SE 575

Final Project

https://github.com/jzlotek/bc

John Zlotek — Kevin Wu — Jainil Patel

December 3, 2020

1 Features

- Proof of work algorithm (SHA 256)
- Backend concurrent worker routine
- Separated front and back end
- Reactive front end
- Deployed via Kubernetes using Docker images

2 Deployment Pipeline

We are using GitHub Actions for our entire deployment lifecycle. We are used to using GitLab CI/CD for building, testing, and deploying so some of this had to be learned. It has some features like dependent pipeline runners (which was not used here at all) which is cool. The pipeline we built runs both Docker images in parallel then pushes them to https://hub.docker.com/u/jzlotek so they can be pulled by the next step in the pipeline which is the Kube deployment step via Helm. Helm charts are essentially packages for Kube. We created our own microservice helm chart for Kube. The values for each service are found in the *helm/values.yaml* in each directory. The Helm deploy stage takes the two built image tags and deploys them with the respected Helm chart to our Kubernetes instance.

3 Using Kubernetes

In terms of scalability and high availability, we are utilizing Kubernetes to spin off our backend processes. Because our front end is a static site, it is only sitting at 2 instances where the backend workers are using 3 to 4 workers. We are using traefik as our ingress controller to route on the subdomain https://bc.dulcimer.live. Traefik is also able to differentiate between the frontend and the backend as the backend has the path prefix /go for all of the requests. From testing with Kubernetes and our memory and cpu limits, when you spin up too many workers, it will crash that instance of the worker request pool. Kubernetes is able to keep the other 2 instances up when the one is deemed OOM and is killed. Thanks to Gin, the startup time for the new instance is almost instant.