KubeCon CloudNativeCon North America 2026

#KubeCon #CloudNativeCon

Breaking Things Safely, Finding Bugs Before They Find You: Fuzzing Metal³ Through Controlled Chaos

Moshiur Rahman
Ericsson Software Technology



The reliability challenge



Bare-metal infrastructure must behave across real-world conditions.

Bare-metal + Kubernetes: Reliability surface ??

Why this is challenging

- Infrastructure inputs vary across hardware, networks, and installations
- Controllers repeatedly reconcile changing desired and observed state
- Rare edge cases can be hard to reproduce with hand-written tests

Metal³ components in scope

- IPAM
- Cluster API Provider Metal³ (CAPM3)
- Bare Metal Operator (BMO)
- Ironic Standalone Operator (IrSO)

Unit testing and fuzzing complement each other

Unit testing

Goal	Verify expected behavior
Inputs	Developer-selected cases
Oracle	Expected results known
Run pattern	Defined set of tests
Best at	Requirements and business logic

Expected cases
Developer-selected

→

Known result
Assert correctness

Fuzzing

Goal	Explore unexpected behavior
Inputs	Engine generates or mutates
Oracle	Check failures and invariants
Run pattern	Repeated; coverage-guided
Best at	Edge cases, crashes, hangs

Many inputs
Generated / mutated

→

Observe
Crash / hang / invariant

Fuzzing finds the case you did not write

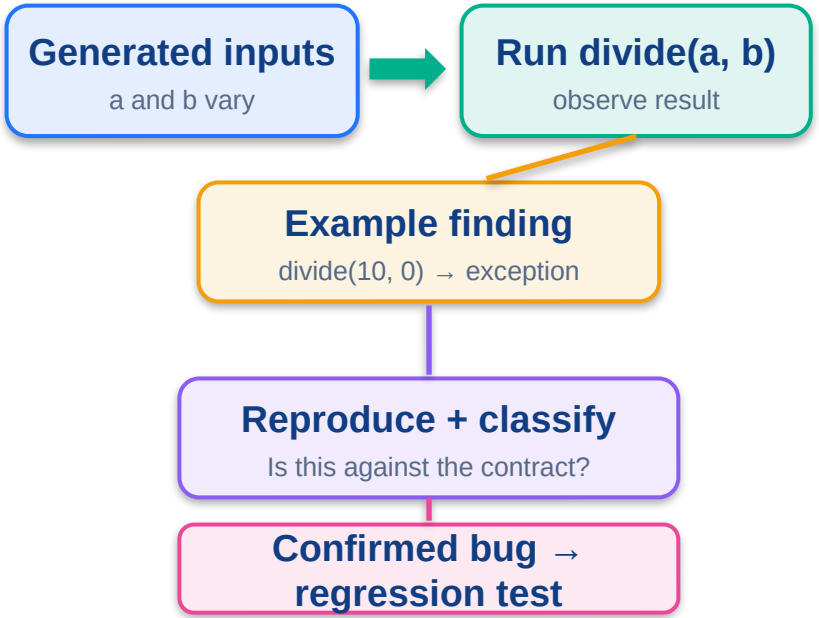
Function under test

```
def divide(a, b):
    return a / b
```

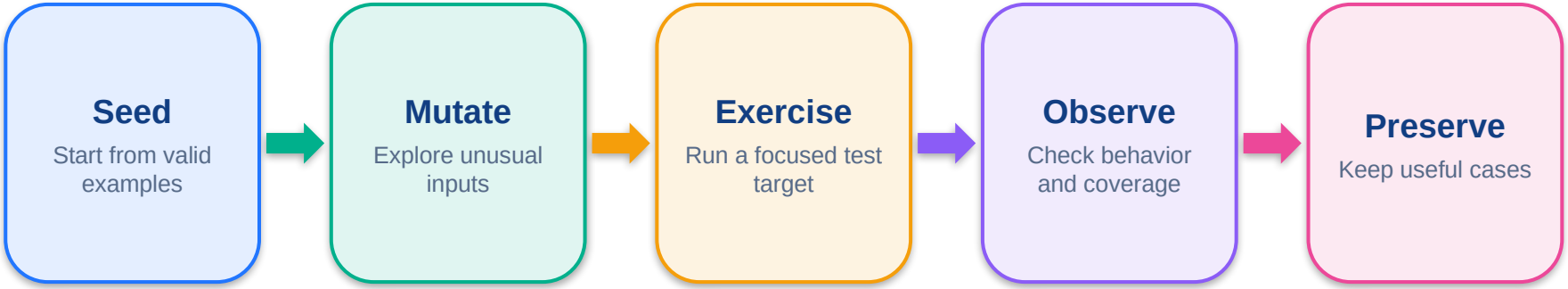
Unit tests: chosen cases

divide(10, 2) → 5
divide(9, 3) → 3
divide(-10, 2) → -5

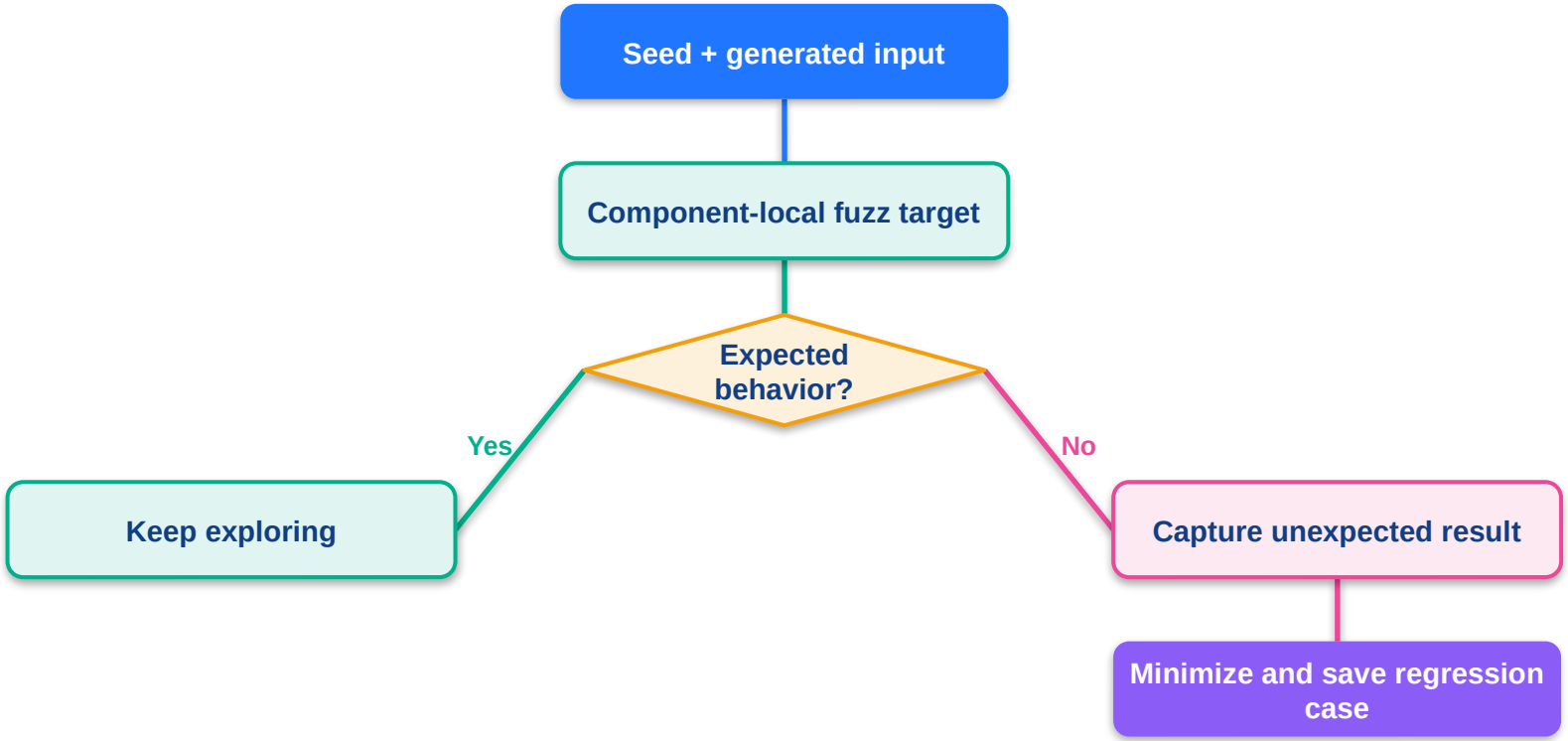
Fuzzer explores variations



Fuzzing: Input variation → repeatable tests



A Focused Fuzz Target



Component boundaries

Good candidates

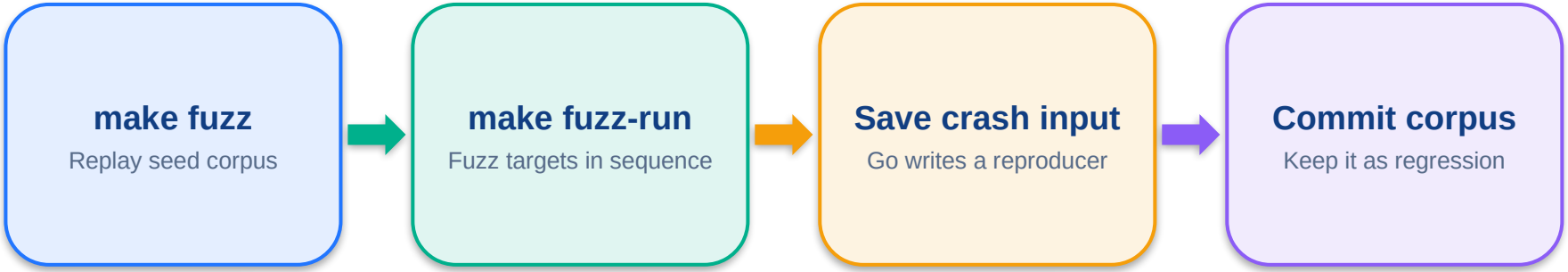
- Parsers, decoders, and validation paths
- Input normalization and boundary-value handling
- Small, deterministic functions with clear expected behavior

Keep the harness focused

- Exercise IPAM, CAPM3, BMO, and IrSO at their own testable seams
- Avoid requiring a live cluster or external infrastructure per input
- Assert invariants; do not treat every rejected input as a failure

BMO: Upstream starting point

Upstream reference: github.com/metal3-io/baremetal-operator/tree/main/test/fuzz

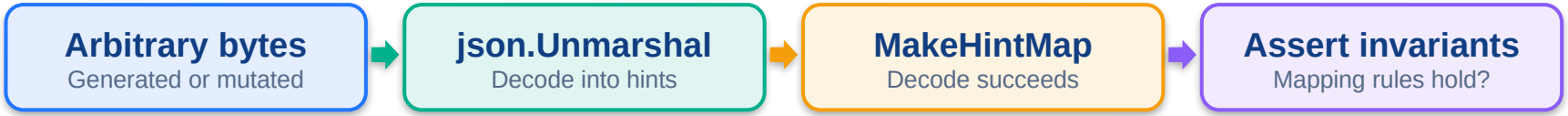


Reading fuzz output: skipped, interesting, failed

FuzzMakeHintMap | illustrative Go output and supplied run counters

```
fuzz: ... gathering baseline coverage: 170/170 completed
fuzz: ... now fuzzing with 8 workers
fuzz: ... execs: 738333, new interesting: 203 (total: 221)
```

738,333 executed | 203 new coverage cases | 221 corpus entries



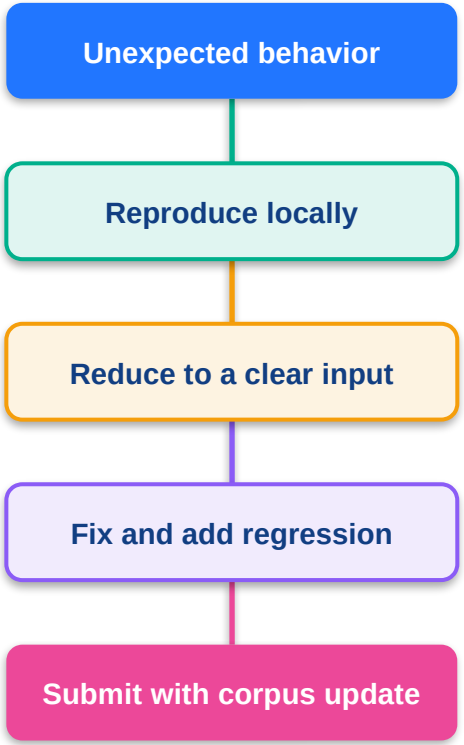
Decode error → t.Skip(): this iteration ends; no MakeHintMap call

Interesting ≠ failure
New coverage → keep in cache; continue exploring

Failure / crasher
Panic or failed assertion → stop; save reproducer

Skipped ≠ failed. Decodable ≠ semantically valid. Invariants still matter.

Anomaly → Upstream regression



Adopt with the Development Rhythm

Fast feedback

When	Routine test runs
Focus	Seed-corpus regressions
Design	Fast and deterministic
Outcome	Reproducible tests

Deeper exploration

When	Scheduled or pre-release
Focus	Longer fuzz sessions
Guide	Coverage and useful cases
Outcome	Reviewed corpus updates

Routine tests → **Replay corpus**

Fast feedback → Catch regressions

Longer runs → **Triage findings**

Explore inputs → Keep useful cases

Fuzzing coverage across Metal³ components

Goal: solid fuzzing coverage in every Metal³ project — where we are today and how many more targets each needs.

IPAM

4

targets today

+7 more

for solid coverage

Goal: 11 total

BMO

3

targets today

+20 more

for solid coverage

Goal: 23 total

CAPM3

1

targets today

+13 more

for solid coverage

Goal: 14 total

IrSO

1

targets today

+27 more

for solid coverage

Goal: 28 total

9 fuzz targets today → 67 more to reach 76 — solid coverage across IPAM, BMO, CAPM3 and IrSO

Questions & Feedback

Breaking Things Safely, Finding Bugs Before They Find You: Fuzzing Metal³ Through Controlled Chaos

