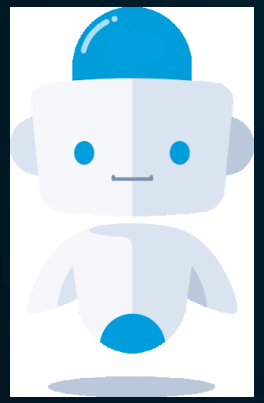
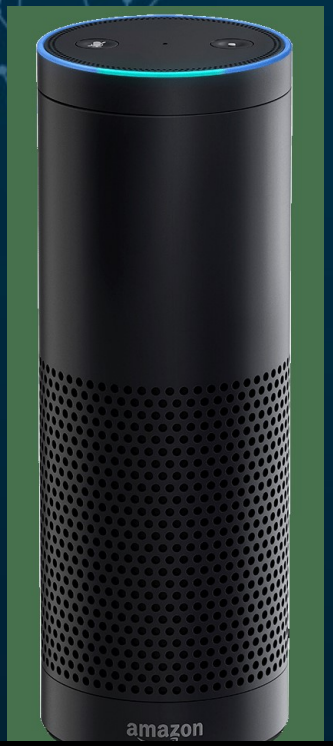


Alexa Skills-

Towards Building a Customized Personal Assistant

BY:

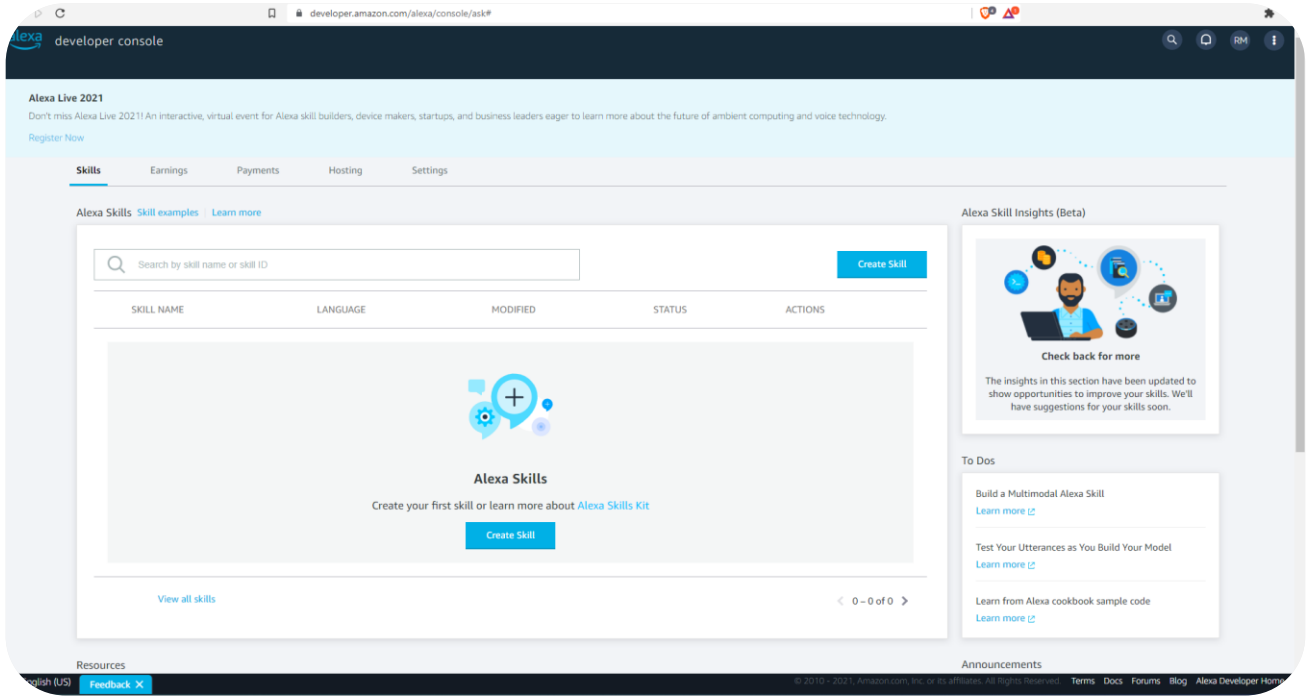
- REDA MASTOURI



amazon alexa

Getting Started

- Creating a developer account on Alexa Developer Console Portal:
- https://developer.amazon.com/alexa/console/ask#
  - - Skill Configuration
  - - Intents
  - - Slots
  - - Utterances
- Skill Testing:
  - - Lambda Test Events
  - - Developer Portal Skill Simulator
  - - Alexa App
  - - Amazon Echo -Hardware



What is Alexa Skills?

- Alexa is a voice service from Amazon.
- Alexa powers devices like Echo Show, Echo Spot, Echo Dot etc.
- The most important feature of the Alexa is off course it's hands-free voice control, which lets users ask questions without touching the device.
- Alexa listen to the voice commands and respond with appropriate responses to fulfill.
- Alexa comes with many capabilities like playing music, reading the news, creating your shopping list, setting alarms, know about the weather etc.
- Alexa comes with many capabilities like playing music, reading the news, creating your shopping list, setting alarms, know about the weather etc.

Alexa Features

Smart home

Simplify your life by automating and securing your home with smart devices.

Productivity

Alarms, timers, calendars, email Alexa helps with the details so you can stay focused.

Shopping

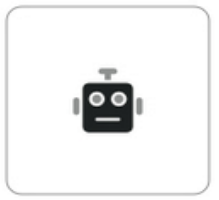
Use your voice to manage your shopping list, reorder household items, and track orders.

Entertainment

Listen to music, radio, podcasts, and audiobooks or watch movies, TV, and videos.

Care Hub

This free feature in the Alexa app gives you new ways to check in on loved ones.



Kids and Family

Enable parental controls and discover features that help kids learn and grow.

Information

Ask Alexa questions, get language translations, general knowledge and more.

Kitchen

Cook along with recipes, get unit conversions, set timers, and multitask with ease.

Communications

Communicate with friends and family with voice or video calling, Drop In and Announcements.

News

Alexa can give you news when you need it from your favorite providers.

Routines

Set up custom shortcuts and make multiple things happen at the same time.

Fun and games

Download new Alexa game skills to play by yourself or with family and friends.

Multi-room audio

See how Alexa can bring music and entertainment to your entire home.

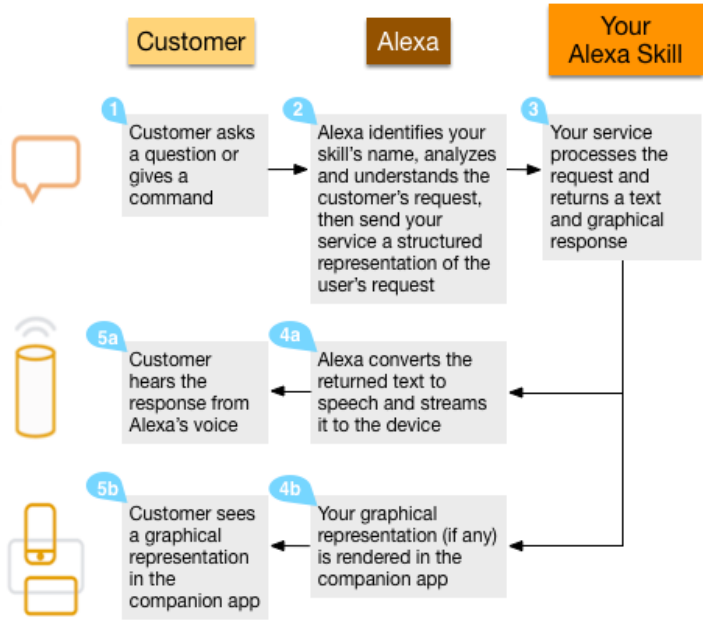
Bedroom

Alexa helps you rest easy and always wake up on the right side of the bed.

Photos

Showcase your best photos, share favorites with your Alexa contacts, and even take pictures.

How Alexa Skills Work? (Part I)



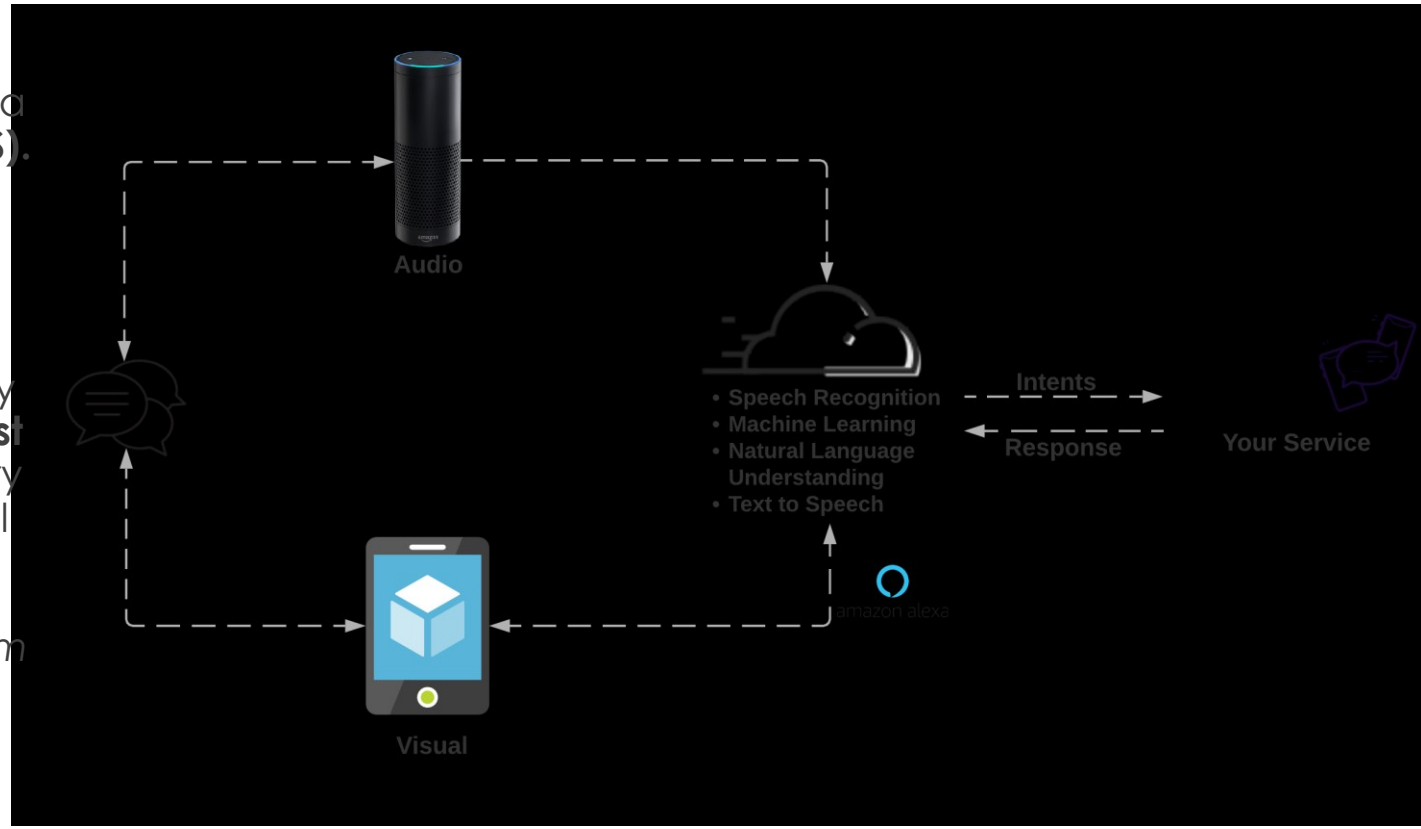
- **Wake word** When users say 'Alexa' which wakes up the device. The wake word put the Alexa into the listening mode and ready to take instructions from users. Of Course you can change the wake word via Alexa assistant mobile app.

- **Invocation name** Invocation name is the keyword used to trigger a specific skill. It just likes the name of that skill which we used to identify. Users can combine the invocation name with an action, command or question. All the custom skills need an invocation name to trigger or start it.

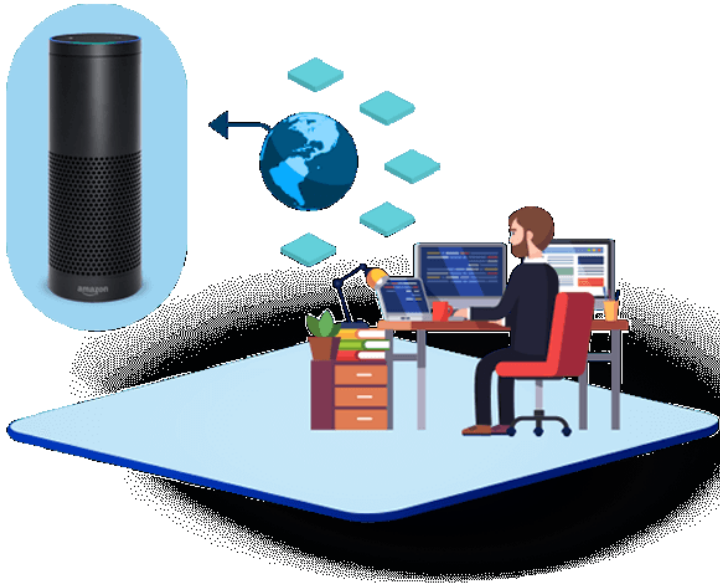
- **Utterance** In the above example, the keyword 'Taurus' is an utterance. Utterances are phrases the users will use when making a request to Alexa. Alexa identifies the user's intent from the given utterance and responds accordingly. So basically the utterance decide what user want Alexa to perform.

How Alexa Skills Work? (Part II)

- Alexa enabled devices sends the user's instruction to a cloud-based service called **Alexa Voice Service (AVS)**. Think the Alexa Voice Service as the brain of Alexa enabled devices and perform all the complex operations such as **Automatic Speech Recognition (ASR)** and **Natural Language Understanding (NLU).**
- Alexa Voice Service process the response and identify the user's **intent**, then it makes the web service **request** to third party server if needed. This request is necessary if the Skills needs to fetch information from an external service.
For example, Flash briefing skills need to interact with news website/server to fetch the latest headlines. From there, the Alexa service sends the response back to the Alexa and read it to the users.



How does FD benefit from developing an Alexa Skill? (Part I)



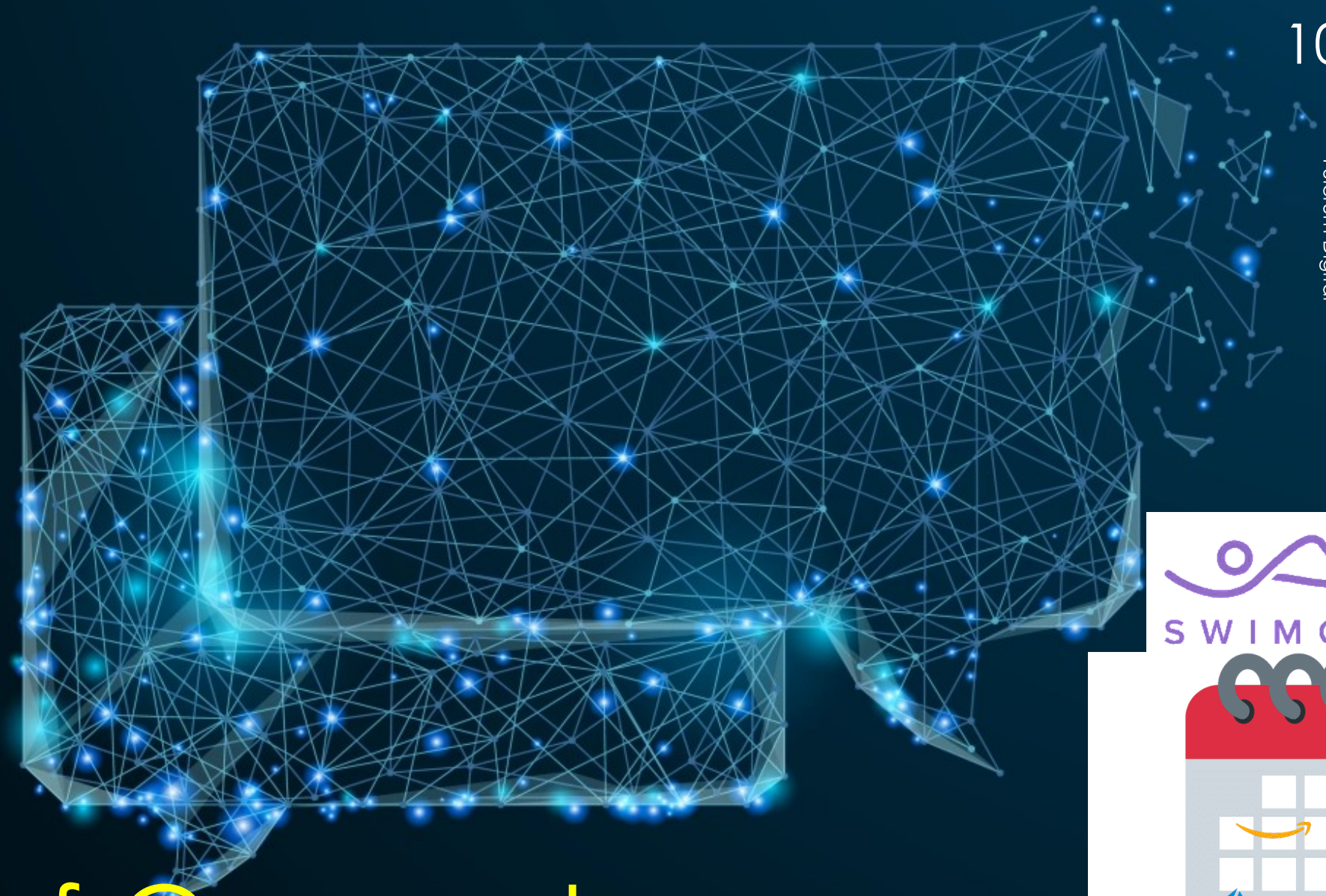
- Digital assistants are quickly becoming a part of our lives. Amazon Alexa, Apple's HomePod, Google Assistant, Microsoft Cortana, Samsung's Bixby, and China's Chumenwenwen and Xiaoice are just a few to mention.
- Virtual assistants are fast becoming commonplace in the home through products like these, with more than 100 million Alexa-enabled devices having been sold till date.
- Amazon has extended Alexa's API so developers can build Alexa Skills, which are essentially third-party apps that take advantage of Alexa's amazing capabilities.

How does FD benefit from developing an Alexa Skill? (Part II)

- The Alexa Skills marketplace is rapidly growing, from 1,000 to 15,000 in just one year in the Alexa Skill Store, part of the reason that analysts estimate Alexa to become a $10 billion industry by 2020. For businesses, it's time to get on board now to build with Alexa as it grows.
- And there will be fight over talent skilled in Alexa Skills and other voice-assistant technologies.
- With the expected meteoric growth, voice could become the primary way people interact with the internet in as little as 10–15 years.
- The unprecedented opportunities are appealing not only to the owners of smart devices.
- Having taken residence in millions of homes, Amazon and Google are paving the way for other brands that offer entertainment, information, goods, and services.

How does FD benefit from developing an Alexa Skill? (Part III)

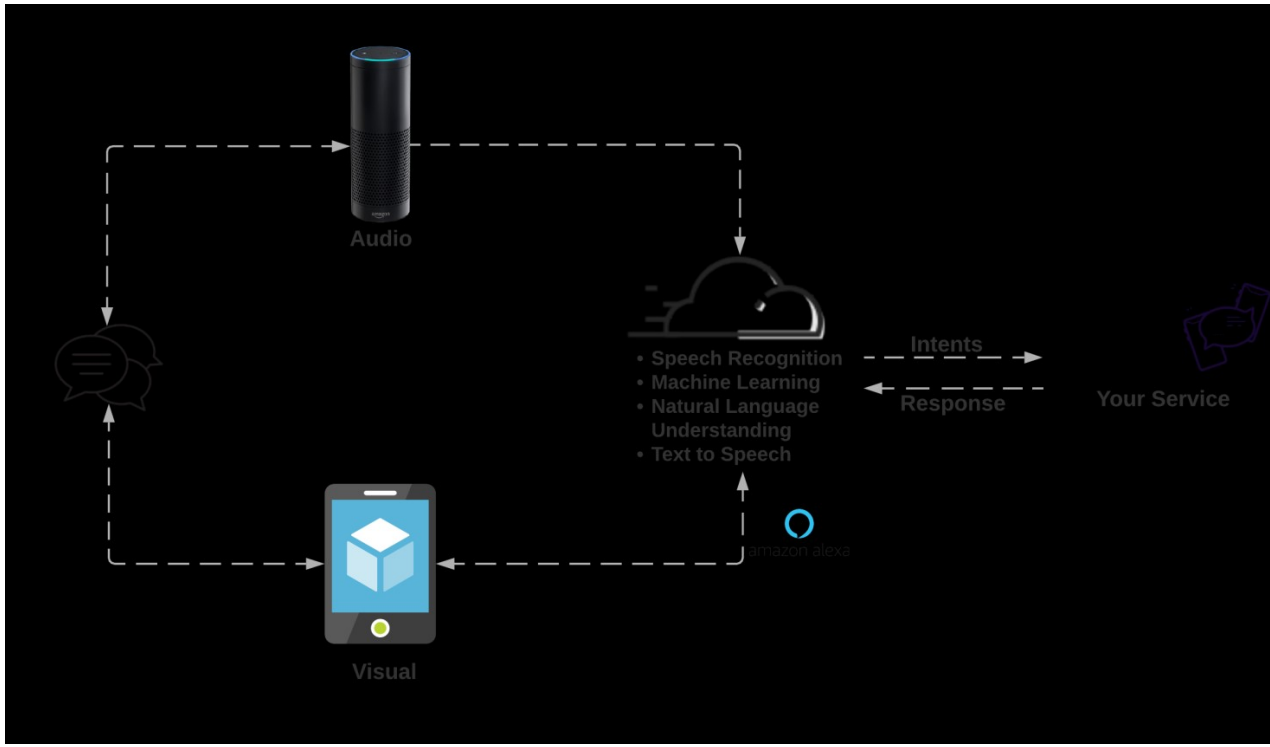
- The first advantage is the vast market. Amazon Echo family commands almost 72% market share of current U.S. smart speakers. Loyal users will likely fill their houses with compatible devices. With a legion of connected fridges, door locks, thermostats, lamps, and other appliances that may use Alexa Voice Service API, the potential of Alexa brand market penetration is infinite.
- Your investment in the technology is safe. With Alexa controlling such a huge percentage of the marketplace already, it's likely to maintain that lead for some time. Choosing to develop an Alexa Skills (rather than for another voice platform) simplifies development from both the product strategy and technical development standpoints. Keeping things simple at first means you don't need to develop for 3–4 platforms at one time. You can develop for Siri, Cortana and the Google Assistant later as necessitated by the market.
- Promoting more efficient sorting and accurate matching in the marketplace, AI platforms will be simultaneously winning people's trust and loyalty.
- At the moment, Amazon has tremendous incentive to help companies succeed with Alexa Skills and offers strong customer support for Skills development. This level of support will be likely to dissipate as the market grows.
- The Alexa Skills platform will grow more complex. As the technology matures, so will expectations for features and functionality. Getting in early and developing a basic Skill gives you the opportunity to learn without much consequence and iterate as you mature. Right now, you're not expected to have a completely mature Skill. There's time to evolve as Alexa's technology evolves. This will change, just as it did for mobile apps.
- With voice being relatively new, it's an incredible opportunity to beat your competitors to market with a compelling consumer experience that can differentiate you in your industry. Getting ahead of the curve means you won't be scrambling to figure things out after your competitors have already done so.
- As an added bonus, the companies that are first to market will be acknowledged as innovators and may receive a substantial amount of positive media attention.



Demo:
Proof-of-Concept

Starting to build a Schedulify Skill:
Checklist

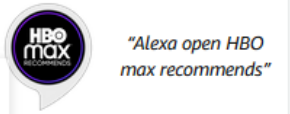
- Personal Assistant Bot:
  - Check financial portfolio
  - Manage servers
  - Send voice messages
  - Read out emails
  - Tell monthly goals
- Build a publicly accessible Alexa Custom Skill & Monetize:
  - Schedulify

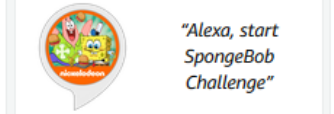


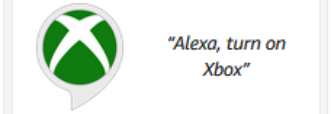
Starting to build a Schedulify Skill

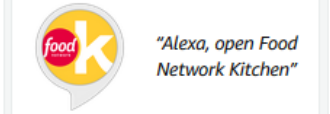
Example of available skills

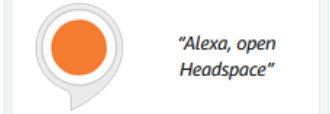
Editor's Picks



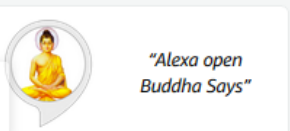


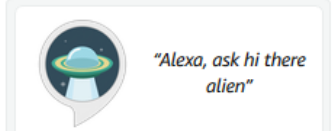


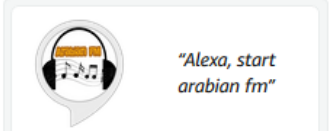


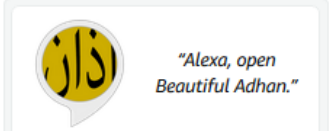


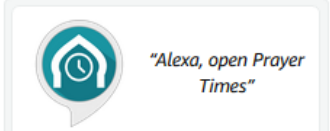
"Alexa, recommend some skills."



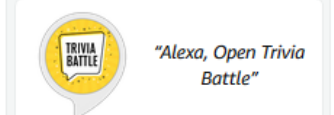


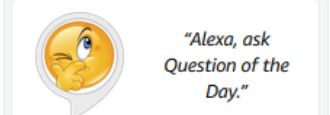


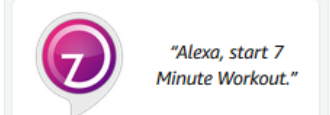


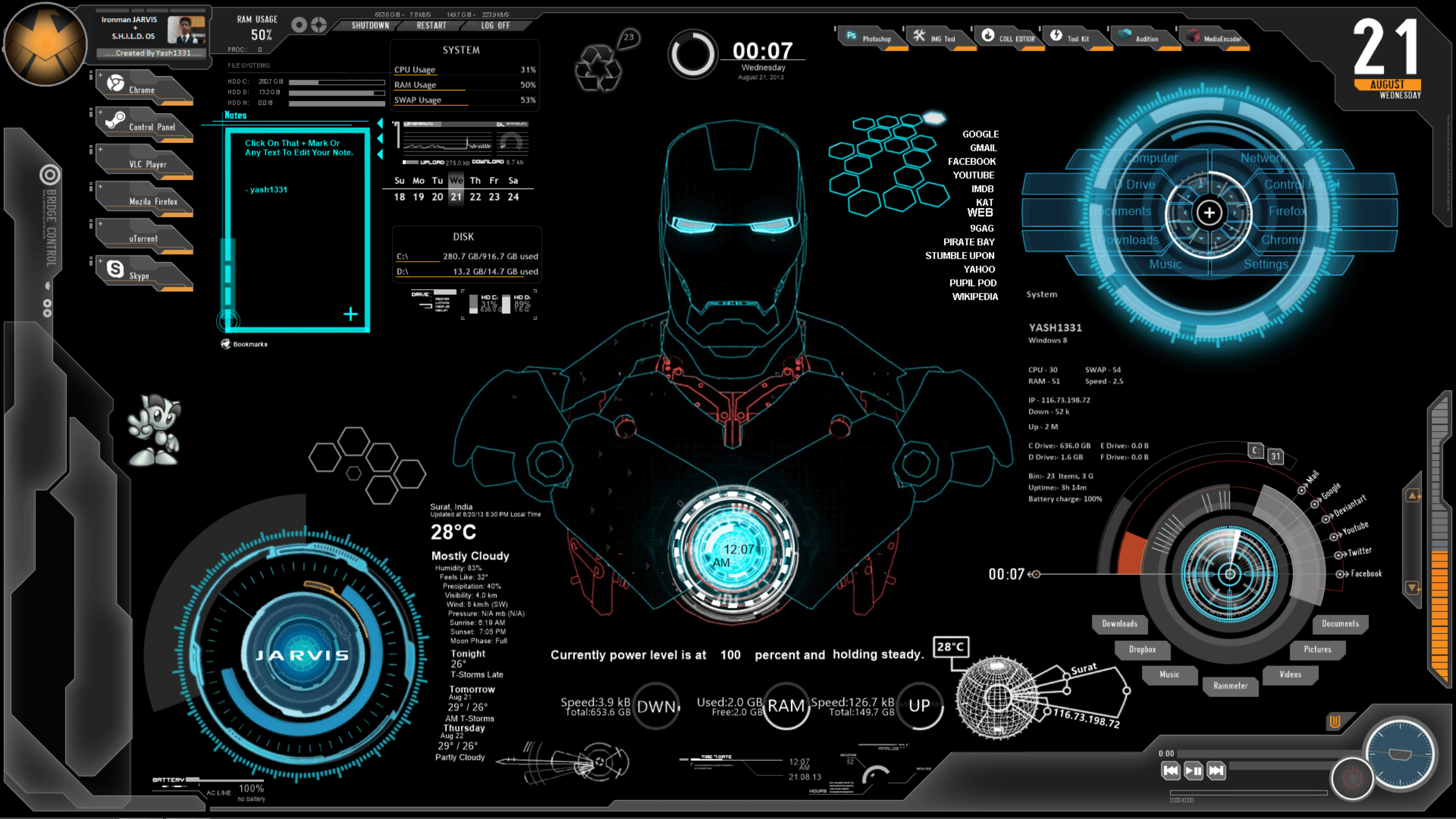


Premium Skills





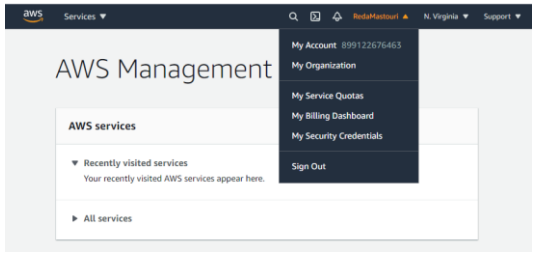



Ironman JARVIS
+
S.H.I.L.D. OS
.....Created By Yash1331.....
RAM USAGE
50%
SHUTDOWN
RESTART
LOG OFF
SYSTEM
CPU Usage 31%
RAM Usage 50%
SWAP Usage 53%
Chrome
Control Panel
VLC Player
Mozila Firefox
uTorrent
Skype
Notes
Click On That + Mark Or Any Text To Edit Your Note.
- yash1331
Bookmarks
00:07
Wednesday
August 21, 2013
Photoshop
IMG Tool
COLL EDITOR
Tool Kit
Audition
MediaEncoder
21
AUGUST
WEDNESDAY
Su Mo Tu We Th Fr Sa
18 19 20 21 22 23 24
DISK
C:\ 280.7 GB/916.7 GB used
D:\ 13.2 GB/14.7 GB used
GOOGLE
GMAIL
FACEBOOK
YOUTUBE
IMDB
KAT
WEB
9GAG
PIRATE BAY
STUMBLE UPON
YAHOO
PUPIL POD
WIKIPEDIA
Computer
Network
D Drive
Control Panel
Documents
Firefox
Downloads
Chrome
Music
Settings
System
YASH1331
Windows 8
CPU - 30
SWAP - 54
RAM - 51
Speed - 2.5
IP - 116.73.198.72
Down - 52 k
Up - 2 M
C Drive:- 636.0 GB
E Drive:- 0.0 B
D Drive:- 1.6 GB
F Drive:- 0.0 B
Bin:- 23 Items, 3 G
Uptime:- 3h 14m
Battery charge- 100%
Mail
Google
Deviantart
Youtube
Twitter
Facebook
00:07
Downloads
Documents
Dropbox
Pictures
Music
Videos
Rainmeter
12:07 AM
JARVIS
Surat, India
Updated at 8/20/13 8:30 PM Local Time
28°C
Mostly Cloudy
Humidity: 83%
Feels Like: 32°
Precipitation: 40%
Visibility: 4.0 km
Wind: 5 km/h (SW)
Pressure: N/A mb (N/A)
Sunrise: 6:19 AM
Sunset: 7:05 PM
Moon Phase: Full
Tonight
26°
T-Storms Late
Tomorrow
Aug 21
29° / 26°
AM T-Storms
Thursday
Aug 22
29° / 26°
Partly Cloudy
Currently power level is at 100 percent and holding steady.
Speed:3.9 kB
Total:653.6 GB
DWN
Used:2.0 GB
Free:2.0 GB
RAM
Speed:126.7 kB
Total:149.7 GB
UP
28°C
Surat
116.73.198.72
BATTERY
AC LINE
100%
no battery
12:07 AM
21.08.13
0:00

Working Environment:
Cloud Computing

Amazon S3

Amazon CloudWatch



JavaScript Basics: *Source-Code*

- https://github.com/technotablet/alexa-course-custom-skill.git
- GitHub Repository https://github.com/technotablet/alexa-course-custom-skill
- Checklist – Interactive, Web Based https://vivekk.com/teach-alexa-cs
- Amazon S3 REST API Introduction http://docs.aws.amazon.com/AmazonS3/latest/API/
- Alexa Built-in Slot Type Reference https://developer.amazon.com/docs/custom-skills/slot-type-reference.html
- Alexa Built-in Intents https://developer.amazon.com/docs/custom-skills/standard-built-in-intents.html
- Alexa Skills Kit (ASK) for Node.js (:tell, :ask, :tellWithCard etc. Response Methods) https://github.com/alexa/alexa-skills-kit-
- sdk-for-nodejs Google Maps Directions API Documentation https://developers.google.com/maps/documentation/directions/intro
- Time zones Syntax https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

JavaScript Basics:

JS101- Promises and Await Expressions in Async Functions- 2

- Changing to asynchronous setup, where function a is executed within an anonymous func tion through setTimout with 3000 ms delay

Welcome | promises.html U

JS_Fund > promises.html > html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Promises-Reda</title>
</head>
<body>
    <script>

        var d = new Date(); // current date
        var n = 3000 + d.getTime(); //Add 3 seconds to current date

        function a() {
            while (new Date() < n ) { } // Do nothing if current time is < advanced time 'n'
            console.log("Function A compltes in "+ (new Date() - d) + "ms / " + new Date());
        }

        function b() {
            console.log("Function B completes. " +  new Date());
        }

        console.log("Starts at " + d);
        // Calling the two function a and then b
        // Changing to asynchronous setup, where function a is executed within an anonymous function through setTimout with 3000 ms delay
        setTimeout(function(){
                        a();
                        }, 3000);
        b();

    </script>


</body>
</html>
```

Inspector | Console | Debugger | Network | Style Editor | Performance

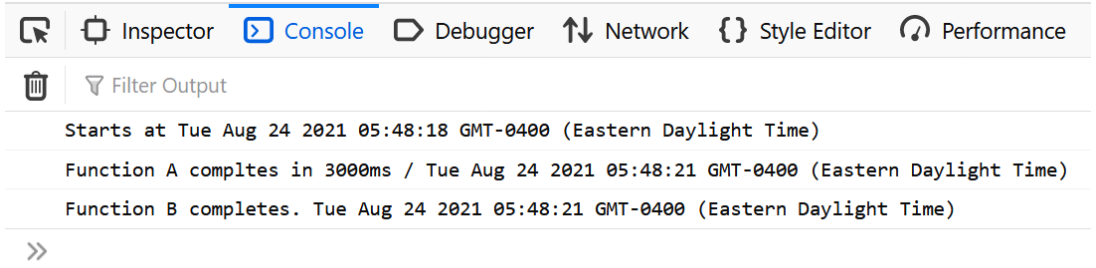
Filter Output

```
Starts at Tue Aug 24 2021 05:54:35 GMT-0400 (Eastern Daylight Time)
Function B completes. Tue Aug 24 2021 05:54:36 GMT-0400 (Eastern Daylight Time)
Function A compltes in 3011ms / Tue Aug 24 2021 05:54:39 GMT-0400 (Eastern Daylight Time)
```

JavaScript Basics:

JS101- Promises and Await Expressions in Async Functions- 2

- setTimeout() is one of the earliest asynchronous methods in JS.
  - Pros:
    - Execute multiple functions in parallel
    - Non-blocking code
  - Cons:
    - Functions might be dependent on each other
    - One function's output is another function's imput

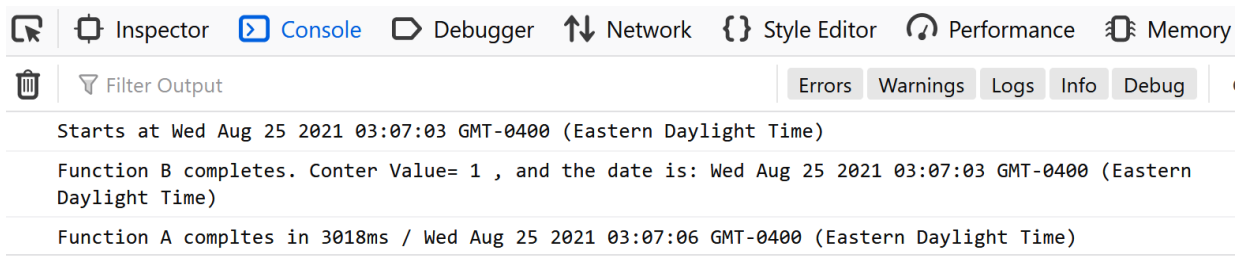



```
Welcome
promises.html U
JS_Fund > promises.html > html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Promises-Reda</title>
</head>
<body>
    <script>

        var d = new Date();
        var n = 3000 + d.getTime(); //Add 3 seconds to current date

        function a() {
            while (new Date() < n ) { } // Do nothing if current time is < advanced time 'n'
            console.log("Function A compltes in "+ (new Date() - d) + "ms / " + new Date());
        }

        function b() {
            console.log("Function B completes. " +  new Date());
        }

        console.log("Starts at " + d);
        a();
        b();

    </script>


</body>
</html>
```

JavaScript Basics:

JS101- Promises and Await Expressions in Async Functions- 3

- Removing this loop: while (new Date() < n) { } // Do nothing if current time is < advanced time 'n'
- and adding an asynchronous method setTimeout of 3s and and print a counter as 2




```
JS_Fund > promises.html > html > body > script
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Promises-Reda</title>
</head>
<body>
    <script>

        var d = new Date(); // current date
        var n = 3000 + d.getTime(); //Add 3 seconds to current date

        // Introducing a variable called couter
        // and assign it a value of 1 by default
        var counter = 1;

        function a() {
            // Removing this loop: while (new Date() < n ) { } // Do nothing if current time is < advanced time 'n'
            // and adding an asynchronous method setTimeout of 3s and  and print a counter as 2
            setTimeout(function() {
                console.log("Function A compltes in "+ (new Date() - d) + "ms / " + new Date());
                counter = 2;
            }, 3000);

        }

        function b() {
            //
            console.log("Function B completes. Conter Value= " + counter + " , and the date is: " + new Date());
        }

        console.log("Starts at " + d);
        a();
        b();

    </script>


</body>
</html>
```

JavaScript Basics:

JS101- Promises and Await Expressions in Async Functions- 4

- A Callback is a function which is passed to another function as an argument
- E.g. An anonymous function as an argument to setTimeout function
- It let the code executes in a linear way

```
<script>
    function outerFunction(param) {
        var x = "Some Info";
        param(x);
    }

    function callbackFunction(data) {
        console.log(data);
    }

    outerFunction(callbackFunction);
</script>
```

```
setTimeout(
  function () {
    console.log("It works!");
  },
  3000
);
```

Inspector | Console | Debugger | Network | Style Editor | Performance | Memory

Filter Output | Errors | Warnings | Logs | Info | Debug

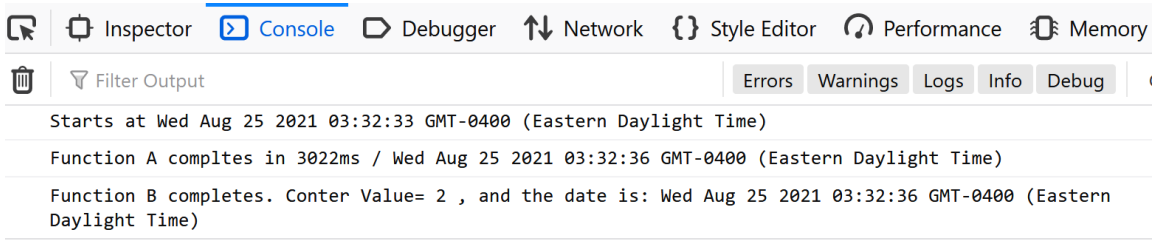
```
Starts at Wed Aug 25 2021 03:24:50 GMT-0400 (Eastern Daylight Time)
Function A compltes in 3017ms / Wed Aug 25 2021 03:24:53 GMT-0400 (Eastern Daylight Time)
Function B completes. Conter Value= 2 , and the date is: Wed Aug 25 2021 03:24:53 GMT-0400 (Eastern Daylight Time)
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Promises-Reda</title>
</head>
<body>
    <script>

        var d = new Date(); // current date
        var n = 3000 + d.getTime(); //Add 3 seconds to current date

        // Introducing a variable called couter
        // and assign it a value of 1 by default
        var counter = 1;

        function a(newfunc) { //3- the argument newfunc is going to be the callback function
            // 1 - We will define an argument to function a() as a newfunc
            setTimeout(function() {
                console.log("Function A compltes in "+ (new Date() - d) + "ms / " + new Date());
                counter = 2;
                //2 - call newfunc after setting counter at 2
                newfunc();
            }, 3000);

        }

        function b() {
            //
            console.log("Function B completes. Conter Value= " + counter + " , and the date is: " + new Date());
        }

        console.log("Starts at " + d);
        // 4 - When calling the function a we will add an anomymous function to it where it executes the function b()
        a(function(){ b();} ); //5- this essentially means that I assigned an anomymouse function to the variable newfunc
        // in the argement of function a()

        //0 - Commenting out calling the function B
        //b();

    </script>


</body>
</html>
```

JavaScript Basics:

JS101- Promises and Await Expressions in Async Functions - 5

- Using asynchronous function in sequence

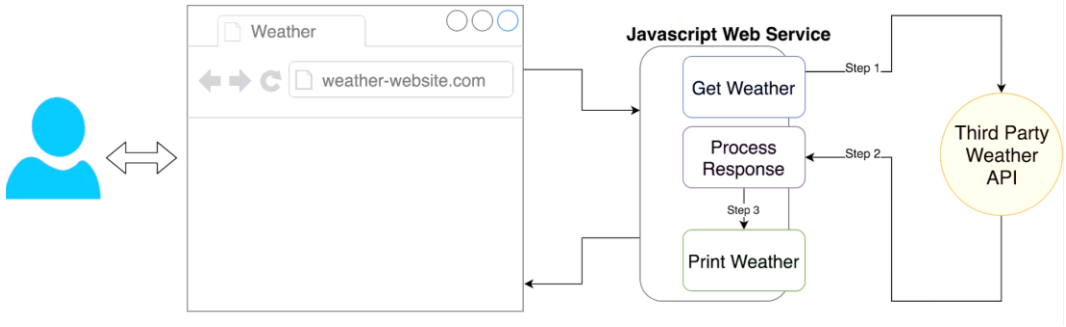



```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Promises-Reda</title>
</head>
<body>
    <script>

        var d = new Date(); // current date
        var n = 3000 + d.getTime(); //Add 3 seconds to current date

        // Introducing a variable called couter
        // and assign it a value of 1 by default
        var counter = 1;

        function a(newfunc) {
            setTimeout(function() {
                console.log("Function A compltes in "+ (new Date() - d) + "ms / " + new Date());
                counter = 2;
                newfunc();
            }, 3000);

        }

        function b() {
            //
            console.log("Function B completes. Conter Value= " + counter + " , and the date is: " + new Date());
        }

        console.log("Starts at " + d);
        // Calling b within a
        a(b); //Special: Whenever I call function a, I can do some activities, and I can pass any function
        // as an argument to fucntion a(),; and since b() doesn't accept any further argument, I can right like a(b)

    </script>
</body>
</html>
```

JavaScript Basics:
Callback

- Primarily used in Node.js 6.10 and below for Lambda
- Establish connections to independent interfaces such as Databases or APIs
- On response, process further (dependent functions)
  - If there are a large number of functions, dependent of each other, then we possibly get into something called Callback Hell (Because we need to nest the dependent functions, within a function). It can be very confusing and messy.

```
a(function (resultsFromA) {
    b(resultsFromA, function (resultsFromB) {
        c(resultsFromB, function (resultsFromC) {
            d(resultsFromC, function (resultsFromD) {
                console.log(resultsFromD);
            })
        })
    })
});
```



JavaScript Basics:
Promises

- Promise is a most sophisticated solution to implementing asynchronous code in JavaScript
- It currently provides 3 methods
  - .then() : handles success and can manage both success and failure scenarios
  - .catch(): handles failure and meant for errors
  - .finally(): handles completion and meant for the tasks that should execute no matter what the result is (Executes irrespective of success or error)

```
function testBoolean() {
    return new Promise(function (resolve, reject) {
        if (Math.random() >= 0.5) {
            resolve('success');
        }
        else {
            reject('failed');
        }
    });
}

testBoolean()
    .then()     // handle success
    .catch()    // handle failure
    .finally()  // handle completion;
```

JavaScript Basics:
Promises 2

- Promise is a most sophisticated solution to implementing asynchronous code in JavaScript
- We can also use a chained promises where next function executes if the previous one was successful.

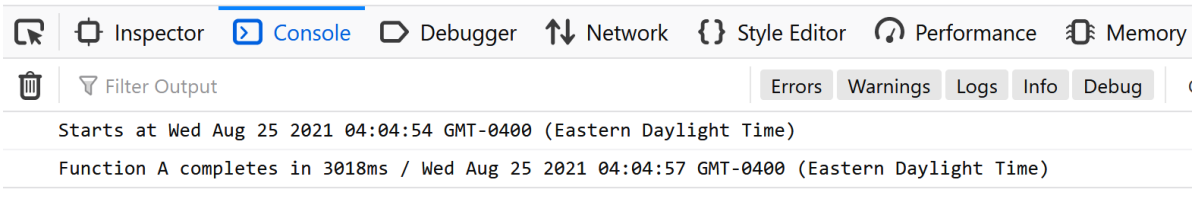
```
new Promise(function (resolve, reject) {
    setTimeout(() => resolve(1), 1000);
}).then(function (result) {
    return result * 2;
}).then(function (result) {
    return result * 3;
}).then(function (result) {
    return result * 2;
});
```

```
JS_Fund > promises.html > html > body > script > a
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Promises-Reda</title>
</head>
<body>
    <script>

        var d = new Date(); // current date
        var n = 3000 + d.getTime(); //Add 3 seconds to current date
        var counter = 1;

        function a() {
            // 1 - Let's modify the code while adding a promise
            let promise =  new Promise(function(resolve, reject) {

                setTimeout(function() {
                    console.log("Function A completes in "+ (new Date() - d) + "ms / " + new Date());
                    counter = 2;
                    resolve(counter); //2 - add resolve meth
                }, 3000);
            }); // end of the promise

            return promise; //3- the var to return

        }

        function b() {
            //
            console.log("Function B completes. Conter Value= " + counter + " , and the date is: " + new Date());
        }

        console.log("Starts at " + d);
        //here calling out all the promise methods .then(), .catch() and .finally()
        a().then(b);
        // in this exampe we didn't use any reject or error scenario

    </script>
</body>
</html>
```

```
Inspector  Console  Debugger  Network  Style Editor  Performance  Memory
Filter Output   Errors  Warnings  Logs  Info  Debug
Starts at Wed Aug 25 2021 03:58:56 GMT-0400 (Eastern Daylight Time)
Function A completes in 3017ms / Wed Aug 25 2021 03:58:59 GMT-0400 (Eastern Daylight Time)
Function B completes. Conter Value= 2 , and the date is: Wed Aug 25 2021 03:58:59 GMT-0400 (Eastern Daylight Time)
```

JavaScript Basics:
Promises 3

- If we comment resolve, no success will be done by the promise
- Therefore, function b() won't be executed



JS_Fund > promises.html > html > body > script > a > promise > <function> > setTimeout() callback

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Promises-Reda</title>
</head>
<body>
    <script>

        var d = new Date(); // current date
        var n = 3000 + d.getTime(); //Add 3 seconds to current date
        var counter = 1;

        function a() {
            // 1 - Let's modify the code while adding a promise
            let promise =  new Promise(function(resolve, reject) {

                setTimeout(function() {
                    console.log("Function A completes in "+ (new Date() - d) + "ms / " + new Date());
                    counter = 2;
                    // resolve(counter); //if we comment resolve, no success will be done by the promise
                }, 3000);
            }); // end of the promise

            return promise; //3- the var to return

        }

        function b() {
            //
            console.log("Function B completes. Conter Value= " + counter + " , and the date is: " + new Date());
        }

        console.log("Starts at " + d);
        a().then(b);

    </script>
</body>
</html>
```

JavaScript Basics:

Promises Hell


```
function a(param) {
    return new Promise(function (resolve, reject) {
        resolve(param * 2);
    });
}
function b(param) {
    return new Promise(function (resolve, reject) {
        resolve(param * 3);
    });
}

a(1)
    .then((result) => {
        b(result)
            .then((result) => {
                console.log('Pyramid of Doom!', result);
            });
    });
```

JavaScript Basics:

Await Expressions in Aync Functions

- Await expression works in an Asynch function
- Supported in Lambda for Node.js 8.10
- Cleaner code
  - Used in Lambda Functions, including Alexa Skills
  - Makes code easier to debug
  - Execution in sequence
  - Reliability

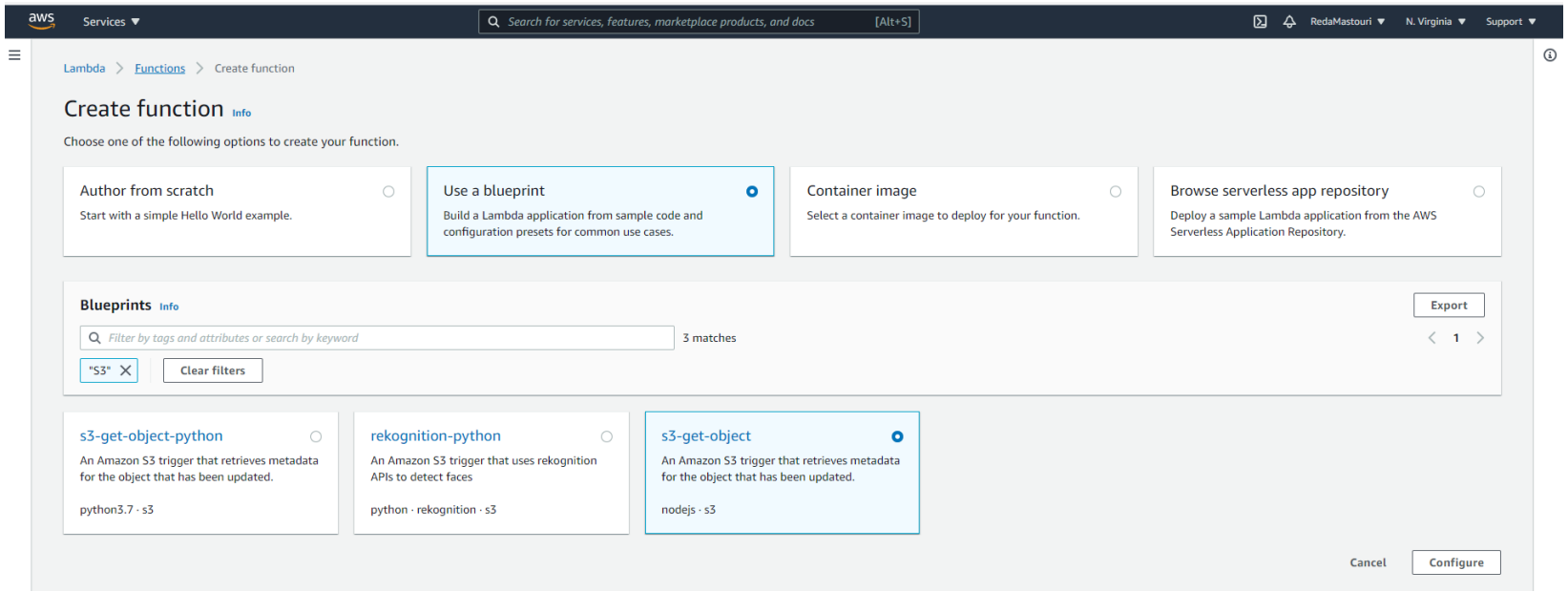
```
Welcome  promises.html U
JS_Fund > promises.html > html > body > script
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Promises-Reda</title>
</head>
<body>
    <script>

        var d = new Date(); // current date
        var n = 3000 + d.getTime(); //Add 3 seconds to current date
        var counter = 1;

        function a() {
            // 1 - Let's modify the code while adding a promise
            let promise =  new Promise(function(resolve, reject) {

                setTimeout(function() {
                    console.log("Function A completes in "+ (new Date() - d) + "ms / " + new Date());
                    counter = 2;
                    // we will prefix aync to function b() because only asynchronous function can contain await Expr.
                    resolve(counter);
                }, 3000);
            });

            return promise;
        }

        // async prefix added
        async function b() {
            // adding this bloc
            var x;
            try{
                x = await a();
                console.log("Function B completes. Conter Value= " + counter + " , and the date is: " + new Date());
            }
            catch (error){
                console.log("Error a ssat!");
            }
        }

        console.log("Starts at " + d);
        // then call just function b
        (b);

    </script>
</body>
</html>
```


10:19:48 AM
STANDARD TIME
Removable Drive (F:)
0.0 B | 0.0 B
[SWAP]
[CPU]
[RAM]
[UPTIME: 0D 1:53]
User :
Host :
OS : Windows 7
System Uptime: 1 hour 51 min
Power: AC Line
Battery: No Battery
Remaining: 100%
[FREE] 2.24 GB
[OF] 7.95 GB
[TEMP] 0.00 °C
CPU 2200 MHz
CPU TEMPERATURE: 0°C
325.7 GB
216.6 GB FREE
66%
LAN
WAN
DN: 28 k
UP: 10 k
Friday
25.05.2012
Computer
Network
Documents
Control Panel
Downloads
Game Booster
Movies
THESIS
Music
Settings
ASUS
Maintenance
General
Visual Tools
Office Tools
Multimedia
Links
INFO

AWS Lambda Basics

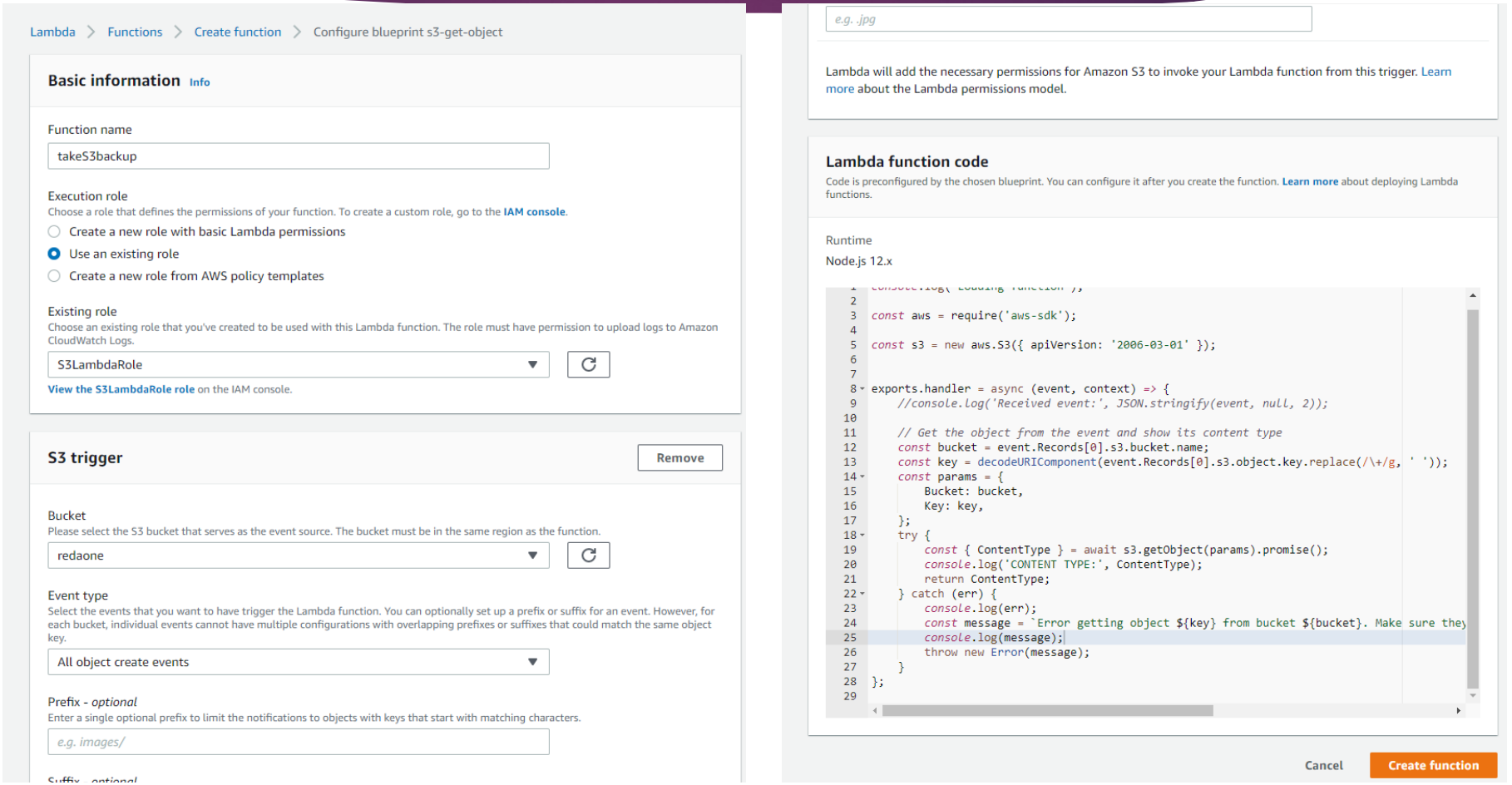
Create Lambda Function

- S3-get-object from Lambda blueprints



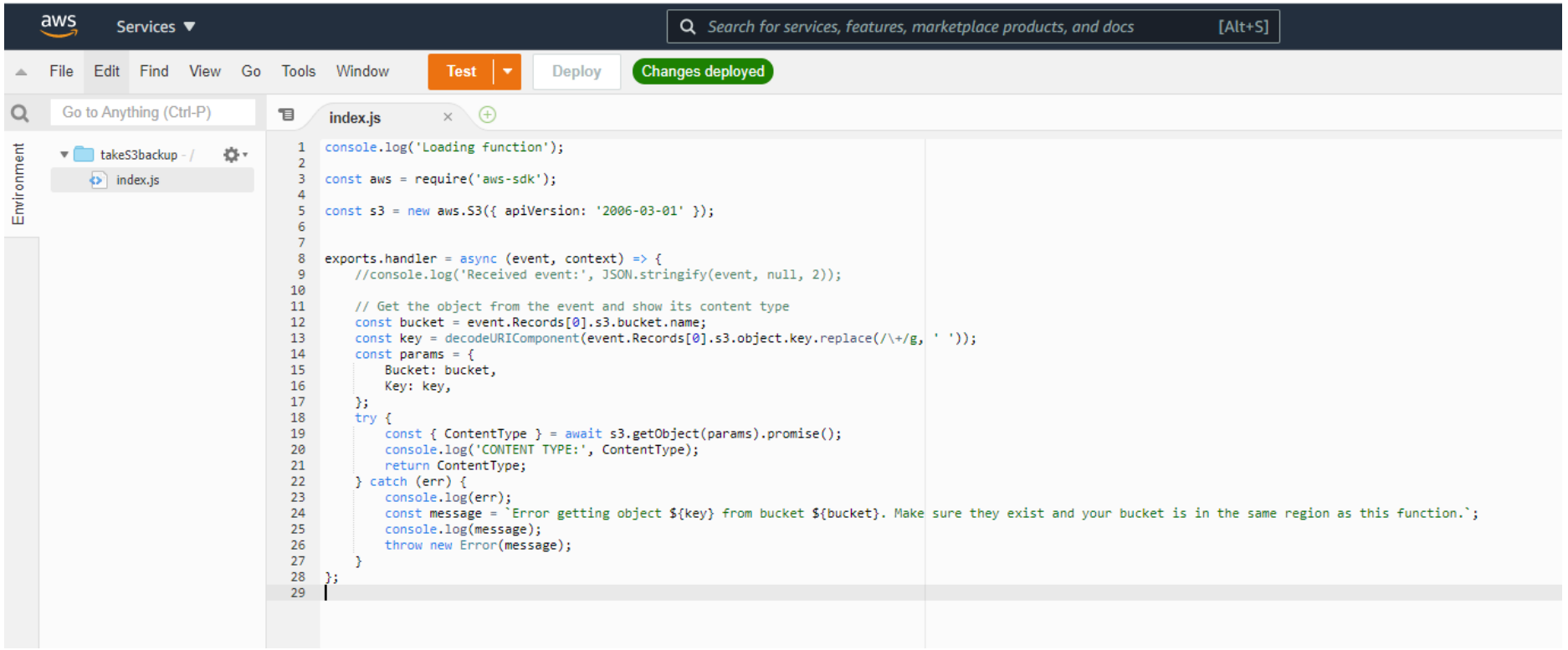
AWS Lambda Basics

Create Lambda Function



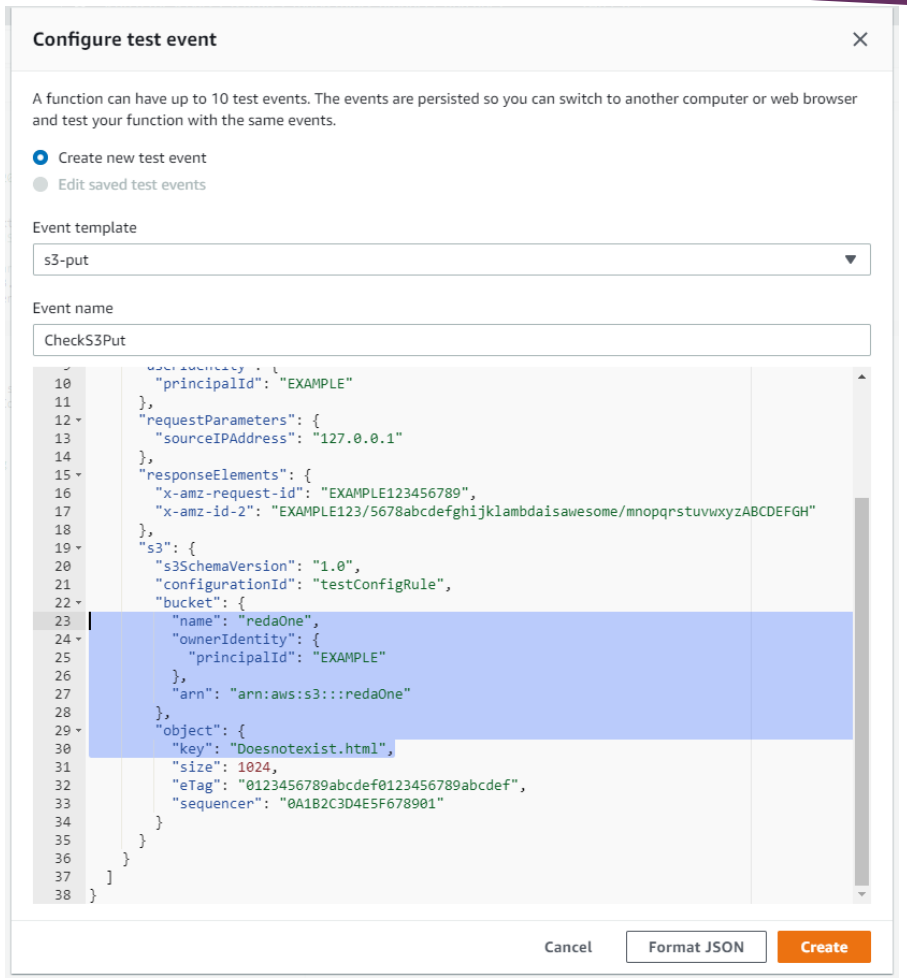
AWS Lambda Basics

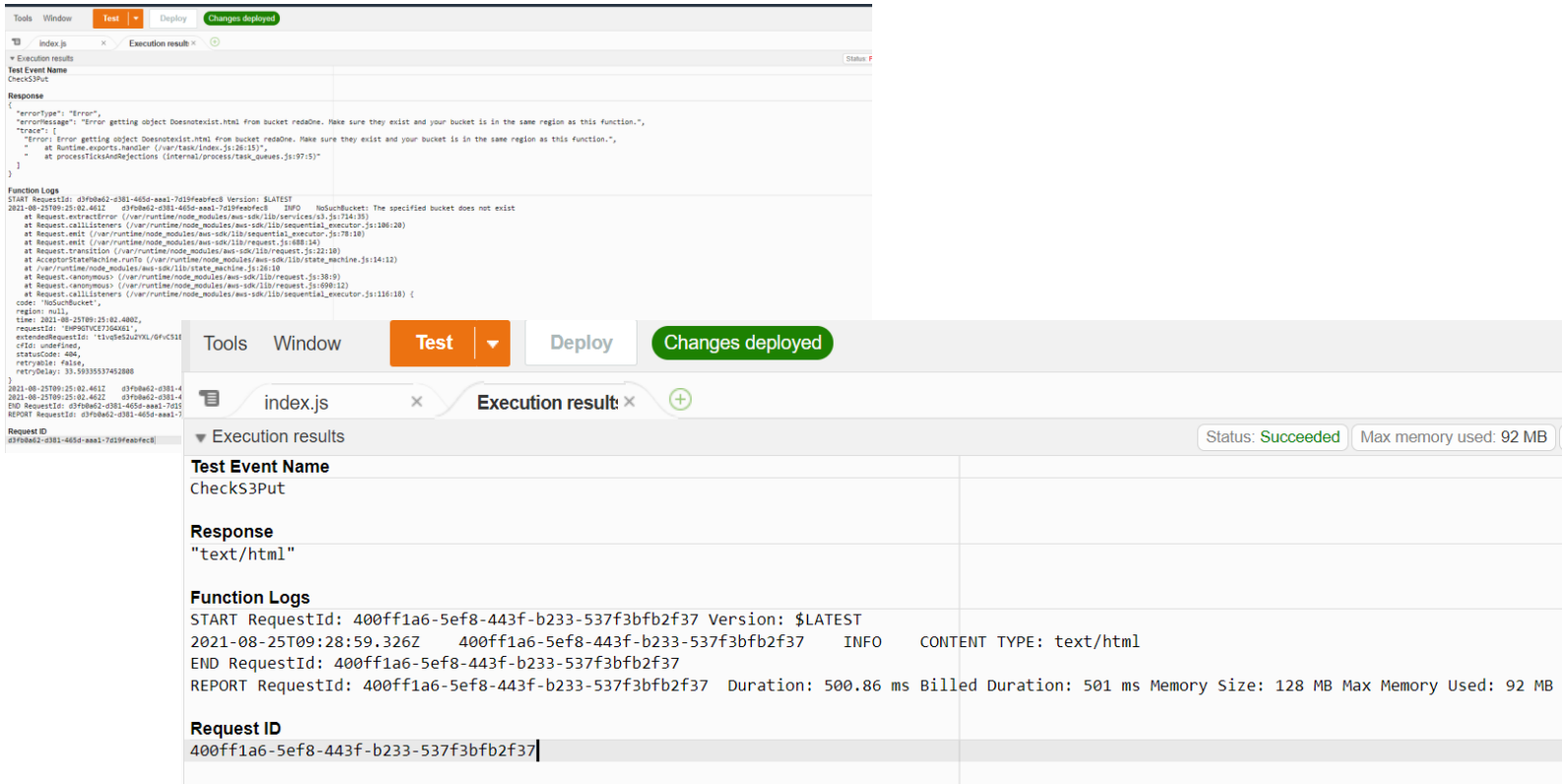
Create Lambda Function



AWS Lambda Basics

Prepare Lambda test Event to Check connectivity





AWS Lambda Basics

Upgrade Lambda Function

Tools Window **Test** **Deploy** Changes not deployed

index.js Execution results

```
const aws = require('aws-sdk');

const s3 = new aws.S3({ apiVersion: '2006-03-01' });


exports.handler = async (event, context) => {
    //console.log('Received event:', JSON.stringify(event, null, 2));

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
    const params = {
        Bucket: bucket,
        Key: key,
    };
    try {
        const { ContentType } = await s3.getObject(params).promise();
        console.log('CONTENT TYPE:', ContentType);
        // return ContentType; //commenting this no longer needed!

    //target variable where the backup will be stored
        var targetBucket = bucket+"-backup"; //redaone-backup
        //printing something to the screen saying that file being copied:
        console.log("Copying files: "+key+" from: "+bucket+" to the following bucket: "+ targetBucket );
        //Creating a contant which we have to specify under CopySource parameters as dictionary with the URL
        // of file location
        // target bucket name
        // and file name
        const paramsNew = {
            CopySource : bucket + "/" + encodeURIComponent(key),
            Bucket : targetBucket,
            Key : key
        };

    // here we create a constant finalOutput where we use await to call S3's copyObject method and pass the parameters for the promise
        const finalOutput = await s3.copyObject(paramsNew).promise();
    // printing out the params
        console.log("File: "+key+ " has been successfully copied from "+ bucket +" to "+ targetBucket+ " as expected!");
        return "File copied successfully!";

        }
    catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

AWS Lambda Basics

Testing & Verification

Tools Window **Test** Deploy Changes deployed

index.js

```
    try {
        const { ContentType } = await s3.getObject(params).promise();
        console.log('CONTENT TYPE:', ContentType);
        // return ContentType; //commenting this no longer needed!

    //target variable where the backup will be stored
        var targetBucket = bucket+"-backup"; //redaone-backup
        //printing something to the screen saying that file being copied:
        console.log("Copying files: "+key+" from: "+bucket+" to the following bucket: "+ targetBucket );
        //Creating a contant which we have to specify under CopySource parameters as dictionary with the URL
        // of file location
        // target bucket name
        // and file name
        const paramsNew = {
            CopySource : bucket + "/" + encodeURIComponent(key),
            Bucket : targetBucket,
            Key : key
        };

    // here we create a constant finalOutput where we use await to call S3's copyObject method and pass the parameters for the promise
        const finalOutput = await s3.copyObject(paramsNew).promise();
    // printing out the params
        console.log("File: "+key+ " has been successfully copied from "+ bucket +" to "+ targetBucket+ " as expected!");
        return "File copied successfully!";
```

Execution results

Execution results — Status: Succeeded

Test Event Name

CheckS3Put

Response

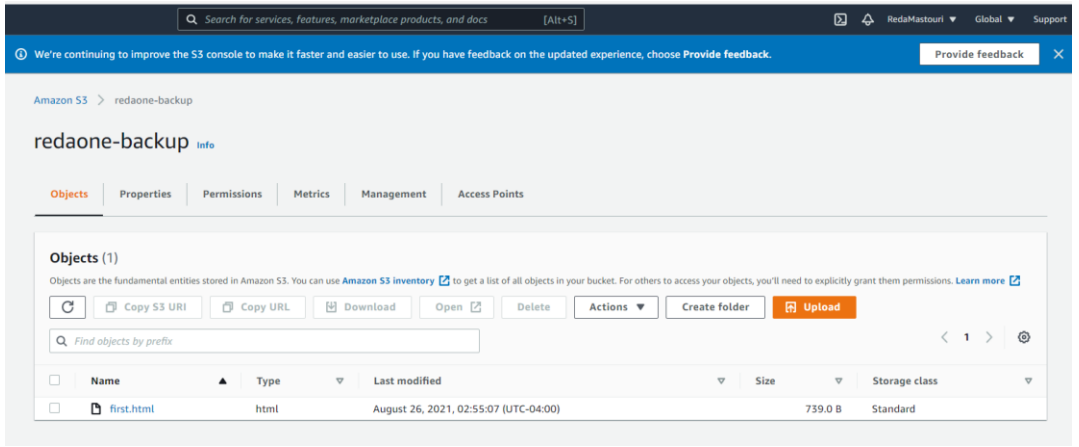
```
"File copied successfully!"
```

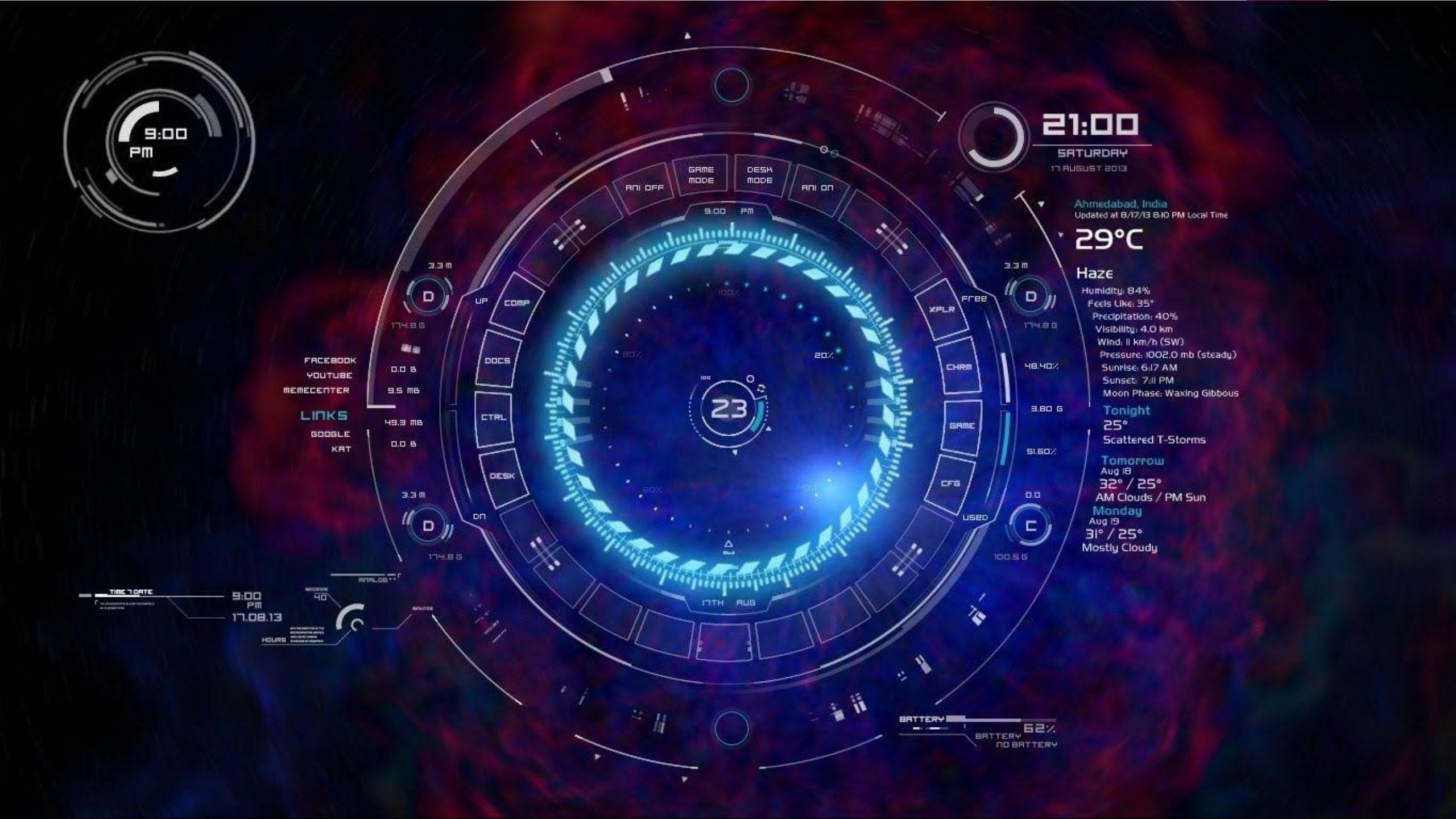
Function Logs

```
START RequestId: df7ee8c6-2e90-461c-a55f-443066f2181d Version: $LATEST
2021-08-26T06:55:05.738Z    df7ee8c6-2e90-461c-a55f-443066f2181d    INFO    CONTENT TYPE: text/html
2021-08-26T06:55:05.738Z    df7ee8c6-2e90-461c-a55f-443066f2181d    INFO    Copying files: first.html from: redaone to the following bucket: redaone-backup
2021-08-26T06:55:06.378Z    df7ee8c6-2e90-461c-a55f-443066f2181d    INFO    File: first.html has been successfully copied from redaone to redaone-backup as expected!
END RequestId: df7ee8c6-2e90-461c-a55f-443066f2181d
REPORT RequestId: df7ee8c6-2e90-461c-a55f-443066f2181d  Duration: 1336.81 ms    Billed Duration: 1337 ms    Memory Size: 128 MB Max Memory Used: 90 MB  Init Duration: 372.47 ms
```

Request ID

df7ee8c6-2e90-461c-a55f-443066f2181d

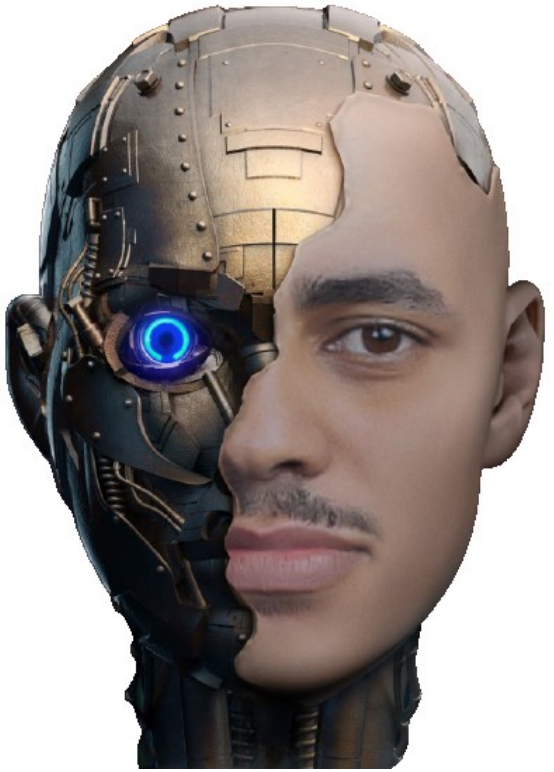



9:00
PM
21:00
SATURDAY
17 AUGUST 2013
ANI OFF
GAME MODE
DESK MODE
ANI ON
9:00 PM
UP
COMP
DOCS
CTRL
DESK
ON
XPLR
FREE
CHRM
GAME
CFG
USED
23
17TH AUG
3.3 m
D
174.8 G
3.3 m
D
174.8 G
3.3 m
D
174.8 G
0.0
C
100.5 G
FACEBOOK
YOUTUBE
MEMECENTER
0.0 B
9.5 MB
LINKS
GOOGLE
KAT
49.3 MB
0.0 B
48.40%
3.80 G
51.60%
Ahmedabad, India
Updated at 8/17/13 8:10 PM Local Time
29°C
Haze
Humidity: 84%
Feels Like: 35°
Precipitation: 40%
Visibility: 4.0 km
Wind: 11 km/h (SW)
Pressure: 1002.0 mb (steady)
Sunrise: 6:17 AM
Sunset: 7:11 PM
Moon Phase: Waxing Gibbous
Tonight
25°
Scattered T-Storms
Tomorrow
Aug 18
32° / 25°
AM Clouds / PM Sun
Monday
Aug 19
31° / 25°
Mostly Cloudy
TIME TO ATE
9:00
PM
17.08.13
ANALOG
HOURS
BATTERY
62%
BATTERY
NO BATTERY

Starting to build a Schedulify Skill

Designing a Voice Interaction Model

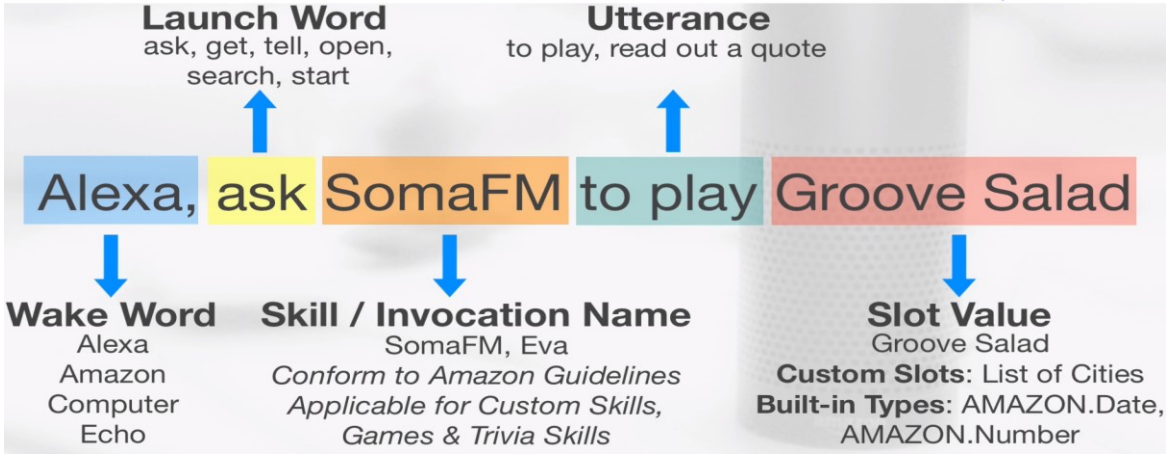
- Invocation Name:
  - Reda
- Job:
  - Read out Quotes
  - *Example: Alexa, Ask Reda for a quote from Einstein*
  - *Example: Alexa, Ask Reda what did Abraham Lincoln say*

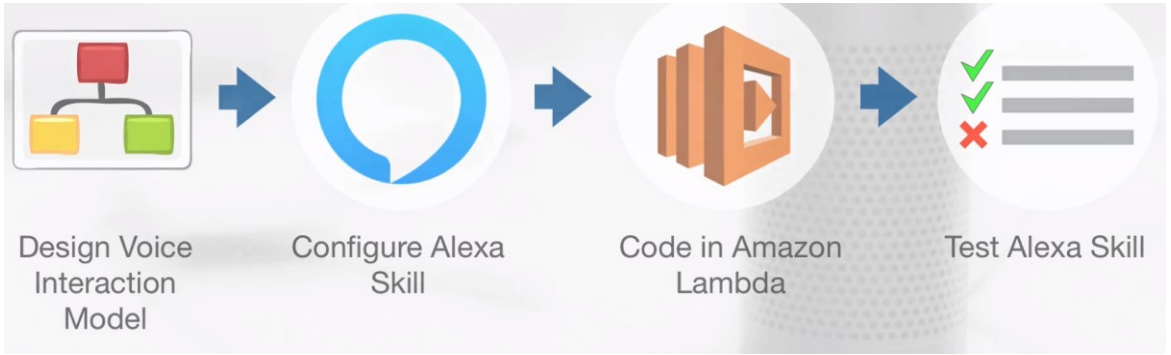


Starting to build a Schedulify Skill

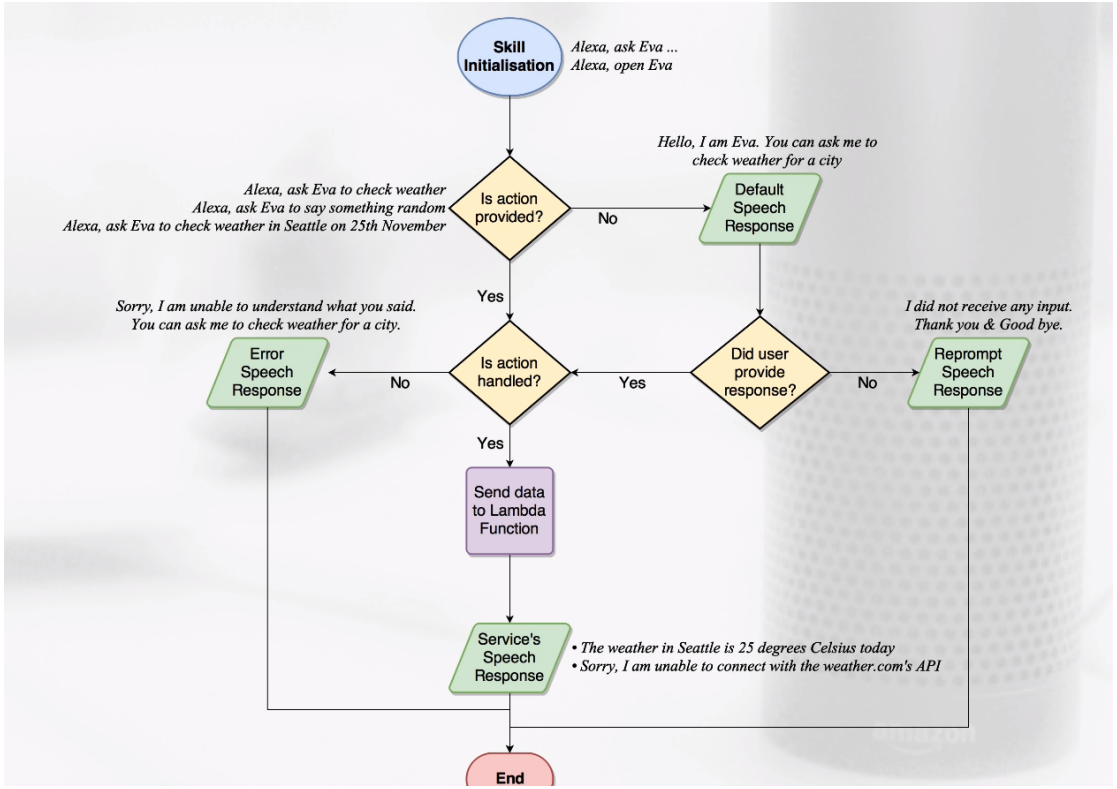
Designing a Voice Interaction Model

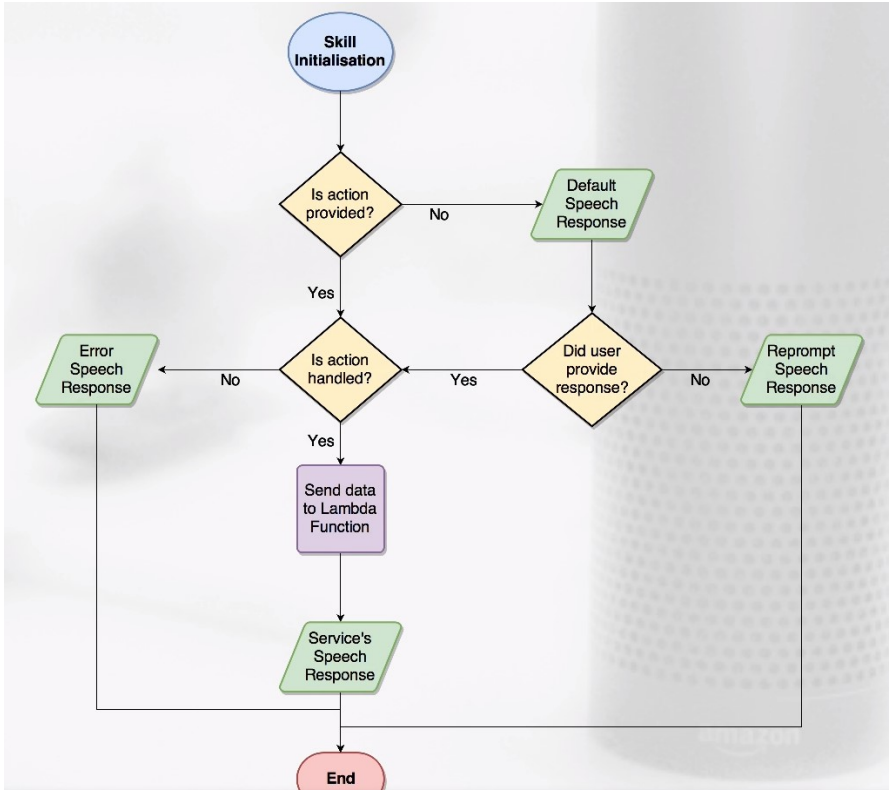
- Step 1: Preliminary work
  - Create a Voice Interaction Model
  - Prepare Intents, Slots, and Utterances
  - Data - Authors and their Quotations
- Step 2: Skill Configuration
  - On Amazon Developer Portal
- Step 3: Skill Development
  - Create a Lambda function
  - Link Lambda function with Alexa Skill through ARN
- Step 4: Skill Testing
  - Lambda test events
  - Skill simulator on Developer Portal
  - Physical device / Amazon Echo





Starting to build a Schedulify Skill





Starting to build a Schedulify Skill

Create a new skill

Cancel | Create skill

Model: Custom
Host: Alexa Hosted (Node.js)
Hosting Region: US East (N. Virginia)

Skill name

Reda - The Awesome Personal Assistant

37/50 characters

Brand names are only allowed if you provide proof of rights in the testing instructions or if you use the brand name in a referential manner that doesn't imply ownership (examples of terms that can be added to a brand name for referential usage: unofficial, unauthorized, fan, fandom, for, about)

Primary locale

A locale refers to a language and the location (country) in which its spoken. Your primary locale is what you will start building your skill in. You can add locales after your skill is created.

English (US)

Sync locales

Sync all locales with the same language to the Primary locale. Changes you make to your skill in the Primary locale automatically propagate to all other locales of the same language. You can manage these settings or turn off this feature anytime in Language settings. Learn more

1. Choose a model to add to your skill

There are many ways to start building a skill. You can design your own custom model or start with a pre-built model. Pre-built models are interaction models that contain a package of intents and utterances that you can add to your skill.

Custom (SELECTED)	Flash Briefing	Smart Home	Music	Video	Knowledge
Design a unique experience for your users. A custom model enables you to create all of your skill's interactions.	Give users control of their news feed. This pre-built model lets users control what updates they listen to. "Alexa, what's in the news?"	Give users control of their smart home devices. This pre-built model lets users turn off the lights and other devices without getting up. "Alexa, turn on the kitchen lights"	Give users complete control of their music. This pre-built model lets users search, pause, skip, or shuffle in your skill. "Alexa, play music by Lady Gaga"	Let users find and consume video content. This pre-built model supports content searches and content suggestions. "Alexa, play Interstellar"	Give users Q&A on bu upload via spreadshee personal devices, Alexa for Hospitality, or Alexa for Residential. "Alexa, how do I get on the Wi-Fi?"

2. Choose a method to host your skill's backend resources

You can provision your own backend resources or you can have Alexa host them for you. If you decide to have Alexa host your skill, you'll get access to our code editor, which will allow you to deploy code directly to AWS Lambda from the developer console.

Alexa-hosted (Node.js) (SELECTED)	Alexa-hosted (Python)	Provision your own
Alexa will host skills in your account and get you started with a Node.js template. You will gain access to AWS Lambda endpoints in all Alexa service regions, a DynamoDB table for data persistence, and S3 for media storage. Learn more	Alexa will host skills in your account and get you started with a Python template. You will gain access to AWS Lambda endpoints in all Alexa service regions, a DynamoDB table for data persistence, and S3 for media storage. Learn more	Provision your own endpoint and backend resources for your skill. This is recommended for skills that have significant data transfer requirements. You will not gain access to the console's code editor.

Choose a template to add to your skill

Select a skill template from the list below or import a skill shared by the Alexa community as a public C

Start from Scratch (SELECTED)

This skill gets you started with the required intents and with code demonstrating "Hello World" functionality if you are building an Alexa-hosted skill. Learn more

By Alexa

Fact Skill

Build topic and s invok

By A

meeting rooms in their office.

Creating your Alexa hosted skill.

It will take about a minute.

Creating a voice model...

Skill builder checklist

Complete these steps to be able to test your skill using the simulator in the test tab, or with your echo device.

1. Invocation Name ›
Enter an invocation name for your skill

2. Intents, Samples, and Slots ›
Add at least one intent and one sample utterance

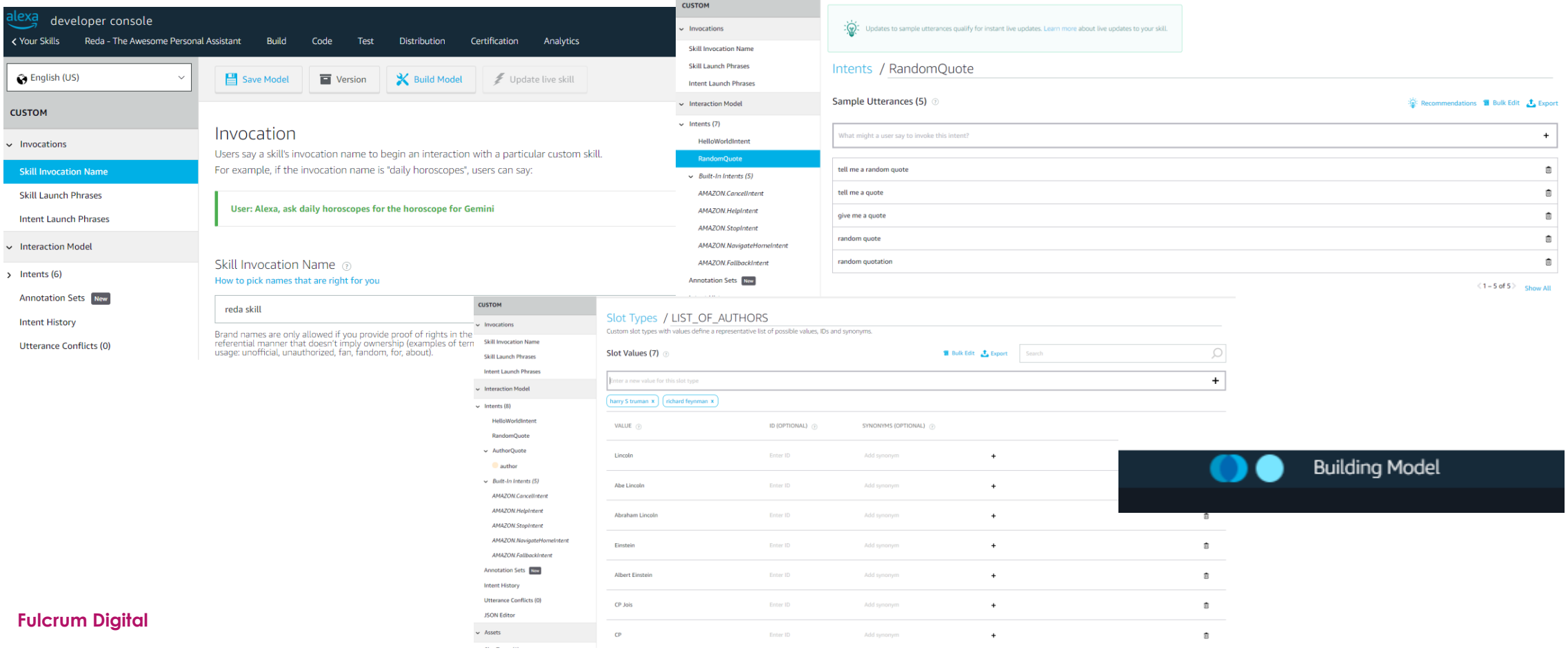
3. Build Model ›
Successfully build your interaction model

4. Endpoint ›
Set a web service endpoint to handle skill requests

Monetize Your Skill ›
Add In-Skill Products, or offer a Paid Skill

Starting to build a Schedulify Skill

Configuration of Reda 1.0 Custom Skill on Dev Portal



Starting to build a Schedulify Skill

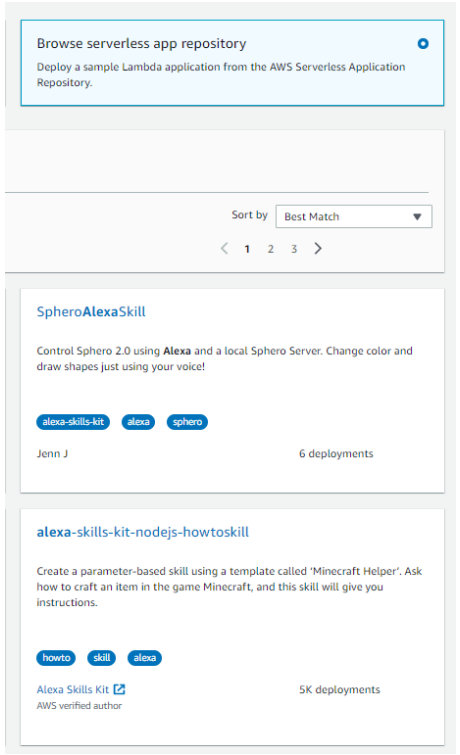
Request handlers, Response Objects and Builder in Lambda function for Alexa

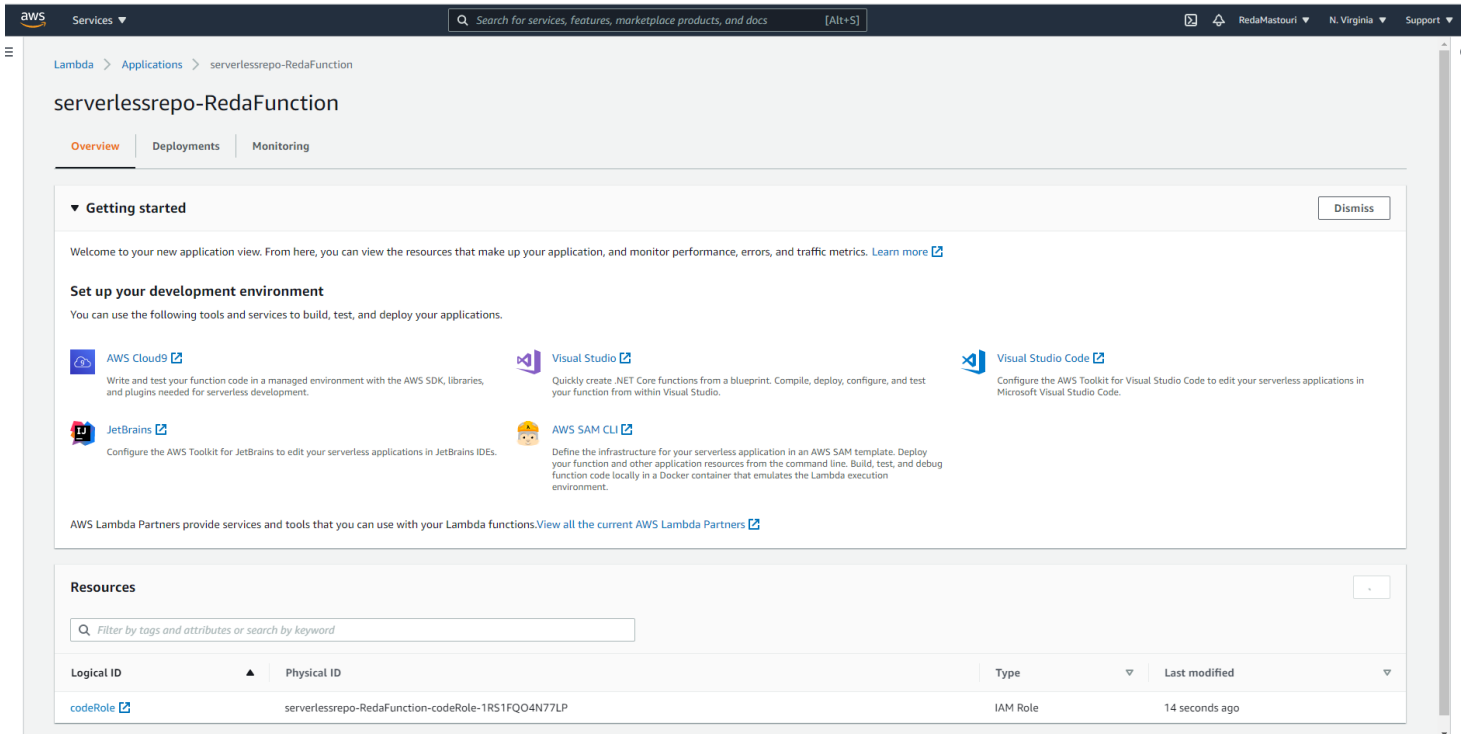
- Request Handlers
  - Custom handlers: Manage intents
  - Pre-defined handlers: Provided by Amazon
- Intent Request Handlers: Managing intents defined in intents schema
  - What is done: **RandomQuote** and **AuthorQuote**
  - It is called when Utterance matches Intent Slot Value is passed on if applicable
- Pre-defined Request Handlers
  - Like **LaunchRequest**: When skill is invoked without any action
    - Example: Alex, open Reda
  - Then there is **SessionEndedRequest**: When an Alexa session ends
- Buil-in Intents
  - Such as **AMAZON.HelpIntent**: if user opens skill, and asks for help
  - Need to be provided in the Intent Schema. No sample Utterance required
- Response Objects
  - There are two: Audible Response and Visual Response

- Response Objects
  - There are two: Audible Response and Visual Response
  - Audible Response Object:
    - Output Speech Response: A skill reads out a quote
    - Reprompt Speech Response: Ask user again if no input provided, while maintaining context
  - Visual Response Object:
    - For echo show/spot, Alexa app
    - **Cards**: they provide **Title** & **Content**
- Using both Audio and Visual
  - In lambda function, provide:
    - Card values
    - Speech output
- Response Builder – ASK v2
  - Helper methods to construct response
    - To end conversation: **.speak()** method
    - To ask for user's input: **.reporompt()** method
    - To use a card response: **.withSimpleCard()** and **.withStandardCard()**

Starting to build a Schedulify Skill

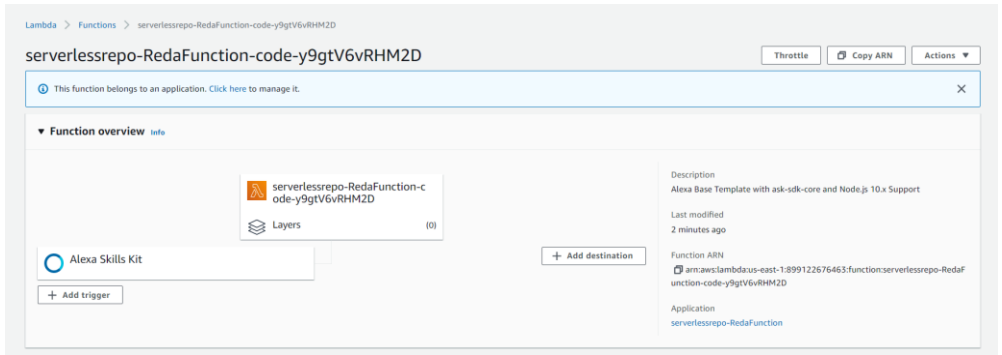
Reda bot introducing itself: On AWS Lambda

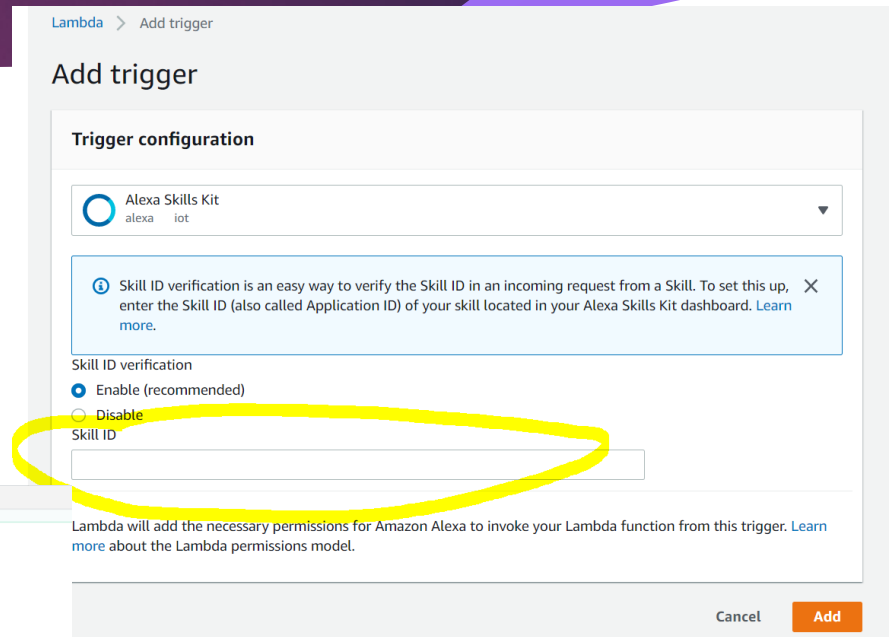


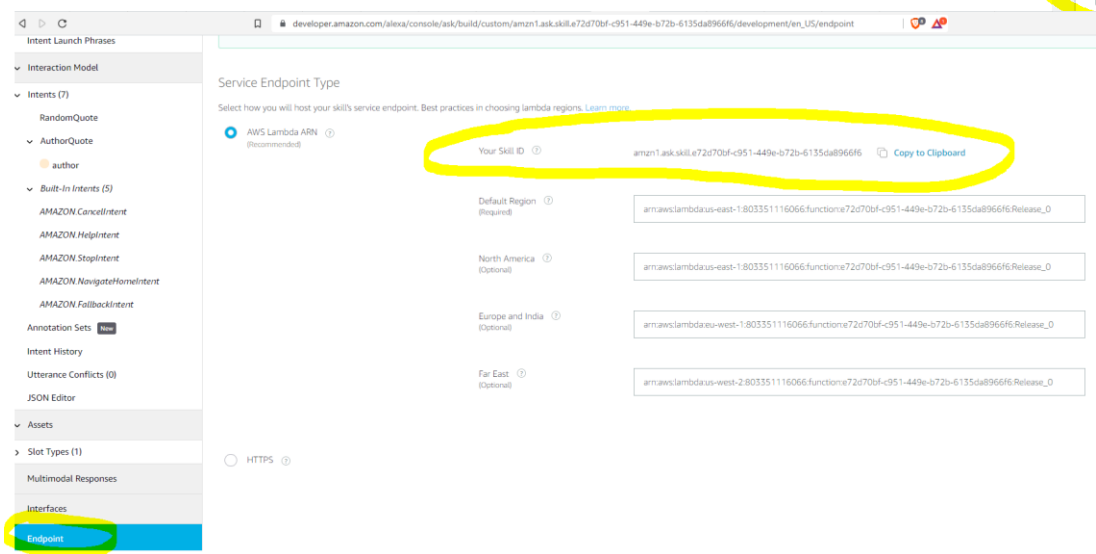


Starting to build a Schedulify Skill

Reda bot introducing itself: On AWS Lambda

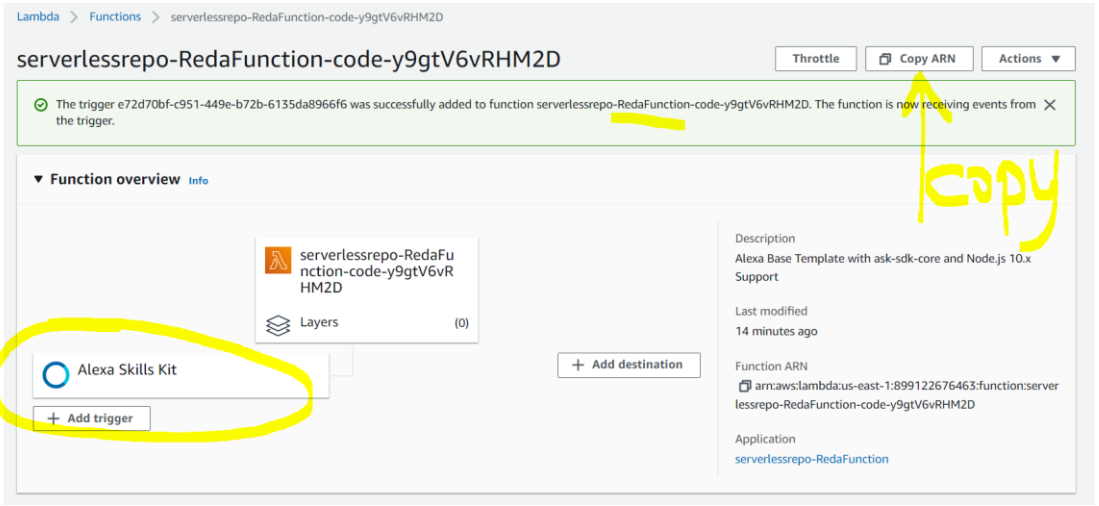


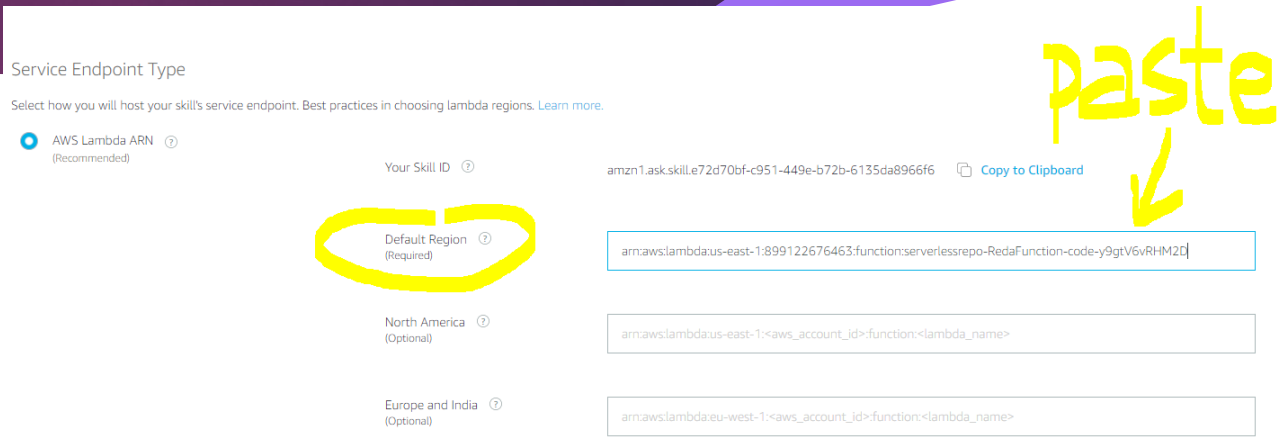


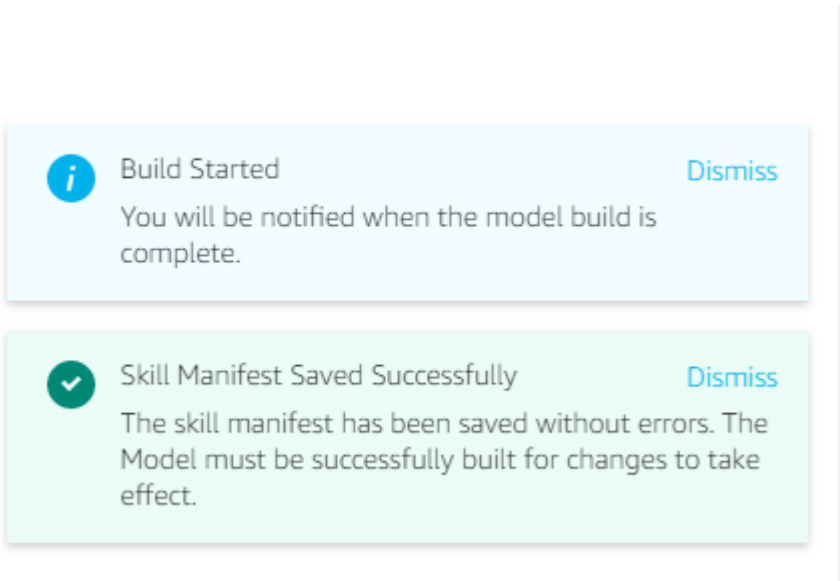


Starting to build a Schedulify Skill

Reda bot introducing itself: On AWS Lambda

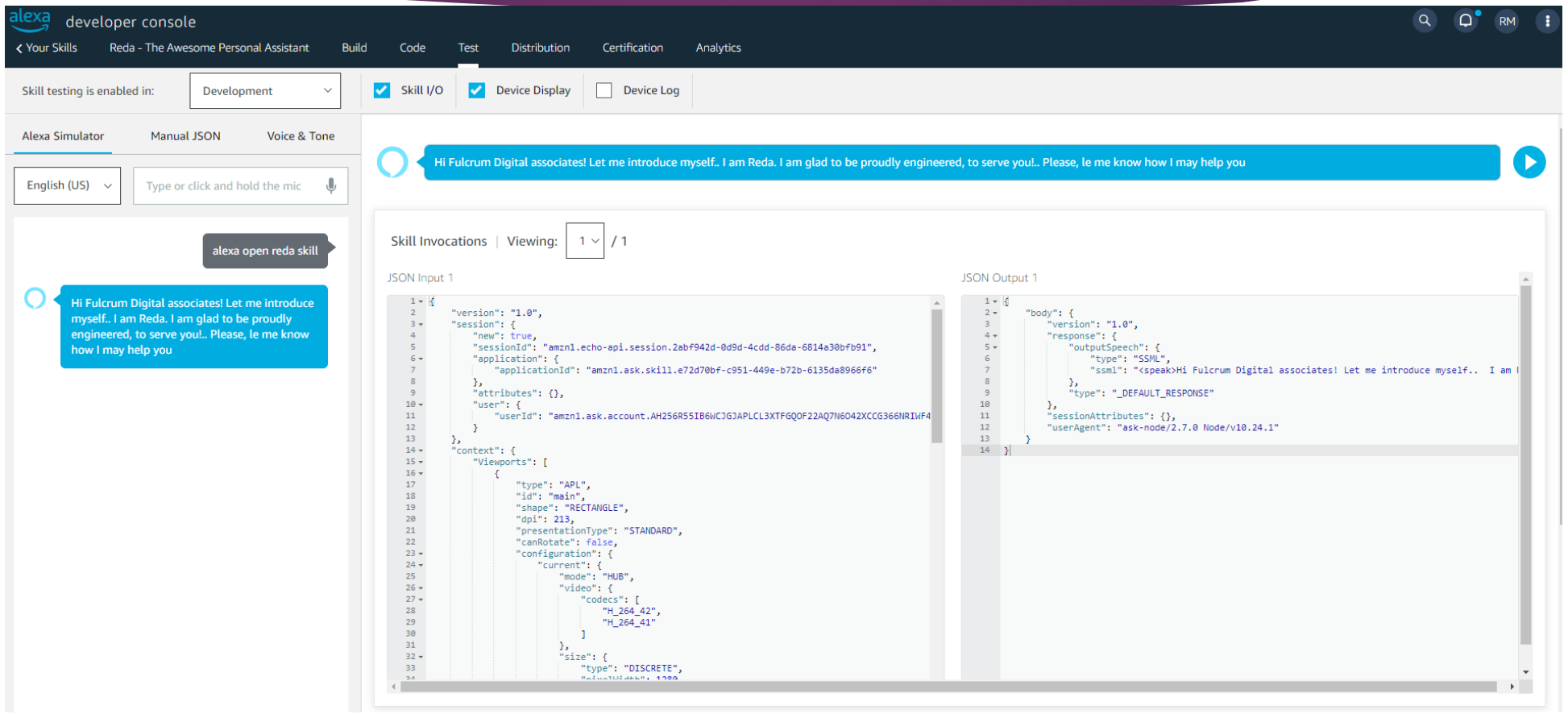






Starting to build a Schedulify Skill

Testing Reda Skill Bot



Starting to build a Schedulify Skill

Extending Lambda function to achieve first goal


```
// Include the Alexa SDK v2const Alexa = require("ask-sdk-core");// The
"LaunchRequest" intent handler - called when the skill is launchedconst
LaunchRequestHandler = {  canHandle(handlerInput) {    return
handlerInput.requestEnvelope.request.type === "LaunchRequest";  },
handle(handlerInput) {    const speechText = "Hi Fulcrum Digital associates! Let
me introduce myself..  I am Reda. I am glad to be proudly engineered,  to serve
you!.. Please, le me know how I may help you";    // Speak out the speechText
via Alexa    return
handlerInput.responseBuilder.speak(speechText).getResponse();  }};// Register
the handlers and make them ready for use in Lambdaexports.handler =
Alexa.SkillBuilders.custom()  .addRequestHandlers(LaunchRequestHandler)
.lambda();
```


JARVIS OS
Ver 1.2.5
User: Alejandro Montoya
GAME MODE
DESK MODE
ANI OFF
ANI ON
UP
COMP
DOCS
CTRL
DESK
PRG
DN
PL
STH
MAY
EQ
VOL
USED
CFG
GAME
CHRM
XPLR
FREE
Juarez, Mexico
Updated at 5/5/13 5:55 PM Local Time
30°C
Fair
Humidity: 14%
Feels Like: 30°
Precipitation: 0%
Visibility: 16.1 km
Wind: 8 km/h (E)
Pressure: 1015.2 mb (steady)
Sunrise: 7:00 AM
Sunset: 8:19 PM
Moon Phase: Waning Crescent
Tonight
15°
Mostly Clear
Tomorrow
May 6
31° / 17°
Isolated T-Storms
Tuesday
May 7
33° / 20°
Mostly Sunny
MOUNTAIN STANDARD TIME (MEXICO)
NETWORK TRAFFIC UP
NETWORK TRAFFIC DOWN
TOTAL TRAFFIC UP
TOTAL TRAFFIC DL
STARK INDUSTRIES

Starting to build a Schedulify Skill

- Revenue
  - Expenditure
- Generating revenue
- Culinary suite

Next ..
Engage your customers with an Alexa skill

- Companies large and small are **building Alexa skills** to capitalize on this growing opportunity — there are now more than 30,000 skills published in Amazon's catalog.
- FulcrumOne team of **Alexa developers** can help your company extend your digital reach through a custom skill on millions of **Amazon Echo devices**.

Wrapping Up

- In a voice-first world, Alexa, Google Assistant, Siri, and other **AI-enabled assistants** will be a powerful marketing medium. It can combine sales and distribution channel, fulfillment, and service center. Following the gold rush of building brand mobile apps, the current trend is building branded skills for voice assistants.
- The increasingly capable voice assistants will be reshaping the brands' and retailers' connections with customers. The consumers' allegiance will be shifting from trusted brands **to a trusted platform that helps them navigate the ocean of choices.** Re-ordered essentials will be flowing to households like water or electricity. Growing creative audio content will advertise without annoying and alienating listeners. The AI will gather information about the consumers' preferences to improve the shopping experience and offer the portal to a boundless marketplace of entertainment, news, goods, and services. Amazon Alexa is sure to be one of those platforms.
- For publishers, music services, and smart-home tech companies, custom Alexa skills seem to be a must for increasing engagement with millions of users. Other company managers and marketers should consider creating a branded skill as well: it offers a natural way to introduce a brand to potential consumers and abundant opportunities for product improvement, marketing, and advertising, and helps cultivate customer loyalty.

References

- **How do Alexa Skills work?:** https://chatbotsmagazine.com/how-does-alexa-skills-works-82a7e93dea04
- **ASK Documentation**: https://developer.amazon.com/en-US/docs/alexa/custom-skills/understanding-custom-skills.html
- **How does your brand benefit from developing an Alexa Skill?:** https://www.deligence.com/blog/benefit-from-developing-an-alexa-skill/
- **Practical Hands-on Guide for Alexa Skill Development (2020):** https://fulcrumdigital.udemy.com/course/build-your-rad-personal-assistant-with-amazon-alexa-custom-skills/