# Configuration Management

A big part of Configuration Management includes identification of the project’s configuration items, and a description of our version and change control process, which are all included in the following sections.

## Configuration Items

Below is the itemization of all our project assets and their location (all gems are **defined** in the Gemfile and are only installed when calling bundle install)

**Environment**

* Docker Engine v20.10.17
* Docker container v4.12.0
* Heroku

**Program**

* Ruby 3.0.2p107
* Rails 6.1.4.1
* Rspec-rails 5.1.2
* PostgreSQL 13.3
* Devise 4.8
* Oauth2 2.0.9
* Omniauth 2.1.0
* Omniauth-oauth2 1.8.0
* Omniauth-rails\_csrf\_protection 1.0.1
* Capybara 3.37.1
* Faker 2.23.0

**Tools**

* GitHub
* RuboCop 1.36.0
* Simplecov 0.21.2
* Brakeman 5.3.1
* Jira
* Visual Studio Code

**Document**

* Data Design Final Version (located in Teams thru ours teams channel -> Files -> Turnover (Project Turnover) -> Documents -> APPSO\_DataDesign.docx)
* Figma v9.0
* User Manual (Final version): <https://docs.google.com/document/d/1mKVl6haz5PUv8I0OmKrQSLZu_gtnL2Ah1lyAzhNzpsw/edit?usp=sharing>

**Data**

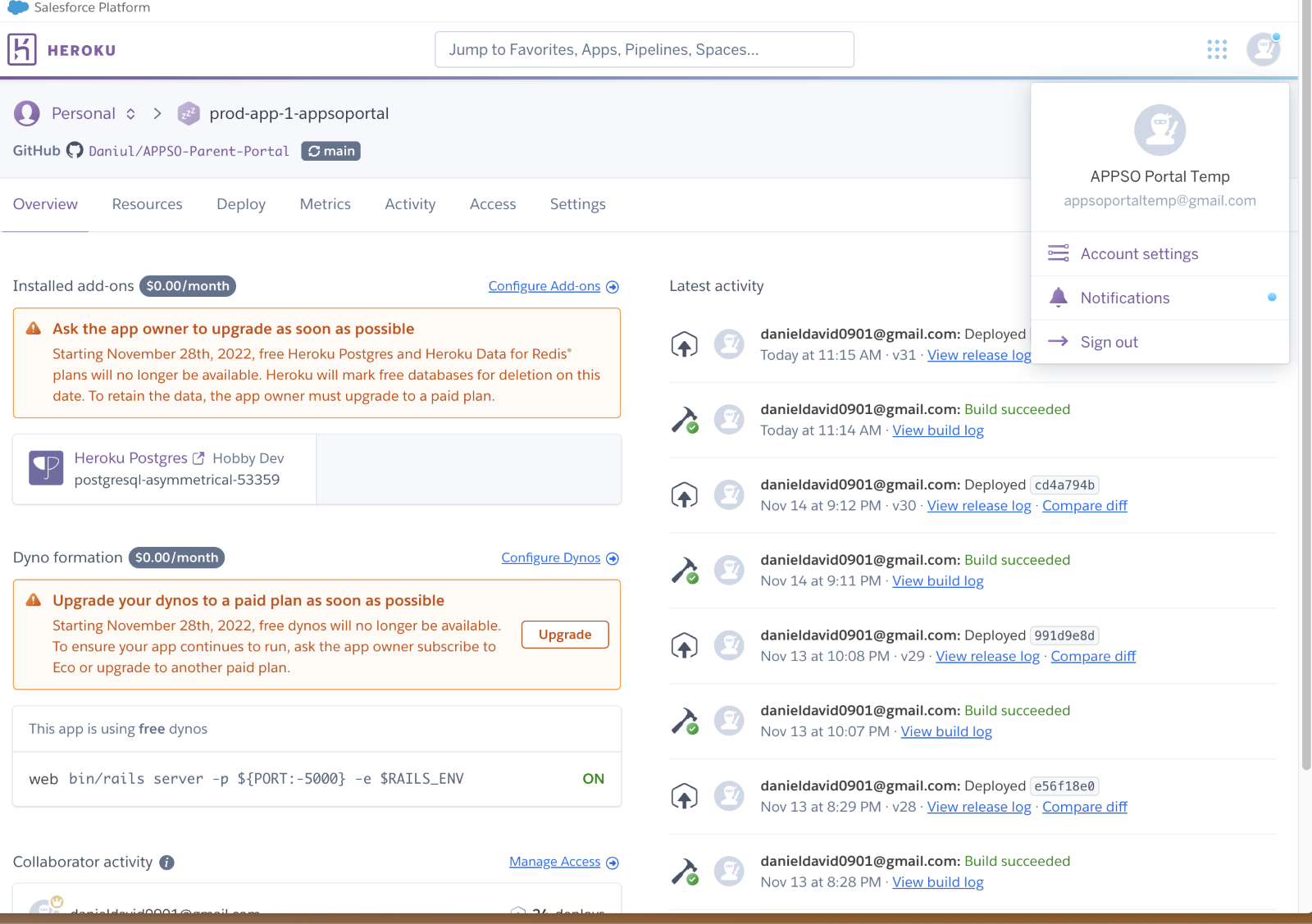
* Final Test Data is located in the GitHub (<https://github.com/FA22-CSCE431-classroom-8am/sprint-3-appso-parent-portal/tree/main/spec>).
  + By following the link above:
  + Integration test Data is in feature/integration\_spec.rb
  + Unit Test Data is in unit\_spec.rb

Our code is maintained in a classroom GitHub Repository, with the latest code in the ‘Main branch.’ See more information in the section below on “Version Control.”

<https://github.com/FA22-CSCE431-classroom-8am/sprint-3-appso-parent-portal>

Our product and sprint backlog can be found in Jira, with organization name N/A (no organization name) and project name Parent Portal

Our application is deployed in our Heroku account <https://stage-appso-portal.herokuapp.com> and in the customer’s Heroku account



In our MS Teams repository, subfolder “Turnover/Documents” are the following:

* Sprint 1 report v2 (latest)
* Sprint 2 report v3 (latest)
* Sprint 3 report v2 (latest)
* Project Turnover report v2 (latest)
* Scope v3 (latest)
* Team Working Agreement v1 (latest)
* Data Design v4 (latest)
* Risk Plan v2(latest)
* Security Risk Plan v2 (latest)
* UAT Form v3 (latest)
* Configuration Item Identification v1 (latest)
* Application Show & Tell video v1 (latest)
* Reflection video v1 (latest) and corresponding slide deck v1
* UAT Results
* Customer Feedback Results
* User/Admin/System Documentation (all-in-one document) v2 (Latest)
* Efficiency Summary v1 (latest)
* FK to PK Connection Diagram v1 (latest)

## Version Control

In software engineering it is necessary to track and control the source code developed. In our project, we decided to use GitHub, which offers functionalities, such as integrated issue tracking, collaborative code review, team management, and highlighting of syntax. It allowed us to work on separate features of the application, track bugs, and manage coding tasks.

All team members created separate, feature branches corresponding to the user stories that we were assigned. We made sure to only work in our respective branches. When one member was done implementing a user story, they created a pull request. The pull request includes Acceptance Criteria and Definition of Done, both of which needed to be completed for the feature to be officially done from a customer and developer standpoint. An additional team member was required to conduct a code review of the changes made to the branch, and check that there are no merge conflicts. GitHub CI/CD checks and unit/integration tests were also reviewed to ensured that they all passed. If there was a merge conflict, a more extensive code review is done with one team member and the author to ensure that no expensive overwritten will occur. The process of extensively using GitHub to conduct Continuous Integration and Continuous Delivery, along with maintain Version Control allowed us to have organization in terms of the code we wrote. It felt more professional and organized when we utilized GitHub.

Included in the classroom GitHub is a ReadMe file describing the following:

-Requirements

-External Dependencies

-Installation

-Tests

-Execute your code

-Environmental Variables/Files

-Deployment

-CI / CD process, including description of GitHub actions code

-Support

## Change Control process

Change control involves controlling changes to the baseline of the project which includes software, data, and documentation.

Whenever there someone requested to make changes to either the database design or the code itself, they were required to tell the whole team (not just the product owner), regarding the proposed changes to the target documentation. However, the product owner has the final say and checks to make sure the proposed changes do not seriously affect our project deliverables. If there was a scenario where it did change one of the deliverables, the product owner would then contact the customer representative and engage in discussion regarding the proposed changes. Essentially there was a loose chain of command, where everyone can listen and add input to changes a member wants to make on the baseline of the project, and then the product owner will then make an informed decision after the group discussion.