INTUIT turbotax creditkarma quickbooks mailchimp

Kubernetes on a Budget: How to Get Pay-per-Use Right

November 8, 2023

Karim Lakhani, Senior Staff Software Engineer, Intuit
Vasuki Prasad, Staff Software Engineer, Intuit

Meet Your Presenters

Karim Lakhani

Vasuki Prasad

Agenda

Intuit's API Gateway at a Glance

Navigating Cost Overruns: Our Journey

Challenges, Learnings, and Solutions

Takeaways

AI-driven expert platform

INTUIT

KubeCon CloudNativeCon North America 2023

Modern Dev Environment

Enables Intuit engineers to develop code in a fast, secure, and compliant fashion by infusing Generative AI in every aspect of dev journey.

9x
Increase in dev productivity since 2019

~1M
Active CPU cores

40M+
AIOps inference/day

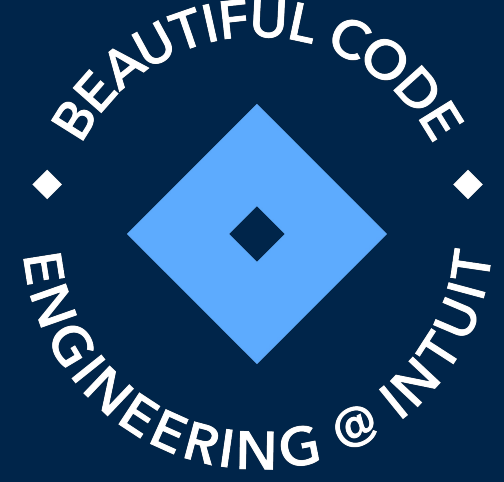
2500+
services

315
Kubernetes clusters

28k+
namespaces

1k+
teams

Intuit's API Gateway at a Glance



The Role of Intuit's API Gateway

- A lot of Service to Service communication, too

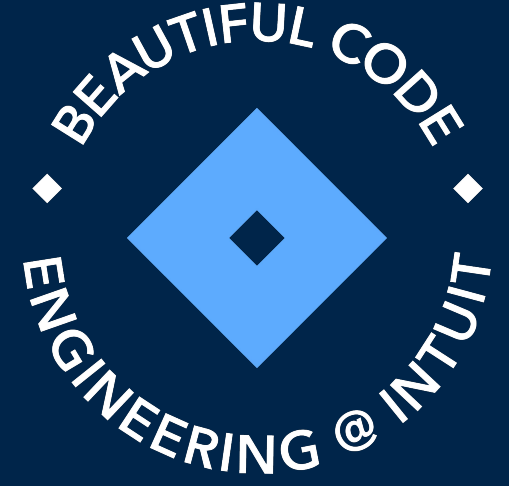
Intuit's API Gateway Provides

- Routing (static and dynamic)
- Security
- Client application and user authentication
- Client application and user authorization
- Metrics (Golden Signals) & Dashboards
- Access Logging
- Rate limiting
- Traffic dialing

Benefits and Stats of Intuit API Gateway

Highly scalable	30+ Billion requests per day at peak 1.16M requests per second at peak 30+ clusters 250+ namespaces
Highly available	99.99% Availability
Highly reliable	Source of Golden Signals metrics
Low Latency	p99 Overhead < 30ms
Self-Service management	Internal DevPortal

Why In-House?



History: A Decade of Innovation and Evolution

Our in-house development journey started around 11 years ago in the Intuit data center.

This marked the beginning of our transition towards micro-services and the decomposition of monolithic structures.

Intuitism

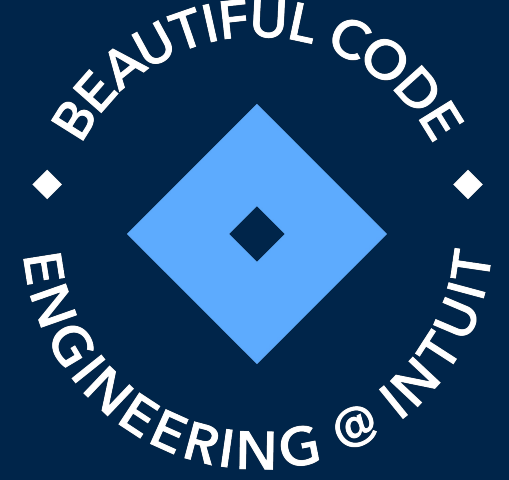
Over the last 11 years, we've made numerous customizations that are unique to Intuit.

- Ticket exchange and verification with Intuit Identity providers
- Features to support business use cases

Our Approach

- Similar to Netflix's Zuul 2 and AWS API Gateway.
- Non-blocking and asynchronous with Jetty
- Partitioning first class feature, to isolate workloads for different products like TurboTax, Quickbooks, Mailchimp, CreditKarma, etc
- More relaxed quotas compared to AWS API Gateway.
- Longer timeouts and larger request and response payloads.

Why Migrate Intuit API Gateway to Cloud Native Technologies?



Istio

- Integration with Intuit's Service Mesh
- Security
- Reliability
- Observability
- Reduce data transfer costs
- Enable Network Abstraction

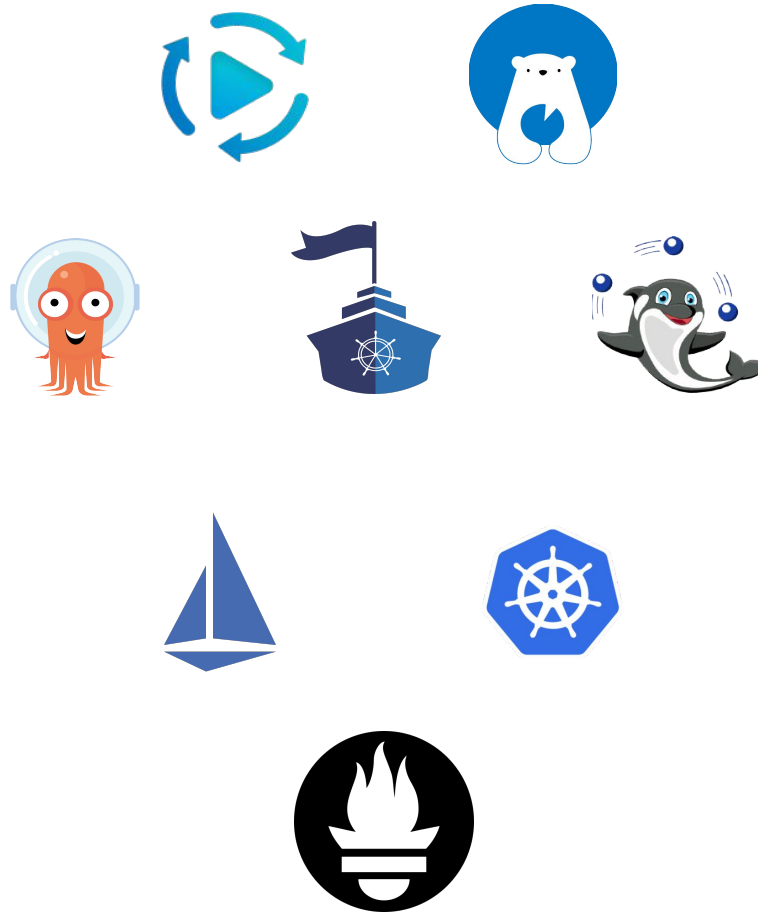
Observability

- Improve system and application metrics
- Improve metrics accuracy
- Improve cost analytics
- Phase out legacy observability technologies

Intuit's Modern SaaS Platform

Take advantage of and contribute back to Intuit's and Cloud native tools and capabilities

- Argo CD
- Argo Rollouts
- Argo Workflows
- Intuit Service Mesh (Istio / Admiral)
- Prometheus
- Phase out legacy tools and deployment scripts

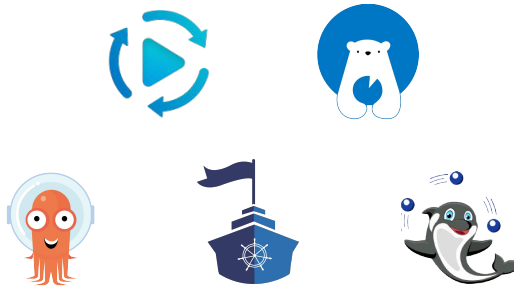


We believe in open source and open collaboration

INTUIT

KubeCon CloudNativeCon North America 2023

Recipient of the **End User Award** in 2019 & 2022

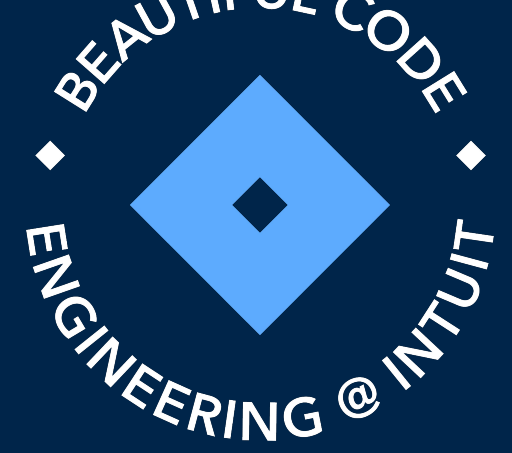


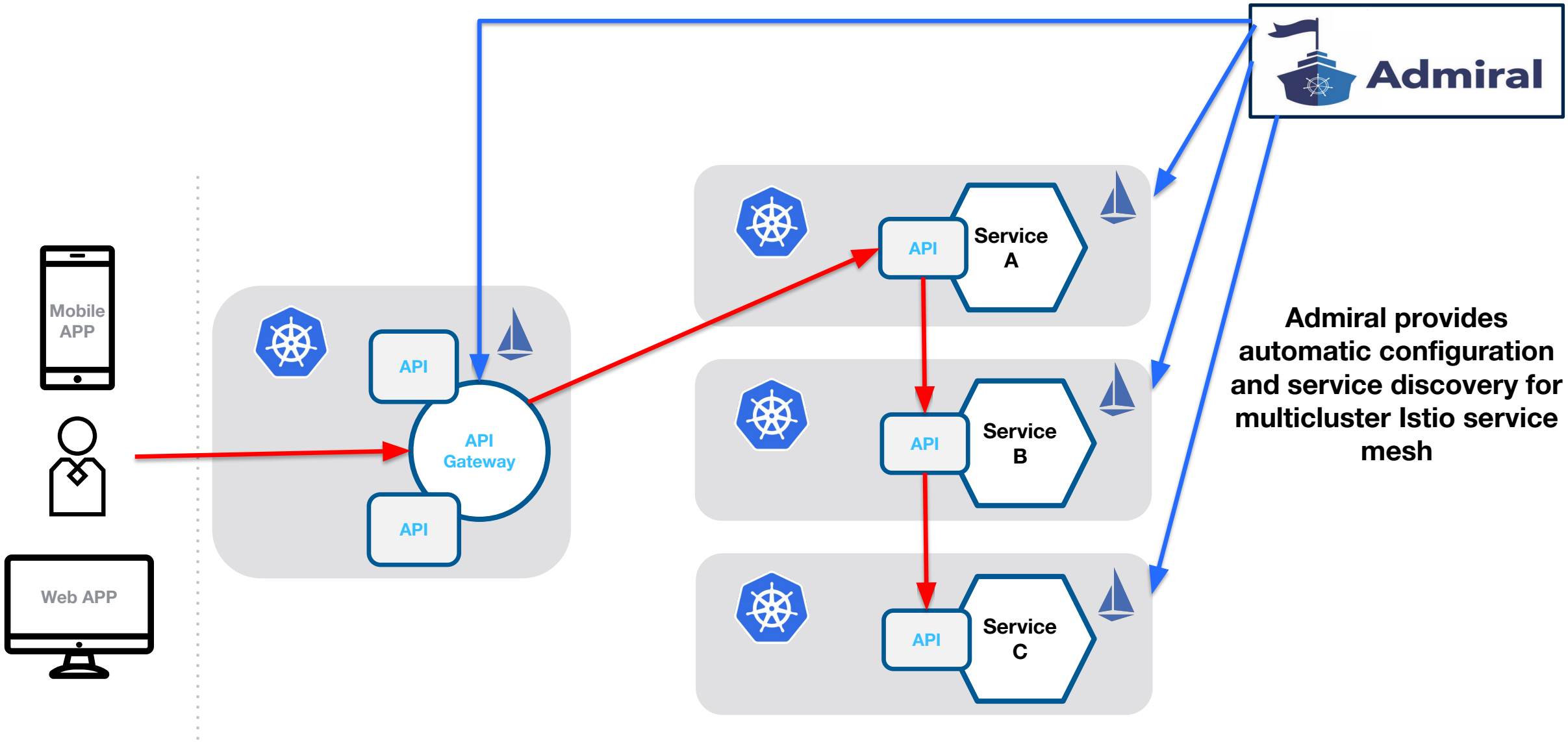
Created, open-sourced, used, and maintained **by Intuit**

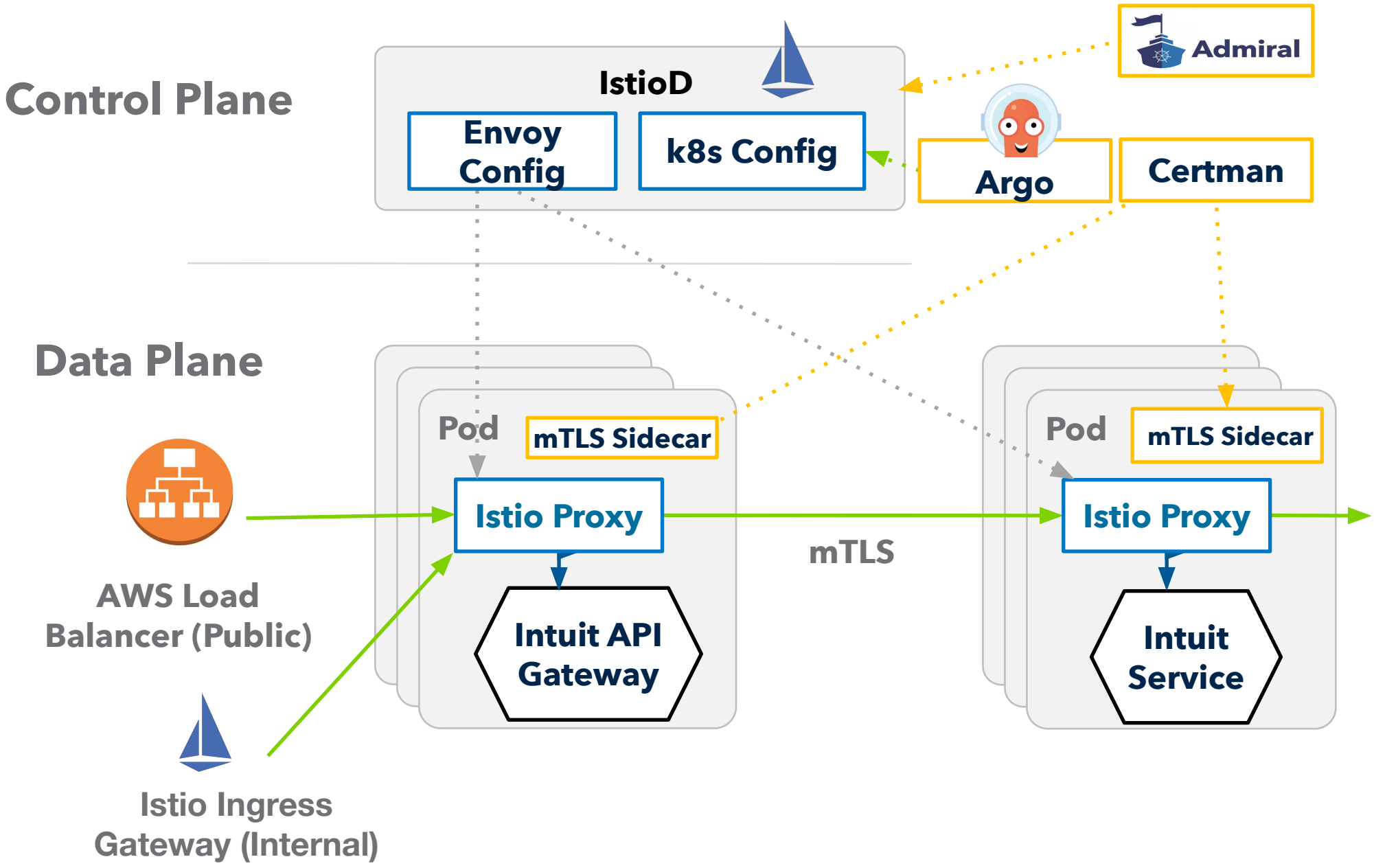
End user of Cloud Native and mobile open source tech

bit.ly/intuit-oss

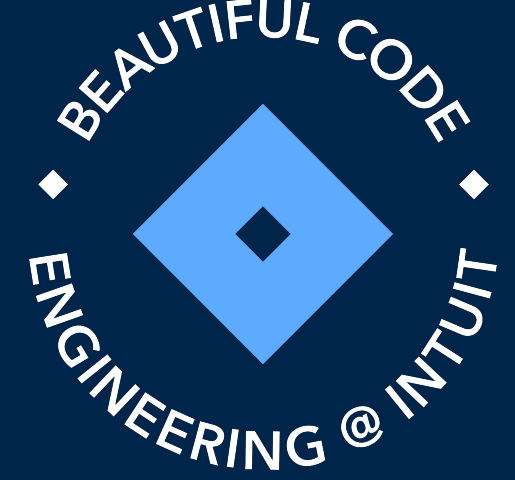
Updated Intuit API Gateway Architecture




Admiral
Mobile APP
Web APP
API
API Gateway
API
API
Service A
API
Service B
API
Service C
Admiral provides automatic configuration and service discovery for multicluster Istio service mesh



Our Cloud Native Cost Optimization Journey



Navigating Cost Overruns: Our Journey

Discovery

We realized that our projected costs were going to be 2x over what we had initially expected. This was a critical moment that required immediate action.

Analysis

We embarked on a comprehensive analysis to understand the root causes of this cost overrun.

This involved scrutinizing every aspect of our infrastructure costs.

Results

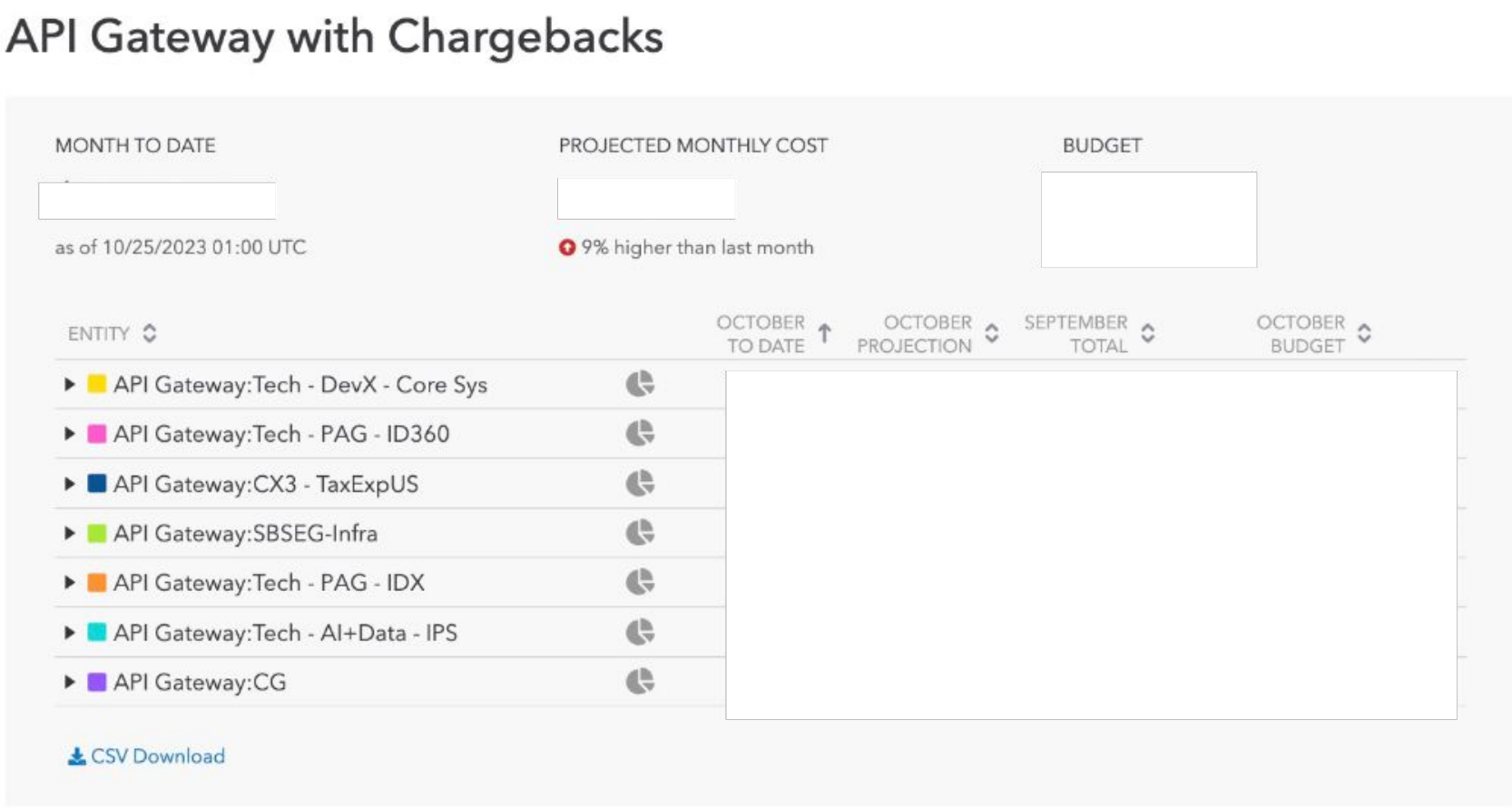
Our analysis revealed several areas where we could optimize costs.

By implementing changes we were able to reduce our expenditure by 50%.

Discovery

Watching Trends

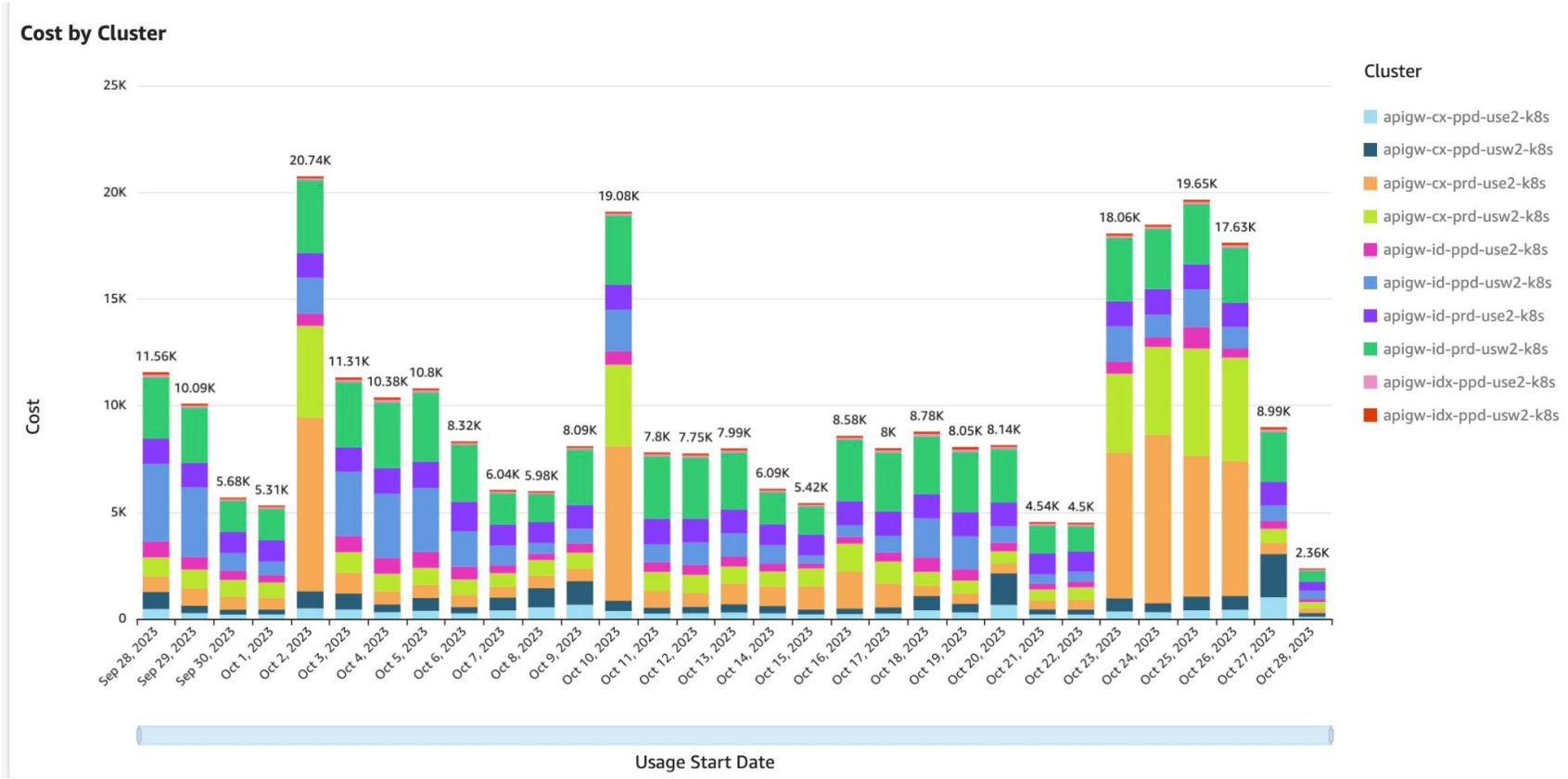
- Month over month
- Week over week
- Daily cost
- Cost by Service



Analysis

Detailed Cost Reporting

- Cluster
- Namespace
- Environment
- Business Unit
- Cloud Service

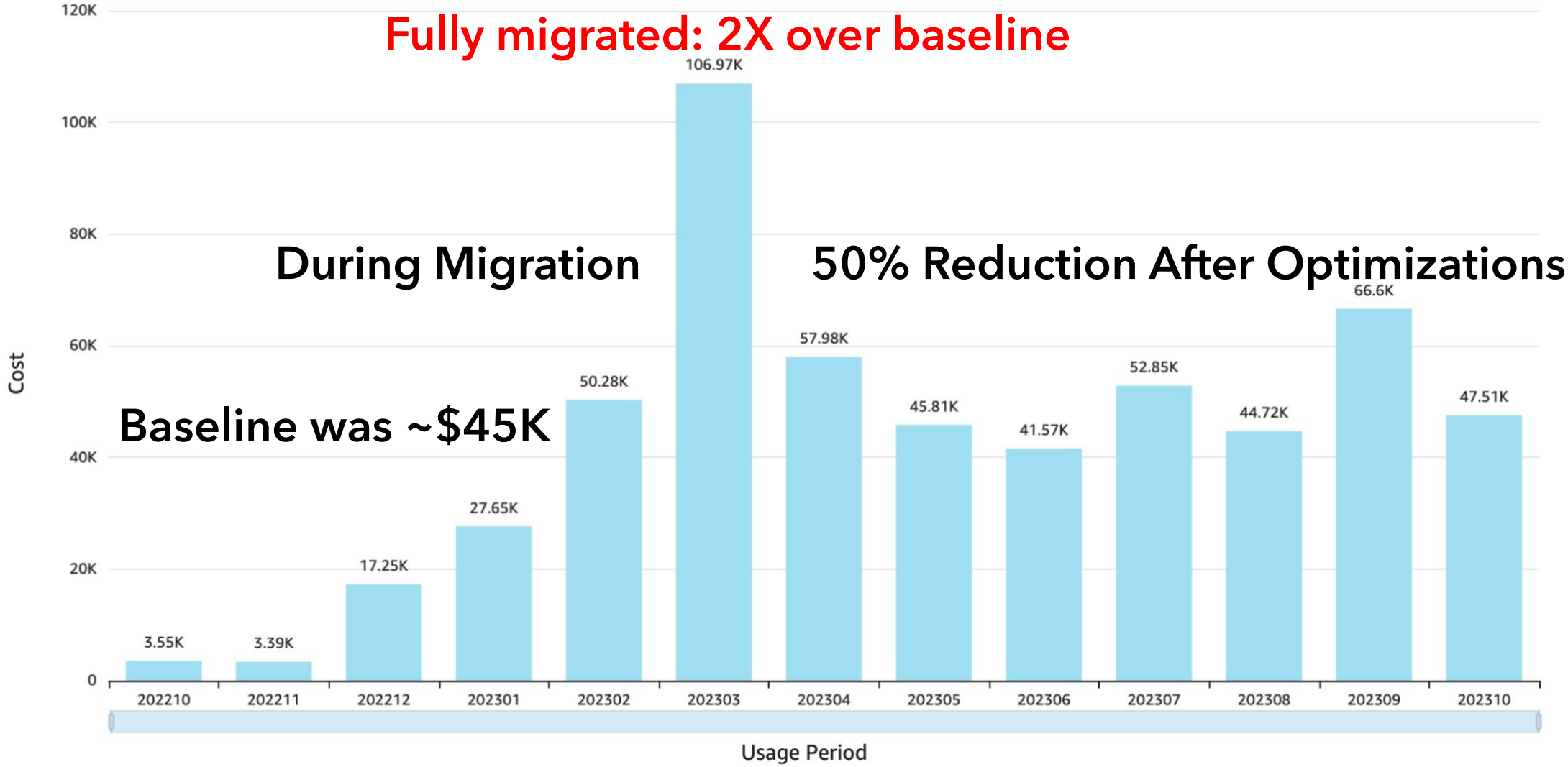


Combining Cost and Usage Metrics

1. **Cost Observability: Utilizing Cloud 360**
2. **Usage Observability: Employing Prometheus and Wavefront**

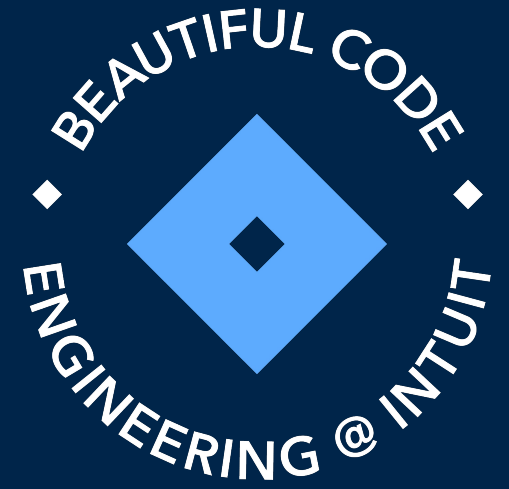
- Request Counts - What is the **cost per million requests**, on average?
- Node Counts
- Pod Counts
- Pod Density
- CPU and Memory Utilization of each Container
- Establishing Baselines

Results In Largest Preprod Account

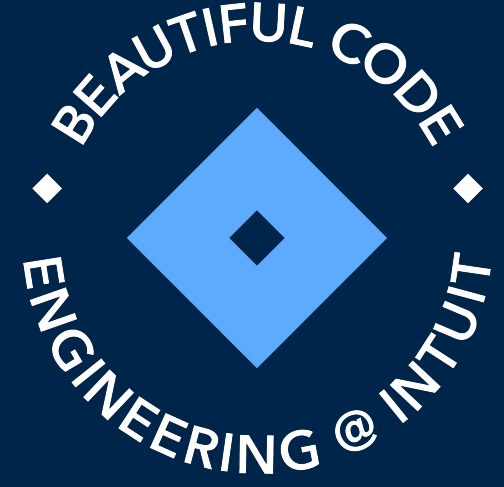


Areas Of Optimization

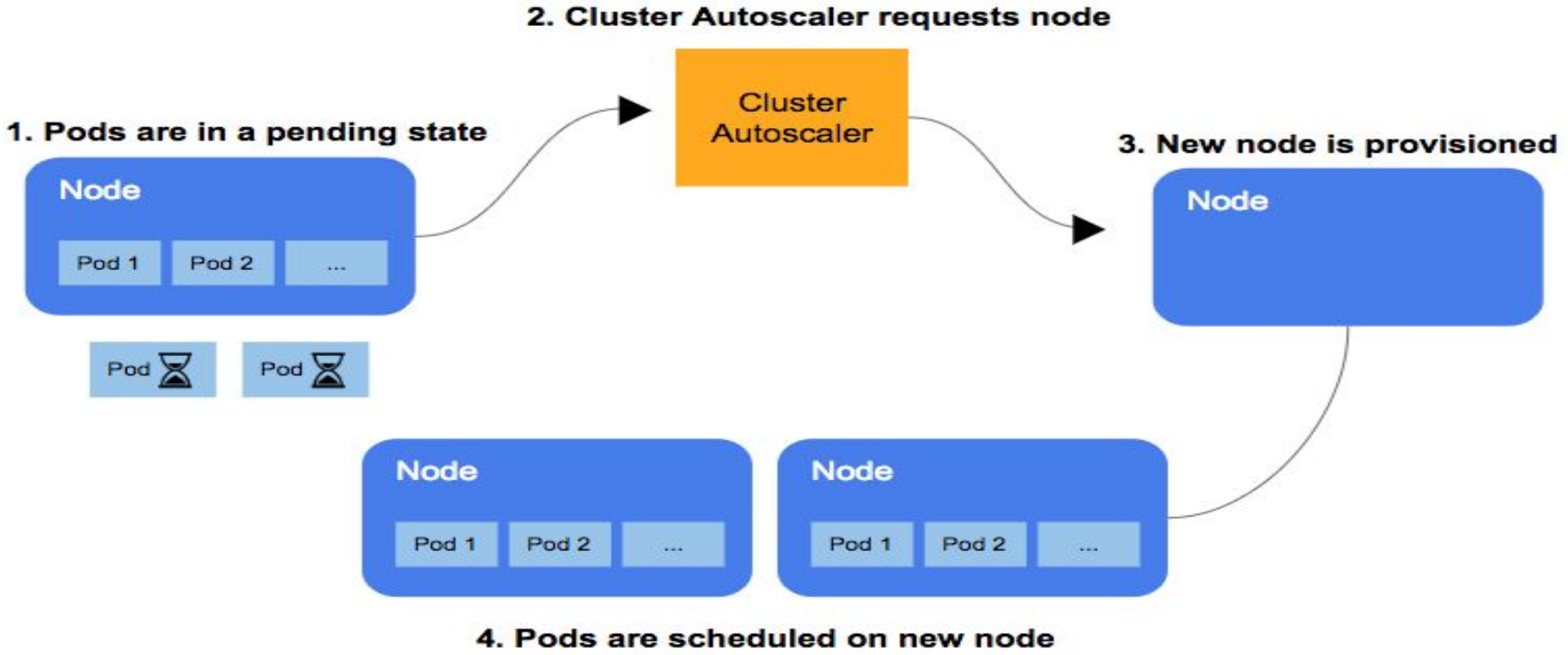
- Compute
  - Autoscaling
  - Underutilized Nodes
  - Pod Density
- Data Transfer
- Observability / Monitoring
- Developer Environments



Compute and Auto Scaling



Scaling on demand!!



Optimizing Autoscaling

Challenges

- Rapid scaling to manage surges and unexpected load ramp-ups
- Simple HPA (Horizontal Pod Autoscaler) did not perform as expected.
- Avoidance of oscillating scaling.
- Higher minimums to accommodate unexpected surge

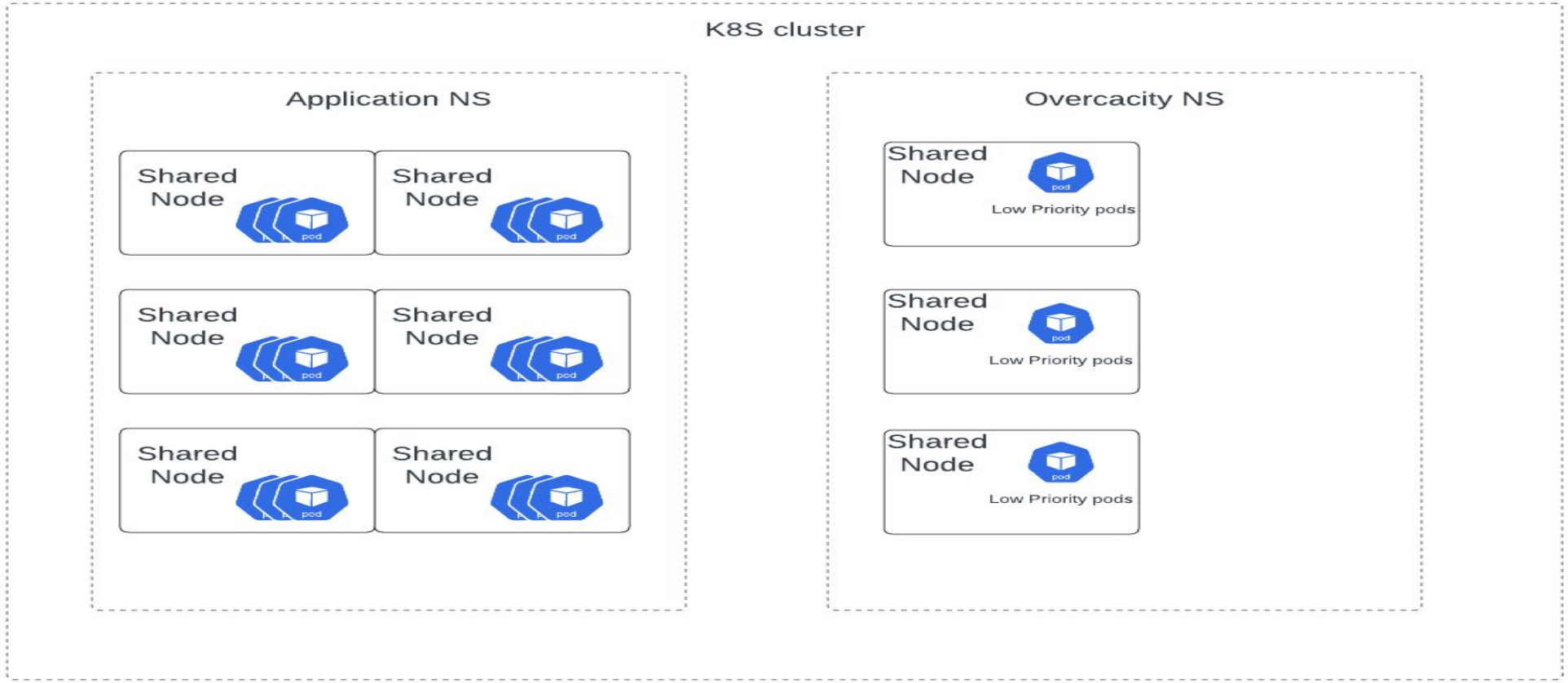
Solutions

- Know your metrics for scaling.
- Know your target CPU.
- Avoid Aggressive scaling.
- Step Scaling algorithm for gradual ramp up and ramp down
- Over capacity pattern to accommodate unexpected surge.

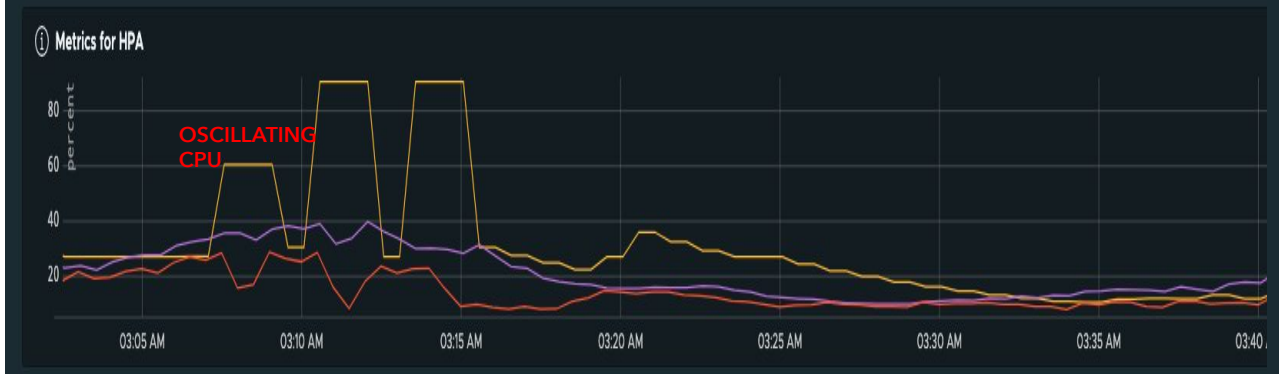
```yaml
metrics:
  - name: namespace_app_container_app_avg_cpu_utilization
    target: 40
    tags:
      app: intuit-gateway
scaleUp:
  algoType: StepScaling
  adjustmentType: PercentChangeInCapacity
  stepAdjustments:
    - metricIntervalUpperBound: 3
      scalingAdjustment: 2
    - metricIntervalLowerBound: 3
      metricIntervalUpperBound: 6
      scalingAdjustment: 6
    - metricIntervalLowerBound: 6
      metricIntervalUpperBound: 9
      scalingAdjustment: 25
    - metricIntervalLowerBound: 9
      metricIntervalUpperBound: 13
      scalingAdjustment: 50
    - metricIntervalLowerBound: 13
      metricIntervalUpperBound: 17
      scalingAdjustment: 100
    - metricIntervalLowerBound: 17
      metricIntervalUpperBound: 20
      scalingAdjustment: 150
    - metricIntervalLowerBound: 20
      metricIntervalUpperBound: 40
      scalingAdjustment: 200
    - metricIntervalUpperBound: 40
      scalingAdjustment: 400
```

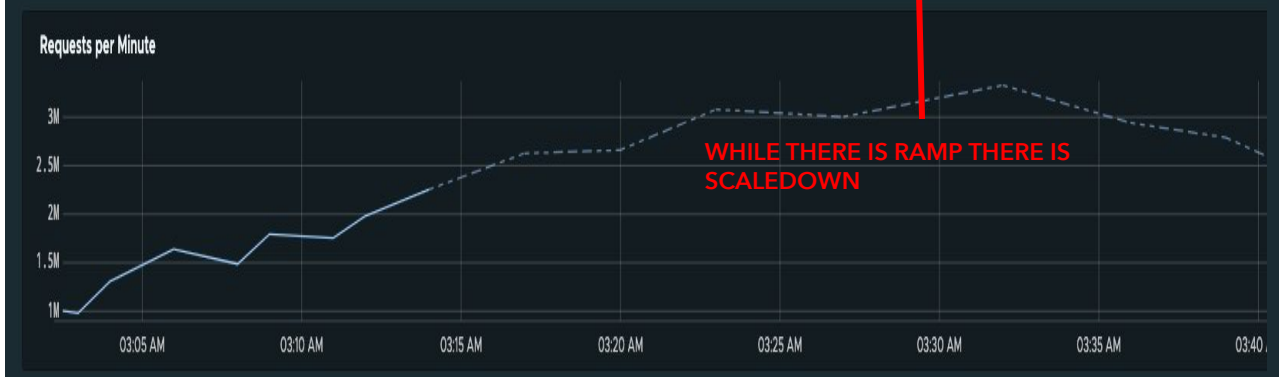
Step Scaling:
Fine-grained control of how much to scale based on how much over the target it is

OverCapacity Pattern

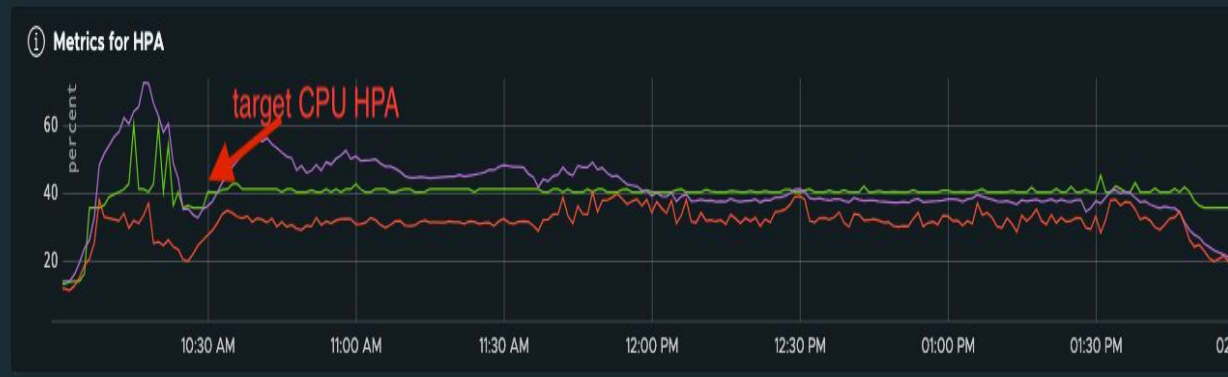


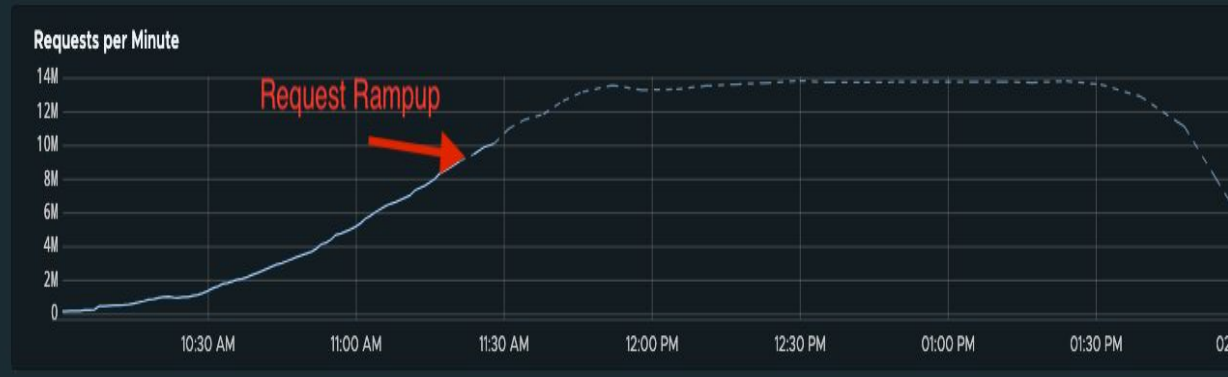
Before





After

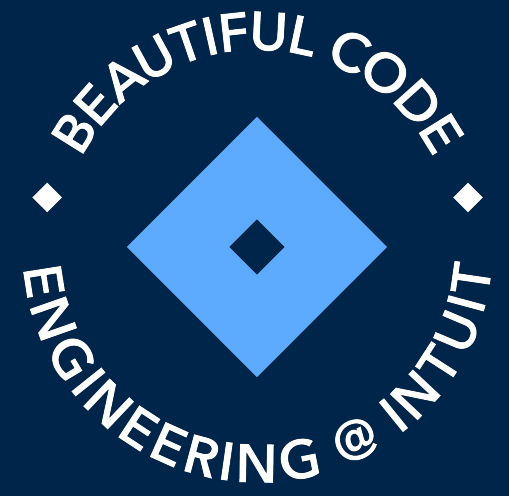




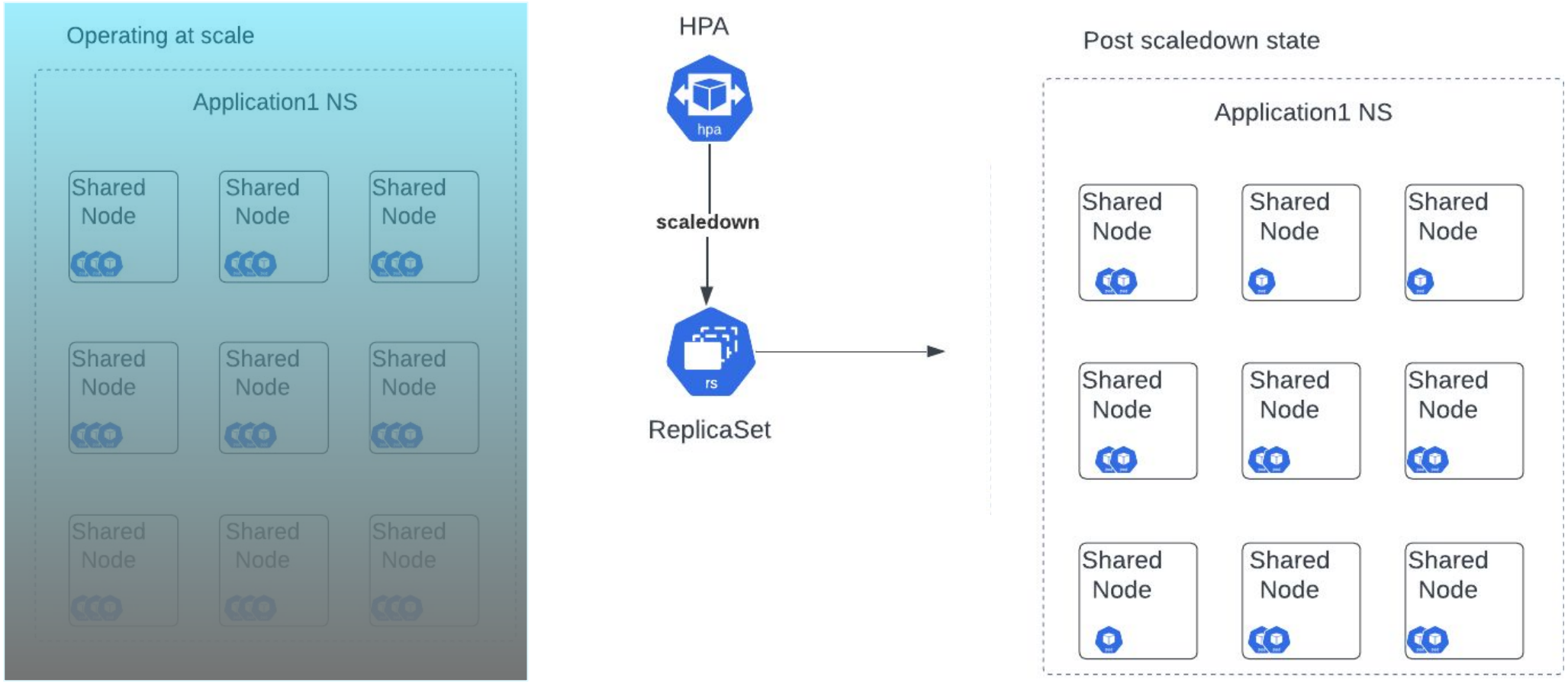
What we saved?

Upto 25% $$ Saving on the Compute.

Underutilized Nodes (Nodes vs Pods)



Cold and Idle Worker Nodes



Balancing Pods and Nodes

Challenges

- Slow scale-down of nodes leading to inefficient resource utilization and higher cost.
- Slow Cluster autoscaler to rebalance node and pods.

Solution

Don't relay on default but fine-tune cluster autoscaler configuration for balanced pods and nodes, including adjustments to scale-down parameters.

Cluster AutoScaler Down scaling strategy

Cluster Autoscaler Settings

- **scale-down-delay-after-add**
  - 5m0s
- **Scale-down-delay-after-delete**
  - 10s
- **Scale-down-utilization-threshold**
  - 0.75
- **Scale-down-unneeded-time**
  - 5m
- **Scale-down-unready-time**
  - 10m

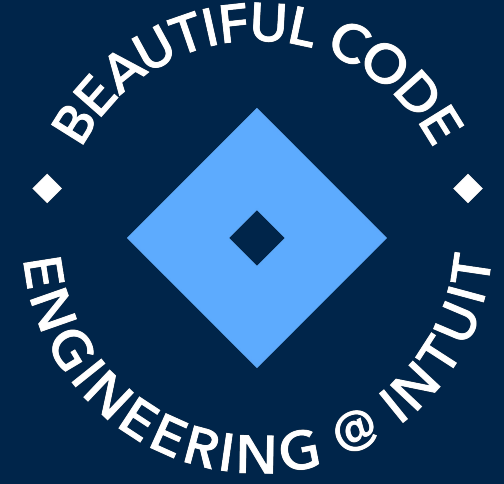
Outcomes

- Better pods to nodes ratio
- Corden and remove underutilized nodes efficiently.
- Resizing/rebalance pods for quick and efficient distributions.

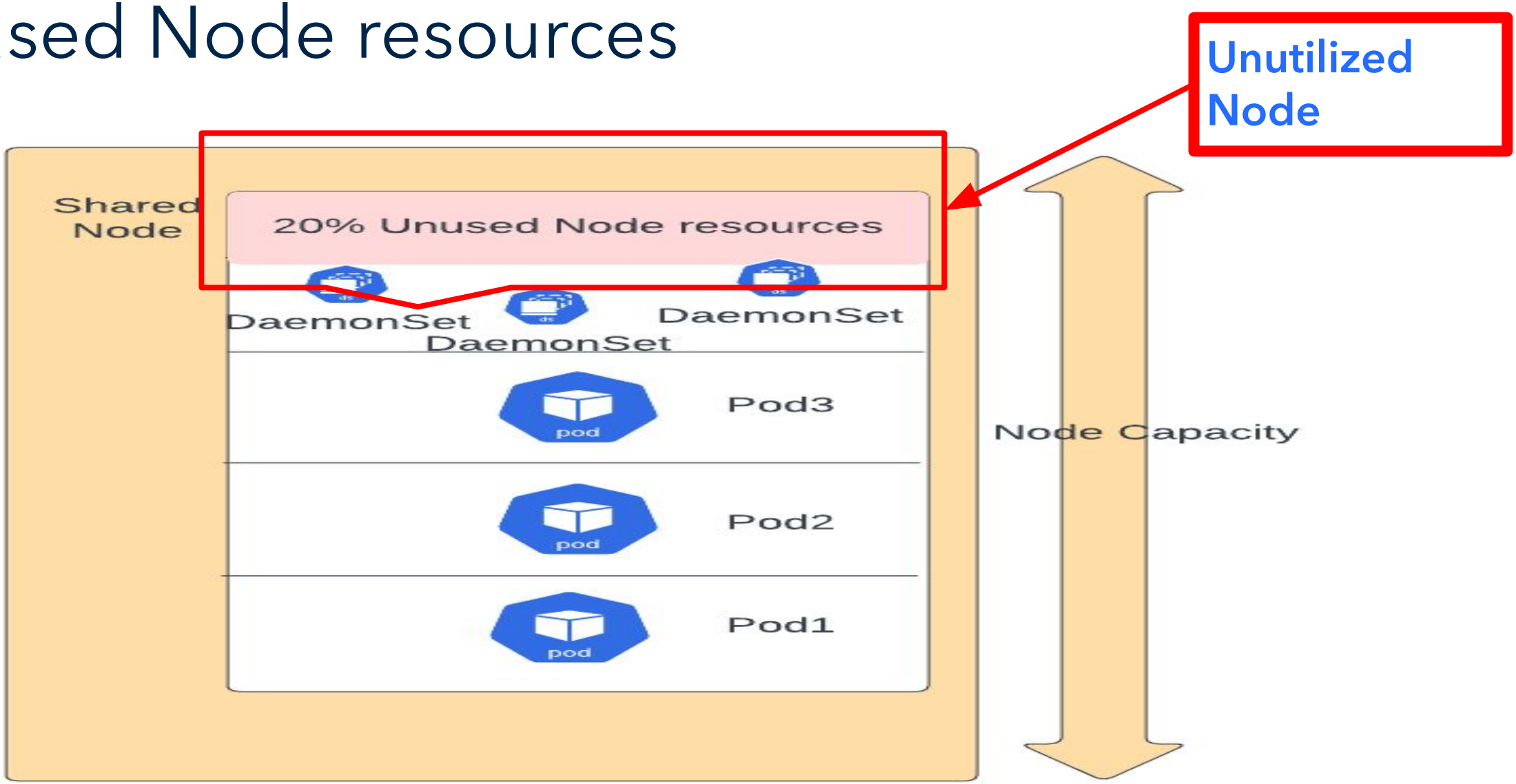
What we saved??

Upto 15% $$ Saving on the Compute.

Pod Density



Unused Node resources



Optimizing Pod Density

Challenges

- Inefficient pod bin packing, leading to underutilization of resources.
- Sidecars reserving resources without fully utilizing them.
- Over provisioned sidecar resources influenced by platform teams.

Solutions

- Fine-tune requests and limits on sidecars for optimized resource usage.
- Fine-tune app requests and limit.

Balancing Smaller vs. Bigger Instances

Challenges

- Larger instances provides better throughput but increase the blast radius, potentially affecting more applications in case of failures.
- Inefficient sidecar sizing

Solutions

- Balance availability and throughput with appropriate instance size selection.
- Optimize node and pod counts for maximum resource utilization.
- Transition to newer generation

Outcome

Boosted resource utilization and cost-efficiency through instance optimization. (c5n vs c6in), 15% price-performance improvement

Pod density calculator

Model	vCPU	Memory (GiB)	Instance Stora	EBS Bandwidth (Mbps)	AWS Cost
c6in.4xlarge	16	32	EBS-Only	Up to 20	0.9072 USD per Hour

Usable Compute/
Compute by pod

Compute per pod

	total CPU	app	istio	splunk	certagent	other / buffer (per pod)	sum per pod	pod density
request	13600	2400	800	100	10	50	3360	4.047619048
limit	13600	2400	1800	0	100	50	4350	3.126436782

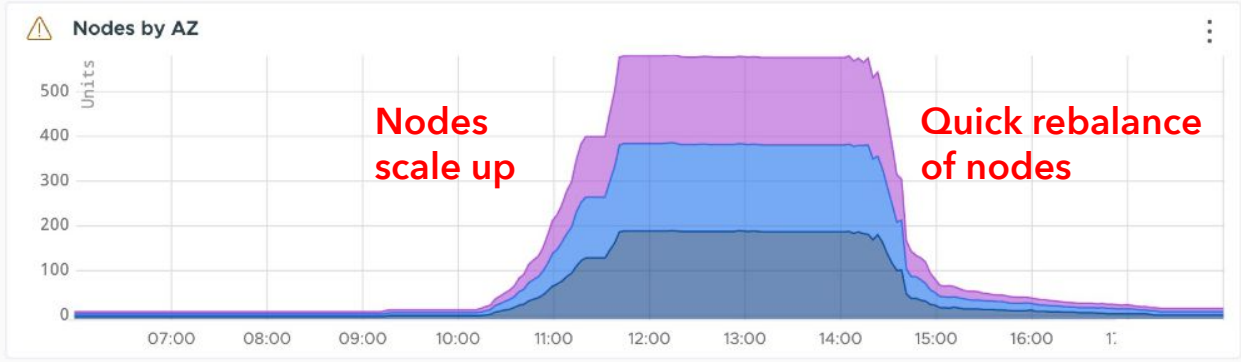
Usable
Compute

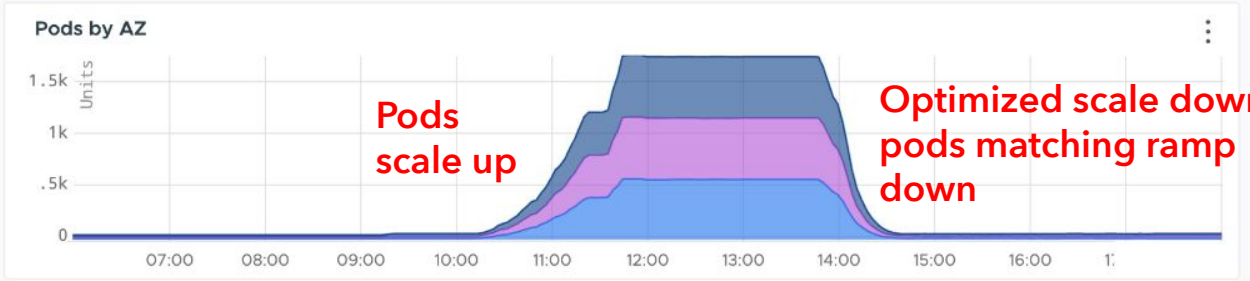
	total memory	app	istio	splunk	certagent	other / buffer (per pod)	sum per pod	pod density
request	27.2	6	0.256	0.256	0.128	0.1	6.74	4.035608309
limit	27.2	7	4	0.5	0.128	0.1	11.728	2.319236016

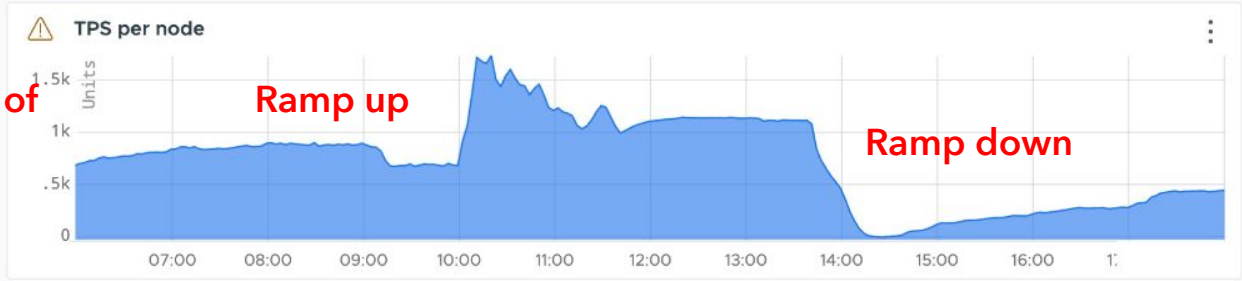
What we saved??

Upto 10% $$ Saving on the Compute.

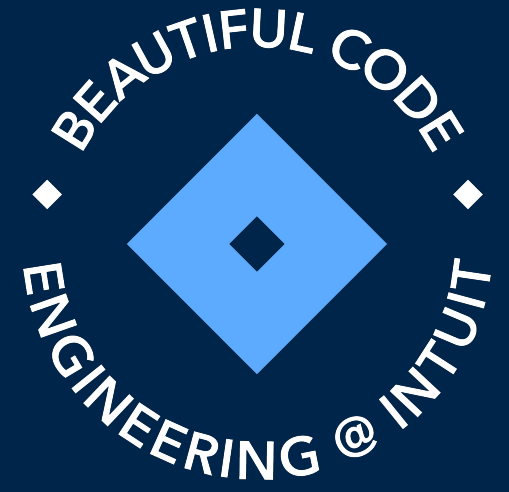
Pod Density and Pod/Node Scaledown



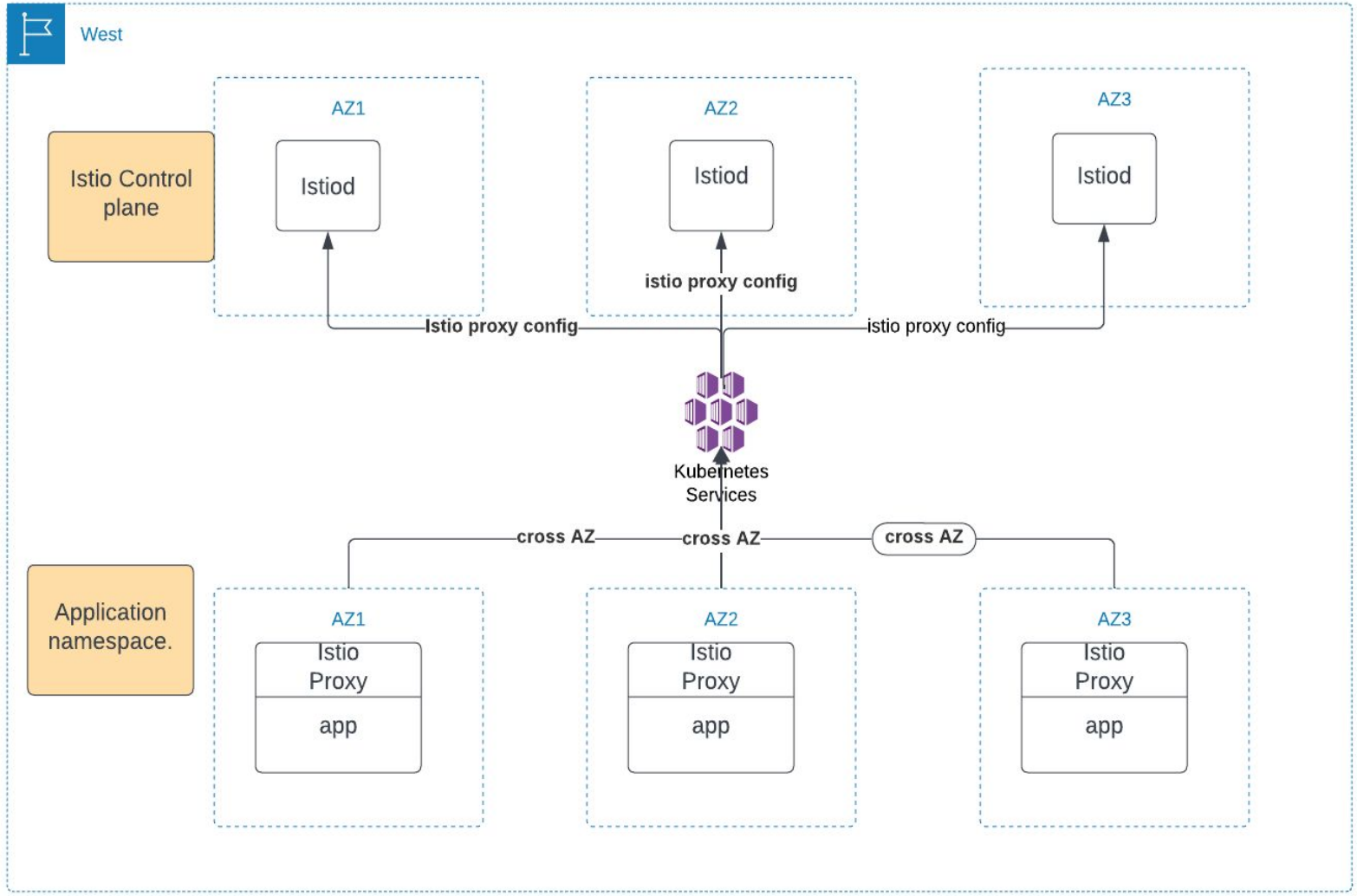




Data Transfer



In-cluster Cross-AZ Data Transfer



Managing Data Transfer

Challenges

- High volume of data transfer within the cluster across Availability Zones (AZs), primarily from IstioD to Istio Proxy, leading to increased costs.
- Difficulty in observing and tracking these data transfers.

Solutions

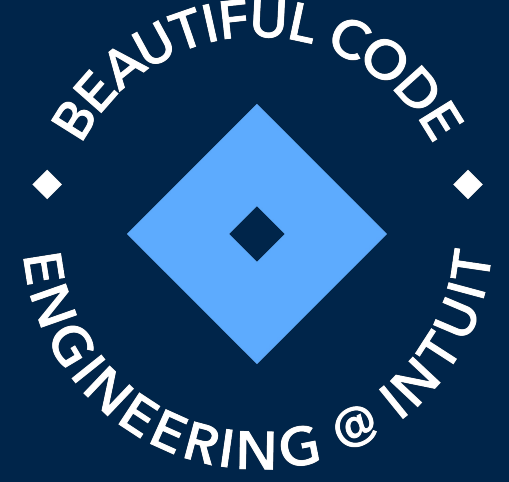
- Minimize data transfer (local caching) and optimize cross-AZ communication.
- Leverage Topology Aware Hints (EKS 1.24) for even pod distribution across zones.

service.kubernetes.io/topology-aware-hints

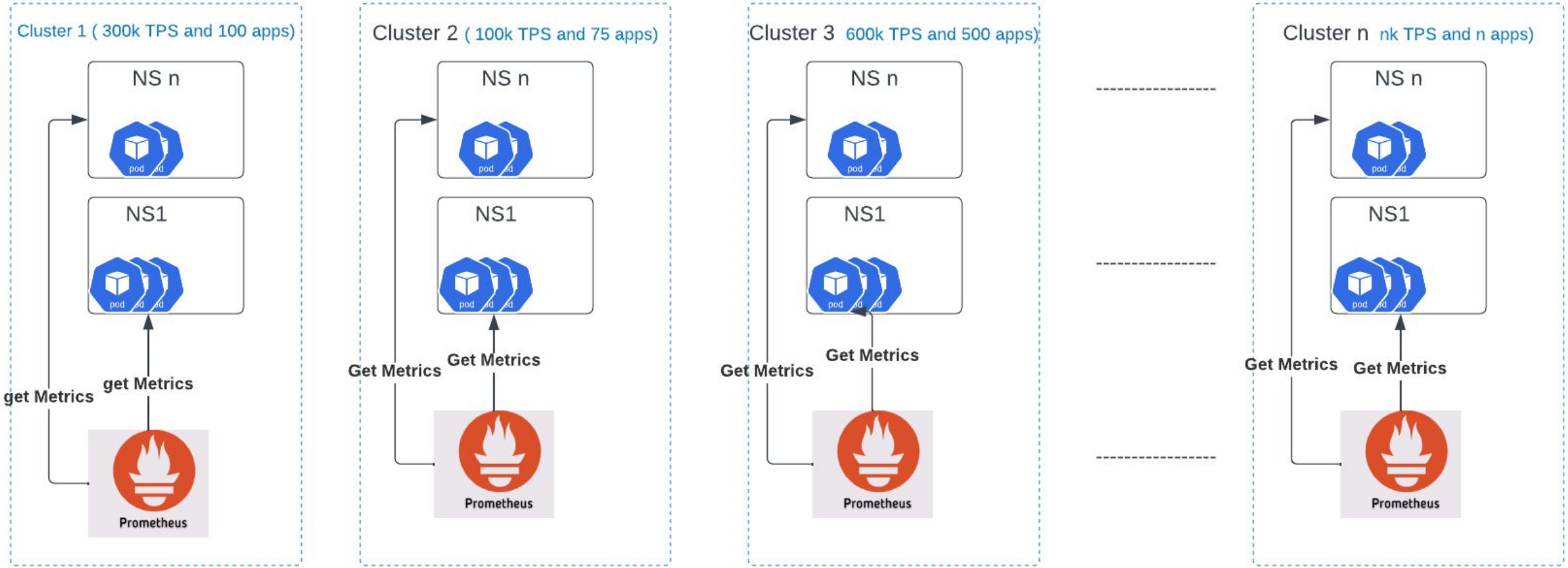
Starting EKS 1.27, new annotations.

service.kubernetes.io/topology-mode auto

Observability Costs



Prometheus deployment



Prometheus right sizing

Challenges

- Increase in cluster monitoring cost.
- Metrics cardinality too high.
- Scrapping irrelevant metrics.

Solutions

- T-shirt sizing based on the instance type for easy operational management.
- Control metrics cardinality with guardrails by limiting number of scrapes per pod.
- Filtering unused metrics in ServiceMonitors.

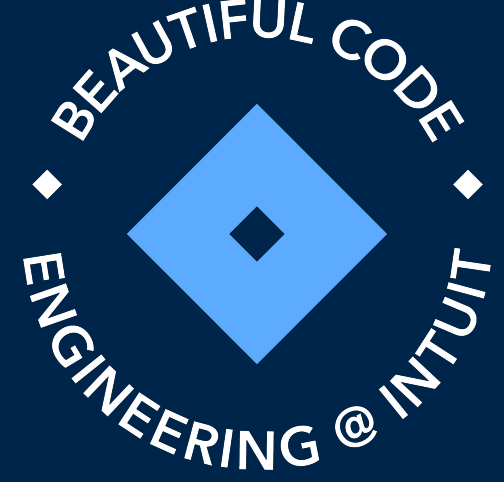
Sample ServiceMonitor CRD for Promenteus

```
+   sampleLimit: 2000
    endpoints:
      - path: /stats/prometheus
        targetPort: http-envoy-prom
        interval: 30s
+       metricRelabelings:
+         - action: drop
+           regex:
  ^istio_(request|response)_bytes_bucket(.*)$|^envoy_cluster_upstream_cx_(.*)$|^istio_(request|response)_bytes_sum(.*)$|^envoy_se
  rver_initialization(.*)$|^istio_request_duration_milliseconds_(count|sum)
  (.*)$|^envoy_cluster_assignment(.*)$|^envoy_cluster_(internal|lb|manager|max|membership|retry)
  (.*)$|^envoy_cluster_upstream_(cx|flow|internal)(.*)$|^envoy_listener_manager_(lds|listener|workers)
  (.*)$|^envoy_server_(concurrency|debug|dynamic|envoy|main|seconds|state|static|stats|uptime|version|worker|initialization)
```

What we saved?

40% reduction in Monitoring costs.

Developer Infrastructure Costs



Development Infrastructure Cost

Challenges

- Hundreds of orphaned Dev clusters leading to high costs.
- Hundreds of Dev namespaces with test resources running during off hours.

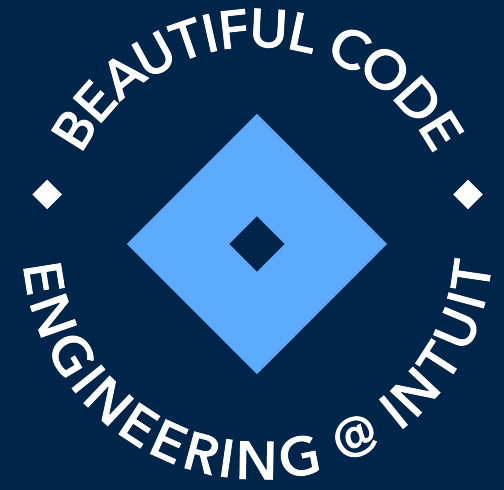
Solution

- Observability for unmanaged Dev clusters.
- Cluster Pauser (Hangman)
  - Scale down all the worker nodes
  - Keep the control plane running to have a option to resume.
- Deployment/Pod Snooze
  - Schedule DEV namespace replica Set deletion.

What we saved?

50-60% reduction in Development Cluster costs.

Takeaways



Takeaways

Stay in the loop

FOLLOW

Intuit Open Source

Don’t miss on exciting OSS events, activities & news

Scan or visit
bit.ly/intuit-oss

Visit our Booth

Get some exciting OSS swag - while supplies last

Feedback