ActualTech Media

PLATFORM9

Production-Grade Kubernetes: Best Practices Checklist

CONTENTS

IN THIS PAPER

Kubernetes is a complex platform that provides for highly scalable, efficient use of computing and storage resources. But it can also be highly problematic for companies that just try to "wing it" and figure out what to do as they go along.

Don't let that be you. This paper details best practices you should follow to help ensure you realize the optimal benefit of your Kubernetes investment.

Highlights include:

- Ensuring open and flexible environments
- Standardizing the container build process
- Cluster observability with a single pane of glass

Kubernetes is rapidly becoming a key element of enterprise IT infrastructure. As with other enterprise platforms, there's a broad array of requirements to keep Kubernetes clusters functioning and running efficiently. Here are several best practices to employ when running Kubernetes in production.

Deployment Best Practices

Kubernetes environments are highly dynamic. Services are deployed and updated frequently. Nodes are added and removed from clusters. Clusters are spun up and down according to workload.

> If you already have an established VM environment, running Kubernetes in that environment can be a logical choice.

Kubernetes handles much of the management of this lifecycle, but when it comes to deploying services, much of the responsibility rests with IT professionals. To streamline the ability to deploy and maintain services, keep in mind deployment best practices—these ones in particular:

- Ensure open and flexible environments
- Standardize the container build process
- Ensure self-service
- Manage applications and storage

ENSURING OPEN AND FLEXIBLE ENVIRONMENTS

Kubernetes runs on a variety of computing infrastructure, including commodity servers. Existing hardware can be redeployed to run Kubernetes along with newly procured servers. You have your choice of running Kubernetes on bare metal, virtual machines (VMs), or in public clouds.

If you already have an established VM environment, running Kubernetes in that environment can be a logical choice. If you would rather not maintain physical infrastructure, then running Kubernetes in a public cloud is a good option and one with low barriers to entry. Also, Kubernetes has prompted the development of additional open source software that runs on the platform. Tools like Helm for deployment and Istio for service management are open source tools that extend the capabilities of the Kubernetes environment.

STANDARDIZE THE CONTAINER BUILD PROCESS

Containers are a key building block of a microservice architecture, and how they're managed directly impacts the efficiency, reliability, and availability of services running in Kubernetes clusters.

The container build process should be automated using a continuous integration/continuous deployment (CI/CD) system. Open source tools such as Jenkins are widely used CI/CD tools. Major public cloud providers also offer CI/CD services. These tools reduce the workload on developers when deploying a service. They also allow for automated testing prior to deployment, and can support rollback operations when needed.

Container images should be stored in an image repository. This centralized store should also support image scanning to check for security vulnerabilities. By providing an image repository, you can promote the consistent use of approved images. This reduces the chance of deploying a misconfigured container. Developers have a range of container registry options, including Docker Hub, JFrog Container Registry, and JFrog Artifactory.

For example, a Docker image may require that several tools be installed, and those tools may require different versions of the same library. Someone unaware of the potential conflict might fail to properly install the library's multiple versions. This could lead to deploying an image that will fail in production and require a team of DevOps engineers to diagnose in production.

A standardized image build process helps remediate failed deployments. For example, the CI/CD pipeline can be configured to perform a canary deployment, in which a small amount of traffic is routed to a newly deployed service. If there's a problem, only a small number of users are adversely affected.

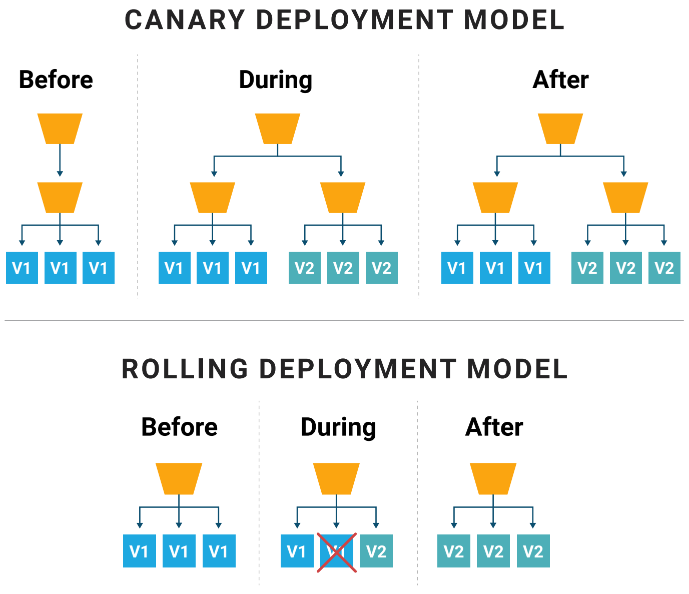

Figure 1: Deployment models like rolling and canary deployments help identify any issues with a new deployment before it causes any significant impact to production.

Alternatively, the CI/CD process could employ a rolling deployment in which pods are replaced one by one, allowing for an incremental transition to a new version of a service. Of course both canary and rolling deployments could be done manually, but that would be more time-consuming and error-prone (see **Figure 1**).

It's important to have visibility into the performance of services so you can correct issues that can't be addressed directly by Kubernetes.

As part of the image build process, be sure to include a monitoring mechanism. Some monitoring tools use agents to collect server and application performance data and send it to a centralized data store for reporting and alerting. It's important to have visibility into the performance of services so you can correct issues that can't be addressed directly by Kubernetes.

ENSURE SELF-SERVICE

Developers should not have to coordinate with IT administrators to deploy and monitor services running in Kubernetes clusters. To ensure developers can manage deployments, it's important to provide tools for deploying and scanning applications.

For example, Helm is a package manager for Kubernetes and supports defining, deploying, and upgrading applications, which can streamline the management of applications. Security scanning tools should be in place as well, to help developers identify vulnerabilities in applications before they're deployed.

APPLICATIONS AND STORAGE

With developers and admins alike working across multiple environments, it's also important to have policies in place to enable efficient use of resources. Consider role-based access controls (RBAC) policies and limits to ensure resources are used fairly, and that no single deployment consumes an excessive amount of resources.

Operations Best Practices

In addition to employing deployment best practices, there are several operations best practices you should strive to implement, including:

- Single pane of glass visibility
- Scaling best practices
- Governance and security
- Upgrading

Together, these best practices can help reduce the operational overhead associated with maintaining Kubernetes clusters.

CLUSTER OBSERVABILITY

The idea behind single pane of glass visibility is that all information needed to understand and diagnose the current state of the cluster, deployments, and other components should be available from a single tool.

For example, from a single application, administrators should be able to configure monitoring, analyze monitoring data, and specify alerts triggered by that monitoring data. Plan to use a standardized set of monitoring tools for collecting, storing, analyzing, and visualizing performance monitoring data.

This monitoring functionality can also be used to monitor compliance with SLAs. Another advantage of standardizing is that you can define templates to promote reusability.

SCALING BEST PRACTICES

When scaling with Kubernetes, you have the option of scaling the size of a cluster or increasing the number of clusters. When workloads vary widely, the Horizontal Pod Autoscaler can be used to adjust the number of nodes in a deployment. Kubernetes also has a Vertical Pod Autoscaler, but that's currently in beta release and shouldn't be used in production.

> Kubernetes can scale to thousands of nodes and hundreds of thousands of pods, so a single cluster can meet many use cases.

One scaling question you'll face is whether to run one cluster or multiple clusters. Kubernetes can scale to thousands of nodes and hundreds of thousands of pods, so a single cluster can meet many use cases.

There are, however, some advantages of using multiple clusters. One is reliability. In the event of a cluster failure, all workloads are affected in a single cluster environment. Also, with multiple clusters different development teams can manage their own clusters—and that can increase the velocity of each team, if it doesn't need to coordinate cluster changes with other teams.

GOVERNANCE AND SECURITY

As with any enterprise platform, you'll need to consider governance and security. To start with, plan to implement granular RBAC. These can be used to implement the principle of "least privilege," which states that users should have only permissions they need to perform tasks assigned to them and no more.

Use audit trails to track security-related changes in the system. For example, operations, like adding a user and changing permissions, should be logged. Also, use encryption to secure communications both within and outside of the cluster.

It's a best practice to scan applications for vulnerabilities. In a similar way, you should review vulnerabilities in Kubernetes software and patch as necessary. This is a situation in which it may be beneficial to have multiple clusters, because a patch can be deployed to a single cluster and evaluated before rolling it out to others.

UPGRADING

Kubernetes is under active development, which provides for new functionality and improved reliability. As part of your Kubernetes management strategy, plan to upgrade while supporting production workloads. For example, the master will need to be upgraded before nodes. To avoid disruption, you can run multiple master nodes and upgrade one master at a time or use rolling upgrades to ensure zero downtime during the upgrade process.

> Kubernetes is under active development, which provides for new functionality and improved reliability.

Similarly, nodes can be upgraded incrementally. Also plan to patch and upgrade operating systems running on nodes. Be sure to maintain up-to-date backups. Backups are an important insurance measure for recovering from a failed upgrade.

They're 'Best' Practices for a Reason

Kubernetes is a complex platform that provides for highly scalable, efficient use of computing and storage resources. But it can also be highly problematic for companies that just try to "wing it" and figure out what to do as they go along.

Don't let that be you. Following the best practices outlined here will help to ensure that you realize the optimal benefit of your Kubernetes investment.