MacPorts Custom Views Plugin for Buildbot

Personal Details

- Name: Rajdeep Bharati
- Email: rajdeepbharati13@gmail.com
- GitHub: https://github.com/rajdeepbharati
- Blog: https://medium.com/@rajdeepbharati
- IRC: rajdeepbharati
- University: Jamia Hamdard, New Delhi
- Major: Computer Science & Engineering (2nd year)
- Location/Timezone: India (UTC + 5:30)

Abstract

MacPorts currently uses a legacy version of Buildbot (0.8) as its continuous integration framework and hasn't upgraded due to certain drawbacks in the Waterfall view of the newer versions. However, the currently deployed version is outdated and fails in several aspects due to the absence of some key features such as:

i. The ability to write custom JavaScript views (UI components).
ii. Triggering a build whenever a pull request is made. Currently, a build can be started only when patches are committed to the master branch.

This has led to some major setbacks with respect to developer productivity. MacPorts also needs some custom views in buildbot to be able to better analyze build history, commits, etc. The legacy version doesn't allow us to write such custom views. This project will involve upgrading the Macports Buildbot infrastructure to the latest version, developing a plugin for buildbot and writing custom views.

Goals

- Develop a plugin for Buildbot that would allow developers to write custom components using modern frameworks like Vue/React, instead of AngularJS.
- Write custom components in Vue.js:
    i. List of commits with links to builds
    ii. Overview of each commit/forced build, and a tabular representation of build status (fail, pass, etc) on different versions of ports.
    iii. Overview of the history of builds of a particular port

iv. A view where the user would be able to filter the builds by portname, maintainer, author, and branch.

- Migrate existing MacPorts Buildbot setup to version 2.1 and test new components.
- Convert the core plugin logic into an npm package.
- Set up “disposable builds”, i.e., builds on every pull request to start CI tests whenever a PR is made to a repository.
- Configure buildmaster to send email notifications to the port maintainer(s), committer(s), and author(s) about build status (started, failed, passed).

Wireframe of Views

View #1: List of commits

There will be a widget to let the user select a timestamp, and an option to display all the commits before or after the selected date. I have implemented a basic prototype.

According to the timestamp, a table will be dynamically populated with builds in inverse chronological order. By default, all the commits before the current date will be displayed (if the user doesn’t select any timestamp):

Build status (failure/success)	Timestamp	Commit shortlog	Committer	Author	Link to view #2
✅	2019-03-23 22:34:36	py-pyscard: update to 1.9.8	user1	user2	commit A
❌	2019-03-04 02:15:00	samp: new port	user3	user3	commit B

View #2: Overview of a particular commit/forced build

Complete info about the commit:

1. Committer name & timestamp
2. Author name & timestamp
3. Subject & full commit message
4. URL to commit on GitHub
5. Potential URLs to track tickets & PRs

Summary of build/commit tests on each port:

MacOS version ➡️	10.13	10.12	10.11	10.10	10.9	10.8
Port List ⬇️	OK	OK	OK	OK	OK	OK
port A	OK	OK	FAIL	OK	FAIL	FAIL
port B	FAIL	FAIL	FAIL	SKIP	SKIP	SKIP
port C	SKIP	OK	OK	FAILDEP	FAILDEP	FAILDEP

View #3: Overview of history of builds of a particular port

Relevant details about the port:

1. Name
2. Current version
3. Latest version
4. Maintainers (thumbnails, with links to their GitHub profiles)

This would be followed by a widget similar to the one in View #1, and a table showing the history of builds in the selected time frame:

Link to view #2	Timestamp	Version	MacOS version					
			10.13	10.12	10.11	10.10	10.9	10.8
commit A		0.9	OK	OK	OK	OK	FAILDEP	SKIP
commit B		0.8	OK	OK	OK	OK	FAILDEP	SKIP
force		0.8			OK			
commit C		0.8	OK	OK	FAIL	FAIL	FAILDEP	SKIP
commit D		0.7	OK	OK	OK	OK	OK	SKIP

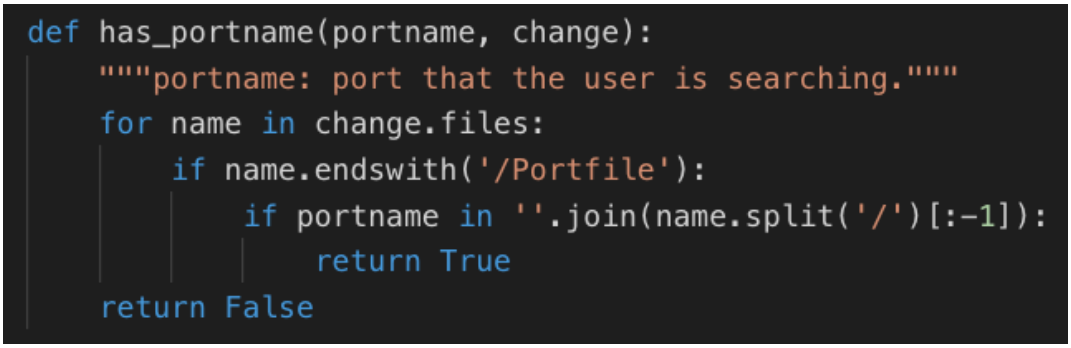
View #4: Filter builds by different parameters

This would contain all the builds sorted by date (using the date widget), with options to filter builds using certain keywords such as portname, maintainer, author, and branch. There would also be an option to show all failed/passed builds of a particular maintainer. This would be configurable in the master.cfg, and new filters may be created as per the needs.

Implementation

- The custom views will be integrated to AngularJS with the help of a plugin based on this boilerplate: https://github.com/uglycoyote/buildbot-react-plugin-boilerplate. A custom directive is used to interconnect the frontend frameworks. It also enables us to get buildbot data from the API and **forward it to Vue as props**.

- View #4 would be implemented using the buildbot *`class`*:
  `buildbot.changes.changes.`**`Change`**
  The portname would be extracted from the `change.files` parameter (which is a list of files that are affected by a change) in the following manner:



```
def has_portname(portname, change):
    """portname: port that the user is searching."""
    for name in change.files:
        if name.endswith('/Portfile'):
            if portname in ''.join(name.split('/')[:-1]):
                return True
    return False
```

  The above method would return True when the portname matches, and the frontend logic would render the builds related to the searched portname.
- The npm package would have an interface to take a component as input, run the logic behind the scenes to connect the component with the AngularJS app. The python setup (that is present in buildbot plugins) would be installed automatically during creation from within the plugin (JavaScript code) using shelljs.

  There would also be a command line application (project generator) similar to `create-react-app` or `vue-cli`, which would create a new plugin app with all the boilerplate code. It would contain sample components written in Vue and React.
  Basic Usage:
  `$ bbview create my_view`
  The above command would generate a boilerplate view having the name `my-view`.
  The developer would have to add `my_view` to the `dict` of plugins in `master.cfg`.
  There would be a development mode (`webpack --watch`) and a production build.

- Disposable builds will be implemented using `GitHubPullrequestPoller`.

This can be done by running the builds on virtual machines with different versions of MacOS (10.14 down to 10.6) with the help of libvirt API. This will prevent malicious pull requests from damaging the builders. This PoC will be referred to while setting up MacOS QEMU/KVM workers on Ubuntu server.

An alternative without using VMs would be to start a build when a PR has been approved by at least *x* developers (where *x* is chosen arbitrarily by MacPorts maintainers).
There would also be another option for maintainers to run a build explicitly by commenting in a GitHub PR.

```
def approve_filter(pr, threshold):
    approves = 0
    for participant in pr['participants']:
        if participant['approved']:
            approves = approves + 1
    if approves < threshold:
        return False
    return True

from buildbot.plugins import changes

c['change_source'] = changes.GitHubPullrequestPoller(
    owner='macports',
    repo='macports-ports',
    token='token',  # GitHub API token
    pullrequest_filter=lambda pr: approve_filter(pr, 3),
    pollInterval=60*5,  # Check PRs every 5 minutes
    pollAtLaunch=True
)
```

In the above example, a pull request is only processed if it has been approved by at least 3 people.

Timeline

I will be needing a break of about 7 days in the month of May for my end semester exams, which I would cover up by chipping in some extra hours daily after the exams.

Legend: Importance & time devoted:

Time Frame	Start Date	End Date	Task(s)	
Initial Phase		May 6	- Familiarize completely with buildbot configuration and JSON API. - Study the current macports buildbot infrastructure. - Write a basic buildmaster config file for MacPorts using the buildbot 2.1 in Python 3. - Learn about portfiles and create a sample portfile for buildbot 2.1.	
Community Bonding	May 6	May 27		
	May 6	May 13	Write the 4 main views (#1, #2, #3, #4) and test them using a “fake builder” (by fetching data from the GitHub API).	
	May 13	May 20	Break for end semester exams.	
	May 20	May 27	Fix issues in collection class in buildbot data to improve compatibility with Vue. Finalize the MacPorts `master.cfg` file. Check the plugin for security vulnerabilities and outdated dependencies; refactor it accordingly.	
Phase 1	May 27	June 24		
	May 27	June 3	Tweak the views #1, #2, #3, #4 and integrate it with the actual buildbot setup and write corresponding unit tests.	
	June 3	June 10	Set up “disposable builds”. Add GitHub authentication to allow developers to log in directly to	

			the buildbot UI, without creating a separate username & password.	
	June 10	June 17	Write documentation.	
	June 17	June 24	Deploy the plugin to a production environment	
End Result			A production-ready plugin with the major functionality deployed. It is provided to MacPorts developers, who could identify flaws, suggest potential improvements.	
Phase 2	June 24	July 22		
	June 24	July 1	Receive feedback from users, fix issues/bugs.	
	July 1	July 8	Add feature to send email notifications to port maintainers, committers, authors about build status (started, failed, passed).	
	July 8	July 15	Optimize the algorithms to render views.	
	July 15	July 22	Create a npm package for the custom views plugin. It will contain the core logic of the plugin, with examples of using it with React and Vue, and documentation.	
End Result			Improved performance and quality of the views. A npm package that can be used by buildbot users who want to use React/Vue to write components.	
Phase 3	July 22	August 19		
	July 22	July 29	Write waterfall view using Vue components.	

	July 29	August 5	Popup notification whenever a new build is complete (passes/fails).	
	August 5	August 12	Document and test the new Waterfall view. Merge features with the buildbot repository.	
	August 12	August 19	Fine tuning of the UI.	

Stretch Goals

- Write a customized waterfall view for MacPorts, similar to the one in buildbot 0.8.
- Set up disposable builds in Virtual Machines, for building PRs.
- Create a docker container to deploy the plugin easily.
- Create a standalone frontend based on a build statistics web API (depends on statistics collection project).
- Write a portfile for the buildbot views plugin.

Availability

I would be able to devote about 50 hours per week to the project. I would also write a weekly blog post highlighting the progress made in my project. I am available full time during summers since I do not have any other internship or job.

About Me

I have been using MacPorts ever since I switched to mac (about 2 years). The main reason why I like it is due to its support for specifying variants during port installation. I have experience with package management systems such as pip and npm. I have some experience with C and would be happy to learn Tcl and start contributing to core MacPorts projects (my project is based on web development and buildbot). I have experience working as a web developer at a startup: https://www.fetchmewishes.com, where I developed a Django app, REST API, integrated payment gateway, built a simple chatbot, and worked on InstaPy bot to increase Instagram followers.
I also love using buildbot (even more than Travis CI) because it's much more hackable and written in Python.
Even after GSoC ends, I would certainly continue to contribute to buildbot and Macports.

I have been actively involved with open source projects such as coala, FOSSAsia, etc. Some of my contributions are:

- https://github.com/coala/coala-bears/pulls?utf8=%E2%9C%93&q=author%3Arajdeepbharati
- https://github.com/coala/coAST/pulls?utf8=%E2%9C%93&q=author%3Arajdeepbharati
- https://github.com/fossasia/labyrinth/pulls?utf8=%E2%9C%93&q=author%3Arajdeepbharati

I can assure you that if I get selected to work with MacPorts this summer, I will give my best to make this project a success, and make the Macports developer workflow more convenient and faster.

Looking forward to working with you.
Rajdeep Bharati