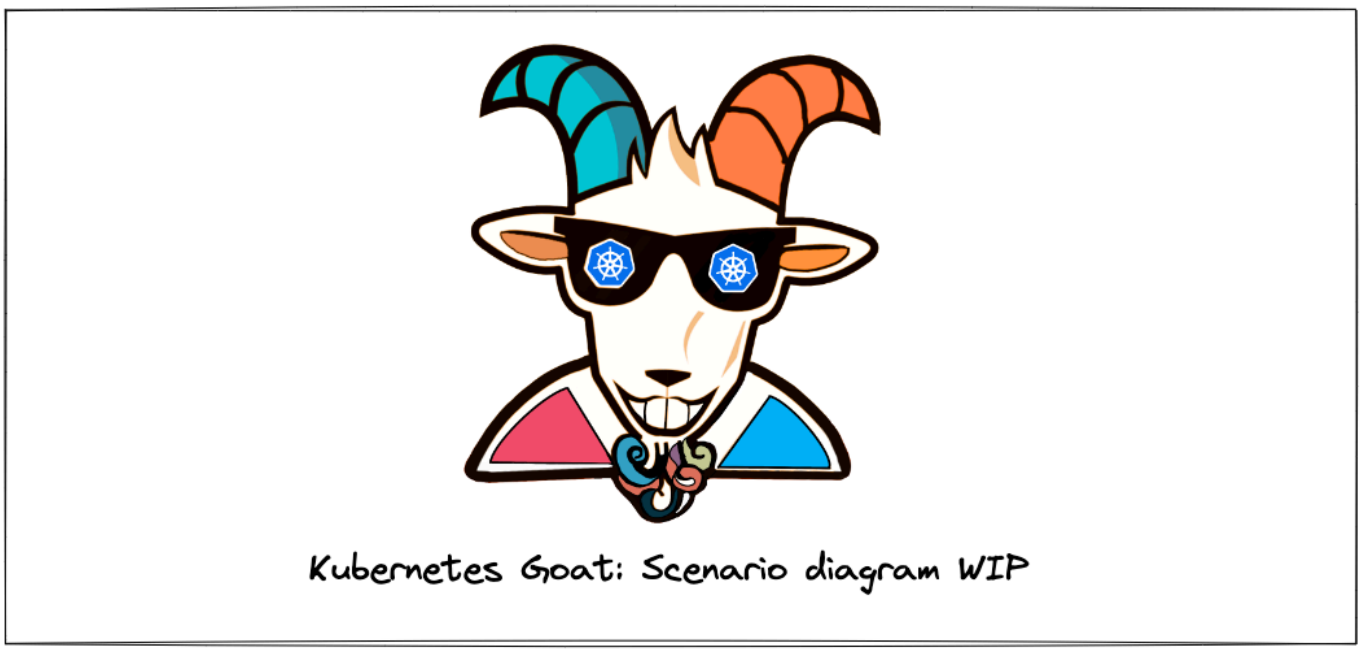
⎈ DoS the Memory/CPU resources

🙌 Overview

Availability is one of the triads in CIA. One of the core problems solved by Kubernetes is the management of the resources like autoscaling, rollouts, etc. In this scenario, we will see how attackers can leverage and gain access to more resources or cause an impact on the availability of the resources by performing the DoS (Denial of Service) if there were no resource management configurations implemented on the cluster resources like memory and CPU requests and limits.



By the end of the scenario, we will understand and learn the following

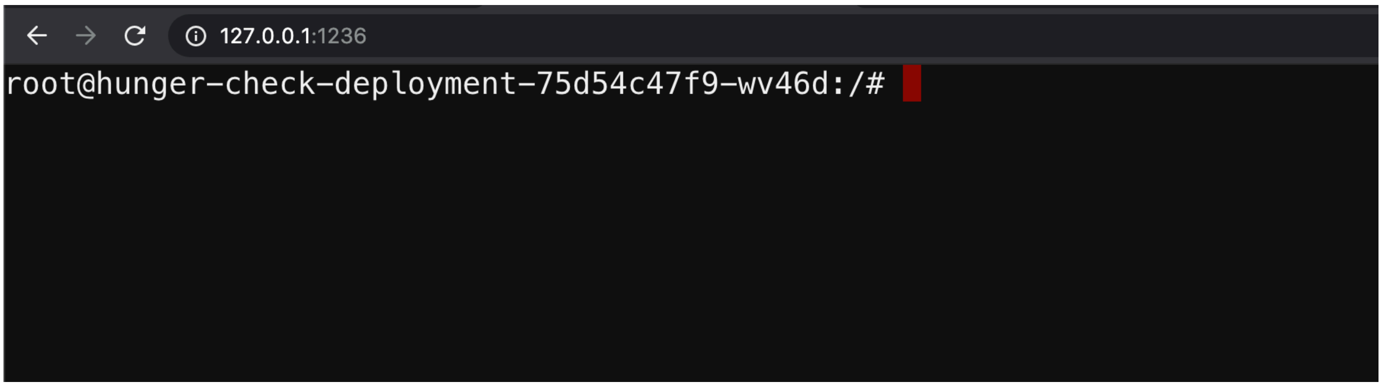
1. Learn to perform the DoS on computing and memory resources using `stress-ng`
2. Understand the Kubernetes resources management for pods and containers
3. Explore the Kubernetes resource monitoring using the metrics and information

⚡ The story

There is no specification of resources in the Kubernetes manifests and no applied limit ranges for the containers. As an attacker, we can consume all the resources where the pod/deployment running and starve other resources and cause a DoS for the environment.

INFO

To get started with the scenario, navigate to http://127.0.0.1:1236



```
root@hunger-check-deployment-75d54c47f9-wv46d:/#
```

🎯 Goal

TIP

Access more resources than intended for this pod/container by consuming 2GB of memory to successfully complete the scenario.

Hints & Spoilers

▶ ✨ How can I DoS resources?

🎉 Solution & Walkthrough

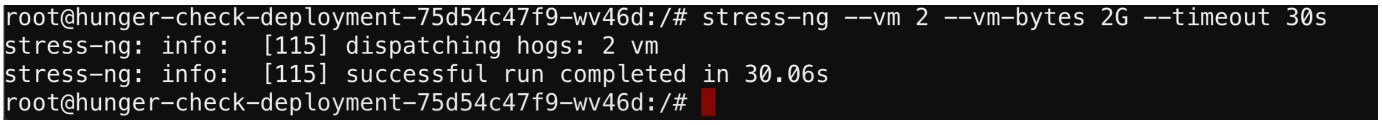
🎲 Method 1

INFO

This deployment pod has not set any resource limits in the Kubernetes manifests. So we can easily perform a bunch of operations that can consume more resources

- We can use simple utilities like `stress-ng` to perform stress testing like accessing more resources. The below command is to access more resources than specified

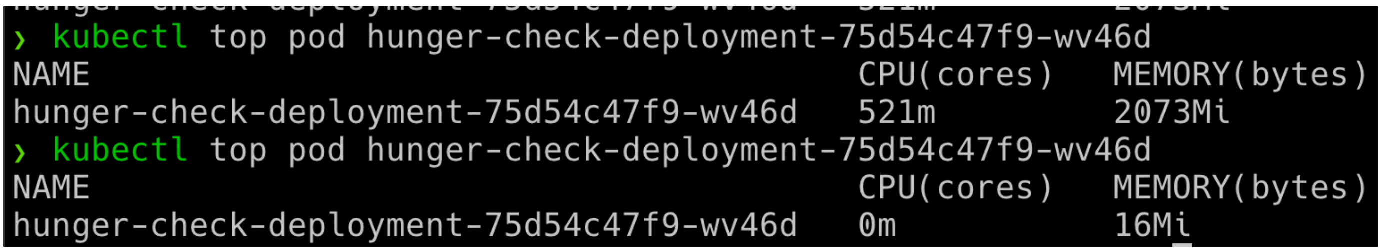
```
stress-ng --vm 2 --vm-bytes 2G --timeout 30s
```



```
root@hunger-check-deployment-75d54c47f9-wv46d:/# stress-ng --vm 2 --vm-bytes 2G --timeout 30s
stress-ng: info:  [115] dispatching hogs: 2 vm
stress-ng: info:  [115] successful run completed in 30.06s
root@hunger-check-deployment-75d54c47f9-wv46d:/#
```

- You can see the difference between the normal resources consumption vs while running `stress-ng` where it consumes a lot of resources than it intended to consume

```
kubectl --namespace big-monolith top pod hunger-check-deployment-xxxxxxxxxx-
xxxxx
```



```
❯ kubectl top pod hunger-check-deployment-75d54c47f9-wv46d
NAME                                        CPU(cores)   MEMORY(bytes)
hunger-check-deployment-75d54c47f9-wv46d   521m         2073Mi
❯ kubectl top pod hunger-check-deployment-75d54c47f9-wv46d
NAME                                        CPU(cores)   MEMORY(bytes)
hunger-check-deployment-75d54c47f9-wv46d   0m           16Mi
```

⚠️ WARNING

This attack may not work in some cases like autoscaling, resource restrictions, etc. Also, it may cause more damage when there are autoscaling is enabled and creating more resources and making more expensive bills to the cloud provider or impacting the availability of the resources and services.

- Hooray 🥳 , now we can see that it can consume more resources than intended which might affect the resource availability and also increase billing

References

- Resource Management for Pods and Containers

Edit this page