Google Summer of Code 2024: Shaman Scheduling for Success

Devansh Singh

1 Personal Information

- **Email:** devanshamity@gmail.com
- **Slack:** @devansh
- **GitHub:** Devansh3712
- **LinkedIn:** devanshsingh3
- **University:** Jaypee Institute of Information Technology
- **Course:** Bachelor of Technology in Computer Science Engineering
- **Timezone:** Indian Standard Time (GMT +5:30)

Contents

2 Project Information

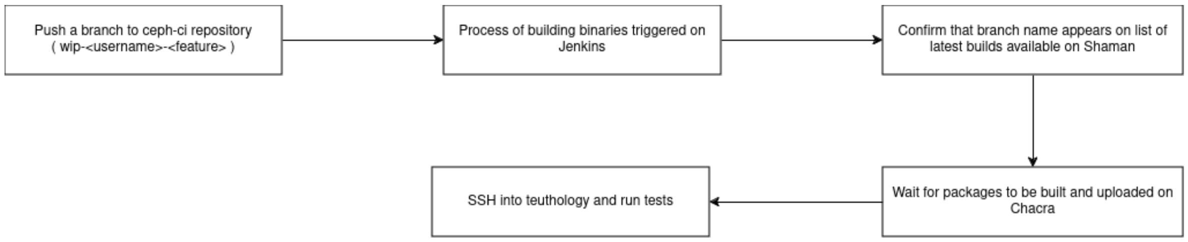
- **Mentors:** Zack Cerza, Aishwarya Mathuria, Kamoltat Sirivadhna, Vallari Agrawal
- **Difficulty**: Medium (350 hours)
- **Goal:** Make process of scheduling Ceph integration tests more efficient by allowing users to auto-schedule teuthology-suite commands when pushing their feature branch to *ceph-ci*

3 Project Proposal

3.1 Current Testing Workflow

In order to schedule a test run, Ceph binaries must be built for the committed branch before one can use teuthology commands to run integration tests on them. The workflow is as follows:

1. Push a branch to the **ceph-ci** repository, which triggers the process of building binaries on the Jenkins CI
    - Branch to be pushed on ceph-ci can be any branch, does not have to be a PR branch
    - Branch should be named in the following syntax: wip-⟨username⟩-⟨feature⟩
2. Ensure build process has been initiated by checking whether the branch name has appeared in the list of "Latest Builds Available" at **Shaman**
3. Wait for the packages to be built and uploaded to Chacra (usually takes between two and three hours depending on availability of machines)
4. SSH into the teuthology machine and run tests using teuthology-suite commands. Wait for the tests to run and the results can be viewed on **Pulpito**

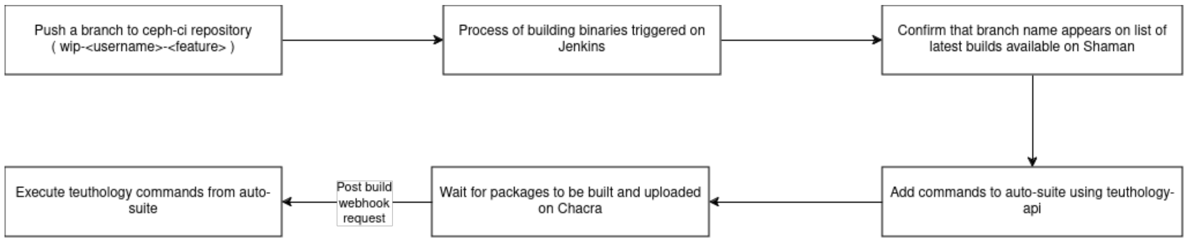

Push a branch to ceph-ci repository (wip-<username>-<feature>)
Process of building binaries triggered on Jenkins
Confirm that branch name appears on list of latest builds available on Shaman
Wait for packages to be built and uploaded on Chacra
SSH into teuthology and run tests

Currently, after pushing a branch to ceph-ci, it takes a few hours for Jenkins to create a build. Once the build is complete, only then the developers are able to schedule teuthology runs.

3.2 Proposed Testing Workflow

The developers should be able to queue teuthology-suite commands before the build is finished. Once the build is completed, the queued commands for a branch would get executed automatically. Modifications in the original workflow are:

1. Once a branch has been pushed to ceph-ci and confirmed that it shows up on Shaman, a developer can schedule test runs using teuthology-api
2. A *submit_final_status* function (**ceph-build/scripts/build_utils.sh**) calls a webhook endpoint after the build is complete
3. **teuthology-api** receives payload from the webhook and executes the queued commands for the branch if the build status is completed


Push a branch to ceph-ci repository (wip-<username>-<feature>)
Process of building binaries triggered on Jenkins
Confirm that branch name appears on list of latest builds available on Shaman
Add commands to auto-suite using teuthology-api
Wait for packages to be built and uploaded on Chacra
Post build webhook request
Execute teuthology commands from auto-suite

3.2.1 Auto-Suite

Auto-Suite will be used to store the commands queued by a developer to be executed after Ceph build is finished.

```
class AutoSuite(BaseModel):
    username: str
    branch: str
    distro: str
    distro_version: str | None
    flavor: str | None
    log_path: str | None
    cmd: SuiteArgs
```

The command to be executed will be stored as a *SuiteArgs* object, which includes all parameters required by the teuthology-suite command. CRUD functionality for Auto-Suite will be added to teuthology-api for managing the queued commands. */auto-schedule* endpoints will require user authorization for access.

Endpoint	Method	Description
/list	GET	Get commands queued by a user
/add	POST	Add a command to be queued
/edit/{id}	PUT	Update a configuration by ID
/delete/{id}	DELETE	Delete a configuration by ID

Table 1: Endpoints for */auto-schedule*

3.2.2 Presets

Presets will allow developers to save teuthology-suite command configuration. They can then reuse commonly used commands without the need to type them out every time.

```
class Preset(BaseModel):
    username: str
    name: str
    cmd: SuiteAgs
```

Similar to *AutoSuite*, the command will be saved as a *SuiteArgs* object. CRUD functionality for Prests will be added to teuthology-api for managing the saved configurations. Each preset of a user will be identified using a unique name. */presets* endpoints will require user authorization for access.

Endpoint	Method	Description
/list	GET	Get all presets of a user
/{name}	GET	Get a preset by its name
/add	POST	Add a preset for a user
/edit/{id}	PUT	Update a configuration by ID
/delete/{id}	DELETE	Delete a configuration by ID

Table 2: Endpoints for */presets*

3.2.3 Webhook

A webhook will be used to trigger the queued teuthology commands for a branch once the Jenkins build is complete. A HTTP POST request is sent using *submit_final_status* function defined in *ceph-build/scripts/build_utils.sh*, which calls the */webhook/build-status* endpoint of teuthology-api. The payload JSON for the POST request will contain the attributes of *BuildStatusWebhook* class.

```
class Status(str, Enum):
    COMPLETED = "completed"
    FAILED = "failed"

class BuildStatusWebhook(BaseModel):
    url: str
    branch: str = Field(..., alias="ref")
    status: Status
    distro: str
    distro_version: str | None
    flavor: str | None
    sha1: str
```

Once the */webhook/build-status* endpoint receives a request, it will fetch queued commands from the *auto-suite* table for that branch and execute them one by one. For future work, multiprocessing or asynchronous programming can be used to achieve concurrency while executing teuthology commands.

3.3 Time Commitment

I will be able to devote 35-40 hours per week during my semester break (June - July), and 20-25 hours otherwise, while I am on campus.

Here is my detailed schedule of my current semester. It is only during my end-semester examinations (10 to 21 May) that my time commitment towards this project will lessen. Barring this, I will ensure to have no commitments other than Google Summer of Code during my summer break. I have not applied for any organization other than Ceph, nor is it my intention to.

3.4 Deliverables

By the end of the project, the following objectives will be achieved:

- Endpoints and table schemas for *AutoSuite* and *Preset*
- Add a post-build action to *ceph-build/scripts/build_utils.sh* and configure webhook endpoint in teuthology-api
- Write unit-tests and documentation for API endpoints
- (Optional) Run queued teuthology runs as a background task

Thus, a working implementation of AutoSuite and Presets will be delivered.

4 Timeline

- **Before May 1**
  - Communicate with mentors to understand details and the goal of the project
  - Go through the resources provided by mentors
  - Join weekly teuthology meetings to discuss about implementation of proposal
  - Complete the **GSoC task**
  - Familiarize myself with Alembic (database migrations tool) and Jenkins
  - Set up development environment and get Sepia lab access for running OpenVPN
- **May 1 - May 26**
  - Go through the current implementation of AutoSuite and Preset in **ceph/teuthology-api/#24**
  - Get in touch with the mentors to discuss how I intend to approach the problem statement
  - Build a ceph-ci branch to understand working of ceph-build better
- **May 27 - July 12**
  - Set up post-build action on Jenkins CI and implement endpoint for receiving webhook payload
  - Create database and table schema for AutoSuite, SQLAlchemy model and functions for CRUD
  - Set up Alembic for database migrations
  - Work on the */auto-schedule* endpoints (refer to Table 1)
  - Write unit-tests for new endpoints using *pytest*
  - Prepare for mid-term evaluation
- **July 12 - August 19**
  - Create table schema for Presets, SQLAlchemy model and functions for CRUD
  - Work on the */presets* endpoints (refer to Table 2)
  - Write unit-tests for new endpoints using *pytest*
  - Fix bugs and improve exception handling
  - Update docker-compose configuration for database
  - Test AutoSuite and Preset for ceph-ci branch builds
- **August 19 - August 26**
  - Buffer week for any unpredictable problems
  - Code style and quality review, add documentation for developers
  - Prepare for final evaluation
  - Deliver working implementation of AutoSuite and Presets

5 About Me

I am a third-year computer science undergrad and a self-taught backend developer. My particular fields of interest are web servers, cryptography, blockchain, and quantum computing. I have also worked extensively as a freelancer and a backend intern for a couple of Indian startups. I have been a part of Major League Hacking's prep fellowship, and contributed to Ceph in Google Summer of Code 2023.

In my free time, I like to work on open-source projects and read about new technologies. Some technologies I use daily include Python, Go, Git & GitHub, Docker, Continuous Integration (GitHub Actions), Relational Databases (MySQL, PostgreSQL), and NoSQL Databases (MongoDB, Redis), etc.

Some of my Python projects are:

- **nasake:** Web app for different mental health resources at one place (FastAPI, SQLAlchemy, PostgreSQL)
- **tsuki:** Minimalistic social media application (FastAPI, PostgreSQL)
- **torweather:** Email notification system for TOR relay nodes (Flask, MongoDB)
- **cmc-py:** Unofficial CoinMarketCap API and wrapper (Selenium, FastAPI, Redis)

My resume can be found *here*.

6 Why Me?

I love to work on web servers and APIs. Python is my primary language of development, and I have five years worth of experience with it. I am proficient with FastAPI and have gained indispensable industrial experience from working in fast-paced environments with Indian startups. I also contributed to Ceph under Google Summer of Code 2023 and thus have gained familiarity with the code base of teuthology-api. I firmly believe my technology stack, coupled with my skills, experience, passion and willingness to learn and contribute to the field make me a viable candidate. If selected, I am sure that Google Summer of Code will provide me with unparalleled exposure and aid my developing skill set greatly.

Following are previous PRs that I have worked on for teuthology-api:

- **ceph/teuthology-api#4:** Update from pydantic to pydantic-settings for handling environment variables in config file
- **ceph/teuthology-api#16:** Update pydantic dict() to model_dump()
- **ceph/teuthology-api#50:** config_yaml parameter contains YAML content instead of file path
- **ceph/teuthology-api#52:** Fix typos in SuiteArg aliases
- **ceph/teuthology-api#53:** Add */suite/default* endpoint for returning default request body

The functioning of open-source appeals to me greatly, and participating in it has intruded me to emerging technologies. It has drastically improved my problem-solving and programming skills. Having been a part of Ceph since last year, I am sure I will be able to contribute effectively to the community and subsequent projects.

References

[1] https://ceph.io/en/developers/google-summer-of-code/#shaman-scheduling-for-success

[2] https://github.com/ceph/ceph-build

[3] https://github.com/ceph/teuthology-api/pull/24

[4] https://docs.ceph.com/projects/teuthology/en/latest/commands/teuthology-suite.html

[5] https://www.youtube.com/watch?v=MSrZvnXVrFI

[6] https://docs.ceph.com/en/latest/dev/developer_guide/testing_integration_tests/tests-integration-testing-teuthology-workflow

[7] https://github.com/VallariAg/ceph-build/commit/217f080a45c00a07829be9c0ce51057f23b27ddc