Hammer-io IoT Extension

Project Plan

Team: sddec19-24
Client/Advisor: Lotfi Ben-Othmane
Members:

- Yussef Saleh
- Brett Wilhelm
- Matt Bechtel
- Chakib Ahlouche

Email: yussefwp@iastate.edu (team admin email)

Table of Contents

1. Problem Statement

The goal of the project is to extend the framework to support IOT applications and also to provide basic services for micro-services such as fail detection. We will continue working on a framework for DevOps for Node.JS that our client has previously developed.

2. Project Deliverables and Specifications

We have two deliverables, the first one is a Framework extension to deploy applications to raspberry pi, collect operation data and provide insight on the health of the application. Which is supposed to be delivered at the end of Spring 2019 semester. Our second deliverable is a Framework extension to support microservices development and deployment.

3. Previous Work / Literature Review

Since our project is to extend a previous senior design project we met with one of the members of the previous team to get a broad overview of the current project. This meeting was extremely informative and gave a great scope for what the project can and can't do at the moment.

Along with meeting with the previous team to discuss the state of the project we looked into existing DevOps frameworks in the marketplace. Researching what enterprise tools looked like gave insight into what the final product of our solution should be.

4. Proposed Design / System / Solutions

Proposed Design will be provided after Requirement gathering and Research period.**

5. Assessment of Proposed Solution

Time and cost efficiency are very important factors in the determining the success or failure of an early startup, using a framework that deals with all javascript deployment within in same environment can be extremely beneficial to development teams.
It is also worth mentioning that using multiple tools and frameworks to run a specific environment can have several things break and ultimately make software maintenance hard, and that would be considered a trade off as far as strengths versus weakness.

6. Validation and Acceptance Test

The requirements are currently not completely defined, and thus we cannot create a clear acceptance criteria. **

7. Project Timeline

JAN				FEB				MARCH				APRIL				MAY			
w 1	w 2	w 3	W 4	w 1	w 2	w 3	w 4	w 1	w 2	w 3	w 4	w 1	w 2	w 3	w 4	w 1	w 2	w 3	w 4
			Requirement Gathering	Requirement Gathering	Requirement Gathering	Requirement Gathering	Requirement Gathering												
						Research	Research	Research	Research										
										Framework Extension Setup	Framework Extension Setup	Framework Extension Setup	Framework Extension Setup						
										Framework Extension Setup	Framework Extension Setup	Framework Extension Setup	Framework Extension Setup						
														Demos	Demos	Demos			

8. Challenges: Risks / Feasibility Assessment, Cost Considerations

Because the actual extension is so loosely defined at this point we cannot accurately access the risk involved. **

9. Standards

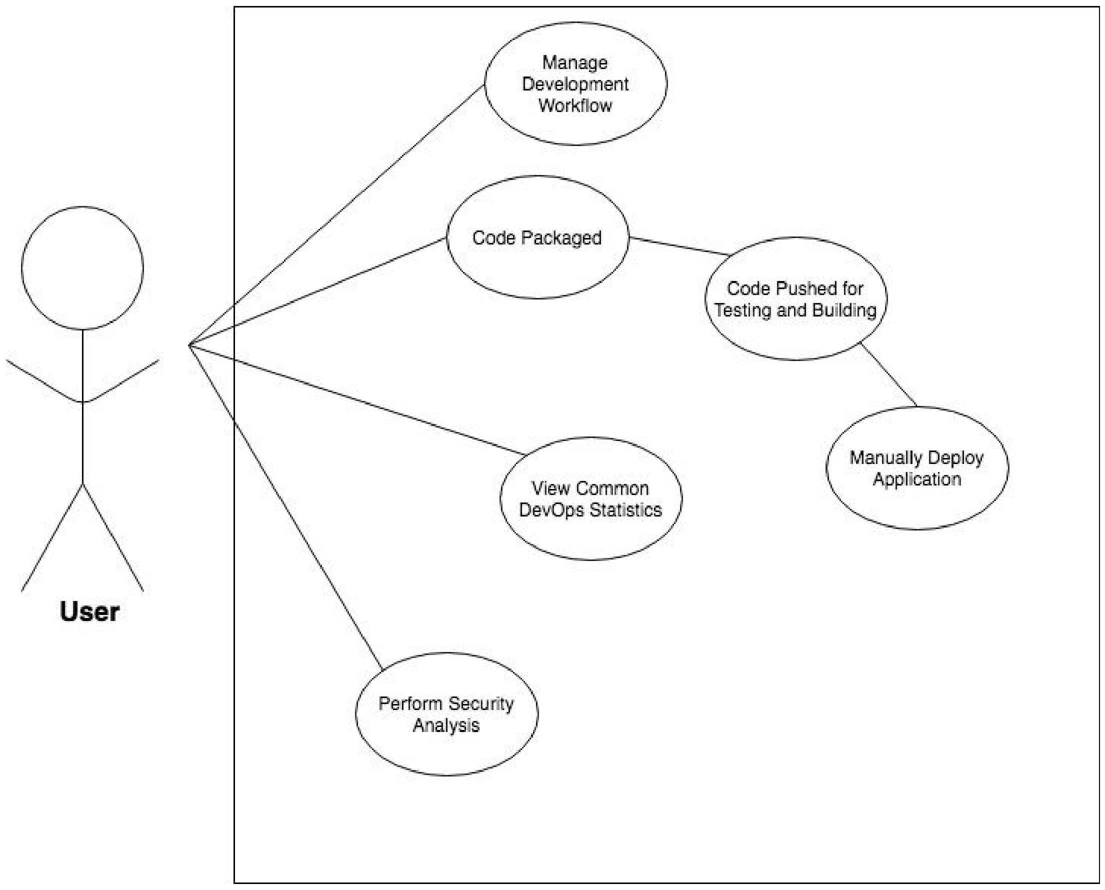
Since there is no client or use case imposed on us for this project, and most likely this frame will be used by the open source community, there should be no unethical practice done on this project.
Based on that, this project will be no threat to safety, health, or welfare of the public and to strive to comply with ethical design.

10. Test Plan

Since our project is an extension we will begin by testing the existing system. This will involve setting it up and running the existing unit tests. Then we will test all of the use cases to check for issues. Beyond that, we are unsure how to test something that we do not yet have full requirements for. **

**will be completed for the second iteration of the project plan

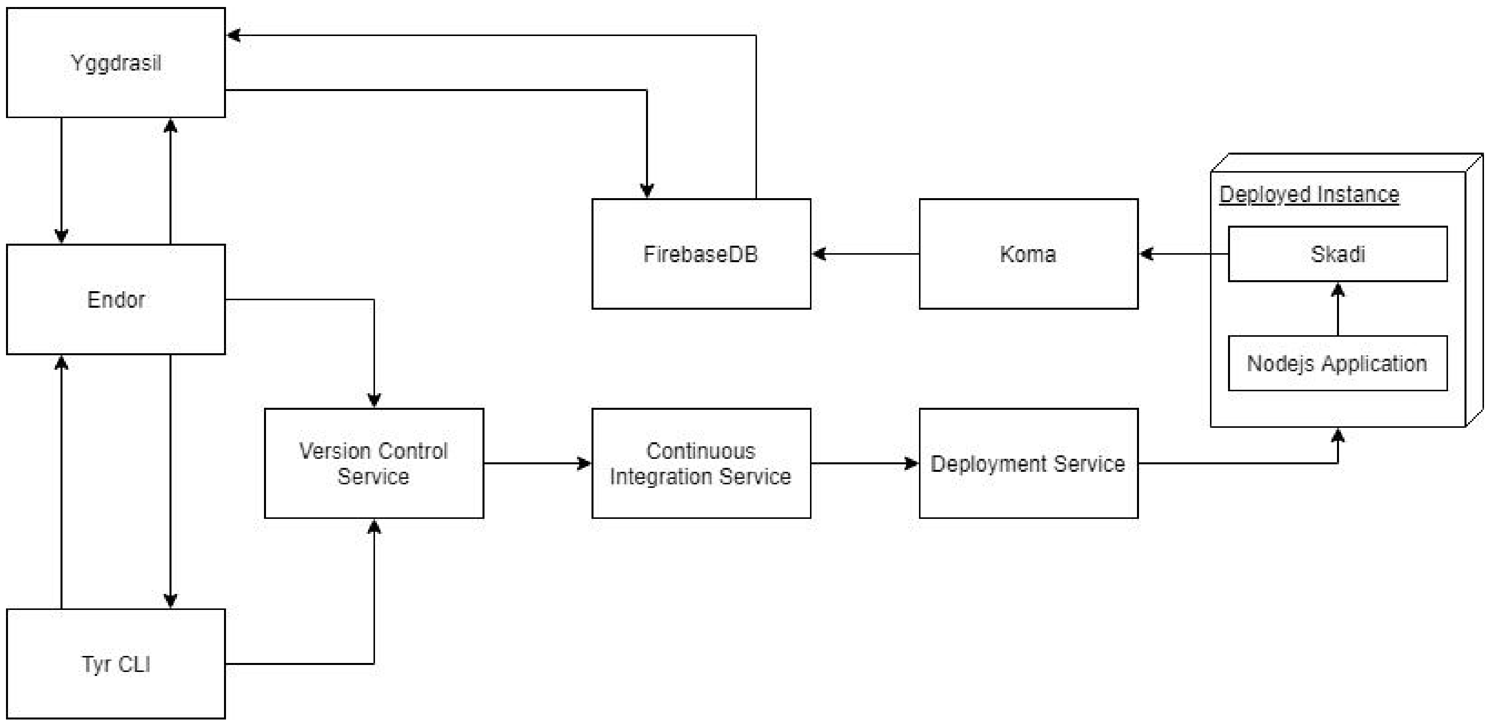
Use Case Diagram


Manage Development Workflow
Code Packaged
Code Pushed for Testing and Building
Manually Deploy Application
View Common DevOps Statistics
Perform Security Analysis
User

Use-Case Diagram Description

- **Common DevOps Statistics**
    - User can view common statistics for DevOps from the web application, Yggdrasil. These include but are not limited to:
        - Code coverage and style stats
        - Build and test failure rate
        - Status of builds
        - Task completion time
- **Manage Development Workflow**
    - All management can be done through both the command line and web application
- **Packaging, Testing, Building, and Deployment**
    - Program will be automatically packaged and sent to version control. From there, hooks will pull it into building and deployment services.

Block Diagram



Block Diagram Description

- **Tyr**
  - Command Line Interface allowing user to generate a new NodeJS project. The tool takes a configuration file which can be used to specify the services required for the project. For example, the user may want to use Github for their Version Control, Travis CI for their continuous integration, and Heroku as their deployment service. These things will be specified within the configuration file. When Tyr is ran, it will attempt to set up these services for the user and then generate the files for the project. (note: the services given in the example above are currently the only services the system is able to use).
- **Yggdrasil**
  - Front end web service for Tyr. Everything that Tyr can do can also be done through Yggdrasil, such as setup and deploy a project. Yggdrasil, beyond deploying, can allow the user to monitor and view data from their systems (via Firebase, provided by Koma). Essentially, Yggdrasil is the ‘DevOps’ hub, similar to the Amazon Web Services Management Console.

- **Endor**
  - Backend service for Yggdrasil. Endor handles all of the user information from the Yggdrasil site as well as all user requests that do not involve simply viewing the data collected from their deployed system, as that is done via Firebase and Koma. Endor is like a wrapper to Tyr that takes web requests instead of direct commands.
- **Skadi**
  - Skadi is a data collection service that is embedded in the system that the user generates with Endor or Tyr. Skadi collects the data from the user's deployed system and sends it to the data aggregation service 'Koma'.
- **Koma**
  - Koma collects the data from each deployed service, aggregates it based on DevOps user account, and sends it off to be stored in Firebase.
- **Firebase**
  - Firebase receives real-time data from Koma about the deployed instances, and stores them for retrieval by Yggdrasil. From this, users can view data about their running instances in near real-time.
- **Version Control Service ***
  - The users will push their code to their Version Control service which will have hooks in place to move the newly pushed code to the Continuous Integration service. This action, of the user pushing their code to their Version Control service, will kick of the continuous integration process.
- **Continuous Integration Service ***
  - The Continuous Integration service will take the newly pushed code from the user, build it, containerize it, and then send the container to the deployment service. It will also keep build artifacts so that user's may easily roll back their systems.
- **Deployment Service ***
  - The Deployment Service will receive the container images from the Continuous Integration service and spin them up when as requested, whether it be on each new commit or on a timed basis. The Deployment Service will also manage the running Docker containers, scaling them as specified by the the user.

* The current version of the project only allows GitHub, TravisCI, and Heroku, for Version Control, Continuous Integration, and Deployment, respectively. Part of the extension scope is to make the Version Control, Continuous Integration, and Deployment services for the user's system be configurable.