1. As our product contains a Chrome Extension and Cloud services on AWS, we had to use two different packaging services. For the Chrome Extension side, we decided to use the Google Chrome Store to package our product. This makes it easy for users to install our extension without hassle, and it is easy for us to upload and package our extension. For our product to be integrated in the Google Chrome Store, we had to have a manifest file that provides metadata about our product. This file is included with the build folder that contains the code for our extension. We then upload our code to the extension developer dashboard where it will be reviewed. Once accepted, the extension will be available on the Google Chrome Store. The manifest file dictates what will appear on the Google Chrome Store and the configurations for our extension. When changes are made, we simply reupload our new build folder and the review process is repeated. As for the resources on AWS, we used terraform (an Infrastructure-as-Code program) to deploy, configure, and maintain them. Our repository contains a folder that has all of the terraform files that specify the resources, policies, and connections that are required for our product on AWS. Installation of the terraform CLI is required to interact with these files and is used to initialize the infrastructure, update it, and manage the resources. The folder is included in our GitHub repository for our project.
2. Given that our product is split up into two different parts, there will be two different release versions. We decided to use semantic versioning as it is easily understandable and used by much of the industry. The first part is the frontend Chrome Extension which will have version 0.1.0 and is a product. As it is still under development, the major version will be 0 to indicate it is not a completed product. The minor version is 1 to indicate that it is the first iteration of the product, and 0 for the update version as there have been no new updates since the release of this version. In Google Chrome Extensions, the version numbers are in the manifest file which contain the metadata on the extension. When added to the store, these version numbers are shown on the extension information page to users. We plan to increment the minor version when making changes, and once we have a finalized product, the major version would be incremented to one to indicate a final working product. For cloud packaging, the same approach to versioning will be taken. With the current version being 0.1.0 and the type would be a Infrastructure-as-Code package. It is at 0.1.0 for the same reason as the extension which is currently under development, but it is the first iteration with no current patches or updates. As it contains several files with Infrastructure-as-Code, we decided that a package would be the best type, as it is not a product on its own. As we make changes, the minor version will increase or if there are slight updates, then the patch version will increase. Once there is a finalized cloud infrastructure, then the major version will be incremented. The version will be saved as a variable within the terraform file so users can simply look at the version listed with the variables.
3. To upload our Google Chrome Extension to the Chrome Store, it requires approval. Our extension has not yet been approved to be uploaded to the Chrome Store. As such, we are unable to provide a link to the artifact on our desired hosting platform for our product. Below is a link to our CI build as requested. As for our cloud artifact, it would contain the private keys and secret credentials of our accounts and infrastructure. As such, we have decided to zip up all the necessary files needed to deploy our cloud infrastructure. A

user can simply unzip the folder and run the terraform apply commands to launch our code into the cloud. The zip file is found in our GitHub repository.
https://github.com/Ssharma1230/HashPass/actions/runs/14135552720