1. **Section 1: Selling CI/CD to your Team/Organization**

* Reduce risk

Finding and fixing bugs late in the development process is expensive and time-consuming. This is especially true when there are issues with features that have already been released to production.

With a CI/CD pipeline, you can test and deploy code more frequently, giving testers the ability to detect issues as soon as they occur and to fix them immediately. You are essentially mitigating risks in real time.

* Deliver faster

Organizations are moving toward releasing features multiple times a day. This is not an easy task; only a handful of companies like Netflix, Amazon, and Facebook have been able to achieve this goal. But, with a seamless CI/CD pipeline, multiple daily releases can be made a reality.

Teams can build, test and deploy features automatically with almost no manual intervention. This is accomplished using various tools, frameworks, and systems like Travis CI, Docker, Kubernetes, and LaunchDarkly.

* Expend less manual effort

To align with the shift-left paradigm, we need automation right from the start. This is also a vital component of having a successful CI/CD implementation. Once you build features and check in code, tests should be automatically triggered to make sure that the new code does not break existing features and that the new features are working correctly.

After the tests run, the code gets deployed to different environments, including QA, staging and production. Throughout this process, you will be getting constant notifications through different channels, giving you plenty of information about the build, test and deploy cycles.

* Generate extensive logs

Observability is one of the biggest aspects of DevOps and CI/CD integration. If something is wrong, you need to understand why. You need a mechanism to study the system in production over time and identify key performance metrics. Observability is a technical solution that helps in this effort.

One key aspect of observability is logging information. Logs are a rich source of information to understand what is happening beneath the UI and study application behavior.

With a CI/CD pipeline, extensive logging information is generated in each stage of the development process. There are various tools available to analyze these logs effectively and get immediate feedback about the system.

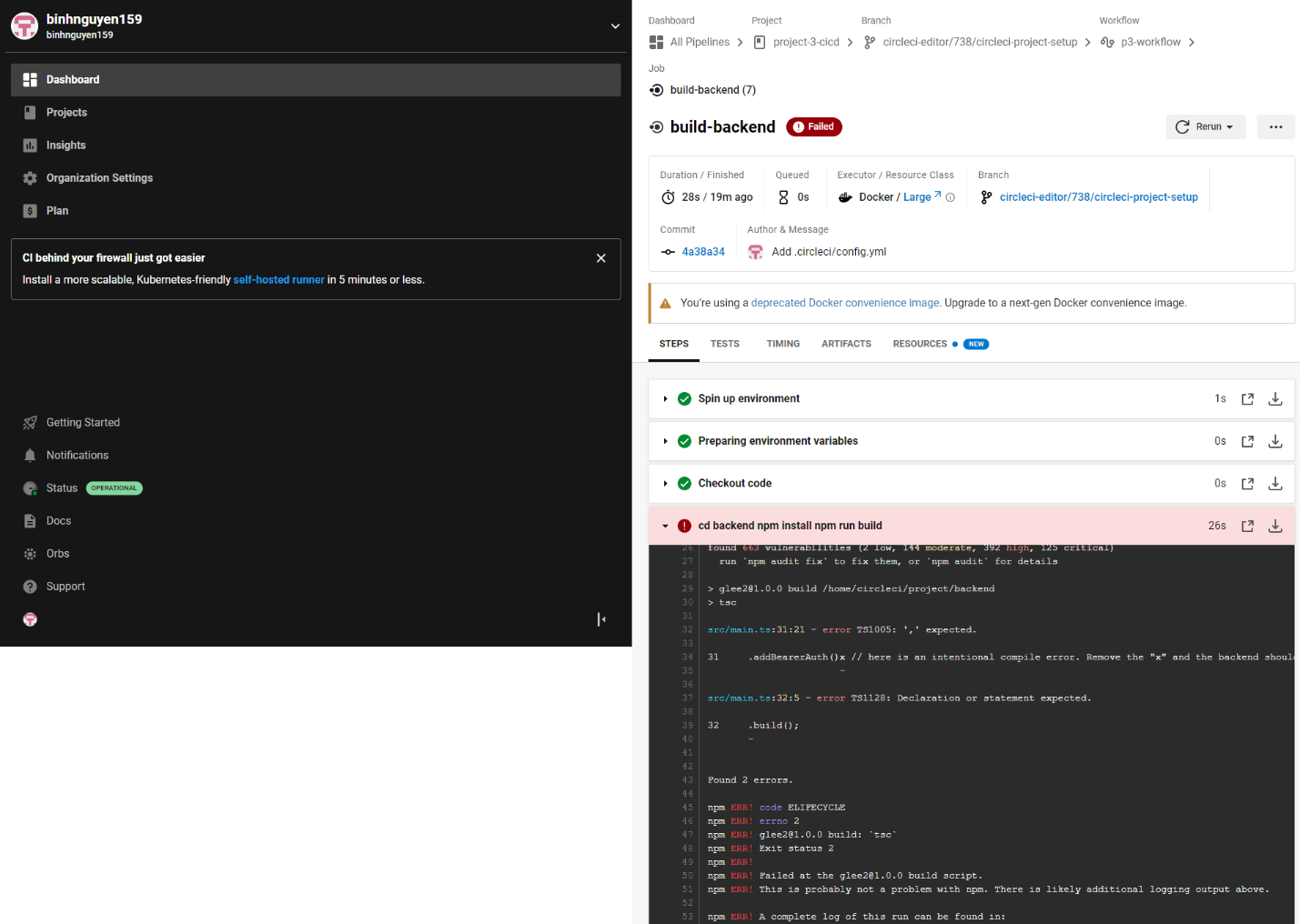
* Make easier rollbacks

One of the biggest advantages of a CI/CD pipeline is you can roll back changes quickly. If any new code changes break the production application, you can immediately return the application to its previous state. Usually, the last successful build gets immediately deployed to prevent production outages.

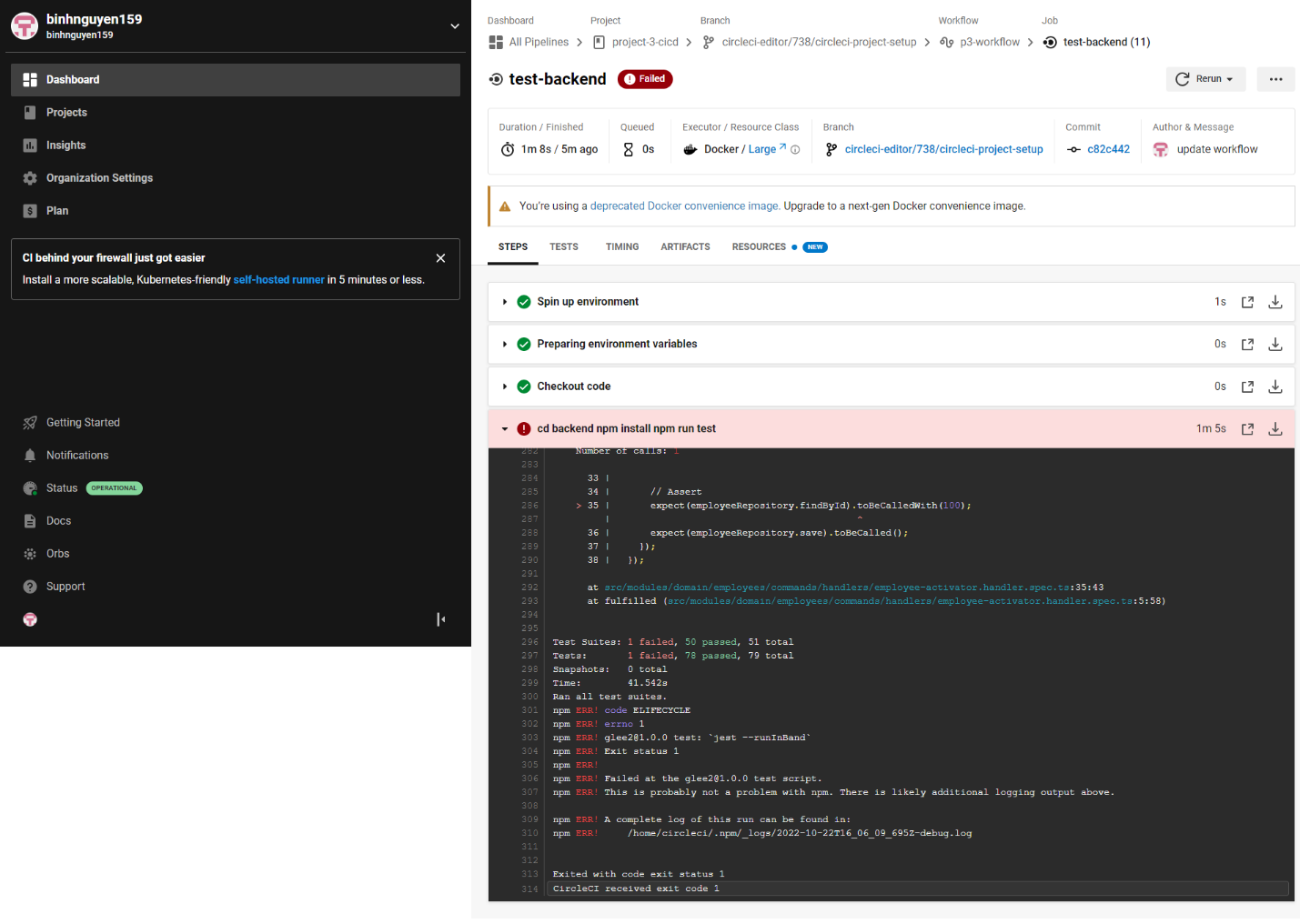
The world is moving toward rapid release cycles, and CI/CD pipelines have accelerated the release rate. With careful planning and implementation, such a pipeline can help you find defects faster, implement fixes immediately, and increase overall customer satisfaction.

1. **Section 2: Deploying Working, Trustworthy Software**

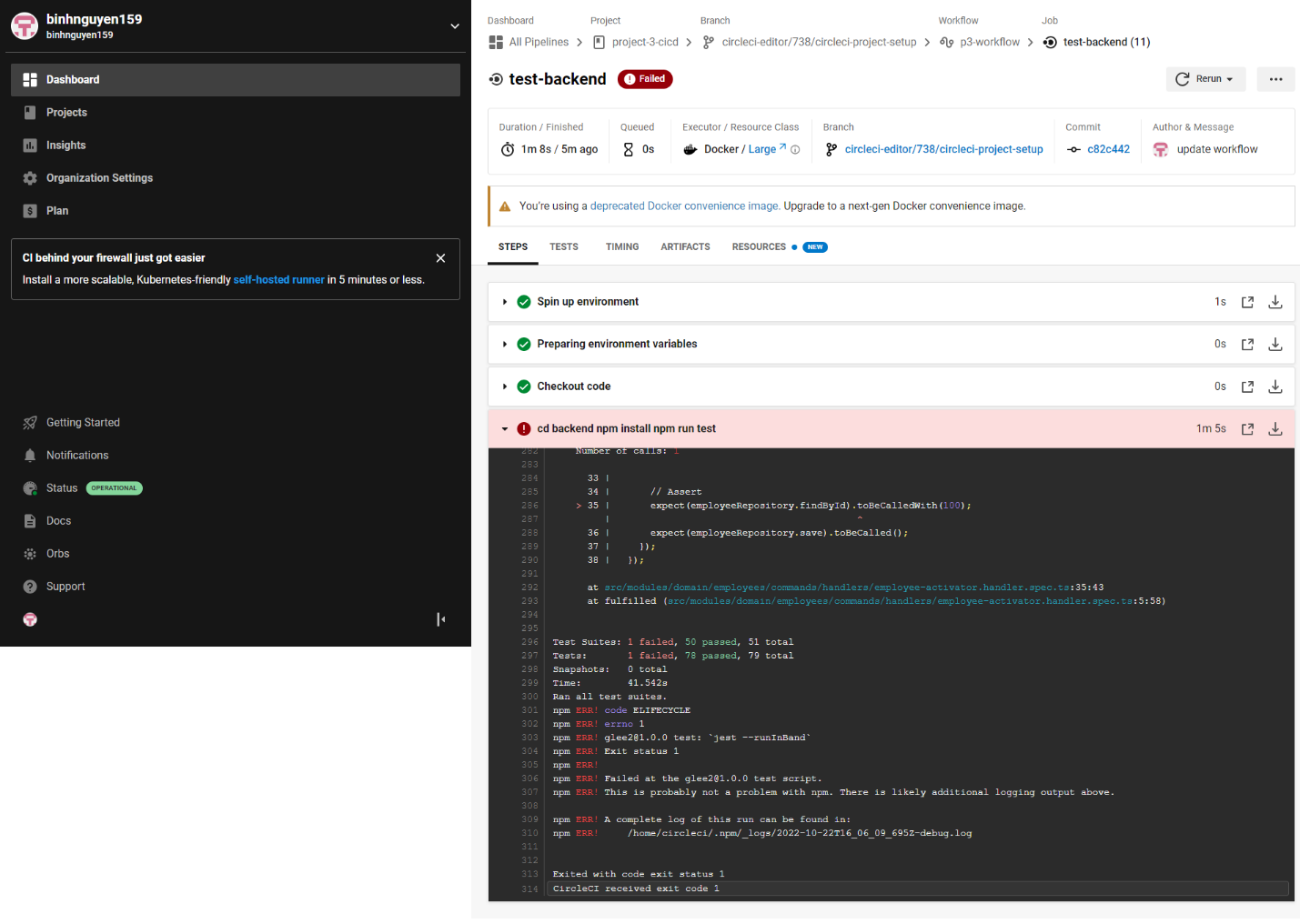
* Job failed because of compile errors. [SCREENSHOT01]



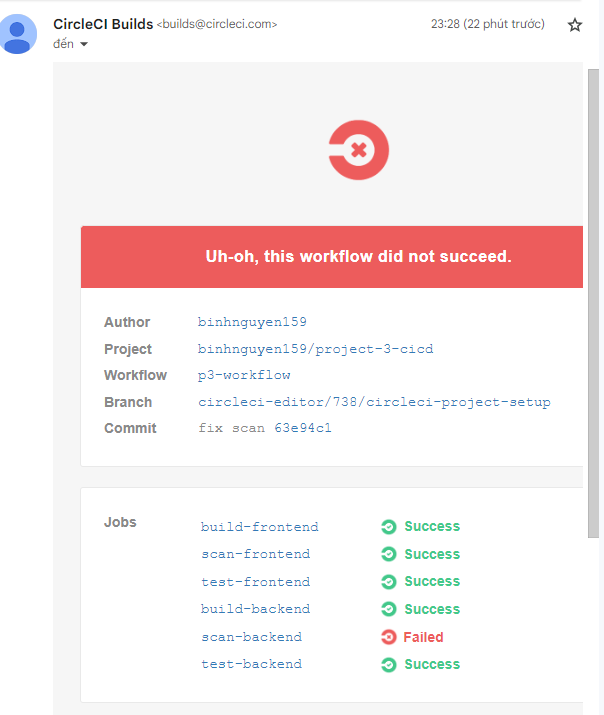
* Job failed because of unit tests. [SCREENSHOT02]



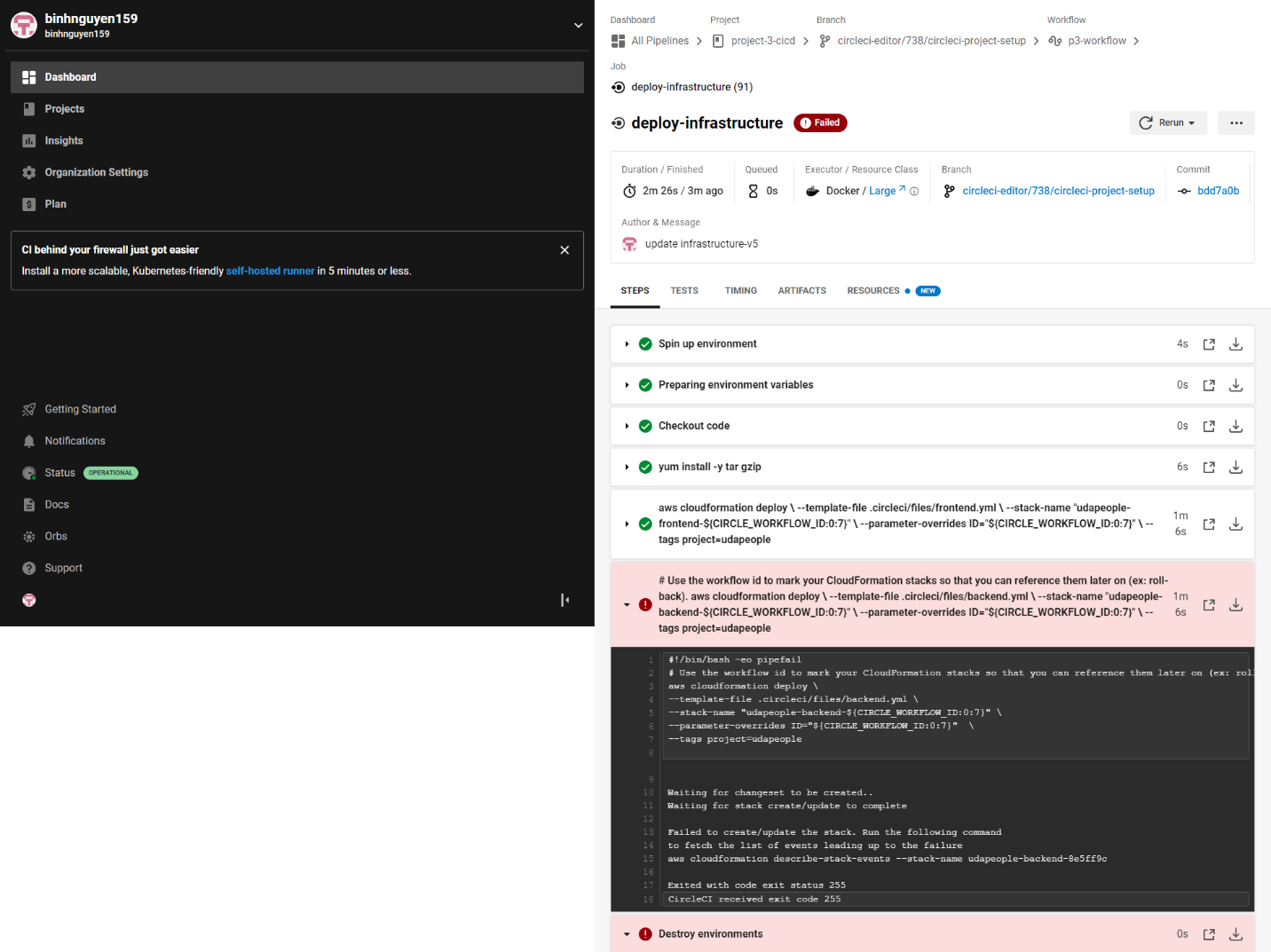
* Job that failed because of vulnerable packages. [SCREENSHOT03]



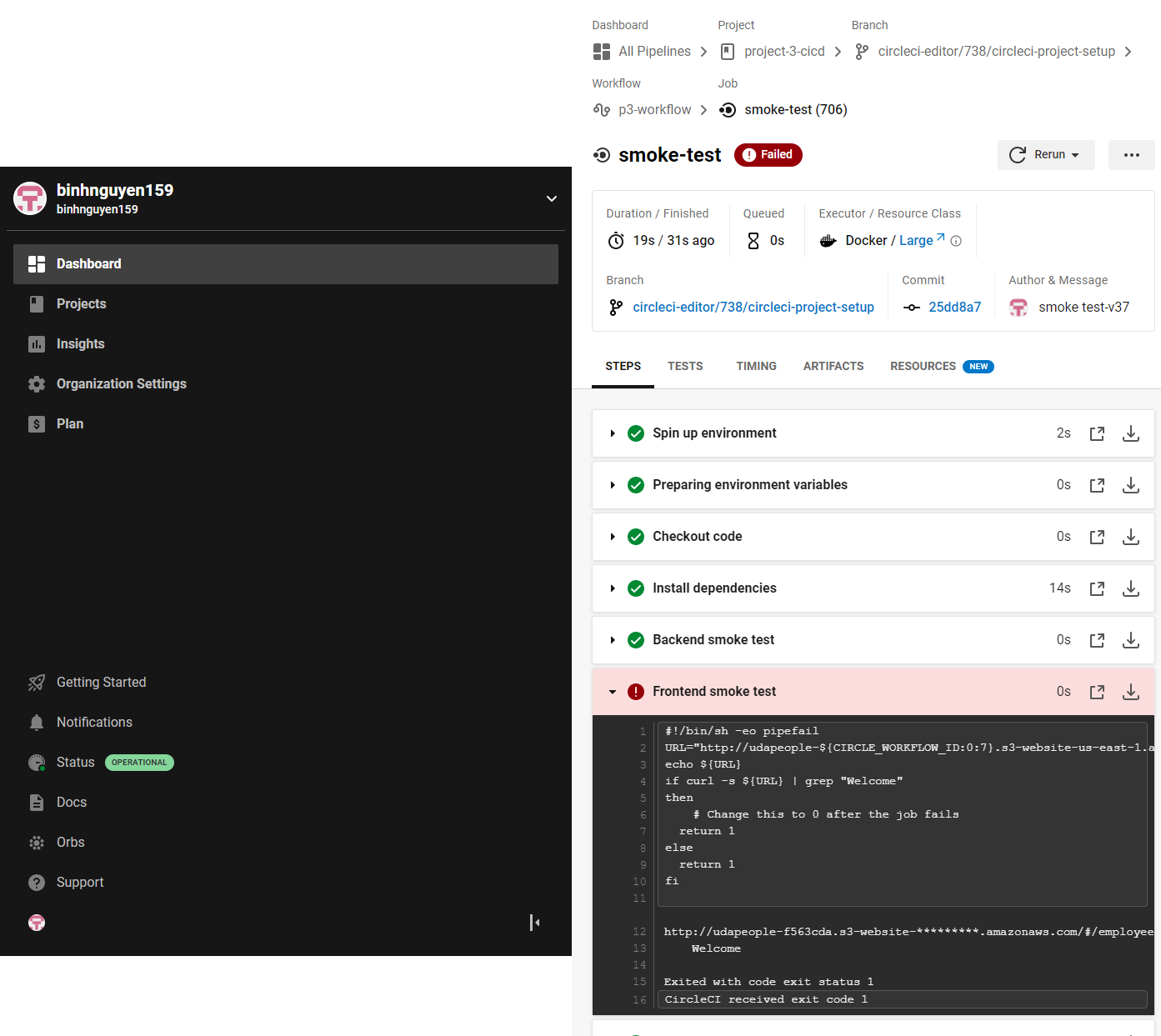
* An alert from one of your failed builds. [SCREENSHOT04]



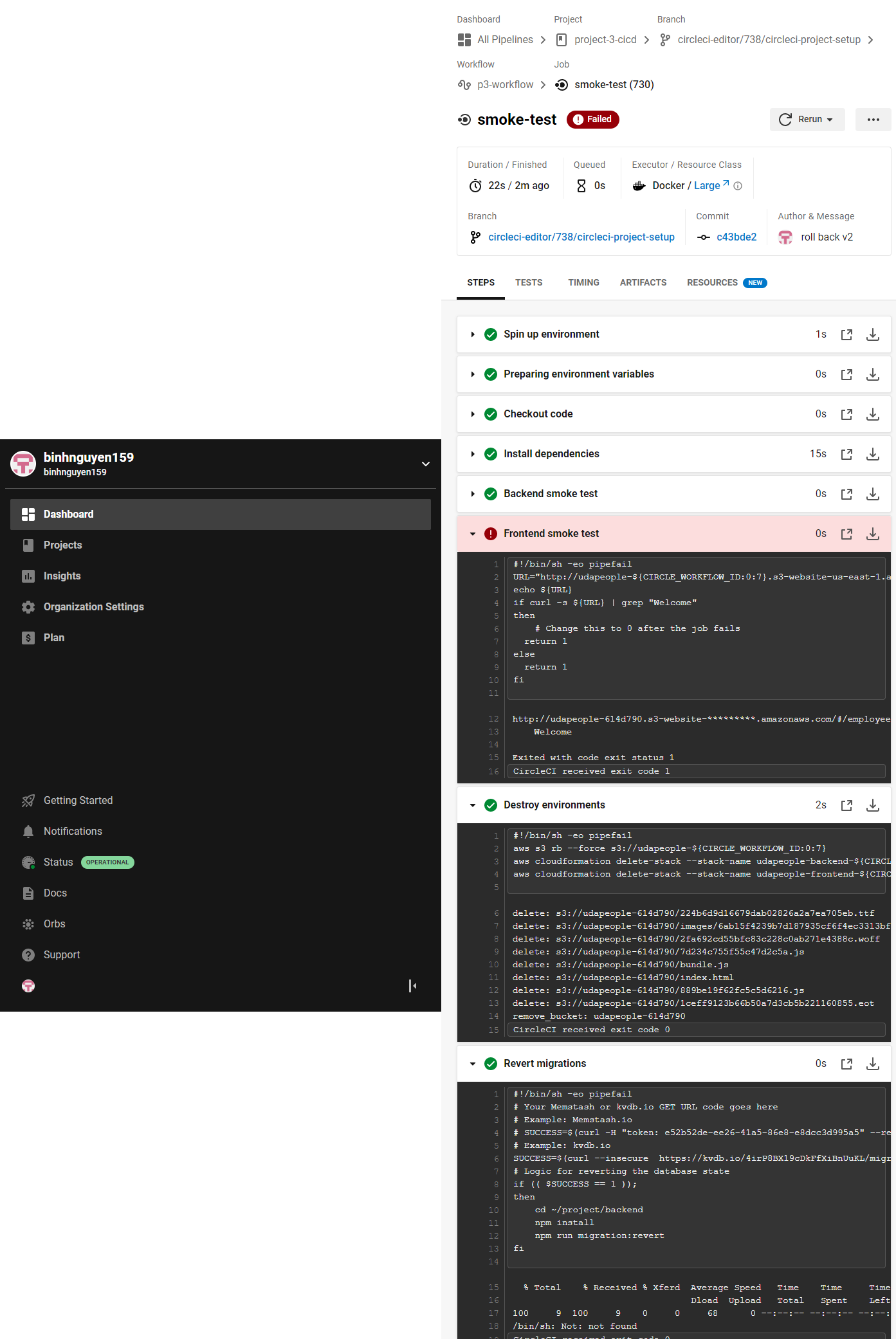
* Appropriate job failure for infrastructure creation. [SCREENSHOT05]



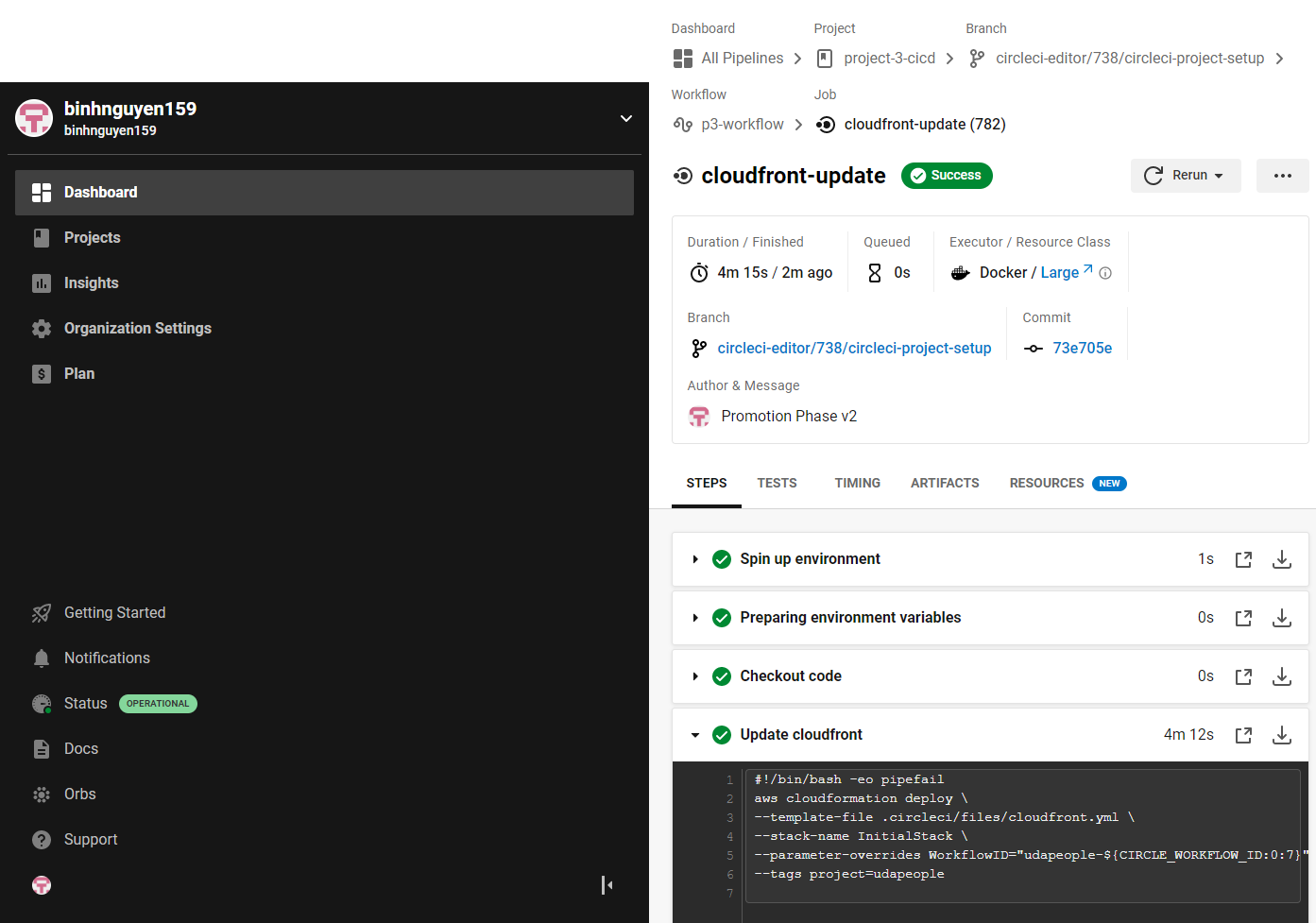
* Appropriate job failure for the smoke test job. [SCREENSHOT06]



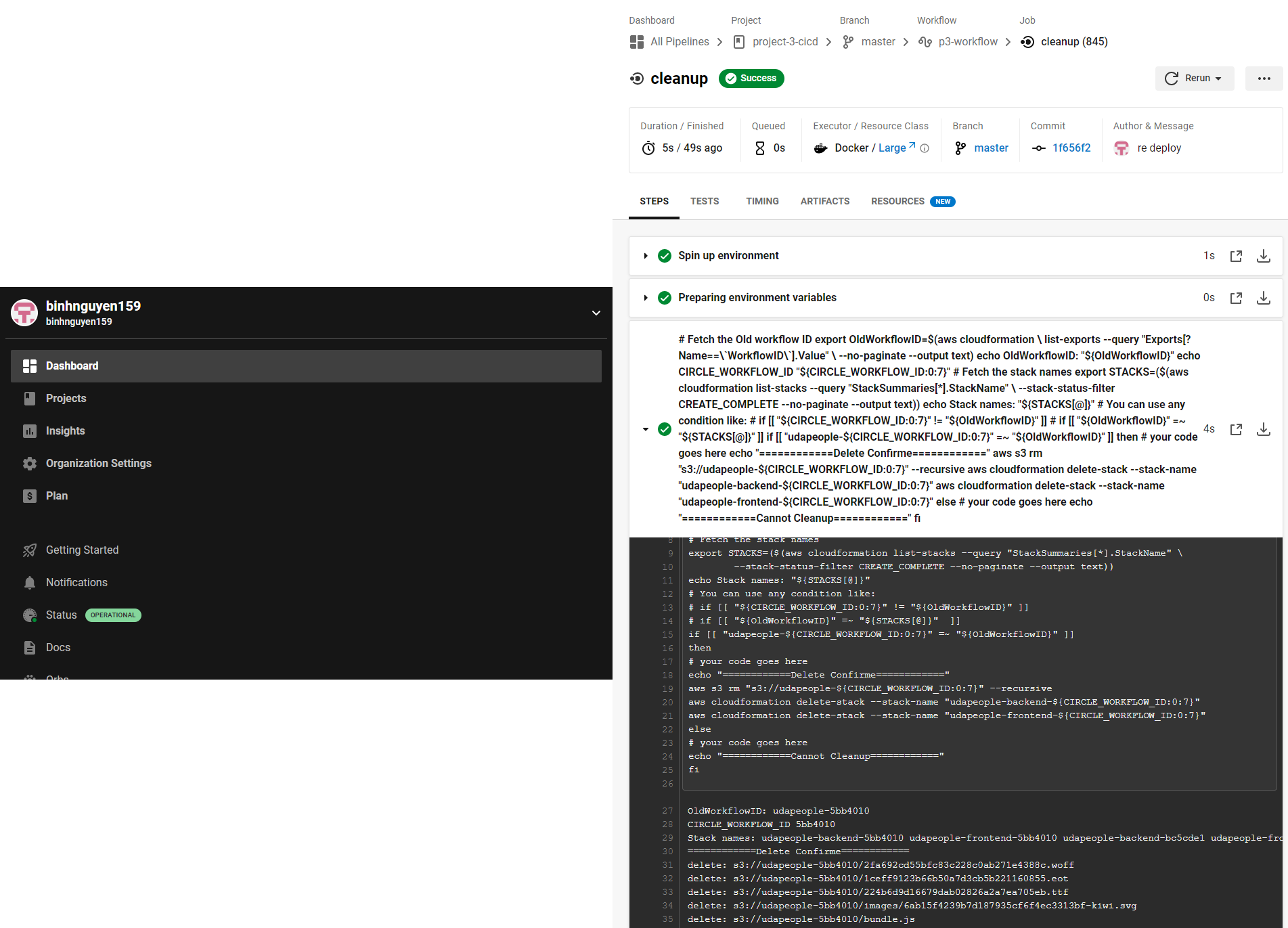
* Successful rollback after a failed smoke test. [SCREENSHOT07]



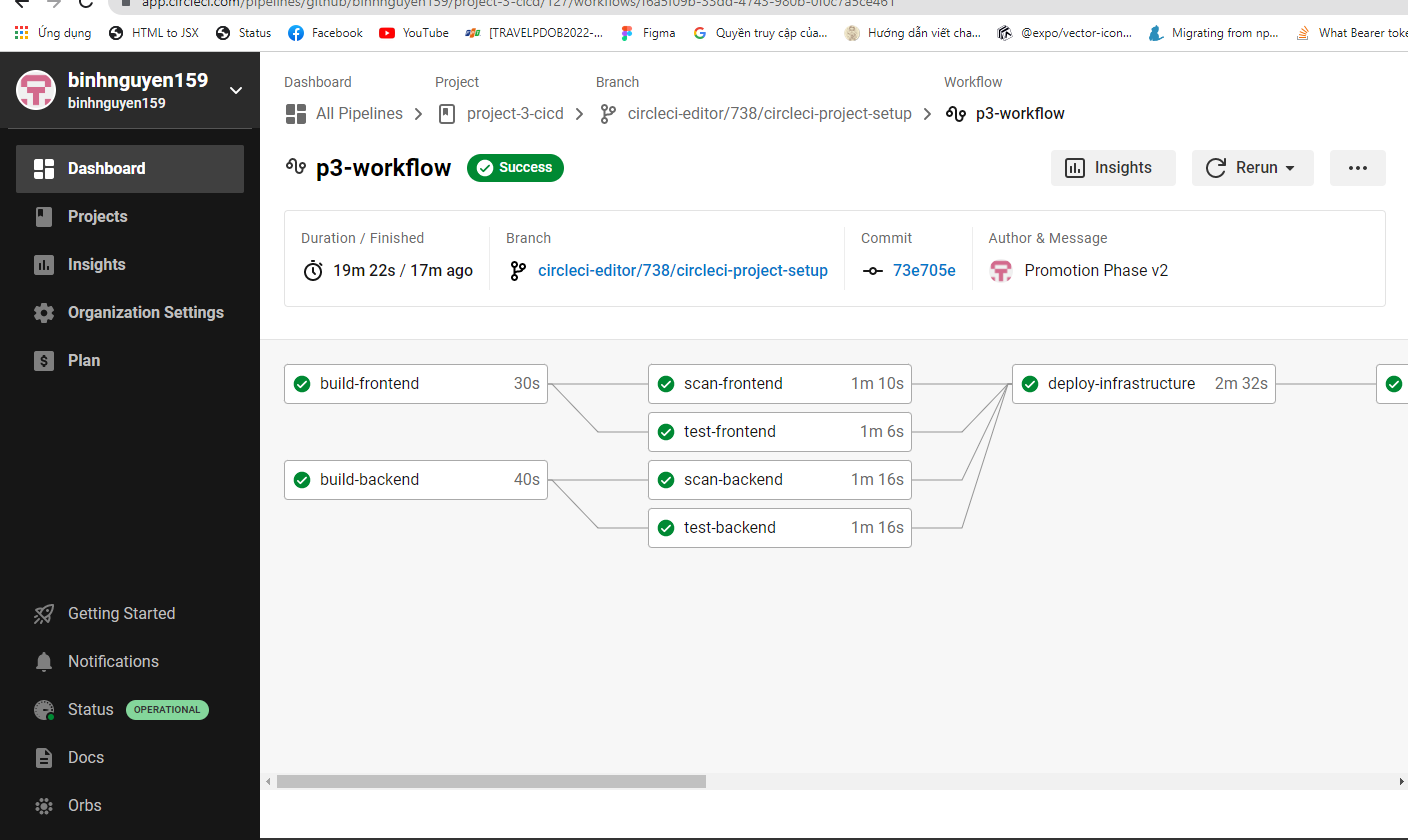
* Successful promotion job. [SCREENSHOT08]



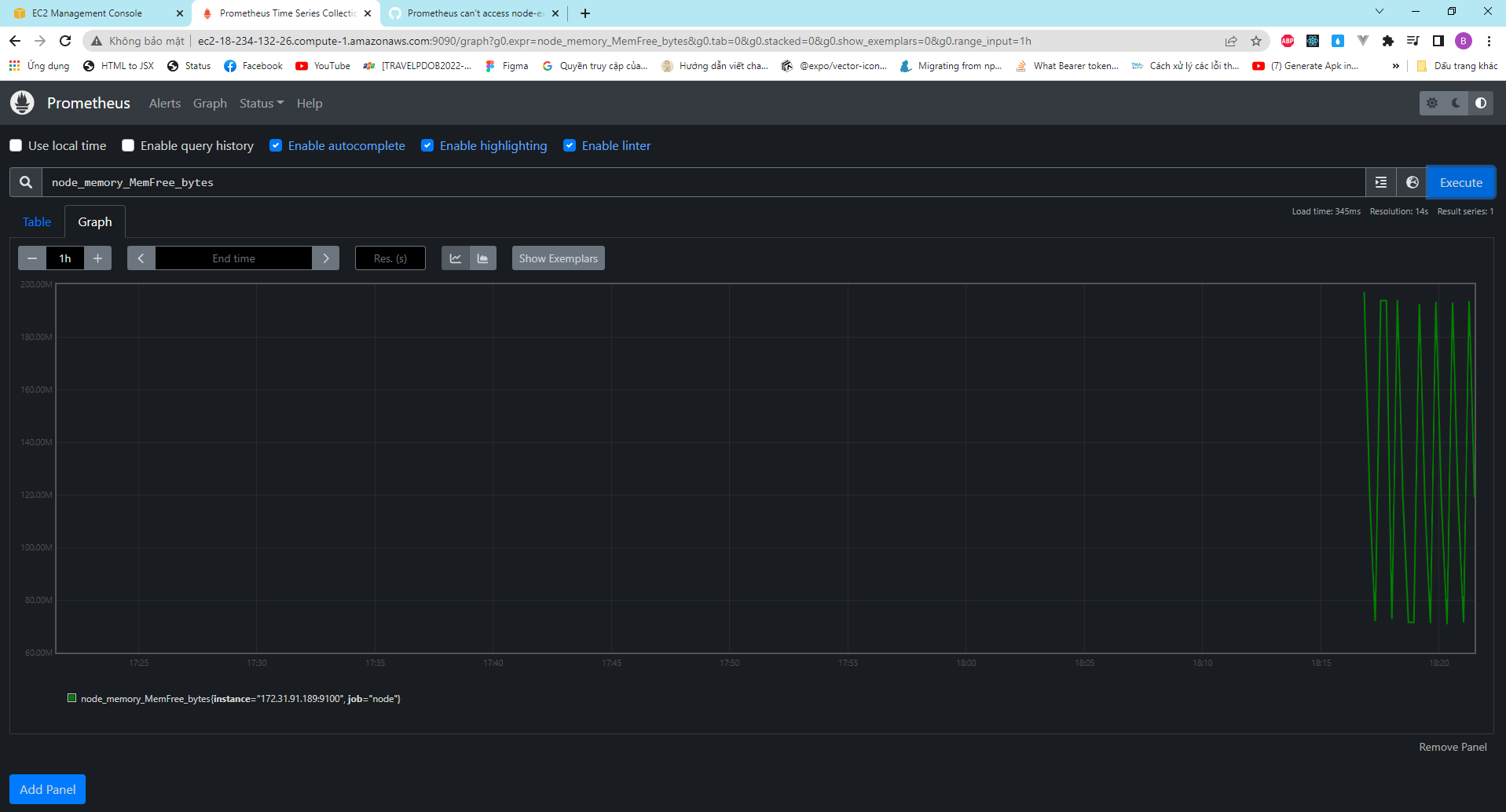
* Successful cleanup job. [SCREENSHOT09]



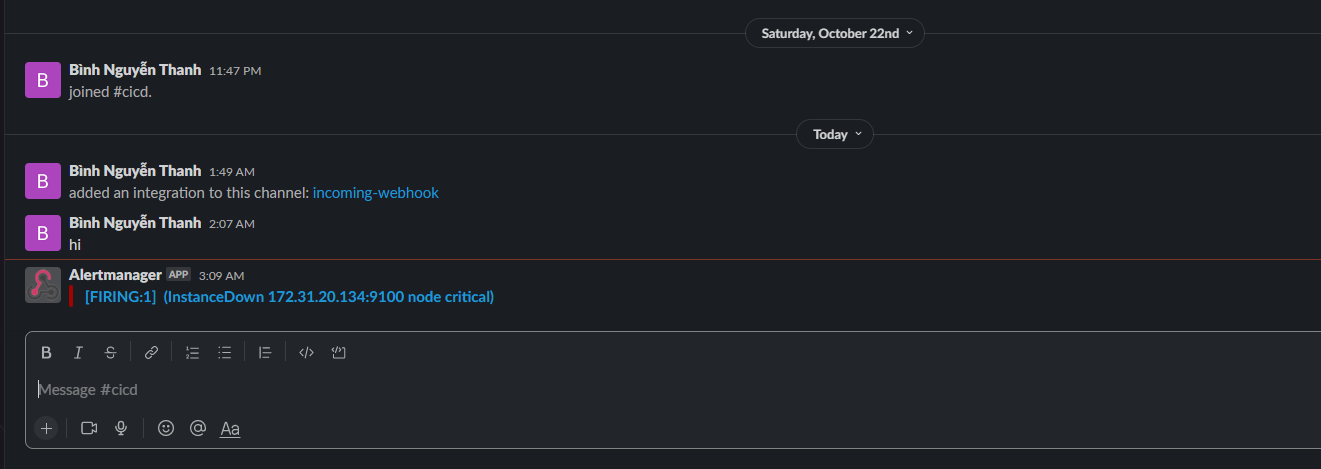
* Only deploy on pushed to master branch. [SCREENSHOT10]



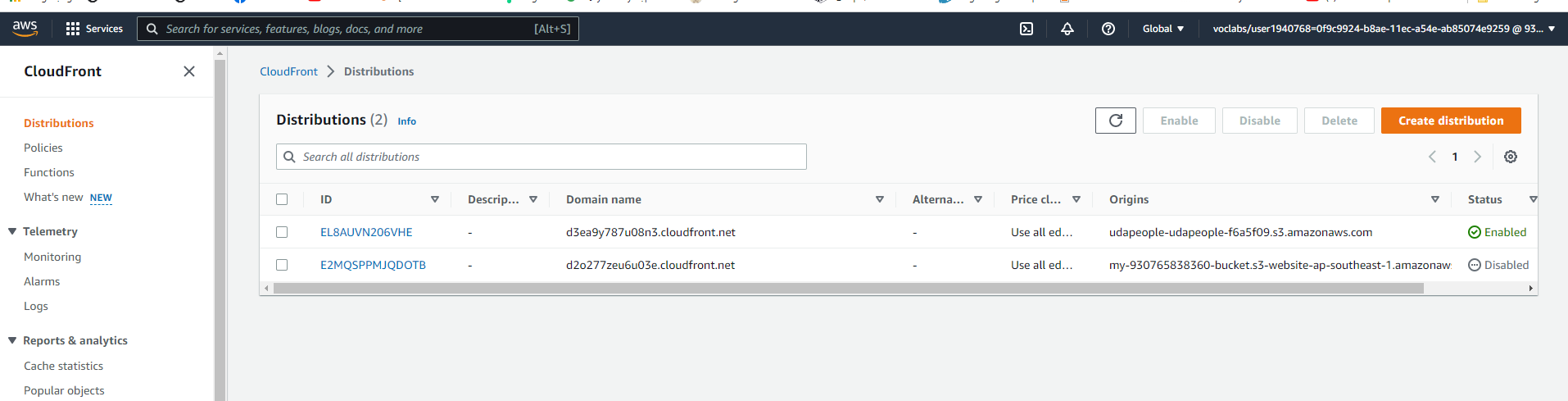
* Provide a screenshot of a graph of your EC2 instance including available memory, available disk space, and CPU usage. [SCREENSHOT11]



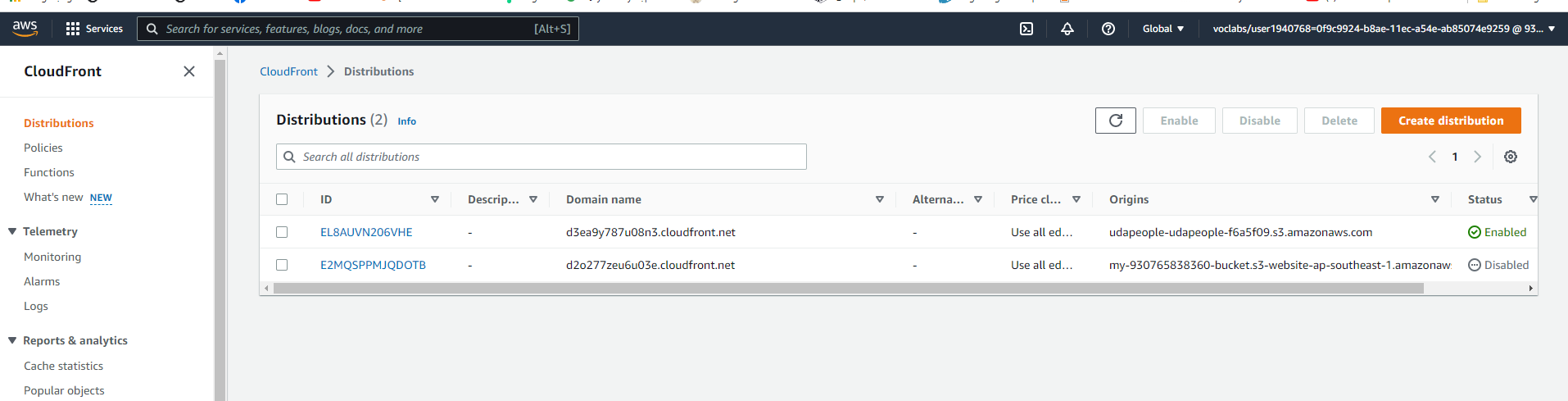
* Provide a screenshot of an alert that was sent by Prometheus. [SCREENSHOT12]



* Provide a screenshot showing the evidence of deployed and functioning front-end application in CloudFront (aka, your production front-end). [URL03\_SCREENSHOT]



* Provide a screenshot showing the evidence of a healthy backend application. The backend endpoint status should show a healthy response. [URL04\_SCREENSHOT]



* Provide a screenshot of your Prometheus server showing UP state [URL05\_SCREENSHOT]

