METRICLY®

Cloud Monitoring as a Service

Built On Machine Learning

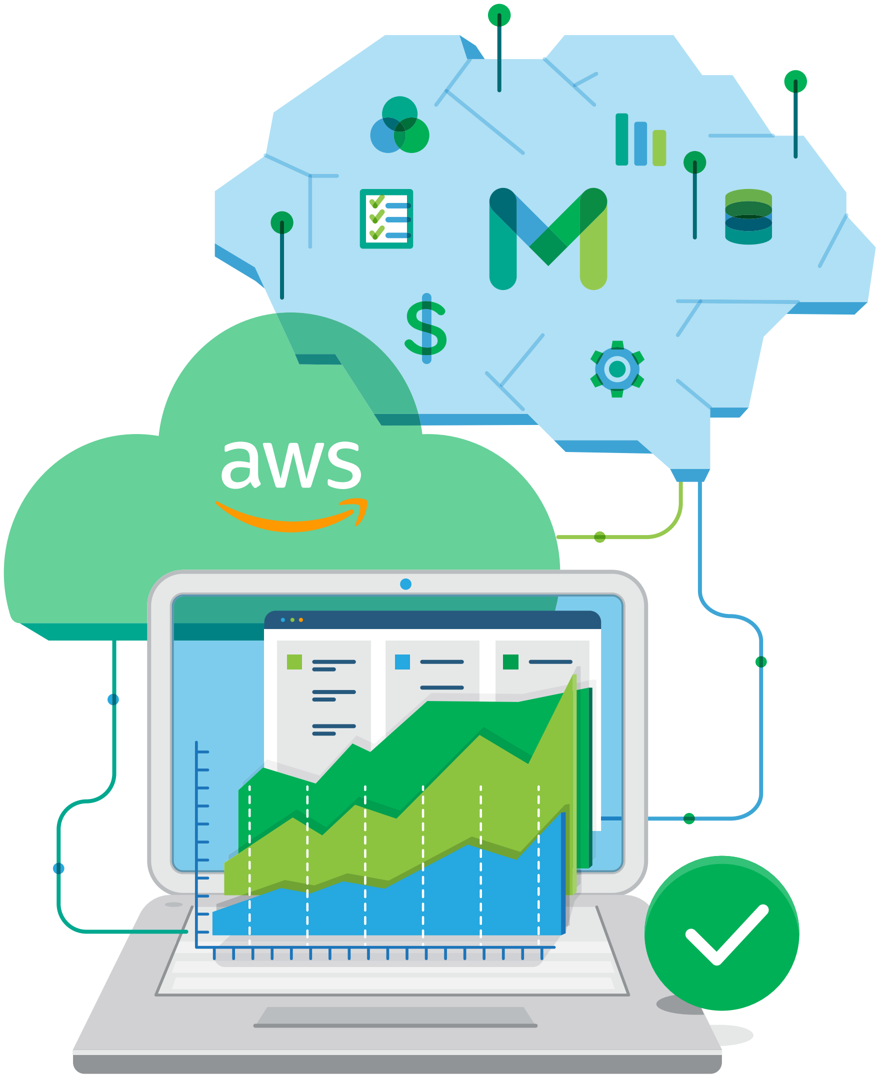


Table of Contents

Why Machine Learning

1 Don't Waste Time

Administering and scaling your own monitoring stack

Creating your own integrations

Setting thresholds one metric at a time

2 Gain Extra Insights

Aggregate data by cluster and by application

See what's abnormal in your environment

Leverage industry best practices for alerting and reporting

3 Don't Forget Capacity & Cost

Reuse the same agent to gather capacity and performance data

Rely on preset metrics to measure multiple dimension of capacity

AWS cost analysis gets complicated fast

Who Cares

DevOps Engineer

I want automation and
I don't want to wake up
in the middle of the night

DevOps Manager

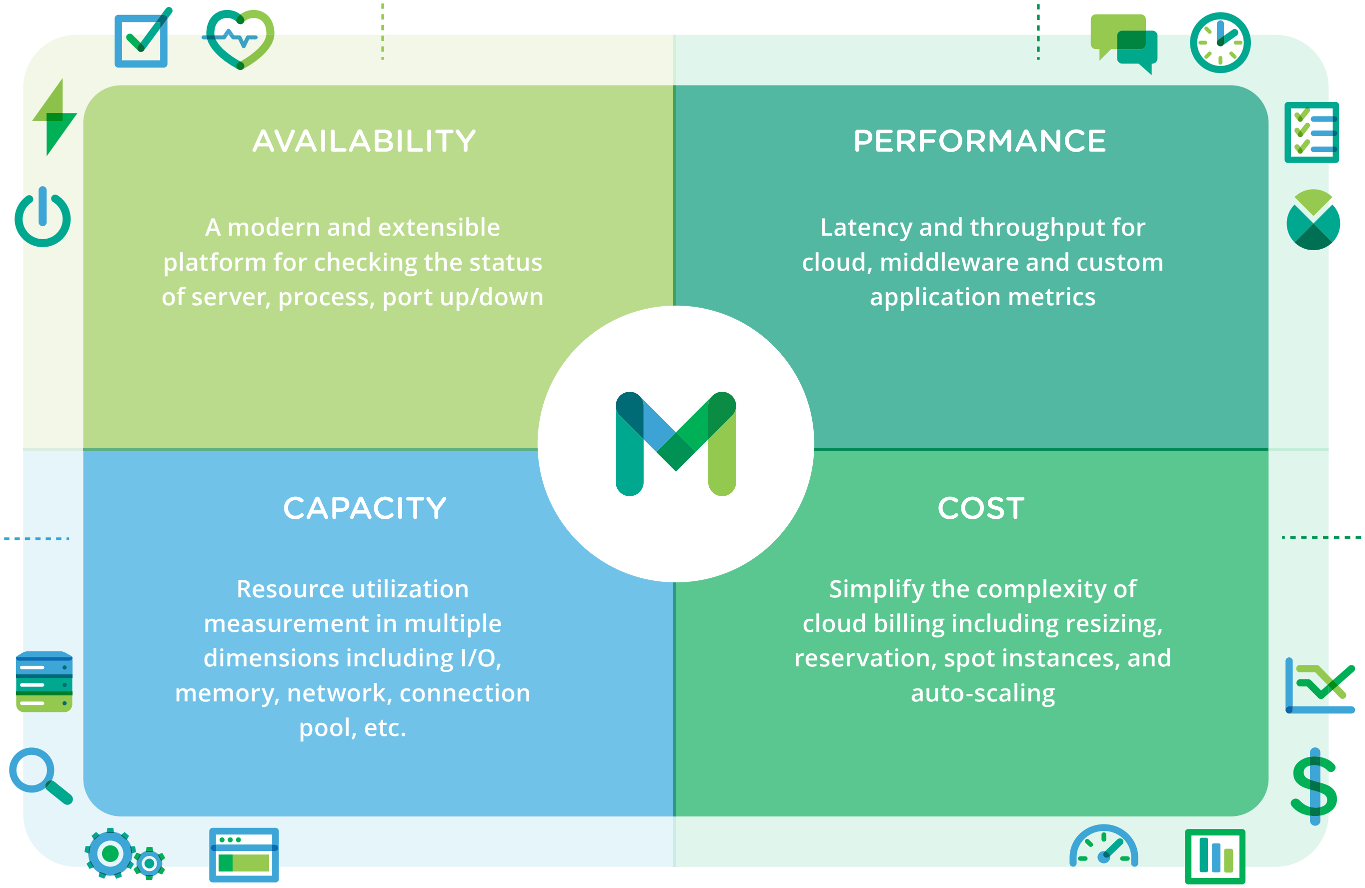
I want my team to be efficient,
I want to avoid outages, and
I don't want to spend weekends
going through AWS bills

Executive

I want our customer-facing
applications to be available
and fast, and our cloud
hosting costs to be low

METRICLY

Four Dimensions To Cloud Monitoring



AVAILABILITY

A modern and extensible platform for checking the status of server, process, port up/down

PERFORMANCE

Latency and throughput for cloud, middleware and custom application metrics

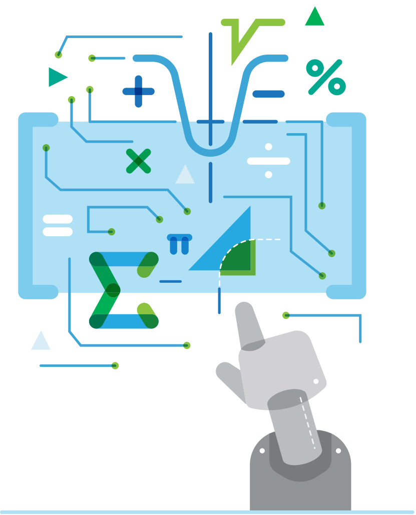
CAPACITY

Resource utilization measurement in multiple dimensions including I/O, memory, network, connection pool, etc.

COST

Simplify the complexity of cloud billing including resizing, reservation, spot instances, and auto-scaling

Data Aggregation



A Function for Each Purpose

For an ever-increasing counter, you want to know the delta over time

For latency, you want to know the weighed average

For errors, you want to know the sum during an interval

Filter and Group Metrics

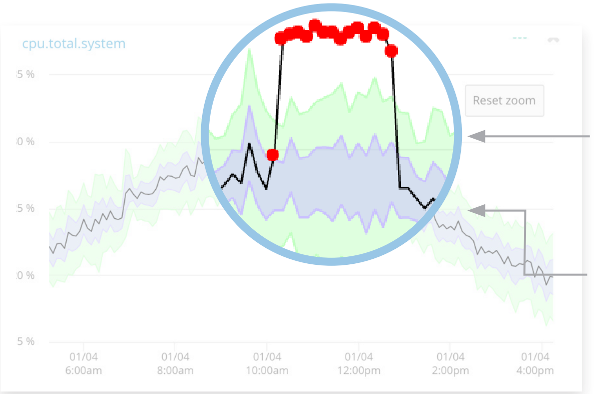
User-specified metric-level aggregation is powerful

For example, you could average CPU utilization across all EC2s with a given application tag in a particular AWS region

Or the average latency of a method call across a cluster of Docker containers

More sophisticated expressions account for attribute meta-data such as hardware rating for IOPS for an m3.xlarge EC2 to identify unusual process queue for I/O, or the number of CPU cores that can be used to normalize the Linux load

Anomaly Detection Algorithms



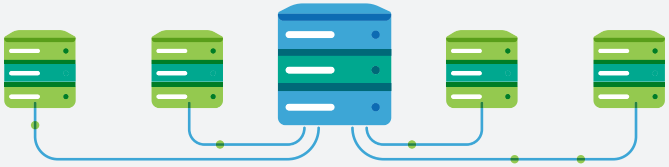
Green Band: Single-Variate. What is the value usually at this time?

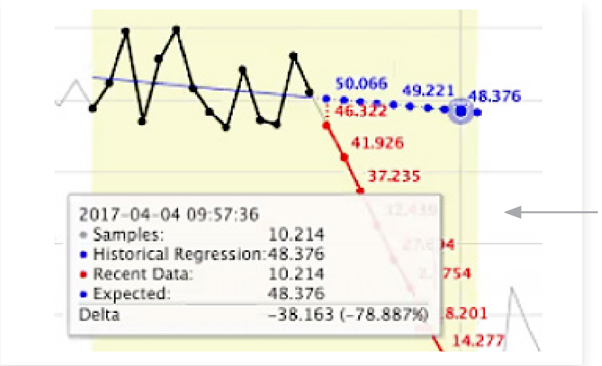
Blue Band: Multi-Variate. What should it be as a function of other correlated metrics?

Multi-variate regression analysis discovers the regression weights and correlation coefficients to determine where a metric should be as a function of other inter-dependent metrics, for a more **accurate** estimation.

Clusters

Anomaly detection on load-balanced clusters of nodes or containers is best accomplished by analyzing the aggregated values across the entire **cluster** vs. at the **ephemeral** node level. The aggregated values conform to the workload patterns driven by the application usage traffic.





```
2017-04-04 09:57:36
• Samples:              10.214
• Historical Regression:48.376
• Recent Data:          10.214
• Expected:             48.376
Delta                   -38.163 (-78.887%)
```

Sudden Change: Is the trend-line drastically changing up or down?

What if you don't have multiple metrics to correlate? What if your green band of normalcy is too wide due to high standard deviation? In those cases, you can detect **sudden** changes in the **trend-lines.** What's important is how steep was the rise and fall and how long it lasted.

Preset Best Practices

Accurate Anomaly Detection

Domain knowledge must complement statistical analysis to achieve an actionable level of accuracy in anomaly detection.

Accuracy
Control

Multi-Variable Statistical Analysis

Learn behavior accurately

Accurate Anomaly Detection

Domain Knowledge Preset Best Practices

Look for combination of deviations

Industry Best Practices

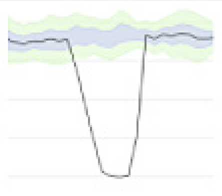
Preset configuration of **dashboards** and **alerts** based on industry best practices avoid the need for manual configuration by end-users upon activation of integrations with common technologies, such as the examples listed below:

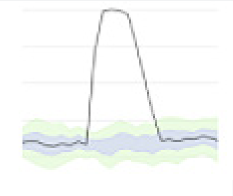
amazon web services docker cassandra elastic

Multi-Conditional Policies

Multi-conditional alerting policies result in control, accuracy, and reasoning that are required to deliver actionable alerts. It is recommended for un-interrupted night of sleep.

DON'T WAKE ME UNLESS...





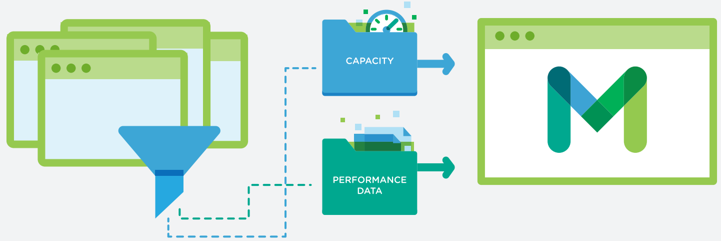
1 | I/O wait > 20%

2 | Database connections drop

3 | ELB round-trip latency jumps

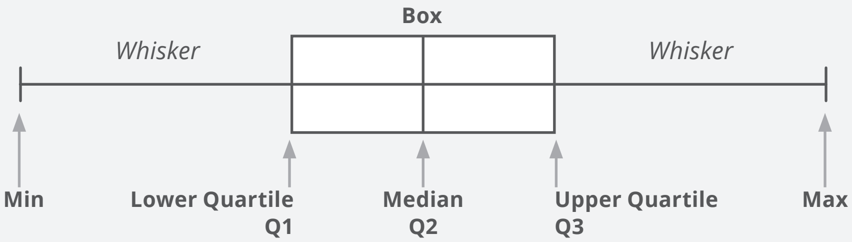
Capacity Analysis

Your application platform has finite computing capacity (0 to 100%), but there are multiple types of resources. Some are **intuitive** to measure (ex. CPU). Some can be measured if you know the meta-data **attribute** (ex. 1G vs. 10G network card). Others will remain subjective, but a high water mark can help **estimate** (ex. we can process 10k concurrent users).

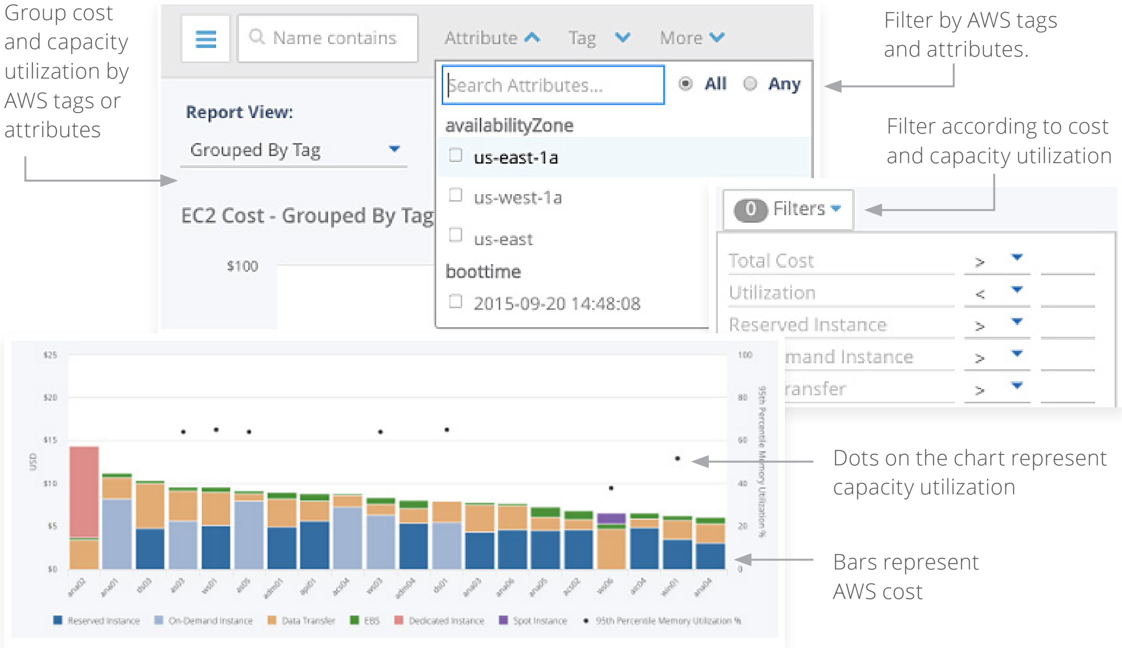


Use only one agent to collect both performance and capacity time-series data.

You would miss a short-lived spike in IOPS if you take the average across a 1 or 5 minute interval. For accurate capacity planning, you must roll-up data while preserving: standard deviation, min, max, percentiles, average and median.



Cost Analysis



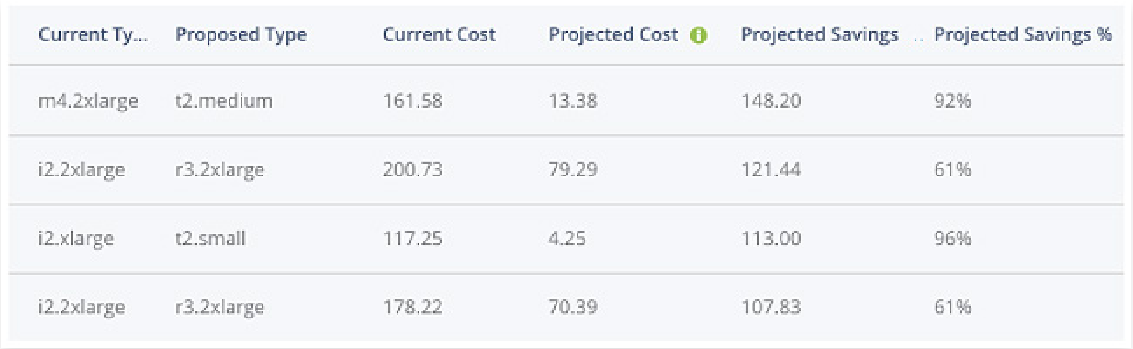
Group cost and capacity utilization by AWS tags or attributes

Filter by AWS tags and attributes.

Filter according to cost and capacity utilization

Dots on the chart represent capacity utilization

Bars represent AWS cost



Current Ty...	Proposed Type	Current Cost	Projected Cost	Projected Savings	Projected Savings %
m4.2xlarge	t2.medium	161.58	13.38	148.20	92%
i2.2xlarge	r3.2xlarge	200.73	79.29	121.44	61%
i2.xlarge	t2.small	117.25	4.25	113.00	96%
i2.2xlarge	r3.2xlarge	178.22	70.39	107.83	61%

Cross-Analyze

Cross-analyze capacity utilization with AWS cost data, and use “business intelligence” oriented concepts to filter and group by meta-data. It helps you attribute cost to application’s micro-services, plan your AWS reservation by instance type, or be notified of sudden changes in utilization or cost.

RELY ON EMAILED REPORTS TO ”PUSH” VS. “PULL” SAVINGS IDEAS

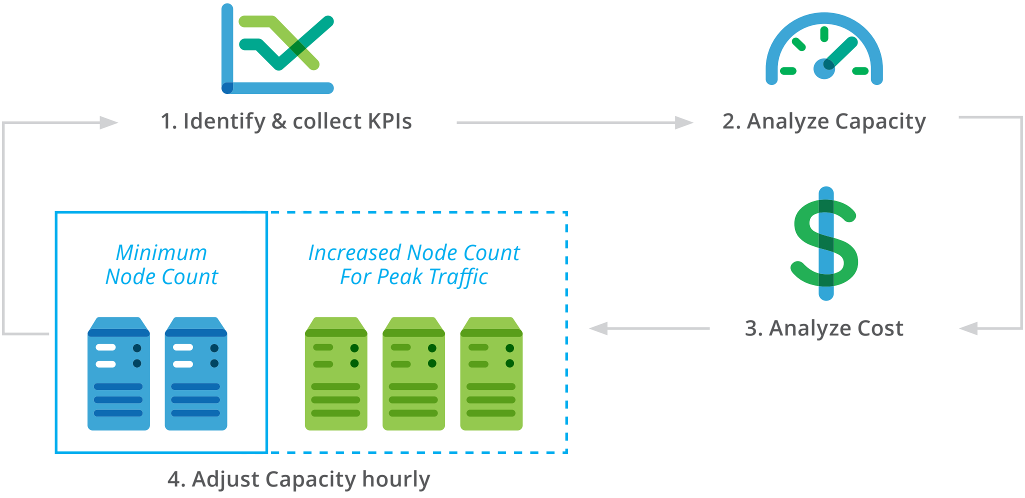
Automate

You can automate the process of searching through thousands of EC2 configurations to match the optimal EC2 configuration according to each individual instance’s workload patterns, including memory utilization and I/O.

Dynamic Resource Allocation

Whether you are using Auto-Scaling Groups (ASG), Elastic Container Service (ECS), or Kubernetes, it helps to:

1. Identify Key Performance Indicators for scaling (ex. queue depth, average cluster IOPS, number of user sessions).
2. Use a tool that allows **what-if analysis** simulation based on your historic data.
3. Use the above analysis to automatically update your ASG rule-set, Cloud Formation template, or Terraform module.



1. Identify & collect KPIs

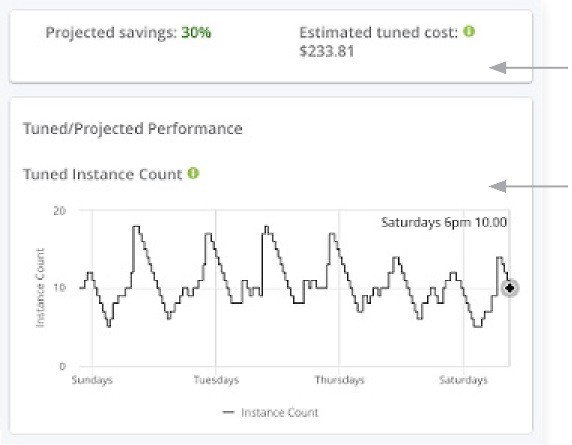
2. Analyze Capacity

3. Analyze Cost

Minimum Node Count

Increased Node Count For Peak Traffic

4. Adjust Capacity hourly

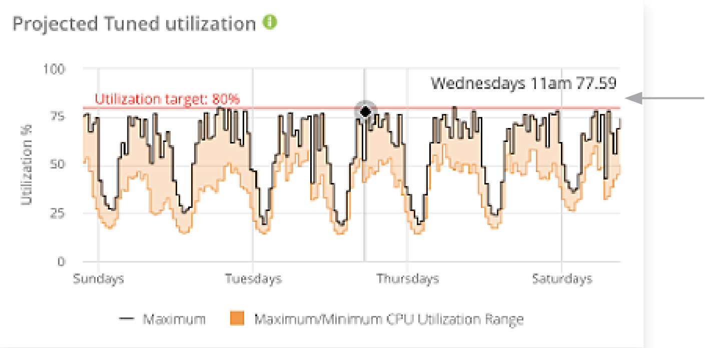


Projected savings: 30%

Estimated tuned cost: $233.81

Estimated cost savings compared to current configuration

Recommended hourly node count based on simulation of historic data



Simulated utilization levels across the cluster

A **what-if analysis simulator** for auto-scaling leverages your historic data to identify workload patterns and recommend optimal hourly node count according to your risk tolerance.

Conclusion

Use Monitoring as a Service to stay focused on managing your main application stack instead of your monitoring stack...and rely on help from technical support.

Use machine learning to automate tedious manual tasks. Not only does it save time, but you will appreciate that certain types of analysis are not humanly possible.

Competition is pushing vendors to simplify and automate. Machine learning, pre-configured dashboards, and alerts make it easy to start in minutes.

METRICLY®

Start a free trial at **www.metricly.com**