An Arrow Company

ENRICHING LIVES THROUGH ENGINEERING EXCELLENCE

DevOps Maturity Assessment Report

Contents

1. Introduction

This document provides a sample of DevOps assessment carried out for a billion dollar telecom customer with IoT home security platform. This assessment is the first step towards becoming more agile and embracing the DevOps philosophy and the related practices and tools.

2. DevOps Assessment Framework

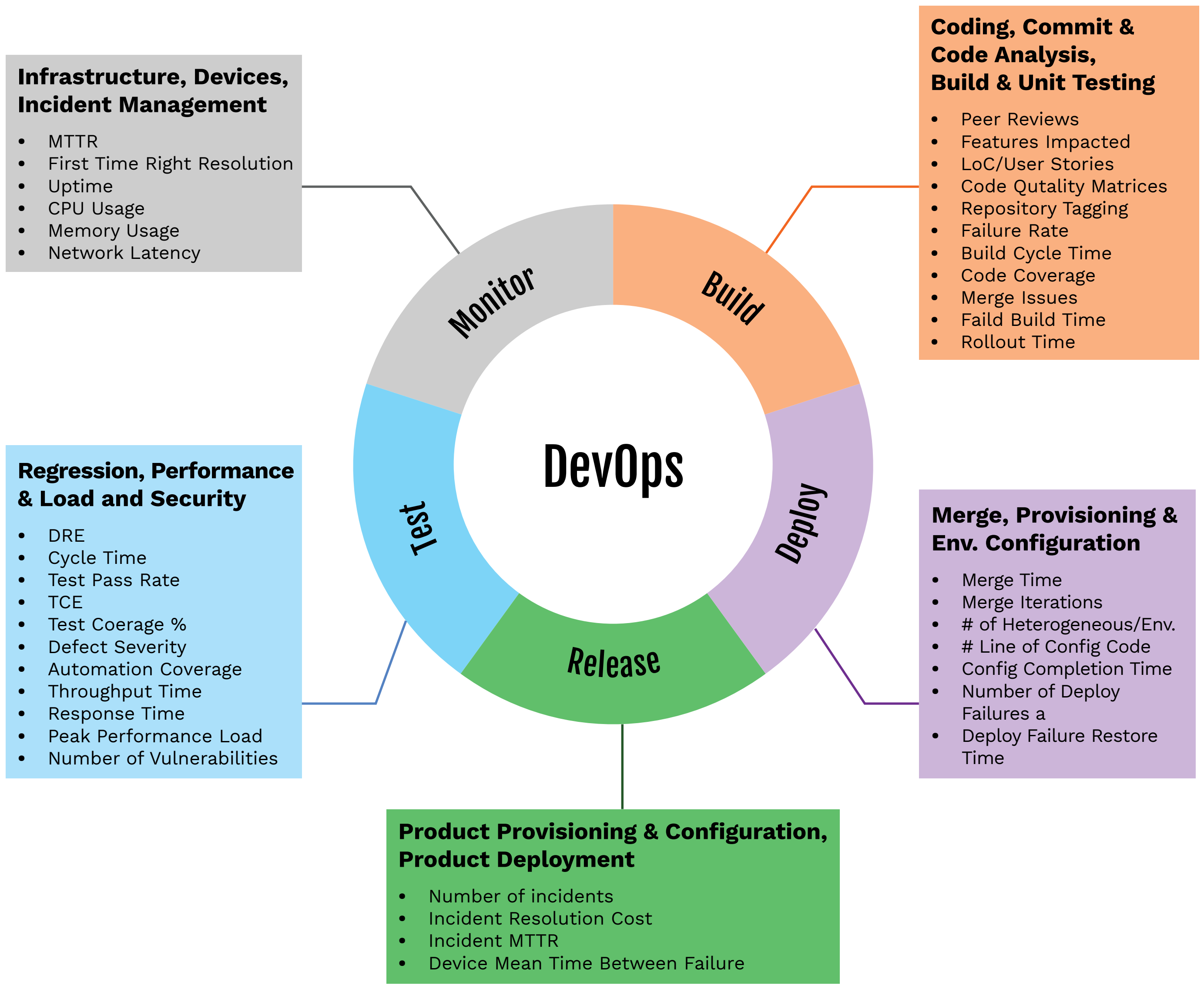
The assessment is based on 100+ code to release metrics across the development life cycle comprising of Build, Deploy, Release, Test and Monitor stages. Each stage comprises of set of questions to assess the DevOps maturity of that particular stage and provide suitable recommendations to improve and move to the next maturity level.

DevOps maturity model describes five levels of maturity which include Initial, Managed, Defined, Measured and Optimized which is in line with CMMi maturity model.

Each level depend on three dimensions:

- Process maturity
- Automation maturity
- Collaboration maturity

Each maturity level is described as the combination of these three dimensions. In order to progress from one level to another it is essential to improve in all three dimensions.



The definition of each level is as mentioned below:

DevOps Maturity Level	Build	Deploy	Release	Test	Monitor
L1- Initial	• Manual process for Build	• Manual process for Environment provisioning • Manual deployment of Build	• Unplanned releases • Major release failures specifically on lower environments	• Manual Testing by Test Team	• Manual monitoring of Infrastructure and Applications using market available tools • No standardization for Alert setup
L2- Managed	• Static Code Quality Analysis • Regular automated Build • Any Build can be recreated from Source Control	• Environment setup optimized to be more cost effective. • Automated Build deployment to Lower Environments • Configurations are versioned	• Reliable unplanned releases • Minimal traceability with requirements / backlog	• Regression test suite automated and executed with each release	• Alert setup in place • Turnaround time for Alerts on higher side • Requires field visit for Device Management
L3- Defined	• Automated Static code quality analysis • Automated Build and Test every time Code is checked in to Master Branch. • Dependencies are managed. • Re-usable scripts for deployment on multiple environments • FMEA in place	• Environment setup process is consistent for all environments • Environment configuration & setup fully automated • Rollback process defined in case of failures	• Traceability with requirements in place • Change Management process defined and tracked using Change Management Tools • Tracking of Customer specific issues in place	• Test team writing CDT (Core Development Tests) and CAT (Core Acceptance Tests) • Development team executes CDT • Test team executes CAT • Automated Test results update in test management tool and shared via email	• Well defined Alert setup • Scheduling of resources for 24/7 monitoring • Escalation procedure in place • Collaboration of Ops and Dev teams for RIM (Rapid Incident Management) • Automated Remedial actions as much as possible • Remote Device Management
L4-Measured	• Build Metrics collected and shared • No Broken Builds	• Environment setup & configuration metrics gathered and shared • Release and rollback tested and automated	• Release cycle time monitored • Service level agreements with Customers	• Quality Metrics Trends tracked	• Metrics Trends monitored Incident Management
L5-Optimized	• Discuss and resolve integration problems using automation • Faster feedback • Continuous improvement by reducing Build time	• Environment provisioning fully automated • Virtualization used if applicable • Discuss and resolve integration problems using automation	• Optimize release cycle time • Operations and Development teams collaborate to devise Release plans to reduce cycle time • Ability to provide frequent releases for high priority unplanned requirements	• Almost all testing is automated, also for non-functional requirements. • Testing of infrastructure code. • Health monitoring for applications and environments and proactive handling of problems.	• Reduce Incident resolution time • Automate Incident resolution

3. DevOps Assessment for Home Automation customer

3.1 Challenges

The client is a leader in connected home technology solutions that enable easy to manage

Solutions for home automation, security, and energy. The IoT home security platform comprises of 2 million edges and 100+ servers. Each device is connected to 80+ sensors via Zigbee and Wi-Fi. A team of 70 engineers with different skill sets across firmware, software, API and QA work in a multi-site agile mode. The client was facing the challenge of integrating multiple development pipelines on different technologies causing longer release cycles and more manual effort in terms of generating builds, setup of virtual machines, environments, build, deployments, verification and functional testing of each release. There were deployment challenges and multiple product pipelines and 31 delivery endpoints in terms of web, mobile, device, server etc.

3.2 Application Summary

eInfochips team assessed the following applications and databases as part of a two week assessment:-

Home Automation & Security Platform

- Remote Device Management Application
  - Management Portal
  - Subscriber portal
- Home security mobile application
- Cluster Location Service
- History Service Application
- Alarm Service Application
- Account Migration Application

Database:

- Oracle
- Cassandra
- Dynamo DB
- MySQL

3.3 DevOps Assessment As-Is Process and Outcome

- eInfochips DevOps Solution Architect worked with the customer to gather inputs on each of the Build, Deploy, Release, Test and Monitor stages for two weeks and detailed the DevOps As-Is process for each phase using eInfochips DevOps Assessment Framework.
- In the second phase, eInfochips DevOps Solution Architect and Technical Lead worked for four weeks and came up with DevOps To - Be Process and Detailed Plan for DevOps adoption including People, process, tools, governance and metrics.
- The plan was accepted and implemented by the customer. The benefit/ROI was tracked along with the customer after seven months and after the new process was implemented leading to significant benefits listed below.

3.3.1 Build Assessment

Stages	Process/ Activities	Current state	Current Tools	To- Be Process	Proposed Tools	Metrics to be measured	Benefit/ROI (after seven months of implementation)
	Coding/ Code Changes	• Continuous Integration tool is not used in current setup • Few teams create manual builds, while for others the build process is automated	Java SVN Junit ANT Gradle	• Use Continuous integration tool like Bamboo to create a CI/CD pipeline • Automate build process	Bamboo Maven Git JSHint Coverity CodePro FishEye Bamboo Java SVN Junit ANT Gradle	# of Peer Reviews, # of Features Implemented, #Major Issues fixes, # Blocker issues fixes, # Minor issues fixes, LoC/ User stories	• Reduction in build cycle time by 25% • Continuous BVT cycle time 35% using automation
	Commit on Repository	• Version control software is used but builds are occasionally recreated		• Use version management tools to restore previous versions or taking backup		# of Repository Tagging	
	Code Analysis	• Automated Unit tests and code analysis partially exists		• Use static code analysis tools		Code Quality Matrices	
Build L2 Managed DevOps Maturity level	Build	• Builds are deployed manually • No Auto build break-up trigger based on predefined event (e.g. compile error, static code analysis error, build verification test error) • Feedback on build failures are provided based on resource availability normally within few hours • No metrics are captured in current scenario • Progress Tracking is not done on daily basis • Teams are currently working in silos		• Automate Build verification testing process to help in risk reduction in early stages of development lifecycle • Align teams to manage dependencies and implement Failure mode and effect analysis to evaluate the build process and impact of failures • Identify and track Build metrics • Track progress on daily basis • Share learnings and best practices to help inculcate continuous improvement culture in team		# of Failure Rate, Build Cycle Time, # of Unit Tests, Code Coverage, Full Build Time	

3.3.2 Deploy Assessment

Stages	Process/ Activities	Current state	Current Tools	To- Be Process	Proposed Tools	Metrics to be measured	Benefit/ROI (after seven months of implementation)
Deploy L1 Managed DevOps Maturity level	Code Merge	• Deployment is done manually • Different environment configuration versions are not maintained	Java SVN Junit ANT Gradle	• Build Deployment Automation and CI integration • Implement Auto environment configuration and version management to ensure production is always updated with latest code	Git Ansible Maven Python vcloud/vsphere Java SVN Junit ANT Gradle	Merge Time, Merge Iterations, Modules, User Stories, Tickets, # Full build time, Merge/Build Issues, Build Mean Time to Resolve	Reduction in Deployment efforts in production upto 65%
	Master Merge & Build on Prod	• Deployment process is not integrated with continuous integration tool • Manually version needs to be updated in case of deployment failure		• Use Continuous deployment and integration tool for environment configurations and deployment automation • Implement Auto deployment roll back for failures, notification, and escalation management		# of Heterogeneous/ Env, # Line of Config Code, Config Completion Time, # Configuration Failure, Config Mean Time to Resolve, # of Config changes	
	Configuration & Deployment in Prod (Nodes & servers)	• Deployment process / framework is not reusable in other projects • Deployment metrics data are not monitored, measured or analysed		• Start developing reusable deployment components leading to a reusable deployment process to ensure faster time to market • Keep all team members aware of 'Definition of Done' to eliminate cost of rework due to different interpretations of same requirement • Identify and track deployment metrics		Deployment Time to Target, # of failed deployment, Deploy Fail time, Deploy Mean Time to Resolve, Deployment Success Rate, # of incidents, # of Incident Resolution Cost, # of Incident MTTR, Mean Time between Failure (MTBF), # of Hetro Env Nodes	

3.3.3 Release Assessment

Stages	Process/ Activities	Current state	Current Tools	To- Be Process	Proposed Tools	Metrics to be measured	Benefit/ROI (after seven months of implementation)
Release L3 Defined DevOps Maturity level	Release Management	• Release management tools are being used for Agile implementation • Some of the agile implementation is in place (daily stand-up meetings, scrum meetings etc.) • Some of the features and backlog are maintained but there is no process/ prioritization • Some of the metrics related to release are captured • Release automation process is not environment agnostic • Sprint retrospective not performed • No clear goals to optimize release cycle time	Rally	• Maintaining a product backlog will provide you with the product vision, engagement tool with the stakeholders and finished product tracking • Start identifying and tracking metrics like MTTR (Mean time to resolve), # of major/ minor releases, # of hot fixes to continuously improve release management process • Sprint retrospective at the end of each sprint helps in identifying what went right and what went wrong during the sprint • Breaking down the goal to optimize release cycle time in specific metrics and tasks will help align the team better to the goal	Rally JIRA Bamboo	MTTR (Mean time to resolve), # of major / minor releases, # of hot fixes maintained	Reduction in release cycle time upto 60%

3.3.4 Testing Assessment

Stages	Process/ Activities	Current state	Current Tools	To- Be Process	Proposed Tools	Metrics to be measured	Benefit/ROI (after seven months of implementation)
Test L2 Managed DevOps Maturity level	Test automation	• Partial regression tests are automated • Some of the agile implementation is in place (daily stand-up meetings, scrum meetings etc.) • Automation does not start based on build creation and deployment from CI • Automation framework is not reusable in other projects • Load testing is performed by simulating application service requests	Rally	• Carry out full unit test automation • Build Verification Test automation is linked to CI process • Implement edge virtualization to save regression testing infrastructure costs • Implement regression automation for end to end use-cases from sensor to cloud • Identify business critical scenarios for load testing • Use CI tools to trigger automation scripts once build is deployed to reduce wait time • Build a unified automation framework that can be reused across products/ applications • Implement automation at early stages where applicable • Implement security management tools, practices to identify vulnerabilities	Rally JIRA Jenkins Bamboo Python Robotium Selenium Appium Shell scripts Junit	#of Unit Test: Positive & negative scenario count, #DRE, #Test Time, #Test Pass Rate, Test Case Effectiveness, #TC Coverage, Defect Severity (#Major/Minor/ Blocker defects), Automation Coverage %, Test Execution Coverage, Throughput time, Response time, # of vulnerabilities, # of Peak Failure Load, # of Peak Performance Load	Reduction in test cycle time upto 60%
	Testing Process	• Test team do not discuss test strategy with operations and development team • Team is performing random testing without goals • Testing metrics are not captured		• Share best practices and lessons learnt • Build a traceability matrix to track test coverage • Measure and monitor test metrics			

3.3.5 Monitor Assessment

Stages	Process/ Activities	Current state	Current Tools	To- Be Process	Proposed Tools	Metrics to be measured	Benefit/ROI (after seven months of implementation)
Monitor L2 Managed DevOps Maturity level	Monitoring Process	• Monitoring process automated but not continuously executed • Some of the monitoring parameters are set with alert/ rule to get monitoring issues • No metrics are captured • No Predictive failure detection/ correction actions in place	In house portal for device management	• Use continuous monitoring tools and implement automation wherever applicable • Start identifying and tracking monitoring metrics • Setup Auto alert and notification for infra, app, service, database monitoring • Setup Predictive failure detection and process to take proactive corrective actions	Zabbix, Nagios, Graphite, Zendesk Slack	Mean time response, First time right resolution, uptime, MTTR, BMI, # CPU, Memory, Storage, Network Latency, Throughput time, Mean Time between Failure (MTBF), Device-side parameters (app, service, network..), Response tome / Page Load time / DTU & Limit, Failed Requests, Service Health	Reduction in Mean time to Respond upto 30%
	Incident Management	• There is no escalation person and time zone defined with process • There is no automation involved to resolve monitoring issues • No incident management process performed • No intruder detection system exists		• Define escalation management process and Opbot/Chatbot usage • Introduce Action based issue resolution • Use ChatOps tools for collaboration automation • Introduce automation for issue resolution • Use incident management tools			

4. DevOps Maturity Implementation plan

As the customer scaled from 4M devices to 22M devices over 3 years, they were able to realize the below improvements in execution metrics against each pillar of assessment/implementation.

	Week 1-6	Week 7-12	Week 13-18	Week 19-24	Week 25-30	Week 31-36
DevOps Assessment Phase 1 - As is state mapping	2 weeks					
DevOps Assessment Phase 2 - Assessment report, implementation plan creation	4 weeks					
Milestone 1 - Build Automation		4 weeks				
Milestone 2 - Deployment & Release Automation			8 weeks			
Milestone 3 - Test Automation				12 weeks		
Milestone 4 - Monitoring Automation						4 weeks

5. Benefits/ROI

As a result of above DevOps automation through all five stages of development life cycle, customer was able to realize year-on-year savings of $220,800. Customer was able to realize below improvements in execution metrics.

- Reduction in build cycle time by 25%
- Continuous Build Verification Testing cycle time 35% using automation
- Reduction in Deployment efforts in production upto 65%
- Reduction in release cycle time upto 60%
- Reduction in test cycle time upto 60%
- Reduction in Mean time to Respond upto 30%