TM DevOps – Use Case

TM DevOps – Use Case

Document Details

Use Case Name	**TMDevOps – Use Case03**
First Draft	**01st Dec 2017**
Author	**Prabhakar D**
Reviewed By	**Pradeep Narayanaswamy**

TM DevOps – Use Case

Contents

TM DevOps – Use Case

Scope

This document provides a detailed use case study on development and operations (DevOps) using Continuous integration and continuous delivery (CI/CD) including AWS services and third-party application, with guidelines for implementation.

About Customer

LnTMetro - The Hyderabad Metro Rail Project is the World's Largest Public-Private Partnership Project (PPP) in the Metro Sector. The Metro Network will cover a total distance of around 72 Km across three corridors, transporting Hyderabad to the future.

Use Case Description

LT Metro is public listed company did metro rail project for Hyderabad HMC and we TechMinfy developed application for passenger ticket issue system on server and mobile app development by third party with stack of Java on Application and database on MySQL on AWS Infrastructure.

This project was not DevOps compliance on any dev/staging/prod environments. Following activities are completely manual process.

- code push
- code merge
- build & test
- deployment

Development team used to spend more time on the above said activities instead of production activities. We at TechMinfy considered CI/CD integration for these activities.

Primary & Supporting Actors

Actors from TechMinfy:

Prabhakar Dharmalingam (Primary)
Raghu Penchala (Supporting)

Actors from LnTMetro Project- at TechMinfy:

Deployment – Anant Joshi (Primary)
Developers: Praveen Vadla (Primary)
Santosh K (Supporting)

TM DevOps – Use Case

Pre & Post Conditions

Pre-implementation state at LnTMetro Project:

- Code push on SVN local
- Java WAR file deployment was a manual process which led to wrong deployments on wrong environments on Tomcat.

Post-implementation state at Metro Project:

- We enabled code repository on centralized and private code repositories using AWS CodeCommit
- Code merge process attached to Jenkins jobs and wrong code merges were prevented and is now a one-click process.
- We restricted code merge job only to DEV team leader, so unauthorized or multiple code merge process is controlled wrong repo or branch code merge is prevented.
- We automated the code pull and checkout process using Jenkins, code pull from wrong branch is now prevented
- We replaced Local Eclipse compile process with Jenkins maven build step and the compile errors are now logged and reviewed for documentation or bug tracking purposes
- Email notification is enabled in case of build failures
- Deployment using Jenkins job with Tomcat plugin prevented wrong environment deployment on Tomcat
- For build rollback purpose each and every successful builds are now archived using Jenkins parameterized build and rollback is performed in-case of build failures
- DEV Team time is utilized effectively on product development instead of build activities

Trigger

- Multiple events of wrong code merge and code pull for process was the main root cause to trigger the decision to implement CI/CD (DevOps) process in the software development life cycle policy.
- Since build and compile errors were not tracked, developers used to spend more time to re-run the compiler to track the build errors which was another reason for the process to be implemented.
- DEV team leader decided to implement CI/CD Process for Code Pull, Compile, Build, Test and Deployment using AWS Services with support of TechMinfy.

TM DevOps – Use Case

Scenario

Steps to automate build process

- Setup Jenkins with JDK 8 on AWS EC2 instance
- Installed all Jenkins base plugins as pre-requisites
- Installed AWS credentials plugin to authenticate with AWSCodeCommit
- Installed AWS CodeDeploy plugin to automate Java WAR deployment
- Create maven type job to automate the build process of
    - Code pull
    - Code checkout to the correct branch
    - Install dependent packages for build
    - Build JAR or WAR file
    - Deploy on Tomcat environment
    - Archive successful builds as artefacts for build release history and rollback plan
- E-Mail notifications on build status with logs

Introduction to DevOps

Dev and Ops are no longer exclusively separate roles within the IT space. Today, they're morphing into one cohesive method and opportunity which is reshaping the way IT teams operate. Most would define DevOps as a movement, practice, or culture that aims to tie IT professionals and software developers together so that they can more efficiently streamline infrastructure changes and software delivery. Essentially, it's rooted in the idea that **building, testing and releasing software can run more smoothly and automatically if the appropriate team of professionals are working together.**

Application Architect on AWS

LnTMetro Application Stack

- Application – JAVA using Tomcat
- Database – MySQL using AWS RDS

Implementation of CI/CD

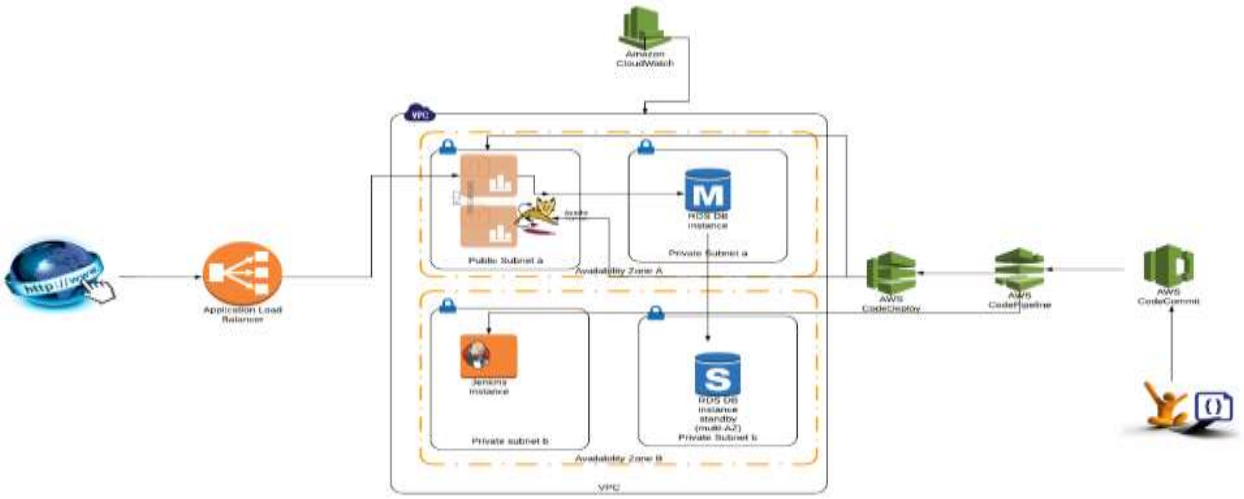
TechMinfy supported LnTMetro to get on to the DevOps (Continues Integration) Platform on AWS using AWS Commit, Jenkins and AWS Code Deploy.

DevOps – CI/CD Architecture for LnTMetro for JAVA

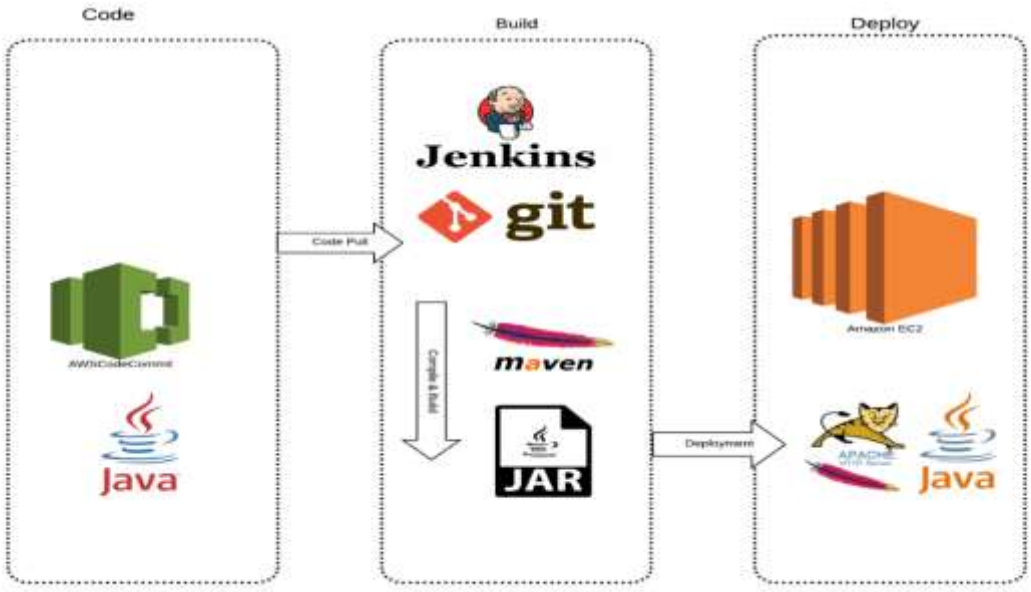
Code Repository: AWS CodeCommit
Build: Jenkins 2.7.9 using git and maven build plugin
Deployment: Tomcat on EC2
CI Integration process steps

TM DevOps – Use Case

- We used freestyle job from AWS CodeCommit Repository to trigger build when developers pushed any code changes to the repo and its branches using SCM poll
- Build use maven build plugin was used to generate jar or war file
- Using Tomcat plugin, we deployed war file on Tomcat
- Build Rollback plan was enabled for build failure
- Email notifications was sent to the team on build status



DevOps process flow diagram



TM DevOps – Use Case

Security with DevOps

Continuous Integrations (Code)
LnTMetro development project was developed and implemented by TechMinfy DEV Team. We migrated Code repositories from SVN to AWS CodeCommit as private code repositories. AWS CodeCommit by default does not allow any developers to push changes without authentication. So we have integrated HTTPS authentication to make developers to push changes on AWS CodeCommit private code repositories.

Automated Jenkins build authentication
We pulled code for Jenkins builds to get authenticated using https git credentials using AWS CLI commands and Jenkins saved AWS credentials for each and every build code pull

Continuous Delivery (Code Merge)
LnTMetro Project has different environments DEV, Staging and Production with different app codes for Frontend, Backend, Recharge Card and Payment. Each and every application codes merges from DEV to Staging to PROD environment are controlled by deploying Jenkins Authorize Project Plugin , code merges are now secured merges. LnTMetro Project needs to keep a track of each build specifically to check who breaks builds and need to get notified, we have configured build tracking logging and configured E-Mail notifications to send notifications when builds are failed with all relevant information.

Continuous Deployment
LnTMetro Project deployments are securely authenticated using AWS access key and secret keys for deployment on AWS EC2 instances for Jenkins deployment AWS CLI commands to run using shell commands

Rollback for deployment failures
Build rollback was enabled using Jenkins Deployment plugin and configured as Parameterized rollback plan, in case build is success but functionality broken due to last code commit

Build versions
All builds are archived and build versions are tracked for specific build version deployment and also for the roll back purpose. All build versions are controlled with parameterized build option in Jenkin Jobs

TM DevOps – Use Case

DevOps – CI / CD overall benefits

Version Control

- Each and every builds artifacts are archived and tracked and can now be redeployed to any target environments on LTMetro Project

Quick Delivery

- Post implementation of CI and CD for LTMetro proejct ,all bug fixes are delivered quickly for all applications (Frontend, Backend, RechargeCards, Payment services)

Saves Time

- LTMetro Project used to run repeated builds before implementation CI and CD, around 20 builds in a week
- Post CI and CD implementation, repeated and deplulicated builds are prevented saving time
- CI and CD helps LTMetro Project to prevent wrong code commits as it is integraated and automated for each and every build
- Continuous Integration saves time for LTMetro Project on AWS code commit, merge, pull and prevents any accidental pushes to wrong repositories and branches
- Continuous delivery also saves time for LTMetro Project as developer need not spend time on manually building and log tracking
- Continuous deployment saves time of for LTMetro Project- IT Ops Manager/System Admins as there is no manual intervention required for deployment
- CI/CD implementation also enabled automated email notification so that LTMetro Project developers and IT Ops team are notified of each and every build sand deployments.
- Faster build time when compared to manuall build and deployment
- Now LTMetro Project requires no manual maintenance as builds and logs cleaner is automated with Jenkins workspace cleanup plugin , saving time for IT Ops team from manual disk space cleaner
- LTMetro Project need not manually track builds and deployments history, automated build and deployment tracking with builds history for any future analysis
- Jenkins rollback features saves time in planning for rollback, as the functionality helps to rollback with last success build version

Productiveness

- Dev team and Ops team can now focus more on productivity as they don't have to spend much time on manual build, error tracking and deployment failures.
- They can focus more on bug fixes of build and deployments
- Better builds release management helps LnT Metro Ops team to stream line deployments on appropriate targets resources

TM DevOps – Use Case