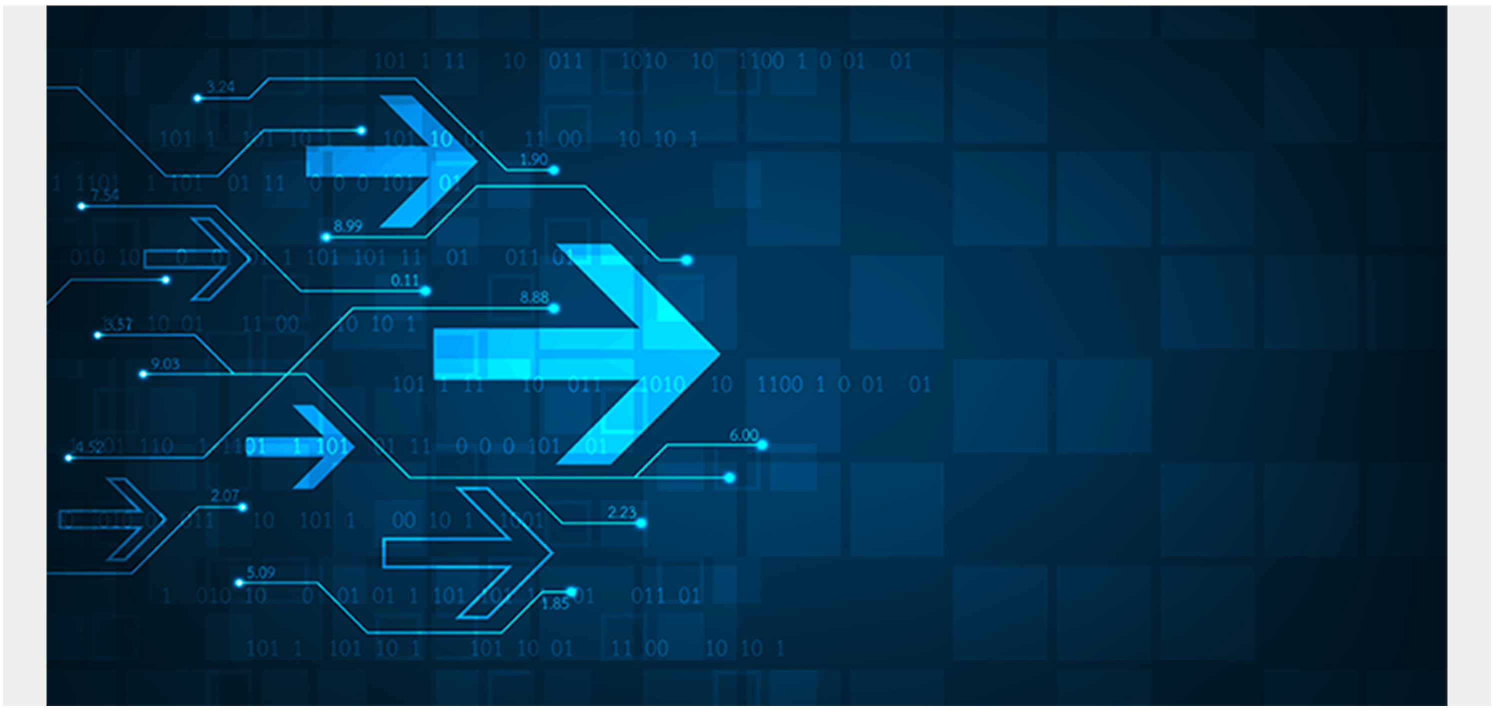
ACCELERATE TESTING USING KUBERNETES MOCK CLUSTERS



Nowadays, many enterprises are moving towards microservices-based and containerized architectures for their applications. Most are using Kubernetes clusters for hosting applications either in the cloud or on-prem.

A Kubernetes cluster is an open-source system for automating deployment, scaling, and managing containerized applications. In fact according to Gartner®, Inc., by 2022, more than 75% of global organizations will be running containerized applications in production, which is a significant increase from fewer than 30% in 2019.[1]

Just imagine: you have over 500 pods or containers running in the Kubernetes cluster within your production environment, and you must keep them alive consistently without any interruption, 24x7. Obviously, to do this, you will need a robust monitoring and operations management solution in place.

BMC Helix Operations Management with AIOps offers best-in-class Kubernetes cluster monitoring and probable cause analysis capabilities. This solution discovers and monitors all entities of a Kubernetes cluster and hosted applications. By combining intelligence, machine learning, topology views, entity health, service health, and probable cause analysis, the solution provides a holistic view of the applications being monitored and managed.

Overcoming challenges in testing Kubernetes cluster monitoring

Each end-user can have their own use case for monitoring an application hosted in a Kubernetes cluster. For example, the use case could be a large number of pods running in an environment, or multiple applications running in different namespaces. The large and diverse number of potential use cases that need to be supported can increase complexity and make it more difficult to achieve faster testing and deployment cycle times.

In addition, optimization of test time is critical in achieving faster software delivery. Many use cases need to be tested in parallel to achieve this but having a separate Kubernetes cluster for each use case requires extra hardware and maintenance that significantly increases costs. Hence, a new way of testing is required, one which can achieve more test coverage within a shorter time, quickly make integration tests, find defects early, and deliver a quality product.

Providing a customer with the desired access to their test environment can be another challenge. End users want to experiment with more use cases on their own to evaluate the product for their specific needs.

The resulting challenges can be summarized as:

1. Multiple Kubernetes clusters for test coverage
2. Time for setting up an environment
3. Availability of hardware resources
4. Maintenance of Kubernetes clusters
5. No sandbox for customers

Mock server for Kubernetes APIs

One way to overcome these challenges is to create a mock server framework to simulate the actual Kubernetes API calls. This involves specifying endpoint routes (e.g., GET/api/v1/nodes) as well as response behaviors (e.g., response payload and status code). The mock server needs to have the ability to mimic the hosted application along with the Kubernetes cluster and simulate the use cases accurately. There are many advantages to this approach:

- It can be used in each phase of the software lifecycle, which could increase the speed of development and accelerate delivery of the solution.
- You can spin off multiple Kubernetes mock clusters in a single virtual machine within a few minutes, reducing maintenance, cost, time, and investment.
- Each mock server can be used to execute different use cases, which increases test coverage.
- The customer can use it as a sandbox to test their own use cases.

The next section will take you through how this method works in a testing environment.

Testing monitoring capabilities with a Kubernetes mock server

The BMC team has designed and developed a mock framework using Python and built using a modern FastAPI web framework running under a Uvicorn ASGI (Asynchronous Server Gateway Interface) web server. The expected API calls for Kubernetes are mimicked using the FastAPI module. The automated script will scrape data from the real Kubernetes cluster in JSON format and then dump it in a mock layer, a one-time activity.

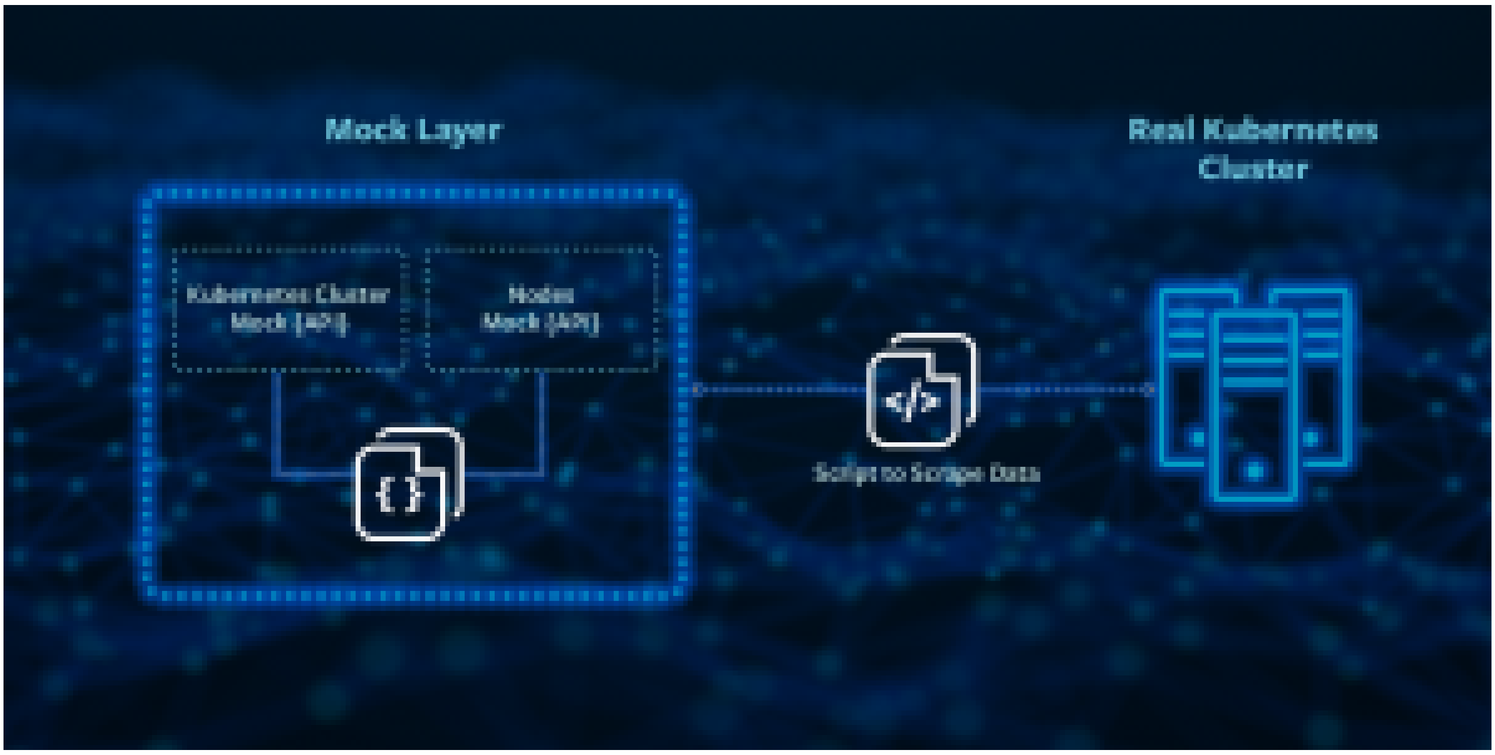


Figure 1: Scraping data from the real Kubernetes cluster and storing it in a mock server

Once all the mock data is available, there is no need for the real cluster anymore. The mock server components are bundled in a Docker container, so you can easily deploy one or more instances of mock servers in a single machine. The container can be deployed on an application platform or your own machine.

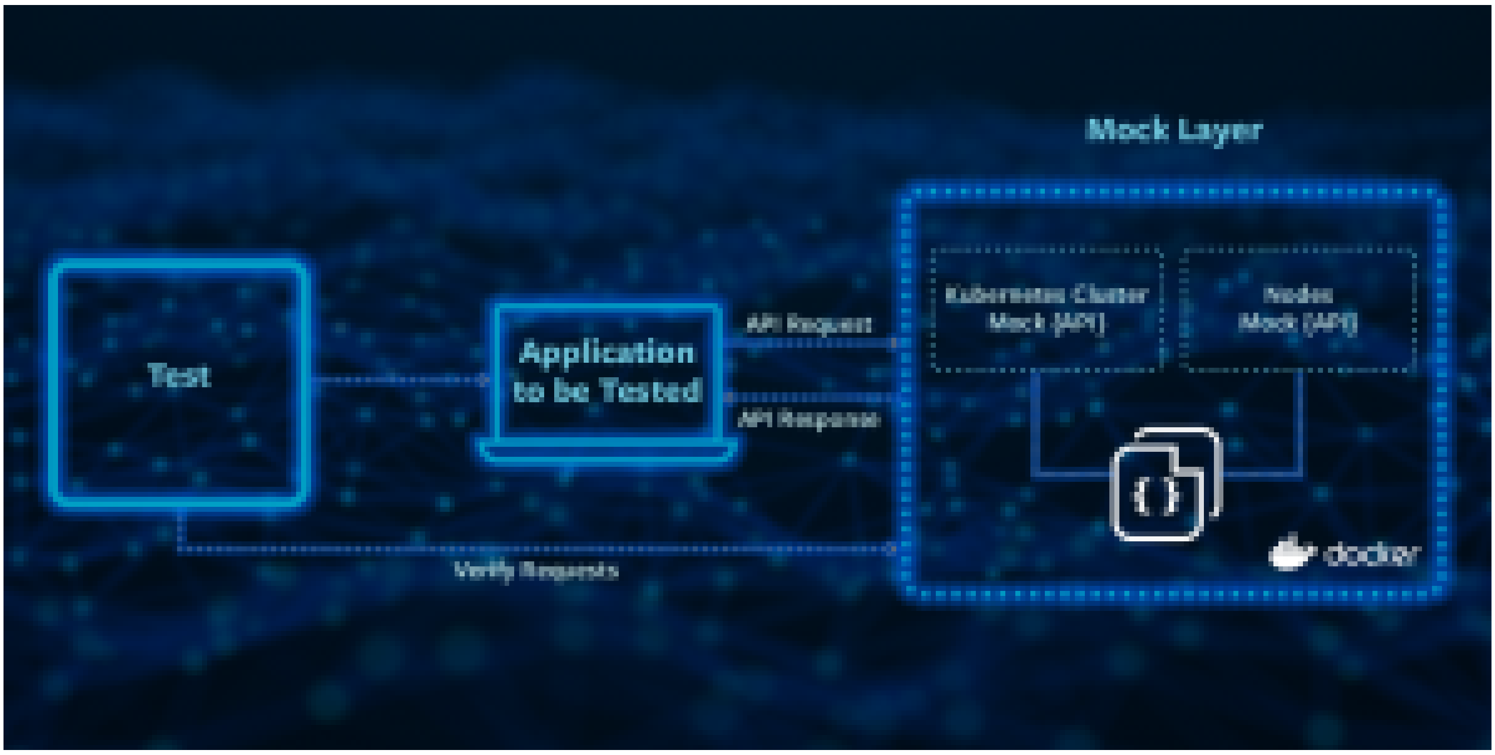


Figure 2: Test execution for an application using a mock server

The monitoring system under test (SUT) should connect to a mock server instead of a real Kubernetes cluster, and should use APIs to retrieve data for monitoring. The mock server matches requests from SUT against mock data, and if it matches, an expected response will be sent back. At the same time, testing scripts can also communicate with the mock server to verify requests sent by

SUT to the mock server.

Here is an example of the end-to-end testing of an AIOps use case on the BMC Helix Platform using a mock server. The workflow of this use case is illustrated in *Figure 3*.

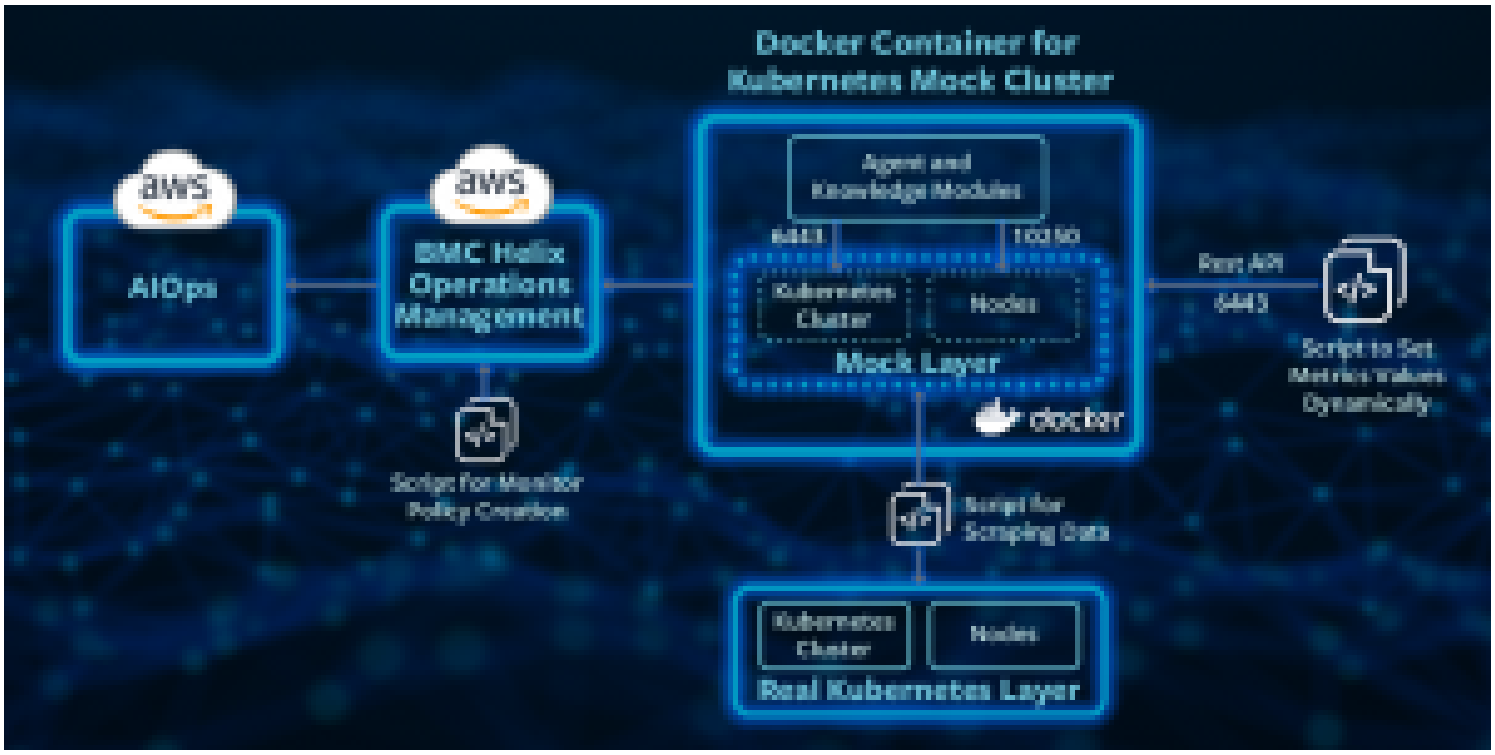

Docker Container for Kubernetes Mock Cluster
aws
AIOps
aws
BMC Helix Operations Management
Script for Monitor Policy Creation
Agent and Knowledge Modules
6443
10250
Kubernetes Cluster
Nodes
Mock Layer
docker
Rest API
6443
Script to Set Metrics Values Dynamically
Script for Scraping Data
Kubernetes Cluster
Nodes
Real Kubernetes Layer

Figure 3: End-to-end workflow while testing different use cases

The test case is to validate the impact of service due to high CPU utilization for multiple pods. The Docker container of the mock server is deployed in the BMC Helix Platform environment. The BMC monitoring agent and Kubernetes Knowledge Module (KM) are embedded in a Docker container. The KM uses REST APIs to retrieve the expected metrics data from the mock server and pushes it to BMC Helix Operations Management through the agent. BMC Helix Operations Management generates events based on the received metric and correlates them with services to set a service health score. In this way, the end-to-end test case can be performed with the help of a mock server.

To do scale tests with more metrics, multiple containers can be spawned to generate a high volume of metrics data.

This concept can be extended to support multiple platforms like Elasticsearch, OpenShift, etc.

Conclusion

Though the challenges of testing enterprise-class software can be daunting, we can overcome and eliminate them with alternative approaches that help you build testing agility. The mock server offers great benefits in validating various Kubernetes clusters and hosted applications, and monitoring use cases.

Gartner, Assessing Kubernetes for Hybrid and Multicloud Application Portability, Alan Waite, Marco Meinardi, 3 June 2020. GARTNER is a registered trademark and service mark of Gartner, Inc. and/or its affiliates in the U.S. and internationally and is used herein with permission. All rights reserved.