Journal of Information Systems Engineering and Management
2025, 10(62s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

Predictive Hybrid Autoscaling for Cloud Workloads: A Machine Learning Approach to Vertical and Horizontal Resource Optimization on AWS EC2

Karthikeyan Rajamani

Independent Researcher, USA

ARTICLE INFO

Received: 08 Oct 2025

Revised: 22 Nov 2025

Accepted: 02 Dec 2025

ABSTRACT

Cloud infrastructure management faces persistent challenges in balancing service reliability with cost efficiency as workload demands fluctuate unpredictably. Traditional autoscaling approaches rely on reactive, threshold-based triggers that respond only after performance degradation begins, whereas AWS native solutions provide exclusive horizontal scaling capabilities. This article introduces a predictive hybrid autoscaling framework that leverages machine learning forecasting models to anticipate resource demands and intelligently orchestrates both vertical instance resizing and horizontal capacity adjustments. Three time-series prediction approaches—ARIMA, Long Short-Term Memory networks, and Facebook Prophet—are evaluated in a comparative study across diverse workload patterns, including periodic traffic cycles, sudden demand spikes, gradual growth trends, and unpredictable variations. The article testbed employs multiple AWS EC2 instance families under realistic application scenarios, measuring Service Level Objective compliance, availability metrics, resource utilization efficiency, and total infrastructure costs. Results demonstrate that predictive hybrid scaling substantially improves reliability while reducing operational expenses compared to reactive autoscaling and static over-provisioning strategies. LSTM networks excel at capturing complex non-linear demand patterns, while Prophet proves superior for seasonal workloads. The article integrates with standard AWS services through open-source implementations, providing cloud operators with practical tools for proactive capacity management. This article bridges predictive analytics with site reliability engineering practices, offering systematic approaches to cost-reliability optimization in dynamic cloud environments.

Keywords: Predictive Autoscaling, Hybrid Scaling, Machine Learning Forecasting, Cloud Reliability Engineering, Service Level Objectives

Introduction

Cloud infrastructure management has become increasingly complex as organizations grapple with unpredictable workload patterns and stringent reliability requirements. Modern applications experience dramatic traffic fluctuations—from baseline operations to sudden spikes during peak hours or viral events. These variations create a persistent challenge: how to maintain service quality without hemorrhaging resources during quiet periods or crashing during demand surges.

AWS Auto Scaling represents the de facto solution for many cloud operators, yet its reactive nature and horizontal-only approach leave significant gaps [1]. When traffic suddenly increases, threshold-based triggers respond only after performance degradation has already begun. This lag frequently results in violated Service Level Objectives, frustrated users, and emergency firefighting by on-call engineers. Meanwhile, purely horizontal scaling—adding or removing identical instances—ignores a fundamental optimization dimension: changing the capacity of existing resources.

The missing piece involves anticipation rather than reaction. If infrastructure could predict demand fifteen minutes ahead, scaling decisions could happen before users experience slowdowns.

Journal of Information Systems Engineering and Management
2025, 10(62s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

Additionally, combining vertical scaling (resizing instances) with horizontal scaling (adjusting instance counts) opens new optimization possibilities that balance cost against reliability in ways previously unexplored.

This research introduces a hybrid autoscaling framework that leverages machine learning forecasting models to predict resource needs and intelligently orchestrates both vertical and horizontal scaling decisions. By comparing ARIMA, LSTM, and Prophet models across diverse workload patterns, the study demonstrates measurable improvements in SLO compliance while reducing infrastructure costs. The approach bridges predictive analytics with practical cloud operations, providing a systematic method for organizations seeking reliable and cost-effective infrastructure management.

2. Background and Related Work

2.1 Cloud Autoscaling Fundamentals

Cloud autoscaling operates through two primary mechanisms. Horizontal scaling adjusts the number of compute instances—spinning up additional servers during traffic peaks and terminating them when demand subsides. Vertical scaling modifies the capacity of individual instances by changing their CPU, memory, or storage specifications. AWS EC2 offers diverse instance families optimized for specific workloads: compute-optimized (C-series), memory-optimized (R-series), and general-purpose (M-series) types [2]. Native autoscaling policies include target tracking (maintaining specific metrics), step scaling (threshold-based adjustments), and scheduled scaling (time-based changes).

2.2 Limitations of Native AWS Autoscaling

CloudWatch-based triggers fundamentally react to problems rather than preventing them. Metrics must breach thresholds before action occurs, guaranteeing performance degradation during the response window. AWS provides no native vertical scaling; instance resizing requires manual stop-modify-start sequences. Cold start delays—the time needed for new instances to become operational—can span several minutes, during which users experience degraded service.

2.3 Time-Series Forecasting in Cloud Computing

ARIMA models decompose historical patterns into autoregressive and moving average components, effectively capturing linear trends. LSTM networks excel at learning complex temporal dependencies through recurrent connections, making them suitable for non-linear workload patterns. Facebook's Prophet explicitly models trends, seasonality, and holiday effects, performing well on data with strong periodic components.

2.4 Existing Autoscaling Research

Academic literature predominantly focuses on rule-based threshold systems or reinforcement learning approaches. Recent work explores proactive scaling using demand prediction, though implementations rarely address vertical-horizontal hybrid strategies.

2.5 Service Level Objectives and Reliability Engineering

SLOs quantify acceptable service quality through metrics like response latency and availability percentages. Violations directly impact user experience and carry measurable business costs.

2.6 Research Gaps

Current research insufficiently addresses hybrid scaling architectures, comparative ML model performance in production environments, and systematic cost-reliability optimization frameworks.

Journal of Information Systems Engineering and Management
2025, 10(62s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

3. System Architecture and Design

3.1 Hybrid Autoscaling Framework Overview

The proposed architecture integrates predictive analytics with AWS native services through a multi-layered design. CloudWatch streams infrastructure metrics to a centralized monitoring layer, which feeds time-series data into machine learning models hosted on AWS Lambda functions [3]. Predictions flow into a decision engine that orchestrates both vertical instance modifications and horizontal Auto Scaling Group adjustments. This hybrid approach enables the system to resize existing instances when capacity adjustments suffice, while adding or removing instances for larger demand shifts.

3.2 Metrics Collection and Monitoring

Comprehensive observability requires metrics across multiple dimensions. Infrastructure telemetry captures CPU utilization, memory consumption, disk throughput, and network bandwidth. AWS-specific indicators like CPU steal time reveal noisy neighbor effects on shared hardware, while T-series credit balances indicate burstable instance exhaustion. Application-level measurements track request latency distributions, message queue depths, and HTTP error rates. Business metrics including transaction volumes and active user sessions, contextualize technical performance. Data flows through CloudWatch for AWS-native metrics, supplemented by Prometheus for custom application instrumentation, with long-term storage in TimescaleDB for historical analysis.

3.3 Predictive Modeling Pipeline

3.3.1 Data Preprocessing

Raw metric streams undergo normalization to eliminate scale differences between CPU percentages and request counts. Deseasonalization removes predictable daily or weekly patterns, allowing models to focus on anomalous variations. Feature engineering constructs lag variables capturing historical dependencies, rolling statistics smoothing short-term noise, and temporal indicators encoding hour-of-day or day-of-week effects. Outlier detection identifies deployment events or incidents that would distort training. The dataset splits chronologically into training, validation, and test segments to prevent temporal leakage.

3.3.2 ARIMA-based Forecasting

ARIMA models decompose time series into autoregressive, integrated, moving average, and integrated moving average components. Parameter selection uses autocorrelation and partial autocorrelation function analysis, with model quality assessed through Akaike and Bayesian information criteria. Stationarity verification via Augmented Dickey-Fuller tests ensures mathematical validity. The implementation generates fifteen-minute-ahead forecasts with confidence intervals quantifying prediction uncertainty.

3.3.3 LSTM Neural Network Approach

The LSTM architecture processes sequences of historical metrics through recurrent layers with dropout regularization, preventing overfitting. Hyperparameter tuning explores sequence lengths from thirty to one hundred twenty minutes, hidden layer configurations, and learning rates. The network accepts multivariate inputs—combining CPU, memory, and request rates—to capture cross-metric dependencies. Multi-step prediction generates forecasts across multiple future intervals simultaneously.

3.3.4 Prophet Model Implementation

Prophet decomposes demand into additive components: piecewise linear trends, multiple seasonal patterns, and holiday effects [4]. Automatic changepoint detection identifies shifts in growth

Journal of Information Systems Engineering and Management
2025, 10(62s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

trajectories without manual intervention. Uncertainty bands provide calibrated confidence estimates. Custom seasonality captures cloud-specific patterns like weekly deployment schedules or monthly billing cycles.

3.4 Decision Engine

3.4.1 Scaling Decision Logic

Predicted demand triggers scaling when confidence exceeds configurable thresholds. A decision tree evaluates whether vertical or horizontal adjustments better address anticipated needs—vertical scaling handles moderate changes efficiently, while horizontal scaling addresses larger shifts. Instance type selection algorithms map predicted CPU and memory requirements to optimal EC2 families. Hysteresis mechanisms prevent rapid oscillations by requiring sustained threshold breaches before reversing recent decisions.

3.4.2 Vertical Scaling Mechanism

Vertical scaling in EC2 follows the standard AWS stop, modify, start workflow, as instance types cannot be changed while an instance is running. Because this introduces a brief disruption window, workloads typically mitigate the impact using blue-green or standby instance patterns, where traffic is temporarily shifted to a healthy pool during modification.

To automate vertical adjustments safely, the system generates a new version of the Launch Template (or creates a cloned template) with the updated instance size based on prediction. This ensures that scaling events remain predictable and repeatable. New EC2 instances launched during scaling operations use this updated template, enabling seamless adoption of larger (or smaller) sizes without manual reconfiguration. Compatibility checks, such as ensuring the target instance type is within the same family or supports the same virtualization mode and storage, are applied before updating or cloning the launch template to prevent invalid transitions.

3.4.3 Horizontal Scaling Integration

Horizontal adjustments modify the Auto Scaling Group's desired capacity through API calls, coordinating with load balancer health checks. New instances undergo gradual warm-up periods before receiving full traffic loads, while departing instances drain existing connections gracefully.

3.5 Feedback Loop and Model Retraining

Continuous validation compares predictions against realized demand, triggering retraining when drift exceeds thresholds. A/B testing frameworks evaluate candidate models against production baselines before promotion.

Model	Strengths	Limitations	Best Suited For	Training Time	Inference Speed
ARIMA	Fast training and inference, interpretable parameters, and well-established theory	Assumes linear relationships, struggles with non-stationary data, requires manual parameter tuning	Stationary workloads with linear trends, short-term predictions	Minutes	Milliseconds

Journal of Information Systems Engineering and Management
2025, 10(62s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

LSTM	Captures complex non-linear patterns, handles multivariate inputs, and learns long-term dependencies	Requires substantial training data, computationally intensive training, "black box" nature	Bursty traffic, complex patterns, non-linear demand shifts	Hours (GPU-accelerated)	Milliseconds
Prophet	Automatic seasonality detection, robust to missing data, interpretable components, handles holidays/events	Limited flexibility for non-seasonal patterns, assumes additive model structure	Periodic workloads with strong daily/weekly cycles, business applications	Minutes to Hours	Seconds

Table 1: Comparison of Time-Series Forecasting Models for Cloud Workload Prediction [4, 8]

4. Implementation

4.1 Technology Stack

Python serves as the primary development language due to its extensive machine learning ecosystem and AWS integration capabilities. The Boto3 SDK provides programmatic access to AWS services, enabling automated infrastructure management through clean API abstractions [5]. Machine learning components leverage scikit-learn for ARIMA implementations, TensorFlow with Keras for LSTM networks, and Facebook's Prophet library for seasonal decomposition models. Infrastructure provisioning uses Terraform for its cloud-agnostic declarative syntax, allowing infrastructure definitions to version alongside application code. Orchestration happens through AWS Lambda functions triggered on schedules or events, with Step Functions coordinating complex multi-stage workflows when vertical scaling requires careful sequencing.

4.2 AWS Service Integration

CloudWatch APIs retrieve metric streams and manage alarm configurations that complement predictive triggers. EC2 modification procedures invoke stop-instance, modify-instance-attribute, and start-instance calls sequentially, with polling loops monitoring state transitions. Auto Scaling Group APIs adjust desired capacity, minimum, and maximum instance counts based on forecasted demand. IAM role configurations follow least-privilege principles—Lambda functions receive permissions scoped to specific resource ARNs, preventing accidental modifications to unrelated infrastructure [6].

4.3 Prediction Service Architecture

The prediction service deploys as a containerized microservice exposing RESTful endpoints for forecast retrieval. API routes accept metric identifiers and time horizons, returning predictions with confidence intervals in JSON format. Model versioning maintains multiple trained variants simultaneously, enabling seamless rollbacks when new models underperform. A Redis caching layer stores recent predictions, reducing computational overhead for frequently queried forecasts. This architecture separates model training—a resource-intensive batch process—from inference serving, which demands low-latency responses.

Journal of Information Systems Engineering and Management
2025, 10(62s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

4.4 Safety Mechanisms and Constraints

Production deployments require robust guardrails against runaway costs or availability failures. Hard limits cap maximum instance counts regardless of predicted demand, preventing billing surprises from model errors. Budget thresholds trigger alerts when hourly spending exceeds predefined rates, with emergency shutoffs available for severe overspending scenarios. Minimum instance count requirements ensure baseline availability even when models predict zero demand, protecting against catastrophic prediction failures. Rollback procedures automatically revert scaling decisions when health check failures spike, restoring previous configurations within minutes. These safety nets acknowledge that no predictive system achieves perfect accuracy—operational resilience demands graceful degradation when predictions miss their mark.

Component	Specification	Purpose
AWS Region	us-east-1 (3 availability zones)	Geographic distribution and fault tolerance testing
Instance Families	T3 (burstable), M5 (general-purpose), C5 (compute-optimized)	Evaluate scaling across different performance profiles
Workload Types	CPU-intensive (video transcoding), Memory-intensive (in-memory caching), Balanced (web serving)	Assess framework performance under varied resource demands
Load Generators	Apache JMeter, Locust, Custom simulators	Realistic traffic pattern simulation
Monitoring Stack	CloudWatch, Prometheus, TimescaleDB	Comprehensive metrics collection and storage
Training Period	30-60 days of historical data	Establish baseline patterns for ML models
Evaluation Duration	30 days per workload type	Statistical significance and long-term performance assessment

Table 2: Experimental Testbed Configuration [3-9]

5. Experimental Setup

5.1 Testbed Configuration

The experimental environment is deployed across the AWS US-East-1 region, spanning three availability zones to ensure realistic distribution and fault tolerance. Instance type evaluation focused on three representative families: T3 (burstable baseline performance), M5 (general-purpose balanced compute), and C5 (compute-optimized workloads). Application workloads varied in resource consumption patterns—CPU-intensive operations like video transcoding, memory-intensive data processing with large in-memory caches, and balanced web application serving. Load generation employed Apache JMeter for HTTP traffic simulation [7], Locust for distributed load testing with Python-based scenario scripting [8], and custom simulators mimicking specific application behaviors like batch processing jobs.

Journal of Information Systems Engineering and Management
2025, 10(62s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

5.2 Workload Patterns

Four distinct demand patterns replicated real-world scenarios. Periodic patterns exhibited predictable daily cycles with morning traffic ramps and evening declines, characteristic of e-commerce platforms experiencing timezone-driven usage. Bursty patterns introduced sudden traffic spikes lasting minutes to hours, resembling news websites during breaking events or social media viral content propagation. Growing trend workloads simulated SaaS platforms experiencing gradual capacity expansion as user bases increased over weeks. Unpredictable patterns generated random load variations, modeling event-driven architectures or ad-hoc batch processing.

5.3 Baseline Comparisons

Four approaches underwent comparative evaluation. Native AWS Auto Scaling Groups provided reactive horizontal scaling using target tracking policies, monitoring CPU utilization. Static provisioning maintained a fixed capacity sized for peak historical demand, representing traditional over-provisioning strategies. Rule-based hybrid implemented simple threshold logic combining vertical and horizontal adjustments without prediction. The proposed predictive hybrid leveraged machine learning forecasts guiding intelligent scaling decisions.

5.4 Evaluation Metrics

Reliability assessment measured SLO compliance rates by tracking the percentage of measurement windows meeting latency and availability targets. Availability calculations targeted standard thresholds—three nines and four nines uptime. Response time distributions captured at P50, P95, and P99 percentiles revealed tail latency behavior under various scaling strategies. Error rates quantified failed requests per second during capacity transitions.

Resource efficiency metrics included CPU and memory utilization distributions, identifying over-provisioning through the ratio of provisioned minus consumed capacity divided by total provisioned resources. Under-provisioning incidents counted occurrences where demand exceeded available capacity alongside duration measurements.

Cost analysis tracked total EC2 expenses under on-demand pricing, normalized per-transaction costs enabling fair comparisons across different traffic volumes, and waste reduction percentages relative to static provisioning baselines. Scaling behavior captured latency from trigger detection to resource availability, prediction accuracy via mean absolute error and root mean square error metrics, and classification accuracy identifying unnecessary scaling events (false positives) versus missed opportunities (false negatives).

5.5 Experimental Duration

Model training consumed thirty to sixty days of historical metrics, establishing baseline patterns. Each workload type underwent thirty-day live evaluation periods, generating statistically meaningful samples. Paired statistical tests assessed the significance of performance differences between approaches.

Approach	Scaling Type	Trigger Mechanism	Response Time	Vertical Scaling	Horizontal Scaling	Prediction Capability
Native AWS ASG	Horizontal only	Reactive threshold (CloudWatch	3-5 minutes	Not supported	Yes	No [1, 2]

Journal of Information Systems Engineering and Management
2025, 10(62s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

		metrics)				
Static Provisioning	Manual	Pre-configured capacity	N/A	Manual process	Manual process	No
Rule-Based Hybrid	Vertical + Horizontal	Simple threshold logic	2-4 minutes	Yes (manual stop-start)	Yes	No
Predictive Hybrid (Proposed)	Vertical + Horizontal	ML-based forecasting	1-3 minutes (proactive)	Yes (automated)	Yes	Yes (15-60 min ahead)

Table 3: Autoscaling Approach Comparison [1-6]

6. Results and Analysis

6.1 Forecasting Model Performance

6.1.1 Model Accuracy Comparison

LSTM networks demonstrated superior accuracy for complex non-linear workloads, particularly during transition periods between demand states. Prophet excelled with periodic patterns featuring strong seasonality, automatically detecting weekly cycles without manual configuration. ARIMA performed adequately for stationary workloads but struggled with trend shifts. Prediction horizons significantly impacted accuracy—five-minute forecasts achieved substantially lower error rates than one-hour predictions across all models. Statistical testing confirmed LSTM superiority with high confidence levels for bursty workloads, while Prophet marginally outperformed alternatives on periodic patterns.

6.1.2 Model Training and Inference Time

LSTM training required hours on GPU-accelerated instances, but inference completed within milliseconds, satisfying real-time requirements. Prophet trained faster but demanded more computational resources during prediction generation. ARIMA offered the fastest training and inference, trading accuracy for speed.

6.2 Scaling Decision Quality

Predictive approaches triggered scaling events significantly earlier than reactive baselines, providing lead time before performance degradation manifested. Vertical scaling dominated adjustments for moderate demand fluctuations, while horizontal scaling was activated during larger shifts. This hybrid pattern optimized cost-efficiency by avoiding unnecessary instance launches for transient load increases [9].

7. Discussion

7.1 Key Findings Summary

The experimental results demonstrate substantial benefits from predictive hybrid autoscaling compared to traditional reactive approaches. LSTM models captured complex non-linear demand patterns more effectively than ARIMA, while Prophet's strength emerged in workloads with pronounced seasonality. Vertical scaling proved particularly valuable during demand transitions, adjusting capacity faster than horizontal instance launches.

Journal of Information Systems Engineering and Management
2025, 10(62s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

7.2 Theoretical Implications

Predictive modeling fundamentally shifts cloud infrastructure from reactive problem-solving to proactive capacity planning. Hybrid scaling expands the optimization space beyond horizontal-only approaches, enabling finer-grained cost-reliability balance. Confidence threshold tuning provides operators with systematic control over this trade-off.

7.3 Practical Implications for Cloud Operators

Production deployment requires careful model selection aligned with workload characteristics—seasonal applications benefit from Prophet, while variable traffic patterns favor LSTM. Integration with existing Site Reliability Engineering practices demands tooling investments and team skill development in machine learning operations.

7.4 Limitations and Challenges

7.4.1 Vertical Scaling Downtime

AWS stop-modify-start workflows introduce unavoidable downtime during instance resizing. Multi-instance deployments with load balancing mitigate this limitation, though stateful applications face data persistence challenges.

7.4.2 Model Generalization

Novel workload patterns degrade prediction accuracy until retraining occurs. New applications lack historical data, creating cold-start challenges.

7.4.3 Complexity and Operational Overhead

The framework adds system complexity beyond native autoscaling, requiring specialized monitoring and debugging capabilities.

7.5 Comparison with Industry Practices

Google Cloud offers predictive autoscaling features within its managed services [10], though lacking hybrid vertical-horizontal orchestration. The open-source nature of this approach enables broader adoption across cloud providers.

Metric Category	Specific Metrics	Measurement Method
Reliability	SLO compliance rate, Availability (uptime %), P50/P95/P99 latency, Error rate	Continuous monitoring over a 30-day evaluation period
Resource Efficiency	CPU utilization (avg/distribution), Memory utilization, Over-provisioning ratio, Under-provisioning incidents	CloudWatch and custom instrumentation
Cost	Total EC2 costs, Cost per transaction, Waste reduction vs. static provisioning	AWS Cost Explorer, normalized transaction analysis
Scaling Behavior	Scaling latency, Prediction accuracy (MAE, RMSE, MAPE), False positive/negative rates	Time-series analysis and classification metrics
Operational	Vertical vs. horizontal scaling frequency, Instance type transitions, CPU steal time correlation	Event logging and statistical analysis

Table 4: Evaluation Metrics Framework [9]

Journal of Information Systems Engineering and Management
2025, 10(62s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

Conclusion

This article addresses a critical gap in cloud infrastructure management by introducing a predictive hybrid autoscaling framework that combines machine learning forecasting with intelligent vertical and horizontal scaling orchestration. The experimental validation across diverse workload patterns demonstrates that anticipating resource demands rather than reacting to capacity exhaustion yields measurable improvements in service reliability while reducing operational costs. LSTM networks proved particularly effective for capturing complex demand fluctuations, while Prophet excelled at handling seasonal traffic patterns, providing cloud operators with evidence-based model selection guidance. The hybrid approach—resizing existing instances alongside adjusting instance counts—unlocks optimization possibilities that purely horizontal scaling cannot achieve. Despite inherent challenges like vertical scaling downtime and model training requirements, the framework offers practical value for organizations prioritizing Service Level Objective compliance without excessive infrastructure spending. The article leverages standard AWS services and open-source machine learning libraries, lowering adoption barriers compared to proprietary solutions. As cloud workloads grow increasingly dynamic and cost pressures intensify, predictive capacity management transitions from academic curiosity to operational necessity. Future work exploring reinforcement learning policies, multi-cloud implementations, and automated cost-reliability optimization will further refine these approaches. Organizations implementing site reliability engineering practices now possess a validated methodology for bridging reactive infrastructure management with proactive reliability engineering.

References

[1] Amazon Web Services. "Amazon EC2 Auto Scaling User Guide", https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html

[2] Amazon Web Services, "Amazon EC2 Instance Types". https://aws.amazon.com/ec2/instance-types/

[3] Amazon Web Services, "AWS Lambda Developer Guide". https://docs.aws.amazon.com/lambda/

[4] Facebook GitHub, "Prophet: Forecasting at Scale," Facebook Open Source, https://facebook.github.io/prophet/

[5] Amazon Web Services, "Boto3 Documentation", Boto3 1.40.69 documentation. https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

[6] Amazon Web Services, "AWS Identity and Access Management Documentation". https://docs.aws.amazon.com/iam/

[7] Apache Software Foundation, "Apache JMeter". https://jmeter.apache.org/

[8] Locust, "Locust Documentation," https://docs.locust.io/

[9] Amazon Web Services, "Amazon CloudWatch User Guide". https://docs.aws.amazon.com/cloudwatch/

[10] Google Cloud, "Autoscaling groups of instances" https://cloud.google.com/compute/docs/autoscaler