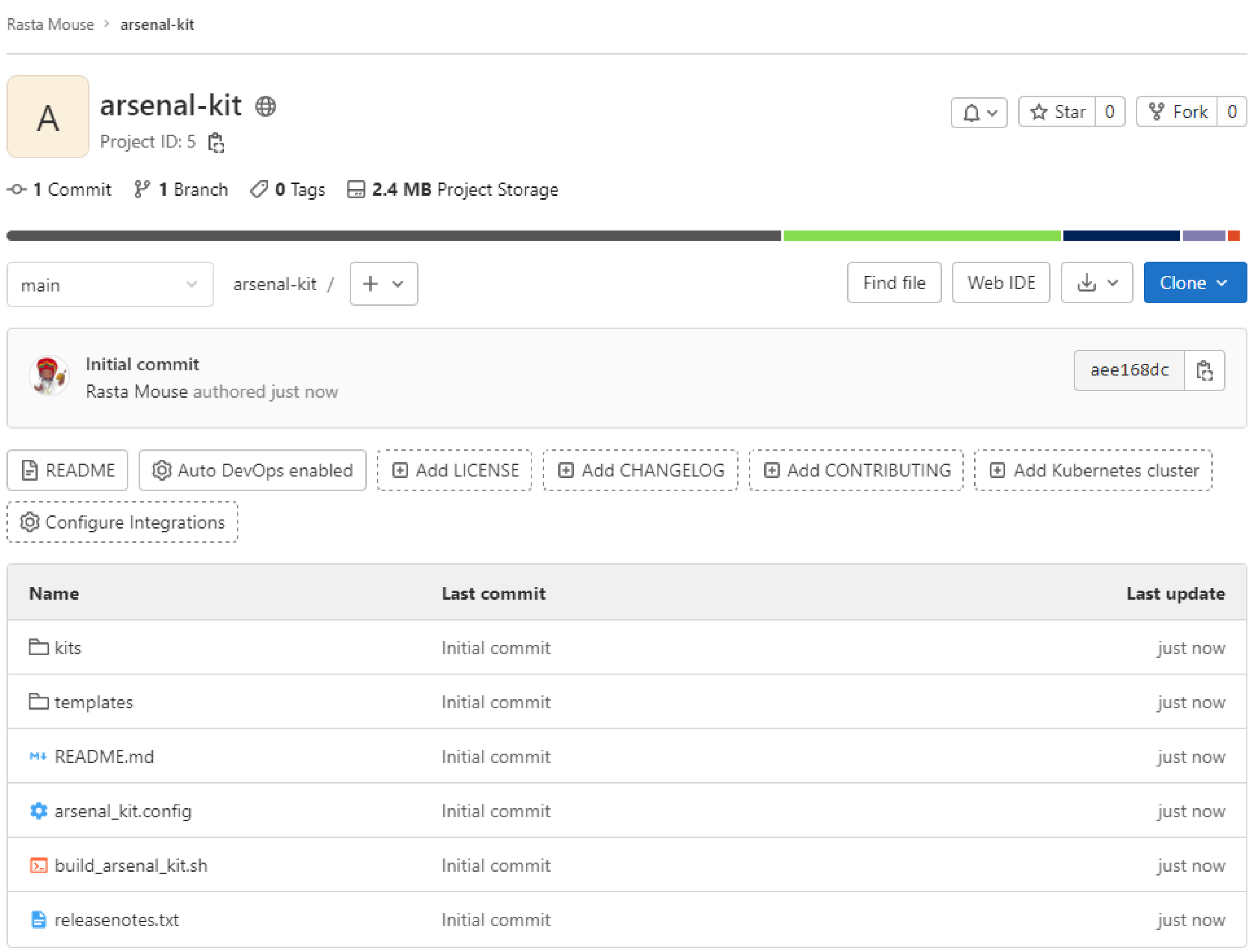
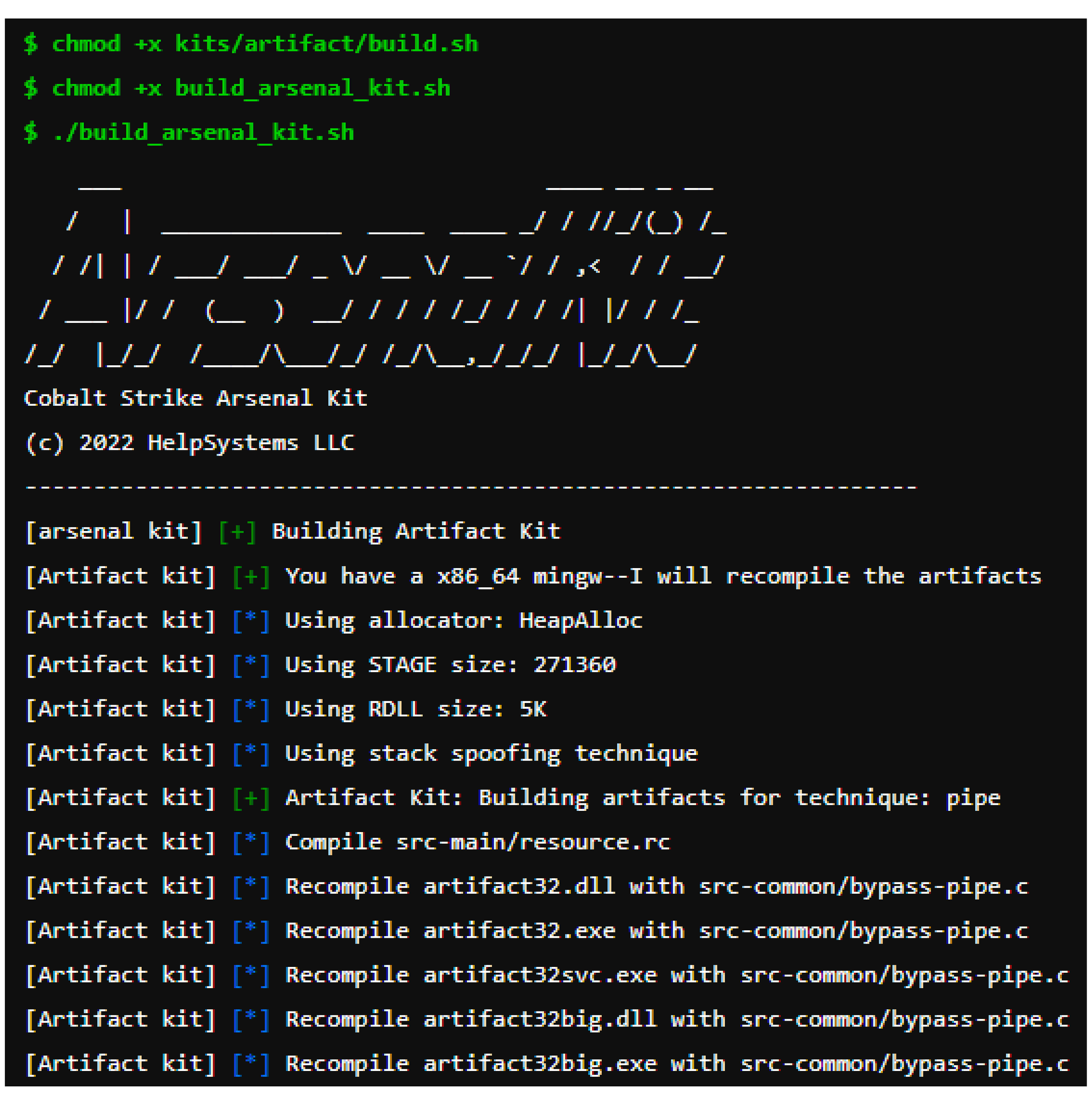
We can also publish built artifacts from GitLab somewhere that's easy for users to consume. For this, I'm using Cobalt Strike's Arsenal Kit as an example. For instance, a team may want to source control customisations to their kit(s) and publish complete builds for its members to pick up and use from a central location.



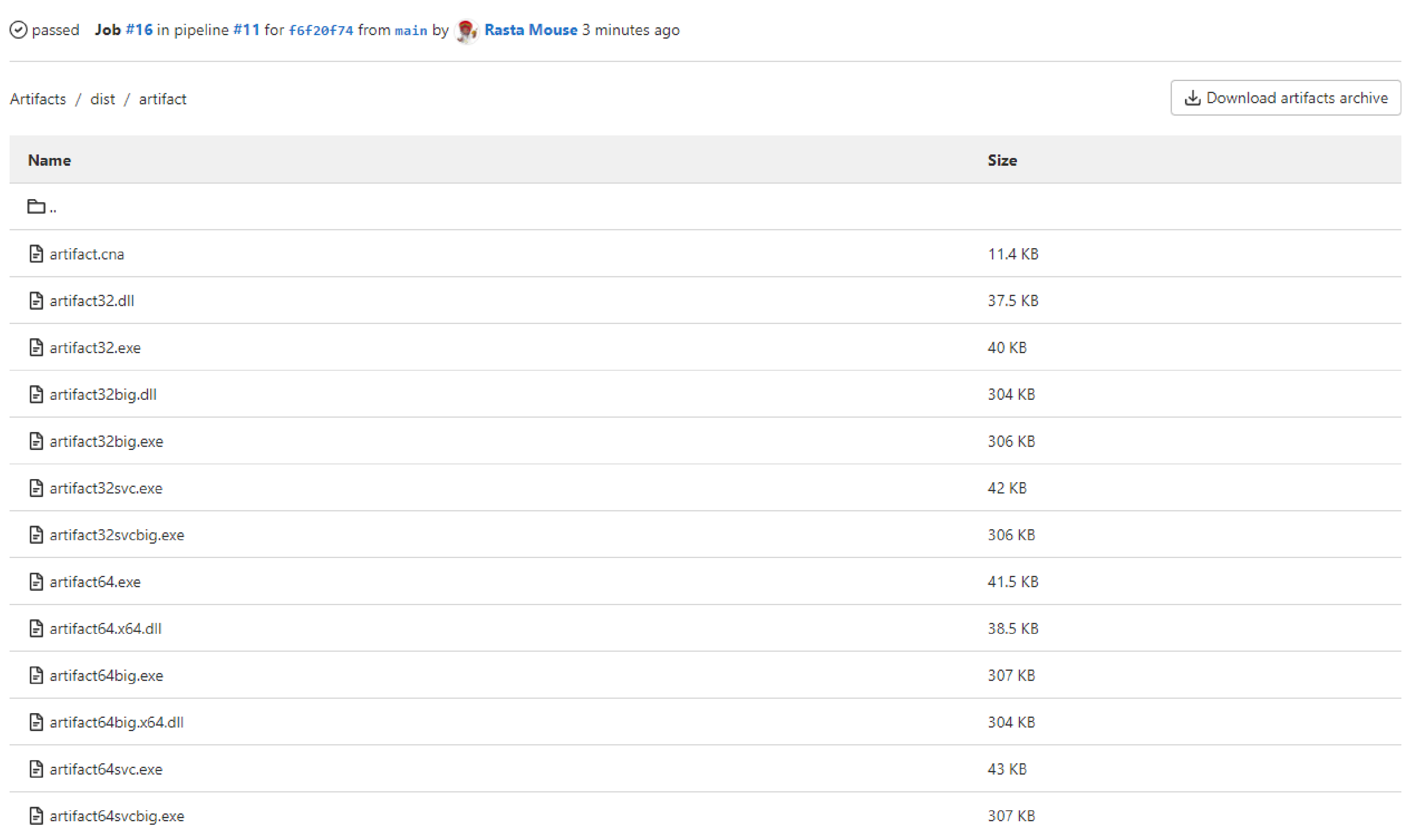
This can be built on a standard Ubuntu Docker image using the following:

```
image: ubuntu

stages:
  - build

build:
  stage: build
  variables:
    output_path: "dist/artifact"
  script:
    - "apt -y update"
    - "apt -y install mingw-w64 cifs-utils"
    - "chmod +x kits/artifact/build.sh"
    - "chmod +x build_arsenal_kit.sh"
    - "./build_arsenal_kit.sh"
  artifacts:
    paths:
      - "$output_path/"
```



This will output everything to a new directory called `dist`.

However, specifying the output path under `artifacts` will tell GitLab to pull those out of the job and store them in the UI. This is also mandatory to ensure that the built artifacts are passed to any future jobs (such as deploy) without being deleted.



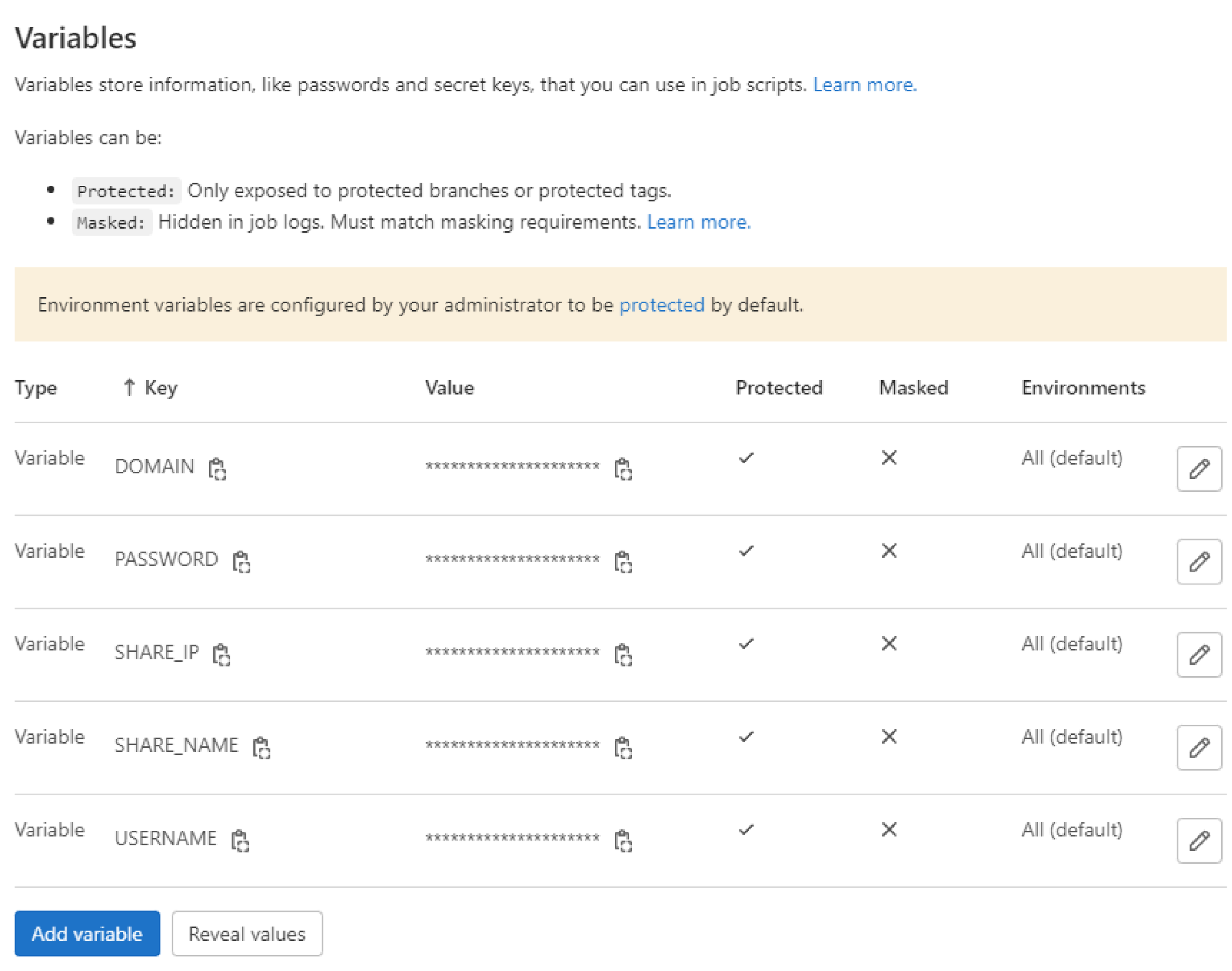
I shall now deploy these to an SMB file server, using the following addition to `.gitlab-ci.yml`:

```
deploy:
  stage: deploy
  script:
    - "apt -y update"
    - "apt -y install cifs-utils"
    - "mkdir /mnt/share"
    - "mount -t cifs //$SHARE_IP/$SHARE_NAME /mnt/share -o rw,domain=$DOMAIN,username=$USERNAME,password=$PASSWORD"
    - "cp -r dist/* /mnt/share/artifact-kit"
```

You cannot usually mount an SMB share from inside a Docker container. That's why we had to set the privileged setting to true in the runners.docker configuration.

There are several variables defined here, including the credentials to authenticate to the share. These should not be hardcoded in your YAML file, but rather defined inside the GitLab project. With the project, go to **Settings > CI/CD > Variables** and click the **Expand** button.

Here, you can click **Add variable** and provide any key/value pair that you need. These will then be passed down into the Docker runner as environment variables.



After running the pipeline, all the files are present.

```
PS C:\> ls -r \\172.23.156.140\artifacts$\

    Directory: \\172.23.156.140\artifacts$

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        17/11/2022     10:24                artifact-kit

    Directory: \\172.23.156.140\artifacts$\artifact-kit

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        17/11/2022     10:24          11624 artifact.cna
-a----        17/11/2022     10:24          38400 artifact32.dll
-a----        17/11/2022     10:24          40960 artifact32.exe
-a----        17/11/2022     10:24         311296 artifact32big.dll
-a----        17/11/2022     10:24         312832 artifact32big.exe
-a----        17/11/2022     10:24          43008 artifact32svc.exe
-a----        17/11/2022     10:24         313344 artifact32svcbig.exe
-a----        17/11/2022     10:24          42496 artifact64.exe
-a----        17/11/2022     10:24          39424 artifact64.x64.dll
-a----        17/11/2022     10:24         313856 artifact64big.exe
-a----        17/11/2022     10:24         311296 artifact64big.x64.dll
-a----        17/11/2022     10:24          44032 artifact64svc.exe
-a----        17/11/2022     10:24         314368 artifact64svcbig.exe
```

You're obviously not constrained to SMB - you can transfer files over whatever means works best for you.