



Zowe Desktop Administration Application

Open Mainframe Project Mentorship Program 2020

Name: Yash Sharma

Undergraduate Student at IIT Kharagpur

Mentor : Sean Grady

About Me

Name - Yash Sharma

University - IIT Kharagpur

Major - Integrated BS-MS in Mathematics and Computing

Github - [yashrsharma44](#)

Blog - [Medium](#)

Timezone - IST (UTC +5:30)

Contact Email - yashrsharma44@gmail.com


Why would you like to execute this particular project and why would you be the best individual to do so ?

I believe that this project would be really beneficial to the Zowe community, considering that Zowe helps in providing a software to control the mainframe virtually. While Zowe has an awesome architecture where all the microservices are working independently, along with different plugins and applications, there is a need to track them globally. A Zowe administrator should be able to track all the plugins and perform REST actions so that he / she has the freedom to control the scope and internal workings of the plugins and applications.

Through Zowe provides API endpoints to access the data for each plugin and application, there does not exist a GUI for doing the same. Through my project, a Zowe admin can perform the same task as we enlisted earlier, and use the GUI to achieve that.

I believe that I can achieve the goals of the project, because of the following reasons -

Experience with Front-End - I have fair experience with using React and currently learning AngularJS. I had a talk with my mentor, and we mutually decided to use AngularJS.



Interested in learning about mainframe - I have been exploring DevOps, and through Zowe, I came to know about the utility of mainframes in large companies. I was really excited to try out my hand with mainframe, and this project provides me the right opportunity to use my front-end skills along with learning mainframe.

Academic / Industry / Open Source Development

I have attached a [link](#) to my resume, which enlists all the internships and open source projects.

Open Source Contribution

I have been an avid open source contributor since my freshman year. I have been a part of a college society, Kharagpur Open Source Society ([KOSS](#)) where we spread the culture of Open Source Development. I have been a part of numerous workshops for teaching Python, Git and GitHub, JavaScript and general workshops teaching how to contribute to opensource projects.

I have been a participant in Google Summer of Code, 2019. [Here](#) is the link that explains my work for the previous summer that I worked in.

I had participated in Kharagpur Winter of Code, and [here](#) is my description of the work that I did during the winters of my freshman year.

I am also an active developer, and you can track my progress in GitHub through my github id - [yashrsharma44](#)



Development Methodology


For development purposes, I would use Agile Methodology to complete my project. Here are some features -

- Breaking of the project into sub-parts, and completing each part within a time bound time-frame
- Creating a Work in Progress PR, where I would commit my changes for each week, for each sub-task
- I would be regularly documenting my progress for each week, and make sure that it can be directly used in production
- I would make sure that each patch would be supported with test-suites which would contain the test cases for each changes and components that I have created. This would make sure that the previous work is solid enough for being used for further use
- I would be writing blog-posts to explain / blogging my progress for each week. This would make sure that the mentor can track my progress and provide me suggestions if I go off track anytime

Project Overview

Abstract

Zowe™ is an open source project created to host technologies that benefit the IBM Z platform for all members of the Z community, including Integrated Software Vendors (ISVs), System Integrators, and z/OS consumers. Zowe, like Mac or Windows, comes with a set of APIs and OS capabilities that applications build on and also includes some applications out of the box. Zowe offers modern interfaces to interact with z/OS and allows you to work with z/OS in a way that is similar to what you experience on cloud platforms today. You can use these interfaces as delivered or through plug-ins and extensions that are created by clients or third-party vendors.



Zowe helps in bridging the community gap of mainframe and system engineers and helps in providing an interface for the web developers to interact with the mainframe machine. Zowe achieves the same through numerous components, which are enlisted below -

Zowe Application Framework

Zowe provides a web user interface (UI) that provides a virtual desktop containing a number of apps allowing access to z/OS function. Base Zowe includes apps for traditional access such as a 3270 terminal and a VT Terminal, as well as an editor and explorers for working with JES, MVS Data Sets and Unix System Services.

z/OS Services

Provides a range of APIs for the management of z/OS JES jobs and MVS data set services.

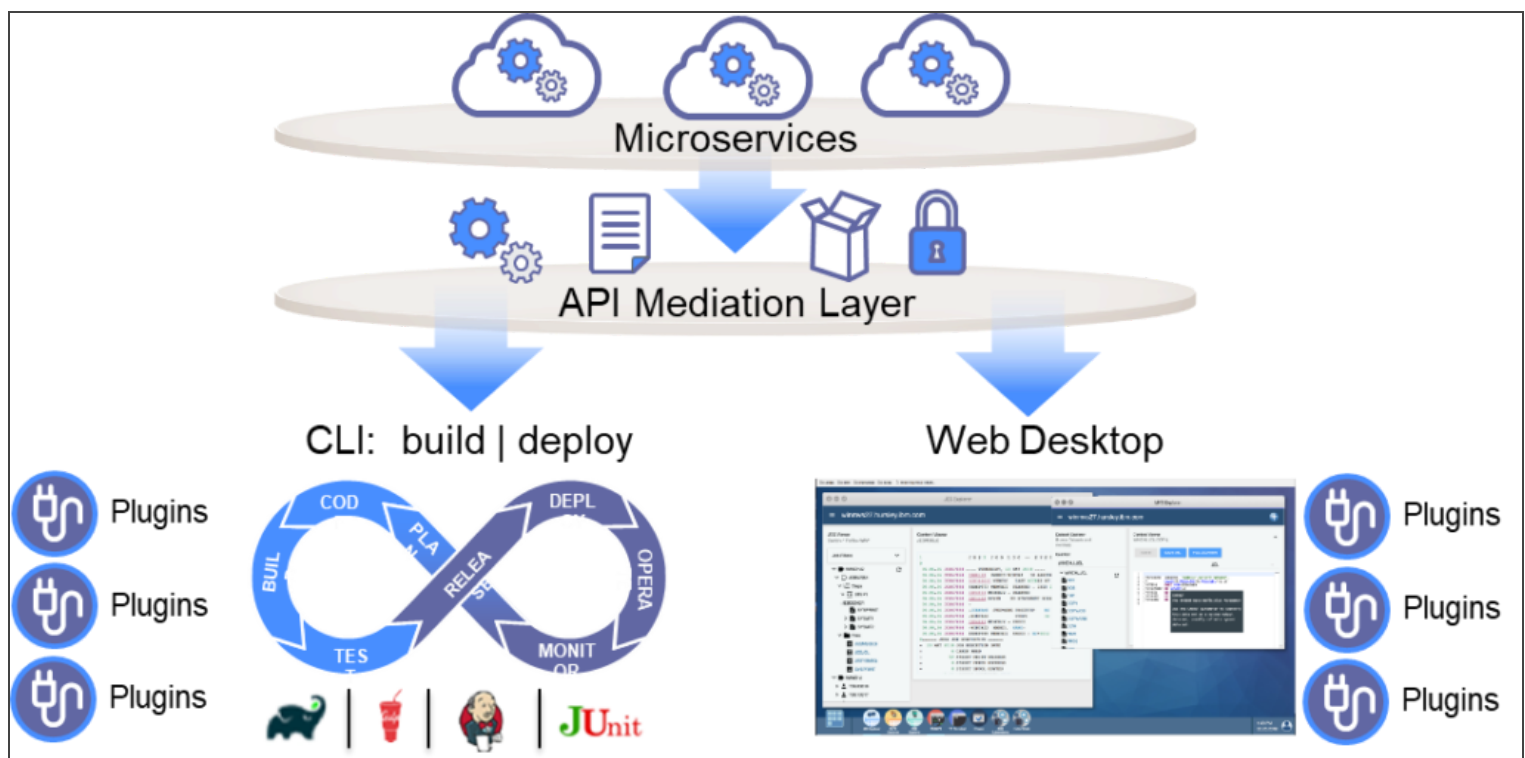
Zowe CLI

Zowe CLI is a command-line interface that lets application developers interact with the mainframe in a familiar, off-platform format. Zowe CLI helps to increase overall productivity, reduce the learning curve for developing mainframe applications, and exploit the ease-of-use of off-platform tools. Zowe CLI lets application developers use common tools such as Integrated Development Environments (IDEs), shell commands, bash scripts, and build tools for mainframe development. It provides a set of utilities and services for application developers that want to become efficient in supporting and building z/OS applications quickly.


API Mediation Layer

Provides a gateway that acts as a reverse proxy for z/OS services, together with a catalog of REST APIs and a dynamic discovery capability. Base Zowe provides core services for working with MVS Data Sets, JES, as well as working with z/OSMF REST APIs. The API Mediation Layer also provides a framework for Single Sign On (SSO).

Below here is a diagram that provides the architecture of Zowe and how different microservices interact with each other -



The entire Zowe architecture is broken down into microservices. Each microservices performs a specific role, which makes this software robust and easy to scale.



At the top of the hierarchy, there are numerous microservices at the top, which are installed in the mainframe. These provide all backend support for the Zowe services. There is an API mediation layer, which helps in routing through all the requests through one layer. It is a reverse proxy which is also used by the individual microservices to send and communicate with each other.

The user interface for the consumption by the client is broken into two main parts -

1. ***Command Line Interface***
2. ***Graphical User Interface***

The CLI helps the users in using Zowe through the command line itself. CLI is universally accepted as a versatile source for running commands. It provides intuitive commands for the users which helps in getting hold of the mainframe through the command line itself.

The CLI also provides plugins for the modern IDEs such as Visual Studio Code, which helps the users getting work done through the IDE itself.

The GUI also known as the Zowe Desktop Application is another interface for the users who are much more comfortable in controlling the mainframe through the Desktop interface. The desktop interface is provided through the frameworks of JavaScript. The application is run as a web application which is powered through NodeJS and uses ExpressJS as a framework. The frontend GUI uses AngularJS for breaking down the components of the GUI into logical parts.

The Desktop Application can be extended through use of plugins, which can be added in the Desktop for delegating different sub-tasks. 3rd party plugins can also be installed, which helps in extending Zowe for different purposes. Zowe, by default provides some plugins such as 3270 terminal and a VT Terminal, as well as an editor and explorers for working with JES, MVS Data Sets and Unix System Services.

Plan to approach the project

The goal of this mentorship program is to implement a Desktop Administration Manager. Zowe has numerous plugins which have different scope of usage. Many plugins can access different parts of the mainframe, which depending on the scope of the plugin might be useful. However, there might be plugins which are scoped upto a certain level, and providing access to them might leak some sensitive data about others. Fortunately, Zowe provides Authentication and Authorization to scope out the user different user level. However, Zowe only provides a list of API endpoints which help in accessing the different data about the users and the plugins, which might help us in tracking people and application, and what level of access each of them have.


However, the same as a GUI does not exist for Zowe, which would help the administrator in tracking the scope of the application, plugin and the user. My project would achieve in creating a sub-application that would help the admin in getting the above data in a concise manner and in a GUI interface.

Currently, Zowe provides the API endpoint for accessing the above data as follows -

```
List plugins
List users
List groups
View & Edit preferences per-plugin
Add plugins
Remove plugins
View & Edit server settings
View server logs
View server environment parameters
```

We can create a dashboard, which would have different tabs for categorizing the data -

Plugin management - We would have a tab for accessing all the details of the plugins. We can also provide a feature to create a plugin and modify the plugin. This makes the admin control all the plugins and add / remove at their convenience.



User management - We can use the server and plugin endpoints to access the data of each user and access the metadata. For new APIs, we might want to allow admins to edit user permissions for our RBAC feature.

Server management - We would have a tab for accessing different API endpoints for the server management. Firstly, we can access the different metadata of the server, such as list of accessible server endpoints, server config files, contents of the log files, verbosity of the log files, details of the environment variables. We can also provide the option to reload the server, get the list of the accessible server agents. We can query based on server log contents, log-levels of each agent, and the environment variables of each agent.

Implementation

There are at the moment, two ways which I am thinking of implementing this Desktop Application Manager. The first way of implementing it would be to add this as a default plugin in the Desktop, and then it would behave as an app which would have control over all other plugins and all the features could be implemented as discussed above. This seems to be relatively better, since we can add more features to the desktop application application. Since it is loosely coupled, we don't have to worry about its robustness, as it would not fail our application.

Another way of implementing it would be to introduce the Desktop Application as an application layer between the API mediation layer and the GUI layer. This would be a nice feature since we can always add more plugins and not worry about extending the same, since our Application would lie between and receive modified API calls. We only need to modify the GUI to use the new API calls, and not worry about introducing the API support in the first method.

I would like to discuss the above with my mentor, and get his viewpoints about which method would be used for implementing the Application.



Timeline

Here is a tentative timeline for scoping out the different parts of the project and what I want to achieve tentatively -

Pre Coding Period (March 01 - April 31)

During this period, I am planning to discuss the intricacies of the project with my mentor. I have got some fair ideas, and I want to fix up the implementation details with my mentor. This would help me in filtering out unnecessary details and pin up specific goals for the project.

I would also use this opportunity to solve some issues related to Zowe, as I got introduced to this program very late. This would also help me in getting used to with the contribution manner of Zowe and by the time coding period starts, I would be fairly good to start with the project.

Week 1 (May 1 - May 7)

During this period, I am planning to discuss the intricacies of the project with my mentor. I have got some fair ideas, and I want to fix up the implementation details with my mentor. This would help me in filtering out unnecessary details and pin up specific goals for the project.

Week 1 - 2 (May 7 - May 21)

- Set up the local machine with all the current changes from the latest master branch.
- Document the logical breakdown of the application.
- Set up the repository and skeleton code of the application using AngularJS.
- Set up a Work in progress PR for tracking the progress of the work.

Week 3 - 4 (May 21 - Jun4)

- Design and code up the different components of the application, namely set up the different management pages.
- Document the logical workflow for this week's progress and design the HTML / CSS for the application

- 
- Set up the components of the application for the different managements

Week 5 (Jun 4 - Jun 11) - (Mid Term Evaluation)

- Buffer Period to catch up with any left over tasks
- Document all the changes that have been implemented uptill now.
- Create tests for testing out the changes made uptil now.
- Mid Term Evaluations and discuss with the mentors about the scope of the project.

Week 6 - 7 (Jun 11 - Jun 25)

- Start coding up the Server Management
- Document all the changes made for implementing the server management
- Write test suites for testing the server management.

Week 8 - 9 (Jun 18 - July 2)

- Start coding up the Plugin Management
- Document all the changes made for implementing the plugin management
- Write test suites for testing the plugin management.

Week 10 - 11 (July 2 - July 16)

- Start coding up the Server Management
- Document all the changes made for implementing the user management
- Write test suites for testing the user management.



Week 12 (July 16 - July 23)

- Consolidate all the work that has been done
- Prepare a report for explaining my progress for the project
- Finish up the documentation if any leftover parts are there
- Finish up writing up the tests for the application for any leftover parts
- Get the Work in Progress PR merged.

Contributing to Zowe

I have gone through the Zowe community, and I have been quite impressed by its community of helpful people. I was really happy to see like minded people in the community and am looking forward to learning from them.

I plan to solve some issues during my pre coding period, and am looking forward to keep contributing to this awesome community. I am currently exploring the field of DevOps engineering, and the kind of work that can be done in Zowe would certainly increase my practical understanding of DevOps engineering.

During my mentorship program, I plan to complete my tasks according to the schedule, since I don't have any prior commitments during my summer. I plan to put in 40 Hours of work each week, and would document my progress for each week.

I would get in touch with my mentor as per his convenience, and would share and discuss all my logical workflow, before coding. This would ensure that I am at par with the schedule that I have formulated.

In case my mentor becomes inactive, I plan to use help from the Zowe community and the maintainers, and would make sure that it does not become a blocker of time.