**DWR CI creation Steps**

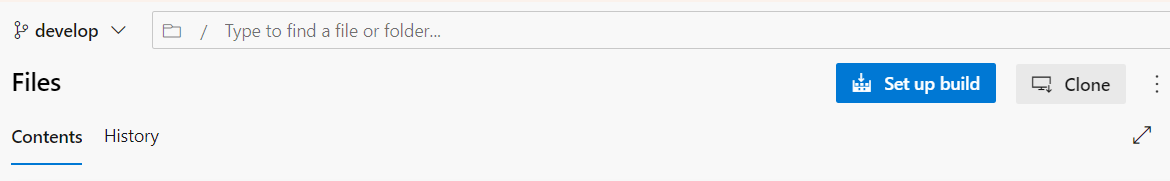
**Existing process:**

1. User login to bte-uapp-09 server with putty with his user credentials and then switch as a was1 user
2. Then navigate to the script path “cd Build/HudsonBuild/DWR\_WAS8\_DEV\_BUILD/TFS\_BUILD”.
3. Then make change in script with the branch name required for the Build and run the script.
4. The script will copy the source code from TFS and paste it in “/data/Build/DEV\_BUILD/” location.
5. Then run the build.xml file as ant build.
6. Finally, it creates a **tosdwr.ear** in dist location.

**Steps for CI creation:**

1. **Creation of azure-pipelines.yml file.**

* Create a new branch or setup build for existing branch by clicking on the button.

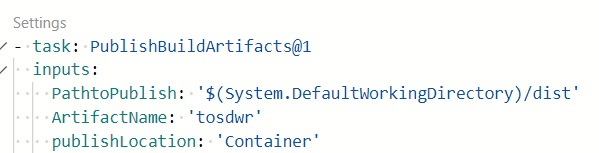


* Now start with a starter pipeline and keep **adding relevant task** from the scheduler assistant.
* Assign your **branch name in trigger** and whenever a developer commits to the branch the **pipeline automatically runs**.
* For DWR we require 2 task for now **task1: Ant@1** and **task2: PublishBuildArtifacts@1**

1. **Ant task** is required to execute the build.xml present in the codebase. We need to assign some java configuration **and the target** (which is the entry target of our build.xml).



1. **PublishBuildArtifacts task** is required to publish our artifact (final ear of our application) to Azure container. We need to assign **the name and location of our artifact** which is created by ant task.

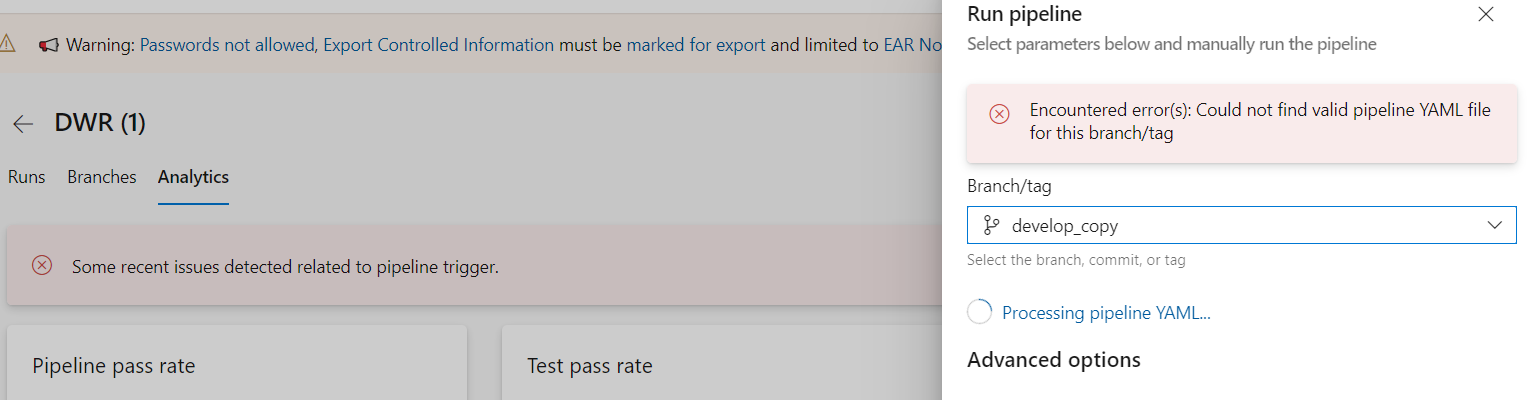


1. Once yaml file is completed then click on save which will commit the new file in the same branch and create a pipeline for the repository.

**Running the pipeline**

Manual running the pipeline

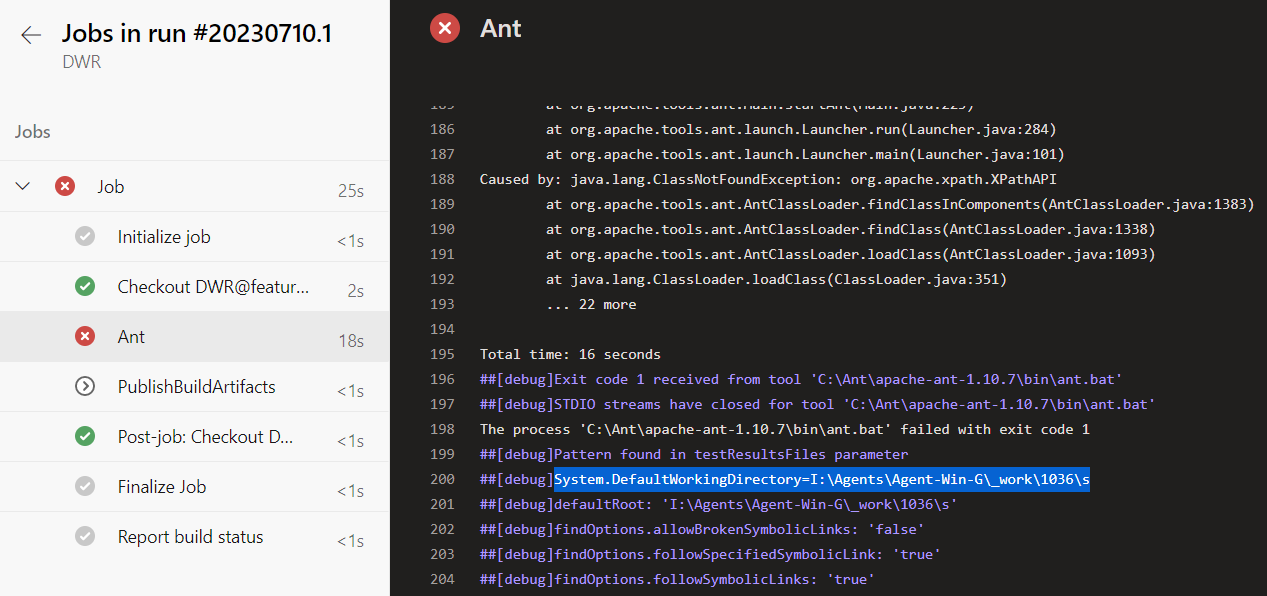
1. Once pipeline is created for a project. Go to the pipeline section in ADO, select your project and now it will ask you the select the branch.
2. If you are trying to run pipeline for a branch which **doesn’t have YAML file** then it will give error message.



1. Once everything is ready then click on the run button to run the pipeline.

**ANALYSING THE PIPELINE AFTER RUN**

**Different jobs are associated with the pipeline**

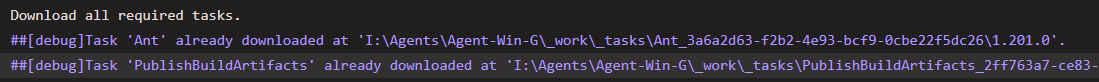


**Initialize Jobs**

1. For every pipeline, TFS will create a local registry in which it checkout the code and build it.
2. First it deletes the existing storage location and artifacts for the pipeline which is

I:\Agents\Agent-Win-G\\_work\1036\a

1. It then downloads the tasks assign for the pipeline

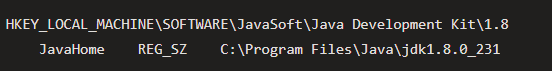


**Checkout job**

1. First task it does is creates a new storage location for the code which is ‘s’ earlier it was ‘a’ I:\Agents\Agent-Win-G\\_work\1036\s
2. Then it checks out the branch code to this location 

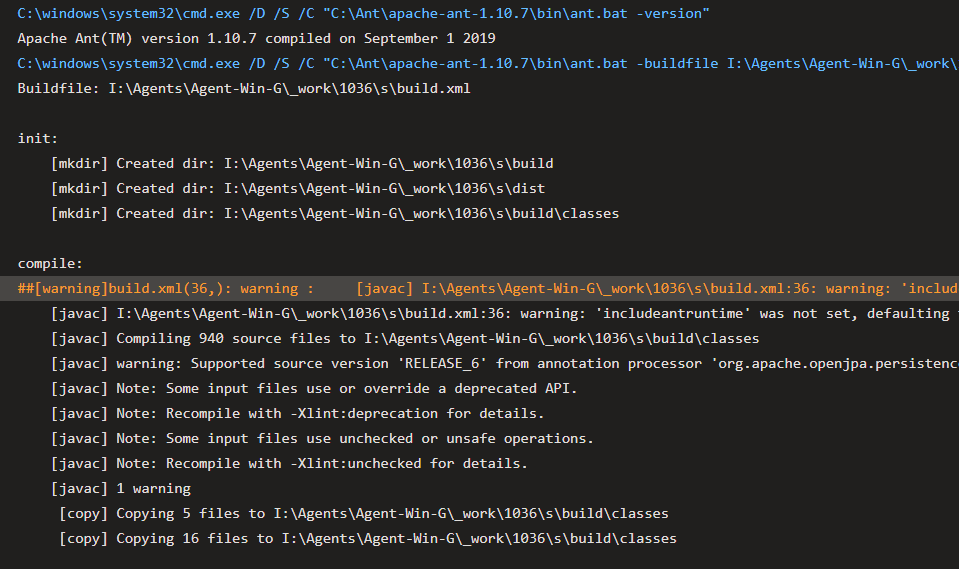
**Ant job**

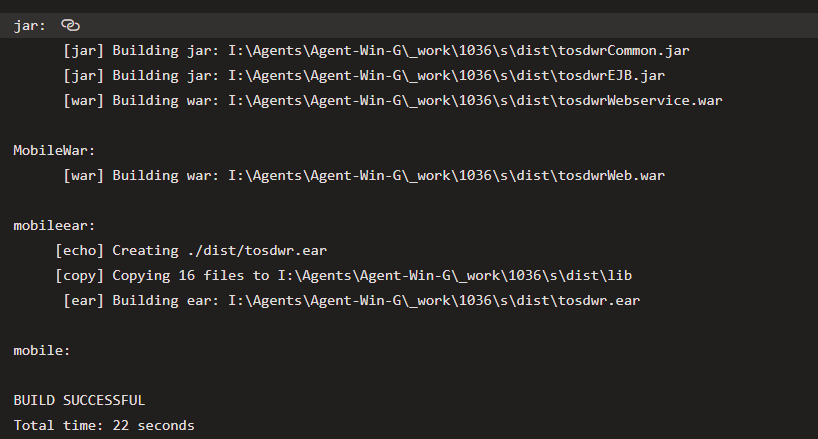
1. Assign java home path



1. Start executing build.xml file: It starts executing tasks one by one

**Init target:**



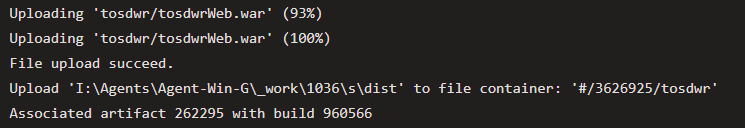


1. Finally it creates the artifact at location

I:\Agents\Agent-Win-G\\_work\1036\s\dist\tosdwr.ear

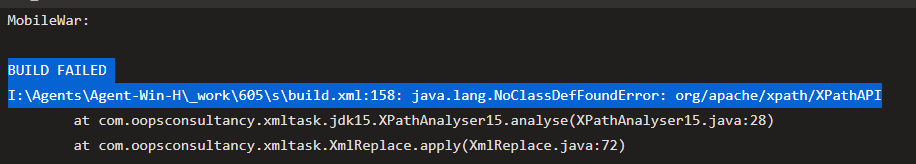
**PublishBuildArtifact Job**

1. It publishes/upload the artifact created at dist location to container.



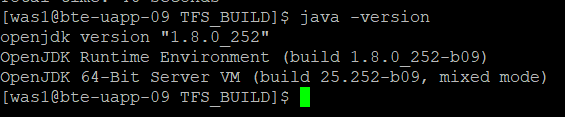
**ISSUE with DWR build.xml in azure pipeline**

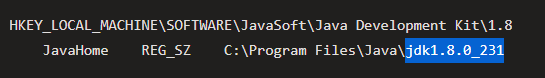
**Inside Mobilewar**



But the same script in working fine in app-09 DWR server.

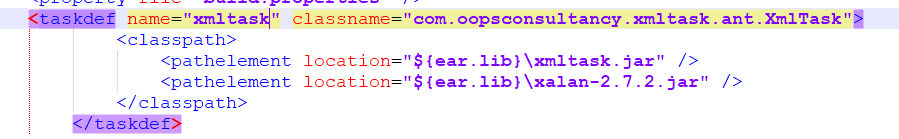
**Possible difference between Azure and DWR server:**

1. **Java version in server-** 
2. **Java version in azure**



**SOLUTION**

**ADDING xalan to the classpath in build.xml resolved the issue**



**WHY WE NEED A taskdef**

1. The **taskdef** task is typically used when you have custom Ant tasks defined in a separate JAR file or classpath and you want to make those tasks available for use in your build script.
2. **Here xmltask is our customtask which define XmlTask present in the classpath elements.**
3. **Now this XmlTask class is used to below task in mobilewar**



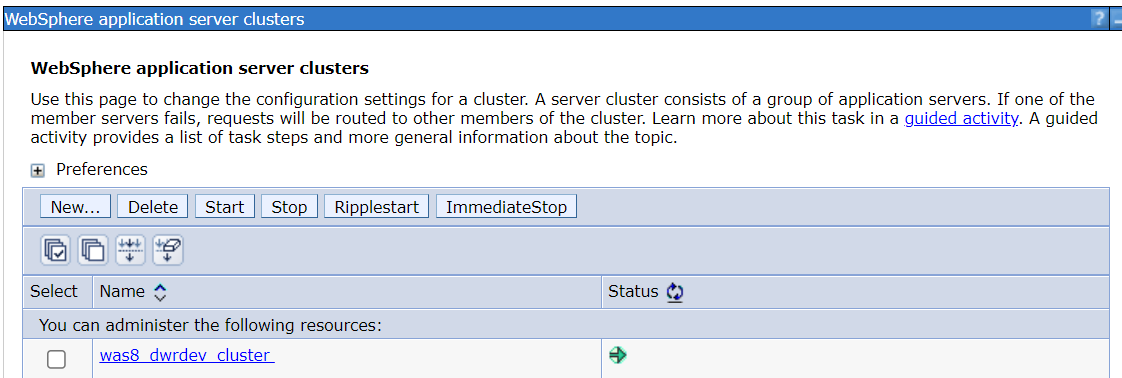
**CD pipeline**

**Existing process**

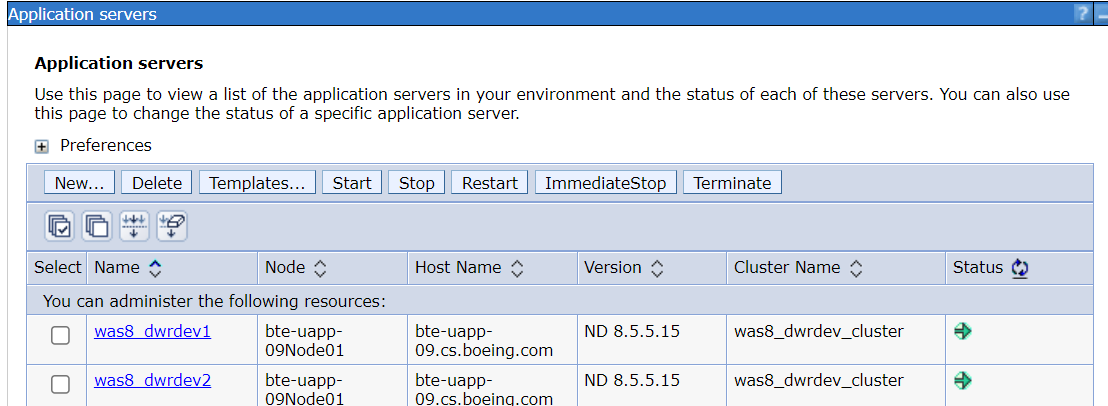
Once artifact **tosdwr.ear** is created at DWR server location. We do manual deployment by login to dev instance of IBM websphere server.

**IBM websphere application server process**:

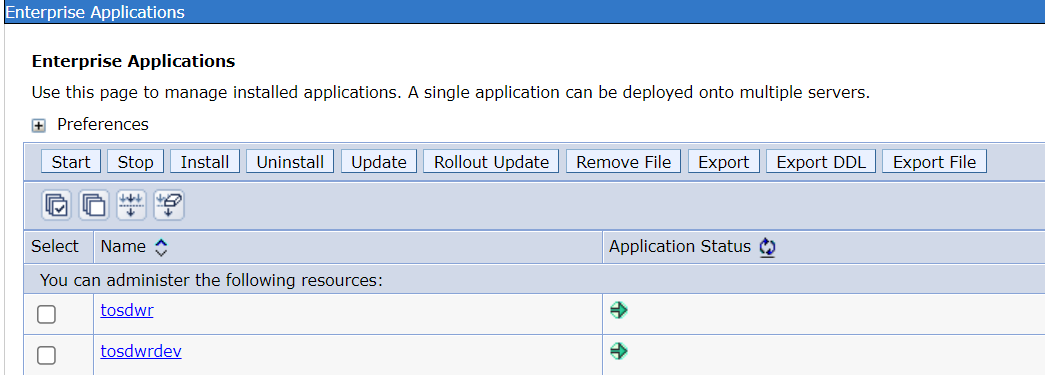
1. We have a **cluster** created for every env which contains multiple servers on which dev instance is running on.



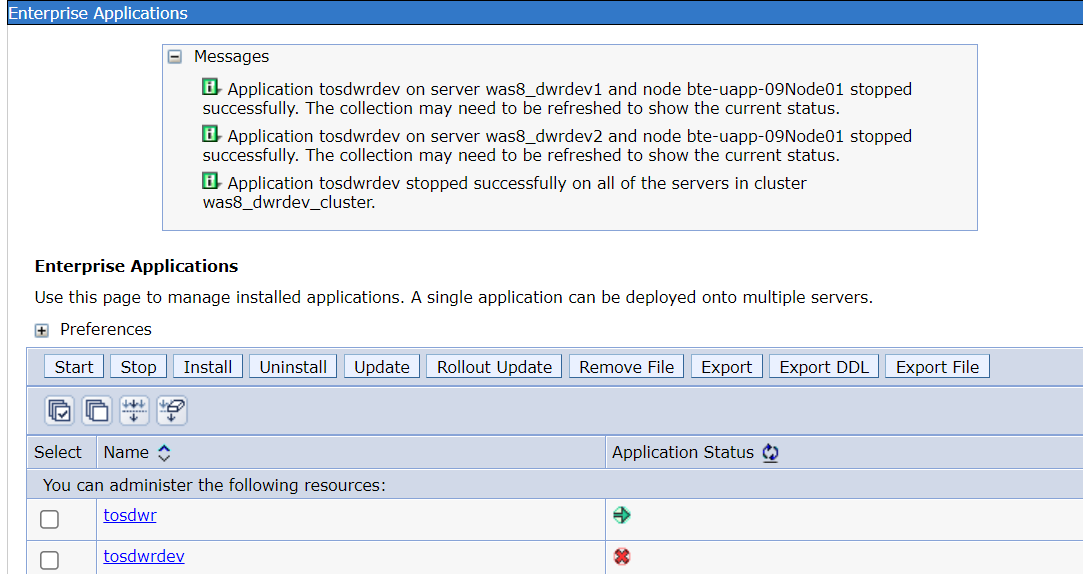
1. After cluster we have 2 **application servers** dedicated for each application instance so if one server is down it will route to another.



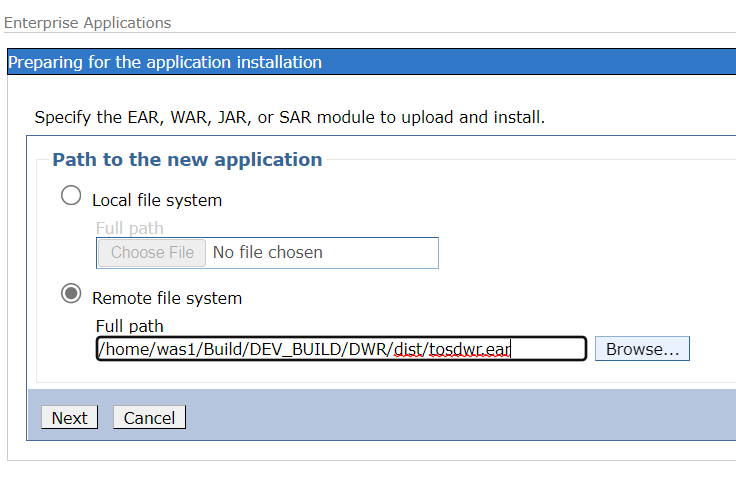
1. For each instance we have an application which is deployed on multiple application servers.



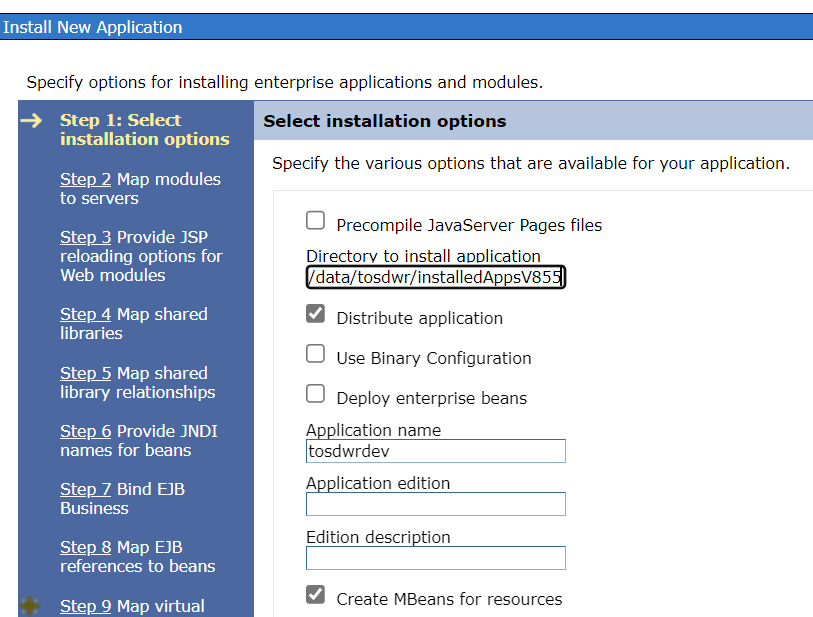
1. We need to stop this instance of application and then uninstall the existing instance in order to redeploy changes.



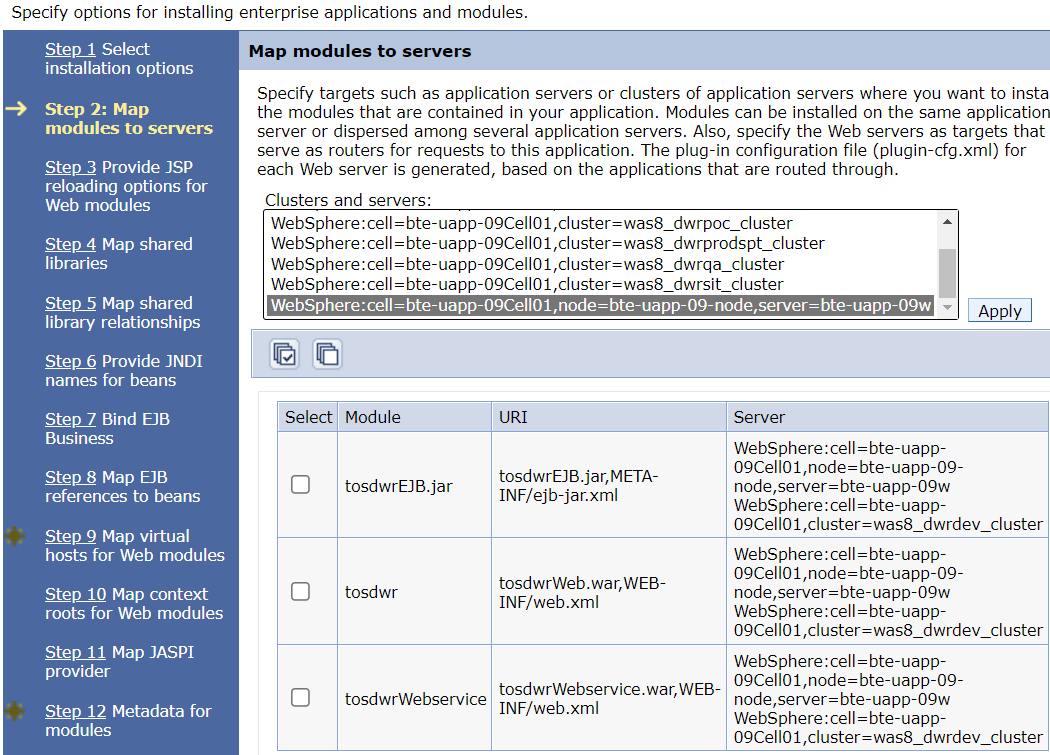
1. Then click on install and choose the artifact location.



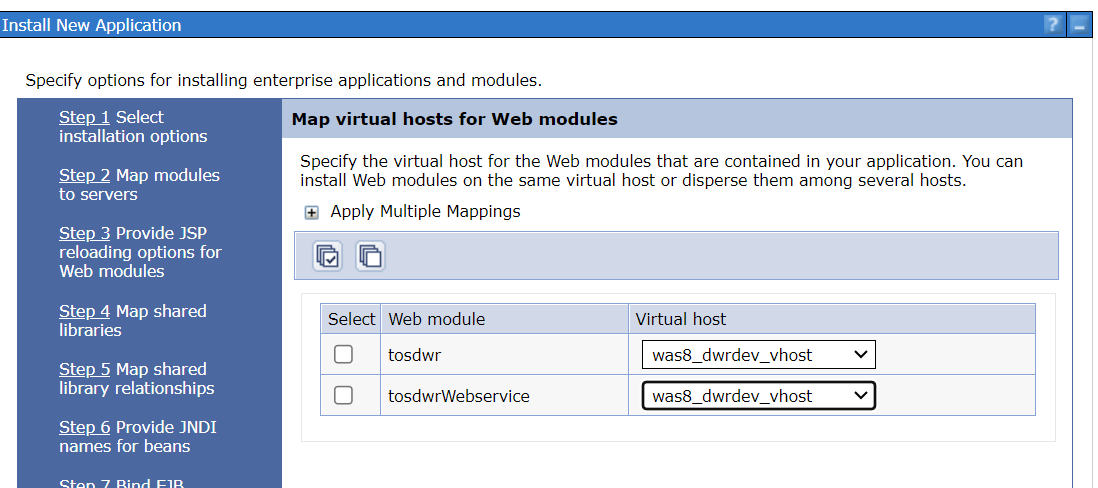
1. Select the directory in which application is deployed as of now we choose the DWR server location as directory.



1. Mapping module to server



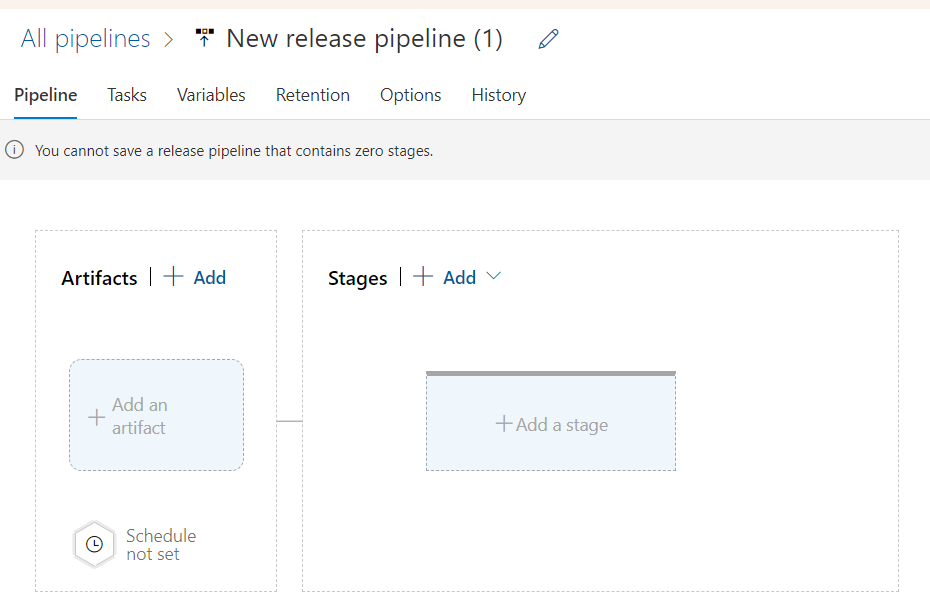
1. Map virtual host to .war projects



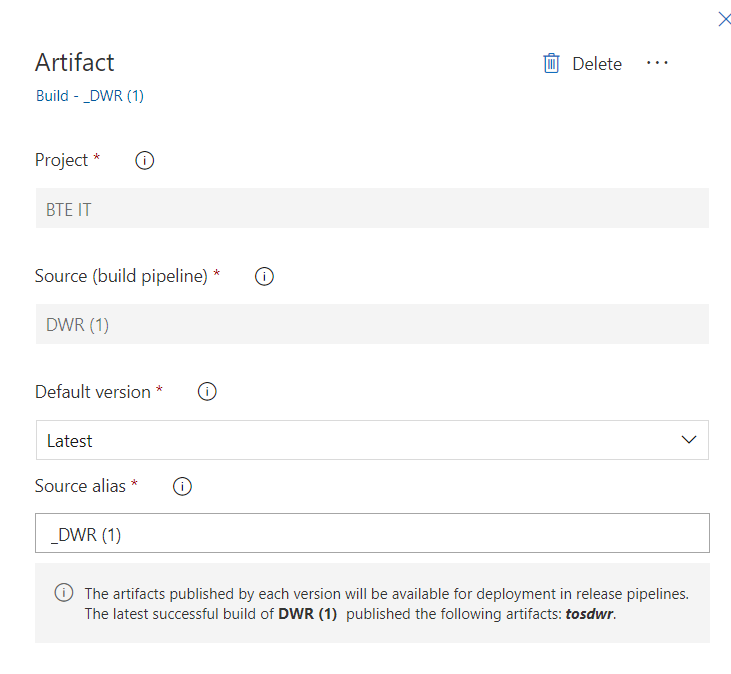
1. Save to master configuration and save it.

**AZURE CD pipeline setup**

1. Go to Release in azure and create a new release pipeline
2. **Every release pipeline has 2 primary configurations:**
3. Artifact and Stages

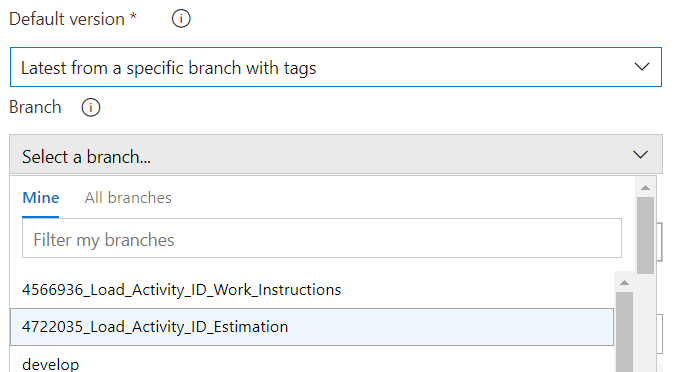


**ARTIFACTS**

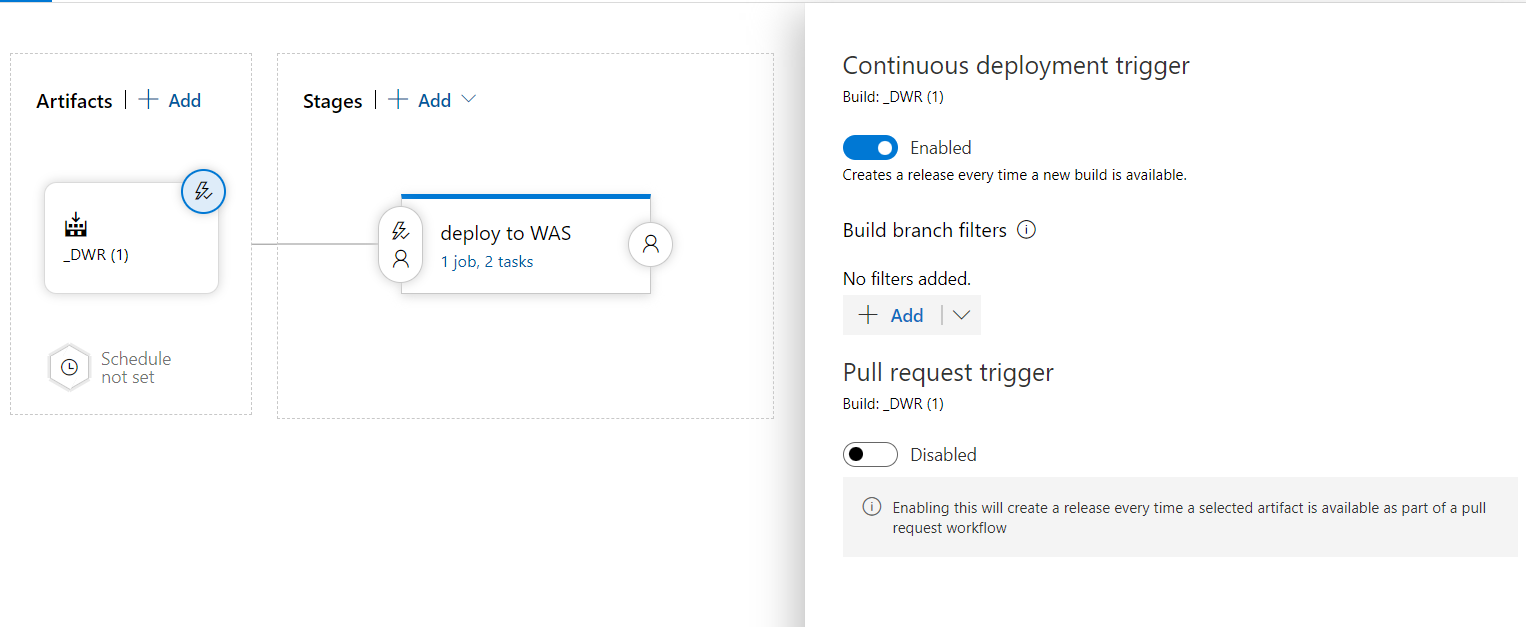


1. **Adding artifacts** - The artifact that we created and publish during CI pipeline we need to configure here.
2. **Source build pipeline -** It asks to select the source build pipeline (which is the name of our CI pipeline)
3. **Default version** - When we select latest it will pick the latest artifact created for the pipeline.

Ex: Let suppose a project has a CI pipeline setup for 2 branch – feature and develop . Then artifacts are created for both the branches hence we need to configure a version. If we want to configure of a specific branch we can select other option.



1. In the artifact we also have to select the **trigger icon** so that we can enable the trigger which creates a release every time we have a new build.

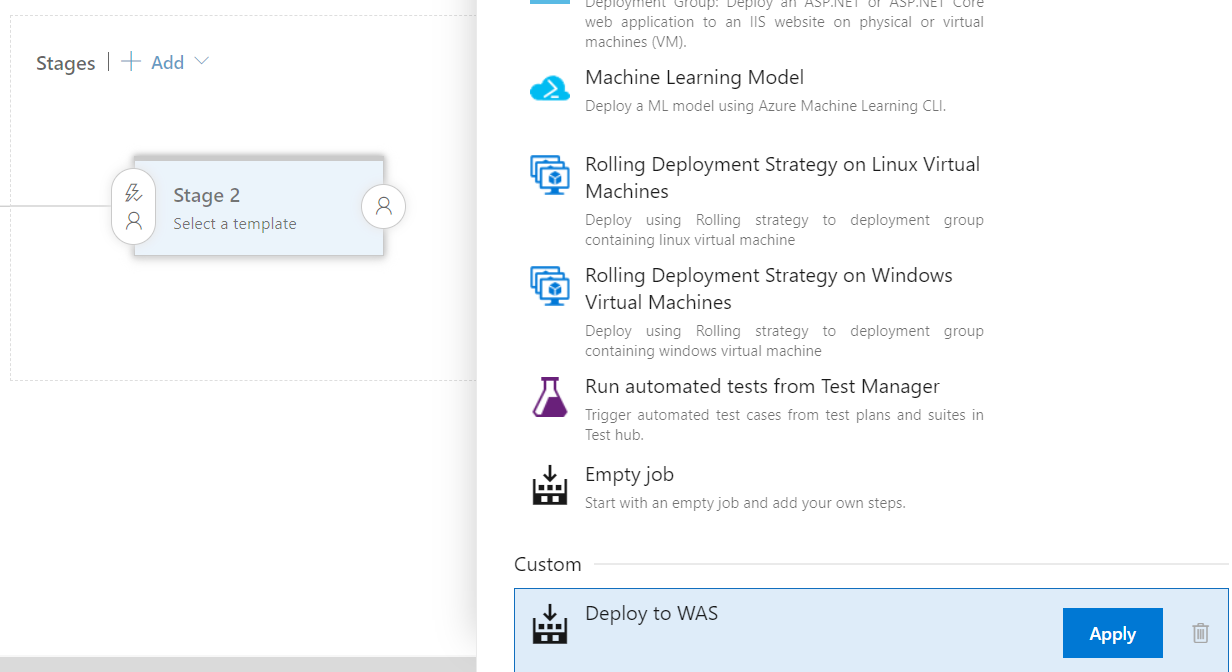


**STAGES**

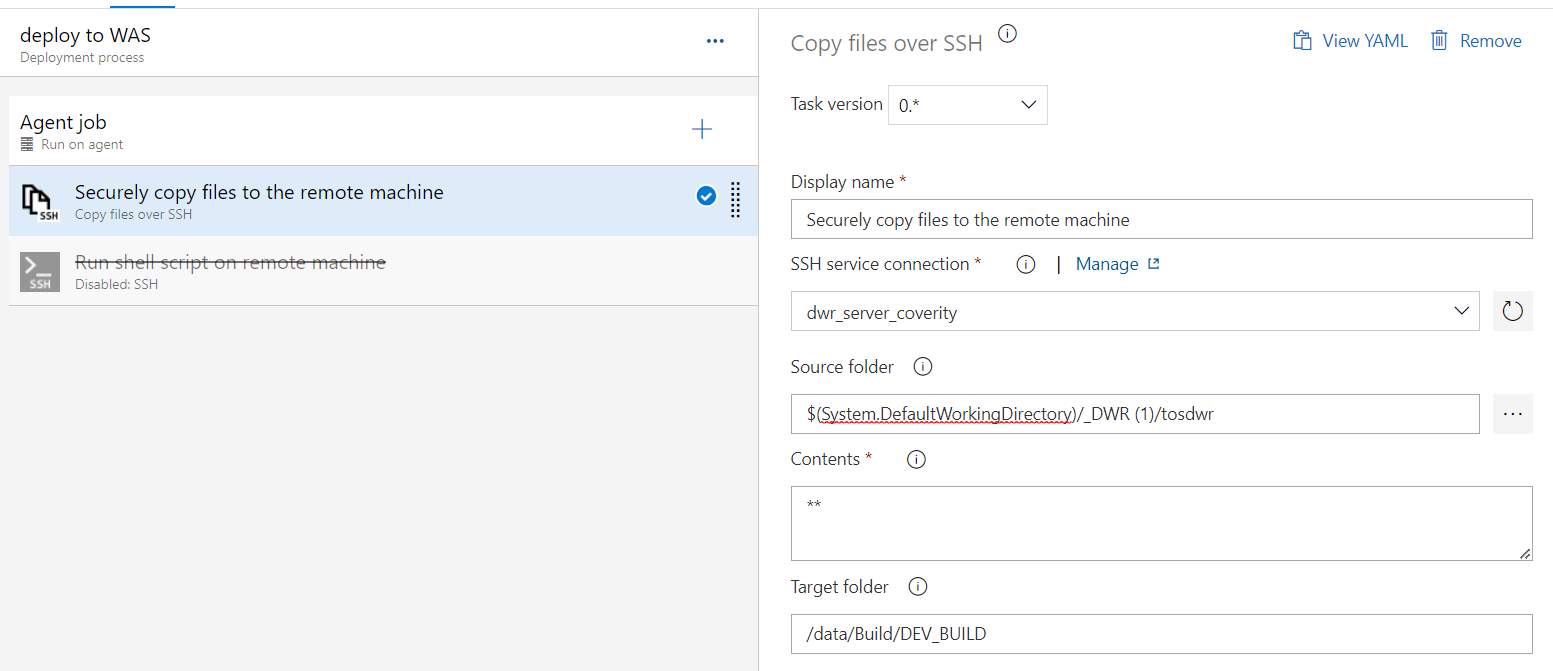
Stages define our release strategy what we are going to do with our artifact and where we are going to deploy it.

**DEPLOY STRATEGY**

* Our task is to deploy in the WebSphere server via our DWR **app-dev-09 server**. Hence, we select a template deploy to WAS in which we define our task.

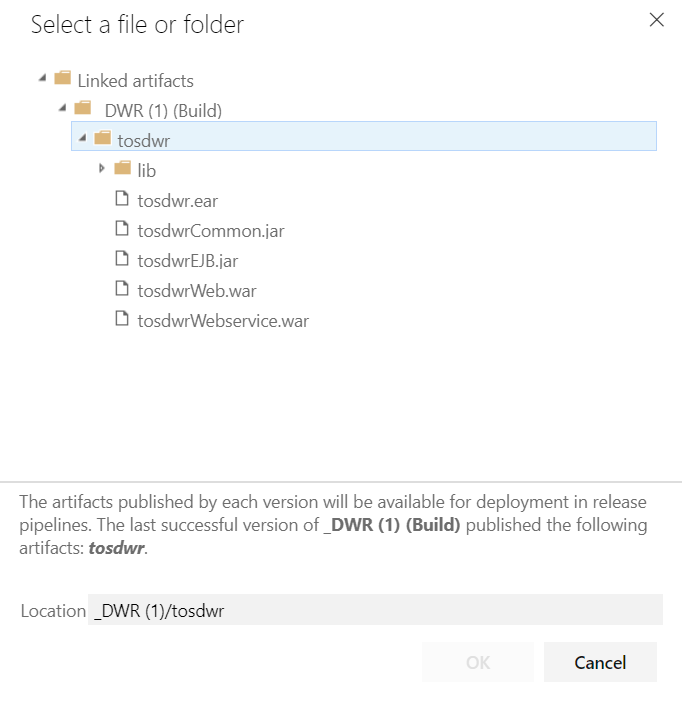


* We create 2 tasks 1) which copies our artifact to DWR server(app-dev-09) and 2) which runs a shell script which does the deployment in the server.

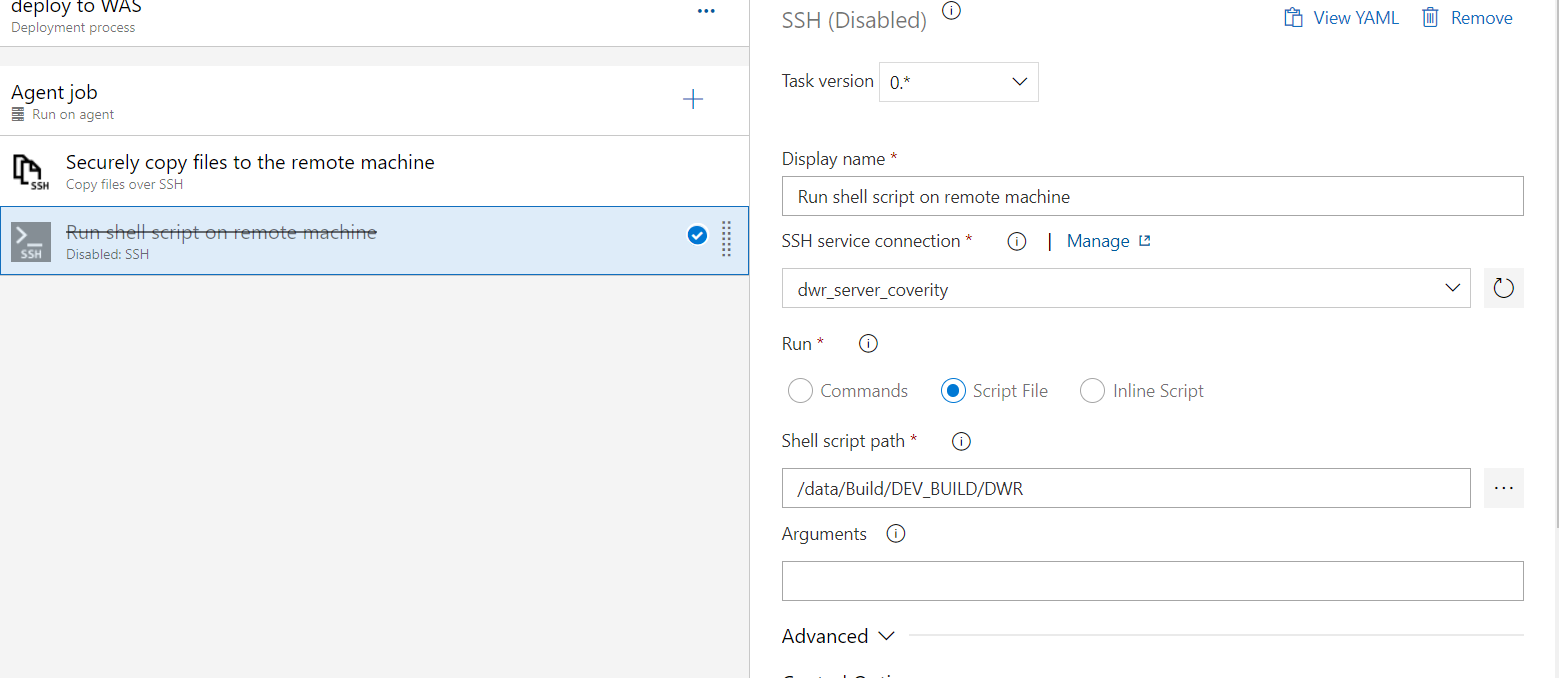


**STEPS for task creation:**

1. Create a ssh service connection to your application server using manage button.
2. **Source folder:** select a source folder from the azure artifact location which we want to move to our DWR server.
3. **Target folder:** this is a location in our server where we want to copy the artifact.



**Steps for 2nd task:**



1. Again, we have to specify our ssh server connection.
2. We need to specify the shell script path location in our DWR server to run and deploy the application in WAS.

**QUESTIONS:**

Q) What is pool used for?

Q) How a separate project- DWR-pipeline is showing

Q) Does the checkout job first clones the repo and then checkout to our branch.

Q) How to define name for the pipeline

Q) ssh service connection to connect to bte-upp-09 box and copy artifact.