Project Title : Add Docker Swarm Support to Sugar

Open Science Labs - GSOC Program - 2025

Candidate Info :

1. **Name** : Sai Sanjay Kottakota
2. **Github** : sanjay7178
3. **Email** : saisanjay7660@gmail.com
4. **Twitter** : sanjay7178
5. **University Course**: Integrated Master's in Computer Science and Engineering (Current)
6. **University**: VIT University
7. **Time Zone**: IST (Indian Standard Time)

Bio:

I'm Sai Sanjay Kottakota, currently pursuing an Integrated Master's degree in Computer Science at Vellore Institute of Technology, India. My passion for programming ignited when I was 18. During my freshman year at university, I completed a MOOC course on Ethical Hacking, which introduced me to CLI applications such as NMAP, SOCAT, hashcat, john the ripper, and more. I also explored OWASP Top 10 web tools like Burpsuite, Dirbuster, ffuf, Hydra, and others, which sparked my interest in web application security.

Research Experience:

In my sophomore year, I developed an IoT-based remote bird specimen identification system using ESP32 and various sensors. I trained custom models using TensorFlow and benchmarked them to achieve optimal image recognition performance, resulting in a research paper publication.

During my summer internship at Digital Fortress Inc, I developed and deployed custom computer vision solutions , working on a State-of-the-Art Facial Anti-spoofing model. This project is currently a Work In Progress for submission to an A* Conference and is available as an open- source project: DeepPairNet. Additionally, I authored a preprint on deploying custom computer vision models in serverless environments.

I also contributed to Saila, a FIDO-enabled smartphone-based passwordless authentication system designed to protect against HID injection. This experience provided me with extensive knowledge in building CLI applications and deepened my understanding of Passkeys, Golang, and cross-platform HID device interactions in Windows and Linux.

Open Source Projects :

I've developed several CLI and Golang-based applications:

- **go-ukip**: A daemon process running on cross-platform devices to provide runtime protection against USB keystroke injection and DNS assignment over DHCP spoof attacks from BadUSB

- **file-compressor**: A versatile file compression tool implemented in Go that supports multiple compression algorithms and chaining them together

  openfortiapauth: An open-source implementation for automatic Fortinet FortiAP (Captive Portal) client login

Addressing usability issues with my university's management site (VTOP), I created a Chrome extension called **vRevamp** to enhance user experience. The extension has surpassed **3000+** downloads, providing me with valuable experience in version control, tagging, and release management.

I participated in Hacktoberfest 2024, contributing to open-source organizations including Null Open Security Community, torcheeg, and Mattermost.

Community Leadership & Research

Later in the year, I worked as a Research Intern at **Georgia Southern University's Brain Computer Interface lab** under Dr. Meenalosini Vimal Cruz, focusing on state-of-the-art multimodal EEG emotion recognition.

I also established and currently lead the Null Vijayawada chapter, where I organize and host Capture The Flag (CTF) competitions, monthly meetups, and workshops to educate students, professors, and industry professionals in the security domain.

Project Overview

- **Project:** Sugar
- **Project Idea/Plan:** Add Docker Swarm Support to Sugar
- **Expected Time (hours):** 350 hours

Abstract

The goal of this project is to extend the Sugar library with Docker Swarm support. This enhancement will enable Sugar users to manage Docker Swarm clusters directly through Sugar's command-line interface (CLI). By adding commands for initializing, joining, creating, scaling, updating, and inspecting Swarm services, the project aims to streamline container orchestration workflows for developers.

Mentors

- Ivan Ogasawara
- Luis Casas
- Sandro Loch
- Felipe Paes

Technical Details

Pull Request Related and Wiki to This Project :
https://github.com/osl-incubator/sugar/issues/126 ,
https://github.com/osl-incubator/sugar/wiki/Project-Ideas#project-idea-2-add-docker-swarm-support-to-sugar

Implementation of new Sugar commands for

Docker Swarm:

- sugar swarm init - Initialize a swarm
- sugar swarm join - Join a swarm as a node and/or manager
- sugar swarm create - Create a new service
- sugar swarm inspect - Display detailed information on one or more services
- sugar swarm logs - Fetch logs of a service or task
- sugar swarm ls - List services
- sugar swarm ps - List the tasks of one or more services
- sugar swarm rm - Remove one or more services
- sugar swarm rollback - Revert changes to a service's configuration
- sugar swarm scale - Scale one or multiple replicated services
- sugar swarm update - Update a service

Updates to the following files:

- src/sugar/cli.py
- src/sugar/schema.json
- src/sugar/core.py

Creation of a new extension file:

- src/sugar/extensions/swarm.py

Proposal Description

Sugar is a Python package that offers an abstraction layer over Docker commands with a focus on developer experience. It currently supports Docker Compose commands through the `SugarCompose` class hierarchy. The proposed enhancement adds comprehensive Docker Swarm orchestration support through a new `SugarSwarm` class that maintains Sugar's existing design patterns such as `SugarBase` .

Sample Project Implementation

I'm proposing `SugarSwarm` class, extending Sugar's `SugarBase` class to provide an elegant interface to Docker Swarm operations. The implementation follows Sugar's design principles:

```
class SugarSwarm(SugarBase):
    """SugarSwarm provides the docker swarm commands."""

    def _load_backend_app(self) -> None:
        self.backend_app = sh.docker
        self.backend_args = []

    def _call_swarm_command(self, subcommand: str, ...) -> None:
        """Call docker swarm commands with proper structure."""
        self.backend_args = ['swarm']
        self._call_backend_app(...)
```

Maintained consistent API Design , which this interface follows Sugar's existing command pattern's

```python
@docparams({**doc_stack, **doc_scale_options})
def _cmd_scale(
    self,
    stack: str = '',
    replicas: str = '',
    detach: bool = False,
    options: str = '',
) -> None:
    """Scale one or multiple replicated services."""
    # Implementation
```

Note : for proof working for some of sugar swarm commands such as `sugar swarm node demote` needed to make it as `_subcmd_node_demote` .

I've completed the sample implementation and proceeded with the implementation of swarm commands into the sugar repository, as mentor also reviewed changes in pull request #149 which are minimal implementations to prove an initial idea. In the sample implementation, I've made the initial set of the sugar swarm commands functional.

I've created a detailed system architecture diagram that illustrates the proposed Docker Swarm implementation, showing how the new commands will integrate with Sugar's existing codebase and the data flow between components. This diagram is available in this GitHub Gist: https://gist.github.com/sanjay7178/560b0f97cccc85ca937524d4aa87e6a1

Conformance of Sample Implementation (as of now):

- With regards to the commitment from my mentor In his comment, I tried to move forward to begin unit testing using PyTest on the SugarSwarm class in test_swarm.py.
- I had written smoke tests in the .makim.yml .

The logs of the PyTest run are present in this GitHub Gist:
https://gist.github.com/sanjay7178/702d49cdec19925153ddda6cfe844507

Test Case Result	**#**
Passed	42
Failed	0
Skipped (Experimental sugar swarm command added for documentation	1

purpose , but didn't implemented fully functional features of ie … sugar swarm update)	

The docs related to the sugar swarm I've generated using MkDocs , here is the link https://sugar-5ir.pages.dev/swarm/

Based on community needs and enhancing usability, I propose implementing these additional features:

Adding `sugar swarm lock ` to protect swarm state from current state

- For integrating this we can use : To enter key `docker swarm unlock` or Print Unlock Key `docker swarm unlock-key` (verbose way) , Only to print key `docker swarm unlock-key -q` , To generate new key `docker swarm unlock-key --rotate`
- Sugar Commands:
  - `sugar swarm unlock`
  - `sugar swarm unlock-key`

Template-Based Service Addition

```
service_templates:
  redis:
    image: redis:latest
    ports:
      - "6379:6379"
  postgresql:
    image: postgres:13
    environment:
      POSTGRES_PASSWORD: example
    ports:
      - "5432:5432"
  mysql:
    image: mysql:8
    environment:
      MYSQL_ROOT_PASSWORD: example
    ports:
      - "3306:3306"
```

- Implement a CLI command to add pre-configured templates for common services (Redis, PostgreSQL, MySQL, etc.)

Command :

sugar service add redis --profile profile1

- Create a template system that allows quick integration of additional services into existing projects
- Speed up service creation with standardized configurations

Enhanced Network Management

```yaml
networks:
  my-bridge-network:
    driver: bridge
  my-overlay-network:
    driver: overlay
    attachable: true
    scope: swarm

profiles:
  profile1:
    project-name: project1
    config-path: tests/containers/profile1/compose.yaml
    env-file: .env
    networks:
      - my-bridge-network
    services:
      default: service1-1,service1-3
      available:
        - name: service1-1
          networks:
            - my-bridge-network
        - name: service1-2
          networks:
            - my-overlay-network
```

- Extend the configuration to support custom network definitions in .sugar.yaml
- Enable cross-node communication via overlay networks in Swarm mode
- Include DNS entries from Docker DNS resolver in the get-ip feature
- Implement network commands for creation, listing, and removal of networks

Advanced Logging Mechanism

- Implement side-by-side log viewing for services running on different nodes/replicas in a stack , for reference TMUX has multiple tabs in bottom of terminal , so user can switch between those tabs using shortcuts

Secret/Config Management

- Develop mechanisms for changing secrets/configs without shutting down stacks
- Implement automation for config rotation using Swarm capabilities
- Add naming convention system for secrets/configs
- Create config history logging
- Adding third party secrets vendor support such as vault

Reference : https://www.youtube.com/watch?v=hWEg7aDbVLs , https://stackoverflow.com/questions/45171564/using-vault-with-docker-compose-file

Volume Management

- Add functionality to identify all pre-defined volumes of an image
- Provide tools to inspect and manage anonymous volumes
- Simplify volume discovery and management

Benefits to the community

Adding Docker Swarm support to Sugar will provide several benefits to the community:

1. Users can move from development environments to production-ready deployments using the same tool.
2. The community will be able to easily scale services across multiple nodes without leaving the familiar Sugar interface.
3. Users can leverage Swarm's high availability features for critical applications.
4. With the additional features for network, volume, log, and secret management, users will have more granular control over their container resources.

Deliverables and Timeline

Timeline

Dates	**Deliverables/Tasks**
Community Bonding Period **May 8 - June 1**	
Week 1 **May 8 to May 14**	● Read up on Docker Swarm and Sugar documentation ● Focus on end-semester exams
Week 2 **May 15 - May 21**	Fill in more details in the proposal with all research performed till date
Week 3 **May 22 - June 1**	● Get to Know my mentors and fellow sugar contributors ● Hash out expectations, development conventions and schedules for meetings
Goals for Period ● Gain an user perspective on the Sugar usage ● Become a part of the waycrate community ● Improve understanding of the codebase ● Try improving documentation of the codebase in this process ● Get used to university coursework and GSoC	

Week 4 **June 2**	Coding officially begins
Week 4 & 5 **June 3 - June 13**	• Implement basic Swarm initialization and node management commands
Week 5 **July 14 - 18:00 UTC**	Mentors and GSoC contributors can begin submitting midterm evaluations
Week 6 **June 15 - June 17**	• Develop mechanisms for changing secrets/configs without shutting down stacks • Implement automation for config rotation using Swarm capabilities • Adding third party secrets vendor support such as vault • Write blog on easier usage of secrets exchange of secrets in production and testing environments
Week 6 **July 18 - 18:00 UTC**	Midterm evaluation deadline (standard coding period)
July 14 - August 25 Work Period | GSoC contributors work on their project with guidance from Mentors	
Week 6 **July 19 - July 20**	• Create unit tests for new commands • Write blog post about initial implementation
Week 7 **July 21 - July 28**	• Implement service deployment and management commands • Add configuration handling for Swarm services • Write integration tests • Blog post on service deployment implementation
Week 8 **July 29 - August 04**	• Implement service scaling and update commands • Add stack deployment functionality for Swarm • Create tests for scaling operations • Mid-term evaluation preparation • Blog post on scaling implementation
Week 9 **August 05 - August 11**	• Implement swarm rollback commands • Create tests for rollback operations • Blog post on simplification of rollback management with sugar

Week 10 **August 12 - August 18**	• Implement network and volume management for Swarm • Add secret and config management • Create comprehensive tests • Blog post on resource management
Week 11 **August 19 - August 25**	• Implement monitoring and logging commands for Swarm services • Add health check functionality • Create tests for monitoring features • Blog post on monitoring implementation
August 25 - September 1 - 18:00 UTC Final week: GSoC contributors submit their final work product and their final mentor evaluation (standard coding period)	
Week 12 **August 26 - September1**	• Complete documentation • Fix any remaining bugs • Perform final integration testing • Prepare final submission • Final blog post summarizing project achievements

Previous contribution to the project

Pull Requests

My experience with Sugar has been really amazing, Sugar has been my very best Open Source Organization contributions and I owe my learnings about open source and community to Sugar mentors who have helped me through the journey. I have been contributing to Sugar since March beginning,Below are some of my Contributions to the Organization:

Pull Request Title/Number	Link	Status
feat: Add help flag support to CLI in any OS path	#136	Merged
refactor: Simplify constructor arguments in StatsPlotWidget and Stats…	#141	Merged
refactor: Rename 'group' to 'profile' in configuration files and update related paths	#146	Merged

docs: Fix the entry/key defaults in an example in the README.md	#148	Merged
feat: Add support for 'swarm' backend in schema ,cli and core extensions	#149	Merged
feat: Add support for 'podman-compose' in sugar	#167	Merged
feat(cli): enhance Typer app with subcommand help fallback upon empty command args	#176	Merged

Based on what I've learned, I intend to discuss with mentors to integrate these features via pull requests effectively. These Open Source practices have inspired me to further explore more real life practice of Test Driven Development (TDD) using PyTest and Integration of Smoke tests using Makim . I am deeply grateful for the opportunity to learn and grow in this fascinating field so far.

Why this project?

I'm passionate about cloud-native technologies and container orchestration, which are revolutionizing how we build, deploy, and manage applications at scale. This project to add Docker Swarm support to Sugar aligns perfectly with my interests and expertise in this domain.
Docker Swarm, while simpler than Kubernetes, offers a robust solution for container orchestration that's ideal for small to medium-sized deployments. So integrating into Sugar provides essential features like service discovery, load balancing, and rolling updates, making it would provide an excellent choice for teams looking to adopt cloud-native practices without the complexity of larger platforms.
It's an exciting intersection of my skills in Python development, my interest in containerization, and my desire to contribute to open-source projects that have a real impact on how developers work.

Availability

I can dedicate 30-35 hours per week to this project during the GSoC period. My current academic commitments include coursework at VIT University, but I've arranged my schedule to ensure I have sufficient time for GSoC work.

If I fall behind schedule, I plan to:

1. Communicate proactively with mentors about any delays
2. Allocate additional hours during weekends to catch up
3. Reprioritize tasks to ensure critical functionality is completed first

Post-GSoC

I see this project as the beginning of a longer-term contribution to the Sugar ecosystem, and I'm committed to ensuring its success beyond the GSoC period.