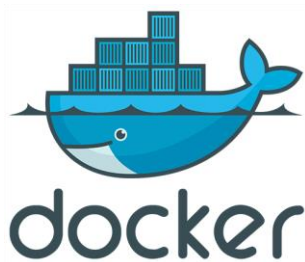


Build, Run, & Ship Containerized Apps with Docker Datacenter

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 Datacenter and CloudBees Jenkins.



Introduction

The **Build, Run, & Ship Containerized Apps with Docker Datacenter** Quickstart template launches Docker Datacenter, an integrated platform that enables users for agile application development and management. This Launch & Learn session will introduce you to Azure Partner Quickstart templates, how they are made, and how this specific Docker Datacenter-based template is used to enable container-based microservices workflows with a multi-product solution stack.

What You Will Learn

The goal of this solution stack is to provide a continuous integration experience through creating Docker images of any code change and deploying the solutions as containers. By walking through the lab in this Launch & Learn session, attendees will learn how this Quickstart template can be used to build and deploy modern container-based applications pilots on Azure, with continuous integration/continuous delivery (CI/CD) enabled.

Solution Summary

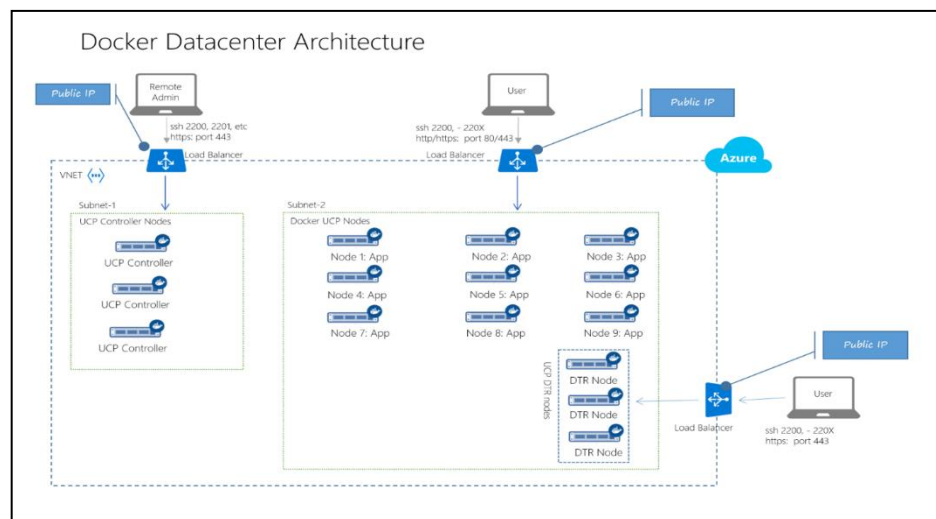
The Build, Run, & Ship Containerized Apps with Docker Datacenter Quickstart template allows users to deploy solutions as containers. This is achieved using products from multiple partners (e.g. Docker, CloudBees Jenkins) and integrating them in an automated way. The core component of this stack is Docker Datacenter. This stack is an integrated solution including open source and commercial software, the integrations between them, full Docker API support, validated configurations and commercial support for your Docker Datacenter environment.

A configurable architecture allows flexibility in compute, networking and storage providers used in your Containers-as-a-Service (CaaS) infrastructure without disrupting the application code. Plus, open APIs allow Docker Datacenter CaaS to easily integrate into existing systems like LDAP/AD, monitoring, logging and more.

You can find more information here: <https://www.docker.com/products/docker-datacenter>

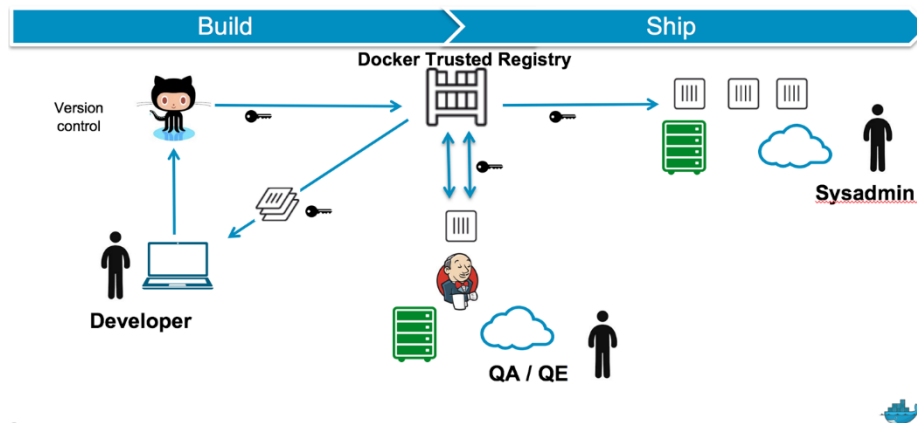
Docker Datacenter consists of 3 components:

1. **Docker Universal Control Plane (UCP):** UCP is an enterprise-grade cluster management solution from Docker that helps you manage your cluster using a single plane of glass. It is architected on top of a swarm that comes with Docker engine. The UCP cluster consists of controllers (masters) and nodes (workers).
2. **Docker Trusted Registry (DTR):** DTR is the enterprise-grade image storage solution from Docker that helps securely store and manage the Docker images used in applications. DTR is made of DTR replicas only that are deployed on UCP nodes.
3. **Commercially Supported Engine (CS Engine):** The CS engine adds support to the existing Docker engine. This becomes very useful when patches and fixes will have to be backported to engines running in production, instead of updating the entire engine.



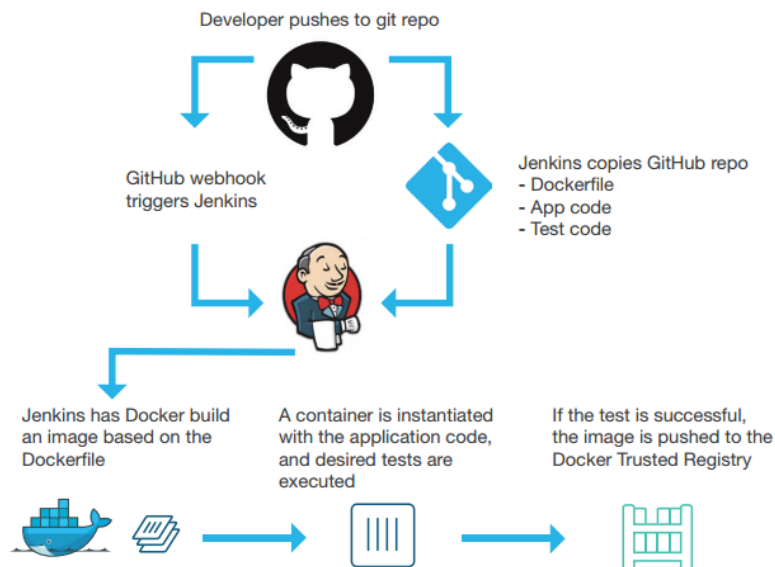
CI & CD Architecture

- We are Going to provision a CI & CD piloting environment using Docker Datacenter, Jenkins and GitHub.



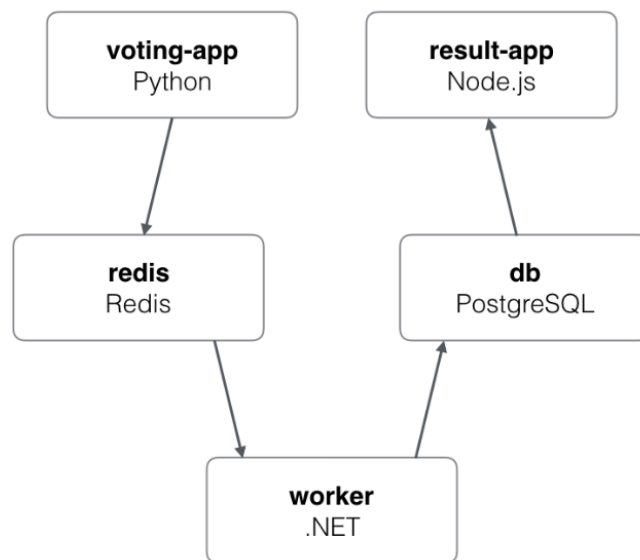
- The above image gives a solid graphical representation of a typical software development team, and their roles.
- Sysadmins make hardened infrastructure images available to be consumed by developers.
- Developers use the hardened images and build their software on top of them to be consumed by QA and QE.
- Docker Trusted Registry acts as the single source of truth.

Below is a typical developer work-flow, and is self-explanatory:



Sample Application

- To test out or understand better how the Continuous Integration and Continuous Deployment (CI/CD) is set up, we will use a sample voting application.
- All the Components of the sample application is as shown in the image below



- A Python web-app, which lets you vote between two options.
- A Redis queue, which collects new votes.
- A .NET worker, which consumes votes and stores them in...
- A Postgres database backed by a Docker volume.
- A Node.js web-app, which shows the results of the voting in real-time.

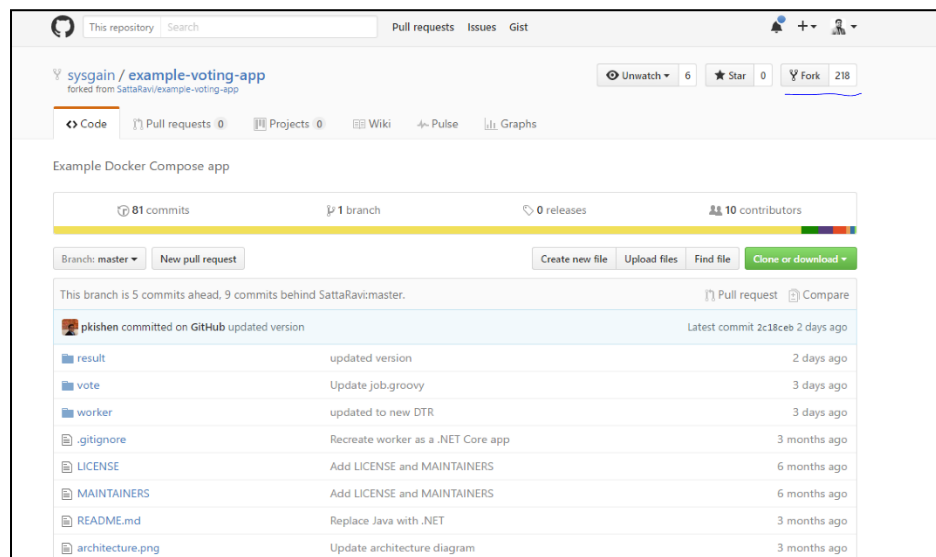
Lab Prerequisites

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.

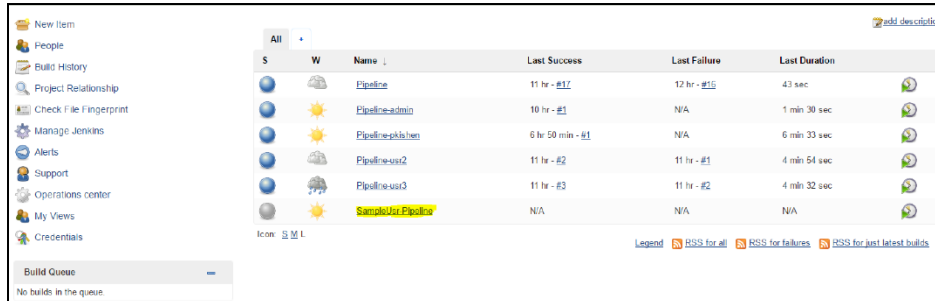
Lab Steps

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>



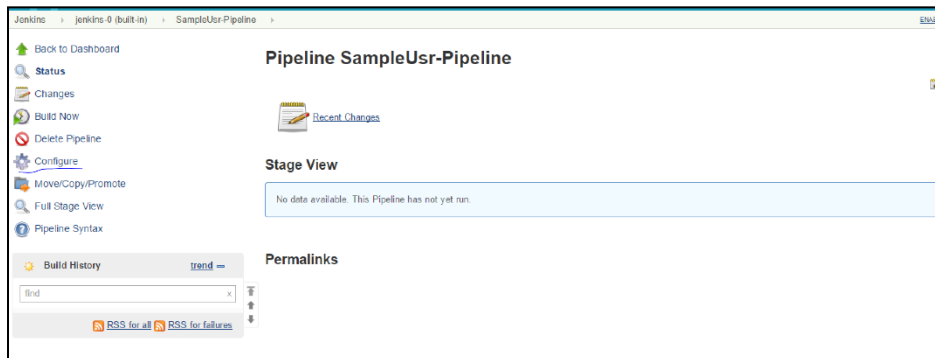
2. Once you've forked the repo, go to the Jenkins environment at <http://jenkins-0-ignite-jen.westus.cloudapp.azure.com/>.

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.

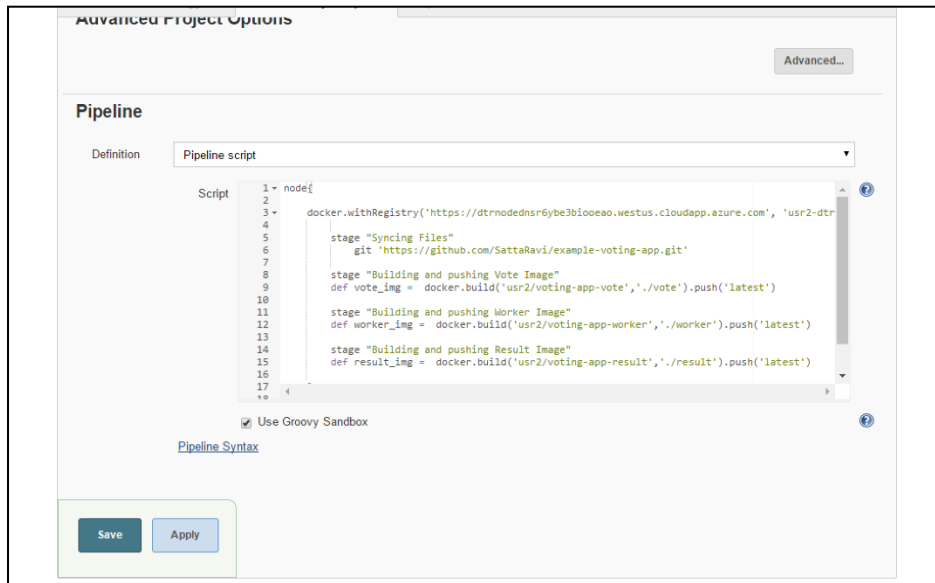


The screenshot shows the Jenkins dashboard with a list of jobs. The 'SampleUser Pipeline' job is highlighted in yellow. The left sidebar contains navigation links like 'New Item', 'People', 'Build History', etc. The bottom left shows a 'Build Queue' with 'No builds in the queue'.

S	W	Name	Last Success	Last Failure	Last Duration
		Pipeline	11 hr - #17	12 hr - #16	43 sec
		Pipeline-admin	10 hr - #1	N/A	1 min 30 sec
		Pipeline-pkhien	6 hr 50 min - #1	N/A	6 min 33 sec
		Pipeline-us2	11 hr - #2	11 hr - #1	4 min 54 sec
		Pipeline-us3	11 hr - #3	11 hr - #2	4 min 32 sec
		SampleUser Pipeline	N/A	N/A	N/A



The screenshot shows the 'Configure' page for the 'SampleUser Pipeline'. The left sidebar has links like 'Back to Dashboard', 'Status', 'Changes', 'Build Now', etc. The main content area shows 'Pipeline SampleUser-Pipeline' with a 'Recent Changes' section and a 'Stage View' section indicating 'No data available. This Pipeline has not yet run.' There are also 'Permalinks' for RSS feeds.



The screenshot shows the 'Advanced Project Options' for the 'SampleUser Pipeline'. The 'Pipeline' section is expanded, showing a 'Script' definition. The script is a Groovy pipeline script that defines stages for syncing files, building and pushing images, and building and pushing worker and result images. The 'Use Groovy Sandbox' checkbox is checked. There are 'Save' and 'Apply' buttons at the bottom.

```

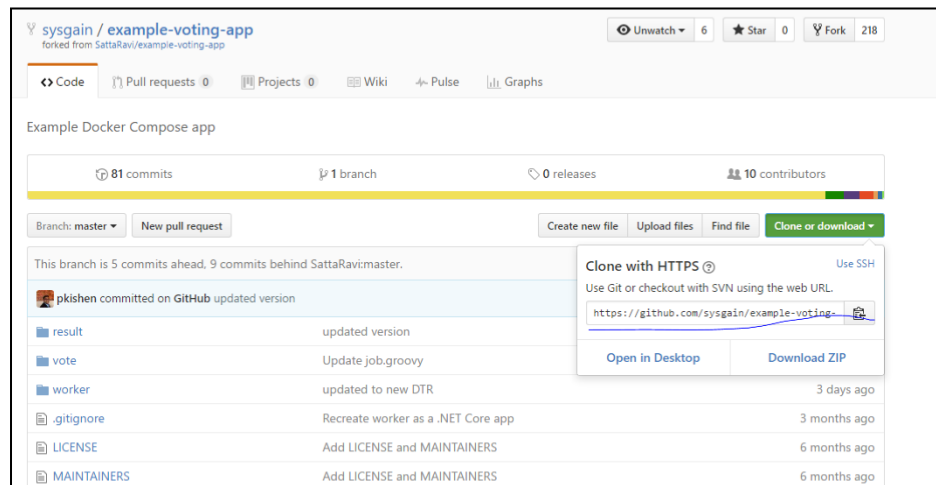
1 - node{
2
3 -   docker.withRegistry('https://dtrnodesnrf6ybe3bloeeao.westus.cloudapp.azure.com', 'usr2-dtr
4
5   stage "Syncing Files"
6     git 'https://github.com/SattaRavi/example-voting-app.git'
7
8   stage "Building and pushing Vote Image"
9     def vote_img = docker.build('usr2/voting-app-vote', './vote').push('latest')
10
11  stage "Building and pushing Worker Image"
12    def worker_img = docker.build('usr2/voting-app-worker', './worker').push('latest')
13
14  stage "Building and pushing Result Image"
15    def result_img = docker.build('usr2/voting-app-result', './result').push('latest')
16
17
18

```

3. 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.

4. Next, in the same Pipeline text box, replace the **GitHub URL** on **line 6** with your forked GitHub repository's URL from step 1.

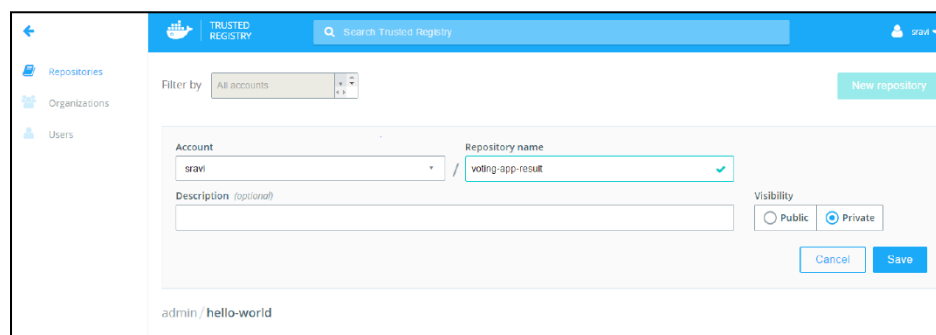


5. Next, replace the username **usr** from **lines 9, 12, and 15** with your user name.
6. Now, scroll up to the **Build Trigger** section and check the **Poll SCM** box to set up SCM polling. Then, enter five asterisks with spaces (“* * * *”) in the **Schedule** text box. This will enable Jenkins to look for changes on our GitHub repository every minute.



7. Now select **save** to save your changes and close the job configuration.
8. Login to the DTR URL in a new tab/window (<https://dtrnodednstpal2zkqvglmcs.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').

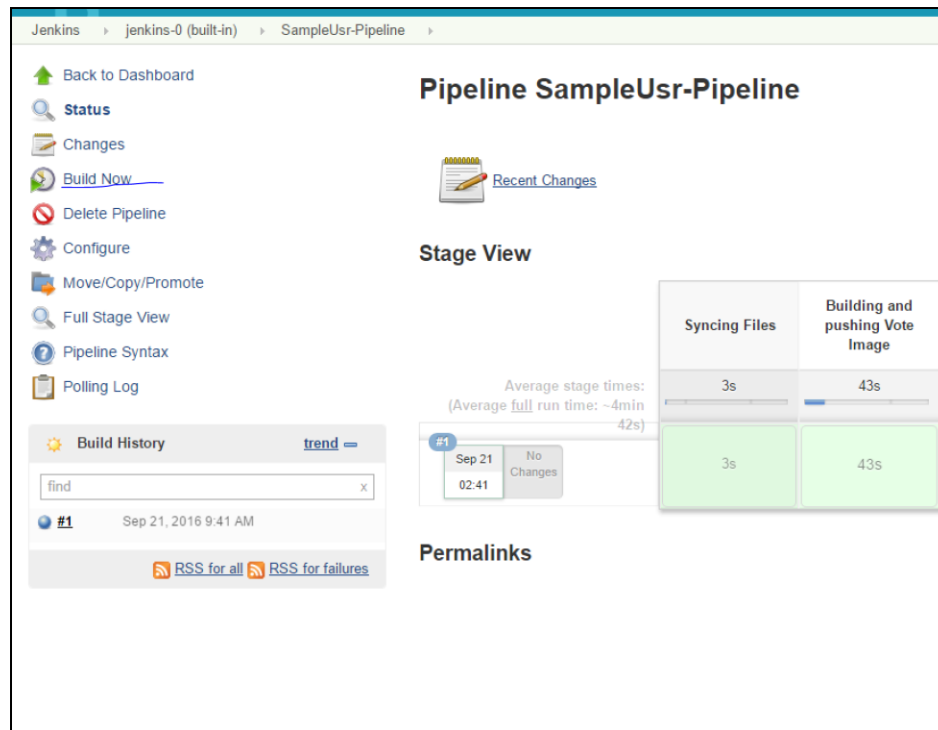


9. With this in place we can now build Docker images to be pushed to the DTR using the Jenkins job you configured earlier.

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

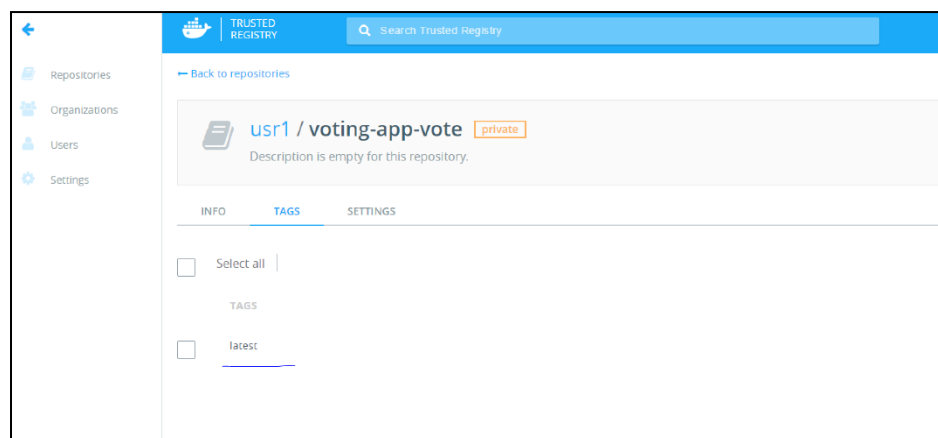
images and pushes them to the DTR.

You should be able to watch the job progress on your screen.



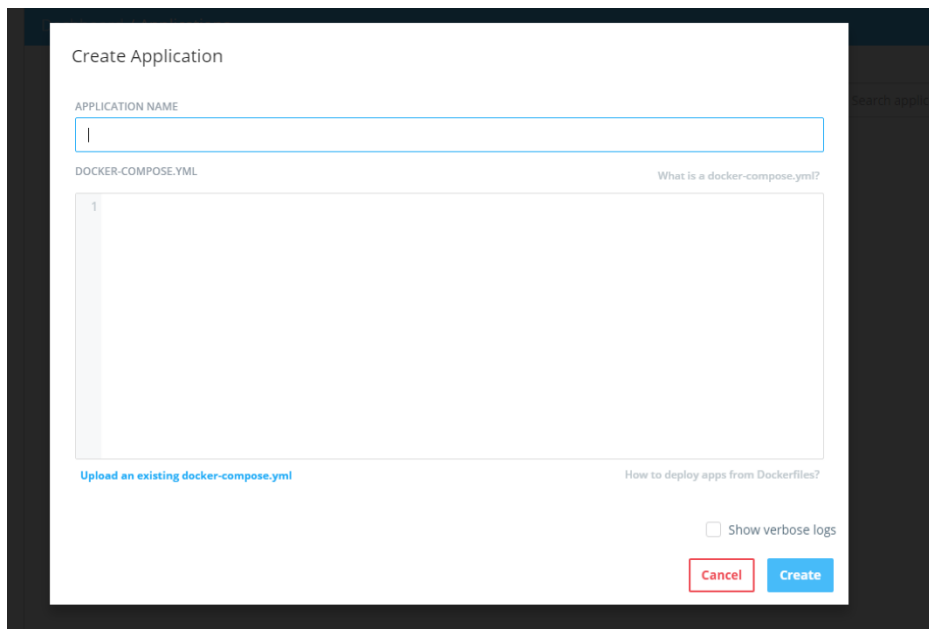
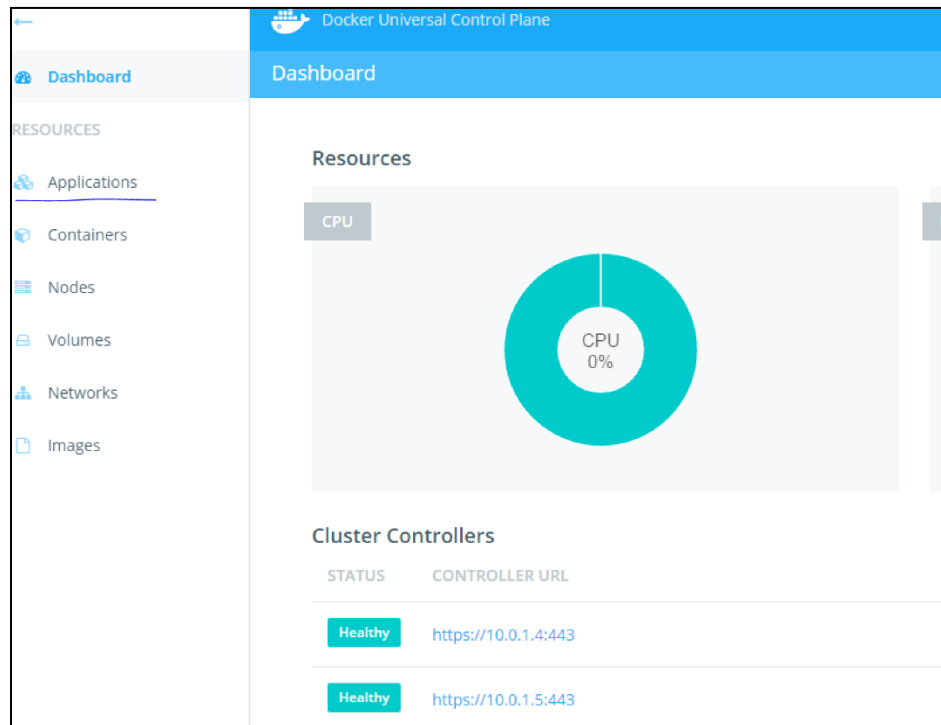
- Once the job succeeds we can find the images being pushed to the repositories we created on DTR from step 9.

Go to the DTR page, and select any of the three repositories you created previously. Select the '**Tags**' tab, and look for the latest tag on the repository. Before, this was empty.



11. With the images pushed, now you can deploy the voting application from the UCP. Login to the UCP URL:

<https://dockercontrdnstpal2zkqvglmc.westus.cloudapp.azure.com>.



The screenshot shows the 'Create Application' form. It includes a text input for 'APPLICATION NAME', a text area for 'DOCKER-COMPOSE.YML' (with a link 'What is a docker-compose.yml?'), and a checkbox for 'Show verbose logs'. At the bottom are 'Cancel' and 'Create' buttons. A link 'Upload an existing docker-compose.yml' is also present.

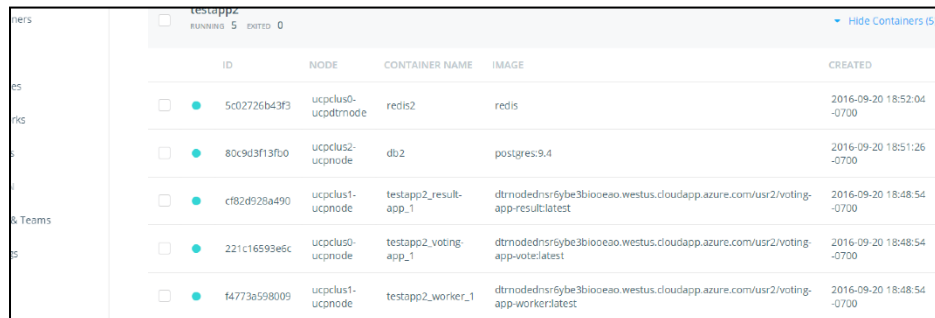
12. Select '**Applications**'. Then select the '**New Application**' button.

Paste the voting application **docker-compose.yml** from [here](#) and make the following modifications:

- Name your application with the following format: [your_username-appname].
- Replace the user **admin** on **lines 5, 14, and 23** with your username, along with the appropriate repository names on the same lines.
- Rename the interlock hostname and domain on **lines 11, 12, 20, and 21** to something different like **app.[username].com**

enable **verbose mode** with the checkbox near the bottom of the dialog screen and click '**Create**' to deploy the application.

13. Once the deployment succeeds you can see your application deployed along with the containers on the same page.



ID	NODE	CONTAINER NAME	IMAGE	CREATED
5c02726b43f3	ucpclus0-ucpdtmode	redis2	redis	2016-09-20 18:52:04 -0700
80c9d3f13fb0	ucpclus2-ucpnode	db2	postgres:9.4	2016-09-20 18:51:26 -0700
c782d928a490	ucpclus1-ucpnode	testapp2_result-app_1	dtmmodednrs6ybe3bioceo.westus.cloudapp.azure.com/usr2/voting-app-result:latest	2016-09-20 18:48:54 -0700
221c16593e5c	ucpclus0-ucpnode	testapp2_voting-app_1	dtmmodednrs6ybe3bioceo.westus.cloudapp.azure.com/usr2/voting-app-vote:latest	2016-09-20 18:48:54 -0700
14773a598009	ucpclus1-ucpnode	testapp2_worker_1	dtmmodednrs6ybe3bioceo.westus.cloudapp.azure.com/usr2/voting-app-worker:latest	2016-09-20 18:48:54 -0700

14. Now let's set up a local host to actually see the results in a browser (using the interlock domains that we provided in the docker-compose.yml).

For this application, there will be one URL to cast votes, and one URL to see the results.

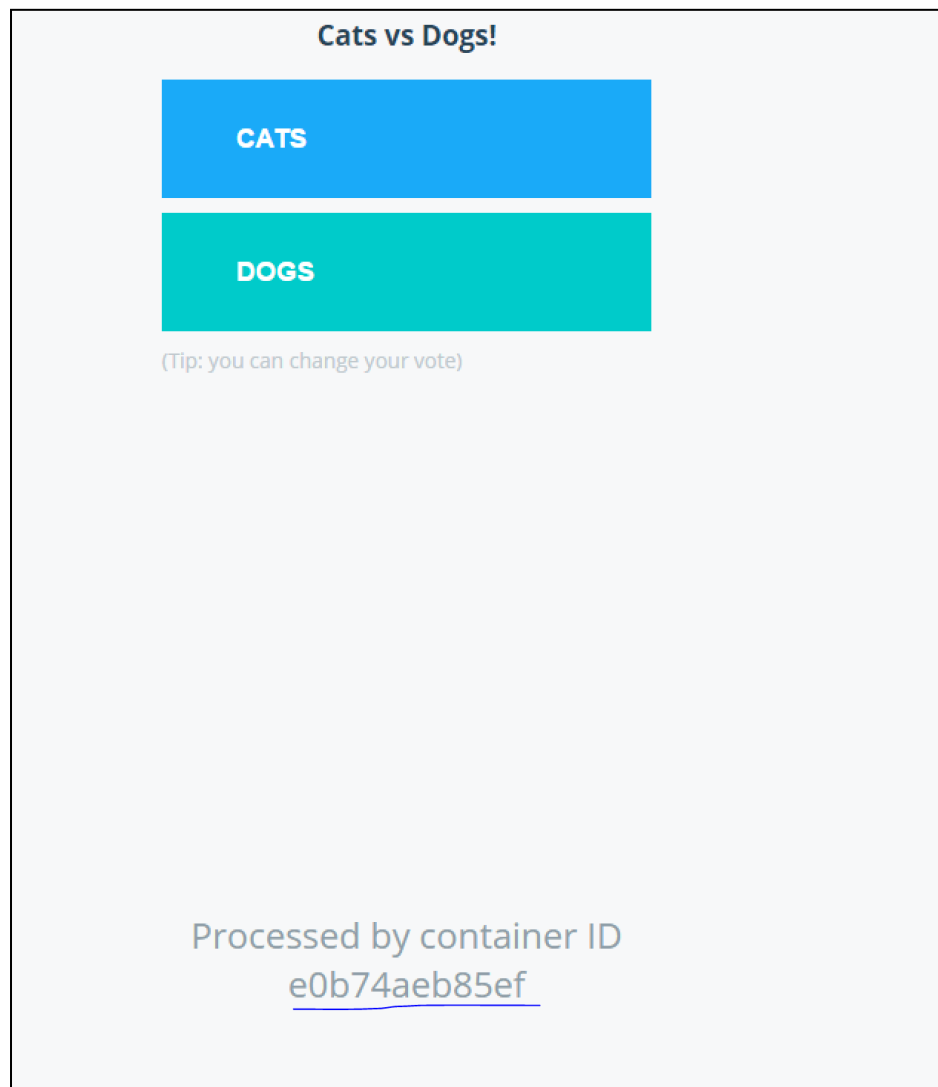
15. Follow [this](#) documentation to see how to modify your host file corresponding to your OS to add:

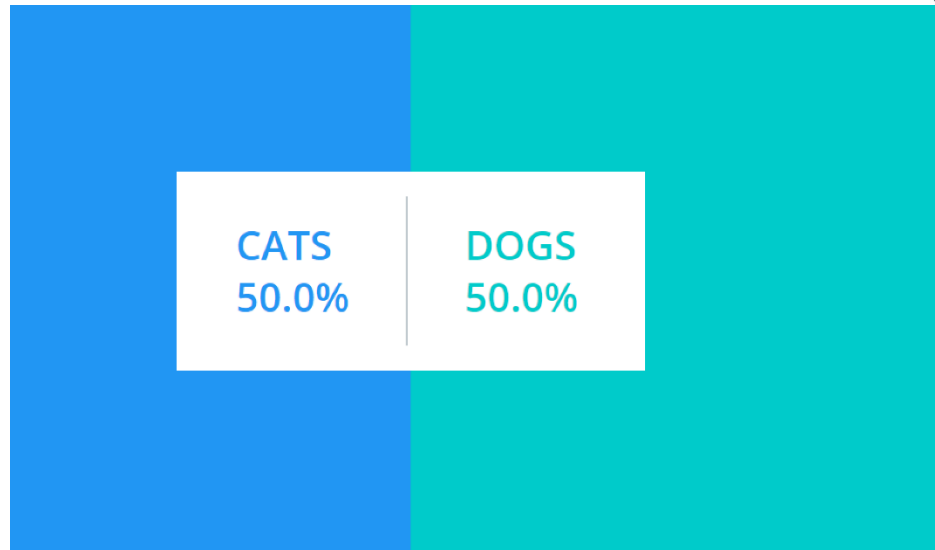
40.78.67.0 vote.[username].com

40.78.67.0 results.[username].com

then **save** and **exit** the host file.

16. Next open up these domains in a new web browser tab/window to see the deployed application in action.



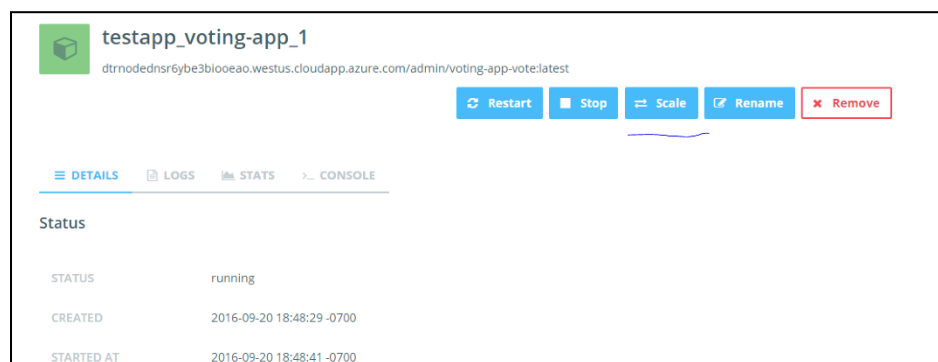


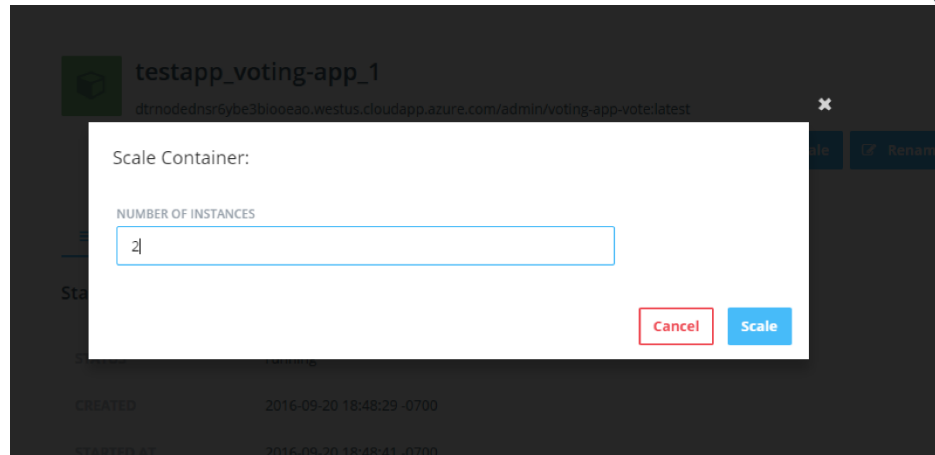
17. From the images above, and your browser, we can see the voting image runs on only one container ("Processed by container ID..." at the bottom of the page).

Now let's scale up the voting app to 2 more containers.

Return to the **Applications** list page and select the container with the voting app (do a Ctrl+F search for "voting-app").

Then, select the "Scale" button on the top left navigation:





You will be presented with a dialog, as shown above. Select a number of instances to scale up to, then select "Scale."

Note: Please do not scale up to more than 5 instances, since the lab environment is shared by all attendees. 😊

Now if you go back and try reloading the voting app in the web browser window, you should notice that the container ID cycles randomly between any of the new containers you just scaled up to.