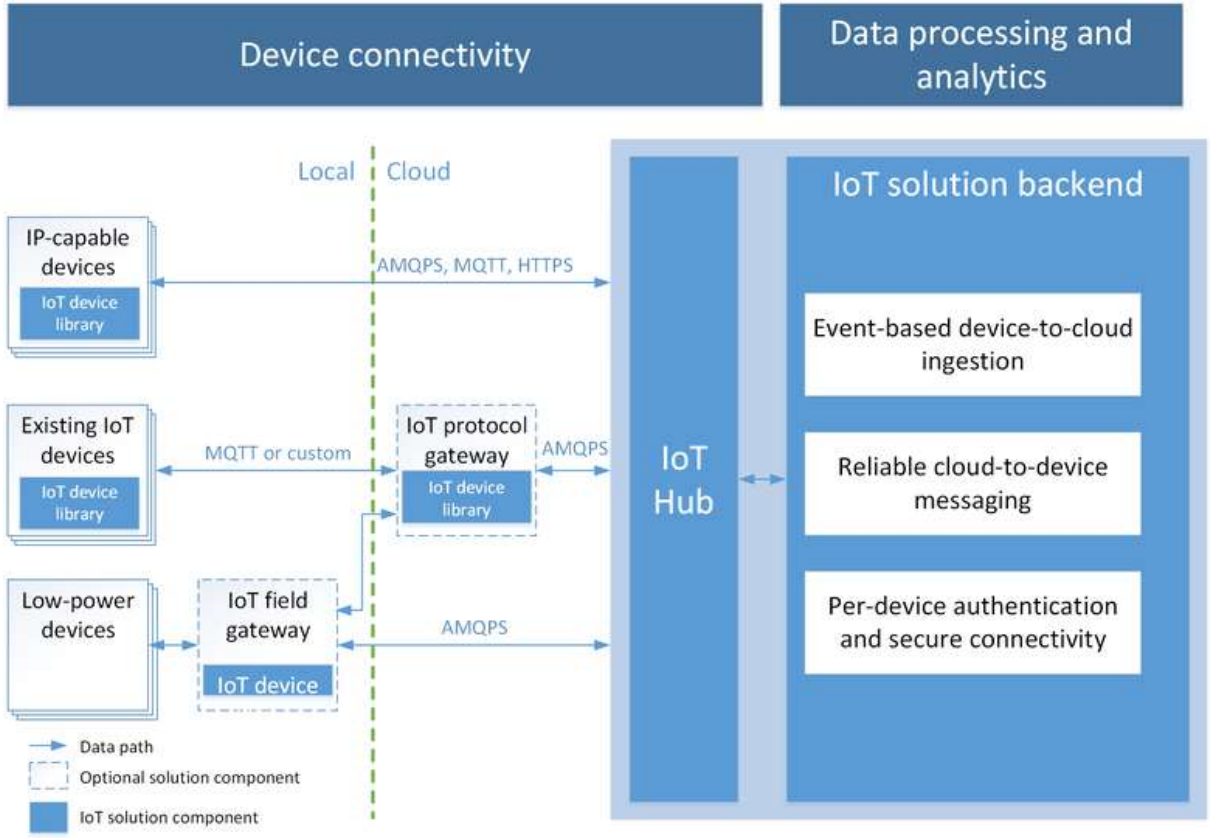
AZURE IOT HUB

Device to Cloud Communication

BaltoMSDN March 22, 2017

MICROSOFT IOT HUB

- IoT Hub is an Azure service to communicate with limited resource devices in the field
- Connect your Internet of Things (IoT) assets.
- Use device-to-cloud telemetry data to understand the state of your devices and assets,
- In cloud-to-device messages, reliably send commands and notifications to your connected devices.
- Accommodate intermittently connected devices.



IOT HUB COMMUNICATION

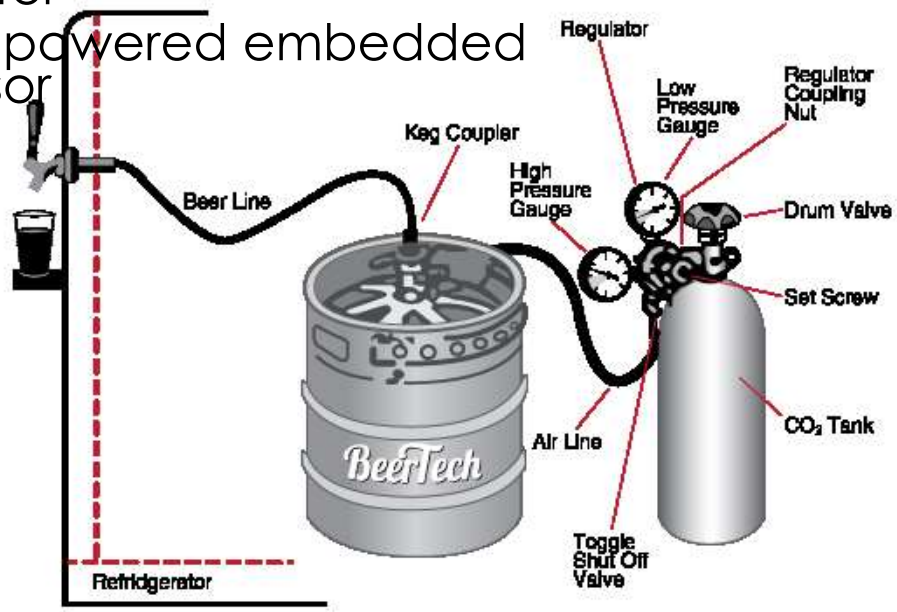
- Device to Cloud
  - Message Up
  - File Upload
- Cloud to Device
  - Message Down
- IoT Hub support devices protocol standards
  - MQTT v3.1.1 (http://www.amqp.org/)
  - HTTP 1.1
  - AMQP 1.0 (http://mqtt.org/)

- Messages are small
  - Limit 256 KBs
- IoT Hub Limits see https://github.com/Azure/azure-content/blob/master/includes/iot-hub-limits.md

MEGA-CRAFT-BREWER KEG CONTROLLER

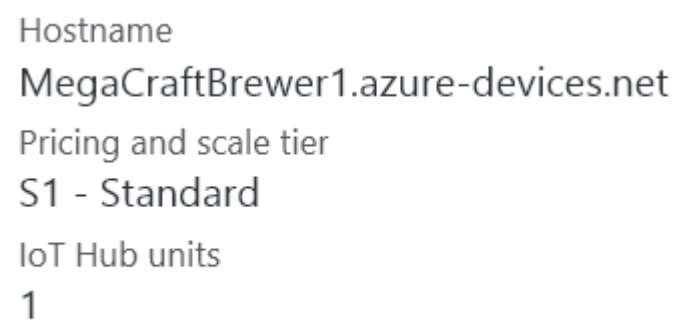
- Collect Keg dispense information in a data warehouse
  - Unique id
  - Keg #
  - Location of Keg
  - Date/Time/Time zone
  - Ounces dispense in one minute period
  - Temperature
  - Product contained (nut brown, stout, pale ale, lager, etc…)
  - Customer
- Control to shutdown dispense if
  - Too old or skunked
  - Customer is not paying invoices
  - Customer resells to another location

- Keg Controller
  - Flow Meter below keg coupler
  - Shutoff Value
  - GSP
  - Satellite modem
  - Thermistor
  - Battery powered embedded processor



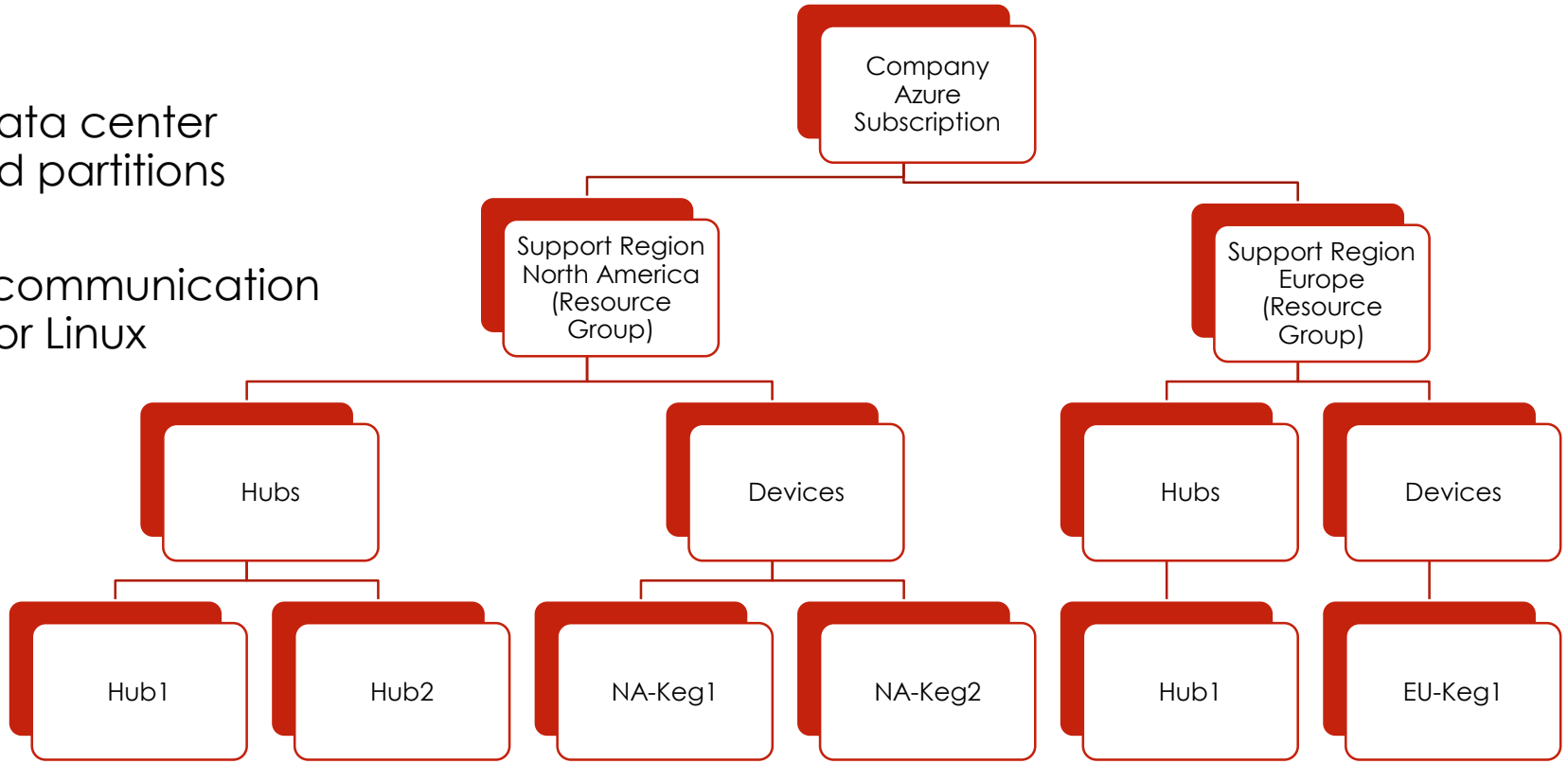
AZURE IOT HUB

- An Azure Resource
- Centralizes Communications
- It is not a originating source or final destination for messages
- Use Other Azure services to process or store the device data
- IOT easily scales to support more devices and more messages
- Allows redundancy for high availability
- Allow basic monitor of hub communications



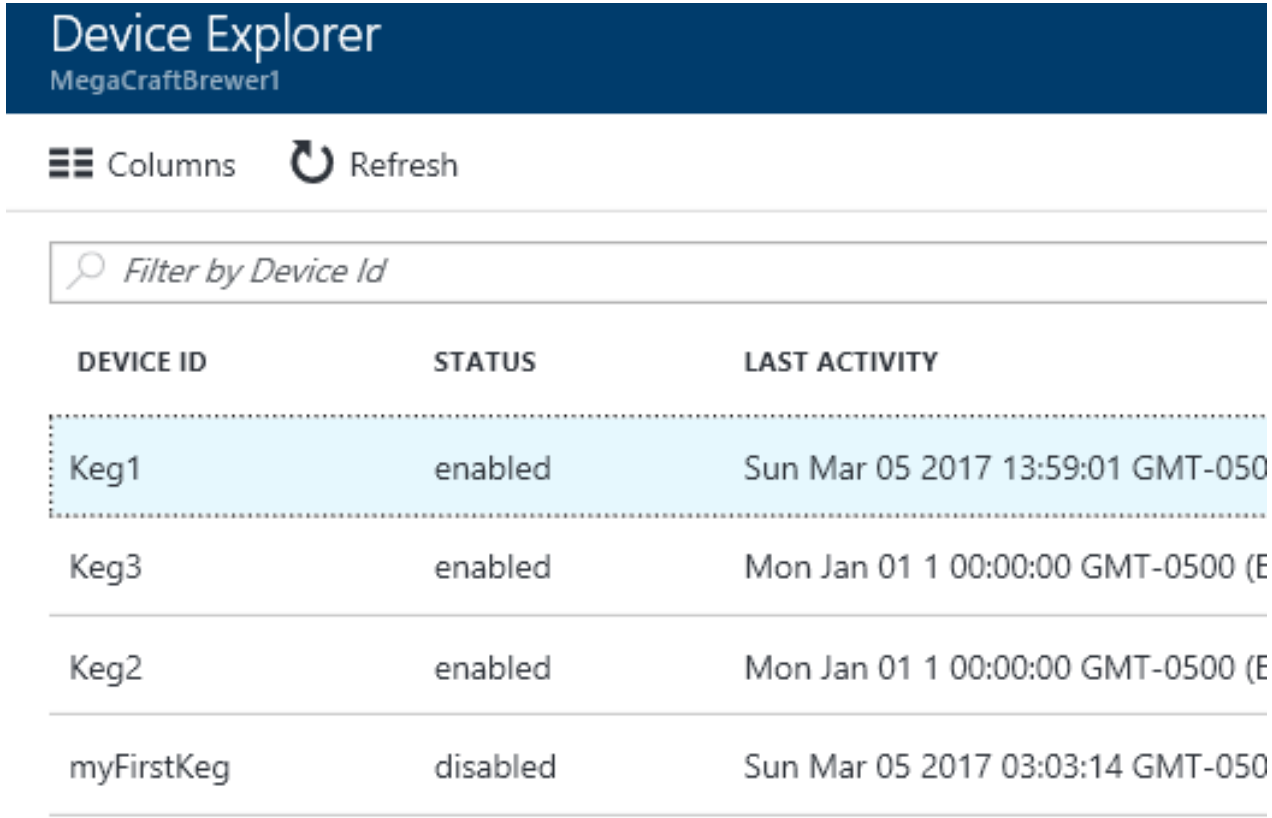
IOT HUB ORGANIZATION

- One Subscription for Organization
- One resource group per support region
  - Logical organization of resources
  - Authorization point
- IoT Hub
  - Located in an Azure data center
  - Scale by : tier, units and partitions
- Device
  - A source or target for communication
  - Embedded, Windows or Linux



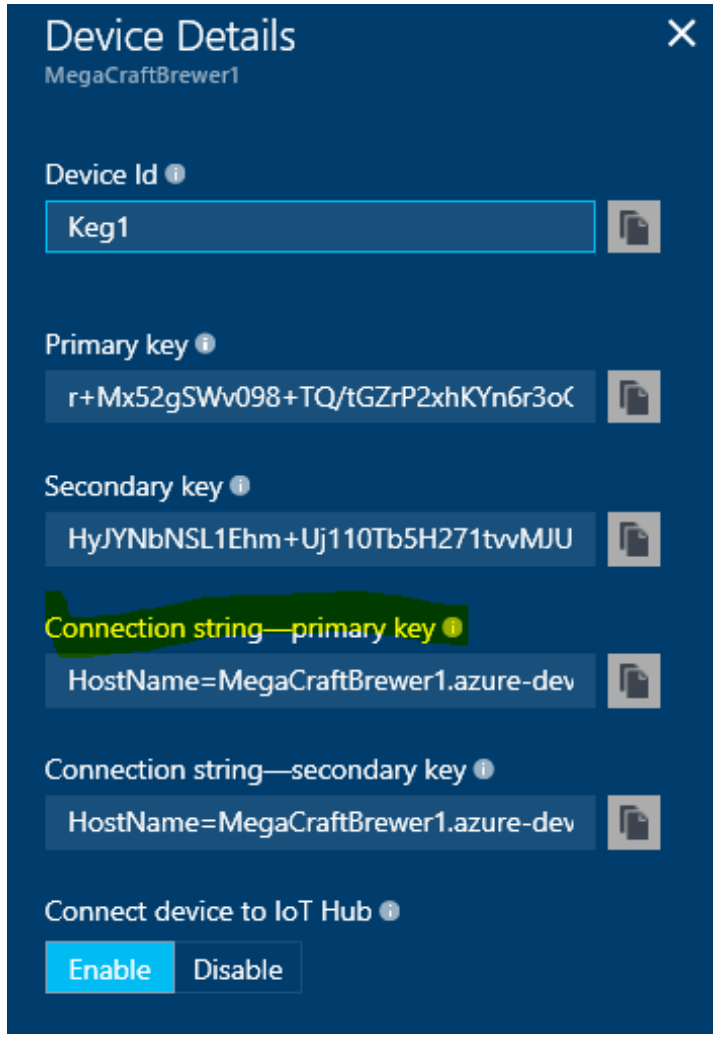
DEVICE REGISTRATION

- Devices must be registered to communicate with IoT Hub
- More privilege required to register than communicate
  - Device secret generated at registration
- A device is a communication source or destination
  - In our example a Keg flow meter and value
- Registration can occur
  - At time of manufacture dispense controller
  - Alternatively on the fly in the field



REGISTERED DEVICE

- Has an ID and secret
- Connection string for connecting device to Hub
- Last time of message send or received
- Enable and disable device
  - Stops traffic from device



SENDING A MESSAGE TO THE HUB

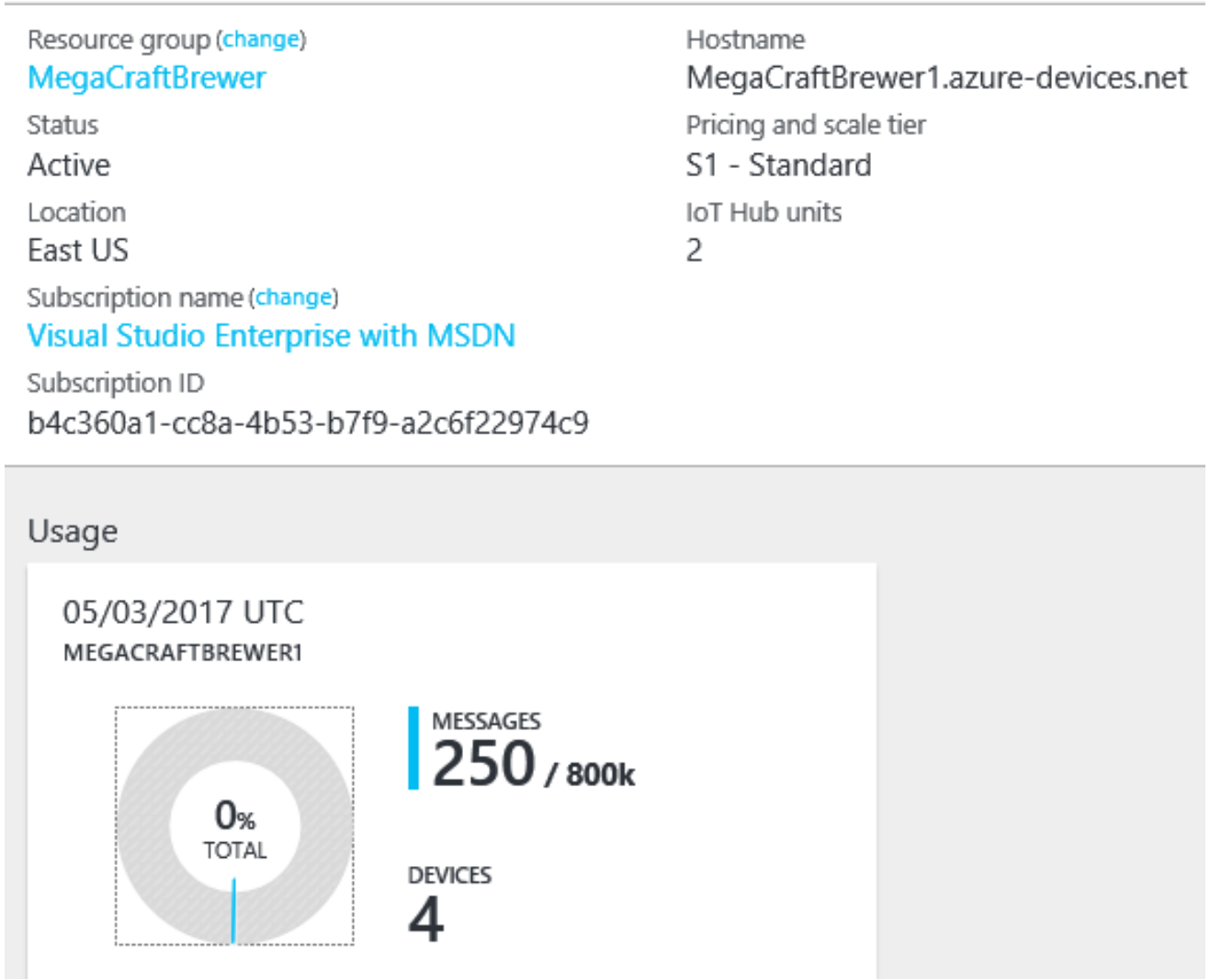
- Message
  - Envelop
    - To whom
    - From whom (connection string)
    - Message ID (must be unique)
    - Enqueue & Dequeue times
  - Payload
    - Binary

```
deviceClient =
DeviceClient.CreateFromConnection
String(
    deviceConnectionString,
    TransportType.Http1);
deviceClient.OpenAsync();
```

```
public void SendDeviceToCloudMessages<T>
    (T obj)
    where T : IPayload
{
    _stm.Seek(0, SeekOrigin.Begin);
    _formatter.Serialize(_stm, obj);
    _stm.Seek(0, SeekOrigin.Begin);
    var message = new Message(_stm);
    message.To = obj.GetType().FullName;
    message.MessageId = obj.Id.
        ToString("n");

    var task = _deviceClient.
         SendEventAsync(message);
   task.Wait(timeOut);
}
```

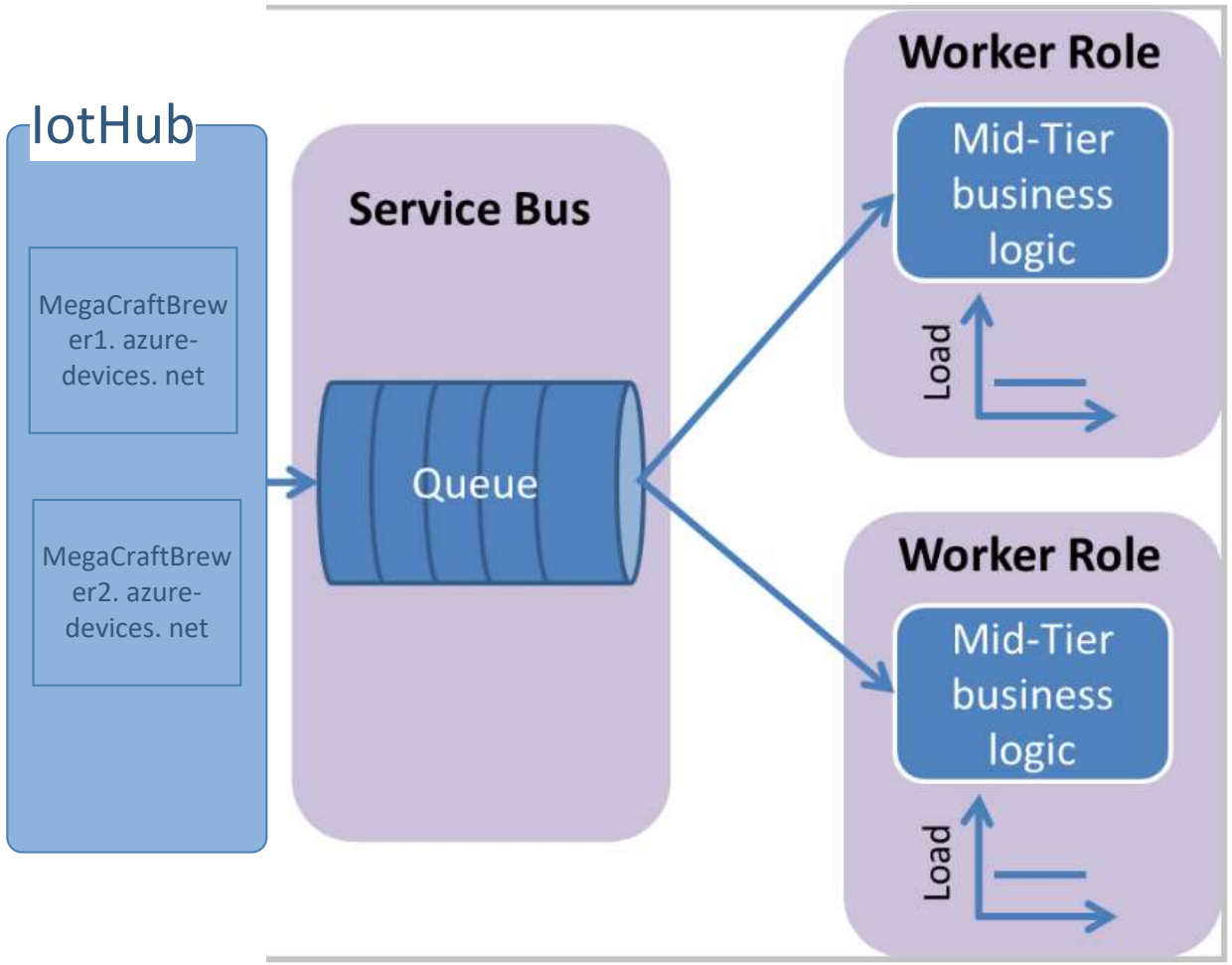
HUB OVERVIEW & STATUS



- Resource Group
- Hostname
- Location (Data Center)
- Tier
- Hub Units
- Message per day per hub unit
  - Percentage of message used
  - S1 400K x 2 hubs
- Number of registered devices

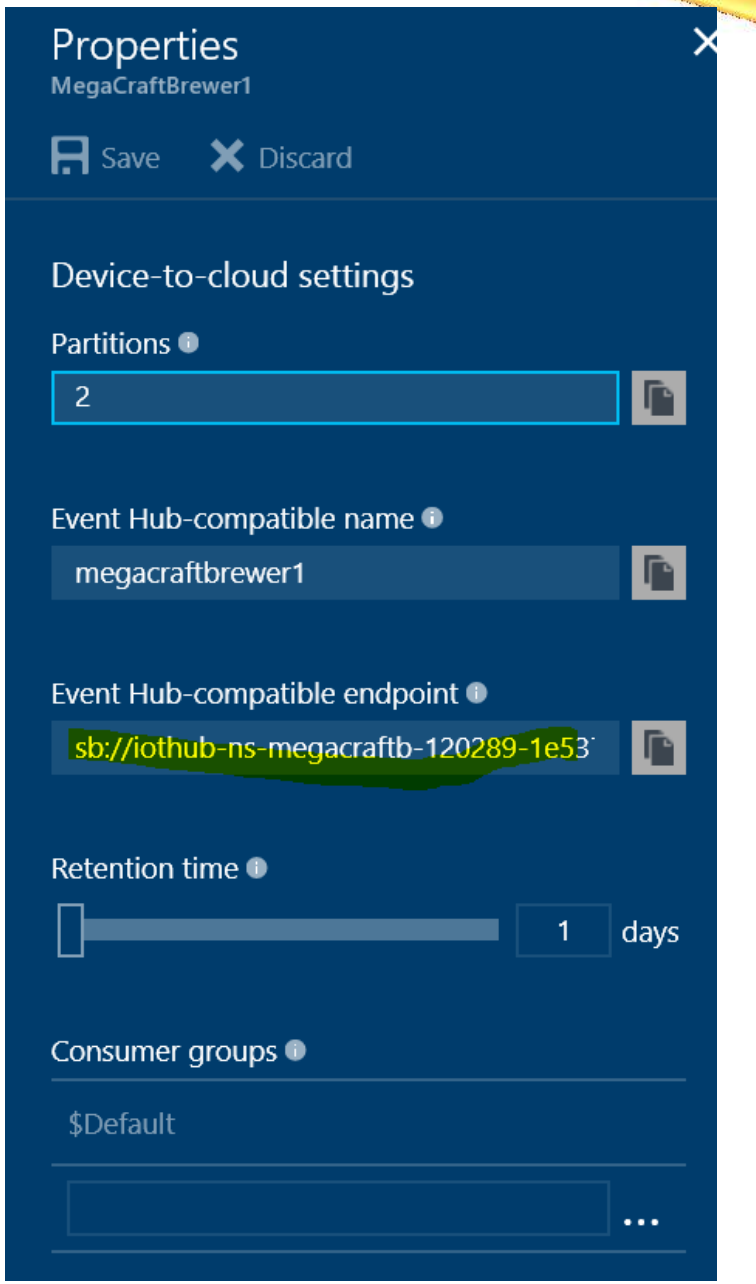
SERVICE BUS AND WORKER ROLE

- IotHub places income messages onto service bus
- Worker role de-queue and process message from service bus
  - Not really a queue, more of a stream
- Worker role is a very scalable Windows Service



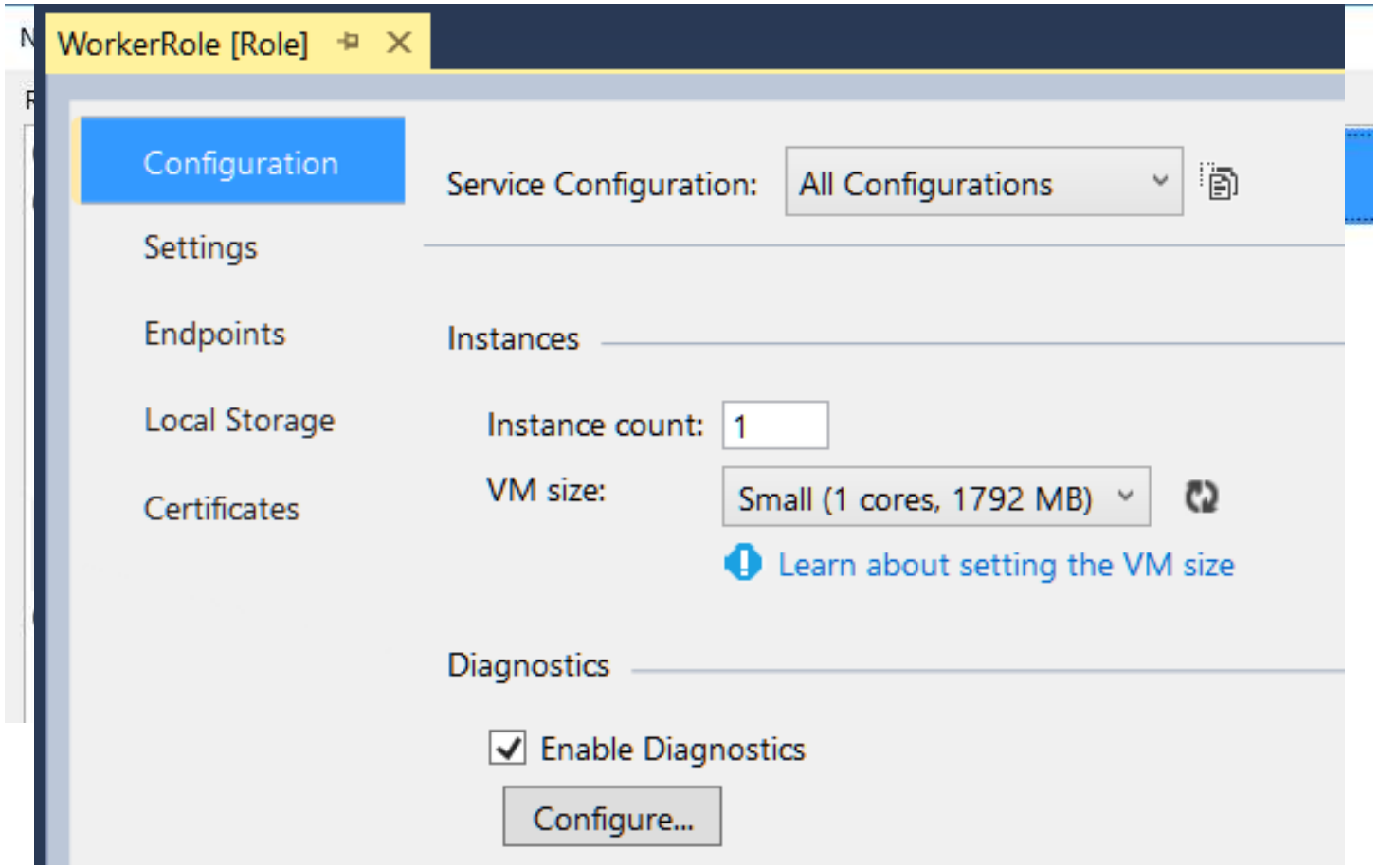
CREATE A SERVICE BUS & QUEUE

- IoT Hub has two default end-points
- The event end-point receives messages sent by devices to the cloud
- End-points allow process of messages from any
  - Consumer group
  - Partition (multiple streams)
- Use a built in service user to authenticate and retrieve messages



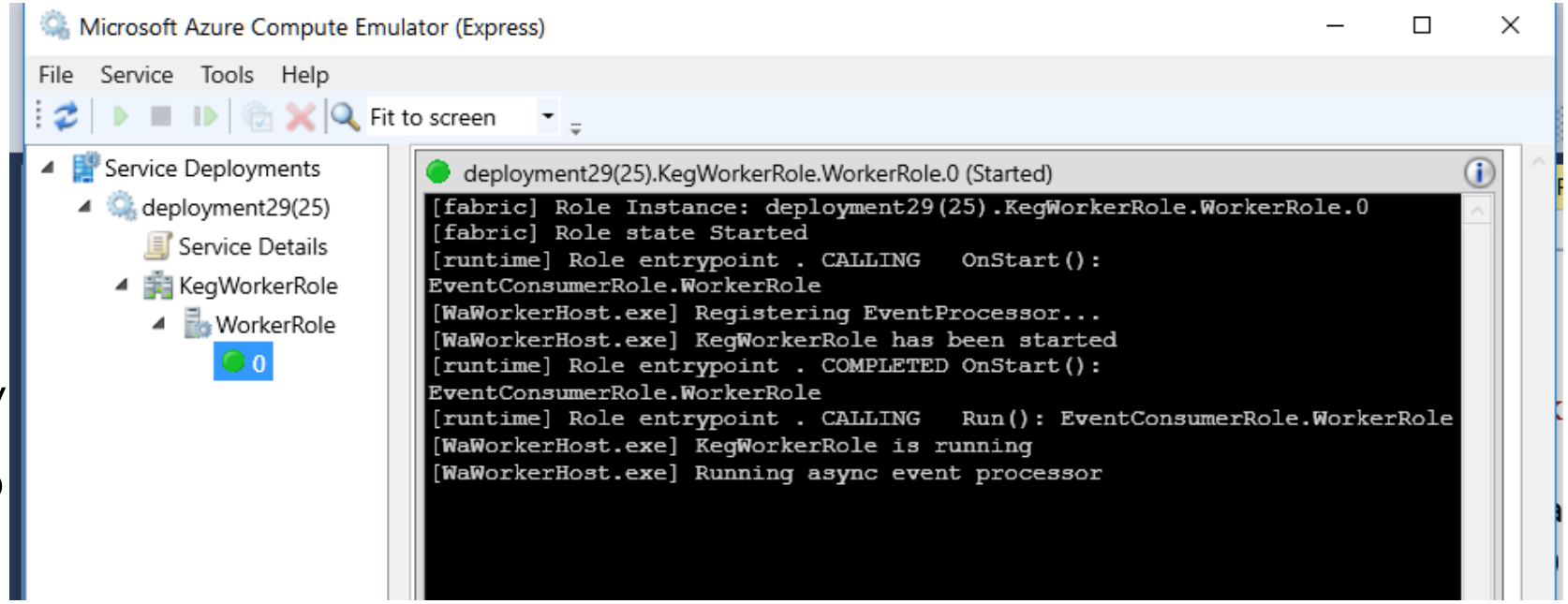
AZURE WORKER ROLE

- Create a C# Azure Cloud Service
- Then select worker role
  - With service bus
- Basically a deployment framework
  - Virtual Machine
  - Worker roles



KEG CONSUMER ROLE

- Standard Library Assembly
- <RoleType>Worker</RoleType>
- Compute simulator lets you debug locally
- Trace.TraceInformation appear in emulator UI and Azure storage
- RoleEntryPoint is like ServiceBase

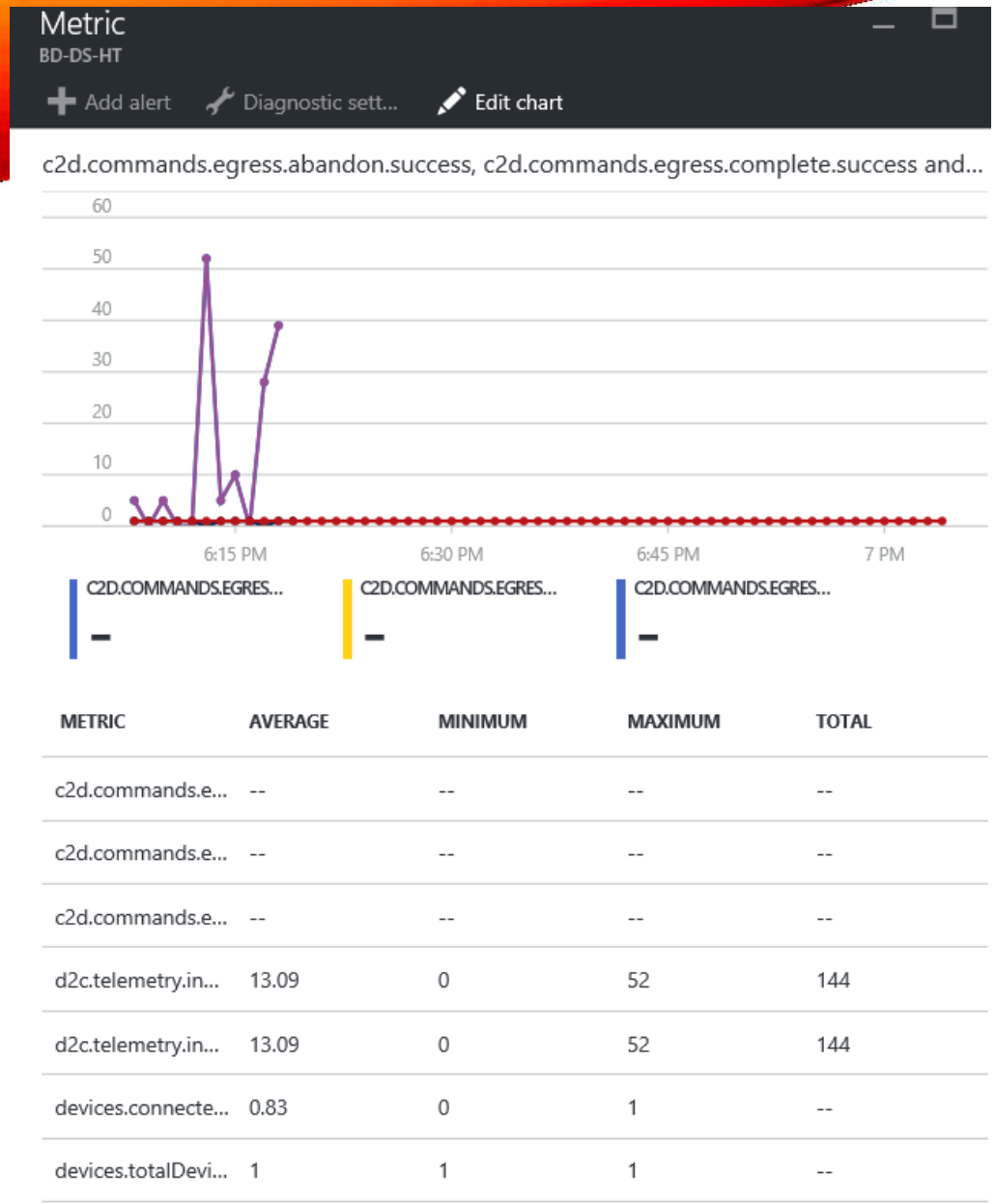


SHUT THE VALVE

- Cloud to Device command
- When our beer is too hot for two minutes, shut the valve
- Address to a single device
- Hangs around for a day waiting for the device to connect and receive it

3/17/2017 7:09:47 PM > Sending message: 496f9272e3b54f69a3511b7bcb86215f to Payload.DispenseInformation Received 57344666-9572-4ff7-b309-e881b05806bc Dispenser 1 valve Closed.

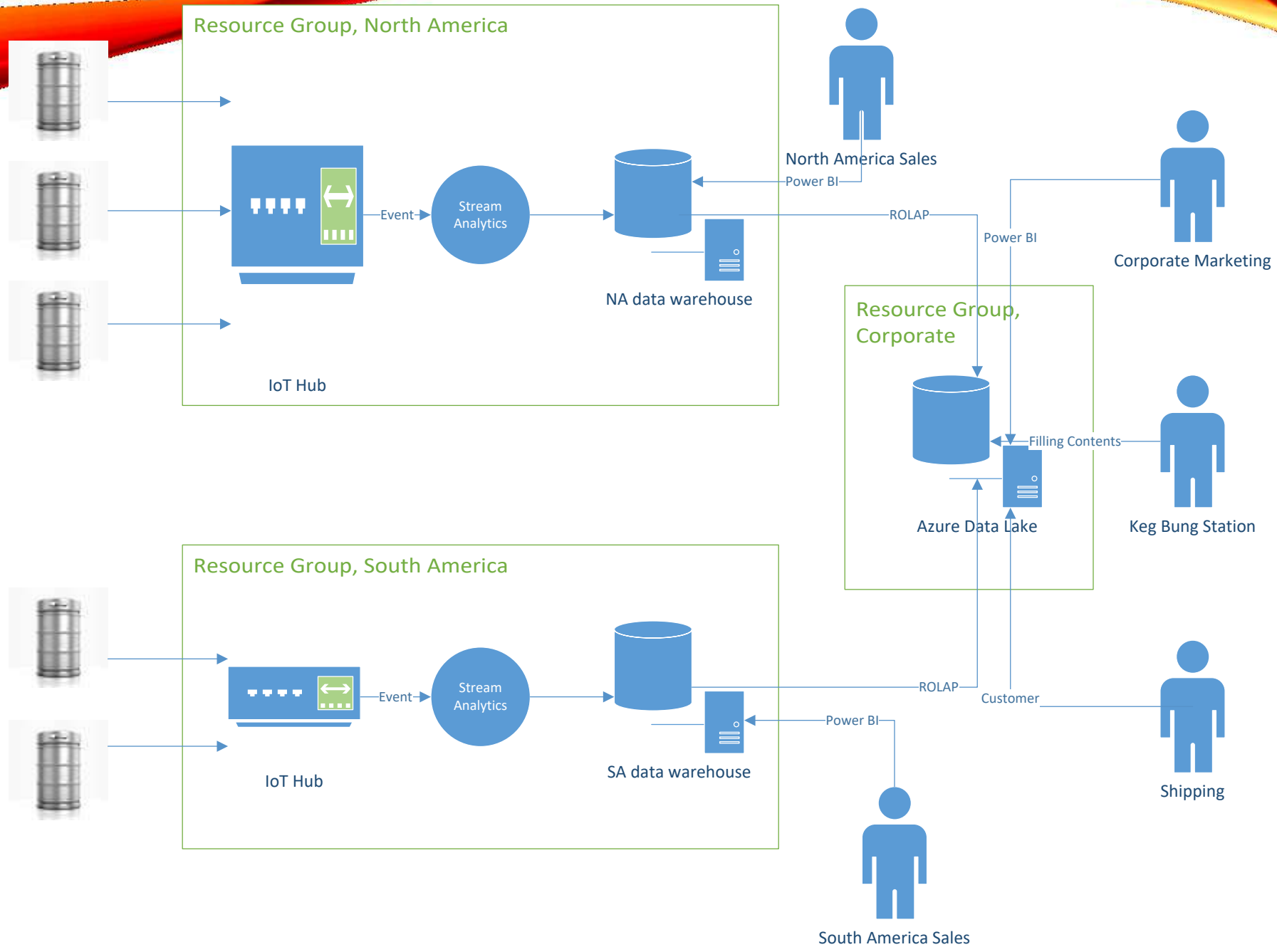
```
Message deviceMessage =
        _formatter.CreateMessages(dispCommand);
ServiceClient serviceClient;
serviceClient =
        ServiceClient.CreateFromConnectionString(
            serviceConnectionString,
            Microsoft.Azure.Devices.TransportType.Amqp);
serviceClient.OpenAsync().Wait(maxTime);
var sendTask = serviceClient.SendAsync(
        deviceName, deviceMessage);
sendTask.Wait(maxTime);
serviceClient.CloseAsync().Wait(maxTime);
Trace.TraceError($"{dispCommand} sent to {deviceName}.");
```



MONITORING

- Hub Communication Monitoring
  - Monitors health of communication
  - Does not monitor health of devices
  - Time of last communication
- Metrics of messages
  - Send/received
  - Processed

MEGA-CRAFT-BREWER IOT HUB APPLICATION



BEWARE

- Azure is Evergreen
- Nuget packages galore
- Message duplicate dequeue and concurrent dequeuer of same message