Node-RED AI Photo Booth Workshop

OpenJS World 2021

John Walicki

None

Table of contents

1. Node-RED AI Photo Booth Workshop

1.1 Welcome to the Node-RED AI Photo Booth Workshop

In this workshop, participants will learn how to use Node-RED to create a photo booth web app infused with AI through the use of TensorFlow. The workshop will step through getting started with Node-RED, creating the web app and then containerizing it, ready to be deployed into the cloud or onto edge devices.

1.2 Prerequisites

This workshop requires:

- A laptop/computer with a Web Cam attached
- Node.js 12.x or 14.x
- Git

1.3 Navigation

To move through the workshop you can use the side panels to select a specific section or use the navigation links at the bottom of each page to move to the next or previous section as required.

1.4 Access to workshop material

The source for this workshop is hosted on GitHub and this site is automatically generated from it. The repository also contains examples and other content that can be used through the workshop. You may want to clone the repository to your local computer so you have them readily available.

You can also download a PDF version of this workshop here.

1.5 Getting help

If you need help with the workshop, join the `#openjs_world-ibm_workshops` channel on the OpenJS Foundation Community slack.

1.6 Workshop Outline

1.6.1 1 - Node-RED

The first part of the workshop introduces Node-RED - the low-code programming tool for event-driven applications. It will help you:

- get Node-RED running on your local computer
- enable the Projects feature
- learn how to install additional nodes into its palette
- create a simple application to learn how Node-RED works

If you are already familiar with Node-RED, you can skip this part.

1.6.2 2 - Node-RED Dashboard

In this part you will install the Node-RED Dashboard set of nodes and learn how to quickly create a simple photo booth application using them.

1.6.3 3 - TensorFlow in Node-RED

This part brings TensorFlow into Node-RED. It will look at some of the different nodes for TensorFlow that are available from the community and compares their capabilities.

You will then integrate the TensorFlow nodes into your photo booth application.

1.6.4 4 - Summary and next steps

Finally, we'll review what you have covered in this workshop and highlight a number of areas where the application you've created could be expanded as a follow-on activity.

2. 1 - Node-RED

2.1 Introducing Node-RED

Node-RED is a programming tool for building event-driven application. It takes a low-code approach - which means rather than write lots of code, you build applications by using its visual editor to create flows of nodes that describe what should happen when particular events happen.

For example, the `HTTP In` node can be configured to listen on a particular path. When an HTTP request arrives on that path the node is triggered. It generates a message containing information about the request and passes it on to the nodes it is wired to. They in turn do whatever work they need to do using the message.

Such as generating HTML content and adding it to the message before being passed on through the flow. In this example, the flow ends with an `HTTP Response` node which responds to the original HTTP request using the information in the message.

2.1.1 Next Steps

In this part of the workshop you will:

- Install Node-RED
- Enable the Projects feature
- Install extra nodes into the palette
- Create a simple flow
- Commit your changes

2.2 Installing Node-RED

Node-RED is published as a node.js module available on npm, as well as a container available on Docker Hub.

The full guide for installing and running Node-RED is available here.

> **Linux**
>
> The following steps assume you are running on Windows or OSX. If you are running on a Linux OS, or a device like a Raspberry Pi, the project provides a set of install scripts that will get node, npm and Node-RED all installed at the latest stable versions. Refer to the docs linked above.

You must have a supported version of Node.js installed. Node-RED supports the Active and LTS releases, 12.x and 14.x.

You can then install Node-RED as a global module with the command:

```
npm install -g --unsafe-perm node-red
```

Depending on your Node.js installation, you may need to run this command using `sudo`.

The install log output may contain some warnings - these can be ignored as long as the output ends with something like:

```
+ node-red@1.2.2
added 332 packages from 341 contributors in 18.494s
```

2.2.1 Running Node-RED

Once installed, you should now have the `node-red` command available to run.

> **Command not found**
>
> If you do not have the `node-red` command available it may be a problem with your `PATH` configuration.
>
> Find where your global node modules are installed by running:
>
> ```
> npm get prefix
> ```
>
> Then ensure the `bin` subdirectory of that location is on your `PATH`.

When you run `node-red`, the log output will appear

```
23 Oct 00:12:01 - [info]

Welcome to Node-RED
===================

23 Oct 00:12:01 - [info] Node-RED version: v1.2.2
23 Oct 00:12:01 - [info] Node.js  version: v12.19.0
23 Oct 00:12:01 - [info] Darwin 18.7.0 x64 LE
23 Oct 00:12:01 - [info] Loading palette nodes
23 Oct 00:12:03 - [info] Settings file  : /Users/nol/.node-red/settings.js
23 Oct 00:12:03 - [info] User directory : /Users/nol/.node-red
23 Oct 00:12:03 - [info] Server now running at http://127.0.0.1:1880/
23 Oct 00:12:03 - [info] Flows file     : /Users/nol/.node-red/flows.json
23 Oct 00:12:03 - [info] Starting flows
23 Oct 00:12:03 - [info] Started flows
```

This output contains an important piece of information you will need - the location of your `User directory`.

2.2.2 Accessing the Node-RED editor

Assuming you are running Node-RED on your local computer, open a browser and access the url http://127.0.0.1:1880/. This will load the Node-RED editor - the tool used to build your applications.

2.2.3 Next Steps

The next task is to enable the Projects feature.

2.3 Enabling the Projects feature

Node-RED comes with the Projects feature that allows you to version control your flows by creating a git repository around them. You can then commit changes from directly within the editor. You can also connect to a remote repository and push or pull changes to that remote.

This feature needs to be enabled before it can be used.

1. Stop Node-RED by pressing `Ctrl-C` in the terminal window its running in.
2. Find your Node-RED `settings.js` file in your User directory. By default this will be `~/.node-red/settings.js`. Open it in a text editor.
3. Find the `editorTheme` section at the bottom of the file. Set the `enabled` property to `true`:

```
editorTheme: {
    projects: {
        // To enable the Projects feature, set this value to true
        enabled: true
    }
}
```

4. Save the file and restart Node-RED

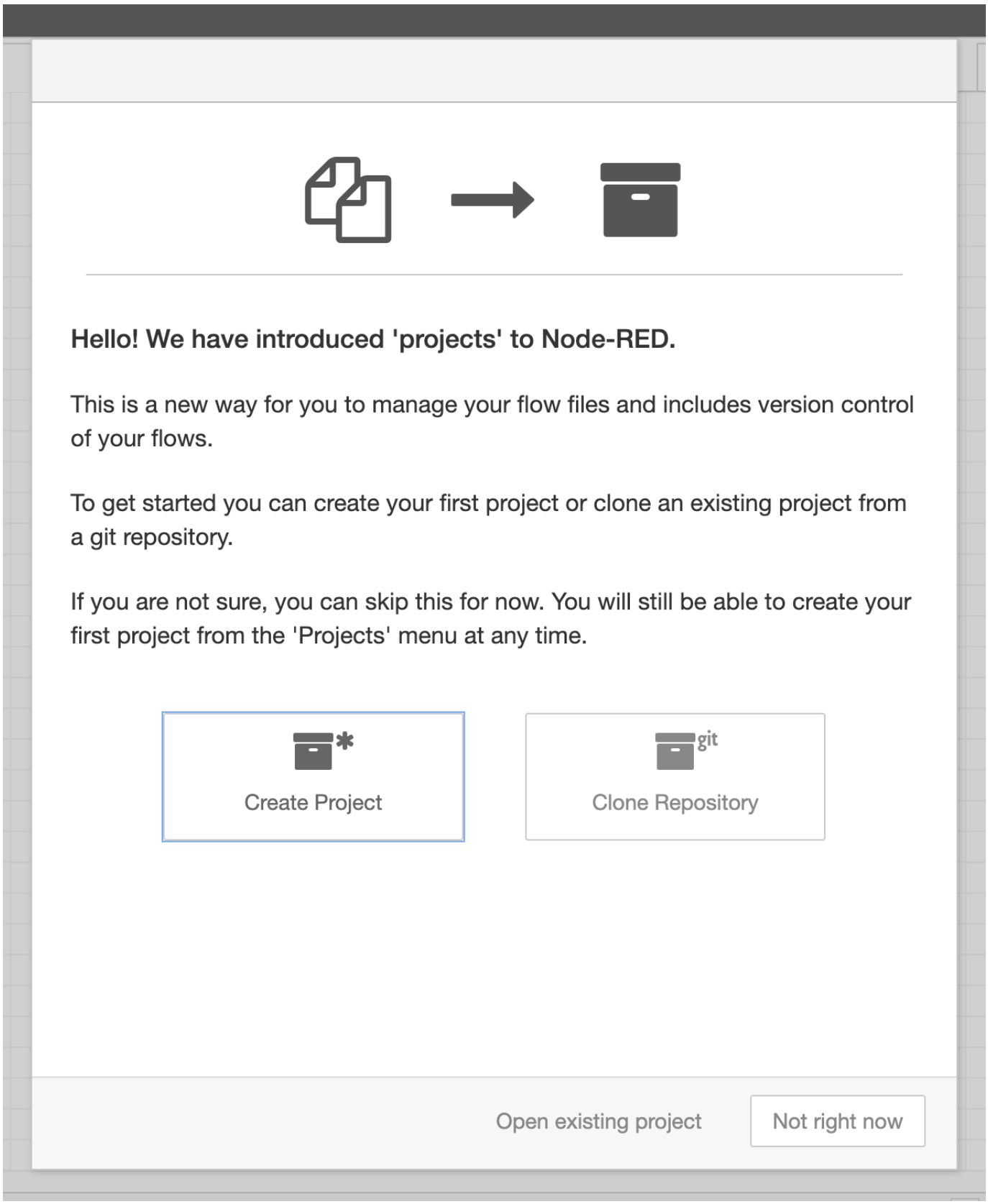
The log output should now include the line:

```
23 Oct 10:49:09 - [warn] No active project : using default flows file
```

> **Troubleshooting**
>
> If you see a line saying `[warn] Projects disabled` then you are missing a prerequisite. The line should tell you what is missing. For example, if it says `git command not found` then you need to install the `git` command-line tool and ensure its on your path before running Node-RED.

2.3.1 Creating a Node-RED Project

Once you have enabled the Projects feature, the next time you load the editor in your browser, you will be greeted with a dialog inviting you to create your first project.



Click the `Create Project` button and follow the steps it takes you through:

```
- Setup your username/email used to create commits
- Give your project a name and an optional description
- Set the flow file name to `flows.json`
- Configure the encryption of your credentials file.
```

The project will then be created under `~/.node-red/projects/<name-of-project>`.

2.3.2 Next Steps

The next task is to install some extra nodes into the palette.

2.4 Installing Nodes

The building blocks of any Node-RED application are the nodes in its palette.

Node-RED comes with a number of core nodes that provide the basic components, but the palette can be easily extended by installing additional nodes.

Nodes are published as npm modules and the project provides an online catalogue of them at https://flows.nodered.org.

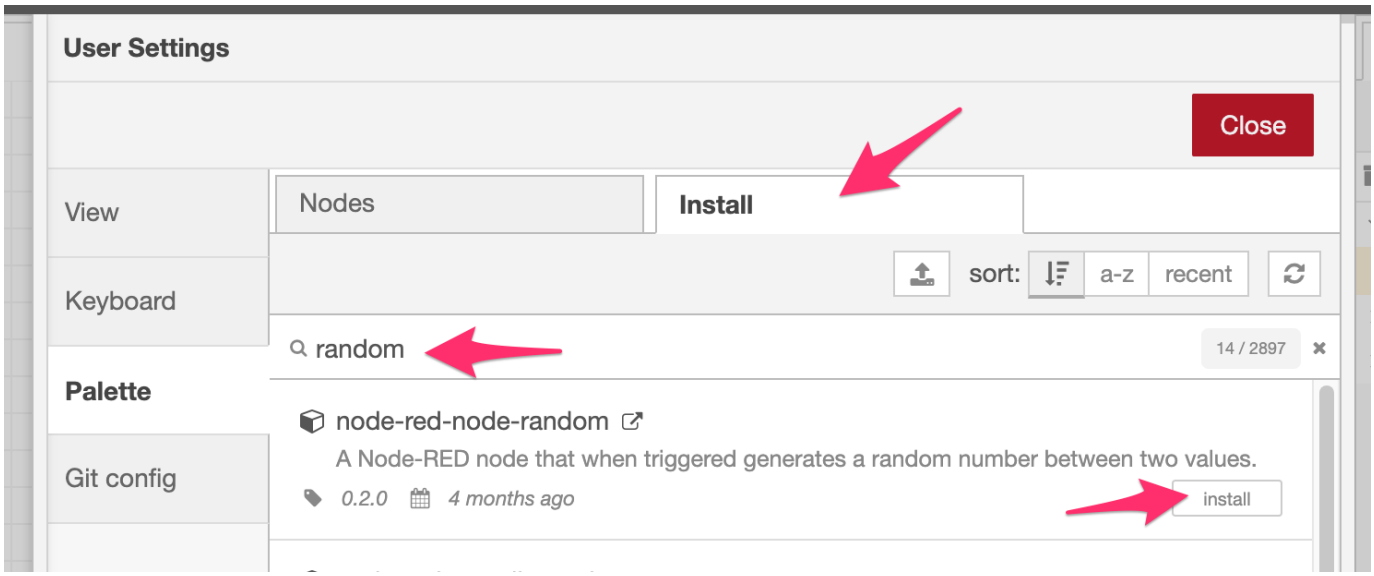
There are two ways to install nodes - via the command-line or from within the Node-RED editor.

Node-RED Palette Manager

To install a node from within the editor, select the Manage Palette option from the main menu.

This opens the Palette Manager which shows two tabs - a list of the modules you have installed and a searchable catalogue of modules available to install.

Switch to the Install tab and search for `random` - you should see `node-red-node-random` in the list below. Click the `install` button next to it.



After a short time the node will be installed and added to the palette.

Command-line

To install on the command-line, switch to the Node-RED user directory and run the appropriate `npm install` command. For example:

```
npm install node-red-node-random
```

> **Node-RED User Directory**
>
> By default, Node-RED creates a directory called `.node-red` in the user's home directory. As it starts with a `.` it may be hidden from view by your file browser.
>
> As mentioned in the Install section, Node-RED logs the full path to the user directory when it starts up. If in doubt, check what it says.

> **Note**
>
> Some nodes will have external dependencies that cannot be automatically installed by Node-RED or npm. You should always check a module's readme for further information. This will be particularly true of some of the TensorFlow nodes we'll be using later in this workshop.

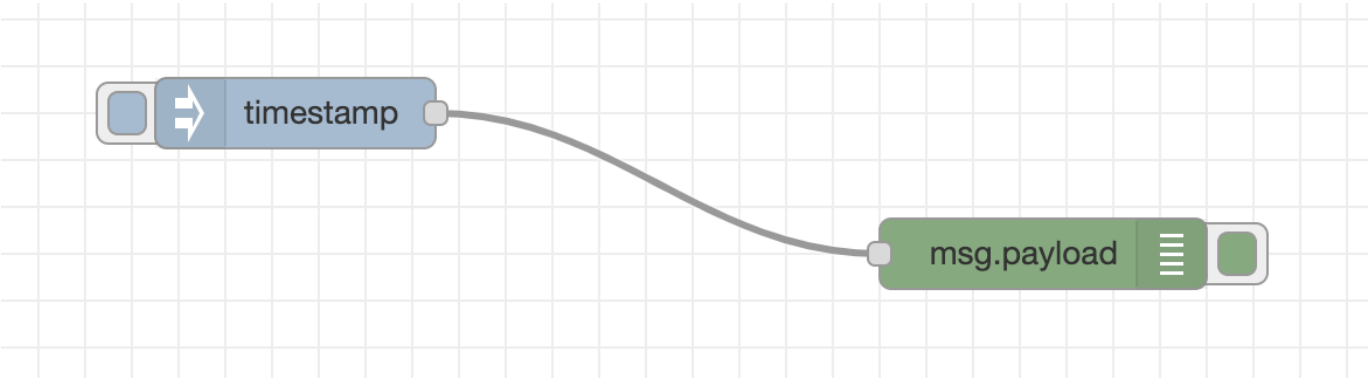
2.4.1 Next Steps

The next task is to create you first flow in Node-RED.

2.5 Creating a flow

Before we start on the photo booth application, we're going to step through a quick example to help get you more familiar with Node-RED.

1. From the palette, drag an `Inject` node into workspace. This node can be used to manually inject messages into a flow, or to inject them at a regular interval.
2. Drag a `Debug` node into the workspace. This is a very useful node that can be used to examine messages in a flow by displaying them in the Debug sidebar.
3. Drag a wire from the output of the `Inject` node to the input of the `Debug` node. The wires determine where messages go when they pass from one node to another.
4. Click the red Deploy button to save your changes. At this point your flow is now running.
5. Click the button to the left of the Inject node - this triggers it to send a message. By default it will send a message with its `payload` set to the current time.
6. You should see the message get displayed in the Debug sidebar.



Core Nodes

Node-RED comes with a number of core nodes that are the basic building blocks for any flow. It's worth spending a bit of time exploring what is available.

The documentation has more information about them.

Debug sidebar

The Debug node and sidebar are invaluable tools when creating flows. They help you understand the structure of the messages you are working.

The "Working with messages" section of the documentation gives a good introduction in how to make the most of the Debug tools.

2.5.1 Import/Exporting Flows

Node-RED flows can be imported and exported from the editor using a JSON format.

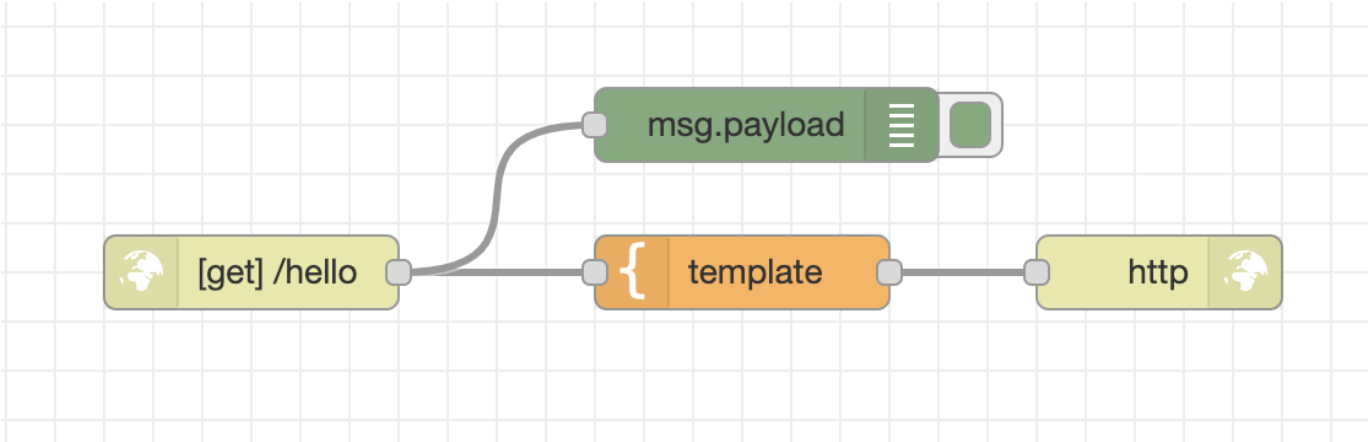
To import a flow, select the Import option from the menu (or hit `Ctrl-I`). This opens the import dialog.

Copy the JSON below, paste it in and click `Import`. You will find a number of nodes attached to your mouse that you can click to place into the workspace.

```
[{"id":"10cd970c.dcde99","type":"http in","z":"ddeb3b89.ad9748","name":"",
"url":"/hello","method":"get","upload":false,"swaggerDoc":"","x":140,"y":140,
"wires":[["ec1f6830.222e38","43f8e071.77d4a"]]},{"id":"ec1f6830.222e38","type":"template",
"z":"ddeb3b89.ad9748","name":"","field":"payload","fieldType":"msg","format":"handlebars",
"syntax":"mustache","template":"{{# payload.name }}\n<p>Hello, {{{payload.name}}}!</p>\n{{/ payload.name }}\n{{^ payload.name }}\n<form action=\"/hello\"
method=\"GET\">\n    What is your name? <input name=\"name\">\n    <button>Submit</button>\n</form>\n\n\n{{/ payload.name }}\n",
"output":"str","x":340,"y":140,"wires":[["87f3adff.c4e43"]]},{"id":"87f3adff.c4e43",
"type":"http response","z":"ddeb3b89.ad9748","name":"","statusCode":"","headers":{},"x":510,
"y":140,"wires":[]},{"id":"43f8e071.77d4a","type":"debug","z":"ddeb3b89.ad9748",
"name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,
"complete":"false","statusVal":"","statusType":"auto","x":350,"y":80,"wires":[]}]
```

This gives you a flow that starts with a `HTTP In`, configured to listen for requests on `/hello`. When a request arrives, it gets passed to a `Template` node that generates some simple HTML using the contents of the message. It then gets passed to a `HTTP Response` node to send back the response.

You can see it in action by opening http://localhost:1880/hello.



2.5.2 Next Steps

The final task in this part of the workshop is to commit your changes.

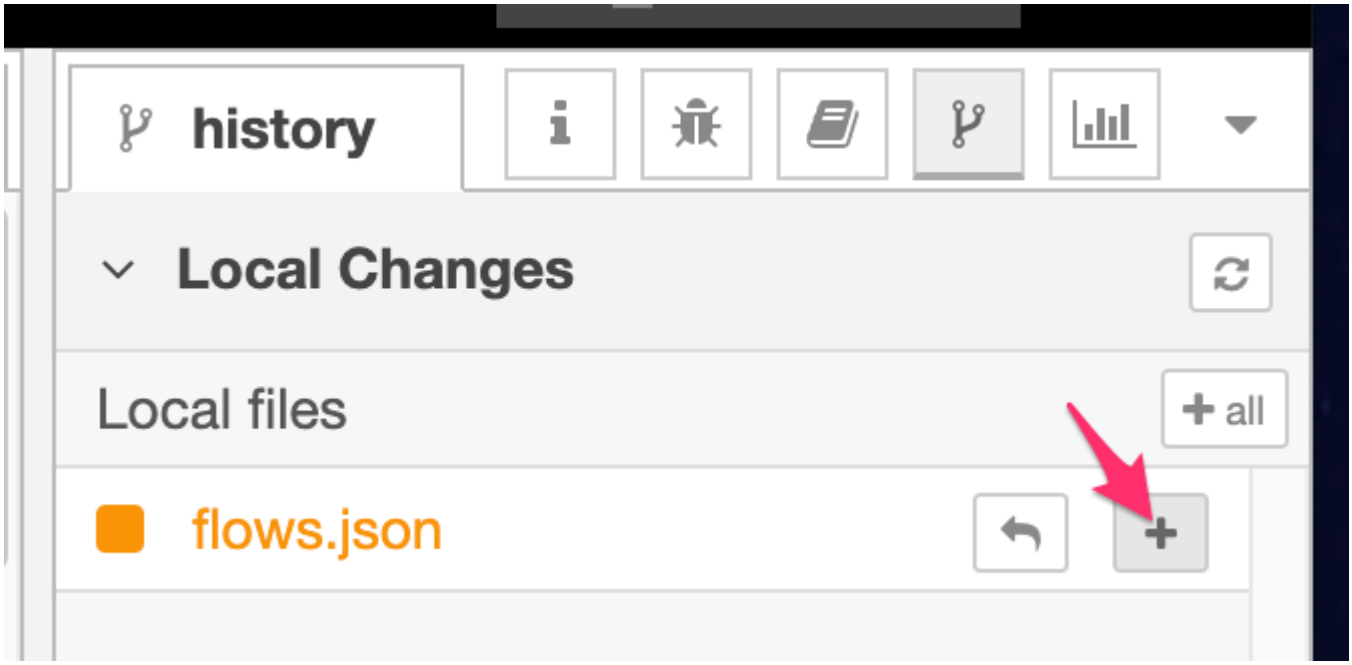
2.6 Committing Changes

Before we move on with the workshop, the next thing to learn about is how to commit changes you've made in your project.

Note

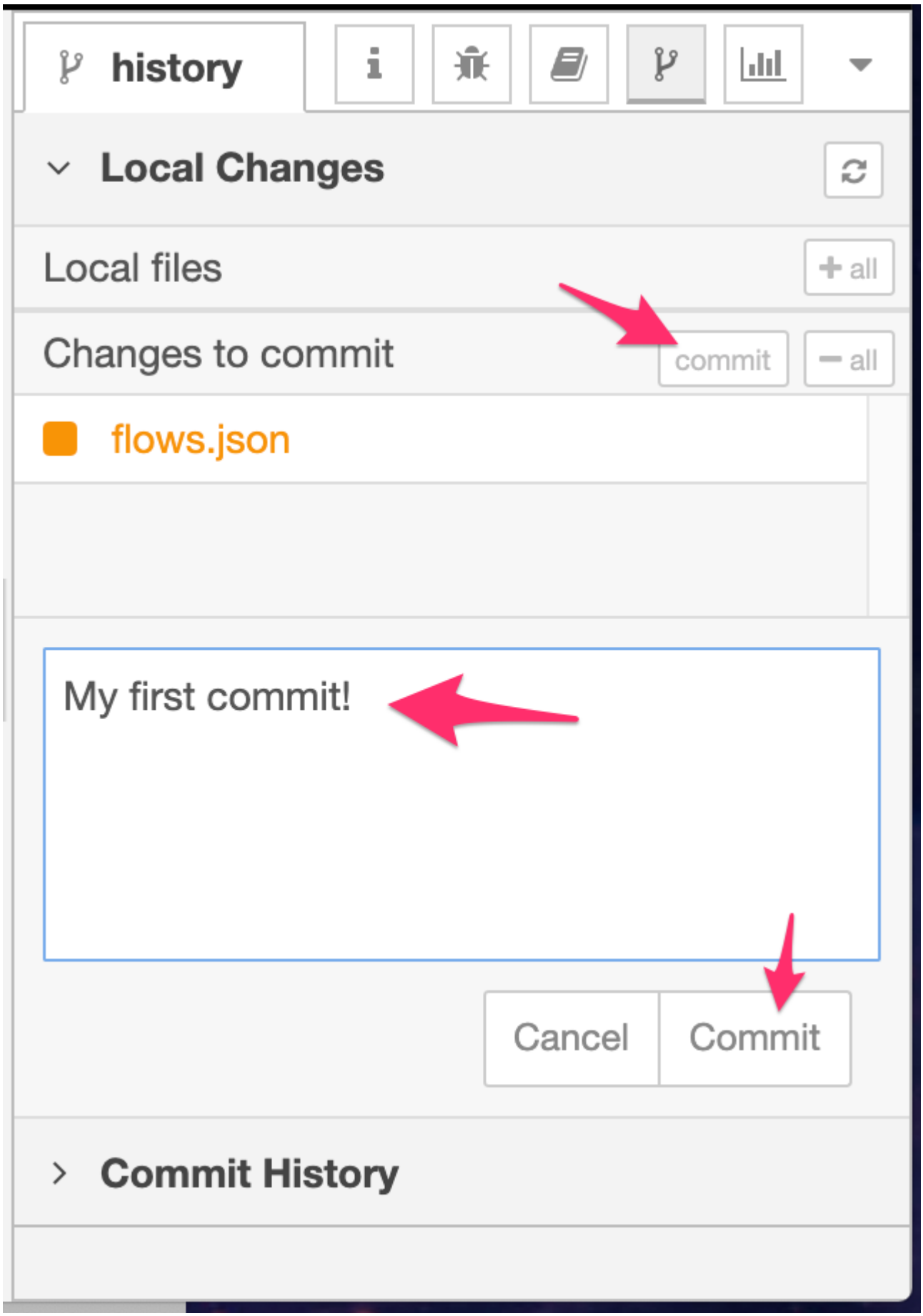
If you skipped over the previous step to create a flow, you won't have any changes to commit. Add a node to your workspace and click the Deploy button so you have a change to commit.

1. Open the History sidebar tab. This is where you can see the changes to your project that are ready to be staged and committed to the git repository.
2. You should see your flow file listed in the Local Files section. Clicking on it will open a dialog showing the changes to the file since it was last committed.

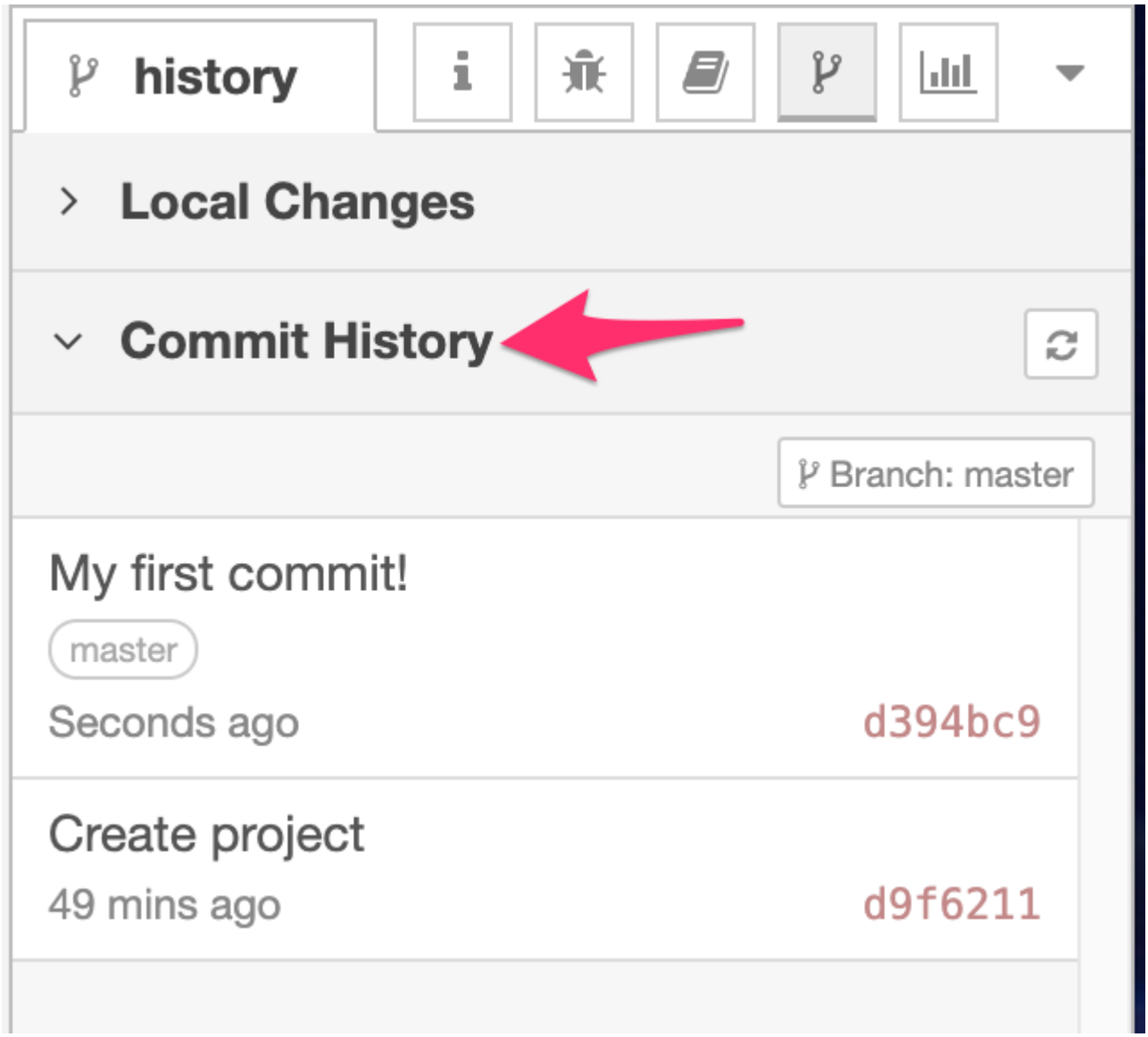


3. Click the `+` that appears when you hovered over the file name. The entry will move to the 'Changes to commit' section.
4. Do the same for any other files in the Local Files section.
5. Click the `commit` button at the top of the 'Changes to commit' section.

6. Enter a commit message and click the Commit button.


history
Local Changes
Local files
all
Changes to commit
commit
all
flows.json
My first commit!
Cancel
Commit
Commit History

7. Expand the 'Commit History' section of the sidebar. This lists the history of commits for the project. At this point there should be two commits - the original commit from when the project was created, and the one you've just created.



2.6.1 Next Steps

Your Node-RED environment is all setup now for the rest of this workshop.

In the next part, we'll start looking at Node-RED Dashboard.

3. 2 - Node-RED Dashboard

3.1 Introducing Node-RED Dashboard

Node-RED Dashboard is a set of nodes you can install into Node-RED that make is easy to create a web page that can interact with your flows.

It comes with nodes to add buttons, text inputs, charts and other widgets.

It does not provide the full flexibility of creating a page from scratch, but it is a good choice for getting something created quickly and easily.

> **Other dashboard options**
>
> There are other sets of nodes available from the community that can be used to create web pages connected to Node-RED flows. One such example is UIbuilder that does not provide the pre-build widgets of Node-RED Dashboard, but does allow you to build the page from your own HTML/JavaScript/CSS.
>
> We use Node-RED Dashboard in this workshop for the convenience it provides in building the web page with a minimum of any custom coding.

3.1.1 Installing Node-RED Dashboard

Using the Manage Palette feature in the editor, install the following modules:

- `node-red-dashboard` - be sure to install this one *first*.
- `node-red-node-ui-table`
- `node-red-node-ui-webcam`

3.1.2 Next Steps

Having installed the dashboard nodes, in this part of the workshop you will:

- Create your initial Photo Booth Dashboard
- Add more controls to the dashboard

3.2 Create a dashboard

In this part of the workshop you will begin to create your Photo Booth application.

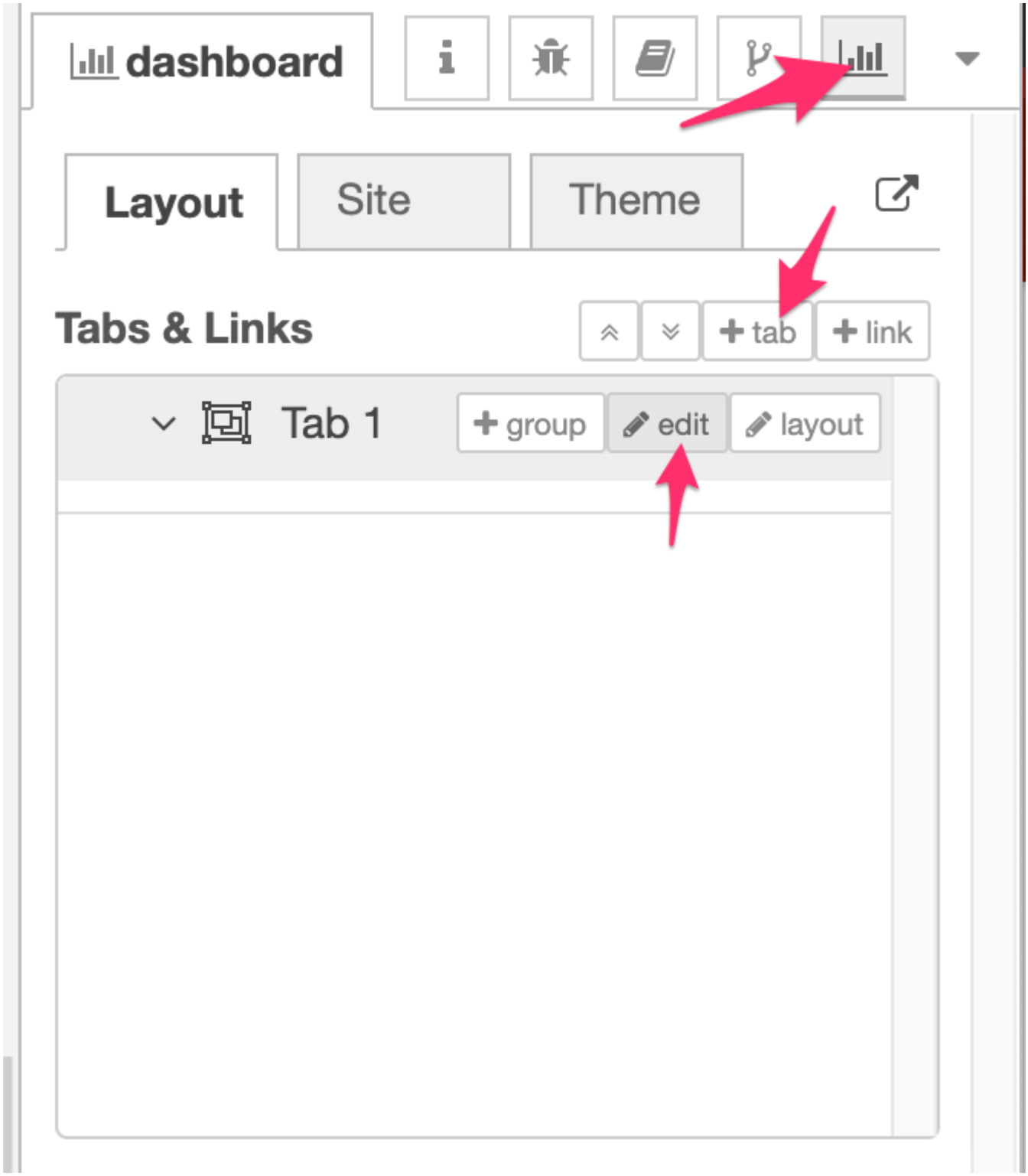
3.2.1 Dashboard Layouts

The Dashboard uses a grid based layout. Widgets, such as buttons or text boxes, are given a size in grid-cells. They are packed into a group that has a defined width. The Groups are then placed on a Tab - laid out using a flow-based layout, filling the width of the page before wrapping. This arrangement provides some flexibility in how the page displays on different screen sizes.

3.2.2 Dashboard Sidebar

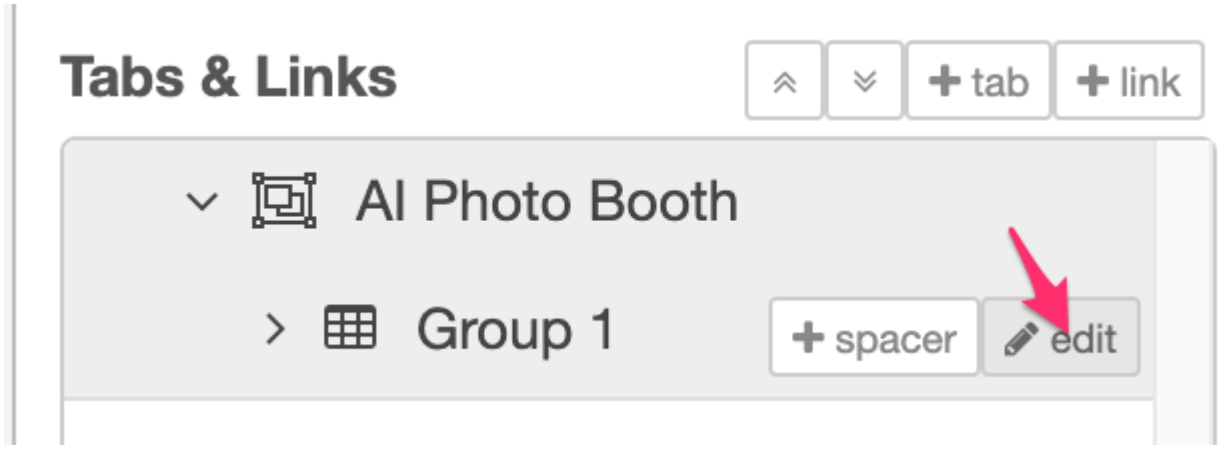
Within the editor, the Dashboard module provides a sidebar where you can customise its features as well as manage its tabs, groups and widgets.

1. Open the Dashboard sidebar
2. Click the `+ tab` button to create a new tab.
3. Hover over the newly added tab and click the `edit` button.

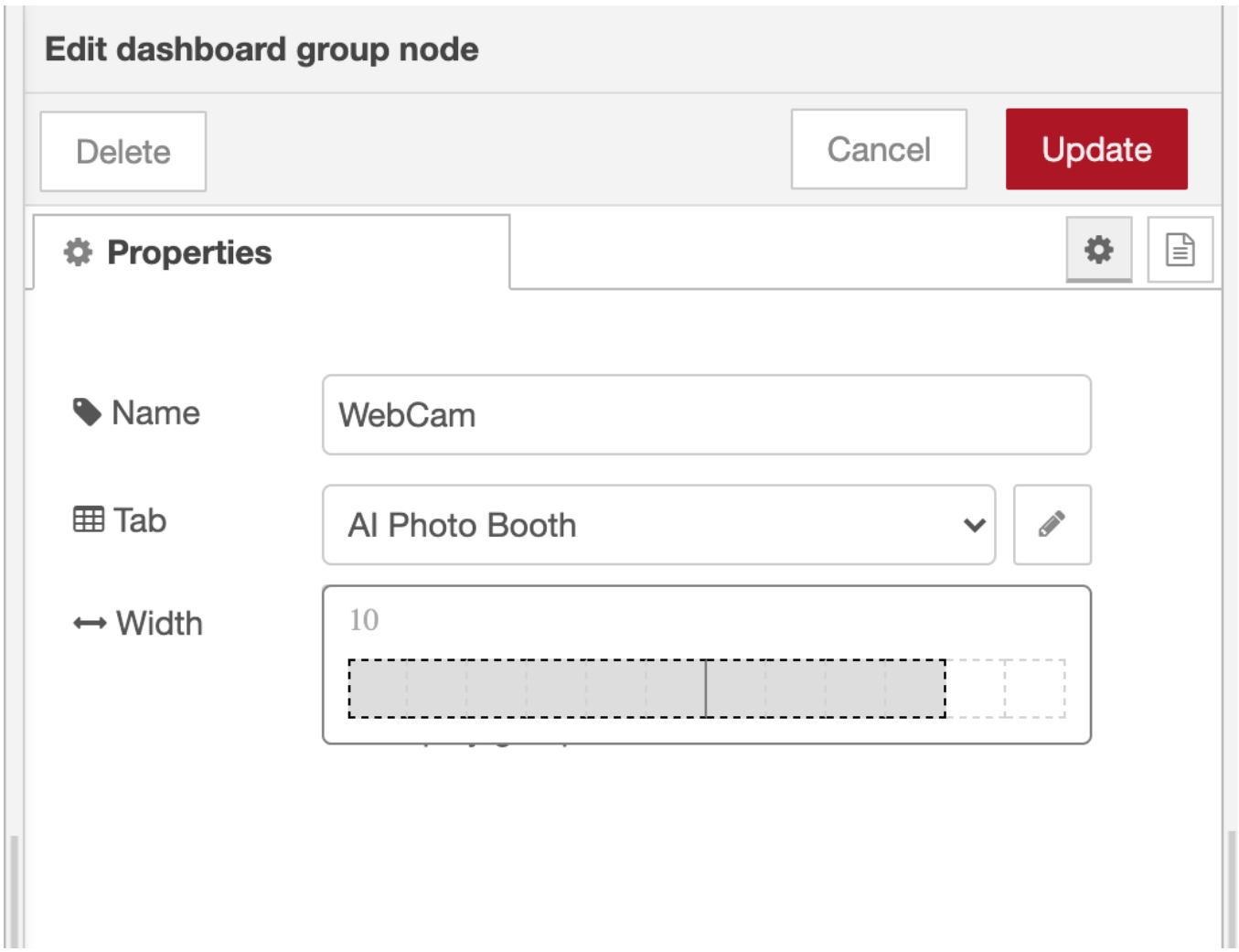


4. In the edit dialog, give the tab a name of `AI Photo Booth` and click Update to close the dialog.
5. Hover over the tab again and click the `+ group` button.

6. Edit the new group and set its properties:



- Set the name to `WebCam`
- Set the width to `10` by clicking the button and dragging the box out to 10 units wide.
- Untick the 'Display group name' option.

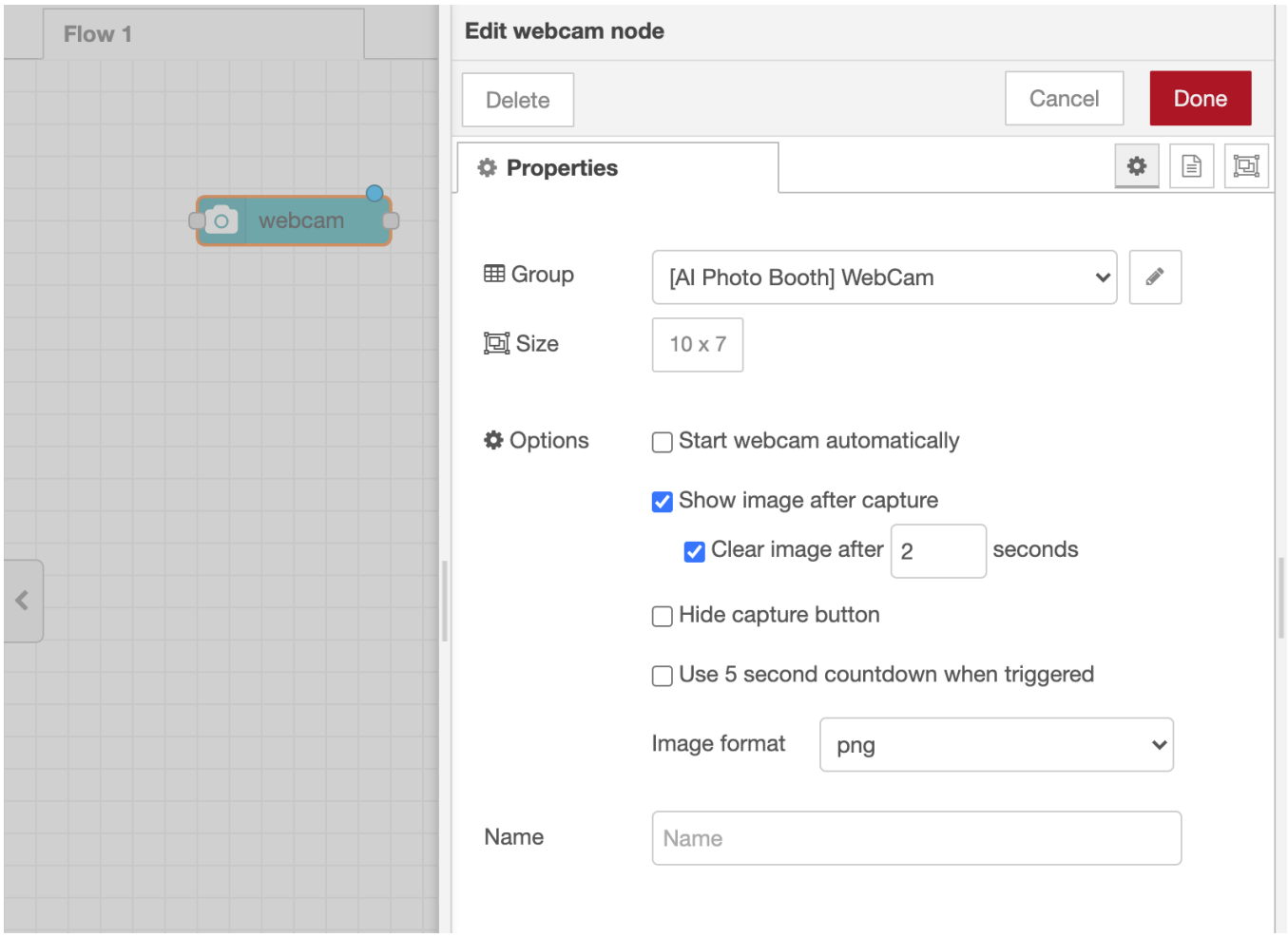


This has created the initial layout components needed for the dashboard. You can now start to add content.

3.2.3 Adding a `ui_webcam` node

> **Tidy up existing flows**
>
> If you followed the previous part of the workshop, you'll have some example flows in your workspace. You can delete those flows as you won't need them for the rest of the workshop.

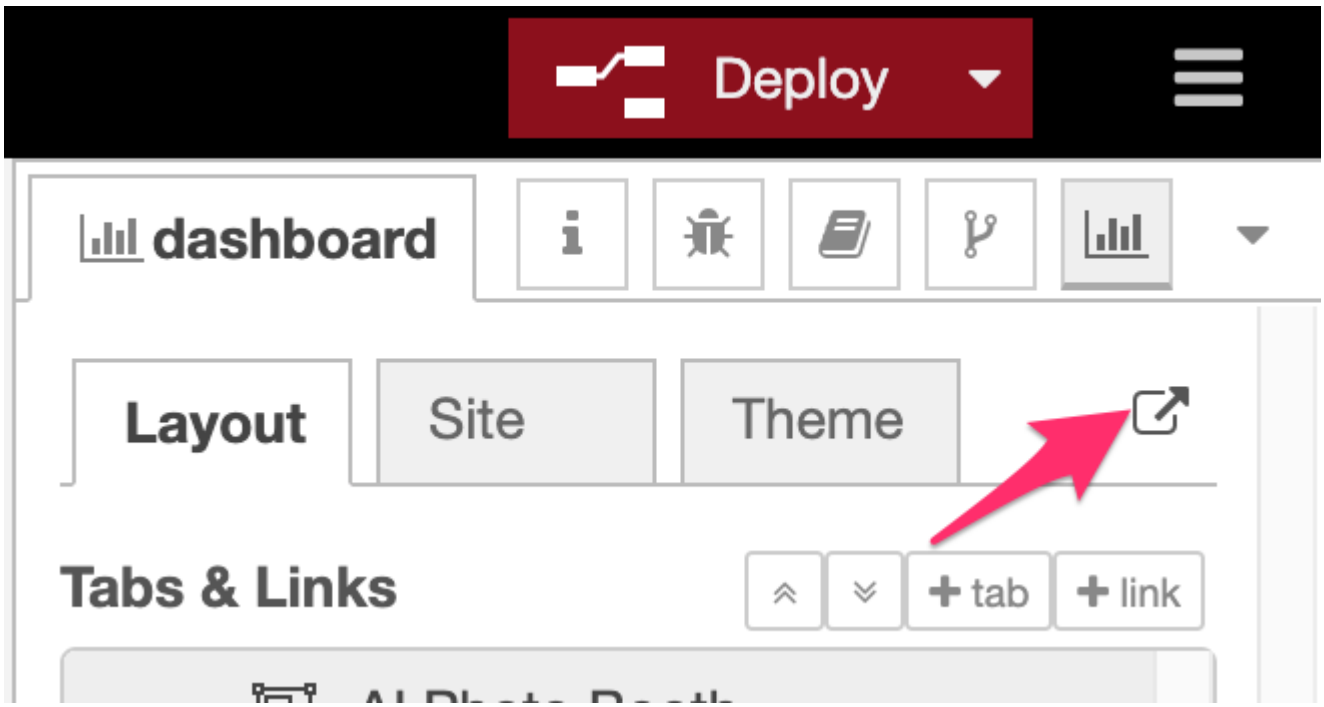
1. Drag a `ui_webcam` node from the `dashboard` category of the palette into your workspace. This node can be used to capture images from the webcam on the device displaying the dashboard.



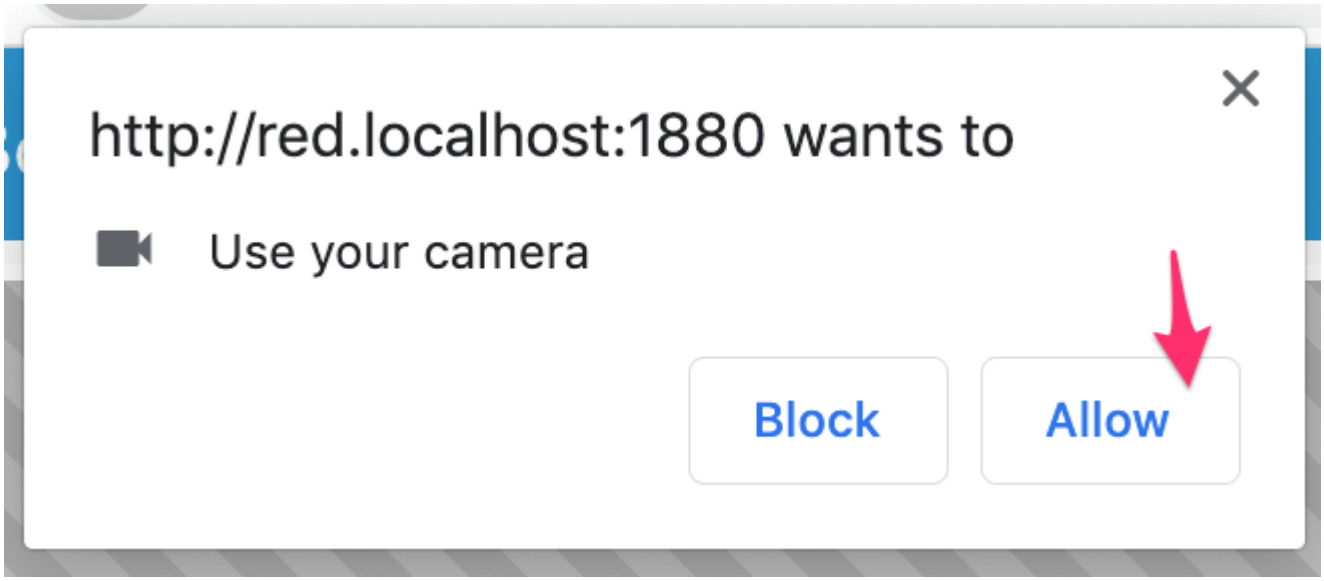
2. Double click on the node to edit its properties.
3. Make sure the `WebCam` group you created earlier is selected in the select box.
4. Set the size to `10x7` - note that it will not let you make it wider than the group it is in.
5. Leave the rest of the options as their defaults for now and click Done.

   At this point you have created a dashboard with a single tab, containing a single group that contains a webcam widget.
6. Click the Deploy button to save your changes.

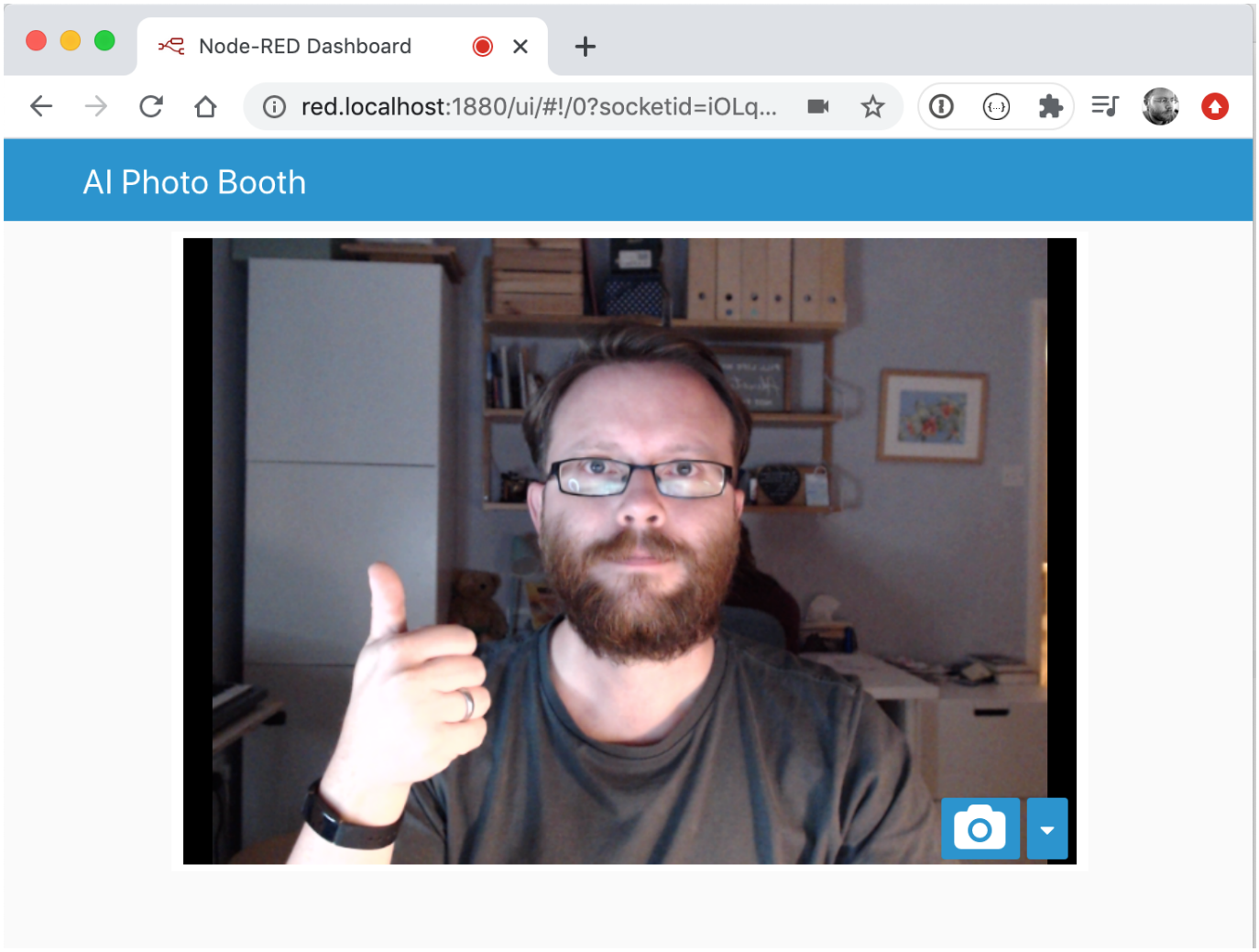
To access the Dashboard, click the button in the top-right corner of the Dashboard sidebar. This will open the Dashboard in a new browser tab.



Click on the camera button in the middle of the screen to turn on the web cam. Your browser will ask your permission for the page to access the web cam - make sure to allow access or you won't get much further.



You should then get a live feed of your web cam on the page.



> **Troubleshooting**
>
> Due to standard browser security practices, you will only be able to use the web cam if you are accessing the page using `localhost` or `127.0.0.1` on the local device, or that you are using `https` if accessing it on another device.
>
> It's beyond the scope of this workshop to get `https` setup - so we assume you are running Node-RED on the same device you are using to access the dashboard.

You can click the camera button to take a photo - the image should pause for a couple seconds before resuming the live feed.

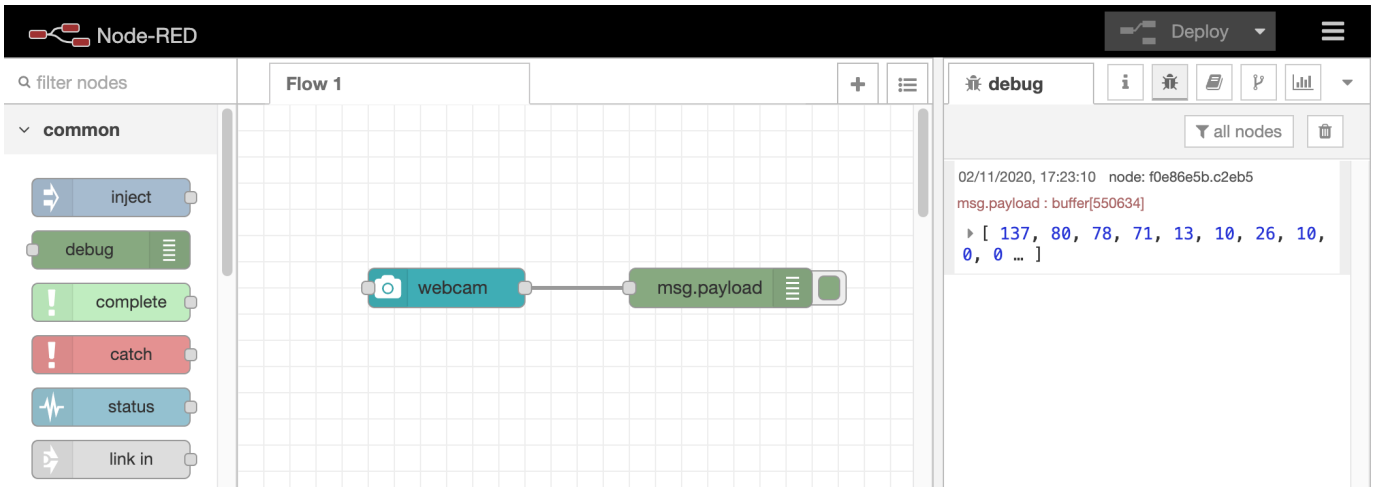
But at this point, nothing has happened with the photo you just took. The next task is to build a flow to do something with it.

3.2.4 Capturing photos

1. Back in the Node-RED editor, edit the `ui_webcam` node as follows:
    - Enable the 'start webcam automatically' option - this removes the need to click the camera button each time the dashboard is refreshed.
    - Change the Image format to `jpeg` - this is the format needed by the TensorFlow nodes later on in this workshop. Make the change now so we don't forget - but we'll remind you later on.
    - Click Done to close the dialog.
2. Add a new Debug node into the workspace. Drag a wire from the output of the Webcam node to the input of the Debug node.
3. Click Deploy to save your changes.

When you switch back to the Dashboard page the camera should already be running. Click the button to take a photo then switch back to the Node-RED editor.

Open the Debug sidebar. You should see a message has been logged showing the received payload was a Buffer. This is the raw image data in `jpeg` format.



The next task is to start writing the photo to a file.

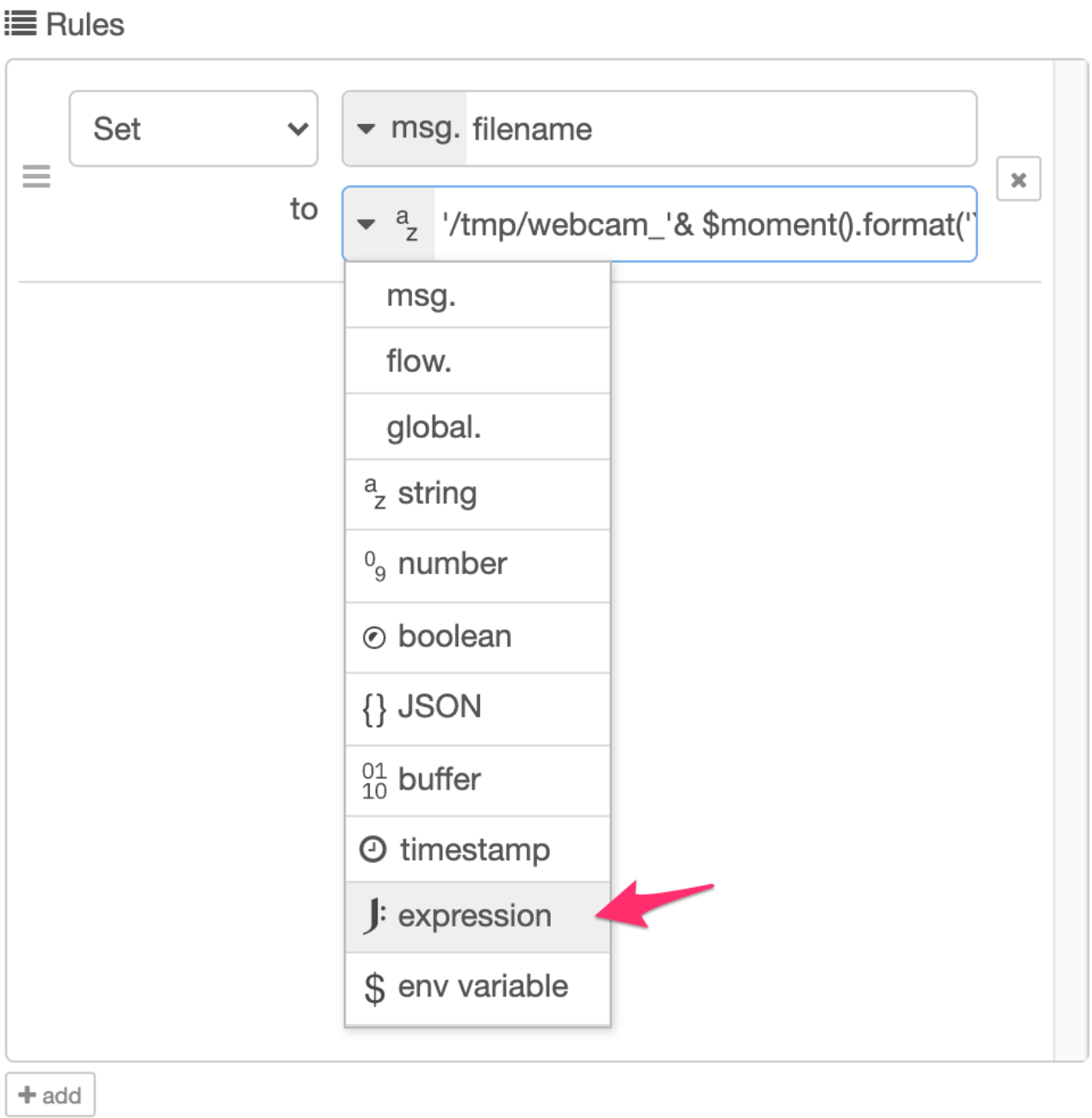
3.2.5 Saving photos to a file

1. From the output of the webcam node, wire in a Change node followed by a File node.
2. Configure the Change node as follows:
   - Change the default rule to set `msg.filename` instead of `msg.payload`.
   - Change the type of the `to` field to `expression` (click the drop-down arrow on the left-hand edge of the field to select the type)
   - Set the value of the `to` field to:

     ```
     '/tmp/webcam_'& $moment().format('YYYY-MM-DD-hhmmss') & '.jpeg'
     ```

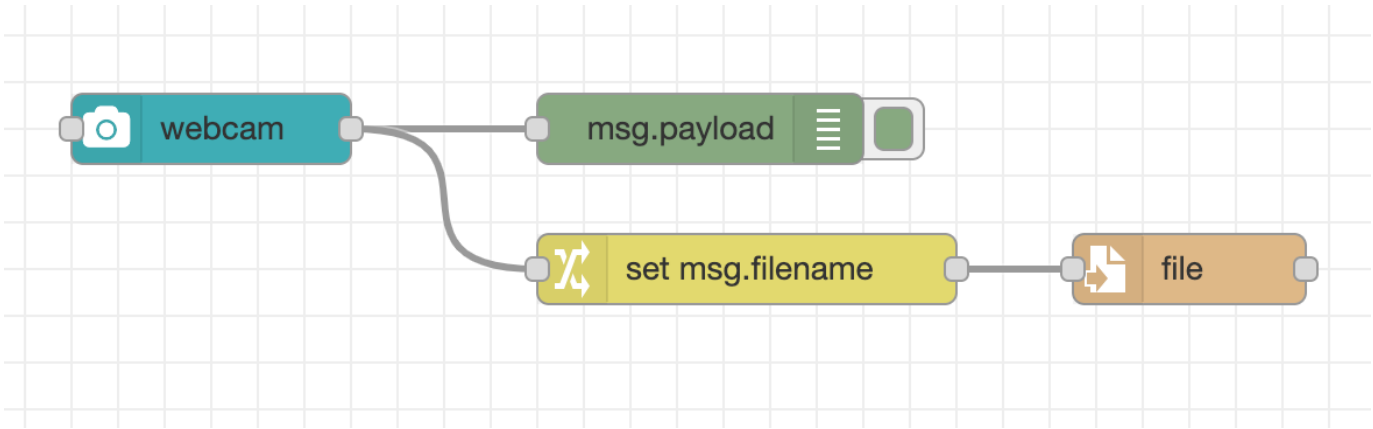
     This will generate a new filename containing the current date and time each time a message passes through the node.

     If you are running on Windows, be sure to modify the `/tmp/` part of the path to something suitable.



3. Configure the File node as follows:
   - Set the Action to 'overwrite file'
   - Untick the 'Add newline to each payload' option

4. Deploy the updates.



Now when you take photo from the dashboard it will get saved to a file.

3.2.6 Next Steps

The next task is to add some different controls to the dashboard.

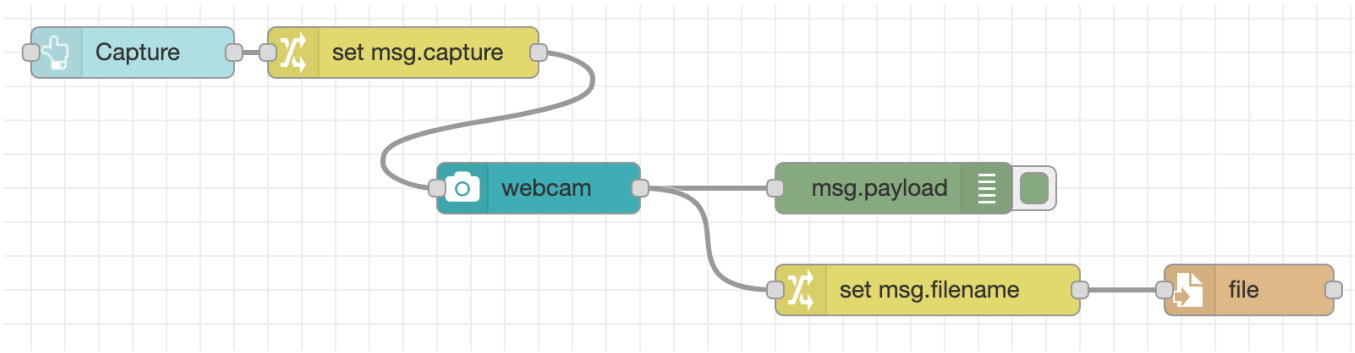
3.3 Adding controls

3.3.1 Adding a capture button

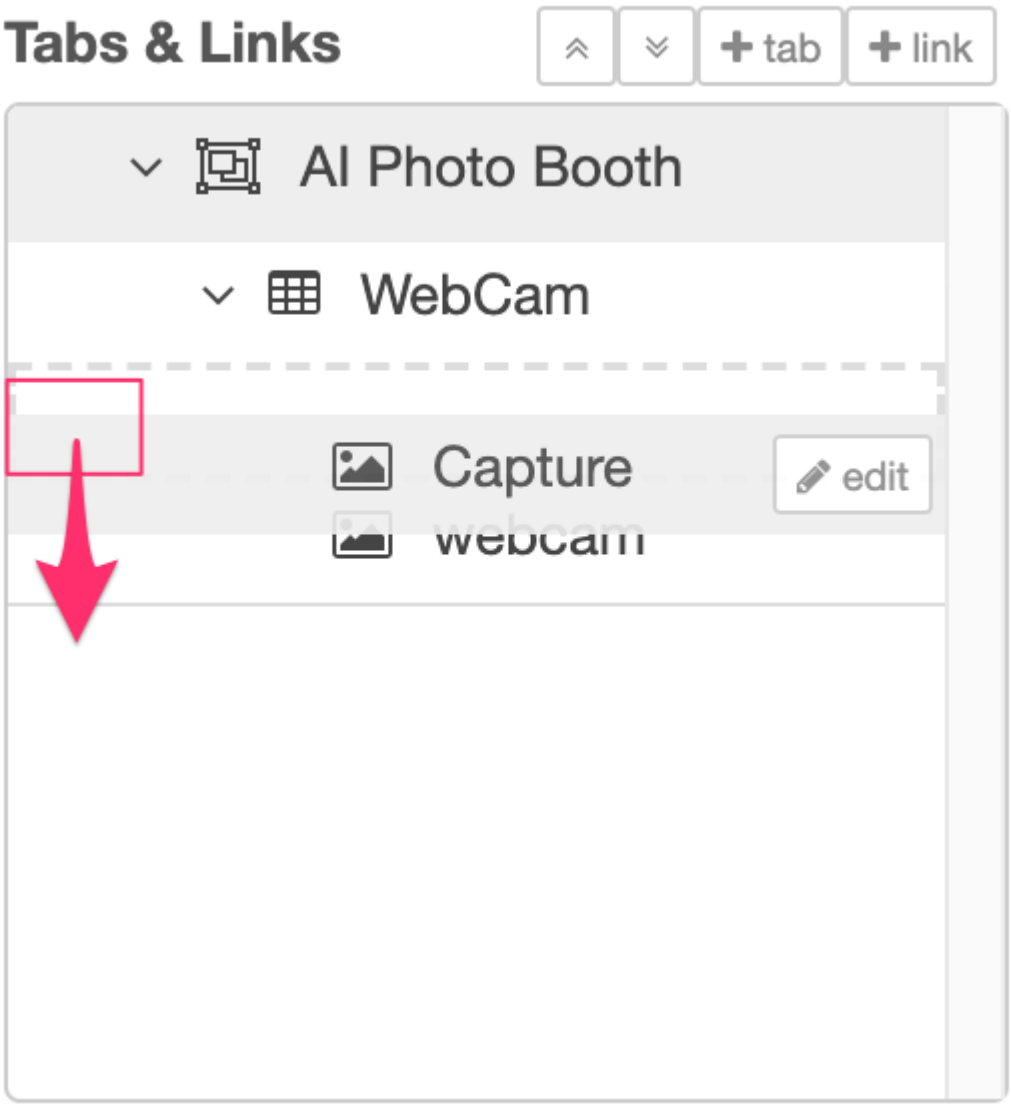
The `ui_webcam` node is fairly self-contained - providing both the live view of the camera as well as the button to trigger taking a photo.

The node also supports being triggered by passing it a message with the `msg.capture` property set - something we'll make use of next.

1. Add a `ui_button` node. Configure it as follows:
    - Set its group to the existing `WebCam` group.
    - Set its size to `9x1`
    - Set the icon to `fa-camera fa-2x`
    - Set the name to `Capture`
2. Wire its output to a new Change node, configured to set `msg.capture` to the `boolean` value `true`.
3. Wire the output of the Change node to the input of the WebCam node.
4. Edit the WebCam node and select the 'Hide capture button' option.

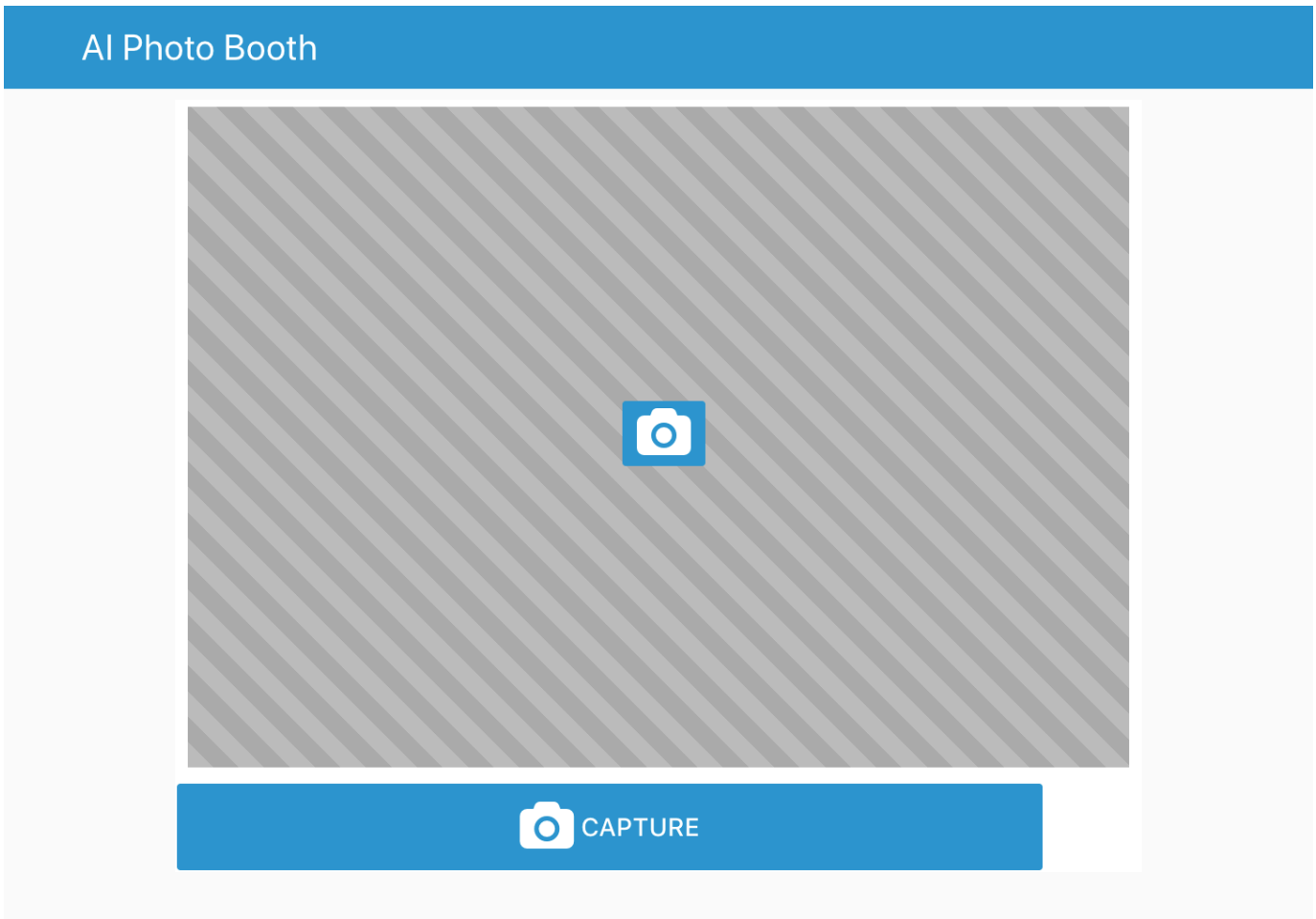


5. In the Dashboard sidebar, the `Capture` node should appear *below* the `WebCam` node. If it doesn't, you can drag the `Capture` node down into the right order.



6. Deploy the changes.

The Dashboard will now show the new button beneath the webcam widget - clicking it will trigger a photo to be taken.



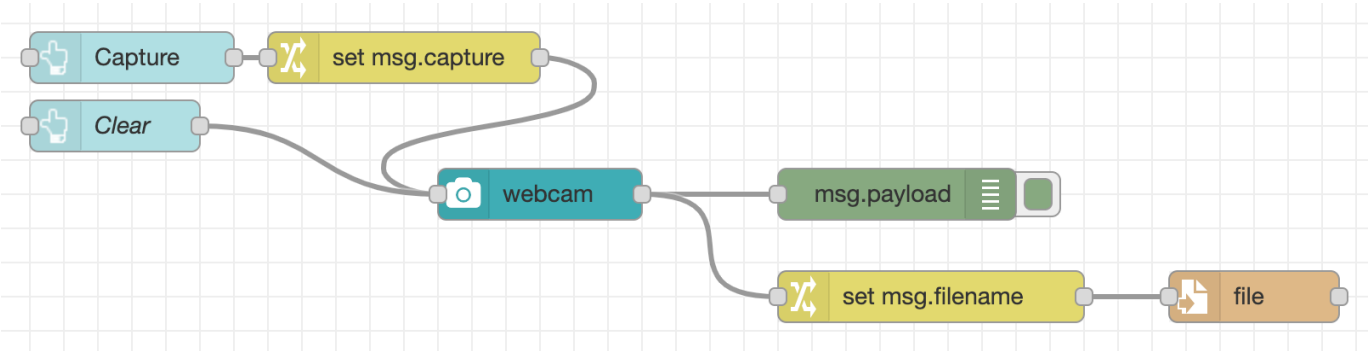
3.3.2 Adding a Clear button

By default, the webcam node shows the captured image for two seconds before returning to the live feed.

We're going to change that so it displays the captured image until the user either takes another photo or clicks another button to clear it.

1. Edit the WebCam node and untick the 'Clear image after...' option.
2. Add a new `ui_button` node.
    - Set its group to the existing `WebCam` group.
    - Set its size to `1x1`
    - Set the icon to `fa-trash fa-2x`
    - Clear the label field.
    - Set the Payload option to the `JSON` type and a value of `""`.

3. Wire its output to the input of the WebCam node.
4. As before, check the new node appears below the `WebCam` and `Capture` nodes in the Dashboard sidebar.
5. Deploy the changes.



Now when you click the camera button the captured image will be shown. Clicking the new button will clear the image and return to the live feed.

3.3.3 Next Steps

With the initial dashboard created, its now time to add some TensorFlow infused AI.

4. 3 - TensorFlow

4.1 Introducing TensorFlow

TensorFlow is open source library for Machine Learning. It provides a platform that can be used to develop and train ML models and integrate them into applications.

The project also provides TensorFlow.js - a JavaScript library that makes it easy to bring TensorFlow into browser and node applications.

Once you have a trained model, you can pass it some input data, such as an image, and it will give you back the result of the model - such as identifying objects in the image.

This is well suited to a Node-RED programming environment - where the underlying complexity of the TensorFlow model can be hidden behind a node. As an low-code developer, you just need to take care of passing in data to the model and then doing something with the resulting output.

Models

The key to creating a TensorFlow backed application is the model it uses. The model is what transforms the input to the output. TensorFlow provides the tools for developing and training custom models for specific applications. That's beyond the scope of this workshop. Thankfully, they provide a number of pre-trained models that can be used.

The one we're interested in is the Object Detection model - commonly refered to as `coco-ssd`. Given an image in jpeg format, the model is capable of detecting 80 types of object, such as `cat`, `tennis racket`, `banana` amongst many others.

4.1.1 TensorFlow nodes

The Node-RED community have created a number of different nodes that wrap TensorFlow.js. In this section we'll briefly look at three of the modules that are available.

> **Warning**
>
> Each of the modules takes a slightly different approach in how it handles its dependency on the TensorFlow libraries. Do not install all of the modules listed here at the same time as that will cause conflicts.

`node-red-contrib-tfjs-coco-ssd`

The node-red-contrib-tfjs-coco-ssd module provides a single node that wraps the Object Detection coco-ssd model and is based on TensorFlow 1.x.

You pass the node an image as either a `Buffer` object, or a string containing the filename to load. It will then return an array of the detected objects and optionally the image with all of the detected objects highlighted and labelled.

This is the module we'll be using in the workshop.

`node-red-contrib-tf-model`

The node-red-contrib-tf-model module provides a node based on TensorFlow 2.x. It does not come with any model built-in and you must provide the url of a model in JSON format. That makes it useful if you have a custom model, or want to quickly switch what model it is using. The downside is you either need to be online or host the model locally for the node to access it.

This module has a companion module that provides a custom Function node with the TensorFlow.js APIs exposed. This has to be used to prepare the image data before it gets passed to the model node.

Whilst that allows for more flexible use the nodes, they do have a steeper learning curve as you need to be familiar with the APIs.

This module also requires you to manually install the `@tensorflow/tfjs-node` dependency before installing the node.

`node-red-contrib-tensorflow`

The node-red-contrib-tensorflow module provides a set of nodes that each wraps a different pre-built model. This includes the CoCo-SSD Object Detection model, as well as human pose and hand pose detection.

4.1.2 Next Steps

Having introduced some of the TensorFlow nodes available in Node-RED, in this part of the workshop you will:

- Add TensorFlow to the Photo Booth dashboard
- Display detected objects on the dashboard
- Allow the user to select what object to display

4.2 TensorFlow in Node-RED

4.2.1 Installing TensorFlow nodes

For this workshop, we're going to use the `node-red-contrib-tfjs-coco-ssd` module that provides the `tf coco ssd` node.

This module can be installed from the Manage Palette option in the editor, or by running the following command in `~/.node-red`:

```
npm install node-red-contrib-tfjs-coco-ssd
```

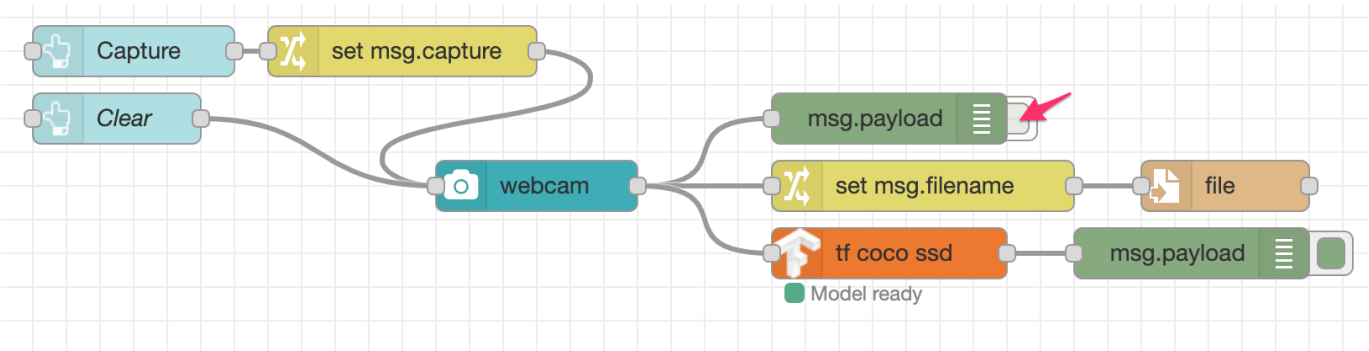
This will install the module and the TensorFlow library it depends on.

4.2.2 Connecting TensorFlow to the WebCam

In this part, we'll setup the TensorFlow node to receive images from the WebCam.

1. Add an instance of the `tf coco ssd` node from the "analysis" category of the palette into your workspace.
2. Wire the output of WebCam node to the input of the tf node.
3. Make sure the WebCam node is configured to capture `jpeg` images - we said we'd remind you about this.
4. Add a Debug node and connect it to the output of the tf node.
5. Click the Deploy button to save your changes.

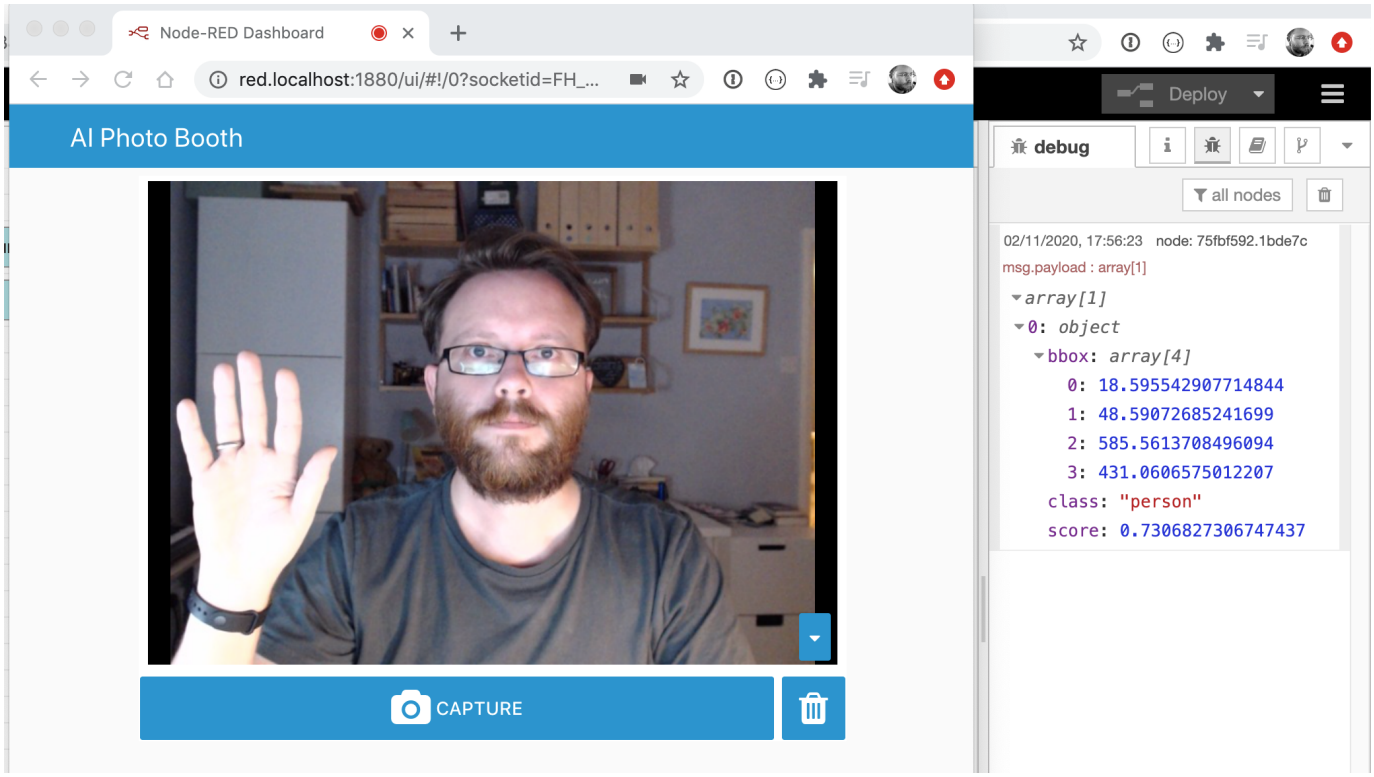
> **Muting Debug nodes**
>
> In this screenshot you can see I have muted the Debug node attached to the webcam node by clicking its button in the workspace. This can be useful to turn off different bits of Debug without unwiring or removing the nodes entirely.



On the dashboard, make sure your webcam can see you and click the capture button. Switch back to the Node-RED editor and open the Debug sidebar panel. You should see the message sent by the tf node. Its payload consists of a list of the objects it has detected.

Each entry in the list has:

- `class` - the type of object
- `score` - the confidence level of the detection, from `0` to `1`.
- `bbox` - an array giving the corners of the bounding box surrounding the detected object



`msg.payload` format

Each of the TensorFlow nodes uses a slightly different object format. For example, the `node-red-contrib-tf-*` nodes set the `className` property rather than `class` as we have here. If you experiement with the other nodes make sure you read their documentation and use the Debug node to understand their message format.

4.2.3 Next Steps

With TensorFlow integrated into the dashboard, the next task is to display the detected objects on the dashboard.

4.3 Displaying the detected objects

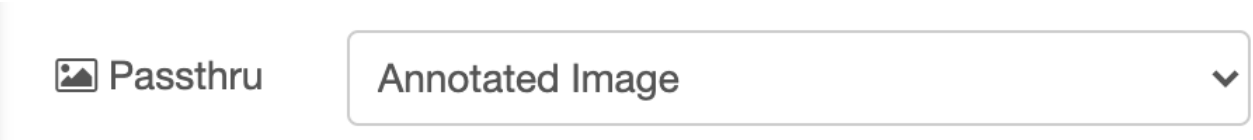
In this part we're going to display the detected objects on the dashboard in two different ways.

First we will display an annotated version of the captured image with all of the objects highlighted. We will then add a table to the dashboard that lists them out.

4.3.1 Displaying an annotated image

The `tf coco ssd` node has an option to output an annotated version of the image with all of the detected objects highlighted. The image is set on the `msg.image` message property.

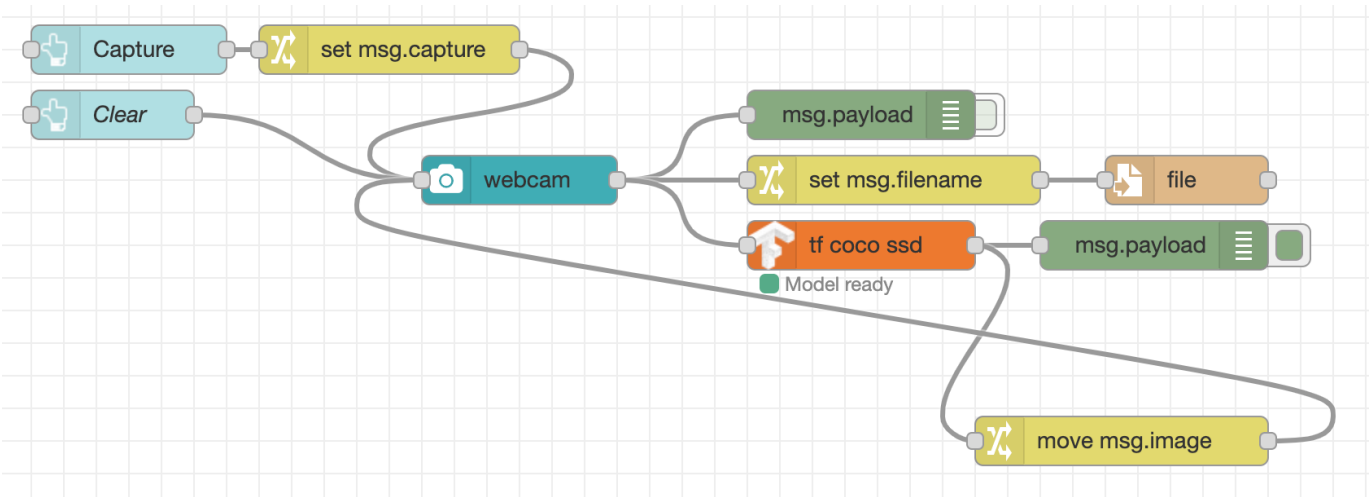
1. Edit the `tf` node and configure the "Passthru" field to `Annotated Image`



2. Add a Change node, wired to the output of the `tf` node and configure it to move `msg.image` to `msg.payload`.

3. Wire the Change node to the input of the WebCam node.

4. Click the Deploy button to save your changes.

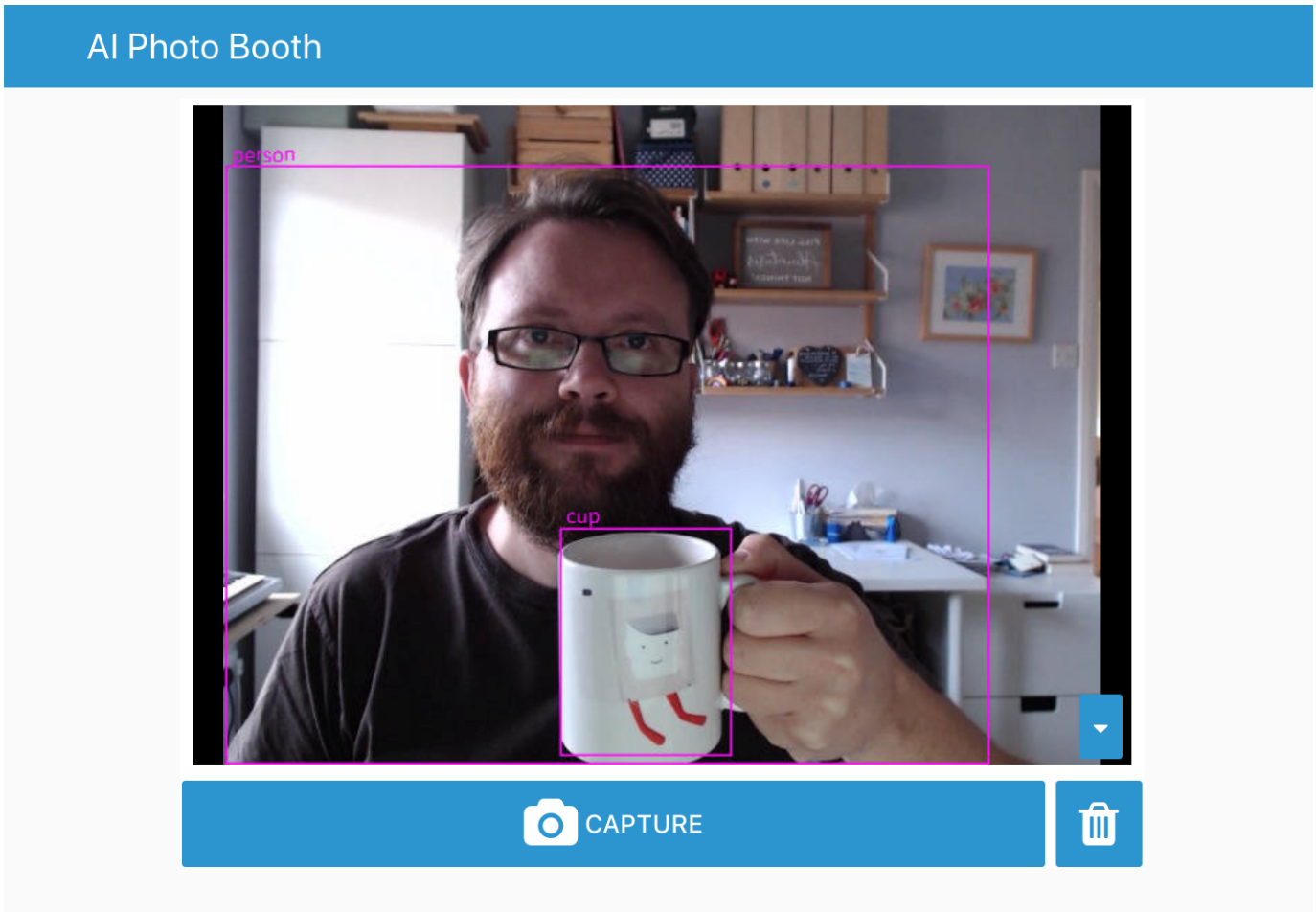


Laying out flows

With this latest addition, you can see we now have wires crossing each other and looping back on themselves. As flows evolve, their wiring can become quite complex. It is always worth spending some time trying to find a layout that remains 'readable'.

There is a Flow Developer guide in the Node-RED documenation that provides a number of tips on how to layout flows.

Now when you take an image on the dashboard, you should see the annotated version of the image.



4.3.2 Adding a table of objects

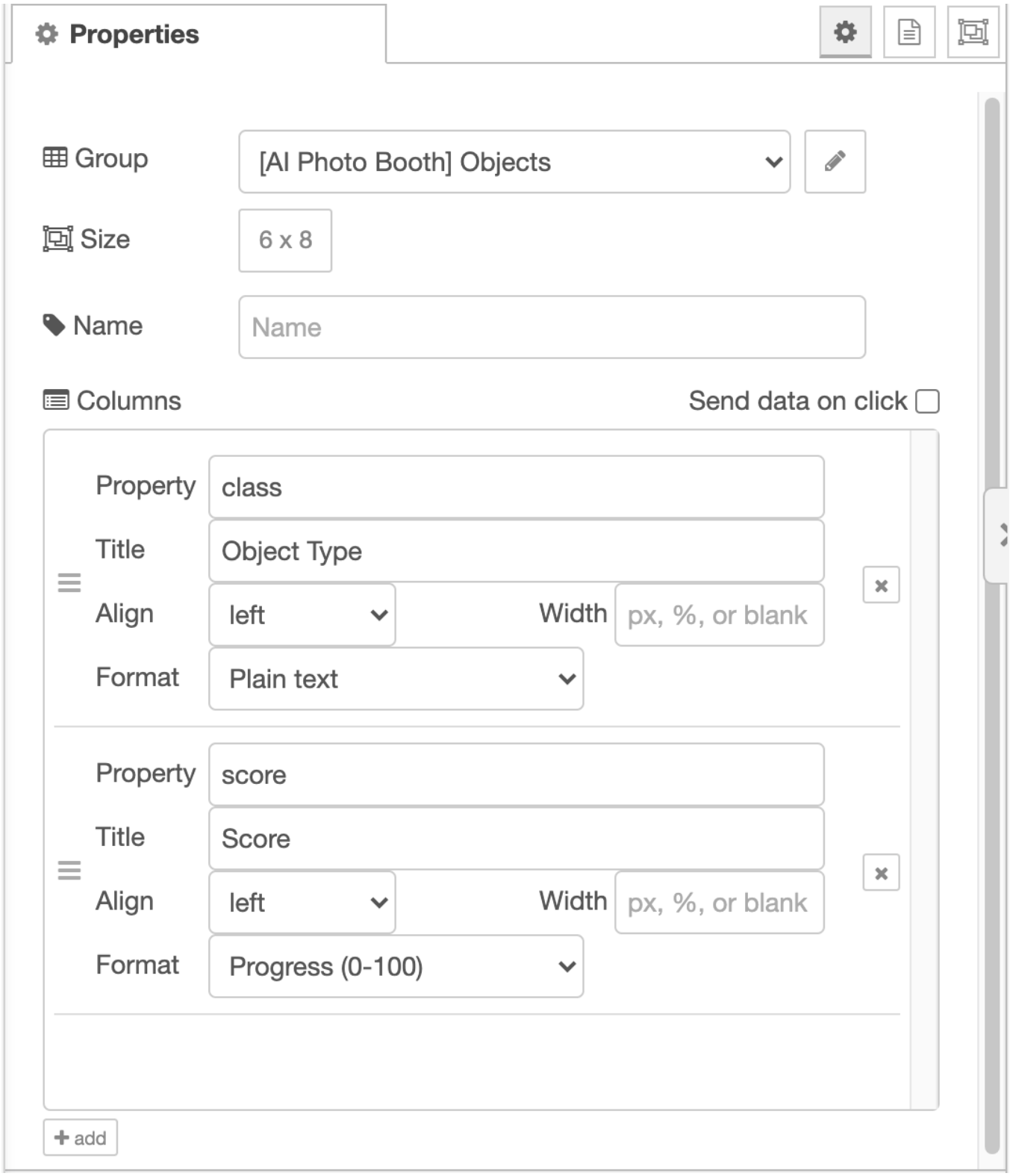
Install the module `node-red-node-ui-table` using the Manage Palette option in the editor, or run the following command in `~/.node-red`:

```
npm install node-red-node-ui-table
```

This adds the `ui_table` node to the palette which can be used to display tabular data.

1. In the Dashboard sidebar of the Node-RED editor, hover over the `AI Photo Booth` tab and click the `+ group` button.
2. Edit the new group and set its properties:
    - Set the name to 'Objects'
    - Set the width to `6` by clicking the button and dragging the box out to 6 units wide.
    - Untick the 'Display group name' option.

3. Add a new `ui_table` node from the "dashboard" section of the palette into your workspace. Edit its properties as follows:
   - Add it to the 'Objects' group
   - Set its size to `6x8`
   - Add two columns by clicking the `+ add` button at the bottom. Configure them as:
     - Property: `class`, Title: `Object Type`
     - Property: `score`, Title: `Score`, Format: `Progress (0-100)`

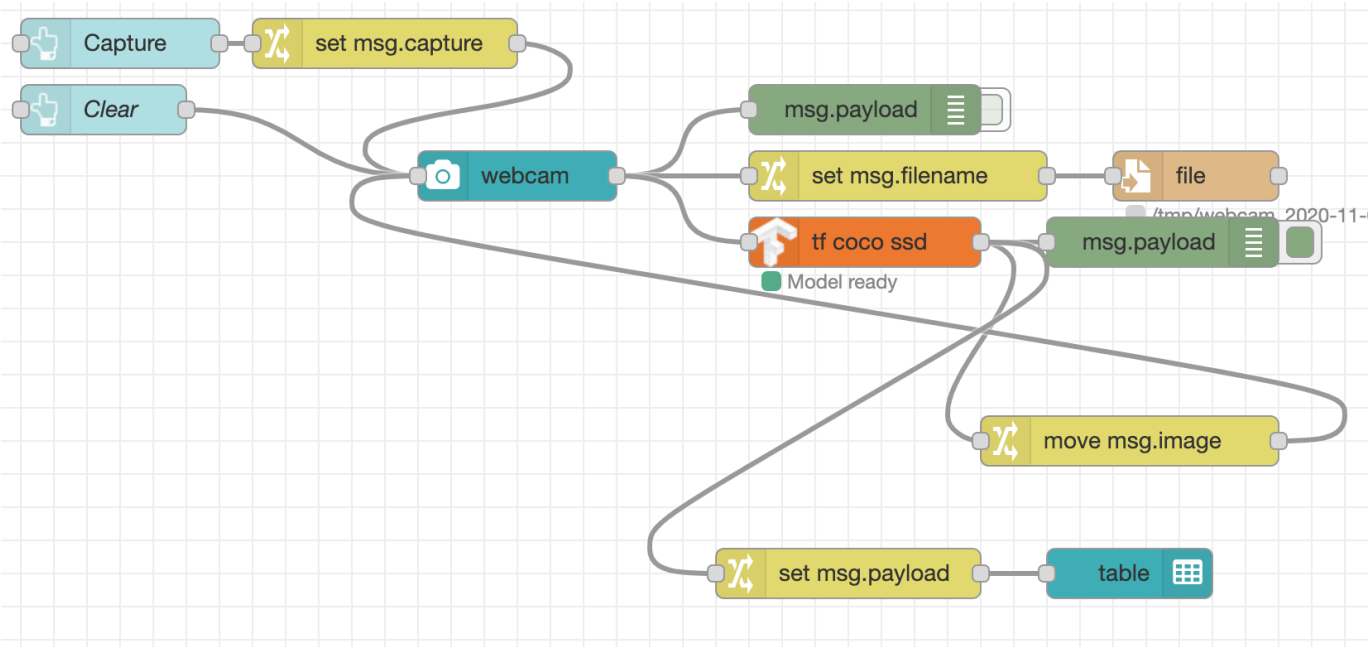


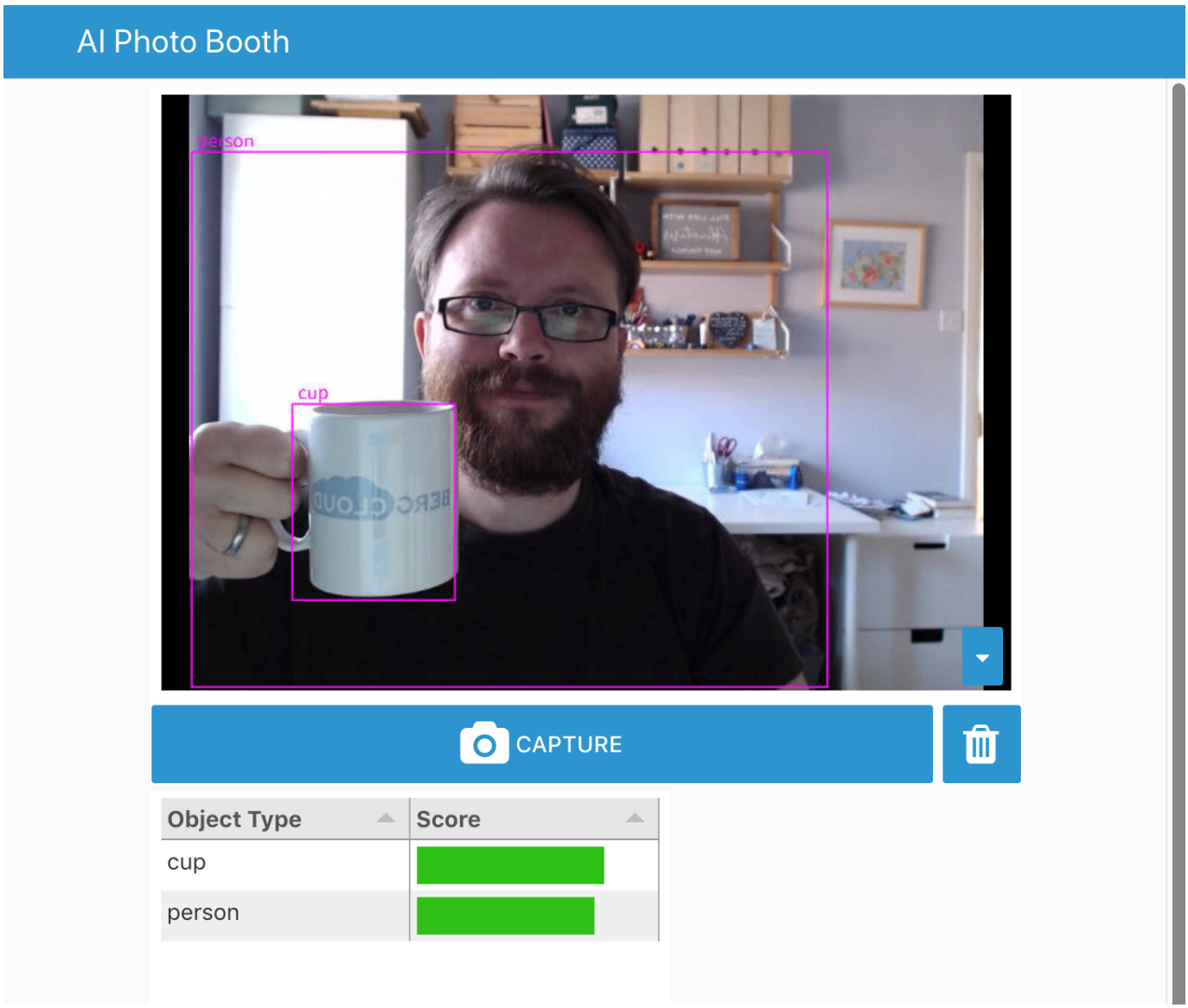
4. Add a Change node to the workspace. Configure it to set `msg.payload` to the expression `$append([],payload.{"class":class,"score":score*100,"bbox":bbox})`

> **Note**
>
> Make sure you select the `expression` type for the `to` field of the Change node. This uses the JSONata expression language.

5. Create the following wires between nodes:
   - wire the output of the `tf` node to the Change node.
   - wire the output of the Change node to the Table node
6. Click the Deploy button to save your changes.

Now when you capture an image on the dashboard, the table should list the detected objects.





Side Quest - Star Ratings

The JSONata expression used in the Change node mapped the `score` property of each detected object from its original `0-1` range to the `0-100` range expected by the `ui_table` node's "Progress" column type.

The table supports a number of other formats of displaying numeric values. For example, it can map a number in the 0-100 range to a traffic light colour. It can also display a value in the range 0-5 as a number of stars.

Edit the table node to display the score using the star format. See if you can modify the expression in the Change node to map the original score to the required `0-5` range.

Side Quest - Clear the table

With the current dashboard, when an image is captured it gets displayed in place of the live web cam view until the clear button is clicked.

However clicking the button does not clear the table we've just added.

Using what you've learnt so far, build a flow between the Clear button and the table node that will clear the table when the button is clicked.

Hint: think about what payload must be passed to the table in order to clear it.

4.3.3 Next Steps

With the list of detected objects on the dashboard, the next task is to let the user select which object to display.

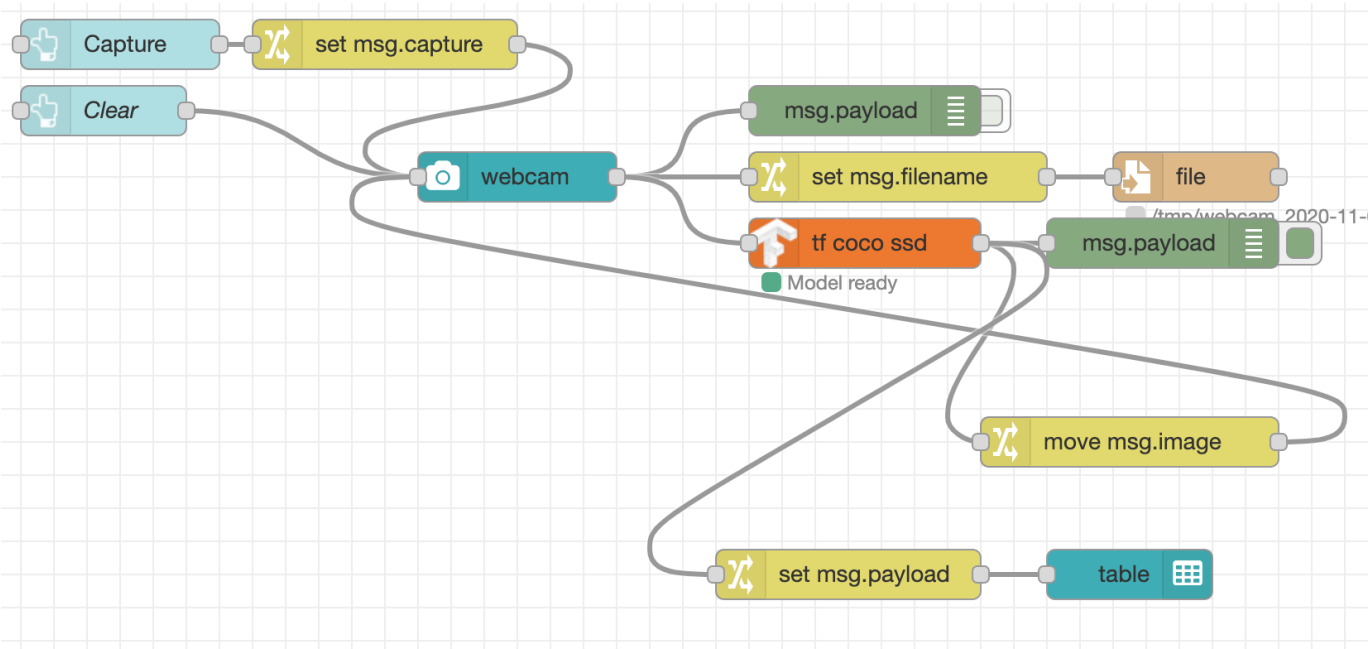
4.4 Selecting objects to display

So far we have a flow that lets you take photo and then display all of the detected objects in it.

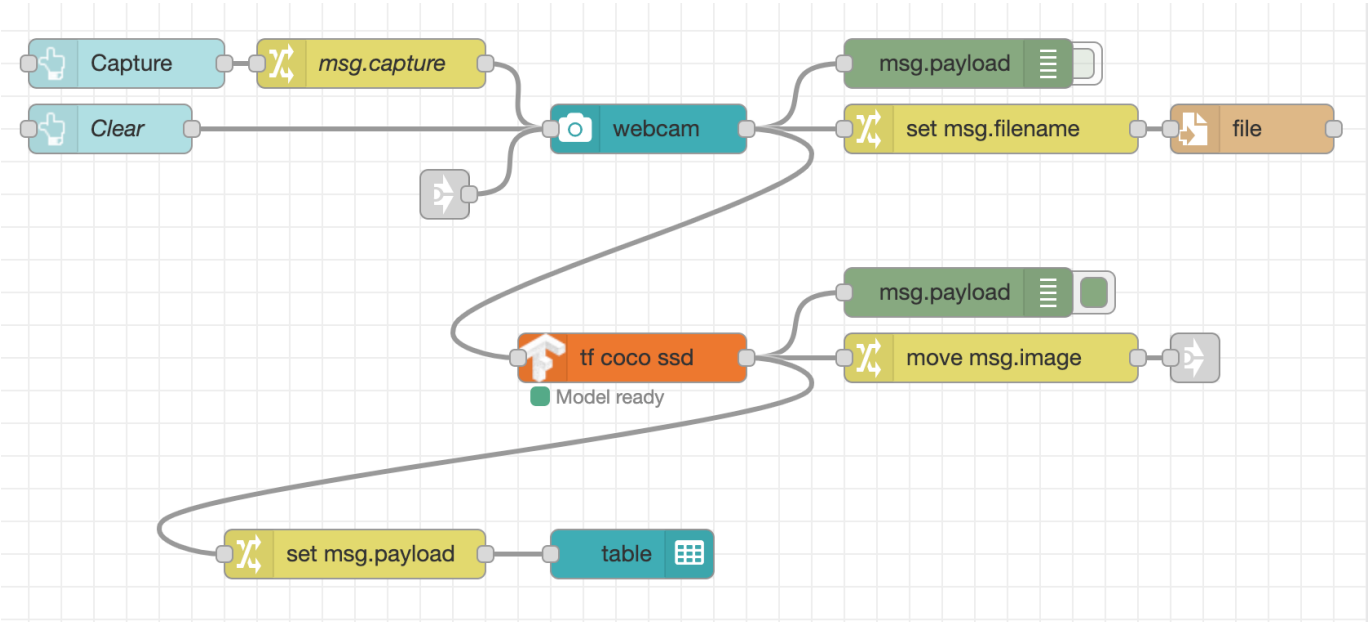
In this part of the workshop we're going to change the flow to let the user select which object to highlight in the displayed image.

Before we do that, we need to rearrange the nodes to make some space for what we'll be adding in this section.

Here is what we have so far:



With a bit of moving around we can get to the following:



The main addition is the pair of Link nodes - a Link In node wired to the input of the webcam node, and a Link Out node wired to the output of the `move msg.image` Change node. Link nodes allow you to create virtual wires that only get shown when you select a link node at either end. This can help reduce some of the visual complexity of a flow.

Wiring Link nodes

You can wire Link nodes together in two ways. One way is to double click on a Link node to open its edit dialog. Then select which nodes it should be connected to from the list shown. The nodes are listed by their name or id. It is always worth naming your link nodes to help you identify them in the list. This method can be used to join link nodes that are on different tabs in the editor.

The other way of connecting a pair of link nodes is to first select a node to reveal its 'virual port' and then drag a wire from that port to another link node just as you would wire regular nodes.

4.4.1 Making the table selectable

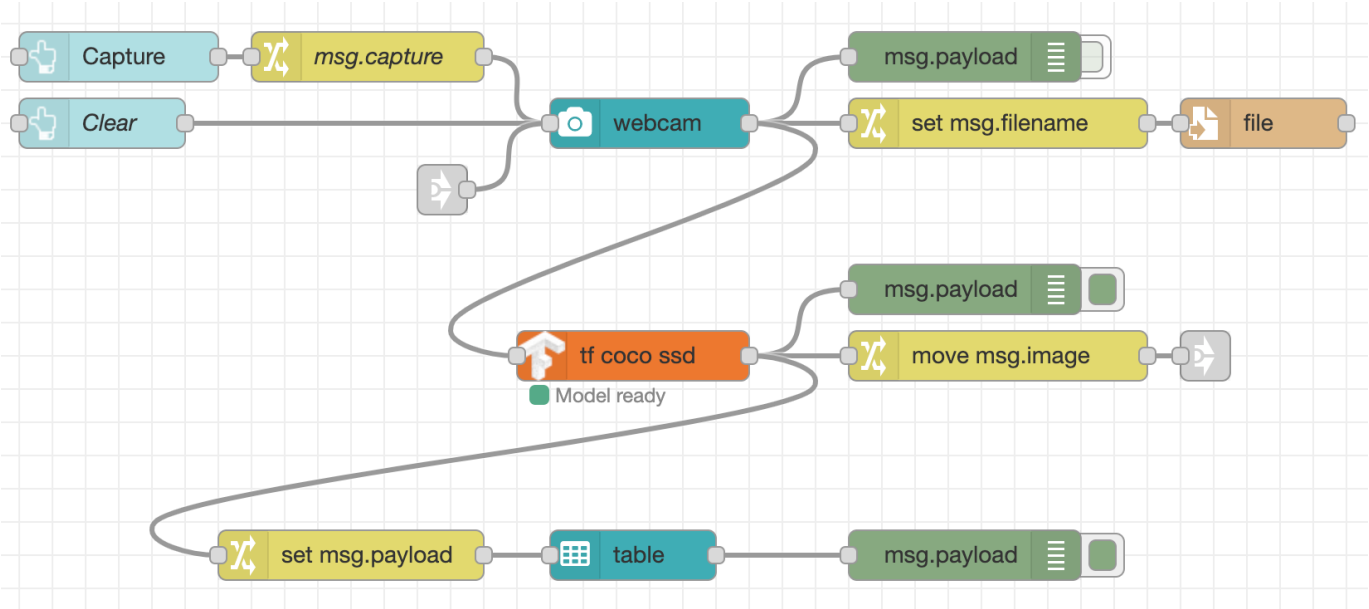
We need a way for the user to select the object to display and then regenerate the annotated image with just the selected object.

The `ui_table` we added in the last step can be used to select the object.

1. Edit the table node and enable the 'Send data on click' option.

   The table node should now have an output port.

2. Add a Debug node and wire it to the table node output.



Now when you select an object in the dashboard table you will see a message arrive in the Debug sidebar with the object's information. It will look something like:

```
{
    "class": "person",
    "score": 64.49049711227417,
    "bbox": [
        5.880241394042969,
        49.79872226715088,
        517.5165557861328,
        429.3963575363159
    ],
    "id": 0
}
```

4.4.2 Annotating the image

The TensorFlow node we are using generates the image with all detected objects annotated. It doesn't provide a way to retrospectively generate the image with only a particular object highlighted.

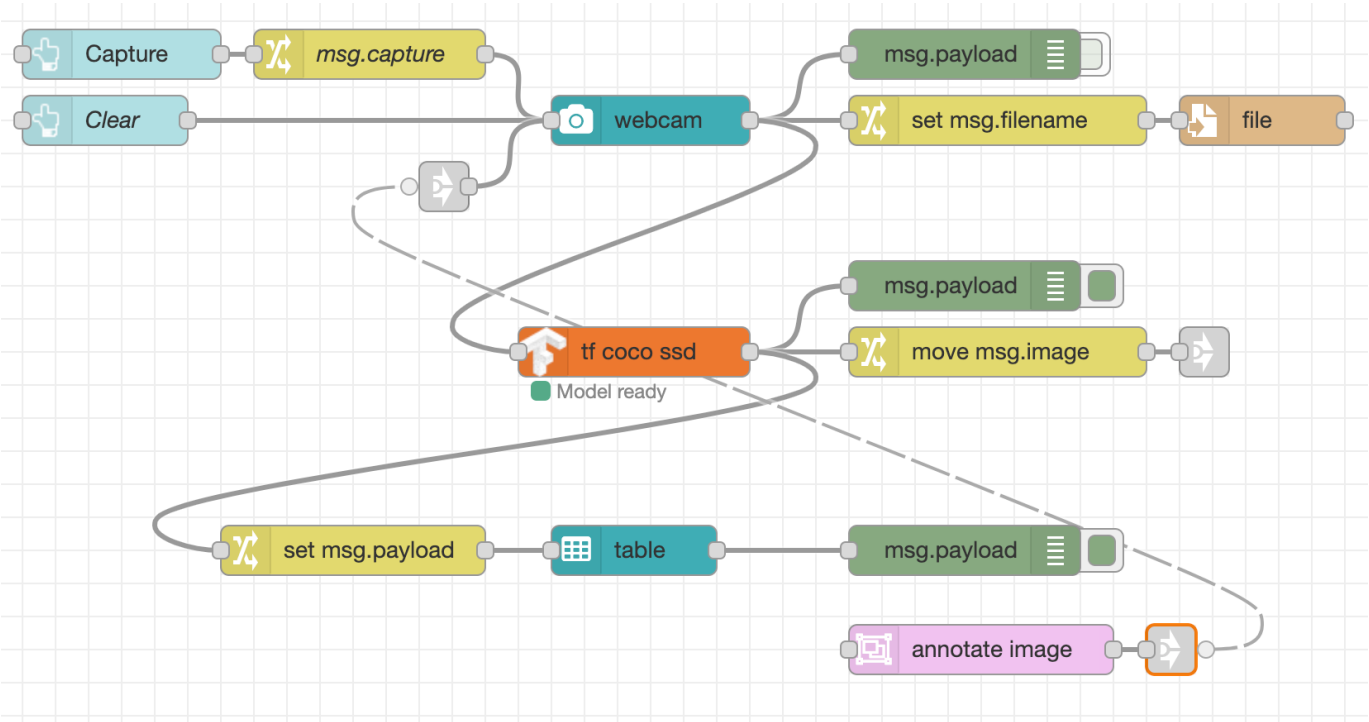
To do that we can use another module: node-red-node-annotate-image.

This module can be installed from the Manage Palette option in the editor, or by running the following command in `~/.node-red`:

```
npm install node-red-node-annotate-image
```

Once installed, you will find a node called `annotate-image` in the Utility category of the palette.

1. Add a new `annotate-image` node into your workspace.
2. If you've added the Link nodes like we did at the start of this section:
   a. Wire the output of the `annotate-image` node to the add a new Link Out node
   b. Link the new Link Out node to the Link In node connected to the webcam node.
3. If you haven't added the Link nodes, wire the output of the `annotate-image` node directly to the input of the `ui_webcam` node.



> **Note**
>
> In this screenshot, the virtual wire is show because the Link Out node is selected.

The next task is to take the output of the `ui_table` node when an object is clicked and get it into the right format required by the `annotate-image` node.

The annotate node expects two properties to be set on the message it is sent: - `msg.payload` should be a Buffer containing the image in JPEG format - `msg.annotations` should be an array of the annotations it should apply to the image.

Storing the image

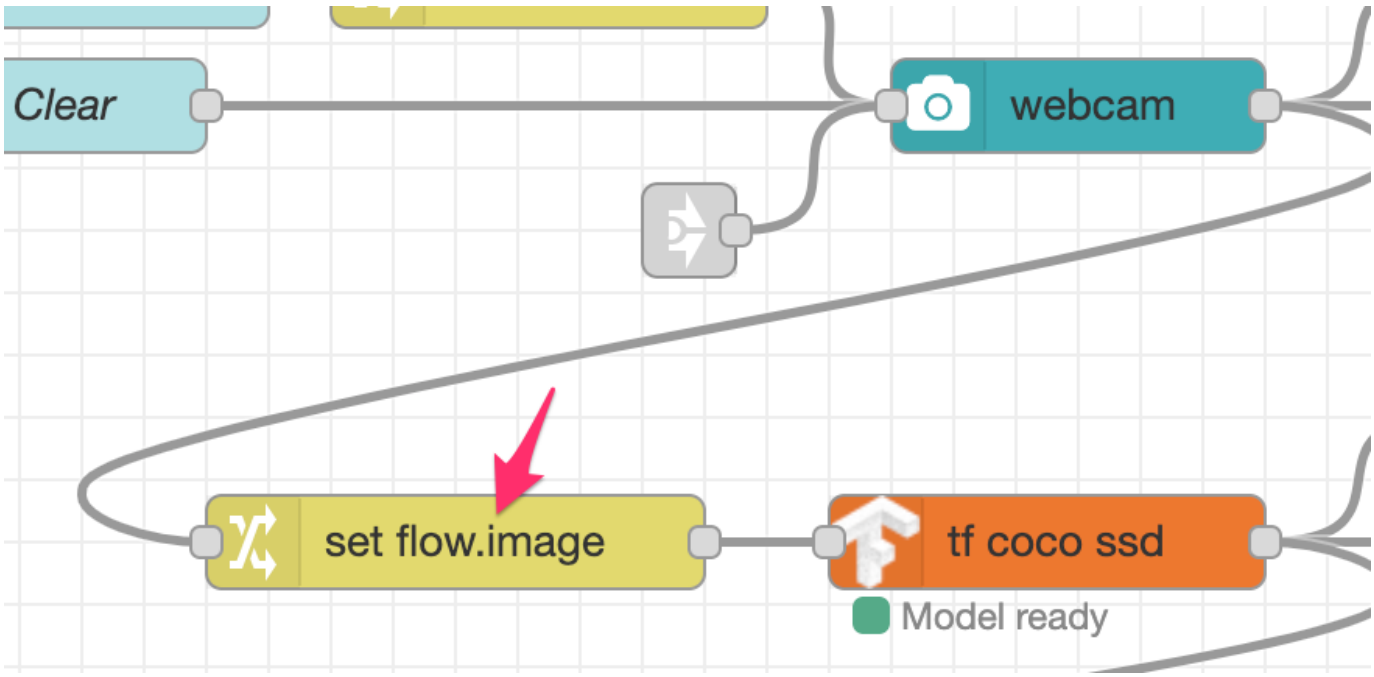
In order to annotate the image, we need a copy of the image to work on. The flow we're working on is triggered when the user clicks on a row in the `ui_table`. This happens as a new event - it isn't tied to original event that captured the image.

This means we need some way to get ahold of the original image in this new flow. We can use Context to do that.

> **Introducing "Context"**
>
> "Context" is a way to store information in Node-RED that is not tied to individual messages passing through the flow. For example, it can be used to store global state that needs to be shared amongst flows.
>
> For more information about context, read the documentation.

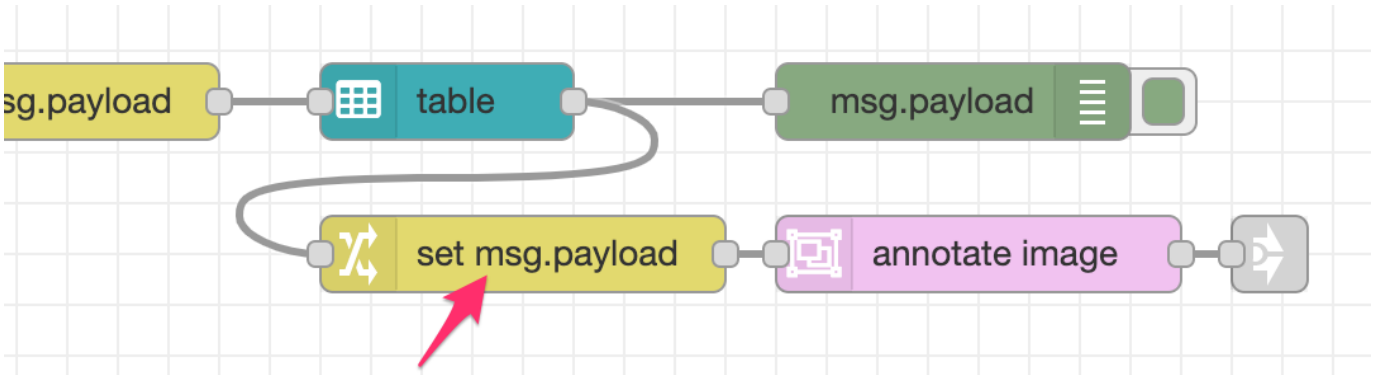
The first task is to store the image in context whenever a photo is taken.

1. Add a Change node in between the `ui_webcam` and `tf coco ssd` nodes.
2. Configure the node to set `flow.image` to the value of `msg.payload`.



The next task is to retrieve that image and add it to the messages coming from the `ui_table` node.

1. Add a Change node and wire in between the output of the `ui_table` node and the `annotate-image` node.
2. Configure the node to set `msg.payload` to the value of `flow.image` - this is essentially the reverse of the operation done in the previous task.



Creating the annotations

As shown earlier, the payload coming from the `ui_table` node looks like this:

```
{
    "class": "person",
    "score": 64.49049711227417,
    "bbox": [
```

```
        5.880241394042969,
        49.79872226715088,
        517.5165557861328,
        429.3963575363159
    ],
    "id": 0
}
```

In order draw that as an annotation on the image, the annotation node expects `msg.annotations` to be a array containing an object like this:

```
[
    {
        "label": "person",
        "bbox": [
            5.880241394042969,
            49.79872226715088,
            517.5165557861328,
            429.3963575363159
        ]
    }
]
```

You can see they are very similar, but some work is needed to map between them.

1. Edit the Change node added in the previous step (between the `ui_table` and `annotate-image` nodes)
2. Add a new rule, and then **drag it to the top of the list of rules**. This is important because the new rule will use the information in `msg.payload` so needs to be applied *before* the rule that overwrites `msg.payload` with the image data.
3. Configure the new rule to set `msg.annotations` to the *expression* type and set the expression to:

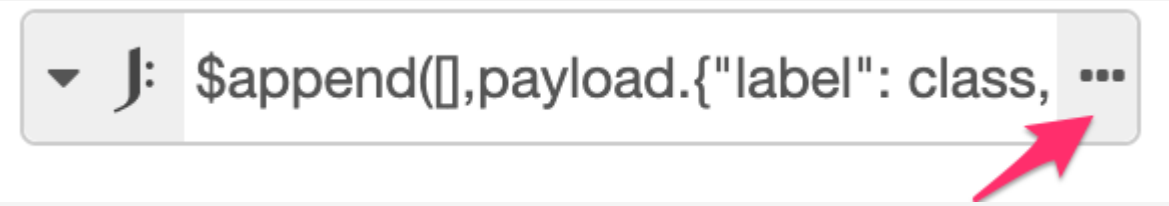
```
$append([],payload.{"label": class, "bbox": bbox})
```

This will create an array of annotations with just the `label` and `bbox` properties set from the original object.

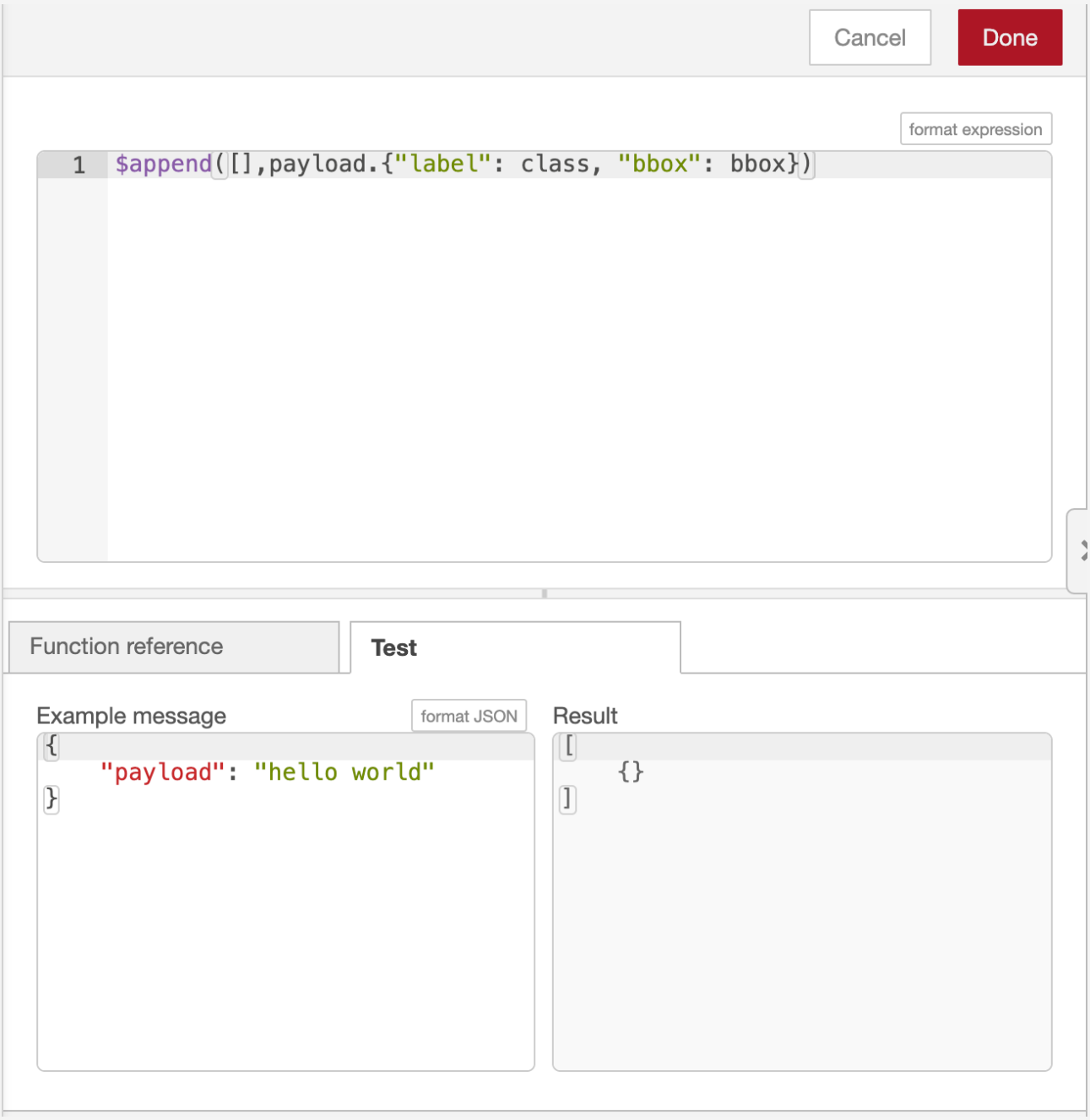
Testing JSONata Expressions

Node-RED provides an editor for JSONata expressions with the ability to test the expression whilst working on it.

1. Click on the expand button next to the Expression:



2. This opens the JSONata editor. You can edit the expression in the top pane. In the lower pane is a Function Reference and a Test tab. Click on the Test tab. This shows an example message on the left and the result of the expression on the right.



3. Copy the example output of the annotate node from above and replace the `"hello world"` example payload. You should see the result update with shown here.

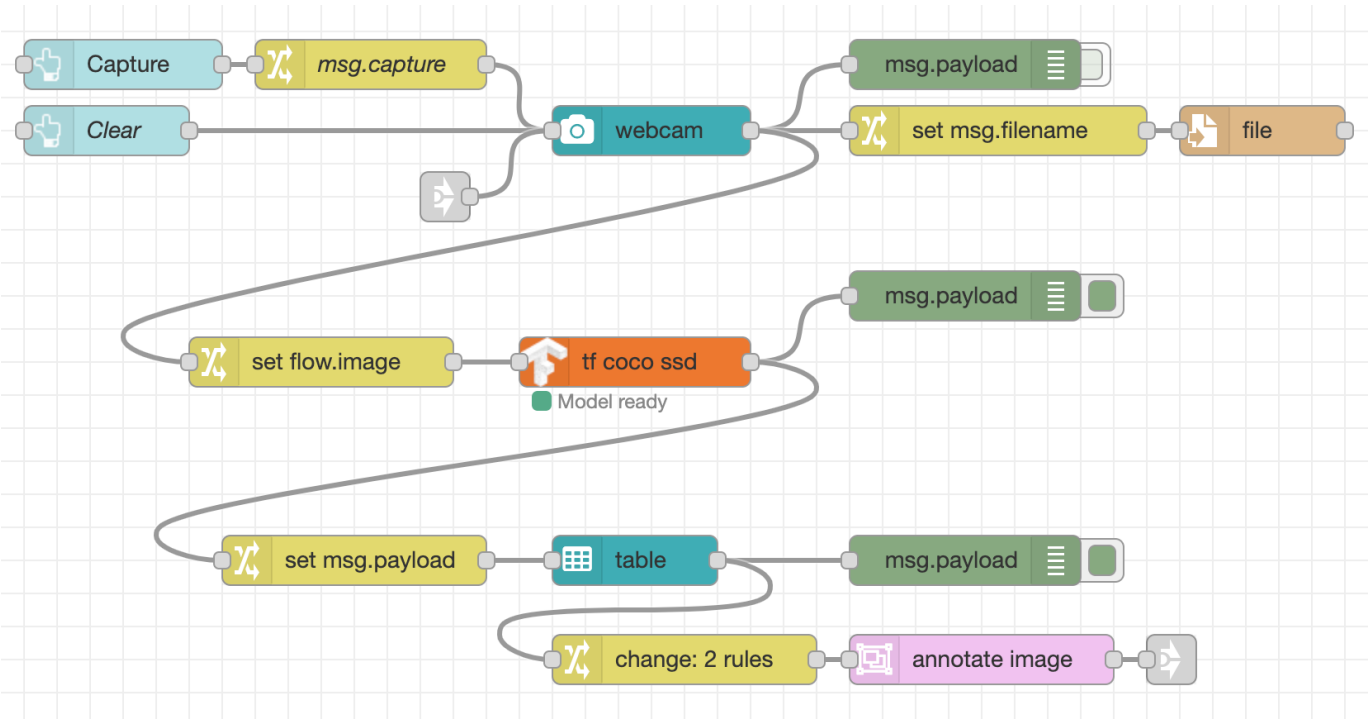
You should now have a flow that starts with the `ui_table` node, passes through a `Change` node to format the message, then to an `annotate image` node and finally to the `ui_webcam` node, possibly via a `Link` node.

Before deploying these changes there are couple more changes to the flow needed. We aren't going to use the image produced by the `tf coco ssd` node anymore.

1. Delete the `move msg.image` Change node that is connected to the output of the `tf coco ssd` node. If you added the Link Out node, delete that as well.
2. Edit the `tf coco ssd` node to set the Passthru option to `nothing`
3. Click the Deploy button to save your changes.

Now when you capture an image on the webcam, you'll see the captured image on the dashboard without any annotations. The table below the image will list the detected objects. Clicking on an object will update the image to highlight the selected image.

This is what the final flow should look like:



4.4.3 Next Steps

With the application complete, the next task is to wrap it in a container.

5. Summary

That's all folks. If you've reached this point of the workshop, you've covered a lot of ground. Whether you were already familiar with Node-RED, TensorFlow, or if it was all new to you, we hope you've found it interesting.

The goal for this workshop was to show how you can quickly start building applications using low-coding tools. While the application we've built today is a bit of fun, it shows how much you can achieve with very little code being written.

5.1 Next Steps

Where you go next is very much up to you.

If you're new to Node-RED, you can explore the Flow Library to see what other nodes are available and what example flows the community have shared

If you're new to TensorFlow, you can explore the other TensorFlow nodes we mentioned and some of the other prebuilt models that are available. You could explore building your own model and getting that integrated into your Node-RED application.

Check out the resources section for more useful links.

6. Resources

Here are some useful links and information to help you to continue to explore Node-RED and TensorFlow.js.

6.1 Node-RED Resources

- You can find a complete copy of the Node-RED project this workshop creates here.

  This includes the flow file you can import if you want to skip to the end.
- Alternatively, you can access the final flow from this repository here.
- For more information about using Node-RED, start with the Node-RED Essentials playlist on YouTube.
- The Working with Messages section of the documentation provides a good introduction to understanding the structure of a message.
- The Flow Developer Guide provides lots of useful information on best practices in creating flows. It covers how to layout your flows, how to structure your messages and also how to properly document your flows (something this workshop hasn't really touched on).
- The Node-RED community forum is a great place to get help, as is the project's Slack.

6.2 TensorFlow.js Resources

- You can find out more about TensorFlow.js at its site here
- We've used the Object Detection model in this workshop. For more information about other pre-trained models that are available, check out this page.
- To use a different model, you'll need to change over to the node-red-contrib-tf-model set of nodes. Make sure you read its documentation before installing it as you have to manually install the TensorFlow libraries first.

  You should not try installing it at the same time as the `node-red-contrib-tfjs-coco-ssd` module we've used in this workshop.