Buckaroo Documentation

Release

LoopPerfect Limited

Jul 05, 2017

Contents

Buckaroo is a dependency manager for C/C++. Using Buckaroo, it is easy to add external modules to your project in a controlled and cross-platform manner.

To use Buckaroo, you will also need to install Buck. Buck is a build system developed and maintained by Facebook, and it is used by all Buckaroo packages.

CHAPTER 1

Contents

Installation

Buckaroo is available for macOS, Linux and Windows (preview).

macOS

Buckaroo can be installed using Homebrew.

Add Facebook's tap so that Homebrew can find Buck.

```
brew tap facebook/fb
brew install --HEAD loopperfect/lp/buckaroo
```

The Homebrew formula will install Buck and Java, if required.

Verify your installation with:

```
buckaroo version
```

Linux

Buckaroo can be installed using Linuxbrew.

Add Facebook's tap so that Linuxbrew can find Buck.

```
brew tap facebook/fb
brew install --HEAD loopperfect/lp/buckaroo
```

The Linuxbrew formula will install Buck and Java, if required.

Verify your installation with:

```
buckaroo version
```

Windows (preview)

Ensure that you have Buck installed, then clone the Buckaroo source-code from GitHub:

```
git clone git@github.com:njlr/buckaroo.git
cd buckaroo
```

Build Buckaroo with Buck:

```
buck build :buckaroo-cli
```

Buck will output a runnable Jar file in the output folder:

```
java -jar .\\buck-out\\gen\\buckaroo-cli.jar
```

Ensure that this command is on your PATH.

Analytics

By default, Buckaroo will report usage statistics to our servers. These logs allow us to improve Buckaroo by targeting real-world usage. All logs are transferred over HTTPS and are not shared with any third-party.

What is Shared?

The analytics events are as follows:

```
{
  session, // Random UUID generated on installation
  data: {
    os, // The OS name, e.g. "macOS"
    version, // The version of Buckaroo installed
    command // The command sent to Buckaroo
  }
}
```

If in doubt, please refer to the source-code of Buckaroo or drop us an email.

Disabling Analytics

If you wish to disable analytics, follow these steps:

1. Launch Buckaroo at least once:

```
buckaroo version
```

2. Open the config.json file in your Buckaroo home folder:

```
open ~/.buckaroo/config.json
```

3. Remove the property “analytics”. For example:

```
{
  "cookBooks": [
    {
      "name": "buckaroo-recipes",
      "url": "git@github.com:loopperfect/buckaroo-recipes.git"
    }
  ]
}
```

Quickstart

Creating a Project

The fastest way to get started is to use the Quickstart feature.

First, create a directory for your project:

```
mkdir my-project
cd my-project
```

Now, run the Buckaroo quickstart command:

```
buckaroo quickstart
```

This command will generate a hello-world application, folders for your source-code and a Buck build file. Buckaroo does not require a particular project layout, so feel free to tweak the Buck file.

Let's verify that the project is working as expected:

```
buck run :my-project
```

You now have everything ready to start installing dependencies.

Adding a dependency

Once you have a project file, we can start adding dependencies. Let's add range-v3 by Eric Niebler. range-v3 is a powerful range library for C++ 11 and up.

```
buckaroo install ericniebler/range-v3
```

Buckaroo will have downloaded the range-v3 source-code from GitHub and installed it locally in your project folder. We can now use the library in a sample application!

Sample Application

Our example requires some C++ 14 features, so if your compiler does not enable them by default we will need to update the project's BUCK file.

Update BUCK to:

```
include_defs('//BUCKAROO_DEPS')

cxx_binary(
  name = 'my-project',
  header_namespace = 'my-project',
  srcs = glob([
    'my-project/src/**/*.cpp',
  ]),
  headers = subdir_glob([
    ('my-project/include', '**/*.hpp'),
    ('my-project/include', '**/*.h'),
  ]),
  compiler_flags = [
    '-std=c++14',
  ],
  deps = BUCKAROO_DEPS,
)
```

Now, let's update the main.cpp file to a simple range-v3 example:

```
#include <iostream>
#include <vector>
#include <range/v3/all.hpp>

int main() {
  auto const xs = std::vector<int>({ 1, 2, 3, 4, 5 });
  auto const ys = xs
    | ranges::view::transform([](auto x) { return x * x; })
    | ranges::to_vector;
  for (auto const& i : ys) {
    std::cout << i << std::endl;
  }
  return 0;
}
```

Run the project again and you will see a list of square numbers, computed by range-v3.

```
buck run :my-project
```

.gitignore

If you are tracking your project with Git, add the following to your .gitignore:

```
/buck-out/
/.buckd/
/buckaroo/
BUCKAROO_DEPS
.buckconfig.local
```

Explore Buckaroo

range-v3 is just one of the many packages already available for Buckaroo. You can browse them on buckaroo.pm or request more on the wishlist.

Command Line Interface

Buckaroo provides a command-line interface for managing your project's dependencies. The following commands are the recommended way to use Buckaroo.

Init

```
buckaroo init
```

Init is used to generate a project file in the current directory.

Quickstart

```
buckaroo quickstart
```

Quickstart is similar to Init, but also generates the necessary boiler-plate for a new C++ project. You should use Quickstart when starting a new project.

Install

```
buckaroo install google/gtest
```

Install can be used to add dependencies to your project.

If you do not supply a module name, then the existing dependencies of the project are fetched and installed.

```
buckaroo install
```

Uninstall

```
buckaroo uninstall google/gtest
```

Uninstall can be used to remove a dependency from your project. Note that the remaining dependencies are recomputed since their resolved versions may have changed as a result.

Upgrade

```
buckaroo upgrade
```

Upgrades the cook-books installed on your system. This allows you to use benefit from recipe improvements, additions and fixes since you first installed Buckaroo.

Version

```
buckaroo version
```

Outputs the version of Buckaroo that is installed.

Help

```
buckaroo help
```

Anatomy of a Recipe

A Buckaroo recipe is a file that describes a module that can be imported into your projects. Recipes are encoded as JSON.

Example

For example, here's a recipe for the Beast HTTP library:

```
{
  "name": "Beast",
  "license": "BSL-1.0",
  "url": "https://github.com/vinniefalco/beast",
  "github": {
    "owner": "vinniefalco",
    "project": "beast"
  },
  "versions": {
    "0.0.1": {
      "source": {
        "url": "https://github.com/vinniefalco/Beast/archive/
↪6d5547a32c50ec95832c4779311502555ab0ee1f.zip",
        "sha256": "8a600dfa3394164f79ad7dfa6942d8d4b6c6c7e5b8cc9b5f82519b682db25aae",
        "subPath": "Beast-6d5547a32c50ec95832c4779311502555ab0ee1f"
      },
      "buck": {
        "url": "https://raw.githubusercontent.com/njlr/Beast/
↪71d1bde64bf5ee52579441b00ad446959231d8d2/BUCK",
        "sha256": "1dce5d9f5c883e193e54c2dfc033a5485b9e8e87bb34d8c38ab03ed148ccd968"
      },
      "dependencies": {
        "boost/range": "1.63.0",
        "boost/intrusive": "1.63.0",
        "boost/lexical-cast": "1.63.0",
        "boost/math": "1.63.0",
        "boost/system": "1.63.0",
        "boost/asio": "1.63.0"
      }
    }
  }
}
```

Name

```
"name": "Beast",
```

This is the friendly name of the package.

Note that the file-name of the recipe is the identifier used by the `install` command, so the `name` field may contain spaces, etc.

Meta-data

```
"license": "BSL-1.0",
"url": "https://github.com/vinniefalco/beast",
"github": {
  "owner": "vinniefalco",
  "project": "beast"
},
```

This optional meta-data is used to generate the pages on buckaroo.pm. Note that the license should be a comma-separated list of SPDX codes.

Versions

```
"versions": {
```

The `versions` element is a dictionary of semantic versions to source locations.

```
"0.0.1": {
  "source": {
    "url": "https://github.com/vinniefalco/Beast/archive/
↪6d5547a32c50ec95832c4779311502555ab0ee1f.zip",
    "sha256": "8a600dfa3394164f79ad7dfa6942d8d4b6c6c7e5b8cc9b5f82519b682db25aae",
    "subPath": "Beast-6d5547a32c50ec95832c4779311502555ab0ee1f"
  },
  "buck": {
    "url": "https://raw.githubusercontent.com/njlr/Beast/
↪71d1bde64bf5ee52579441b00ad446959231d8d2/BUCK",
    "sha256": "1dce5d9f5c883e193e54c2dfc033a5485b9e8e87bb34d8c38ab03ed148ccd968"
  },
  "dependencies": {
    "boost/range": "1.63.0",
    "boost/intrusive": "1.63.0",
    "boost/lexical-cast": "1.63.0",
    "boost/math": "1.63.0",
    "boost/system": "1.63.0",
    "boost/asio": "1.63.0"
  }
}
```

Each version object contains information on how to fetch the source-code. In case the source-code does not contain a `BUCK` file, one may be injected using the information in the `buck` element.

Source

```
"source": {
  "url": "https://github.com/vinniefalco/Beast/archive/
↪6d5547a32c50ec95832c4779311502555ab0ee1f.zip",
  "sha256": "8a600dfa3394164f79ad7dfa6942d8d4b6c6c7e5b8cc9b5f82519b682db25aae",
  "subPath": "Beast-6d5547a32c50ec95832c4779311502555ab0ee1f"
},
```

Every `source` should point to a zip-file containing the source-code of the module. It is hashed to ensure integrity, so always choose a URL that is stable.

The `subPath` element can be used to change the root of the source-code.

For example, suppose the zip-file has this structure:

```
.
+-- module.zip
    +-- sources
        +-- include
            +-- math.hpp
            +-- utils.hpp
```

If the `subPath` is `"sources/include"`, then the extracted code would be:

```
.
+-- math.hpp
+-- utils.hpp
```

This feature is particularly useful for GitHub, which puts source-code into a sub-folder called `<project>-<commit>`.

Buck

```
"buck": {
  "url": "https://raw.githubusercontent.com/njlr/Beast/
↪71d1bde64bf5ee52579441b00ad446959231d8d2/BUCK",
  "sha256": "1dce5d9f5c883e193e54c2dfc033a5485b9e8e87bb34d8c38ab03ed148ccd968"
},
```

The `buck` element points to a remote Buck definition. If present, the remote `BUCK` file gets saved into the root folder of the source-code. This allows us to support packages that do not currently support Buck.

The `buck` element is only required when the source-code does not already contain a `BUCK` file.

Dependencies

```
"dependencies": {
  "boost/range": "1.63.0",
  "boost/intrusive": "1.63.0",
  "boost/lexical-cast": "1.63.0",
  "boost/math": "1.63.0",
  "boost/system": "1.63.0",
  "boost/asio": "1.63.0"
}
```

The `dependencies` element defines the modules that the recipe requires to build. These are of the following format:

```
<owner>/<project>: <version>
```

A small DSL is provided for versioning:

```
*                   // Any version
2                   // Exactly version 2.0.0
1.2                 // Exactly version 1.2.0
```

```
1.2.3                // Exactly version 1.2.3
1.0.1 - 4.3          // Between 1.0.1 and 4.3 inclusive
[ 7.2, 7.3, 8 ]      // Either 7.2.0, 7.3.0 or 8.0.0
>= 4.7               // Version 4.7 or greater
<= 6.5.1             // Version 6.5.1 or greater
```

When multiple versions of a dependency can be resolved, the higher version is always chosen.