Application resource management

300-level live demo script

Introduction

In this growing digital economy, the application is the business. Application performance, therefore, is one of the highest CIO priorities.

Home Robots Inc. is a fully digital company selling innovative “household chores” robots globally via its RobotShop online marketplace. Clients browse and purchase through RobotShop’s microservices-based cloud native app. Promotions and other marketing events, however, generate unpredictable load patterns. This often results in poor application performance and bad customer experiences. Without full stack visibility, IT Ops teams tend to either over or under-provision resources based on best guesses - which is highly inefficient, costly and risky.

In this demo, I’ll show you how IBM Turbonomic, a solution based on the principles of Application Resource Management (ARM), helps SREs and IT Ops teams proactively assure application performance and operational efficiency across their mission critical deployments.

We will:
- See how Turbonomic stitches together a full-stack view from business applications down to the supporting infrastructure
- Examine the resource optimization recommendations generated by analytics
- Demonstrate how to automate the execution of platform-derived “actions”

Note: This demo will be focused on application resource management in private clouds.

1 - Getting a global view of the applications and their infrastructure dependencies

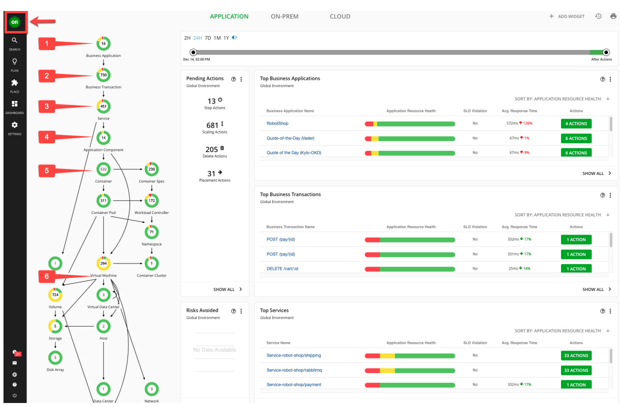
1.1 - Examine the global supply chain

Narration
Turbonomic requires configuration and operational performance data to make resource optimization recommendations. The more data the better.

The RobotShop application is being observed by Instana. Turbonomic ingests data from Instana and builds a common data model to stitch together a graphical view of the application and its resource dependencies. In the Turbonomic nomenclature, this is called the “supply chain.”

The global supply chain models the dynamic relationship of the managed application and its dependent infrastructure layers.

Action 1.1.0
- Log in to the Turbonomic instance and click the **On** button in the upper left corner.
- **Note**: The next six steps will refer to the graphic below.



Action 1.1.1
- Hover the cursor over the **Business Application** entity (1), which shows the business applications of which Turbonomic is aware.

Action 1.1.2
- Hover the cursor over the **Business Transaction** entity (2), which shows logical business functions that an end-user would execute (such as a purchase or add-to-cart). Business applications are composed of these business transactions.

Action 1.1.3
- Hover the cursor over the **Service** entity (3). A service is basically a REST endpoint. Transactions use the services.

Action 1.1.4

- Hover the cursor over the **Application Component** entity (4). Services are hosted and executed in an application component, such as a Java virtual machine (JVM).

Action 1.1.5

- Hover the cursor over the **Container** entity (5). Application components run on a container platform like Kubernetes.

Action 1.1.6

- Hover the cursor over the **Virtual Machine** entity (6). Application platforms are hosted in virtualized environments like vSphere.

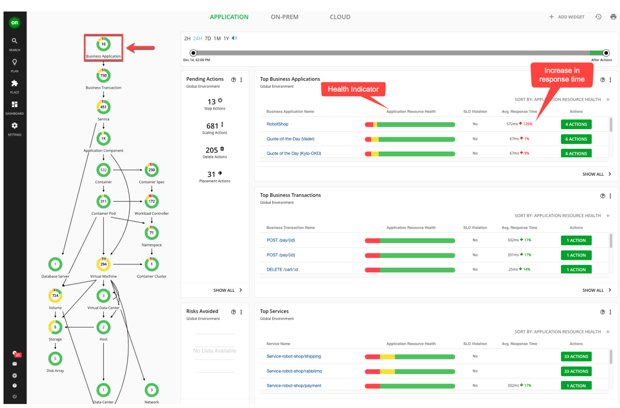
1.2 - Explore the top business applications view

Narration

Turbonomic displays the applications in order of risk.

Action 1.2.1

- Click the **Business Application** entity.



Narration

We see that there's an increase in the average response time.

In the Application Resource Health bar, the color of each circle reflects the current health of the entities: “red” represents performance recommendations, “yellow” represents efficiency recommendations, and “green” is healthy. The current status indicates that there are some critical performance issues as well as some areas to improve efficiency.

The Actions button enables you to take the recommended actions directly from Turbonomic. **Note**: We won't click the Actions button at this time.

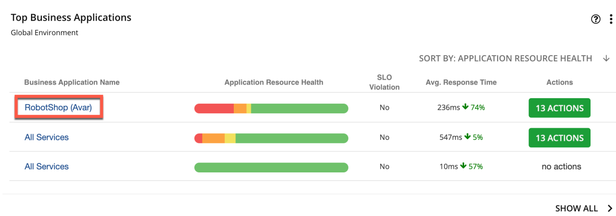
2 - Drilling into the RobotShop application

2.1 - Examine RobotShop resource dependencies

Narration
Now that we have a broad understanding of the global view, let's examine the health of the RobotShop application. This is called "scoping."

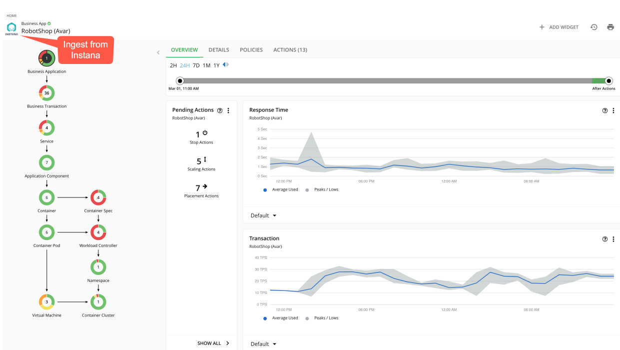
Action 2.1.1

- Click the **RobotShop** link to scope to RobotShop.



Action 2.1.2

- Notice that we are scoped to RobotShop, with data coming from Instana.

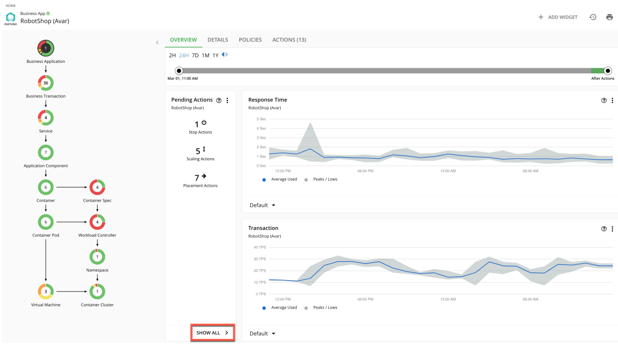


Narration
The supply chain is scoped to RobotShop, and the charts provide a quick view of RobotShop's overall operating health.

Since RobotShop is a Kubernetes-based cloud native application, all the entities in the supply chain are specific to a container infrastructure.

Action 2.1.3

- On the **Pending Actions** chart, click **SHOW ALL**.



Narration

The Turbonomic engine performs an ongoing (real-time) holistic analysis of the environment, generating resource optimization recommendations (and associated actions) that you can follow to resolve and avoid emerging problems.

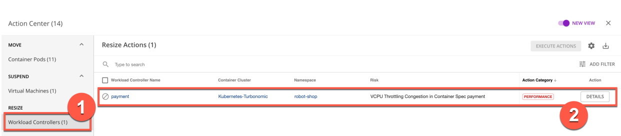
Here we see container resize actions, categorized as ‘performance’ and ‘efficiency’ actions. These are displayed for you to either investigate further or execute manually.

Container resize *UP* actions are typically performance-centric actions that are driven to resolve an underlying resource congestion issue.

Container resize *DOWN* actions are typically efficiency-centric actions that are pointing to a resource optimization opportunity, likely a consequence of resource over-provisioning.

Action 2.1.4

- On the **Action Center** panel, under **RESIZE**, select **Workload Controllers** (1). Then, click **DETAILS** (2) in the **payment** row.
- **Note**: If the view does not match the screenshot below, click **NEW VIEW** at the upper right corner to enable the new view.



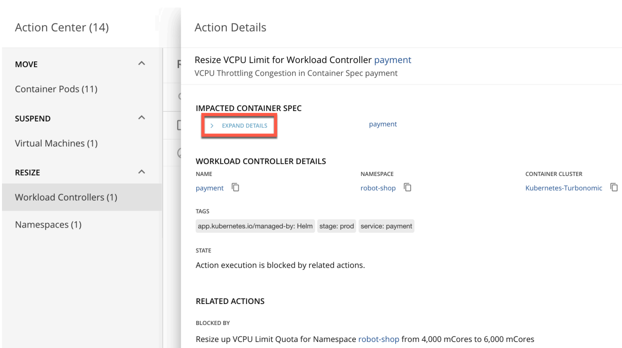
3 - Understanding the resource optimization recommendations

3.1 - Analyze a container right-sizing performance recommendation

Narration
Let’s explore one of the performance recommendations in more detail.

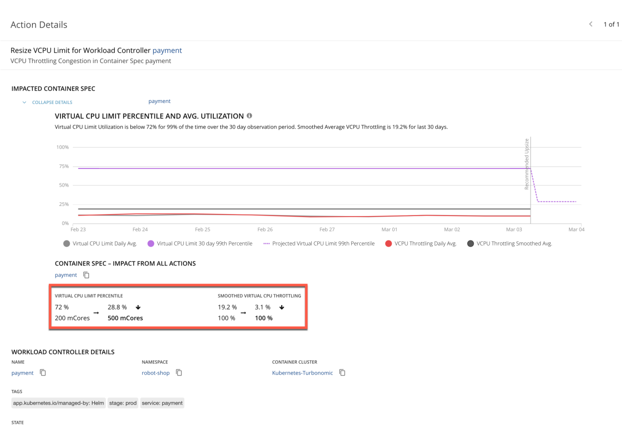
Action 3.1.1

- On the **Action Details** page, click **Expand Details** to inspect the rationale behind the recommendations.



Action 3.1.2

- Review the performance action details. Notice the recommendation to upsize the virtual CPU limit from 200 mCores to 500 mCores.
- **Note**: Since this is a dynamic system, the metrics displayed on your screen may not necessarily reflect the exact numbers in the screenshot.



Narration
CPU throttling directly impacts the response time of a service and therefore user experience. Turbonomic observes and tracks CPU throttling. In the case of RobotShop, the response time latency of the payment service can adversely impact the ecommerce checkout experience. Given the rate of throttling being observed, Turbonomic recommends increasing the CPU limit from 200 mCores to 500 mCores, thereby reducing the CPU throttling from 19.2% to 3.1%. Proactively alleviating CPU congestion pressures by providing these analytics-driven recommendations helps keep the application in the desired state where it meets its defined service-level objectives (SLOs).

Action 3.1.3

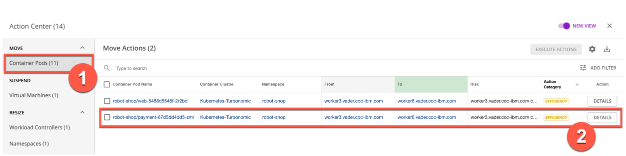
- Click the **X** in the upper right corner to close the **Action Details** page.

3.2 - Analyze a proactive workload redistribution recommendation

Narration
The underlying cloud native environment to which the RobotShop microservices are deployed is highly dynamic. The initial node to which a pod was placed may not always remain the most optimal place to continue executing this workload. An unhealthy application can cause cascading issues, potentially impacting healthy neighboring pods. Additional node capacity may present alternative optimal placement options that were not available at the time when this pod was initially placed. Continuously and proactively redistributing workloads, in line with shifting load patterns and available capacity, helps assure application performance and infrastructure operational efficiency.

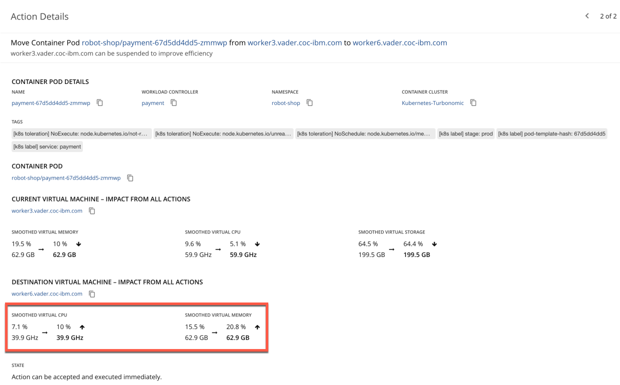
Action 3.2.1

- On the **Action Center** panel, under **MOVE**, select **Container Pods** (1). Then, click **DETAILS** (2) in the **robot-shop/payment** row.



Action 3.2.2

- Review the efficiency action details. Notice the recommendation to relocate the payment pod from where it is currently running on worker node 3 to worker node 6.
- **Note**: Since this is a dynamic system, the metrics displayed on your screen may not necessarily reflect the exact numbers in the screenshot.



Narration

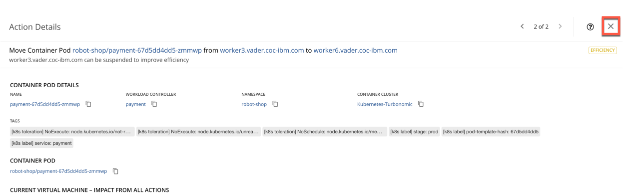
If the underlying node is getting congested, it will attempt to identify an alternate node with more capacity. If, on the other hand, as is the case here, the underlying node is underutilized, it makes sense to proactively redistribute the workload to other appropriate nodes to improve operational efficiency.

The Turbonomic analysis engine computes the current and possible future state of the cluster if the Move action is accepted for execution. As we can observe here, the migration of workloads from worker node 3 to worker node 6 will result in a marginal increase of virtual CPU utilization from 7.1% to 10% and an increase of virtual memory consumption from 15.5% to 20.8% of worker node 6.

Continuous redistribution of workloads helps better optimize overall cluster resources in terms of performance and cost of ownership.

Action 3.2.3

- Click the **X** in the upper right corner to close the **Action Details** page.



3.3 - Analyze an intelligent cluster scaling recommendation

Narration

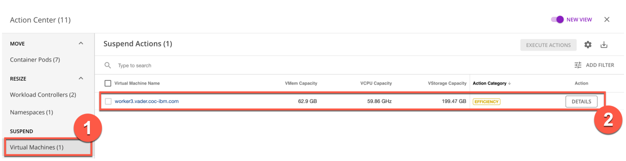
The Turbonomic analysis engine is continuously exploring opportunities to optimize overall cluster efficiency, essentially balancing cluster capacity with workload demand.

Pods consume resources from the underlying nodes on which they are placed. Nodes are represented as virtual machines (VMs) in the supply chain. When pods begin experiencing performance issues due to diminishing resources at the underlying node level, Turbonomic will provide early recommendations to provision additional cluster capacity.

On the other hand, it will seek out opportunities to consolidate workloads on a fewer number of nodes, thereby driving down operational costs.

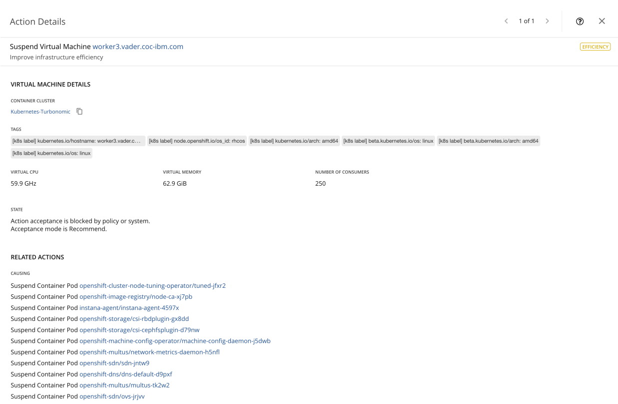
Action 3.3.1

- On the **Action Center** panel, under **SUSPEND**, select **Virtual Machines** (1). Then, click **DETAILS** (2) in the **worker3.vader.coc-ibm.com** row.



Action 3.3.2

- Review the action details.

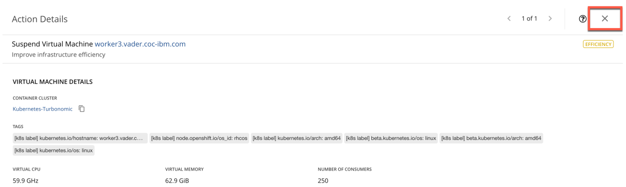


Narration
Executing the Suspend action will result in reclaiming and possibly repurposing the compute resources of worker node 3.

These actions enhance overall cluster operational efficiency and naturally yield significant costs savings, while not compromising application performance and availability.

Action 3.3.3

- Click the **X** in the upper right corner to close the **Action Details** page.



4 - Automating actions [Optional]

NOTE: If you want to demonstrate how to create and configure an automation policy, you will need additional credentials. To do so, log out and log back in using the **cocadmin** username. The password remains the same: **CoC#Rulz!**.

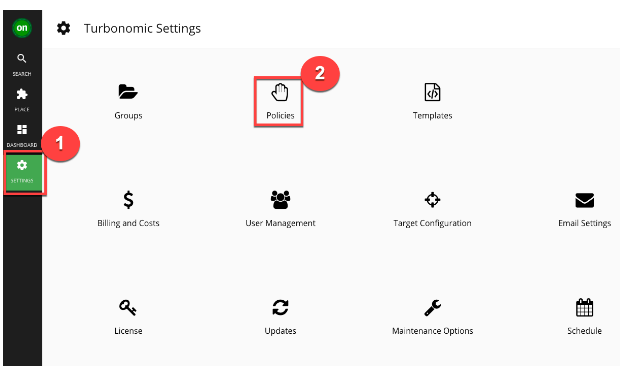
4.1 - Automate the execution of actions and eliminate manual intervention

Narration
Though Turbonomic allows you to initiate an action natively from the platform with the click of a button, it is a best practice to automate the execution of these actions.

We will now define a policy that enables you to automate the platform-derived actions without having to jump into multiple tools. This significantly enhances operator productivity.

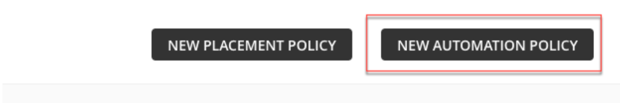
Action 4.1.1

- Click **SETTINGS** (1) and select **Policies** (2).



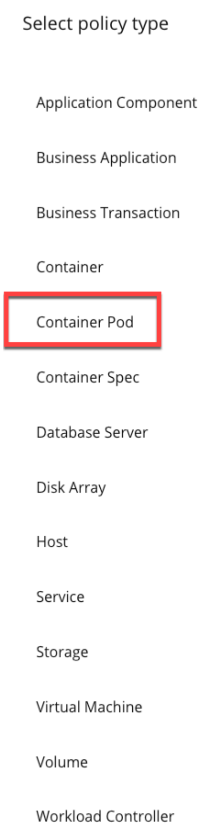
Action 4.1.2

- Click **NEW AUTOMATION POLICY**.



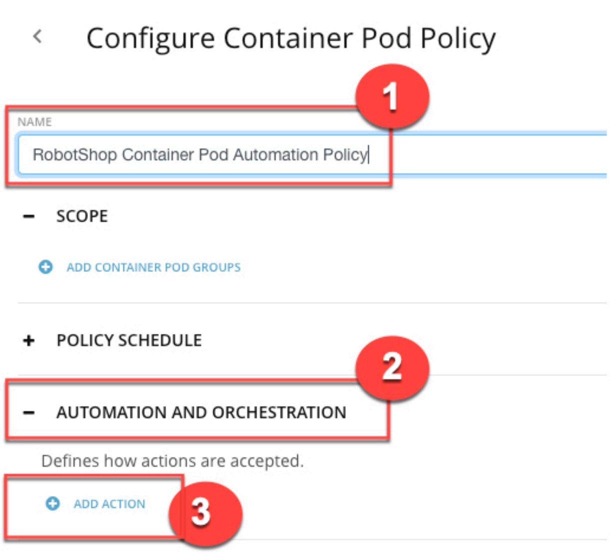
Action 4.1.3

- Select **Container Pod**.



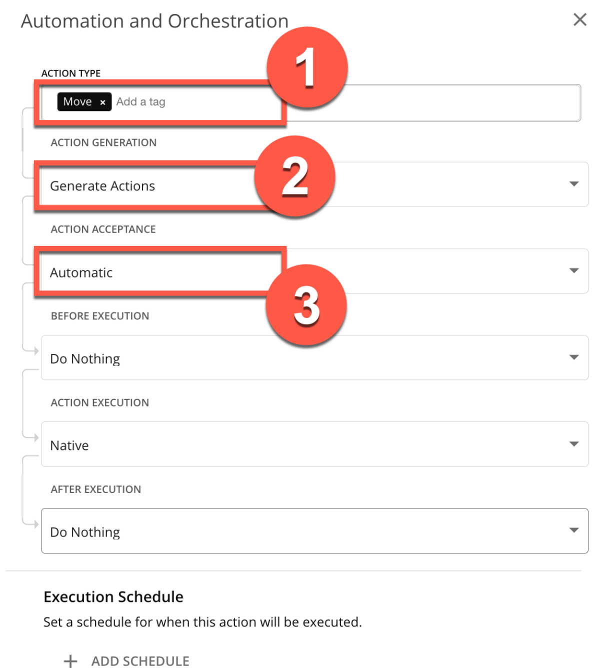
Action 4.1.4

- Give the policy a custom **NAME**. Then, expand **AUTOMATION AND ORCHESTRATION** (2) and click **ADD ACTION** (3) to define how an action is accepted.



Action 4.1.5

- Fill out the **Automation and Orchestration** panel:
  - Set the **ACTION TYPE** to **Move** (1).
  - Set **ACTION GENERATION** to **Generate Actions** (2).
  - Set **ACTION ACCEPTANCE** to **Automatic** (3).
- **IMPORTANT NOTE**: Do **NOT** click **Submit**, as this is a read-only environment.



Narration

Once the automation policy is saved, it will go into effect. All configured actions will now be executed automatically.

The main benefit and best practice of Turbonomic is to execute the platform-derived actions automatically. The underlying goal is to reduce or remove human intervention and leverage automation to maintain application performance and improve operational efficiency.

Although we have demonstrated how actions can be taken automatically from Turbonomic, it is typical that IT organizations first build a level of trust where actions are initially executed manually by a human operator. And as the organization's level of comfort matures over time, we evolve into a semi-automatic and then a relatively fully automatic mode, where an increasing number of actions is being driven by automation.

Summary

In this demo, we showed how Home Robots Inc. leveraged Turbonomic to assure application performance and improve the operational efficiency of the supporting infrastructure. We demonstrated how Turbonomic can augment the well-known benefits of a container platform to provide additional and high-value capabilities, ranging from full-stack visibility and analytics-driven resource optimization recommendations (e.g., vertical scaling, container right-sizing, and proactive workload redistribution) to intelligent platform-derived automatable actions.