

ORACLE

Custom components overview

Agenda

- 1 About custom components
- 2 Custom component service architecture
- 3 Custom component SDK
- 4 Recommendations
- 5 Debugging

Agenda

- 1 **About custom components**
- 2 Custom component service architecture
- 3 Custom component SDK
- 4 Recommendations
- 5 Debugging

About custom components

Like built-in components, just custom

- Expose properties
- Can trigger (action) transitions
- May render a user interface (bot response)

Use cases

- Backend system calls¹⁾
- Integrate custom logic into a conversation

Developed using Node.js

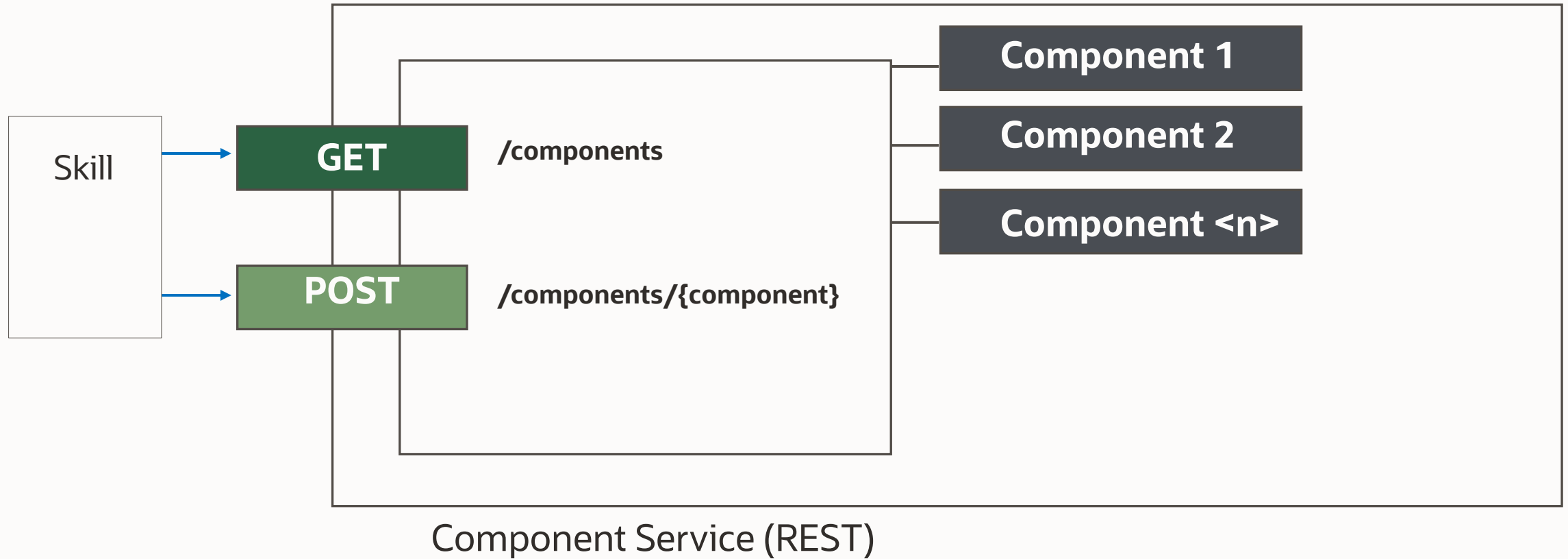
Packaged (.tgz) and deployed in a custom component services (CCS)

- Technically, a custom component service is a REST service developed in Node

¹⁾ You can also use the built-in REST component for accessing backend systems



Custom component service architecture



Deployment options

Local skill component container

- Custom component service resides in skill

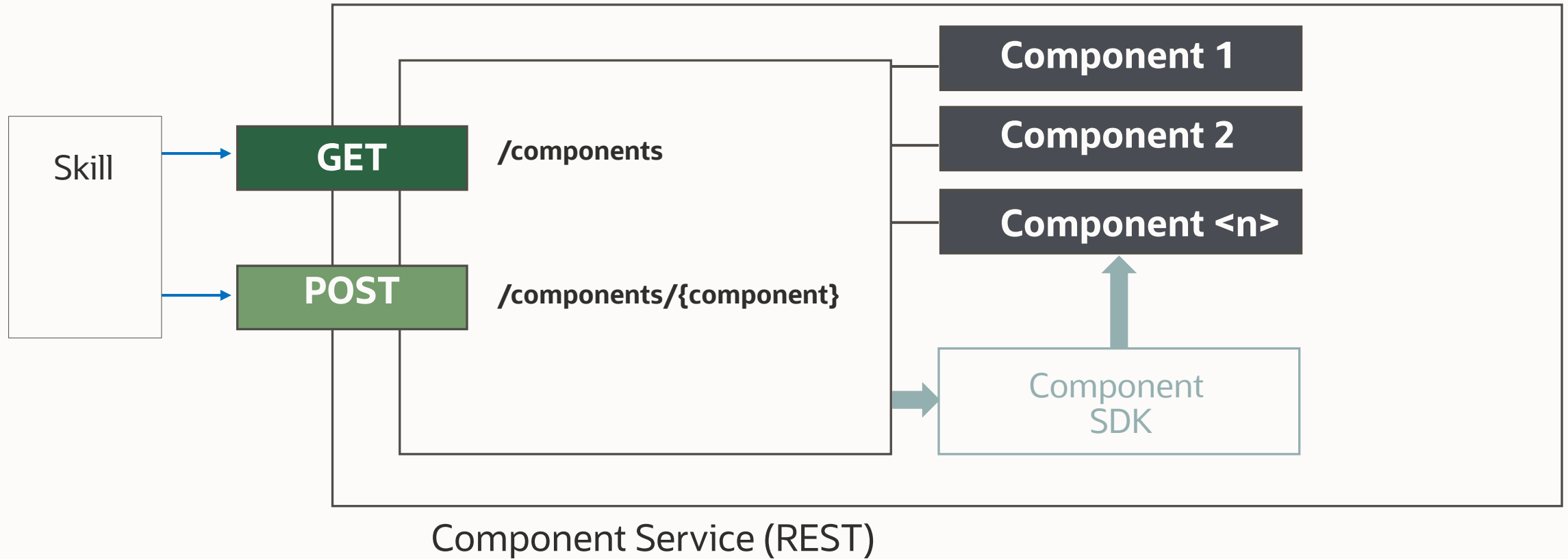
External deployment

- Oracle external functions
- Oracle Kubernetes Engine (OKE) in OCI
- External Node server

Agenda

- 1 About custom components
- 2 **Custom component service architecture**
- 3 Custom component SDK
- 4 Recommendations
- 5 Debugging

Custom component service architecture



Agenda

- 1 About custom components
- 2 Custom component service architecture
- 3 **Custom component SDK**
- 4 Recommendations
- 5 Debugging

About the custom component SDK

Oracle bots-node-sdk available on GitHub to simplify component development

- Requires Node.js and Node Package Manager (NPM) to be installed on developer machine

Reads from the request payload and write back into the response payload

- No need for you to understand how the payloads look like

Provides utility functions to

- Read from input properties and write back to variables
- Optionally, generate user interfaces (prompts, messages)
- Access other skill and request specific information
 - E.g., type of request: text message, button pressed, attachment upload
 - Channel type (websdk, ios, android, slack, msteams, facebook ...)

Access to resolved Intent and extracted entity information

”

The bots-node-sdk does it all
for you

Create a component service
and custom component

Provide a command to package
component service for deployment

Provide embedded Node server
for debugging and testing

2 commands to get from zero to hero

Create a "Hello World" custom component project using the bots-node-sdk

```
bots-node-sdk init <subfolder name> --name <component service name>
```

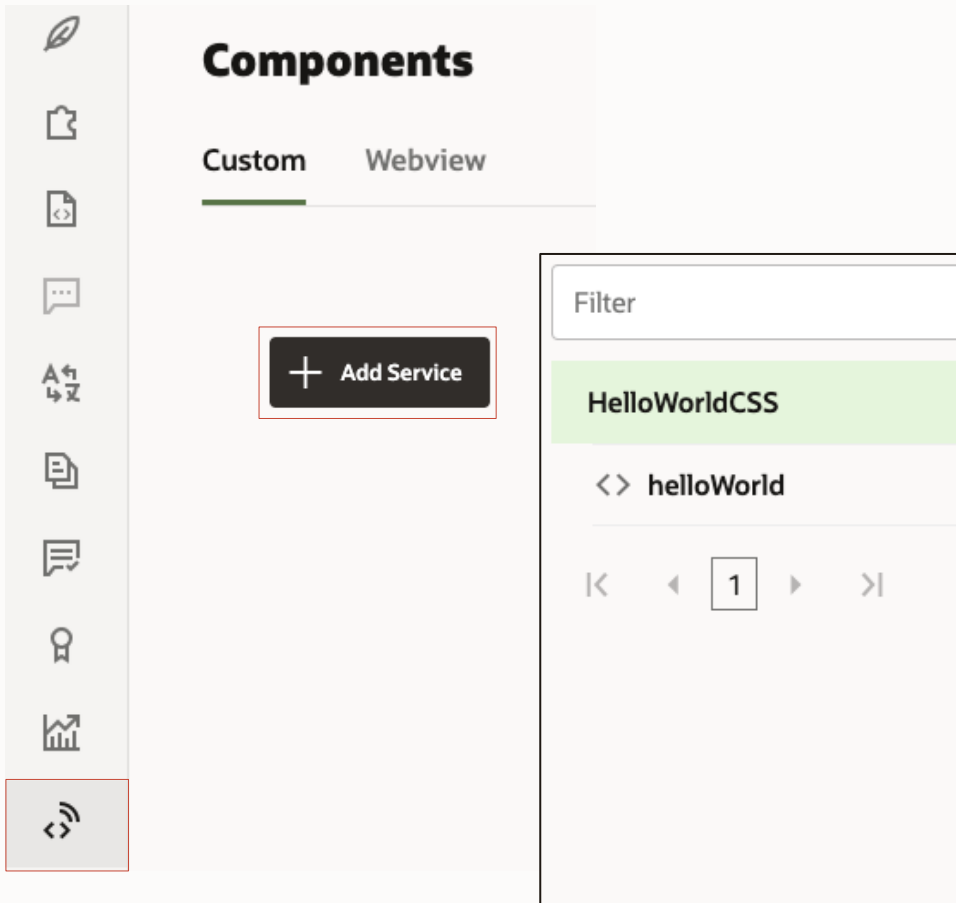
e.g.

```
bots-node-sdk init HelloWorld --name HelloWorldCCS
```

Package component for deployment

```
bots-node-sdk pack
```

How to register a custom component service in a skill



Use "Add Service" button to add a new custom component service registration

Provide a meaningful name

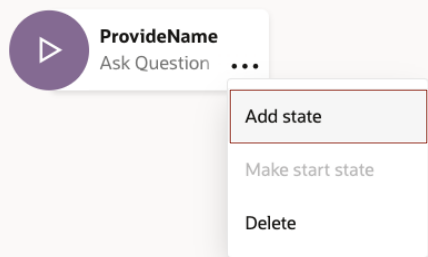
- Name is displayed when adding a component to the dialog flow

Select a deployment option

- Embedded Container
- External
- Oracle Function

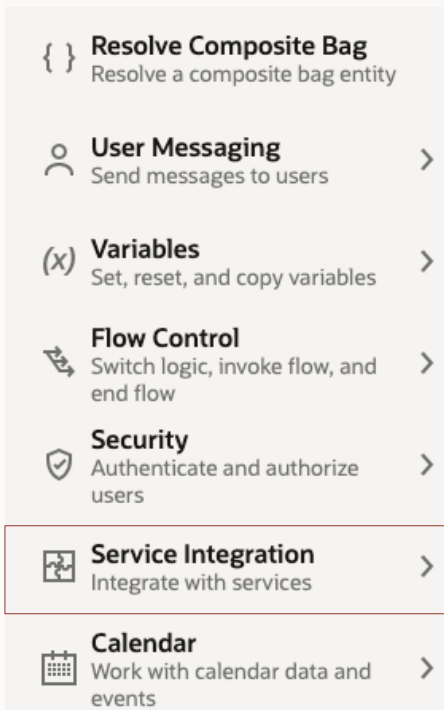
When registered, all custom components in a service are displayed

How to add a custom component to a flow



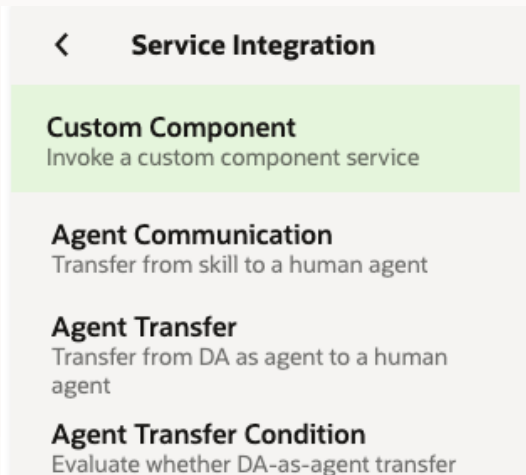
ProvideName
Ask Question ...

- Add state
- Make start state
- Delete



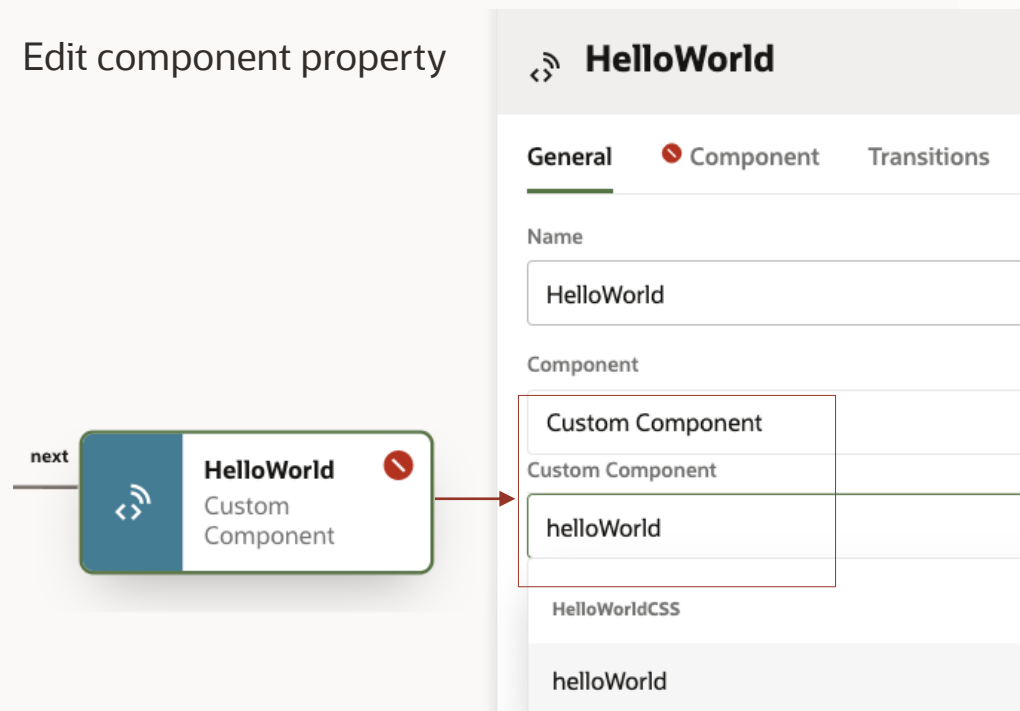
- Resolve Composite Bag**
Resolve a composite bag entity
- User Messaging**
Send messages to users
- Variables**
Set, reset, and copy variables
- Flow Control**
Switch logic, invoke flow, and end flow
- Security**
Authenticate and authorize users
- Service Integration**
Integrate with services
- Calendar**
Work with calendar data and events

Select custom component from template



- Service Integration**
- Custom Component**
Invoke a custom component service
- Agent Communication**
Transfer from skill to a human agent
- Agent Transfer**
Transfer from DA as agent to a human agent
- Agent Transfer Condition**
Evaluate whether DA-as-agent transfer

Edit component property



HelloWorld

General **Component** Transitions

Name
HelloWorld

Component

- Custom Component
- Custom Component
- helloWorld
- HelloWorldCSS
- helloWorld

Name
HelloWorld



Agenda

- 1 About custom components
- 2 Custom component service architecture
- 3 Custom component SDK
- 4 **Recommendations**
- 5 Debugging

What you should do

Build custom components with reuse in mind

- Try at least

Choose a unique component name

- Components might be used together with custom components developed by 3rd party

Make no assumptions

- Don't assume dialog flow variables to exist
- Pass anything the custom component needs to know or work with as input parameters



Consider this when developing custom components

” Keep code in custom components simple and clean

Build integration and transformation layers

- Remote services that optimize data payloads for use in bots

Move reusable code logic into JavaScript libraries deployed with the custom component service

Document and image handling should be done in the cloud, especially if the custom component is deployed to the local skill container

Have an error handling strategy

- Backend service access errors, input validation errors, component exceptions

I am sorry. But our kitchen reported a sy-xp009 'malfunction of sys core ctrl-op' for the grill. It causes temperature fluctuations.



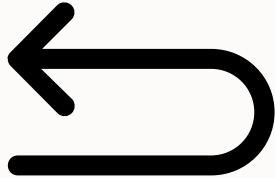
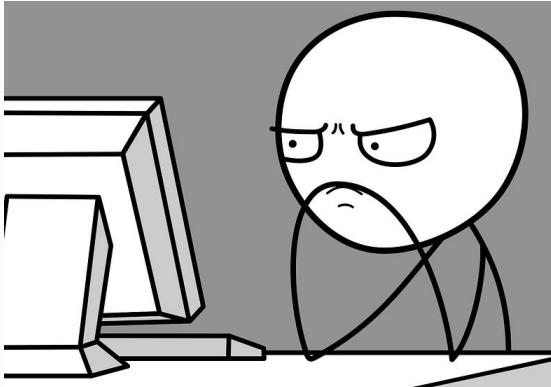
Don't show error messages to users.
They still think they are in a conversation.

Agenda

- 1 About custom components
- 2 Custom component service architecture
- 3 Custom component SDK
- 4 Recommendations
- 5 **Debugging**

No more!

```
10110000110110001111101101110110001010000000111
00010110000110110001111101101110110001010000000
11000011011000111110110111011000101000000011100
01100011111011011101100010100000001110000110001
00011011000111110110111011000101000000011100001
01100001101100011111011011101100010100000001110
01101100011111011011101100010100000001110000110
11000111110110111011000101000000011100001100011
10001011000011011000111110110111011000101000000
11000011011000111110110111011000101000000011100
10000110110001111101101110110001010000000111000
00110110001111101101110110001010000000111000011
01100001101100011111011011101100010100000001110
00101100001101100011111011011101100010100000001
10000110110001111101101110110001010000000111000
11000111110110111011000101000000011100001100011
00110110001111101101110110001010000000111000011
```



CODE

Deploy

Test

Repeat



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

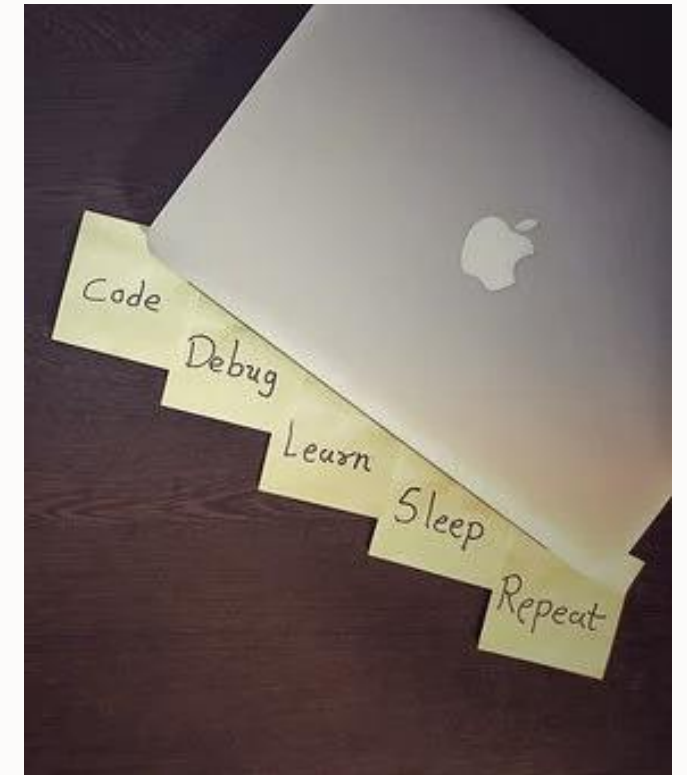
Expose port 3000 to the Internet

- May require tunneling

Register custom component service URL in skill

- `https://<url>/components`

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



ORACLE