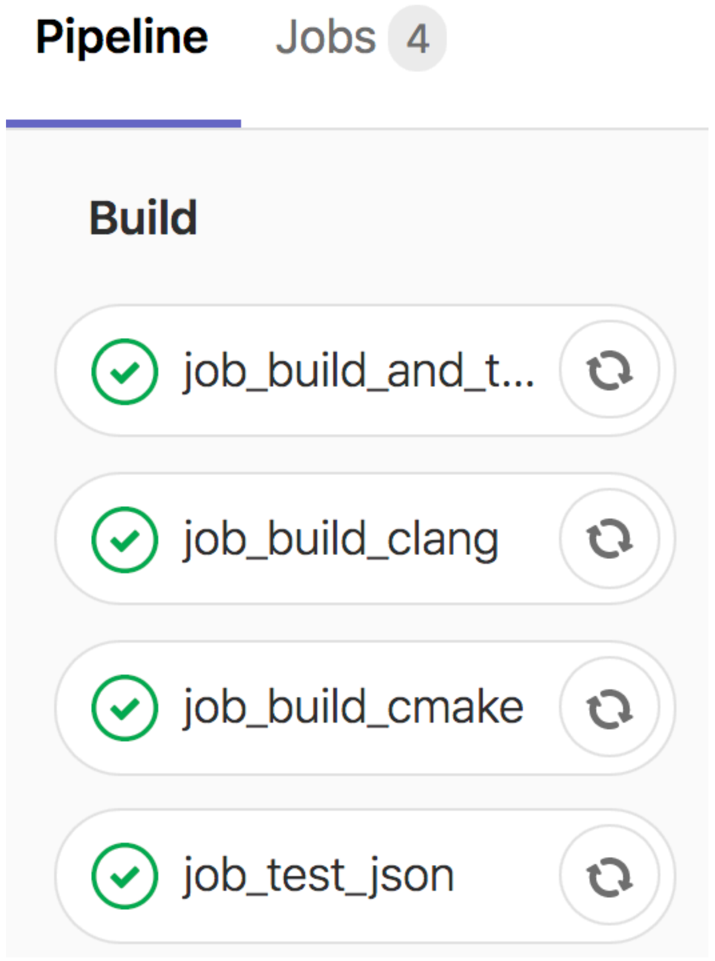
CMake

- New branch to merge after testing
  - https://gitlab.cern.ch/YARR/YARR/tree/devel_rogue_test
- Changes/features (need to test netio):
  - Add support for SLAC rogue
    - FW/SW library hardware abstraction layer
      - PCIe cards, ethernet, RCE
  - Only CMake supported
  - Requires gcc7 for C++17 features
  - Toolchains: arm32, arm64, x86_64/gcc+clang
  - TBB, netio, felixbase, rogue build as external CMake packages from git
  - Keep static YARR executable

CMake

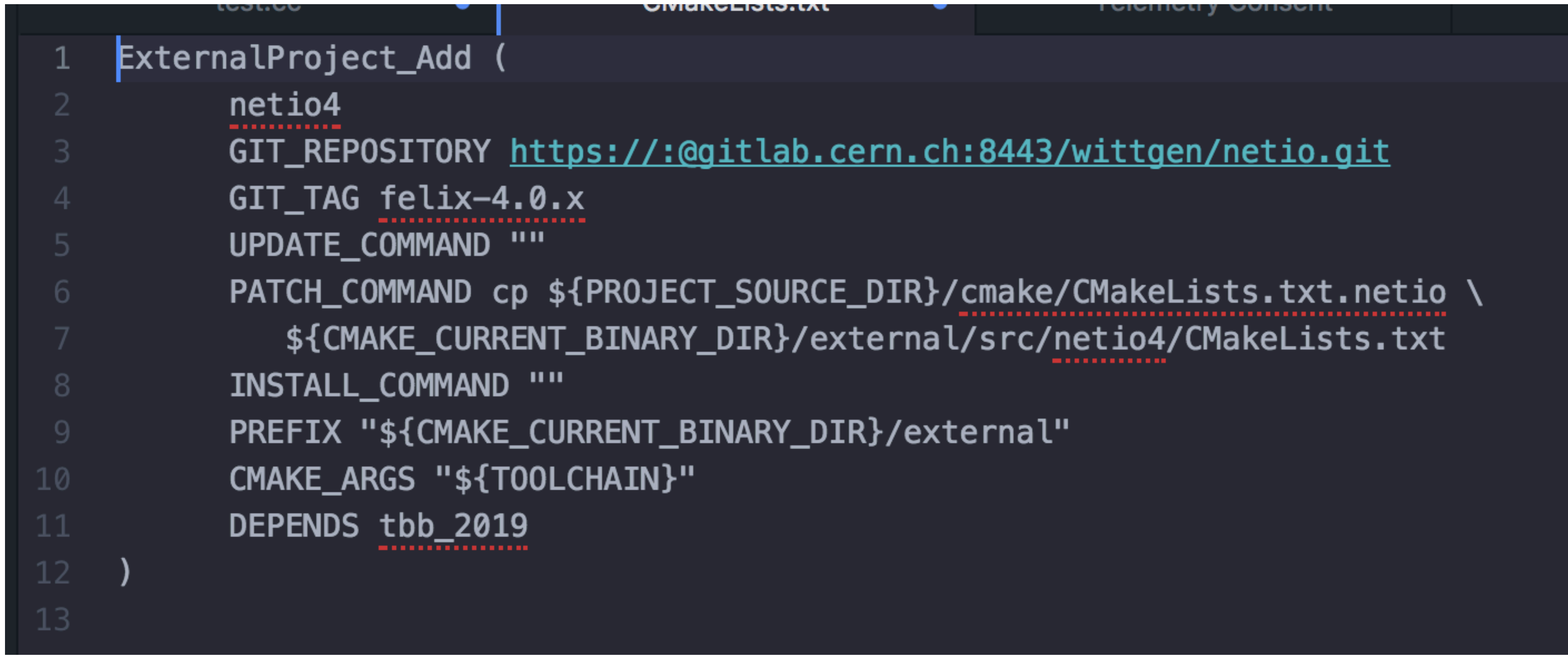
Cleaned up CI targets passing rogue/netio all YARR inclusive


Pipeline
Jobs 4

Build

job_build_and_t...
job_build_clang
job_build_cmake
job_test_json

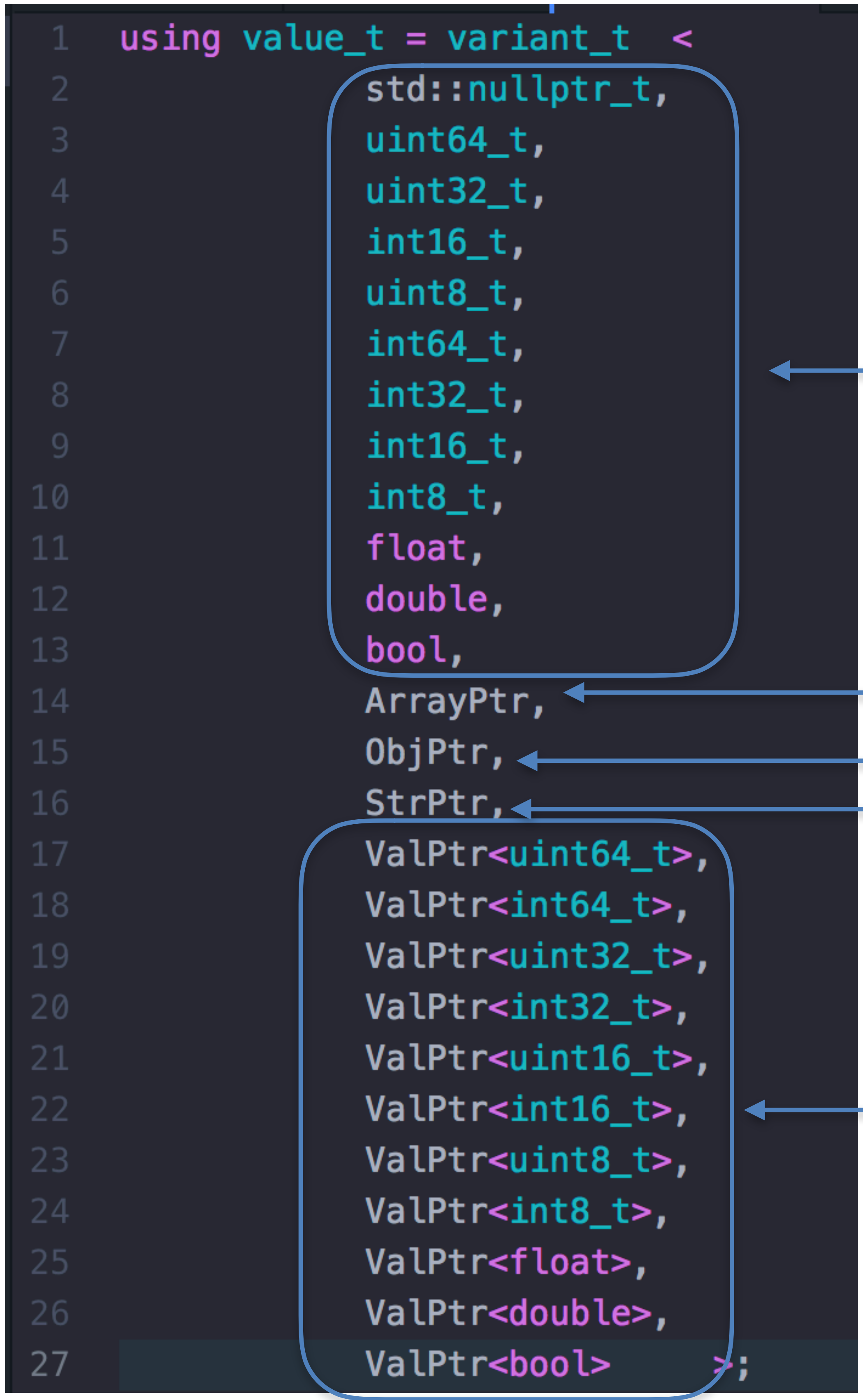
NETIO on-the-fly build with patched CMakeLists.txt


```
ExternalProject_Add (
        netio4
        GIT_REPOSITORY https://:@gitlab.cern.ch:8443/wittgen/netio.git
        GIT_TAG felix-4.0.x
        UPDATE_COMMAND ""
        PATCH_COMMAND cp ${PROJECT_SOURCE_DIR}/cmake/CMakeLists.txt.netio \
            ${CMAKE_CURRENT_BINARY_DIR}/external/src/netio4/CMakeLists.txt
        INSTALL_COMMAND ""
        PREFIX "${CMAKE_CURRENT_BINARY_DIR}/external"
        CMAKE_ARGS "${TOOLCHAIN}"
        DEPENDS tbb_2019
)
```

MessagePack

- Settled on MessagePack as serialization format
- https://msgpack.org/index.html
- “It’s like JSON but fast and small”
  - not quite fast enough
  - implemented a custom zero-copy encoder/decoder
- Another inspiration:
  - JSON for modern C++
  - https://github.com/nlohmann/json
  - aims to make “JSON” a first class C++ data type
  - very inefficient memory use, especially on 64 bit systems
  - slow parsing
  - no direct storage of std::vector<type>, each element is stored as a JSON object using up 16/8 bytes on 64/32 bit systems (pointer or value and type information = 2 words)

New C++ Recursive Variant



```
using value_t = variant_t  <
                std::nullptr_t,
                uint64_t,
                uint32_t,
                int16_t,
                uint8_t,
                int64_t,
                int32_t,
                int16_t,
                int8_t,
                float,
                double,
                bool,
                ArrayPtr,
                ObjPtr,
                StrPtr,
                ValPtr<uint64_t>,
                ValPtr<int64_t>,
                ValPtr<uint32_t>,
                ValPtr<int32_t>,
                ValPtr<uint16_t>,
                ValPtr<int16_t>,
                ValPtr<uint8_t>,
                ValPtr<int8_t>,
                ValPtr<float>,
                ValPtr<double>,
                ValPtr<bool>      >;
```

memory efficient recursive C++ variant class, which translates directly into msgpack
C++17 required

scalar types stored directly in variant

unique pointer to vector<value_t>

unique pointer to map<string,value_t>

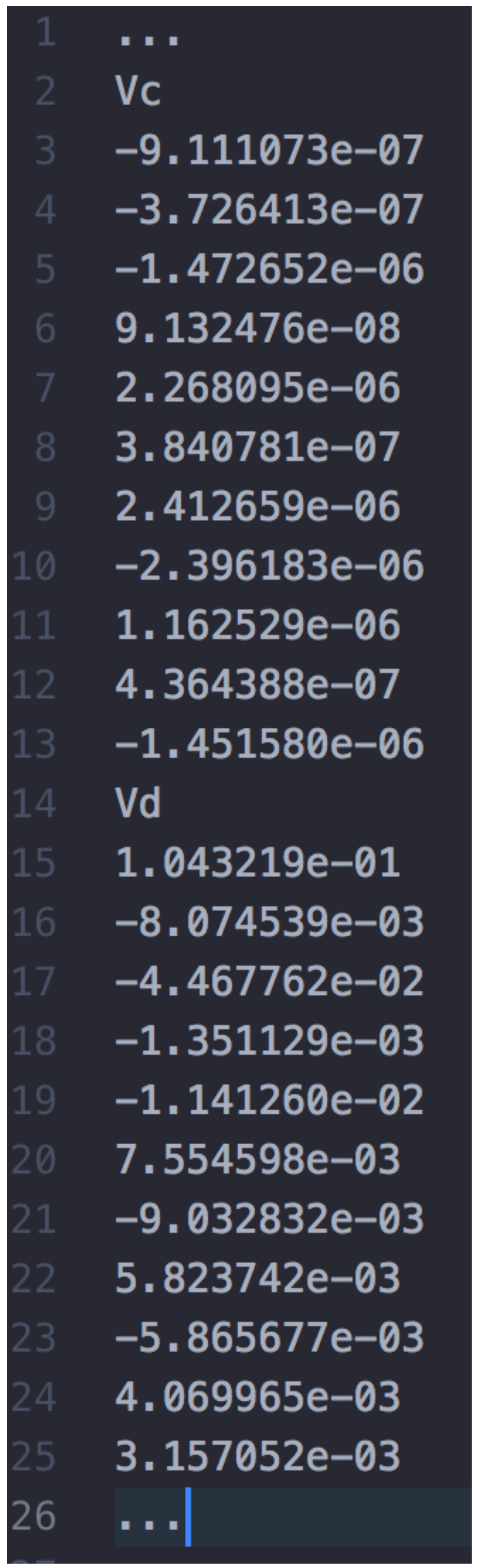
unique pointer to string

unique pointer to vector<scalar_type>
for homogenous arrays

sizeof(value_t)=16 bytes

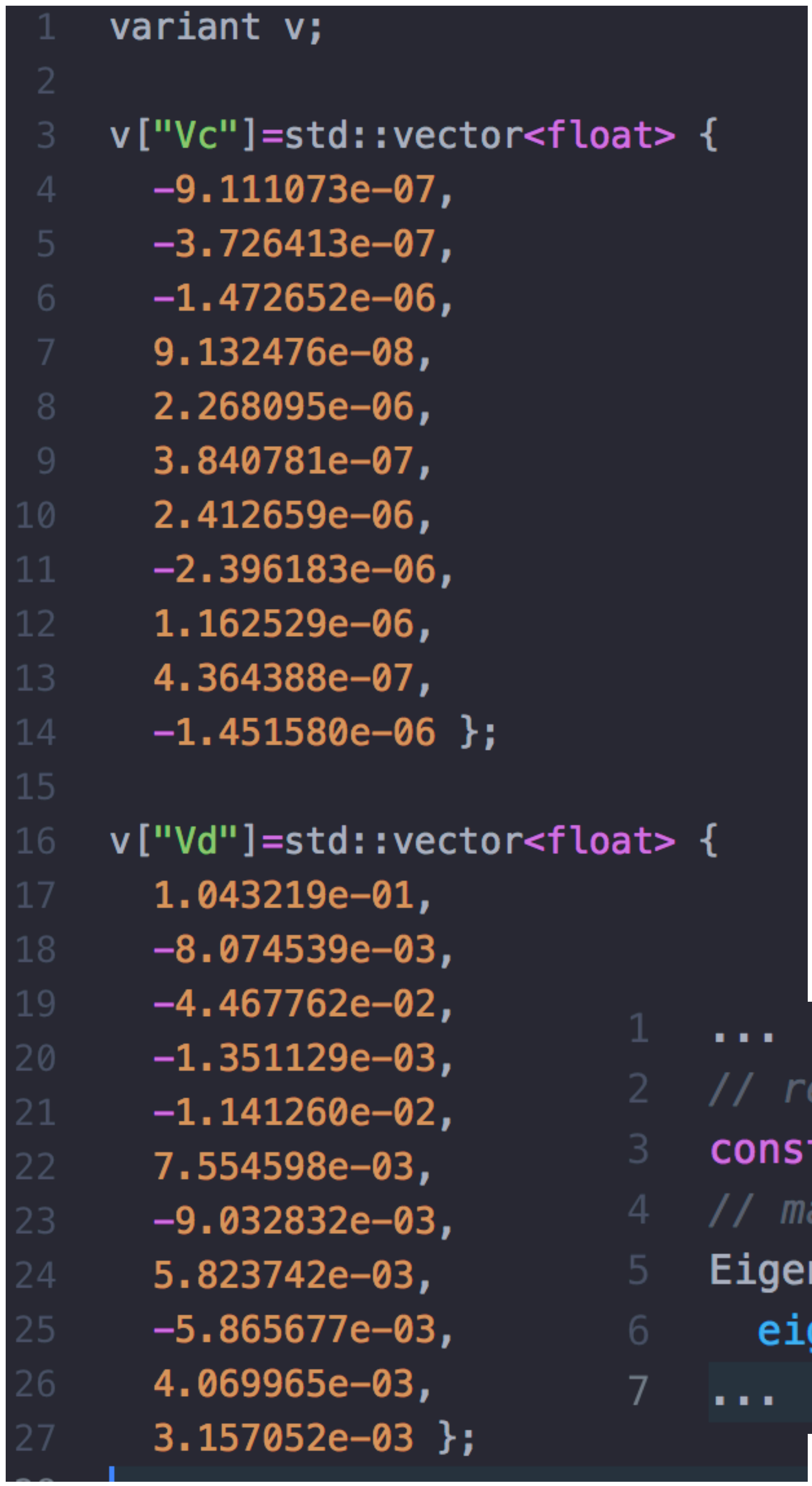
FTK DB

GCON



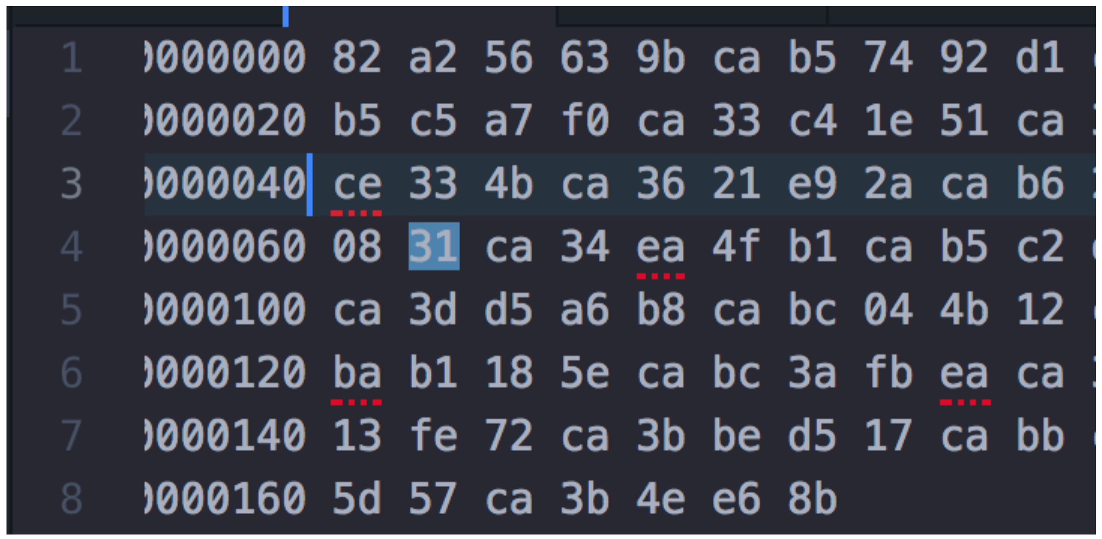
```
...
Vc
-9.111073e-07
-3.726413e-07
-1.472652e-06
9.132476e-08
2.268095e-06
3.840781e-07
2.412659e-06
-2.396183e-06
1.162529e-06
4.364388e-07
-1.451580e-06
Vd
1.043219e-01
-8.074539e-03
-4.467762e-02
-1.351129e-03
-1.141260e-02
7.554598e-03
-9.032832e-03
5.823742e-03
-5.865677e-03
4.069965e-03
3.157052e-03
...
```

C++ variant



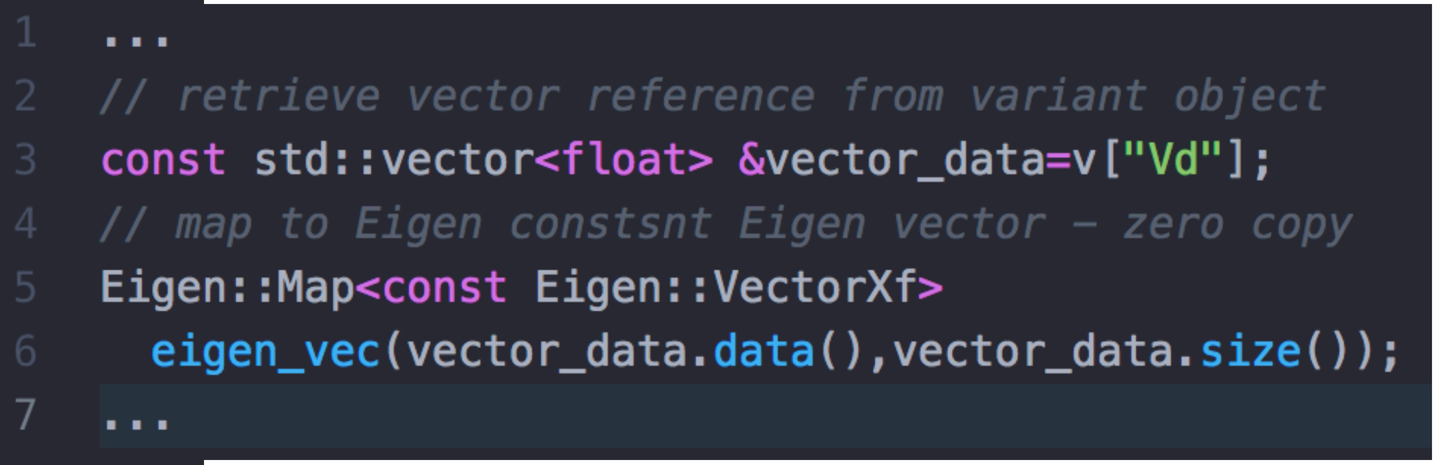
```cpp
variant v;

v["Vc"]=std::vector<float> {
  -9.111073e-07,
  -3.726413e-07,
  -1.472652e-06,
  9.132476e-08,
  2.268095e-06,
  3.840781e-07,
  2.412659e-06,
  -2.396183e-06,
  1.162529e-06,
  4.364388e-07,
  -1.451580e-06 };

v["Vd"]=std::vector<float> {
  1.043219e-01,
  -8.074539e-03,
  -4.467762e-02,
  -1.351129e-03,
  -1.141260e-02,
  7.554598e-03,
  -9.032832e-03,
  5.823742e-03,
  -5.865677e-03,
  4.069965e-03,
  3.157052e-03 };
```

MSGPACK (DB BLOB)



```
000000 82 a2 56 63 9b ca b5 74 92 d1
000020 b5 c5 a7 f0 ca 33 c4 1e 51 ca
000040 ce 33 4b ca 36 21 e9 2a ca b6
000060 08 31 ca 34 ea 4f b1 ca b5 c2
000100 ca 3d d5 a6 b8 ca bc 04 4b 12
000120 ba b1 18 5e ca bc 3a fb ea ca
000140 13 fe 72 ca 3b be d5 17 ca bb
000160 5d 57 ca 3b 4e e6 8b
```

Map to Eigen vector



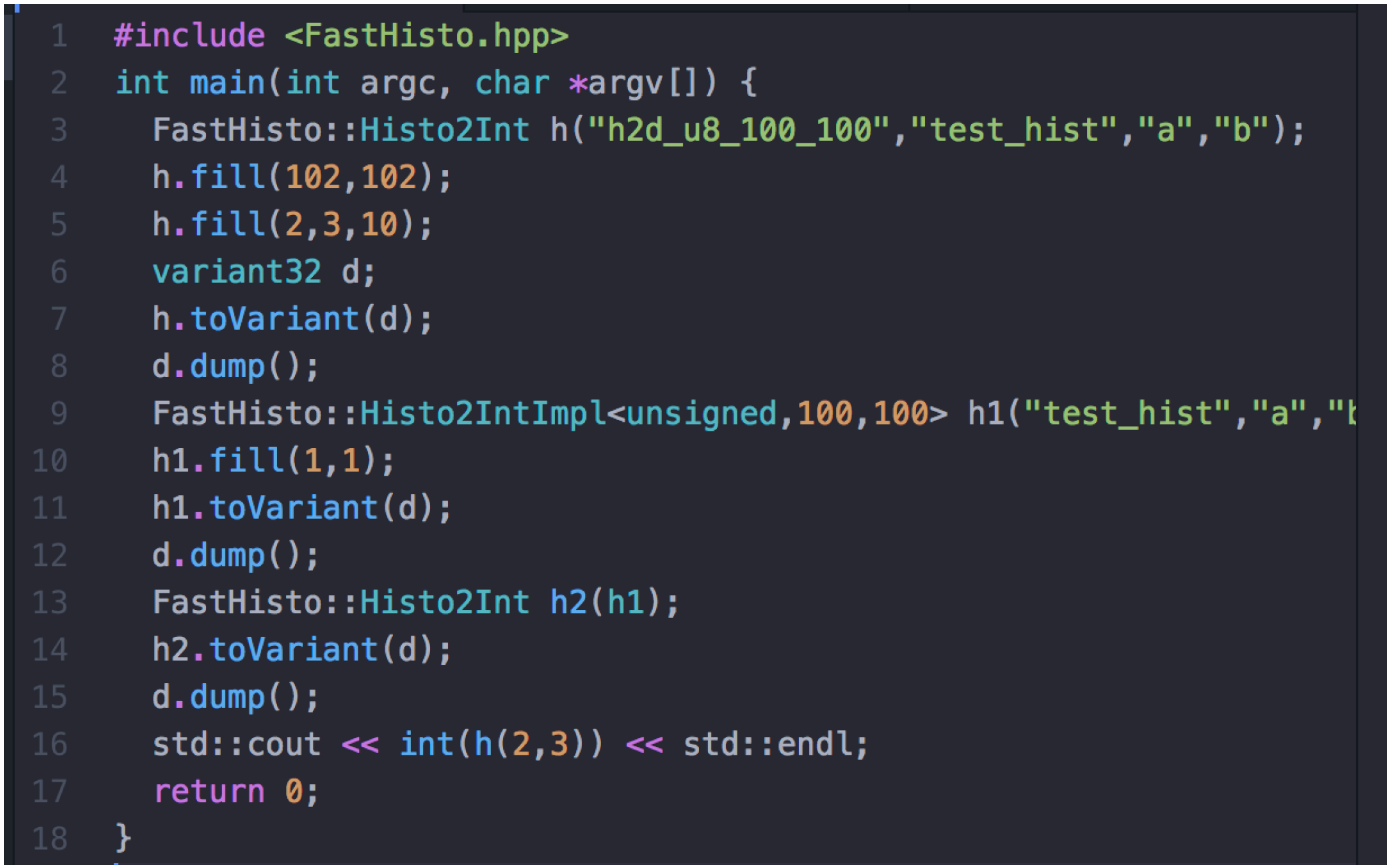
```cpp
...
// retrieve vector reference from variant object
const std::vector<float> &vector_data=v["Vd"];
// map to Eigen constsnt Eigen vector - zero copy
Eigen::Map<const Eigen::VectorXf>
  eigen_vec(vector_data.data(),vector_data.size());
...
```

C++ Recursive Variant

- Uses C++17 std::variant as storage
- https://gitlab.cern.ch/wittgen/variant
- Optimized templates for 32/64bit system
- Custom msgpack
- uses rapidJson (fast) to import/export JSON
    - useful for dumping data object/debugging
- When using msgpack data types are preserved
    - for example: uint8_t, int16_t, array of float, etc
    - JSON standard only supports integers and floats

New Histogram Package

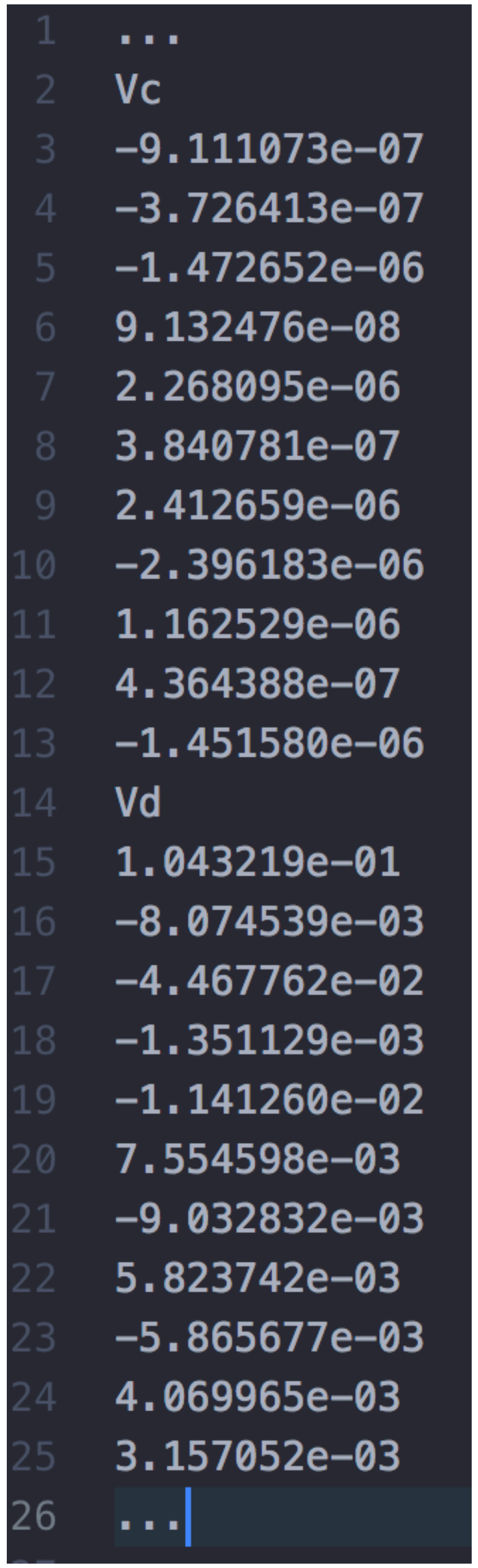
- Based on variant
- https://gitlab.cern.ch/wittgen/fasthisto



```cpp
#include <FastHisto.hpp>
int main(int argc, char *argv[]) {
  FastHisto::Histo2Int h("h2d_u8_100_100","test_hist","a","b");
  h.fill(102,102);
  h.fill(2,3,10);
  variant32 d;
  h.toVariant(d);
  d.dump();
  FastHisto::Histo2IntImpl<unsigned,100,100> h1("test_hist","a","
  h1.fill(1,1);
  h1.toVariant(d);
  d.dump();
  FastHisto::Histo2Int h2(h1);
  h2.toVariant(d);
  d.dump();
  std::cout << int(h(2,3)) << std::endl;
  return 0;
}
```

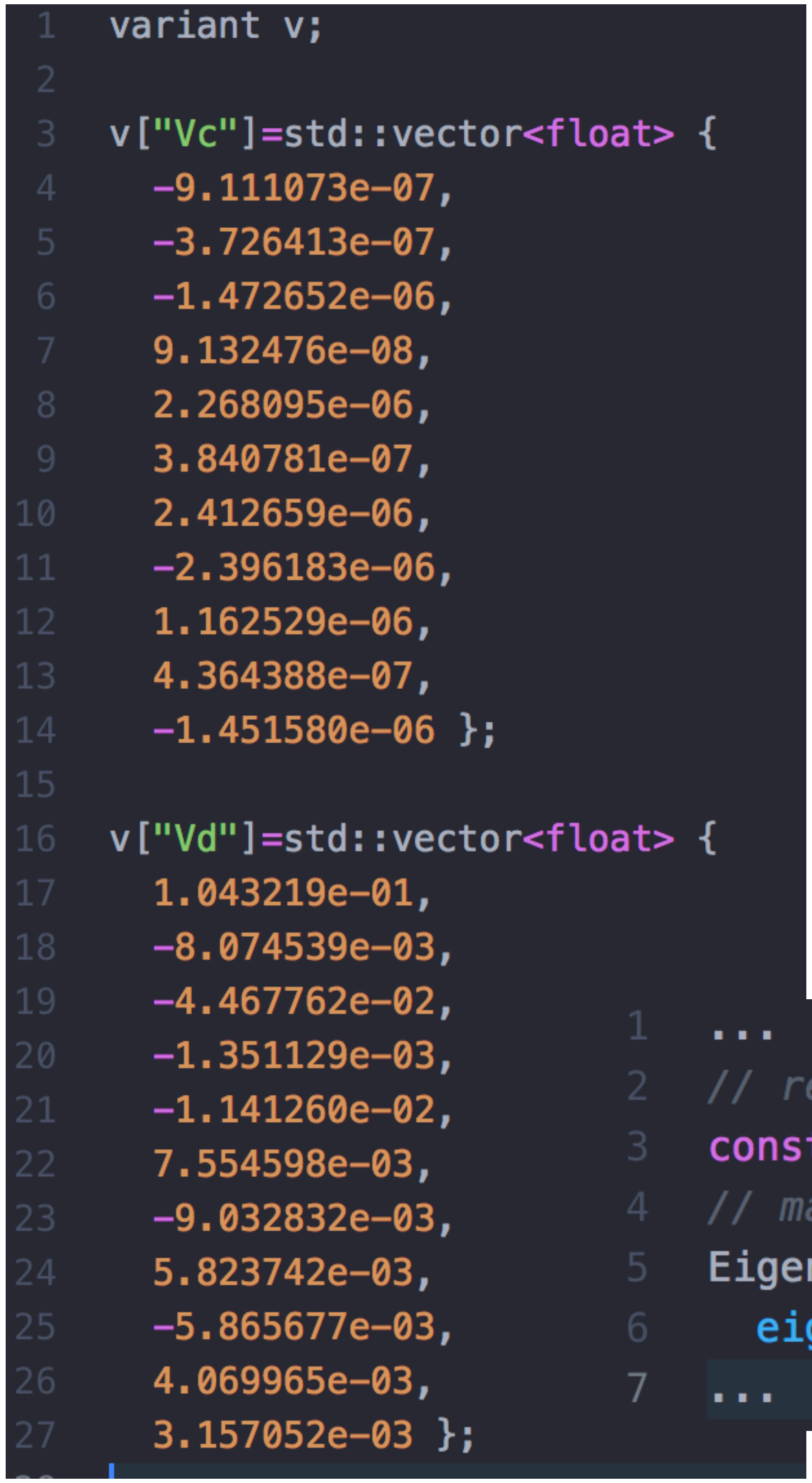
FTK Variant Use Case

GCON



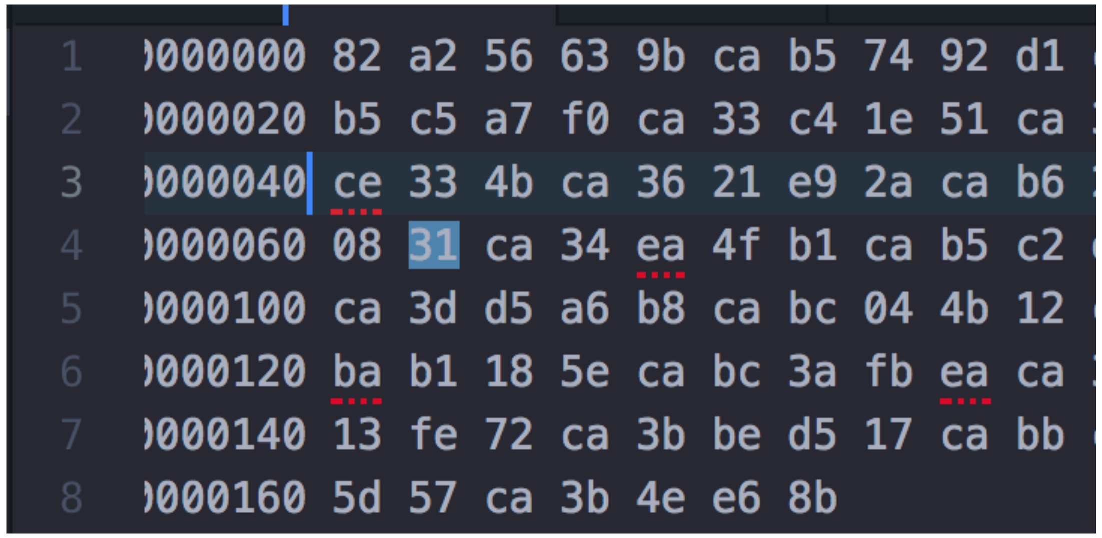
```
...
Vc
-9.111073e-07
-3.726413e-07
-1.472652e-06
9.132476e-08
2.268095e-06
3.840781e-07
2.412659e-06
-2.396183e-06
1.162529e-06
4.364388e-07
-1.451580e-06
Vd
1.043219e-01
-8.074539e-03
-4.467762e-02
-1.351129e-03
-1.141260e-02
7.554598e-03
-9.032832e-03
5.823742e-03
-5.865677e-03
4.069965e-03
3.157052e-03
...
```

C++ variant



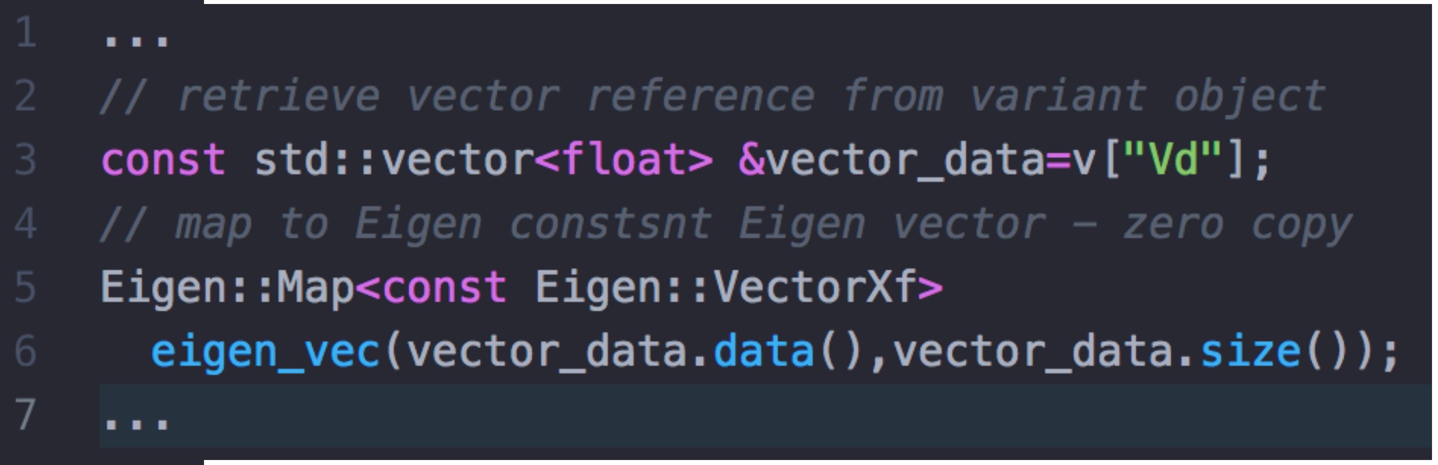
```
variant v;

v["Vc"]=std::vector<float> {
  -9.111073e-07,
  -3.726413e-07,
  -1.472652e-06,
  9.132476e-08,
  2.268095e-06,
  3.840781e-07,
  2.412659e-06,
  -2.396183e-06,
  1.162529e-06,
  4.364388e-07,
  -1.451580e-06 };

v["Vd"]=std::vector<float> {
  1.043219e-01,
  -8.074539e-03,
  -4.467762e-02,
  -1.351129e-03,
  -1.141260e-02,
  7.554598e-03,
  -9.032832e-03,
  5.823742e-03,
  -5.865677e-03,
  4.069965e-03,
  3.157052e-03 };
```

MSGPACK (DB BLOB)



```
000000 82 a2 56 63 9b ca b5 74 92 d1
000020 b5 c5 a7 f0 ca 33 c4 1e 51 ca
000040 ce 33 4b ca 36 21 e9 2a ca b6
000060 08 31 ca 34 ea 4f b1 ca b5 c2
000100 ca 3d d5 a6 b8 ca bc 04 4b 12
000120 ba b1 18 5e ca bc 3a fb ea ca
000140 13 fe 72 ca 3b be d5 17 ca bb
000160 5d 57 ca 3b 4e e6 8b
```

Map to Eigen vector



```
...
// retrieve vector reference from variant object
const std::vector<float> &vector_data=v["Vd"];
// map to Eigen constsnt Eigen vector - zero copy
Eigen::Map<const Eigen::VectorXf>
  eigen_vec(vector_data.data(),vector_data.size());
...
```