Intro:

Through this exercise, we'd like to get a sense of how you approach infrastructure tasks, building CI/CD pipelines, and how you write code.

If you wish to, you have the option to start coding a solution on your own beforehand **but please do not spend more than 2 hours on it**. The live pairing will continue from wherever you have gotten up to.

Some tips for undertaking this exercise:

1. Please read the guidelines carefully and keep it simple. The test is designed so that each requirement builds on top of the previous one.
2. Whatever steps you aren't able to cover in the 2 hours we will talk through in a follow up so that you have a chance to explain your process and reasoning for extending your solution.

Task - Create a CI pipeline in code that deploys a single service.

Goals:

1. Create a service, "service-a". Write a script/application that retrieves the bitcoin value in dollars from an API on the web (any API of your choice) every minute and displays it. You can choose the language you are most comfortable with. Popular options are Node.js, Python, and Ruby. There needs to be a dockerfile that can be built but we're not expecting it to be published.
2. "service-a" has a helm chart or other declarative template manifest that is validated and deployable through a CI/CD pipeline. We are expecting the service to expose itself in order to serve traffic outside of the cluster.
3. Create a CI/CD pipeline that could deploy "service-a", defined as code.. The tools you choose (Jenkins, Travis, CircleCI) are completely up to you.
4. Create a README to briefly capture your rationale for the solution. For any requirements that aren't completed within the recommended 2 hours time limit, describe in the README how you would go about approaching and implementing those requirements.

Guidelines:

1. The CI pipeline should be automated and defined as code.

2. We are not expecting a live example, but we are expecting to see code and configuration for the solution.
3. Share any templates and config files you used as well as the code prior to the Interview.
4. **Keep it simple**. There are many tools out there that can help you accomplish this.
5. Monitoring, Security, Testing, Source Control, and CSS **are all great topics to discuss later** - don't spend time on them now. You can add comments if you want, explaining what you'd do if you had more time.