Ecoscape v2: Declarative Performance Resilience Benchmarking in the Edge-Cloud Continuum

Hendrik Reiter
hendrik.reiter@email.uni-kiel.de
Christian-Albrechts-University, Kiel, DE

Wilhelm Hasselbring
hasselbring@email.uni-kiel.de
Christian-Albrechts-University, Kiel, DE

Abstract

Developing software for the Edge-Cloud continuum is challenging for engineers who lack access to physical testbeds but need to know how hardware faults impact their Service Level Objectives (SLOs). Existing tools often rely on loose, external scripts, hindering reproducible resilience testing. This paper introduces Ecoscape v2, a benchmarking framework designed to evaluate application performance resilience. Ecoscape provides a unified, declarative model to model edge-cloud topologies, reproducible chaos scenarios, and monitoring of application SLOs in one place. The framework fully automates the experiment lifecycle and provides a quantitative resilience score directly to the user. A case study featuring a stream-processing pipeline demonstrates how Ecoscape enables developers to systematically verify system behaviour under hardware faults. We evaluate Ecoscape by validating its reproducibility. The results show that in the evaluated scenario Ecoscape measures the same qualitative behaviour under injected CPU stress but with significant quantitative variance.

1 Introduction

Edge and fog computing paradigms [1, 3] process data closer to its source, enabling low-latency responses for real-time applications. However, developing software for these distributed, resource-constrained environments presents significant challenges due to frequent network variations and hardware faults [3]. Software engineers building edge-fog applications need early feedback on how their system behaviour and SLOs deteriorate under volatile conditions to design resilient software services.

To evaluate and improve edge application resilience during development, engineers require realistic, fault-prone testing environments. In practice, accessing physical, heterogeneous edge and fog testbeds is costly and impractical. Discrete simulation [1] tools overcome hardware limits but lack direct portability to container orchestration platforms like Kubernetes. Conversely, existing Cloud Chaos Engineering [2] tools often neglect edge-specific topological constraints. Crucially, there is a lack of empirical evidence demonstrating whether synthetic fault injections in emulated Kubernetes environments yield reproducible results across repeated experiment runs.

We address these challenges by providing software engineers with a declarative benchmarking tool. Thereby the paper contributes the following:

- *The Ecoscape Operator*: We present a native Kubernetes Operator that extends the Ecoscape framework. It enables developers to define edge-fog topologies, chaos scenarios, and automated experiment lifecycles through declarative custom resources. We provide a open-source implementation *Ecoscape v2* as a full rewrite of the Ecoscape-Benchmark [3] as a full Kubernetes-operator.
- *Empirical Reproducibility Analysis*: We evaluate whether Ecoscape v2 provides a reliable benchmarking foundation by analysing if SLOs exhibit consistent qualitative behaviour under identical fault conditions. Specifically, we measure the data processing lag of a distributed stream pipeline over eight automated experiment runs.

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 outlines the architecture of the Ecoscape Operator. Section 4 presents the experimental setup, the reproducibility findings and discusses the results. Section 5 concludes with future work.

2 Related Work

Existing benchmarking and testing methodologies for fault-prone environments can be categorized into cloud chaos engineering, physical hardware testbeds, and discrete simulation platforms.

Tools like *Frisbee* [2] automate chaos injection and fault-tolerance testing within cloud environments. However, these tools are not edge-specific. They neglect the heterogenous topologies, constrained nodes, and varying latency characteristics inherent to edge-fog environments. Simulation tools like *EdgeCloudSim* [1] emulate edge-cloud continuums without hardware dependencies. Nevertheless, discrete event simulations lack direct portability to real-world deployment environments. They cannot run containerized production workloads or interact with native orchestration platforms like Kubernetes. The original

Ecoscape [3] concept introduced fault-tolerance benchmarking for Kubernetes-based environments. However, its initial design lacked developer usability, requiring manual execution steps and rigid scenario setups without a declarative interface. The Go-based Ecoscape Operator resolves this by introducing native Custom Resource Definitions (CRDs) for automated and reproducible experimentation.

3 Design & Implementation

Ecoscape is implemented as a Kubernetes-native operator and is publically available on Github[1]. Ecoscape relies on two external dependencies: Prometheus for metrics collection and SLO evaluation, and Chaos Mesh for fault injection. The entire benchmark configuration is expressed declaratively through three CRDs: `Topology`, `ChaosPhase`, and `Experiment`.

The `Topology` CRD (shown in Figure 1) describes the logical structure of the edge environment. Each zone represents an isolated network domain with configurable resource limits (e.g. CPU, memory, and storage) reflecting the hardware constraints of real edge devices. Connections between zones are declared as links and model realistic WAN characteristics such as latency, jitter, bandwidth, and packet loss. The topology can be activated and deactivated, so that network conditions are selectively applied for the duration of an experiment while the deployed system environment is preserved.

The `ChaosPhase` CRD defines a reusable fault scenario that is specified independently of any concrete experiment. A fault scenario consists of one or more fault definitions referencing named zones of the associated topology. Supported fault types include network-based faults (latency, bandwidth, packet loss, partitioning) as well as resource-based faults (CPU and memory stress), covering typical stress scenarios encountered in edge operation. Additionally, arbitrary platform-specific fault scenarios can be integrated via a generic extension mechanism.

The `Experiment` CRD is the central orchestration resource. It references a `Topology` and a `ChaosPhase`, and defines the benchmark lifecycle: following a configurable warm-up phase, a baseline measurement is conducted without active faults, succeeded by a chaos measurement phase with injected faults. This cycle is executed for a configurable number of repetitions. System quality is assessed through SLOs, declared as PromQL expressions.

4 Evaluation & Discussion

Experiment Setup. To assess the reproducibility of Ecoscape, the following SUT was simulated within the framework. The used CRDs and the result file are published publically on Zenodo[2]. The SUT is an edge sensor pipeline consisting of three logical zones: two edge zones (`zone1`, `zone2`) and one fog zone. Each edge zone runs a redis instance acting as a local stream buffer, and a simulated sensor producer that continuously writes readings to the local redis stream. The fog zone hosts an aggregator service that maintains a consumer group per zone, continuously reading from both edge redis instances and exposing aggregated metrics via Prometheus. The topology is declared as a `Topology` CR and defines baseline network conditions as links: fog ↔ zone1 at 30 ms / 3 ms jitter, fog ↔ `zone2` at 50 ms / 5 ms jitter. The injected fault is defined as the `ChaosPhase` which applies 80 % CPU stress on the fog zone, simulating compute pressure on the aggregator. The `Experiment` CR references the topology and the ChaosPhase. The lifecycle runs for 8 repetitions, each consisting of a warm-up phase, a 90 s pre-chaos baseline measurement, fault injection, and a 90 s chaos measurement window, followed by a recovery pause. System quality is assessed through two SLOs, both measuring redis stream consumer group lag.

```yaml
apiVersion: ecoscape.cau-se.de/v1alpha1
kind: Topology
metadata:
  name: two-zone-edge
  namespace: default
spec:
  zones:
    - name: fog
    - name: zone2
  links:
    - zones: [fog, zone2]
      latency: "50ms"
      jitter: "5ms"
```

Figure 1: Simplified instance of the Topology CRD used in the evaluating experiment. It configures network latency of 50ms with jitter of 5ms between the fog node and the edge node in `zone2`

Results. To assess the impact of the injected chaos fault on each SLO chaos impact δ is computed as the ratio between the pre-chaos baseline and the chaos phase: $\delta = (\mu_{\text{chaos}} - \mu_{\text{pre}})/\mu_{\text{pre}}$, where μ denotes the per-phase mean. The mean and standard deviation of δ across all repetitions characterise both the magnitude and the consistency of the observed effect. Table 1 shows the chaos impact δ for each zone.

Table 1: Summary of the chaos impact (δ) across eight repetitions per SLO.

SLO	mean(δ)	std(δ)	min(δ)	max(δ)
Z1	+1.48	1.53	−0.32	+4.56
Z2	+25.83	10.52	+7.19	+41.49

For `zone2`, the chaos fault produces a large and

[1] https://github.com/HenryWedge/ecoscape-v2
[2] https://zenodo.org/records/22093520

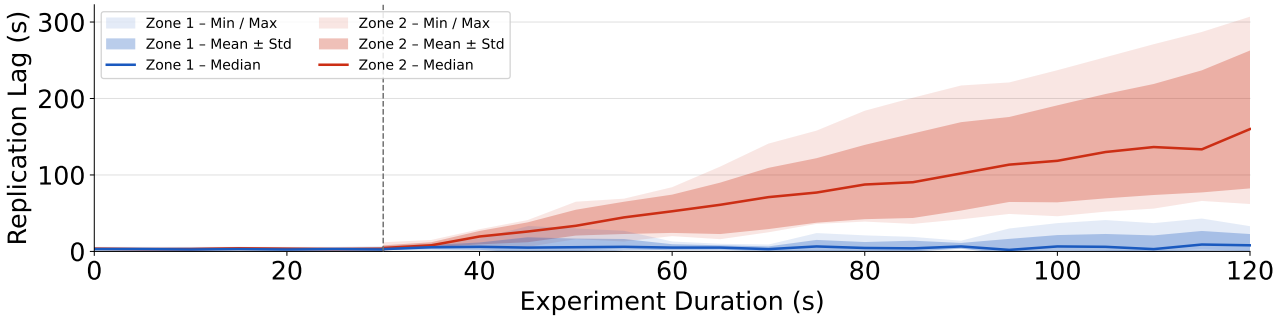

Figure 2: Combined view of `stream-lag-zone2` (red) and `stream-lag-zone1` (blue) over time. Shaded bands show min/max (light) and mean ± std (medium) across eight repetitions; solid lines show the median. The dashed line marks the transition from pre-chaos baseline to chaos phase.

consistently positive effect across all eight repetitions (mean $\delta = +25.83$, std $= 10.52$). For `zone1`, the picture is more nuanced. In absolute terms, `zone1`'s values remain negligibly small across all runs (std(δ) $=$ 1.53), staying consistently at negligible backpressure under CPU stress. However, the standard deviation of the chaos impact is close to the mean (mean $\delta = +1.48$, std $= 1.53$), with one repetition even showing a marginal decrease ($\delta = -0.32$). Figure 2 shows the development of the consumer group lag in both zones over time with its standard deviations.

Discussion. To evaluate the reproducibility of Ecoscape, we distinguish between qualitative trend consistency and quantitative metric variance across identical execution runs. The empirical data confirms that Ecoscape provides qualitative reproducibility. In all eight benchmark repetitions, the system behaviour responded consistently to the injected CPU stress: `zone2` exhibited a massive, continuous increase in stream consumer lag (minimum observed effect $\delta = +7.19$), whereas `zone1` remained unaffected near the baseline. This demonstrates that Ecoscape reliably captures the system's structural degradation trends under continuous fault conditions. While the qualitative outcome is deterministic, the exact magnitude of degradation shows non-negligible variance (`zone2` std(δ) $= 10.52$). The high relative standard deviation (compared to the mean) in `zone1` shows that the chaos fault does induce major relative fluctuations in `zone1`'s stream lag. However, the absolute impact remains far below `zone2`. This quantitative deviation may stem from non-deterministic overheads inherent to container-emulated multi-node environments, including host OS scheduling jitter or runtime garbage collection. Consequently, executing single-run chaos experiments is insufficient for resilience verification. The automated lifecycle of Ecoscape directly addresses this threat by orchestrating continuous multi-run benchmarking, allowing developers to derive statistically sound confidence intervals rather than relying on noisy point measurements.

5 Conclusion

In this paper, we presented the reimplementation of Ecoscape as a native Kubernetes Go-Operator, delivering a fully declarative performance resilience benchmark for the edge-cloud continuum. Our empirical evaluation demonstrated high qualitative reproducibility, reliably capturing systemic failure trends across repeated executions, while also highlighting inherent quantitative variance in containerized emulation environments.

In future work, we plan to leverage Ecoscape to compare diverse edge systems and architectures, as well as validate its external validity by executing benchmark experiments on physical edge hardware testbeds.

Acknowledgments

This work received funding from the Deutsche Forschungsgemeinschaft (DFG), grant 496119880.

References

[1] C. Sonmez, A. Ozgovde, and C. Ersoy. "EdgeCloudSim: An environment for performance evaluation of Edge Computing systems". In: *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, May 2017, pp. 39–44.

[2] F. Nikolaidis et al. "Event-Driven Chaos Testing for Containerized Applications". In: *High Performance Computing*. Springer Nature Switzerland, 2023, pp. 144–157.

[3] H. Reiter et al. "Ecoscape: Fault Tolerance Benchmark for Adaptive Remediation Strategies in Real-Time Edge ML". In: *2025 IEEE International Conference on Quantum Software (QSW)*. IEEE, 2025, pp. 275–280.