bmc

Solution White Paper

# Build and Test Applications Faster with Cloud Lifecycle Management (CLM)

Learn from the dynamic, cloud-based
BMC dev/test CLM environment

# Table of Contents


**1** EXECUTIVE SUMMARY

**2** TACKLING DEVOPS CHALLENGES

Challenge 1. Accelerating the CI/CD process

Challenge 2. Maintaining consistent development and test environments

Challenge 3. Reducing costs of infrastructure and application environments

**3** AUTOMATING THE CI/CD PIPELINE USING CLM

High-level architecture for automating CI/CD pipeline with CLM

Building and testing CLM on CLM

Converting CLM deployable artifacts to service offerings in the service catalog

Taking CLM application environment snapshots

Benefits of using CLM

**6** PROMOTING INNOVATION WITH CLM

**7** CONCLUSION

# Executive Summary

Application innovation demands repeated iterations, testing, and trials by developers. However, obtaining required resources can be a complex, manual process involving dozens or hundreds of development and testing infrastructures and application environments, which must be quickly provisioned and de-provisioned. **Without automation, developers are distracted from their primary goal of creating innovative software.**

To solve this challenge, BMC's internal development team for Cloud Lifecycle Management (CLM) uses CLM itself. **By creating a dynamic, cloud-based development environment with CLM,** BMC engineers worldwide can provision consistent application environments on demand, with a single click.

Benefits of using CLM in the development environment include:

- **Speed** through deployment automation
- **Agility** to build on-demand development, test and production environments
- **Efficiency** in post-deployment management
- **Consistency** across deployments to improve software quality
- **Savings** on infrastructure and administration costs

BMC's experience will show you how **CLM can play an essential role in building and testing enterprise applications,** including deploying full-stack applications, managing deployed environments, and decommissioning resources.

## TACKLING DEVOPS CHALLENGES

As an enterprise software company with a lean DevOps team, BMC faced several challenges in implementing and automating our continuous integration/continuous development (CI/CD) process for BMC Cloud Lifecycle Management (CLM). As you can see in Figure 1, CLM product development goes through multiple stages, including build, test, and deployment.

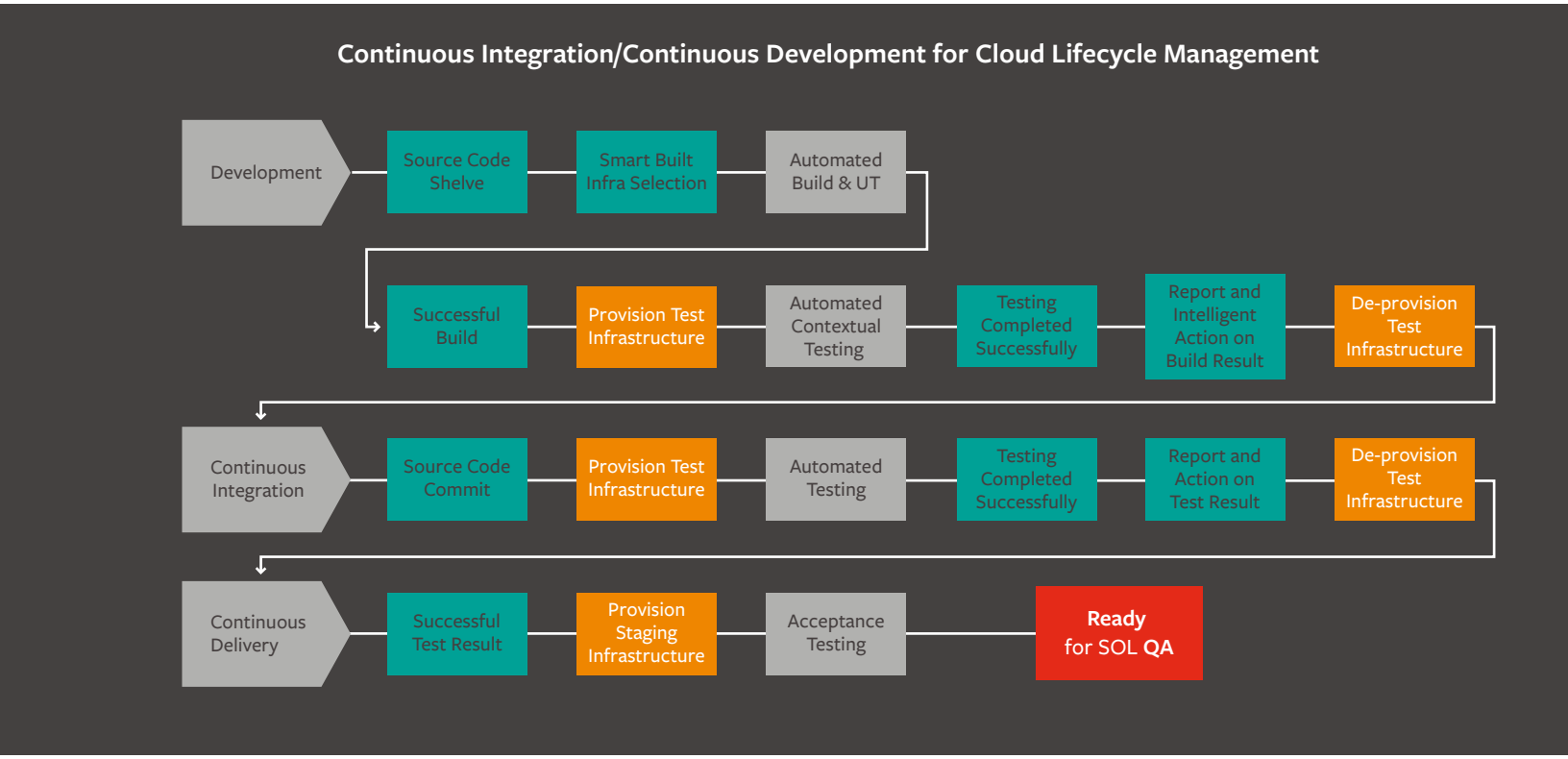


**Figure 1. CI/CD DevOps pipeline for CLM; orange indicates steps where CLM is used to provision/decommission infrastructure dynamically.**

### Challenge 1. Accelerating the CI/CD process

**BMC's manual CI/CD pipeline required engineers to complete many steps** when deploying daily or weekly builds to multiple target environments. Days or weeks of effort were often needed to produce a good working environment, which had a negative impact on developer productivity.

### Challenge 2. Maintaining consistent development and test environments

Consistent environments help developers and testers identify and isolate code defects faster so that they can be fixed quickly and included in subsequent builds. Many **manually built application and test environments were not easily reproducible or traceable**, which wasted resources and delayed actual coding and testing. Also, with developers continually checking in code, the team needed a complete, consistent view of all these check-ins to perform unit, integration, and system testing.

### Challenge 3. Reducing costs of infrastructure and application environments

As a part of the CI/CD pipeline, BMC uses a large number of infrastructure environments for unit, integration, quality, system, performance, and security testing that must be provisioned and decommissioned each day. Prior to automating the environment builds with CLM, infrastructure provisioning was time-consuming and error prone. It also required expensive, dedicated environments to be available at all times. Over time, the number of **dedicated environments continually increased to support new platforms or technologies used by our customers.** Adding to the costs, these resources often remained dedicated indefinitely because there was no automated reclamation of unused environments.

## AUTOMATING THE CI/CD PIPELINE USING CLM

Using CLM along with Jenkins (an open source continuous integration server) and a few other automated testing tools and scripts, the BMC CLM development team created **a complete, fully automated DevOps pipeline for building and testing the CLM application itself.** This approach saves time by relieving developers of the burden of manually building environments.

Other benefits include environments that are more consistent and sanitized, and easy decommissioning of unused resources to save money. Innovation is flourishing because the project promotes rapid experimentation.

### High-level architecture for automating CI/CD pipeline with CLM

There are **two main ways CLM can be used in CI/CD pipeline automation.** A typical CI/CD pipeline consists of three stages—build, test, and production. **Automating the creation and decommissioning of environments** for these stages is the first use case shown. The second use case for CLM is **provisioning multiple versions of application stacks through a self-service portal.** CLM can provide this self-service console for on-demand provisioning of machines as well as full application stacks (see Figure 2).

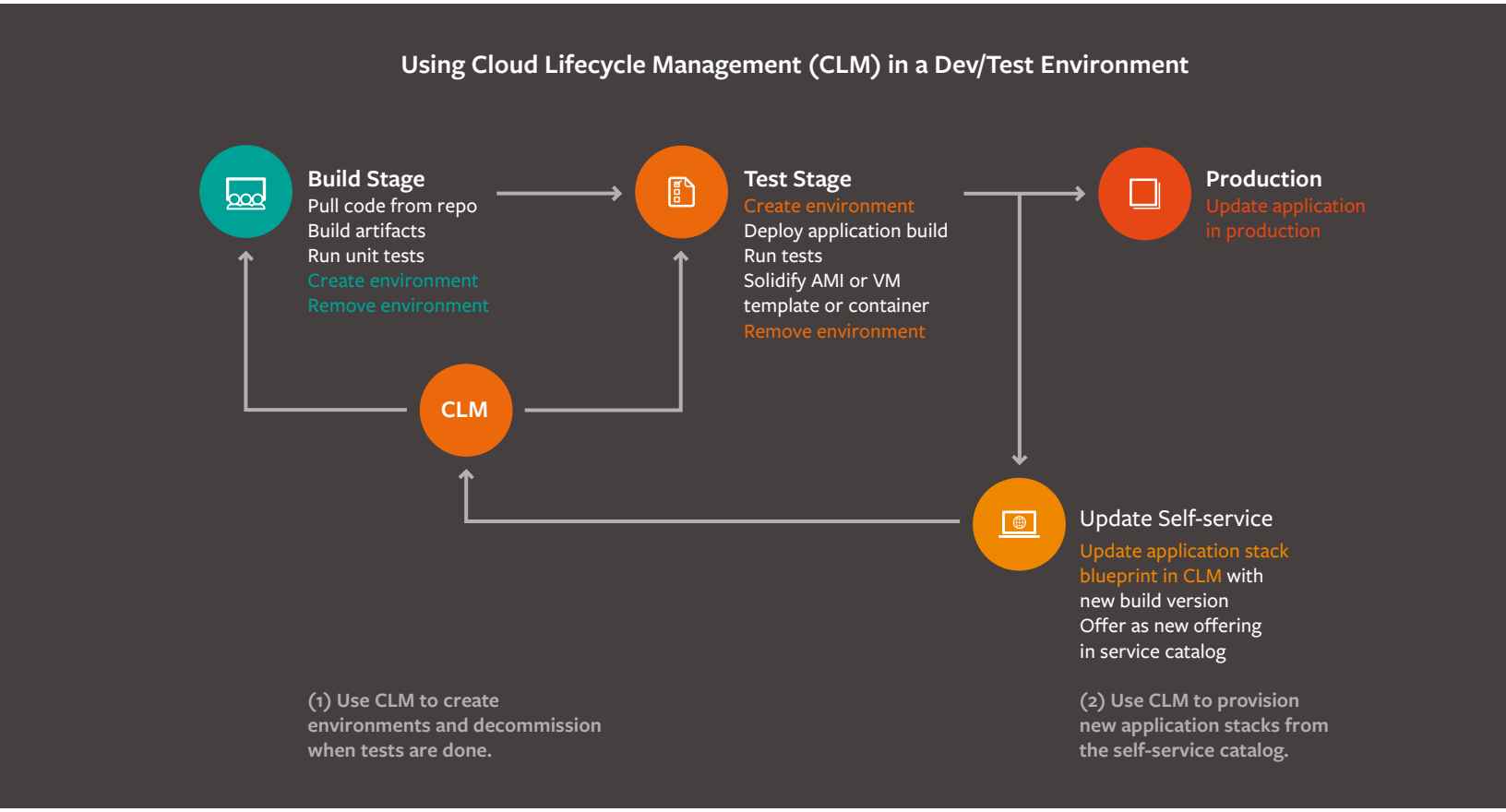


Figure 2. Primary CLM DevOps use cases

Use case #1: A CI/CD pipeline, that includes a Build stage followed by a Test stage, can be implemented by any DevOps release automation tool such as BMC Release Lifecycle Management. Within each stage, developers must dynamically create environments, run tests, and then decommission each environment. **CLM can help users connect and deploy cloud services** to more than 20 public and private clouds without knowing all of the nuances of those clouds. CLM also provides a governance model that allows CLM administrators to control and govern which services are accessible on different clouds, helping to guard against sprawl and improve visibility of usage.

Use case #2: With complete and versioned application stacks being created multiple times per day, the entire application stack with current and prior versions needs to be deployable by a development engineer. During the Test stage, artifacts in the form of AMIs, VM templates, or containers are created after the build is completed, using automated custom scripting to represent the full application stack. These full application stacks are made available in CLM's self-service catalog for on demand provisioning. As an added benefit, **CLM's strong governance capabilities can help control application stack sprawl** in the environment by cleaning up unused systems through automatic decommissioning, helping to reduce infrastructure costs and relieving developers of this chore.

## Building and testing CLM on CLM

As soon as the build is successful, automated testing of the CLM application takes place. Thousands of tests are run on infrastructures that are automatically provisioned using CLM. After tests are completed successfully the hardened CLM application is automatically converted to a "service offering" in CLM for that specific build. The CLM service catalog is updated with this new service offering and made available to the development and test teams for downstream activities such as provisioning the latest CLM application stack for individual testing procedures. Hundreds of **BMC developers use CLM to request and automatically create new CLM application stacks** or environments from the CLM service catalog every day.

Figure 3 shows an example from the service catalog used by the CLM engineering team for on-demand deployment of CLM application stacks for current and older releases. CLM developers have increased their productivity by using the CLM self-service catalog to quickly deploy application environments for development and testing of CLM releases.

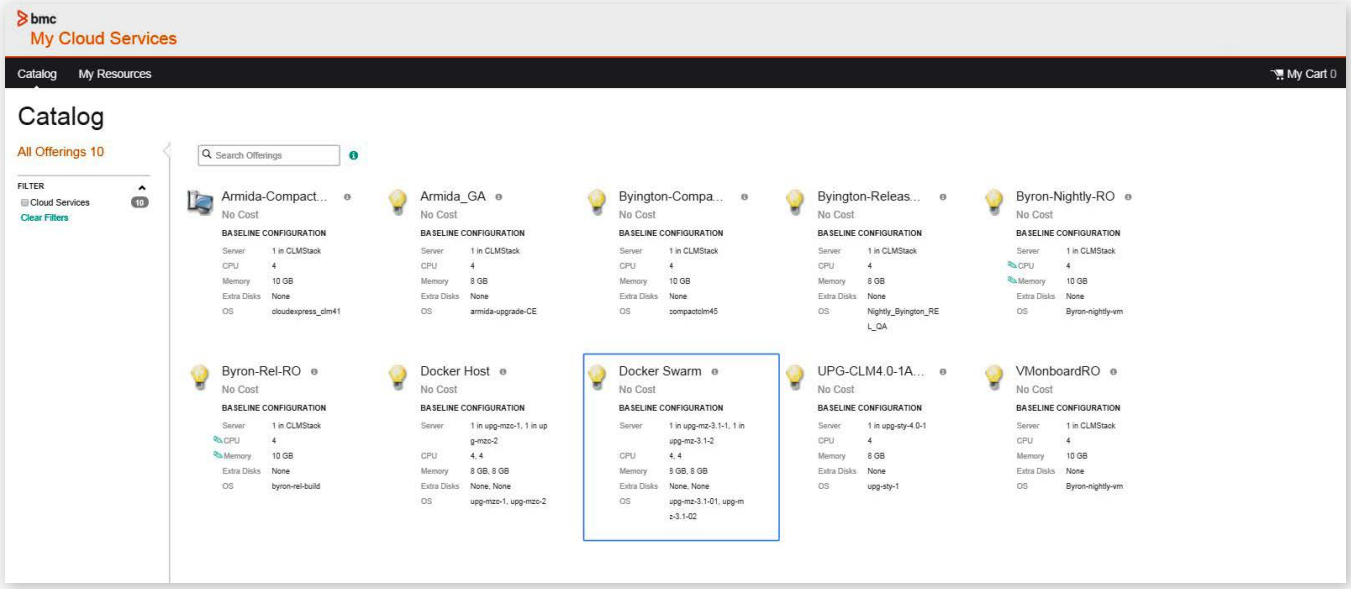


Figure 3. Catalog of CLM service offerings

## Converting CLM deployable artifacts to service offerings in the service catalog

Once the CLM application has been built and tested through the DevOps cycle, **CLM deployable artifacts are automatically converted into service offerings that developers can request** through the service catalog. This process comprises a number of automated steps:

**Dev/Test teams can provision the CLM application stack with one click from the service catalog.**

## Taking CLM application environment snapshots

After provisioning a CLM application stack, a developer can take snapshots of the complete stack using an action such as "TakeVMSnapshot," as shown in Figure 4. A snapshot can be useful for saving application and machine state for debugging during the dev cycle or reverting to a consistent state.

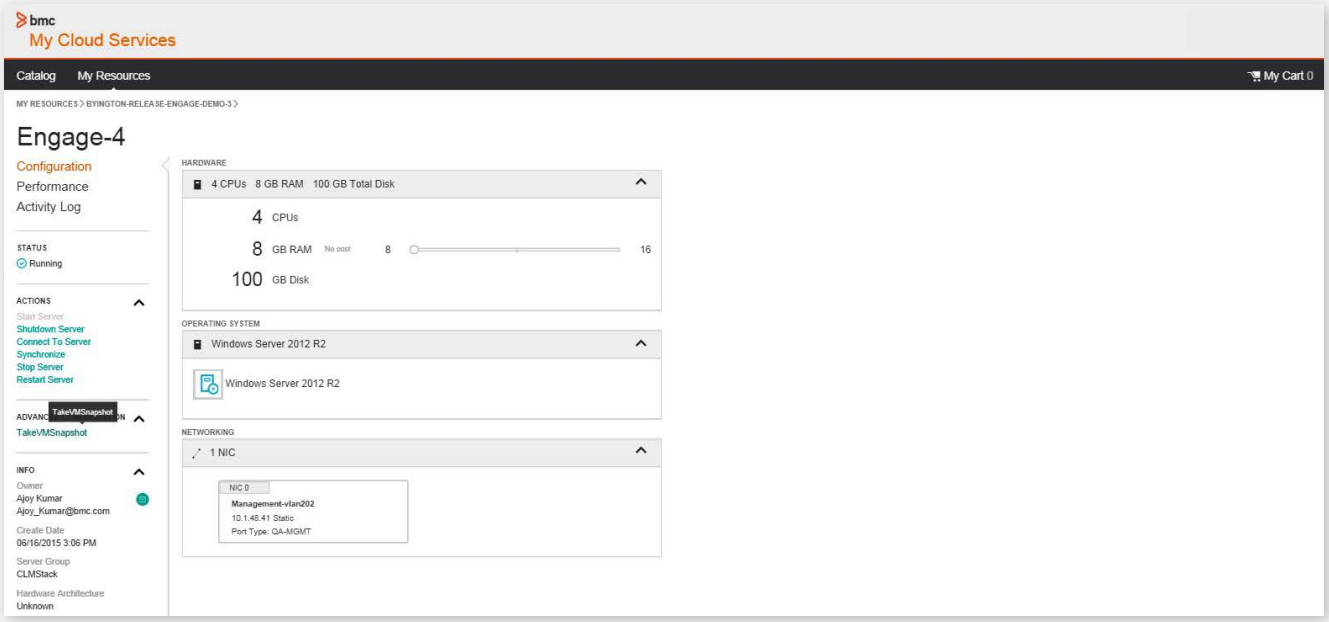


Figure 4. Taking a snapshot of the developer's CLM environment

This new custom action was implemented by creating a BMC Atrium Orchestrator (AO) workflow that implements the action to take a snapshot and configure the Callout Provider, and then uses API calls to import the BMC AO workflow to CLM. Following are the steps required:

This new custom action was implemented by creating a BMC Atrium Orchestrator (AO) workflow that implements the action to take a snapshot and configure the Callout Provider, and then uses API calls to import the BMC AO workflow to CLM. By using Atrium Orchestrator, the team is able to extend automation beyond the limits of each tool to encompass the complete snapshotting process, saving time and eliminating potential errors.

## Benefits of using CLM

Table 1 illustrates the tools used to help automate various tasks of our DevOps pipeline and the benefits derived.

| Task | Tool | Benefits |
|---|---|---|
| CI builds | Jenkins | Automated over 5,000 builds/release in six months |
| Automated deployment | CLM | Hourly, nightly, daily, and weekly deployments are automated using CLM resulting in time and cost savings. In addition, the development team now can deploy new environments more frequently on an ad hoc basis. |
| Automated testing | Silk and Selenium | Thousands of tests run automatically each day. This test automation improves product quality since more tests can be run in a consistent, repeatable environment each time a build happens. |
| Environment infrastructure provisioning and de-provisioning | CLM | On a daily basis, 30 environments are deployed and reclaimed, reducing hardware and software licenses needed. |
| Post-deployment actions | CLM | Each day, hundreds of post-deployment actions such as changing stack configurations, increasing memory, and taking snapshots are performed by developers as CLM self-service actions that don't require a cloud administrator. |
| Service catalog and portal | **CLM** | Over 20 service offerings are accessed by 200 users through the self-service portal. Higher developer productivity has resulted from consistent build and deploy environments and one-click deployment of application environments. |
| **Automated reclamation** | **CLM** | By decommissioning unused application stacks in a timely manner, we reduce dedicated infrastructure, lower hardware and license costs, and conserve power, cooling, and floor space. |
| Development and testing across hybrid cloud | **CLM** | Using CLM, developers can create application stacks across private and public cloud environments without requiring specific cloud skills. This accelerates the testing process by allowing simultaneous tests to be run across multiple cloud platforms. |
| Maintenance of older releases | CLM | Prior releases can now be deployed in one click through service offerings, helping to improve upgrade and compatibility testing and create environments for support purposes. |

**Table 1. Automation tools and benefits**

## PROMOTING INNOVATION WITH CLM

In addition to CLM application stacks as service offerings for our developer community, **we provide many other stacks to support innovation and experimentation.** For example, the open source project Docker has gained popularity with developers as a quick way to package application components in Docker Containers and deploy to a Docker Host for testing. CLM makes this easier with Docker Hosts, available as a service offering in our catalog. Any developer can request and receive a Docker Host in a few minutes, and then start deploying applications or components in Docker Containers to the Docker Host. The CLM development team also plans to leverage additional CLM capabilities to make platform-as-a-service (PaaS) and other middleware application environments available to CLM developers so they can **experiment with new technologies and innovations.**

## CONCLUSION

BMC development teams continuously build, test, and deploy our enterprise CLM application using CLM capabilities. This strategy has enabled the development team to make major improvements in product quality, and to deliver releases faster and far more efficiently, as well as significantly reducing infrastructure costs, particularly for test environment creation and decommissioning.

Just like BMC, your organization can use CLM to effectively manage the DevOps application pipeline and give developers self-service access to machine, PaaS, middleware, and application environments to increase productivity and satisfaction while encouraging innovation.

---

### FOR MORE INFORMATION

To learn more about how your organization may benefit from BMC Cloud Lifecycle Management, please visit **bmc.com/clm**



*475894*