Pivotal Interface for Kubernetes

Product Requirements Document v2

Position	Name	Contact
Team Lead	Jesmar Castillo	jesmar@ucsb.edu
Team Scribe	Marco Chavez	mchavez00@ucsb.edu
Developer	Durva Kapadne	durva@ucsb.edu
Developer	Jack Liu	jackliu@ucsb.edu
Developer	Kindy Tan	ktan@ucsb.edu

Product Name: kubernetes konekt Team Name: The Goodfellas

1. Background

Kubernetes is a platform that is made to revolutionize the way that container based applications are released and tested. Inside of a container we can package an application and all its dependencies to create an isolated environment. This allows applications to run independently from the host operating system, eliminating errors experienced from running an application on different machines. This makes container based applications more portable than others. With the added benefit of containers not needing a guest operating system, deployment has become much more lightweight.

Kubernetes builds on top of this by automating the deployment, scaling, and management of containerized applications within a cluster of computers. It handles the scheduling and deployment of containers. Without Kubernetes, containers have to be manually scheduled and it's easy to see how this can be time consuming for an application that depends on many containers with short life cycles. Through a master server, a cluster and its nodes can be managed, distributing containers among the nodes. Once deployment specifications are passed to the Kubernetes cluster master server it is up to the master to make sure the specifications are true at all times.

2. Existing Technologies

Currently there are companies that host clusters that users can manage and upload containers to, such as AWS and GCP. However, there are no mainstream companies that offer individuals the opportunity to provide managed Kubernetes Clusters as a service to other individuals. These interfaces sometimes require the use of a command line, a tool that can add unnecessary complexity. Users will have to learn the documentation of pulling containers and uploading them to a cluster.

Scheduling must also be done manually by the user. Through a YAML file they can specify a Cron Job and submit it to the Kubernetes cluster. This YAML file must also specify the container. If the user wishes to schedule his container on a remote cluster, they will also need valid certificates to use the Kubernetes API.

3. Project Overview

The lack of connectivity between users with containers and users with clusters acts as a barrier for deployment. Currently there is not an interface that streamlines connecting these two groups. To solve this issue our Minimum Viable Product (MVP) will be a Web UI that will connect users all over the world and allow them to run containers on remote clusters. Our MVP will provide users a dashboard to upload and manage containers. Users will be able to view the status of their uploaded containers. Providers will have a dashboard to view and manage their clusters.

We are removing much of the overhead associated with scheduling containers. The web service will automate tasks and eliminate the need to execute commands on a terminal to run a container.

4. Project Specifics

The frontend of the Web UI consists of JSP, CSS, and Java elements. The backend makes use of the Spring Boot and Spring MVC frameworks. Spring Boot provides the initial configuration for Spring Applications and dependencies. Spring MVC provides many of the tools for front and backend interactions by taking a model, modifying or attaching data from the database, and sending it to the view to display it to the user. Models are also used to read information from forms filled out by users in a secure manner.

The Hibernate framework is used as an object-relational mapper tool for Java, but provides other APIs as well. The Hibernate APIs we are using include Hibernate ORM which allows us to map class objects to table entries on a database and Hibernate Validator to validate that the user input has correct structure using annotation driven development. Hibernate Validator also passes error messages back to JSP when invalid information is detected.

The security of the website will make use of Spring Security. This framework streamlines user authentication and access control. This allows for easy management of user logins and registrations. Spring Security keeps track of current user logged in and roles associated to the user.

The Web UI will be hosted on Pivotal Cloud Foundry. MySQL database will be hosted on Pivotal Web Service (PWS) and accessed with Hibernate ORM. Google Cloud Platform (GCP) will be used to help with launching and testing Kubernetes for our personal use.

Scheduling of the containers will be automated through a generated YAML file. When the user selects their container they will also select the scheduling time and desired cluster. These arguments will be used to help generate the YAML file. This file is then executed within the cluster to schedule the container. The Kubernetes API is used to access the provider's clusters. We can easily send a commands over to the designated cluster and be able to access it.

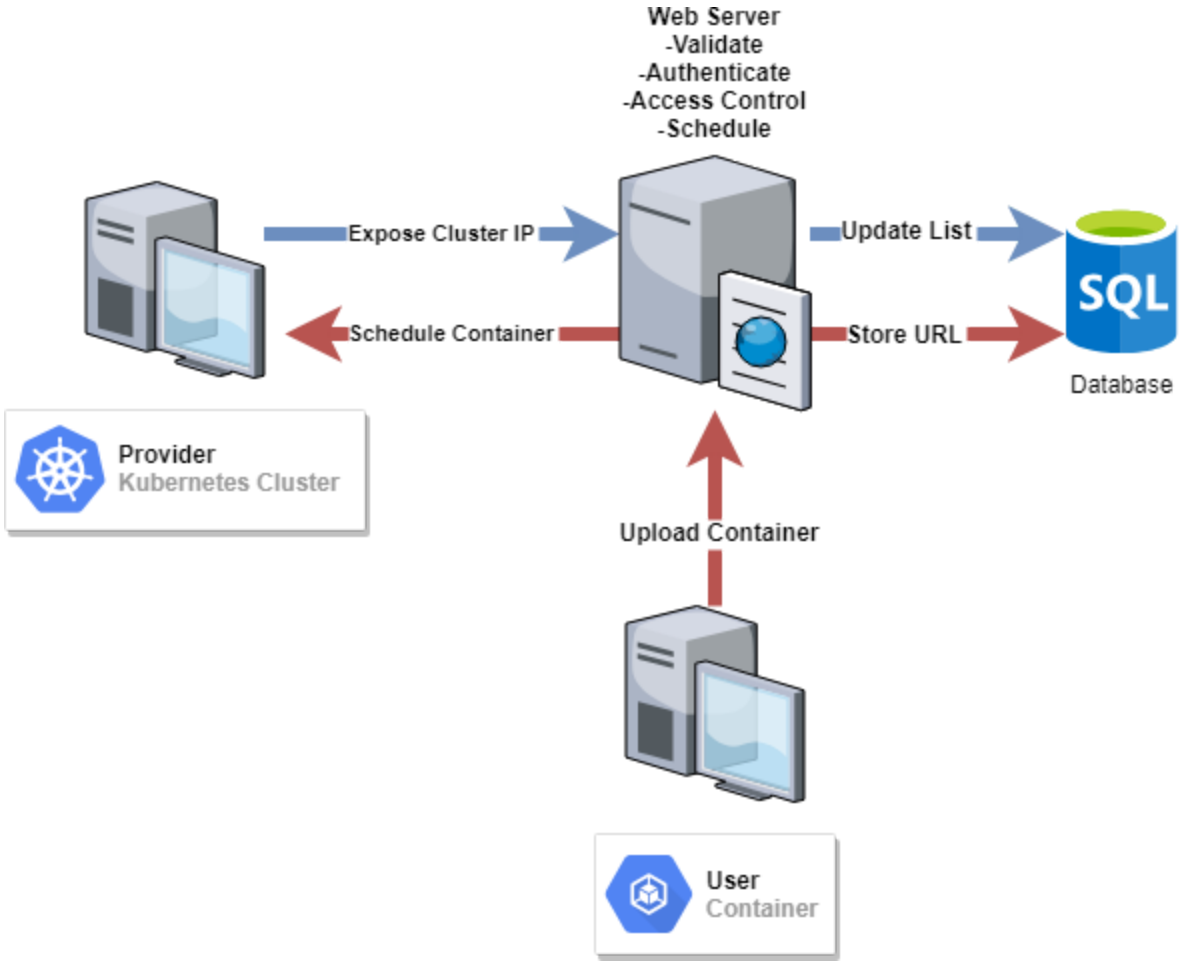
5. Assumptions

We are assuming that users have containers that are fully functional and smoothly running. Users will be responsible for managing all their own containers and making sure their containers run smoothly so that providers are able to run them on their clusters. Containers uploaded to the website should be hosted on DockerHub and the user should provide a link to the image. This ensures containers are properly formatted to run on a Kubernetes cluster.

We are also assuming that most users understand how to use Kubernetes or have a basic understanding thereof. Providers have already set up their clusters before being listed to other users and are ready to accept containers. The provider must supply certificates associated with the Kubernetes clusters they upload so that the web service can use the Kubernetes API against their cluster.

6. System Architecture Overview

High Level Diagram



User Interaction and Design

Users of the service must register for a user account, provider account, or both. Users must choose a unique username and provide an email that has not been registered before. The website will validate their form and if the information provided is valid they continue to a confirmation page. The user will then be able to login and access the user/provider dashboard depending on what they registered for.

Providers are users that already have a Kubernetes cluster running. To list their cluster onto our service, they shall fill out a form with the cluster information needed to access and get authority to their cluster, they then will submit the form for a review. Once the

cluster has been reviewed and approved the cluster will be visible on their dashboard. When a user requests to run a container on their cluster the provider will be able to accept/decline the request. Providers also have ability to hide clusters so that they're not listed publicly, if that is desirable for them.

A user is anyone who has containers uploaded on DockerHub and would like to rent a Kubernetes cluster provided by a provider. In order for a user to upload their container they must provide a link to their container on DockerHub. The website will confirm the link is valid by accessing the link, then the approved container will appear on the user dashboard. A user can upload a container to a cluster by choosing a container, cluster, deployment preferences, and/or scheduling preferences. The provider of the Kubernetes cluster will then accept/reject requests. Assuming the provider accepts to run the container, the website will use Kubernetes API to deploy the container onto the cluster. When users no longer want their container running on a cluster they may request to stop the cluster and remove their container from the cluster. A user may also request to delete a previously uploaded container. The website will then remove all information related to the container from the website and dashboard, and display an updated table of uploaded containers.

7. User Stories

USER STORY	ACCEPTANCE CRITERIA
As a user/provider, I can register to the website. **Pivotal Tracker Link:** **https://www.pivotaltracker.com/story/show/161301075**	**Scenario 1:** User/provider has not registered. Given user/provider has never registered before. And user/provider is not currently logged in. When user/provider fills out registration form. And user/provider clicks submit. And account has been created. Then user/provider will be prompted to confirm account via email. **Scenario 2:** User/provider is registered. Given the user/provider has an account. And the user/provider is not currently logged in. When user/provider fills out registration form. And user/provider clicks submit Then error message will appear that account already exist.

As a user/provider, I can login to the website. **Pivotal Tracker Link:** **https://www.pivotaltracker.com/story/show/161301077**	**Scenario 1:** User/provider is registered. Given the user/provider is not logged in And enters a valid username and password When the user/provider clicks submit Then user/provider is redirected to home page. **Scenario 2:** Invalid username or password. Given the user/provider is not logged in And enters an invalid username or password When the user/provider clicks submit Then an invalid message will appear And will be asked to re-enter their information. **Scenario 3:** User/provider forgets username. Given the user/provider is not logged in And they are registered When they click username recovery Then they will receive an email with their username And will be redirected to the login page. **Scenario 4:** User/provider forgets password. Given the user/provider is not logged in And they are registered When they click password recovery Then they will receive an email to change their password And will be redirected to the password change page.
As a provider, I can upload a cluster IP to my account so that it is listed publicly. **Pivotal Tracker Link:** **https://www.pivotaltracker.com/story/show/161300973**	**Scenario 1:** Provider has not listed his/her cluster already. Given the provider is logged in. And they have their cluster IP address When the provider inputs their IP address Then the IP address is posted And a confirmation message pops up. **Scenario 2:** Provider attempts to upload a duplicate of cluster Given provider has previous uploaded cluster provider is attempting to upload. When provider inputs their IP address Then the posting is ignored And a rejection message pops up.
As a provider, I can view clusters I have	Provider wants to see clusters that he/she has

previously uploaded. **Pivotal Tracker Link:** **https://www.pivotaltracker.com/story/show/161300999**	previously uploaded. Given provider is logged in and has cluster uploaded. Then provider can see all clusters the he/she has previously uploaded.
As a provider, I can delete previously uploaded cluster **Pivotal Tracker Link:** **https://www.pivotaltracker.com/story/show/162331277**	Provider wants to delete previously upload cluster. Given provider is logged in and has uploaded clusters. Then provider has the option to delete cluster from website. And see a success/failure message upon completion.
As a provider, I can hide cluster so that they are not publicly listed as available without deleting the cluster. **Pivotal Tracker Link:**	Provider wants to hide cluster from public without deleting it. Given provider is logged in and has clusters uploaded. Then provider can choose to make clusters private so that they are only visible on their dashboard. And cluster is not listed to users.
As a provider, I can see current status of uploaded clusters. **Pivotal Tracker Link:** **https://www.pivotaltracker.com/story/show/161734649**	Provider wants to see what containers are uploaded on their clusters. Given provider is logged in and has clusters uploaded. Then provider can see the name of the containers uploaded and status (running,stopped,container not assigned).
As a provider, I can accept request to run a container on my cluster. **Pivotal Tracker Link:**	Provider is receives container to run. Given provider is logged in and a user has requested to run container on cluster. Then provider has the option to accept/decline to run a container on his/her cluster.
As a user, I can upload a container. **Pivotal Tracker Link:** **https://www.pivotaltracker.com/story/show/161301100**	User has not uploaded container before. Given user is logged in and on his/her dashboard. Then user has option to upload container. When they upload the container Then their dashboard will update And a confirmation message pops up.

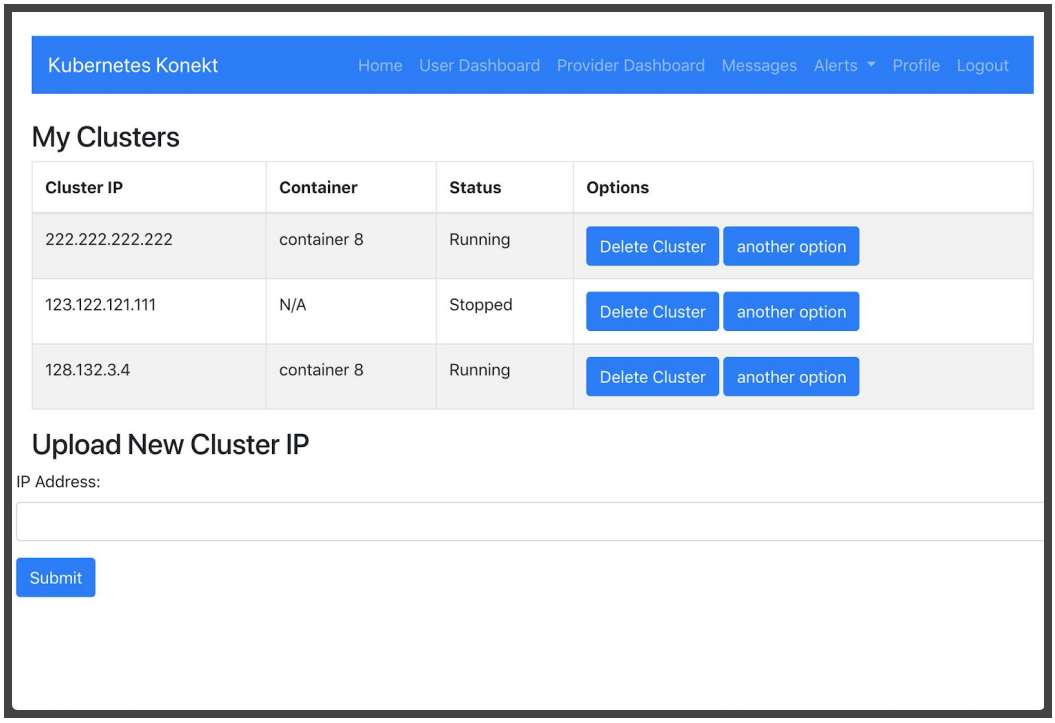
As a user, I can schedule a container to run on a cluster at a different time. **Pivotal Tracker Link:**	User has their container already uploaded Given that they found which cluster to use And when they want the cluster to run the container When they schedule the cluster in their dashboard Then the request is sent to the provider And a confirmation/rejection message appears after the provider confirms/denies the request.
As a user, I can run a Cron Job. **Pivotal Tracker Link:**	User has a Cron Job and an image to run. Given that user has already uploaded their container as an image. And uploaded their Cron Job file Then provider will run uploaded Cron Job and uploaded image on cluster
As a user, I can set scheduling preferences for container that do not have a Cron Job included in the image already. **Pivotal Tracker Link:**	User wants to upload container to cluster with no Cron Job. Given user is logged in and has containers uploaded with no Cron Job. Then user can fill out form with scheduling preferences. And website will create Cron Job and recreate image to include Cron Job.
As a user, I can delete an container. **Pivotal Tracker Link:** **https://www.pivotaltracker.com/story/show/161301153**	User does not need image anymore Given that user has already uploaded an image And user no longer needs the image Then user is be able to delete image by clicking on a ‘delete container button And user will be prompted with a message to confirm deletion, where the user can then confirm or cancel to stop deletion
As a user, I can view my payment information on my dashboard after I have ran my container on a cluster. **Pivotal Tracker Link:**	User has deployed container on a cluster Given that they have navigated to their account settings And they selected the payment information option Then the user will gain access to their payment information And will be given an option to update their payment information
As a user, I can communicate (chat) with	**Scenario 1:** chat between two users

the cluster owner. **Pivotal Tracker Link:**	Given that user wants to start a chat with another user And both users have a Kubernetes Konekt account Then user can start a chat by clicking on the 'start chat' button And user will be connected and they will be able to chat **Scenario 2:** group chat Given that user wants to start a chat with other users. And all users have Kubernetes Konekt accounts Then a users can start a chat by clicking on the 'start chat' button and add other users.
As a user, I can see available clusters that are listed publicly. **Pivotal Tracker Link:** https://www.pivotaltracker.com/story/show/161301093	User wants to browse available clusters. Given that they have navigated to their dashboard And selected the available clusters option Then the user will be displayed a list of available clusters And will be given a description with information about each cluster.
As a user, I can manage my user settings on a dashboard. **Pivotal Tracker Link:**	User wants to change settings (email, password, payment information). Given that the user is logged into their account. And has selected the account settings. Then the user will be shown a list of settings that they can change. And they will be sent a confirmation email.
As a user, I can see current status of uploaded containers **Pivotal Tracker Link:** https://www.pivotaltracker.com/story/show/162229331	User wants to see which containers are running on a cluster. Given user is logged in. Then user can see table displaying all container that are currently uploaded and their status (running, stopped, not assigned to cluster).
As a user, I can remove container from cluster. **Pivotal Tracker Link:**	User wants to remove container from cluster. Given the user is logged in and has container currently running on a cluster. Then user can choose to stop running container

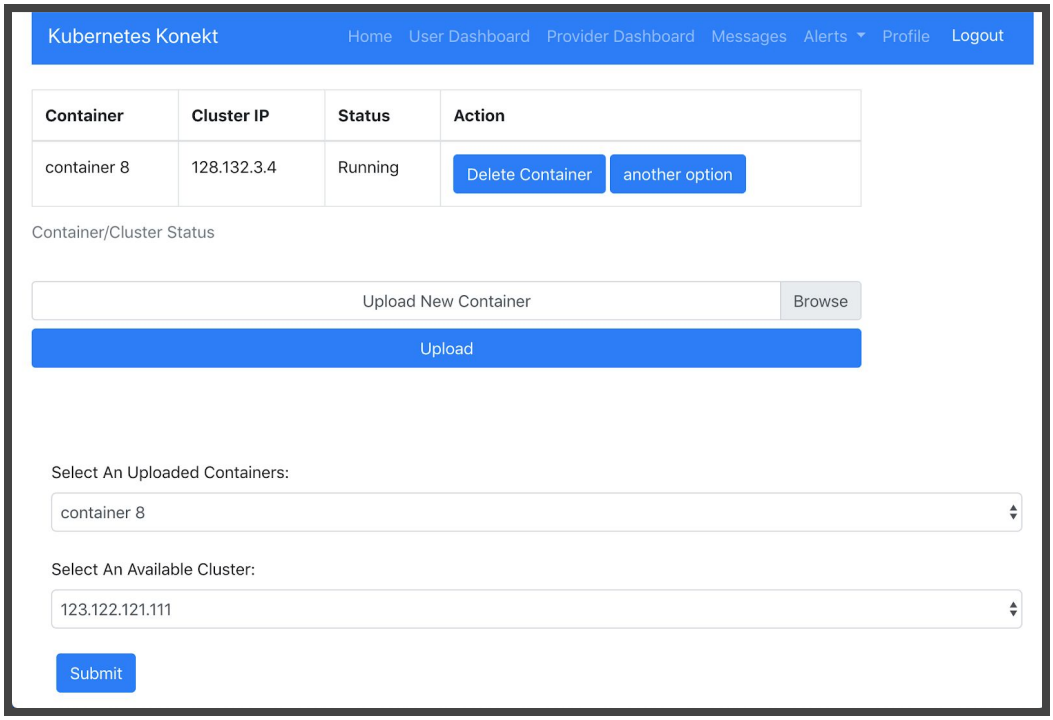
	on cluster and remove container from cluster.
As a user, I can report a faulty cluster. **Pivotal Tracker Link:**	User wants to report problems with cluster. Given user is currently running container on a cluster. And the cluster is not functioning properly, namely cluster is inaccessible. Then user can report faulty cluster to admin and cluster owner.

8. Dashboards

Provider Dashboard

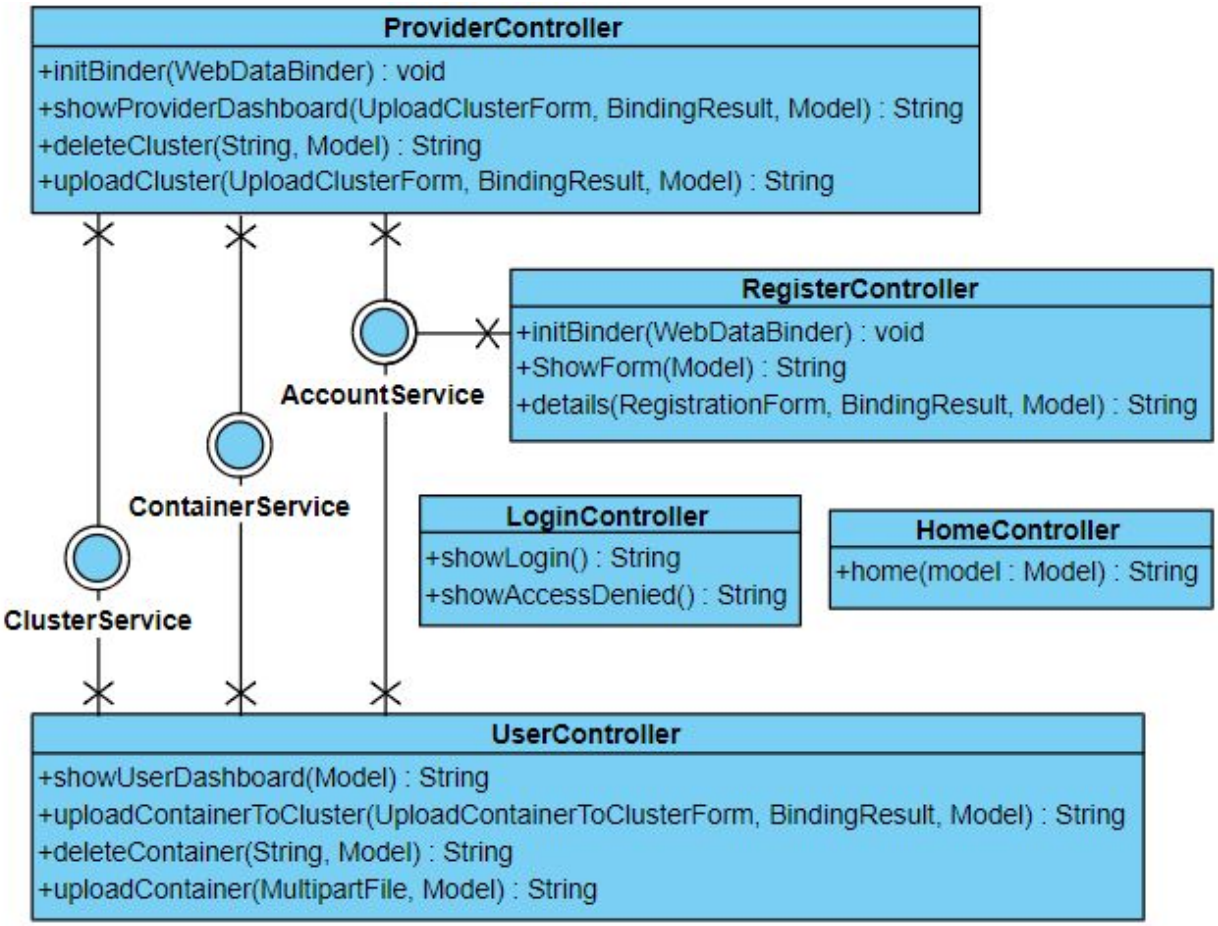


User Dashboard

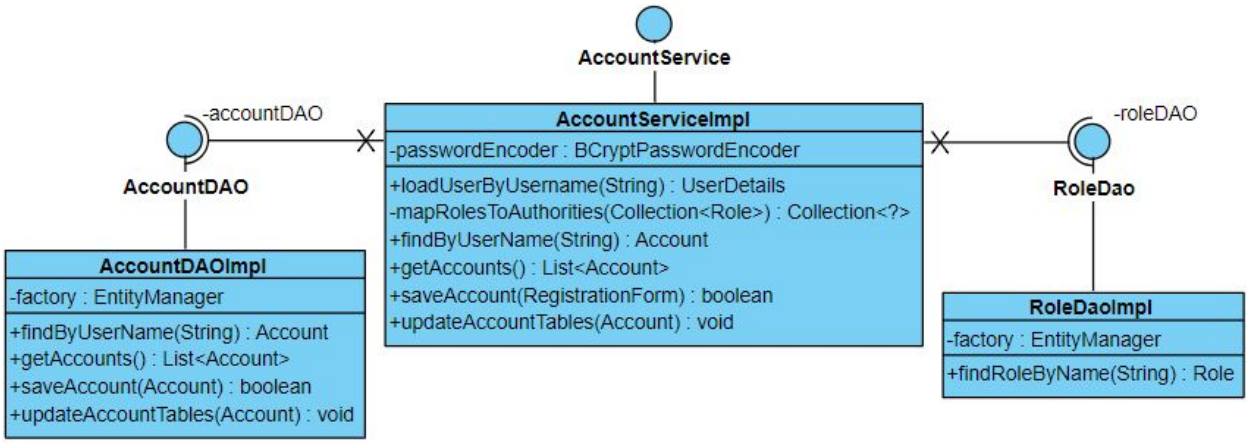


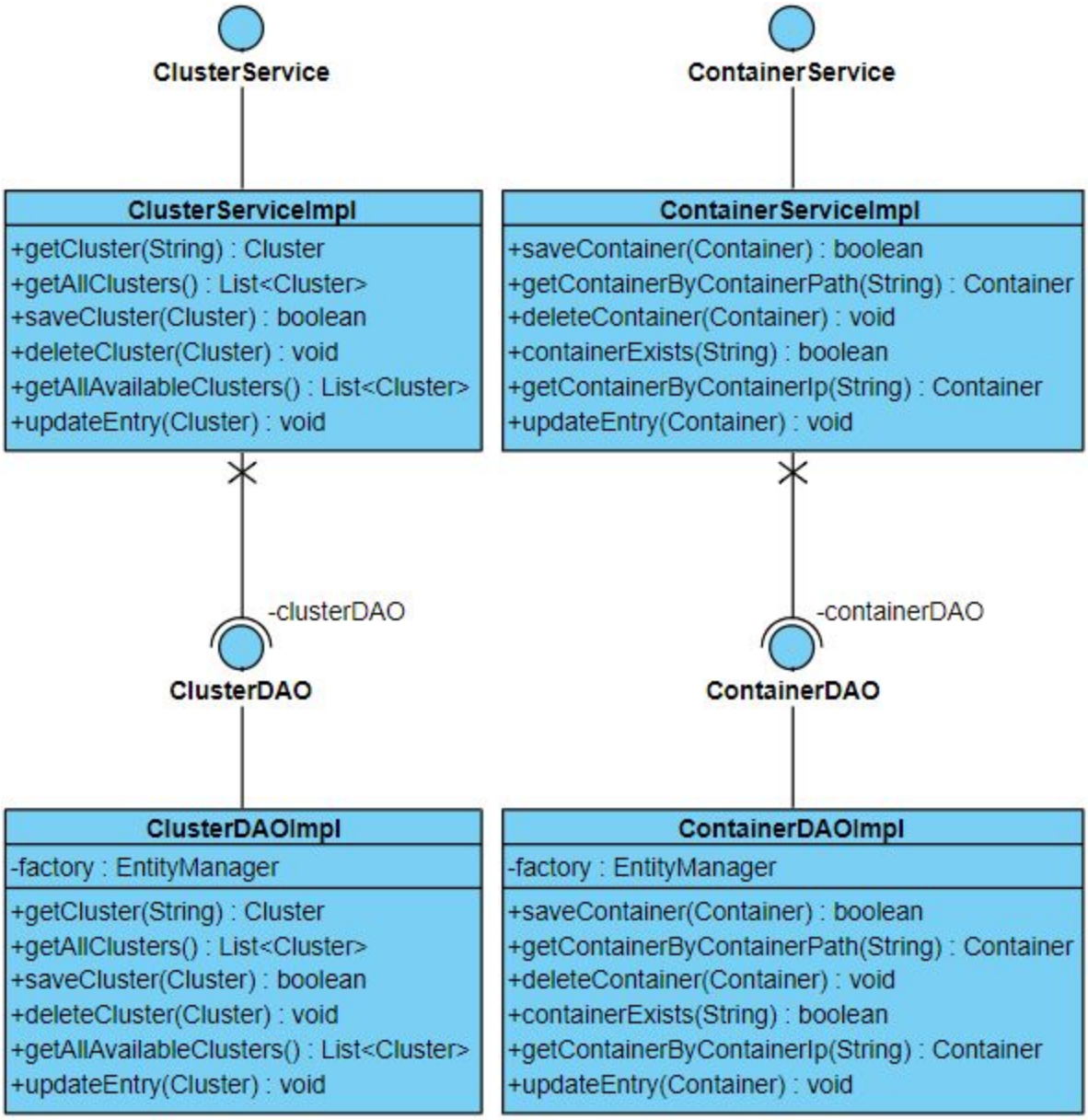
9. System Models

Controllers

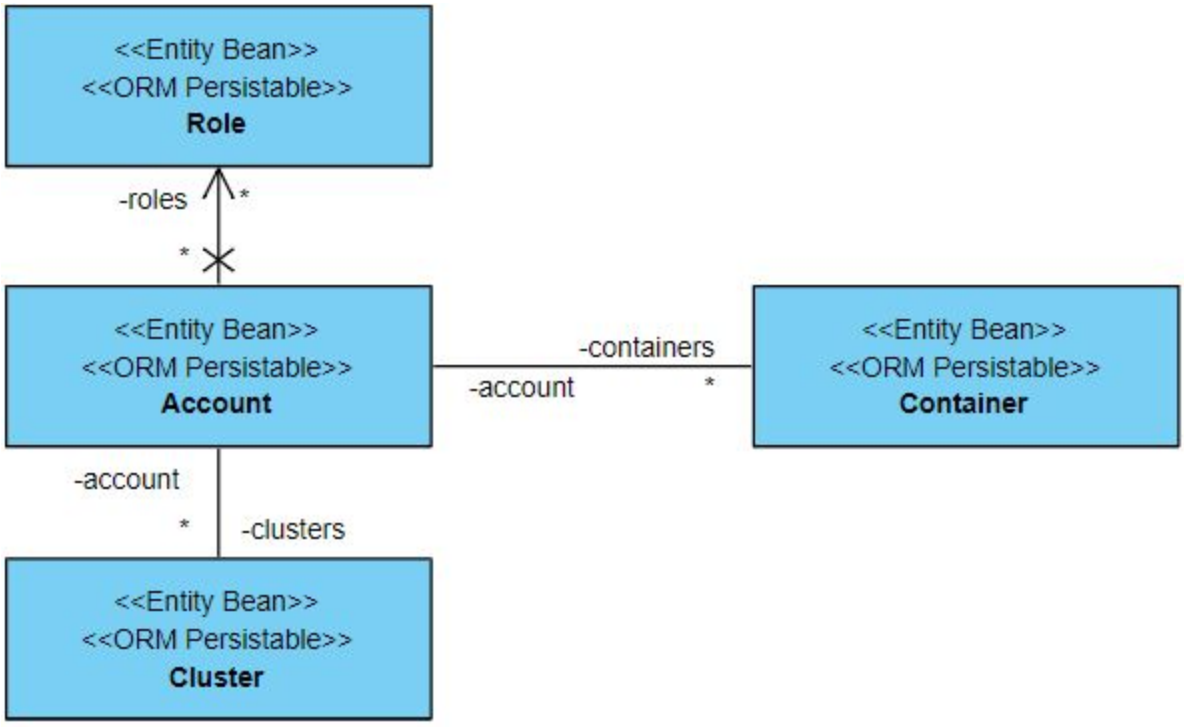


Database Services

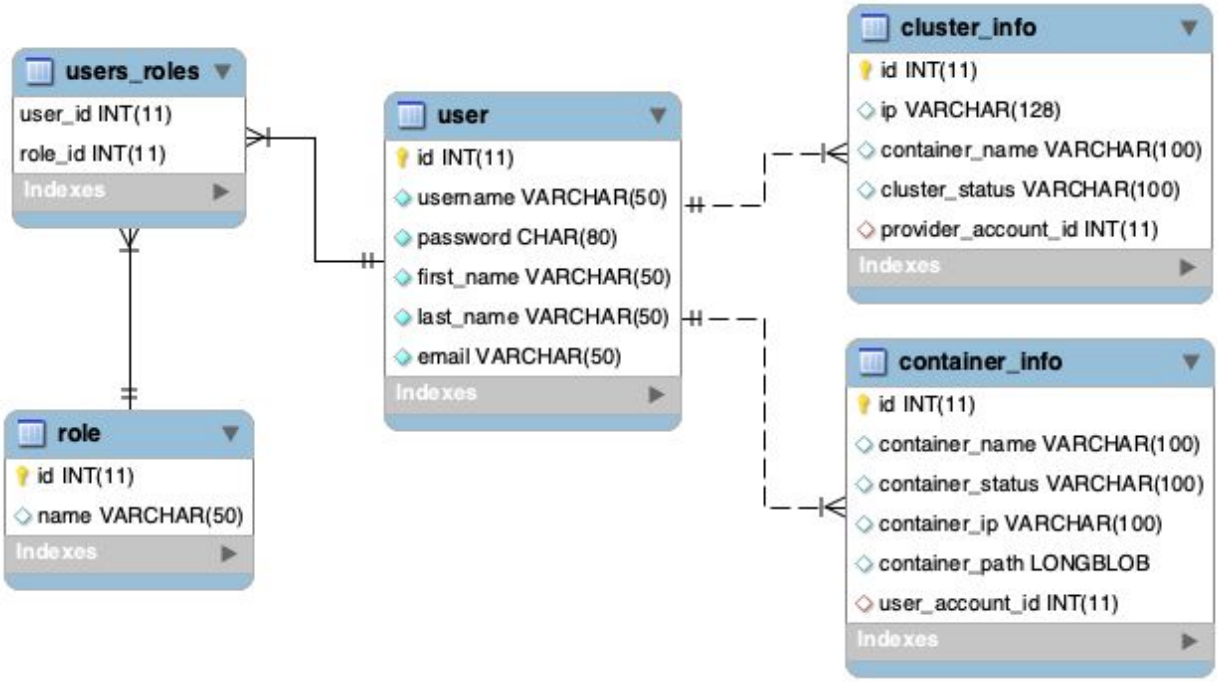



ClusterService
ContainerService
ClusterServiceImpl
+getCluster(String) : Cluster
+getAllClusters() : List<Cluster>
+saveCluster(Cluster) : boolean
+deleteCluster(Cluster) : void
+getAllAvailableClusters() : List<Cluster>
+updateEntry(Cluster) : void
ContainerServiceImpl
+saveContainer(Container) : boolean
+getContainerByContainerPath(String) : Container
+deleteContainer(Container) : void
+containerExists(String) : boolean
+getContainerByContainerIp(String) : Container
+updateEntry(Container) : void
-clusterDAO
-containerDAO
ClusterDAO
ContainerDAO
ClusterDAOImpl
-factory : EntityManager
+getCluster(String) : Cluster
+getAllClusters() : List<Cluster>
+saveCluster(Cluster) : boolean
+deleteCluster(Cluster) : void
+getAllAvailableClusters() : List<Cluster>
+updateEntry(Cluster) : void
ContainerDAOImpl
-factory : EntityManager
+saveContainer(Container) : boolean
+getContainerByContainerPath(String) : Container
+deleteContainer(Container) : void
+containerExists(String) : boolean
+getContainerByContainerIp(String) : Container
+updateEntry(Container) : void

Entities

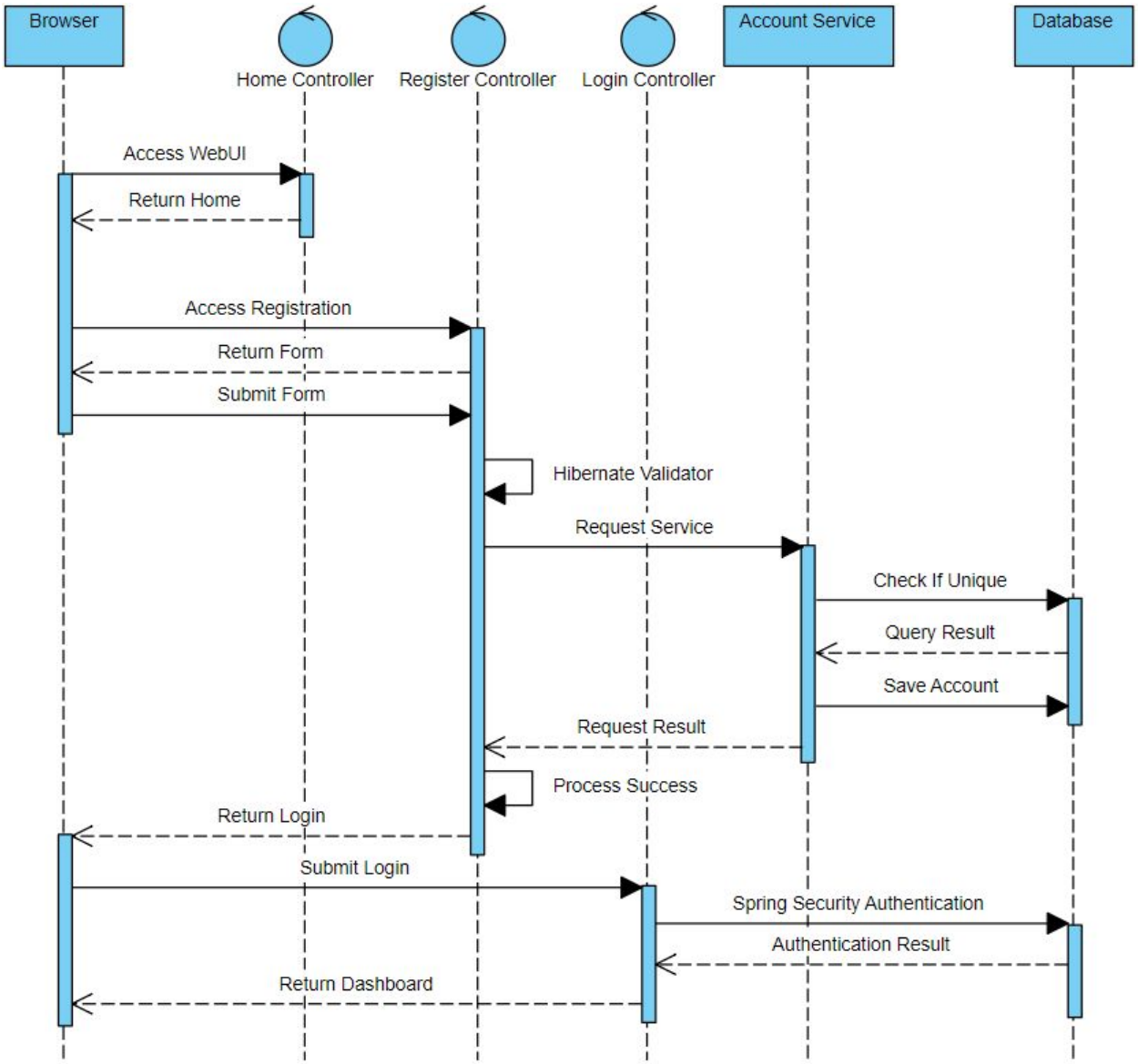


10. Database Model

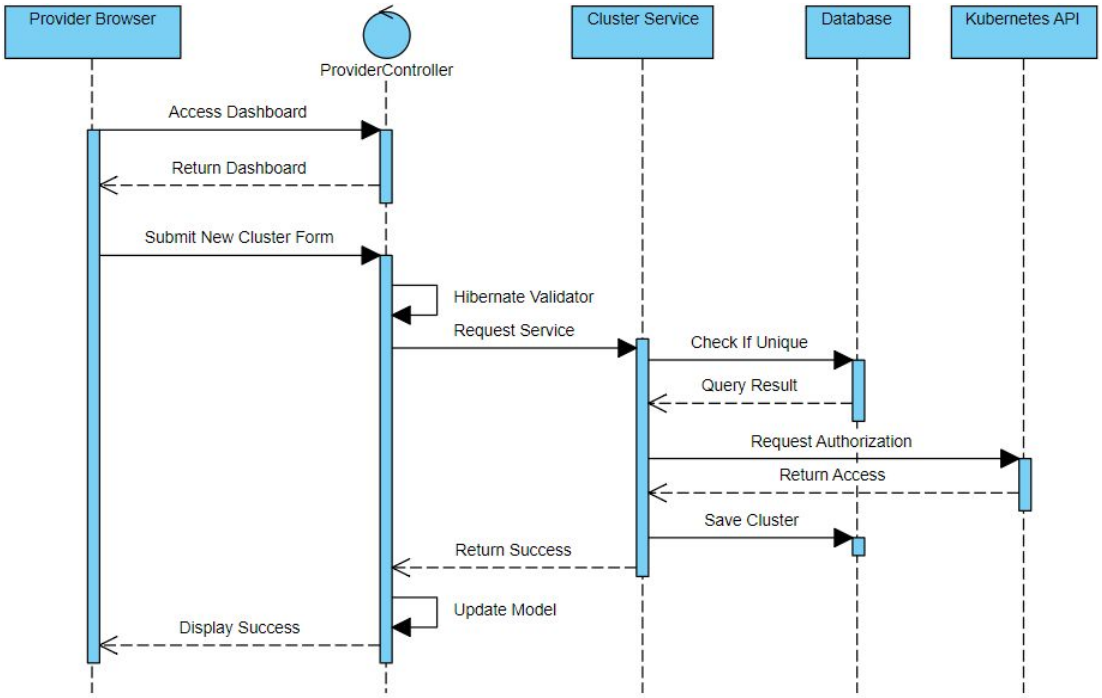


11. Sequence Diagrams

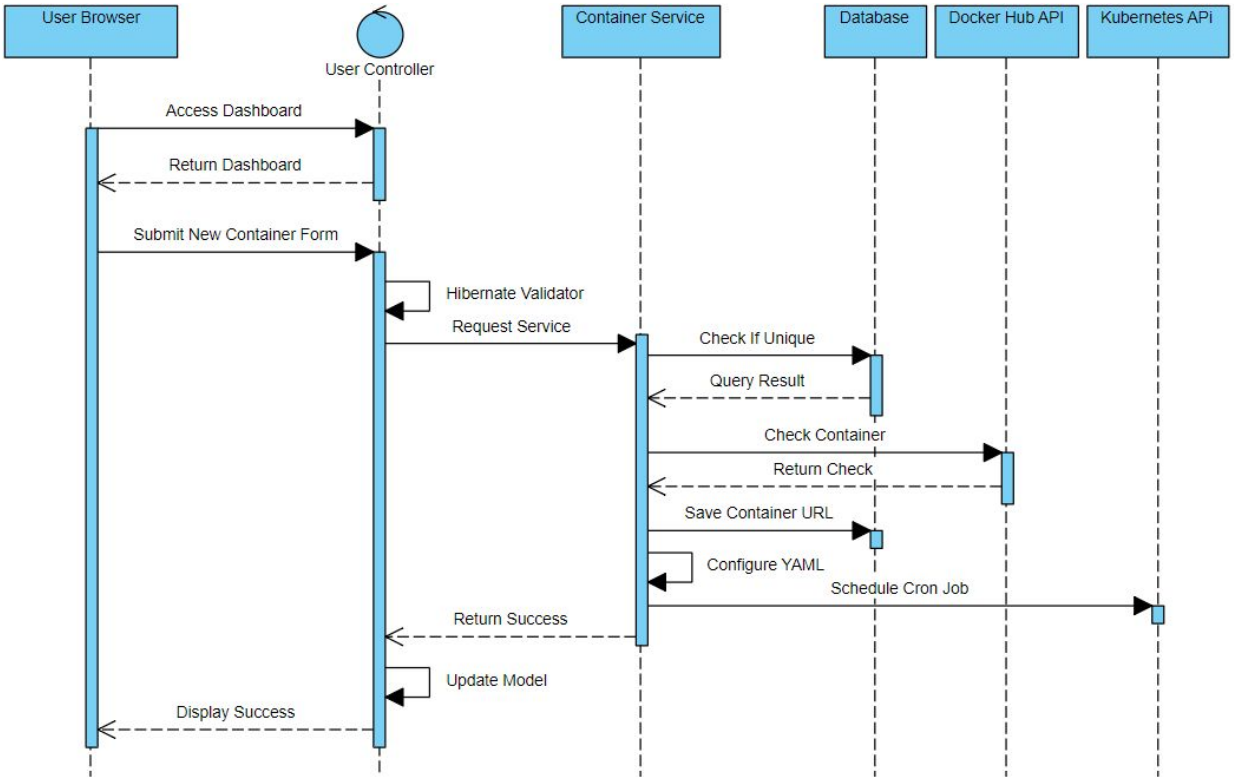
Registration and Login



Upload Cluster



Upload Container



12. Appendices

What we're NOT Doing

We are not going to make this into a mobile app. Connecting users through Kubernetes would be possible through a mobile, but we want to focus on developing a web application first. We are also not creating an interface to create a Kubernetes cluster. As previously mentioned, providers will be responsible for making their own clusters, and then using our web app to connect to other users.

List of Technologies

- **Web Stack**
    - **Java:** Main language used in creating spring based application(s).
    - **Spring Boot:** Configuration for Spring applications.
    - **Spring Core:** IoC and Dependency Injection features.
    - **Spring MVC:** Model-View-Controller (MVC) architecture and components that can be used to develop flexible and loosely coupled web applications.
    - **Spring Security:** Authentication, authorization and other security features for enterprise applications.
    - **Hibernate ORM / Validator:** Object-relational mapper tool and validates user input.
    - **MySQL:** Relational database management system.
- **Cluster Stack**
    - **Minikube:** Local cluster service.
    - **Pivotal Container Service (PKS):** Deploy and run containerized workloads across private and public clouds.
    - **Google Cloud Platform (GCP/GKE):** Series of cloud services including cluster management for running your Docker containers.