turbonomic
an IBM Company

Automating application-driven container elasticity

For platform and DevOps engineers looking to operationalize speed to market while assuring application performance

Contents

Executive summary

Your competitive advantage depends on how quickly ideas become business transactions, and how well they perform for your customers. Technology is the enabler.

Containers offer the speed, agility, elasticity and scale that's fundamentally changing the way we build, deploy and run these applications. They usher in a world where applications can truly run anywhere; updates and new capabilities can deploy into production several times a day, and dynamic fluctuating workload demand can be managed with elastic infrastructure supply—wherever, whenever. Kubernetes is a platform that can enable organizations to be agile and elastic, but it doesn't manage trade-offs on how to assure performance while being efficient.

For all the simplicity and agility containerization provides, the orchestration platform only provides a way to manage the lifecycle of these services—deploying and maintaining your services in the way you describe.

Container platforms don't natively assure services meet SLOs and can't dynamically manage resources
Threshold-based policies don't solve continuous performance; this approach has never worked, and for the speed of change in container platforms, uncorrelated triggered autoscaling can end up actually causing problems. Elastic infrastructure is key to delivering performance, but needs automated analytics that continuously manages demand, supply and constraints to meet desired service-level objectives (SLOs).

This white paper discusses key concepts to consider for container platform adoption as the way to run your business, and how to protect that investment with automation that assures performance while minimizing cost and remaining compliant. It outlines why you need top-down-driven analytics for a self-managing Kubernetes platform to run your services. Building for multicloud scale early on in your journey gives your IT organization the operational "muscle memory" that will fundamentally transform how—and when—you deliver on more innovation.

A promise of speed, agility, elasticity and scale

Kubernetes enables elasticity; it doesn't automatically ensure you meet and assure application SLOs.

The success of adopting containerization depends on how well you give developers the agility they need, the elasticity required to adapt at scale to continuously fluctuating demands and the assurance the application will perform at the required speed.

Adopting a cloud-native approach and decomposing your applications into distinct sets of services can drive more agile application development and deployment. Containers provide the packaging that makes your services portable and scalable. Kubernetes provides a framework and control points to run your digital applications and services. But to deliver a performant, enterprise-scale platform for your business, you still need to add capabilities to unleash the elasticity enabled by the platform to meet and assure application SLOs.

Deploy faster with CICD and production feedback
The right continuous integration continuous deployment (CICD) methodology, based on automation, is key to achieving faster time to market. In the Google Cloud State of DevOps 2021 report,[1] respondents cited significant improvements because of implementing CICD:

Deployment frequency	**Weekly – monthly**	**Hourly – daily**
Change lead time	More than six months	Less than one hour
Change failure rate	16% – 30%	0% – 15%

With speed comes the need to have a way to manage constant change in production and have a feedback loop regarding how your services are performing, and how to predict what's needed from the infrastructure. The goal is to have a way to define your SLOs, and have the platform provide feedback on how to configure your containers and infrastructure to reduce the risk of performance issues.

- Who decides how resources should be allocated to services? How do they decide it? Stress testing and benchmarking against set SLOs and so on.
- How do you measure performance? Is there a feedback loop in your CICD pipeline to ensure containers and pods are configured correctly?
- How do you ensure that there's always enough capacity for new deployments?

Options	Limitations	Turbonomic answers
Manually analyze container and pod utilization data to determine resource specifications.	– Data collection set up – Labor for analysis	– Top-down, application-driven analytics that determines how to size your containers – Feedback into CICD – Opportunities to reduce requests when not required
Manually analyze resource data from all points in the stack to determine production capacity.	– Labor to collect data from multiple sources – Labor for analysis	Utilization-based analysis to identify resource needs throughout the full stack

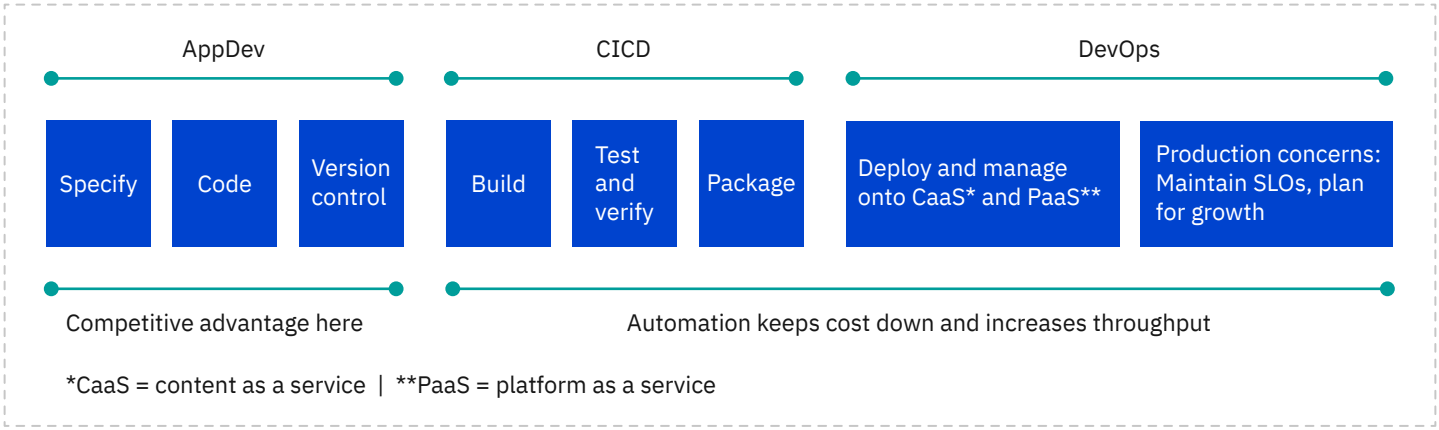


Figure 1. Process for application agility

Platform and infrastructure

Why you need app-driven, full stack management
Regardless of your choice of container platform—or underlying infrastructure—private cloud, public cloud, hybrid cloud, multicloud or even bare metal—the operational challenges of your platform as a service (PaaS) are the same:

- How do you determine whether there's enough capacity to accommodate current and scaling demand?
- How do you decide when to spin up more application nodes?
- How do you decide when to suspend?
- How do you handle peak demand?
- How do you utilize public cloud resources for bursting?
- How do you assure high availability (HA) and resiliency throughout the stack?
- How do you enforce business constraints?

The elasticity enabled by container platforms provides the opportunity to provision for the sum of the average demands of your application instead of the sum of the peak demands of your applications. To take advantage of this ability, delivering a platform that continuously scales up and down as demand fluctuates requires software that continuously makes resourcing decisions to ensure that applications get the compute, storage and network they need when they need it.

Options	Limitations	Turbonomic answers
Run on service providers that provide auto-scaling groups, such as affiliated service groups (ASGs), availability sets and so on.	– Threshold-based policies – Can't scale a specific node: All nodes must be the same constraints, node labels and so on	– Top-down application-driven SLOs – Continuously adjusts infrastructure resources to meet application demand – Continuously scales up, down, vertically and horizontally, the right containers, pods and nodes – Continuously places pods in the proper nodes
Analyze resource data from all points in the stack to determine production capacity. .	– Labor to collect data from multiple sources – Labor for analysis	– Utilization-based analysis to identify resource needs throughout the full stack – Continuously scales up, down, vertically and horizontally, the right containers, pods and nodes – Continuously triggers actions to prevent bottlenecks

Operating for SLOs at scale

The purpose of the container platform is to run your applications at the desired level of service for your business. You need to continuously assure performance as the number of applications grows. Typically, we see customers take over 12 months for the first 1 – 3 applications. For subsequent applications, with the benefit of learned skills and best practices, it can take an additional 6 – 12 months. When lines of business learn what's possible, the scale of the number of individual services to manage is beyond management by humans. Even if you have built stateless services, taking advantage of the ephemeral nature of containers, what's your tolerance for degradation of performance for your end-user's experience? What can you do to manage not only demand, but the increasing rate of change? The answer lies in automation, through actions that are based on an analysis of the trade-offs of how many instances of your service are needed to assure SLO, the configuration of the size and placement of your workload and making compliant resources available from the infrastructure.

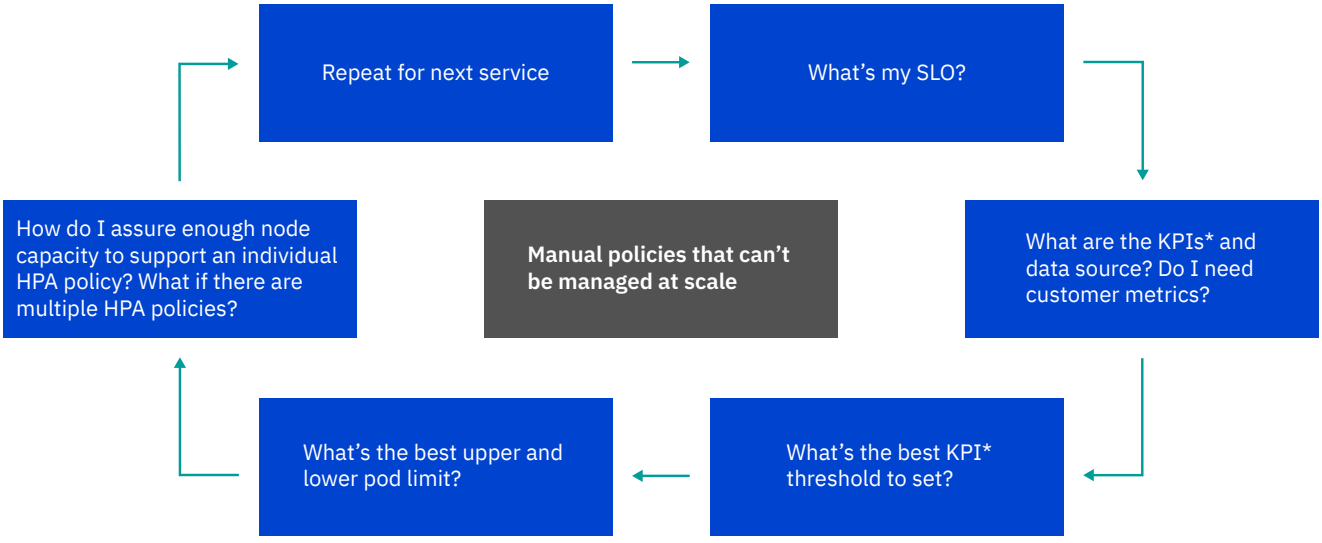
Thresholds don't solve the problem

A container platform will assure you have a minimum number of services available; if one crashes it will attempt to spin it up again. But if you want to assure a good user experience, you want the system to respond before performance degradation and a crash occurs. You can set native horizontal autoscaling to meet demand, but you need to decide what metrics best express the resources needed, configure thresholds and upper and lower limits, test and extrapolate if it will function under production demand, and then repeat for every service deployed.

Imagine if you had over 100 services for a single application? Each of these policies have no correlation with each other. How do you assure that adding more pods of a service isn't introducing congestion in another area? Are you cloning a pod that was poorly configured, and does it need vertical scaling first? How do you manage node congestion, account for noisy neighbors and identify unused allocated resources that could be freed up to meet this demand?

Moreover, configuring your containers, pods, and horizontal pod autoscaler (HPA) or cluster autoscaling policies isn't a one-and-done exercise. Best-guess efforts must be continuously monitored and redefined if they're not. What could your teams do with the time saved if they didn't have to manually set and reset these thresholds?

The importance of getting these configurations right has direct implications for the successful rollout of your digital transformation strategy. A few bad deployments can significantly slow the adoption of the platforms and systems you're building. And too much time and labor spent manually configuring these control points can significantly encumber your organization's ability to become platform-first. Can your business afford that delay? What's needed is a control system that can manage trade-offs across all resources, and define container vertical scaling limits and requests, number of pods needed, and placement decisions to redistribute pods and manage cluster resources using a single analytics engine.



*KPIs = Key Performance Indicators

Figure 2. Manual policies that can't be managed at scale

Options	Limitations	Turbonomic answers
HPA threshold-based policy when to trigger scaling pods in and out	– Configures per service – Based on the average of all pods for the service – Manually defined KPIs and thresholds, and upper and lower pod limits	– Top-down application-driven SLOs – Uses response-time data to drive horizontal scaling of services to meet SLOs – Continuously scales up and down, vertically and horizontally, the right containers, pods and nodes – Continuously places pods in the proper nodes – Continuously adjusts infrastructure resources to meet application demand
Vertical pod autoscaler (VPA) threshold-based policy to vertically scale containers	– Must define for every service – Beta project: use at your own risk – Doesn't access node capacity to take action	
Lets pods crash to redeploy them onto a better node	Poor user experience for transactions on the pod that's ready to crash	
Prometheus observability solutions collect and consolidate data	– Doesn't provide analysis of data – Doesn't provide actions	

An app-driven approach

Application SLOs should drive the infrastructure
Containerization of mission-critical applications is an investment with numerous benefits. But to fully reap those benefits of speed, elasticity and portability you need software to make the right resourcing decisions at the right time, 24x7x365. Otherwise, the complexity will slow you down.

Turbonomic stitches your mission-critical applications to the Kubernetes platform and the underlying infrastructure, essentially wherever your applications run. Based on real-time application demand and accounting for constraints and interdependencies at every layer of the stack—from the logical to the physical—software determines the right actions at the right time to help ensure applications always get exactly what they need to perform. Execute in real time, scheduled or as part of your DevOps pipeline.

Intelligent sizing: How should you size containers?
– Automate with deployment—execute and persist resize as part of the pipeline, for example, YAML, Jenkins and so on.
– Automate in real time—dynamically execute through Kubernetes.

Continuous placement: When do you need to move pods? To which nodes?
– Dynamically execute in real time through Kubernetes. Only for nondisruptive stateless services.

Dynamic scaling: When do you need to scale out or scale back the cluster? By how much?
– Dynamically execute cluster scaling in real time through infrastructure as code or Kubernetes Cluster API.

SLO-driven scaling: When do you need to scale out or scale back pods to meet application response-time SLOs? By how much?

Prerequisites for SLO-driven scaling:
- Applications are designed for horizontal stateless microservices.
- They have a definition and source of SLO data that Kubernetes doesn't provide.

What does this kind of intelligent automation mean for you, your teams and your business? The following are the unique benefits that Turbonomic offers, whether you're running Kubernetes on prem, in the cloud, on bare metal servers or any combination.

"Cruise control" for your apps: Your teams set response-time SLOs; AI-powered software helps ensure that the platform and underlying infrastructure always provide the resources they need to meet those SLOs—wherever the apps run.

Minimize the manual labor: Developers, DevOps and site reliability engineers (SREs) don't need to set thresholds, constraints or autoscaling policies. The software makes the right resource decisions for you, providing actions you can actually automate.

Don't overspend on capacity: No need to rely on developers to make resourcing decisions. They often overprovision just to be safe, right? Our software determines exactly what resource services are needed—all based on application demand.

Confidently accelerate DevOps: Safely increase the frequency and scale of deployments. Our analytics integrates with your DevOps workflows, helping ensure newly deployed and existing services always perform.

Plan for growth with greater ease: Simulate the onboarding of new services with our software. Determine exactly how many more nodes you need to support new growth.

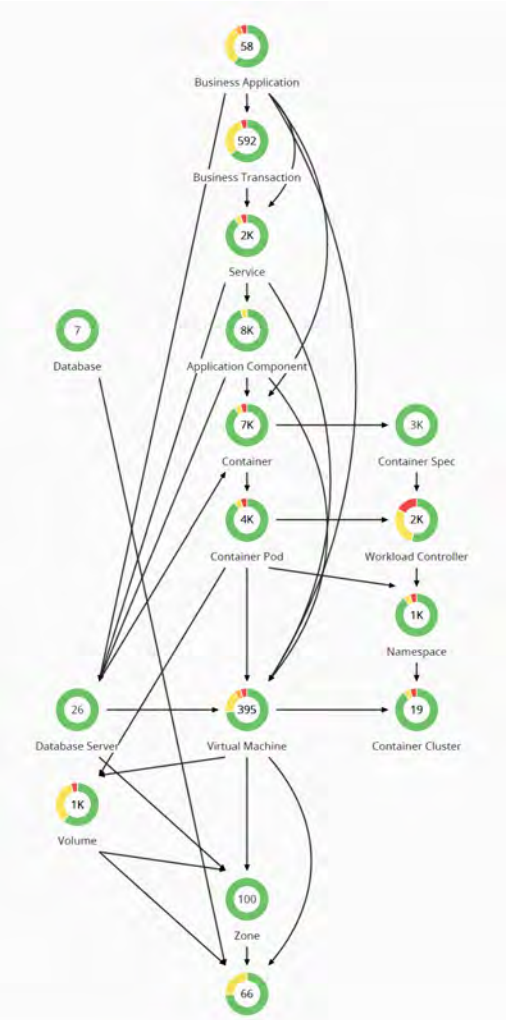
Customer highlight

Accelerating digital transformation during a pandemic

Turbonomic's dynamic resourcing within the Kubernetes platform and the underlying infrastructure kept response-time low.

This customer is one the largest insurance companies in South America with over 6 million customers. Its industry-standard approach to managing the resourcing of existing and next-generation environments was slowing down digital transformation and the company's response to the pandemic.

Turbonomic automation kept response time low during the holiday spike in demand
This customer has a business app that integrates with one of the largest low-cost airlines operating in the region. Travel insurance is booked from this app so the peak we see in Figure 3 is related to the multiday Easter holidays. While demand on the app increased, Turbonomic's dynamic resourcing within the Kubernetes platform and the underlying infrastructure was able to keep response time low.



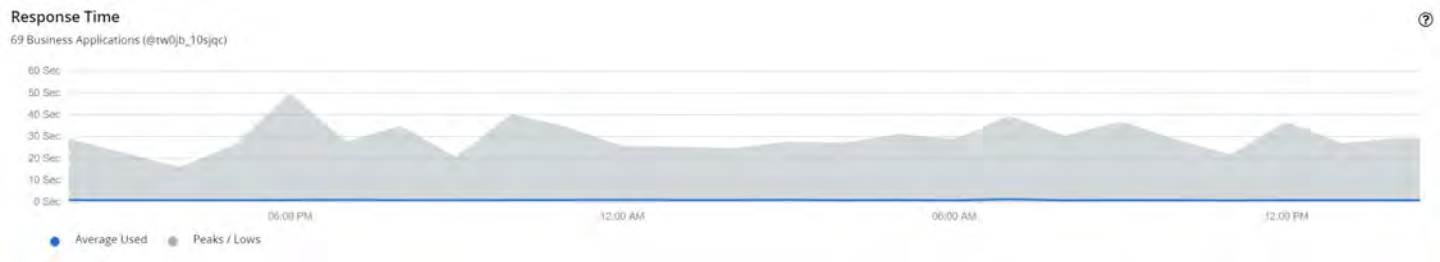


Figure 3. A full-stack view of the individual business application and its response time with automation response time kept low, even during peak demand

57 mission-critical applications
- Example, GPS in car: Report theft of vehicle, quote for new policies and so on
- ~3,000 pods (comprised of ~7,000 containers)
- Stitched to Dynatrace

Automated
- Container resizing (staging)
- Continuous placement (all)

~70%
reduction in tickets

About Turbonomic, an IBM Company

Turbonomic, an IBM Company, provides application resource management (ARM) software used by clients to help assure application performance and governance by dynamically resourcing applications across hybrid and multicloud environments. Turbonomic network performance management (NPM) provides modern monitoring and analytics solutions to help assure continuous network performance at scale across multivendor networks for enterprises, carriers and managed services providers.

To learn more, contact your IBM Business Partner:

i1 Solutions
0823375905 | kduplessis@i1solutions.co.za
www.i1solutions.co.za

IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the United States of America
November 2021

[1] State of DevOps 2021, Google Cloud, 2021

turbonomic
an IBM Company