Requirements and Design - Docks

TripleParity

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to enumerate the requirements of the project and provide the design of the architecture.

1.2 Product Scope

Docks is a system to visualize a Docker Swarm. It's purpose is to provide a graphical interface to view and manage the Swarm that is easier and quicker to use than the Command Line Interface.

The GUI will be in the form of a Web Application. This provides the benefit of managing the Docker swarm using any device that has a web browser.

2 Overall Description

2.1 Product Perspective

To understand the purpose of Docks and the benefits it provides we first need to describe what Docker is.

> Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications, whether on laptops, data center VMs, or the cloud.
> - docker.com

While it is possible to manage a Docker Swarm using the provided Command Line Interface, it requires experience and access to a terminal. Docks will make it possible for graphical interface orientated users to manage a Docker Swarm.

2.2 Product Functions

A high level summary of the required functions provided by the graphical user interface.

- All management and view operations require authentication and the correct authorization
- Fine-grain permission control for user accounts
- Account roles with different permissions
- View Docker Nodes that are part of the Swarm
- View Services that are running in the Swarm
- View Tasks running in the Swarm
- Start and stop services
- Remove services
- Deploy a Stack to the Swarm using a docker-compose file
- Deploy a Service to the Swarm
- View Networks in the Swarm
- View Volumes in the Swarm
- Deploy images from a private repository

3 External Interface Requirements

3.1 Software Interfaces

Since the frontend cannot securely interface with the Docker API, an intermediate interface will be developed (Docks API). The Docks API will communicate between the frontend (Docks-UI) and the Docker API. The Docks API will provide a simplified interface for interacting with the Docker API.

4 System Features

4.1 Authentication and Authorization

4.1.1 Description

Due to the power the Docks system exposes only Authorized users should be able to use the system. Resources will be managed by Teams. Account roles are assigned in the context of a team, i.e A user can be a Guest in one team and a Super User in another team.

4.1.2 Functional Requirements

- R1.1 The system shall allow an authorized user to interact with the Docks API
- R1.2 The system shall allow a user to perform actions only when they are authorized to do so.
- R1.3 The system shall provide a global administrative account role without restrictions
- R1.4 The system shall provide Teams
- R1.5 The system shall provide the ability to add Users to Teams
- R1.6 The system shall only allow a User to manage resources that are part of their team.
- R1.7 The system shall provide the following account roles within Teams: (Role (Permissions))
  - R1.7.1 Team Leader
  - R1.7.2 Super User
  - R1.7.3 Normal User
  - R1.7.4 Guest
- R1.8 Roles shall have the following permissions:
  - R1.8.1 Team Leader
    - Add users to team
    - Bind Mount
    - Deploy Resources
  - R1.8.2 Super User
    - Bind Mount
    - Deploy Resources
  - R1.8.3 Normal User
    - Deploy Resources
  - R1.8.4 Guest
    - Read only access. No access to Inspect
- R1.9 The system shall provide the following user management features to be used by administrative accounts
  - R1.9.1 Create new user

  - R1.9.2 Remove user
  - R1.9.3 Update user password
  - R1.9.4 Set permissions as stated in R1.3

4.2 Docker Swarm Management

4.2.1 Description

Docks needs to provide features for managing the Docker Swarm. While it is already possible to fully mange the Docker Swarm using the Command Line Interface, Docks will provide a simplified web interface.

4.2.2 Functional Requirements

- R2.1 The system shall display all nodes in the swarm.
- R2.2 The system shall display all services running in the swarm.
- R2.3 The system shall display the following attributes about a Task:
  - R2.3.1 Container Name (if set)
  - R2.3.2 Uptime
  - R2.3.3 Container ID
  - R2.3.4 State of the container (running/stopped)
  - R2.3.5 Image used to create the task
- R2.4 The system shall allow a user to stop a running service
- R2.5 The system shall allow a user to start a stopped service
- R2.6 The system shall allow a user to upload a docker-compose file to deploy a Stack
- R2.7 The system shall allow a user to remove a stopped service
- R2.8 The system shall allow a user to destroy a Stack
- R2.9 The system shall display the networks on a Node
- R2.10 The system shall display the volumes on a Node
- R2.11 The system shall be able to deploy images from a private repository
- R2.12 The system shall display the following attributes about a Service
  - R2.12.1 Service Name
  - R2.12.2 Stack (if deployed from Stack)
  - R2.12.3 Tasks running in the service
  - R2.12.4 State of the Service
  - R2.12.5 Image used to create the Service

5 Architecture

5.1 Subsystems

The Docks system consists of two main subsystems. The Docks API Server and the Docks UI.

5.1.1 Docks API Server

The Docks API Server acts as a middleman between the UI and the Docker API running on the Manager Node. The Docks API also adds a layer of authentication and session management over the Docker API.

5.1.2 Docks UI

Docks UI is the web interface for managing the Docker Swarm through the Docks API. It consumes the API provided by the Docks API server.

5.2 Domain Diagram

The Domain diagram gives highlights the relationship between the objects in the system.

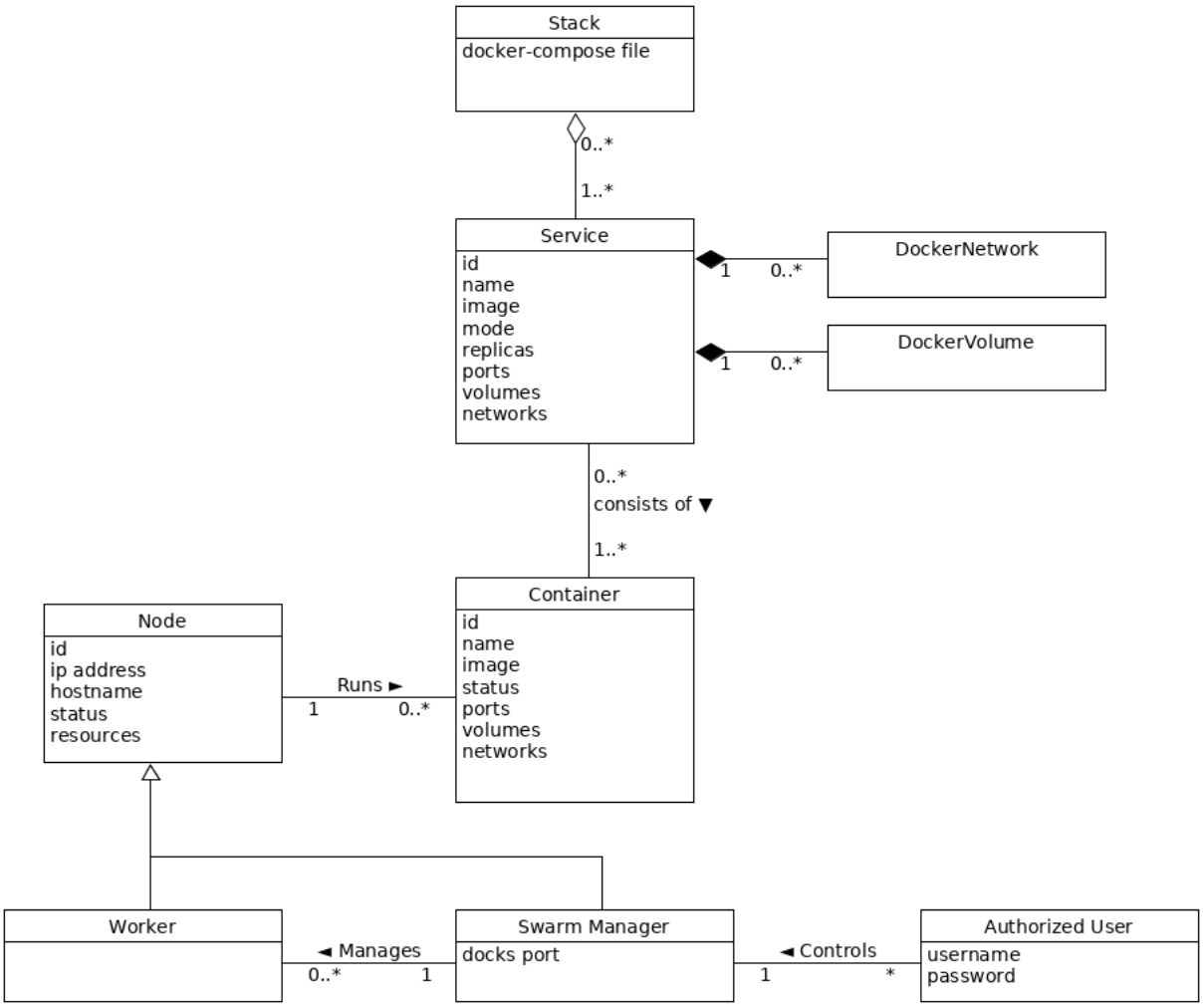

Figure 1: UML Domain Diagram for the Docks System

5.3 Deployment Diagram

The Deployment diagram shows the architecture from the device perspective.

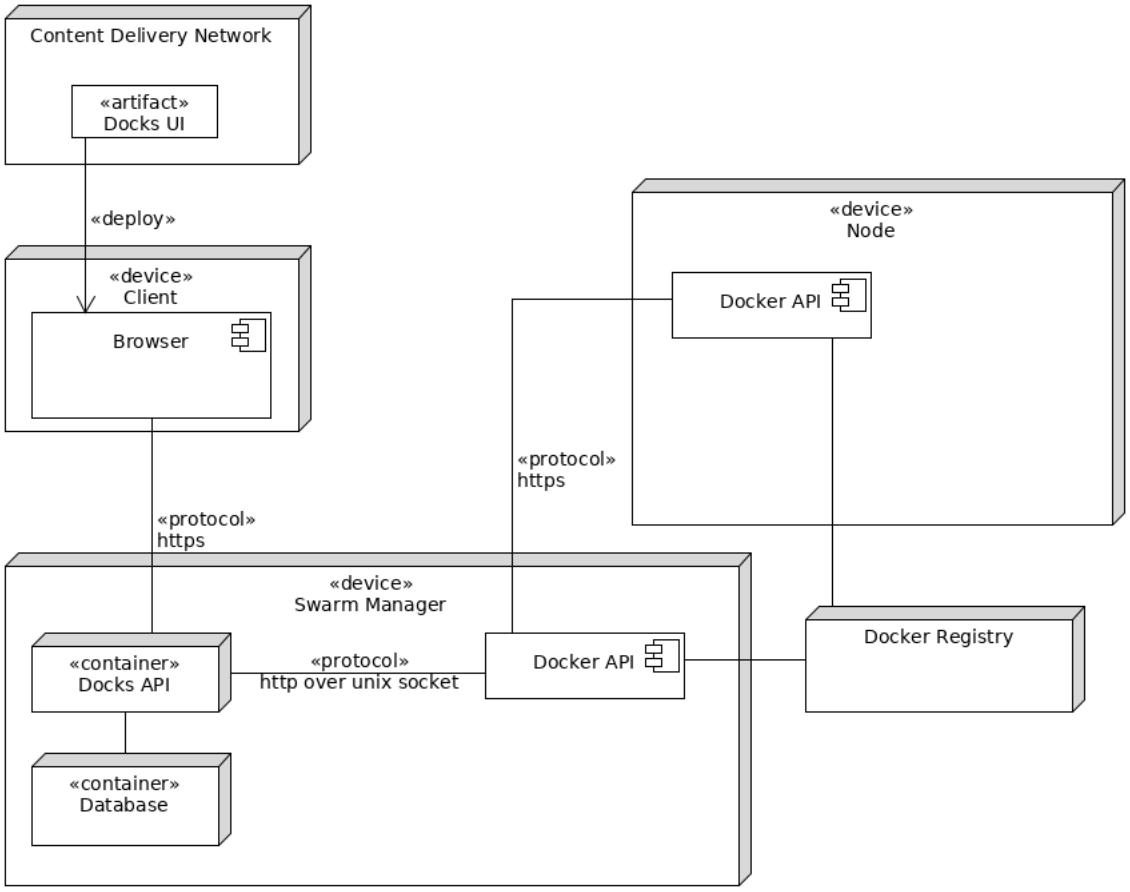

Figure 2: UML Deployment Diagram for the Docks System

5.4 ERD Diagram

The Deployment diagram shows the database architecture for the Docks API system

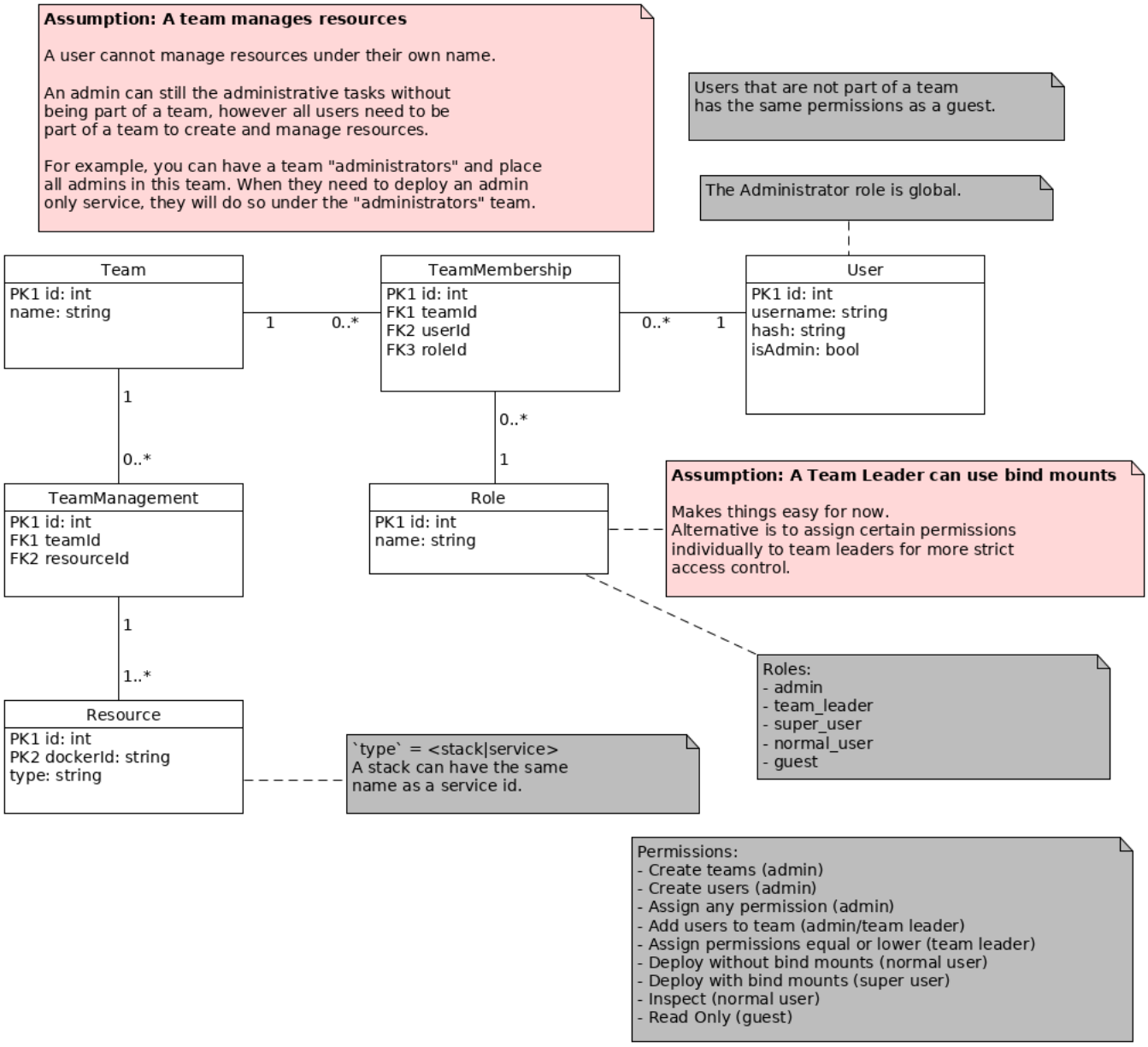

Figure 3: ERD diagram for the Docks API database

5.5 Architectural Structure

The User Interface uses the Model View Controller architecture. Nodes and Containers have a data model. The user interacts with the view to manipulate the data model. The view is updated when the data model retrieves data using an N-Tier architecture. The 3-Tier architecture can be seen by the actor interacting with the view, the request is then delegated to the models, which in turn communicate with other objects to retrieve and set the required data from the Docker API server and Docker Swarm.

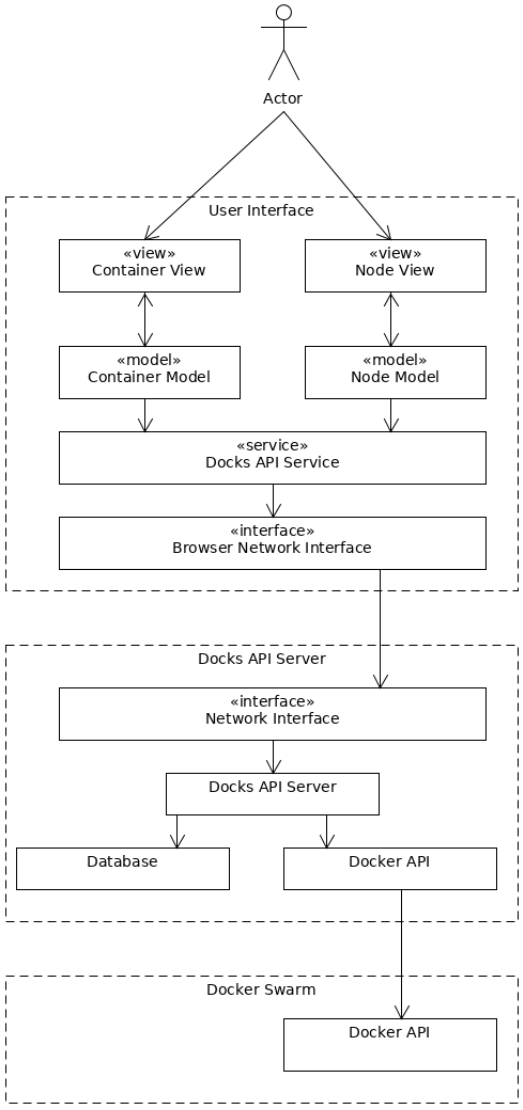

Figure 4: MVC and 3-Tier Architecture diagram for the Docks system