# Filtering out the unused K8s Network Policies

**Task to Achieve:**

List all the native Kubernetes network policies, which have 0 endpoints attached to them.

**Requirements:**

1. Have a working kubectl binary installed.
2. Access to the cluster where we want to analyze the policies.
3. Python3 Installed.
4. Pip for Python3
5. Kubernetes Python Client
6. kube-config file.

**Installing Pre-Requisites:**

**1. Python 3**

Ensure Python 3 is installed on your system. The script uses Python 3 syntax and libraries. You can verify if Python 3 is installed and find its version by running:

```
python3 --version
```

If you don't have Python 3, you will need to install it. The installation method depends on your operating system. For example, on Ubuntu, you can install Python 3 by running:

```
sudo apt-get update
sudo apt-get install python3
```

**2. Pip for Python 3**

Pip is the package installer for Python. You'll use it to install the Kubernetes Python client. You can check if pip for Python 3 is installed by running:

```
pip3 --version
```

If it's not installed, you can install it on Ubuntu with:

```
sudo apt-get install python3-pip
```

**3. Kubernetes Python Client**

The script uses the Kubernetes Python client to interact with your Kubernetes cluster. You can install it using pip:

```
pip3 install kubernetes
```

**4. Kubernetes Configuration**

The script uses your local Kubernetes configuration file (`~/.kube/config`) to connect to your cluster. Ensure you have `kubectl` installed and configured to communicate with your cluster. You can verify `kubectl` is correctly configured by running:

```
kubectl get nodes
```

## RUNNING THE SCRIPT

After having the pre-reqs sorted, we can create the script as follows:

1. Create an executable script file using the file below:

```
vi k8s-policy.py
```

Use the following code:

```python
#!/usr/bin/env python3
from kubernetes import client, config

def list_unmatched_network_policies():
    # Load the kubeconfig file
    config.load_kube_config()

    # Create API clients
    v1 = client.CoreV1Api()
    networking_v1 = client.NetworkingV1Api()

    print("Kubernetes Policies with 0 endpoints attached are as below:\n")

    # Open a file to write the unmatched policies
    with open('kubernetes-unused-policies.txt', 'w') as file:

        # List all network policies
        network_policies = networking_v1.list_network_policy_for_all_namespaces()

        for np in network_policies.items:
            # Check if pod_selector.match_labels is None or empty
            if np.spec.pod_selector.match_labels:
                # Build a selector string from the policy's podSelector
                selector = ','.join([f'{k}={v}' for k, v in np.spec.pod_selector.match_labels.items()])
            else:
                # If match_labels is None or empty, it means all pods are selected
                selector = None

            # List all pods in the same namespace that match the policy's podSelector, if selector is not None
            if selector:
                pods = v1.list_namespaced_pod(np.metadata.namespace, label_selector=selector)
            else:
                # If selector is None, list all pods in the namespace
                pods = v1.list_namespaced_pod(np.metadata.namespace)

            pod_names = [pod.metadata.name for pod in pods.items]

            # If no pods matched, print and write the policy name
            if not pod_names:
                policy_info = f"Policy '{np.metadata.namespace}/{np.metadata.name}' has no endpoints attached.\n
                print(policy_info)
                file.write(policy_info.strip() + "\n")  # Write policy info to file without extra newlines

    print("\nThe output has also been written to kubernetes-unused-policies.txt. Analyze it for further informat

if __name__ == "__main__":
    list_unmatched_network_policies()
```

```

```

Save the file.

2. Make it executable by

   ```
   chmod +x k8s-policy.py
   ```

3. Run the script:

```
./k8s-policy.py
```

This should run and print all the policies with 0 endpoints attached. This will also create a text file named “`kubernetes-unused-policies.txt`" which the user can save to analyze the policies later and can also use it to create another script that backs up and deletes those policies provided by the text file as the input.

The output from running the script should look something like this (names of the policies will vary in your environment):

```
tigera@bastion:~/rbc$ ./k8s-policy.py
Kubernetes Policies with 0 endpoints attached are as below:

Policy 'test-policy/allow-same-namespace' has no endpoints attached.


The output has also been written to kubernetes-unused-policies.txt. Analyze it for further information.
```

That generates a text file named: “`kubernetes-unused-policies.txt`", If you cat that file name: it should show the list of policies, which have 0 endpoints selected (names of the policies will vary in your environment)::

```
tigera@bastion:~/rbc$ cat kubernetes-unused-policies.txt
Policy 'test-policy/allow-same-namespace' has no endpoints attached.
```