Zero Touch Secure Provisioning Kit for AWS IoT

I. Introduction

This user's guide provides a detailed walkthrough of provisioning the **Zero Touch Secure Provisioning Kit** to connect and communicate with the **Amazon Web Services (AWS) IoT** service.

Zero Touch Secure Provisioning Kit

(Part Number: AT88CKECC-AWS-XSTK-B)

The kit consists of:

- SAM G55 Xplained Pro Evaluation Kit (Part Number: ATSAMG55-XPRO)

The **SAM G55 Xplained Pro** comes programmed with the AWS IoT Zero Touch firmware project. To update to the latest firmware or program another SAM G55, follow these steps:

1. Open **Atmel Studio 7** and open the Zero Touch firmware solution: `AWS_IoT_Zero_Touch_SAMG55.atsln`.
2. Plug the SAM G55 Xplained Pro into the computer via the EDBG USB Port.
3. Within Atmel Studio, use the **Debug > Start Without Debugging** menu option to rebuild and load the firmware onto the board.

- ATWINC1500 Xplained Pro Extension board (Part Number: ATWINC1500-XPRO)

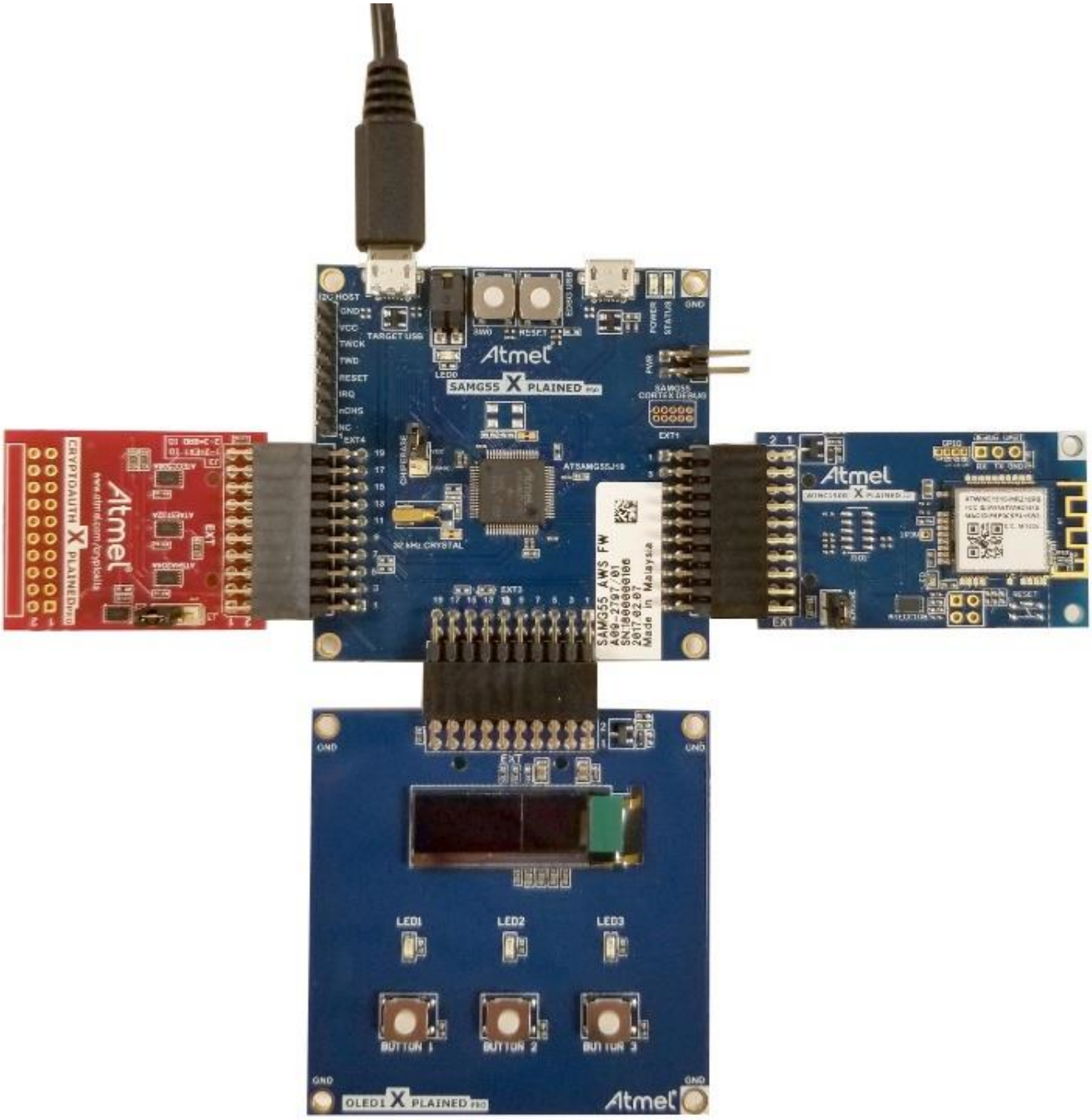
Ensure that the latest firmware is installed on the ATWINC1500. Instructions on how to upgrade the firmware are located on the ATWINC1500-XPRO product web page. Scroll to the bottom of the page and select 'Flash Memory Download Procedure'.

The latest firmware version for the ATWINC1500 is 19.5.4 (as of October 2017).

- OLED1 Xplained Pro Extension Kit (Part Number: ATOLED1-XPRO)
- CryptoAuth Xplained Pro Extension board (Part Number: ATCRYPTOAUTH-XPRO)

You will need:

- Two (2) Micro USB cables



What does "Zero Touch Secure Provisioning" mean?

One of the most difficult aspects of securing a device on the cloud is securely maintaining the keys.

- At manufacture time the keys must be installed in the device.
    - The Microchip Technology ATECC508A CryptoAuthentication Device securely maintains security keys.
    - The ATECC508A can be securely provisioned by Microchip Technology, eliminating loss of security keys.
- Certificates (Signer and User) are maintained securely inside the ATECC508A.

Industry standard cryptographic processes are hardware accelerated in the ATECC508A and ATWINC1500 ensuring a quick and secure connection.

The final product provides an ease of use connection to the Cloud.

What you will learn

- How to connect a device to AWS IoT
- Create a unique device identity for one or many devices
- Configuring AWS IoT for Just-In-Time Registration (JITR)
- How Zero Touch Secure Provisioning works
- Study the firmware: How the WINC1500 manages the overall TLS protocol the with ECC508A performing cryptographic primitives for TLS

Prerequisites

What you should know before opening the kit:

- Familiar with Public Key Infrastructure (PKI)
- AWS Services: AWS IoT, AWS Lambda, AWS IAM
- Transport Layer Security (TLS) security protocol

The Steps you will Follow

- Software Installation
- Create and Administer your own AWS Account
- Configure AWS Credentials
- AWS IoT JITR Setup
- Certificate Authority Setup
- Provision the Device
- AWS IoT Interaction
- Summary and Next Steps
- Troubleshooting

Glossary

- Keys - represents your individual identity (extremely sensitive; must be protected; secret)
- Provisioning - preparing a device to talk to the Cloud
- Certificate - a piece of paper that says something about you. However, you need an authority to (digital signature) cannot be forged - also tells AWS a little about yourself
- Certificate Authority - responsible for signing the certificate
- Transport Layer Security (TLS) - security protocol to communicate with AWS
- Register a device - in order to use AWS resources, you have to register ahead of time. JITR helps make this task easier.
- Just-In-Time Registration (JITR) - simplifies logistics by allowing devices to be registered individually at connection time.
- Secure element - A device that protects a device's identity and securely contains keys and through internal processes uses them in such a way that they cannot be revealed.

II. Software Installation

Project Software Files

The URL below will take you to the Zero Touch Secure Provisioning Kit product web page. A link to the latest version of the software files is located at the bottom of the page. The files are contained in a compressed file (*.ZIP). Download and install them on your computer.

- Zero Touch Secure Provisioning Kit Software Files

Note the location of the software library. The directory name is: `aws-iot-zero-touch-kit`

AWS Command Line Interface (CLI)

You will be using the *AWS Command Line Interface* to manage your AWS services. Go to the following URL to find the **Windows** installer:

- https://aws.amazon.com/cli/

Terminal Emulator

You will use a terminal emulator to monitor the Zero Touch Secure Provisioning Kit. Popular choices are TeraTerm and PuTTY.

- TeraTerm - https://ttssh2.osdn.jp/index.html.en
- PuTTY - https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

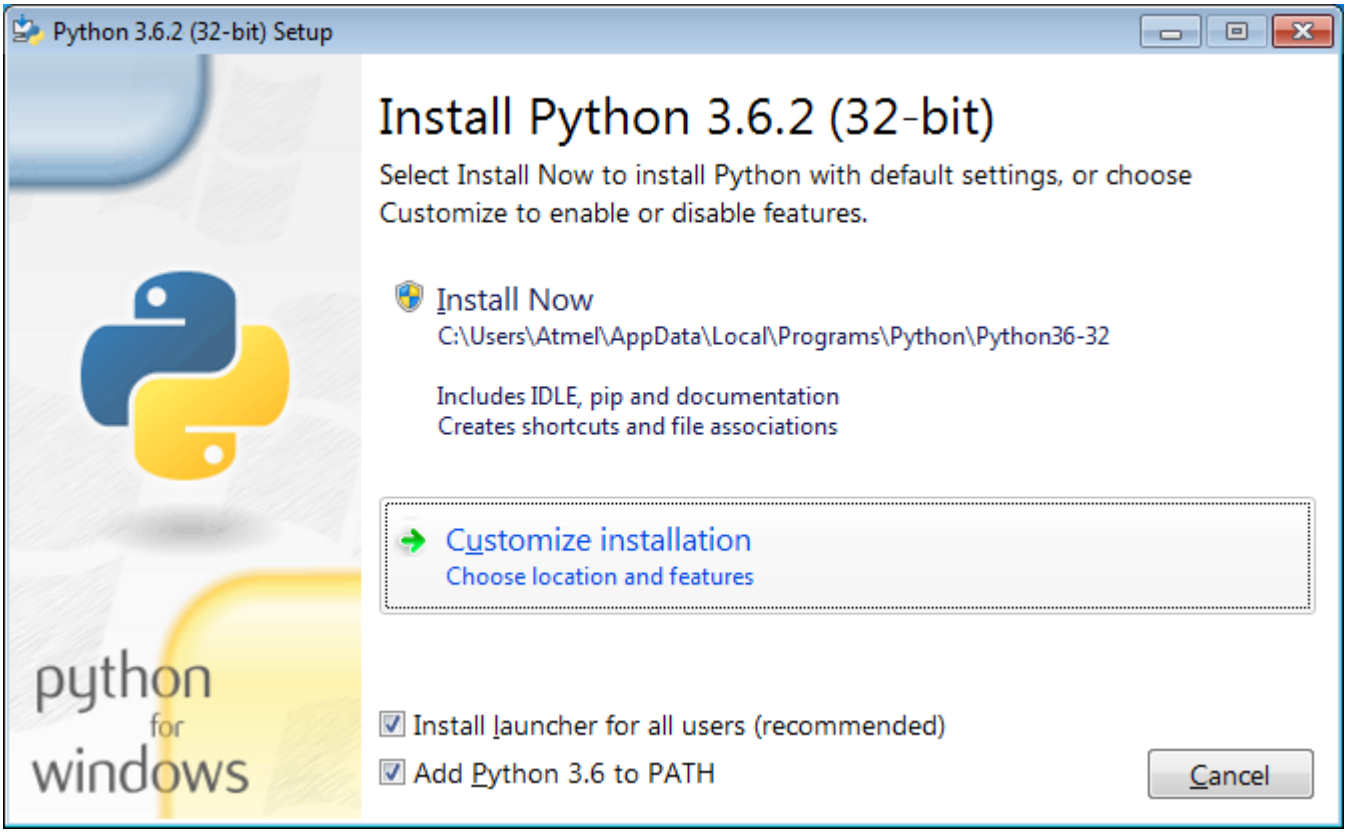
Python 3.6.x

You will be using Python scripts to assist you in configuring your AWS account to communicate with your Zero Touch Secure Provisioning Kit. You can view the Python scripts to see the detailed steps involved.

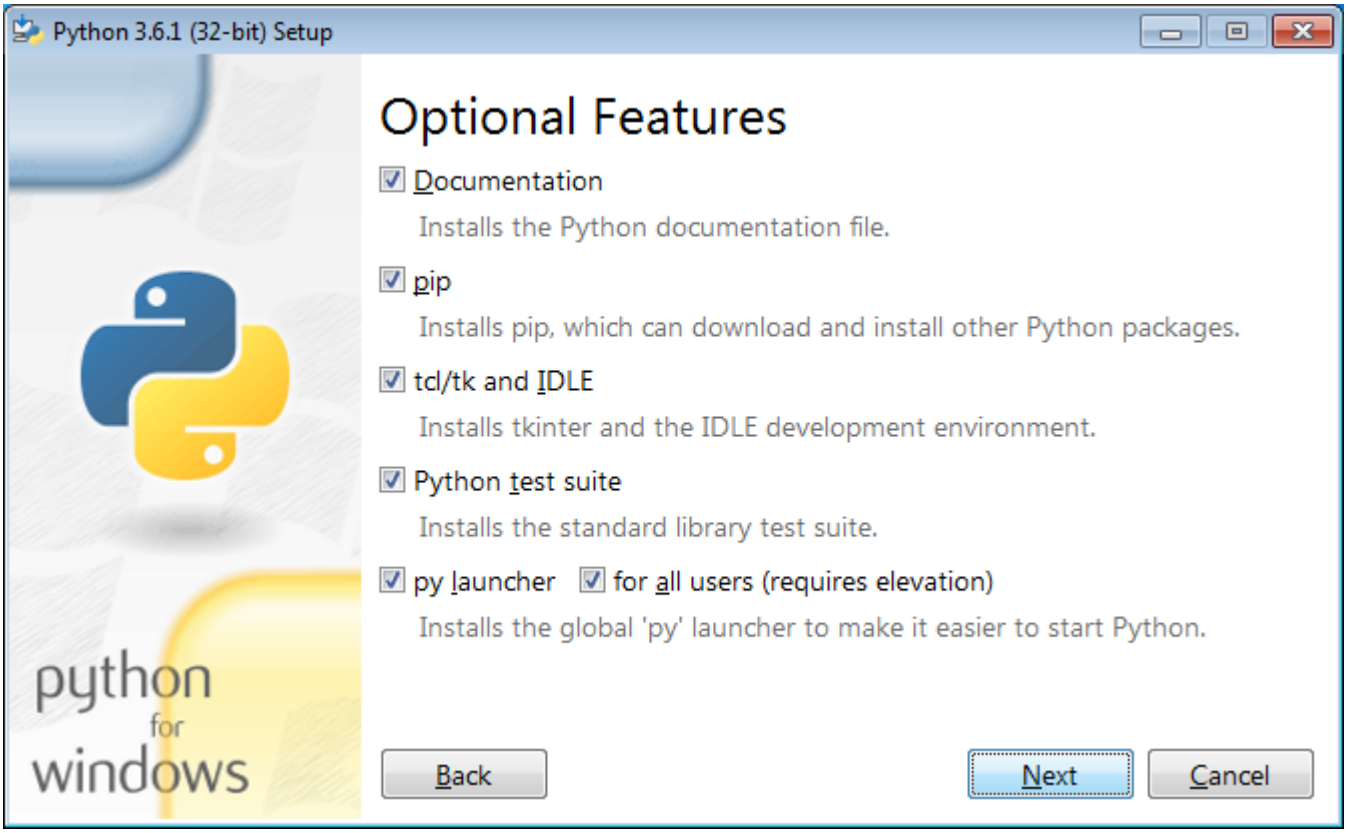
The latest version of Python as of this writing is 3.6.2.

- https://www.python.org/downloads/

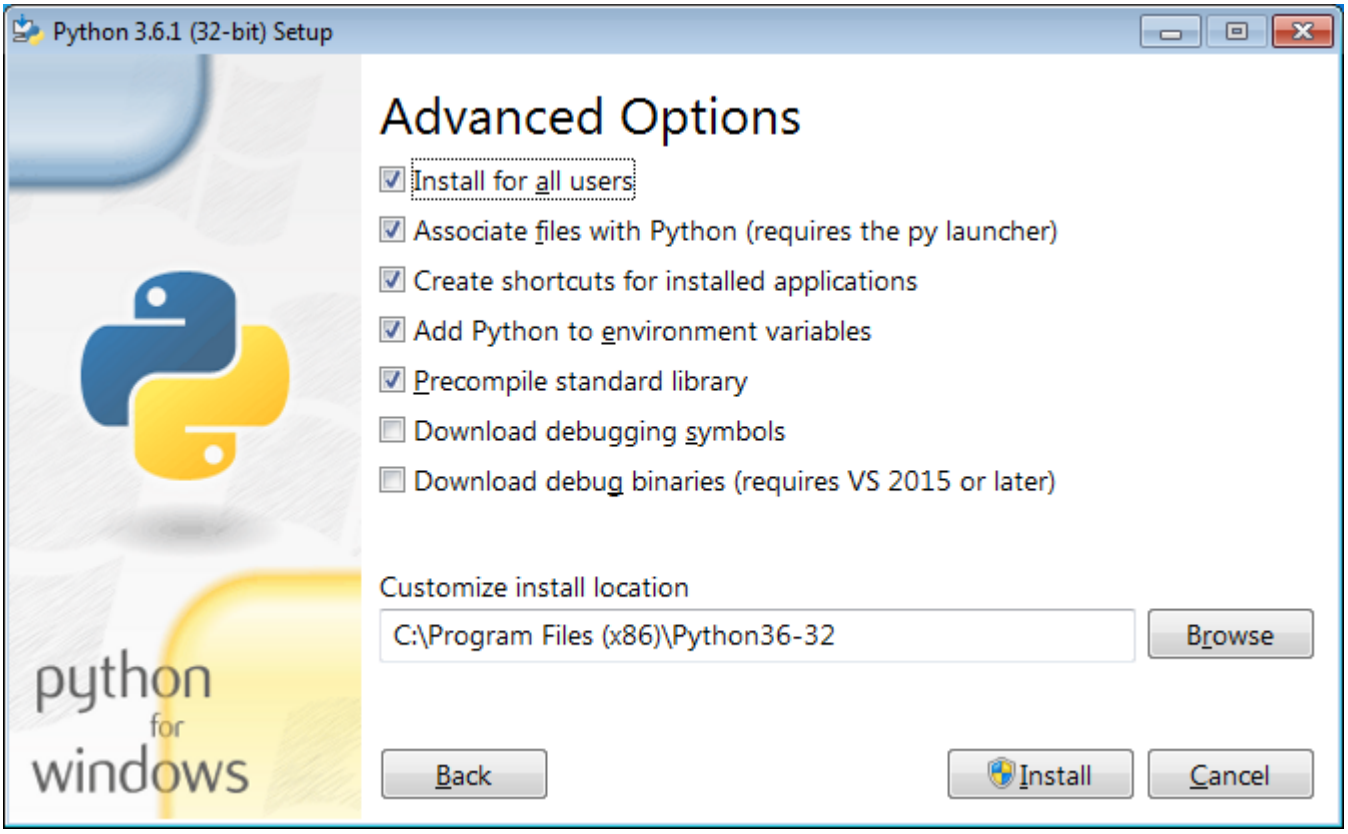
When installing Python, check 'Add Python 3.6 to PATH'.



Choose 'Customize Installation' and make sure everything is selected.



Click **Next**, then select 'Install for all users' and 'Precompile standard library'.



Click **Install**.

Visual C++ 2015 Build Tools

You may already have these tools installed. They are needed for the Python packages (to be installed next).

- http://landinghub.visualstudio.com/visual-cpp-build-tools

Python Packages

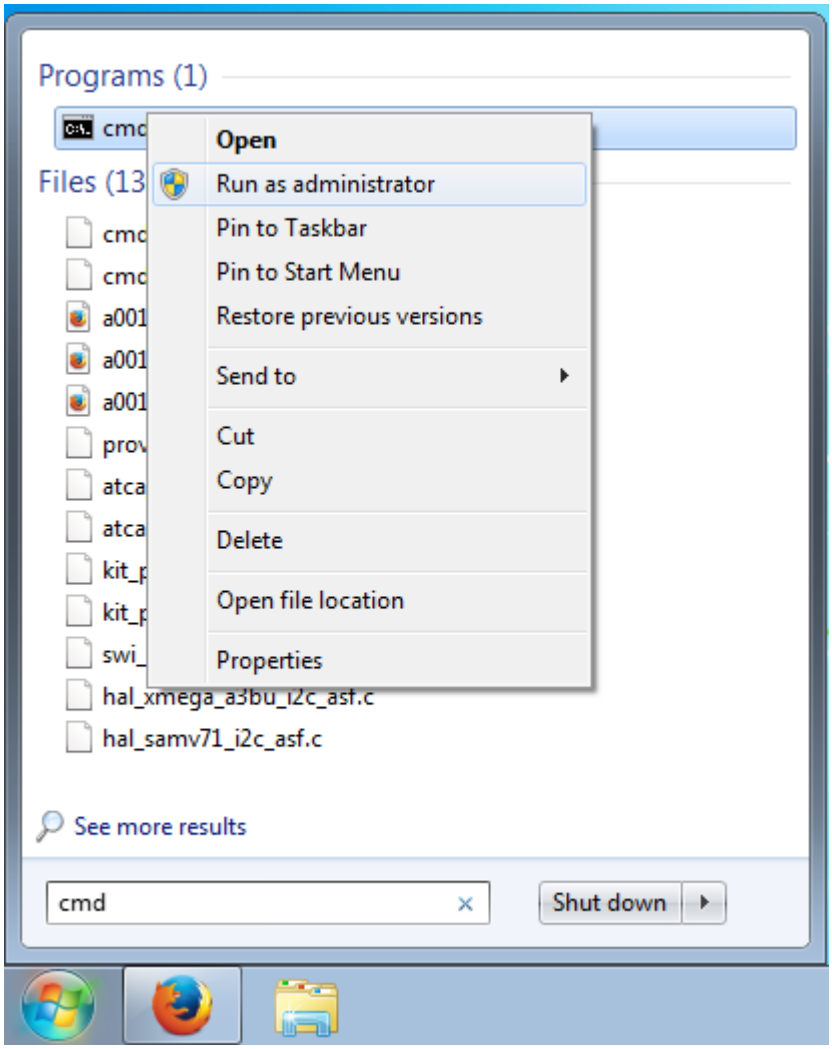
You will be using the Python package manager (pip) to install the required packages used in this guide.

Locate `requirements.txt` in the Project Software files you installed earlier:

- `aws-iot-zero-touch-kit\requirements.txt`

These packages will be installed from an administrative command prompt.

- Open the start menu (bottom left window) and search for 'cmd'
- Right-click on 'Command Prompt (CMD)' and select 'Run as Administrator'



From the CMD, navigate to the directory and run the following command:

```
pip install -r requirements.txt
```

It may take a while to install.

Optional Software Packages

The following programs are not required, but can be useful:

OpenSSL

- https://wiki.openssl.org/index.php/Binaries

Standard software for working with certificates and keys.

Notepad++

- https://notepad-plus-plus.org/

Text editor with good syntax highlighting for a variety of files.

ASN.1 Editor

- https://www.codeproject.com/Articles/4910/ASN-Editor

Tool for inspecting and editing ASN.1 data including X.509 certificates.

Let's summarize what you have done so far:

- You have installed the software needed to administer your AWS account and communicate with the Zero Touch Secure Provisioning Kit.

III. Create and Administer your own AWS Account

Amazon Web Services (AWS) provides computing services for a fee. Some are offered for free on a trial or small-scale basis. By signing up for your own AWS account, you are establishing an account to gain access to a wide range of computing services.

Think of your AWS account as your root account to AWS services. It is very powerful and gives you complete access. Be sure to protect your username and password.

You control access to your AWS account by creating individual users and groups using the **Identity and Access Management (IAM) Console**. From the IAM Console, you also assign policies (permissions) to the group.

For the Zero Touch Secure Provisioning Kit, you will be creating a user (ZTUser) and a group (ZTGroup). Once created, you log into the ZTUser account to administrate the Zero Touch Secure Provisioning Kit.

Amazon AWS provides a wealth of documentation and instructions in the form of getting started guides and videos. We encourage you to explore these to learn more about what Amazon AWS can provide for you.

- AWS 10-Minute Tutorials

The specific AWS services you will use for the Zero Touch Secure Provisioning Kit are:

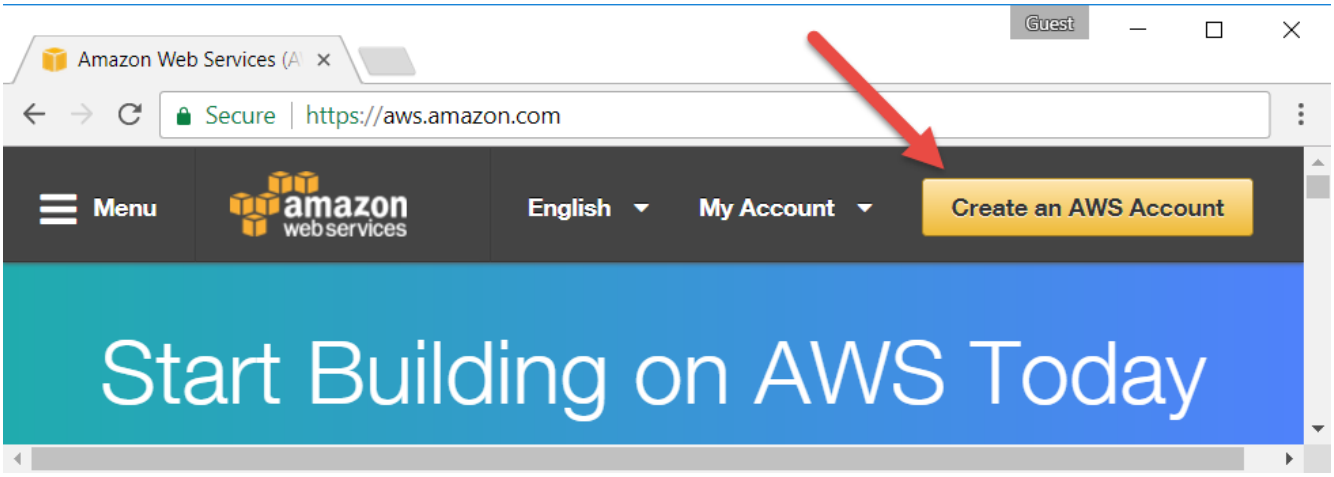
- AWS IoT
- AWS Lambda

1

Create your own AWS account

Click on the URL below and follow the instructions to create your own AWS account:

- https://aws.amazon.com/



2

Sign in to the AWS Console to manage user access and permissions

Once your AWS account is created and the next time you visit the https://aws.amazon.com/ URL, you will see a new button:

Sign into your AWS account by clicking on the **Sign In to the Console** button and entering your username and password.

You will limit access to the Zero Touch Secure Provisioning Kit by creating a user (ZTUser) that you will later log into and administer.

3

Access the IAM Console

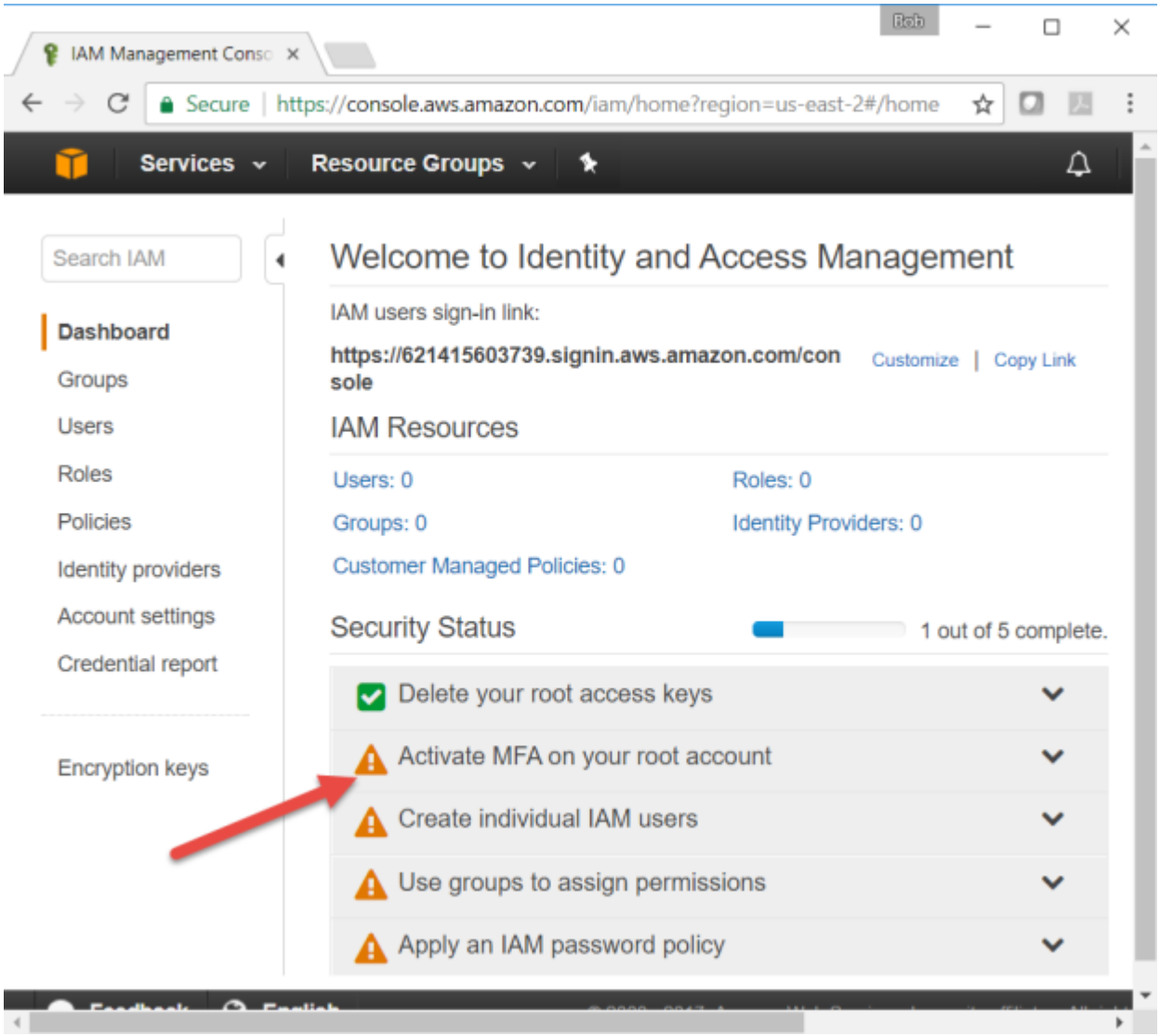
IAM enables you to control access to your AWS account. By using IAM, you will create and manage AWS users and groups and assign policies (permissions) to control access to AWS services and resources. A policy is a document that formally states one or more permissions.

a

From your AWS Console, type `IAM` in the search box. Click on the link that takes you to the IAM Console.

b

(***Highly Recommended***) Click on 'Activate MFA (Multi-factor Authentication) on your root account'.



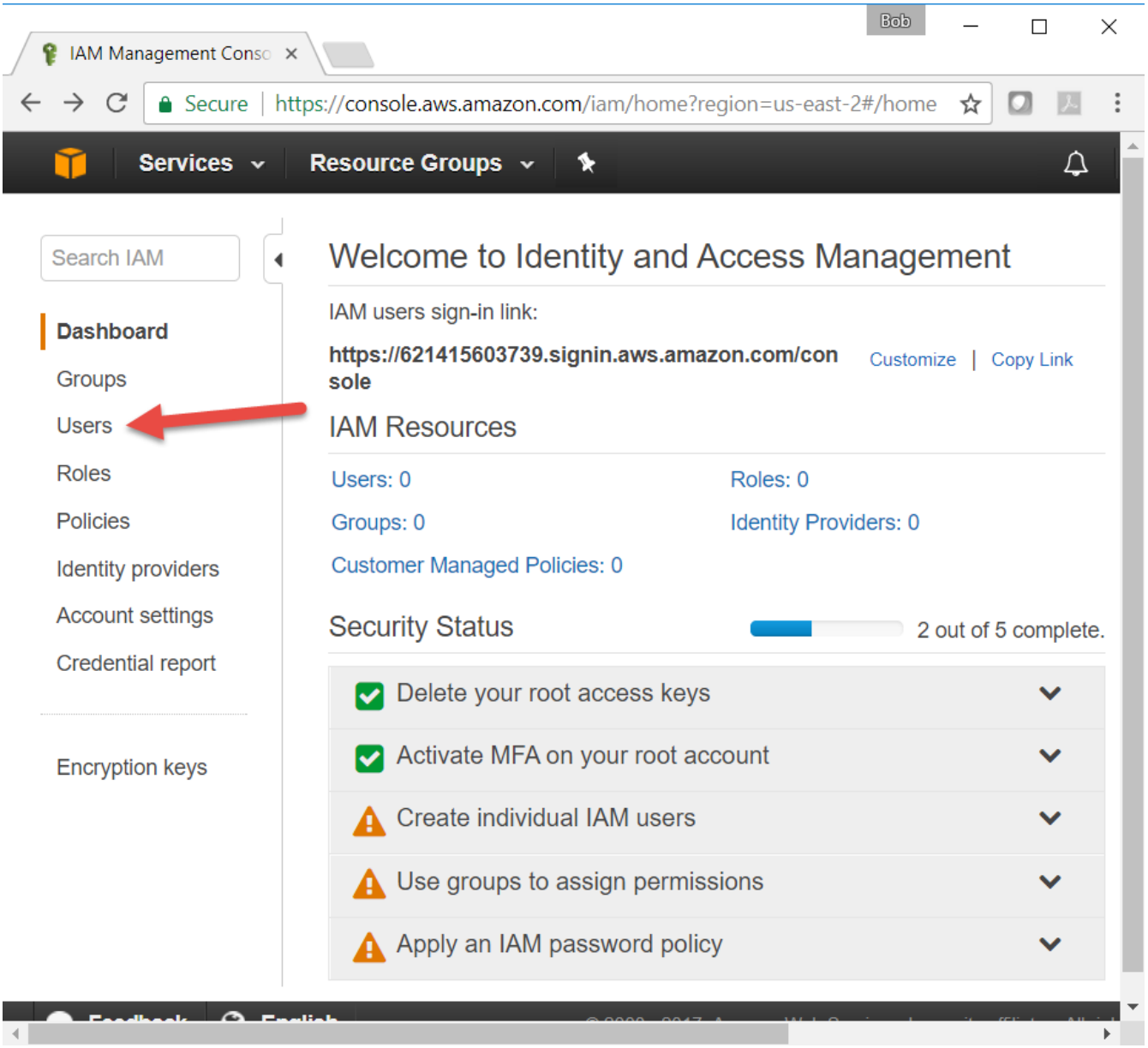
- This is an important step to better secure your root account against attackers. Anyone logging in not only needs to know the password, but also a constantly changing code generated by an MFA device.
- AWS recommends a number of MFA device options at the following link: https://aws.amazon.com/iam/details/mfa/
- The quickest solution is a virtual MFA device running on a phone. These apps provide the ability to scan the QR code AWS will generate to set up the MFA device.

c

Create a new user for your AWS account.

You will be performing a four step process to create user ZTUser. During this process, you will also be creating a new group ZTGroup to assign policies to and assign ZTUser to the ZTGroup and its associated policies.

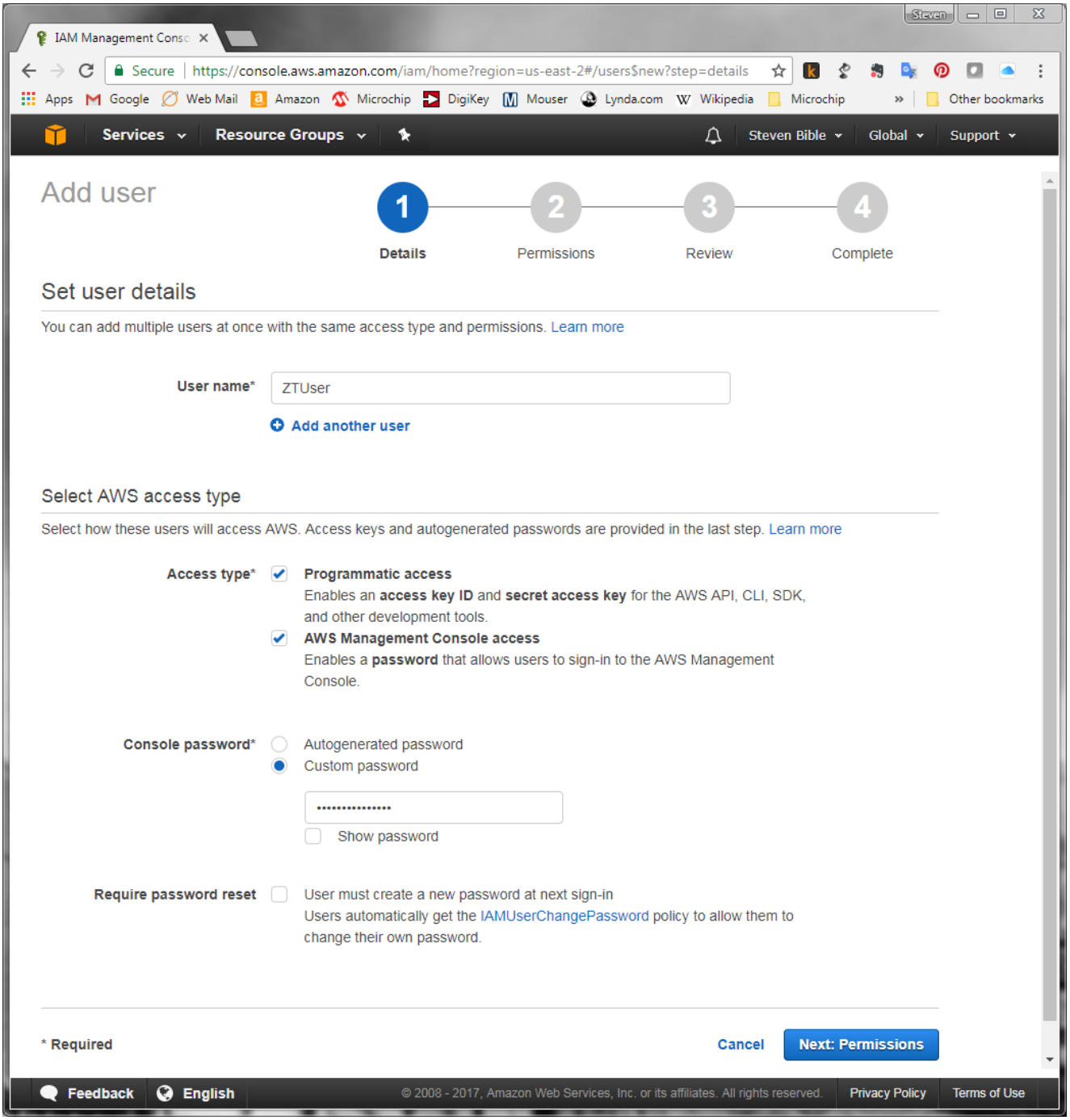
- In the IAM Console window, click on 'Users'.



From the "Users" management page, click on the **Add user** button at the top of the page.

When the "*Add user - Step 1: Details*" page is displayed, enter the following information:

- Set user details:
  - Username: `ZTUser`
- Select AWS access type:
  - Access type:
    - Select 'Programmatic access'
    - Select 'AWS Management Console access'
  - Console password:
    - Select 'Custom password'
    - Enter a password for user ZTUser.
    - Un-select 'Require password reset'
    - Record the password for logging in to the console later
- Click on the **Next: Permissions** button at the bottom of the page



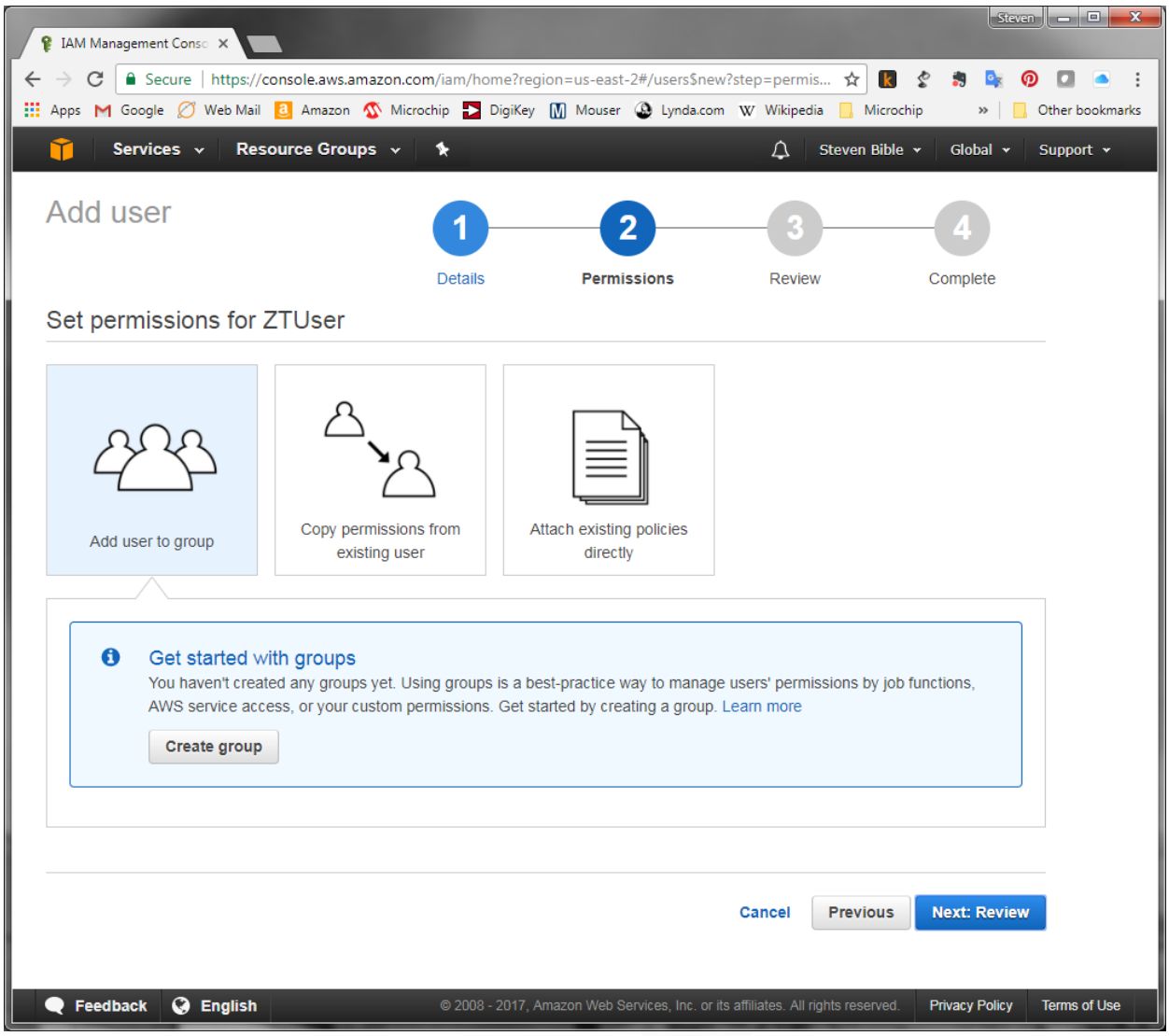
d

Create a new group for your AWS account

"*Add user - Step 2: Permissions*" for adding a new user requires you to assign permissions to ZTUser. This is done by creating a group and selecting policies you specify for the group and add user(s) to the group.

- Click on **Create group**.

You can also create new groups from the IAM Console by clicking on 'Groups'.



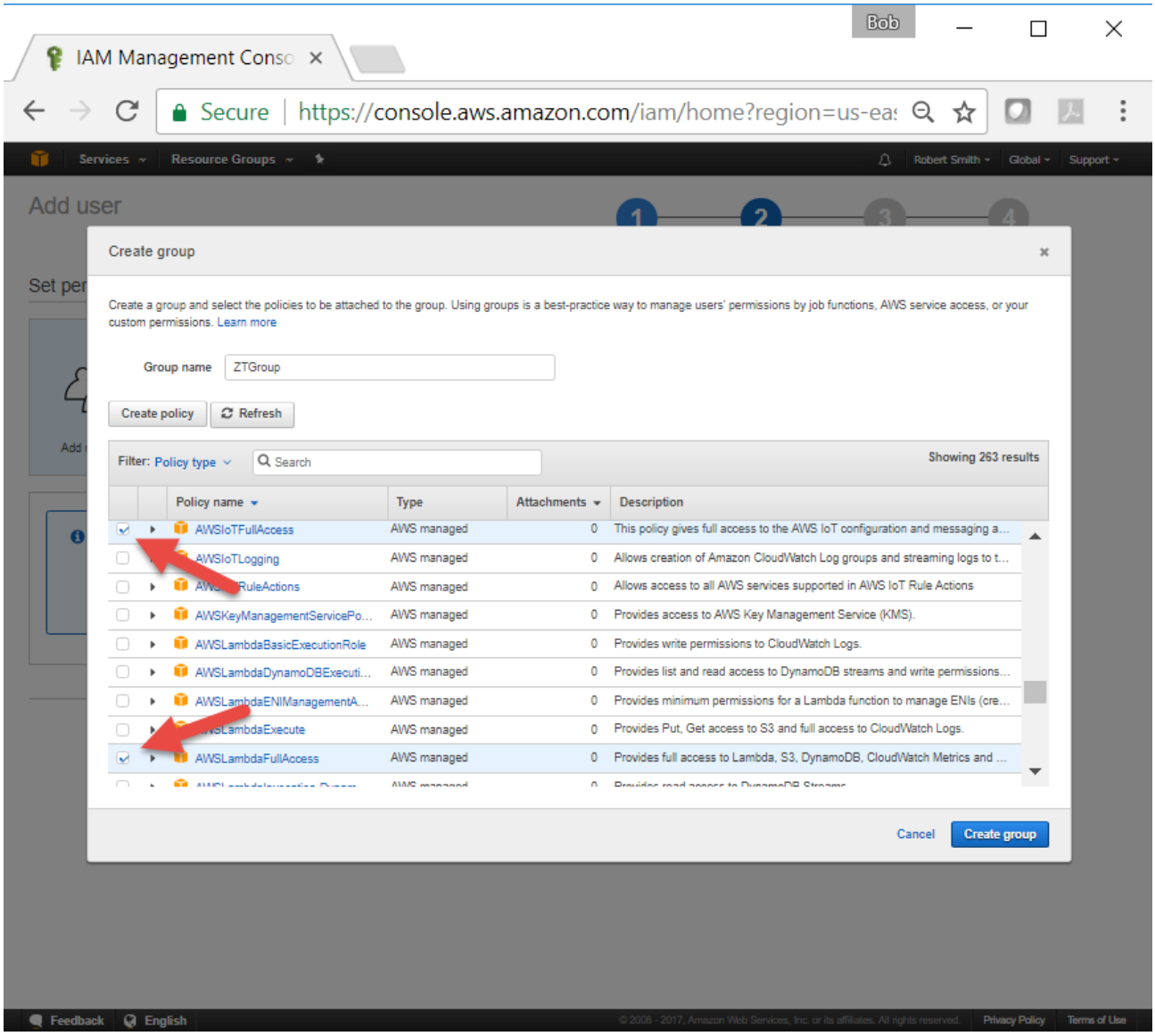
From the *Create group* window, enter the group name: `ZTGroup`.

Next, we want to attach the following policy types:

Attached policy types:

- 'AWSIoTFullAccess'
- 'AWSLambdaFullAccess'

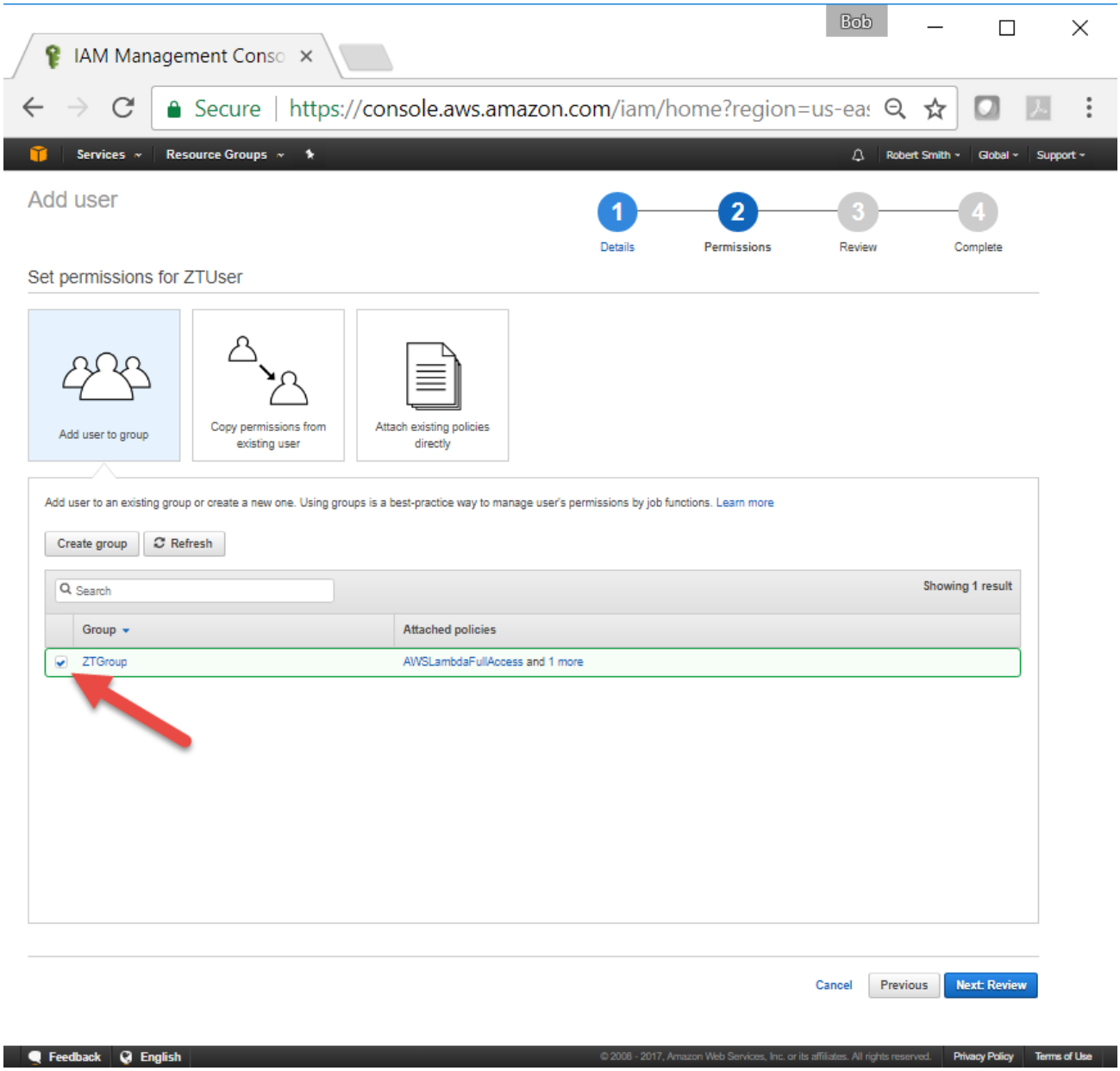
Click on the **Create Group** button at the bottom of the window.



You are now back at the "*Add user - Step 2: Permissions*" page.

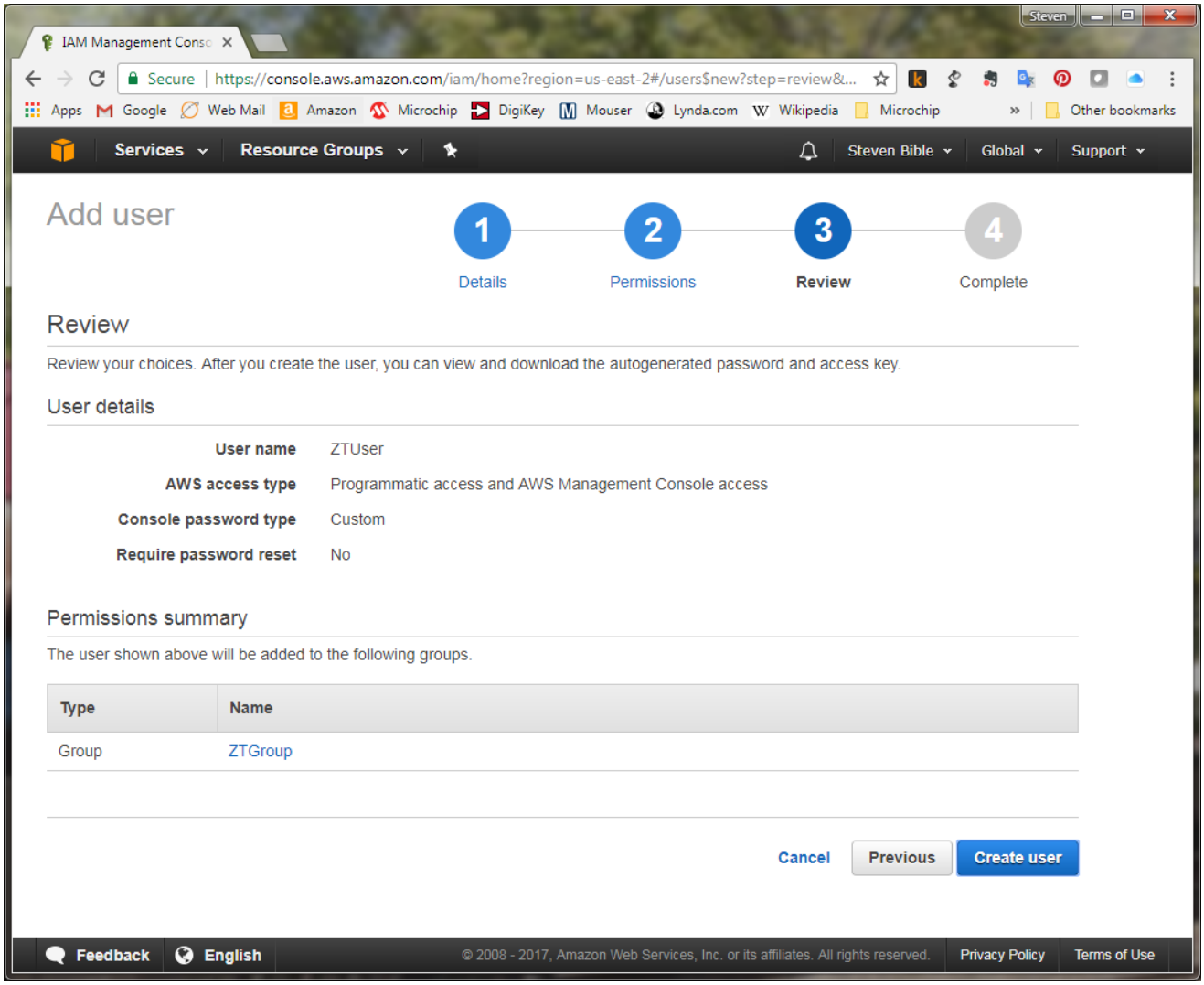
Notice that ZTGroup is selected for you. This sets permissions for user ZTUser to group policies specified to ZTGroup.

- Click on the **Next: Review** button at the bottom of the page.



e

The "*Add user - Step 3: Review*" page is displayed. Review your choices. When you are satisfied that your entries are correct, click on the **Create user**button at the bottom of the page.

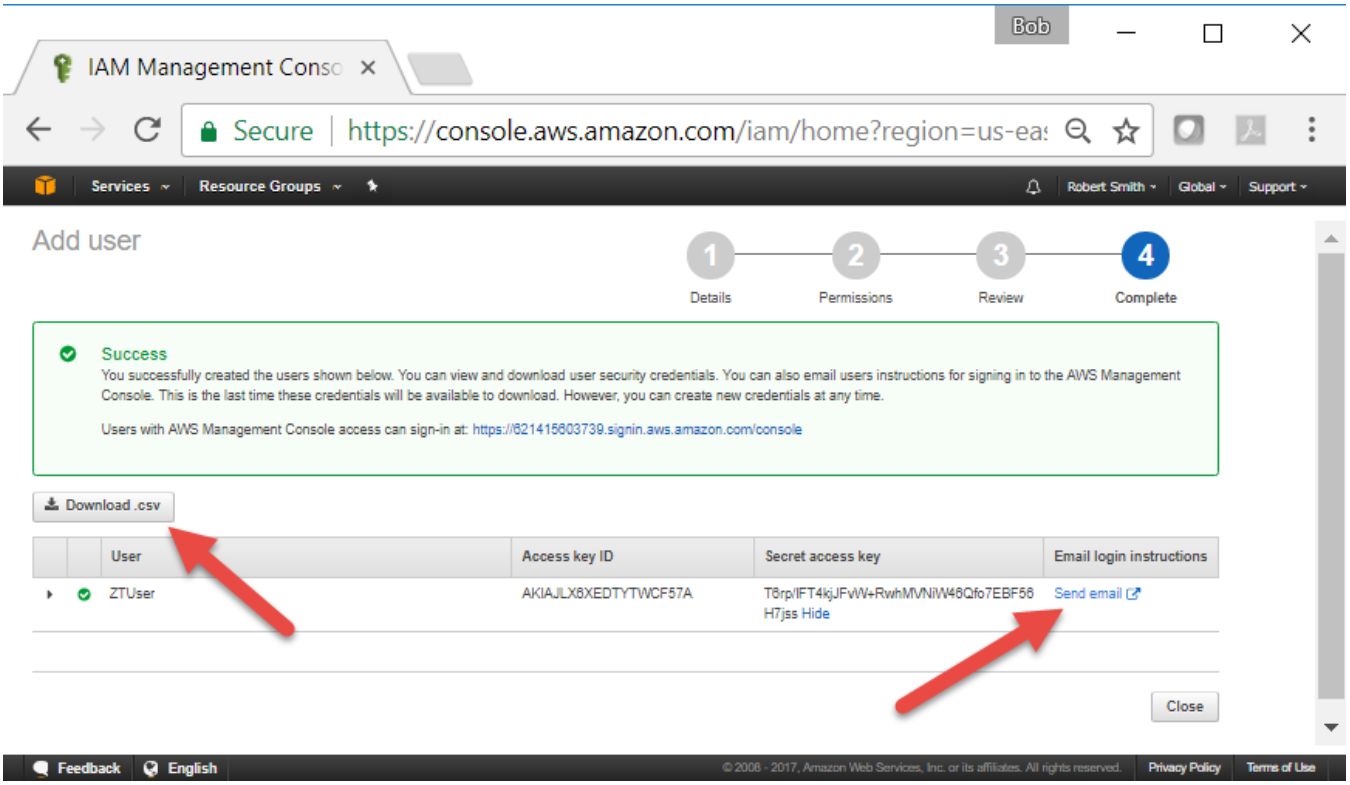


f

The "*Add user - Step 4: Complete*" page is displayed.

AWS creates a unique account sign-in URL and access credentials (Access key ID and Secret access key). ***Save this information***. There are two ways to get easy access to these security credentials:

- Download a `*.csv` file
- Send an email to yourself



In a later step, you will use these credentials to configure and use the account under user ZTUser.

Just-In-Time Registration (JITR)

Just-In-Time Registration (JITR) allows you to register a device at connection time. JITR reduces the manufacturing burden of registering a device with AWS before it is connected.

In a later step, you will create a Lambda function that will be responsible for registering new devices.

In the next two steps, you will create a custom policy and role that will be used by the JITR Lambda function.

4

Create a JITR Lambda Function Policy

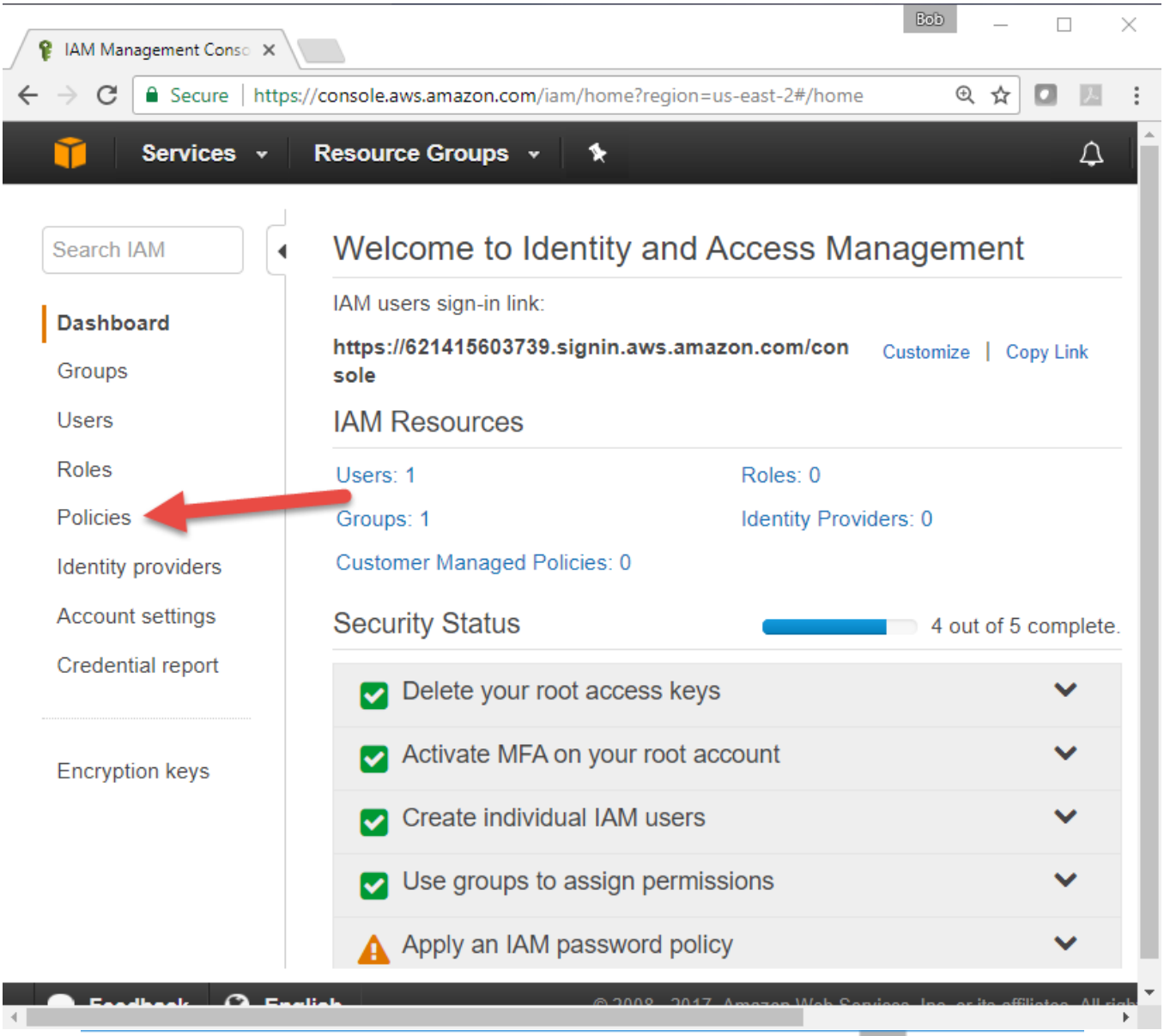
To assign permissions to a user, group, role, or resource, you create a policy, which is a document that explicitly lists permissions. In its most basic sense, a policy lets you specify the following:

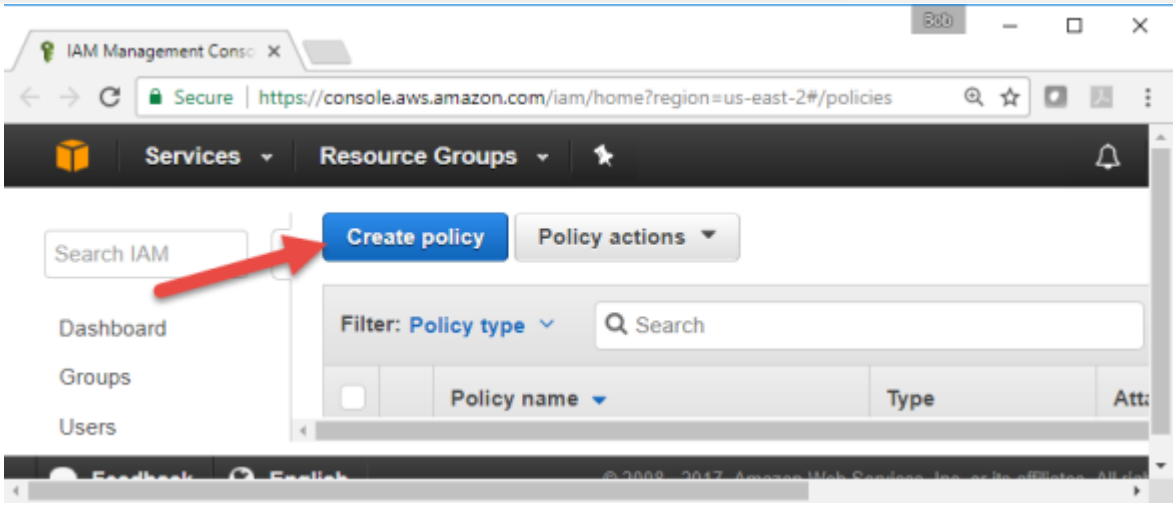
- **Actions**: what actions you will allow. Each AWS service has its own set of actions. Any actions that you don't explicitly allow are denied.
- **Resources**: which resources you allow the action on. Users cannot access any resources that you have not explicitly granted permissions to.
- **Effect**: what the effect will be when the user requests access—either allow or deny. Because the default is that resources are denied to users, you typically specify that you will allow users access to a resource.

Reference: Overview of IAM Policies

a

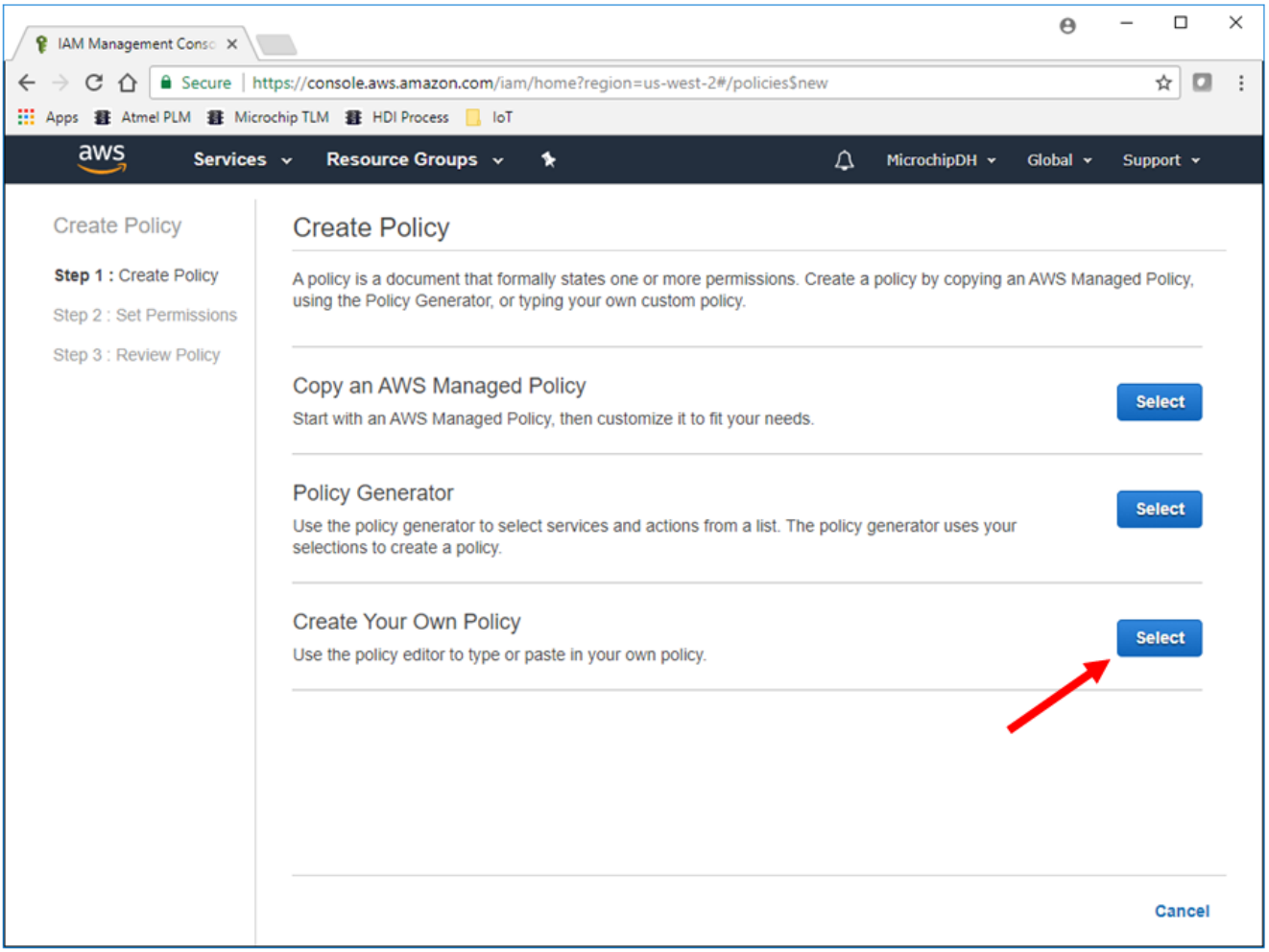
From the IAM Console, click on 'Policies' then **Create policy**





b

Select 'Create Your Own Policy'



c

Policy Name: `ZTLambdaJITRPolicy`

d

Description: none

e

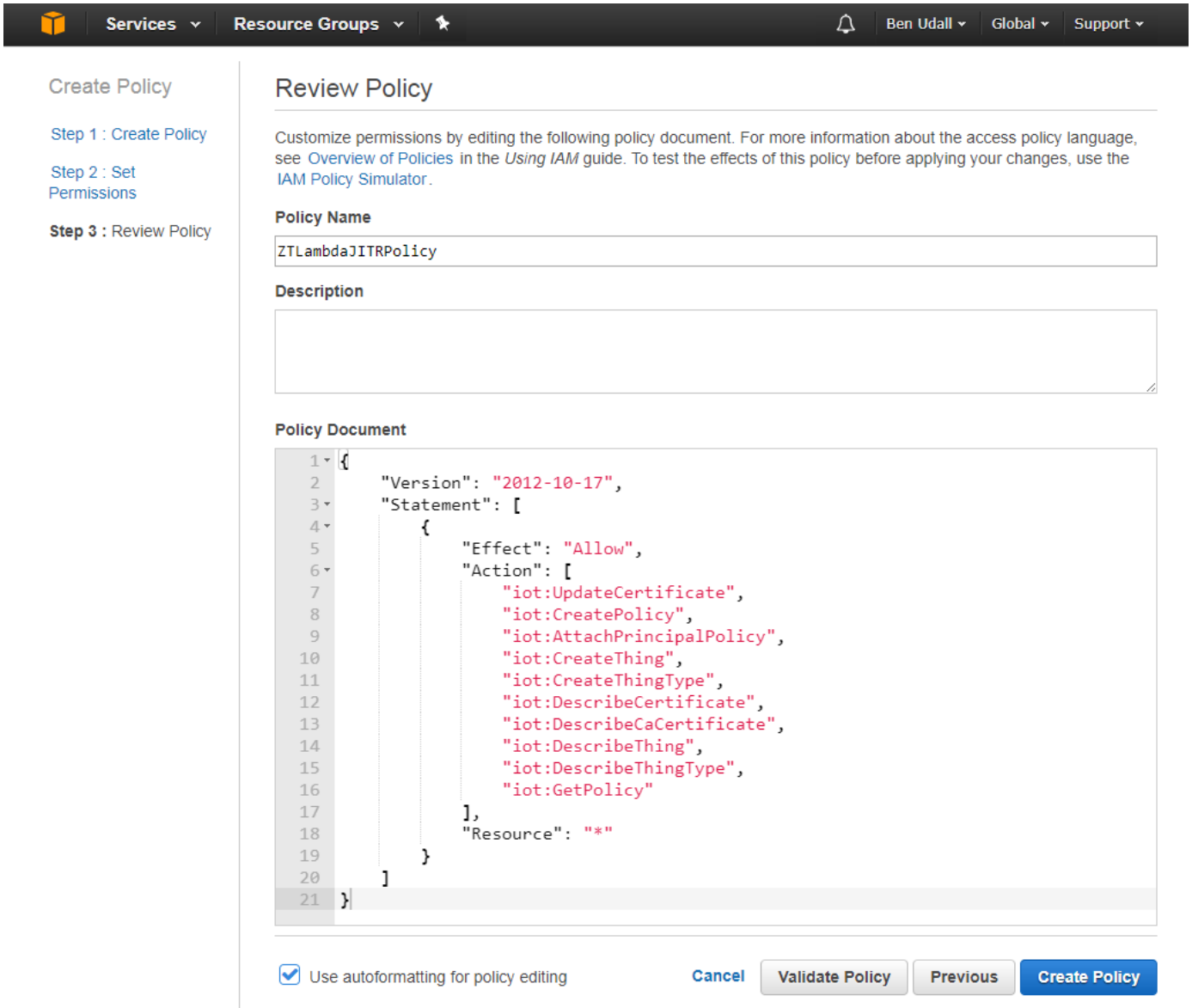
Cut and paste the following code into 'Policy Document':

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCertificate",
                "iot:CreatePolicy",
                "iot:AttachPrincipalPolicy",
                "iot:CreateThing",
                "iot:CreateThingType",
                "iot:DescribeCertificate",
                "iot:DescribeCaCertificate",
                "iot:DescribeThing",
```

```
                "iot:DescribeThingType",
                "iot:GetPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

4

Click on the **Create Policy** button at the bottom of the page



5

Create a JITR Lambda Function Role

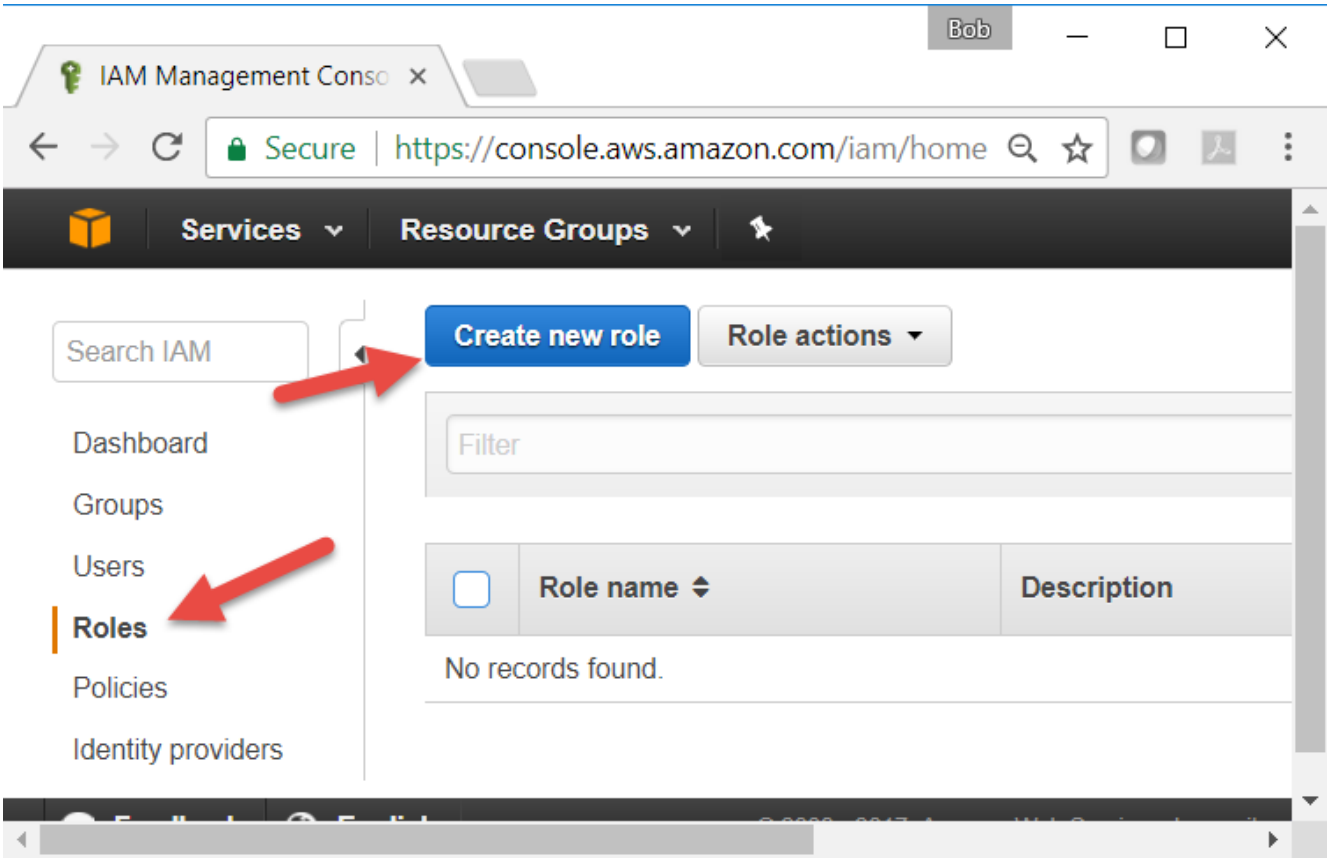
An **IAM role** is similar to a user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have any credentials

(password or access keys) associated with it. Instead, if a user is assigned to a role, access keys are created dynamically and provided to the user.

Reference: IAM Roles

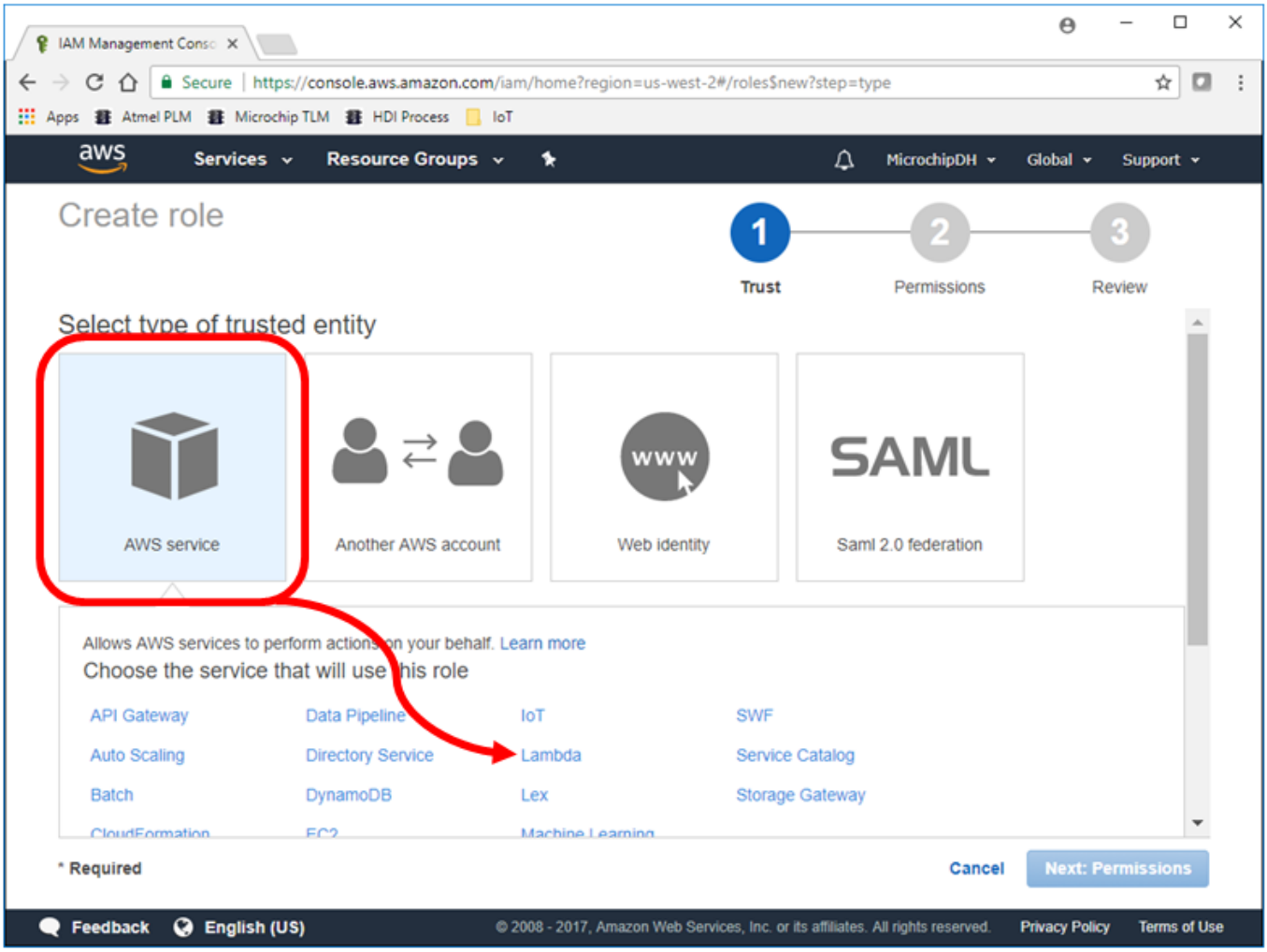
a

From the IAM Console, click 'Roles' then click **Create new role**



b

Under 'Select role type', select the 'AWS Service ' box and select 'Lambda' service, then click the 'Next: Permissions' button.



c

Attach the following policies:

- AWSLambdaBasicExecutionRole
- AWSXrayWriteOnlyAccess
- ZTLambdaJITRPolicy

d

Click the **Next Step** button at the bottom of the page.

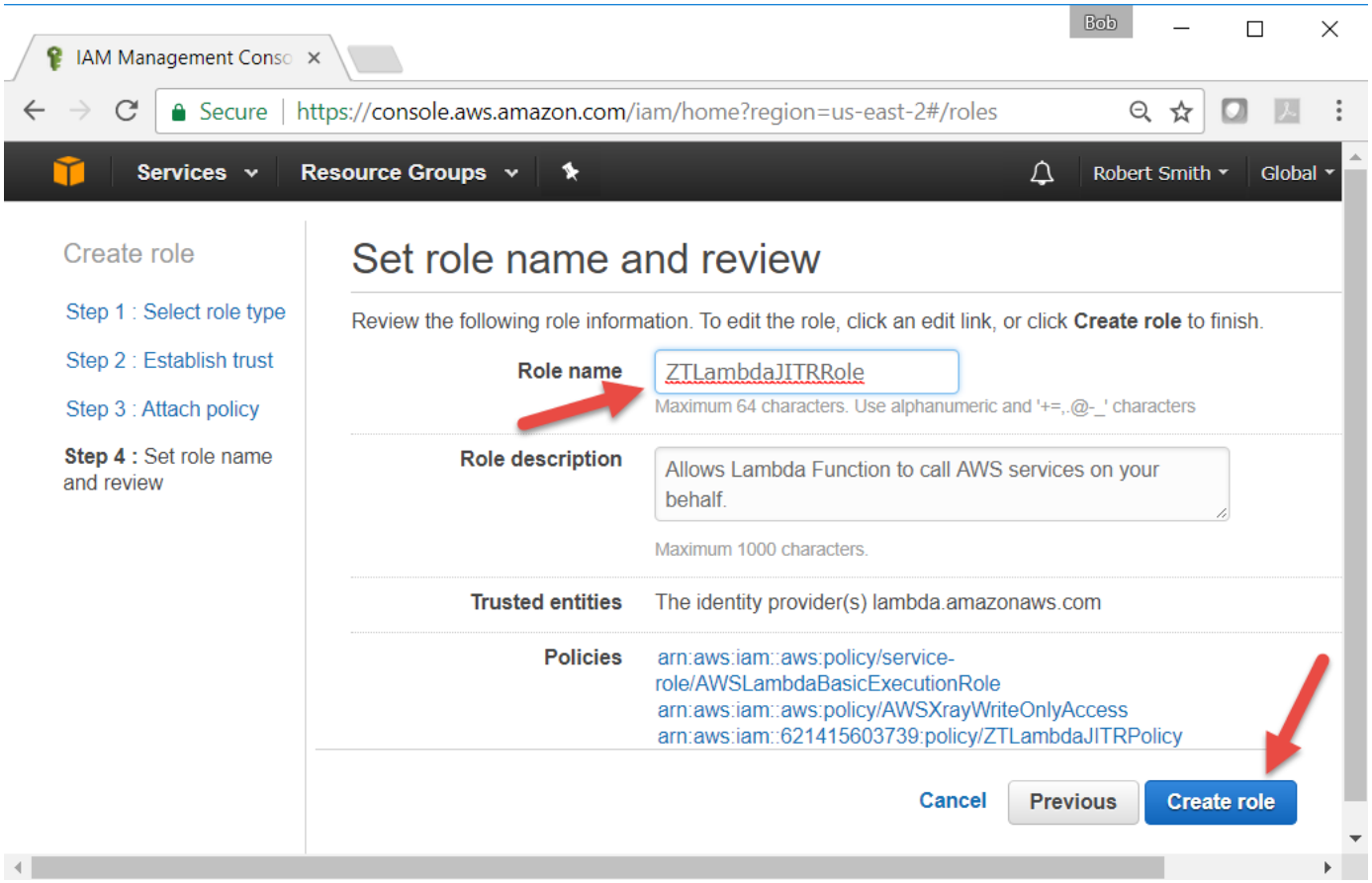
e

Set role name and review:

- Role Name: `ZTLambdaJITRRole`

f

Click on the **Create role** button at the bottom of the page.



Let's summarize what you have done so far:

- You created an AWS account
- Created a user, ZTUser
- Created a group, ZTGroup and attached two policy types (AWSIoTFullAccess and AWSLambdaFullAccess)
- Assigned user ZTUser to group ZTGroup
- Created a lambda function policy ZTLambdaJITRPolicy and role ZTLambdaJITRRole

In the next step, you will use the credentials that AWS gave you to configure the **AWS Command Line Interface (CLI)** tool.

IV. Configure AWS Credentials

Before you can perform actions with your AWS account, you need to configure the AWS CLI tool with the appropriate user AWS credentials. These user credentials (Access Key ID and Secret Access Key) were given to you when you created ZTUser. Once the AWS CLI is configured, the Zero Touch Secure Provisioning Kit's Python scripts can use the credentials to further configure your AWS account to communicate with the kit.

The AWS CLI is a unified tool to manage your AWS services. You can control multiple AWS services from the command line and automate them through scripts.

Reference: AWS Command Line Interface

The kit's Python scripts perform actions with your AWS account within a region. In order to perform these actions, we need credentials for a user which has permission to perform these actions. You will give the Python scripts permission to:

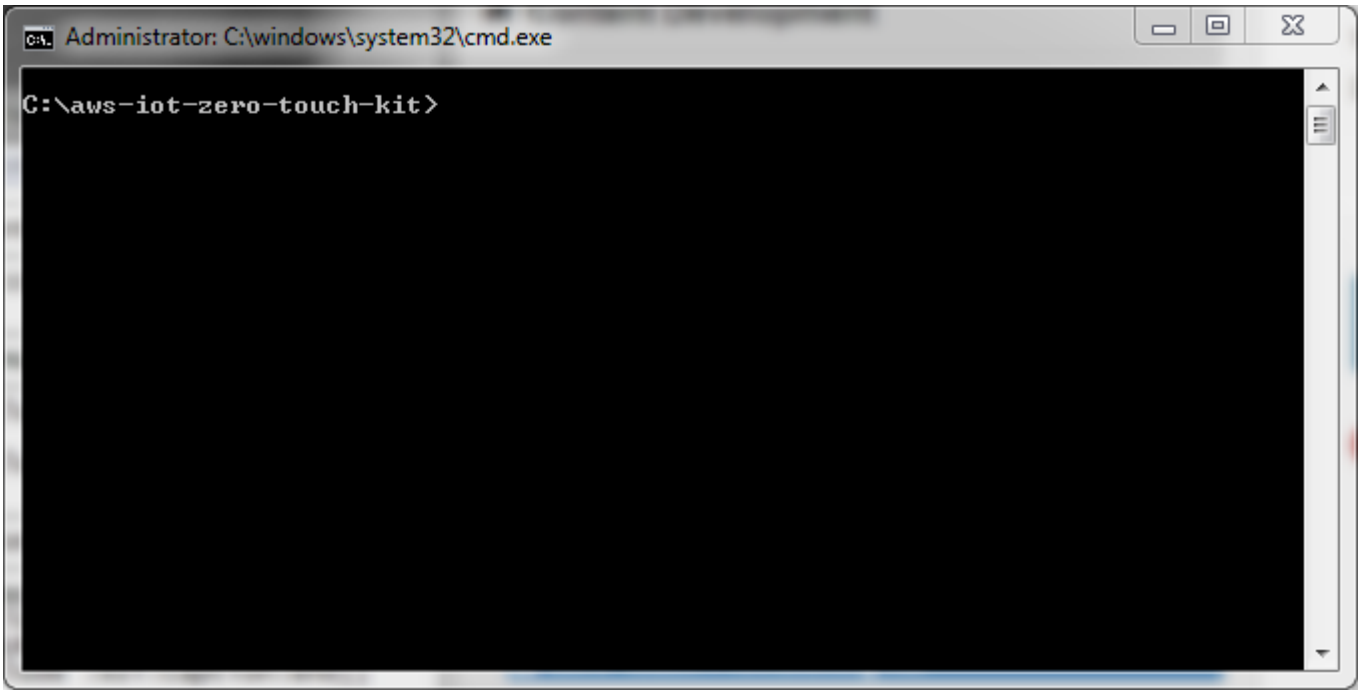
- register Certificate Authorities (CA) within AWS IoT
- access "thing" shadow documents with AWS IoT

Amazon AWS refers to a "thing" as a device that communicates with the AWS IoT service.

1

Open a *Command* window and browse to the following location:

```
aws-iot-zero-touch-kit\
```



2

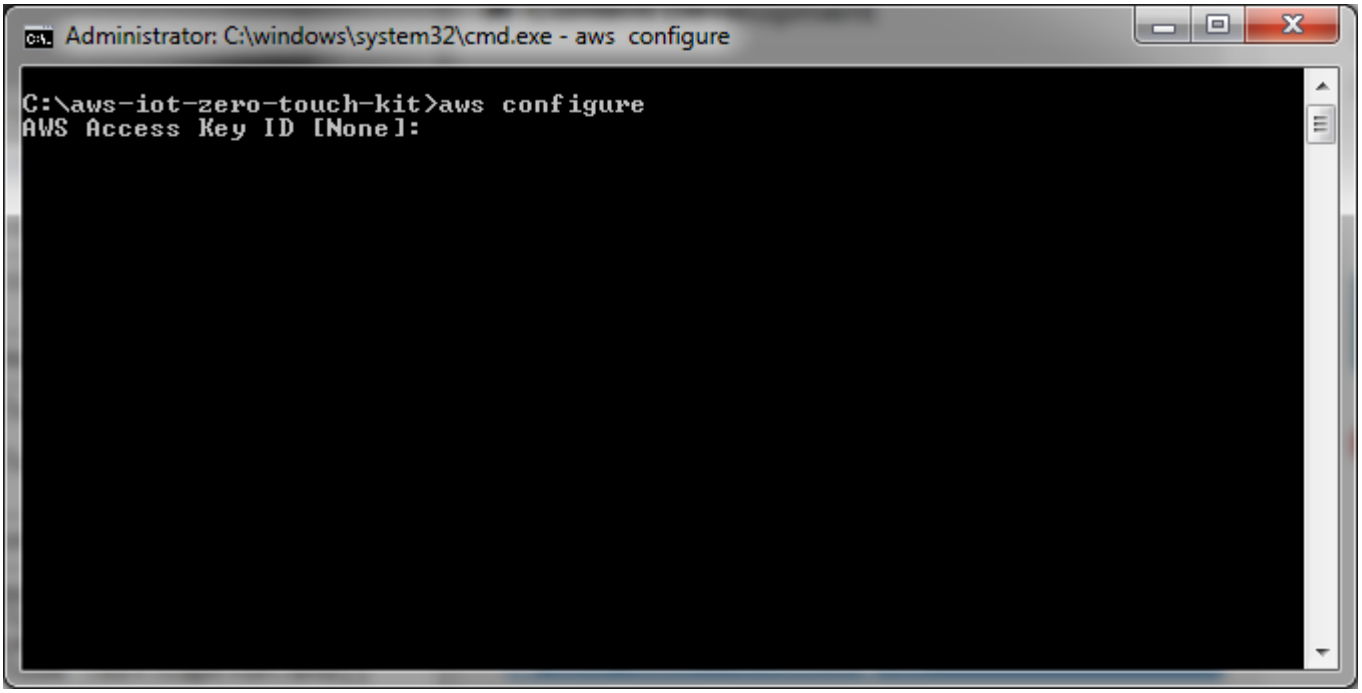
From the command prompt, run the following command:

```
aws configure
```

3

Enter your Access Key ID and Secret Access Key when prompted. You should copy and paste the credentials to avoid any typing mistakes.

Pasting in the command prompt is performed by right-clicking and selecting the 'Paste' option.



You will see the following results:

```
>aws configure
AWS Access Key ID [None]: ACCESSKEYID
AWS Secret Access Key [None]: SECRETACCESSKEY
Default region name [None]: us-west-2 ( <-- Enter the region that you selected )
Default output format [None]:
```

Once configured, these settings will be used by both the AWS CLI and Python scripts.

More information can be found at the following links:

- Configuring the AWS CLI
- Boto 3 Configuration

Let's summarize what you have done so far:

- You configured the AWS CLI with ZTUser's credentials.

This is a one time step.

V. AWS IoT Just-In-Time Registration Setup

In Step III you created the JITR Lambda function role which defined what services the Lambda function is allowed to access.

In this step, you will create a Lambda function responsible for registering new devices. You will also create a trigger from the AWS IoT rules engine so that your Lambda function will execute each time a new device connects. The trigger will execute a Lambda function to perform the following :

- The device identifies itself to AWS
- AWS reads the unique device name from its certificate
- Create a policy and attached it to the device certificate
- Create a "thing" which represents a single IoT device
- Activate the device's certificate

- AWS Lambda is a computing service that runs code in response to events and automatically manages the computing resources required by that code.

Reference: AWS Lambda

- AWS IoT is a managed cloud platform that lets connected devices easily and securely interact with cloud applications and other devices.

Reference: AWS IoT

You will log in as ZTUser using the credentials that you saved in "III. Create and Administer your own AWS Account."

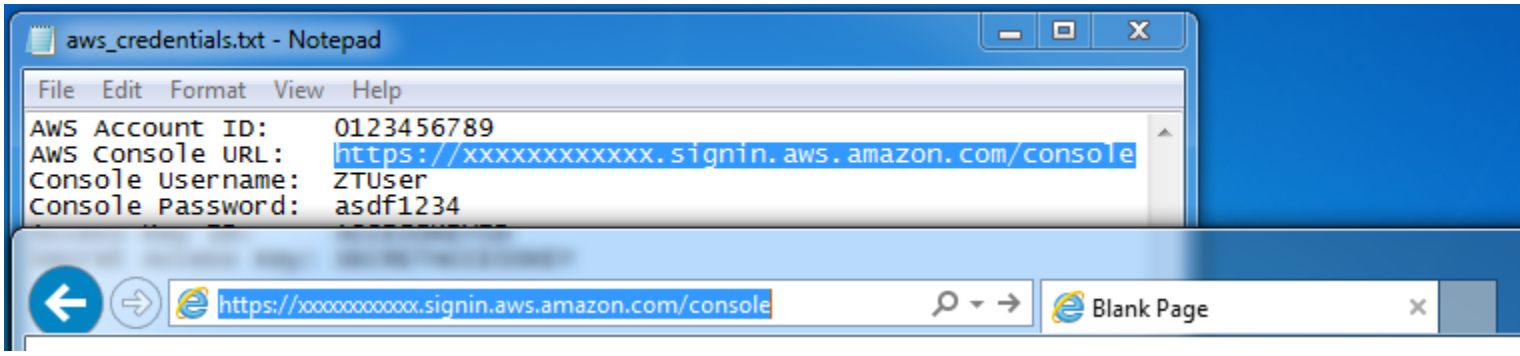
1

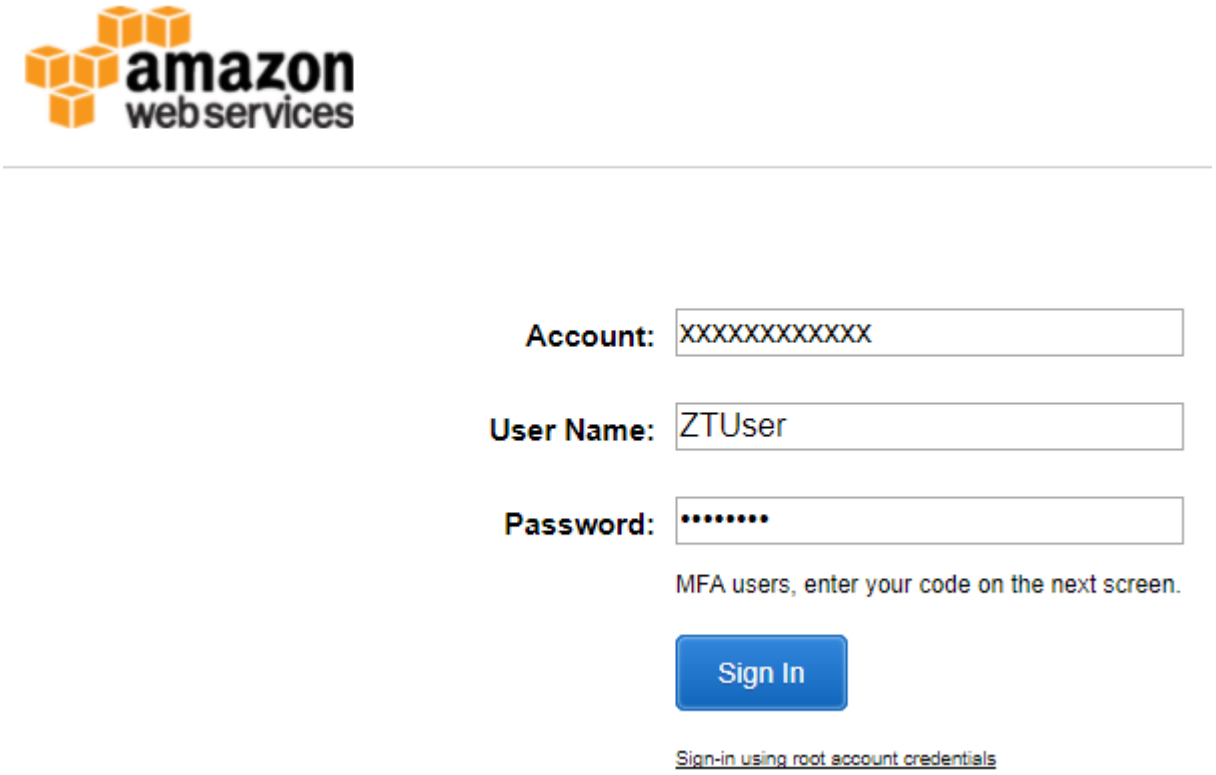
Log into the AWS console

a

Open a web browser and go to the user sign-in URL that you were given when you created ZTUser. The URL will have the following format:

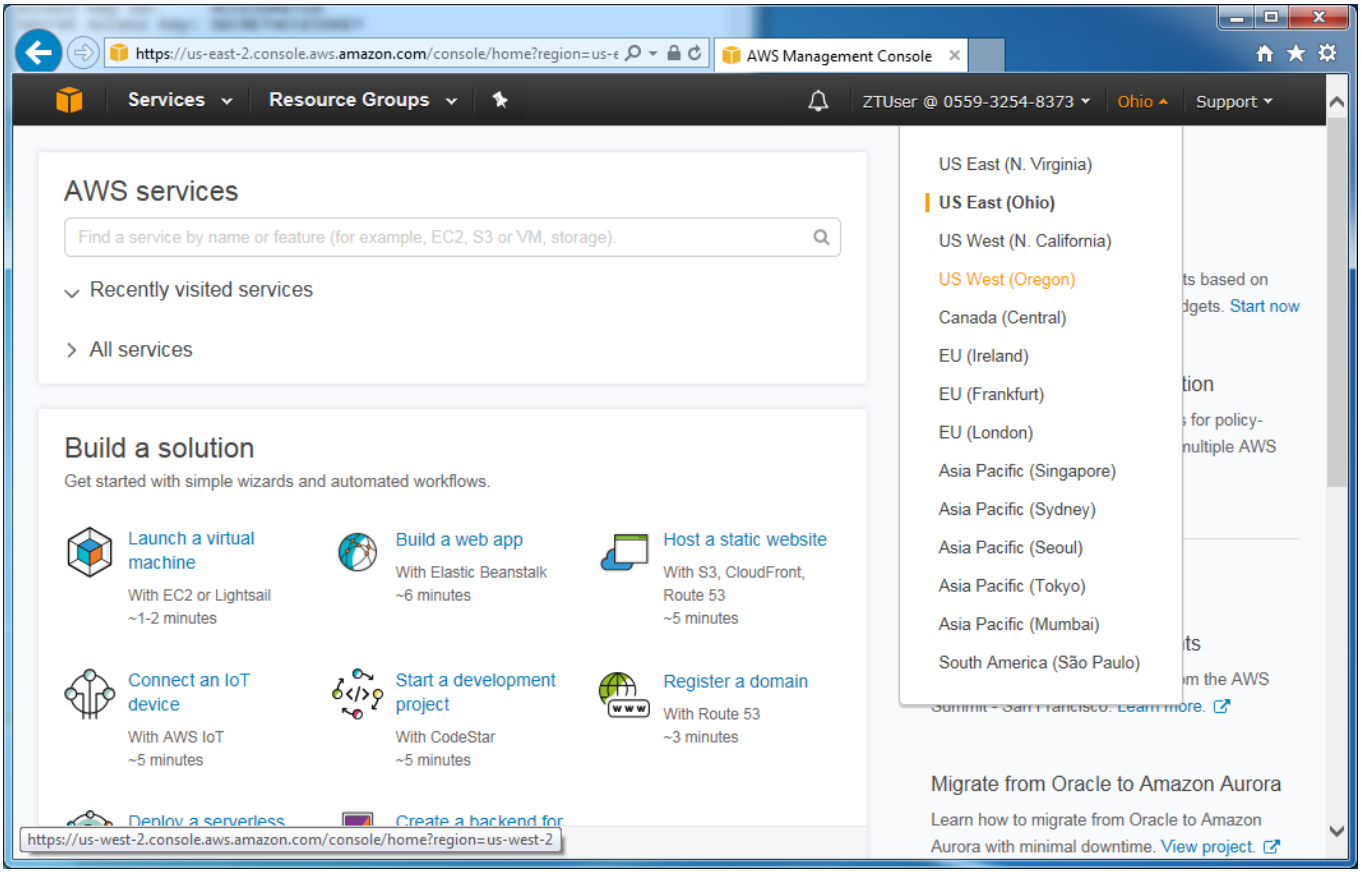
- `https://xxxxxxxxxxxx.signin.aws.amazon.com/console` where `xxxxxxxxxxxx` is the account ID
- Enter the User Name `ZTUser`
- Enter the Password you entered when creating the user account



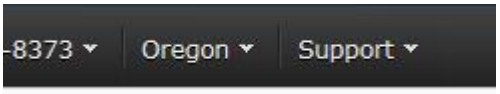


b

Once logged in, change your region to the one closest to you by selecting the region menu (upper-right, left of support menu). We'll use US West (Oregon) in the following steps.



The region menu should now display the region you selected.



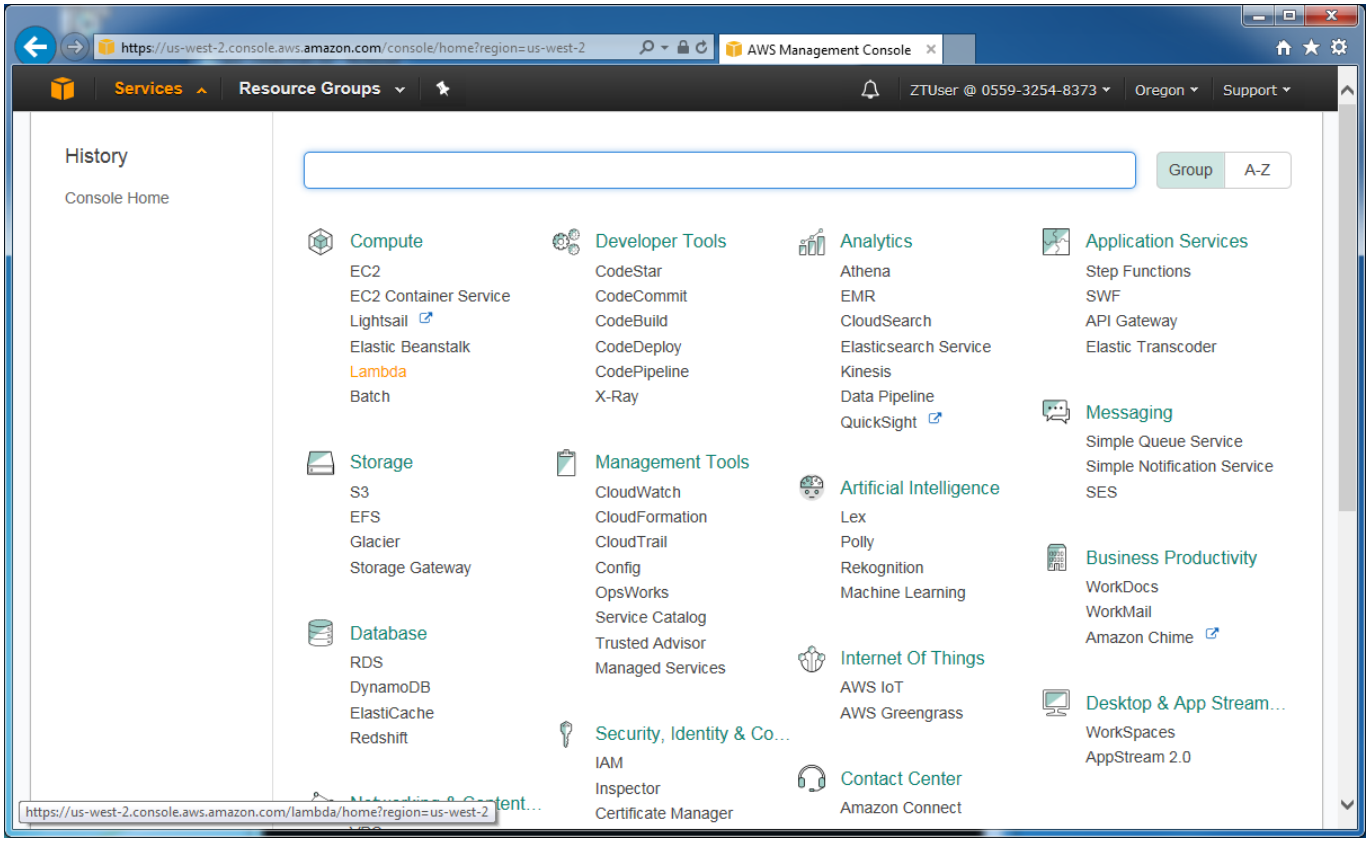
2

Create the *JITR Lambda Function*

The JITR Lambda function is code that is called from AWS Iot when a new device attempts to connect but has not registered yet. It is the function's responsibility to perform the actual registration of the device with AWS IoT.

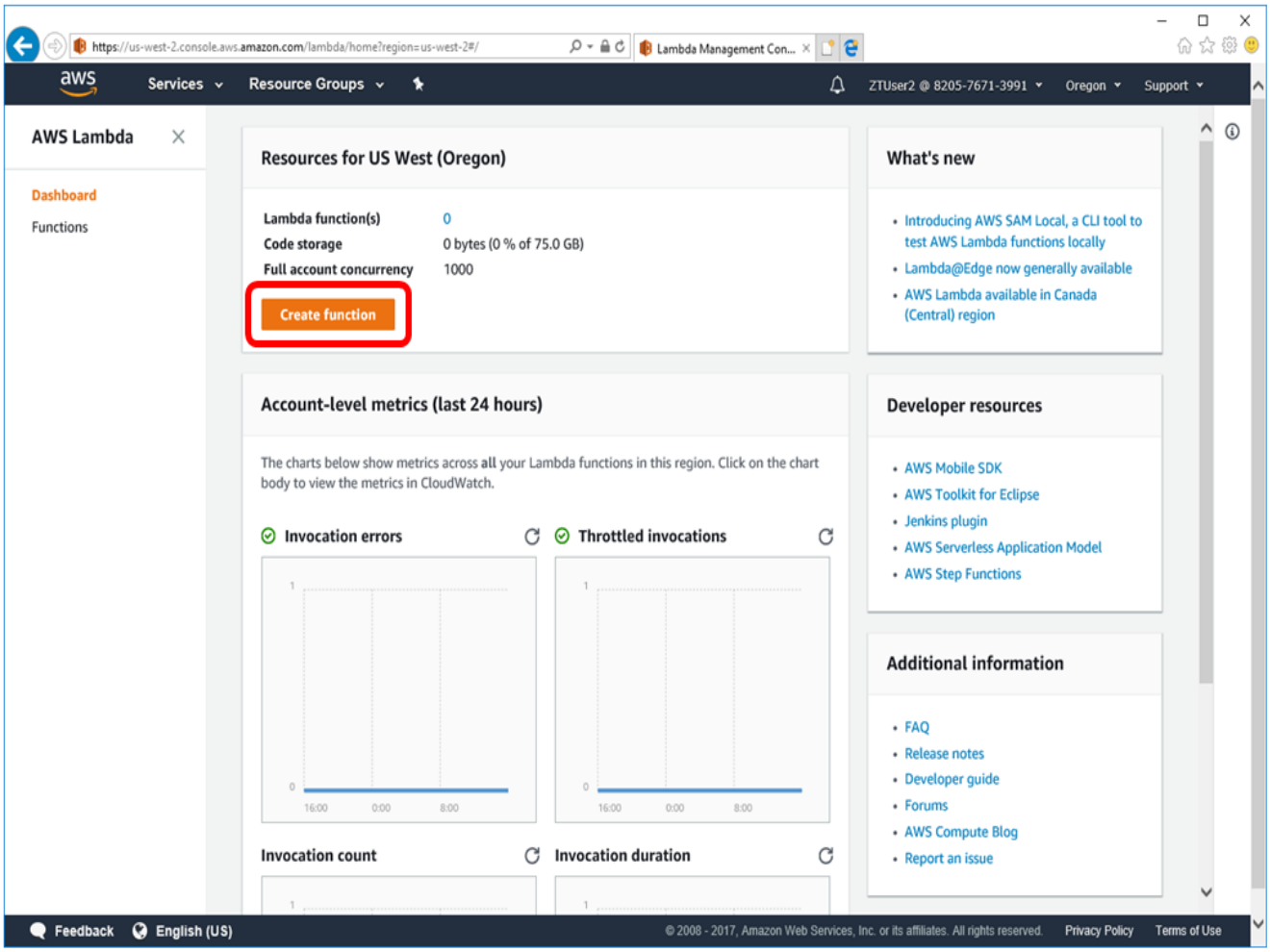
a

Go to the Lambda service under the 'Services' menu and 'Compute' category.



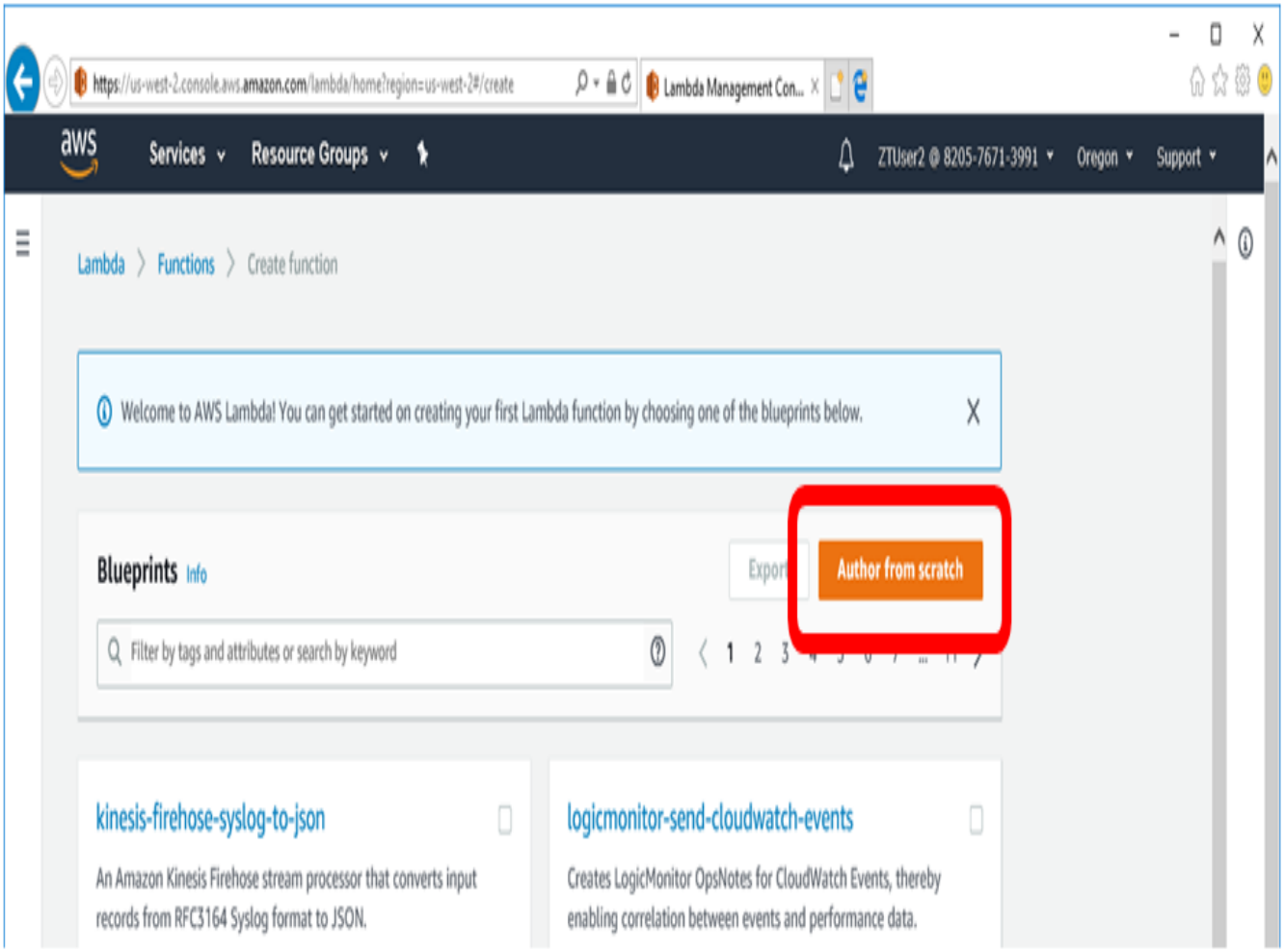
b

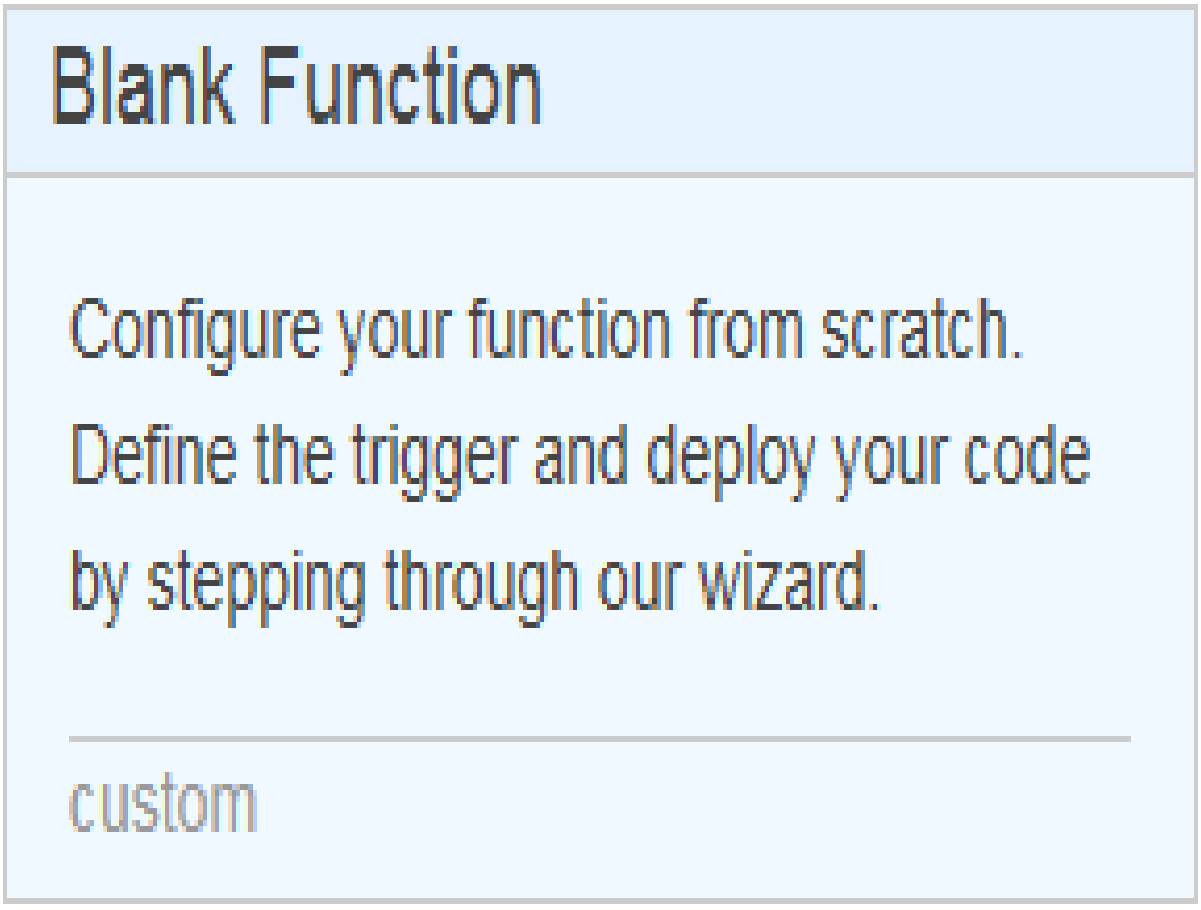
Click on **Create function**.



c

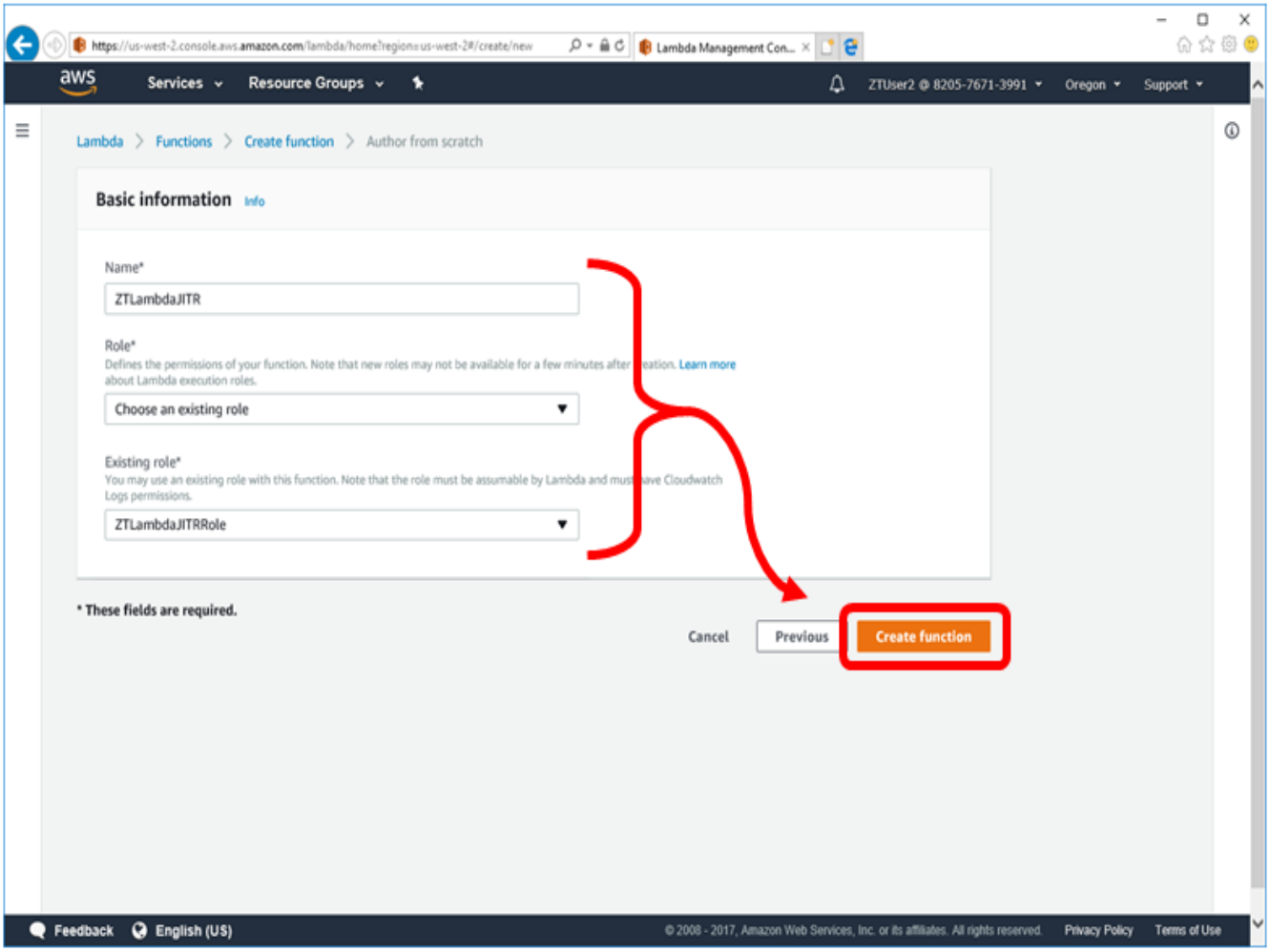
Click on **Author from scratch**.


https://us-west-2.console.aws.amazon.com/lambda/home?region=us-west-2#/create
Lambda Management Con...
aws
Services
Resource Groups
ZTUser2 @ 8205-7671-3991
Oregon
Support
Lambda
Functions
Create function
Welcome to AWS Lambda! You can get started on creating your first Lambda function by choosing one of the blueprints below.
Blueprints Info
Export
Author from scratch
Filter by tags and attributes or search by keyword
1 2 3
kinesis-firehose-syslog-to-json
An Amazon Kinesis Firehose stream processor that converts input records from RFC3164 Syslog format to JSON.
logicmonitor-send-cloudwatch-events
Creates LogicMonitor OpsNotes for CloudWatch Events, thereby enabling correlation between events and performance data.


Blank Function
Configure your function from scratch.
Define the trigger and deploy your code by stepping through our wizard.
custom

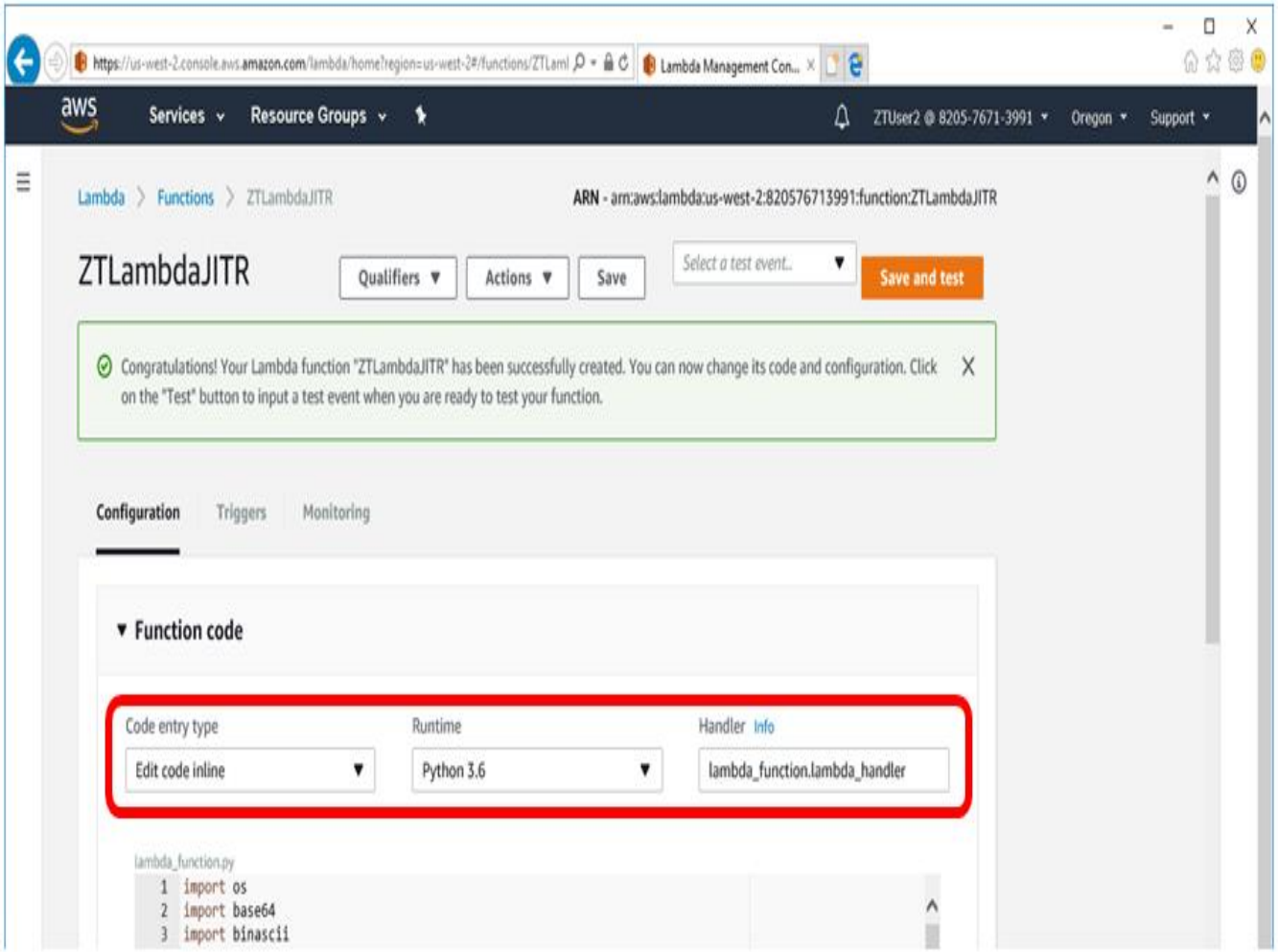
d

Name the new function “ZTLambdaJITR”, select “Choose an existing role” under the 'Role' field, and select the previously created “ZTLambdaJITRRole” under the 'Existing role' field.



e

Next tell AWS Lambda some information about the lambda function you have created. Under Code Entry Type, select 'Edit code inline.' Under the 'Runtime' dropdown box, select 'Python 3.6.' Under the 'Handler' textbox, make sure 'lambda_function.lambda_handler' is entered.



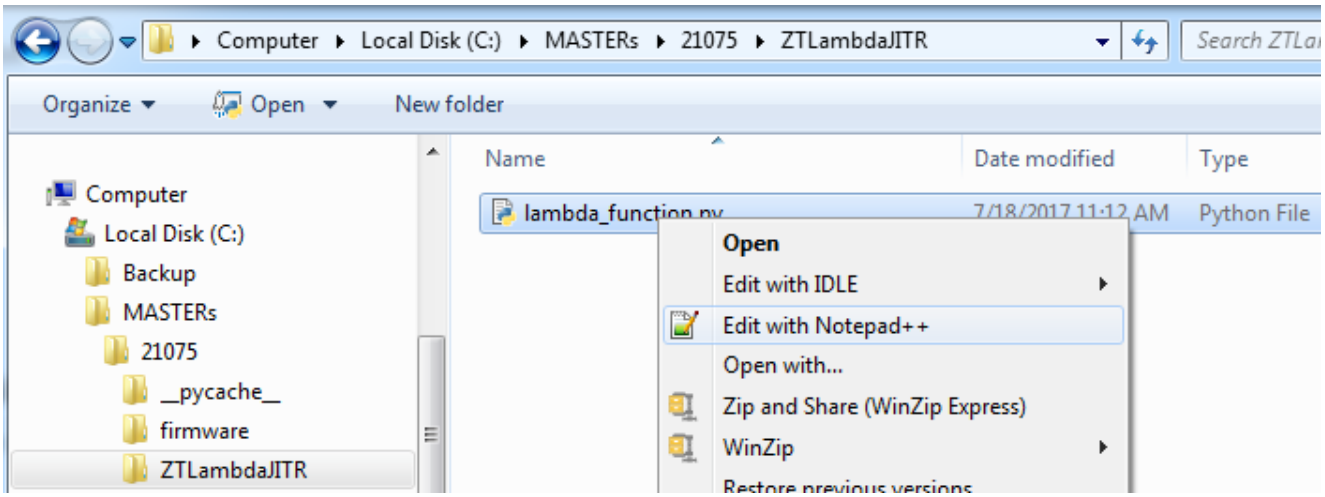
f

Enter the Python code that is to be executed by AWS Lambda when an unregistered device attempts to connect for the first time. Switch to Windows File Explorer and open:

- `aws-iot-zero-touch-kit\ZTLambdaJITR\lambda_function.py`

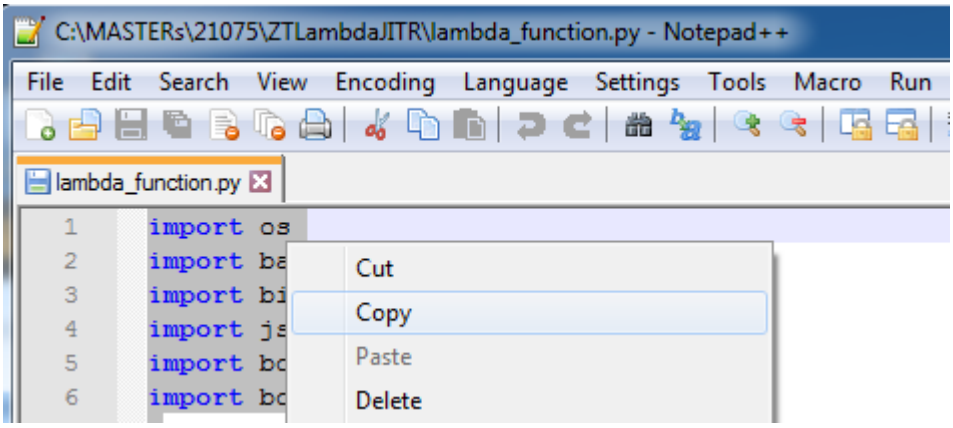
in your favorite text editor.

If you are using Notepad++ editor, you can right-click on the file and select 'Edit'.



g

Select all the code and 'Copy'.

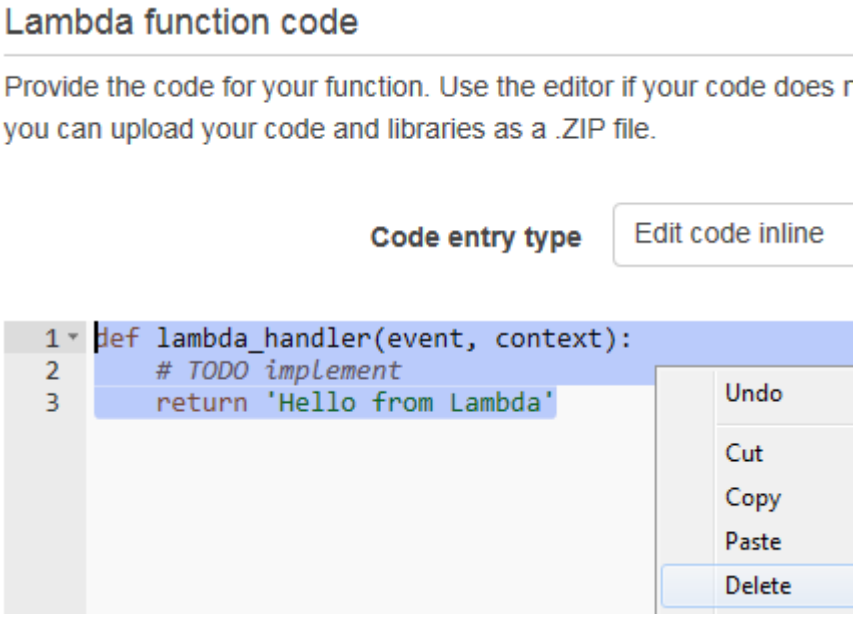


h

Switch back to the AWS console web page. Under Lambda function code, make sure 'Edit code inline' is selected.

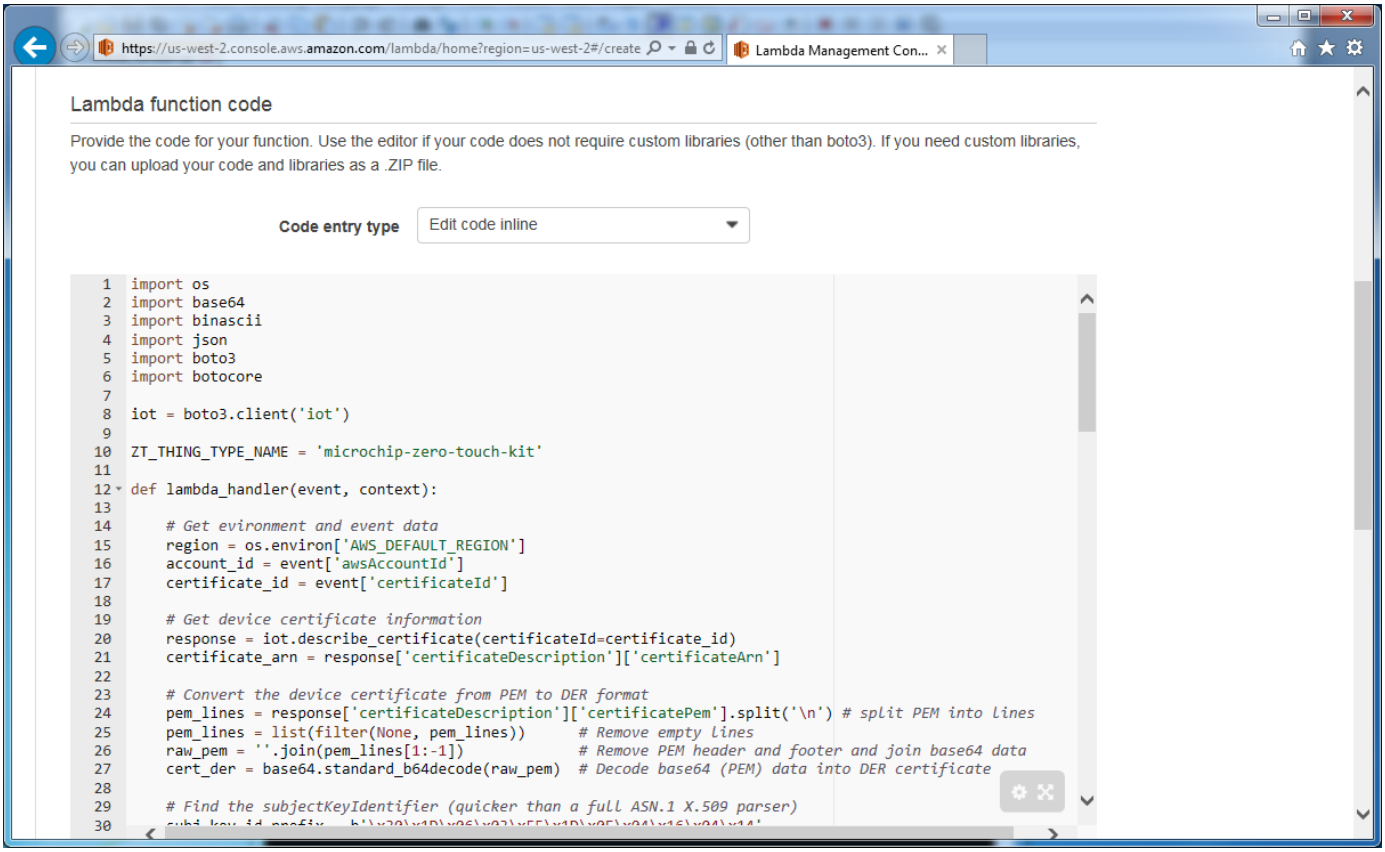
i

Delete the contents of the code entry area by selecting everything and hitting 'Delete'.



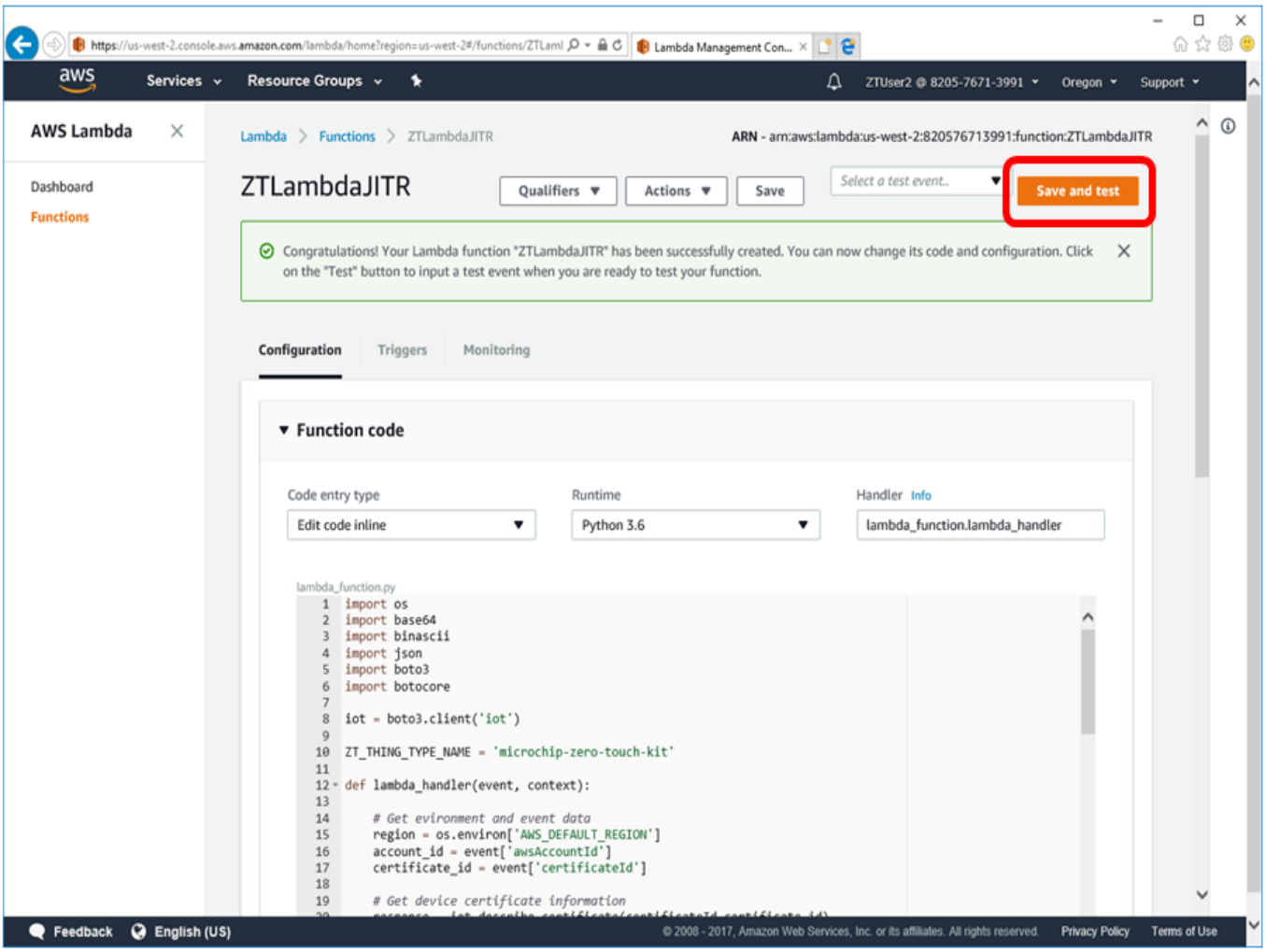
j

Paste the new code from the `aws-iot-zero-touch-kit\ZTLambdaJITR\lambda_function.py` file into the code entry area.



k

Finally, save changes to the lambda function code.



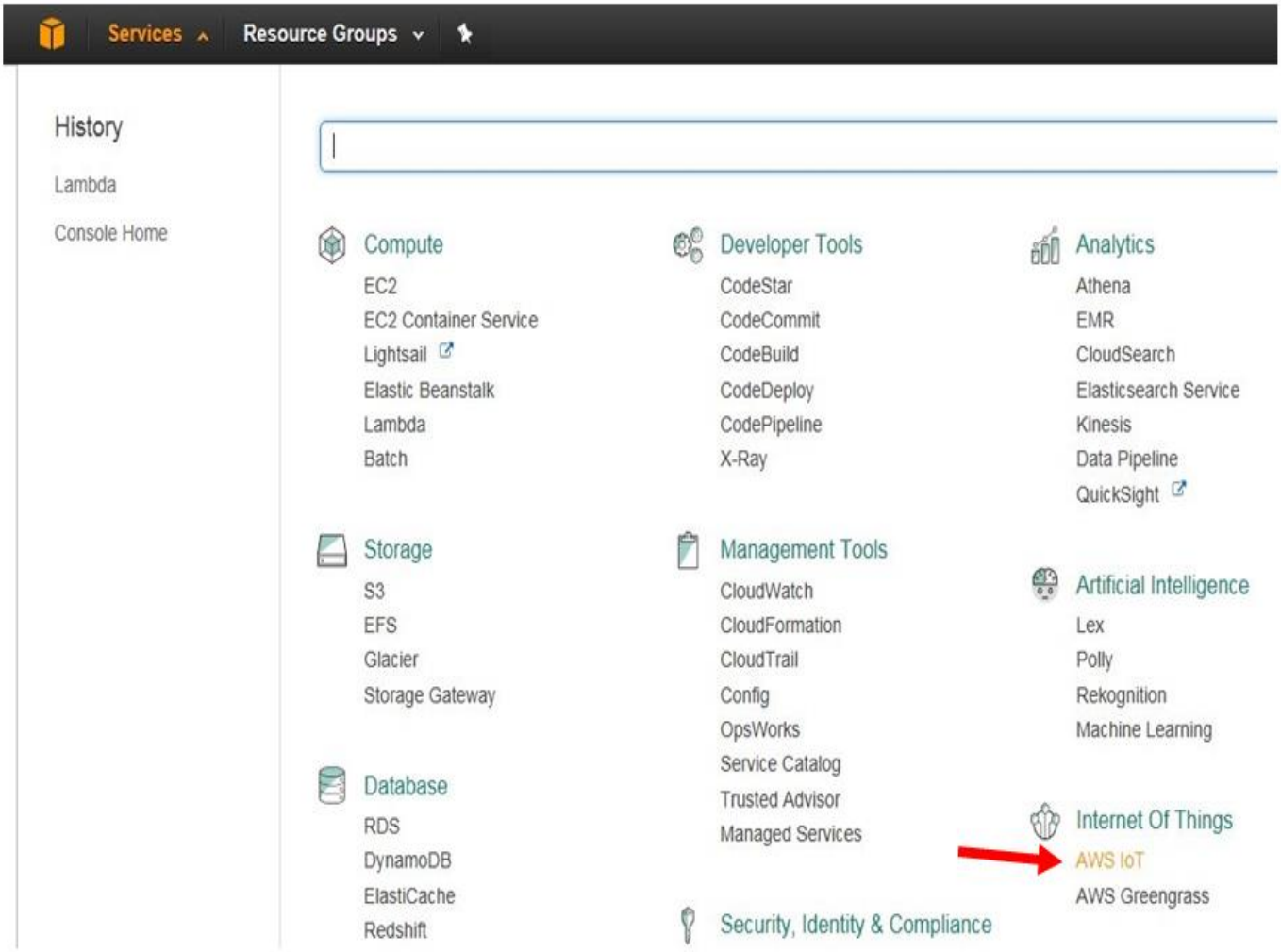
3

Create IoT Rules Engine Rule

While the Lambda function performs the registration it needs to be triggered by an event, the following instructions will create a rule that will run the Lambda function when a device connects for the first time.

a

Go to the AWS IoT service under the 'Services' menu and 'Internet of Things' category.



b

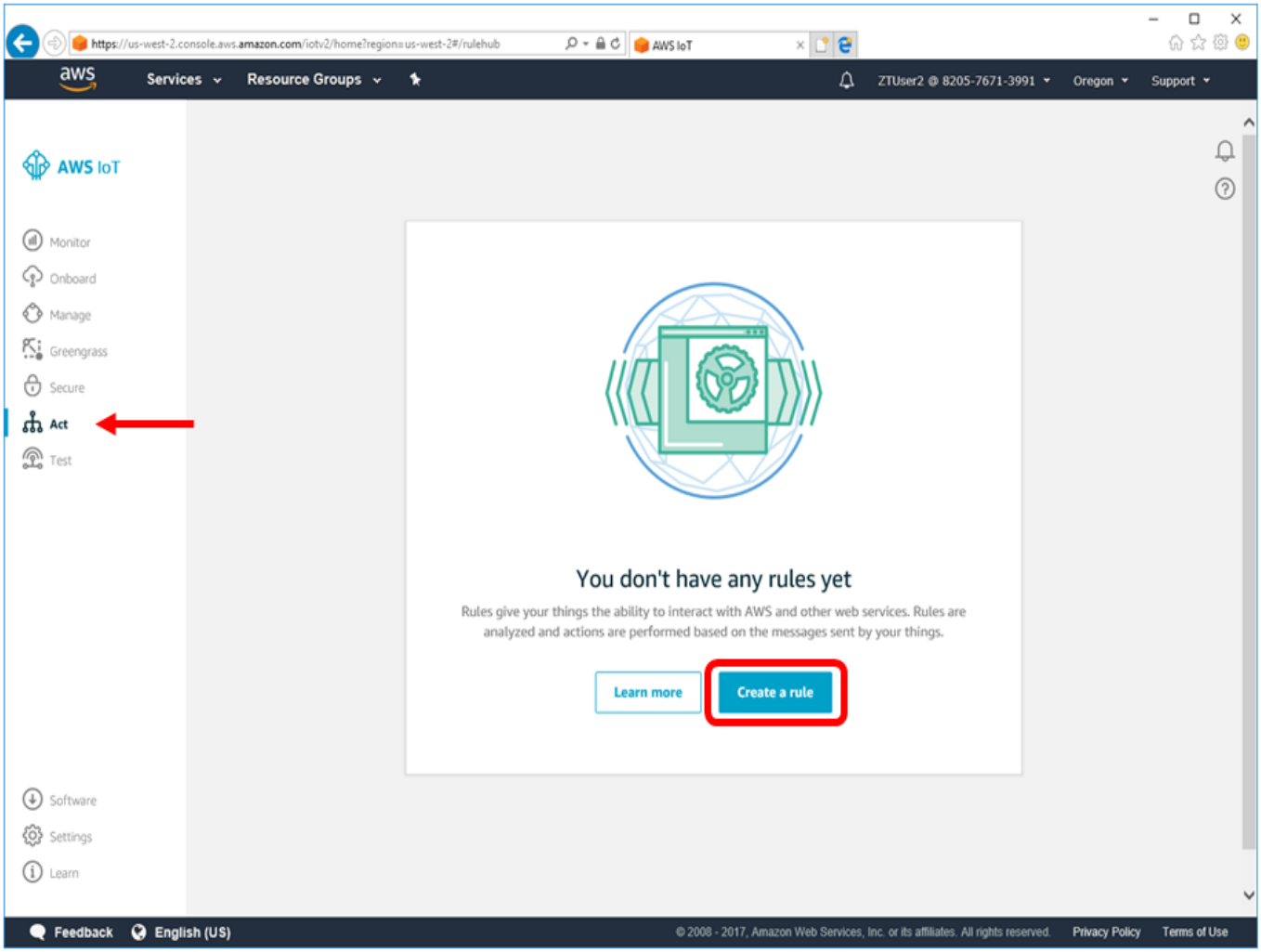
Sometimes the AWS IoT Console will show a getting started window. Click the **Get started** button to dismiss the intro screen.

c

Go to the 'Act' section from the menu at the left.

d

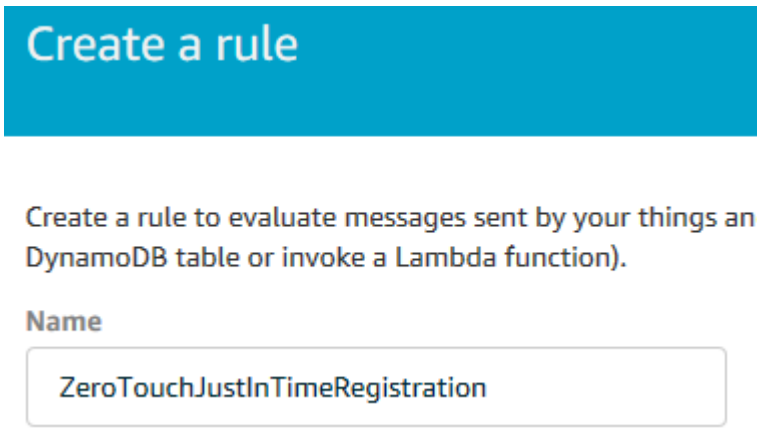
Click the 'Create a rule' button.



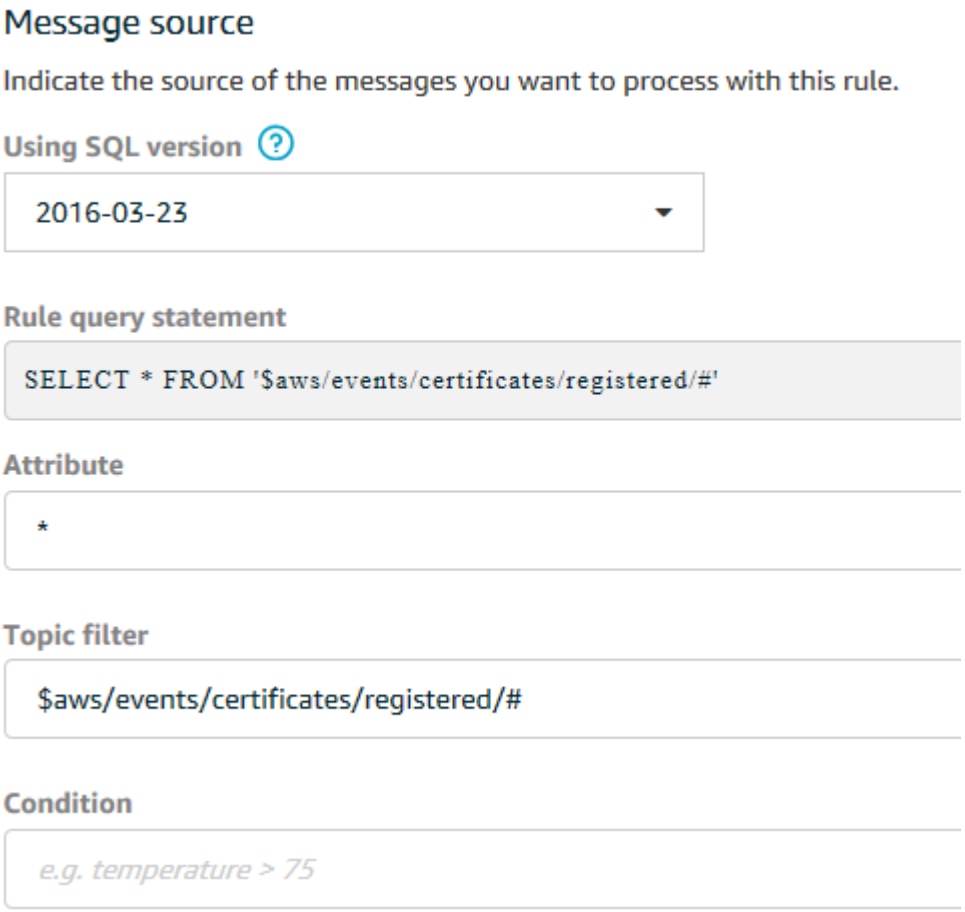
e

Fill in the following fields:

Name: `ZeroTouchJustInTimeRegistration`



SQL version: `2016-03-23`
Attribute: `*`
Topic filter: `$aws/events/certificates/registered/#`
Condition:

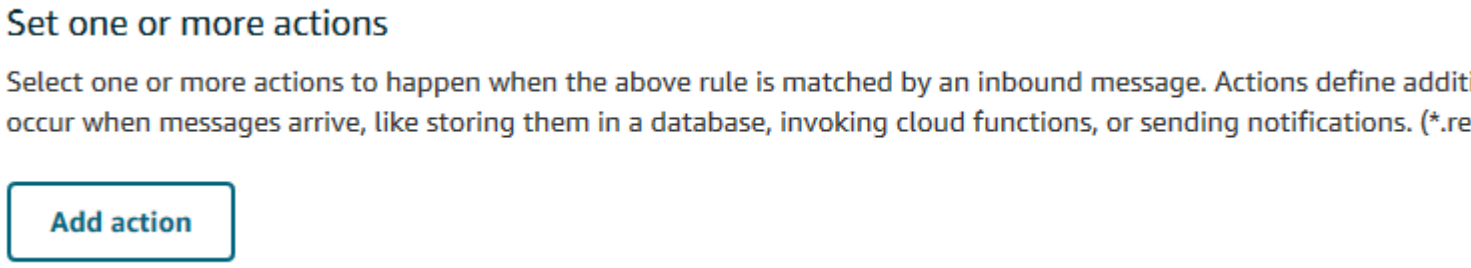


`$aws/events/certificates/registered/#` is a special administrative MQTT topic that AWS IoT will publish to when a device connects with a certificate that hasn't been seen before but has been signed by a CA that was registered in the account.

The `#` at the end indicates we want to trigger this rule for any CA registered with the account.

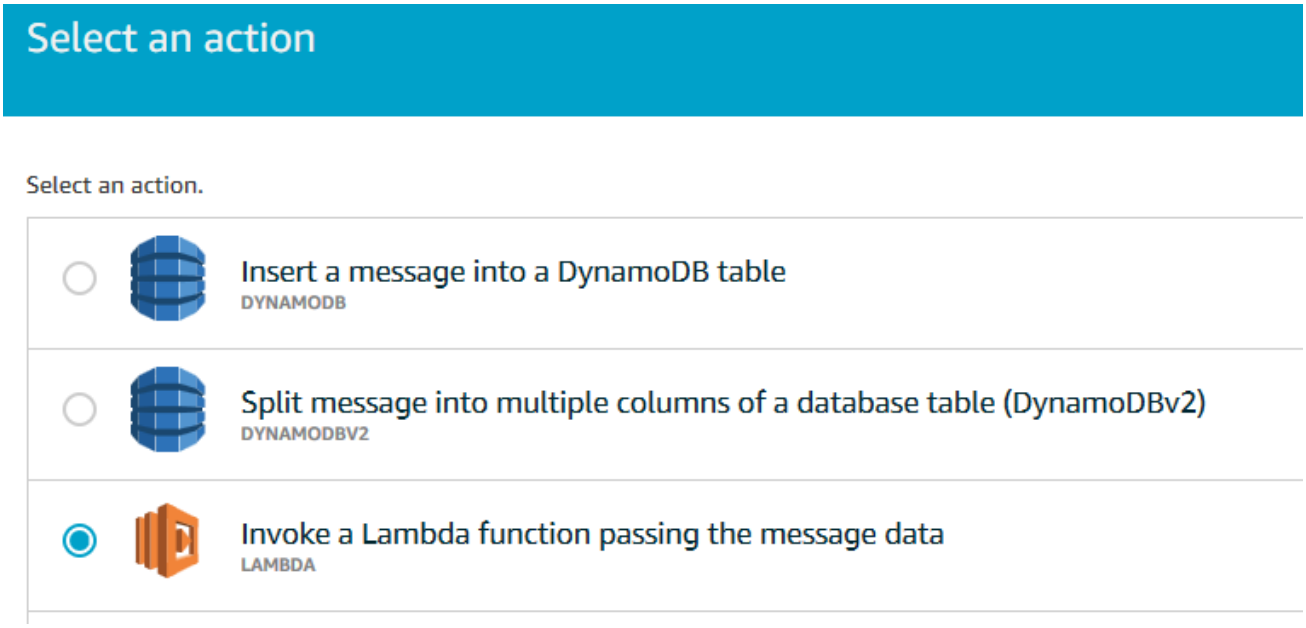
i

Click **Add action**.



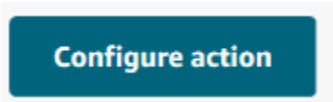
g

Select 'Invoke a Lambda function passing the message data'.



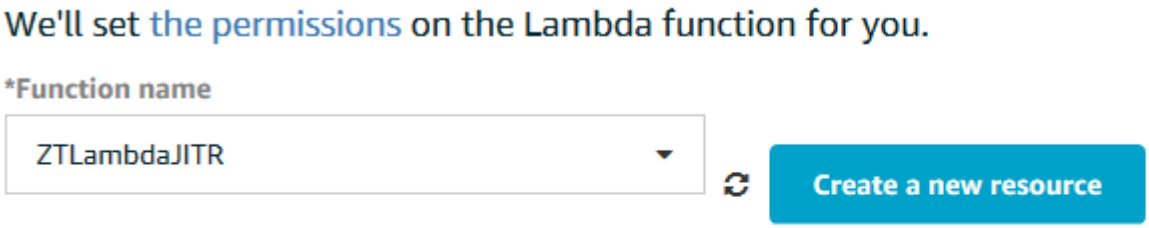
h

Click **Configure action**.



i

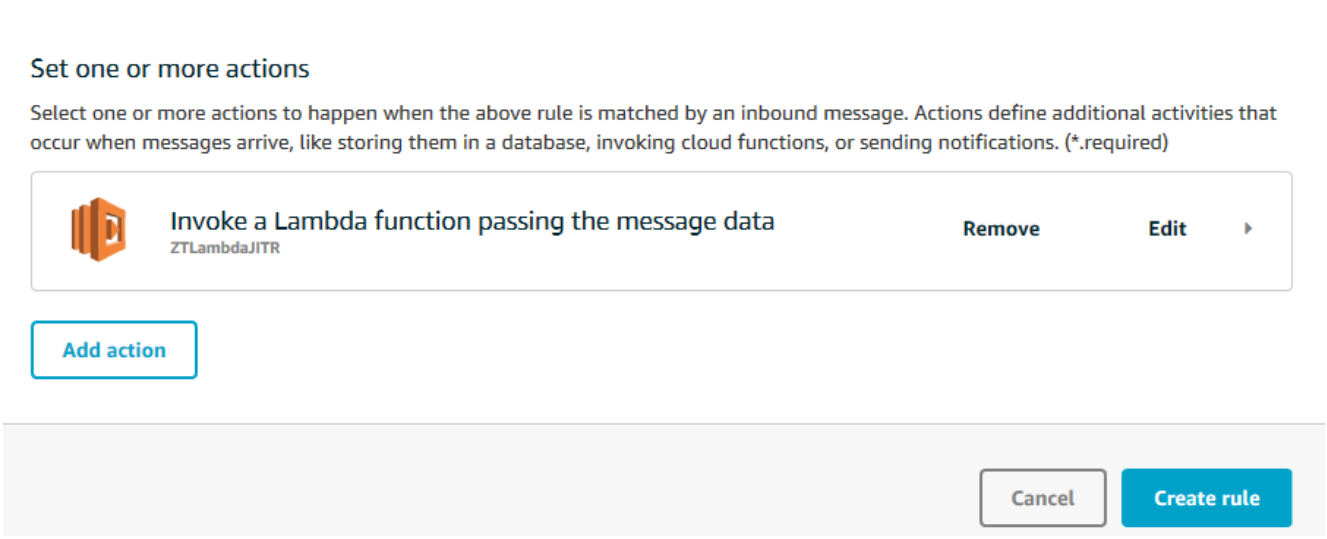
Select the 'ZTLambdaJITR' function and click **Add action**.



Now that this action is configured, this rule will trigger our registration 'Lambda function' when a new device is seen.

j

Finish by clicking 'Add action' and then 'Create rule.'



Let's summarize what you have done so far:

You created:

- a Lambda function to perform JITR
- A trigger for the JITR Lambda function in AWS Iot rules engine

The JITR function is available to any user within the AWS account. Recall that you assigned policy AWSLambdaFullAccess to ZTUser. Therefore, ZTUser has access to the JITR function (resource).

AWS provides many services. Within these services, there are unique-to-the-service actions, things, databases, tables, and much more that can be created by you that are termed *resources*. So far you have created two resources—JITR Lambda function and IoT trigger rule. However, the resources you create are only available in the region that you created them in. For example, the JITR Lambda function that you created in the previous step is only available in the region you selected. Keep this in mind when you create your own IoT ecosystem.

VI. Certificate Authority Setup

In this step, you will create the Certificate Authorities (CA) and register them with AWS IoT so that it can use them to authenticate your IoT devices.

To assist you in the creating the CA, you will use Python scripts. The scripts are broken down into multiple steps to show what is required to set up the CA. While these scripts could be combined into one, we are providing them individually so that you can better understand the creation of the CA's. You can view these Python scripts to see the detailed steps involved.

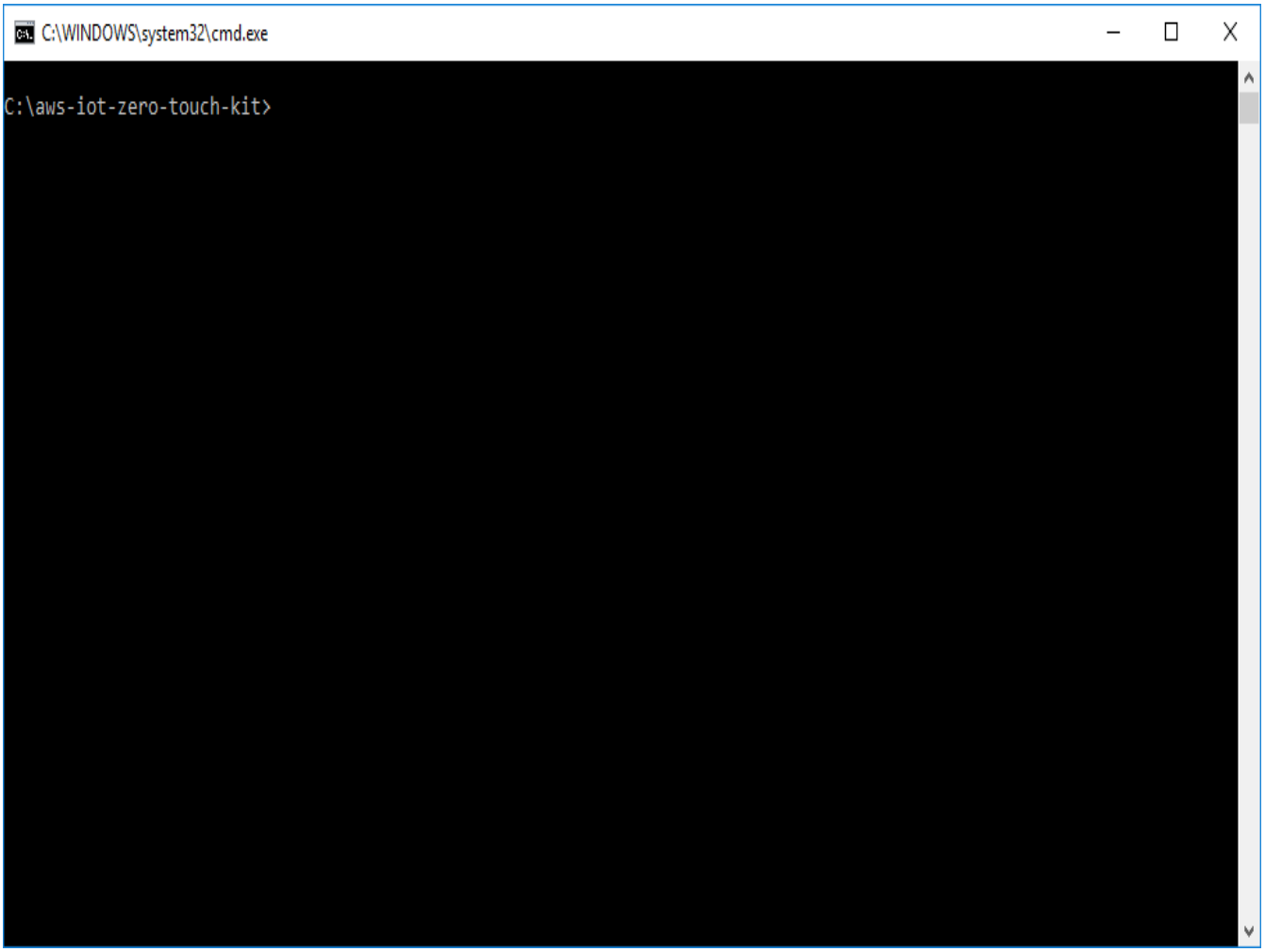
The following steps are for illustration purposes only. Use industry accepted security processes and procedures in the creation and operation of your IoT ecosystem CA. Security of the CA's depends on controlling access to and use of the keys.

1

Open command window and browse to:

- `aws-iot-zero-touch-kit\`

You should get a command prompt that looks like this:



2

Create the **Root Certificate Authority (Root CA)**

The Root CA serves as a single authority over an IoT ecosystem.

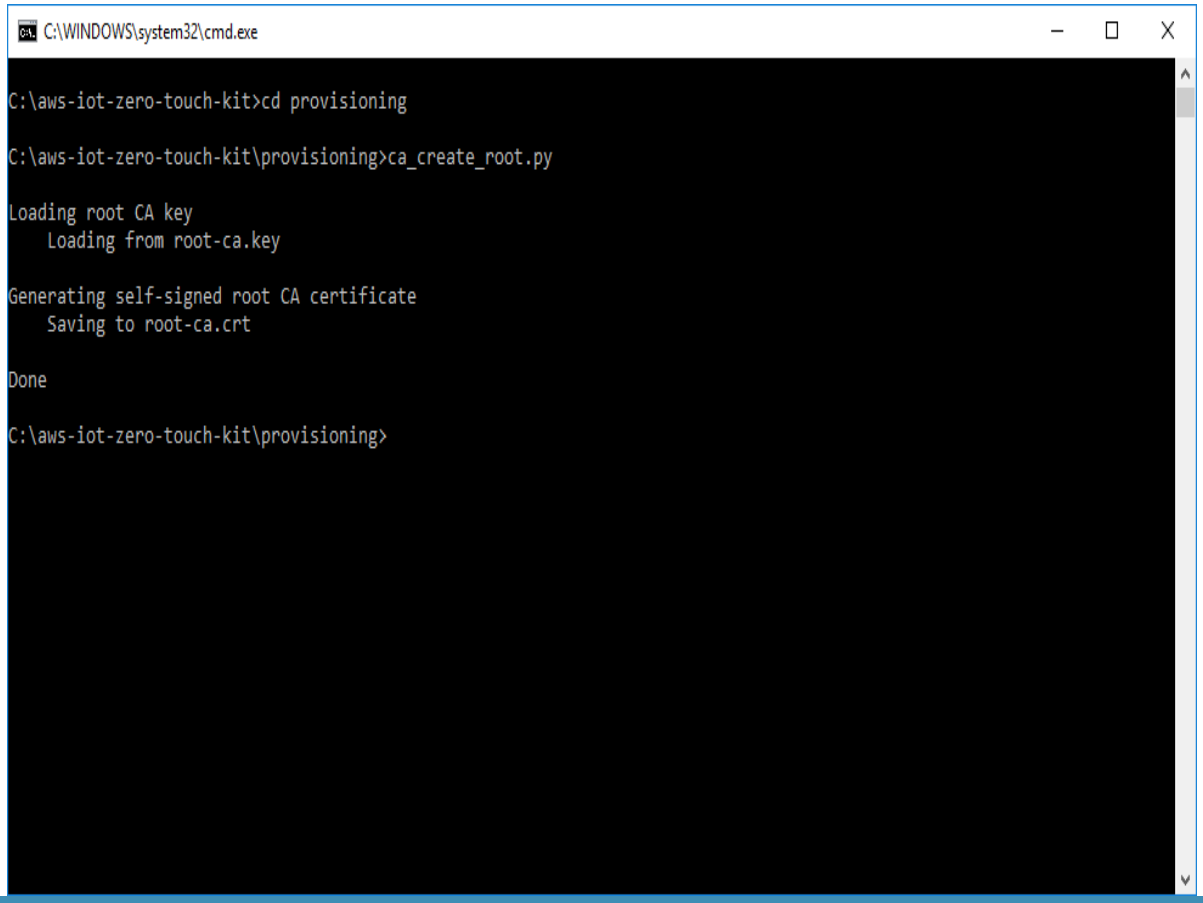
Change directory to the 'provisioning' sub-directory: {{cd provisioning]]

Run the `ca_create_root.py` Python script

This script will create:

- root key (stored in the `root-ca.key` file), and
- root certificate (stored in the `root-ca.crt` file)

Because this is the Root CA, its certificate is signed by its own key.



The file formats of the `root-ca.key` and `root-ca.crt` files are standard PEM encoding used by openSSL and other *Public Key Infrastructure (PKI)* software.

If the `root-ca.key` file already exists, the Python script will use that existing key and generate a new certificate.

3

Create the **Signer Certificate Authority (Signer CA)**

The Signer CA is used during manufacturing and is responsible for directly signing the device certificates. This process is known as "provisioning".

a

Signer creation is split into two (2) steps, the first is generating its key and a **Certificate Signing Request (CSR)**.

Run the `ca_create_signer_csr.py` python script.

This script will create the signer key, `signer-ca.key` and its CSR, `signer-ca.csr`.

If the `signer-ca.key` file already exists, the Python script will use that existing key and generate a new CSR.

b

The Root CA is now used with the Signer CSR created above to complete creation of the Signer CA. While this could technically be done in a single Python script, there are two Python scripts to represent the split in responsibilities between the authority (Root CA) and subject (Signer CA) in PKI systems.

Run the `ca_create_signer.py` python script.

This script will create the signer certificate, `signer-ca.crt`.

4

Register the Signer CA with AWS IoT

The final step in setting up the certificate chain is to register the Signer CA with AWS IoT.

Using the JITR process, we need to register the Signer CA for the devices. This relieves us from registering individual device certificates with AWS IoT at manufacturing time. When an individual device connects with AWS IoT for the first time, AWS IoT does not recognize the individual device but will recognize its Signer CA.

As a security feature, AWS IoT requires that you prove you have access to the CA private key before registering that CA. This involves the following steps:

- Request a registration code from AWS IoT
- Create a verification certificate around that registration code
- Sign the verification certificate with the Signer CA
- Supply both the Signer CA certificate and verification certificate when registering

Run the `aws_register_signer.py` python script.

This script will perform the above steps and save the verification certificate to `signer-ca-verification.crt`. This file is not required by any other step but is saved for reference.

Let's Summarize What You've Done So Far:

- Created two CAs: Root and Signer
- Registered the Signer CA with AWS IoT

VII. Provision the Device

In this step, you will provision the Zero Touch Secure Provision Kit with the credentials required to connect and communicate with your AWS account.

The SAM G55 Xplained Pro comes programmed with the AWS IoT Zero Touch firmware project. To update to the latest firmware or program another SAM G55, follow these steps:

1. Open Atmel Studio 7 and open the zero touch firmware solution: `AWS_IoT_Zero_Touch_SAMG55.atsln`
2. Plug the SAM G55 Xplained Pro into the computer via the EDBG USB Port
3. Within Atmel Studio, using the **Debug > Start Without Debugging** menu option to rebuild and load the firmware onto the board

Ensure that the latest firmware is installed on the ATWINC1500. Instructions on how to upgrade the firmware are located on the ATWINC1500-XPRO product web page. Scroll to the bottom of the page and select 'Platform Getting Started Guide (Flash Memory Download Procedure)'.

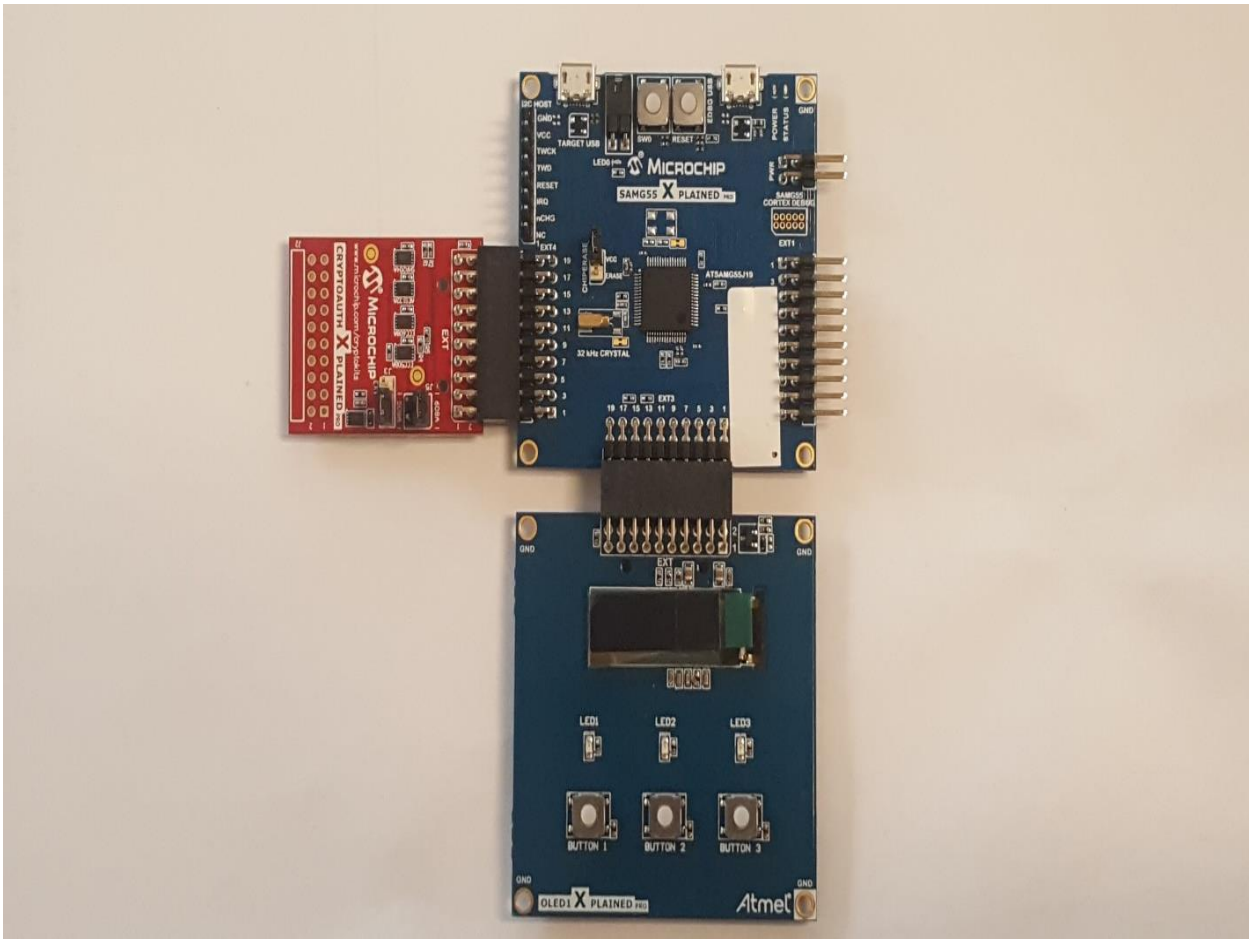
The latest firmware version for the ATWINC1500 is 19.5.4 (as of October 2017).

1

Assemble and plug in the kit

The SAM G55 Xplained Pro forms the central hub, while the other boards plug into the following connectors:

EXT3: OLED1 Xplained Pro
EXT4: CryptoAuth Xplained Pro



2

Plug in the board to the PC from the TARGET USB port on the SAM G55 board

3

Connect a second USB cable, connect the EDBG USB port to the PC as well

Debugging information is exposed via a com port available through the EDBG connection.

To see the debugging information we will need to connect to the COM port using a terminal program.

a

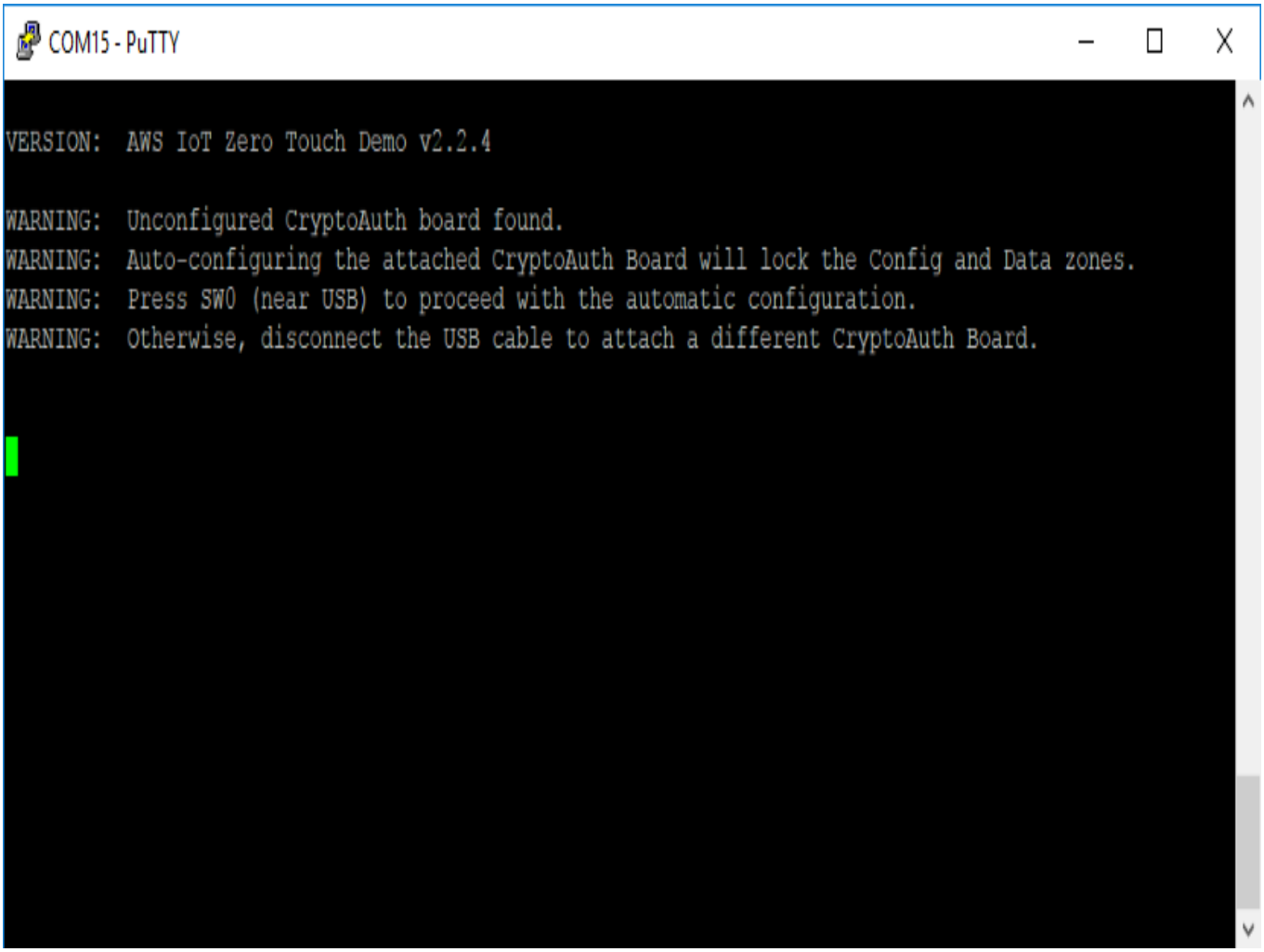
If using PuTTY:

To find the right com port number, open device manager, expand ports and look for the port labeled EDBG Virtual Comport (COMx), where x is the number you're looking for.

Next, to see the board status, open PuTTY and enter the following:

Connection type: Serial
Serial line: `COMx` – where x is the number from the previous step
Speed: `115200`

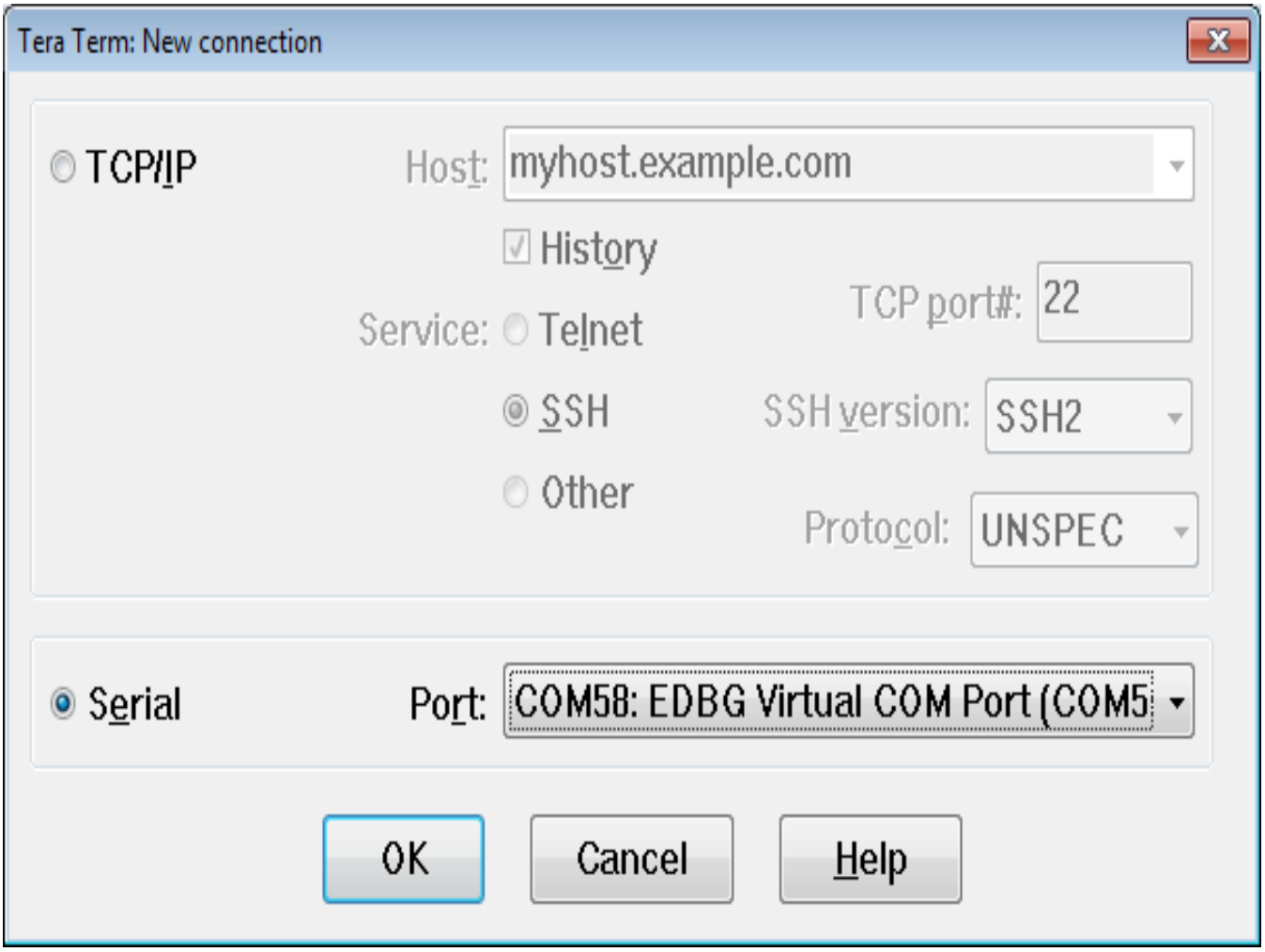
Click 'Open' and you should see a window with status messages. If nothing appears, try pressing the RESET button on the SAMG55 board.



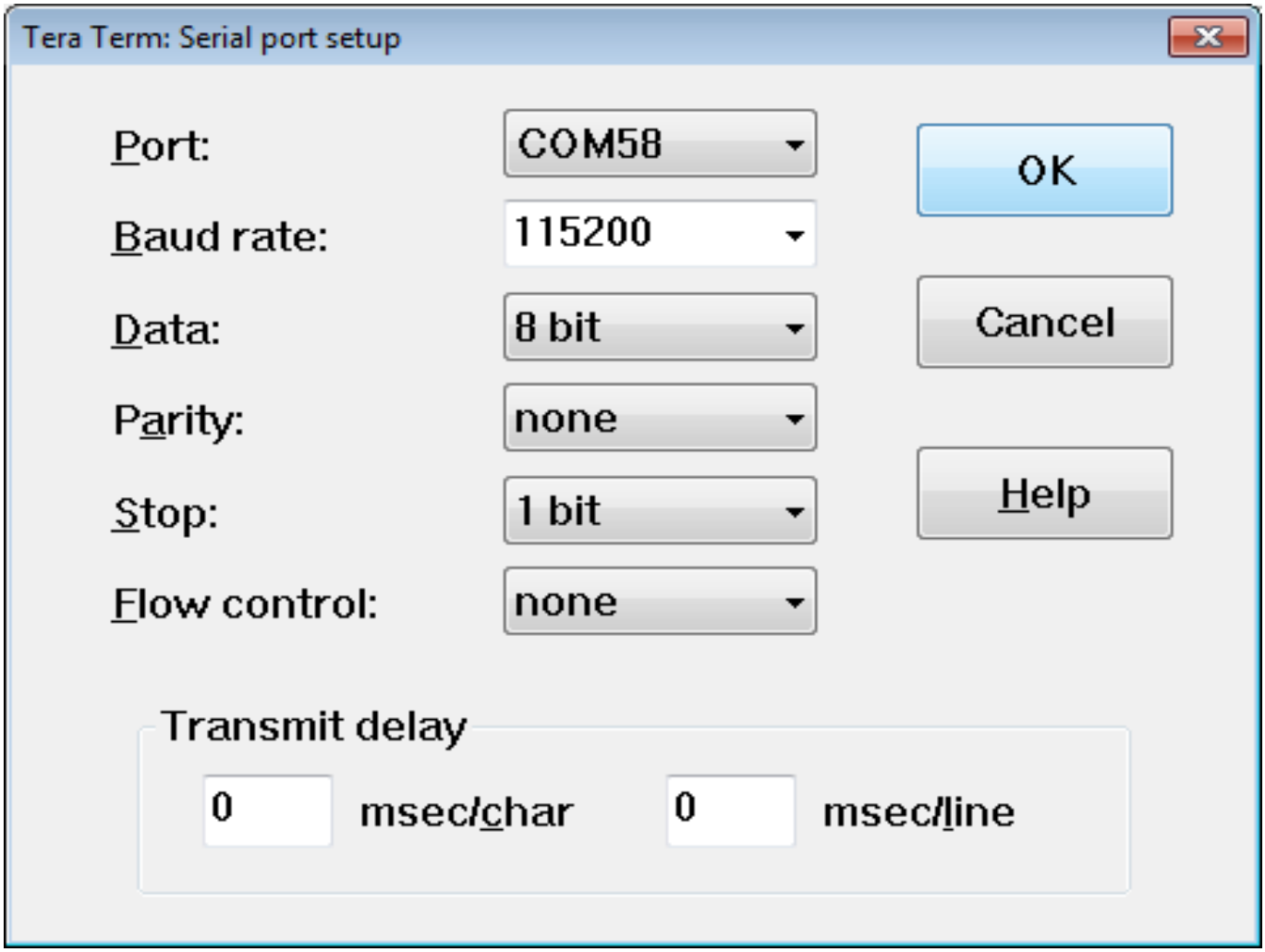
b

If using Tera Term:

Open Tera Term, select 'Serial,' select the EDBG Virtual COM Port (actual COM number may be different), and click **OK**.



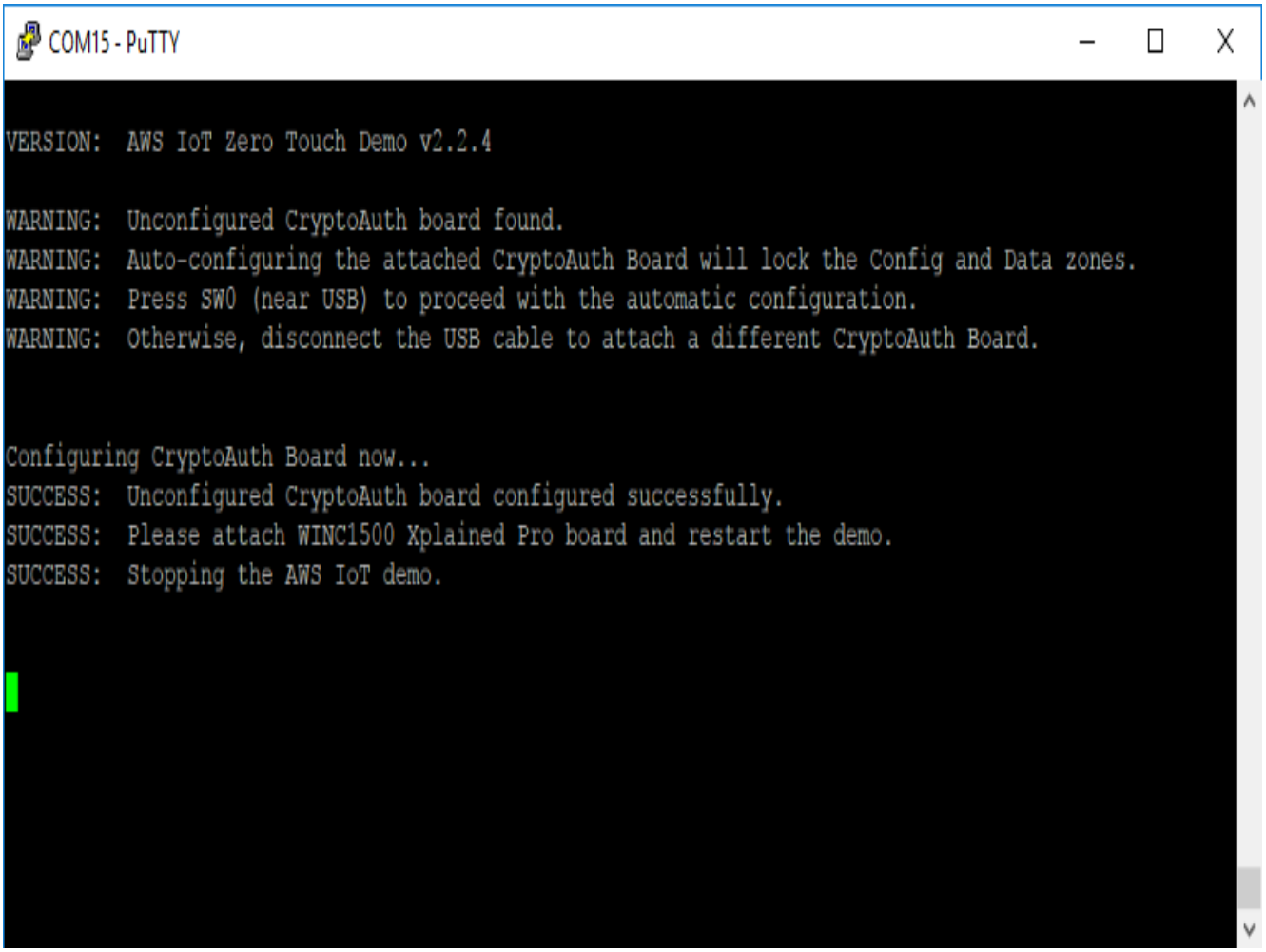
Go to the 'Setup' menu and select 'Serial'. Change the Baud rate to 115200, click **OK**.



You should see a window with status messages. If nothing appears, try pressing the RESET button on the SAMG55 board.

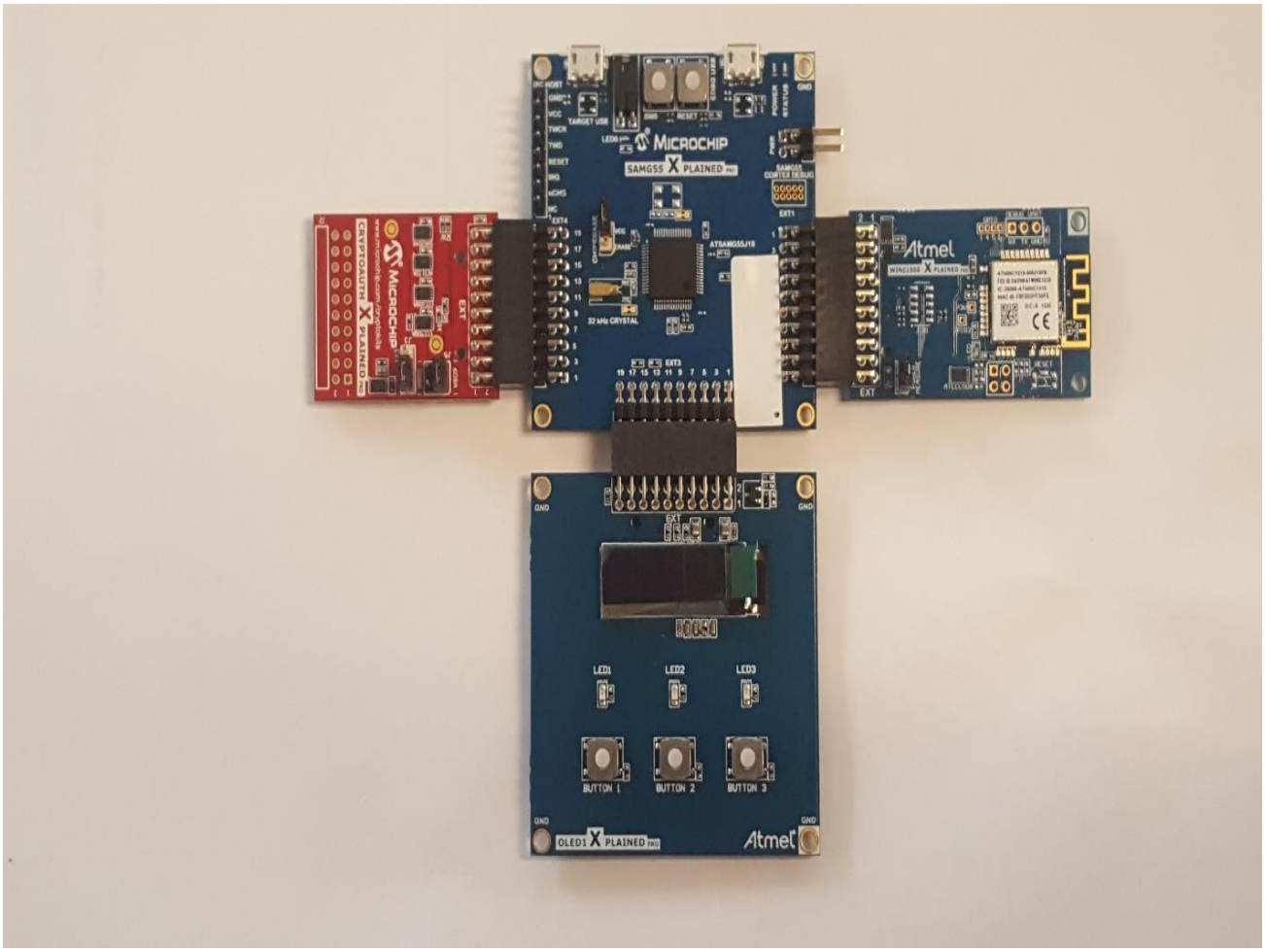
4

The terminal window will show the status of the pre-configuration process. An unconfigured board should be detected and appropriate messages shown. This message will repeat every ~2.5 seconds until SW0 is press or power is removed. Press the SW0 button at the top of the SAMG55 Xplained Pro board to proceed with the automatic configuration of the CryptoAuth board.



5

Once the CryptoAuth board has been automatically configured, attach the ATWINC1500 Xplained Pro board to the EXT1 port on the SAMG55 Xplained Pro board. Reset the SAMG55 to restart the demo with the newly connected ATWINC1500 board.



6

If you haven’t already connected USB cables from your PC to the SAMG55 Xplained Pro board, do that now.

- Plug in the board to the PC from the TARGET USB port on the SAM G55 board.
- Connect a second USB cable, connect the EDBG USB port to the PC as well.

Debugging information is exposed via a com port available through the EDBG connection. To see the debugging information you will need to connect to the COM port using a terminal program.

a

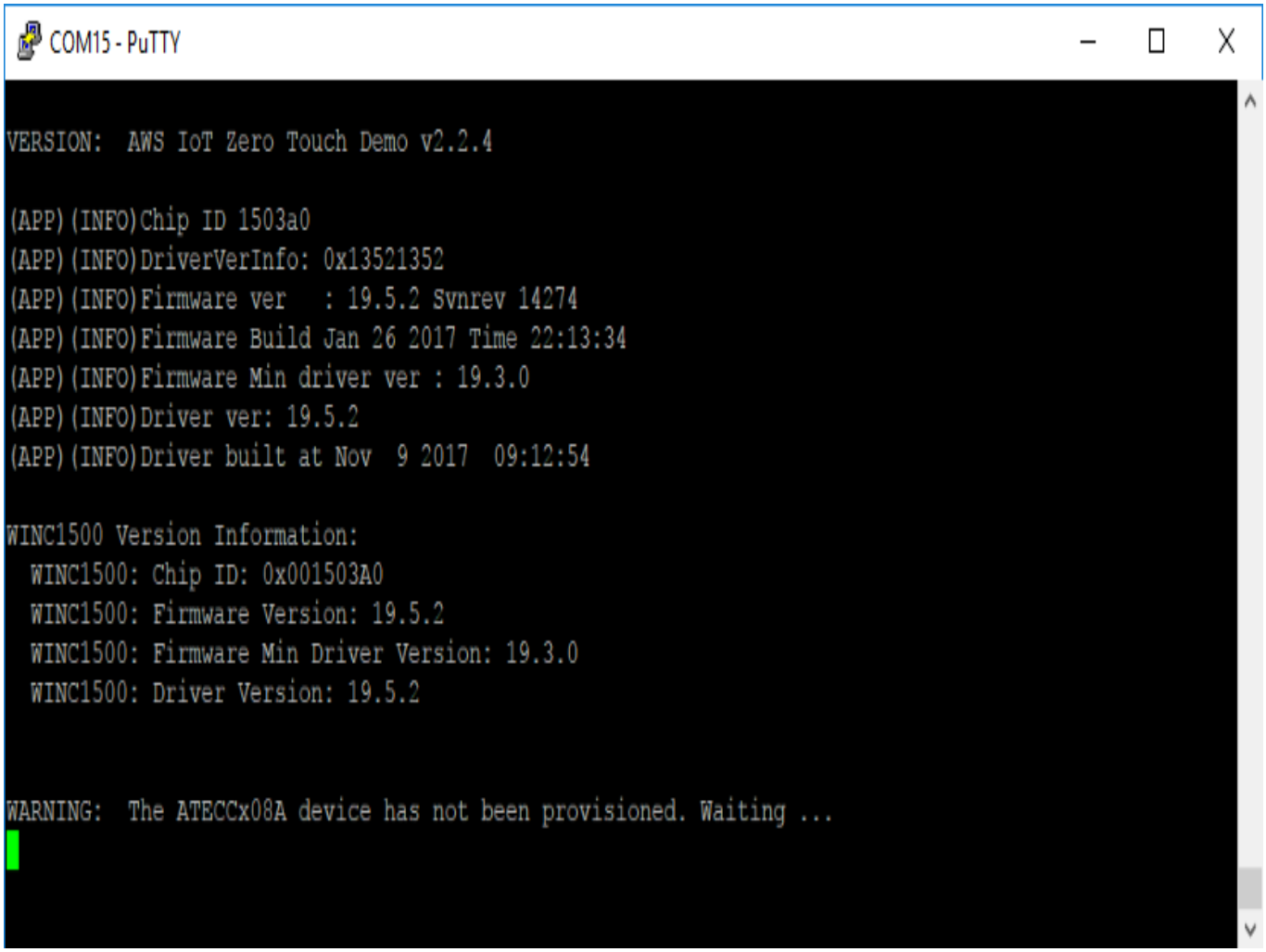
If using PuTTY:

To find the com port number associated with the EDBG port, open device manager, expand ports and look for the port labeled EDBG Virtual Comport (COMx), where x is the number you're looking for.

Next, to see the board status, open PuTTY and enter the following:

- Connection type: Serial
- Serial line: COMx – where x is the number from the previous step Speed: 115200

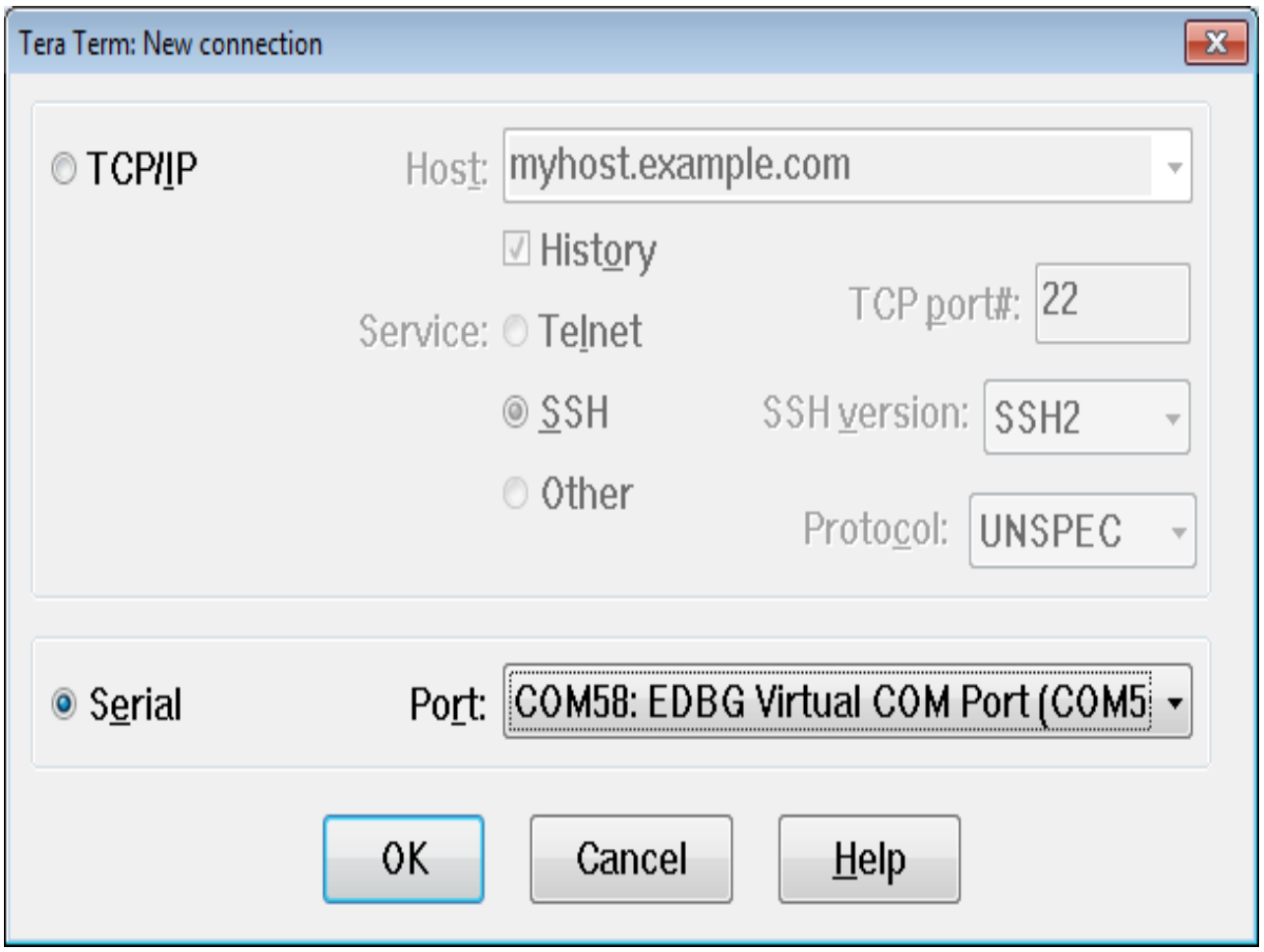
Click 'Open' and you should see a window with status messages. If nothing appears, try pressing the RESET button on the SAMG55 board.



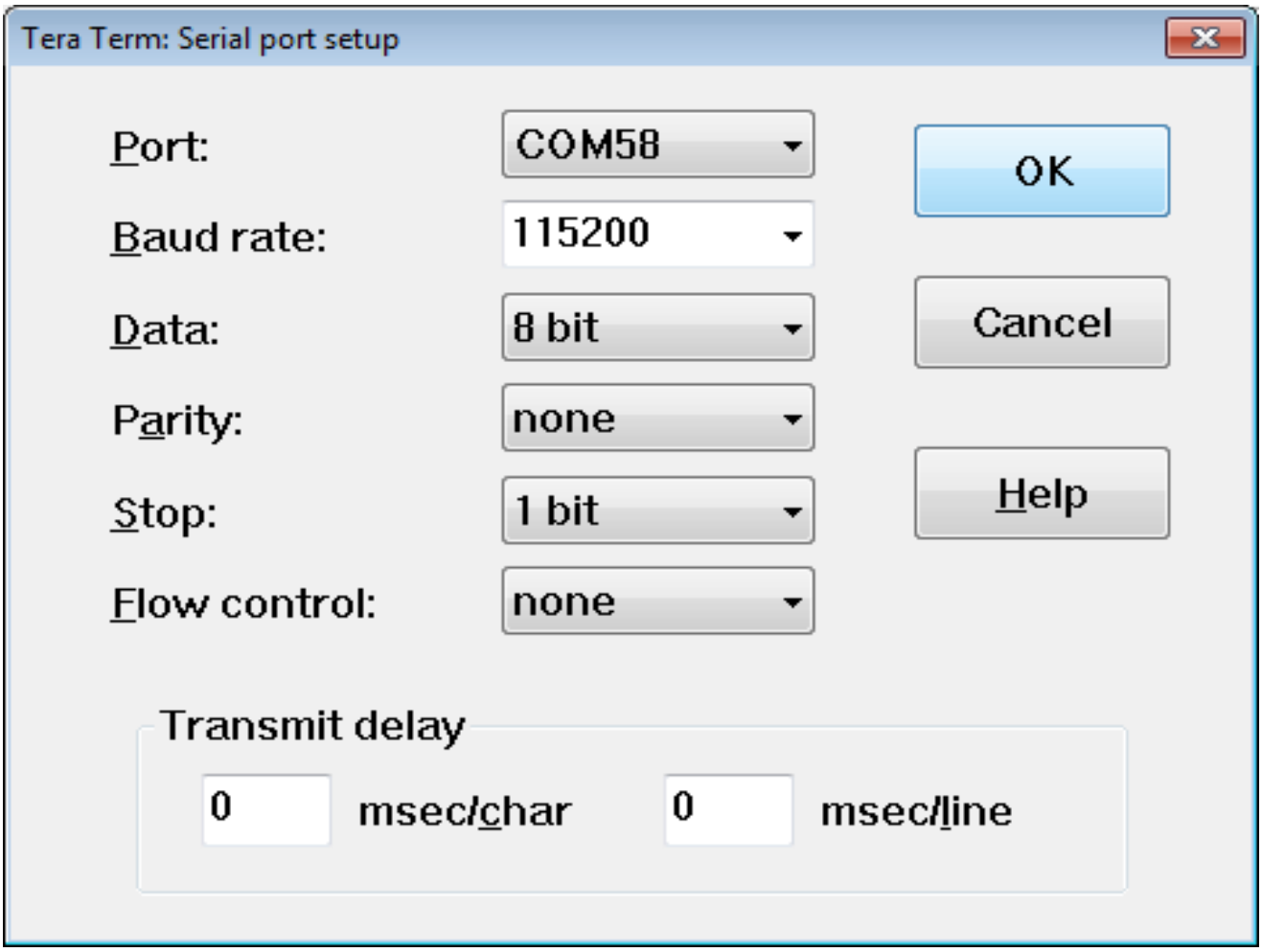
b

If using Tera Term:

Open Tera Term, select 'Serial', select the EDBG Virtual COM Port (actual COM number may be different), and click OK:



Go to the 'Setup' menu and select 'Serial'. Change the Baud rate to 115200, click OK:



You should see a window with status messages. If nothing appears, try pressing the RESET button on the SAMG55 board.

7

Set **Wi-Fi™** credentials

For the kit to connect to a Wi-Fi access point you need the following:

- Access Point operating in WPA2 personal mode
- SSID
- Password
- Internet ports 123 and 8883 open

You will not be able to connect to an access point that has open access or enterprise security.

Run the `kit_set_wifi.py –ssid wifi-name –password wifi-password` python script.

Where wifi-name = SSID and wifi-password = PASSWORD of your Wi-Fi access point.

8

Provision the device

Run the `kit_provision.py` python script. The script will:

- Request a Certificate Signing Request (CSR) from the device.

The CSR will use the key pair stored in slot 0 of the ATECC508A. The ATECC508A is a secure container for the private key. The key internally generated with its secure RNG and the ATECC508A provides no mechanism for reading out a private key.

This key provides a secure identity for the IoT device that can't be copied, either intentionally, by an attacker or through a software bug.

- Create a device certificate using the CSR and signer CA.
- Send the device certificate, signer certificate and AWS connection information to the board.

These certificates and the AWS connection information is all stored on the ATECC508A:

Slot 8 – AWS Connection Information (including wifi credentials)
Slot 10 – Device compressed certificate
Slot 11 – Signer public key
Slot 12 – Signer compressed certificate
Slot 14 – Signer certificate serial number and full validity dates

Once the board has been successfully provisioned, LED0 on the SAM G55 Xplained Pro board should blink five times. Additionally, if you are watching the debug output from the EDBG virtual com port, you should see the following message:

You will see a lot of scrolling, but you want to see the following:

```
SUCCESS:  Subscribed to the MQTT update topic subscription
```

It should take the board at least two attempts to successfully connect after being provisioned. On the first attempt, AWS IoT will disconnect the device because the device certificate is not registered yet. However, this should kick off the device registration Lambda function (ZTLambdaJITR) in AWS to perform the actual registration. The board's second attempt to connect should succeed assuming the registration process has completed by then.

Note that all asymmetric math (authentication and key agreement) used during the TLS handshake is routed through the ATECC508A from the WINC1500. The WINC1500 has a callback system that sends requests for ECC crypto operations to the MCU. The MCU then sends these requests to the ATECC508A and returns the results back to the WINC1500.

The board uses AWS IoT's shadow system topics to inform AWS of state changes (button presses) and to learn of requested state changes (LED status).

- Device Shadows - http://docs.aws.amazon.com/iot/latest/developerguide/iot-thing-shadows.html
- Device Shadow Topics - http://docs.aws.amazon.com/iot/latest/developerguide/thing-shadow-mqtt.html

The board subscribes to the `$aws/things/thingName/shadow/update/delta` topic, which will send out messages whenever the reported device state differs from the desired device state. The board receives LED state updates through this topic.

The board separately publishes to the `$aws/things/thingName/shadow/update` topic to inform AWS of button state changes.

Let's summarize what you have done so far:

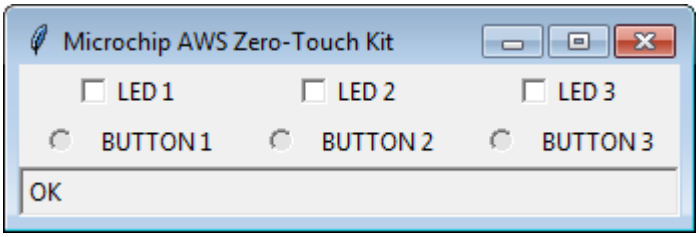
- Created a device certificate from the "kit's" identity key,
- Signed it with the Signer CA,
- Saved the kit's device certificate and signer certificate to the secure element (ATECC508A)
- Told the kit where to connect (AWS IoT endpoint)

VIII. AWS IoT Interaction

Now that the board has been provisioned, we will pass some simple messages back and forth to toggle the LEDs and show button state.

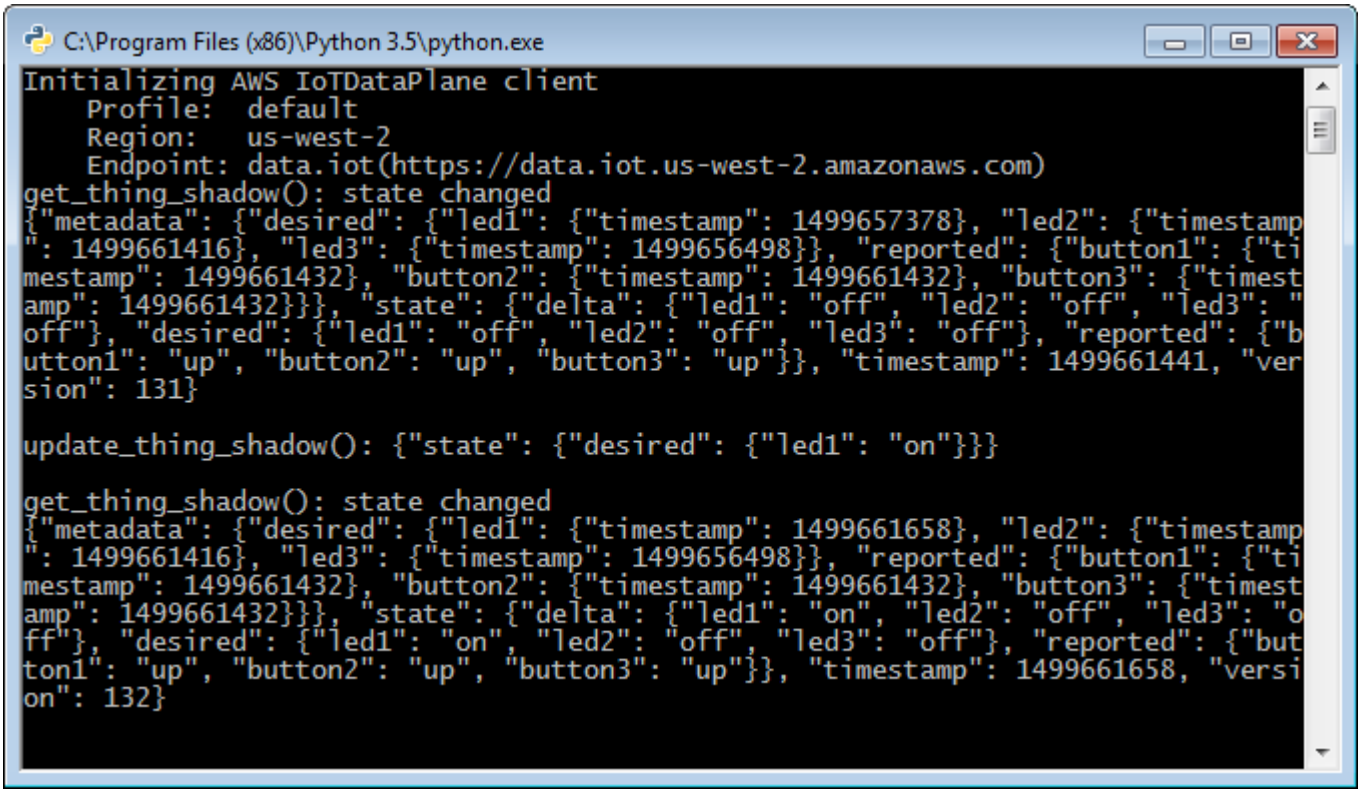
Run the `aws_interact_gui.py` python script.

After successfully connecting the AWS from the PC side, it will create a simple interface for interacting with the board.

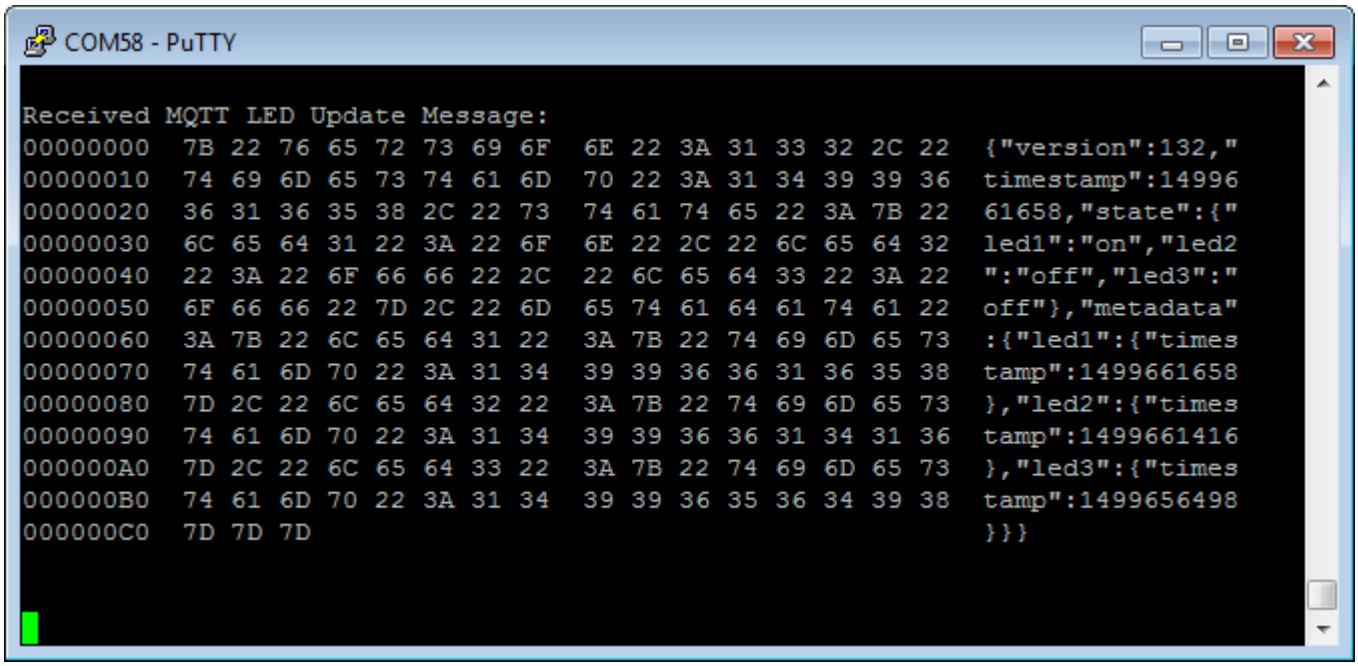


Selecting any of the LED checkboxes will turn on or off the LEDs on the OLED1 Xplained Pro board. Likewise, pressing the buttons on the board will light up the indicators in the interface, showing their current state.

The script console window will show the messages being passed back and forth.



Likewise, the debug output from the EDBG virtual com port in PuTTY/TeraTerm will show the corresponding messages on the device side.



Let's summarize what you have done so far:

Allowed the kit to:

- Connect and perform the JITR
- Communicate via its shadow

IX. Summary and Next Steps

You have created a device that is able to communicate with the Cloud (Amazon AWS).

The device (thing) shadow is the place you communicate with your device via a smart device app or web browser.

Explore:

- Firmware that comes in the ZIP to see how the ARM SAM G55 communicates with the secure element (ATECC508A) and the Wi-Fi module WINC1500.

X. Troubleshooting

If you are having problems, please refer to the Microchip Support pages: http://www.microchip.com/support/