Experiment: Getting started with KPT

Kubernetes Package Tool (KPT) is an OSS tool for building declarative workflows on top of resource configuration. Its git + YAML architecture means it just works with existing tools, frameworks, and platforms. Kpt includes solutions to fetch, display, customize, update, validate, and apply Kubernetes configuration.

For Windows:

Download a KPT binary if you don't already have it installed.

https://googlecontainertools.github.io/kpt/installation/binaries/

For this experiment we'll put **kpt.exe** in a **c:\bin** folder and run it from there

C:\bin> **dir kpt***

```
 Volume in drive C is OS
 Volume Serial Number is 5081-CA53

 Directory of C:\bin

09/12/2020  04:47 PM        39,939,584 kpt.exe
               1 File(s)     39,939,584 bytes
               0 Dir(s)  143,198,261,248 bytes free
```

For MacOS:

```
$> brew tap GoogleContainerTools/kpt
https://github.com/GoogleContainerTools/kpt.git
$> brew install kpt
```

For both MacOS and Windows:
$> **cd ~/k3d**

KPT is a swiss-army knife tool for Kubernetes packaging, deployment, and maintenance. As we mentioned it's a combination of git and yaml resources. We'll create a new repository folder for this experiment. We could also put this into a GitHub or other git repository and have cloned this, to be able to push and pull updates.

Create our cluster for this experiment
$> **k3d cluster create local**

```
[36mINFO[0m[0000] Created network 'k3d-local'
[36mINFO[0m[0000] Created volume 'k3d-local-images'
[36mINFO[0m[0001] Creating node 'k3d-local-server-0'
[36mINFO[0m[0001] Creating LoadBalancer 'k3d-local-serverlb'
[36mINFO[0m[0007] Cluster 'local' created successfully!
[36mINFO[0m[0007] You can now use it like this:
kubectl cluster-info
```

Update our KUBECONFIG_FILE for this new cluster
$> **set KUBECONFIG_FILE=C:\k3d\.kube\local**

Pull our local kubernetes configuration to create our KUBECONFIG_FILE reference
$> **k3d kubeconfig get local > %KUBECONFIG_FILE%**

Update our KUBECONFIG environment variable to reference our experiment cluster
$> **set KUBECONFIG=%KUBECONFIG_FILE%**

Verify the cluster information
$> **kubectl cluster-info**
Kubernetes master is running at https://0.0.0.0:52508
CoreDNS is running at https://0.0.0.0:52508/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://0.0.0.0:52508/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

Make our repository folder

$> **mkdir kpt-repo**

Change to the rep folder

$> **cd kpt-repo**

Initialize the folder with git

~/k3d/kpt-repo $> **git init**

Initialized empty Git repository in C:/k3d/kpt-repo/.git/

Create a folder for our first KPT project

~/k3d/kpt-repo $> **mkdir nginx**

Initialize the package with **kpt pkg init** to allow for the creation of the framing and in this case a very simple **Kptfile** and **README.md**

~/k3d/kpt-repo $> **c:\bin\kpt pkg init nginx --tag kpt.dev/app=nginx --description "kpt nginx package"**

writing nginx\Kptfile
writing nginx\README.md

Retrieve the nginx deployment manifest we'll use for our first KPT experiment

~/k3d/kpt-repo $> **cur**l
https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/controllers/nginx-deployment.yaml --output nginx/nginx-deployment.yaml

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   341  100   341    0     0    341      0  0:00:01 --:--:--  0:00:01  1364
```

View the repository folder

~/k3d/kpt-repo $> **dir**

```
 Volume in drive C is OS
 Volume Serial Number is 5081-CA53

 Directory of C:\k3d\kpt-repo

09/12/2020  05:14 PM    <DIR>          .
09/12/2020  05:14 PM    <DIR>          ..
09/12/2020  05:14 PM    <DIR>          nginx
               0 File(s)              0 bytes
               3 Dir(s)  143,375,958,016 bytes free
```

Move to our repository folder

~/k3d/kpt-repo $> **cd nginx**

View the contents of our nginx KPT project folder

~/k3d/kpt-/kpt-repo/nginx $> **dir**

```
 Volume in drive C is OS
 Volume Serial Number is 5081-CA53

 Directory of C:\k3d\kpt-repo\nginx

09/12/2020  05:14 PM    <DIR>          .
09/12/2020  05:14 PM    <DIR>          ..
09/12/2020  05:14 PM               152 Kptfile
09/12/2020  05:14 PM               341 nginx-deployment.yaml
09/12/2020  05:14 PM               171 README.md
               3 File(s)            664 bytes
               2 Dir(s)  143,375,958,016 bytes free
```

Change back to our parent repo folder

~/k3d/kpt-/kpt-repo/nginx $> **cd ..**

View the yaml file that we've just retrieved

~/k3d/kpt-repo $> **notepad nginx\nginx-deployment.yaml**

Note: Notice that we don't have any kpt references currently in this file

View our parent folder. Nothing should have changed, so it better look the same, not surprisingly.

~/k3d/kpt-repo $> **dir**

```
 Volume in drive C is OS
 Volume Serial Number is 5081-CA53

 Directory of C:\k3d\kpt-repo

09/12/2020  05:14 PM    <DIR>          .
09/12/2020  05:14 PM    <DIR>          ..
09/12/2020  05:14 PM    <DIR>          nginx
               0 File(s)              0 bytes
               3 Dir(s)  143,375,958,016 bytes free
```

Configure our required git globals, if not already set

~/k3d/kpt-repo $> **git config --global user.email "you@example.com"**

~/k3d/kpt-repo $> **git config --global user.name "Your Name"**

Commit our nginx package to allow us to apply our resource to our cluster with KPT

~/k3d/kpt-repo $> **git commit -m "Add nginx package"**

```
[master (root-commit) d18fe7d] Add nginx package
 3 files changed, 46 insertions(+)
 create mode 100644 nginx/Kptfile
 create mode 100644 nginx/README.md
 create mode 100644 nginx/nginx-deployment.yaml
```

Tag our repo for release into the wild

~/k3d/kpt-repo $> **git tag nginx/v0.1.0**

If this was cloned from our git repo manager config we would do the following, neither of the next two commands for "git push" will work if you're in a standalone repo, as created in this experiment, but their here for informational reference and of course you're welcome to create a GitHub or other repo and tie this KPT project to that.

~/k3d/kpt-repo $> **git push nginx/v0.1.0**

fatal: The current branch master has no upstream branch.

Similarly if we were tied to a git repo manager without an upstream we could

~/k3d/kpt-repo $> **git push --set-upstream nginx/v0.1.0 master**

fatal: 'nginx/v0.1.0' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

Now that we're ready to load these resources to a cluster with KPT we'll **initialize** the **nginx** project for that usage

~/k3d/kpt-repo $> **c:\bin\kpt live init nginx**

namespace: default is used for inventory object
Initialized: C:\k3d\kpt-repo\nginx\inventory-template.yaml

Now we'll apply the resources using the live apply kpt command.

Note: Because kpt packages are composed of resource configuration can be applied with `kubectl apply -R -f DIR`, however kpt includes the next-generation **apply** commands developed out of the Kubernetes [cli-utils] repository as the [kpt live apply] command. This update provides additional functionality beyond kubectl including returning back status and pruning unused deleted resources.

~/k3d/kpt-repo $> **c:\bin\kpt live apply nginx/ --reconcile-timeout=2m**

deployment.apps/nginx-deployment created
1 resource(s) applied. 1 created, 0 unchanged, 0 configured
deployment.apps/nginx-deployment is NotFound: Resource not found
deployment.apps/nginx-deployment is InProgress: Available: 0/3
deployment.apps/nginx-deployment is InProgress: Available: 1/3
deployment.apps/nginx-deployment is Current: Deployment is available. Replicas: 3
all resources has reached the Current status
0 resource(s) pruned, 0 skipped

View our cluster pods with kubectl

~/k3d/kpt-repo $> **kubectl get pods**

NAME	READY	STATUS	RESTARTS	AGE
nginx-deployment-6b474476c4-dc7d6	1/1	Running	0	36s
nginx-deployment-6b474476c4-sr9vk	1/1	Running	0	36s
nginx-deployment-6b474476c4-tsjww	1/1	Running	0	36s

Check our deployment

~/k3d/kpt-repo $> **kubectl get deployments**

```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           65s
```

Let's take a look at a really cool feature of KPT with the help guides

~/k3d/kpt-repo $> **c:\bin\kpt guide --help**

There are various guides and we can look at the Kustomize option listed under the Ecosystem guides

~/k3d/kpt-repo $> **c:\bin\kpt guide kustomize**

unknown guide "kustomize"

Note: in the guide help there are listed as drop cap, and not surprising are case sensitive names for our help, as well. Let's try that again

View the Ecosystem Kustomize guide in KPT help

~/k3d/kpt-repo $> c:\bin\kpt guide Kustomize

*A kustomization.yaml is just another configuration file and works great for
breaking packages into pieces.*
…

View our existing namespace in this cluster. You're view might vary if you've been experimenting

$> **kubectl get namespaces**

```
NAME              STATUS   AGE
kube-system       Active   23m
default           Active   23m
kube-public       Active   23m
kube-node-lease   Active   23m
```

View our deployments in wide format so that we can see not only the basic detail, but additional like the selector metadata

$> **kubectl get deployments -o wide**

```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS   IMAGES
SELECTOR
nginx-deployment   3/3     3            3           9m43s   nginx        nginx:1.14.2   app=nginx
```

Our deployment was created in the default namespace for this simple experiment, so we can view the pods for our application in that namespace using the selector

$> **kubectl get pods -n default -l app=nginx --watch**

```
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6b474476c4-dc7d6   1/1     Running   0          10m
nginx-deployment-6b474476c4-sr9vk   1/1     Running   0          10m
nginx-deployment-6b474476c4-tsjww   1/1     Running   0          10m
```

KPT allows you to use getters and setters as we noted in the discussion. Let's check our setters for the nginx KPT project

~/k3d/kpt-repo $> **c:\bin\kpt cfg list-setters nginx**

```
  NAME   VALUE   SET BY   DESCRIPTION   COUNT   REQUIRED
```

Note: This file doesn't not have them and as you can imagine with a simple app there might not be boatloads of setters, but replicas seems like an easy one to start with. This would allow someone taking our templates that normally always expect three containers for a microservice and running it as a singleton for a testing or local development need.

Create our first KPT setter for replicas in the nginx project

~/k3d/kpt-repo $> **c:\bin\kpt cfg create-setter nginx/ replicas 3**

List setters again to verify our creation went correctly

~/k3d/kpt-repo $> **c:\bin\kpt cfg list-setters nginx**

```
   NAME     VALUE   SET BY   DESCRIPTION   COUNT   REQUIRED
  replicas   3                 1       No
```

Do a git add to ensure we capture file changes

~/k3d/kpt-repo $> **git add .**

Commit those changes on our nginx package made via the KPT commands

~/k3d/kpt-repo $> **git commit -m "Update replica setters"**

[master 63a4d3c] Update replica setters
 3 files changed, 43 insertions(+), 1 deletion(-)
 create mode 100644 nginx/inventory-template.yaml

Change directory into our nginx package folder
~/k3d/kpt-repo $> **cd nginx**

Let's **grep** the nginx package folder for **replicas**

~/k3d/kpt-/kpt-repo/nginx $> **grep replicas ***

Kptfile: io.k8s.cli.setters.replicas:

Kptfile: name: replicas

nginx-deployment.yaml: replicas: 3 # {"$kpt-set":"replicas"}

Note: we see that when we made our KPT changes that not only did we update the affected yaml resource file, but also our Kptfile

Change back to our parent project repo folder

~/k3d/kpt-/kpt-repo/nginx $> **cd ..**

Apply our KPT nginx package with **kpt live apply**

~/k3d/kpt-repo $> **c:\bin\kpt live apply nginx/ --reconcile-timeout=2m**

deployment.apps/nginx-deployment unchanged
1 resource(s) applied. 0 created, 1 unchanged, 0 configured
deployment.apps/nginx-deployment is Current: Deployment is available. Replicas: 3
all resources has reached the Current status
0 resource(s) pruned, 0 skipped

View the pods that we have running for our nginx app

$> **kubectl get pods -n default -l app=nginx --watch**

NAME   READY   STATUS   RESTARTS   AGE
nginx-deployment-6b474476c4-dc7d6   1/1   Running   0   14m
nginx-deployment-6b474476c4-sr9vk   1/1   Running   0   14m
nginx-deployment-6b474476c4-tsjww   1/1   Running   0   14m

Note: we see the same three pods that we noted after our initial application, that's because although we added a setter, we left the value as the default that we initially had. We'll change that in the next steps.

Update our setter for replicas to 2

~/k3d/kpt-repo $> **c:\bin\kpt cfg set nginx replicas 2 --set-by georgeniece --description 'cost optimization'**

set 1 fields

Ensure we capture all the changes

~/k3d/kpt-repo $> **git add .**

Commit our update for cost reduction and limiting our running pods to 2

~/k3d/kpt-repo $> **git commit -m "Update for cost reduction in running container counts for nginx"**

[master 4674567] Update for cost reduction in running container counts for nginx
2 files changed, 5 insertions(+), 2 deletions(-)

Apply our update nginx KPT package

~/k3d/kpt-repo $> **c:\bin\kpt live apply nginx/ --reconcile-timeout=2m**

deployment.apps/nginx-deployment configured
1 resource(s) applied. 0 created, 0 unchanged, 1 configured
deployment.apps/nginx-deployment is Current: Deployment is available. Replicas: 3
deployment.apps/nginx-deployment is Current: Deployment is available. Replicas: 2
all resources has reached the Current status
0 resource(s) pruned, 0 skipped

View our pods to ensure we see the resulting reduction

~/k3d/kpt-repo $> **kubectl get pods -n default -l app=nginx --watch**

```
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6b474476c4-dc7d6   1/1     Running   0          23m
nginx-deployment-6b474476c4-sr9vk   1/1     Running   0          23m
```

View our yaml resource deployment file

~/k3d/kpt-repo $> **notepad nginx\nginx-deployment.yaml**

Note: Notice that we now have kpt references. We can see how the kpt reference which was added are commented. Not surprisingly, kpt references don't affect our usage if we wanted to deploy this file normally with kubectl, due to their being written as commented annotations.

Disintegrate our experiment cluster

~/k3d/kpt-repo $> **k3d cluster delete local**

[36mINFO[0m[0000] Deleting cluster 'local'
[36mINFO[0m[0000] Deleted k3d-local-serverlb
[36mINFO[0m[0001] Deleted k3d-local-server-0
[36mINFO[0m[0001] Deleting cluster network 'bd7bd4bd8ec595f0bbcc402f5f1090db29db7d27428ed2fa5877bc97a2189367'
[36mINFO[0m[0001] Deleting image volume 'k3d-local-images'
[36mINFO[0m[0001] Removing cluster details from default kubeconfig...
[36mINFO[0m[0001] Removing standalone kubeconfig file (if there is one)...
[36mINFO[0m[0001] Successfully deleted cluster local!