Rancher Part 4: Using the Catalog Example with GlusterFS

As promised, it’s time to get to the catalog goodness. Since GlusterFS hasn’t gotten enough love from me lately, it’s time to bring that into the process as part of my Rancher demo. The goal with this exercise is to spin up a GlusterFS environment and to expand it using the built-in capabilities provided by the catalog.

Deploying an Application from the Rancher Catalog

Go to the Applications menu option and select Catalog from the secondary menu. That will bring you to the built-in application catalog screen:

There are a neat array of applications to choose from which target a number of typical core services. In the left hand pane, you can see that they can be searched and sorted by category. The default shows the full list.

We want to try out a GlusterFS demo because it will show us how a multi-container stack is deployed, plus we get to expand it as well using the Rancher UI. Find the GlusterFS link in the catalog and click the Launch button:
We have a few options available as you would expect. We can choose the name of the stack, a free-form description text field, and also the

What’s even cooler is that you can expand the Preview link to expand the window. This will give us the two YML files that could be used from the command line interface to create this same cluster:
You can see the contents of the sample YML files here for each of the docker-compose and rancher-compose that I've copied out to GitHub gists for better viewing:

**Docker Compose:** [https://gist.github.com/discoposse/4cb03c3abfa60d1d9a40](https://gist.github.com/discoposse/4cb03c3abfa60d1d9a40)

**Rancher Compose:** [https://gist.github.com/discoposse/2f7c1a24aca645a93567](https://gist.github.com/discoposse/2f7c1a24aca645a93567)

Once you complete the wizard, the GlusterFS stack will spin up and you will see the results within a couple of minutes.

You can look at your running stack by clicking on the name in the Stacks view. This will bring up the screen showing some details about the application, commands, status, and also the containers further down the page.
Yes, it is just that easy.

Scal up the GlusterFS Cluster in Rancher

Now that we have our GlusterFS environment running, we can expand it by a couple of nodes to illustrate how easy it is to scale the application.

In the Stacks view, you simply click the \texttt{+Scale Up} button
As you wait and watch, the view will update showing the network address, and container status. Once completed, it will look something like this:
You can do it again to show just how easy it is to add another node:

To confirm that the application itself has recognized the new nodes, let’s spark up a terminal shell and see how the health of the GlusterFS cluster is. You can do this easily by hovering on the container in the view and the clicking the context menu which brings up the **Execute Shell** option:
We can use the `glusterfs pool list` which will tell us which servers are actively attached to the GlusterFS pool. As you can see from the example here, there are 5 nodes in the pool:
To manage the scale level, including reducing the scale manually, you can use the + and - buttons that appear in the Stack details view:
Join us in our upcoming 5th post in the series as we explore the advanced container configuration options. This is a great opportunity to explore core Docker concepts as well, so put on your thinking caps for that one!

Rancher Part 3: Adding the DockerHub to our Rancher Registry

So far in the series, we've covered installing your Rancher instance on Ubuntu, and attaching Rancher to the Docker host on Ubuntu. Next up we want to be able to stretch our wings a bit and add some other container images.

Rather than downloading the images individually and having to then work from a local instance, Rancher allows us to connect to an external registry which can be the DockerHub, Quay.io, or a custom registry which could include private Docker hub instance.

**Adding Docker Hub as a Registry in Rancher**

In the upper right corner, you will see your user icon. Click the down arrow which will bring up the menu, and select the **Registries** option:
In your **Registries** screen, click the **Add Registry** button:

You can see that the default setting button in our case is the DockerHub, and you could select others as well, but for now let’s start with the public DockerHub. You’ll need a DockerHub account which you can get by registering at [https://hub.docker.com](https://hub.docker.com). Use the same credentials to add the DockerHub registry in Rancher:

That will bring you to the **Registries** screen again and you’ll see DockerHub with the state as **Active**:
That’s the first part of the fun, so let’s up our game and add a container in Rancher straight from DockerHub.

**Using DockerHub Images with Rancher**

Let’s start by going to [https://hub.docker.com/](https://hub.docker.com/) and logging in. This will be helpful as we pick which container images to try out:

As it turns out, I happen to have a test image already stored in my repository called **discoposse/apache-test** which is a good place to start:

In your Rancher UI, click the **Infrastructure** top menu and then the **Containers** link on the secondary menu, then click the **Add Container** button:
The Name and Description options are free form text. For the sample, I’m calling this Apache Test, and the description can be anything you want. Under the **Select Image** field, you have to type the same name that appeared in the DockerHub in the form of `[username]/image` but sometimes you may find some that are also “official” images that won’t have a username at all.

The example for us is **discoposse/apache-test** as shown here:

In the **Hosts** view you will see the container as it is being added:
Once it’s completed, you can click the name to bring up the detailed view:

That’s one example which gives you an idea on a user-configured container that has been uploaded to repository. Let’s also grab another image from one of the official images that we find in the DockerHub.

I’m a fan of nginx, so that’s a great place to start. In the DockerHub, use the search field to find
nginx which will bring up a number of results. At the top of the list will be the official nginx image provided by, yup you guessed it, nginx:

Back in Rancher, go to the Containers view under Infrastructure and click the Add Container button to create our nginx container. Again, the name and description fields are free form which you can fill in as you wish, but for the Select Image field, type in nginx:

Under the Containers view you will see our list of active containers that includes both the Apache test and our freshly launched nginx instance:

Click on the nginx-01 name, or whatever you happened to have named yours, and this will bring up the detailed view:
This is a great way to extend our Rancher capabilities to allow anyone using the system to launch containers directly from the public repositories. All with a few simple steps.

What’s Next?

Now that we have a DockerHub registry, a running Docker host, and a basic understanding of how to launch containers from the local images as well as from the DockerHub, it’s time to start using the Catalog view next. We can then get into building out some more examples using pre-built applications from the catalog in Part 4, and then begin to do advanced container configuration in Part 5.

---

Rancher Part 2: Adding a Docker Host to Rancher

Now that we’ve gotten Rancher up and running from our first post, it’s time to take the next steps and configure our first host. We are going to run Rancher pointing to the same host that we are running Rancher on. That may seem like some sort of Docker inception thing, and definitely not the recommended way to run in production, but it’s a great way to test the waters on how it will work.

Adding your First Custom Host in Rancher

Using the Rancher UI, we can add hosts by clicking on the Infrastructure menu which brings you to the Hosts view. We will select the Add Host option and in the dialog, make sure to click the Custom provider at the top.
The instructions for adding our custom host are neatly laid out for us on the page, so our next step is to jump to number 4 on the list. Highlight and copy the code on the instructions page which we will paste into the console of our host:

```
root@RancherDemo:~# sudo docker run -d --privileged -v /var/run/docker.sock:/var/run/docker.sock rancher/agent:v0.8.2 http://rancherdemo.my.cyclingstore.com:8088/v1/scripts/3F1670FA31E3EB8
Unable to find image 'rancher/agent:v0.8.2' locally
v0.8.2: Pulling from rancher/agent
cb394f4c4a7a: Pull complete
c45bcb63430b7: Pull complete
c8ac8c62e03e38: Pull complete
f65d74652b89: Pull complete
624e6a4cd68d: Pull complete
42c4a1a2b6f638: Pull complete
814da5c763f4: Pull complete
494f517e19e1: Pull complete
ad2a3f313a92e: Pull complete
b5c028a526d4: Pull complete
1af8c63c5e303: Pull complete
5aaf08c12f38: Pull complete
c132b2b84e8c7: Pull complete
92b2acefd6c8: Pull complete
af4ac1609f9c: Pull complete
51894e46e9f6: Pull complete
b4d17832148c: Pull complete
Digest: sha256:db82ca7a6a109adeb27e6a6a362b4f92d3f4c335826679e2ac2c5cdcaec6d205
Status: Downloaded newer image for rancher/agent:v0.8.2
d940eac6b718b5f11b64455d435cecf147f760a3c5e9ee1d1b53beee1ff44e43d9371
root@RancherDemo:~#
```

That command will pull the Rancher agent container and spawn it onto the host to connect it to our Rancher environment for us. After a few moments, you will see the RancherDemo (or whatever your demo machine is named) appear in the **Hosts** view.
Now that we’ve added our host, we can see that there is already a standalone container running that has been detected. It’s our Rancher UI container running on the RancherDemo host. As you can imagine, it’s not a good idea to mess around with that one, so we will move on to test out Rancher in a few other neat ways.

Time to get started with some more Rancher goodness. The first thing we will do is to create a new container on the host to test that workflow out. Using the **Hosts** view, click the **Add Container** section:
This will bring up the new container dialog with some options we can apply to our container. We will start in the shallow end of the pool and work our way up. Let’s just name the container and pick the **ubuntu:14.04.3** container in the **Select Image** option. There will be only one image to choose from because we used the Ubuntu image to build our initial Rancher container.

There are a number of other advanced options, but we will explore that later on in another post. Now we will be put back to our **Hosts** view and you will see the container build. As the first container is built, another container will also appear called **Network Agent**. This is the networking container that sets up the container networking including routing and DHCP options.
You can click the Ubuntu-demo container in the **Hosts** view and it will bring you to the details pane with a number of charts showing IP address information, and some performance statistics in the active graphs.

In the upper right area of the details page, you can click the icon next to the state display, in this case where it says **Running**.
The options here include the control options to start, stop, delete, and also to clone, view logs, and view in the API screen. For our quick example, let's select the **Execute Shell**:

This brings up the console of our container, in this case running Ubuntu server and logged in with root privileges:
You have complete operational access just as if you had a full virtual machine. This isn’t really news, but it’s still pretty cool when you run apt-get inside a container that took a few seconds to launch:

Next we will close the shell and go to the **Containers** view on the **Infrastructure** tab. This is where we see the containers on our Rancher environment with details including container state (running, stopped), the container name, IP address, image name, and also any active command that is running.

Click on the **RancherDemo** container name which will bring you to the details view for that container. As you can see, there is more information here:
The Docker details appear in the Info box, and because it’s a Docker host, you can see the list of containers at the bottom of the details view. This is a handy way to navigate around the UI because each entity has details and drill-down capability based on the context of that entity.

**Thinking at the Application Layer**

The joy of moving up the stack is that Rancher lets you see multiple layers, and to build out applications within your container environment. Remember that the whole purpose of all of this infrastructure is to enable the rapid deployment and simpler administration of applications.

By clicking on the **Applications** menu at the top, and then the **Catalog** view in the secondary menu, you can see the **Application Catalog** that comes baked in with Rancher:

I know that you’re eager to keep going, but that’s the end of this post. If you want to click around the interface, I would recommend that. It’s good to be familiar with the various options, but don’t worry, we will also walk through lots of the features and views in the upcoming posts in this series.
What’s Next?

We have a limited number of containers and applications, so the next thing we will do is to connect the Rancher environment to the DockerHub to be able to pull down some other images and extend our testing a little bit.

Rancher Part 1: Installing Rancher and Setting Access Control

If you’re looking towards running Docker, and want a good way to use it with orchestration and a nifty UI, then a great place to get started is by using Rancher. For those who aren’t already familiar, Rancher and RancherOS are two very cool projects that allow a thin OS to easily spin up and run docker containers, and Rancher is a UI to run in front of those systems so you can visually create and manage stacks.

This is the first post in a series. There is a lot to go through, so this post will tackle the initial setup and configuration of the Rancher UI. You’ll need a couple of things to get rolling, including:

- An Ubuntu 14.04 server with AT LEAST 1GB of RAM
- Access to the internet from that server
- A GitHub account (we use GitHub for access control in this example…plus you should just have one for other awesome reasons)
- DNS record or IP address of your Ubuntu server

In my case, I’m running the instance in DigitalOcean. The reason that I’ve said “AT LEAST 1GB of RAM” is that early testing I did using less RAM proved to be fatal to the build. The web server will attempt to start and will run out of memory and restart, which can be profoundly frustrating.

My build is a 14.04 LTS Trusty Tahr machine with 2 GB of RAM and 20 GB SSD.

NOTE: The commands I have shown below require the sudo command, but you will notice from screenshots that my account is running as root and sometimes do not use sudo. That is because I’m connected via the console to the DigitalOcean server as root. It is always a best practice to use a standard account and sudo for commands.

Installing and Configuring Rancher

The first thing I always like to do is update my machine to be sure we are up to date. This is done with the classic sudo apt-get update command. Once you’re updated, we will get the ball rolling by installing Docker:

wget -qO- https://get.docker.com/ | sh

root@RancherDemo:~# wget -qO- https://get.docker.com/ | sh
That will run for a minute or two, and we will see the results showing the installed version upon completion:

Starting up the Rancher application is super easy because it is entirely encapsulated inside a Docker container. How cool is that?!

Let's spin up our Rancher application container using this command: `sudo docker run -d --name myrancher --restart=always -p 8080:8080 rancher/server`

If this is the first time that you've run the Rancher application container, you will see the download process happening which looks like this:
Once the container is downloaded and launched, we can monitor the build using the `sudo docker logs -f myrancher` command which will show you the log activity.

During the build, the screen will look like this:

```
[loading host-api] 2015-11-08 20:33:59,412 INFO [main] [ConsoleStatus] [28/102] [31284ms] [73ms]
Loading host-stats
[loading settings-api] 2015-11-08 20:33:59,547 INFO [main] [ConsoleStatus] [29/102] [31419ms] [135ms]
Loading api
[loading storage-service-api] 2015-11-08 20:33:59,756 INFO [main] [ConsoleStatus] [31/102] [31628ms] [84ms]
Loading encryption
[loading core-model] 2015-11-08 20:34:00,339 INFO [main] [ConsoleStatus] [33/102] [32211ms] [529ms]
Loading core
Loading process
```

Once the install is completed, you will see the phrase “Connection established” in the log output.
Yay! Now, let’s go to the URL and configure our access control. The reason that this is important is because the server has no access control at all on initial launch, and is publicly accessible on the http port.

Go to the domain name for your server or IP address using port 8080. In my case it was http://rancherdemo.mycyclingstore.com:8080 which brings me to the home screen:

The important warning is up at the top which states that Access Control is not configured:

Just click the Settings link and we will get your initial access control set up. There will be a setup screen showing your options to configure, and in this case I want to use GitHub as my
authentication. Simply click the GitHub icon:

It even has all the instructions for setting up your GitHub application, so make sure to use those settings as shown on screen:

1. Setup a GitHub Application
   - For standard GitHub, click here to go applications settings in a new window.
   - For Github Enterprise, login to your account. Click on Settings, then Applications.
   - Click "Register new application" and fill out the form:
     - Application name: Anything you like, e.g. My Rancher
     - Homepage URL: http://rancherdemo.mycyclingstore.com:8080/
     - Application description: Anything you like, optional
     - Authorization callback URL: http://rancherdemo.mycyclingstore.com:8080/
   - Click "Register application"

In your GitHub account, set up a new OAuth application using the settings from the Rancher instructions. This was what mine looked like, and then we click Register Application:

You’ll see your completed settings in the results page on GitHub, and it will also have your client ID and client secret, so keep this page open. Now, go to the Rancher UI again and fill out the
information as provided by GitHub:

Once that’s filled in, click the handy Authenticate with GitHub button:

This will bring you to the GitHub authorization confirmation screen which will look something like this:

Congratulations, you’re all done with your initial access control settings! You can confirm by going to the Rancher UI and click Admin in the top menu and the default Accounts page shows your user configuration:
What’s Next?

We’ve done the initial setup here, so our next tasks will be setting up hosts and services, and spinning up some Rancher goodness in Part 2. The important first step was configuration and access control, and our upcoming posts will explore a few different host configurations and sample applications.

Remember that if you shut down your host, you will have to restart the Docker instance using the same commands we used above. Luckily, the launch is much quicker now that we’ve downloaded the initial components.