

Developing the LaunchIT REST API

Kailey Wong, Gabriel Maxfield

LC DevOps Internship Team (DO-IT)

Mentors: Otto Venezuela, Zeke Morton, Matthew Ramirez



Meet the team!



- Kailey Wong
- 3rd year @ UC San Diego
- Computer Science Major
- Mathematics and Music Minors



- Gabe Maxfield
- Senior at BYU
- Computer Science Major
- Mathematics Minor

What is LaunchIT?

- User-facing web application
 - started as an internship project for 2021 DO-IT
 - continued by WEG team + DevOps Interns
- Persistent Data Services (PDS)
 - databases, message broker/queueing systems, object storage
- Why? users couldn't instantaneously allocate PDS resources
- LaunchIT 2.0 (2022 DO-IT)
 - OpenShift: Extended resource deployment management
 - StorageGRID: Implemented admin and user level management



REST API - Introduction

- Problem: LaunchIT frontend, server, API wrapper, and backend are tightly coupled
- Goal: develop a new scalable + independent RESTful API module for LaunchIT
- Tech Stack
 - API Layer: OpenShift, StorageGRID
 - Backend: Flask, Python
 - Documentation: Swagger (OpenAPI)



Authentication & Endpoints

- Authentication

- Wrapper decorator handles authentication before each request
- 2 options
 - LaunchIT GUI login
 - Use authentication endpoint to retrieve token



- Endpoints

- GET requests: read
- POST requests: create/update
- DELETE requests
- Leverage pre-existing OpenShift, StorageGRID, client wrapper API modules
 - Parse and return desired data as JSON
 - New “whoami” OpenShift function to retrieve authenticated user



Swagger Documentation

- Allocations
- Authentication
- Buckets
- Catalog
- Resources
- Workspaces

Allocations			
DELETE	/api/allocation	Deletes the specified allocation	delete_api_allocation
POST	/api/allocation	Create an allocation	post_api_allocation
GET	/api/allocations	Get all allocations for a user	get_api_allocations
POST	/api/allocations/s3reset	Reset s3 keys	post_api_allocations_s3reset
GET	/api/allocations/{allocation}	Get a specific allocation for a user	get_api_allocations_allocation_
Authentication			
POST	/api/auth	Authentication endpoint for API	post_api_auth
Buckets			
DELETE	/api/bucket	Deletes the specified bucket	delete_api_bucket
POST	/api/bucket	Create a bucket	post_api_bucket
Catalog			
GET	/api/catalog	Gets available instance types in catalog	get_api_catalog
GET	/api/catalog/{instance}	Gets template arguments for specified instance	get_api_catalog_instance_
Resources			
DELETE	/api/resources	Delete a resource in the specified workspace	delete_api_resources
GET	/api/resources/config/{workspace}/{resource}/	Gets config file(s) for resource in specified workspace	get_api_resources_config_workspace_resource_
GET	/api/resources/config/{workspace}/{resource}/ {config_type}	Gets config file(s) for resource in specified workspace	get_api_resources_config_workspace__resource__config_type_
POST	/api/resources/resetpassword	Reset a password for an instance	post_api_resources_resetpassword
POST	/api/resources/restart	Restart an instance	post_api_resources_restart
GET	/api/resources/{workspace}	Gets resources in a workspace	get_api_resources_workspace_
GET	/api/resources/{workspace}/{resource}	Gets resource information in a workspace	get_api_resources_workspace__resource_
Workspaces			
DELETE	/api/workspaces	Deletes the specified workspace	delete_api_workspaces
GET	/api/workspaces	Get all workspaces for a user	get_api_workspaces
POST	/api/workspaces	Create a workspace	post_api_workspaces
DELETE	/api/workspaces/user	Delete a user from a workspace	delete_api_workspaces_user
POST	/api/workspaces/user	Add a user to workspace	post_api_workspaces_user

Security/Business Logic & Logging

- Security & Business Logic



- Handling errors/invalid permissions gracefully
 - Ensuring returned errors don't expose too much information
- Not allowing actions that are disabled through the GUI
 - Prevent users from adding other users to their personal project
 - Prevent users from deleting their personal project
 - Limiting project creation per user
 - Limiting allocation creation to correspond with personal workspace name

- Logging

splunk>

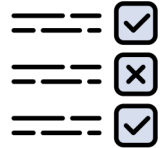
- Leveraged pre-existing logging class to log requests
- Logs are saved to Splunk, a Mongo database, & Mattermost



Testing & Looking Ahead

- CI Testing

- pytest + python requests against pre-existing Selenium test instance
- Authenticate as LaunchIT service account
- GUI vs. API Issues
 - Session configured for LaunchIT application session
 - Updating session parameters after API initialization



- Looking Ahead

- Integration with refactored LaunchIT frontend + server layer
- Separate logging for LaunchIT GUI and API

Integrating FireWorks with LaunchIT

Urwah Mir
DevOps Internship Team

August 17, 2023



Intro to LaunchIT and FireWorks

- LaunchIT

- Web application that allows users to allocate Persistent Data Services (PDS)
- PDS: databases, message broker/queueing systems, and object storage.
- Various workflow tools rely on underlying databases to store/manage data sets





- FireWorks

- Open-source workflow organizational tool
- Employs a MongoDB backend and a built-in web interface
- More here: <https://materialsproject.github.io/fireworks/>



Updating the FireWorks Template


- LaunchIT leverages OpenShift Templates to spin up PDS
 - Templates define a set of objects that can be parameterized
- Creates objects based on user parameters
- FireWorks uniquely creates two deployments
 - MongoDB Pod: Containerizes a MongoDB
 - FireWorks Pod: Manages two containers
 - FireWorks web interface to allow for ease of monitoring
 - FireWorks Proxy for authentication and authorization

	Name	Status	Resource Type
	test-fireworks	Running	fireworks
	test-mongo	Running	fireworks

Modifying LaunchIT

- Prior to FireWorks, LaunchIT served one-deployment resources
- Deletion and Configuration Display had to be modified to work with FireWorks
- Tech Stack:
 - Front-end: HTML5, CSS3
 - API Layer: OpenShift
 - Back-end: Flask, Python

Resource: mongotest
Status: Running

Connection Details: 

service-port:


service-host:

database-name:

database-password:

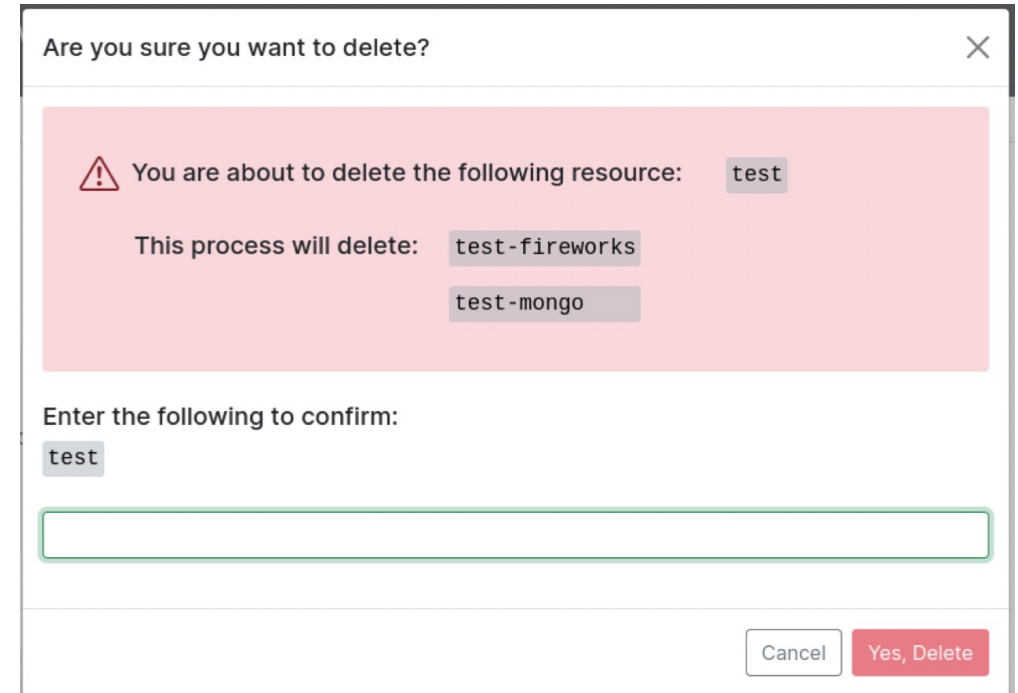
database-user:

Config Files: [mongorc.js](#) [Connection String](#) [config.py](#) [JSON](#) [YAML](#)


Name	Status	Resource Type
 mongotest	Running	mongodb

Modifying LaunchIT: Deletion

- Problem: Deleting a resource would not take down both FireWorks deployments
- Solution: Find and delete template instantiation instead of individual deployments
 - Used OpenShift wrapper API to create requests to TemplateInstance endpoint
- Made necessary front-end changes to inform users when deleting instances



Are you sure you want to delete?

 You are about to delete the following resource: `test`



This process will delete: `test-fireworks`
`test-mongo`

Enter the following to confirm:
`test`

Modifying LaunchIT: Configurations

- FireWorks has 2 services and 1 route
 - Service for MongoDB backend
 - Service for FireWorks web interface
 - Route to web interface
- Used specific naming conventions to distinguish between the two
- Problem: Needed to display the route to FireWorks web interface
- Solution: Create a function to fetch data from OpenShift's Route endpoint

Resource: test-fireworks
Status: Running

Connection Details:  

web-url:

service-host:

database-name:

database-password:

database-user:

Config Files:

Challenges and Future Plans

- Iterative Development: Revised template after LaunchIT integration challenges
- Scalability: Integrated FireWorks with LaunchIt for seamless additions of future workflow resources without significant changes
- Ongoing Testing: Continuously evaluating FireWorks with LaunchIt, uncovering and addressing edge cases



Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

ViewIt Web Application

Phoebe Schwartz

LC DevOps Internship Team (DOIT)

Otto Venezuela, Zeke Morton, Matthew Ramirez



About Me

- Phoebe Schwartz
- Current senior at the University of Wisconsin – Madison
- Studying Computer Science and Environmental Studies



The Problem

- LaunchIT
 - DOIT Internship project in 2021
 - PDS – Persistent Data Services
 - Collects data from users and stores it into a MongoDB
 - Create instance request
 - Delete request
 - Password reset request

- Solution: ViewIt!

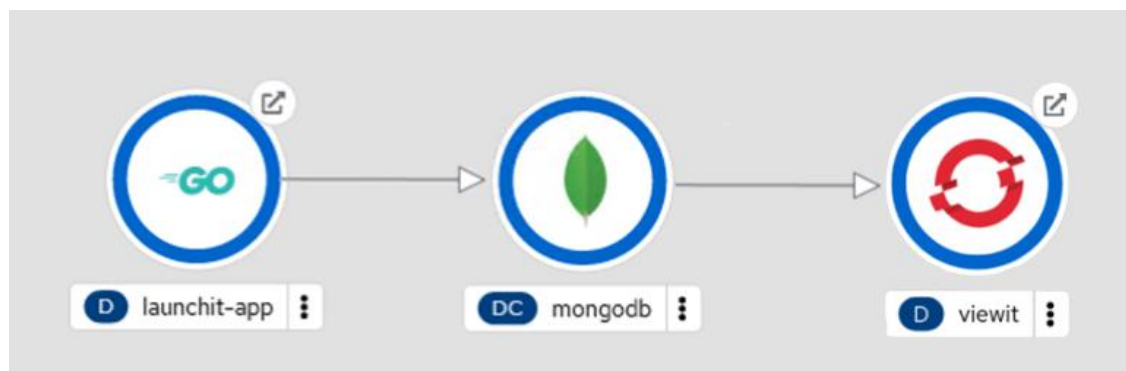
LaunchIT 

Easy PDS Deployment

LaunchIT is a web application that provides a quick and easy way to deploy Persistent Data Services (PDS).

What is ViewIt?

- A web application to easily view the request data collected from LaunchIT
- Multiple pages to help users filter what data they want to see
- Deployed in OpenShift alongside LaunchIT



How ViewIt Was Made

- Python and Flask



- Plotly



- HTML/CSS



- Pandas

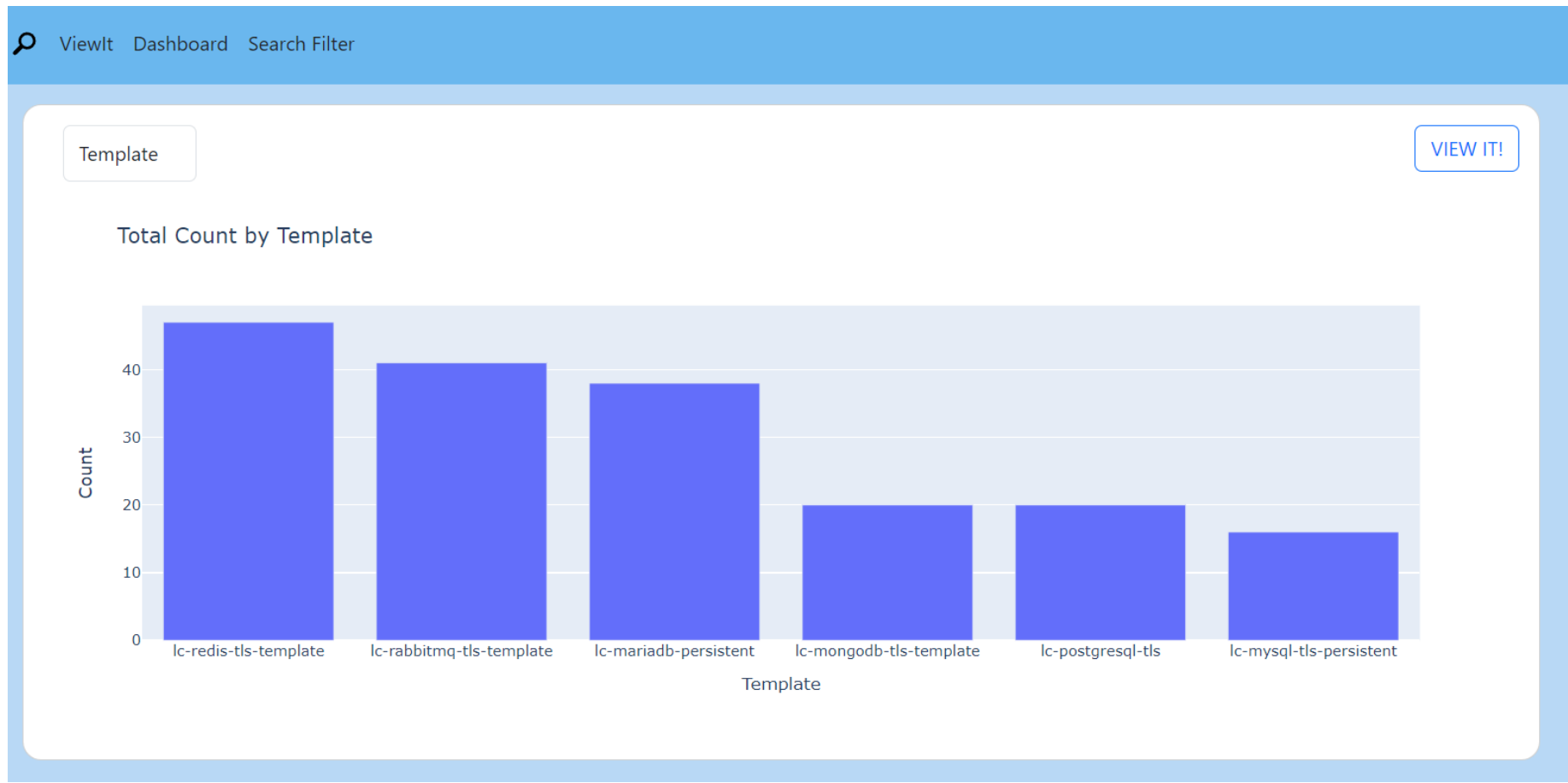


- MongoDB



mongoDB.

Dashboard

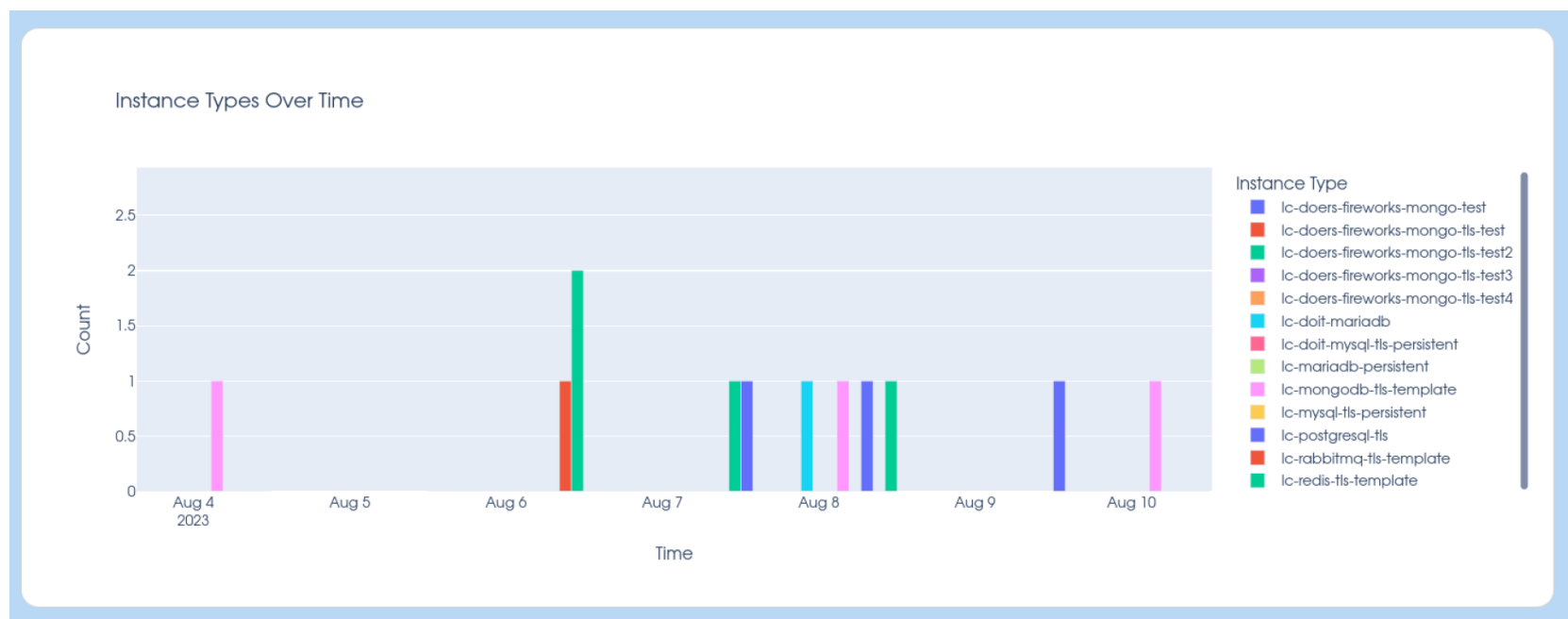


Filter Search

Instance Counter All Time All Enter Username Enter namespace

[VIEW IT!](#)

- All Time
- 1 month
- 3 months
- 6 months
- 1 year



Problem Areas

- Adjusting the Pandas DataFrame to work with Plotly
 - Resample DataFrame to count different templates by date

- HTML
 - Connecting backend variables to display on the frontend
 - Layout of elements on pages

What More Can Be Done?

- An overall filter to be able to specify what kind of data the user wants to visualize (Deletions, creations, failures)
- More information shown with hover option on graphs
- Update the front end with more user-friendly options



**Lawrence Livermore
National Laboratory**