

Getting Started with Octoblu



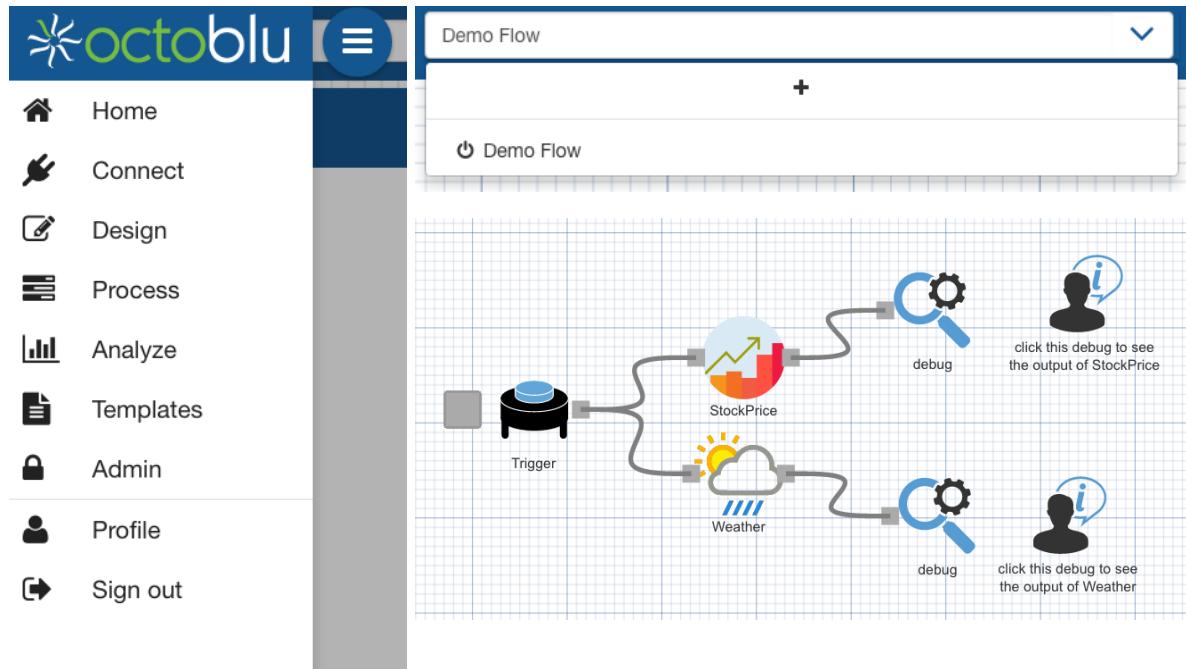
Octoblu enables companies to create IoT services with secure real-time exchange of data. The services are built on an open communications and management platform that supports a variety of protocols for physical devices to communicate seamlessly with each other, people, legacy applications, and cloud services. Through public, private, or hybrid clouds users can connect, design, process, and analyze the flow of information. All services have been designed through a robust security and right management architecture.

Using Octoblu APIs you can easily integrate existing or new applications on to the Octoblu IoT platform. In this way you can add exponential value to your product via the power of being connected to a myriad of platforms, devices, and web-services through one messaging standard.

We've provided resources to get started with our API, applications, visual designer, and open plugin architecture. Our goal is to help you harness the power of the Octoblu ecosystem to meet your IoT goals.

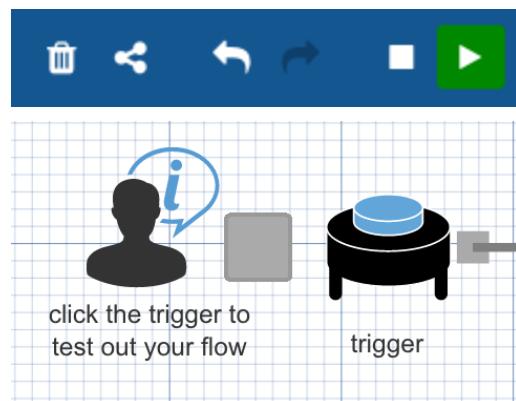
<http://www.citrix.com/go/citrix-developer/octoblu-developer-community.html>

Open the Design Page



The screenshot shows the Octoblu web interface. On the left is a vertical sidebar with the Octoblu logo at the top. Below the logo are eight menu items: Home, Connect, Design, Process, Analyze, Templates, Admin, Profile, and Sign out. The 'Design' menu item is highlighted with a blue background. To the right of the sidebar is the main workspace. At the top of the workspace is a header bar with the text 'Demo Flow' and a dropdown arrow. Below the header is a search bar with the placeholder text 'Demo Flow'. The main area contains a flow diagram on a grid. The flow starts with a 'Trigger' block (represented by a grey square with a black circle) connected to a 'StockPrice' block (represented by a blue circle with a yellow arrow). The 'StockPrice' block is connected to a 'Weather' block (represented by a yellow sun inside a blue cloud). Both the 'StockPrice' and 'Weather' blocks have associated 'debug' blocks (represented by blue magnifying glasses with a gear icon) connected to them. To the right of the 'StockPrice' block is a callout bubble with the text 'click this debug to see the output of StockPrice'. To the right of the 'Weather' block is another callout bubble with the text 'click this debug to see the output of Weather'.

Deploy and Test Flow



The screenshot shows a simplified interface for deploying and testing a flow. At the top is a blue header bar with several icons: a trash can, a share icon, a left arrow, a right arrow, a square, and a green play button. Below the header is a flow diagram on a grid. It consists of a 'trigger' block (a grey square with a black circle) and a 'Trigger' block (a black circle with a blue button). A callout bubble to the left of the first 'Trigger' block says 'click the trigger to test out your flow'. A callout bubble to the right of the second 'Trigger' block says 'trigger'.

Open Debug Console

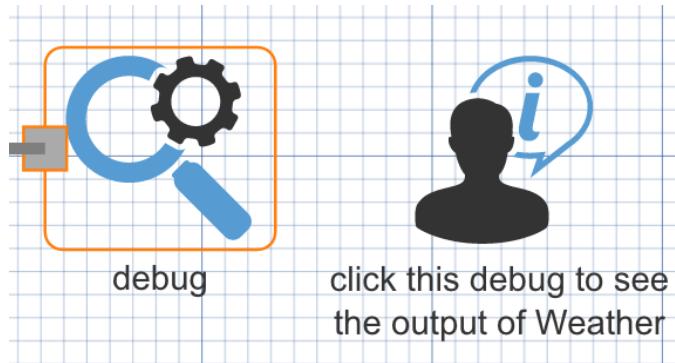


1/28/15 12:18 PM - debug

```
{  
  "date": "2015-01-28T19:18:17.476Z",  
  "message": {  
    "node": "Zeb29667-a722-11e4-81b1-19e62bd918bd",  
    "name": "debug",  
    "msgType": "input",  
    "msg": {  
      "temperature": 68.59  
    }  
  }  
}
```

You'll see output like this = "msg" : { "temperature" : 68.59 }

This is the **JSON message** that was sent from the **Weather** node to the **debug** node.



You can tie the output of any node to a **Debug**.
You can also flip a **debug flag**.

—● Debug

Select a channel node to edit it. You can search for and select any available API endpoint and the parameters will appear.

The screenshot shows two separate channel configuration panels. The left panel is for 'Edit > StockPrice' and the right panel is for 'Edit > Weather'. Both panels have a dark blue header with a question mark icon and a close button. The StockPrice panel has a 'Name' field with 'StockPrice' and a 'Filter Endpoints' dropdown. The Weather panel has a 'Name' field with 'Weather' and a 'Filter Endpoints' dropdown. Both panels have an 'Endpoint' section with dropdown menus showing 'Get Last Trade Price' and 'Get Temperature In Fahrenheit' respectively. Below the endpoints are 'Symbol' fields ('GOOG' for StockPrice, empty for Weather), 'City' fields ('Chandler' for Weather, empty for StockPrice), 'State' fields ('Arizona' for Weather, empty for StockPrice), and 'Country' fields ('USA' for Weather, empty for StockPrice). Each panel has a 'Debug' toggle switch at the bottom. The StockPrice panel also has a 'Symbol' field with the value '{{msg.payload}}' and a 'Payload' field with the value 'GOOG'.

We're going to edit the **Stock price node**.

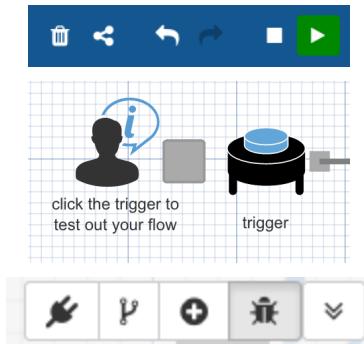
Using **{{double curly braces}}** you can **reference an incoming message**. Earlier we saw that the message the Weather node outputs is a **JSON** object of `msg.temperature`.

The **trigger node** can be told to send out a specified message as `msg.payload`

The screenshot shows a 'trigger' node configuration panel. The header is 'Edit > trigger' with a question mark and close icon. The 'Name' field is 'trigger'. The 'Topic' field is empty. The 'Payload Type' section shows 'String' selected (radio button is green). The 'Payload' field contains the value 'GOOG'. Below the payload field is a 'Debug' toggle switch. To the right of the trigger panel is a 'Payload Type' section with three options: 'Timestamp' (radio button is empty), 'Blank' (radio button is empty), and 'String' (radio button is green). Below that is a 'Payload' section with the value 'GOOG' and a 'Debug' toggle switch.

We're going send out a Stock Symbol

Re-Deploy your flow, hit the trigger, and open the debug console. You'll see that the "GOOG" stock price is shown. The **Stock node** referenced the message that was passed from the trigger and used that in an **API call** to the Stock Price channel.

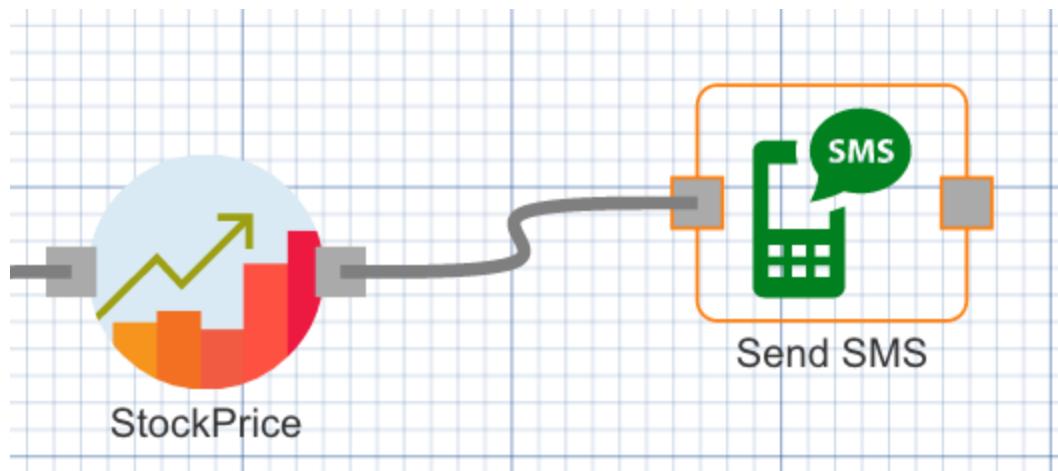


Lets edit our flow to send the Stock price as an SMS!

Open the Configured nodes tab.



Select the SMS node and drop it into your flow.



Wire your Stock Price node to the SMS node.

Edit your SMS node as such.

- Add your number and include a Country code, use “1” for the US.
- Reference `msg.price` , the expected output of the Stock Price node

Edit > Send SMS ? X

Name
Send SMS

Filter Endpoints

Endpoint
/message

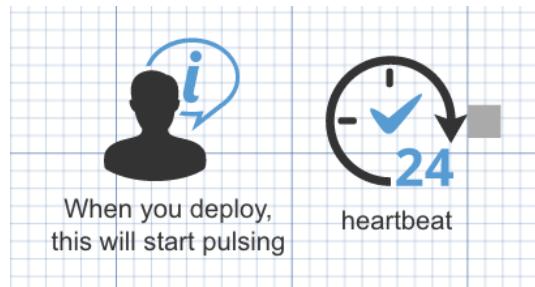
POST http://sms.octoblu.com/message

Destination Number(s)
1800YOUFLOW

Message (Add data to text using {{payload.values}})
{{msg.price}}

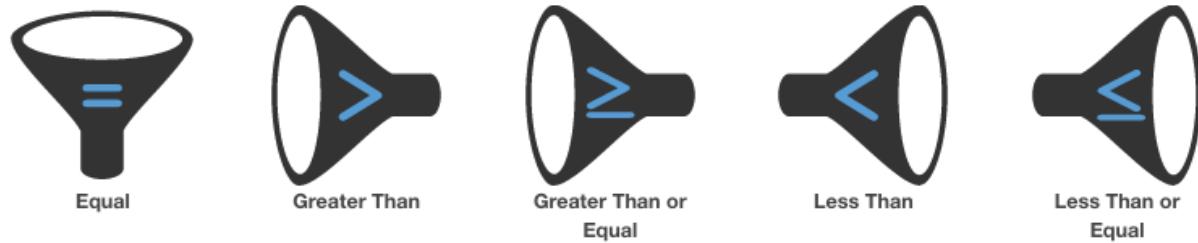
○ — Debug

Re-Deploy, and press the trigger! You should receive a text message in a few seconds with the current **GOOG Stock Price!**



You can use the **interval node** in place of a **trigger**. It will behave the same way except at an **interval in milliseconds!**

There is more functionality that you can add from the **Logic Operators tab**. For example, you can use these comparator nodes to control the flow of messages based on different conditions.



Edit > Greater Than

?

X

Name

Greater Than

Left

`{{msg.payload}}`

Right

`> 10`

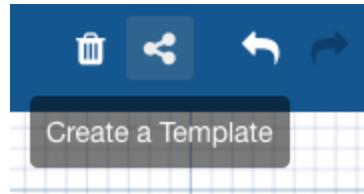
Debug

This screenshot shows the configuration of a 'Greater Than' logic operator node. The node is titled 'Edit > Greater Than'. It has a 'Name' field set to 'Greater Than'. The 'Left' input is connected to the expression `{{msg.payload}}`. The 'Right' input is connected to the value `10`. A 'Debug' button is visible at the bottom.

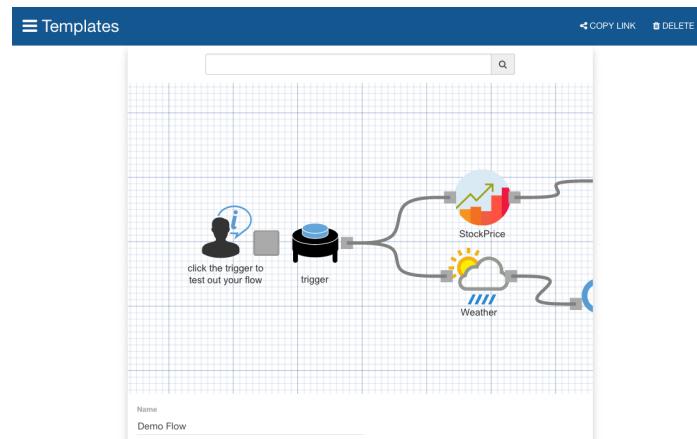
For example, using the **Greater Than** node to check the referenced `msg.payload` to see if it is greater than 10. If the condition is met, `msg.payload` will be passed to the output of the Greater Than node. Anything wired on that other side will receive the original `msg.payload`.

Sharing and Importing Flows

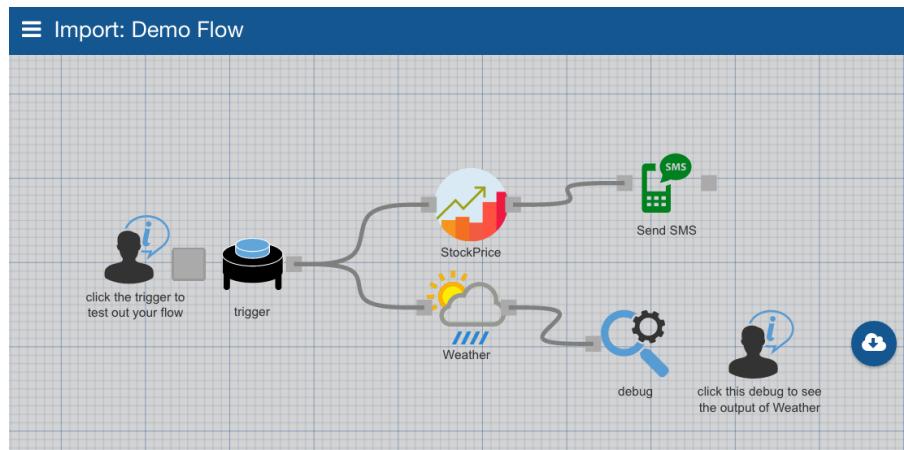
Click the **Create a Template** icon.



In the follow page, name your flow and then click **copy link**



That link can be shared and will open to an Import flow page. Press the **cloud icon** to add it to your account.



Congratulations!

You should now have a basic mastery of Octoblu usage! The best way to get better at creating amazing IoT applications is to play and experiment. Use triggers and the debug node to test and see the output from different nodes to help yourself understand more. We'll be posting more tutorials and documentation. Also be sure to checkout www.hackster.io/octoblu for tutorials on adding hardware and devices to your project using Gateblu and Microblu!

The diagram illustrates the Octoblu ecosystem. At the top is a cloud icon containing the Octoblu logo and text: "octoblu M2M Instant Messaging IoT Platform". Below the cloud are three hexagonal icons representing different components:

- Gateblu** (left): Icons for nest, hue, weMo, and INSTEON.
- Mobiblu** (middle): Icons for Apple, Android, fitbit, and hue.
- Microblu** (right): Icons for an infinity symbol and intel.

Text descriptions for each component are as follows:

- Octoblu** - Web app for configuring and securing connections, designing flows, and querying analytics
- Meshblu** - Multi-protocol & API powered open source IoT platform that can be deployed as a cloud or mesh network.
- Gateblu & Mobiblu** - Fixed and mobile gateways for connecting home/office/field devices and BLE wearable devices to Meshblu and Octoblu
- Microblu** - Open source microcontroller OS for connecting Arduino, Spark, Pinoccio, Raspberry Pi, Galileo, Edison, etc to Meshblu and Octoblu