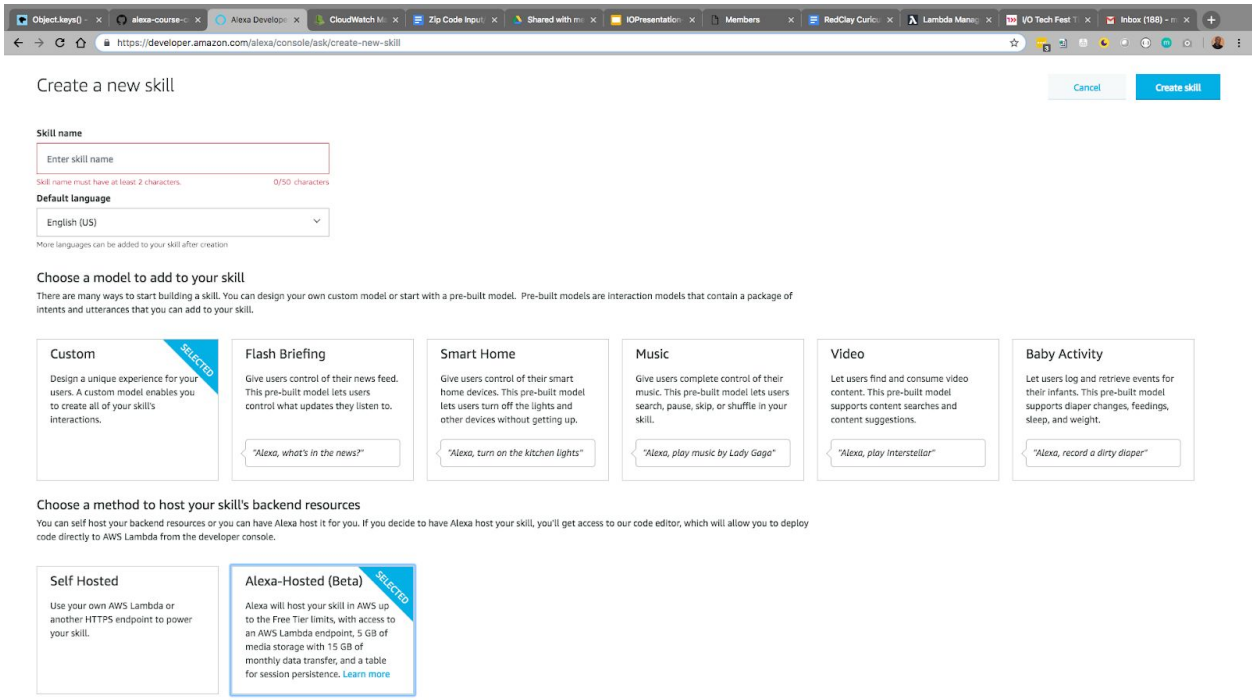
Step 1: Set Up Your Alexa Skill / Developer Portal

1. **Go to the Amazon Developer Portal. In the top-right corner of the screen, click the "Sign In" button.** (If you don't already have an account, you will be able to create a new one for free.)
2. Once you have signed in, move your mouse over the **Developer Console** text at the top of the screen and Select the **Skills** Link.
3. From the **Alexa Skills Console** select the **Create Skill** button near the top-right of the list of your Alexa Skills.
4. Give your new skill a **Name**. This is the name that will be shown in the Alexa Skills Store, and your default **invocation name**. Eg: **"*Wakanda Facts*"**
5. Scroll down till you see the **Alexa-Hosted (Beta) box**
6. select the **Alexa-Hosted** model box to add it to your skill, then click the **Create Skill** button at the top right.



7. Select the **HelloWorldIntent** by expanding the **Intents** from the left side navigation panel. Change it to **HelloIntent.** Add the utterances below under **Sample Utterances**. Make sure you hit the enter key to save each utterance. If prompted click add to add the **name** slot. *See below for a screenshot*

call me {name}

you can call me {name}

my name is {name}

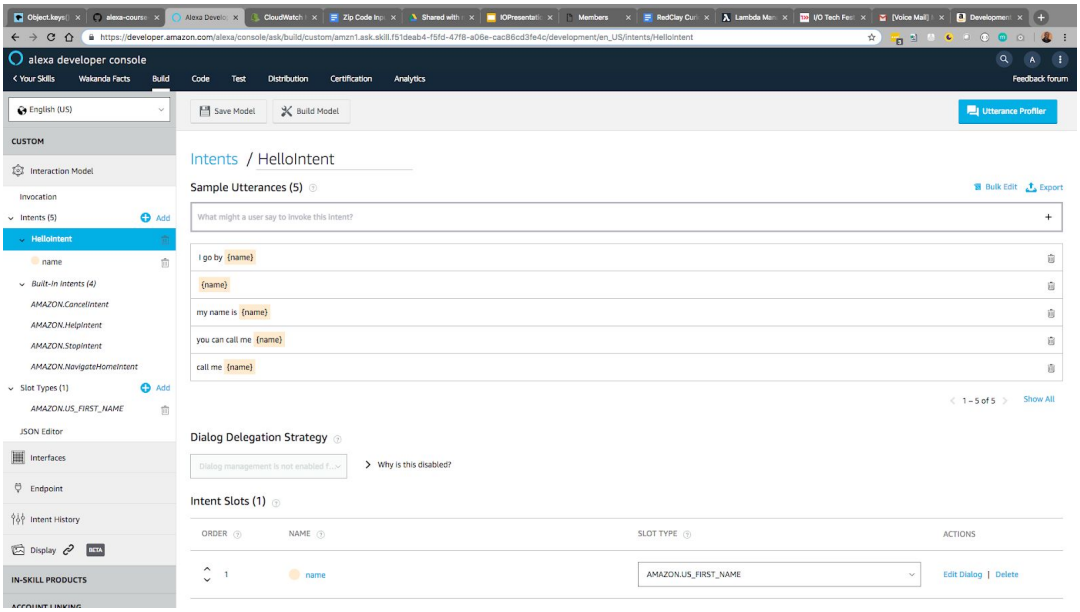
{name}

I go by {name}

Add some more sample utterances for your newly generated intents. Think of all the different ways that a user could introduce themselves so Alexa can make the intent happen.

8. Scroll down till you see the name slot. Next to it is a drop down box. Click the drop down box and select **AMAZON.US_FIRST_NAME**

Be sure to click **Save Model** and **Build Model** after you're done making changes here.

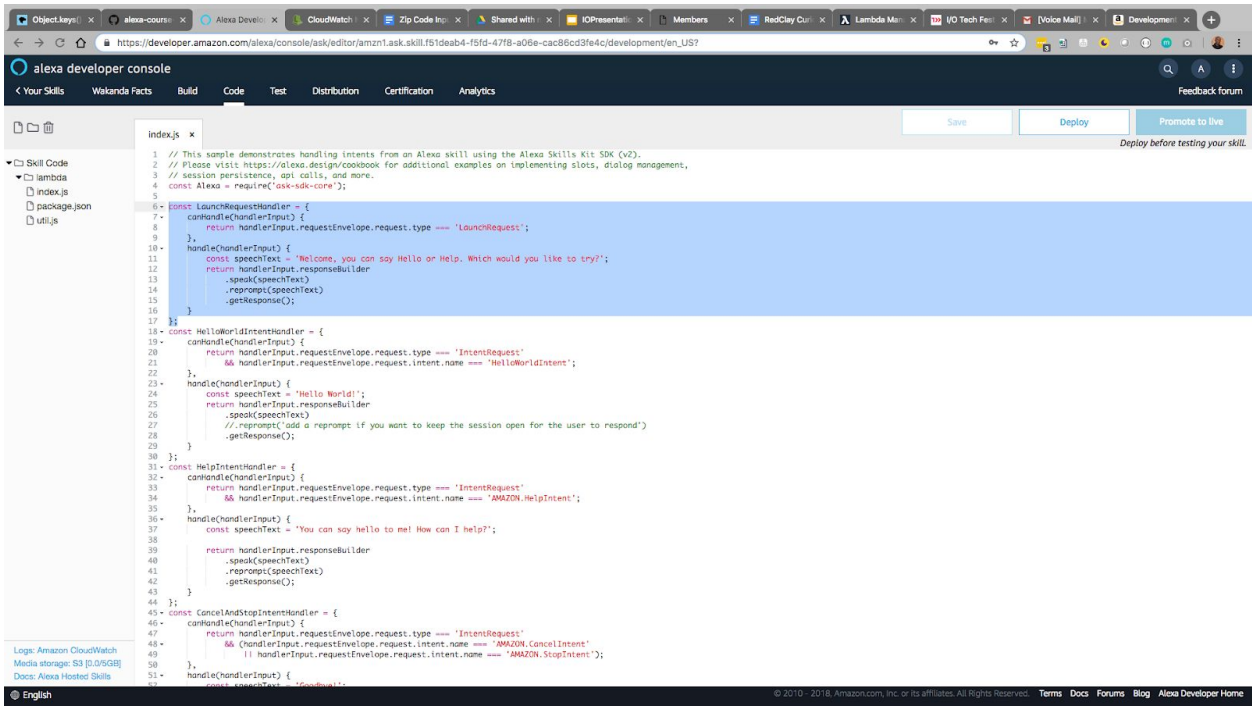


IF YOU RAN INTO ANY ERRORS OR ISSUES BELOW IS THE FINAL CODE FOR THE FRONTEND/ INTERACTION MODEL


```
{
  "interactionModel": {
    "languageModel": {
      "invocationName": "wakanda facts",
      "intents": [
        {
          "name": "AMAZON.CancelIntent",
          "samples": []
        },
        {
          "name": "AMAZON.HelpIntent",
          "samples": []
        },
        {
          "name": "AMAZON.StopIntent",
          "samples": []
        },
        {
          "name": "HelloIntent",
          "slots": [
            {
              "name": "name",
              "type": "AMAZON.US_FIRST_NAME"
            }
          ],
          "samples": [
            " I go by {name}",
            " {name}",
            "my name is {name}",
            "you can call me {name}",
            "call me {name}"
          ]
        },
        {
          "name": "AMAZON.NavigateHomeIntent",
          "samples": []
        }
      ],
      "types": []
    }
  }
}
```

Step 2: Backend / AWS Lambda / Logic

After building your interaction model successfully, the next step is to select the **Code** tab which can be found at the top navigation menu in between the Build and Test tabs. We will begin with the **LaunchRequestHandler** highlighted below



1. Modify the code by changing the speechText to something like this:

   “'Hello my name is Shuri, what is your name?'”

2. Save your changes by clicking the save tab at the top right of the screen. Once it’s saved, click the **Deploy** tab (located next to the save tab)

Your **LaunchRequestHandler** should now look like this:

```
const LaunchRequestHandler = {
  canHandle(handlerInput) {
    return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
  },
  handle(handlerInput) {
    const speechText = "Hello my name is Shuri, what is your name?'?';
    return handlerInput.responseBuilder
      .speak(speechText)
      .reprompt(speechText)
      .getResponse(); } };
```

3. Next we will modify the HelloWorldIntentHandler *as seen below*.

- Change the *HelloWorldIntentHandler's* name to **HelloIntentHandler**. Next, we also need to change the intent name to **HelloIntent** because that is what we named it

4. Now we want to retrieve the user's response to our LaunchRequestHandler's question of *what's your name*. To do so, we need to retrieve the user's response , store it in a variable then use it with our speechText. To do so, add the following **bolded** code

Your **HelloIntentHandler** should now look like this:

```
const HelloIntentHandler = {
  canHandle(handlerInput) {
    return handlerInput.requestEnvelope.request.type === 'IntentRequest'
      && handlerInput.requestEnvelope.request.intent.name === 'HelloIntent';
  },
  handle(handlerInput) {
 let name = handlerInput.requestEnvelope.request.intent.slots.name.value;
const speechText = `<audio src='soundbank://soundlibrary/magic/amzn_sfx_fairy_melodic_chimes_01'/>  ${name},
welcome to wakanda`;
    return handlerInput.responseBuilder
      .speak(speechText)
      //.reprompt('add a reprompt if you want to keep the session open for the user to respond')
      .getResponse();
  }
};
```

5. Next scroll all the way down to modify the exports as seen below:

-the only change here is to ensure our **HelloIntentHandler** handler gets called

```
exports.handler = Alexa.SkillBuilders.custom()
  .addRequestHandlers(
    LaunchRequestHandler,
    HelloIntentHandler,
    HelpIntentHandler,
    CancelAndStopIntentHandler,
    SessionEndedRequestHandler,
    IntentReflectorHandler) // make sure IntentReflectorHandler is last so it doesn't override your custom intent handlers
  .addErrorHandlers(ErrorHandler) .lambda();
```

6. Save and deploy your AWS Lambda code. Then test to ensure everything works.

Step 3: Test your Alexa Skill & Challenge yourself

1. Access the **Alexa Simulator**, by selecting the **Test** link from the top navigation menu.
2. Enable Testing by activating the **Test is disabled for this skill** slider. It should be underneath the top navigation menu. Enabling should change it to read **Test is enabled for this skill**.
3. To validate that your skill is working as expected, invoke your skill from the **Alexa Simulator**. You can either type or click and hold the mic from the input box to use your voice.
    1. **Type** "Open" followed by the invocation name you gave your skill in For example, **"Open wakanda facts"**.
    2. **Use your voice** by clicking and holding the mic on the side panel and saying "Open" followed by the invocation name you gave your skill.
    3. **If you've forgotten the invocation name** for your skill, revisit the **Build** panel on the top navigation menu and select **Invocation** from the sidebar to review it.
4. Ensure your skill works the way that you designed it to.
5. Guess what? That's it for now but you can make it better. See below!!!

CHALLENGE - Feel free to ask for help if stuck working on the challenge

The expected behavior should be that you introduce yourself and the bot remembers your name and uses it to greet you with a random wakanda fact. Get it? No, Ask ???

Create an array of facts. **Create a random function** that will **randomly select a fact** from your array. Invoke the function in your **HelloIntentHandler** so that your skill can **surprise users with a random wakanda fact**

FINAL BACKEND CODE WITHOUT SOLUTION TO CHALLENGE

```
const Alexa = require('ask-sdk-core');
const LaunchRequestHandler = {
    canHandle(handlerInput) {
        return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
    },
    handle(handlerInput) {
        const speechText = 'Hello my name is Shuri, what is your name?';
        return handlerInput.responseBuilder
            .speak(speechText)
            .reprompt(speechText)
            .getResponse();
    }
};
const HelloIntentHandler = {
    canHandle(handlerInput) {
        return handlerInput.requestEnvelope.request.type === 'IntentRequest'
            && handlerInput.requestEnvelope.request.intent.name === 'HelloIntent';
    },
    handle(handlerInput) {
        let name = handlerInput.requestEnvelope.request.intent.slots.name.value;
        const speechText = `<audio src='soundbank://soundlibrary/magic/amzn_sfx_fairy_melodic_chimes_01'/> ${name}, welcome to
wakanda`;
        return handlerInput.responseBuilder
            .speak(speechText)
            .getResponse();
    }
};
const HelpIntentHandler = {
    canHandle(handlerInput) {
        return handlerInput.requestEnvelope.request.type === 'IntentRequest'
            && handlerInput.requestEnvelope.request.intent.name === 'AMAZON.HelpIntent';
    },
    handle(handlerInput) {
        const speechText = 'You can say hello to me! How can I help?';

        return handlerInput.responseBuilder
            .speak(speechText)
            .reprompt(speechText)
            .getResponse();
    }
};
```

```
const CancelAndStopIntentHandler = {
    canHandle(handlerInput) {
        return handlerInput.requestEnvelope.request.type === 'IntentRequest'
            && (handlerInput.requestEnvelope.request.intent.name === 'AMAZON.CancelIntent'
                || handlerInput.requestEnvelope.request.intent.name === 'AMAZON.StopIntent');
    },
    handle(handlerInput) {
        const speechText = 'Goodbye!';
        return handlerInput.responseBuilder
            .speak(speechText)
            .getResponse();
    }
};
const SessionEndedRequestHandler = {
    canHandle(handlerInput) {
        return handlerInput.requestEnvelope.request.type === 'SessionEndedRequest';
    },
    handle(handlerInput) {
        return handlerInput.responseBuilder.getResponse();
    }
};
const IntentReflectorHandler = {
    canHandle(handlerInput) {
        return handlerInput.requestEnvelope.request.type === 'IntentRequest';
    },
    handle(handlerInput) {
        const intentName = handlerInput.requestEnvelope.request.intent.name;
        const speechText = `You just triggered ${intentName}`;
        return handlerInput.responseBuilder
            .speak(speechText)
            .getResponse();
    }
};

const ErrorHandler = {
    canHandle() {
        return true;
    },
    handle(handlerInput, error) {
        console.log(`~~~~ Error handled: ${error.message}`);
        const speechText = `Sorry, I couldn't understand what you said. Please try again.`;
        return handlerInput.responseBuilder
            .speak(speechText)
            .reprompt(speechText)
      .getResponse();  } };
```

```
exports.handler = Alexa.SkillBuilders.custom()
    .addRequestHandlers(
        LaunchRequestHandler,
        HelloIntentHandler,
        HelpIntentHandler,
        CancelAndStopIntentHandler,
        SessionEndedRequestHandler,
        IntentReflectorHandler) // make sure IntentReflectorHandler is last so it doesn't override your custom intent handlers
    .addErrorHandlers(
        ErrorHandler)
    .lambda();
```