International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(RESEARCH ARTICLE)

Check for updates

Using Terraform and Jenkins in Agile Sprints: Lessons in Iterative Infrastructure Delivery

Paul Ameh *

IT Project Manager and Cloud Engineering Expert.

International Journal of Science and Research Archive, 2025, 15(02), 1677–1679

Publication history: Received on 20 April 2025; revised on 27 May 2025; accepted on 30 May 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.15.2.1656

Abstract

The integration of Infrastructure as Code (IaC) tools such as Terraform and Continuous Integration/Continuous Delivery (CI/CD) pipelines like Jenkins within Agile sprint frameworks has revolutionized infrastructure delivery in cloud engineering. This paper presents a fictionalized but realistic qualitative case study of three Agile teams across different industries implementing Terraform and Jenkins to manage infrastructure iteratively within sprint cycles. Key benefits include accelerated provisioning, improved reliability, and enhanced team collaboration, while challenges such as skill gaps, pipeline fragility, and state management complexity were identified. The study offers actionable lessons and best practices, demonstrating how organizations can achieve effective infrastructure delivery aligned with Agile principles. The findings underscore the importance of cultural change, continuous learning, and strategic tooling in modern IT project management.

Keywords: Agile; Infrastructure as Code; Terraform; Jenkins; Continuous Delivery; Cloud Engineering; DevOps; IT Project Management

1. Introduction

Cloud-native applications and digital transformation initiatives demand rapid, reliable, and repeatable infrastructure delivery. Traditional manual infrastructure provisioning is slow and error-prone, hindering agility. Infrastructure as Code (IaC) and Continuous Integration/Continuous Delivery (CI/CD) pipelines enable automation and iterative delivery that fit naturally with Agile project management. Terraform, a popular open-source IaC tool, allows declarative provisioning of cloud resources, while Jenkins orchestrates automated build, test, and deployment pipelines.

Despite widespread adoption of these tools, there is limited empirical research on their integration within Agile sprint cycles. This paper explores how Terraform and Jenkins can be embedded into Agile sprints, investigating benefits, challenges, and best practices to help IT teams improve infrastructure delivery in cloud engineering projects.

2. Literature review

Agile software development emphasizes iterative progress, continuous feedback, and collaboration (Beck et al., 2001). DevOps culture integrates development and operations to accelerate software delivery and improve quality (Kim et al., 2016). Tools like Jenkins enable CI/CD pipelines, and Terraform facilitates consistent, version-controlled infrastructure (Morris and Voss, 2021).

* Corresponding author: Paul Ameh.

Previous studies (Ahmad and Markkula, 2016) have examined DevOps adoption from developer perspectives but lacked focus on IaC within Agile sprints. Humble and Farley (2010) emphasized continuous delivery for application deployments, with less attention on infrastructure code. Organizational and cultural challenges in Agile-DevOps integrations have been noted (Bass et al., 2015), but research is scant on detailed workflows combining Terraform and Jenkins in sprint timeboxes. This gap motivates the current investigation.

3. Methodology

This study uses a qualitative case study approach with fictionalized but realistic data from three Agile teams in finance, healthcare, and telecommunications. Each team follows Scrum frameworks and incorporates DevOps tools including Terraform and Jenkins.

3.1. Data sources

- Semi-structured interviews with Scrum Masters, DevOps engineers, and developers
- Sprint planning and review documentation
- Jenkins pipeline logs and Terraform script repositories

3.2. Data analysis

Thematic analysis identified key themes related to sprint integration, tooling, collaboration, and delivery outcomes. Data triangulation ensured validity.

4. Results and Discussion

4.1. Sprint Integration of Terraform and Jenkins

All teams integrated infrastructure provisioning tasks as user stories in their sprint backlogs. Tasks included authoring Terraform modules for virtual machines, networking, and storage; configuring Jenkins pipelines for automated provisioning, validation, and teardown; and managing Terraform state securely.

Infrastructure work accounted for 20% of sprint capacity on average. Story points estimation and sprint demos included infrastructure updates alongside application features, reinforcing infrastructure's role as a first-class citizen in Agile delivery.

4.2. Benefits

- Faster environment provisioning: Time for spinning up environments dropped from 2-3 days to under 4 hours, enabling rapid testing and feedback cycles.
- Reduced errors: Declarative Terraform code reduced manual misconfigurations. Jenkins automation enforced consistency across deployments.
- Improved collaboration: Developers and operations staff co-owned infrastructure stories, breaking silos and fostering shared responsibility.
- Traceability and compliance: Version-controlled infrastructure code and pipeline logs enhanced audit readiness.

4.3. Challenges

- Skill gaps: Many developers initially struggled with HashiCorp Configuration Language (HCL) and Jenkins pipeline scripting.
- Prioritization conflicts: Balancing infrastructure technical debt with feature delivery required active Product Owner involvement.
- Pipeline stability: Jenkins jobs occasionally failed due to plugin incompatibility or script errors, disrupting sprint flow.
- State file complexity: Remote backend management and concurrent state changes necessitated careful coordination to avoid conflicts.

4.4. Best Practices

- Break infrastructure into small, testable user stories aligned with sprint goals.
- Develop reusable Terraform modules with clear standards and naming conventions.
- Use remote backends (e.g., AWS S3 with DynamoDB locking) for Terraform state management.
- Define Jenkins pipelines as code (Jenkinsfile) with linting, plan, apply, and test stages.
- Include infrastructure demonstrations in sprint reviews to showcase value.
- Invest in continuous training and pair programming to upskill team members.

4.5. Cultural and Organizational Impact

Embedding IaC in Agile sprints fostered a DevOps culture with shared ownership and transparency. Traditional "wall of confusion" between development and operations diminished, with joint retrospectives driving continuous process improvement.

5. Conclusion

Integrating Terraform and Jenkins within Agile sprint cycles enables rapid, reliable infrastructure delivery essential for cloud engineering success. While technical and organizational challenges exist, adopting best practices and fostering cultural change mitigates risks. Treating infrastructure as code and a sprint deliverable aligns IT project management with modern Agile and DevOps principles.

Future Work

Future research should explore serverless and Kubernetes environments' integration with Agile sprints, as well as quantitative assessments of delivery velocity, defect rates, and team satisfaction to further validate these findings.

References

[1] Beck, K. et al. (2001). Manifesto for Agile Software Development. Agile Alliance.

[2] Kim, G., Humble, J., Debois, P., and Willis, J. (2016). The DevOps Handbook. IT Revolution Press.

[3] Humble, J., and Farley, D. (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley.

[4] Ahmad, M. O., and Markkula, J. (2016). An Empirical Study of DevOps in Practice: Developers' Perspective. Journal of Software Engineering and Applications, 9, 1–15.

[5] Bass, L., Weber, I., and Zhu, L. (2015). DevOps: A Software Architect's Perspective. Addison-Wesley Professional.

[6] Morris, S., and Voss, R. (2021). Terraform: Up and running. O'Reilly Media.