AWS GreenGrass IoT with Enterprise Message Queue

Version 3.0

Team GreenFace

Viet Nguyen

Minh Ngo

Hoa Nguyen

Instructor

Andrew Bond

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| October 15 | 1 | Initial document | The team |
| November 20 | 2 | Design document | The team |
| December 5 | 3 | Final report | The team |

[AWS GreenGrass IoT with Enterprise Message Queue 1](#_Toc532047547)

[Version 3.0 1](#_Toc532047548)

[Team GreenFace 1](#_Toc532047549)

[Revision History 2](#_Toc532047550)

[1. Introduction 5](#_Toc532047551)

[1.1 Purpose of this document 5](#_Toc532047552)

[1.2 Intended Audience 5](#_Toc532047553)

[1.3 Scope 5](#_Toc532047554)

[1.4 Definitions and acronyms 5](#_Toc532047555)

[1.5 References 6](#_Toc532047556)

[2. Background and Objectives 6](#_Toc532047557)

[2.1 Background 6](#_Toc532047558)

[2.2 Objectives & solution description 7](#_Toc532047559)

[3. Project Requirements and Use Cases 8](#_Toc532047560)

[3.1 Requirements 8](#_Toc532047561)

[3.2 Use Cases 9](#_Toc532047562)

[4. Architecture & High-Level Design 11](#_Toc532047563)

[4.1 System Diagram 11](#_Toc532047564)

[4.2 Message Queue’s High-level diagram 12](#_Toc532047565)

[4.3 Traffic Light Simulation Architecture 13](#_Toc532047566)

[5. Organization 14](#_Toc532047567)

[5.1 Project group 14](#_Toc532047568)

[5.2 Stakeholders 14](#_Toc532047569)

[6. Development process 15](#_Toc532047570)

[7. Deliverables 15](#_Toc532047571)

[8. Project Risks 16](#_Toc532047572)

[9. Communication 16](#_Toc532047573)

[9.1 Collaboration 16](#_Toc532047574)

[9.2 Git 16](#_Toc532047575)

[10. Project Plan 17](#_Toc532047576)

[10.1 Project Gantt Chart 17](#_Toc532047577)

[10.2 Project Timeline Remark 17](#_Toc532047578)

[10.3 Test Plan 17](#_Toc532047579)

[11. References 19](#_Toc532047580)

# Introduction

## Purpose of this document

The purpose of this document is to provide a detailed project description of the application called AWS GreenGrass IoT with Enterprise Message Queue, which is designed to demonstrate the capabilities of AWS GreenGrass IoT platform in the context of enterprise environment. The message queue is an example of GreenGrass’s built-in service. Besides that, a traffic light simulation shall be used as proof of concept to the platform’s add-on services.

To be more specific, this document shall focus on the system design, use cases, project timeline, schedule and relevant design detail.

## Intended Audience

This document shall be used in all phases of the project as a guideline. Intended audiences of this project are all project stakeholders:

* Class Instructor
* Grader
* Project leader
* Team members

## Scope

This document defines the project plan of the AWS GreenGrass IoT with Enterprise Message Queue application. The overview includes objectives of the project, organization of the project team, development process that is going to be used during the project, communication used between project stakeholders and project plan that includes time schedule and activity plan.

## Definitions and acronyms

|  |  |
| --- | --- |
| Keyword | Definitions |
| AWS GreenGrass IoT with Enterprise Message Queue | The name of the project |
| AWS | Amazon Web Services |
| Device Shadow | where we do all the commands and control operations. A critical part of GreenGrass. It stores operational state of our devices, reported back to Device Gateway. |
| IoT | Internet of things |
| Lambda | Event driven compute functions  Write lambda function right on the browser, upload to the cloud and deploy it to the devices locally as we wish. |
| MQTT | an ISO standard publish-subscribe-based messaging protocol. It works on top of the TCP/IP protocol. |
| Rule | defines the way how we want to receive down-streamed data (applied rules (AWS IoT action) on multiple incoming messages |
| Subscription List | It acts like the routing table defining the data going back and forth between a device/service on the cloud to a target or vice versa. The target can be a particular lambda function on the core. |
| GreenGrass Core | Json document used to keep tracks of the state of your IoT devices and Lambdas. |
| Message Queue | An enterprise messaging system (EMS) or messaging system is a set of published enterprise-wide standards that allows organizations to send semantically precise messages between computer systems. |
| Project Leader | A person in charge of organizing the team and communicating with the project supervisor |
| Team Member | An active member of the team responsible for making the job done |
| Product owner | Responsible for product management and its quality |
| Registry | Registry contains all of the information about the publisher and subscriber’s information such as certificate |

## References

1. <https://aws.amazon.com>
2. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>

# Background, Objectives and Solution description

## Background

As we are moving to the Internet 4.0 era, Internet of things will be the next big thing for the near future. Realizing that potentiality, Amazon quickly introduced GreenGrass IoT platform to have a place in the new race for all big companies in the industry. Considering the context of this class, developing a GreenGrass-based project is a great opportunity to learn and get ready for the future. In addition, since documentation and other relevant resources are enough for development and researching purposes, it is easier to explore and put more efforts into project development.

## Objectives & solution description

Upon the completion of this project, a simplified version of enterprise message queue and simulation of traffic light shall be created. With the message queue, the system consists of the subscriber and publisher. One will send encrypted messages to the other side via a message queue configured with the help of AWS GreenGrass Core. The message queue is expected to securely transfer messages from the publisher to subscriber and report additional information to the console including timestamp and the order in which the message is sent or received. On the side of the traffic light simulation, the system needs 3 separate parts including the switch, traffic light and device shadow. The switch shall changes its status from Red to Yellow and Green for every 20 seconds. The traffic light shall also observe the the switch’s status and perform the same exact operations (changes its light from Red to Yellow and Green eventually). On the cloud, the device shadow shall display this information in real timne.

# Project Requirements and Use Cases

## Requirements

1. The system shall consists of two different parts: message queue and traffic light simulation.

2. For the first part (message queue), two end devices shall be created with connection established to the GreenGrass core: Publisher and Subscriber.

* The Publisher shall be able to send message to the message queue
* The Subscriber shall be notified by the message queue and receive the messages from the Publisher.
* All of the messages being transferred on the queue must be preserved in order (from which they are pushed to the queue)
* Timestamp shall be displayed on the console on both sides. The sender shall put the timestamp on the message, and the subscriber shall receive the message with a timestamp associated with each message.

3. For the second part of the system (traffic light simulation), two end devices shall be created with connection to GreenGrass core, API and device shadow: the Switch and the Traffic Light.

* The Switch shall be able to changes its state from Red to Yellow and evetually Green. It shall keep switching its state endlessly in that order until being stopped.
* The Traffic Light shall keep observe the Switch. As the Switch changes its state, the Traffic Light shall do the same.
* The Traffic Light shall update its state changes to the device shadow on the Cloud.
* Both the Switch and the Traffic Light change their states in every 20 seconds.

## Use Cases

|  |  |
| --- | --- |
| Use Case Name | Publisher sending a message |
| Participating Actors | Publisher and Subscriber |
| Flow of Events | |  |  | | --- | --- | | Publisher | Subscriber | | 1. Publisher enters its ready state |  | |  | 2. Subscriber enters its ready state | | 3. Publisher sends message to the queue |  | |  | 4. Subscriber retrieves the message from the queue | |
| Entry Condition | The Publisher and Subscriber are already established on two end devices. |
| Exit Condition | The Subscriber should be able to receive messages sent by the Publisher in the correct sequence. |
| Quality Requirements | The Subscriber should not wait for more than 30 seconds to receive the message once it is sent by the Publisher. |

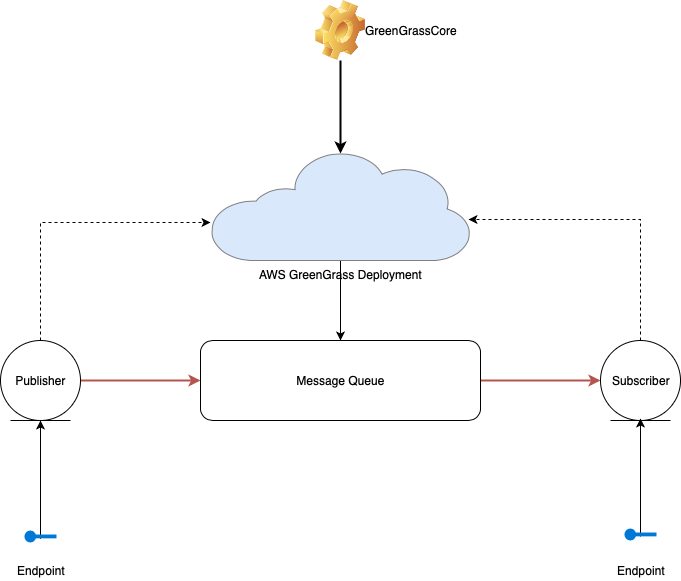
|  |  |
| --- | --- |
| Use Case Name | Traffic Light changing its light status |
| Participating Actors | The Switch, the Traffic Light and the device shadow |
| Flow of Events | |  |  | | --- | --- | | The Switch | The Traffic Light | | 1. The Switch changes its state |  | |  | 2. The Traffic Light changes its state accoring to the Switch. | | 3. The Switch is ready to changes to another state. |  | |  | 4. The information is published to the device shadow. | |
| Entry Condition | The Traffic Light, the Switch and device shadow are in ready state. |
| Exit Condition | Updated information of device state should be updated to the device shadow on the cloud. |
| Quality Requirements | The whole process for each change made to the state should happen in less than 30 seconds. |

# Architecture & High-Level Design

## System Diagram

Diagram 1 AWS IoT Architecture

## Message Queue’s High-level diagram



## Traffic Light Simulation Architecture

# Organization

## Project group

|  |  |  |
| --- | --- | --- |
| Name | Initials | Responsibility (roles) |
| Viet Nguyen | VN | Developer, Tester |
| Minh Ngo | MN | Developer, Tester |
| Hoa Nguyen | HN | Developer, Tester |

## Stakeholders

The stakeholders are listed below:

Instructor

Grader

Team members

# Development process

This project will mostly use waterfall approach for the whole development process. Due to the nature of our work, it is more efficient to define workflow from stage to stage with a clearly defined objective per each stage. By doing so, we are able to achieve better performance and finish the project on time.

# Deliverables

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| To | Output | Planned week | Late | Delivered week | Notes |
| Instructor | Project Demo | First week of December | On-time | December 6 2018 | Delivered as planned. |
| Grader | Project Report and Documentation Files. | First week of December | On-time | December 9 2018 | Delivered as planned. |

# Project Risks

|  |  |  |
| --- | --- | --- |
| Possibility | Risk | Preventive action |
| Technical Limitations of AWS | Low | Tutorials and documentation research |
| Budget | High | Develop work-around to overcome AWS limitation by free tier. |
| Hardware | Low | Borrow from friends and school club room |
| Schedule (conflict with many concurrent projects) | High | Use waterfall methodology to ensure on-time delivery |

# Communication

Method of communications:

* + - 1. Social networking service (Facebook Messenger Group)
      2. Slack
      3. Github

4. Direct group meeting in school

## Collaboration

|  |  |
| --- | --- |
| Remarks: | The group has been effectively communicated with each other well throughout the semester. No major issue has been found. |

## Git and demo

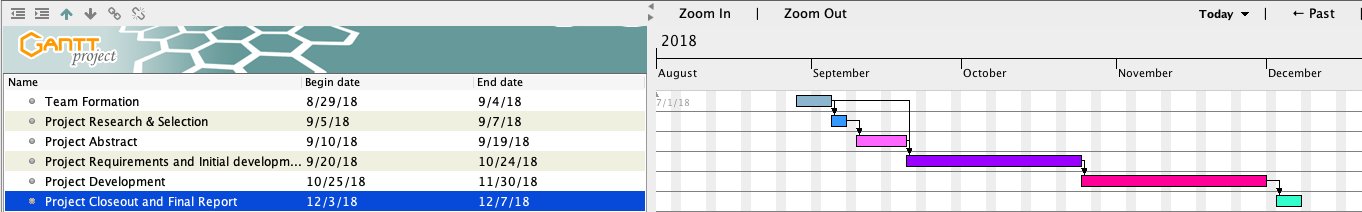
For project which requires code, they will be uploaded on this link:

<https://github.com/MrJc010/twitterapp>

<https://app.box.com/s/3lpwenr0e2nmjawjhapexj4pfwwce7oj>

# Project Plan and Workflow

## Project Gantt Chart



## Project Timeline Remark

|  |  |
| --- | --- |
| Task | Progress and remark |
| Team formation | 100%. Finished before expected end date |
| Project research and selection | 100% Finished past expected date due to project topic complications. |
| Project Abstract | 100%. Finished before expected end date. |
| Project requirements and initial development | 100%. Finished before expected end date. |
| Project Development | 100%. Finished before expected end date |
| Project closeout and final report | 100%. Finished before expected end date |

## Test Plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No. | 001 | Phase: | 1 | Author: | Viet Nguyen & Minh Ngo | Date: October 27 2018 |
| Test Category: | | Functional Testing | | | | |
| Software Product: | | Message Queue and Traffic Light simulation | | | |  |
| Test Title: | | Initial test | | | | |
| Test Purpose: | | Make sure that the project conforms to its requirements and error-free | | | | |
| Test Setup: | | Raspberry Pi 3 with GreenGrass at home network | | | | |
| Prerequisites: | | Establishing device subscription and device shadow config | | | | |
| Expected Results | | All major modules passed without errors. | | | | |
| Result: | | All passed. | | | | |
| Reason for Failure: | | None | | | | |
| Remarks: | | The test was finished successfully without any major issue. | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No. | 002 | Phase: | 2 | Author: | Viet Nguyen, Minh Ngo and Hoa Nguyen | Date: December 4 2018 |
| Test Category: | | Functional Testing | | | | |
| Software Product: | | Message Queue and Traffic Light simulation | | | |  |
| Test Title: | | Final Test | | | | |
| Test Purpose: | | Make sure that the project is ready for demo and submission | | | | |
| Test Setup: | | Raspberry Pi 3 with GreenGrass, portable router at school network | | | | |
| Prerequisites: | | Establishing device subscription & shadow config and pass the inital test plan | | | | |
| Expected Results: | | All major modules passed without errors. | | | | |
| Result: | | All passed. | | | | |
| Reason for Failure: | | None | | | | |
| Remarks: | | The test was finished successfully without any major issue. | | | | |

# References

1. <https://aws.amazon.com>

2. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>