Intro:

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

If you wish to, you have the option to start coding a solution on your own before hand. The live pairing will continue from whereever you have gotten up to.

If you find yourself spending a lot of time on this:

1. Please re-read the guidelines 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 Kubernetes Cluster using code.

Goals:

1. Have an EKS/GKE Kubernetes cluster using AWS/GCP cloud (free tier).
2. Have 2 services: "service-a" and "service-b".
3. For "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.
4. "service-b" should redirect requests from **service-b/rate** to **service-a/rate** and simply return "*nothing here*" for anything else (/*).

Bonus section: (only if you can spare the time)

5. Have an Ingress controller, redirecting traffic into **service-b** and only allowing the **/rate** route. No need for a fancy DNS address or SSL.
6. Make sure service B has a basic health check for the Ingress to work properly.
7. Have RBAC enabled. And don't allow **service-a** to "talk" with **service-b** internally (policy disabled).

Guidelines:

1. The cluster buildout should be automated and fully repeatable.
2. Share any templates and config files you used as well as the code prior to the Interview.
3. **Keep it simple**. There are many tools out there that can help you accomplish this.

4. If you're creating new docker images, storing them publicly on dockerhub might be simpler than authenticating with a private repo. **It's also perfectly fine to run it all locally and just explain what the diff would be to get it up in the cloud.**
5. Monitoring, Security, Testing, Source Control, CI/CD, Automation 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.