DevOps for IoT: A Hammer-io Extension

Client/Advisor: Lotfi ben-Othmane
Matt Bechtel, Brett Wilhelm, Chakib Ahlouche, Yussef Saleh

http://sddec19-24.sd.ece.iastate.edu/

What is DevOps?

- “ **‘DevOps’** is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.”

Defined by: https://aws.amazon.com/devops/what-is-devops/

What is Hammer-io?

- Previous Senior Design Project
- A DevOps framework offering an opinionated approach to developing and managing microservice applications in Node.js.
- Microservice design with 5 components:
  - UI: Yggdrasil
  - Backend Web server: Endor
  - CLI: Tyr
  - Device Data Collector: Skadi
  - Device data aggregation service: Koma

What is Hammer-io? (cont’d)

- In its current configuration, Hammer-io only allows for software deployment to a cloud service, namely ‘Heroku’. The purpose of our extension is to provide the client an alternative form of deployment, specifically deployment onto IoT Devices.

What is an IoT Device?

- The term ‘IoT’ has a broad meaning, being the acronym for ‘Internet of Things’
- IoT Devices are ‘non-standard’ computing devices that connect to a network and have the ability to transmit data
- Examples: Wearables with internet connection, smart appliances, smart meters, etc.
- IoT Device in the context of our extension:
  - Any device with internet connection that runs a form of Linux. The device may be lacking in resources, taking virtualization off of the table. Thus we cannot leverage technologies such as Docker to perform deployment.

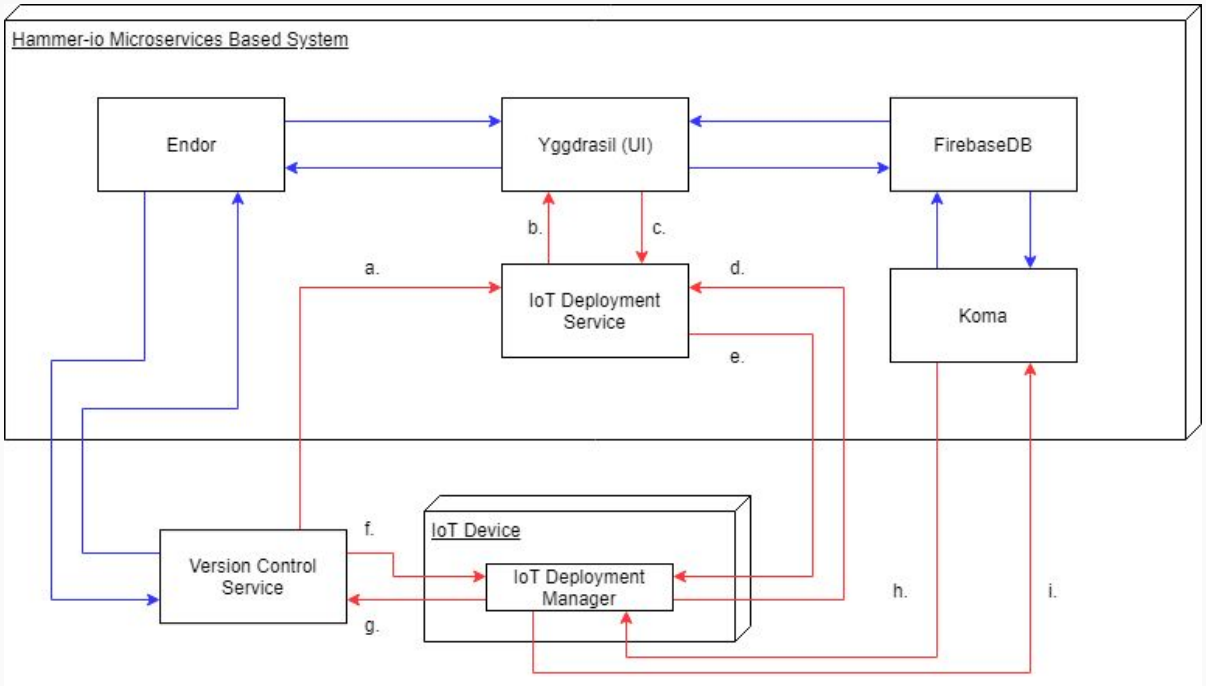
Why?

- With the use of IoT devices growing rapidly, it is important to be able to automatically deploy new builds to those devices.
- With automated deployment comes the ability to ensure that a service is always up and serving its customers.
- A ‘DevOps’ approach to this deployment is ideal because it keeps business interests in mind, while also making life easier on the developers, a ‘win-win’ situation.

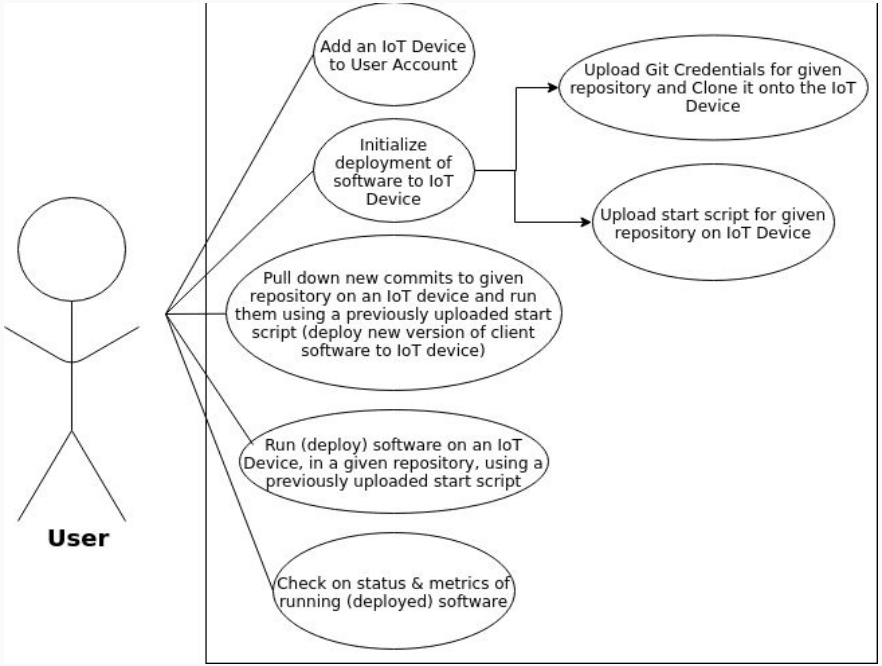
Our Plan

- How do we extend the existing Hammer-io framework to handle deployment to a client's IoT device?
  - Answer: Create an IoT Deployment Service and Manager. Service runs in the cloud and the Manager runs on the client's IoT device. Service and Manager work together to perform software deployment to the IoT device.

System Sketch: Block Diagram



System Sketch: Use Case Diagram



IoT Deployment Service: Functional Reqs

- Provides API for deployment requests
  - Initial Setup
    - Sets up initial communication with IoT instance, using the credentials provided by the client (endpoint)
    - Passes client's version control credentials and the client code's start script to our service, 'IoT Deployment Manager' via HTTPS, which is running on the client's IoT instance
    - Once the connection is successfully set up, the 'IoT Deployment Service' will set the state of the client's instance to 'Ready for Deployment'
  - Deployment
    - If the instance is 'Ready for Deployment' the 'IoT Deployment Service' will send a request to the running 'IoT Deployment Manager' to tell it to pull the code and check for updates, if there are updates, the manager will run the client's code with the given start script that was provided by the client
    - The 'IoT Deployment Service' will field a request from the 'IoT Deployment Manager' to notify the client about the state of their deployment (this notification will be passed back to the UI for client viewing)

IoT Deployment Service: Functional Reqs

- Provides API for the client to acquire feedback about a client's instance
  - Instance State
    - Information about whether the software deployed into the IoT device is running/in a healthy state
  - Deployment History
    - Information about the deployment history of the instance, this should include:
      - Deployment times
      - Commit information (version) about said deployments
      - User that executed said deployment
- Security
  - Only allow users to control deployment on their own devices. Authenticated user will only have access to the IoT Devices they have added.

IoT Deployment Service: Non-Functional Reqs

- Usability
  - The ‘IoT Deployment Service’ API should be easy to utilize and should be easily extensible if new functionality is needed
- Supportability
  - The ‘IoT Deployment Service’ should be written in NodeJS and supported by Linux systems
- Reliability
  - The service should be continuously running so that clients can control their instances with complete reliability
  - Uptime should be 99.9%, meaning that updates to it must be done in a rolling fashion

More Design Details: IoT Deployment Service

- NodeJS web server leveraging the Express framework
- Handles requests from existing UI, ‘Yggdrasil’, to facilitate software deployment to an IoT Device
- Uses existing authentication in Hammer-io to verify user has access to deploy to a given device
- Stateless and Scalable
- https://git.ece.iastate.edu/sddec19-24/iot-deployment-service

IoT Deployment Manager: Functional Reqs

- Provides API for deployment requests
  - Initial Setup
    - Controller exists to receive a request with the client's information, including git credentials and start up script for the client's code
  - Deployment
    - Controller exists to handle requests to deploy (start) software
    - Controller exists to update a given repository on the device (this will be a fetch and a merge, or as many know it, a pull)
- Security
  - Verification that incoming requests are from 1) the IoT Deployment Service and 2) from an authenticated user

IoT Deployment Manager: Non-Functional Reqs

- Usability
  - The ‘IoT Deployment Manager’ API should be easy to utilize and should be easily extensible if new functionality is needed
- Supportability
  - The ‘IoT Deployment Manager’ should be written in NodeJS and should be usable on (supported by) Linux IoT devices
- Reliability
  - Service will be continuously running on the IoT device so that clients can control their instances with complete reliability.

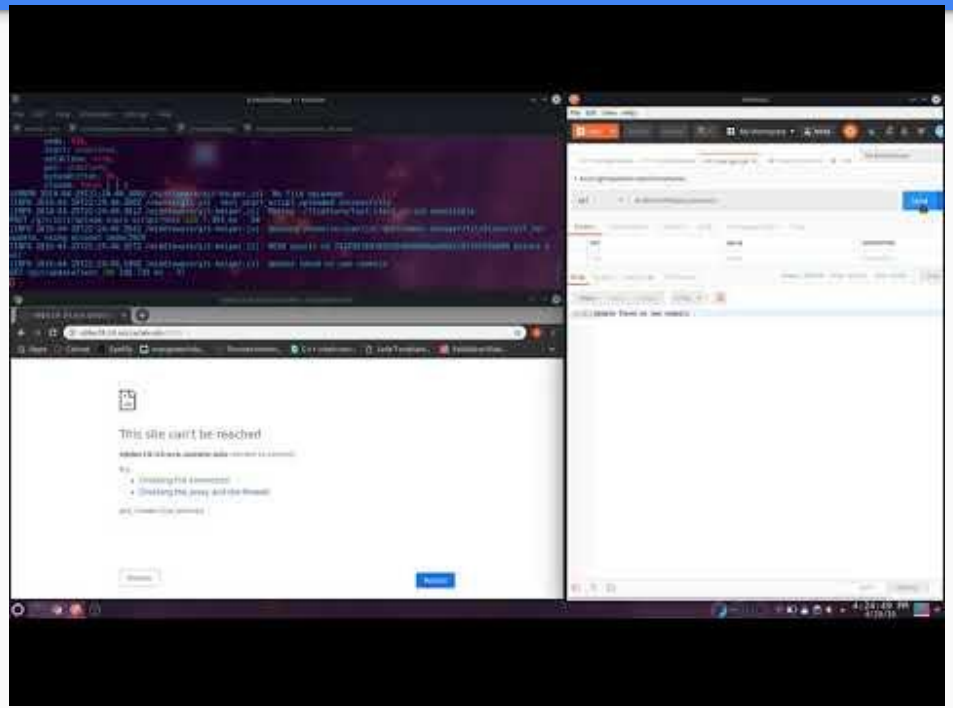
More Design Details: IoT Deployment Manager

- NodeJS web server leveraging the Express framework
- Lightweight
- Able to handle requests from the IoT Deployment Service
- Stateless
- https://git.ece.iastate.edu/sddec19-24/iot-deployment-manager

Current Extension Functionality

- As of now you can use the IoT Deployment Service and Manager to:
  - Clone an existing repository from its remote to the IoT Device
  - Pull down new commits from repository (update)
  - Upload a start script for a given repository
  - Run code in a given repository on the IoT Device

Demo

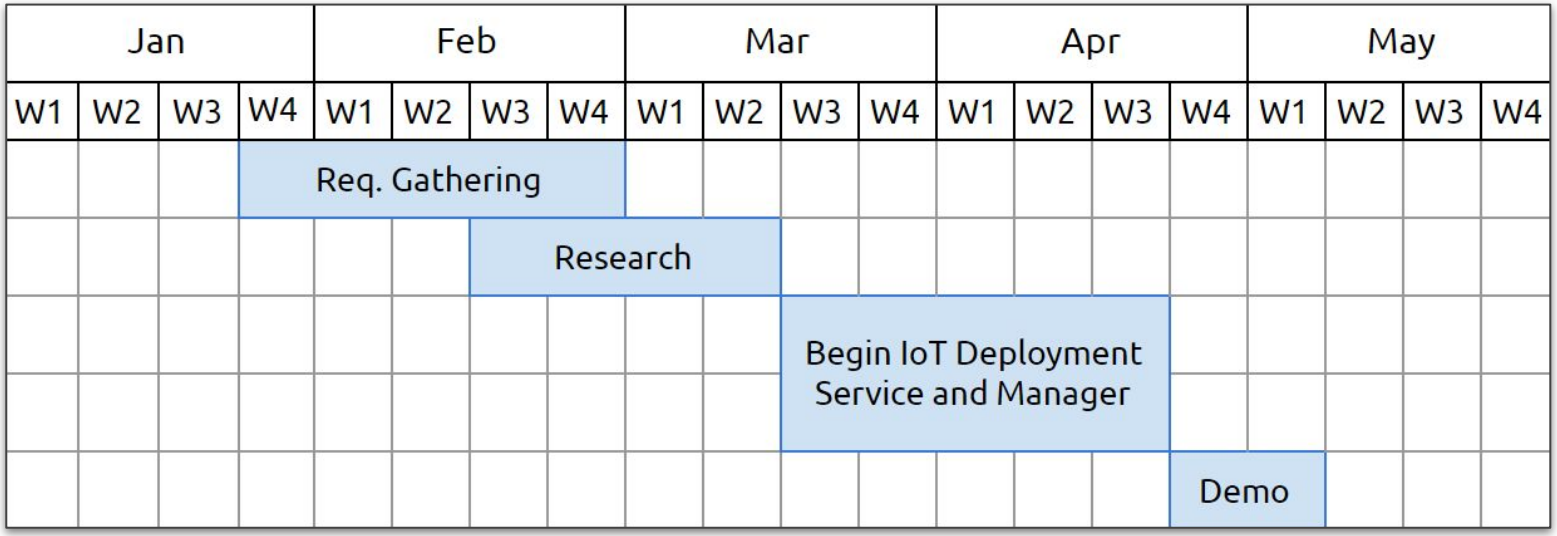


Current Project Status

- Design is complete*
- Development has begun on the Deployment Service and Manager, and both “function”

* Tentative because of our agile development strategy

Project Schedule (First Semester)



Test Plan

- Manual
- Unit
- Regression
- Integration

Risks With Our Extension

- Integrating the new components, ‘IoT Deployment Service’ and ‘IoT Deployment Manager’ may take longer and be more challenging than expected
  - Chance of Occuring: Medium
- The scope of our extension is too broad
  - Chance of Occuring: Medium
- The knowledge level of the team in the given technologies may lead to slow development
  - Chance of Occuring: Low

Is This Extension Feasible?

We believe so, based on our assessment of the problem and the solution we have come up with. We have gotten a pretty good start on the development of our two new components and feel confident that, with more experience going into next semester, we can achieve our goal.

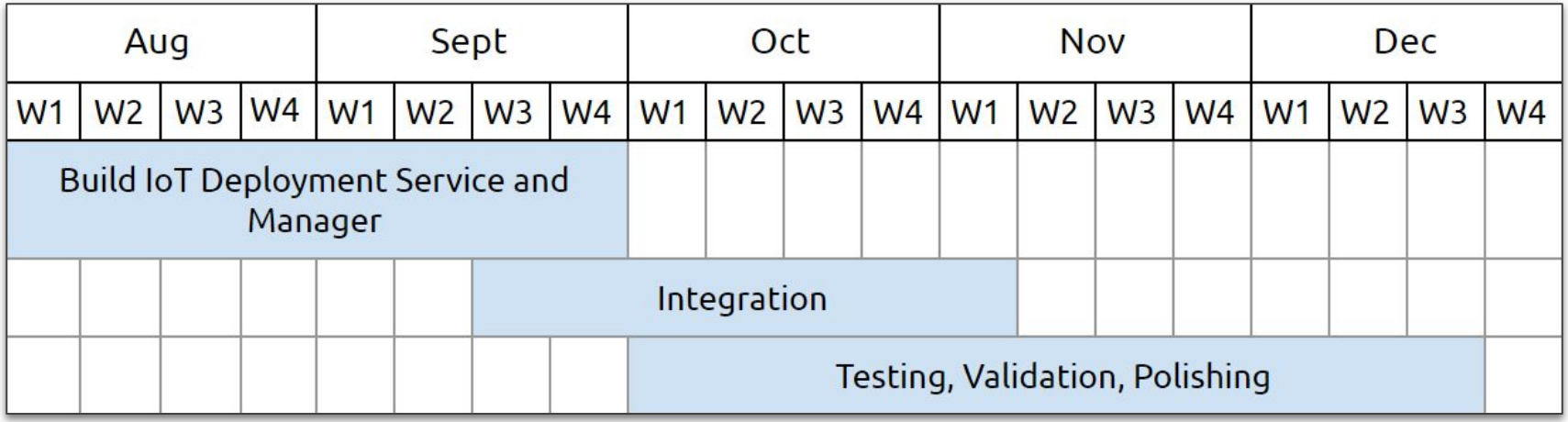
Resource/Cost Estimate

- Cost will be low
  - Only cost will be for hosting of the Hammer-io system, which will scale based on user traffic. All software used will be open source.

Plan For Next Semester

- Continue development of the IoT Deployment Service and Manager
- Integrate our new components with Hammer-io
- After integration with Hammer-io, we will execute our Testing Plan.

Project Schedule (Second Semester)



Contributions

- Yussef
  - Integration Research and planning
  - Preparing Testing requirements
- Matt
  - Led design of ‘Deployment Service’ and ‘Deployment Manager’
  - Built what we have now, the Service and the Manager
- Brett
  - Integration Research
- Chakib
  - Research and work on design of ‘Deployment Service’

References

- Github: https://developer.github.com/v4/
- Hammer-IO: http://sdmay18-19.sd.ece.iastate.edu/docs/
- Mocha: https://mochajs.org/api/mocha/
- NodeJS: https://nodejs.org/en/docs/
- NPM: https://docs.npmjs.com/
- Docker: https://docs.docker.com/
- NodeGit: https://www.nodegit.org/api/
- Original Hammer-io project plan: https://hammer-io.github.io/docs/Project_Plan_v3.pdf

Questions?