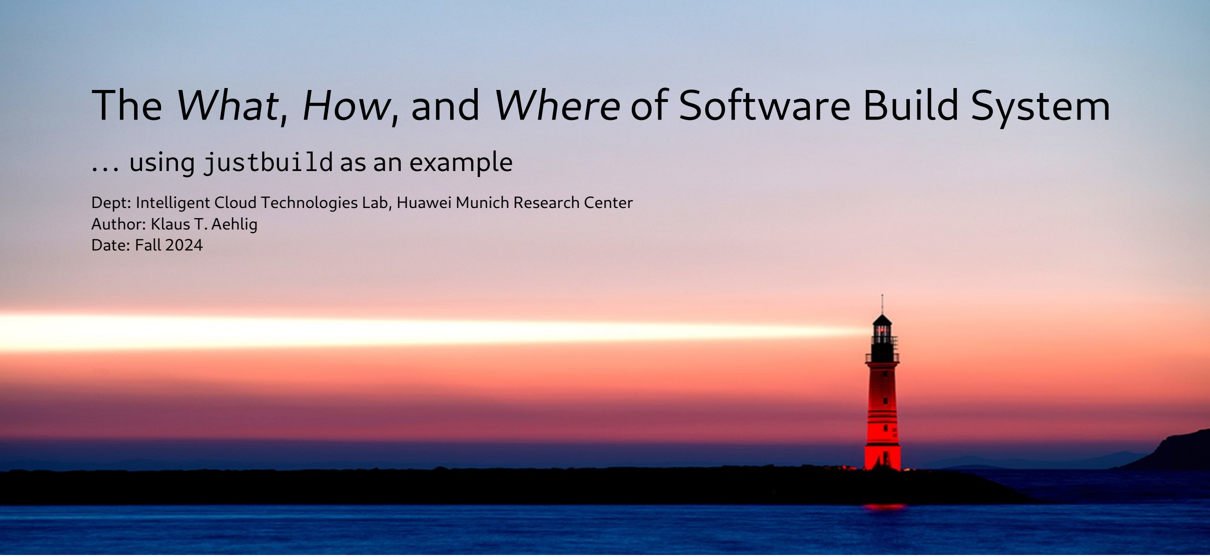


The *What*, *How*, and *Where* of Software Build System

... using `justbuild` as an example

Dept: Intelligent Cloud Technologies Lab, Huawei Munich Research Center
Author: Klaus T. Aehlig
Date: Fall 2024

The What, Why, and Where Phases

The History of every major Galactic Civilization tends to pass through three distinct and recognizable phases, those of Survival, Inquiry and Sophistication, otherwise known as the How, Why and Where phases.
For instance, the first phase is characterized by the question "How can we eat?" the second by the question "Why do we eat?" and the third by the question "Where shall we have lunch?".

Douglas Adams, The hitchhiker's guide to the galaxy

What? How? Where?

- What is defined? (extensional)

What? How? Where?

- What is defined? (extensional)
  - a mathematical object, e. g., a group

What? How? Where?

- What is defined? (extensional)
    - a mathematical object, e. g., a group
- How is it defined? (intensional)

What? How? Where?

- What is defined? (extensional)
  - a mathematical object, e. g., a group
- How is it defined? (intensional)
  - Different formulae defining the same object:
    “A monoid such that $\forall a \exists b. ab = e = ba$”
    “A monoid such that $\forall a \exists b. ab = e$”

What? How? Where?

- What is defined? (extensional)
  - a mathematical object, e. g., a group
- How is it defined? (intensional)
  - Different formulae defining the same object:
    "A monoid such that $\forall a \exists b. ab = e = ba$"
    "A monoid such that $\forall a \exists b. ab = e$"
- Where is it defined?
  - "An object with the properties of Defintion 1.2.3 of the lecture notes"

What? How? Where?

- What is defined? (extensional)
  - a mathematical object, e. g., a group
  - software build: the contents of a file, e. g., "`hello world\n`"
- How is it defined? (intensional)
  - Different formulae defining the same object:
    "A monoid such that $\forall a \exists b. ab = e = ba$"
    "A monoid such that $\forall a \exists b. ab = e$"
- Where is it defined?
  - "An object with the properties of Defintion 1.2.3 of the lecture notes"

What? How? Where?

- What is defined? (extensional)
  - a mathematical object, e. g., a group
  - software build: the contents of a file, e. g., `"hello world\n"`
- How is it defined? (intensional)
  - Different formulae defining the same object:
    "A monoid such that $\forall a \exists b. ab = e = ba$"
    "A monoid such that $\forall a \exists b. ab = e$"
  - "Output `hello.txt` of running `sh -c 'echo hello world > hello.txt'`"
    "Output `foo` of running `sh -c 'echo Hello World | tr A-Z a-z > foo'`"
- Where is it defined?
  - "An object with the properties of Defintion 1.2.3 of the lecture notes"

What? How? Where?

- What is defined? (extensional)
  - a mathematical object, e. g., a group
  - software build: the contents of a file, e. g., "`hello world\n`"
- How is it defined? (intensional)
  - Different formulae defining the same object:
    "A monoid such that $\forall a \exists b . ab = e = ba$"
    "A monoid such that $\forall a \exists b . ab = e$"
  - "Output `hello.txt` of running `sh -c 'echo hello world > hello.txt'`"
    "Output `foo` of running `sh -c 'echo Hello World | tr A-Z a-z > foo'`"
- Where is it defined?
  - "An object with the properties of Defintion 1.2.3 of the lecture notes"
  - `make`: "The file that will be created at location `hello.txt`"
    `bazel`: "Target `//:hello` hence artifact `//:hello.txt`"
    implies overlapping-output check, module containment check, . . .

Names for Artifacts

- extensional

Names for Artifacts

- extensional
    - The contents of the file itself

Names for Artifacts

- extensional
    - The contents of the file itself
      … maybe prefixed with a tag to mark it as blob

Names for Artifacts

- extensional
  - The contents of the file itself
    ... maybe prefixed with a tag to mark it as blob
    ... or a cryptographic hash of the tagged content (a. k. a. the `git` blob identifier)

Names for Artifacts

- extensional
    - The contents of the file itself
      … maybe prefixed with a tag to mark it as blob
      … or a cryptographic hash of the tagged content (a. k. a. the `git` blob identifier)
- intensional

Names for Artifacts

- extensional
  - The contents of the file itself
    ... maybe prefixed with a tag to mark it as blob
    ... or a cryptographic hash of the tagged content (a. k. a. the `git` blob identifier)
- intensional
  - the extensional name (e. g., a committed source file)

Names for Artifacts

- extensional
  - The contents of the file itself
    ... maybe prefixed with a tag to mark it as blob
    ... or a cryptographic hash of the tagged content (a. k. a. the `git` blob identifier)
- intensional
  - the extensional name (e. g., a committed source file)
  - a specific (given by a string) output of an action given by
    - inputs: map from logical paths to itensional artifact names
    - command argv as list of strings
    - environment as map from strings to strings
    - execution properties, timeout, ...

Names for Artifacts

- extensional
  - The contents of the file itself

    … maybe prefixed with a tag to mark it as blob

    … or a cryptographic hash of the tagged content (a. k. a. the `git` blob identifier)
- intensional
  - the extensional name (e. g., a committed source file)
  - a specific (given by a string) output of an action given by
    - inputs: map from logical paths to itensional artifact names
    - command argv as list of strings
    - environment as map from strings to strings
    - execution properties, timeout, …

    … or a hash of the canonical serialisation thereof

Names for Artifacts (examples)

```
$ cat TARGETS
{ "":
  { "type": "install"
  , "files":
    { "extensional": ["FILE", null, "hello.txt"]
    , "intensional: action-1": "echo"
    , "intensional: action-2": "tr"
    }
  }
, "echo":
  { "type": "generic"
  , "outs": ["hello.txt"]
  , "cmds": ["echo hello world > hello.txt"]
  }
, "tr":
  { "type": "generic"
  , "outs": ["foo"]
  , "cmds": ["echo Hello World | tr A-Z a-z > foo"]
  }
}
$
```

Names for Artifacts (examples)

```
$ just-mr --log-limit 0 analyse --dump-plain-graph graph.json --dump-artifacts-to-build artifacts.json && cat artifacts.json
{
  "extensional": {
    "data": {
      "file_type": "f",
      "id": "3b18e512dba79e4c8300dd08aeb37f8e728b8dad",
      "size": 12
    },
    "type": "KNOWN"
  },
  "intensional: action-1": {
    "data": {
      "id": "81144019e16ed507b24e98763852721d1717fd749a9700732adefa22439c878d",
      "path": "hello.txt"
    },
    "type": "ACTION"
  },
  "intensional: action-2": {
    "data": {
      "id": "e60453859488bd5b2210e12b31879df7bd7e7195b826fbb690fec62c1923bafa",
      "path": "foo"
    },
    "type": "ACTION"
  }
}
$
```

Names for Artifacts (examples)

```
$ cat graph.json
{
  "actions": {
    "81144019e16ed507b24e98763852721d1717fd749a9700732adefa22439c878d": {
      "command": [
        "sh",
        "-c",
        "echo hello world > hello.txt\n"
      ],
      "output": [
        "hello.txt"
      ]
    },
    "e60453859488bd5b2210e12b31879df7bd7e7195b826fbb690fec62c1923bafa": {
      "command": [
        "sh",
        "-c",
        "echo Hello World | tr A-Z a-z > foo\n"
      ],
      "output": [
        "foo"
      ]
    }
  },
  "blobs": [],
  "trees": {}
}
$
```

Names for Artifacts (examples)

```
$ just-mr build
INFO: Performing repositories setup
INFO: Found 1 repositories to set up
INFO: Setup finished, exec ["just","build","-C","/worker/build/62bb828623cd3b74/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/46/a6c90d2e919142492edf200d8d356e323cf43e"]
INFO: Requested target is [["@","","",""],{}]
INFO: Analysed target [["@","","",""],{}]
INFO: Discovered 2 actions, 0 trees, 0 blobs
INFO: Building [["@","","",""],{}].
INFO: Processed 2 actions, 0 cache hits.
INFO: Artifacts built, logical paths are:
        extensional [3b18e512dba79e4c8300dd08aeb37f8e728b8dad:12:f]
        intensional: action-1 [3b18e512dba79e4c8300dd08aeb37f8e728b8dad:12:f]
        intensional: action-2 [3b18e512dba79e4c8300dd08aeb37f8e728b8dad:12:f]
$
```

```
$ just install-cas 3b18e512dba79e4c8300dd08aeb37f8e728b8dad
hello world
$
```

Targets

- Software is *not* structured around individual artifacts!
  Basic concepts: library, binary, . . .

Targets

- Software is *not* structured around individual artifacts!
  Basic concepts: library, binary, . . .

⇝ Use (user-defined) “rules” to define targets
  from sources, headers, dependencies, . . .

Targets

- Software is *not* structured around individual artifacts!
  Basic concepts: library, binary, . . .

⇝ Use (user-defined) “rules” to define targets
  from sources, headers, dependencies, . . .

- Rules are expressions for primitive-recursive functions,
  but usually the actual complexity is a lot less (typically linear)

Targets

- Software is *not* structured around individual artifacts!
  Basic concepts: library, binary, . . .

⇝ Use (user-defined) “rules” to define targets
from sources, headers, dependencies, . . .

- Rules are expressions for primitive-recursive functions,
  but usually the actual complexity is a lot less (typically linear)

⇝ Can evalute while analysing the targets

Targets (cont'd)

⇝ Can evalute while analysing the targets

Targets (cont'd)

⇝ Can evalute while analysing the targets

- We obtain
    - Map from logical paths (e. g., `libfoo.a`) to intensional artifact names

Targets (cont'd)

⇝ Can evalute while analysing the targets

- We obtain
  - Map from logical paths (e. g., `libfoo.a`) to intensional artifact names
  - another such map (e. g., `foo.h`)

Targets (cont'd)

⇝ Can evalute while analysing the targets

- We obtain
  - Map from logical paths (e. g., `libfoo.a`) to intensional artifact names
  - another such map (e. g., `foo.h`)
  - any other information (keyed by logical names)
    Headers of transitive dependencies, transitive link dependencies, . . .
    (including intensional artifact names of artifacts needed to build against, but not to be installed with this library)

Targets (cont'd)

⇝ Can evalute while analysing the targets

- We obtain
  - Map from logical paths (e. g., `libfoo.a`) to intensional artifact names
  - another such map (e. g., `foo.h`)
  - any other information (keyed by logical names)
    Headers of transitive dependencies, transitive link dependencies, . . .
    (including intensional artifact names of artifacts needed to build against, but not to be installed with this library)
- And that's the only thing a consuming target can see!
  (no place of definition, no provienence, not the defintion (i.e., no "reflection"))

Targets (cont'd)

⇝ Can evalute while analysing the targets

- We obtain
    - Map from logical paths (e. g., `libfoo.a`) to intensional artifact names
    - another such map (e. g., `foo.h`)
    - any other information (keyed by logical names)
      Headers of transitive dependencies, transitive link dependencies, ...
      (including intensional artifact names of artifacts needed to build against, but not to be installed with this library)
- And that's the only thing a consuming target can see!
  (no place of definition, no provienence, not the defintion (i.e., no "reflection"))
- Note: during that evaluation we need equality to detect staging conflicts: different files at the same location for actions—same file is OK (common dep)

Targets (example)

```
$ cat TARGETS
{ "foo":
  { "type": ["@", "rules", "CC", "library"]
  , "name": ["foo"]
  , "hdrs": ["foo.hpp"]
  , "srcs": ["foo.cpp"]
  }
, "bar":
  { "type": ["@", "rules", "CC", "library"]
  , "name": ["bar"]
  , "hdrs": ["bar.hpp"]
  , "srcs": ["bar.cpp"]
  , "deps": ["foo"]
  }
}
$
```

Targets (example)

```
$ just-mr analyse --dump-plain-graph graph.json foo
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","analyse","-C","/worker/build/62a7b58f6bbd080b/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/65/4a76578e97c71d9d3e862c9634c7a70dade92f","--dump-plain-graph","graph.json","foo"]
INFO: Requested target is [["@","","","foo"],{}]
INFO: Analysed target [["@","","","foo"],{}]
INFO: Dumping action graph to file graph.json.
INFO: Discovered 2 actions, 1 trees, 0 blobs
INFO: Result of target [["@","","","foo"],{}]: {
        "artifacts": {
          "libfoo.a": {"data":{"id":"be8956137a1fc8938c9eb55858a88d05260acafa25f886800b871a10529f5a6e","path":"work/libfoo.a"},"type":"ACTION"}
        },
        "provides": {
          "compile-args": [
          ],
          "compile-deps": {
          },
          "debug-hdrs": {
          },
          "debug-srcs": {
          },
          "link-args": [
            "libfoo.a"
          ],
          "link-deps": {
          },
          "lint": [
          ],
          "package": {
            "cflags-files": {},
            "ldflags-files": {},
            "name": "foo"
          }
        },
        "runfiles": {
          "foo.hpp": {"data":{"file_type":"f","id":"6bf31ad480f47c568bebcb638f3fd3e037ca5870","size":53},"type":"KNOWN"}
        }
      }
$
```

Targets (example)

```
$ cat graph.json
{
  "actions": {
    "9343ad77e828ca97cee825cf51939c335d469fdf4dd613dfe57980462101ccf4": {
      "command": [
        "c++",
        "-I",
        "work",
        "-isystem",
        "include",
        "-c",
        "work/foo.cpp",
        "-o",
        "work/foo.o"
      ],
      "env": {
        "PATH": "/bin:/usr/bin"
      },
      "input": {
        "include": {
          "data": {
            "id": "44136fa355b3678a1146ad16f7e8649e94fb4fc21fe77e8310c060f61caaff8a"
          },
          "type": "TREE"
        },
        "work/foo.cpp": {
          "data": {
            "file_type": "f",
            "id": "223e5e3701f42fe5584aa4c6543437b758328166",
            "size": 111
          },
          "type": "KNOWN"
        },
        "work/foo.hpp": {
          "data": {
            "file_type": "f",
            "id": "6bf31ad480f47c568bebcb638f3fd3e037ca5870",
            "size": 53
          },
          "type": "KNOWN"
        }
      },
      "output": [
        "work/foo.o"
      ]
    },
    "be8956137a1fc8938c9eb55858a88d05260acafa25f886800b871a10529f5a6e": {
      "command": [
        "ar",
        "cqs",
        "work/libfoo.a",
        "work/foo.o"
      ],
      "env": {
        "PATH": "/bin:/usr/bin"
      },
      "input": {
        "work/foo.o": {
          "data": {
            "id": "9343ad77e828ca97cee825cf51939c335d469fdf4dd613dfe57980462101ccf4",
            "path": "work/foo.o"
          },
          "type": "ACTION"
        }
      },
      "output": [
        "work/libfoo.a"
      ]
    }
  },
  "blobs": [],
  "trees": {
    "44136fa355b3678a1146ad16f7e8649e94fb4fc21fe77e8310c060f61caaff8a": {}
  }
}
$
```

Targets (example)

```
$ just-mr analyse bar
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","analyse","-C","/worker/build/62a7b58f6bbd080b/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/65/4a76578e97c71d9d3e862c9634c7a70dade92f","bar"]
INFO: Requested target is [["@","","","bar"],{}]
INFO: Analysed target [["@","","","bar"],{}]
INFO: Result of target [["@","","","bar"],{}]: {
        "artifacts": {
          "libbar.a": {"data":{"id":"aac7a11cc78afefb0c8df28ed6bb2904ad9381511814c2ebb4ef2b6b90991e5c","path":"work/libbar.a"},"type":"ACTION"}
        },
        "provides": {
          "compile-args": [
          ],
          "compile-deps": {
            "foo.hpp": {"data":{"file_type":"f","id":"6bf31ad480f47c568bebcb638f3fd3e037ca5870","size":53},"type":"KNOWN"}
          },
          "debug-hdrs": {
          },
          "debug-srcs": {
          },
          "link-args": [
            "libbar.a",
            "libfoo.a"
          ],
          "link-deps": {
            "libfoo.a": {"data":{"id":"be8956137a1fc8938c9eb55858a88d05260acafa25f886800b871a10529f5a6e","path":"work/libfoo.a"},"type":"ACTION"}
          },
          "lint": [
          ],
          "package": {
            "cflags-files": {},
            "ldflags-files": {},
            "name": "bar"
          }
        },
        "runfiles": {
          "bar.hpp": {"data":{"file_type":"f","id":"427bf71c95adb1f0db3c0bb9c6044fbc594528d4","size":53},"type":"KNOWN"}
        }
      }
$
```

Building

- When building, the actions are run in topological order and results are recorded

⇝ We’re recursively projecting intensional names to extensional ones

Building

- When building, the actions are run in topological order and results are recorded

⇝ We’re recursively projecting intensional names to extensional ones

- Can also be mapped over targets

Building

- When building, the actions are run in topological order and results are recorded

⇝ We're recursively projecting intensional names to extensional ones

- Can also be mapped over targets
- Projecting only certain artifacts preserves intensional equality
  (if applied consistently)

Multi Repositories

- Not everybody has a mono-repo; often dependencies come from different sources *(at least rules are typically shared between projects)*

Multi Repositories

- Not everybody has a mono-repo; often dependencies come from different sources *(at least rules are typically shared between projects)*
- Approach
  - Build descriptions may use open repository names to refer to dependencies
  - Global configuration with all involved repositories
    ... including the mapping of (repo, local name) pairs to the actual dependency repo

Multi Repositories

- Not everybody has a mono-repo; often dependencies come from different sources *(at least rules are typically shared between projects)*
- Approach
  - Build descriptions may use open repository names to refer to dependencies
  - Global configuration with all involved repositories
    ... including the mapping of (repo, local name) pairs to the actual dependency repo
- Need to use that graph to resolve local names—but otherwise nothing changes! *(Caring about how things are defined anyway, not where)*

Multi Repositories

- Not everybody has a mono-repo; often dependencies come from different sources *(at least rules are typically shared between projects)*
- Approach
  - Build descriptions may use open repository names to refer to dependencies
  - Global configuration with all involved repositories
    . . . including the mapping of (repo, local name) pairs to the actual dependency repo
- Need to use that graph to resolve local names—but otherwise nothing changes! *(Caring about how things are defined anyway, not where)*
- However, now symbolic target names can be used for a meaningful description
  . . . as often the reachable repositories for a target are a small subset.

Target Descriptions in Multi Repositories

- The global repository graph forms a finite automaton
- Each of those repositories can be given by a file tree (hash)

Target Descriptions in Multi Repositories

- The global repository graph forms a finite automaton
- Each of those repositories can be given by a file tree (hash)

⇝ Take the minimal automaton (for a given start node)
with the tree as locally observable data
(again, the name or equality of symbolic target references does not matter!)

Target Descriptions in Multi Repositories

- The global repository graph forms a finite automaton
- Each of those repositories can be given by a file tree (hash)

⇝ Take the minimal automaton (for a given start node)
with the tree as locally observable data
(again, the name or equality of symbolic target references does not matter!)

- Together with the symbolic name (and configuration) this describes a target

Target Descriptions in Multi Repositories

- The global repository graph forms a finite automaton
- Each of those repositories can be given by a file tree (hash)

⇝ Take the minimal automaton (for a given start node)
with the tree as locally observable data
(again, the name or equality of symbolic target references does not matter!)

- Together with the symbolic name (and configuration) this describes a target
- Can store the extensional value of the target for that description as key

... because analysing a target is cheap, but not for free
(especially when bootstrapping deps)

Target Caching (example)

```
$ cat TARGETS
{ "foo":
  { "type": ["@", "rules", "CC", "library"]
  , "name": ["foo"]
  , "hdrs": ["foo.hpp"]
  , "srcs": ["foo.cpp"]
  }
, "exported foo": {"type": "export", "target": "foo"}
, "bar":
  { "type": ["@", "rules", "CC", "library"]
  , "name": ["bar"]
  , "hdrs": ["bar.hpp"]
  , "srcs": ["bar.cpp"]
  , "deps": ["exported foo"]
  }
}
$
```

Target Caching (example)

```
$ just-mr analyse bar
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","analyse","-C","/worker/build/62216bdf0811e1fe/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/db/1322880208a9b0a3d98c2e67ce04d02e32e793","bar"]
INFO: Requested target is [["@","","","bar"],{}]
INFO: Analysed target [["@","","","bar"],{}]
INFO: Export targets found: 0 cached, 1 uncached, 0 not eligible for caching
INFO: Result of target [["@","","","bar"],{}]: {
        "artifacts": {
          "libbar.a": {"data":{"id":"aac7a11cc78afefb0c8df28ed6bb2904ad9381511814c2ebb4ef2b6b90991e5c","path":"work/libbar.a"},"type":"ACTION"}
        },
        "provides": {
          "compile-args": [
          ],
          "compile-deps": {
            "foo.hpp": {"data":{"file_type":"f","id":"6bf31ad480f47c568bebcb638f3fd3e037ca5870","size":53},"type":"KNOWN"}
          },
          "debug-hdrs": {
          },
          "debug-srcs": {
          },
          "link-args": [
            "libbar.a",
            "libfoo.a"
          ],
          "link-deps": {
            "libfoo.a": {"data":{"id":"be8956137a1fc8938c9eb55858a88d05260acafa25f886800b871a10529f5a6e","path":"work/libfoo.a"},"type":"ACTION"}
          },
          "lint": [
          ],
          "package": {
            "cflags-files": {},
            "ldflags-files": {},
            "name": "bar"
          }
        },
        "runfiles": {
          "bar.hpp": {"data":{"file_type":"f","id":"427bf71c95adb1f0db3c0bb9c6044fbc594528d4","size":53},"type":"KNOWN"}
        }
      }
$
```

Target Caching (example)

```
$ just-mr build --log-limit 4 -f log -s bar
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","build","-C","/worker/build/62216bdf0811e1fe/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/db/1322880208a9b0a3d98c2e67ce04d02e32e793","--log-limit","4","-f","log","-s","bar"]
INFO: Requested target is [["@","","","bar"],{}]
PERF: Export target ["@","","","exported foo"] registered for caching: [e758b376f74f11712da6992b54c97954926621cf:132:f]
INFO: Analysed target [["@","","","bar"],{}]
INFO: Export targets found: 0 cached, 1 uncached, 0 not eligible for caching
INFO: Discovered 4 actions, 2 trees, 0 blobs
INFO: Building [["@","","","bar"],{}].
INFO: Processed 4 actions, 0 cache hits.
INFO: Artifacts built, logical paths are:
        libbar.a [b94c10968fefa3a1b19e2958b0b3834ab24370b4:2816:f]
        bar.hpp [427bf71c95adb1f0db3c0bb9c6044fbc594528d4:53:f]
INFO: Backing up artifacts of 1 export targets
$

$ just-mr install-cas $CACHE_KEY
INFO: Setup finished, exec ["just","install-cas","[e758b376f74f11712da6992b54c97954926621cf:132:f]"]
{
  "effective_config": "{}",
  "repo_key": "8c776b7f5fe76f313403547ca48d87b970310d29",
  "target_name": "[\"\",\"exported foo\"]"
}$
```

Target Caching (example)

```
$ just-mr install-cas $GRAPH_KEY
INFO: Setup finished, exec ["just","install-cas","8c776b7f5fe76f313403547ca48d87b970310d29"]
{
  "0": {
    "bindings": {
      "rules": "1"
    },
    "expression_file_name": "EXPRESSIONS",
    "expression_root": [
      "git tree",
      "a02b13b0cdaf2c6c3a1f0cd02c23a36d9db0384a"
    ],
    "rule_file_name": "RULES",
    "rule_root": [
      "git tree",
      "a02b13b0cdaf2c6c3a1f0cd02c23a36d9db0384a"
    ],
    "target_file_name": "TARGETS",
    "target_root": [
      "git tree",
      "a02b13b0cdaf2c6c3a1f0cd02c23a36d9db0384a"
    ],
    "workspace_root": [
      "git tree",
      "a02b13b0cdaf2c6c3a1f0cd02c23a36d9db0384a"
    ]
  },
  "1": {
    "bindings": {},
    "expression_file_name": "EXPRESSIONS",
    "expression_root": [
      "git tree",
      "ebcf2668f2f71029bd7e86824fc6022825cc560c"
    ],
    "rule_file_name": "RULES",
    "rule_root": [
      "git tree",
      "ebcf2668f2f71029bd7e86824fc6022825cc560c"
    ],
    "target_file_name": "TARGETS",
    "target_root": [
      "git tree",
      "ebcf2668f2f71029bd7e86824fc6022825cc560c"
    ],
    "workspace_root": [
      "git tree",
      "ebcf2668f2f71029bd7e86824fc6022825cc560c"
    ]
  }
}$
```

Target Caching (example)

```
$ just-mr analyse --log-limit 4 -f log --dump-plain-graph graph.json bar
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","analyse","-C","/worker/build/62216bdf0811e1fe/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/db/13...
INFO: Requested target is [["@","","","bar"],{}]
PERF: Export target ["@","","","exported foo"] taken from cache: [e758b376f74f11712da6992b54c97954926621cf:132:f] -> [1d6123e6c4aba0e781ad5eabda0b3379726fe...
INFO: Analysed target [["@","","","bar"],{}]
INFO: Export targets found: 1 cached, 0 uncached, 0 not eligible for caching
INFO: Dumping action graph to file graph.json.
INFO: Discovered 2 actions, 1 trees, 0 blobs
INFO: Result of target [["@","","","bar"],{}]: {
        "artifacts": {
          "libbar.a": {"data":{"id":"aac7a11cc78afefb0c8df28ed6bb2904ad9381511814c2ebb4ef2b6b90991e5c","path":"work/libbar.a"},"type":"ACTION"}
        },
        "provides": {
          "compile-args": [
          ],
          "compile-deps": {
            "foo.hpp": {"data":{"file_type":"f","id":"6bf31ad480f47c568bebcb638f3fd3e037ca5870","size":53},"type":"KNOWN"}
          },
          "debug-hdrs": {
          },
          "debug-srcs": {
          },
          "link-args": [
            "libbar.a",
            "libfoo.a"
          ],
          "link-deps": {
            "libfoo.a": {"data":{"file_type":"f","id":"6e4334d2748e6a942f434f5cbfa61c5ab6e37feb","size":2760},"type":"KNOWN"}
          },
          "lint": [
          ],
          "package": {
            "cflags-files": {},
            "ldflags-files": {},
            "name": "bar"
          }
        },
        "runfiles": {
          "bar.hpp": {"data":{"file_type":"f","id":"427bf71c95adb1f0db3c0bb9c6044fbc594528d4","size":53},"type":"KNOWN"}
        }
      }
$
```

Target Caching (example)

```
$ just-mr install-cas $CACHE_VALUE
INFO: Setup finished, exec ["just","install-cas","[1d6123e6c4aba0e781ad5eabda0b3379726fef52:2315:f]"]
{
  "artifacts": {
    "libfoo.a": {
      "data": {
        "file_type": "f",
        "id": "6e4334d2748e6a942f434f5cbfa61c5ab6e37feb",
        "size": 2760
      },
      "type": "KNOWN"
    }
  },
  "implied export targets": [
    "e758b376f74f11712da6992b54c97954926621cf"
  ],
  "provides": {
    "entry": "4b147c37be121dfa8365b904d3f0058817eaeb89c1ed8f8e5d0fe0b27debc483",
    "nodes": {
      "00cf3b618472e2c936630ee5230536bf29a845a077561eba5ed97000a7a2fc80": [
        "c2e5d2a5a547ec1b408edac5e1aecc86c53c8b73d78d631ba5e4d2254874bcf0"
      ],
      "021fb596db81e6d02bf3d2586ee3981fe519f275c0ac9ca76bbcf2ebb4097d96": {},
      "245843abef9e72e7efac30138a994bf6301e7e1d7d7042a33d42e863d2638811": [],
      "4b147c37be121dfa8365b904d3f0058817eaeb89c1ed8f8e5d0fe0b27debc483": {
        "compile-args": "245843abef9e72e7efac30138a994bf6301e7e1d7d7042a33d42e863d2638811",
        "compile-deps": "021fb596db81e6d02bf3d2586ee3981fe519f275c0ac9ca76bbcf2ebb4097d96",
        "debug-hdrs": "021fb596db81e6d02bf3d2586ee3981fe519f275c0ac9ca76bbcf2ebb4097d96",
        "debug-srcs": "021fb596db81e6d02bf3d2586ee3981fe519f275c0ac9ca76bbcf2ebb4097d96",
        "link-args": "00cf3b618472e2c936630ee5230536bf29a845a077561eba5ed97000a7a2fc80",
        "link-deps": "021fb596db81e6d02bf3d2586ee3981fe519f275c0ac9ca76bbcf2ebb4097d96",
        "lint": "245843abef9e72e7efac30138a994bf6301e7e1d7d7042a33d42e863d2638811",
        "package": "c732b09787cf8e52bd646af540e68ee0f4bd4b57e212e068b60ba7b69f57ebb7"
      },
      "b2213295d564916f89a6a42455567c87c3f480fcd7a1c15e220f17d7169a790b": "foo",
      "c2e5d2a5a547ec1b408edac5e1aecc86c53c8b73d78d631ba5e4d2254874bcf0": "libfoo.a",
      "c732b09787cf8e52bd646af540e68ee0f4bd4b57e212e068b60ba7b69f57ebb7": {
        "cflags-files": "021fb596db81e6d02bf3d2586ee3981fe519f275c0ac9ca76bbcf2ebb4097d96",
        "ldflags-files": "021fb596db81e6d02bf3d2586ee3981fe519f275c0ac9ca76bbcf2ebb4097d96",
        "name": "b2213295d564916f89a6a42455567c87c3f480fcd7a1c15e220f17d7169a790b"
      }
    },
    "provided_artifacts": [],
    "provided_nodes": [],
    "provided_results": []
  },
  "runfiles": {
    "foo.hpp": {
      "data": {
        "file_type": "f",
        "id": "6bf31ad480f47c568bebcb638f3fd3e037ca5870",
        "size": 53
      },
      "type": "KNOWN"
    }
  }
}$
```

Target Caching (example)

```
$ cat graph.json
{
  "actions": {
    "aac7a11cc78afefb0c8df28ed6bb2904ad9381511814c2ebb4ef2b6b90991e5c": {
      "command": [
        "ar",
        "cqs",
        "work/libbar.a",
        "work/bar.o"
      ],
      "env": {
        "PATH": "/bin:/usr/bin"
      },
      "input": {
        "work/bar.o": {
          "data": {
            "id": "f8a51437ed68eeb25b8f9a21649890a28f4c6c8b8ecbf419a9448ce4c2ce5c01",
            "path": "work/bar.o"
          },
          "type": "ACTION"
        }
      },
      "output": [
        "work/libbar.a"
      ]
    },
    "f8a51437ed68eeb25b8f9a21649890a28f4c6c8b8ecbf419a9448ce4c2ce5c01": {
      "command": [
        "c++",
        "-I",
        "work",
        "-isystem",
        "include",
        "-c",
        "work/bar.cpp",
        "-o",
        "work/bar.o"
      ],
      "env": {
        "PATH": "/bin:/usr/bin"
      },
      "input": {
        "include": {
          "data": {
            "id": "7d36579a8f63014a983891be885d2823a292838990b376d41894372bba625ac7"
          },
          "type": "TREE"
        },
        "work/bar.cpp": {
          "data": {
            "file_type": "f",
            "id": "a474d1e99ae6689e18c8fea424859388ffe0232e",
            "size": 140
          },
          "type": "KNOWN"
        },
        "work/bar.hpp": {
          "data": {
            "file_type": "f",
            "id": "427bf71c95adb1f0db3c0bb9c6044fbc594528d4",
            "size": 53
          },
          "type": "KNOWN"
        }
      },
      "output": [
        "work/bar.o"
      ]
    }
  },
  "blobs": [],
  "trees": {
    "7d36579a8f63014a983891be885d2823a292838990b376d41894372bba625ac7": {
      "foo.hpp": {
        "data": {
          "file_type": "f",
          "id": "6bf31ad480f47c568bebcb638f3fd3e037ca5870",
          "size": 53
        },
        "type": "KNOWN"
      }
    }
  }
}
```

Target Caching (example)

```
$ just-mr build --log-limit 4 -s bar
INFO: Performing repositories setup
INFO: Found 2 repositories to set up
INFO: Setup finished, exec ["just","build","-C","/worker/build/62216bdf0811e1fe/root/home/.cache/just/protocol-dependent/generation-0/git-sha1/casf/db/1322880208a9b0a3d98c2e67ce04d02e32e793","--log-limit","4","-s","bar"]
INFO: Requested target is [["@","","","bar"],{}]
PERF: Export target ["@","","","exported foo"] taken from cache: [e758b376f74f11712da6992b54c97954926621cf:132:f] -> [1d6123e6c4aba0e781ad5eabda0b3379726fef52:2315:f]
INFO: Analysed target [["@","","","bar"],{}]
INFO: Export targets found: 1 cached, 0 uncached, 0 not eligible for caching
INFO: Discovered 2 actions, 1 trees, 0 blobs
INFO: Building [["@","","","bar"],{}].
INFO: Processed 2 actions, 2 cache hits.
INFO: Artifacts built, logical paths are:
        libbar.a [b94c10968fefa3a1b19e2958b0b3834ab24370b4:2816:f]
        bar.hpp [427bf71c95adb1f0db3c0bb9c6044fbc594528d4:53:f]
$
```

Remote Execution

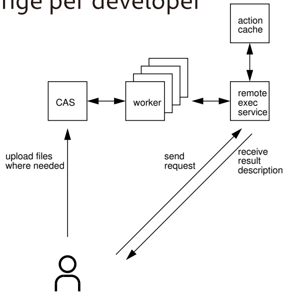
- As actions are completely described, can run on any machine *(with the correct external programs installed, if used)*

Remote Execution

- As actions are completely described, can run on any machine *(with the correct external programs installed, if used)*

⇝ Provide a service, that also caches the extensional input-output relation

  - Build once per change, not once per change per developer
  - Environment fixed once and for all *(No "works on my machine")*
  - Easy to share test logs, etc



Serving

- Tree identifier is a function of the typical repository-root descriptions: commit id, the result of unpacking an archive with a given blob id, . . .

Serving

- Tree identifier is a function of the typical repository-root descriptions: commit id, the result of unpacking an archive with a given blob id, . . .

$\rightsquigarrow$ Can provide this mapping as a service
(Of course, that service needs all the sources; but I need to store them anyway)

Serving

- Tree identifier is a function of the typical repository-root descriptions: commit id, the result of unpacking an archive with a given blob id, . . .

$\rightsquigarrow$ Can provide this mapping as a service
(Of course, that service needs all the sources; but I need to store them anyway)

- From only that can compute the cache key for a target

Serving

- Tree identifier is a function of the typical repository-root descriptions: commit id, the result of unpacking an archive with a given blob id, . . .

⇝ Can provide this mapping as a service
(Of course, that service needs all the sources; but I need to store them anyway)

- From only that can compute the cache key for a target

⇝ Can provide a service evaluating and caching the target
. . . and uploading the artifacts to the remote-execution service

Serving

- Tree identifier is a function of the typical repository-root descriptions: commit id, the result of unpacking an archive with a given blob id, . . .

⇝ Can provide this mapping as a service
(Of course, that service needs all the sources; but I need to store them anyway)

- From only that can compute the cache key for a target

⇝ Can provide a service evaluating and caching the target
. . . and uploading the artifacts to the remote-execution service

- So, as a user, can simply checkout the repo I want to work on and dependencies are taken care of—even in a bootstrapped setup!

Serving

- Tree identifier is a function of the typical repository-root descriptions: commit id, the result of unpacking an archive with a given blob id, . . .

⇝ Can provide this mapping as a service
(Of course, that service needs all the sources; but I need to store them anyway)

- From only that can compute the cache key for a target

⇝ Can provide a service evaluating and caching the target
. . . *and uploading the artifacts to the remote-execution service*

- So, as a user, can simply checkout the repo I want to work on and dependencies are taken care of—even in a bootstrapped setup!
- Additonal advantage: different cache rotation frequencies

tl;dr

Summary

- Care about *how* things are defined, not *where*.
- Definitions are interesting objects in their own right.

HUAWEI

tl;dr

Summary

- Care about *how* things are defined, not *where*.
- Definitions are interesting objects in their own right.

Sources

- `https://github.com/just-buildsystem/justbuild`, License: Apache-2.0
- Packaged in AUR, `nixpkgs`, `spack`, `Void`