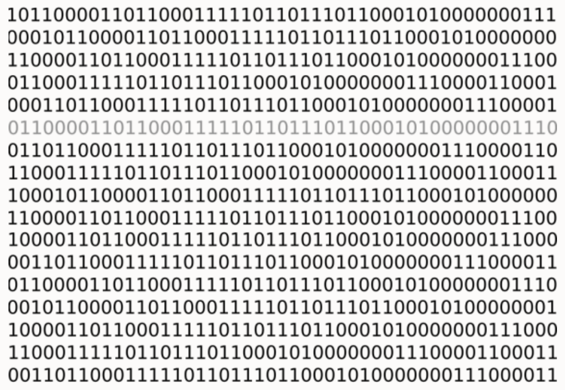
ORACLE

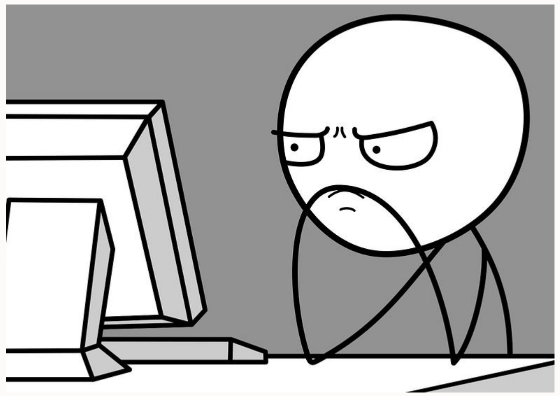
How to debug custom components

V1.0

No more!


CODE

Deploy


Test

Repeat

Image courtesy of pixabay.com

Custom components local debugging steps

Use an IDE that supports Node debugging

- E.g. MS Visual Studio Code

Run `bots-node-sdk service` command in custom component project

- Starts node server on port 3000

Expose port 3000 to the Internet

- May require tunneling (not allowed within Oracle network)

Register custom component service (Type External) URL in skill

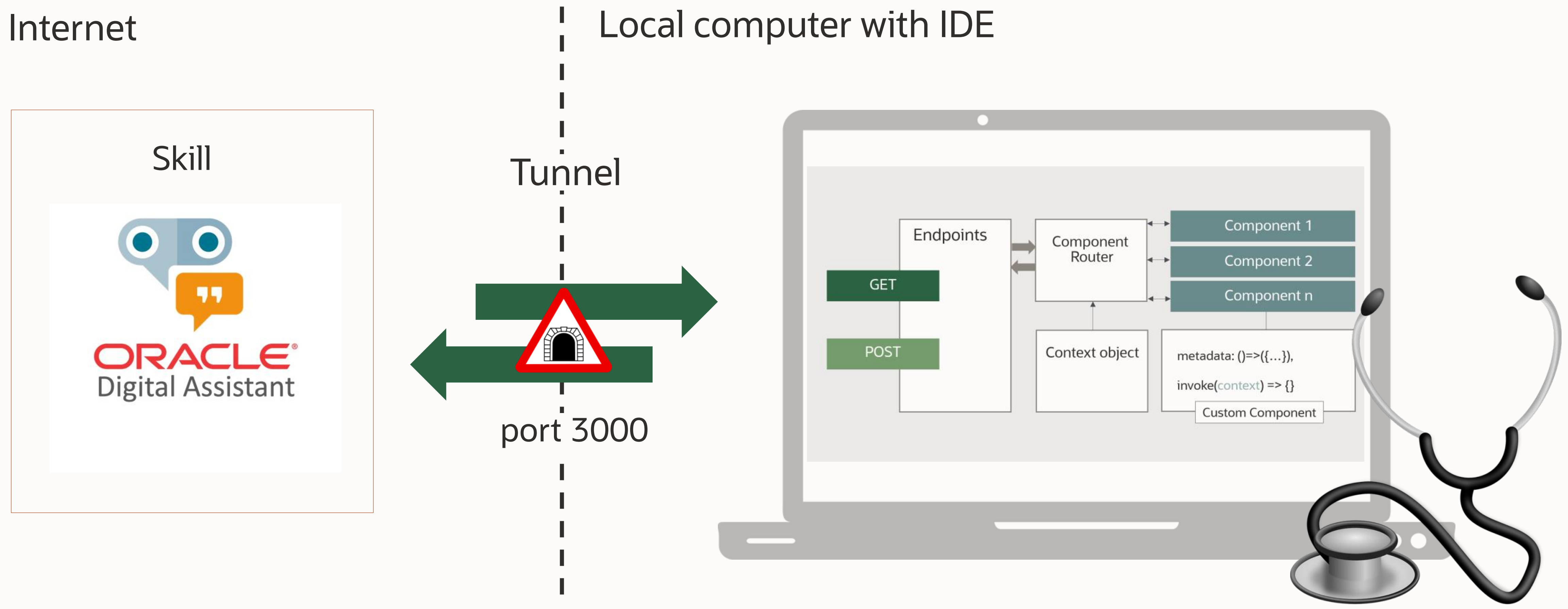
- https://<url>/components

Set breakpoint(s) and run skill in embedded conversation tester

http://bit.ly/ccslocaldebug

Image courtesy of pixabay.com

Local debugging architecture

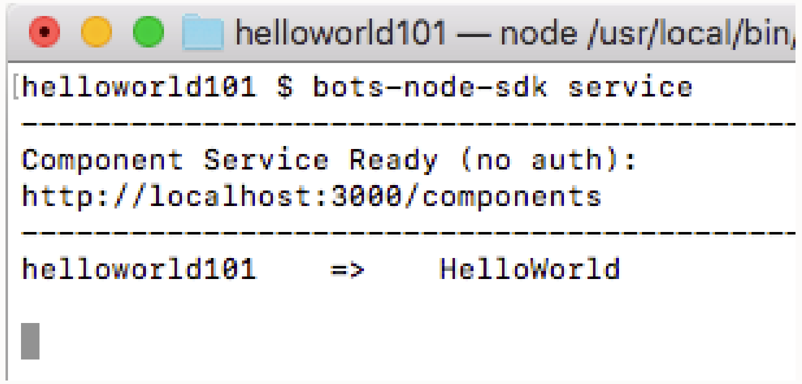


Run custom component service locally

Bots Node SDK configures local express server

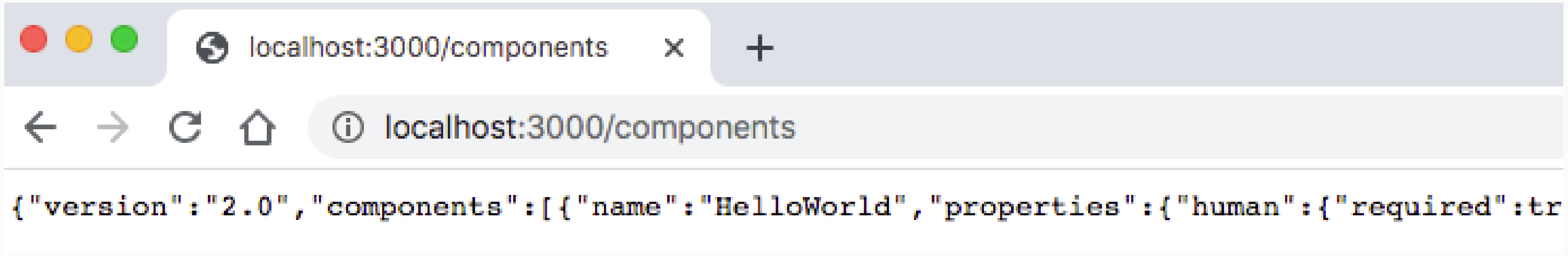
- Custom component service could run locally
- From the custom component service root folder, issue the following command: `bots-node-sdk service`

Use localhost:3000/components to call component service in browser



helloworld101 — node /usr/local/bin,

```
helloworld101 $ bots-node-sdk service
----------------------------------------
Component Service Ready (no auth):
http://localhost:3000/components
----------------------------------------
helloworld101     =>     HelloWorld
```



localhost:3000/components

```
{"version":"2.0","components":[{"name":"HelloWorld","properties":{"human":{"required":tr
```

Configure external component service

Create new component service registration

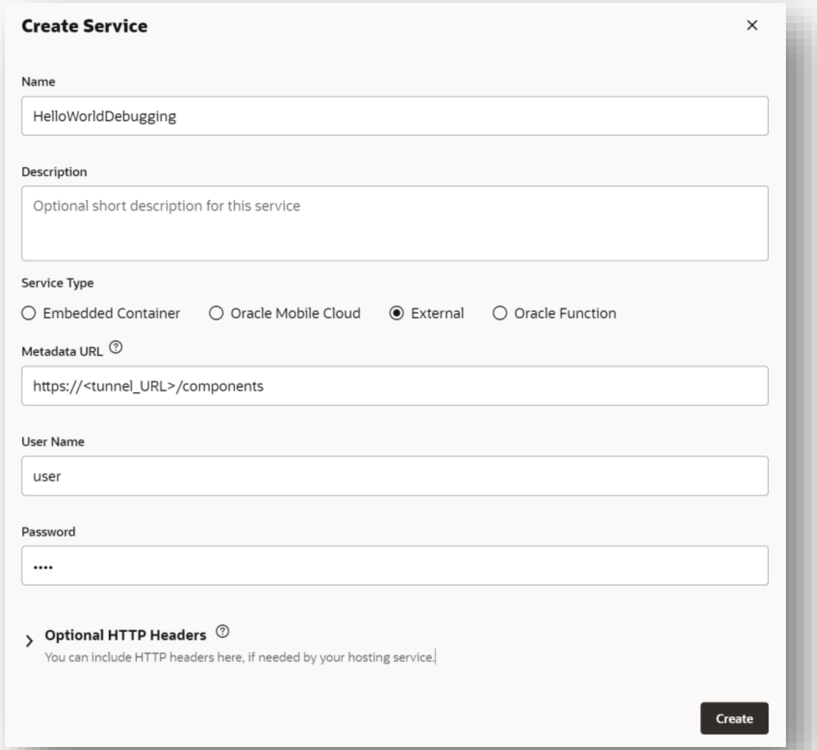
- Choose 'External' option

Register remote URL

- Https URL exposed by tunnel

Disable locally deployed service

- Ensure remote service to be used



Create Service

Name: HelloWorldDebugging

Description: Optional short description for this service

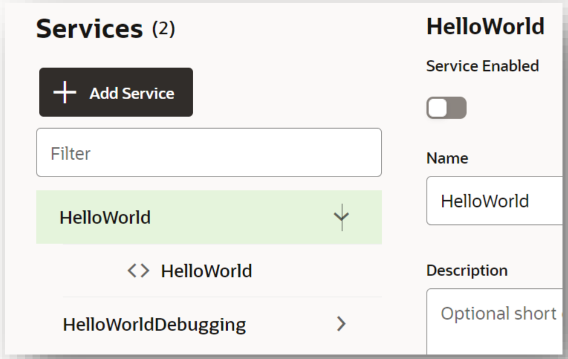
Service Type: ( ) Embedded Container ( ) Oracle Mobile Cloud [✓ External] ( ) Oracle Function

Metadata URL: https://<tunnel_URL>/components

User Name: user

Password: ••••

Optional HTTP Headers
You can include HTTP headers here, if needed by your hosting service.



Services (2)

HelloWorld
- HelloWorld
- HelloWorldDebugging

HelloWorld

Service Enabled: off

Name: HelloWorld

Description: Optional short

Debugging with MS Visual Studio Code (i / iii)

Open the custom component source file

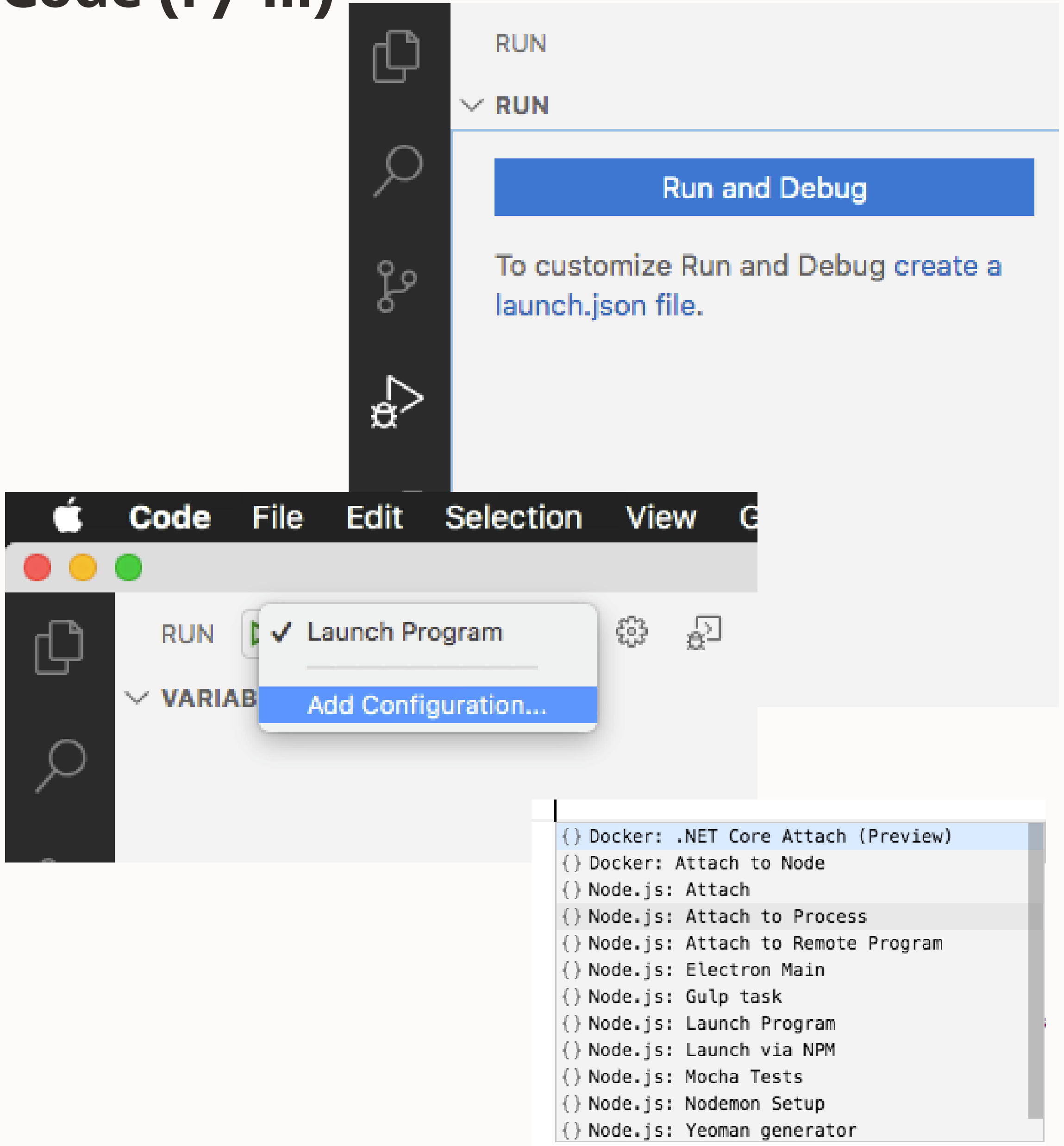
- HelloWorld.js

Select the run / debug icon

Choose **create a launch.json file**

Select **Add Configuration**

Choose **Attach to Process**



To customize Run and Debug create a launch.json file.

Debugging with MS Visual Studio Code (ii / iii)

Name the configuration

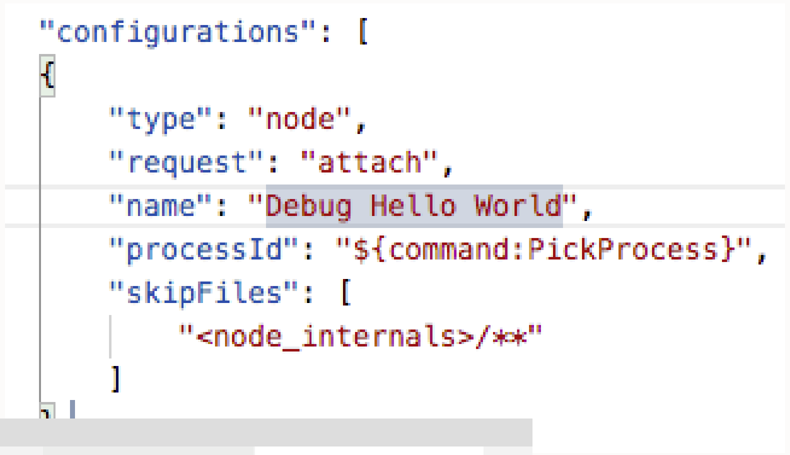
- Save the changes

Set breakpoint(s) inside the custom component's *invoke()* function

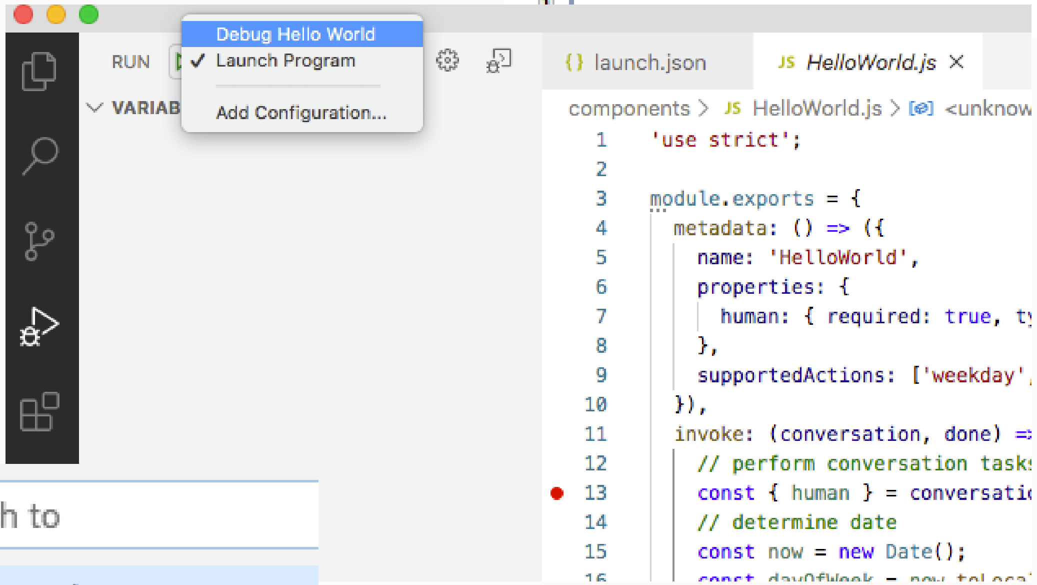
Select run / debug icon

- Choose debug configuration
- Press Run icon

Select process id

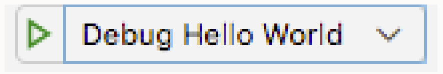


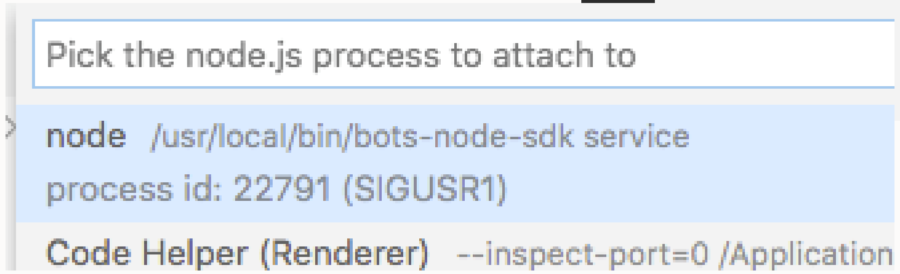
```
"configurations": [
{
    "type": "node",
    "request": "attach",
    "name": "Debug Hello World",
    "processId": "${command:PickProcess}",
    "skipFiles": [
        "<node_internals>/**"
    ]
```



```
'use strict';

module.exports = {
  metadata: () => ({
    name: 'HelloWorld',
    properties: {
      human: { required: true, ty
    },
    supportedActions: ['weekday',
  }),
  invoke: (conversation, done) =
    // perform conversation task
    const { human } = conversatio
    // determine date
    const now = new Date();
```





Pick the node.js process to attach to

node /usr/local/bin/bots-node-sdk service
process id: 22791 (SIGUSR1)

Code Helper (Renderer) --inspect-port=0 /Application

Debugging with MS Visual Studio Code (iii / iii)

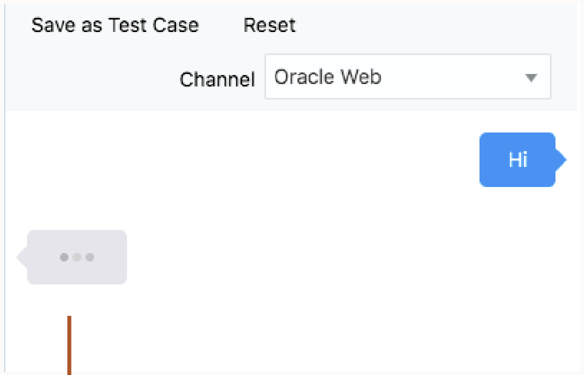
Run skill in embedded conversation tester

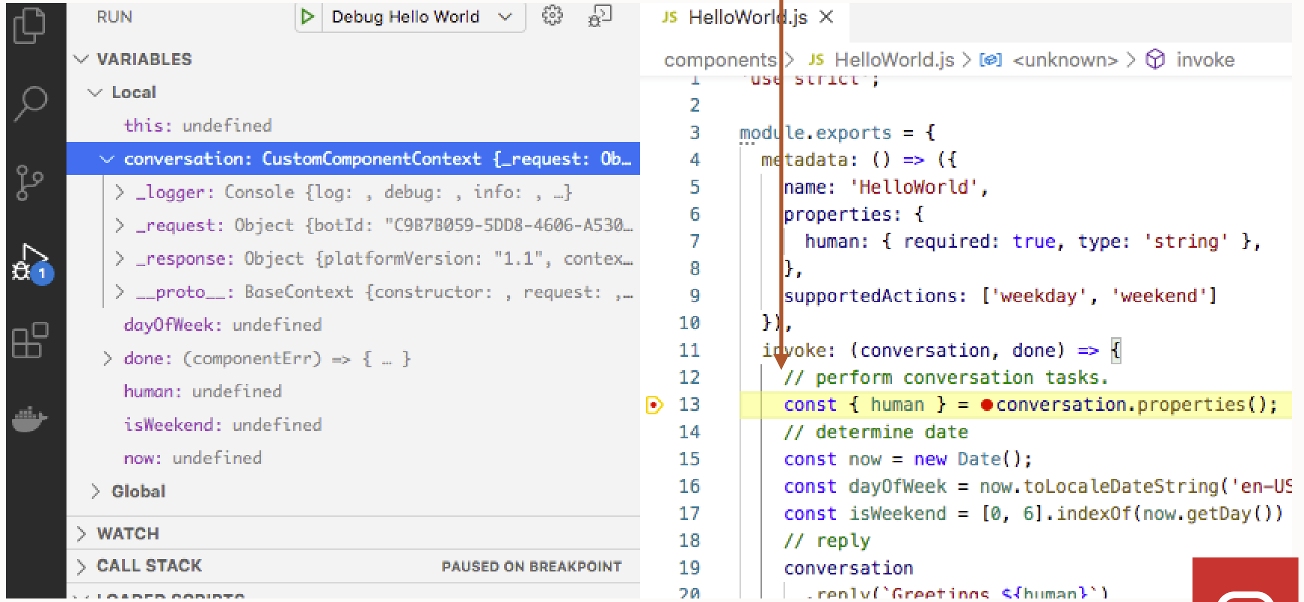
- Execution stops at breakpoint

Start debugging in IDE

Skill tester may not resume after a longer debugging session

- Connection timed out
- Reset tester and repeat testing





```
VARIABLES
  Local
    this: undefined
  ∨ conversation: CustomComponentContext {_request: Ob…
    > _logger: Console {log: , debug: , info: , …}
    > _request: Object {botId: "C9B7B059-5DD8-4606-A530…
    > _response: Object {platformVersion: "1.1", contex…
    > __proto__: BaseContext {constructor: , request: ,…
    dayOfWeek: undefined
  > done: (componentErr) => { … }
    human: undefined
    isWeekend: undefined
    now: undefined
  > Global
> WATCH
> CALL STACK                                  PAUSED ON BREAKPOINT
```

```
'use strict';

module.exports = {
  metadata: () => ({
    name: 'HelloWorld',
    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
    const isWeekend = [0, 6].indexOf(now.getDay())
    // reply
    conversation
      .reply(`Greetings ${human}`)
```

Use nodemon utility to automatically restart the Node server

Automatically restarts custom component service when file changes in the directory are detected.

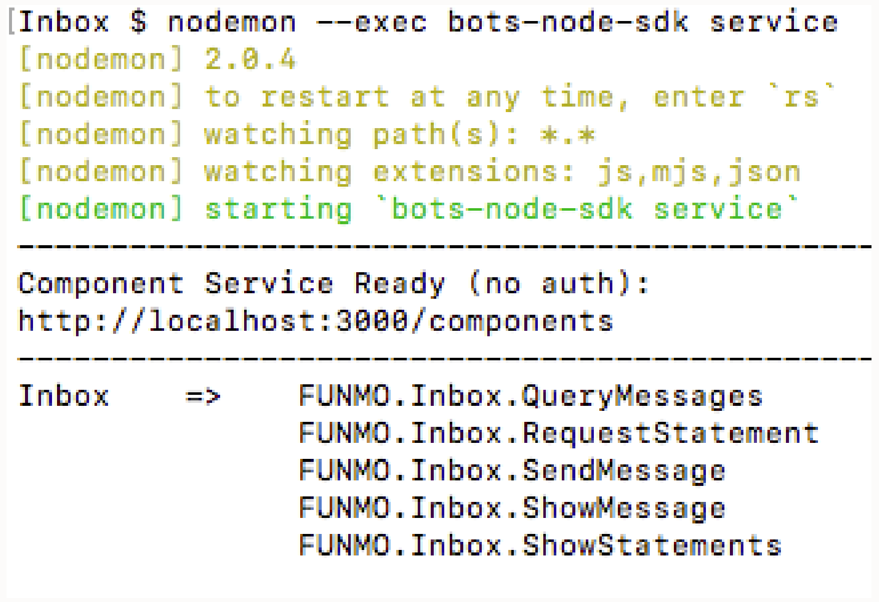
- Does **not** require changes to your code

How to install

- sudo npm install -g nodemon

How start

- On a command line, navigate into your custom component service project
- Issue: `nodemon –exec bots-node-sdk service`



```
Inbox $ nodemon --exec bots-node-sdk service
[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `bots-node-sdk service`
----------------------------------------------
Component Service Ready (no auth):
http://localhost:3000/components
----------------------------------------------
Inbox     =>     FUNMO.Inbox.QueryMessages
                 FUNMO.Inbox.RequestStatement
                 FUNMO.Inbox.SendMessage
                 FUNMO.Inbox.ShowMessage
                 FUNMO.Inbox.ShowStatements
```

https://www.npmjs.com/package/nodemon

Recommendation

Consider using version control

- Git with MS Visual Studio Code or SourceTree

If you can, use local debugging

- Time saving
- Tunneling not allowed in Oracle network
    - Find machine that is exposed to Internet
    - Develop outside Oracle network

Image courtesy of pixabay.com

ORACLE