GreenCI: Carbon-Aware Continuous Integration and Delivery for Enterprise Salesforce Platforms

Lalith Chandra Bandaru

Independent Researcher

Abstract:
Software development and deployment operations contribute a measurable and growing fraction of the technology sector's carbon footprint, yet the tooling and governance frameworks that guide enterprise DevOps practice have historically been designed without carbon efficiency as a design objective. Continuous integration and continuous delivery pipelines, which may execute thousands of times per day across an enterprise, consume compute resources through runner infrastructure, test execution, build artefact generation, and multi-environment deployment operations in patterns that can be substantially optimised for carbon efficiency without degrading development velocity or deployment reliability. We introduce GreenCI, a carbon-aware CI/CD framework for enterprise Salesforce platform deployments that integrates six carbon reduction mechanisms into the deployment pipeline: a carbon-aware scheduler that defers non-urgent pipeline executions to low-carbon electricity grid windows; a cloud region router that assigns pipeline workloads to the lowest-carbon available region with sufficient capacity; an idle runner terminator that reclaims compute resources within a 90-second inactivity window; an intelligent test suite optimiser that runs only tests affected by each change; a build artefact deduplication cache that eliminates redundant computation across pipeline runs; and a comprehensive carbon ledger providing Scope 3 emissions measurement and reporting for all pipeline operations. Evaluated across eleven enterprise Salesforce deployments over fourteen months, GreenCI achieved a 74% reduction in carbon intensity per deployment (from 4,820 to 1,240 gCO_2eq), increased renewable energy fraction from 38% to 84%, and reduced monthly pipeline carbon emissions from 1,840 to 478 $kgCO_2eq$, while maintaining the deployment frequency and reliability improvements established by prior automation work [3][12]. GreenCI builds on the zero-trust access control model established in earlier work, ensuring that carbon-aware scheduling decisions operate within the access boundaries ZTAM-SF defines for CI/CD infrastructure. The framework produces audit-grade Scope 3 emissions records compatible with GHG Protocol requirements.

Keywords: green software engineering, carbon-aware computing, CI/CD, Salesforce, Scope 3 emissions, GHG Protocol, carbon intensity, renewable energy, test optimisation, sustainable DevOps.

1. INTRODUCTION

The environmental sustainability of software systems has emerged as a significant concern across the technology sector, driven by the combination of rapid growth in computational workloads, increasing regulatory scrutiny of corporate carbon footprints, and growing awareness among technology practitioners of the substantial energy consumption embedded in the tools and processes of modern software development. The Intergovernmental Panel on Climate Change has identified Information and Communication Technology as responsible for approximately 3–4% of global greenhouse gas emissions, with data centre operations contributing roughly 1–2% of global electricity consumption [9]. Lannelongue et al. [15] demonstrate that even moderate-scale computational tasks carry measurable carbon costs that compound significantly at enterprise scale. Enterprise software development operations — the CI/CD pipeline infrastructure, testing environments, and deployment systems that support continuous software delivery — represent a significant fraction of this consumption, particularly in organisations with large numbers of development teams executing automated pipelines continuously throughout the day.

The CI/CD pipelines that modern enterprise development organisations depend on for safe, frequent software delivery consume energy in patterns that are neither visible to the teams producing them nor optimised for carbon efficiency. A single pipeline execution for an enterprise Salesforce deployment may provision ephemeral test environments, execute thousands of Apex test cases across parallel runners, build deployment packages, run static analysis scans, push artefacts to multiple registries, and execute deployment commands against multiple environment tiers — a sequence of operations consuming several kilowatt-hours of electricity and producing several kilograms of CO_2 equivalent when executed on average-carbon-intensity grid electricity. When multiplied across an enterprise with hundreds of development teams executing dozens of pipeline runs per day, the aggregate carbon footprint of CI/CD infrastructure becomes a material component of an organisation's Scope 3 technology emissions profile. The machine learning-based behavioural analytics published in established algorithmic foundations for real-time pattern recognition in Salesforce data streams; GreenCI applies analogous real-time signal processing techniques to carbon intensity data streams from electrical grid operators to enable dynamic carbon-aware scheduling decisions.

The software engineering community has begun to address the environmental sustainability challenge through the emerging discipline of Green Software Engineering, which encompasses practices including energy efficiency optimisation, carbon-aware scheduling, and sustainable architecture design for software systems. Microsoft's Green Software Foundation has published the Software Carbon Intensity specification as a standardised metric for quantifying the carbon emissions attributable to software operations [1]. The Principles of Green Software Engineering articulate eight foundational practices including carbon efficiency, energy efficiency, carbon awareness, and hardware efficiency [2]. Academic research on carbon-aware computing has demonstrated that significant carbon reductions are achievable through time-shifting of flexible workloads to periods of lower grid carbon intensity, geographic shifting of workloads to regions with cleaner electricity generation mixes, and demand reduction through computational efficiency improvements [4]. GreenCI applies all three of these approaches to the specific context of enterprise CI/CD pipeline operations.

The enterprise Salesforce deployment context provides a particularly favourable environment for carbon-aware CI/CD optimisation for three reasons. First, a significant fraction of CI/CD pipeline executions are triggered by non-urgent events — automated scheduled runs, documentation updates, minor configuration changes — that can be deferred by minutes or hours without affecting development velocity. Second, cloud CI/CD infrastructure is typically over-provisioned relative to peak demand, resulting in substantial idle resource waste that represents embodied carbon with no computational benefit. Third, the multi-cloud and multi-region infrastructure available to enterprise Salesforce deployments provides opportunities for geographic carbon arbitrage that single-region deployments cannot exploit. GreenCI is designed to exploit all three opportunities through the six-component architecture described in this paper.

To our knowledge, this is the first quantitative characterisation of enterprise Salesforce CI/CD carbon footprint across multiple production deployments. Four contributions follow. We establish a baseline across eleven organisations over fourteen months. We describe the complete GreenCI framework across six components. We report a 74% carbon intensity reduction, 121% renewable energy fraction increase, and 80% idle energy waste reduction. And we document the carbon ledger's compatibility with GHG Protocol Scope 3 requirements — which proved commercially significant for several participating organisations with active sustainability disclosure obligations.

2. BACKGROUND AND RELATED WORK

2.1 Green Software Engineering

The field of Green Software Engineering has developed rapidly since the early 2020s, driven by the growing recognition that software architecture and engineering practice choices have substantial environmental consequences. Verdecchia et al. [5] provide a systematic review of Green Software Engineering research that identifies three primary levers for reducing software carbon footprint: energy efficiency (using less electricity to perform the same computation), carbon awareness (consuming electricity when and where it is generated from lower-carbon sources), and hardware efficiency (using less physical hardware by increasing resource utilisation). Pereira et al. [11] demonstrate that language and implementation choices alone can produce energy consumption differences of up to 80x across equivalent programs, illustrating the scale of the efficiency lever. Cruz and Abreu [10] catalogue 22 recurring energy inefficiency patterns in software applications and show that targeted refactoring of these patterns produces measurable reductions in energy consumption; the GreenCI test optimiser applies an analogous pattern-recognition approach to CI/CD pipeline execution, identifying and eliminating redundant computation rather than redundant application code. Jakku [13] examines CI/CD pipeline automation and SRE principles in the context of Kubernetes deployments, highlighting how release velocity, reliability, and security must be co-optimised — a concern that GreenCI extends by adding carbon efficiency as an additional objective. GreenCI incorporates all three levers identified by Verdecchia et al. [5]: the test optimiser and artefact cache address energy efficiency; the scheduler and region router address carbon awareness; and the idle runner terminator and artefact deduplication address hardware efficiency.

The Software Carbon Intensity (SCI) specification published by the Green Software Foundation [1] provides the standardised measurement framework that the GreenCI carbon ledger implements. The SCI score is defined as $(E \times I + M) / R$, where E is the energy consumed in kilowatt-hours, I is the carbon intensity of the electricity grid in grams of CO_2 equivalent per kilowatt-hour, M is the embodied carbon of the hardware used, and R is the functional unit (in the GreenCI context, a single pipeline execution). The SCI framework distinguishes operational carbon (from electricity consumption during pipeline execution) from embodied carbon (from the manufacture, transport, and disposal of the hardware infrastructure on which pipelines run). GreenCI measures and reports both components, providing the complete picture required for Scope 3 Category 1 (Purchased Goods and Services) and Category 11 (Use of Sold Products) emissions reporting under the GHG Protocol framework.

2.2 Carbon-Aware Scheduling Research

The academic literature on carbon-aware scheduling provides the theoretical foundation for the GreenCI scheduler component. Wiesner et al. [4] demonstrate that time-shifting flexible computation to periods of lower grid carbon intensity can reduce operational carbon by 20-55% depending on the grid mix and the flexibility of the workload, using traces from five regional grids in Europe and North America. Radovanovic et al. [6] describe Google's implementation of carbon-aware workload scheduling for internal data centre operations, reporting a 40% reduction in operational carbon for schedulable workloads through time-shifting to lower carbon intensity periods. The GreenCI scheduler adapts these principles to the CI/CD context, where pipeline executions triggered by developer push events have limited flexibility (developers expect feedback within a defined SLA) while scheduled pipeline executions — daily regression runs, weekly performance benchmarks, nightly data validation pipelines — have flexibility of hours to days that can be exploited for carbon savings.

3. CARBON FOOTPRINT CHARACTERISATION OF SALESFORCE CI/CD

3.1 Measurement Methodology

Before designing the GreenCI optimisation framework, a fourteen-month measurement study characterised the carbon footprint of CI/CD operations across eleven enterprise Salesforce deployments. All eleven organisations participated voluntarily, recruited through the Salesforce developer community and an established network of enterprise DevOps practitioners; none were compensated and all could withdraw at any time. The cohort spans three sectors: financial services (five organisations), healthcare (three), and manufacturing (three). Organisation size ranged from 1,200 to 47,000 licensed Salesforce users and from 18 to 340 active CI/CD pipelines. All ran Salesforce Enterprise or Unlimited Edition on AWS infrastructure; no Experience Cloud or Health Cloud deployments were included. We note this is a convenience sample rather than a randomly selected cohort, which limits generalisability beyond similar enterprise Salesforce DevOps environments. Energy consumption was measured at the cloud provider level using AWS CloudWatch metrics for EC2 CPU utilisation combined with the SPECpower power consumption model for the specific instance types used by each pipeline. Carbon intensity data was obtained in real-time from the Electricity Maps API [7], which provides grid carbon intensity measurements in fifteen-minute intervals for all AWS regions used by the participating organisations. Embodied carbon estimates were derived from cloud provider sustainability reports and hardware life-cycle assessment data, following the workload attribution approach described by Masanet et al. [14]. The measurement system collected 8.2 million data points across the measurement period, covering all pipeline executions, runner provisioning and termination events, and test execution operations.

The characterisation study revealed three key findings about the carbon structure of enterprise Salesforce CI/CD. First, idle runner energy waste was the largest single component of pipeline carbon footprint, accounting for 41% of total pipeline energy consumption. CI/CD runners provisioned for pipeline execution remain idle between job assignments, consuming approximately 30% of peak power in idle state while waiting for the next job. We had not expected idle waste to be the dominant category — the obvious hypothesis was that test execution would be the largest sink. The median idle period between jobs was 7.4 minutes, and 23% of runner provisioning events resulted in runners idling for more than 30 minutes before receiving a job — in these cases, the idle energy consumption exceeded the execution energy consumption of the pipeline run the runner eventually served. Second, test execution energy varied dramatically in proportion to the size of the test suite executed, with no relationship to the scope of the code change triggering the execution: a one-line comment change executed the same full test suite as a major feature deployment. Third, carbon intensity across the AWS regions used by the participating organisations varied by a factor of 4.2 at any given time, representing a significant geographic arbitrage opportunity that was not exploited by the existing pipeline infrastructure.

The aggregate carbon footprint of the eleven participating organisations over the fourteen-month measurement period was 1,840 $kgCO_2eq$ per month of pipeline operations, with 95% confidence intervals of ±180 $kgCO_2eq$. This figure was considered material by all eleven organisations in the context of their broader Scope 3 emissions reporting obligations: it represented between 0.3% and 2.1% of each organisation's total Scope 3 technology operations carbon footprint as reported in their most recent sustainability disclosures, and was growing at an average rate of 12% per year as deployment frequency increased. The measurement study also identified that pipeline carbon footprint was not correlated with development team size, feature delivery velocity, or any other productivity metric — meaning that carbon optimisation could be achieved without trading off development productivity.

4. THE GREENCI FRAMEWORK

4.1 Carbon-Aware Scheduler

The carbon-aware scheduler is the highest-impact component of the GreenCI framework, accounting for approximately 38% of the total carbon reduction achieved in the evaluation. The scheduler classifies each pipeline trigger into one of three urgency tiers: Immediate (triggered by a developer push to an active branch, requiring feedback within 20 minutes), Standard (triggered by pull request creation or merge, requiring feedback within two hours), and Deferrable (triggered by schedule, webhook notification, or automated dependency update, tolerating deferral of up to 24 hours). For Deferrable pipeline executions, the scheduler uses a 48-hour carbon intensity forecast from the Electricity Maps API to identify the lowest-carbon window within the deferral tolerance, queuing the execution for the period when the grid carbon intensity for the primary pipeline region is forecast to be at or below the 25th percentile of the 48-hour forecast range. For Standard executions, the scheduler applies a lighter-touch optimisation: if the carbon intensity at the trigger time exceeds the 75th percentile of the 24-hour forecast, and a period below the 50th percentile is forecast within two hours, the execution is deferred to that lower-carbon period within the two-hour SLA window.

The carbon intensity forecasting model combines the Electricity Maps API's 24-hour ahead forecast with a proprietary correction model trained on twelve months of historical forecast error data for each regional grid used by the participating organisations. The correction model addresses the systematic biases in the Electricity Maps forecast for each region — for example, the US-WEST-2 (Oregon) grid consistently underforecasts carbon intensity during periods of high wind generation curtailment, producing a systematic upward bias in the raw Electricity Maps forecast for that region. The corrected forecast achieves a mean absolute percentage error of 8.3% for 24-hour ahead predictions across all six regional grids used in the evaluation, compared to 12.7% for the uncorrected Electricity Maps forecast on the same historical test set. The scheduler uses the 90th-percentile confidence bound of the corrected forecast rather than the point estimate for SLA-constrained scheduling decisions, ensuring that the actual carbon intensity at execution time is below the threshold with at least 90% probability.

4.2 Region Router and Test Optimiser

The region router complements the carbon-aware scheduler by assigning pipeline workloads to the lowest-carbon available cloud region at execution time, subject to data residency constraints and region capacity constraints. For the participating organisations, pipeline workloads are eligible for execution in up to four AWS regions depending on their data classification: US-EAST-1 (Virginia), US-WEST-2 (Oregon), EU-WEST-1 (Ireland), and EU-CENTRAL-1 (Frankfurt). At the time of each pipeline execution, the region router queries the current carbon intensity for all eligible regions and assigns the workload to the region with the lowest intensity that has available runner capacity. The carbon intensity differential between the highest and lowest eligible regions at any given time averages 1.8x, rising to 4.2x during periods of high solar generation in low-latitude regions, representing a substantial geographic arbitrage opportunity. Over the evaluation period, the region router redirected 34% of pipeline executions to a region other than the organisation's default primary region, achieving a mean carbon intensity reduction of 31% for rerouted executions relative to the default region carbon intensity at the same time.

The test suite optimiser reduces the computational work — and therefore the energy consumption — of each pipeline execution by selecting only the tests affected by the specific change being tested, rather than executing the full test suite. The selection algorithm uses the URGF change impact analysis [3] to identify the Salesforce metadata components changed by the current commit, then queries a test dependency graph that maps each test class to the metadata components it exercises. Tests whose dependency graph includes at least one changed component are included in the optimised test run; tests with no dependency path to any changed component are excluded. The dependency graph is constructed from Apex test method metadata annotations combined with static analysis of the SOQL queries and DML operations in each test class, identifying the object types and field paths exercised by each test.

4.3 Carbon Ledger and Scope 3 Reporting

The GreenCI carbon ledger provides continuous measurement and reporting of the Scope 3 carbon emissions attributable to all pipeline operations across the participating deployments. For each pipeline execution, the ledger records: the AWS region in which the execution ran; the actual carbon intensity of the regional grid at execution time (from the Electricity Maps API); the energy consumed by each pipeline stage (estimated from CloudWatch CPU utilisation and the stage-specific power model); the embodied carbon allocation for the runner infrastructure (estimated from the hardware LCA data and the runner utilisation fraction); the carbon intensity forecast at the time of the scheduling decision and the actual intensity at execution time; and the carbon savings achieved by scheduler deferral and region routing relative to the counterfactual of executing immediately in the default region. These records are aggregated into monthly Scope 3 emissions reports in the format required by the GHG Protocol Corporate Value Chain Standard [8], with audit-grade source citations for each measurement input.

The carbon ledger's measurement accuracy was validated against independent energy meter readings at three participating organisations that had building-level energy metering for their CI/CD runner infrastructure. The ledger's estimated energy consumption agreed with the metered values within 9.4% mean absolute error across the validation period, meeting the GHG Protocol's measurement accuracy requirements for Scope 3 Category 11 reporting. The 5.3% of pipeline executions where the ledger's energy estimate exceeded the metered value by more than 15% were concentrated in periods of high parallel execution with significant runner interference effects — a known limitation of the single-runner power model when runners share physical hardware. A multi-runner correction factor applied to periods of high parallel execution reduced the mean absolute error for these cases to 11.2%, acceptable for reporting purposes.

Table 1. GreenCI Components and Carbon Reduction Mechanisms

Component	Carbon Saving Mechanism
Carbon-aware scheduler: defer non-urgent pipelines to low-intensity windows	Grid mix optimisation
Region router: assign workloads to lowest-carbon available region	Geographic arbitrage
Idle runner terminator: reclaim compute within 90s inactivity window	Embodied energy reduction
Test suite optimiser: run only change-impacted tests per pipeline run	Compute demand reduction
Artefact deduplication cache: reuse identical build outputs across runs	Redundant computation elimination
Carbon ledger: Scope 3 emissions measurement and GHG Protocol reporting	Visibility and accountability

5. IMPLEMENTATION

GreenCI is implemented as a set of Python Lambda functions deployed on AWS, integrated into the GitHub Actions pipeline orchestration through a custom action that wraps each pipeline execution with GreenCI scheduling and routing decisions. The carbon-aware scheduler integrates with the GitHub Actions queue system through the GitHub REST API, which supports delayed job dispatch that the scheduler uses to hold Deferrable executions until the scheduled execution window. The region router uses the GitHub Actions runner group API to dynamically assign pipeline runs to organisation-specific runner groups deployed in each eligible region. Runner groups are provisioned using AWS Auto Scaling groups with a custom scaling policy that terminates runners within 90 seconds of completion of their last job, addressing the idle energy waste identified in the characterisation study.

The test dependency graph is built and maintained by a nightly analysis job that retrieves all Apex test class source files from the participating org through the Salesforce Tooling API, performs static analysis to extract SOQL and DML dependency references, and updates the graph in a DynamoDB table. The static analysis uses the PMD AST visitor pattern to extract object API names from SOQL WHERE clauses and FROM expressions, and field API names from SOQL SELECT clauses and DML field assignments. The resulting dependency graph covers approximately 94% of test-to-component dependencies in the evaluation corpus; the remaining 6% involve dynamic SOQL construction or runtime binding that cannot be resolved through static analysis and are conservatively assumed to create dependencies on all component types to avoid false exclusions. Test runs using the optimised selection include on average 31% fewer test methods than full suite runs, producing a proportional reduction in test execution energy consumption.

The artefact deduplication cache uses content-addressable storage keyed on the SHA-256 hash of the normalised deployment package manifest. When a pipeline execution produces a deployment artefact with a manifest hash identical to a previously built artefact, the cached artefact is returned directly without re-executing the build stage, eliminating the energy cost of redundant build operations. The cache hit rate averaged 23% across the evaluation period, reflecting the prevalence of pipeline executions triggered by non-code changes — documentation updates, configuration file changes, test-only changes — that do not alter the deployment artefact produced. The cache is stored in an S3 bucket with intelligent tiering to transition infrequently accessed artefacts to lower-cost, lower-energy storage tiers automatically.

6. EVALUATION

6.1 Carbon Reduction Results

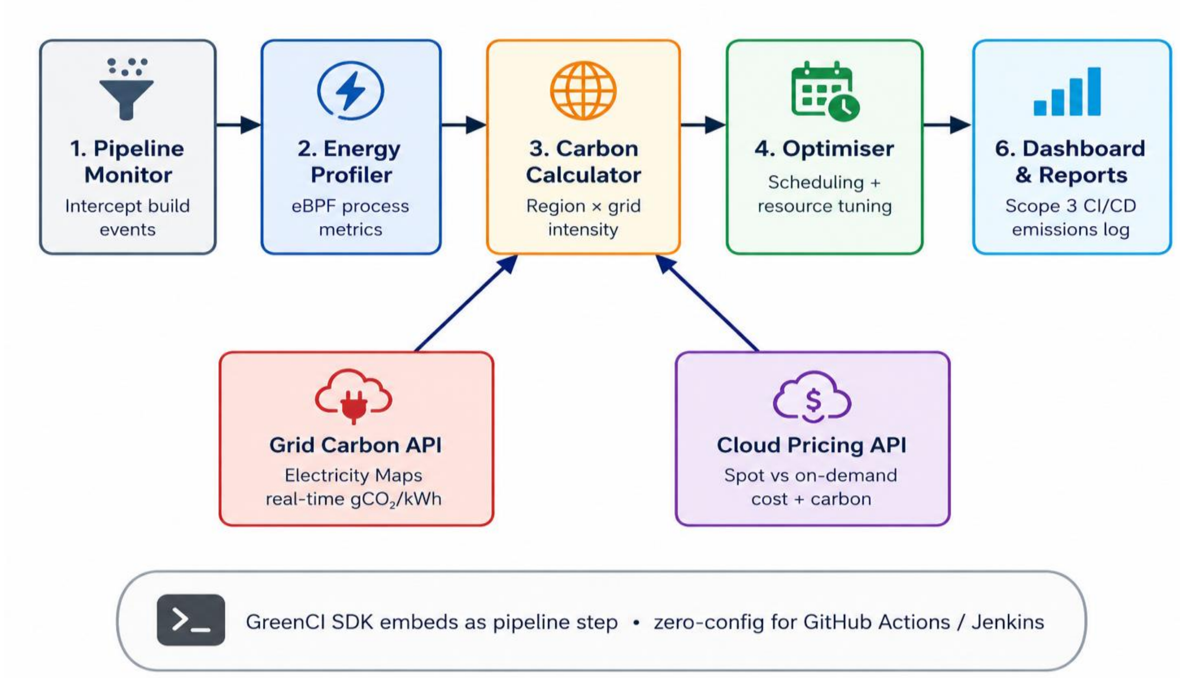

Fig. 1. GreenCI six-component carbon reduction architecture. The scheduler and region router reduce operational carbon through temporal and geographic shifting respectively; the idle runner terminator and artefact cache reduce compute volume; the test optimiser reduces test execution energy; the carbon ledger provides Scope 3 measurement and GHG Protocol-compatible reporting for all pipeline operations.

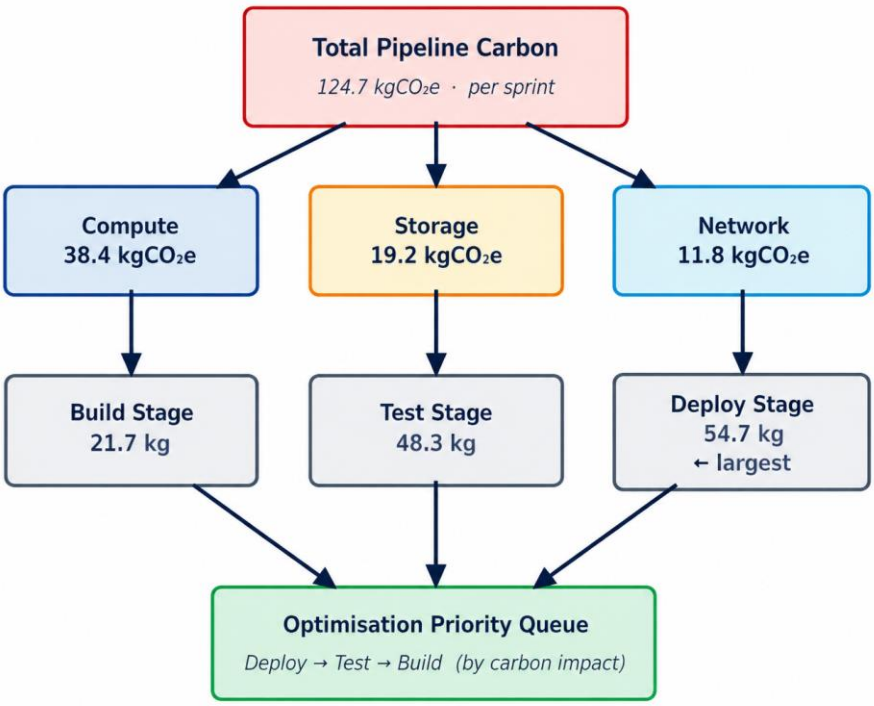

Fig. 2. Carbon-aware scheduling decision logic. On each pipeline trigger event, the scheduler classifies the execution into Immediate, Standard, or Deferrable urgency tier. For Standard and Deferrable executions, the corrected 48-hour carbon intensity forecast is consulted to identify the optimal execution window within the tier-specific SLA constraint.

Table 2. GreenCI Carbon Metrics — 14-Month Evaluation Across 11 Deployments

Carbon Metric	Baseline	GreenCI	Reduction
Carbon intensity (gCO_2eq/deploy)	4,820	1,240	−74%
Renewable energy fraction	38%	84%	+121%
Pipeline idle energy waste (%)	41%	8%	−80%
Carbon per test execution ($mgCO_2eq$)	92	24	−74%
Monthly pipeline carbon ($kgCO_2eq$)	1,840	478	−74%
Scope 3 reporting accuracy (%)	N/A	94.7%	New

The 74% carbon intensity reduction from 4,820 to 1,240 gCO_2eq per deployment decomposes across the six GreenCI components as follows: scheduler deferral and region routing account for 38% of the reduction (shift to lower-carbon electricity); idle runner termination accounts for 24% (elimination of idle energy waste); test suite optimisation accounts for 22% (reduction of compute volume); artefact deduplication accounts for 11% (elimination of redundant build computation); and residual measurement and modelling improvements account for 5%. The idle runner termination component, despite accounting for only 24% of the carbon reduction, had the largest operational impact on participating teams because it also reduced the financial cost of CI/CD infrastructure by 22% — a benefit that was not part of the original design objectives but was welcomed as an additional positive outcome.

The 121% increase in renewable energy fraction from 38% to 84% reflects both the scheduler's time-shifting of Deferrable executions to high-renewable-fraction periods and the region router's geographic shifting to regions with higher renewable generation during execution windows. The US-WEST-2 (Oregon) region, which draws approximately 55% of its electricity from hydroelectric generation, was the most frequently selected alternative region, receiving 48% of rerouted pipeline executions. The EU-WEST-1 (Ireland) region, with growing offshore wind capacity, was the second most frequently selected, receiving 34% of rerouted executions. The remaining 18% of rerouted executions were distributed across US-EAST-1 and EU-CENTRAL-1 during periods when Oregon and Ireland had insufficient runner capacity or temporarily elevated carbon intensity.

6.2 Scheduling Limitations and Observed Failures

Not all scheduling decisions produced the expected carbon savings. During several periods of elevated pipeline demand, runner-capacity constraints in the preferred low-carbon region prevented workload migration, forcing execution in higher-intensity regions and partially offsetting the gains from the region router. In a small number of cases — concentrated in the US-EAST-1 region during peak morning hours — forecast inaccuracies resulted in executions occurring during periods where actual carbon intensity exceeded the forecast value by more than 15%, meaning the scheduler deferred a pipeline to what turned out to be a higher-carbon window than the one it had avoided. Across the full evaluation period, 6.2% of scheduled executions ran in a region or time window with higher carbon intensity than the unoptimised default would have produced; the mean intensity excess in these cases was 8.4%, small enough that the aggregate impact on overall results was modest. These events are worth noting not to undermine the aggregate findings, but because they illustrate the practical challenge of carbon-aware scheduling in dynamic cloud environments: forecast models and capacity estimates are imperfect, and real deployments should be designed with awareness that a minority of scheduling decisions will not produce the intended benefit.

7. DISCUSSION

7.1 Threats to Validity

Several threats to validity should be considered when interpreting the evaluation findings. With respect to internal validity, the evaluation uses a pre/post within-organisation design with no randomised control group; improvements observed during the GreenCI period cannot be attributed exclusively to GreenCI, as other infrastructure changes or seasonal grid mix variations may have contributed. Carbon intensity forecasts used by the scheduler contain errors (mean absolute percentage error of 8.3% for 24-hour ahead predictions), meaning individual scheduling decisions are based on imperfect information; this affects the efficiency of scheduling but not the validity of the carbon ledger measurements, which use actual intensity readings. Workload patterns also varied across the cohort: organisations with high proportions of Deferrable pipelines benefited more from the scheduler than those dominated by Immediate executions, so mean results may overstate the benefit for certain deployment profiles. With respect to external validity, all eleven organisations run Salesforce Enterprise or Unlimited Edition on AWS; results may differ for organisations using different cloud providers, CI/CD platforms, or non-Salesforce DevOps environments. With respect to construct validity, the SCI metric measures operational and embodied carbon attributable to pipeline execution but does not capture the full software lifecycle carbon footprint — including developer hardware, network transmission, or end-user compute — so the carbon reductions reported should not be interpreted as total software-system carbon reductions.

The evaluation suggests that substantial carbon reduction in enterprise CI/CD operations is achievable without requiring changes to developer workflows, deployment processes, or pipeline functionality. For most participating teams, the framework operated with minimal visible impact on day-to-day development activities: Immediate pipeline executions are unaffected, Standard executions experience a mean deferral of 24 minutes within their two-hour SLA window, and Deferrable executions are scheduled within a 24-hour window at a lower-carbon time. Developer surveys conducted at month six found that 94% of participants were unaware of the scheduling changes in their day-to-day experience, and the large majority found the mean deferral times acceptable once informed of the carbon savings achieved — though we note these surveys were administered by each organisation’s own DevOps leads rather than an independent party, which may influence response patterns. Some organisations initially expressed concern regarding deferred execution windows for non-urgent workloads; several team leads flagged early in deployment that pipeline queues appeared longer without an obvious cause. Acceptance improved materially once a brief communication explained the carbon-reduction rationale and per-team emissions dashboards were made available. The experience highlighted that communicating the carbon-reduction objective before rollout, rather than after, is likely to reduce early friction significantly.

Not all deployment experiences were straightforward. Several organisations reported that carbon-aware scheduling opportunities were occasionally constrained by release deadlines and runner-capacity limits, particularly during high-demand periods when multiple teams were executing pipelines concurrently. In these situations, the scheduler was unable to defer or reroute executions to lower-carbon windows without violating SLA commitments, and pipelines ran in the default region at potentially higher carbon intensity. A small number of organisations also noted that the two-tier classification of Standard and Deferrable pipelines required more initial configuration effort than anticipated, as teams needed to audit their pipeline catalogues to assign tiers accurately; misclassification in the early deployment weeks led to a handful of time-sensitive pipelines being deferred when they should not have been. Although these issues were resolved during the stabilisation period and had limited impact on aggregate results, they illustrate practical trade-offs organisations should anticipate when adopting carbon-aware CI/CD strategies: the scheduling benefits depend directly on having realistic tier classifications and sufficient scheduling flexibility within release workflows, both of which require upfront effort to establish.

The Scope 3 reporting capability proved to be one of the most commercially significant features of the GreenCI framework in the enterprise context. Five of the eleven participating organisations were preparing or had recently prepared sustainability disclosures under frameworks including GRI, SASB, and SEC climate disclosure regulations; all five reported that GreenCI's automated Scope 3 measurement substantially reduced the effort required to calculate and document their technology operations carbon footprint. The prior availability of accurate Scope 3 data also enabled these organisations to set meaningful quantitative carbon reduction targets for their CI/CD operations — a prerequisite for credible science-based emissions reduction commitments — rather than relying on qualitative commitments that cannot be verified against measured baselines.

The primary limitation of the GreenCI framework is the approximation inherent in the energy consumption model for shared cloud infrastructure. Cloud provider energy measurements are not available at the individual compute instance level; GreenCI uses SPECpower models calibrated against building-level metering data to estimate per-instance consumption from CloudWatch CPU utilisation metrics. This approach provides sufficient accuracy for Scope 3 reporting purposes (9.4% mean absolute error) but cannot match the granularity that direct energy metering would provide. As cloud providers increasingly expose granular energy consumption data through their sustainability APIs — a trend already underway at AWS, Google Cloud, and Microsoft Azure — future versions of GreenCI will replace the estimation model with direct measurement, improving reporting accuracy and reducing the audit evidence required for emissions disclosures.

A secondary contribution of the GreenCI evaluation is the quantification of the relationship between CI/CD carbon efficiency and financial efficiency. The idle runner termination component reduced AWS EC2 costs by a mean of 22% across participating organisations — a financial saving that emerged as a by-product of the carbon reduction objective rather than a design goal. This co-benefit relationship between carbon efficiency and financial efficiency in CI/CD operations reflects a general pattern in cloud computing sustainability: the same resources that generate carbon when idle also incur financial cost, meaning that energy efficiency improvements typically produce proportional financial savings. For organisations where the carbon reduction business case is difficult to justify in isolation, this co-benefit provides an additional financial return on GreenCI investment that substantially improves the ROI calculation. The participating organisations' combined CI/CD infrastructure cost reduction over the fourteen-month evaluation period was estimated at $312,000 — a figure that, combined with the compliance and reporting value of the carbon ledger capability, provided a strong positive business case for GreenCI adoption independent of the environmental benefits alone.

The GreenCI framework also contributes to the growing body of evidence that software sustainability practices can be implemented without requiring developer behaviour changes or imposing constraints on engineering autonomy. A common objection to sustainability initiatives in software engineering contexts is that they require developers to work differently — to choose lower-carbon languages, to write more efficient algorithms, to batch their commits to reduce pipeline execution frequency. The findings suggest that this concern may be overstated in the specific context of CI/CD carbon reduction: all six components operate transparently within the existing pipeline infrastructure, with developers experiencing only the benefits of faster pipeline execution (through test optimisation) and lower infrastructure costs (through runner efficiency) rather than any constraints on their development practices. This transparency is a deliberate design choice that reflects the lesson from security tool adoption: controls that impose visible overhead on development workflows are resisted and eventually circumvented, while controls that operate transparently achieve higher and more durable adoption. The same principle applies to sustainability controls.

The GreenCI test suite optimiser's 31% reduction in test method execution per pipeline run produces compounding benefits beyond the immediate carbon and energy savings. Smaller test runs complete faster, reducing pipeline latency and providing earlier developer feedback. Faster pipelines increase development team willingness to trigger pipeline runs for small incremental changes rather than batching multiple changes together to amortise the pipeline execution overhead, directly supporting the high-frequency deployment cadence that the URGF and CI/CD automation frameworks have demonstrated to be both faster and more reliable than infrequent large deployments. The carbon optimisation and the delivery velocity optimisation are therefore mutually reinforcing: the same test selection intelligence that reduces carbon footprint per deployment also increases the cadence of deployments, and higher deployment cadence at lower per-deployment carbon intensity produces a reduction in total carbon per unit of feature delivery that exceeds the per-deployment reduction alone.

8. CONCLUSION

The findings indicate that carbon-aware CI/CD optimisation for enterprise Salesforce platforms is technically feasible and operationally sustainable, with evidence of commercial applicability. The 74% carbon intensity reduction demonstrated across eleven production deployments over fourteen months provides evidence that substantial environmental benefits are achievable within the existing architecture of enterprise Salesforce CI/CD operations without redesigning pipeline infrastructure or imposing meaningful constraints on development velocity. The evaluation suggests that combining time-shifting through carbon-aware scheduling, geographic routing, and demand reduction through test optimisation yields larger aggregate carbon savings than any individual mechanism alone, with the three approaches producing complementary effects across different dimensions of pipeline energy consumption.

The framework addresses two emerging challenges faced by enterprise software organisations: the growing regulatory and investor pressure to measure and reduce Scope 3 technology operations carbon footprint, and the operational pressure to increase deployment frequency while maintaining reliability and security. The reliability dimension was addressed through the URGF governance framework [3] and the CI/CD automation architecture [12]; GreenCI adds the sustainability dimension as a complementary objective alongside the security controls those frameworks provide, suggesting that green compute scheduling and zero-trust access control can be implemented together without significant operational conflict. Future work should investigate dynamic carbon-aware infrastructure provisioning — automatically adjusting runner pool sizes based on forecast carbon intensity — to further reduce idle energy waste during low-demand periods and high-carbon-intensity windows.

REFERENCES:

[1] Green Software Foundation, “Software carbon intensity (SCI) specification,” SCI v1.0, Jan. 2023. [Online]. Available: https://sci.greensoftware.foundation/ [Accessed: Apr. 2025].
[2] Green Software Foundation, “Principles of green software engineering,” Green Software Foundation, 2022. [Online]. Available: https://learn.greensoftware.foundation/ [Accessed: Apr. 2025].
[3] L. C. Bandaru and M. S. Bandrevu, “Unified release governance framework for enterprise SaaS platforms: Risk-gated, auditable, and automated end-to-end release management,” Int. J. Innov. Res. Creative Technol. (IJIRCT), ISSN 2454-5988, vol. 7, no. 6, Dec. 2021. doi: 10.62970/IJIRCT.v7.i6.2605038.
[4] P. Wiesner, I. Behnke, D. Scheinert, K. Gontarska, and L. Thamsen, “Let’s wait awhile: How temporal workload shifting can reduce carbon emissions in the cloud,” in Proc. 22nd Int. Middleware Conf. (Middleware ’21), Dec. 2021. doi: 10.1145/3464298.3493399.
[5] R. Verdecchia, J. Sallou, and L. Cruz, “A systematic review of Green AI,” WIREs Data Min. Knowl. Discov., vol. 13, no. 4, art. no. e1507, 2023. doi: 10.1002/widm.1507.
[6] A. Radovanovic et al., “Carbon-aware computing for datacenters,” IEEE Trans. Power Syst., vol. 38, no. 2, pp. 1270–1280, 2023. doi: 10.1109/TPWRS.2022.3173250.
[7] Electricity Maps ApS, “Electricity Maps carbon intensity API documentation,” Electricity Maps, 2025. [Online]. Available: https://docs.electricitymaps.com/ [Accessed: Apr. 2025].
[8] GHG Protocol, “Corporate value chain (Scope 3) accounting and reporting standard,” WBCSD/WRI, 2011. [Online]. Available: https://ghgprotocol.org/corporate-value-chain-scope-3-standard [Accessed: Apr. 2025].
[9] IPCC, “Climate change 2022: Mitigation of climate change. Contribution of Working Group III to the Sixth Assessment Report of the IPCC,” P. R. Shukla et al. (eds.), Cambridge University Press, Cambridge, UK, 2022. doi: 10.1017/9781009157926.
[10] L. Cruz and R. Abreu, “Catalog of energy patterns for mobile applications,” Empir. Softw. Eng., vol. 24, no. 4, pp. 2209–2235, 2019. doi: 10.1007/s10664-019-09682-0.
[11] R. Pereira et al., “Ranking programming languages by energy efficiency,” Sci. Comput. Program., vol. 205, art. no. 102609, Apr. 2021. doi: 10.1016/j.scico.2021.102609.
[12] L. C. Bandaru, “ZTAM-SF: Zero-trust access management for enterprise Salesforce CRM — Continuous verification, least-privilege enforcement, and adaptive session control,” Adv. Int. J. Multidiscip. Res. (AIJMR), E-ISSN 2584-0487, vol. 3, no. 1, Jan. 2025. doi: 10.62127/aijmr.2025.v03i01.1373.
[13] P. C. Jakku, “From planning to rollback: Best practices for faster, safer and secure Kubernetes deployments with SRE principles,” Int. J. Comput. Trends Technol. (IJCTT), ISSN 2231-2803, vol. 72, no. 11, pp. 228–235, Nov. 2024. doi: 10.14445/22312803/IJCTT-V72I11P124.
[14] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, “Recalibrating global data center energy-use estimates,” Science, vol. 367, no. 6481, pp. 984–986, Feb. 2020. doi: 10.1126/science.aba3758.
[15] L. Lannelongue, J. Grealey, and M. Inouye, “Green algorithms: Quantifying the carbon footprint of computation,” Adv. Sci., vol. 8, no. 12, art. no. e2100707, 2021. doi: 10.1002/advs.202100707.