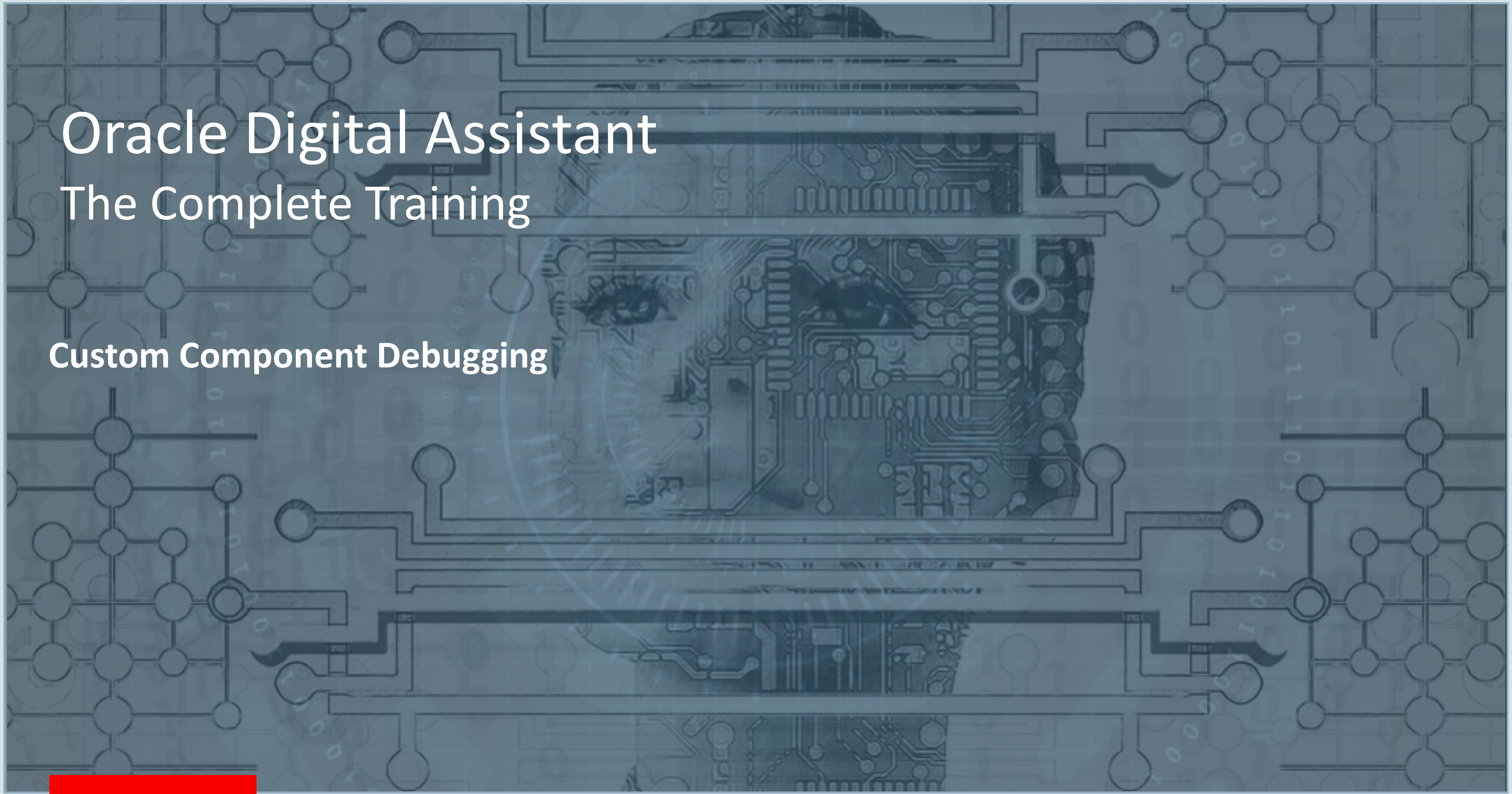


Oracle Digital Assistant

The Complete Training

Custom Component Debugging

Safe Harbor Statement

Topic agenda

1. Debugging vs. logging
2. Debugging architecture
3. Debugging practices
4. Debugging with MS Visual Studio Code

Topic agenda

1 Debugging vs. logging

2 Debugging architecture

3 Debugging practices

4 Debugging with MS Visual Studio Code

Logging

- Logging is good to find problems that occur at runtime
  - Digital Assistant version 19.1.3 and later provide a diagnostic panel for displaying logs for locally deployed custom components
  - Mobile Hub and 3rd party Node containers have their own consoles for displaying logs
- Logging is not efficient enough for finding problems at design time
  - Not all problems are exceptions
    - Custom component doesn't render a response
    - Values returned by a custom component don't show in skill
    - Code logic is not getting executed
    - Broken dialog flow navigation after custom component was added

Debugging

- Monitors bot-component interactions
  - Access to bot message payload
  - Insights to variable states
  - Steps through code execution
- Use your JavaScript IDE of choice
  - E.g. MS Visual Studio Code, JetBrains Webstorm
- Run custom component service from your local computer
  - Connect from Oracle Digital Assistant skill
- If it works, then it continues working after deployment

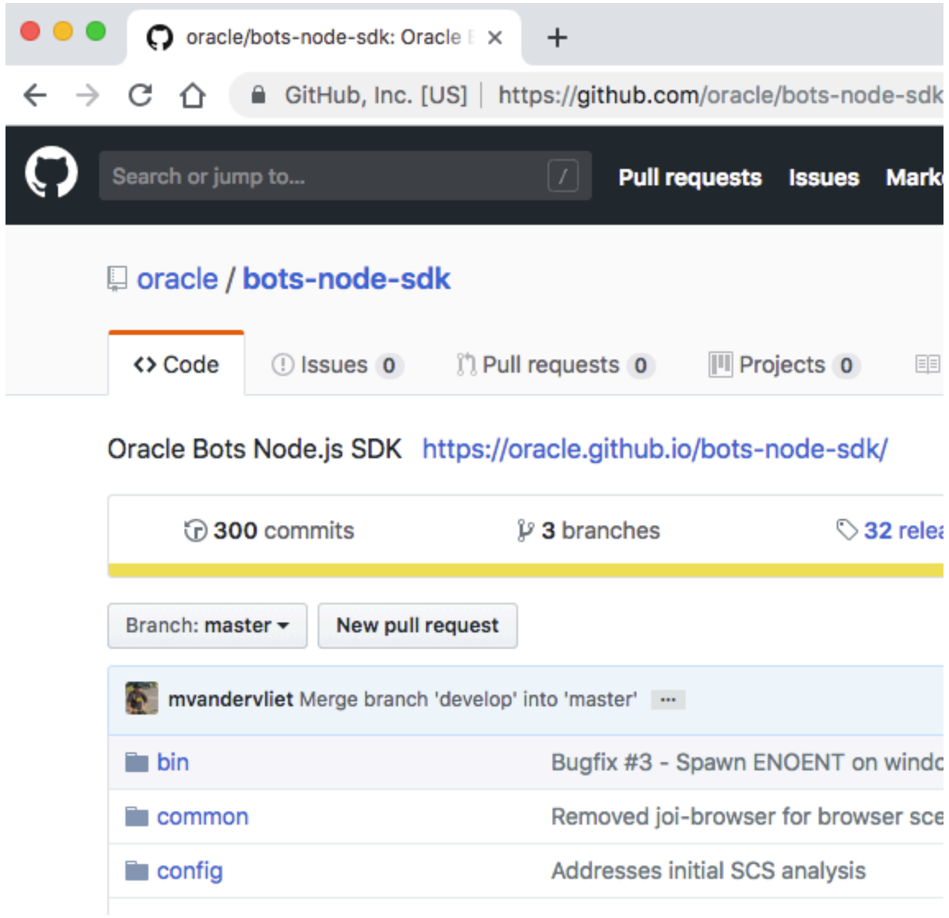
Before you can debug custom component services

- Create local environment
  - Local runtime for custom component REST service
  - Install Oracle Bots Node.js
    - Command line to create custom component services
    - Includes local runtime
- Make local machine accessible from the Internet
  - Oracle Digital Assistant needs to access local component service runtime URL
- Have JavaScript IDE installed that supports Node debugging

Topic agenda

About Oracle Bots Node.js SDK

- Provides local environment for
  - Component development
  - Component debugging
  - Packaging for deployment
- NPM installable
- Command line to
  - Create custom component service project
  - Create custom components
- Adds bots custom component SDK



Oracle Bots Node.js SDK https://oracle.github.io/bots-node-sdk/

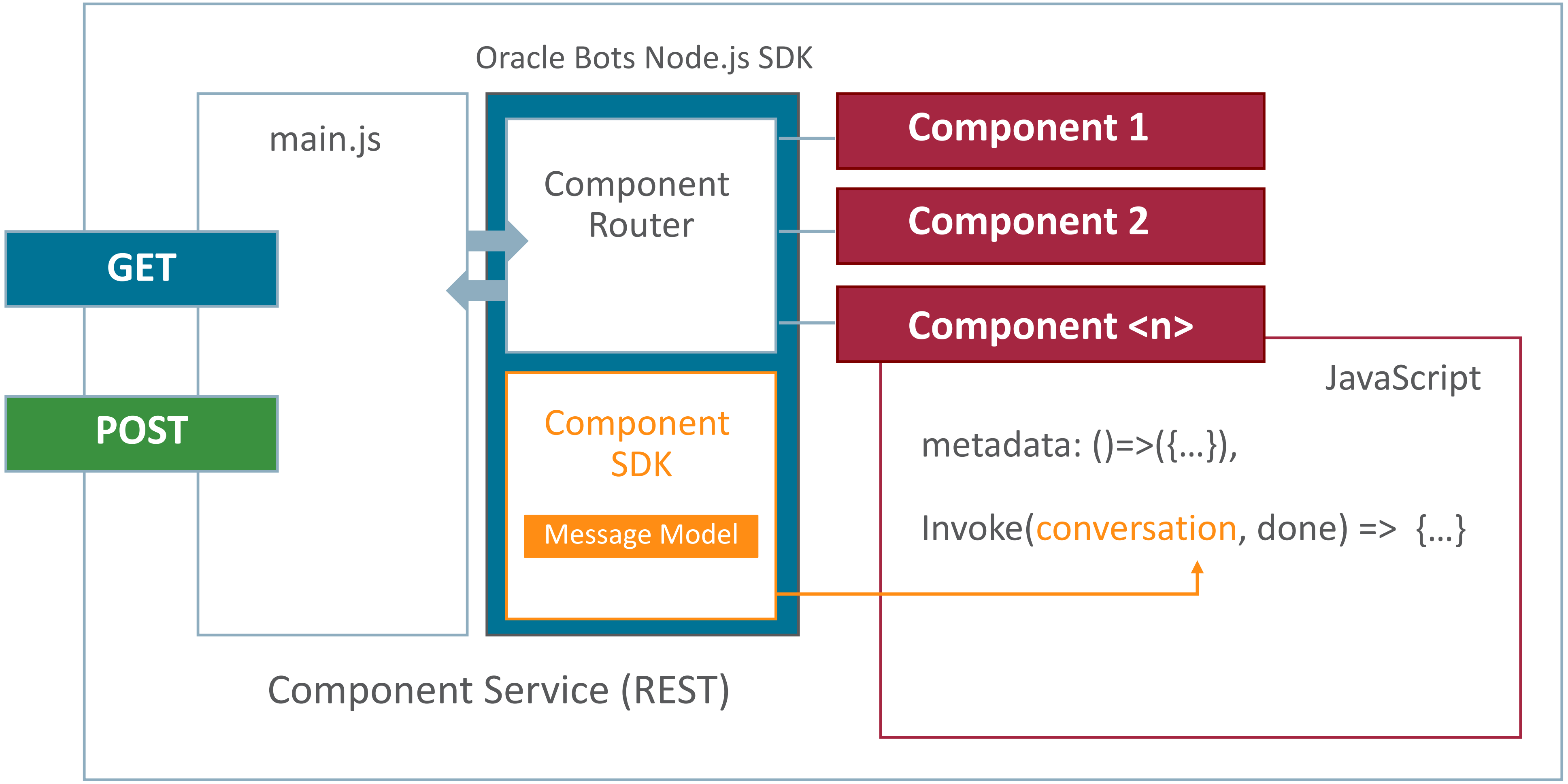
300 commits 3 branches 32 relea

Branch: master New pull request

mvandervliet Merge branch 'develop' into 'master'

bin	Bugfix #3 - Spawn ENOENT on windo
common	Removed joi-browser for browser sce
config	Addresses initial SCS analysis

Oracle Bots Node.js SDK component service architecture



For debugging, you **local machine must be accessible from** Oracle Digital Assistant running on **the Internet**

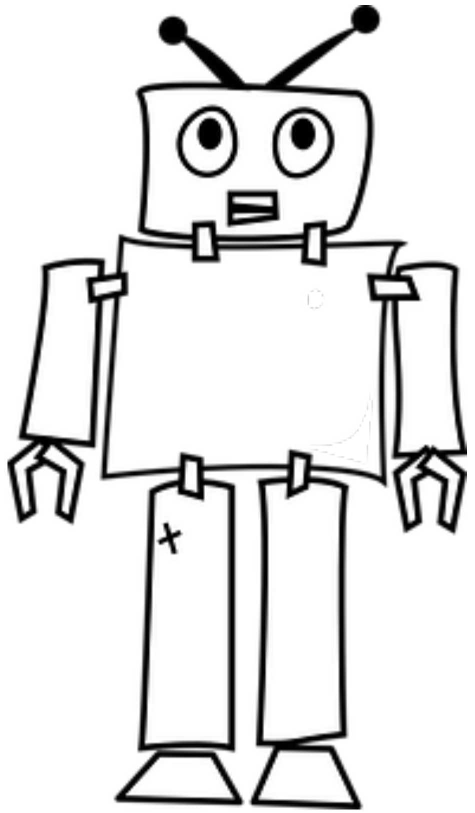
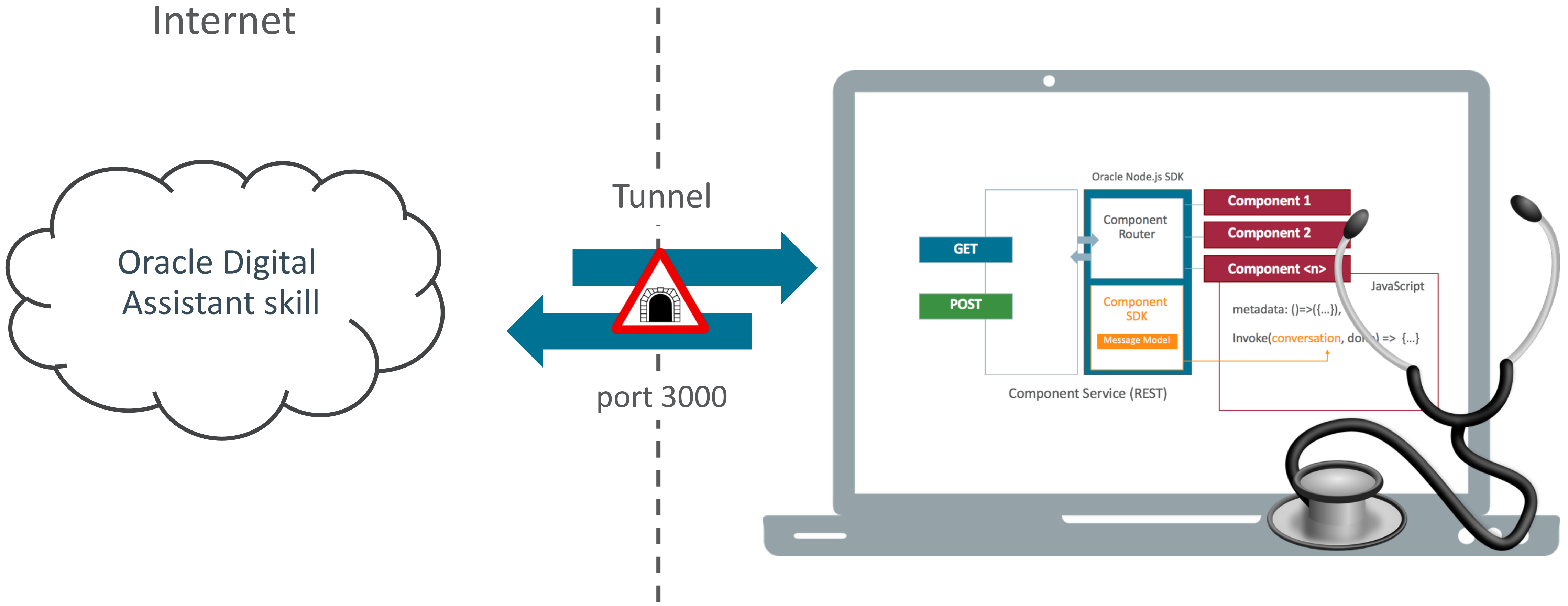


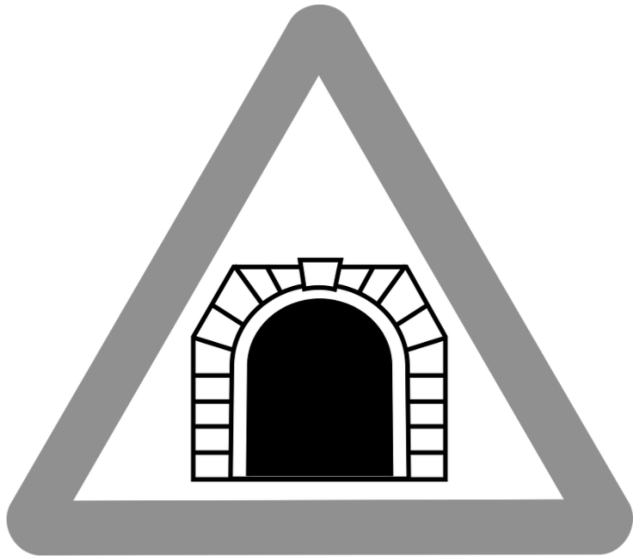
Image courtesy of pixabay.com

Debugging Architecture



Popular tunneling options

- ngrok
  - Well known
  - Doesn't require account
  - https://ngrok.com/download
- localtunnel
  - GitHub
  - Doesn't require account
  - https://github.com/localtunnel/localtunnel



ORACLE

Hints and tips

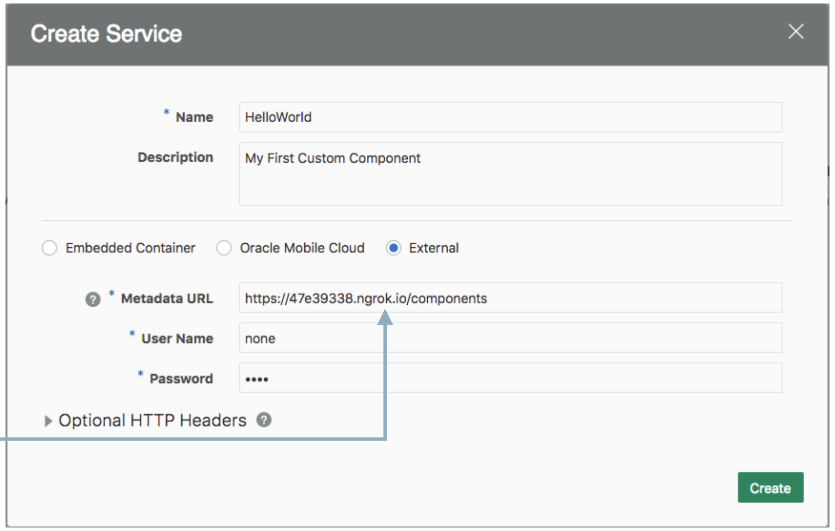
- Generally speaking, ngrok works on Windows and Mac
  - Oracle Windows images (OBI) don't allow the use of ngrok
  - When ngrok doesn't work, use "localtunnel" instead instead
- Ensure you have direct Internet access
  - Public or home network
  - Mobile hotspot
- If you are behind a proxy
  - Get the external IP address of your proxy (http://www.whatismyproxy.com/).
  - In the terminal window where you'll be using ngrok enter
    - export https_proxy=http://<external ip>:80
    - export http_proxy=http://<external ip>:80

ORACLE

Topic agenda

How-to debug custom component services

- Run Node server in Oracle Bots Node.js SDK
- Configure tunnel to expose port 3000
- Register component service 'External' service
  - Local runtime is an 'External' deployment
  - https://tunnel-url/components
- Connect JavaScript IDE debugger to local node server process
- Set breakpoints and use conversation tester in skill to start the debug session


Create Service

* Name	HelloWorld
Description	My First Custom Component

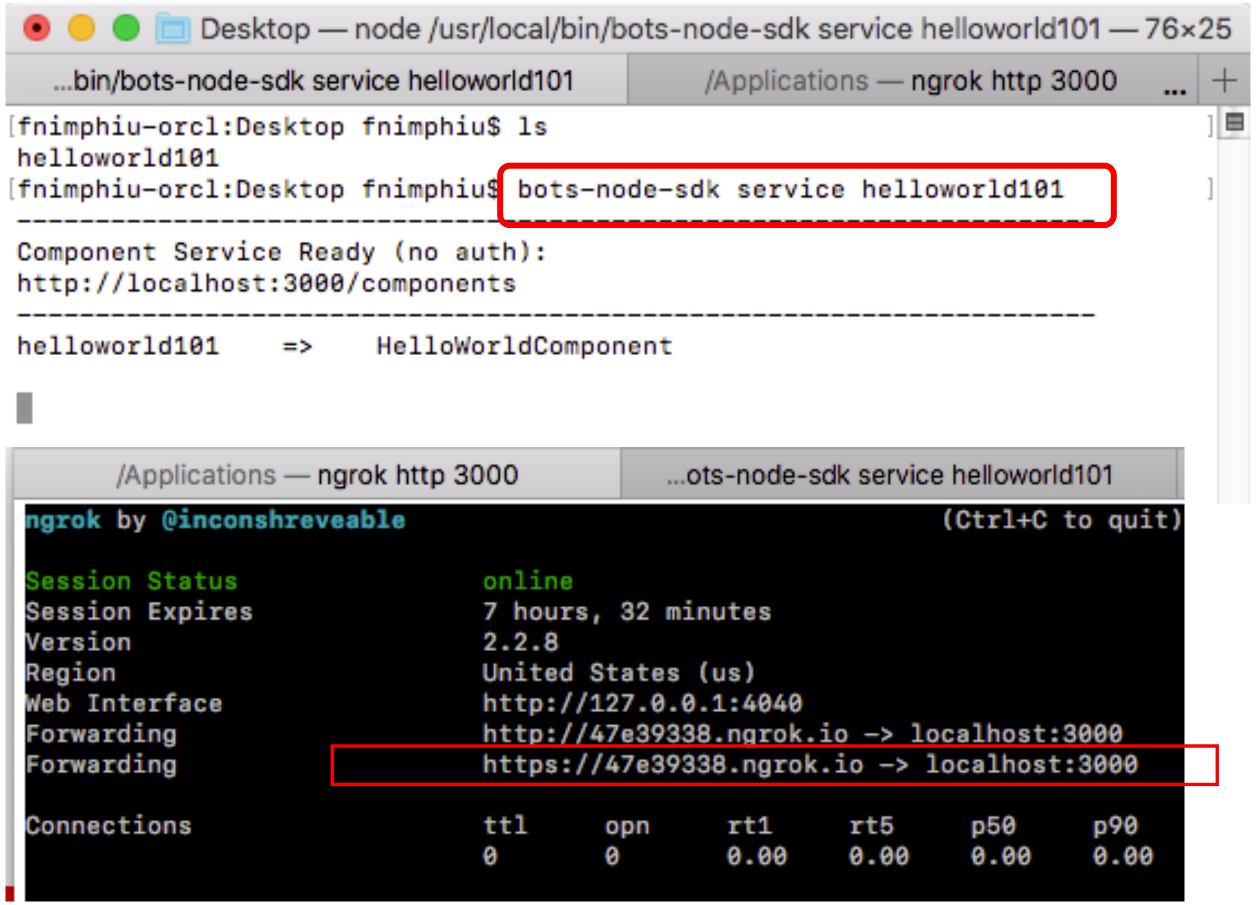
( ) Embedded Container ( ) Oracle Mobile Cloud [✓ External]

* Metadata URL	https://47e39338.ngrok.io/components
* User Name	none
* Password	••••

Starting the Node server on the local machine

- On local machine, start component service from parent folder
  - E.g. bots-node-sdk service helloworld101

- Open tunnel for port 3000
  - E.g. ngrok


Desktop — node /usr/local/bin/bots-node-sdk service helloworld101 — 76×25
...bin/bots-node-sdk service helloworld101
/Applications — ngrok http 3000

```
fnimphiu-orcl:Desktop fnimphiu$ ls
helloworld101
fnimphiu-orcl:Desktop fnimphiu$ bots-node-sdk service helloworld101
-----------------------------------------------------------------------
Component Service Ready (no auth):
http://localhost:3000/components
-----------------------------------------------------------------------
helloworld101     =>     HelloWorldComponent
```

/Applications — ngrok http 3000
...ots-node-sdk service helloworld101

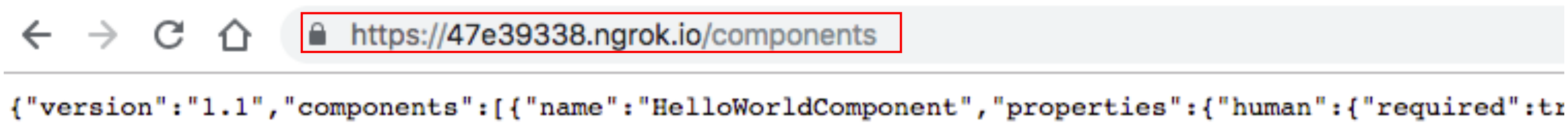
```
ngrok by @inconshreveable                                  (Ctrl+C to quit)

Session Status                online
Session Expires               7 hours, 32 minutes
Version                       2.2.8
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://47e39338.ngrok.io -> localhost:3000
Forwarding                    https://47e39338.ngrok.io -> localhost:3000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

Testing the custom component service access in browser

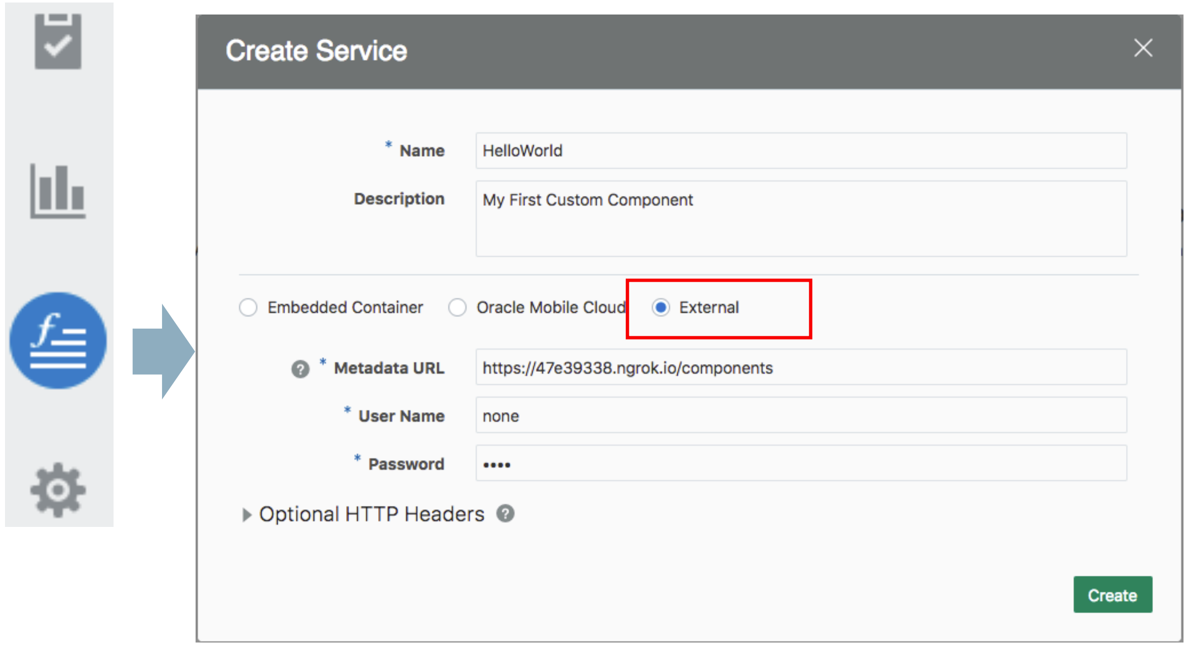
- Type https://<tunnel URL>/components
  - Displays list of components hosted by component service
- Be aware
  - Tunnel URL changes with each re-start of the tunnel



https://47e39338.ngrok.io/components

```
{"version":"1.1","components":[{"name":"HelloWorldComponent","properties":{"human":{"required":tr
```

Register custom component service in skill



Create Service

* Name: HelloWorld

Description: My First Custom Component

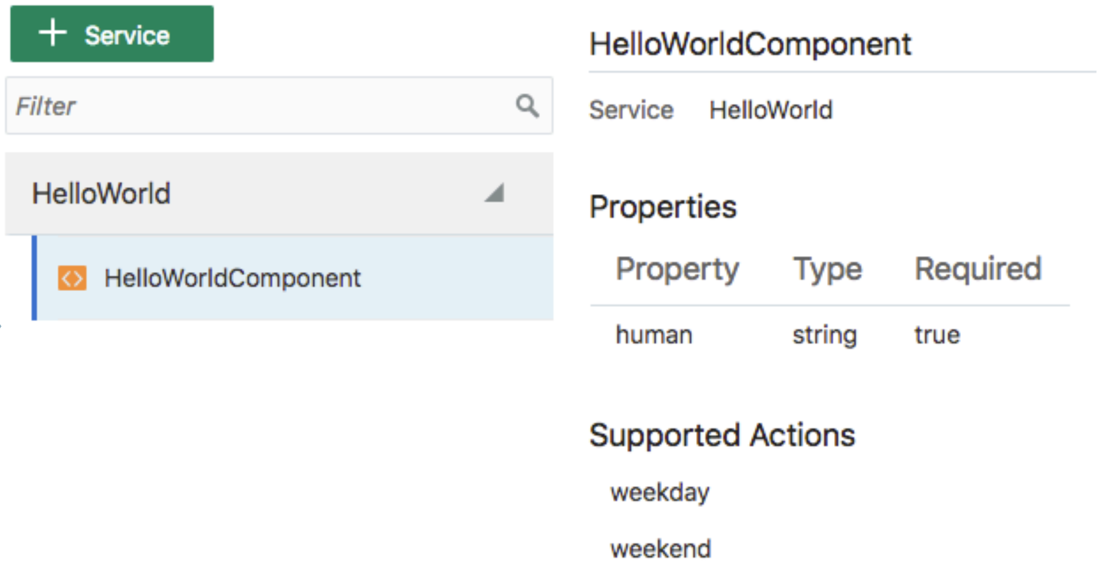
Embedded Container Oracle Mobile Cloud [✓ External]

* Metadata URL: https://47e39338.ngrok.io/components

* User Name: none

* Password: ••••

Optional HTTP Headers



HelloWorldComponent

Service HelloWorld

Properties

Property	Type	Required
human	string	true

Supported Actions

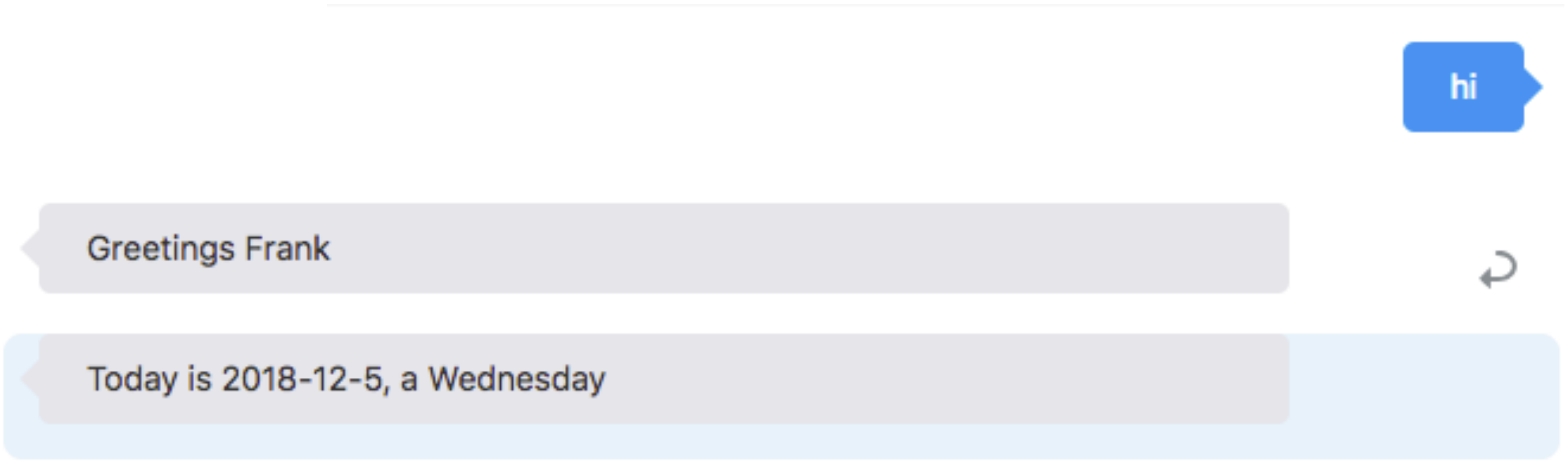
weekday

weekend

- Configure custom component service using ngrok or local tunnel URL
  - https://47e39338.ngrok.io/components
- No authentication required (just put something into the username and password field)

Dialog Flow configuration & runtime output

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


hi

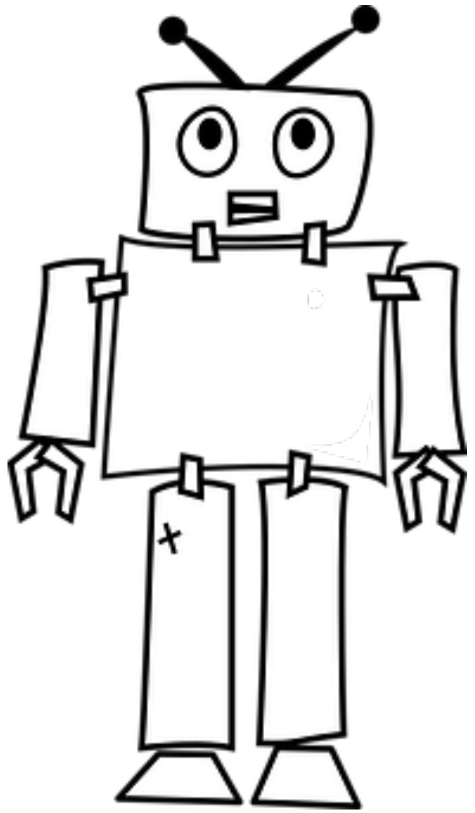
Greetings Frank

Today is 2018-12-5, a Wednesday

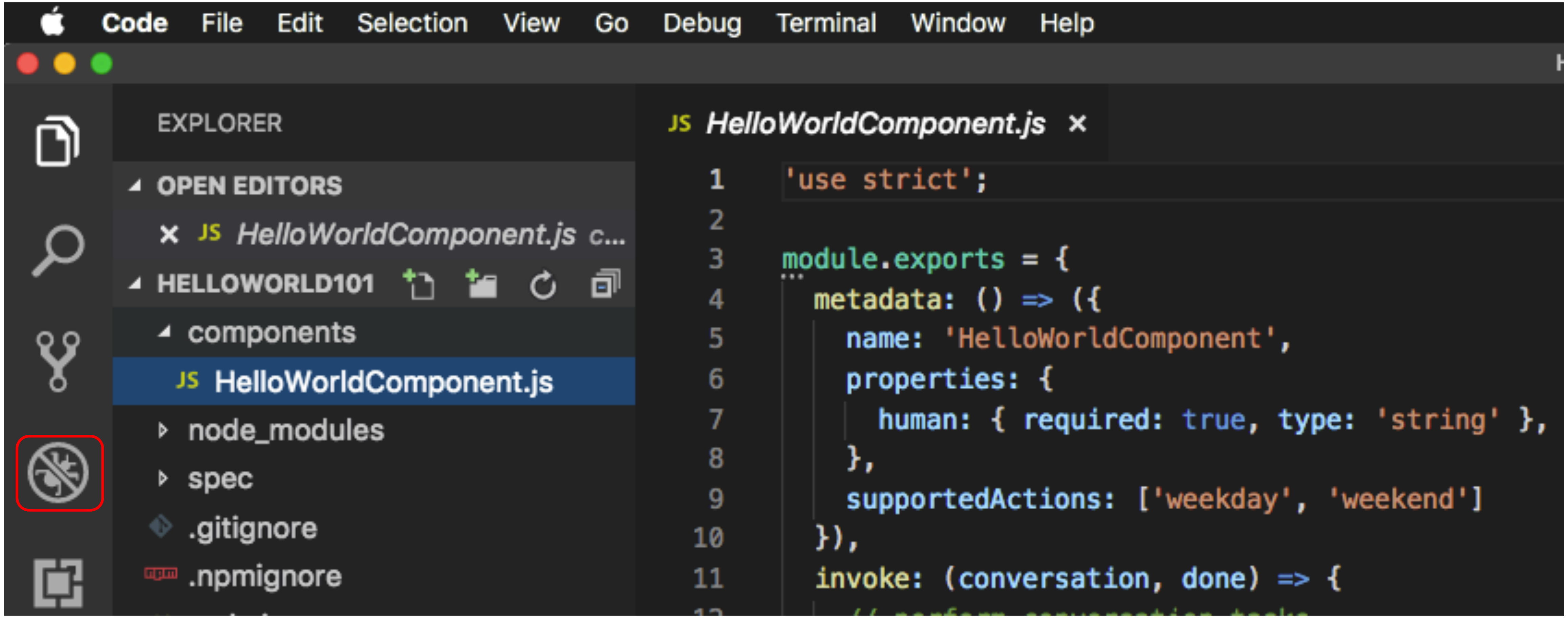
Topic agenda

ORACLE

There is **no dependency to a specific JavaScript IDE**. The IDE must be able to debug local Node process.



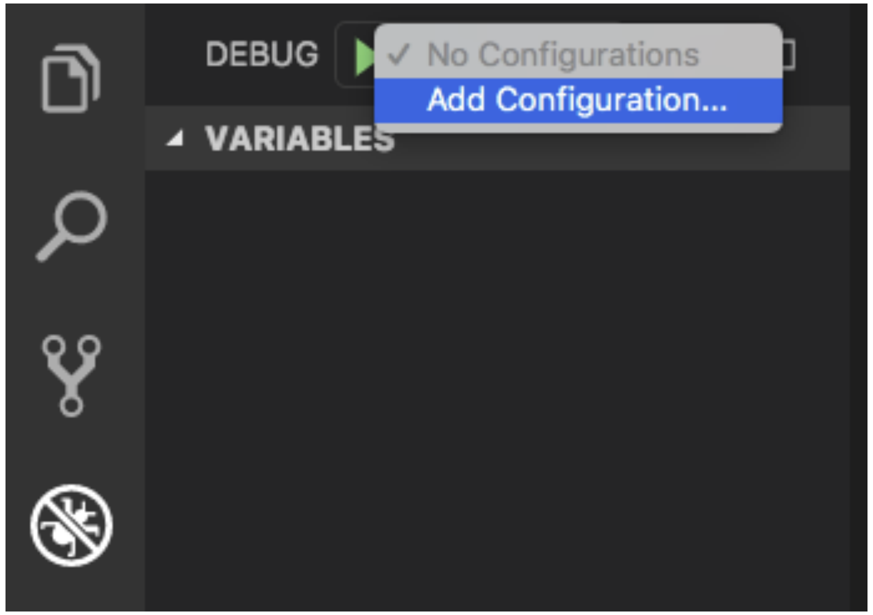
Open project in IDE and select debug option

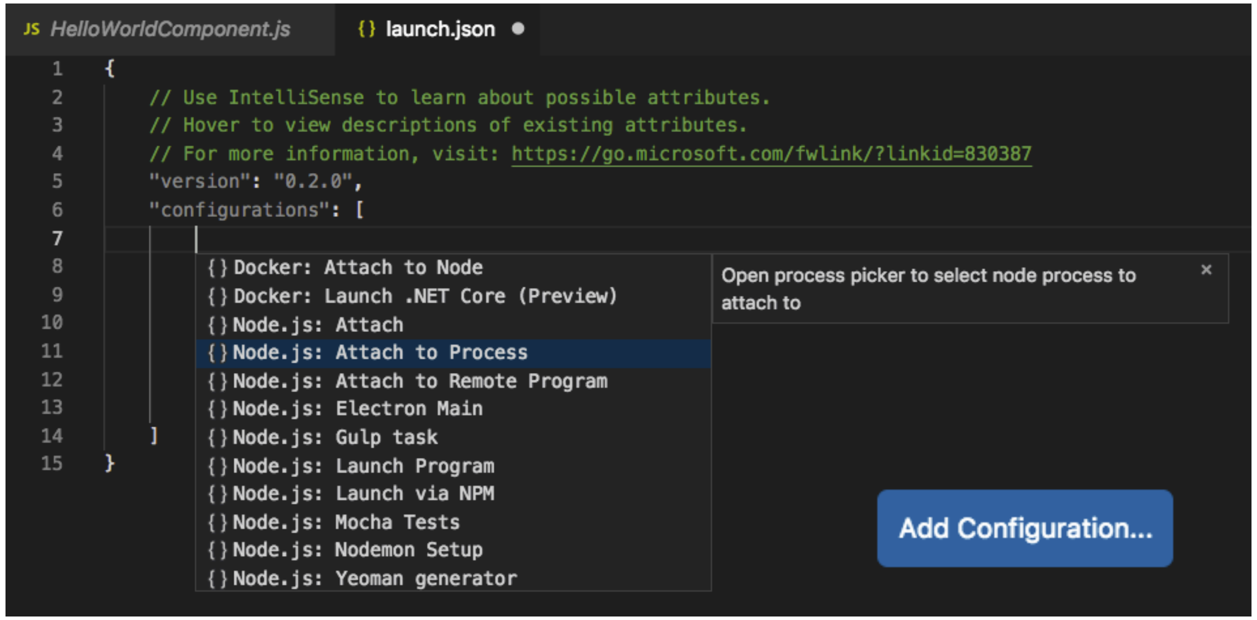


```
'use strict';

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

Create debugging configuration





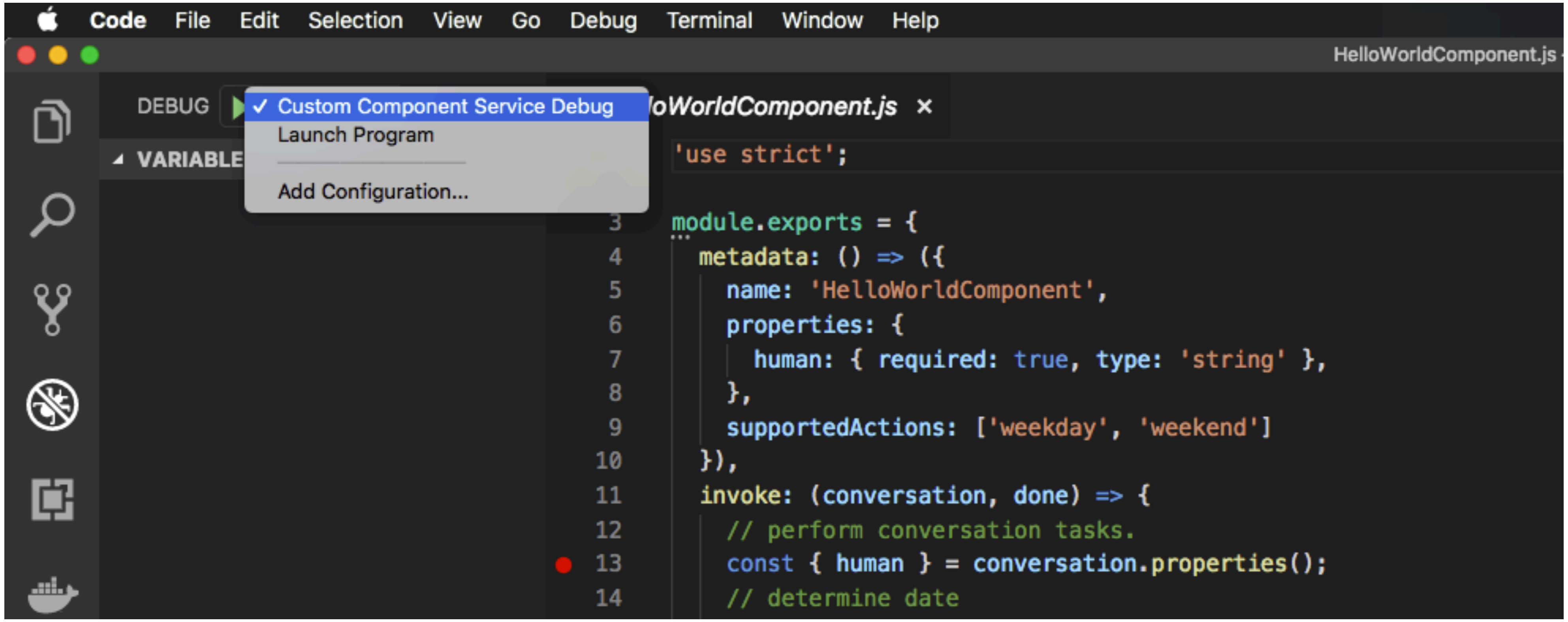
```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        |
    ]
}
```

```
{} Docker: Attach to Node
{} Docker: Launch .NET Core (Preview)
{} Node.js: Attach
{} Node.js: Attach to Process
{} Node.js: Attach to Remote Program
{} Node.js: Electron Main
{} Node.js: Gulp task
{} Node.js: Launch Program
{} Node.js: Launch via NPM
{} Node.js: Mocha Tests
{} Node.js: Nodemon Setup
{} Node.js: Yeoman generator
```

Open process picker to select node process to attach to

- Add a new debug configuration
  - Choose "Attach to Process" from "Add Configuration" button list
- Optionally, provide a custom name for the debug configuration

Set breakpoint and start debugging session

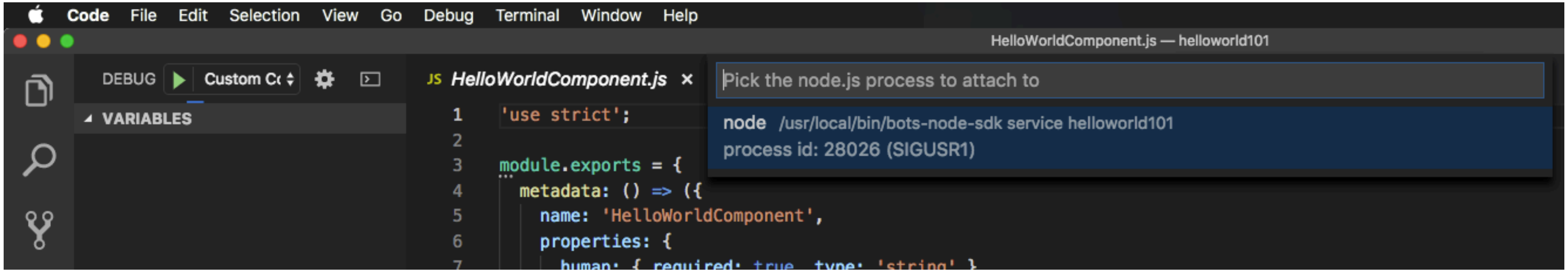

Code File Edit Selection View Go Debug Terminal Window Help
HelloWorldComponent.js
DEBUG
✓ Custom Component Service Debug
Launch Program
Add Configuration...
VARIABLE
loWorldComponent.js ×

```
'use strict';

module.exports = {
  metadata: () => ({
    name: 'HelloWorldComponent',
    properties: {
      human: { required: true, type: 'string' },
    },
    supportedActions: ['weekday', 'weekend']
  }),
  invoke: (conversation, done) => {
    // perform conversation tasks.
    const { human } = conversation.properties();
    // determine date
```

- Set break-points in the *invoke* function
- Select your custom debug configuration

Select process


Code File Edit Selection View Go Debug Terminal Window Help
HelloWorldComponent.js — helloworld101
DEBUG
Custom Cc
VARIABLES
HelloWorldComponent.js
Pick the node.js process to attach to

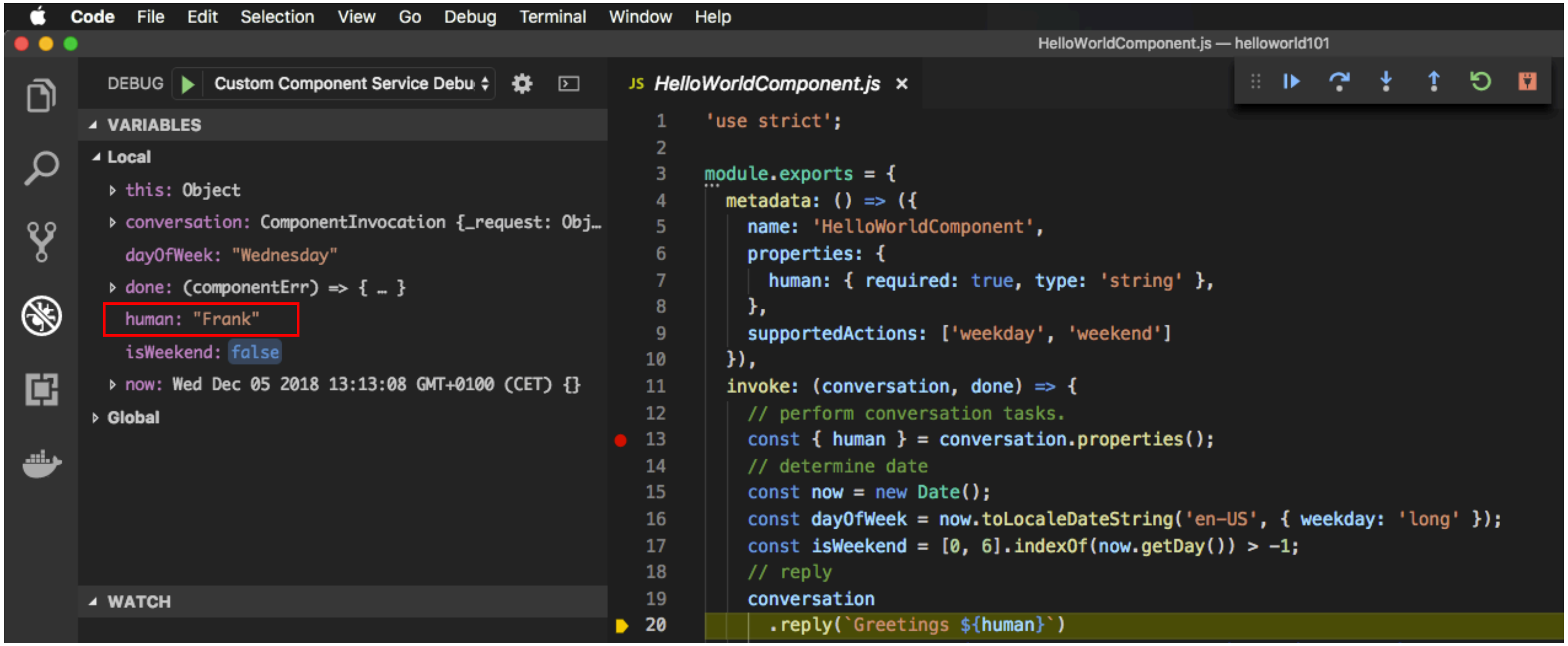
node /usr/local/bin/bots-node-sdk service helloworld101
process id: 28026 (SIGUSR1)

```
'use strict';

module.exports = {
  metadata: () => ({
    name: 'HelloWorldComponent',
    properties: {
      human: { required: true, type: 'string' }
```

- Press the green run icon
- Select the Node process from the list

Debug component service by running skill bot tester


Code File Edit Selection View Go Debug Terminal Window Help
HelloWorldComponent.js — helloworld101
DEBUG
Custom Component Service Debu
HelloWorldComponent.js
VARIABLES
WATCH

```
Local
  this: Object
  conversation: ComponentInvocation {_request: Obj…
  dayOfWeek: "Wednesday"
  done: (componentErr) => { … }
  human: "Frank"
  isWeekend: false
  now: Wed Dec 05 2018 13:13:08 GMT+0100 (CET) {}
Global
```

```
'use strict';

module.exports = {
  metadata: () => ({
    name: 'HelloWorldComponent',
    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}`)
```

Rerun the '**bots-node-sdk service**' command each time you change the component source code in a debug session for changes to take affect.

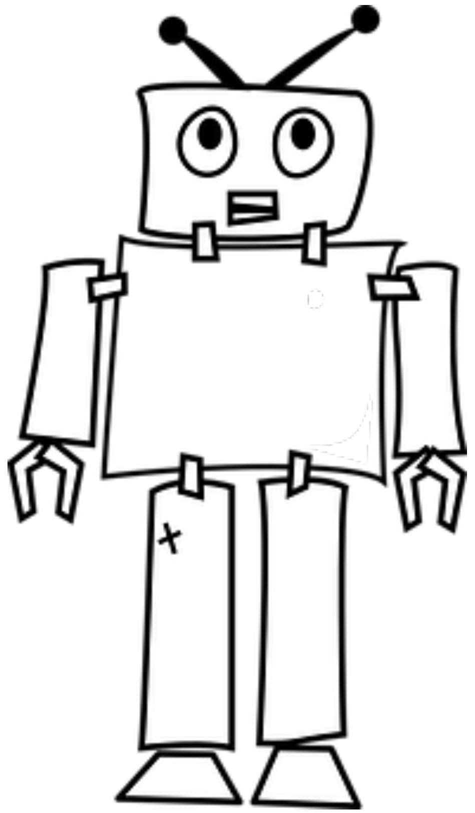
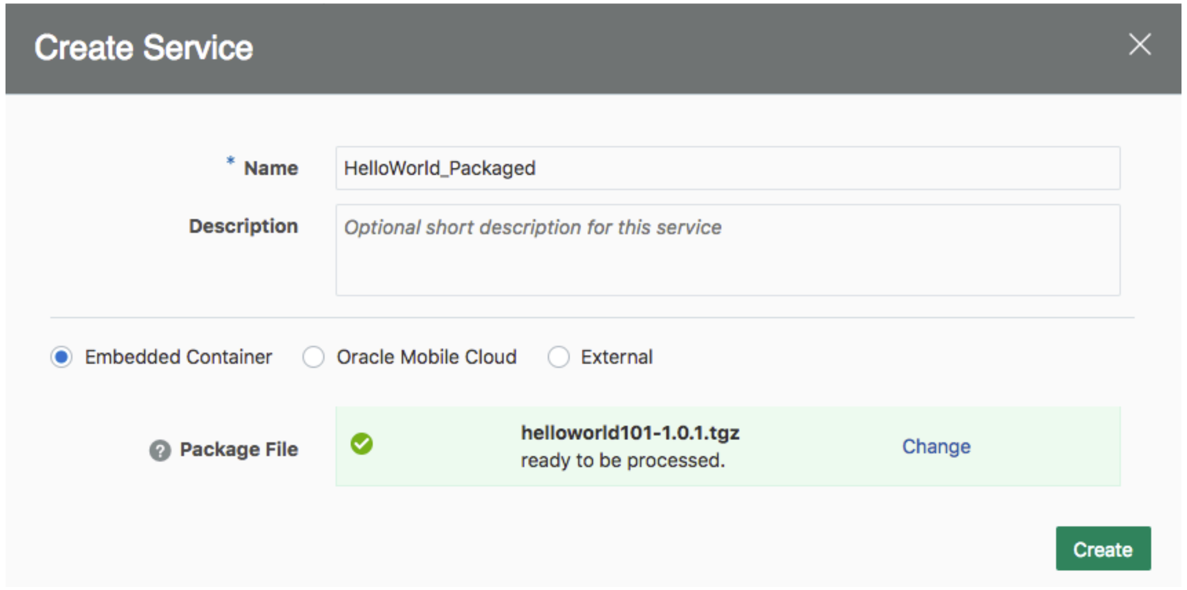


Image courtesy of pixabay.com

Custom component deployment to local container

- Create a .tgz file for local container deployment
  – Run *bots-node-sdk pack* in the component service project folder
- Delete or disable the current custom component service configuration
- Create new service configuration using packaged component service
  – Name different if you kept the debugging configuration


Create Service

* Name	HelloWorld_Packaged
Description	Optional short description for this service

[✓ Embedded Container] Oracle Mobile Cloud External

Package File	helloworld101-1.0.1.tgz ready to be processed. Change

Integrated Cloud

Applications & Platform Services

ORACLE®

Oracle Digital Assistant Hands-On

TBD