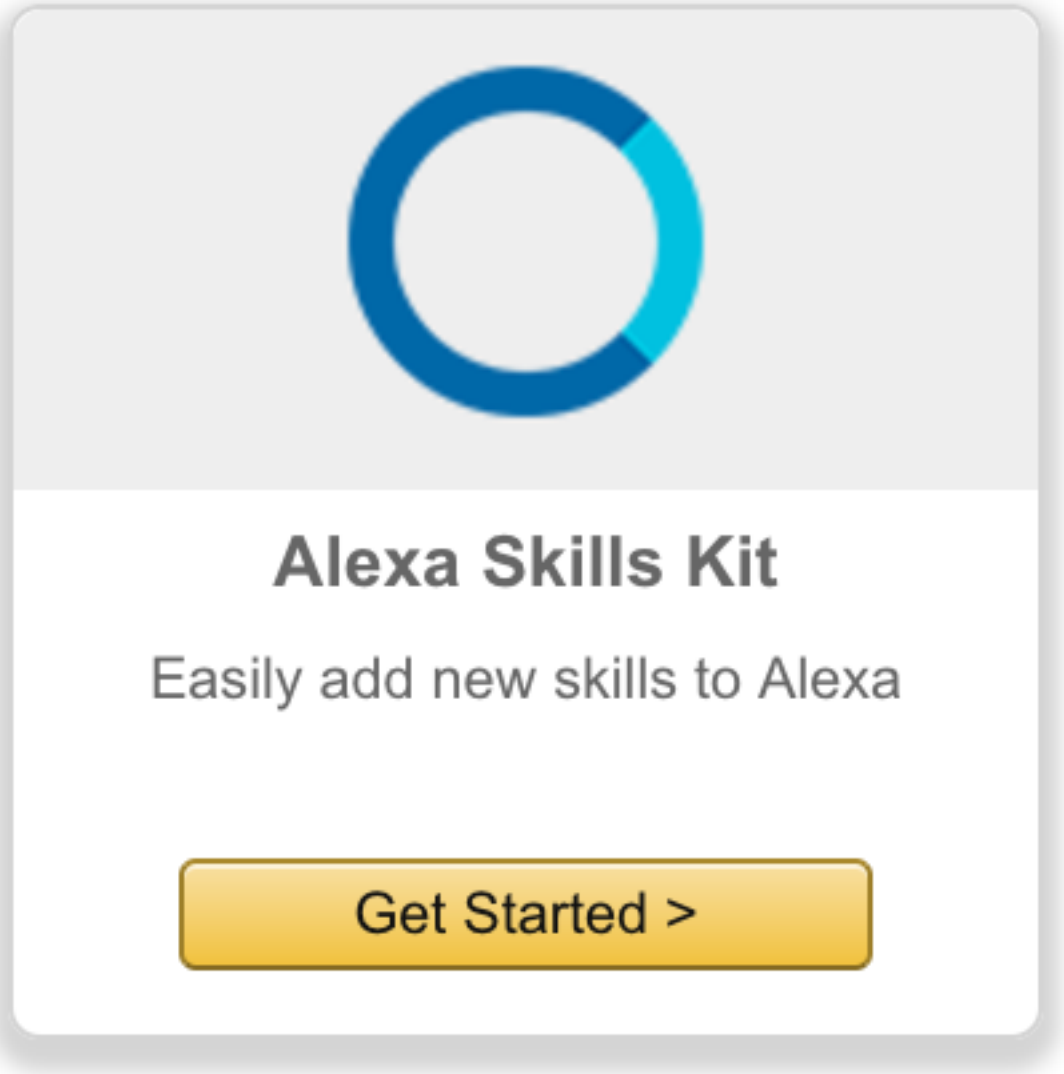
Alexa 1: Hello World

1. Amazon-side setup

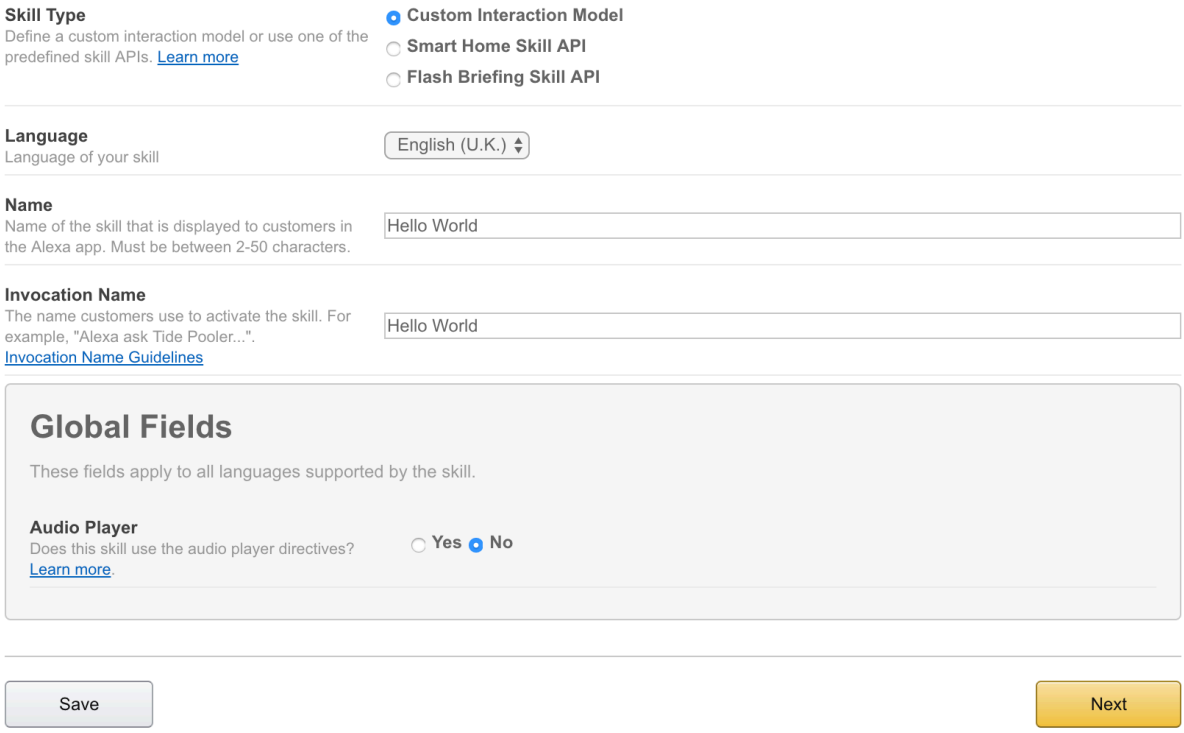
Our first step is to set up the Skill on Amazon.

- Sign in to the Amazon Alexa Developer Dashboard here
- Get Started with the Alexa Skills Kit:



- ‘Add a New Skill’:

- Set up the app:
  - Language (English (U.K.))
  - Name ('Hello World')
  - Invocation Name ('Hello World')



- Make Intent Schema:

The Intent Schema lists all the possible requests Amazon can make to your application.

```
{
  "intents": [
    {
      "intent": "HelloWorld"
    }
  ]
```

```
}
```

The `intent` *property gives the name of the intent.*

- Make Utterances:

Utterances map Intents to phrases spoken by the user.

```
HelloWorld say hello world
```

Utterances are written in the form `IntentName utterance`.

2. Local Tunnelled Development Environment setup

Our second step is to set up our local Ruby application, ready to receive encrypted requests from Amazon's servers (i.e. HTTP requests over SSL, or 'HTTPS' requests).

We will walk through setting up a Ruby server using Sinatra, running locally, and capable of receiving HTTPS requests through a Tunnel.

Alternatively, you could set up a **remote** development server using Heroku (with Heroku SSL), Amazon Elastic Beanstalk (with a self-signed SSL certificate), or any other method you can think of.

We're going to use ngrok to Tunnel to a local development server.

- Set up a Sinatra Application with a single route, `/`:
  - Make the directory with `mkdir hello_world_app`
  - Head into the directory with `cd hello_world_app`

- Set up a Ruby application with `bundle init` (you may need to install Bundler with `gem install bundler` first)
- Add the Sinatra gem to your Gemfile, by adding the line `gem 'sinatra'`
- Install Sinatra to your project using `bundle install`
- Create a server file with `touch server.rb`
- For now, create a single `POST` route, `'/'`, that prints out the request body we are going to receive from Amazon:

```
require 'sinatra'

post '/' do
 p request.body.read
end
```

- Download the appropriate ngrok package for your Operating System from the ngrok downloads page
- Unzip the package and transfer the executable to your `hello_world_app` directory
- Start ngrok using `./ngrok http 4567`
- Copy to the clipboard (`command-C`) the URL starting 'https' and ending '.ngrok.io' from your ngrok Terminal
- In a second Terminal, start your Sinatra application using `ruby server.rb`

3. Linking Amazon to our Endpoint

Our third step is to link the Skill we set up on Amazon (1) with the Tunnel Endpoint (2) so our Skill can send requests to our local application.

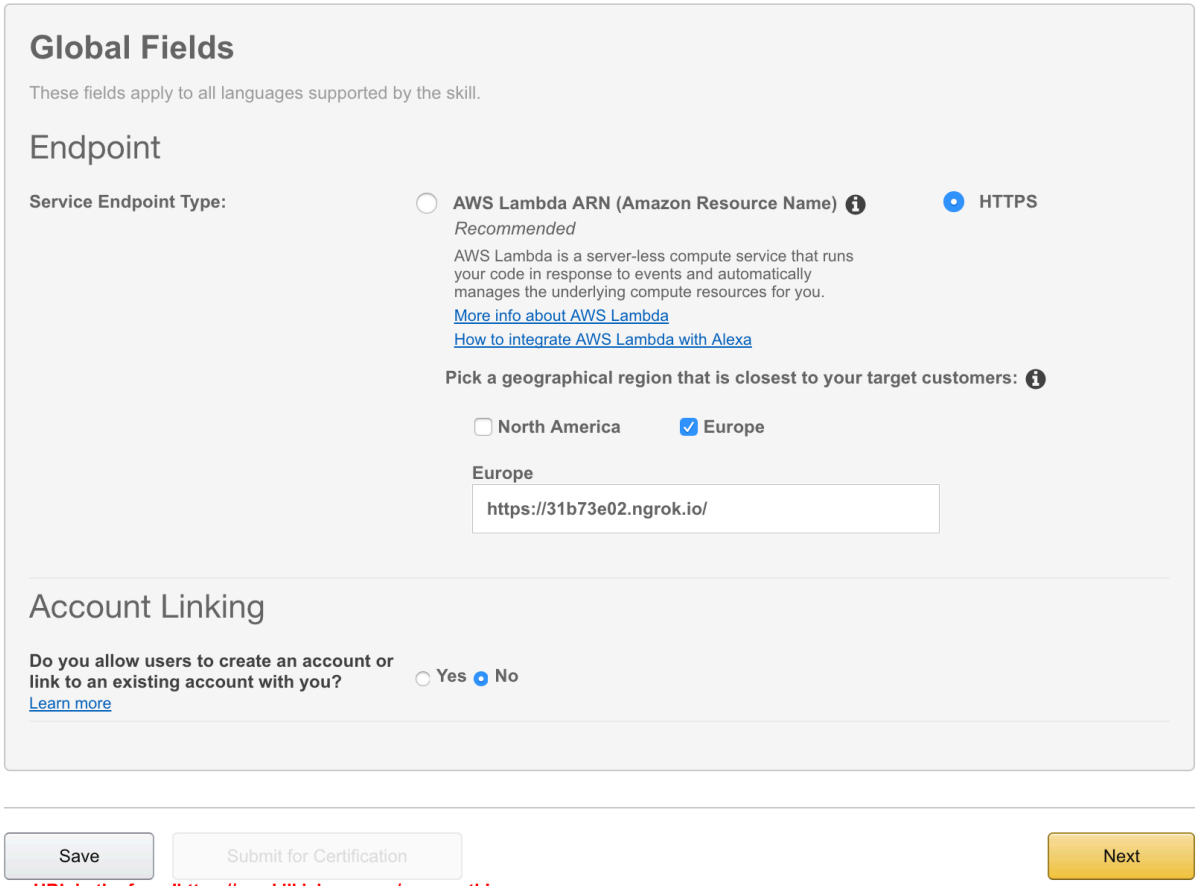
- Head back to your Alexa Skill (for which you just entered Intents and Utterances).
- Hit Next, then set up the Endpoint:

When Amazon invokes an Intent, Amazon sends a `POST` *request to the specified Endpoint (web address).*

- Use HTTPS, not Lambda (no Ruby on Lambda)
    - Geographical Region: Europe
    - Paste the Endpoint to your application into the text input field

If using ngrok, your Endpoint is the URL you copied, starting with 'https' and ending with '.ngrok.io'.

- No Account Linking



- Hit next, then set up the SSL Certificate:

Amazon's servers will only send requests to HTTPS web addresses, which need to be signed with an SSL Certificate.

- If you used ngrok to set up a Tunnel, select 'My development endpoint is a sub-domain of a domain that has a wildcard certificate from a certificate authority'.
    - Hit 'next'
- Use the Service Simulator to test that the `say hello world` utterance causes Amazon to send an Intent Request to your local application, and

observe that the request body printed to the command-line matches the JSON request sent in the Service Simulator.

The Service Simulator allows you to try out utterances. Once you've written an utterance into the Simulator, you can send test requests to the application endpoint you defined. You can see your application's response to each request that you send.

You've now hooked up your local development environment to an Alexa Skill!

4. Responding to Alexa Requests

Now we have built an Alexa development Skill (1), built a local development server with an endpoint tunnelled via HTTPS (2), and can make requests from Amazon to our local development server through that endpoint (3).

Our final step is to construct a response from our endpoint such that Amazon can interpret the response to make Alexa say 'Hello World' to us.

- `require 'json'` at the top of `server.rb`
- Replace the body of our Sinatra POST '/' route with the smallest possible response:

```
{
  version: "1.0",
  response: {
    outputSpeech: {
      type: "PlainText",
      text: "Hello World"
    }
  }
```

```
}
```

- `version` (string): required. Allows you to version your responses.
- `response` (object): required. Tells Alexa how to respond: including speech, cards, and prompts for more information.
  - `outputSpeech` (object). Tells Alexa what to say.
    - `type` (string) required. Tells Alexa to use Plain Text speech, where Alexa will guess pronunciation, or Speech Synthesis Markup Language (SSML), where you can specify pronunciation very tightly.
      - **EXTRA CREDIT**: Change the response to use custom pronunciation using SSML.
    - `text` (string) required. Tells Alexa exactly what to respond with.
      - **EXTRA CREDIT**: Play around with this response, restarting the server and sending an Intent Request from the Service Simulator each time.
- Restart the server, and test out the new response in the Service Simulator.

If you would like to try your new Hello World skill out live, hit 'Next' to complete the Publishing Information, and 'Save'!