Experiment No._06

Title: MQTT based communication between Client and Server

Batch: B2 **Roll No.: 16010421059** **Experiment No.:06**

Aim: MQTT based communication between Client and Server

Resources needed: Internet, R-Pi and Desktop System

Theory:

MQTT (MQ Telemetry Transport):

MQTT (MQ Telemetry Transport) is a lightweight messaging protocol that provides resourceconstrained network clients with a simple way to distribute telemetry information. The protocol, which uses a publish/subscribe communication pattern, is used for machine-to-machine (M2M) communication and plays an important role in the internet of things (IoT).

Activity:

MQTT based communication between Client and Server:

Step 1: Go to cloudmqtt.com.

Step 2: Signup on the website

Step 3: Create a new Instance.

● Enter Name, Select Plan and Data center

Step 4:Now go to your Instance and you will get Details about your instances

Step 5: Create a phpMQTT.php file as below

<?php class phpMQTT { private $socket; /* holds the socket */ private

$msgid = 1; /* counter for message id */ public $keepalive = 10; /*

default keepalive timmer */ public $timesinceping; /* host unix time,

used to detect disconects */ public $topics = array(); /* used to store

currently subscribed topics */ public $debug = false; /* should output

debug messages */

```
public $address; /* broker address */ public
$port; /* broker port */ public $clientid; /*
client id sent to brocker */ public $will; /*
stores the will of the client */ private
$username; /* stores username */ private
$password; /* stores password */ function
construct($address, $port, $clientid){
$this->broker($address, $port, $clientid);
}
/* sets the broker details */ function
broker($address, $port, $clientid){
$this->address = $address;
$this->port = $port;
$this->clientid = $clientid;
}
function connect_auto($clean = true, $will = NULL, $username = NULL, $password = NULL){
while($this->connect($clean, $will, $username, $password)==false){ sleep(10);
}
return true;
}
/* connects to the broker inputs: $clean: should the client send a clean session flag
*/ function connect($clean = true, $will = NULL, $username = NULL, $password =
NULL){ if($will) $this->will = $will;
```

```
if($username) $this->username = $username; if($password)
$this->password = $password;
$address = gethostbyname($this->address);
$this->socket = fsockopen($address, $this->port, $errno, $errstr, 60);
if (!$this->socket ) { if($this->debug) error_log("fsockopen() $errno,
$errstr \n"); return false;
}
stream_set_timeout($this->socket, 5);
stream_set_blocking($this->socket, 0);
$i = 0;
$buffer = "";
$buffer .= chr(0x00); $i++;
$buffer .= chr(0x06); $i++;
$buffer .= chr(0x4d); $i++;
$buffer .= chr(0x51); $i++;
$buffer .= chr(0x49); $i++;
$buffer .= chr(0x73); $i++;
$buffer .= chr(0x64); $i++;
$buffer .= chr(0x70); $i++; $buffer
.= chr(0x03); $i++;
//No Will $var = 0;
if($clean) $var+=2;
//Add will info to header
```

```
if($this->will != NULL){
$var += 4; // Set will flag
$var += ($this->will['qos'] << 3); //Set will qos
if($this->will['retain']) $var += 32; //Set will retain
}
if($this->username != NULL) $var += 128; //Add username to header if($this->password
!= NULL) $var += 64; //Add password to header
$buffer .= chr($var); $i++;
//Keep alive
$buffer .= chr($this->keepalive >> 8); $i++;
$buffer .= chr($this->keepalive & 0xff); $i++;
$buffer .= $this->strwritestring($this->clientid,$i);
//Adding will to payload if($this->will
!= NULL){
$buffer .= $this->strwritestring($this->will['topic'],$i);
$buffer .= $this->strwritestring($this->will['content'],$i);
}
if($this->username) $buffer .= $this->strwritestring($this->username,$i); if($this->password)
$buffer .= $this->strwritestring($this->password,$i);
$head = " ";
$head{0} = chr(0x10); $head{1}
= chr($i); fwrite($this->socket,
$head, 2); fwrite($this->socket,
```

```
$buffer); $string =
$this->read(4);
if(ord($string{0})>>4 == 2 &&
$string{3} == chr(0)){
if($this->debug) echo
"Connected to Broker\n";
}else{ error_log(sprintf("Connection failed! (Error: 0x%02x
0x%02x)\n", ord($string{0}),ord($string{3}))); return false;
}
$this->timesinceping = time(); return
true;
}
/* read: reads in so many bytes */ function
read($int = 8192, $nb = false){
// print_r(socket_get_status($this->socket));
$string=""; $togo = $int; if($nb){
return fread($this->socket, $togo);
}
while (!feof($this->socket) && $togo>0) {
$fread = fread($this->socket, $togo);
$string .= $fread;
$togo = $int - strlen($string);
}
```

```
return $string;
}
/* subscribe: subscribes to topics */ function
subscribe($topics, $qos = 0){
$i = 0;
$buffer = "";
$id = $this->msgid;
$buffer .= chr($id >> 8); $i++; $buffer
.= chr($id % 256); $i++;
foreach($topics as $key => $topic){
$buffer .= $this->strwritestring($key,$i);
$buffer .= chr($topic["qos"]); $i++;
$this->topics[$key] = $topic;
}
$cmd = 0x80;
//$qos
$cmd += ($qos << 1);
$head = chr($cmd); $head .=
chr($i); fwrite($this->socket,
$head, 2); fwrite($this->socket,
$buffer, $i); $string =
$this->read(2);
$bytes = ord(substr($string,1,1));
```

```
$string = $this->read($bytes);

}

/* ping: sends a keep alive ping */ function

ping(){

$head = " ";

$head = chr(0xc0); $head .=

chr(0x00); fwrite($this->socket,

$head, 2); if($this->debug) echo

"ping sent\n";

}

/* disconnect: sends a proper disconect cmd */

function disconnect(){ $head = " ";

$head{0} = chr(0xe0); $head{1}

= chr(0x00);

fwrite($this->socket, $head, 2);

}

/* close: sends a proper disconect, then closes the socket */

function close(){ $this->disconnect(); fclose($this->socket);

}

/* publish: publishes $content on a $topic */ function

publish($topic, $content, $qos = 0, $retain = 0){

$i = 0;

$buffer = "";
```

```
$buffer .= $this->strwritestring($topic,$i); //$buffer
.= $this->strwritestring($content,$i); if($qos){

$id = $this->msgid++;
$buffer .= chr($id >> 8); $i++;
$buffer .= chr($id % 256); $i++;
}
$buffer .= $content;
$i+=strlen($content);
$head = " "; $cmd = 0x30;
if($qos) $cmd += $qos <<
1; if($retain) $cmd += 1;
$head{0} = chr($cmd); $head .=
$this->setmsglength($i);
fwrite($this->socket, $head, strlen($head));
fwrite($this->socket, $buffer, $i);
}
/* message: processes a recieved topic */ function
message($msg){
$tlen = (ord($msg{0})<<8) + ord($msg{1});
$topic = substr($msg,2,$tlen);
$msg = substr($msg,($tlen+2)); $found
= 0; foreach($this->topics as
$key=>$top){ if(
```

```
preg_match("/^".str_replace("#",".*",
str_replace("+","[^\/]*",
str_replace("/","\/",

str_replace("$",'\$', $key)))))."$/",$topic) ){
if(is_callable($top['function'])){
call_user_func($top['function'],$topic,$msg);
$found = 1;
}
}
}
if($this->debug && !$found) echo "msg recieved but no match in
subscriptions\n";
}
/* proc: the processing loop for an "allways on" client set true when you are doing
other stuff in the loop good for watching something else at the same time */ function
proc( $loop = true){
if(1){
$sockets = array($this->socket);
$w = $e = NULL;
$cmd = 0;
//$byte = fgetc($this->socket); if(feof($this->socket)){
if($this->debug) echo "eof receive going to reconnect for good
measure\n"; fclose($this->socket);
```

```
$this->connect_auto(false);
if(count($this->topics))
$this->subscribe($this->topics);
}
$byte = $this->read(1, true);
if(!strlen($byte)){ if($loop)
{
usleep(100000);
}
}else{
$cmd = (int)(ord($byte)/16); if($this->debug)
echo "Recevid: $cmd\n";
$multiplier = 1;
$value = 0; do{
$digit = ord($this->read(1));
$value += ($digit & 127) * $multiplier;
$multiplier *= 128; }while (($digit & 128) !=
0); if($this->debug) echo "Fetching:
$value\n"; if($value)
$string = $this->read($value,"fetch");
if($cmd){ switch($cmd){ case 3:

$this->message($string); break;
}
```

```
$this->timesinceping = time();
}
}
if($this->timesinceping < (time() - $this->keepalive )){
if($this->debug) echo "not found something so ping\n";
$this->ping();
}
if($this->timesinceping<(time()-($this->keepalive*2))){ if($this->debug)
echo "not seen a package in a while, disconnecting\n";
fclose($this->socket); $this->connect_auto(false);
if(count($this->topics))
$this->subscribe($this->topics);
}
}
return 1;
}
/* getmsglength: */ function
getmsglength(&$msg, &$i){
$multiplier = 1;
$value = 0 ;
do{

$digit = ord($msg{$i});
$value += ($digit & 127) * $multiplier;
```

```
$multiplier *= 128;
$i++;
}while (($digit & 128) != 0); return
$value;
}
/* setmsglength: */ function
setmsglength($len){
$string = ""; do{
$digit = $len % 128;
$len = $len >> 7;
// if there are more digits to encode, set the top bit of this digit if
( $len > 0 )
$digit = ($digit | 0x80);
$string .= chr($digit);
}while ( $len > 0 ); return
$string;
}
/* strwritestring: writes a string to a buffer */ function
strwritestring($str, &$i){
$ret = " ";
$len = strlen($str);
$msb = $len >> 8;
$lsb = $len % 256;
```

```
$ret = chr($msb);
$ret .= chr($lsb);
$ret .= $str; $i
+= ($len+2);
return $ret;
}
function printstr($string){
$strlen = strlen($string);
for($j=0;$j<$strlen;$j++){ $num
= ord($string{$j}); if($num >
31)
$chr = $string{$j}; else $chr = " "; printf("%4d: %08b :
0x%02x : %s \n",$j,$num,$num,$chr);
}
}
}
?>
```

Step 6: Create publish.php file with code below.change your instance details accordingly

```
<?php require("phpMQTT.php");
$server = "m13.cloudmqtt.com"; // change if necessary
$port = 19902; // change if necessary
$username = ""; // set your username
$password = ""; // set your Password
```

```
$client_id = "phpMQTT-publisher";
$msg=rand(0,100);
$mqtt = new phpMQTT($server, $port, $client_id); if
($mqtt->connect(true, NULL, $username, $password)) {
$mqtt->publish("demo", $msg , 0);
$mqtt->close();
} else {
echo "Time out!\n";
}
?>
```

Step 7: Create database with data fields

Data_id Data

Step 8: Create subscribe.php with code below

```
<?php require("phpMQTT.php");
$server = "m16.cloudmqtt.com"; // change if necessary
$port = 12712; // change if necessary
$username = "wgsqxzrv"; // set your username
$password = "h_gYrgZ_C4CO"; // set your password
$clientid="ClientID".rand();
// make sure this is unique for connecting to server - you could use uniqid()
$mqtt = new phpMQTT($server, $port, $clientid);
if(!$mqtt->connect(true, NULL, $username, $password)) {
exit(1);
```

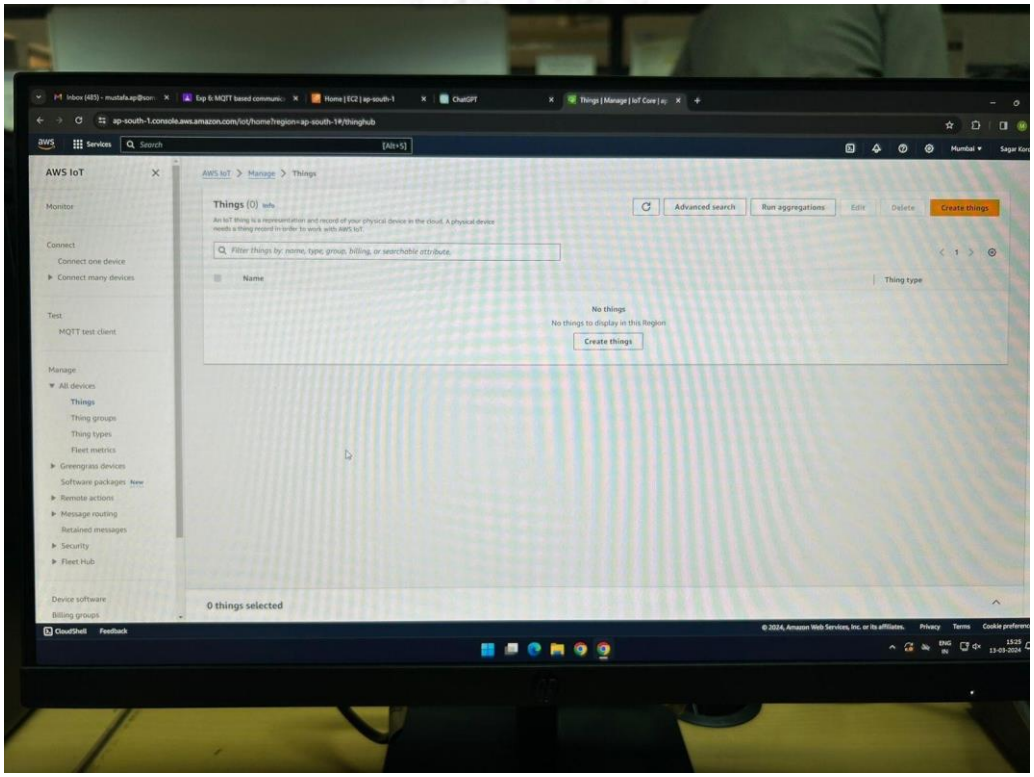
```
}
$topics['demo'] = array("qos" => 0, "function" => "procmsg");
$mqtt->subscribe($topics, 0); while($mqtt->proc()){
}
$mqtt->close(); function
procmsg($topic, $msg){ echo "Msg
Recieved: " . date("r") . "\n"; echo
"Topic: {$topic}\n\n"; echo
"\t$msg\n\n";
$conn=new mysqli("localhost","root","","mqttsample"); //connection to DB $query="select
max(data_id) as id from datasample"; //Change your table name(if applicable)
$result=mysqli_query($conn,$query);
$row=mysqli_fetch_array($result); if(is_null($row['id']))
{
$id=1;
}
else
{
$id=$row['id']+1;
}
$time=date('d-m-Y H:i:s');
$null=0;
$query=$conn->prepare("insert into datasample values (?,?)");//Change your table name(if
applicable)
```

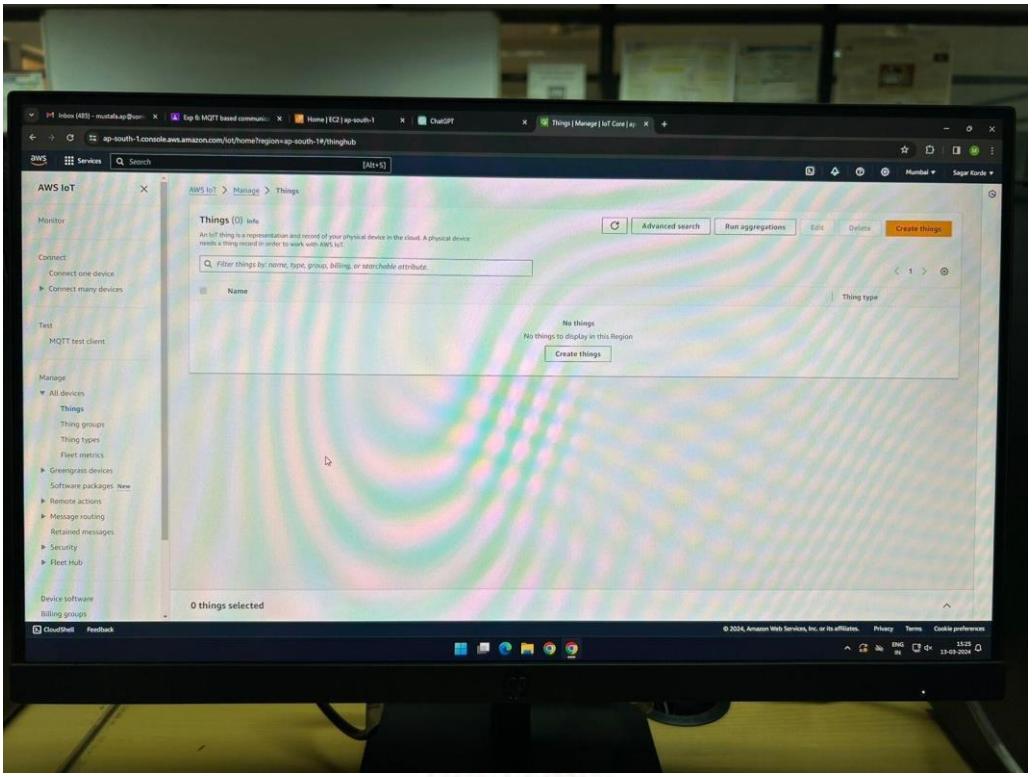
```
$query->bind_param('ii',$id,$msg);

$query->execute();

}
```

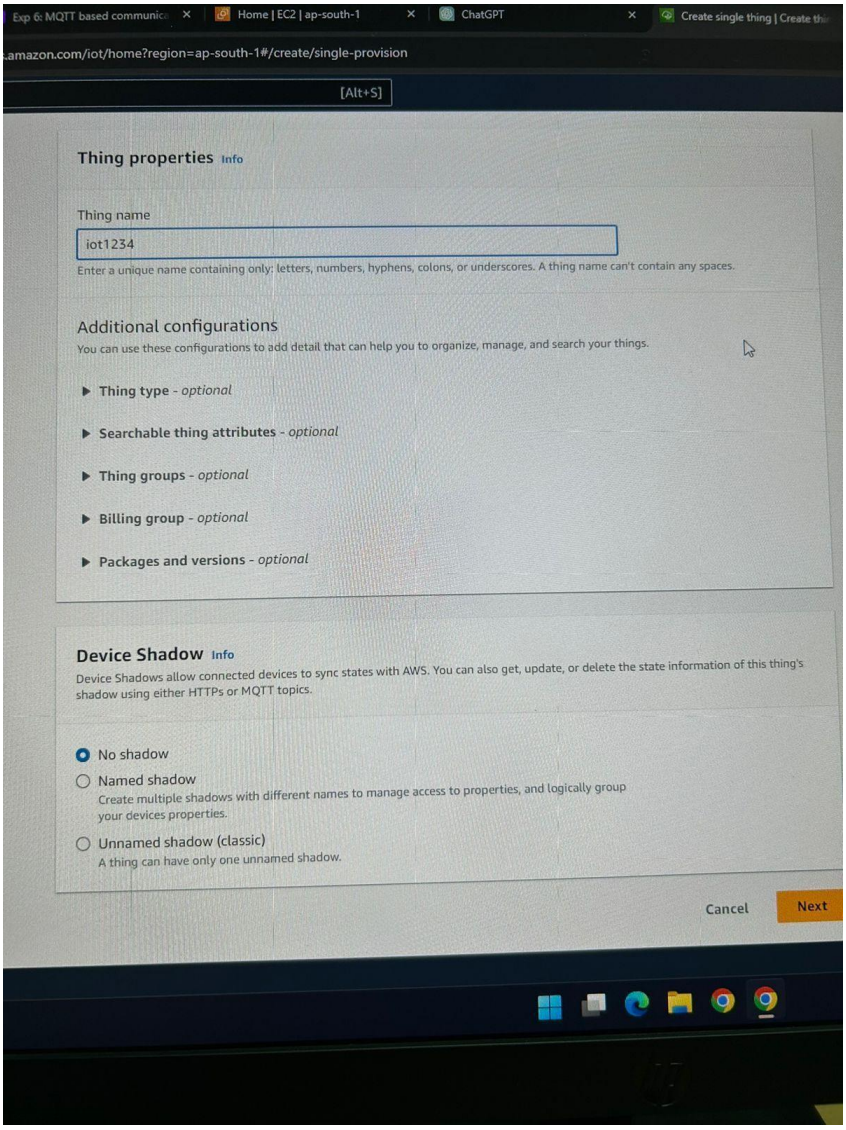
Step 9: Run File publish.php when you want to publish some contents

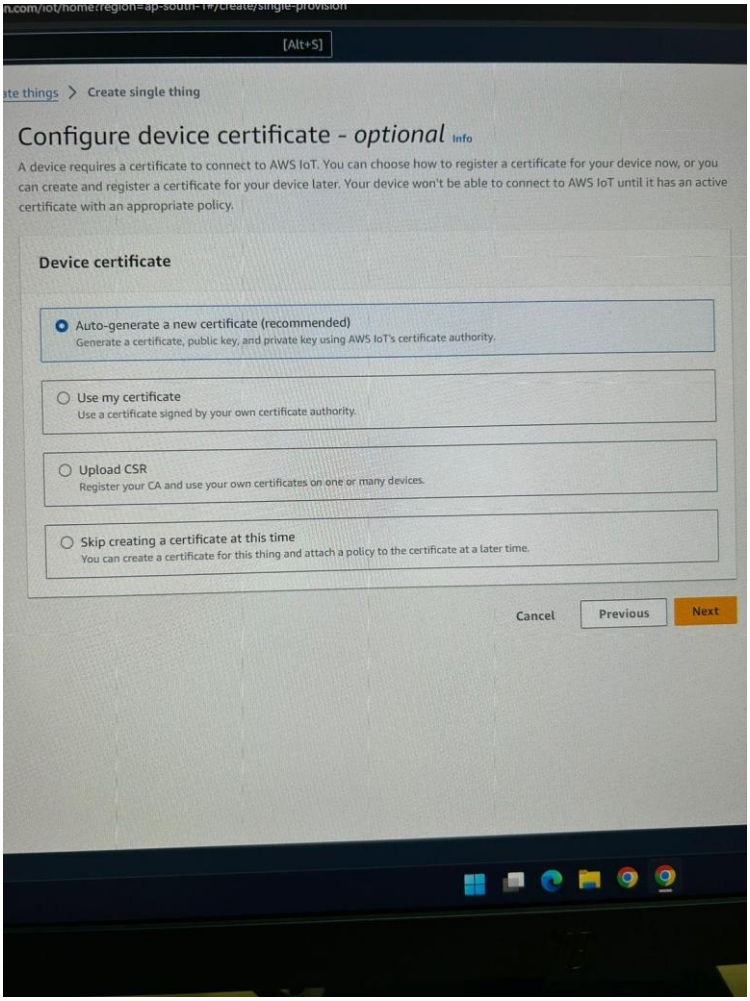
Step 10: Run Subscribe.php it will run continuously until browser/tab is closed and collect data published for topic subscribed in the database.

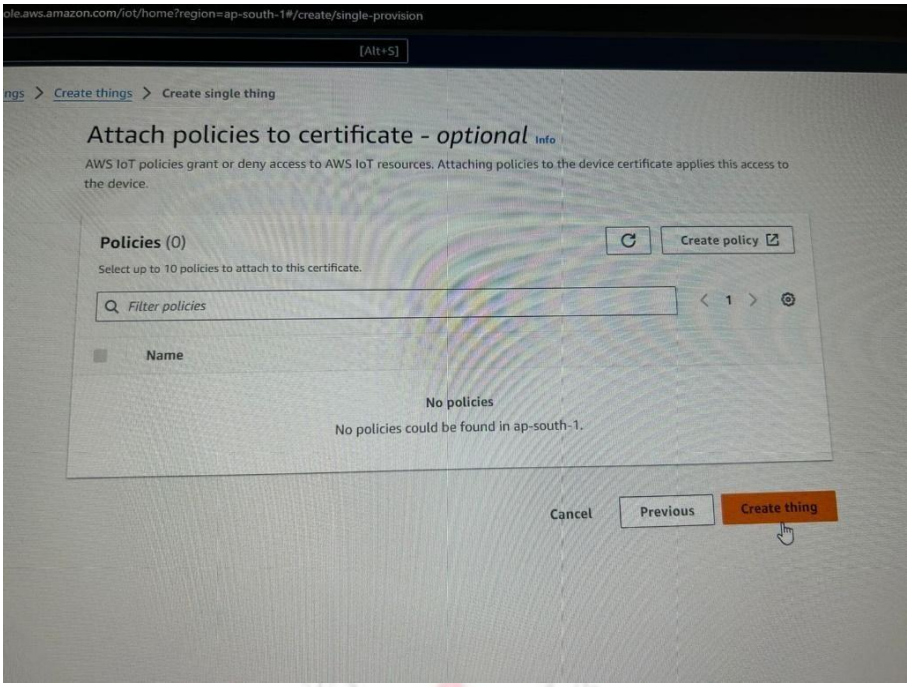
Results: (Program printout with output / Document printout as per the format)

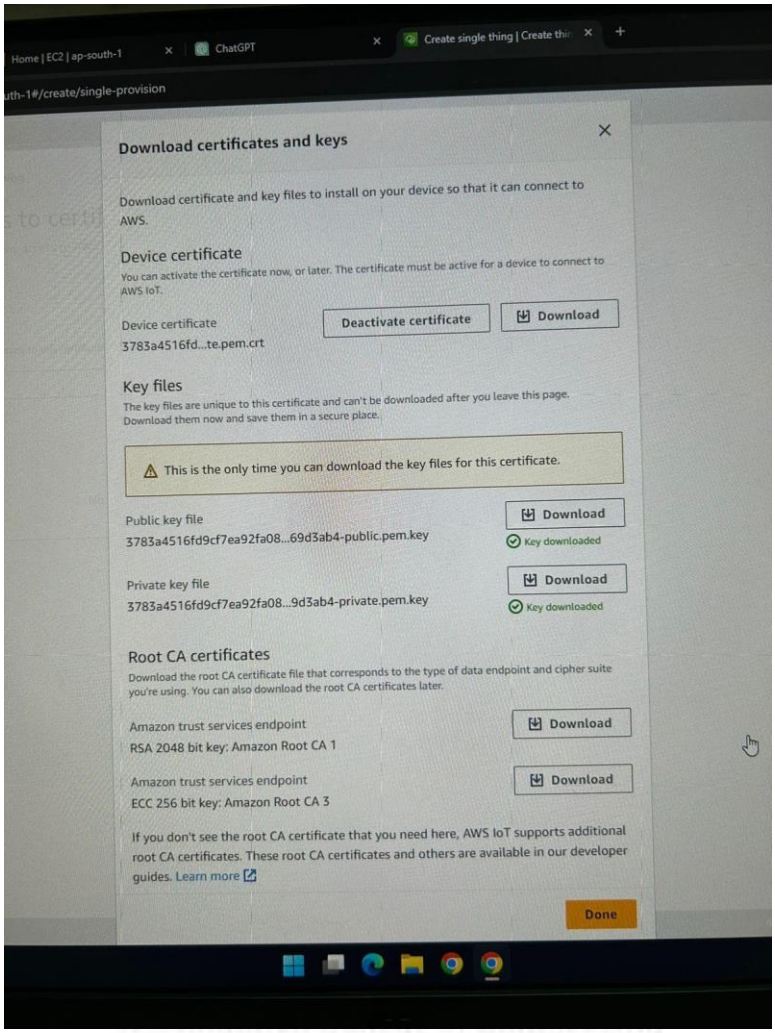


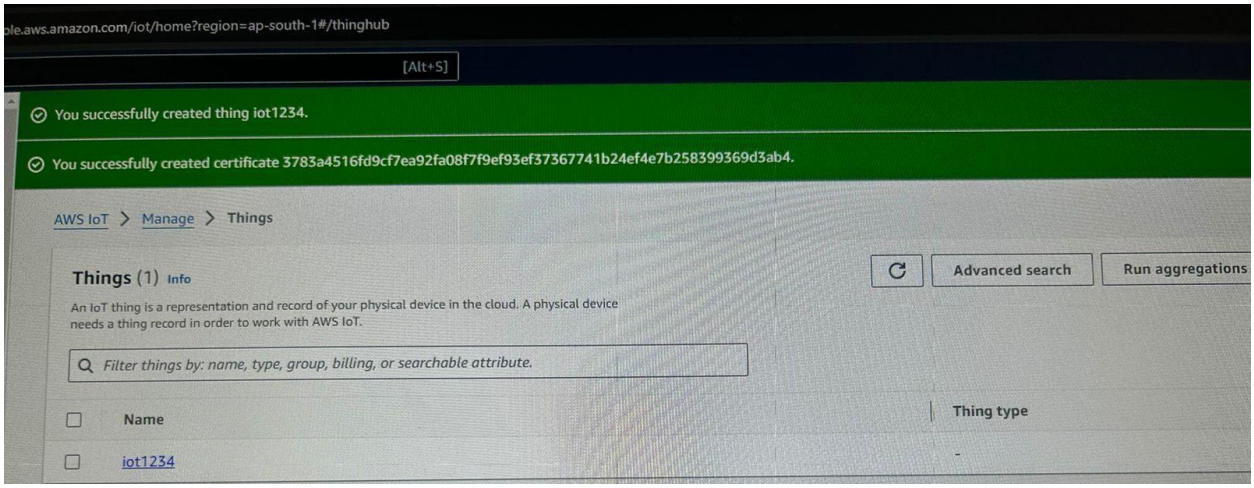


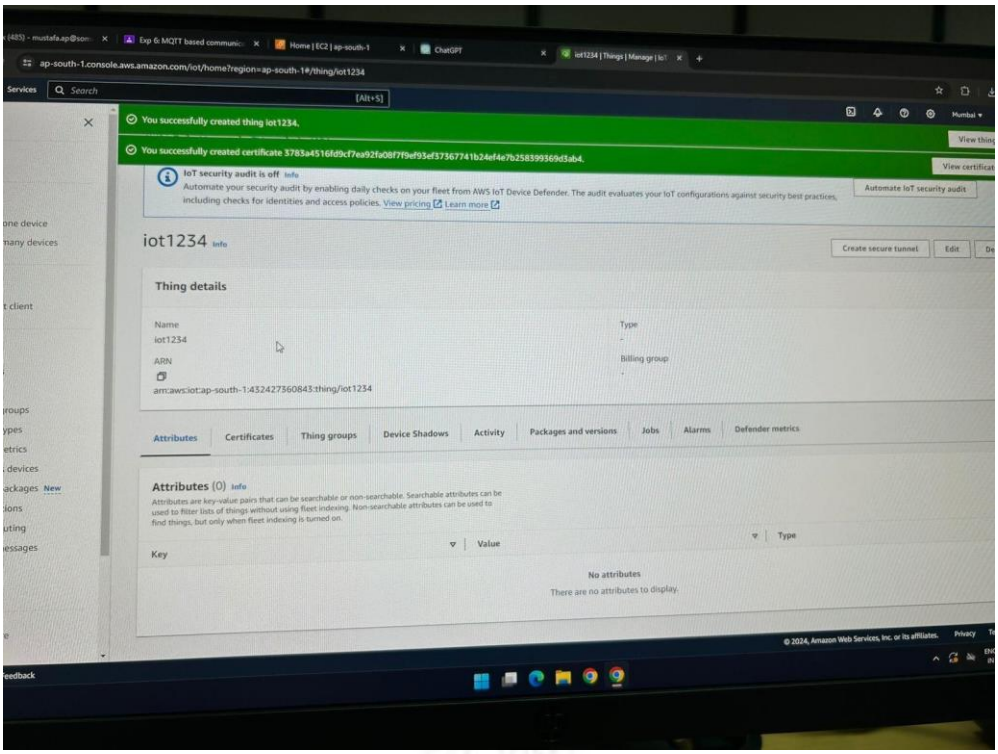

Exp 6: MQTT based communica
Home | EC2 | ap-south-1
ChatGPT
Create single thing | Create thi
amazon.com/iot/home?region=ap-south-1#/create/single-provision
[Alt+S]
Thing properties Info
Thing name
iot1234
Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.
Additional configurations
You can use these configurations to add detail that can help you to organize, manage, and search your things.
Thing type - optional
Searchable thing attributes - optional
Thing groups - optional
Billing group - optional
Packages and versions - optional
Device Shadow Info
Device Shadows allow connected devices to sync states with AWS. You can also get, update, or delete the state information of this thing's shadow using either HTTPs or MQTT topics.
No shadow
Named shadow
Create multiple shadows with different names to manage access to properties, and logically group your devices properties.
Unnamed shadow (classic)
A thing can have only one unnamed shadow.
Cancel
Next

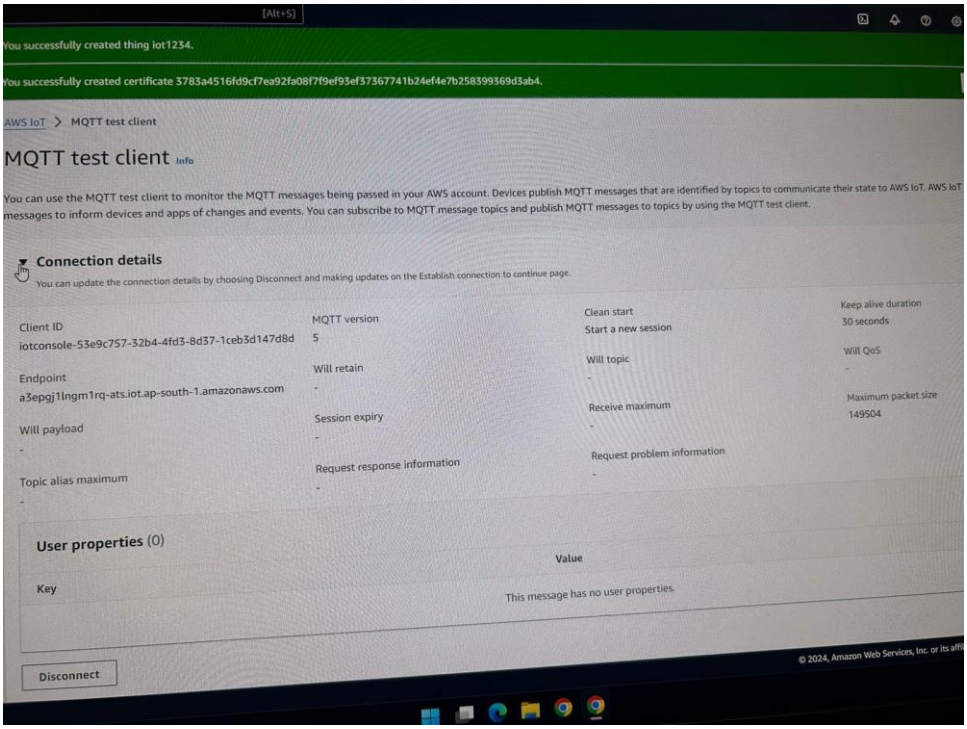

[Alt+S]
te things > Create single thing
Configure device certificate - optional Info
A device requires a certificate to connect to AWS IoT. You can choose how to register a certificate for your device now, or you can create and register a certificate for your device later. Your device won't be able to connect to AWS IoT until it has an active certificate with an appropriate policy.
Device certificate
Auto-generate a new certificate (recommended)
Generate a certificate, public key, and private key using AWS IoT's certificate authority.
Use my certificate
Use a certificate signed by your own certificate authority.
Upload CSR
Register your CA and use your own certificates on one or many devices.
Skip creating a certificate at this time
You can create a certificate for this thing and attach a policy to the certificate at a later time.
Cancel
Previous
Next

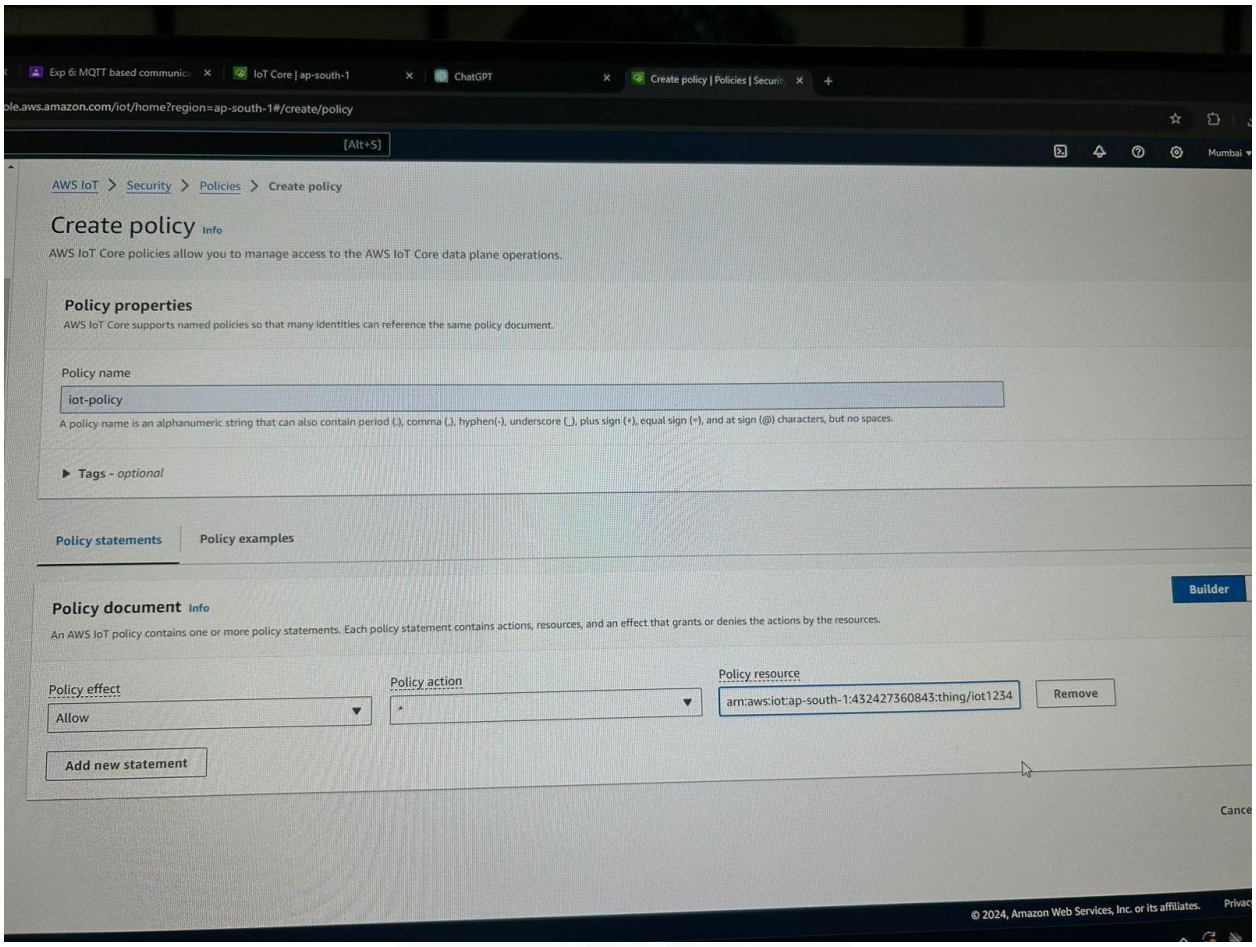


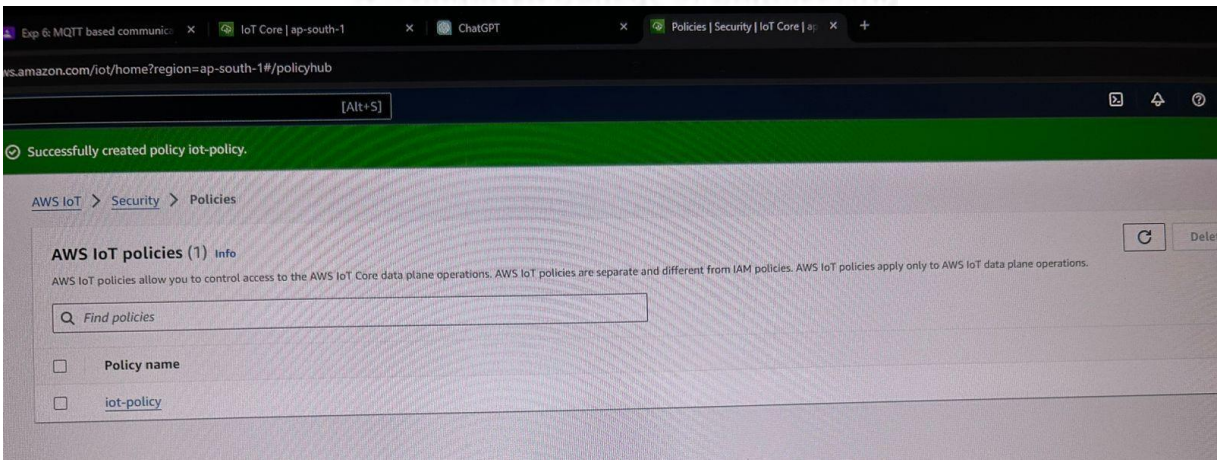

Home | EC2 | ap-south-1
ChatGPT
Create single thing | Create thi
uth-1#/create/single-provision
Download certificates and keys
Download certificate and key files to install on your device so that it can connect to AWS.
Device certificate
You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.
Device certificate
3783a4516fd...te.pem.crt
Deactivate certificate
Download
Key files
The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.
This is the only time you can download the key files for this certificate.
Public key file
3783a4516fd9cf7ea92fa08...69d3ab4-public.pem.key
Download
Key downloaded
Private key file
3783a4516fd9cf7ea92fa08...9d3ab4-private.pem.key
Download
Key downloaded
Root CA certificates
Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.
Amazon trust services endpoint
RSA 2048 bit key: Amazon Root CA 1
Download
Amazon trust services endpoint
ECC 256 bit key: Amazon Root CA 3
Download
If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. Learn more
Done

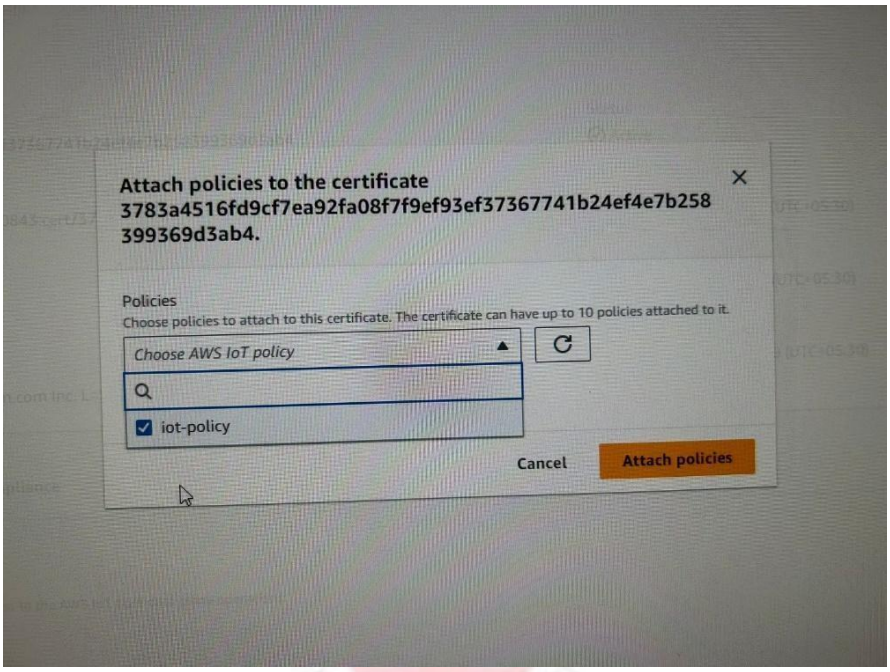

le.aws.amazon.com/iot/home?region=ap-south-1#/thinghub
[Alt+S]
You successfully created thing iot1234.
You successfully created certificate 3783a4516fd9cf7ea92fa08f7f9ef93ef37367741b24ef4e7b258399369d3ab4.
AWS IoT > Manage > Things
Things (1) Info
An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.
Advanced search
Run aggregations
Filter things by: name, type, group, billing, or searchable attribute.
Name
Thing type
iot1234
-

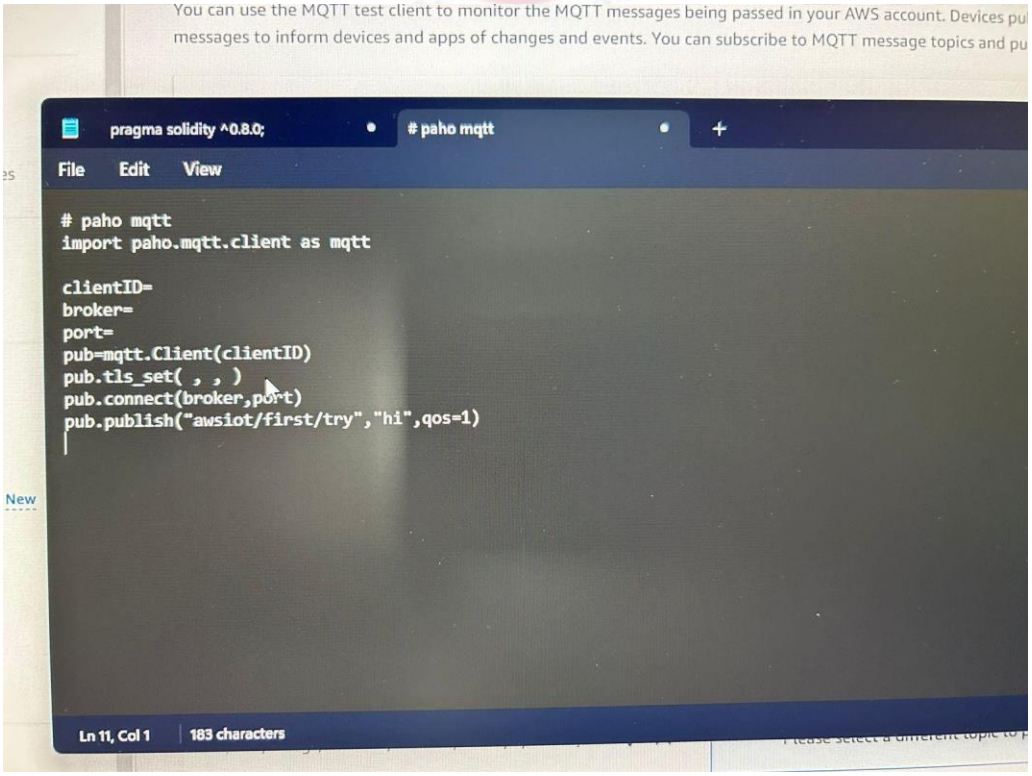

You successfully created thing iot1234.
You successfully created certificate 3783a4516fd9cf7ea92fa08f7f9ef93ef37367741b24ef4e7b258399369d3ab4.
IoT security audit is off
Automate your security audit by enabling daily checks on your fleet from AWS IoT Device Defender. The audit evaluates your IoT configurations against security best practices, including checks for identities and access policies.
iot1234
Thing details
Name
iot1234
ARN
arn:aws:iot:ap-south-1:432427360843:thing/iot1234
Type
Billing group
Attributes
Certificates
Thing groups
Device Shadows
Activity
Packages and versions
Jobs
Alarms
Defender metrics
Attributes (0)
Key
Value
Type
No attributes
There are no attributes to display.

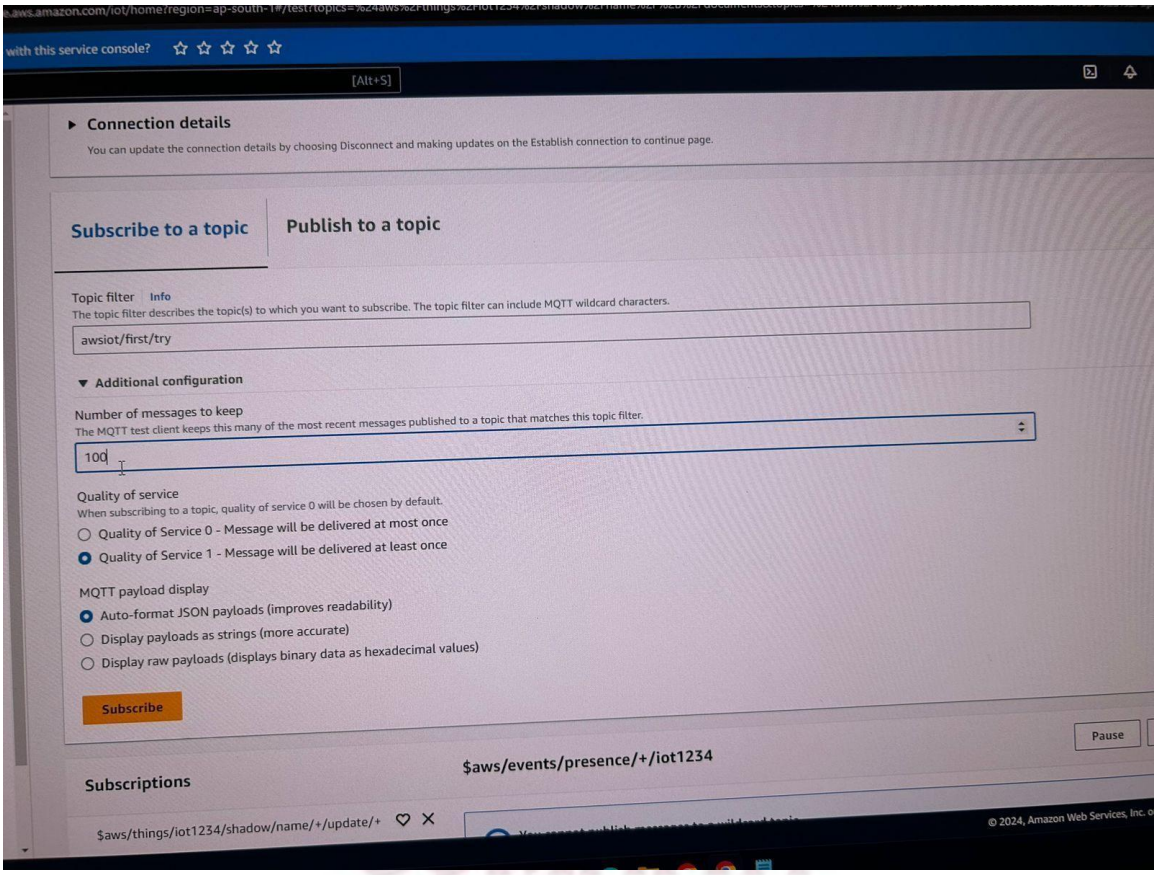

You successfully created thing iot1234.
You successfully created certificate 3783a4516fd9cf7ea92fa08f7f9ef93ef37367741b24ef4e7b258399369d3ab4.
AWS IoT > MQTT test client
MQTT test client
You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate their state to AWS IoT. AWS IoT messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.
Connection details
You can update the connection details by choosing Disconnect and making updates on the Establish connection to continue page.
Client ID
iotconsole-53e9c757-32b4-4fd3-8d37-1ceb3d147d8d
MQTT version
5
Clean start
Start a new session
Keep alive duration
30 seconds
Endpoint
a3epgj1lngm1rq-ats.iot.ap-south-1.amazonaws.com
Will retain
Will topic
Will QoS
Maximum packet size
149504
Will payload
Session expiry
Receive maximum
Topic alias maximum
Request response information
Request problem information
User properties (0)
Key
Value
This message has no user properties.
Disconnect

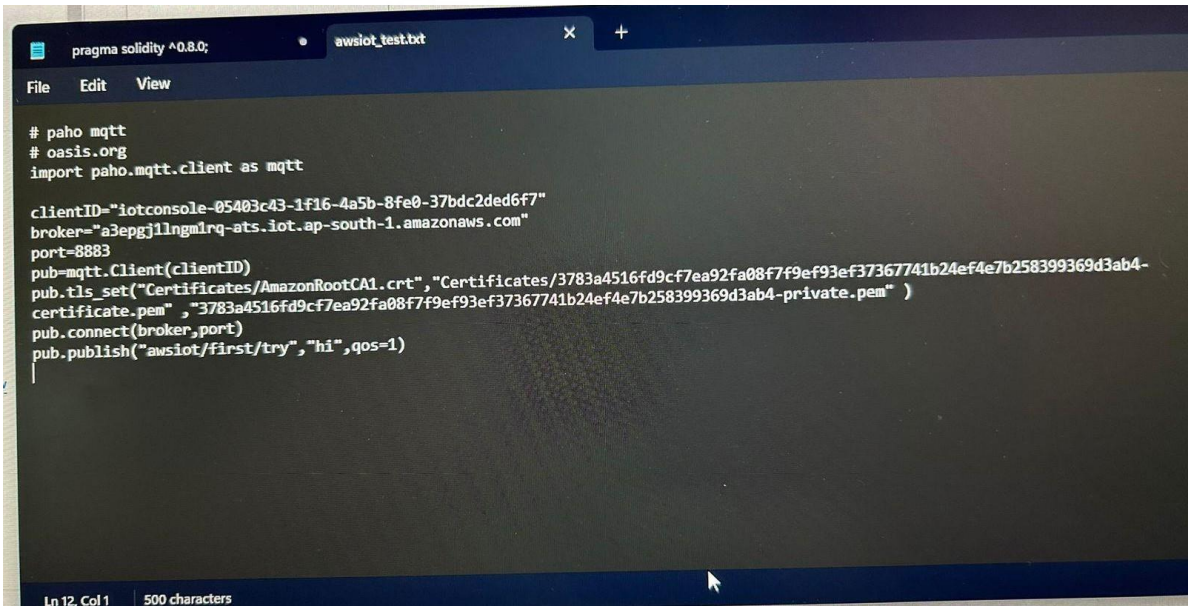

Exp 6: MQTT based communica
IoT Core | ap-south-1
ChatGPT
Create policy | Policies | Securit
ole.aws.amazon.com/iot/home?region=ap-south-1#/create/policy
[Alt+S]
Mumbai
AWS IoT > Security > Policies > Create policy
Create policy Info
AWS IoT Core policies allow you to manage access to the AWS IoT Core data plane operations.
Policy properties
AWS IoT Core supports named policies so that many identities can reference the same policy document.
Policy name
iot-policy
A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen(-), underscore (_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.
Tags - optional
Policy statements
Policy examples
Builder
Policy document Info
An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.
Policy effect
Allow
Policy action
*
Policy resource
arn:aws:iot:ap-south-1:432427360843:thing/iot1234
Remove
Add new statement
Cance
© 2024, Amazon Web Services, Inc. or its affiliates.
Privac

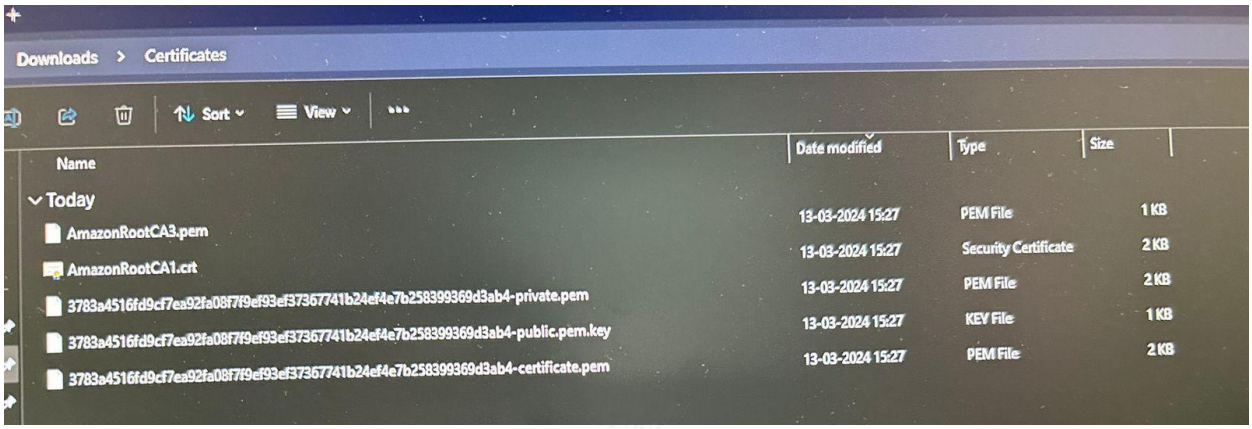

Exp 6: MQTT based communica
IoT Core | ap-south-1
ChatGPT
Policies | Security | IoT Core | ap
s.amazon.com/iot/home?region=ap-south-1#/policyhub
[Alt+S]
Successfully created policy iot-policy.
AWS IoT > Security > Policies
AWS IoT policies (1) Info
AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies. AWS IoT policies apply only to AWS IoT data plane operations.
Find policies
Policy name
iot-policy

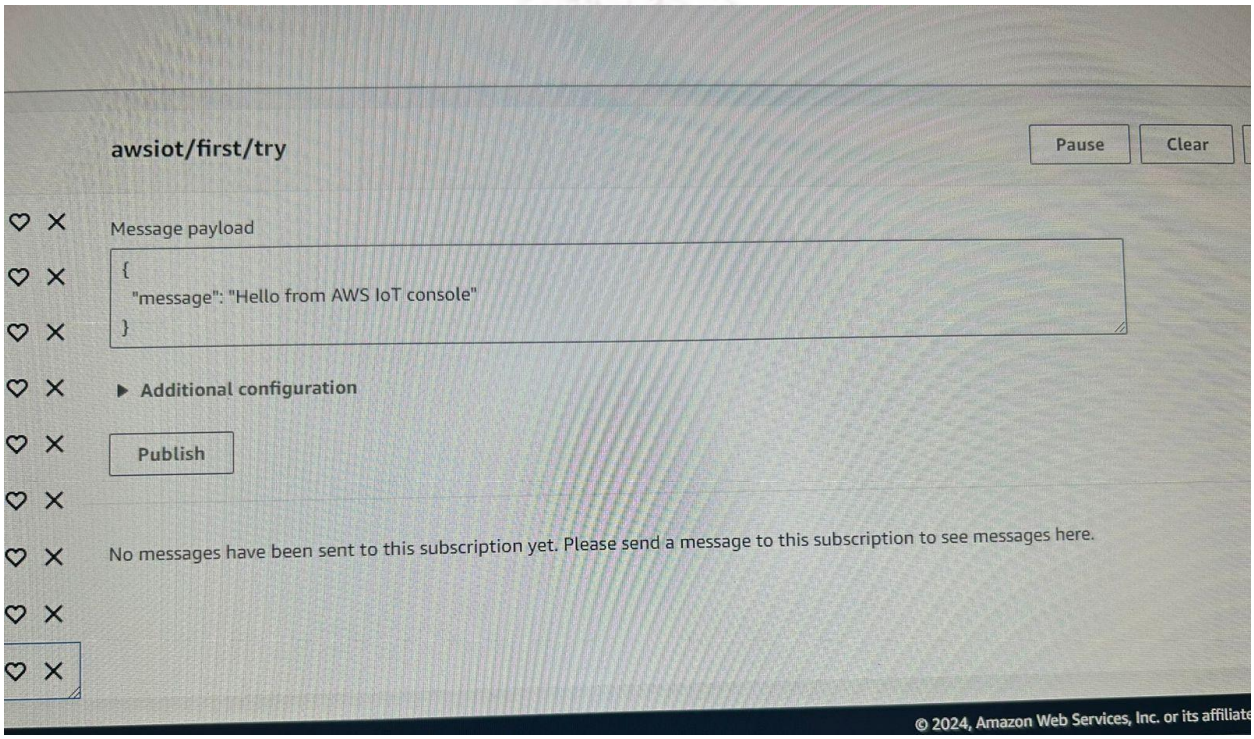

Attach policies to the certificate
3783a4516fd9cf7ea92fa08f7f9ef93ef37367741b24ef4e7b258
399369d3ab4.
Policies
Choose policies to attach to this certificate. The certificate can have up to 10 policies attached to it.
Choose AWS IoT policy
iot-policy
Cancel
Attach policies

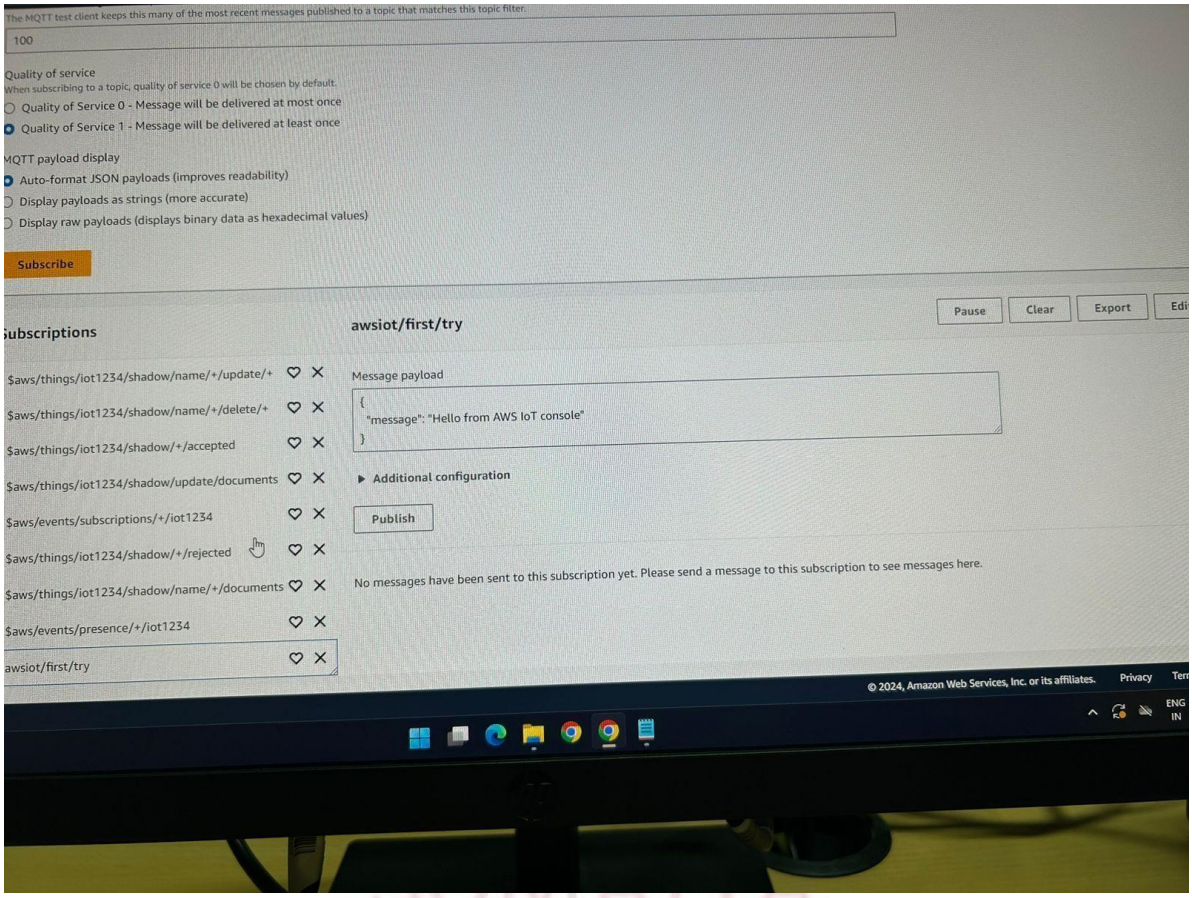

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices pu
messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and pu
pragma solidity ^0.8.0;
paho mqtt
File
Edit
View
# paho mqtt
import paho.mqtt.client as mqtt
clientID=
broker=
port=
pub=mqtt.Client(clientID)
pub.tls_set(, ,)
pub.connect(broker,port)
pub.publish("awsiot/first/try","hi",qos=1)
New
Ln 11, Col 1
183 characters

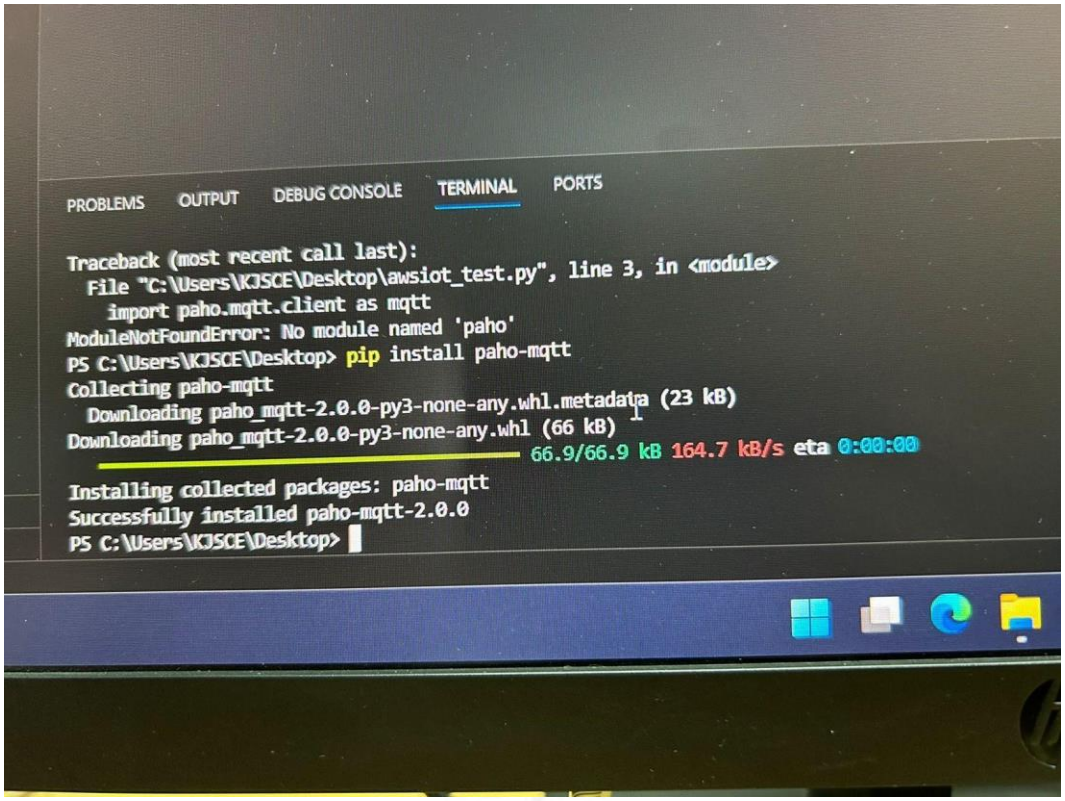

with this service console?
[Alt+S]
Connection details
You can update the connection details by choosing Disconnect and making updates on the Establish connection to continue page.
Subscribe to a topic
Publish to a topic
Topic filter Info
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.
awsiot/first/try
Additional configuration
Number of messages to keep
The MQTT test client keeps this many of the most recent messages published to a topic that matches this topic filter.
100
Quality of service
When subscribing to a topic, quality of service 0 will be chosen by default.
Quality of Service 0 - Message will be delivered at most once
Quality of Service 1 - Message will be delivered at least once
MQTT payload display
Auto-format JSON payloads (improves readability)
Display payloads as strings (more accurate)
Display raw payloads (displays binary data as hexadecimal values)
Subscribe
Pause
Subscriptions
$aws/events/presence/+/iot1234
$aws/things/iot1234/shadow/name/+/update/+
© 2024, Amazon Web Services, Inc.


pragma solidity ^0.8.0;
awsiot_test.txt
File
Edit
View
# paho mqtt
# oasis.org
import paho.mqtt.client as mqtt
clientID="iotconsole-05403c43-1f16-4a5b-8fe0-37bdc2ded6f7"
broker="a3epgj1lngm1rq-ats.iot.ap-south-1.amazonaws.com"
port=8883
pub=mqtt.Client(clientID)
pub.tls_set("Certificates/AmazonRootCA1.crt","Certificates/3783a4516fd9cf7ea92fa08f7f9ef93ef37367741b24ef4e7b258399369d3ab4-
certificate.pem" ,"3783a4516fd9cf7ea92fa08f7f9ef93ef37367741b24ef4e7b258399369d3ab4-private.pem")
pub.connect(broker,port)
pub.publish("awsiot/first/try","hi",qos=1)
Ln 12, Col 1
500 characters

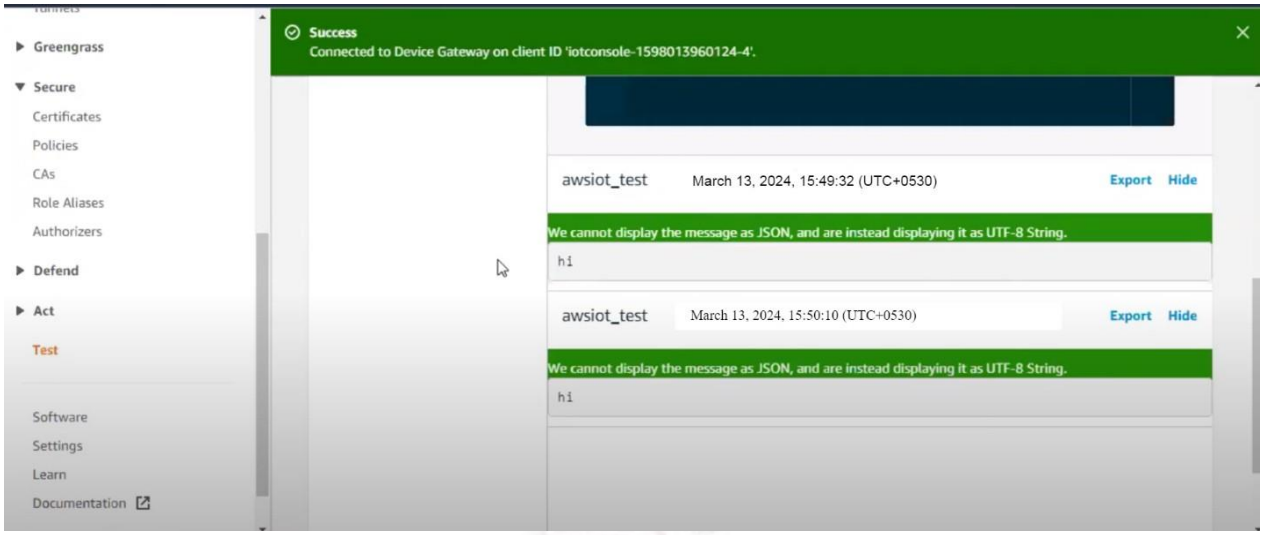

Downloads > Certificates
Sort
View
Name
Date modified
Type
Size
Today
AmazonRootCA3.pem
13-03-2024 15:27
PEM File
1 KB
AmazonRootCA1.crt
13-03-2024 15:27
Security Certificate
2 KB
3783a4516fd9cf7ea92fa08f7f9ef93ef37367741b24ef4e7b258399369d3ab4-private.pem
13-03-2024 15:27
PEM File
2 KB
3783a4516fd9cf7ea92fa08f7f9ef93ef37367741b24ef4e7b258399369d3ab4-public.pem.key
13-03-2024 15:27
KEY File
1 KB
3783a4516fd9cf7ea92fa08f7f9ef93ef37367741b24ef4e7b258399369d3ab4-certificate.pem
13-03-2024 15:27
PEM File
2 KB


awsiot/first/try
Pause
Clear
Message payload
{
"message": "Hello from AWS IoT console"
}
Additional configuration
Publish
No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here.
© 2024, Amazon Web Services, Inc. or its affiliate


The MQTT test client keeps this many of the most recent messages published to a topic that matches this topic filter.
100
Quality of service
When subscribing to a topic, quality of service 0 will be chosen by default.
Quality of Service 0 - Message will be delivered at most once
Quality of Service 1 - Message will be delivered at least once
MQTT payload display
Auto-format JSON payloads (improves readability)
Display payloads as strings (more accurate)
Display raw payloads (displays binary data as hexadecimal values)
Subscribe
Subscriptions
awsiot/first/try
Pause
Clear
Export
$aws/things/iot1234/shadow/name/+/update/+
$aws/things/iot1234/shadow/name/+/delete/+
$aws/things/iot1234/shadow/+/accepted
$aws/things/iot1234/shadow/update/documents
$aws/events/subscriptions/+/iot1234
$aws/things/iot1234/shadow/+/rejected
$aws/things/iot1234/shadow/name/+/documents
$aws/events/presence/+/iot1234
awsiot/first/try
Message payload
{
"message": "Hello from AWS IoT console"
}
Additional configuration
Publish
No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here.
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy




```
#paho mqtt #oasis.org
import paho.mqtt.client as mqtt
clientID = "'iotconsole-1598013960124-4"
broker ="al6lgataej 7hrq-ats.iot.ap-south-1.amazonaws.com"
port 8883
pub=mqtt.Client (clientID)
pub.tls_set ("certificates/AmalzonRootCA1.crt", certfile = "certificates/95bb06ece6_certificate.pem",
keyf pub.connect (broker, port, 45)
pub.publish("awsiot_test", "hi", qos=1)
```



Questions:

1. **List out all protocol which can use in IoT along with their application area.**

Ans:

Sure, here's a list of some common protocols used in IoT along with their application areas:

1. MQTT (Message Queuing Telemetry Transport):

- Application Area: Lightweight, efficient messaging protocol suitable for IoT devices with limited processing power and bandwidth. It's commonly used for telemetry, sensor data, and device-to-cloud communication in various IoT applications including home automation, industrial monitoring, and smart agriculture.

2. HTTP (Hypertext Transfer Protocol):

- Application Area: Widely used for communication between IoT devices and web servers. It's suitable for scenarios where devices need to interact with web-based services, such as accessing REST APIs or serving web pages. Commonly used in smart home applications, web-based dashboards for monitoring and control, and IoT applications involving cloud integration.

3. CoAP (Constrained Application Protocol):

- Application Area: Designed for resource-constrained devices and low-power networks such as

IoT deployments using constrained devices and wireless sensor networks (WSNs). It's used for machine-to-machine communication, enabling devices to exchange data with minimal overhead. Commonly used in smart cities, industrial automation, and environmental monitoring.

4. AMQP (Advanced Message Queuing Protocol):

- Application Area: A messaging protocol designed for reliable and efficient message queuing. It's suitable for scenarios requiring real-time data exchange, event-driven architectures, and high-throughput messaging. Commonly used in industrial automation, logistics, and supply chain management.

5. DDS (Data Distribution Service):

- Application Area: A publish-subscribe messaging protocol designed for real-time data distribution and communication between distributed systems. It's used in scenarios requiring high reliability, low latency, and deterministic communication, such as mission-critical systems, autonomous vehicles, and healthcare monitoring.

6. LoRaWAN (Long Range Wide Area Network):

- Application Area: A low-power, long-range wireless communication protocol used for connecting IoT devices over large geographical areas. It's suitable for applications like smart cities, smart agriculture, asset tracking, and environmental monitoring where long-range communication and low power consumption are essential.

7. Bluetooth/BLE (Bluetooth Low Energy):

- Application Area: Short-range wireless communication protocol used for connecting IoT devices in proximity to each other. It's commonly used in wearable devices, healthcare monitoring, smart home devices, and proximity-based marketing applications.

8. Zigbee:

- Application Area: Low-power, mesh networking protocol used for building home automation and industrial control systems. It's suitable for applications requiring low latency, high reliability, and scalability, such as smart lighting, HVAC control, and building automation.

9. Modbus:

- Application Area: A widely used protocol in industrial automation and SCADA systems for communication between electronic devices. It's suitable for monitoring and control applications in industrial environments, including manufacturing, energy management, and process automation.

These are just a few examples of protocols used in IoT, and the choice of protocol depends on factors such as device constraints, communication requirements, network topology, and application-specific needs.

Outcomes:

CO3: Realize design process of IoT applications and IoT challenges

---

Conclusion:

Thus , we successfully implemented the MQTT protocol communication between Client and Server using AWS IOT Core Service.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References: https://raw.githubusercontent.com/tbird20d/grabserial/master/grabserial

Books:

1. Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, David Boyle, "From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence", 1st Edition, Academic Press, 2014.

2. Vijay Madisetti and Arshdeep Bahga, "Internet of Things (A Hands-on-Approach)", 1stEdition, VPT, 2014.

3. Dr. Ovidiu Vermesan, Dr. Peter Friess, “Internet of Things - From Research and Innovation to Market Deployment”, River Publisher, 2014