Creating a Kubernetes cluster in 16 steps

A Kubernetes cluster is a set of node machines—including a control plane and one or more compute machines—for running containerized applications. While Kubernetes is a powerful tool for building highly scalable systems, the technology is complex and setup can prove difficult. The following 16 steps can help you configure your containers and Kubernetes clusters for production traffic—and reduce the risk of misconfiguration.

1 Use minimal base images

Containers are application stacks built into a system image. Everything from your business logic to the kernel gets packed inside. Minimal images strip out as much of the operating system (OS) as possible and force you to add back any components you need.

By including only the software you intend to use in your container, you boost security and performance.

2 Use a high-availability registry

Registries are repositories for images, making those images available for download and launch. Since your cluster will rely on your registry to launch newer versions of your software, any downtime will prevent updates to Kubernetes services or microservices.

Red Hat® OpenShift® provides a built-in container image registry that runs as a standard workload on the cluster. Most public cloud providers, like Alibaba, AWS, Google, IBM, and Microsoft, also offer private image registry services.

3 Use ImagePullSecrets to authenticate your registry

ImagePullSecrets are Kubernetes objects that let your cluster authenticate with your registry, so the registry can be selective about who is able to download your images.

If your registry is exposed enough for your cluster to pull images from it, then it needs authentication.

4 Isolate environments with namespaces

Namespaces are the most basic and powerful grouping mechanism in Kubernetes. They work almost like virtual machine (VM) clusters. Most objects in Kubernetes are limited to affecting a single namespace at once.

Namespaces provide strong isolation and are perfect for isolating environments with different purposes.

5 Organize your clusters with labels

Labels are the most basic and extensible way to organize your Kubernetes cluster. They allow you to create arbitrary key:value pairs that separate your Kubernetes objects.

Kubernetes also uses labels for selection. Since they represent such an open-ended type of organization, do your best to keep things simple, and only create labels where you require the power of selection.

6 Use annotations to track important system changes

Annotations are arbitrary key-value metadata you can attach to your pods, much like labels. However, Kubernetes does not read or handle annotations, so the rules about what you can and cannot annotate a pod with are fairly liberal, and they cannot be used for selection.

Annotations in Kubernetes are a fairly powerless construct, but they can be an asset to your developers and operations teams when used to track important system changes.

7 Implement access control using Kubernetes RBAC

Role-based access control (RBAC) allows you to control who can view or modify different aspects of your cluster. You can use RBAC to launch a Kubernetes application programming interface (API) server.

8 Prevent risky behavior and configurations using OPA Gatekeeper

Gatekeeper provides a Kubernetes admission controller built around the Open Policy Agent (OPA) engine to integrate OPA and the Kubernetes API service. OPA is useful for Kubernetes cluster security compliance and for practical resource configuration management. The Gatekeeper controller constantly monitors existing cluster objects to detect policy violations. Providing enforceable standard policies across ecosystems ensures consistent security and compliance.

9 Implement network control and firewalls using network policies

Network policies are objects that allow you to explicitly state which traffic is permitted. With these policies in place, Kubernetes will block all other nonconforming traffic. By default, Kubernetes allows open communication between all services. Leaving this “default open” configuration in place can put sensitive information at risk.

10 Use Kubernetes Secrets to store and manage sensitive information

Secrets are used to store sensitive data in Kubernetes, including passwords, certificates, and tokens. Your services may need to authenticate one another, other third-party services, or your users. Avoid loading secrets as environment variables, and instead, mount secrets into read-only volumes in your container.

11 Use an image scanner to identify and remediate image vulnerabilities

Scanners inspect the components installed in your images, including everything from the OS to your application stack. Scanners are useful for finding vulnerabilities in the versions of software that your image contains. You need to know where vulnerabilities reside in your system so you know what images may need updating.

12 Follow CI/CD methodologies

Continuous integration and continuous delivery (CI/CD) is the belief that every modification committed to your codebase should add incremental value and be production-ready. Following CI/CD helps your engineering team keep quality top of mind. If a functionality breaks, fixing it becomes an immediate priority for the whole team because every subsequent change that relies on the broken commit will also be broken.

13 Use canary methodologies for rolling out updates

A canary deployment is a way of bringing service changes from a commit in your codebase to your users. You bring up a new instance running your latest version, and you migrate your users to the new instance slowly, gaining confidence in your updates over time, instead of swapping over all at once. Using canary methodologies limits your users’ exposure to issues.

14 Implement monitoring and integrate it with SIEM

Monitoring tracks and records what your services are doing. There are two steps to successfully monitor a service—the code needs to be instrumented, and the output of that instrumentation needs to be fed somewhere for storage, retrieval, and analysis. How you perform instrumentation is largely dependent on your toolchain. For storage, consider a managed security information and event management (SIEM) technology.

15 Manage inter-service communication using a service mesh

A service mesh is a way to manage your inter-service communications, effectively creating a virtual network that you use when implementing your services. Using a service mesh can alleviate some of the more tedious aspects of managing a cluster, such as ensuring that communications are properly encrypted.

16 Use admission controllers to unlock advanced features in Kubernetes

Admission controllers manage what is going into your cluster. They allow you to set up webhooks that Kubernetes will consult during bring up. There are two types of admission controllers: mutating and validating. Mutating admission controllers alter the configuration files of the deployment before it is launched. Validating admission controllers get permission from your webhooks that a given deployment is allowed to be launched.

Get started

Security platforms built to protect Kubernetes offer powerful security and operational advantages. Kubernetes-native security applies controls at the Kubernetes layer, ensuring consistency, automation, and scale. Organizations successfully deploy security as code with security that is built in, not bolted on.

Read the full whitepaper to learn more about each of these steps—and get tutorials on how to implement them.

About Red Hat

Red Hat helps customers standardize across environments, develop cloud-native applications, and integrate, automate, secure, and manage complex environments with award-winning support, training, and consulting services.

NORTH AMERICA	EUROPE, MIDDLE EAST, AND AFRICA	ASIA PACIFIC	LATIN AMERICA
1 888 REDHAT1	00800 7334 2835	+65 6490 4200	+54 11 4329 7300
www.redhat.com	europe@redhat.com	apac@redhat.com	info-latam@redhat.com

facebook.com/redhatinc
@redhat
linkedin.com/company/red-hat

redhat.com
O-F29599

Copyright © 2021 Red Hat, Inc. Red Hat, the Red Hat logo, and OpenShift are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.