

User Manual for Docks

Team: TripleParity

Client: Compiax



Contents

1	System Overview	3
2	System Configuration	3
3	Installation	4
3.1	Configuration	4
3.1.1	Docks Web Interface	4
3.1.2	Docks API	4
4	Getting Started	4
5	Using the System	6
5.1	Tasks	6
5.1.1	Viewing Tasks	6
5.1.2	Viewing Task Details	6
5.2	Services	7
5.2.1	Viewing Services	7
5.2.2	Viewing Service Details	8
5.2.3	Viewing Service Logs	9
5.3	Stacks	9
5.3.1	Viewing Stack	9
5.3.2	Deploying a Stack	10
5.3.3	Updating a Stack	11
5.4	Node Management	12
5.4.1	Viewing Nodes	12
5.5	Network Management	13
5.5.1	Viewing Networks	13
5.6	Volume Management	14
5.6.1	Viewing Volumes	14
5.6.2	Creating Volumes	15
5.7	User Management	16
5.7.1	Viewing Users	16
5.7.2	Creating Users	17

5.7.3	Editing Users	18
5.8	Webhooks Management	18
5.8.1	Creating Webhooks	18
6	Updating Docks	19
7	Troubleshooting	19
7.1	Error: bind: address already in use	19

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](#)



Login

Username

Password

LOGIN

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 table displays the following information:

- Name of the task
- Image that the task is running
- State of the task. Green means the task is running, red indicating it has stopped and blue indicating it is preparing
- Node ID on which the task is running



The screenshot shows the Docks application interface. On the left is a sidebar with navigation links: Applications (Tasks, Services, Stacks), Infrastructure (Nodes, Networks), Data (Volumes), Users, and Webhooks. The main area has a 'Search Tasks' input field and a table of tasks. The table has columns: Name, Image, Node, and Status. The status is indicated by a colored pill: green for 'running', red for 'failed', and blue for 'preparing'.

Name	Image	Node	Status
docks.1	tripleparity/docks-api:latest@sha256:63fb1d4c	wk3bb388a4sh8bbq5gn7j0eyt	running
docks-api.1	postgres:latest@sha256:1d26fae6c056760ed5	wk3bb388a4sh8bbq5gn7j0eyt	failed
docks-api.1	postgres:latest@sha256:1d26fae6c056760ed5	wk3bb388a4sh8bbq5gn7j0eyt	running
docks.1	postgres:latest@sha256:1d26fae6c056760ed5	wk3bb388a4sh8bbq5gn7j0eyt	failed
docks.1	postgres:latest@sha256:1d26fae6c056760ed5	wk3bb388a4sh8bbq5gn7j0eyt	running
docks.1	tripleparity/docks-api:latest@sha256:63fb1d4c	wk3bb388a4sh8bbq5gn7j0eyt	failed
docks.1	tripleparity/docks-api:latest@sha256:63fb1d4c	wk3bb388a4sh8bbq5gn7j0eyt	failed
docks.1	postgres:latest@sha256:1d26fae6c056760ed5	wk3bb388a4sh8bbq5gn7j0eyt	failed
docks-api.1	postgres:latest@sha256:1d26fae6c056760ed5	wk3bb388a4sh8bbq5gn7j0eyt	failed

Figure 4: Task View

5.1.2 Viewing Task Details

From the task view, a task can be clicked to open up an new page on which details of that specific task is displayed.

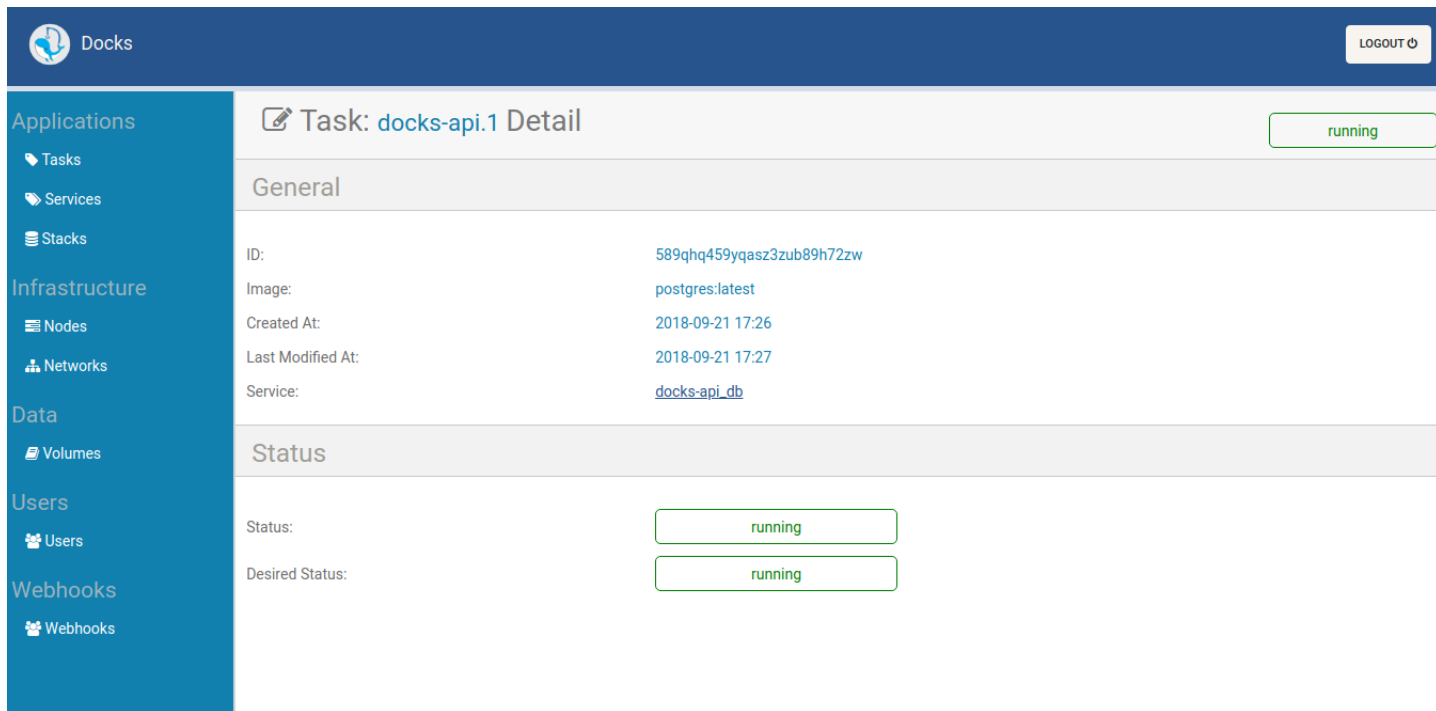


Figure 5: Task Detail View

5.2 Services

A stack consists of services. The service list view provides a sortable and searchable table for deployed services.

5.2.1 Viewing Services

The table displays the following information:

- Name of the service
- Image that the service is running
- Mode of the service
- Replicas of that services running
- Port of which the service is listening

 Docks

LOGOUT 

Applications

- Tasks
- Services
- Stacks

Infrastructure

- Nodes
- Networks

Data

- Volumes


Users

- Users

Webhooks

- Webhooks

Search Services 

CREATE SERVICE 

Name	Image	Mode	Replicas	Ports
docks-api_db	postgres:latest	vip	1	-
docks_api	tripleparity/docks-api:latest	vip	1	8080
docks_db	postgres:latest	vip	1	-

0 selected / 3 total

Figure 6: Service View

5.2.2 Viewing Service Details

From the service view, a service can be clicked to open up an new page on which details of that specific service are displayed.



Figure 7: Service Detail View

5.2.3 Viewing Service Logs

From the service detail view, the logs page can be opened to view the complete log for that service.



Figure 8: Service Log 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.

5.3.1 Viewing Stack

The table displays the following information:

- Name of the stack
- Number of services that belongs to the stack



The screenshot shows the Docks application interface. On the left is a blue sidebar with navigation links: Applications (Tasks, Services, Stacks), Infrastructure (Nodes, Networks), Data (Volumes), Users, and Webhooks. The main content area has a header with a 'Docks' logo and a 'LOGOUT' button. Below the header is a search bar labeled 'Search Stacks' and a 'DEPLOY STACK' button. The main area displays a table with two columns: 'Stack Name' and 'Services Count'. The table lists two stacks: 'docks' with a count of 2, and 'docks-api' with a count of 1. At the bottom of the table, it says '0 selected / 2 total'.

Stack Name	Services Count
docks	2
docks-api	1

0 selected / 2 total

Figure 9: List of Stacks in the Swarm

5.3.2 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 10: Deploying a Stack

5.3.3 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 11: Updating a Stack

5.4 Node Management

Docker swarm is a concept that clusters single, separate docker hosts into a single virtual docker host. It consists of manager nodes that delegate jobs and load balance between worker nodes.

5.4.1 Viewing Nodes

The table displays the following information:

- Name of the nodes
- Engine version of the node
- IP Address of the node
- Availability of the node Green means the nodes is active, red indicating it has stopped and blue indicating it is preparing
- Role of the node
- State of the node Green means the nodes is active, red indicating it has stopped and blue indicating it is preparing
- Number of CPU's assigned to the node
- Memory assigned to the node


Docks

LOGOUT

Applications

- Tasks
- Services
- Stacks

Infrastructure

- Nodes
- Networks

Data

- Volumes

Users

- Users

Webhooks

- Webhooks

Search Nodes

Q

Name	Engine version	IP address	Availability	Role	State	CPUs	Memory
trashPanda	17.05.0-ce	192.168.101.247	active	manager	ready	4	3.76 GB

1 total

Figure 12: Nodes View

5.5 Network Management

Docker networks allow a user to connect containers with services or containers and services to non-Docker workloads.

5.5.1 Viewing Networks

The table displays the following information:

- Name of the network
- Driver assigned to the network

Docks		LOGOUT
<div>Applications</div> <ul style="list-style-type: none"> Tasks Services Stacks <div>Infrastructure</div> <ul style="list-style-type: none"> Nodes Networks <div>Data</div> <ul style="list-style-type: none"> Volumes <div>Users</div> <ul style="list-style-type: none"> Users <div>Webhooks</div> <ul style="list-style-type: none"> Webhooks 	Networks	
	Name	Driver
	ingress	overlay
	docks_default	overlay
	docks-api_default	overlay
	bridge	bridge
	docker_gwbridge	bridge
	appichapsapi_default	bridge
	appichaps-network-private	bridge
	host	host
	none	null
	9 total	

Figure 13: Networks View

5.6 Volume Management

Volumes are used with persistent data generated and used by Docker containers.

5.6.1 Viewing Volumes

The table displays the following information:

- Name of the volume
- Driver assigned to volume


Docks

LOGOUT

Applications

- Tasks
- Services
- Stacks

Infrastructure

- Nodes
- Networks

Data

- Volumes

Users

- Users

Webhooks

- Webhooks

Search Volumes

Q

CREATE NEW VOLUME

Name	Driver
7279c3ac138b85072ce78011e133a005453db981623bcd7f1fba1fb893e4e10d	local
docksapi_docks_data	local
fa25bb0f7794c92a84c5f37fe7700d0bd00d4d57d3b2772983769daf3996d267	local
247ad0364a13fb101a14bc9ccb70372821b30be4507fcc3cf6ad1daee457c805	local
45c03e2d1bbfd76997c678fb7f7613672295bd3b9f7ead013a1303f72f011a63	local
92fbd08c8006466e3e5c9c12d0d1f94941d441ba093eaa23ff3f044bb93b3512	local
d36adad35e7f7a467e13123095ea9924344a06d944075b1a23aad93236119c47	local
031fc8761a0b54351aa000e5532606175097e2972437ff1555c28a80aea5f174	local
45e5041a18b938740c8ad72daa30ba4f0c7f0936b69ea46145668f8e91222f4b	local

Figure 14: Volumes View

5.6.2 Creating Volumes

Docks allows the user to create their own volumes. The Volumes should be given a unique name and assigned to a driver.

The screenshot shows the 'Docks' application interface. On the left is a blue sidebar with a menu containing 'Applications' (with sub-items 'Tasks', 'Services', 'Stacks'), 'Infrastructure' (with sub-items 'Nodes', 'Networks'), 'Data' (with sub-item 'Volumes'), 'Users', and 'Webhooks' (with sub-item 'Webhooks'). The main content area is titled 'Create Volume' and contains a form with the following fields: 'Name:' with the value 'Tardis', 'Driver:' with the value 'Custom', 'Add Driver Options' (with a help icon), and 'Add a Label' (with a help icon). At the bottom of the form are two buttons: 'CANCEL' and 'CREATE VOLUME'. A 'LOGOUT' button is located in the top right corner of the header.

Figure 15: Deploying a Stack

5.7 User Management

5.7.1 Viewing Users

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.



Figure 16: Users

5.7.2 Creating Users

Users can be created.



Figure 17: Users

5.7.3 Editing Users

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

The screenshot shows the 'Edit User' interface. The top header is dark blue with the 'Docks' logo on the left and a 'LOGOUT' button on the right. A blue sidebar on the left lists navigation categories: Applications (Tasks, Services, Stacks), Infrastructure (Nodes, Networks), Data (Volumes), Users, and Webhooks. The main content area is titled 'Edit User' and contains three input fields: 'Username' (with 'admin' entered), 'Password', and 'Confirm Password'. At the bottom are 'CANCEL' and 'SUBMIT' buttons.

Figure 18: Users

5.8 Webhooks Management

Webhooks make it easy for our application to communicate flexibly with external services (such as slack). This allows us to generate slack notifications

5.8.1 Creating Webhooks

This pages allows you create webhooks.

The screenshot shows the 'Docks' application interface. The top header is dark blue with the 'Docks' logo and a 'LOGOUT' button. The left sidebar is blue and contains a menu with the following items: Applications (Tasks, Services, Stacks), Infrastructure (Nodes, Networks), Data (Volumes), Users, and Webhooks (Webhooks). The main content area is titled 'Webhooks' and contains a form with the following fields: 'Name' (text input), 'Webhook URL' (text input), 'Volumes' (checkbox), 'Network' (checkbox), 'Service' (checkbox), 'Node' (checkbox), and 'Image' (checkbox).

Figure 19: Webhooks Creation

6 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`

7 Troubleshooting

7.1 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
```