turbonomic
an IBM Company

WHITE PAPER

Automating Application-Driven Container Elasticity

For Platform & DevOps Engineers Looking to Operationalize Speed to Market While Assuring Application Performance

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 is 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 does not 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 do not natively assure services meet SLOs and cannot dynamically manage resources.

Threshold based policies do not solve for 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 manage demand, supply and constraints to meet desired 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 being compliant. It outlines why you need top-down driven analytics for a self-managing Kubernetes platform to run your services. Building for multi-cloud 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 that your lines of business are looking toward.

A Promise of Speed, Agility, Elasticity, and Scale

Kubernetes enables elasticity; it does not automatically ensure you meet and assure application SLOs.

The success of adopting containerization depends on how well you enable developers the agility they are looking for, the elasticity required to adapt at scale to continuously fluctuating demands, and to assure 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 platfrom to meet and assure application SLOs.

Deploy Faster and CI/CD- Production Feedback

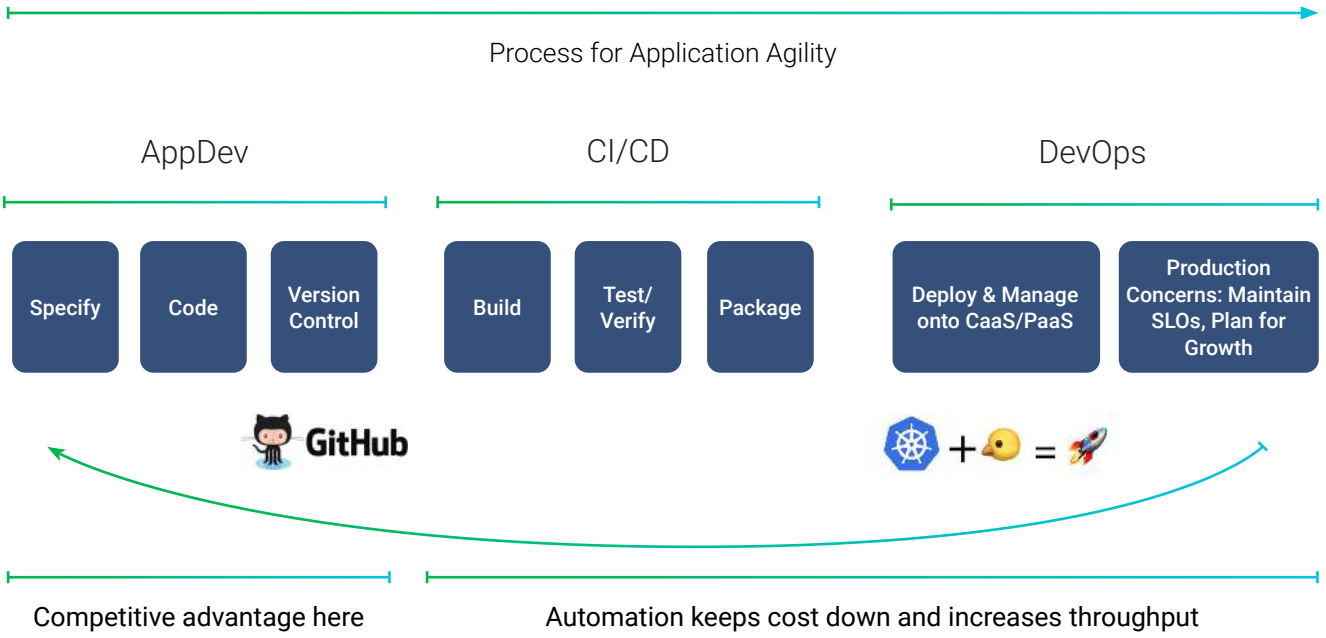
The right Continuous Integration / Continuous Deployment (CI/CD) methodology based on automation is key to achieving faster time to market. In the 2018 DORA State of DevOps report, respondents cited significant improvements as a result of implementing CI/CD:

Deployment frequency	Weekly-monthly	→	Hourly-daily
Change lead time	One – six months	→	One – seven days
Change failure rate	46-60%	→	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 is needed from the infrastructure. The goal is to have a way to define your service level objectives, and have the platform provide back 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 this – stress testing, benchmarking against set SLOs, etc.?**
- **How do you measure performance? Is there a feedback loop in your CI/CD pipeline to ensure containers and pods are configured correctly?**
- **How do you ensure that there is always enough capacity for new deployments?**

Options	Limitations	Turbonomic Answer
Manually analyze container/ pod utilization data to determine resource specifications	• Set up data collection • Labor for analysis	• Top down, application-driven analytics that determine how to size your containers • Feedback into CI/CD • 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



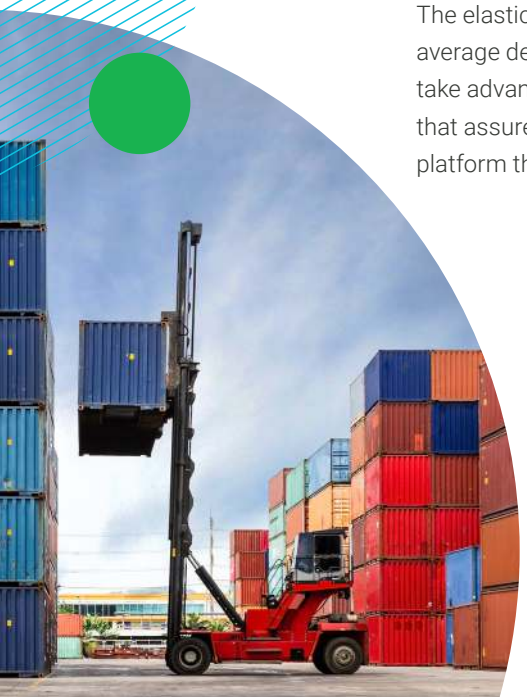
Platform & Infrastructure

Why You Need App-Driven, Full Stack Management

Regardless of your choice of container orchestration technology (PCF or Openshift/ Kubernetes) and/or underlying infrastructure (private cloud, public cloud, hybrid cloud, multi cloud and/or even bare metal) the operational challenges of your 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 leverage public cloud resources for bursting?**
- **How do you assure HA and resiliency throughout the stack?**
- **How do you enforce business constraints?**

The elasticity enabled by container clatforms opens 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 and realize the potential gains a top-down application-driven control platform that assures applications get the resources they need when they need them to perform is required. A platform that continuously scales up and down these resources as demand fluctuates..



Options	Limitations	Turbonomic Answer
Run on-service providers that provide auto-scale groups (ASGs, Availability Sets, etc)	• Threshold based policies • Cannot scale a specific node: All nodes need to be the same (same constraints, node labels, etc.)	• Top-down application-driven SLOs • Continuously adjusts infrastructure resources to meet application demand • Continuously scales up/down, vertically/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/horizontally, the right containers, pods and nodes. • Continuously triggers actions to prevent bottlenecks.

Operating for SLO at Scale

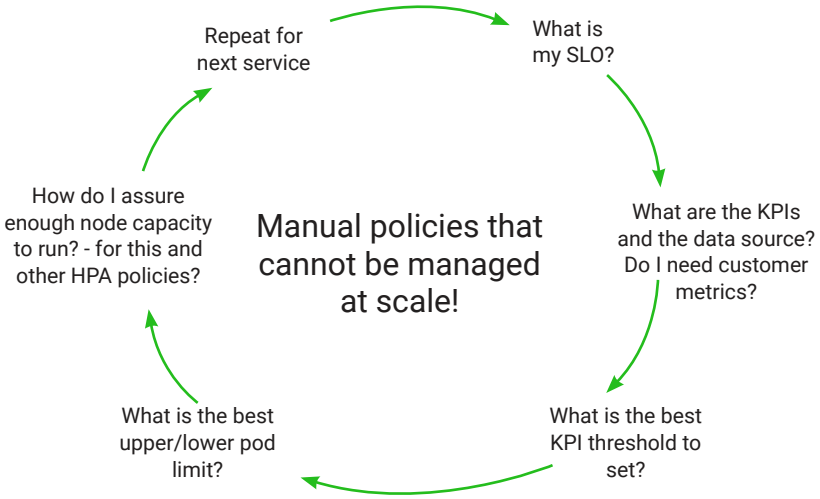
The purpose of the container platform is to run your applications at the desired level of service for your business. You need to continuously assure performance as the number of applications grows. Typically, we see customers take 12+ months for the first 1-3 applications. For subsequent applications, 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, and consider containers as cattle and not pets, what is 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 my service are needed to assure SLO, the configuration of my workload (size, placement), and making compliant resources available from the infrastructure.

Thresholds Do Not 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 look to set native horizontal autoscaling to meet demand, but you need to decide what metric(s) best express the resources needed, configure thresholds and upper/lower limits, test and extrapolate if this 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 is not introducing congestion in another area? Are you cloning a pod that was poorly configured, and is in need of 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 HPA or cluster autoscaling policies is not a one-and-done exercise. Best-guess efforts have to be continuously monitored and redefined if they're not. What could your teams do with the time back if they didn't have to manually set and re-set these thresholds?

The importance of getting these configurations right has direct implications for the successful roll-out 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 is needed is a control system that can manage trade-offs across all resources, 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.



Options	Limitations	Turbonomic Answer
HPA – Horizontal Pod Auto-scaler Threshold based policy when to trigger scaling pods in and out	• Configure per service • Based on the average of all pods for the service • Manually defined KPIs and thresholds, and upper/lower pod limits.	• Top-down application-driven SLOs • Leverages response-time data to drive horizontal scaling of services to meet SLOs • Continuously scales up/down, vertically/horizontally, the right containers, pods and nodes. • Continuously places pods in the proper nodes • Continuously adjusts infrastructure resources to meet application demand
VPA – Vertical Pod Auto-scaler. Threshold based policy to vertically scale containers	• Must define for every service • Beta project (use at your own risk) • Does not access node capacity to take action	
Let pods crash to have them redeploy onto a better node	• Poor user experience for transactions on the pod that is ready to crash	
Prometheus – observability solutions. Collects and consolidates data.	• Does not provide analysis of data • Does not 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 portabilty you need software to make the right resourcing decisions at the right time 24/7/365. Otherwise, the complexity will slow you down.

Only Turbonomic stitches your mission-critical applications to the Kubernetes platform and the underlying infrastructure, 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 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 pipeline (ex. yaml, Jenkins, etc)
- Automate in Real Time…Dynamically execute via Kubernetes

Continuous placement: When do you need to move pods? To which nodes?

- Dynamically execute in real-time via Kubernetes. Only for stateless services (non-disruptive)

Dynamic scaling: When do you need to scale out (or back) the cluster? By how much?

- Dynamically execute cluster scaling in real-time via Infrastructure-as-Code or Kubernetes Cluster API.

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

Pre-requisites for SLO-driven scaling:

- Applications are designed for horizontal stateless microservices
- They have a definition and source of SLO data (K8s doesn't provide)

What does this mean for you, your teams, and your business? Below are the unique benefits that Turbonomic offers, whether you're running Kubernetes on-prem, in the cloud, on bare metal, or a combination.

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

Minimize the manual labor: Dev, DevOps, and SREs don't need to set thresholds, constraints, or autoscaling policies. Software makes the right resource decisions for you, providing actions you can actually automate.

Don't overspend on capacity: No need to rely on Dev to make resourcing decisions (they often overprovision just to be safe, right?) Our software determines exactly what resources services need—all based on application demand.

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

Quickly and easily plan for growth: Simulate the onboarding of new services with our software. Determine exactly how many more nodes you need to support new growth.

About Turbonomic, an IBM Company

Turbonomic, an IBM Company, provides Application Resource Management (ARM) software used by customers to 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.

For further information, please visit www.turbonomic.com

*www.turbonomic.com/resources/case-studies

Customer Highlight

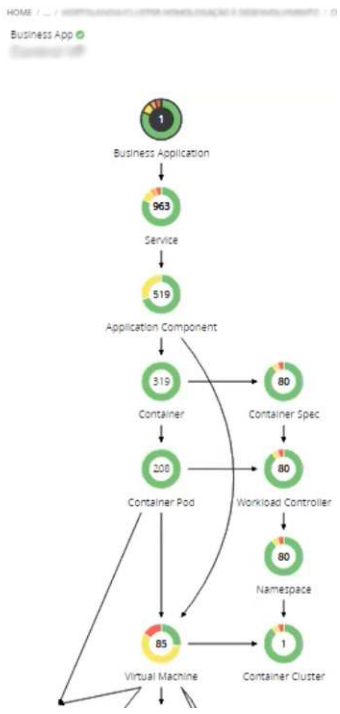
Turbonomic's dynamic resourcing within the Kubernetes platform and the underlying infrastructure kept response-time low.

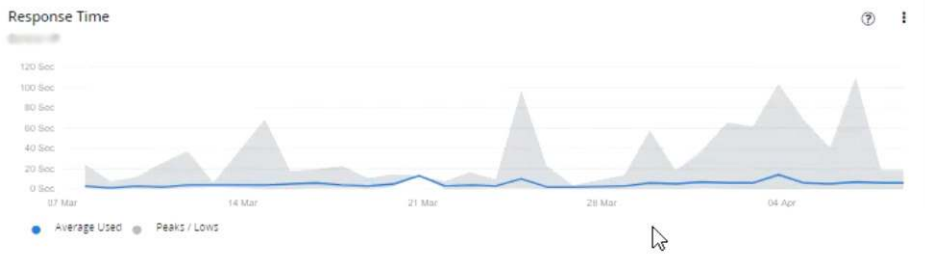
Accelerating Digital Transformation During a Pandemic

This customer is one the largest Insurance companies in South America with over 6 million customers. Their industry standard approach to managing the resourcing of 'legacy' and 'next generation' environments was slowing down digital transformation and their response to the pandemic.

Turbonomic Automation Kept Response-Time Low During Holiday Spike in Demand

This customer has a business app, which integrates with a one of the largest low-cost airlines operating in the region. Travel insurance is booked from this app so the peak we see on the graph is related to the multi-day 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.





57 mission-critical applications

- Ex. GPS in car - report theft of vehicle; quote for new policies; etc.
- (~7k containers /~3k pods)
- Stitched to Dynatrace

Automated

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