Introduction to Kustomize

Pavan Kumar
Oct 31, 2020 · 4 min read

How to use Kustomize to efficiently manage Your Kubernetes manifests.

What is Kustomize?

Kustomize is a standalone tool to customise the creation of Kubernetes objects through a file called kustomization.yaml. It introduces a template-free way to customize application configuration. It lets you customize an entire Kubernetes application without touching the actual YAML files. All the customization can be specified and can also be overridden in a special file called kustomization.yaml file. It is natively built into kubectl and follows a purely declarative approach to configuration customization. Without any further due, let's get into the hands-on section of Kustomize.

Kustomize

Install Kustomize:

```
curl -s "https://raw.githubusercontent.com/\
kubernetes-sigs/kustomize/master/hack/install_kustomize.sh"  | bash

sudo mv kustomize /usr/local/bin
```

The above script automatically detects your OS and downloads the corresponding binary to your current working directory. And then move the binary to the bin directory. You can enable kustomize auto-complete feature by adding this to your .bashrc file.

echo “source <(kustomize completion bash)” >> ~/.bashrc

Creating a kustomization.yaml file

```
#The namespace in which all the components should be installed
namespace: kustomize-demo

#The order in which the Kubernetes resources should be installed
resources:
- ns.yaml
- deployment.yaml
- svc.yaml
```

kustomization.yaml hosted with ♡ by GitHub view raw

A sample kustomization.yaml file

Let us understand what is in the above kustomization.yaml file.

1. namespace: This will add the namespace (kustomize-demo) to all the resources.
2. resources: The list of resources to be included. All the resources will be applied in depth-first order.

Make sure that the file ns.yaml, deployment.yaml and sc.yaml are present in the current directory.

You can clone my repo. I have added the sample files to start with kustomize.

```
git clone https://github.com/pavan-kumar-99/medium-manifests.git

cd medium-manifests/
```

And once you are in the folder you can find all the files which are specified in the kustomization.yaml file. Now let us apply our manifests.

```
kustomize build | kubectl apply -f -
```

```
[root@master kustomize-demo]# kustomize build | kubectl apply -f -
namespace/kustomize-demo configured
service/kustomize-test created
deployment.apps/kustomize-test created
[root@master kustomize-demo]#
```

The output of Kustomize build command

Now you can see that your resources got created. Well yeah, this was a very simple scenario. Let us explore a few more scenarios.

Let us suppose you want to change the replica count of the deployment and also add the requests and limit section in the deployment file. This can be achieved by **patchesStrategicMerge** in Kustomize. **patchesStrategicMerge** applies patches to the matching resource config by group, version, kind and name/namespace. This section can either contain the patch that has to be applied or the file from which the patch has to be applied.

```
1  #The namespace in which all the components should be installed
2  namespace: kustomize-demo
3
4  #The order in which the Kubernetes resources should be installed
5  resources:
6  - ns.yaml
7  - deployment.yaml
8  - svc.yaml
```

```
- svc.yaml

#The number of replicas for the kustomize-test deployment
replicas:
- name: kustomize-test
  count: 7

#The portion of patch to be added for the kustomize-test deployment
patchesStrategicMerge:
- |-
  apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: kustomize-test
  spec:
    template:
      spec:
        containers:
          - name: nginx
            image: nginx
            resources:
              requests:
                memory: "64Mi"
                cpu: "250m"
              limits:
                memory: "128Mi"
                cpu: "500m"
```

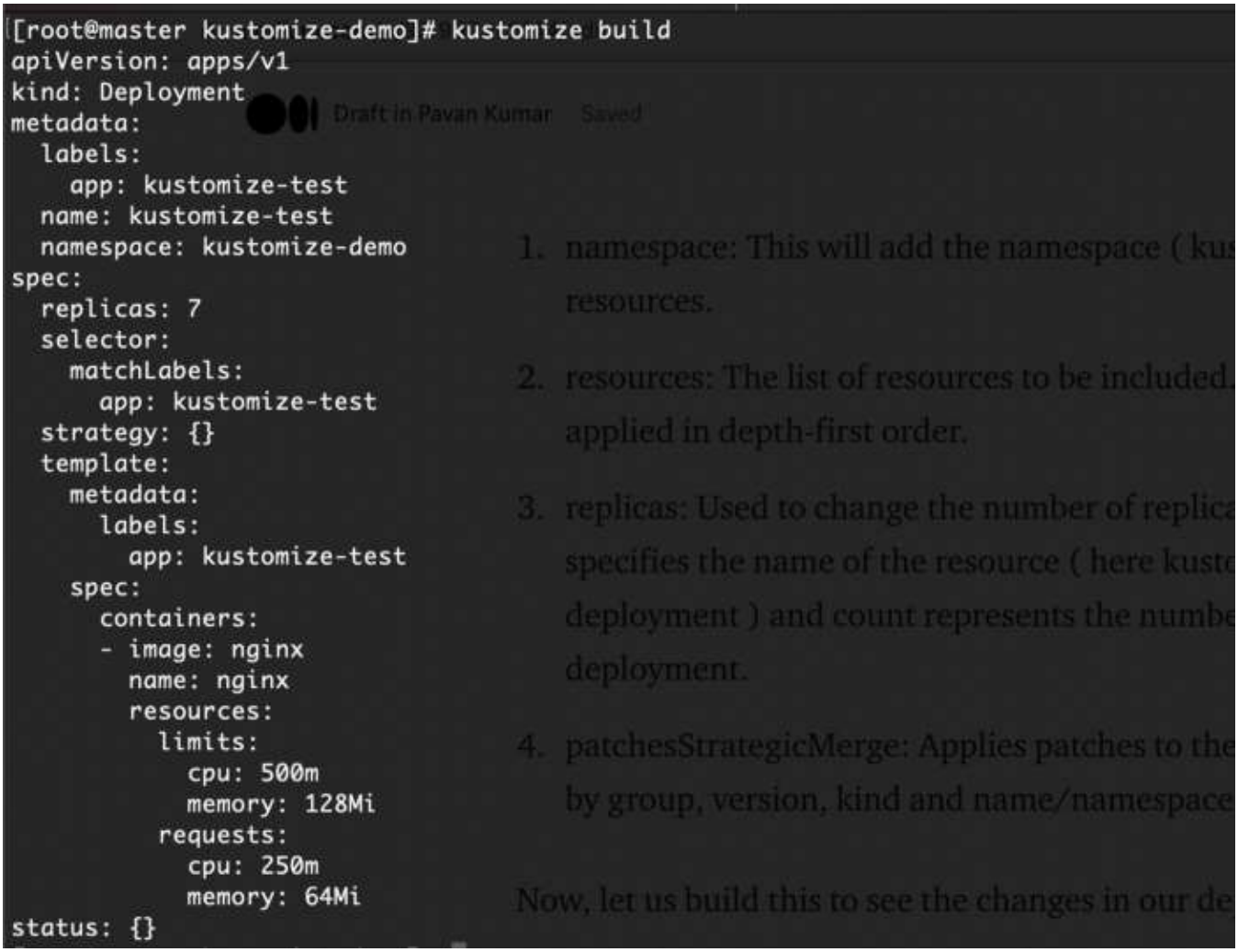
kustomization.yaml hosted with ♡ by GitHub view raw

kustomization.yaml file for patch strategic merge

Let us understand what is in the above kustomization.yaml file.

1. namespace: This will add the namespace (kustomize-demo) to all the resources.
2. resources: The list of resources to be included. All the resources will be applied in depth-first order.
3. replicas: Used to change the number of replicas for a deployment. name specifies the name of the resource (here kustomize-test is the name of the deployment) and count represents the number of replicas for the deployment.
4. patchesStrategicMerge: Applies patches to the matching resource config by group, version, kind and name/namespace.

Now, let us build this to see the changes in our deployment.



kustomize build

Once you are satisfied with your changes go ahead and apply the changes by

```
kustomize build | kubectl apply -f -
```

Wow !!!

And there are many other scenarios where kustomize can be used to customize your actual application without touching the actual yaml files. We would explore a lot more of them in my next article.

Recommended

Introduction to Istio Service Mesh

What is Istio Service Mesh?

medium.com

Securing your secrets using vault-k8s in Kubernetes — Part 1

Kubernetes secrets let you store and manage sensitive data such as passwords, ssh keys, TIs certificates, etc. However...

medium.com

If this post was helpful, please click the clap 👏 button below a few times to show your support for the author! ⬇