

**ORACLE®**

# Oracle Digital Assistant

## The Complete Training

**Custom Component Development with Mobile Hub**

# Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

# Topic agenda

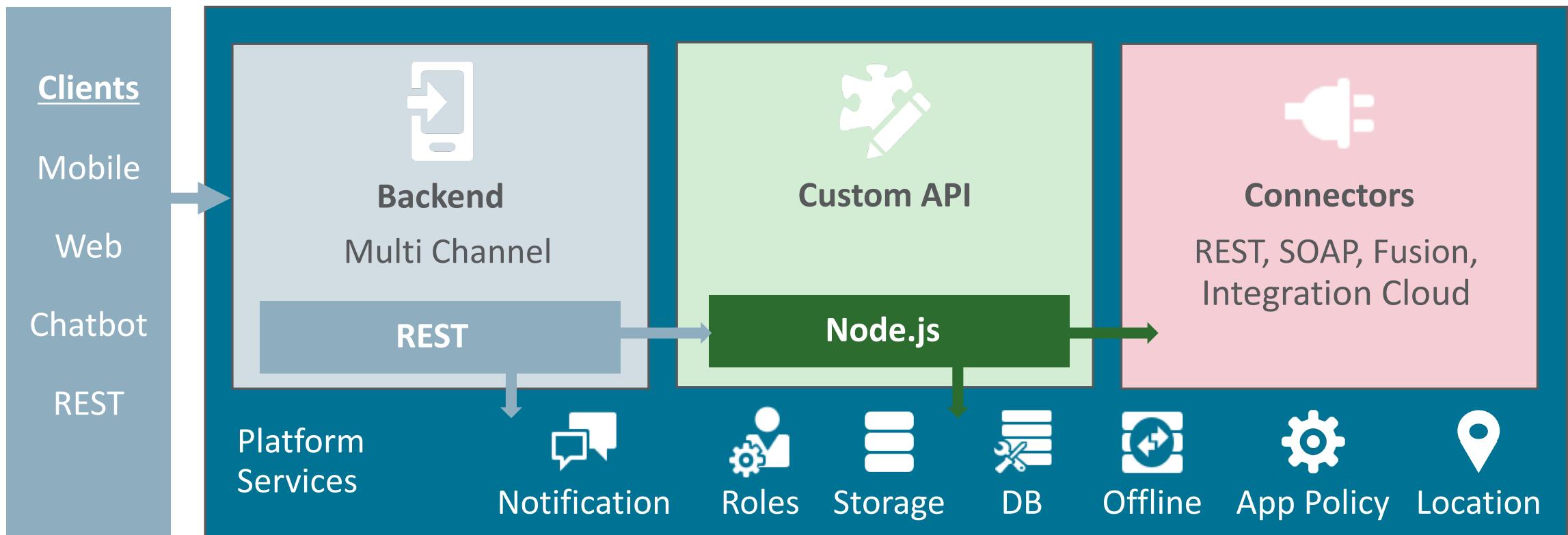
- 1 ➤ Mobile Hub introduction
- 2 ➤ Custom component services in Mobile Hub
- 3 ➤ Building custom components in Mobile Hub
- 4 ➤ Backend integration
- 5 ➤ Local development and debugging

# Topic agenda

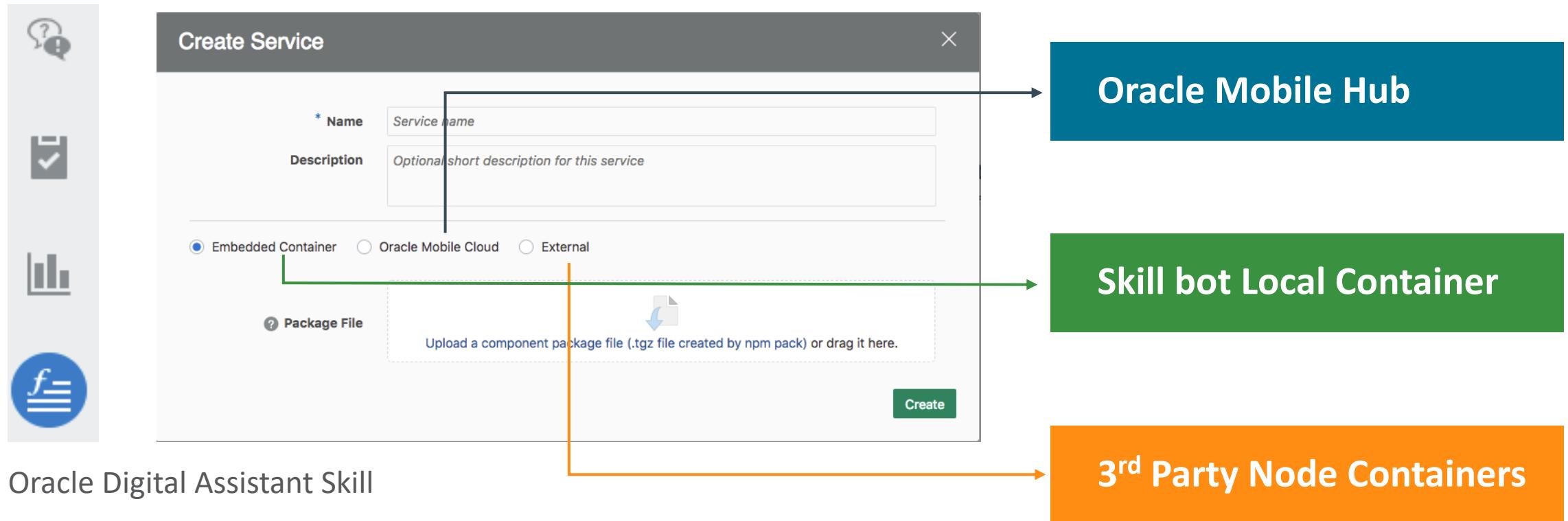
- 1 ➤ Mobile Hub introduction
- 2 ➤ Custom component services in Mobile Hub
- 3 ➤ Building custom components in Mobile Hub
- 4 ➤ Backend integration
- 5 ➤ Local development and debugging

# Oracle Mobile Hub

## Multi channel backend



# Custom component service deployment options



# Mobile Hub benefits

- Multi channel backend service
  - API sharing between web, mobile and bot applications
  - Secure API access
  - Payload shaping
  - Platform services: storage, analytics, database, location, push etc.
- API and API Implementation versioning
- Declarative REST, SOAP and Fusion Apps connectors
- Single point of administration and maintenance
- Diagnostics

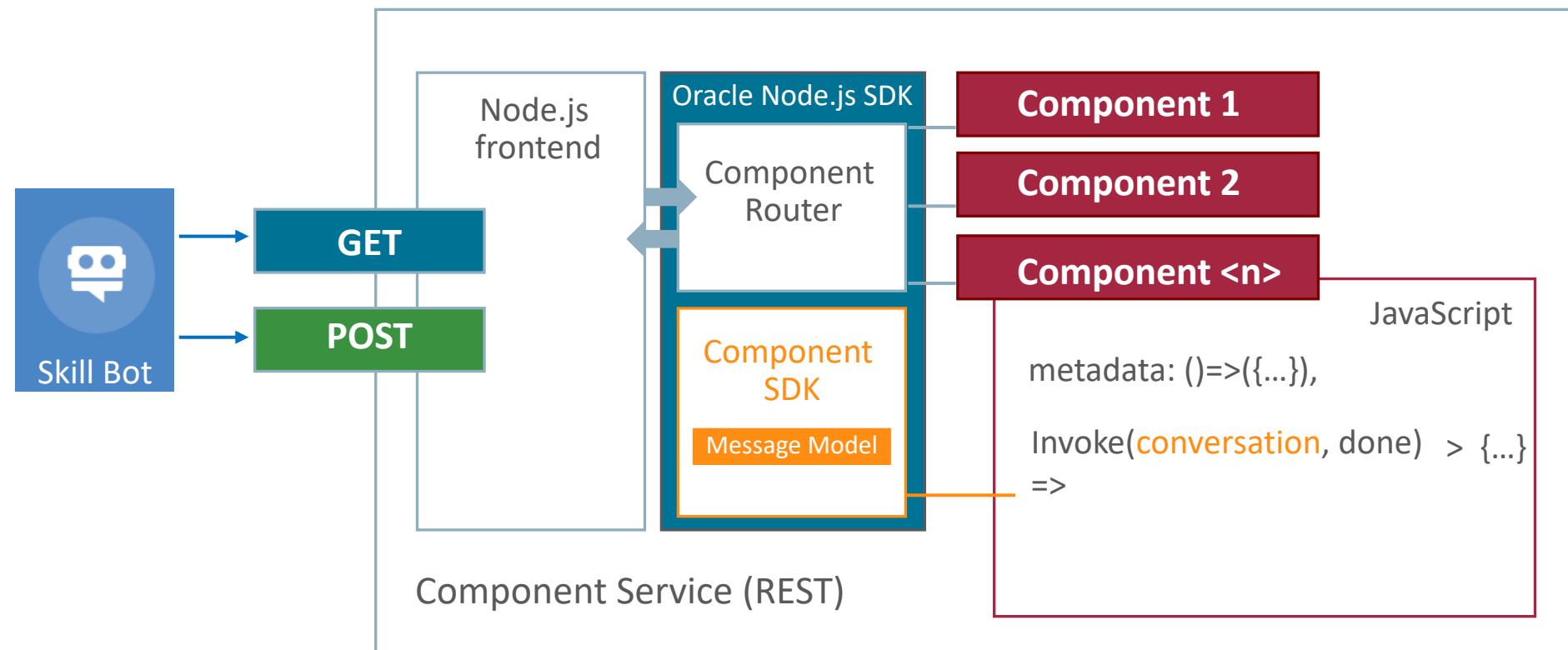
# Topic agenda

- 1 → Mobile Hub introduction
- 2 → Custom component services in Mobile Hub
- 3 → Building custom components in Mobile Hub
- 4 → Backend integration
- 5 → Local development and debugging

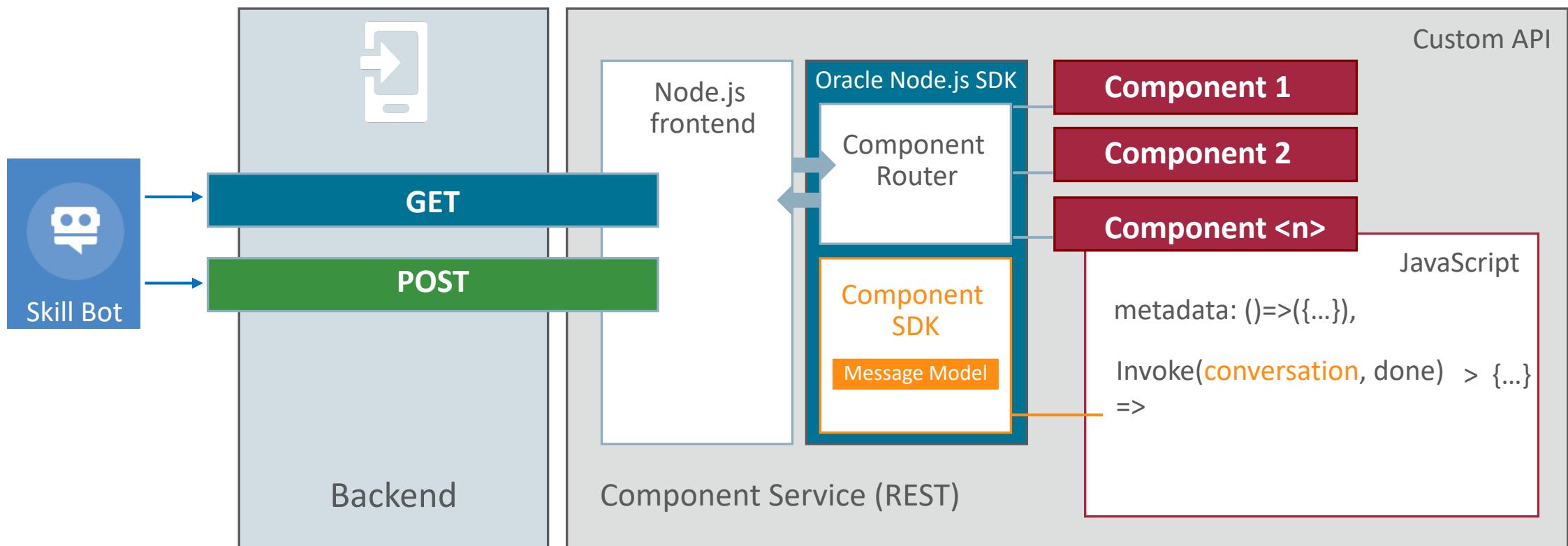
# Custom component service development in Mobile Hub

- Custom component service built as Custom API
  - Component Service exposed through a Backend
  - Access to Mobile Hub services and SDK
  - Leverages Mobile Hub connector framework (REST, SOAP, Fusion, ICS)
- Component service API and Implementation versioned in Mobile Hub
  - Node programming
- Logs and diagnostic information saved in backend analytics

# Custom component service architecture



# Custom component service architecture in Mobile Hub



# Topic agenda

- 1 → Mobile Hub introduction
- 2 → Custom component services in Mobile Hub
- 3 → Building custom components in Mobile Hub**
- 4 → Backend integration
- 5 → Local development and debugging

# Installing Oracle Node.js SDK

- Global installation provides the command line to create custom components
- Requires Node and Node Package Manager (NPM) to be installed
- Open a terminal window and type

## MAC / Linux

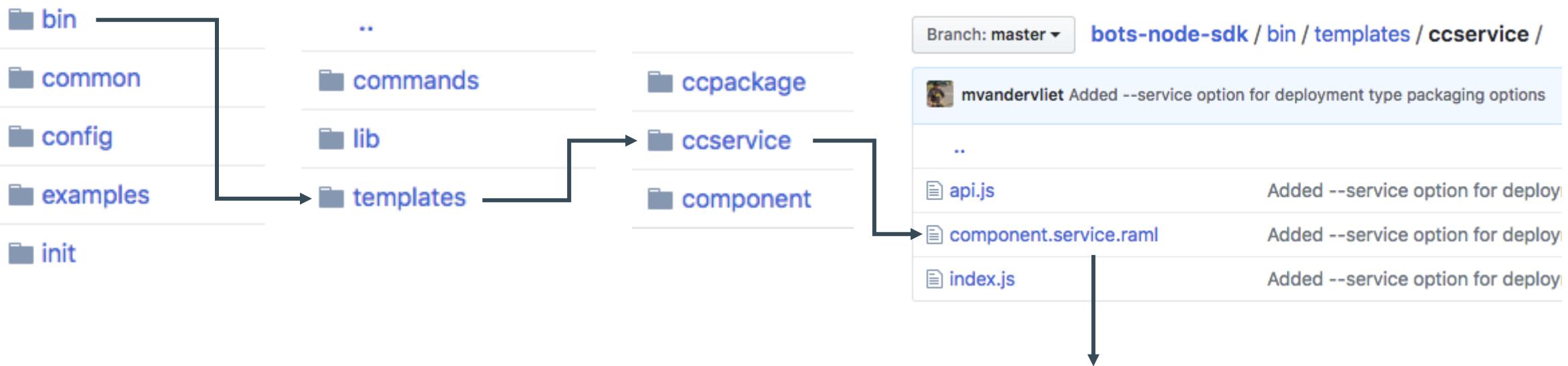
```
sudo npm install -g @oracle/bots-node-sdk
```

## Windows

```
npm install -g @oracle/bots-node-sdk
```

# Downloading the custom API template

- Oracle Mobile Hub Custom API starter template is available in Oracle Bots Node.js SDK on GitHub
  - <https://github.com/oracle/bots-node-sdk>
  - Defines GET and POST methods required for Bot custom component services



# Creating a custom component service API

The diagram illustrates the process of creating a custom component service API using Oracle API Designer.

**Left Panel:** Shows the "New API" creation interface. It includes fields for "API Display Name" (helloworldCCS), "API Name" (helloworldCCS), and "Short Description" (hello world custom component s). A "Create" button is at the bottom.

**Middle Panel:** Shows the "helloworldCCS 1.0" API design screen. It displays the RAML code for the API. A red box highlights the first four lines of the RAML code:

```
1 %%RAML 0.8
2 title: helloworldCCS
3 version: 1.0
4 baseUri: /mobile/custom/helloworldCCS
```

A red arrow points from the "Save" button in the middle panel to the "Save" button in the right panel, indicating where to save the RAML file.

**Right Panel:** Shows the RAML editor with the full RAML code. A red box highlights the entire code area. A red arrow points from the "Copy & paste RAML (keep title, version, baseUri)" text above to the RAML code in the editor.

**Text Labels:**

- Create new API
- Open API
- Copy & paste RAML (keep title, version, baseUri)

# Custom component service endpoints

DEVELOPMENT > APIs > helloworldCCS 1.0

Save Test

The screenshot shows the Oracle API Platform interface for managing API resources. On the left, a sidebar lists categories: General, Endpoints (selected), Security, Schema, Types, and Traits. The main area displays two resource configurations under the 'Endpoints' tab.

**Resource 1:** Path: /components, Display Name: Components context root, Resource Type: G (Global). This is the root endpoint for components.

**Resource 2:** Path: /components/{component}, Display Name: Component invocation, Resource Type: P (Protected). This endpoint allows invocation of specific components.

A green button labeled '+ New Resource' is visible at the top of the main configuration area.

# Disable login requirement

DEVELOPMENT > APIS > helloworldCCS 1.0 Save Test

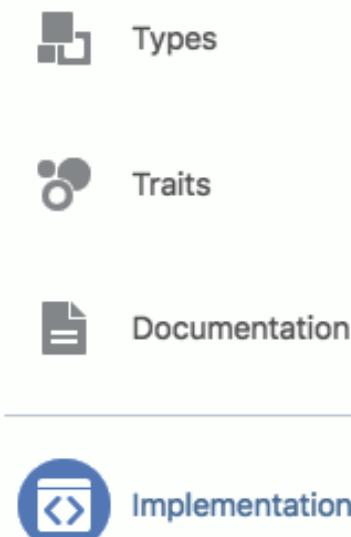
**General** **Endpoints** **Security** **Schema** **Types**

**Info** Login Required controls whether credentials are required to test this API's endpoints. When enabled, credentials are required. Additionally, Enterprise mobile users must have at least one of the roles selected here to access the API. When disabled, credentials are not required. These properties aren't saved in the associated RAML document, so you won't see them in the Source view.

[Tell me more about API security](#)

Login Required

# Downloading the custom API scaffold



You don't have any API implementation

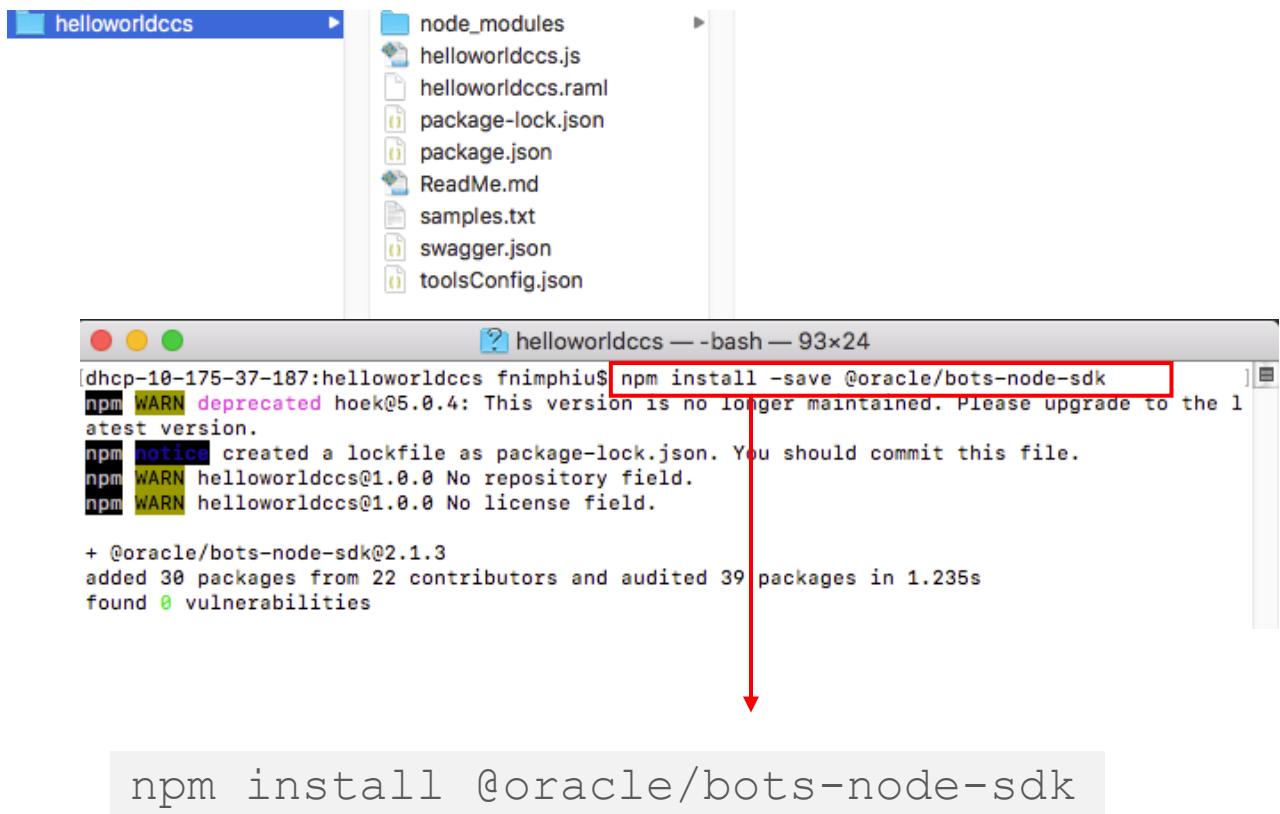
Download a JavaScript scaffold of your API to help you get started, or your implementation is ready to go.

[Tell me what's expected in my implementation archive](#)

 [JavaScript Scaffold](#)

# Setting up the local development environment

- Unzip the downloaded scaffold
- Open command line and navigate into custom API root folder
  - Folder that contains package.json
- Install Oracle Bots Node.js SDK locally
  - Provides custom component SDK
  - Handles component request routing



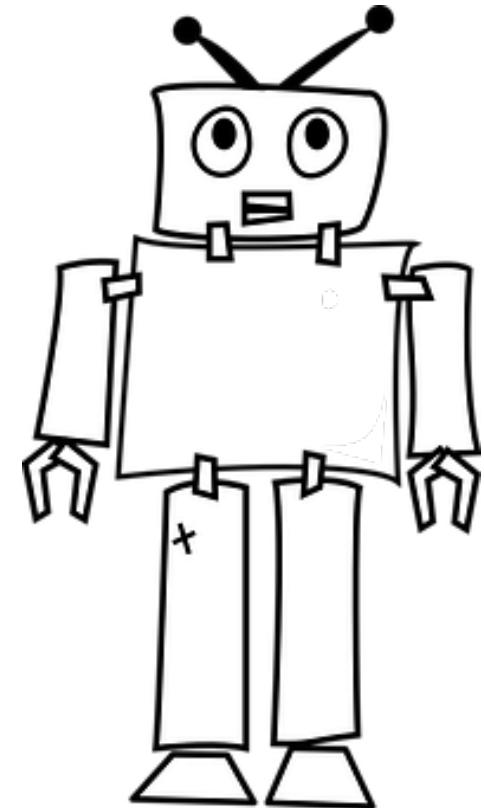
```
helloworldccs └──┬ node_modules
  └──┬ helloworldccs
    ├── helloworldccs.js
    ├── helloworldccs.raml
    ├── package-lock.json
    ├── package.json
    ├── ReadMe.md
    ├── samples.txt
    ├── swagger.json
    └── toolsConfig.json

? helloworldccs — bash — 93x24
[ dhcp-10-175-37-187:helloworldccs fnimphiu$ npm install -save @oracle/bots-node-sdk
npm WARN deprecated hoek@5.0.4: This version is no longer maintained. Please upgrade to the latest version.
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN helloworldccs@1.0.0 No repository field.
npm WARN helloworldccs@1.0.0 No license field.

+ @oracle/bots-node-sdk@2.1.3
added 30 packages from 22 contributors and audited 39 packages in 1.235s
found 0 vulnerabilities
```

npm install @oracle/bots-node-sdk

For better code organization, **create a "components" folder** and then a folder for each custom component you build therein



# Editing the custom API

<https://github.com/oracle/bots-node-sdk/blob/master/bin/templates/ccservice/api.js>

Copy Content from api.js

Copy Content

```
/**  
 * The ExpressJS namespace.  
 * @external ExpressApplicationObject  
 * @see {@link http://expressjs.com/3x/api.html#app}  
 */  
  
/**  
 * Mobile Cloud custom code service entry point.  
 * @param {external:ExpressApplicationObject} service  
 */  
module.exports = function (service) {  
  
    const OracleBot = require('@oracle/bots-node-sdk');  
    OracleBot.init(service);  
  
    // implement custom component api  
    OracleBot.Middleware.customComponent(service, {  
        baseUrl: '/mobile/custom/helloworldCCS/components',  
        cwd: __dirname,  
        register: [  
            '. /components'  
        ]  
    });  
};
```

Edit baseUrl, cwd, register properties

# Custom component service code explained

```
module.exports = function (service) { → Node module definition

  const OracleBot = require('@oracle/bots-node-sdk'); → Load command for Bot Node.js SDK
  OracleBot.init(service);

  // implement custom component api
  OracleBot.Middleware.customComponent(service, { → Request to custom component router
    baseUrl: '/mobile/custom/helloworldCCS/components', → REST URI to invoke custom component
    cwd: __dirname, → (should match your API URI)
    register: [
      './components' → Relative folder custom components are
    ] → Searched in
  });

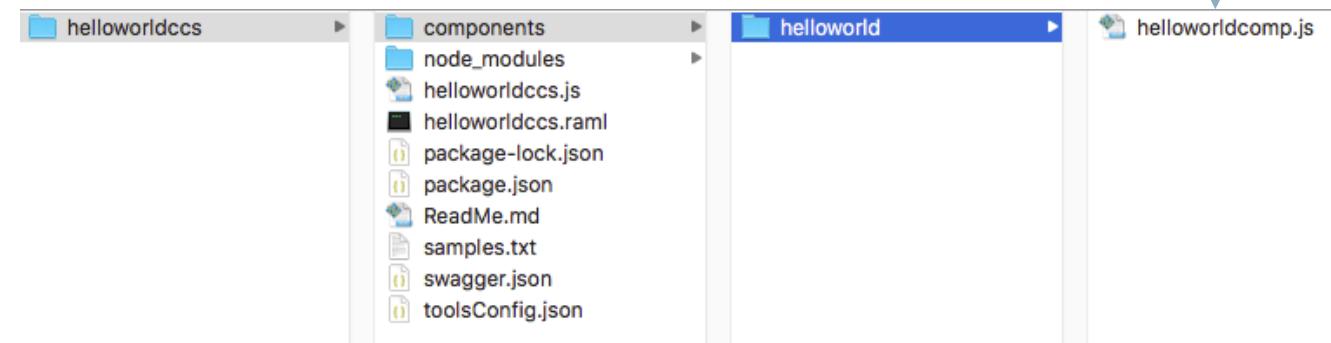
};
```

# Creating a custom component

```
[fnimphiu-orcl:helloworldccs fnimphiu$ bots-node-sdk init component --name helloworldcomp components/helloworld  
Writing file: /Users/fnimphiu/Downloads/helloworldccs/components/helloworld/helloworldcomp.js
```

```
-----  
Added Custom Component: 'helloworldcomp'  
-----
```

```
fnimphiu-orcl:helloworldccs fnimphiu$
```



```
bots-node-sdk init component --name helloworldcomp components/helloworld
```

# Generated custom component file & code

The screenshot shows a code editor with a file named `helloworldcomp.js`. The code defines a component with the following structure:

```
'use strict';

module.exports = {
  metadata: () => ({
    name: 'helloworldcomp',           → Component invocation name
    properties: {
      human: { required: true, type: 'string' }, → Component properties
    },
    supportedActions: ['weekday', 'weekend'] → Action transitions
  }),
  invoke: (conversation, done) => {
    // perform conversation tasks.
    const { human } = conversation.properties();
    // determine date
    const now = new Date();
    const dayOfWeek = now.toLocaleDateString('en-US', { weekday: 'long' });
    const isWeekend = [0, 6].indexOf(now.getDay()) > -1;
    // reply
    conversation
      .reply(`Greetings ${human}`)
      .reply(`Today is ${now.toLocaleDateString()}, a ${dayOfWeek}`)
      .transition(isWeekend ? 'weekend' : 'weekday');

    done();                                → Callback that must be called at the end
  }
};
```

Annotations with arrows point from specific parts of the code to their descriptions:

- `name: 'helloworldcomp'` → Component invocation name
- `properties: { human: { required: true, type: 'string' } }` → Component properties
- `supportedActions: ['weekday', 'weekend']` → Action transitions
- `done();` → Callback that must be called at the end

# Deploying the custom component service to Mobile Hub

- compress project root folder to a *zip-file*
- Upload *zip-file* as custom API implementation
- Use embedded tester in Oracle Mobile Hub to test GET method
- Expose custom API on Mobile Hub backend

The screenshot shows the Oracle Mobile Hub interface. On the left, there is a sidebar with icons for General, Endpoints, Security, Schema, Types, Traits, Documentation, and Implementation. The Implementation icon is highlighted with a blue circle. The main area has a header: "Download a new JavaScript scaffold at any time to include changes you make to the API design." Below this is a green button labeled "JavaScript Scaffold" with a download icon. A table follows, with columns: Status, Default, Name, Version, and Uploaded. It shows two rows: one for "helloworldccs" (Status: warning, Default: checked, Name: helloworldccs, Version: 1.1.0, Uploaded: Fri, 2/15/2019 14:05) and one for "Mock" (Status: info, Default: unchecked, Name: Mock, Version: N/A). Below the table is a section titled "Dependencies for helloworldccs 1.1.0". At the bottom right, there is a dashed box containing a file icon and the text "Upload an implementation archive or...".

Status	Default	Name	Version	Uploaded
!	✓	helloworldccs	1.1.0	Fri, 2/15/2019 14:05
		Mock	N/A	

# 'Deployment' to Oracle Digital Assistant

## Component registration in Oracle Digital Assistant skill

### Oracle Mobile Hub

DEVELOPMENT > BACKENDS > SayHelloBackend 1.0

The screenshot shows the Oracle Mobile Hub interface under the 'Development' section. On the left, there are navigation icons for Diagnostics, Settings, Clients, Security, and APIs. The main panel displays 'Access Keys' for an OAuth Consumer, showing a Client ID (8dcb27eea1994f7da8a4d27981252df8) and a Client Secret (Show). Below this is the 'Environment URLs' section with a Base URL (https://006B186491194B64A833A511C6F8A56) and an 'HTTP Basic' toggle switch (on). The 'Backend ID' field contains the value 97fa003a-8ed3-4fd1-b853-a3358baac132, which is highlighted with a red box and connected by a red arrow to the 'Backend ID' field in the 'Create Service' dialog. The 'Anonymous Key' field also has a red box around it and is connected by another red arrow to the 'Anonymous Key' field in the dialog.

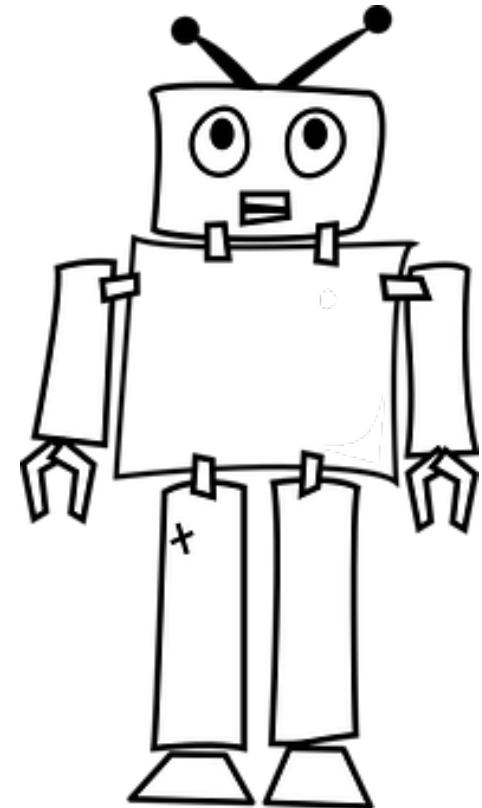
### Skill

The screenshot shows the 'Create Service' dialog for a new skill. The 'Name' field is set to 'MobileHubService'. The 'Backend ID' field is populated with 97fa003a-8ed3-4fd1-b853-a3358baac132. The 'Metadata URL' field contains the URL https://006B186491194B64A833A511C6F8A566.mobile.ocp.oraclecloud.com:443/mobile/. The 'Anonymous Key' field contains the value MDA2QjE4NjQ5MTE5NEI2NEE4MzNBNTExQzZGOEE1NjZfTW9laWxIQW5vbntb3VzX0FQU. A checkbox for 'Use anonymous access' is checked. There is also an 'Optional HTTP Headers' section. A large blue circular icon with a white 'f=' symbol is centered at the bottom of the dialog. A green 'Create' button is located at the bottom right.

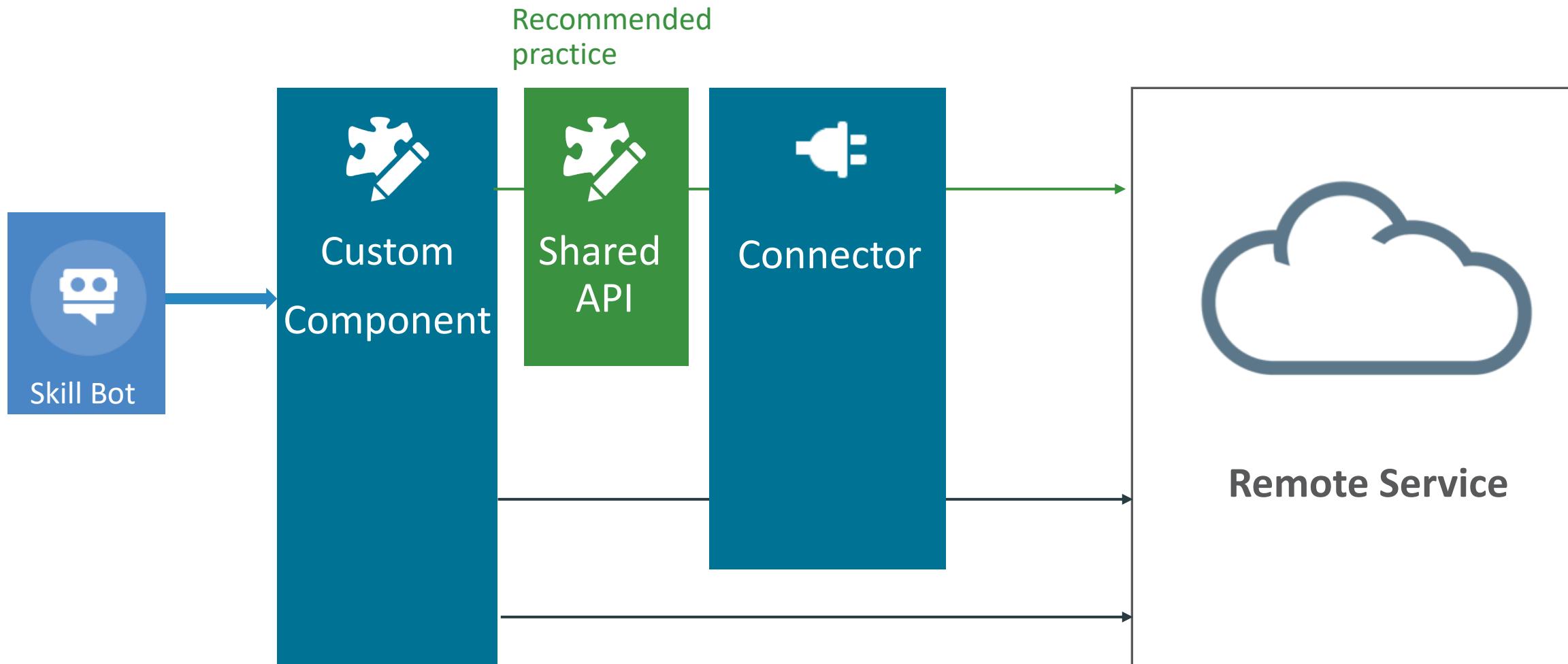
# Topic Agenda

- 1 → Mobile Hub introduction
- 2 → Custom component services in Mobile Hub
- 3 → Building custom components in Mobile Hub
- 4 → Backend integration
- 5 → Local development and debugging

In addition to multi-channel support,  
**backend integration is a good argument**  
**for using Oracle Mobile Hub to create**  
custom components in

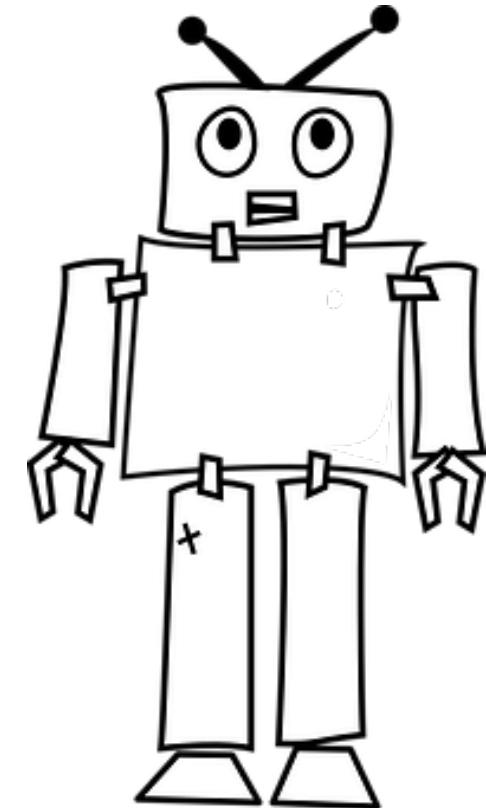


# Mobile Hub backend integration options



You can access the **Oracle Mobile Hub SDK**  
through the **Custom Component SDK** to  
access Mobile Hub custom APIs, platform  
APIs and Connectors

`conversation.oracleMobile.<function>`



# Topic agenda

- 1 → Mobile hub introduction
- 2 → Custom component services in Mobile Hub
- 3 → Building custom components in Mobile Hub
- 4 → Backend integration
- 5 → Local development and debugging

# Custom component debugging with Oracle Mobile Hub

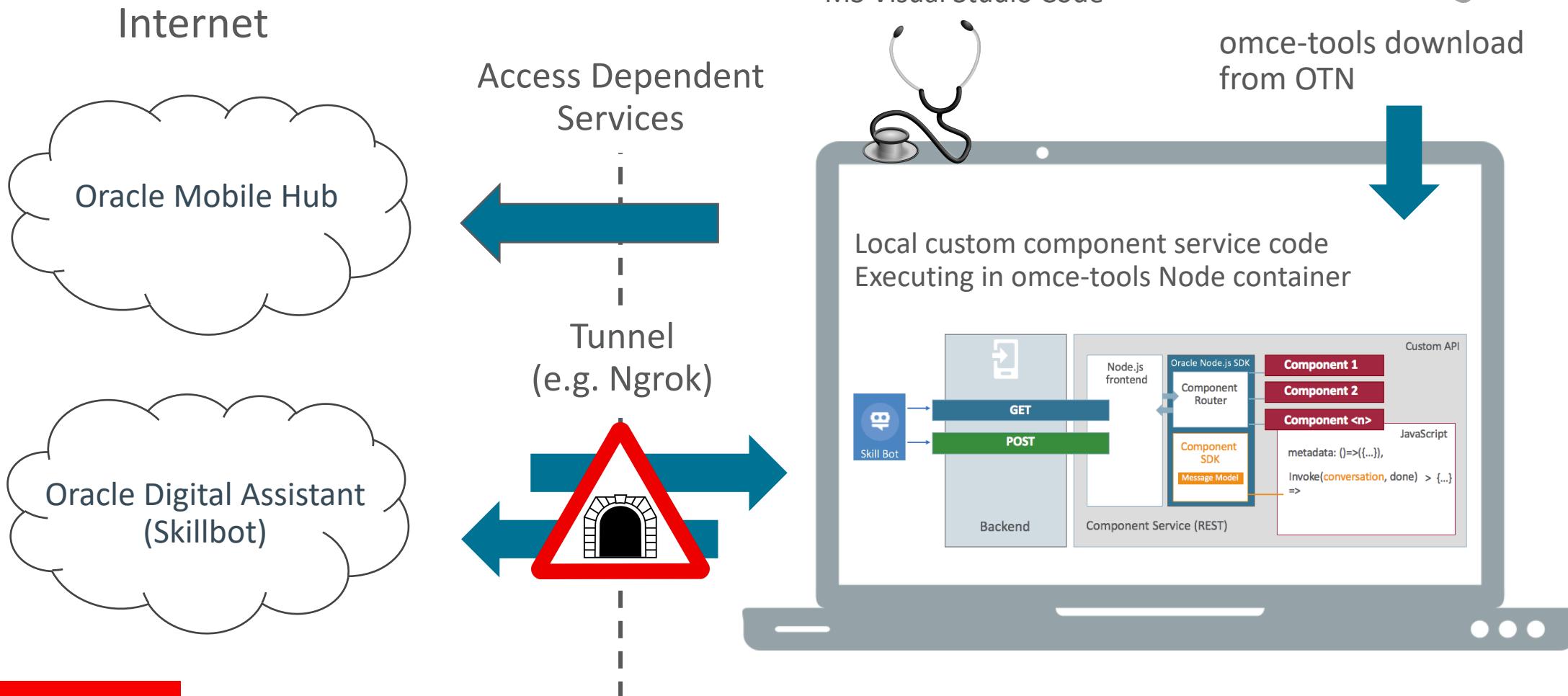
- Install Oracle Custom Code Test Tool
  - Download from OTN
  - Follow instructions in readme
- Configure backend with code test tool reference
  - Code test tool proxy installed as custom API
- Have local copy of component service custom API
  - Configure toolsConfig.json with backend access
- Start Local Debugging
  - Code test tool command

<https://www.oracle.com/technetwork/topics/cloud/downloads/mobile-cloud-service-3636470.html>

The screenshot shows a web page with a navigation bar at the top containing links for 'What's New', 'Downloads' (which is highlighted), 'Articles & Whitepapers', 'Community', and 'Learn More'. Below the navigation bar, the main content area has a title 'Oracle Mobile Hub (OMH) and Oracle Mobile Cloud (MCS) downloads'. A sub-section titled 'OTN License Agreement' contains the message 'Thank you for accepting the OTN License Agreement; you may now download this software.' followed by three dashed lines. Another sub-section titled 'ORACLE MOBILE HUB (OMH) TOOLS' lists a single item: 'Custom Code Test Tools v18.3.1' with a download link labeled 'omce-tools-v18.3.1' and a description 'Custom code offline deployment and debugging'.

Tool	Download	Description
Custom Code Test Tools v18.3.1 09/28/2018	<a href="#">omce-tools-v18.3.1</a>	Custom code offline deployment and debugging

# Mobile Hub debugging architecture



# Integrated Cloud Applications & Platform Services

**ORACLE®**



## Oracle Digital Assistant Hands-On

TBD