

ORACLE®

Oracle Digital Assistant

The Complete Training

Custom Component Development

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.

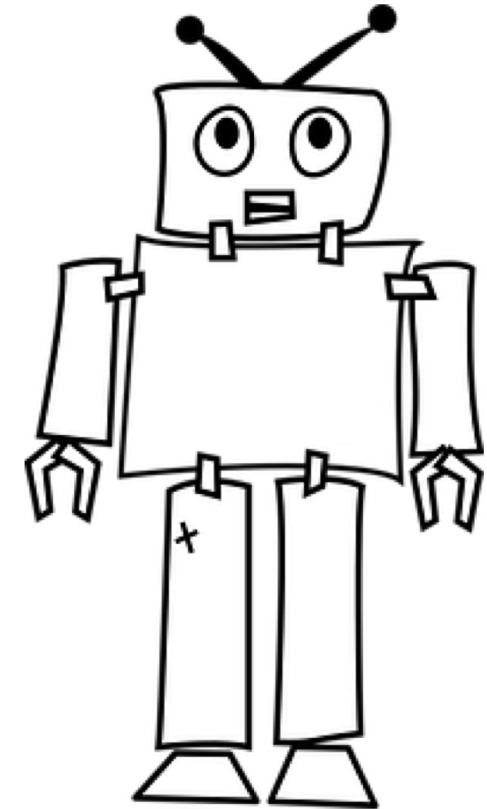
Topic agenda

- 1 Oracle Bots Node.js SDK
- 2 Getting started
- 3 Local container deployment

Topic agenda

- 1 Oracle Bots Node.js SDK
- 2 Getting started
- 3 Local container deployment

The Oracle Bots Node.js SDK is **all the tooling you need** for building custom component services and components



Development environment

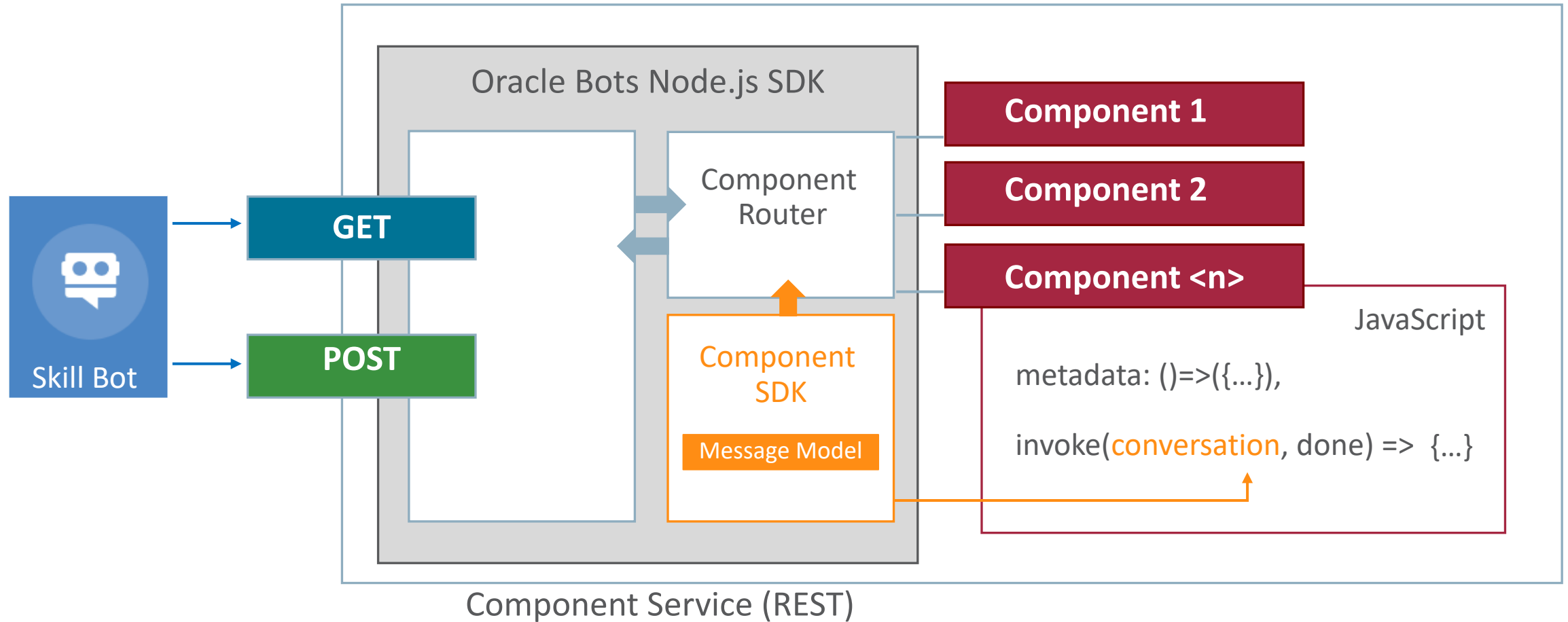
- Node Package Manager, Node version 8.11.4 or lower
 - <https://nodejs.org/en/download/>
- Install JavaScript IDE with support for Node.js debugging
 - Microsoft Visual Studio Code, IntelliJ WebStorm, etc.
- Access to Internet
 - If behind a proxy call
 - `npm config set proxy http://company-proxy:port`
 - `npm config set https-proxy http://company-proxy:port`

Oracle Bots Node.js SDK

<https://github.com/oracle/bots-node-sdk>

- Bots Node.js SDK functionality
 - Request Routing
 - Custom component SDK
 - Helper functions for conversational message model
 - Logging
- Webhook development support
 - Assists in building webhook clients that dispatch between messengers and bots
 - E.g. Alexa integration

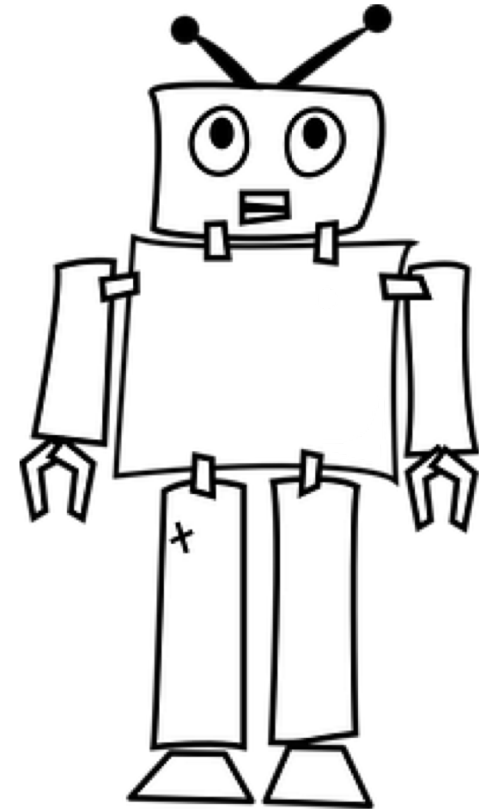
Custom component service architecture

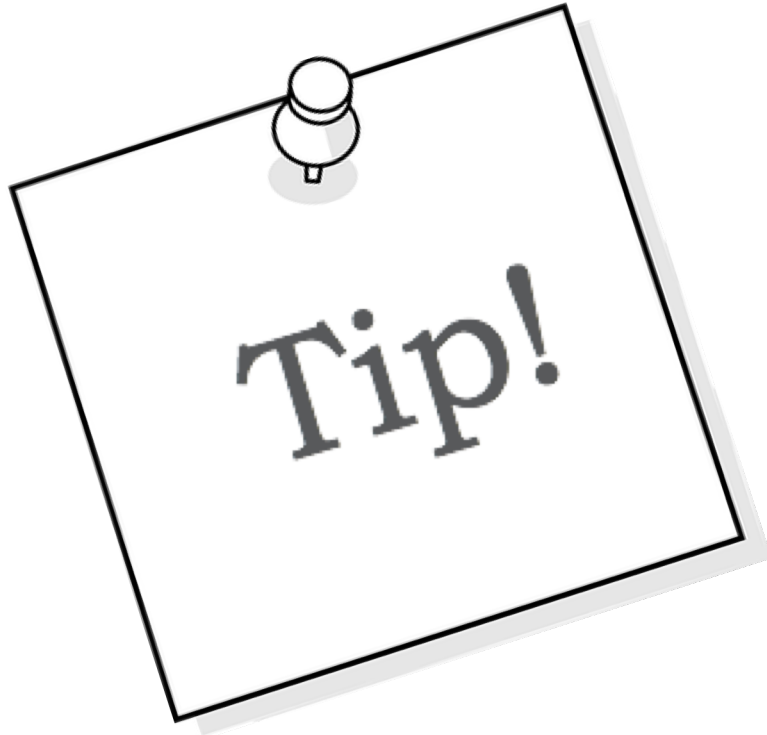


Oracle Bots Node.js SDK installation

- Global installation
 - Install once, use anywhere
- For global installation, open a command line window and type
 - Mac
 - `sudo npm install -g @oracle/bots-node-sdk`
 - Windows
 - `npm install -g @oracle/bots-node-sdk`

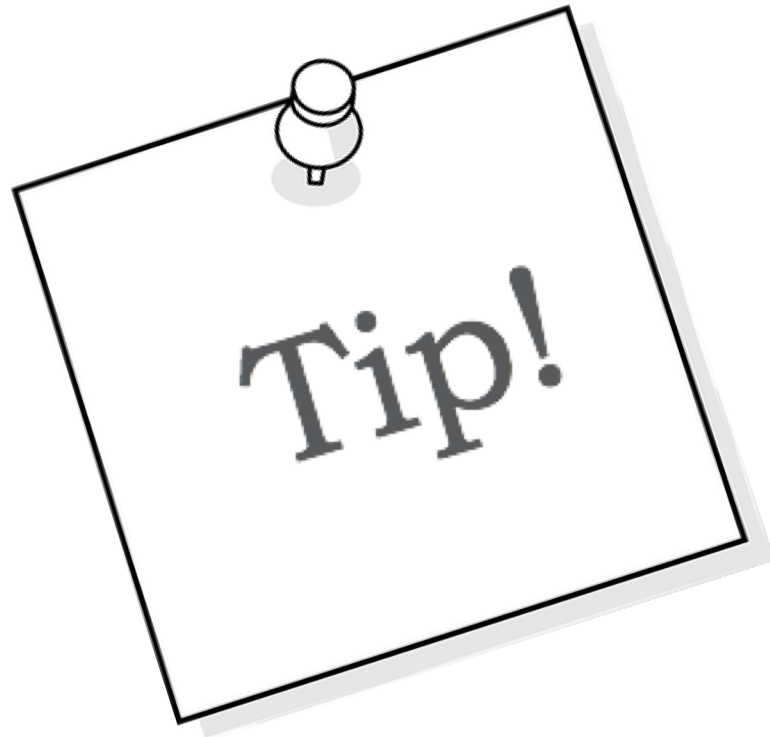
Basically there are **three classes to know about**: `sdk.js`, `shell.js`, `messageModel.js`.





The sdk.js and shell.js classes are physically located in

```
<custom component service>  
|-- node_modules  
    |-- @oracle  
        |-- bots-node-sdk  
            |-- lib  
                |-- component
```



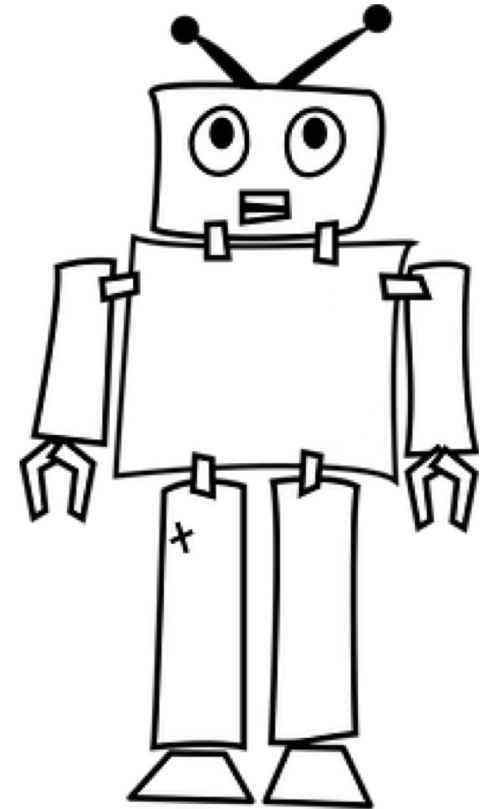
The MessageModel implementation
Is defined in

```
<custom component service>  
|-- node_modules  
    |-- @oracle  
        |-- bots-node-sdk  
            |-- lib  
                |-- component  
                    |-- message  
                        |-- messageModel.js
```

Topic agenda

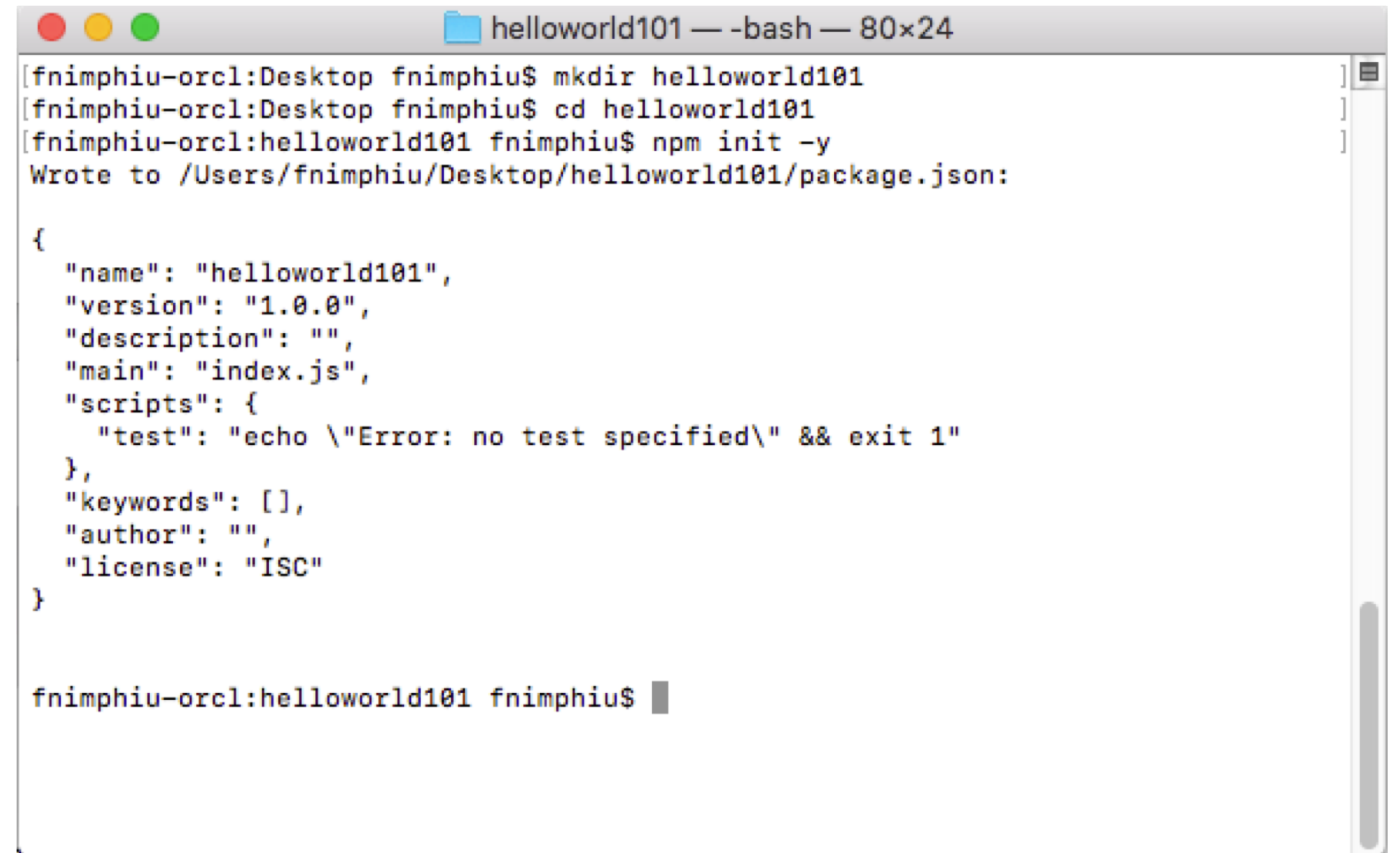
- 1 Oracle Bots Node.js SDK
- 2 Getting started
- 3 Local container deployment

Generations of **software developers**
learned with '**hello world**'.



Creating a "hello world" Node.js project

- Open a terminal window
 - Display Bots Oracle Node.js SDK version to ensure it is available
 - bots-node-sdk -v
- Create folder
 - mkdir helloworld101
 - cd helloworld101
- Initialize node project
 - npm init -y
 - Creates package.json file

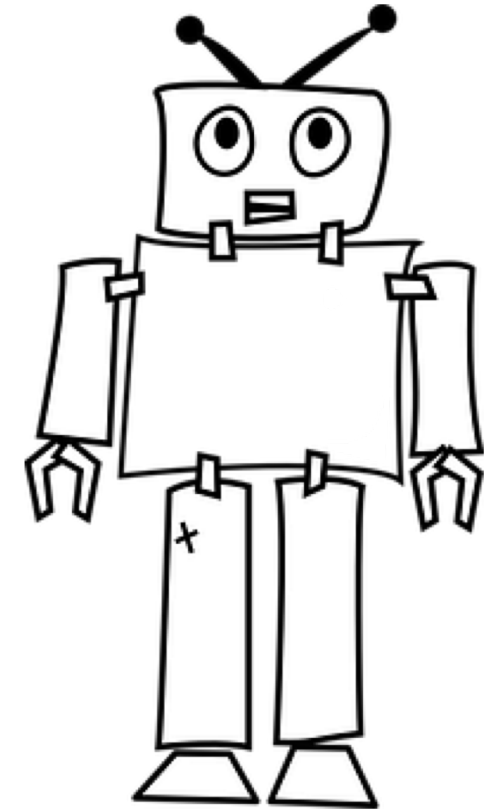
A terminal window titled "helloworld101 — -bash — 80x24" showing the execution of commands to create a Node.js project. The commands are: mkdir helloworld101, cd helloworld101, and npm init -y. The output shows the creation of a package.json file with the following content: { "name": "helloworld101", "version": "1.0.0", "description": "", "main": "index.js", "scripts": { "test": "echo \"Error: no test specified\" && exit 1" }, "keywords": [], "author": "", "license": "ISC" }.

```
helloworld101 — -bash — 80x24
[fnimphiu-orcl:Desktop fnimphiu$ mkdir helloworld101
[fnimphiu-orcl:Desktop fnimphiu$ cd helloworld101
[fnimphiu-orcl:helloworld101 fnimphiu$ npm init -y
Wrote to /Users/fnimphiu/Desktop/helloworld101/package.json:

{
  "name": "helloworld101",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

fnimphiu-orcl:helloworld101 fnimphiu$
```


There is **something to know** about naming the folder where your custom component service resides



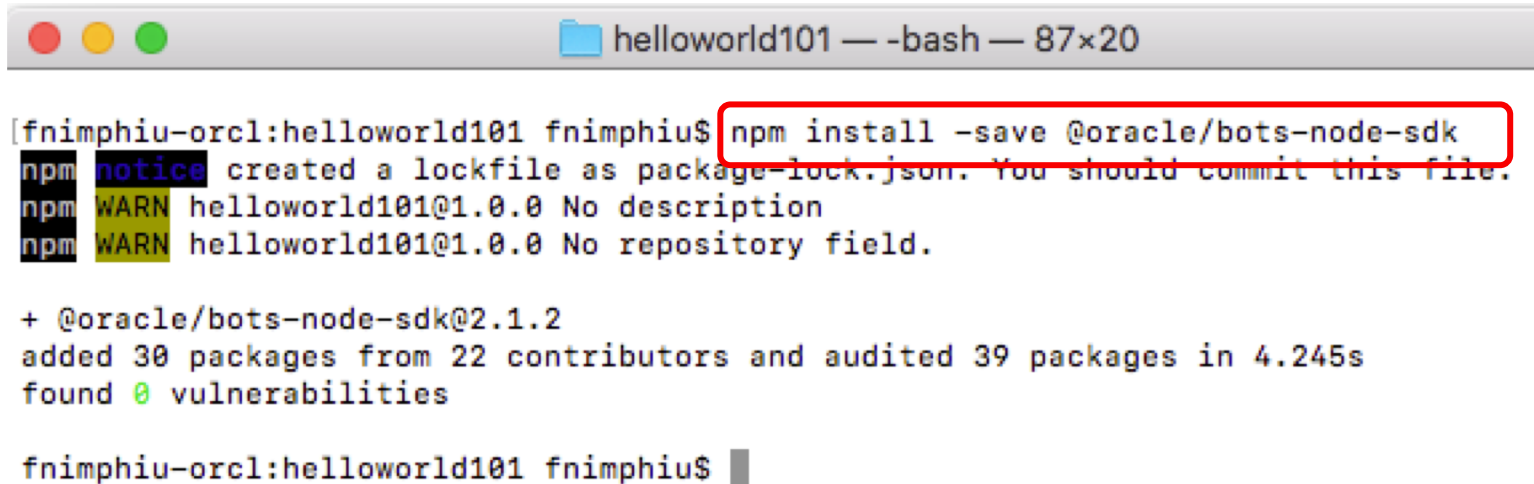
Naming the custom component service folder

- **Rules.** The folder name ...
 - must be less than 214 characters
 - cannot start with a dot or underscore
 - should not use uppercase letters
 - can't contain non-URL-safe characters
- **Tips.** The folder name ...
 - should not match core node modules
 - should not contain '.js' or 'node'

Install Oracle Bots Node.js SDK to the project

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

- Installs Bots Node.js SDK for deployment
- Adds the bot custom component SDK files to project



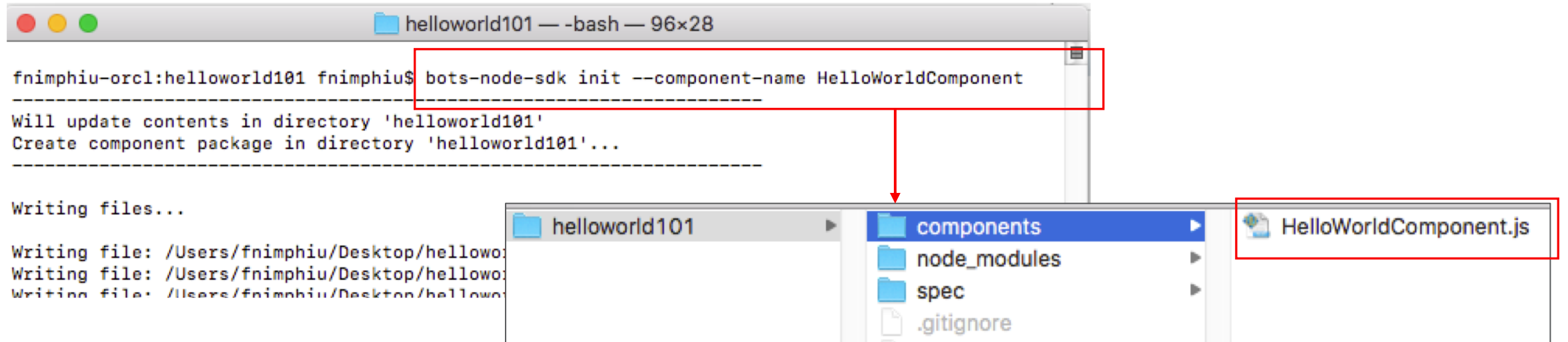
```
helloworld101 — -bash — 87x20
[fnimphiu-orcl:helloworld101 fnimphiu$ npm install -save @oracle/bots-node-sdk
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN helloworld101@1.0.0 No description
npm WARN helloworld101@1.0.0 No repository field.

+ @oracle/bots-node-sdk@2.1.2
added 30 packages from 22 contributors and audited 39 packages in 4.245s
found 0 vulnerabilities

fnimphiu-orcl:helloworld101 fnimphiu$
```

Create a custom component

bots-node-sdk init --component-name HelloWorldComponent



The screenshot shows a terminal window titled 'helloworld101 — -bash — 96x28'. The command `fnimphiu-orcl:helloworld101 fnimphiu$ bots-node-sdk init --component-name HelloWorldComponent` is entered and highlighted with a red box. Below the command, the terminal output reads: 'Will update contents in directory 'helloworld101'', 'Create component package in directory 'helloworld101'...', 'Writing files...', and 'Writing file: /Users/fnimphiu/Desktop/hellowo...'. A red arrow points from the command to a file explorer window. The file explorer shows the directory structure: 'helloworld101' containing 'components', 'node_modules', 'spec', and '.gitignore'. The 'components' folder is selected, and a file named 'HelloWorldComponent.js' is visible in the right pane, also highlighted with a red box.

- All components are created in components folder
 - Can be changed with additional command line argument
- Default component content read from template
 - Hello world type of content

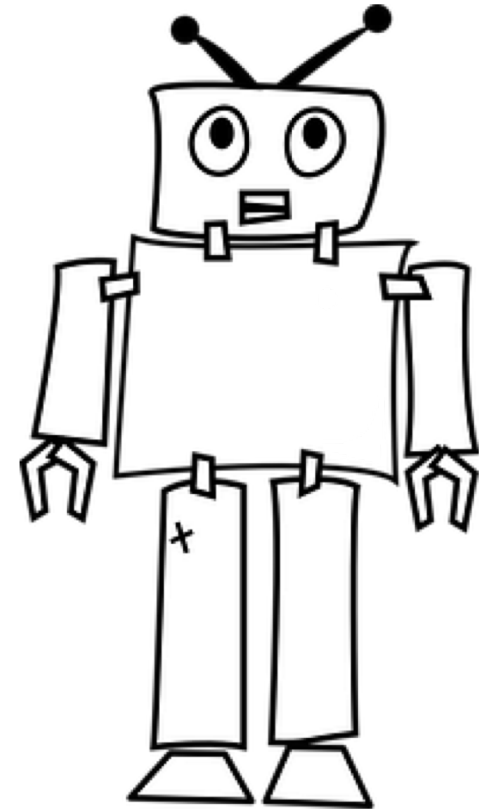


You can use

```
bots-node-sdk init component --name <comp_name>  
components/<sub_dir_name>
```

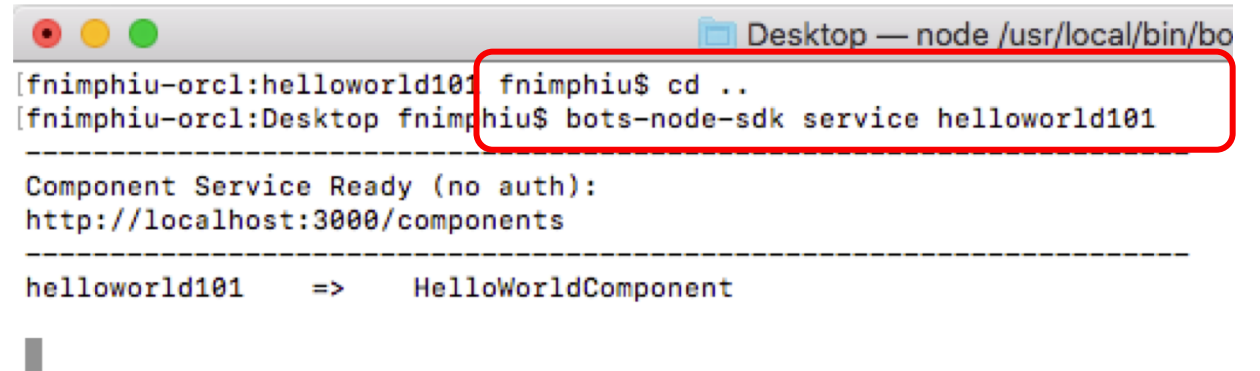
to create additional components
directly in a sub-folder

To create custom components in a **subfolder** of the components folder, you must first create the subfolder

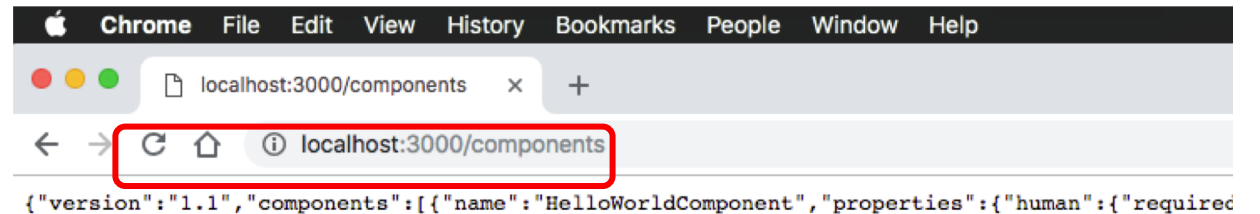


Run component service locally for testing in browser

- Use bots-node-sdk service command to start
 - Start from outside the project folder
- Default port is 3000
 - Can be changed if required
- Browser displays JSON message with all project components



```
Desktop — node /usr/local/bin/bo
[fnimphiu-orcl:helloworld101 fnimphiu$ cd ..
[fnimphiu-orcl:Desktop fnimphiu$ bots-node-sdk service helloworld101
-----
Component Service Ready (no auth):
http://localhost:3000/components
-----
helloworld101 => HelloWorldComponent
█
```



```
Chrome File Edit View History Bookmarks People Window Help
localhost:3000/components x +
localhost:3000/components
{"version":"1.1","components":[{"name":"HelloWorldComponent","properties":{"human":{"required
```

HelloWorldComponent.js

```
module.exports = {  
  metadata: () => ({  
    name: 'HelloWorldComp',  
    properties: {  
      human: { required: true, type: 'string' },  
    },  
    supportedActions: ['weekday', 'weekend']  
  }  
),  
  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();  
  }  
};
```

Component invocation name

Component properties

Action transitions

Function invoked at runtime

Callback function that must be called at the end

Topic agenda

- 1 Oracle Bots Node.js SDK
- 2 Getting started
- 3 Local container deployment**

Creating a deployment package

- npm pack
 - For deployment to local component container
 - Creates compressed deployment file
 - <folder name>-<version as in package.json>.tgz
 - E.g. helloworld101-1.0.0.tgz

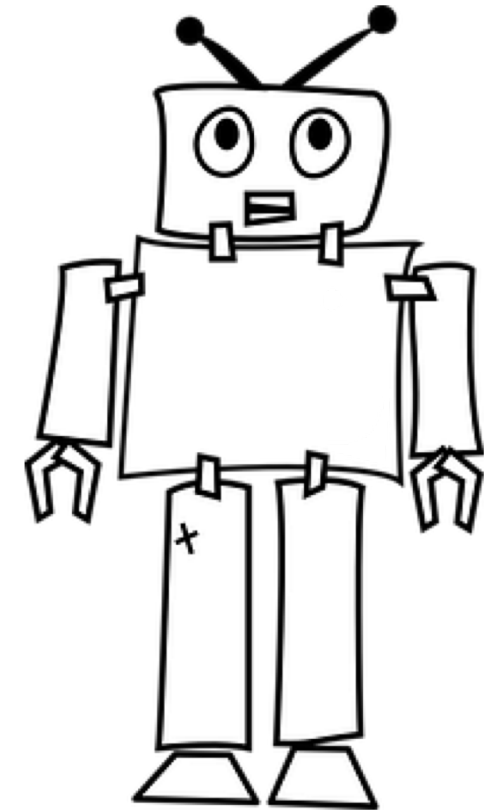
```
helloworld101 — -bash — 76x28
fnimphiu-orcl:helloworld101 fnimphiu$ npm pack

> helloworld101@1.0.0 prepack /Users/fnimphiu/Desktop/helloworld101
> npm run bots-node-sdk -- --pack --dry-run

> helloworld101@1.0.0 bots-node-sdk /Users/fnimphiu/Desktop/helloworld101
> bots-node-sdk "--pack" "--dry-run"

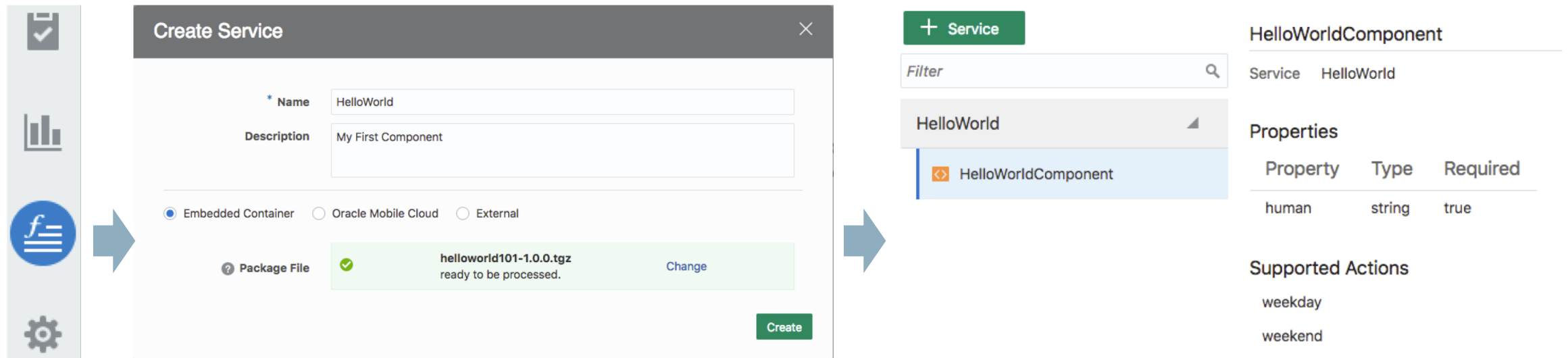
npm notice
npm notice 📦 helloworld101@1.0.0
npm notice === Tarball Contents ===
npm notice 605B package.json
npm notice 60B main.js
npm notice 1.9kB README.md
npm notice 740B components/HelloWorldComp.js
npm notice === Tarball Details ===
npm notice name: helloworld101
npm notice version: 1.0.0
npm notice filename: helloworld101-1.0.0.tgz
npm notice package size: 1.8 kB
npm notice unpacked size: 3.3 kB
npm notice shasum: 89e7bf2b945b17f42e8757dd860f839c019d9b07
npm notice integrity: sha512-azZxD+/wM1P/b[...]imSXgY/aGC10w==
npm notice total files: 4
npm notice
helloworld101-1.0.0.tgz
fnimphiu-orcl:helloworld101 fnimphiu$
```

The compressed project **does not contain the Oracle Bots Node.js SDK** files or other dependent Node modules. These are added in Oracle Digital Assistant as part of the **deployment process.**



Installing custom component service in local container

- Drag and Drop .tgz file into Create Service dialog
 - Ensure 'Embedded Container' is selected



The image shows a two-step process. On the left, the 'Create Service' dialog is open. The 'Name' field is 'HelloWorld' and the 'Description' is 'My First Component'. The 'Embedded Container' radio button is selected. A package file 'helloworld101-1.0.0.tgz' is shown as ready to be processed. On the right, the service 'HelloWorldComponent' is displayed in the console. It has a 'human' property of type 'string' which is required. Supported actions are 'weekday' and 'weekend'.

Create Service Dialog:

- Name: HelloWorld
- Description: My First Component
- Container Type: Embedded Container, Oracle Mobile Cloud, External
- Package File: helloworld101-1.0.0.tgz ready to be processed. [Change](#)
- [Create](#)

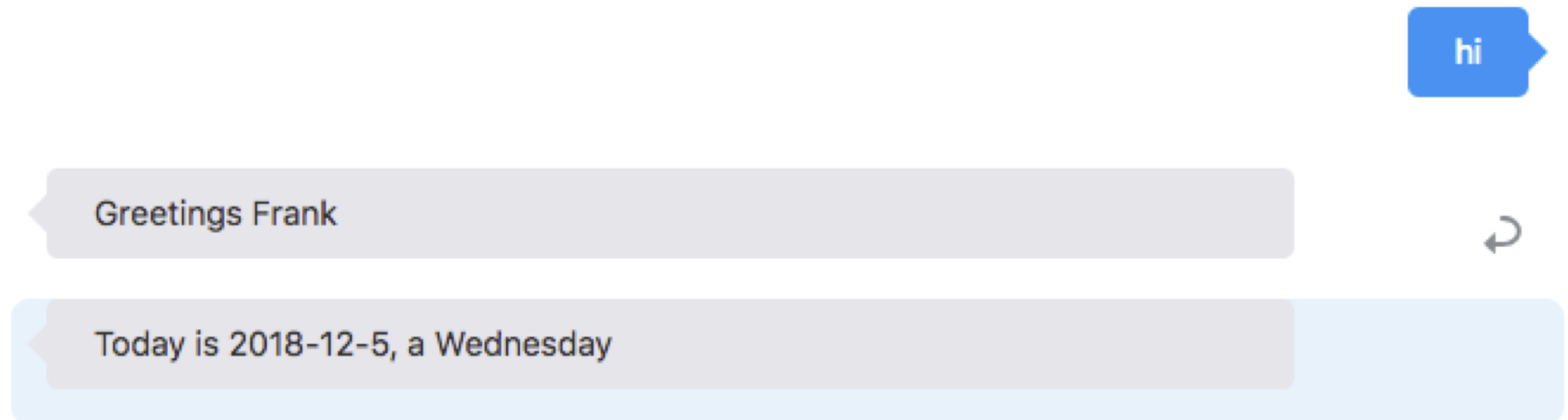
Service Configuration:

- Service: HelloWorldComponent
- Properties:

Property	Type	Required
human	string	true
- Supported Actions:
 - weekday
 - weekend

Dialog flow configuration & output at runtime

```
states:  
  askGreeting:  
    component: "HelloWorldComponent"  
    properties:  
      human: "Frank"  
    transitions:  
      return: "done"
```



Integrated Cloud

Applications & Platform Services

ORACLE®