**Deploy Java App on Container**

**Overview**

With Azure Web App for containers, it is easy to deploy container-based web apps. You can pull container images from Docker Hub or a private Azure Container Registry, and deploy the containerized app with your preferred dependencies to production in seconds.

**What's covered in this lab**

In this lab, you will learn how you can set up a Continuous Integration (CI) and Continuous Delivery (CD) pipelines with Azure Pipelines to deploy a container-based Java web application to an Azure web app for containers.

This lab will show how you can

* Create a new Azure App Service with Container and configure it to use Apache Tomcat
* Create a new MySQL database
* Use Azure App Service Task to deploy a WAR file

**Prerequisites for the lab**

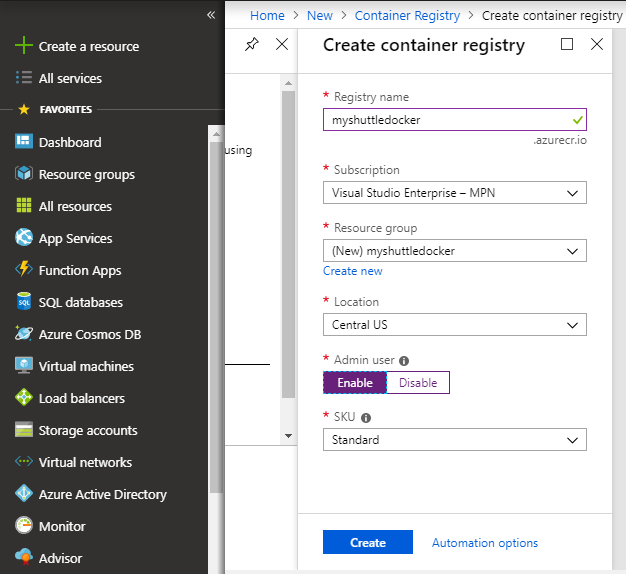
1. Use the Azure DevOps Demo Generator to provision the project to your Azure DevOps Org. Use the **MyShuttle** template.

<https://azuredevopsdemogenerator.azurewebsites.net/?TemplateId=77371&Name=MyShuttle>

**Exercise 1: Configuring a CI pipeline to build and publish Docker image**

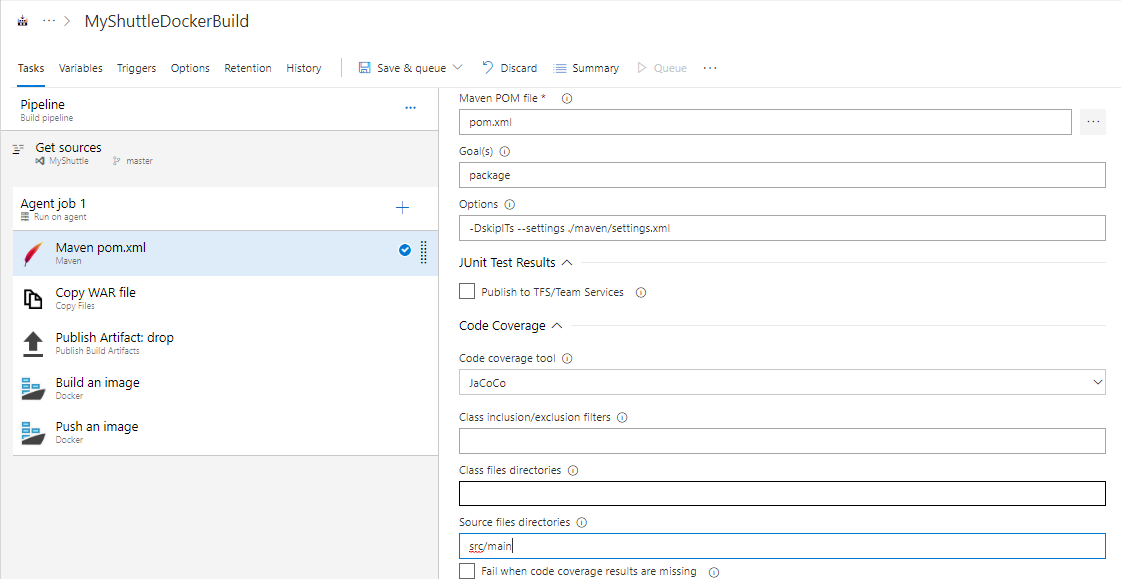
In this task, you will configure a CI pipeline that will build and push the image to Azure Container Registry.

1. Open the [Azure Portal](https://portal.azure.com/).
2. Select **+ Create a resource** and search for **Container Registry**. Select **Create**. In the *Create Container Registry* dialog, enter a name for the service, select the resource group, location, **Enable** Admin User and click **Create**.

[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/createacr.png)

1. In the Azure DevOps portal, select **Builds** from **Pipelines**. Select **MyShuttleDockerBuild** and click **Edit**.
2. Lets look at the tasks used in the build definition.
3. Select the **Maven** task. This task is used to build the pom.xml file. The maven task is updated with the following additional settings

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| Options | -DskipITs --settings ./maven/settings.xml | Skips integration tests during the build |
| Code Coverage Tool | JaCoCo | Selects JaCoCo as the coverage tool |
| Source Files Directory | src/main | Sets the source files directory for JaCoCo |

1. [](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/vsts-mavensettings.png)
2. Then there is **Copy** task. We will copy the WAR file from the sources directory to the staging folder.

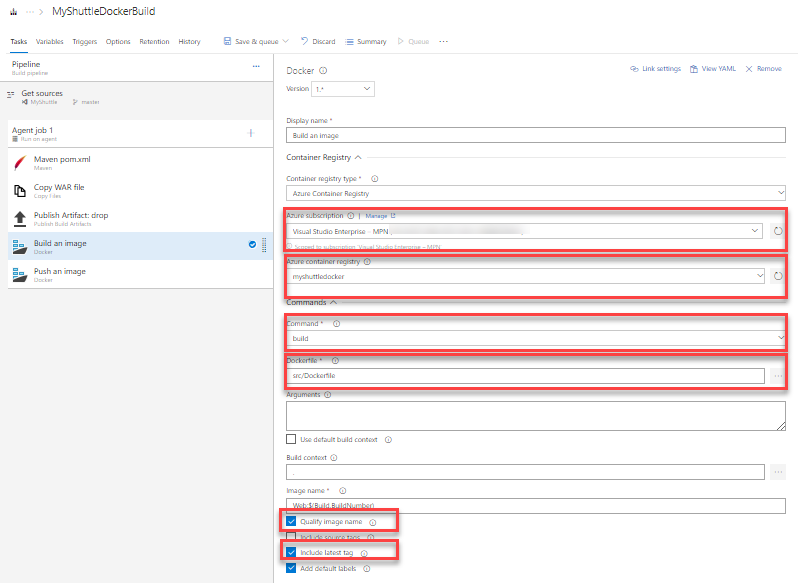
| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| Source Folder | $(build.sourcesdirectory) | Copy from the source folder |
| Contents | target/myshuttledev\*.war, \*.sql | Copy the MyShuttle WAR file |
| Target Folder | $(build.artifactstagingdirectory) | Copy it to the default staging folder |

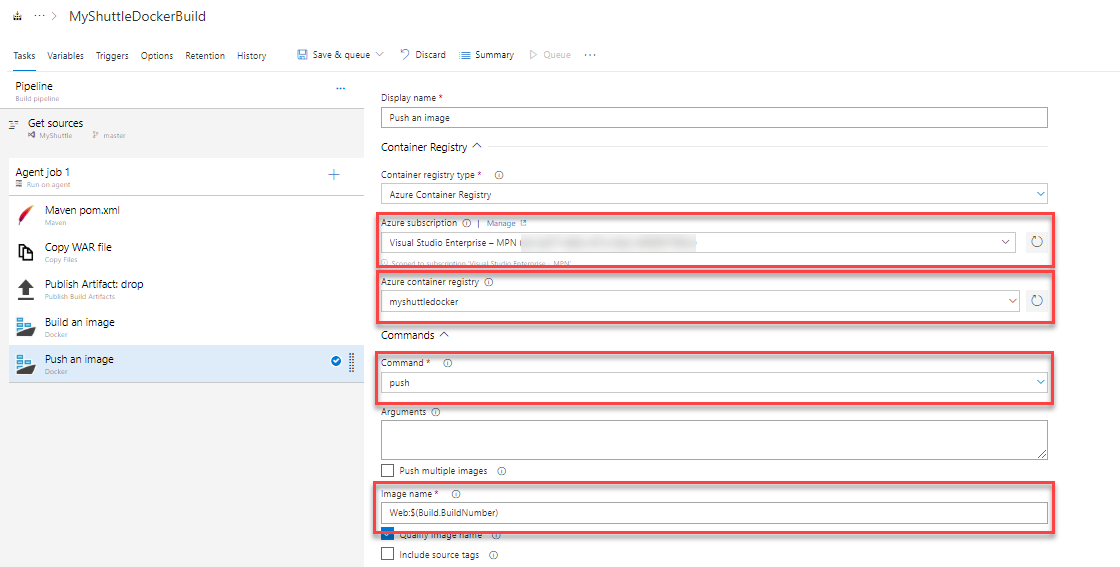
1. Next, we have a **Publish** task to publish the build artifacts to Azure Pipelines.

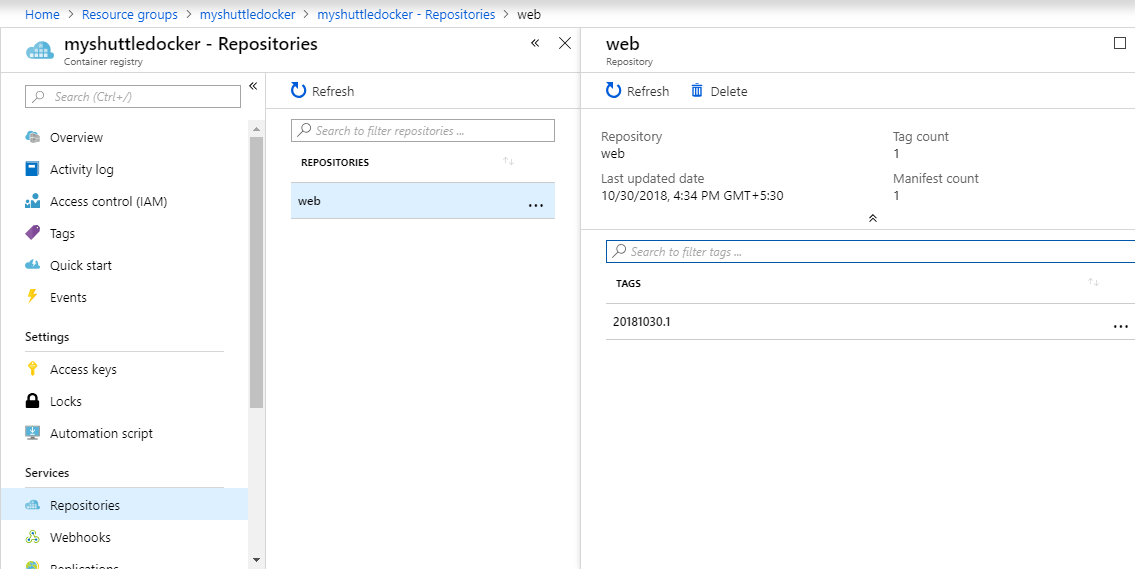
| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| Path to publish | $(build.artifactstagingdirectory) | Copy contents from the staging folder |
| Artifact name | drop | Provide a name for the artifact folder. |
| Artifact publish location | Visual Studio Team Services/TFS | we will publish it to Azure pipelines |

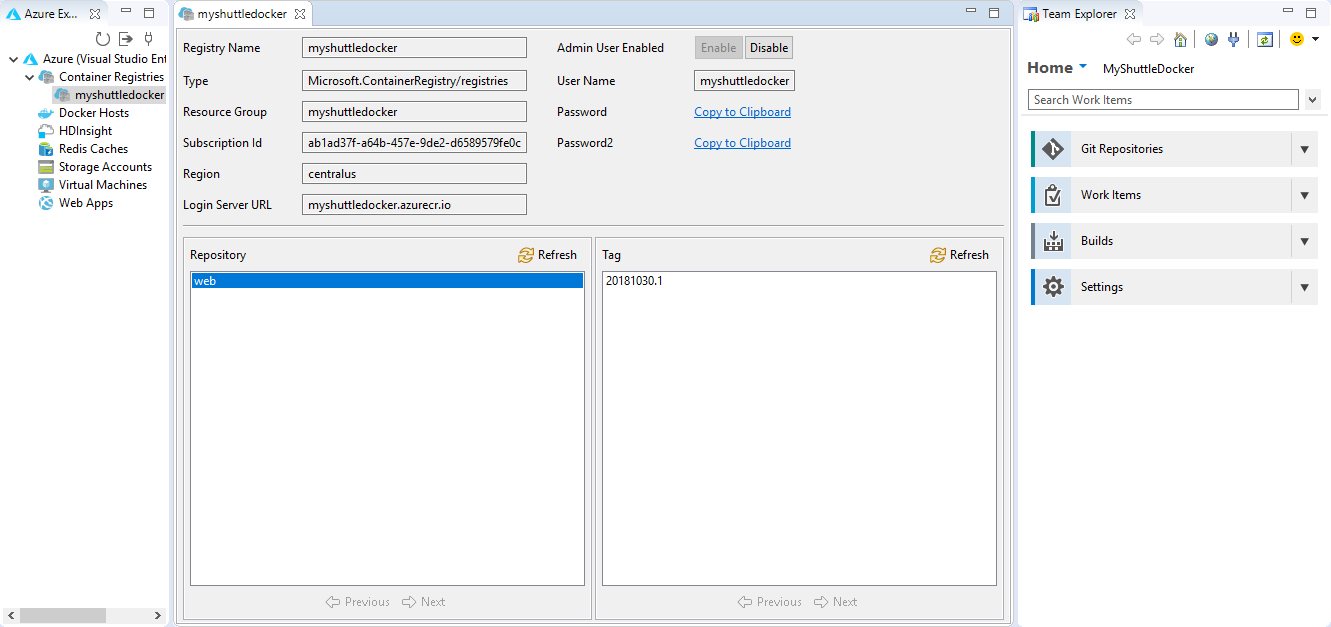
1. Next, there are two **Docker** tasks to build and publish the images. Select the first **Docker** task and notice that the **Command** is set to **Build**. The other settings of the Docker compose tasks are as follows:

| **Parameter** | **Value** | **Notes** |
| --- | --- | --- |
| Azure Subscription | Authorize your Azure subscription | The subscription that contains your registry |
| Container Registry Type | Azure Container Registry | This is to connect to the Azure Container Registry you created earlier |
| Azure Container Registry | Your registry | Select the Azure Container registry you created earlier |
| Command | build | Docker command |
| Dockerfile | src/Dockerfile | Point to the src folder for the docker file |
| Use default build context | Uncheck this option |  |
| Build context | . (dot representing the root folder) | The build context should be the root folder |
| Image name | Web:$(Build.BuildNumber) | Sets a unique name for each instance of the build |
| Qualify image name | Check (set to true) |  |
| Include Latest Tag | Check (set to true) | Adds the latest tag to the images produced by this build |

1. [](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/dockerbuildtask.png)
2. There is a second **Docker** task with almost the same settings. The only change is the **Command** is set to **Push** and the **Image name** is set to Web:$(Build.BuildNumber). This action will instruct the task to push the Web image to the container registry.

[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/dockerpublishtask.png)

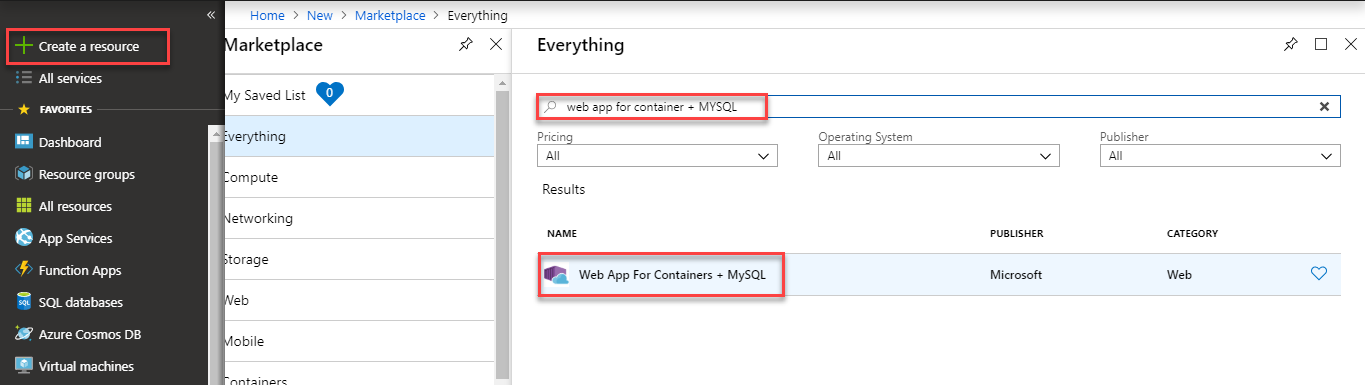
1. Click the **Save and Queue** button to save and queue this build. Make sure you are using the **Hosted Ubuntu 1604** build agent.
2. Wait for the build to complete. When it is successful, you can go to your Azure portal and verify if the images were pushed successfully. [](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/portal-acrrepo.png)
3. If you are following this from the Eclipse lab, you can also verify if the images were pushed correctly from the **Azure Explorer** view. *Sign in* to Azure, refresh Azure Container Registry. Right click and select **Explore Container Registry**. You should see the image - tagged with the build number.

[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/exploreacr.png)

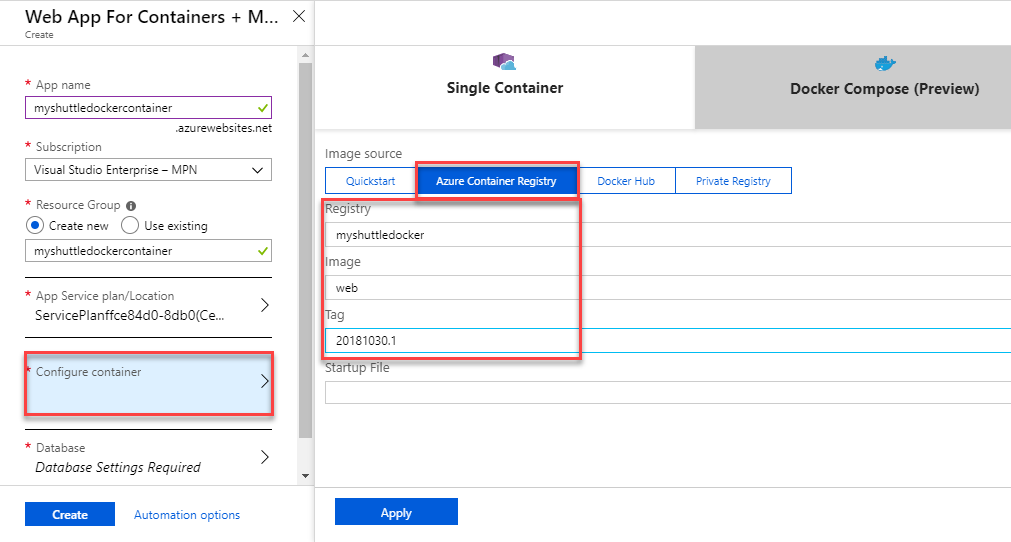
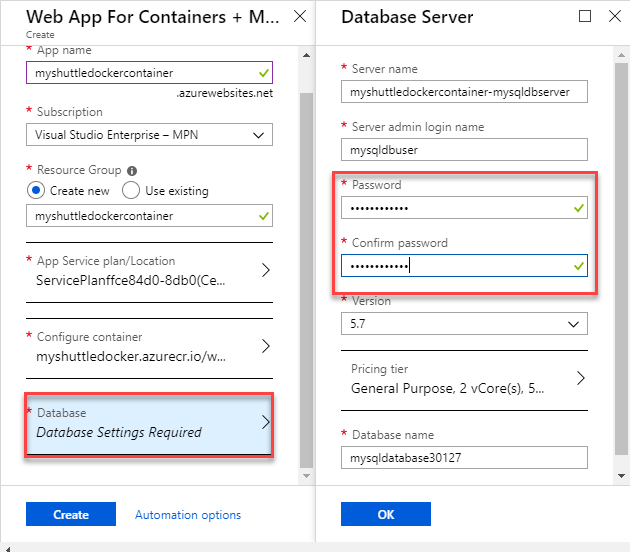
**Exercise 2: Deploying to an Azure Web App for containers**

In this exercise, we will setup a Release pipeline to deploy the web application to an Azure web app. First,let's create a Web App for Container with MYSQL.

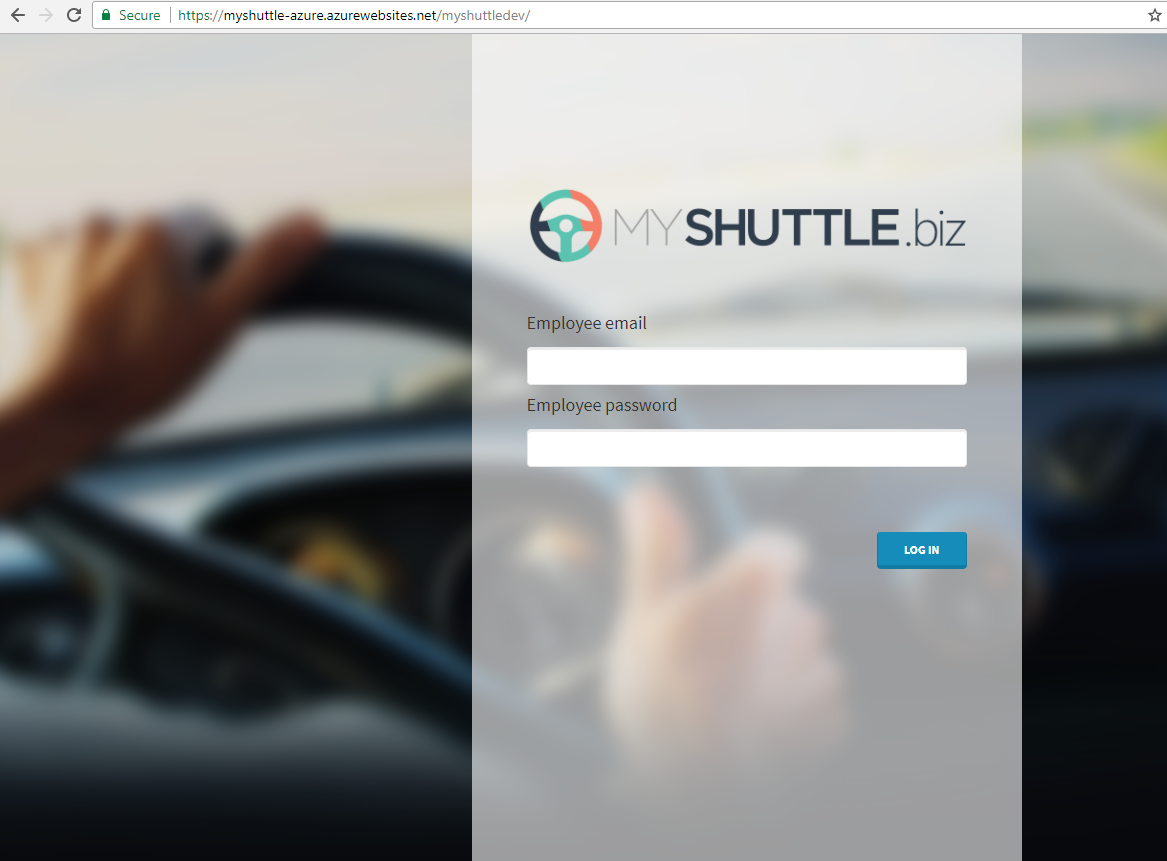
1. Sign into your [Azure Portal](https://portal.azure.com/)
2. In the Azure Portal, choose **+ Create a resource**, search for **Web App for Containers + MYSQL**, select and click *Create*.

[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/newwebapp.png)

1. Provide the following details and click **Create**-
   * Enter a name for the new web app
   * Choose the Azure subscription
   * Select existing or create new resource group for the web app.
   * Leave the App Service plan/Location as it is.
   * In the *Configure Container* option, select **Azure Container Registry**. Select the **Registry, Image and Tag** from the respective drop-downs and click **Apply**.
   * In the *Database* section, provide all the required mandatory information and note down **Server Name, Server admin login name, Password** to a notepad. We will use it later in the Deployment pipeline.

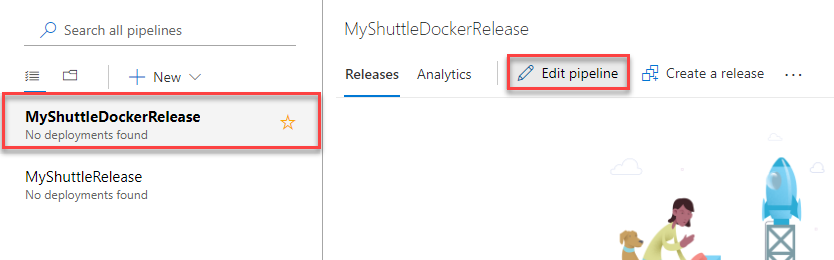
[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/myshuttle-webapp2.png)[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/mysql-webapp.png)

1. Once the provisioning is complete, go to the web app Overview page, and select the URL to browse the web app. You should see the default **Tomcat** page.
2. Append **/myshuttledev** to the web application context path in the URL to get to the MyShuttle login page. For example if your web app URL is https://myshuttle-azure.azurewebsites.net/ , then your URL to the login page is https://myshuttle-azure.azurewebsites.net/myshuttledev/

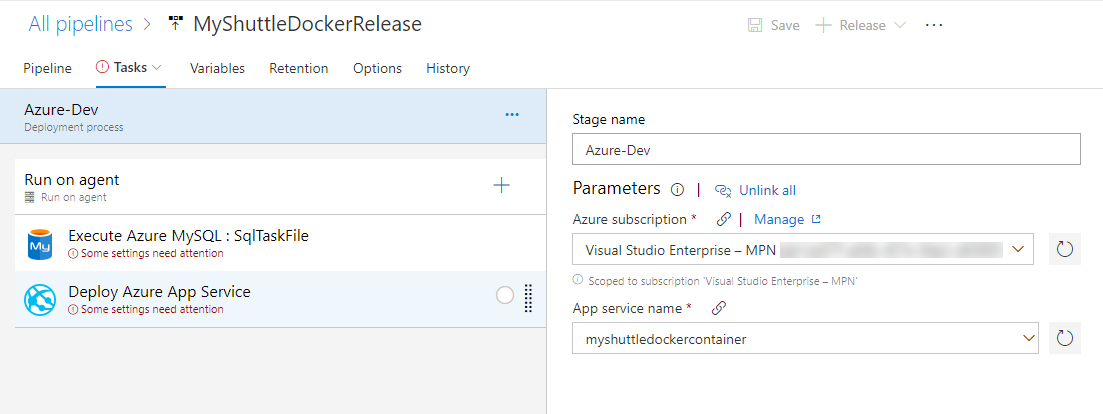
[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/loginpage.png)

We could configure *Continuous Deployment* to deploy the web app when a new image is pushed to the registry, within the Azure portal itself. However, setting up an Azure Pipeline will provide more flexibility and additional controls (approvals, release gates, etc.) for application deployment.

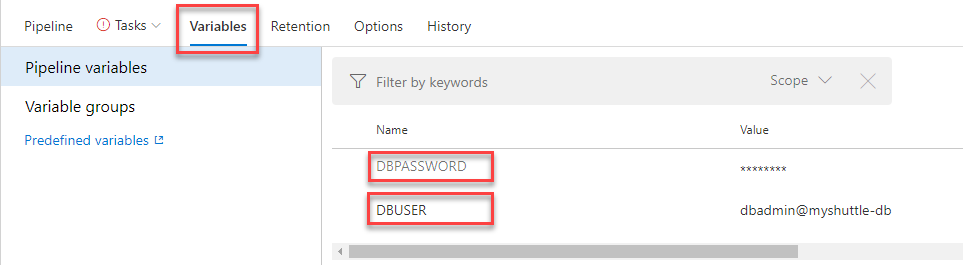
1. Back in Azure DevOps account, select **Releases** from the **Pipelines** hub. Select the Release definition - **MyShuttleDockerRelease** and click *Edit Pipeline*.

[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/editrelease.png)

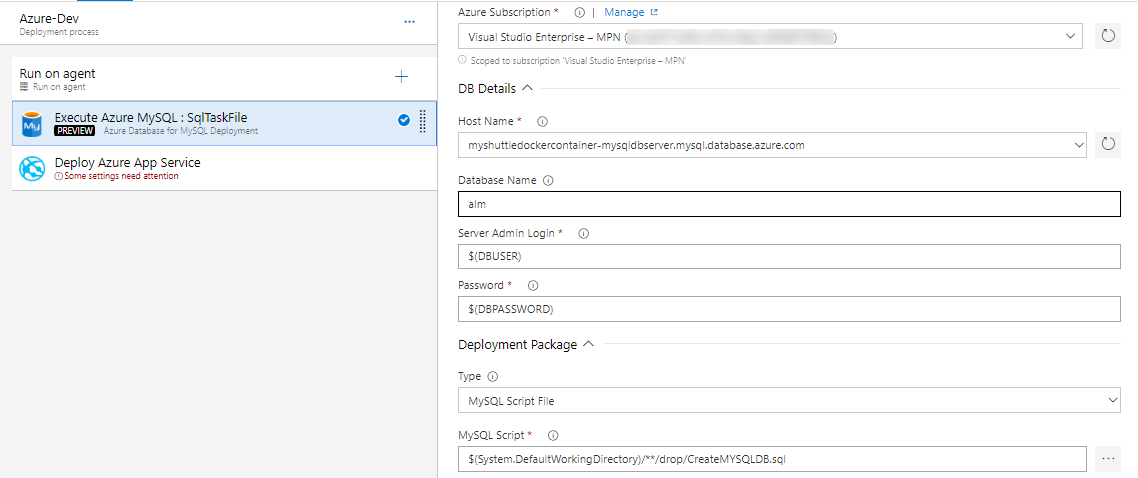
1. Hover the mouse on **Tasks** and select **Azure-Dev**. Configure the environment as below -
   * Pick the Azure subscription
   * Enter the **App Service** that you created

[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/vsts-cd-webapp.png)

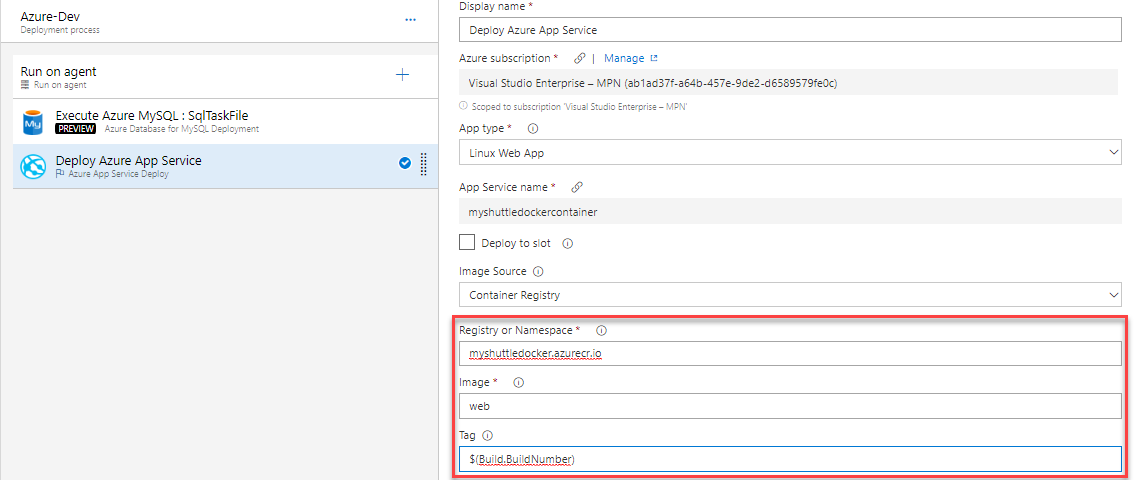
1. Select the **Execute Azure MYSQL:SqlTaskFile** task, choose the Azure subscription, and provide the DB details which were noted down earlier during the creation of the database server.
   * Select the *Host Name* from the drop down. You can find this value in the **Properties** page of the created MYSQL database in Azure portal.
   * *Server Admin Login* - You can find this value in the **Properties** page of the created MYSQL database in Azure portal. Go to **Variables** section and enter the value for the variable - *$(DBUSER)*.
   * Enter the *Password*. This is the password provided during the creation of MYSQL database in Azure portal. Go to **Variables** section and enter the value for the variable - *$(DBPASSWORD)*. Click the **lock** icon to decrypt the dummy value and then, enter the password.

[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/variables.png)

* + A *MYSQL script* that is version controlled and provided here which creates the database, tables and populate records.

[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/mysqlcreatetask.png)

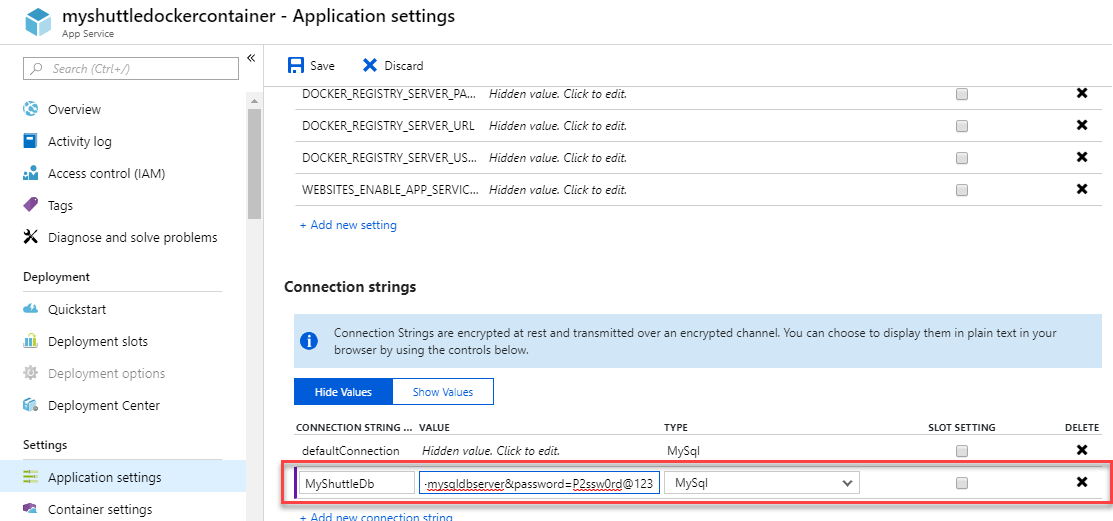
1. Select the Deploy Azure App Service task and make sure that the following values are provided. Note that the task allows you to specify the Tag that you want to pull. This will allow you to achieve end-to-end traceability from code to deployment by using a build-specific tag for each deployment. For example, with the Docker build tasks you can tag your images with the Build.Number for each deployment.
   * Registry or Namespace - Provide the value of Login server of the created Container Registry. You will find it in the Overview section.
   * Image - Provide the value as web. This is where the container image is stored after build.
   * Tag - Provide the value as $(Build.BuildNumber).

[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/vsts-buildtag.png)

1. Select **Save** and then click **+ Release** | **Create Release**.
2. Check the artifact version you want to use and then select **Create**.
3. Wait for the release is complete and then navigate to the URL http://{your web app name}.azurewebsites.net/myshuttledev. You should be able to see the login page.

**Exercise 3: Configuring MySQL connection strings in the Web App**

1. Navigate to the Web app that you have created. Click **Application Settings** and scroll down to the **Connection Strings**section
2. Add a new MySQL connection string with **MyShuttleDb** as the name and the following string - jdbc:mysql://**{MySQL Server Name}**.mysql.database.azure.com:3306/alm?useSSL=true&requireSSL=false&autoReconnect=true&user=**{your user name}**@**{MySQL Server Name}**&password=**{your password}**.
3. Replace the following with values that you have noted down
   * MYSQL Server Name - **Server Name** in the MYSQL Server *Properties* page
   * Your user name - **SERVER ADMIN LOGIN NAME** in the MYSQL Server *Properties* page
   * Your password -**Password** that you provided during the creation of MYSQL server in Azure

[](https://github.com/microsoft/azuredevopslabs/blob/master/labs/vstsextend/dockerjava/images/mysqldbconn.png)

1. Click **Save** to save the connection string.
2. You should be able to login to the application now. Return to the web application and try logging in using any of the below *username/password* combination:
   * *fred/fredpassword*
   * *wilma/wilmapassword*
   * *betty/bettypassword*