CARIAD

Master Thesis: AI Usage in CI/CD/CT Pipelines for Compute Platforms in Automotives – Status 3

We transform automotive mobility

CARIAD
A VOLKSWAGEN GROUP COMPANY

Agenda

// Previous Developments

// Status

// Method

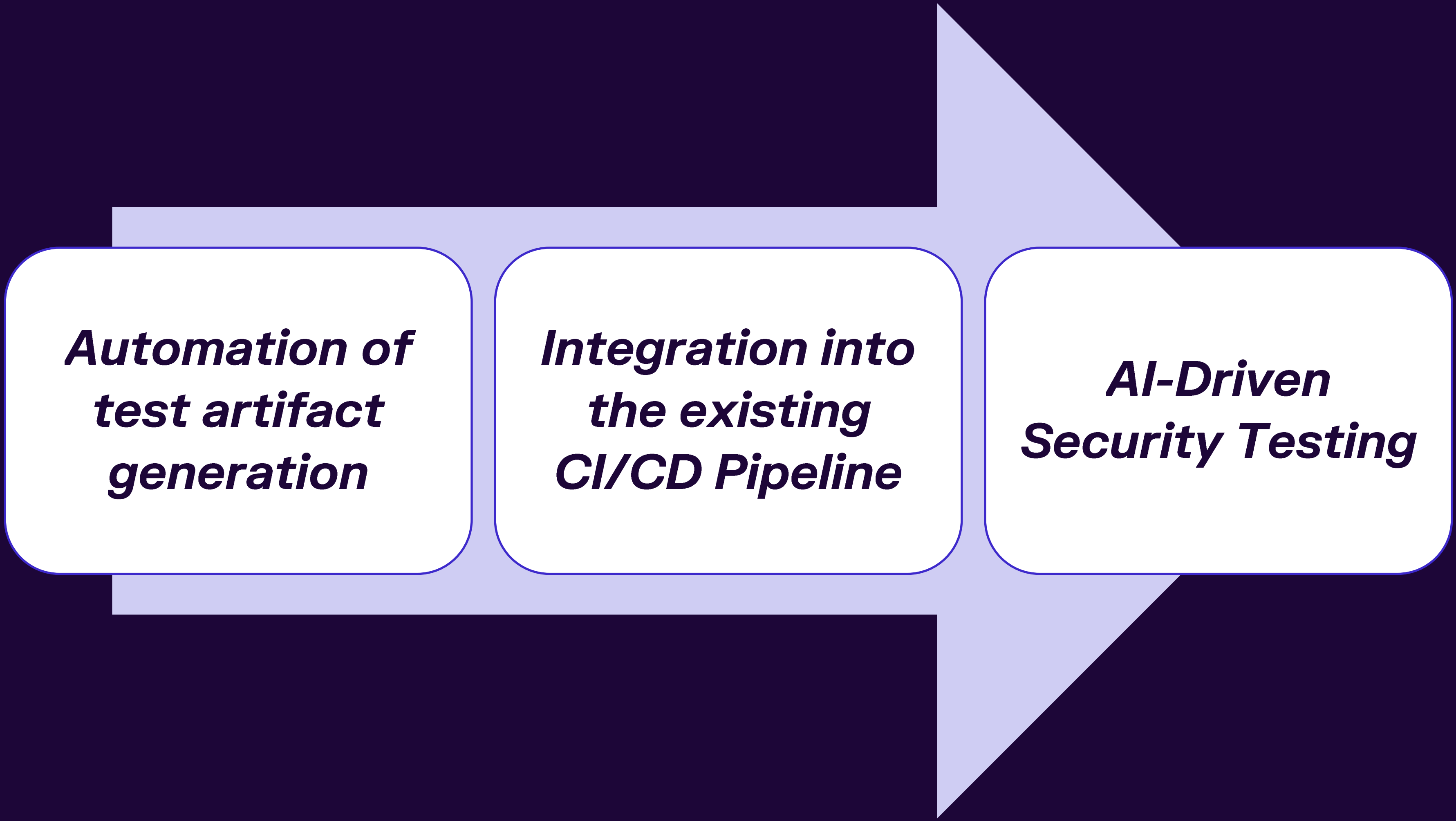
// Results

// Literature Review

// Next Steps

Previous Developments

Expected Outcomes



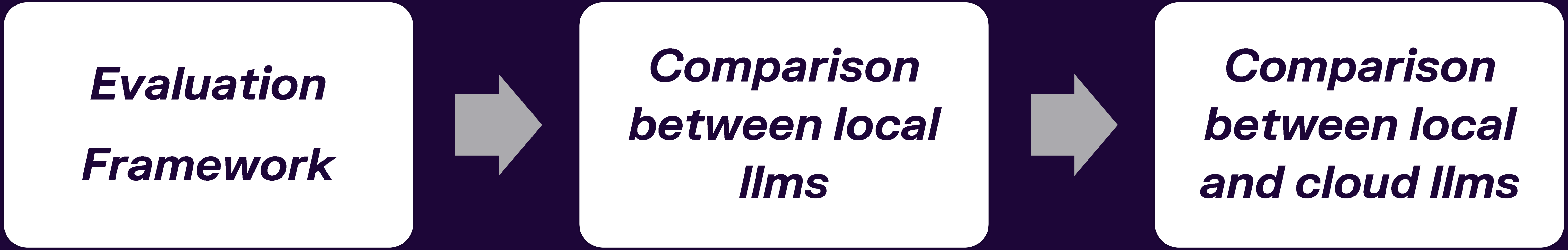
Automation of test artifact generation

Integration into the existing CI/CD Pipeline

AI-Driven Security Testing

May

Previously



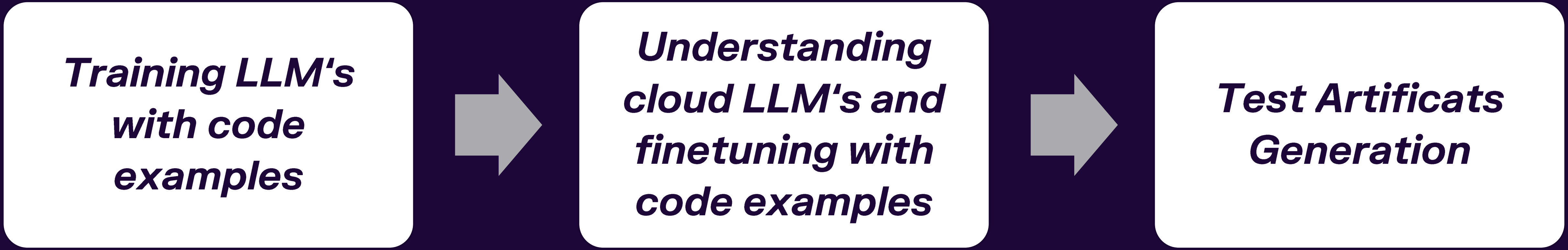
Evaluation Framework

Comparison between local llms

Comparison between local and cloud llms

June

Previously



Training LLM‘s with code examples

Understanding cloud LLM‘s and finetuning with code examples

Test Artificats Generation

July

Finetuning local llms

Qwen 2.5 coder 32b instruct full model 60gb

LoRA (Low-Rank Adaptation)

- LoRA rank (r): 16 – controls adaptation size
- LoRA alpha: 32 – scaling for adaptation
- Dropout: 0.1 – prevents overfitting
- Target modules: q_proj, v_proj, etc. – efficient fine-tuning
- Device: auto – runs on best available hardware
- Dtype: float16 – faster, less memory
- Efficient model loading and saving (safetensors)

Status

Status

- *Finetuning local llms*
- *Comparing various results*
- *Automated workflow – Success*
- *CI/CD Pipeline – Started*

Finetuning

Finetuning

- 32b – needs more than 60gb ram
- 14b – (32 – 37gb) ram
- 7b – 24gb ram without any background process
- 1.5b – 14gb ram

Finetuning

NAME	SIZE
qwen2.5-coder:1.5b	986 MB
qwen-fuzzer-1.5-709-examples:latest	3.1 GB
qwen-fuzzer-1.5-172-examples:latest	3.1 GB

Finetuning - Successful llms

Yaml cpp (35 files, 1061 candidates)

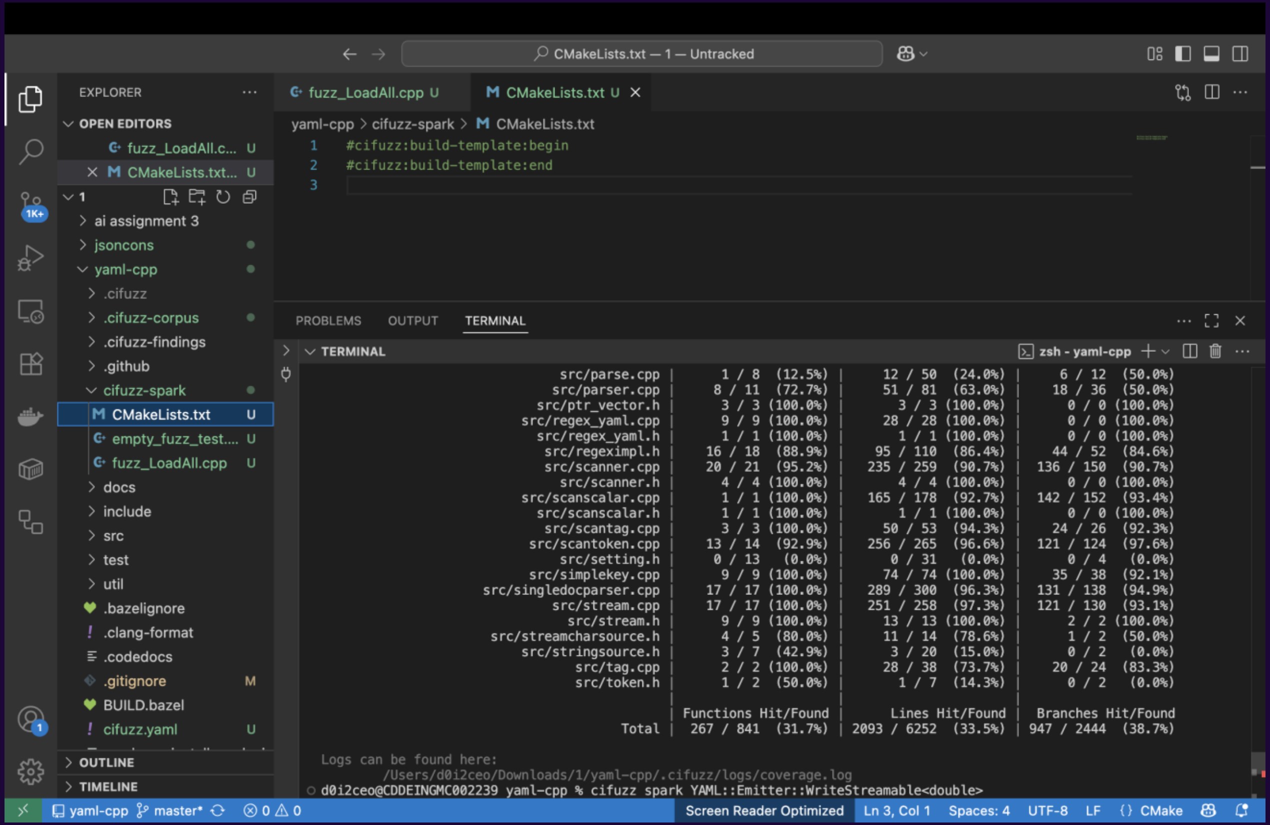
Models	Code Coverage	Time Taken	No of tokens used	Unique test cases	Successfull fuzz tests
Qwen 2.5 coder 1.5b	same	15 m	112k		
172 examples	same	12 m	65k		
709 examples	same	10 m	50k		

Comparison

Comparison

- Cifuzz run without spark - varies
- Cifuzz run with spark - same
- Cifuzz spark generated fuzz test case with cifuzz run - same

Comparison

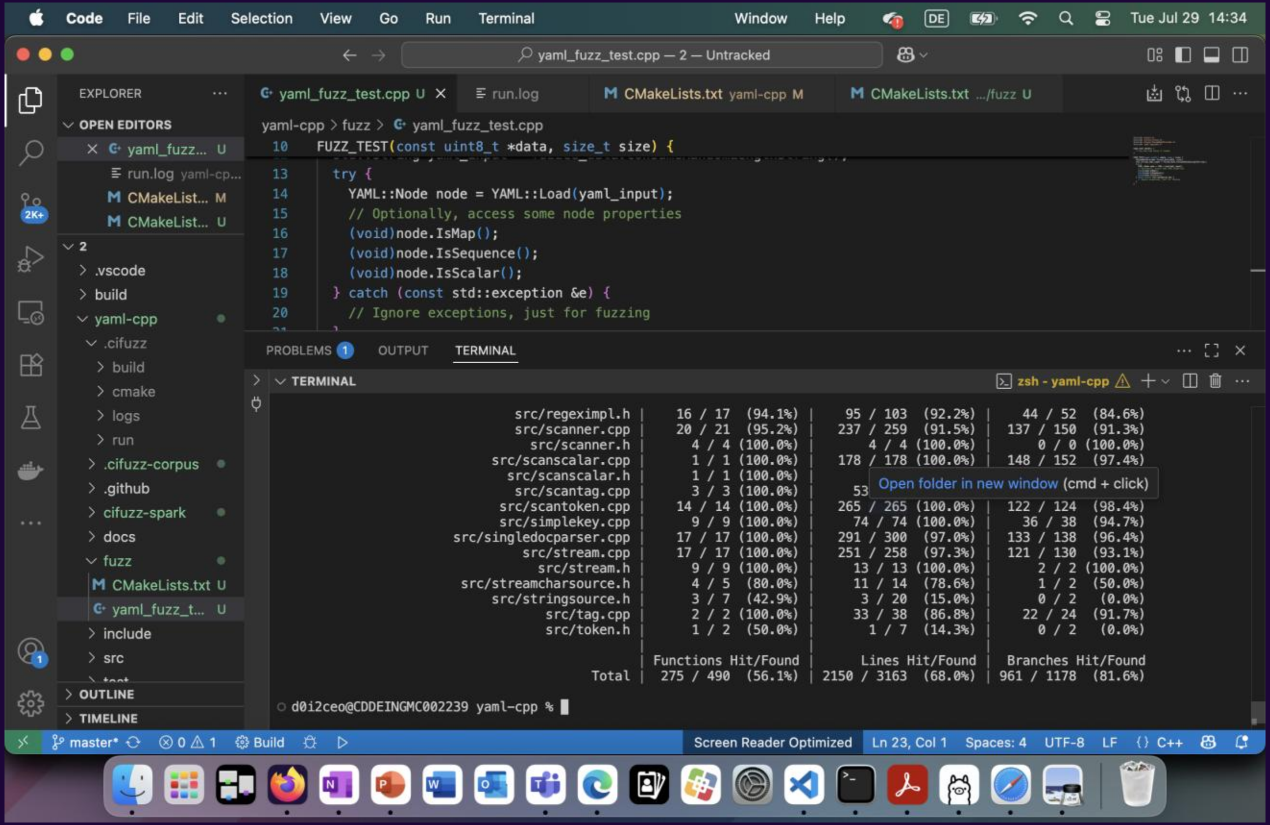


```
#cifuzz:build-template:begin
#cifuzz:build-template:end
```

```
           src/parse.cpp |   1 / 8  (12.5%) |   12 / 50  (24.0%) |    6 / 12  (50.0%)
          src/parser.cpp |   8 / 11 (72.7%) |   51 / 81  (63.0%) |   18 / 36  (50.0%)
         src/ptr_vector.h |   3 / 3 (100.0%) |    3 / 3 (100.0%) |    0 / 0 (100.0%)
       src/regex_yaml.cpp |   9 / 9 (100.0%) |   28 / 28 (100.0%) |    0 / 0 (100.0%)
         src/regex_yaml.h |   1 / 1 (100.0%) |    1 / 1 (100.0%) |    0 / 0 (100.0%)
          src/regeximpl.h |  16 / 18  (88.9%) |  95 / 110  (86.4%) |   44 / 52  (84.6%)
          src/scanner.cpp |  20 / 21  (95.2%) | 235 / 259  (90.7%) |  136 / 150  (90.7%)
            src/scanner.h |   4 / 4 (100.0%) |    4 / 4 (100.0%) |    0 / 0 (100.0%)
       src/scanscalar.cpp |   1 / 1 (100.0%) | 165 / 178  (92.7%) |  142 / 152  (93.4%)
         src/scanscalar.h |   1 / 1 (100.0%) |    1 / 1 (100.0%) |    0 / 0 (100.0%)
          src/scantag.cpp |   3 / 3 (100.0%) |   50 / 53  (94.3%) |   24 / 26  (92.3%)
        src/scantoken.cpp |  13 / 14  (92.9%) | 256 / 265  (96.6%) |  121 / 124  (97.6%)
            src/setting.h |   0 / 13   (0.0%) |    0 / 31   (0.0%) |    0 / 4   (0.0%)
        src/simplekey.cpp |   9 / 9 (100.0%) |   74 / 74 (100.0%) |   35 / 38  (92.1%)
   src/singledocparser.cpp |  17 / 17 (100.0%) | 289 / 300  (96.3%) |  131 / 138  (94.9%)
           src/stream.cpp |  17 / 17 (100.0%) | 251 / 258  (97.3%) |  121 / 130  (93.1%)
             src/stream.h |   9 / 9 (100.0%) |   13 / 13 (100.0%) |    2 / 2 (100.0%)
   src/streamcharsource.h |   4 / 5  (80.0%) |   11 / 14  (78.6%) |    1 / 2  (50.0%)
       src/stringsource.h |   3 / 7  (42.9%) |    3 / 20  (15.0%) |    0 / 2   (0.0%)
              src/tag.cpp |   2 / 2 (100.0%) |   28 / 38  (73.7%) |   20 / 24  (83.3%)
              src/token.h |   1 / 2  (50.0%) |    1 / 7  (14.3%) |    0 / 2   (0.0%)
                          |                   |                    |
                          | Functions Hit/Found |   Lines Hit/Found |  Branches Hit/Found
                    Total |  267 / 841 (31.7%) | 2093 / 6252 (33.5%) | 947 / 2444 (38.7%)

  Logs can be found here:
        /Users/d0i2ceo/Downloads/1/yaml-cpp/.cifuzz/logs/coverage.log
○ d0i2ceo@CDDEINGMC002239 yaml-cpp % cifuzz spark YAML::Emitter::WriteStreamable<double>
```

Comparison



yaml-cpp > fuzz > yaml_fuzz_test.cpp

```cpp
FUZZ_TEST(const uint8_t *data, size_t size) {
  try {
    YAML::Node node = YAML::Load(yaml_input);
    // Optionally, access some node properties
    (void)node.IsMap();
    (void)node.IsSequence();
    (void)node.IsScalar();
  } catch (const std::exception &e) {
    // Ignore exceptions, just for fuzzing
```

```
          src/regeximpl.h |   16 / 17  (94.1%) |    95 / 103  (92.2%) |   44 / 52  (84.6%)
          src/scanner.cpp |   20 / 21  (95.2%) |   237 / 259  (91.5%) |  137 / 150  (91.3%)
            src/scanner.h |    4 / 4 (100.0%) |      4 / 4 (100.0%) |     0 / 0 (100.0%)
       src/scanscalar.cpp |    1 / 1 (100.0%) |   178 / 178 (100.0%) |  148 / 152  (97.4%)
         src/scanscalar.h |    1 / 1 (100.0%) |
          src/scantag.cpp |    3 / 3 (100.0%) |    53
        src/scantoken.cpp |   14 / 14 (100.0%) |   265 / 265 (100.0%) |  122 / 124  (98.4%)
        src/simplekey.cpp |    9 / 9 (100.0%) |    74 / 74 (100.0%) |   36 / 38  (94.7%)
   src/singledocparser.cpp |   17 / 17 (100.0%) |   291 / 300  (97.0%) |  133 / 138  (96.4%)
           src/stream.cpp |   17 / 17 (100.0%) |   251 / 258  (97.3%) |  121 / 130  (93.1%)
             src/stream.h |    9 / 9 (100.0%) |    13 / 13 (100.0%) |     2 / 2 (100.0%)
  src/streamcharsource.h |    4 / 5  (80.0%) |    11 / 14  (78.6%) |     1 / 2  (50.0%)
      src/stringsource.h |    3 / 7  (42.9%) |     3 / 20  (15.0%) |     0 / 2   (0.0%)
              src/tag.cpp |    2 / 2 (100.0%) |    33 / 38  (86.8%) |   22 / 24  (91.7%)
              src/token.h |    1 / 2  (50.0%) |     1 / 7  (14.3%) |     0 / 2   (0.0%)
                          |                    |                      |
                          | Functions Hit/Found |    Lines Hit/Found  | Branches Hit/Found
                    Total |  275 / 490  (56.1%) | 2150 / 3163  (68.0%) | 961 / 1178  (81.6%)

d0i2ceo@CDDEINGMC002239 yaml-cpp %
```

Costs

Costs

Daily Token Usage	Input Tokens	Output Tokens	Daily Cost	Monthly Cost (22 working days)	Annual Cost
Light Usage	5,000	7,500	€0.28	€6.16	€73.92
Moderate Usage	15,000	22,500	€0.83	€18.26	€219.12
Heavy Usage	50,000	75,000	€2.75	€60.50	€726.00
Enterprise Usage	100,000	150,000	€5.50	€121.00	€1,452.00

Costs

Test Scenario	Tokens Consumed	Cost per Run	Runs per Day	Daily Total
Single code file fuzzing	2,000 in + 3,000 out	€0.11	10	€1.10
Module testing	8,000 in + 12,000 out	€0.44	5	€2.20
Full application scan	25,000 in + 35,000 out	€1.30	2	€2.60
CI/CD pipeline integration	15,000 in + 20,000 out	€0.75	8	€6.00

Literature Review

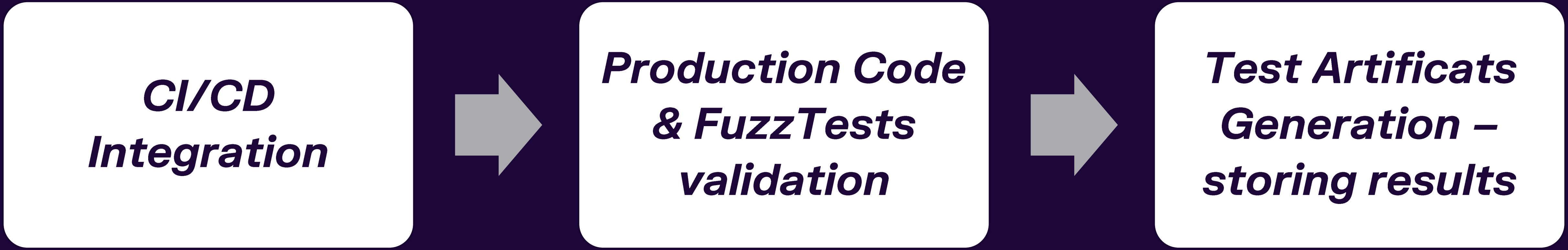
Literature Review

1. Fuzz4All- Universal Fuzzing with Large Language Models
2. Large Language Models Are Edge-Case Fuzzers- Testing Deep Learning Libraries via FuzzGPT
3. Large Language Models are Zero-Shot Fuzzers- Fuzzing Deep-Learning Libraries via Large Language Models
4. Large Language Models Based Fuzzing Techniques- A Survey

Literature Review

Next Steps

Next Steps



CI/CD Integration

Production Code & FuzzTests validation

Test Artificats Generation – storing results

Thank you!

Any Questions