ElasticBox

HOW ELASTICBOX AUTOMATES CI/CD

OVERVIEW

ElasticBox is a cloud application manager platform committed to accelerating software delivery to any cloud. Customers like Netflix, Genentech, DeNA, Splunk, Rightster run ElasticBox to deploy workloads faster and more frequently with reduced risk of failure. Their DevOps teams take advantage of the platform's self-service catalog, collaboration over deployment artifacts, unified configuration management, infrastructure policy control, and CI/CD (continuous integration, continuous delivery). To operate at scale, ElasticBox practices one of its core values and is a huge believer in "eating its own dog food" as a way to continually test and demonstrate the platform. So how does ElasticBox automate its software development lifecycle from delivery to release? This case study showcases how ElasticBox does it using the ElasticBox Jenkins plugin.

THE CHALLENGE

Shipping software involves complex hurdles. Developers maintain source code in a version control system, which is GitHub in our case. We must integrate GitHub with dev, test, staging, and production build environments where every code update deploys seamlessly. Before the code moves to the next phase, say from dev to test, it must pass unit and end-to-end automated tests. The build environments must match the production environment as closely as possible to catch bugs before primetime. Maintaining a uniform, scalable environment across dev, test, staging, and production is not easy. Even a patch, platform, library, or runtime update introduces deployment complexity. Imagine automating builds across any cloud.

Besides managing a complex development lifecycle, ElasticBox specifically wanted to tackle these deployment pain points:

- **Too many manual steps.** In Jenkins jobs, we wrote deployments in command line scripts, which involved too many manual steps and integration points.
- **Long build times.** Code updates in development, test, staging, and production environments took too long to launch or terminate on demand.
- **Cost and waste of cloud resources.** Idle and orphaned machines and resources, when not in use, inflated cloud costs.
- **Labor intensive lifecycle management.** Operations engineers labored days or weeks to manage application lifecycles across multiple Jenkins jobs resulting in a time and cost overhead.

To learn more about how **ElasticBox** can help you catalyse the DevOps movement in your organization or to schedule a custom demo, please email us at **info@elasticbox.com.**

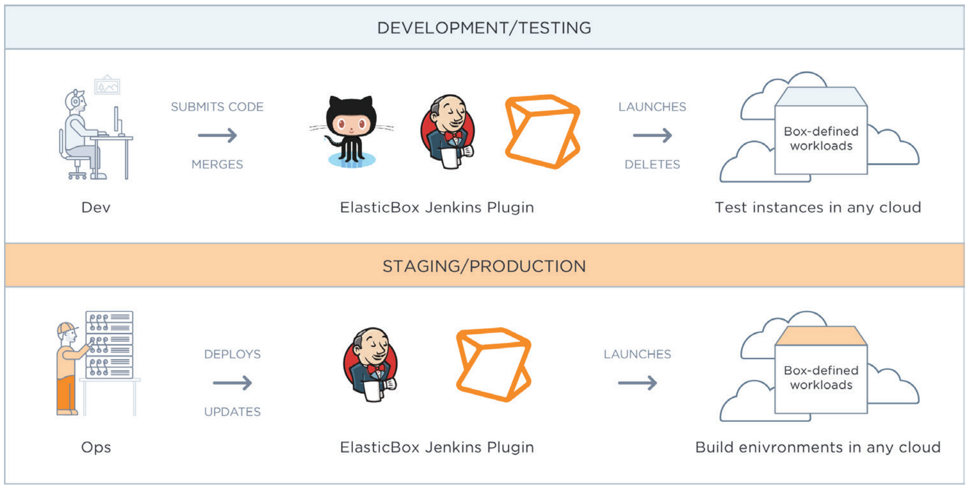
AUTOMATING WITH THE ELASTICBOX JENKINS PLUGIN

ElasticBox achieved full automation as well as managed delivery and release processes in one place using the ElasticBox Jenkins plugin. Here's how we did it.

Simply put, the ElasticBox Jenkins plugin integrates Jenkins with ElasticBox. What that means is in Jenkins, the plugin surfaces the full CI/CD deployment and lifecycle management features of ElasticBox.

Our development, test, and staging environments resemble production environments as close as possible to lower the risk of product defects. In the development environment, for example, we deploy the same stack as production. We achieved this last year using ElasticBox, the plugin, and Git. As GitHub is our source code and version control system, we integrate with GitHub plugins.

Whenever a developer submits or updates a pull request, Jenkins launches a test instance using a Jenkins slave via ElasticBox. When we merge and close a pull request, the plugin instructs Jenkins to delete any attached resources. In this way, we avoid consuming unnecessary cloud resources and adding up cloud bills.



In staging and production too, we deploy and update build environments using Jenkins slaves. Through ElasticBox, the slaves build workloads pre-defined in boxes. Since boxes are infrastructure independent, we attach infrastructure policies to run the builds on any cloud such as AWS, Google Cloud, vSphere, Azure, and more.

We use ElasticBox build steps in the plugin such as deploy, manage, and update to automate the application service lifecycle fully. With the deploy build step, we launch and provision slaves and build environments. It is in these environments that Jenkins installs the application packages. Through the manage build step, we manage the application service lifecycle. It lets us reconfigure, reinstall, start, stop, or terminate an instance. Here again, if an instance already exists, we don't spin up unnecessary machine resources thus saving on additional cloud costs.

To learn more about how **ElasticBox** can help you catalyse the DevOps movement in your organization or to schedule a custom demo, please email us at **info@elasticbox.com.**

“ *For a high-growth software company like ElasticBox, velocity beats everything. In order to compete against large established companies and to innovate for our customer needs, working at scale is an imperative.* ”

— *Carol Carpenter, CEO,* **ElasticBox**

BENEFITS

Integrating the plugin with ElasticBox and Jenkins helped us achieve speed, frequency, and reduced-risk deployments in a CI/CD pipeline. Here are the top four benefits of the plugin:

- **Faster deploy time.** Build environments and slaves launched to any cloud in 30% less time. With 20-30 deployments a day, we saved nearly 100 hours a month of manual overhead.
- **Increased deployment frequency.** In six months, we quadrupled the number of deployments. We built and tested our product 7000 times, on an average of 20-30 times a day. We deployed about 8-10 times per day to staging and shipped about once a week to production.
- **Cost-savings on cloud resources.** The plugin optimizes resource use and cuts cloud costs by killing idle Jenkins slaves and terminating test resources no longer used. For a small operations team, the cost savings ran into several thousand dollars last year.
- **Simpler application service lifecycle management.** The plugin manages the entire software development lifecycle, from a developer committing code and testing it to operations pushing it into staging and production, seamlessly through Jenkins and the ElasticBox build steps. By automating with the plugin and ElasticBox, we saved time, resources, and the overhead of writing and maintaining tons of scripts directly in the Jenkins jobs. Without ElasticBox and the plugin, we could not have achieved complete automation.

NEXT STEPS

To learn how ElasticBox can automate your CI/CD workflows end-to-end, see the ElasticBox Jenkins plugin, try it for free, or contact us for a demo.