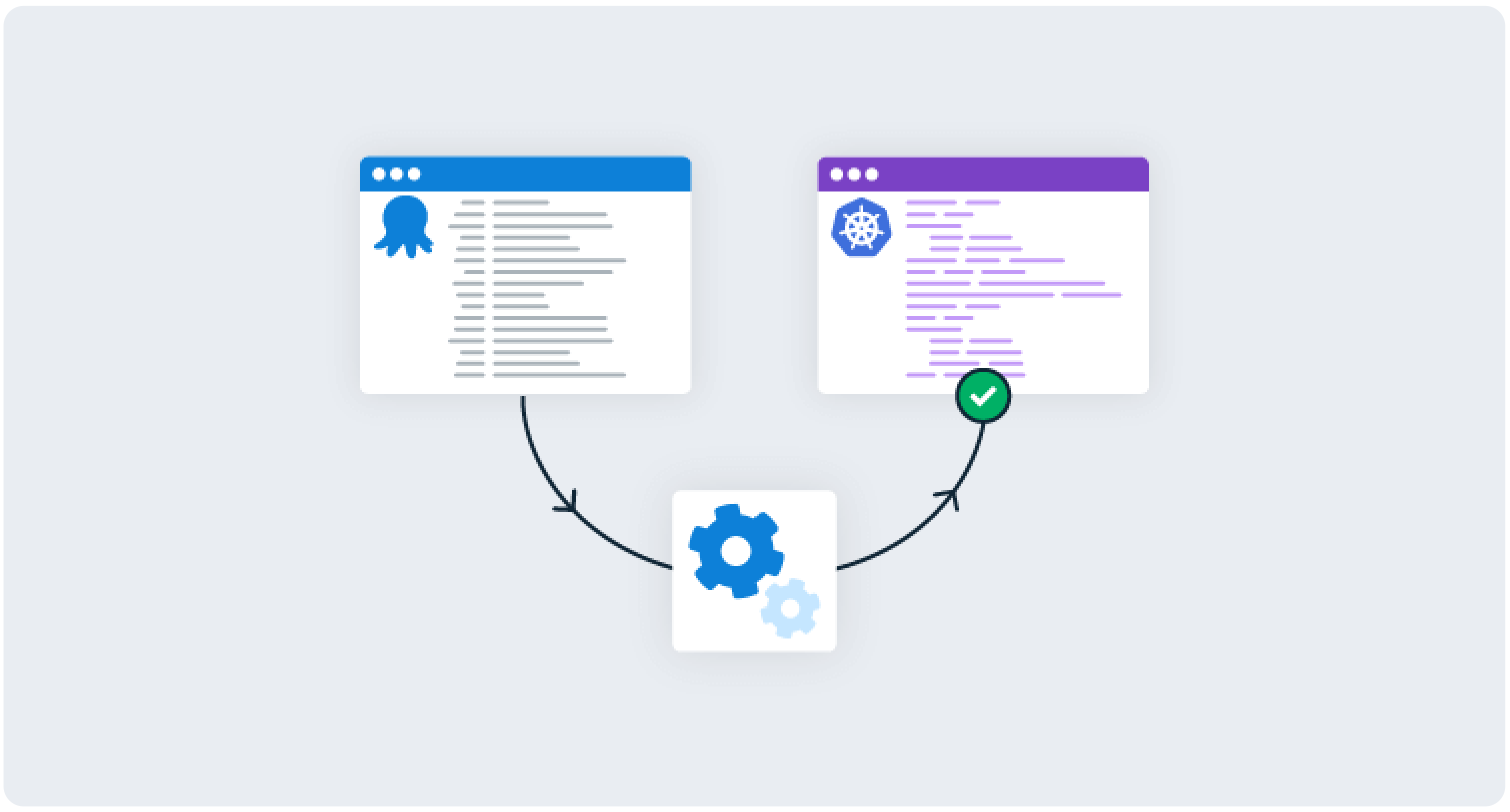


Octopus Deploy's Kubernetes YAML generator

Terence Wong
October 26, 2021 • 6 mins

Kubernetes (K8s) is a powerful container orchestration tool.

Kubernetes reads YAML files that define the resources you're deploying to. Of course, not everybody loves writing YAML. To make this easier, we released a tool that helps developers build YAML files for Kubernetes cluster deployments.

You can find the tool at https://k8syaml.com/.

Overview of our K8s YAML generator

A YAML file is a human-readable configuration file that tells Kubernetes how to provision and deploy a service.

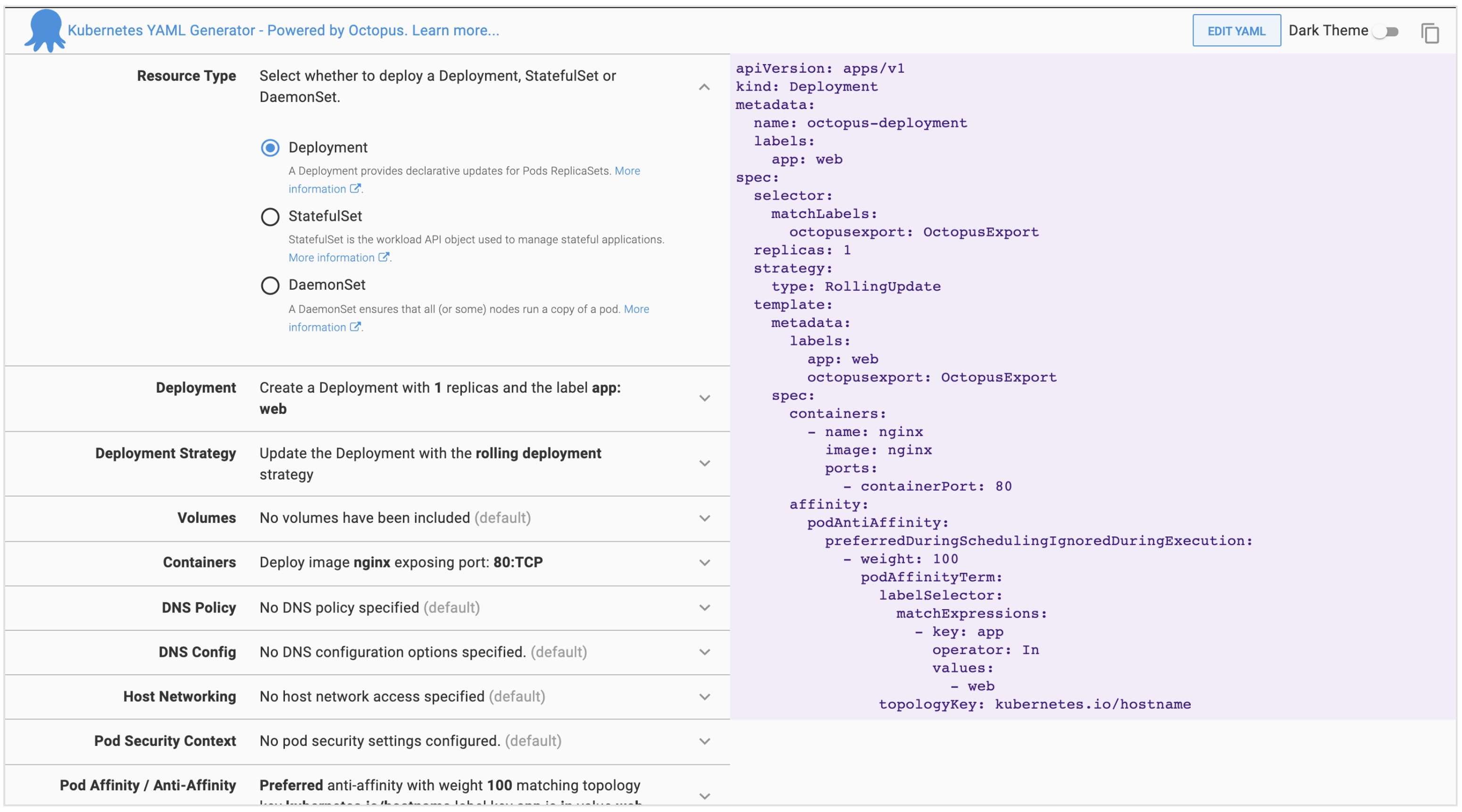
The left-hand side of our tool contains the various options for the YAML file. Each option has a drop-down menu that is populated.

The right-hand side contains the YAML file that Kubernetes will use.

Two features built into the K8s YAML tool are live updates and two-way sync.

Live updates

When changing fields on the left-hand side of the tool, the YAML file on the right-hand side updates to match. Changing the resource type from **Deployment** to **StatefulSet** on the left-hand side will change the YAML file to match the new option. Live updates can add new options to the YAML file that weren't there before.

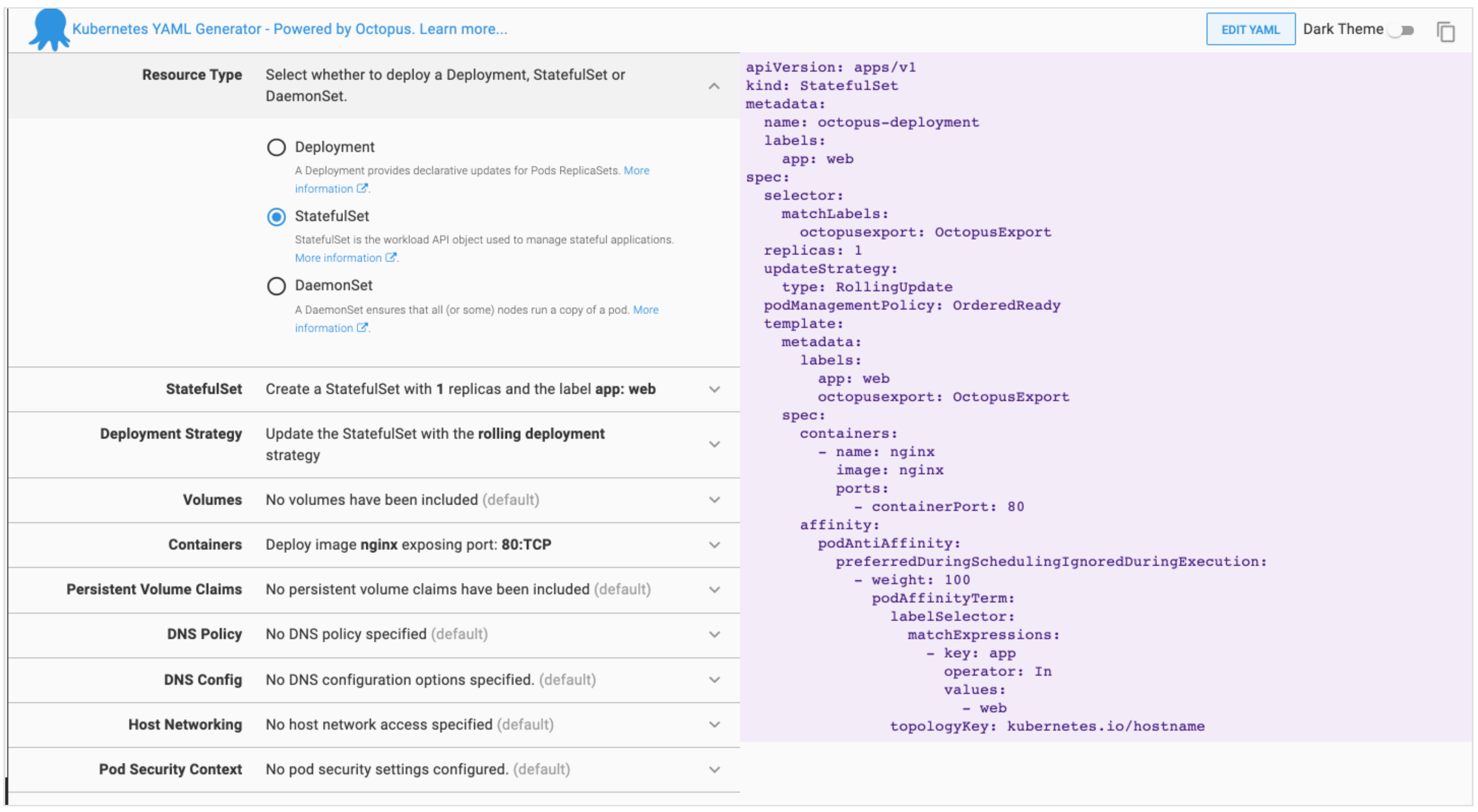


Kubernetes YAML Generator - Powered by Octopus. Learn more... EDIT YAML Dark Theme

Resource Type	Select whether to deploy a Deployment, StatefulSet or DaemonSet.
	[✓ Deployment] A Deployment provides declarative updates for Pods ReplicaSets. More information
	StatefulSet. StatefulSet is the workload API object used to manage stateful applications. More information
	DaemonSet. A DaemonSet ensures that all (or some) nodes run a copy of a pod. More information
Deployment	Create a Deployment with **1** replicas and the label **app: web**
Deployment Strategy	Update the Deployment with the **rolling deployment** strategy
Volumes	No volumes have been included (default)
Containers	Deploy image **nginx** exposing port: **80:TCP**
DNS Policy	No DNS policy specified (default)
DNS Config	No DNS configuration options specified. (default)
Host Networking	No host network access specified (default)
Pod Security Context	No pod security settings configured. (default)
Pod Affinity / Anti-Affinity	**Preferred** anti-affinity with weight **100** matching topology

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: octopus-deployment
  labels:
    app: web
spec:
  selector:
    matchLabels:
      octopusexport: OctopusExport
  replicas: 1
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: web
        octopusexport: OctopusExport
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
      affinity:
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
            - weight: 100
              podAffinityTerm:
                labelSelector:
                  matchExpressions:
                    - key: app
                      operator: In
                      values:
                        - web
                topologyKey: kubernetes.io/hostname
```

Deployment resource type



Kubernetes YAML Generator - Powered by Octopus. Learn more... EDIT YAML Dark Theme

Resource Type	Select whether to deploy a Deployment, StatefulSet or DaemonSet.
	Deployment. A Deployment provides declarative updates for Pods ReplicaSets. More information
	[✓ StatefulSet] StatefulSet is the workload API object used to manage stateful applications. More information
	DaemonSet. A DaemonSet ensures that all (or some) nodes run a copy of a pod. More information
StatefulSet	Create a StatefulSet with **1** replicas and the label **app: web**
Deployment Strategy	Update the StatefulSet with the **rolling deployment** strategy
Volumes	No volumes have been included (default)
Containers	Deploy image **nginx** exposing port: **80:TCP**
Persistent Volume Claims	No persistent volume claims have been included (default)
DNS Policy	No DNS policy specified (default)
DNS Config	No DNS configuration options specified. (default)
Host Networking	No host network access specified (default)
Pod Security Context	No pod security settings configured. (default)

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: octopus-deployment
  labels:
    app: web
spec:
  selector:
    matchLabels:
      octopusexport: OctopusExport
  replicas: 1
  updateStrategy:
    type: RollingUpdate
  podManagementPolicy: OrderedReady
  template:
    metadata:
      labels:
        app: web
        octopusexport: OctopusExport
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
      affinity:
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
            - weight: 100
              podAffinityTerm:
                labelSelector:
                  matchExpressions:
                    - key: app
                      operator: In
                      values:
                        - web
                topologyKey: kubernetes.io/hostname
```

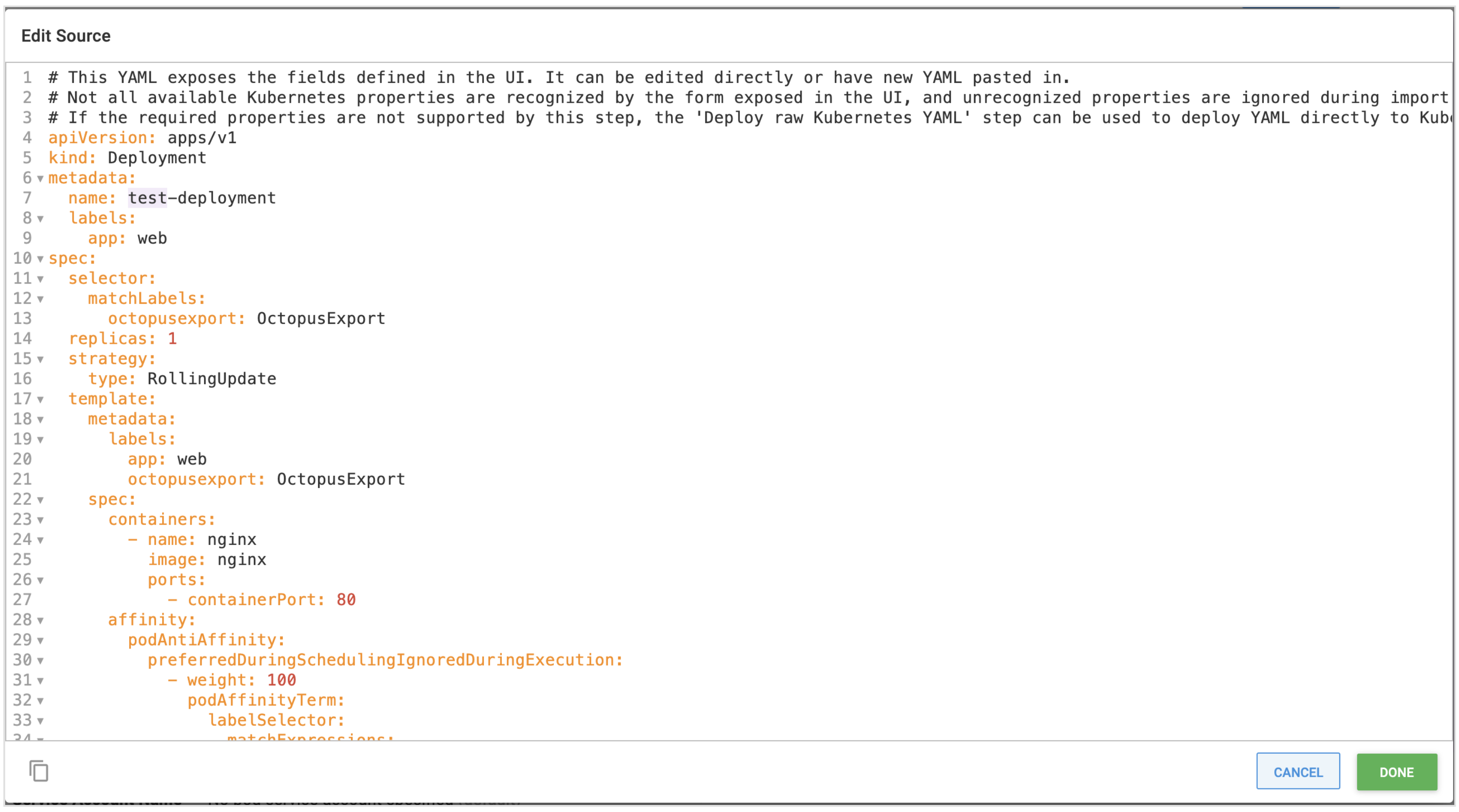
Stateful set resource type

Two-way sync

You can edit the YAML in the tool by selecting the **EDIT YAML** button.

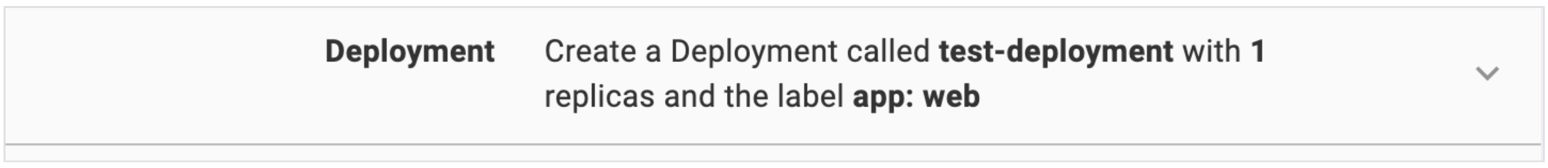
In the edit pane, you can edit any field.

Below, I edited the name of the deployment to `test-deployment` and clicked **DONE**.



```
# This YAML exposes the fields defined in the UI. It can be edited directly or have new YAML pasted in.
# Not all available Kubernetes properties are recognized by the form exposed in the UI, and unrecognized properties are ignored during import
# If the required properties are not supported by this step, the 'Deploy raw Kubernetes YAML' step can be used to deploy YAML directly to Kub
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-deployment
  labels:
    app: web
spec:
  selector:
    matchLabels:
      octopusexport: OctopusExport
  replicas: 1
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: web
        octopusexport: OctopusExport
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
      affinity:
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
            - weight: 100
              podAffinityTerm:
                labelSelector:
                  matchExpressions:
```

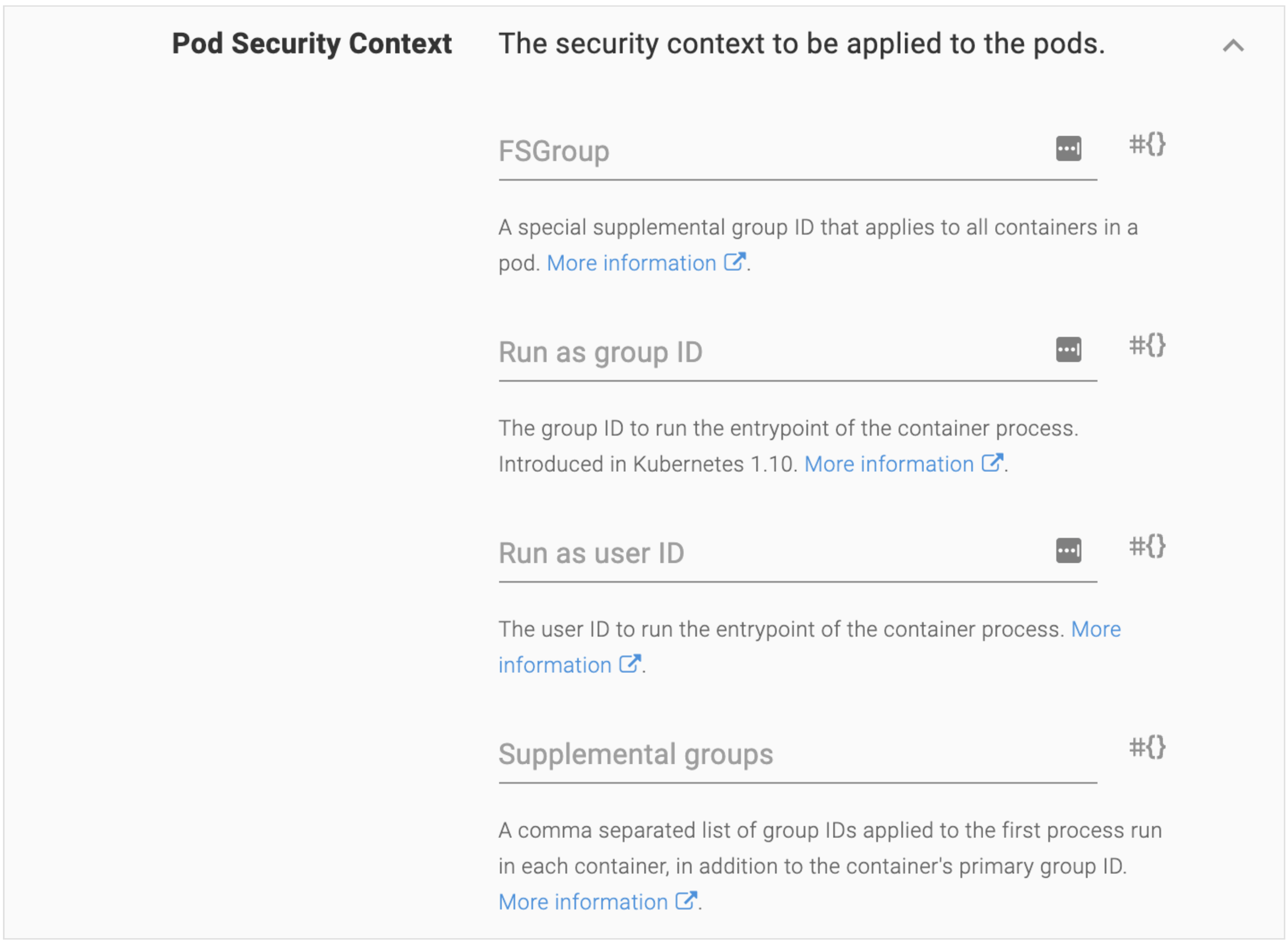
Two-way sync updates the left-hand side of the tool to match the edits.



Deployment Create a Deployment called **test-deployment** with **1** replicas and the label **app: web**

More information

There are several configurable options in a Kubernetes YAML file. Rather than explaining them in detail, the tool links to the official Kubernetes documentation. Most options have a **More information** link that takes you directly to the documentation.



Pod Security Context The security context to be applied to the pods.

FSGroup #{}

A special supplemental group ID that applies to all containers in a pod. More information.

Run as group ID #{}

The group ID to run the entrypoint of the container process. Introduced in Kubernetes 1.10. More information.

Run as user ID #{}

The user ID to run the entrypoint of the container process. More information.

Supplemental groups #{}

A comma separated list of group IDs applied to the first process run in each container, in addition to the container's primary group ID. More information.

Use case

To show the YAML file in action, I use it to deploy a web application to an Azure Kubernetes Service with Octopus Deploy. Feel free to follow along.

Fill out the fields according to the image, by populating the left-hand side.

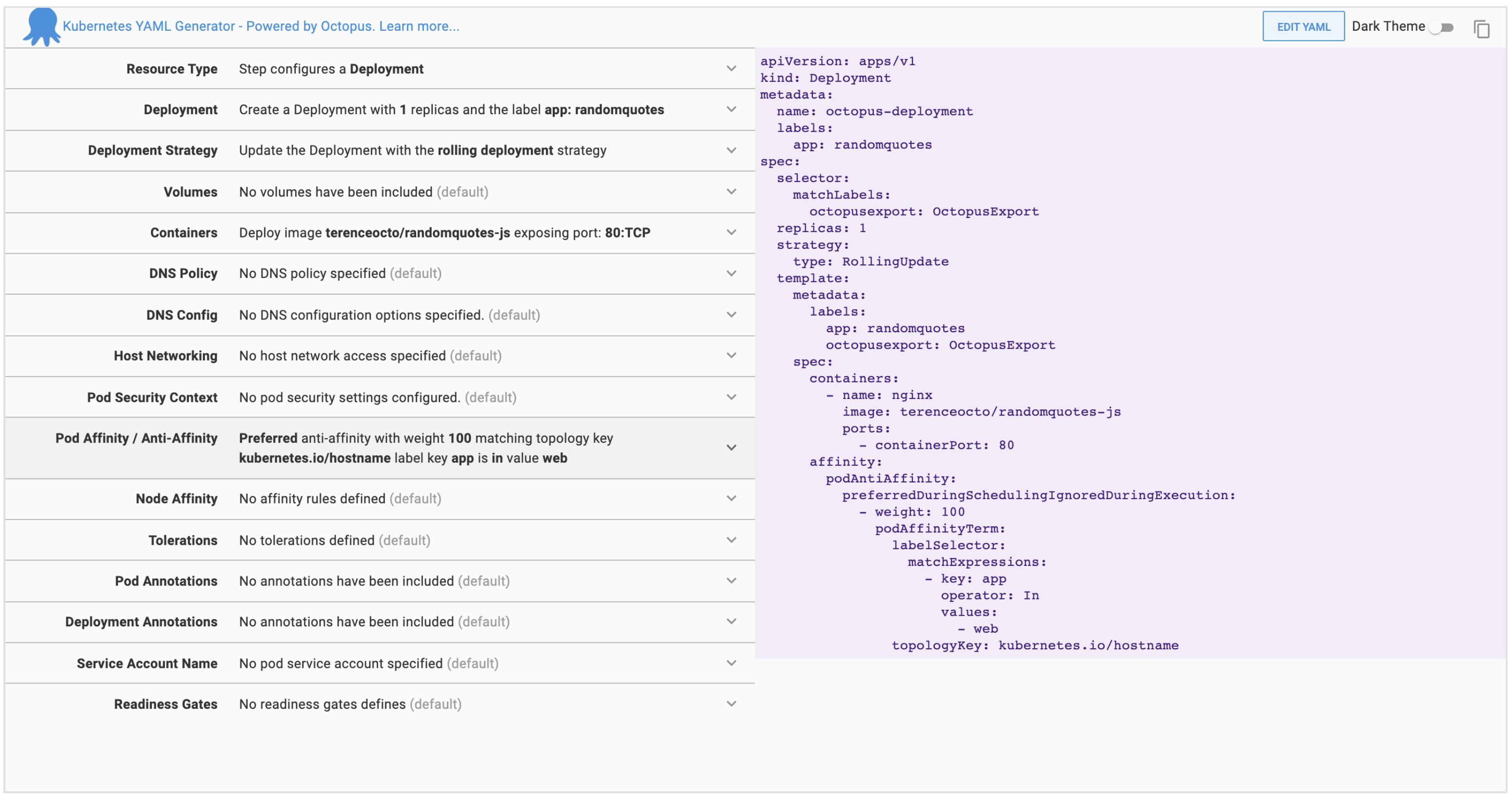
Deployment - Change the value of the app to `randomquotes`

Containers - Delete the nginx default container and add a new container with:

- **Name**: `randomquotes`
- **Package Image**: `terenceocto/randomquotes-js`
- **Add Port**: `TCP:80`

Click **OK** to confirm.

The text on the right is the YAML file we use to deploy to Azure. Copy this file to use later.


Kubernetes YAML Generator - Powered by Octopus. Learn more...
EDIT YAML
Dark Theme

Resource Type	Step configures a **Deployment**
Deployment	Create a Deployment with **1** replicas and the label **app: randomquotes**
Deployment Strategy	Update the Deployment with the **rolling deployment** strategy
Volumes	No volumes have been included (default)
Containers	Deploy image **terenceocto/randomquotes-js** exposing port: **80:TCP**
DNS Policy	No DNS policy specified (default)
DNS Config	No DNS configuration options specified. (default)
Host Networking	No host network access specified (default)
Pod Security Context	No pod security settings configured. (default)
Pod Affinity / Anti-Affinity	**Preferred** anti-affinity with weight **100** matching topology key **kubernetes.io/hostname** label key **app** is **in** value **web**
Node Affinity	No affinity rules defined (default)
Tolerations	No tolerations defined (default)
Pod Annotations	No annotations have been included (default)
Deployment Annotations	No annotations have been included (default)
Service Account Name	No pod service account specified (default)
Readiness Gates	No readiness gates defines (default)

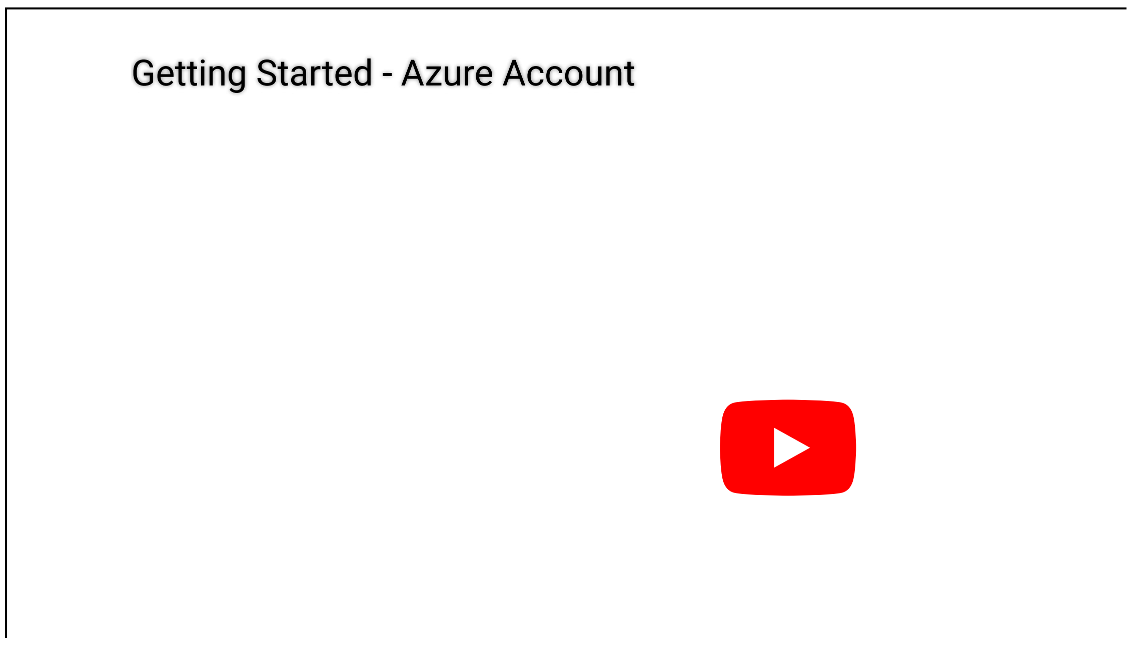
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: octopus-deployment
  labels:
    app: randomquotes
spec:
  selector:
    matchLabels:
      octopusexport: OctopusExport
  replicas: 1
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: randomquotes
        octopusexport: OctopusExport
    spec:
      containers:
        - name: nginx
          image: terenceocto/randomquotes-js
          ports:
            - containerPort: 80
      affinity:
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
            - weight: 100
              podAffinityTerm:
                labelSelector:
                  matchExpressions:
                    - key: app
                      operator: In
                      values:
                        - web
                topologyKey: kubernetes.io/hostname
```

Configuring an Azure account

You need to configure an Azure account and web application to act as a target for the deployment from Octopus. Other targets are possible, such as AWS, or Windows or Linux servers.

Next, you need to create an account in Azure, by navigating to the Azure portal.

Creating an Azure service principal with the Azure portal



1. In the Azure Portal, open the menu, and navigate to **Azure Active Directory**, then **Properties**.
2. Copy the value from the **Tenant ID** field. This is your **Tenant ID**.
3. Next you need your **Application ID**:
   - If you created an AAD registered application, navigate to **Azure Active Directory**, **App Registrations**, click **View all applications**, select the app and copy the **Application ID**. Please note, the Azure UI defaults to **Owned applications** tab. Click the **All applications** tab to view all app registrations.
   - If you haven't created a registered app, navigate to **Azure Active Directory**, **App registrations**, click on **New registration** and add the details for your app, and click **Save**. Make note of the **Application ID**.
4. Generate a one-time password by navigating to **Certificates & Secrets** then **New client secret**. Add a new secret, enter a description, and click **Save**. Make note of the displayed application password for use in Octopus. You can change the expiry date, if you don’t want to accept the default one-year expiry for the password.

You now have the following:

- **Tenant ID**
- **Application ID**
- **Application Password/secret**

Next, you need to configure your resource permissions.

Resource permissions

Resource permissions ensure your registered app has permission to work with your Azure resources.

1. In the Azure Portal navigate to **Resource groups** and select the resource group(s) that you want the registered app to access. If a resource group doesn't exist, create one by going to **Home**, then **Resource groups** and selecting **Create**. After it's created, take note of the Azure subscription ID of the resource group.
2. Click the **Access control (IAM)** option. Under **Role assignments**, if your app isn't listed, click **Add role assignment**. Select the appropriate role (**Contributor** is a common option) and search for your new application name. Select it from the search results and then click **Save**.

The next step is setting up an Azure web application and configuring its properties.

Web application set up

1. In your **Resource group** click **Create**, then **Kubernetes Service**
2. Give the cluster a name and select an appropriate region
3. Click **Review + create**
4. The cluster name will be the AKS cluster name in Octopus Deploy - make note of the resource group name

Add the service principal account in Octopus

With the following values, you can add your account to Octopus:

- Application ID
- Tenant ID
- Application Password/Key

1. Navigate to **Infrastructure**, then select **Accounts**
2. Select **ADD ACCOUNT**, then click **Azure Subscription**
3. Give the account the name you want it to be known by in Octopus
4. Give the account a description
5. Add your Azure Subscription ID - found in the Azure portal under **Subscriptions**

6. Add the **Application ID**, the **Tenant ID**, and the **Application Password/Keyword**

Click **SAVE AND TEST** to confirm the account can interact with Azure. Octopus will attempt to use the account credentials to access the Azure Resource Management (ARM) API and list the Resource Groups in that subscription.

You may need to safelist the IP addresses for the Azure Data Center that you're targeting. See deploying to Azure via a firewall for more details.

> A newly created service principal can take several minutes before the credential test passes. If you've double-checked your credential values, wait 15 minutes and try again.

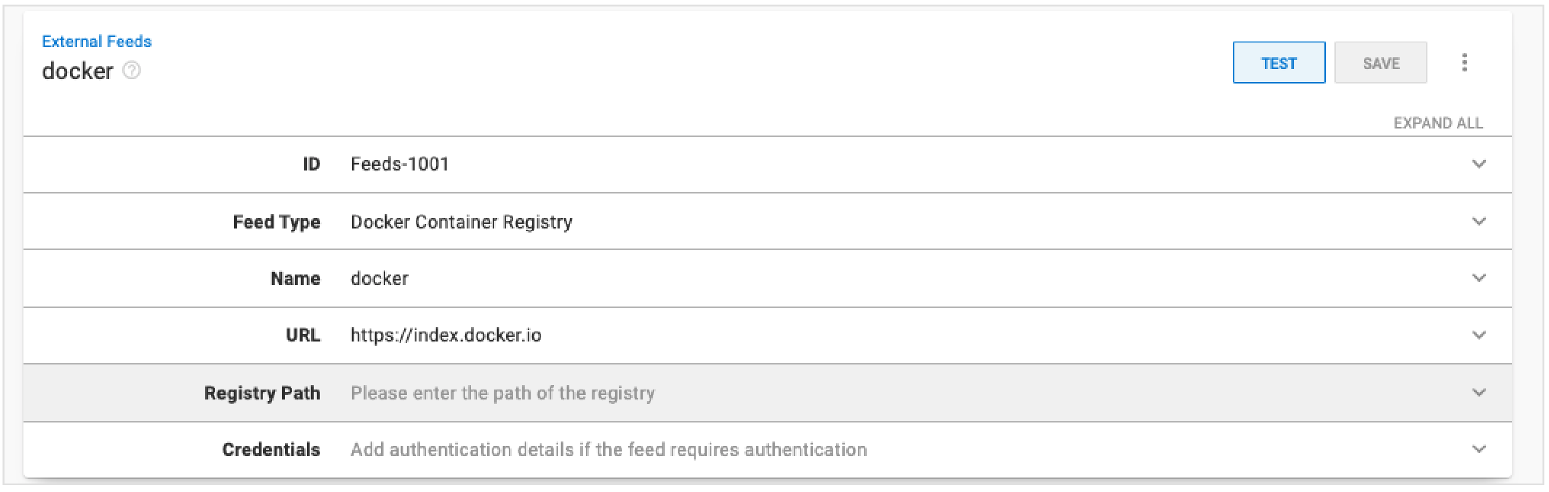
Next, set up Octopus Deploy to load the YAML file to set up a Kubernetes cluster.

Octopus Deploy set up

Create a project with a production environment in your Octopus Deploy instance.

To do this, go to **Infrastructure**, then **Environments**, then **Add Environments** to add the production environment. Then, go to **Projects**, **Add Project** to add a project.

Go to **Library**, then **External Feeds** and set up a docker registry. Since we're using the public repository, you can leave **Credentials** blank.



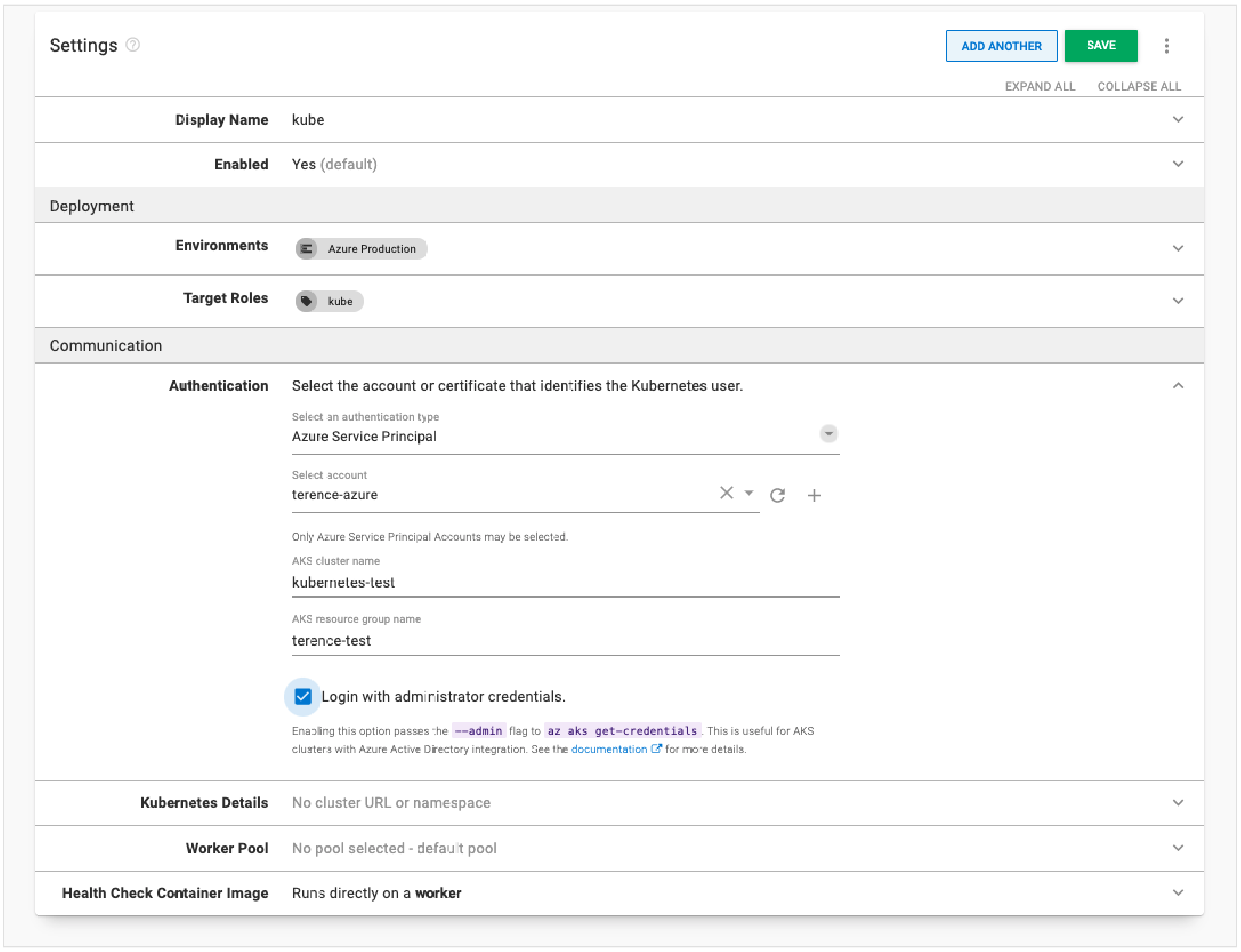
ID	Feeds-1001
Feed Type	Docker Container Registry
Name	docker
URL	https://index.docker.io
Registry Path	Please enter the path of the registry
Credentials	Add authentication details if the feed requires authentication

Set up the Kubernetes target by going to **Infrastructure Deployment Targets**, then **Add Deployment Target**, then **Kubernetes Cluster**.

Fill out the step:

- **Environments** - The environment you set up in Octopus
- **Target Roles** - `kube`
- **Authentication** - `Azure Service Principal`
  - **Select Account** - The Azure account you set up in Octopus
  - **AKS cluster name** - The name of your Kubernetes cluster
  - **AKS resource group name** - The name of your Azure resource group
  - **Login with administrator credentials** - Check this box

Leave everything else as default and click **SAVE** to finish.



Display Name	kube
Enabled	Yes (default)
Deployment	
Environments	Azure Production
Target Roles	kube
Communication	
Authentication	Select the account or certificate that identifies the Kubernetes user.

Select an authentication type: Azure Service Principal

Select account: terence-azure

Only Azure Service Principal Accounts may be selected.

AKS cluster name: kubernetes-test

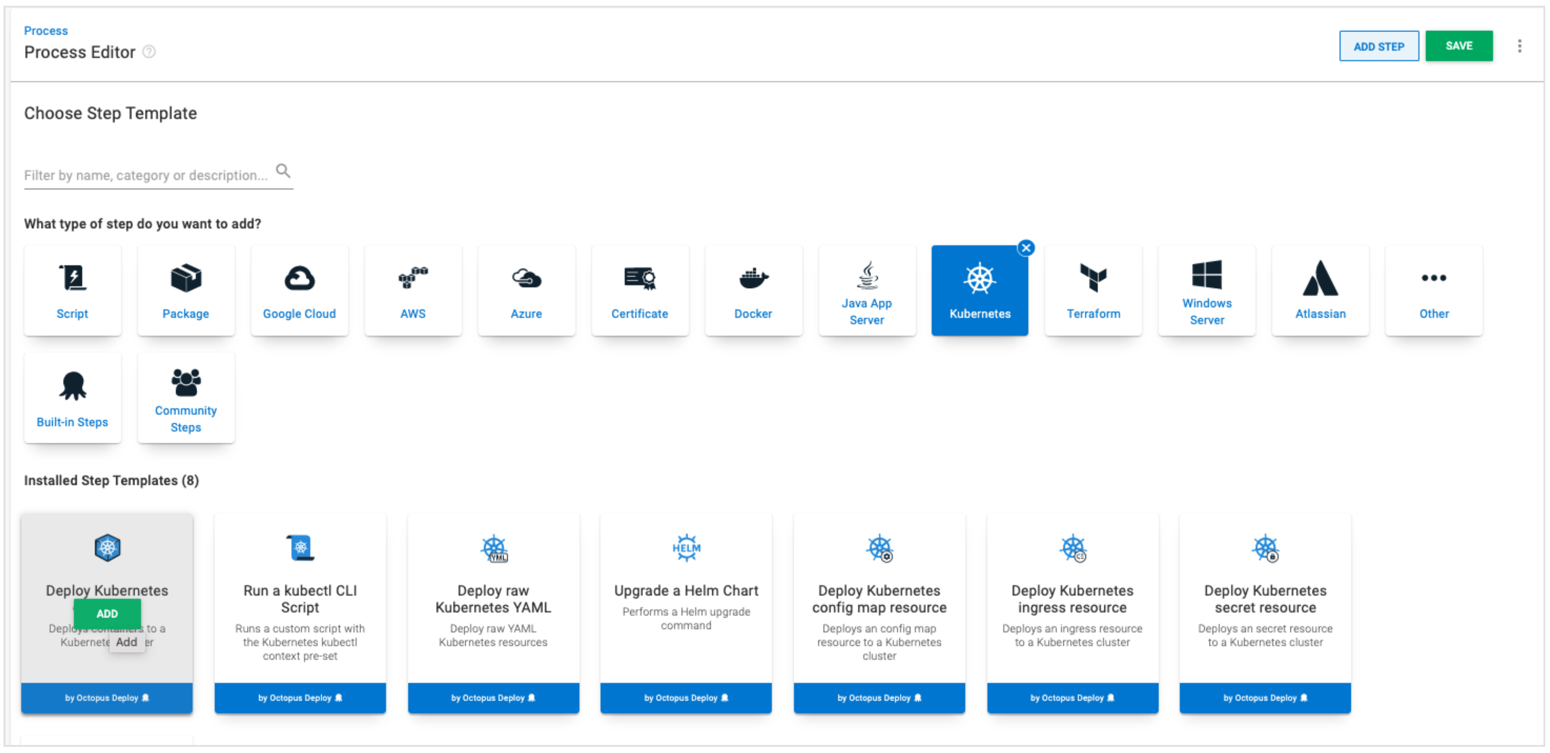
AKS resource group name: terence-test

[✓] Login with administrator credentials.

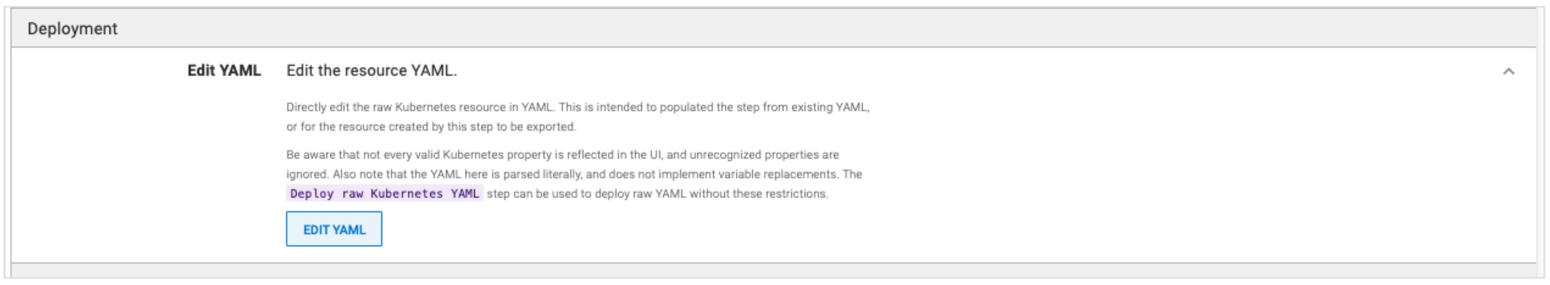
Enabling this option passes the `--admin` flag to `az aks get-credentials`. This is useful for AKS clusters with Azure Active Directory integration. See the documentation for more details.

Kubernetes Details	No cluster URL or namespace
Worker Pool	No pool selected - default pool
Health Check Container Image	Runs directly on a **worker**

In your new project, create a **Deploy Kubernetes** container step by going to **Process**, then **Add Step**, then **Kubernetes**, and then select **Deploy Kubernetes Containers**.


Process
Process Editor
ADD STEP
SAVE
Choose Step Template
Filter by name, category or description...
What type of step do you want to add?
Script
Package
Google Cloud
AWS
Azure
Certificate
Docker
Java App Server
Kubernetes
Terraform
Windows Server
Atlassian
Other
Built-in Steps
Community Steps
Installed Step Templates (8)
Deploy Kubernetes
ADD
Deploys containers to a Kubernetes cluster
Add
Run a kubectl CLI Script
Runs a custom script with the Kubernetes kubectl context pre-set
Deploy raw Kubernetes YAML
Deploy raw YAML Kubernetes resources
Upgrade a Helm Chart
Performs a Helm upgrade command
Deploy Kubernetes config map resource
Deploys an config map resource to a Kubernetes cluster
Deploy Kubernetes ingress resource
Deploys an ingress resource to a Kubernetes cluster
Deploy Kubernetes secret resource
Deploys an secret resource to a Kubernetes cluster
by Octopus Deploy

Make sure to add the `kube` role under the **on behalf of** option to trigger the build for the Kubernetes deployment target. Paste the YAML file from the K8s YAML tool into the **Edit YAML** section. Leave everything else as default and click **SAVE** to finish.


Deployment

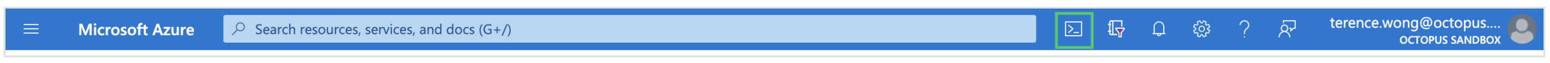
Edit YAML Edit the resource YAML.

Directly edit the raw Kubernetes resource in YAML. This is intended to populated the step from existing YAML, or for the resource created by this step to be exported.

Be aware that not every valid Kubernetes property is reflected in the UI, and unrecognized properties are ignored. Also note that the YAML here is parsed literally, and does not implement variable replacements. The `Deploy raw Kubernetes YAML` step can be used to deploy raw YAML without these restrictions.

Click **Create a release**, and then click the deploy steps to deploy the release. Wait for the success message.

After the deployment is successful, access the web application by exposing the cluster to the internet. Go to the Azure portal and bring up the PowerShell Azure CLI.



```
az aks get-credentials --resource-group myResourceGroup --name myAKSCluster
```

This command will point the CLI to your cluster:

```
kubectl get deployments
```

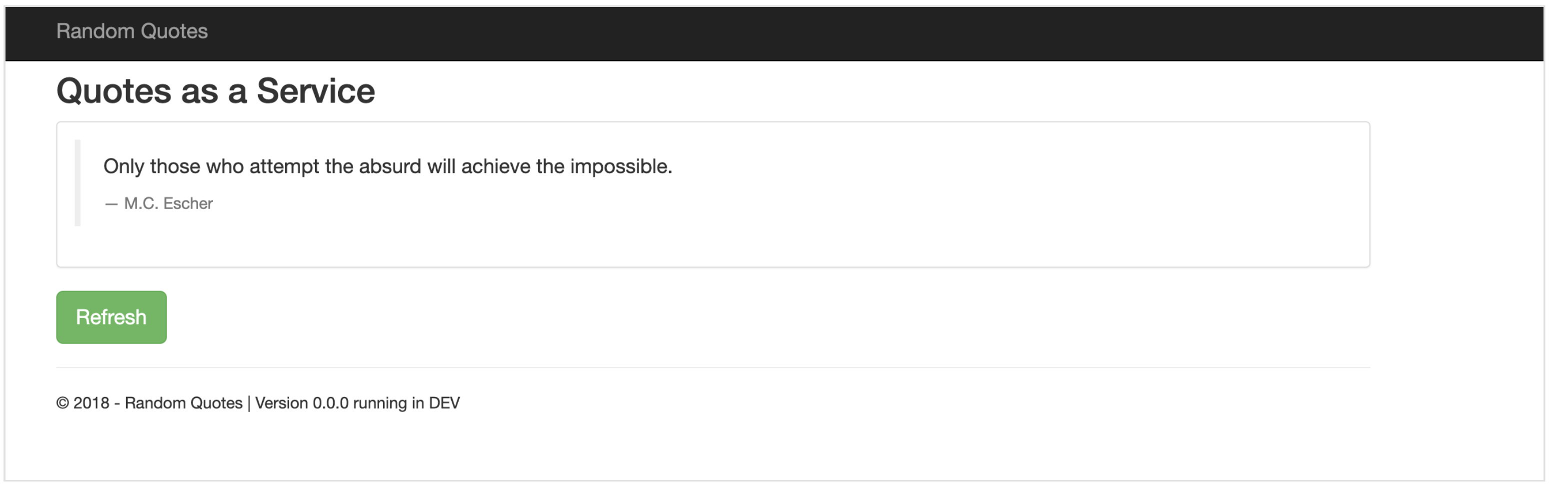
Running this command will get the list of deployments on the cluster. You should see the deployment `octopus-deployment`. Use this name to expose the web application:

```
kubectl expose deployment octopus-deployment --type=LoadBalancer --name=my-service
```

This command creates a service named 'my-service' that generates a public IP to view the web application:

```
kubectl get services
```

Run this command, and you will see "pending" under the External-IP. Wait one minute, run again, and you should see a public IP in that field. Go to the IP address in the browser to view your web application.



Only those who attempt the absurd will achieve the impossible.

— M.C. Escher

© 2018 - Random Quotes | Version 0.0.0 running in DEV

Conclusion

In this post, you learned about the new Kubernetes YAML tool with live updates and two-way sync. You can use the tool to generate a YAML file that is compatible with Kubernetes.

You also ran through a simple use case with Octopus Deploy and Azure Kubernetes Service. You deployed a web application using the YAML file generated by K8s YAML.

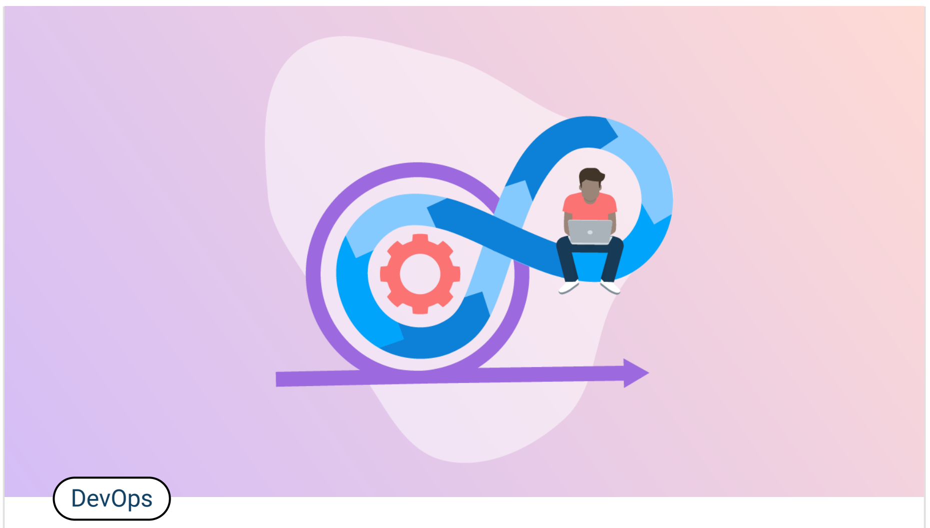
Happy deployments!

Tagged with: DevOps Kubernetes

Related posts

DevOps

Common mistakes in DevOps metrics

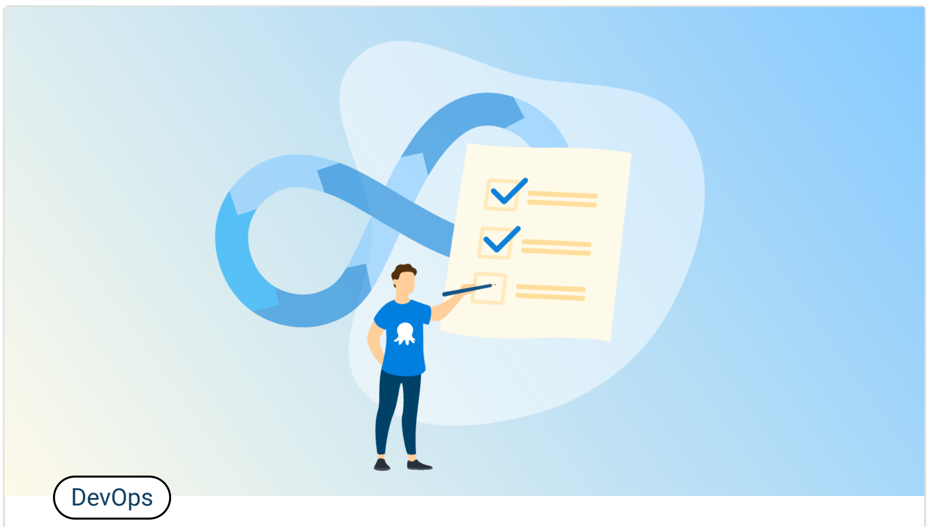


DevOps

Comparing Lean, Agile, and Continuous Delivery

Steve Fenton

December 5, 2022 • 6 mins



DevOps

Best practices for CI/CD

Terence Wong

November 30, 2022 • 5 mins

Newsletter

Logged in as Terence Wong (terence.wong@octopus.com)

Join ~48,000 DevOps professionals and sign up for the latest Octopus news, events, and opinions. No spam. Unsubscribe at any time.

Subscribe

Your privacy is important to us. Read more in our Privacy Policy.