Tutorial: Role Based Access Control in Universal Control Plane

By Vivek Saraswat March 18 2016

datacenter, docker, docker datacenter, Docker tutorial, tutorial, ucp, universal control plane

It's been an exciting time for us since we released Docker Datacenter a couple of week ago, which included the GA of Universal Control Plane (UCP), our on-premises or VPC deployable management and orchestration solution for containers. Thank you for the enthusiastic response, and we'll continue to deliver enterprise-grade feature sets into the Docker Datacenter platform.

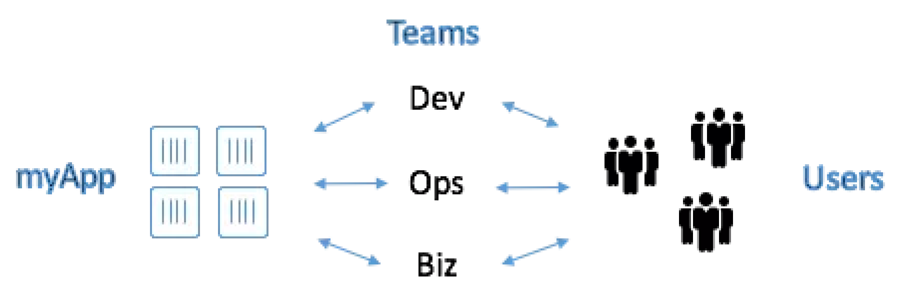
One such feature we delivered in UCP 1.0 was Role Based Access Control (RBAC)—deciding and enforcing who gets access to which resources. This is a really important and complex component of commercial IT infrastructure. This tutorial walks through how the feature is implemented in UCP and demonstrating it with an example.

Why RBAC?

RBAC is important because it addresses the following questions for your infrastructure:

1. Which users get access to a resource?
2. What level of access do these users get to that resource?
3. How do I enforce this access within my environment?

The easiest way to understand RBAC is through an example. Let's say you are the system administrator for AppCo, which is developing a containerized application called myApp. You have three teams within your organization—a Dev team, an Ops team, and a Biz team—each of which interacts with myApp in a different way. Dev needs to build the app and test it on the cluster. Ops needs to deploy the application and access kernel-level resources on hosts. Biz wants to take a look at myApp from time to time.

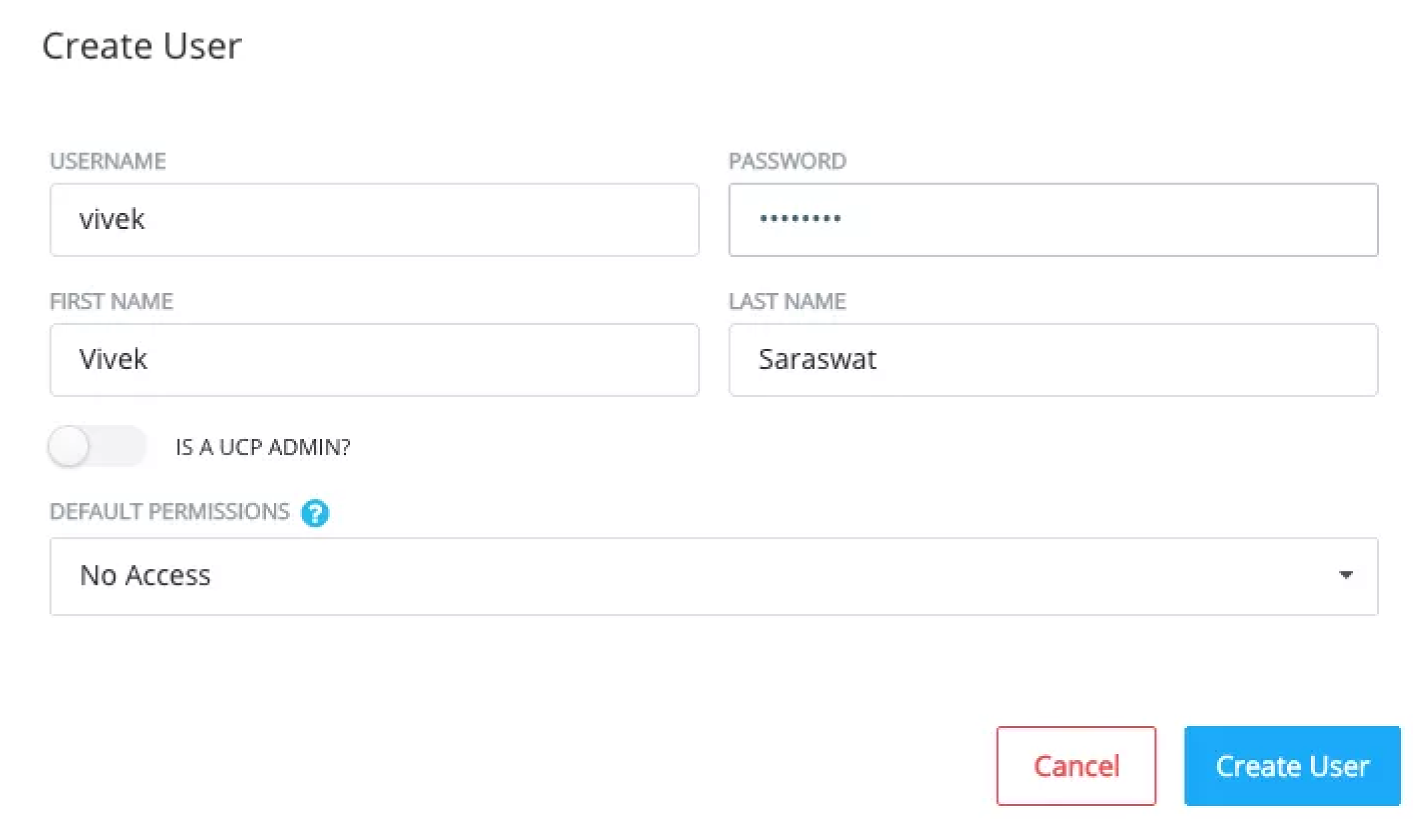


AppCo at work (applications, teams, users)

All of these teams need to access myApp, but how do you ensure that each team onlys get the specific level of access it needs to get the job done? Let's walk through how RBAC works in UCP, and use AppCo along the way as an example.

Users and Teams

The first step to setting up RBAC is to create your users. UCP users can either be created manually or imported via LDAP/AD integration, and come in two flavors—admins, and users. Admins have the ability to do everything within a UCP environment: They have the ability to access all resources, change UCP settings, manage users accounts, and set access permissions. Non-admin users, by default, have no permissions beyond what is granted by admins. They can change their password, but that's about it. In the case of AppCo, you are the plucky admin (the first admin account is created as part of installation) so you create non-admin user accounts for everybody else in the organization.


Create User

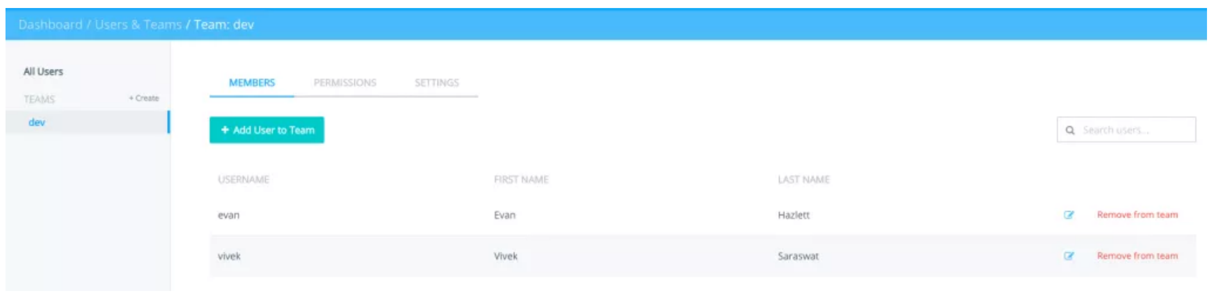
USERNAME	PASSWORD
vivek	•••••••••
FIRST NAME	LAST NAME
Vivek	Saraswat

IS A UCP ADMIN?

DEFAULT PERMISSIONS: No Access

Creating a user

Now that you've created your users, you'll want to put them in teams. A team is a grouping of users for access control purposes. Teams can be manually created via UCP or can be synced through LDAP/AD integration. For AppCo, you've created three teams in UCP to match the real world ones: Dev, Ops, and Biz. You can then assign each user to their given teams. Note that a user can be a part of multiple teams at the same time.


Dashboard / Users & Teams / Team: dev

USERNAME	FIRST NAME	LAST NAME		
evan	Evan	Hazlett		Remove from team
vivek	Vivek	Saraswat		Remove from team

Adding users to a team

Levels of Permissions

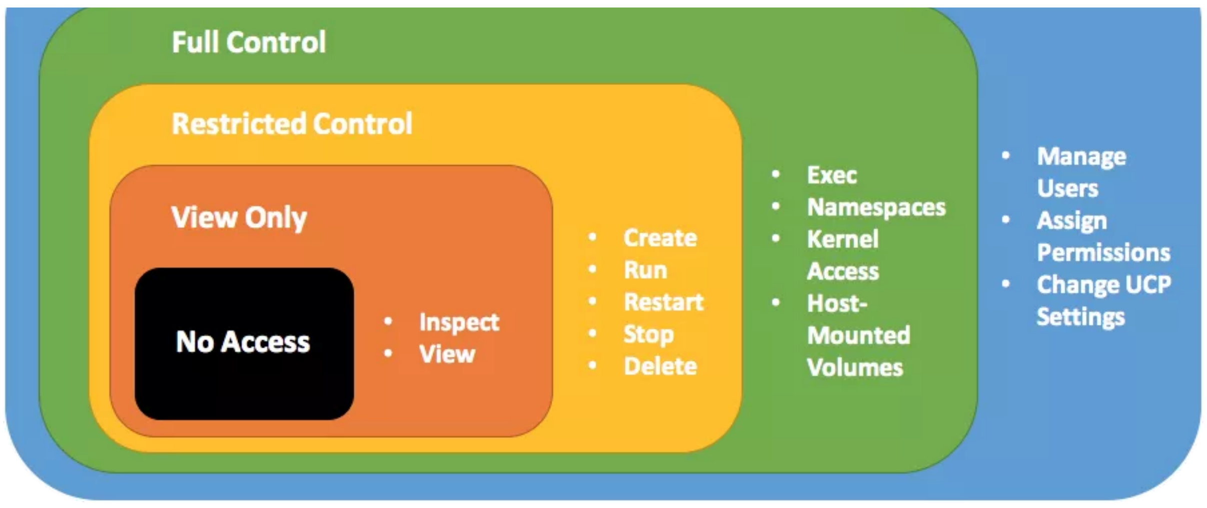
Next, you want to figure out what level of permission to give to your users. UCP provides four distinct sets of permissions. This makes it easier for you to assign and understand user access without having to set individual permissions for each and every action possible. These four sets of permissions are:

Full Control: Can do anything possible to resources. Create, restart, kill, view containers, etc. This is the highest level of access a non-admin user can have.

Restricted Control: Similar to Full Control, but with restrictions around container exec, privileged containers, host-mounted volumes, and other particularly sensitive operations. This is best suited for when you want a group to run containers in production but not access kernel capabilities or modify a container using exec privileges.

View Only: Look, but don't touch. Can view and inspect resources, but nothing else.

No Access: Cannot view or otherwise access resources.



How permissions work in Universal Control Plane

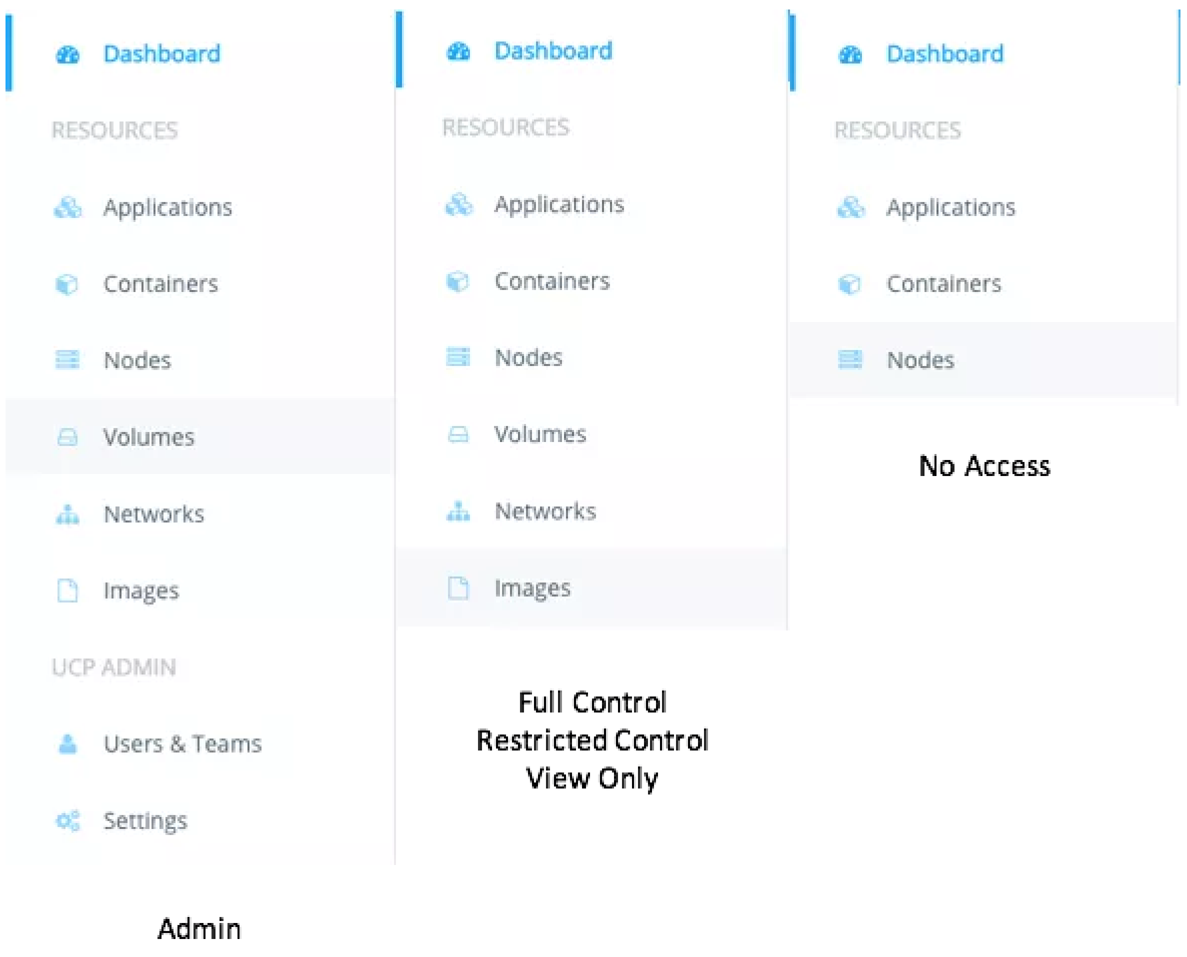
You can find more detailed information on permissions in the UCP documentation.

Enforcing Coarse-Grained Access through User Default Permissions

UCP allows you to set and enforce resource access through two distinct methods: user-assigned default permissions, and team-assigned container labels. First we'll discuss default permissions.

Every user has a default permissions setting which is assigned at account creation, and can be edited by an admin at any time. The default permissions are enforced for all non-container resources such as images, networks, and volumes.

In AppCo, when you created the users they had the default setting of "No Access." Thus, each user won't even be able to see the tabs for images, networks, and volumes. Thinking again, you realize that the folks on the Ops team will need the ability to create and destroy non-container resources, so you change all of their user default permissions to "Full Control." The folks on the Dev and Biz team should be able to see these resources to understand the cluster deployment, so you change all of their user default permissions to "View Only."

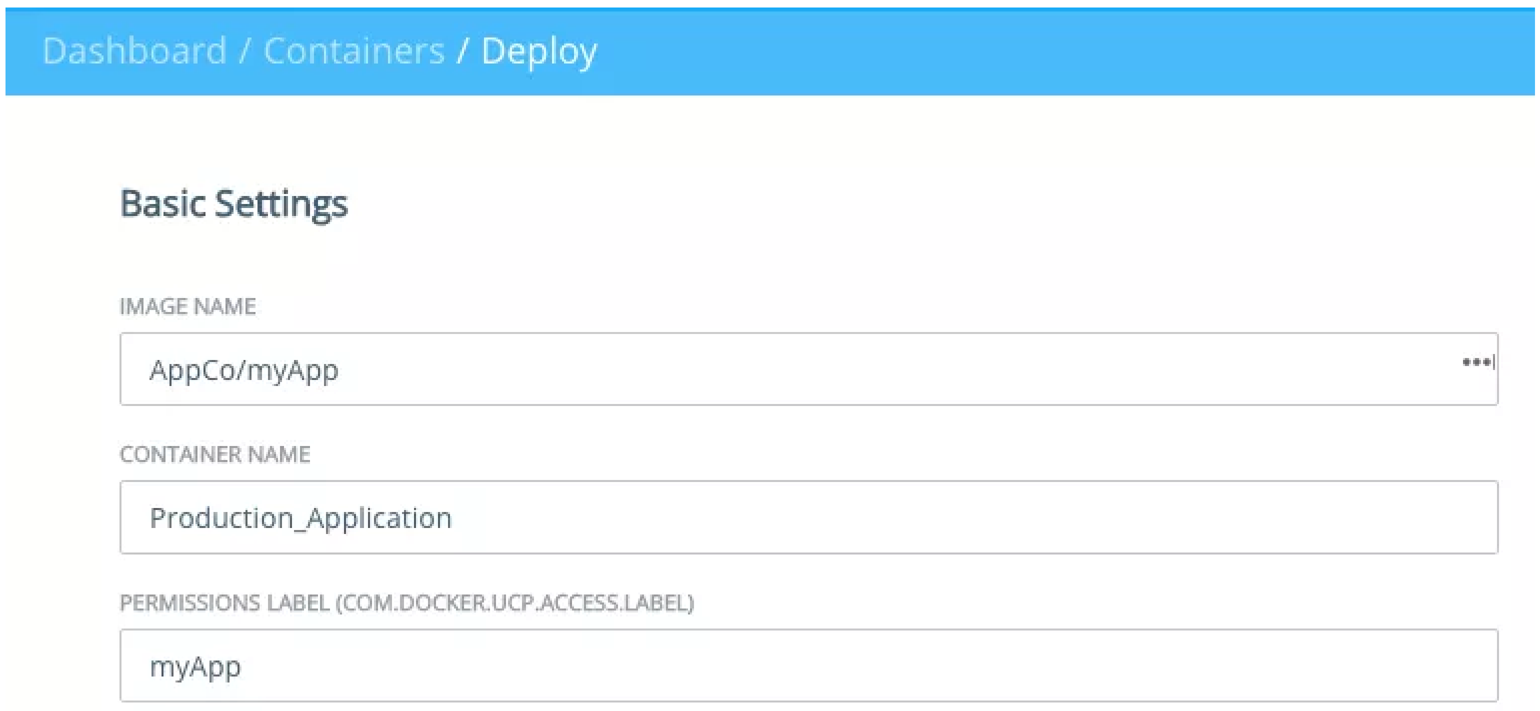


Control Panel sidebar for different user default permissions

Enforcing Fine-Grained Access through Team Container Labels

Default permissions allows you to provide broad, coarse-grained access to resources. However, in most cases you will want to provide finer-grained access to specific containers. In addition, you may want to set preferences at a team level rather than individuals for easier organization and convenience. This is where label permissions come in.

In the case of AppCo, you (as an admin) realize that various teams are going to need different levels of access to any of the containerized components of the myApp application. Your first step is to ensure that anyone starting containers as a part of myApp uses an access label with the key `com.docker.ucp.access.label` and the value `myApp`. This can be done either through the UCP GUI (as shown below) or when running containers (e.g. `docker run -label com.docker.ucp.access.label="myApp"`).


Dashboard / Containers / Deploy

Basic Settings

IMAGE NAME: AppCo/myApp

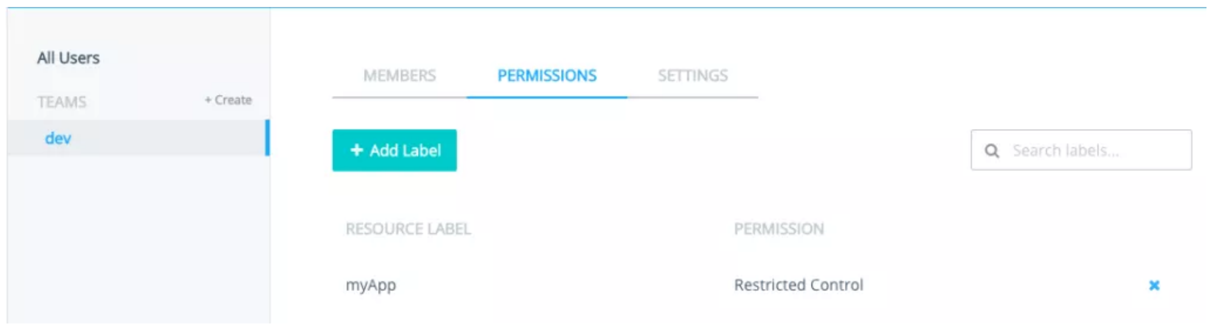
CONTAINER NAME: Production_Application

PERMISSIONS LABEL (COM.DOCKER.UCP.ACCESS.LABEL): myApp

Deploying an app with a permissions label

Next, you go to Team UI screen and start changing the permissions of each team:

1. The Biz Team wants to see the myApp containers in action, but you don't think they need permission to change the application. So, you give them "View Only" access to the "myApp" label.
2. The Dev Team needs to be able to start and stop myApp containers, but you don't want them to have edit currently running containers or get kernel-level access to the hosts. So, you give them "Restricted Control" access to the "myApp" label.
3. The Ops Team needs the freedom to make changes to the application as necessary. So, you give them "Full Control" access to the "myApp" label.



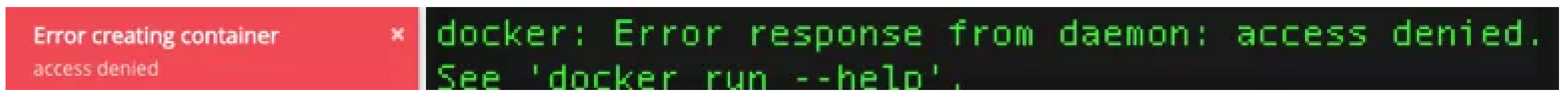
RESOURCE LABEL	PERMISSION
myApp	Restricted Control

Adding permission labels to a team

Keep in mind that any user who is a part of multiple teams with access to the same label gets the best possible access out of those teams. From the AppCo example, let's say the admin added user Jane to both the Dev and Ops teams. The Ops team has Full Control and Dev team has Restricted Control access to the myApp label. Thus, Jane has Full Control to the myApp label, which is the higher access out of those two teams. In addition, a non-labeled container will be visible only to the user who created that container, as well as to all admins.

Putting It All Together

Now, all the users of AppCo have both user-assigned default permissions and well as team-assigned label permissions. If a user tries to do an action that they do not have permission for, he or she will get an "access denied" error. This occurs whether the user tries this in the UCP GUI or in the CLI via Docker Client.


Error creating container
access denied

```
docker: Error response from daemon: access denied.
See 'docker run --help'.
```

Access denied in both GUI and CLI

We hope you've found this overview of Role-Based Access Control useful. We're very excited about RBAC and look forward to building more enterprise-grade features for Universal Control Plane in the future. If you have any questions please feel free to discuss on the UCP forums.

Additional Resources:

- Try a free 30-day trial of Docker Datacenter
- Get training on Docker Datacenter
- Register for the next Docker Datacenter demo

Learn More about Docker

- New to Docker? Try our 10 min online tutorial
- Share images, automate builds, and more with a free Docker Hub account
- Read the Docker 1.10 Release Notes
- Subscribe to Docker Weekly
- Sign up for upcoming Docker Online Meetups
- Attend upcoming Docker Meetups
- Register for DockerCon 2016
- Watch DockerCon EU 2015 videos
- Start contributing to Docker

Continue reading...

Tutorial: Role Based Access Control in Universal Control Plane

By Vivek Saraswat
Vivek works in product management at Docker, where he helps enterprise customers to manage their containerized applications. Prior to Docker, Vivek held product roles at VMware and AWS and was an engineer in a past life. He also enjoys singing and gaming in his spare time. Vivek tweets at @theVSaraswat.

Feedback

4 thoughts on "Tutorial: Role Based Access Control in Universal Control Plane"

Ravi Upad
January 6, 2017 at 2:18 pm

Do we need to create the tag com.docker.ucp.access.label even at services level?

Reply

Dhaval
June 27, 2017 at 11:41 pm