SUSE

Kubernetes: 10 Tips to Overcome the Overwhelming

SUSE

Introduction

Containers solve problems around application portability and ensure the consistency of any underlying dependencies, like software libraries. Since the application code and libraries are all contained within the container, they are easily portable. However, containers cannot run on their own — they need a system to manage deployment and keep things operational — in other words, an orchestration system. Kubernetes has become the orchestration system of choice for many organizations with its high availability, networking and security functions. While this functionality is very powerful, it can be overwhelming for newer users or even experienced DevOps personnel who lack advanced system administrator experience.

In this e-book, we've compiled 10 tips (plus a bonus tip) to help you and your organization make your Kubernetes deployments more successful.

SUSE

TIP 01:

Use the Right Tool for Local Kubernetes Development

One of the major benefits of using Kubernetes, and containers generally, is that the same application runtime and code that is used on the developer's workstation is deployed into production, a concept called workload portability. However, the requirements around high availability, networking and storage that are required for a production implementation are not required for running Kubernetes locally.

Developers have a few choices. There are a host of open source solutions like Rancher Desktop, K3s, Minikube and Docker Desktop that are used to emulate the full-scale Kubernetes API and support your development processes. Comparing these solutions, Minikube supports a lot of options around container runtime and node management, but at a higher resource cost. Docker Desktop and Rancher Desktop are comparable services that can both run Kubernetes, but with one major caveat: If your organization has more than 250 employees or more than $10 million in revenue, you need to license Docker Desktop for all of your developers. Rancher Desktop allows you to have a fully fledged Kubernetes experience without having to manage licensing for your development teams.

There are a host of open source solutions like Rancher Desktop, K3s, Minikube and Docker Desktop that are used to emulate the full-scale Kubernetes API and support your development processes.

SUSE

TIP 02:

Set Pods' Resource Requests, Limits and Health Checks

Proper allocation and control of available resources are essential to ensuring the maximum performance of any Kubernetes pod as well as maintaining the overall health and concurrency of your cluster as your projects grow. Resource requests and limits are used to set upper and lower boundaries for resources like CPU and memory for a given container within a pod. These limits prevent resource starvation while the request indicates what is minimally required for the container to function effectively.

```
resources:
        limits:
cpu: 4000m
memory: 2Gi
        requests:
cpu: 250m
        memory: 1Gi
```

Health probes are used by Kubernetes to continuously monitor the health of the containers to ensure maximum efficiency. A readiness probe is used to indicate when a container is ready to allow incoming traffic after starting. A liveness probe can monitor the overall state of the container; if a container is down, the cluster will attempt to restart it. The startup probes are used to indicate when a container application has been started. If a startup probe is configured, the liveness and readiness probes are disabled until it is successful. This ensures these two probes do not interfere with the startup of the application.

```
readinessProbe:
        tcpSocket:
                port: 8080
        initialDelaySeconds: 5
        periodSeconds: 10

livenessProbe:
        tcpSocket:
                port: 8080
        initialDelaySeconds: 15
        periodSeconds: 10
```

- # of seconds after container start
- how often to perform the probe
- # of seconds after container start
- how often to perform the probe

SUSE

TIP 03:

Use Horizontal Pod Autoscaling to Scale Pods

One of the benefits of cloud native computing platforms like Kubernetes is that it's easy for your application to both scale in and out based on workload metrics. This means that whether you are in the cloud or running locally, you can automate the scaling of pods. Before containers or Kubernetes, it was a lot harder to scale an application horizontally or load balance network traffic. When you encapsulate that same Web application into a Kubernetes pod, you can use workload metrics by deploying the metric-server to enable CPU and memory based autoscaling. In order to configure your deployments to enable scaling both in and out, consider deploying the code sample at the right to your cluster.

In this deployment file, you can see a php-apache web server pod that scales up to 10 replicas. It will scale up when average CPU utilization reaches 50 percent (across all of the currently existing pods). You can also use more advanced metrics and control the quality of service.

Code sample: Autoscaling

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
              apiVersion: apps/v1
              kind: Deployment
              name: php-apache
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
              resource:
              name: cpu
              target:
              type: Utilization
              averageUtilization: 50
status:
  observedGeneration: 1
  lastScaleTime: <some-time>
  currentReplicas: 1
  desiredReplicas: 1
  currentMetrics:
  - type: Resource
              resource:
              name: cpu
              current:
              averageUtilization: 0
              averageValue: 0
```

SUSE

TIP 04:

Use an Ingress Controller

When it comes to getting external network traffic to and from your application, services are actually only sufficient to a certain degree.

1. For starters, they only operate at Layer 4 in the OSI network model, so they only forward TCP and UDP traffic. That's a problem if you want to forward HTTP traffic that operates at Layer 7 of the network OSI model.
2. The second issue is that every time you use the NodePort service to expose traffic to an application, you're exposing a unique port on your node which is insecure. If you opt for the LoadBalancer service, this can quickly become expensive because a load balancer will be created by the cloud provider for every application.

The solution to these problems is to make use of the Kubernetes Ingress object. The Kubernetes Ingress object will create a single external load balancer listening for HTTP traffic and will then route the traffic to the relevant service in your cluster, based on the rules that you define.

Now, Kubernetes objects cannot be created and deployed in isolation. You first have to deploy a Kubernetes ingress controller. In many cases, Kubernetes installers and distributions come with them already installed. Ingress controllers act on the Ingress objects you create and essentially manage them in a similar way that deployments manage pods. Like the rest of Kubernetes, you can define your ingress rules in YAML as shown below.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-wildcard-host
spec:
  rules:
  - host: “foo.bar.com”
               http:
               paths:
               - pathType: Prefix
               path: “/bar”
               backend:
               service:
               name: service1
               port:
               number: 80
  - host: “*.foo.com”
               http:
               paths:
               - pathType: Prefix
               path: “/foo”
               backend:
               service:
               name: service2
               port:
               number: 80
```

SUSE

TIP 05:

Use Kubernetes Secrets Optimally

Security should always be top of mind, with ransomware attacks becoming a ubiquitous aspect of software utilization. Beyond managing your container supply chain, you also need to think about secrets management. Kubernetes secrets, while a native object that is base 64-encoded and not encrypted by default, are stored within the API servers' data store, and contain sensitive information like passwords and API keys. Maintaining the secrets can be painful and cumbersome, which in turn allows for mistaken exposure of the secret itself. Any individual with the proper API access can obtain and alter a secret natively within Kubernetes. It is crucial to pay attention to secrets and ensure their continued safety.

Because Kubernetes secrets are insecure, you need to use an external secrets manager. Some of the most popular secrets managers include:

- Bitnami Sealed Secrets takes your Kubernetes Secret and injects them into a SealedSecret, which can only be decrypted by the controller that is running the target cluster. This is done via Sealed Secret controllers deployed to a Kubernetes cluster. These secrets can only be decrypted by the controller running in the target cluster, and not other users or services, or even the author of the secret.
- Mozilla SOPS facilitates a similar experience as Bitnami's Sealed Secrets in that it can encrypt the secret to allow it to be checked into a repository alongside the code itself. Able to integrate with common cloud key vault providers (Azure, AWS or Google Cloud), the deployed encryption key remains locked safely in the cloud.
- Helm Secrets, built on Mozilla's SOPS framework, ensures that sensitive data is not leaked when encrypted properly.

The best secret is the one that cannot be seen.

Because Kubernetes secrets are insecure, you need to use an external secrets manager. The best secret is the one that cannot be seen.

SUSE

TIP 06:

Use Helm for Package Management of YAML Resources

Everyone's favorite part of managing Kubernetes is managing the vast array of YAML files used for deployments—except, not at all. YAML can be a headache. Managing a small Kubernetes cluster is manageable, but at a much larger scale opens the door to YAML sprawl. YAML manifest files are the deployment mechanism for pods, services and deployments, all of which are vital to the orchestration of Kubernetes, but they can spread uncontrollably. Helm is a package manager that gives IT professionals the ability to efficiently configure, package and deploy applications and associated services into Kubernetes clusters.

Helm deploys through use charts. A chart is just a collection of YAML files that chronicle similar Kubernetes resources. An individual chart could deploy a simple pod or something more complex like the entire application stack from database servers to Web servers to network components and storage.

You can read more about best practices for Helm here.

SUSE

TIP 07:

Grant Authorization with RBAC and ABAC

One of the ways applications have traditionally managed security was by providing service accounts. In the Windows ecosystem, these service accounts lived in Active Directory and used Kerberos for authentication. Kubernetes also has the concept of service accounts which can be used to identify a given pod, eliminating the need for passing a secret to perform tasks in the cluster. This allows pods to be assigned role-based access control privileges in the cluster, and perform specific operations on cluster objects. Beyond RBAC, you can use attribute-based access control (ABAC) to grant permissions based on policies that combine attributes together.

```
{“apiVersion”: “abac.authorization.
kubernetes.io/v1beta1”, “kind”: “Policy”,
“spec”: {“user”: “bob”, “namespace”:
“projectCaribou”, “resource”: “pods”,
“readonly”: true}}
```

In the above code example, user Bob can perform read operations on all of the pods in the namespace **ProjectCaribou**.

SUSE

TIP 08:

Use a Cluster Management Platform

Many very good open source projects have been hampered by a lack of a clean management tool stack. While Hadoop had Zookeeper, it was still a challenge for organizations to manage, limiting adoption. With the continued expansion of Kubernetes as the container orchestration platform of choice, many organizations face challenges managing their clusters and using a consistent Kubernetes policy approach. The key, however, is the continuous management of the environment. When a single cloud or management provider is chosen, the lateral movement of the clusters is more difficult and cumbersome.

Rancher solves that problem with a single control plane that provides the tools and functionality for teams to advance their container ecosystem. Across all types of providers (cloud, on-prem, cloud native), Rancher offers simplistic means to manage provisioning, monitoring and version management for your Kubernetes clusters. Rancher helps you automate cluster management tasks and ensures consistent policies for user access and security within your cluster. While Kubernetes manages your pods and containers, Rancher manages Kubernetes to help make your life easier. Rancher also provides apps and marketplace features that allow for easy installation of other cloud native tools in your cluster.

SUSE

TIP 09:

Shift Security Left: Secure Your K8s Supply Chain

After the Solarwinds hack, supply chain security became even more important than ever. The recent security bugs in the Log4J package have further highlighted the need for software developers to understand the components that make up their applications. Your software build process produces both artifacts and metadata. To verify the integrity of your builds, you also need to have attestations and policies.

Artifacts include container images, packages or WAR files, while metadata is the information that describes those artifacts, including provenance (what machine built the code, CI/CD tools info and all of those details), the software bill of materials (which packages and libraries were used in the code) and vulnerability scanning as part of your build process. It is important to frequently rerun vulnerability scans against your builds to check for updates to vulnerabilities. Attestation at a basic level is signing your code and images with a certificate to ensure they are trusted. Finally, you can use policies within your cluster to require image verification—including validation against provenance data, vulnerability scans and your bill of materials.

Your software build process produces both artifacts and metadata. To verify the integrity of your builds, you also need to have attestations and policies.

SUSE

TIP 10:

Deploy a Monitoring Stack

While containers offer the ability to easily add more components like caches into your application stack, having an application stack that is more broadly distributed means more challenges in terms of monitoring the performance of your applications and troubleshooting error conditions. Prometheus is a popular open source monitoring framework that stores metrics as time-series data (a key-value pair of the metric and the timestamp where it was collected). Prometheus has out-of-the-box capabilities for monitoring Kubernetes natively and can perform service discovery for services running on your clusters once deployed. While Prometheus has its own dashboard, many organizations use the open source dashboarding tool, Grafana, to provide visualization and insights into their data.

Beyond these tools, each of the cloud providers has application monitoring stacks that you can include in your application code to track performance across your application.

SUSE

BONUS TIP:

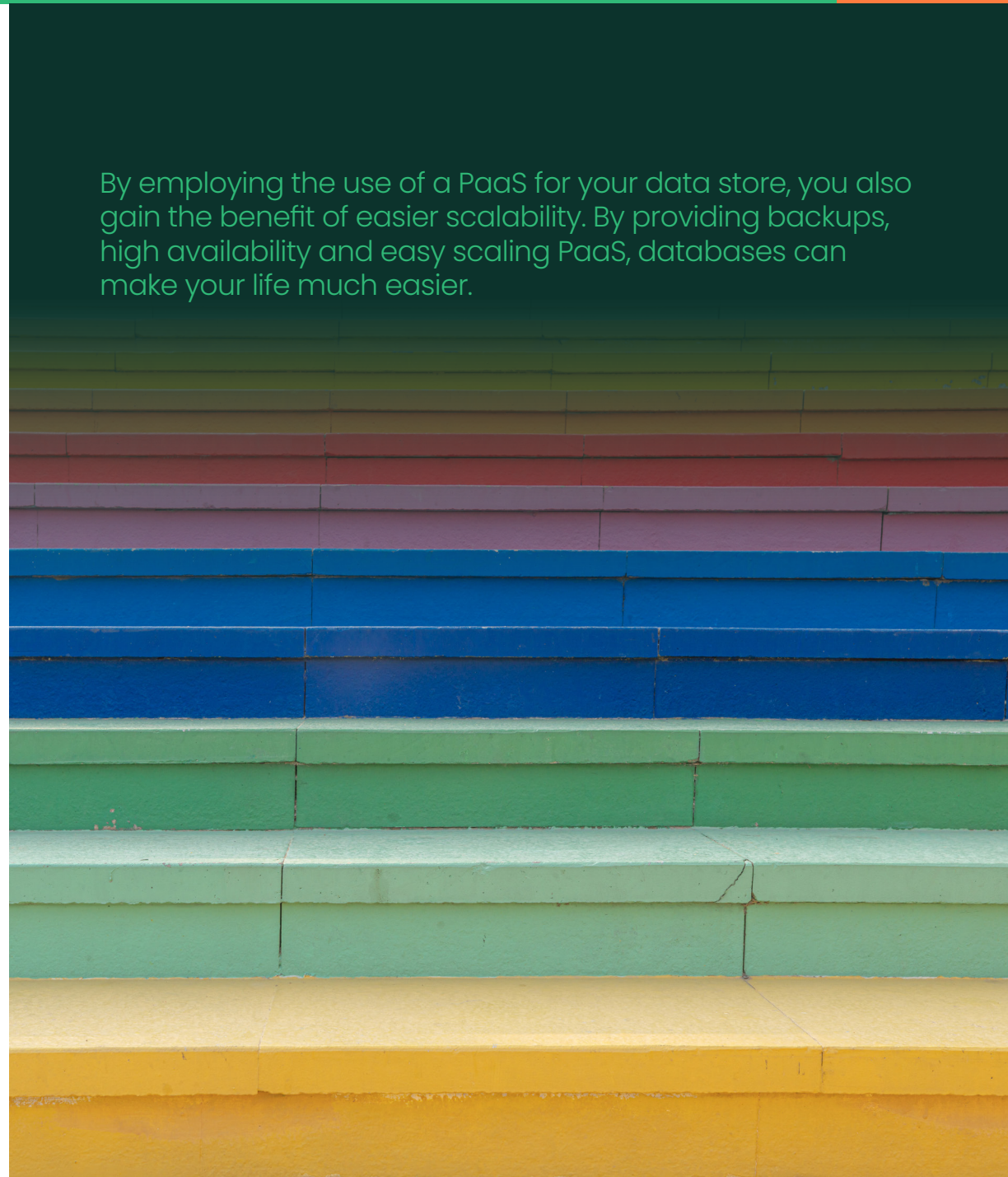
Use a Cloud-Managed Data Store

Kubernetes was architected to provide hosting for stateless applications and services. While it can host stateful applications, such as MySQL or other database platforms, it's a more complex solution than hosting a simple Web server. When you are looking at deploying containerized databases, you need to examine and evaluate whether you want to manage and support the complexities of the persistent data store within a container.

The public cloud offers an easy solution. All of the major clouds support platform as a server (PaaS) database offerings. The cloud provider can facilitate and ensure that critical aspects of database management — such as high availability, OS and RDBMS patching, database backups and recovery options, firewall security, as well as several others — are managed for you.

Any ecosystem is sensitive to the ups and downs of continuous utilization and growth. By employing the use of a PaaS for your data store, you also gain the benefit of easier scalability. By providing backups, high availability and easy scaling, PaaS databases can make your life much easier.

By employing the use of a PaaS for your data store, you also gain the benefit of easier scalability. By providing backups, high availability and easy scaling PaaS, databases can make your life much easier.



SUSE

Conclusion

With its proven security, networking and high-availability capabilities, Kubernetes is a popular — and powerful – container orchestration system. However, it can be overwhelming to manage, particularly for inexperienced DevOps teams.

Becoming familiar with these tips will help you better manage your Kubernetes resources and their performance. You will learn how to better scale their workloads and load balance them using ingress controllers. These tips will also help you gain a more complete understanding of your build process — an important asset given the recent prevalence of supply chain vulnerabilities and instances of secrets mismanagement.

We want to help you improve your productivity and security across all of your workloads. As a complete container management platform for Kubernetes, Rancher gives you the tools you need to successfully run Kubernetes anywhere.

Learn more at suse.com/rancher

SUSE is a global leader in innovative, reliable, secure enterprise-grade open source solutions, relied upon by more than 60% of the Fortune 500 to power their mission-critical workloads. We specialize in Business-critical Linux, Enterprise Container Management and Edge solutions, and collaborate with partners and communities to empower our customers to innovate everywhere – from the data center, to the cloud, to the edge and beyond.

SUSE puts the "open" back in open source, giving customers the agility to tackle innovation challenges today and the freedom to evolve their strategy and solutions tomorrow. The company employs more than 2,000 people globally. SUSE is listed on the Frankfurt Stock Exchange.

For more information, contact SUSE at:

+1 800 796 3700 (U.S./Canada)

Maxfeldstrasse 5

90409 Nuremburg

www.suse.com