|  |
| --- |
|  |
| PCB DevOps Platform |
| High Level Design Document |
|  |
|  |
| **24/06/2015** |

Table of Contents

[Introduction 3](#_Toc423439739)

[Objectives of PCB DevOps Platform 3](#_Toc423439740)

[Document - Scope 3](#_Toc423439741)

[Platform High Level Design 4](#_Toc423439742)

[Software Development Lifecycle using DevOps Platform 7](#_Toc423439743)

[DevOps Platform – Infrastructure Requirements 14](#_Toc423439744)

# Introduction

At Barclays PCB, current development lifecycle is suboptimal from both an efficiency and quality perspective resulting in delays and avoidable production incidents due to poor tooling adoption, inconsistent tool standards across the organisation, lack of automation, reduced environment availability and weak configuration and quality management controls.

PCB DevOps program is one of the several initiatives to address the above mentioned concerns. PCB DevOps program is part of the overall Change Optimization Program that has a mandate to drive productivity increase through Agile Adoptions, DevOps tooling and automation.

# Objectives of PCB DevOps Platform

Here are the key objectives for creating PCB DevOps Platform:

* Reduce time and effort needed for deployment and release activity by DevOps tools automation.
* Improve quality of code by testing more often during the SLDC
* Standardize and improve predictability of releases by automating release processes.
* Reduce time and effort to commission new environment, reduction in infra footprint

# Document - Scope

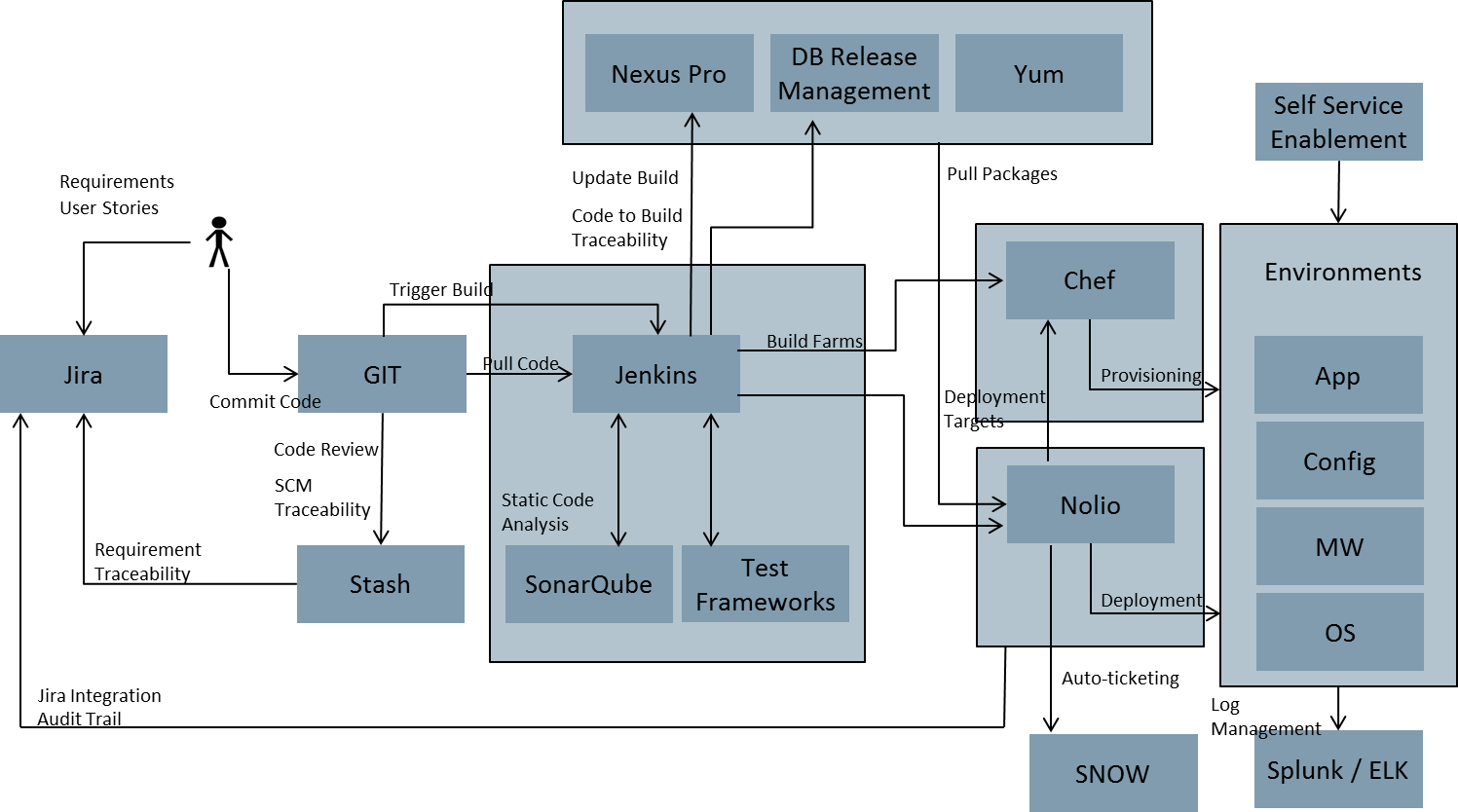
This design document addresses following details:

* Set of recommended tools for DevOps Platform
* Tools chain integration and workflow in SDLC

This document will not have details on how to install/configure the individual tool.

# Platform High Level Design

Below diagram explains high level the building blocks and recommended tools for PCB DevOps Platform.



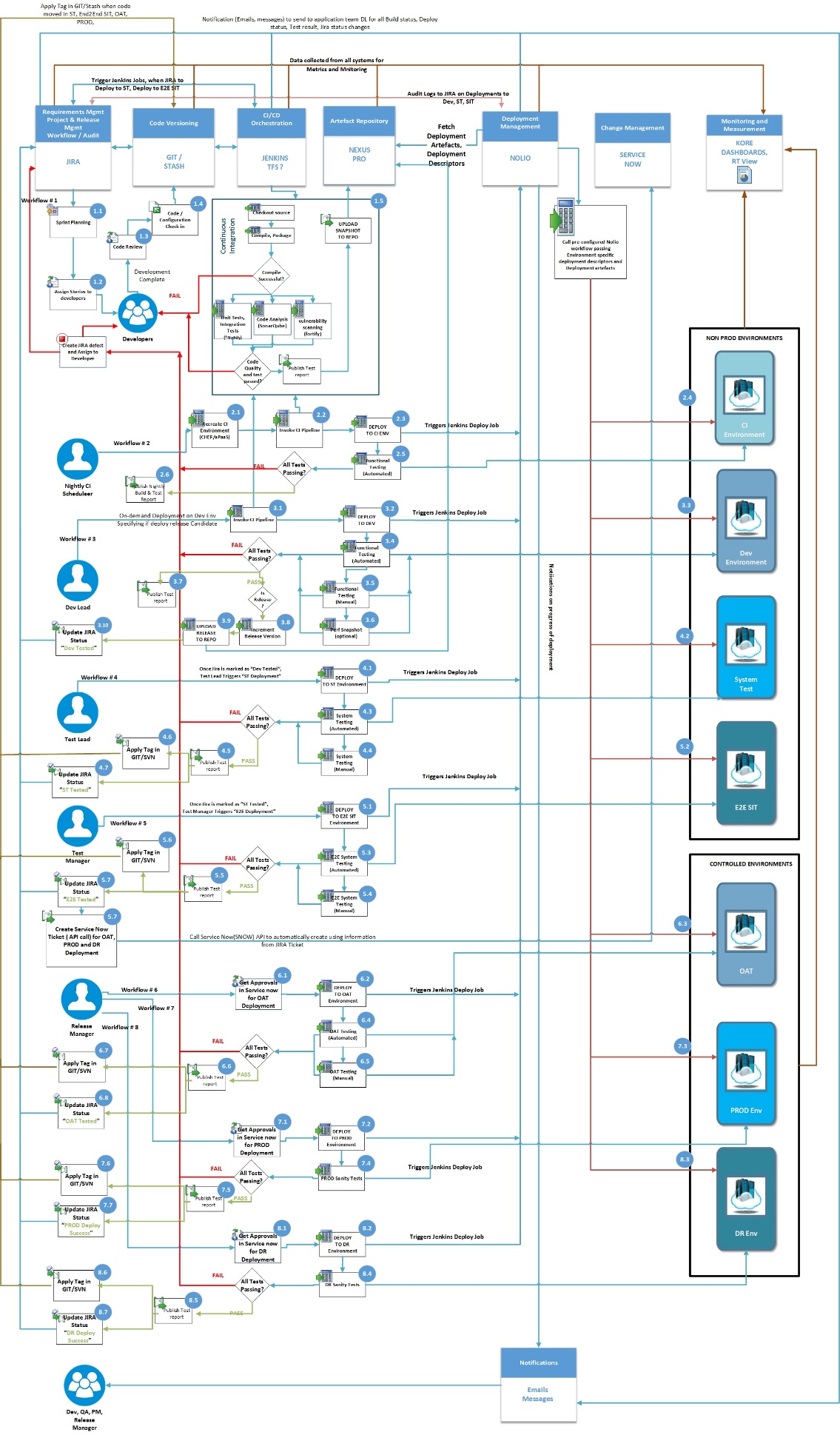
* **JIRA –** Jira is issue tracking systems and comes with flexibility of ease of defining customer workflows and API support, which makes it easy to integrate with other system. JIRA is also recommended as go to tool by “Agile Adoption”. JIRA would be used for:
  1. Requirement Management
  2. Sprint and Release Plans
  3. Sprint and Release tasks tracking
  4. Defect Tracking
  5. Requirement Traceability
  6. Audit Trails for Release activities
* **Source Control Management (SCM)** – Git/Stash is recommended as preferred tool for Source control for DevOps Platform. It would be recommended for application teams to move to GIT/Stash to get all the benefits of DevOps platform, however it will not be a blocker for onboarding applications to DevOps platform. Source control will be used for storing following:
  1. Application Code
  2. Application Configurations
  3. Environment Configurations (?)
  4. DB scripts
  5. Any customer build scripts

The platform will interact with SCM using plugins and pull necessary code, scripts and config as needed.

* **Jenkins** – Jenkins would be used as an overall orchestration engine. This is the heart of the DevOps Platform, which does all the activities including build, Unit test, Integration test, code quality checks, deployments and execute functional tests. Jenkins will be used here due to the stability and readily available plugins to integrate with the other systems. Jenkins will be deployed in a master slave mode in order to distribute the load among the different jobs.
* **SonarQube** –SonarQube will be integrated with Jenkins to do plugin in order to enforce quality standards. SonarQube will be used for:
  1. Static Code analysis
  2. Find Bugs
  3. Code Quality Gates- build are failed in case # of code violation are above threshold, or if Unit test coverage is below threshold.
* **Nexus Pro** – Nexus Pro would be used as to version deployable artefacts. There are 2 types of repository available in Nexus
  1. Snapshot repository – This would contain the build done during development phase. Every successful checking (passing, unit tests and code quality checks) would result in a snapshot version. Since there will be lot of snapshot version generated during development cycle, it’s recommended to have a Nexus policy to keep certain number of builds in repository.
  2. Release Repository – Once development for the sprint is complete, Development lead would release candidate build. Release build result in to incrementing Release version and store the generate artefacts in Release Repository. Once the release build is stored in Nexus repository, all further deployments to ST, SIT, OAT, PROD would be done using the release build from Nexus.
* **Nolio –** Nolio would be used for deployment to all the different environment. Depending on the application technology stack a Nolio template would be defined (or reused) and associated for each application. Application teams would create deployment manifests for each environment and those would be checked in as part of source code. During deployment process Nolio would pick up deployment manifests for the environment, and deployment template to be used for deployment and perform the deployment according to the workflow defined in the template and applying values from manifest file.

# Software Development Lifecycle using DevOps Platform

Below diagram explains how different workflows on SDLC would be implemented in PCB DevOps Platform.



* **Workflow # 1 Continuous Integration –** This workflow represent a development phase of SDLC. Development teams would develop code based on user stories assigned to them in JIRA. Once the Code is checked in the GIT/SVN Jenkins would trigger a Continuous Integration pipeline to get immediate feedback on the check-in to GIT. Continuous Integration pipeline would Checkout code, Compile, run Unit tests, Code analysis. Integration tests would not run as part of Continuous Integration, as they are time consuming and would result in delay in completion of CI builds. Any failures in any of these steps would trigger notification to developer (and other folks) who made the code changes, that has caused the build/test failures.
* **Workflow # 2 Nightly Builds –** Nightly build would be setup for all the application. Night build would carry following tasks in addition to CI workflow
  1. Create CI environment from scratch. This would be done using automated process to create such environment using Chef Recipes or by using aPaaS infrastructure.
  2. Along with Unit tests, also run Integration tests.
  3. Once the code is build, it will be deployed on the CI environment and automated functional tests are executed.
  4. Publish test reports to application teams
  5. Any failure would result in creation of Jira issue and would be assigned to Dev Lead to resolve.
* **Workflow # 3 Deployment to Dev –** This workflow would allow development teams to do on-demand deployments on the development environment. Typically once 1 or more User stories are completed and checked in, Dev Lead would initiate Deployment to Development environment. This pipeline would trigger:
  1. CI pipeline,
  2. Create a snapshot version of artefacts in Nexus
  3. Deployment to Dev using Nolio
  4. Execute automated tests
  5. Execute manual tests

**Release Candidate Builds:**

Once all the user stories are completed and application is ready to push to ST, Dev Lead would initiate the deployment indicating this as a Release candidate in the Jenkins pipeline. The release candidate build would increment the release version number create new version in the Nexus release repository. This point onwards all the deployment to ST, End2End SIT, OAT and Prod would be done using the release version. Any defect fixes during ST, SIT, OAT, etc., would require first testing code changes in Dev and then releasing them to other environment.

At the end of successful testing in Development cycle, JIRA ticket would be updated with status “Dev Tested”.

* **Workflow # 4 Deployment to ST -** Once theapplication is tested in Development environment, Test Lead would initiate Jenkins job to initiate the deployment to ST Environment. Here are the activities performed by Jenkins job
  1. Trigger Nolio deployment job to initiate deployments to ST Environment.
  2. After successful deployment of code, Jenkins would initiate automated function tests against ST Environment
  3. Once automated test and manual tests, test report would be publish
  4. Tag “Release x.x.z ST Tested” is applied on the code base
  5. Jira ticket would be updated “Release x.x.z ST tested”
* **Workflow # 5 Deployment to End 2 End SIT -** Once theapplication is tested in ST environment, Test manager would initiate Jenkins job to initiate the deployment to End 2 End SIT Environment. Here are the activities performed by Jenkins job
  1. Trigger Nolio deployment job to initiate deployments to End 2 End SIT Environment.
  2. After successful deployment of code, Jenkins would initiate automated function tests against End 2 End SIT Environment
  3. Once automated test and manual tests, test report would be publish
  4. Tag “Release x.x.z End 2 End SIT Tested” is applied on the code base
  5. Jira ticket would be updated “Release x.x.z End 2 End SIT tested”
  6. At the completion of End 2 End SIT Tests a SNOW tickets for OAT, PROD and DR are created by Jenkins workflow. Jenkins would call SNOW API sending the information already captured in JIRA tickets.
* **Workflow # 6 Deployments to OAT –** Once the SNOW tickets for OAT deployment are approved Release manager would initiate Jenkins job to deploy to OAT Environment. After successful deployment and passing sanity tests, Jenkins workflow would:
  1. Tag “Release x.x.z OAT Tested” is applied on the code base
  2. Update Jira ticket with “Release x.x.z OAT Tested”
* **Workflow # 7 Deployments to PROD –** PostOAT deployment and SNOW tickets for PROD deployment are approved Release manager would initiate Jenkins job to deploy to PORD Environment. After successful deployment and passing sanity tests, Jenkins workflow would:
  1. Tag “Release x.x.z PROD Release” is applied on the code base
  2. Update Jira ticket with “Release x.x.z PROD Release”
* **Workflow # 8 Deployments to DR –** PostPROD deployment and SNOW tickets for DR deployment are approved Release manager would initiate Jenkins job to deploy to DR Environment. After successful deployment and passing sanity tests, Jenkins workflow would:
  1. Tag “Release x.x.z DR Release” is applied on the code base
  2. Update Jira ticket with “Release x.x.z DR Release”

Below table shows different activities involved in the SDLC with DevOps Platform enablement and corresponding owner for the activities.

|  |  |  |  |
| --- | --- | --- | --- |
| Step # | Activity Description | Manual / Automated | Activity Owners |
| 1.1 | Setup backlog, Sprint Planning in Jira, Creation of Master JIRA ticket for Sprint/Release tracking | Manual | Scrum Master/ProductOwner |
| 1.2 | Assign User Stories and Tasks to developers | Manual | Developers/Leads |
| 1.3 | Code Review using tools like Crucible or manual | Manual | Tools team |
| 1.4 | Code checked in to GIT/Stash | Manual | Developers |
| 1.4 | Pre-commit hooks on GIT/SVN to validate if code check-in have valid JIRA ticket mentioned in comments | Automated | DevOps platform team + Tools Owner for GIT/SVN |
| 1.5 | Setup Jenkins jobs, security, plugins etc | Manual | DevOps platform team |
| 1.5 | Build scripts, library required for compiling/packaging application |  | Dev Lead |
| 1.5 | Provide unit tests, Integration tests, |  | Developers |
| 1.5 | Sonar setup for code analysis, quality gates, test coverage | Automated | DevOps platform team |
| 1.5 | Upload generated deployable artifact to Nexus repository | Automated | DevOps platform team |
|  |  |  |  |
| 2.1 | Setup nightly Jenkins job for application |  | DevOps platform team |
| 2.1 | Drop previously created CI environment and create new CI environment provisioning for application |  | DevOps platform team + tools owner |
| 2.2 | Jenkins CI pipeline for nightly builds | Automated | DevOps platform team |
| 2.3 | Trigger deployment to CI environment | Automated | DevOps platform team |
| 2.4 | Deploy to CI environment using Nolio | Automated | DevOps platform team + tool owner |
| 2.5 | Create Automated Functional tests | Manual | Testing Team |
| 2.5 | Trigger Automated Functional tests on CI env | Automated | DevOps platform team |
| 2.6 | Publish Test results | Automated | DevOps platform team |
|  |  |  |  |
| 3.1 | Initiate on-demand build in Jenkins to deploy to Dev Environment, specify if this is a release candidate build | Manual | Dev Lead |
| 3.2 | Trigger deployment to Dev environment | Automated | DevOps platform team |
| 3.3 | Deploy to Dev environment using Nolio | Automated | DevOps platform team + tool owner |
| 3.4 | Create Automated Functional tests | Manual | Testing Team |
| 3.4 | Trigger Automated Functional tests on Dev env | Automated | DevOps platform team |
| 3.5 | Execute manual functional tests | Manual | Test Lead |
| 3.6 | Execute test to get application performance snapshot ( Optional) | Manual | Test Lead |
| 3.7 | Publish Test results | Manual + Automated | Test Lead |
| 3.8 | For Release candidate build once the build and tests are successful, increment release version | Automated | DevOps platform team |
| 3.9 | Upload artifact to release repository | Automated | DevOps platform team |
| 3.10 | Updated JIRA ticket status to “Dev Tested” | Automated | DevOps platform team |
|  |  |  |  |
| 4.1 | Initiate on-demand build in Jenkins to deploy to ST Environment | Manual | Test Lead |
| 4.1 | Trigger deployment to ST environment | Automated | DevOps platform team |
| 4.2 | Deploy to ST environment using Nolio | Automated | DevOps platform team + tool owner |
| 4.3 | Create Automated Functional tests | Manual | Testing Team |
| 4.3 | Trigger Automated Functional tests on Dev env | Automated | DevOps platform team |
| 4.4 | Execute manual functional tests | Manual | Test Lead |
| 4.5 | Publish Test results | Manual + Automated | Test Lead |
| 4.6 | Apply Source code with Tag “ST Tested” | Automated | DevOps platform team |
| 4.7 | Updated JIRA ticket status to “ST Tested” | Automated | DevOps platform team |
|  |  |  |  |
| 5.1 | Initiate on-demand build in Jenkins to deploy to End 2 End SIT Environment | Manual | Test Manager |
| 5.1 | Trigger deployment to SIT environment | Automated | DevOps platform team |
| 5.2 | Deploy to SIT environment using Nolio | Automated | DevOps platform team + tool owner |
| 5.3 | Create Automated Functional tests | Manual | Testing Team |
| 5.3 | Trigger Automated Functional tests on end to end SIT env | Automated | DevOps platform team |
| 5.4 | Execute manual functional tests | Manual | Test Lead |
| 5.5 | Publish Test results | Manual + Automated | Test Lead |
| 5.6 | Apply Source code with Tag “E2E Tested” | Automated | DevOps platform team |
| 5.7 | Updated JIRA ticket status to “E2E Tested” | Automated | DevOps platform team |
| 5.8 | Once the status of JIRA ticket is changed SNOW tickets for OAT, PROD and DR deployments would be created by calling SNOW APIS from Jenkins, passing information available from the JIRA ticket | Automated | DevOps platform team + tools owner |
|  |  |  |  |
| 6.1 | Release manager would work with different teams to get approval for OAT deployments | Manual | Release manager |
| 6.2 | Initiate on-demand build in Jenkins to deploy to OAT Environment | Manual | Release Manager |
| 6.2 | Trigger deployment to OAT environment | Automated | DevOps platform team |
| 6.3 | Deploy to OAT environment using Nolio | Automated | DevOps platform team + tool owner |
| 6.4 | Create Automated OAT tests | Manual | Testing Team |
| 6.4 | Trigger Automated Functional tests on end to end SIT env | Automated | DevOps platform team |
| 6.5 | Execute manual OAT tests | Manual | Test Lead |
| 6.6 | Publish Test results | Manual + Automated | Test Lead |
| 6.7 | Apply Source code with Tag “OAT Tested” | Automated | DevOps platform team |
| 6.8 | Updated JIRA ticket status to “OAT Tested” | Automated | DevOps platform team |
|  |  |  |  |
| 7.1 | Release manager would work with different teams to get approval for PROD deployments | Manual | Release manager |
| 7.2 | Initiate on-demand build in Jenkins to deploy to PROD Environment | Manual | Release Manager |
| 7.2 | Trigger deployment to PROD environment | Automated | DevOps platform team |
| 7.3 | Deploy to PROD environment using Nolio | Automated | DevOps platform team + tool owner |
| 7.4 | Execute Sanity tests on PROD | Manual | Test Lead |
| 7.5 | Publish Test results | Manual | Test Lead |
| 7.6 | Apply Source code with Tag “PROD Deployed” | Automated | DevOps platform team |
| 7.7 | Updated JIRA ticket status to “PROD Deployed” | Automated | DevOps platform team |
|  |  |  |  |
| 8.1 | Release manager would work with different teams to get approval for DR deployments | Manual | Release manager |
| 8.2 | Initiate on-demand build in Jenkins to deploy to DR Environment | Manual | Release Manager |
| 8.2 | Trigger deployment to DR environment | Automated | DevOps platform team |
| 8.3 | Deploy to DR environment using Nolio | Automated | DevOps platform team + tool owner |
| 8.4 | Execute Sanity tests on DR | Manual | Test Lead |
| 8.5 | Publish Test results | Manual | Test Lead |
| 8.6 | Apply Source code with Tag “DR Deployed” | Automated | DevOps platform team |
| 8.7 | Updated JIRA ticket status to “DR Deployed” | Automated | DevOps platform team |

# DevOps Platform – Infrastructure Requirements

Here is the indicative list of server requirements for DevOps Platform. These requirement will be fine-tuned along with tools SMEs for scalability and performance of DevOps platform.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. NO** | **Tools** | **No of Nodes** | **Operating System** | **Hardware Description** |
| 1 | Jenkins Master with replica | 2 | Linux | 2 CPU Cores with 8GB of RAM and at least 300GB |
| 2 | Jenkins Slaves | 2 | Linux | 2 CPU Cores with 8GB of RAM and at least 300GB |
| 3 | Jenkins Slaves | 2 | Windows | 2 CPU Cores with 8GB of RAM and at least 300GB |
| 4 | Sonar | 1 | Linux | 2 CPU Cores with 8GB of RAM and at least 300GB |
| 5 | Jira | 1 | Linux | 2 CPU Cores with 8GB of RAM and at least 300GB |
| 6 | Git/Stash | 1 | Linux | 2 CPU Cores with 8GB of RAM and at least 300GB |
| 7 | Nexus | 1 | Linux | 2 CPU Cores with 8GB of RAM and at least 300GB |
| 8 | Nolio |  |  | (Existing installation in Barclays) |
| 9 | Chef | 1 | Linux | 2 CPU Cores with 8GB of RAM and at least 300GB |
| 10 | Application specific Test Beds | 2 | Linux | 2 CPU Cores with 8GB of RAM and at least 300GB |
| 10 | Application specific Test Beds | 2 | Windows | 2 CPU Cores with 8GB of RAM and at least 300GB |
|  | Total | 15 |  |  |