

James Carppe

KubeOps: Deploy prod-ready Kubernetes in 120 Minutes

Get a production-ready Kubernetes cluster up and running with Portainer and Omni in under two hours.

- Introduction
- Provision a management environment
- Configure external authentication
- Set up Omni
- Create your Talos machines
- Create your Kubernetes cluster

Secure your cluster

- Upgrade the Kubernetes version
- Role-based Access Control (RBAC)
- Deploy an application
- Summary

Lesson 1 of 11

Introduction



Welcome

Welcome to the KubeOps workshop! In this session we will be provisioning and configuring a Kubernetes cluster using Omni, managed by Portainer. By the end of the workshop you will have:

A management server running Docker Standalone with Portainer Business Edition.
A secured, production-ready three-node Kubernetes cluster running Talos Linux, provisioned via Omni and managed by your Portainer management server.
External authentication to your Portainer server configured via OAuth.
Role-based Access Control (RBAC) set up for your cluster.
A sample application deployed on your cluster.

What you'll need

For this workshop we'll assume you have the following:

A laptop or other computer that you can use to complete the lesson.
An internet connection.
A GitHub account. You'll use this as your OAuth provider. If you have another OAuth provider you prefer you can use that, but this workshop will assume you are using GitHub. You'll also use GitHub to deploy the sample application.

As part of the workshop you'll also need the following:

A Digital Ocean account. This is for the management server and
cluster nodes you'll create. We'll provide a referral link so you can
get started with some credit. If you have another hosting provider
you would prefer you can use them, but this workshop will assume
you are using Digital Ocean (and has some steps and configurations
that are Digital Ocean specific).

An Omni account with Sidero Labs. The workshop will walk you through the setup process for this.

A Portainer Business Edition license. If you do not already have a license, you can <u>sign up here</u>.

Let's get started!

Lesson 2 of 11

Provision a management environment



The first thing we need when setting up a cluster with Portainer is, well, Portainer. In this lesson you will:





Provisioning a server on Digital Ocean

For this workshop, we're going to spin up a Ubuntu Server VM in Digital Ocean to act as our "management server". We'll install Docker on this server, then Portainer itself.

i If you are using a different provider or have already created your management server, you can skip ahead to the next section.

Log into your Digital Ocean account. Click the **Create** button, then select **Droplets**.



Choose the **region** you want to deploy in and optionally the **datacenter** within that region. Make a note of what you choose as we'll want to use the same region and datacenter when we create the Kubernetes cluster. Also make note of the **VPC Network** - we'll need that too.

Choose Region

New York	San Francisco	Amsterdam
Singapore	London	Frankfurt
Toronto	Bangalore	🗱 Sydney
Datacenter		
London • Datacenter 1 • LON1		
Tip: Select the datacenter closes: Avoid any potential latency by sel one or more datacenters.	t to you or your users ecting a region closest to you - a region i	Dismiss is a geographic area where we have
VPC Network - default-lon1 DEFAU	LT	

All resources created in this datacenter will be members of the same VPC network. They can communicate securely over their Private IP addresses.

Next we choose the image. For this workshop we'll pick Ubuntu 24.04 (LTS) x64.

Choose an image

OS Marketplace (281) Custom images



Now we pick the size of the VM. Since this VM is only going to run the Portainer server and not any actual workloads, we can choose a Basic type, Regular CPU type and the 2GB / 2 CPUs plan.

Choose Size

Need help picking a plan? Help me choose 🗹

Droplet Type

SHARED CPU		DEDICA	TED CPU		
Basic (Plan selected)	General Purpose CPU-Optimized M		General Purpose CPU-Optimized Memory-Optimized Stora		Storage-Optimized

Basic virtual machines with a mix of memory and compute resources. Best for small projects that can handle variable levels of CPU performance, like blogs, web apps and dev/test environments.

CPU options



Scroll down to the **Choose Authentication Method** section. Here you can decide how you'll access this server for the initial setup. If you have a SSH key pair you can use, we recommend selecting this option and providing your key pair. Otherwise you can choose to use a password instead.

You will need to have SSH key authentication configured to create the Kubernetes cluster machines later on in this course, so we recommend configuring this now.

Choose Authentication Method ?

(i)

SSH Key

• Connect to your Droplet with an SSH key pair

Password

Connect to your Droplet as the "root" user via password

Add a public SSH key

SSH keys are a more secure method of logging into an SSH server, because they are not vulnerable to common brute-force password hacking attacks.

B We can walk you through setting up your first SSH key

Add SSH Key

Finally, scroll down to the **Finalize Details** section and give your VM a **Hostname**, for example manager.

Finalize Details

Quantity Deploy multiple Droplets with the same configuration.	Hostname Give your Droplets an identifying name you will remember them by.		
- 1 Droplet +	manager		
Tags Type tags here			
Project	~		

When you're ready, click **Create Droplet**. Once the creation completes you will be returned to the list of droplets, where you'll see yours provisioning. When it completes, you'll see the server's IP address listed.

Resources	Activity	Settings	
DROPLETS (1)			
• 💧 ma	nager	68.183.41.75	+ 🥹 + 👌 Upsize

We'll use this IP address to SSH in (using the credentials we provided earlier) and complete the setup.





Installing Docker Standalone

Now that we have our management server provisioned, let's set it up with Docker Standalone.

Use your favorite SSH application to log into the server with your root credentials. You should see something like the following:

james@SPRINGHAWK:~\$ ssh root@68.183.41.75 The authenticity of host '68.183.41.75 (68.183.41.75)' can't be established. ED25519 key fingerprint is SHA256:8hhiedfp0vrCVMX6c0kPNDkvygzUriwF7HxrrcaQcBc. This key is not known by any other names Are you sure you want to continue connecting (yes/no/[fingerprint])? yes Warning: Permanently added '68.183.41.75' (ED25519) to the list of known hosts. root@68.183.41.75's password: Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-51-generic x86_64) * Documentation: https://help.ubuntu.com * Management: https://landscape.canonical.com * Support: https://ubuntu.com/pro System information as of Wed Mar 19 04:20:50 UTC 2025 System load: 0.0 Processes: 122 Usage of /: 3.1% of 57.08GB Users logged in: Θ Memory usage: 9% IPv4 address for eth0: 68.183.41.75 Swap usage: 0% IPv4 address for eth0: 10.16.0.5 Expanded Security Maintenance for Applications is not enabled. 162 updates can be applied immediately. 52 of these updates are standard security updates. To see these additional updates run: apt list --upgradable Enable ESM Apps to receive additional future security updates. See https://ubuntu.com/esm or run: sudo pro status The programs included with the Ubuntu system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright. Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law. root@manager:~#

From here, the first thing we should do is ensure the server's packages are up to date. As this is Ubuntu, we can do this with apt. Run the following two commands:

```
apt update
apt upgrade -y
```

The first command updates the apt repository with the latest list of packages. The second command upgrades your currently installed packages to their latest versions according to the repository we just updated. This may take a few minutes to complete, and we generally recommend rebooting the server once it completes in order to ensure all updated packages are loaded.

reboot

Once the server completes rebooting, log back in with SSH. We're now ready to install Docker Standalone. For this we'll rely on <u>Docker's official installation instructions</u> for Docker Engine on Ubuntu, the commands for which are summarized below:

```
apt install ca-certificates curl
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyring
chmod a+r /etc/apt/keyrings/docker.asc
echo \
```

"deb [arch=\$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/dock{
\$(. /etc/os-release && echo "\${UBUNTU_CODENAME:-\$VERSION_CODENAME}") stable
tee /etc/apt/sources.list.d/docker.list > /dev/null

```
apt update
apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docl
```

Alternatively you can run this helper script to perform the necessary steps:

curl -sSfL https://get.docker.io | sh

Once this is complete, Docker should be installed. You can check this by running:

docker ps

You should see something similar to the below:

root@manager:~# docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES If you do, then Docker is up and running.



Installing Portainer

We have a server, we have Docker installed, so now we need to install Portainer itself.

Installation instructions for Portainer can be found <u>in our documentation</u>. Since we're installing on Docker Standalone, we just need to run these two commands:

```
docker volume create portainer data
```

docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always

The first command creates a persistent data volume for Portainer's configuration. The second command starts the Portainer container.

Once the second command completes you can check Portainer's status by running docker ps again. You should see the Portainer container listed:

```
root@manager:~# docker ps
CONTAINER ID IMAGE COMMAND CREATED
b881e36f9355 portainer/portainer-ee:lts "/portainer" 28 seconds ago
```

Now that the container is running, we need to connect to the Portainer UI in order to complete the setup.

This must be done within 5 minutes of starting the Portainer container for the first time. This is a security measure. If the initial setup has not been completed within 5 minutes of starting, the container will stop listening and will need to be recreated.

Open your web browser of choice and navigate to the following URL (replace ipaddress with the IP address of your management server):

```
https://ipaddress:9443/
```

(i)

Accept the certificate warning (Portainer generates a self-signed certificate during installation) and continue, and you'll be presented with the initial setup.

First we must set up an initial administrator user. You can choose to use a different username or stick with the default of admin. Enter a secure password, confirm it, and click **Create user**.

	portainer.io	
✓ New Portainer installation		
Please create the initial administrator user.		
Username	admin	
Password	••••••	
Confirm password	••••••	~
riangle The password must be at least 12 cf	haracters long. ✓	
Create user		
Allow collection of anonymous statistics. Y	ou can find more information about this in our privacy policy.	
> Restore Portainer from backup	,	
Restore Portainer from backup)	

Next we'll enter the license key. Paste your key into the box provided and click **Submit**.

	po	rtainer.io	
License registrat	ion		
Please register you	· Portainer license.		
Your license key sl	ould start with "2-" or "3-".		
License			
Submit			

Once this is complete, you'll be logged into Portainer. The local Docker environment will be automatically detected and configured, and you're ready to proceed.

	DORTAINER.IO	«	Environment Wizard			白	Ĉ	0	2	admin	~
â	Home		Environment Wizard								
Adı	ministration		Welcome to Portainer	ment of Docker to Porteiner							
ŝ	User-related	~	Get started below with your local porta	ainer or connect more container envi	ronments.						
۵	Environment-related	~									
((*))	Registries										
ទួ	Licenses										
	Logs	~		Ŷ							
Ģ	Notifications			•							
\$	Settings	~	Get Started Proceed using the local environment which Portainer is running in	Add Environments Connect to other environments							

CONTINUE



Summary

In this lesson we have:

Provisioned a management server.
Installed Docker on the management server.
Installed Portainer and completed the initial setup.

In the next lesson we'll set up external authentication for Portainer using OAuth.

Lesson 3 of 11

Configure external authentication

James Carppe

We're up and running with a Portainer management server, but right now we only have one user - the internal administrator. In the real world, you'll want to have multiple users with different levels of access using your cluster. While this can be done with Portainer's internal authentication, when you have an existing authentication provider it makes more sense to leverage that - giving you one place where you manage your users.

In this lesson we will:



START



Setting up GitHub as an OAuth provider

Configuring GitHub to provide OAuth authentication is a relatively straightforward process. You don't need a commercial account - a free GitHub user is plenty to get started. So let's do that.

Log into GitHub as your user, then click your profile picture in the top right and select **Settings**. You'll be taken to your profile page.



From here, scroll down and click **Developer settings** at the bottom of the left menu.

Integrations
88 Applications
Scheduled reminders
Archives
🗟 Security log
🔀 Sponsorship log
<> Developer settings

Next, select **OAuth Apps** from the left menu. This will take you to a list of your OAuth apps configured in GitHub. In my case, I have none set up currently.

	E Settings / Developer Settings	Q Type / to search
1	88 GitHub Apps	OAuth Apps
	Personal access tokens V	
		8
		No OAuth apps
		OAuth apps are used to access the GitHub API. Read the docs to find out more.
		New OAuth app
		View documentation

We'll create a new app now for Portainer. Click the **New OAuth app** button then fill out the form.

Application name should identify the application you're using OAuth with - for example, Portainer. The **Homepage URL** and the **Authorization callback URL** should be the URL

that you use to access the Portainer UI, including the https:// prefix and the port. The **Application description** is optional.

Register a new OAuth app

Application name *
Portainer
Something users will recognize and trust.
Homepage URL *
https://68.183.41.75:9443/
The full URL to your application homepage.
Application description
OAuth for Portainer
This is displayed to all users of your application.
Authorization callback URL *
https://68.183.41.75:9443/
Your application's callback URL. Read our OAuth documentation for more information
Enable Device Flow

Allow this OAuth App to authorize users via the Device Flow. Read the Device Flow documentation for more information.

Register application Cancel

Once you have the form filled out as required, click **Register application**. The application will be created and you'll be taken to the details page. From here, take note of the **Client ID** value displayed - we'll need this for the next step.

Settings / Developer settings / Portainer



To go along with the Client ID, we need a Client secret. Click the **Generate a new client secret** button. If you have two factor authentication set up on your GitHub account, you may be asked to reauthenticate at this point.

Once you have done so, a **Client secret** will be created. Take careful note of this value, as you won't be able to display it again after this.



This is all we need from GitHub. You can now switch back to Portainer to continue the OAuth setup there.



Configuring Portainer for OAuth

With an OAuth application created in GitHub and our client ID and secret obtained, we're ready to configure Portainer for OAuth.

In Portainer, scroll down and expand the **Settings** menu at the bottom left and select **Authentication**.



Here you'll see the authentication options for your Portainer instance. The **Internal** method is currently selected, but we want to change this so select **OAuth** instead.

Authentication method



When OAuth is selected, the options below the selection will change. Here we want to ensure that **Use SSO** is enabled, **Hide internal authentication prompt** is disabled, and **Automatic user provisioning** is enabled.



Since we're using GitHub as our OAuth provider, we can select that option now, and fill in the **Client ID** and **Client secret** fields with the appropriate values from our previous step.

Provider

	0	G	° (C) °	•
Microsoft Microsoft OAuth prov	vider	Google Google OAuth provider	Github Github OAuth provider	Custom Custom OAuth provider
OAuth Configurat	ion			
Client ID (2) Client secret	Ov23lieq0JBG7xQFNOxs			

With this completed, click the **Save settings** button to apply your configuration changes.



Testing OAuth

Let's now test to make sure that the OAuth configuration was applied successfully. Log out of Portainer by clicking your username in the top right and selecting **Log out**.

Once you're logged out, return to the URL for your Portainer instance. The login screen should look a little bit different now.



Click **Login with GitHub** to log back in, and enter your GitHub credentials (and two-factor code) if asked.

Sign in to GitHub to continue to Portainer Username or email address			
			Jamescarppe
Password	Forgot password?		
•••••	•••••		
Sign in			
Sign in v	vith a passkey		
New to GitHub? Create an account			

As this is the first time you're logging in to this OAuth application, you will be asked to authorize access to your GitHub account from your OAuth application. Click the **Authorize** button to proceed.

	Authorize Portainer	
	Portainer by James Carppe wants to access your jamescarppe account	
	Public data only Limited access to your public data	

With authorization granted, the login will proceed and you will be taken to the Portainer dashboard, as your GitHub user.

What you may note is that you don't have access to very much at all as this user.

💼 portainer io 🛛 🔍	Environments	
BUSINESS EDITION	Home \mathcal{C}	👜 🗘 ⑦ A jamescarppe 🗸
Home Honiter View Selected	Information You do not have access to any environment. Please contact your administrator. 	
Administration		
수 Notifications	Environments	Q Search by name, group, 1 ×
	Click on an environment to manage	
	Platfo V Status Tags V Groups	✓ Agent ✓ Clear all Sort ✓ ↓↑
	No environments available.	
		Items per page 10 v

By default, new users logging into Portainer for the first time have no access to any environments or other settings. Once we have our Kubernetes environment configured, we'll configure access to it for this user.

For now, we've shown that OAuth is working and you can log into Portainer as expected. So let's log out again and back in as the local administrator account you created during the initial Portainer setup. You can do this by selecting **Use internal authentication** at the login page instead of Login with GitHub.



Created an OAuth application in GitHub.
Configured our Portainer server to use the GitHub OAuth application as the authentication provider.
Tested logging into Portainer with our GitHub user.

In the next lesson we'll set up an Omni account and configure Portainer to interact with it.

Lesson 4 of 11





To provision our Kubernetes cluster we'll be using Portainer's integration with Sidero's Omni platform. To do this, we'll need to set up an Omni account.

In this lesson we will:



Add the Omni Service Account to our Portainer installation.

START



Creating an Omni account

Sidero offer a free trial of Omni SaaS that you can sign up for <u>on their website</u>. Head there and fill out the form, then click **Start Trial**.

Sign up for a free Omni trial - no credit card needed.

First Name	Last Name
First Name	Last Name
Email	
Email	
What's your role?	
Team member	•
Omni Domain https://	.omni.siderolabs.io
Omni plan	
Business	•
By clicking here, I state that I have conditions.	e read and agree to the terms and
Star	t Trial

Once you've completed the signup you'll receive an email that will include your Omni account address.



Hello!

Your Omni account is now available at:

https://jamesportainer.omni.siderolabs.io

Thanks for signing up for a trial of Omni!

The **first** time you access Omni, create your username and password by clicking Create your Omni username/password on the sign-in page, or by signing in with GitHub or Google. Note: the username or GitHub/Google account must match the email you signed up with.

Go to the URL provided, then click on Create your omni username/password.


Welcome

Log in to continue to Sidero Labs Omni.

Email address*	
Password*	0
Forgot password?]
Continue	
First time logging in? Create your Omni	

username/password

On the Register form, fill in the email address you used when signing up and set a strong password, then click **Continue**.

»)(<

Register Account

Sign Up to continue to Sidero Labs Omni. If registering the first account for this Omni instance, the email *MUST* match the email you used to sign up for Omni.



Already have an account? Log in

You should now receive a second email to verify your email address. Click on the link provided in the email to do so.



Welcome to Sidero Labs Omni!

Thank you for signing up. Please verify your email address by clicking the following link:

Confirm my account

If you are having any issues with your account, please don't hesitate to contact us by replying to this mail.

> Thanks! Sidero Labs Omni

Once verified you will be able to return to the Omni login page and authenticate UI access. Click **Log in** to complete the login process.



You should now arrive at the Omni dashboard.

>)(<	SIDERO OMNI					() Documentation $$ Report an issue $$ \cdots Ongoing Tasks $$
•	Home Clusters	Home				
88 42	Machines Machine Classes	General Information		Machines		Add Machines
\$	Settings	Backend Version API Endpoint	v0.47.1 https://jamesportainer.siderolink.omni.siderolabs.io			\downarrow Download Installation Media
		WireGuard Endpoint	136.144.56.184:10173 🔿			↓ Download Machine Join Config
			_ _	Olised 0%		ර Copy Kernel Parameters
				0 Total		
						CLI
		Recent Clusters		+ Create Cluster	88 View All	Download talosctl

CONTINUE

2

Creating an Omni Service Account

Now that our Omni account is set up, we need to configure a Service Account. This Service Account is how Portainer communicates with Omni in order to provision and manage clusters.

From the Omni dashboard, expand the **Settings** menu on the left and click **Service Accounts**.



On the **Service Accounts** page we'll see that we have none currently created. So let's create one. Click the **Create Service Account** button.

SIDERO OMNI	🛈 Documentation 🔗 Report an i	ssue $ \cdots $ Ongoing Tasks $ \lor $
Home Home Clusters Machines Addenine Classes	Settings ServiceAccounts + C	reate Service Account
Settings Settings Settings Service Accounts Secures Backups	No Records No entries of the requested resource type are found on the server.	ltems per Page: 10 V

Complete the form that appears. For **ID** you can use a name to identify the application that is going to be using the Service Account (in this case, Portainer). Configure the **Expiration Days** to a value that makes sense to you, and ensure the **Role** is set to Admin. When you're ready, click **Create Service Account**.

ID: Portainer	
Expiration Days: 365	
Role: Admin	~

The Service Account will create and you will be provided with a block of code containing a pair of environment variables - OMNI_ENDPOINT and OMNI_SERVICE_ACCOUNT_KEY. Copy both now - they will not be shown again and are needed for the next step.



We're now ready to add the Service Account to Portainer. But before we do that, while we're in the Omni dashboard we can download the machine image we'll use for provisioning our cluster nodes.

CONTINUE



Downloading the machine image

(i)

Return to the **Home** page of the Omni dashboard if you aren't there already. From here, click the **Download Installation media** button on the right.

SIDERO OMNI				() Documentation $$ Report an issue $$ $$ Ongoing Tasks $$ $$ $$
Home Clusters	Home			
Machines Machine Classes	General Information		Machines	Add Machines
Settings Settings	Backend Version			↓ Download Installation Media
දී: Users	API Endpoint	https://jamesportainer.siderolink.omni.siderolabs.io 🗗		
Service Accounts	WireGuard Endpoint	136.144.56.184:10173 🗗		↓ Download Machine Join Config
L 🔊 Backups	Join Token			
				O ^J Copy Kernel Parameters
			0 Used 0%	

In the popup window, from the **Options** dropdown select the Digital Ocean (amd64) option. This is a machine image designed for use with Digital Ocean droplets.

If you are using a different provider than Digital Ocean you may want to choose a different option (or the ISO option for bare metal deployments).

Talos Version: 1.9.2 \vee	Options: Digital Ocean (amd64) 🔨
Pre-Install Extensions	
	Akamai (arm64) Azure (amd64)
NAME	Azure (arm64)
amdgpu	Banana Pi BPI-M64 (arm64) Digital Ocean (amd64)
amd-ucode	Digital Ocean (arm64)
Machine User Labels	Equinix Metal (amd64)
new label 🚫	Equinix Metal (arm64)
	GCP (amd64)

We don't need to make any further customizations to the image here, so click **Download**. The approximately 100MB machine image will generate and then download to your local computer, and we will upload it to Digital Ocean in the next lesson.

This image is specific to your Omni account and contains preconfigured credentials that will join your machines with your Omni account on boot.

We're now done in the Omni dashboard. Next we'll add the Omni Service Account to Portainer.

CONTINUE



Adding the Omni Service Account to Portainer

Log into your Portainer server as the administrator. Expand the **Settings** menu on the left and click **Shared Credentials**.



On this fresh install of Portainer you won't have any credential sets configured. Click the **Add credentials** button to create one.

Settings > Shared credentials		
Shared credentials \mathcal{C}		萨 🤇 ⑦ A admin 🗸
Shared credentials	Q Search	X Remove + Add credentials
Name ↓↑	Provider ↓↑	
	No items available.	
		Items per page 10 v

Under **Provider**, ensure that Sidero Omni is selected and fill out the form. The **Credentials** name can be any value that identifies this credential set. The **Endpoint URL** should be the value of OMNI_ENDPOINT we retrieved in the previous step. The **Service Account key** should be the value of OMNI_SERVICE_ACCOUNT_KEY. With these values entered, click **Add credentials**.

Provider

Sidero Omni Kubernetes management platform	CIVO Civo Kubernetes	Akamai Connected Cloud Linode Kubernetes Engine (LKE)	DigitalOcean DigitalOcean Kubernetes (DOKS)	Google Cloud Google Kubernetes Engine (GKE)	Amazon Web Services (AWS) Elastic Kubernetes Service (EKS)	Microsoft Azure Azure Kubernetes Service (AKS)	SSH Provision a Kubernetes cluster and install Portainer using SSH
Credential details () You will need to create a Any credentials that you	n Omni Service Account wit set up will be usable by all a	h an admin role . You can crea admin users (although the act	ate one through the Sidero O tual values themselves canno	mni UI or using omnictl. See o ot be viewed).	our documentation on creatin	g Sidero Omni credentials foi	r more information.
Credentials name		Omni					
Endpoint URL (?)		https://jamesportain	er.omni.siderolabs.io				
Service account key (2) eyJuYW1IIjoiUG9ydGFpbmVyIiwicGdwX2tleSl6li0tLS0tQkVHSU4gUEdQIFBSSVZBVEUgS0VZIEJMTONLLS0tLS1cblxueFZnRVo5 dZQkJBSGFSdzhCQVFkQTFzQlhiZ0LLdUdBWnladkVaS3U2TXVFV3hicnVv0WvXG5ZdC9nTHNjMVhEOEFBUURmZW5rMFUXU TBmQWFoQzZBQytwdmi2QzhyMG5NakVPRnR3NVF3MmxcbnpTbzhVRzI5ZEdGcGJtVnIRSE5sY25acFkyVmhZMk52ZFc1MExtO sa1pYSnZMbVJsZGo3Q1xua2dRUUZnb0FSQVdDWjI0WkVnV0pBZUV6Z0FRTENRY0IDWkN0VS8xREGa1ZUQU1WQ0FvRUZnSU BUUtiQXdJ2UFSWMhCTGJANFFuQytzVWRLdS9IR28xVC9VTU1XUIZNQUFCVUZnRCtJNIJSUWtVUFZqMWlcbjkvUXY5MW1vQ UMrN0t4U3B1TIISVDN5dVFCQUpPSjN4MEVsUDIrQIqrN2hFb1diY3FkQzFLV1xuTmFWMHdGQmixQIhDU3IwQXgxMEVa0XRaR UJCQUdYVIFERFERUINRRmYmSk5F0TZ6MWdha0hYSG4cron5d72R4WidCYTbLi0a3eM0MVdkbW4z7mNGQAt5U30LF						ZnRVo5dFpFaFIKS3 MFUxUnNPSmNBO 1MExt0XRibWt1YzJ RUZnSUJBQUIaXG5 MV1vQXNGUUQ4Q a0XRaRWhJS0t3W UEvM0pVb1A3V0Q	

Add credentials

Your Omni credential set should now appear in the list of credentials.

Settings > Shared credentials								
Shared credentials $\ensuremath{\mathcal{G}}$	p 🤹 🗘 ③ 🕺 admin 🗸							
Shared credentials	Q Search × 🖻 Remove + Add credentials							
Name ↓↑	Provider ↓↑							
Omni	Sidero Omni							
	Items per page 10 V							



5

Summary

In this lesson we have:

Created a new Omni account.
Created an Omni Service Account for use with Portainer.
Downloaded our personalized Digital Ocean machine image for our cluster machines.



Added our Omni Service Account to Portainer's shared credentials.

Next up, we'll use that machine image to provision our cluster machines.

Lesson 5 of 11

Create your Talos machines



With our Digital Ocean and Omni accounts set up, our Portainer manager server deployed and configured, we're now ready to start creating our cluster. The first step in doing so is to create our machines.

In this lesson we will:

1

2

Upload our personalized Talos machine image to Digital Ocean.

Spin up three droplets to act as our Kubernetes cluster machines.



Uploading the machine image

In the previous lesson we downloaded a Digital Ocean specific machine image to use for our Kubernetes nodes. Now we need to upload that image to our Digital Ocean account.

Log into Digital Ocean and expand the **Manage** menu on the left, then click **Backups & Snapshots**.



From here, select the **Custom Images** tab and click **Upload image**.

Custom Images		Learn
	Imp	ort via URL Upload Image

Locate the image you downloaded from the Omni dashboard and select it. You'll then be asked to answer a few questions about the image.

You can edit the **image name** if you prefer to have a more user-friendly name, or leave it as the same as the filename. For **Distribution** you can set it as Unknown. For the **datacenter region**, choose the region and datacenter you want to provision the cluster in.

Upload an Image

EDIT IMAGE NAME

digital-ocean-amd64-omni-jamesportainer-v1.9.2.raw.gz

Image Size: 104 MB

DISTRIBUTION

Unknown

Choose a datacenter region

Your Image will be located in a single datacenter, but can be transferred later.



When the form is complete click the **Upload** button. The image will now upload to Digital Ocean's servers and become available in the list of custom images. It may take a few minutes for the upload to complete and for the image to then be distributed to the region you selected.

Snapshots	Backups	Custom Image	s				
Custom	Images						Learn 🔁
						Import via URL	Upload Image
Image Na	me		Regions	Uploaded 🔺	Tags	Notes	
O digit Unkn	tal-ocean-amd own OS	64-omn	LON1	2 minutes ago			More ∨

Next we'll provision some servers using this image.



Creating the Talos machines

We can now use the Digital Ocean image to provision the machines we'll use for our Kubernetes cluster. Using the image, the machines will authenticate with your Omni account and become available for cluster provisioning. In Digital Ocean, click the Create button and choose Droplets.



Choose the region and datacenter you uploaded the custom image to.

Choose Region

New York	San Francisco	Amsterdam				
Singapore	London	Frankfurt				
Toronto	Bangalore	🗱 Sydney				
Datacenter London • Datacenter 1 • LON1						
 Tip: Select the datacenter closest to you or your users Avoid any potential latency by selecting a region closest to you - a region is a geographic area where we have one or more datacenters. 						
VPC Network - default-lon1 DEFAU	LT					

All resources created in this datacenter will be members of the same VPC network. They can communicate securely over their Private IP addresses.

Scroll down to **Choose an image** and click the **Custom images** tab. You should see your custom image listed - select it.

Choose an image

OS Marketplace (283) Custom images

digital-ocean-amd6... Unknown OS

For the droplet size we'll again stick with the Basic shared CPU, the Regular disk and the 2GB / 2 CPU option.

Choose Size

Droplet Type

SHARED CPU	DEDICATED CPU								
Basic (Plan selected)	General Purpose	CPU-Optimized	Memory-Optimized	Storage-Optimized					

Basic virtual machines with a mix of memory and compute resources. Best for small projects that can handle variable levels of CPU performance, like blogs, web apps and dev/test environments.

CPU options



The Talos images require using a **SSH key** for authentication, so if you don't already have one set up from earlier you should do so now. If you already have one, select that.

Choose Authentication Method ?

SSH Key Connect to your Droplet with an SSH key pair Password Connect to your Droplet as the "root" user via password Choose your SSH keys james@S... New SSH Key

Under **Finalize Details**, we want to provision 3 servers so change the **Quantity** to 3 Droplets. Set **Hostnames** for each droplet as well.

Finalize Details

same configuration.

Deploy multiple Droplets with the

Quantity

Hostname

Give your Droplets an identifying name you will remember them by.

_	3 Droplets	+	talos01
			talos02
			talos03

When you're ready, click **Create Droplet**. The servers will begin provisioning, and once complete will appear in your list of droplets.

Resources Activity Settings

DDC	CTC	(1)
	E 3	1-41

DRO			
•	♦ talos02	178.128.35.0	+ (*) Upsize
•	♦ talos03	167.99.203.65	+ (>) (Upsize)
•	♦ talos01	167.99.192.84	+ () (Upsize)
	💧 manager	68.183.41.75	+ 🥹 + 👌 Upsize

Once your machines have completed provisioning, there is one last piece of information we need to gather - the **Private IP** range. We'll need this for some Digital Ocean specific configuration adjustments in the next step.

From the list of droplets, click the name of one of your newly-provisioned machines. In the details for the machine, note down the **Private IP** that is listed. In my case, this is 10.106.0.4.



Once you have this, we're done in the Digital Ocean control panel.

CONTINUE

Summary

In this lesson we:

Uploaded our custom image to Digital Ocean.

Provisioned three droplets using our custom image to act as our cluster nodes.

We're now ready to create our cluster.

3

Lesson 6 of 11

Create your Kubernetes cluster



With all our preparation work done, we're now ready to create our Kubernetes cluster!

In this lesson we will provision our environment from Portainer on our Digital Ocean machines, with:



START	
	_
1	

Adding a new environment

Log into Portainer as the local administrator. Expand the **Environment-related** menu on the left and select **Environments**. Here you'll see a list of the environments configured in Portainer - right now you'll only have the local environment where Portainer itself is running.

1	portainer io	«	Environment ma	nagement							
	BUSINESS EDITION		Environr	nents ຂ	7				<u>ب</u>	⑦ Å ad	min ~
â	Home		Env	ironments			Q Search	× 通 Remove	+ Auto onboarding	+ Add enviro	nment
Envi											
Adva	inistration		Name	e↓î	Type ↓↑	URL ↓↑		Group Name ↓↑	Actions		
Adin م	User-related	~	local		 Docker 	unix:///var/	run/docker.sock	Unassigned	兴 Manag	e access	
8	Environment-related	^								Items per page	10 ~
	Environments										
	Groups										
	Tags										

To start provisioning our cluster, click the **Add environment** button. Check the **Create a Kubernetes cluster** option in the Environment Wizard and click **Start Wizard**.

Environment Wizard				
Select your environment(s) You can onboard different types of enviro	nments, select all that apply.			
	•			
Docker Standalone Connect to Docker Standalone via URL/IP, API or Socket	Docker Swarm Connect to Docker Swarm via URL/ IP, API or Socket	Podman Connect to Podman via URL/IP or Socket	Kubernetes Connect to a Kubernetes environment via URL/IP or via kubeconfig import	ACI Connect to ACI environment via API
Set up new environments				
Provision KaaS Cluster	Create a Kubernetes			
Provision a Kubernetes cluster via a cloud provider's Kubernetes as a Service	Cluster Create a Kubernetes cluster on existing infrastructure			
Start Wizard				

On the next page, ensure **Talos Kubernetes** is selected. Give your cluster a **Name**. The Portainer server details should be automatically populated with the correct values.

Create a Kubernetes cluster

»)(<	0	МісгоКВз				
Talos Kubernete	es	MicroK8s				
Omni Kubernetes manag management of Kuberne	ement platform, simplifying the creation and tes clusters	Lightweight Kubernetes				
 β Beta feature - so far, Om ① This will allow you to create 	ni cluster management has only been tested in a limited ate a Kubernetes cluster using Omni with your own exist	set of scenarios. ng Talos nodes, and will then deploy the Portainer agent to it.				
Name* 🕐	talos-cluster					
Portainer server details						
Portainer API * 🕐	https://68.183.41.75:9443					
SERVER URL						
Portainer tunnel * ⑦ server address	68.183.41.75:8000					

In the **Omni cluster summary** section, your Omni credential set should be automatically selected. For the **Talos version** we can keep it at the default value, but for **Kubernetes version** we'll change this to an older version - for this example, let's choose v1.31.7.

Omni cluster summary	
Credentials ③	Omni
Talos version ③	v1.9.5
Kubernetes version ⑦	v1.31.7 ~

In **Cluster machines**, we can select the machines we want to use. You'll see the machines we provisioned in the previous step listed in each dropdown.

For this cluster we're going to use one control plane and two workers, so make your selections accordingly.

Cluster machines

 To add Talos nodes, documentation. 	ownload the	e On	ni IS	30 image <mark>a</mark>	nd t	poot	t your machines using this image. For more information, see the Omni	
Control Plane ③	talos01	٢	×				×	~
Main worker pool	talos02	٢	×	talos03	\$	×	X	~

When provisioning a cluster you can make customizations to the base configuration on both an individual machine and a whole cluster basis. In our case we're going to make a cluster-wide change, and we can do that by expanding the **Cluster configuration patch** section.

In the resulting text box, paste the following block of code:

```
cluster:
network:
cni:
flannel:
extraArgs:
- --iface-can-reach=10.106.0.1
name: flannel
extraManifests:
- https://raw.githubusercontent.com/alex1989hu/kubelet-serving-cert-app
- https://github.com/kubernetes-sigs/metrics-server/releases/latest/dow
machine:
kubelet:
extraArgs:
rotate-server-certificates: true
```

Adjust the value of the --iface-can-reach option to suit the **Private IP** you copied from the machines you provisioned in the previous step. This is a configuration adjustment that is required for cluster networking to function properly on Digital Ocean. The other changes in the patch enable the necessary functionality for the Kubernetes metrics server to run.

Cluster Configuration patch

Edit	Cluster Configuration patch Ctrl+F for search ③
Clu Co	uster Configuration Patches allow you to customize settings for all nodes in a cluster or specific groups (such as control plane or worker nodes). See the Talos nfig Reference for more information.
	Copy to clipboard
1	cluster:
2	network:
3	cni:
4	flannel:
5	extraArgs:
6	iface-can-reach=10.106.0.4
7	name: flannel
8	extraManifests:
9	- https://raw.githubusercontent.com/alex1989hu/kubelet-serving-cert-approver/main/deploy/standalone-install.yaml
10	- https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
11	machine:
12	kubelet:
13	extraArgs:
14	rotate-server-certificates: true

When you're ready, click **Provision environment**. You should get a success notification in the top right - when you do, you can click **Close** to exit the Environment Wizard.

You'll be returned to the **Environments** page, where you should now see your cluster provisioning.

Environment management						
Environments C 🖗 🗘 🕐 A admin						③ A admin ∽
0	Environments		Q Search	× Remove	+ Auto onboarding	+ Add environment
	Name ↓↑	Type ↓↑	URL ↓↑	Group Name ↓↑	Actions	
	local	Docker	unix:///var/run/docker.sock	Unassigned	유 Manage	access
	talos-cluster	⊕ Kubernetes	⊖ Provisioning Omni cluster ③	Unassigned	-	
					I	Items per page 10 🗸

Depending on the size of your cluster and the machine selections you made, the cluster provision can take a few minutes to complete. Once it has, the Environments list will update to display the Portainer URL and additional options for the environment.

Environment management							
Envi	ronments C				☆ ⁽¹⁾ ⁽²⁾ ⁽		
	Environments		Q Search	× Remove	+ Auto onboarding + Add environment		
	Name ↓↑	Type ↓↑	URL↓↑	Group Name $\downarrow\uparrow$	Actions		
	local	Docker	unix:///var/run/docker.sock	Unassigned	A Manage access		
	talos-cluster	& Kubernetes	https://68.183.41.75:9443	Unassigned	兴 Manage access		
					Items per page 10 v		

If you return to the Portainer home page, you should now see the environment listed.

Environments						
Home C 🖗 🧔 🖗 admin 🗸						
Environments	Q Search by name, group, 1 × CR Refresh					
Click on an environment to manage						
Platform V Connection V Tags V Groups	; V Agent Versi V Clear all Sort By V					
local OUp 1/2 2025-03-20 17:06:31 Standalone 28.0.1 /var/run/docker.sock	S Browse snapshot					
© logo chassigned v no tage v total S 0 stacks S 1 container (0 1 (0 0 ⊂ 0 ⊂ 0 目 1 volume := 1 image (0)	2 CPU					
talos-cluster Sheartbeat 1/2 2025-03-20 17:10:30 TalosVersion 1.9.5 CRunning Kuber	netetes API ready Control plane ready 3 Browse snapshot					
Group: Unassigned 🛇 No tags 🗇 Edge Agent Standard 2.27.2						
No snapshot available	院 Disconnected ¹⁰³					
	Items per page 10 V					

CONTINUE



Summary

Congratulations, you have provisioned a three node Kubernetes cluster through Portainer and Omni! Our cluster includes:

The Kubernetes metrics server enabled through a cluster configuration patch.
Some tweaks to the configuration for Digital Ocean.
Kubernetes v1.31.7 deployed so that we can upgrade it later.

Technically you are now ready to go with a working Kubernetes cluster. But next, we'll make some additional tweaks to improve security.

Lesson 7 of 11

Secure your cluster

James Carppe

Now that we have our cluster up and running, there are some configuration changes we should make to secure it. Omni clusters are quite secure out of the box, but there are still adjustments that can be made.

In this lesson we will:

Restrict access to the default namespace.
 Configure resource quotas on the cluster.
 Examine the options available via OPA Gatekeeper.

START

1

Restricting access to the default namespace

Kubernetes clusters ship with a default namespace configured, which is where workloads that do not specify a namespace would be deployed. For organizational purposes as well as security reasons, we recommend creating namespaces for your workloads. Namespaces can be configured with access restrictions and resource quotas (both of which we'll cover), and in addition can make it easier to keep track of workloads.

To avoid the possibility of your users deploying to the default namespace (which cannot be configured with resource quotas or access restrictions), we can restrict the usage of it to administrators only.

Log into Portainer as the local administrator, then select your Kubernetes cluster. Expand the **Cluster** menu on the left and select **Setup**.



This page lets you configure your cluster settings to suit your needs. You can find a full description of the options here <u>in our documentation</u>. For now, scroll down to the **Security** section and enable **Restrict access to the default namespace**.

Security

() By default, all the users have access to the default namespace. Enable this option to set accesses on the default namespace.

Restrict access to the default namespace



Restrict secret contents access for non-admins (UI only) ③
Click the **Save configuration** button to apply your changes.

CONTINUE

Configuring resource quotas

Limiting the amount of CPU and memory that can be consumed by individual applications and namespaces is a good idea in many cases. Doing so can help to avoid situations where a deployment starts to consume all the available resource on a cluster, causing other deployments to grind to a halt. In addition, Kubernetes by default allows you to over-commit resources when deploying, letting you assign more CPU and memory than is actually available in your cluster to namespaces. For most production workloads, we recommend turning this option off.

First, we'll need to configure resource quotas on our existing namespaces. With your Kubernetes environment selected, click on **Namespaces** in the left menu. This will take you to the list of namespaces on your environment.

Namesp	aces					
Nam	hespace list \mathcal{C}				<u>ب</u>	⑦ A admin ∨
9	Namespaces		Q Search	X 🗎 Remove	+ Add with form	+ Create from code
() S	ystem resources are hidden, this can be changed in the table s	ettings				
	Name ↓↑	Status ↓↑	Quota ↓↑	Created ↓↑	Actions	
	default	Active	-	2025-03-20 17:04:53	-	
	kubelet-serving-cert-approver	Active	-	2025-03-20 17:06:58	뽔 Manage acc	cess
						Items per page 10 v

At this point you should have only two namespaces:

- The default namespace, which we can't adjust quotas on but, thanks to the option we just set, cannot be provisioned to by nonadministrator users.
- A namespace called kubelet-serving-cert-approver, which contains the deployment used to rotate the certificates used with the metrics server component we added with our customizations during the deployment.

We want to set a quota on this second namespace, so click on kubelet-serving-certapprover to be taken to the **Namespace details** page.

amespaces > kubelet-serving-cert-approver							
Namespace detail	amespace details $arphi$						
Solution Namespace Solution	ts 🗘 YAML						
Name	kubelet-serving-cert-approver						
Annotations (2)							
+ Add annotation							
Resource Quota							
 A resource quota sets bounda assigning a quota for unrestric 	ies on the compute resources a namespace can use. It's good practice to set a quota for a namespace to manage resources effect ted access (not recommended).	ively. Alt	ternative	ely, you c	an disa	ible	
Resource assignment							
Registries							
O Define which registries can be	used by users who have access to this namespace.						
Select registries	No registries available. Head over to the registry view to define a container registry.						
Actions							
Update namespace Mark	as system						

Here you can see the details for this namespace. Under **Resource Quota**, enable the **Resource assignment** toggle. This will display two new options for Memory limit (MB) and CPU limit. For this resource, we're going to set these to values of 512 MB for memory and 0.5 for CPU. You can use the sliders to set these values or for memory, type it in manually.

Resource Quota					
 A resource quota set Alternatively, you car 	s boundaries n disable assi	on the compute resources a namespace can use. It's good practice to set a quota for a namespace to n gning a quota for unrestricted access (not recommended).	nanage res	ources effectiv	ely.
Resource assignment					
Resource Limits					
Memory limit (MB)	unlimited	512	4476	512	\$
CPU limit	_	0.5			5.85

With these values set, click the **Update namespace** button to save your changes, and click **Update** in the popup box to confirm. If you now return to the Namespaces list, you can see that the namespace is now listed as having quota enabled.

Namespaces System resources are hidden, this can be changed	d in the table settings	Q Search	n X	+ Add with form + Create from code
○ Name ↓↑	Status ↓↑	Quota ↓↑	Created J1	Actions
default	Active	-	2025-03-20 17:04:53	
kubelet-serving-cert-approver	Active	Enabled	2025-03-20 17:06:58	兴 Manage access
				Items per page 10 v

Let's now disable resource over-commit across the cluster. With your Kubernetes environment selected, expand **Cluster** in the left menu and click **Setup**, then scroll down to the **Resources and Metrics** section.

Resources and Metrics • By DISABLING resource over-commit (highly recommended), you can ONLY assign namespaces CPU and memory resources that are less (in aggregate) than the cluster total minus any system resource reservation. • By ENABLING resource over-commit, you can assign namespaces more resources than are physically available in the cluster. This may lead to unexpected deployment failures if there are insufficient resources to service demand. Allow resource over-commit • Enabling the metrics feature allows users to use horizontal pod autoscaling and to see container and node resource usage. This requires metrics server or prometheus to be running in your cluster. On any subsequent disabling of the feature, existing deployed applications with autoscaling will still autoscale (you would have to remove their autoscaler definitions to stop this). Enable features using the metrics API

Disable the **Allow resource over-commit** toggle. This will show a new option to specify the **system resource reservation**. This is a percentage of resource (CPU and memory) that Kubernetes will keep in reserve for system functionality. For now we'll leave this on the default of 20%.



When you're ready, click **Save configuration** to apply the changes.



OPA Gatekeeper

OPA Gatekeeper is a pod security policy application and lets us specify more fine-grained policies on what can and cannot be done on our cluster. For this workshop we'll look at some of the options that are available through OPA Gatekeeper.

With your Kubernetes environment selected, expand the **Cluster** menu on the left and select **Security Constraints**.

nvironments > talos-cluster > Security constraints					
Security constraints C	¢ ل	?	A admin ∨		
O Pod security constraints					
Enable pod security constraints					

Toggle on the **Enable pod security constraints** option, and a list of possible security constraints will be shown.



A full description of each option is available <u>in our documentation</u>. For now, let's enable **Restrict running privileged containers**.

Enable pod security constraints

Constraints

Controls whether any container in a pod can enable privileged mode

Restrict running privileged containers





Summary

In this lesson we adjusted the configuration of our Kubernetes cluster to make it more secure. We:

Restricted access to the default namespace to administrators only.
Set resource quotas on our existing namespace and disabled resource over-commit on the cluster.
Looked at the policy options that OPA Gatekeeper provides and restricted running privileged containers on the cluster.

In the next lesson we'll demonstrate how you can update your Kubernetes cluster from within Portainer.

Lesson 8 of 11

Upgrade the Kubernetes version

James Carppe

Deploying a Kubernetes cluster with Portainer and Omni gives us some powerful tools when managing the cluster. New versions of Kubernetes are released regularly containing fixes and new features, and you can easily upgrade your cluster's Kubernetes version with a few clicks.

In this lesson we will upgrade our Kubernetes cluster to the latest version.



Upgrading Kubernetes

Log into Portainer as the local administrator. Select your Kubernetes environment then expand the **Cluster** menu on the left and select **Details**.



On the **Cluster** page you will see information about your cluster including resource reservation and usage, as well as tools to manage your cluster. Scroll down to the **Omni cluster management** section.

eta feature - so far, Om	i cluster management functionality has only been tested in a limited set of scenar	rios.
Kubernetes (2)	v1.31.7 (current)	~
Update Kubernetes ve	rsion	
Talos version ③	v1.9.5 (current)	~

Here we can see the current Kubernetes and Talos versions. When we provisioned the cluster we chose an older version of Kubernetes (1.31.7) which is listed as current, and you will note the message indicating a new Kubernetes version is available. If we click the **Kubernetes version** dropdown we can see the options available to us.

(v1.31.7 (current)	~	
	v1.31.4	^	
	v1.31.5		
	v1.31.6		
	v1.31.7 (current)		
	v1.32.0		
	v1.32.1		
	v1.32.2		10000
	v1.32.3	~	

Scroll down in the list to find the most recent version of Kubernetes available (in my case, v1.32.3) and select it, then click **Update Kubernetes version**. You'll be shown a warning

that the cluster may become unavailable during this upgrade - since we're not running any workloads yet, this is fine. Click **Confirm** to proceed.



The upgrade process will now begin. Note that this process may take a few minutes to complete. You'll see status updates with detail on how the upgrade is proceeding.

Updating Kubernetes version pre-pulling images (4/7): pulled "registry.k8s.io/kube-proxy:v1.32.3"	

Once the upgrade completes, the current version listed will be updated to the version we selected.



We can confirm this has rolled out to all nodes in the cluster by scrolling down to the **Nodes** section and checking the listed version.

Nodes			Q Sea	rch	X S Reboo	t 🗑 Remove 🕇 🗛	dd nodes
Name ↓↑	Role ↓↑	Status ↓↑	$\textbf{CPU} \hspace{0.1 in} \downarrow \hspace{0.1 in} \uparrow$	Memory $\downarrow\uparrow$	Version $\downarrow\uparrow$	IP Address 🕸	Actions
talos01 api	Control plane	Ready	1.95	1.4 GB	v1.32.3	167.99.192.84	at 🗎
talos02	Worker	Ready	1.95	1.5 GB	v1.32.3	178.128.35.0	at 🗎
talos03	Worker	Ready	1.95	1.5 GB	v1.32.3	167.99.203.65	al 🗎
						ltems per	page 10 v

CONTINUE

2

Summary

In this (quick) lesson we successfully upgraded our Kubernetes cluster to the latest version in just a few clicks, without having to touch the command line at all.

In the next lesson we'll start configuring user-level access using Role-based Access Control (RBAC).

Lesson 9 of 11

Role-based Access Control (RBAC)

James Carppe

Role-based Access Control, or RBAC, is a way in which you can specify the access level of users within your organization to the resources available via Portainer. Users can be provided with roles at both an individual and group level, for environments themselves as well as resources within those environments.

In this lesson we will:



START

Giving our user environment access

To start with, we need to give our user access to the new Kubernetes environment. As you'll recall from when we configured OAuth, the GitHub user could not see any environments when logged in.

1

🗈 portainer io 🛛 🔍	Environments	
BUSINESS EDITION	Home S	☆ 〇 ペ jamescarppe ∨
Home Environment: / None selected	Information O You do not have access to any environment. Please contact your administrator.	
Administration	Environments	Q Search by name, group, 1 ×
	Click on an environment to manage Platfo Connecti Status Tags Grou	ups ∨ Agent ∨ Clear all Sort ∨ ↓↑
	No environments availa	ible. Items per page 10 🗸

Let's remedy that. Log into Portainer as the local administrator, then in the left menu expand **Environment-related** and select **Environments**.

۵	ක Home							
Envi	Environment: / None selected							
Adm	Administration							
ŝ	User-related	~						
⊜	Environment-related	^						
	Environments							
	Groups							
	Tags							

You should see our two environments listed - local (the local Portainer server environment) and the Kubernetes cluster we created (in my example, this is called talos-cluster).

Environments		Q Search	× Remove + Au	to onboarding + Add environment
Name ↓↑	Type ↓↑	URL↓↑	Group Name ↓↑	Actions
local	Docker	unix:///var/run/docker.sock	Unassigned	兴 Manage access
talos-cluster	✤ Kubernetes	https://68.183.41.75:9443	Unassigned	뽃 <u>Manage access</u>
				Items per page 10 v

Next to your Kubernetes environment, click the **Manage access** link. Here you can manage access to this environment. Scroll down to the **Access** section and you will see that there are no users or groups listed with access to the environment currently (the initial administrator has access to everything so is not listed here).

Access		Q Search ×	Remove Vpdate
Access tagged as inherited are inherit overridden. Access tagged as override are overrid	ed from the group access. They cannot be re ing the group	moved or modified at the environment leve	I but they can be
O Updating user access will require the afference of the afference of the second se	cted user(s) to logout and login for the changes to b	be taken into account.	
Name ↓↑	Type ↓↑	Role ↓↑	
	No items availabl	e.	
			Items per page 10 v

Scroll back up to the **Create access** section and from the **Select user(s) and/or team(s)** dropdown choose your GitHub user. In the **Role** dropdown, choose the Standard user role.

 (\mathbf{i})

The Standard user role has full access to resources that they create or are given access to, or that a member of their team has deployed. They do not have any administrative access to either the cluster or to Portainer itself. For more on roles and their respective access levels, <u>refer to our documentation</u>.

A Create access							
① Adding user access will	O Adding user access will require the affected user(s) to logout and login for the changes to be taken into account.						
Select user(s) and/or team(s)	A jamescarppe x X V						
Role	Standard user v						
+ Create access							

When you're ready, click **Create access**. The user will be given the **Standard user** role on the environment and will now appear in the **Access** list.

Access		Q Search	X Remove	✓ Update		
Access tagged as inherited are inherited from the group access. They cannot be removed or modified at the environment level but they can be overridden. Access tagged as override are overriding the group ① Updating user access will require the affected user(s) to logout and login for the changes to be taken into account.						
Name ↓↑	Type ↓↑	Role	∎ ↓↑			
jamescarppe	user	Stan	dard user 🗹 Edit			
			Items per p	age 10 v		



2

Creating a namespace for the user

Our user now has access to the Kubernetes cluster, but they don't have any namespaces to work in. As an administrator, let's create one for them.

Select your Kubernetes environment then select **Namespaces** from the left menu.



In the **Namespace list** you'll see our existing namespaces. Click on **Add with form** to create a new namespace.

Namespaces					
Namespace list \mathcal{C}				ф 🗘 📀 А	admin 🗸
 Namespaces O System resources are hidden, this can be chan 	Q Sear	rch	X Remove +	Add with form + Create from	n code
Name ↓↑	Status ↓↑	Quota ↓↑	Created J1	Actions	
default	Active	-	2025-03-20 17:04:53	😤 Manage access	

In the **Create a namespace** form, set a **Name** for the namespace. You will also note that we must set memory and CPU limits because we disabled over-commit earlier. Let's give this namespace a memory limit of 2048 MB and a 2 CPU limit.

Name*	ume* user-namespace					
Annotations (?)						
+ Add annotation						
Resource Quota						
 A resource quota sets Alternatively, you can 	boundaries on the compute resour disable assigning a quota for unres	rces a namespace can use. It's good practice to set a quota for a namespace to mana stricted access (not recommended).	age resources effectively.			
Resource assignment						
Resource Limits						
Memory limit (MB)	unlimited	2048	3069 2048 🗘			
	unlimited	2	418			
CPU limit	animited		4.10			

When you're ready, click the **Create namespace** button. The namespace will be created and you'll be returned to the namespace list, where you'll see your new namespace listed.

kubelet-serving-cert-approver	Active	Enabled	2025-03-20 17:06:58	은 Manage access
user-namespace	Active	Enabled	2025-03-21 14:43:18 by admin	은 Manage access

Now we need to give our user access to the namespace. Click on **Manage access** for the namespace we just created. Similar to the environment access management page, this page lists who has access to the namespace (currently nobody but administrators).

A Namespace access		Q Search X	
Name ↓↑	Type ↓↑		
	No items available.		
		Items per page 10	~

Scroll up to the **Create access** section and select your user. You will note that the environment access level for that user is displayed next to their username.

Preate access		
O Adding user access will represent the second s	equire the affected user(s) to logout and login for the changes to be taken into account.	
Select user(s) and/or team(s)	% jamescarppe Standard user ×	X v
+ Create access		

When you're ready, click **Create access**. The user will be given access to the namespace and will appear in the **Namespace access** list.

A Namespace access	Q Search X
○ Name ↓↑	Type ↓↑
jamescarppe	user
	Items per page 10 V

Now let's make sure that all worked, and log in as the GitHub user.

CONTINUE

3

Testing access

Log out of Portainer (click the arrow next to your username in the top right and select **Log out**) and log back in as the GitHub user by clicking **Login with GitHub** and providing the credentials if necessary.



Once you're logged in, you will see you now have access to one environment - our Kubernetes cluster. You'll also note that we don't have the full Administration menu options on the left, as we are a standard user.

BUSINESS EDITION	Environments Home C	声 🗘 ⑦ 将 jamescarppe v
Home Environment: / None selected	Environments	Q Search by name, group, 1 × طلا Kubeconfig
	Platform Connecti Status Tags Groups	Agent V Clear all Sort By V
Y NUMERIONS	talos-cluster ♥ heartbeat ∿ 2025-03-21 15:02:35 Kubernetes v1.32.3	S Browse snapshot </th
		Items per page 10 v

Select the environment and then select **Namespaces** in the left menu. In the list of namespaces, you'll see only the namespace we just created.

Namespaces			
Namespace list \mathcal{C}			p 🗘 🕜 🎗 jamescarppe 🗸
Solution Namespaces			Q Search ×
Name ↓↑	Status ↓↑	Quota ↓↑	Created ↓↑
user-namespace	Active	Enabled	2025-03-21 14:43:18 by admin
			Items per page 10 v



Summary

In this lesson, we:

Gave our user access to the Kubernetes environment as the Standard user role.
Created a namespace for our user and gave them access to it.
Logged in as our user and confirmed we only have access to what we specified.

In the next lesson we'll use this user to deploy an application on the Kubernetes cluster.

Lesson 10 of 11

Deploy an application



Our cluster is now configured and we have given our user access to it as a standard user. Now let's deploy an application to the cluster.

In this lesson we will:

Deploy an application using GitOps.

Make a change to the application configuration in Git.

Confirm that our update has automatically applied.

START

Deploying an application with GitOps

There are a number of methods you can use to deploy an application on a Kubernetes cluster. In this example, we'll demonstrate how you can use a Git repository containing a YAML manifest to deploy an application through Portainer.

Before we start, we're going to fork an example repository that we have created for this purpose. We want to create a fork as we are going to make changes to the repository contents in the next step.

You can find the example repository at the following URL:

https://github.com/portainerdemos/yachtops-demo

Go to the above repository in a web browser and click the **Fork** button.



Choose from the list of owners to fork to (if you have access to multiple owners) and edit the name of the repo if you so desire, then click **Create fork**.

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Owner *			Rono	sitory name	*
Required [leius ure	murkeu	wanun	usterisk ().	
Doguirod t	Fields are	marked	with an	actorick (*)	

Owner *		Repository name *	
🚯 jamescarppe 👻	/	yachtops-demo	
		yachtops-demo is available.	

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

A demonstration Kubernetes application for deployment via GitOps in Portainer.



(i) You are creating a fork in your personal account.

Create fork

Once the fork has been created you'll have a copy of the repo in your own GitHub account. We'll deploy our application from there.



Back in Portainer, log in as your GitHub user and select your Kubernetes environment. Next, click **Applications** in the left menu.



In the **Applications list**, you can see we have no applications currently deployed on the cluster that this user can access. You can use the **Namespace** dropdown to filter the list by namespace, or show all namespaces.

Applications		
Application list \mathcal{D}		🏚 🗘 🕜 A jamescarppe 🗸
♦ Applications 🗉 Stacks		
Applications	Q Search X	+ Add with form + Create from code
		♥ Namespace All namespaces >
Name ↓↑ Stack ↓↑	Namespace $\downarrow\uparrow$ Image $\downarrow\uparrow$ Application Type	e $\downarrow\uparrow$ Status Published Created $\downarrow\uparrow$
	No items available.	
		Items per page 10 v

To create our application, click on the **Create from code** button.

On the **Create from code** page, ensure that **Repository** is selected and that the namespace we created earlier is selected in **Namespaces**.

Deploy from						
Web editor		°	Repository	Custom Custom Custom	HELM Helm chart	
Deploy to						
Use namespace(s) specified from manifest ⑦						
Namespace	user-namespace				~	

As we're deploying from a Git repository, we need to specify the details. This repository is public, so we don't need to specify authentication. Provide the URL of your forked repository in the **Repository URL** field. The **Repository reference** should automatically populate, and enter deployment.yaml in the Manifest path field.

Git repository

Authentication		
O You can use the URL of a	git repository.	
Repository URL*	https://github.com/jamescarppe/yachtops-demo	G
Skip TLS Verification ⑦	e repository using the following syntax: branches with refs/heads/branch_name or tags with refs/tags/tag_name .	
Repository *	refs/heads/main	~
Indicate the path to the I	Manifest file from the root of your repository (requires a yaml, yml, json, or hcl file extension).	
Manifest path*	deployment.yaml	

At the bottom of the form, toggle on **GitOps updates**. Here we can configure how Portainer checks for updates to your deployment. Ensure **Polling** is selected as the mechanism and set the **Fetch interval** to 1m.

GitOps updates ③		
O Any changes to this stack which may cause service	k or applica	tion that have been made locally via Portainer or directly in the cluster will be overwritten by the git repository content n.
Mechanism	Polling	Webhook
Fetch interval* (?)	1m	

When you're ready, click **Deploy**. Portainer will retrieve the manifest YAML from the Git repository and deploy it on your environment. You should see it appear in the application list, initially with a status indicating 0 of 1 replicas.

Applications		
Application list C		声 🗘 ③ A jamescarppe 🗸
Applications	Q Search X Remove	+ Add with form + Create from code
		♥ Namespace All namespaces ✓
Name ↓↑ Stack ↓↑ Namespace ↓↑ Image ↓↑	Application Type 👘 Status	Published Created ↓↑
nginx-deployment - user-namespace nginx:1.14	.2 Deployment Replicated 0 / 1	No 2025-03-21 16:06:10 by jamescarppe
		Items per page 10 v

Once Kubernetes pulls the image and determines where to place it, this will update to 1 of 1 replicas. You can refresh the list with the refresh button in the top left.

Applications			
Application list \mathcal{G}		ф <u>с</u> О	s jamescarppe ∽
♀ Applications ≔ Stacks			
Applications	Q Search X	+ Add with form	Create from code
			All namespaces v
Name ↓↑ Stack ↓↑ Namespace ↓↑ Image ↓↑	Application Type ↓↑ Status	Published Created $\downarrow\uparrow$	
nginx-deployment - user-namespace nginx:1.14	.2 Deployment • Replicated 1 / 1	No 2025-03-21	16:06:10 by jamescarppe
			Items per page 10 V



Making a change to the application

The application has deployed successfully to your cluster. But let's say we wanted to make a change. With the way we've configured this application, all we need to do is update the manifest in the Git repository and Portainer will automatically pick up the change and update the deployment.

In your fork of the Git repository, make a change to the deployment.yaml file. You can do this through the GitHub web interface by selecting the file and clicking on the pencil icon.

yachtops-demo / deployment.yaml 🛛 🖓	
nortainerdemos Create deployment.yaml	e5eb52b · 17 minutes ago 🕚 History
	Edit this file
Code Blame 26 lines (26 loc) · 476 Bytes	Raw [] 坐 🖉 👻 🖸
1 apiVersion: apps/v1	

For this example, let's imagine we wanted to increase the number of replicas of our container to 2. On line 9 of the manifest file, change replicas: 1 to replicas: 2.

4	<pre>name: nginx-deployment</pre>
5	spec:
6	selector:
7	<pre>matchLabels:</pre>
8	app: nginx
9	replicas: 2
10	template:
11	metadata:
4.0	

With this changed, click the **Commit changes** button and click **Commit changes** again in the popup window.



Your changes will be committed to the repository.

When we created this application in Portainer, we set the polling interval for GitOps to one minute. This means that every minute, Portainer will check the repository for any changes. If it finds any, it will retrieve those changes and apply them to the application on the cluster. If you return to the **Application list** in Portainer once a minute has passed, you should see the status has changed for the application:

Application list C		ሱ 🗘 💿 ۲ jamescarppe א
Applications	Q Search X	+ Add with form + Create from code
		▼ Namespace All namespaces ×
Name ↓↑ Stack ↓↑ Namespace ↓↑ Image ↓	↑ Application Type ↓↑ Status	Published Created
nginx-deployment - user-namespace nginx:1.14	I.2 Deployment Replicated 2 / 2	No 2025-03-21 16:06:10 by jamescarppe
		Items per page 10 v

If you click the name of the application then go to the **Events** tab, you can see the events that occurred to update the replica count.

		S Events	↔ YAML			
3 Events			Q Search ×			
Date ↓1	Kind ↓↑	Type ↓↑	Message 11			
2025-03-21 16:16:16	Pod	Normal	Started container nginx			
2025-03-21 16:16:15	Pod	Normal	Successfully pulled image "nginx:1.14.2" in 4.381s (4.381s including waiting). Image size: 44710204 bytes.			
2025-03-21 16:16:15	Pod	Normal	Created container: nginx			
2025-03-21 16:16:11	Pod	Normal	Pulling image "nginx:1.14.2"			
2025-03-21 16:16:10	Pod	Normal	Successfully assigned user-namespace/nginx-deployment-59cf44d54c-dgjdj to talos02			
2025-03-21 16:16:10	Deployment	Normal	Scaled up replica set nginx-deployment-59cf44d54c from 1 to 2			
2025-03-21 16:06:11	Pod	Normal	Container image "nginx:1.14.2" already present on machine			
2025-03-21 16:06:11	Pod	Normal	Created container: nginx			
2025-03-21 16:06:11	Pod	Normal	Started container nginx			
2025-03-21 16:06:10	Pod	Normal	Successfully assigned user-namespace/nginx-deployment-59cf44d54c-8rcgg to talos03			
2025-03-21 16:06:10	Deployment	Normal	Scaled up replica set nginx-deployment-59cf44d54c from 0 to 1			
			Items per page 25 v			

CONTINUE

3

Summary

In this lesson we:

Forked a Git repository containing an application manifest and deployed the application from our forked repo to our Kubernetes cluster.

1		_	ר	
			L	
	_		h	

Made a change to the manifest in the Git repository.

Watched the change get automatically picked up by Portainer and applied to the application on the cluster.

Next we'll summarize everything we've covered in this course.
Lesson 11 of 11





Through the lessons in this course, we have:

Provisioned a management server and deployed Portainer Business Edition.
Configured our Portainer server with external authentication via GitHub OAuth.
Created an Omni account and service account, and configured those credentials in Portainer.
Created three Talos droplets in Digital Ocean for our Kubernetes cluster.
Created a three node Kubernetes cluster in Portainer via Omni.
Configured security settings on the cluster through OPA Gatekeeper and set up resource quotas.
Upgraded the version of Kubernetes to the latest release.

Configured Role-based Access Control for our GitHub user and created a namespace for them to use.
Logged in as our GitHub user and deployed an application using GitOps.
Made a change to the application manifest in the Git repository and watched the application automatically update with the change.

Congratulations! You've managed to get a real, working Portainer and Kubernetes cluster up and running with an application deployed in under two hours!