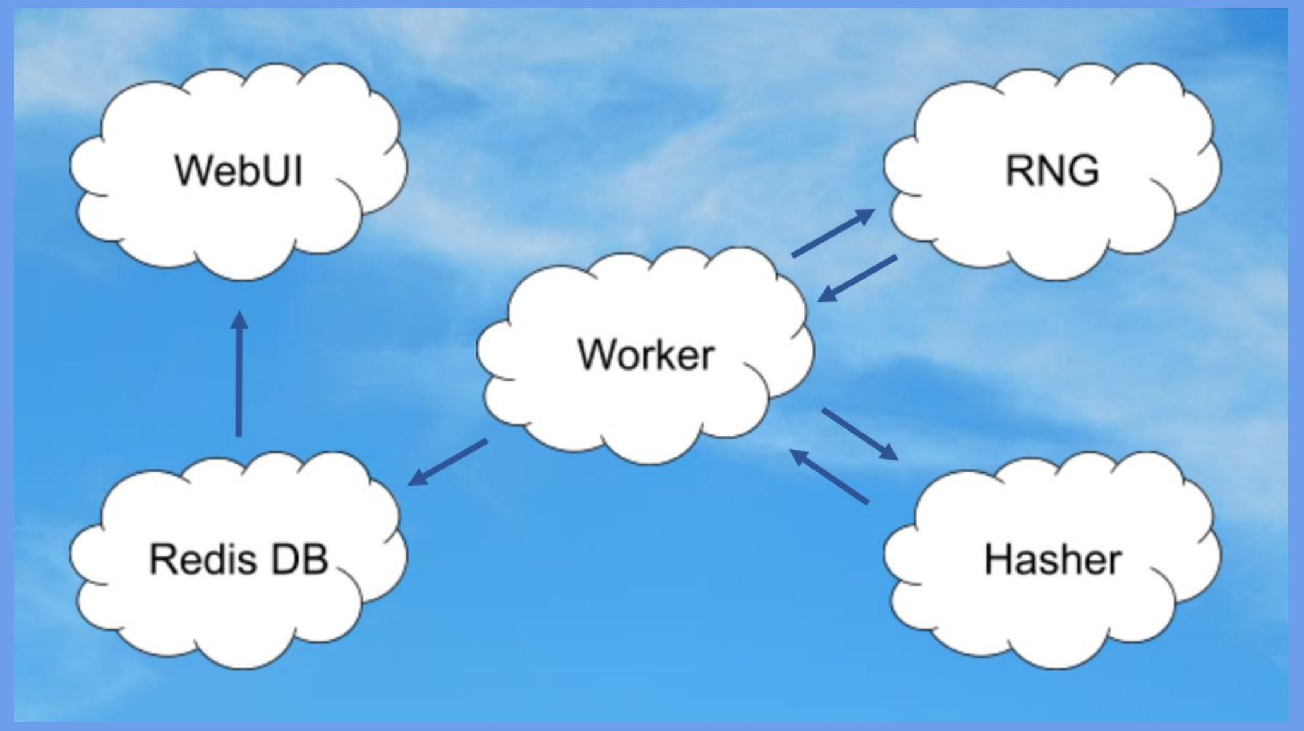
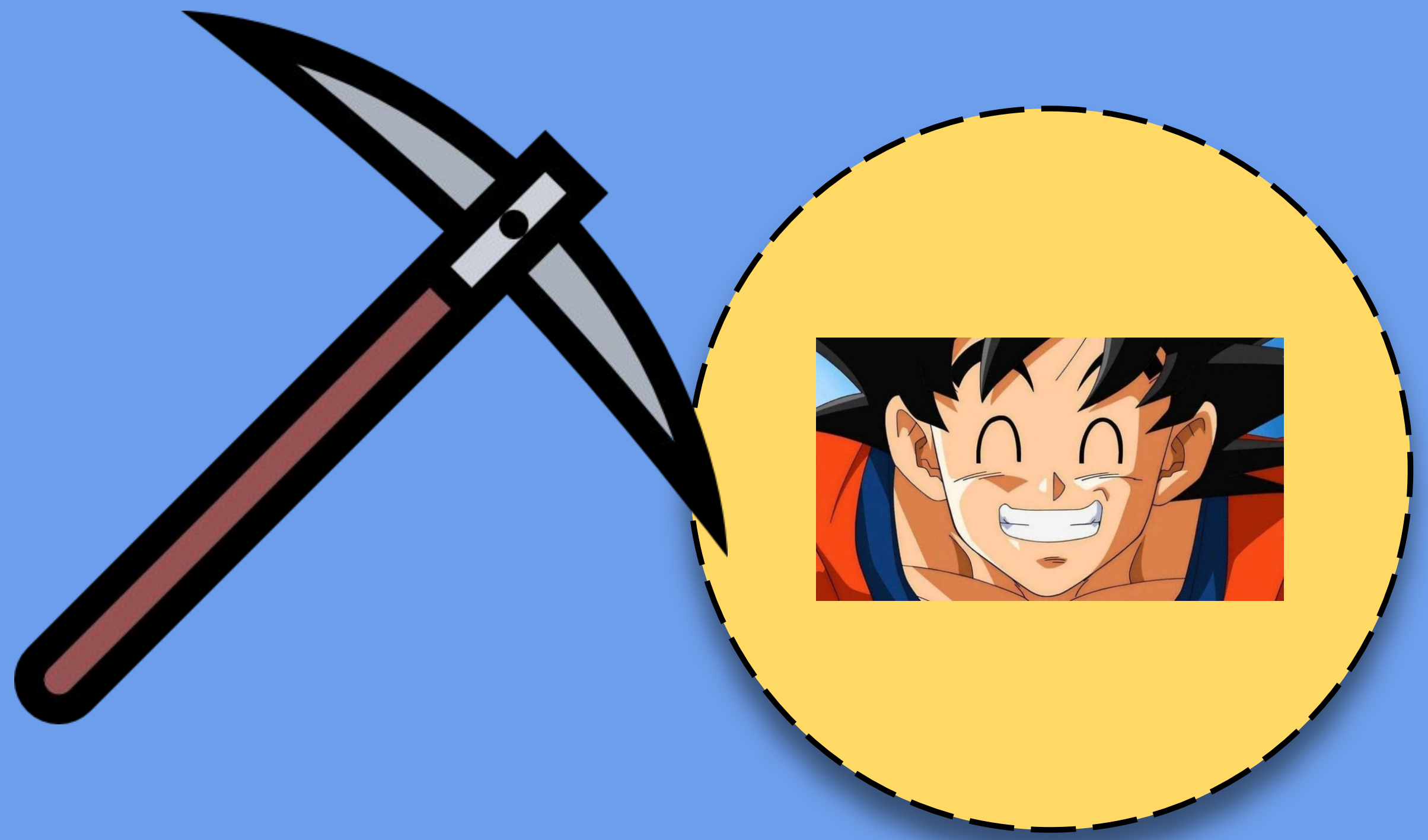


Gokoins Cloud Computing

https://github.com/Every-Villain-Is-Lemons/CSC468-Team-Project

By: Brennan Busza, Bryan Gonzalez-Moyano, Tyler Prehl, Ani Tapia, Matthew Weigand

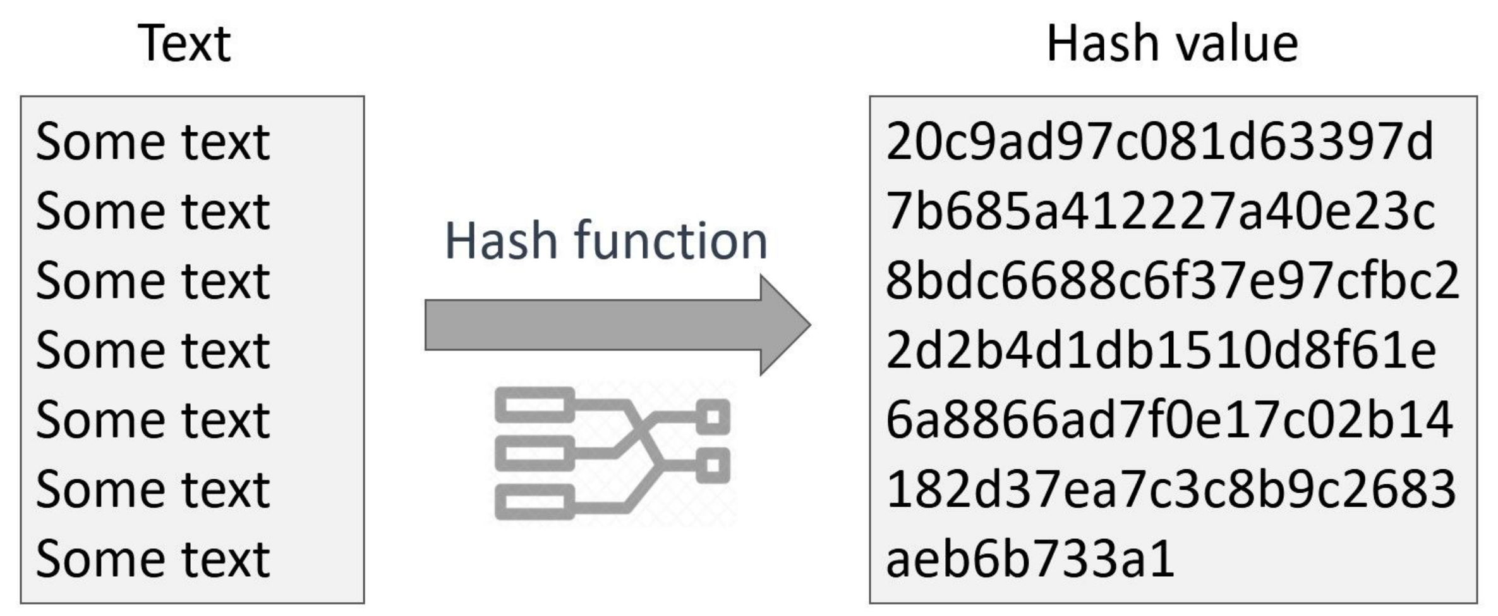


Introduction

The purpose of this project is to help the authors learn more about continuous integration and continuous development (CI/CD) implementation in a cloud environment. The cloud-based project that the authors decided to use as a baseline for what is actually produced is Dr. Linh Ngo's ram_coin project, found at "https://github.com/CSC468-WCU/ram_coin". Through extensive research and training on tools and topics including Docker, Kubernetes, Jenkins, CloudLab, Linux, and GitHub, the authors aim to create a similar project that is fully built and deployed through a Jenkins pipeline that builds from our GitHub repository (link above) when changes are committed to the worker, hasher, rng, and webui branches. Because the authors are essentially creating a simulated cryptocurrency mining experience, the project name is titled after the famous fictional character named Goku to be "Gokoins."

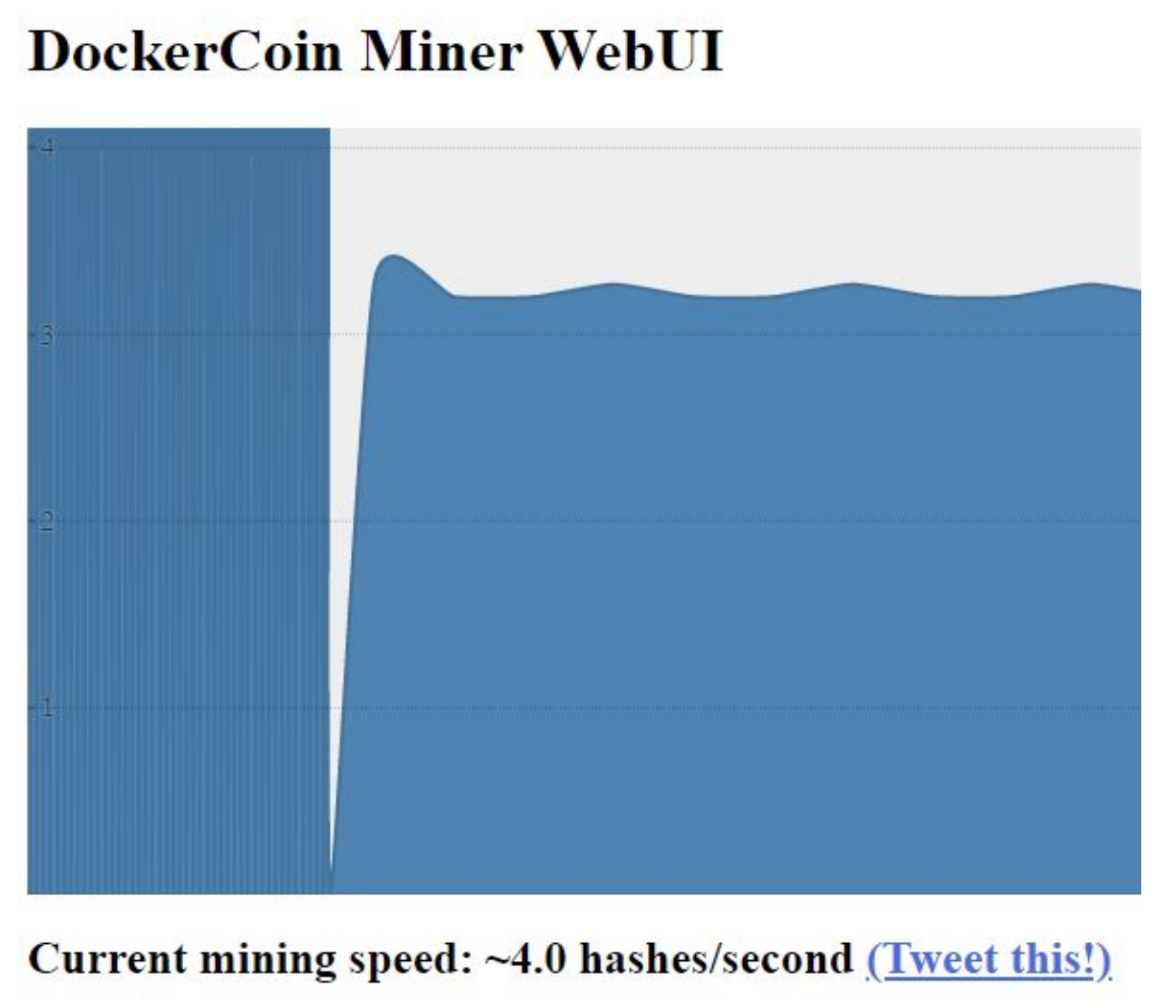
Gokoins Design

The basic premise of Gokoins is to "mine" coins by analyzing hash values. Hash values are simply hexadecimal numbers that are created from text characters and numbers being cycled through various math equations. The diagram below on the left provides an example of a given input text (in this case, repeated "Some text") is turned into a hexadecimal hash value.



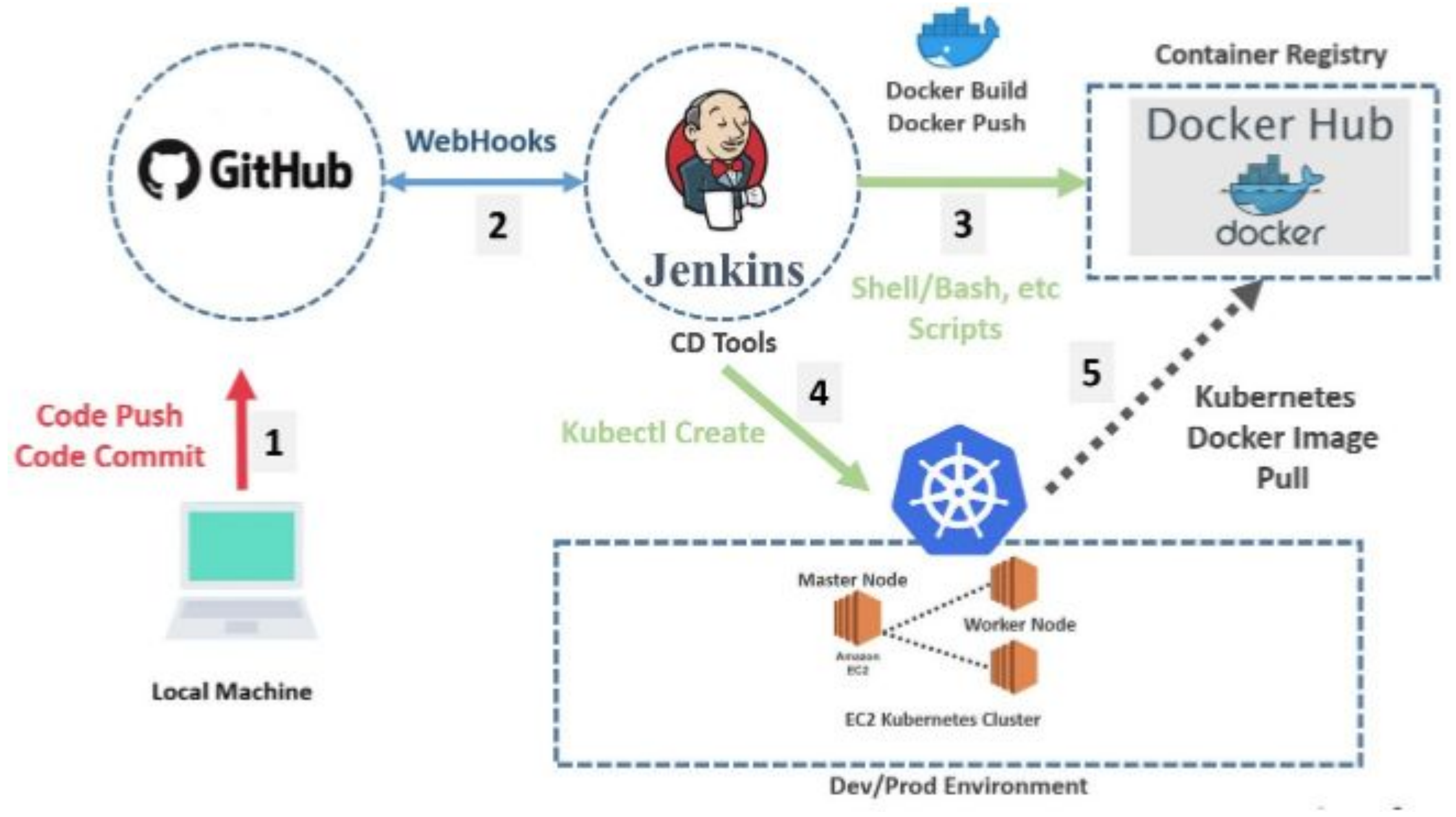
Of course, someone mining cryptocurrency would want to analyze the efficiency of their mining process, so there are five key parts to Gokoins (as seen in the figure in the very top left corner) - the WebUI, Redis database, Worker, Random Number Generator (RNG), and Hasher. The Worker is the centerpiece, requesting and receiving random byte strings from the RNG to be used as the input text for the Hasher to create a hash value from. Once the Worker receives the hash value from Hasher, it decides whether or not it is a valid coin, logs the outcome to the Redis database, and restarts the process to mine more Gokoins. With the outcomes logged to the Redis database, the WebUI pulls that data to display a graph to the user of the rate coins are mined (see the image at the top of the middle column to see the original WebUI by Dr. Linh Ngo for his DockerCoin Miner). Throughout the entirety of the project, we make use of the following programming languages, libraries, tools, and applications: Python, Redis, Jenkins, Kubernetes, Docker(Hub), React, HTML, CSS, CloudLab, Linux, GitHub, YAML, Flask, and more.

Continuous Integration/Continuous Development Design


DockerCoin Miner WebUI

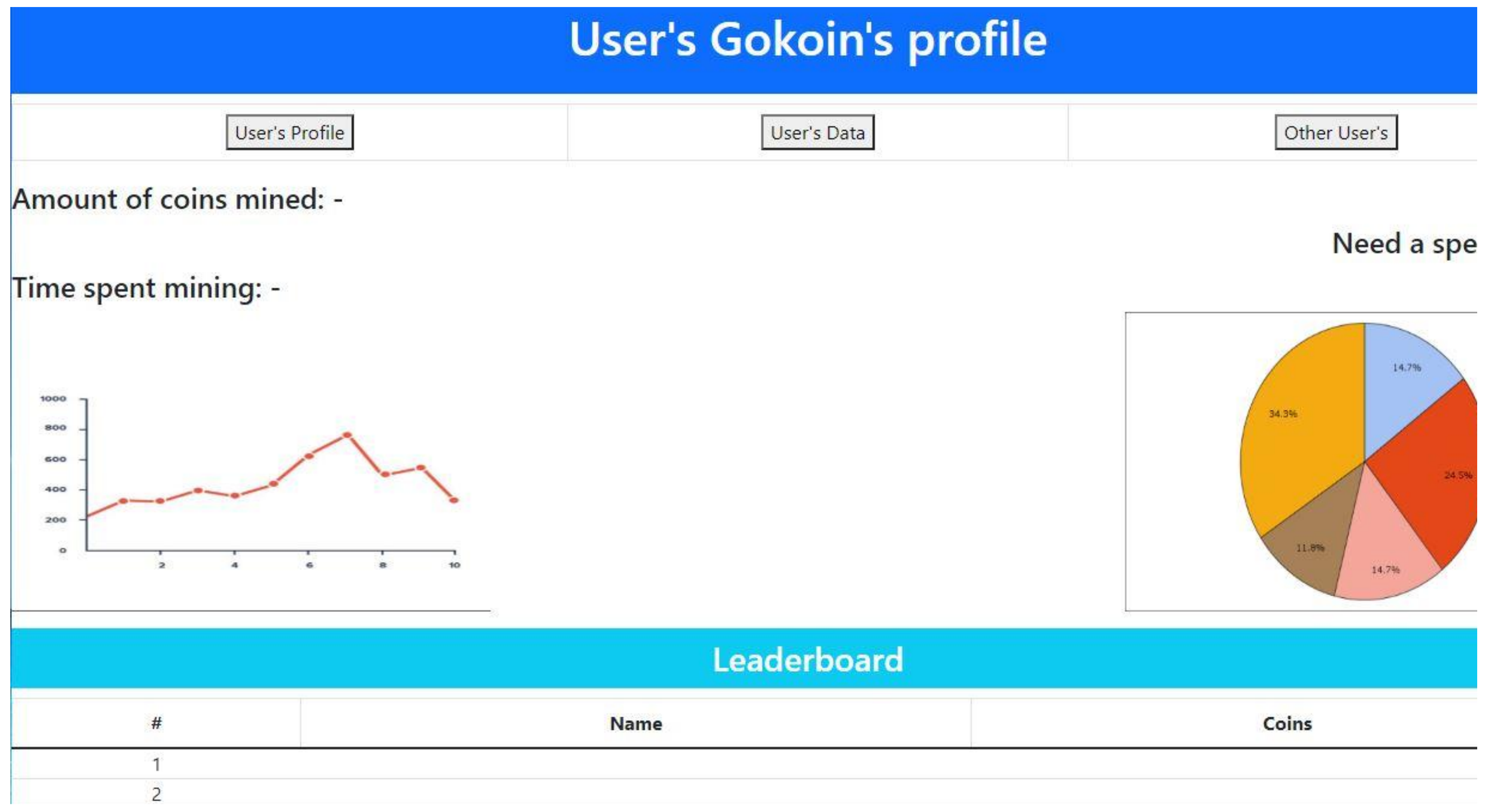
Current mining speed: ~4.0 hashes/second (Tweet this!)

A key learning aspect of this project is learning how to implement and utilize CI/CD services in a cloud environment. The environment we work within is provided by CloudLab, a community cloud service for research and students to use. The final form of Gokoins uses a four-node CloudLab experiment to host one "head" node and three worker nodes. All of our main work is done on the head node, and the three worker nodes act as hosts for various pieces of the Gokoins project. While one worker node may host the deployment of the Docker container for RNG, another worker node may host a Docker container for Worker or Hasher. All of these Docker containers are organized within Kubernetes pods so that if a worker node were to fail, Kubernetes would react by redeploying the dropped containers onto other nodes to resume the process. The final piece of the CI/CD design is the Jenkins aspect, which is the true CI/CD piece as it automatically redeploys any part of the project - the Worker, Hasher, RNG, or WebUI - when any one is changed in the correlating GitHub repository branch. For example, if we updated the Jenkinsfile of the Worker's branch, Jenkins would redeploy the Kubernetes pod and Docker container that host the actual Worker to keep it up to date with code changes. Below, you will find a diagram of how various parts of the CI/CD process interact after a change is made to the code base in GitHub, with our Dev/Prod Environment being our CloudLab experiment. Source:https://ikarimov.com/img/pipeline.jpg



Current Progress

Our current progress is extremely promising - we have currently produced four Jenkins pipelines (for Worker, Hasher, RNG, and WebUI) that use webhooks tied to our GitHub repository to run automatic deployments of code changes and can also build the entire project from scratch. Although we are still currently in the process of updating our WebUI, it is the perfect example to demonstrate how helpful CI/CD is. Previously, any time we wanted to edit the WebUI, we would spend 30 minutes manually redeploying the WebUI piece of the project to our CloudLab experiment. But with the help of our CI/CD pipeline, any changes pushed to our GitHub WebUI branch causes an automatic redeployment process by Jenkins, allowing us to test new versions of our WebUI faster.


User's Gokoin's profile

Amount of coins mined: -

Need a spe

Time spent mining: -

Leaderboard

#	Name	Coins
1		
2		

Challenges

Creating the entire Gokoins project with CI/CD services has been a massive leap of learning for the entire team. When we began the class, none of us knew what Kubernetes was, let alone how to create an automated Kubernetes deployment via Jenkins. Due to our serious lack of background knowledge, there was a steep learning curve from our early phases of understanding the purpose of Docker containers to producing an end-to-end CI/CD pipeline. Additionally, when first conceiving our initial project idea, we had lofty goals for Gokoins, such as the WebUI above where a sign-in page would bring the user to a personalized Gokoins mining page, providing data about time spent mining, mining speed boosts, and mining leaderboards. Other goals we had in mind included using a MySQL database instead of Redis, but to keep our lives simpler while learning the ins-and-outs of crafting CI/CD pipelines, we decided to make simpler (but still challenging) changes to the original project by Dr. Ngo by producing a nicer WebUI and changing the Hasher from using the Ruby programming language to Python.