# TM DevOps – Use Case

# TM DevOps – Use Case

**Document Details**

| | |
|---|---|
| **Use Case Name** | **TMDevOps – Use Case01** |
| **First Draft** | **5th March 2018** |
| **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**

Knowledge Crystals Private Limited is a Private incorporated on 21 October 2014 and product based development firm in initial stage its product development on standard web application and mobile applications.

**Use Case Description**

Knowledge Crystals is software Development Company and developing their product with stack of Java and Angular JS and deployment on AWS Infrastructure.

This company was not DevOps compliance. Following activities were a manual process.

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

Development team used to spend more time on the above said activities instead on productivity. Knowledge Crystals approached us to provide CI/CD integration for these activities.

**Primary & Supporting Actors**

***Actors from TechMinfy:***

Prabhakar Dharmalingam (Primary)
Raghu Penchala (Supporting)

***Actors from Knowledge Crystals:***

Divyanand (Primary)
Prasanth Kumar (Supporting)

# TM DevOps – Use Case

**Pre & Post Conditions**

***Pre-implementation state at Knowledge Crystals:***

- Code merge process was manual
- Wrong code merge or pull had occurred many times i.e. developer needed to merge code from KCWeb/master branch to its sub branch "subranch1", but instead that KCWeb/master branch code merged to KCBBackend/master. This had caused complete mess up on two repositories
- Eclipse Java compile was a manual process and errors were not tracked
- Java WAR file deployment was manual process and wrong deployments also occurred to irrelevant environments on AWS Elastic beanstalk.
- DEV team spent more time on fixing the above mentioned issues instead of product development

***Post-implementation state at Knowledge Crystals:***

- Code merge process attached Jenkins jobs and wrong code merges are prevented and it is automated process with one click by DEV team leader
- Code merge job is restricted only to DEV team leader, so unauthorized or multiple code merge processes are controlled and wrong repo or branch code merger is prevented
- Code pull and checkout process is automated using Jenkins, code pull from wrong branch is prevented
- Eclipse compile process was replaced with Jenkins maven build step and all compile errors are now logged and the errors logged are notified to the respective developer team leader and team by e-Mail notification in case of build failures
- Deployment using Jenkins job with AWS Elastic Beanstalk plugin(Cloudbees) prevented wrong environment deployment on AWS Elastic Beanstalk
- 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
- Install all Jenkins base plugins as pre-requisites
- Install AWS credentials plugin to authenticate with AWSCodeCommit
- Install AWS EB 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 AWS Elastic Beanstalk 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 that 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**

Knowledge Crystal application stack

- Frontend Angular 2 – static content for UI framework deployed on AWS S3 bucket using static website hosting
- Backend application – JAVA using AWS Beanstalk
- Database – MySQL using AWS RDS

**Implementation of CI/CD**

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

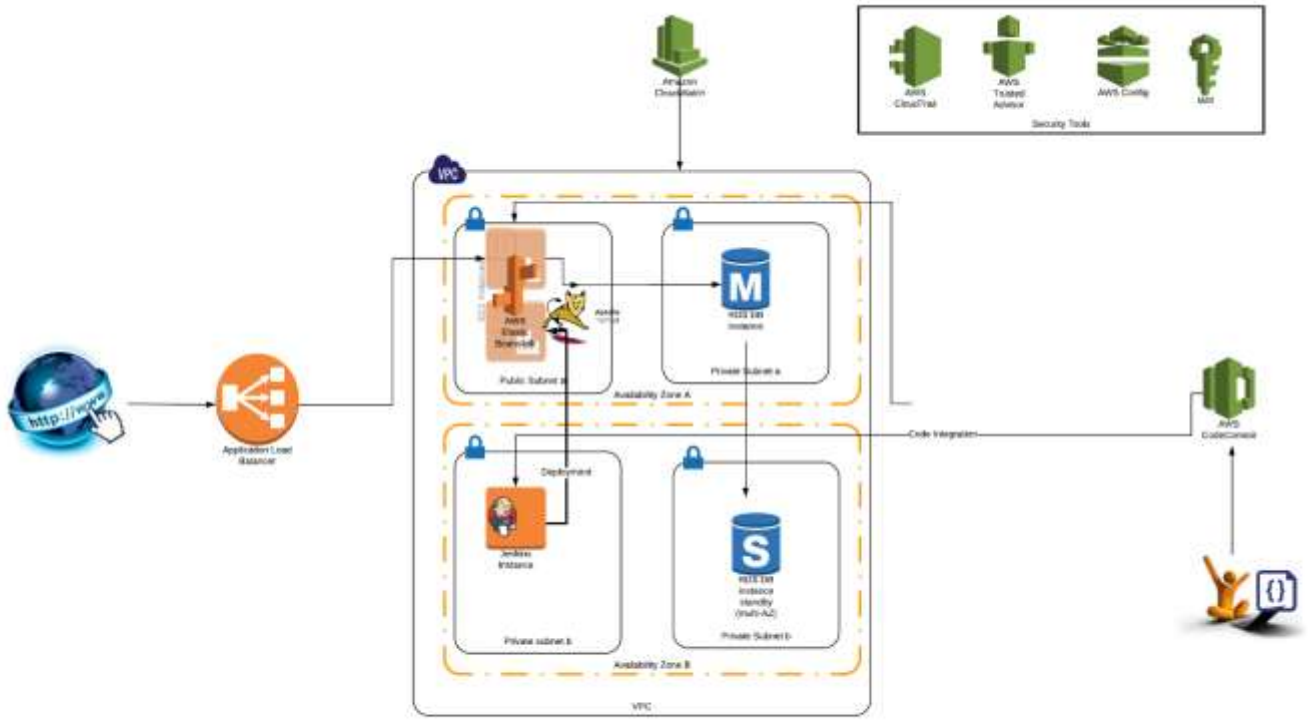
**DevOps – CI/CD Architecture for Knowledge Crystals for JAVA using Elastic Beanstalk**

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

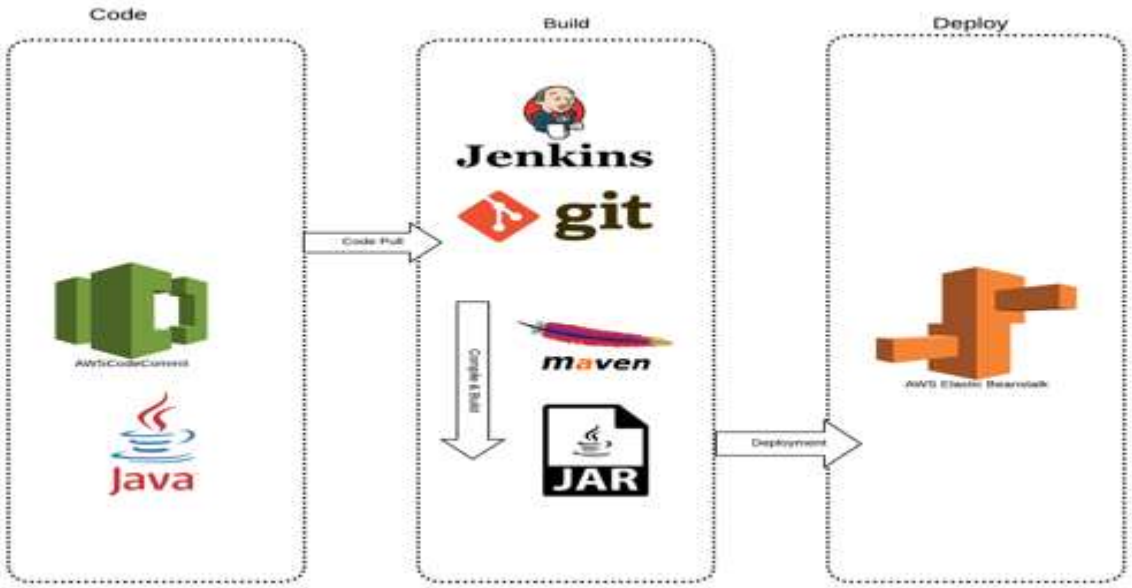
# TM DevOps – Use Case

- We enabled freestyle job using 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 AWS beanstalk Plugin, we deployed war file on AWS beanstalk
- Build Rollback plan was enabled for build failure
- Email notifications was sent to the team on build status

**CI/CD Flow Diagram for Java with AWS Beanstalk**



# TM DevOps – Use Case




**DevOps with Elastic Beanstalk**

Elastic Beanstalk used for this JAVA deployments with blue and green, zero downtime deployment with tomcat 7 version and JDK 1.7 version. Customer finds this Elastic Beanstalk deployment is easy and quick deployment of the JAVA Application stack. Post integration of Elastic beanstalk with Jenkins CI/CD process the completed SDLC become fully automated end to end DevOps solution with Elastic beanstalk and Jenkins

**DevOps – CI/CD Architecture for Knowledge Crystals for Angular 2**

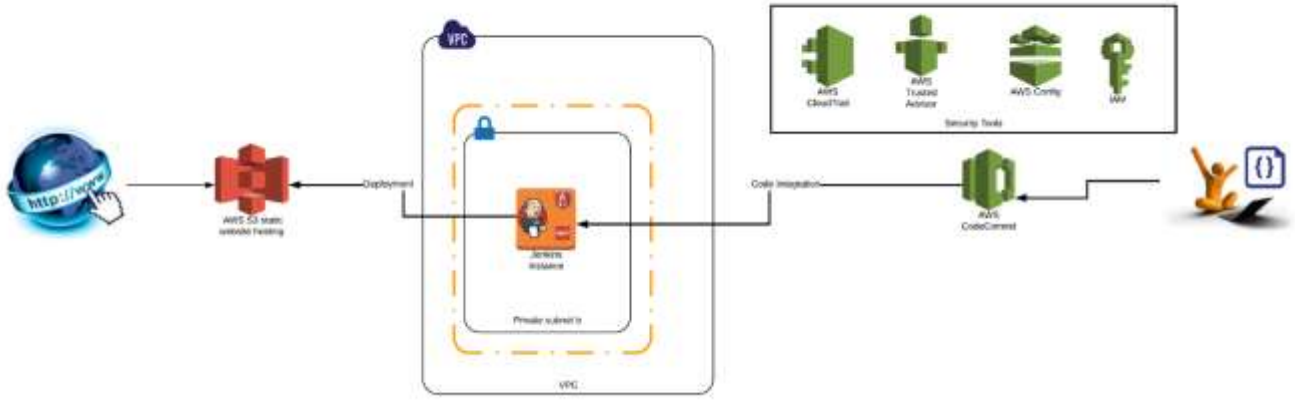
Code Repository: AWS CodeCommit
Build: Jenkins 2.7.9 using git and maven build plugin
Deployment: AWS S3 bucket static website hosting

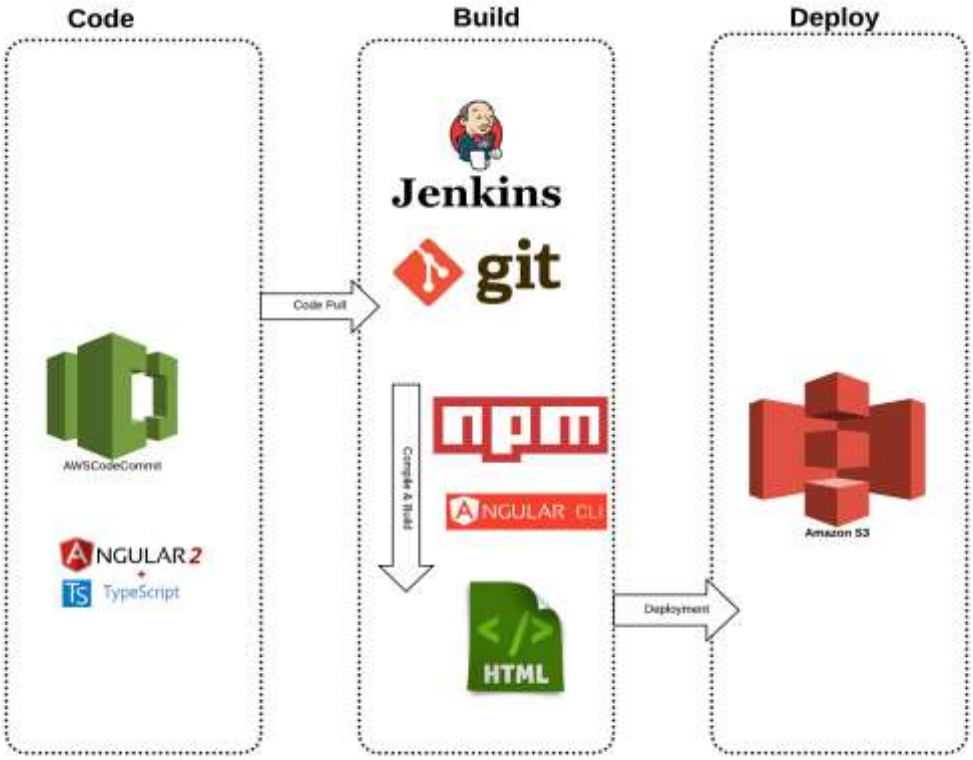
CI Integration process steps

- 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 html files
- HTML files were transferred to S3 bucket using S3CMD
- Email notifications was sent to the team on build status

# TM DevOps – Use Case

**CI/CD Flow diagram for Angular JS**



# TM DevOps – Use Case

**Security with DevOps**

**Security Framework using AWS Services**

- **Cloud Trail** used for security audit for AWS resources deployed for this customer application hosting on AWS Cloud with DevOps platform
- **Trusted advisor** used for security check and monitor for any resources status changes and optimizing infrastructure
- **IAM** used as most important part of this implementation for security authentication with AWS EC2, S3 buckets, AWS Code Commit using user and roll based policies.
- **VPC Flow logs** configured to log the complete traffic flow and security analysis, audit and future tracking purposes

**Continuous Integrations (Code)**

Knowledge crystals dev team's Code repositories were migrated from SVN to AWS CodeCommit as private code repositories. AWS CodeCommit by default does not allowed any developers to push changes without authentication. So we integrated HTTPS authentication to make developers to push changes on AWS CodeCommit private code repositories.

**Code Pull for Jenkins Build**

Code Pull for Jenkins builds gets authenticated using https git credentials using AWS cli commands and Jenkins used saved AWS credentials for each and every build code pull

**Continuous Delivery (Code Merge)**

Here in this use case Knowledge Crystals has various environment i.e. QA, UAT, PIT, SIT, Staging and Production and various respective branches. Code merges from each of these environment to another environment are securely controlled using Jenkins Authorize Project plugin .Knowledge Crystals needs to keep a track of each builds specifically and get notified, we have enabled build tracking logs and E-Mail notifications to send notifications when builds are failed with all relevant information

**Continuous Deployment**

Case deployments were/are securely authenticated using AWS access key and secret keys for deployment on AWS EC2 instances for Jenkins deployment, AWS CLI commands running 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**

**Saves Time**

- Knowledge Crystals used to run repeated builds before implementation of CI and CD, around 10-15 builds on daily basis.
- Post CI and CD implementation, repeated/deplulicated builds were prevented and this saved a lot of time
- Also CI and CD helps Knowledge crystals to prevent wrong code commits as it is integraated and automated for each and every build
- Continuous Integration saves time for Knowledge Crystals on code commit, merge, pull and prevents any accidental pushes to wrong repositories and branches
- Continuous delivery also saves time for Knowledge Crystals as developer need not spend time on manual build and log tracking and can spend time only on build error fixes.
- Continuous deployment saves time of for Knowledge Crystals - IT Ops Manager/System Admins as there is no need of manual intervention during deployments
- CI/CD implementation enabled automated email notification so that for Knowledge Crystals developers and IT Ops team are informed of each and every build sand deployments, Dev and Ops Team get notified on the builds and deployments stayis, this saves lot time for Dev and Ops team from manual tracking
- Faster build time when compared to manuall build and deployment
- Knowledge Crystals 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
- Knowledge Crystals  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 Knowledge Crystals Ops team to stream line deployments  on appropriate targets resources

**Portability**

- Builds and deployments can be portable upon build success, each and every successful builds can be deployed from one environment to another environment easily on any target environments
- Better build release control makes sure that same working version are deployed to all customer environments and this avoids unnecessary conflicts and incorrect version deployments