Portainer Best Practice Installation Guide

This course will guide you through a best practice deployment of Portainer on your production infrastructure, covering recommended configurations, environment architecture, and security considerations and setups.

Introduction

This guide is intended to walk you through our recommended Portainer configuration for use in a production environment. We will cover how to choose and prepare your management environment, how to deploy Portainer Server to that environment and the initial setup. We will then work through our best practice steps for configuring and securing both your management environment and your connected environments, including user authentication and access configuration, role assignments and other considerations.

By the end of this guide you should have a Portainer configuration ready to go for production use.

Requirements and prerequisites

This guide makes certain assumptions about both your environments and your knowledge. Specifically, we assume that you have root / admin access to the environments you're working on, as you will need this to deploy Portainer. We also assume you have functional knowledge of the containerization platforms that you are using, whether that be Docker, Swarm, Kubernetes or a combination of the above, as there are certain prerequisites and configuration steps that should be undertaken to prepare your environments ahead of deploying Portainer.

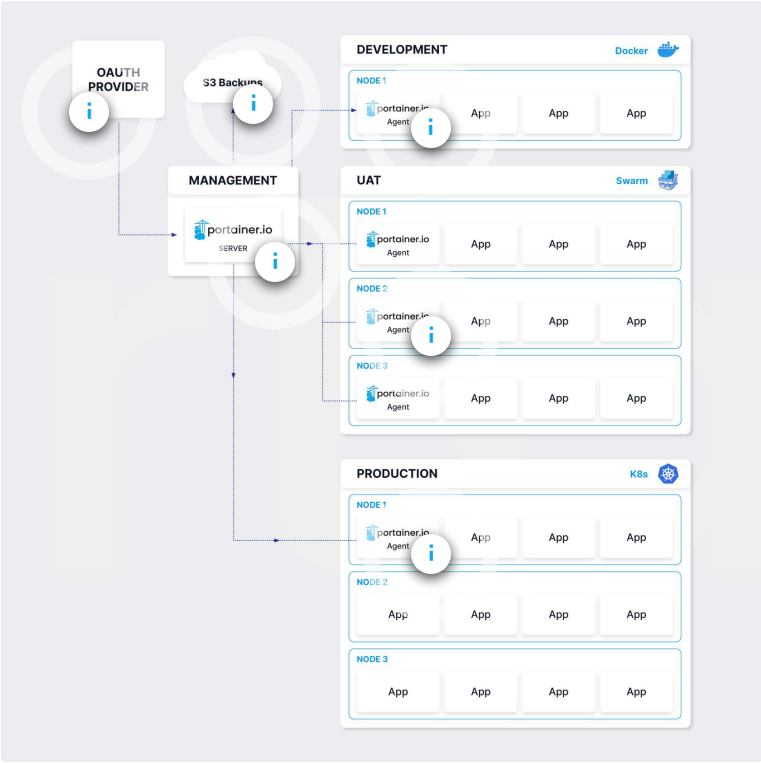
While you can jump around within this guide as you need, this guide also assumes that you will be deploying your setup as you work through it, and sections will assume that you have followed the sections that came before.

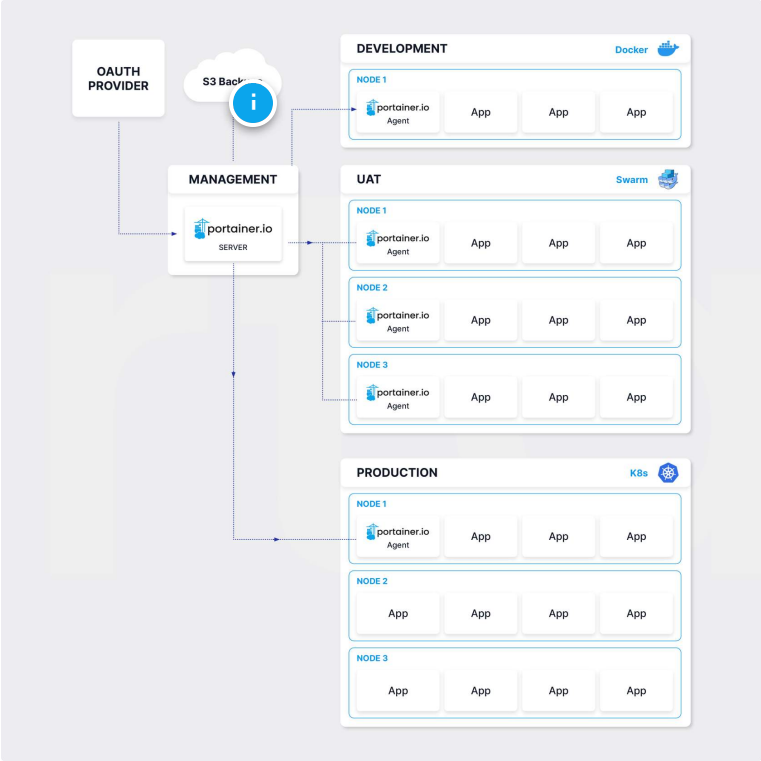
One size fits most

There is no "one true way" of configuring Portainer for production; this guide suggests a recommended, best practice approach to your deployment and may not work for every organization. However, the steps covered here should be considered carefully when deploying in production to ensure you have a secure and performant setup going forward.

A best practice example

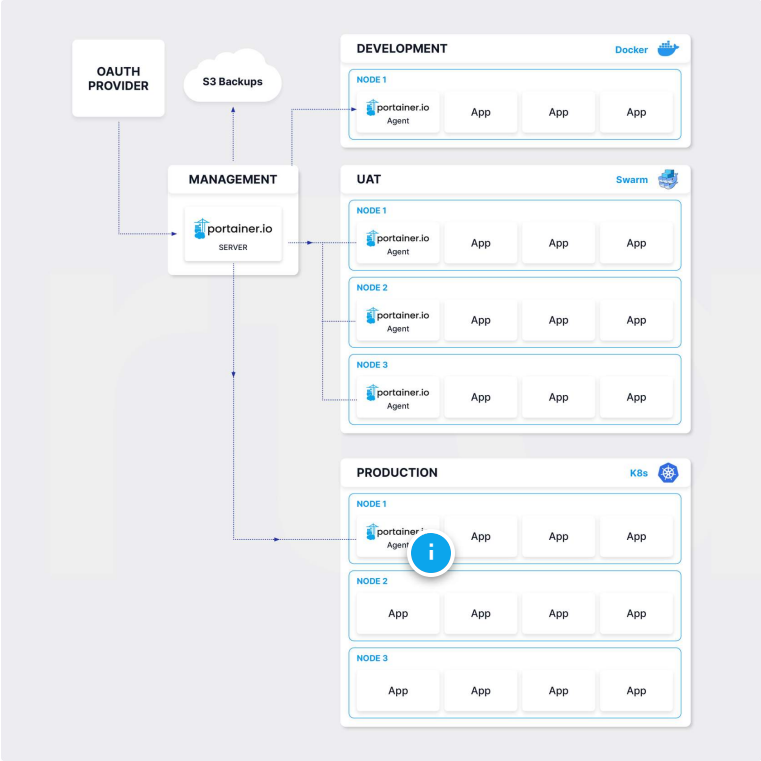
Before we start getting into the detail of your deployment, let's look at an example production deployment of Portainer. In this example, we have Portainer managing multiple environments from one management interface on a management cluster. Authentication is provided by an external OAuth provider, the Portainer UI is secured with SSL and IP whitelisting for access. The remote environments are managed with the Portainer Agent and Edge Agent where relevant, with firewalls configured for the necessary access only, and the Portainer Server configuration is backed up to an S3 bucket daily.





S3 Backups

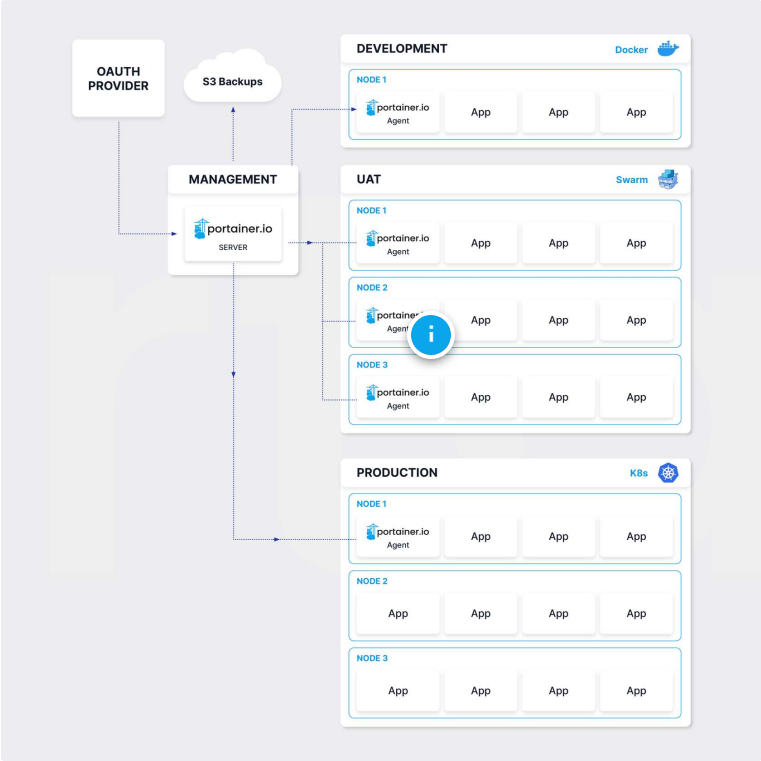
The Portainer configuration is backed up automatically to an S3 bucket. Portainer supports both AWS S3 buckets and S3-compatible providers such as MinIO and Backblaze.



Portainer Agent

The Portainer Agent runs on the managed environment, providing a conduit for the Portainer Server to act upon the environment.

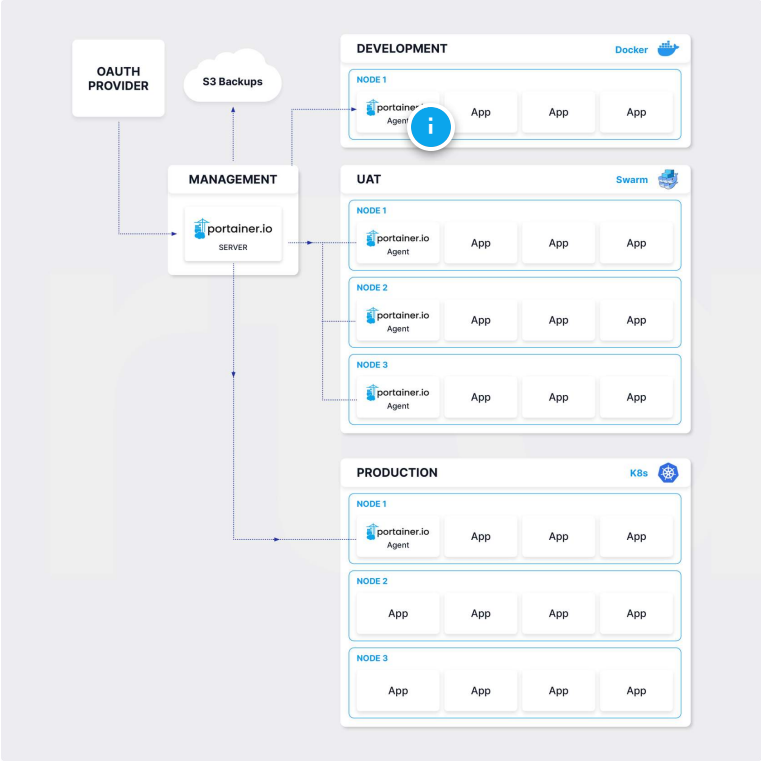
For Kubernetes environments, the Agent runs on one of the nodes in the cluster, leveraging the Kubernetes API for access.



Portainer Agent

The Portainer Agent runs on the managed environment, providing a conduit for the Portainer Server to act upon the environment.

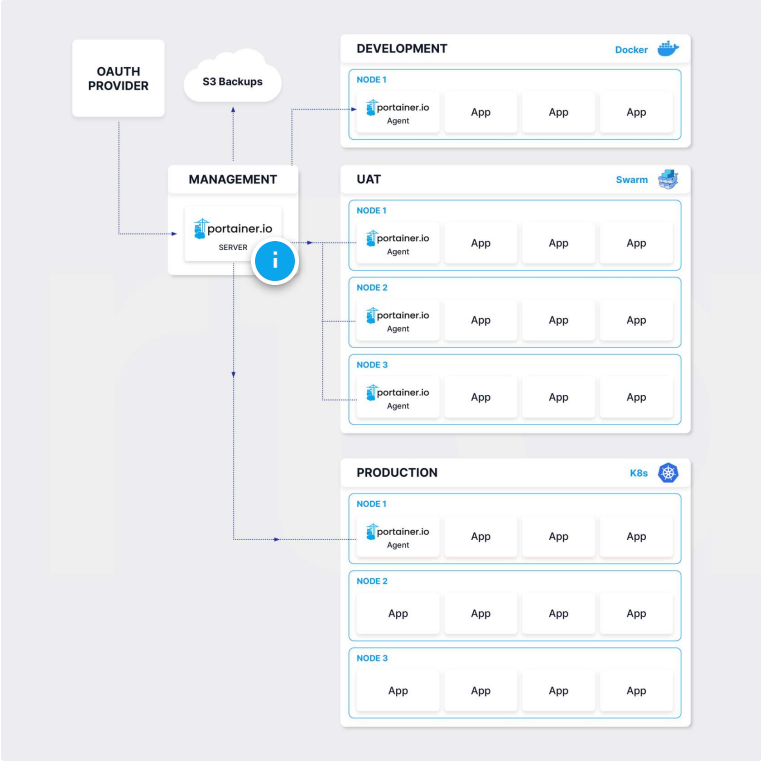
For Docker Swarm environments, the Agent runs as a global service across all nodes in the cluster.



Portainer Agent

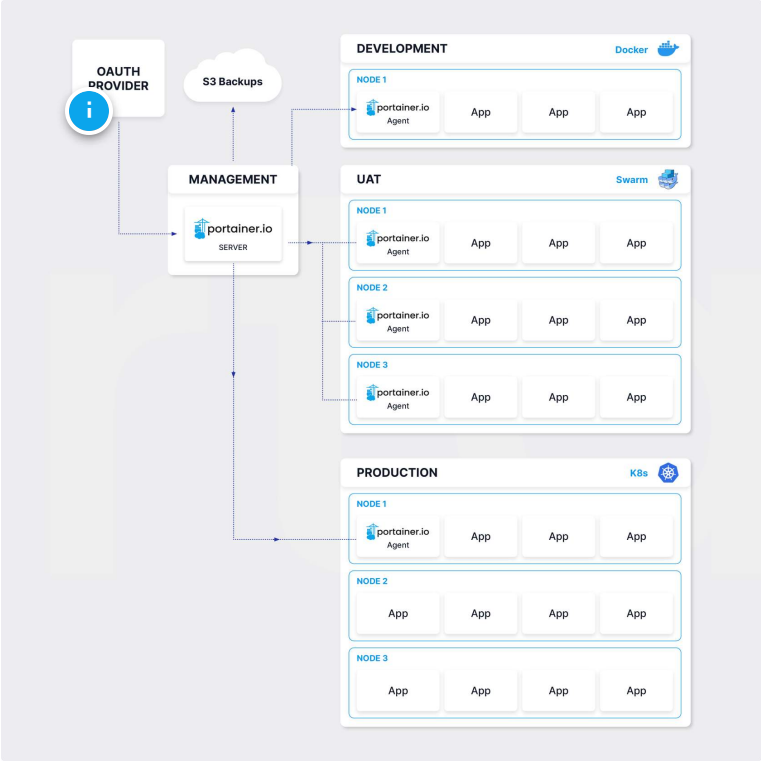
The Portainer Agent runs on the managed environment, providing a conduit for the Portainer Server to act upon the environment.

For Docker Standalone environments, the Agent runs as a container on the node alongside your applications.



Management environment

The Portainer Server is running on an independent management environment from the workloads, ensuring that access is always available to the managed environments even if one of them is offline.



OAuth Provider

Authentication is provided by an external OAuth provider such as Microsoft Azure AD, Google or GitHub, allowing for centralized user management across the organization.

Choosing your architecture

When building a production-ready Portainer deployment, the first crucial decision you will need to make is how to architect your management infrastructure. There are two basic options:

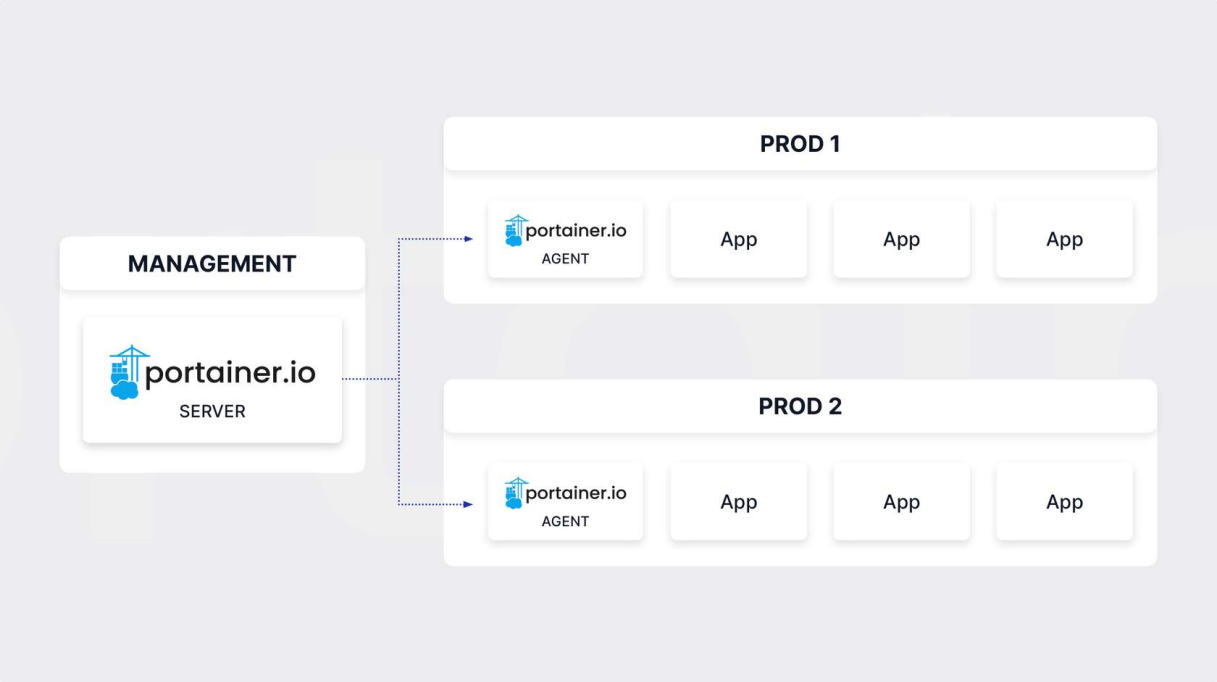
1. Deploy the Portainer Server on a dedicated management environment (recommended).
2. Deploy the Portainer Server within one of the environments it will also manage.

In this lesson we will cover each of the above options in detail, and look at the pros and cons of each approach.

START

Step 1

Recommended: A dedicated management environment



While it is possible to run Portainer alongside your workload in a containerized environment, for production deployments we highly recommend setting up a dedicated management environment that runs just the Portainer Server and no actual production workloads. Doing so means the management system is able to manage all your application environments without being affected by them.

For example, if you were running the Portainer Server on a node of your application environment and that node went down, you wouldn't be able to use Portainer to manage that node or any other nodes or environments. With a separate node, you can manage your deployments without worrying about this scenario.

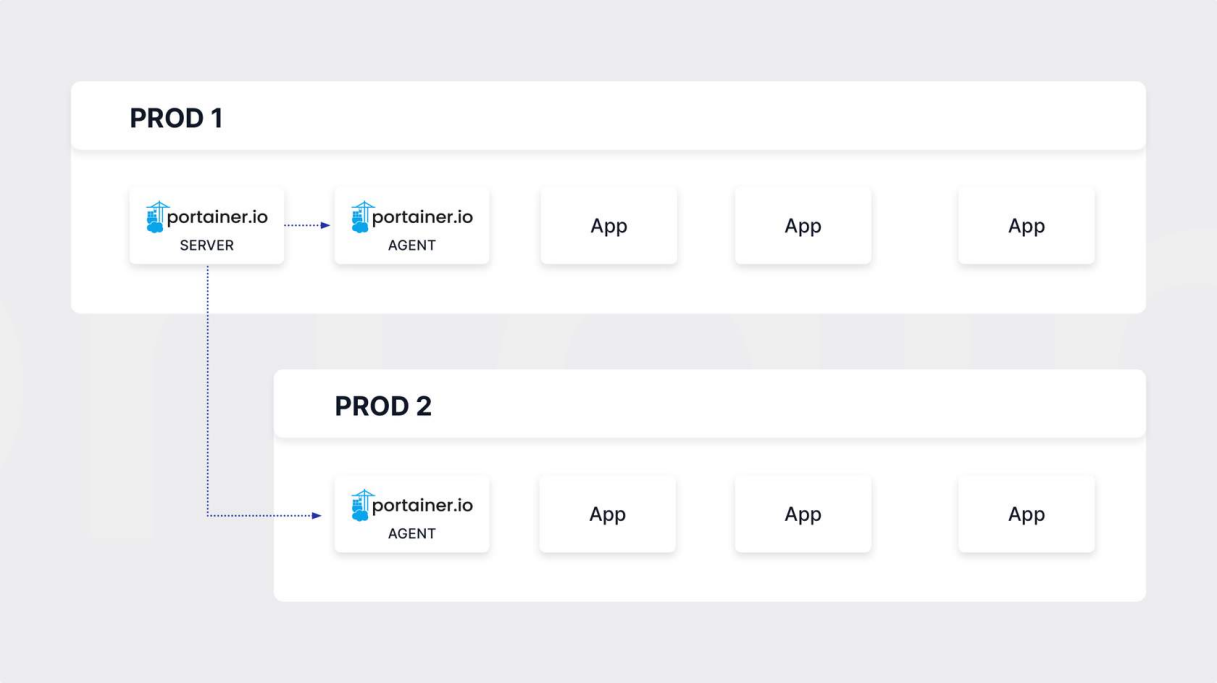
By the same token, if your management environment went down, it wouldn't affect your production environments in this configuration. Your applications would continue to work unaffected, and once you are able to bring your management environment back online you could pick things right up where you left off.

There are also potential security improvements with this approach. With a dedicated management environment, your Portainer users will never be directly connecting to your production environments in any way; all activity will be routed through Portainer. Management access would only need to be configured on the management environment, with the Portainer Agent handling the communication between the production environments and the management interface.

The primary drawback with this configuration is that it requires you to set up a separate management environment, which may incur extra costs and potential management and maintenance overhead. In some situations this may not be desirable or feasible, in which case the Portainer Server can be run on an existing environment.

Step 2

Alternative: Install Portainer Server within one of the environments it will also manage



If running a dedicated management environment is not an option, you can instead run Portainer Server alongside your workflow on an environment it will also manage. This is our recommended approach for development or testing scenarios as it uses less resources to do so than running a dedicated environment, but there are drawbacks to consider with this approach.

Firstly, when running alongside your workflow, there is the potential for your access to be hampered by your deployments. If your node that is running the Portainer Server is under heavy load from your application, this may limit the ability to use Portainer to manage it. While we build Portainer to be as lean and performant as possible, it does require some resources to function and if those resources are unavailable, performance may be impacted.

In addition, if the node running the Portainer Server goes down, you will have no access to Portainer to help troubleshoot the issue until that node comes back up. In a development environment this is less of a concern, but could be a significant issue for production deployments.

In order for your users to manage your environments with Portainer, they will need to be able to log into Portainer. This is generally done through our web interface, access to which would need to be configured on your firewall. Opening a web management port on a production environment may be a security risk, and would not be required if the Portainer Server was running on a separate environment.

CONTINUE

Summary

In this lesson we have covered the two options to consider when setting up your Portainer architecture: dedicated or shared. For production environments we recommend the use of a dedicated management environment to provide:

- Additional resiliency in the case of node outages
- Separation of access between management and production
- Minimal impact from production workloads on the ability to manage environments

A dedicated management environment may mean:

- Additional costs from running a dedicated management node
- Additional maintenance overhead for the management node

If in your situation the drawbacks outweigh the benefits, you can instead run the Portainer Server within one of the environments it will also manage.

In future lessons, we will cover the requirements and installation process for both options.

Preparing your management environment

Now that we have decided on our management environment architecture, we can start preparing that environment for the Portainer deployment. While Portainer is designed to be run (almost) anywhere, there are some basic prerequisites for deployment as well as some extra points to consider for a production-ready deployment.

In this lesson we will cover the basic requirements of the Portainer Server, discuss the options when it comes to orchestrator choice, as well as cover the important persistent storage consideration.

START

1

Basic requirements

The Portainer Server runs as a container within a containerized environment, allowing for easier configuration and maintenance. At present, Portainer Server supports running on the following containerization platforms:

- Docker Standalone

- Docker Swarm
- Kubernetes

The Portainer Server installation process does not install or configure the containerization platform for you, so this will need to be preinstalled and configured beforehand. You can find the most up to date detail on the versions of the platforms we support in our documentation.

Regardless of your chosen platform, to install Portainer Server you will need the following:

- Root, administrator or cluster admin access to the environment where you want to install the Portainer Server instance.
- Persistent storage on (or connected to) the environment for the Portainer Server's database and other files.

We will cover persistent storage in more detail later in this lesson.

CONTINUE

2

Docker vs Swarm vs Managed Kubernetes

This primarily applies to the dedicated management environment architecture. If you are running Portainer Server

within one of the environments it will also manage, this decision has likely been made for you already.

Choosing the platform for your management environment can boil down to the following:

- ## Familiarity

  How familiar are you and your team with the platform? While Portainer is intended to help you manage your environments, some familiarity with the basics of the chosen platform is helpful. For example, if your team has no experience with Kubernetes then it may not be the best choice for your management environment.

- ## Availability

  What kind of resources do you have available to run your management environment? A management environment is generally going to be significantly smaller (in terms of resourcing) than your production environments, but you may want to consider the number of users that will need to be accessing the management environment, where those users are connecting from, and how many environments you intend to manage from the Portainer Server instance. This can affect the amount of resources you will need for your management nodes.

Outside of these points, there are also platform-specific items to consider.

Docker Standalone

A Docker Standalone environment is one of the simplest configurations for a management environment, and is easy to set up and maintain. However, by it's nature Docker Standalone is not multi-node, and while Portainer Server is not multi-node compatible (more on this later) it may be worth considering this when deciding on a management platform. On the other hand, as a purely management node, the redundancy that a multi-node environment provides *may* be less of a concern than it would be for a production workload, but do consider what the impact would be on your workflow (for example, if Portainer is to be a part of your CI/CD pipeline) when making a decision on redundancy.

Docker Swarm

With Docker Swarm, the Portainer Server is deployed as a service on the Swarm cluster (by default restricted to manager nodes). Within a sufficiently sized and configured cluster (3 or more manager nodes), this does provide additional fault tolerance as compared to a Docker Standalone deployment, but does come with it's own considerations.

- While the Portainer Server instance is not fully multi-node compatible (more on this later), in a multiple node environment it is conceivable that the Portainer Server container could be running on different physical nodes. As such, the persistent storage used by Portainer needs to be accessible from all nodes where Portainer could potentially be running. We will cover persistent storage in more detail later in the lesson.
- A common issue when running Docker Swarm is a misconfigured overlay network, preventing the services within the swarm from communicating between nodes. Ensure that the overlay networking is configured and working correctly, including DNS resolution between nodes.

Managed Kubernetes

On a Kubernetes environment, the Portainer Server container is deployed as a single replica pod in a Portainer-specific namespace, and exposed via either ingress, node port or load balancer as chosen during deployment. We highly recommend using a managed Kubernetes cluster rather than a self-deployed and managed cluster due to the complexity of the initial setup and configuration of Kubernetes, though a non-managed cluster is also an option.

Your Kubernetes cluster will need to have:

- The role-based access control (RBAC) addon installed and configured. This is to allow for Portainer's multi-user capabilities and permissions.
- A default StorageClass configured. This is so the Portainer Server can configure persistent storage for the database and configuration. Bear in mind that in some Kubernetes deployments, the default StorageClass simply creates hostPath volumes, which could be problematic if the Portainer Server pod is redeployed on a different node to where the volume was created.
- If you want to use metrics within Portainer (highly recommended), the metrics server addon will also need to be installed.

CONTINUE

3

Portainer and multiple nodes

At present, the Portainer Server is not fully multi-node compatible in that it can only run on a single node at any given time. In multi-node environments with correctly configured failover and persistent storage, the Portainer Server is fault tolerant, but there is no current capability for the Portainer Server to be running on more than one node at once. We are however actively looking at how we can improve this in the future.

CONTINUE

4

Persistent storage

The Portainer Server requires a persistent storage location in order to store the database it uses for the configuration, as well as any supplemental files such as compose files, YAML manifests and the like. Without this storage, the Portainer configuration will be lost and you would be unable to use Portainer to manage your environments. As such, we recommend taking extra care with your choice of storage mechanism. We also recommend doing this at the infrastructure level, in order to reduce the possibility of compatibility issues between your containerization platform and your choice of storage.

In many cases, an off-node storage location is preferable to on-node storage. A common example of this is the use of NFS mounts on the node from an external NFS server, though there are other options available. This can (along with backups, which will be covered in a later lesson) ensure that your Portainer configuration remains intact in case of a failure in your management environment.

Whichever option you choose, when you are deploying on a multi-node cluster, ensure that the storage is available on *all* the nodes and in the *same* way (with the same paths and permissions) on all nodes.

CONTINUE

Summary

In this lesson we've discussed what you need to think about when preparing your management environment, whether that be a dedicated management environment or within one of the environments it will also manage.

You should now:

- [ ] Know which containerization platform you will be using (Docker Standalone, Docker Swarm or Kubernetes), along with any platform-specific requirements you might have.
- [ ] Have root or administrator access to your containerization platform, in order to begin the installation process.
- [ ] Have knowledge on how Portainer uses persistent storage, and a plan on how to implement this on your platform.
- [ ] Understand the limitations around multi-node support in Portainer and how that may affect your deployment decisions.

You're now ready to install Portainer! The next lesson will get you started with this process.

Deploying Portainer Server

Congratulations, you are now ready to install Portainer! Previous lessons have covered the things you need to think about when it comes to planning your Portainer installation, and now that we've done that we're ready to move on to the real thing.

Below you will find links to our installation instructions in our documentation for each of our supported management environments - Docker Standalone, Docker Swarm and Kubernetes. Click the button then choose your platform, and follow the provided instructions.

Docker Standalone
If you are installing Portainer on a Docker Standalone environment, start here.

DOCKER STANDALONE

Docker Swarm
For Docker Swarm environments, start here.

DOCKER SWARM

Kubernetes
For Kubernetes environments (managed or otherwise), start here.

KUBERNETES

Once you have completed the installation, you're ready to move onto the initial setup of Portainer.

Initial setup

Now that you have deployed the Portainer Server, there are a few initial setup steps to complete:

1. Setting a secure administrator user and password.
2. Adding your Portainer license.
3. Connecting to your local environment.

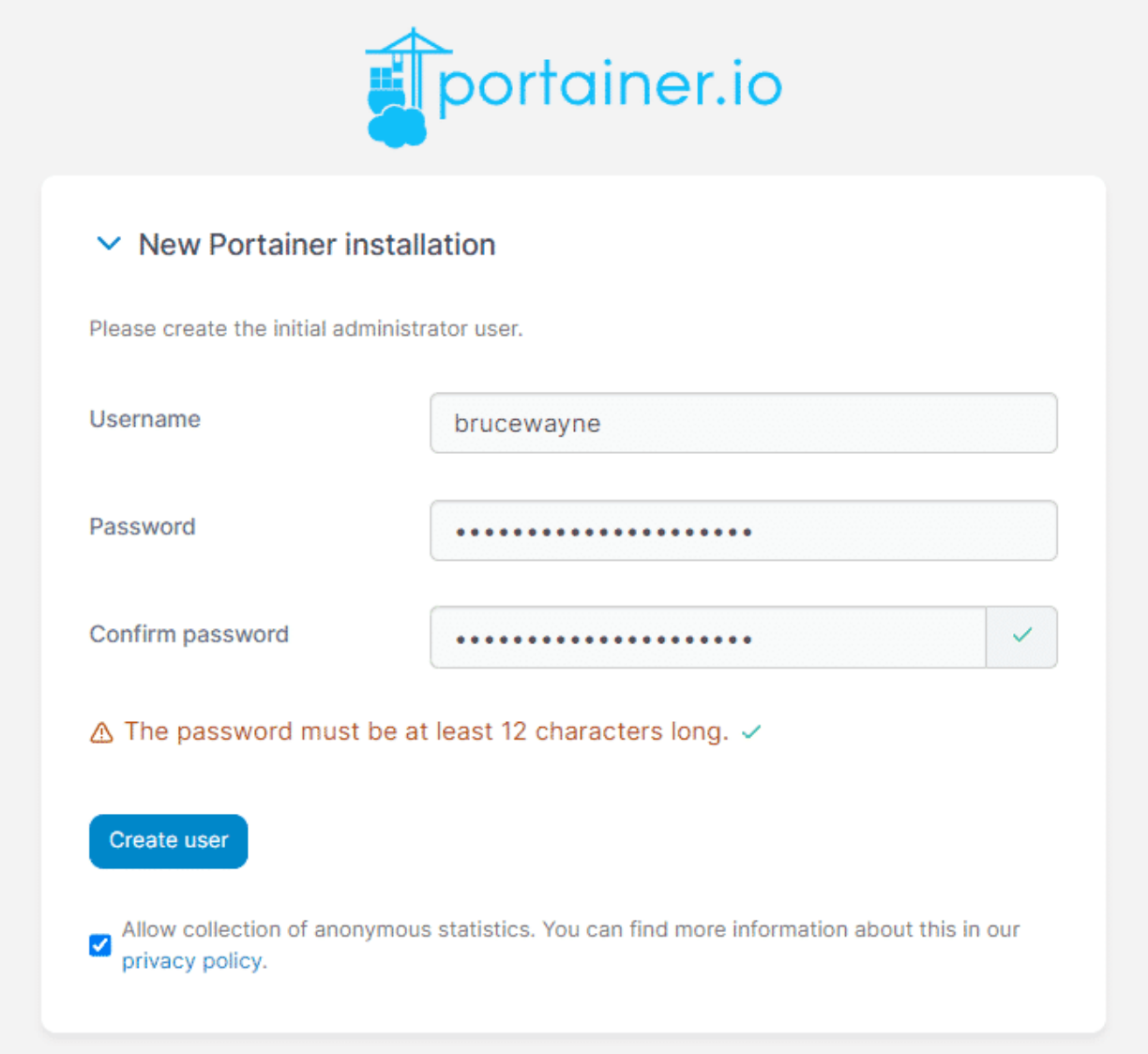
This lesson covers these initial setup steps.

START

Setting a secure administrator

When you first connect to the Portainer UI after installation completes, you will be asked to set up an initial administrator user. This is an important first step, even if you intend to use an external authentication provider, as it can provide a fallback access method if your external auth provider is unavailable or has configuration issues.

Once the Portainer Server container is started, there is a 5 minute period within which you will need to create your initial administrator user. If a user is not created within those 5 minutes, the container will stop responding and will need to be restarted.



We recommend choosing a unique username other than the default of "admin". While this is optional, having a non-default username is another layer of security.

Enter a strong, unique password to use for the administrator account. The minimum length on a fresh installation of Portainer is 12 characters, and the longer the better. We recommend the use of either a passphrase or a complex mix of uppercase, lowercase, numbers and symbols.

Enabling or disabling the collection of statistics

We use a tool called Matomo to collect anonymous information about how Portainer is used. We recommend enabling this option so we can make improvements based on usage, but this is entirely up to you. For more about what we do with the information we collect, read our privacy policy.

If you change your mind later, you can easily update this option under **Settings** in the Portainer UI.

CONTINUE

2

Adding a licence

Once an administrator user has been added, the next step in the initial setup is to add your Portainer license. You should have received this when signing up - if you haven't yet signed up you can click the "Don't have a license?" link to get a trial license or contact our sales team for purchasing. If you have already signed up for a license but haven't received it, reach out to our success team for help.

Portainer licenses are limited to a set number of nodes, so when purchasing a license you will need to be mindful of how many nodes are in your infrastructure.

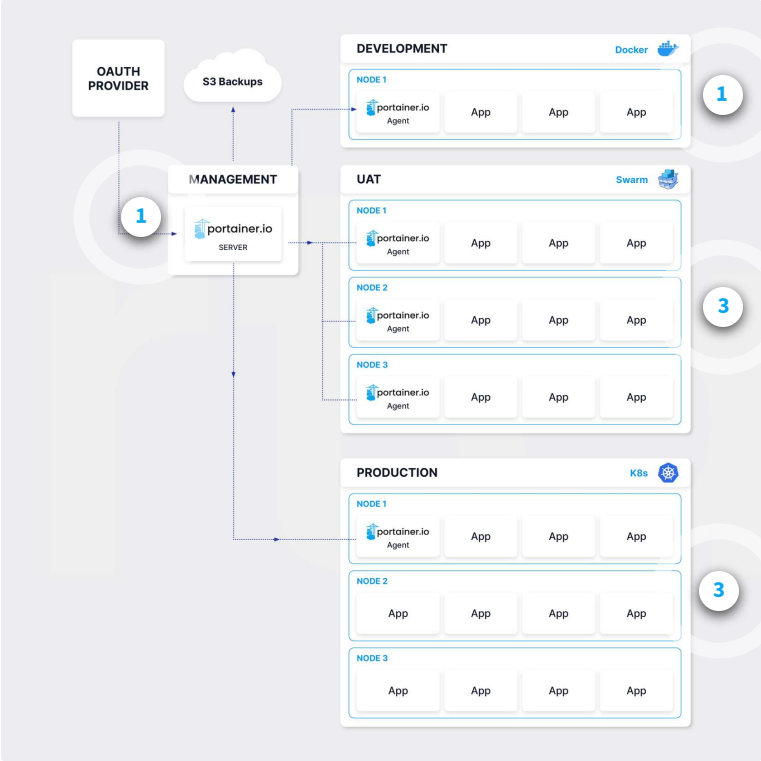
What counts as a node?

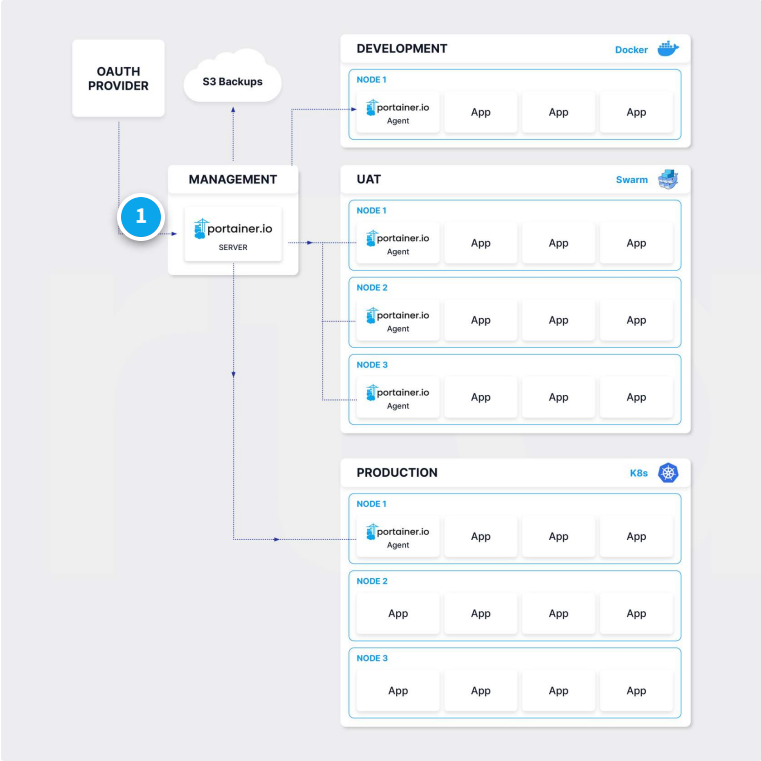
As we describe in our knowledge base:

> "A "node" can be simply described as a "server" (whether this is an actual physical server, a VM, a Raspberry Pi, your desktop or laptop, an industrial computer, or an embedded compute device) that is capable of running containers (via Docker, Kubernetes or another orchestrator) which is either running the Portainer Server or is under the management of a Portainer Server installation. "

In production environments, you are likely to be working with clustered environments such as Docker Swarm or Kubernetes, where there are multiple nodes within each environment. Each of these nodes would count as a node for licensing purposes. It is also important to consider the Portainer Server environment when calculating your node count, as it too is running Portainer.

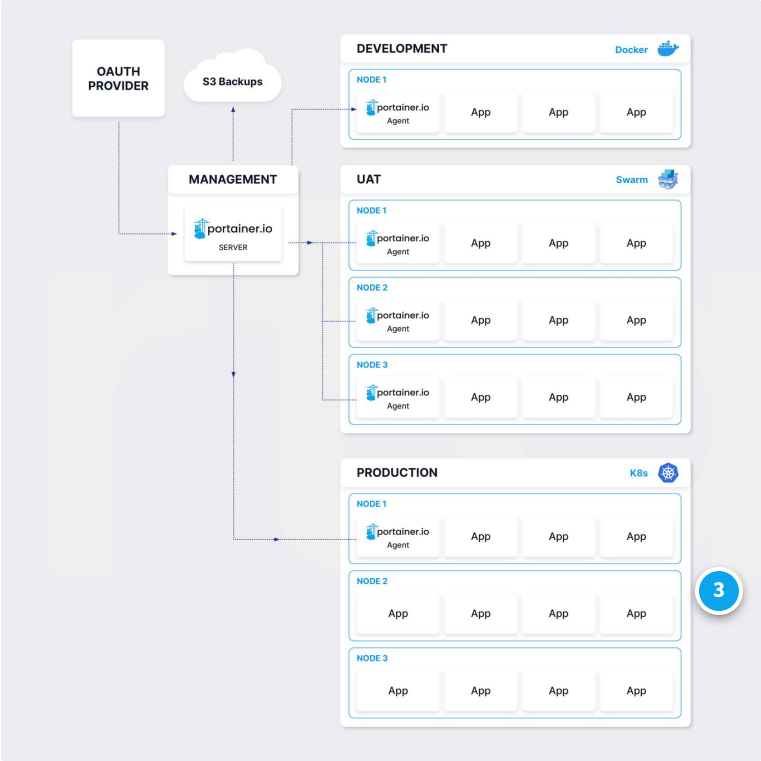
Let's look at our example setup from the beginning of the course:





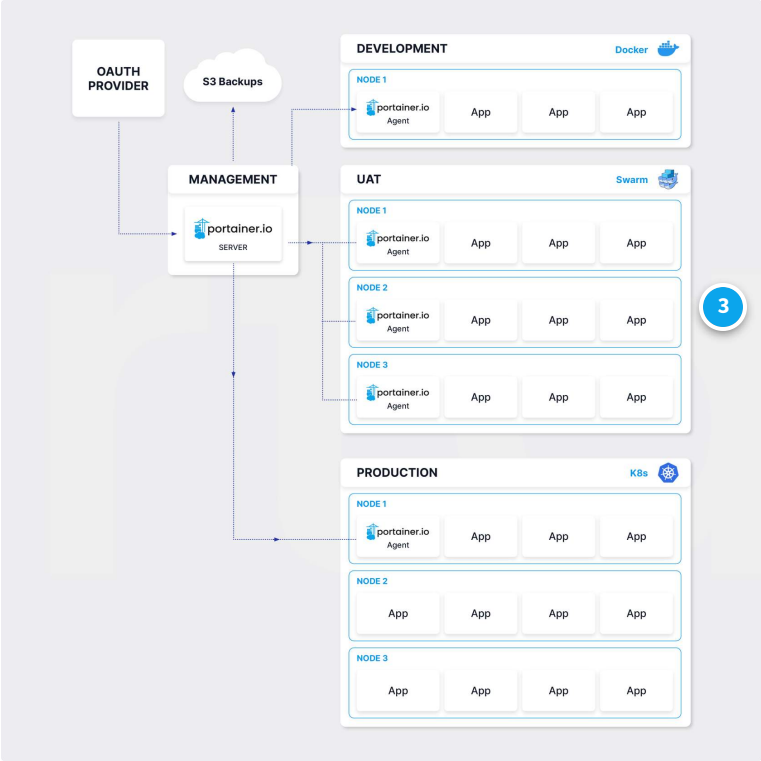
Management environment (1 node)

The management environment is a single Docker Standalone environment in this example, so counts as 1 node for licensing.



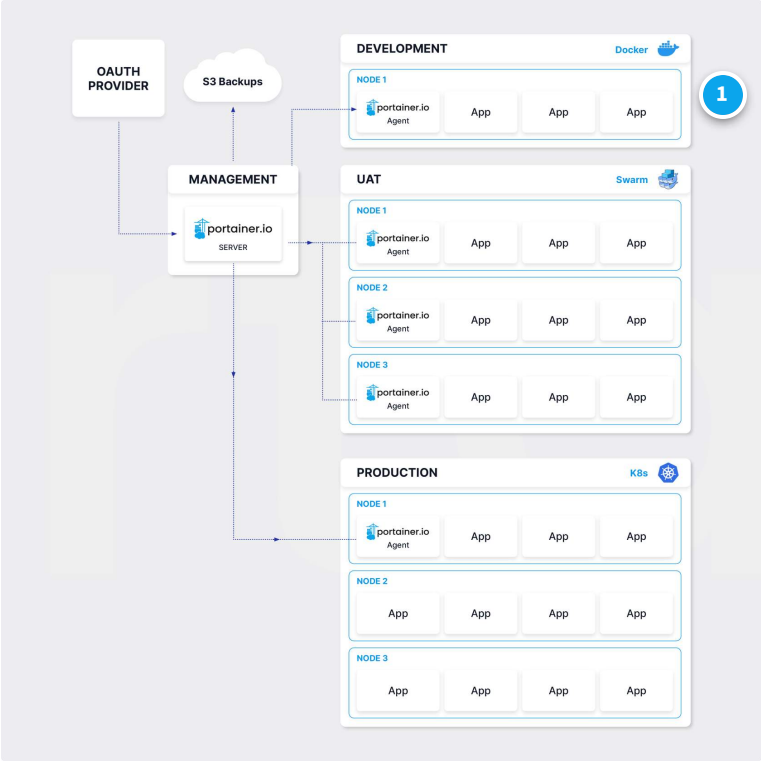
Production environment (3 nodes)

The production environment's Kubernetes cluster is made up of 3 nodes, so counts as 3 nodes for licensing.



UAT environment (3 nodes)

There are 3 nodes in the UAT Docker Swarm cluster, which counts as 3 nodes for licensing.



Development environment (1 node)

The development node in this example consists of a single Docker Standalone installation, so counts as 1 node.

In this example, we have a management environment that is running a single node Docker Standalone environment to host the Portainer Server container. We are using Portainer to manage three other environments; our Development, UAT and Production clusters, each with one, three, and three nodes respectively. Combining this with our single node management cluster, we get 1 + 1 + 3 + 3 nodes, resulting in 8 nodes total for licensing purposes.

Your particular setup is likely to differ from the above example, but it should give you a good place to start from. You can find a more detailed run-down of how nodes pertain to licenses in our knowledge base.

Planning ahead vs starting small

If you're just starting out with your containerized environment, you may want to start with a smaller license count and work your way up. You can always purchase additional node licenses from us if you need to expand. Alternatively, if you know the size of your infrastructure and want to prepare, that is doable as well. You don't need to consume all of your node licenses in your initial setup.

If you're at all unsure about how many nodes you need for your infrastructure or if you have any questions on licensing, you can reach out to our team for help.

CONTINUE

3

Connecting to your environments

Once you have added your license, you will be taken to the Portainer UI and presented with the initial setup environment wizard. The Portainer installation will have detected your local environment and preconfigured it for you, and you now have the option of adding additional environments to manage.

For a production deployment we recommend selecting **Get Started** here and proceeding to the main Portainer interface, where we will make some additional configuration changes before adding additional environments to manage.

CONTINUE

Summary

In this lesson we've completed the initial setup of our Portainer Server installation, including:

- [ ] Creating an initial administrator user with a strong and unique password.
- [ ] Adding the license key (and signing up for one if you hadn't already).
- [ ] Connecting with the local environment in preparation for further configuration.

While technically you can start using Portainer right away (especially if you're running Portainer on an existing environment and it is the only environment you intend to manage) we highly recommend continuing to the next lessons where we will discuss how to secure and configure your Portainer setup, add new environments, set up external authentication, registries and backups, and more.

Securing your installation

Congratulations! Your Portainer Server instance is now installed and running on your environment. While you're technically good to go now, we recommend following a few more steps to get you running smoothly. The first of these is to ensure your Portainer Server installation is secure.

In this lesson we will cover how to secure your Portainer Server installation, including:

1. Adding your SSL certificate and chain to Portainer to help secure access to the Portainer interface.
2. Configuring your firewall and adding access restrictions to the Portainer server interface and tunnel ports.
3. Configuring regular automatic backups of the Portainer Server configuration to an S3 (or S3-compatible) bucket.

START

1

Add SSL certificates

When Portainer is installed, by default we generate self-signed SSL certificates to encrypt access to the web interface. While self-signed certificates provide the same level of encryption as trusted certificates, they do generally throw a warning in your browser. As such, we recommend replacing the self-signed certificates with trusted certificates signed by a recognized certificate authority.

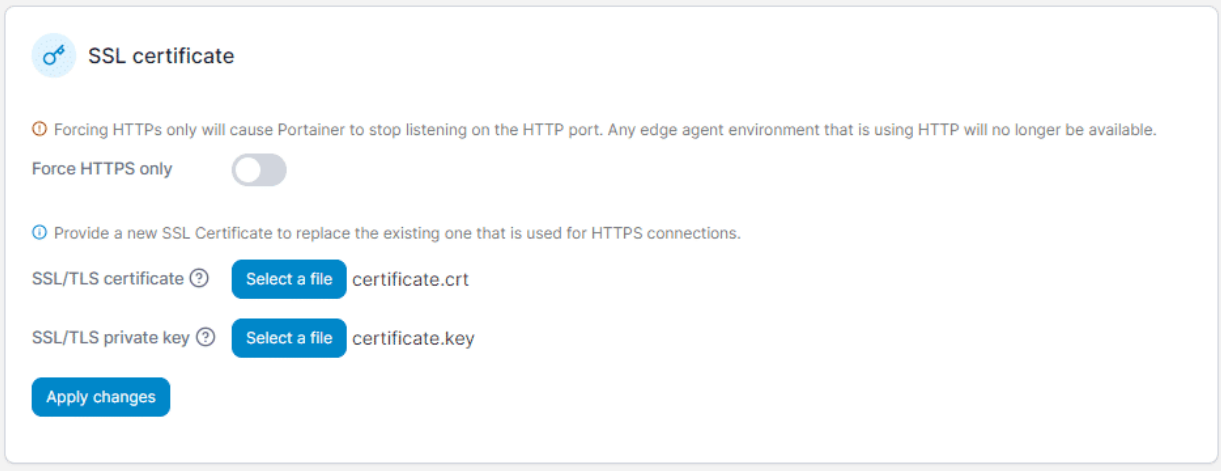
Selecting your certificate provider and purchasing or generating your certificates is outside of the scope of this lesson, but should be relatively straightforward. Once you have obtained your certificate, you will need to ensure you have:

- [ ] The certificate file in PEM format.
- [ ] The private key file for the above certificate, also in PEM format.
- [ ] Any intermediate certificates provided by your certificate authority (CA).

The intermediate certificates, sometimes referred to as chain certificates or a CA bundle, are needed in order to provide the signing path between your certificate and the root certificates trusted by web browsers. Your CA should supply these, but if not you can use What's My Chain Cert to generate these for you.

Once you have all three components, the first step is to merge your certificate file and intermediate certificates into a single file for use by Portainer. The certificate itself should come first in the file, followed by the intermediate certificate bundle.

Now we're ready to update the certificate in Portainer. Log in to your Portainer installation as an admin user, then select **Settings** in the left hand menu and scroll down to the **SSL certificate** section. In this section you'll see a toggle labeled **Force HTTPS only** as well as two buttons labeled **Select a file** - one for the SSL/TLS certificate and one for the private key. Leave the Force HTTPS only toggle off for the time being - we will talk about this later. For now, click the first **Select a file** button and browse to your merged certificate and intermediate cert bundle. The click the second **Select a file** button and browse to your private key file. Once you've done this, you should see something similar to the following (your filenames may differ):



When you're happy with your selections, click **Apply changes** to update your SSL configuration.

This will restart the Portainer service to apply the change, and will require you to log back in to Portainer.

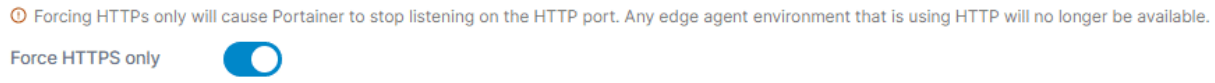
Test your changes!

Before we go any further, let's make sure that we've configured SSL correctly. Log out of Portainer and reload the login page in your web browser. You should see the browser using your newly uploaded SSL certificate. Log back in to test that it works as expected.

If you do run into any errors here you can log in via the HTTP port if you have it exposed. If not, this knowledge base article should help you enable HTTP access.

Optional: Force HTTPS only

Once you have confirmed your SSL configuration is working, you may want to enable the **Force HTTPS only** toggle we saw earlier. This stops Portainer from listening on the HTTP port entirely, requiring that HTTPS be used for all access. We only recommend doing this when you have fully tested HTTPS access, and where all agents and remote systems connecting to the Portainer interface will use HTTPS.



If you have enabled this and run into issues accessing Portainer, this knowledge base article provides a way to re-enable HTTP access.

CONTINUE

Firewalling and access restriction

Another important element of securing your Portainer installation is restricting access to the Portainer Server instance. We generally recommend doing this through the use of a firewall or a similar access control method. Your implementation process will depend highly on your particular IT infrastructure, but the following should give you a good place to start.

Port requirements

The Portainer Server needs the following incoming ports open for access:

- [ ] TCP port 9443 (or 30779 for Kubernetes with NodePort) for the web UI, Edge Agent communication, and the Portainer API.
- [ ] TCP port 8000 (or 30776 for Kubernetes with NodePort) for the Edge Agent tunnel. Depending on your configuration, this may be optional.

For outgoing access, the Portainer Server may use:

- [ ] TCP port 80 and 443 for accessing external resources via HTTP or HTTPS respectively. These could be remote Git repositories containing your deployments, application templates, Helm charts, image registries, external authentication providers, S3 backups, and the like.
- [ ] Any other specific port required for access to an external resource as covered above. For example, you may run your remote registry on port 5000.

Portainer Business Edition also checks in with our license server (license.portainer.io) periodically over HTTPS. However, if Portainer cannot reach the license server (for example if it was running in an airgapped environment with no internet access) it will continue to work as normal.

IP access restriction

In addition to restricting open ports, we highly recommend implementing IP whitelisting on your Portainer Server installation in order to lock down access to only the IP addresses that you know need to connect. This may include:

- [ ] Users that need to log into the Portainer web interface.
- [ ] Remote Edge Agents that will connect to Portainer.
- [ ] Systems that will invoke webhooks on your deployments (for example deployment automation through CI/CD pipelines).
- [ ] Systems that will interact with the Portainer API.

Remember to test that everything is working as expected after making changes to access, as you don't want to accidentally lock yourself out!

CONTINUE

3

Configure backups to S3

Portainer supports backing up its configuration both manually to a downloadable file as well as automatically to a remote S3 or S3-compatible bucket. This backup covers the configuration of Portainer itself - users, roles, environments, registries, stack files - but notably does **not** include containers, volumes, or other deployments. We recommend configuring separate backup systems for your application data.

The Portainer backup can be used to restore a Portainer Server setup in the case of a failure of your management environment, getting you back up and running as quickly as possible. As such, we highly recommend configuring this to run automatically.

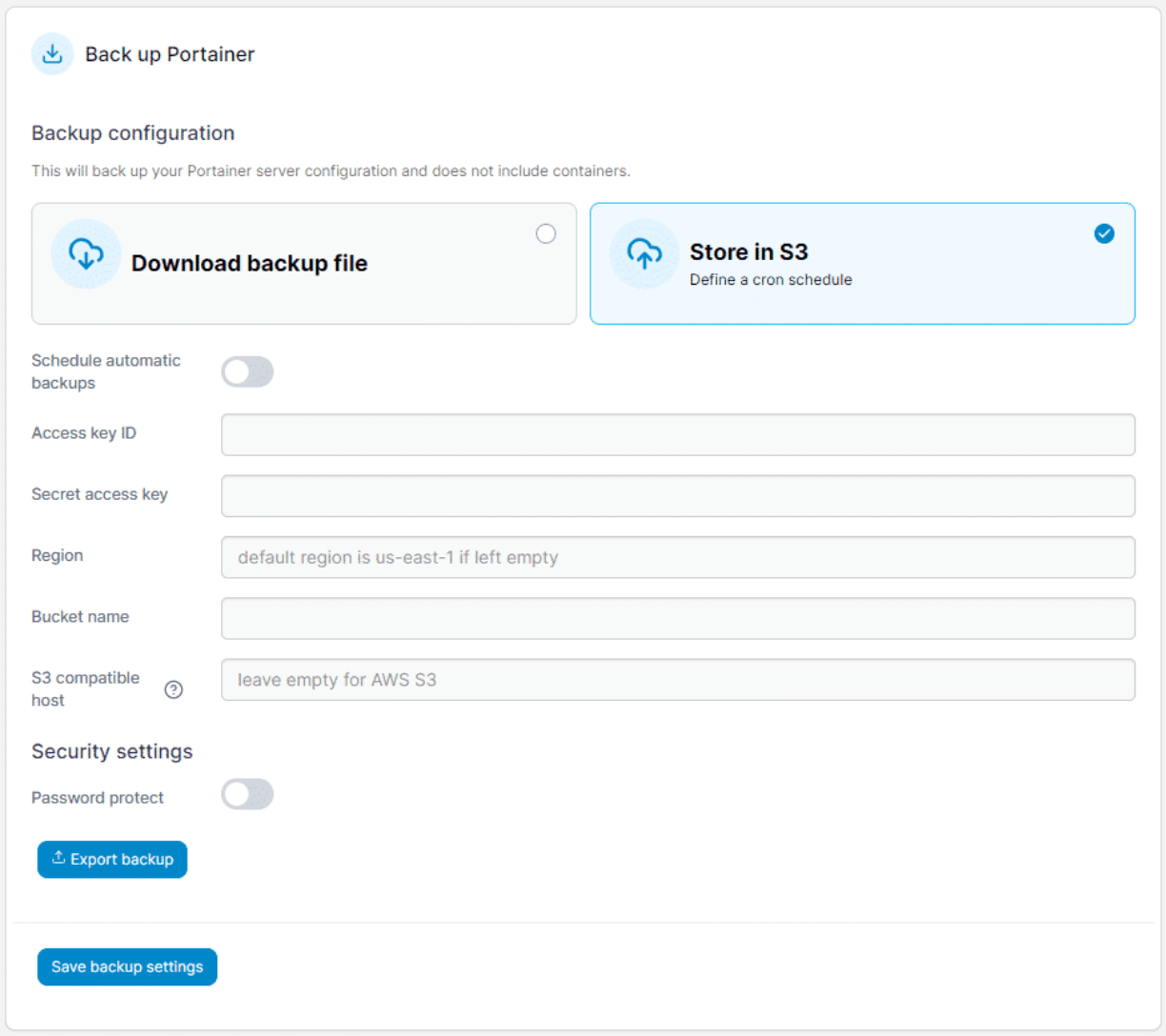
To configure backups to a S3 or compatible bucket, you will need:

- [ ] A bucket to store your backups. Portainer supports AWS S3 as well as S3-compatible systems (for example, MinIO and Backblaze B2).
- [ ] An access key for your bucket. This consists of two parts: an access key ID and a secret key.

We recommend creating an access key specifically for Portainer backups, with permissions for only the bucket you intend to use for your backups.

> **Portainer does not do any cycling or rotation of backups. We recommend periodically checking on the backups in your S3 bucket and manually removing old backups as required, as depending on your configurations these backups can get fairly sizeable.**

Once you have your bucket details and access key, you're ready to set up your backups. Log into Portainer as an admin user, then select **Settings** in the left hand menu and scroll down to the section labeled **Back up Portainer**. Since we want to back up to a S3 (or S3-compatible) bucket, select the **Store in S3** option.



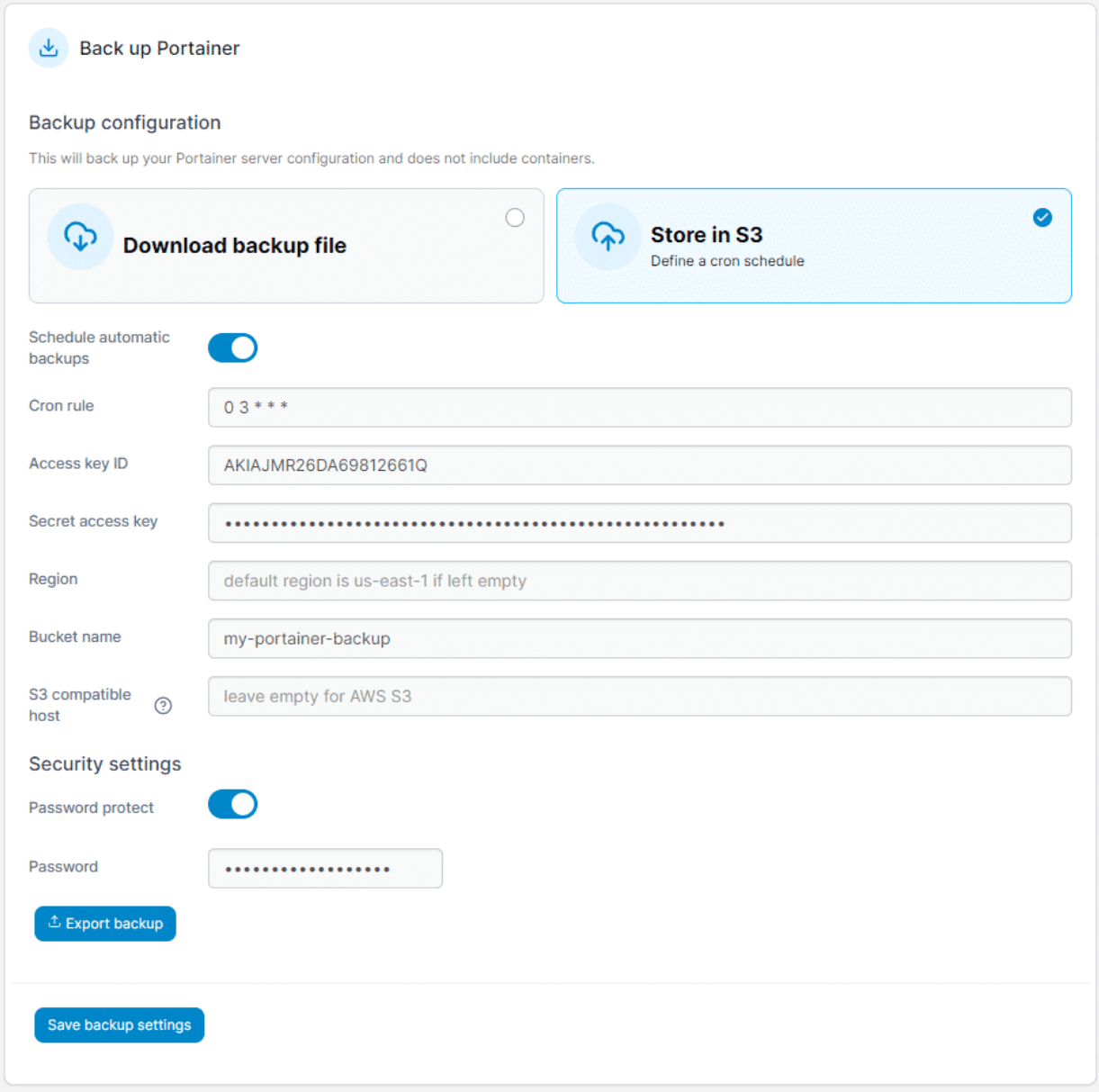
First we want to configure the schedule. We recommend running daily backups, though feel free to adjust this to suit. Toggle on the Schedule automatic backups option and a new field labeled Cron rule will appear. Enter your backup schedule in cron format in the box. For example, if you wanted to run your backup at 3am every day, you would enter:

```
0 3 * * *
```

Next we'll provide the access details for our bucket. Fill in the **Access key ID** and **Secret access key** fields with your access key and corresponding secret key respectively. If you

are using AWS S3 or your S3 provider requires it, enter the region that your bucket is in under **Region** and the name of your bucket in **Bucket name**. If you're using a S3 compatible host rather than AWS S3, provide the hostname (and port if relevant) in **S3 compatible host**.

You can also optionally set a password on your backups by toggling on **Password protect** and entering a **Password**. You will need this password to restore your backup, so make sure you keep it safe.



When you've got everything set up as desired, click **Save backup settings** to apply the configuration.

CONTINUE

Summary

In this lesson we've worked through some important first steps in securing your Portainer instance. We have:

- [ ] Replaced the default self-signed SSL certificate with a trusted certificate.
- [ ] Configured access restrictions for the ports that Portainer uses as well as whitelisting IP access to Portainer.
- [ ] Set up regular automatic backups of the Portainer configuration.

In our next lesson we will cover setting up your users and teams with access to Portainer.

Configuring access

With Portainer Server secured, we can now move on to configuring access for your users and teams to Portainer. How you do this will depend greatly on your specific systems and requirements, but there are some points to consider as you prepare and configure.

In this lesson we will cover:

1. The authentication methods that Portainer supports, and which we recommend.
2. Setting up teams for your users.
3. Configuring your external authentication provider in Portainer (recommended); or:
4. Creating users in Portainer for use with internal authentication

START

1

Authentication methods

There are two basic modes supported by Portainer for user authentication:

1. External authentication using a third-party authentication provider (highly recommended).
2. Internal authentication with users and groups stored within the Portainer database.

EXTERNAL AUTH PROVIDER | INTERNAL AUTHENTICATION

Using an external authentication provider is our recommended approach for production deployments. It is highly likely that your organization already has an authentication provider, so leveraging that instead of maintaining a separate system has benefits from an admin perspective, if nothing else.

Portainer currently supports the following authentication methods:

- LDAP
- Microsoft Active Directory
- OAuth

EXTERNAL AUTH PROVIDER | INTERNAL AUTHENTICATION

If you don't have an existing authentication provider that you wish to connect with Portainer, you can alternatively use Portainer's built-in user and group system. This will mean managing your users, passwords and groups directly in Portainer rather than a centralized location across your organization.

It is also important to note that Portainer's internal authentication is more limited in features as compared with external auth providers. For example, it does not support two-factor authentication, password expiration policies or the ability to lock user accounts without removing them. Unless you have no other choice, we highly recommend avoiding using internal authentication in production deployments.

CONTINUE

2

Set up teams

Regardless of whether you intend to use external or internal authentication, the first step is to set up your teams within Portainer. Teams are used to configure access to environments and the resources within, such as containers, services and volumes.

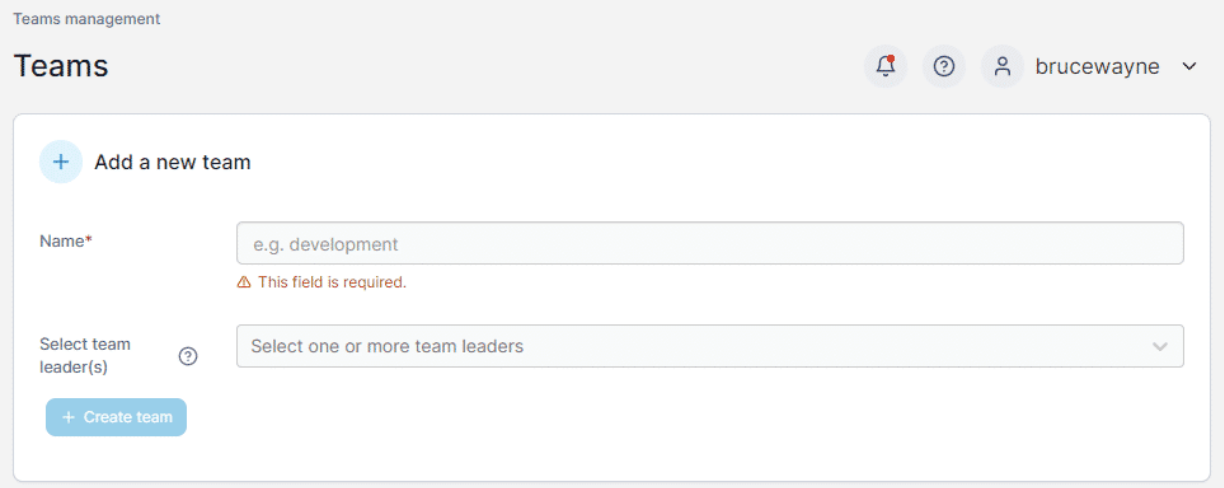
How you configure your teams will be highly specific to your needs and your organizational structure. An example of how you could configure your teams might be:

- **Development** - For access to your development and staging environments, but not to production environments.
- **Support** - For limited access to your production environment, and no access to development or staging.
- **QA** - For access to your staging environments to complete pre-release testing.
- **Tech Leads** - For access to all environments in order to facilitate deployments to production and provide assistance throughout the software chain.

Teams can be added, removed and adjusted at a later date as required, and users can be members of multiple teams, so don't feel you need to get this exactly right immediately. However it is important that you think about who needs access and at what levels early on in the deployment process. In particular, if you are planning to use an external auth provider, for automatic team-to-group mapping to occur your team names must match the groups you use in your external auth provider.

Creating a team

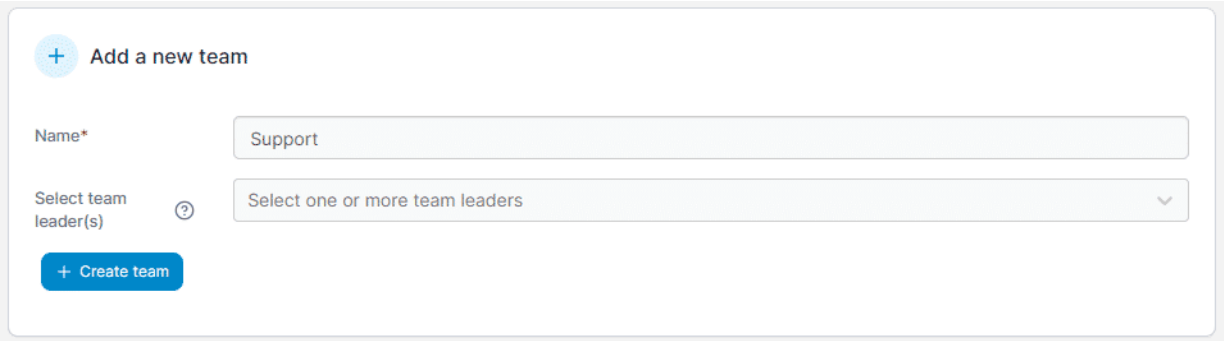
Once you've decided on the teams you want to create, log into Portainer as an administrator and select **Users**, then **Teams** in the left hand menu.



Here you'll see a form to add a new team as well as a list of existing teams. Fill in the **Name** field with the name of the team you want to create.

> **If you intend to use an external authentication provider and the groups that exist within that provider, be sure to match your team names to those group names.**

For now, don't select any team leaders - we'll come back to that option later. Click the **Create team** button to create the team, and repeat for as many teams as you need.



Now that we have our teams set up, we can move on to setting up authentication itself.

What about users?

At this stage, we just need to worry about setting our teams up. We will cover the adding of users to your teams at a later point in the lesson.

CONTINUE

3

Recommended: Set up external auth provider

As mentioned above, Portainer supports external authentication via LDAP, Microsoft Active Directory and OAuth. As long as your auth provider supports one of these methods, it should be compatible with Portainer. For OAuth, as well as the ability to specify a custom OAuth configuration, we have pre-configured provider templates for Microsoft, Google and GitHub available.

Automatic user provisioning

When setting up your external authentication provider you will be given the option of enabling automatic user provisioning. In most cases we recommend enabling this, as it will automatically create a user configuration within Portainer for users that log in successfully, letting that user interact with the Portainer system. If automatic user

provisioning is disabled, users will need to be created manually within Portainer that match the username provided by your auth provider in order for them to be able to login.

You can find more about automatic user provisioning in the Portainer documentation.

LDAP / AD: User and Group search configurations

For LDAP and Active Directory configurations, you will be asked to configure **user search**. This lets you specify a subset of users from your auth provider who will have access to Portainer. You can search your entire organization or limit your search to specific OUs or containers, and from there filter your results to suit your needs.

You can also optionally configure **group search**. This allows you to automatically place users, based on their LDAP / AD group, into an identically named team within Portainer. We highly recommend the use of this feature to simplify the organization of access within Portainer, and is why we created teams before setting up authentication.

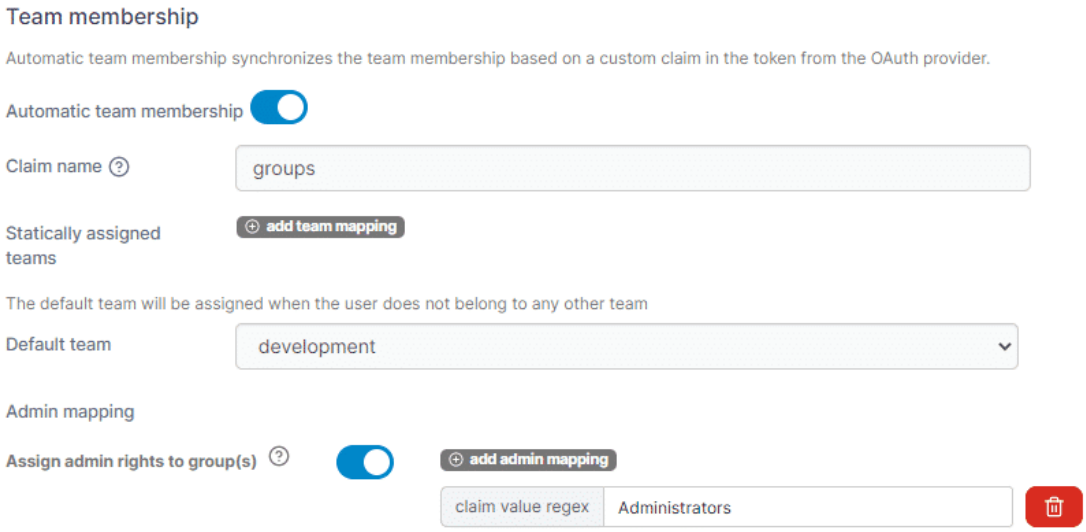
You can select the groups you choose to automate in this method through the use of filters. In addition, you can also configure groups to automatically become Portainer admins if so desired. As always, we recommend caution when applying automatic admin rights.

You can find more about configuring user search and group search in the Portainer documentation.

OAuth: Team membership

For OAuth providers, you can choose to enable **Automatic team membership** to automatically add OAuth users to Portainer teams based on the **Claim name** you configure. Claim names will be matched with teams or you can manually link a claim name (using regex) with Portainer teams under the **Statically assigned teams** option. You can

also define a **Default team** for users who don't belong to any other team, as well as enable the automatic assignment of admin rights to specified groups if needed.



We generally recommend enabling the Automatic team membership option and configuring matching, as it retains the OAuth provider as the "source of truth" for users. If the option is disabled you will need to manually create matching local users first before they can log in with OAuth. Automatic assignment of admin rights should be done with care, however.

OAuth-specific options

For OAuth providers, there are a few specific options to consider. Enabling **Use SSO** lets your users take advantage of pre-existing sessions for their credentials, allowing them to bypass the login form if they have an active session already. Depending on your organization's security stance, this may be something you need to leave disabled.

In addition, when using OAuth you can choose to **hide the internal authentication prompt** to users, forcing them to only log in with the OAuth credentials and not an internal Portainer user. Again, hiding the internal authentication prompt will depend greatly on the security requirements of your particular setup.

Configuring external authentication

You should now be ready to configure your external auth provider within Portainer. You will find links below to our documentation for each of our supported external authentication methods - choose the one that suits your needs and follow the provided instructions.

LDAP
Suitable for using an OpenLDAP or custom LDAP provider.

LDAP

Microsoft Active Directory
Use an on-premise or remote Microsoft Active Directory service for authentication.

ACTIVE DIRECTORY

OAuth
Select from a custom OAuth configuration or from pre-selected configurations for Microsoft, Google

or GitHub.

OAUTH

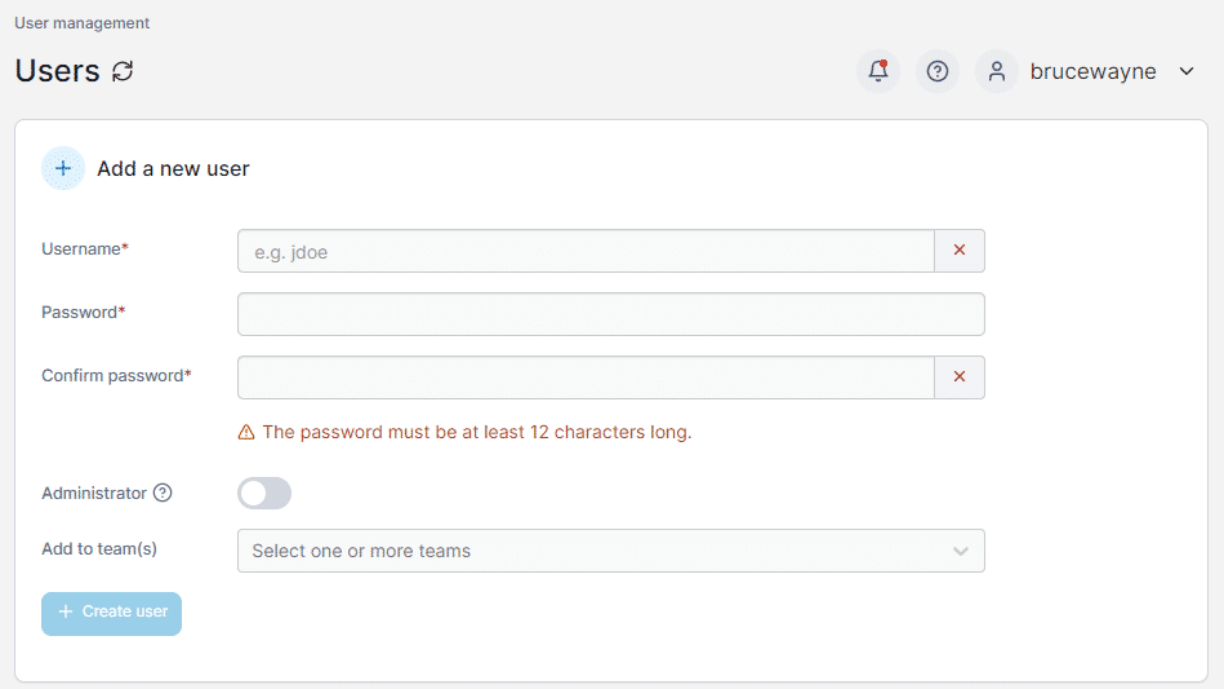
CONTINUE

4

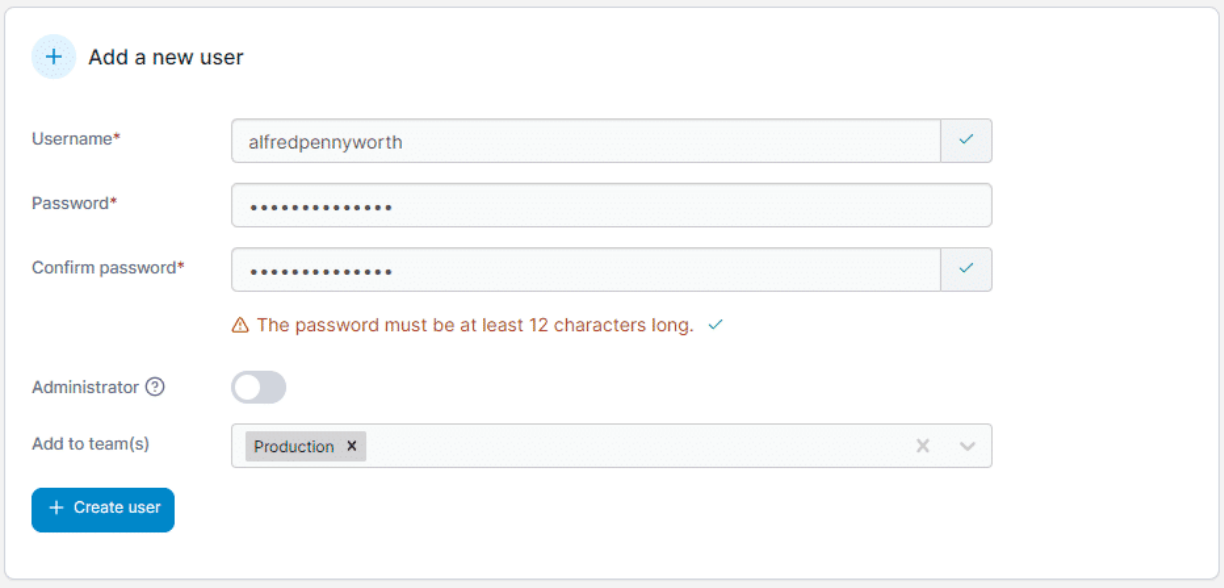
Alternative: Create internal users

If you don't have an external authentication provider to connect to or you would prefer to use a separate system, you can instead configure users directly within Portainer. Bear in mind that any changes to these users would be done within Portainer.

To create users within Portainer, log in as an administrator and select **Users** in the left hand menu. You'll be taken to a list of the current users as well as a form to add new users.



Fill in the **Username** for the user you want to create, then enter a strong password in the **Password** and **Confirm password** fields. On a new install, passwords must be at least 12 characters long, but administrators can adjust this minimum length if required. You can enable the **Administrator** toggle to make this new user an administrator, but we recommend using this sparingly. You can also select one or more teams to add the new user to from the **Add to team(s)** dropdown. When you're ready, click **Create user**.



CONTINUE

5

Summary

In this lesson we've covered the following topics:

- [ ] The authentication methods Portainer supports: external authentication and internal authentication, and when to use each method.
- [ ] Creating teams for your users, with names that match your existing groups (if using external authentication).

- [ ] Setting up your external authentication provider for use with Portainer, and some of the settings you may want to be aware of.
- [ ] Adding internal users to Portainer, if you are using internal authentication.

You may have noticed we haven't given any permissions to our users or teams yet. Permissions in Portainer are tied closely to the environments themselves, and we'll cover adding environment-specific roles to users and teams when we start adding our environments to Portainer in the next lesson.

Adding environments

So far we've installed Portainer, secured the configuration and set up authentication. Now we're ready to move on to setting up our environments. In this lesson we will cover:

1. What an environment is
2. The options for connecting to an environment, and when to use each one; and
3. How to add your environments to Portainer

START

1

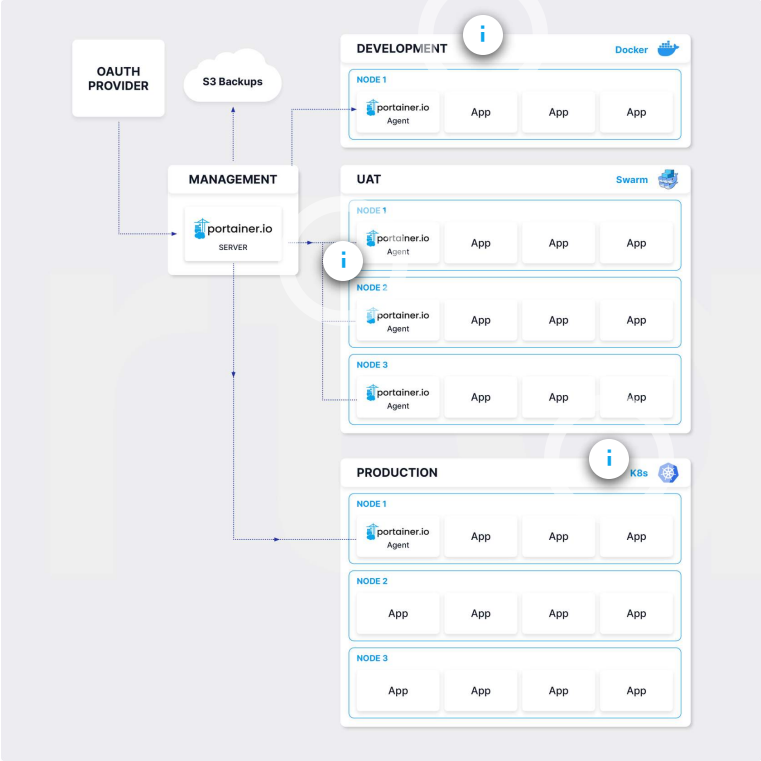
What is an environment?

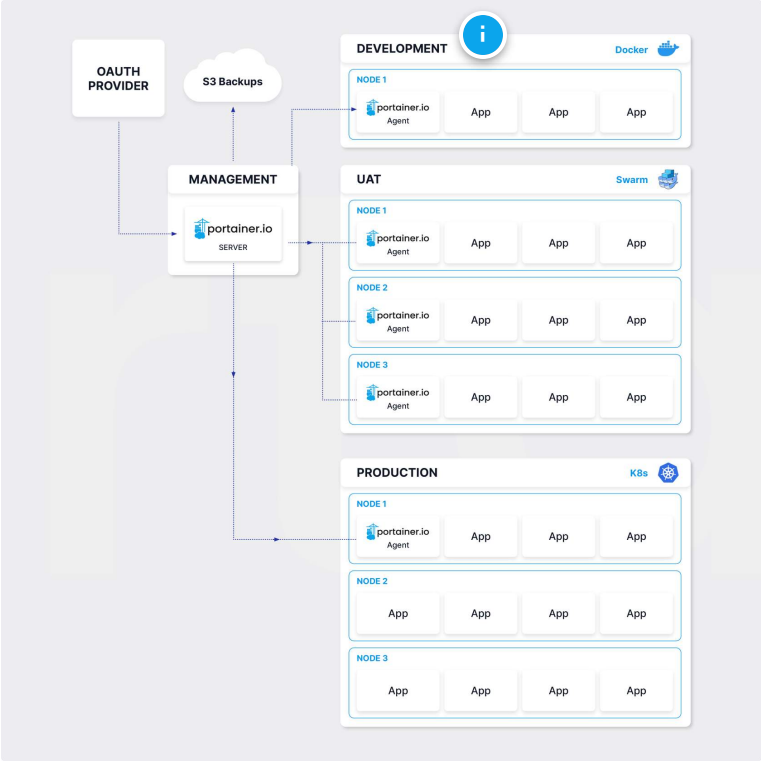
When we refer to environments within Portainer, we are talking about the individual containerization setups you want to manage. For some organizations this may be as small as a single environment, perhaps the one that the Portainer Server is deployed to, in

which case this lesson can be skipped. For most organizations however there will be one or more setups external to the management environment that you need to connect to and manage, whether they be individual Docker Standalone environments, Swarm or Kubernetes clusters, or Edge / IoT / IIoT devices.

> **In previous versions of Portainer, we referred to environments as "endpoints", so you may see them referenced as such in older documents and in the API for legacy support.**

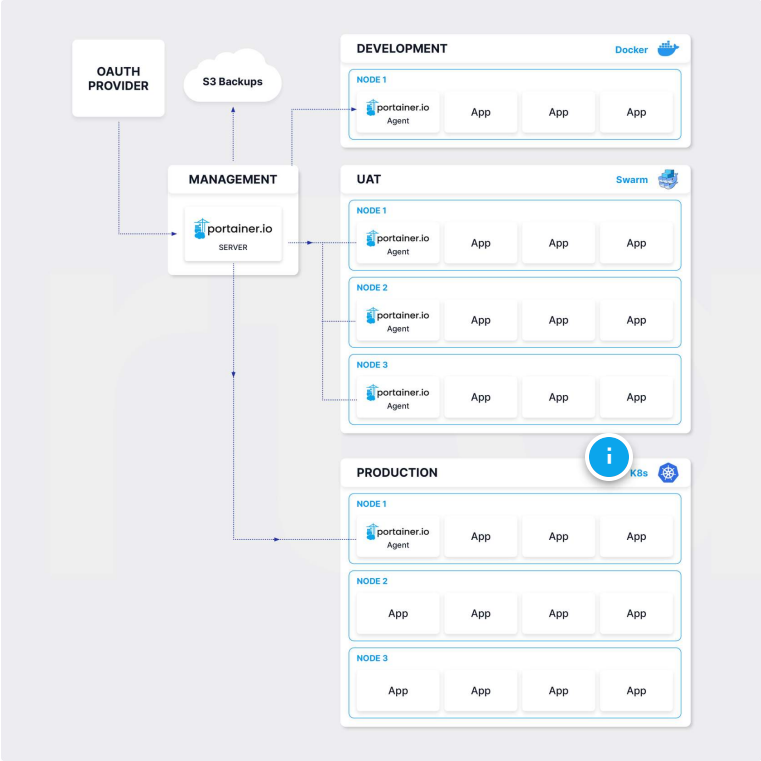
If we refer back to our example architecture diagram, we can get a better picture of what an environment is.





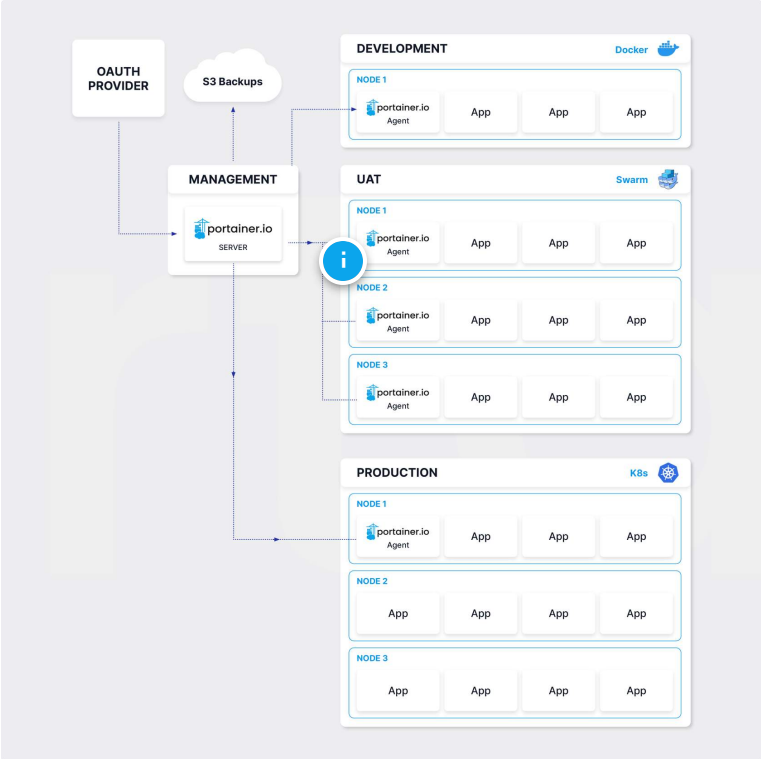
Development environment

In this example, this development environment is a single-node Docker Standalone setup.



Production environment

The production environment in this example is a three node Kubernetes cluster. As with the UAT Swarm environment, despite having three nodes this is considered *one* environment.



UAT environment

The UAT environment here consists of a three node Docker Swarm cluster. Despite the number of nodes, this is *one* environment.

CONTINUE

2

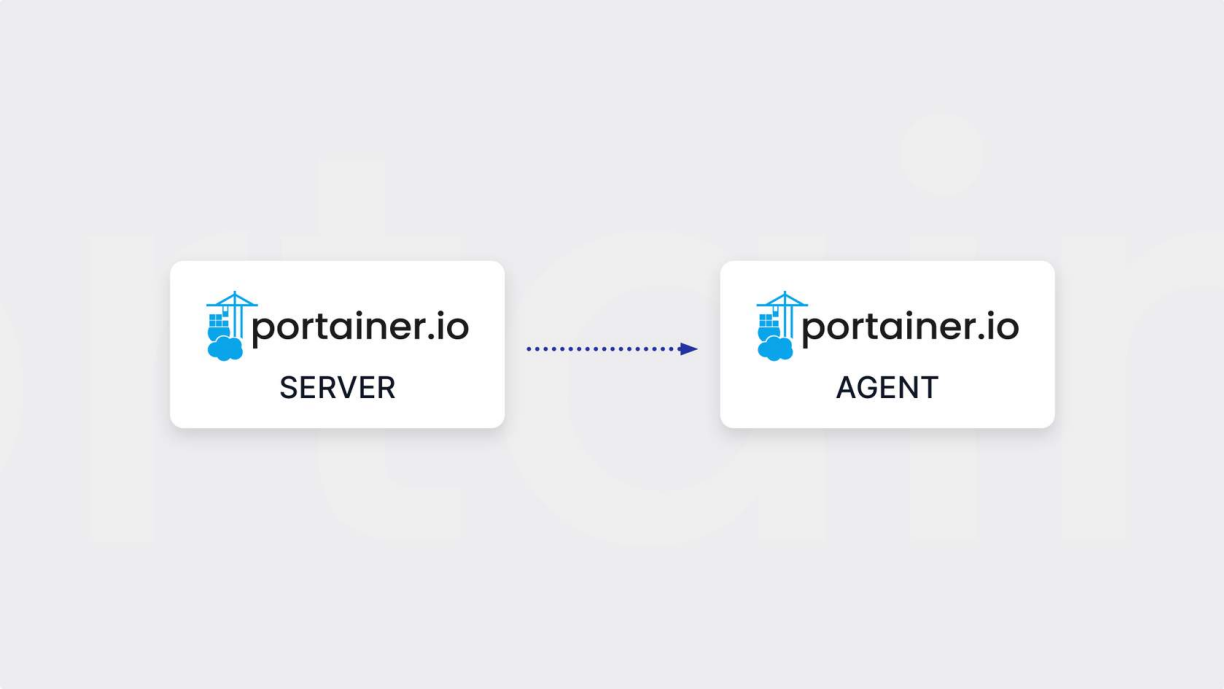
Agent vs Edge Agent

Portainer supports a number of different connection methods for adding environments, but in a production setup we highly recommend the use of the Portainer Agent to connect. The Portainer Agent is a lightweight container that runs on your environment

and facilitates the communication between the environment and the Portainer Server instance. The Portainer Agent can be deployed in two different configurations: Agent and Edge Agent, with the primary difference being how they communicate with the Portainer Server instance.

Agent

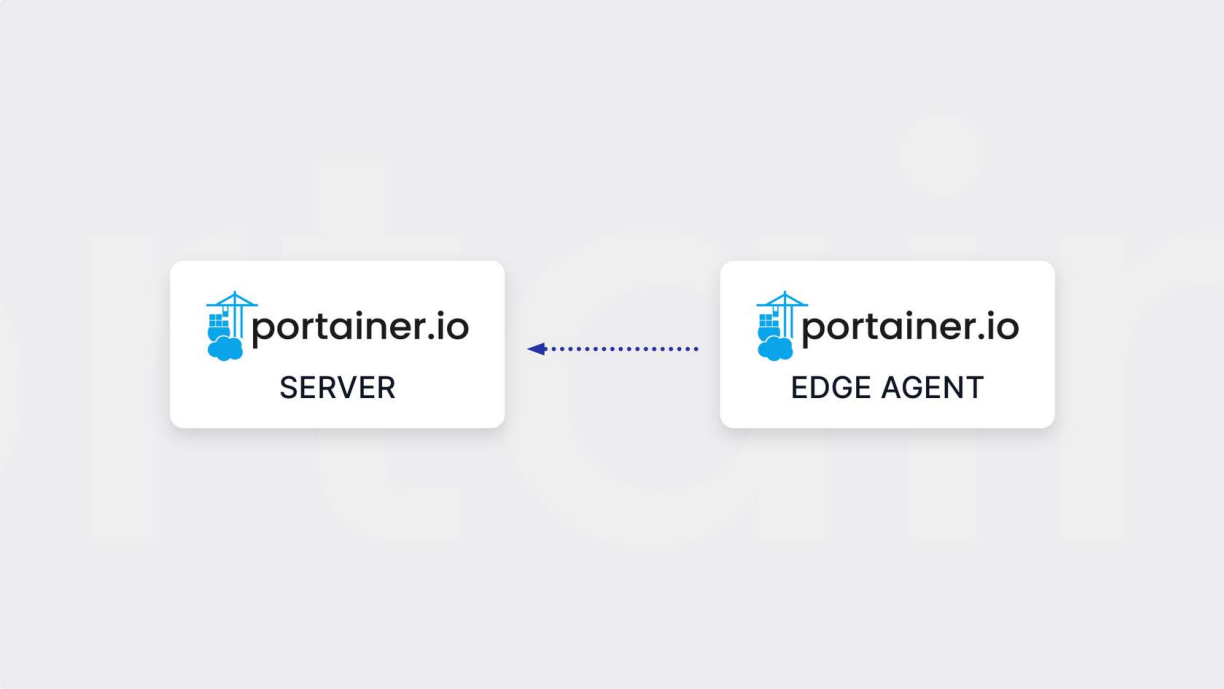
In Agent mode, the Portainer Server instance initiates communication from itself to the Portainer Agent container. With this method you are interacting with your environment in real time.



This requires that the Agent listen on a specific port for connections so that the Server can connect. As such, we generally recommend the use of Agent mode only in private networks where exposing a port on the Agent is acceptable within your organization's security posture.

Edge Agent

In Edge Agent mode, the opposite occurs. The Agent periodically connects back to the Portainer Server instance to check if there are pending tasks to perform. As a result, there is no need to expose any ports on the Agent end, making the Edge Agent mode ideal for remote environments outside of your network, and requiring only that your Portainer Server be accessible from the Agent.



Because the Agent initiates the communication in Edge Agent mode, you don't necessarily have instant access to your environment initially. You can however use a reverse tunnel initiated by the Agent to provide this access. When you select an Edge environment to manage through the Portainer UI, behind the scenes the Portainer Server logs a request for a tunnel to be opened. When the Edge Agent next connects to the Portainer Server to check for updates, it will see the pending tunnel request and initiate the tunnel, providing you access to the remote environment. Because of this check in process, you may need to wait for your tunnel to establish. The check-in interval for Edge Agents defaults to every 5 seconds, but this can be adjusted to suit your needs.

Edge Agent Async

The Edge Agent can also be configured to run in Async mode. For the most part this mode works the same as the standard Edge Agent configuration, with the notable exception that the reverse tunnel functionality is not available. Environment status is available through the use of "snapshots" sent periodically from the remote environment to the Portainer Server. This means that Async mode is best suited for IoT and IIoT devices where direct interaction with the environment is not required, and instead there is a desire for very small amounts of data to be transmitted, which is helpful when there may be limited or intermittent connectivity with the remote device, or when your remote devices are connected over unreliable network connections.

Pros and cons

As we've covered above, there are pros and cons for each deployment option. To summarize, your Agent deployment options are as follows:

Agent Type	Comm. direction	Pros	Cons	Best for
Agent	Server → Agent	Instant access Real-time management	Requires exposing port at the Agent	Environments on local / private networks
Edge Agent Standard	Agent → Server	No exposed ports at the Agent Real-time management (on demand)	Delayed real time access	Environments on remote networks

Agent Type	Comm. direction	Pros	Cons	Best for
Edge Agent Async	Agent → Server	No exposed ports at the Agent Low data usage	No real time access	Remote IoT / IIoT devices

You can have a mix of agent types across your setup, but each environment should only be added once and with one agent type.

CONTINUE

3

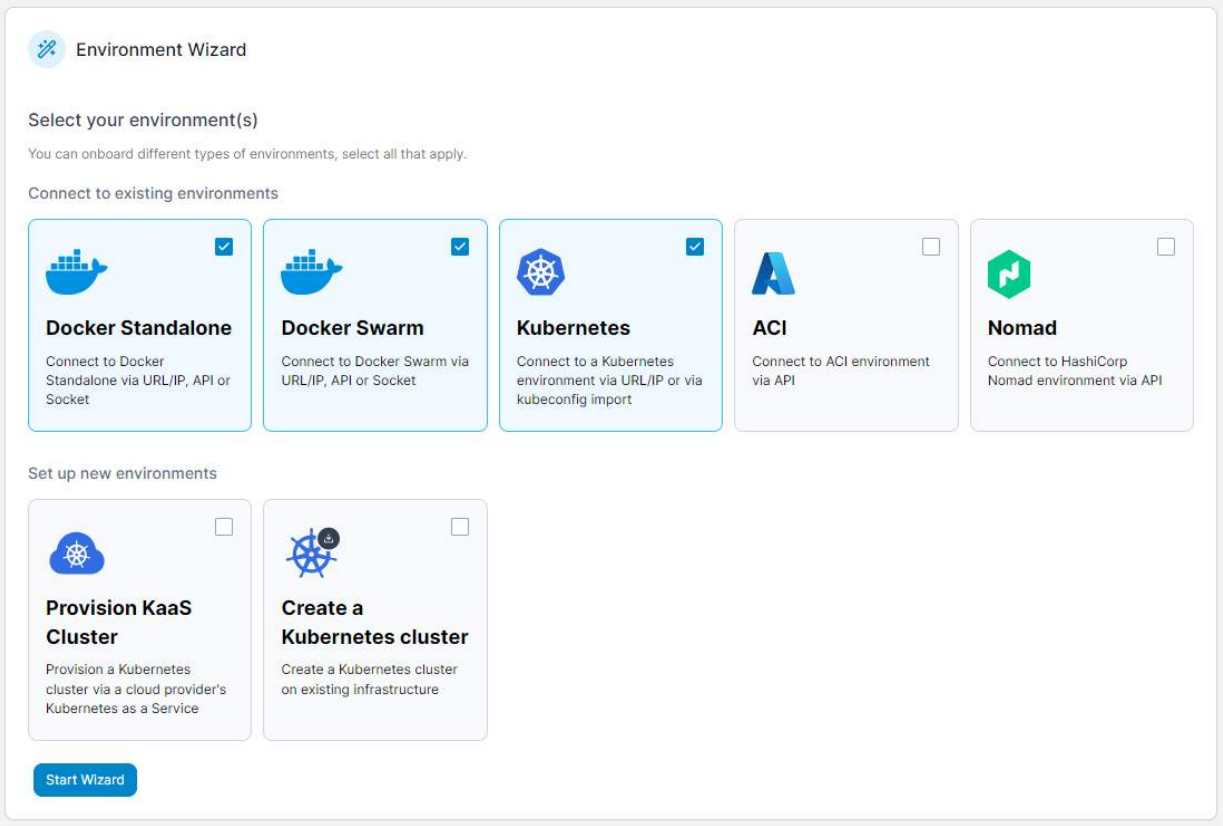
Add an environment

You should now have a better idea on which Agent type you want to use for your environments, and are now ready to start adding those environments to Portainer.

For the purposes of this lesson we will be covering adding existing environments to Portainer rather than provisioning new environments. You can learn more about provisioning a KaaS

cluster or creating a Kubernetes cluster directly from Portainer in our documentation.

To add a new environment in Portainer, log in as an administrator and select **Environments** in the left hand menu, then click the **Add environment** button in the top right. This will take you to the Environment Wizard. Check all the environment types you want to add in the **Connect to existing environments** section and click **Start Wizard**.



You'll now be asked to complete the configuration for each of the environment types you selected. You can find a full list of supported environments in our documentation, with direct links to the instructions for Docker Standalone, Swarm and Kubernetes below.

Docker Standalone

Add a Docker Standalone environment to your Portainer instance.

DOCKER STANDALONE

Docker Swarm

Add a Docker Swarm cluster to your Portainer instance.

DOCKER SWARM

Kubernetes

Add a Kubernetes cluster to your Portainer instance.

KUBERNETES

Once you have filled in the details for your environment, you can click **Next** to proceed to the next environment type. If you have added all your environments, click **Close** to return to the environment list.

CONTINUE

Summary

In this lesson we covered adding the environments you wish to manage to your Portainer installation. We talked about:

- [ ] What we mean when we refer to an "environment" in Portainer.
- [ ] The options for connecting to environments and when you should use each one.
- [ ] How to add your environments to Portainer.

In the next lesson we'll combine the work of this lesson and the previous lesson on configuring access, and set up access to our newly-added environments for our users and teams.

Managing environment access

We've got our users and teams configured, our environments added, and now we need to combine the two and give permissions to our users and teams to access the environments. In this lesson we'll talk about:

1. The Role-based Access Control (RBAC) system in Portainer and how it works.
2. How you could structure your role assignments across users and teams.
3. How to assign roles to your users and teams.

As your particular organizational requirements will be unique to you, we'll be speaking in examples for the most part. Adjust these to suit your needs.

START

Role-based Access Control

In Portainer Business Edition, we provide Role-based Access Control, or RBAC, as a way of providing differing levels of access control to users and teams based on their needs, in a consistent way across environment types. Docker does not natively provide any sort of access control, so we have developed our own to provide this. On Kubernetes environments, we leverage the built-in RBAC functionality that Kube provides in combination with our own role management.

We define a role as a set of privileges, as in the rights to perform actions. Users and teams can be assigned roles, inferring that role's privileges on the user or team. Roles are also environment-specific - that is, you would assign a role to a user or team, and then associate that pairing to an environment. As a result, a single user or team can have different roles for different environments.

Portainer's roles

Portainer has five assignable roles for users and teams. They are:

- **Environment administrator**
  The Environment administrator role has full access within the assigned environment, but cannot make any changes to the infrastructure that underpins the environment (ie, the host), and are also not able to make changes to global Portainer settings. Environment administrators also can't make changes to resource ownership within an environment.

- **Operator**
  The Operator role, as the name suggests, has operational control over the resources deployed within the assigned environment. They

can start, stop, update, and redeploy containers or services, check log files, and connect to containers via the console, but are not able to create new resources or delete existing resources.

- **Helpdesk**
  The Helpdesk role is a read-only user, able to see all the resources in the assigned environment but not able to make changes to them in any way. They also cannot console into a container or make changes to volumes.

- **Standard user**
  The Standard user role has complete control over their own or their team's resources within the assigned environment, including the ability to deploy new resources.

- **Read-only User**
  The Read-only user role is similar to the Helpdesk role in that it is read-only, however it only has access to resources they are entitled to see, for example resources created by members of their team and resources marked as public.

In addition to these five roles, the Administrator role (which your initial user was created with) exists as a "global admin", with complete control over Portainer's configuration and all environments. As such, this account should be used sparingly and protected well.

For a more detailed reference as to how Portainer roles relate to platform permissions, refer to our documentation for Docker and Kubernetes.

CONTINUE

2

An example role structure

Let's look at an example for how you might structure your role assignments. In a previous lesson we talked about some teams you might set up:

- Developers
- Support
- QA
- Tech Leads

Let's also imagine we have the following environments added to Portainer:

- Development
- Staging / UAT
- Production

In this scenario, we would want our developers to have access to the Development and Staging environments, but not necessarily Production. Our support team don't need to get into Staging or Development, but should have limited access to Production. The QA team needs to have access to Staging but nowhere else, and the tech leads will need access to everything.

Based on this, here's how we would configure our role assignments:

Environment	Team	Role
Development	Developers	Standard user
	Support	No access
	QA	No access
	Tech Leads	Environment administrator
Staging / UAT	Developers	Standard user
	Support	No access
	QA	Standard user
	Tech Leads	Environment administrator
Production	Developers	No access
	Support	Helpdesk

Environment	Team	Role
	QA	No access
	Tech Leads	Environment administrator

Developers have the **Standard user** role in both Development and Staging, so that they can create and modify their own (and their team's) workloads for testing. They don't need access to Production. Support has the **Helpdesk** role on Production, giving them read-only access to the production workloads so they can troubleshoot without being able to make changes. QA has the **Standard user** role on Staging so they can do their testing, but no other access. And finally, the Tech Leads have the **Environment administrator** role on all environments, giving them full access as required.

Remember, this is just an example setup. Your particular needs will differ from the above, but this should give you a good idea of the possibilities with roles.

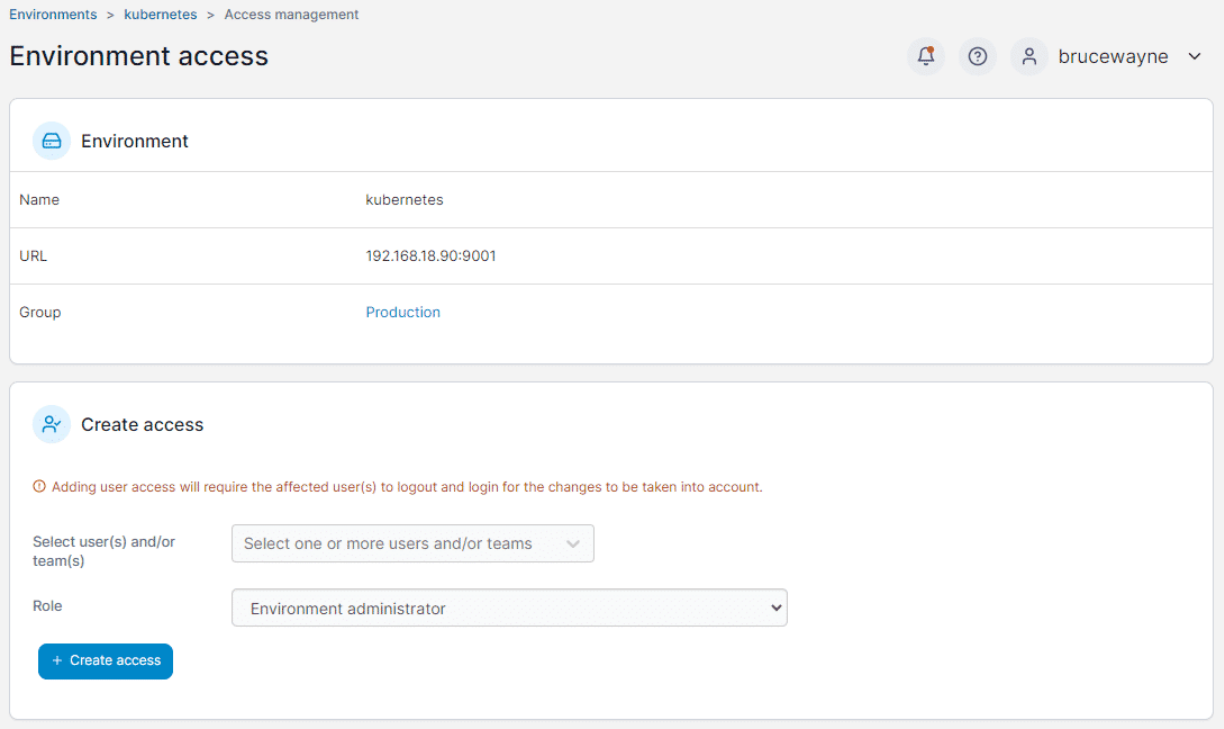
CONTINUE

3

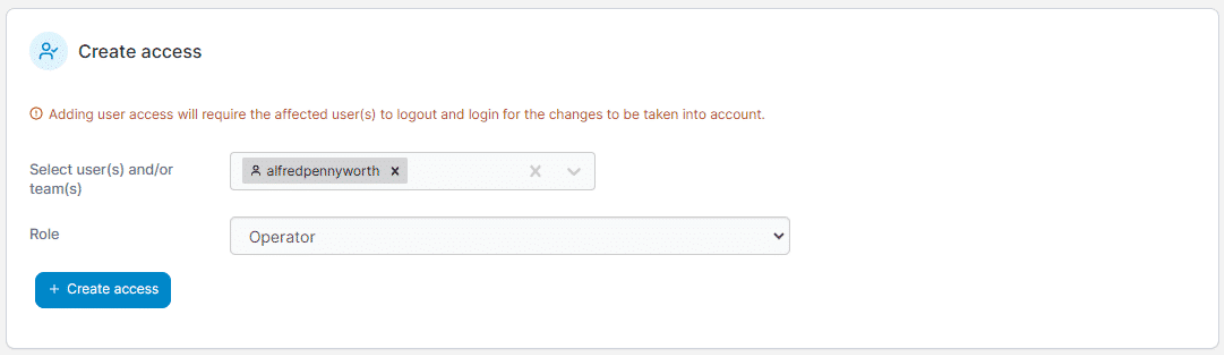
Setting roles for users and teams

Now that we understand the available roles and have an idea as to how we could allocate them, let's assign some roles.

Log into Portainer as an administrator and select **Environments** from the left hand menu. Find the environment you want to configure access for in the list, and select **Manage access** on the right.



Here you'll see some information about your environment, a Create access section, and the current access configuration. To add a new access configuration, in the **Create access** section select the **users and/or teams** from the dropdown and the corresponding **role** to assign to those users and/or teams. To submit, click **Create access**.



Do this as many times as you need to for the various users and teams you want to add with each role. You can check the list below to see your configuration. When you've finished setting up access for this environment, you can return to the Environments page and select your next environment to configure.

CONTINUE

4

Summary

In this lesson we've covered the following topics:

- [ ] What the RBAC system in Portainer is and how it works.
- [ ] An example configuration of role assignments.
- [ ] How to assign roles on environments to users and teams.

Up next, we'll cover configuring your registries with Portainer, and how to provide your users with access to those registries.

Lesson 11 of 13

Adding registries

In a production environment it is highly likely you will have your own container images you need to deploy, or at least custom versions of common images. As such, you will likely store these in image registries. In this lesson we'll talk about adding those registries to Portainer. We'll cover:

1. The types of registries currently supported by Portainer.
2. Adding a new registry in Portainer.
3. Configuring access to your registries on your environment.

START

1

Registry types

Portainer supports the use of multiple types of registries. These include:

- Docker Hub (authenticated accounts)
- AWS Elastic Container Registry (ECR)
- Quay.io
- ProGet
- Azure Container Registry
- GitLab container registry
- GitHub container registry (ghcr.io)

We also support the use of custom registries that adhere to the Docker Registry API v2 standard.

For each registry type, access depends on providing credentials. The configuration and retrieval of these credentials differs from provider to provider, as does the type of information you require, so we advise referring to our documentation for the details needed by your registry type.

CONTINUE

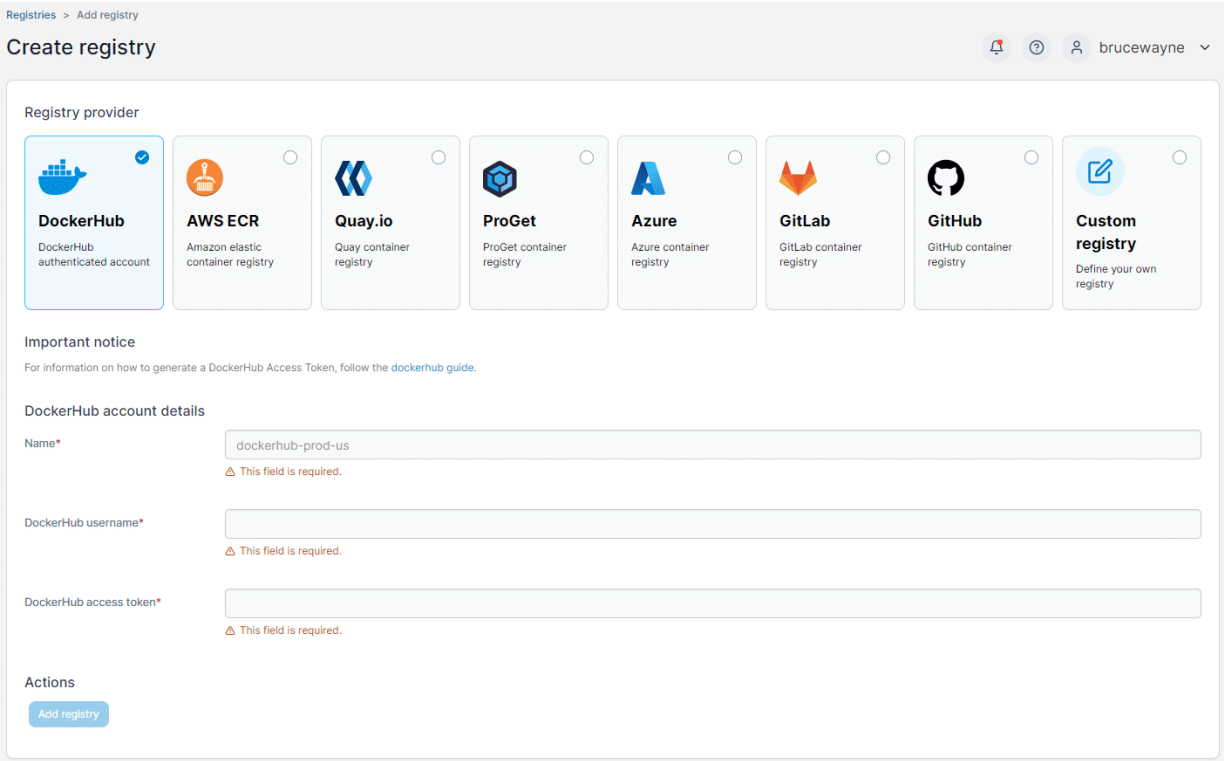
2

Add registries

Once you've determined the type of registry you have and have obtained the necessary credentials for access, you're ready to add it to Portainer.

Log into Portainer as an administrator and select **Registries** from the left hand menu. You'll be taken to a page that lists your current registries. To add a new registry, click the **Add registry** button on the right.

> **On a fresh installation, this list will include Docker Hub (anonymous) by default. If you wish to disable anonymous access to Docker Hub, you can click the Hide for all users button.**



From here, choose the type of registry you want to set up and fill in the relevant details. For specific instructions for each registry type, refer to our documentation:

- Docker Hub
- AWS ECR
- Quay.io
- ProGet
- Azure
- GitLab
- GitHub
- Custom registry

CONTINUE

3

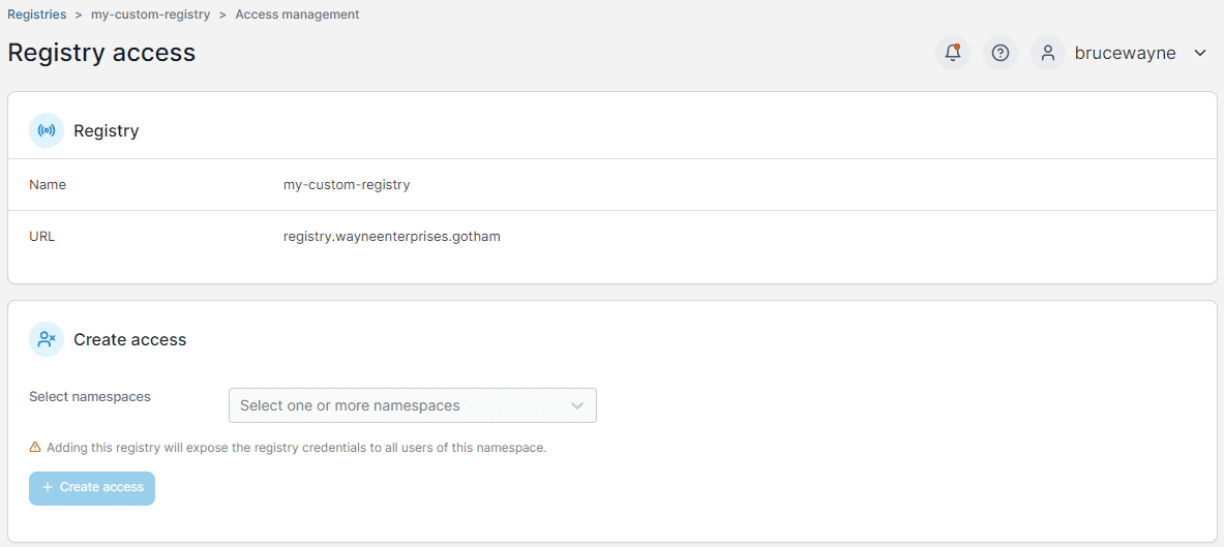
Registry / environment access

Once you've added your registry, you now need to configure access to your registries across your environments. This lets you specify the users and teams that are able to use each specific registry on a per-environment basis.

First, from the home page select the environment you want to configure with registry access. Depending on the environment type, select the following from the new section in the left menu:

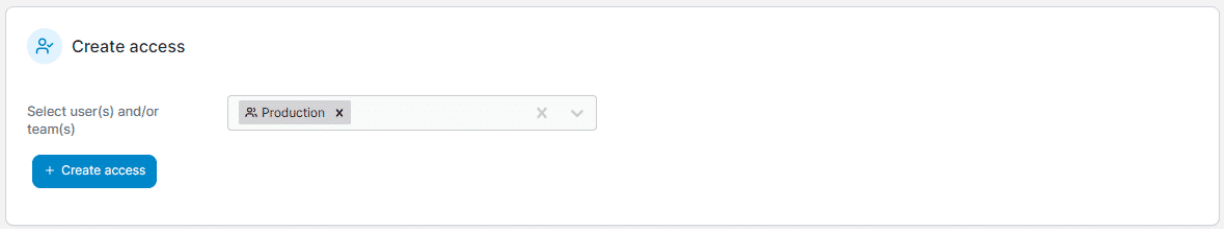
- For **Docker Standalone** environments, expand the **Host** option and select **Registries**.
- For **Docker Swarm** environments, expand the **Swarm** option and select **Registries**.
- For **Kubernetes** environments, expand the **Cluster** option and select **Registries**.

Regardless of your environment type, you should now be at the **Environment registries** page. You will see a list of registries that have been configured in Portainer. Locate the registry you want to configure for access and select **Manage access** to the right of it.

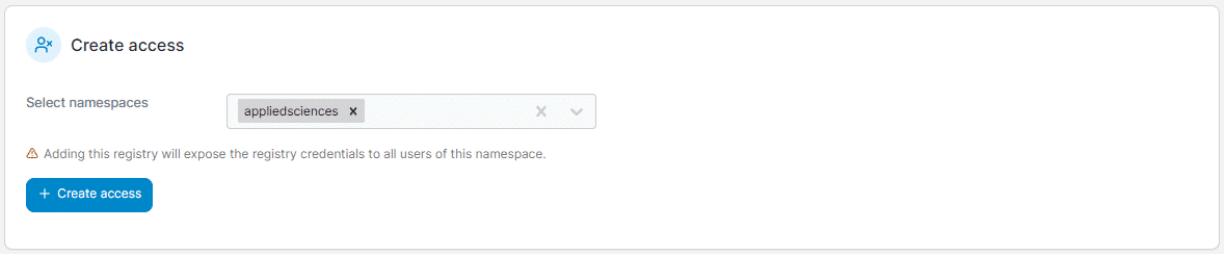


Here you'll see some brief detail about the registry, a Create access section and a list of the current access configuration. For Docker Standalone and Docker Swarm environments, use the dropdown in the Create access section to select the users and/or teams you want to have registry access and click **Create access**.

> **Note that the list of users and teams in the dropdown will be limited to those that have been provided with access to the environment. Any users or teams with no access to the environment will not appear in the list.**



For Kubernetes environments, registry access is provided to namespaces rather than users or teams. Use the dropdown in the Create access section to select the namespaces you want to have registry access and click **Create access**.



Once you have completed adding user and team access, return to the previous page to configure access for other registries on this environment. Once you've completed doing so for this environment, move on to your next environment and repeat the process as needed.

CONTINUE

Summary

In this lesson we covered the addition of registries and their access. Specifically, we talked about:

- [ ] The types of registries that are supported for use in Portainer.
- [ ] The process of adding registries within Portainer.
- [ ] How to set your users and teams up with access to your registries on each environment.

We're almost done! In the next lesson we'll talk about specific configuration and security settings for the different environment types supported by Portainer.

Securing your environments

In this lesson we'll cover some of the environment settings you should adjust to suit your needs, with a focus on making those environments as secure as possible. We'll look at:

1. General options that apply to all types of environment.
2. Options that apply to Docker Standalone and Docker Swarm environments.
3. Options that are specific to Kubernetes environments.

Where possible we will make recommendations for each setting, as well as provide links to more detailed documentation. We recommend working through each of your environments in turn and adjusting the settings as needed.

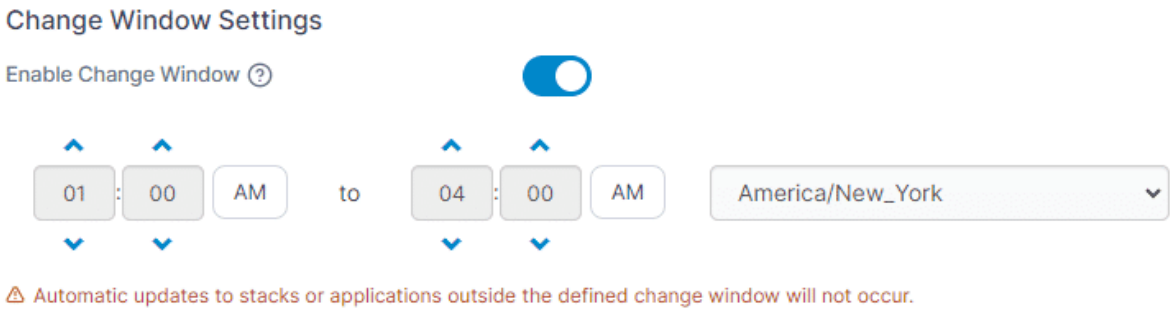
START

General settings

While for the most part each environment type has their own unique configuration settings, there are some that are available for all types.

Change Window

Portainer provides the ability to configure a change window for each environment, letting you specify when automatic updates to your stacks or deployments are allowed to be applied. This setting will depend greatly on your specific need, but we recommend configuring this on production to reduce the impact of any downtime for updates. Note this setting only applies to stacks or applications that have been deployed from Git and use the automatic update functionality.



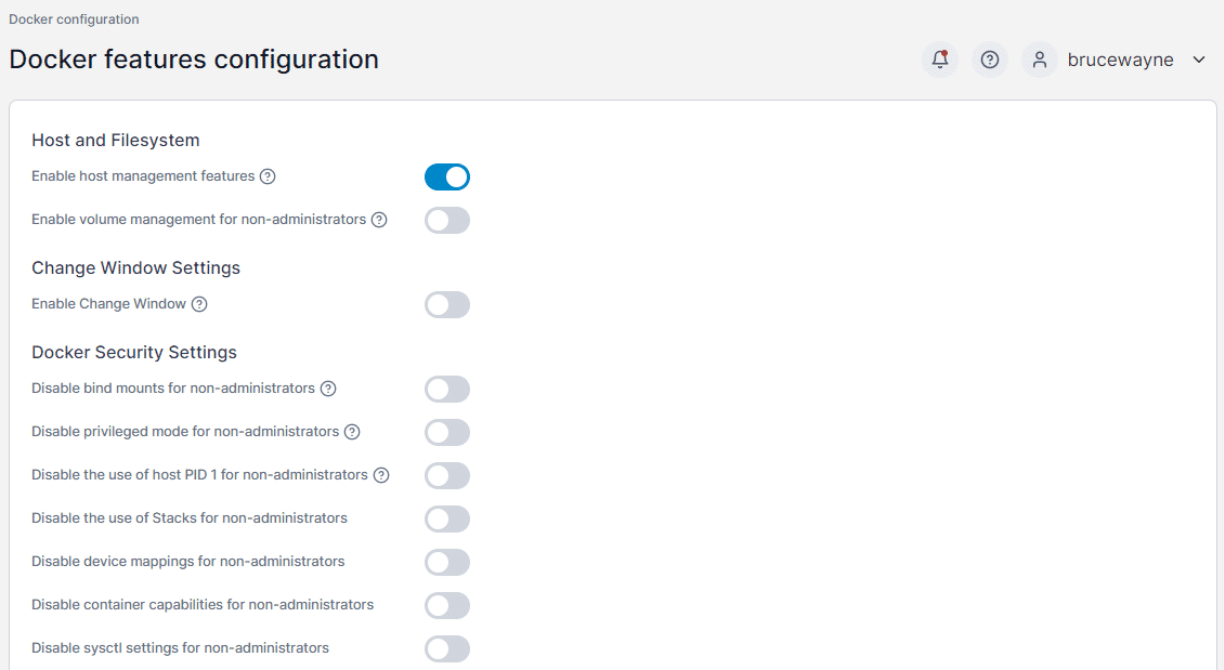
For Docker Standalone and Docker Swarm environments, you can find the change window settings under Host | Setup and Swarm | Setup respectfully. For Kubernetes, you'll find them under Cluster | Setup.

CONTINUE

2

Docker

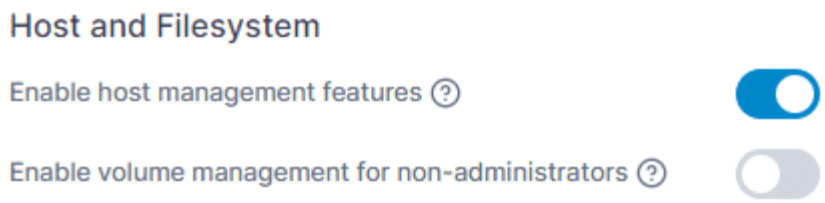
For the most part, the options available for Docker Standalone and Docker Swarm environments are the same. To access the Docker Standalone settings, select a Docker Standalone environment, expand the **Host** option in the left menu and select **Setup**. For Docker Swarm, expand the **Cluster** option and select **Setup**.



There are a number of options in this section that are worth examining and you can find detail on each of them in our documentation for Docker Standalone and Docker Swarm. In this lesson we'll focus on some of the more important security options to consider.

Host management

For Docker Standalone environments deployed with the Portainer Agent and for Swarm environments, you have the option of enabling host management features for the environment. This feature allows you to interact more closely with the underlying host, including listing available devices and browsing the filesystem.

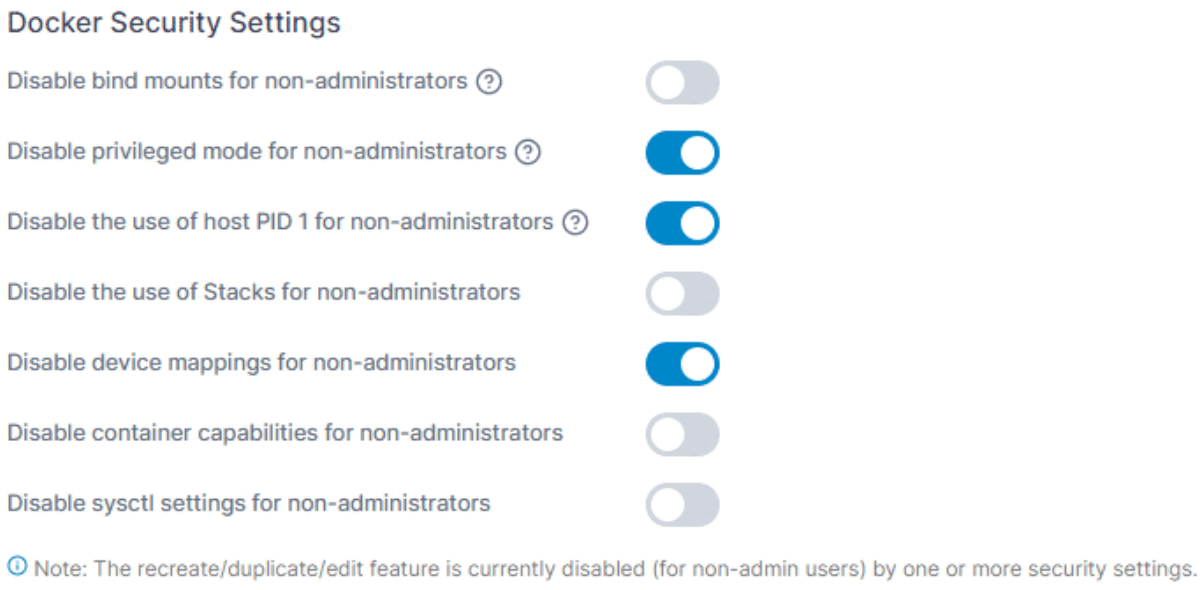


We recommend considering whether this functionality is a requirement for your deployment and only enabling it if it is something you require, as there could be security implications to being able to view and interact with the host directly.

Docker Security Settings

There are a number of toggles in this section and we recommend reading each one thoroughly. In particular, pay attention to:

- **Disable privileged mode for non-administrators**
  Unless you know that your users will need to deploy containers or services that require privileged mode, we recommend toggling this option on.
- **Disable the use of host PID 1 for non-administrators**
  Deploying a container that operates as PID 1 (the host PID) allows that container to run as if it was root on the host environment. If this is not necessary for your deployments, we recommend toggling this on.
- **Disable device mappings for non-administrators**
  When this option is on, non-admin users cannot map host devices to containers. If your workloads don't require the ability to map host devices to containers, we recommend toggling this on.



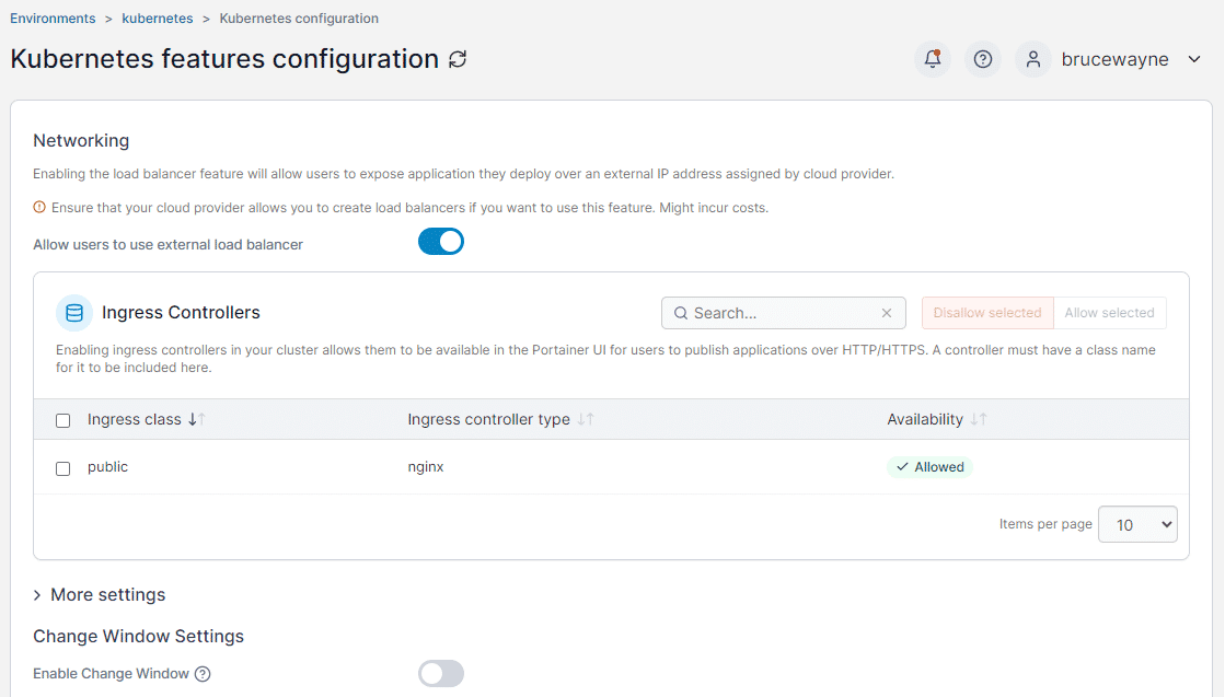
Again, consider each option carefully, especially any warnings that are displayed regarding affected functionality. You'll find more detail in our documentation for Docker Standalone and Docker Swarm.

CONTINUE

3

Kubernetes

For Kubernetes environments there are a number of configuration settings to consider, all of which can be found in our documentation, but in this section we'll cover some of the most important security options. To access the environment's settings, select a Kubernetes environment, expand the **Cluster** option in the left menu and select **Setup**.



Networking

There are two options within the Networking section of the Cluster setup page, both which are important to configure to your requirements.

- **Allow users to use external load balancer**
  Enabling this option lets your users create load balancers from Portainer in order to expose your applications externally. Bear in mind that this functionality requires that your cloud provider allows you to create load balancers, and that doing so may result in costs from your cloud provider for the hosting of the load balancer. If you don't think you'll need this, leave it off.

- **Ingress controllers**
  This section lists all the ingress controllers Portainer was able to discover in your cluster. You can choose to allow or disallow access to each ingress as required. If an ingress is disallowed, users will not be able to use that ingress to publish applications. We recommend setting this for each ingress as required.

Networking

Enabling the load balancer feature will allow users to expose application they deploy over an external IP address assigned by cloud provider.

Ensure that your cloud provider allows you to create load balancers if you want to use this feature. Might incur costs.

Allow users to use external load balancer

Ingress Controllers

Search...

Disallow selected | Allow selected

Enabling ingress controllers in your cluster allows them to be available in the Portainer UI for users to publish applications over HTTP/HTTPS. A controller must have a class name for it to be included here.

Ingress class	Ingress controller type	Availability
public	nginx	Allowed

Items per page 10

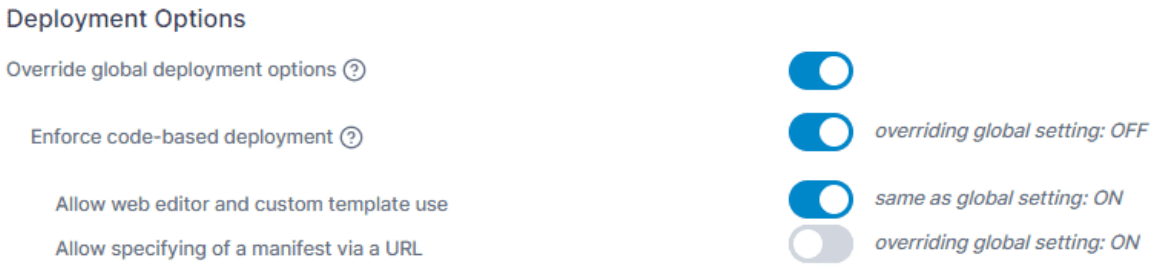
Deployment Options

You can use this section to apply restrictions on the ways your users are able to deploy applications on your environment. Note that this section only appears if the Allow per environment override option is enabled in the global Portainer settings. We recommend either configuring this globally or on an individual environment basis as required.

- **Override global deployment options**
  This lets you override the deployment options set in the global Portainer settings for this environment, otherwise this environment will inherit the global settings. Enable this if you need to change the restrictions on this environment.
- **Enforce code-based deployment**
  If this option is enabled, users will not be able to create applications or other resources via the Portainer forms, only allowing deployments through YAML files. If your organization enforces deployments from Git repositories, then this might be a good option for you.

The following options only appear if **Enforce code-based deployment** is on:

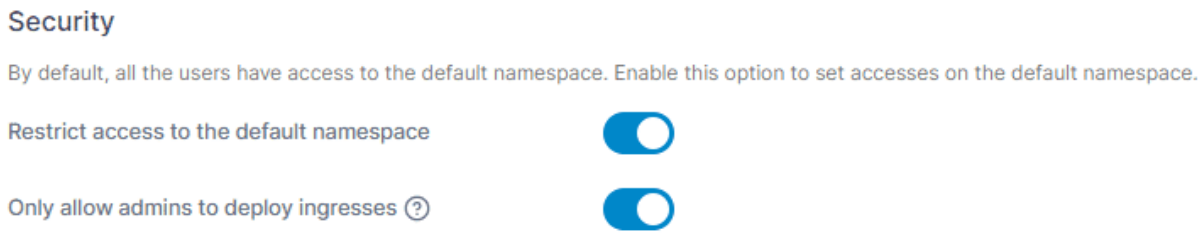
- **Allow web editor and custom template use**
  Enable this to allow users to use the web editor and custom templates to deploy applications.
- **Allow specifying a manifest via a URL**
  Enable this to allow users to deploy applications using the URL option.



Security

The following options appear in this section:

- **Restrict access to the default namespace**
  Enabling this option restricts users from being able to access the default namespace, limiting access to administrators only. Unless you know your users will need to deploy to the default namespace, we recommend enabling this option.
- **Only allow admins to deploy ingresses**
  If this option is enabled, only cluster administrators will be able to provision new ingresses in the environment. If you want to prevent your users from doing so, enable this option.

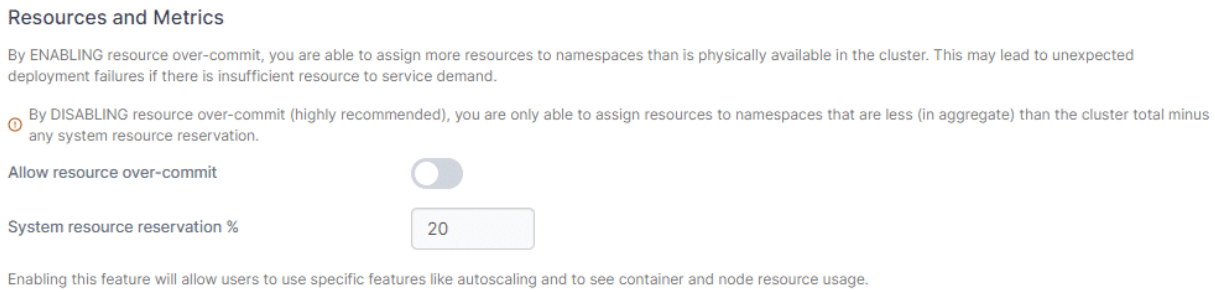


Resources

From a security perspective, the important option here is:

- **Allow resource over-commit**
  Enabling resource over-commit will let you assign more resources than are physically available on your cluster to your workloads. While this is helpful in some scenarios, for production environments we highly recommend leaving this disabled to avoid overprovisioning and potential resulting downtime. When the option is disabled, you can also configure a percentage of resources to reserve for use by the system.

  Allowing resource over-commit is really only useful if you have cluster auto-scaling enabled with your cloud provider, which would allow additional cluster nodes to be spun up automatically as needed. If this isn't functionality you need, you can safely leave this option disabled.

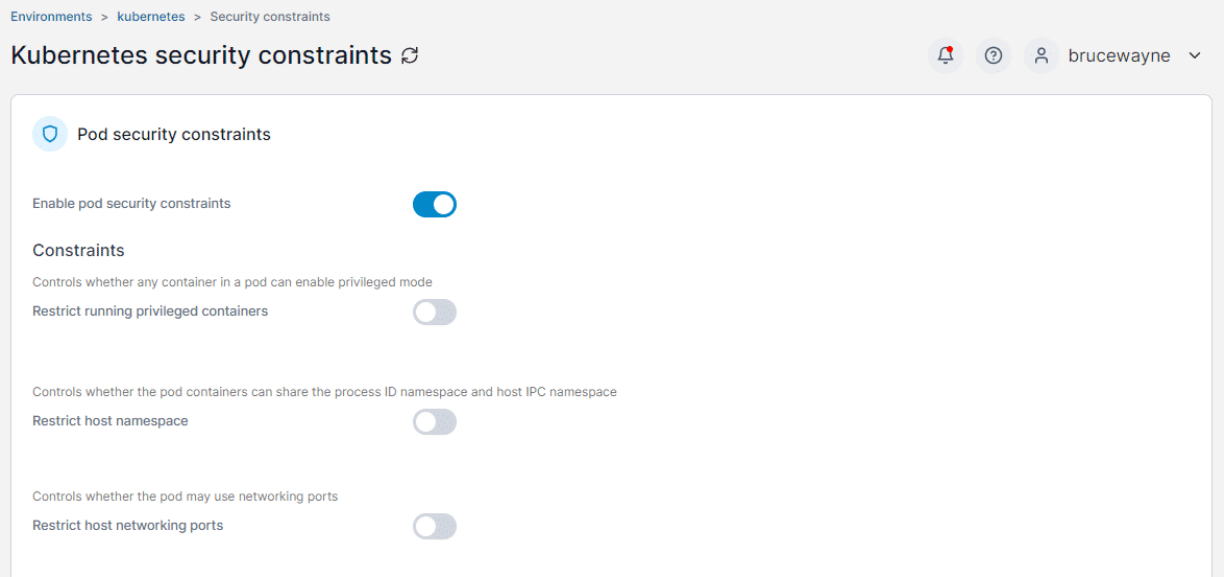


Security constraints

For Kubernetes environments, Portainer implements pod security policies through the use of OPA Gatekeeper. These are per-environment, and let you apply fine-grained restrictions to the functions available to pods deployed on your environment.

> **Be cautious when applying these settings! They are quite powerful, and when a pod is unable to deploy because of one of these restrictions it may not be immediately obvious that these are the cause. We recommend testing first!**

To access these settings, expand the **Cluster** option in the left menu and select **Security constraints**. From here, toggle on **Enable pod security constraints** to show the full set of options.



Because of the potential impact of these restrictions and the variety of functionality a pod may require, we're not able to make recommendations here. However, we do advise checking these carefully and testing your deployments to determine what works best for your needs. We provide full detail on each option in our documentation.

CONTINUE

4

Summary

In this lesson we've covered securing your environments within Portainer, looking at:

- [ ] General options available to all environment types,

- [ ] Docker Standalone and Docker Swarm specific options, and
- [ ] Options specific to Kubernetes environments.

In the final lesson of the course, we'll summarize everything we've covered in this guide.

Summary

Congratulations, you are done! You should now have a running Portainer Business Edition installation, with:

- [ ] Your custom SSL certificate installed
- [ ] Firewall restrictions applied for access
- [ ] Automatic backups of the Portainer database in a S3 bucket
- [ ] External (or alternatively internal) authentication configured
- [ ] Teams configured for your users
- [ ] Environments added to manage from Portainer, configured and secured according to your organizational needs
- [ ] Access to your environments configured for your users and teams with only the permissions they require
- [ ] Registries added for use by your users, teams or namespaces as permitted

If you make changes to your configuration in the future (for example, if you add new environments to manage) we recommend running through the relevant lessons in this guide again to ensure you stay secure.