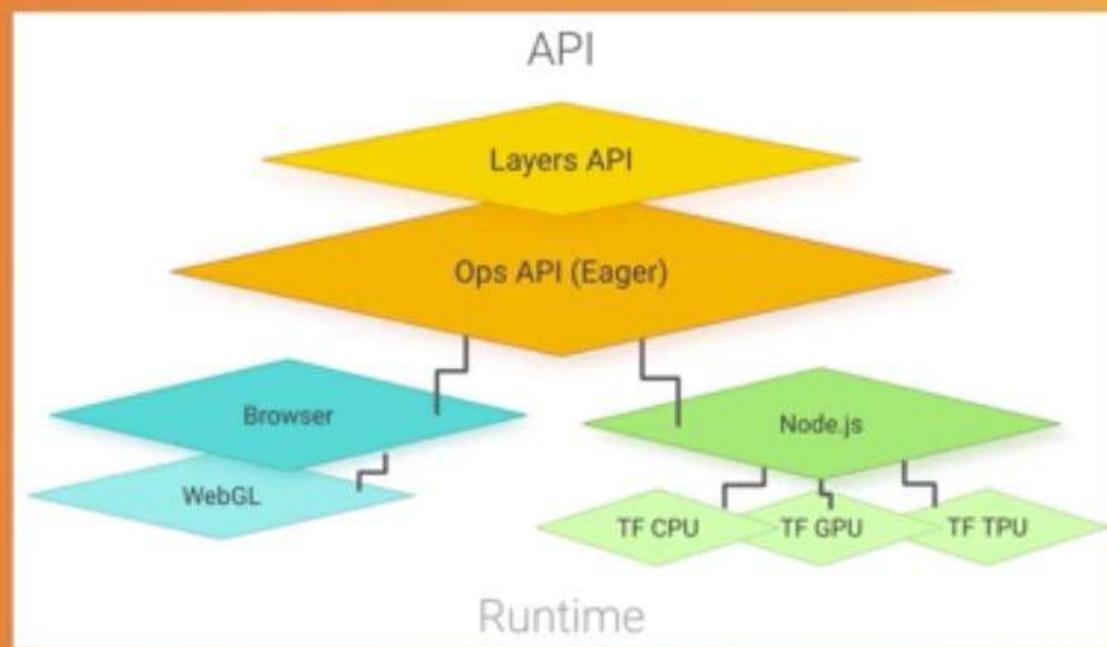
<https://www.tensorflow.org/js>

Connect with attendees on the OSS+ELC Slack Channel!

powered by



VIDEO



<https://arxiv.org/abs/1901.05350>



Take a break! Enjoy our Experiences + Fun & Games!

powered by Intrado



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

## THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows the TensorFlow.js website. At the top, there's a navigation bar with links like 'TensorFlow', 'Install', 'Learn', 'API', 'Resources', 'Community', 'Why TensorFlow', 'Search', 'English', 'GitHub', and 'Sign in'. Below the navigation, there's a section for 'For JavaScript' with links for 'Overview', 'Tutorials', 'Guide', 'Models', 'Demos', and 'API'. The main content area has a large orange header with the text 'TensorFlow.js is a library for machine learning in JavaScript'. Below this, there's a sub-section titled 'How it works' with three icons: a server, a cloud, and a person. Each icon has a corresponding text block: 'Run existing models' (use off-the-shelf JavaScript models or convert Python TensorFlow models), 'Retrain existing models' (retrain pre-existing ML models using your own data), and 'Develop ML with JavaScript' (build and train models directly in JavaScript using flexible and intuitive APIs). There are also sections for 'See tutorials', 'See models', and 'See demos'.



Share Your Experience on social - tag posts with #OSSummit #ELCN  
powered by



The screenshot shows a web browser window with the URL [https://www.tensorflow.org/api\\_docs/python/](https://www.tensorflow.org/api_docs/python/). The page title is "TensorFlow API". The main content is the "Tensors" section of the API documentation. It includes a sidebar with "TENSORS" and a list of tensor creation functions like tf.Tensor, tf.scalar, etc. The main content area has a heading "Tensors" and a sub-section "Tensors / Creation". It describes tensors as the core datastructure of TensorFlow and provides utility functions for common cases like Scalar, 1D, 2D, 3D and 4D tensors, as well as functions to initialize tensors in ways useful for machine learning. A specific function, `tf.tensor`, is highlighted with its parameters: `(values, shape?, dtype?)`. It creates a `tf.Tensor` with the provided values, shape and dtype. Three examples are shown:

```
// Pass an array of values to create a vector.  
tf.tensor([1, 2, 3, 4]).print();
```

```
// Pass a nested array of values to make a matrix or a higher  
// dimensional tensor.  
tf.tensor([[1, 2], [3, 4]]).print();
```

```
// Pass a 2D array and specify a shape, produce 2D.  
tf.tensor([1, 2, 3, 4]).print();
```

Below the examples are "Save" and "Run" buttons. The "Parameters" section lists `values` (Type: `TypedArray`), `shape` (Type: `[number[]]`), and `dtype` (Type: `'float32'|'int32'|'bool'|'complex64'|'string'`). The "Returns" section indicates a `tf.Tensor`.





The screenshot shows a web browser window with the URL <https://www.tensorflow.org/api/api.html>. The page is titled "TensorFlow API" and displays the "Tensors" section. On the left, there is a sidebar with categories like Padding, Noise, Mask, and Operations. The Operations category is expanded, showing sub-categories such as tf.add, tf.sqrt, tf.multiply, etc. The main content area shows the `tf.tensor` function documentation. It includes a brief description: "Creates a `Tensor` with the provided values, shape and dtype.", three examples of code snippets, and detailed parameters for `values`, `shape`, and `dtype`. The `values` parameter is described as a "TypedArray or array-like object" that can be a flat array or a nested array. The `shape` parameter is a "number[] or tuple" that defines the dimensions of the tensor. The `dtype` parameter is a "string" that specifies the data type.

**Tensors**

Tensors are the core datastructure of TensorFlow.js. They are a generalization of vectors and matrices to potentially higher dimensions.

**Tensors / Creation**

We have utility functions for common cases like Scalars, 1D, 2D, 3D and 4D tensors, as well as a number of functions to initialize tensors in ways useful for machine learning.

**tf.tensor (values, shape?, dtype?)**

Creates a `Tensor` with the provided values, shape and dtype.

// Pass an array of values to create a vector.  
`tf.tensor([1, 2, 3, 4]).print();`

// Pass a nested array of values to make a matrix or a higher  
// dimensional tensor.  
`tf.tensor([[1, 2, 3], [4, 5, 6]]).print();`

// Pass a 2D array and specify a shape, position 2.  
`tf.tensor([1, 2, 3, 4, 5, 6]).print();`

**Parameters:**

`values` (`TypedArray` or `array`) The values of the tensor. Can be nested array of numbers, or a flat array or a `TensorLike`. If the values are strings, they will be encoded as utf-8 and kept as `text` memory.

`shape` (`number[]`) The shape of the tensor. Optional. If not provided, it is inferred from `values`. Optional.

`dtype` ('`float32`' | '`int32`' | '`bool`' | '`complex64`' | '`string`') The data type. Optional.

**Returns:** `tf.Tensor`





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a web browser window with the URL [https://www.tensorflow.org/api/api\\_docs/python/tf/Tensor](https://www.tensorflow.org/api/api_docs/python/tf/Tensor). The page title is "TensorFlow API". The main content is titled "Tensors". It describes tensors as the core datastructure of TensorFlow.js. Below this, there's a section titled "Tensors / Creation" with a sub-section for "tf.tensor". The code example shows creating a tensor from a nested array:

```
// Create a tf.Tensor with the provided values, shape and dtype.  
const x = tf.tensor([[[1, 2, 3], [4, 5, 6]],  
  [[7, 8, 9], [10, 11, 12]]]);
```

Below the code, there are three examples of how to use the `tf.tensor` function:

- `// Pass a nested array of values, no value or matrix or a higher  
// dimensional tensor.  
tf.tensor([1, 2, 3, 4, 5, 6, 7, 8, 9])`
- `// Pass a flat array and specify a shape yourself.  
tf.tensor([1, 2, 3, 4], [1, 3]).array()`

At the bottom, there are parameters for the `tf.tensor` function:

- values** (`TypeableArray`) The values of the tensor. Can be nested array of numbers, or a flat array, or a `TensorLike`. If the values are strings, they will be encoded as utf-8 and kept as `TensorString`.
- shape** (`[number]`) The shape of the tensor. Optional. If not provided, it is inferred from `values`. Optional.
- dtype** (`'float32' | 'int32' | 'bool32' | 'complex64' | 'string'`) The data type. Optional.



Connect with attendees on the OSS+ELC Slack Channel!

powered by



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a web browser window displaying the TensorFlow.js Node API documentation. The URL in the address bar is <https://www.tensorflow.org/js/api/2.0.1/>. The page title is "TensorFlow". The main content area is titled "TensorBoard" and contains functions for visualizing training in TensorBoard. One function, `tf.node.summaryFileWriter`, is highlighted with its code snippet:

```
tf.node.summaryFileWriter (logdir, maxQueue?, flushMillis?,  
filenameSuffix?)
```

The code example shows how to create a summary file writer for TensorBoard:

```
const tf = require('tensorflow/lib/tfjs-node');  
  
const summaryWriter = tf.node.summaryFileWriter('tmp/replay_logs');  
  
for (let step = 0; step < 100; ++step) {  
  summaryWriter.write('summary', Math.PI * step / 10, step);  
}
```

Parameters for `tf.node.summaryFileWriter` are listed as follows:

- `logdir` (string) Log directory in which the summary data will be written.
- `maxQueue` (number) Maximum queue length (default: 10). Optional.
- `flushMillis` (number) Flush every ... milliseconds (default: 10ms, i.e., 100 seconds). Optional.
- `filenameSuffix` (string) Suffix of the protocol buffer file names to be written in the writer (default: .pb). Optional.

The `tf.node.tensorBoard` function is also mentioned at the bottom of the page.



Connect with attendees on the OSS+ELC Slack Channel!

powered by



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

## VIDEO

The screenshot shows a web browser window displaying the TensorFlow.js React Native API documentation at [https:// tensorflow.org /api/react\\_native/0.3.0/](https:// tensorflow.org /api/react_native/0.3.0/). The page has a purple header with the Open Source Summit and Embedded Linux Conference logos. The main content area is titled "Platform helpers" and includes sections for "http", "fetch", "Models", and "MEDIA". A detailed description of the "fetch" function is shown, along with its parameters and return value. The bottom of the page shows a navigation bar with icons for refresh, search, and user profile.

**Platform helpers**

`tflReactNative` provides a TensorFlow.js platform adapter for react native.

All symbols are **named exports** from the `tfljs-react-native` package.

**http**

**fetch (path, init?, options?)** Function

Makes an HTTP request.

**Parameters:**

- `path` (`string`) The URL path to make a request to
- `init` (`RequestInit`) The request init. See init here: <https://developer.mozilla.org/en-US/docs/Web/API/Request/Request> optional
- `options` (`RequestDetails`) A RequestDetails object.
  - `options.binary` boolean indicating whether this request is for a binary file. optional

**Returns:** `Promise<Response>`

**Models**

Model loading and saving.

**Models / IOHandlers**

Connect with attendees on the OSS+ELC Slack Channel!

powered by

**VIDEO**

The screenshot shows a web browser displaying the TensorFlow API Reference for version 1.4.3. The URL is [https://www.tensorflow.org/api\\_docs/python/tfvis](https://www.tensorflow.org/api_docs/python/tfvis). The main content is the **Visor & Surfaces** section. On the left, there's a sidebar with navigation links for **API Version** (1.4.3), **VISOR & SURFACES**, **MODELS & TENSORS**, and **CHARTS**. The **VISOR & SURFACES** section contains documentation for `tfvis.visor()` and `tfvis.Visor`. The `tfvis.visor()` documentation includes a code snippet for creating a visor object:

```
tfvis.visor() → Function
The primary interface to the visor is the visor() function.  
This returns a singleton instance of the Visor class. The singleton object will be replaced if the visor is removed from the DOM for some reason.
```

Below this, there's a code editor with the following code:

```
Show the visor
tfvis.Visor()
```

The **MODELS & TENSORS** sidebar lists methods like `Model.inspection`, `Model.callbacks`, and `Model.history`. The **CHARTS** sidebar lists methods like `Chart.barchart`, `Chart.confusionMatrix`, and `Chart.heatmap`.



Connect with attendees on the OSS+ELC Slack Channel!

powered by Intrado



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

## THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

notes.md    run-tfjs-model.js x    build-model.js

```
getting-started-with-ai-in-node.js > src > run-tfjs-model.js > ...
14 // Load COCO-SSD graph model from TensorFlow Hub
15 const loadModel = async function () {
16   console.log('loading model from ${modelUrl}');
17
18   model = await tf.loadGraphModel(modelUrl, {fromTFHub: true});
19
20   return model;
21 }
22
23 // convert image to Tensor
24 const processInput = function (imagePath) {
25   console.log(`preprocessing image ${imagePath}`);
26
27   const image = fs.readFileSync(imagePath);
28   const buf = Buffer.from(image);
29   const uint8array = new Uint8Array(buf);
30
31   return tf.node.decodeImage(uint8array, 3).expandDims();
32 }
33
34 // run prediction with the provided input Tensor
35 const runModel = function (inputTensor) {
36   console.log('running model');
37
38   return model.executeAsync(inputTensor);
39 }
40
41 // process the model output into a friendly JSON format
42 const processOutput = function (prediction) {
43   console.log('processOutput');
44
45   const [maxScores, classes] = extractClassesAndMaxScores(prediction[0]);
46   const indexes = calculateNMS(prediction[1], maxScores);
47
48   return createJSONResponse(prediction[1].dataSync(), maxScores, indexes, classes);
49 }
50
51 // determine the classes and max scores from the prediction
52 const extractClassesAndMaxScores = function (predictionScores) {
53   console.log('calculating classes & max scores');
```



Connect with attendees on the OSS+ELC Slack Channel!

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

## THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

```
notes.md  run-tfjs-model.js  build-model.js
getting-started-with-ai-in-node.js > src > run-tfjs-model.js > ...
125   .then(annotatedImageBuffer => {
126     const f = path.join(path.parse(imagePath).dir, `${path.parse(imagePath).name}-annotate.png`);
127
128     fs.writeFile(f, annotatedImageBuffer, (err) => {
129       if (err) {
130         console.error(err);
131       } else {
132         console.log(`annotated image saved as ${f}\r\n`);
133       }
134     });
135   });
136 }
137
138 // run
139 if (process.argv.length < 3) {
140   console.log('please pass an image to process. ex:');
141   console.log(`  node run-tfjs-model.js /path/to/image.jpg`);
142 } else {
143   // e.g., /path/to/image.jpg
144   let imagePath = process.argv[2];
145
146   loadModel().then(model => {
147     const processInput: (imagePath: any) => any
148     const inputTensor = processInput(imagePath);
149     height = inputTensor.shape[1];
150     width = inputTensor.shape[2];
151     return runModel(inputTensor);
152   }).then(prediction => {
153     const jsonOutput = processOutput(prediction);
154     console.log(jsonOutput);
155     annotateImage(jsonOutput, imagePath);
156   })
157 }
```



Connect with attendees on the OSS+ELC Slack Channel!

powered by Intrado



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

## THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

```
*** notes.md    JS run-tfs-model.js  JS build-model.js X
coding-a-deep-learning-model-using-tensorflow-javascript > src > JS build-model.js > set buildModel
 67 // Define the model architecture
 68 const buildModel = function () {
 69   const model = tf.sequential();
 70
 71   // add the model layers
 72   model.add(tf.layers.conv2d({
 73     inputShape: [imageWidth, imageHeight, imageChannels],
 74     filters: 8,
 75     kernelSize: 5,
 76     padding: 'same',
 77     activation: 'relu'
 78   }));
 79   model.add(tf.layers.maxPooling2d({
 80     poolSize: 2,
 81     strides: 2
 82   }));
 83   model.add(tf.layers.conv2d({
 84     filters: 16,
 85     kernelSize: 5,
 86     padding: 'same',
 87     activation: 'relu'
 88   }));
 89   model.add(tf.layers.maxPooling2d({
 90     poolSize: 3,
 91     strides: 3
 92   }));
 93   model.add(tf.layers.flatten());
 94   model.add(tf.layers.dense({
 95     units: numOfClasses,
 96     activation: 'softmax'
 97   });
 98
 99   // compile the model
100   model.compile({
101     optimizer: 'adam',
102     loss: 'categoricalCrossentropy',
103     metrics: ['accuracy']
104   });
105
106   return model;
107 }
```



[Lobby](#)[Sessions](#)[Sponsor Showcase](#)[Networking](#)[Experiences](#)[Fun & Games](#)[Info](#)[Profile](#)

VIDEO



# Node-RED with TensorFlow.js



Share Your Experience on social - tag posts with #OSSummit #LFELC

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



VIDEO

# Why bring TF.js to Node-RED?

## ML to IoT with ease

- Intuitive visual programming model
- Lower barrier to entry

## Privacy

- Data security
- Data does not leave device

## Network

- Offline
- Low bandwidth
- Remote locations





# flows.nodered.org / GitHub

## [node-red-contrib-tf-model](#)

custom Node-RED node to retrieve/execute a Tensorflow model

## [node-red-contrib-tf-function](#)

Node-RED custom node which mimic the original function node but provide tf in the global context

## [node-red-contrib-bert-tokenizer](#)

Node-RED module to implement bert-tokenizer

## [node-red-contrib-post-object-detection](#)

Node-RED module to handle prediction results of an object detection model.

## [Node-RED TFJS Object Detection](#)

Node-RED module for TensorFlow.js object detection model

## [Node-RED for ML with TFJS](#)

Node-RED nodes with TensorFlow.js to enable machine learning in Node-RED

... and more





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

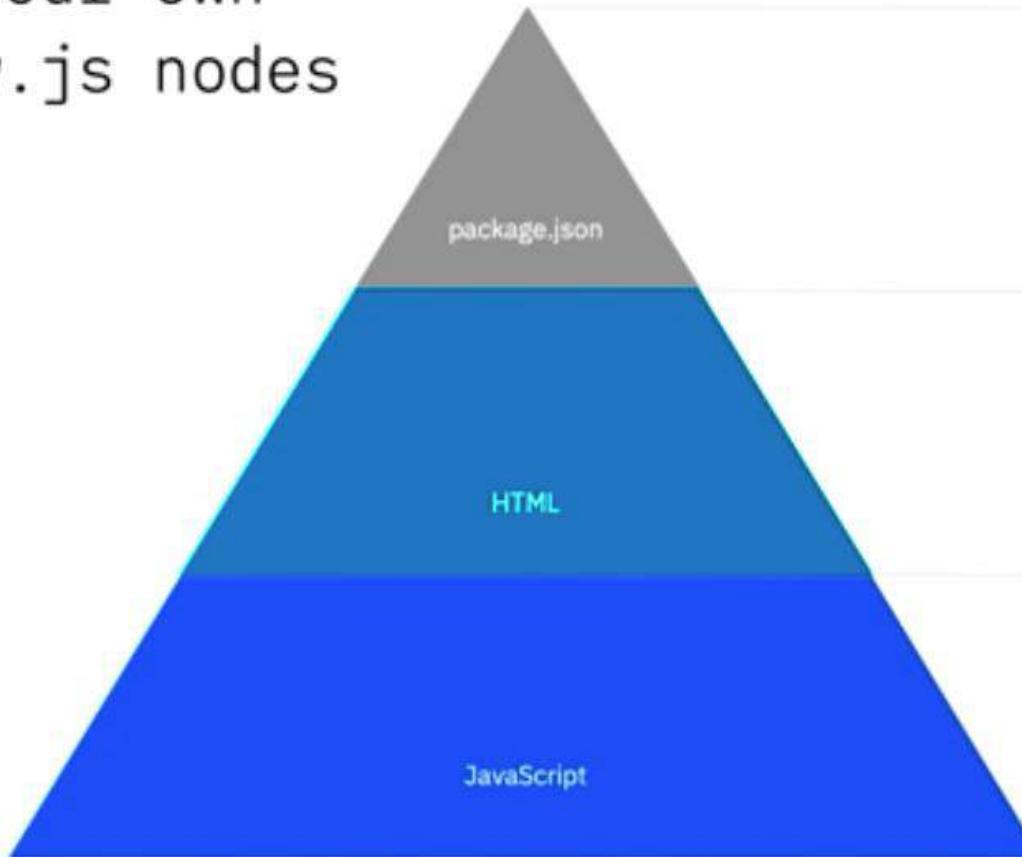
Info

Profile



VIDEO

# Creating your own TensorFlow.js nodes



<https://medium.com/codait/a-low-code-approach-to-incorporating-machine-learning-into-your-iot-device-24f3f2a70717>



Take a break! Enjoy our Experiences + Fun & Games!

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile

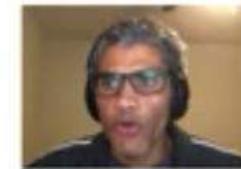


# OPEN SOURCE SUMMIT NORTH AMERICA

## THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

```
notes.md package.json index.html index.js
building-a-machine-learning-node-for-node-red-using-tensorflow.js > src > nodered-dev > node-red-contrib-tfjs-tutorial > package.json > ...
1 {
2   "name": "node-red-contrib-tfjs-tutorial",
3   "version": "0.0.1",
4   "description": "Example of custom Node-RED with TensorFlow.js",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "author": "",
10  "license": "Apache-2.0",
11  "peerDependencies": {
12    "@tensorflow/tfjs-node": "^1.7.2"
13  },
14  "node-red": {
15    "nodes": {
16      "tfjs-tutorial-node": "index.js"
17    }
18  }
19 }
```





# OPEN SOURCE SUMMIT NORTH AMERICA

## THE LINUX FOUNDATION Embedded Linux Conference North America

**VIDEO**

The screenshot shows a video player interface. On the left, there is a vertical toolbar with icons for notes, search, refresh, and other controls. The main area is a code editor with tabs for notes.md, package.json, index.html (which is currently selected), and index.js. The code in index.html is as follows:

```
1 <script type="text/html" data-template-name="tfjs-tutorial-node">
2   <div class="form-row">
3     <label for="node-input-name"><i class="fa fa-tag"></i> Name</label>
4     <input type="text" id="node-input-name" placeholder="Name">
5   </div>
6   <div class="form-row">
7     <label for="node-input-modelUrl"><i class="fa fa-globe"></i> Model Url</label>
8     <input type="text" id="node-input-modelUrl" placeholder="https://modelurl/model.json">
9   </div>
10  <div class="form-row">
11    <label for="node-input-fromHub">Is TFHub?</label>
12    <input type="checkbox" id="node-input-fromHub" checked>
13  </div>
14 </script>
15
16 <script type="text/javascript">
17   RED.nodes.registerType('tfjs-tutorial-node', {
18     category: 'machine learning',
19     defaults: {
20       name: { value: 'tfjs tutorial node' },
21       modelUrl: { value: 'https://tfhub.dev/tensorflow/tfjs-model/ssdlite_mobilenet_v2_1/default/1' },
22       fromHub: { value: 'checked' }
23     },
24     inputs: 1,
25     outputs: 1,
26     paletteLabel: 'tfjs tutorial node',
27     color: '#FF9100',
28     label: function() {
29       return this.name || 'tfjs tutorial node';
30     }
31   });
32 </script>
33
34 <script type="text/html" data-help-name="tfjs-tutorial-node">
35   <p>A TensorFlow.js node to run prediction using the Coco SSD model for object detection.</p>
36   <p>Provide a <strong>Model Url</strong> and indicate whether the URL points to a TFHub hosted model.</p>
37   <p>The node accepts a image buffer and outputs a JSON prediction object with detected objects and their bounding box and confidence score.</p>
38 </script>
39
```

On the right side of the video player, there is a small video feed showing a man with glasses and dark hair, wearing a dark shirt. Below the video player, there is a red footer bar with several icons: a refresh circle, a play button, a list icon, a message icon, and a user profile icon.



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

## THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

A screenshot of a video player interface. The main area is a code editor displaying a file named index.js. The code is written in JavaScript and defines a node type for Node-RED. It includes properties like category ('machine learning'), defaults (name: 'tfjs tutorial node', modelUrl: 'https://tfhub.dev/tensorflow/tfjs-model/ssdlite\_mobilenet\_v2\_1/default/1', fromHub: 'checked'), inputs (1), outputs (1), paletteLabel ('tfjs tutorial node'), color ('#FF9100'), and a label function that returns the name. The code also includes a script tag with help text for the TensorFlow.js node.

```
notes.md package.json index.html index.js
building-a-machine-learning-node-for-node-red-using-tensorflow.js > src > nodered-dev > node-red-contrib-tfjs-tutorial > index.html > script
  9  </div>
 10 <div class="form-row">
 11   <label for="node-input-fromHub">Is TFHub?</label>
 12   <input type="checkbox" id="node-input-fromHub" checked>
 13 </div>
 14 </script>
 15
 16 <script type="text/javascript">
 17   RED.nodes.registerType('tfjs-tutorial-node', {
 18     category: 'machine learning',
 19     defaults: {
 20       name: { value: 'tfjs tutorial node' },
 21       modelUrl: { value: 'https://tfhub.dev/tensorflow/tfjs-model/ssdlite_mobilenet_v2_1/default/1' },
 22       fromHub: { value: 'checked' }
 23     },
 24     inputs: 1,
 25     outputs: 1,
 26     paletteLabel: 'tfjs tutorial node',
 27     color: '#FF9100',
 28     label: function() {
 29       return this.name || 'tfjs tutorial node';
 30     }
 31   });
 32 </script>
 33
 34 <script type="text/html" data-help-name="tfjs-tutorial-node">
 35   <p>A TensorFlow.js node to run prediction using the Coco SSD model for object detection.</p>
 36   <p>Provide a <strong>Model Url</strong> and indicate whether the URL points to a TFHub hosted model.</p>
 37   <p>The node accepts a image buffer and outputs a JSON prediction object with detected objects and their bounding box and confidence score.</p>
 38 </script>
 39
```





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

## THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

```
notes.md  package.json  index.html  index.js  X
building-a-machine-learning-node-for-node-red-using-tensorflow.js > src > nodered-dev > node-red-contrib-tfjs-tutorial > index.js > <unknown> > exports > TfjsTutorialNode > node.on('input' callback > (index.js:19)
  2 module.exports = function(RED) {
  3   // import helper module
  4   const tfutil = require('./tfjs-tutorial-util.js');
  5
  6   // load the model
  7   async function loadModel (config, node) {
  8     node.model = await tfutil.loadModel(config.modelUrl, config.fromHub);
  9   }
 10
 11   // define the node's behavior
 12   function TfjsTutorialNode(config) {
 13     // initialize the features
 14     RED.nodes.createNode(this, config);
 15     const node = this
 16
 17     loadModel(config, node)
 18
 19     // register a listener to get called whenever a message arrives at the node
 20     node.on('input', function (msg) {
 21       // preprocess the incoming image
 22       const inputTensor = tfutil.processInput(msg.payload);
 23       // get image/input shape
 24       const height = inputTensor.shape[1];
 25       const width = inputTensor.shape[2];
 26
 27       // get the prediction
 28       node.model
 29         .executeAsync(inputTensor)
 30         .then(prediction => {
 31           msg.payload = tfutil.processOutput(prediction, height, width);
 32           // send the prediction out
 33           node.send(msg);
 34         });
 35     });
 36   }
 37
 38   // register the node with the runtime
 39   RED.nodes.registerType('tfjs-tutorial-node', TfjsTutorialNode);
 40 }
 41
```



Take a break! Enjoy our Experiences + Fun & Games  
powered by Infradot



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows the Node-RED interface with a single flow named "Flow 1". The flow consists of the following nodes:

```
graph LR; A[inject] --> B[filename:file.js]; B --> C[file:File Read Node]; C --> D[msg:payload];
```

The "filename" node has a "File preview" button below it.

The right panel displays the configuration for the "File Read Node" (Node ID: dc828ffed344f):

- Information**:
  - Node: "dc828ffed344f"
  - Name: "file.js.gpg"
  - Type: File in
- Description**: Reads the contents of a file as either a string or binary buffer.
- Inputs**:
  - filename**: If not set in the node configuration, this property sets the filename to read.
- Outputs**:
  - payload**: The contents of the file as either a string or binary buffer.
  - filename**: If not configured in the node, this optional property sets the name of the file to be read.
  - error**: Deprecated: if enabled in the node, when the node hits an error reading the file, it will send a message with no payload and this error property set to the error details. This mode of behaviour is deprecated and not enabled by default for new instances of the node. See below for more information.
- Details**: The filename should be an absolute path, otherwise it will be relative to the working directory of the Node-RED process.



Questions? Ask the Events team on the #1-help Slack channel.  
powered by Infradeck



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

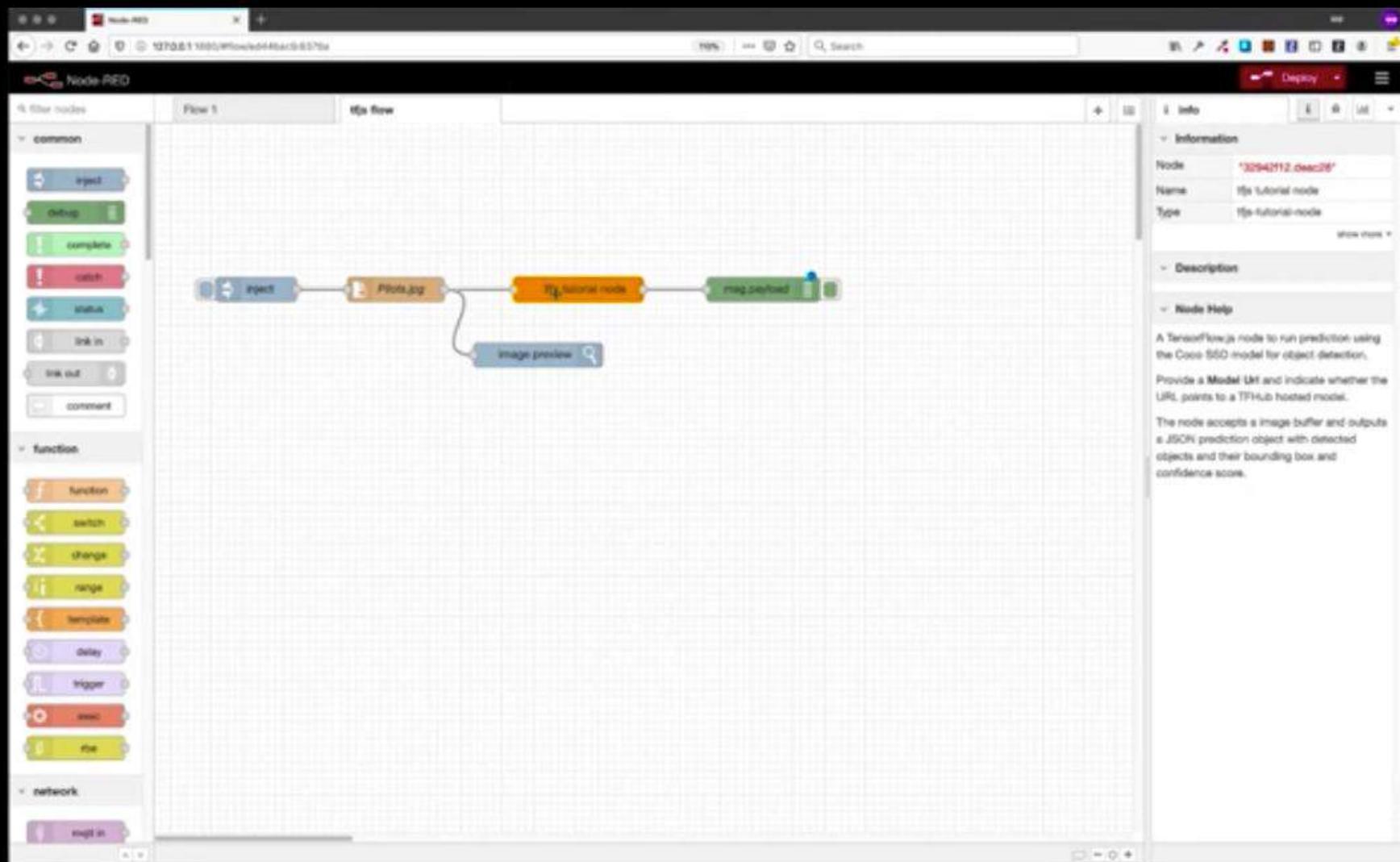
Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO



Questions? Ask the Events team on the #1-helpdesk Slack channel

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED workspace window titled "Node-RED". The URL in the address bar is "127.0.0.1:1880/#flows/448ac80.579e". The workspace contains a flow labeled "Flow 1" with the following nodes connected sequentially:

- An "inject" node (blue)
- A "Plotting" node (orange)
- An "Rx Serialized Node" node (yellow)
- An "msgStamp" node (green)

A connection from the "Plotting" node to the "msgStamp" node is labeled "Image preview". A tooltip for the "msgStamp" node says "Image preview".

The left sidebar lists categories of nodes:

- common: inject, debug, complete, catch, status, link in, link out, comment
- function: function, switch, change, range, template, delay, trigger, eval, rseq
- network: msg in

The right sidebar displays detailed information for the selected "msgStamp" node:

- Information**: Node "448ac80.579e", Type "debug".
- Description**: Displays selected message properties in the debug sidebar tab and optionally the runtime log. By default, it displays `msg.payload`, but can be configured to display any property, the full message or the result of a JSONPath expression.
- Details**: The debug sidebar provides a structured view of the messages it sent, making it easier to understand their structure. JavaScript objects and arrays can be collapsed and expanded as required. Buffer objects can be displayed as raw data or as a string if possible. Alongside each message, the debug sidebar includes information about the time the message was received, the node that sent it and the type of the message. Clicking on the source node id will reveal that node within the workspace.
- Node Help**: Provides help for the "msgStamp" node.



Questions? Ask the Events team on the #1-helpdesk Slack channel

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

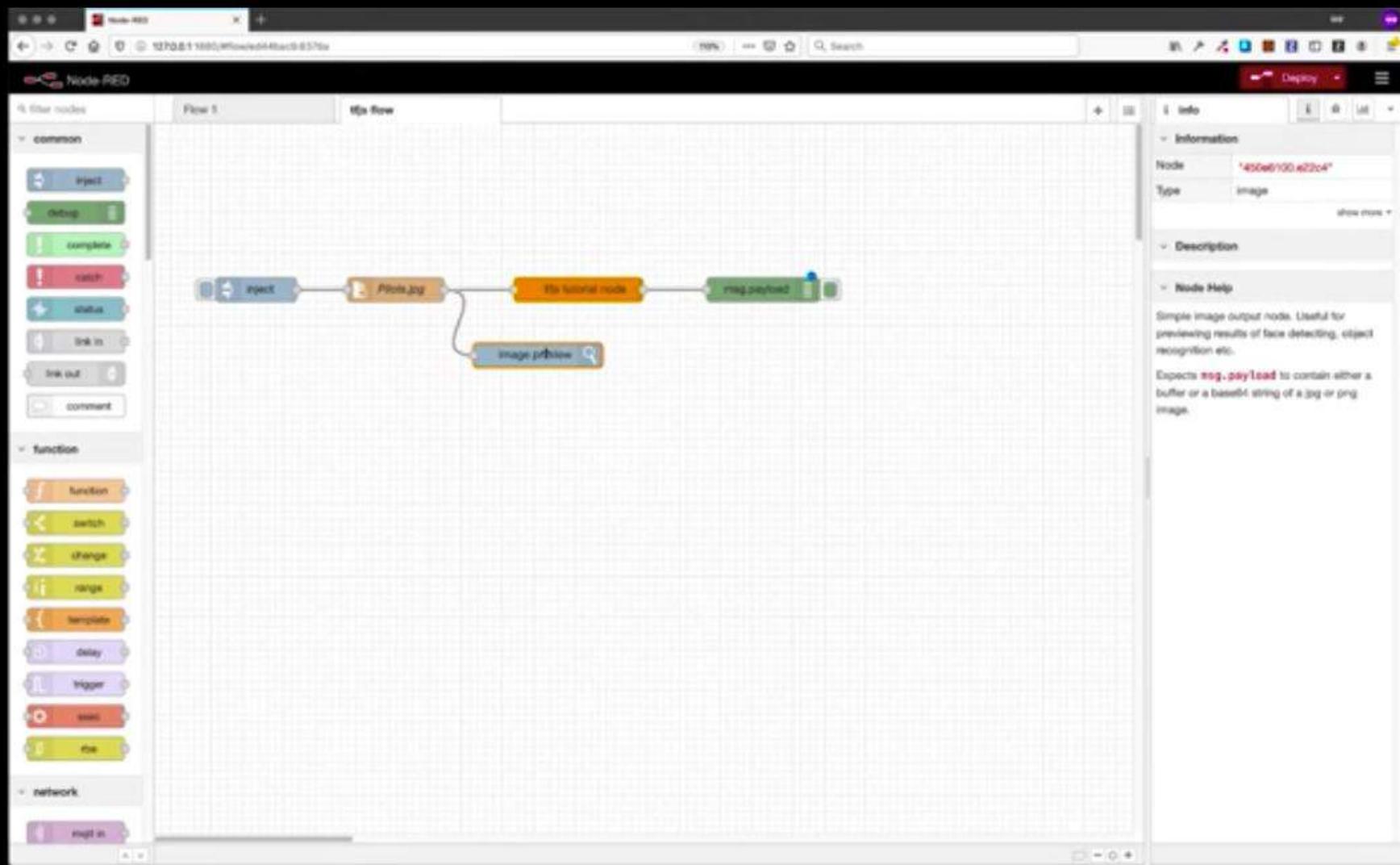
Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED interface with a flow consisting of three nodes: an 'inject' node, a 'PlotImage' node, and an 'Image output' node. The 'PlotImage' node has a preview window displaying a group of people in white space suits. The 'Image output' node has a preview window showing the same image. A success message 'Successfully injected: inject' is displayed in a toast notification.

```
graph LR; inject[inject] --> PlotImage[PlotImage]; PlotImage --> ImageOutput[Image output];
```

**Node-RED**

Flow 1

Successfully injected: inject

common

- inject
- debug
- complete
- catch
- status
- link in
- link out
- comment

function

- function
- switch
- change
- range
- template
- delay
- trigger
- exec
- else

network

- msg in

Information

Node: 400e6100.e22c4

Type: Image

Description

Node Help

Simple image output node. Useful for previewing results of face detecting, object recognition etc.

Requires **msg.payload** to contain either a buffer or a base64 string of a jpg or png image.



Be Sure to Visit the Sponsor Showcase!

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED interface with a flow titled "Flow 1". The flow consists of the following nodes:

- An "inject" node with two outputs.
- A "PlotImage" node connected to the first output of the inject node.
- An "Rasa trained node" connected to the second output of the inject node.
- A "msg payload" node connected to the output of the Rasa trained node.

Below the flow, there is an "image preview" component displaying a thumbnail of three people in white space suits. To the right of the flow, a "debug" panel shows the following JSON data:

```
APR090001 12:06:12 AM node 60910295-0284 msg.payload: array[2] + B: object + B0: array[4] label: "person" score: 0.9464929183851338 + B1: object + B0: array[4] label: "person" score: 0.913060652518677 + B2: object + B0: array[4] label: "person" score: 0.7818169926643372
```



Connect with attendees on the OSS+ELC Slack Channel!

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED interface with a flow titled "Flow 1". The flow consists of the following nodes:

- An "inject" node connected to a "Print Log" node.
- The "Print Log" node has a connection to an "Image" node.
- The "Image" node displays a thumbnail of three astronauts in a circular frame, labeled "Image preview".
- The "Image" node connects to an "R� Is it a person?" node.
- The "R� Is it a person?" node has a connection to a "Print Dashboard" node.

The "R� Is it a person?" node is expanded to show its payload data:

```
2018/09/11 10:06:12 AM node:30910298-00944
msg.payload: array[0]
+ 0: object
  + 0: array[4]
    0: 35.42379144653882
    1: 148.08447735228539
    2: 258.019543152842236
    3: 462.02996838668864
    label: "person"
    score: 0.9664929183851218
+ 1: object
  + 1: array[4]
    0: 35.42379144653882
    1: 148.08447735228539
    2: 258.019543152842236
    3: 462.02996838668864
    label: "person"
    score: 0.9135660653518673
+ 2: object
  + 2: array[4]
    0: 35.42379144653882
    1: 148.08447735228539
    2: 258.019543152842236
    3: 462.02996838668864
    label: "person"
    score: 0.7819169936643372
```



Connect with attendees on the OSS+ELC Slack Channel!

powered by

[Lobby](#)[Sessions](#)[Sponsor Showcase](#)[Networking](#)[Experiences](#)[Fun & Games](#)[Info](#)[Profile](#)

VIDEO

## Challenges and Best Practices

### Models

- loading
- caching

### Performance

- worker thread

### Data

- audio
- video
- sensors



[Lobby](#)[Sessions](#)[Sponsor Showcase](#)[Networking](#)[Experiences](#)[Fun & Games](#)[Info](#)[Profile](#)

VIDEO



# Example Flows & Use Case





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

## THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows the Node-RED interface running in a browser window titled "Node-RED" at "Node-RED : 192.168.1.152". The URL in the address bar is "https://localhost:1880/#flow/901942a5.c76d78". The interface includes a sidebar with categories like "input" (file inject, microphone, camera) and "output" (image, play audio). A main canvas displays a flow starting with a "camera" node, followed by an "rjs tutorial node", and ending with a "msg.payload" node. A "debug" panel on the right shows the output of the flow.



Connect with attendees on the OSS+ELC Slack Channel!

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED interface running in a browser window titled "Node-RED" at "https://localhost:1880/#/flow/eGed8e49-34ead". The flow consists of the following nodes and connections:

- Input:** A "camera" node is connected to a "pre-processing" node.
- Process:** "pre-processing" connects to a "COCO SSD lite" model node, which then connects to "post-processing". A "model is ready" message is sent from the model node to the "post-processing" node.
- Output:** "post-processing" connects to a "bounding-box" node, which then connects to a "mag.payload" node.
- Feedback:** A feedback loop from the "post-processing" node goes back to the "Original Image" node, and another from the "bounding-box" node goes back to the "With bounding boxes" node.

The Node-RED interface includes a sidebar with categories like "input" (file inject, microphone, camera) and "output" (image, play audio), and a "sequence" section with "split", "join", and "sort" nodes. A "debug" panel on the right shows the current state of the flow. A video preview window in the bottom right corner displays a live feed from a camera showing a person's face.



Share Your Experience on social - tag posts with #OSSummit #LFELC

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

## THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED interface running in a browser window. The URL is <https://localhost:1880/#flow/efed8e49-34ea-4>. The flow consists of the following nodes:

- A "Camera" node connected to a "pre-processing" node.
- "pre-processing" connects to a "COCO SSD lite" node.
- "COCO SSD lite" has a "model is ready" output.
- "COCO SSD lite" connects to a "post-processing" node.
- "post-processing" connects to a "bounding-box" node.
- "bounding-box" connects to a "msg.payload" output node.
- "Original Image" and "With bounding boxes" inputs connect to the "Camera" node.
- "With bounding boxes" also connects to a preview video player at the bottom right.

The "msg.payload" output is displayed in the "debug" panel:

```
6/17/2020, 7:20:20 PM node: (8003ae5.68198)
msg.payload: array[2]
+ [ object, object ]
```

On the left sidebar, under "input", there are nodes for "file inject", "microphone", and "camera". Under "output", there are nodes for "image" and "play audio".





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED flow editor window. On the left, there's a sidebar with categories like 'input' (file inject, microphone, camera) and 'output' (image, play audio). The main canvas has a 'Camera' input node connected to a 'pre-processing' function node. The function node's properties tab shows the name 'pre-processing' and the following JavaScript code:

```
1- const image = tf.tidy(() => {
2-   return tf.node.decodeImage(msg.payload, 3).expandDims(0);
3- });
4-   I
5- return {payload: {image_tensor: image} };
```

To the right of the canvas is a 'debug' panel showing log entries from June 17, 2020, at 7:20:20 PM. It includes a preview window showing a person's face.

Connect with attendees on the OSS+ELC Slack Channel!

powered by



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

Node-RED x Node-RED : 192.168.1.152 x https://localhost:1880/#flow/fe6ed8e49-34ead

Node-RED

simple flow general purpose

Camera → pre-processing → [ ]

Original Image

File inject microphone camera

Image play audio

split join sort

Edit tf-model node

Properties

Model URL: https://storage.googleapis.com/tfjs-models/saved

Output Node:

Name: COCO SSD lite

debug

6/17/2020, 7:20:20 PM node: d8089ae5-68198  
msg.payload: array[2]  
+ [ object, object ]





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot displays a Node-RED flow for real-time object detection. The flow starts with a "Camera" node connected to a "pre-processing" node. This is followed by a "COCO SSD lite" node and a "post-processing" node. A "model is ready" message is sent from the "post-processing" node back to the "pre-processing" node. The flow then splits into two parallel paths. The left path leads to a "bounding-box" node, which then connects to an "image" output node labeled "Original Image". The right path connects directly to an "image" output node labeled "With bounding boxes". The "With bounding boxes" view shows a person's face with green bounding boxes for "person" and blue bounding boxes for "cell phone". On the far right, a "debug" panel shows the JSON payload of the "msg.payload" node, which contains two objects: one for a person and one for a cell phone, each with a bounding box, class name, and score.

```
4/17/2020, 7:20:20 PM node: (bb003w6.68158)
msg.payload: array[2]
  * array[2]
    * 0: object
      * bbox: array[4]
        className: "person"
        score: 0.6840768111888777
    * 1: object
      * bbox: array[4]
        className: "cell phone"
        score: 0.6436811685562134
```



Connect with attendees on the OSS+ELC Slack Channel!

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED flow titled "auto garage door opener". The flow starts with a "Camera" node connected to a "pre-processing" node. The output of "pre-processing" goes to a "COCO SSD lite" node, which then feeds into a "post-processing" node. A "model is ready" message is sent from "post-processing" back to "pre-processing". From "post-processing", the flow splits into two parallel paths. The top path leads to a "bounding-box" node, which then outputs to a "msg.payload" node. The bottom path leads directly to "msg.payload". Two preview images are shown: "Original Image" and "With bounding boxes". The "With bounding boxes" image shows a person holding a smartphone with bounding boxes around them and the phone. The "msg.payload" node shows the JSON output of the model's prediction.

```
graph LR; Camera[Camera] --> pre[pre-processing]; pre --> COCO[COCO SSD lite]; COCO --> post[post-processing]; post -- "model is ready" --> pre; post --> bbox[bounding-box]; post --> payload[msg.payload];
```

Original Image

With bounding boxes

msg.payload

debug

```
6/17/2020, 7:20:30 PM node: (bb093a6.68158)
msg.payload: array[2]
  * array[2]
    * 0: object
      * bbox: array[4]
        0: 0.14873957633972168
        1: 0.18795843238056763
        2: 0.8803385866986084
        3: 0.8108344878863965
      className: "person"
      score: 0.6840768111886777
    * 1: object
      * bbox: array[4]
        0: 0.14873957633972168
        1: 0.18795843238056763
        2: 0.8803385866986084
        3: 0.8108344878863965
      className: "cell phone"
      score: 0.6436811685562134
```





# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED application window. On the left, a sidebar lists node categories: input (top request, udp in, udp out), output (file inject, microphone, cameras), and sequence (split, join, sort). The main canvas displays a flow starting with a "Camera" node, followed by a "pre-processing" node, and ending with an "image" output node. A preview window titled "Original Image" shows a person holding a smartphone. To the right, a modal dialog titled "Edit bbox-image node" is open, showing the "Properties" tab with a "Name" field containing "bounding-box". Below the canvas, the URL "https://localhost:1880/#editor-tab-properties" is visible. On the far right, a "debug" panel shows log entries and a preview window showing a person's face.

Node-RED

simple flow general purpose

input

output

sequence

https://localhost:1880/#editor-tab-properties

Camera → pre-processing → image

Original Image

Edit bbox-image node

Name: bounding-box

Enabled

debug

6/17/2020, 7:20:20 PM node: (8003)w5.68158  
msg.payload: array[2]  
+ array[2]  
+ 0: object  
+ bbox: array[4]  
0: 0.14873957633972168  
1: 0.18795843238856763  
2: 0.8883385866986884  
3: 0.8188344878863965  
className: "person"  
score: 0.6840768111888777  
+ 1: object  
+ bbox: array[4]  
className: "cell phone"  
score: 0.6436811685562134



Take a break! Enjoy our Experiences + Fun & Games!

powered by Intrado



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows the Node-RED interface running in a browser window. The flow consists of a camera node connected to a 'post-processing' function node. The output of 'post-processing' is then split into two parallel paths, each ending with a 'debug' node. The left path's debug output shows an array of detected objects, including a person and a cell phone. The right path's debug output shows a video feed of a person holding a smartphone.

simple flow general purpose

Camera → post-processing → debug

Original Image

Properties

- Class URL: https://s3.sjc.us.cloud-object-storage.appdomain
- IoU: 0.5
- Min Score: 0.5
- Name: post-processing

Enabled

debug

```
6/17/2020, 7:20:20 PM node: (8080)@6.68.158
msg.payload: array[2]
  * array[2]
    * 0: object
      * bbox: array[4]
        0: 0.14873957633972168
        1: 0.18795843238856763
        2: 0.8803385066986084
        3: 0.8108344878863965
      className: "person"
      score: 0.6840768111880777
    * 1: object
      * bbox: array[4]
        className: "cell phone"
        score: 0.6436811685562134
```

<https://localhost:1880/#editor-tab-properties>

[Lobby](#)[Sessions](#)[Sponsor Showcase](#)[Networking](#)[Experiences](#)[Fun & Games](#)[Info](#)[Profile](#)

VIDEO



# Example Flows & Use Case



[Connect with attendees on the OSS+ELC Slack Channel!](#)

powered by Intrado

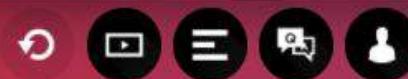
[Lobby](#)[Sessions](#)[Sponsor Showcase](#)[Networking](#)[Experiences](#)[Fun & Games](#)[Info](#)[Profile](#)

VIDEO

## Auto Garage Door



# Auto License Plate Recognition



Share Your Experience on social - tag posts with #OSSummit #LFELC

powered by **Intrado**

[Lobby](#)[Sessions](#)[Sponsor Showcase](#)[Networking](#)[Experiences](#)[Fun & Games](#)[Info](#)[Profile](#)

VIDEO

## Auto Garage Door



# Auto License Plate Recognition



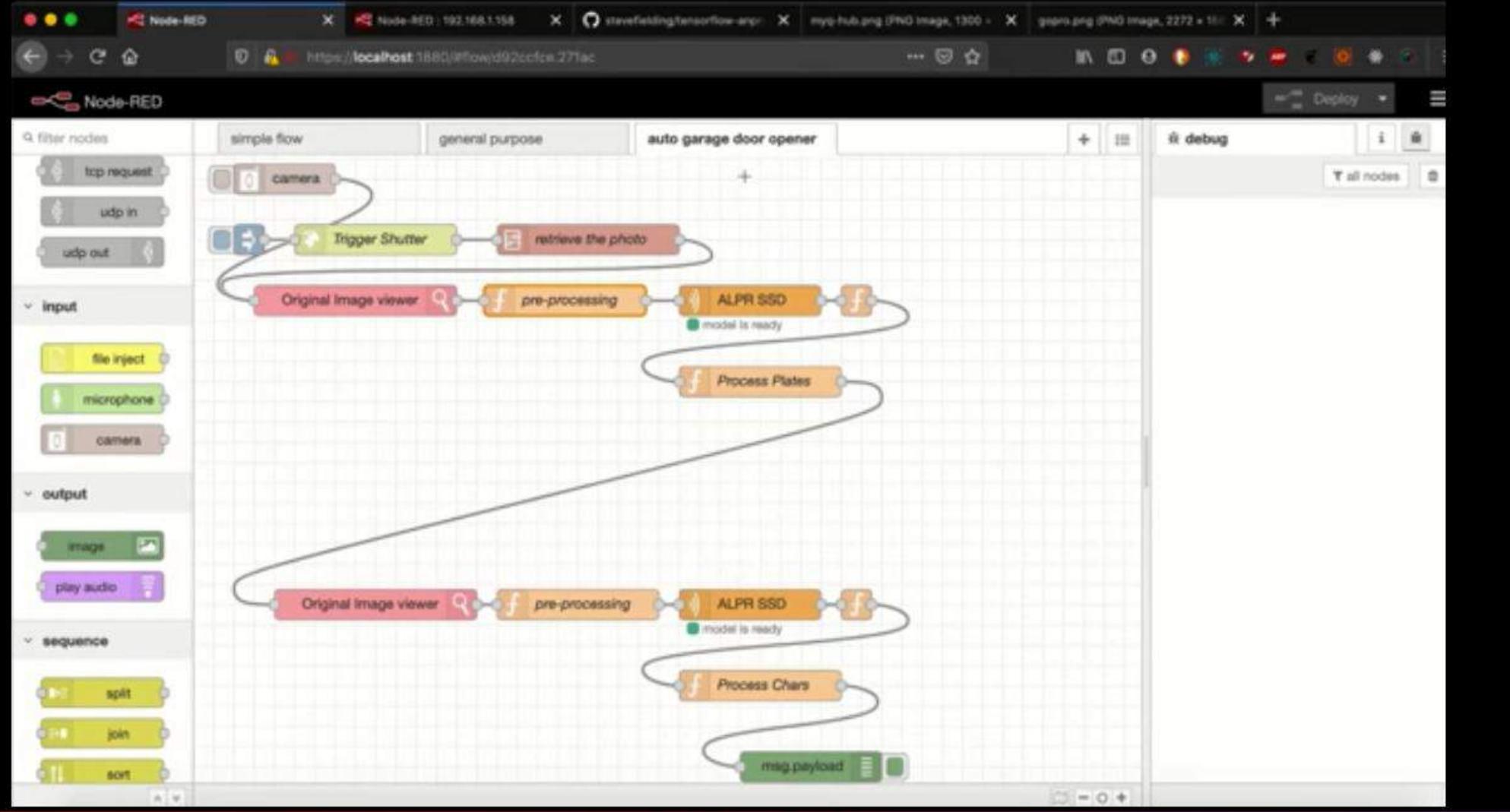
Questions? Ask the Events team on the #1 help desk  
powered by



# THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

VIDEO





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a GitHub repository page for 'stevefielding/tensorflow-anpr'. The repository title is 'Automatic Number (License) Plate Recognition using Tensorflow Object Detection API'. It has 15 stars, 117 forks, and 44 contributors. The repository is public and was created by 'Steve'. The last commit was on Oct 29, 2019. The repository contains 43 commits, 1 branch, 0 packages, and 0 releases. The contributors section shows 1 contributor, Steve. The repository uses the TensorFlow Object Detection API and is implemented in Python. The repository has a master branch and a New pull request button. The commit history includes several changes related to file uploads and detection logic.

Commit	Description	Date
Steve now supports multiple file upload	Combined predict_video.py into a single file with capability to do on...	2 years ago
base2designs	Added missing classes.pbtxt	2 years ago
conf	Modified conf file	2 years ago
dataset_prep/artificial	Added rejectPlates option to plateFinder. Added scaling to build_anpr...	2 years ago
templates	now supports multiple file upload	8 months ago
uploads	Added example images for different stages of detection	2 years ago
README.md	Improved README	2 years ago
analysis_noFile.py	Fixed bugs in analysis_noFile.py	2 years ago





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



VIDEO

Node-RED

Node-RED : 192.168.1.158

stevefielding/tensorflow-apr

myo-hub.png (PNG Image, 1350 × 1000)

gnarly.png (PNG Image, 2272 × 1000)

README.md

## Automatic Number (License) Plate Recognition

Detect vehicle license plates in videos and images using the tensorflow/object\_detection API.  
Train object detection models for license plate detection using TFOD API, with either a single detection stage or a double detection stage.





VIDEO

Node-RED Node-RED : 192.168.1.158 stevefielding/tensorflow-apr myq-hub.png (PNG Image, 1300 × 1000 pixels) gopro.png (PNG Image, 2272 × 1000 pixels)

Detect vehicle license plates in videos and images using the tensorflow/object\_detection API.  
Train object detection models for license plate detection using TFOOD API, with either a single detection stage or a double detection stage.  
The single stage detector, detects plates and plate characters in a single inference stage.  
The double stage detector detects plates in the first inference stage.

Image

crops the detected plate from the image, passes the cropped plate image to the second inference stage, which detects plate





VIDEO



The screenshot shows a Node-RED interface with several nodes connected. One node displays the license plate image '6YOM172'. Another node below it shows the detected characters '6y gm172' with green bounding boxes around the characters '6', 'Y', ' ', 'g', 'm', '1', '7', and '2'. A third node at the bottom is labeled 'detect.chars.zinc'.

crops the detected plate from the image, passes the cropped plate image to the second inference stage, which detects plate characters.

The double stage detector uses a single detection model that has been trained to detect plates in full images containing cars/plates, and trained to detect plate text in images containing tightly cropped plate images.

<https://github.com/stavefield/tensorflow-anonhex/master/uploads/detect.chars.zinc>





# THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED flow editor window. On the left, there's a sidebar with categories: Input (top-request, udp in, udp out), Output (image, play audio), and Sequence (split, join, sort). The main canvas displays a flow starting with a "camera" node, which has two outputs. One output goes to a "Trigger Shutter" node, and the other goes to an "Original Image viewer" node. The "Trigger Shutter" node has one output that goes to a "retrieve the ph" node. The "Original Image viewer" node has one output that goes to a "pre-processing" node. A second "Original Image viewer" node is also present in the flow. The "pre-processing" node has one output that goes to another "Original Image viewer" node. The "retrieve the ph" node has one output that goes to this final "Original Image viewer" node. The "Edit tf-model node" dialog is open on the right, showing properties for a node named "ALPR SSD". The "Model URL" field contains the value "file:///Users/yhw@ng/WS.git/tf-2.0-test/alpr\_ssd\_". The "Name" field is set to "ALPR SSD". Other tabs in the browser include "Node-RED", "Node-RED : 192.168.1.158", "stevefielding/tensorflow-apr", "myq-hub.png (PNG Image, 1300 × 1000)", and "gopro.png (PNG Image, 2272 × 1000)".





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

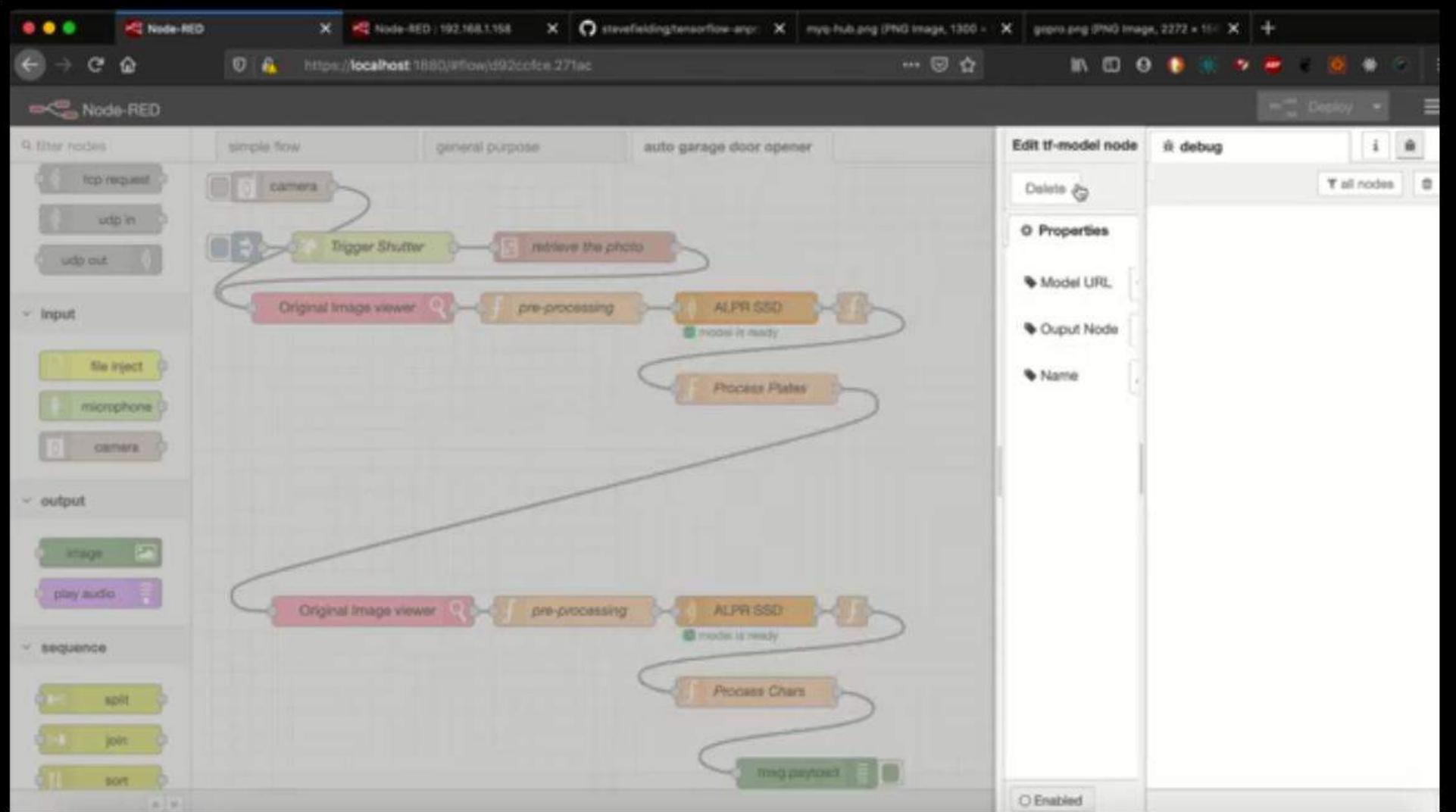
Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

## THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO





# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

Node-RED : 192.168.1.158 | https://localhost:1880/iFlow/d92ccfcf271ac | myo-hub.png (PNG Image, 1350 x 1000) | gopro.png (PNG Image, 2272 x 1000)

Deploy

auto garage door opener

camera → Trigger Shutter → retrieve the photo → Original Image viewer → pre-processing → ALPR SSD → Process Plates

Original Image viewer → pre-processing → ALPR SSD → Process Chars → msg payload

camera

file inject

microphone

camera

image

play audio

split

join

sort

Trigger Shutter

retrieve the photo

Original Image viewer

pre-processing

ALPR SSD

Process Plates

Process Chars

msg payload

BCSD320





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED flow titled "auto garage door opener". The flow starts with a "camera" node connected to a "Trigger Shutter" node. This triggers a "retrieve the photo" node, which then connects to an "Original Image viewer" node displaying a California license plate. The flow continues through "pre-processing", "ALPR SSD" (with a note "model is ready"), and "Process Plates" nodes. A second "Original Image viewer" node shows the processed license plate. The flow then branches into "Process Chars" and "msg payload" nodes. The "msg payload" node is connected to an "image" output node, which is shown in a preview window on the right. The sidebar on the left lists various input and output nodes.

```
graph LR; camera((camera)) --> trigger[Trigger Shutter]; trigger --> retrieve[retrieve the photo]; retrieve --> viewer1[Original Image viewer]; viewer1 --> preproc1[pre-processing]; preproc1 --> alpr1[ALPR SSD<br/>model is ready]; alpr1 --> process1[Process Plates]; process1 --> viewer2[Original Image viewer]; viewer2 --> preproc2[pre-processing]; preproc2 --> alpr2[ALPR SSD<br/>model is ready]; alpr2 --> process2[Process Chars]; process2 --> payload[msg payload]; payload --> image[image];
```





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED flow titled "auto garage door opener". The flow starts with a "camera" node connected to a "Trigger Shutter" node. The output of the "Trigger Shutter" node goes to a "retrieve the photo" node. This is followed by two parallel paths. The top path consists of an "Original Image viewer" node showing a California license plate, a "pre-processing" node, and an "ALPR SSD" node with a "model is ready" status. The bottom path also starts with an "Original Image viewer" node showing the same license plate, followed by "pre-processing" and "ALPR SSD" nodes. The outputs from both "ALPR SSD" nodes converge at a "Process Plates" node, which then connects to a "Process Chars" node. Finally, the output goes to an "msg.payload" node. On the left, there are filter nodes for "top-request", "udp in", and "udp out". On the right, a "debug" panel shows the timestamp "6/17/2020, 11:45:15 PM" and the message payload: "msg.payload: array[7] > [ "8", "C", "S", "D", "3", "2", "0 ]". A video player in the bottom right corner shows a person's face.



Share Your Experience on social - tag posts with #OSSummit #LFELC

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

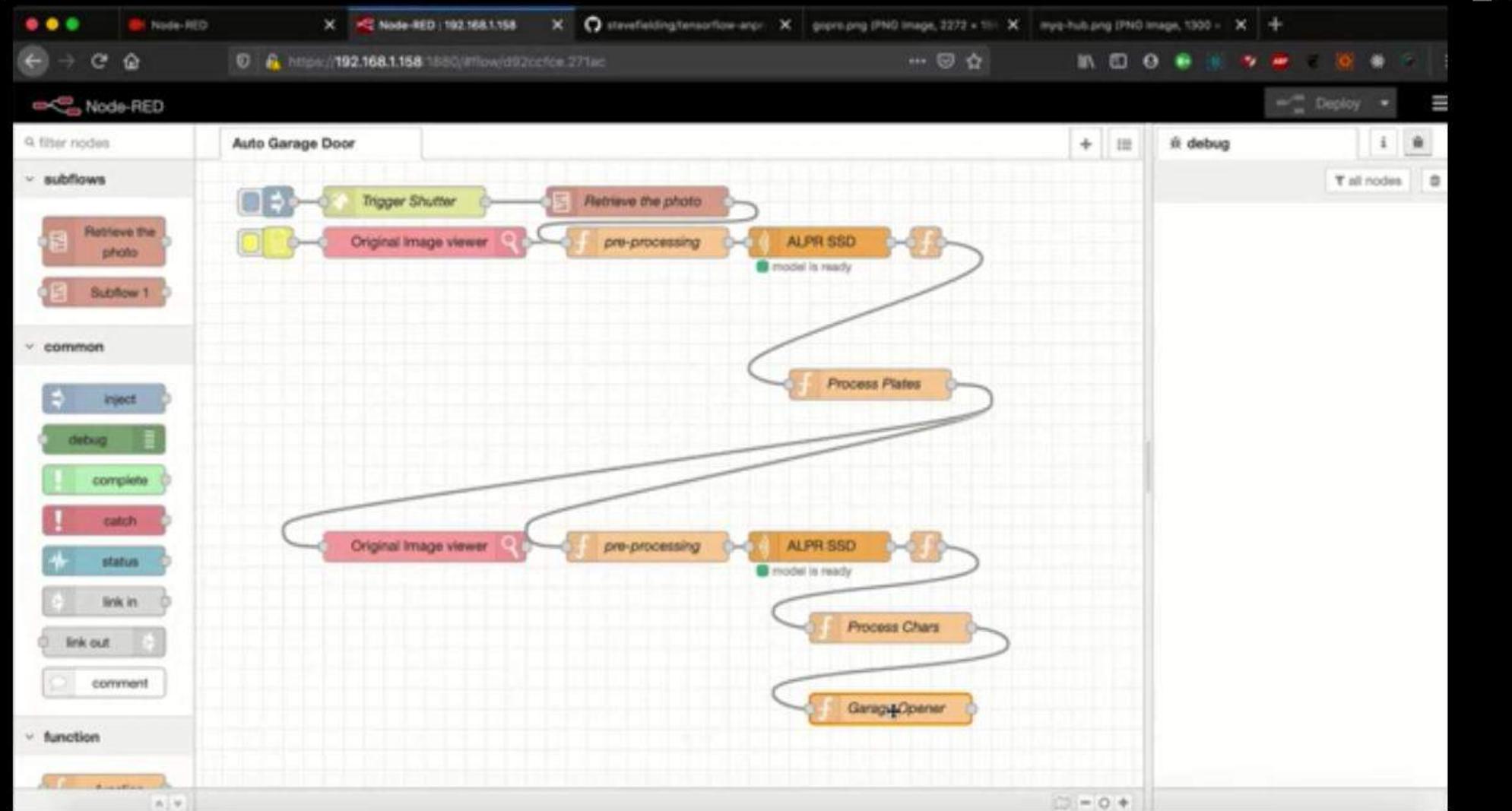
Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO



Visit the Sponsor Showcase!

powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

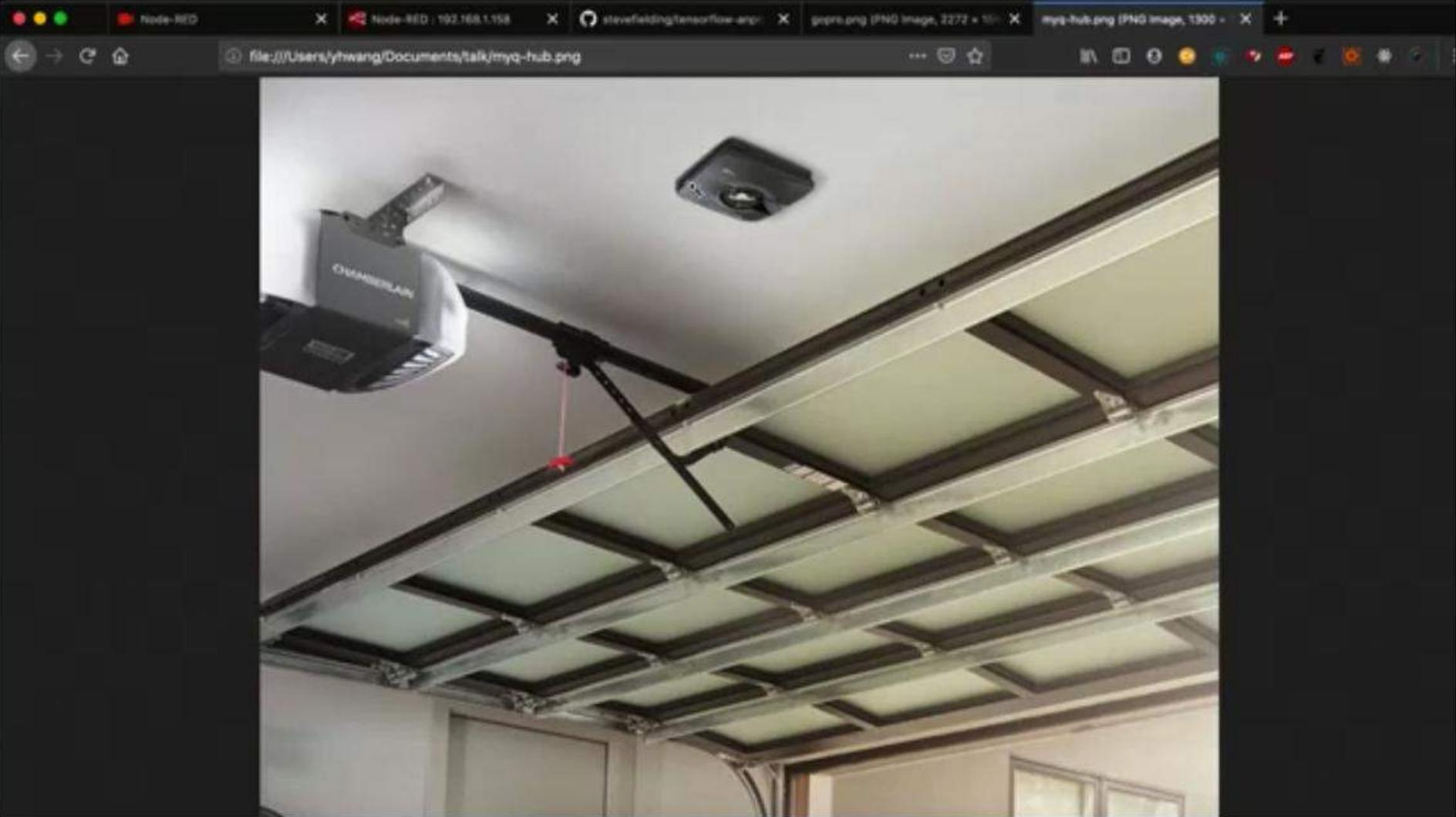
Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO



Connect with attendees on the OSS+ELC Slack Channel  
powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

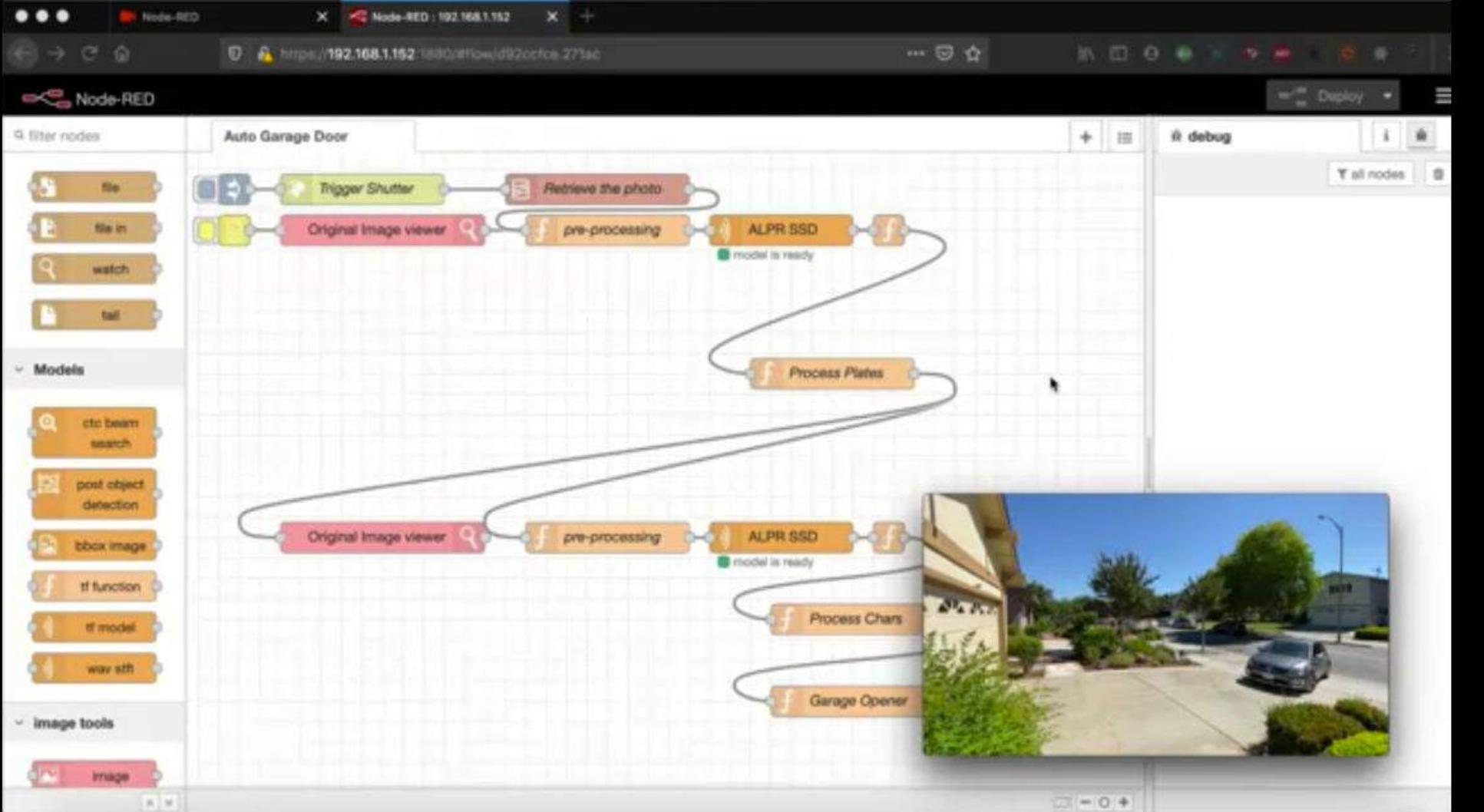
Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO





Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



# OPEN SOURCE SUMMIT NORTH AMERICA

# THE LINUX FOUNDATION Embedded Linux Conference North America

VIDEO

The screenshot shows a Node-RED flow titled "Auto Garage Door". The flow consists of two parallel processing paths. Both paths start with an "Original Image viewer" node showing a car in a driveway. This is followed by a "pre-processing" node and an "ALPR-SSD" node. The "ALPR-SSD" node has a green status indicator "model is ready". The output of the "ALPR-SSD" node is connected to a "Process Plates" function node. The output of the "Process Plates" node triggers a "Trigger Shutter" function node. The "Trigger Shutter" node is connected to a "Retrieve the photo" function node. The "Retrieve the photo" node is connected to another "Original Image viewer" node, which displays a close-up of a California license plate reading "8GUS598". This path also includes a "Process Chars" function node. The final output of the "Process Chars" node triggers a "Garage Opener" function node, which is shown opening a garage door. The "Garage Opener" node is also connected to a "Process Plates" function node. A "debug" tab on the right shows log entries from June 17, 2020, at 9:18:38 PM, indicating the model is ready and a number matched, triggering the garage door to open.

```
graph LR; In1[Original Image viewer] --> Pre1[pre-processing]; Pre1 --> ALPR1[ALPR-SSD]; ALPR1 -- "model is ready" --> Proc1[Process Plates]; Proc1 --> Trigger1[Trigger Shutter]; Trigger1 --> Photo1[Retrieve the photo]; Photo1 --> In2[Original Image viewer]; In2 --> Pre2[pre-processing]; Pre2 --> ALPR2[ALPR-SSD]; ALPR2 -- "model is ready" --> Proc2[Process Plates]; Proc2 --> GarageOpener[Garage Opener]; GarageOpener --> Proc3[Process Plates];
```

Nodes:

- file
- file in
- watch
- tail
- ctc beam search
- post object detection
- bbox image
- tf function
- tf model
- wav stt
- image

Models:

- ctc beam search
- post object detection
- bbox image
- tf function
- tf model
- wav stt

image tools:

- image

Debug Log:

```
6/17/2020, 9:18:38 PM node: Garage Opener
function : (main)
+ [ "B", "g", "u", "s", "5", "9",
"8" ]

6/17/2020, 9:18:38 PM node: Garage Opener
function : (main)
"number matched! open the door"
```



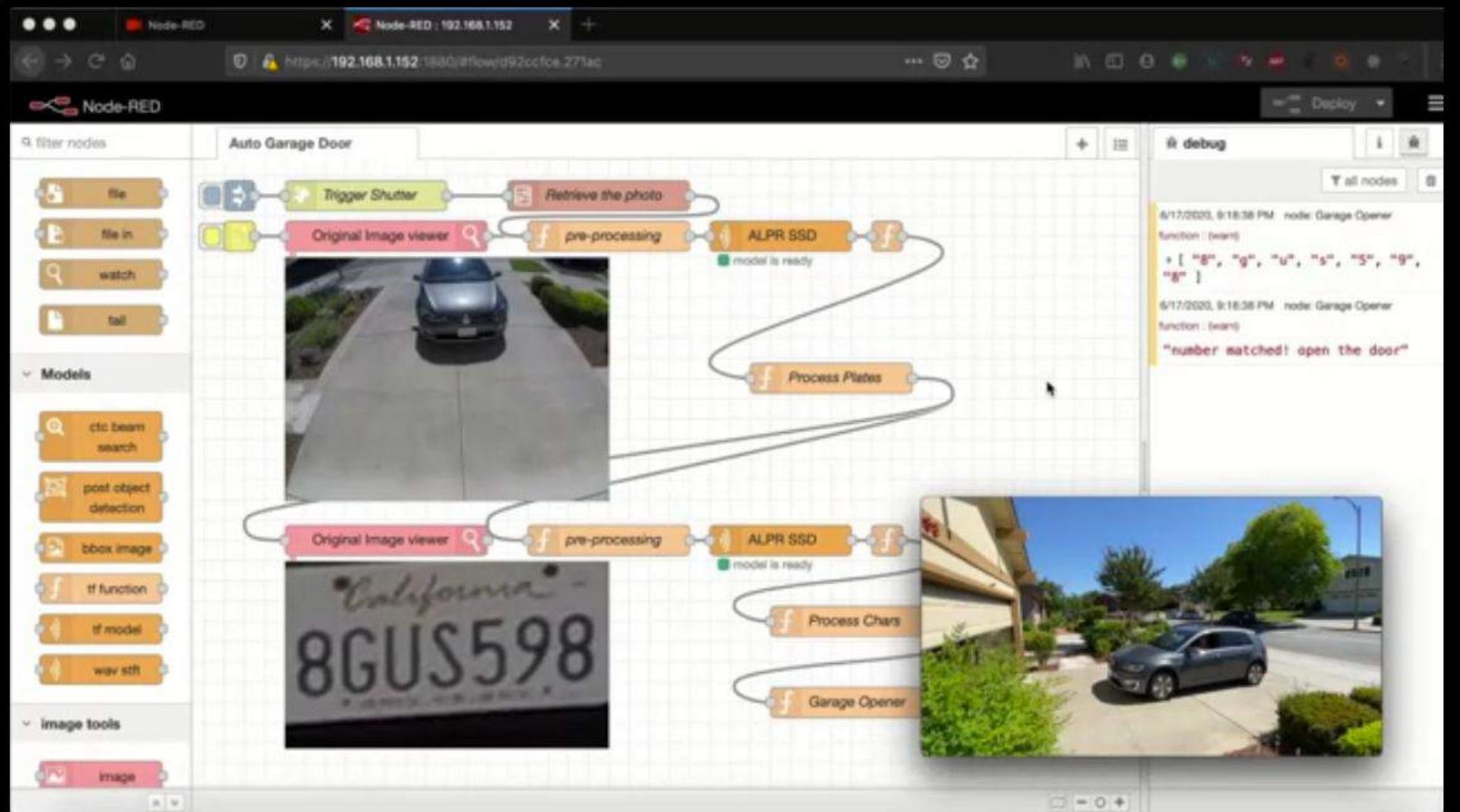
Take a break! Enjoy our Experiences + Fun &amp; Games!

powered by

# THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

 Embedded Linux  
Conference  
North America

**VIDEO**



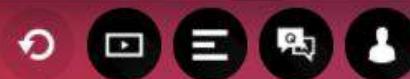
[Lobby](#)[Sessions](#)[Sponsor Showcase](#)[Networking](#)[Experiences](#)[Fun & Games](#)[Info](#)[Profile](#)

VIDEO

## Auto Garage Door



# Auto License Plate Recognition



Share Your Experience on social - tag posts with #OSSummit #LFELC

powered by Intrado



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



VIDEO

## Recap

### Node-RED

- Low-code platform for IoT
- Flexible and extensible
- Large NPM community

### TensorFlow.js

- Machine Learning in JavaScript
- Data Privacy
- Network connectivity

### Node-RED + TensorFlow.js

- Easy packaging of TF.js models for Node-RED usage
- Rapid building of AI-enabled IoT applications
- Leverage TensorFlow.js, Node-RED, Node.js, and NPM communities
- Democratize AI



Questions? Ask the Events team on the #1-helpdesk Slack channel  
powered by



Lobby

Sessions

Sponsor Showcase

Networking

Experiences

Fun &amp; Games

Info

Profile



VIDEO

## Links

<https://js.tensorflow.org>

<https://nodered.org>

<https://github.com/yhwang/node-red-contrib-tf-model>

<https://github.com/vabarbosa/tfjs-node-red>

<https://github.com/IBM/node-red-tensorflowjs>

<https://medium.com/codait/a-low-code-approach-to-incorporating-machine-learning-into-your-iot-device-24f3f2a70717>



Share Your Experience  
powered by Infradeck

[Lobby](#)[Sessions](#)[Sponsor Showcase](#)[Networking](#)[Experiences](#)[Fun & Games](#)[Info](#)[Profile](#)

VIDEO



# Thank you

Yi-Hong Wang | IBM Cognitive OpenTech

[twitter.com/@blacklet](https://twitter.com/@blacklet)  
[github.com/yhwang](https://github.com/yhwang)

Va Barbosa | IBM CODAIT

[twitter.com/@vabarbosa](https://twitter.com/@vabarbosa)  
[github.com/vabarbosa](https://github.com/vabarbosa)



Share Your Experience on social - tag posts with #OSSummit #LFELC  
powered by

[Lobby](#)[Sessions](#)[Sponsor Showcase](#)[Networking](#)[Experiences](#)[Fun & Games](#)[Info](#)[Profile](#)

# THE LINUX FOUNDATION OPEN SOURCE SUMMIT NORTH AMERICA

# Embedded Linux Conference North America

[Q&A WITH SPEAKER](#) [SPEAKER BIO](#)

**Yi-Hong Wang**  
Software Developer  
IBM

YiHong is a software developer with the Cognitive OpenTech Group at the IBM Silicon Valley Lab. He has been an active contributor in the TensorFlow community for the past 2 years. He is also a committer in the node.JS community.



**Va Barbosa**  
Software Developer  
IBM

Va is a software developer with the Developer Advocacy Group at IBM. He has been an active contributor in the TensorFlow community for the past 2 years.

**VIDEO**