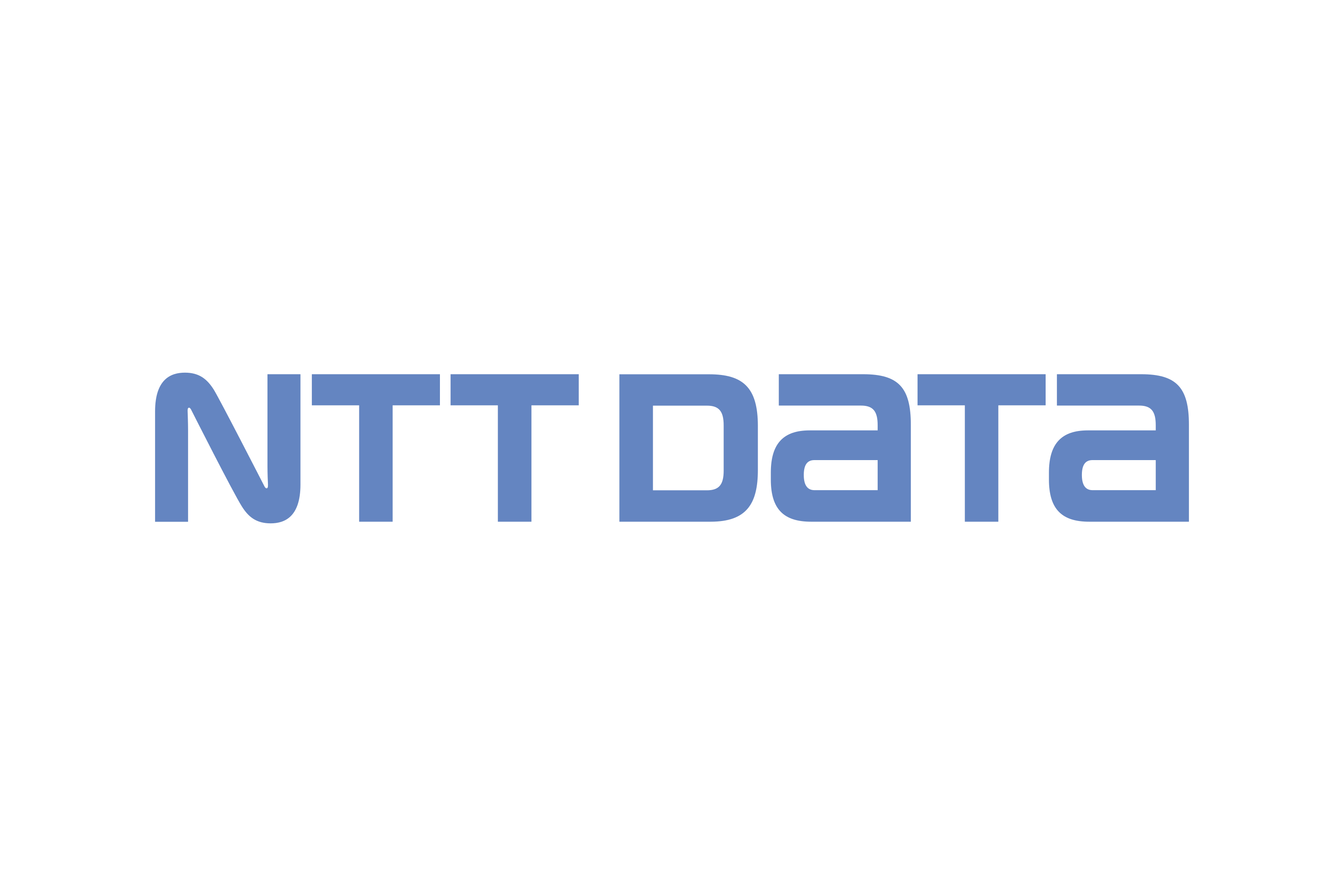
## 





NTT – DevOps

Configuration

Revision History

|  |  |  |
| --- | --- | --- |
| Version | Author(s) | Summary of changes |
| 1 | MOHD GHOUSE | Introduction |
| 2 | MOHD GHOUSE | Environment |
| 3 | MOHD GHOUSE | Continuous Integration & Delivery |
| 3.1 | MOHD GHOUSE | Agent installation |
| 3.2 | MOHD GHOUSE | Pipeline |
| 3.3 | MOHD GHOUSE | Dev Pipeline |
| 3.4 | MOHD GHOUSE | QA-UAT Pipeline |
| 3.5 | MOHD GHOUSE | PROD Pipeline |

**Table of Contents**

[1 Introduction](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082607)  4

[2 Environment](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082608) 5

[2.1 Dev](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082609) 6

[2.2 QA / UAT](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082610) 6

[2.3 PROD](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082611) 7

2.4 Environment Variable…………………………………………………………………………………………………7

2.5 Adding ENV………………………………………………………………………………………………………………….7

[3 Continuous Integration / Continous Delivery](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082642) 8

[3.1 Agent installation](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082643) 8

[3.2 Pipelines](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082644) 9

[3.3 DEV](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082645) 10

[3.3.1 CI/CD Flow](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082646) 11

[3.3.2 Trigger a deploy](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082647) 11

[3.3.3 Deployment steps](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082648) 12

[3.4 QA-UAT](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082649) 12

[3.4.1 Trigger a deploy](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082650) 13

[3.4.2 Deployment steps](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082651) 14

[3.5 PROD](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082652) 20

3.5.1 Adding Approval names………………………………………………………………..………………22

[3.5.2 Trigger a deploy](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082653) 23

[3.5.3 Deployment steps](file:///C:\Users\Mghouse\Downloads\NTT_DevOps_Configuration%20(2).docx#_Toc89082654) 24

4 Troubleshoot steep…………………………………………………………………………..……………………….25

# Introduction

This document has the intent to describe the DevOps processes performed at the NTT Data environment, and provide an explanation about the environments involved throught the process and the technologies utilized.

# Environment

Currently, NTT Data has three environments: DEV, QA/UAT and PROD, and for a better understand the diagram flows are separated per environment as follows.

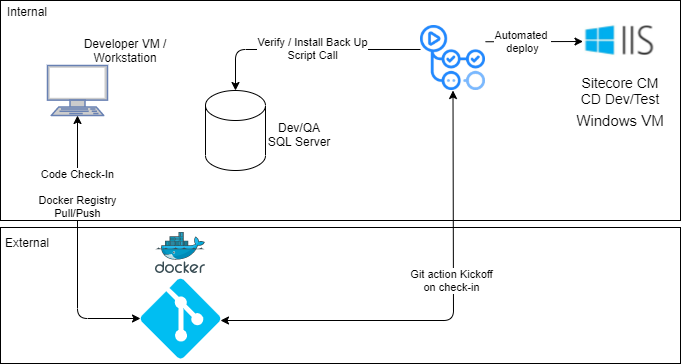
|  |
| --- |
| DEVELOP |
| Uat-Sxa |
| Prod-Sxa |

These are the three branches name at github using for environments: DEV, QA/UAT and PROD.

## 2.1 Develop

The DEV environment consists of developer VM/workstation where Developers will have their code, an check-in to GitHub which kicks off a deployment to a Sitecore CM and CD provisioned in a Windows VM through Git Actions.

Also, developers will have Sitecore Docker Images to build their own local environments in a faster pace by pulling the images from GitHub repository.



## 2.2 Uat-Sxa

The QA/UAT environment is utilized to perform tests and validate the deployed code before going to PROD. Once developers check-in their code, Git Actions will be able to perform an Automated Test Call, as well as the deployment to the Sitecore Windows VMs that host CM and CDs.

## 2.3Prod-Sxa

The PROD environment is where the live website and CM exists, and the deployments are going to be manually triggered during maintenances windows (TBD) and when previous successful deployment happened into QA/UAT environment.

## Environment Variables

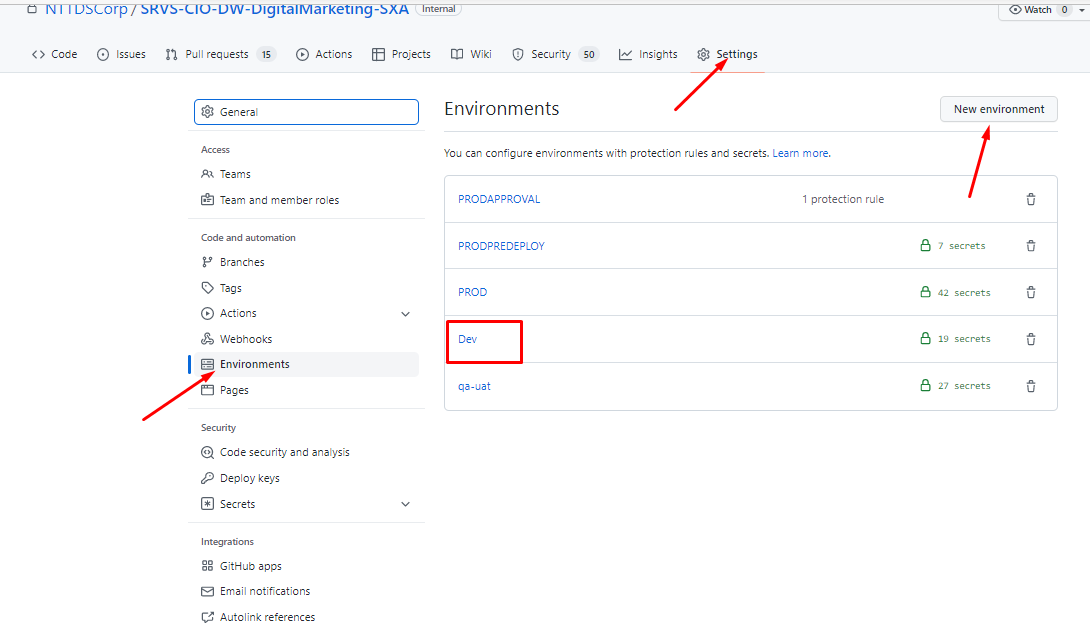
GitHub environment variables for each GitHub Actions workflow run. These environment variables use to store information that you want to reference in your workflow where can set our own custom environment variables. We have setup variable for all there environments.

You have made some modifications to these values when following the Environment variables section in this document.

## 2.5 Adding Environment variable in Github

Add the environment variable according to the branches like DEV, QA and PROD.

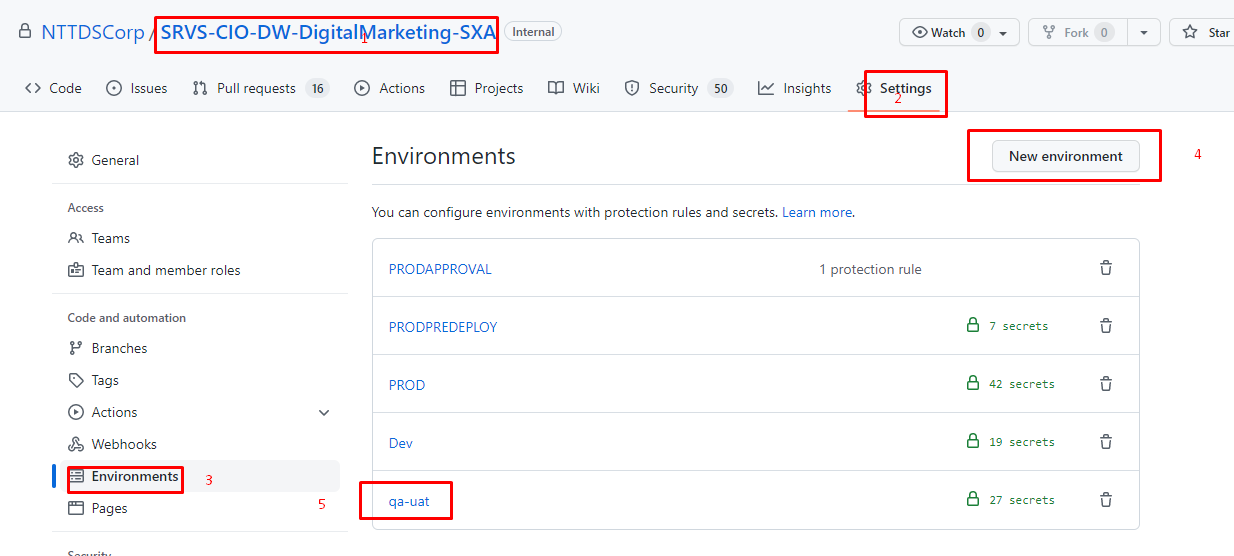
To add an environment variable, you should go to the repository which will have the GitHub Actions and click on **Settings** (1)**Environment** (2) New Environment



# **DEV ENVIRONMENT**

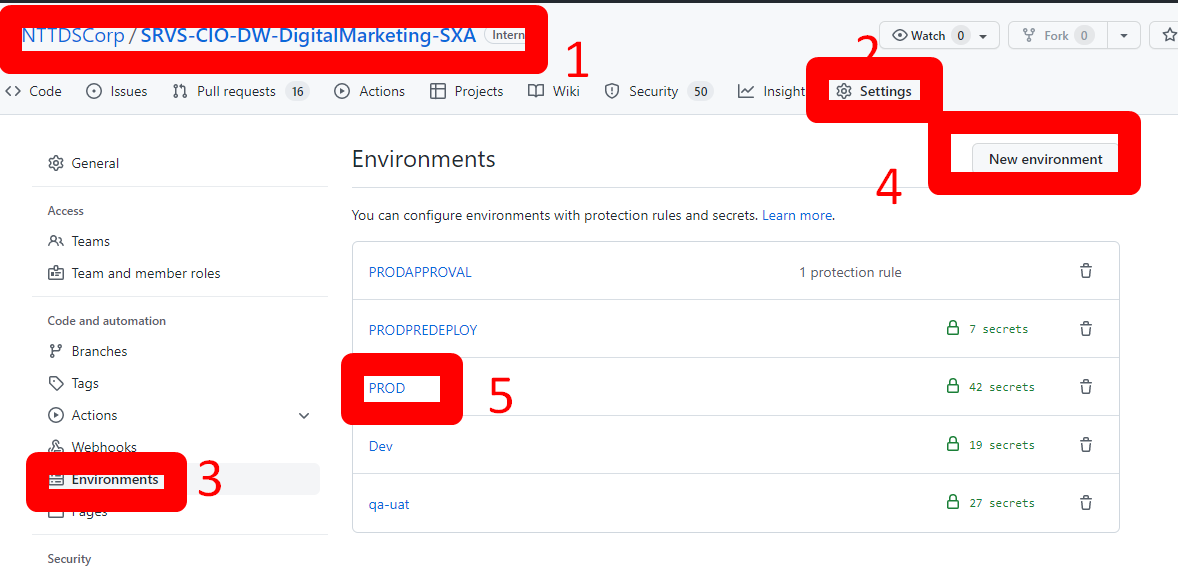
# 

# **QA ENVIRONMENT**





# **PROD ENVIRONMENT**



# 

# 3.0 Continuos Integration / Continous Delivery

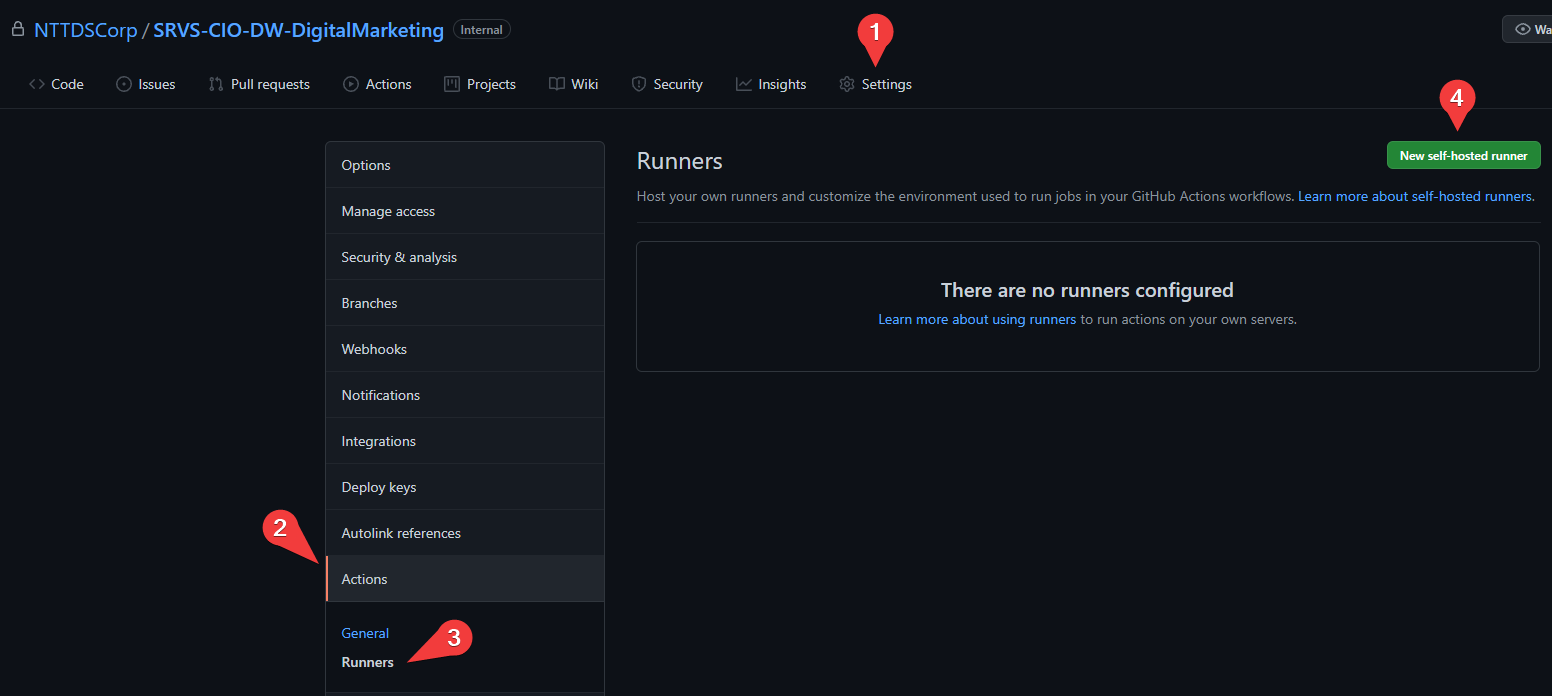
For the Continuous Integration / Continuous Delivery, GitHub Actions has been chosen and it is going to be executed through an agent, named Runner, in NTT network to build and deploy to DEV, QA/UAT, and PROD environments.

## 3.1 Agent installation

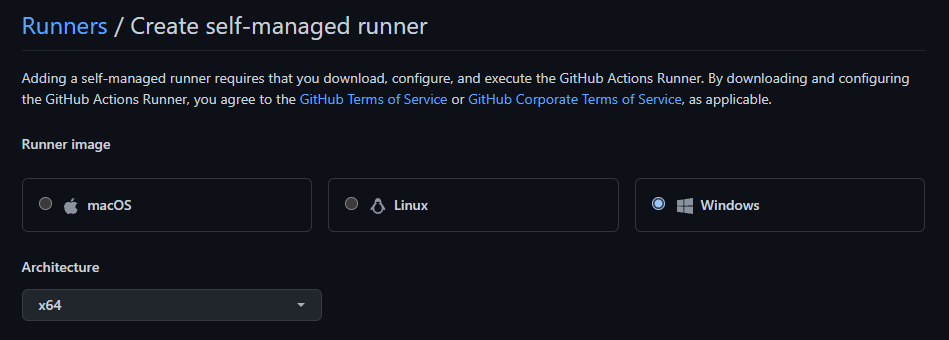
Install the three agent pool for three different environment like DEV,QA and PROD

|  |  |
| --- | --- |
| DEV | [**Dev-WTCRSCCICDDO1**](https://github.com/NTTDSCorp/SRVS-CIO-DW-DigitalMarketing-SXA/settings/actions/runners/26) |
| QA | [**UAT-WTCRSCCICDD01**](https://github.com/NTTDSCorp/SRVS-CIO-DW-DigitalMarketing-SXA/settings/actions/runners/27) |
| PROD | [**prod-WTCRSCCICDD01**](https://github.com/NTTDSCorp/SRVS-CIO-DW-DigitalMarketing-SXA/settings/actions/runners/28) |

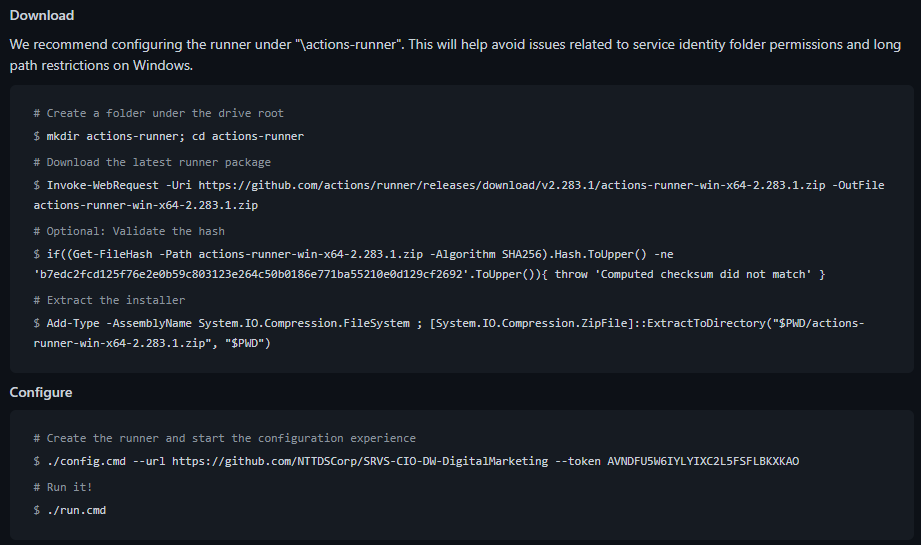
To install the agent, you should go to the repository which will have the GitHub Actions and click on **Settings** (1), **Actions** (2), **Runners** (3), and **New self-hosted runner** (4).



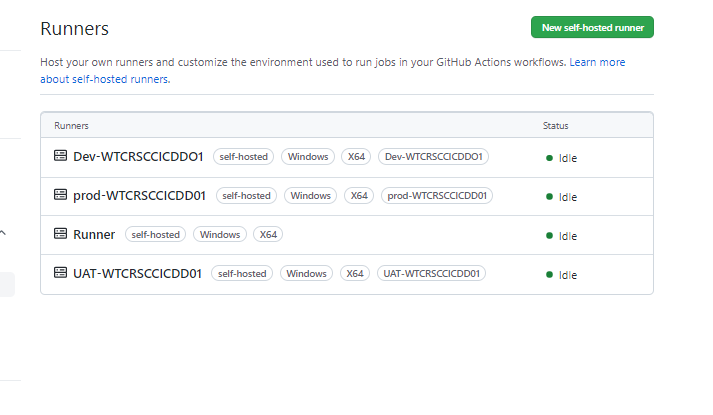
Then, you will have to choose the **Runner image** (e.g. Windows), and the **Architecture** (e.g. x64).



Once you select the options, you will be provided with the steps to execute in the server you want to install the **runner agent**.



Once all three agent has been install it look like below image

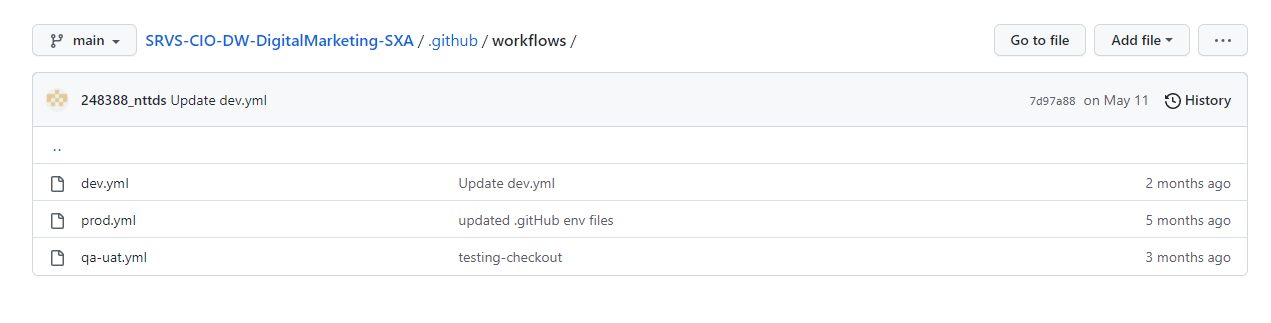


## 

## 3.2 Pipelines

The pipelines are separated accordingly to the environment they are going to build and deploy, so there are going to exist 3 pipelines: DEV, UAT, and PROD.

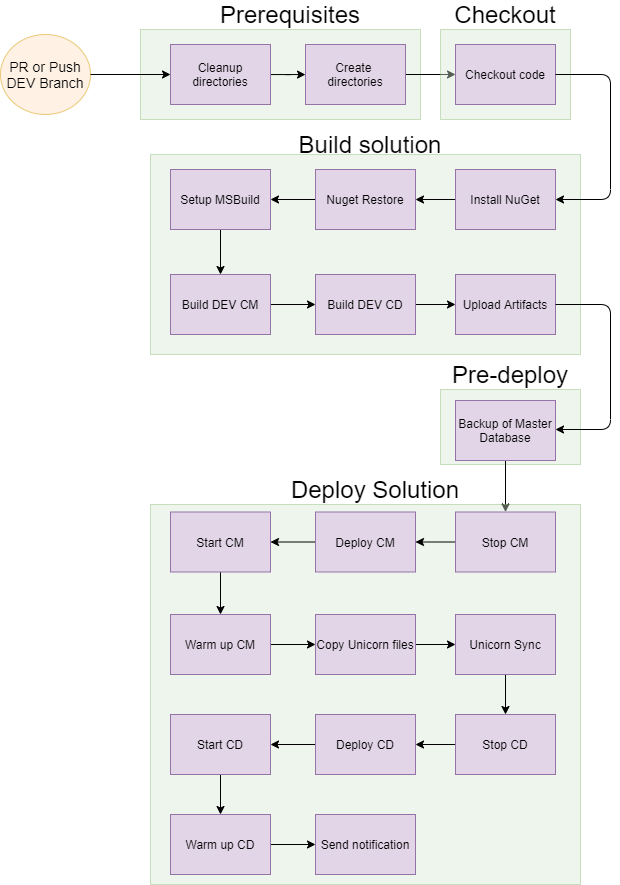
The configuration can be found in the repository level, under **.github/workflows**



## 3.3DEV

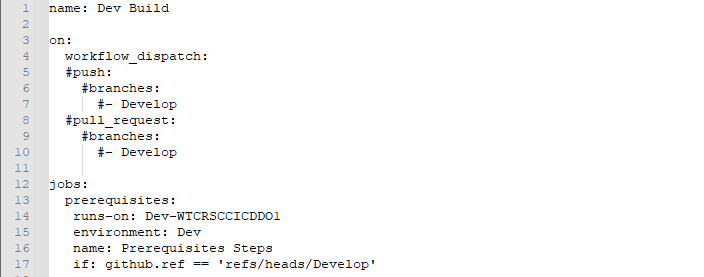
The DEV pipeline is going to be responsible to build the code and deploy to the server WTCRSCGITD01.AMERICAS.GLOBAL.NTTDATA.COM.

### 3.3.1 CI/CD Flow



### 3.3.2 Trigger a deploy

In DEV the deployment is configured to start as soon as a Push or a Pull Request is performed to the DEV branch.



### 3.3.3 Deployment steps

The deployment steps are composed of 5 jobs

* Prerequisites
* Checkout
* Build
* Pre-deploy
* Deploy solution

#### 3.3.3.1 Prerequisites

The **Prerequisite** job is to ensure that no old deploy data prevent the execution of the next one.

The jobs that are part of the **Prerequisites** job are

* Clean up deployment directories
* Create deployment directories

#### 3.3.3.2 Checkout

The **Checkout** job consists to get the latest code from the repository and downloading it to the GitHub Actions Runner in the following directory

C:\actions\Dev\\_work\ SRVS-CIO-DW-DigitalMarketing-SXA \ SRVS-CIO-DW-DigitalMarketing-SXA \d\c

* d means **DEV**
* c means **Checkout**

#### 3.3.3.3 Build

The **Build** job is responsible for building the source code and generating the artifacts of the deployment, and to accomplish that, the steps executed are:

* Install NuGet
* NuGet restore
* Setup Msbuild
* Build DEV CM
* Build DEV CD
* Upload artifacts from both build

#### 3.3.3.4 Pre-deployment

The **Pre-deployment** job can be explained as a checkpoint because it is going to trigger a backup of the DEV Master database before deploying the artifacts to the DEV CM and DEV CD servers.

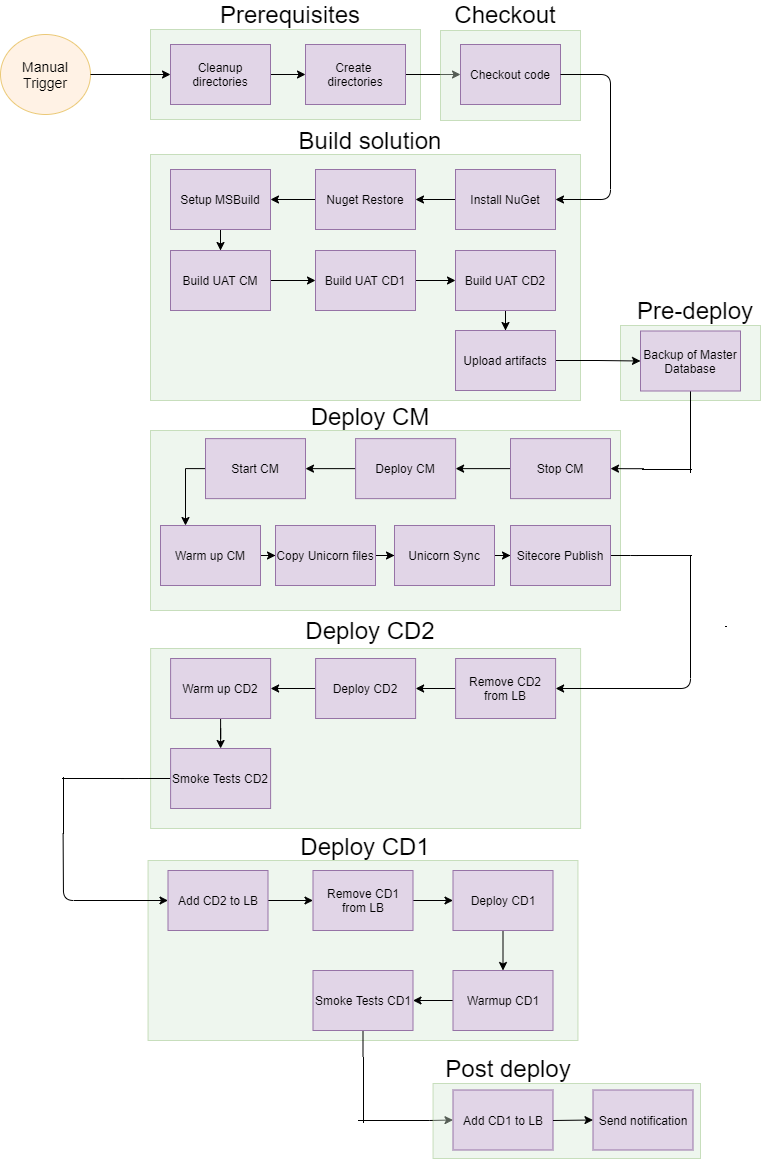
#### 3.3.3.5 Deploy Solution

The **Deploy Solution** job is responsible for copying the artifacts to the destination servers, in this case, DEV CM and DEV CD.

To accomplish this job, there are a few steps that assist it:

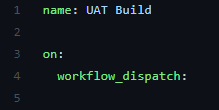
* Stop CM
  + It is going to stop the IIS CM Site, to avoid conflicts while copying the files to the server
* Deploy CM
  + Copy of the artifacts with the new deploy version
* Start CM
  + It is going to start the IIS CD Site to continue the deployment process
* Warm-up CM
  + Initial communication with the CM site, to ensure it is up and running for the next steps
* Copy Unicorn files
  + Copy of Unicorn serialized files from repository to the CM server
* Unicorn Sync
  + Perform the Sync of Unicorn serialized files to CM
* Stop CD
  + It is going to stop the IIS CD Site, to avoid conflicts while copying the files to the server
* Deploy CD
  + Copy of the artifacts with the new deploy version
* Start CD
  + It is going to start the IIS CD Site to continue the deployment process
* Warmup CD
  + Initial communication with the CD Site, to ensure it is up and running
* Send notification
  + Send a notification to communicate whether the deployment succeeded or not

## 3.4 QA-UAT

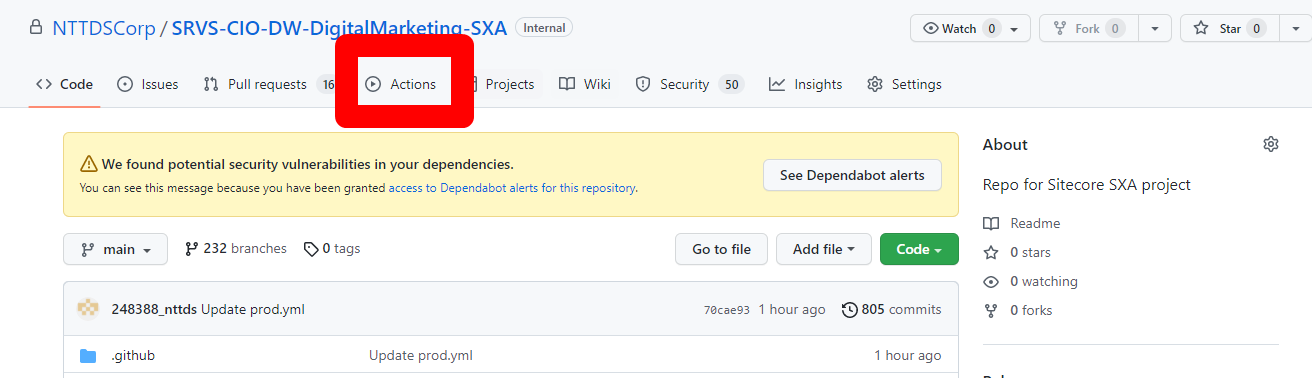


### 3.4.1 Trigger a deploy

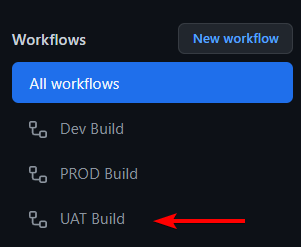
In UAT the deployment is configured to be triggered manually, and the option **workflow\_dispatch** is used in GitHub Actions to perform that way.



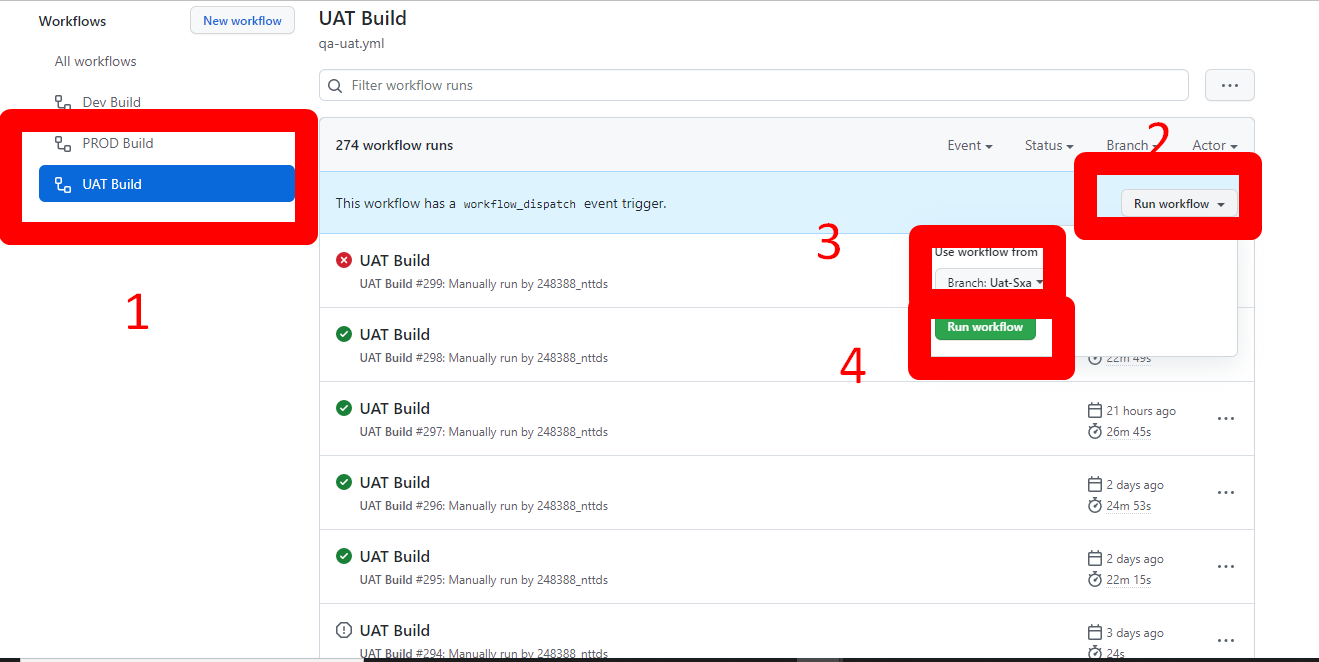
To start a deployment, you should go to **Actions**



Choose the Workflow you would want to execute, in this case **UAT Build**



Then click **Run workflow** (1), select **Use workflow from** (2) and finally **select the branch** (3)

ss

### 3.4.2 Deployment steps

The deployment steps are composed of 5 jobs

* Prerequisites
* Checkout
* Build
* Pre-deploy
* Deploy CM
* Deploy CD2
* Deploy CD1
* Post deploy

#### 3.4.2.1 Prerequisites

The **Prerequisite** job is to ensure that no old deploy data prevent the execution of the next one.

The jobs that are part of the **Prerequisites** job are

* Clean up deployment directories
* Create deployment directories

#### 3.4.2.2 Checkout

The **Checkout** job consists to get the latest code from the repository and downloading it to the GitHub Actions Runner in the following directory

C:\actions\Uat\\_work\ SRVS-CIO-DW-DigitalMarketing-SXA \ SRVS-CIO-DW-DigitalMarketing-SXA \u\c

* u means **UAT**
* c means **Checkout**

#### 3.4.2.3 Build

The **Build** job is responsible for building the source code and generating the artifacts of the deployment, and to accomplish that, the steps executed are:

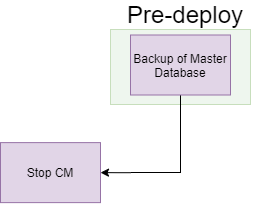
* Install NuGet
* NuGet restore
* Setup Msbuild
* Build UAT CM
* Build UAT CD1
* Build UAT CD2
* Upload artifacts from both build

#### 3.4.2.4 Pre-deployment

The **Pre-deployment** job can be explained as a checkpoint because it is going to trigger a backup of the UAT Master database before deploying the artifacts to the UAT CM and UAT CD servers.

#### 3.4.2.5 Deploy CM

The **Deploy CM** job is responsible for copying the artifacts to the UAT CM server.

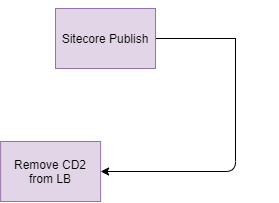


To accomplish this job, there are a few steps that assist it:

* Stop CM
  + It is going to stop the IIS CM Site, to avoid conflicts while copying the files to the server
* Deploy CM
  + Copy of the artifacts with the new deploy version
* Start CM
  + It is going to start the IIS CD Site to continue the deployment process
* Warm-up CM
  + Initial communication with the CM site, to ensure it is up and running for the next steps
* Copy Unicorn files
  + Copy of Unicorn serialized files from repository to the CM server
* Unicorn Sync
  + Perform the Sync of Unicorn serialized files to CM
* Publish Sitecore
  + Perform a publish of the Sitecore items required

#### 3.4.2.6 Deploy CD2

The **Deploy CD2** job is responsible for copying the artifacts to the UAT CD2 server.

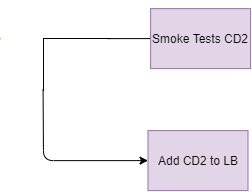


To accomplish this job, there are a few steps that assist it:

* Remove CD2 from LB
  + During the deployment process, the CD2 is restarted, to avoid visitors experiencing downtime, CD2 is removed from the live Load Balancer traffic
* Deploy CD2
  + Copy of the artifacts with the new deploy version
* Warmup CD2
  + Initial communication with the CD2 site, to ensure it is up and running for the next steps
* Smoke Tests CD2
  + Perform initial tests on specific URLs, and ensure tests are returning 200 status

#### 3.4.2.7 Deploy CD1

The **Deploy CD1** job is responsible for copying the artifacts to the UAT CD1 server.

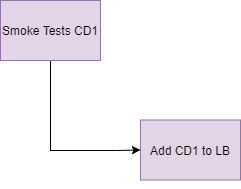


To accomplish this job, there are a few steps that assist it:

* Add CD2 to LB
  + Now, that CD2 has been tested and the deployment was approved to continue, the CD2 is added to the LB and start to receive traffic
* Remove CD1 from LB
  + During the deployment process, the CD1 is restarted, to avoid visitors experiencing downtime, CD1 is removed from the live Load Balancer traffic
* Deploy CD1
  + Copy of the artifacts with the new deploy version
* Warmup CD1
  + Initial communication with the CD1 site, to ensure it is up and running for the next steps
* Smoke Tests CD1
  + Perform initial tests on specific URLs, and ensure tests are returning 200 status

#### 3.4.2.8ss Post Deploy

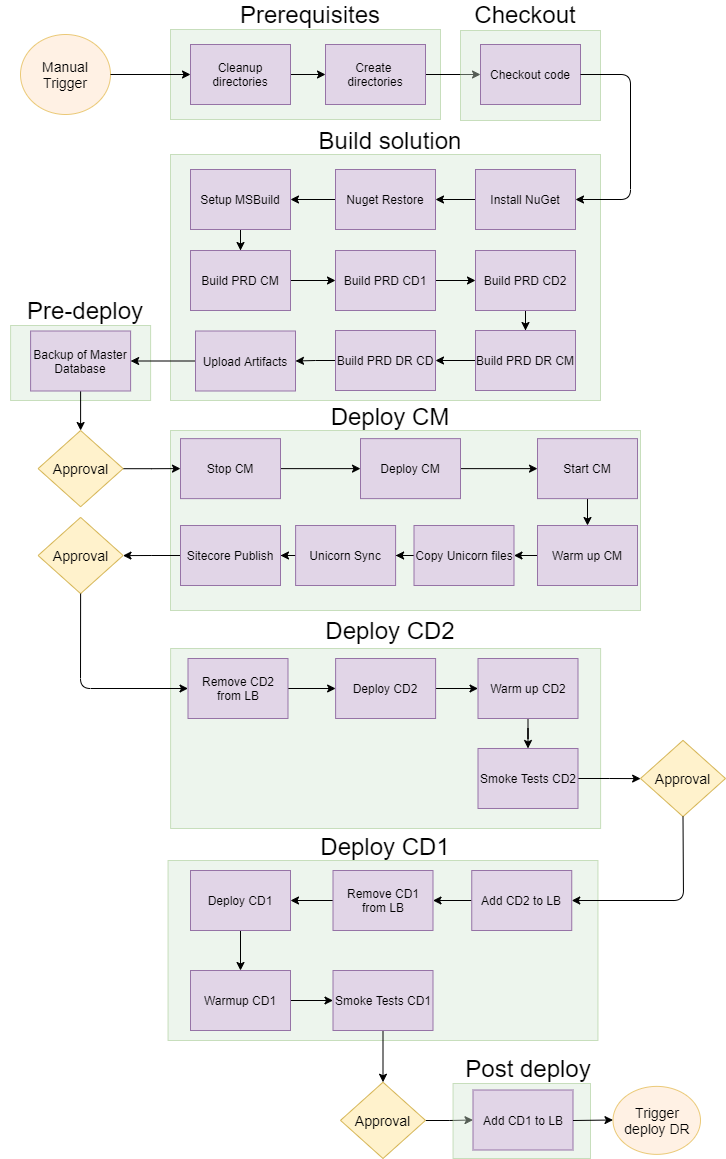
The **Post Deploy** job is responsible for adding the CD1 back to the load balancer and sending notification of the deployment.

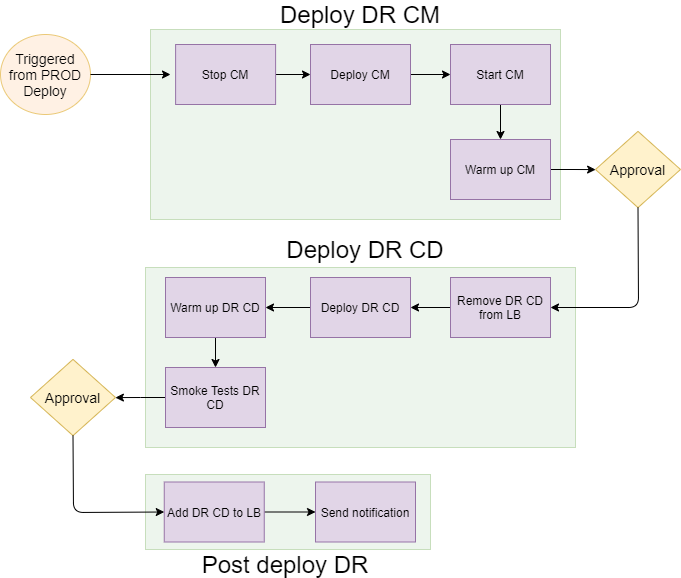


To accomplish this job, there are a few steps that assist it:

* Add CD1 to LB
  + Now, that CD1 has been tested and the deployment was approved to continue, the CD1 is added to the LB and start to receive traffic
* Send notification
  + Send a notification to communicate whether the deployment succeeded or not

## 3.5 PROD



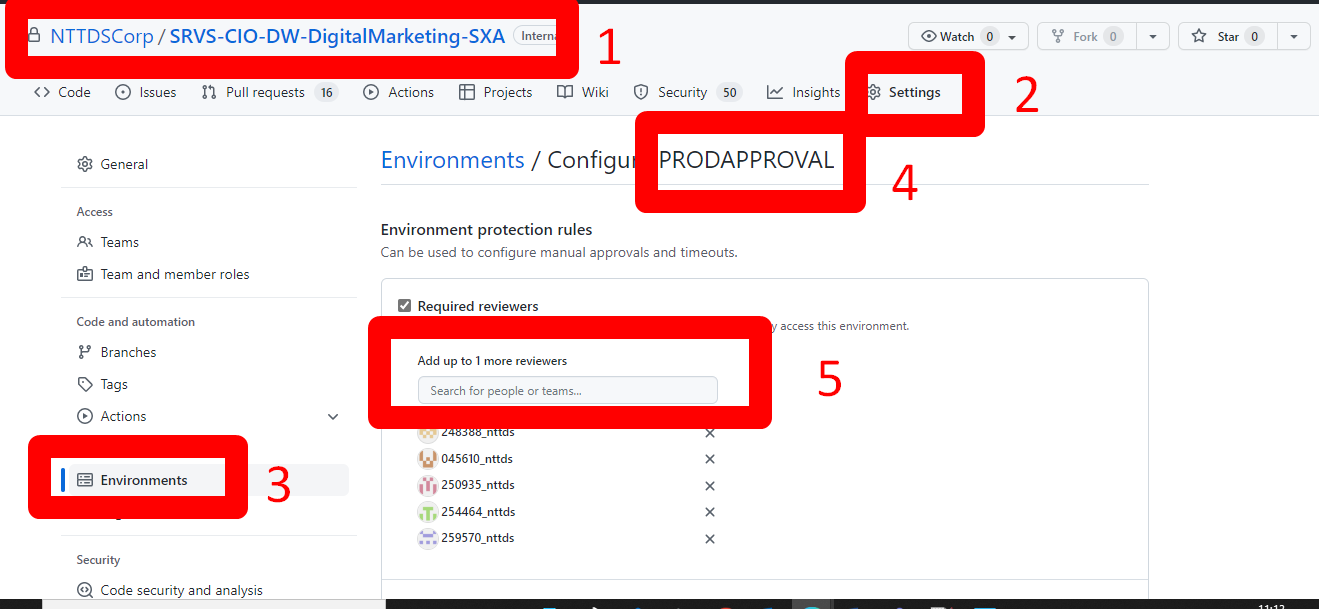


### 

### 3.5.1 Adding Approval names to approve job

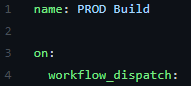
### In PROD need to give approval in each deployment step. Whoever approval names are added they can only give an approval to execute the job.

To add an approval, you should go to the repository which will have the GitHub Actions and click on Settings(1) Environments(2) New Environment(3) add name ex(PRODAPPROVAL) (4) Required Viewers

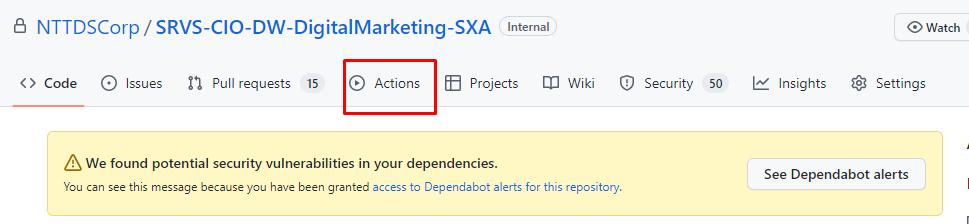


### 3.5.2 Trigger a deploy

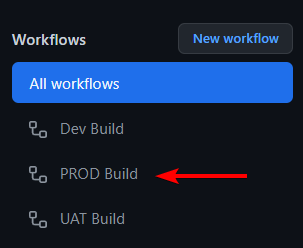
In PROD the deployment is configured to be triggered manually, and the option **workflow\_dispatch** is used in GitHub Actions to perform that way.



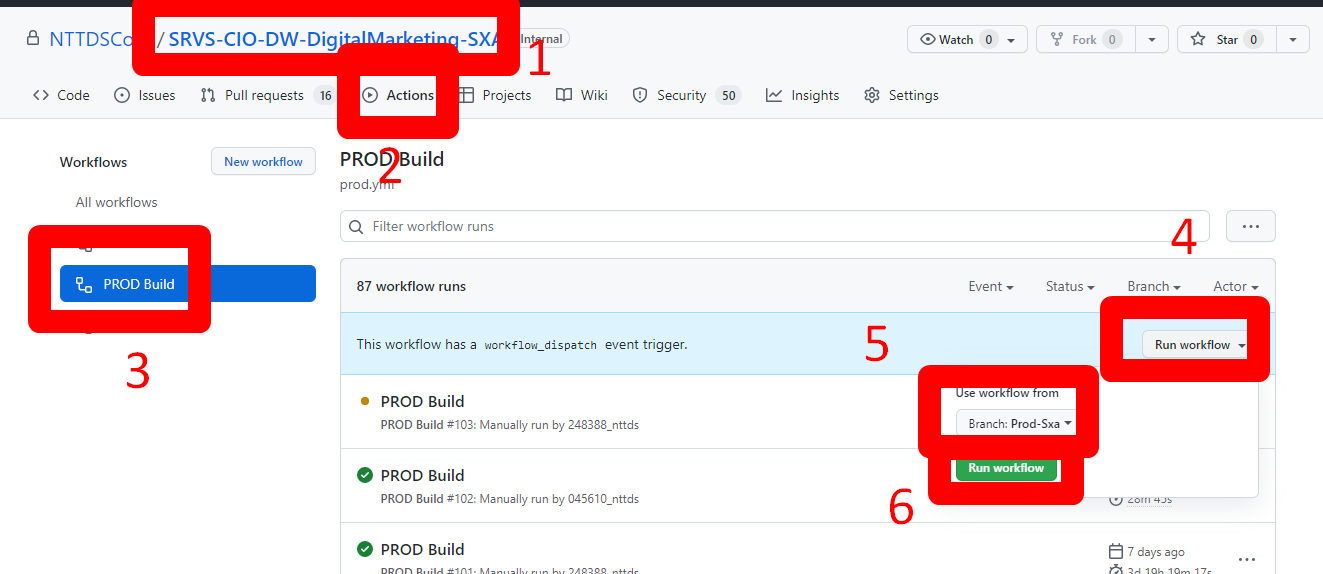
To start a deployment, you should go to **Actions**

ss

Choose the Workflow you would want to execute, in this case **PROD Build**



Then click **Run workflow** (1), select **Use workflow from** (2) and finally **select the branch** (3)



### 3.5.3 Deployment steps

The deployment steps are composed of 5 jobs

* Prerequisites
* Checkout
* Build
* Pre-deploy
* Deploy CM
* Deploy CD2
* Deploy CD1
* Post deploy
* Deploy DR CM
* Deploy DR CD
* Post deploy DR

#### 3.5.2.1 Prerequisites

The **Prerequisite** job is to ensure that no old deploy data prevent the execution of the next one.

The jobs that are part of the **Prerequisites** job are

* Clean up deployment directories
* Create deployment directories

#### 3.5.2.2 Checkout

The **Checkout** job consists to get the latest code from the repository and downloading it to the GitHub Actions Runner in the following directory

C:\actions\Prod\\_work\SRVS-CIO-DW-DigitalMarketing-SXA\SRVS-CIO-DW-DigitalMarketing-SXA\p\c

* p means **PRODss**
* c means **Checkout**

#### 3.5.2.3 Build

The **Build** job is responsible for building the source code and generating the artifacts of the deployment, and to accomplish that, the steps executed are:

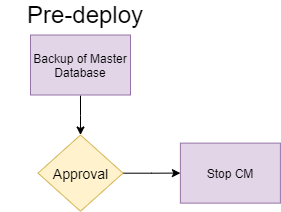
* Install NuGet
* NuGet restore
* Setup Msbuild
* Build PRD CM
* Build PRD CD1
* Build PRD CD2
* Build PRD DR CM
* Build PRD DR CD
* Upload artifacts

#### 3.5.2.4 Pre-deployment

The **Pre-deployment** job can be explained as a checkpoint because it is going to trigger a backup of the PRD Master database before deploying the artifacts to the PRD CM, PRD DR CM, PRD CD and PRD DR CD servers.

#### 3.5.2.5 Deploy CM

The **Deploy CM** job is responsible for copying the artifacts to the PRD CM server. Also, the Deploy CM job only starts after getting approved.

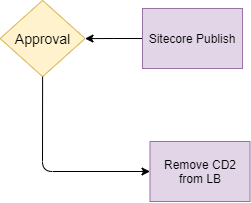


To accomplish this job, there are a few steps that assist it:

* Stop CM
  + It is going to stop the IIS CM Site, to avoid conflicts while copying the files to the server
* Deploy CM
  + Copy of the artifacts with the new deploy version
* Start CM
  + It is going to start the IIS CM Site to continue the deployment process
* Warm-up CM
  + Initial communication with the CM site, to ensure it is up and running for the next steps
* Copy Unicorn files
  + Copy of Unicorn serialized files from repository to the CM server
* Unicorn Sync
  + Perform the Sync of Unicorn serialized files to CM
* Publish Sitecore
  + Perform a publish of the Sitecore items required

#### 3.5.2.6 Deploy CD2

The **Deploy CD2** job is responsible for copying the artifacts to the PRD CD2 server. Also, the Deploy CD2 job only starts after getting approved.

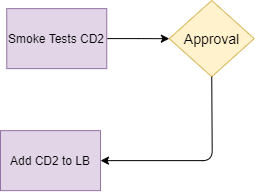


To accomplish this job, there are a few steps that assist it:

* Remove CD2 from LB
  + During the deployment process, the CD2 is restarted, to avoid visitors experiencing downtime, CD2 is removed from the live Load Balancer traffic
* Deploy CD2
  + Copy of the artifacts with the new deploy version
* Warmup CD2
  + Initial communication with the CD2 site, to ensure it is up and running for the next steps
* Smoke Tests CD2
  + Perform initial tests on specific URLs, and ensure tests are returning 200 status

#### 3.5.2.7 Deploy CD1

The **Deploy CD1** job is responsible for copying the artifacts to the PRD CD1 server. Also, the Deploy CD1 job only starts after getting approved.

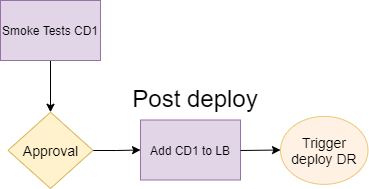


To accomplish this job, there are a few steps that assist it:

* Add CD2 to LB
  + Now, that CD2 has been tested and the deployment was approved to continue, the CD2 is added to the LB and start to receive traffic
* Remove CD1 from LB
  + During the deployment process, the CD1 is restarted, to avoid visitors experiencing downtime, CD1 is removed from the live Load Balancer traffic
* Deploy CD1
  + Copy of the artifacts with the new deploy version
* Warmup CD1
  + Initial communication with the CD1 site, to ensure it is up and running for the next steps
* Smoke Tests CD1
  + Perform initial tests on specific URLs, and ensure tests are returning 200 status

#### 3.5.2.7 Post Deploy

The **Post Deploy** job is responsible for adding the CD1 back to the load balancer and to trigger the deployment to the DR Region. Also, the Post Deploy job only starts after getting approved.



To accomplish this job, there are a few steps that assist it:

* Add CD1 to LB
  + Now, that CD1 has been tested and the deployment was approved to continue, the CD1 is added to the LB and start to receive traffic
  + Trigger deploy to DR Region

#### 3.5.2.8 Deploy DR CM

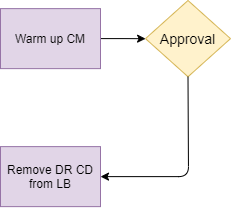
The **Deploy DR CM** job is responsible for copying the artifacts to the PRD DR CM server.

To accomplish this job, there are a few steps that assist it:

* Stop CM
  + It is going to stop the IIS CM Site, to avoid conflicts while copying the files to the server
* Deploy CM
  + Copy of the artifacts with the new deploy version
* Start CM
  + It is going to start the IIS CM Site to continue the deployment process
* Warm-up CM
  + Initial communication with the CM site, to ensure it is up and running for the next steps

#### 3.5.2.9 Deploy DR CD

The **Deploy CD2** job is responsible for copying the artifacts to the PRD DR CD server. Also, the Deploy CD2 job only starts after getting approved.

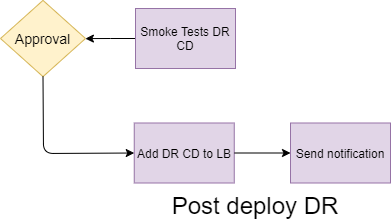


To accomplish this job, there are a few steps that assist it:

* Remove DR CD from LB
  + During the deployment process, the DR CD is restarted, to avoid visitors experiencing downtime, DR CD is removed from the live Load Balancer traffic
* Deploy DR CD
  + Copy of the artifacts with the new deploy version
* Warmup DR CD
  + Initial communication with the DR CD site, to ensure it is up and running for the next steps
* Smoke Tests DR CD
  + Perform initial tests on specific URLs, and ensure tests are returning 200 status

#### 3.5.2.10 Post DR Deploy

The **Post DR Deploy** job is responsible for adding the DR CD back to the load balancer and sending notification of the deployment. Also, the Post Deploy job only starts after getting approved.



To accomplish this job, there are a few steps that assist it:

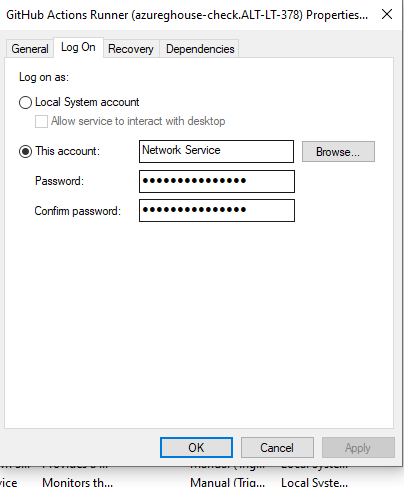
* Add DR CD to LB
  + Now, that DR CD has been tested and the deployment was approved to continue, the DR CD is added to the LB and start to receive traffic
* nUnit
  + Execute nUnit validation
* Send notification
  + Send a notification to communicate whether the deployment succeeded or not.

# 4 TROUBLESHOOT STEP:

* Once agent runner has been installed need to change the user name and password

|  |  |
| --- | --- |
| User Name | Global\Sitecoredev |
| Password | D3vs1t3acct |

* To add this user name and password in agent pool
* Login into the machine where agent has been installed ex(10.111.36.109)
* Search service in search box, search the name of agent and click (1) logon (2) enter user and password.



s

If extra folder has been added in path NTTDSCorp/SRVS-CIO-DW-DigitalMarketing-SXA/p/c/NTTDataAmerica/src then need to add the extra folder name in yamil code in copying unicorn steps

