Kubernetes Blog

PodSecurityPolicy Deprecation: Past, Present, and Future

Tuesday, April 06, 2021

Author: Tabitha Sable (Kubernetes SIG Security)

PodSecurityPolicy (PSP) is being deprecated in Kubernetes 1.21, to be released later this week. This starts the countdown to its removal, but doesn’t change anything else. PodSecurityPolicy will continue to be fully functional for several more releases before being removed completely. In the meantime, we are developing a replacement for PSP that covers key use cases more easily and sustainably.

What are Pod Security Policies? Why did we need them? Why are they going away, and what’s next? How does this affect you? These key questions come to mind as we prepare to say goodbye to PSP, so let’s walk through them together. We’ll start with an overview of how features get removed from Kubernetes.

What does deprecation mean in Kubernetes?

Whenever a Kubernetes feature is set to go away, our deprecation policy is our guide. First the feature is marked as deprecated, then after enough time has passed, it can finally be removed.

Kubernetes 1.21 starts the deprecation process for PodSecurityPolicy. As with all feature deprecations, PodSecurityPolicy will continue to be fully functional for several more releases. The current plan is to remove PSP from Kubernetes in the 1.25 release.

Until then, PSP is still PSP. There will be at least a year during which the newest Kubernetes releases will still support PSP, and nearly two years until PSP will pass fully out of all supported Kubernetes versions.

What is PodSecurityPolicy?

PodSecurityPolicy is a built-in admission controller that allows a cluster administrator to control security-sensitive aspects of the Pod specification.

First, one or more PodSecurityPolicy resources are created in a cluster to define the requirements Pods must meet. Then, RBAC rules are created to control which PodSecurityPolicy applies to a given pod. If a pod meets the requirements of its PSP, it will be admitted to the cluster as usual. In some cases, PSP can also modify Pod fields, effectively creating new defaults for those fields. If a Pod does not meet the PSP requirements, it is rejected, and cannot run.

One more important thing to know about PodSecurityPolicy: it’s not the same as PodSecurityContext.

A part of the Pod specification, PodSecurityContext (and its per-container counterpart `SecurityContext`) is the collection of fields that specify many of the security-relevant settings for a Pod. The security context dictates to the kubelet and container runtime how the Pod should actually be run. In contrast, the PodSecurityPolicy only constrains (or defaults) the values that may be set on the security context.

The deprecation of PSP does not affect PodSecurityContext in any way.

Why did we need PodSecurityPolicy?

In Kubernetes, we define resources such as Deployments, StatefulSets, and Services that represent the building blocks of software applications. The various controllers inside a Kubernetes cluster react to these resources, creating further Kubernetes resources or configuring some software or hardware to accomplish our goals.

In most Kubernetes clusters, RBAC (Role-Based Access Control) rules control access to these resources. `list` , `get` , `create` , `edit` , and `delete` are the sorts of API operations that RBAC cares about, but *RBAC does not consider what settings are being put into the resources it controls*. For example, a Pod can be almost anything from a simple webserver to a privileged command prompt offering full access to the underlying server node and all the data. It's all the same to RBAC: a Pod is a Pod is a Pod.

To control what sorts of settings are allowed in the resources defined in your cluster, you need Admission Control in addition to RBAC. Since Kubernetes 1.3, PodSecurityPolicy has been the built-in way to do that for security-related Pod fields. Using PodSecurityPolicy, you can prevent "create Pod" from automatically meaning "root on every cluster node," without needing to deploy additional external admission controllers.

Why is PodSecurityPolicy going away?

In the years since PodSecurityPolicy was first introduced, we have realized that PSP has some serious usability problems that can't be addressed without making breaking changes.

The way PSPs are applied to Pods has proven confusing to nearly everyone that has attempted to use them. It is easy to accidentally grant broader permissions than intended, and difficult to inspect which PSP(s) apply in a given situation. The "changing Pod defaults" feature can be handy, but is only supported for certain Pod settings and it's not obvious when they will or will not apply to your Pod. Without a "dry run" or audit mode, it's impractical to retrofit PSP to existing clusters safely, and it's impossible for PSP to ever be enabled by default.

For more information about these and other PSP difficulties, check out SIG Auth's KubeCon NA 2019 Maintainer Track session video:

Today, you're not limited only to deploying PSP or writing your own custom admission controller. Several external admission controllers are available that incorporate lessons learned from PSP to provide a better user experience. K-Rail, Kyverno, and OPA/Gatekeeper are all well-known, and each has its fans.

Although there are other good options available now, we believe there is still value in having a built-in admission controller available as a choice for users. With this in mind, we turn toward building what's next, inspired by the lessons learned from PSP.

What's next?

Kubernetes SIG Security, SIG Auth, and a diverse collection of other community members have been working together for months to ensure that what's coming next is going to be awesome. We have developed a Kubernetes Enhancement Proposal (KEP 2579) and a prototype for a new feature, currently being called by the temporary name "PSP Replacement Policy." We are targeting an Alpha release in Kubernetes 1.22.

PSP Replacement Policy starts with the realization that since there is a robust ecosystem of external admission controllers already available, PSP's replacement doesn't need to be all things to all people. Simplicity of deployment and adoption is the key advantage a built-in admission controller has compared to an external webhook, so we have focused on how to best utilize that advantage.

PSP Replacement Policy is designed to be as simple as practically possible while providing enough flexibility to really be useful in production at scale. It has soft rollout features to enable retrofitting it to existing clusters, and is configurable enough that it can eventually be active by default. It can be deactivated partially or entirely, to coexist with external admission controllers for advanced use cases.

What does this mean for you?

What this all means for you depends on your current PSP situation. If you're already using PSP, there's plenty of time to plan your next move. Please review the PSP Replacement Policy KEP and think about how well it will suit your use case.

If you're making extensive use of the flexibility of PSP with numerous PSPs and complex binding rules, you will likely find the simplicity of PSP Replacement Policy too limiting. Use the next year to evaluate the other admission controller choices in the ecosystem. There are resources available to ease this transition, such as the Gatekeeper Policy Library.

If your use of PSP is relatively simple, with a few policies and straightforward binding to service accounts in each namespace, you will likely find PSP Replacement Policy to be a good match for your needs. Evaluate your PSPs compared to the Kubernetes Pod Security Standards to get a feel for where you'll be able to use the Restricted, Baseline, and Privileged policies. Please follow along with or contribute to the KEP and subsequent development, and try out the Alpha release of PSP Replacement Policy when it becomes available.

If you're just beginning your PSP journey, you will save time and effort by keeping it simple. You can approximate the functionality of PSP Replacement Policy today by using the Pod Security Standards' PSPs. If you set the cluster default by binding a Baseline or Restricted policy to the `system:serviceaccounts` group, and then make a more-permissive policy available as needed in certain Namespaces using ServiceAccount bindings, you will avoid many of the PSP pitfalls and have an easy migration to PSP Replacement Policy. If your needs are much more complex than this, your effort is probably better spent adopting one of the more fully-featured external admission controllers mentioned above.

We're dedicated to making Kubernetes the best container orchestration tool we can, and sometimes that means we need to remove longstanding features to make space for better things to come. When that happens, the Kubernetes deprecation policy ensures you have plenty of time to plan your next move. In the case of PodSecurityPolicy, several options are available to suit a range of needs and use cases. Start planning ahead now for PSP's eventual removal, and please consider contributing to its replacement! Happy securing!

Acknowledgment: It takes a wonderful group to make wonderful software. Thanks are due to everyone who has contributed to the PSP replacement effort, especially (in alphabetical order) Tim Allclair, Ian Coldwater, and Jordan Liggitt. It's been a joy to work with y'all on this.

← Previous Next →