

We've made changes to our [Terms of Service](#) and [Privacy Policy](#). They take effect on September 1, 2020, and we encourage you to review them. By continuing to use our services, you agree to the new Terms of Service and acknowledge the Privacy Policy applies to you.

[Continue](#)

You have 3 free stories left this month. [Upgrade for unlimited access.](#)

# Kubernetes Imperative Commands Every Engineer Should Learn



Luisapreciado

Aug 5 · 3 min read ★

In this post, you will learn about the basic imperative commands that kubernetes offers, which will allow you to create and deploy objects more efficiently and stand out as a DevOps Engineer.

```
43 <body <?php body_class() ?></div>
44 <div id="fb-root"></div>
45 <script>(function(d, s, id) {
46   var js, fjs = d.getElementsByTagName(s)[0];
47   if (d.getElementById(id)) return;
48   js = d.createElement(s); js.id = id;
49   js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.6&appId=289044463428117";
50   fjs.parentNode.insertBefore(js, fjs);
51 })(document, 'script', 'facebook-jssdk');</script>
52 <div id="page" class="site">
53   <a class="skip-link screen-reader-text" href="#content"><?php esc_html_e( 'Skip to content', 'urduube' ); ?></a>
54   <header id="masthead" class="site-header" role="banner">
55     <div class="site-branding">
56       <div class="navBtn pull-left">
57         <?php if(is_home() && $xpanel['homepage-style'] == 1) { ?>
58           <a href="#" id="openMenu"><i class="fa fa-bars fa-3x"></i></a>
59         <?php } else { ?>
60           <a href="#" id="openMenu2"><i class="fa fa-bars fa-3x"></i></a>
61         <?php } ?>
62       </div>
63       <div class="logo pull-left">
64         <a href="<?php echo esc_url( home_url() ) ?>">
65           
66         </a>
67     </div>
68     <div class="search-box hidden-xs hidden-sm pull-left ml-10">
69       <?php get_search_form(); ?>
70     </div>
71     <div class="submit-btn hidden-xs hidden-sm pull-left ml-10">
72       <a href="<?php echo get_page_link($xpanel['submit-link']) ?>" class="header-submit-btn"><i class="fa fa-search fa-3x"></i></a>
73     </div>
74     <div class="user-info pull-right mr-10">
75       <?php
76       if ( is_user_logged_in() ) {
77         <?php echo $xpanel['user-logged-in']; ?>
78       }
79     </div>
80   </header>
81 </div>
```

...

When working with kubernetes, you will mostly be creating objects in a declarative way using YAML definition files.

We've made changes to our [Terms of Service](#) and [Privacy Policy](#). They take effect on September 1, 2020, and we encourage you to review them. By continuing to use our services, you agree to the new Terms of Service and acknowledge the Privacy Policy applies to you.

Continue

. . .

Before we begin, familiarize yourself with the following command parameters:

`--dry-run` : By default, as soon as this is executed, the resource will be created.

`--dry-run=client` : This will **NOT** create the resource. Instead, kubernetes will tell you if it is possible to summon the object.

`-o yaml` : This will output the resource definition file in a YAML format on the screen.

Use the last two in combination to conveniently generate a resource definition file. Later, you can modify it as required, instead of writing the file from scratch.

For example, the following generates a POD manifest file:

```
kubectl run nginx --image=nginx --dry-run=client -o yaml
```

Let's look at more imperative commands for generating PODs.

. . .

## Pod Commands

Create an NGINX Pod:

```
kubectl run nginx --image=nginx
```

Generate a POD manifest YAML file:

```
kubectl run nginx --image=nginx --dry-run=client -o yaml
```

We've made changes to our [Terms of Service](#) and [Privacy Policy](#). They take effect on September 1, 2020, and we encourage you to review them. By continuing to use our services, you agree to the new Terms of Service and acknowledge the Privacy Policy applies to you.

Continue

Let's now learn about deployment related imperative commands.

. . .

## Deployment Commands

To create a deployment from the command line:

```
kubectl create deployment --image=nginx nginx
```

Generate a deployment YAML file template:

```
kubectl create deployment --image=nginx nginx --dry-run=client -o  
yaml
```

***kubectl create deployment** does not have a **— replicas** option. Having said that, if you wish to scale the deployment, you would have to first create it and then scale it using the **kubectl scale** command.*

The following command will create a deployment YAML definition file with the name “nginx-deployment.yaml”:

```
kubectl create deployment --image=nginx nginx --dry-run=client -o  
yaml > nginx-deployment.yaml
```

Again, the **-o** parameter tells kubectl to output a yaml file, while the **> nginx-deployment.yaml** argument provides kubectl with the desired output file name.

You can then modify the YAML file to your needs.

. . .

We've made changes to our [Terms of Service](#) and [Privacy Policy](#). They take effect on September 1, 2020, and we encourage you to review them. By continuing to use our services, you agree to the new Terms of Service and acknowledge the Privacy Policy applies to you.

Continue

```
kubectl expose pod redis --port=6379 --name redis-service --dry-run=client -o yaml
```

*Note: This will automatically use the POD's labels as selectors*

Alternative:

```
kubectl create service clusterip redis --tcp=6379:6379 --dry-run=client -o yaml
```

The above command will not use the POD's labels as selectors, instead it will assume selectors as app=redis. You cannot pass in selectors as an option. Alternatively, generate the file and modify the selectors before creating the service.

. . .

The following command creates a service named nginx of type NodePort to expose the nginx pod's port 80 on port 30080 on the nodes:

```
kubectl expose pod nginx --port=80 --name nginx-service --type=NodePort --dry-run=client -o yaml
```

*Note: This will automatically use the pod's labels as selectors, but you cannot specify the node port. You have to generate a definition file and then add the node port in manually before creating the service with the pod.*

Or

```
kubectl create service nodeport nginx --tcp=80:80 --node-port=30080 --dry-run=client -o yaml
```

We've made changes to our [Terms of Service](#) and [Privacy Policy](#). They take effect on September 1, 2020, and we encourage you to review them. By continuing to use our services, you agree to the new Terms of Service and acknowledge the Privacy Policy applies to you.

Continue

the other cannot accept a node port. In my opinion, I recommend going with the `kubect1 expose` command.

If you need to specify a node port, generate a definition file using the same command and manually input the nodeport before creating the service.

. . .

If you enjoyed this tutorial, I recommend you check out my post regarding YAML files in kubernetes.

### Understanding YAML Files in Kubernetes

After reading this post you will understand the basics of creating kubernetes objects using a YAML definition file. You...

medium.com

*Originally published at <https://luispreciado.blog>.*

---

## Sign up for Best Stories

By Dev Genius

The best stories sent monthly to your email. [Take a look](#)

Get this newsletter

Emails will be sent to [rosanacastillorapp@gmail.com](mailto:rosanacastillorapp@gmail.com).  
[Not you?](#)

DevOps

Programming

Software Engineering

Kubernetes

Software Development

[About](#) [Help](#) [Legal](#)

We've made changes to our [Terms of Service](#) and [Privacy Policy](#). They take effect on September 1, 2020, and we encourage you to review them. By continuing to use our services, you agree to the new Terms of Service and acknowledge the Privacy Policy applies to you.

Continue