WEST UNIVERSITY OF TIMIȘOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
STUDY PROGRAM: COMPUTER SCIENCE IN ENGLISH

BACHELOR THESIS

COORDINATOR:
Lect.Univ.Dr.Stelian Mihalaș

GRADUATE:
Nagy Gabriel Alexandru

TIMIŞOARA
2018

WEST UNIVERSITY OF TIMIȘOARA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

STUDY PROGRAM: COMPUTER SCIENCE IN ENGLISH

CONTINUOUS INTEGRATION WITH BUILDBOT

COORDINATOR:
Lect.Univ.Dr.Stelian Mihalaș

GRADUATE:
Nagy Gabriel Alexandru

TIMIŞOARA

2018

Abstract

The purpose of this thesis is to shed light on Buildbot as a Continuous Integration tool. Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control. We target to do this by automating most of the delivery process related tasks, freeing the developer to do other software development related tasks.

Using Buildbot, developers cand test their components for possible errors without the risk of committing to version control and breaking the code. Buildbot is an open source project, written in Python on top of the Twisted network programming framework.

In this thesis we will emphasize the extensibility of Buildbot, and how we can use Python and Angular to strengthen the capabilities of this tool, customizing it to best suit our project.

Contents

Chapter 1

Introduction

1.1 Motivation

Generally, every software company, regardless how large or small, uses Continuous Integration principles. Verifying and testing code before delivering and merging with the mainline is of paramount importance, as it saves money as well as time.

While smaller companies might make use of more flexible ways of testing, the larger ones usually run a CI tool such as Jenkins, on which they run builds of every commit that finds its way to the master/trunk branch. This is a widespread and useful practice, albeit narrow, as it only tracks main branches. If a commit happens to break the code, the trunk gets locked and valuable time is spent to fix the regressions. What I will present in this thesis is how developers can use Buildbot as a tool to remotely test their software, without interfering with other people's work. We tell Buildbot the same step sequence the developer uses to test their code, and it does the job for us.

By transferring these tasks to Buildbot, the developer can work on other important issues without worrying about the testing part.

1.2 Contribution

Even though Buildbot is a powerful tool out-of-the-box, it really shines if properly customized and extended. Being an open-source project, it receives frequent contributions from users around the world. The codebase is modular and properly documented, and one can easily begin writing their own extensions and helpers.

Some of Buildbot's default features are:

- run builds on a variety of worker platforms
- arbitrary build process: if you can build it locally, so can Buildbot
- minimal requirements: Python and Twisted
- status delivery through a variety of protocols: web page, email, IRC
- tracks builds in progress, estimates completion time
- flexible configuration by subclassing generic build process classes

- released under the GPL[1] license

The custom implementations that we will go over consist in:

- send an email with the build results to the user using LDAP to search the directory for the user's email address
- parse custom output logs, counting errors and creating a summary of the passed and failed tests
- preferential build steps: do not run this step if the previous failed
- use Flask to deploy a custom web dashboard
- use Angular to deploy an asynchronous, live-reloading, custom web dashboard
- buildbot try script: the developer executes a script in his working directory, and his changes are instantly send to the Buildbot queue to be built
- extend buildbot commands to support additional arguments such as multiple patchfiles

1.3 Structure

Besides presenting the technologies used, before getting into the extensibility of Buildbot, I will also shed light on some of its default implementations to see their purposes.

[1]The **GNU General Public License** (**GPL**), originally written by Richard Stallman of the Free Software Foundation, is a widely used free software license, which guarantees end users the freedom to run, study, share and modify the software.

Chapter 2

Technologies used

2.1 Backend

By definition, Buildbot is a system to automate the compile/test cycle required by most software projects to validate code changes. By automatically rebuilding and testing the tree each time something has changed, build problems are pinpointed quickly, before other developers are inconvenienced by the failure. By running the builds on a variety of platforms, developers who do not have the facilities to test their changes everywhere before checkin will at least know shortly afterwards whether they have broken the build or not. The overall goal is to reduce tree breakage and provide a platform to run tests or code-quality checks that are too annoying or pedantic for any human to waste their time with. Developers get immediate (and potentially public) feedback about their changes, encouraging them to be more careful about testing before checkin.

The Buildbot was written to automate the human process of walking into the office, updating a tree, compiling (and discovering the breakage), finding the developer at fault, and complaining to them about the problem they had introduced. With multiple platforms it was difficult for developers to do the right thing (compile their potential change on all platforms); the buildbot offered a way to help.[10]

2.1.1 Python

The bulk of Buildbot is written in *Python*.

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. It is an interpreted language, meaning that it executes its instructions directly and freely, without previously compiling a program into machine-language instructions. The interpreter executes the program directly, translating each statement into a sequence of one or more subroutines already compiled into machine code.

Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as *C++* or *Java.*[1] *Python* features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.[2] Besides being present in the Buildbot

backend, we will also encounter it when implementing custom web dashboards, and in *Angular* with *CoffeeScript*, which is a language that cross-compiles to *JavaScript* and has Python-inspired syntax.

Since 2003, Python has consistently ranked in the top ten most popular programming languages in the TIOBE Programming Community Index.[1] As of November 2017, it is the fourth most popular language.[3]

An empirical study found that scripting languages, such as Python, are more productive than conventional languages, such as C and Java, for programming problems involving string manipulation and search in a dictionary, and determined that memory consumption was often "better than Java and not much worse than C or C++".[4]

2.1.2 Twisted

The cornerstone of Buildbot is Twisted, an open-source, event-driven network programming framework written in Python and licensed under the MIT License.

Twisted supports many common transport and application layer protocols, including TCP, UDP, SSL/TLS, HTTP, IMAP, SSH, IRC, and FTP. Like the language in which is written, it is "batteries-included"; Twisted comes with client and server implementations for all of its protocols, as well as utilities that make it easy to configure and deploy production-grade Twisted applications from the command line.

Twisted includes both high- and low-level tools for building performant, cross-platform applications. One can deploy a web or mail server with just a few lines of code, or one can write their own protocol from scratch. At every level, Twisted provides a tested, RFC-conforming, extensible API that makes it possible to rapidly develop powerful network software.[5]

2.1.2.1 Protocol/transport separation

Twisted is designed for complete separation between logical protocols (usually relying on stream-based connection semantics, such as HTTP or POP3) and physical transport layers supporting such stream-based semantics (such as files, sockets or SSL libraries). Connection between a logical protocol and a transport layer happens at the last possible moment — just before information is passed into the logical protocol instance. The logical protocol is informed of the transport layer instance, and can use it to send messages back and to check for the peer's identity. Note that it is still possible, in protocol code, to deeply query the transport layer on transport issues (such as checking a client-side SSL certificate). Naturally, such protocol code will fail (raise an exception) if the transport layer does not support such semantics.

2.1.2.2 The Deferred API

Central to the Twisted application model is the concept of a deferred (elsewhere called a future). A deferred is an instance of a class designed to receive and process a result which has not been computed yet, for example because it is based on data from a remote peer. Deferreds can be passed around, just like regular objects, but cannot be asked for their value. Each deferred supports a callback chain. When the deferred

[1] A measure of popularity of programming languages, created and maintained by the TIOBE Company based in Eindhoven, Netherlands. TIOBE stands for "The Importance of Being Earnest", taken from the name of a comedy play written by Oscar Wilde at the end of the nineteenth century.

gets the value, it is passed to the functions on the callback chain, with the result of each callback becoming the input for the next. Deferreds make it possible to arrange to operate on the result of a function call before its value has become available.

For example, if a deferred returns a string from a remote peer containing an IP address in quad format, a callback can be attached to translate it into a 32-bit number. Any user of the deferred can now treat it as a deferred returning a 32-bit number. This, and the related ability to define "errbacks" (callbacks which are called as error handlers), allows code to specify in advance what to do when an asynchronous event occurs, without stopping to wait for the event. In non-event-driven systems, for example using threads, the operating system incurs premature and additional overhead organizing threads each time a blocking call is made.

2.1.2.3 Thread support

Twisted supports an abstraction over raw threads — using a thread as a deferred source. Thus, a deferred is returned immediately, which will receive a value when the thread finishes. Callbacks can be attached which will run in the main thread, thus alleviating the need for complex locking solutions. A prime example of such usage, which comes from Twisted's support libraries, is using this model to call into databases. The database call itself happens on a foreign thread, but the analysis of the result happens in the main thread.[7]

2.1.2.4 Example: A TCP Echo Server and Client

A basic implementation for a simple TCP echo server and client pair can be seen below. The server's job is to listen for TCP connections on a particular port and echo back anything it receives. The client's job is to connect to the server, send it a message, receive a response, and terminate the connection.[6]

```
from twisted.internet import protocol, reactor

class Echo(protocol.Protocol):
    def dataReceived(self, data):
        self.transport.write(data)

class EchoFactory(protocol.Factory):
    def buildProtocol(self, addr):
        return Echo()

reactor.listenTCP(8000, EchoFactory())
reactor.run()
```

Listing 2.1: *echoserver.py*

```
from twisted.internet import reactor, protocol

class EchoClient(protocol.Protocol):
    def connectionMade(self):
        self.transport.write("Hello, world!")
```

```
    def dataReceived(self, data):
        print "Server said:", data
        self.transport.loseConnection()

class EchoFactory(protocol.ClientFactory):
    def buildProtocol(self, addr):
        return EchoClient()

    def clientConnectionFailed(self, connector, reason):
        print "Connection failed."
        reactor.stop()

    def clientConnectionLost(self, connector, reason):
        print "Connection lost."
        reactor.stop()

reactor.connectTCP("localhost", 8000, EchoFactory())
reactor.run()
```

Listing 2.2: *echoclient.py*

To test the scripts, the server should be run in a first terminal using `python echoserver.py`. This will start a TCP server listening for connections on port 8000. The client should be run in a second terminal with `python echoclient.py`.

A sample command sequence would be:

```
$ python echoserver.py # in terminal 1

$ python echoclient.py # in terminal 2
Server said: Hello, world!
Connection lost.
```

Listing 2.3: Command sequence example

2.1.3 Database

As of version 0.8.0, Buildbot has used a database as part of its storage backend. All access to the Buildbot database is mediated by database connector classes. These classes provide a functional, asynchronous interface to other parts of Buildbot, and encapsulate the database-specific details in a single location in the codebase.

The connectors all use *SQLAlchemy Core* to achieve (almost) database-independent operation. Note that the *SQLAlchemy ORM* is not used in Buildbot. Database queries are carried out in threads, and report their results back to the main thread via Twisted Deferreds. The database schema is maintained with *SQLAlchemy-Migrate*. This package handles the details of upgrading users between different schema versions.[11]

In the default configuration Buildbot uses a file-based *SQLite* database, stored in the `state.sqlite` file of the master's base directory.[12]

SQLite is a relational database management system (DBMS) contained in a C programming library. In contrast to many other database management systems, SQLite

is not a client–server database engine. Rather, it is embedded into the end program. SQLite is ACID-compliant[2] and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.[8] SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages, including Python. Among its most popular users are browsers like *Google Chrome, Safari* and *Mozilla Firefox* which stores a variety of configuration data (bookmarks, cookies, contacts etc.) inside internally managed SQLite databases.[9]

Although Buildbot's default database is SQLite, it also supports MySQL and PostgreSQL.

2.2 Web interface

In previous releases, the web part of buildbot was implemented using a WebStatus plugin. That has since become deprecated, and as of Buildbot 0.9.0, the built-in web server supersedes it.

The client side of the web UI is written in JavaScript and based on the Angular framework and concepts.

Being a Single Page Application, all Buildbot pages are loaded from the same path, at the master's base URL. A JavaScript technique is used to avoid reloading the whole page over HTTP when the users changes the URI or clicks a link.

2.2.1 Angular

Angular is a framework for building client applications in HTML and either JavaScript or a language like TypeScript that compiles to JavaScript (in Buildbot's case, CoffeeScript, because of its similarities to Python).[13]

The framework works by first reading the HTML page, which has additional custom tag attributes embedded into it. Angular interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic JSON resources.[14]

Angular is built around the belief that declarative code is better than imperative when it comes to building UIs and wiring software components together, while imperative code is excellent for expressing business logic.

Angular's design goals include:

- decoupling DOM manipulation from app logic, improving the testability of the code
- decoupling the client side of an application from the server side, allowing development work to progress in paralel

[2]Acronym for *Atomicity, Consistency, Isolation, Durability.* A set of properties of database transactions intended to guarantee validity even in the event of errors, power failures, etc.

- providing guidance through the entire journey of building an app, from designing the UI, through writing the business logic, to testing
- making common tasks trivial and difficult tasks possible[15]

2.2.2 NodeJS

Node.js (Node) is an open source development platform for executing JavaScript code server-side. Node is useful for developing applications that require a persistent connection from the browser to the server and is often used for real-time applications such as chat, news feeds and web push notifications.

Node.js is intended to run on a dedicated HTTP server and to employ a single thread with one process at a time. Node.js applications are event-based and run asynchronously. Code built on the Node platform does not follow the traditional model of receive, process, send, wait, receive. Instead, Node processes incoming requests in a constant event stack and sends small requests one after the other without waiting for responses.

This is a shift away from mainstream models that run larger, more complex processes and run several threads concurrently, with each thread waiting for its appropriate response before moving on.

One of the major advantages of Node.js, according to its creator Ryan Dahl, is that it does not block input/output (I/O). Some developers are highly critical of Node.js and point out that if a single process requires a significant number of CPU cycles, the application will block and that the blocking can crash the application. Proponents of the Node.js model claim that CPU processing time is less of a concern because of the high number of small processes that Node code is based on.[16]

2.2.3 npm

npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment *Node.js*. It consists of a command line client, also called *npm*, and an online database of public and paid-for private packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website.

npm is included as a recommended feature in Node.js installer.[17] npm consists of a command line client that interacts with a remote registry. It allows users to consume and distribute JavaScript modules that are available on the registry.[18] Packages on the registry are in CommonJS format and include a metadata file in JSON format.[19] Over 477,000 packages are available on the main npm registry.[20] The registry has no vetting process for submission, which means that packages found there can be low quality, insecure, or malicious.[19] Instead, npm relies on user reports to take down packages if they violate policies by being insecure, malicious or low quality.[21] npm exposes statistics including number of downloads and number of depending packages to assist developers in judging the quality of packages.[22]

npm can manage packages that are local dependencies of a particular project, as well as globally-installed JavaScript tools.[23] When used as a dependency manager for a local project, npm can install, in one command, all the dependencies of a project through the `package.json` file.[24] In the package.json file, each dependency can specify a range of valid versions using the semantic versioning scheme, allowing

developers to auto-update their packages while at the same time avoiding unwanted breaking changes.[25] npm also provides version-bumping tools for developers to tag their packages with a particular version.[26]

There are a number of alternatives to *npm* for installing modular JavaScript, most popular of which is *Yarn*, which was open-sourced by Facebook in October 2016.[27] One of the main reasons for this development was that *npm* broke down on Facebook's continuous integration environments, due to sandboxing and no internet access. Yarn replaces the existing workflow for the npm client while remaining compatible with the npm registry.

In the Node ecosystem, dependencies get placed within a `node_modules` directory in your project. However, this file structure can differ from the actual dependency tree as duplicate dependencies are merged together. The npm client installs dependencies into the `node_modules` directory non-deterministically. This means that based on the order dependencies are installed, the structure of a `node_modules` directory could be different from one person to another. These differences can cause "works on my machine" bugs that take a long time to hunt down.

Yarn resolves these issues around versioning and non-determinism by using lockfiles and an install algorithm that is deterministic and reliable. These lockfiles lock the installed dependencies to a specific version, and ensure that every install results in the exact same file structure in `node_modules` across all machines. The written lockfile uses a concise format with ordered keys to ensure that changes are minimal and review is simple.[28]

2.2.4 Gulp

gulp.js is an open-source JavaScript toolkit by Fractal Innovations and the open source community at GitHub, used as a streaming build system in front-end web development.[29]

It is a task runner built on *Node.js* and *npm*, used for automation of time-consuming and repetitive tasks involved in web development like minification, concatenation, cache busting, unit testing, linting, optimization, etc.[32]

gulp is a build tool in JavaScript built on node streams. These streams facilitate the connection of file operations through pipelines.[30] gulp reads the file system and pipes the data at hand from its one single-purposed plugin to other through the .pipe() operator, doing one task at a time. The original files are not affected until all the plugins are processed. It can be configured either to modify the original files or to create new ones. This grants the ability to perform complex tasks through linking its numerous plugins. The users can also write their own plugins to define their own tasks.[31] Unlike other task runners that run tasks by configuration, gulp requires knowledge of JavaScript and coding to define its tasks. gulp is a build system which means apart from running tasks, it is also capable of copying files from one location to another, compiling, deploying, creating notifications, unit testing, linting, etc.[29]

Task-runners like gulp and grunt are built on Node.js rather than npm, because the basic npm scripts are inefficient when executing multiple tasks. Even though some developers prefer npm scripts because they can be simple and easy to implement, there are numerous ways where gulp and grunt seem to have an advantage over each other and the default provided scripts.[33] Grunt runs tasks by transforming files and saves as new ones in temporary folders and the output of one task is taken as input for

another and so on until the output reaches the destination folder. This involves a lot of I/O calls and creation of many temporary files. Whereas gulp streams through the file system and does not require any of these temporary locations decreasing the number of I/O calls thus, improving performance.[34] Grunt uses configuration files to perform tasks whereas gulp requires its build file to be coded. In grunt, each plugin needs to be configured to match its input location to the previous plugin's output. In gulp, the plugins are automatically pipe-lined.[30]

The gulp tasks are run from a **Command Line Interface** (CLI)[33] shell and require `package.json` and `gulpfile.js` (simply gulpfile) in the project root directory. The gulpfile is where plugins are loaded and tasks are defined. First, all the necessary modules are loaded and then tasks are defined in the gulpfile. All the necessary plugins specified in the gulpfile are installed into the devDependencies.[24] The default task is run with `gulp`. Individual tasks can be defined by gulp.task and are run by `gulp <task> <othertask>`.[35] Complex tasks are defined by chaining the plugins with the help of the `.pipe()` operator.[36]

2.2.5 CoffeeScript

CoffeeScript is a programming language that transcompiles to *JavaScript*. It adds syntactic sugar inspired by Ruby, Python and Haskell in an effort to enhance JavaScript's brevity and readability.[37] Specific additional features include list comprehension and pattern matching.

Almost everything is an expression in CoffeeScript, for example `if`, `switch` and `for` expressions (which have no return value in JavaScript) return a value. As in Perl, these control statements also have postfix versions; for example, `if` can also be written after the conditional statement.

Many unnecessary parentheses and braces can be omitted; for example, blocks of code can be denoted by indentation instead of braces, function calls are implicit, and object literals are often detected automatically.

CoffeeScript was chosen for Buildbot development because of its familiarities with Python. As most of its developers are Python experts, they found it helpful to code the frontend using a similar syntax as the backend.[42]

As of December 2017, switching the buildbot frontend codebase from CoffeeScript to TypeScript or another more-modern language is being considered.[41]

2.2.5.1 Code snippets

To compute the greatest common divisor of two integers using the euclidean algorithm, in JavaScript one usually needs a `while` loop:

```
gcd = (x, y) => {
  do {
    z = x % y
    x = y
    y = z
  } while (y !== 0)
  return x
}
```

Listing 2.4: *loop.js*

Whereas in CoffeeScript one can use `until` and pattern-matching instead:

```
gcd = (x, y) ->
  [x, y] = [y, x%y] until y is 0
  x
```

Listing 2.5: *loop.coffee*

A linear search can be implemented in one line using the `when` keyword:

```
names = ["Long", "John", "Silver"]
linearSearch = (searchName) -> alert(name) for name in names when name
    is searchName
```

Listing 2.6: *linearSearch.coffee*

Ruby-style string interpolation is included in CoffeeScript. Double-quoted strings allow for interpolated values using `#{...}` and single-quoted strings are treated as literals:[38]

```
muppet = "Beeker"
favorite = "My favorite muppet is #{muppet}!"

# => "My favorite muppet is Beeker!"
```

Listing 2.7: *interpolation.coffee*

CoffeeScript allows us to chain two comparisons together in a form that more closely matches the way a mathematician would write it:[39]

```
maxDwarfism = 147
minAcromegaly = 213

height = 180

normalHeight = maxDwarfism < height < minAcromegaly
# => true
```

Listing 2.8: *comparingRanges.coffee*

A simple way to integrate small snippets of JavaScript code into a CoffeeScript codebase, without converting its syntax would be to wrap the JavaScript with backticks:[40]

```
`function greet(name) {
return "Hello "+name;
}`

# Back to CoffeeScript
greet "Coffee"
# => "Hello Coffee"
```

Listing 2.9: *embedJS.coffee*

2.2.6 pug

Formerly known as Jade (a registered trademark, and as a result a rename was needed), *pug* is a high performance and feature-rich templating engine. Simply put, *pug* is a clean, whitespace/indentation sensitive syntax for simplifying and escaping the archaic syntax of HTML.

Just like SASS, Pug is a prepocessor and, as such it facilitates accomplishing tasks like wrapping away repetitive work by providing features not available in plain HTML. It provides the ability to write dynamic and reusable HTML documents, it's an open source HTML templating language for Node.js (server-side JavaScript), totally free to use and provides fast and easy HTML.[43]

A basic syntax comparison can be seen below:

```
<html>
  <head>
    <title>This is my first Pug file</title>
  </head>
  <body>
    <header>
      <p>My soon to be menu</p>
    </header>
    <section>
      <p>This is a post about Pug template engine</p>
    </section>
    <footer>
      lots of copyrights
    </footer>
  </body>
</html>
```

Listing 2.10: *example.html*

```
html
    head
        title This is my first Pug file
    body
        header
            p My soon to be menu
        section
            p This is a post about Pug template engine
        footer
            lots of copyrights
```

Listing 2.11: *example.pug*

Chapter 3

Default implementations

3.1 Concepts and terminology

In its 15 years of existence, buildbot has defined some basic concepts that need to be understanded if one wants to configure and extend it.

3.1.1 Source Stamps

Source code comes from repositories, provided by version control systems. Repositories are generally identified by URLs, e.g., `git://github.com/buildbot/buildbot.git`.

In these days of distributed version control systems, the same *codebase* may appear in multiple repositories. For example, `https://github.com/mozilla/mozilla-central` and `http://hg.mozilla.org/mozilla-release` both contain the Firefox codebase, although not exactly the same code.

Many *projects* are built from multiple codebases. For example, a company may build several applications based on the same core library. The "app" codebase and the "core" codebase are in separate repositories, but are compiled together and constitute a single project. Changes to either codebase should cause a rebuild of the application.

Most version control systems define some sort of *revision* that can be used (sometimes in combination with a *branch*) to uniquely specify a particular version of the source code.

To build a project, Buildbot needs to know exactly which version of each codebase it should build. It uses a *source stamp* to do so for each codebase; the collection of sourcestamps required for a project is called a *source stamp set*.[44]

3.1.2 Changes

3.1.2.1 Who

Each `Change` has a `who` attribute, which specifies which developer is responsible for the change. This is a string which comes from a namespace controlled by the VC repository. Frequently this means it is a username on the host which runs the repository, but not all VC systems require this. Each `StatusNotifier` will map the `who` attribute into something appropriate for their particular means of communication: an email address, an IRC handle, etc.[45]

3.1.2.2 Files

It also has a list of `files`, which are just the tree-relative filenames of any files that were added, deleted, or modified for this `Change`. These filenames are used by the `fileIsImportant` function (in the scheduler) to decide whether it is worth triggering a new build or not, e.g. the function could use the following function to only run a build if a C file were checked in:[46]

```
def has_C_files(change):
    for name in change.files:
        if name.endswith(".c"):
            return True
    return False
```

Listing 3.1: *has_C_files.py*

3.1.2.3 Comments

The Change also has a `comments` attribute, which is a string containing any checkin comments.[47]

3.1.2.4 Project

The `project` attribute of a change or source stamp describes the project to which it corresponds, as a short human-readable string. This is useful in cases where multiple independent projects are built on the same buildmaster. In such cases, it can be used to control which builds are scheduled for a given commit, and to limit status displays to only one project.[48]

3.1.2.5 Repository

This attribute specifies the repository in which this change occurred. In the case of DVCS's, this information may be required to check out the committed source code. However, using the repository from a change has security risks: if Buildbot is configured to blindly trust this information, then it may easily be tricked into building arbitrary source code, potentially compromising the workers and the integrity of subsequent builds.[49]

3.1.2.6 Codebase

This attribute specifies the codebase to which this change was made. As described in the Source Stamps section, multiple repositories may contain the same codebase. A change's codebase is usually determined by the `codebaseGenerator` configuration. By default the codebase is '' (empty); this value is used automatically for single-codebase configurations.[50]

3.1.2.7 Revision

Each Change can have a `revision` attribute, which describes how to get a tree with a specific state: a tree which includes this Change (and all that came before it)

but none that come after it. If this information is unavailable, the `revision` attribute will be `None`. These revisions are provided by the `ChangeSource`.

Revisions are always strings.[51]

3.1.2.8 Branches

The Change might also have a `branch` attribute. This indicates that all of the Change's files are in the same named branch. The schedulers get to decide whether the branch should be built or not.[53]

3.1.2.9 Change Properties

A Change may have one or more properties attached to it, usually specified through the Force Build form or `sendchange`. These are useful for custom buildbot implementations.[52]

3.1.3 BuildSets

3.1.4 BuildRequests

3.1.5 Builders

3.1.6 Schedulers

3.1.7 Builds

3.1.8 Master

3.1.9 Workers

3.1.10 Users

3.1.11 Data API

3.2 Web interface

3.2.1 Home page

3.2.2 Grid view

3.2.3 Waterfall view

3.2.4 Console view

3.2.5 Settings

Chapter 4

Custom implementations

4.1 Backend componentization

split master configuration into multiple modules to facilitate scalability

4.1.1 Email look-up using LDAP

4.1.2 Log parsing

4.1.3 Preferential build steps

4.2 Web interface development

4.2.1 Flask/WSGI dashboards

pros/cons

4.2.2 Angular dashboards

pros/cons

4.3 User scripts

user try scripts to send patches with uncomitted code to buildbot for testing

4.4 Extending the source code

extending buildbot source to allow multiple patchfiles and more API entrypoints maybe?

Chapter 5

Conclusion

Bibliography

[1] McConnell, Steve, Code Complete, Microsoft Press, 2009, p. 100 2.1.1

[2] About Python, URL: https://www.python.org/about/ 2.1.1

[3] TIOBE Index for November 2017, URL: https://www.tiobe.com/tiobe-index/ 2.1.1

[4] Prechelt, Lutz, An empirical comparison of C, C++, Java, Perl, Python, Rexx, and TCL, Fakultät für Informatik, Universität Karlsruhe, 2000 2.1.1

[5] McKellar, J. and Fettig, A., Twisted Network Programming Essentials, O'Reilly Media, 2013, p. xiii 2.1.2

[6] McKellar, J. and Fettig, A., Twisted Network Programming Essentials, O'Reilly Media, 2013, p. 11 2.1.2.4

[7] Twisted Framework: Core ideas, URL: https://en.wikipedia.org/w/index.php?title=Twisted_(software)&oldid=808304626 2.1.2.3

[8] Owens, Michael, The Definitive Guide to SQLite, Apress, 2014, p. 133 2.1.3

[9] Most Widely Deployed SQL Database Engine, URL: https://sqlite.org/mostdeployed.html 2.1.3

[10] Introduction - Buildbot latest documentation, URL: http://docs.buildbot.net/current/manual/introduction.html 2.1

[11] Database - Buildbot latest documentation, URL: http://docs.buildbot.net/latest/developer/database.html 2.1.3

[12] Global Configuration - Buildbot latest documentation, URL: http://docs.buildbot.net/latest/manual/cfg-global.html 2.1.3

[13] Angular - Architecture Overview, URL: https://angular.io/guide/architecture 2.2.1

[14] AngularJS - Wikipedia, URL: https://en.wikipedia.org/w/index.php?title=AngularJS&oldid=812169171 2.2.1

[15] AngularJS: Developer Guide: Introduction, URL: https://docs.angularjs.org/guide/introduction 2.2.1

[16] What is Node.js? - Definition from WhatIs.com, URL: http://whatis.techtarget.com/definition/Nodejs 2.2.2

[17] A Beginner's Guide to npm - the Node Package Manager, URL: https://www.sitepoint.com/beginners-guide-node-package-manager 2.2.3

[18] Ampersand.js - Learn, URL: https://ampersandjs.com/learn/npm-browserify-and-modules 2.2.3

[19] Ojamaa, Andres; Duuna, Karl, Assessing the Security of Node.js Platform, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6470829 2.2.3

[20] Understanding npm, URL: https://unpm.nodesource.com 2.2.3

[21] npm Code of Conduct: acceptable package content, URL: https://www.npmjs.com/policies/conduct#acceptable-package-content 2.2.3

[22] npm-stat: download statistics for NPM packages, URL: https://npm-stat.com 2.2.3

[23] How To Use npm to Manage Node.js Packages on a Linux Server, URL: https://www.digitalocean.com/community/tutorials/how-to-use-npm-to-manage-node-js-packages-on-a-linux-server 2.2.3

[24] npm-install, URL: https://docs.npmjs.com/cli/install 2.2.3, 2.2.4

[25] semver, URL: https://docs.npmjs.com/misc/semver 2.2.3

[26] npm-version, URL: https://docs.npmjs.com/cli/version 2.2.3

[27] The npm Blog - Hello, Yarn!, URL: http://blog.npmjs.org/post/151660845210/hello-yarn 2.2.3

[28] Yarn: A new package manager for JavaScript | Engineering Blog | Facebook Code, URL: https://code.facebook.com/posts/1840075619545360/yarn-a-new-package-manager-for-javascript 2.2.3

[29] Mao, Jed; Schmitt, Maximilian; Stryjewski, Tomasz; Country Holt, Cary; Lubelski, Wiliam, Developing a Gulp Edge (1st ed.), Bleeding Edge Press, 2014 2.2.4

[30] substack/stream-handbook: how to write note programs with streams, URL: https://github.com/substack/stream-handbook 2.2.4

[31] gulpjs/gulp: Writing a plugin, URL: https://github.com/gulpjs/gulp/blob/master/docs/writing-a-plugin/README.md 2.2.4

[32] Building With Gulp - Smashing Magazine, URL: https://www.smashingmagazine.com/2014/06/building-with-gulp 2.2.4

[33] gulpjs/gulp: gulp CLI docs, URL: https://github.com/gulpjs/gulp/blob/master/docs/CLI.md 2.2.4

[34] Gulp for Beginners, URL: https://css-tricks.com/gulp-for-beginners 2.2.4

[35] gulpjs/gulp: Getting Started, URL: https://github.com/gulpjs/gulp/blob/master/docs/getting-started.md 2.2.4

[36] Maynard, Travis, Getting Started with Gulp, Packt Publishing Ltd., 2015 2.2.4

[37] MacCaw, Alex, The Little Book on CoffeeScript (1st ed.), O'Reilly Media, 2012 2.2.5

[38] CoffeeScript Cookbook - String Interpolation, URL: https://coffeescript-cookbook.github.io/chapters/strings/interpolation 2.2.5.1

[39] CoffeeScript Cookbook - Comparing Ranges, URL: https://coffeescript-cookbook.github.io/chapters/syntax/comparing_ranges 2.2.5.1

[40] CoffeeScript Cookbook - Embedding JavaScript, URL: https://coffeescript-cookbook.github.io/chapters/syntax/embedding_javascript 2.2.5.1

[41] coffeescript -> javascript ES6 (ES2015) ? · Issue \#3804 · buildbot/buildbot - Embedding JavaScript, URL: https://github.com/buildbot/buildbot/issues/3804 2.2.5

[42] CoffeeScript Coding Style - Buildbot latest documentation, URL: http://docs.buildbot.net/latest/developer/coffeescript-style.html 2.2.5

[43] Getting started with Pug template engine - Codeburst, URL: https://codeburst.io/getting-started-with-pug-template-engine-e49cfa291e33 2.2.6

[44] Concepts - Buildbot latest documentation: Source Stamps, URL: http://docs.buildbot.net/latest/manual/concepts.html#source-stamps 3.1.1

[45] Concepts - Buildbot latest documentation: Who, URL: http://docs.buildbot.net/latest/manual/concepts.html#who 3.1.2.1

[46] Concepts - Buildbot latest documentation: Files, URL: http://docs.buildbot.net/latest/manual/concepts.html#files 3.1.2.2

[47] Concepts - Buildbot latest documentation: Comments, URL: http://docs.buildbot.net/latest/manual/concepts.html#comments 3.1.2.3

[48] Concepts - Buildbot latest documentation: Project, URL: http://docs.buildbot.net/latest/manual/concepts.html#project 3.1.2.4

[49] Concepts - Buildbot latest documentation: Repository, URL: http://docs.buildbot.net/latest/manual/concepts.html#repository 3.1.2.5

[50] Concepts - Buildbot latest documentation: Codebase, URL: http://docs.buildbot.net/latest/manual/concepts.html#codebase 3.1.2.6

[51] Concepts - Buildbot latest documentation: Revision, URL: http://docs.buildbot.net/latest/manual/concepts.html#revision 3.1.2.7

[52] Concepts - Buildbot latest documentation: Change Properties, URL: http://docs.buildbot.net/latest/manual/concepts.html#change-properties 3.1.2.9

[53] Concepts - Buildbot latest documentation: Branches, URL: http://docs.buildbot.net/latest/manual/concepts.html#branches 3.1.2.8

[54] Concepts - Buildbot latest documentation: Source Stamps, URL: http://docs.buildbot.net/latest/manual/concepts.html#source-stamps

[55] Concepts - Buildbot latest documentation: Source Stamps, URL: http://docs.buildbot.net/latest/manual/concepts.html#source-stamps

[56] Concepts - Buildbot latest documentation: Source Stamps, URL: http://docs.buildbot.net/latest/manual/concepts.html#source-stamps