|  |  |  |
| --- | --- | --- |
| Cell | **Visuals** |  |
| 1 | TEXT ON SCREEN:  Download the manual and solution projects at:  [www.cypress.com/training/wicedwifi-101](http://www.cypress.com/training/wicedwifi-101) | Hi, I’m Alan Hawse, welcome back to WICED WiFi 101. In this video, I'm going to introduce you to Amazon Web Services. After that, I'll show you how to interact with it from a WICED IoT device using MQTT. |
| 2 |  | Amazon Web Services, or AWS, is a secure cloud services platform, offering compute power, database storage, content delivery and other functionality - which incidentally makes Amazon more money than everything else combined. |
| 3 | TEXT ON SCREEN:  AWS IoT  Elastic Cloud  Lamda  Dynamo DB  Aurora  Amazon SNS | AWS is built from a vast array of both virtual and actual servers and networks as well as a boatload of webserver software and administrative tools such as AWS IoT, the Amazon Elastic Cloud which is a virtualized compute capability for rent, Amazon Lamda which allows you to send event driven tasks through the Cloud, Databases like Dynamo DB and Amazon Aurora, the Amazon simple notification system which allows you to send SMS and email messages, and so on. More features are being added all the time to increase the power of what AWS can do for you. And no, Amazon didn't pay me to say all of that stuff – it really is remarkable how well it works. |
| 4 |  | Because AWS keeps adding new functionality, one of the annoying things is that by the time you watch this video you will find a bunch of small changes on the screens from what I'm actually showing to what's actually happening in the cloud. Oh well. I'm sure you'll sort it out – it's not that hard – but it will be a little bit different.  In this chapter we're going to focus on AWS IoT which provides a set of Cloud services that are optimized for IoT devices. |
| 5 | TEXT ON SCREEN:  Virtual MQTT Message Broker  Virtual HTTP Server  Thing Registry  Certificate Management  Thing Shadow  Rules Engine  IoT Application SDK | The AWS IoT Cloud service supports MQTT and HTTP based access, **plus** a bunch of server-side functionality that provides:   * A virtual MQTT Message Broker * A virtual HTTP Server * A Thing Registry, which is a web interface to manage the access to your *things* – you know – the "T" in IoT. Unfortunately that "things" word makes some of these sentences really awkward to say, so if you start laughing, I get it – I did too. * A web interface to manage the certificates and rules about *things* to help you handle security. * A "shadow", which is an online cache of the most recent state of your *thing***.** * A Rules Engine application that runs in the cloud that can subscribe to Topics and take programmatic actions based on your messages. For example, you could configure it to subscribe to an "Alert" topic, and if a *thing* publishes a warning message to the Alert topic, it would use Amazon SNS to send a SMS Text Message to your cell phone. That's pretty cool, right? I like it. It is pretty cool. * And finally an IoT applications SDK to help you build Web pages and cell phone Apps to go along with your IoT device. |
| 6 | TEXT ON SCREEN:  Things  Certificates  Policies | There are three types of resources in AWS: *Things*, *Certificates*, and *Policies*.  A *thing* is a representation of a device or a logical entity. It can be a physical device or sensor (for example, a light bulb or a switch on the wall). It can also be a logical entity like an instance of an application or a physical entity that does not connect to AWS IoT but can be related to other devices that do (for example, a car that has engine sensors or a control panel).  Certificates are used by AWS to verify the identities in both directions. Remember, you don't want your IoT hijacked by some crazy person in the cloud. So remember, your IoT device validates that it is talking to AWS and AWS validates that it's talking to your device and they both are who they claim to be. Certificates for your device can be created and managed on the AWS web site, which I'll demonstrate in a minute.  After creating a certificate for your internet-connected *thing*, you must create and attach an AWS IoT policy that will determine what AWS IoT operations the *thing* may perform. AWS IoT policies are JSON documents and they follow the same conventions as AWS Identity and Access Management policies. Remember earlier I talked to you about JSON being used everywhere and I hope you listened because all of the stuff we're going to do in this chapter is JSON based, and I'll show it to you all in a minute. |
| 7 | TEXT ON SCREEN:  AWS Console  (note – this is a transitional heading – that is, we want to give the viewer a clue that we are about to change subjects here)  5:25  SCREEN CAPTURE:  7-3-AWS-intro\_capture-1.trec | AWS provides a web "console" at console.aws.amazon.com that allows you to create resources, interact with the thing shadows, publish test messages, subscribe to updates, and do other IoT'ish things. When I teach this class in person, I provide access to my own AWS account, but since you are watching this as a video, you will need to create your own account to try out these exercises. Don't worry, the basic account is free for a year and actually it's really very inexpensive to use.  Once you get logged into the console, select IoT Core. On the left side you'll see a menu with some activities that you can do. For example, if I select Manage, I can see a list of all of the things that have been created on my account. As I said earlier, Amazon is changing these screens like semi-continuously and it's actually been frustrating to keep my material up to date so it may not look exactly the same as I'm showing you here but it should be close and I don't think you'll have a problem figuring it out, but don't panic if it looks a little bit different.  By Clicking on a thing, AWS will show me its details like its resource name, its security certificate, its shadow state, and so on.  Going back, I can see all of the certificates that I have created by clicking on secure. Then if I choose a certificate, I can see its policy and which things are attached to it.  Back to the top again, I can look at the settings to see my MQTT endpoint. This is the DNS name for my virtual MQTT broker that I'll connect to from my IoT device.  The Test selection opens up a virtual MQTT client that I can use to publish messages to a topic or subscribe to a topic. I'll show you how to use this once we have an MQTT device that we want to test out.  Finally, there is a selection that says "Learn" and I would recommend that you go there now and watch the tutorial by clicking "Start tutorial". It only takes a few minutes, but it gives you a good idea of how these pieces fit together. |
| 8 | TEXT ON SCREEN:  Thing Shadow  (note – this is a transitional heading) | A *thing* shadow (sometimes referred to as a device shadow) is a JSON document that's used to store and retrieve the current state of information for a *thing*- there's that JSON stuff coming back again. The *Thing* Shadow service maintains a *thing* shadow for each *thing* you connect to AWS IoT with. See - I told you that word thing would bounce around a bunch of times. You can use *thing* shadows to get and set the state of a *thing* over MQTT or over HTTP regardless of whether the *thing* is currently connected to the Internet. Each *thing* shadow is uniquely identified by its name. |
| 9 | TEXT ON SCREEN:  SEE TEXT1 at the end of this document.  (note: I can provide this as text if the picture isn't clear enough) | A JSON Shadow document looks like this….  As you can see, the top levels are state, metadata, version, clientToken, and timestamp.  Most of the time you will only care about the state. The state has 2 sub levels called desired and reported. Applications will change the values under "desired" to request an update to the state of a thing without having to directly connect to it. In other words, your thing will listen for changes to the desired state and will act upon those things. On the other side, things will change their values under "reported" to keep track of their actual state. Remember, each time the thing changes its state it will update the actual state in the shadow so that the rest of the world doesn't have to talk directly to the device to find out its state.  This is useful because it means two devices don't need to be simultaneously connected to the Cloud to be able to communicate. For example, consider an IoT connected light bulb. If the power goes out at your house, the light bulb will lose connectivity. Once the power comes back on, the light bulb can connect again and read the desired state to remember if it is supposed to be on or off.  The entries under desired and reported can be anything you want. In this example, there is a state called color and another state that's called sequence that indicates a color sequence. The names of the states and what they mean are entirely up to you and your application. You can define the states to mean whatever you need to make your system work correctly. |
| 10 | TEXT ON SCREEN:  Topics  (note – this is a transitional heading) | Topics are the way you Publish and Subscribe to MQTT messages. You can think of them as an address for the message.  The AWS Message Broker will allow you to create Topics with almost any name, with one exception: Topic names that start with "$aws/" are reserved by AWS IoT for specific functions. |
| 11 | TEXT ON SCREEN:  SEE TEXT2 at the end of this document. | The main use of $aws/ Topics is to interact with Thing Shadows. Topics that look like this will interact with the shadow for a thing called Thing1: |
| 12 | TEXT ON SCREEN:  SEE TEXT3 at the end of this document. | The first one is the Topic you would Publish to in order to send a new value to the Shadow for Thing1. The message you send is just a JSON message with the value or values in it that you want to change. For example, you could publish this message to say that you want the color to be blue. |
| 13 | TEXT ON SCREEN:  Show TEXT2 again. Highlight the lines with update/accepted, updated/rejected, and update/documents when Alan mentions them. | After sending your message, the server will publish a message to either the update/accepted or the update/rejected topic depending on if your message was, oh say, accepted and put into the Shadow or if it was rejected. If your message was accepted, the server will also publish the complete updated Shadow document to update/documents. |
| 14 | TEXT ON SCREEN:  Show TEXT2 again. Highlight the lines with get, get/accepted, and get/rejected. | The next group are used when you want to get the state of a shadow. Publishing a message to get will cause the server to respond with either get/accepted or get/rejected. In the case of get/accepted the message it sends will be the complete Shadow document. |
| 15 | TEXT ON SCREEN:  Show TEXT2 again. Highlight the line with /delete. | Finally, you can use the /delete to remove a Shadow document completely. There are a few other Shadow Topics available and I cover them in detail in the manual, and you can also read on the AWS website about them. |
| 16 |  | That's it for my introduction to AWS. In the next video, I'll show you how to create things, certificates and policies. |
| 17 | TEXT ON SCREEN:  Cypress Developer Community  community.cypress.com  Show video of email and twitter windows. | As always, you can post your comments and questions in the Wifi developer community, which I go to quite a bit and I'll answer there, or you are welcome to email me a private comment to alan\_hawse@cypress.com or tweet me @askioexpert. |

TEXT1:



TEXT2:

$aws/things/Thing1/shadow/update

$aws/things/Thing1/shadow/update/accepted

$aws/things/Thing1/shadow/update/rejected

$aws/things/Thing1/shadow/update/documents

$aws/things/Thing1/shadow/get

$aws/things/Thing1/shadow/get/accepted

$aws/things/Thing1/shadow/get/rejected

$aws/things/Thing1/shadow/delete

TEXT3:

Topic: $aws/things/Thing1/shadow/update

Message: {"state" : {"desired" : {"color" : "BLUE"}}}