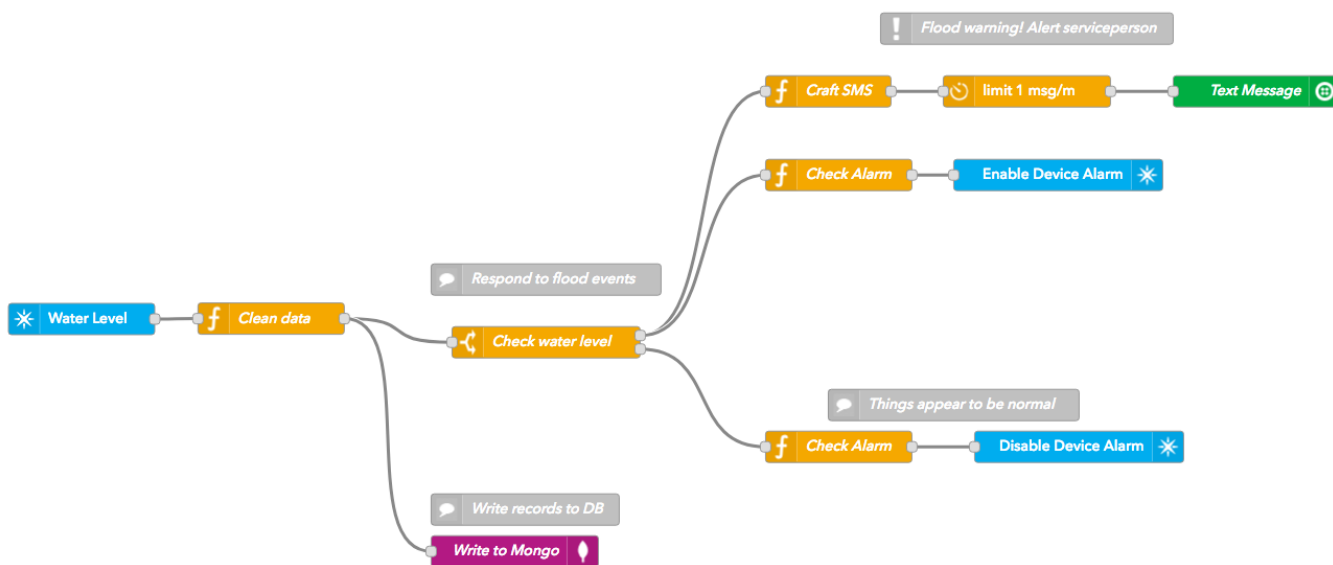


# IOT RULES ENGINE

The Particle Rules Engine is an **IF/THEN** workflow builder specifically designed for IoT data. It lets anyone on your team easily create conditional business logic using a visual interface.



The Rules Engine can trigger activities in the cloud based on something that is happening in a device's environment. That is:

**IF** certain conditions are met in the *physical world*, **THEN** take an action in the *digital world*.

Some of the specific ways you could use the Rules Engine in this way include:

- Notify a device owner when sensor readings leave a healthy range
- Open a new case in your CRM of choice when a hardware component stops functioning
- Alert your on-call engineers when the error rate of your fleet exceeds a certain threshold

The opposite goal can also be achieved:

**IF** certain conditions are met in the *digital world*, **THEN** take an action in the *physical world*.

Some of the specific ways you could use the Rules Engine in this way include:

- Send an over-the-air firmware update to a device when it has been added to a [device group](#)
- Activate certain features of your product when customer payment information is collected
- Disable access to using a broken device when a serviceperson is en-route to fix it

The Rules Engine is a Particle-integrated, professional-grade deployment of a popular tool called [Node-RED](#). Node-RED not only provides a robust flow-based programming framework, but also access to a variety of open-source contributions to the project made by its thriving community of IoT builders.

## Key Terms

There are some key terms that will help you get started with the Rules Engine:

### Nodes



The building blocks of the Rules Engine are called **nodes**. Each node represents a distinct piece of logic and has a well-defined purpose: it gets some data, does something with that data, and then passes that data on.

There are many different types of nodes that carry out specific functions. Each node can be configured with pre-defined properties to work to your specifications.

There are 3 main types of nodes:

- **Input nodes** that inject data into a flow. These nodes are placed at the beginning of a flow and trigger flow execution.
- **Output nodes** that send data out of a flow. These nodes are placed at the end of a flow.
- **Processing nodes** that receive data from the previous node, execute some logic, then send the result onto the next node. These nodes are placed in the middle of a flow.

## Flows



Multiple nodes are connected to one another, creating networks called **flows**. The flow is responsible for moving data between the nodes.

If nodes are the building blocks of the Rules Engine, flows are the end-products. They represent the combination of many pieces of logic into an end-to-end workflow.

## Palette

The "menu" of available nodes available to you when assembling flows is known as a **palette**. From the palette, you can drag nodes onto the workspace and incorporate them into a flow.

The Rules Engine comes standard with a default set of nodes that caters to the most common use cases. However, there are many available nodes that do not come with the default palette that may be useful to you. Learn more in the section on [adding nodes to your palette](#).

## Message

The Rules Engine executes a flow by passing **messages** from node to node. These messages are simple objects that always have an ID, and often also have a set of associated properties.

In the Rules Engine, a message object is normally referred to as `msg`. Associated properties are most commonly stored in `msg.payload`.

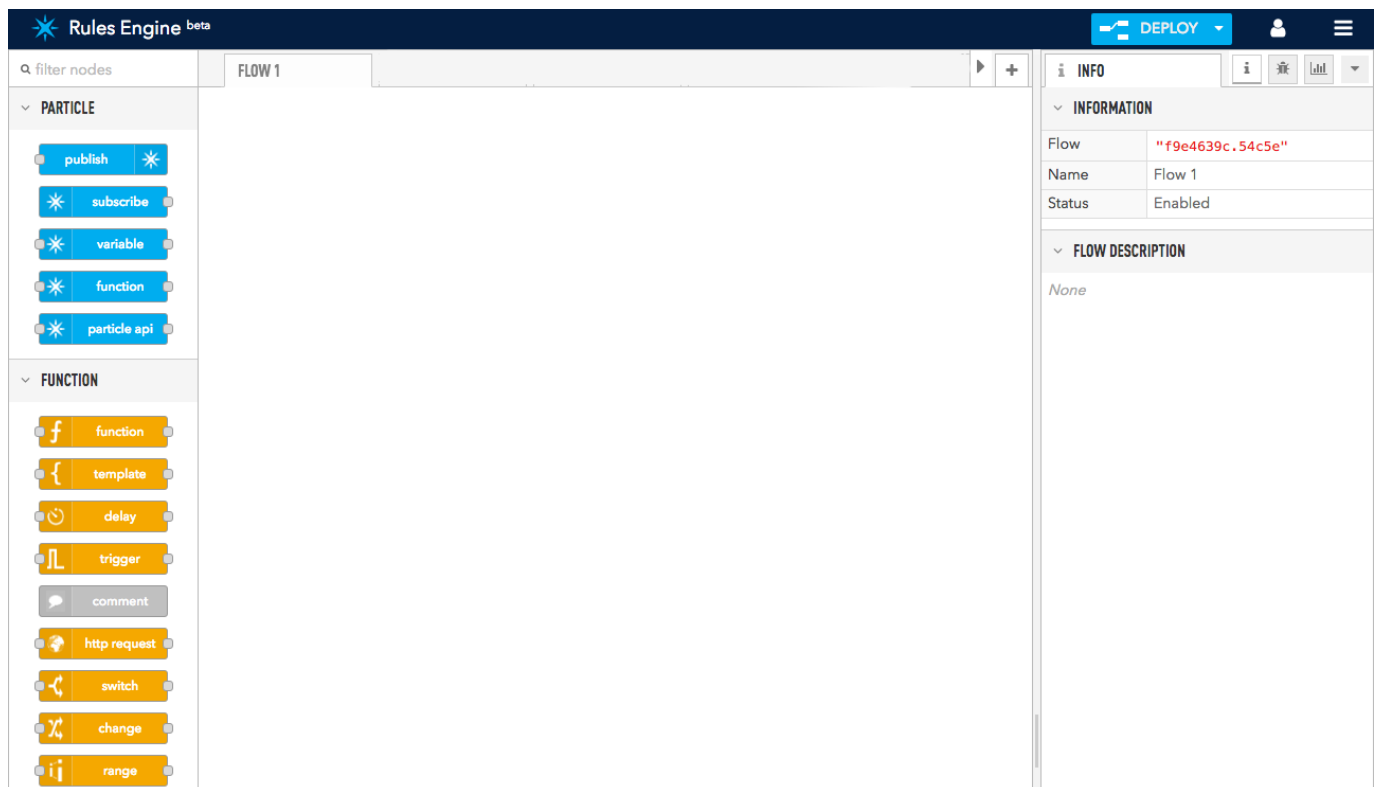
## Getting Started

Let's get you up and running with the Rules Engine.

To apply for beta access to the Rules Engine, please visit <https://www.particle.io/iot-rules-engine>. Once accepted, you should get an email with your Rules Engine instance URL. The URL should look something like `https://myname.rules.particle.io`.

When visiting your Rules Engine instance, you will first be asked to log in with your Particle account. Only approved members of your team will be given access to your Rules Engine.

After successfully logging in, you will be taken to the Rules Engine workspace.



On the left, you'll see the [palette](#) of nodes available to you. Nodes in the palette are categorized, and can be filtered using the search box above the palette pane.

On the right-hand side of the screen, you'll notice an info pane that can provide you with context and instructions as you build flows. By default, you will see metadata on the flow you're currently working on, including its name and ID. The right pane can also be toggled from *info* to *debug*, which can provide helpful debugging information.

The majority of the screen is dedicated to your flow-builder. Drag-and-drop any node onto the grid to incorporate it into a flow. Your flow is given a default name of "Flow 1." Double-click the tab at the top of the flow builder to rename it at any time. You can also click on the **+** button in the top-right corner of the flow builder to create new flows.

## Hello World Flow

We can start with the simplest flow possible: injecting some data in, and logging it out using the debug node.

First, find the **inject node** in your palette on the left pane. The inject node does what you would expect – injects a message to initiate a flow. The message can take on different flavors, like a piece of text, a number, a timestamp, or a more complex structured data object.

Drag the inject node onto your flow builder grid. You should see the right pane change to include information about the inject node.

The screenshot shows the Rules Engine interface. On the left, the 'INPUT' section of the node palette contains an 'inject' node. In the center, a 'timestamp' node is placed on the flow grid. On the right, the 'INFO' pane displays details for the selected 'inject' node:

INFORMATION	
Node	"74c56b41.2c8b44"
Type	inject

Below the information pane, the 'NODE HELP' section provides a description: 'Injects a message into a flow either manually or at regular intervals. The message payload can be a variety of types, including strings, JavaScript objects or the current time.' The 'Outputs' section lists 'payload' (various) and 'topic' (string).

*Click and drag the inject node onto the grid to add it to your flow*

By default, the inject node will initiate a flow with a timestamp. Double click on the inject node to customize its behavior. Let's inject the current time at an interval of every 5 seconds as shown below:

### Edit inject node

Delete Cancel Done

▼ **node properties**

✉ Payload

☰ Topic

🔄 Repeat

every

Inject once at start?

📌 Name

Click **Done** to save the node and return to the flow builder. Nice! You've added your first node. You'll notice that the inject node has a blue dot in the top-right corner. This means that there have been changes to this node that need to be deployed (more on that later).

Next, find the green **debug node** and drag that onto the grid. Note the node connectors on the right edge of the inject node, and the left edge of the debug node. Connect the inject node to the debug node by drawing a line from one node connector to the other with your mouse. This is the method by which you connect nodes to one another to allow them to pass data through the flow.

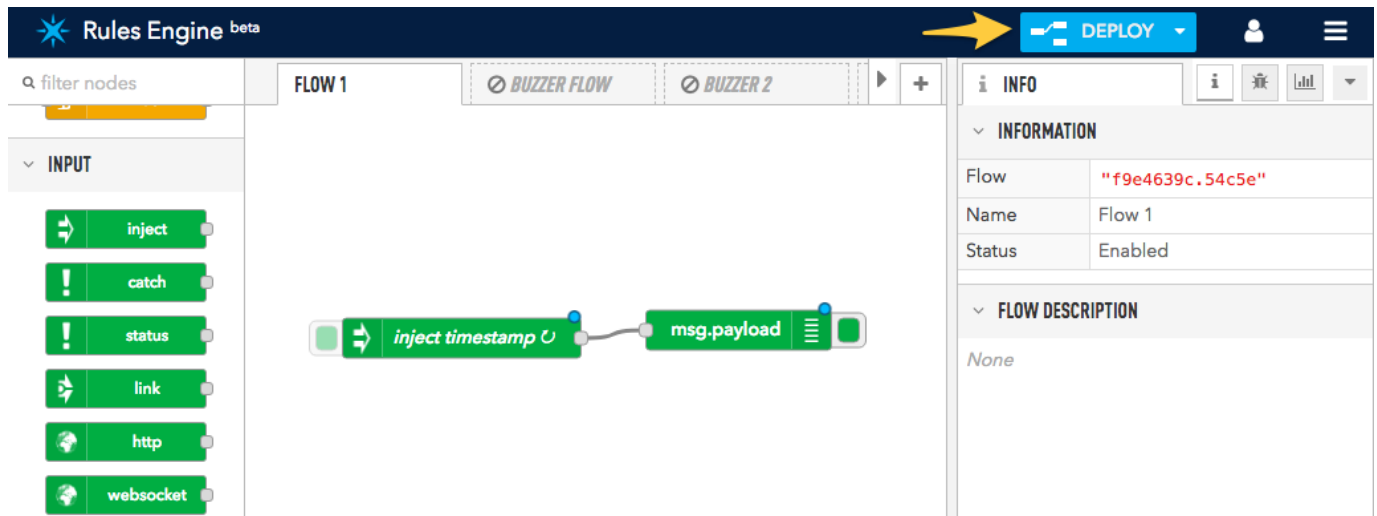


*Nodes are connected to one another to pass data through the flow*

In order for the flow to begin running, you must **deploy** it. Deploying the flow saves any pending changes and begins processing inbound messages. Whenever there are pending

changes, the deploy button will illuminate in the nav bar.

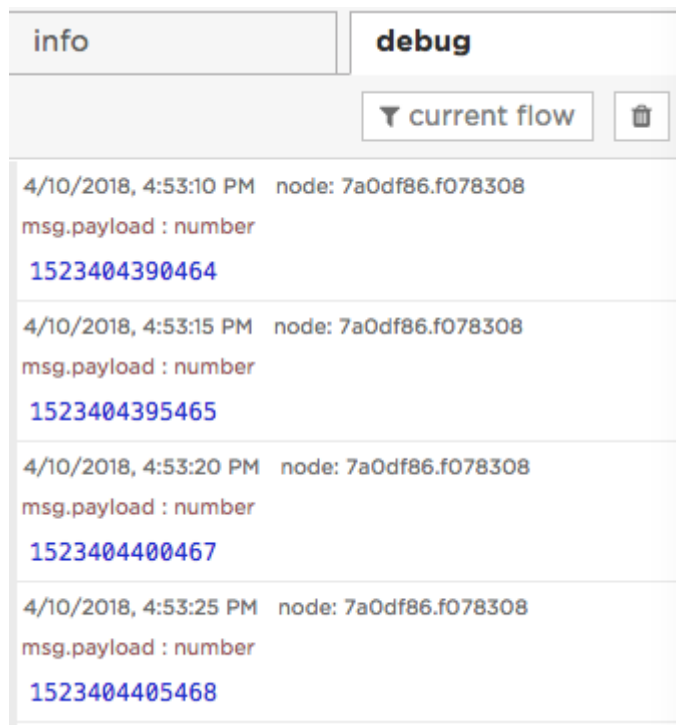
Click the deploy button to get our Hello World flow up and running:



*Deploy flows to begin processing messages and pick up on any pending changes*

You should receive notification that your flow has been successfully deployed. In addition, the blue dots have disappeared, suggesting the pending changes have been deployed.

At the top of the right pane, click on the **debug tab**. Voila! You should see a new up-to-date timestamp added to the debug logs every 5 seconds.



At any time, you can click on the trash can icon to clear the debug log. You can also filter the logs to all nodes, selected nodes, or the current flow.

Note that the debug pane is actually logging `msg.payload`. The timestamp is being attached to the `payload` key of the message object. This method of accessing properties of a message using `msg.[property]` is a very common nomenclature in the Rules Engine.

Congrats! You've built your first flow successfully.

## Adding Nodes to Your Palette

Beyond the nodes included with the default palette, you have the ability to install a variety of open-source nodes that have been contributed to Node-RED by the community.

There are thousands of community-contributed nodes and flows that extend the capability of the Rules Engine. Many of these nodes provide integrations with 3rd-party tools and services.




To explore the library of open-source nodes available to you, click on **≡ > Settings** in the top nav bar.

From here, click **Palette**, then the **Install** tab. You now can search the library of nodes to find the one that suits your needs:



### User Settings

[Close](#)

View	Nodes	<b>Install</b>	
Keyboard	sort: <a href="#">a-z</a> <a href="#">recent</a> <a href="#">↻</a>		
🔍 twilio <span style="float: right;">3 / 1393 ✕</span>			
<b>Palette</b>	 <b>node-red-node-twilio</b> <a href="#">↗</a> A Node-RED node to send SMS messages via the Twilio service. 📅 0.1.0 📅 2 weeks ago <span style="float: right;">installed</span>		
	 <b>node-red-bluemix-nodes</b> <a href="#">↗</a> A collection of extra Node-RED nodes for IBM Bluemix. 📅 1.1.10 📅 1 year, 2 months ago <span style="float: right;">install</span>		
	 <b>node-red-contrib-sms-twilio</b> <a href="#">↗</a> A Node-RED node to send bulk SMS messages via the Twilio service. 📅 0.0.2 📅 1 year, 7 months ago <span style="float: right;">install</span>		

When you find the node you're looking for, click the **install** button to add it to your palette. You will need to confirm this action.