



Evaluation of Infrastructure as Code (IaC) Approaches for Automated Provisioning and Configuring of IoT Devices in Smart Factory Environments

W. K. N. Weerasekara
(Reg. No.: MS23046566)

A THESIS
SUBMITTED TO
SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE IN INFORMATION TECHNOLOGY SPECIALISATION IN
ENTERPRISE APPLICATION DEVELOPMENT
(TECHNOLOGY/MANAGEMENT/SYSTEM)

December 2024

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Dr. Dharshana Kasthurirathna

Approved for MSc. Research Project:

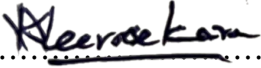
MSc. Programme Co-ordinator, SLIIT

Approved for MSc:

Head of Graduate Studies, FoC, SLIIT

DECLARATION

This is to certify that the work is entirely my own and not of any other person, unless explicitly acknowledged (including citation of published and unpublished sources). The work has not previously been submitted in any form to the Sri Lanka Institute of Information Technology or to any other institution for assessment for any other purpose.

Sign: 

W. K. N. Weerasekara

Date: 10/11/2024

ABSTRACT

Evaluation of Infrastructure as Code (IaC) Approaches for Automated Provisioning and Configuring of IoT Devices in Smart Factory Environments

Kasun Weerasekara

MSc. in Information Technology Specialisation in Enterprise Application Development

Supervisor: Dr. Dharshana Kasthurirathna

December 2024

The Internet of Things (IoT) area is gaining with time and predictions are showing that Industrial IoT (IIoT) will gain more and more in the future. Here this research was done to find out the best Infrastructure as Code (IaC) tool from model-driven, Terraform and code-centric Ansible for automatic configuring and provisioning IoT devices in large-scale IIoT systems such as automated factory environments.

This research has shown the use of IaC within IIoT to automatically provision and configure components of the IIoT system alongside improving productivity, less human involvement in provisioning and configuring components, minimising the errors in device provisioning, cost-effectiveness with increased portability and maintainability of the large scale IIoT system with the benefit of the IaC. Furthermore, the research assessed Terraform and Ansible by analysing the elapsed time, resource utilisation, scalability, and error rate, in provisioning and configuring as well as reconfiguring using a prototyped simulated environment for a factory. Also, the research is contributing to the design and development of a cross-platform IaC script generation and execution application including the monitoring capabilities. This tool is named “KFactory Device Provisioner and Configurator”. This application allows to generation of IaC provisioning scripts and executes those with monitoring capabilities as users’ need via a Graphical User Interface (GUI). The tool also has a system monitoring tool that is very helpful to view the variation of CPU usage, Memory usage and inbound-outbound Network usages in a GUI.

Furthermore, the tool also collects the provision data to create a Machine Learning (ML) model to predict and show the expected provisioning time, reconfiguring time, CPU, Memory and Network

inbound and outbound usages according to the scale of the provisioning tasks based on the host system's capabilities.

Moreover, with the conclusion of this research, the researchers are encouraged to come up with a fine-tuned, production-grade IaC solution for automatic provisioning and configuring IoT devices in large-scale IIoT systems with reduced deployment time, optimized resource utilizations, having scalability, having low error rate, and reduced reconfiguring time.

ACKNOWLEDGEMENT

While at the Sri Lanka Institute of Information Technology, I have benefited from having great advisors who seem to agree about very little. Dr Dharshana Kasthurirathna was a great mentor, providing advice, constant constructive criticism of my ideas and writing, access to his web of contacts and friends, financial support, and the freedom to work on my own projects on his research account's time.

Furthermore, I would like to thank my father, mother and my spouse for supporting me to find the time and focus on the research alongside a lot of work and dedication.

TABLE OF CONTENTS

DECLARATION	ii
ABSTRACT	iii
ACKNOWLEDGEMENT	v
TABLE OF CONTENTS	vi
List of Figures	xii
List of Tables	xvi
Chapter 1 Introduction	1
1.1 Justification for the Needfulness of this Type of Comparison Using a Survey	3
Chapter 2 Literature Review	8
2.1 Automatic Provisioning and Configuring IoT Devices	8
2.2 Automatic Provisioning and Configuring IoT Devices Using Infrastructure as Code (IaC) Principles	12
2.3 Defects and Quality Measures of Infrastructure as Code (IaC)	14
2.4 Challenges and Emerging Tech in IoT Configurations of Smart Factories	16
2.5 Similar Implementations	17
Chapter 3 Research Problem and Conceptual Framework	19
3.1 Research Gap	19
3.2 Research Problem	20
3.3 Research Contribution	20
3.4 Research Questions	21
3.4.1 Research Question 1	21
3.4.2 Research Question 2	21
3.4.3 Research Question 3	21
3.4.4 Research Question 4	21
3.5 Research Objectives	21
3.6 Conceptual Framework	22
3.6.1 Use of Model-driven Tools	22
3.6.2 Use of Code-centric Tools	23
3.6.3 Other Ways to Do the Same Rather than Use IaC Tools	24
3.6.3.1 Coding from Scratch	24
3.6.3.2 Cloud Services that Provide Nearly Similar Functionality	24
Chapter 4 Methodology	26
4.1 Research Design	26
4.1.1 Qualitative Elements of the Research	27

4.1.2 Quantitative Elements of the Research	27
4.2 Simplified Methodology	27
4.2.1 Survey	28
4.2.2 Design	28
4.2.3 Implementation	28
4.2.4 Experimentation	28
4.2.5 Application	28
4.2.6 Conclusion	29
4.3 Sampling Strategy	29
4.3.1 Sampling Size	29
4.4 Data Collection Methods	30
4.4.1 Survey	30
4.4.2 Experiments	30
4.4.3 Gathering Other Required Data	30
4.5 Data Analysis Techniques	30
4.6 Ethical Considerations	31
4.6.1 For Survey	31
4.6.2 For Factory Inspections	31
Chapter 5 Prototype Application	33
5.1 Introduction	33
5.2 Plan the Implementation for the Evaluation	33
5.3 Preparation of the IoT Project Development Environment	35
5.4 Preparation of the Simulation Environment for Targetted ESP32 Devices, Dashboards and MQTT Brokers	36
5.5 Plan the Evaluations	37
5.6 Design, Implementation and Testing Iterations to Come Up with an Optimal Prototype	39
5.6.1 Pure Code-based ESP32 Simulation	41
5.6.2 Using Terraform to Run the Separate EMQX MQTT Broker and Node-RED Dashboard as Docker Containers	41
5.6.3 Using Ansible to Run the Separate EMQX MQTT Broker and Node-RED Dashboard as Docker Containers	42
5.6.4 Test the Mluis/qemu Emulator as a Container (Docker)	42
5.6.5 Try Renode for Emulation	42
5.6.6 Test the ESP32 Built-in Example on Mluis/qemu Emulator (Docker Container)	43
5.6.7 Create an ESP-IDF Sample Project and Test it on Mluis/qemu Emulator (Docker Container) ..	43
5.6.8 Using Terraform to Build and Run the Built-in Example Project of Mluis/qemu ESP32 Emulator, Separate EMQX MQTT Broker and Node-RED Dashboard as Docker Containers	43

5.6.9 Using Terraform to Run a Sample Project on Mluis/qemu ESP32 Emulator, Separate EMQX MQTT Broker and Node-RED Dashboard as Docker Containers.....	44
5.6.10 Using Terraform to Run Different Types of Projects on Mluis/qemu ESP32 Emulators, Separate EMQX MQTT Broker and Node-RED Dashboard as Docker Containers	45
5.6.11 Using Ansible to Run a Sample Project on Mluis/qemu ESP32 Emulator, Separate EMQX MQTT Broker and Node-RED Dashboard as Docker Containers.....	46
5.6.12 Using Ansible to Run Different Types of Projects on Mluis/qemu ESP32 Emulators, Separate EMQX MQTT Broker and Node-RED Dashboard as Docker Containers.....	47
5.6.13 The Optimal Prototype	48
5.6.13.1 Factory for the Optimal Prototype.....	48
5.6.13.2 IoT Device Plan.....	49
5.6.13.3 Facts Considered in Implementation of the Optimal Prototype.....	49
5.7 Experimentation	52
5.7.1 The Experimentation Plan.....	53
5.7.2 IaC Scripts for Experimentation	53
5.7.3 Initiate the Experimentation.....	53
5.7.4 The Execution of Experimentations	54
Chapter 6 Results	57
6.1 Introduction	57
6.2 The Results of Using Terraform in Provisioning and Configuring	57
6.3 The Results of Using Terraform in Reconfiguring	59
6.4 The Results of Using Ansible in Provisioning and Configuring.....	60
6.5 The Results of Using Ansible in Reconfiguring.....	61
Chapter 7 Discussion on Experimentation Results	64
7.1 Variation of the Elapsed Time in Provisioning and Configuring IoT Devices in KFactory by using Terraform and Ansible	64
7.2 Variation of the CPU Load Average in Provisioning and Configuring IoT Devices in KFactory by using Terraform and Ansible	65
7.3 Variation of the Memory Gap in Provisioning and Configuring IoT Devices in KFactory by using Terraform and Ansible	66
7.4 Variation of the Elapsed Time in Reconfiguring IoT Devices in KFactory by using Terraform and Ansible.....	67
7.5 Variation of the CPU Load Average in Reconfiguring IoT Devices in KFactory by using Terraform and Ansible.....	68
7.6 Variation of the Memory Gap in Reconfiguring IoT Devices in KFactory by using Terraform and Ansible.....	69
7.7 Variation of the Elapsed Time in Provisioning and Configuring and the Reconfiguring of IoT Devices in KFactory by Using Terraform	70

7.8 Variation of the CPU Load Average in Provisioning and Configuring and the Reconfiguring of IoT Devices in KFactory by Using Terraform	71
7.9 Variation of the Memory Gap in Provisioning and Configuring and the Reconfiguring of IoT Devices in KFactory by Using Terraform	72
7.10 Variation of the Elapsed Time in Provisioning and Configuring and the Reconfiguring of IoT Devices in KFactory by Using Ansible.....	73
7.11 Variation of the CPU Load Average in Provisioning and Configuring and the Reconfiguring of IoT Devices in KFactory by Using Ansible.....	74
7.12 Variation of the Memory Gap in Provisioning and Configuring and the Reconfiguring of IoT Devices in KFactory by Using Ansible.....	75
Chapter 8 KFactory Device Provisioner and Configurator	77
8.1 Introduction	77
8.2 Introduction to KFactory Device Provisioner and Configurator.....	77
8.2.1 The Usefulness of the Application	77
8.3 Design of the KFactory Device Provisioner and Configurator.....	77
8.3.1 Use Case Diagram	78
8.3.2 The Architecture of the KFactory Device Provisioner and Configurator	78
8.3.3 Component Diagram.....	79
8.3.4 Main Components of the KFactory Device Provisioner and Configurator.....	80
8.3.4.1 The RootApp Component	80
8.3.4.2 Script Generators	81
8.3.4.2.1 The Terraform Script Generator Component	81
8.3.4.2.2 The Ansible Script Generator Component	82
8.3.4.3 Script Executors.....	82
8.3.4.3.1 The Terraform Script Executor Component.....	83
8.3.4.3.2 The Ansible Script Executor Component	84
8.3.4.4 Performance Monitor	84
8.3.5 Best Design Practices Followed.....	84
8.4 Implementation of the KFactory Device Provisioner and Configurator	85
8.4.1 Development Platform and Environment.....	85
8.4.2 Python Modules Used.....	85
8.4.3 Implemented Application	86
8.4.3.1 The RootApp (Application's Open Screen).....	86
8.4.3.2 The Terraform Script Generator	87
8.4.3.3 The Ansible Script Generator	88
8.4.3.4 The Terraform Script Executor	89
8.4.3.5 The Ansible Script Executor	90

8.4.3.6	The Performance Monitor	90
8.4.3.7	The Other Small Screens Included in the RootApp (Application's Open Screen)	91
8.4.3.8	The Appearances in Windows and Linux Platforms.....	92
8.4.4	Best Practices Followed	94
8.5	Testing of the KFactory Device Provisioner and Configurator.....	94
8.5.1	Functional Testing	94
8.5.2	Non-functional Testing.....	105
8.6	Using the KFactory Device Provisioner and Configurator	107
8.6.1	Using the Terraform Script Generator	107
8.6.2	Using the Ansible Script Generator.....	107
8.6.3	Using the Terraform Script Executor	108
8.6.4	Using the Ansible Script Executor	109
8.6.5	Using the Performance Monitor	109
8.7	Deployment of the KFactory Device Provisioner and Configurator.....	111
8.8	Running the Deployed KFactory Device Provisioner and Configurator	111
8.8.1	Requirements to Run the Executables.....	111
8.8.2	Running the Executables.....	111
8.9	Limitations of the KFactory Device Provisioner and Configurator.....	112
Chapter 9	Conclusion and Future Work.....	113
9.1	Conclusion	113
9.2	Future Work	114
9.2.1	Development of the ML Model.....	114
9.2.2	Development of the Docker Image and Mac OS Executables	115
9.2.3	Adding the Support for Other IaC Tools.....	115
References		116
Appendix I	The Complete Experimentation Plan for Evaluate Terraform and Ansible	120
Appendix II	The Steps that were Followed when Performing Experiments on the Prototype Application with Terraform.....	124
Appendix III	The Steps that were Followed when Performing Experiments on the Prototype Application with Ansible	126
Appendix IV	The All Collected Experiment Results of Using the Terraform in Provisioning and Configuring IoT Devices in KFactory	127
Appendix V	The All Collected Experiment Results of Using the Terraform in Reconfiguring IoT Devices in KFactory	129
Appendix VI	The All Collected Experiment Results of Using the Ansible in Provisioning and Configuring IoT Devices in KFactory	131

Appendix VII The All Collected Experiment Results of Using the Ansible in Reconfiguring IoT Devices in KFactory	133
---	-----

List of Figures

Figure 1.1. A simple IoT system.....	1
Figure 1.2 The role of the respondent.....	4
Figure 1.3 The IaC familiarity of the respondent.....	4
Figure 1.4 The scale of the use of IaC of respondents.....	5
Figure 1.5 Rating for recommending IaC solutions for automatic provisioning and configuring IoT devices in the Smart Factory environment.....	5
Figure 1.6 Most preferred Model-Driven IaC tool for automatic provisioning and configuring IoT devices in the Smart Factory environment.....	6
Figure 1.7 Most preferred Code Centric IaC tool for automatic provisioning and configuring IoT devices in the Smart Factory environment.....	6
Figure 1.8 The attributes most likely to be selected as comparison factors if implementing prototypes using a Model Driven IaC tool vs Code Centric IaC tool for automatic provisioning and configuring IoT devices in an Automated Factory environment.....	7
Figure 2.1 The summary of the challenges, design principles and main enablers.....	9
Figure 2.2 The architecture of the prototype developed for an agricultural scenario.....	10
Figure 2.3 The infrastructure overview of LEONORE.....	13
Figure 2.4 The architecture of ARGON.....	14
Figure 2.5 Industry insights of different IaC tools.....	15
Figure 2.6 The 12 source code properties with the measurement technique.....	16
Figure 2.7 The system architecture for the automatic provision and configuring of IoT devices in a smart environment.....	18
Figure 4.1 The simplified methodology that the research followed.....	27
Figure 5.1 The action flow is followed for the sub-phase: Plan the implementation for the evaluation....	34
Figure 5.2 The planned system for the evaluation.....	34
Figure 5.3 The action flow is followed for the sub-phase: Preparation of the IoT project development environment.....	36
Figure 5.4 The action flow is followed for the sub-phase: Preparation of the simulation environment for targetted ESP32 devices, dashboards and MQTT brokers.....	37
Figure 5.5 The action flow is followed for the sub-phase: The action flow is followed for the sub-phase: Plan the evaluations.....	38
Figure 5.6 The action flow is followed for the sub-phase: Design, implementation and testing iterations to come up with an optimal prototype.....	40

Figure 5.7 The pure code-based ESP32 simulation system.....	41
Figure 5.8. The use of Terraform to run the separate EMQX MQTT broker and Node-RED dashboard.....	41
Figure 5.9. The use of Ansible to run the separate EMQX MQTT broker and Node-RED dashboard.....	42
Figure 5.10 The use of Terraform to build and run the built-in example project of mluis/qemu ESP32 emulator, separate EMQX MQTT broker and Node-RED dashboard.....	43
Figure 5.11 The use of Terraform to run a sample project on mluis/qemu ESP32 emulator, separate EMQX MQTT broker and a Node-RED dashboard.....	44
Figure 5.12 The use of Terraform to run three distinct projects on mluis/qemu ESP32 emulator, a separate EMQX MQTT broker and a Node-RED dashboard.....	45
Figure 5.13 The use of Ansible to run a sample project on mluis/qemu ESP32 emulator, separate EMQX MQTT broker and a Node-RED dashboard.....	46
Figure 5.14 The use of Ansible to run three distinct projects on mluis/qemu ESP32 emulator, a separate EMQX MQTT broker and a Node-RED dashboard.....	47
Figure 5.15 The planned map for the Factory.....	48
Figure 5.16 The optimal prototype for evaluating Terraform.....	51
Figure 5.17 The optimal prototype for evaluating Ansible.....	52
Figure 7.1 The variation of the elapsed time in provisioning and configuring IoT devices in KFactory by using Terraform and Ansible.....	65
Figure 7.2 The variation of the CPU load average in provisioning and configuring IoT devices in KFactory by using Terraform and Ansible.....	66
Figure 7.3 The variation of the Memory gap in provisioning and configuring IoT devices in KFactory by using Terraform and Ansible.....	67
Figure 7.4 The variation of the elapsed time in reconfiguring IoT devices in KFactory by using Terraform and Ansible.....	68
Figure 7.5 The variation of the CPU load average in reconfiguring IoT devices in KFactory by using Terraform and Ansible.....	69
Figure 7.6 The variation of the Memory gap in reconfiguring IoT devices in KFactory by using Terraform and Ansible.....	70
Figure 7.7 The variation of the elapsed time in provisioning and configuring and the reconfiguring of IoT devices in KFactory by using Terraform.....	71
Figure 7.8 The variation of the CPU load average in provisioning and configuring and the reconfiguring of IoT devices in KFactory by using Terraform.....	72
Figure 7.9 The variation of the Memory gap in provisioning and configuring and the reconfiguring of IoT devices in KFactory by using Terraform.....	73
Figure 7.10 The variation of the elapsed time in provisioning and configuring and the reconfiguring of IoT devices in KFactory by using Ansible.....	74

Figure 7.11 The variation of the CPU load average in provisioning and configuring and the reconfiguring of IoT devices in KFactory by using Ansible.....	75
Figure 7.12 The variation of the Memory gap in provisioning and configuring and the reconfiguring of IoT devices in KFactory by using Ansible.....	76
Figure 8.1 The use case diagram for KFactory Device Provisioner and Configurator.....	78
Figure 8.2 The architecture of the KFactory Device Provisioner and Configurator.....	79
Figure 8.3 The component diagram for KFactory Device Provisioner and Configurator.....	79
Figure 8.4 The flow for the Root App component in KFactory Device Provisioner and Configurator.....	80
Figure 8.5 The flow for the Script Generator components in KFactory Device Provisioner and Configurator.....	81
Figure 8.6 The flow for the Script Executor components in KFactory Device Provisioner and Configurator.....	83
Figure 8.7 The RootApp screen of the KFactory Device Provisioner and Configurator.....	87
Figure 8.8 The Terraform Script Generator screen of the KFactory Device Provisioner and Configurator.....	88
Figure 8.9 The Ansible Script Generator screen of the KFactory Device Provisioner and Configurator.....	89
Figure 8.10 The Terraform Script Executor screen of the KFactory Device Provisioner and Configurator.....	89
Figure 8.11 The Ansible Script Executor screen of the KFactory Device Provisioner and Configurator.....	90
Figure 8.12 The Performance Monitor screen of the KFactory Device Provisioner and Configurator.....	91
Figure 8.13 The other small screens included in KFactory Device Provisioner and Configurator's RootApp screen.....	92
Figure 8.14 The appearance of the KFactory Device Provisioner and Configurator on Windows Platform.....	93
Figure 8.15 The appearance of the KFactory Device Provisioner and Configurator on Linux Platform.....	94
Figure 8.16 The result of using the Terraform Script Generator.....	107
Figure 8.17 The result of using the Ansible Script Generator.....	108
Figure 8.18 The result of using the Terraform Script Executor.....	108
Figure 8.19 The result of using the Ansible Script Executor.....	109
Figure 8.20 The result of using the Performance Monitor while on a provisioning by Terraform.....	110
Figure 8.21 The result of using the Performance Monitor while on a provisioning by Ansible.....	110

Figure 9.1 The action flow is going to be followed for the creation of the ML model to predict and show the expected provisioning time, reconfiguring time, CPU, Memory and Network inbound and outbound usages according to the scale of the provisioning tasks based on the host system's capabilities.....115

List of Tables

Table 3.1 Comparison between the most popular and widely used model-driven IaC tools.....	23
Table 3.2 Comparison between the most popular and widely used model-driven IaC tools.....	24
Table 5.1 The evaluation plan	38
Table 5.2 The execution of the experimentations.....	54
Table 6.1 The results of using Terraform in provisioning and configuring	57
Table 6.2 The results of using Terraform in reconfiguring	59
Table 6.3 The results of using Ansible in provisioning and configuring.....	60
Table 6.4 The results of using Ansible in reconfiguring	61
Table 8.1 The Python modules used for KFactory Device Provisioner and Configurator development with their usage	85
Table 8.2 The functional tests performed on KFactory Device Provisioner and Configurator	94
Table 8.3 The non-functional tests performed on KFactory Device Provisioner and Configurator	105
Table Appendix I.1 The experimentation plan.....	120
Table Appendix IV.1 The all collected results of using the Terraform in provisioning and configuring IoT devices in KFactory	127
Table Appendix V.1 The all collected results of using the Terraform in reconfiguring IoT devices in KFactory	129
Table Appendix VI.1 The all collected results of using the Ansible in provisioning and configuring IoT devices in KFactory	131
Table Appendix VII.1 The all collected results of using the Ansible in reconfiguring IoT devices in KFactory	133