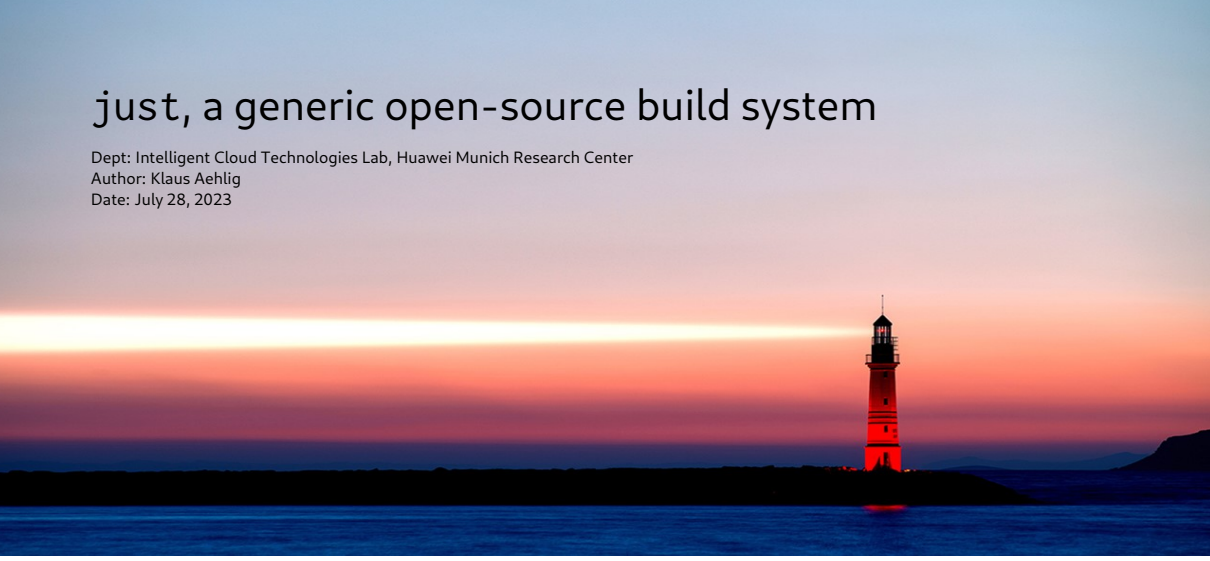
just, a generic open-source build system

Dept: Intelligent Cloud Technologies Lab, Huawei Munich Research Center
Author: Klaus Aehlig
Date: July 28, 2023



HUAWEI

Introduction

- `justbuild` is an open-source software build system (Apache 2.0 license)
  - Version 1.0.0 on December 12, 2022
  - active ongoing development
- Designed and built from scratch
  - no legacy requirements
  - can explore new ideas (staging, target-level caching, …)

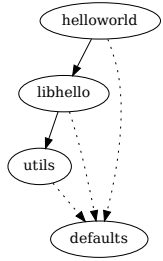
  … but building on successful ideas of existing tools
- Recall: a build system
  - computes a function (from source tree to artifacts)
  - uses concepts meaningful to a programmer (“library”, “binary”, …)
    not: individual compiler invocations, object files, …
  - must be correct, should be fast

just Example

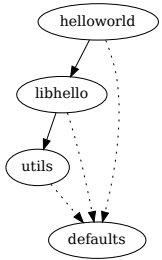
```
$ cat TARGETS
{ "helloworld":
  { "type": ["@", "rules", "CC", "binary"]
  , "name": ["helloworld"]
  , "srcs": ["main.cpp"]
  , "private-deps": ["libhello"]
  }
, "libhello":
  { "type": ["@", "rules", "CC", "library"]
  , "name": ["hello"]
  , "srcs": ["hello.cpp"]
  , "hdrs": ["hello.hpp"]
  , "deps": ["utils"]
  }
, "utils":
  { "type": ["@", "rules", "CC", "library"]
  , "name": ["utils"]
  , "srcs": ["utils.cpp"]
  , "hdrs": ["utils.hpp"]
  }
}
$
```

just Example

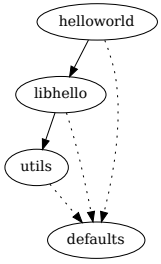
```
$ cat TARGETS
{ "helloworld":
  { "type": ["@", "rules", "CC", "binary"]
  , "name": ["helloworld"]
  , "srcs": ["main.cpp"]
  , "private-deps": ["libhello"]
  }
, "libhello":
  { "type": ["@", "rules", "CC", "library"]
  , "name": ["hello"]
  , "srcs": ["hello.cpp"]
  , "hdrs": ["hello.hpp"]
  , "deps": ["utils"]
  }
, "utils":
  { "type": ["@", "rules", "CC", "library"]
  , "name": ["utils"]
  , "srcs": ["utils.cpp"]
  , "hdrs": ["utils.hpp"]
  }
}
$
```

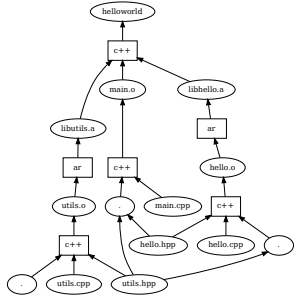


just Example

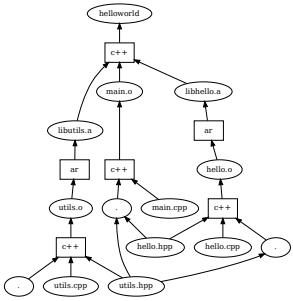


just Example

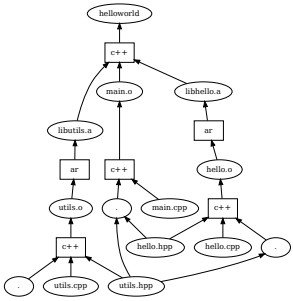




just Example

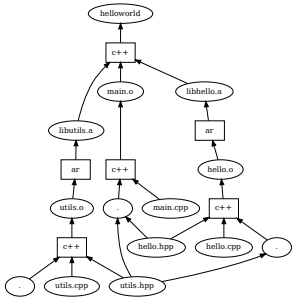


just Example




```
$ just build -C repos.json helloworld
INFO: Requested target is [["@","","","helloworld"],{}]
INFO: Analysed target [["@","","","helloworld"],{}]
INFO: Discovered 6 actions, 3 trees, 0 blobs
INFO: Building [["@","","","helloworld"],{}].
INFO: Processed 6 actions, 0 cache hits.
INFO: Artifacts built, logical paths are:
        helloworld [9dafb06bf5eacc62eb8f538ab1e4dab8a6339dc3:24704:x]
$
```

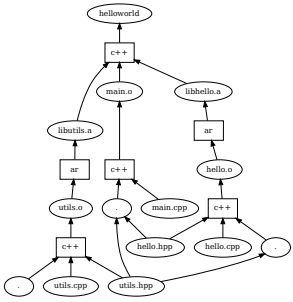
just Example




```
$ just build -C repos.json helloworld
INFO: Requested target is [["@","","","helloworld"],{}]
INFO: Analysed target [["@","","","helloworld"],{}]
INFO: Discovered 6 actions, 3 trees, 0 blobs
INFO: Building [["@","","","helloworld"],{}].
INFO: Processed 6 actions, 0 cache hits.
INFO: Artifacts built, logical paths are:
        helloworld [9dafb06bf5eacc62eb8f538ab1e4dab8a6339dc3:24704:x]
$

$ just install -C repos.json -o . helloworld
INFO: Requested target is [["@","","","helloworld"],{}]
INFO: Analysed target [["@","","","helloworld"],{}]
INFO: Discovered 6 actions, 3 trees, 0 blobs
INFO: Building [["@","","","helloworld"],{}].
INFO: Processed 6 actions, 6 cache hits.
INFO: Artifacts can be found in:
      /worker/build/62dbe7fb60915e38/root/work/helloworld/./helloworld [9dafb06bf5eacc62eb8f538ab1e4dab8a6339dc3:2
$
```

just Example




```
$ just build -C repos.json helloworld
INFO: Requested target is [["@","","","helloworld"],{}]
INFO: Analysed target [["@","","","helloworld"],{}]
INFO: Discovered 6 actions, 3 trees, 0 blobs
INFO: Building [["@","","","helloworld"],{}].
INFO: Processed 6 actions, 0 cache hits.
INFO: Artifacts built, logical paths are:
        helloworld [9dafb06bf5eacc62eb8f538ab1e4dab8a6339dc3:24704:x]
$

$ just install -C repos.json -o . helloworld
INFO: Requested target is [["@","","","helloworld"],{}]
INFO: Analysed target [["@","","","helloworld"],{}]
INFO: Discovered 6 actions, 3 trees, 0 blobs
INFO: Building [["@","","","helloworld"],{}].
INFO: Processed 6 actions, 6 cache hits.
INFO: Artifacts can be found in:
      /worker/build/62dbe7fb60915e38/root/work/helloworld/./helloworld [9dafb06bf5eacc62eb8f538ab1e4dab8a6339dc3:2
$

$ ./helloworld
Hello World!
$
```

Remote Build Execution

A remote build execution system consists of

Remote Build Execution

A remote build execution system consists of

- a Content-Adressable Store (CAS)
  (files, indexed by (essentially) their hash)

Remote Build Execution

A remote build execution system consists of

- a Content-Adressable Store (CAS)
  (files, indexed by (essentially) their hash)
- the actual execution service

Remote Build Execution

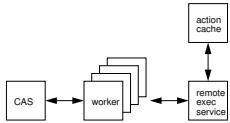
A remote build execution system consists of

- a Content-Adressable Store (CAS)
  (files, indexed by (essentially) their hash)
- the actual execution service
  - using many workers, sharing files via the CAS

Remote Build Execution

A remote build execution system consists of

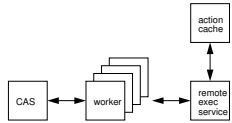
- a Content-Adressable Store (CAS)
  (files, indexed by (essentially) their hash)
- the actual execution service
  - using many workers, sharing files via the CAS
  - using an action cache (AC)



Remote Build Execution

A remote build execution system consists of

- a Content-Adressable Store (CAS)
- the actual execution service



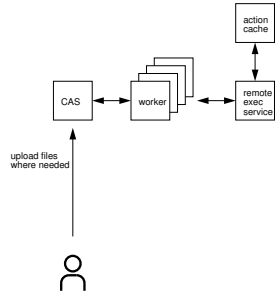
Remote Build Execution

A remote build execution system consists of

- a Content-Adressable Store (CAS)
- the actual execution service

To execute an action

- files unknown to the CAS are uploaded



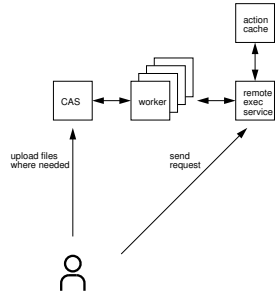
Remote Build Execution

A remote build execution system consists of

- a Content-Adressable Store (CAS)
- the actual execution service

To execute an action

- files unknown to the CAS are uploaded
- the action is requested



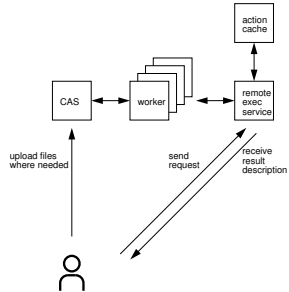
Remote Build Execution

A remote build execution system consists of

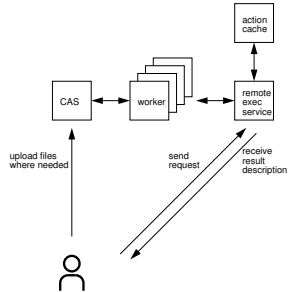
- a Content-Adressable Store (CAS)
- the actual execution service

To execute an action

- files unknown to the CAS are uploaded
- the action is requested
- a description of the output is received, typically from AC



Remote Build Execution



A remote build execution system consists of

- a Content-Adressable Store (CAS)
- the actual execution service

To execute an action

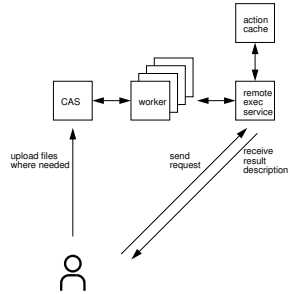
- files unknown to the CAS are uploaded
- the action is requested
- a description of the output is received, typically from AC
- actual artifacts can be downloaded from CAS, should they be needed

Remote Build Execution

A remote build execution system consists of

- a Content-Adressable Store (CAS)
- the actual execution service

To execute an action upload, request, receive answer

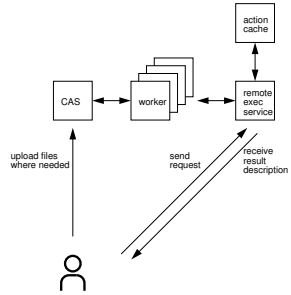


Remote Build Execution

A remote build execution system consists of

- a Content-Adressable Store (CAS)
- the actual execution service

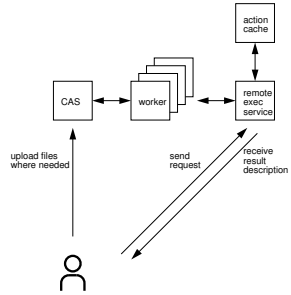
To execute an action upload, request, receive answer



Benefits of remote execution

- every action executed in isolation; dependencies are correct
- AC can be shared between developpers
- better parallelism

Remote Build Execution



A ~~remote~~ build execution system consists of

- a Content-Adressable Store (CAS)
- the actual execution service

To execute an action upload, request, receive answer

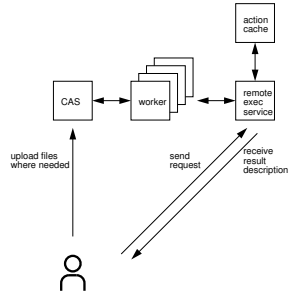
Benefits of remote execution

- every action executed in isolation; dependencies are correct
- AC can be shared between developpers
- better parallelism

But also works locally!

⇝ actions can have their own view and output convention *(conflict-free by design)*

Remote Build Execution



A ~~remote~~ build execution system consists of

- a Content-Adressable Store (CAS)
- the actual execution service

To execute an action upload, request, receive answer

Benefits of remote execution

- every action executed in isolation; dependencies are correct
- AC can be shared between developpers
- better parallelism

But also works locally!

⇝ actions can have their own view and output convention *(conflict-free by design)*

As people use `git` as VCS, let's use `git` blob/tree identifiers everywhere!

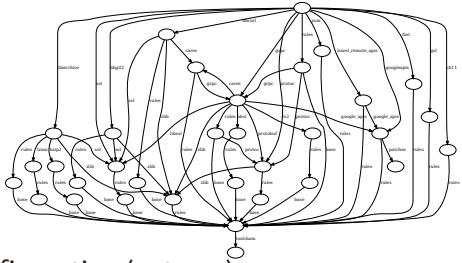
Multi-Repository Builds

- Code can be split over many repositories
  (also good do avoid duplication, e.g., rules)

Multi-Repository Builds

- Code can be split over many repositories
  (also good do avoid duplication, e.g., rules)
- Have to refer to other repositories
  - agreeing on global names doesn't work
  - often "any `libfoo` will do"

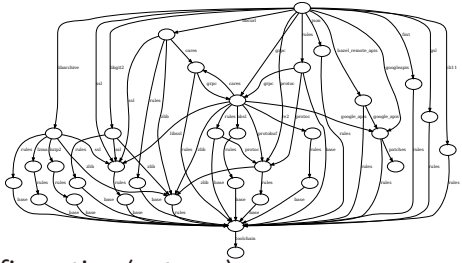
Multi-Repository Builds



- Code can be split over many repositories
  (also good do avoid duplication, e.g., rules)
- Have to refer to other repositories
  - agreeing on global names doesn't work
  - often "any `libfoo` will do"

⇝ use local names and bind in a project configuration (get DFA)

Multi-Repository Builds

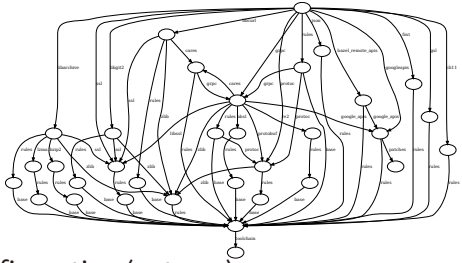


- Code can be split over many repositories
  (also good do avoid duplication, e.g., rules)
- Have to refer to other repositories
  - agreeing on global names doesn't work
  - often "any `libfoo` will do"

  ⇝ use local names and bind in a project configuration (get DFA)
- Repo semantics must be independent of caller—or if "main" repository

  ⇝ let targets decide where to logically place artifacts; we have staging anyway

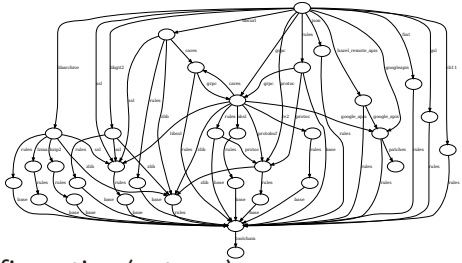
Multi-Repository Builds



- Code can be split over many repositories
  (also good do avoid duplication, e.g., rules)
- Have to refer to other repositories
  - agreeing on global names doesn't work
  - often "any `libfoo` will do"

  ⇝ use local names and bind in a project configuration (get DFA)
- Repo semantics must be independent of caller—or if "main" repository
  ⇝ let targets decide where to logically place artifacts; we have staging anyway
- As location doesn't matter, can as well use `git` trees as roots ⇝ quickly get blob ids
  (and use one default repository to store everything)

Multi-Repository Builds



- Code can be split over many repositories
  (also good do avoid duplication, e.g., rules)
- Have to refer to other repositories
  - agreeing on global names doesn't work
  - often "any `libfoo` will do"

  ⇝ use local names and bind in a project configuration (get DFA)
- Repo semantics must be independent of caller—or if "main" repository
  ⇝ let targets decide where to logically place artifacts; we have staging anyway
- As location doesn't matter, can as well use `git` trees as roots ⇝ quickly get blob ids
  (and use one default repository to store everything)
- Additional benefit: target-level caching if reachable part of DFA unchanged
  (minimal DFA as canonical representation; plus target name, configuration)
  ⇝ keep graphs to handle small; still flexible to build in different configurations

Getting Started

- Start with multi-repo right away
  - rules from a seperate repository
  - `src` versus `test`

Getting Started

- Start with multi-repo right away
  - rules from a seperate repository
  - `src` versus `test`

⇝ template with the local repositories (at `etc/repos.template.json`)

```
{ "repositories":
  { "":
    { "repository": {"type": "file", "path": "src"}
    , "bindings": {"rules": "rules-cc"}
    }
  , "test":
    { "repository": {"type": "file", "path": "test"}
    , "bindings": {"rules": "rules-cc", "src": ""}
    }
  }
}
```

Getting Started

- Start with multi-repo right away
  - rules from a seperate repository
  - `src` versus `test`

⇝ template with the local repositories (at `etc/repos.template.json`)

- Programmatic imports via `just-import-git`
  - Repository plus transitive dependencies
  - `file` repositories become subdirs of the `git` tree
  - Appropriate renaming to avoid conflicts

```
{ "repositories":
  { "":
    { "repository": {"type": "file", "path": "src"}
    , "bindings": {"rules": "rules-cc"}
    }
  , "test":
    { "repository": {"type": "file", "path": "test"}
    , "bindings": {"rules": "rules-cc", "src": ""}
    }
  }
}
```

```
#!/bin/sh
set -eu
readonly ROOT=$(readlink -f $(dirname $0)/..)

just-import-git -C ${ROOT}/etc/repos.template.json \
      --as rules-cc -b master \
      https://github.com/just-buildsystem/rules-cc \
    | cfmtjson \
    > ${ROOT}/etc/repos.json
```

Getting Started

- Start with multi-repo right away
  - rules from a seperate repository
  - `src` versus `test`

⇝ template with the local repositories (at `etc/repos.template.json`)

```
{ "repositories":
  { "":
    { "repository": {"type": "file", "path": "src"}
    , "bindings": {"rules": "rules-cc"}
    }
  , "test":
    { "repository": {"type": "file", "path": "test"}
    , "bindings": {"rules": "rules-cc", "src": ""}
    }
  }
}
```

- Programmatic imports via `just-import-git`
  - Repository plus transitive dependencies
  - `file` repositories become subdirs of the `git` tree
  - Appropriate renaming to avoid conflicts

⇝ Complete `etc/repos.json` with pinned commits
(simple update by running the import script again)

```
#!/bin/sh
set -eu
readonly ROOT=$(readlink -f $(dirname $0)/..)

just-import-git -C ${ROOT}/etc/repos.template.json \
      --as rules-cc -b master \
      https://github.com/just-buildsystem/rules-cc \
    | cfmtjson \
    > ${ROOT}/etc/repos.json
```

Getting Started

- Start with multi-repo right away
  - rules from a seperate repository
  - `src` versus `test`

⇝ template with the local repositories (at `etc/repos.template.json`)

```
{ "repositories":
  { "":
    { "repository": {"type": "file", "path": "src"}
    , "bindings": {"rules": "rules-cc"}
    }
  , "test":
    { "repository": {"type": "file", "path": "test"}
    , "bindings": {"rules": "rules-cc", "src": ""}
    }
  }
}
```

- Programmatic imports via `just-import-git`
  - Repository plus transitive dependencies
  - `file` repositories become subdirs of the `git` tree
  - Appropriate renaming to avoid conflicts

⇝ Complete `etc/repos.json` with pinned commits
(simple update by running the import script again)

```
#!/bin/sh
set -eu
readonly ROOT=$(readlink -f $(dirname $0)/..)

just-import-git -C ${ROOT}/etc/repos.template.json \
      --as rules-cc -b master \
      https://github.com/just-buildsystem/rules-cc \
    | cfmtjson \
    > ${ROOT}/etc/repos.json
```

- Simply use by: `just-mr build`
  (or `just-mr --main test build`)
  - `just-mr` fetches the dependencies (if commit not present already)
  - then launches `just` with the correct repository configuration

Rules: Data of a Target

- Rules are used to describe targets of a given type, like a C++ library

Rules: Data of a Target

- Rules are used to describe targets of a given type, like a C++ library
- Targets are given by

Rules: Data of a Target

- Rules are used to describe targets of a given type, like a C++ library
- Targets are given by
  - the actual artifact, like `libfoo.a`

Rules: Data of a Target

- Rules are used to describe targets of a given type, like a C++ library
- Targets are given by
  - the actual artifact, like `libfoo.a`
  - additional files that should be installed with the target, like headers

Rules: Data of a Target

- Rules are used to describe targets of a given type, like a C++ library
- Targets are given by
  - the actual artifact, like `libfoo.a`
  - additional files that should be installed with the target, like headers
  - any additional information needed to use the target
    (no reflection on the dependency graph!)
    - Headers of public dependencies
    - Information on how to link, including libraries depended upon
    - …

Rules: Data of a Target (Example)

```
$ cat TARGETS
{ "helloworld":
  { "type": ["@", "rules", "CC", "binary"]
  , "name": ["helloworld"]
  , "srcs": ["main.cpp"]
  , "private-deps": ["libhello"]
  }
, "libhello":
  { "type": ["@", "rules", "CC", "library"]
  , "name": ["hello"]
  , "srcs": ["hello.cpp"]
  , "hdrs": ["hello.hpp"]
  , "deps": ["utils"]
  }
, "utils":
  { "type": ["@", "rules", "CC", "library"]
  , "name": ["utils"]
  , "srcs": ["utils.cpp"]
  , "hdrs": ["utils.hpp"]
  }
}
$
```

Rules: Data of a Target (Example)

```
$ just analyse -C repos.json libhello
INFO: Requested target is [["@","","","libhello"],{}]
INFO: Result of target [["@","","","libhello"],{}]: {
        "artifacts": {
          "libhello.a": {"data":{"id":"bfd40392f507f1cea464b32003b0b7f79f70cf2f","path":"libhello.a"},"type":"ACTION"}
        },
        "provides": {
          "compile-args": [
          ],
          "compile-deps": {
            "utils.hpp": {"data":{"path":"utils.hpp","repository":""},"type":"LOCAL"}
          },
          "link-args": [
            "libhello.a",
            "libutils.a"
          ],
          "link-deps": {
            "libutils.a": {"data":{"id":"b8b7e151fc1a3c73f7cd3b8d6f6bcc2a353493a1","path":"libutils.a"},"type":"ACTION"}
          },
          "package": {
            "cflags-files": {},
            "ldflags-files": {},
            "name": "hello"
          }
        },
        "runfiles": {
          "hello.hpp": {"data":{"path":"hello.hpp","repository":""},"type":"LOCAL"}
        }
      }
$
```

Rule Language

Rules are mainly given by a functional expression defining this value; language

Rule Language

Rules are mainly given by a functional expression defining this value; language

- variables, `let*`-binding, conditional expressions, ...
- constructor functions for lists, maps, ...
- standard operations: accessor functions, concatenation, iteration (lists/maps), `foldl`, (conflict-free) map union, `nub_right`, ...

Rule Language

Rules are mainly given by a functional expression defining this value; language

- variables, `let*`-binding, conditional expressions, ...
- constructor functions for lists, maps, ...
- standard operations: accessor functions, concatenation, iteration (lists/maps), `foldl`, (conflict-free) map union, `nub_right`, ...
- Accessor functions to the data of the targets in the target fields

Rule Language

Rules are mainly given by a functional expression defining this value; language

- variables, `let*`-binding, conditional expressions, ...
- constructor functions for lists, maps, ...
- standard operations: accessor functions, concatenation, iteration (lists/maps), `foldl`, (conflict-free) map union, `nub_right`, ...
- Accessor functions to the data of the targets in the target fields
- actions are a means to define artifacts

Rule Language

Rules are mainly given by a functional expression defining this value; language

- variables, `let*`-binding, conditional expressions, ...
- constructor functions for lists, maps, ...
- standard operations: accessor functions, concatenation, iteration (lists/maps), `foldl`, (conflict-free) map union, `nub_right`, ...
- Accessor functions to the data of the targets in the target fields
- actions are a means to define artifacts
  - function returning a map of the output artifacts

Rule Language

Rules are mainly given by a functional expression defining this value; language

- variables, `let*`-binding, conditional expressions, ...
- constructor functions for lists, maps, ...
- standard operations: accessor functions, concatenation, iteration (lists/maps), `foldl`, (conflict-free) map union, `nub_right`, ...
- Accessor functions to the data of the targets in the target fields
- actions are a means to define artifacts
  - function returning a map of the output artifacts
  - inputs: stage of input artifacts, command vector, environment, expected outputs *(as well as execution properties, timeout-related information, ...)*

Rule Language

Rules are mainly given by a functional expression defining this value; language

- variables, `let*`-binding, conditional expressions, …
- constructor functions for lists, maps, …
- standard operations: accessor functions, concatenation, iteration (lists/maps), `foldl`, (conflict-free) map union, `nub_right`, …
- Accessor functions to the data of the targets in the target fields
- actions are a means to define artifacts
  - function returning a map of the output artifacts
  - inputs: stage of input artifacts, command vector, environment, expected outputs *(as well as execution properties, timeout-related information, …)*
  - mathematical function $\rightsquigarrow$ intensional equality on artifacts

Equality: Intensional versus Extensional

```
$ cat TARGETS
{ "foo":
  { "type": "generic"
  , "outs": ["out.txt"]
  , "cmds": ["echo Hello World > out.txt"]
  }
, "bar":
  { "type": "generic"
  , "outs": ["out.txt"]
  , "cmds": ["echo Hello World > out.txt"]
  }
, "baz":
  { "type": "generic"
  , "outs": ["out.txt"]
  , "cmds": ["echo -n Hello > out.txt && echo ' World' >> out.txt"]
  }
, "foo upper":
  { "type": "generic"
  , "deps": ["foo"]
  , "outs": ["upper.txt"]
  , "cmds": ["cat out.txt | tr a-z A-Z > upper.txt"]
  }
, "bar upper":
  { "type": "generic"
  , "deps": ["bar"]
  , "outs": ["upper.txt"]
  , "cmds": ["cat out.txt | tr a-z A-Z > upper.txt"]
  }
, "baz upper":
  { "type": "generic"
  , "deps": ["baz"]
  , "outs": ["upper.txt"]
  , "cmds": ["cat out.txt | tr a-z A-Z > upper.txt"]
  }
, "ALL":
  { "type": "install"
  , "files":
    {"foo.txt": "foo upper", "bar.txt": "bar upper", "baz.txt": "baz upper"}
  }
}
$
```

Equality: Intensional versus Extensional

```
$ cat TARGETS && just build -J 1
{ "foo":
  { "type": "generic"
  , "outs": ["out.txt"]
  , "cmds": ["echo Hello World > out.txt"]
  }
, "bar":
  { "type": "generic"
  , "outs": ["out.txt"]
  , "cmds": ["echo Hello World > out.txt"]
  }
, "baz":
  { "type": "generic"
  , "outs": ["out.txt"]
  , "cmds": ["echo -n Hello > out.txt && echo ' World' >> out.txt"]
  }
, "foo upper":
  { "type": "generic"
  , "deps": ["foo"]
  , "outs": ["upper.txt"]
  , "cmds": ["cat out.txt | tr a-z A-Z > upper.txt"]
  }
, "bar upper":
  { "type": "generic"
  , "deps": ["bar"]
  , "outs": ["upper.txt"]
  , "cmds": ["cat out.txt | tr a-z A-Z > upper.txt"]
  }
, "baz upper":
  { "type": "generic"
  , "deps": ["baz"]
  , "outs": ["upper.txt"]
  , "cmds": ["cat out.txt | tr a-z A-Z > upper.txt"]
  }
, "ALL":
  { "type": "install"
  , "files":
    {"foo.txt": "foo upper", "bar.txt": "bar upper", "baz.txt": "baz upper"}
  }
}
INFO: Requested target is [["@","","","ALL"],{}]
INFO: Analysed target [["@","","","ALL"],{}]
INFO: Discovered 4 actions, 0 trees, 0 blobs
INFO: Building [["@","","","ALL"],{}].
INFO: Processed 4 actions, 1 cache hits.
INFO: Artifacts built, logical paths are:
       bar.txt [4e3dffe834ac70600a7cb71fbc1f6a694c9d041f:12:f]
       baz.txt [4e3dffe834ac70600a7cb71fbc1f6a694c9d041f:12:f]
       foo.txt [4e3dffe834ac70600a7cb71fbc1f6a694c9d041f:12:f]
$
```

Reproducible Builds

- We consider build actions as functions—should also behave as such!

⇝ Reproducible builds (see also `https://reproducible-builds.org/`)

- Why
  - Independent verification possible that the binaries were built from those sources
  - Reconstruct the precise binary you used at a particular point in time
  - Also: more cache hits (if actions fall out of cache; formatting, comments, …)
- How
  - Don't include time stamps (other than `SOURCE_DATE_EPOCH`), working directory, hostname, user name, …
  - Don't rely on `readdir` order, non-specified behaviour, race conditions, …
  - Fix your enviroment: dependencies, tool chains, …—or bootstrap them!
- Also verify: `just rebuild`

Rebuilding

```
$ cat TARGETS
{ "":
  { "type": ["@", "rules", "CC", "binary"]
  , "name": ["hello"]
  , "srcs": ["main.cpp", "version.cpp", "greet.cpp"]
  , "private-hdrs": ["version.hpp", "greet.hpp"]
  }
}
$
```

Rebuilding

```
$ cat TARGETS
{ "":
  { "type": ["@", "rules", "CC", "binary"]
  , "name": ["hello"]
  , "srcs": ["main.cpp", "version.cpp", "greet.cpp"]
  , "private-hdrs": ["version.hpp", "greet.hpp"]
  }
}
$

$ just-mr build
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","build","-C","/worker/build/62bec0ed32723571/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/cd/e7ff54cc85bad8b8f5b1fedc42d1c03e884b02"]
INFO: Requested target is [["@","","",""],{}]
INFO: Analysed target [["@","","",""],{}]
INFO: Discovered 4 actions, 1 trees, 0 blobs
INFO: Building [["@","","",""],{}].
INFO: Processed 4 actions, 0 cache hits.
INFO: Artifacts built, logical paths are:
       hello [2f47037aef458e6a8cc131b865ec795041922e0e:17616:x]
$
```

Rebuilding

```
$ just-mr build
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","build","-C","/worker/build/62bec0ed32723571/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/cd/e7ff54cc85bad8b8f5b1fedc42d1c03e884b02"]
INFO: Requested target is [["@","","",""],{}]
INFO: Analysed target [["@","","",""],{}]
INFO: Discovered 4 actions, 1 trees, 0 blobs
INFO: Building [["@","","",""],{}].
INFO: Processed 4 actions, 0 cache hits.
INFO: Artifacts built, logical paths are:
        hello [2f47037aef458e6a8cc131b865ec795041922e0e:17616:x]
$
```

Rebuilding

```
$ just-mr build
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","build","-C","/worker/build/62bec0ed32723571/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/cd/e7ff54cc85bad8b8f5b1fedc42d1c03e884b02"]
INFO: Requested target is [["@","","",""],{}]
INFO: Analysed target [["@","","",""],{}]
INFO: Discovered 4 actions, 1 trees, 0 blobs
INFO: Building [["@","","",""],{}].
INFO: Processed 4 actions, 0 cache hits.
INFO: Artifacts built, logical paths are:
       hello [2f47037aef458e6a8cc131b865ec795041922e0e:17616:x]
$

$ just-mr rebuild
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","rebuild","-C","/worker/build/62bec0ed32723571/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/cd/e7ff54cc85bad8b8f5b1fedc42d1c03e884b02"]
INFO: Requested target is [["@","","",""],{}]
INFO: Analysed target [["@","","",""],{}]
INFO: Discovered 4 actions, 1 trees, 0 blobs
INFO: Rebuilding [["@","","",""],{}].
INFO: 4 actions compared with cache.
INFO: Artifacts built, logical paths are:
       hello [2f47037aef458e6a8cc131b865ec795041922e0e:17616:x]
$
```

Rebuilding

```
$ just-mr build
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","build","-C","/worker/build/62bec0ed32723571/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/cd/e7ff54cc85bad8b8f5b1fedc42d1c03e884b02"]
INFO: Requested target is [["@","","",""],{}]
INFO: Analysed target [["@","","",""],{}]
INFO: Discovered 4 actions, 1 trees, 0 blobs
INFO: Building [["@","","",""],{}].
INFO: Processed 4 actions, 0 cache hits.
INFO: Artifacts built, logical paths are:
       hello [2f47037aef458e6a8cc131b865ec795041922e0e:17616:x]
$

$ just-mr rebuild -L '["env", "--", "FAKETIME=1970-01-01 00:00:00", "LD_PRELOAD='$PREFIX'/lib/libfaketime.so.1"]'
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","rebuild","-C","/worker/build/62bec0ed32723571/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/cd/e7ff54cc85bad8b8f5b1fedc42d1c03e884b02","-L"
INFO: Requested target is [["@","","",""],{}]
INFO: Analysed target [["@","","",""],{}]
INFO: Discovered 4 actions, 1 trees, 0 blobs
INFO: Rebuilding [["@","","",""],{}].
WARN: Found flaky action:
      - id: afad430115d04a713e0de06770ca63e43f2f8dfe
      - cmd: ["c++","-I","work","-isystem","include","-c","work/version.cpp","-o","work/version.o"]
      - output 'work/version.o' differs:
        - [b951c2204c8e9a083d869817493acb28dc10cb0d:2792:f] (rebuilt)
        - [55f0079885057b199446bceb6b82fe0fe7f21def:2792:f] (cached)
INFO: 3 actions compared with cache, 1 flaky actions found (0 of which tainted), no cache entry found for 1 actions.
INFO: Artifacts built, logical paths are:
       hello [4095831467e82db7f2980b620deecf8f44dcc813:17616:x]
$
```

Rebuilding

```
$ just-mr build
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","build","-C","/worker/build/62bec0ed32723571/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/cd/e7ff54cc85bad8b8f5b1fedc42d1c03e884b02"]
INFO: Requested target is [["@","","",""],{}]
INFO: Analysed target [["@","","",""],{}]
INFO: Discovered 4 actions, 1 trees, 0 blobs
INFO: Building [["@","","",""],{}].
INFO: Processed 4 actions, 0 cache hits.
INFO: Artifacts built, logical paths are:
       hello [2f47037aef458e6a8cc131b865ec795041922e0e:17616:x]
$

$ just-mr rebuild -L '["env", "--", "FAKETIME=1970-01-01 00:00:00", "LD_PRELOAD='$PREFIX'/lib/libfaketime.so.1"]' --dump-flaky flaky.json
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","rebuild","-C","/worker/build/62bec0ed32723571/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/cd/e7ff54cc85bad8b8f5b1fedc42d1c03e884b02","-L"
INFO: Requested target is [["@","","",""],{}]
INFO: Analysed target [["@","","",""],{}]
INFO: Discovered 4 actions, 1 trees, 0 blobs
INFO: Rebuilding [["@","","",""],{}].
WARN: Found flaky action:
      - id: afad430115d04a713e0de06770ca63e43f2f8dfe
      - cmd: ["c++","-I","work","-isystem","include","-c","work/version.cpp","-o","work/version.o"]
      - output 'work/version.o' differs:
        - [b951c2204c8e9a083d869817493acb28dc10cb0d:2792:f] (rebuilt)
        - [55f0079885057b199446bceb6b82fe0fe7f21def:2792:f] (cached)
INFO: 3 actions compared with cache, 1 flaky actions found (0 of which tainted), no cache entry found for 1 actions.
INFO: Artifacts built, logical paths are:
       hello [4095831467e82db7f2980b620deecf8f44dcc813:17616:x]
$
```

Rebuilding (cont'd)

```
$ cat flaky.json
{
  "cache misses": [
    "f94e0c38ca48c0ae590b81db7075537f8e480d69"
  ],
  "flaky actions": {
    "afad430115d04a713e0de06770ca63e43f2f8dfe": {
      "work/version.o": {
        "cached": {
          "file_type": "f",
          "id": "55f0079885057b199446bceb6b82fe0fe7f21def",
          "size": 2792
        },
        "rebuilt": {
          "file_type": "f",
          "id": "b951c2204c8e9a083d869817493acb28dc10cb0d",
          "size": 2792
        }
      }
    }
  }
}$
```

Rebuilding (cont'd)

```
$ cat flaky.json
{
  "cache misses": [
    "f94e0c38ca48c0ae590b81db7075537f8e480d69"
  ],
  "flaky actions": {
    "afad430115d04a713e0de06770ca63e43f2f8dfe": {
      "work/version.o": {
        "cached": {
          "file_type": "f",
          "id": "55f0079885057b199446bceb6b82fe0fe7f21def",
          "size": 2792
        },
        "rebuilt": {
          "file_type": "f",
          "id": "b951c2204c8e9a083d869817493acb28dc10cb0d",
          "size": 2792
        }
      }
    }
  }
}$
```

```
$ for b in `jq -rM '."flaky actions" | .[] | .[] | .[] | .id ' flaky.json`
      do just install-cas $b | od -t x1 > blob-$b
  done
  diff -u blob*
--- blob-55f0079885057b199446bceb6b82fe0fe7f21def   2023-07-14 11:10:46.393008231 +0000
+++ blob-b951c2204c8e9a083d869817493acb28dc10cb0d   2023-07-14 11:10:46.425008450 +0000
@@ -17,8 +17,8 @@
 0000400 89 c7 e8 00 00 00 00 48 89 d8 48 89 c7 e8 00 00
 0000420 00 00 48 8b 85 68 fe ff ff 48 8b 5d f8 c9 c3 48
 0000440 65 6c 6c 6f 20 57 6f 72 6c 64 20 31 2e 30 00 2c
-0000460 20 28 63 29 20 00 4a 75 6c 20 31 34 20 32 30 32
-0000500 33 00 20 45 78 61 6d 70 6c 65 2e 63 6f 6d 00 ff
+0000460 20 28 63 29 20 00 4a 61 6e 20 20 31 20 31 39 37
+0000500 30 00 20 45 78 61 6d 70 6c 65 2e 63 6f 6d 00 ff
 0000520 ff 01 0e 1d 05 00 00 36 61 a9 01 00 c1 01 05 00
 0000540 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0000560 00 47 43 43 3a 20 28 44 65 62 69 61 6e 20 31 32
$
```

Rebuilding (cont'd)

```
$ cat flaky.json
{
  "cache misses": [
    "f94e0c38ca48c0ae590b81db7075537f8e480d69"
  ],
  "flaky actions": {
    "afad430115d04a713e0de06770ca63e43f2f8dfe": {
      "work/version.o": {
        "cached": {
          "file_type": "f",
          "id": "55f0079885057b199446bceb6b82fe0fe7f21def",
          "size": 2792
        },
        "rebuilt": {
          "file_type": "f",
          "id": "b951c2204c8e9a083d869817493acb28dc10cb0d",
          "size": 2792
        }
      }
    }
  }
}$
```

```
$ for b in `jq -rM '."flaky actions" | .[] | .[] | .[] | .id ' flaky.json`
      do just install-cas $b | od -t x1 > blob-$b
  done
  diff -u blob*
--- blob-55f0079885057b199446bceb6b82fe0fe7f21def   2023-07-14 11:10:46.393008231 +0000
+++ blob-b951c2204c8e9a083d869817493acb28dc10cb0d   2023-07-14 11:10:46.425008450 +0000
@@ -17,8 +17,8 @@
 0000400 89 c7 e8 00 00 00 48 89 d8 48 89 c7 e8 00 00
 0000420 00 00 48 8b 85 68 fe ff ff 48 8b 5d f8 c9 c3 48
 0000440 65 6c 6c 6f 20 57 6f 72 6c 64 20 31 2e 30 00 2c
-0000460 20 28 63 29 20 00 4a 75 6c 20 31 34 20 32 30 32
-0000500 33 00 20 45 78 61 6d 70 6c 65 2e 63 6f 6d 00 ff
+0000460 20 28 63 29 20 00 4a 61 6e 20 20 31 20 31 39 37
+0000500 30 00 20 45 78 61 6d 70 6c 65 2e 63 6f 6d 00 ff
 0000520 ff 01 0e 1d 05 00 00 36 61 a9 01 00 c1 01 05 00
 0000540 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0000560 00 47 43 43 3a 20 28 44 65 62 69 61 6e 20 31 32
$
```

```
$ cat version.cpp
#include "version.hpp"

#include <sstream>

auto version() -> std::string {
  std::ostringstream vstr{};
  vstr << "Hello World 1.0";
  vstr << ", (c) " << &__DATE__[7] << " Example.com";
  return vstr.str();
}
$
```

Non-Build Actions

- Not everything we build are pure functions, e.g., tests
- Still similar: we build the test report
- However, we expect tests to fail
  …but still want the test log and run the other tests ⇝ don't abort the build
- … even for spurious reasons ⇝ don't cache failures
- Optionally want to check for flakyness ⇝ support non inspecting the cache

⇝ Allow actions to declare

- `"may_fail"`: continue on failure, mark as failed, don't cache
- `"no_cache"`: run unconditionally without inspecting the cache

but mark those special actions as "tainted", as well as everything depending

Sources

- https://github.com/just-buildsystem/justbuild
- https://gitee.com/justbuild/justbuild
- License: Apache 2.0