Contents

[1.0 Current infrastructure and issues. 1](#_Toc26190592)

[2.0 Assignment 2](#_Toc26190593)

[2.1 - Part 1 - Transformation and Migration to the Public Cloud 2](#_Toc26190594)

[2.2 Part 2 - CI/CD 2](#_Toc26190595)

[3.0 Architecture 2](#_Toc26190596)

[4.0 Azure Components and Tools and Code. 2](#_Toc26190597)

[4.1 Code Structure: 2](#_Toc26190598)

[4.1 MySQL database WordPress sites: 3](#_Toc26190599)

[4.2 Azure Container Registry. 4](#_Toc26190600)

[4.3 WebApp for Docker Containers: 4](#_Toc26190601)

[4.4 GitHub. 4](#_Toc26190602)

[4.5 Azure ARM template: 4](#_Toc26190603)

[5.0 Architectural design of the CI/CD process. 4](#_Toc26190604)

[5.1 Architectural diagram 4](#_Toc26190605)

[5.2 Continues Integration process 5](#_Toc26190606)

[5.2 Continuous Delivery process 7](#_Toc26190607)

[5.2.1 Infrastructure Deployment Release Pipeline 8](#_Toc26190608)

[5.2.2 WordPress Deployment Release Pipeline: 15](#_Toc26190609)

[5.3 Triggering the WordPress Pipeline. 23](#_Toc26190610)

# 1.0 Current infrastructure and issues.

The customer is currently hosting ten WordPress sites using WordPress multisite in a private datacentre.

* \*10 WordPress instances\*
* They achieve HA by using 2 servers and having 2 copies of their multisite
* For the database they are using 2 MySQL servers behind HA proxy to achieve HA.
* The past few months, they have been having a lot of issues because some of their websites have increased in popularity, especially during certain timeframes.
* So, the customer has decided to move away from WordPress multisite and have independent WordPress applications.
* They have also pointed out that they have 5 more sites in the making that will reach Production in the next 12 months.
* The client is only interested in developing the WordPress sites from an application perspective. They work using GIT repositories, and they have agreed to provide access to the application source code in one or more repositories.
* We have undertaken the task to design the future state of this environment in the public cloud. The solution needs to:
* Be scalable and flexible.
* Be futureproof and expandable with new WordPress sites with minimal effort.

# 2.0 Assignment

## 2.1 - Part 1 - Transformation and Migration to the Public Cloud

Use control version system to IaC templates.

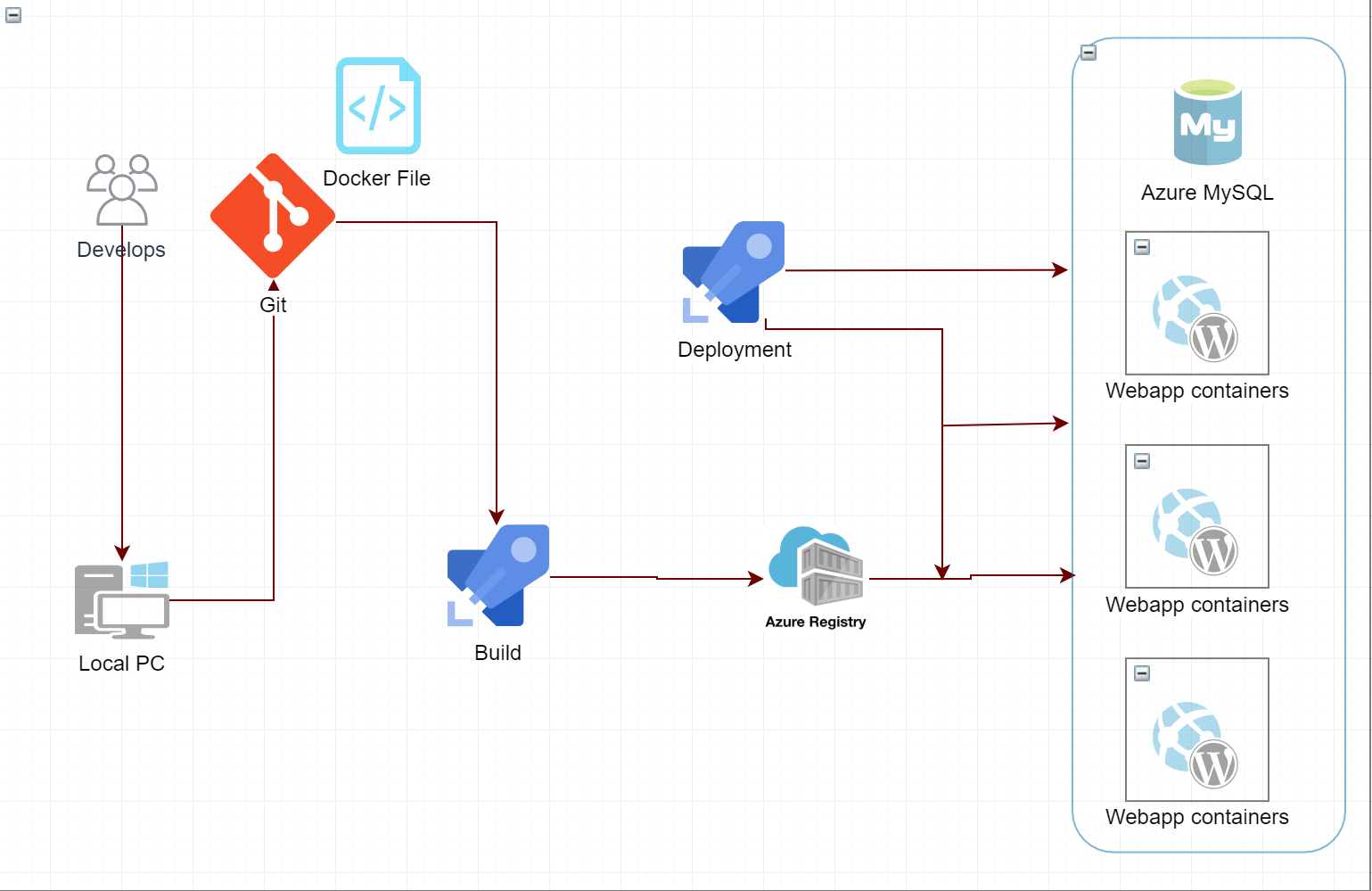
Provide a design for the designated Azure architecture.

Provide Azure ARM templates.

## 2.2 Part 2 - CI/CD

Provide a design for the CI/CD pipeline that you will use to deliver the changes to the environment, every time the client updates any of their WordPress applications in GIT

# 3.0 Architecture



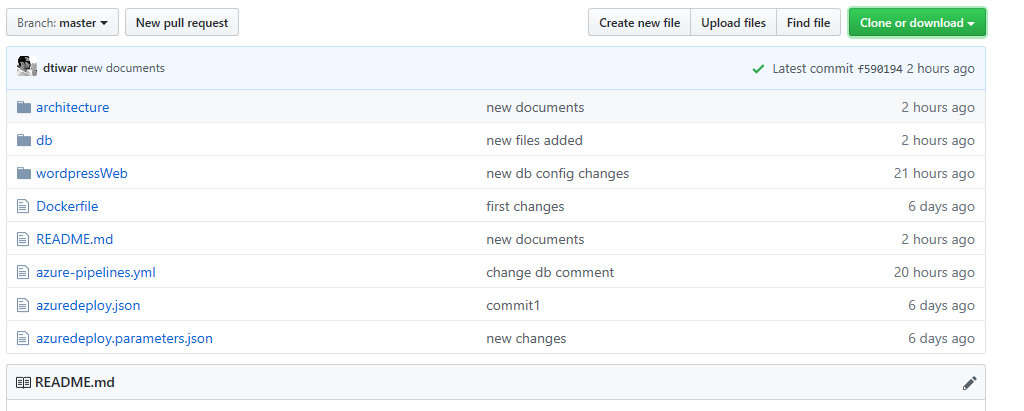
# 4.0 Azure Components and Tools and Code.

## 4.1 Code Structure:

WordPress code is available in GitHub <https://github.com/dtiwar/WordpressWebsites.git>

All ARM template is located <https://github.com/dtiwar/SentiaARMTemplate.git>

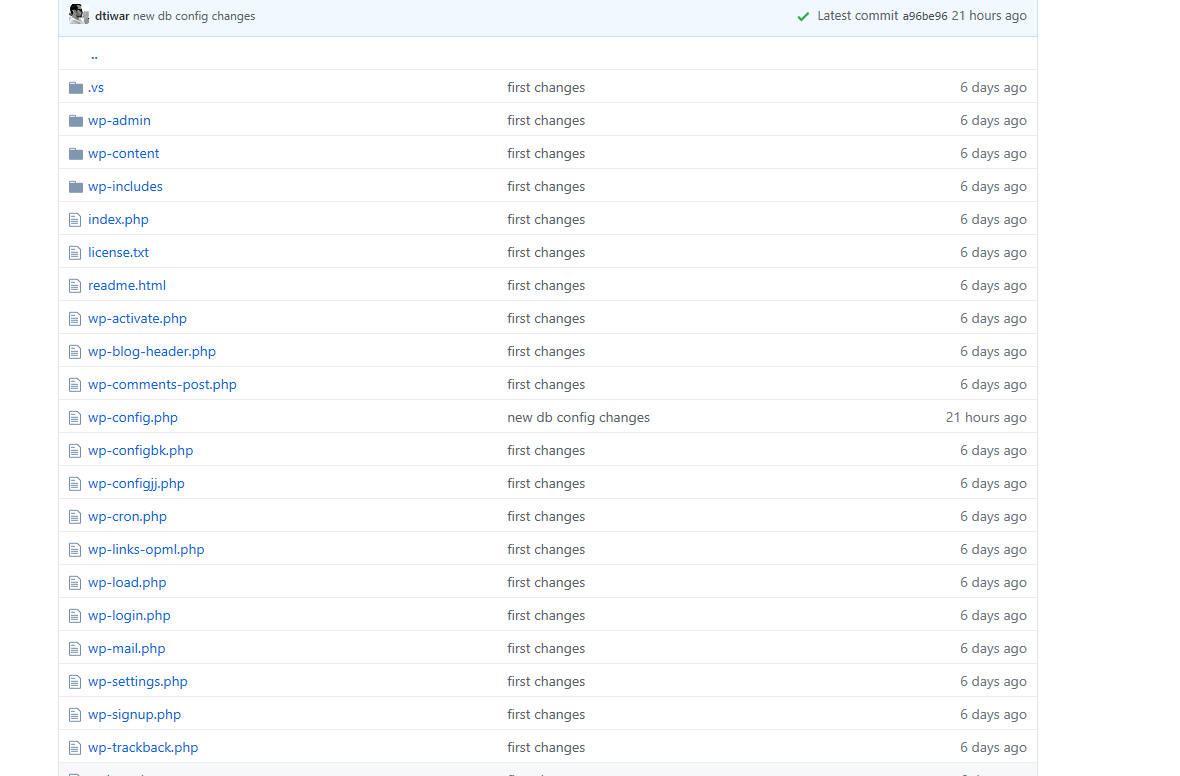
The code repository structure looks like this:



[**WordpressWeb**](https://github.com/dtiwar/WordpressWebsites/tree/master/wordpressWeb) **folder** – sample WordPress files.

**Db folder** – sample WordPress script DB file that we need to restore into Azure Database for MySQL

**Dockerfile** – for building the docker image.



We can run the following docker command on our local machine to run the sample WordPress locally. It will connect to MySQL DB on Azure, we might need to add our IP to the firewall rule of this instance.

docker run -e DB\_ENV\_HOST=[your mysql db url]:[your mysql port number] -e DB\_ENV\_USER=[your mysql db user name] -e DB\_ENV\_PASSWORD=[your mysql db password] -e DB\_ENV\_NAME=[your mysql database name] -p 5000:80 -d [your docker image name]

## 4.1 MySQL database WordPress sites:

* Azure Database for MySQL was used. The WordPress service was deployed as stateless application, it means "do not store data or application state to the cluster or to persistent storage". That's why is used MySQL as platform as service. Of this way, the WordPress sites will be easily scalable.
* In addition, use MySQL as an external service have the following benefits:
* Easy backups
* Easy maintainability
* Compliance
* High Availability
* Create "databases on the fly".

## 4.2 Azure Container Registry.

To store container images Build and store WordPress docker image. it store, secure, scan, replicate, and manage container images and artefacts with a fully managed Connect across environments, including Azure Kubernetes Service and Azure Red Hat OpenShift, and across Azure services like App Service, Machine Learning, and Batch etc.

## 4.3 WebApp for Docker Containers:

To deploy container-based web apps. Just pull container images from Docker Hub or a private Azure Container Registry, (we used Azure container Registry here) and Web App for Containers will deploy the containerized app with your preferred dependencies to production in seconds. The platform automatically takes care of OS patching, capacity provisioning, and load balancing.

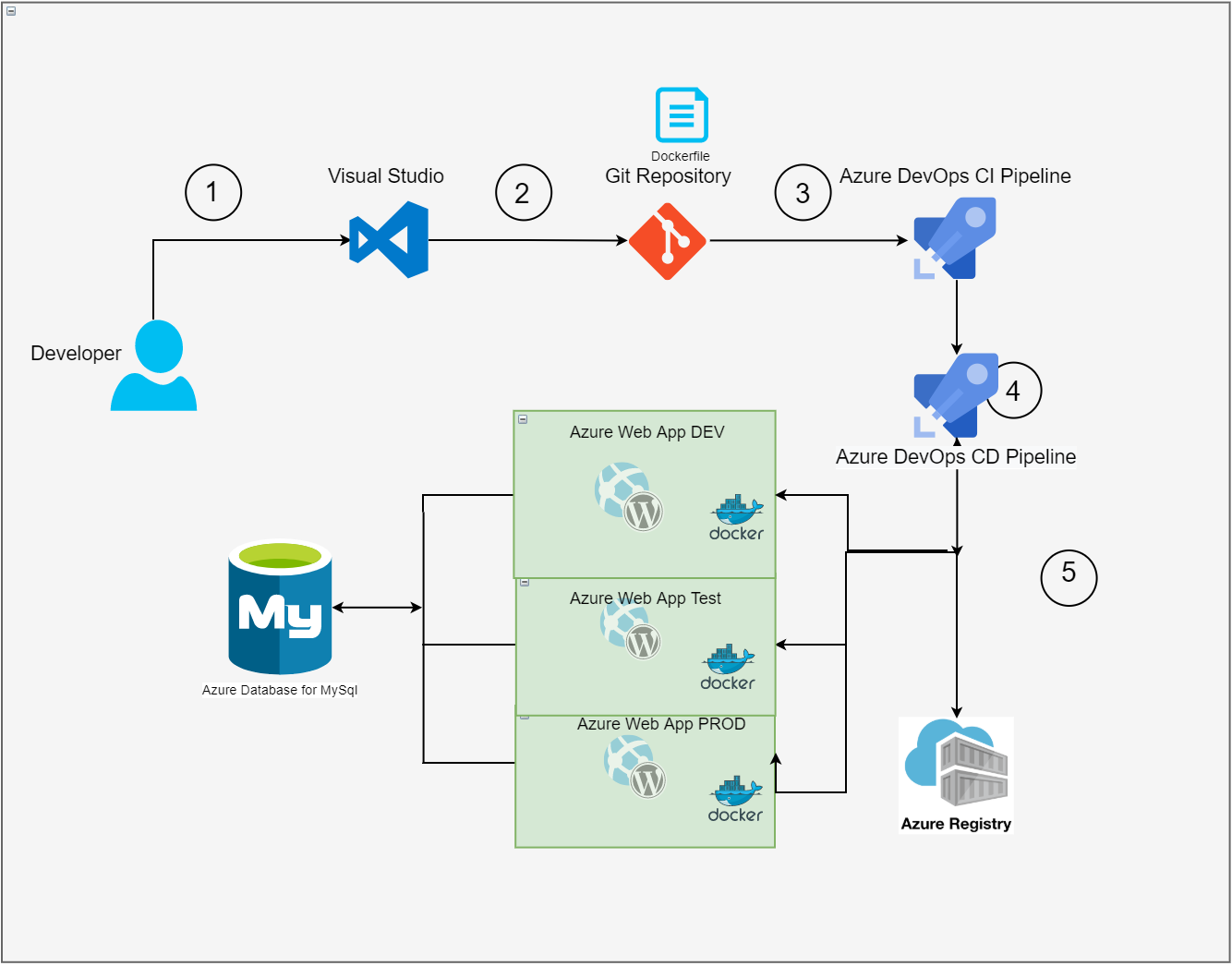
## 4.4 GitHub.

Git hub used for version control system

## 4.5 Azure ARM template:

# 5.0 Architectural design of the CI/CD process.

## 5.1 Architectural diagram



The Build pipeline contains the activities that impact on the general artefact The Release pipeline contains the activities that impact on the environment (Dev, testing, Preproduction and production)

## 5.2 Continues Integration process

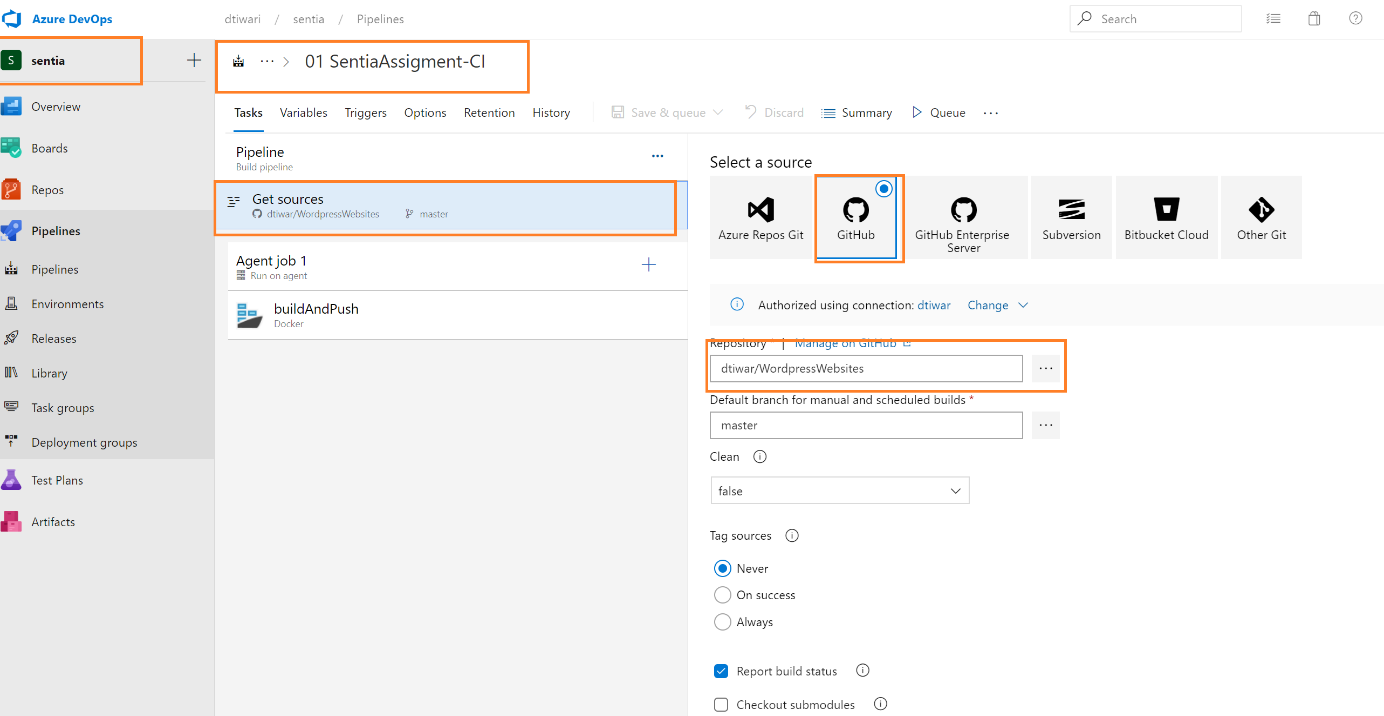
Here will be only activities related to the WordPress application sites upgrade process, via Docker.

In the CI process I am getting the source code from [github repository](https://github.com/dtiwar/WordpressWebsites/tree/master/wordpressWeb)

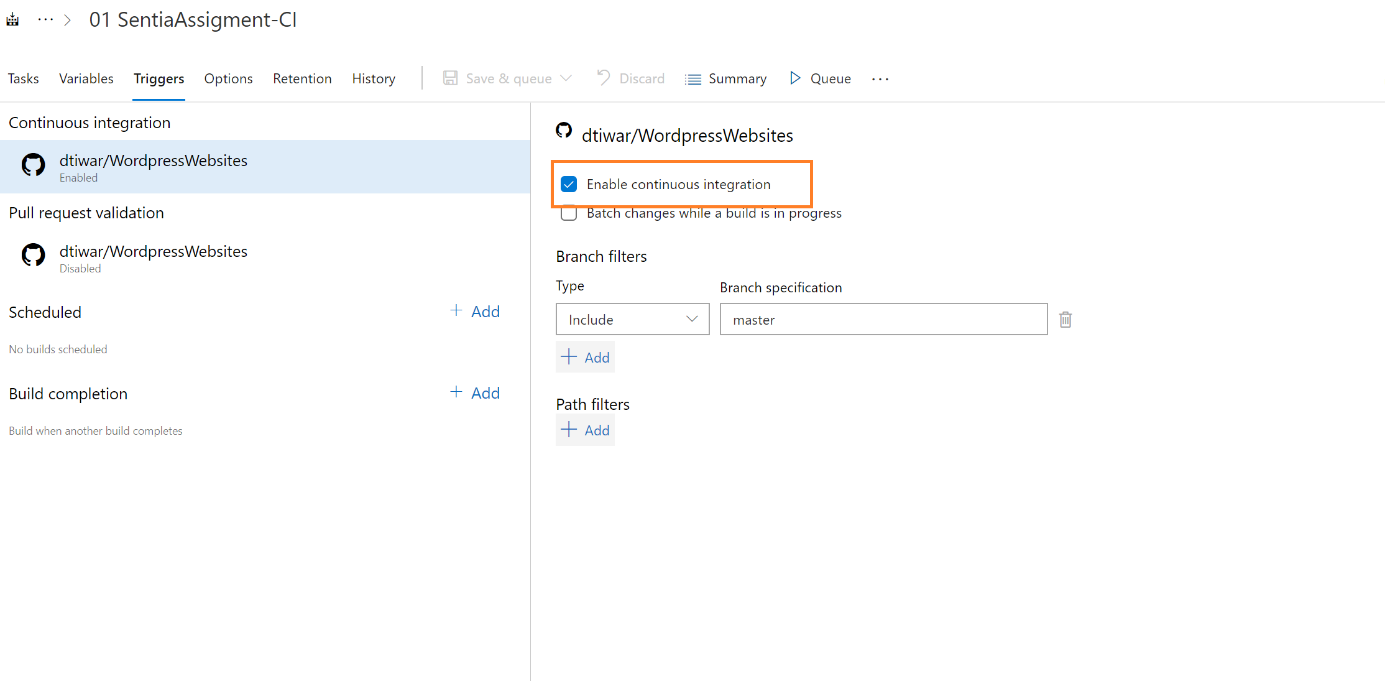
In the CI approach will be all the activities related with the creation and uploading Wordpress Docker image to ACR [These activities](https://github.com/bgarcial/sentia-assessment/blob/master/Documentation/README.md#44-azure-container-registry), in order to automatize the continuous integration every time the client updates any of their WordPress applications in GIT.

CI-CD process is build using Azure for DEVOPS tool. The link for the project is here

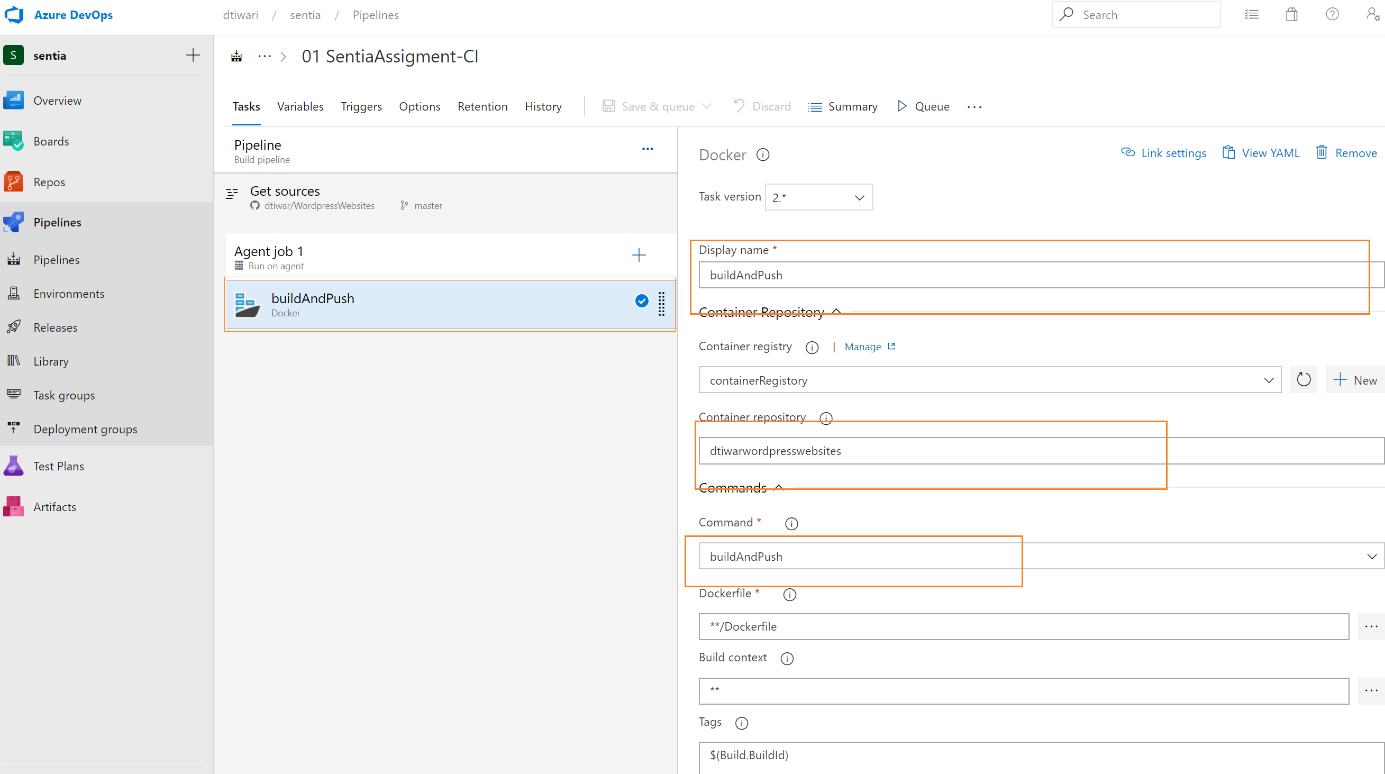
<https://dev.azure.com/dtiwari/sentia>



So the CI build pipeline has enabled the continuous integration trigger: Below



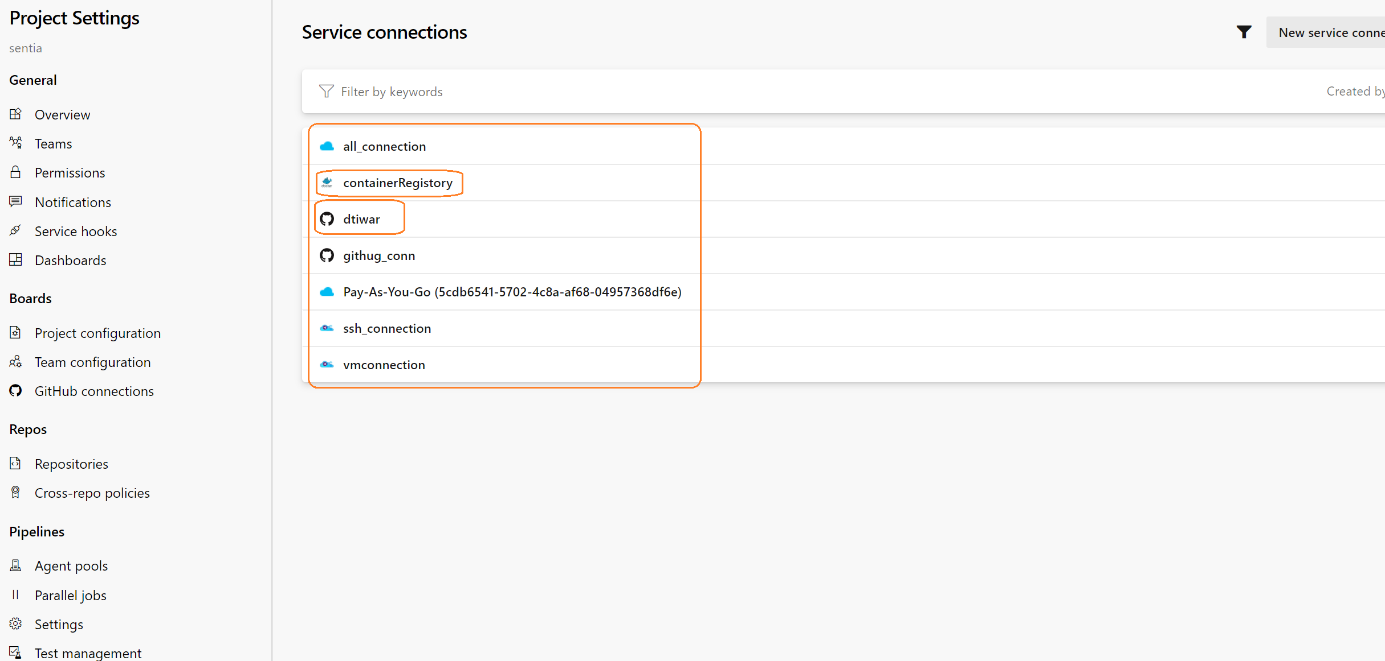
The build process will take place and its will create docker image and push this image to the container registory.



## 5.2 Continuous Delivery process

The continuous delivery process where the environment related activities will be executed. So, the CD process has divided into two categories.

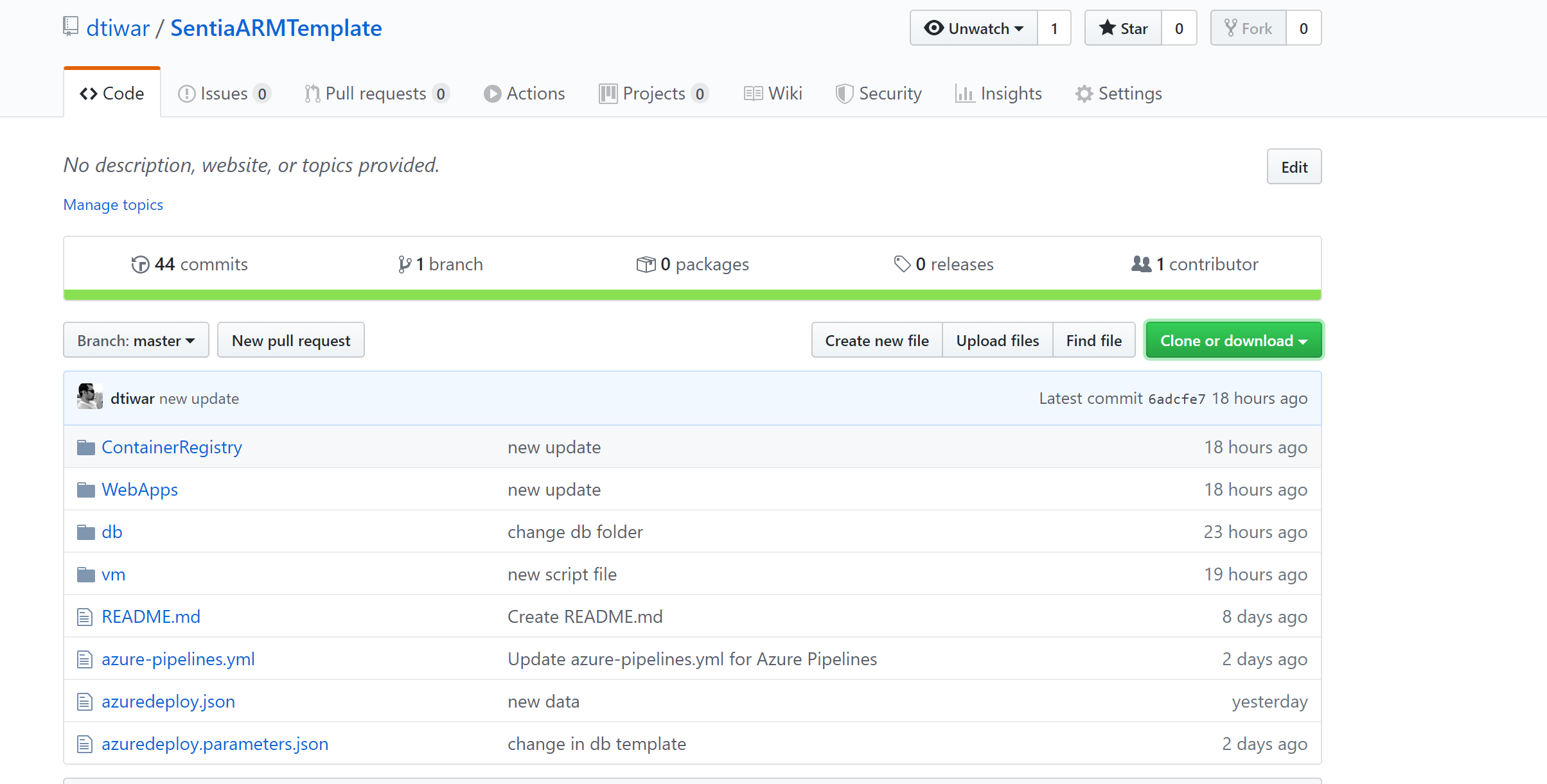
Before that we have to make some project service connection:



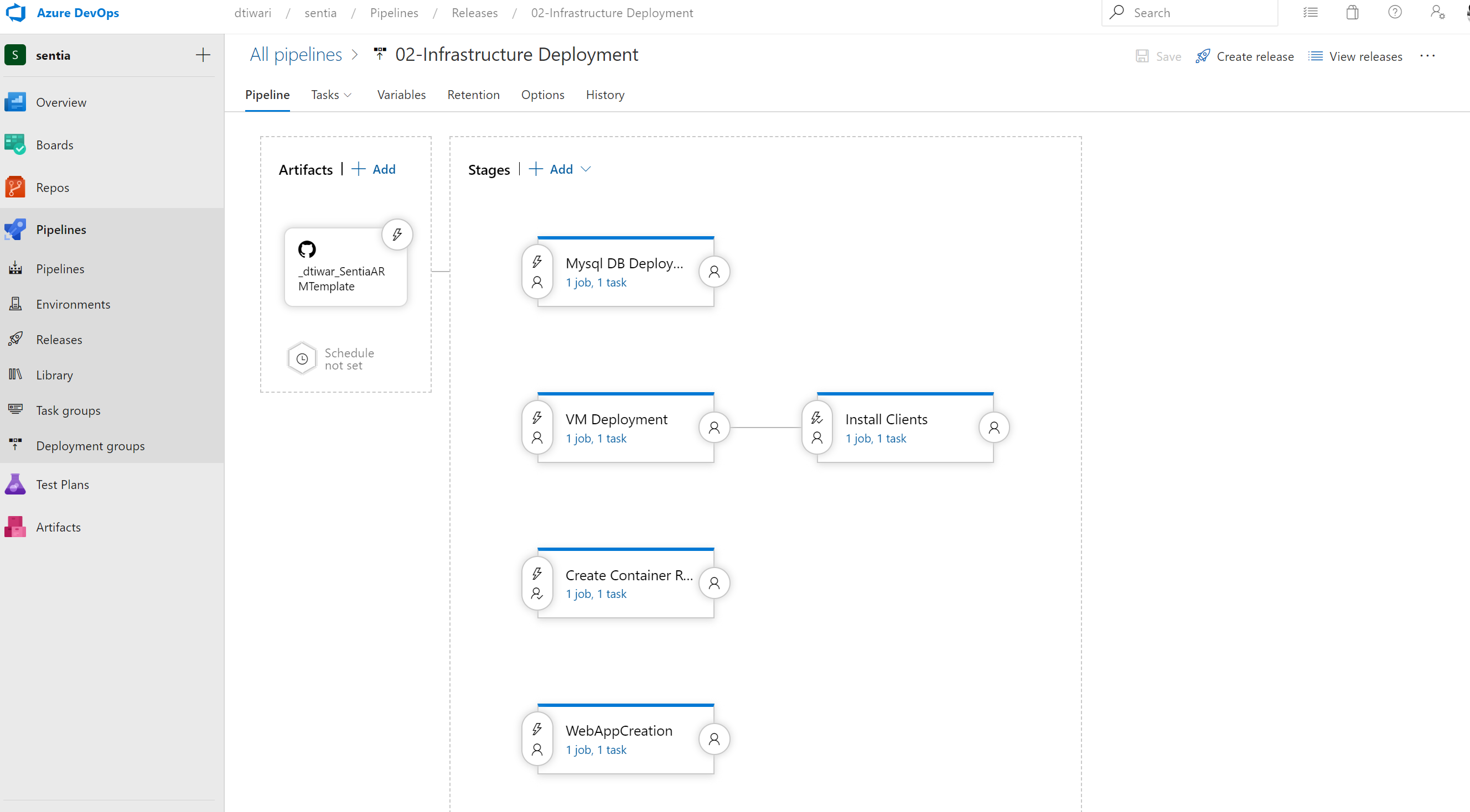
### 5.2.1 Infrastructure Deployment Release Pipeline

All the ARM template used for creating the azure infrastructure is link here

<https://github.com/dtiwar/SentiaARMTemplate.git>



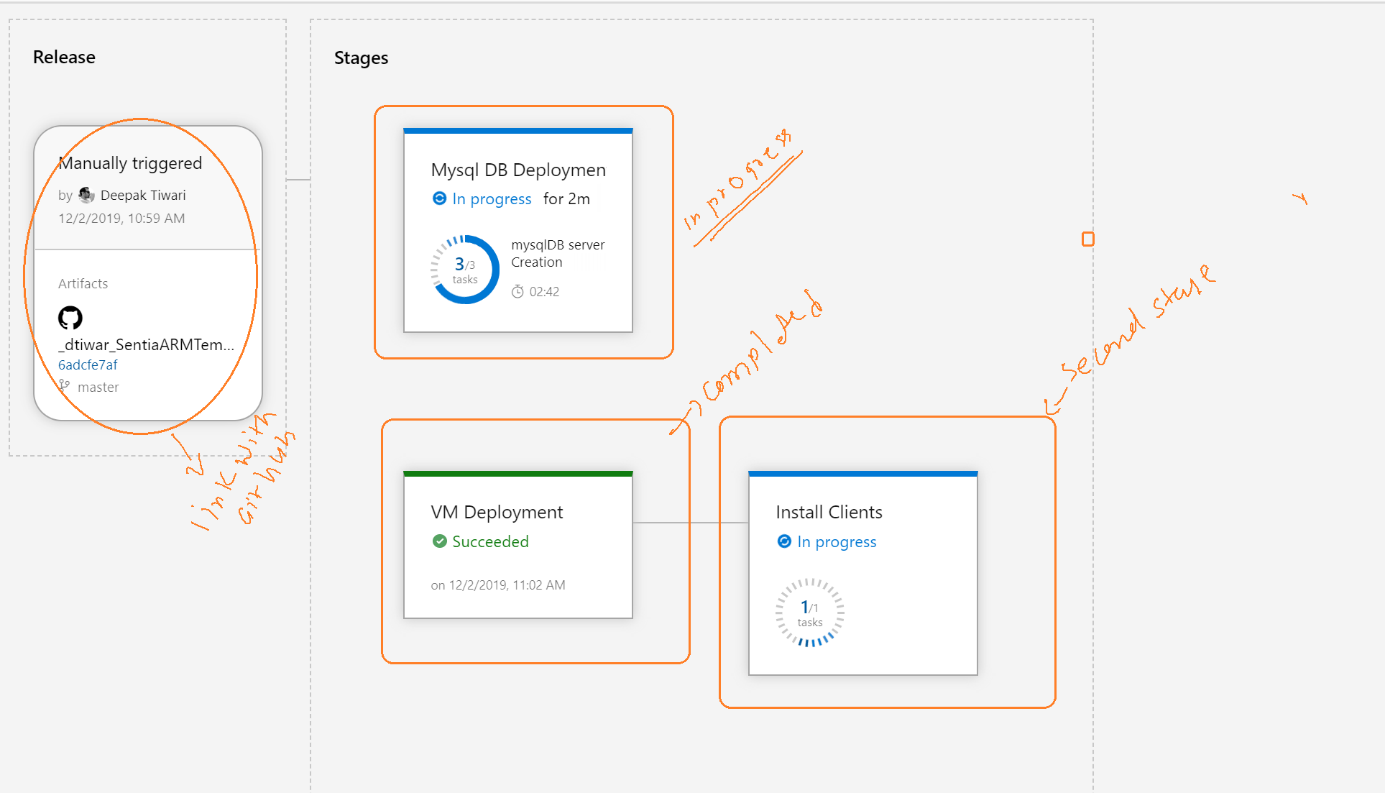
Azure DevOps Pipeline for Infrastructure setup. Using ARM template



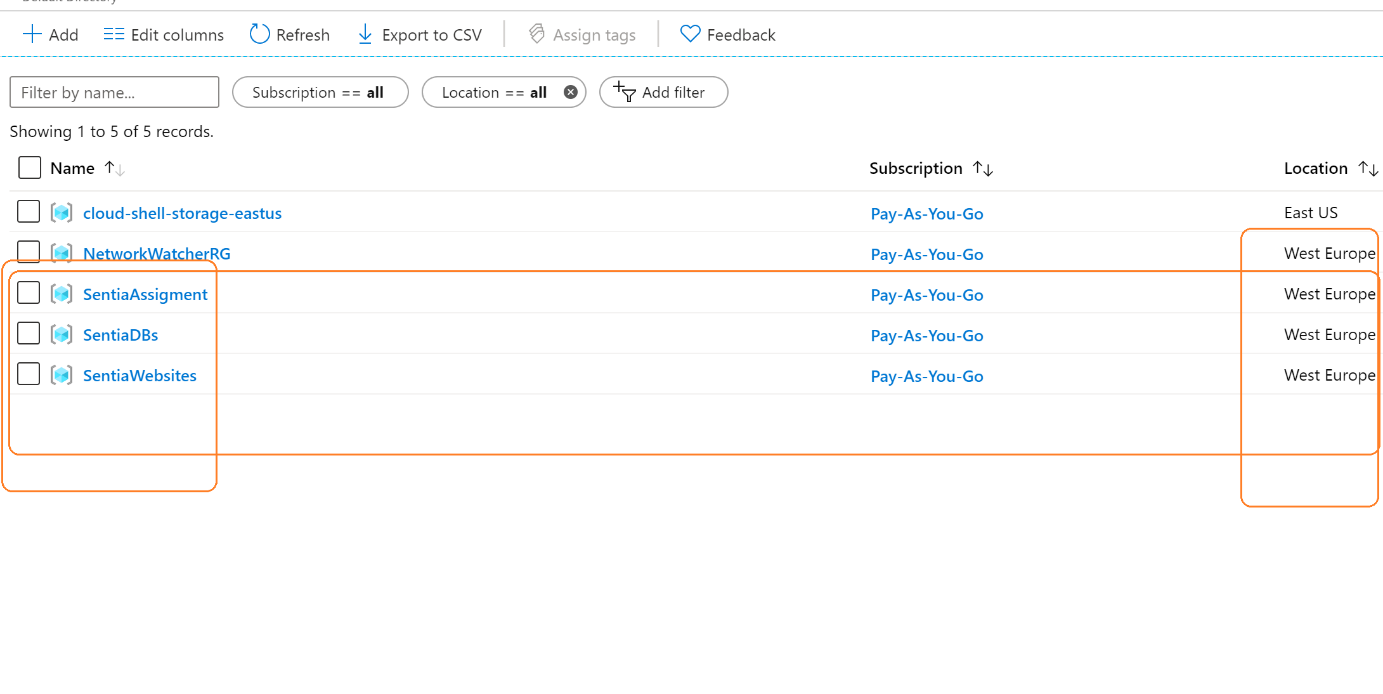
I have split the different infrastructure creating into different stage because I wanted to follow the modular way of building the infrastructure these stages can be run independently

Also we can add all these ARM template into single and run It once.

We can create a new release from infrastructure creation pipeline . And we can choose them to create independently.

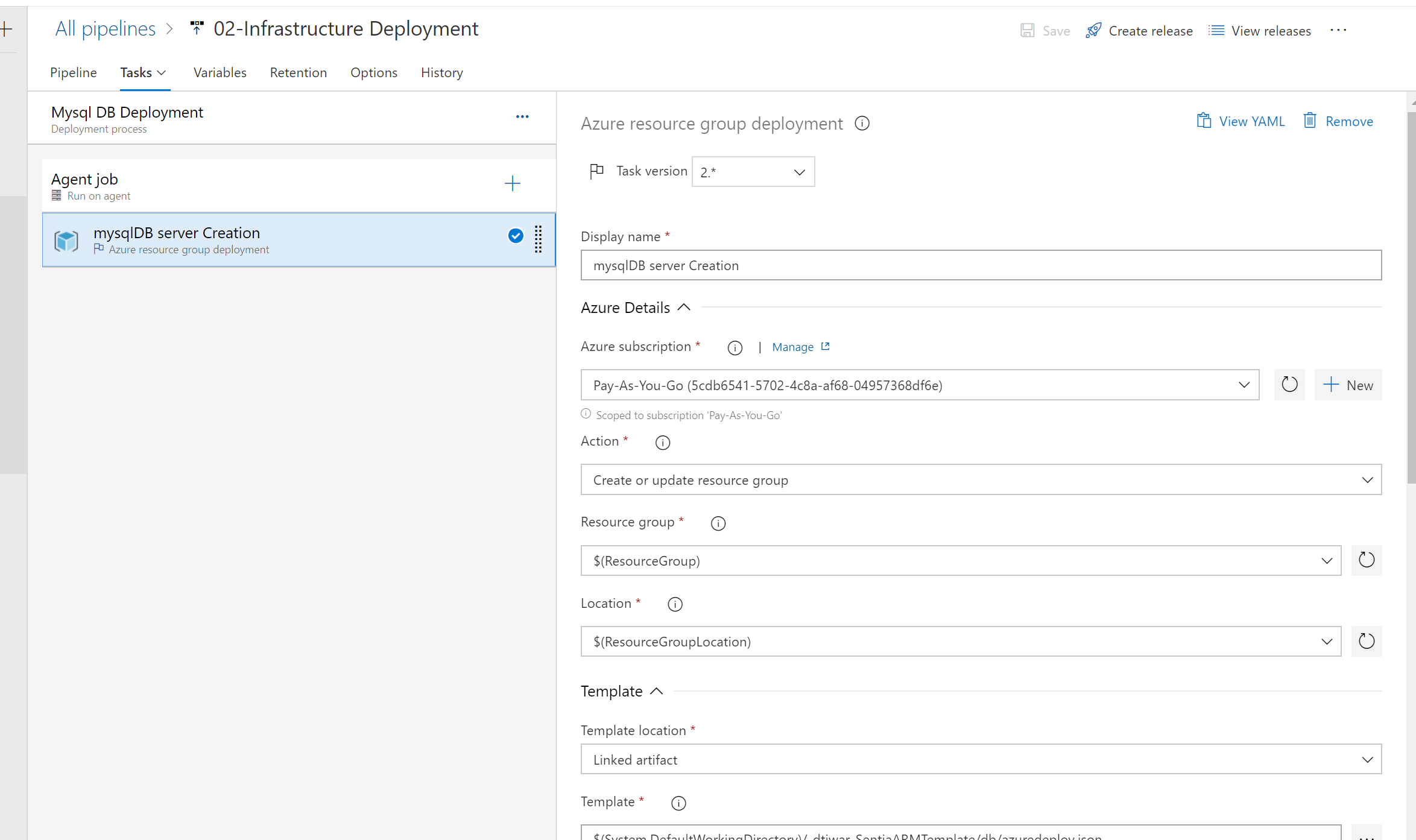


after run it resources will be created inside the below resource groups:



###### 5.2.1.1 MySQL DB Deployment:

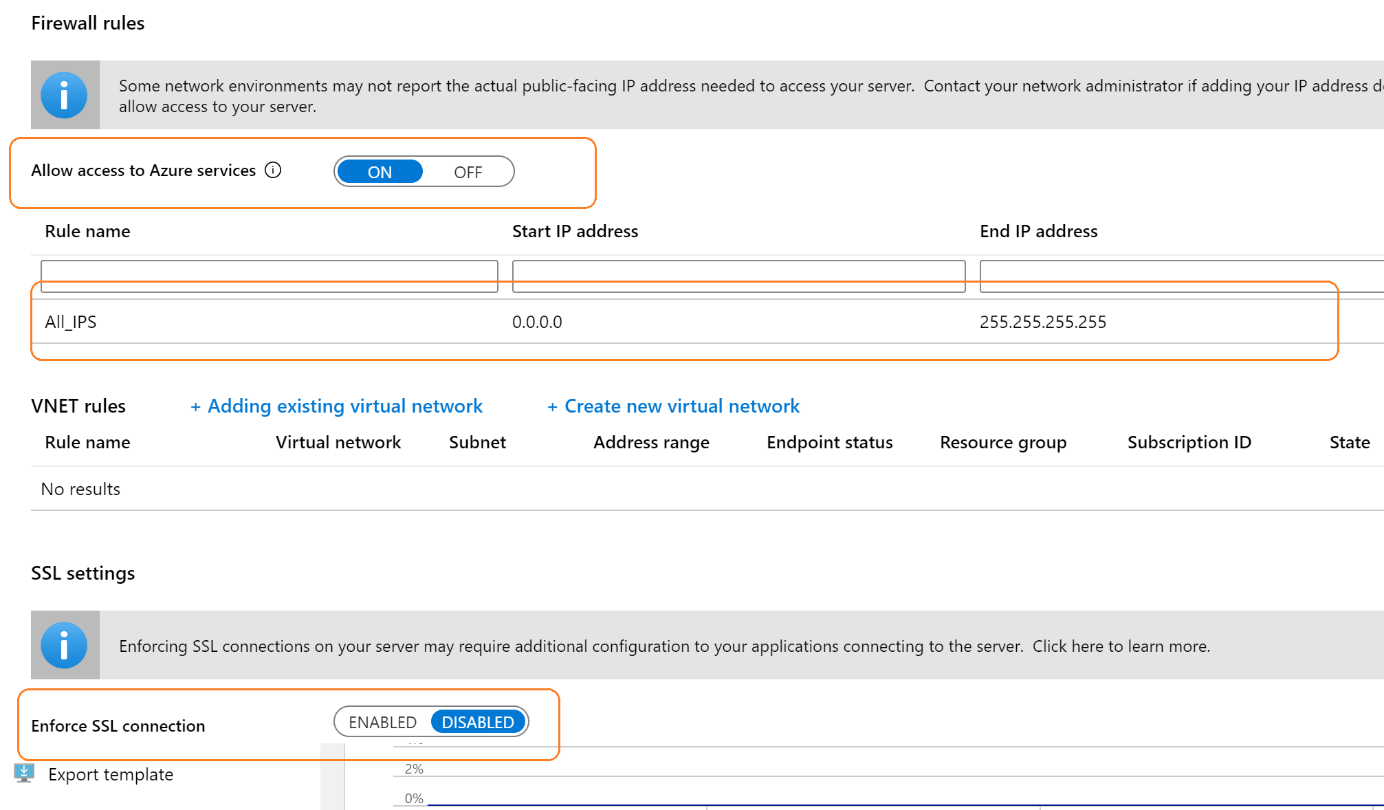
This stage will create a MYSQL DB for azure instance. This stage will take the ARM template from GIT hub and create the mysql instance in azure.



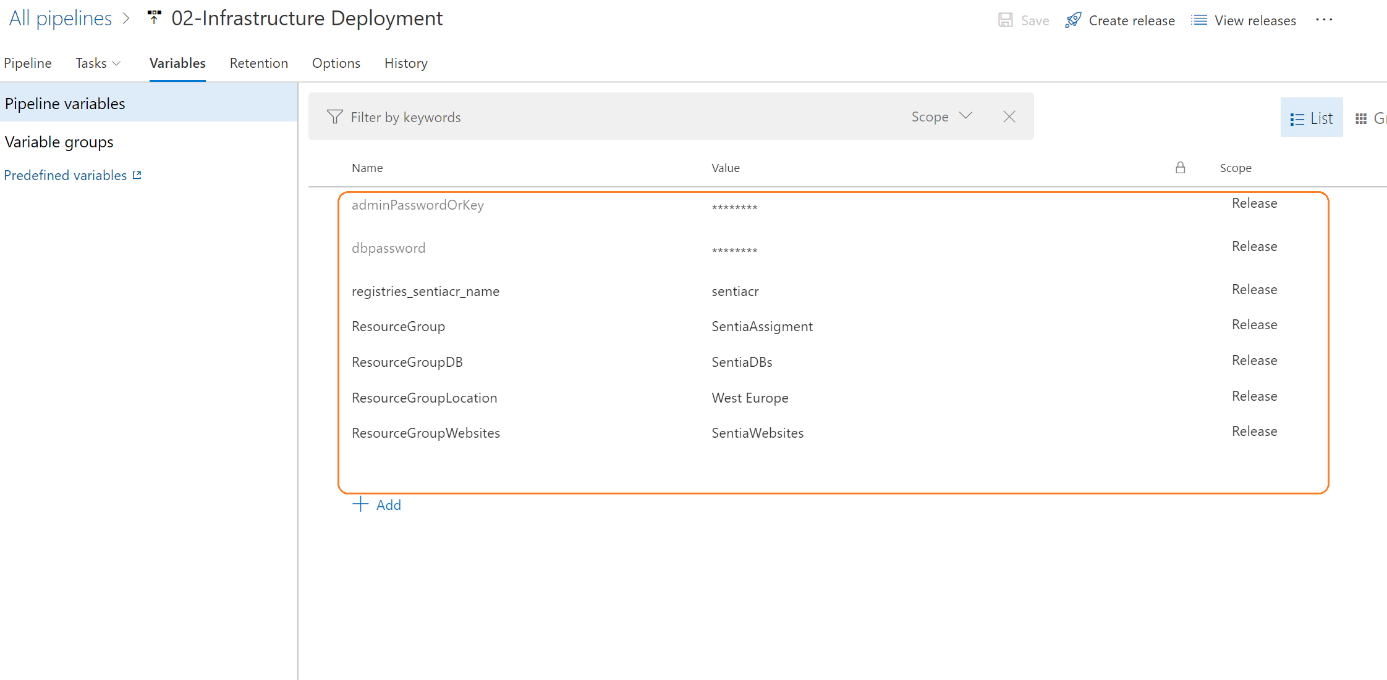
After deployment MySQL for Db will be created with configuration described in the ARM template:



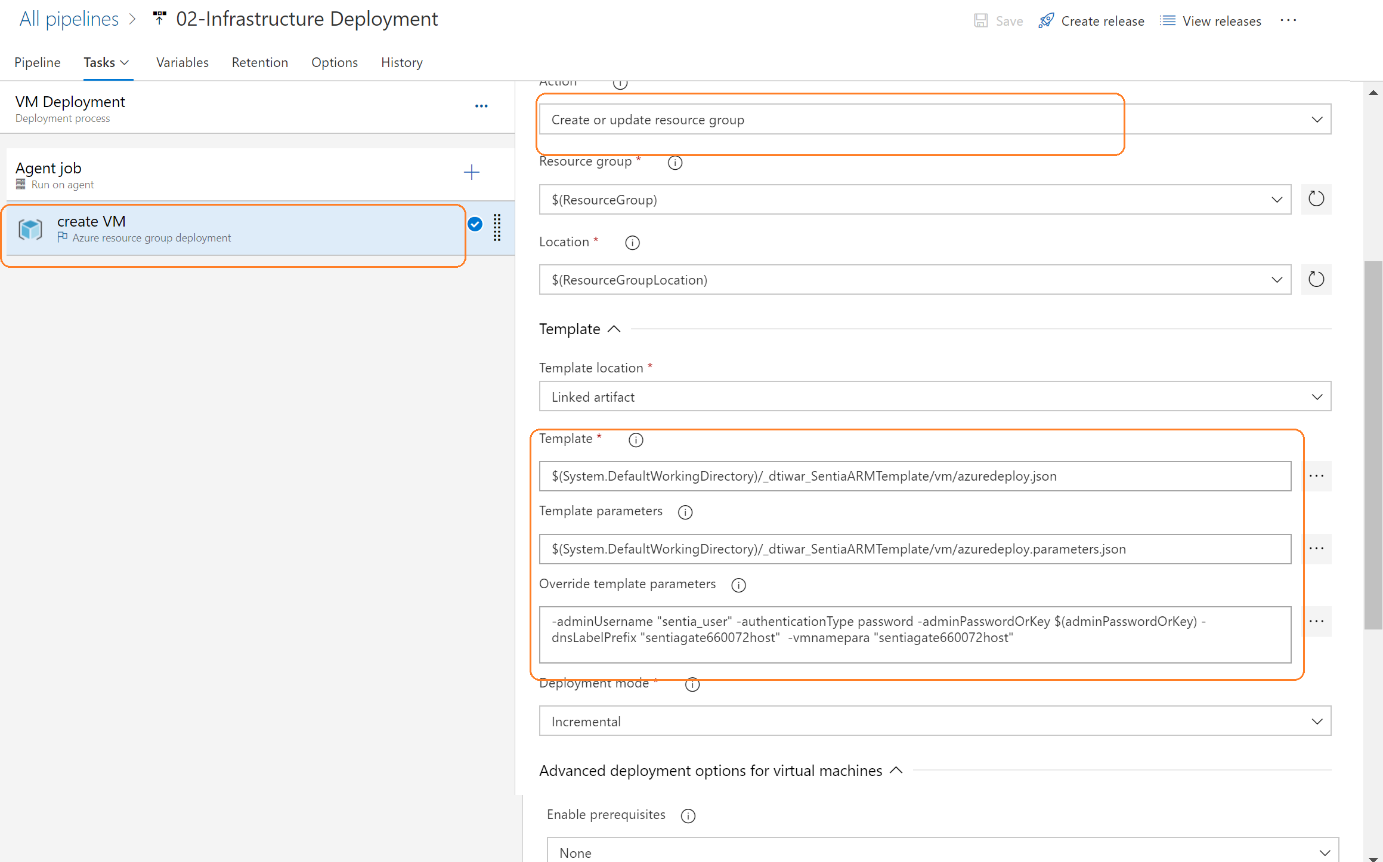
Special consideration: Firewall rules added within a template to allow access to azure services. Also for connecting my MySQL Workbench I have open the addition firewall rules just to demonstrate we can add the firewall rule within the template.

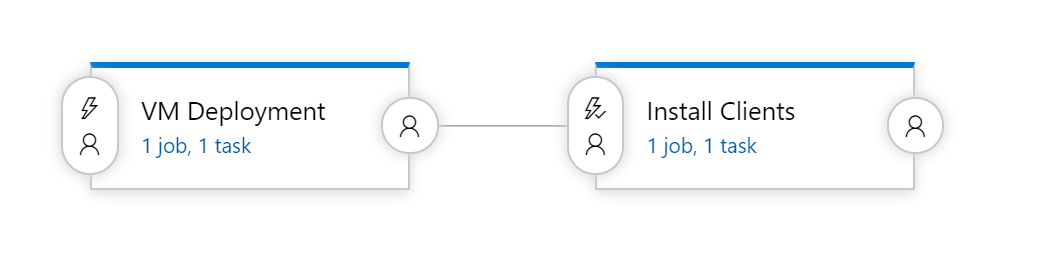


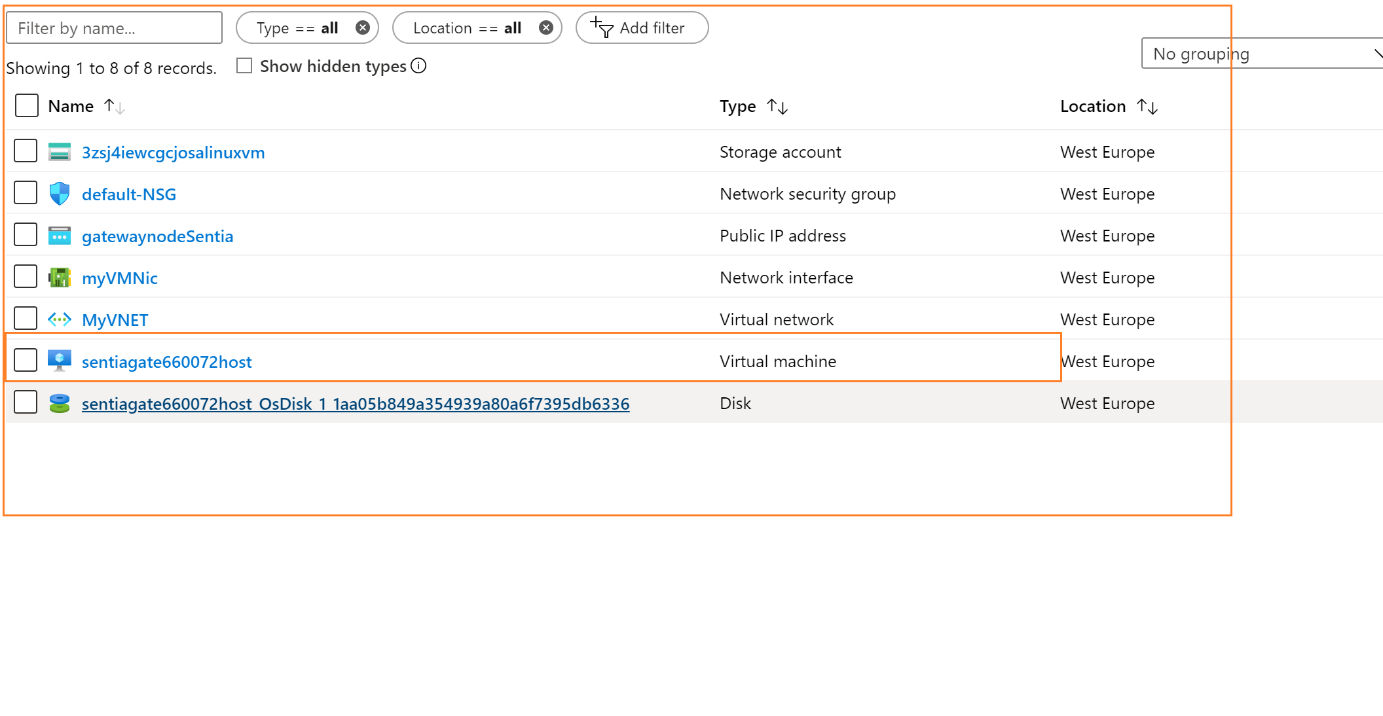
Pipeline variable used:



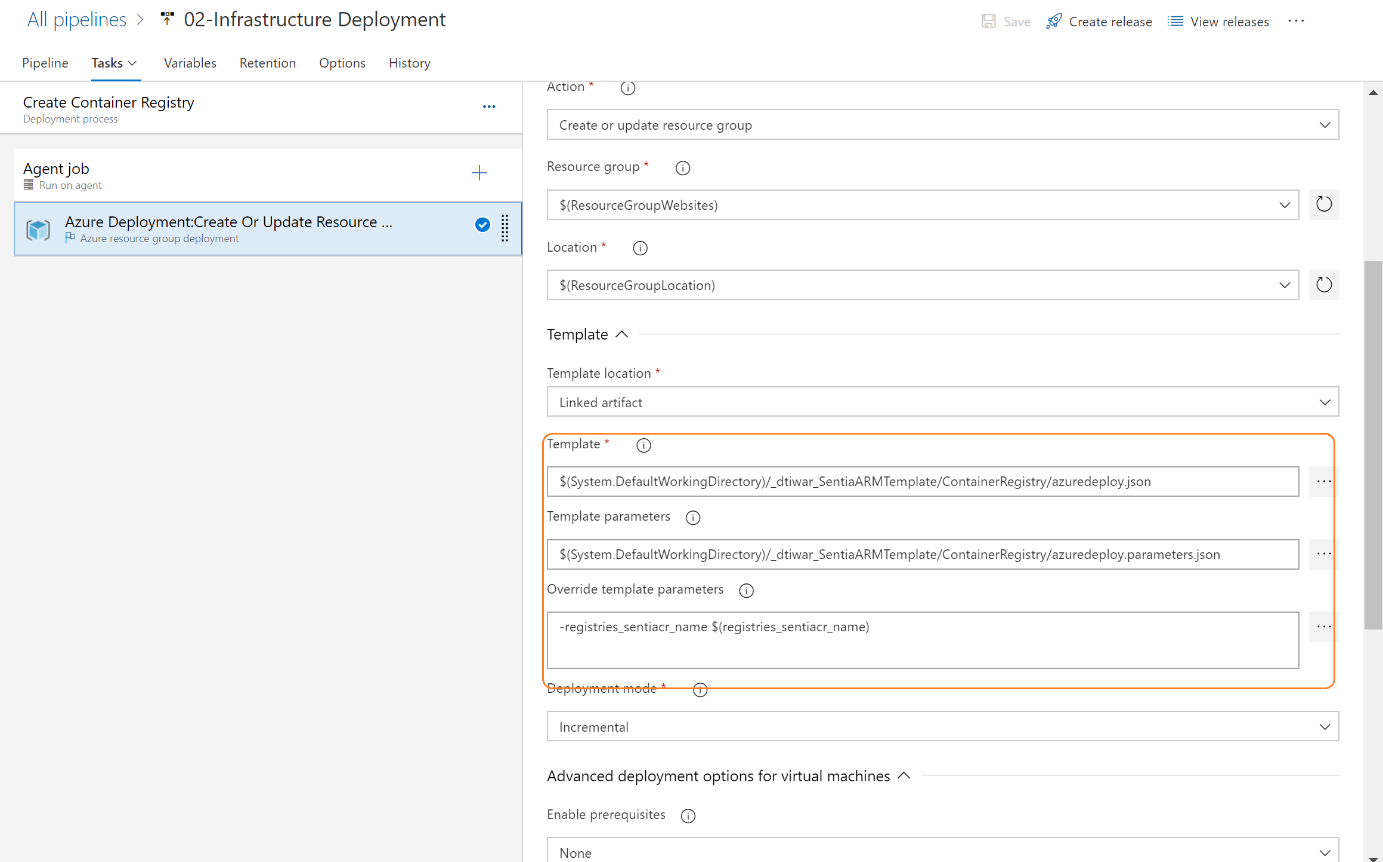
###### Virtual Machine Deployment: I used Virtual machine for Code deployment utility purpose and also install MySQL client on this in this virtual machine. Later described in this document. This template will cerate a Virtual machine with addition supporting resource like public DNS security group storage account and VNET etc.

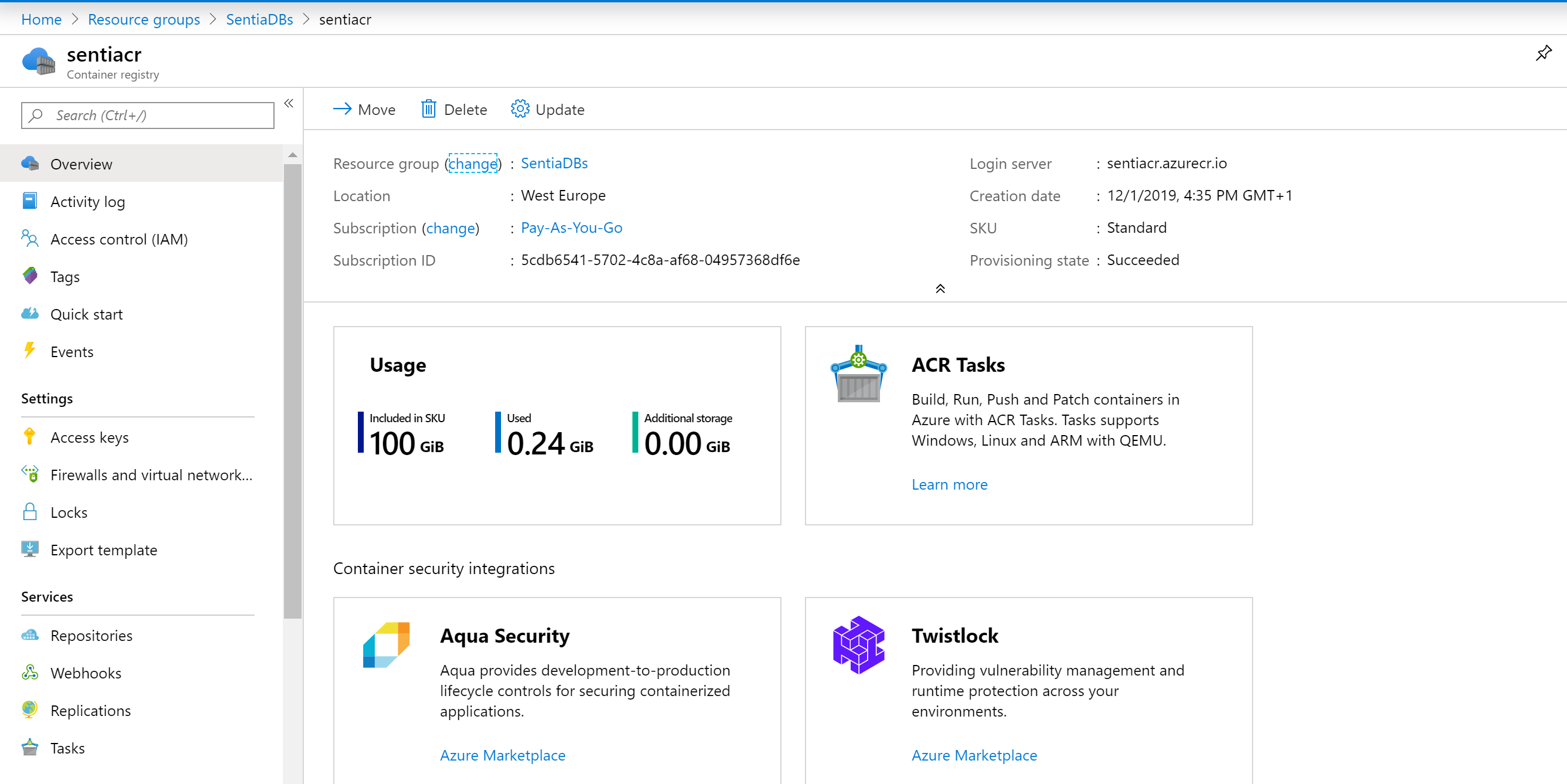




After template Run: 

###### Create Container Registry: for creating container registry :



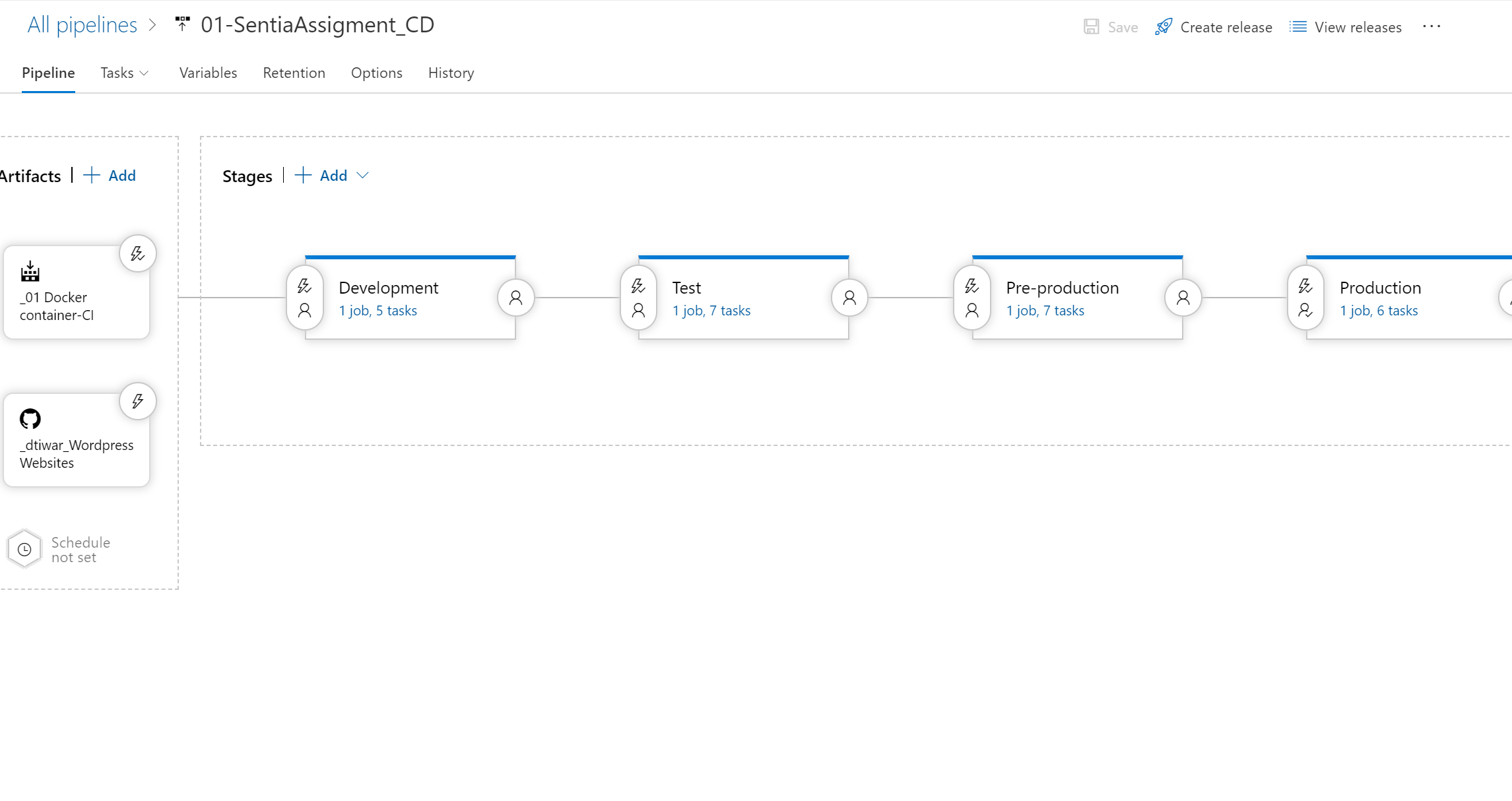




### WordPress Deployment Release Pipeline:

The Release pipeline is show below where it takes the build images (Docker images from container repository )from CI pipeline and deploy into different platform environment like DEV, TEST preproduction and Production sequentially.

First, we need to create a new empty release definition, go to **releases**, click on the **+ icon** and choose **create release definition**, select **empty process** as the template.



The WordPress websites will be deployed here. Few additional services will be required to setup this.

We need to add artefacts to our pipeline. Click on **add** in the Artefacts section (step 1), a right blade will be presented, choose the following:

* Source type – Build
* Project – The relevant project name
* Source – The name of the build definition we created
* Default version – Latest
* Source alias – Keep the default

After clicking onAdd, click on the **trigger icon** under the Artefacts section a right blade will be shown, choose to **enable the continuous deployment** trigger.

We need to move to configure the environments section, click on **pre-deployment conditions** icon (step 3), select **After release** trigger and close the right blade.

Next step is adding tasks to the **development pipeline**, click on **tasks**.

#### 5.2.2.1 Release Pipeline Variable and connections:

In order to make the deployment pipeline more dynamic I have created environmental variable each environment will be having a different value from these variables.

$(desturl) – App Service URL of Dev environment

$(migrationfile) – temp file name when executing DB backup and restore operation

$(mysqldestdb) – DB name of Dev environment

$(mysqlhost) – Server name of Azure Database for MySQL

$(mysqlpass) – Password for Azure Database for MySQL

$(mysqlport) – The Port for Azure Database for MySQL

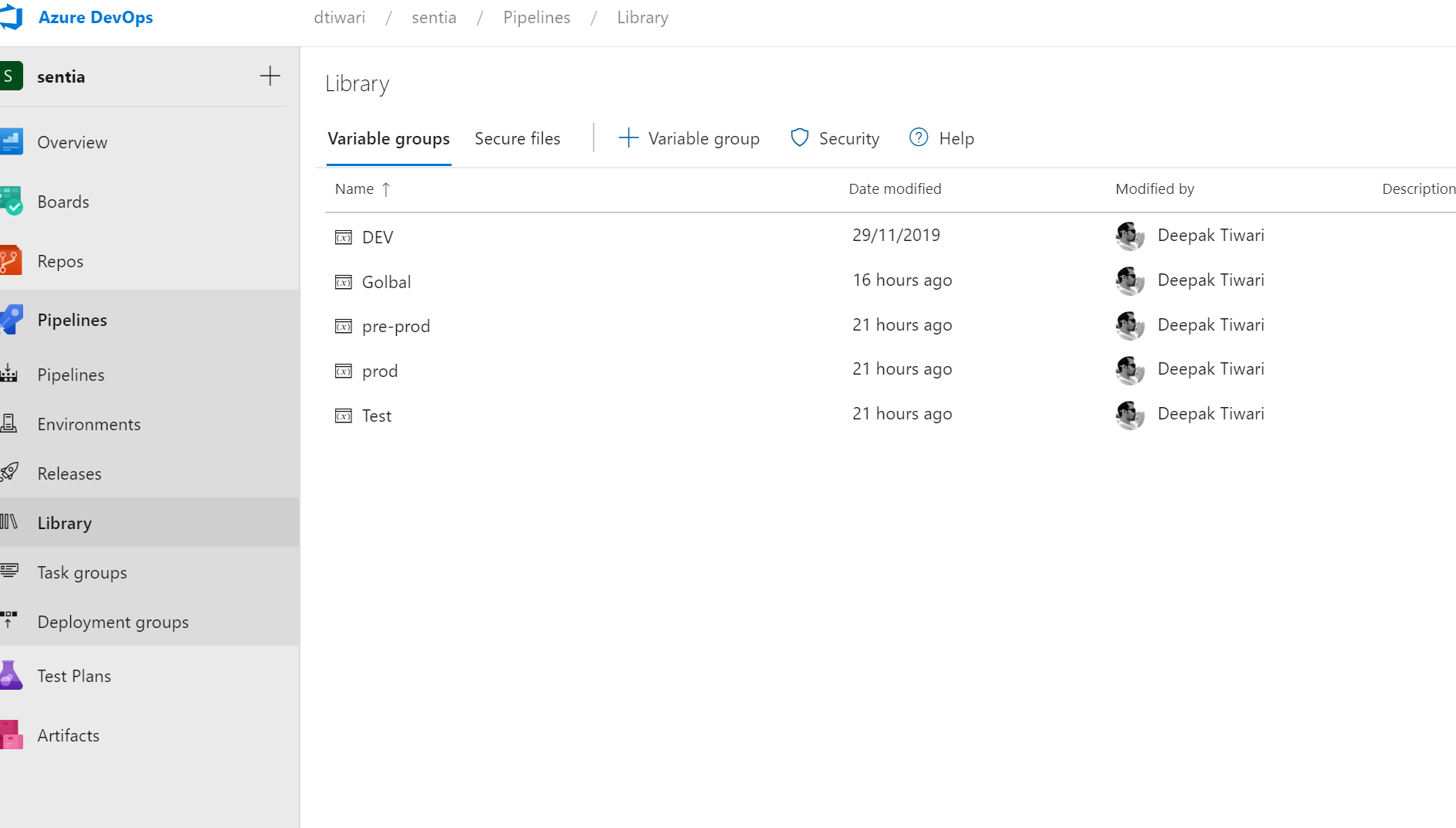
$(mysqlsourcedb) – DB name of Local environment

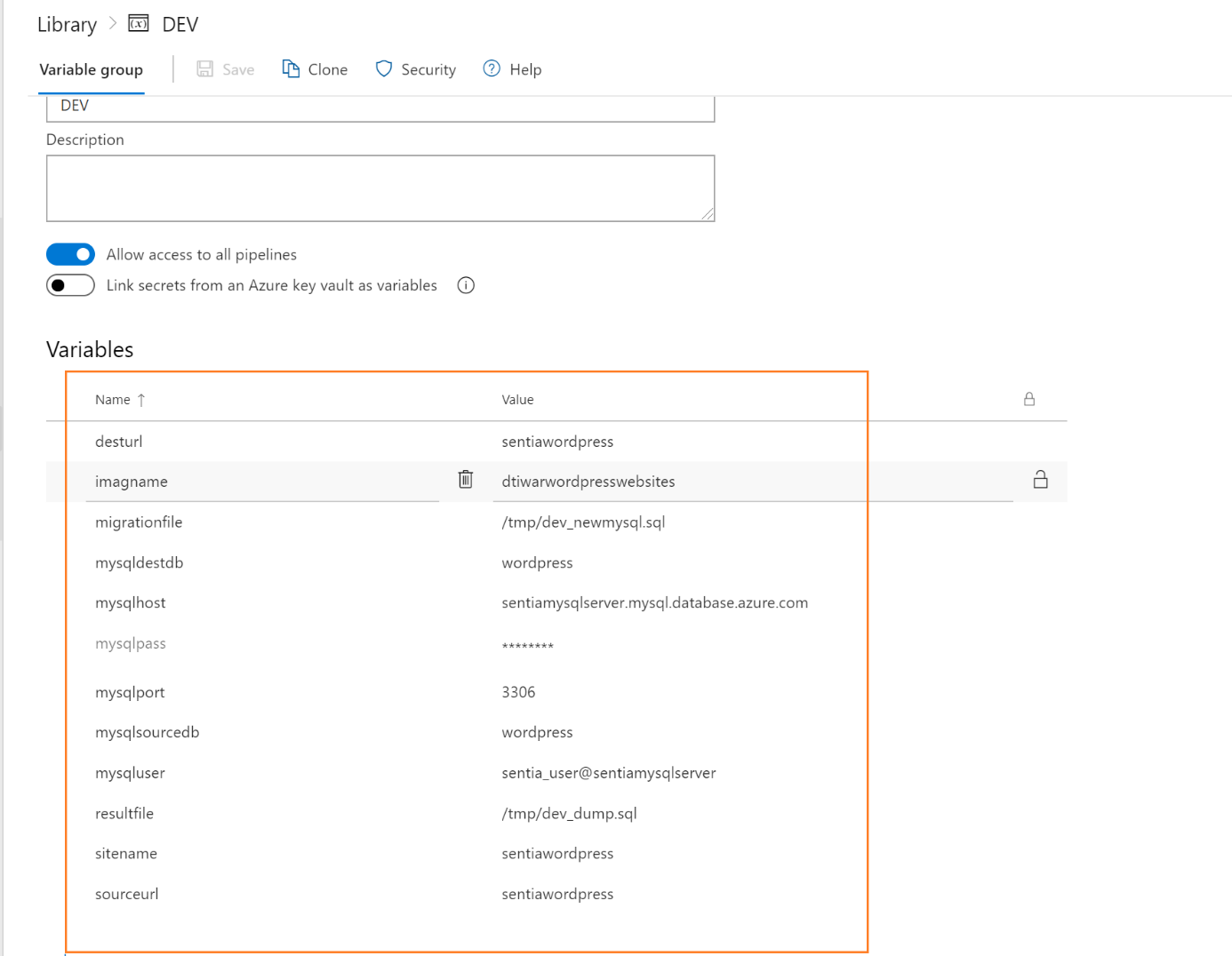
$(mysqluser) – User name for Azure Database for MySQL

$(resultfile) - temp file name when executing DB backup and restore operation

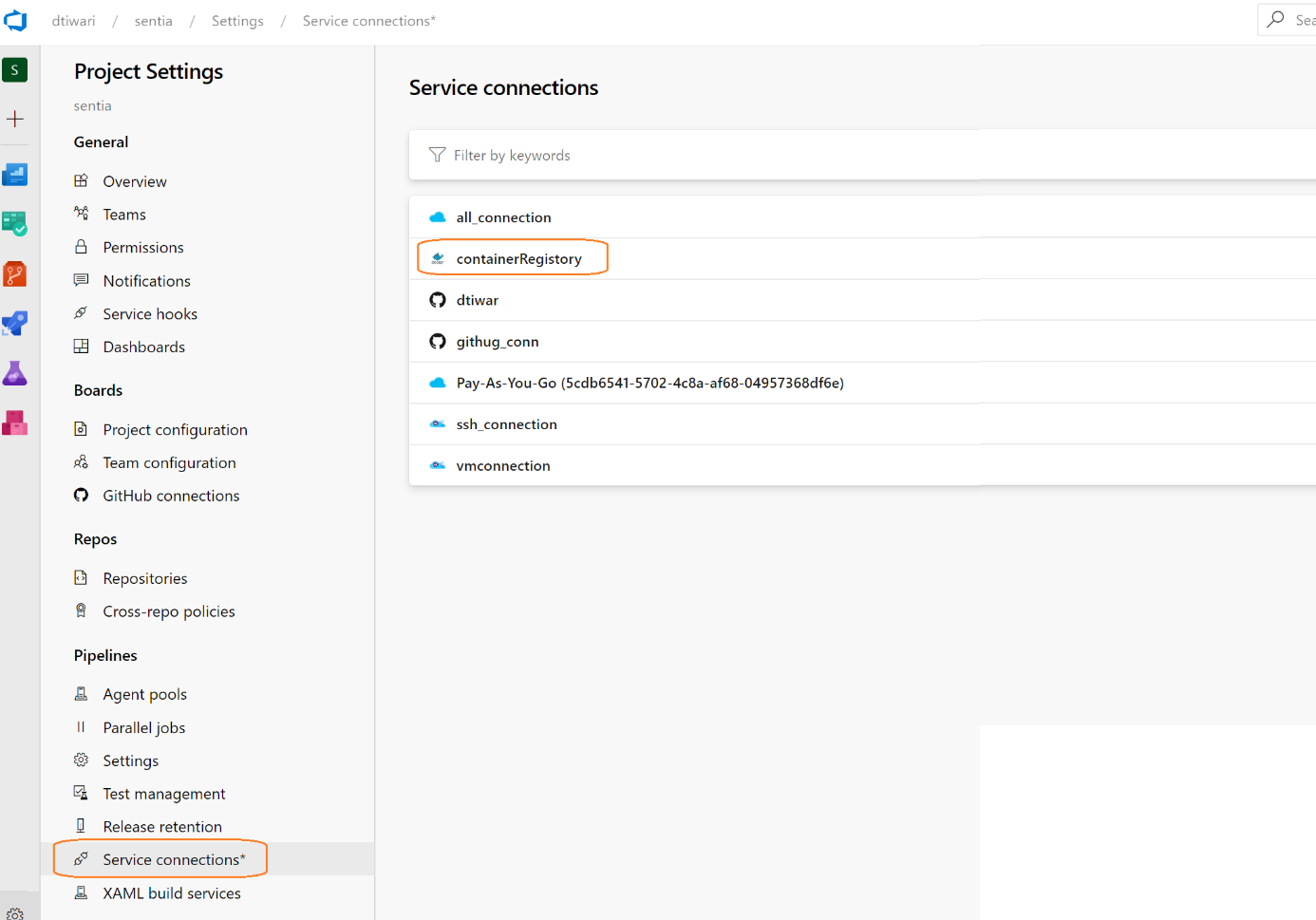
$(sourceurl) – Local environment URL

Sensitive value can be protected and not be visible until you have right role.

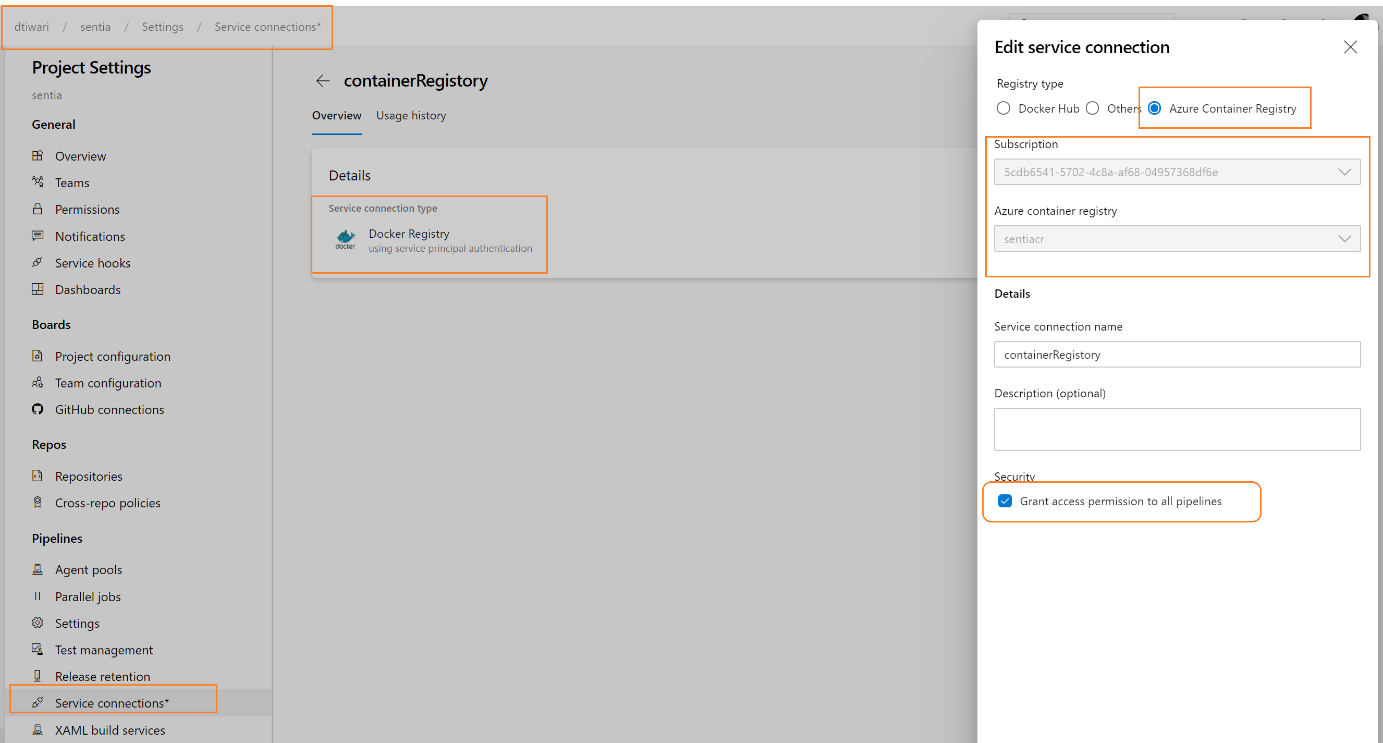




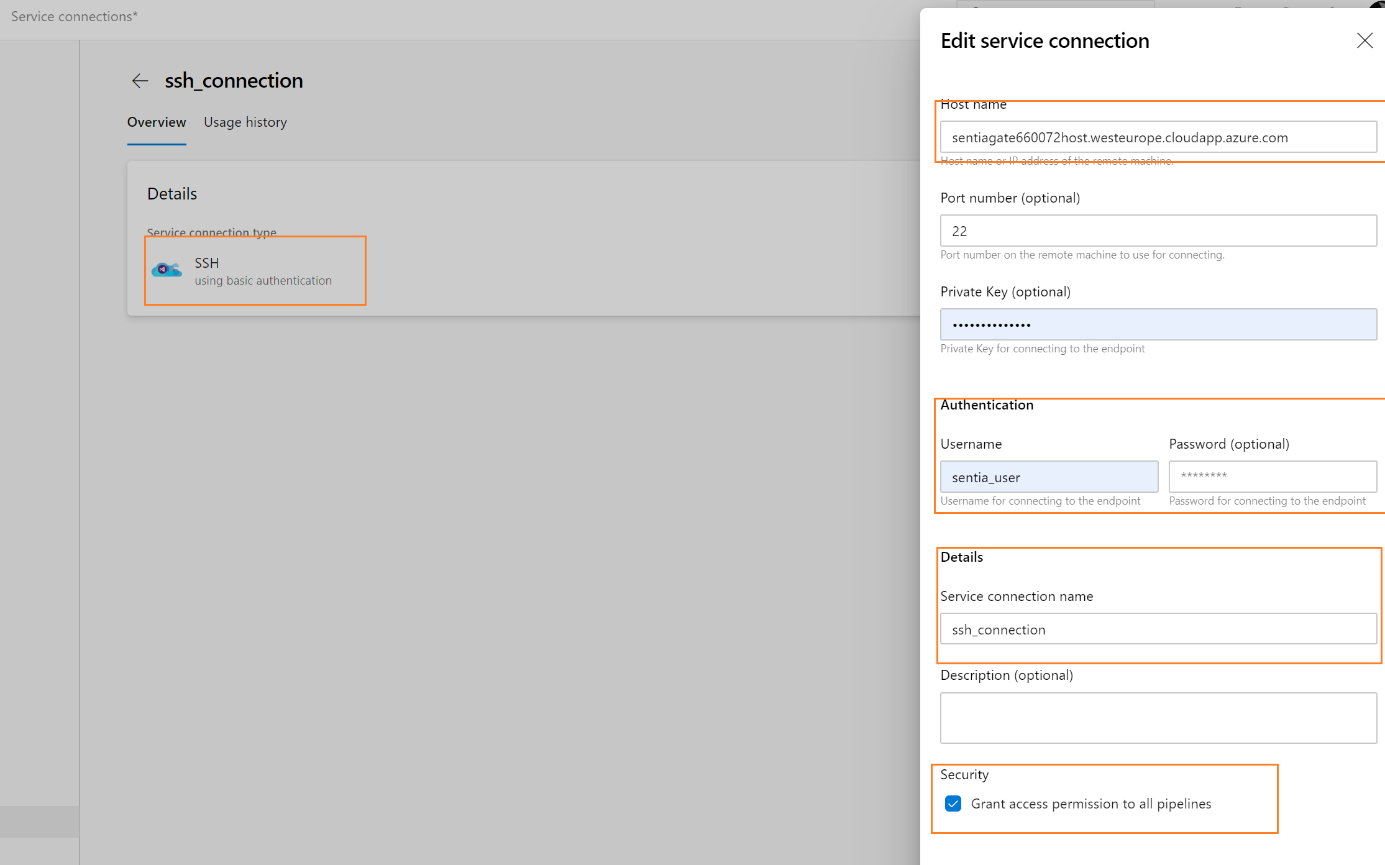
Service Connection:



These connection make the service connection with Azure Container repository and we have to Authorise the access permission to all the pipeline within this project by clicking the checkbox.



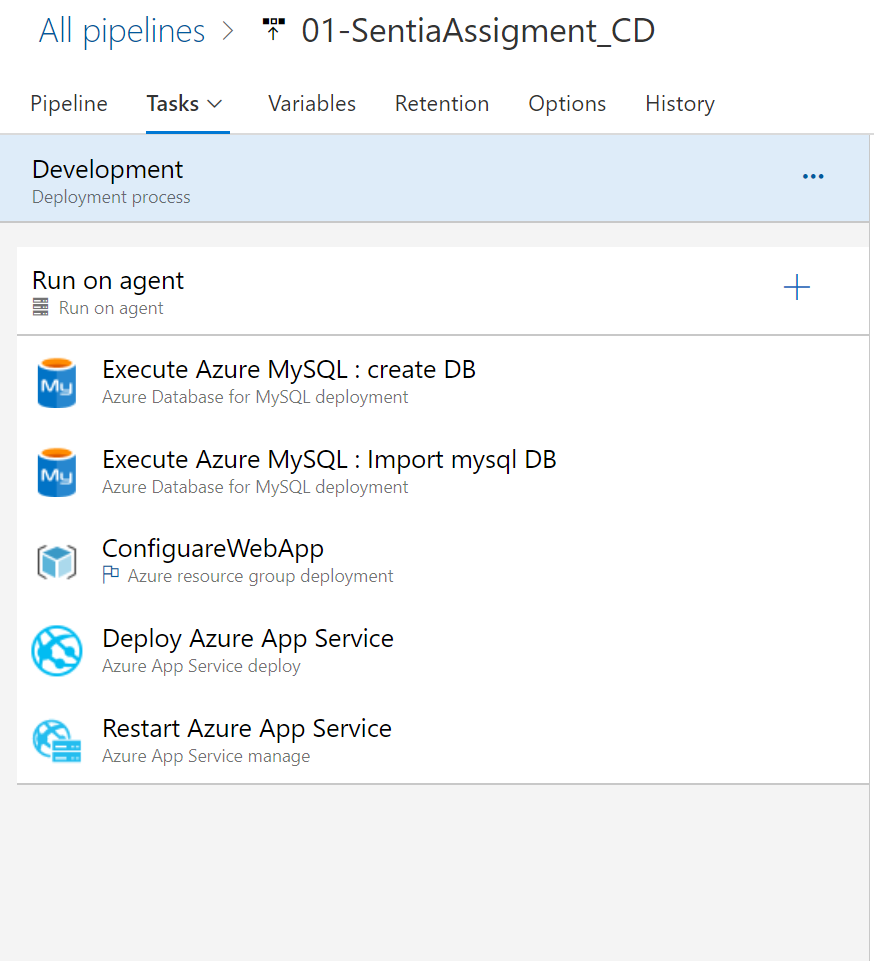
Before returning to the **tasks** tab, we need to add a new SSH endpoint (Settings/Services/New Service Endpoint/SSH). Fill your Azure Virtual Machine details.



#### Stages and Tasks :

The Pipeline is spliced into the four stages below.

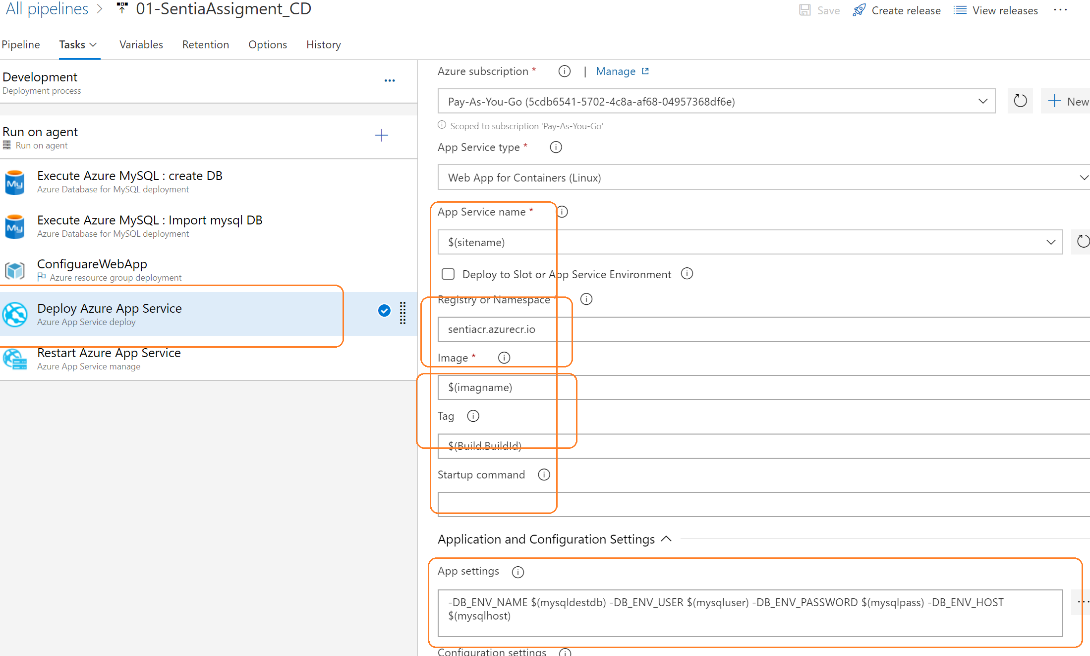
* Development:



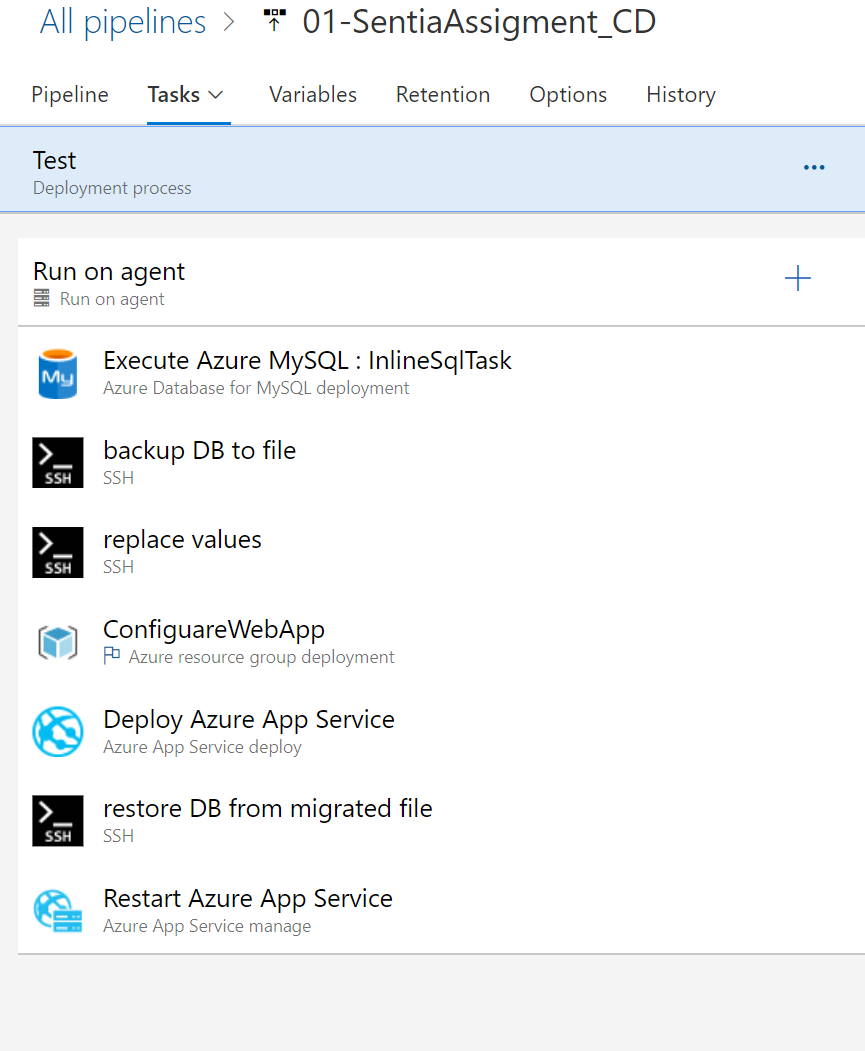
* Tasks:
  + - * **Execute Azure MySQL : InlineSqlTask**: Mysql DB task to create WordPress DB.
      * **Execute Azure MySQL : Import mysql DB:** export the .sql file from git repository
      * **ConfiguareWebApp:** create Webapp for container resource dynamically using ARM template
      * **(PS) –** there is a separate template for creating the Webapp for container with single template it will create 4 webapp but I am not using this here since I decided to create on fly.
      * **The other template link is below.**

<https://github.com/dtiwar/SentiaARMTemplate/tree/master/WebApps>

* + - * **Deploy Azure App Service: pull the docker image from container registery and deploy in the webapp for container resource**

****

* Restart Azure App Service: and finally, it will restart the Webapp application in order to take the image in effect.
* Test:



**Execute Azure MySQL : InlineSqlTask** : for creating the WordPress DB if not

exist.

**backup DB to file:** take the export from MYSQL Db and place into the .sql file . Where it can be referred later just in case of any issue during the deployment.

mysqldump -P $(mysqlport) -h $(mysqlhost) -u $(mysqluser) -p$(mysqlpass) $(mysqlsourcedb) > $(resultfile)

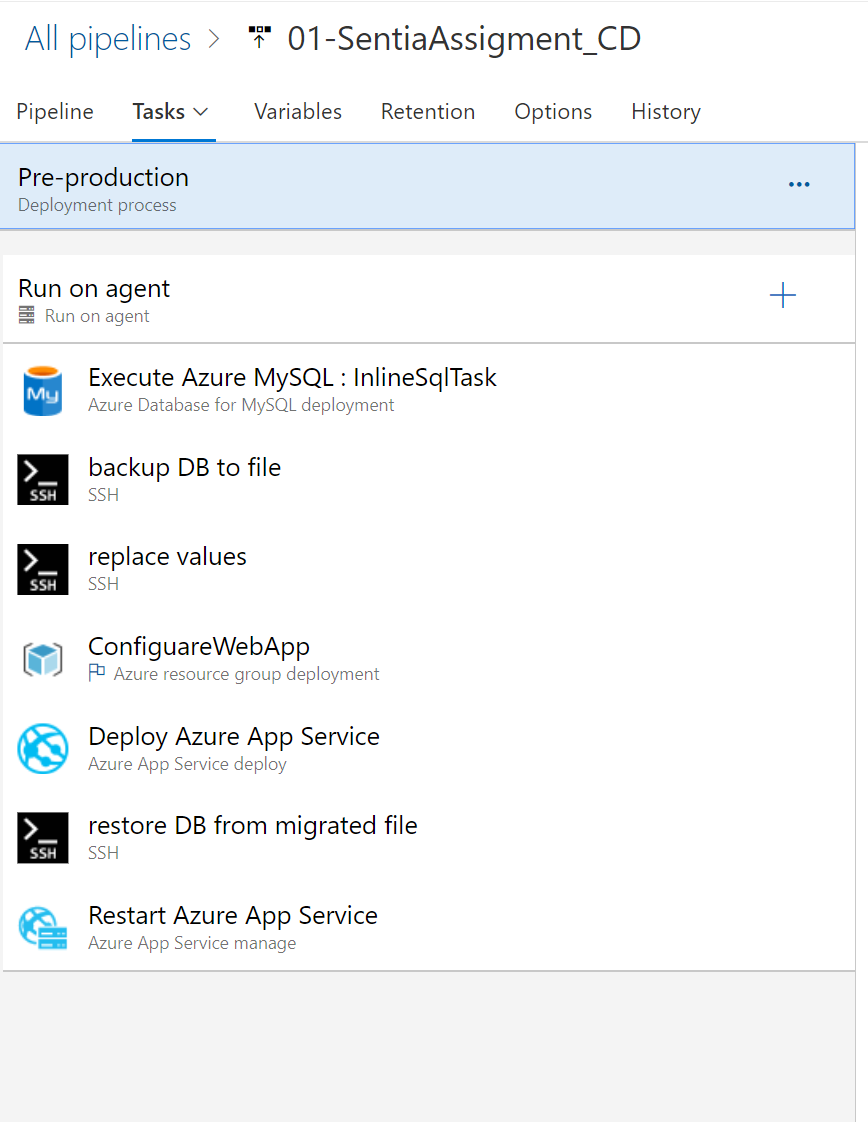
**Replace values:** since I am importing the database from DEV environment. This tasks to replace the website name from .sql file with the Test naming conversion and will create a new .sql file for imports.

sed 's/$(sourceurl)/$(desturl)/g' $(resultfile) > $(migrationfile)

**restore DB from migrated file**: import task into the database.

**ConfiguareWebApp and Deploy:** Tasks similer to the task mention in Development stage.

* Preproduction



All the Steps and same and self-explanatory for Prepoduction and Production as well.

## Triggering the WordPress Pipeline.

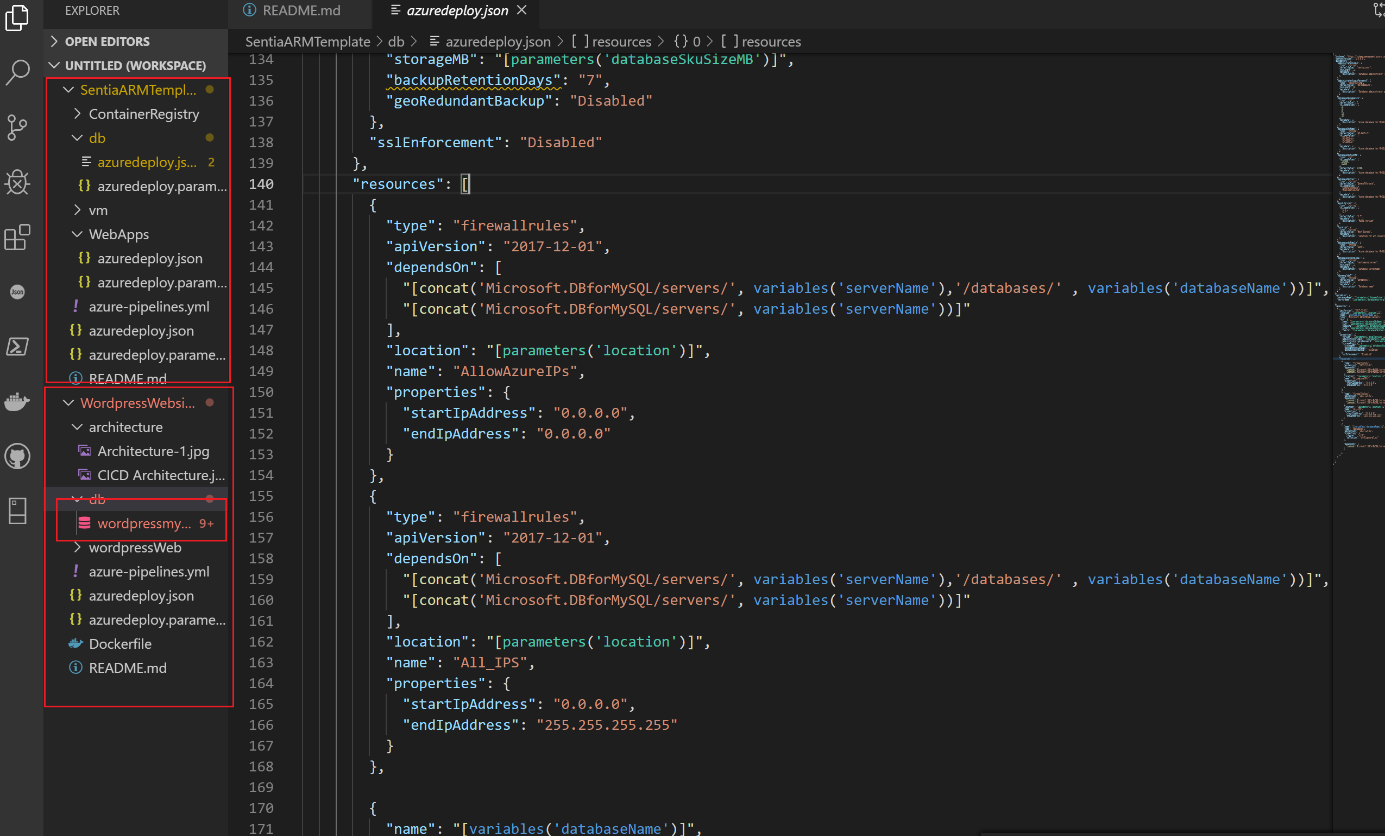
Now we have setup the process we can start the CI-CD Pipeline by simply modify the code in visual studio code -> commit and push the code into githib.

This action will trigger the 01 SentiaAssigment-CIpipeline in azure devops and start the build container process and push the container into the container registry.

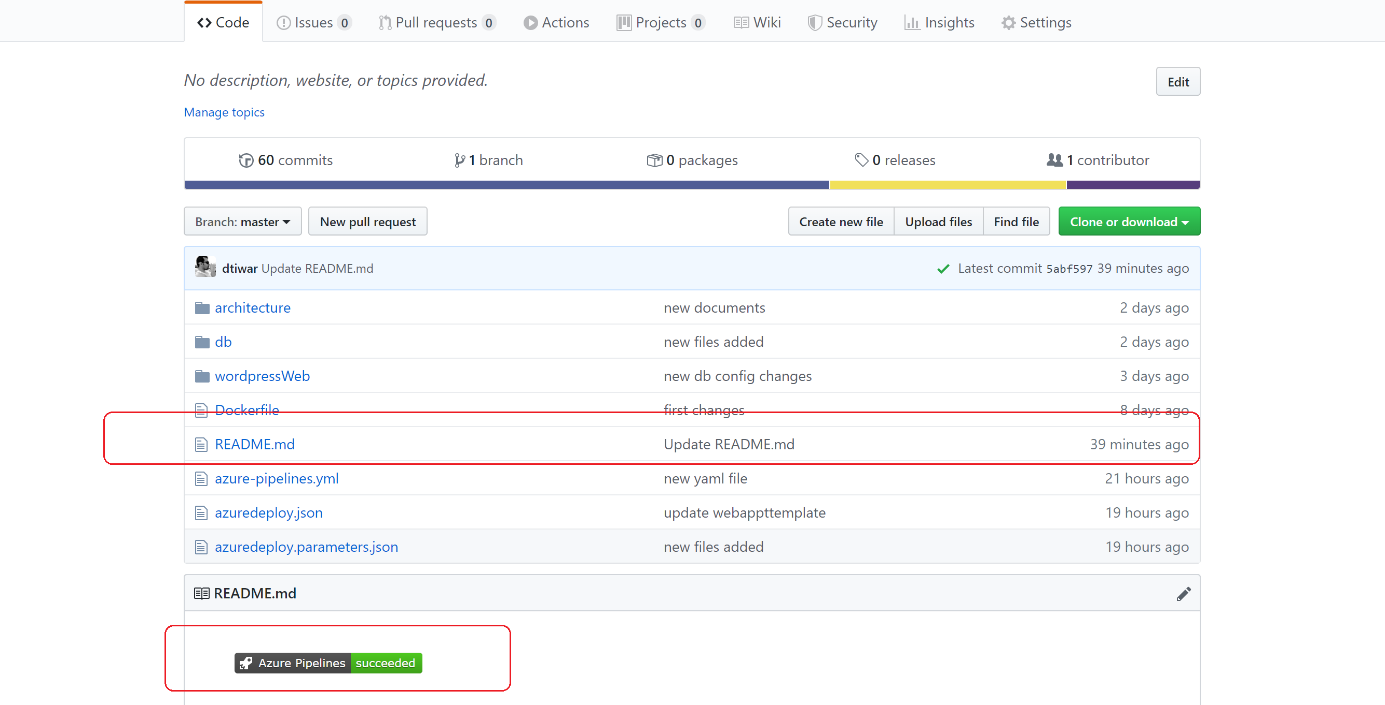
Once the Build process is complete the new build will trigger the release pipeline. And deploy the docker container in environmental wise web apps.

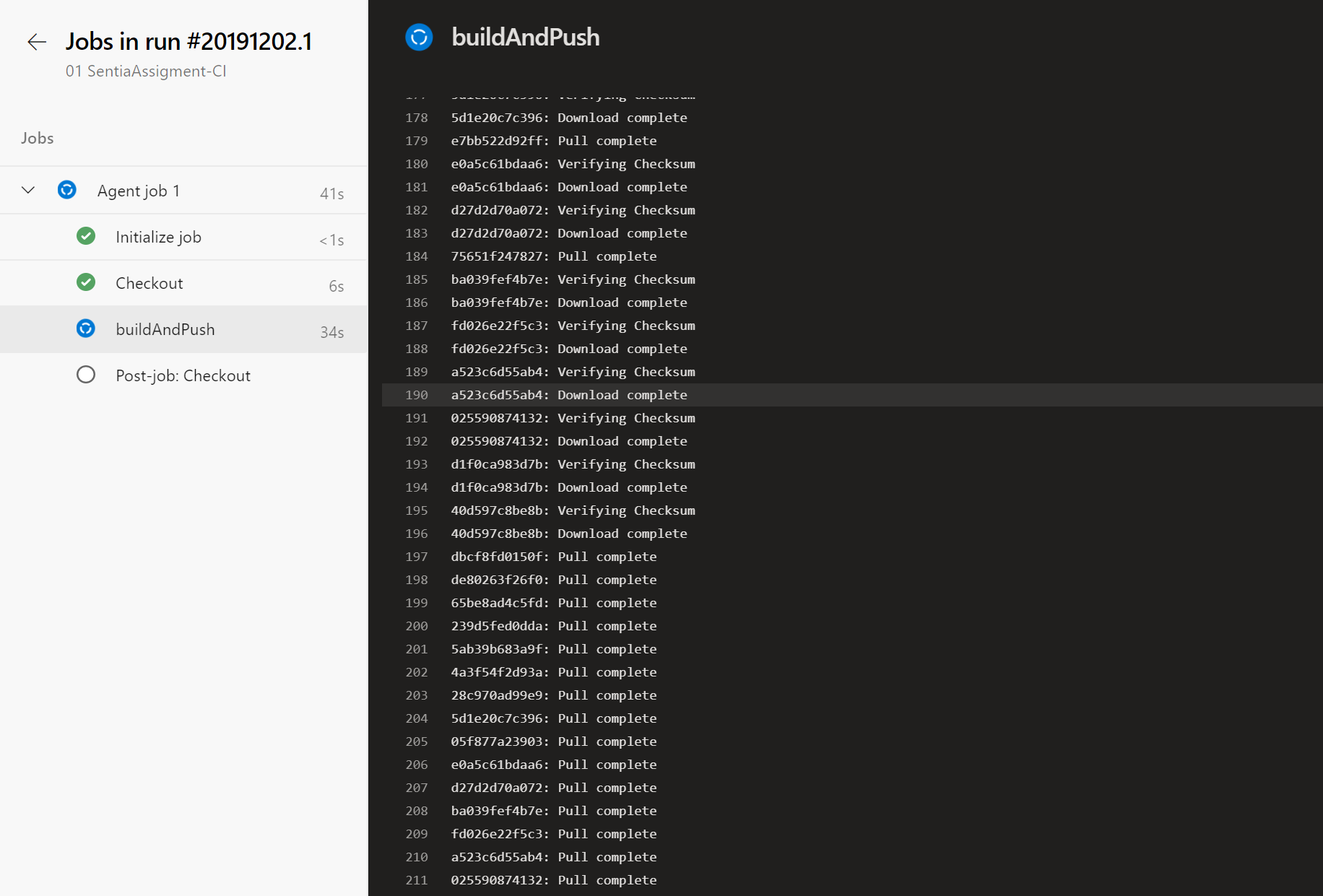
The below Screenshot show the steps by steps process for CI-CD process for WordPress project.

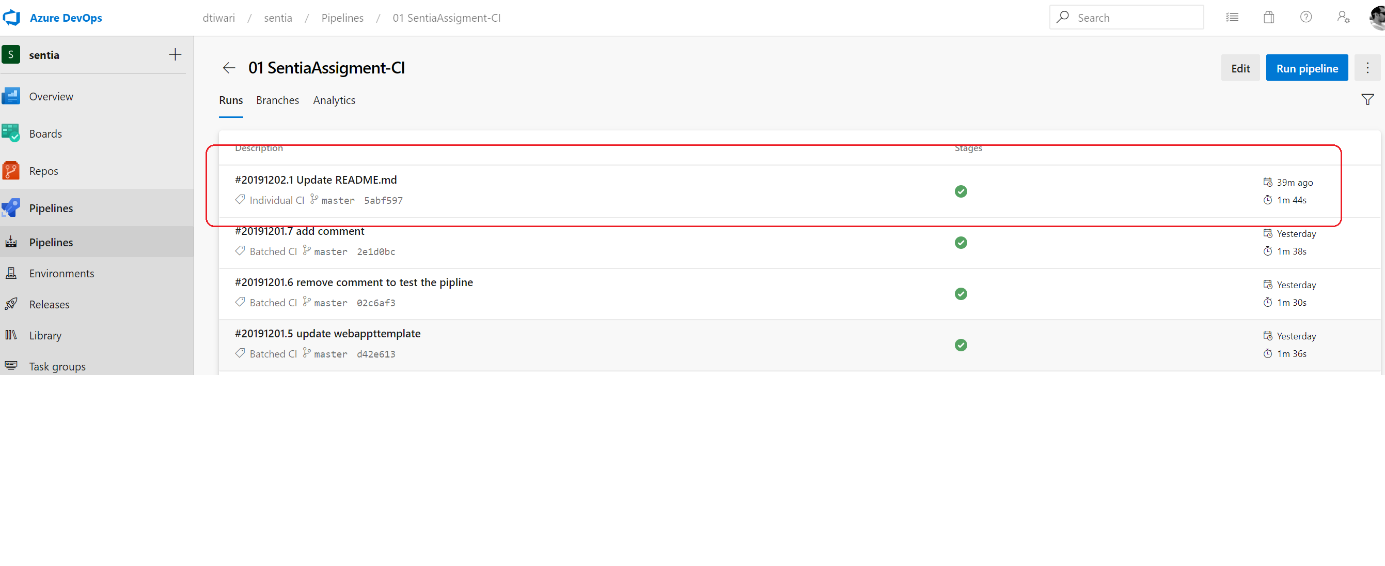
#### Visual studio code containing the artefact for development



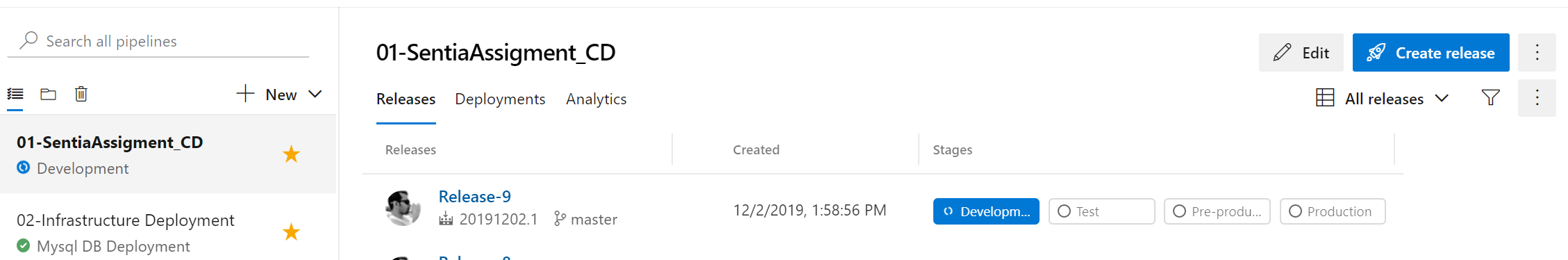
Make the changes on artefacts and push it to the GitHub. This will create the new version and will trigger the CI pipeline.



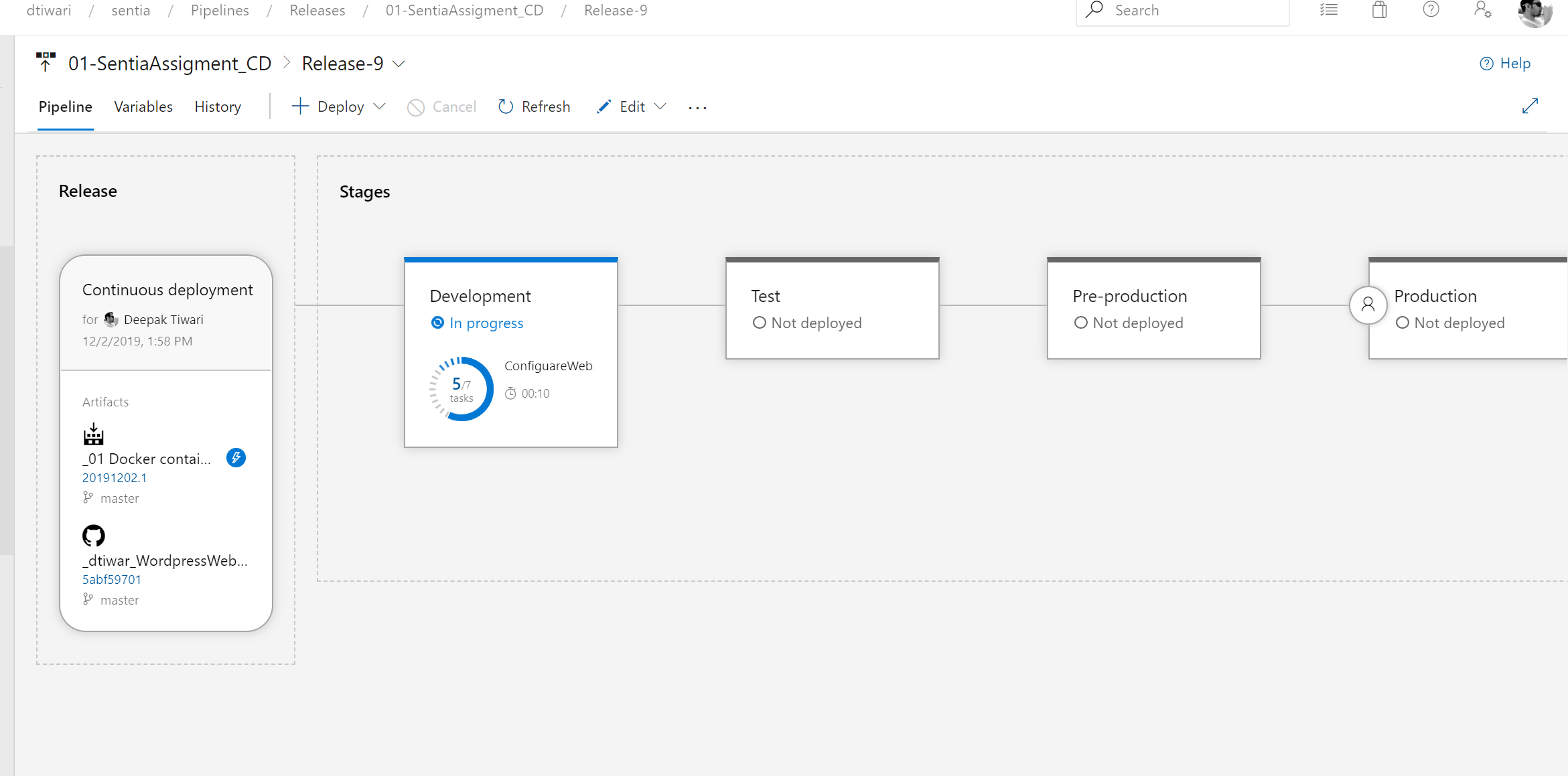




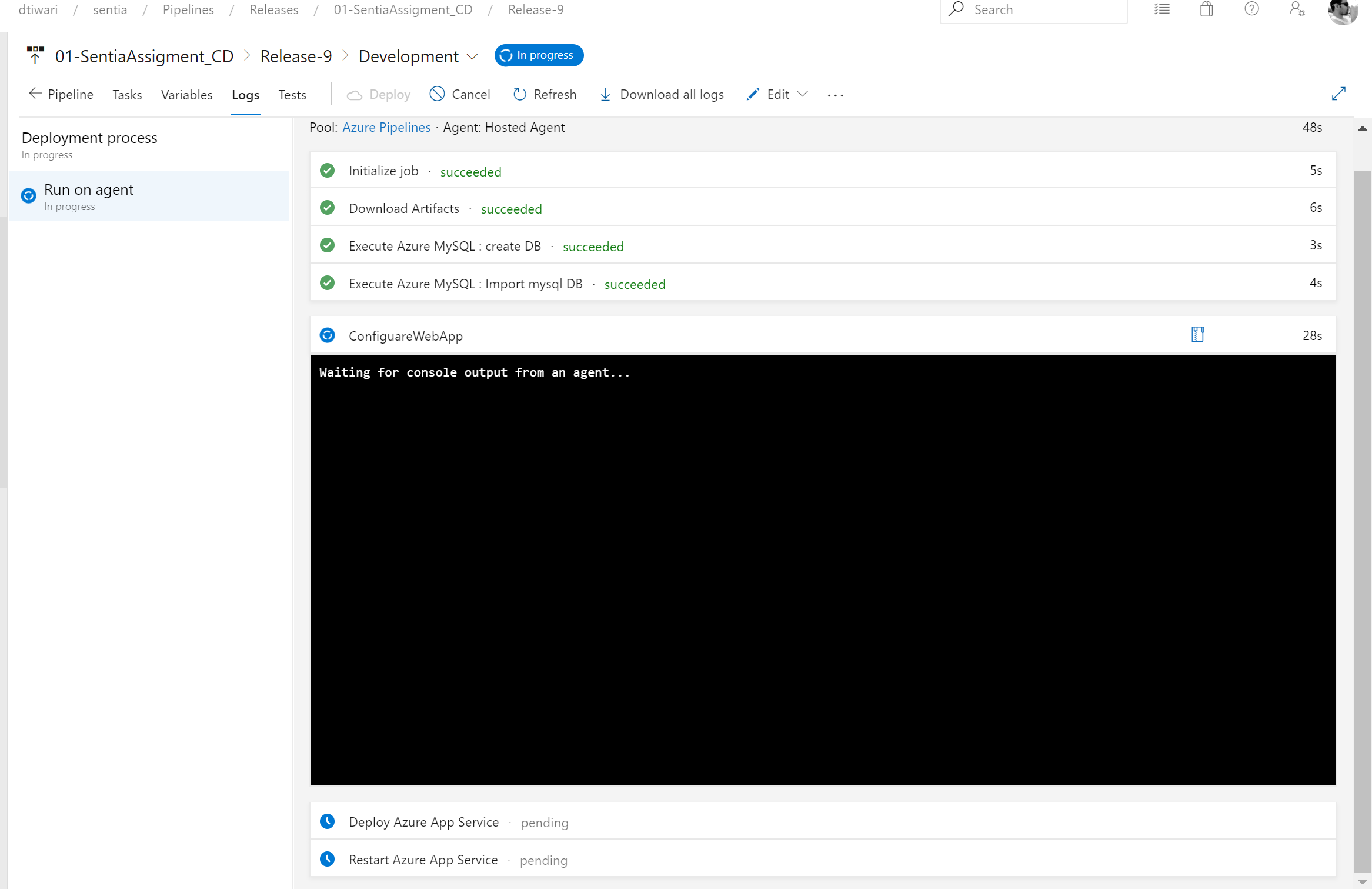
The above SS shows the Ci pipeline is completed now and its ready for trigger the CD pipeline.



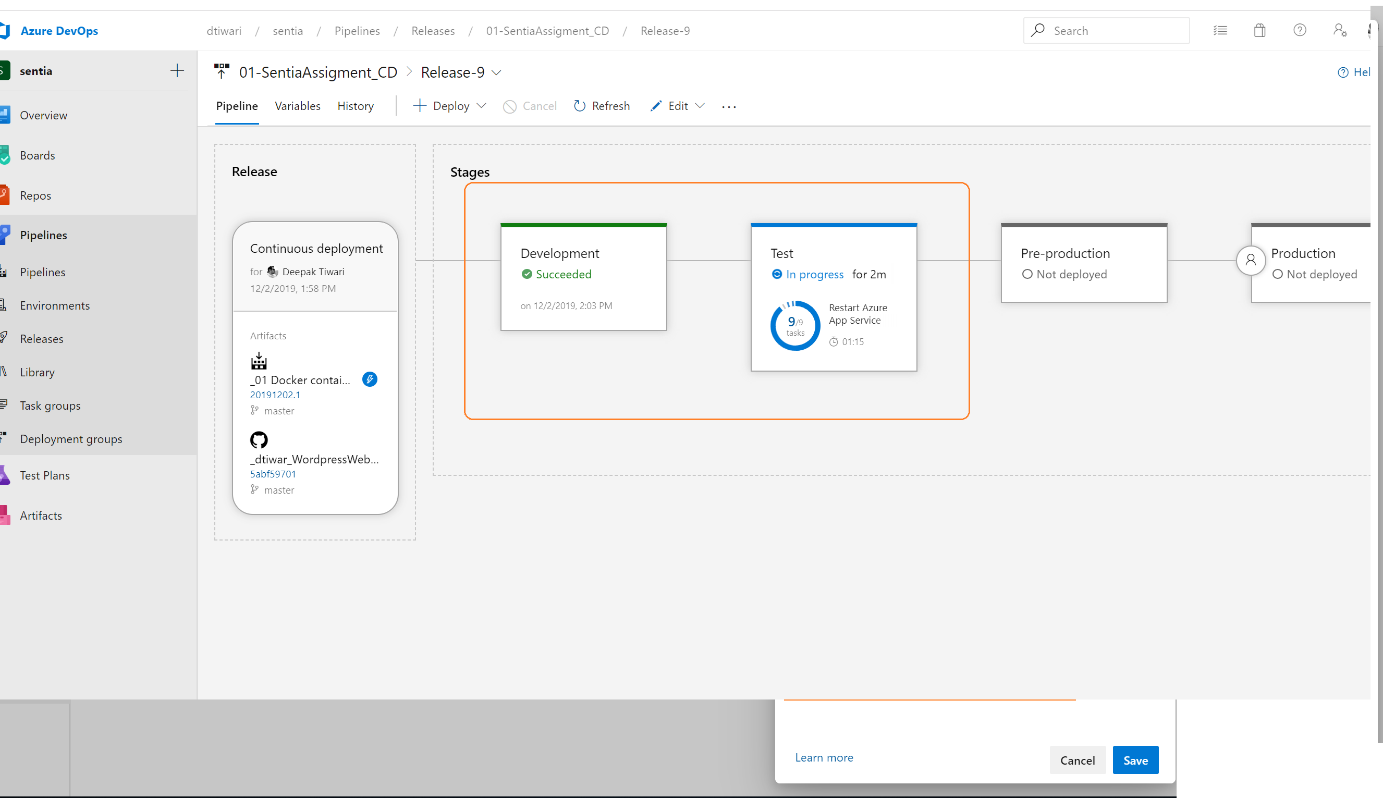
A new release pipeline has been trigger



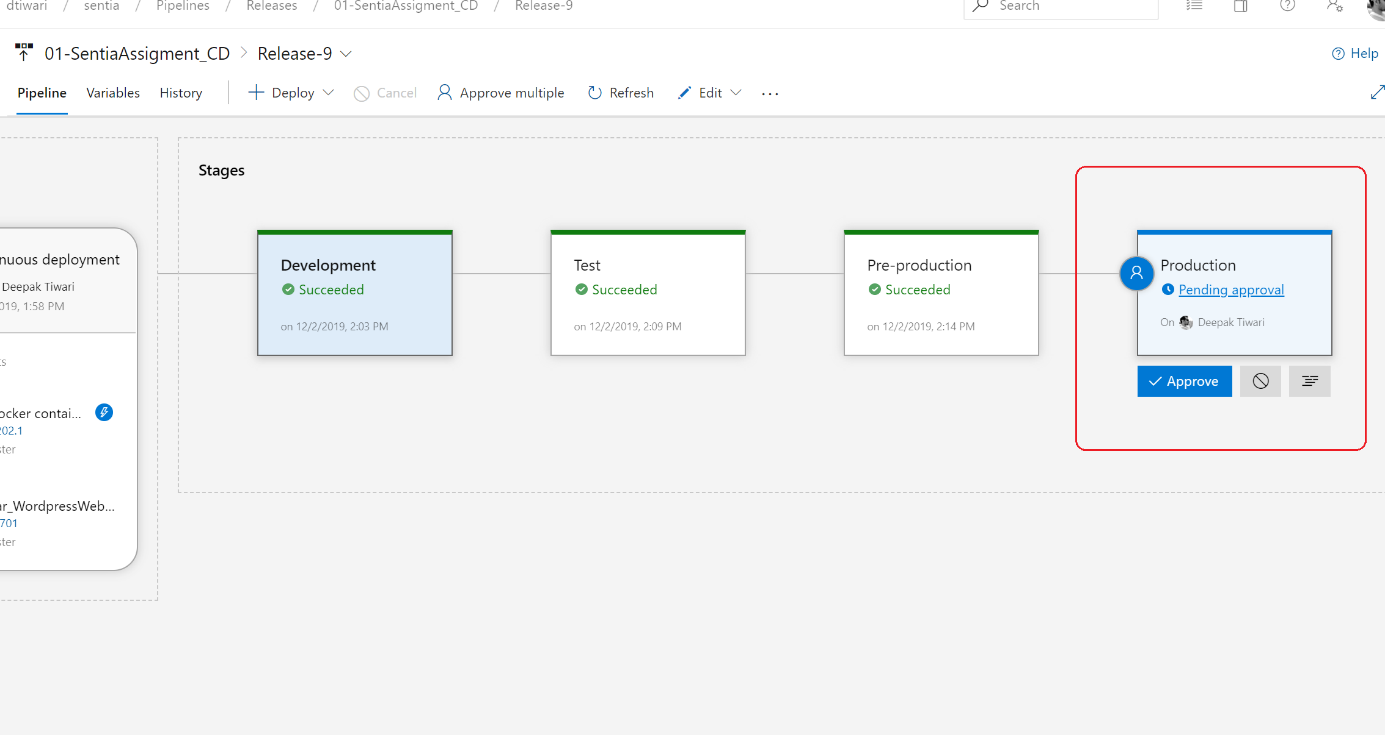
Triggering the All the tasks sequentially in the Development stage.



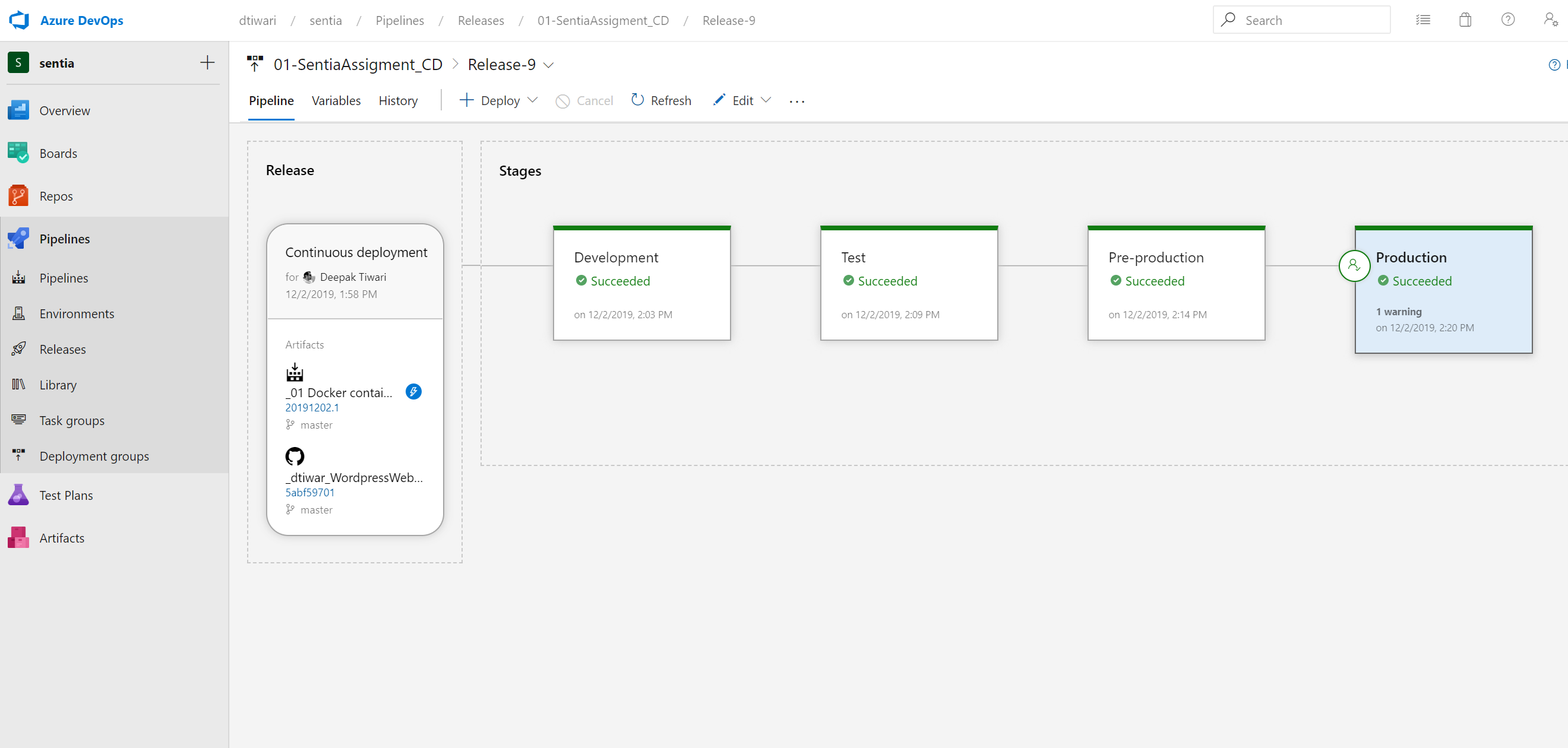
Below screenshot shows the Development stage has been completed and it triggered the Test Stage and start executing the Test Tasks.

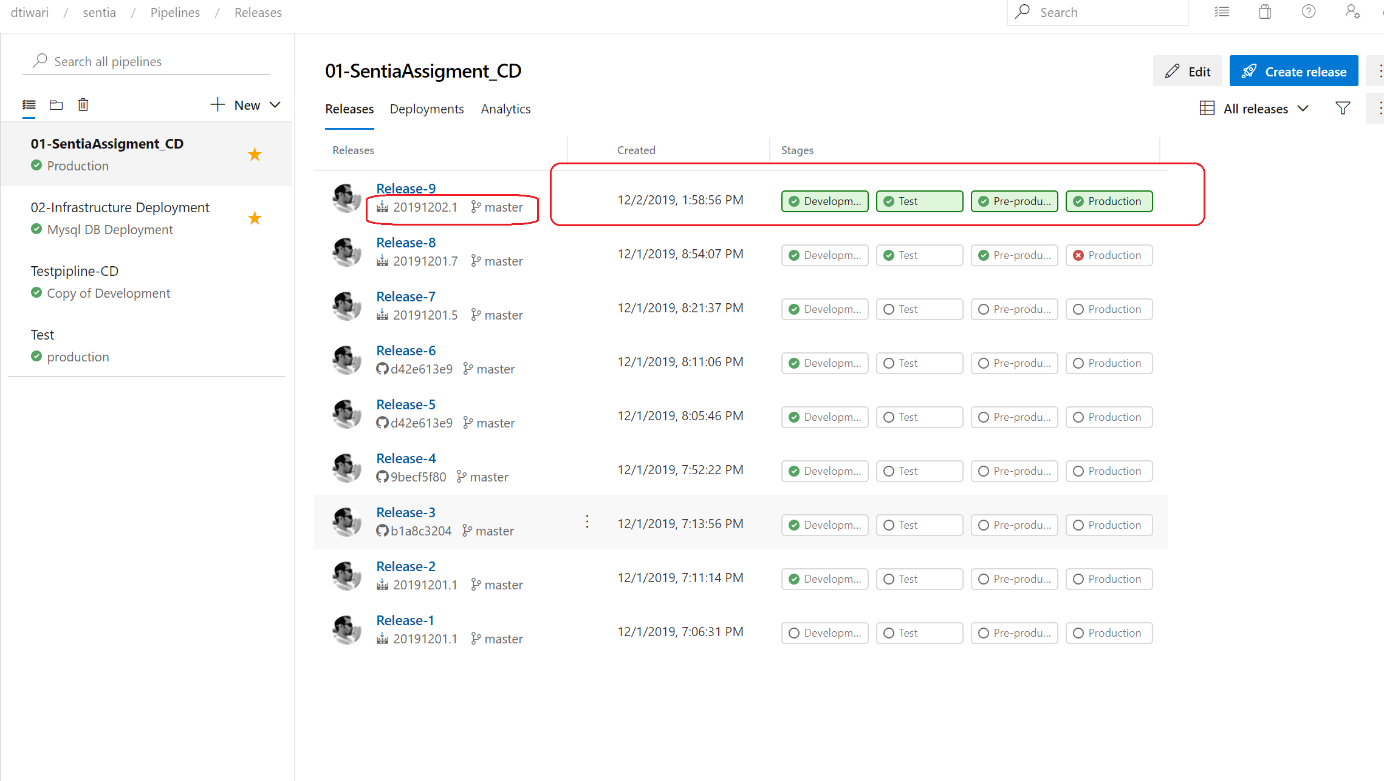


After completion of Preproduction stage the and before executing the Production stage we can configure the approval (This can be done for all the stages)

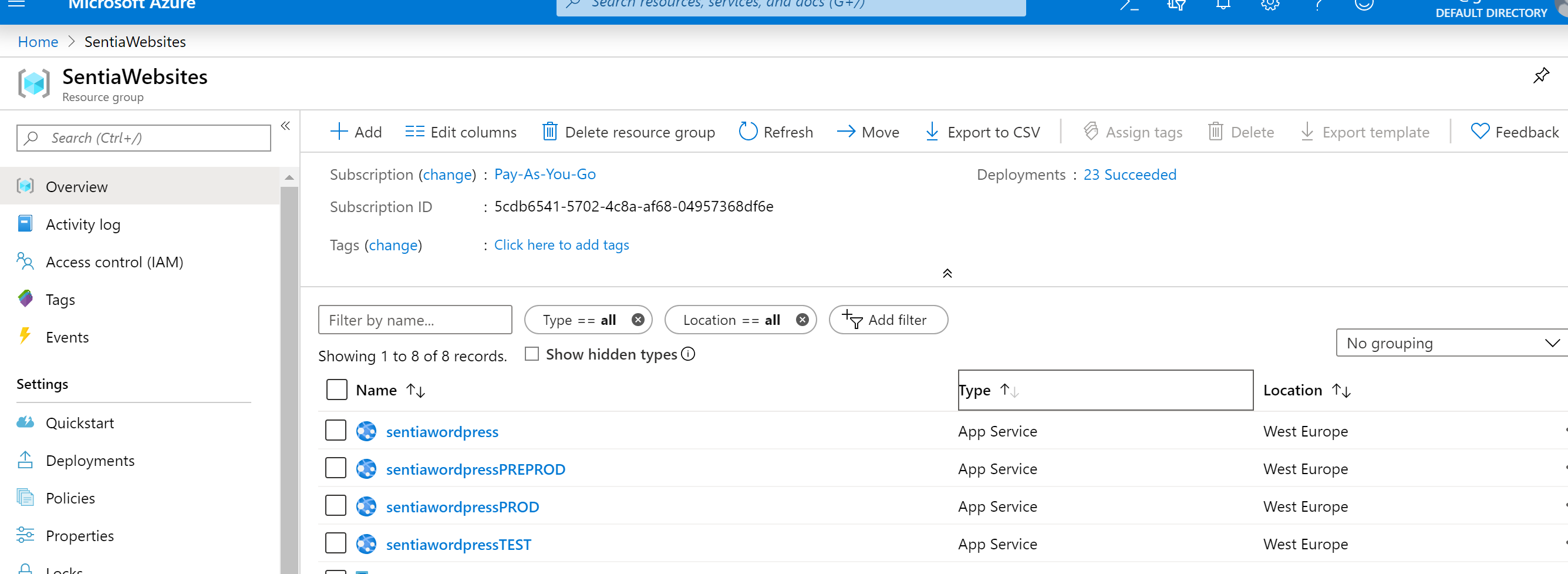


All Release Pipeline stages Completed successfully.





All WordPress websites are created:



Verifying:

