MFE Internship Report

Internship for
Software-defined Storage
at SUSE

Joël Lin LIU

An internship report presented for the
diplôme d'ingénieur
of
École Centrale de Pékin

Internship Period:	October 8, 2018 - March 29, 2019
Company:	SUSE (Beijing, China)
Internship Mentor:	PeiFeng LIN
ECPkn Tutors:	ChuanTao YIN, Ahmed RAHMANI

Contents

Acknowledgment

Before the specification of this report about my internship of Final Study Mission at SUSE Beijing, I would like to express my sincerest thankfulness to:

- Mr. Peifeng LIN for coaching me during the entire internship, taking his time to teach me about the basic knowledge of the company products: SUSE Enterprise Linux Server and SUSE Enterprise Storage, and also the use of Linux operating system, and giving me lots of time and opportunities to learn new things that I had never known before. Thanks for his patience and kindness.
- Mr. Bary ZHU and Ms. Jessica LI for offering me the opportunity to work with so many intelligent and kind people in such an international and fast-growing company, which also opened my eyes to a lot of interesting things such as Linux, open source community and Software-defined distributed storage, and strengthened my professional abilities to work with a multicultural team to deal with problems so that I can be well prepared for my future career.
- All the other colleagues at SUSE, especially Jiaqi, Arthur, Alex, Lance, Lydia and Zejin, for their enthusiastic reception, support and the companion.

Last but not least, I also want to express my thanks to my lovely roommates - Adam, Paul and Hugo, who give me a hand in being always positive and confident in life.

1 Introduction

Finishing the MFE (Mission de Fin d'Étude) internship is a necessary step before we the students graduate from the École Centrale de Pékin (ECPkn), step into the industry and begin our professional career. This demands us to involve in a project, work with a team, and also be well prepared for the future professional life in a company in the end.

I therefore chose to take an internship at SUSE Beijing to work around the company's outstanding product - the **SUSE Enterprise Storage (SES)**, just out of my curiosity in Linux and open source spirit, and also my previous knowledge of its another famous product, openSUSE (community version of SUSE Enterprise Linux distribution).

Besides, SES is a company product developed based on the open source project **Ceph** using distributing storage design, such a work would help me learn to not only involve in the development of open source project, but also take professional software development practice for a real company product, which is thus perfect for me as the MFE internship.

Inside the SES product, an important feature is the **openATTIC**, an open source system with GUI, which can greatly simplify the daily management and monitoring of the Ceph distributed system. While this is an open source project with many contributors involved, there is still a lot of work to improve. The first thing is that one version developed for commercial use has diverged from the open source version, and some code merging and feature transplanting tasks need help.

And another thing important is in the newest SES beta program, openATTIC has been replace by new **Ceph dashboard** module. But it doesn't mean openATTIC is abandoned because Ceph dashboard is just derived from openATTIC code base, openATTIC is just becoming the new Ceph dashboard gradually. However, for this time, the coexistence of these two projects has made inconvenience to developers, and the first thing is different APIs without any documentation, therefore, an effort to collect and sort out these APIs is necessary.

All in all, during this internship, my work will concentrate on the openATTIC and Ceph dashboard, but also Ceph operations,for which there is also lack of well-documented medium. And the final work results will be distributed in different ways according to the demands, not limited to documentations, websites, or code base.

This report is redacted to present and analyze the work during my internship at SUSE Beijing. Firstly, an introduction is given about the company and the internship details, where basic concepts of technologies used will be explained and mission and planning will be also given. Then, major problems during the internship will be listed, and the corresponding strategies will be detailed. The results will be introduced next for three parts with respect to three major contributions during this internship. And a final conclusion is given for emphasizing what I have achieved and what I have learned.

1.1 Introduction to Company

SUSE is a German-based, multinational, open-source software company that develops and sells Linux products to business customers. Founded in 1992, it was the first company to market Linux for the enterprise. It is also the primary sponsor of the community-supported openSUSE Project, which develops the openSUSE Linux distribution.[1]

SUSE, a pioneer in open source software, provides reliable, software-defined infrastructure and application delivery solutions that give enterprises greater control and flexibility. SUSE's solutions range from enterprise Linux to OpenStack private Cloud, software defined distributed storage, Kubernetes container management and Cloud Foundry Platform as a Service, see Figure 2.

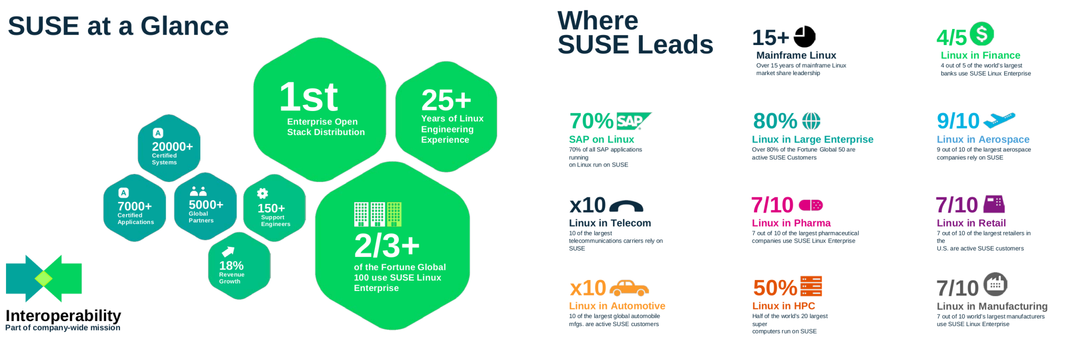

Figure 1: SUSE at a galance

Figure 2: SUSE's *portfolio* of products

Over the years, SUSE has changed owners several times. First, it was acquired by Novell in 2004. Then, Attachmate, with some Microsoft funding, bought Novell and SUSE in 2010. This was followed in 2014, when Micro Focus purchased Attachmate and SUSE was spun off as an independent division [2]. And in July 2018, Micro Focus announced its plan to sell the business unit to a subsidiary of EQT[1] Partners in the first quarter of calendar year 2019. This acquisition was completed on March 15, 2019, making SUSE a standalone business again and also the industry's **largest independent open source company** after the acquisition of Red Hat by IBM in late 2018.

[1] EQT is a leading investment firm with more than EUR 61 billion in raised capital across 29 funds and around EUR 40 billion in assets under management. EQT funds have portfolio companies in Europe, Asia and the US with total sales of more than EUR 19 billion and approximately 110,000 employees. EQT works with portfolio companies to achieve sustainable growth, operational excellence and market leadership. [3]

2 Internship Presentation

2.1 Background

IDC estimates that by the year 2020, worldwide there will be 40 zetabytes (ZB) of data. This represents a 5000% growth rate from 2010 to 2020. The data explosion continues!

As businesses evolve to become more data driven, managing the explosion of data is proving to be a significant challenge. Solving the need for extra capacity is pushing existing storage administration personnel beyond their capabilities, requiring additional operational investment [4].

And limiting factors of traditional enterprise storage include:

- ∗ Difficult to scale and manage data growth
- ∗ Expensive
- ∗ Won't extend to the software-defined data center

SUSE Enterprise Storage is an intelligent software-defined storage management solution, powered by Ceph Technology that enables IT benefit from [5]:

- **Cost-effective storage.** Use industry-standard hardware for SDS to save both short- and long-term CAPEX costs. SUSE is the price leader for archive, bulk and back-up storage.
- **Quick, infinite scalability.** Scale quickly and easily. Need more storage? Just drop in an industry-standard server. Scale from terabytes to a petabytes quickly and cost effectively, significantly reducing OPEX costs. An up to 200% increase in write performance means multi-node environments that value performance and capacity can enjoy fast-scale cost advantages.
- **Automated management.** Reduce complexity and minimize the time administrators spend managing storage: An intelligent, self-healing, self-managing distributed storage solution automates almost everything, so you can support morecapacity per storage administrator. New SALT-based orchestration reduces many management tasks from days to hours and from hours to minutes.
- **Store data of any type.** Unified block, object and file storage adds native file-system access, so customers can store data from Windows, Linux, UNIX, Mac and more.
- **Build in redundancy automatically.** Every time you add to the SDS infrastructure, you also add resiliency into your storage solution, no matter where it's located.

- **Manage intuitively.** Manage in real time. Use the advanced graphical user interface with openATTIC to simplify management and improve cost efficiencies. Get a real-time view of what's happening in the cluster, with immediate alerts and notifications.
- **Upgrade without disruption.** The entire process is automated and occurs in the background.

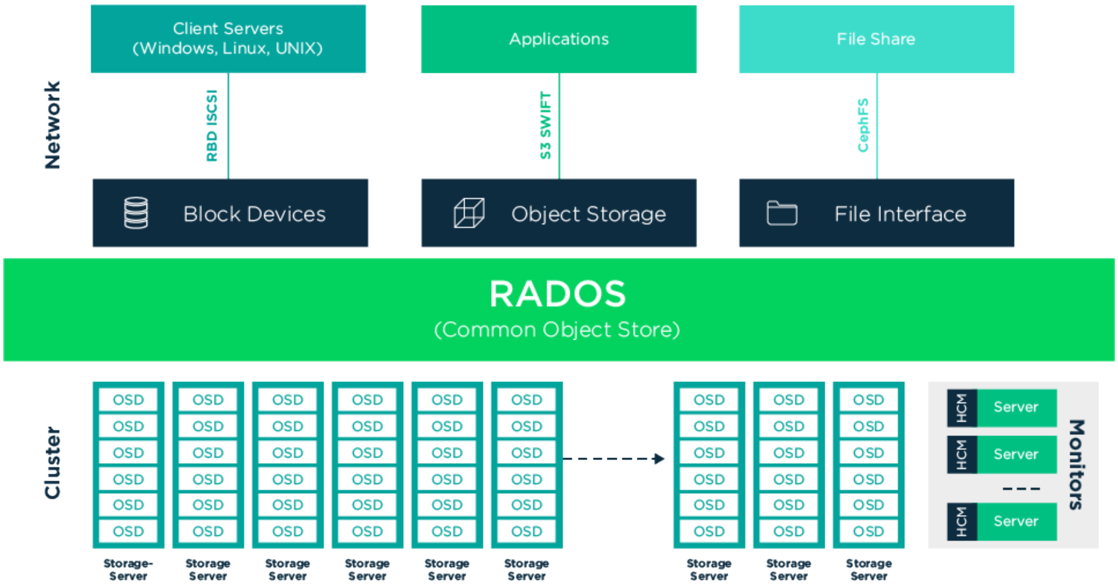

Figure 3: SUSE Enterprise Storage architecture

While we continue to create success stories and receive positive reviews from partners and customs, the SUSE Enterprise Storage product is still under active development and evolves towards the target of better serving the customers. Hence, an engagement in this project is obviously not only necessary for the continuous improvement and growth of a company product, which no doubt benefits our clients a lot, but can also possibly rewards back to the open source community, which is in line with the definition of ourself as an open source company.

2.2 Introduction to Concepts

Software-Defined Infrastructure

Software-defined infrastructure (SDI) [6] combines software-defined compute (SDC), software-defined networking (SDN) and software defined storage (SDS) into a fully

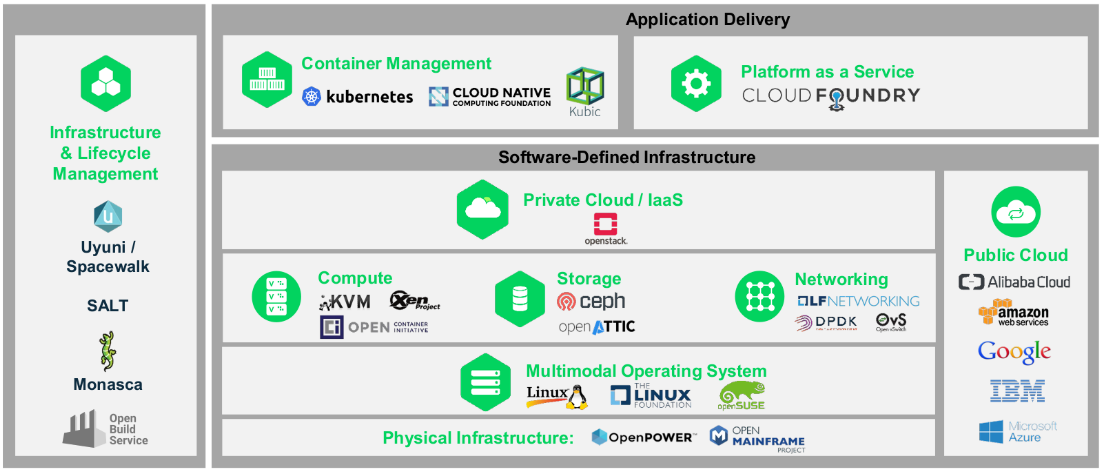

Figure 4: SUSE Software-defined Infrastructure and application delivery approach

software-defined data center (SDDC). A software-defined data center is an IT facility where infrastructure elements such as networking, storage, processing and security are virtualized and delivered as a service. With SDI, software can control the entire computing infrastructure without human intervention. SDI is hardware independent and programmatically extensible, providing unlimited growth potential for heterogeneous environments.

The SDI model allows many critical IT functions to be fully integrated and automated, such as backups and data recovery. Applications can specify and configure the hardware they need to run on as part of their code. Thus, SDI automatically handles application requirements, data security and disaster-preparedness functions. Software-defined infrastructure is open source, allowing IT resources to be flexibly configured per-application on commoditized hardware. This improves data center agility and efficiency while decreasing hardware costs. SDI supports configuration rollback and cloning by versioning the data center landscape. Management dashboards can be used to provision and monitor the software-defined infrastructure. SDI is capable of placing workloads in private or public clouds.

Software-Defined Storage

Software-defined storage (SDS) [7] is a computer platform that creates a virtualized network of storage resources by separating the management software from its underlying storage hardware. SDS resources may be spread across multiple servers and pooled or shared as if they reside on one physical device. Businesses using applications that generate large amounts of unstructured data (such as data analytics, genomics and multimedia websites) or DevOps environments that require flexible storage provisioning for new applications may adopt SDS technology to add storage capacity as needed. Software-defined storage is part of a larger industry trend that also includes software defined networking (SDN), software-defined infrastructure (SDI) and software-defined data center (SDDC).

Enterprise use of software-defined storage is increasing due to the rapid accumulation of unstructured data (such as email messages, documents, videos, graphics, audio files and web pages) and the need for scale-out object storage and file storage; the availability of high-performance server hardware with multicore processors; and the growth of virtualization in servers, desktops, applications and networking. Software-defined storage apps can run on industry-standard hardware and enable users to upgrade the software separately from the hardware. Thus, SDS makes storage more affordable and agile than traditional storage area network (SAN) and network-attached storage (NAS) systems running on proprietary hardware that is tightly coupled to its software.

Ceph – Open Source Project

Ceph [8] is a software defined storage (SDS) platform that unifies the storage of block, object and file data into a distributed computer cluster. It is a component of the OpenStack set of open source cloud management tools. Ceph storage clusters can run on standard x86 servers, using the Controlled Replication Under Scalable Hashing (CRUSH) algorithm to distribute data evenly across the cluster. This allows cluster nodes to access data without the bottlenecks common to centralized storage architectures.

Ceph is scale-out, allowing multiple storage nodes (servers) to cooperate and present a single storage system with unlimited capacity. It also provides storage management tools such as thin provisioning, erasure coding (replication), inline compression and cache tiering. Ceph is the only storage platform that is open source, software defined, enterprise class and unified (object, block and file). It supports iSCSI block storage, allowing Linux, VMWare, Unix and Windows servers to access block storage from a Ceph cluster. En-

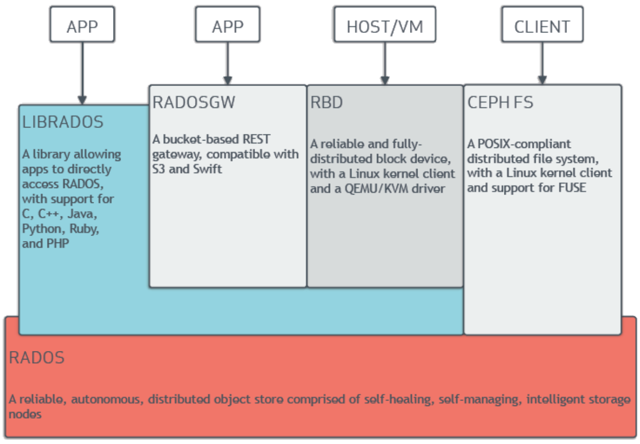

Figure 5: Ceph architecture diagram
source: Ceph Documentation

terprise storage administrators use Ceph for its unified storage management, multi-platform support, scalability and built-in fault tolerance.

SUSE Enterprise Storage is based on Ceph technology. It allows storage to manage itself when conditions within the cluster change. For example, if a node goes down, the storage software automatically redistributes that node's workload. If a new node is added to the cluster, Ceph incorporates it into the storage system and distributes data to it. The openATTIC storage management tool provides a central GUI console for managing the cluster, in addition to the command-line utilities included in SUSE Enterprise Storage. Enterprises can quickly provision or de-provision data storage as business needs change. Ceph automatically rebalances the storage cluster without human intervention.

openATTIC and Ceph Dashboard

openATTIC [9], a project sponsored by SUSE and joins SUSE finally in 2016, is an *Open Source Management and Monitoring System* for the Ceph distributed storage system.

Various resources of a Ceph cluster can be managed and monitored via a web-based management interface. It is no longer necessary to be intimately familiar with the inner workings of the individual Ceph components. Any task can be carried out by either using openATTIC's clean and intuitive web interface or via the openATTIC REST API.

The Ceph Dashboard [10] is a built-in web-based Ceph management and monitoring application to administer various aspects and objects of the cluster. It is implemented as a Ceph Manager Daemon module.

The original Ceph Dashboard that was shipped with Ceph Luminous started out as a simple read-only view into various run-time information and performance data of a Ceph cluster. It used a very simple architecture to achieve the original goal. However, there was a growing demand for adding more web-based management capabilities, to make it easier to administer Ceph for users that prefer a WebUI over using the command line.

The new Ceph Dashboard module is a replacement of the previous one and adds a built-in web based monitoring and administration application to the Ceph Manager. The architecture and functionality of this new plugin is just derived from and inspired by the openATTIC. And the development is also actively driven by the team behind openATTIC at SUSE, with a lot of support from companies like Red Hat and other members of the Ceph community.

2.3 Mission Description

This internship will focus on, step by step, several major works around openATTIC and Ceph Dashboard, including:

1. Setting up the software-defined storage clusters with both bare and virtual machines and enabling the management and monitoring modules on them;
2. Involving in the improvement and development of openATTIC project, helping merging two version releases and cleaning the source code, and fixing bugs based on testing under the environment setted up in the first step;
3. Providing and arranging systematically the REST APIs of openATTIC and Ceph Dashboard, distributing the results in a clear way (website, application, or technical manuals, etc) to help ease the further development;
4. Composing a complete and tested documentation for Ceph operations, including the deployment, administration and troubleshooting details during the previous steps.

2.4 Planning

The detailed planning of the internship is as follows:

Stage	Details
Set up the development environment of openATTIC [10/08/18-10/21/18]	* Be familiar with basic knowledge of Ceph and KVM; * Set up the Ceph Cluster with both bare and virtual machines; * Deploy openATTIC and build a local dev environment; * Contribute to the documentation;
Transplant features and merge into new version of openATTIC [10/22/18-12/23/18]	* Understand the code base structure and be familiar with basic knowledge of AngularJS framework; * Sort out the major different features and make plan for the work; * Work following the plan: write, test, fix bugs, etc.;
Synthesize the REST APIs for both openATTIC and Ceph Dashboard with providing test codes [12/24/18-02/24/19]	* Dig out all the APIs for both openATTIC and Ceph Dashboard; * Manage and sort out APIs according to different endpoints, and complement each with details; * Reform APIs definitions into OpenAPI specification and distribute results in a website;
Incorporate the deployment, administration and troubleshooting parts of Ceph operations into reference medium [02/25/19-03/29/19]	* Reference existing documentations for different parts; * Combine previous experiences and take merge; * Contribute to the documentation;

Figure 6: Planning for the work

3 Methodology

3.1 Difficulties

The difficulties possibly encountered during the conduct of this internship come from different places:

- First of all is my personal **limited experience of involving in a large open source project**, where we need to co-work with different people from different countries. How to commit your contribution and cooperate with the development of others is the first thing to learn, and even this question is well explained in the documentation of project, it will still be very different when doing in the reality and you are sure to have problems. Also, trying to avoid asking stupid questions and not hesitating to ask stupid questions seem to be contradictory but in fact are the same thing in the end. All these principles of involving in a large open source project will be completely different from your experience of working on a small project with your well-knowing friends in the school.

- Ceph is a (super) large project with hundreds of contributors and almost hundreds of thousands of commitment already in Github, and even for a single openATTIC source code release, there are over 487 directories and 1486 files. There does not exist a well documented introduction to the project architecture, which results in a situation that **such a large source code base will be difficult to learn and understand from the beginning even for an experienced programmer**, and not to say the possible inconsistent quality of codes and comments.

- Even though the intention of the Ceph Dashboard module is to reuse as much of the existing openATTIC functionality as possible, **these two are still different projects and therefore have different source codes**. For example, while openATTIC is based on Django and the Django REST Framework, the dashboard module's backend code uses the CherryPy framework and a custom REST API implementation. As a result, you can repeat doing the same thing for the two projects but you also need to pay attention to the difference between them, in some places such as API definitions, special syntax of different python frameworks, etc.

- Troubleshooting is hard because **problems may differ in different testing environment**, nobody can be sure about the right answer, and this becomes more difficult especially for those uncommon problems, for which you couldn't find previous cases for reference.

3.2 Strategies

The previous *Difficulties* part is explained at great length, but this does not mean that we have no solutions[2]. I will now introduce the major strategies used in the different subtasks described above.

Virtualization

The minimal cluster configuration for SUSE Enterprise Storage is to have four object storage nodes, with each 4 GB RAM, four cores, and 1 TB capacity. And a network of 10 GB Ethernet is also recommended (You can check the official documentation for full details on system and hardware specifications). Such a configuration could be too exigent because of limited number of servers, and the hardware resources are also wasted for just building a testing and developing environment.

Virtualization is the elegant way to solve this problem. In computing, virtualization refers to the act of creating a virtual (rather than actual) version of something, including virtual computer hardware platforms, storage devices, and computer network resources[11]. We can therefore use the virtualization technology to create multiple isolated virtual machines from just one single host machine, which makes the hardware resources fully used.

There are various choices of virtualization platforms (see: Comparison of platform virtualization software from wikipedia) in the market: Xen, KVM, Virtualbox, Hyper-V, etc. And in the end, the support for Linux OS, the built-in feature in Linux kernel and the reduced hardware cost compared to full virtualization technology such as virtualbox make me pick KVM as the choice for the work.

KVM[12] (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko. Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc.

And together with *virt-manager* application, which is a desktop user interface for managing virtual machines through *libvirt*, we can now build cluster environment by just creating one single virtual machine and using *clone* feature of *libvirt* to make multiple virtual machines in one second.

[2] In fact, for lots of time, you don't need to reinvent the wheels because there is very possibly already someone, who had the same question, found the solution and shared it on the Internet with kind heart, especially the open source spirit becomes now more and more popular.

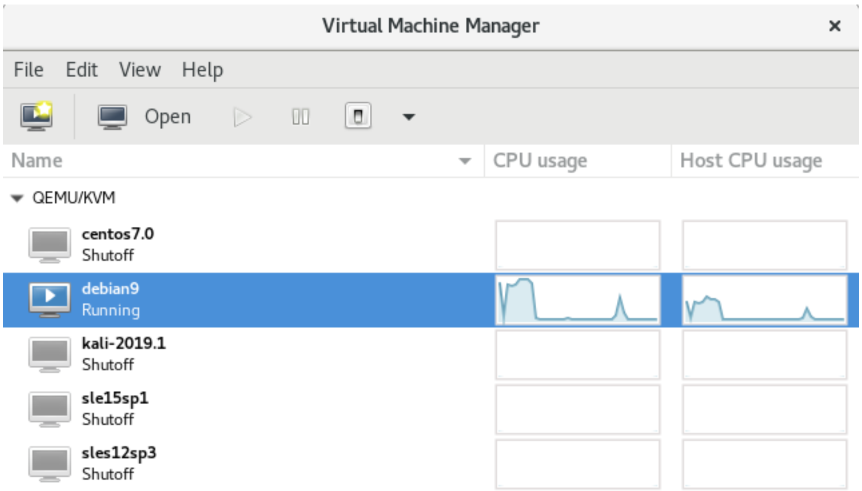

Figure 7: *virt-manager* application main window

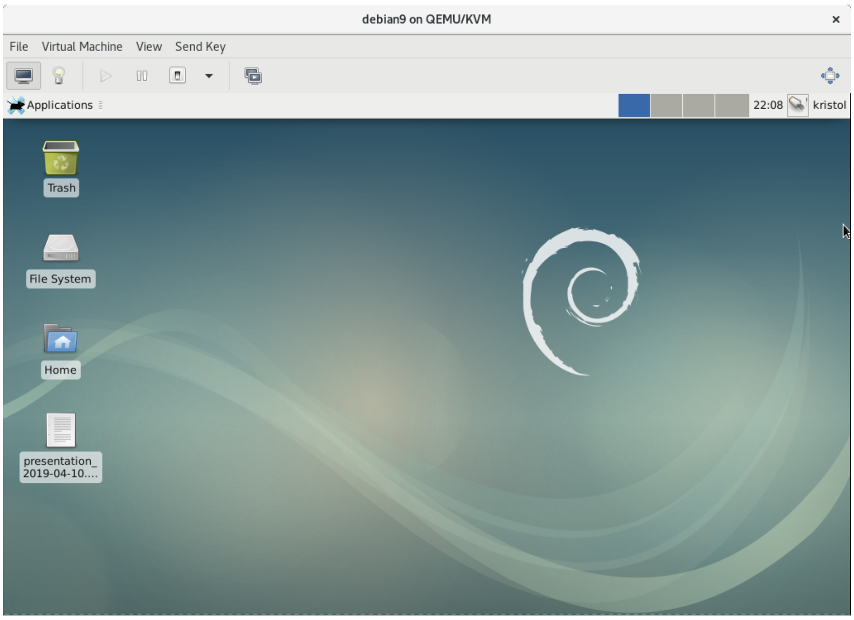

Figure 8: Virtual machine example - Debian9

Another type of virtualization used in the internship is containerization, or more specifically, Docker. Containerization is a lightweight alternative to full machine virtualization that involves encapsulating an application in a container with its own operating environment. This provides many of the benefits of loading an application onto a virtual machine, as the application can be run on any suitable physical machine without any worries about dependencies.

Since the Ceph project team has provided bash scipts to simply create a functional test cluster, we can then write a sigle Dockerfile including those accessible scripts for automating the building of a cluster. The idea is already realized by my SUSE colleague and the final Dockerfile is shared below:

```
FROM opensuse/tumbleweed
LABEL maintainer="rimarques@suse.com"

RUN zypper --gpg-auto-import-keys ref
RUN zypper -n dup
RUN zypper -n install \
        iproute2 net-tools-deprecated zsh lttng-ust-devel babeltrace-devel \
        bash vim tmux git aaa_base ccache wget jq google-opensans-fonts psmisc \
        python python3-pip \
        python-devel python3-devel \
        python3-bcrypt \
        python3-CherryPy \
        python3-Cython \
        python3-Jinja2 \
        python3-pecan \
        python3-PrettyTable \
        python3-PyJWT \
        python3-pylint \
        python3-pyOpenSSL \
        python3-requests \
        python3-Routes \
        python3-Werkzeug

# temporary fix for error regarding version of tempora
RUN pip3 install tempora==1.8 backports.functools_lru_cache

ADD /shared/docker/ /docker

# Chrome
RUN /docker/install-chrome.sh
ENV CHROME_BIN /usr/bin/google-chrome

# oh-my-zsh
ENV ZSH_DISABLE_COMPFIX true
RUN /docker/install-omz.sh

ENV CEPH_ROOT /ceph
ENV BUILD_DIR /ceph/build

VOLUME ["/ceph"]
VOLUME ["/shared"]

CMD ["zsh"]
```

Figure 9: Dockerfile for the automation of building a functional test Ceph cluster

Version Merging

Like said in the previous section, the openATTIC project has a large code base, including hundreds of directories and thousands of files. Merging directly two version releases, between which there are thousands of files changed and over 200 different commits, is impossible in limited time.

Luckily, modern version management uses mainly the git tool, and so as the openATTIC project. You can use git to compare two different code base to see the differences with just one single command *"git diff"*. But as a command line tool, and its *diff* functionality does not support GUI. Here is the *diff* result of two branches using commands:

```
@@ -164,6 +165,23 @@ module.exports = (function makeWebpackConfig () {
           plugins: [autoprefixer]
         }
       }
+    }),
+
+    /**
+     * Internationalization
+     * Reference: https://github.com/augusto-altman/angular-gettext-plugin
+     *
+     */
+    new AngularGetTextPlugin({
+      compileTranslations: {
+        input: 'locale/*.po',
+        outputFolder: 'locale',
+        format: 'json'
+      },
+      extractStrings: {
+        input: 'app/**/*.html',
+        destination: 'locale/template.pot'
+      }
     })
   ];

@@ -274,7 +292,7 @@ module.exports = (function makeWebpackConfig () {
     proxy: {}
   };
   config.devServer.proxy[contextRoot + "api"] = {
-    target: apiConfig.target || "http://192.168.100.200",
+    target: apiConfig.target || "http://192.168.49.70",
     secure: false
   }

diff --git a/webui/webpack.config.json.sample b/webui/webpack.config.json.sample
old mode 100755
new mode 100644
lines 4264210-4264245/4264245 (END)
```

Figure 10: *diff* result of two versions/branches of openATTIC

You can see that 4264245 lines are presented in just a single *diff* file, it's impossible to classify the differences with respect to file or directory, and you even couldn't view all the differences in just one screen! Even if you could separate the whole code base and check differences, file by file, using *git diff*, you still need to input manually the same command thousands of times, again and again, obviously a huge nightmare for a developer.

GitLens comes here to help, which is a plugin in the Visual Studio Code[3] platform. According to its introduction:

> GitLens supercharges the Git capabilities built into Visual Studio Code. It helps you to visualize code authorship at a glance via Git blame annotations and code lens, seamlessly navigate and explore Git repositories, gain valuable insights via powerful comparison commands, and so much more.

One of its features provided is the *Compare view* feature to help visualize comparisons between branches, tags, commits, and more.

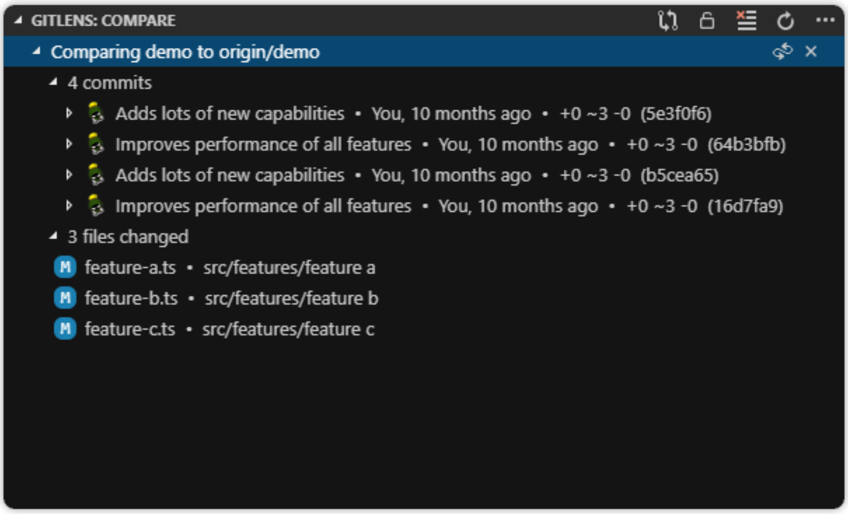

Figure 11: GitLens *Compare View* functionality

Now, with the help of this tool, we could now, first of all, create a new branch from one version release of source code in another version release. And then using this *Compare view* feature to check the differences with respect to file and continue to work finally.

[3]Visual Studio Code is a source-code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. (*Source*: Wikipedia.org)

API Specification

A “web service” is a web-based application that provides resources in a format consumable by other computers. Web services include various types of APIs, including both REST and SOAP APIs. Web services are basically request-and-response interactions between clients and servers (a computer requests a resource, and the web service responds to the request).

Both openATTIC and Ceph Dashboard are just such a web service that provides REST APIs for clients to send different requests or commands to the server for responses. But REST is an architectural style, not a standard protocol. This is why REST APIs are sometimes called RESTful APIs — REST is a general style that the API follows. A RESTful API might not follow all of the official characteristics of REST as outlined by Dr. Roy Fielding, who first described the model. Hence these APIs are “RESTful” or “REST-like”, and most people just refer to them as “REST APIs” regardless.

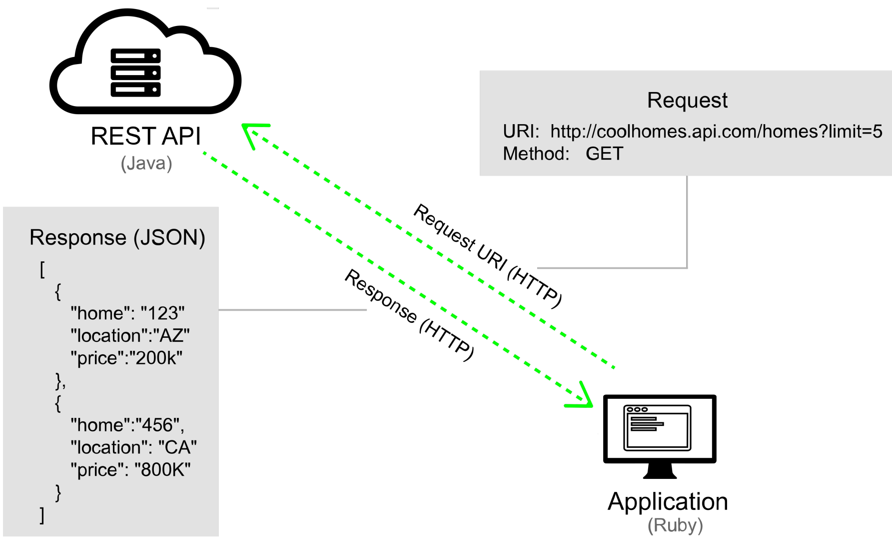

Figure 12: REST API model

Even so REST APIs follow an architectural style, not a specific standard, however, several REST specifications have been developed to try to provide standards in the way that REST APIs are described. The three most popular REST API specifications are as follows: OpenAPI (formally called Swagger), RAML, and API Blueprint [13].

In the early years of specifications, there was healthy competition between the formats. But now, without a doubt, the OpenAPI specification (OAS) is the most popular, with the largest community, momentum, and tooling. Because of this, I choose the OAS as the API describing specification instead of former office word style, which is difficult to use and hard for quick navigation.

The OpenAPI definition is in fact a YAML or JSON file, and you can render it with free tools such as Swagger UI to generate the documentation. Here is the comparison between original word style and the rendered openAPI specification style:

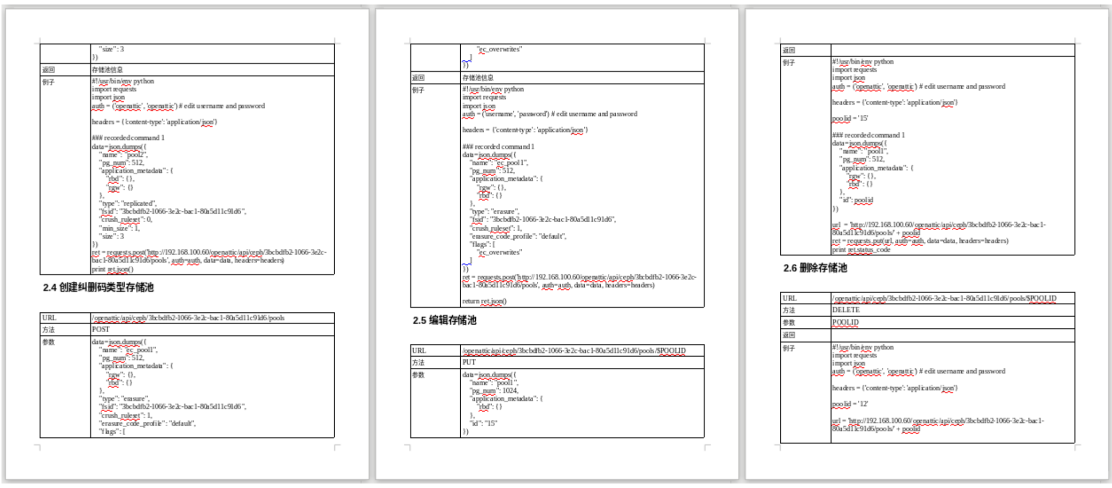

Figure 13: API definitions in the Office Word style

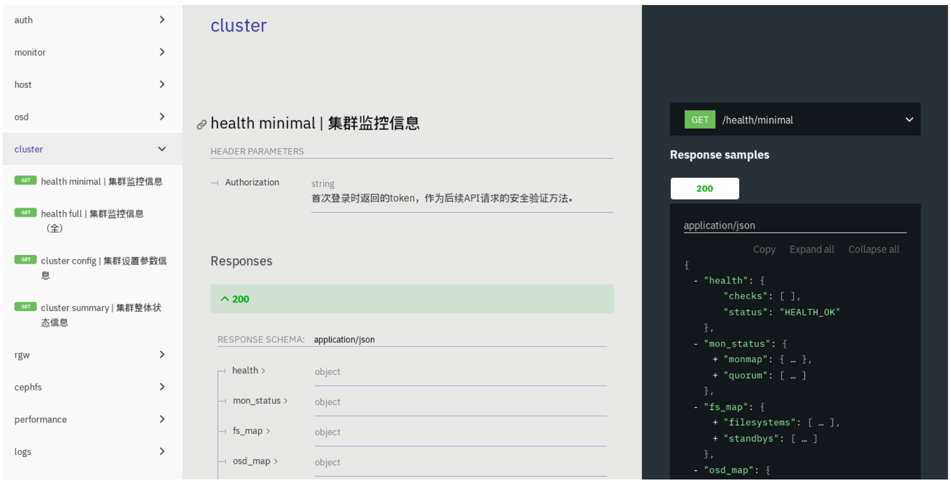

Figure 14: API definitions in the rendered OAS style

4 Results

4.1 Code Merging for openATTIC

Prerequisites

The first step for the code merging is setting up the development environment, so the result of first stage in the planning (see Figure 6) is also included here.

Only three bare servers are available for this specific work, but with the help of virtualization technology, I was able to create 5 and more virtual machines in one single server using KVM/QEMU and the *clone* feature of *virt-manager*. And the virtual machines are listed as below:

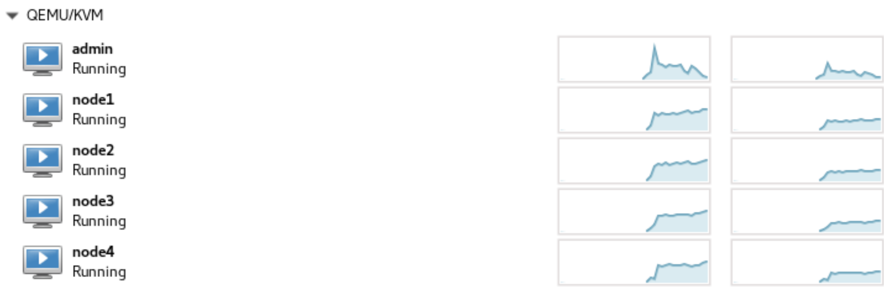

Figure 15: List of virtual machines

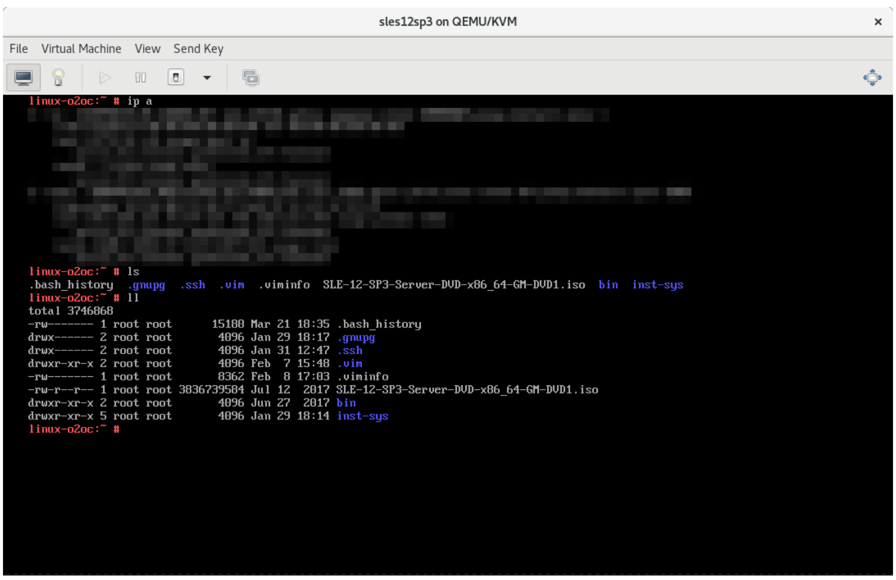

Figure 16: One of the virtual machine without GUI

Since the demand of operation system is different for openATTIC and Ceph dashboard, these 5 nodes are not valuable for the development of Ceph dashboard, so another five virtual nodes with different OS are created, in the same host server. Since the hardware performance of server is sufficient, all these ten nodes can run normally without any problems of delay.

After this step, we need to build the Ceph cluster and deploy openATTIC service. The detailed deployment steps are much complicated, and the final documentation is shared in my Github account (see Figure 17), and you can check that with this link.

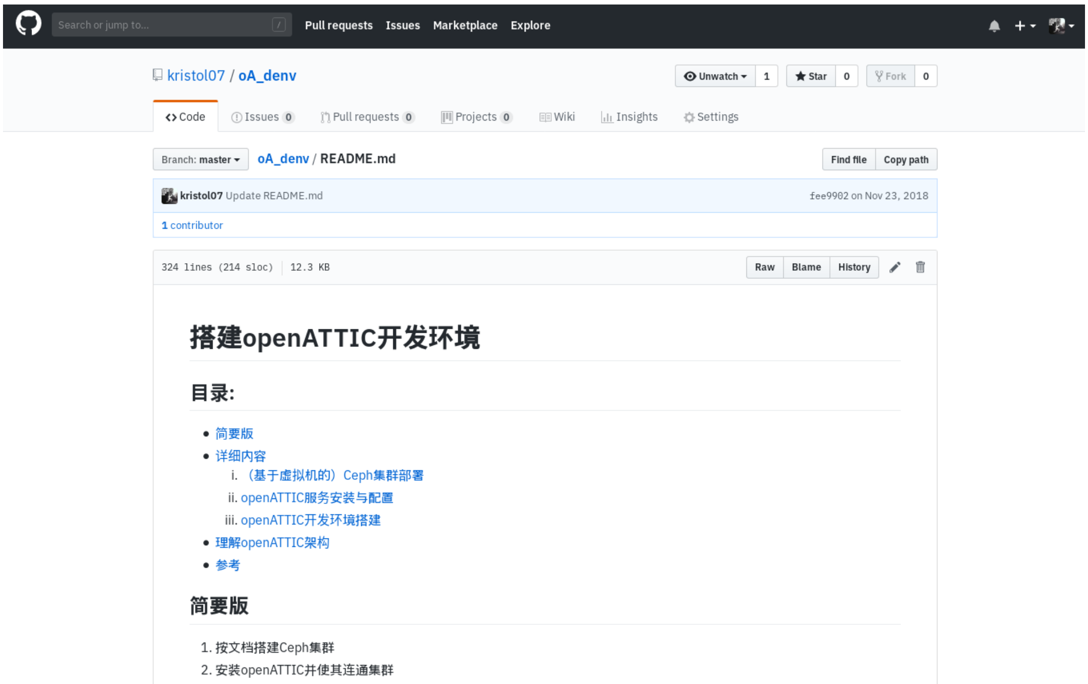

Figure 17: Documentation for building openATTIC developemnt environment

And after the deployment, we can see the final openATTIC WebUI in Figure 18, and continue the code merging work.

Current Situation

A code base of openATTIC with 4032 directories and 25520 files is given like Figure 19, and there are only two commits: first commit and a commit of removing useless files. So basically you don't have any useful information of this version of openATTIC with only knowledge that there are some features not available in the newest version, and the target is to transplant those features into the newest one.

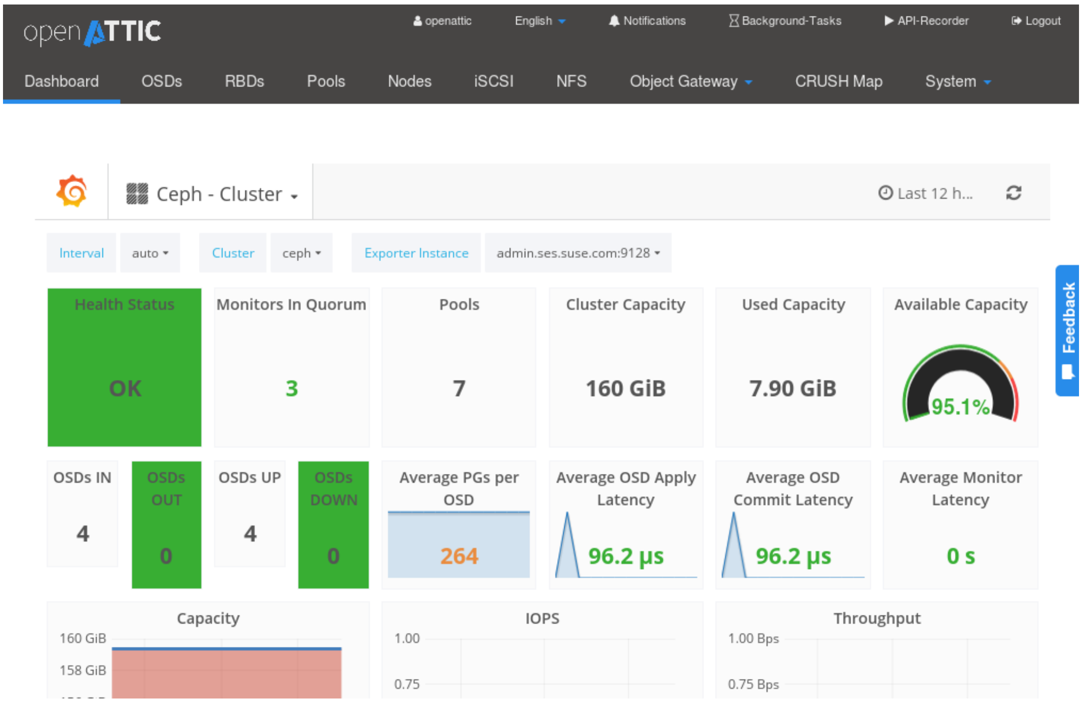

Figure 18: openATTIC WebUI

```
4032 directories, 25520 files
➜ openattic-FH git:(master) ✗ git log | cat
commit d7594993da4dc50fe26924cb8b072699055c92bb
Author: Pei Feng Lin <linpeifeng@gmail.com>
Date:   Wed Aug 15 11:55:42 2018 +0800

    remove orig file

commit dfbef71a68a0a67a66a23136eb817bc261e1d83e
Author: Pei Feng Lin <linpeifeng@gmail.com>
Date:   Tue Aug 14 14:47:31 2018 +0800

    first commit
```

Figure 19: Overview of openATTIC original code base

The following features are found in the WebUI of this version release by comparing to that of the newest version:

1. Chinese translation is updated and is hard-coded in the source code;
2. A *Alerts* tab/feature is added in the navigation bar;
3. User permission control functionality;
4. OSD items are listed in the CRUSH tree map;

5. When chosen, user details are shown;
6. When chosen, OSD details are shown;
7. You can check logging info in the WebUI;
8. You can check iSCSI info in the WebUI;
9. You can check RBD QoS info in the WebUI;

Results Distribution

With the help of the GitLens plugin in VS code, 12 commits are made with each for an added different feature or fixed bugs, see the Figure 20:

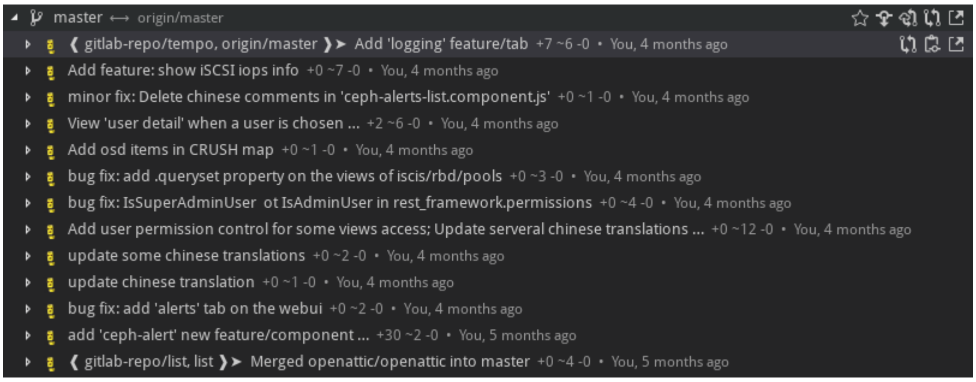

Figure 20: Commits made for different features and fixed bugs

Also, a private Gitlab server is built, and all the codes are pushed into a gitlab repository, see the Figure 21.

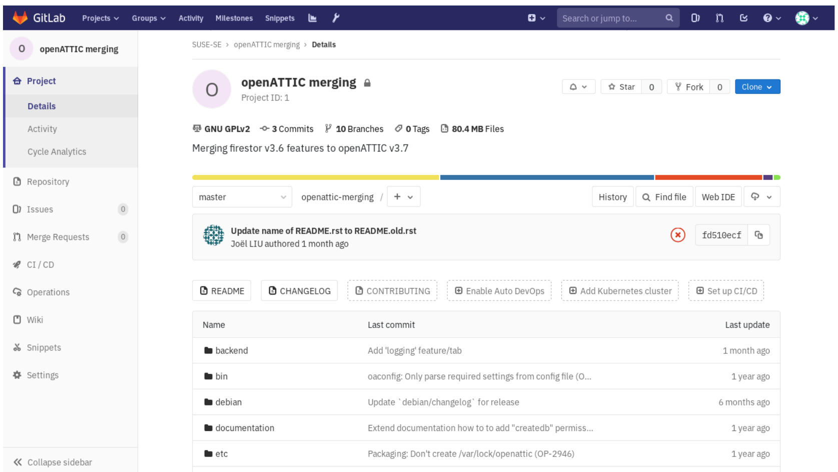

Figure 21: Code of result pushed to remote self-built Gitlab server

Problems Found

Even so most of features are merged into the newer version, there are still some problems worth to be noticed.

For example, for the feature 6, there is always a compiling error of front-end even so the codes are checked carefully, we can still not view the OSD details when one is chosen. And finally after the stage of API work, the reason is found that the related API is not correctly the same as before, and we need to update that for fixing the bug.

Another problem is that original code base uses an older version of *restframework*, which makes the merging can not be as easy as before, because some python modules such as *adminuser* module have changed name in a newer version *restframework*.

Finding these tiny bugs takes lots of time, and can often be very tricky, but it's worthwhile for the completion of the task.

4.2 Ceph dashboard/openATTIC API Synthesis

Even though the WebUI has a *button* for querying API, but it's still a next-coming and unfinished feature. While there is a great demand for APIs to make secondary develop easier for developers, it's then necessary to complete an API documentation for make this happen. And this is different thing for openATTIC and Ceph dashboard, because the new Ceph dashboard module or the beta program SES 6 has changed from the openATTIC, so as the APIs.

According to the introduction in the subsection *API Specification*, (REST) API is what openATTIC or Ceph dashboard provides for clients. When you make a modification to, for example, the name of a user, and click on the *OK* button, the actual action happening under the WebUI is that an http request is sent to the server and WebUI is responsible for showing you the response it got from the server. Therefore, the simplest and common way to find all the APIs is using the built-in *Web Developer* functionality in your web browser (mine is Firefox), and for each click on the UI, check for if there is a special request corresponding to an API we want.

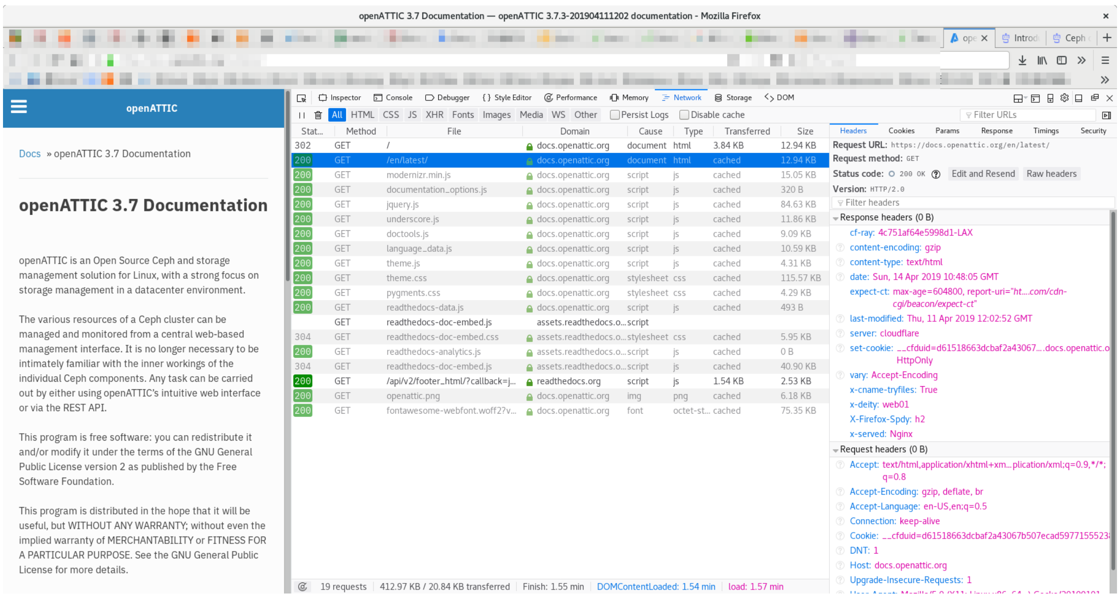

Figure 22: Built-in web browser tool for finding APIs

When an API is found, record the following key values and finally sort them into several categories based on their endpoints:

- request URL

- request method
- special request headers
- important parameters sent
- status code of response
- response headers
- possible response payload
- examples

Table 1 and Table 2 are given as follows for openATTIC APIs and Ceph dashboard APIs respectively.

user	**snapshot**	**nfs**	**iscsi**
list users create user list user update user delete user	list snapshots create snapshot protect snapshot delete snapshot	list nfs exports create nfs export list nfs export delete nfs export update nfs	list iscsi target update iscsi target delete iscsi target create iscsi target stop iscsi target list iscsi targets start iscsi target
rbd	**pool**	**auth**	**osd**
list rbd update rbd delete rbd list rbds create rbd ← snapshot copy rbd image rollback rbd	create pool update pool delete pool list pools	login logout	list osds

Table 1: API lists (35) for openATTIC

Also the respective JSON format files are shared with links opeATTIC API OAS specification file and Ceph dashboard API OAS specification[4] file at my Github account.

[4]Check for openAPI specification details at https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md.

logs	**monitor**	**host**	**iscsi**
logs all	monitors info	hosts info	iscsi info
osd	**cluster**	**cephfs**	**role**
osds info	health minimal	cephfs info	role info
osd detail	health full	cephfs detail	role create
osd scrub	cluster config	cephfs clients info	role delete
	cluster summary	cephfs counter	role update
rbd	**snapshot**	**pool**	**user**
images info	snapshot create	pools info	user info
image create	snapshot update	pool create	user create
image trash info	image ← snapshot copy	pool detail	user update
image delete	image ← snapshot clone	pool update	user detail
image move to trash	image flatten	pool update	user delete
image trash restore	image rollback ← snapshot	pool delete	
rbd mirror info			
rgw		**performance**	**auth**
rgw status	rgw daemon info	performance info	login
rgw daemon detail	rgw user list		logout
rgw user detail	rgw user create	**others**	
rgw user quota	rgw user delete	erasure code profile	
rgw bucket create	rgw bucket list		
rgw bucket delete	rgw bucket detail		
rgw bucket change owner			

Table 2: API lists (60) for Ceph dashboard

Result Distribution

Based on the two specification files, we can render documentation with different generators which follow OpenAPI specification, such as Swagger UI[5] or ReDoc[6]. Because of the simplicity and its complete functionalities of designing, debugging, testing and publishing, I chose the Stoplight.io as the platform for the documentation.

You can check the website SES REST API documentation and the website Ceph dashboard API for the details. A screenshot of the latter is given as Figure 23.

Besides, for the Ceph dashboard, additional support is provided for how to test

[5]Swagger UI, see https://petstore.swagger.io/.
[6]ReDoc, see https://rebilly.github.io/ReDoc/.

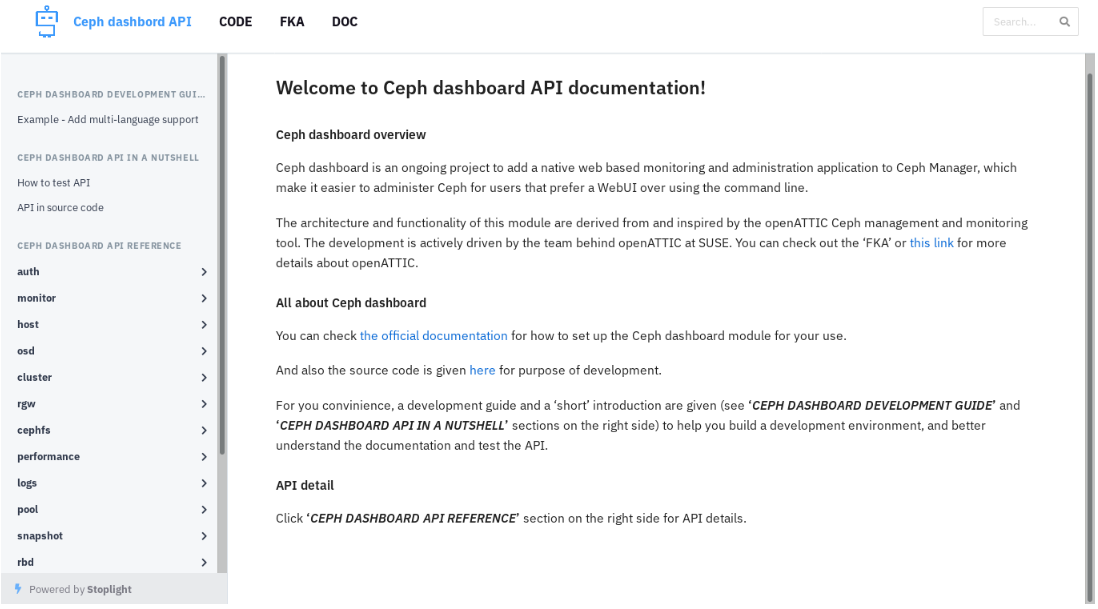

Figure 23: Ceph dashboard API website screenshot

API, how to add language support for Ceph project, and how to build development environment[7].

[7] Notice that this is for Ceph dashboard, previous section 4.1 - *Prerequisites* is for openATTIC.

4.3 Ceph/SES Operations Reference "Book"

Ceph is 'clever' enough for self recovering from mistakes in many situations, while on the other side, the skill for Ceph cluster management is very worthwhile to master just because when your cluster has problems, that usually means you have real problems.

And the situation becomes more difficult because Ceph is a large project composed of many elements, it's hard to figure out the root of problem in a short time. Then in this moment, a reference medium for Ceph operations values a lot, and I was asked to edit such a "book" with my previous experiences of working with Ceph, to ease the **deployment**, **administration** and **troubleshooting** in the future.

The final result is shared with link SES DevOps Book, where you can find a file in three different formats: markdown, odt[8] and pdf. The output pdf file has over 140 pages, and the outline is given in Figure 25. While you can also read directly in this Github page, since markdown file can be rendered by Github automatically[9].

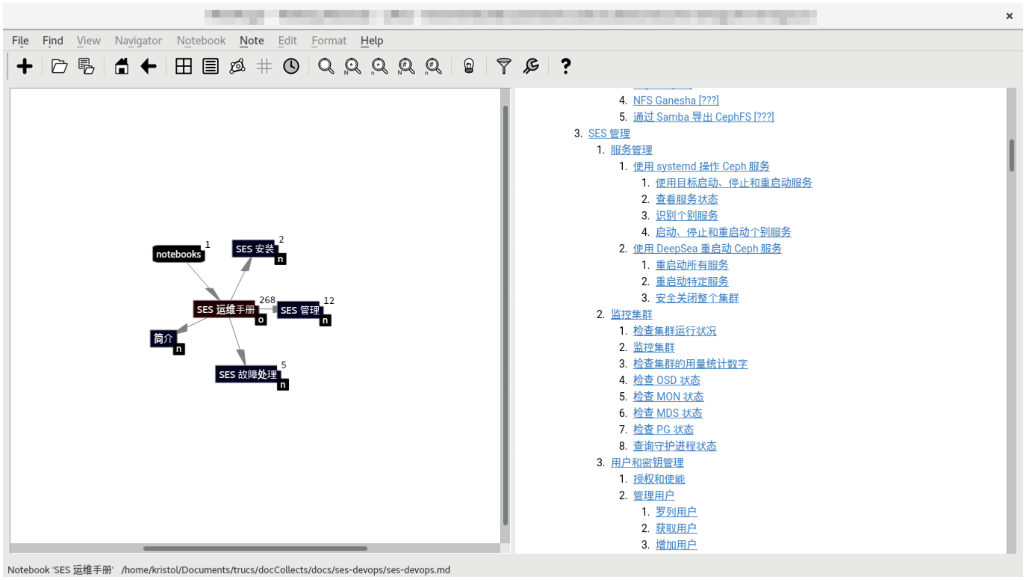

Figure 24: The "book" in knowledge graph view

[8]A file with the .ODT file extension is an OpenDocument Text Document file. These files are most often created by the free OpenOffice Writer word processor program. ODT files are similar to the popular DOCX file format used with Microsoft Word. [14]

[9]But notice that markdown file is not updated as often as the two other formats.

5 Conclusion

This internship is both interesting and challenging, during which I have learned a lot of new things about Linux, open source community, and also various technologies I never had chance to use before. As an intern, I was able to involve in the development and improvement of a product, which not only is sold for business customers, but also evolves quickly with the help of many freewill contributors in the open source community. This experience is unusual and will be valuable for a long time.

Here is a list of some detailed works that I have accomplished during this internship, which also concludes the previous sections:

- Setting up Ceph cluster with both bare and virtual machines, and building the development environments with different virtualization methods for both openATTIC and Ceph dashboard;
- Transplanting features into a version release of openATTIC and make pull requests for real contribution;
- Building a private Gitlab server for holding the code base of mering result, and the future use of team;
- Documenting the steps in detail for how to deploy the newest SES product and how to build the development environment of openATTIC/Ceph dashboard;
- Collecting REST APIs for both openATTIC and Ceph dashboard, with following the OpenAPI Specification (v3);
- Distributing the API results in the way of websites, with additional details on minor questions, such as how to test APIs, how to add language support in the code, etc;
- Contributing to the Chinese language support in Ceph and openATTIC project;
- Editing a Ceph operations reference book with over 140 pages for future convenience;
- Helping with correcting various documentations distributed to customers and technicians, such as the automating installation project for SPD Bank using PXE technology;

I have also strengthened my abilities in this internship, including but not limited to:

- Basic knowledge of distributed storage, API specifications, open source contributing methods, AngularJS framework, Django python framework, and common use of Linux
- Different ways and tools to collect APIs and make documentation from the collection
- Basic Ceph operations and troubleshooting methods
- Use of virtualization technology, especially KVM/QEMU, *libvirt*, and Docker
- Technical writing methods
- Solution-driven thinking
- Teamwork spirit

At last, I would like to thank again both my school ECPKn and SUSE Beijing for allowing me to spend such a wonderful and rewarding internship, which helps me find clearer direction in the future career.

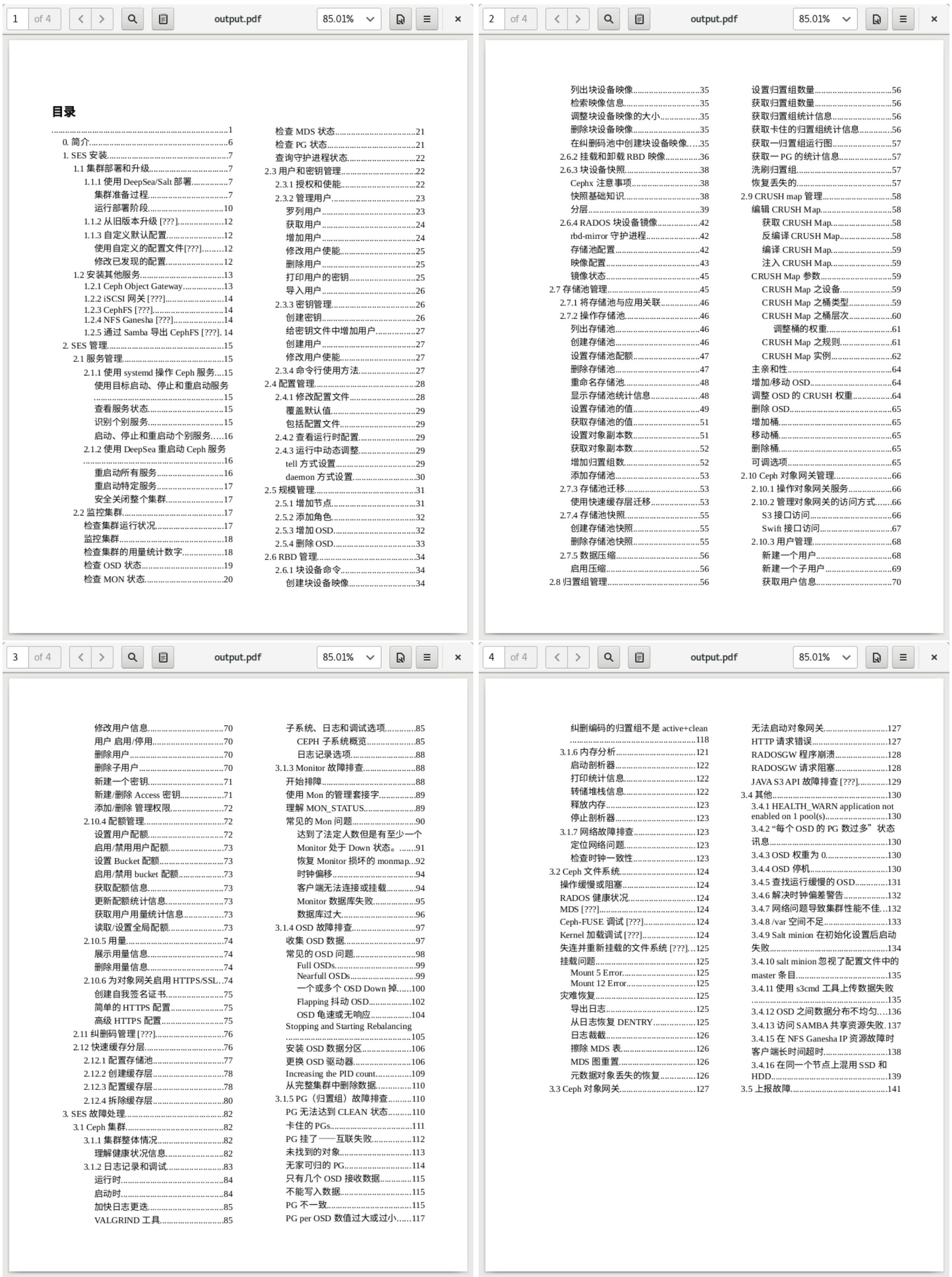

Figure 25: *SES DevOps Book* outline

Bibliography

[1] Wikipedia.org. Suse. https://en.wikipedia.org/wiki/SUSE.

[2] Steven J. Vaughan-Nichols. SUSE acquired from Micro Focus by EQT - The oldest Linux business, SUSE, is changing hands. https://www.zdnet.com/article/suse-acquired-from-micro-focus-by-eqt/, 2018.

[3] SUSE News. Suse completes move to independence, reaffirms commitment to customers, partners and open source communities as industry's largest independent open source company. https://www.suse.com/c/news/suse-completes-move-to-independence-reaffirms-commitment-to-customers-partners-and-open-source-communities-as-industrys-largest-independent-open-source-company/, March 2019.

[4] SUSE Official Website. Storage growth shouldn't hurt. https://www.suse.com/programs/data-explosion/.

[5] SUSE Official Website. Suse enterprise storage. https://www.suse.com/products/suse-enterprise-storage/.

[6] WhatIs by SUSE. What is software-defined infrastructure? https://whatis.suse.com/definition/software-defined-infrastructure/.

[7] WhatIs by SUSE. What is software-defined storage? https://whatis.suse.com/definition/software-defined-storage/.

[8] WhatIs by SUSE. What is ceph? https://whatis.suse.com/definition/ceph/.

[9] openattic.readthedocs.io. openattic documentation. https://openattic.readthedocs.io/en/latest/.

[10] docs.ceph.com. Dashboard plugin - ceph documentation. http://docs.ceph.com/docs/master/mgr/dashboard/.

[11] Wikipedia.org. Virtualization. https://en.wikipedia.org/wiki/Virtualization.

[12] linux kvm.org. Kvm main page. http://www.linux-kvm.org/page/Main_Page.

[13] Tom Johnson. Overview of rest api specification formats. https://idratherbewriting.com/learnapidoc/pubapis_rest_specification_formats.html, 2017.

[14] Tim Fisher. What is an odt file? https://www.lifewire.com/odt-file-4135594, March 2019.