WHITE PAPER–APRIL 2019

Establishing a Software Development Lifecycle for Cloud Native Operations

Tools and practices for continuous delivery

By Hart Hoover, VMware Cloud Native Architect

vmware®

Table of Contents

Introduction

As enterprises look to release software at a higher velocity, they are turning to containers as a standardization method to ensure consistency. To manage those containers across clouds and on-premises infrastructure, companies are adopting Kubernetes as a common API between these environments. Many of the organizations that are adopting Kubernetes are doing so to leverage a cloud native approach to building and delivering software.

This paper describes the tools, practices, and patterns that some of our customers are using to adopt a cloud native mindset. A curated cloud native application toolset, enabled by open-source software, allows teams to gain more insight into their applications.

Once Kubernetes is up and running, developers need to be able to deploy application software to clusters for testing and production workloads. Companies that can deploy software more frequently with measurably lower failure rates are likely to be more productive and have a larger market share. Although the industry has called this software deployment lifecycle *continuous delivery*, there are several stages of continuous delivery:

Continuous development: How do developers get fast feedback on what they are making? Do developers have to engage with other teams to get the resources they need to write and test code for their applications? In a cloud native world, applications don't live in a monolith but are interdependent on other microservices. Teams that can develop continuously against other APIs and services, while making small changes at a rapid rate to push software forward, make a larger impact than those that do not.

Continuous integration: Once the developer has written code, can the change be tested and integrated with the rest of the codebase in a continuous way, or do changes require sign off or approval from multiple teams? Teams practicing continuous integration can fix errors more quickly before pushing changes to production environments.

Continuous delivery and deployment: Is the code ready to deploy to production at any time? Engineers practicing continuous delivery and deployment of their changes are more productive and can see the value they add immediately in the application. The ability to see this value also attracts more highly skilled engineers, which amplifies high functioning cloud native teams even more.

Components

Here's a curated set of tools that establishes a cloud native software delivery pipeline. This combination of tools is proven to work effectively with VMware® Essential PKS, and these tools can also work well with VMware® Enterprise PKS.

Source code management: Git

Git is a free and open-source distributed version control system that has become extremely popular worldwide as the de facto standard for teams developing software. Git is separated from alternatives by its branching model. Most teams that use Git either consume repository hosting as a service through GitHub, GitLab, or others, or they host repositories on premises using similar software. Git code repositories can be used for application software, but in the last decade, application deployment tooling and even infrastructure is driven by code under version control.

Container registry: Harbor

One of the hidden costs of a container orchestration solution like Kubernetes is the need to access a myriad of container images as more applications are deployed into clusters. Harbor can help control that cost by providing an open-source solution that stores images in a private registry, signs them as trusted, and scans them for vulnerabilities. Harbor solves the common challenges inherent in hosting container images by delivering trust,

compliance, performance, and interoperability. It fills a gap for organizations that cannot use a public or cloud-based registry, or want a consistent experience across clouds. Harbor was open sourced by VMware and is an official project of the Cloud Native Computing Foundation.

Continuous development: Skaffold

Skaffold is an open-source command-line tool from Google that handles the workflow for building, pushing, and deploying Kubernetes applications. Skaffold facilitates continuous development for Kubernetes applications, allowing developers to iterate on application source code locally and then deploy to local or remote Kubernetes clusters. Skaffold allows for rapid feedback loops on a Kubernetes cluster, where developers can see how their changes immediately impact not only their own applications but the relationships to other applications that run in and outside Kubernetes. Finally, Skaffold also provides building blocks and describes customizations for a CI/CD pipeline.

Continuous integration and delivery: Jenkins

Jenkins is a self-contained, open-source automation server. It can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software. Jenkins is a highly extensible product whose functionality can be extended through the installation of plugins.

Continuous deployment: Spinnaker

Spinnaker is an open-source, multi-cloud platform for releasing software. Created at Netflix, it has been proven reliable in production by hundreds of teams over millions of deployments. It combines a powerful and flexible pipeline management system with integrations to the major cloud providers and Kubernetes. Spinnaker can handle multiple deployment patterns natively.

Software delivery pipelines

Below are two examples of delivery pipelines from end to end, starting with a development pipeline.

Development pipeline

The pipeline runs on each commit to a staging environment, where a developer can demo the change or potentially run integration testing with other services managed by other teams.

In this pipeline, development starts in a local environment, where it uses kind with "skaffold dev" in a quick feedback loop. Git then pushes the code to staging. In staging, Jenkins tests the code. After the Jenkins tests are complete, the Skaffold build command pushes the code to Harbor and the Skaffold run command deploys the code to a staging cluster.

Production pipeline

In the production pipeline, a release is cut and the new version is deployed to the production cluster. The process flows like this: After the release is cut, the released container is pushed to Harbor. Spinnaker polls Harbor for the new release image and asks for approval to deploy it. The approval step allows the release manager to manually approve changes before a deployment occurs. Spinnaker then deploys the new release to a production Kubernetes cluster.

A note on cluster organization

The strategy of many early adopters of Kubernetes was to use one cluster for everything; however, as Kubernetes has matured, several issues have been discovered with this

strategy. True multi-tenancy is still not a solved problem in Kubernetes, and organizations can have multiple teams or users competing for cluster resources. VMware recommends a multi-cluster strategy, whether divided among teams, applications, stage of an application (development vs. production), lines of business, or intersectional clusters depending on corporate needs. For the purpose of this document, a single team uses a different cluster for each stage of the application lifecycle.

Application development

Individual developers often develop a microservice locally without using a container or Kubernetes environment. The microservice components are not put into containers until the developer is ready to test in a container or commit the code and push the change to a Git repository.

Using Skaffold for development

Skaffold can be used when a developer is ready to test an application in a containerized environment. A skaffold.yaml file, which should reside in the same Git repository as the application code, provides instructions on how the containers should be built and how the application should be deployed. This process could follow Kubernetes manifests, Helm charts, or kustomize templates. Kubernetes manifests should also live with the application code, so the same team of developers can manage deployment code and application code together—and most importantly, keep it all version controlled.

Skaffold can deploy to a Kubernetes cluster running locally, which allows the developer to test code changes locally before pushing them through a build system to a remote cluster. Popular local cluster options include Minikube, kind, or Docker Desktop. Skaffold should tag images based on the Git commit ID so images can be pinned to version-controlled changes.

Commit, push, and pull request

Once developers are ready to commit their changes and push to a remote Git repository, they open a pull request to the master branch to merge their changes. A continuous integration server should be watching for pull requests and test the code for quality.

Application testing: Jenkins, Skaffold, and Harbor

In this case, Jenkins is watching a repository for pull requests and testing appropriately. The testing, which varies by language and application, may include syntax checking, unit tests, and integration tests. As container images are being used, Container Structure Tests should also be performed. In any case, the results of testing should be quickly available to developers and efforts to minimize testing time to maximize developer productivity should be encouraged.

Once Jenkins has tested the developer's code changes for quality assurance, the following steps occur in sequence:

1. A Harbor registry login occurs using docker login
2. A container image is built by Skaffold using the Jenkins host's Docker daemon
3. The resulting image is tagged and pushed to Harbor to be deployed to Kubernetes.

A note on Harbor best practices

When using Harbor for container images as part of a pipeline, you should follow several best practices. Please refer to the Harbor user and administrator guides for more information.

- A project in Harbor should be created to host the team's container images.
- Members of the team should have Guest access to the project that allows pull-access only, regardless of whether that is driven by LDAP or Active Directory groups or handled directly by Harbor's internal authentication.
- The CI/CD system (in this case Jenkins) should have Developer access to the project. This access level ensures that only automated systems and pipelines that are version controlled have access to push container images to the repository, outside of Harbor administrators.
- Kubernetes clusters should have Guest access, because the cluster only needs to pull container images, never push.
- HTTPS should be enabled in Harbor with a certificate that is signed by an accepted certificate authority. This configuration prevents Kubernetes cluster administrators from having to set -insecure-registry in the Kubernetes Docker daemon configuration and is more secure.

Application deployment

Deploying feature branches to staging environments

Once a container has been pushed as part of the same Jenkins pipeline, the pull request can be deployed to a staging Kubernetes cluster for further testing or demonstration purposes. Skaffold can be coupled with a Jenkins pipeline to deploy to the staging cluster. Scripted inter-application integration tests can be performed to confirm the code change is ready for release.

Deploying releases to production with Spinnaker

When the pull request has been completely tested and is deemed ready to merge and deploy to production, a release is cut. A container image should be built with a version tag. Spinnaker can poll an image registry to watch for this image and deploy it to a production cluster by using a built-in Kubernetes deployment pattern.

Kubernetes deployment patterns

Kubernetes has several patterns built into the system that development teams can use for deploying software. Two of the patterns are the blue-green pattern and the canary pattern. These patterns require orchestration through a deployment pipeline (driven by Jenkins) and the Harbor container registry to host container images. The Kubernetes Deployment resource has RollingUpdate capability to ensure smooth deployments of updated container images when teams release software.

Blue-green deployment

In a blue-green application deployment strategy in Kubernetes, two Kubernetes ReplicaSets are used with different labels. A Kubernetes Service with a selector is used to determine which pods get traffic, giving the administrator the ability to move back and forth as needed.

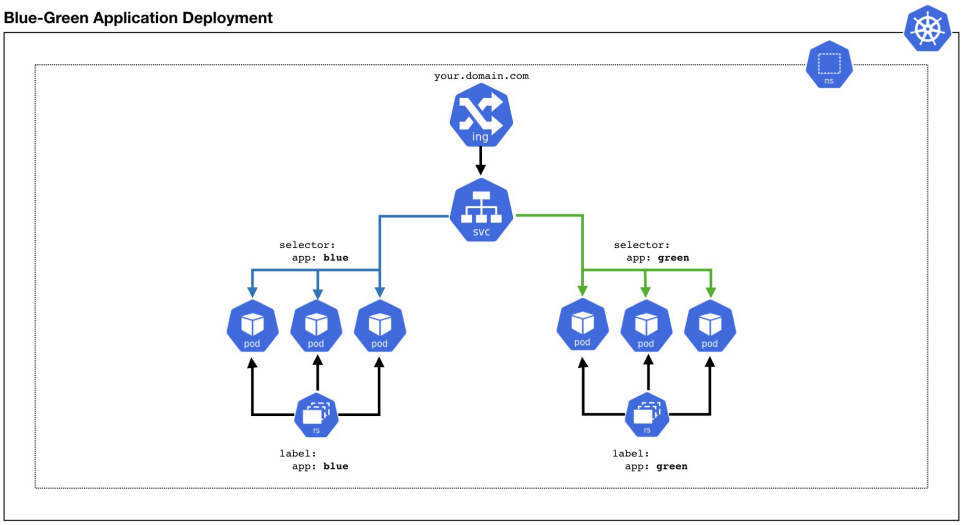

Figure 1: A blue-green application deployment strategy in Kubernetes.

Canary deployment

In a canary deployment strategy, two ReplicaSets are used in a way that's similar to a blue-green deployment, but the service selector is the same instead of being different, as Figure 2 shows.

By running a smaller ReplicaSet of the canary in the same Service pool, a percentage of traffic can be sent to the canary for testing. Once initial testing passes, the canary ReplicaSet can be scaled up to send a higher percentage of traffic to the canary. Once testing is complete, the existing stable ReplicaSet's pods can be replaced with the updated image in sequence, and the canary deployment can be deleted, as Figure 3 shows.

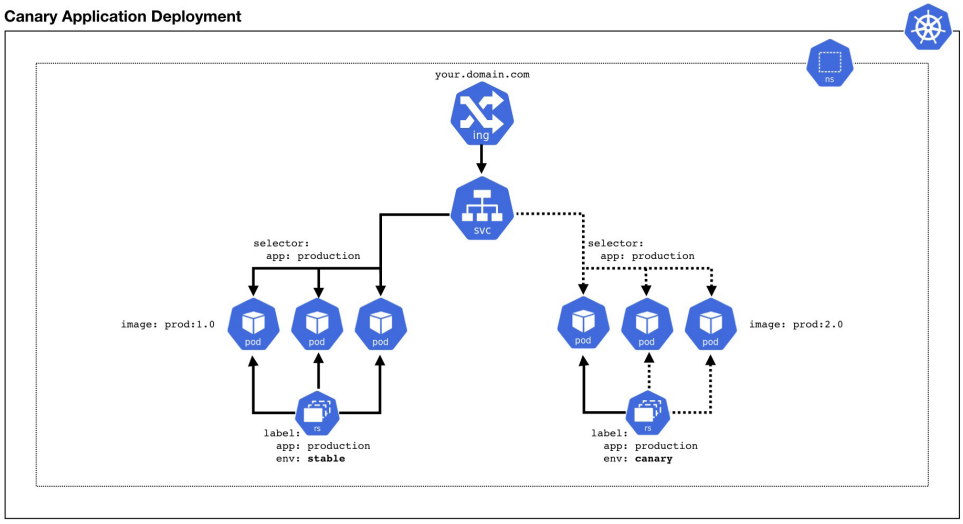

Figure 2: In a canary deployment strategy, two ReplicaSets with the same service selector are used.

AUTHOR BIO

Hart Hoover is a Cloud Native Architect at VMware who gets customers from zero to production with Kubernetes and other Cloud Native Computing Foundation projects. Hart is a former speaker at several OpenStack Summits, Cloud Expo, DevOpsDays, Cisco Live, Texas Linux Fest, and others. He is also a co-organizer of the San Antonio Kubernetes Meetup.

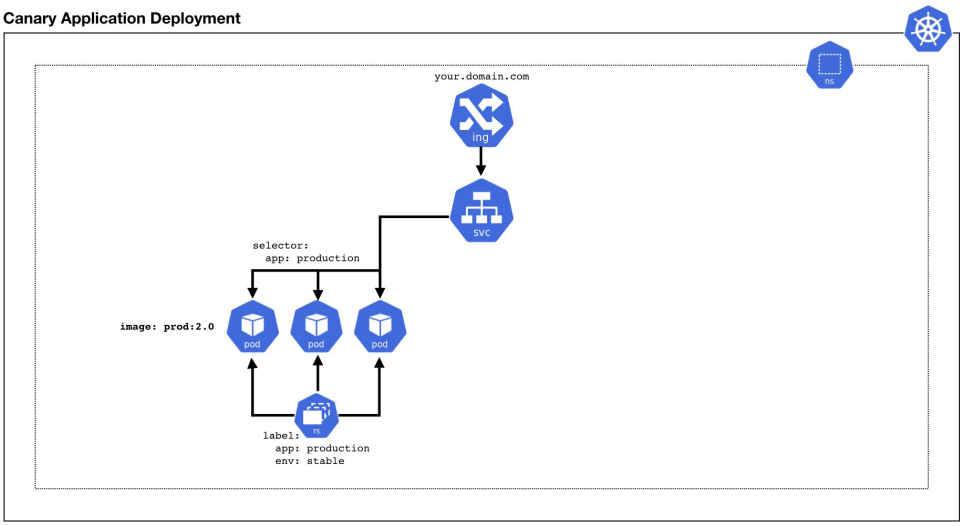

Figure 3: The canary deployment is deleted after testing completes.

Conclusion

By bringing together a curated set of cloud native tools, software development teams can be successful with Kubernetes. By establishing quick feedback loops in development, automating testing, and practicing continuous delivery, businesses can increase developer velocity. The Kubernetes API is the common element that ensures consistency across local development environments, on-premises staging environments, and public cloud production environments.

Resources

- A Real World Example: The Jenkins X SSO Operator
- Harbor
- Skaffold
- Jenkins Pipeline Examples
- Spinnaker

LEARN MORE ABOUT CLOUD NATIVE TECHNOLOGY FROM VMWARE

To learn about how VMware can help you deploy, manage, and secure cloud native applications, see *https://cloud.vmware.com*.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com