1. Background

Kubernetes is a platform that is made to revolutionize the way that container based applications are released and tested. Kubernetes makes use of Docker containers that package and run an application along with all its dependencies in an isolated environment, eliminating errors experienced from running an application on different machines. Unlike a regular virtual machine, Docker containers do not need a guest operating system, making deployment lightweight.

Kubernetes builds on top of this by automating the deployment, scaling, and management of containerized applications within a cluster of computers. Through a master server, a cluster and its nodes can be managed, distributing containers among the nodes.

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.
2. Project Overview

The 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.

3. Project Specifics

The frontend of the Web UI will consist of HTML, CSS, and JAVA elements. The backend will make use of Spring MVC framework. Spring MVC provides many of the tools for front and back end logic, taking a model, modifying data with the controller, and sending it to the view to display it to the user. The Web UI will be hosted on the Pivotal Web Service (PWS) and the database will be managed with MySQL.

The Hibernate framework is used as an object-relational mapper tool for JAVA, but it also provides other APIs. 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 user input has correct structure using annotation driven development.

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.

Google Cloud Platform (GCP) will be used to help with launching and testing Kubernetes for our personal use. The Kubernetes API is used to access the seller’s clusters. We can easily send a commands over to the designated cluster and be able to access it.

4. Assumptions

We are assuming that users have containers that are fully functional and smoothly running. Buyers will be responsible for managing all their own containers and making sure their containers run smoothly so that sellers are able to run them on their clusters. We are also assuming that most users understand how to use Kubernetes or have a basic understanding thereof. Sellers have already set up their clusters before being listed to other users and are ready to accept containers.
5. System Architecture Overview

High Level Diagram

User Interaction and Design

Users of the service must register for either a buyer or seller account. Sellers 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’s IP address and submit for a review. Buyers on the other hand will upload images to our server and manage them through their dashboard.

The general workflow for the buyer will consist of a uploading an image, selecting a cluster from a list of IP addresses, and submitting the request to use the cluster. The seller at this point will accept incoming requests and will automatically deploy the image onto their cluster through the Kubernetes API.

6. User Stories

<table>
<thead>
<tr>
<th>USER STORY</th>
<th>ACCEPTANCE CRITERIA</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a buyer/seller, I can register to the website.</td>
<td>Scenario 1: User has is not registered. Given user has never registered before. And user is not currently logged in.</td>
</tr>
<tr>
<td>Scenario 1: User is registered. Given the user is not logged in And enters a valid username and password When the user clicks submit Then their dashboard is displayed And a successful login message pops up.</td>
<td></td>
</tr>
<tr>
<td>Scenario 2: Invalid username or password. Given the user is not logged in And enters an invalid username or password When the user clicks submit Then an invalid message will appear And will be asked to re-enter their information.</td>
<td></td>
</tr>
<tr>
<td>Scenario 3: User forgets username. Given the user 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.</td>
<td></td>
</tr>
<tr>
<td>Scenario 4: User forgets password. Given the user 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.</td>
<td></td>
</tr>
</tbody>
</table>

As a seller, I can upload a cluster IP to my

<p>| Scenario 1: Seller has not listed their cluster. |</p>
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
</tr>
</thead>
</table>
| **As a buyer, I can upload a container image.** | Scenario 1: Buyer has not uploaded their container image  
Given that the buyer has navigated to the storage menu  
And they have their container image  
When they upload the image  
Then their storage menu will update  
And a confirmation message pops up.  

Scenario 2: Buyer has uploaded container images before.  
Given that the buyer has navigated to the storage menu.  
And that they want to use an old container image.  
When they click on their desired image.  
And they click on submit.  
Then a confirmation message will pop up. |
| **As a buyer, I can schedule a container image so that it can run on a cluster at a different time.** | Buyer has their container image already uploaded  
Given that they found which cluster to work use  
And when they want the cluster to run the image  
When they schedule the cluster in their dashboard  
Then the request is sent to the seller  
And a confirmation/rejection message appears after the seller confirms/denies the request. |
| **As a buyer, I can run a cronJob.** | Seller has a cronJob and an image to run.  
Given that buyer has already uploaded their account so that it can be listed publicly. |
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
</tr>
</thead>
</table>
| As a user, I can remove an image.                                       | User does not need image anymore  
Given that user has already uploaded an image  
And user no longer needs the image  
Then user should be able to delete image by clicking on a 'delete image' 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 bought or sold a cluster. | User has bought/sold 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 buyer, I can communicate (chat) with the cluster owner.             | Scenario 1: chat between two users  
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 users 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 user can start a chat by clicking on the 'start chat' button and add other users  
And users will be connected and they will be able to chat |
| As a buyer, I can see other available clusters that are listed publicly.  | Buyer wants to browse available publicly listed clusters.                                                                                     |
| Given that they have navigated to their dashboard  
And selected the available clusters option  
Then the user will be displayed a list of publicly listed clusters  
And will be given a description with information about each cluster. |
| As a user, I can manage my user settings on a dashboard. |
| 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. |
7. 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, sellers 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 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.