

ORACLE®

Oracle Digital Assistant

The Complete Training

Custom Component Development



Image courtesy of pixabay.com

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

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.

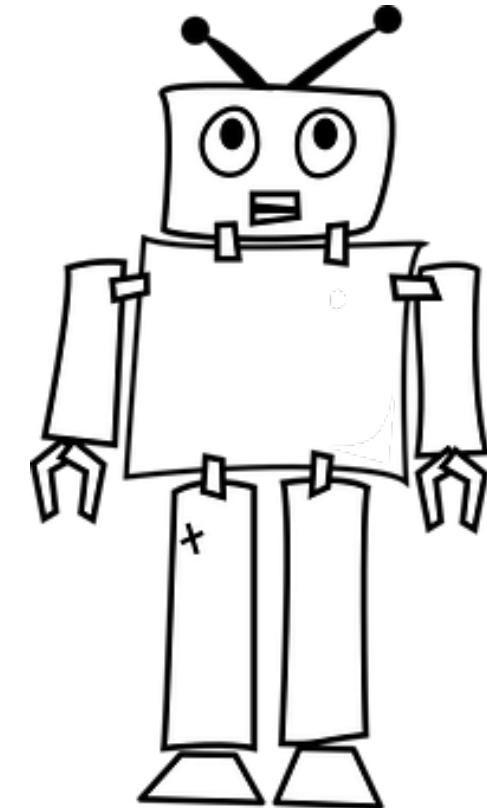
Topic agenda

- 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

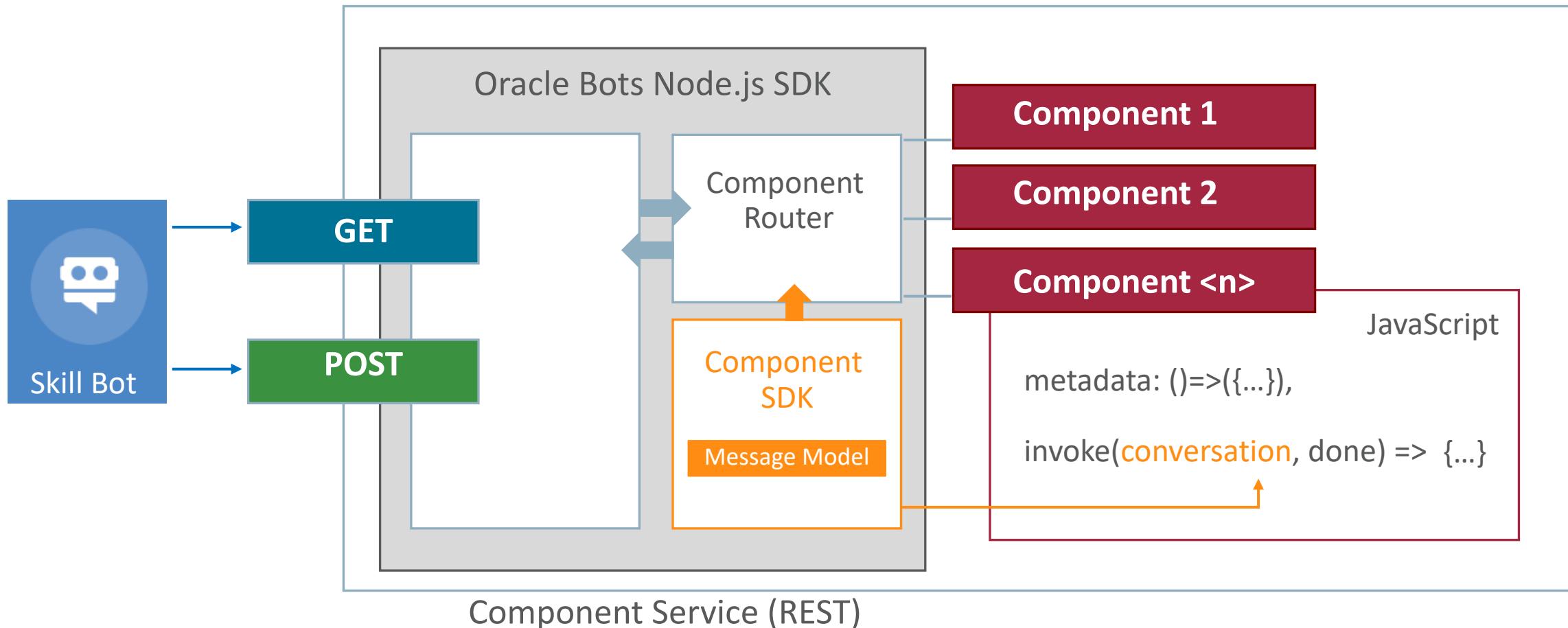
- Node Package Manager, Node version 8.11.4 or lower
 - <https://nodejs.org/en/download/>
- Install JavaScript IDE with support for Node.js debugging
 - Microsoft Visual Studio Code, IntelliJ WebStorm, etc.
- Access to Internet
 - If behind a proxy call
 - `npm config set proxy http://company-proxy:port`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

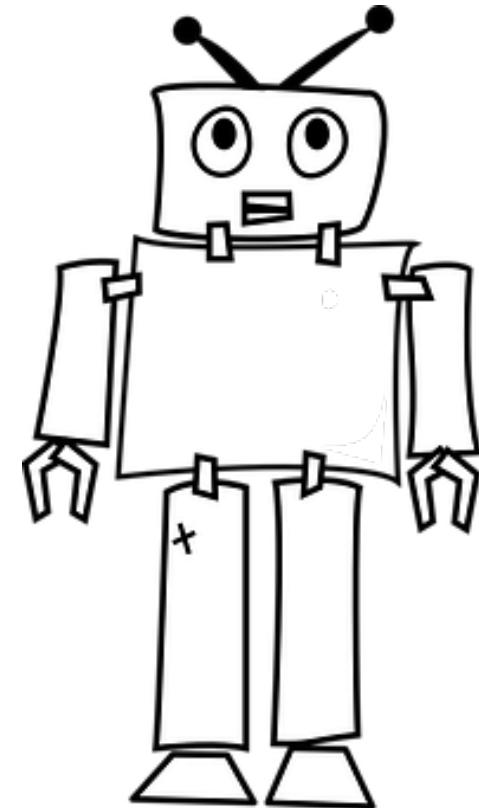
Custom component service architecture



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
```



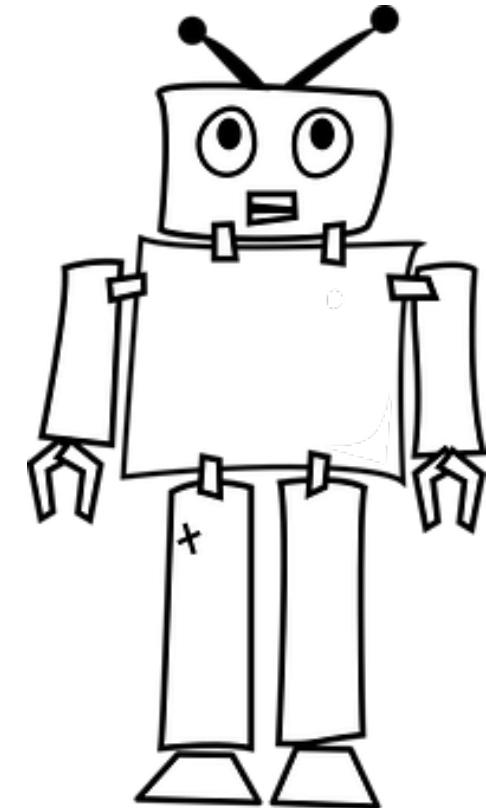
The MessageModel implementation
Is defined in

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

Topic agenda

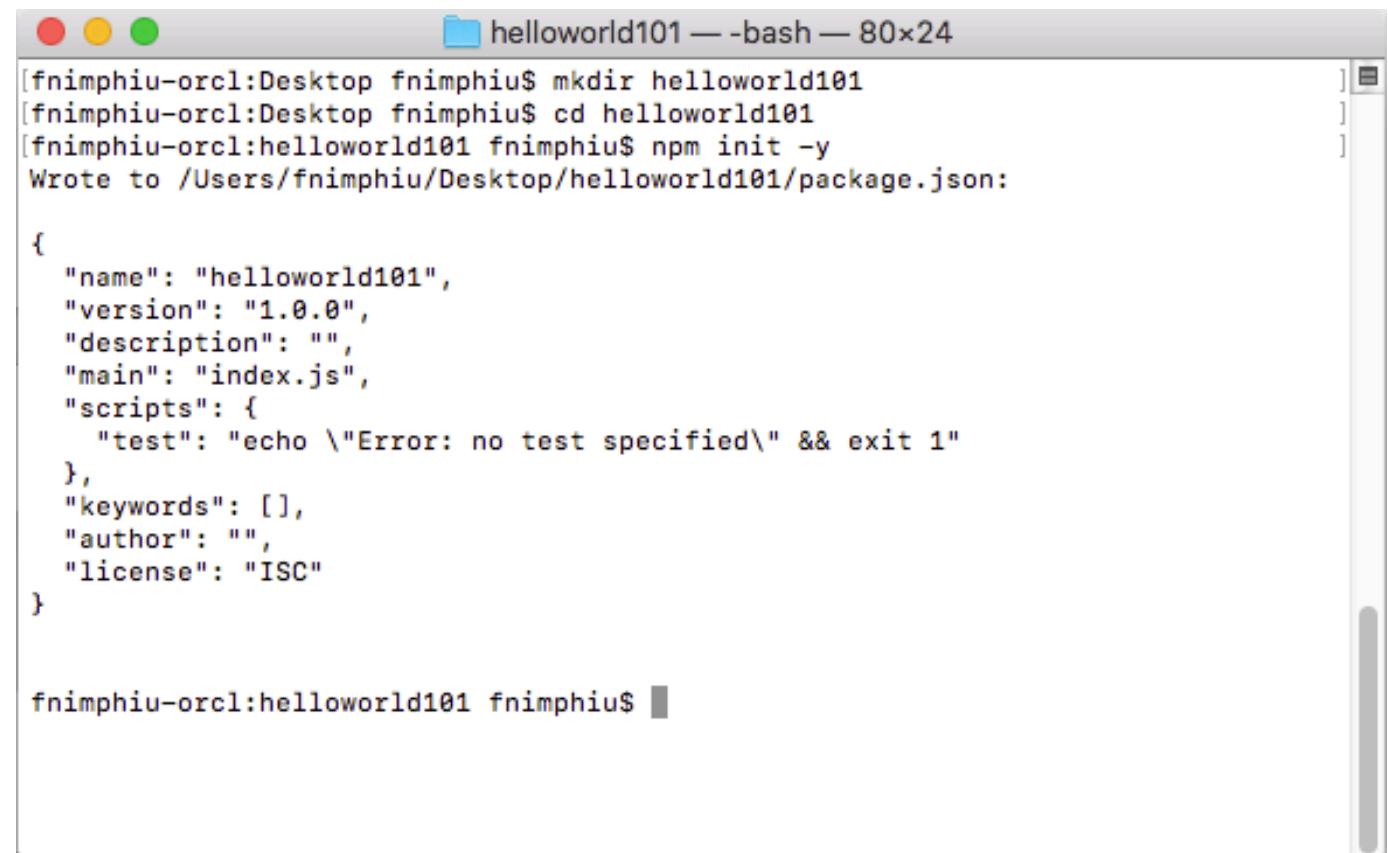
- 1 Oracle Bots Node.js SDK
- 2 Getting started
- 3 Local container deployment

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

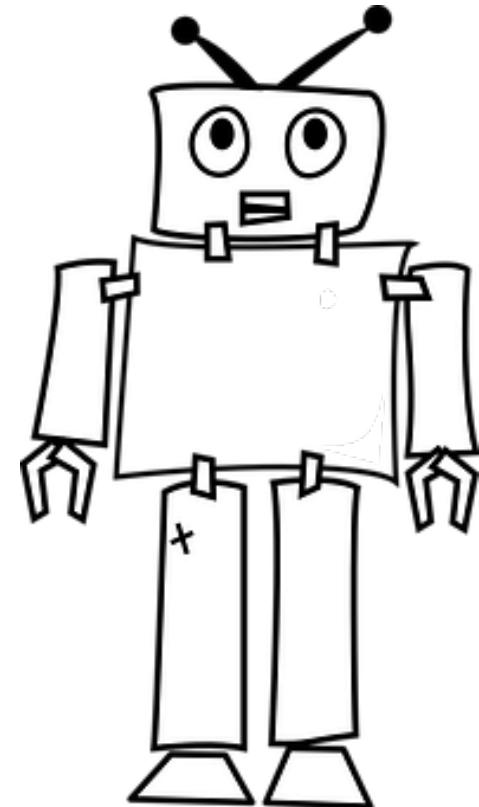


```
[fnimphiu-orcl:Desktop fnimphiu$ mkdir helloworld101
[fnimphiu-orcl:Desktop fnimphiu$ cd helloworld101
[fnimphiu-orcl:helloworld101 fnimphiu$ npm init -y
Wrote to /Users/fnimphiu/Desktop/helloworld101/package.json:

{
  "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 matchg core node modules
 - should not contain '.js' or 'node'

Create a custom component

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

The screenshot shows a terminal window titled "helloworld101 — bash — 96x28" and a file explorer window. In the terminal, the command "bots-node-sdk init --component-name HelloWorldComponent" is run, followed by a message indicating it will update the directory and create a component package. Below this, "Writing files..." and several file paths are listed. A red box highlights the command in the terminal. In the file explorer, the directory "helloworld101" is shown with subfolders "components", "node_modules", "spec", and ".gitignore". A file named "HelloWorldComponent.js" is visible in the "components" folder, also highlighted with a red box. An arrow points from the terminal output to the file in the file explorer.

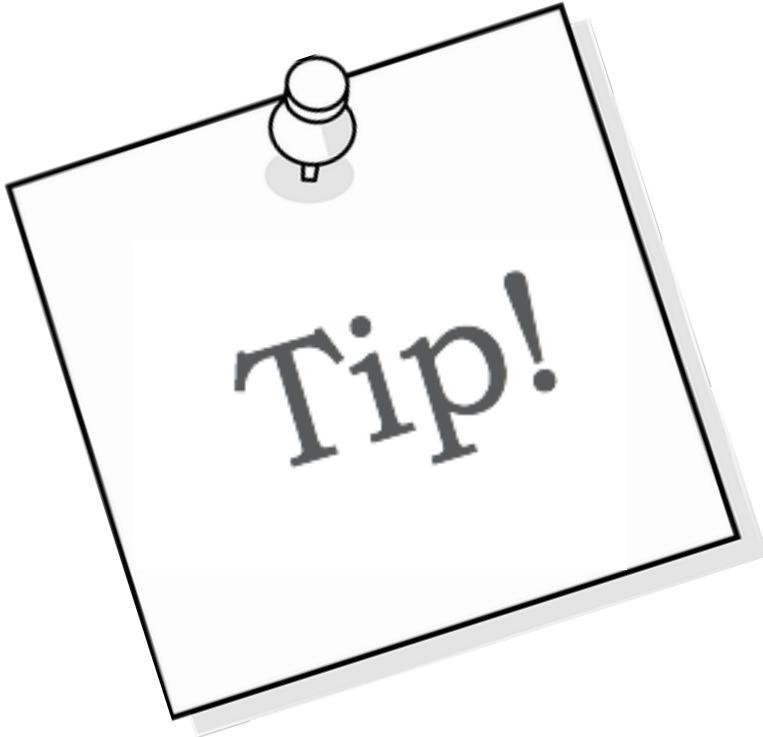
```
fnimphiu-orcl:helloworld101 fnimphiu$ bots-node-sdk init --component-name HelloWorldComponent
-----
Will update contents in directory 'helloworld101'
Create component package in directory 'helloworld101'...
-----
Writing files...
Writing file: /Users/fnimphiu/Desktop/helloworld101/components/HelloWorldComponent.js
Writing file: /Users/fnimphiu/Desktop/helloworld101/components/HelloWorldComponent.spec.js
Writing file: /Users/fnimphiu/Desktop/helloworld101/components/HelloWorldComponent.test.js
```

helloworld101

- components
- node_modules
- spec
- .gitignore

HelloWorldComponent.js

- 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



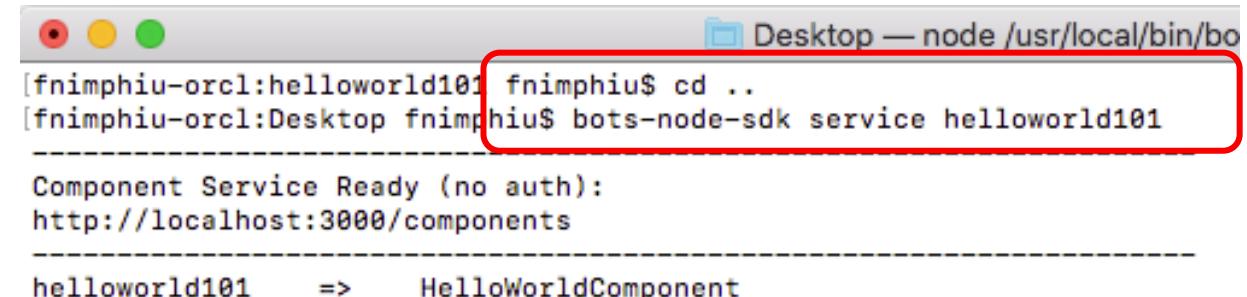
You can use

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

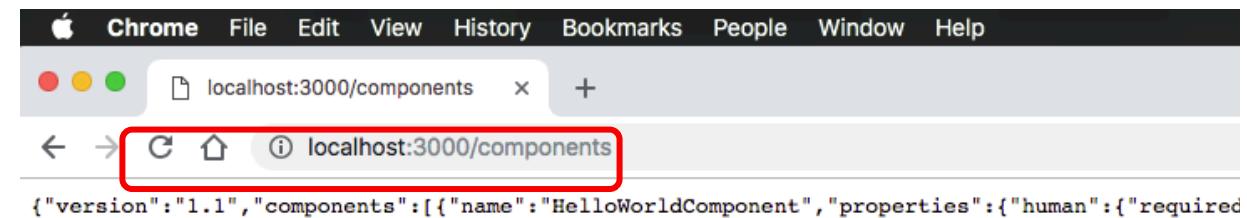
to create additional components
directly in a sub-folder

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



```
[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
```



HelloWorldComponent.js

```
module.exports = {
  metadata: () => ({
    name: 'HelloWorldComp',           → Component invocation name
    properties: {
      human: { required: true, type: 'string' },
    },
    supportedActions: ['weekday', 'weekend'] → Action transitions
  }),
  invoke: (conversation, done) => {           → Function invoked at runtime
    // 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');

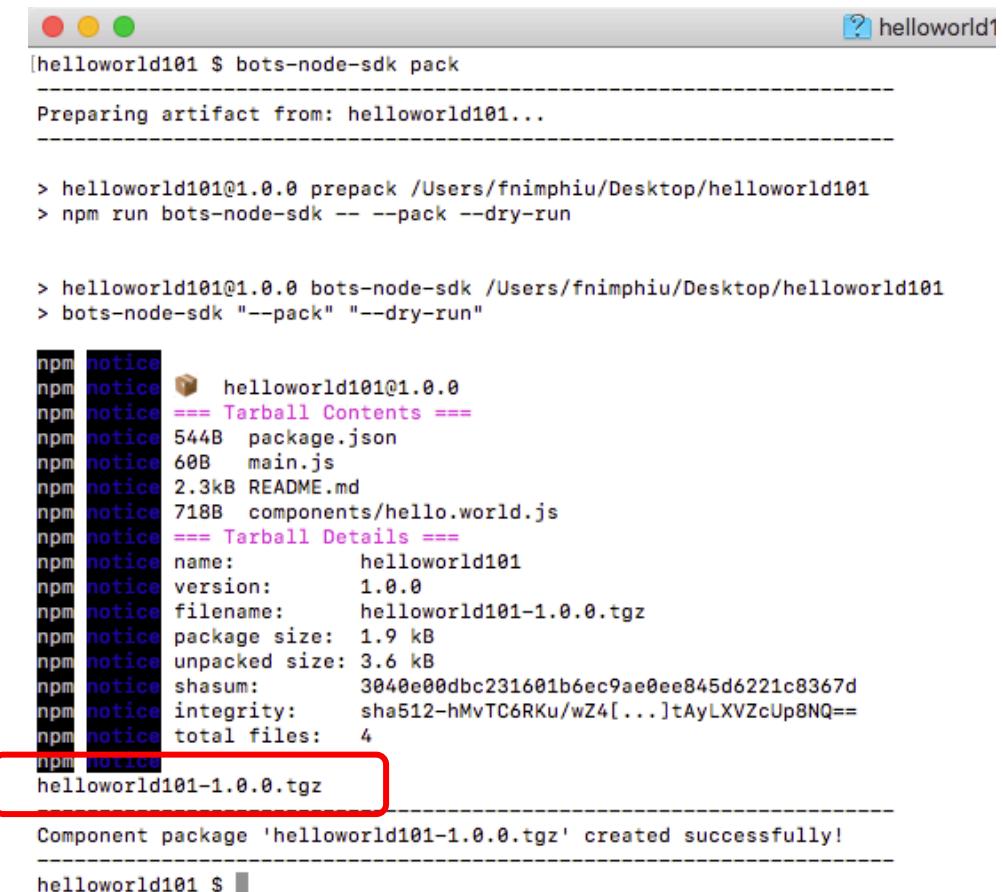
    done();                                     → Callback function that must be called at the end
  }
};
```

Topic agenda

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

Creating a deployment package

- bots-node-sdk 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
 - Adds all dependent node modules



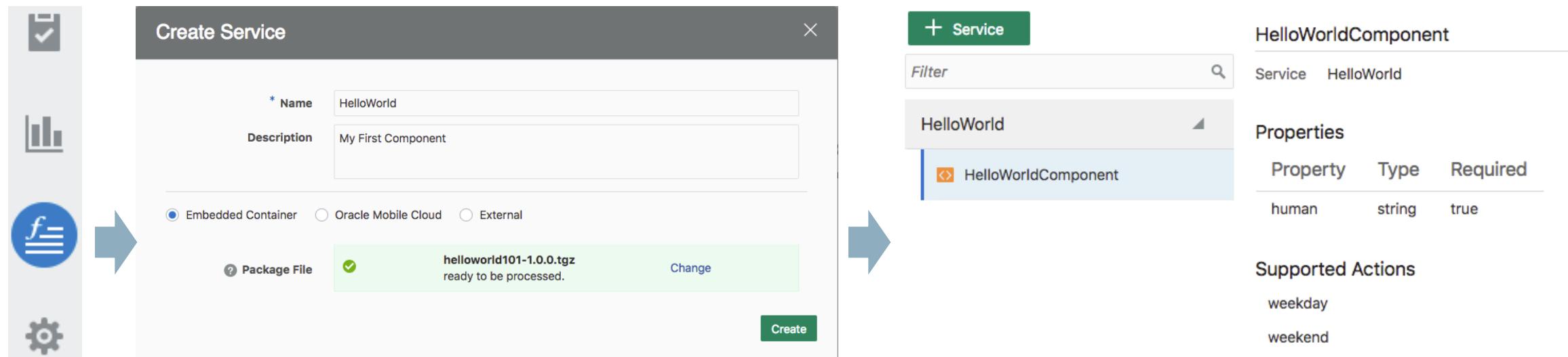
```
[helloworld101 $ bots-node-sdk pack
-----
Preparing artifact from: helloworld101...
-----
> helloworld101@1.0.0 prepack /Users/fnimphiu/Desktop/helloworld101
> npm run bots-node-sdk -- --pack --dry-run

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

npm notice   helloworld101@1.0.0
npm notice === Tarball Contents ===
npm notice 544B package.json
npm notice 60B main.js
npm notice 2.3kB README.md
npm notice 718B components/hello.world.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: 1.9 kB
npm notice unpacked size: 3.6 kB
npm notice shasum: 3040e000dbc231601b6ec9ae0ee845d6221c8367d
npm notice integrity: sha512-hMvTC6RKu/wZ4[...]tAyLXVZcUp8NQ==
npm notice total files: 4
npm notice
helloworld101-1.0.0.tgz
-----
Component package 'helloworld101-1.0.0.tgz' created successfully!
-----
helloworld101 $
```

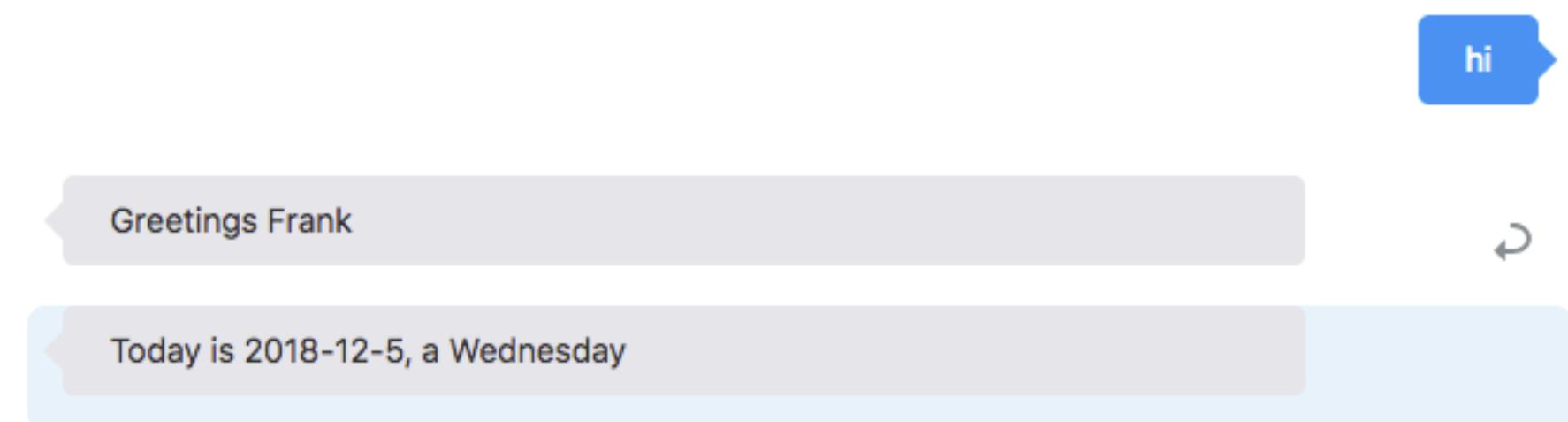
Installing custom component service in local container

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



Dialog flow configuration & output at runtime

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



Integrated Cloud Applications & Platform Services

ORACLE®