Deployment Process in TagworX Test, PreProd and Production

1. The development team will inform us of any new deployments to the Test or preprod or prod environments via the **#devops** channel in Slack.
2. When requesting a deployment, the developers need to provide the following information:

a. Which is the **microservice** to which the deployment is taking place?
b. Is the deployment toTest or pre-production or production?
c. What **branch** is to be used for this deployment?

Example : Hello, @sd-team Please proceed with the deployment in TEST (18.133.130.52) (**test environment b**) for the following micro-services: tagworx-ukho branch develop

3. Once you have the details, you must prepare the deployment checklist. You must create a checklist for each microservice that is being deployed.
4. Recently New servers were added in Test and production accounts.

List of Servers added newly :

1. Tagworx_v2 (AWS Test account)
2. V2_pre-prod-replica (AWS Production account)
3. V2_pre-prod-main (AWS Production account)
4. V2_Prod-replica (AWS Production account)
5. V2_Prod-main (AWS Production account)

Checklist For the deployments

For the Test server (Tagworx - 18.133.130.52), there is a Pipeline setup for the test server (Tagworx). Whenever deployment comes in the Test server. We need to give manual approval and Bug will generate automatically.

Steps :

1. Login into the AWS test account and go to the code pipeline.
2. Check with Microservice as mentioned by the development team.
3. Go inside it and cross verify the deployment with commit ID in pipeline and bitbucket repository.
4. Then Give the Manual approval and wait for deployment to complete.

For **Test_v2, Pre-prod, Pro-prod_v2 and prod, Prod_v2.** We have to do the manual deployment on servers and also create bugs manually.

Checklist for deployment things need to add in the bug.

Example : Hi @Sd-team Please proceed with the deployment in PP-B (v2_PreProd), **both servers** for the following micro-services: tagworx-inventory branch stage

~~~

Test or Pre- Prod or Prod: V2_pre-prod

Instances: Main and replica

Microservice: tagworx-inventory

Branch to be deployed: stage

~~~

Before going to below steps, First create a bug and add the above checklist in bug, then proceed with deployment and mention all steps performed in the server and copy and paste in the bug for tracking the deployment.

Steps :

1. Login into the AWS Prod account and go to EC2 instances.
2. Check the Microservice as mentioned by the development team.
3. Go inside Target groups and select the group
4. Each target group has a “main” and a “replica” instance. All deployments are first done to the replica and then to the main instance.
5. To do this, in the Targets tab, select the replica instance and click on deregister. Deregistration takes approximately 3-5 minutes
6. Each instance has its own security group. The current security groups do not allow direct access to port 80 on the prod instances. So the port needs to be opened in order for developers to test the updates on the specific instance to which the updates are deployed. For test and preprod port 80 is always open.
7. Once the replica is de-registered, go to EC2 replica instance and modify its security group inorder to add HTTP inbound rule inside it and allow it for anywhere.
8. After this, connect with the replica instance using AWS session manager. All dev work is done under the “ubuntu” user on all instances. The micoservice files are deployed to the corresponding folder under “/home/ubuntu/”

   sudo -i

   su - ubuntu
9. As a fallback option, backup the current microservice folder.
   e.g. If we are deploying to tagworx-ukho

mkdir -p /backup/`date +%m-%d-%Y`

cp -prvf tagworx-ukho /backup/`date +%m-%d-%Y`/

***NOTE:* Make sure to keep an eye on disk space usage before creating a new backup.**

10. Go inside the relevant folder
    e.g. If we are doing deployment for ukho microservice go inside the folder:

cd /home/ubuntu/tagworx-ukho

11. Now check the previous commits we had on this branch

git log --oneline

Copy and paste the first 5 commit details in the notepad.

12. After that, run the following commands to check which branch you are in:

git branch

13. If the branch is the sames as the branch being deployed, pull the latest code from repository with:

git pull origin branchname

E.g. git pull origin release/2.19.019

14. If the branch is a new one, checkout the new branch and then pull to confirm latest code:

git fetch origin

git checkout branchname

git pull origin branchname

E.g. git fetch origin

git checkout release/2.20.0

git pull origin release/2.20.0

15. Now go inside scripts folder with

cd scripts

16. Run AfterInstallHook script with

./AfterInstallHook.sh

17. Once the script is completed that means deployment is complete on the replica server. Confirm with the developer if the changes are reflected on replica or not.

If no issues are reported by developers

1. Cross check the deployment checklist to see if more deployments are required.
2. After this we have to repeat these steps for the other instance if required.
3. Go back to the AWS console and register the instance in the target group again and remove the security inbound rule for HTTP from the instance's security group.
4. After this the deployment is complete, inform the same in slack.

ROLLBACK PROCEDURES

If issues are reported by developers

1. If the deployment was to a new branch, do the following:
    a. Go inside the relevant folder

e.g. If the deployment was for ukho microservice, go inside the folder:

cd /home/ubuntu/tagworx-ukho

    b. And run the following, where branchname is the name of the previous branch

git checkout branchname

    c. Then run AfterInstallHook script with

cd scripts

./AfterInstallHook.sh

2. If the deployment was to an existing branch, request the developers to revert the last commit and push it to bitbucket. Then do the following:

a. Go inside the relevant folder

e.g. If the deployment was for ukho microservice go inside the folder:

cd /home/ubuntu/tagworx-ukho

b. And run the following, where branchname is the name of the current branch:

git pull origin branchname

c. Then run AfterInstallHook script with

cd scripts

./AfterInstallHook.sh

3. In the unlikely event there is a corruption of the git library and we are unable to revert the changes via git, restore the backed up microservice folder.

a. Go inside ubuntu user home folder as ubuntu user

cd /home/ubuntu/

b. Stop the current microservice instance

e.g. If the failed deployment is for tagworx-ukho

sudo pm2 delete tagworx-ukho

c. Backup the current microservice folder labelled as "-failed-date"

E.g. **mv tagworx-ukho /backup/tagworx-ukho-failed-`date +%m-%d-%Y`**

d. Recover the microservices folder:

e.g. If the microservice is tagworx-ukho

cp -prf /backup/DATE-BACKUP-WAS-TAKEN/tagworx-ukho tagworx-ukho

e. Then run AfterInstallHook script with

e.g. If the microservice is tagworx-ukho

cd tagworx-ukho/scripts

./AfterInstallHook.sh

f. After confirmation from developers that no issues are reported, remove the backup

e.g.

cd /backup

rm -rf tagworx-ukho-failed-DATE-BACKUP-WAS-TAKEN

4. If the issue is reported after the deployment has been done for both instances, then the above mentioned steps will have to be performed on both the main and replica instances.

Issues, Error and queries

1. If deployment fails in the deploy stage in the test server via pipeline and also via manual deployment. It will fail in "afterinstallhook" script because of either "Java heap memory" or "client:build" failure.

   Troubleshooting : Check the available memory on the server with **" free -h"** and this can be done by modifying the AfterInstallHook script and adding **"NODE_OPTIONS=--max-old-space-size=4096"** and redeploying it.

   Trello : https://trello.com/c/QmA6ixow/435-jacascript-heap-out-of-memory

2. Recently there were issues with the pre-prod login page is not loading. As developers are unable to login, even after login it shows a blank page.

   Troubleshooting : To avoid these issues. There are few steps to resolve

   1. Access the url and check if the login page is working or not. If not, restart the tagworx-user micro service in both main and replica and check again accessing it.
   2. Then after login, check whether the data is showing or not. If not, restart the tagworx-inventory micro-service in both main & replica and check again. If issues is still persist
   3. Check the code and commit id difference in both servers (main and replica). If both servers have different code, then ask the developer to push new code to the server and redeploy it.
   4. If the issue persists, then restart the server and check again.

3. Chris : hi @Sd-team please can you check that both PROD servers are ok...I'm seeing some intermittent issues.

   Troubleshooting : To avoid these issues. There are few steps to follow

   1. Access the url and check if the login page is working or not. If the login page is working fine. But even after developers are informed regarding the same. Try to access the server with there individual IP’s
   2. Even after getting the same issue. Try to remove one server from the target group and check with the URL, if it’s working fine. Try to attach the other server as the same and check again. If you find any issues with the login page then remove it from the target group and restart the server and attach it back. check with URL whether accessing or not.