POLITECNICO
MILANO 1863

AWS IoT

Parte prima

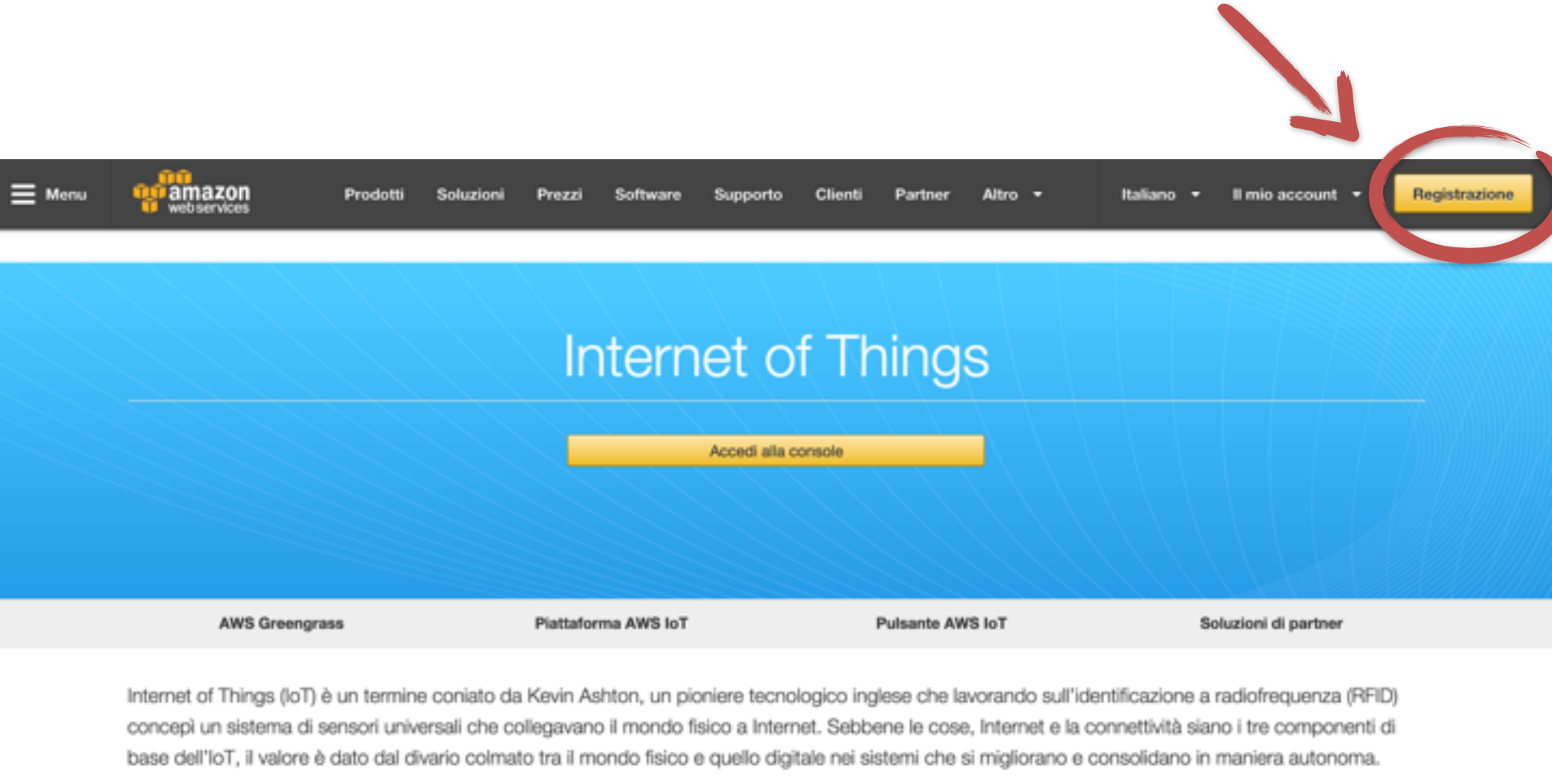
Marco D. Santambrogio – marco.santambrogio@polimi.it
Emanuele Del Sozzo – emanuele.delsozzo@polimi.it
Lorenzo Di Tucci – lorenzo.ditucci@mail.polimi.it
Giuseppe Natale – giuseppe.natale@polimi.it
Marco Rabozzi – marco.rabozzi@polimi.it
Alberto Scolari – alberto.scolari@polimi.it
Matteo Ferroni – matteo.ferroni@polimi.it
Ver. aggiornata al 18/12/2016

Outline

- Registration
- The AWS Command Line Interface
- Setup your first “thing”
- Connect, publish and subscribe using an MQTT client
- Connect, publish and subscribe using the Python SDK
- MQTT over WebSocket

Registration

https://aws.amazon.com/it/iot/

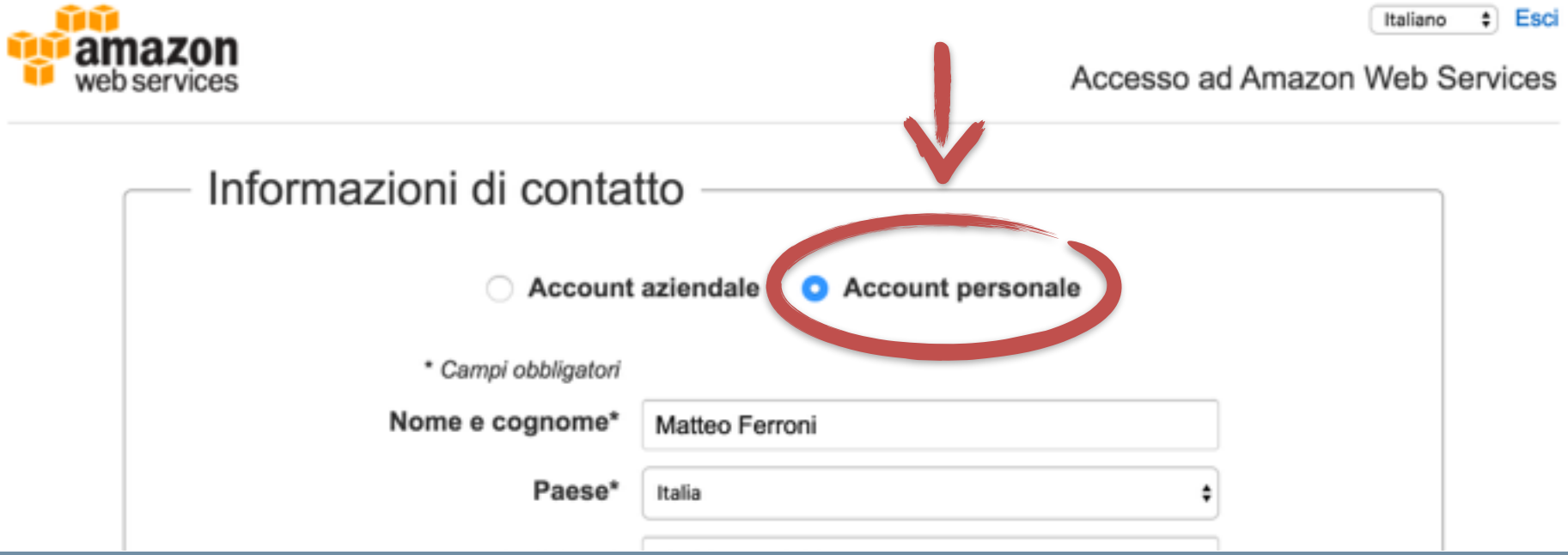


Registration

https://aws.amazon.com/it/iot/

Registration

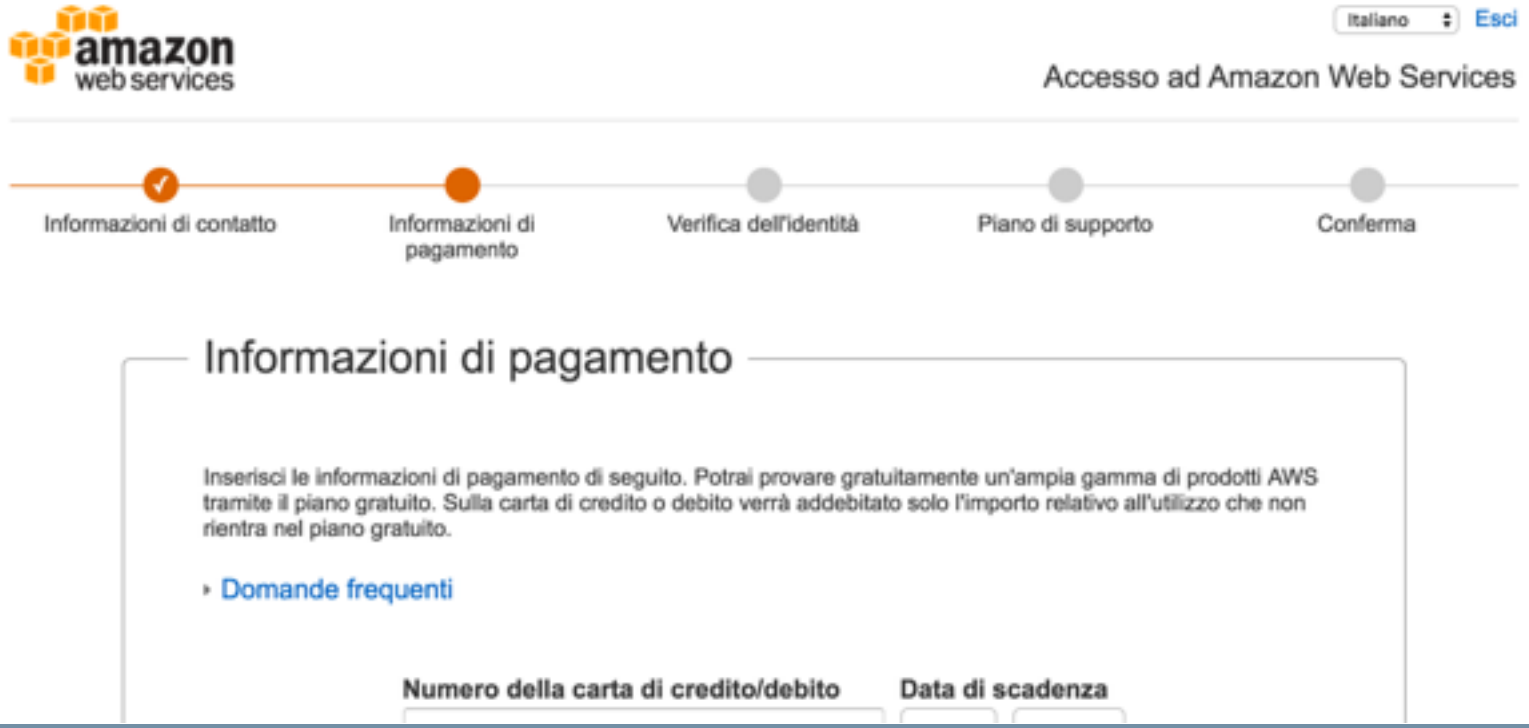
- Potete crearvi un account personale per iniziare a sperimentare le possibilità di AWS IoT
- Il giorno della hackathon, avrete a disposizione degli account illimitati, cortesemente offerti da techedge



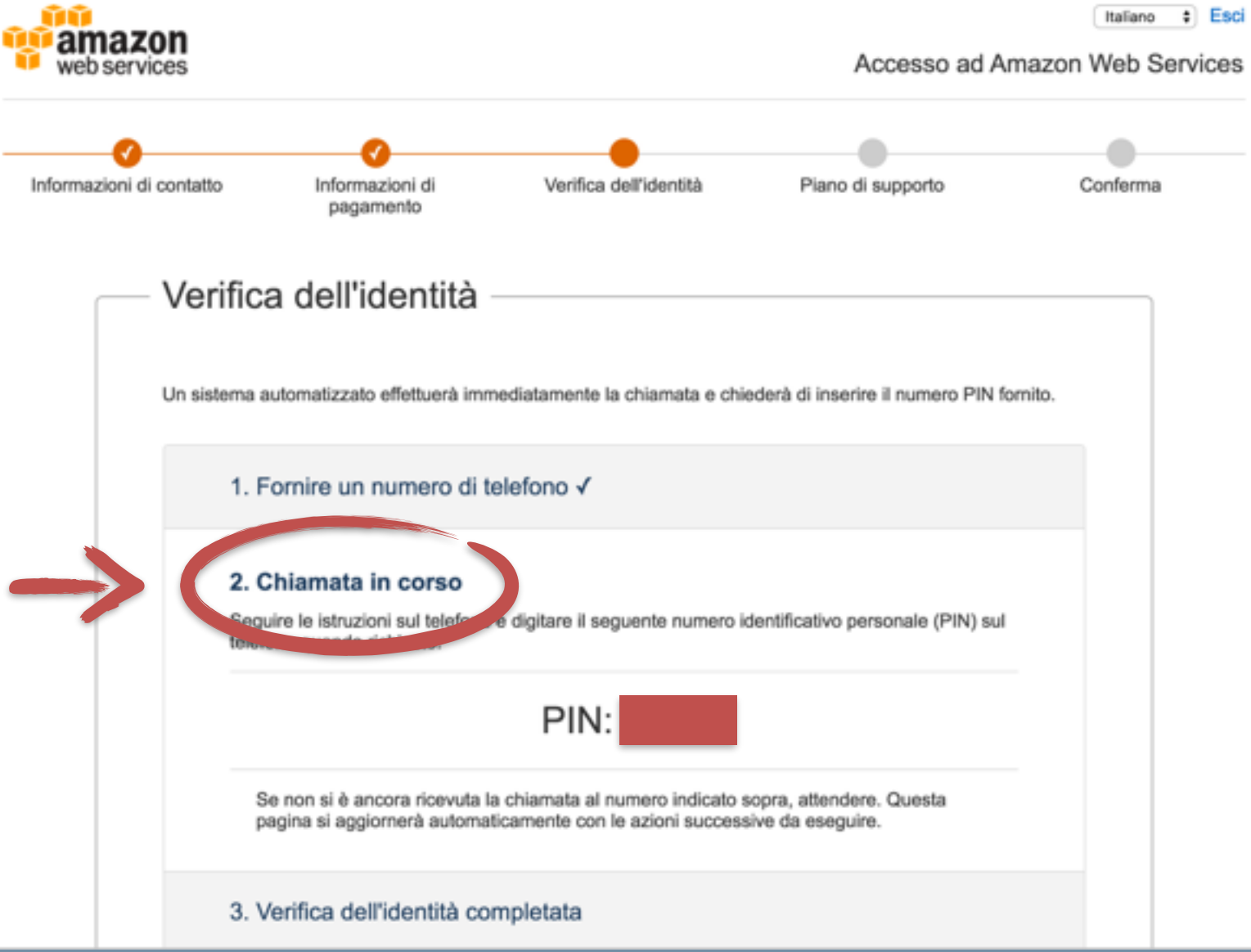
Registration

Perché devo fornire le informazioni di pagamento?

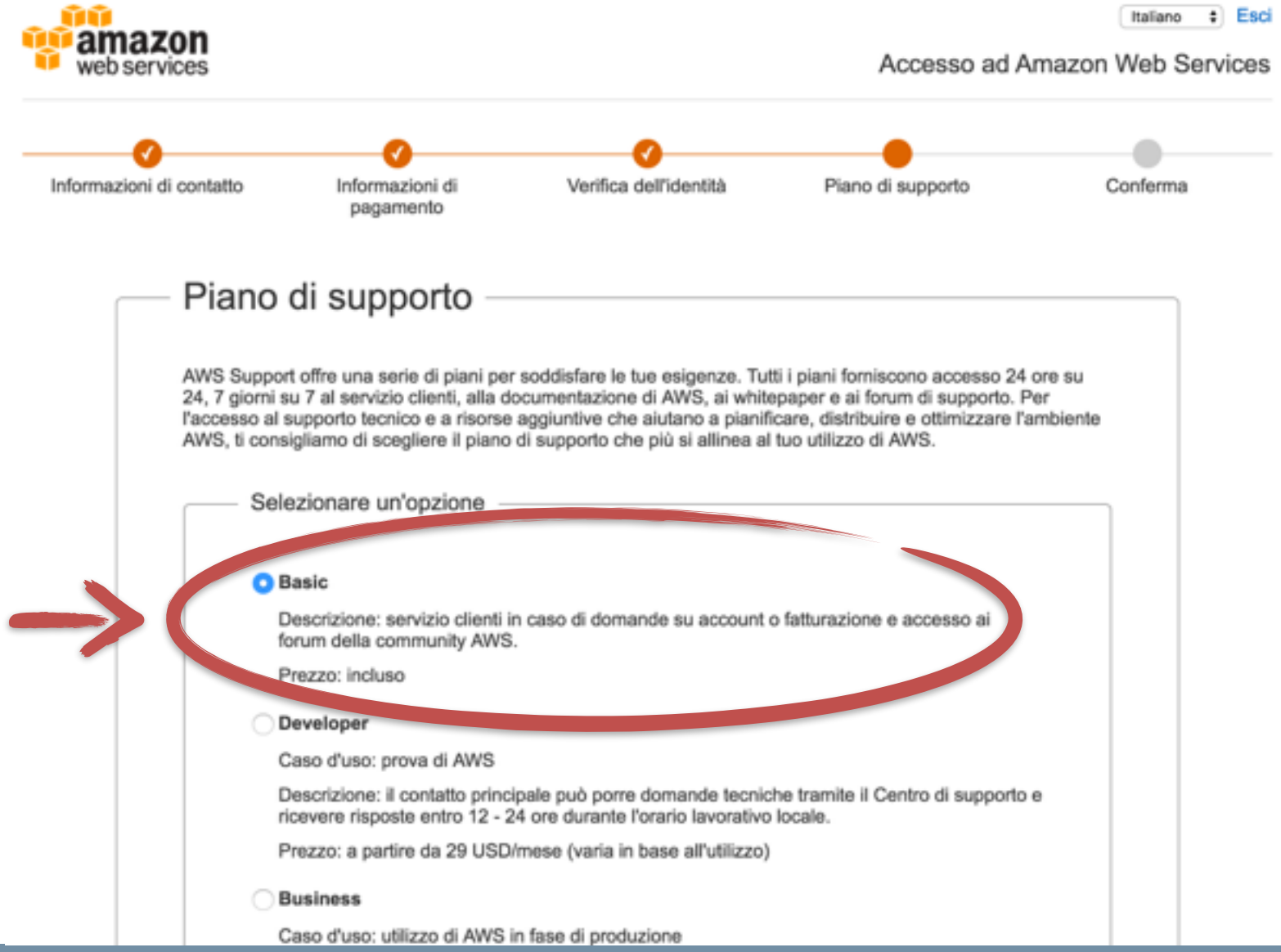
Chiediamo una carta di credito o di debito per agevolare il passaggio all'utilizzo dei servizi AWS a pagamento qualora l'account superi i limiti del piano gratuito di AWS. Inoltre, utilizziamo le informazioni di pagamento per verificare l'autenticità del tuo account e per prevenire attività fraudolente.



Registration



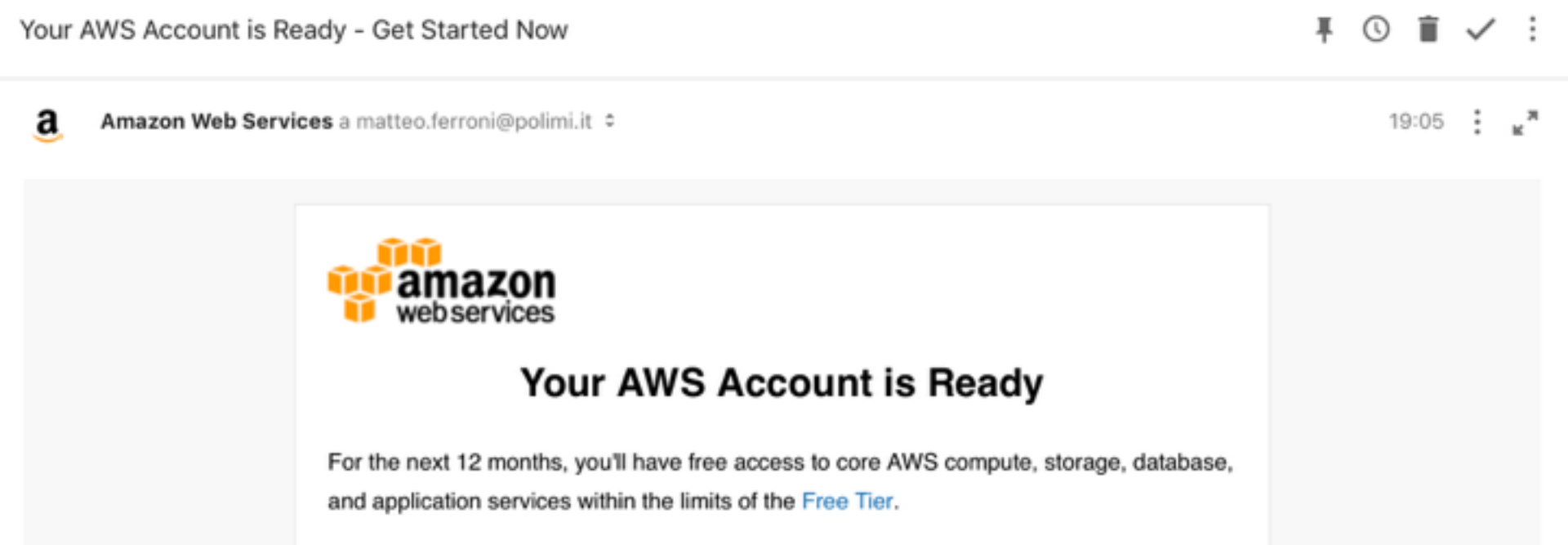
Registration



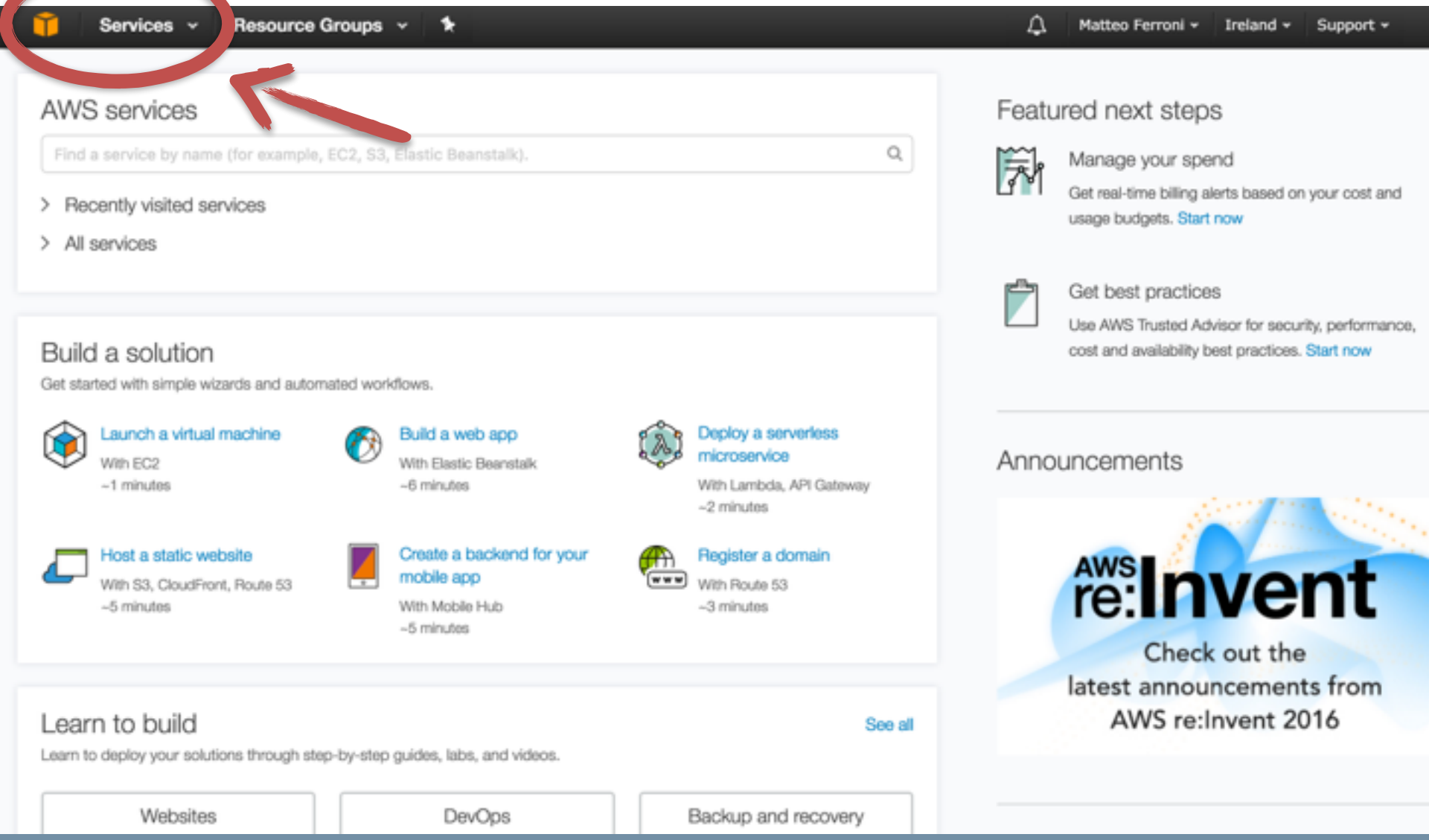
Registration

https://aws.amazon.com/it/free/

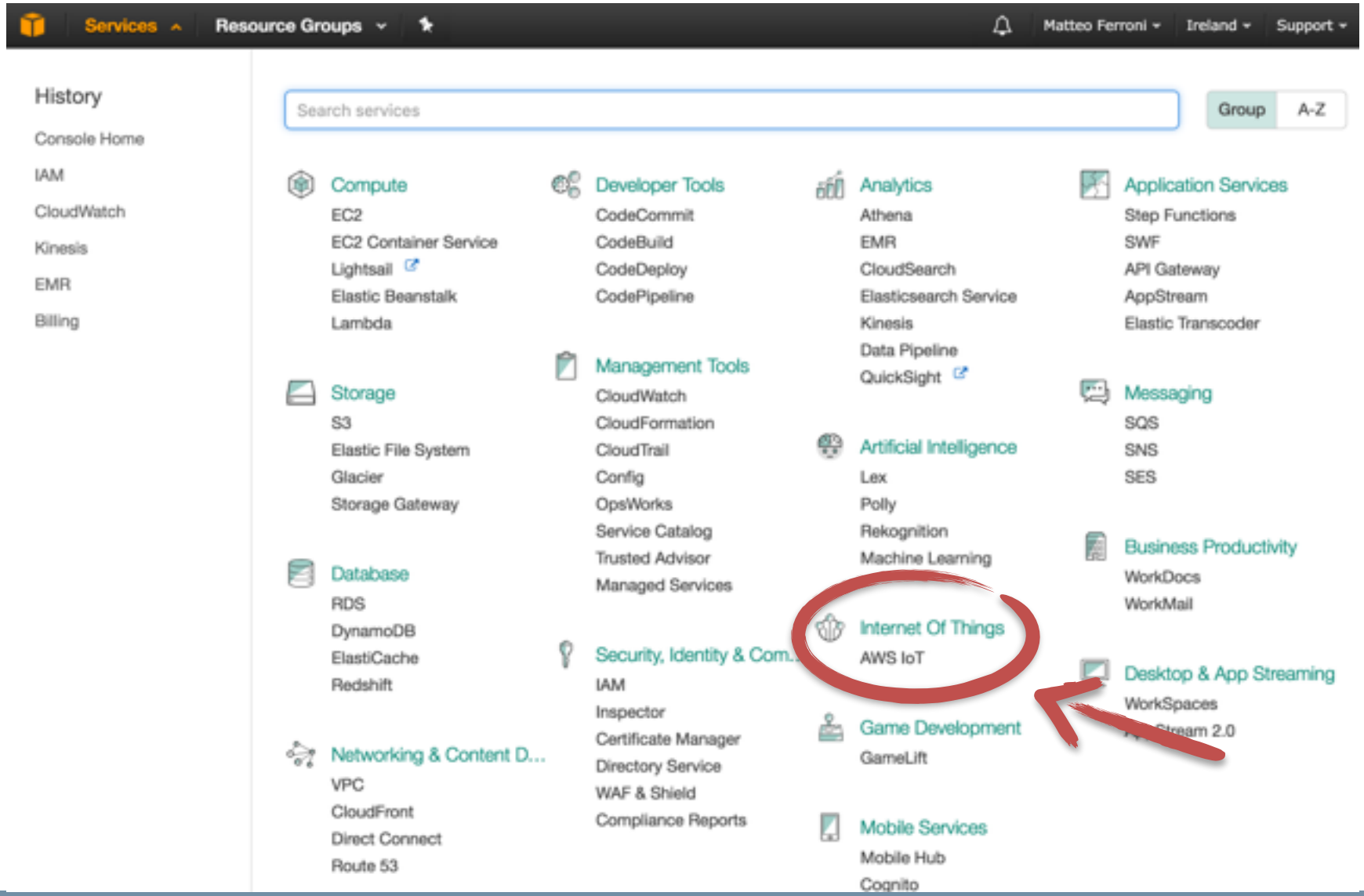
Il piano gratuito di Amazon Web Services (AWS) è stato ideato per consentirti di acquisire esperienza diretta dei servizi cloud di AWS. Il piano gratuito di AWS consente di usare una serie di servizi per 12 mesi a partire dalla data di registrazione più altre offerte che non scadono automaticamente dopo i 12 mesi previsti dai termini del piano gratuito.



Services


Services
Resource Groups
Matteo Ferroni
Ireland
Support
AWS services
Find a service by name (for example, EC2, S3, Elastic Beanstalk).
Recently visited services
All services
Build a solution
Get started with simple wizards and automated workflows.
Launch a virtual machine
With EC2
~1 minutes
Build a web app
With Elastic Beanstalk
~6 minutes
Deploy a serverless microservice
With Lambda, API Gateway
~2 minutes
Host a static website
With S3, CloudFront, Route 53
~5 minutes
Create a backend for your mobile app
With Mobile Hub
~5 minutes
Register a domain
With Route 53
~3 minutes
Learn to build
See all
Learn to deploy your solutions through step-by-step guides, labs, and videos.
Websites
DevOps
Backup and recovery
Featured next steps
Manage your spend
Get real-time billing alerts based on your cost and usage budgets. Start now
Get best practices
Use AWS Trusted Advisor for security, performance, cost and availability best practices. Start now
Announcements
AWS re:Invent
Check out the latest announcements from AWS re:Invent 2016

Services


Services
Resource Groups
Matteo Ferroni
Ireland
Support
History
Console Home
IAM
CloudWatch
Kinesis
EMR
Billing
Search services
Group
A-Z
Compute
EC2
EC2 Container Service
Lightsail
Elastic Beanstalk
Lambda
Developer Tools
CodeCommit
CodeBuild
CodeDeploy
CodePipeline
Analytics
Athena
EMR
CloudSearch
Elasticsearch Service
Kinesis
Data Pipeline
QuickSight
Application Services
Step Functions
SWF
API Gateway
AppStream
Elastic Transcoder
Storage
S3
Elastic File System
Glacier
Storage Gateway
Management Tools
CloudWatch
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Trusted Advisor
Managed Services
Artificial Intelligence
Lex
Polly
Rekognition
Machine Learning
Messaging
SQS
SNS
SES
Database
RDS
DynamoDB
ElastiCache
Redshift
Business Productivity
WorkDocs
WorkMail
Internet Of Things
AWS IoT
Security, Identity & Com...
IAM
Inspector
Certificate Manager
Directory Service
WAF & Shield
Compliance Reports
Desktop & App Streaming
WorkSpaces
Networking & Content D...
VPC
CloudFront
Direct Connect
Route 53
Game Development
GameLift
Mobile Services
Mobile Hub
Cognito

AWS IoT - Useful references

https://console.aws.amazon.com/

Console di gestione AWS

http://docs.aws.amazon.com/iot/latest/developerguide/iot-sdks.html

AWS IoT SDKs

https://github.com/aws/aws-iot-device-sdk-python

AWS IoT SDK for Python

https://github.com/dwyl/learn-aws-iot

Learn how to use Amazon Web Services Internet of Things (IoT) service to build connected applications

Installing the AWS Command Line Interface

http://docs.aws.amazon.com/cli/latest/userguide/installing.html

*The **AWS Command Line Interface** is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.*

Choose an Installation Method

There are a number of different ways to install the AWS CLI on your machine, depending on what operating system and environment you are using:

- **On Microsoft Windows** – use the MSI installer.
- **On Linux, OS X, or Unix** – use pip (a package manager for Python software) or install manually with the bundled installer.

Note

On OS X, if you see an error regarding the version of six that came with distutils in El Capitan, use the `--ignore-installed` option:

```
$ sudo pip install awscli --ignore-installed six
```

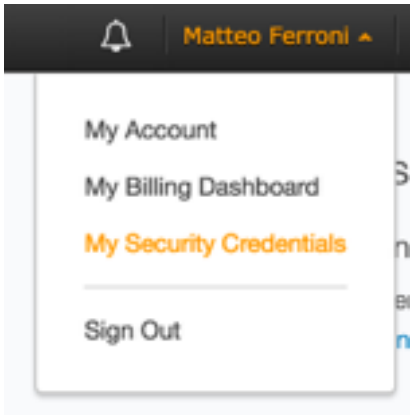
Access key ID and secret access key

https://console.aws.amazon.com/iam/home?#/home

Access keys *consist of an access key ID and secret access key, which are used to sign programmatic requests that you make to AWS. If you don't have access keys, you can create them by using the AWS Management Console. We recommend that you use IAM access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account.*

Access key ID and secret access key

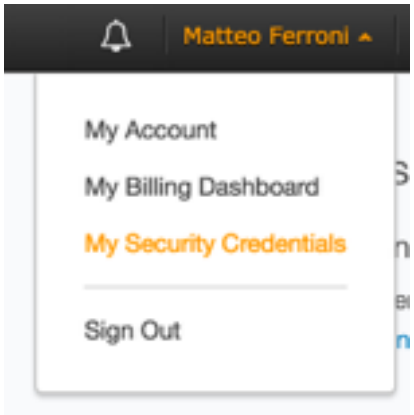
https://console.aws.amazon.com/iam/home?#/home



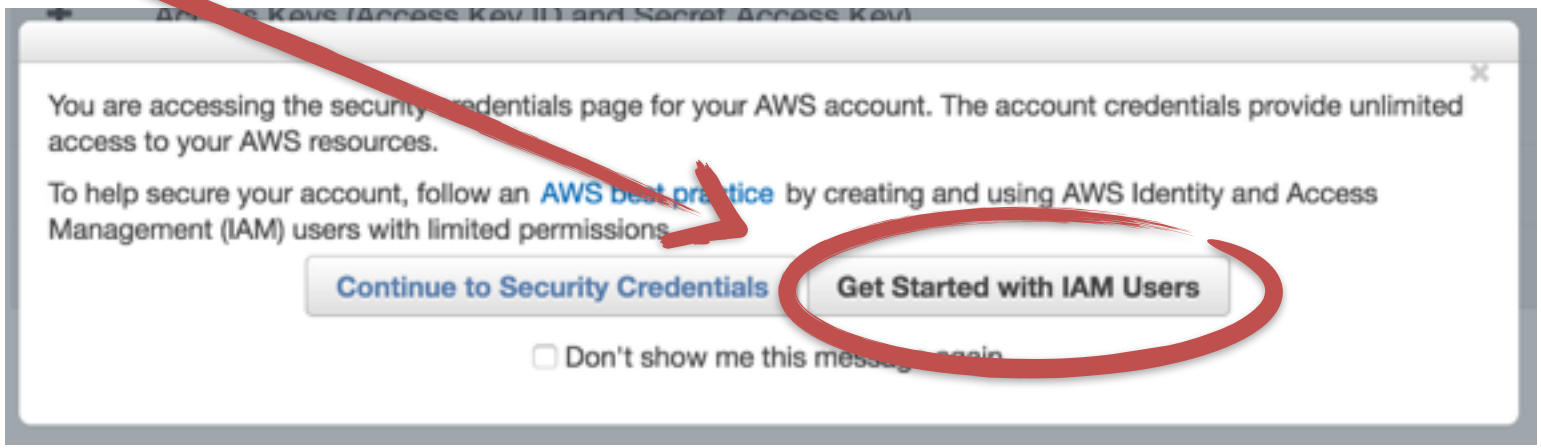
Access keys *consist of an access key ID and secret access key, which are used to sign programmatic requests that you make to AWS. If you don't have access keys, you can create them by using the AWS Management Console. We recommend that you use IAM access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account.*

Access key ID and secret access key

https://console.aws.amazon.com/iam/home?#/home



Access keys *consist of an access key ID and secret access key, which are used to sign programmatic requests that you make to AWS. If you don't have access keys, you can create them by using the AWS Management Console. We recommend that you use IAM access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account.*

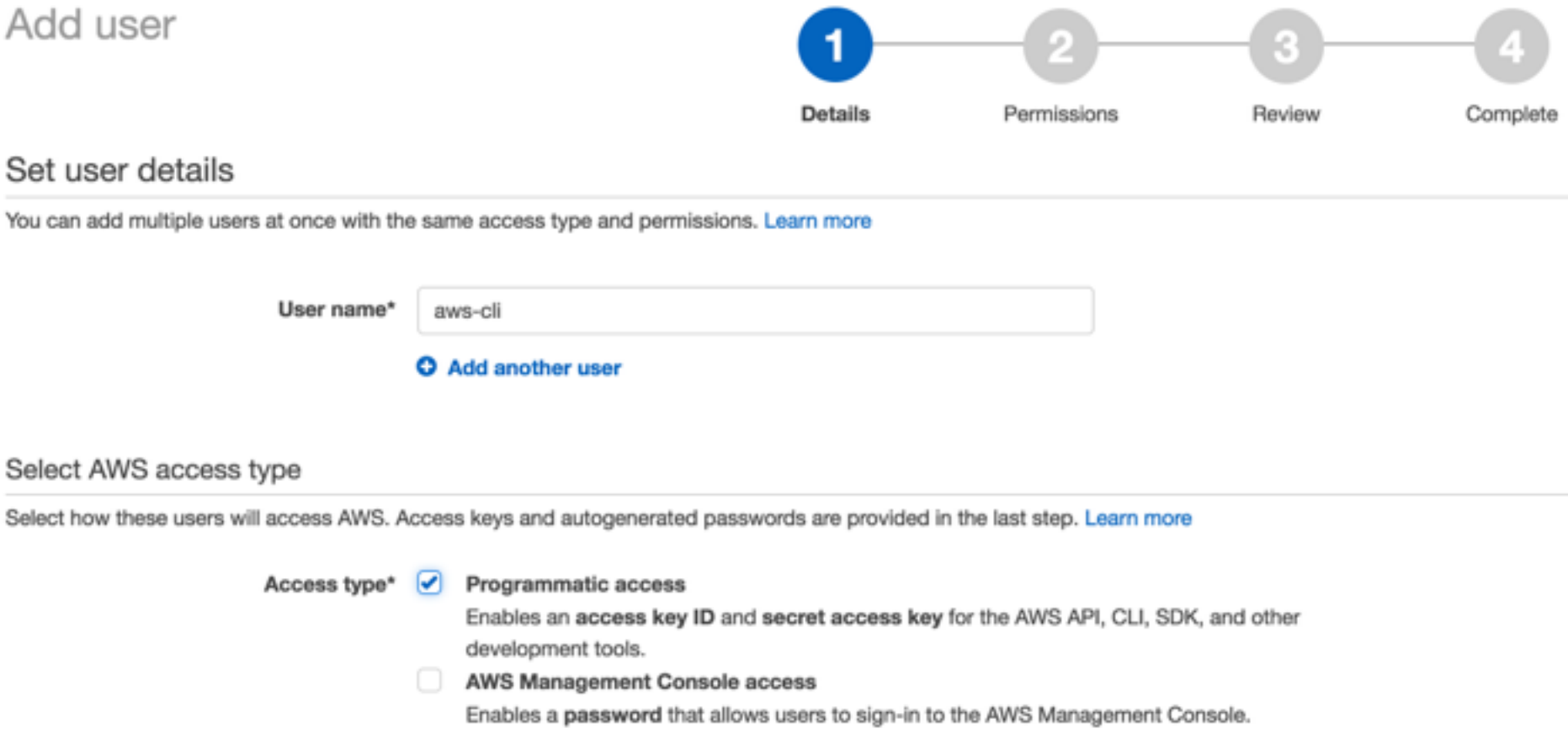


Access key ID and secret access key

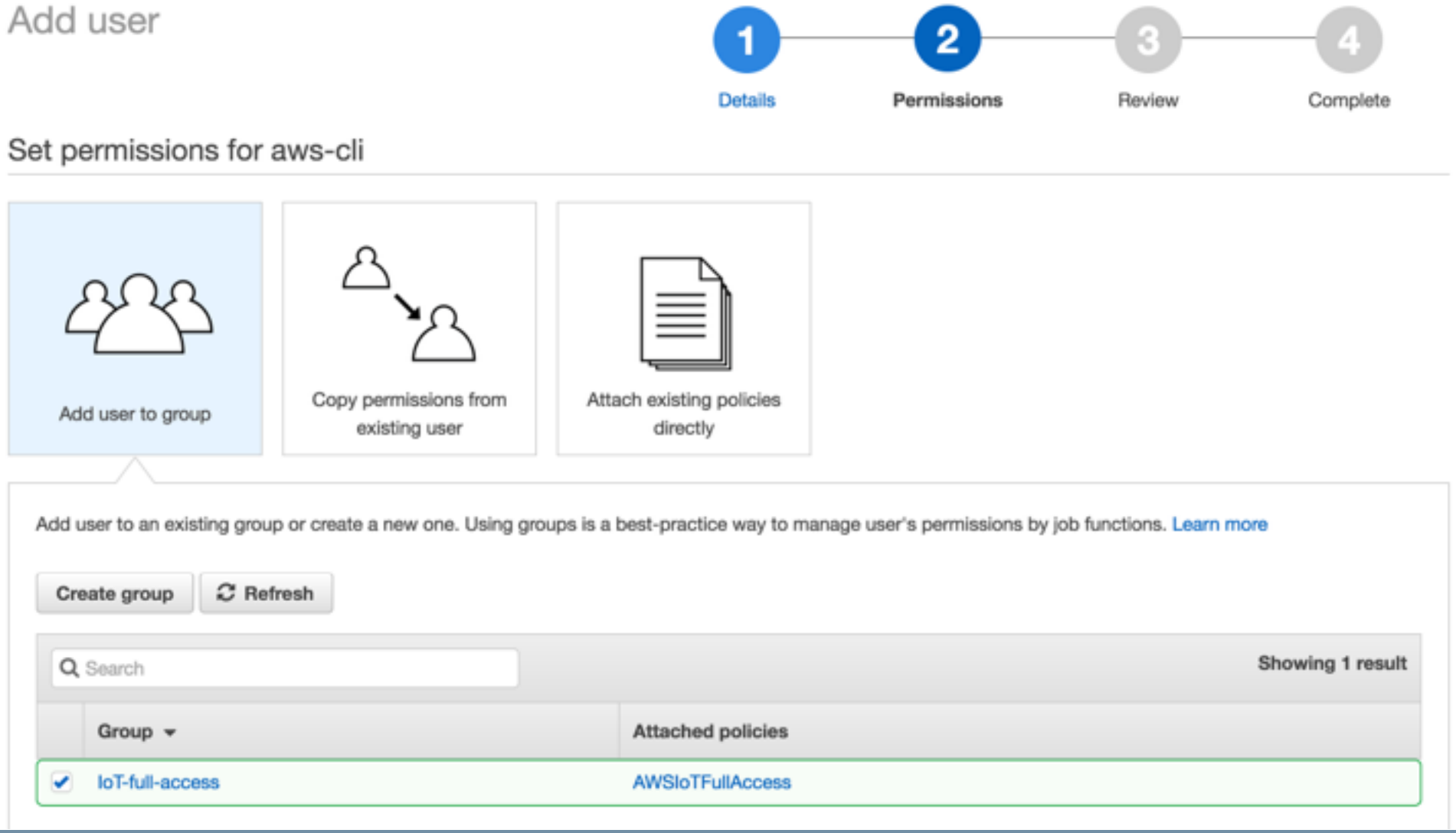
1. Open the IAM console.
2. In the navigation pane, choose **Users**.
3. Choose your IAM user name (not the check box).
4. Choose the **Security Credentials** tab and then choose **Create Access Key**.
5. To see your access key, choose **Show User Security Credentials**. Your credentials will look something like this:
   - Access Key ID: AKIAIOSFODNN7EXAMPLE
   - Secret Access Key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. Choose **Download Credentials**, and store the keys in a secure location.

Your secret key will no longer be available through the AWS Management Console; you will have the only copy. Keep it confidential in order to protect your account, and never email it. Do not share it outside your organization, even if an inquiry appears to come from AWS or Amazon.com. No one who legitimately represents Amazon will ever ask you for your secret key.

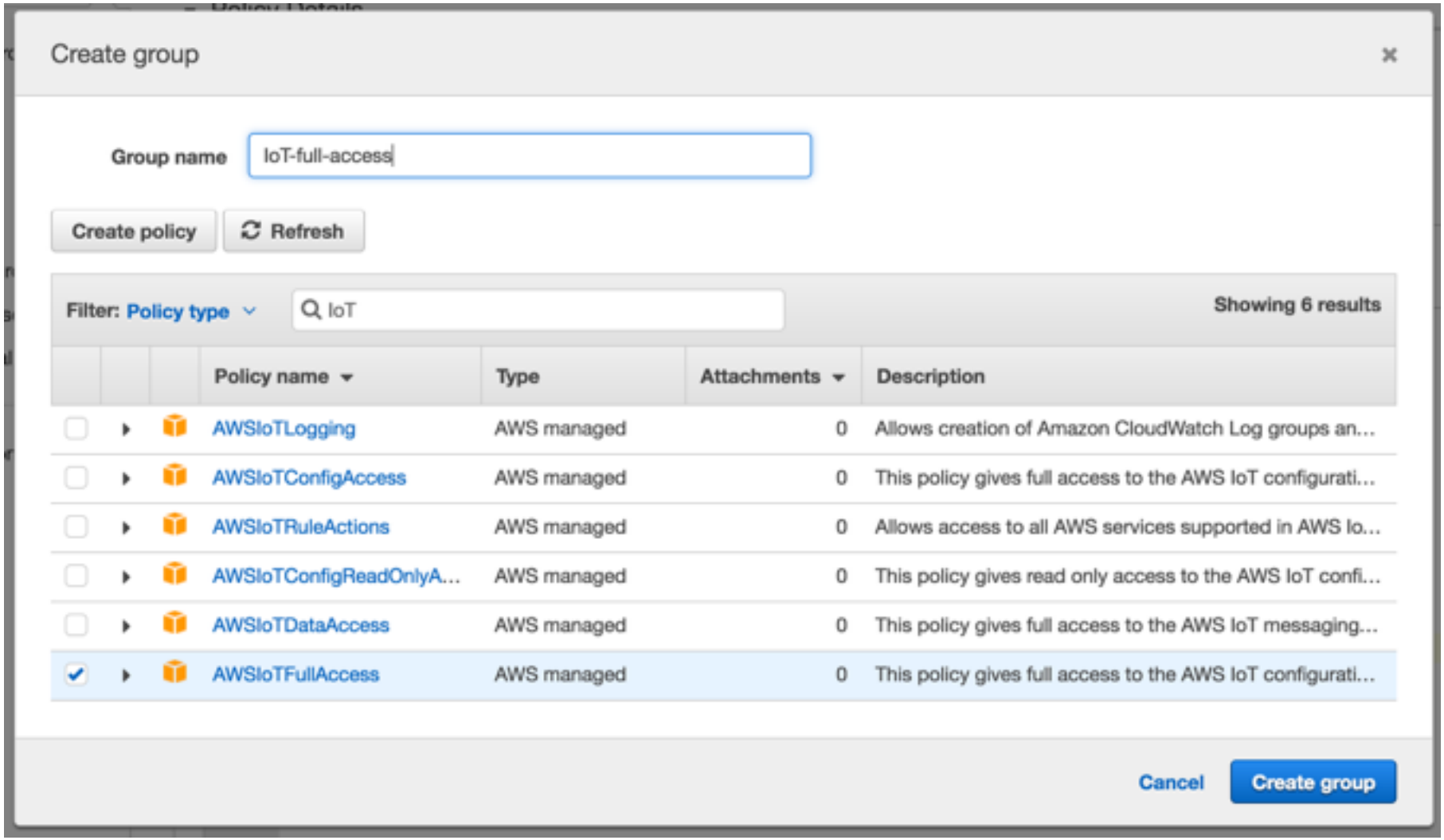
Add user - Details



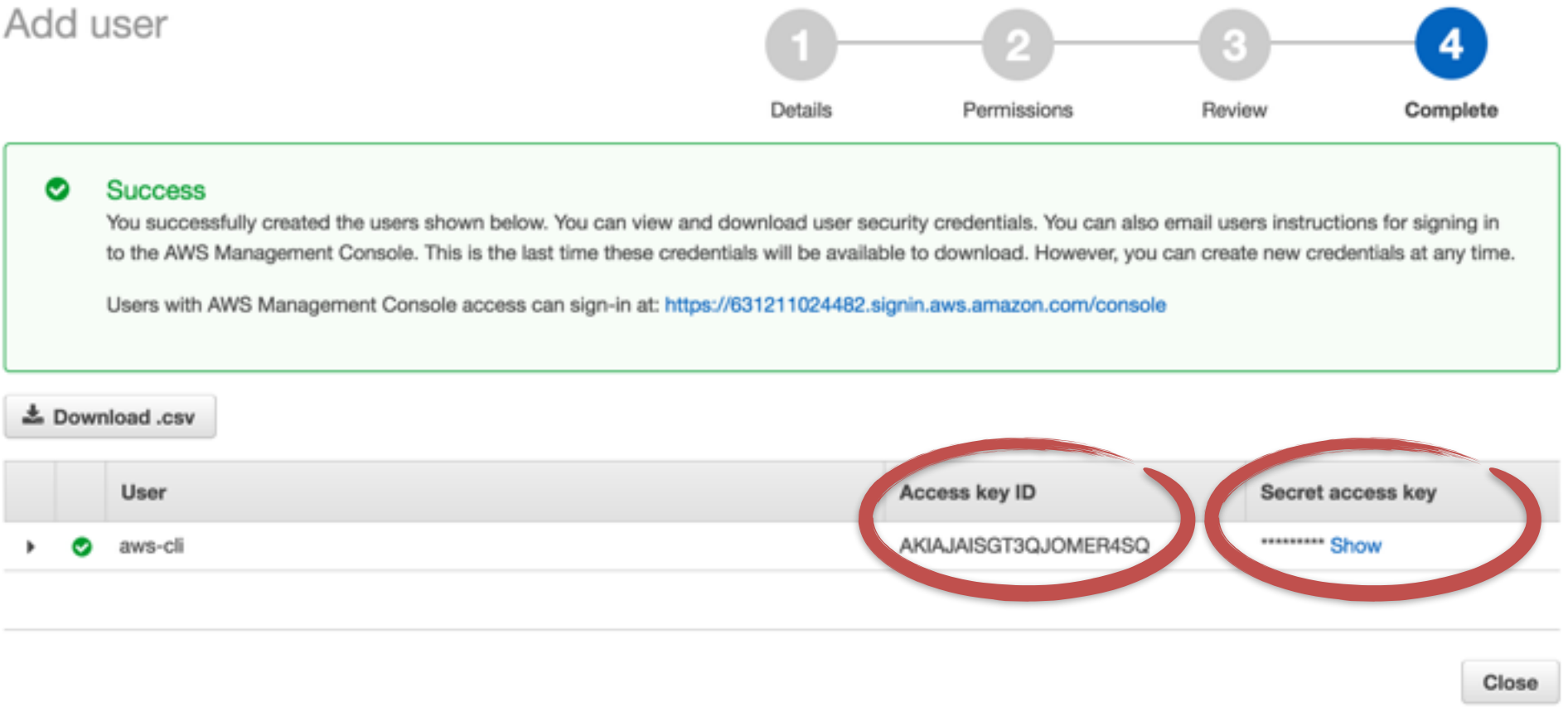
Add user - Permissions



Add user - Create group



Add user - Completed



Configuring the AWS Command Line Interface

http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html

Regione (suggerita): ***eu-west-1***

Output (suggerito): ***json***

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: ENTER
```

Using the AWS Command Line Interface

- **Create a "thing"**

```
aws iot create-thing
    --thing-name "TestDevice1"
```

Response:

```
{
 "thingArn":
    "arn:aws:iot:eu-west-1:631211024482:thing/
    TestDevice1",
 "thingName":
    "TestDevice1"
}
```

Using the AWS Command Line Interface

Thing

- **Generate a certificate**

```
aws iot create-keys-and-certificate
    --set-as-active
    --certificate-pem-outfile cert.pem
    --public-key-outfile publicKey.pem
    --private-key-outfile privateKey.pem
```

Certificate

Response:

```
{
    “certificateArn”:
        "arn:aws:iot:eu-west-1:631211024482:cert/
        b0a162f3f5bfc207346e598a706c098db4003fa4330b5c1bddfd08f512
        987f31",
    “certificateId":
        "b0a162f3f5bfc207346e598a706c098db4003fa4330b5c1bddfd08f51
        2987f31",
    ...
}
```

Using the AWS Command Line Interface

- **Create a policy file: *policy.json***

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action":["iot:*"],
        "Resource": ["*"]
     }]
}
```

- **And load it in AWS**

```
aws iot create-policy
  --policy-name "first-policy"
  --policy-document file://PATH-TO-FILE
```

Using the AWS Command Line Interface

Thing

Certificate

Policy

Using the AWS Command Line Interface

- **Link certificate and policy**

```
aws iot attach-principal-policy
  --principal "ARN-CERTIFICATE"
  --policy-name “first-policy"
```

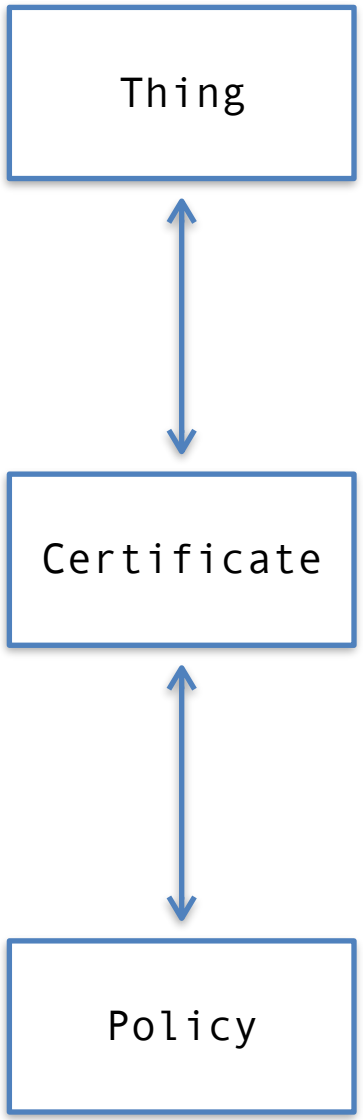
Using the AWS Command Line Interface

- **Link certificate and policy**

```
aws iot attach-principal-policy
  --principal "ARN-CERTIFICATE"
  --policy-name “first-policy"
```

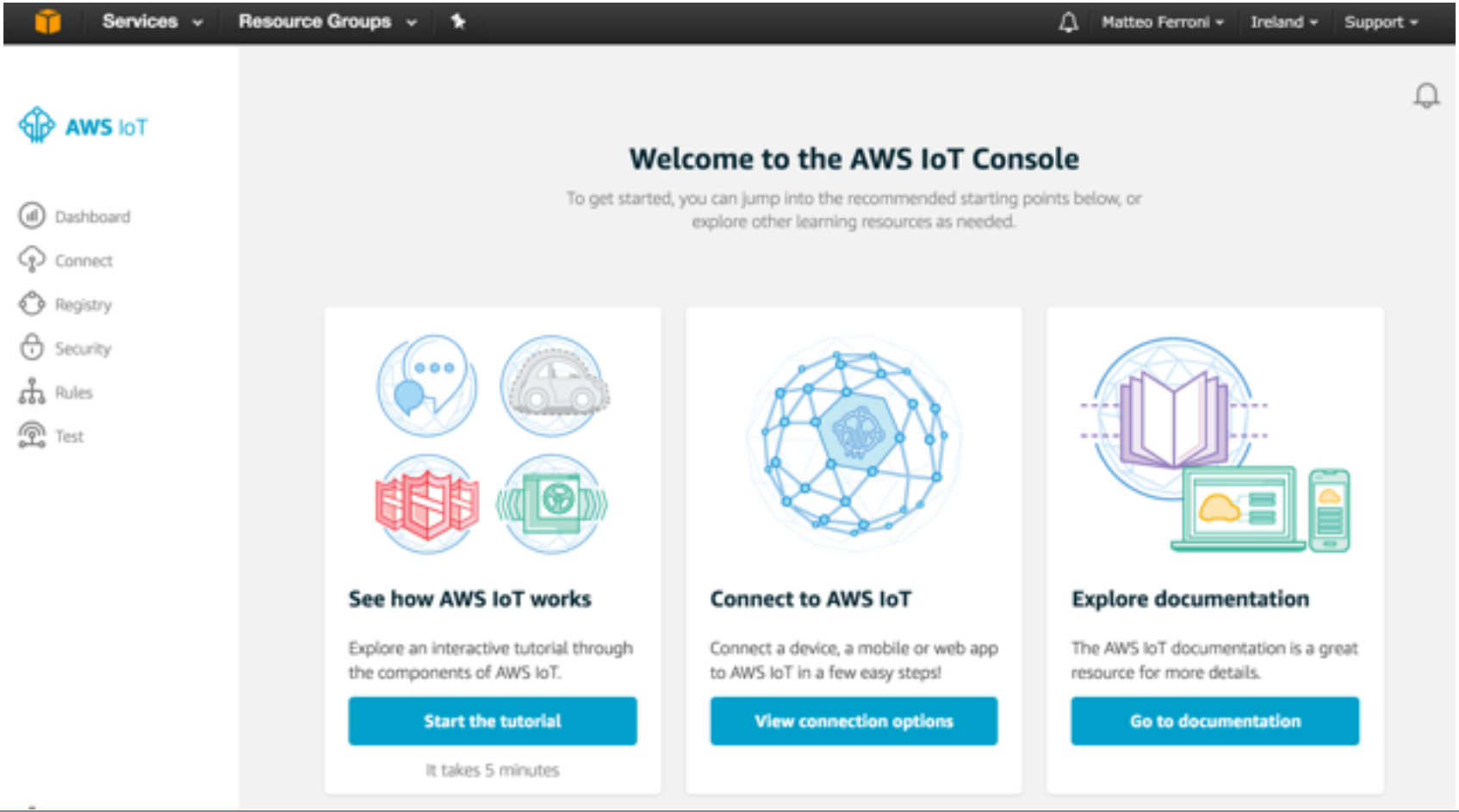
- **Link certificate and “thing”:**

```
aws iot attach-thing-principal
  --thing-name "TestDevice1"
  --principal "ARN-CERTIFICATE"
```



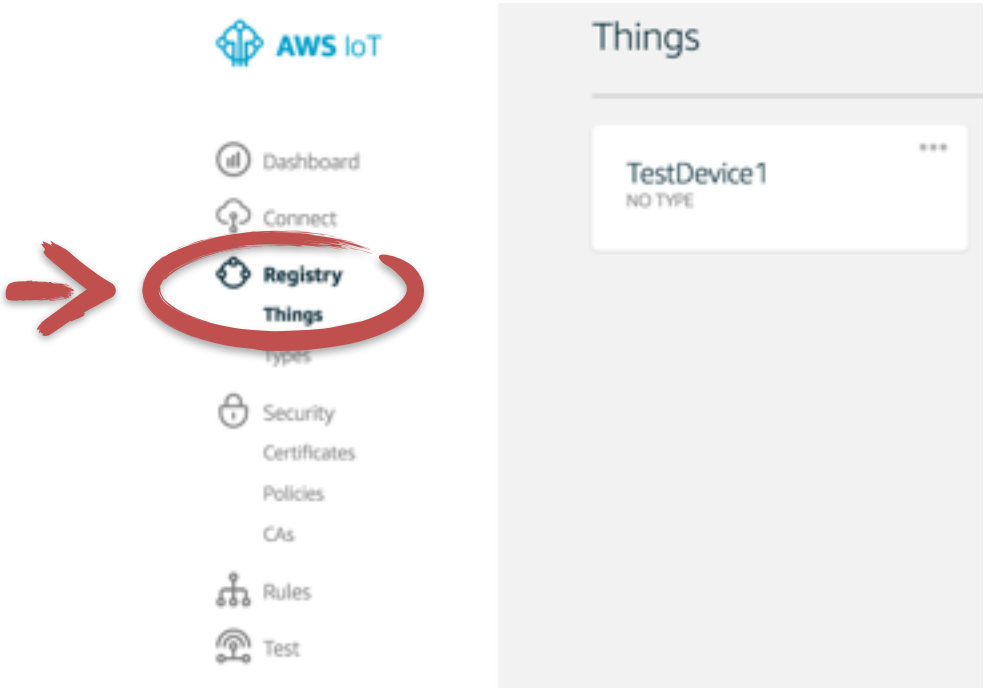
AWS IoT - Dashboard

Le stesse operazioni possono essere fatte da UI web



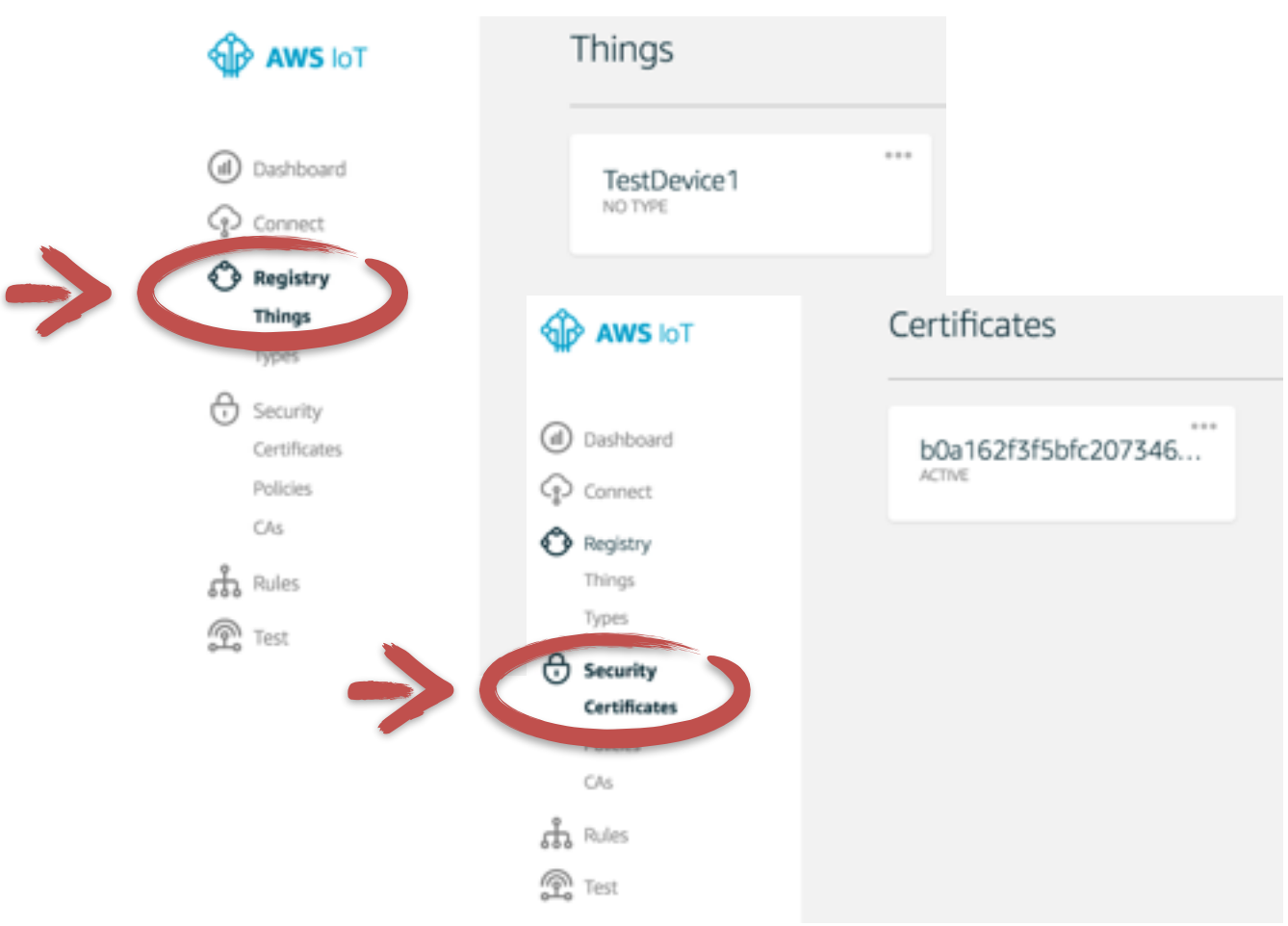
AWS IoT - Dashboard

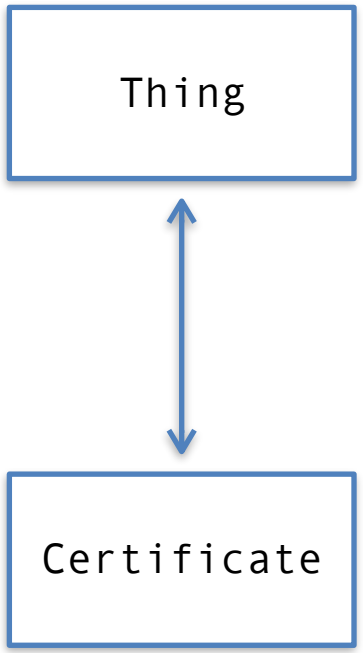
AWS IoT - Dashboard



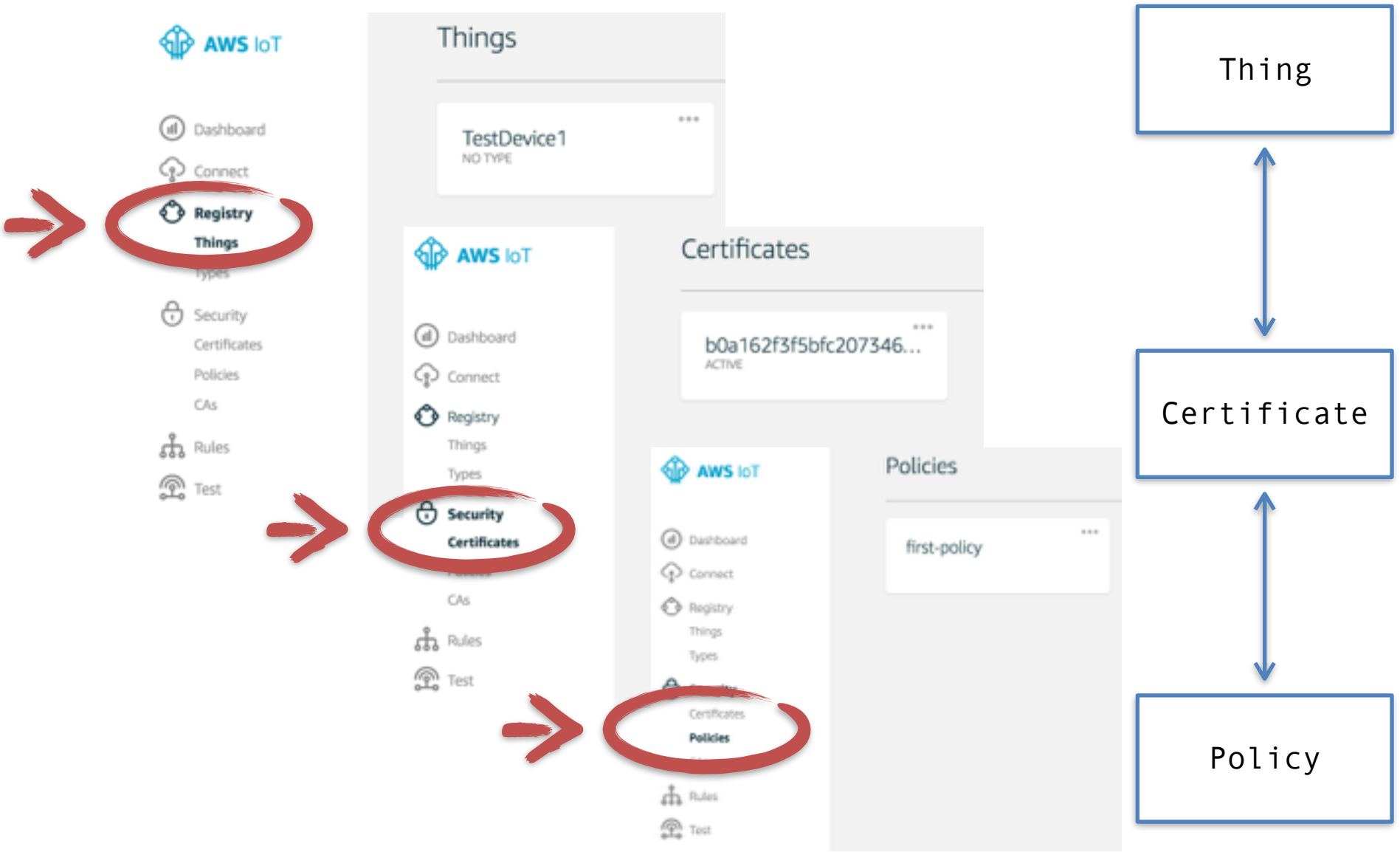
Thing

AWS IoT - Dashboard





AWS IoT - Dashboard



MQTT client example

- **Download *rootCA.pem* :**

  ```
  https://www.symantec.com/content/en/us/enterprise/verisign/
  roots/VeriSign-Class 3-Public-Primary-Certification-Authority-
  G5.pem
  ```

- **Download *MQTT.fx* :**

  ```
  http://mqttfx.jfx4ee.org/index.php/download
  ```

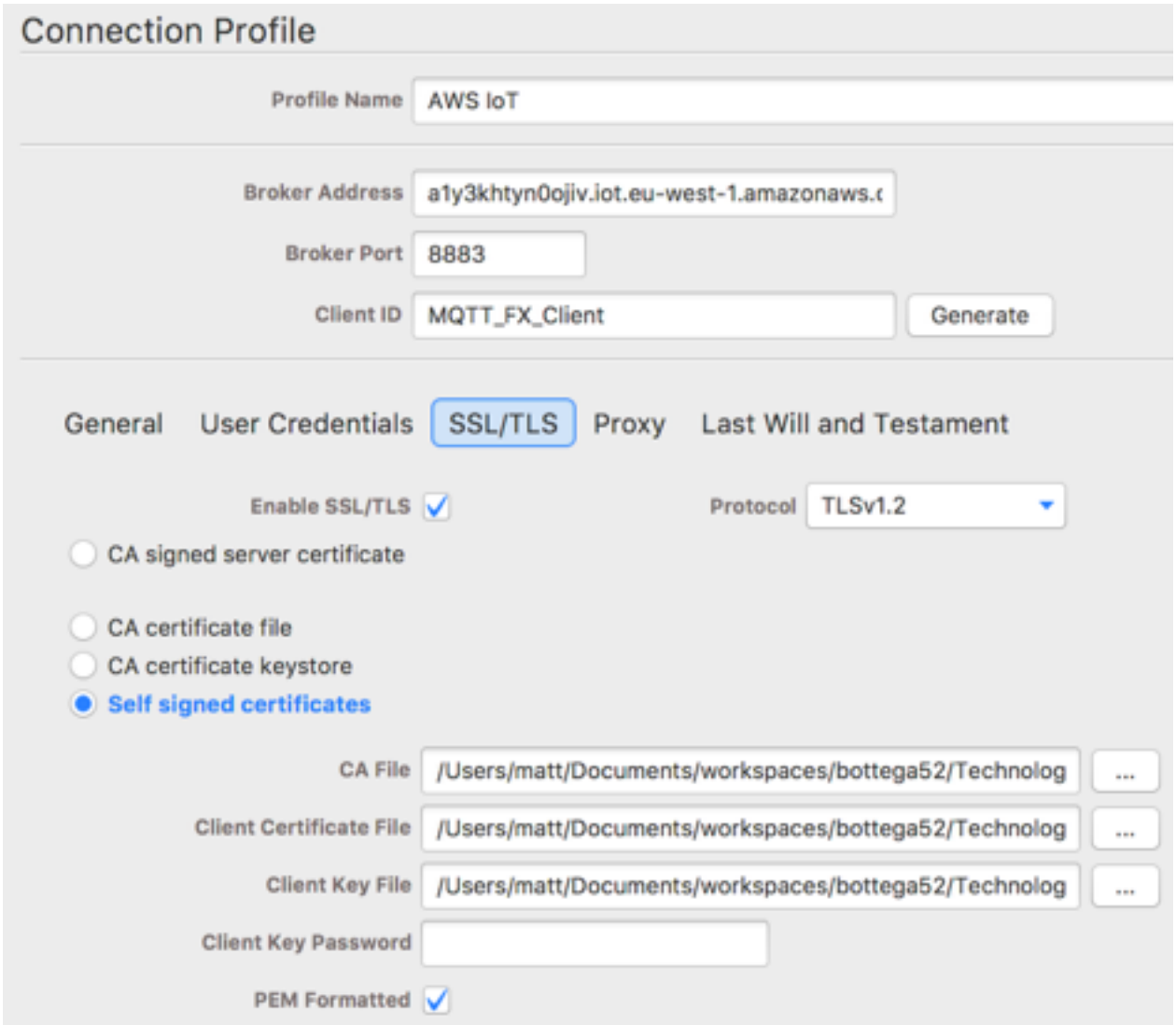
- **Trova l'indirizzo a cui connetterti:**

  ```
  aws iot describe-endpoint
  ```

Response:

```
{
    "endpointAddress":
        "a1c2qwq6b794bg.iot.eu-west-1.amazonaws.com"
}
```

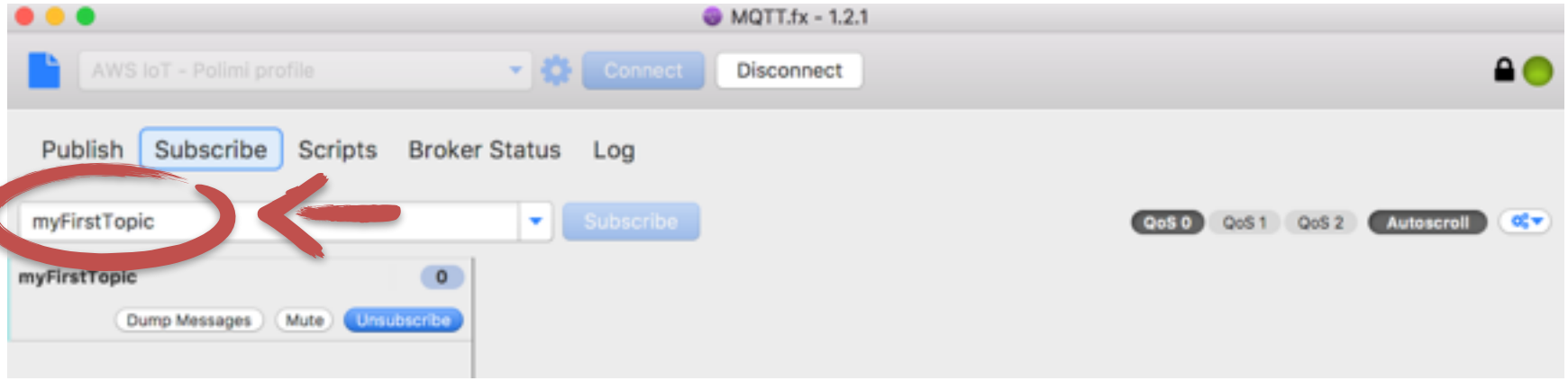
MQTT client example



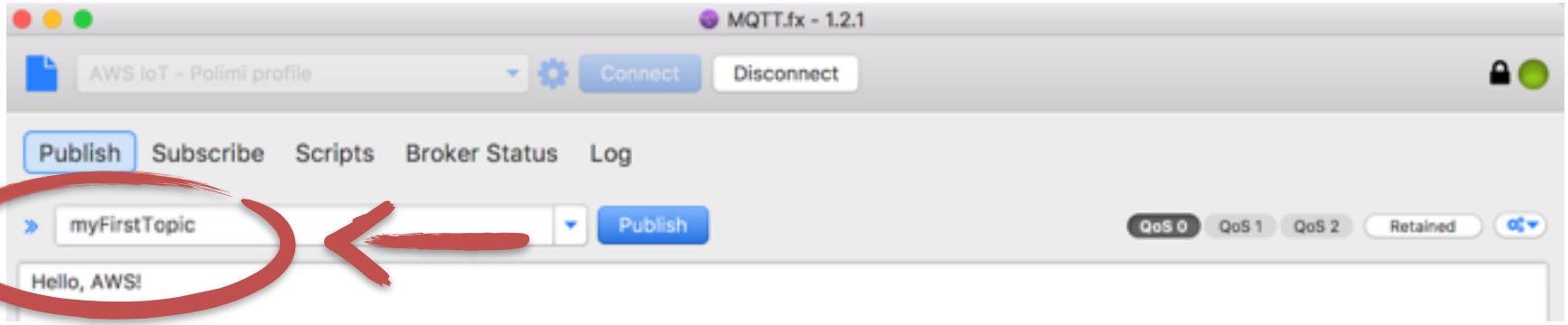
- Configura MQTT.fx usando l'indirizzo trovato e i certificati generati

Connect, Publish and Subscribe

- **Connect and subscribe**

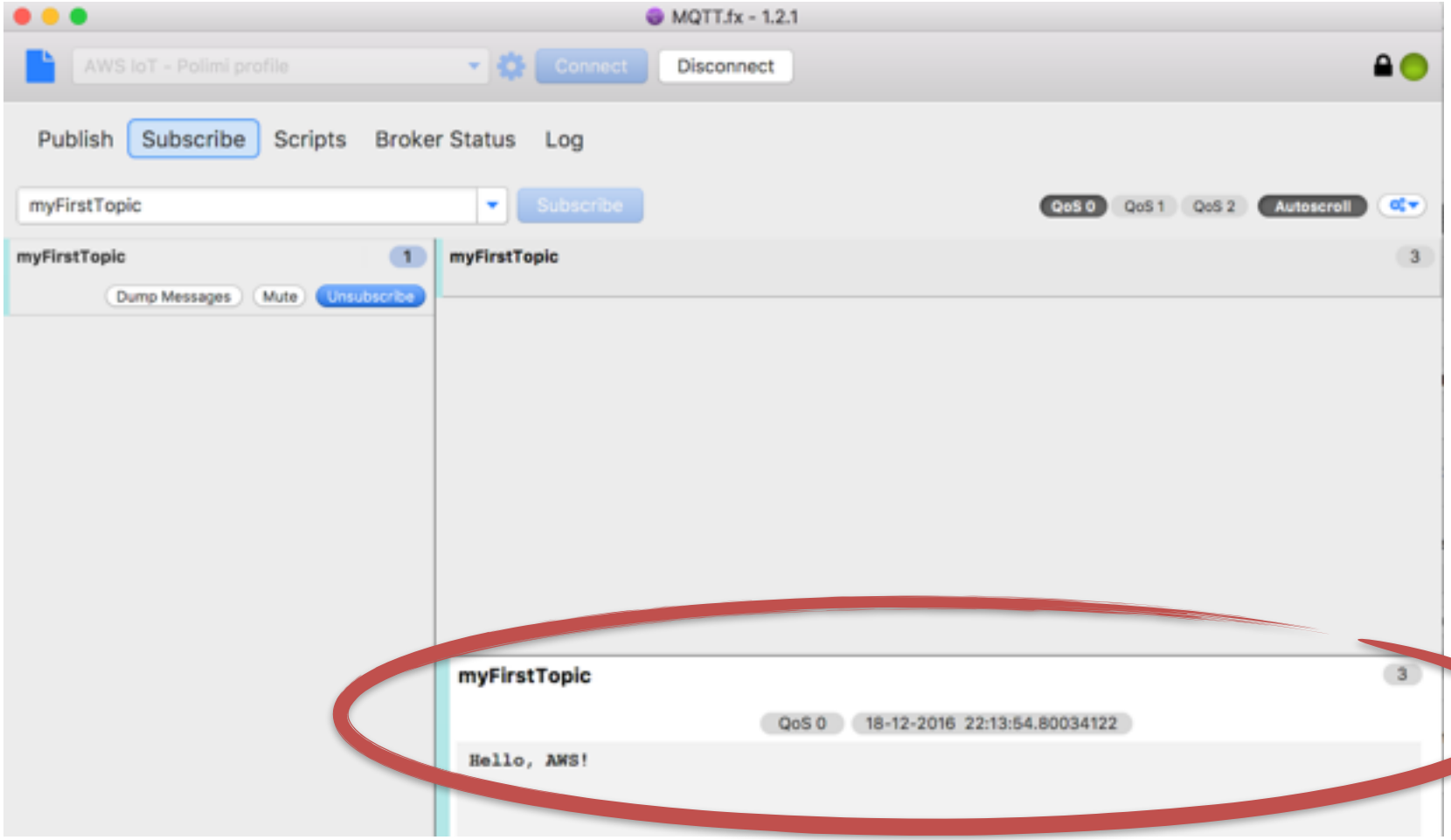


- **Publish**

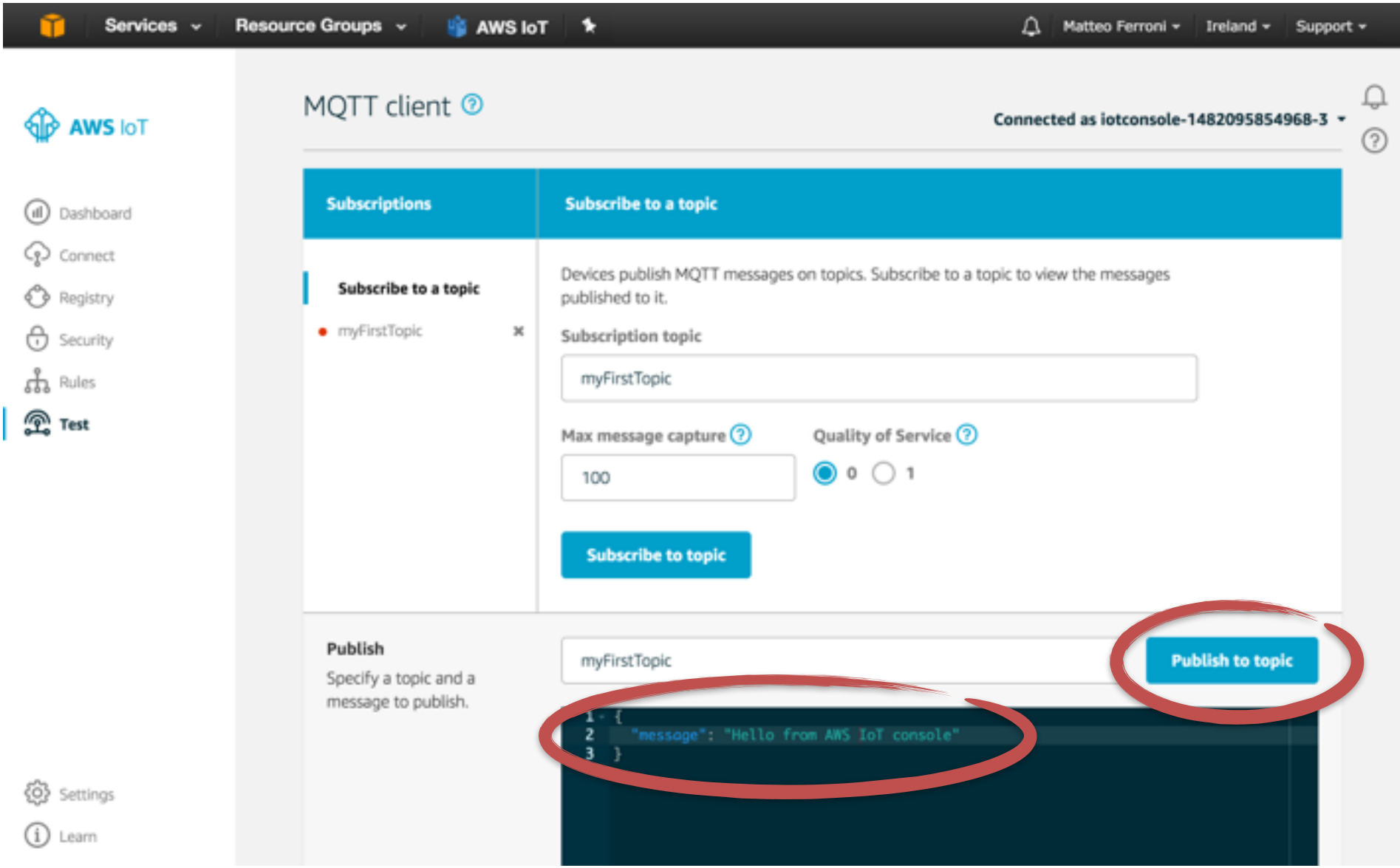


Connect, Publish and Subscribe

- **…Message received!**

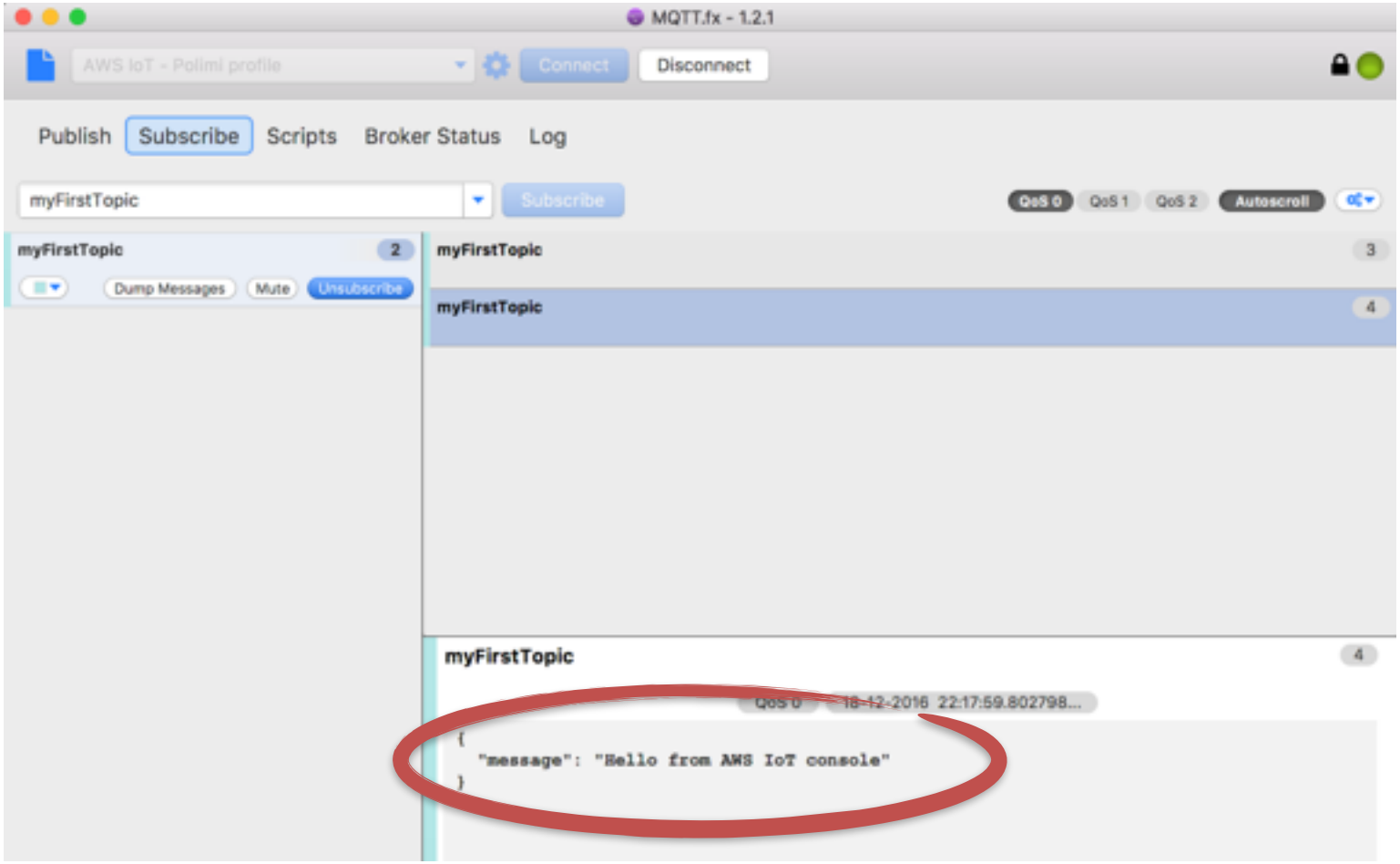


Publish and subscribe using AWS Dashboard



From the dashboard to MQTT.fx

- …Message received!



The Python SDK - Install

https://github.com/aws/aws-iot-device-sdk-python

Install from pip

```
pip install AWSIoTPythonSDK
```

Build from source

```
git clone https://github.com/aws/aws-iot-device-sdk-python.git
cd aws-iot-device-sdk-python
python setup.py install
```

Download the zip file

The SDK zip file is available here. Unzip the package and install the SDK like this:

```
python setup.py install
```

The Python SDK - Connection types

https://github.com/aws/aws-iot-device-sdk-python

Credentials

The SDK supports two types of credentials that correspond to the two connection types:

- X.509 certificate

  For the certificate-based mutual authentication connection type. Download the AWS IoT root CA. Use the AWS IoT console to create and download the certificate and private key. You must specify the location of these files when you initialize the client.

- IAM credentials

  For the Websocket with Signature Version 4 authentication type. You will need IAM credentials: an access key ID, a secret access key, and an optional session token. You must also download the AWS IoT root CA. You can specify the IAM credentails by:

Your first MQTT message using the Python SDK

```
# Import SDK packages
import ssl
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

# For certificate based connection
myMQTTClient = AWSIoTMQTTClient("TestDevice1", 4, False, True);

# Configurations
# For TLS mutual authentication
myMQTTClient.configureEndpoint("a1y3khtyn0ojiv.iot.eu-west-1.amazonaws.com", 8883)

myMQTTClient.configureCredentials("rootCA.pem", "privateKey.pem", "cert.pem")

myMQTTClient.configureOfflinePublishQueueing(-1)  # Infinite offline Publish queueing
myMQTTClient.configureDrainingFrequency(2)  # Draining: 2 Hz
myMQTTClient.configureConnectDisconnectTimeout(10)  # 10 sec
myMQTTClient.configureMQTTOperationTimeout(5)  # 5 sec

myMQTTClient.connect()
myMQTTClient.publish("myFirstTopic", "hello NECST!", 0)

myMQTTClient.unsubscribe("myFirstTopic")
myMQTTClient.disconnect()
```

The Polimi firewall…

Problema

- *sulle reti WiFi Polimi la **porta 8883 è chiusa**!*

Soluzione per connessioni da laptop

- usa una VPN

  (e.g., TunnelBear: https://www.tunnelbear.com/)

Soluzione per connessioni da scheda

- Usiamo **MQTT over WebSocket**

  http://docs.aws.amazon.com/iot/latest/developerguide/protocols.html#mqtt-ws

MQTT over WebSocket - Credentials

https://github.com/aws/aws-iot-device-sdk-python

Credentials

The SDK supports two types of credentials that correspond to the two connection types:

- X.509 certificate

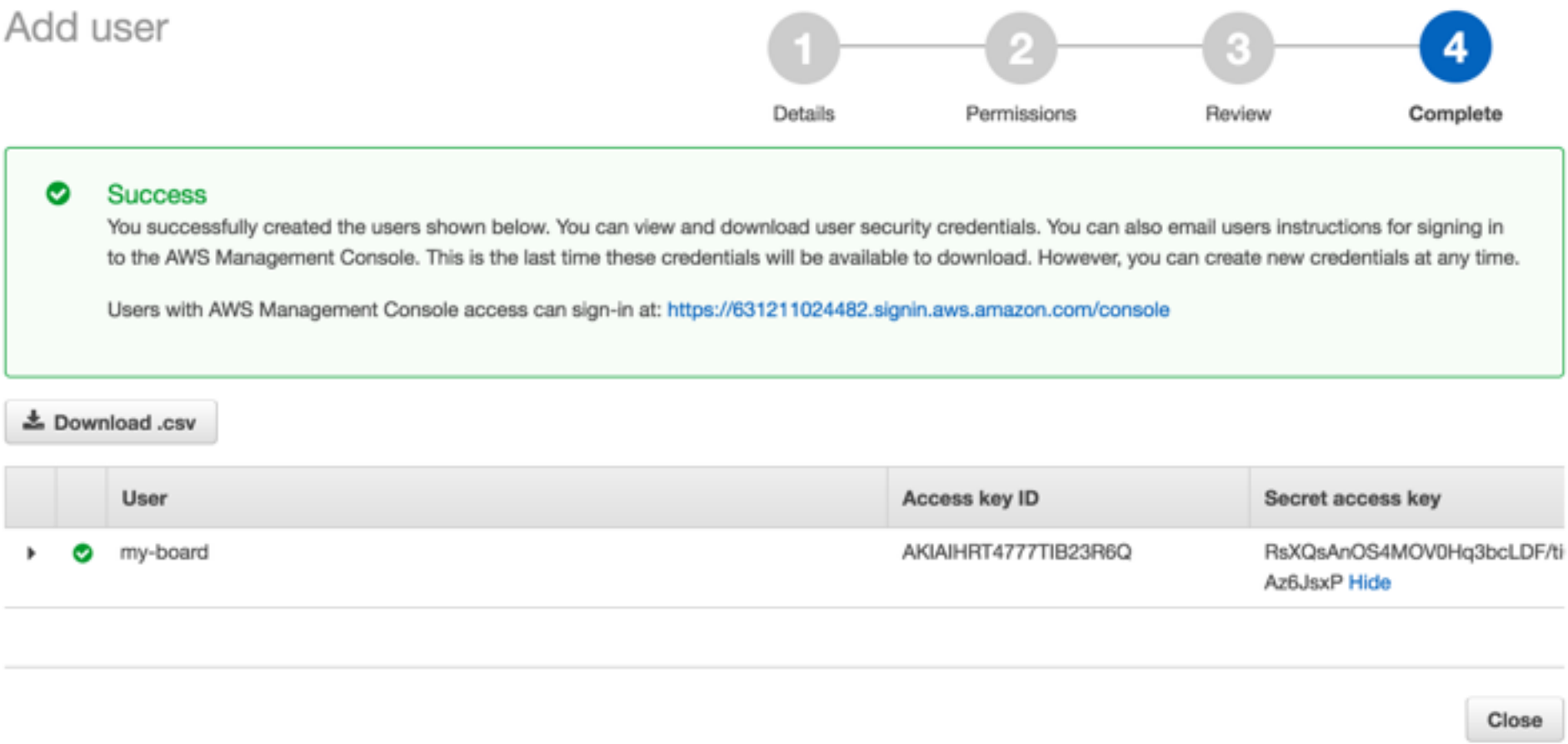
  For the certificate-based mutual authentication connection type. Download the AWS IoT root CA. Use the AWS IoT console to create and download the certificate and private key. You must specify the location of these files when you initialize the client.

- IAM credentials

  For the Websocket with Signature Version 4 authentication type. You will need IAM credentials: an access key ID, a secret access key, and an optional session token. You must also download the AWS IoT root CA. You can specify the IAM credentails by:

MQTT over WebSocket - Credentials

- **Creiamo un nuovo utente “my-board”**



Your second MQTT message, over WebSocket

```python
# Import SDK packages
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

# For Websocket connection
myMQTTClient = AWSIoTMQTTClient("my-board", useWebsocket=True)

# AWS IoT MQTT Client
myMQTTClient.configureIAMCredentials("ID", "SECRET")

# Configurations
# For Websocket
myMQTTClient.configureEndpoint("a1c2qwq6b794bg.iot.eu-west-1.amazonaws.com", 443)

# For Websocket, we only need to configure the root CA
myMQTTClient.configureCredentials("rootCA.pem")

myMQTTClient.configureOfflinePublishQueueing(-1)  # Infinite offline Publish queueing
myMQTTClient.configureDrainingFrequency(2)  # Draining: 2 Hz
myMQTTClient.configureConnectDisconnectTimeout(10)  # 10 sec
myMQTTClient.configureMQTTOperationTimeout(5)  # 5 sec

myMQTTClient.connect()
myMQTTClient.publish("myFirstTopic", "hello from my-board, I'm using a WebSocket!", 0)
# myMQTTClient.subscribe("test/topic", 1, customCallback)
myMQTTClient.unsubscribe("myFirstTopic")
myMQTTClient.disconnect()
```