USING SPINNAKER WITH KUBERNETES FOR CONTINUOUS DELIVERY

In this era of fast-paced technology, more and more organizations and teams are moving towards agile practices with rapidly evolving software development lifecycles. With that, Continuous Delivery (CD) has become a major part of the DevOps process, where the whole software release process is automated—the build, the testing, the deployment.

With the popularity of Kubernetes for developing containerized applications, the need is growing for a reliable continuous delivery platform with native support for deployments on Kubernetes clusters.

In this article, we will explore Spinnaker, one tool that facilitates integrating Kubernetes into a CD pipeline.

(This article is part of our Kubernetes Guide. Use the right-hand menu to navigate.)

What is Spinnaker?

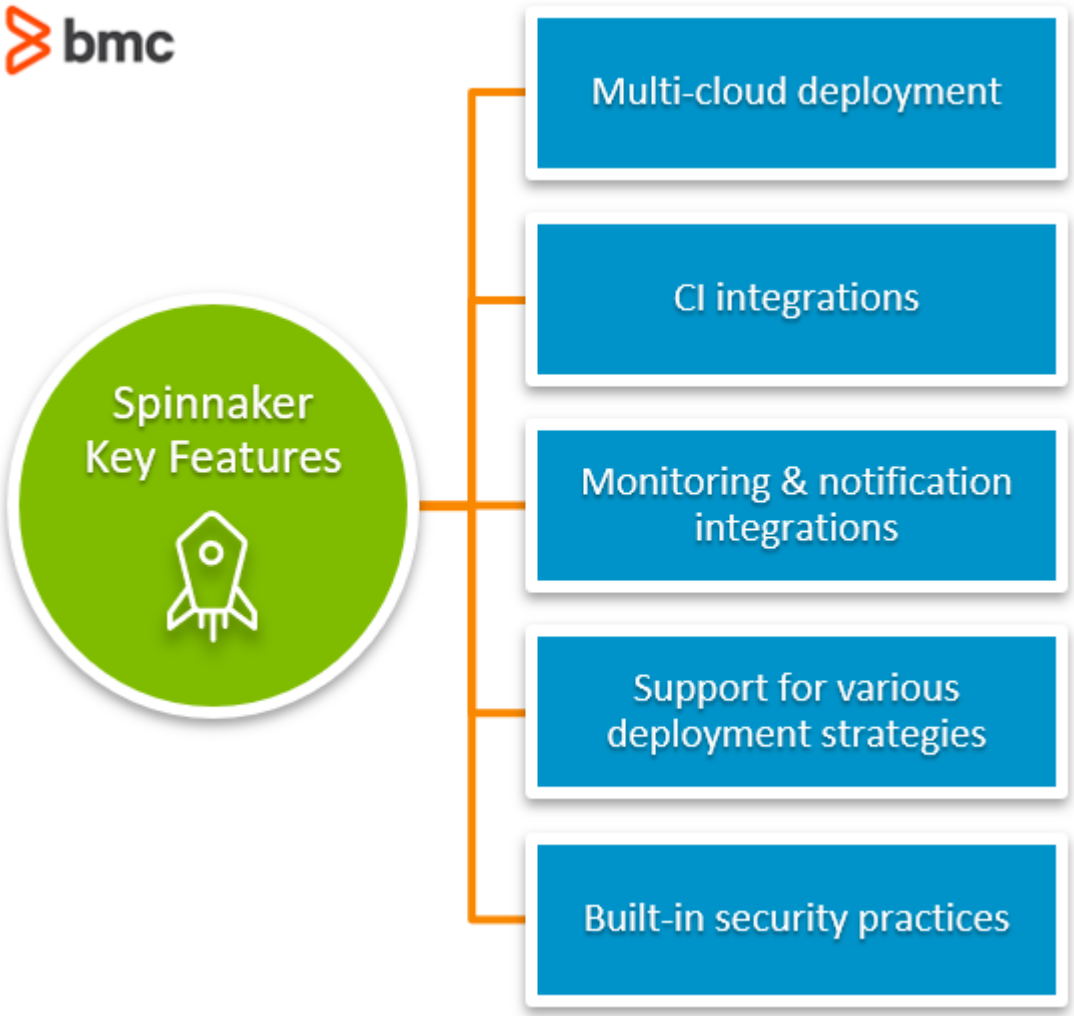
Spinnaker is an open-source continuous delivery platform targeted at multi-cloud deployments. Initially developed by Netflix as a successor for their internal Asgard platform, Spinnaker has now evolved into a standalone continuous delivery platform.

Spinnaker provides the necessary tools to manage your code from commit to multi-cloud delivery with its powerful abstraction layer.

(Read our comprehensive Spinnaker introduction.)

Spinnaker features

Let's look at Spinnaker's key features.



Multi-cloud deployment

Spinnaker natively supports deploying applications in multiple cloud providers. It supports cloud platforms such as AWS, Azure, GCP, Oracle Cloud, Cloud Foundry, and OpenStack.

CI integrations

Spinnaker's robust and automated release pipelines can be easily integrated with other tools such as:

- Jenkins
- Travis CI
- Git event
- CRON jobs

This allows developers to effectively carry out various tasks from artifact collection to pipeline triggering.

Monitoring & notification integrations

Spinnaker can be easily integrated with monitoring solutions such as Datadog, Prometheus, SignalFx, and Stackdriver for metrics collection and analysis.

Additionally, you can configure Spinnaker to work with services such as Slack, HipChat, SMS to provide notifications.

Support for different deployment strategies

Spinnaker includes built-in deployment strategies like highlander, canary, and red/black and can be easily configured to adapt to any custom deployment strategy.

Additionally, Spinnaker's ability to create and deploy immutable images allows developers to:

- Deploy faster with easier rollbacks
- Handle config drift issues much more effectively.

Baked-in security practices

Spinnaker comes with strong security features such as built-in role-based access control with support for multiple authentication mechanisms, including Oauth, LDAP, X.509 certs, Azure Groups, etc. Moreover, it allows convenient isolation of projects for maximum security.

Other features of Spinnaker, such as manual judgments and chaos monkey integrations, enable developers to review and test the deployments for instance failures before releasing them.

Continuous Delivery with Spinnaker

Spinnaker consists of two core feature sets that are helpful in the deployment:

- Application management
- Application deployment

Application management

The application management feature consists of all the necessary tools to view and manage cloud infrastructure. Spinnaker operates on a services model where multiple services are used to facilitate a product. These types of services are sometimes referred to as "applications" or "microservices."

Applications, clusters, and server groups are the key components in the application management feature. They define services. Then, load balancers and firewalls expose those services to the wider world.

- **Applications** are any collection of clusters with load balancers and firewalls. It represents a service that needs to be deployed with all the service and infrastructure configurations.
- **Clusters** are the logical groupings of server groups. Importantly, clusters do not correspond to Kubernetes clusters. They are simply collections of server groups regardless of the underlying hardware or software architecture.
- **Server groups** are the base resource of Spinnaker, which identifies the deployable artifact (container image, source, VM image, etc.) with the basic configurations such as no of instances,

scaling policies, metadata, etc. When deployed, these server groups correspond to a collection of instances of running software like Kubernetes pods or VM instances. Server groups can also be associated with load balancers and firewalls.
- **Load balancer** manages the traffic to each instance of Server Groups by associating with ingress protocols and ports.
- **Firewall** acts as a gateway for managing network traffic access. Furthermore, it incorporates firewall rules defined by IP ranges, communication protocols, and ports to allow traffic.

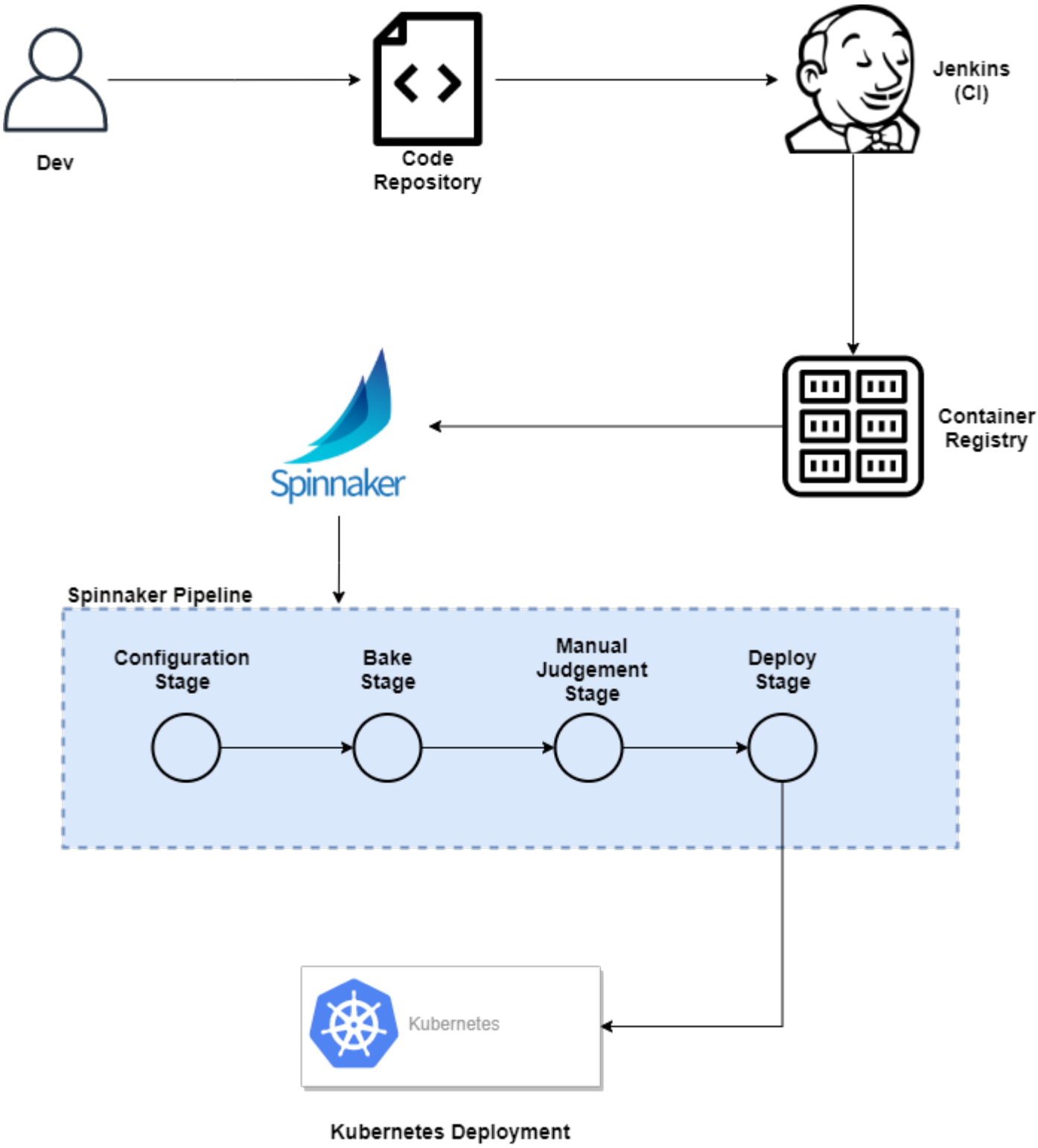
Application deployment

The deployment feature set creates and manages the continuous delivery pipelines. It consists of:

- **Pipeline.** The core component of the Spinnaker application deployment, the pipeline consists of multiple stages where different tasks are performed to deploy the application. These pipelines can be integrated with other CI tools and notification methods to build a seamless CI/CD pipeline.
- **Stage.** This is a set of sequential tasks to be carried out in the pipeline. Spinnaker provides some common stages that can be easily integrated into the pipeline. For instance, we can provide stages such as Deploy, Path, Scale, Undo Rollout, etc., for Kubernetes. Additionally, users can create custom stages utilizing webhooks and run jobs.
- **Task.** The smallest component of the deployment feature, a task consists of one specific task to be carried out within a stage in the pipeline.

Simple example: Spinnaker CD pipeline

The diagram illustrates a simple CI/CD workflow that incorporates Spinnaker to build a container image and deploy it in a Kubernetes cluster:



Let's look at each step of the workflow:

1. Developers commit/push code to a centralized repository.
2. The completed code triggers Jenkins to test and build a container.
3. This container gets pushed to a container registry.
4. Spinnaker listens for a new image in the container registry and triggers the delivery pipeline for deployment.
5. The Spinnaker pipeline creates the necessary configurations (application configs, instances, scaling, etc.) and deploys the container in a target Kubernetes cluster after manual review.

Creating a solid CI/CD workflow allows developers to easily automate almost all the stages of the

development and deployment process.

On top of that, leveraging tools like Spinnaker, which offers a solid continuous delivery pipeline with Kubernetes, will lead to faster, efficient, and less error-prone software development lifecycles. Refer to the official Hello Deployment tutorial for a complete deployment pipeline setup.

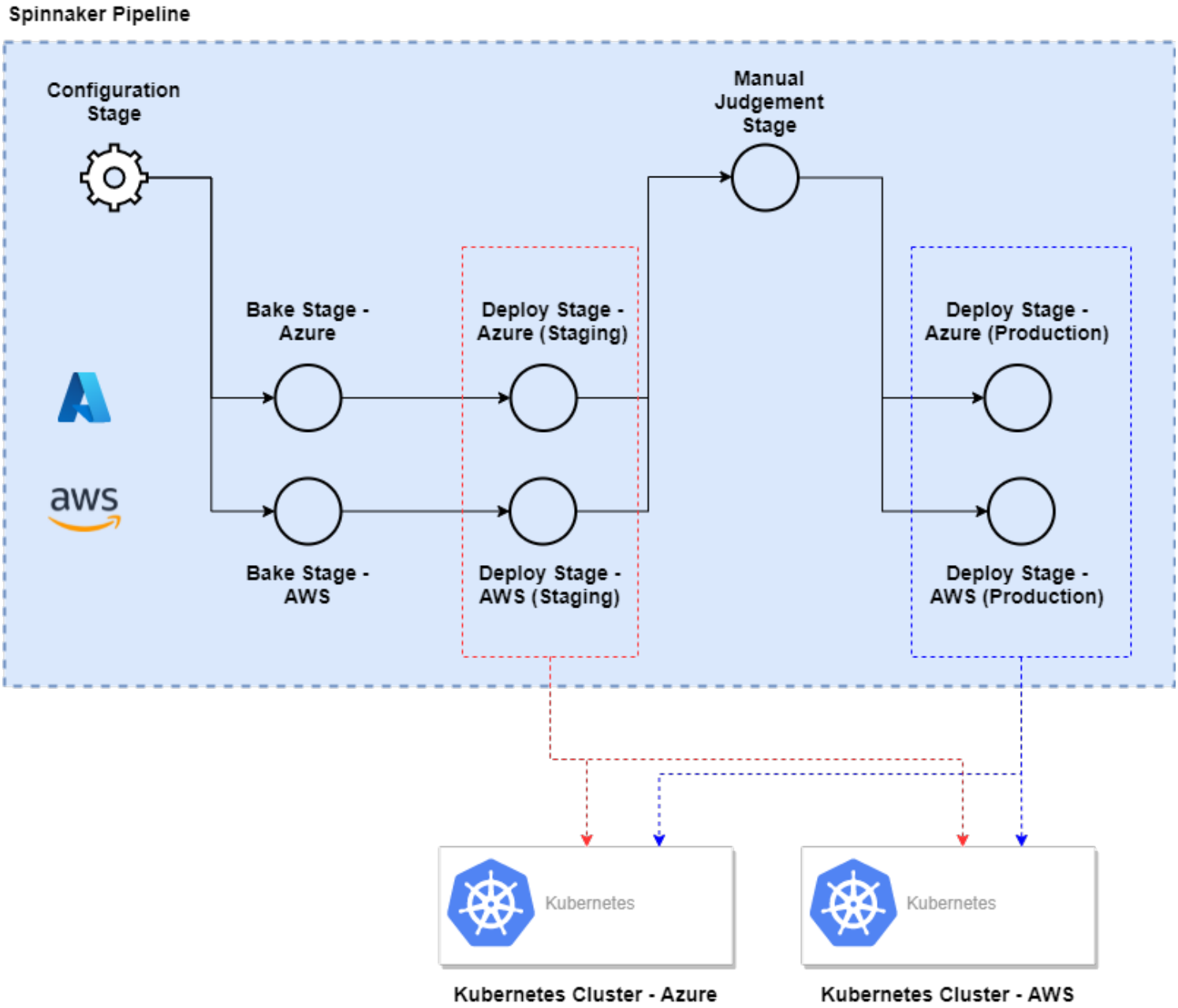
(Set up your own CI/CD pipeline.)

Extending Spinnaker pipelines

In this section, let us expand the above workflow a bit further. Consider a scenario with the following requirements:

- Deploy the application to a staging environment and then push it to the production environment after testing and manual review.
- Deploy the application to Kubernetes Clusters hosted on both Azure and AWS.

We can simply configure a Spinnaker pipeline to accommodate all the above requirements, as shown below.



In the above pipeline, we have defined multiple stages to accommodate deployments in two different cloud providers. (This is possible all thanks to Spinnaker's robust multi-cloud support.) Once the initial configurations are made, the pipeline will execute multiple stages simultaneously to bake and deploy the images in Kubernetes clusters in both Azure and AWS.

Then, you can verify the images and staging deployment at the manual judgment stage and finally deploy them to the production environments of AWS and Azure. Spinnaker's capabilities eliminate the need to create separate pipelines or define different workflows to handle deployments and infrastructure in multi-cloud deployments. All the necessary configurations and infrastructure are managed through Spinnaker.

These pipelines can then be further extended to provide notifications to inform the DevOps teams of the pipeline progression, deployment state, and so on.

Moreover, the ability to integrate chaos monkey allows developers to easily test the resiliency of a deployed application by simulating instance failures and address any identified issues.

The use cases for Spinnaker are endless. Whatever the requirement, Spinnaker can be adopted to facilitate it. Spinnaker can even be deployed inside a Kubernetes Cluster and utilize pipelines to

deploy applications in the host cluster itself.

Robust CI/CD pipeline

The combination of Spinnaker and Kubernetes helps users to achieve the end goal of having a robust and automated CI/CD pipeline for containerized application deployments with baked-in best practices.

We can enjoy the following benefits by leveraging these two technologies.

- Seamless resource mapping using the Kubernetes provider v2
- Easily integrate using advanced development strategies
- Easily rollback deployments. Spinnaker will utilize the versioned manifests of the applications for these rollbacks
- Easily deploy with native K8s support
- Monitor health for the deployed applications and the Kubernetes cluster
- Integrate with multiple artifact stores and image registries to easily obtain applications and container images
- Leverage excellent Kubernetes features like scaling, load balancing, and high availability in the application deployment
- Easily integrate and extend using third-party tools within the CI/CD pipeline

Get the most out of containers

Spinnaker and Kubernetes complement each other to get the best out of containerized application deployment. Spinnaker is the leading platform for managing multi-cloud continuous delivery with an active community that includes Netflix, Google, Microsoft, etc.

Related reading

- BMC DevOps Blog
- Deploying vs Releasing Software: What's The Difference?
- Deployment Pipelines (CI/CD) in Software Engineering
- Testing Automation Explained: Why & How To Automate Testing
- 6 Benefits of Deployment Automation
- Top 5 Best Practices for Software Development