Connecting IoT to cloud

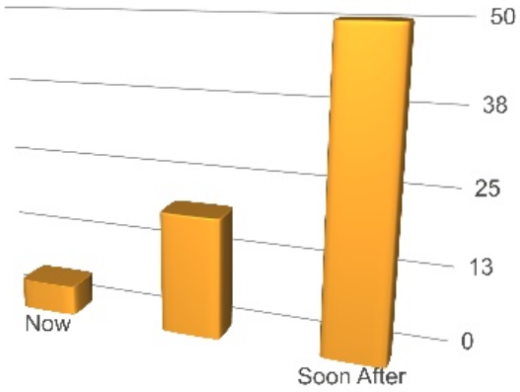
Darryl Ng

darryl.ng@nus.edu.sg

AWS IoT

Securely connect one or one billion devices to AWS,
So they can interact with applications and other devices

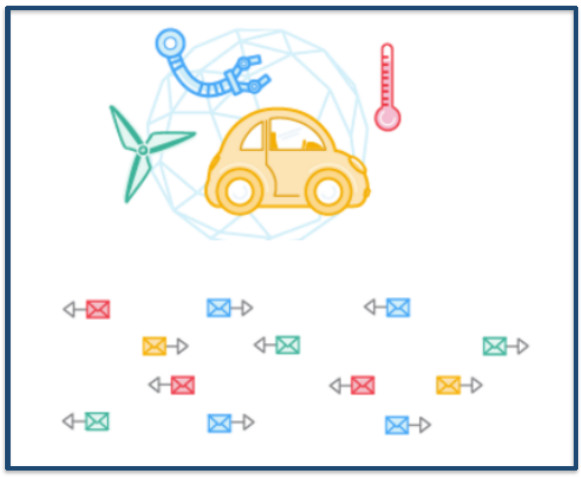
In the near future…
Things are becoming connected…



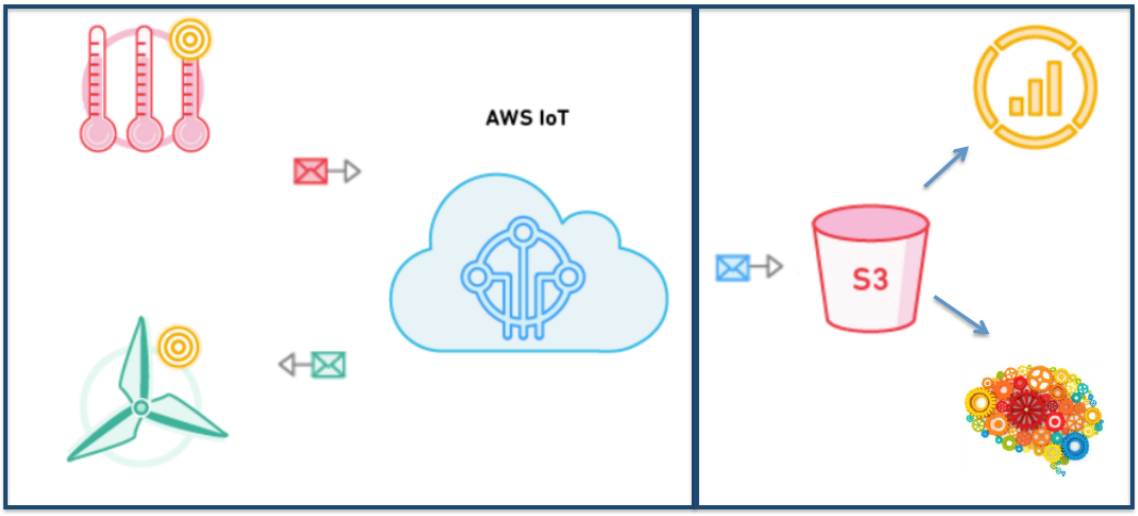
Today's hype	Tomorrow's normal
Today's Things	Tomorrow's insights
Today's technology	Tomorrow's solutions
Today's definition of "at scale"	Dwarfed by tomorrow's definition
Today's cost reduction	Tomorrow's growth engine
Today's confusion	Tomorrow's blue print
Today's connectivity issues	Solved on the edge

Internet-of-Things

- Things (Devices)
  - Many of them
    - Different Types
    - Isolated Systems
  - Data and Command
    - Sensing
    - Response
  - Challenge
    - United: Connected + Communication
    - Smart: Data Analytics + Strategy

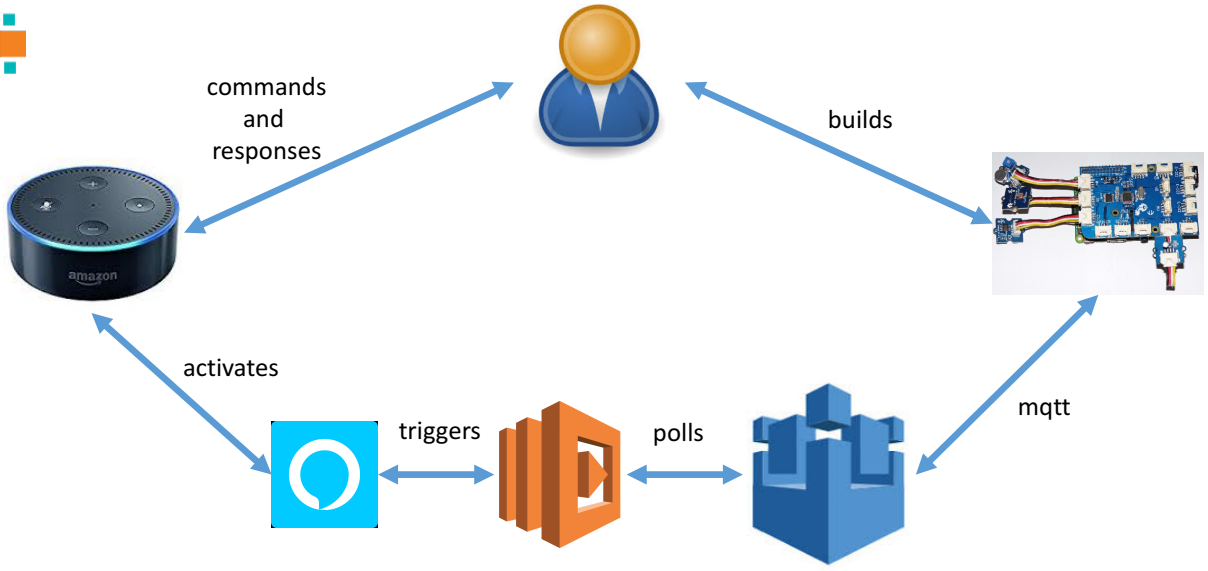


Solution: AWS IoT

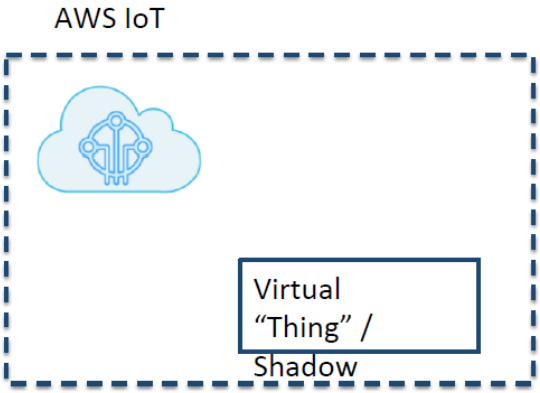


- United: Connect + Communicate
- Smart: Storage + Machine Learning

Tutorial: Alexa and AWS IoT

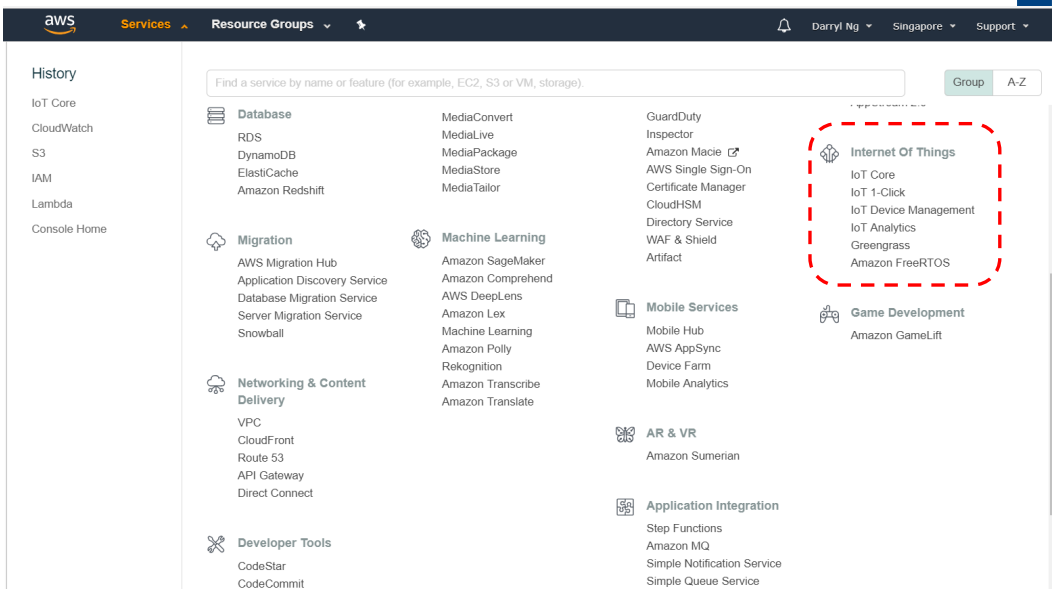


Step 1: Create a Virtual “Thing”



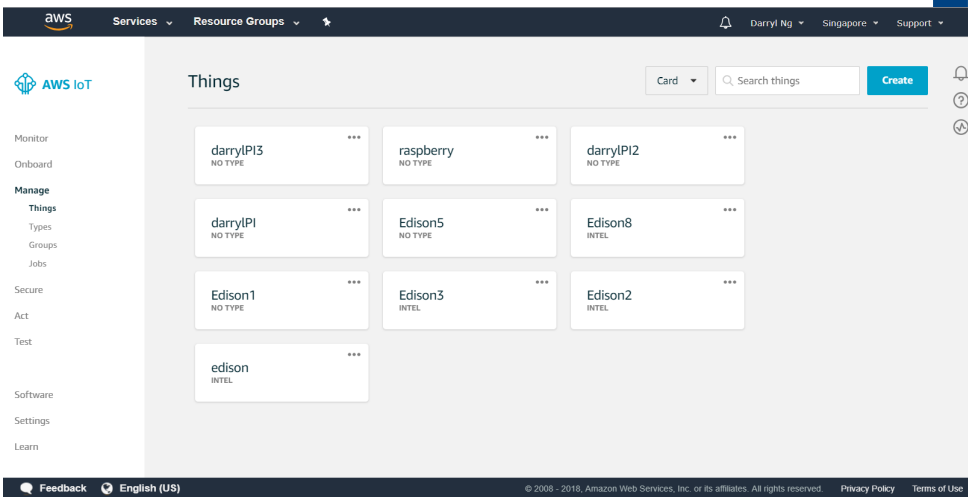
AWS Management Console

- Create an AWS account
- Sign In to AWS Management Console
  - https://aws.amazon.com/iot/



Create a thing

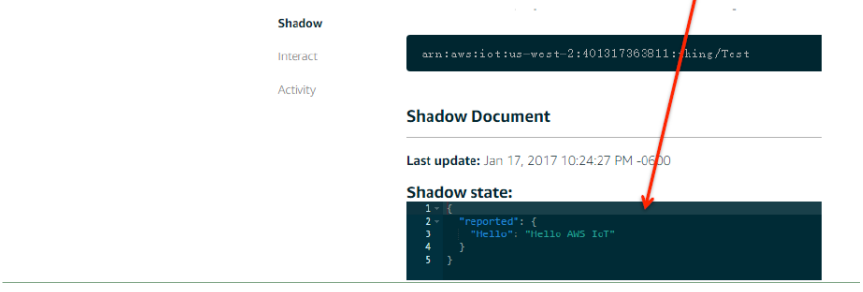
- AWS IoT menu
  - Registry
    - Things → Create
  - Give a name



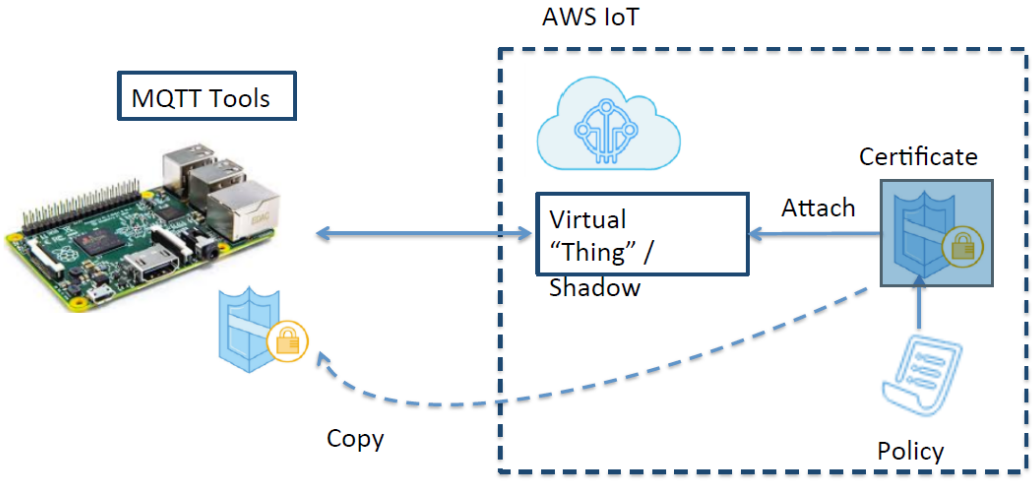
Basic Interaction: Publish

- Use Embedded MQTT Client to Test

- Use Thing Shadow

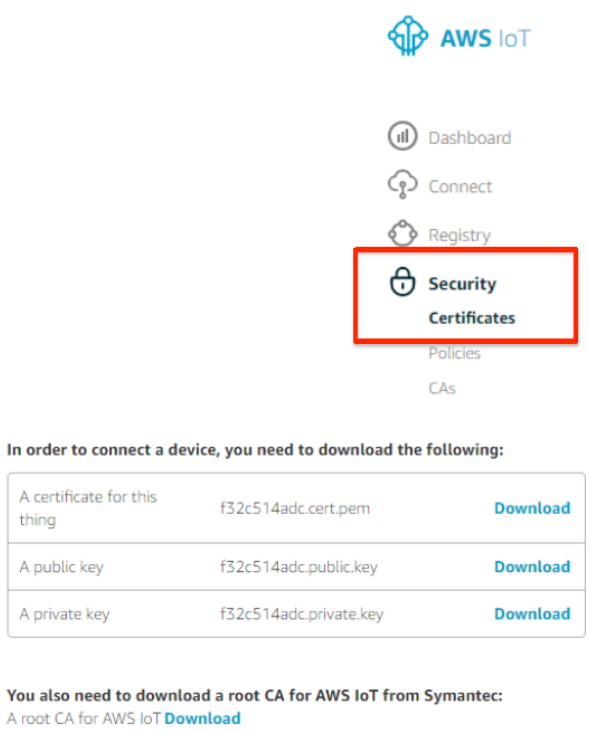


Step 2: Connect a Physical Device



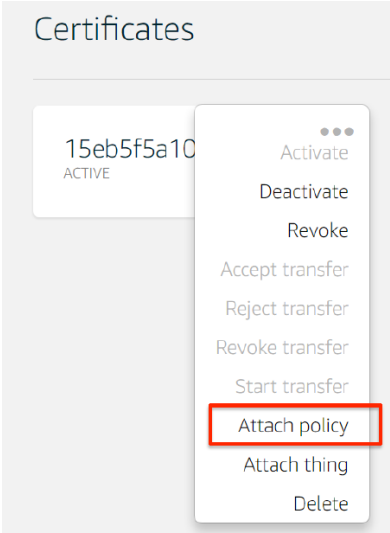
Certificates and Keys

- Create Certificates
  - Security → Certificates → Create
- Download Cert Files
  - Public & private key
  - Thing cert
  - Root CA for AWS



Create Policy and Attach to certificate

- Create Policy



- Attach Policy to Cert

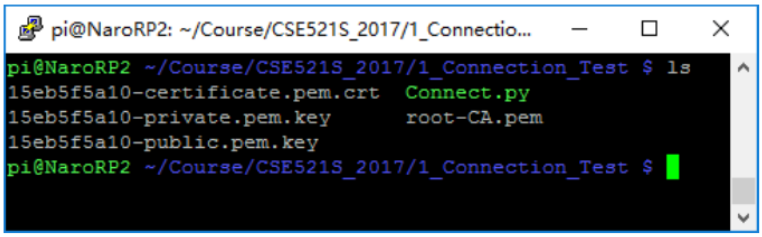
Connect your Device

- Copy certificates to RPI

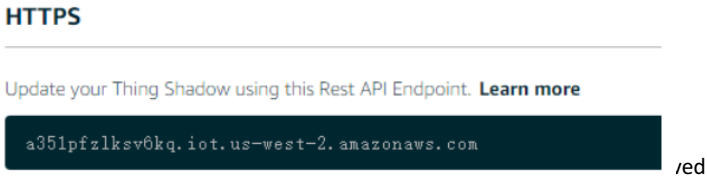
- Choose your AWS SDK (support MQTT)
  - NodeJS
  - Python
  - Java
  - Embedded C
- You can also use 3rd party MQTT tools
  - Python (paho mqtt library)

Important Notes

- You will need these certification when setting up TLS 1.2 verification



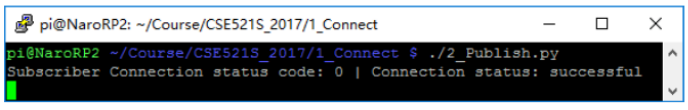
- You will need the AWS IoT endpoint and port (8883) when connect to AWS IoT



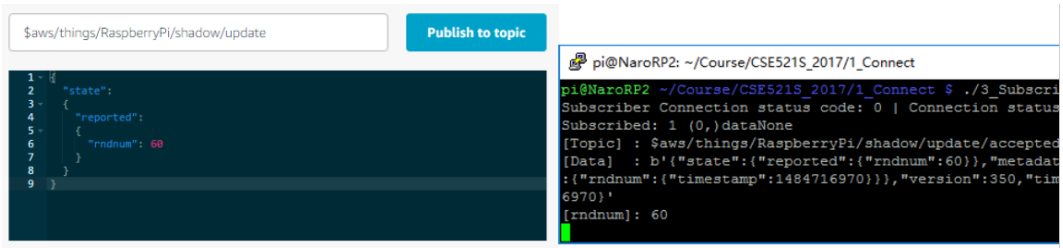
ved

Publish / Subscribe

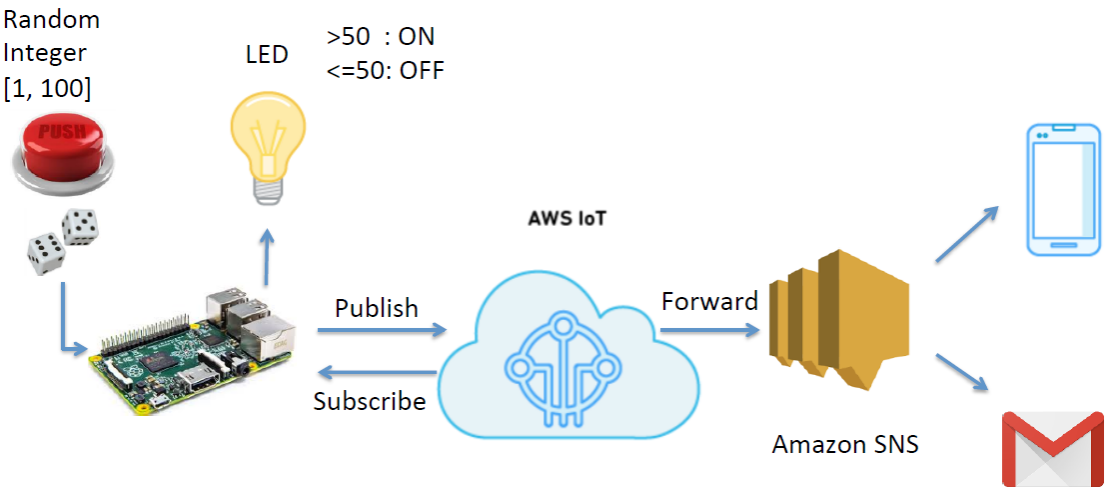
- Publish ➢ payload = "{\"state\":{\"reported\":{\"rndnum\":**50**}}}"



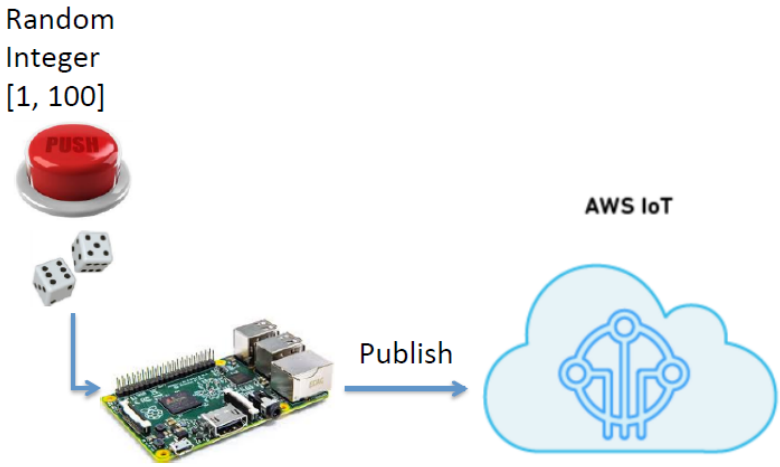
- Subscribe



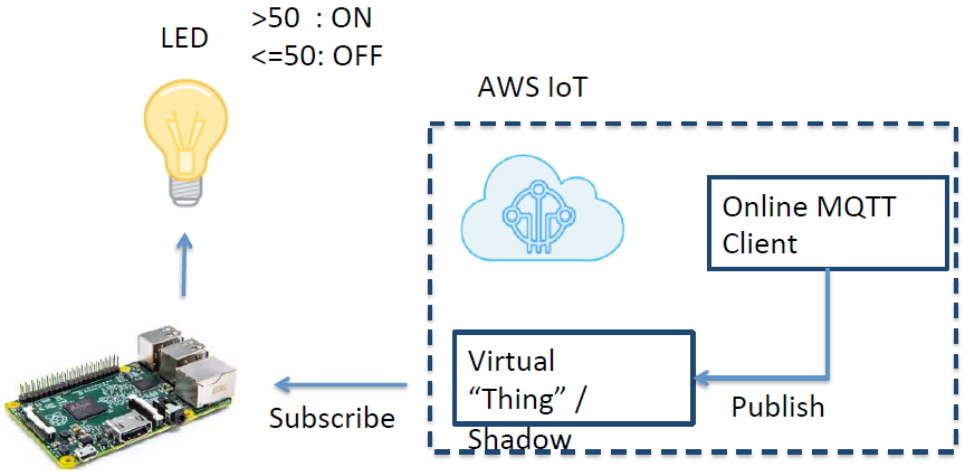
What you can do!!!



Push Button and Publish

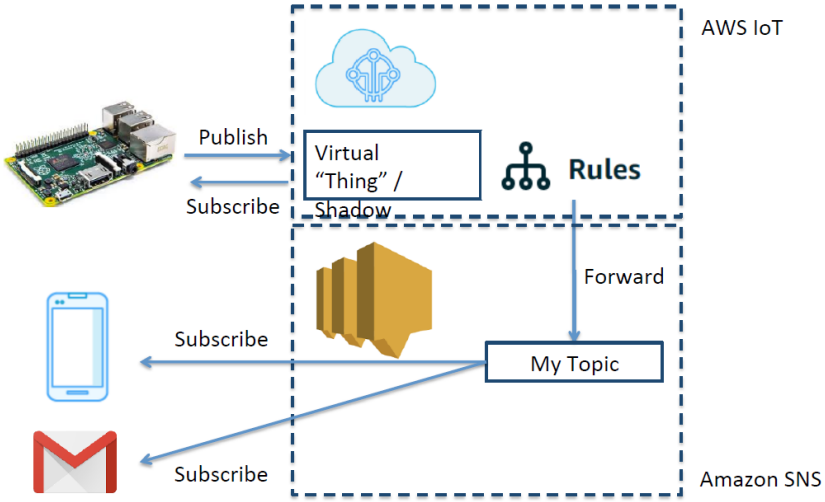


Subscribe and Lit up LED



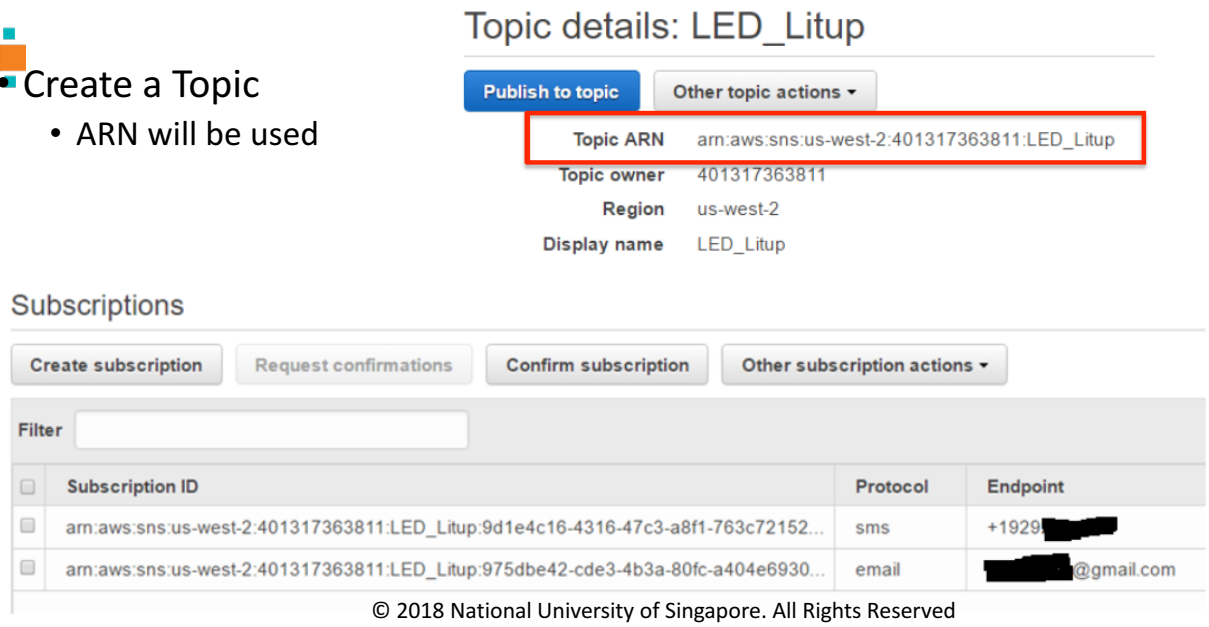
More Fancy: SNS services

- Simple Notification Service



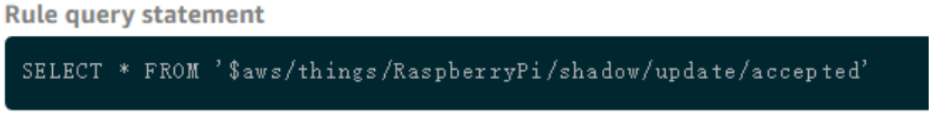
Amazon SNS

- Create a Topic
  - ARN will be used

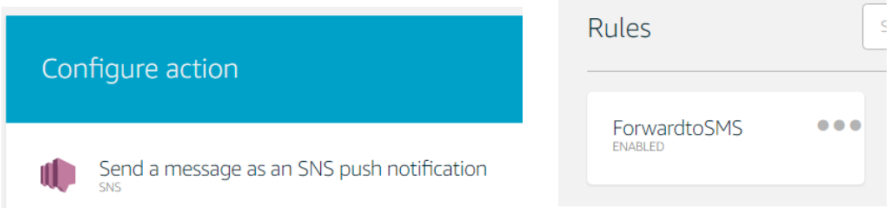


Create a Rule in AWS IoT

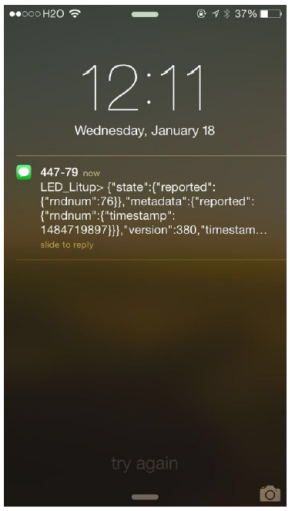
- Add a query to filter your topic (event)

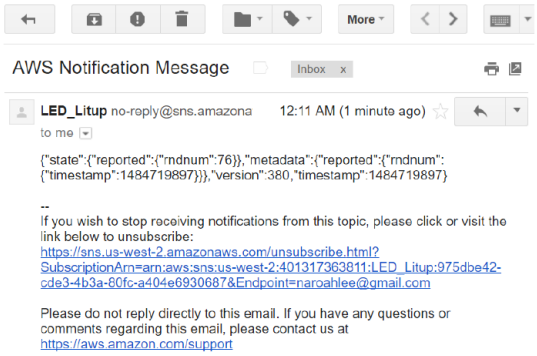


- Add an action:
  - Forward message to SNS
  - Specify destination ARN
  - Enable Rule



Notification on SMS & Email

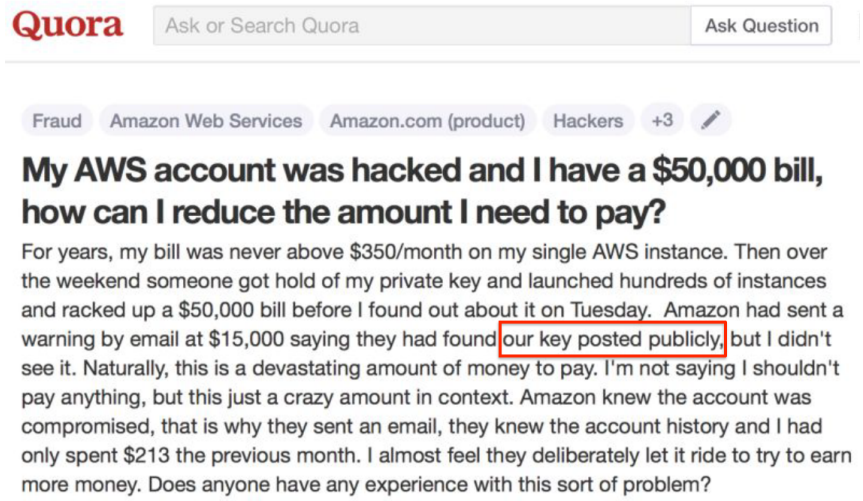




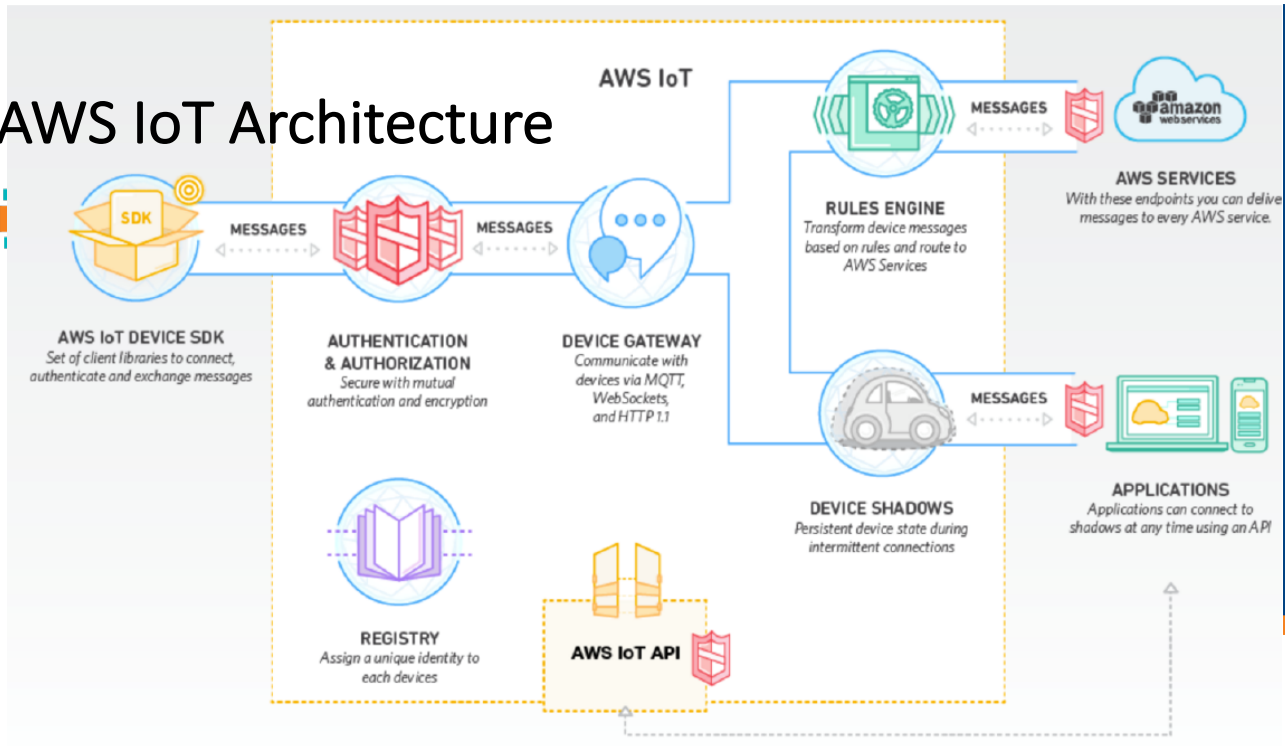
Important Thing: Security

- DON'T **UPLOAD** YOUR PUBLIC KEY!!!

What if…



AWS IoT Architecture



AWS IoT Authentication

- four types of identity principals
  - X.509 certificates
  - IAM users, groups, and roles
  - Amazon Cognito identities
  - Federated identities

Security and Identity for AWS IoT

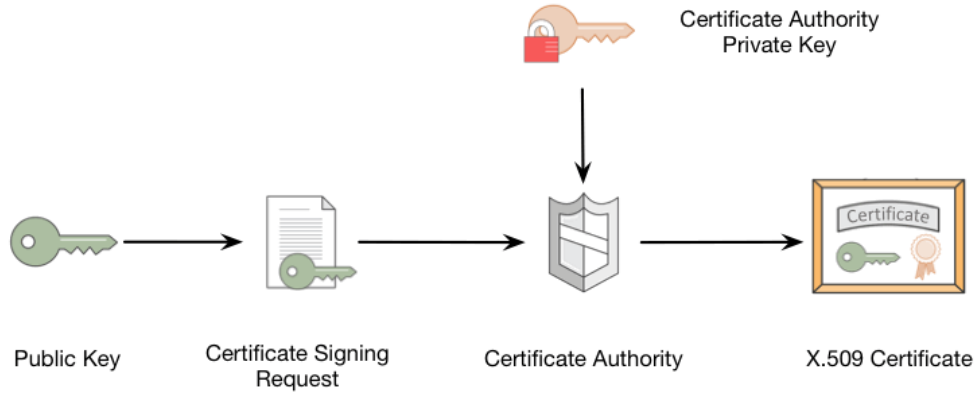
Security and Identity for AWS IoT

- managing device credentials (X.509 certificates, AWS credentials) on your devices and policies in AWS IoT
- assigning unique identities to each device and managing the permissions for a device or group of devices
- Determining Devices connect using your choice of identity (X.509 certificates, IAM users and groups, Amazon Cognito identities, or custom authentication tokens) over a secure connection according to the AWS IoT connection model

X.509 certificate for identity

- public key cryptography, sometimes called asymmetric cryptography
- Public key cryptography uses a pair of keys to enable messages to be securely transferred
- A message can be encrypted using a public key and the only way to decrypt it is to use the corresponding private key:

X.509 certificate for identity



X.509 certificate for identity

Root Certificate:
- Self-Signed, well known
- Has Root Certificate public key
- Signed by Root Certificate private key

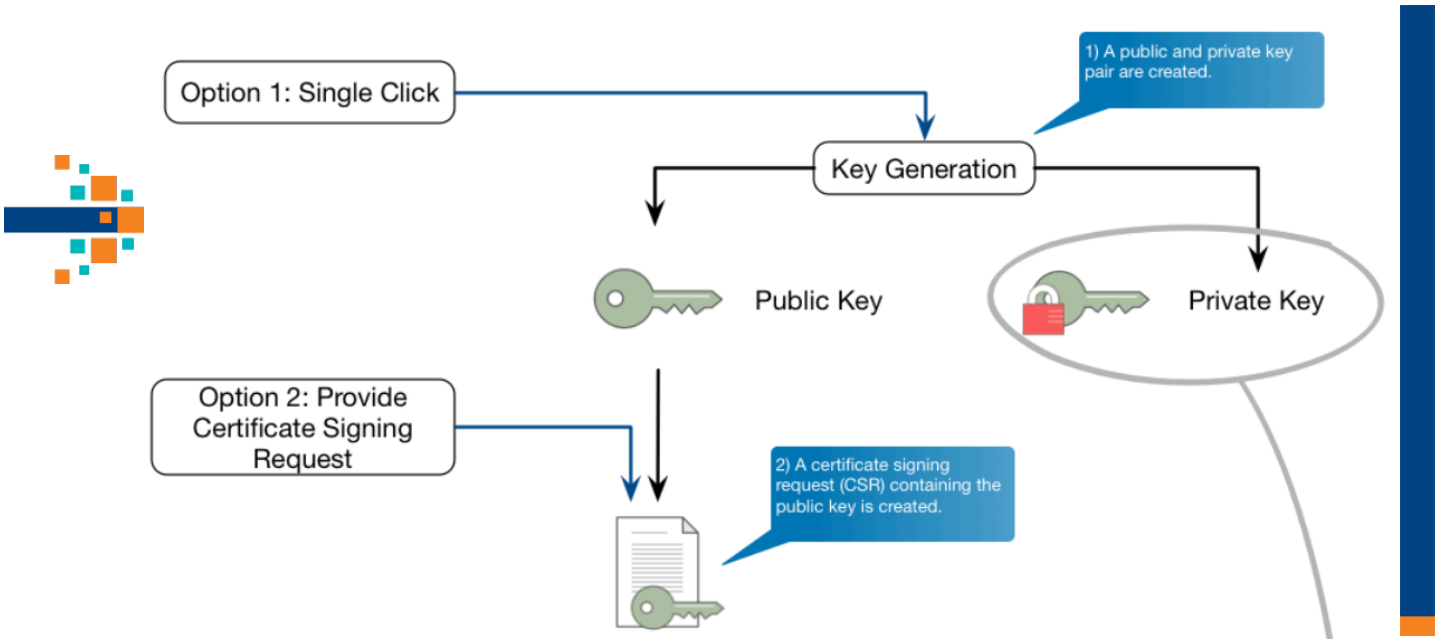
Certificate 1:
- Has own public key
- Signed by Root Certificate private key
- Use Root Certificate public key to prove authenticity

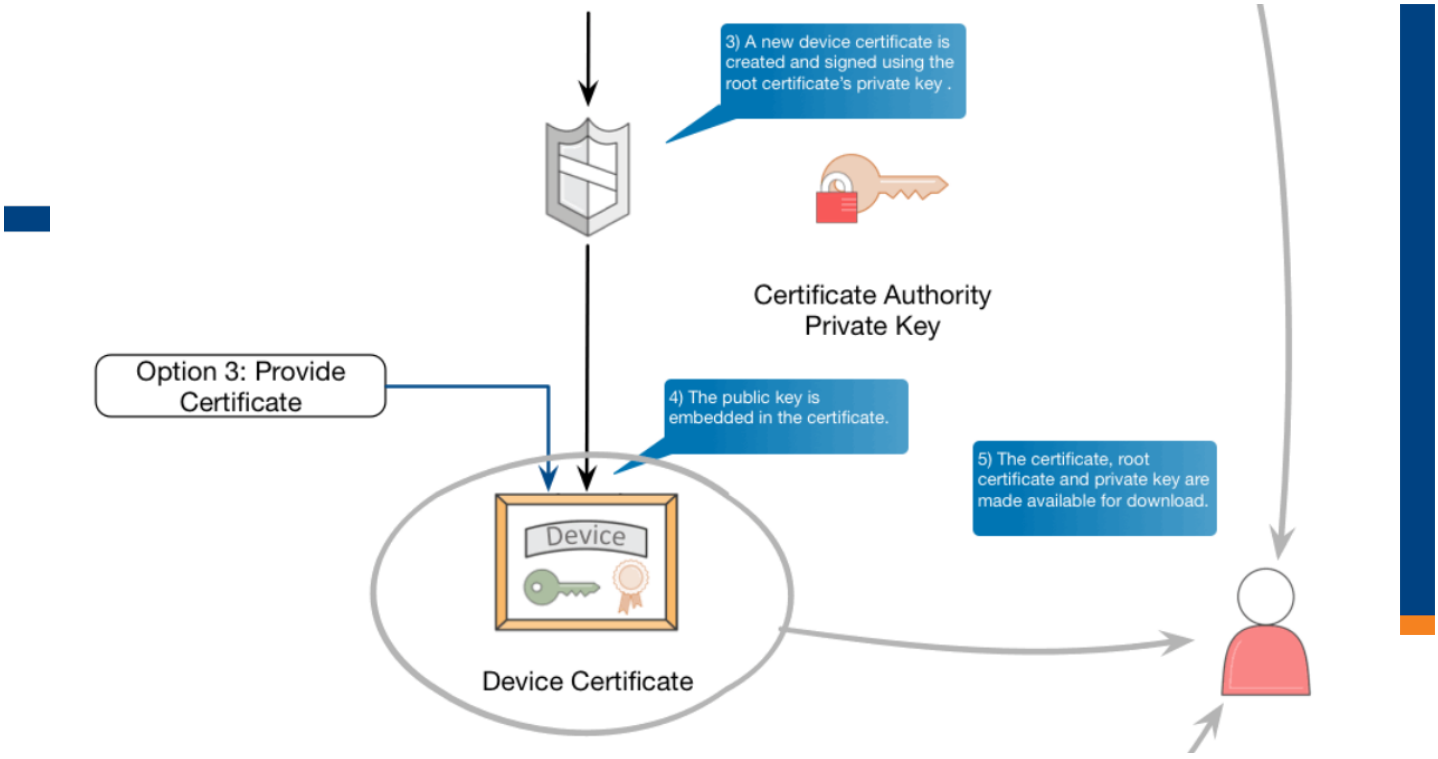
Certificate 2:
- Has own public key
- Signed by Certificate 1 private key
- Use Certificate 1 public key to prove authenticity

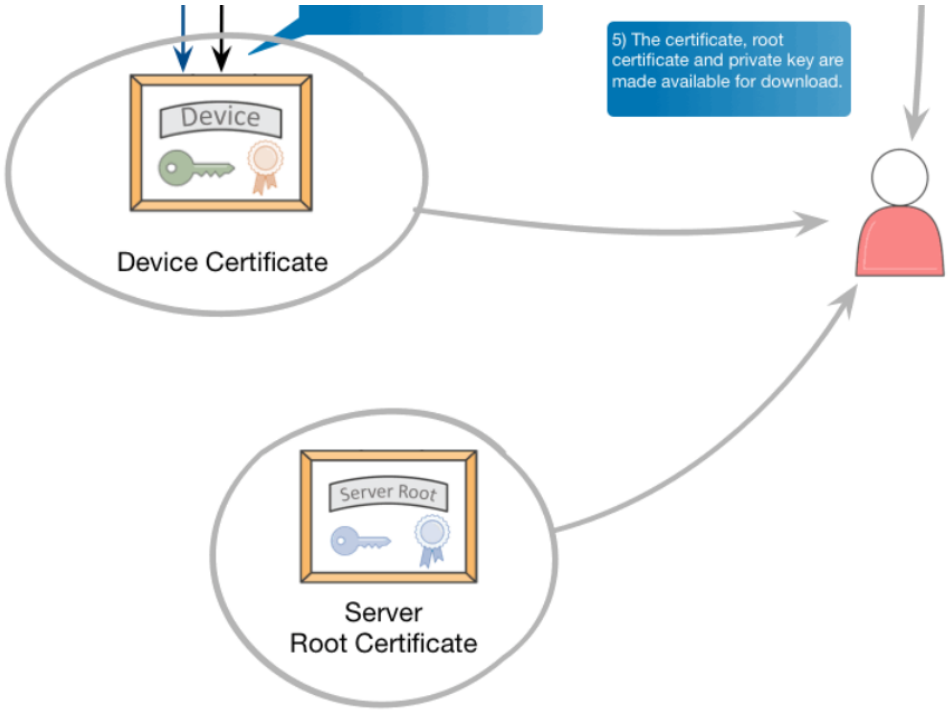
Public Private Key… Continued…

- A key pair is a great way for others to send you secret data
- if you keep your private key secure, anyone with access to the public key can send you an encrypted message that only you can decrypt and read.
- public and private keys also allow you to sign documents.
- a private key is used to add a digital signature to a message. Anyone with the public key can check the signature and know the original message hasn't been altered:

Securing and Identifying Things: Mutual Auth TLS







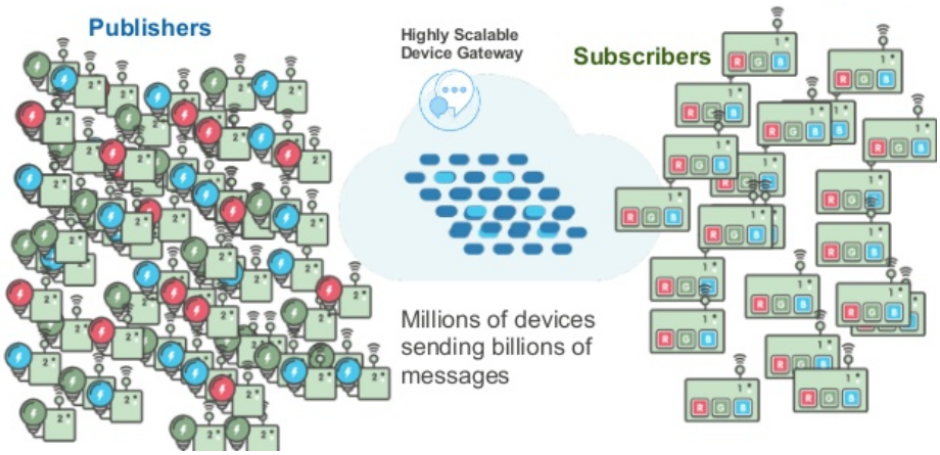
Message Broker

Message Broker (MQTT)

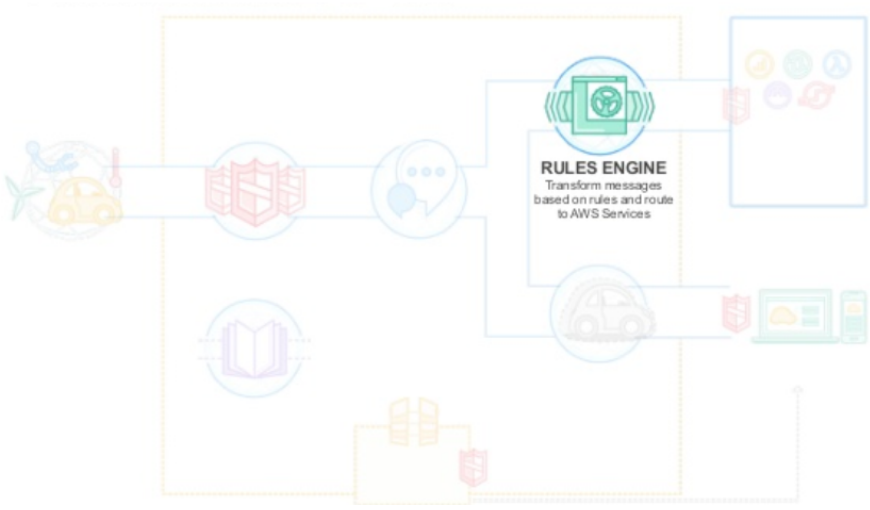
- OASIS standard protocol
- Lightweight, pub-sub, transport protocol
- Application Usage
  - Oil rigs
  - Connected trucks
  - Sensitive and resource-sensitive scenarios

- MQTT vs HTTPS
  - Benchmark
    - 93x faster throughput
    - 11.89x less power consumption to send
    - 170.9x less power consumption to receive
    - 50% less power to keep connected
    - 8x less network overhead
- Source
  - http://stephendnicholas.com/archives/1217

Message Broker: managed service



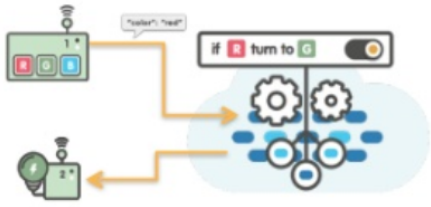
Rules Engine



Rules Engine Basics

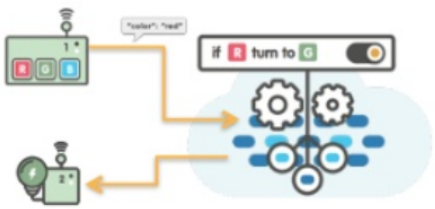
- **Simple & familiar syntax**
  - SQL statement to define topic filter
  - Optional WHERe clause
  - Advanced JSON support

- **Functions improvements**
  - String manipulation (regex support)
  - Mathematical operations
  - Context-based helper functions
  - Crypto support
  - UUID, timestamp, rand, etc.




```
SELECT * FROM 'things/thing-2/color'
WHERE color = 'red'
```

Rules Engine's flexibility

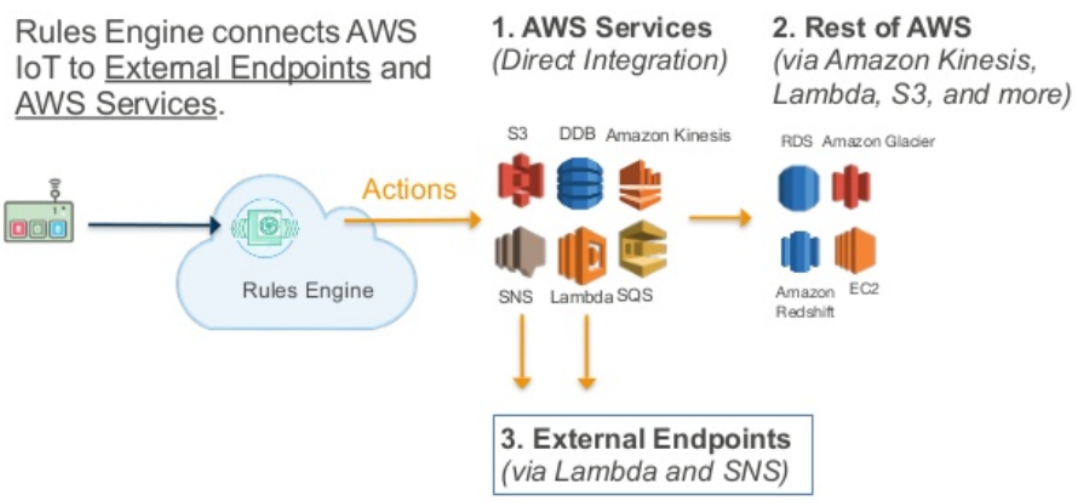



```
SELECT *, clientId() as MQTTClientId
FROM 'one/rule'
WHERE
startsWith(topic(2), 'IME33') AND
(state = 'INIT' OR hydro_temp >
surface_temp)",
"actions":
[{
"republish": {
    "topic":              "controllers/
${substring(topic(3),     3, 5)}",
}]
```

- Complex Evaluations
  - Respond to fleet, not just a simgle unit
  - Dozens of functions available
- Multiple/Simultaneous Actions
  - Sometimes a situation requires you to take many actions

Rules Engine's integration

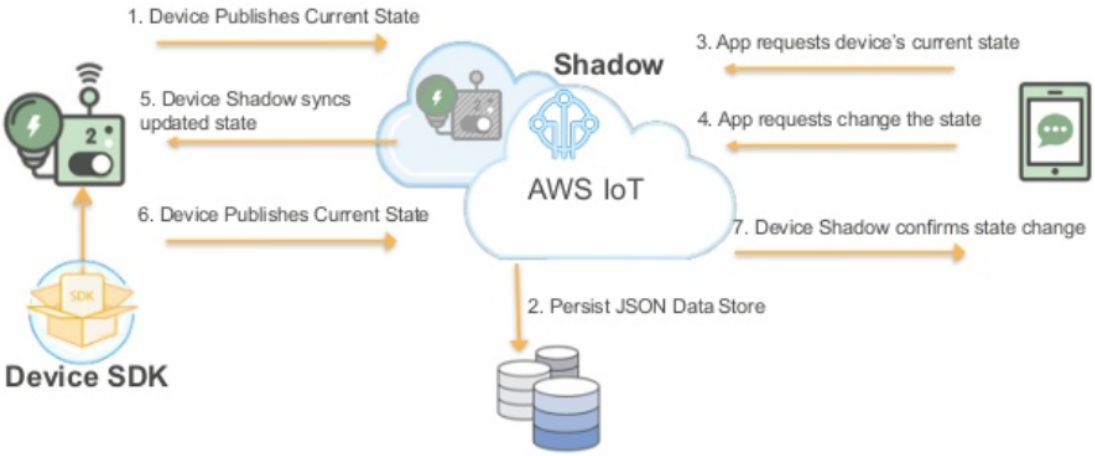
Rules Engine connects AWS IoT to External Endpoints and AWS Services.



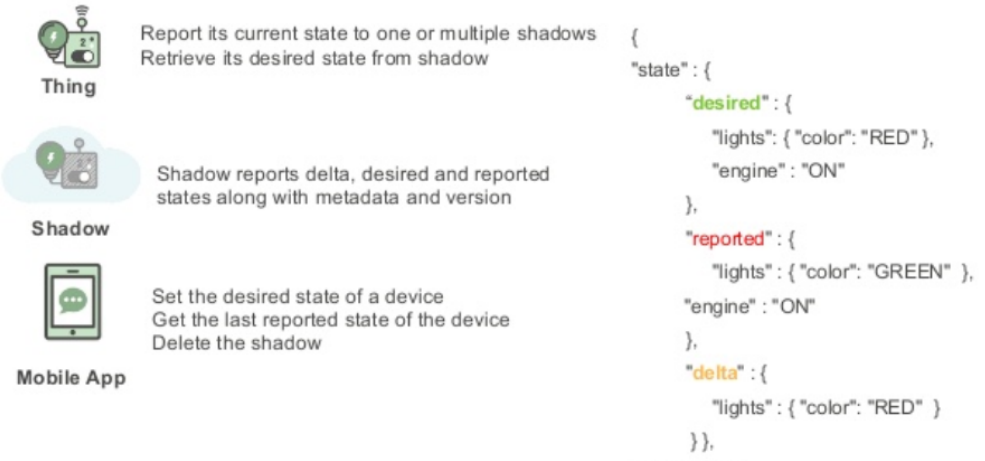
Rules Engine's Actions

- Evaluates inbound messages published into AWS IoT
- Transforms and delivers to appropriate endpoint based on business rules
- External endpoints can be reached via
  - Lambda
  - Simple Notification Services (SNS)

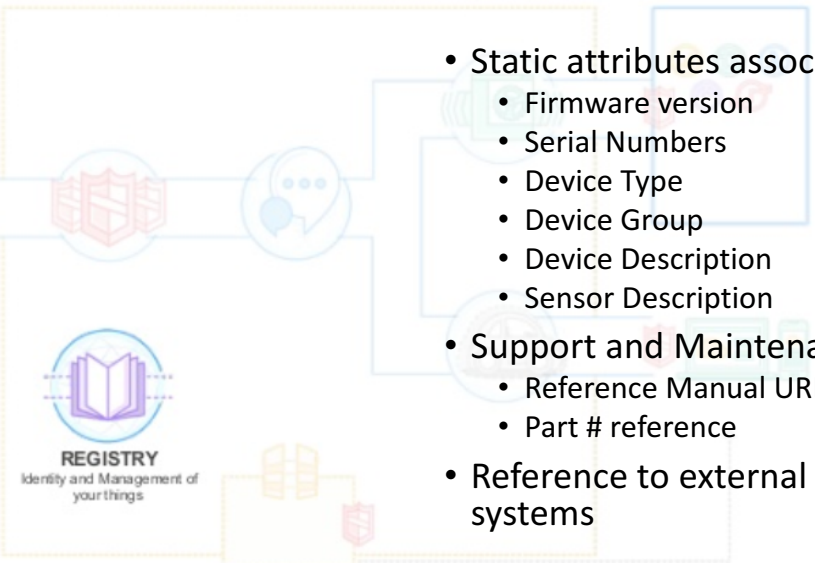
Thing Shadow and Shadow Flow



Thing Shadow and Shadow Flow



AWS IoT Registry



- Static attributes associate to Thing
  - Firmware version
  - Serial Numbers
  - Device Type
  - Device Group
  - Device Description
  - Sensor Description
- Support and Maintenance
  - Reference Manual URL
  - Part # reference
- Reference to external support systems

Device Management

- Ability to update global or within a Region
- Rules Engine keeps state of updates and tracks progress in a DynamoDB Table
- Store Version in Registry Entry

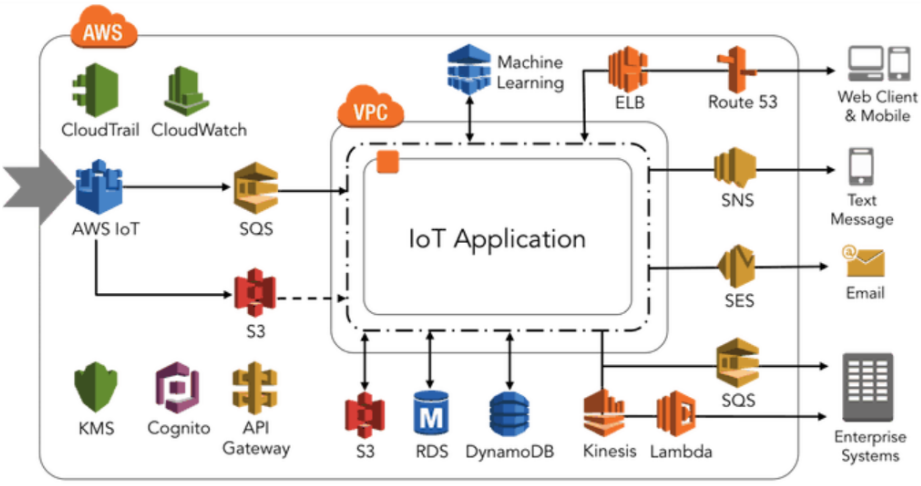
S3 Holds Versioned Firmware Distributions

Organize and secure your firmware binaries in S3

Message Broker notifies groups of the fleet using Topic Patterns

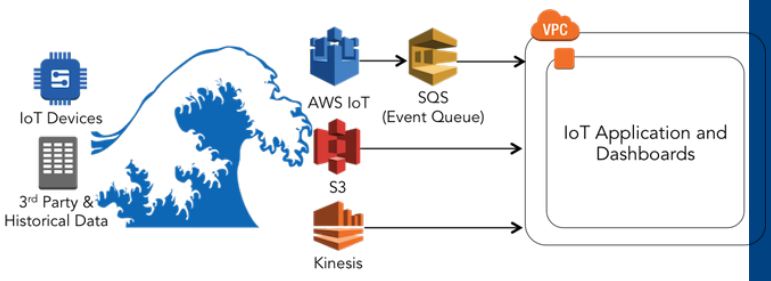
Alert the fleet (or part of it) of the update, and send the URL to the S3 download

Run IoT solutions at scale on AWS



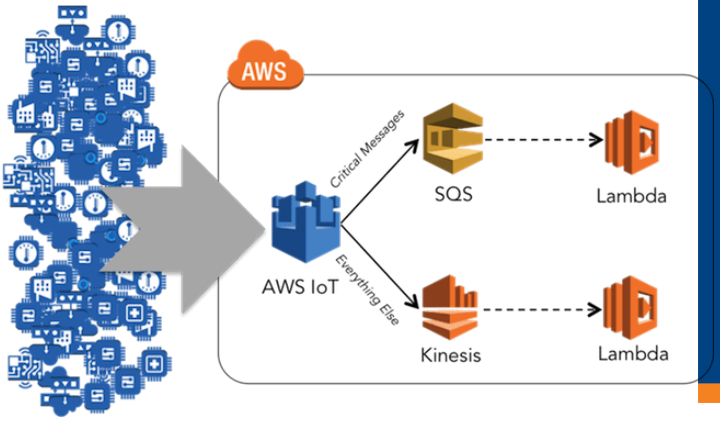
BE READY FOR THE DATA TSUNAMI

- flood of incoming data comes from
  - sudden surge in business
  - consistent monthly growth
  - malicious attack
- architect your system and AWS components
  - ensure all data is always reliably processed and consistent
- focus on building application intelligence
  - value is created from the data



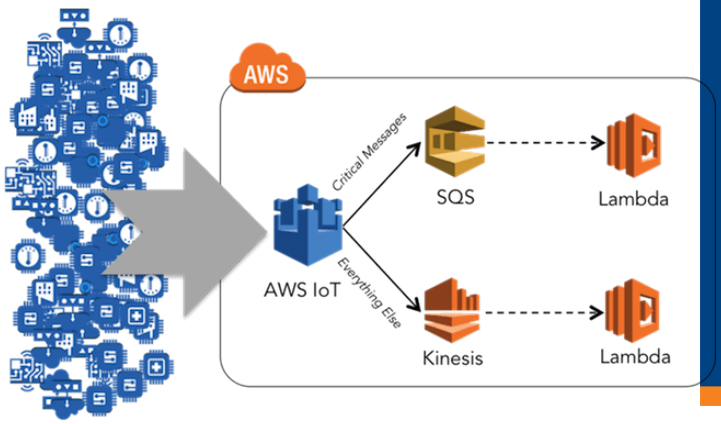
PAY ATTENTION TO THE WEAKEST LINK - 1

- **AWS IoT Rules** allow you to trigger many different actions upon receipt of a message
- not all services are designed to be used as the single point of entry into the system
- some have behaviors that could prevent all of your data from being processed as desired



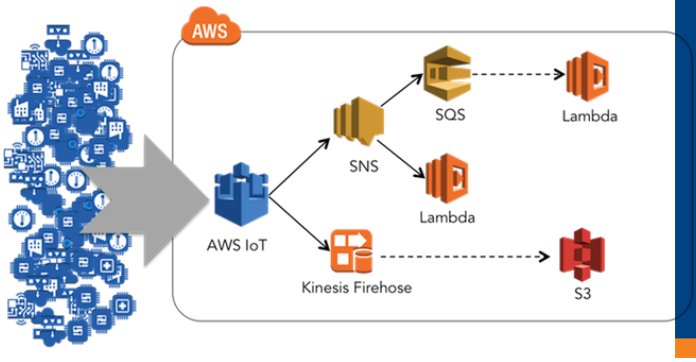
PAY ATTENTION TO THE WEAKEST LINK - 2

- For example, in high volume applications you should consider buffering or queueing incoming data before invoking downstream services like AWS Lambda, so that in the event of subsequent failures the application still has the ability to recover.
- With unpredictable spikes in data load, in the time it takes for a Lambda function to spin up there could already be 100's of thousands of messages in the pipeline. Going to Lambda directly in this example could result in message loss. But If our data is buffered in Kinesis first, this will give downstream systems time to process any large spikes of incoming data.



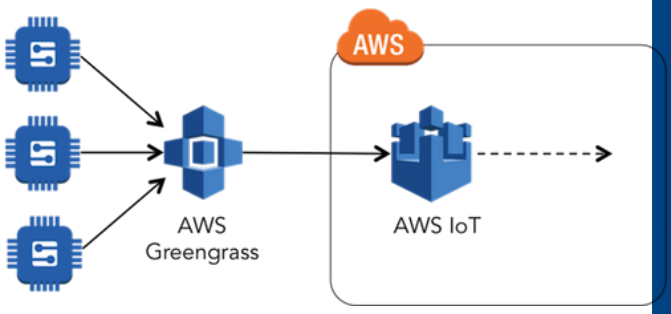
GET IOT DATA INTO A QUEUE AS SOON AS POSSIBLE

- ensure all incoming data is processed correctly
  - go directly from AWS IoT into a Simple Notification Service (SNS) queue
  - allows AWS fan out your data for processing and ensure the entire flood is reliably captured every time
- get your data somewhere 'safe' prior to processing
  - queue, Amazon Kinesis, Amazon S3, and Amazon Redshift



THE BEST WAY TO SCALE IS NOT TO SCALE AT ALL

- reduce traffic in the cloud
  - run AWS Greengrass at the edge
- intelligently process and filter data locally
- eliminating the need to send all device data upstream
- need to send 100% of your data all the time
- capture it all, and hold for a limited time, sending to the cloud only when requested or upon certain error conditions
- send a periodic sample or percentage of device data for use by
  - AWS Machine Learning models
  - other cloud analytics tools



CHOOSE THE INGEST THAT IS BEST

- MQTT is perfect for small payloads of sensor data typical in IoT systems
- MQTT was intended for enabling communications on extremely small embedded devices
  - why use your device's limited power for parsing when it can be offloaded to the cloud?
- Example scenario
  - In a legacy system with log files or XML file dumps, however, it may make more sense to parse the files and send selective values up via MQTT in small chunks.
- key point is that AWS provides several options for data ingest, processing, and storage
  - which should considered as options for different types of data for use in your IoT solution?
- historical and 3rd party data
  - ingest into Amazon RDS database or via a your own custom REST API backed by Amazon API Gateway
  - Or whether S3 is the answer?

References

- AWS IoT
  - http://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html
  - https://docs.aws.amazon.com/iot/latest/developerguide/iot-security-identity.html
  - https://aws.amazon.com/blogs/iot/understanding-the-aws-iot-security-model/
- AWS SNS
  - http://docs.aws.amazon.com/sns/latest/dg/welcome.html