# gem5 Resources

What you need for running reproducible experiments

# What are Resources? (Disks, kernels, binaries, etc.)

- gem5 resources are prebuilt artifacts that can be used to run gem5 simulations.
- Each gem5 resource falls into one of 13 categories (such as binary or kernel) and supports one of 6 ISAs (including ARM, x86, and RISC-V).
- For more information about categories, visit resources.gem5.org/category
- The gem5 resources website is an easy way to search for the resources you want to use.
  - There are filters based on category, ISA, and gem5 version that help you narrow down the resources based on your requirements.

Let's look at the gem5 resources website.

# Important categories and their description

## Resources

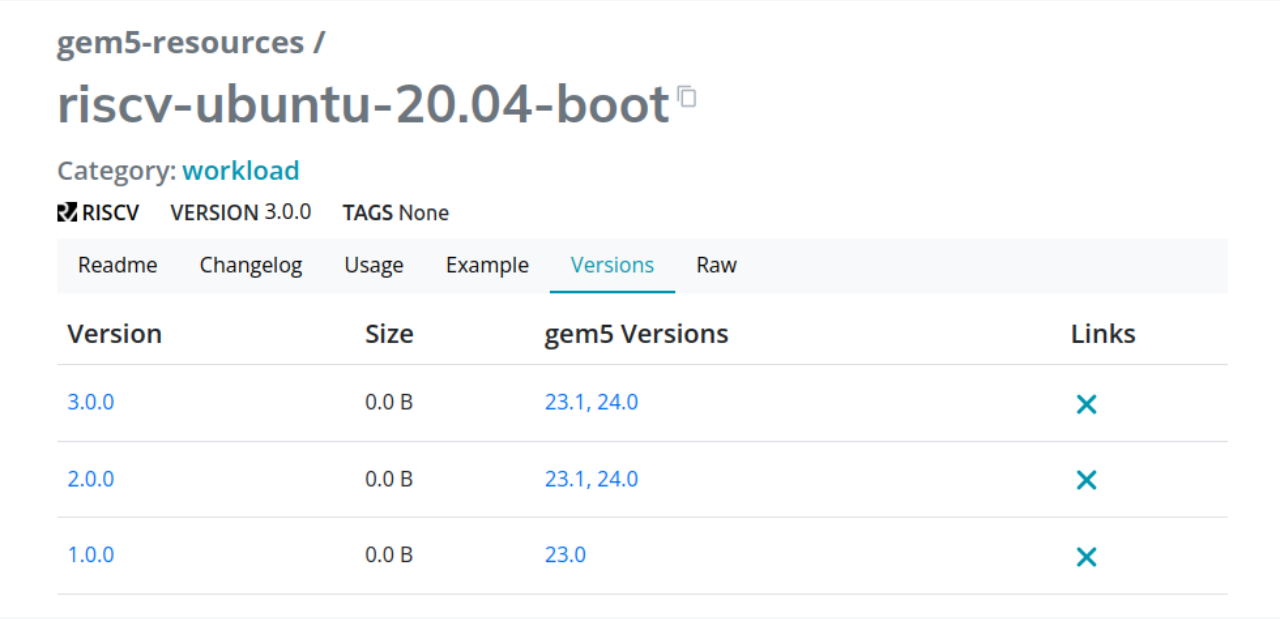
**binary**: A program that is used to test the performance of a computer system.
**kernel**: A computer program that acts as the core of an operating system by managing system resources.
**disk-image**: A file that contains an exact copy of the data stored on a storage device.
**bootloader**: A small program that is responsible for loading the operating system into memory when a computer starts up.
**checkpoint**: A snapshot of a simulation.
**simpoint**: This resource stores all information required to create and restore a Simpoint.
**file**: A resource consisting of a single file.

## Combinations of resources

**workload**: Bundles of resources and any input parameters that can be run directly in gem5.
**suite**: A collection of workloads.

# Resource Versioning

- In gem5, all resources have an `id` and any update to the resource will update the `resource_version`.
- Each unique resource is represented by its `id` and `resource_version`.
- When an existing resource is updated the `id` remains the same but the `resource_version` is updated.
- Each resource also has a field called `gem5_versions` which shows which releases of gem5 the resource is compatible with.



# Using Resources in gem5 Simulations

To use the resources in gem5, we can use the `obtain_resource` function.

This function takes the `resource_id` as an argument and returns the resource object.
It downloads the resource if it is not already present in the gem5 resources cache.

You can optionally specify the `version` of the resource you want to use.

On the website, use the copy button to copy the `id` of the resource you want to use.

# Workloads

A workload is a package of one or more resources that can have pre-defined parameters.

Let's see the `x86-npb-is-size-s-run` workload.

This workload runs the NPB IS benchmark in SE mode.

You can see the JSON of the workloads in the raw tab on the resources website.


```
{
    "category": "workload",
    "id": "x86-npb-is-size-s-run",
    "description": "This workload run the npb-is-size-s binary in SE mode.",
    "function": "set_se_binary_workload",
    "resources": {
        "binary": {
            "id": "x86-npb-is-size-s",
            "resource_version": "1.0.0"
        }
    },
    "architecture": "X86",
    "tags": [
        "npb",
        "x86"
    ],
    "code_examples": [],
    "license": "NASA Open Source Agreement (NOSA)",
    "author": [
        "NASA"
    ],
    "source_url": "",
    "resource_version": "1.0.0",
    "gem5_versions": [
        "23.1",
        "24.0"
    ],
    "example_usage": "obtain_resource(\"x86-npb-is-size-s-run\")",
    "additional_params": {}
}
```

# Workloads (Cont.)

The workload specifies a function that will be run when the workload is set on the board.
For instance, we previously saw how to use the `set_se_binary_workload` function to set the workload on the board.

For instance, in the `x86-npb-is-size-s-run` workload we use `set_se_binary_workload`.

The `resources` field specifies the resources that are required to run the workload. Each item under "resources" will be passed as an argument to the function specified in the `function`.

In this case: `board.set_se_binary_workload(binary=obtain_resource("x86-npb-is-size-s"))`

```
"function": "set_se_binary_workload",
"resources": {
    "binary": {
        "id": "x86-npb-is-size-s",
        "resource_version": "1.0.0"
...
```

# Other types of workloads

When we get to full system simulation, we'll see how there are many other types of resources and other functions that can be used to set the workload on the board.

# Suites

Suites are a collection of workloads. Suites can be downloaded the same way as resources using `obtain_resource`.

Each workload in a suite has `input_groups` that can be used to filter the suite.

You can iterate over the suite to get all the workloads in the suite.

```
getting_started_suite = obtain_resource("x86-getting-started-benchmark-suite")
for workload in getting_started_suite:
    print(f"Workload ID: {workload.get_id()},"
          f"version: {workload.get_resource_version()}")
```

# Suites example

## You can run this in gem5

```
from gem5.resources.resource import obtain_resource

getting_started_suite = \
  obtain_resource("x86-getting-started-benchmark-suite")

for workload in getting_started_suite:
    print(f"Workload ID: {workload.get_id()},"
          f"version: {workload.get_resource_version()}")
```

Each of these workloads can be used in `set_workload` on a board.

Note: These are compatible with the x86 SE-mode boards.

## Output

```
Workload ID: x86-llvm-minisat-run,version: 1.0.0
Workload ID: x86-gapbs-bfs-run,version: 1.0.0
Workload ID: x86-gapbs-tc-run,version: 1.0.0
Workload ID: x86-npb-is-size-s-run,version: 1.0.0
Workload ID: x86-npb-lu-size-s-run,version: 1.0.0
Workload ID: x86-npb-cg-size-s-run,version: 1.0.0
Workload ID: x86-npb-bt-size-s-run,version: 1.0.0
Workload ID: x86-npb-ft-size-s-run,version: 1.0.0
Workload ID: x86-matrix-multiply-run,version: 1.0.0
```

# Other suite functions

You can also filter the suite by `input_groups`.

Input groups are tags for workloads. For example, SPEC CPU has input groups like "test", "train", "ref", etc.

You can also filter based on the input groups.

```
print(f"Input groups: {getting_started_suite.get_input_groups()}")
```

```
print("Workloads from npb:")
for workload in getting_started_suite.with_input_group("npb"):
    print(f"Workload ID: {workload.get_id()},"
          f"version: {workload.get_resource_version()}")
```

# Exercise: Use suites and workloads

We'll come back to this after the multisim section.

# Local resources

You can also use resources that you have created locally in gem5.

You can create a local JSON file to use as a data source, then set the:

- `GEM5_RESOURCE_JSON` environment variable to point to the JSON, if you want to just use the resources in the JSON.
- `GEM5_RESOURCE_JSON_APPEND` environment variable to point to the JSON, if you want to use local resources along with gem5 resources.

For more details on how to use local resources, read the local resources documentation

# Why use local resources

gem5 has two main ways to use local resources.

- Directly create the resource object by passing the local path of the resource.
  - `BinaryResource(local_path=/path/to/binary)`
  - We can use this method when we are making new resources and want to quickly test the resource.
- If we are going to use or share the resource that we created, it is better to create a JSON file and update the data source as mentioned in the above slide.
  - With this method we can use `obtain_resource`.
  - This method makes the simulations more reproducible and consistent.

# Specifying resources in JSON

To use a local resource, you need to create a JSON file that specifies the resource.
The JSON file should have the following fields:

- `category`: The category of the resource.
- `id`: The unique identifier of the resource.
- `resource_version`: The version of the resource.
- `description`: A brief description of the resource.
- `author`: The author of the resource.
- `license`: The license of the resource.
- `source_url`: The source of the resource.
- `tags`: Tags for the resource.
- `gem5_versions`: The gem5 versions the resource is compatible with.
- `example_usage`: An example of how to use the resource.

There are other optional fields that can be added to the JSON file as well. Examples are on the gem5-resources website.

# Exercise: Create a JSON file for your matrix-multiply

Create a JSON file called `my_resources.json` for the matrix-multiply binary that you created in the previous exercise.

Create both a binary resource for the binary and a workload to run the binary.
Call the binary "matrix-multiply" and the workload "matrix-multiply-run".

There is a template JSON file in the `exercises/02-Using-gem5/07-gem5-resources` folder.

Use `GEM5_RESOURCE_JSON_APPEND=my_resources.json gem5 run-mm.py` to run the simulation.

# Answer

```
[
    {
        "category": "binary",
        "id": "matrix-multiply",
        "description": "A simple matrix multiplication
                        binary with ROI annotations",
        "architecture": "X86",
        "tags": [],
        "source": "",
        "url": "file:///workspaces/latin-america-2024/
               exercises/02-Using-gem5/06-custom-benchmarks/
               completed/matrix-multiply",
        "license": "",
        "author": [
            "<Your name>"
        ],
        "resource_version": "1.0.0",
        "gem5_versions": [
            "24.0"
        ]
    },
```

```
    {
        "category": "workload",
        "id": "matrix-multiply-run",
        "function": "set_se_binary_workload",
        "resources": {
            "binary": {
                "id": "matrix-multiply",
                "resource_version": "1.0.0"
            }
        },
        "resource_version": "1.0.0",
        "gem5_versions": [
            "24.0"
        ]
    }
]
```

# Summary

- gem5 resources are prebuilt artifacts that can be used to run gem5 simulations.
- Resources are categorized and versioned.
- You can use resources in gem5 simulations using the `obtain_resource` function.
- Workloads are a collection of resources that can be run on a board.
- Suites are a collection of workloads.
- You can use local resources in gem5 by creating a JSON file and setting the `GEM5_RESOURCE_JSON` or `GEM5_RESOURCE_JSON_APPEND` environment variables.