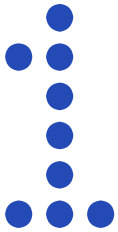
Google Cloud computing at scale.

How Punch scaled test simulations to hundreds of machines.

numin

punch case study

Contents



About Numin

NUMIN IS A LEADING QUANT FUND DEDICATED TO ENGINEERING AND TRADING THE LATEST STATISTICAL AND ARTIFICIAL INTELLIGENCE DRIVEN STRATEGIES IN PUBLIC FINANCIAL MARKETS.

FINANCE MEETS CLOUD

Numin's sophisticated algorithmic backtesting systems required a level of scale and distributed cloud computing that wasn't achievable through a manual testing process.

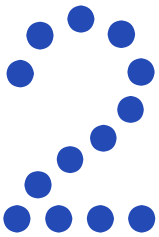
Punch worked as Numin's platform engineering solutions provider to help Numin dramatically expand its cloud infrastructure on demand. We focused on performance, cost, and ease of use in architecting a robust Google Cloud Platform solution.

The result is a fault tolerant backtesting system that is incredibly low cost, highly sophisticated, and easy to use.

PUNCH SERVICES PROVIDED

Punch provided expertise in engineering and platform engineering to help Numin meet deadlines and goals for rapid development.

Google Cloud Platform engineering, Platform engineering, Developer Operations, Site Reliability, QA, Project Management



A solution in need of scale

NUMIN WAS FACING A PROBLEM. THEIR DATA SCIENTISTS AND ENGINEERS HAD DEVELOPED A SOPHISTICATED BACKTESTING AND ARTIFICIAL INTELLIGENCE TRAINING SYSTEM: THE PROBLEM WAS, HOW TO SCALE?

The number of trials required to train a population of agents and achieve model convergence was well beyond what any local hardware setup could achieve. And the data would be silo'd.

1. **Data Silos.** Each test was silo'd to each engineers machine. While the backtesting code was in the cloud, the implementation and results of the tests driven by that code were not.

2. **Insufficient hardware.** The scale of machinery required to run hundreds of parallel trials was too much for the team's computers to achieve on any reasonable time horizon. The team looked into building their own local data center, but that carried with it a whole new set of challenges, such as further scaling.

3. **Human error.** Translation of each test trial to the master record was tedious and laborious. Mistranslation was a common problem that could result in faulty decisions by management.

800

NUMBER OF VIRTUALIZED CPUS

The system needed to accommodate hundreds of CPUs to perform all the mathematics required for accurate backtests.

The fix for Numin

PUNCH WORKED WITH NUMIN TO SOLVE BACKTESTING ISSUES THROUGH A DYNAMIC RESOURCE ALLOCATION MODEL.

Punch worked closely with Numin and their team to break-apart the problem into several steps, resulting in a streamlined, first-principles approach to the problem.

PROBLEM PROCESS

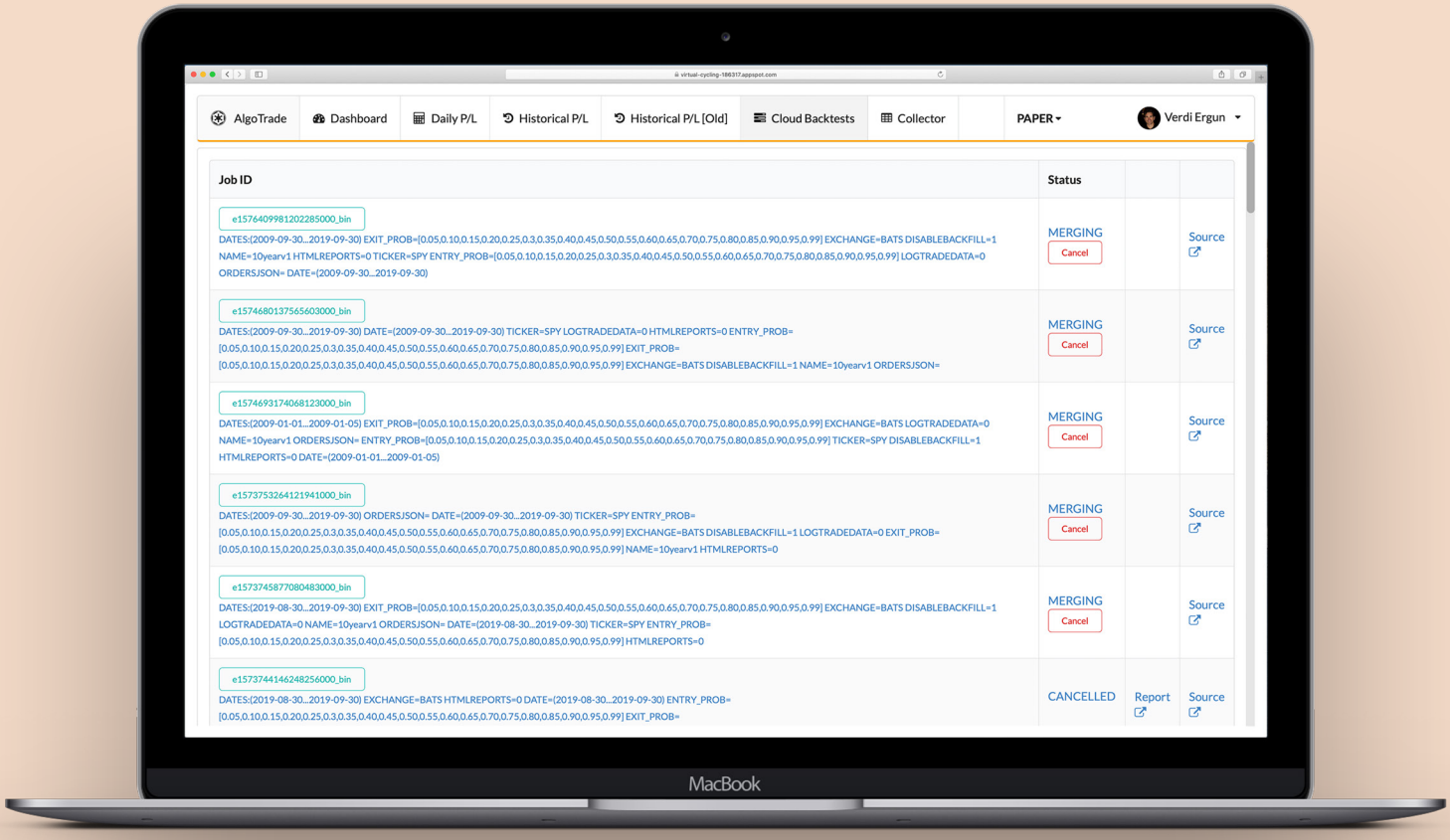
Demand	Engineers from Numin would queue a backtest job to the cloud.
Supply	A series of scripts would begin allocating resources to the job.
Scaling	The CPU utilization of each resource was monitored, as the CPU utilization grew beyond certain thresholds, more virtualized machines were dynamically spawned. Each virtualized machine would pick up jobs from the stack when its job was complete. Machines were fault-tolerant.
Descaling	As the tests finished, machines would be automatically despawned to minimize cost.
Collection	Results from each machine would be reported back in real-time to a centralized machine that would combine the results into a single results graph.

1

DEMAND

SPAWN A CLOUD JOB FROM ANYWHERE IN THE WORLD.

Numin engineers could spawn a cloud job from their local terminal using Google Auth and Google Cloud SDK. A single cloud machine would begin the test and self-monitor for CPU utilization. A separate reporting dashboard would show the progress of each individualized backtest over time.

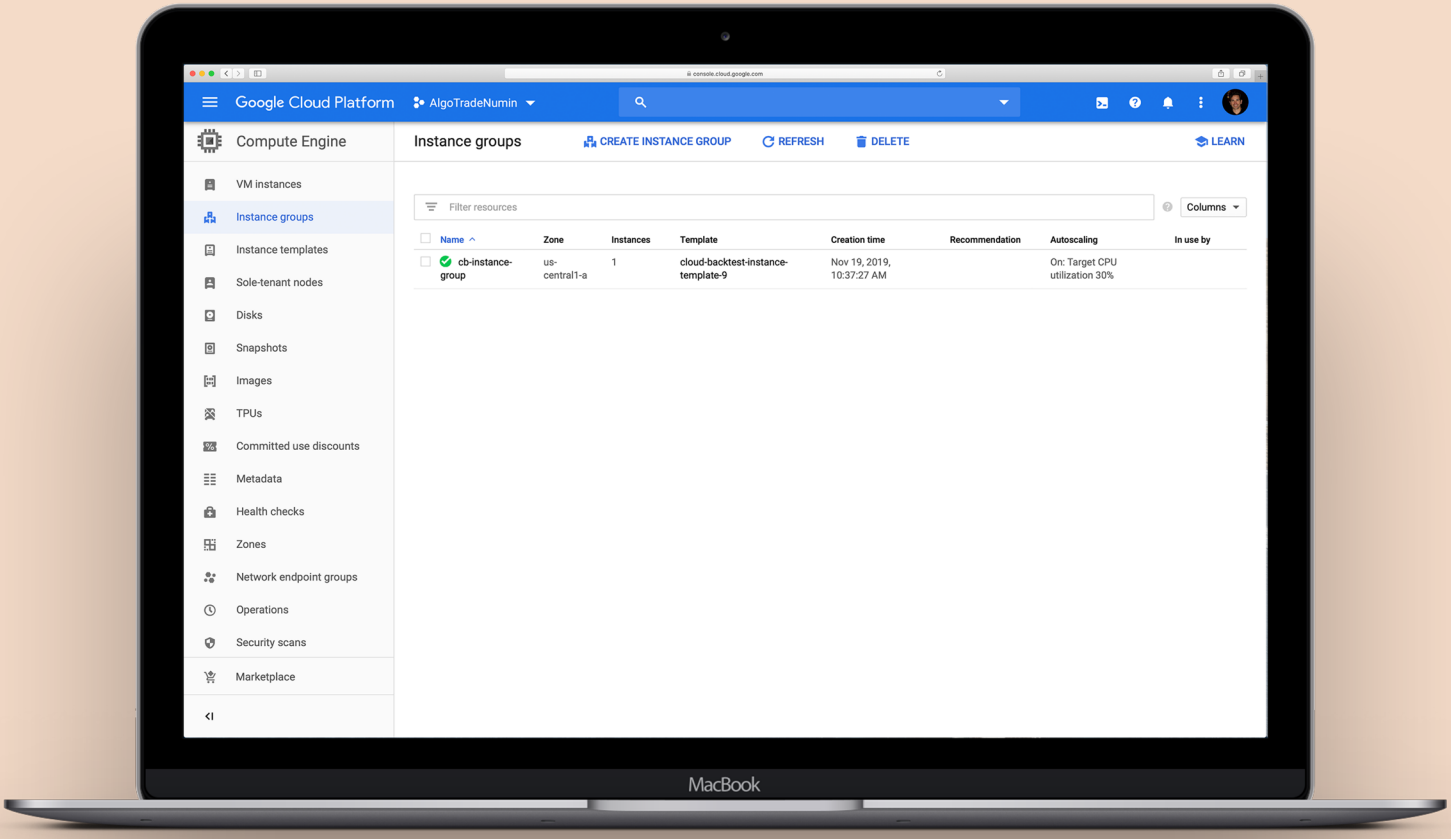


2

SUPPLY

HAVE THE JOB EXECUTED VIA SCRIPTS TO GOOGLE CLOUD.

The backtesting job would hit Google's Cloud servers. A job queue would allocate resources to the machine which would begin the job.

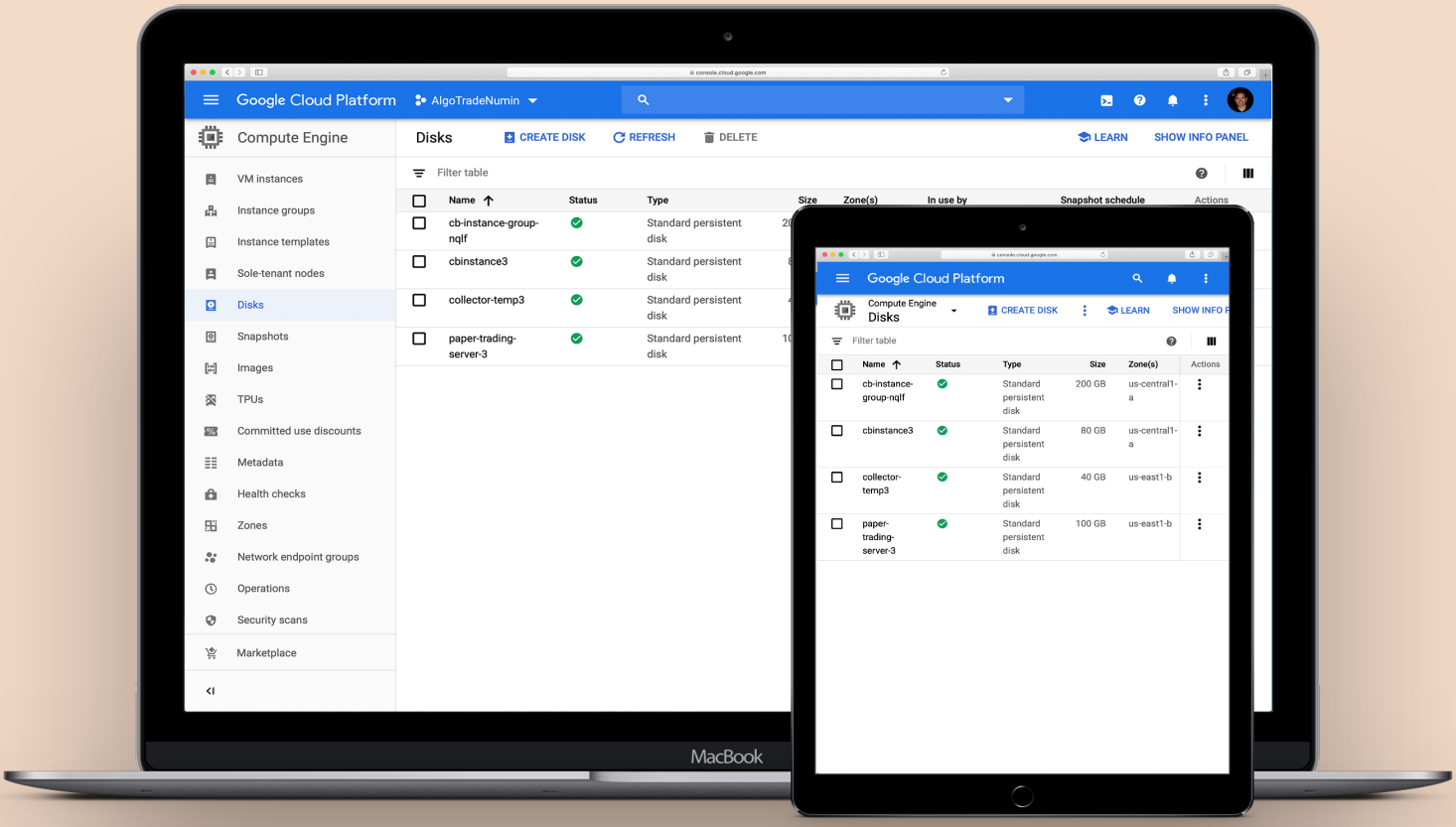


3

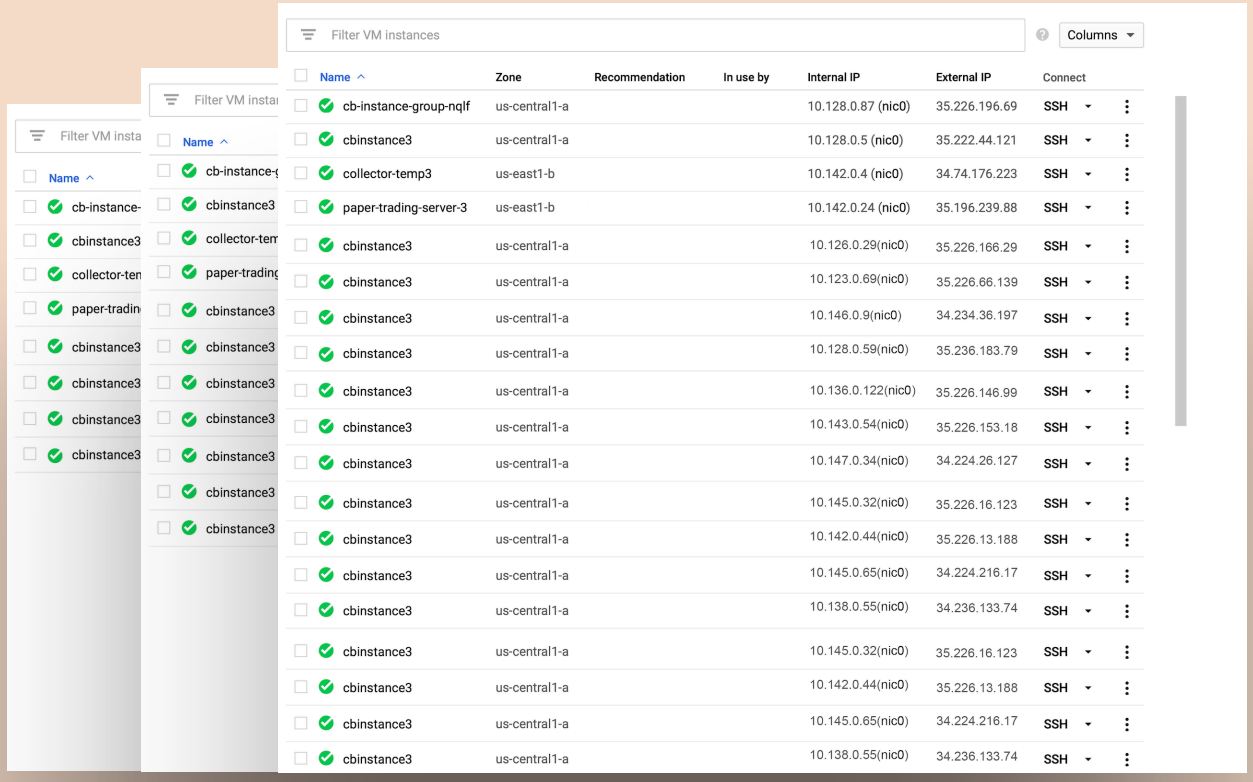
SCALING

FROM 1 TO 800.

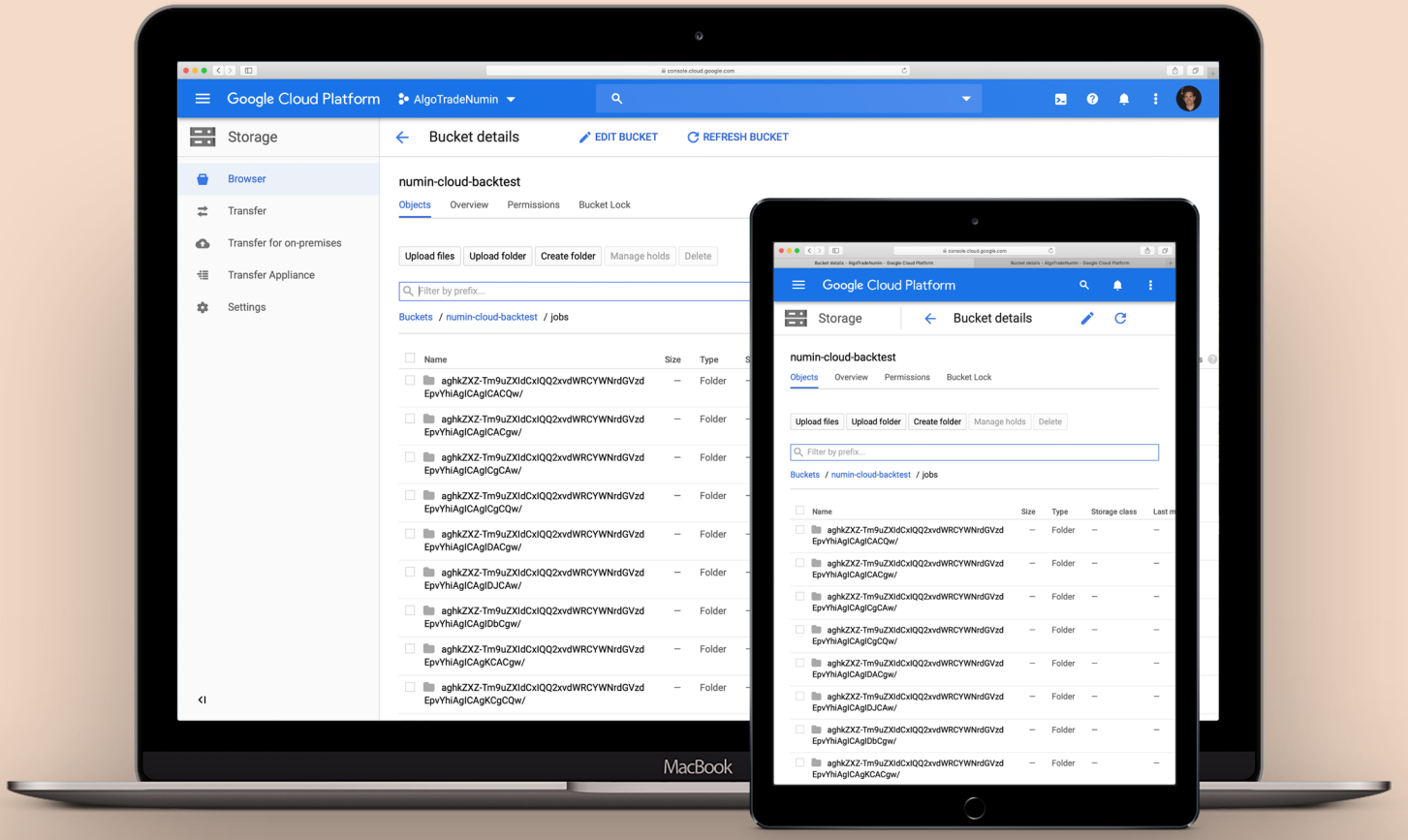
Each instance, once its CPU utilization reached beyond a certain threshold, would self-spawn a copy spot instance which would then start pulling job queues off a job stack. As spot instances can be dequeued by Google at any moment, if a machine was dequeued a new machine would be spawned in its place. In cases where a new spot instance was not available, the stack would wait and attempt to respawn at key time intervals.



Hundreds of spawned machines would coordinate together to complete the task.



Results would be collated and pushed to a unique Cloud Storage bucket specific to that backtest.



> *"The compartmentalization of the quants on both teams have **owned their domain**, creating a **research factory** of **continuous improvement** that would have been difficult to achieve otherwise."*

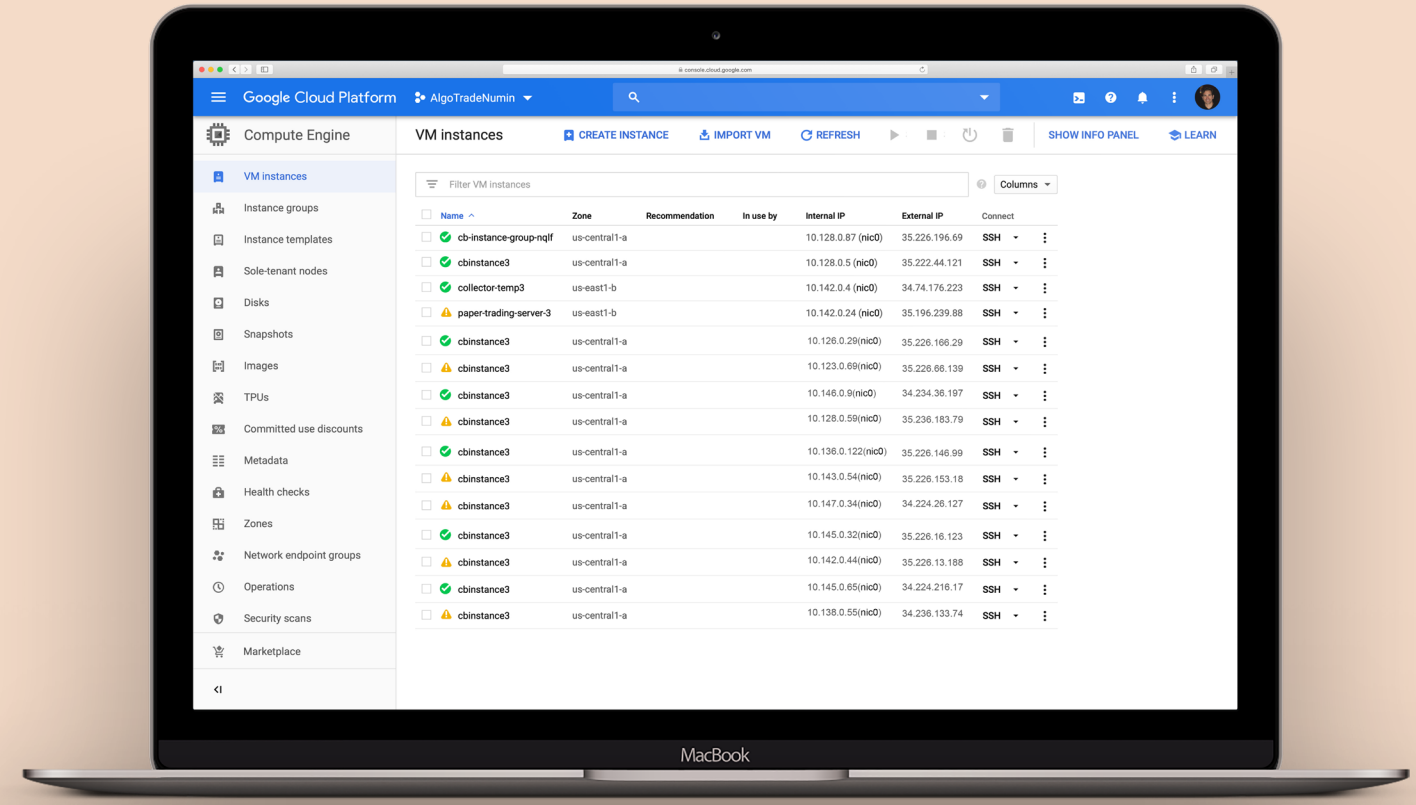
ANUP

Director of Engineering
Numin

4

DESCALING

As the job neared completion virtualized machines would dequeue at the rate of spawning to keep costs to an absolute minimum.

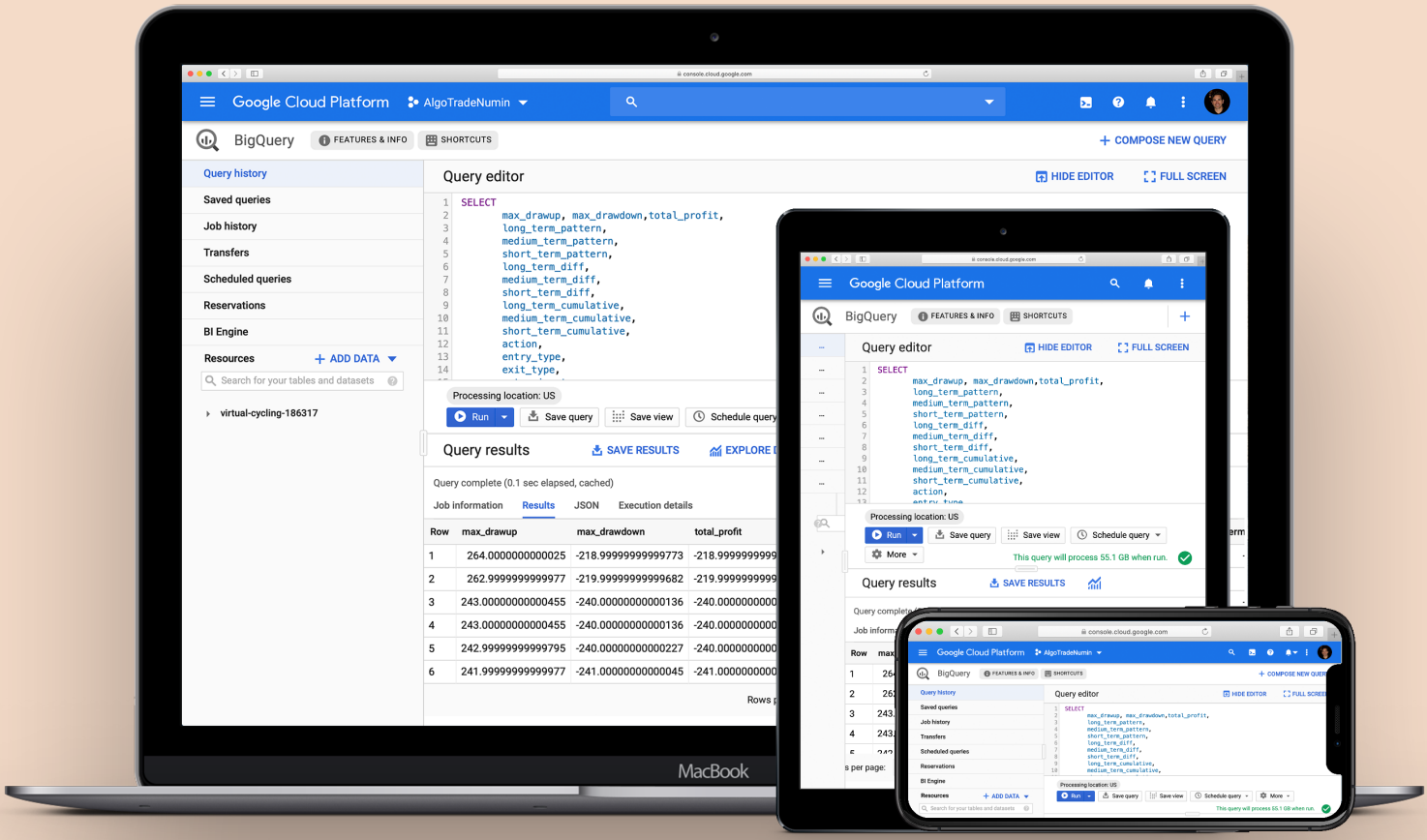


5

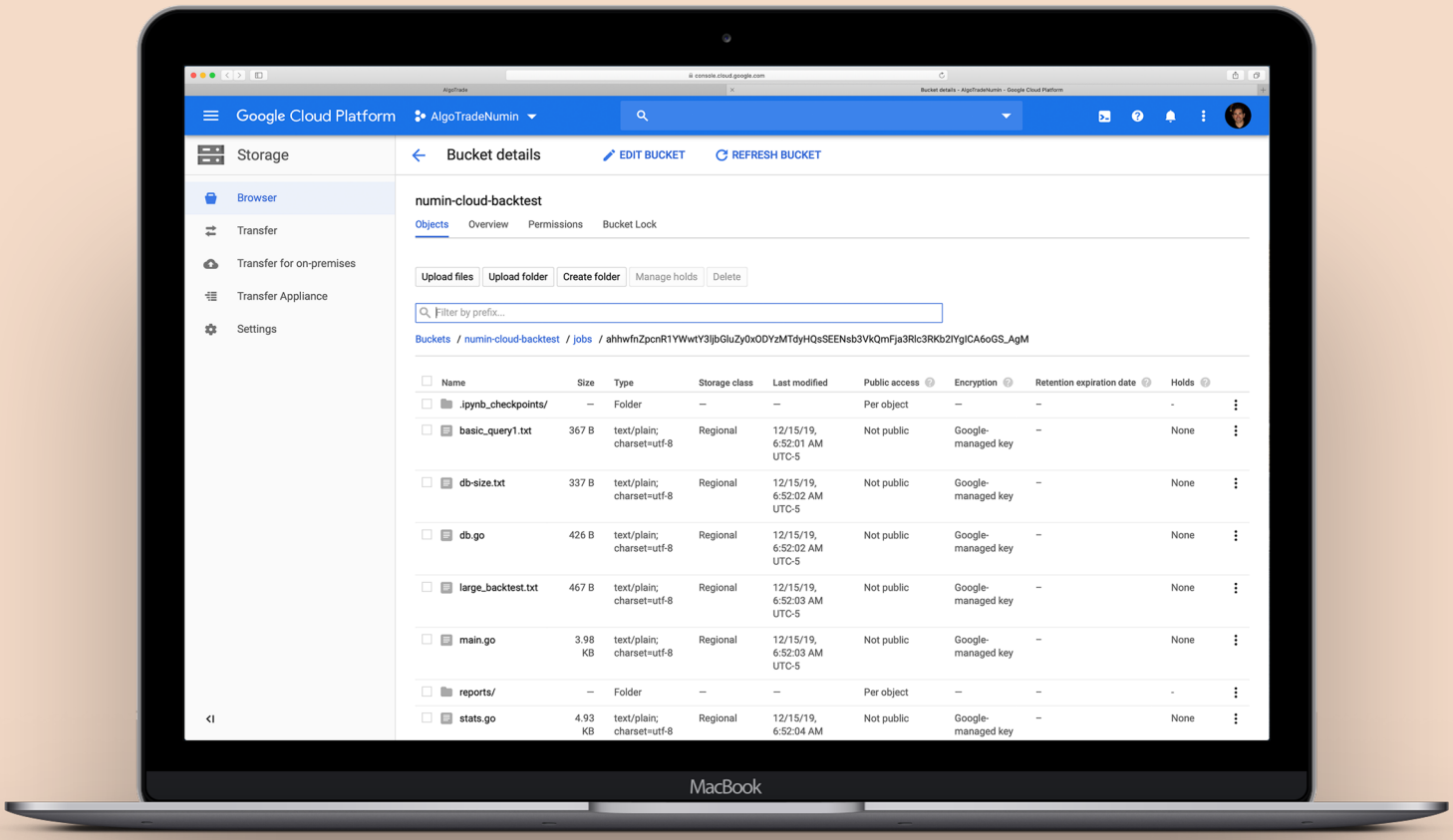
COLLECTION

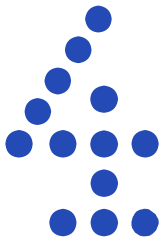
RETRIEVING THE DATA IN THE RIGHT FORMAT WAS AS IMPORTANT AS CREATING IT.

Total backtest results would be automatically broadcast to Big Query for storage, assigned a trial ID with metadata about the trial itself.



Medata included how many virtualized CPUs were used, the test parameters, how long the test took, number of machines that were terminated during the test (through error or by Google), and how long dequeuing took as a percentage of queueing. Test metadata was replicated back into Google Cloud Storage unique to the backtest job.





Results

PUNCH PROVIDED A LARGE-SCALE ENTERPRISE BACKTESTING AND CLOUD PROVISIONING AND DEPROVISIONING SOLUTION TO HELP POWER CUTTING EDGE FINANCIAL TRADING ALGORITHMS.

APP STATISTICS

	Items
Spawned Virtualized CPUs	**800**
Length of Project	**4 months**
Tech stack	Google Cloud Platform, Big Query, Google Cloud SK
Teams involved	San Francisco & Lahore

Legal

Prepared on February 14, 2020. This document is confidential. It contains material intended solely for the original recipient. Ideas presented here are the property of Punch and are copyrighted.

punch