# Cinder Pipeline Improvements: Automation of Droplet Administrative Overhead

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

**Sion Kim**

Spring, 2023

Briana Morrison, Department of Computer Science

# Cinder Pipeline Improvements: Automation of Droplet Administrative Overhead

CS4991 Capstone Report, 2022

Sion Kim
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia, USA
sk9uth@virginia.edu

**Abstract**
The Amazon EC2 Nitro Firmware team spends around 130 hours of manual work per week on obtaining droplet loans and generating host config files in order to run the Cinder Qualification Testing Pipeline. I created a system that automates this administrative overhead. The system utilizes a droplet capacity management service (re:Stack) to obtain droplet loans, and a python script plus an internal API (AskEC2 Coral) to generate the host config files. The system cuts down on hours of labor weekly and provides a reliable structure that is scalable and maintainable to future Cinder developments and changes. Future work consists of obtaining permanent team credentials for AskEC2 Coral, linking in firmware types from the Cinder configuration page, and adding more usages for the script.

## 1 Introduction and Background

The EC2 Nitro Firmware Team (NFT) is a division of Amazon AWS. One of the NFT's key project workstreams is Cinder Qualification Automation. Cinder is a bootable binary responsible for bringing in the Carbon Operating system on host servers (droplets) used by AWS customers. Before a new update to Cinder is released, NFT needs to run tests on the update. This process is called Cinder Qualification.

Previously, NFT ran Cinder Qualification manually. In the past year, they have implemented a semi-automatic testing pipeline that runs Cinder Qualification. Since the pipeline was bootstrapped quickly, it is not yet fully automatic and requires significant manual administrative overhead. For NFT to run the pipeline, they must borrow droplets of certain host types to run the tests on by applying for loans. Then, they must generate a host configuration file to specify these droplets to the pipeline. In total, droplet administrative overhead takes around 130 hours of manual work per week, which is a significant loss of time. Loss of time (and other problems) arise for these reasons:

1. When applying for droplet loans, NFT must manually click through a UI system called Pawnshark which takes manual effort. In addition, the host types they need may not be available for loaning at all.
2. If a new loan is acquired, the NFT must wait for the loan to build to their testing stack, which takes 5+ hours.
3. NFT has a previously existing script to assist with generating the host config file. However, there are three problems with this script:
   a. The script requires manual updates for values of local variables in code before each use. Changing the code for each use case instead of taking in script parameters takes manual effort and is bad practice.
   b. The script gets droplet information by parsing html from the droplet admiral web page. This is not maintainable long term since the html source can change. This is not scalable since the host config file will eventually need droplet information not displayed on the web page.
   c. Admiral access is required to execute the script because it works by accessing the droplet admiral web page. This is a problem because some employees (including interns) are not allowed admiral access and cannot use the script.

NFT required an automated, scalable, and maintainable solution to these problems in order to reduce droplet administrative overhead and make their Cinder Qualification pipeline fully automatic.

## 2 Related Works

In the computer science industry, there are commonly practiced automation work overflows that handle administrative overhead to increase productivity and shorten the systems' development life cycle. Continuous Integration (CI), Continuous Delivery (CD), and Continuous Testing (CT) are well-used examples. Issue and Project tracking software (such as JIRA) automates the coordination required to delegate tasks across individual workers resulting in efficient collaboration.

Previous works either: 1) presented practical approaches to design and implement such automated systems; 2) conducted socio-technical studies that evaluate the impact of such systems; or 3) designed a new system for a specific use case:

1. Kazlauskas and Picus (1990) presented a "systems approach" to design and implement automated systems for educational administrative purposes using microcomputers.
2. Olson and Lucas (1982) conducted evaluations of the impact of office automation on organizations. Levine and Aurand (1994) developed an event simulation model to evaluate the impact of employing work-flow automation at Pacific Northwest Laboratory.
3. Ganger, et. al. (2003) designed a self-managing system for storage bricks by borrowing organizational ideas from corporate structure and automation technologies from Artificial Intelligence (AI) and control systems.

While automation for administrative work has certainly been done before, the solution NFT needed was for a very specialized use case. Automation technology is designed to perform well only in the context of its intended task. It is hard to design a general all-purpose technology that will have good performance in every situation, since it requires adaptable intelligence lacking in traditional computational systems. Even with the use of AI, true human-like thinking is difficult to achieve across all domains. Therefore, it was reasonable to implement a new solution for this specific situation with NFT.

## 3 Project Design

The NFT has a previously existing semi-automatic pipeline to handle Cinder Qualification. Stage 2 of the pipeline requires manual droplet administrative overhead of obtaining droplet loans and generating host configuration files. I designed an independent system that automates these tasks that fully automates the existing pipeline.

### 3.1 Overview of Cinder Qualification Pipeline

Cinder Qualification is the process of testing a new commit to the mainline Cinder image before release. Cinder Qualification is run through a semi-automatic testing pipeline. The steps of the testing pipeline can be seen in Figure 1. The stages of the Cinder Qualification pipeline are activated by a new commit to the mainline Cinder image. This pipeline is semi-automatic; all other Stages are automatically handled by the pipeline except for Stage 2. NFT must manually obtain droplets of certain required host types by applying for loans. Then, they must generate a host configuration file to specify these droplets to the pipeline. Stage 2 contains significant manual droplet administrative overhead.

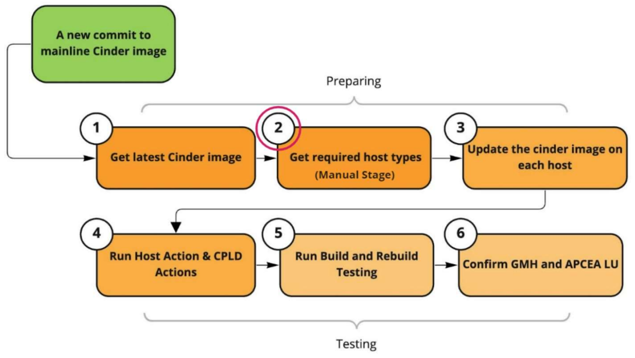
A new commit to mainline Cinder image
Preparing
1
Get latest Cinder image
2
Get required host types
(Manual Stage)
3
Update the cinder image on each host
4
Run Host Action & CPLD Actions
5
Run Build and Rebuild Testing
6
Confirm GMH and APCEA LU
Testing

Figure 1. Diagram of the Semi-Automatic Cinder Qualification Pipeline

### 3.2 Problems with Manual Droplet Loans and Host Config File Generation

The manual droplet administrative overhead in Stage 2 of the Cinder Qualification Pipeline arises from droplet loans and host configuration files. When applying for droplet loans, NFT must manually click through a UI system called Pawnshark. In addition, the host types they need may not be available for loaning at all. If the required host type is available and a new loan is acquired, NFT must wait for the loan to build to the right testing stack, which takes five or more hours. Since NFT runs Cinder Qualification for up to 26 required host types per week, the total process can take around 130 hours per week.

After obtaining droplets, NFT must generate a host configuration file to specify these droplets to the pipeline. NFT has a previously existing script to assist with generating the host config file. However, this script relies on html parsing of each droplet's web page to obtain droplet information, which causes four problems. First, the script requires copying and pasting the url for each droplet's web page to a local variable before each use. Second, the script is not maintainable since the html source code can change. Third, the script is not scalable since the host config file will need droplet information not available on the web page in the future. Fourth, the script requires admiral permissions to execute, they are required to access the droplet web page.

### 3.3 Implemented Solution

I designed an independent system that can be run and used by the existing Cinder Qualification Pipeline to make the pipeline fully automatic. To implement the system, I utilized python scripting and two internal services: reStack and AskEC2 Coral API. re:Stack automatically maintains a set capacity of droplets. The service automatically creates loans to maintain a set number of droplets of certain host types, and automatically builds the loaned droplets to NFT's testing stack. With re:Stack, NFT always has a pool of correctly configured, "ready to go" droplets that can be used instantly for Cinder qualification. AskEC2 Coral API can return droplet information queried by request parameters. AskEC2 Coral has a wide range of schema and many input parameters that allow specific queries and return of detailed information about droplets.

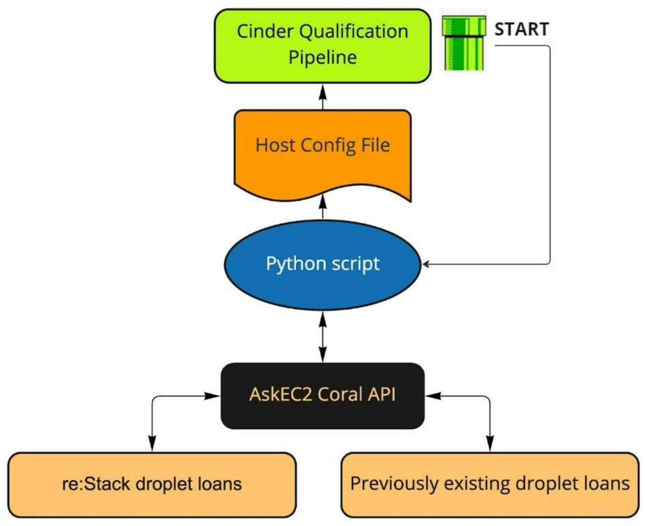

Figure 2. Workflow of System for Droplet Administrative Overhead Automation

The workflow of the system is shown in Figure 2. The system is started when the Cinder Qualification pipeline runs the python script. The python script makes an API request to AskEC2 Coral to obtain droplet information for all droplets owned by NFT configured to the testing stack. The response returns droplet information for correctly configured droplets loaned by re:Stack, and also existing droplet loans obtained manually. The python script then parses the returned droplet information into the correct format and writes it into a host config file. The generated host config file is fed back into the Cinder Qualification Pipeline to run Cinder Qualification testing.

## 4 Results

The Amazon EC2 Nitro Firmware team spends around 130 hours of manual work per week on obtaining droplet loans and generating host config files in order to run the Cinder Qualification Testing Pipeline. I created a system that automates this administrative overhead, and fully automates Cinder Qualification testing pipeline. The system utilizes re:Stack and the AskEC2 Coral API to eliminate manual work and previous problem points, and provides a reliable structure that is scalable and maintainable to future Cinder developments.

## 5 Conclusion

The Amazon EC2 Nitro Firmware team spends around 130 hours of manual work per week on obtaining droplet loans and generating host config files in order to run the Cinder Qualification Testing Pipeline. I created a system that automates this administrative overhead, and fully automates Cinder Qualification testing pipeline. The system utilizes re:Stack and the AskEC2 Coral API to eliminate manual work and previous problem points, and provides a reliable structure that is scalable and maintainable to future Cinder developments.

## 6 Future Work

The AskEC2 Coral API does not require admiral permissions. However, it does require a certificate from the client obtained through registration. I was able to register my Amazon user id to obtain a temporary certificate. I was not able to accomplish permanent team registration due to the short duration of my internship. The next step would be registering the team's group alias to obtain a permanent team certificate that members of the NFT and the Cinder Automatic Pipeline could use.

Another future step would be adding command line parameters to the python script so that different types of queries and droplet information generation would be possible. For instance, the script could generate a host config file for droplets of a specific host type only, or for only ten droplets, based on the command line parameters. The NFT will have different test cases for Cinder Qualification and their other products to make use of command line parameters.

## 7 Acknowledgments

I would like to thank the members of the EC2 NitroFirmware Team (NFT) for their support, and for giving me this internship opportunity. I would like to especially thank my mentor, Gauri Bendre, and my manager, Paru Subbiah. In addition, I would like to acknowledge Jarrett Tinerney and Tasnim Makada from the EC2 NitroService Team (NST) for their assistance.

## References

[1] Gregory R Ganger, John D Strunk, and Andrew J Klosterman. 2003. Self-storage: Brick-based storage with automated administration. Technical Report. Carnegie-Mellon University, Pittsburgh PA School of Computer Science.
[2] Edward John Kazlauskas and Lawrence O Picus. 1990. A Systems Analysis Approach to Selecting, Designing and Implementing Automated Systems: Administrative Uses of Microcomputers in Schools. *ERIC.*
[3] Lawrence O Levine and Steven S Aurand. 1994. Evaluating automated work-flow systems for administrative processes. *Interfaces* 24, 5 (1994), 141–151.
[4] Margrethe H. Olson and Henry C. Lucas. 1982. The Impact of Office Automation on the Organization: Some Implications for Research and Practice. *Commun. ACM* 25, 11 (nov 1982), 838–847. https://doi.org/10.1145/358690.358720