Sugarizer School Portal

31.03.2020

Basic Details

Full Name: Nikhil Mehra

Email: nmehra@ph.iitr.ac.in (College Email),

nikhilmehra998@gmail.com (Personal Email)

GitHub Username: NikhilM98

IRC Nickname: NikhilM98

First Language: Hindi, fluent in English

Location and Timezone: Roorkee, Uttarakhand, India (UTC +5:30)

Institute: Indian Institute of Technology, Roorkee

Degree: Bachelors' in Technology (B. Tech)

Major: Engineering Physics

Resume: LINK://to.resume

Share links, if any, of your previous work?

1. **Google Summer of Code 2019 with Sugar Labs:**
   I have been contributing to Sugar Labs since February 2019. Last year I got selected for the **Improve Sugarizer Server Dashboard** project during GSoC'19. Not only did I complete the tasks mentioned in the proposal on time, but I was also able to work on additional features like implementing GridFS storage and look into IIAB Sugarizer-Server MongoDB issue.
   Blog: **https://medium.com/@nikhilmehra998.**

2. **Thomso, IIT Roorkee:**
   Thomso is the annual cultural fest of IIT Roorkee. I have been involved as a volunteer, junior team member and as the technical head responsible for building and maintaining the web presence of Thomso. It was built using various technologies including PHP, Node.js and ReactJS over the years.
   Link: **https://github.com/pv-912/Thomso-18**.

3. **Internship at IoTrek:**
   I interned at IoTrek in the summer of 2018 where I converted the dashboard based on outdated version of AngularJS to ReactJS, remodelled the functionalities, and integrated new features.

4. **Contribution to Probot**
   In the winter of 2018, I was looking forward to contributing to open source. I created a few pull requests on Probot:
   #782 (Merged): [FIX] PRIVATE_KEY_PATH error handling (#778)
   #784 (Pinned): [FIX] Port in use error handling (#767)

5. **Internship at DocConsult:**
   I did my first development internship at the DocConsult startup during the winter of 2017 where I gained my first experience of working in a company and learnt to work under deadlines. The project was based on PHP and I worked on fixing bugs and improving the UI.

Contribution to Sugar Labs

To familiarise myself with the contribution and review process at Sugar Labs, I have contributed to two repositories. I made a number of pull requests to these repositories with invaluable input from Lionel Laské and Tarun K. Singhal. I have also been assisting the maintainers by reviewing pull requests by other users.

My significant contributions to Sugar Labs includes:

Sugarizer-Server:

- **#247 (Open):** [FIX] Language consistency
- **#244 (Open):** [FEAT] Display title on charts in statistics view
- **#237 (Merged):** [FEAT] Tutorial Update
- **#231 (Merged):** [STYLE] Code cleanup
- **#230 (Merged):** [CHORE] Updated Locale
- **#229 (Merged):** [FEAT] Journal Upload
- **#228 (Merged):** [FIX] Correct icons in recent entries table
- **#225 (Merged):** [FEAT] Journal download
- **#224 (Merged):** [FEAT] Allow teacher login on Sugarizer
- **#223 (Merged):** [FEAT] Private journal checkbox
- **#218 (Merged):** [FEAT] Delete script
- **#216 (Merged):** [FIX] Classroom inconsistency fix
- **#215 (Merged):** [FEAT] Classroom user search
- **#214 (Merged):** [FIX] no-prototype-builtins error removed
- **#213 (Merged):** [FEAT] User search in journal view
- **#212 (Merged):** [FIX] Remove text field from aggregate journal query
- **#211 (Merged):** [FIX] UI fixes
- **#209 (Merged):** [CHORE] Updated eslint config
- **#207 (Merged):** [CHORE] Package vulnerability fix
- **#206 (Merged):** [CHORE] Nodemon as dev-dependency
- **#205 (Merged):** [DOCS] Documentation update
- **#200 (Merged):** [FEAT] Updated tutorial for Stats View
- **#198 (Merged):** [FEAT] New charts added
- **#197 (Merged):** [FIX] Fix popup border

#196 (Merged): [FEAT] Statistics View Improvement
#195 (Merged): [FEAT] Sugarizer Server tutorial
#194 (Merged): [FEAT] Sort journal by title
#193 (Merged): [FEAT] Store journal text in GridFS bucket
#192 (Merged): [FEAT] Sorting by name and timestamp
#191 (Merged): [FIX] Stroke and fill in imp/exp
#190 (Merged): [FEAT] Update import users script
#189 (Merged): [FEAT] Import/Export users as CSV
#188 (Merged): [FEAT] Import users from CSV from Users View for Admin
#186 (Merged): [FEAT] Show classrooms in edit user view
#185 (Merged): [FEAT] Teacher role
#184 (Merged): [FIX] Disable user role change
#183 (Merged): [REFACTOR] Restructured graph controller
#182 (Merged): [STYLE] Fix miscellaneous linting errors
#181 (Merged): [FEAT] Minify CSS and Images using Grunt
#180 (Merged): [STYLE] Indentation and semicolon error fix
#178 (Merged): [FIX] Aggregate Journals API
#177 (Merged): [REFACTOR] Fix no-def and no-unused-vars errors
#176 (Merged): [FEAT] Grunt to minify public resources
#175 (Merged): [FIX] Update dashboard reference in addClassroom
#174 (Merged): [REFACTOR] Restructured auth controller
#173 (Merged): [REFACTOR] Restructured classrooms controller
#172 (Merged): [REFACTOR] Restructured journal controller
#171 (Merged): [REFACTOR] Restructured stats controller
#170 (Merged): [REFACTOR] Restructured users controller
#169 (Merged): [REFACTOR] Restructured dashboard controller
#168 (Merged): [FEAT] Script to import Users from CSV
#167 (Merged): [FEAT] Use a random color for new user/classroom
#164 (Merged): [FIX] Flickering of sidebar fix
#162 (Merged): [FIX] Removed redundant translation on language change
#160 (Merged): [FEAT] UX Improvement. Added tool-tips for icons
#159 (Merged): [FEAT] Improve navigation on mobile view
#158 (Merged): [FIX] Fix classroom name on view students from classroom view

#156 (Merged): [FIX] Improve responsive view
#155 (Merged): [FIX] Initiate language change on sidebar toggle
#151 (Merged): [FIX] UX improvement. Increased language consistency
#149 (Merged): [FIX] UX improvement. Fixed links and cursor
#144 (Merged): [FIX] Re-initiated locale on non-index views
#143 (Merged): [FIX] Fix classroom for similar students
#141 (Merged): [FEAT] Show view journal button only for students in users view
#140 (Merged): [FIX] Fix API inconsistency for asynchronous calls
#138 (Merged): [FEAT] Display name of the activity while deleting journal entry
#137 (Merged): [FEAT] Display name in notification after create/update/delete
#135 (Merged): [FIX] Fixed placeholder translation in addEditClassroom
#134 (Merged): [FIX] Re-init l10n in stats controller
#129 (Merged): [FIX] Fix launching activity with null text
#128 (Merged): [FIX] Fix console error on bar chart onClick
#124 (Merged): [FIX] Fix incorrect value in journal select field
#121 (Merged): [FIX] Dropdown language change fix
#115 (Merged): [FIX] Delete private journal on delete user
#110 (Merged): [FEAT] Translate stats labels. Corrected and added translations
#108 (Merged): [FIX] Updated deprecated mongodb functions
#107 (Merged): [FIX] Search for stroke and fill fixed
#103 (Merged): [FEAT] List admins after create/update/delete admin
#101 (Merged): [FIX] Fixes breaking tests on dev. Handling admin deletion
#93 (Merged): [FIX] Fix the ordering of activities while sorting
#92 (Merged): [FIX] Classroom user count fix
#89 (Merged): [FEAT] UX Improvement. Hide non editable empty fields
#86 (Merged): [FIX] Fix add another classroom/user issue
#76 (Merged): [FIX] Updated locale
#75 (Merged): [FIX] Creating classroom with single student
#73 (Merged): [FEAT] UX improvement
#72 (Merged): [FIX] Added missing translations
#58 (Merged): [FEAT] Installing linter in the project
#55 (Merged): [FIX] Fixes broken dev branch
#52 (Merged): [FIX] Server crash fix

#48 (Merged): [FIX] Added missing Hindi and Spanish translations

#46 (Merged): [FIX] Variable scope and typos issue fixed

#45 (Merged): [FEAT] Use single instance of db connection

ExerciserReact:

#2 (Merged): [FIX] Fixes crashes in development mode

#1 (Merged): [REFACTOR] Unused parameters removed from routes

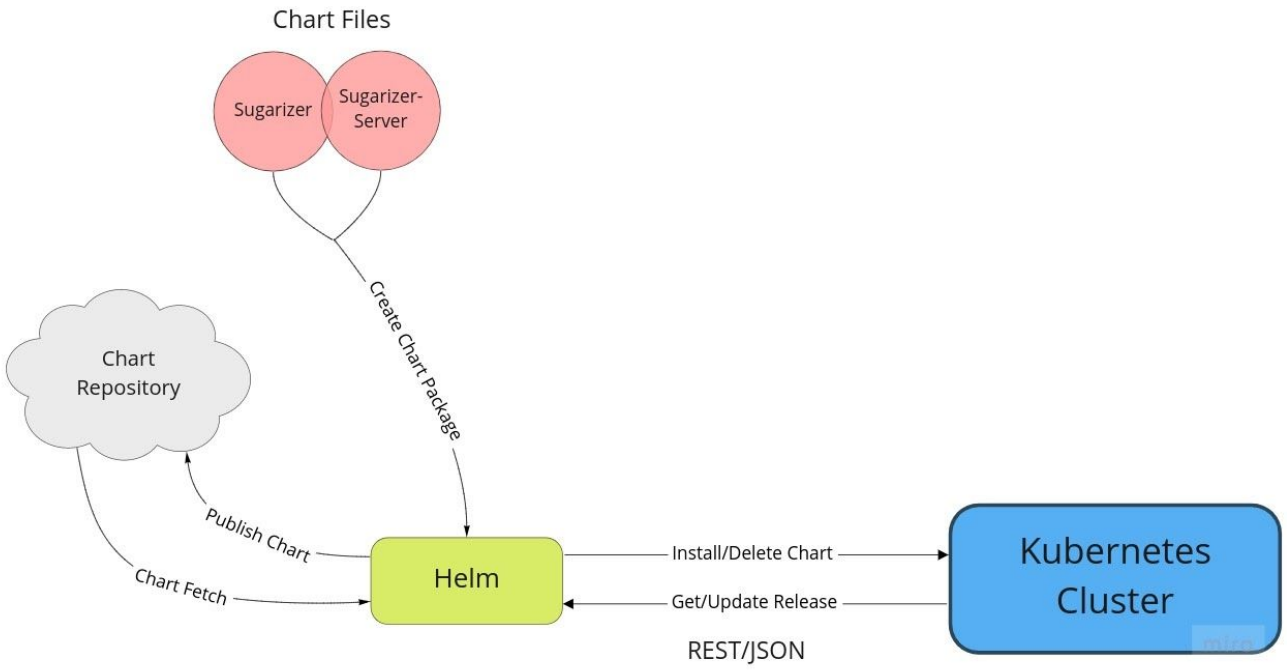
Project Details

What are you making?

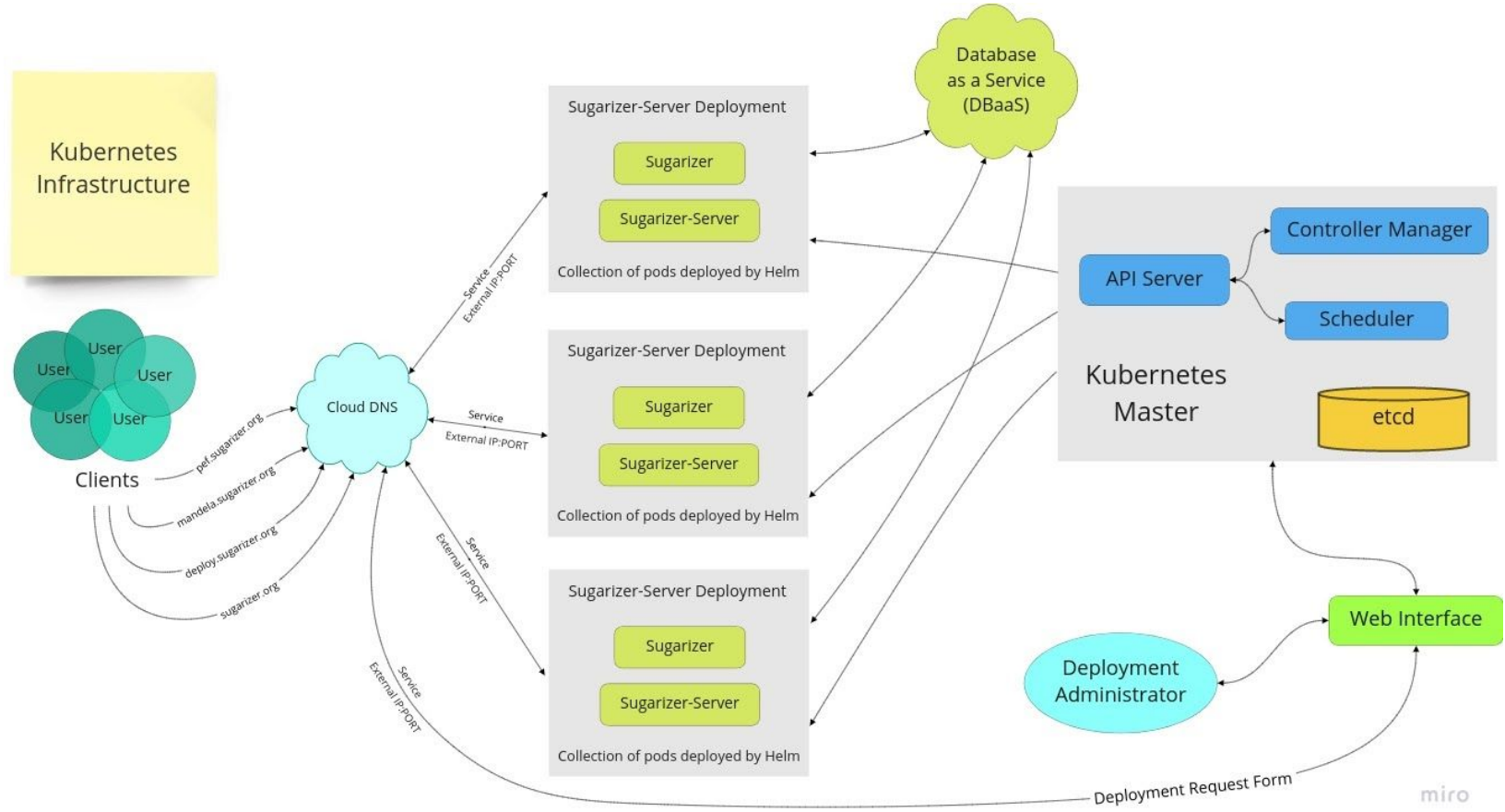
The goal of my project is to create a Kubernetes based portal to provide on-demand Sugarizer Server Deployment. My proposed features include (The list may grow as the discussion progresses):

- Use Helm to create a Sugarizer and Sugarizer Server package (Sugarizer-Server Chart). Helm Charts helps define, install, and upgrade even the most complex Kubernetes application. The chart will be used to deploy the Sugarizer-Server on the Kubernetes cluster and it will be released on the Helm Chart repositories for easy installation and up-gradation of the chart.

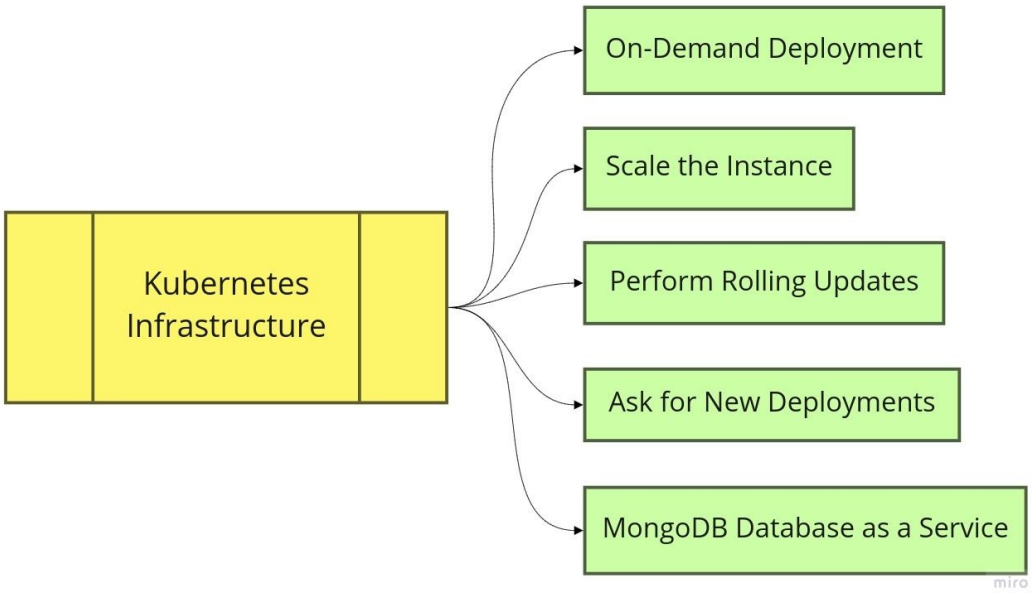
Helm Architecture



- Create a Kubernetes infrastructure that could deploy on-demand Sugarizer-Server instances.
    - Create a web interface that will use Kubernetes API and Helm API to communicate with the infrastructure and allow the deployment administrator to create/manage Sugarizer Server instances.
    - The infrastructure will be able to expose the service onto an external IP address that's outside of the cluster. Clients should be able to use Sugarizer and Sugarizer-Server using the external IP and the node port. We may expose the service externally using a cloud provider's load balancer like Google Cloud DNS depending on the discussion.



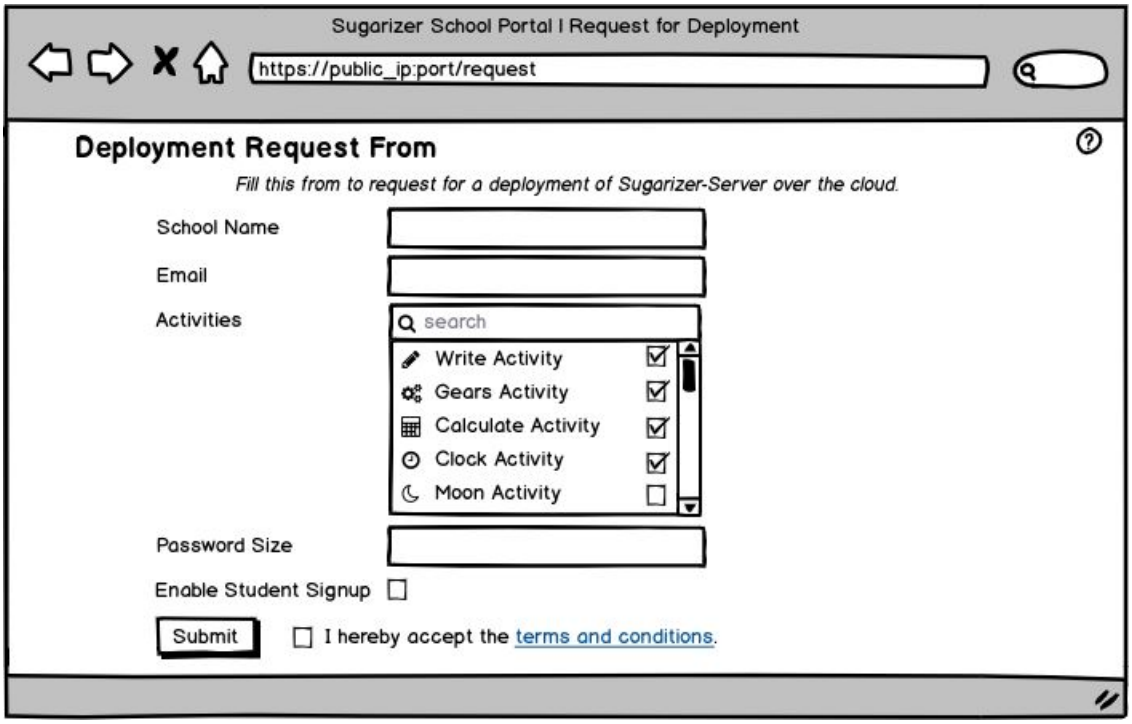
- The infrastructure will use MongoDB Database as a Service (DBaaS). The database will run on a cloud computing platform, and access to the database will be provided as-a-service. Database services take care of scalability and high availability of the database, and provide backup and recovery support for the database.
- The infrastructure will allow scaling of the instance to manage the deployment by adjusting the pod's CPU and memory requests.
- The infrastructure will be able to perform rolling updates, allowing deployment updates to take place with zero downtime by incrementally updating pods instances with new ones.

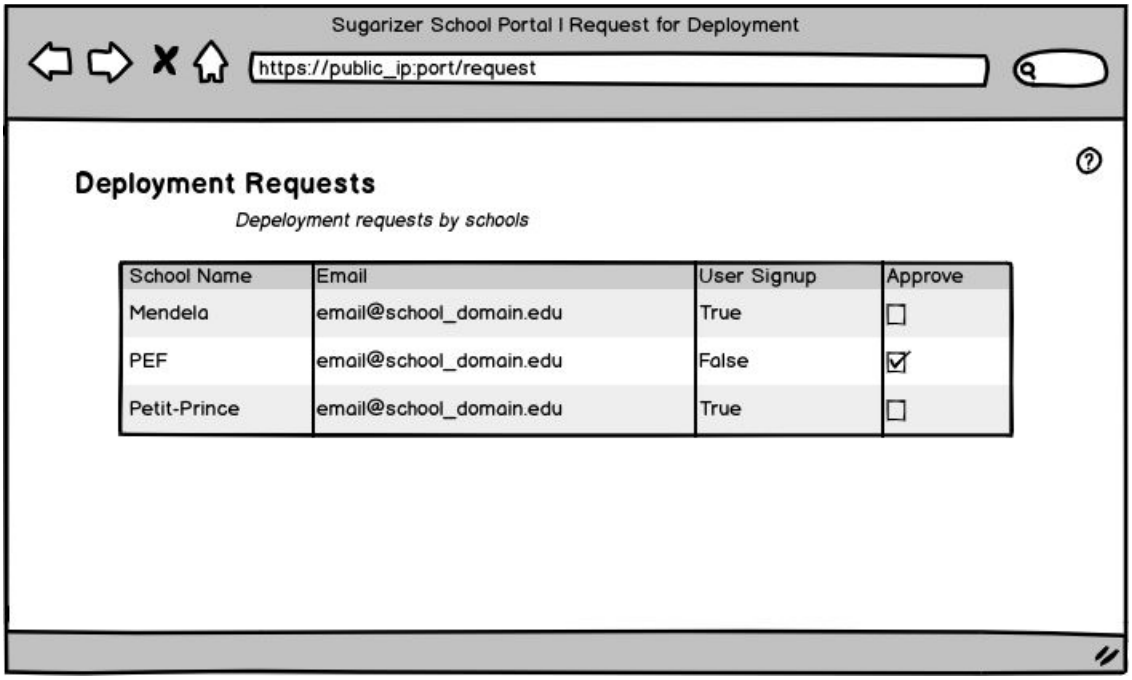


- Deploy the infrastructure on the Scaleway Kubernetes Kapsule or Google Kubernetes Engine (GKE) or on some other Kubernetes cloud provider. These services provide high availability managed orchestrators which will allow the deployment of Sugarizer School Portal on the cloud and would make the Sugarizer-Server instances accessible over the internet.

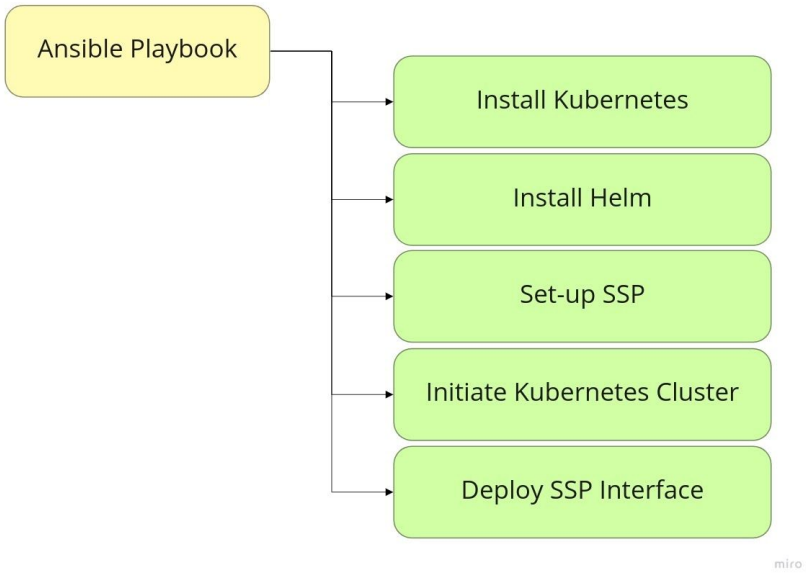
- Create a set of scripts to extract the stats usage data of the Sugarizer Server and save it in a JSON format. The JSON data then could be presented in a more human-friendly way on the basis of the discussion.

- Create a web interface to let users ask for a new deployment. The web interface will be a part of the same app and the users can ask for a new deployment by filling a "Deployment request form". The deployment administrator would approve the request and the credentials of the Sugarizer-Server and the address of the instance will be sent to the user.





- Create an Ansible playbook to bring the Kubernetes stack up for on-premise configurations. The playbook will set up the Kubernetes cluster on the system and initiate the web interface which will be used to manage the Kubernetes infrastructure.



- Manage TLS certificates in the Kubernetes cluster using Kubernetes certificates.k8s.io API. This API uses a protocol that is similar to the ACME draft. Also, we will be using cert-manager to automate the management and issuance of TLS certificates. These CA and certificates will be used by the pods to establish trust.
- If time permits, try to integrate Sugarizer APK builder with the Sugarizer School Portal, allowing the schools to generate and download their own pre-configured APK which enables them to connect with their Sugarizer-Server instance hosted on the cluster.

How will it impact Sugar Labs?

With the successful implementation of this project, Sugarizer Server will become more easily accessible to the users. Schools will be able to request for new deployments by filling a simple form and the deployment administrator will be able to deploy a new instance of Sugarizer Server for that school in a few clicks. This will allow every school to create a Sugarizer Server to host its own deployment without any technical skill.

What technologies (programming languages, etc.) will you be using?

We will be using Node.js on the server-side as a runtime environment, Express as a web application framework and EJS as the templating engine. Kubernetes will be used for automating deployment, scaling, and management of containerized applications. Helm will be used to manage the Kubernetes applications. JQuery and Bootstrap will be used to design and make an interactive and responsive user interface. Apart from that, we will be using Shell for scripting. We will be writing Ansible playbooks to bring the Kubernetes stack up. Most of the programming would be done in Javascript, HTML and CSS

.

Timeline

Mention how much time will you spend each week working on your project?

I have no other commitments during my summer vacation. I can easily target about 50 hours a week.

Project timeline:

Timeframe	Start Date - End Date	Task
Phase 0: Community Bonding	May 4 - May 10	- Interact with the mentors of the project and set up feedback loops. - Explore the technologies to be used in the project.
	May 11 - May 17	- Continue to refine the plans for the project in consultation with the mentors. - Discuss the structure of the project.
	May 18 - May 24	- Discuss the features of the project. - Continue exploring the technologies to be used in the project.

	May 25 - May 31	- Discuss and finalize the architecture of the Sugarizer Server Helm Chart. - FInalize the idea and structure of the project
Phase 1: The goal of this phase is to: - Create a Helm Chart for the Sugarizer-Server. - Set up the Kubernetes Infrastructure	June 1 - June 7	- Set-up a testing environment consisting of a small Kubernetes cluster. - Start working on creating a Helm chart for the Sugarizer Server.
	June 8 - June 14	- Connect the infrastructure with an external database provider. - Test and secure the connection and review the database backup/restore strategy.
	June 15 - June 21	- Host the chart on the cloud repository. - Test Scaling and Rolling Updates on the infrastructure and find a way to automate it through an API.
	June 22 - June 28	- Test the deployment of the Sugarizer Chart on the cluster using HELM and find a way to automate it through an API. - Test the networking of the infrastructure. - Prepare for the first evaluation.

Phase 2:
The goal is this phase is to:
- Create the web app to create/manage Sugarizer-Server instances.
	June 29 - July 5	// First Evaluation
		
		// Start working on the Web App.
		
		- Finalize the UI, routes, and features of the app.
		
		- Initiate a simple web app with the discussed endpoints.

	July 6 - July 12	- Start working on the APIs for the app. The APIs should be using Kubernetes API and Helm API to communicate with the cluster. - Add user authentication to the app and secure the routes. - Write unit tests for the APIs
	July 13 - July 19	- Start working on the front-end of the app. - Add user login page and user dashboard. - Show basic cluster-info on the dashboard.
	July 20 - July 26	- Add the feature to deploy/delete Sugarizer-Server Instance. - Add the feature to provide rolling updates through the UI.

Phase 3: The goal of this phase is to: - Add more features to the web app to manage the infrastructure. - Create an Ansible Playbook to do the configuration and set up the Kubernetes cluster. - Review the docs	July 27 - August 2	// Second Evaluation. - Add the feature to scale to an instance. - Show the Infrastructure statistics on the dashboard.
	August 3 - August 9	- Add the feature to request for a new deployment. - Start working on creating an Ansible Playbook for the product.
	August 10 - August 16	- Continue working on creating the Ansible Playbook. - Start working on the documentation for the product.
	August 17 - August 23	- Polish up the UI and review the documentation. - Finish any remaining work and prepare for the final evaluation.
Finalize	August 24 - August 31	// Final Evaluation
Post-GSoC Period	-	Continue to work on the project, finishing items that have been put on an if condition in the last two weeks. Improve the codebase in aspects that will only come to light when integrations are getting merged.

Distant Future	-	In the distant future, I would love to work with Sugar Labs as they explore creative ways to provide educational opportunities to the children. I would also love to mentor new-coming contributors to the project in any way I can.

Motivation

What is your motivation to take part in Google Summer of Code?

I have utilised a lot of open-source technologies in my projects. GSoC provides me with a chance to work on software that has a tangible effect on millions of people around the globe. The prospect of developing software actually capable of making a difference means a lot to me and I would very much like to be able to give back to the community in some way.

Why did you choose Sugar Labs?

I have always wanted to do something for society. Sugar Labs presented me with the means by which I can learn and give back to society by providing children with equal opportunities to learn, irrespective of their background.

I have been contributing to Sugar Labs since February 2019 and my journey so far has been a great one. I am fortunate to have met this vibrant community of motivated developers and creators. Every pull request I have made to Sugar Labs has taught me something new and interesting, which was only possible because of the presence and feedback of the community on issues and pull requests.

Even before GSoC has begun, I have learned so much, which only makes me look forward to all the more things I could learn from this endeavour.

Why do you want to work on this particular project?

I've worked on Sugarizer-Server during GSoC'19. This project has a familiar tech stack so I will be more comfortable in contributing to the project.

What are your expectations from us during and after successful completion of the program?

During the progress of the project, I expect at least weekly feedback and suggestions from the mentors. After the successful completion of the project, I plan to continue my contributions to Sugar Labs. Even afterwards I will be looking forward to possible mentorship opportunities from Sugar Labs.