Comparison with Docker Swarm

This lesson is a comparison between Kubernetes Secrets and the Docker Swarm equivalent.

WE'LL COVER THE FOLLOWING

The Similarities

Secrets are very similar to Kubernetes ConfigMaps and Docker Swarm configs. Everything we said for configurations applies to Secrets, with a few additional features.

Both Kubernetes and Docker Swarm stores Secrets in *tmpfs* inside containers. From that aspect, they are equally secure.

The Differences

> The significant difference is in the way Secrets are stored internally.

Kubernetes stores Secrets in etcd. By default, they are exposed, and we need to take extra precautions to protect them.

Docker Swarm secrets are, on the other hand, more secure by default. They are synchronized between managers using SSL/TLS, and they are encrypted at rest. We prefer “secured by default” approach behind Docker Swarm secrets.

In Kubernetes, we need to take extra steps to reach a similar level of security as with Docker Swarm.

On the other hand, Kubernetes integration with third-party solutions for secrets is much better. For example, plugging in HashiCorp Vault into a Kubernetes workflow is much smoother than if we’d try to integrate it with Docker Swarm. Using Vault is a better solution than what Kubernetes and Swarm offer.

Conclusion

Even though Kubernetes can be made more secure with Vault and similar products, for now, we are evaluating secrets management that comes with Kubernetes and Docker Swarm.

If we exclude third-party solutions, Docker Swarm has a clear advantage over Kubernetes. Its secrets are more secure by default. Even after tweaking Kubernetes (especially etcd), Docker Swarm is still more secure.

That does not mean that secrets management with both products does not have a lot to be desired. Both have their shortcomings.

> We must proclaim **Docker Swarm as a winner** in this round. Its secrets are more secretive.

Now we’ll move on to the next chapter which is about Namespaces.