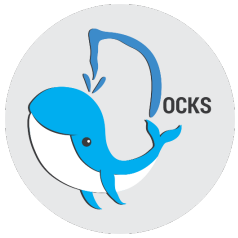
User Manual for Docks

Team: TripleParity
Client: Compiax



Contents

1 System Overview

This document assumes knowledge of how Docker works. For more information read:

- About Docker Engine
- Docker Overview
- Swarm Mode Key Concepts

Docks provides a web interface for managing a Docker Swarm. Along with an easy to use interface Docks provides security by only allowing registered users to manage Docker. Docks exposes the same functionality as the Docker Command Line Interface.

Docks allows developers and system administrators to manage the deployment of applications without requiring SSH access to a server.

2 System Configuration

Docks consists of two subsystems:

- Docks Web Interface
- Docks API Server

The web interface can be served from any static file server such as GitHub pages and will communicate with the Docks API server through the web browser.

The Docks API server is deployed on a Manager Node in the Docker Swarm

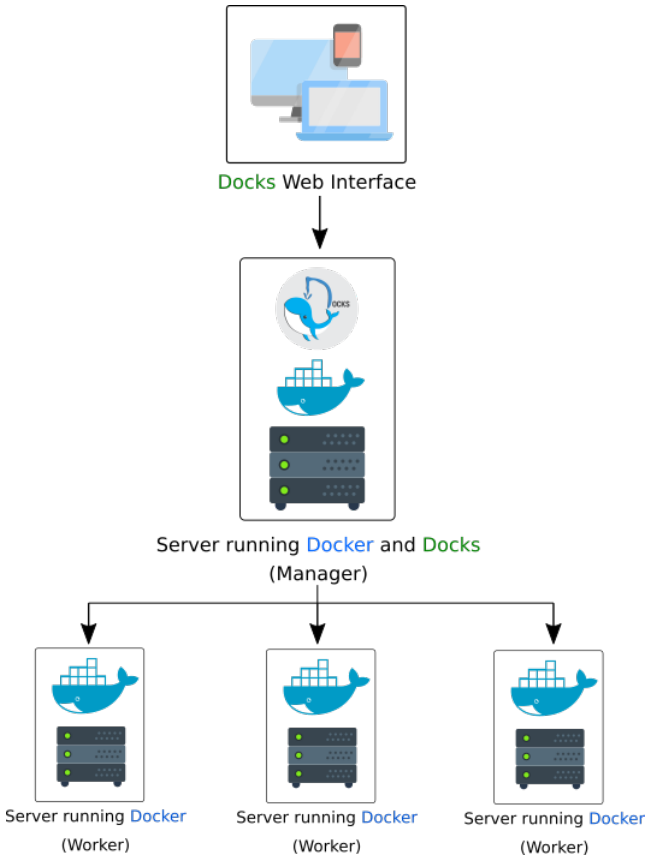


Figure 1: Docks Deployment Diagram

3 Installation

1. Install Docker 17.06.2-ce or higher
2. Install Docker Compose
3. Create a Swarm using `sudo docker swarm init`
4. Clone `https://github.com/TripleParity/docks.git`
5. Run `sudo docker-compose pull` to download the required images
6. Run `sudo docker stack deploy -c docker-compose.yml docks` to deploy Docks
7. Run `sudo docker stack deploy -c docker-compose-nginx.yml demo` to deploy a sample application
8. Browse to `http://127.0.0.1:4200` to view the Docks web interface
9. To remove Docks from the system run the following commands:
    - `sudo docker stack rm docks`
    - `sudo docker stack rm demo`

3.1 Configuration

3.1.1 Docks Web Interface

The following parameters can be configured in the `docker-compose.yml` file

- `DOCKS_API_ADDRESS` - The address of the Docks API server
- `ports` - The ports to listen on

3.1.2 Docks API

The following parameters can be configured in the `docker-compose.yml` file

- `JWT_SECRET` - The secret key used during authentication requests
- `DOCKS_DB_ADDRESS` - The address of the database
- `POSTGRES_PASSWORD` - The password for the `postgres` database user
- `ports` - The ports to listen on

4 Getting Started

The web interface will be available at `http://127.0.0.1:4200` after following the Installation instructions. The database will automatically be initialized and the default user will be created:

- Username: `admin`
- Password: `admin`

It is strongly recommended to change the password as soon as possible. This will be explained further in User Management

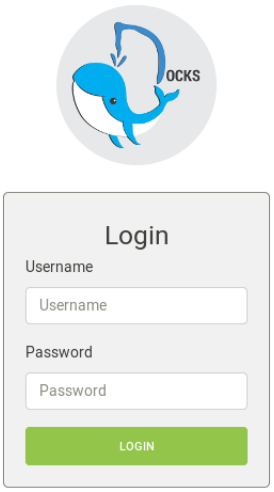


Figure 2: Login Page

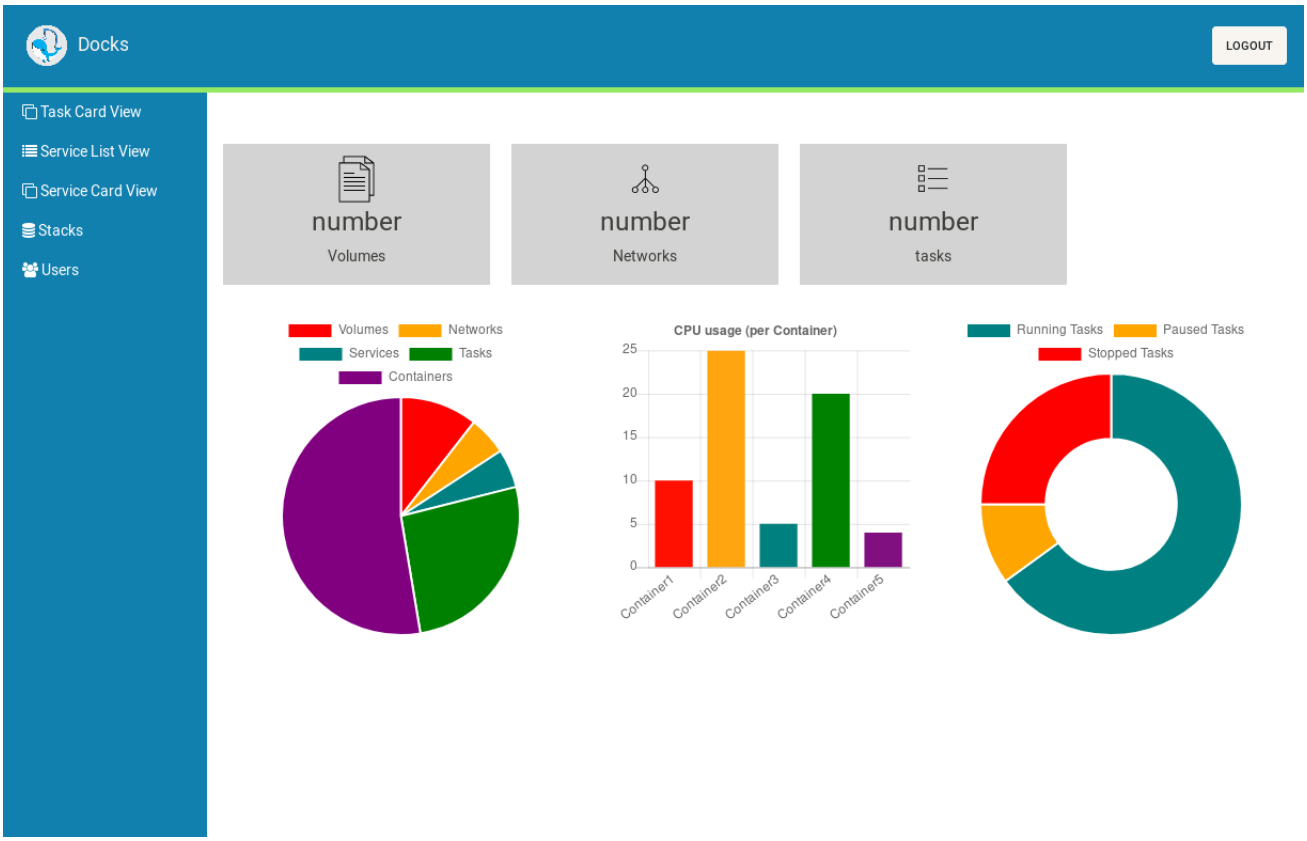


Figure 3: Home Screen

5 Using the System

5.1 Tasks

A Service consists of Tasks. Tasks can only be viewed - they are managed by Docker as part of services

5.1.1 Viewing Tasks

The card view displays the following information:

- ID of the task
- State of the task. Indicated by the circle and card outline colour. Green means the task is running, red indicating it has stopped and blue indicating it is preparing
- Node ID the task is running on
- Last Updated

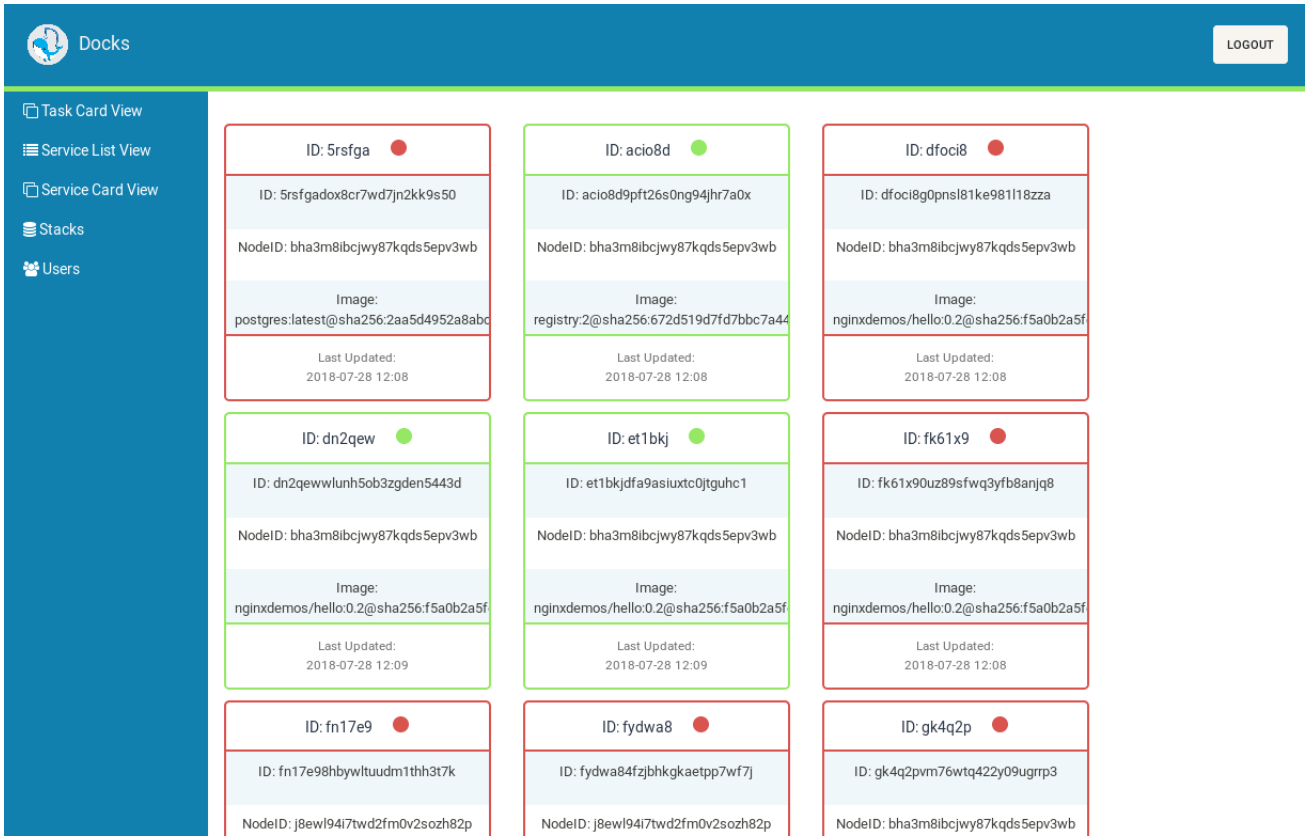


Figure 4: Task Card View

5.1.2 View Task Logs

From the task card view, a task can be clicked to bring up a modal for viewing the log associated with the task.

Figure 5: Task Card Modal

5.2 Services

The service list view provides a sortable and searchable table for deployed services.

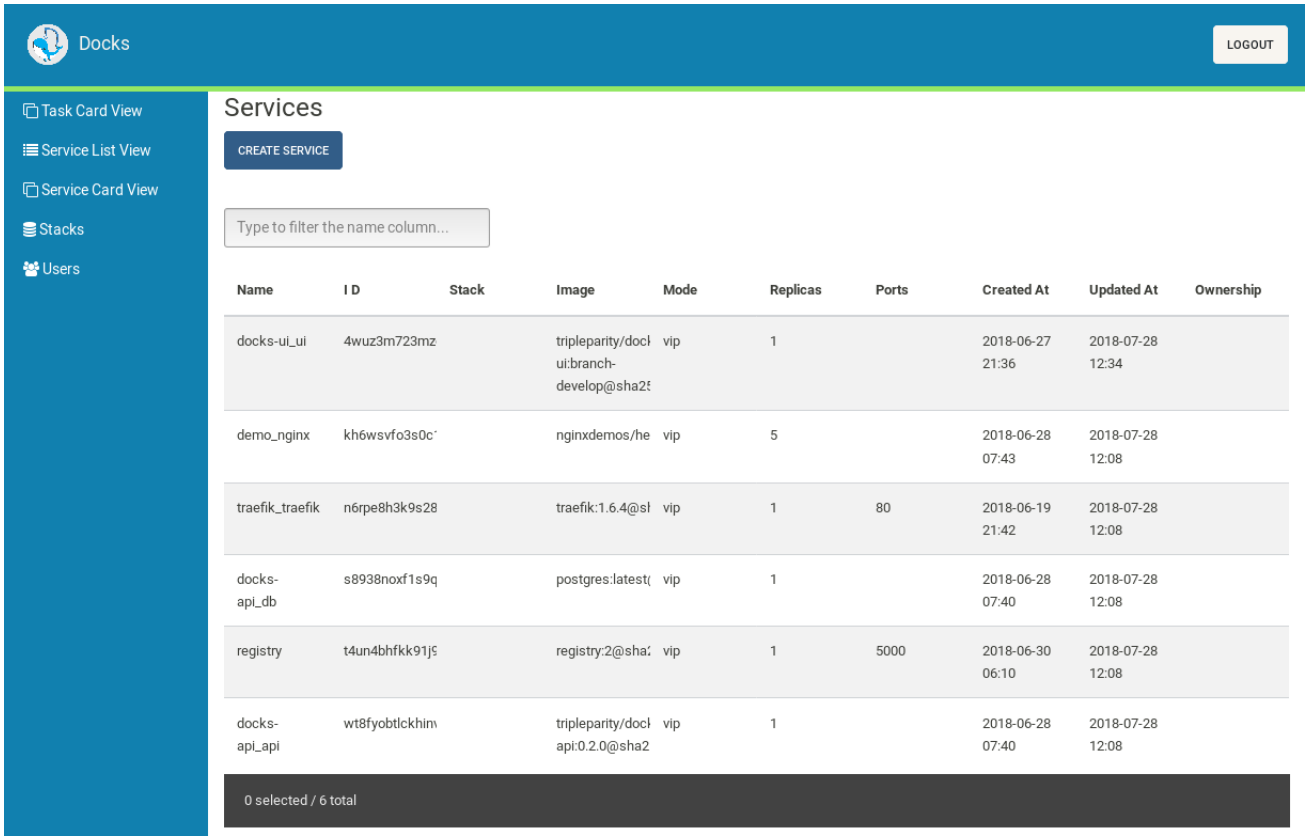


Figure 6: Service List View

Figure 7: Service Card View

5.3 Stacks

The concept of a Stack is the core feature of Docker. A Stack describes Services (applications) and how they should be deployed.

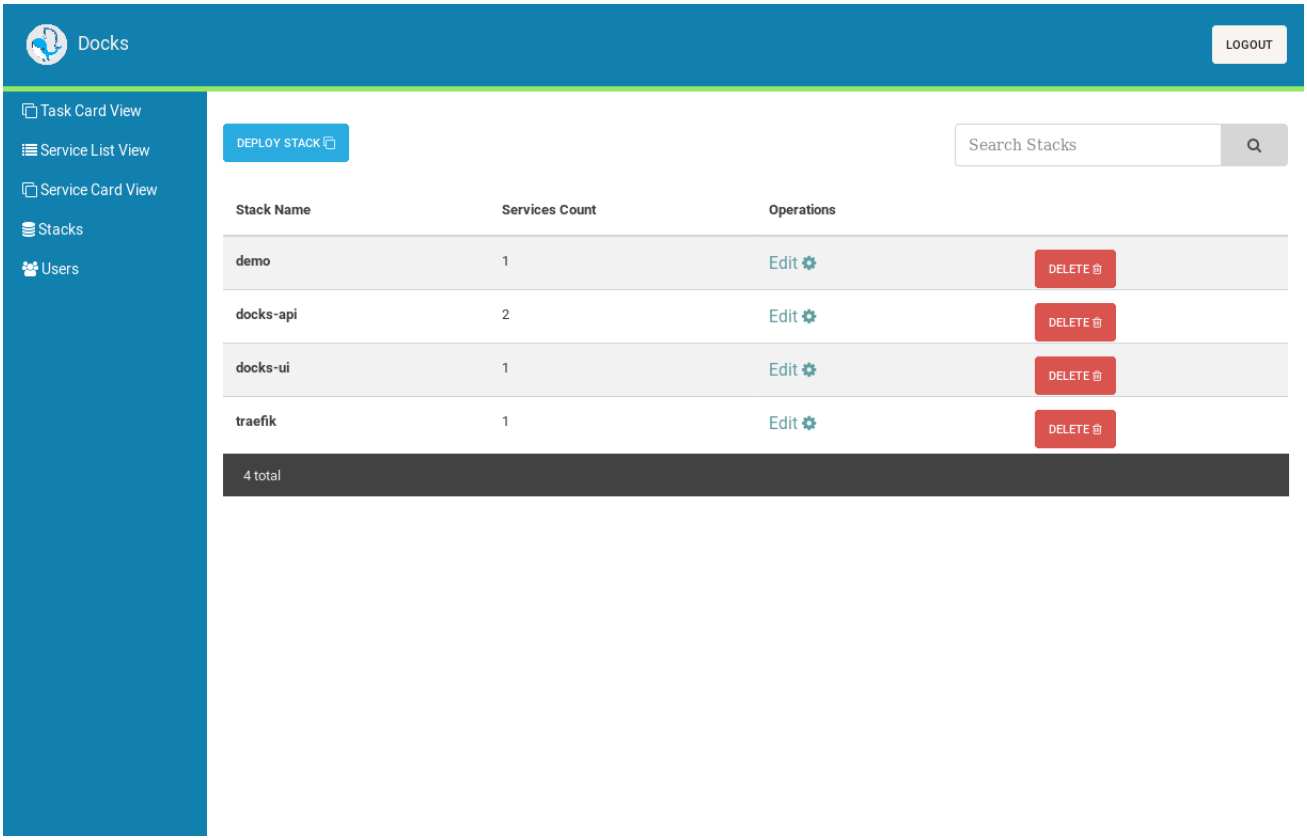


Figure 8: List of Stacks in the Swarm

5.3.1 Deploying a Stack

Docks allows the user to deploy their own Stack (Application) by uploading a Stack file. Pre-configured stacks can also be selected for deployment, such as Wordpress, Nginx, and MongoDB. The Stack File should be modified to fit the needs of the administrator and the system.

The Stack should be given a unique name to identify it in the system.

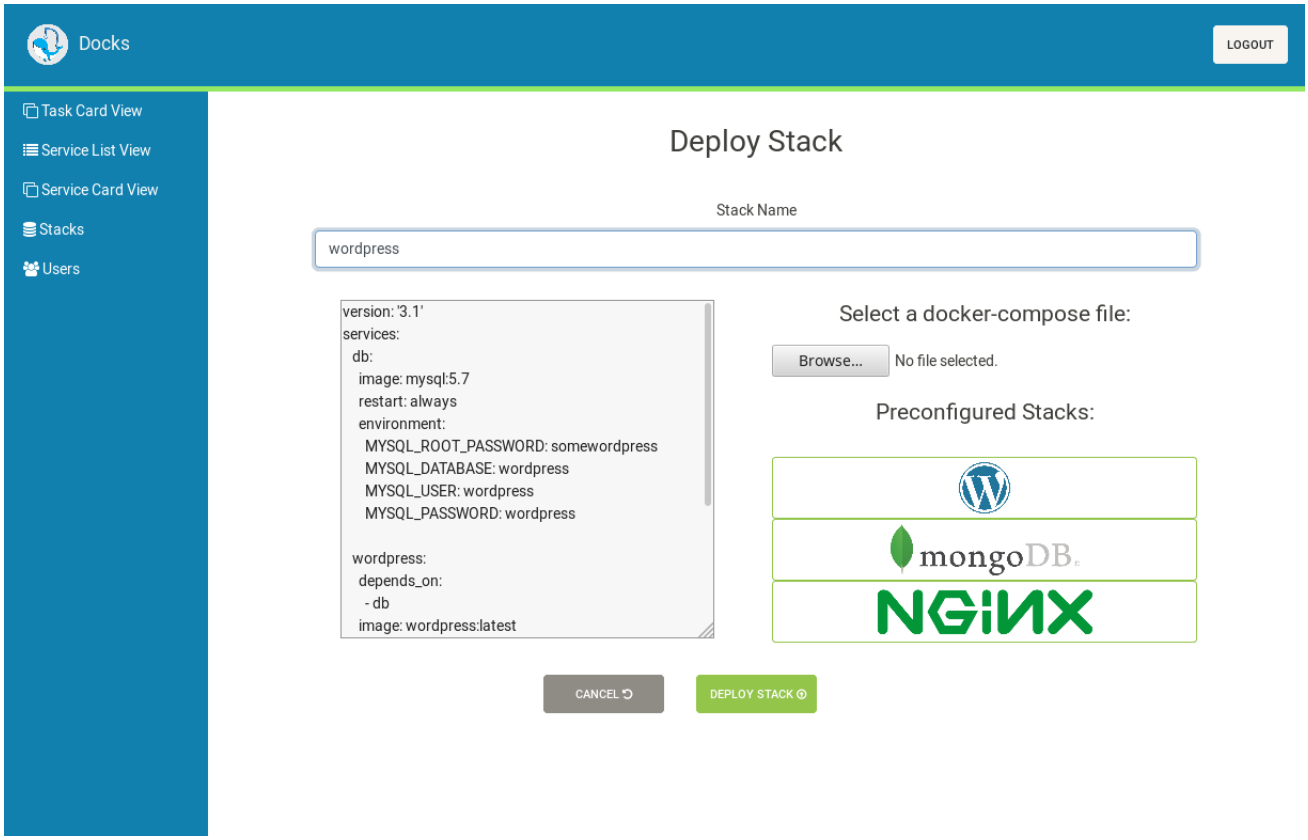


Figure 9: Deploying a Wordpress Stack

5.3.2 Updating a Stack

By uploading a modified Stack file Docker will automatically update the deployed Stack to match the Stack file. Reasons for updating the stack may include:

- Updating to a new version
- Adding a new application to the Stack
- Changing the port for a running application
- Attaching new volumes to applications for extra storage
- Changing environmental variables used for configuration

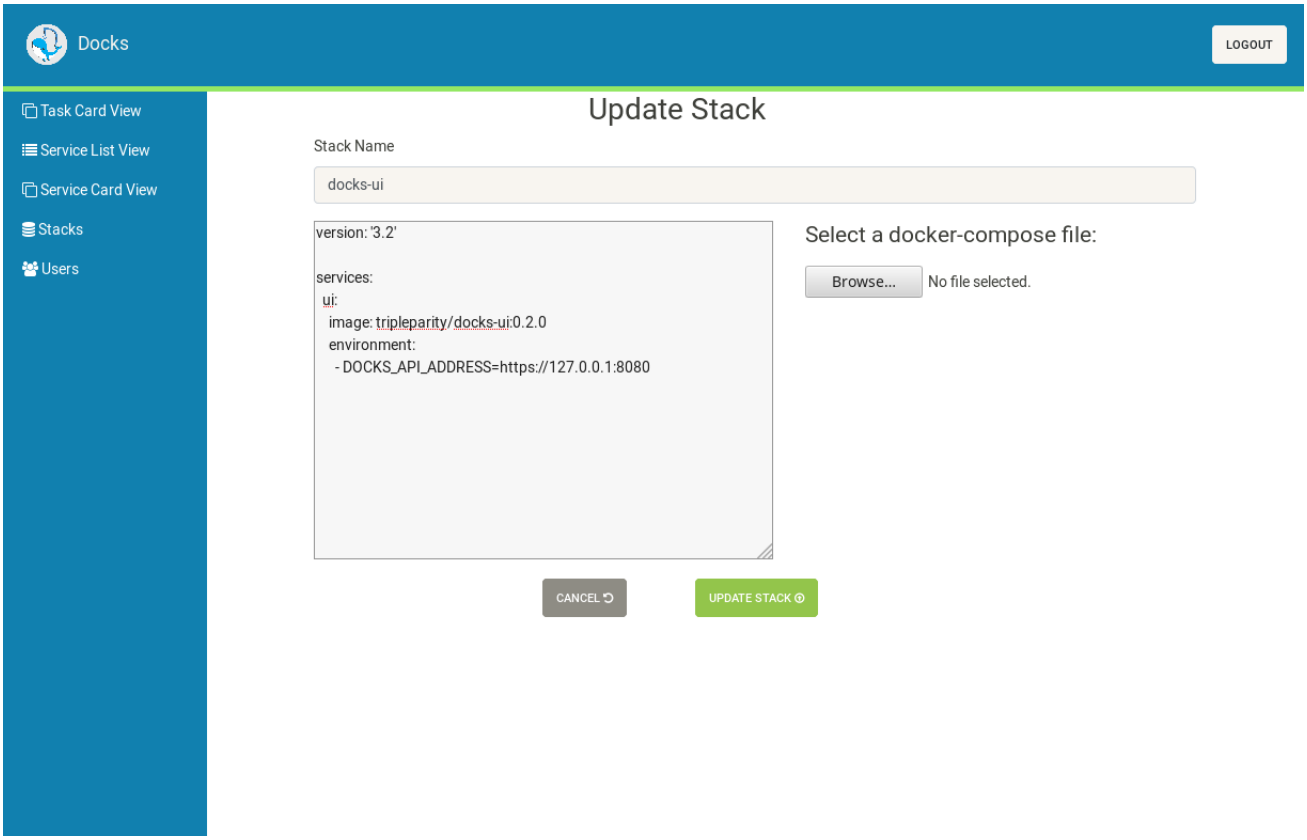


Figure 10: Updating the Docks Web Interface Stack

5.4 User Management

Only registered users can use Docks. An admin user can create more admin users. Currently there exists only admin users, so be sure to trust the person that will be using the account.

Users can be created using the Create User Page (Users -¿ Create):

Passwords can be changed using the Change Password page (Users -¿ Edit)

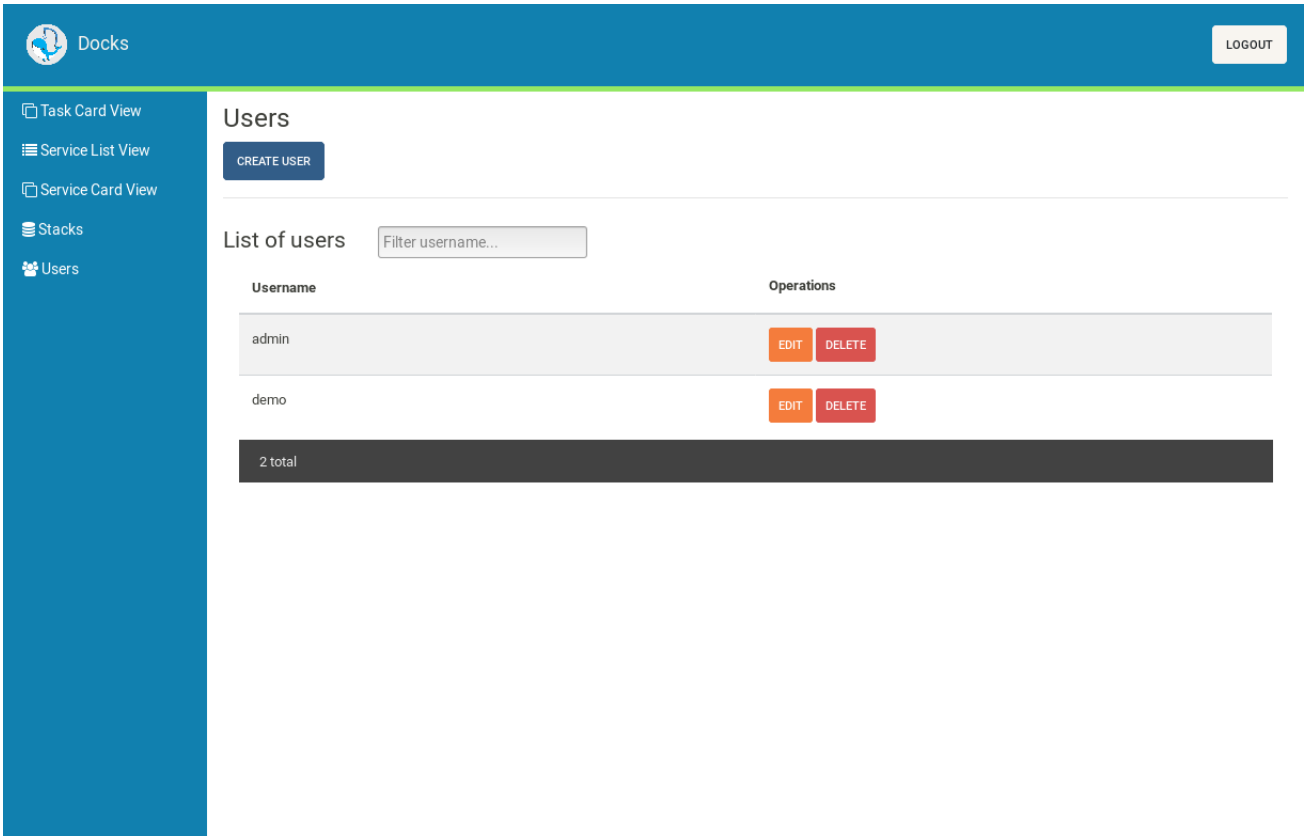


Figure 11: Users

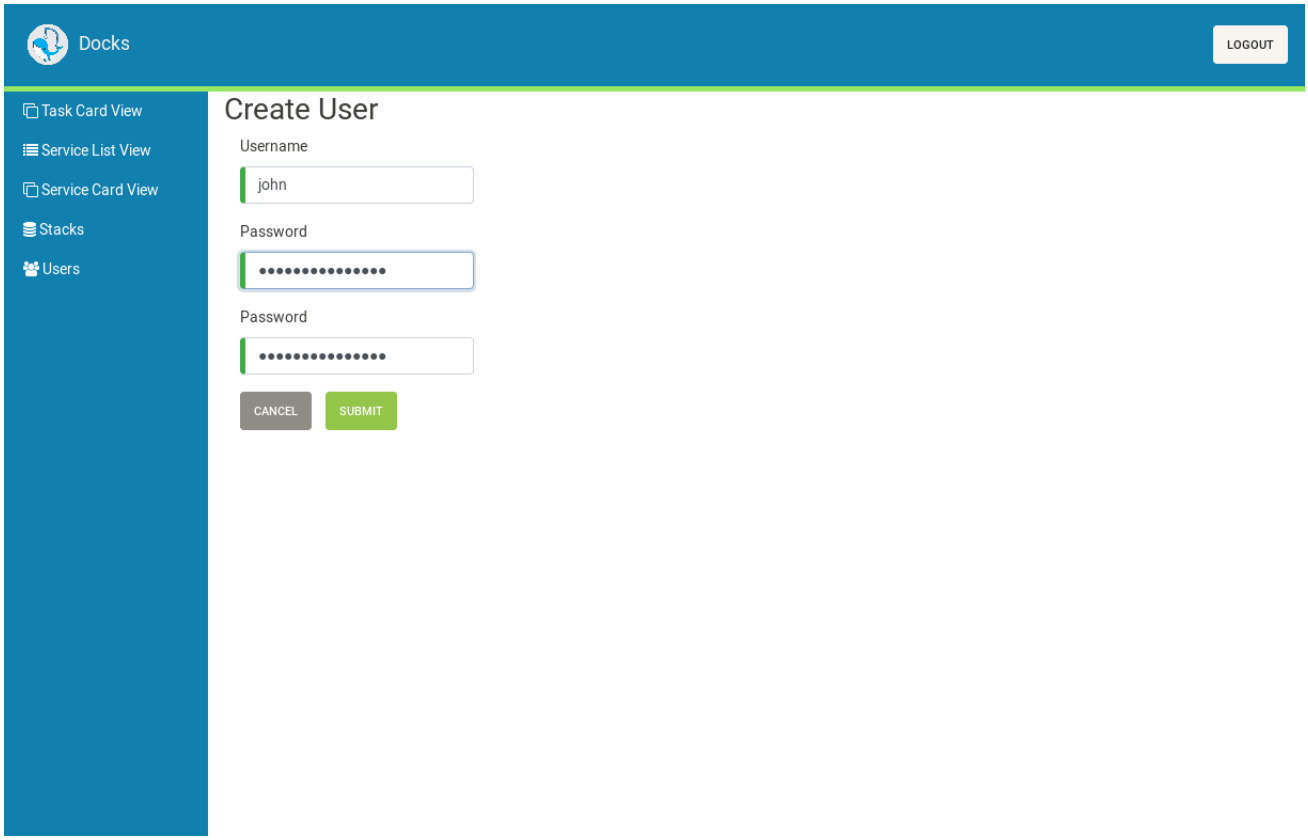


Figure 12: Users

5.5 Updating Docks

Once Docks is deployed, it can be updated as described in Updating a Stack

Alternatively the respective Stack file can be modified locally as deployed using `sudo docker stack deploy -c docker-c`

6 Troubleshooting

6.1 Error: bind: address already in use

Another service is most likely running on port `4200` , `8080` or `8081` . The ports for Docks and nginx-demo can be specified in the *docker-compose.yml* and *docker-compose-yml* files.

For example to run on port 9000 instead of 4200 make the following changes:

```
ports:
  - 4200:80
```

to

```
ports:
  - 9000:80
```