HOW TO BUILD YOUR FIRST VOICE APP

HEARST
tech connect

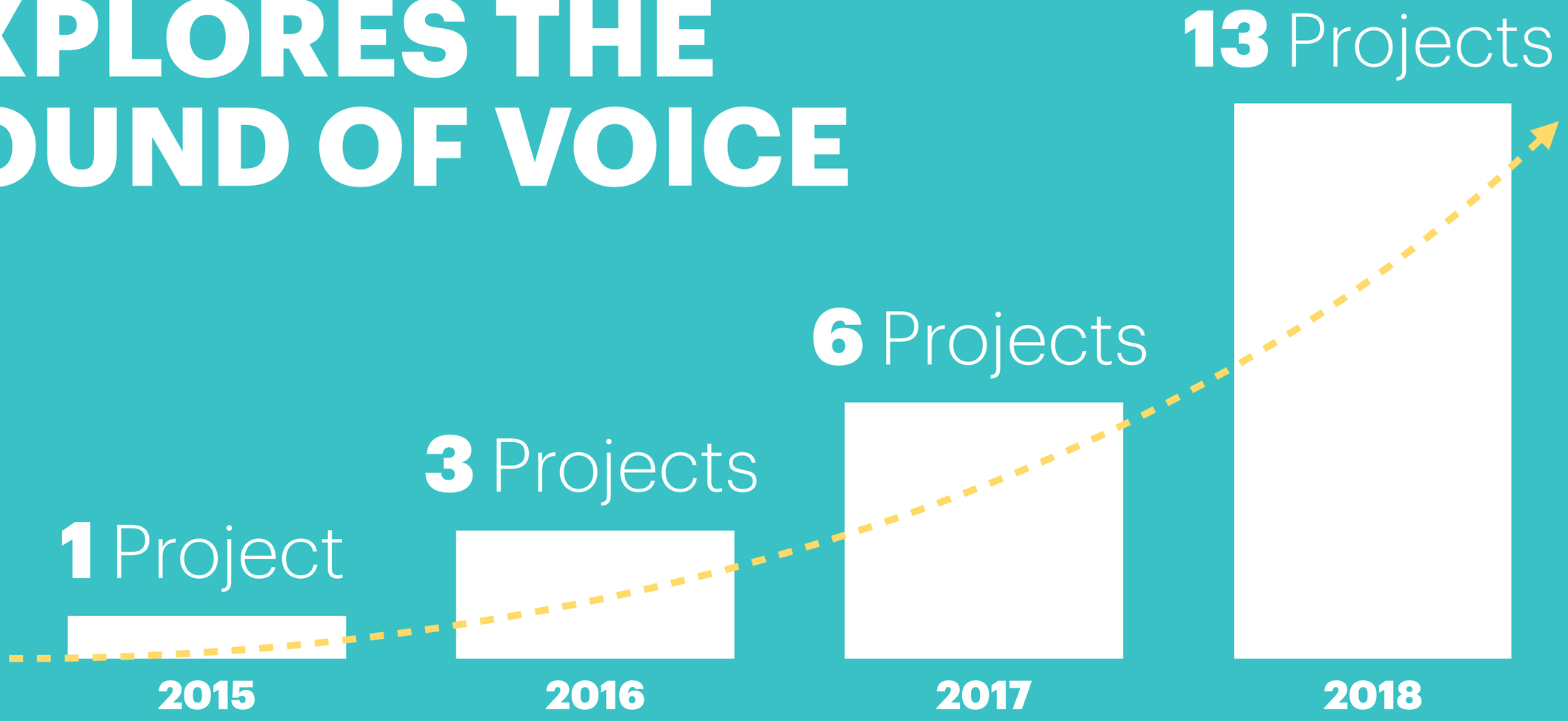
WHY VOICE

Nearly 1 in 5 U.S. Adults
Have Access to a Smart Speaker Today

By 2020, 55% of U.S. Households
Will Have Smart Speakers

More Than 175M Smart Speakers
Will Be Installed in The U.S. by 2020

HEARST EXPLORES THE SOUND OF VOICE



AGENDA

Alexa, Can I Get Some Context?
Understanding the platforms

The Brainstorm
Things to consider when designing a Voice app

What To Build
Types of Alexa Voice products

The Workshop
Build a horoscope skill

Moving Forward
Tools & resources

Google Assistant

Devices

Google Home, Home Mini, Home Max

Built into select smartphones

Adoption

26.9% of U.S. smart speaker market

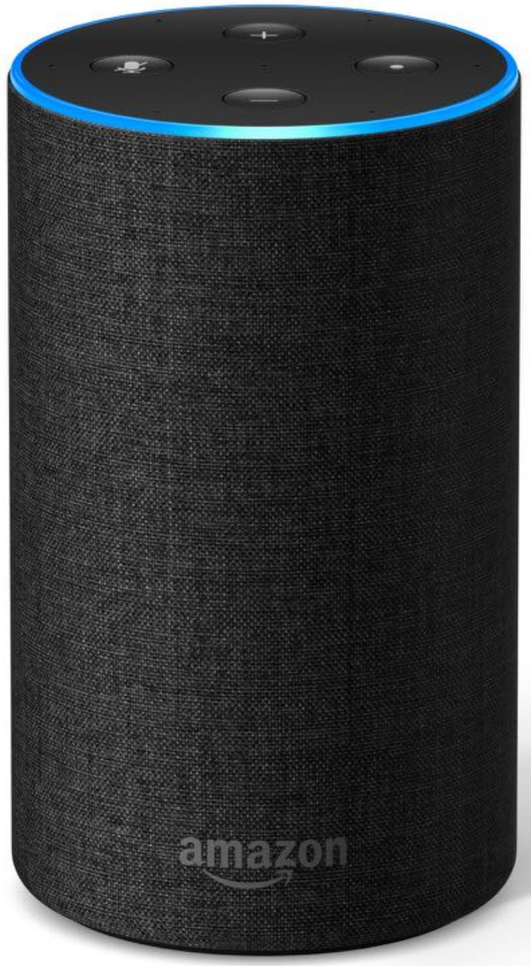
Voice Apps

2,400 as of January 2018

Revenue Strategy

Skill Purchase

amazon alexa



Devices

Echo, Echo Dot,
Echo Show, Echo Spot,
Echo Look, Fire TV Cube,
Fire Tablet

Amazon Alexa app

Adoption

61.9% of U.S. smart speaker market

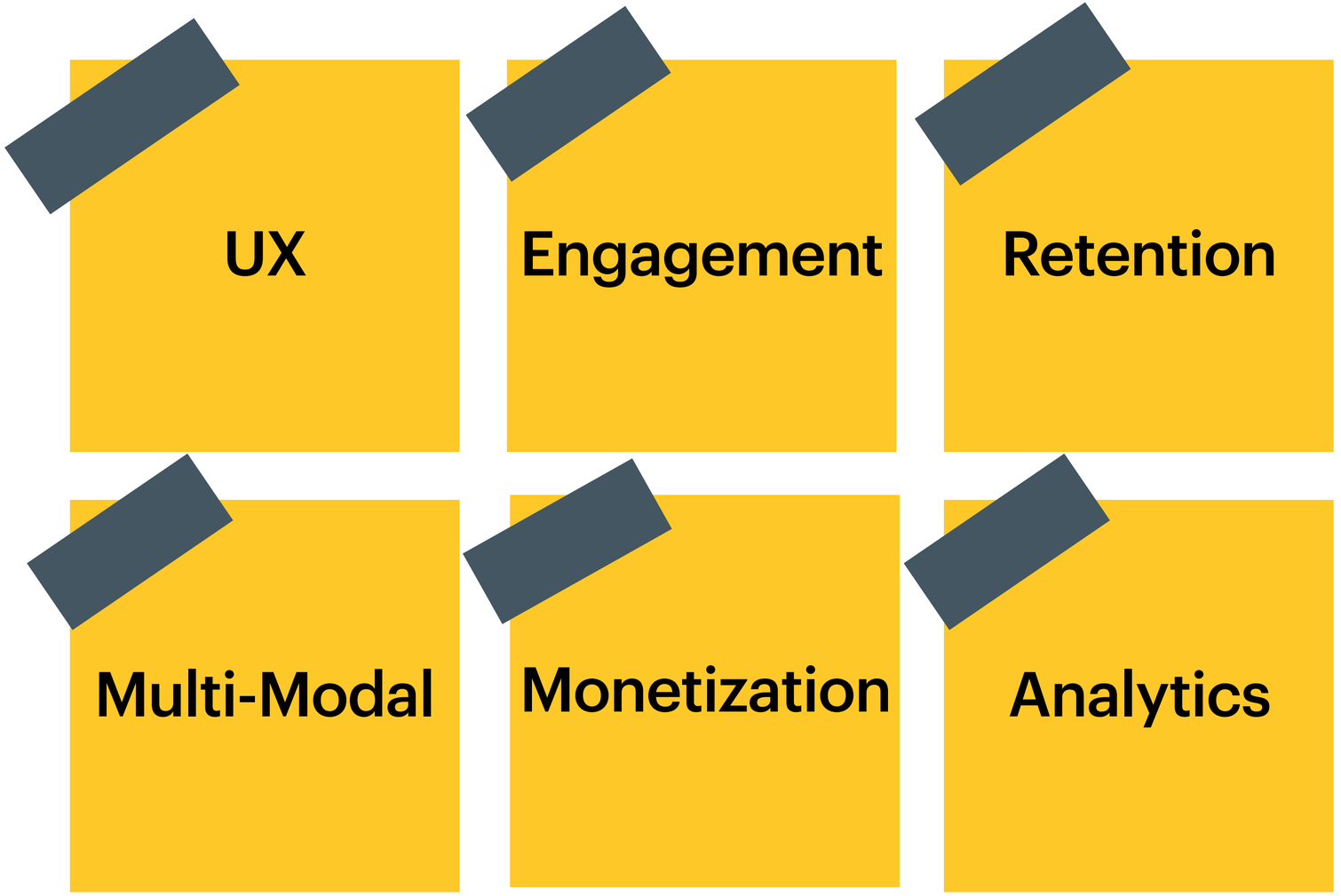
Voice Apps

+45,000 as of July 2018

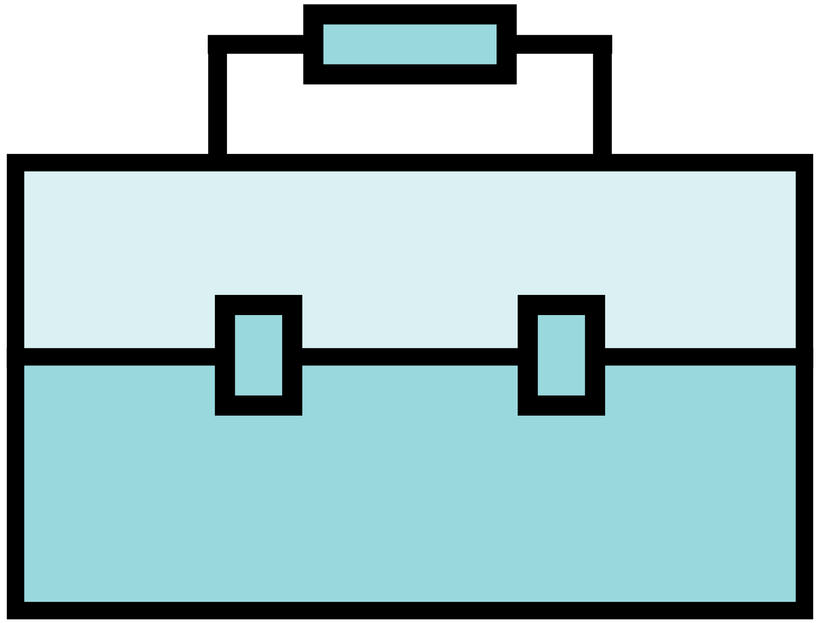
Revenue Strategy

Sponsorship,
Subscription,
In-Skill Purchase

THE BRAIN-STORM



WHAT TO BUILD



- Custom Skills
- Flash Briefings
- Multi-Modal
- Video Skills
- Integrations

READY FOR TAKE-OFF

LET'S REACH FOR THE STARS &
BUILD A HOROSCOPE SKILL!

// WHAT MAKES A SKILL

Utterances

Words and phrases that users can say to invoke intents in your skill.

Invocation Name

A name that identifies your skill that users can say to Alexa to launch your skill.

Slots

Specific words/phrases within your sample utterances to represent variables of information.

Intents

A set of actions users can do with your skill.

SSML

A text to speech markup language that allows you to manipulate how Alexa speaks your content.

Voice User Interface (VUI)

Establish the purpose and user stories

- Determine the capabilities of the skill.
- Identify unique steps & actions.

Write scripts

- Show the conversation between the user and Alexa.
- Keep interactions brief, avoid repetition, and don't assume the user will know what to do or what will happen.

Develop the flow

- Outline the shortest route to completion.
- Outline alternate paths and decision trees.

Get ready to build

- Identify intents & utterances

1. New User

Invocation · Response · Utterance

VOICE

User: Alexa, open Elle Horoscopes by the Astro Twins
Alexa: ELLE Horoscopes by The Astro Twins. What sign do you want the horoscope for?
Reprompt: With ELLE Horoscopes by The Astro Twins, you can get your horoscope for today. For example, you can say what is the horoscope for Gemini, or you can say exit. Now, which sign do you want?

User: Leo
Alexa: ELLE Horoscopes by The Astro Twins. Leo, August 10th. With get-‘er-done Saturn in your hard-working sixth house throwing shade at the moon…
Alexa: You can either give me a new sign or say 'goodbye' to exit. What would you like?
Reprompt: I didn't hear you. You can either give me a new sign or say 'goodbye' to exit. What would you like?

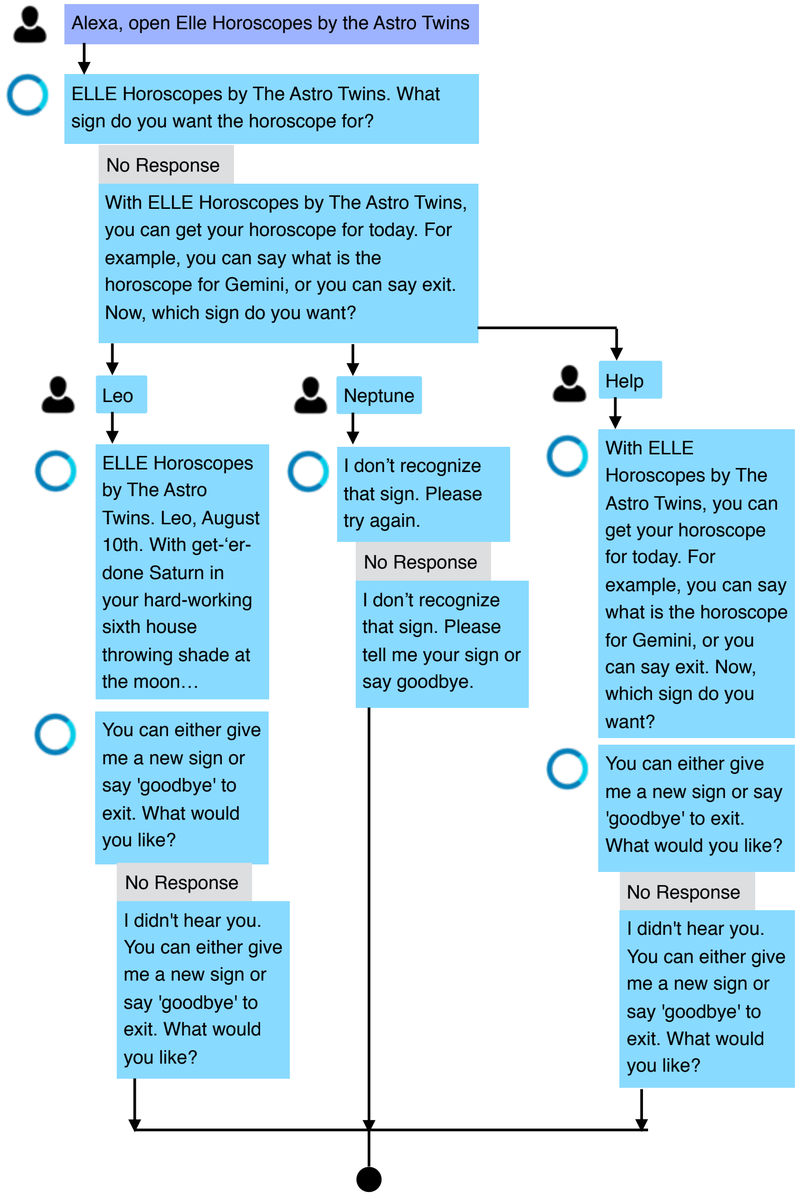
OR

User: Neptune
Alexa: I don’t recognize that sign. Please try again.
Reprompt: I don’t recognize that sign. Please tell me your sign or say goodbye.

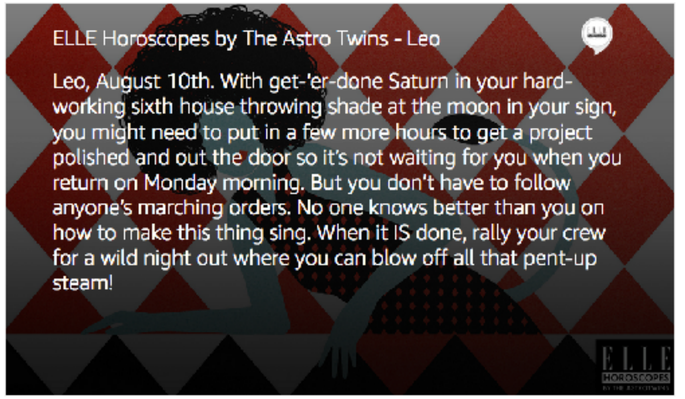
OR

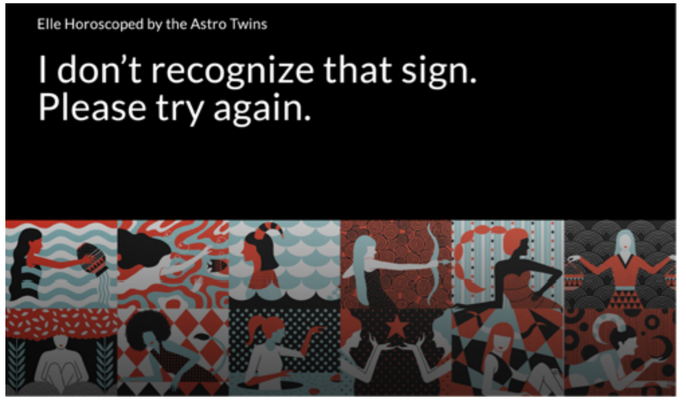
User: Help
Alexa: With ELLE Horoscopes by The Astro Twins, you can get your horoscope for today. For example, you can say what is the horoscope for Gemini, or you can say exit. Now, which sign do you want?
Reprompt: Which sign do you want?

FLOW

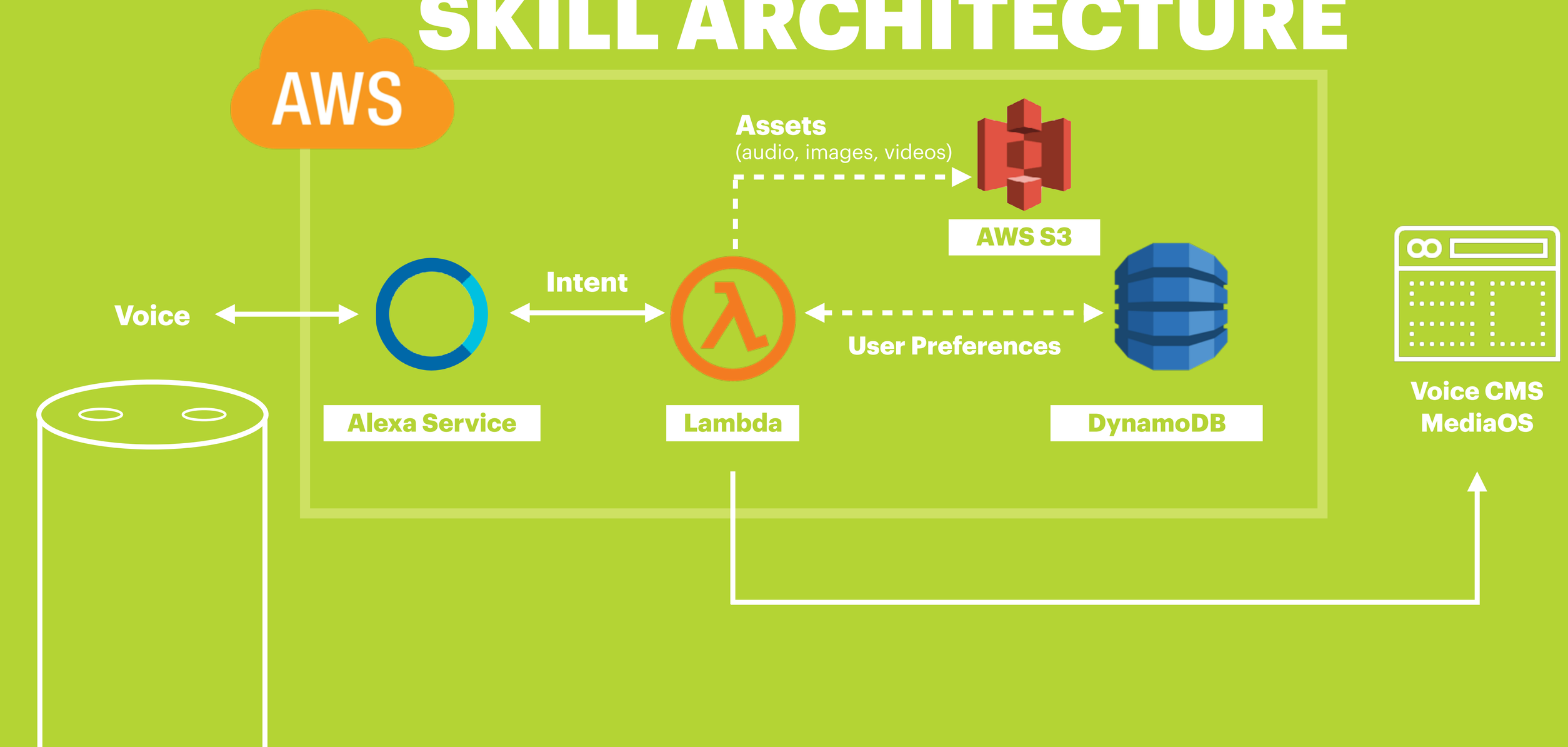


GUI





SKILL ARCHITECTURE



REQUEST

The Alexa service processes incoming user utterances and converts them to intents.

This data is sent to your lambda function as a JSON event which includes the following information:

- Intent
- Slot
- Session Attributes
- User Attributes
- Persistence Attributes from DynamoDB

RESPONSE

The skill code needs to handle these incoming requests & return an appropriate response based on user state.

Responses need to include the following information:

- Speech Prompt
- Optional Re-prompt
- ShouldEndSession Parameter
- JSON representation of any visuals for Echo Show, Spot, FireTV & Alexa Companion App

// HOW TO DEFINE YOUR MODELS

The **model file** defines the following parts of your Alexa Skill:

- Invocation name
- Intents
- Utterances
- Slots

A **slot type** can either be custom or can use one of Amazons predefined types

- Date
- City
- Search Query
- etc

```
"interactionModel": {
  "languageModel": {
    "invocationName": “Bartender”,
    "intents": [
      {
        "name": "CocktailForLiquor",
        "slots": [
          {
            "name": "liquor",
            "type": "LiquorType"
          }
        ],
        "samples": [
          "What are some cocktails using {liquor}",
          "Give me a {liquor} cocktail"
        ]
      }
    ],
    "types": [
      {
        "name": "LiquorType",
        "values": [
          {
            "name": {
              "value": "gin"
            }
          },
          {
            "name": {
              "value": "tequila"
            }
          },
          {
            "name": {
              "value": "rum"
            }
          }
        ]
      }
    ]
  }
}
```

```
const CocktailForLiquorHandler = {
    canHandle( handlerInput ) {
        const request = handlerInput.requestEnvelope.request;

        return request.type === e.REQUEST_TYPE.INTENT
           && request.intent.name === e.INTENT.COCKTAIL_FOR_LIQUOR;
    },
    async handle( handlerInput ) {
        // Get inputted liquor from request
        const liquorSlot = handlerInput.requestEnvelope.request.intent.slots.liquor;

        // validate liquorSlot (check if empty or contains unknown values)
        const liquorType = cocktailUtility.validate( liquorSlot.value );

        if ( liquorType === constants.errorCode.UNSUPPORTED ) { // Unknown sign returned
            return defaultResponses.generalError( handlerInput );
        } else {
            const requestAttributes = handlerInput.attributesManager.getRequestAttributes();

            try {
                // Fetch cocktail data for liquorType
                const cocktails = await cocktailService.getCocktailsForLiquor( liquorSlot.value );

                const replaceObj = {
                    liquor: liquorType,
                    cocktails: cocktails
                };

                const responseObj = requestAttributes.t( "cocktailForLiquor", replaceObj );

                return responseBuilder.tell( handlerInput, responseObj );
            } catch ( error ){
                const responseObj = requestAttributes.t( "error" );

                return responseBuilder.ask( handlerInput, responseObj );
            }
        }
    }
};
```

// RESPONSE HANDLERS

// STRING MANAGEMENT & MANIPULATION

All of a skill's response strings should be stored in the **strings/en-us.json** file:

```
"welcome": {
    "speech": {
        "output": "Welcome to the bartender. I can provide you with cocktail ideas. What type of liquor do you want to use?",
        "reprompt": "What type of liquor do you want to use?"
    }
},
"cocktailForLiquor": {
    "speech": {
        "output": "Here are some cocktails you can make with {{liqour}}: {{cocktails}}"
    },
    "display": {
        "template": "BodyTemplate1",
        "backgroundImage": {
            "url": "display/1024x600/background.png",
            "width": 1024,
            "height": 600
        },
        "text": {
            "primary": {
                "type": "RichText",
                "text": "<font size=\"7\">Here are some cocktails you can make with {{liqour}}: {{cocktails}}</font>"
            }
        }
    }
}
```

The alexa-sdk has tools to parse JSON from strings file & **substitute content**:

```
const replaceObj = {
    liquor: liquorType,
    cocktails: cocktails
};

const responseObj = requestAttributes.t( “cocktailForLiquor", replaceObj );
```

// SIMPLIFYING THE DEV PROCESS USING ASK-TOOLKIT

The *ask* function below is used to form valid Alexa response objects from JSON objects.

```
ask: async ( handlerInput, data, options ) => {
    data = data || {};

    data.speech = responseUtils.buildSsml( data );

    handlerInput.responseBuilder
        .speak( data.speech.output )
        .reprompt( data.speech.reprompt );

    return builder.finalizeResponse( handlerInput, data, options );
}
```

Example usage:

```
const response = {
    "speech": {
        "output": "Welcome to the bartender. I can provide you with cocktail ideas. What type of liquor do you want to use?",
        "reprompt": "What type of liquor do you want to use?"
    },
    "display": {
        …
    },
    “card": {
        …
    }
};

responseBuilder.ask( handlerInput, response );
```

// ASK-TOOLKIT OUTPUTTED RESPONSE

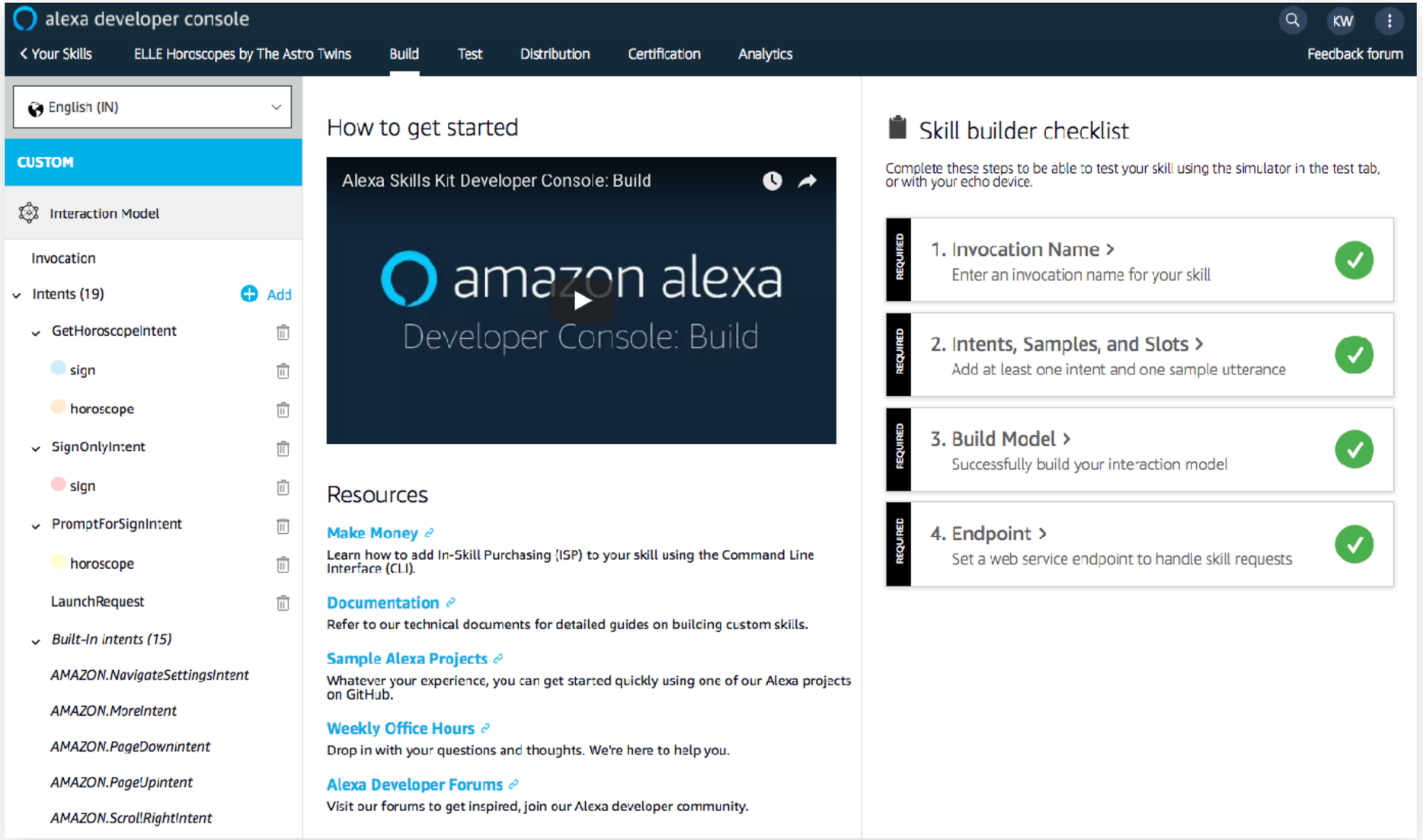

```
{
    "outputSpeech": {
        "type": "SSML",
        "ssml": "<speak>Welcome to the bartender. I can provide you with cocktail ideas. What type of liquor do you want to use?</speak>"
    },
    "reprompt": {
        "outputSpeech": {
            "type": "SSML",
            "ssml": "<speak>What type of liquor do you want to use?</speak>"
        }
    },
    "shouldEndSession": false,
    "directives": [
        {
            "type": "Display.RenderTemplate",
            "template": {
                "type": "BodyTemplate1",
                "textContent": {
                    "primaryText": {
                        "type": "RichText",
                        "text": "<font size=\"7\">What type of liquor do you want to use?</font>"
                    }
                },
                "backButton": "HIDDEN",
                "title": "",
                "token": "notSure"
            }
        }
    ]
}
```

// MORE WITH ASK-TOOLKIT

- Ask (response with a prompt)
- Tell (response that ends a session)
- Builds display output for Echo Show, Spot, FireTV, etc
- Account linking
- Slot input validation
- Analytics
- Key encryption
- List API
- And more...

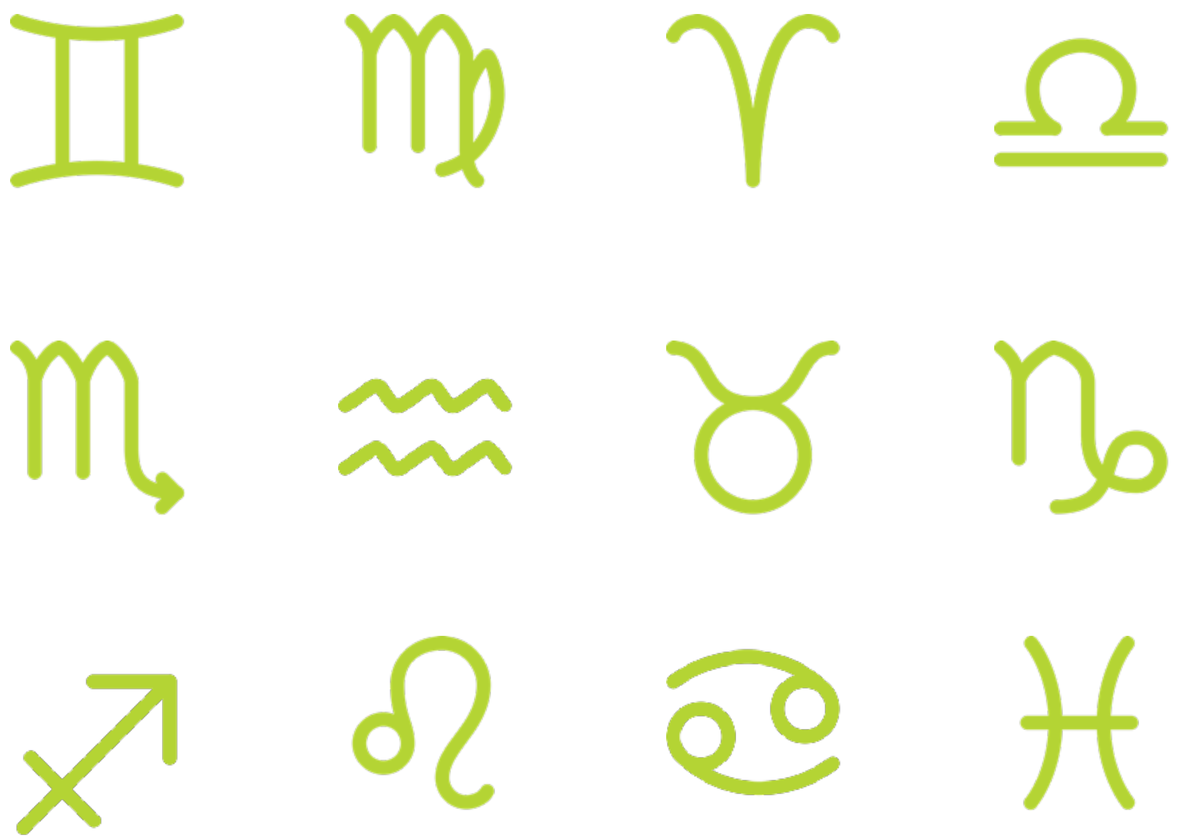
https://github.com/Hearst-DD/ask-toolkit

// DEVELOPER CONSOLE



The Workshop

- Create HoroscopeForSign Intent
- Add a Response Handler for new intent
- Add visuals to the HoroscopeForSign response



Get started at **https://github.com/Hearst-DD/alexa-horoscope-workshop**