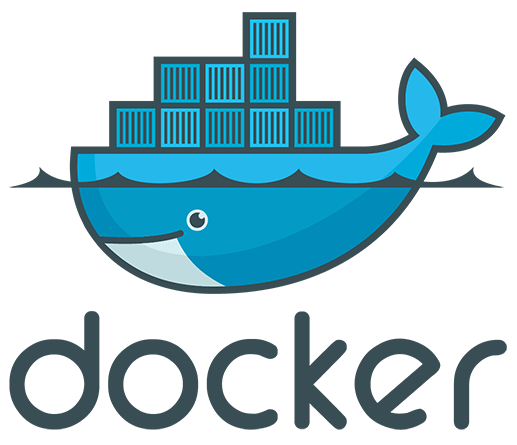
Build, Run, & Ship Containerized Apps with Docker Enterprise Edition

**Hands-on Lab Manual**

*This hands-on lab session will give you a brief introduction and lab experience for a Containerized Applications solution stack on Azure, built with Docker Enterprise and CloudBees Jenkins.*





# Introduction

The **Build, Run, & Ship Containerized Apps with Docker Enterprise** **Edition** Azure  
Partner Quick start template launches Docker Enterprise, an integrated platform that  
enables agile application development and management.

# What You Will Learn

# In this Launch & Learn session, you will learn how to leverage the Jenkins Continuous Integration/Continuous Deployment (CI/CD) platform to build and push a container-based application into the Docker Trusted Registry (DTR). Once the images are pushed, you will then leverage the Docker Universal Control Plane (UCP) to compose the application and run it on Docker Enterprise. You will also scale up the application and see how Docker Enterprise seamlessly scales your application.

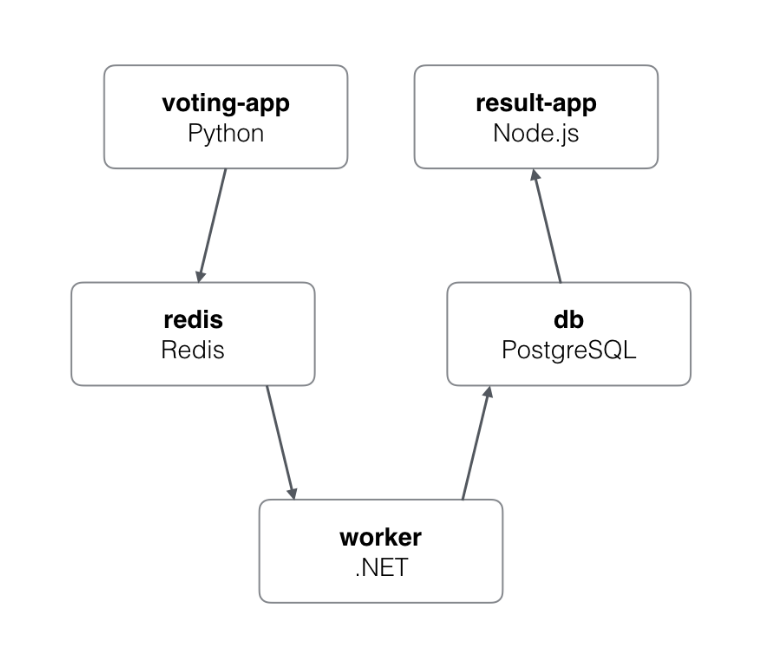
# CI & CD Workflow

# The diagram below shows a typical developer/operations (DevOps) workflow using Docker Enterprise Edition.

# 

# Sample Application

* To better understand the Continuous Integration and Continuous  
  Deployment (CI/CD) process, we will use the Sample Voting Application that  
  has been pulled from GitHub.
* The Components of the Sample Application are shown below:



* **Voting App:** A Python web-app to let you vote between two options.
* **Caching App:** A Redis queue to collect new votes.
* **Worker App:** A .NET worker to consume votes and store them in a  
  Postgres database.
* **Results App:** A Node.js web-app to show the results of the voting in  
  real-time

# Lab Prerequisites

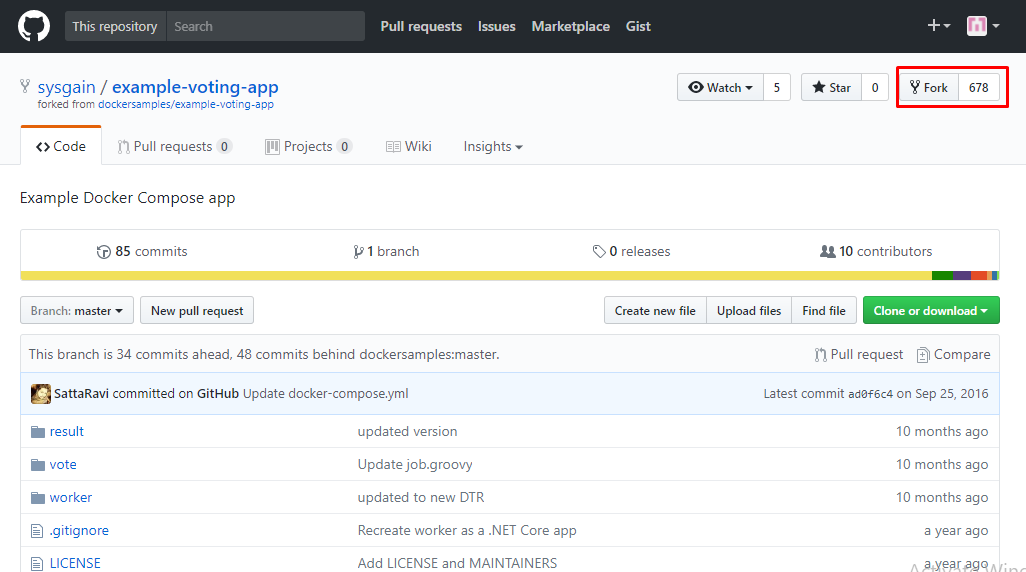
The following will already be provided by the lab instructor before or during the session:

* GitHub account.
* Voting App Git repository. ( here )
* A Jenkins environment. ( here )
* Jenkins Credentials plugin ID.
* Docker Trusted Registry. ( here )
* Universal Control Panel. ( here )
* Login credentials for all the above, except GitHub account.

# Lab Steps

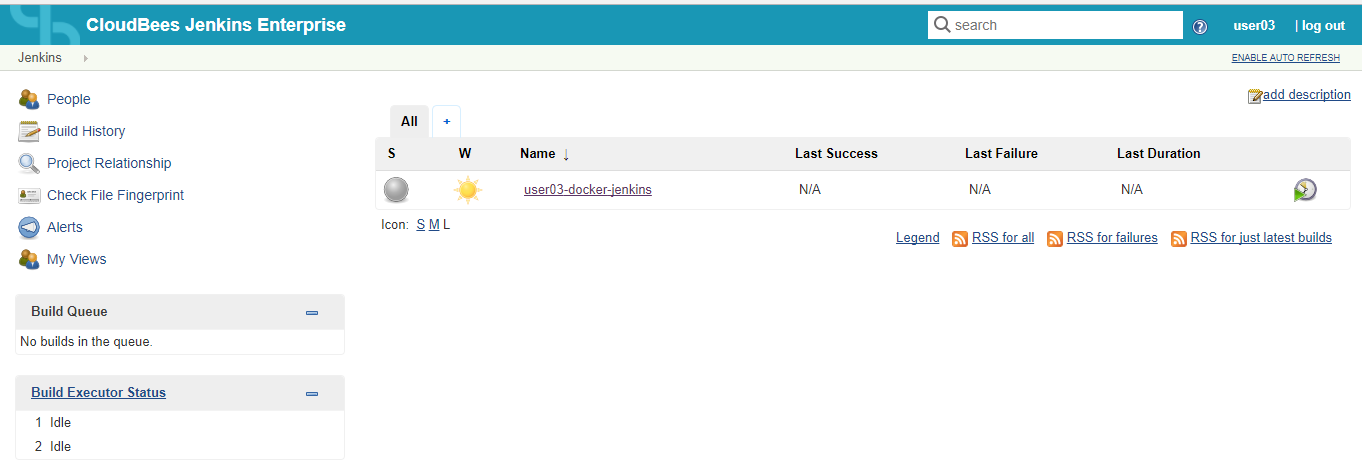
1. Login to your GitHub account (http://www.github.com) and fork the sample voting application by going to the below URL and selecting the **“Fork”** option in the top right of the page.

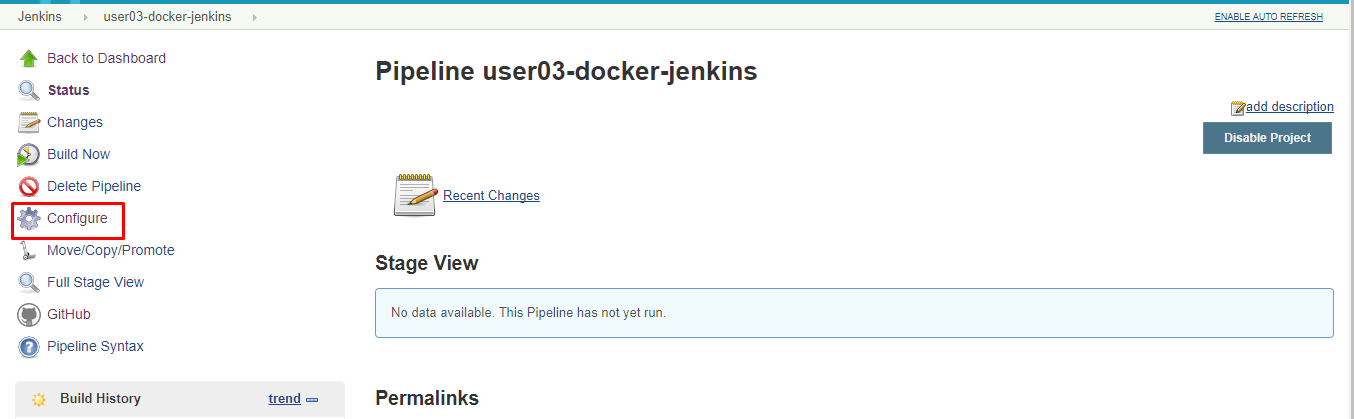
* Available here: <https://github.com/sysgain/example-voting-app>

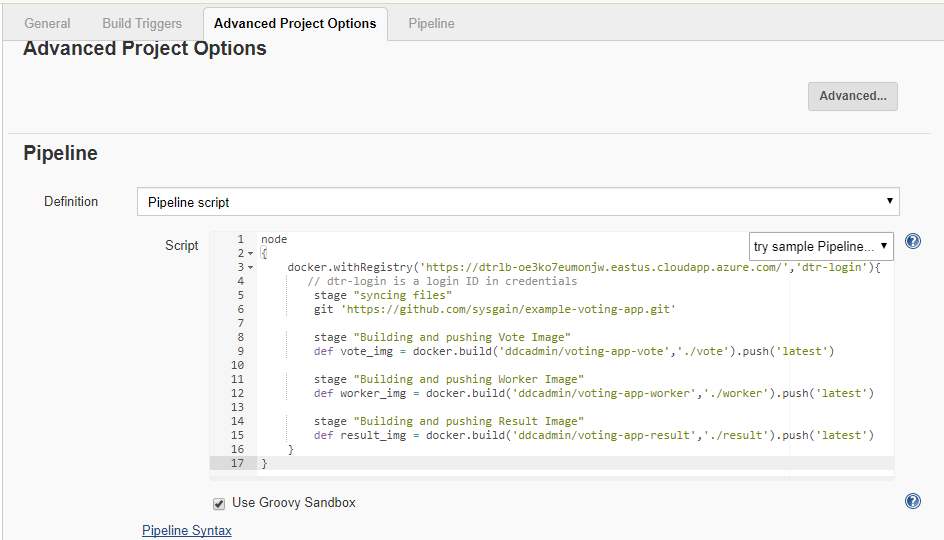


1. Once you’ve forked the repo, go to the Jenkins environment at (need to update the URL)

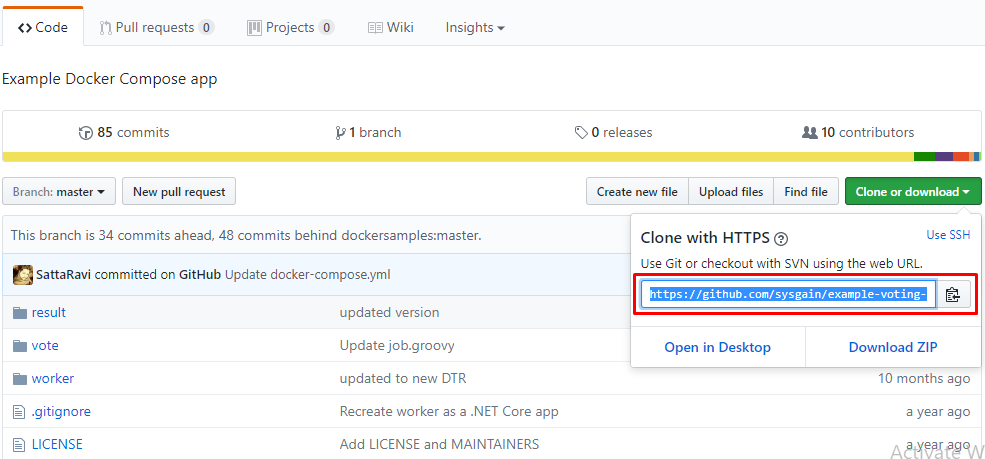
There, you can configure the job assigned to you (starts with the username assigned to you). Select the Job from the list, then select ‘**Configure’** from the left side navigation.





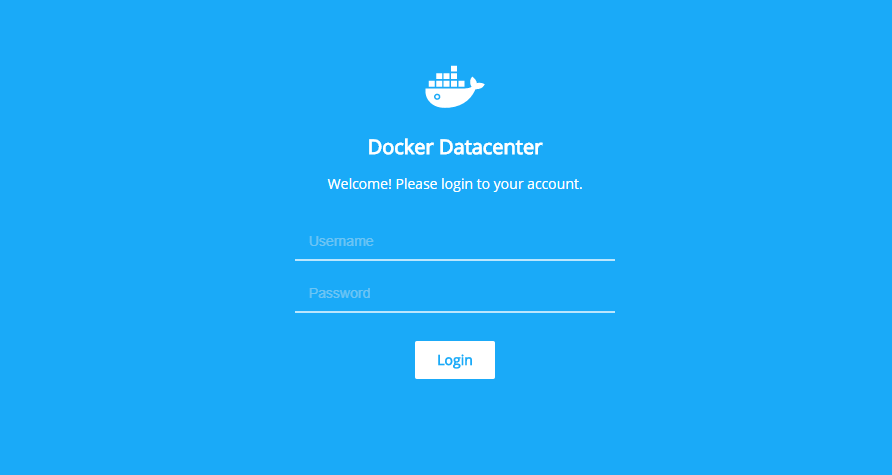


1. After selecting Configure, go to the bottom of the page and perform a search with Ctrl+F for “**usr-dtr-login**” in the Pipeline text box. Replace that on **line 3** with the DTR credential ID you were provided at the start of the session.
2. Next, in the same Pipeline text box, replace the **GitHub URL** on **line 6** with your forked GitHub repository's URL from step 1.

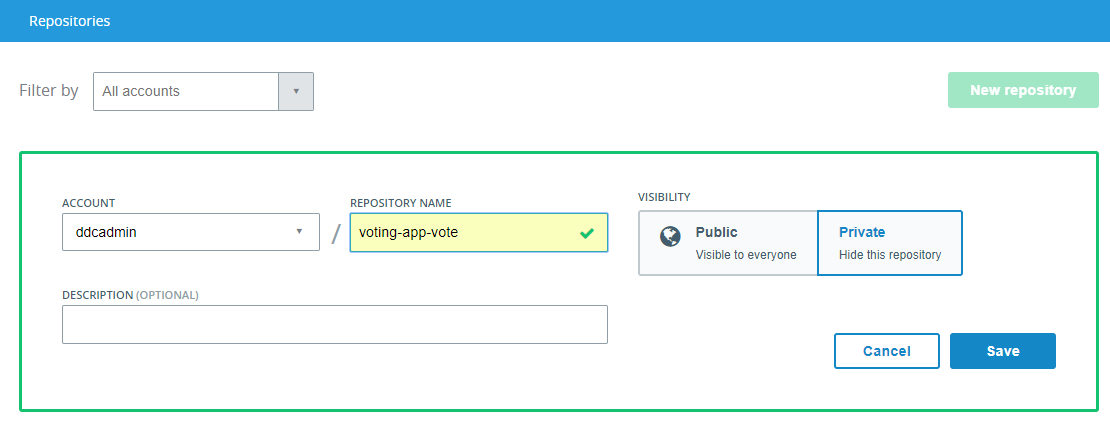


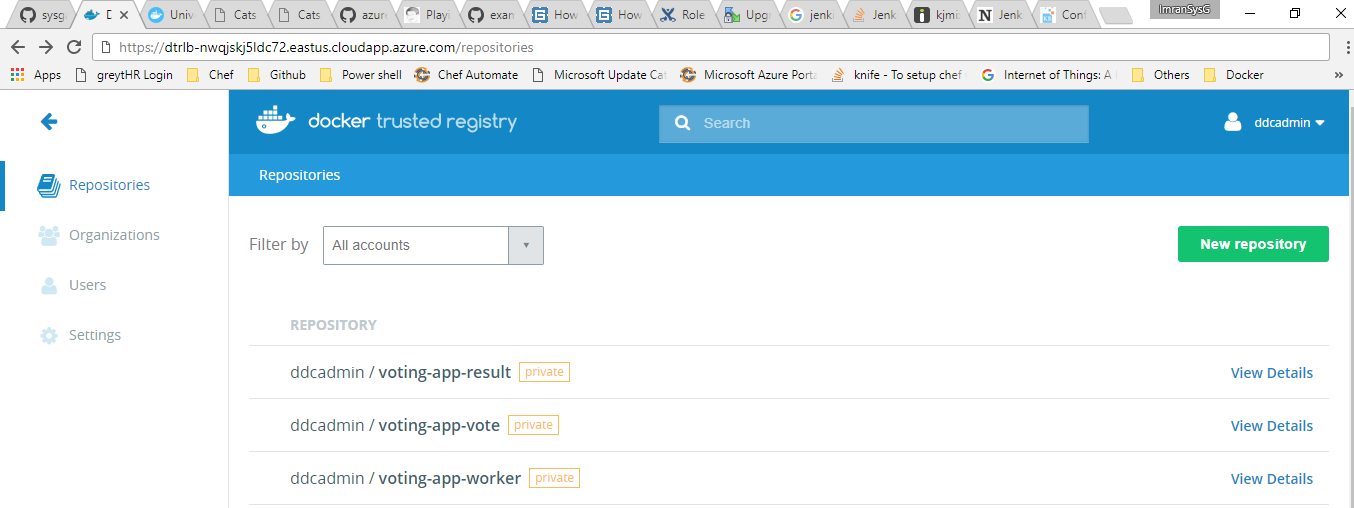
1. Next, replace the username **usr** from **lines 9**, **12**, and **15** with your user name.
2. Now select **save** to save your changes and close the job configuration.
3. Login to the DTR URL in a new tab/window (https://dtrnodednstpal2zkqvglmc.westus.cloudapp.azure.com/) and select ‘**Create’** create repositories with the following names:

* voting-app-result
* voting-app-worker
* voting-app-vote

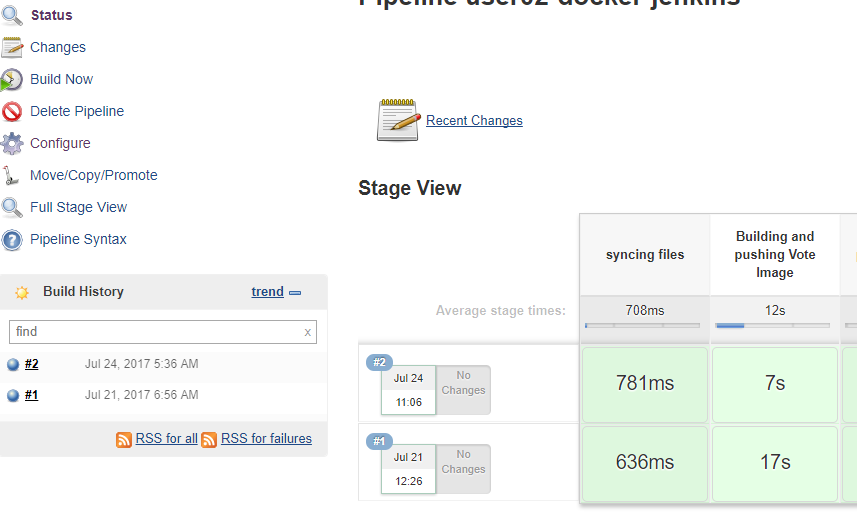
and make sure to set them as **private** (selected under ‘Visibility’). 

* Enter your User name and the password for the Docker Trusted  
  Registry at the login page. You can find the User name & password  
  in the same DTR section mentioned before – replace “**user2xx**” with  
  your assigned user name. (Example: “user230”)



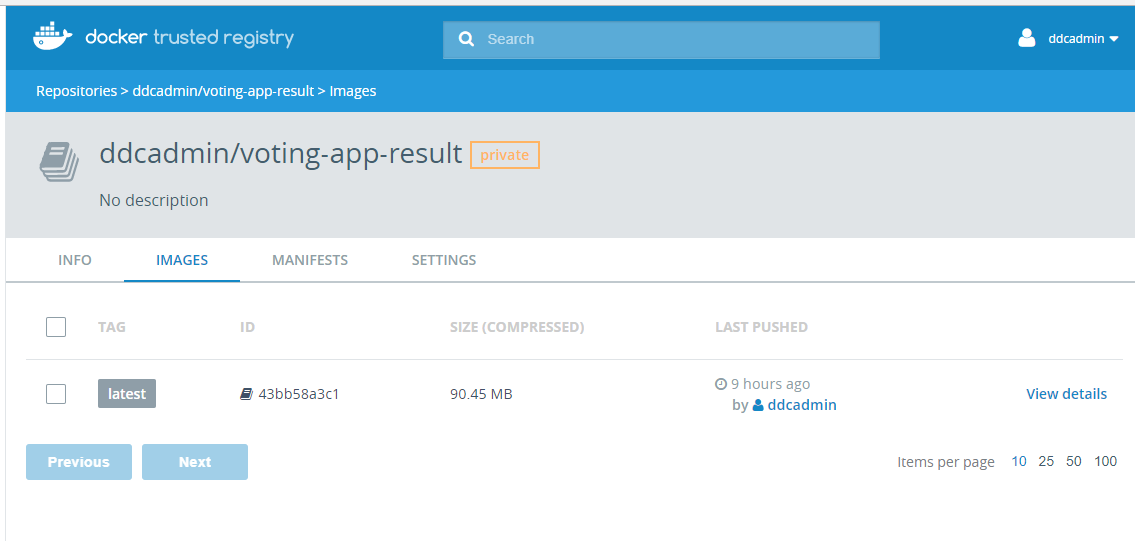


1. With this in place we can now build Docker images to be pushed to the DTR using the Jenkins job you configured earlier. Return to the Jenkins page and click on ‘**Build Now’** button on the left side navigation to trigger the build process. This process builds the 3 images and pushes them to the DTR.



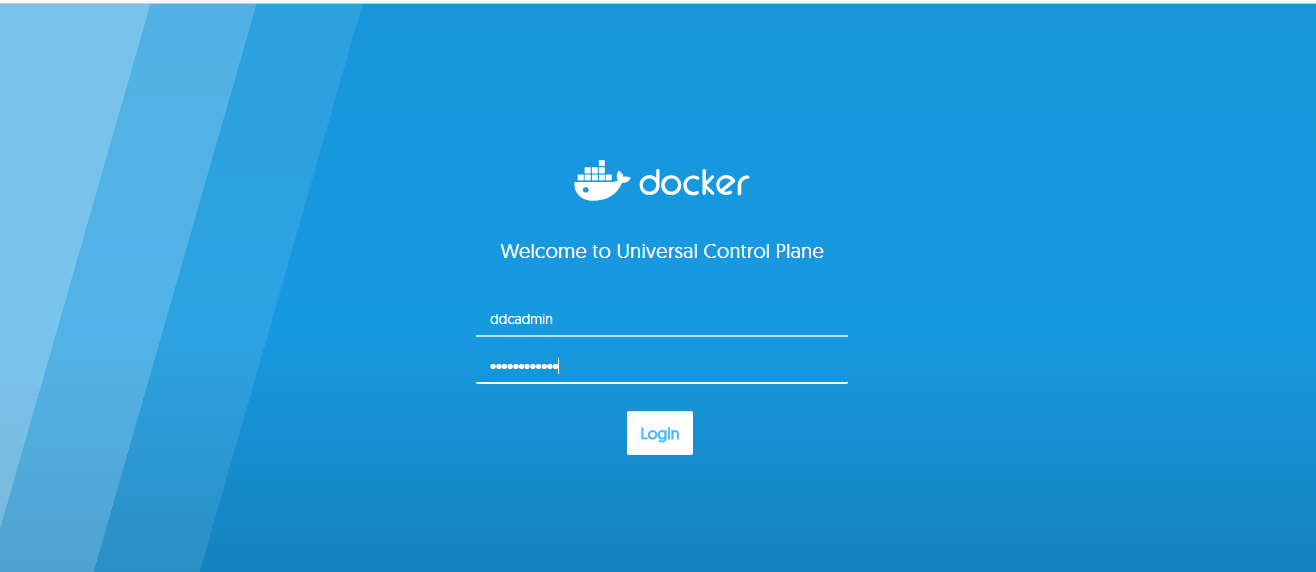
1. Once the job succeeds we can find the images being pushed to the repositories we created on DTR from step 8.

* Select one of the images. It will show details about that image, such as its tags, permissions, and settings – as shown below.

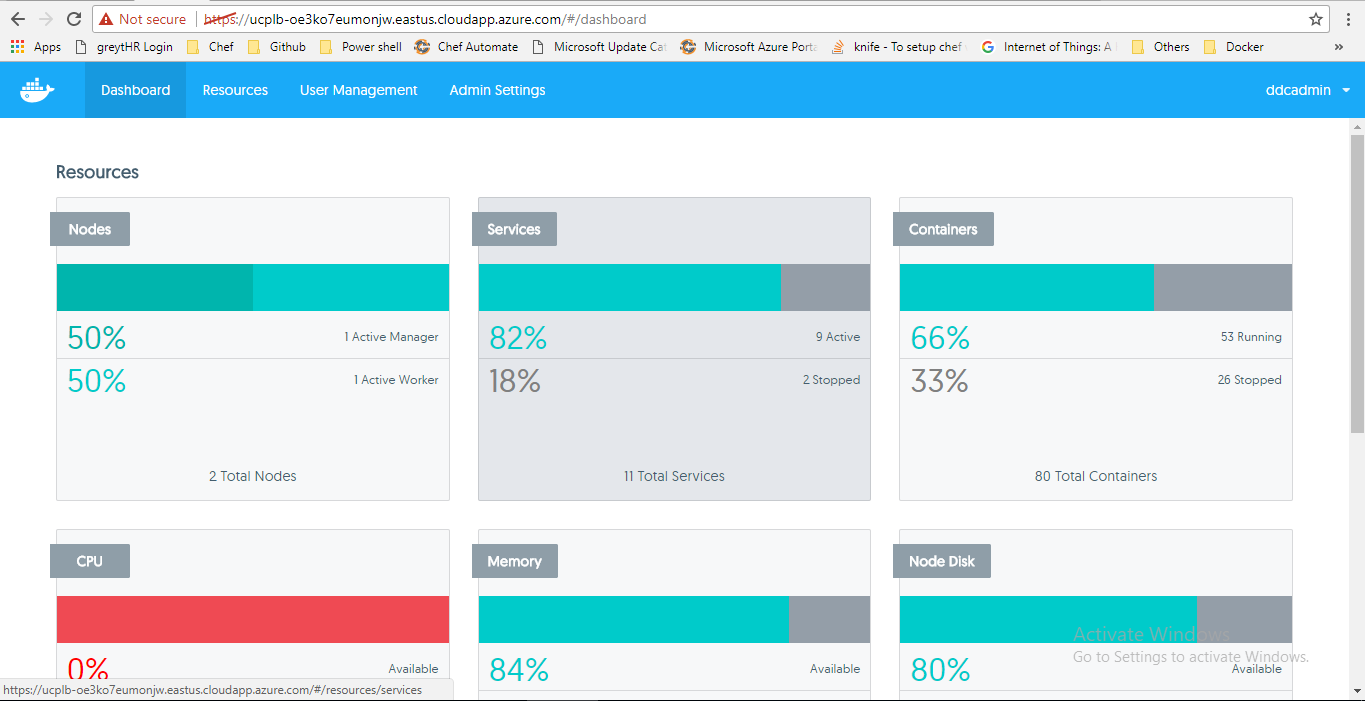


Now that you’ve seen these images in DTR, in the next section we will see how these images will be deployed as a containerize application in Docker Enterprise.

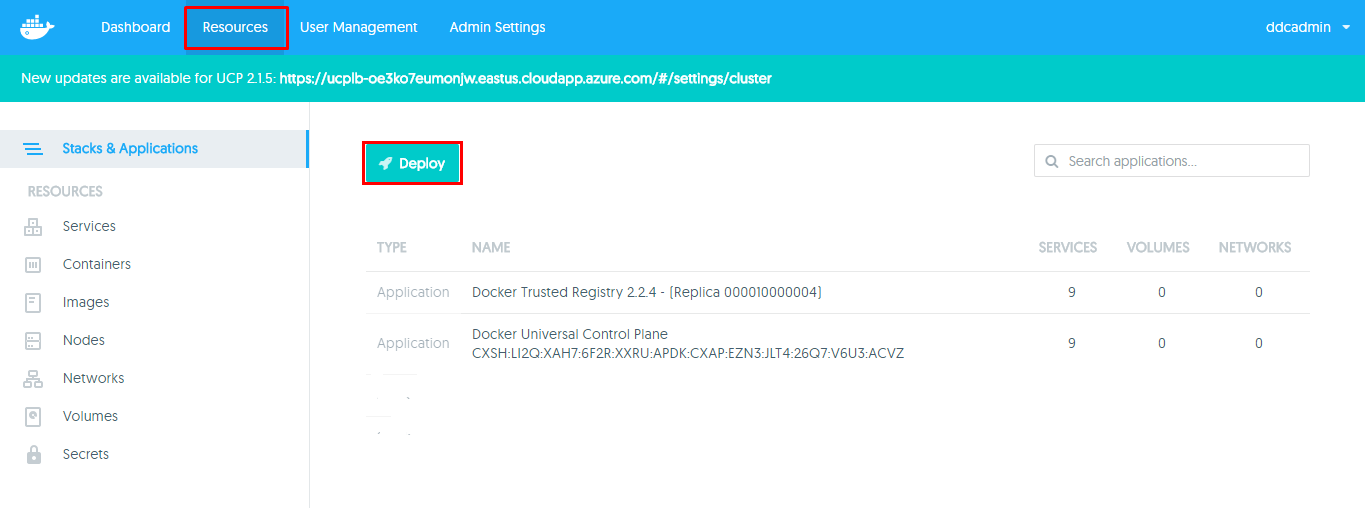
1. With the images pushed, now you can deploy the voting application from the UCP. Login to the UCP URL. Open the UCP URL in a new browser tab/window. The UCP login page will look like the image below:



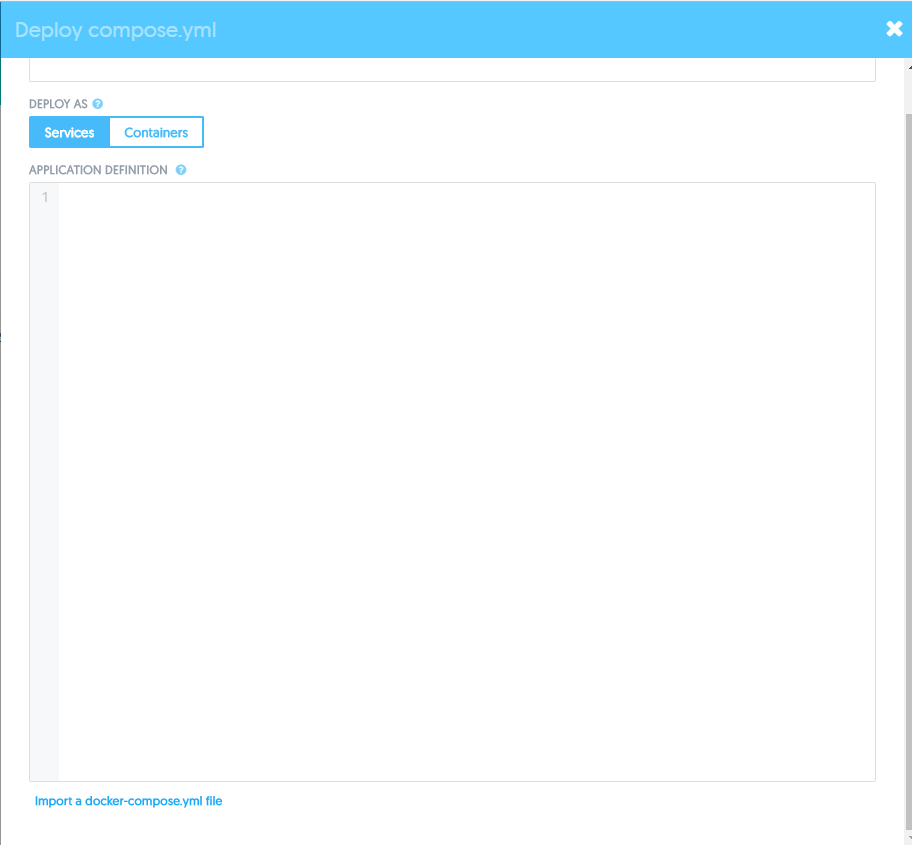
1. Once you have logged in to the UCP, the screen should look like the image below:



1. Now that you have logged in to the UCP, you will now deploy the **Sample Voting Application.**
2. Select ‘**Applications**’ from the left side navigation menu in the UCP window. Then select the ‘**Compose Application**’ button.



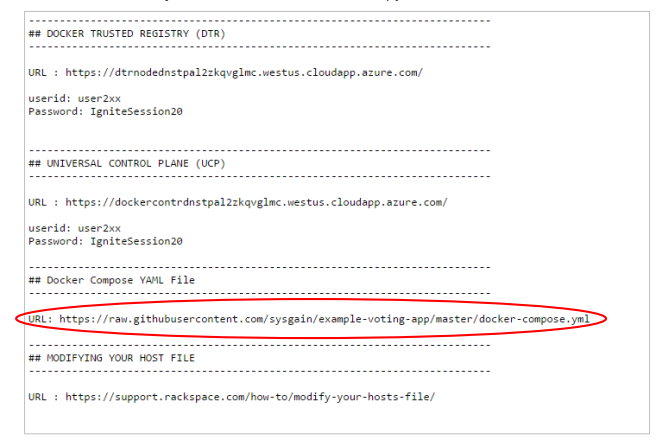
1. The **Create Application** dialog box should appear, as shown below:



Keep this browser tab open, as we will return to it in the next section.

* 1. Return to the GitHub URL and go to the **Docker Compose YAML File**section. Copy the contents of the **docker-compose.yml** file to compose  
     and deploy the application.

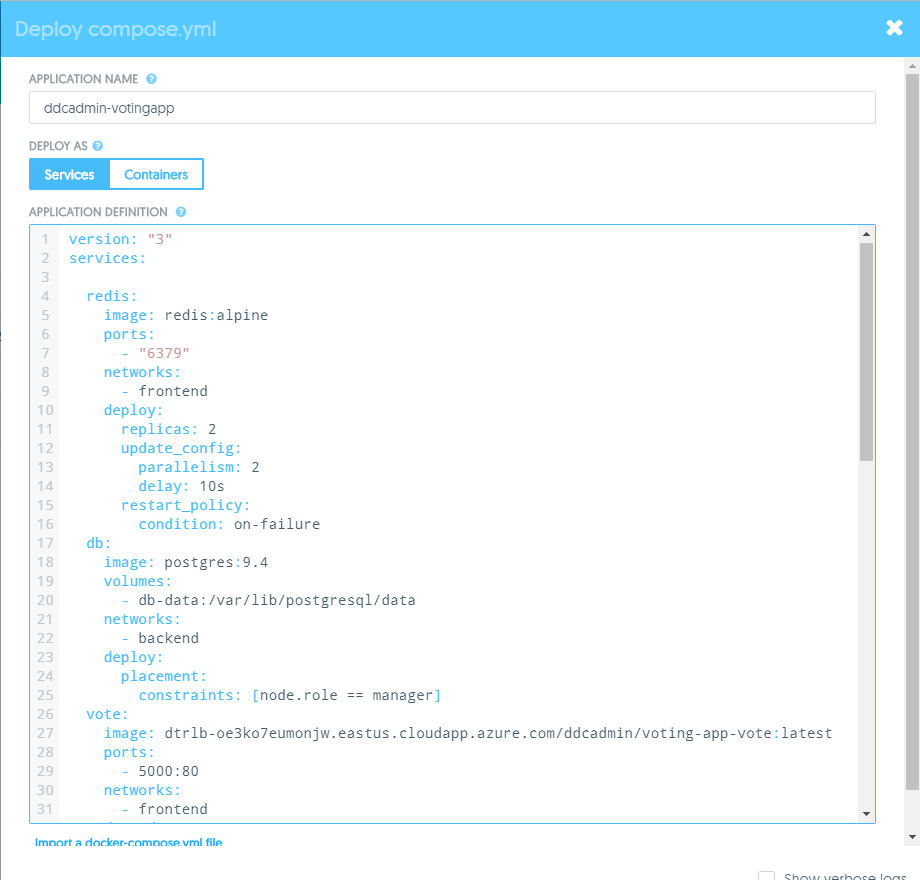
1. (Need to update the credentials and URL’s screen here) Return to the GitHub file you opened in Step 2, navigate to the **Docker Compose YAML File** section, and copy the URL, as shown below (next page):



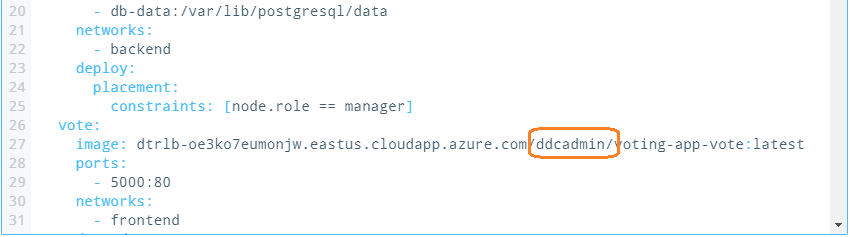
1. Open the URL in a new browser window/tab. This should show the  
   docker-compose.yml file contents.
2. Copy the entire contents of the **docker-compose.yml** file, as shown below:



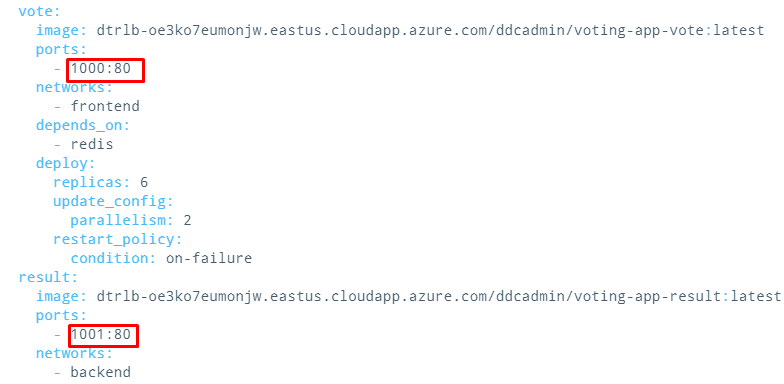
1. Then, return to the UCP window/tab in your web browser and paste the  
   copied docker-compose.yml file code into the window as shown below:
2. For the **Application Name** field, enter a unique name with the  
   following format, as shown below:
   1. [your\_username-app\_name]



1. Go to lines **27, 41, and 55** in the docker-compose.yml window and  
   replace the user “**admin**” with your own username and change the port numbers, as shown below:



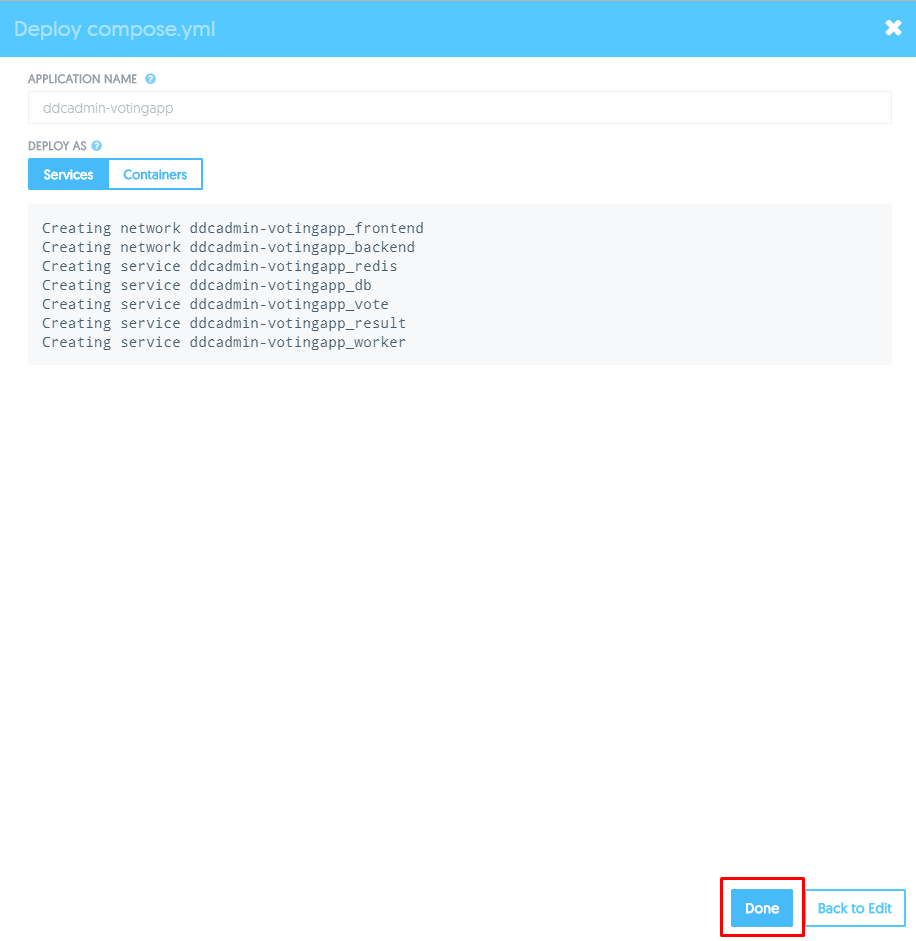
1. Go to lines **29 and 43** in the docker-compose.yml window. The default outbound port number will be 5000 & 5001. Replace the ‘5’ with the user number: if the user name is **user20,** the port should be 20000 and 20001, respectively. If the user name is **user01** then change the number should be 1000 and 1001, as shown below:



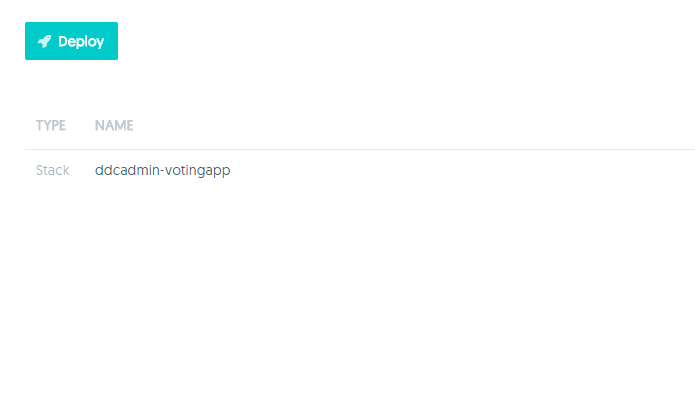
1. Go to line **71** in the docker-compose.yml window and change the port number as per the user. If the user name is **user01**, change it to 8001:8001, as shown below:



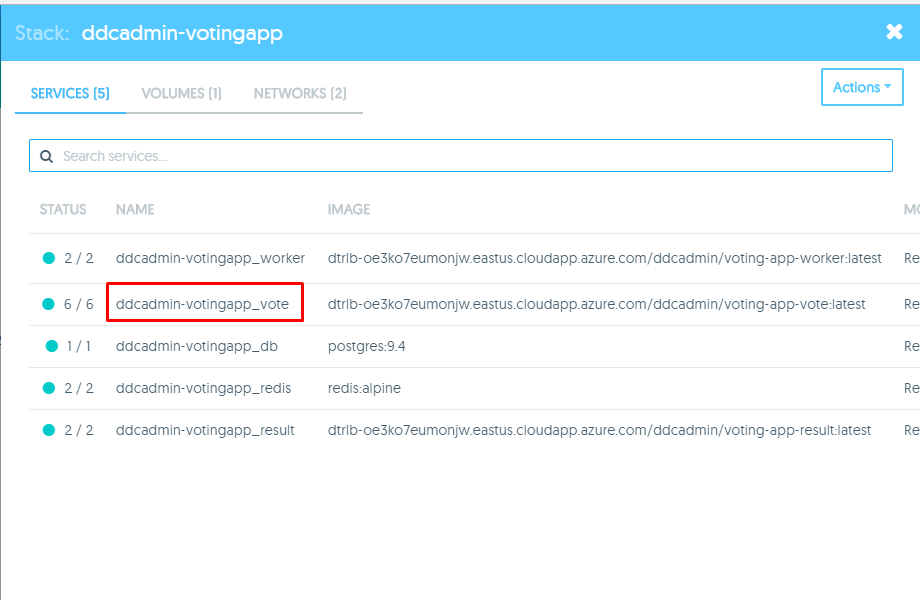
1. Enable the **Show Verbose Logs** checkbox near the bottom of the  
   dialog box, and click **Create** to deploy the application.
2. Once the deployment succeeds (it may take a few minutes) you will  
   now see that your application deployed successfully, as shown below:



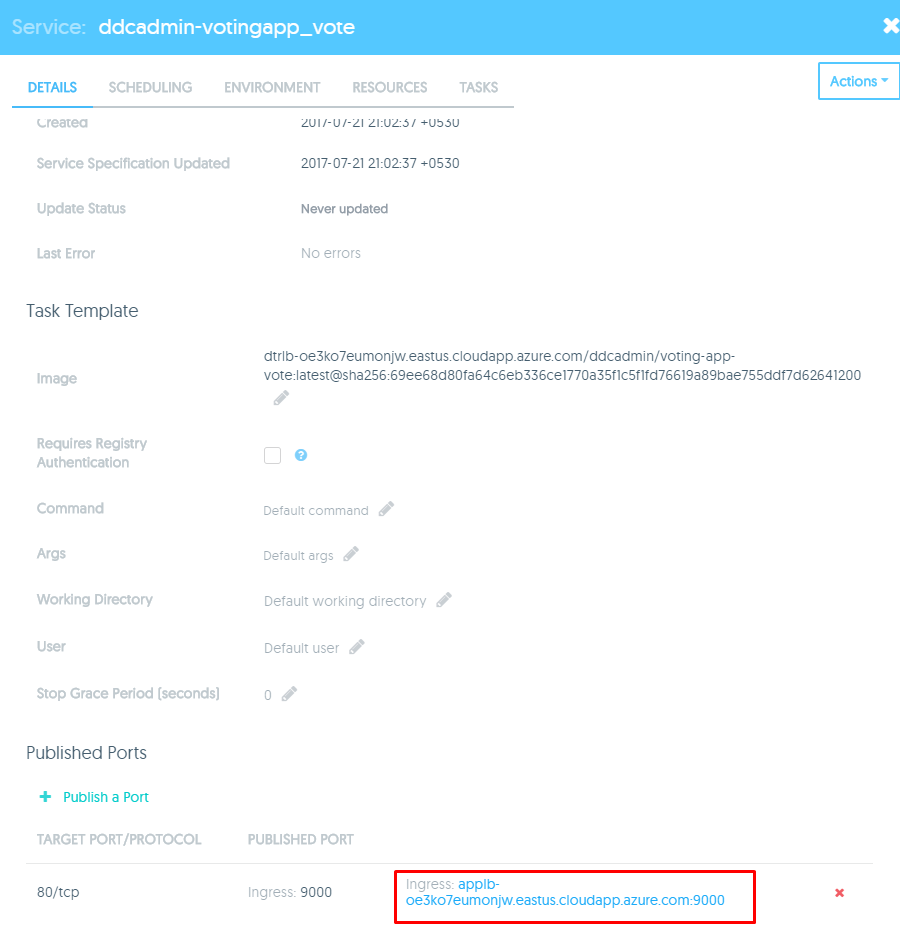
* Click the **Done** button to see your application deployed in the  
  Universal Control Plane, as shown below:



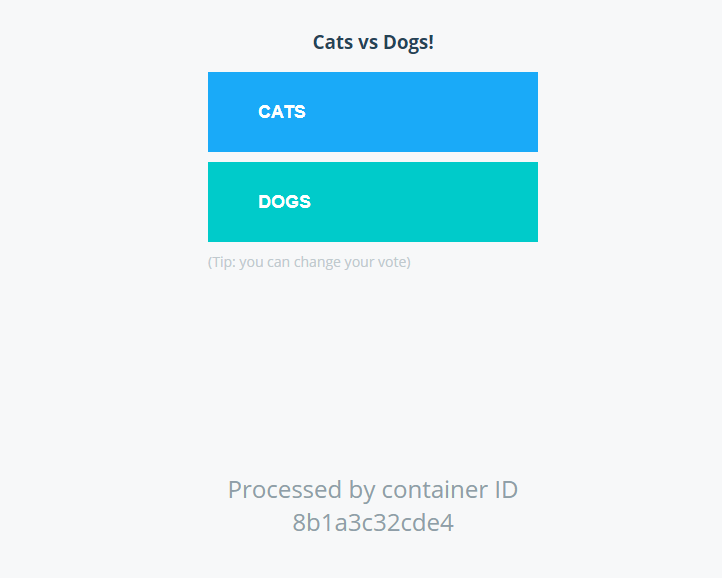
* 1. Click on the app that we have just created, then click on the vote item, as shown below.



* 1. On the voting item screen, scroll down and find the URL. Click through to vote for Cat or Dog.



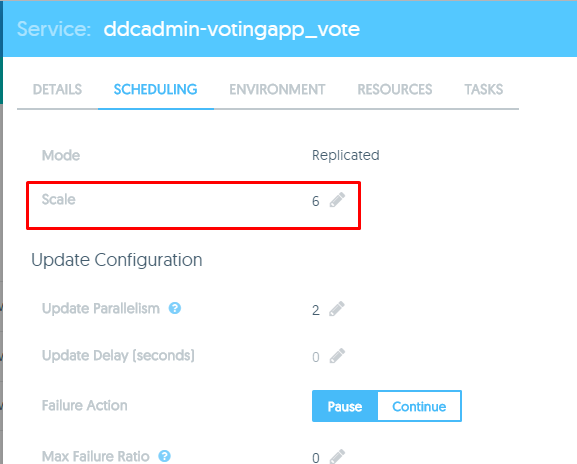
* You should see the running **Voting Application**, as shown below:



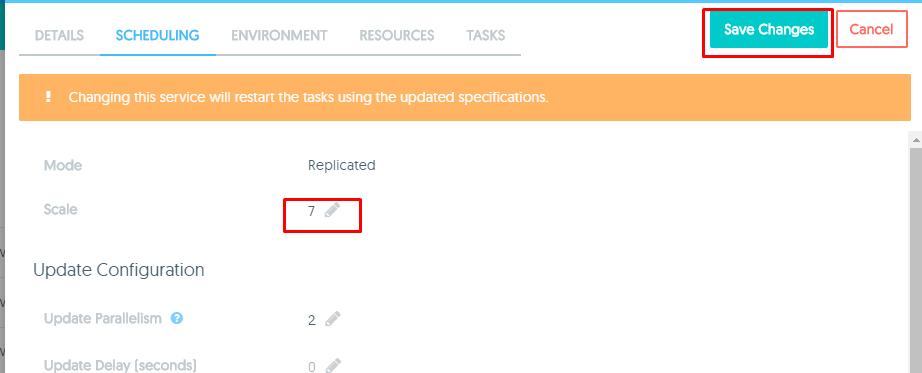
* In the application, go to result item and click on it. Scroll down to get the result URL. Copy and paste that URL in the browser to see the result.



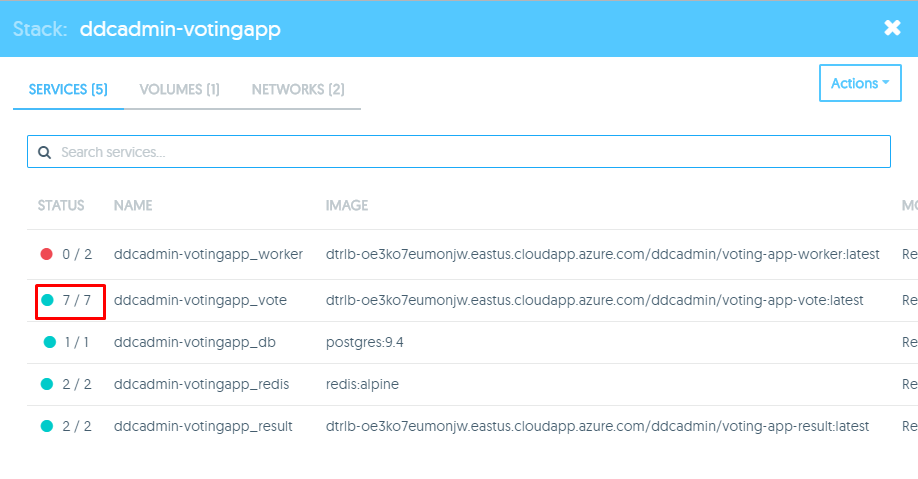
* Congratulations! You now have both the Voting and Results  
  applications running.
  1. Next, we will scale up the Voting App to 7 containers, because we believe we need additional resources to handle the load.
* Return to the UCP web page and select ‘**Resources’.**
* Go to your app and select the “voting” item. In the next pane, go to scheduling, where you see the scaling.



* Change the scaling to 7 and save it.



* You should see that the total containers running on your application  
  have increased by 1.



* **Congratulations**! You’ve now scaled up your containers running in the  
  application you created in UCP.