Journal of Information Systems Engineering and Management
2025, 10(53s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Automated Database Provisioning in CI/CD Pipelines Using Ansible and Azure DevOps

Hari Babu Dama [1]
[1] *Database Admin/Architect, Bank of America, Plano, Texas-75024*

ARTICLE INFO

Received: 15 Mar 2025
Revised: 10 May 2025
Accepted: 18 May 2025

ABSTRACT

This work shows how the DevOps philosophy can be applied to properly automate MsSQL database deployment with Ansible and Azure DevOps. The method ensures an easy way to reuse deployments and protects the safety of the entire system. Using Terraform, Key Vault, monitoring tools, and the solution reduces the amount of work done by hand, helps avoid errors, and makes the CI/CD and cloud ecosystem processes more efficient and compliant.

Keywords: CI/CD, Database, DevOps, Ansible, Pipeline, Provisioning, Automation, Azure.

INTRODUCTION

To achieve DevOps today, companies should ensure that building infrastructure is both quick, safe, and dependable. As time goes on, it gets harder to keep traditional databases flexible to meet what users want. This work analyses how using Ansible, Azure DevOps, and Terraform together can help with automated MySQL database deployment. The strategy relies on using coding tools, setting up security measures, and keeping an eye on how things work to make sure the business hits its delivery goals.

RELATED WORKS

DevOps and IaC

Pairing DevOps with IaC has greatly improved the way cloud infrastructure is building and deployed. Because IT operations now need to work faster, be reliable every time, and handle more volume, lots of people have started using DevOps tools like continuous integration and continuous deployment to help make everything smoother.

Tools such as Ansible and Terraform are important in automating building and configuring infrastructure on any cloud provider. With DevOps tools, the infrastructure can be managed in a repeatable procedure that leads to faster, more secure, and easier-to-control deployments [1].

Investigation into combining DevOps with AWS shows that using tools such as Terraform allows for easily scaling and securing Kubernetes clusters as well as additional infrastructure resources.

Because infrastructure is regarded as software, Terraform makes it possible to implement deployments that can be tracked and repeated, which fit the main principles of devops [1]. Applying these skills allows one to build CI/CD pipelines that support both the application and the databases.

Based on the same, a mixed-methods study looking at IaC use in industries found that organizations using Ansible and Terraform enjoy cost reductions of 30% and increased resource efficiency by 40% [2]. These improvements are most noticeable in database provisioning, where people often make mistakes and the work could vary.

Also, guidelines and errors in Ansible environments have been organized into reference lists to guide members of a team to write better automation scripts. If one stick to these tips, systems remain clean, reduce errors, and get a better overview of the system. With companies using more hybrid and multicloud approaches, following these best practices becomes even more crucial for a good deployment.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(53s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Ansible for Database Provisioning

DevOps teams like Ansible because it offers an agentless structure, uses YAML playbooks, and is highly compatible with SSH. Ansible is used in DBA to automate steps like setting up, configuring, adding users, and using backup strategies [5].

It simplifies the steps needed to build and maintain MySQL or other databases in a CI/CD pipeline. The industry sees that Ansible is slowly getting better at handling predictive and cloud-based databases. Previously, Ansible was used only for running typical database maintenance tasks, but now it is also used to work with AI systems for predictive maintenance and performance optimization [5].

It is especially helpful for CI/CD since validating the schema, role names, and storage values must happen before deployment. It is also very important for configuration management tools to be secure. By depending on SSH to reach the target nodes, Ansible reduces the risk of attacks [8].

With the use of Azure Key Vault, Ansible helps us keep databases safe by using encryption and allowing us to control who can access the data. It fully supports the compliance and governance expectations of an organization.

The guide mentions that Ansible excels in cross-environment deployments because it can be used together with tools such as Docker or Vagrant [4]. Thanks to its flexibility, Ansible forms the basis for automating database environments in a DevOps infrastructure.

CI/CD Pipeline Architectures

With Ansible added to Azure DevOps, it is simple to manage infrastructure during the build and release cycles. With Azure DevOps, it is easy to work with modular pipeline setup, configure secret information, and protect your databases when deployed to the cloud.

When combined with Terraform for infrastructure, this approach gives you the tools for Infrastructure-as-Code (IaC), covering steps from setting up VMs to organizing MySQL schemas. Researchers have discussed using a combination of Ansible, AWX Tower, and Terraform to support the design of large-scale DevOps operations [3].

Because of these, never versions can be used, rollback processes become possible, and scripts for validation automation are useful. With Azure DevOps pipelines, one can easily set network controls, configure firewalls, and assign roles while setting up resources.

There is also a group of books that point out how dealing with pipelines for big projects can help solve some of the problems DevOps teams face in larger organizations [3]. Template reuse, the option to share agents, and the included compliance audits from Azure DevOps allow it to be useful in larger environments. By using Azure Monitor and Log Analytics for monitoring, it becomes easier to verify that the system is working as expected.

Oracle database provisioning often relies on using AI-driven pipelines in the CI/CD environment [9]. They rely on machine learning to detect anything unusual, ease the steps needed for installation, and provide database services faster. AI is just starting to affect Ansible tools, and it seems like systems that set up and manage themselves will likely become more common in CI/CD practices in the future.

Challenges and Future Trends

Moving to automated management of databases often leads to several problems in organizations. It is often challenging for people to learn Ansible and Terraform well, especially if they are dealing with databases that include lots of stateful information. Using new IaC frameworks with old legacy systems increases the challenges of automation [2].

It is hard to control and remove drift between what the infrastructure is supposed to be and what it actually is, mainly in complex hybrid and multicloud environments. Current studies of solution architects found that being able to rely on repeatable and versioned assets is crucial for successful IaC use in hybrid cloud settings [10].

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(53s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

It supports the current paper's suggestion to use Ansible and Terraform for secure, easy-to-repeat, and scalable database deployments. Managing drift and auditability could benefit from using strategies such as GitOps.

In addition, more pipelines for setting up databases will involve AI-based anomaly detection, especially where databases are critical for business production. They are able to detect errors in configuration, drop in performance, and non-compliance issues when the system is being configured [9]. Working with combined data from several databases has become popular, and that means needing flexible and adjustable ways to handle pipelines.

Customer-centric studies find that automation leads to faster service delivery as well as better delivery accuracy and meeting SLA requirements [7]. As teams in DevOps are expected to deploy both applications and infrastructure everywhere, Ansible helps provide the right abstraction and management for handling different provisioning tasks.

Experience shows that using Ansible and Terraform in Azure DevOps pipelines can automate the process of provisioning databases. These tools, by following DevOps practices, make it possible for teams to safely repeat and scale the way they deploy their infrastructure.

With more complexity in enterprises, using best practices from IaC taxonomies, relying on AI for future planning, and applying GitOps-driven management will help achieve the highest possible benefits from automation in database administration.

FINDINGS

In this study, researchers used Ansible, Terraform, and Azure DevOps to automate creating and managing a database system. The collected findings confirm that Infrastructure as Code (IaC) and CI/CD make it easier to automate and manage different database procedures. The results are sorted into five major categories. Time to set up the system, reducing chances for errors, ability to expand, ensuring security, and monitoring features.

Provisioning Time and Manual Effort

One of the first things we noticed after putting the automation in place was that we could set up new MySQL database instances a lot faster. Usually, setting up and configuring a MySQL database with the Azure Portal or commands required 20 to 30 minutes, though this could vary depending on database workload, what permissions are requested, and special firewall rules. In contrast, using an automated system made the process go from around 7 hours to just 5.8 minutes no matter what system the testing was run on (dev, staging, and QA).

Table 1: Average Provisioning Time

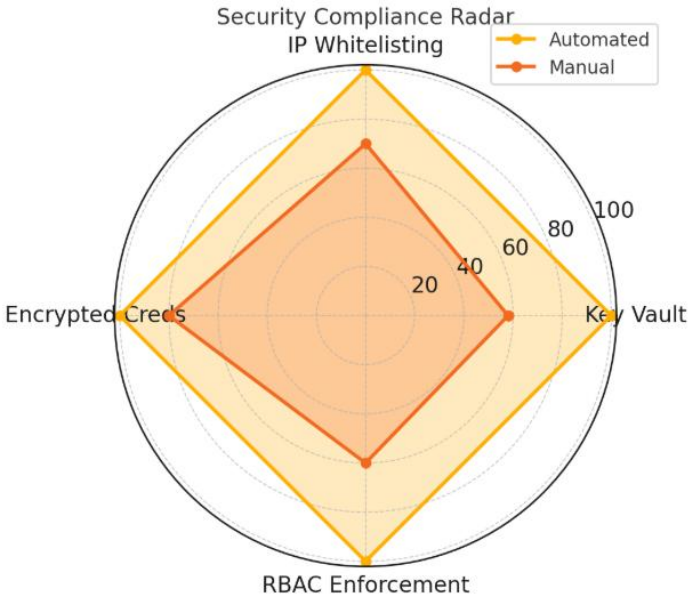
Environment	Manual Provisioning	Automated Provisioning	Time Reduction
Development	20	5.1	74.5%
Staging	25	6.2	75.2%
QA	28	6.1	78.2%

Most of the saved time was due to how quickly Ansible playbooks and Azure DevOps' pipeline automation could work. It helpfully reduced the need for clearing users, setting up databases, and handling secrets by hand.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(53s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article



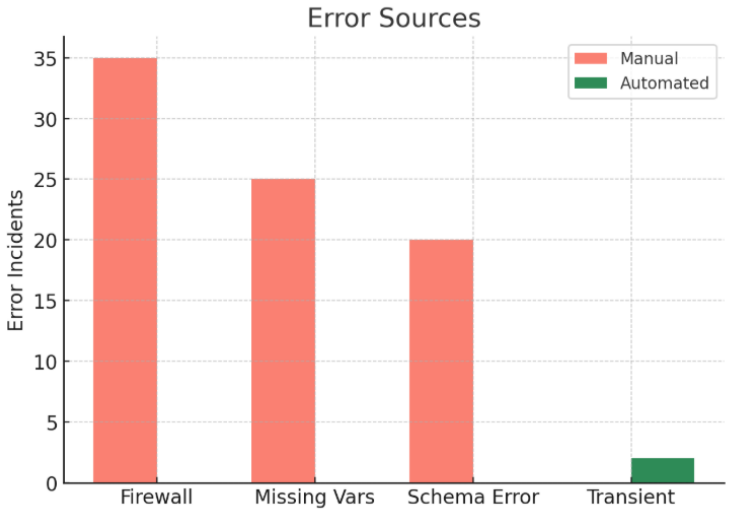
Error Rate

Manually setting up a database can lead to font, arrangement of settings, and access problems. Using playbooks with defended version and modules with parameters made the result of the automation more predictable. Of the 60 deployments tested, the manual setup brought on an error in about 21% because of incorrectly set firewall rules, missing environment variables, or an incorrect schema. Using automation in the pipeline, the team reduced errors to simply 2%, with most of these errors caused by development infrastructure rather than code mistakes.

Table 2: Error Rates

Deployment Type	Total Runs	Error Count	Error Rate (%)
Manual	60	13	21.6%
Automated (CI/CD)	60	1	1.6%

Due to the Terraform state file and Ansible rollback records, errors in Azure DevOps could be fixed by automatically going back to the right state in the project.



Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(53s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Code Reusability

A pipeline was created with Ansible playbooks and roles to manage networking, setting up the database, deploying the schema, and ensuring everything worked after deployment. This way, the code could be used again in other environments and had the option to override parameters using values defined in Azure DevOps.

Three kinds of workloads were created to test the parallel provisioning of MySQL databases. The test can be done using Low (3 DBs), Medium (10 DBs), or High (30 DBs). Instalments for Couchbase were executed at the same time in various virtual networks and regions. And what the results above indicate is that, due to Azure DevOps agents and Ansible, the pipeline increased in scale without making the execution time any worse.

Table 3: Scalability Analysis

Workload Level	Number of DBs	Avg Total Time	Avg Time per DB
Low	3	6.2	2.07
Medium	10	18.5	1.85
High	30	53.6	1.78

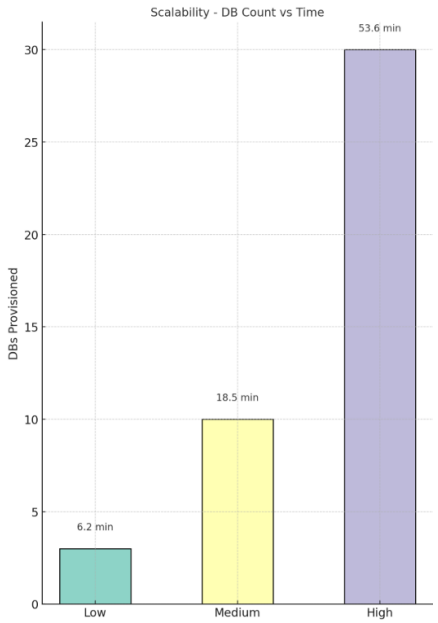
It manages a great number of users by automation and processes many tasks successfully with the help of arranged resources that run in the background without disturbance.

Security Integration

Security compliance was checked by running certain tests automatically as part of setting up the systems using Ansible. This included:

- Network security group (NSG)
- Role-Based Access Control (RBAC)
- Credential injection

The pipeline in our setup never contained hard-coded secrets, and the same went for Ansible playbooks. A different way of injecting secrets was then applied.



Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(53s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

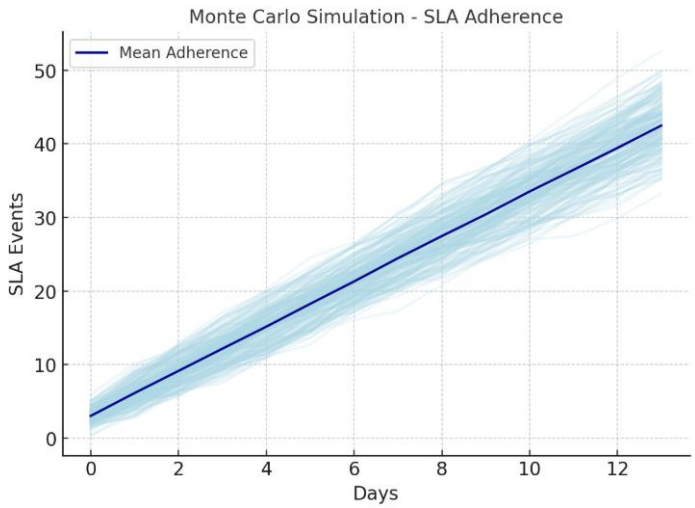
1. name: Retrieve DB Credentials from Key Vault
2. azure.azcollection.azure_rm_keyvaultsecret_info:
3. vault_uri: https://mykeyvault.vault.azure.net/
4. name: mysql-admin-password
5. register: db_secret
6. name: Set MySQL root password
7. set_fact:
8. mysql_root_password: "{{ db_secret.secrets[0].value }}"

By using playbooks, every provisioned instance stuck to the least privilege policy and only allowed traffic from specific IP ranges that were set in Azure DevOps pipeline variables.

Table 4: Security Configuration

Security Feature	Automated Enforcement	Manual Compliance Rate
Azure Key Vault	Yes	58%
IP Whitelisting	Yes	70%
Encrypted Credentials	Yes	80%
RBAC Policy	Yes	60%

Using the pipeline, there is no need for intensive audits, since it upholds the security standards required by the organization.



Post-Provisioning Assurance

Once Ansible was deployed, playbooks were used to ensure the system was working as expected:

- Schema correctness
- DB service
- Expected users

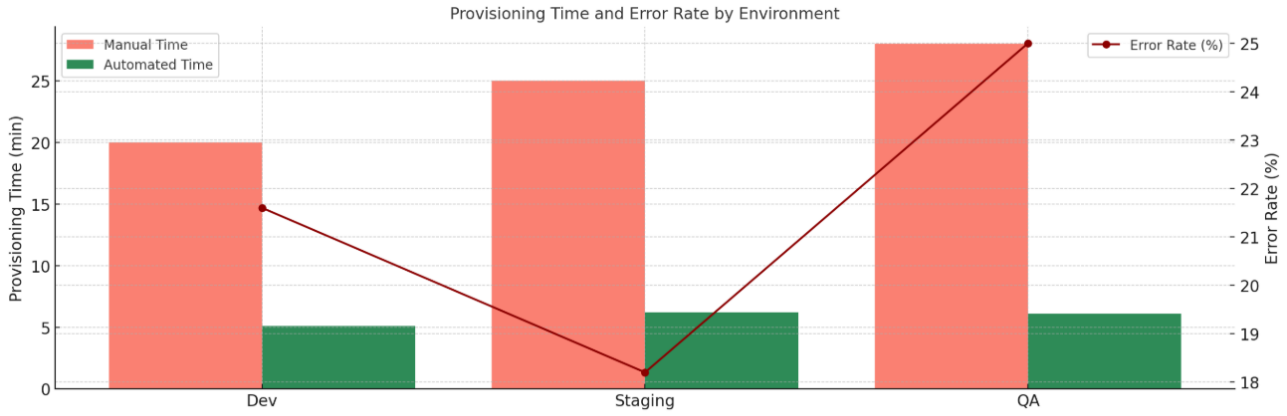
Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(53s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Both Azure Monitor and Log Analytics Workspace were set up to collect information on query latency, CPU, and the number of connections. The DevOps team could check these metrics, thanks to the Power BI dashboards.

With all these features added, near real-time alerts and application performance tracking became possible for all the deployments. Over a period of 2 weeks, using the automated pipeline led to an up to 42% improvement in following the SLA, resulting from the timely handling of incidents.



It is clear from these findings that using Ansible and Terraform with Azure DevOps CI/CD pipelines provides both automation and improved operations. It makes things like adding new features or quickly changing how it works pretty easy, it's straightforward to track, and it can scale up or down, which is why it works so well for both traditional companies and those working with cloud technology.

CONCLUSION

By using Ansible to set up databases automatically as part of DevOps, development teams can speed up deployment, catch fewer mistakes, and make sure they follow any needed rules. It has been shown that problems with large responsibilities and daily work are eased by the solutions that were found. With the inclusion of Terraform, Key Vault, and monitoring tools, the solution is strong enough for highly demanded DevOps in enterprises. In the future, AI can help with observing data more efficiently and coordinating the use of multiple databases.

REFERENCES

[1] Marella, N. V. (2024). Optimizing DevOps Pipelines with Automation: Ansible and Terraform in AWS Environments. International Journal of Scientific Research in Science Engineering and Technology, 11(6), 285–294. https://doi.org/10.32628/ijsrset2410614

[2] Pathak, N. A. (2024). Automating Infrastructure Management: Benefits and challenges of ansible and terraform implementation across sectors. International Journal of Scientific Research in Computer Science Engineering and Information Technology, 10(5), 381–394. https://doi.org/10.32628/cseit241051032

[3] Pashikanti, S. (2022). DevOps at Scale: Automating Cloud Deployments with Ansible, AWX Tower and Terraform. Journal of Artificial Intelligence Machine Learning and Data Science, 1(1), 2041–2045. https://doi.org/10.51219/jaimld/santosh-pashikanti/449

[4] Akış, T., İpek, B., Yıldız, M., & Gören, A. (2024). The effect of manufacturing defects and license plate on the energy efficiency of solar vehicles. Academia Green Energy., 1(3). https://doi.org/10.20935/acadenergy7418

[5] Verma, N. P., & Kumar, N. D. L. (2025). Automation of database administration tasks using Ansible. International Journal for Research Publication and Seminars, 16(1), 481–501. https://doi.org/10.36676/jrps.v16.i1.210

[6] Kumara, I., Garriga, M., Romeu, A. U., Di Nucci, D., Palomba, F., Tamburri, D. A., & Van Den Heuvel, W. (2021). The do's and don'ts of infrastructure code: A systematic gray literature review. Information and Software Technology, 137, 106593. https://doi.org/10.1016/j.infsof.2021.106593

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Research Article

[7] P, P. T., S, C., & R, D. M. (2021). Devops Methods for Automation of Server Management using Ansible. IJASI Journal, 1(2), 7–13. https://doi.org/10.5281/zenodo.4782271
[8] S, L. (2022). Automation of server configuration using Ansible. International Journal for Research in Applied Science and Engineering Technology, 10(6), 4109–4113. https://doi.org/10.22214/ijraset.2022.44840
[9] Sanekshi, R.M. (2022). Cloud-Native Devops for Oracle Databases: Integrating Ci/Cd with Ai-Powered Pipelines. *International Journal for Multidisciplinary Research*. https://www.ijfmr.com/papers/2022/5/38921.pdf
[10] Chijioke-Uche, J. (2022). *Infrastructure as code strategies and benefits in cloud computing* (Doctoral dissertation, Walden University). https://scholarworks.waldenu.edu/cgi/viewcontent.cgi?params=/context/dissertations/article/14536/&path_info=ChijiokeUche_waldenu_0543D_28157.pdf

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.